Supported by:

Federal Ministry
for Economic Affairs
and Energy

on the basis of a decision
by the German Bundestag

Speeding up Energy System Models - a Best Practice Guide

Yvonne Scholz[1], Benjamin Fuchs[1], Frieder Borggrefe[1], Karl-Kien Cao[1], Manuel Wetzel[1], Kai von Krbek[1], Felix Cebulla[1], Hans Christian Gils[1], Frederik Fiand[2], Michael Bussieck[2], Thorsten Koch[3], Daniel Rehfeldt[4], Ambros Gleixner[4], Dmitry Khabi[5], Thomas Breuer[6], Daniel Rohe[6], Hannes Hobbie[7], David Schönheit[7], Hasan Ümitcan Yilmaz[8], Evangelos Panos[9], Samir Jeddi[10], and Stefanie Buchholz[11]

[1]German Aerospace Center
[2]GAMS Software GmbH
[3]Technische Universität Berlin
[4]Zuse Institute Berlin
[5]High Performance Computing Center Stuttgart
[6]Juelich Research Centre
[7]Technische Universität Dresden
[8]Karlsruhe Institute of Technology
[9]Paul Scherrer Institute
[10]University of Cologne
[11]Technical University of Denmark

June 30, 2020

BEAM-ME

HLRS GAMS JÜLICH Forschungszentrum DLR Deutsches Zentrum für Luft- und Raumfahrt German Aerospace Center ZIB

Contents

Acronyms and Abbreviations

ASIC	Application-Specific Integrated Circuit
API	Application Programming Interface
AVX	Advanced Vector Extensions
BEAM-ME	Realisierung von Beschleunigungsstrategien der Anwendungsorientierten Mathematik und Informatik fuer Optimierende Energiesystemmodelle
CA	Local Cache Agent
CHP	Combined Heat and Power
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
DIMM	Dual Inline Memory Module
DLR	German Aerospace Center
DTU	Danish Technical University
DVFS	Dynamic Voltage and Frequency Scaling
ECM	Execution-Cache-Memory
ED	Economic Dispatch
ESM	Energy System Model
EWI	Institute of Energy Economics at the University of Cologne
EWL	Lehrstuhl für Energiewirtschaft
FLOP	Floating Point Operation
FLOPS	Floating Point Operations per Second
FMA	Fused-Multiply-Add instruction
FPGA	Field Programmable Gate Arrays
GAMS	General Algebraic Modeling System
GDX	GAMS Data eXchange file format
GEP	Generation Expansion Planning
GPFS	General Parallel File-System
GPU	Graphics Processing Unit
HA	Home Agent
HLRS	Hoechstleistungsrechenzentrum Stuttgart (High Performance Computing Center Stuttgart)
HPC	High Performance Computing
HPCG	High Performance Conjugate Gradients Benchmark
HPL	High-performance linpack benchmark
HSM	Hierarchical Storage Management
IC	Interconnect Connections

ILP	Instruction Level Parallelism
IMC	Internal Memory Controller
IPM	Interior Point Method
I/O	Input and Output
JSC	Juelich Supercomputing Centre
JUST	Juelich Storage Cluster
KIT	KITKarlsruhe Institute of Technology
LP	Linear Program
LRZ	Leibniz Rechenzentrum
MG	Multi-Grid
MIMD	Multiple Instruction Multiple Data
MIP	Mixed Integer Program
MILP	Mixed Integer Linear Program
MINLP	Mixed Integer Non-Linear Program
MPI	Message Passing Interface
MOS	Metal-Oxide-Semiconductor
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NLP	Non Linear Program
NUMA	Non-Uniform Memory Access
ESOM	Energy System Optimization Model
OMP	Open Multi-Processing
OP	Optimization Problem
OPF	Optimal Power Flow
OS	Operating System
PBS	Portable Batch System
PIPS	Parallel solvers for optimization problems
PSR	Paul Scherrer Institute
QPI	QuickPath Interconnect
SCIP	Solving Constraint Integer Programs
SDRAM	Synchronous Dynamic Random Access Memory
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SMT	Simultaneous Multithreading
SRAM	Static Random Access Memory
SSH	Secure Shell

TEP	Transmission Expansion Planning
TUB	Technische Universität Berlin
TUD	Technische Universität Dresden
UC	Unit Commitment
VPN	Virtual Private Network
ZIB	Zuse Istitute Berlin

Glossary

Bandwidth	Rate of data transfer, bit rate or throughput, measured in bits per second (bit/s)
Cluster	Set of highly networked computers, mostly actuated by a batch system
Defined variable	Variable that stores the result of a function (e.g. a sum) of other variables for further use.
Distributed memory	
Farming	Simple parallelization without interdependencies
FLOP	Number of floating point operations
FLOPS	Number of floating point operations per second
Model instance	A unique combination of model formulation and parametrisation
Label	GAMS jargon for the elements of one-dimensional sets, a fundamental data structure in GAMS (frequently referred to as "unique element" or "UEL").
Linking constraint	A constraint that links two or more variables (columns of the generated model) with each other, making independent solving of these impossible. A linking constraint can link variables e.g. temporally (e.g. regional hourly power generation using a fuel resource that is limited) or regionally (e.g. regional generation by different technologies if a minimum hourly domestic generation is set). In a decomposed model instance, a linking constraint refers to blocks instead of single variables.
Linking variable	A variable that appears in two or more equations (lines of the generated model) and thus precludes independent solving of these. A variable can be linking e.g. temporally (e.g. fill level of a storage unit) or spatially (e.g. transmission between regions). In a decomposed model instance, a linking variable refers to blocks instead of single equations. A variable in a decomposed model instance is "strongly" linking when it links multiple or all blocks and "weakly" linking when it connects just two neighboring blocks.
Load balancing (IT)	Distribution of sub-problems in parallel computing
Load balancing (ESM)	Balancing of residual electric load in power supply systems
Message Passing Interface	The standard for message passing in parallel computing
Model generation	GAMS jargon for an initial step in processing a solve statement, where a model instance is generated (based on the equation definitions and the data referenced by those definitions) for the solver
Node (IT)	Server in a cluster
Node (ESM)	Representative of one region
OpenMP	Interface for shared memory programming
PARDISO	The package PARDISO is a thread-safe, high-performance, robust, memory efficient and easy to use software for solving large sparse symmetric and unsymmetric linear systems of equations on shared-memory and distributed-memory multiprocessors.
PIPS	Parallel solvers for optimization problems, developed by Argonne National Laboratory
Profiling	Analysis of computing times of single program modules
Residual load	Instantaneous power demand that remains after subtracting power generation in inflexible power plants
Scalability	Change of performance of a system proportional to its size.

SCIP	Solving Constraint Integer Programs, LP/MILP/MINLP/CIP solver developed by Zuse-Institut Berlin.
Server farm	Large amount of servers (with or without fast links)
Unique element	GAMS jargon for the elements of one-dimensional sets, a fundamental data structure in GAMS (frequently abbreviated UEL and often referred to as "label").
Vectorization	Vectorization is a parallel hardware-assisted paradigm, whose approach is automatically supported by the compiler.

Chapter 1

Introduction[1]

Deregulation and growing decentralization lead to an increasing complexity of energy systems. Given the envisaged creation of a common European energy market and the transformation of energy supply towards sectoral coupling and electricity generation from variable, renewable energy sources, this trend can be expected to continue.
In this context, new energy policies are often investigated with the help of linear optimization models (Baños et al., 2011). However, the increasing complexity of the system to be modelled results in energy system optimization models (ESOM) that reach their limits in terms of memory demand and reasonable computing time. For example, very complex models on single servers could achieve or exceed computing times of weeks to a month with memory consumptions of 100 GB and more. The mathematical optimization problems formulated in the models reached orders of magnitude that pushed the then most powerful solvers such as CPLEX, Gurobi, XPress, or SCIP to their limits. Existing and especially future research questions in the field of energy systems analysis could thus only be addressed to a limited extent. Many energy system models were then, as they are now, based on the General Algebraic Modeling System (GAMS) modeling software. Since models with high spatial and temporal resolution can only be parallelized trivially to a limited extent, the potential of High Performance Computers (HPC) could not be used without considerable effort. Accordingly, the solution of temporay and spatial high-resolution ESOM proved to be difficult. Furthermore, the methodological uncertainties from typically used reduction approaches such as temporal or spatial aggregation were only insufficiently known and investigated. In addition, the heterogeneity of applied strategies results in the fact that the comparability of model-based scenario studies is difficult and the trade-off between implementation costs and achievable performance is often unknown. Since the used models show similarities in essential characteristics (e. g. with regard to fundamental equations or applied solver software packages), it can be assumed that effective speed-up strategies for energy system models are transferable.
In order to address these technical problems in scientific work, the BEAM-ME project ("Realisierung von Beschleunigungsstrategien der Anwendungsorientierten Mathematik und Informatik fuer Optimierende Energiesystemmodelle", funded by the German Federal Ministry for Economic Affairs and Energy) has developed solutions to the following questions:

1. What are useful strategies and approaches for solving highly detailed ESOM?
2. Which requirements for modern energy system model formulations result from solver- and model-based acceleration methods?
3. How can the transfer of methods and approaches between ESOM and HPC be facilitated? Which

[1]This chapter is based on the preprint of "Cao, Karl-Kiên and von Krbek, Kai and Wetzel, Manuel and Cebulla, Felix and Schreck, Sebastian (2019): *Classification and Evaluation of Concepts for Improving the Performance of Applied Energy System Optimization Models*, DOI: 10.3390/en12244656; Journal: Energies; URL: https://www.mdpi.com/1996-1073/12/24/4656"

guidelines can be formulated?

4. How can methods and approaches be translated into universally applicable algorithms and how can they be made available to the scientific community?

5. What are the possible contributions of acceleration methods to future research questions in the field of energy system analysis?

At the beginning of the project, the basic principles were initially developed. On the technical side, an overview of HPC-enabled solvers available at that time was created with a special focus on open source availability. On the basis of this research, it was considered whether a completely new development or building on an existing solver should be pursued in the future. At the same time, a systematic overview of common model-based performance enhancement methods, e.g. by clever reduction of the model dimensions, was compiled. These were then examined in detail using the ESOM REMix developed at DLR.
A central element of the project was the Model Experiment (MEXT), whose aim was to ensure the transferability of the methods investigated in the project to other ESOMs. In three calls for proposals, six partner institutions were selected to provide their GAMS-based models. These models were used to demonstrate the transferability of the model-based performance enhancement methods over the course of the project.
This introductory section provides a first overview of ESOM characteristics (1.1), the modeling system GAMS (1.1.2), solution algoritms for linear optimization problems (1.2) and High Performance Computing (1.3). It also includes a description of the BEAM-ME model experiment (1.4).
Part I Modeling Based Performance Enhancement of this guide outlines speed-up methods on the modelling side. It comprises chapter 2 and chapter 3. Chapter 2 describes model based performance enhancement methods in the three model dimensions space, time and technology. Chapter 3 presents the evaluation of selected methods using the ESOM REMix.
Part II Technical Performance Enhancement, deals with technical performance enhancement, i.e. the combination of high performance computing and parallelized solving of linear ESOMs. This part contains chapters 4 and 5. Chapter 4 presents the components of the technical performance enhancement approach: The development of a new solver, high performance computing and the preparation of ESOMs to obtain a block structure of the coefficient matrix which separates independent blocks from linking equations and variables. The performance enhancement that can be achieved by this technical approach is evaluated in chapter 4.
Part III Results of the Model Experiment describes the benchmark comparison between REMix and six additional models from other institutions in chapter 6. This chapter covers both model-based as well as technical performance enhancement methods applied in the model experiment part of BEAM-ME.
Part IV Conclusions summarizes the major insights from this best practice guide and lessons learned from the overall BEAM-ME project.
Part IV Appendices provides three chapters: Appendix A presents the simplified ESOM "SIMPLE", Appendix B provides an instruction how to run SIMPLE on the supercomputer Hazel Hen and Appendix C provides additional information about node-level HPC architecture.

1.1 Energy system optimization models: characteristics and dimensions

In the context of energy systems analysis a broad spectrum of research questions is addressed by ESOM to support decision making in both energy politics and energy industry. In particular, this concerns the development of future strategies such as energy scenarios for mitigation of climate change (Paltsev, 2016) or fundamental analyses of electricity markets (Ventosa, Baillo, Ramos, & Rivier, 2005) and investment planning by system operators (Kagiannas, Askounis, & Psarras, 2004), (Wu, Zheng, & Wen, 2006). Therefore, the objective of the associated optimization problems (OPs) is either the optimal operation or the optimal configuration of the analyzed system which consist of a diverse set of technologies. With regard to electricity generation, the former is originally known as Unit Commitment (UC) or Economic Dispatch (ED) problem (Zhu, 2015), while the latter is referred to as Generation Expansion Planning (GEP) (Oree, Hassen, &

Table 1.1: Characteristics of Energy System Optimizing Models

Dimension	Model characteristic	Descriptive characteristic	Example
			Short-term (sub-annual operation) / Long-term (configuration/ investment)
Time	Set of time steps	Temporal resolution	hourly / each 5 years
		Planning horizon	one year / from 2020 until 2050
Space	Set of regions	Spatial resolution	Administrative regions (e.g. NUTS3 (Eurostat, 2017))
		Geographical scope	European Union
Technology	Variables and constraints per technology	Technological detail	Consideration of start-up behavior, minimum downtimes
	Set of technologies	Technological diversity	Power and heat generation, transmission grids and storage facilities

Fleming, 2017). If these problems are resolved on the spatial scale, the consideration of transport infrastructures, such as high voltage transmission grids, and thus modeling of multi-area OPs becomes relevant. Typical examples are Optimal Power Flow (OPF) problems (Frank, Steponavice, & Rebennack, 2012) on the operational side and Transmission Expansion Planning (TEP) (Quintero, Zhang, Chakhchoukh, Vittal, & Heydt, 2014) on the configurational side.
Furthermore, due to the increasing relevance of renewable energy sources in todays and future energy systems, also the evaluation of strategies which make use of electricity storage facilities to integrate fluctuating power generation becomes more and more important (Haas et al., 2017).
The problems addressed by energy systems analysis are typically combinations of the above mentioned aspects which result in integrated Bottom-Up models that differentiate three major scales: technologies, time and space. Table 1.1 shows these scales together with their characteristics for exemplary applications. Two kinds of characteristics are distinguished here. While the descriptive characteristic is related to the description of the underlying real world problem, the model characteristic refers to the way how this problem is translated into a mathematical model formulation.
Depending on the application, the three dimensions are differently pronounced or resolved in energy system analysis. For example, on the one hand, ESOMs are strongly spatially resolved with the aim of cost-optimized network expansion planning by TEP. On the other hand, also the temporal resolution becomes important as soon as a study tries to capture the variability of power generation from renewable energy sources. However, formulating a mathematical model with these characteristics usually results in coupling of time, space and technology among each other. Even more importantly, the need of addressing flexibility demands in future energy systems (Hendrik Kondziella & Thomas Bruckner, 2016) also leads to couplings within these dimensions. In particular, these couplings are caused by temporally shifting of generation and consumption with storage facilities or demand side management measures which links discrete points in time, by power exchange over transmission grids that results in an interdependency of regions as well as by cross-sectoral technologies such as combined heat and power (CHP) plants.

1.1.1 Specific characteristics of different energy system model types

One substantial common characteristic of optimization models, we refer to as ESOMs, is the use of a cost-based objective function (1.1)

$$\text{Objective function: Minimize:} \sum_{t \in T} \sum_{n \in N} \sum_{u \in U} c_{t,n,u} p_{t,n,u} \tag{1.1}$$

in conjunction with a power balance equation (1.2)

$$\text{subject to:} \sum_{u \in U} p_{t,n,u} = d_{t,n} \quad \forall t \in T,\ \forall n \in N$$
$$p_{t,n,u} \geq 0 \tag{1.2}$$

where p is a variable of power supply, c are specific costs, d is power demand, T is the set of time steps, N the set of regions and U the set of technologies.

Although different ESOMs consist of a large variety of further constraints, such as capacity- activity, flow or security constraints, they share another similarity concerning the structure of the coefficient matrix A of the appropriate linear program (Figure 1.1): The above mentioned interdependencies of time, space and technologies translate either into linking variables or linking constraints. Both are characterized by the fact that they prevent the OP from being solved by solving independent sub-problems (indicated by the colored blocks in Figure 1.1). From an applied point of view, this means, for example, that for a selected time frame the dispatch of reservoir power plants cannot be determined without the information about the storage level. However, the storage level of the actual time frame also relies on the dispatch of previous points in time.

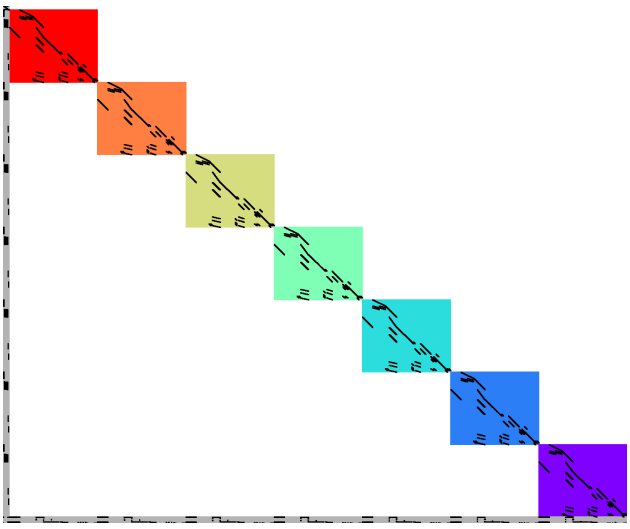


Figure 1.1: Non-zero entries (black dots) in an exemplary coefficient matrix A of an integrated ESOM with linking variables (grey area at the left), linking constraints (grey area at the bottom) and independent blocks (colored blocks).

In this context, variables that occur simultaneously in several equations are generally referred to as linking variables (or sometimes complicating variables). Provided that an appropriate permutation is given, as shown in Figure 1.1, linking variables appear as vertical lines of non-zero entries in the coefficient matrix. With regard to the temporal scale, representatives of linking variables in ESOMs are used when performing capacity expansion as the appropriate investment decision variables (e.g. opposed to activity variables) are not defined for each time step of the operational time horizon. This is illustrated by inequality 1.3

$$p_{t,n,u} \leq P_{n,u} + I_{n,u} \quad \forall t \in T,\ \forall n \in N,\ \forall u \in U \tag{1.3}$$

where P is existing capacity and I the variable of capacity expansion. Inequality 1.3 is defined for each time step t, and I stays the same for each t. In contrast to linking variables, horizontal lines of non-zero entries in the coefficient matrix indicate linking constraints (Figure 1.1), sometime referred to as complicating constraints. For example, fuel availability constraints, such as used for modeling biomass fired power plants,

typically define a temporally non-resolved value as an annual limit. To ensure that the total fuel consumption within the operation period stays within this limit, a linking constraint (equation 1.4) couples the involved variables:

$$\text{Fuel-availability constraint: } \sum_{t \in T} \sum_{u \in U_{bio}} p_{t,n,u} \leq \mu_u F_n \quad \forall n \in N \tag{1.4}$$

where F is the available fuel, μ is the conversion efficiency and U_{bio} the set of biomass power plants.

1.1.2 The Optimization Environment GAMS

Today, algebraic modeling languages such as the General Algebraic Modeling System (GAMS) are widely used to represent and solve mathematical programming problems. Their main distinguishing features are the use of relational algebra and the ability to provide partial derivatives on multidimensional, very large and sparse structures. The usage of data structures specifically tailored for the purpose of algebraic modeling make GAMS models easily readable for both, humans and by machines. To achieve its goals to improve the model builder's productivity, reduce costs, and improve reliability and overall credibility of the modeling process, GAMS established the following key principles:

- The problem representation is independent of the solution method.
- The data representation follows the relational data model.
- The problem and data representations are independent of computing platforms.
- The problem and data representations are independent of user interfaces.

The aforemantioned characteristcis make GAMS an efficient modeling tool for the field of energy system analysis where it is widely used.

1.2 Solution algorithms

Practically successful solving algorithms for linear programs (LPs) predominantly fall into one of the following two categories: Simplex and interior-point. For either of these methods a wealth of books and research articles exists, see for instance (Matousek & Gärtner, 2007) Introduction to Linear Optimization. Most state-of-the-art commercial linear solvers provide both Simplex and interior-point algorithms. This diversity does not come without reasoning: While for large-scale LPs the interior-point approach usually prevails, the Simplex algorithm is often faster for smaller problems and is moreover indispensable for the solution of mixed-integer programs (due to its powerful warm-start capabilities). Although they solve the same problem, Simplex and interior-point methods pursue fundamentally different approaches. To illustrate this behavior, the reader is reminded that the set of feasible solutions to a linear program can be mathematically represented as a *polyhedra*, a finite intersection of halfspaces. While the Simplex method moves along the surface of this polyhedra (or more precisely along its vertices) towards an optimal solution, interior-point methods start inside the polyhedra and make their way towards the surface (but unlike the Simplex method not in general towards a vertex). As already mentioned, the interior point algorithm has proven to usually be the better choice for tackling large-scale LPs. It also brings the advantage that the number of its iterations rarely surpasses 50—an empirical magic number with little theoretical foundation. In contrast, one cannot in general deduce the total required number of iterations of the Simplex algorithm from its current number of iterations. Furthermore, for linear programs that come with a block-diagonal structure and both linking constraints and linking variables (a structure often found in time-parameterized energy system models), the linear algebra within interior point methods can often be efficiently parallelized. This property allows to solve problems of a scale intractable for desktop machine on supercomputers. Consequently, for solving large-scale LPs derived from energy systems, interior-point algorithms are usually the preferred choice, be it for off-the-shelf solving as provided by commercial software packages or hand-tailored specialized algorithms (as for instance developed in the BEAM-ME project). Nevertheless, one certainly should test both interior-point and Simplex algorithms for each particular energy model whenever this is possible.

1.3 High performance computing

High-Performance Computing (HPC) provides the resources that allow scientists and engineers to solve complex problems, which couldn't be solved before at all or not within an acceptable time frame. The high performance of supercomputers comes from a lot of identical servers, so-called „compute nodes“, which use a high performance interconnect to synchronize their work. While there is no strict, commonly accepted definition of what constitutes a supercomputer, in practical terms, the top 500 of them mark out the boundary of HPC systems. The TOP500 list shows the known most powerful computer systems[2] around the world (TOP500.org, 2018). What is needed to be among them?

Above all, the system has to solve a dense n by n system of linear equations $A = bx$ with a certain aggregated performance, which is measured in units *Flops* (Floating-point operations per second). The TOP500 Benchmark HPL (High-performance linpack benchmar) (ICL, 2018) used for this purpose solves a random dense system of linear equations in double precision using LU decomposition with partial row pivoting. A supercomputer needs to perform $O(n^3)$ floating point operations to find the solution on that way. Due to the nature of the HPL, the *Byte/Flop* ratio is small $\approx \frac{8 \times n^2[Byte]}{n^3[Flop]}$ and it is therefore compute-bound and to the fact that, a performance of one CPU chip is limited by the power consumption, size constraints and memory capacity, HPC systems grow substantially through the use of more processor units, especially the so-called CPU cores[3].

However, if the performance of memory and interconnect are not scaled accordingly, it makes the machine unattractive for most of the real-world applications: The higher the *Byte/Flop* ratio of an application, the more the HPC system loses its efficiency. The increasing number of processors and their aggregated performance results in increasing requirements on the interconnect connections, which became one of the bottle necks in HPC: The performance (latency and the bandwidth) of the high speed interconnect is by one to two orders of magnitude lower than those of the memory.

Figure 1 shows the trend of supercomputer performance in the top 500 lists of the past 26 years. The results are based on the sum of the performance of all supercomputers and the performance of the slowest one and the fastest one in each year of the time frame considered. The performance-optimized HPL shows almost identical behavior as a dense matrix-matrix multiplication and achieves around 90% of the theoretical peak performance on a small cluster and between 70% and 80% on the top supercomputers. Last several years dominate China and the U.S. the TOP500 list: while two fastest supercomputers are from the U.S. (2019), the third one is from China. Its name is „Sunway TaihuLight“[4]. Although still in 2017, „Sunway TaihuLight“ was the world's largest supercomputer. Another distinctive property is that the Sunway TaihuLight has no accelerators compared to Summit (rank 1) and Sierra (rank 2).

As one can see, the growth rate of performance has slowed down after 2011. The primary reason for this is the difficulty for Moore's Law to make the same advances as during the early stages of the development (see section 4.2.1).

Unfortantly, HPL doesn't reflect many challenges of HPC, so the HPL results are not representative of the performance of real-world applications. Real-world applications usually don't have such expensive compute-bound kernels as HPL. Those compute kernels are mostly memory-bound, which leads to a drop in performance to few percent of peak.

The new Benchmark „High Performance Conjugate Gradients“ (HPCG) has been developed with these discussions in mind. The HPCG benchmark assembles a distributed sparse linear system using a 27-point stencil at each grid point in a 3D computational domain, which has to be solved with a conjugate gradient

[2] „Powerfull“ is to be taken quite literally here, because the largest of the supercomputers consumes dozens of megawatt, 24 hours a day throughout the whole year.

[3] We are talking about CPUs here. Although accelerators such as GPUs are widely used in TOP500, the application of GPUs in scalable scientific HPC applications is much more limited than that of CPUs.

[4] Sunway TaihuLight is a Chinese supercomputer located at the National Supercomputing Center in Wuxi. It uses a total of 40,960 Chinese-designed SW26010 manycore 64-bit RISC processors with 256 cores.

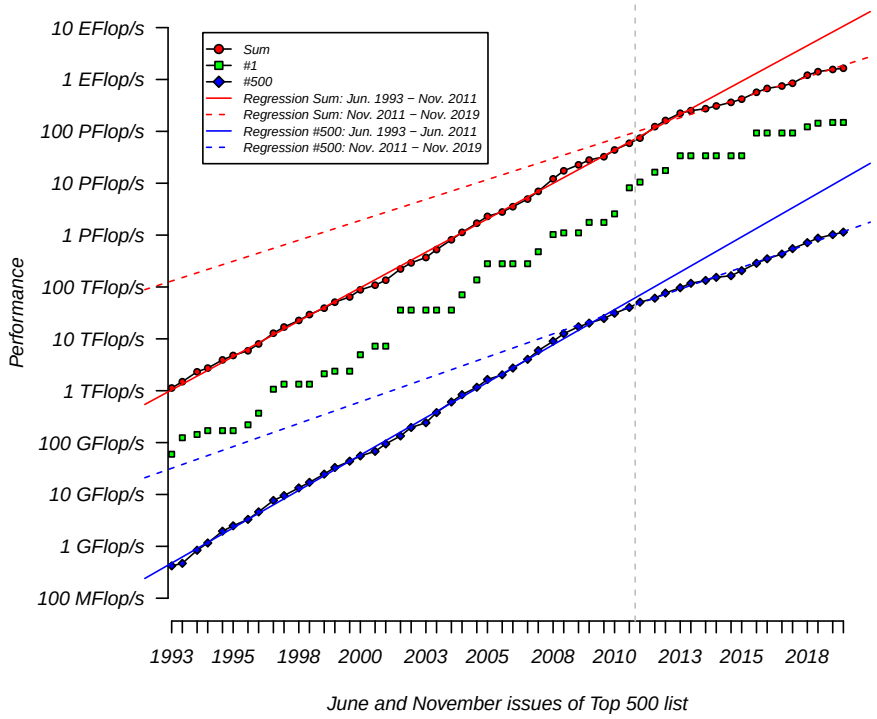


Figure 1.2: 26 years of TOP500 trend

solver, where the pre-conditioner is a three level hierarchical multi-grid (MG) method with Gauss-Seidel. The kernel of HPCG has a *Byte/Flop* ratio bigger than 4, which shows, in opposite to HPL, memory-bound behavior for all current processors, which limits the overall performance approximately proportional to the effective memory bandwidth of a single processor. On the other hand, HPCG as well as HPL do not specify the size of the sparse linear systems, which makes it possible to hide the communication costs. Nevertheless: in the year 2017, the world's largest supercomputer is capable of 480.85 TFlops but only occupies the fifth place in the ranking list of HPCG.

1.3.1 High performance computing in Germany

The Gauss Centre for Supercomputing (GCS, 2019) was founded in 2007 as a joint initiative between federal and state government and combines the three national supercomputing centers: the High Performance Computing Center Stuttgart (HLRS), Jülich Supercomputing Centre (JSC), and Leibniz Supercomputing Centre, Garching (LRZ). The three national supercomputing centers operate germany's most powerful supercomputers, providing the largest and most powerful supercomputing infrastructure in all of Europe. GCS enables access to HPC resources and training opportunities for German and European researchers. The HPC users and the developers are offered more than 60 courses addressing different aspects of HPC, which are conducted at HLRS, JSC and LRZ and are focusing on scientific simulations on HPC. Additionally, the centers offer HPC workshops and coordinate other activities to attract attendees from broad areas including industry, for example „Supercomputing-Akademie" at HLRS (Supercomputing-Akademie, 2019).

Another important task of GSC is the distribution of computation time on the Tier-0/Tier-1[5] systems of these centers. The regular „Large Scale Projects" calls enable applying for computing time on one or more

[5]Tier-0 and Tier-1 refer to the largest systems on european (Tier-0) and national (Tier-1) levels.

GCS's Systems[6]. For testing before submitting a proposal one can get a test account directly at one of the centers.

Following a consistent German HPC strategy to complement GCS, Germany's medium sized supercomputing centers founded the Gauß-Allianz (GA) (GA, 2019) in 2008. GA brings together 21 HPC centers with total 40 HPC systems (status for 2019).

1.4 The model experiment in BEAM-ME

A key objective of the project BEAM-ME was to ensure the transferability of the project results. A „model experiment" was integrated into the project with the aim to apply the developed methods and solvers to state of the art energy system models currently used by six renowned research institutions.

1.4.1 Participating models and partner institutions

The model experiment partners were selected through a series of competitive open tenders during the project. Important criteria in the selection process included the excellence in scientific work in energy system modelling especially with regard to research projects for the federal government, as well as complexity and scalability of the models contributed to the project BEAM-ME.

Outside of BEAM-ME, the models are used on a day to day basis to provide technology/market analyses and policy recommendations for the German, Swiss and European government and industry. The tenders were designed to select different types of energy system models allowing to benchmark the newly developed solver for a broad array of applications. The models distinguish the following modelling approaches:

- One model examines energy trading in particular and considers the technical requirements of many individual power plants in detail (EWL).
- Other models optimise the use of power plants in the electricity market for individual scenario years, partly coupled with the modelling of other sectors (heat, transport) (TUD, KIT).
- A third type of model examines the long-term development and transformation paths of the energy system. Besides the annual use of power plants, this type also models investments in different generation technologies for a period of 30-50 years. (EWI, DTU, PSI)

1.4.2 Aim of the model experiment

With regard to high performance computing, the models were adapted in order to be able to use the new methods developed in this project. In order to decompose the energy system models for parallel computing, the models had to be annotated in order to structure the problem into different blocks/sub problems suitable for calculation on the distributed nodes. Model annotation is a feature in the modelling language developed by the project partner GAMS. Members of the model experiment had to learn to develop models with a high number of blocks (to be distributed on a high number of cores) and at the same time not to increase complexity of the sub problems and the number of linking variables between the sub problems in order to allow the new algorithm to still be able to solve these problems.

The second goal of the model experiment, apart from running energy models on HPC, was to investigate different model based speed-up methods. Based on the results from experiences with REMix and experience on performance improvements by the partners, five methods for improving computational speed-up of energy system models were selected. In small groups these different speed-up methods were investigated and a comparison of current implementations took place. The group of modelers developed best practice methods and implementations.

1.4.3 Structure of the model experiment

The model experiment consisted of different parts as shown in figure 1.4.

[6]Projects are classified as „Large Scale" if they require in total more than 35 million core-hours in one year on a GCS member

1. **Training**
   The aim of several training sessions was to pass on the knowledge acquired in the project to the partners and to exchange information among themselves. A central part of the training was the use of the possibility to annotate GAMS models with respect to certain block structures and thus to create the basis for parallel solving with PIPS-IPM++. A second component of the training was the usage of the supercomputers in Jülich (JSC) and Stuttgart (HLRS).

2. **Runs on the fat nodes (JSC)**
   All partners provided model instances of different sizes in the second phase. These were calculated on individual nodes with particularly large memory („fat nodes“) on JUWELS (JSC) to enable comparability. Model instances of different sizes could be tested with different existing commercial solvers. The runs also represented the reference runs for the comparison with the distributed computations using the PIPS solver developed in this project.

3. **Annotation and PIPS runs on HPC**
   An important part of the project was the annotation of the individual models of the MEXT partners. Only by assigning variables and restrictions to blocks could the participating models be upgraded to allow decentralized computing on the high-performance computers. The partners jointly applied the annotation methods developed by GAMS. Different tools were made available to them by GAMS and DLR. Since this method had not been used before, this part of the project consisted of a committed exchange of experience and the development of suitable methods and processes for annotating the models. On this basis the model runs on HPC were carried out. This was done by the BEAM-ME consortium or the MEXT partners received assistance in starting and evaluating their own instances on the HPCs. The benchmark runs and lessons learned from the model experiment are presented in chapter 6.

4. **Development and comparison of model-based acceleration mechanisms**
   Besides the HPC calculations, the investigation of model-based performance enhancement methods was an important part of the model experiment. The partners agreed on five methods which were to be implemented and in some cases further developed by the partners within the framework of MEXT. The methods could then be compared and suitable methods identified.

centre's HPC systems.

Model	Model type	Instititution		Location	Model characteristic	Tender
REMix	Dispatch (main) and investment	**DLR**	**German Aerospace Center (DLR),** Department for Energy System Analysis	Stuttgart, Germany	Dispatch and investment model for the European Energy market and north Africa.	Main partner
DIMENSION	Dispatch and investment	**EWI**	ewi Research and Scenarios GmbH	Cologne, Germany	Dispatch and investment model for the European Energy market.	1
BALMOREL	Dispatch and investment	**DTU**	**Technical University of Denmark, DTU**	Copenhagen, Denmark	Open source model analysing the energy sector with focus on electricity and power to heat. The model comprises dispatch as well as long term investments.	1
WILMAR Joint Market Model (JMM)	High resolution dispatch model	**EWL**	**University of Duisburg Essen**, Chair for Management Science and Energy Economics	Duisburg, Germany	JMM is a dynamic dispatch planning model for electricity systems. It depicts the European Electricity system based on high resolution of available technologies and it is used to investigate different techno-economic research questions.	2
PERSEUS	High resolution dispatch model	**KIT**	**Karlsruhe Institute of Technology,** Institute of Industrial Production (IIP), Chair of Energy Economics	Karlsruhe, Germany	Long term dispatch model.	2
ELTRAMOD	High resolution dispatch model	**TUD**	**Technical University of Dresden**, Faculty of Business Management, Chair of Energy Economics	Dresden, Germany	Dispatch and load coverage in the European electricity sector (incl. Switzerland, Norway and Balkan states) with links to the heat sector	3
EU-STEM	Dispatch and Investment	**PSI**	**Paul Scherrer Institut**, Energy Economics Group	Villigen, Switzerland	EU-STEM is a European model for the long term analysis of the electricity sector. It comprises dispatch and long term investment analysis.	3

Figure 1.3: Models and research institutions contributing to the model experiment

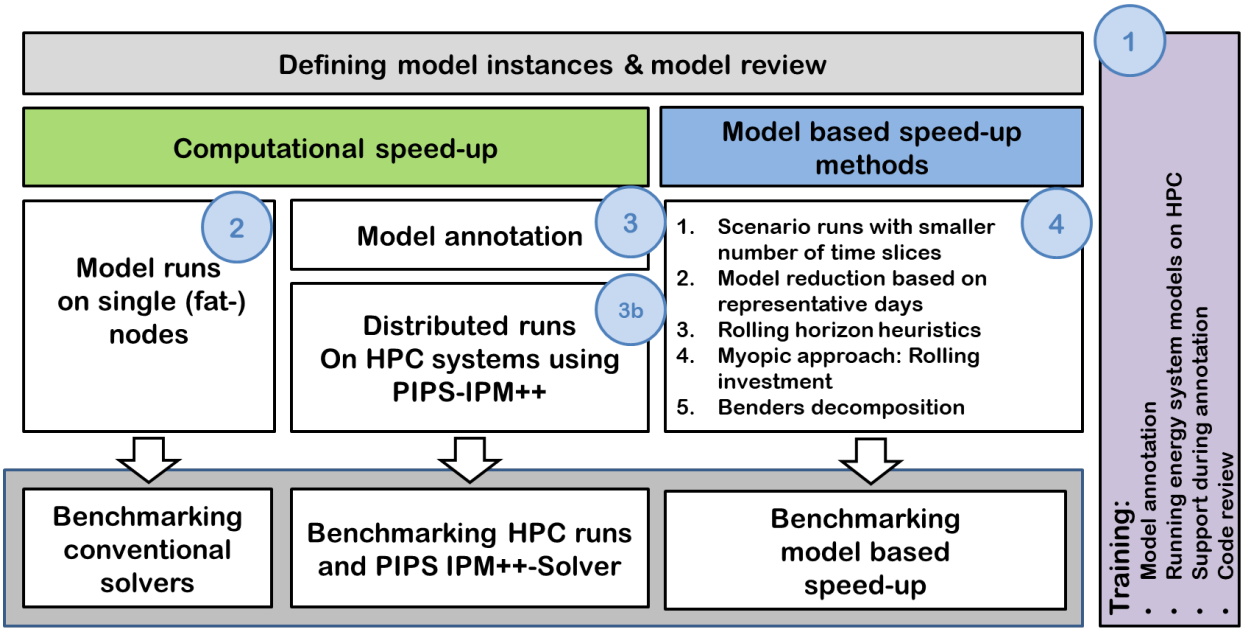


Figure 1.4: Models and research institutions contributing to the model experiment

Part I

Modeling Based Performance Enhancement

Chapter 2

Overview of modeling based performance enhancement methods[1]

Modeling based strategies for ESOM performance enhancement are related to the content and formulation of the model. Energy system modelers can implement and apply these strategies using their usual computing capacities. Modeling based strategies comprise heuristics (model reduction, stepwise or nested solving) and exact methods (different types of decomposition). We use the term "modeling based" interchangeably with the term "conceptual".

Despite the existence of a large number of modeling based speed-up approaches for ESOMs, it is not clear which methods are the most promising ones to improve the performance of ESOMs that are used in the field of applied energy system analysis. A majority of these models shares two characteristics (Zerrahn & Schill, 2017):

- Due to the assessment of high shares of power generation from vRES the time set that represents the sub-annual time horizon shows the largest size (typically 8760 time steps)
- To be able to increase the descriptive complexity of the models, the mathematical complexity is simplified. This means often the formulation of large monolithic LPs which are solved on shared memory machines.

The aim of this report is therefore to systematically assess the effectiveness of different performance enhancement approaches for such ESOMs. Rather than the comparison of models that deliver exactly the same results, we explore possible improvements in terms of needed computing time that can be achieved by implementing different conceptual speed-up techniques into an ESOM while staying within a defined range of accuracy.

2.1 Model reduction

Model reduction approaches are very common since they are effective due to the simply reduction of the size of the appropriate OP (less variables and constraints). Furthermore, they are also implicitly applied to ESOMs, for instance, due to limited input data access. Thus, these approaches usually manipulate input data in a pre-processing step, instead of changing the way how an ESOM is solved. Based on the treatment of available data we distinguish two forms of model reduction techniques: i) slicing and ii) aggregation.

[1] This chapter is based on the preprint of "Cao, Karl-Kiên and von Krbek, Kai and Wetzel, Manuel and Cebulla, Felix and Schreck, Sebastian (2019): *Classification and Evaluation of Concepts for Improving the Performance of Applied Energy System Optimization Models*, DOI: 10.3390/en12244656; Journal: Energies; URL: https://www.mdpi.com/1996-1073/12/24/4656"

2.1.1 Slicing

Slicing approaches translate into focusing to a specific sub-problem by ignoring existing interdependencies, and therefore only a part of the input data that could be analyzed is used. This means, for example, excluding technologies such as CHP plants from a model (Brouwer, van den Broek, Zappa, Turkenburg, & Faaij, 2016) or ignoring power exchange beyond neighboring regions on the spatial side (Weigt, Jeske, Leuthold, & von Hirschhausen, 2010). Regarding the temporal dimension, analyses are conducted only for a specific target year (Bussar et al., 2015) or time-slices are selected (Loulou & Labriet, 2008). These sub-sets represent either critical situations, such as the peak load hour, or typical time periods are defined which are supposed to be characteristic for the entire set of operational time steps. By this means, it needs to be noted that slicing approaches can lead to significant deviations of results compared to the global optimum of the full OP as they do not ensure that the relevant information within the available data is captured. However, if for the selection of specific slices a pre-analysis is conducted, we refer to this process as part of an aggregation as this approaches aim to take into account all input data. Therefore, they reduce the input data set in a way that relevant information is maintained as far as possible. In the context of ESOMs, aggregation can also be described as coarsening of resolutions for each of the characteristic model dimensions.

2.1.2 Spatial Aggregation

The treatment of large, spatially explicit data sets is a common challenge in the context of power network analysis. However, corresponding to the area of responsibility of system operators, methods for power networks were developed to study certain slices of the entire interconnected network. The objective of these classical network reduction techniques is therefore to simplify the neighborhood of the area of interest by the derivation of network equivalents based on given power flows. These equivalents, such as derived by Ward or REI methods, represent the external area which is required to show the same electrical behavior as the original network (Deckmann, Pizzolante, Monticelli, Stott, & Alsac, 1980). In the case of Ward equivalents, the networks' nodal admittance matrix is reduced by Kron's reduction (Dorfler & Bullo, 2013). In contrast, however, the REI procedure applies a Gaussian elimination to external buses. Power injections are preserved by aggregating them to artificial generators which are connected to a representative, radial network which is referred to as REI.

The principle of creating network equivalents is also applicable to ESOMs, although their scope is rather the interaction of different technologies than the exclusive assessment of stability or reliability of electrical networks. Recently, (Shayesteh, Hamon, Amelin, & Söder, 2014) adapted the REI approach to use-cases with high vRES penetration. However, this step of creating aggregated regions for a multi-area ESOM needs to be preceded by a partitioning procedure which allows for defining of regional clusters. In general, the appropriate clustering algorithms, such as k-means, group regions or buses with similar attributes together. In (Shayesteh et al., 2014) the admittance between two buses is used to account for strongly connected regions. Opposed to this, (Shi & Tylavsky, 2015) as well as (HyungSeon Oh, 2011) derive network equivalents based on reduced power transfer distribution factor (PTDF-) matrices which rely on the linearization of certain system operating points.

Despite the availability of a broad spectrum of sophisticated aggregation techniques, in the context of energy system analysis, the applied literature is governed by simple spatial aggregation approaches. In particular, they are usually characterized by a summation of demand and generation capacities, whereas intra-regional flows are neglected and regions are grouped based on administrative areas, such as market or country borders (Bethany A. Corcoran, Nick Jenkins, & Mark Z. Jacobson, 2012), (Katrin Schaber, Florian Steinke, & Thomas Hamacher, 2012), (Christian Bussar et al., 2016). Reasons therefore are, on the one hand side, the availability of required, large data sets of spatially explicit data for the broad diversity of technologies, such as potentials or existing infrastructure. On the other hand, the majority of network equivalents is based on pre-computed system states of the spatially highly resolved model, for example, a solved power flow study. This in turn requires the application of nested approaches (see section 2.2), where first simplifications to other scales of an ESOM are required in order to obtain the power flows of the entire network. By this means, reasonable simplifications are the use of time-slices in form of operational snapshots and the summation of power supply from all generation technologies.

Nevertheless, concerning scenarios of the European energy system (Thomas Anderski et al., 2014), as well as (Hörsch & Brown, 2017) take a step towards improved methodologies regarding aggregation of spatially highly resolved data sets. Both use power demand as well as installed generation capacities as attributes for state-of-the art clustering algorithms. However, while in (Thomas Anderski et al., 2014) PTDF-based equivalents are built, the authors in (Hörsch & Brown, 2017) apply a more or less straight forward process for creating spatially aggregated regions.

2.1.3 Temporal Aggregation

Temporal aggregation refers to representative time periods or the process of data down sampling derived from a highly resolved initial data set.

Down sampling is a method where time series based input data is coarsen to a lower temporal resolution (e.g. by averaging from 1-hourly to 6-hourly). In ESOM, down sampling typically affects demand profiles (e.g. electric or heat load) or the feed-in from vRES power. Although the approach is an effective way to reduce computing times – (Stefan Pfenninger, 2017) for example shows a reduction of CPU time up to 80% (scenario 90% 2014)–the method is rarely applied. This is due to the claim to capture the dynamics of variable power provision from renewable energy technologies. By this means, ESOMs typically rely on their highest resolved data and often use hourly input (Zerrahn & Schill, 2017). Exceptions can be found in studies that analyze the impact of different temporal resolutions in unit commitment approaches, e.g. in (Deane, Drayton, & Gallachóir, 2014) (5 min, 15 min, 30 min, 60 min) or in (O'Dwyer & Flynn, 2015) as well as in (Pandzzic, Dvorkin, Yishen Wang, Ting Qiu, & Kirschen, 2014) who both compare a 15 min resolution with hourly modeling.

More common is the combination of down sampling and the selection of representative time periods, such as applied in (Sylvie Ludig, Markus Haller, Eva Schmid, & Nico Bauer, 2011) or (Leuthold, Weigt, & von Hirschhausen, 2012). Representative time periods are intended to illustrate typical or extreme periods of time. These time intervals are then stacked up to derive the overall time horizon, e.g. one year. Moreover, also challenges exist to account for the chronological relationship between hours which in particular becomes important if time-linking constraints are incorporated in an ESOM. One approach to tackle this issue is presented by (Wogrin, Dueñas, Delgadillo, & Reneses, 2014) who define transitions between system states derived by applying a k-means-like clustering algorithm to wind and demand profiles. As stated in (Stefan Pfenninger, 2017), the selection of time-slices is either based on a clustering algorithm, such as k-means (Green, Staffell, & Vasilakos, 2014) or hierarchical clustering (Paul Nahmmacher, Eva Schmid, Lion Hirth, & Brigitte Knopf, 2016), or simple heuristics (Spiecker, Vogel, & Weber, 2013).

While temporal aggregation is an effective method to reduce computing times, it is not always clear which error is introduced by it. This issue has been tackled by a number of recent papers, such as (Stefan Pfenninger, 2017), (Haydt, Leal, Pina, & Silva, 2011) or (Sylvie Ludig et al., 2011). The studies unanimously highlight the rising importance of high temporal resolution with increasing vRES share. The authors also state that there exists no best practice temporal aggregation and emphasize that it strongly depends on the modeling setup. For instance, (James H. Merrick, 2016) recommends ten representative hours for robust scenarios when only variable demand is considered. This number, however, increases significantly when vRES and especially several profiles per technology are taken into account. With regard to representative days, necessary to avoid errors by aggregation, he finds that the number of 300 is appropriate. This represents a clear difference compared to the sufficient number of six representative days resulting in (Paul Nahmmacher et al., 2016). (Paul Nahmmacher et al., 2016) use the same clustering technique, but assess model outputs, such as total system costs, rather than the variance of clustered hours of the input time series.

2.1.4 Technological Aggregation

We define technology resolution as the abstraction level in a modeling approach to characterize the technologies relevant for the analysis. In this context, it can be stated that the higher the abstraction level, the better the performance of an ESOM. This applies to both the aggregation of input data and the mathematical model of a particular technology. The former, for example, refers to the representation of generation units

(electricity, heat, fuels) or flexibility options (e.g. grid, storage). More precisely, classifications of power plant types can be based on several attributes such as rated power, conversion efficiency, fuel or resources type. Technological resolutions therefore range from very detailed modeling of individual generation units (Mitra, Sun, & Grossmann, 2013) to general distinctions based on fuel consumption and resource (Bethany A. Frew, Sarah Becker, Michael J. Dvorak, Gorm B. Andresen, & Mark Z. Jacobson, 2016). However, the methods for deriving appropriate classifications or aggregations are rather based on simple grouping of attributes than on specific clustering algorithms.

Moreover, the classification of technologies is strongly connected to the mathematical description since physically more accurate models typically require more detailed data. In this regard, a broad body of literature investigates the necessary technological detail for power plant modeling. Often, these analyses compare simplified linear programming approaches (ED) with more detailed mixed integer linear programming (UC) models for least cost power plant dispatch. As a result, such studies assess differences in power plant dispatch (e.g. in (Bryan Stephen Palmintier, 2013), (Raichur, Callaway, & Skerlos, 2016), (Brady Stoll, Gregory Brinkman, Aaron Townsend, & Aaron Bloom, 2016), (Kris Poncelet, Erik Delarue, Daan Six, Jan Duerinck, & William D'haeseleer, 2016)) and, additionally, highlight effects on resulting metrics (e.g. storage requirements in (Cebulla & Fichter, 2017) or marginal prices of electricity generation in (Jan Abrell, Friedrich Kunz, & Hannes Weigt, 2008), (Langrene, van Ackooij, & Breant, 2011).

The same applies to transmission technologies where (Munoz, Sauma, & Hobbs, 2013) for instance, study modeling approaches (discrete vs. continuous grid capacity expansion) and their effects on the total system costs. Also technological classifications can be made for different voltage levels or objectives of grid operation (e.g. transmission or distribution). Regarding mathematical models, resolutions range from detailed, nonlinear AC-power flow over decoupled and linear DC-power flow to simple transshipment or transport models (Nolden, Schönfelder, Eßer Frey, Bertsch, & Fichtner, 2013).

2.2 Heuristics: Nested Approaches

Even though also mathematical exact decomposition techniques (see section 2.3) could be interpreted as nested approaches, in this section, we explicitly refer to methods that usually find near-optimal solutions rather than a theoretically guaranteed exact optimum. In this context, nested approaches are used as a synonym for heuristics. In contrast to meta-heuristics, this concerns methods that imply modifications of the ESOM regarding the conceptual layer and thus base on the same mathematical solver algorithm. In general, nested approaches are built on model reduction techniques (see section 2.1). Therefore, combinations of several reduced instances of the same initial ESOM (original problem) are usually solved sequentially. This means, that after the solution of the first reduced model is obtained, certain outputs are used as boundary conditions (e.g. in the form of additional constraints) for the following model(s) to be solved.

As mentioned above, ESOMs have linking constraints or variables that globally link points of one dimension. These characteristics are crucial for the decomposition of an OP into smaller instances of the same problem, regardless of whether it should be solved by an exact decomposition (see section 2.3) or heuristic approach. Often this is intuitively done by the application of a nested performance enhancement method where linking variables, such as power flows or endogenously added capacities are used to interface between the different reduced models.

In the literature, a wide range of examples for the applications of nested performance enhancement approaches exists. For instance, (Romero & Monticelli, 1994) propose an approach for TEP where they gradually increase the technological detail starting with a simple transport model, and finally taking into account Kirchhoff Voltage Law constraints as in a DC-power flow model.

With regard to the spatial scale, one obvious methodology can be described as "spatial zooming", which is similar to the classical methodology applied for power network analysis (see section 2.1.2). Possible implementations can look like as follows: First a large geographical coverage is considered in a coarse spatial resolution to study macroscopic interdependencies. In a second step, these interdependencies, such

as transnational power flows, can be fixed in order to conduct a detailed analysis of the region of interest (Gils, 2015). In (Haikarainen, Pettersson, & Saxen, 2016) the spatial dimension is simplified by the derivation of network clusters, while for the solution of the original problem a selection of binary variables related to pipelines and suppliers is restricted.

Comparing the different reduced models used in a nested approach, typically, a decrease of resolution on one scale is often accompanied by an increase on another. In this regard, one common approach is decoupling of investment decisions by "temporal zooming". Therefore, first, a power plant portfolio is developed over the analyzed planning horizon using a simplified dispatch model and pre-defined time-slices to simulate the operation. In order to check whether the derived power plant portfolio performs well for a selected target year, UC constraints are added and capacities are fixed in the subsequent model run(s) (Kris Poncelet et al., 2016), (Scholz, Sandau, & Pape, 2016), (Brouwer et al., 2016). A similar method call (Babrowski, Heffels, Jochem, & Fichtner, 2013) al "myopic approach" . In this case, for each year of the planning horizon a model run is performed, whereas the resulting generation expansion is taken as an offset of installed power generation for the subsequently analyzed target year.

In terms of size, rather than high resolutions on the technological or spatial scale, in applied energy system analysis, ESOMs often need to consider large sets that represent the temporal scale (i.e. time series of 8760 hours) in order to capture the variability of vRES (Pfenninger, Hawkes, & Keirstead, 2014). In the following, we therefore introduce two heuristic methods for this particular dimension in detail.

2.2.1 Rolling Horizon

The general idea behind rolling horizon methods is to split up the temporal scale (temporal slicing) into smaller intervals to obtain multiple reduced ESOMs to be solved sequentially. In particular, this method is used for two reasons. One is to account for uncertainties by frequently updating limited knowledge concerning the future. This applies, for instance, to forecasts of load or electricity production from renewable energy sources. Although the main principles of a rolling horizon approach apply to both operational and investment planning, in the following we mainly refer to the former, the rolling horizon dispatch. Therefore, a typical application is short-term scheduling of power systems with a high penetration of renewables (Tuohy, Denny, & O'Malley, 2007), (Barth, Brand, Meibom, & Weber, 2006), (Silvente, Kopanos, Pistikopoulos, & Espuña, 2015).

The other purpose of implementing a rolling horizon approach to an ESOM is the premise that the total computing time for solving individual reduced problems stays below the one for obtaining a solution for the original problem.(Marquant, Evins, & Carmeliet, 2015) report of a wide variety of speed-up achievements ranging from 15 up to 100 times. Depending on the model size there usually exists an optimal number of time windows in terms of computing time, since the computational overhead for creating reduced models increases with the number of intervals. Furthermore, the planning horizon of an individual time window usually includes more time steps than necessary for the partial solution. In the context of energy system analysis, this overlap is important to emulate the continuing global planning horizon. Especially the dispatch of seasonal storage units is strongly affected by this as, without any countermeasures, it is more cost-efficient to fully discharge the storage until the end of an operational period. Also time-linking variables and constraints, such as annual limits on emissions, can only barely be considered in this way since global information regarding the temporal scale can only be roughly estimated for each time window. For this reason, inter alia indicated by a trend to overestimate the total system costs (Marquant et al., 2015), the aggregation of interval solutions does not necessarily end up at the global optimum of the original problem.

2.2.2 Temporal Zooming

Concerning their capability to improve the performance of an ESOM, rolling horizon approaches have one particular disadvantage. Since each partial solution is updated by a subsequent one, the reduced ESOM instances are sequentially coupled. This prevents parallel solving.

The heuristic, we refer to as temporal zooming, overcomes this issue and allows for solutions closer to the exact optimum of the original problem. Therefore, the rolling horizon approach is adapted in the following

way. In a first step, time-linking information is gathered from the solution of an additional ESOM instance which is reduced on the temporal scale. But, in contrast to the reduced ESOMs which consider specific intervals within the full operational horizon, the temporal resolution is down sampled. This in turn allows optimizing the dispatch of the original problem for the full planning period. Values of variables from this first model run can subsequently be used to tune the consideration of global time-linking variables and constraints within the intervals. Despite the need for an additional model run, total computing times for obtaining a final solution can be expected to be at least competitive compared to rolling horizon approaches. This is due to the fact that, on the one hand side, overlaps are not required and, on the other hand, the temporally sliced ESOMs can be solved in parallel.

2.3 Mathematically exact decomposition techniques

Decomposition approaches are a well-known instrument for reducing the computing time in OPs. In this case, an OP is broken down into a master and sub-problem(s). With regard to the structure of the OP's coefficient matrix, the decomposition can be exploited for the creation of individual blocks. Ideally, block structures with globally linking variables or constraints can be isolated from the sub-problems, making them solvable independently of each other, i.e. in parallel.

Despite this similarity to nested approaches, such as temporal zooming, the crucial difference concerning exact decomposition techniques is the theoretically proven guarantee to find the optimal solution of the original problem (Conejo, Castillo, Mínguez, & García Bertrand, 2006). However, this requires an iterative solution of a master and sub-problems. Therefore, it can be stated, that compared to nested approaches, decomposition techniques provide the best accuracy possible, but at the expense of additional computing time.

2.3.1 Dantzig-Wolfe Decomposition

In particular, approaches that can treat linking constraints are Dantzig-Wolfe decomposition and Lagrangian relaxation. The general idea behind both is to remove the linking constraints from the original problem to observe a relaxed problem that decomposes into sub-problems. In the case of Dantzig-Wolfe decomposition the objective function of the appropriate master problem consists of a linear combination of solutions of the relaxed problem. Starting with an initial feasible solution, this function is extended with each iteration if the new solution of the relaxed problem verifiably reduces the objective value (i.e. costs). Accordingly, this process is called column generation since each iteration literally creates also new columns in the master problems' coefficient matrix. (Angela Flores Quiroz, Rodrigo Palma Behnke, Golbon Zakeri, & Rodrigo Moreno, 2016) use this approach in order to decouple discrete investment decisions from dispatch optimization for a GEP with UC-constraints. Although performance enhancements are examined for realistic applications of different sizes, due to memory issues of not-decomposed benchmark models, these improvements are only quantified for small model instances (ca. 3 times faster, 95 % less memory usage).

2.3.2 Lagrange Relaxation

The Lagrangian relaxation is derived from the common mathematical technique of using Lagrange multipliers to solve constrained OPs where linking constraints are considered in the form of penalty terms in the objective function of the master problem. In the applied literature, Lagrangian relaxation is used by (Virmani, Adrian, Imhof, & Mukherjee, 1989) to treat the linking constraints, that couple individual generation units in the UC problem. More recently, (Q. Wang, McCalley, Zheng, & Litvinov, 2016) applied Lagrangian relaxation on a security-constrained OPF problem in order to decouple a security constraint that links variables of two scales, contingencies and circuits. However, as the treated problem consists of both linking constraints and linking variables, also Benders decomposition is applied.

2.3.3 Benders Decomposition

Opposed to the previously described decomposition approaches, Benders decomposition can be applied to OPs with linking variables. The general concept of splitting an OP by this approach is based on fixing the

linking variables in the sub-problem(s) using their values from the master problem's solution. To improve the solution of the master, the sub-problems are approximated by additional constraints. These so called Benders cuts in turn rely on the dual variables of the obtained solutions in the sub-problems. As ESOMs are often formulated as linear programs, due to duality of these problems, a translation of linking constraints into linking variables is possible and thus Benders decomposition can be applied to almost all kinds of ESOMs. Accordingly, it is a frequently exploited decomposition technique in the applied literature. Table 2.1 lists a number of publications that apply decomposition techniques to ESOMs that are at least partially formulated as linear programs (LPs) or mixed-integer linear programs (MILP). However, due to the non-linearity of AC-power flow constraints, also non-linear programs (NLPs) are considered.

Table 2.1: Overview of decomposition techniques applied to ESOMs

Authors	Math. problem type	Descriptive problem type	Decomposed model scale	Decomposition technicque	Decomposition purpose
(Alguacil & Conejo, 2000)	MIP/NLP	Plant and grid operation	Time, single sub-problem	Benders decomposition	Decoupling of UC and multi-period DC-OPF
(Nima Amjady & Mohammad Reza Ansari, 2013)	MIP/NLP	Plant operation		Benders decomposition	Decoupling of UC and AC-OPF
(Binato, Pereira, & Granville, 2001)	MIP/LP	TEP		Benders decomposition	Decoupling of discrete investment decisions and DC-OPF
(Esmaili, Ebadi, Shayanfar, & Jadid, 2013)	NLP/LP	Grid operation		Benders decomposition	Decoupling of AC-OPF and congestion constraints
(Angela Flores Quiroz et al., 2016)	MIP/LP	GEP	Time, 1-31 sub-problems sequentially solved	Dantzig-Wolfe decomposition	Decoupling of discrete investment and UC
(Habibollahzadeh & Bubenko, 1986)	MIP/LP	Plant operation		Benders decomposition	Decoupling of UC and ED
(Khodaei, Shahidehpour, & Kamalinia, 2010)	MIP/LP	GEP-TEP	Time, 2 sub-problem types, sequentially solved	Benders decomposition	Decoupling of discrete investments into generation and transmission capacity, security constraints and DC-OPF
(Jorge MartÃnez Crespo, Julio Usaola, & JosÃ© L. FernÃ¡ndez, 2007)	MIP/NLP	Plant and grid operation	Time, 24 sub-problems, sequentially solved	Benders decomposition	Decoupling of UC and security constraint AC-OPF
(Roh, Shahidehpour, & Fu, 2007)	MIP/LP	GEP-TEP	Time, up to 10 • 4 sub-problems, sequentially solved	Benders decomposition and Lagrangian Relaxation	Decoupling of discrete investments into generation and transmission capacity, security constraints and DC-OPF
(Virmani et al., 1989)	LP/MIP	Plant operation	Technology (generation units), up to 20 sub-problems, sequentially solved	Lagrangian Relaxation	Decoupling of unit specific(UC) and cross-park (ED) constraints
(S. Wang, Shahidehpour, Kirschen, Mokhtari, & Irisarri, 1995)	LP/MIP	Plant and grid operation	Space, 26 sub-problems, sequentially solved	Lagrangian Relaxation	Decoupling of DC-OPF and UC
(Jianhui Wang, Shahidehpour, & Zuyi Li, 2008)	MIP/NLP	Plant and grid operation	Scenarios and time, 10 • 4 sub-problems, sequentially solved	Benders decomposition	Decoupling of UC, scenario specific system adequacy constraints and network security constraints

(Q. Wang et al., 2016)	LP	Plant and grid operation	Technology (circuits) and time (contingencies), 2 sub-problem types, sequentially solved	Lagrangian Relaxation and Benders decomposition	Decoupling of DC-OPF, system risk constraints and network security constraints

2.3.4 Further Aspects

Besides the already presented decomposition techniques that rely on iteratively solving a master and sub-problem(s), also mathematically exact approaches exist that are based on individual information exchange between sub-problems rather than on the coordination provided by a master.(Zhao, Litvinov, & Zheng, 2014), for instance, use this marginal based approach for independent scheduling in a multi-area OPF problem. Compared to the heuristics presented above, this can be interpreted as the spatially decomposed counterpart to the (temporally decomposed) rolling horizon approach.

Although decomposition approaches provide the capability to improve the performance of solving independent sub-problems of an ESOM in parallel, these techniques are mostly applied for another purpose which results in the iterative solution of a master and one sub-problem. A complicated mathematical problem, such as a large NLP, is simplified by splitting it up into two problems, a smaller NLP on the one hand and a less complicated problem, such as a MIP, on the other. This applies especially to the examples in table 2.1 for which nothing is listed in the column "Decomposed model scale". And even though the most frequently identified, decomposed model scale is found to be the temporal dimension, this usually refers to the separation of sub-annual operation scheduling and long-term investment planning in GEP or TEP. According to table 2.1, the other typical application of exact decomposition techniques is decoupling of power-flow or security constraints from an UC model which generally refers to a spatial decomposition.

The computational benefits of parallel computing are especially exploited in the context of stochastic OPs. Here the temporal scale is extended by almost independent branches which are referred to as scenarios. These scenarios represent different possible futures which can be determined in parallel (sub-problems) while the assessment of these several alternatives is done by the master problem. Besides the classical decoupling of investment and operation decisions, this approach is also suitable in the context of short-term scheduling. For example, (Papavasiliou, Oren, & Rountree, 2015) apply Lagrangian relaxation to decompose by scenarios for a stochastic unit commitment model with dc-power flow constraints. Opposed to most ESOMs, they solve their model on a high performance computer with distributed memory architecture. As can be expected, (Papavasiliou et al., 2015) find a significant speed-up due to parallelization. This performance increase, however, poorly scales with the number of cores (e.g. speed-up factor 7 for a hundred times the number of cores). Nevertheless, the main goal of the presented approach is to stay below a threshold of computing time that is suitable for day-ahead operation planning.

Chapter 3

Performance analyses of modeling based strategies in REMix[1]

This chapter describes the performance analysis for a selection of modelling based strategies. From the modeling based methods introduced in the previous chapter, five approaches for complexity reduction have been identified for a detailed benchmark analysis. In the first part we introduce these methods and outline the model setup as well as essential model characteristics of the REMix energy system model (Section 3.1.2). The following section 3.1.3 outlines the approaches for model reduction and describes how to implement the selected methods for (1) temporal and spatial aggregation, (2) rolling horizon dispatch, as well as (3) sub-annual temporal zooming approaches. The second part of this chapter describes in depth the results of the benchmark analysis for the modeling based strategies and outlines lessons learned for future model development: Different ways of model formulation are evaluated. The results in section 3.2 provide a systematic benchmark of the different speed-up approaches. It comprises of (1) an evaluation of pre-analyses methods, (2) spatial and temporal aggregation methods as well as (3) heuristic decomposition methods. The chapter concludes in section 3.3 with a summary of the main findings including a critical discussion of limitations and methodological improvements.

Despite the existence of a large number of speed-up approaches for Energy System Optimization Methods (ESOMs), it is not clear which methods are the most promising ones to improve the performance of ESOMs that are used in the field of applied energy system analysis. In addition to the arrow-head structure of the coefficient matrix (presence of linking constraints and linking variables, see section 1), a majority of these models shares three characteristics (Zerrahn & Schill, 2017):

1. To be able to increase the descriptive complexity of the models, the mathematical complexity is often simplified. This frequently means the formulation of large monolithic linear programs (LPs) which are then solved „as one block" on shared memory machines.

2. Due to the assessment of high shares of power generation from variable renewable energy sources (vRES) the set that represents the sub-annual time horizon shows the largest size (typically 8760 time steps)

3. A great number of applied ESOMs are based on mathematical programming languages such as GAMS or AMPL rather than on classical programming languages. Those languages enable model formulations which are close to the mathematical problem description and take the task of translation into a format that is readable for solver software. For this reason, the execution time of the appropriate ESOMs can

[1]This chapter is based on the preprint of "Cao, Karl-Kiên and von Krbek, Kai and Wetzel, Manuel and Cebulla, Felix and Schreck, Sebastian (2019): *Classification and Evaluation of Concepts for Improving the Performance of Applied Energy System Optimization Models*, DOI: 10.3390/en12244656; Journal: Energies; URL: https://www.mdpi.com/1996-1073/12/24/4656"

by roughly divided into two parts, the compilation and generation of the model structure requested by the solver and the solver time.

For the following analyses, we also use GAMS which is, according to a review conducted by (Zerrahn & Schill, 2017), a very popular modelling language in the field of energy systems analysis. We focus on large GAMS models for which total computing time is mainly dominated by solver time.
The general aim of this report is to systematically assess the effectiveness of different performance enhancement approaches for ESOMs that share the above mentioned characteristics. Rather than the comparison of models that deliver exact the same results, we explore possible improvements in terms of required computing time that can be achieved by implementing different conceptual speed-up techniques into an ESOM while staying within a sufficient accuracy range.
By this means, our aim is not to compare all above presented speed-up approaches, but those which are able to achieve the comparatively best performance enhancement. In this context, our hypothesis for the selection of model-based speed-up approaches to be systematically evaluated relies on three basic premises:

1. We focus on very large LPs that have a sufficiently large size (therefore allowing the computing time to be dominated by the solver time) and still maintaining the possibility to be solved on a single shared memory computer.
   If we implement an approach that allows for reduction or parallelization of the initial ESOM by treating a particular dimension, the highest potential therefore can be explored by applying such an approach to the largest dimension (generally the time set). Accordingly:

2. We place emphasize on speed-up strategies that work along the temporal scale of an ESOM.
   A high potential for performance enhancement still lies in parallelization, even though, for this study, it is limited to parallel threads on shared memory architectures. Exact decomposition techniques have the advantage to enable parallel solving of sub-problems. We claim that each exact decomposition technique can be replaced by a heuristic, where the iterative solution algorithm is terminated early. In this way, the highest possible performance should be explored, because further iterations only improve the model accuracy; however they require more resources in terms of computing time. In addition, according to the literature in Table 2.1, it can be observed, that mathematically exact decomposition techniques are applied less often with the objective of parallel model execution, but instead the separation of a more complicated optimization problem (e.g. MIPs from non-linear programms) from an easy-to-solve one. Even though for very large linear programming problems this is not necessary. For these reasons:

3. We only analyze model reduction by aggregation and heuristic decomposition approaches.

3.1 Materials and methods

3.1.1 Overview

Our evaluation approach should provide an assessment of model-based performance enhancement approaches for a very large ESOM that is intended to produce results for real life use-cases. However, this implies a couple of challenges. A proper adaptation of a large applied ESOM for the comparison of a broad set of speed-up strategies is very time-consuming. Accordingly, we limit the evaluation to the following performance enhancement approaches:

- model reduction by spatial and temporal aggregation
- rolling horizon
- temporal zooming

Moreover, to meet the requirement for an assessment of very large ESOM instances, we want to prevent the implementation of speed-up strategies into a model that is easily solvable by a commercial solver. Nevertheless, for having references for benchmarking solving the problem must still be possible. Hence, we select an existing ESOM for which we know from experience that obtaining a solution is time consuming but not impossible.

Besides, for fair benchmarking, it must be ensured that the reference model already performs well, e.g. with regard to solver parameterization. To meet this requirement our first methodological step is to conduct a source code review for the applied ESOM and follow recommendations by GAMS developers and (B. A. McCarl, 2000). Although most of the corresponding suggestions for improvements by GAMS aim at the reduction of the GAMS execution time, the main objective of this review step is the identification of source code snippets that cause the creation of redundant constraints. In practical terms, this means an explicit exclusion of unnecessary cases by broadly applying conditional statements ($-conditions). Otherwise, unnecessary large models would be passed to the solver.
Finally, it is essential that all model instances that should be compared are executed on identical hardware which should be exclusively available for the ESOM-related computing processes. Ensuring this across the whole evaluation exercise would require a large number of computers with comparatively large memory (>200 GB) to conduct the analysis within practical time spans. Due to a limited access to such equally equipped computers, we guarantee this only for benchmarks across each particular performance enhancement strategy.
The remainder of this section is structured as follows: The modeling setup consisting of a description of the applied ESOM and its characteristics and data, as well as the used solver and its basic parameterization, are described in section 3.1.2. The implementations of speed-up approaches to be evaluated are then presented in sub-chapter 3.1.3. Finally, we set up an evaluation framework that ensures at least a fair comparison of model performance and accuracy across different parametrizations of a particular speed-up approach.

3.1.2 Modeling setup

We use the ESOM REMix for the benchmark analysis. As there exist several parameterizations of the model which, on the one hand, share the same source code but, on the other hand, focus on various research questions and thus have different scopes in terms of available technologies, geographical study area and time horizon, REMix can also be regarded as a modeling framework. The benchmark in this study was conducted with two model setups which were partially extended during the time of the project. Although most of our analyses are performed for both of them, the results presented in section 3.2 build on the REMix instance presented in (Cao, Metzdorf, & Birbalta, 2018). The corresponding LP represents the German power system for an energy scenario of the year 2030. In its basic configuration it is a CO2-emission-constrained DC-OPF problem that considers renewable and fossil power generators, electricity transport within the high voltage transmission grid as well as storage facilities such as pumped hydro power plants and lithium-ion batteries. In addition, no generation capacities are optimized but capacities of both transmission lines and energy storage are optionally considered for expansion planning. To be able to observe a significant expansion of these technologies, their initial values for installed capacities represent the state of 2015. Hence, the installed capacity of lithium-ion batteries is zero. It needs to be noted that this configuration can lead to loss of load situations if capacity expansion is omitted. This is due to the fact that the power plant portfolio of the underlying scenario relies on the assumption that suitable load balancing capability of the power system can be provided by lithium-ion batteries and additional power transmission capacities.
A fact sheet of the appropriate REMix model setup is shown in Table 3.1 which also provides information about the input and output data.

Table 3.1: Model fact sheet of the applied configuration of REMix based on (Cao et al., 2018).

Model name	**REMix**			
Author/Institution	German Aerospace Center (DLR), Institute of Engineering Thermodynamics			
Model type	Linear programming Minimization of total costs for system operation and expansion Economic dispatch Optimal dc power flow with expansion of storage and transmission capacities			
Sectoral focus	Electricity			
Geographical focus	Germany			
Spatial resolution	488 nodes			
Scenario year	2030			
Temporal resolution	8760 time steps (hourly)			
Input parameters		Dependencies		
		Temporal	Technical	Spatial
	Conversion efficiencies (Teruel, 2015)		x	
	Operational costs (Teruel, 2015)		x	
	Fuel prices and emission allowances (Egerer et al., 2014)		x	
	Electricity load profiles (*Open Power System Data Data Package Time series*, 2017)	x		x
	Power generation, storage and grid transfer capacities and annual electricity demand (Rippel, Preuß, Meinecke, & König, 2017), (Wiegmans, 2016), (Hofmann, Hörsch, & Gotzens, 2018)		x	x
	Renewable energy resources feed-in profiles	x	x	x
	Import and export cross-border power flow time series (ENTSO-E, 2012)	x		x
Evaluated output parameters	System costs (objective value)			
	Generated power		x	x
	Added storage/transmission capacities			x
	Storage levels	x	x	x

3.1.2.1 Essential model constraints

The majority of the mathematical formulations of REMix is presented in (Gils, Scholz, Pregger, de Tena, & Heide, 2017). As discussed in section 1.1 and 1.1.1, the coefficient matrix structure of the corresponding LPs contains linking variables and constraints. Besides variables that are induced by enabling capacity expansion (equation 1.1), a great number of linking elements result from modeling power transmission using the dc approximation (spatially linking) or storage facilities (temporally linking). Furthermore, constraints that reflect normative targets, e.g. necessary for modeling greenhouse gas mitigation scenarios, cause interdependencies between large sets of variables (spatially and temporally linking). For a better comprehensibility equation 3.1 to 3.4 describe these constraints in a simplified manner, i.e. without conditional statements, additional index sets or scaling factors (as implemented in REMix).

Storage energy balance:

$$
\begin{aligned}
&p_{s+}(t,n,u_s) - p_{s-}(t,n,u_s) - p_{ls}(t,n,u_s) = \frac{E_s(t,n,u_s) - E_s(t-1,n,u_s)}{\Delta t} \\
&\forall t \in T, n \in N, \forall u \in U, U_s \subset U \\
&\\
&p_{s+}/p_{s-} : \text{storage charge/discharge power} \\
&p_{ls} : \text{storage self-discharge (losses)} \\
&E_s : \text{stored energy}
\end{aligned}
\tag{3.1}
$$

DC power flow:

$$
\begin{aligned}
&p_{im}(t,n) - p_{ex}(t,n) - p_{lt}(t,n) = \sum_{n'} B(n,n')\Theta(n',t) \\
&\forall t \in T, \forall n \in N
\end{aligned}
\tag{3.2}
$$

$$
\begin{aligned}
&p_{f+}(t,l) - p_{f-}(t,l) = \sum_{l} \sum_{n'} B_{diag}(l,l') K^T(l,n)\Theta(n,t) \\
&\forall t \in T, \forall l \in L \\
&p_{im}/p_{ex} : \text{power import/export} \\
&p_{lt} : \text{transmission losses} \\
&p_{f+}/p_{f-} : \text{active power flow along/against line direction} \\
&\Theta : \text{voltage angle} \\
&B : \text{susceptances between regions} \\
&B_diag : \text{diagonal matrix of branch susceptances} \\
&K : \text{incidence matrix} \\
&L : \text{set of links (e.g. transmission lines)}
\end{aligned}
\tag{3.3}
$$

Emission cap:

$$
\begin{aligned}
&\sum_{t} \sum_{n} \sum_{u} p(t,n,u)\eta_e(u) \leqslant m \\
&\eta_e : \text{fuel specific emissions} \\
&m : \text{maximal emissions}
\end{aligned}
\tag{3.4}
$$

3.1.2.2 Solver parametrization and hardware environment

In preliminary experiments resulting from a broad spectrum of REMix applications, ranging from country specific cross-sectoral energy systems ((Gils & Simon, 2017), (Gils, Simon, & Soria, 2017)) to multi-regional ((Gils, Scholz, et al., 2017), (Cao, Gleixner, & Miltenberger, 2016), (Scholz, Gils, & Pietzcker, 2017), (Gils, Bothor, Genoese, & Cao, 2018)) and spatially highly resolved power systems (Cao et al., 2018), for monolithic LPs, we observed the best performance in terms of computing time and RAM requirements with the following solver parameters when using CPLEX:

1. LP-method: barrier
2. Cross-Over: disabled
3. Multi-threading: enabled (16 if not otherwise stated)

4. Barrier tolerance (barepcomp)
    - 1e-5 for spatial aggregation with capacity expansion
    - default (1e-8) for the remaining approaches
5. Automatic passing of the presolved dual LP to the solver (predual): disabled
6. Aggressive scaling (scaind): enabled

Especially in the case of the first three solver options, LPs that previously could not be solved within time spans of multiple days, turned out to be solvable in less than 24h. With regard to the solver parameter 5, the amount of required RAM could be significantly decreased. For example, model instances that showed a peak memory demand of 230 GBs when setting predual to -1, otherwise exceeded the available RAM of 300 GBs. For these reasons, all of the following analyses are conducted with GAMS release 25.1.3 using CPLEX 12.8.0 with the above listed solver parameters. In addition, for all implementations of heuristic decomposition approaches either the GAMS option solvelink=5 (rolling horizon, temporal zooming) or solvelink=6 (temporal zooming with grid computing) are used to avoid delay times due to frequent read and write operations on the hard disk.
With regard to available hardware, computers with the following specifications (see Table 3.2) are available:

Table 3.2: Specifications of available computers for solving model instances.

#	**Processor**	**Available Threads**	**Available memory**
1	Dual Intel Xeon Platinum 8168	2x 24 @ 2.7 GHz	192 GB
2	Intel Xeon Gold 6148	2x 40 @ 2.4 GHz	368 GB

3.1.2.3 Original REMix instances and their size

As indicated in Table 3.1 the applied REMix model performs a DC-OPF which is optionally extendable by capacity expansion planning for storage and transmission infrastructures. Depending on this optional setting, two original model instances can be distinguished. We refer to them as „REMix Dispatch“ and „REMix Expansion“. Due to the different purposes of the decomposition heuristics to be evaluated, the two original models are only investigated for a sub-set of speed-up approaches. The rolling horizon approach is only sufficiently applicable to dispatch problems since investment decisions for especially short time intervals lead to a significant overestimation of required capacity expansion. In contrast, temporal zooming is explicitly suited for problems that account for capacity expansion.
To get an impression of model size, we measure the number of constraints, variables and non-zero elements of the coefficient matrix reported by the solver after performing the pre-solve routines. The appropriate values are indicated in Table 3.3. They show that enabling expansion planning is costly, especially with regard to the number of constraints. Compared to the number of variables which is increased by approximately 30%, the number of constraints is more than tripled. Nevertheless, this results in a coefficient matrix with lower density - since the number of non-zeros is only doubled.

Table 3.3: Specifications of initial REMix model instances.

Original model instance name	**Applied speed-up approaches**	**Number of variables**	**Number of constraints**	**Number of non-zeros**
REMix Dispatch	spatial aggregation temporal aggregation rolling horizon dispatch	30,579,396	9,214,488	69,752,951
REMix Expansion	spatial aggregation temporal aggregation sub-annual temporal zooming	43,169,135	32,805,201	137,967,269

3.1.3 Implementations

3.1.3.1 Approaches for model reduction by aggregation

The implemented aggregation approaches either treat the temporal or spatial scale. In case of the temporal scale, simple down-sampling is applied to load and feed-in profiles from vRES. Those parameters are available as hourly time series (temporally resolved). For down-sampling they are averaged to achieve a data aggregation and accordingly a reduction of the model size by factor M. For instance, when transforming a demand time series and, for reasons of simplicity, index sets of the other dimensions are ignored, the appropriate calculation rule is (equation 3.5):

$$
\begin{aligned}
& d_{agg}(t_M) = \sum_t \Pi_t(t_M, t) d(t) \\
& \forall t_M \in T_M, M \in \mathbb{N} \\
& T_M : \text{set of merged (down-sampled) time steps} \\
& \Pi_t : \text{map that assigns time steps to merged time steps} \\
& d_{agg} : \text{temporally aggregated power demand time-series}
\end{aligned}
\tag{3.5}
$$

Setting M=4 thus results in input time series that have a 4-hourly resolution. In other words, instead of t=1, ..., 8760 only t_M=1, ..., $\frac{8760}{4}$ consecutive data points need to be considered in a REMix instance which we refer to be "temporally aggregated".

With regard to the spatial aggregation methodology, we apply the following data processing: First a network partitioning is performed to define which regions of the original model parameterization must be merged. Therefore, an agglomerative clustering is used by applying the implementation of this algorithm from scikit learn (Pedregosa et al., 2011) to the adjacency matrix of the original model's network. We chose this clustering methodology as it ensures that merged regions are only built from neighboring regions. In addition, the clustering algorithm itself scales well with regard to various numbers of clusters.

Secondly, we create "network equivalents". The applied data aggregation relies on the premise that regions represent so called "copper plates" which means that transmission constraints are ignored within these areas. As a consequence, most nodal properties, such as installed power generation capacity or expansion potentials as well as power demand are spatially aggregated by simple summation. A special case is the aggregation of feed-in time series. Here a case distinction is applied, where the profiles of renewable power generation are aggregated by weighted averaging. The weights are taken from the installed power generation capacities of the respective regions, normalized by the sum over the installed capacities within the aggregated region. If there are no capacities installed (e.g. in the case of green-field expansion planning), the maximum capacities resulting from a renewable energy potential analysis are used.

Data that is related to links between regions, such as power transmission lines, is also specially treated: Transmission lines that would lie within an aggregated region are ignored. The transmission capacities of parallel cross-border links are summed up, while link lengths, used for approximation of losses and susceptances of parallel lines, are combined as it is common for parallel circuits, for instance:

$$
\begin{aligned}
& B_{agg}(l_M) = \frac{1}{\sum_l \Pi_l(l_M, l) \frac{1}{B(l)}} \\
& \forall l_M \in L_M \\
& \\
& L_M : \text{set of merged links} \\
& \Pi_l : \text{map that assigns links to merged links} \\
& B_{agg} : \text{susceptances of merged links}
\end{aligned}
\tag{3.6}
$$

3.1.3.2 Rolling horizon dispatch

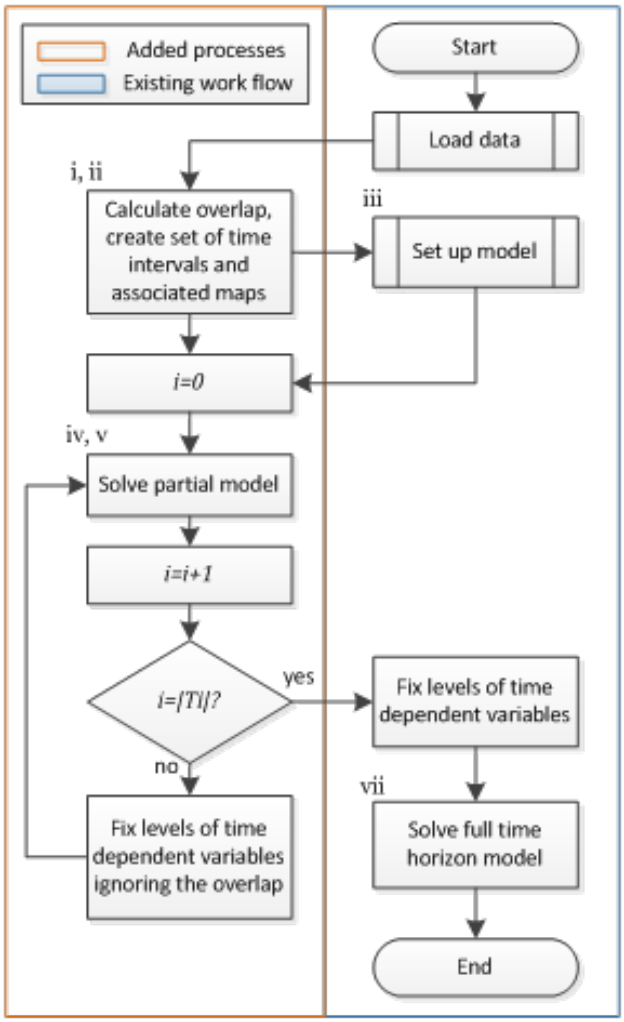
Apart from the previously described aggregation method, we also implement a rolling horizon dispatch approach into REMix. This method uses a decomposition of the original model with regard to time, where the full time horizon of 8760 time steps is divided into a number of overlapping time periods (intervals). For each of these time intervals only the hourly system operation is optimized. Accordingly, capacity expansion is not considered in the appropriate model instances. This is due to the fact that variables that are related to capacity expansion are not resolved on the temporal scale. These temporally linking elements would prevent an easy decomposition in time and thus limit the application of rolling horizon approaches to dispatch optimization problems.

The emission cap (equation 3.4) is also temporally linking and therefore requires changes compared to the native implementation of REMix. A straightforward approach is the distribution of the annual emission budget to the time intervals. In the simplest case the corresponding distribution factors are constant and calculated from the reciprocal of the number of intervals. More sophisticated distributions may take into account input data such as load and feed-in time series to define sub-annual emission caps that correspond to the residual load. However, such a distribution still does not account for regional differences. For reasons of simplicity we use the constant distribution for our implementation of the rolling horizon dispatch.

Storage facilities are only weakly temporally linking as the appropriate energy balance constraint (equation 3.1) only couples neighboring time steps. The error induced by decomposing in time is small as long as the length of time intervals is much greater than the typical energy-to-power ratio of a particular storage technology. Importantly, the overlap prevents that energy storage facilities are always fully discharged at the end of the evaluated part of a time interval to save costs. In the full time-horizon implementation of REMix this undesired effect is addressed by coupling the very last time step to the initial time step. In other words, it is enforced that the storage levels of the first and the last hour of the year are equal. However, this circular coupling is not suitable with regard to the boundaries of sub-annual time intervals.

For the rolling horizon approach this means that full discharging still appears by the end of a computed time interval, but the effect decreases the longer the overlap. However, there is a trade-off to be made with regard to the length of overlaps since they imply dispatch optimization of redundant model parts and therefore lead to greater overall computing times. Another drawback of using overlaps is also that only sequentially solving of multiple model instances is possible.

The discussed characteristics of the rolling horizon approach require a couple of modifications and extensions of the REMix source code especially with regard to the execution phases. In Figure 3.1 required adaptions are visualized.



(a) A new set T_i that represents the time intervals is defined.
(b) The number of overlapping time steps between two intervals as well as a map that assigns the time steps t to the corresponding intervals (with or without overlap) are defined. The larger the overlap the greater the number of subsequent time steps that are redundantly assigned to both the end of the i^{th} and the beginning of the $(i+1)^{th}$ interval.
(c) It must be ensured that all time dependent elements (variables and constraints) are declared over the whole set of time steps, whereas their definitions are limited to a subset of time steps that depends on the current time interval.
(d) A surrounding loop is added that iterates over the time intervals
(e) With each iteration a solve statement is executed.
(f) The values of all time dependent variables are fixed for all time steps of the current interval but not for those that belong to the overlap.
(g) To easily obtain the objective value of the full-time horizon model, a final solve is executed that considers only cost relevant equations. As all variable levels are already fixed at this stage, this final solve is not costly in terms of performance.

Figure 3.1: Flow chart of implementation of rolling horizon.

The chosen source code adjustments require a manageable amount of effort and can be seen as a processing friendly implementation since all input data is read in the beginning, whereas data is processed slice by slice. Also partial results are held in memory which facilitates an easy creation of a single output file. Established post-processing routines do not have to be changed. Nevertheless, for memory constrained ESOMs, memory friendly implementations are preferable. Data would accordingly be loaded and written to disc slice by slice. The downside of this solution is the fact that these processes must be executed multiple times which results in additional processing costs. Furthermore, the aggregation of output from different files requires further post-processing and is characterized by multiple read routines of the partial result files.

3.1.3.3 Sub-annual temporal zooming

Our implementation of the temporal zooming heuristic is an extension of the previously described rolling horizon approach that enables capacity expansion planning. For this reason, also other temporally linking elements can be treated differently. In particular, each time interval represents a sub-problem where - from a global model perspective - missing information is gathered from a temporally down-sampled full time-horizon model run.
In the case of the energy storage equation, at the boundaries of each time interval, the variables for the storage level are fixed to the resulting values of the corresponding variables of the down-sampled model.

Furthermore, for each time interval, factors that define the share of allowed annual emissions are determined with respect to the resulting emissions in the down-sampled model run. This allows a much better distribution of these actually time independent parameter values than an equal distribution as in the implementation of the rolling horizon dispatch.
Even though solving a simplified model instance causes additional costs in terms of computing time, the advantage of this approach is the independence of partial models where overlaps are no more necessary. However, as the number of parallel threads is limited on shared memory architectures, this parallelization on the conceptual layer comes at the expense of less parallelization on the technical layer, i.e. parallel threads when using the barrier algorithm. For this reason, we implement two versions of the temporal zooming approach (where I corresponds to the variable of capacity expansion introduced in equation 1.3):

1. A sequential version that is executed in the same chronological manner as the rolling horizon approach where parallelization only takes place on the solve side (Figure 3.2).
2. A parallel version that uses the grid computing facility of GAMS where a defined number of time intervals is solved in parallel. Parallelization takes place on both the model side and the solver side (Figure 3.3).

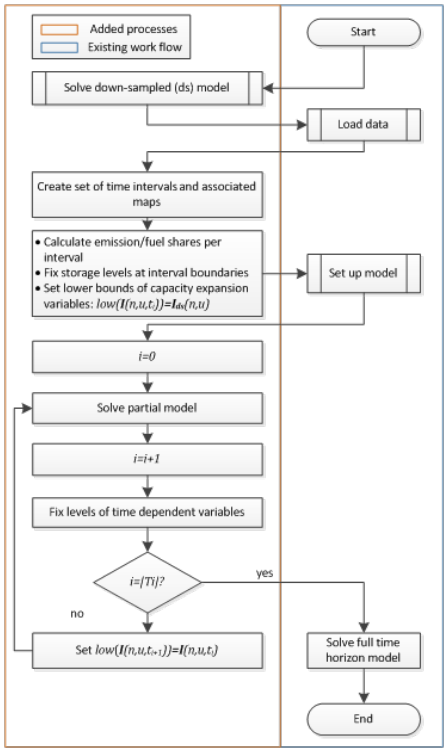


Figure 3.2: Flow chart of sequential implementation of temporal zooming.

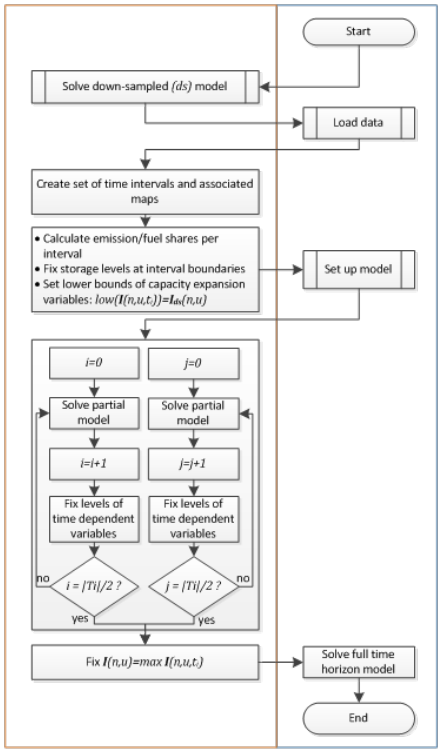


Figure 3.3: Flow chart of grid computing implementation of temporal zooming, exemplarily shown for two parallel runs.

Besides the different ways of parallelization, the two implementations also differ in the treatment of capacity expansion variables.In both cases an initial lower bound is defined with regard to the outcome of the simplified model run, in the sequential implementation, however this lower bound is raised based on the results of a particular interval and then shifted to the next interval. On the contrary, the parallel implementation determines the final values of expansion planning variables by selecting the maximum across their interval dependent counterparts.

3.1.4 Evaluation framework

3.1.4.1 Parameterization of speed-up approaches

Each of the implemented model-based speed-up approaches is characterized by parameters that influence the model performance. We refer to these parameters as SAR-parameters (speed-up approach related parameters). In this context, the challenge is to identify SAR-parameter settings that provide both an effective performance enhancement and a sufficient accuracy. We tackle this issue by performing parameter studies. The evaluated parameter value ranges are shown in Table 3.4.

In the case of aggregation the SAR-parameters are more or less equivalent to the degree of aggregation. It can be expected that there is a continuous interdependency between these parameters for aggregation and the achievable performance as well as accuracy: Increasing the degree of aggregation will reduce the required computing resources at the expense of less accuracy.
However, the implemented rolling horizon as well as the temporal zooming approaches can be tuned by changing a set of SAR-parameters (Table 3.4). Thus, the relation between speed-up approach parameterization and the evaluated indicators becomes more complex. For instance, with regard to total computing time, one can expect that there is always an optimal number of intervals since increasing the appropriate

Table 3.4: Overview of speed-up approach related parameters and value ranges to be evaluated.

Speed-up approach	Parameter	
	Name	**Evaluated range**
Spatial aggregation	number of regions (clusters)	{1, 5, 18, 50, 100, 150, 200, 250, 300, 350, 400, 450, 488}
Down-sampling	temporal resolution	{1, 2, 3, 4, 6, 8, 12, 24, 48, 168, 1095, 4380}
Rolling horizon dispatch	number of intervals	{4, 16, 52,365}
	overlap size	{1%, 2%, 4%, 10%}
Temporal zooming (sequential)	number of intervals	{4, 16, 52}
	temporal resolution of down-sampled run	{4, 8, 24}
Temporal zooming (grid computing)	number of intervals	{4, 16, 52}
	number barrier threads	{2, 4, 8, 16}
	number of parallel runs	{2, 4, 8, 16}
	temporal resolution of down-sampled run	{8, 24}

value allows faster solving of sub-models, but at the same time the computational burden for GAMS code compilation will grow.

3.1.4.2 Computational indicators

When referring to performance we always mean the computing time composed of time spent for model building and solving (solver time). The internal profiling options of GAMS are activated using the command-line option stepsum=1. All relevant information is then extracted from the logging and listing files of GAMS. The elapsed seconds listed in the last step summary represent the total wall-clock time needed for executing all processes. As in our analyses the CPLEX solver is used exclusively, the solver time represents the time consumed by CPLEX. This quantity is usually listed above the solver's report summary which also provides the information whether an optimal solution was found. As the CPLEX time reported in seconds can vary depending on the load of the computer system as well as on the used combination of software and hardware, we primarily use the deterministic number of ticks (a computer independent measure) as indicator for required computing time by the solver (IBM, 2013). The quantity we refer to as GAMS time is accordingly calculated by subtracting solver time from total wall-clock time.
An approximation for peak memory usage is also partially taken from the step summary denoted as Max heap size which represents the memory used by GAMS. An indicator for the memory use on the solver side—in the case of CPLEX's barrier algorithm—is provided by the number of equations and the logging information of the integer space required (Corporation, 2017).

3.1.4.3 Accuracy indicators

To measure the accuracy of an ESOM one could argue that all variable levels of a model instance treated by a particular speed-up approach should be compared to their counterparts of the original model. However, especially in the case of aggregation approaches the direct counterparts do not always exist. Besides the fact that the computational effort for such a comparison would be great due to the number of variables, an aggregation of the resulting differences would still be necessary to give an indication of accuracy by only a hand-full of comprehensible values. We therefore use only a selection of partially aggregated variable levels for comparison. Nevertheless, we emphasize indicators which are of practical relevance. As indicated in Table 3.1 these indicators are:

1. the objective value of the optimization problem,
2. the technology specific, temporally and spatially summed, annual power supply of generators, storage and electricity transmission,
3. the spatially summed values of added capacity for storage and electricity transmission and
4. the temporally resolved, but spatially summed storage levels of certain technologies.

3.2 Results

3.2.1 Pre-analyses and qualitative findings

3.2.1.1 Order of sets

Concerning an efficient execution of GAMS, in addition to the suggestions mentioned in section 3.1, we observed that it is always advisable to use a consistent order of sets. An illustrative example considering this issue is provided by Ramos in (Ramos, 2018). We also investigated the hypothesis that ordering the index sets from the largest cardinality to the smallest would reduce the time for the model generation. In summary, reductions of up to 40% of the GAMS generation time are observed in some cases. However, the results strongly vary between different model instances. Furthermore, the time spent for model generation can also increase depending on the used version of GAMS. From this experience we conclude that tuning the source code by using particular index orders cannot be considered as a generally effective improvement of model performance.

3.2.1.2 Sparse vs. dense

Especially with regard to the way of implementing the equations for storage energy balance and DC power flow, constraint formulations are conceivable that differ from the ones implemented in REMix (equations 3.1 to 3.3). These formulations make use of fewer variables and constraints and therefore lead to a smaller but denser coefficient matrix. Equations 3.7 and 3.8 give an impression of how such dense formulations can look like.
On the one hand, in the case of the storage energy balance equation, the alternative formulation allows that the storage level variables are no more required. On the other hand, instead of an interdependency of consecutive time steps, the power generation or consumption of each time step is linked with all of its previous pendants. This leads to strong linkages across the temporal scale especially for the balance equations that address the elements at the end of the time set. Concerning the DC power flow, 3.8 can be derived from substitution of the voltage angle and merging of equations 3.2 and 3.3. However, the resulting PTDF matrix requires a matrix inversion that leads to a dense matrix structure.

Storage energy balance (dense):

$$\begin{aligned} &\sum_{t'=t_0}^{t'=t} p_{s+}(t', n, u_s) - p_{s-}(t', n, u_s) - p_{ls}(t', n, u_s) \\ &= p_{s+}(t, n, u_s) - p_{s-}(t, n, u_s) \\ &\forall t \in T, n \in N, \forall u \in U_s, U_s \subset U \end{aligned} \tag{3.7}$$

DC power flow (dense):

$$\begin{aligned} &p_{f+}(t, l) - p_{f-}(t, l) = \sum_n PTDF(l, n)(p_{im}(t, n) - (p_{ex}(t, n) - (p_{lt}(t, n) \\ &\forall t \in T, \forall l \in L \\ &PTDF : \text{power transfer distribution factors} \end{aligned} \tag{3.8}$$

The results of our experiments with these alternative model formulations showed that, for REMix, sparse implementations are usually better in terms of model performance. While already small model instances with the dense storage balance equation are nearly unsolvable, the application of PTDF matrices for the DC power flow turns out to be useable but still less performant compared to the implementation that uses the voltage angle.
In this context, on its left y-axis, Figure 3.4 shows the computing times for two exemplary scenarios (A and B), where, transmission capacity expansion is either enabled or disabled. The size of underlying model instances ranges between 20 to 38 million variables and 9 to 24 million constraints. To give an indication of the population density of the corresponding coefficient matrices, the number of non-zeros relative to the product of the number of constraints and the number of variables is plotted on the right y-axis. Each of the resulting four model instances is solved using either the dense (triangles) or sparse (circles) DC power flow formulation. As it can be deduced from comparing the blue markers, the computing times for the PTDF-based instances are 15 to 60% greater than in the case of their sparse counterparts.

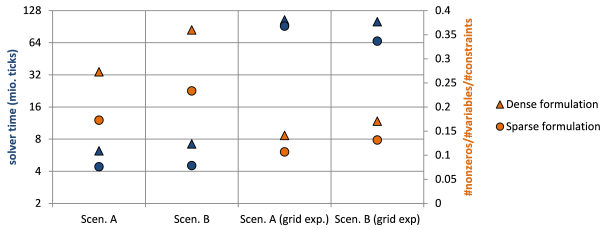


Figure 3.4: Solver time (blue) and non-zero density of the coefficient matrix (orange) for different DC power flow implementations, circles: sparse (with voltage angle), triangles: dense (with PTDF).

Due to the results of these preliminary experiments the following analyses are exclusively based on model implementations which aim for sparse constraint formulations.

3.2.1.3 Slack variables and punishment costs

A common approach to ensure the solvability of REMix, even for scenarios where the power balance equation 1.2 would be violated (e.g. by providing too small power generation potentials), is the use of slack generators. These generators do not have a technological equivalent in the reality and represent the last option to be used in the model for covering a given demand. The associated costs for power supply can be seen as the value of loss of load and thus are considerably high compared to costs caused by real technologies. However, even if very high cost values could be particularly justified by macroeconomic damage, from a model performance perspective it is advisable to set these costs in the same order of magnitude as their real counterparts.

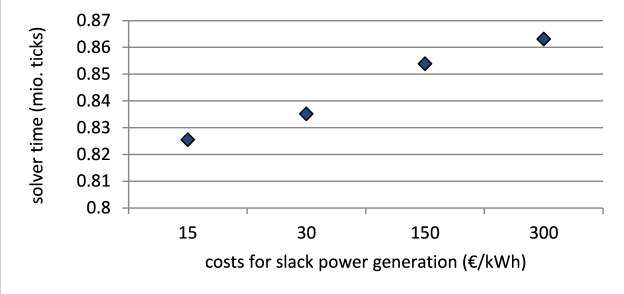


Figure 3.5: Computing time for different values for power generation by slack power generators.

Figure 3.5 therefore shows exemplary computing times of identical model instances of a relatively small size (3 Mio. variables, 2 Mio. constraints). We deliberately analyze small models to prevent the model to run

into numerical issues. The differences in the resulting solver time are exclusively caused by changing the model parameter that concerns the costs associated with slack power generation. The increasing computing time with increasing values of this parameter are due to inefficient model scaling.
Despite the fact that scaling is also automatically applied by the solver, it is advisable that in the coefficient matrix of the resulting LP, coefficients stay within a certain range of order of magnitudes. As described by (B. McCarl, 2017) the factor between the smallest and largest values should ideally be less than 1e5. Since ESOMs such as REMix consider both operational costs of almost zero (e.g. for photovoltaics) and annuities for investments into new infrastructures of several millions (e.g. large thermal units), the corresponding cost ratios are already out of the ideal range. For this reason, the cost factors for slack power generation should not expand this range. Otherwise, especially for large models, the bad scaling leads to numerical issues of the solver and at least extended computing times.

3.2.1.4 Coefficient scaling and variable bounds

Processing of input data during the generation of equations can pose problems concerning the aforementioned maximum range of coefficients. For example, this is relevant when calculating the fuel consumption based on the power generation divided by the fuel efficiency. Moreover, it is advisable to bound variables to restrict the space of possible solutions which may also lead to a better solver performance. However, finding appropriate bounds for future states of the energy system and claiming to analyze a broad range of conceivable developments implies possible contradictions.
To get a more systematic picture, in Figure 3.6, we compare a selection of model instances in three spatial resolutions with two different solver precisions. The solver precisions are labelled as "1e-5" and "Default" (1e-8) while further measures such as explicit rounding of parameters and conscious bounding of variables are varied. The idea behind rounding of input time series and efficiencies is to avoid implicit coefficients with more than five decimals. As a further step in the instance denominated as "bounded variables" we add upper bounds on most variables according to model heuristics. For instance, the power production from slack generators is limited to 10% of the exogenously given electricity demand profile. Additionally, we set upper bounds on decision variables for investments into storage and transmission capacities based on the maximum peak load and annual energy demand of the corresponding regions.

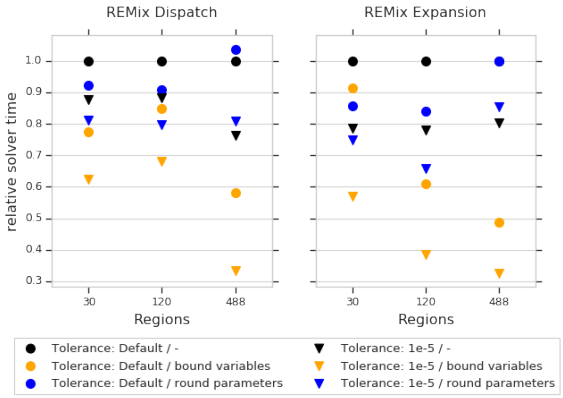


Figure 3.6: Comparison of solver times as a function of numerical properties and solver accuracy.

In Figure 3.6 the conducted comparison is shown for three differently sized instances of both the "REMix Expansion" and the "REMix Dispatch" model. The solver time is depicted relative against the number of ticks required to solve the appropriate model with default settings as presented in 3.1.2. In this context, the black circles represent the reference values at y=1.0. While for the small instances with 30 and 120 regions the gains from coefficient rounding (blue markers) seem to indicate better performance, in large scale instances the effect is inverse. For the 488 region instance there is an increase in ticks for the barrier algorithm with the presumably improved numerical properties. In contrast, the additional bounds on variables (orange markers) have a rather little impact on the small-sized instances with only a few regions, while the performance gains

for the large scale instances are significant by effectively bringing down the solver time to less than 50% compared to instances with default settings.
From the comparison of triangle and circle markers in Figure 3.6, it can be furthermore concluded, that the observed effects are independent of the solver precision. However, the possible speed-up highly depends on the general model formulation and may not apply for other solution algorithms than interior point.

3.2.2 Aggregation of individual dimensions

This section presents the behavior of performance and accuracy indicators for scaling experiments. This means that the original REMix instances ("REMix Dispatch" and "REMix Expansion") are either reduced by spatial or temporal aggregation whereas the degree of aggregation is varied. The number of aggregated regions or time steps of a respective model instance are depicted on the x-axes of the following benchmark figures (3.7 to 3.14). In this context, the degree of aggregation is simply defined by:

Degree of aggregation:

$$\begin{aligned} &a(x,v) = (1 - \frac{x(v)}{x_{REF}(v)}) \cdot 100\% \\ &\forall v \in \{spatial, temporal\} \\ &x_{REF} : \text{x-value (number of regions/time steps) of the original model instance} \end{aligned} \tag{3.9}$$

In the figures 3.7 and 3.8, the curves show computing and accuracy indicators relative to their counterparts of the original model instances. For each indicator, the reference is indicated at the greatest x-value (x_{REF} (spatial)=488 regions or x_{REF} (temporal)=8760 time steps). Accordingly, the figures are usually read from right to left. The associated absolute y-values are provided in the caption of the respective figure.

3.2.2.1 Spatial aggregation

The results for the spatial aggregation of the „REMix Dispatch" model are shown in Figure 3.7 and Figure 3.8. In the former, the computational indicators are depicted by colored curves that represent total wall-clock time, solver time, the number of constraints (equations), the number of non-zeros, and the memory consumed by GAMS as well as an approximation of the memory demand of the solver. On the right hand side, Figure 3.8 shows the accuracy indicators. Besides the objective value, the annual power generation of selected power generator groups, gas-fired and coal-fired power plants, and wind turbines, are drawn. Even though the REMix model instances consider a broader spectrum of technologies such as photovoltaics, biomass or run-of-river power plants, these technologies are omitted for the sake of clarity.

With regard to accuracy indicators, up to a degree of aggregation of about 80% (100 regions) most of the curves in Figure 3.8 show minor deviations within a range of ±5% compared to the reference at y=1.0. While the annual power generation from coal is slightly increasing with stronger aggregation, the opposite can be observed in the case of the objective value and power generation from gas turbines. Wind power and storage utilization are almost constant up to this point. However, for model instances that spatially aggregate to a degree below 100 regions, the use of storage facilities strongly increases. Compared to the reference model, deviations of more than 40% for storage are observable for highly aggregated model instances.
Considering that the number of transmission lines taken into account becomes smaller for more aggregated model instances, it can be expected that most of the effects that come with spatial aggregation stem from unconstrained power transmission. Thus, the strongest influence of this model reduction technique can be observed for the power transmission indicator where deviations greater than 25% already occur for degrees of aggregation $> 40\%$ (300 regions).
That said, the results can be interpreted as follows: The absence of power flow constraints affects the model accuracy especially when the number of aggregated regions is low and their geographical extent is comparatively large. This facilitates large central power generation units such as pumped hydro storage and coal fired power plants to extensively distribute their electricity in wide areas to the cost of less power generation from probably better sited but more expensive gas turbines.

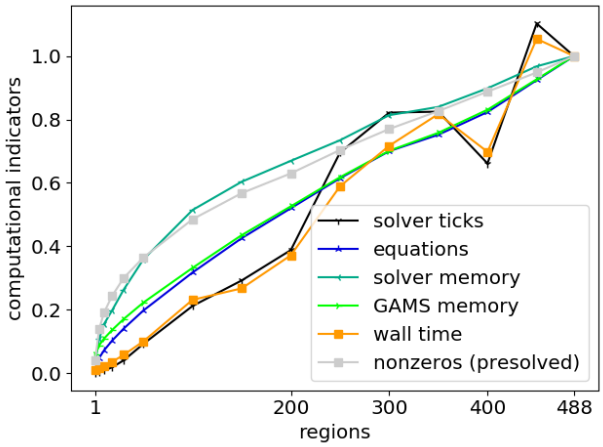


Figure 3.7: Computational indicators for spatial aggregation of the "REMix Dispatch" model. Reference model: CPLEX ticks 16.3 Mio.; Total memory 79 GB; GAMS time 0.6 h; Total wall-clock time 3.6 h.

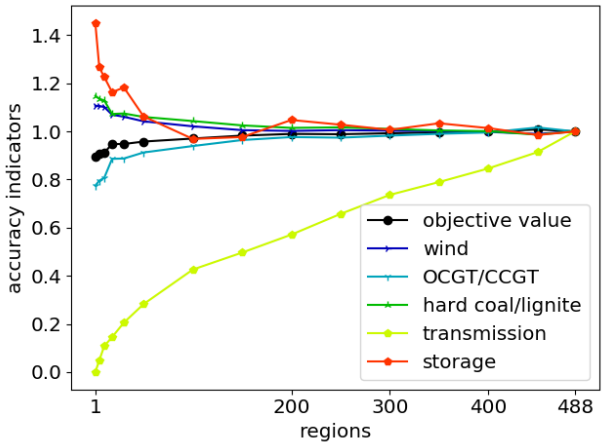


Figure 3.8: Accuracy indicators for spatial aggregation of the "REMix Dispatch" model. Reference model: Objective value 29.7 Bio €; Objective value (cleaned) 21.9 Bio €; Wind 162 TWh; Gas 174 TWh; Coal 105 TWh; Storage 4.1 TWh; Transmission 434 TWh.

If the accuracy error for 100 regions is considered to be acceptable for answering a particular research question, the reachable speed-up factor can be determined from Figure 3.7. For both the solver time (CPLEX ticks) and the total wall-clock time relative to the maximum model time of about 0.2 is observable which corresponds to a speed-up factor of nearly 5. A smaller reduction can be observed for the model size which is characterized by the number of equations as well as the RAM required by the solver (y≈0.4) and the GAMS (y≈0.3). In terms of reachable speed-up, a linear reduction of the model size by spatial aggregation usually leads to a more than linear reduction of computing time (e.g. solver time), particularly for weak aggregations. However, especially for these model instances an oscillation of the solver time can be observed which makes the estimation of reachable speed-up more uncertain.

For understanding this oscillation better, we analyzed further indicators provided in the logging and listing files as well as more content-related accuracy indicators such as the number of transmission line congestion events or slack power generation. We found that the number of non-zeros appearing within the Choleksy factorization of the barrier algorithm (reported as "total non-zeros in factor") shows a similar behavior. Nevertheless, no correlation between any of the content-related indicators and the solver time was observed. In addition, we cross-checked our results shown in Figure 3.7 and Figure 3.8 by performing the scaling experiment with different solver parameters (barrier tolerance 10^{-5}) as well as based on slightly different clustering algorithm parameters. Both led again to an oscillation of the solver time curve. Thus, we conclude that even if the accuracy indicators scale in a stable manner, especially the solver time depends on how specific nodes are assigned to clusters. Solving of the DC-OPF problem can turn out to be harder for the solver even if the number of regions is smaller than in a less spatially aggregated model instance.

As mentioned in section 3.1.2, the initial power plant portfolio of the German power system scenario for the year 2030 is slightly under-dimensioned since storage and power transmission capacities represent the state of the year 2015 ignoring planned expansion of these technologies. In addition, historical weather data of the year 2012 is used which is below the long-time average in terms of renewable power generation. As a consequence the slack power generators are active especially in the "REMix Dispatch" model instances (between 565 and 773 GWh). Total power supply derived from the objective value can thus become more expensive than in the case of "REMix Expansion" depending on the selected specific punishment costs. For this reason, we report two objective values in the caption of the figures of accuracy indictors. Firstly, the objective value of the mathematical optimization problem including costs of punishment terms. Secondly, the cleaned objective value (representing costs for total power supply) derived from assuming the same costs

for slack power generation as for operating fictitious gas turbines.
Figure 3.9 and Figure 3.10 show the performance and accuracy indicators for spatial scaling of the "REMix Expansion" model instances. Here, storage (i.e. stationary lithium-ion batteries) and transmission capacities (AC and DC lines) can be added to the system to balance power demand and generation with the installed generation capacities. In accordance to this, the accuracy indicators are extended by storage and transmission expansion. Exceptionally, only the results in this experiment are computed with extensive logging in GAMS's listing files is enabled which automatically leads to an increase of GAMS time.

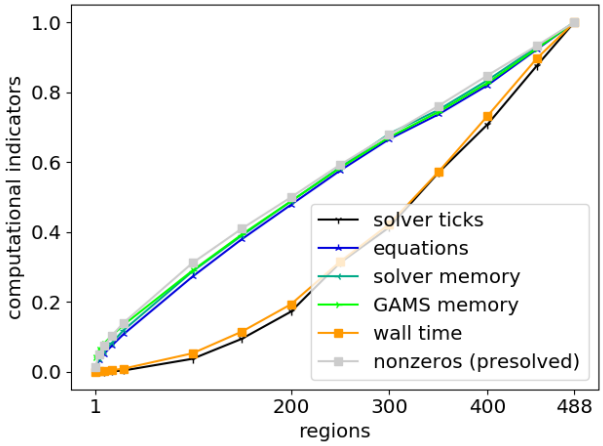


Figure 3.9: Computational indicators for spatial aggregation of the "REMix Expansion" model. Reference model (only in this experiment): CPLEX ticks 381.3 Mio.; Total memory <256 GB; GAMS time 6.6 h; Total computing time 50.9 h.

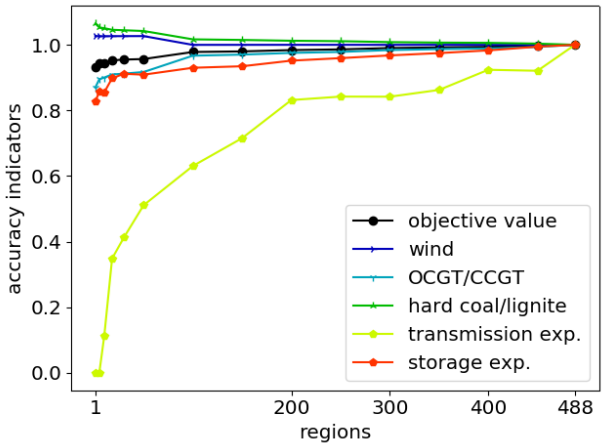


Figure 3.10: Accuracy indicators for spatial aggregation of the "REMix Expansion" model. Reference model (only in this experiment): Objective value 23.7 Bio €; Objective value (cleaned) 23.2 Bio €; Wind 175 TWh; Gas 153 TWh; Coal 115 TWh; Storage expansion 123 GWh; Transmission expansion 28.8 GW.

As reported in the caption of Figure 3.7 and Figure 3.9, enabling capacity expansion leads to a significant increase in total computing time from about 3 hours to almost 50 hours. Nevertheless, compared to the "REMix Dispatch" model instances, similarities concerning the over- or underestimation as well as the scaling behavior of the technology specific errors can be observed. For instance, capacity factors of energy storage are increasing for higher degrees of aggregation. This directly affects storage expansion which decreases with the smaller spatial resolution.
One exception are power transmission-related indicators where more significant deviations from the reference values occur, especially for degrees of aggregation $> 50\%$ (< 200 regions). On the one hand, model instances with such an aggregation even reach reductions in computing time of more than 80%. On the other hand, transmission capacity expansion already experiences significant deviations ($> 10\%$ compared to the values of the original model) for degrees of aggregation that go below 400 regions. Remarkably, this has only a minor impact on both the objective value and the generation-related accuracy indicators which is observable from the almost horizontal course of the wind, gas, coal, and storage expansion indicators in Figure 3.10.
A further similarity to the "REMix Dispatch" model is the linear scaling behavior of computational indicators corresponding to the model size as well as the super-linear scaling of the solver time. However, in Figure 3.9, the solver ticks resemble a rather exponential curve and no superposed oscillation occurs. This means that enabling the expansion of transmission (and storage) capacities leads to a rather expectable scaling behavior of the computing time: The fewer regions in a spatially aggregated model instance, the smaller the time required to solve the optimization problem. If the slope of the solver time curve is regarded as a measure of effectiveness in terms of model acceleration, it can be concluded that spatial aggregation is mainly effective for degrees up to 40%.

3.2.2.2 Temporal aggregation

The results for temporal aggregation of the „REMix Dispatch“ model are shown in Figure 3.11 and Figure 3.12. As in the case of spatial aggregation computational indicators are depicted in the figures on the left while accuracy indicators are illustrated on the right. The reference model is the same as in the spatial scenario.

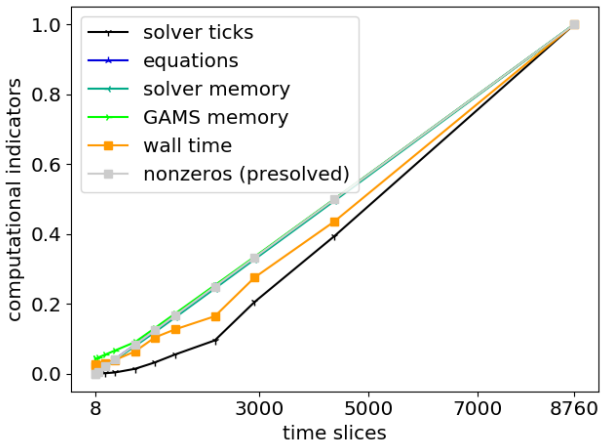


Figure 3.11: Computational indicators for temporal aggregation of the “REMix Dispatch” model.

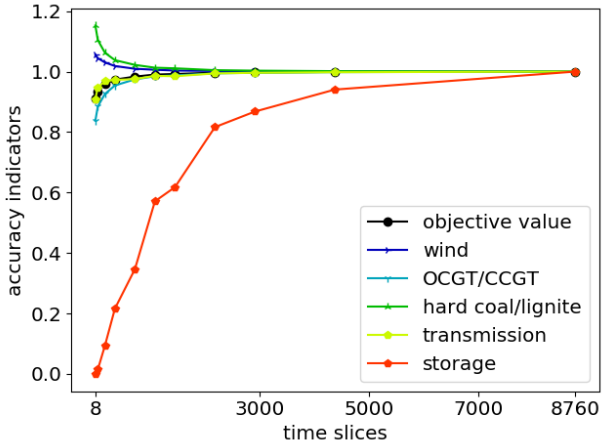


Figure 3.12: Accuracy indicators for temporal aggregation of the “REMix Dispatch” model.

In contrast to spatial aggregation, in Figure 3.12, the slope of the cost curve (objective value) appears much flatter. However, it should be noted that temporal aggregation representing two-hourly time steps already results in an aggregation factor of 50%. For this reason, all of the observed data points in Figure 3.11 and Figure 3.12 are located in the half closer to the y-axis. Concerning the solver time this already leads to speed-ups greater than factor 2. Nevertheless, it is not guaranteed that the total computing time (GAMS time + solver time) can be reduced in the same manner. This is due to the additional computing effort for aggregating hourly input data. Compared to such model instances, the greater GAMS time, e.g. in the case of 4380 time steps, results from this additional input data processing. This effect becomes significant for small model instances where the total computing time is not necessarily dominated by solver time. However, for those model instances total computing time is only a few minutes and thus represents no bottleneck. Opposed to this, for the non-aggregated “REMix Dispatch” model the ratio between solver time and GAMS time is still about factor 10.

While the objective value as well as most of the technological specific power generation indicators show an absolute error below 5% even for daily averaged time steps (365 time slices; corresponding speed-up factor: 40), significant deviations can be observed for the storage use. For this technology (i.e. pumped storage power plants) the underestimation of power generation compared to the original model is already 5% in the case of diurnal time steps. Also open cycle gas turbines (OCGT) are affected at degrees of aggregation greater than 70% (e.g. three-hourly time steps). But due to their small electricity production compared to combined cycle gas turbines (CCGT) they have only a minor impact on the slope of the corresponding curve in Figure 3.12.

Remarkably, power generation from photovoltaics (PV) is almost independent from the degree of temporal aggregation. Because its deviation is less than 0.1‰ across all analyzed model instances, the corresponding curve is not depicted in all figures concerning accuracy indicators. In other words, ignoring day-night periods has no effect on the dispatch of photovoltaics but rather on the need for storage. However, given that in the analyzed model parameterizations the amount of electricity from photovoltaics is only 10% of the annual power generation it becomes clear that PV-integration is possible at almost each point in time. Significant deviations due to temporal aggregation would therefore rather be expected in scenarios with high shares of renewables.

The results for temporal scaling behavior if expansion of storage and transmission capacities is possible can be seen in Figure 3.13 and Figure 3.14. For both figures the reference values of the original instance of "REMix Expansion" are denoted a second time. They stay the same for all following analysis with this model.
A difference compared to temporal aggregation of the "REMix Dispatch" model instances is the lager area between the green curve that represents the solver time and the blue and violet curves representing the size of a particular model instance. According to this, the reachable speed-up in terms of solver time is greater for instances with two-hourly (factor 3) or three-hourly (factor 7) time steps. On the other hand, in Figure 3.13, the slope of the solver ticks is much flatter in its lower part. By this means, going beyond degrees of aggregation of 90% (twelve-hourly time steps) appears to be less effective regarding the reachable speed-up.

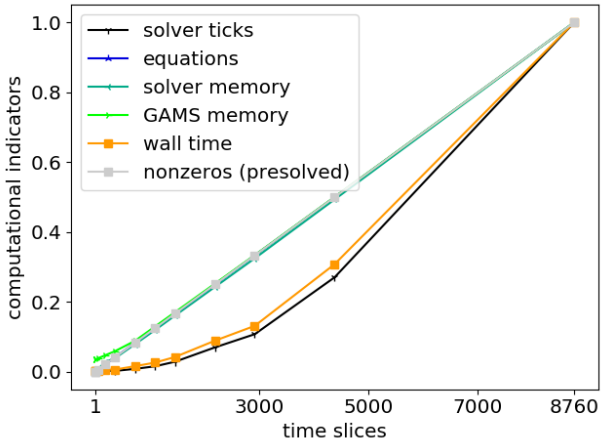


Figure 3.13: Computational indicators for temporal aggregation of the "REMix Expansion" model. Reference model: CPLEX ticks 534.3 Mio.; Total memory $>$ 256 GB; GAMS time 0.6 h; Total computing time 62.3 h.

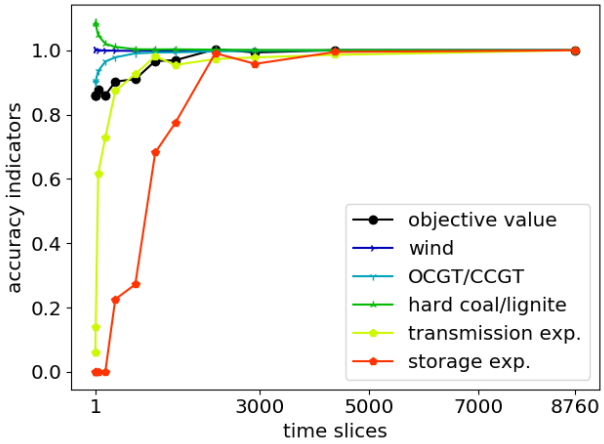


Figure 3.14: Accuracy indicators for temporal aggregation of the "REMix Expansion" model. Reference model: Objective value 22.8 Bio €; Objective value (cleaned) 22.3 Bio €; Wind 180 TWh; Gas 146 TWh; Coal 117 TWh; Storage expansion 122 GWh; Transmission expansion 29.2 GW.

Concerning the scaling behavior of model accuracy, significant errors occur for storage-related indicators. Similar to "REMix Dispatch" the annual power generation from storage facilities already decreases by 10% for two-hourly time steps. However, the storage expansion indicator stays below an error of 5% up to an aggregation factor of 75% (four-hourly time steps) while the transmission expansion indicator falls below this value at 730 time slices (twelve-hourly time steps). Therefore, it can be concluded that for observing widely accurate results for capacity expansion of transmission lines and lithium-ion batteries, four-hourly time steps appear to be sufficient, especially assessed against the background of an approximate reduction of computing time by a factor of 13.

3.2.3 Heuristic decomposition

This section presents the behavior of computational and accuracy indicators for model-based speed-up approaches that make use of heuristic decomposition techniques applied to the temporal scale of both the "REMix Dispatch" and the "REMix Expansion" model. Since the corresponding benchmark experiments vary over different parameters the appropriate figures are built up on hierarchical indices on the x-axes. However, still the relative deviations compared to the monolithically solved instances of "REMix Dispatch" and "REMix Expansion" are depicted for each of the analyzed indicators.

3.2.3.1 Rolling horizon dispatch without grid computing

The "REMix Dispatch" model is executed with the rolling horizon approach presented in section 3.1.3.2 while the interval size and the number of intervals are varied. The resulting computational and accuracy indicators are shown in Figure 3.15 and Figure 3.16. Both the settings for the overlap size and the number of intervals occur on the x-axis.

With regard to the first, it is striking that the intended behavior of total computing time is achieved – compared to the original model instance speed-up factors between two and three can be observed especially for model instances that decompose the temporal scale into more than four intervals.

In particular, with increasing numbers of time intervals the total time consumed by the solver decreases (down to less than 5% of the monolithic model) as well as the maximal memory required by the solver. On the contrary, memory required and time elapsed for executing GAMS increase by factors around 1.6 and 3.5, respectively. This is due to the additional need for generating smaller but multiple sub-model instances to be solved one after another. Even though the ratio between GAMS time and solver time is around factor four in the original model instance, when the rolling horizon approach is used, the GAMS time already dominates all model instances but those with four intervals. The total wall-clock time accordingly barely scales with the number of intervals, especially for those with more than 16 intervals.

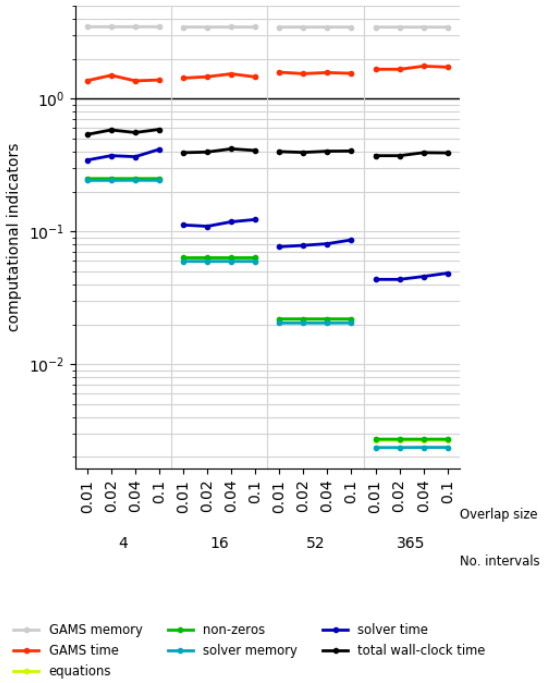


Figure 3.15: Computational indicators for rolling horizon dispatch applied to the "REMix Dispatch" model.

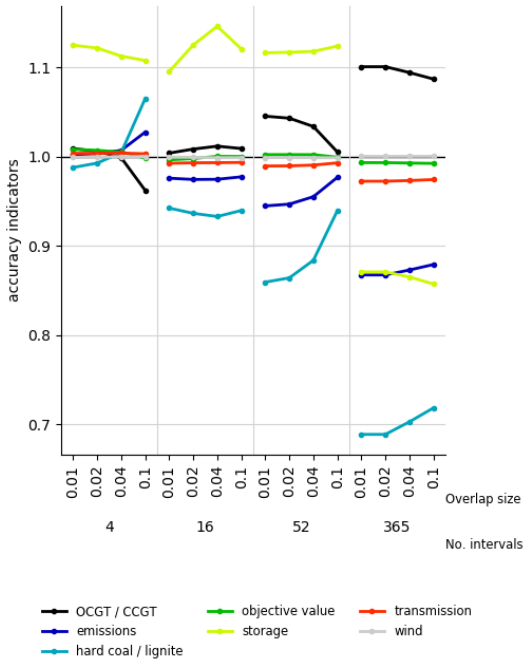


Figure 3.16: Accuracy indicators for rolling horizon dispatch applied to the "REMix Dispatch" model.

The overlap size is determined relative to the absolute length of a particular time interval. Compared to the number of intervals, it has only a minor impact on the computational indicators: As it can be expected, the greater the overlap, the more computing resources are required. This is due to the fact that all model parts that lie within the overlap are redundantly considered and thus, the total amount of equations to be solved as well as the number of non-zeros (and variables) increases for greater overlap sizes. However, even if these model size measures increase by 10% (overlap size: 0.1), the resulting total wall-clock time only experiences

changes within a range of 2% (4 intervals) to 5% (365 intervals).
Different observations can be made for the accuracy indicators where comparatively large overlaps mostly improve the accuracy of the corresponding model instances. The objective value as well as the indicators for power transport and electricity production by wind turbines have errors smaller than 3% across all investigated model instances. In this context it needs to be considered that we do not observe lower total costs than for the original model instance. Objective values smaller than 1.0 occur since slack generator costs are not considered.
The dispatch of fossil fired power plants and pumped hydro storage units shows stronger deviations. Remarkably for the latter, first overestimations of around 10% are observable for intervals numbers of four, 16 and 52. However, for intervals on a daily level, the storage accuracy indicator shows an underestimation of more than 10%.
These deviations occur, on the one hand, due to the missing circular restriction for the storage level balance that is omitted when the rolling horizon approach is applied. The appropriate constraint enforces the equality of storage levels at the beginning and at the end of the analyzed time period and thus prevents a total discharge for monolithic model instances with perfect foresight. Opposed to that, without this constraint and due to the limited foresight, (even for large overlap sizes in model instances with rolling time horizons) storage levels still tend to zero at the end of an interval ("discharge effect") and thus, average storage levels are smaller than when comparatively long time spans are considered. For example, the mean storage level of 4.6 GWh in the model instance with 365 intervals and 10% overlap is significantly smaller than in the case of four intervals with the same overlap size (20.7 GWh).
In particular, when time interval lengths are in the range of typical storage cycling periods (in the presented case daily periods for pumped hydro storage), storage charging over several energy surplus periods is not cost-efficient for an individual time interval and, in addition, the overlap size cannot be large enough to compensate the "discharge effect". Such a tipping point can be seen in Figure 3.16 for the 16-interval model instances where storage utilization first increases but decreases as soon as the overlap size changes from 4% overlap (21 hours) to 10% (55 hours).
On the other hand, the overutilization of energy storage in model instances with less than 365 time intervals stems from another effect. As shown in the upper part of Figure 3.17, significant deviations between the storage levels of the original (solid black line) and the model instance with seasonal rolling horizon time intervals (solid green line) occur mainly in the middle of the observed scenario year. Furthermore, in the case of weekly intervals (solid grey line), differences from the shape of the black curve appear over the whole time period.

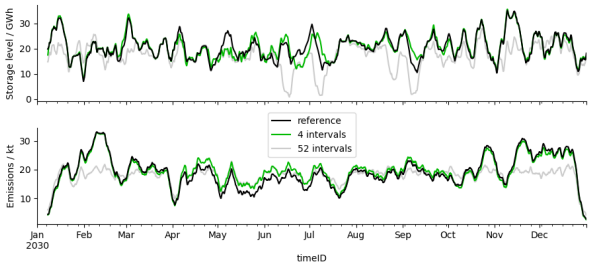


Figure 3.17: Weekly rolling average of spatially cumulated storage levels (top) and greenhouse gas emission (bottom) for two model instances with four and 52 time intervals, computed with the rolling horizon approach, compared to the corresponding results of the original "REMix Dispatch" model instance (reference).

The described behavior shows that the deviations in storage dispatch also occur independent of the intersection areas of time intervals. The reason for this is related to the treatment of the annual greenhouse gas emission budget. In the current rolling horizon implementation the annual emission budget is simply equally distributed to the individual time intervals:

Proportional emission budgets:

$$
\begin{aligned}
& m_i(i) = \frac{m}{|T_i| + |T_0(i)|} \cdot 100\% \\
& \forall i \in T_i \\
& T_0 : \text{Set of time steps that belong to overlaps}
\end{aligned}
\tag{3.10}
$$

According to equation 3.10, the resulting cumulated proportional emission budget can be greater than its annual counterpart. However, this especially applies when the absolute size of overlaps becomes large. The reason therefore is the following: Although emission produced within the overlaps are not considered for the final result, model setups exist where the proportional emission budget (that considers also emissions for the time steps within the overlap) is almost fully utilized within the time steps before the overlap begins and thus the total emission may be higher than intended. In Figure 3.16 this can be observed for the model instance with 4 intervals and 10% overlap. With regard to emissions we call this "negative overlap effect" in the following.
Apart from that, the equal distribution of allowed greenhouse gas emissions rather leads to less total emissions than in the original model instance as they are caused by fossil-fired power plants which are usually in operation in time periods with less electricity feed-in from renewable energies. Such time periods with high residual load are naturally not equally distributed. Consequently, according to the blue lines in Figure 3.16 and the grey line in the lower part of Figure 3.17, the more time intervals are considered the more restrictive the proportional emission budget. This also leads to the decrease in dispatch of coal-fired power plants observable for an increasing number of intervals in Figure 3.16.
Moreover, also the over-utilization of energy storage can be traced back to this effect: In the case of seasonal time intervals, in time spans with low residual load, the slightly higher emission potential allows a technology shift from flexible gas-fired turbines to less cost-intensive coal-fired power plants where the missing flexibility of that latter is provided by energy storage facilities ("negative interval effect"). This finally results in the deviating storage levels and higher emissions for the seasonally sliced model instance in Figure 3.17 observable in the middle of the analyzed scenario year. The opposite of this technology shift takes place when the emission limit is binding for time periods with high residual load ("positive interval effect"). In this case emission-intensive power generation of coal-fired power plants needs to be replaced by electricity production based on gas. Energy storage then comes into play to increase the capacity factor of CCGT and OCGT plants. However, as it can be seen especially for weekly time intervals in Figure 3.16, this "positive interval effect" is compensated by the "negative overlap effect".

3.2.3.2 Temporal zooming

This subsection presents the results for the sequential implementation of the temporal zooming approach applied to "REMix Expansion" model. In this regard, sequential means that multi-threading is only used on the solver level. For a better understanding, we refer to the execution of the temporally down-sampled model instance as "first execution phase" while post-sequent solving of multiple temporally decomposed models is denoted as "second execution phase". In Figure 3.18 and Figure 3.19 the resulting performance and accuracy are shown where the parameterization of these two execution phases (temporal resolution of the down-sampled model instance and the number of intervals) is varied. As for the visualization of computational indicators in case of the rolling horizon approach, the x-axes in Figure 3.18 are hierarchically labeled for the variation of two SAR-parameters (see 3.1.4.1). In this figure, computing times represent cumulative quantities while for the GAMS memory the maximum value is shown. Opposed to that, the indicators that concern the number of non-zeros, the number of equations and the memory demand by the solver show average values reported when solving each sub-model.
Given that all computational indicators scale with temporal aggregation (see section 3.2.2.2), it can be expected that the stronger the temporal aggregation of the down-sampled model instance, the less memory and computing time is required. This expectation matches the results shown in Figure 3.18. Furthermore, obvious similarities compared to the computational behavior of the rolling horizon dispatch (see 3.2.3.1) can be observed for the GAMS related indicators. Both the GAMS time and the required memory significantly

increase compared to the monolithic reference model. Nevertheless, opposed to the observations made for rolling horizon, GAMS execution times are slightly reduced for an increasing number of time intervals. The total wall-clock time, however, is significantly dominated by the solver performance as the ratio between solver time and GAMS time is greater than factor 100 for the original model and never below 1 for the model instances computed with temporal zooming. Therefore, in Figure 3.18, the shape of the black curve mirrors the shape of the dark-blue curve that depicts the solver time.
Concerning the solver time, it is striking that there is a significant minimum observable for 16 intervals. This means, even though the solver time can be reduced due to creation of smaller partial models for shorter time intervals, a tipping point exists, when this reduction cannot anymore compensate the additional computing effort for solving multiple sub-models. It becomes clearer when the super-linear scaling behavior for model instances with different numbers of time steps is taken into account. As discussed for Figure 3.11 in section **??**, the slope of the curve that represents the scaling of solver time vs. model size, is much flatter for small models (between one and 168 aggregated time steps) than for large models (between 1095 and 8760 time steps). In a temporally decomposed model with four time intervals, the length of an individual interval lies at 2190 time steps and therefore, a more than linear reduction of solver time can be expected. Opposed to that, for 52 time intervals, the time span that is covered by a single sub-model is 168 time steps. In this area of the scaling curve in Figure 3.13, a reduction of model size by factor two only causes a reduction of total computing time of less than 0.1%.

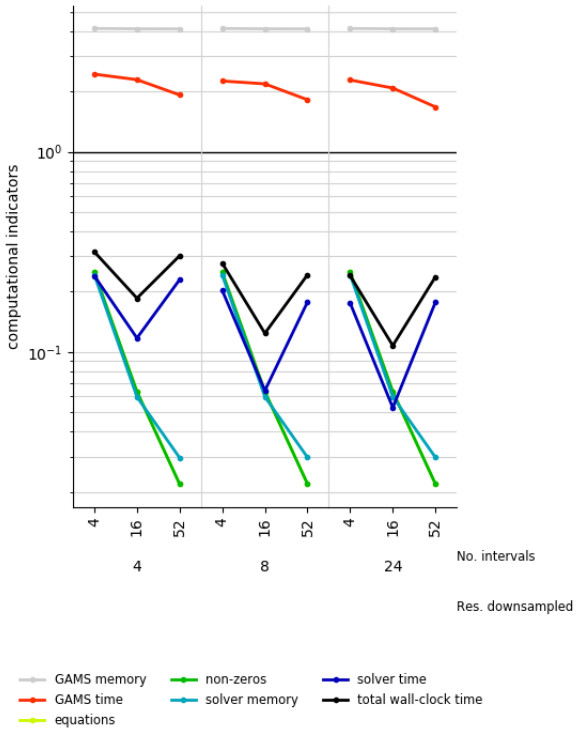


Figure 3.18: Computational indicators for sequential temporal zooming applied to the "REMix Expansion" model.

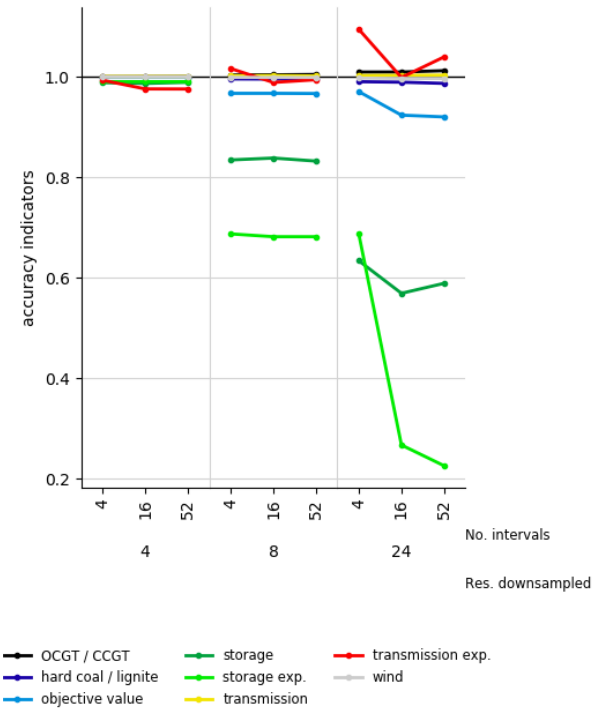


Figure 3.19: Accuracy indicators for sequential temporal zooming applied to the "REMix Expansion" model.

This decreasing effectiveness of model reduction is also the reason for the less significant increase of speed-up when comparing the total wall-clock time for different temporal resolutions in the "first execution phase". Although the model size between the instances with an eight-hourly and a 24-hourly down-sampled basis is reduced by factor three, the reduction in total computing time is around 1-3%. In contrast, when the instances with 4-hourly and 8-hourly down-sampled bases are compared, the model size is only halved, while the total wall-clock time shows a reduction of 2-6%.
In summary, it can be concluded that speed-ups around factor eight to nine can be achieved. However it needs to be considered that, due to the super-linear scaling behavior, saturation takes place in terms of further performance enhancements.
The error of accuracy indicators of the model instances that are treated by the temporal zooming approach is especially small if a temporally down-sampled model instance with four-hourly resolution is used. It stays below 3% for all accuracy indicators whereas, compared to the outcome of the original model, the largest deviation is observable for transmission expansion when more than seasonal time intervals are considered. For stronger temporal aggregations in the "first execution phase", significant underestimations of storage expansion as well as of storage utilization occur in Figure 3.19. However, while in case of an eight-hourly resolution the impact of different interval sizes is rather negligible, down-sampling on daily level results in large errors across interval sizes especially for storage expansion.
Given that the storage capacity expansion concerns lithium-ion-batteries that are usually used to smooth the daily feed-in pattern of PV plants, it becomes clear that those energy storage facilities are no longer necessary in the 24-hourly down-sampled model instance. The sudden decrease of the storage expansion for greater numbers of intervals can be accordingly explained as follows: As for the "second execution phase" lower bounds for investments into new capacities are taken from the results of the "first execution phase", this lower bound is obviously binding for models based on the eight-hourly down-sampled model instance, regardless of the number of intervals in the "second execution phase". For this reason, the storage expansion indicator is at approximately y=0.7 (light-green line). Opposed to that, in the 24-hourly case (right section of Figure 3.19), the lower bound gathered from the "first execution phase" is considerably smaller as it is depicted in the case of weekly time intervals (y=0.22). However, additional storage expansion appears particularly for seasonal time intervals (y=0.69). It can therefore be concluded that the shorter the observed time periods of a sub-model, the less attractive are investments into storage capacities.
The objective value accordingly decreases the less storage capacities are built. In this context, it is necessary to have in mind that the effective objective value still includes additional costs for slack power generation and, opposed to the cleaned costs in Figure 3.19, total costs for power supply are not automatically lower than in the original model.

3.2.3.3 Temporal zooming with grid computing

When we apply the GAMS grid computing facility to the temporal zooming approach, an additional SAR-parameter is to be considered. Although the total number of parallel threads is limited by the available processors on a shared memory machine (in the current study we use 16 threads), their utilization is variable in the grid computing case. While in the previous analyses all 16 threads are used for parallelization of the barrier algorithm, in this section, also the capability to run several GAMS models in parallel is examined. Therefore, the variation parameter "Threads", indicated on the x-axes of Figure 3.20 and Figure 3.21, distinguishes the number of runs times the number of parallel barrier threads accessible for the solver.
Opposed to the sequential implementation of temporal zooming, we do not show results for a variation of the temporal resolution used in the "first execution phase" but only for model runs based on an eight-hourly down-sampled instance. This is due to the fact that for the relation between this SAR-parameter and accuracy, it can be expected that the findings from section 3.2.3.2 also hold for benchmark experiments with temporal zooming and grid computing. Using a down-sampled model instance with eight-hourly resolution represents a compromise between desired high speed-up and acceptable loss in accuracy.
Furthermore, for efficient in-memory communication between GAMS and the solver the current analysis is conducted with the GAMS option solvelink=6. This implies that the sub-models that represent the different time intervals are solved in parallel in an asynchronous manner while partial results are hold in memory.
Depending on the combined settings of the number of intervals and the number of parallel threads, the ma-

jority of model instances cannot completely be solved in parallel. For example, in the case of 16 intervals and eight threads (and presuming almost equal solver times) it is likely that two sets of sub-models are treated after each other. First, time interval one to eight is solved within eight parallel threads and afterwards time interval nine to 16. In the following we refer to this as "serial part". However, due to the asynchronous solution process and non-equal solver times, it is not guaranteed that each thread processes exactly two sub-models.
Given that the machine independent, total solver time (reported in ticks) is not provided by the GAMS logging files, but for each time interval, we post-process the solver time indicator for the performance evaluation. For this reason, solver time is depicted in two forms in Figure 3.20: The dark blue line, denoted as "solver time single thread", represents the median calculated over the solver times of all time interval-specific sub-models. To account for the "serial part" we multiply this indicator by a factor α

Serial solve factor:

$$\alpha = \frac{|T_i|}{n_g} \tag{3.11}$$
$$n_g : \text{Number of threads for parallel runs when using grid computing}$$

to determine an approximation for the effective "solver time" (light-blue line).
In this context, a clear distinction between solver time and GAMS time is also difficult since generation (part of the GAMS time) and solving of particular sub-models are executed in parallel. Deriving an approximation for the GAMS time and normalizing it with respect to its counterpart of the original model appears accordingly less useful. The appropriate computational indicator is therefore not depicted in Figure 3.20.

Looking at the results for the total wall-clock time, a similar relation between computing time and the number of intervals can be observed as for sequential temporal zooming. Independent of the settings regarding the distribution of threads, the best performance occurs for 16 intervals. On the one hand, this is due to the decreasing effectiveness of model reduction as explained in section 3.2.3.2. On the other hand, considering the number of parallel runs n_g=2,4,8, it becomes clear, that especially instances that are decomposed into a number of intervals that represents an integer multiple of n_g are candidates for high speed-ups. In these cases the available resources (threads) can be equally utilized. This applies to all model instances with 16 time intervals but only occasionally for seasonally and weekly decomposed model instances.
The most important outcome shown in Figure 3.20 is the achievable speed-up compared to the sequential temporal-zooming approach. For 16 time intervals and 4x4 threads the resulting total wall-clock times go down to values of 10% of computing time of the original model. This additional speed-up appears due to the following effects: In contrast to a pure parallelization on the solver level, grid computing also allows to execute the model generation at least partially in parallel. Furthermore, it can be shown that computing times for implementations of the barrier algorithm in commercial solvers scale only up to 16 parallel threads (Breuer, 2019). A further reduction of computing time by stronger parallelization ($>$ 16 threads) is accordingly only beneficial if it is applied elsewhere within the computing process. Logically, the application of grid computing is especially useful, if more than 16 threads are available in total.
However, the current benchmark analysis shows that parallelization by grid computing is similarly effective as solver parallelization for comparably small numbers of threads. As depicted in Figure 3.20, different distributions of the number of parallel model runs and the number of barrier have a rather small impact on resulting solver and total wall-clock times. Also for more than 16 threads the additional value of grid competing can only poorly be demonstrated: Taking into account the results for the model instance labelled with 2x16 threads, it can be stated that despite the total number of threads is doubled, only slight improvements concerning the computing speed are achieved (speed-up factor $<$ 10.8).
Apart from that, Figure 3.21 shows the accuracy for temporal zooming with grid computing relative to the original model instance but also against the outcome of the eight hourly down-sampled model instance used computed in the "first execution phase". For storage utilization significant improvements are observable: While in the down-sampled model instance the accuracy is only 55%, it reaches levels around 82%. This

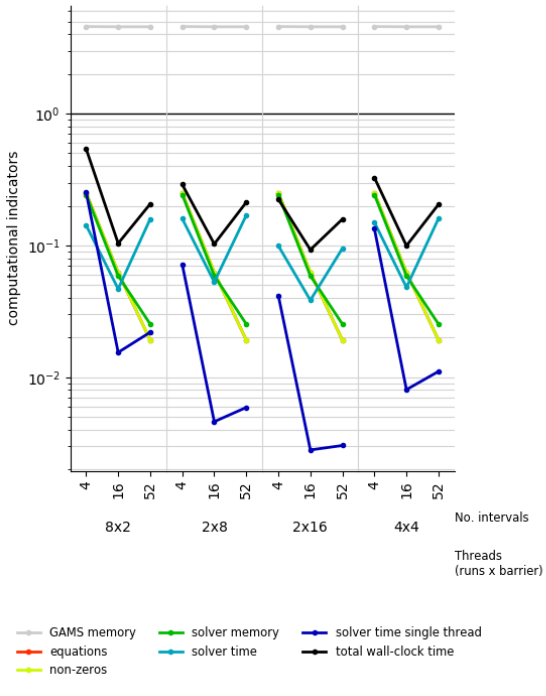


Figure 3.20: Computational indicators for temporal zooming with grid computing and eight-hourly down-sampled basis applied to the "REMix Expansion" model.

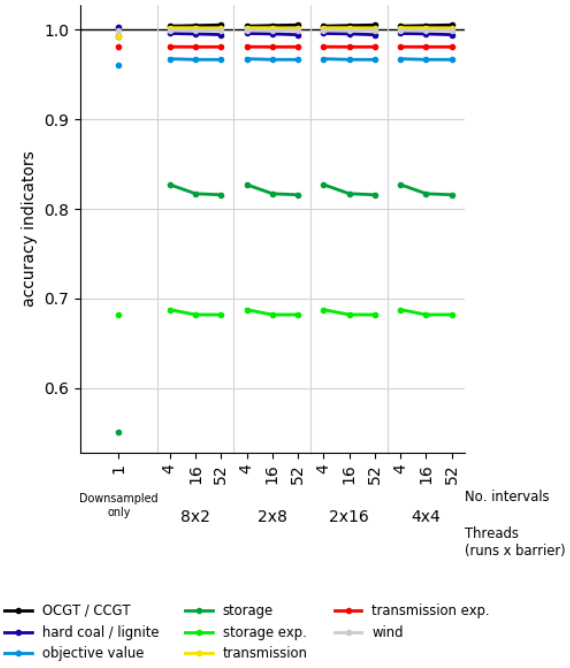


Figure 3.21: Accuracy indicators for temporal zooming with grid computing and eight-hourly down-sampled basis applied to the "REMix Expansion" model.

increase in accuracy, however, comes with the costs of less performance (for pure down-sampling on an eight-hourly basis the speed-up is around factor 37). Nevertheless, as discussed in section 3.2.2.2, the strongest errors occur with regard to storage utilization and storage capacity expansion. Other accuracy indicators, such transmission expansion, deviate less than 6% from the solution of the original model instance. If only dispatch-related indicators, such as capacity factors of wind, gas-fired or coal-fired power plants are assessed, the appropriate error is smaller than 1%. This outcome is only slightly affected when the number of intervals differs. As discussed in section 3.2.3.2 for Figure 3.19, this SAR-parameter only plays role if the "second execution phase" is based on down-sampled model instances that show stronger temporal aggregations than eight hourly time steps.

3.3 Discussion

3.3.1 Summary

With this report, we provide systematic evaluations of different approaches to improve the computing performance of applied ESOMs. Besides a number of preliminary measures such as source code reviewing and solver parameterization based on experiences gathered from former model applications, we implemented two kinds of commonly used speed-up approaches to the ESOM REMix. These are, on the one hand, spatial and temporal aggregation methods that showed effective speed-ups up to factor 10 if expansion of storage and transmission capacities is to be considered.

We showed that the majority of analyzed accuracy indicators stay within an error range of about 5 % reaching computing time reductions of 60-90% for spatial and temporal aggregation, respectively. Moreover, if particularly affected technologies such as either power transmission or storage are of secondary interest, for dispatch models speed-up factors between 4 and 20 are possible. In this context, it is important to select an appropriate aggregation approach based on the model outputs to be evaluated in particular. For example, if the competition between technologies that provide spatial or temporal flexibility to the energy system is to be examined, the presented aggregation techniques are not suited for this purpose. For model instances that consider capacity expansion, we also observed that significant speed-ups are particularly reached for low to intermediate degrees of aggregation. In contrast, strong aggregations (beyond 90 %) showed only relatively small additional improvements in computing performance.

Based on these findings, we conclude that model reduction by aggregation offers the possibility to effectively speeding-up ESOMs by at least factor two without the implication of significant losses in accuracy. In contrast, strong degrees of aggregation are less useful because speed-up gains are comparatively small while accuracy errors reach inacceptable levels ("effectiveness of model reduction").

On the other hand, we applied nested model heuristics that aim at the decomposition of the temporal scale of an ESOM. As these speed-up concepts imply manipulations on the temporal scale of an ESOM, they affect accuracy indicators that are related to modeling energy storage. The benchmark analyses of the rolling horizon approach for pure dispatch-models revealed that large overlap sizes and interval periods that cover full storage cycles are recommendable. Their additional costs with regard to computing effort are low, but may increase accuracy significantly. For the computational performance of the rolling horizon dispatch the ratio between GAMS time and solver time is crucial since only for dominating solver times, significant speed-ups around factor 2.5 could be observed for "REMix Dispatch". In this regard, it needs to be considered that "REMix Dispatch" is still a quite easy-to-solve model instance (total wall-clock time $<$ 4h). Based on our knowledge about "effectiveness of model reduction" we assume that this performance enhancement approach will be even faster for larger dispatch models.

Considerably higher speed-ups were observed for the lager "REMix Expansion" model that was treated by the temporal zooming approach. We showed that within the limited capabilities for parallelization on shared memory hardware, speed-ups of more than factor 10 were possible, especially if grid computing was used. However, besides the limitation imposed by hardware resources, the reachable performance enhancement is also restricted due to scaling behavior of very small models. This means, that additionally to the ratio between GAMS time and solver time, it needs to be considered that as soon as sub-models are reduced to a certain size, further size reductions only slightly decrease solver time (downside of "effectiveness of model

reduction"). Hence, with regard to speed-up by parallelization, it is remarkable that at first glance, many intervals appear to be more effective. However, according to the results in 3.2.3.2 and 3.2.3.3, medium sized intervals performed best.

3.3.2 Into context

Our findings, especially concerning temporal aggregation, are also in-line with those of (Stefan Pfenninger, 2017) who reports reductions of CPU time of more than 80% at three-hourly time resolution for scenarios of the ESOM Calliope applied to scenarios for the UK. With regard to accuracy, Pfenninger reports the values for capacity expansion of wind energy converters. His results show that the higher the wind penetration of a particular scenario is, the stronger the errors that occur due to temporal aggregation. However, the availability of storage technologies puts the effect of strong deviations compared to an hourly-resolved model instance into perspective.
This indicates that the scaling behavior of computing time rather depends on the model characteristics than on the composition of input parameters. Opposed to this, the scaling behavior of accuracy measures indicates a strong dependency on the parameter setup.
In contrast to the here applied "REMix Expansion" model, Calliope also considers the expansion of generation capacities. In (Stefan Pfenninger, 2017), for a scenario with extensive capacity expansion of renewables, the steep decrease of the curve of computing time for low degrees of aggregation is more pronounced than in our model instances which rather show a smooth transition to the area with a flatter slope ("effectiveness of model reduction").
For the examined heuristic decomposition techniques, our observations concerning accuracy are in-line with expectations derivable from known strengths and weaknesses occurring when differently treating the temporal scale: The down-sampled model instance allows a better approximation of capacity expansion indicators due to the consideration of the full time-horizon to be analyzed. In contrast, solving model instances with the best temporal discretization enables an accurate dispatch of available power generators (and storage units). However, as results for accuracy gains by the latter show, running a temporally decomposed model instance - when the solution for its down-sampled counterpart is known – was only beneficial for a more accurate dispatch of storage units or when the temporal resolution in the "first execution phase" was poor. In this case it needs to be considered, that for sufficient accuracy enhancements the selection of an appropriate number of intervals is crucial since errors of accuracy indicators only decrease for comparably large interval sizes.
Given that the "effectiveness of model reduction" becomes more significant when going from the comparatively easy-to-solve "REMix Dispatch" to the more complicated "REMix Expansion" model and that it is also observable for different scenarios analyzed by Pfenninger, it can be generally concluded, that already low degrees of aggregation with small accuracy errors become the more valuable the harder it is to solve a particular monolithic ESOM. This makes model speed-up approaches that are based on model reduction techniques even more attractive for application to ESOMs programed with mixed-integer variables.

3.3.3 Limitations

The claim of conducting analyses for comparably large model instances implies several challenges that only partially could be addressed. As mentioned in section 3.1, the whole benchmarking should ideally be carried out on the same computer hardware ensuring no influence on the solving process by parallel processes of other applications. However, due to a limited access to equally equipped computers, the instances of the „REMix Dispatch" model with rolling horizon were solved on the JUWELS cluster of the Juelich Supercomputing Center (first row in Table 3.2). For all of the other benchmark experiments other hardware was used (second row in Table 3.2).
Also minor bug-fixes were applied to REMix between the different benchmark experiments. One remarkable change is the indicated reduction of solver precision from 1e-8 to 1e-5 to reduce total computing times for experiments related to spatial aggregation with capacity expansion (see section 3.1.2.2) while extensive logging in GAMS's listing files was enabled. This obviously changed the ratio between GAMS time and solver time and probably led to smaller speed-ups observed for spatial aggregation with instances of "REMix Expansion".

For these reasons, speed-ups found for the individual performance enhancement approaches are not fully comparable with each other. Despite this circumstance, it can be expected that ideal conditions are also hardly achievable if speed-up approaches are used in applied studies. And still, for large models, the relation between achievable speed-up by a particular performance enhancement approach and impact on the computing time by parallel third-party processes should be negligible.
Moreover, the two selected REMix models that were used for this evaluation of speed-up approaches share many similarities with other applied ESOMs, especially if these are formulated in GAMS. However, we do not claim to provide general findings - such as the specific number of intervals to use for a rolling horizon method - that are representative for all of these models. For instance, because our results are only based on a single model parameterization, the impact of different data sets especially on accuracy indicators could not be assessed which limits the general transferability of our findings. Nevertheless, the outcome of this study provides a clear indication which speed-approaches show the highest potential for significantly reducing computing times. Furthermore, we mainly used straight-forward implementations that can still be tuned towards greater accuracy if required. This is particularly necessary if other indicators than the ones that were used in this study (mainly on an annual basis) are of interest; e.g. shadow prices.

3.3.4 Methodological improvements

In this report, we mainly focused on reachable improvements concerning the computational indicators, i.e. the required total wall-clock time. However, as all of the presented methodological approaches do not provide exact solutions of the original model instances, improvements regarding the accuracy can be considered if necessary. In the case of model reduction, a broad variety of conceivable methods to increase the accuracy of particular model outputs exists (see section 2.1). As methods such as representative time slices or more sophisticated network equivalences are more or less related to smart treatment or preprocessing of input data, the total time consumption for the overall modeling exercise will not significantly increase.
With regard to the applied rolling horizon dispatch approach, similar improvements are conceivable by using temporally aggregated data for the time steps within the overlap. The idea behind is an extension of the foresight horizon while keeping the number of redundant time steps to be considered low. For instance, for the operation of long-term storage, down-sampling of the residual load for the next annual period would be valuable to avoid the undesired effect of full discharging towards the end of an interval.
Moreover, improved estimations for emission budgets for each interval are conceivable. In the actual implementation the annual emission budget is simply equally distributed which, on the one hand, prevents the dispatch of thermal power plants particularly in points in time with high residual load. On the other hand, time intervals where sufficient renewable energy resources are available may require a smaller emission limit instead. To address this issue, it could be considered to shift unused emissions from one time interval to the next and to select a summer date as starting point for an annual model run.
Heuristic decomposition approaches such as the presented temporal zooming method offer a starting point for improvements that could go into two directions:

- Improved performance can be gained by running the independent model parts (such as the time intervals in case of grid computing presented in 0) on different computers. By this means, the drawback of being limited to memory and CPU resources of shared memory machines could be overcome. In this context, for a better coordination and utilization of available computing resources the application of workload managers such as (*Slurm Workload Manager*, 2019) would be beneficial.
- Improved accuracy can be reached by an extension to an exact decomposition approach that decomposes the temporal scale. However, this requires additional source code adaptions. For instance, in case of Benders decomposition, the distribution of emission budgets to the respective intervals needs to be realized by interval specific variables necessary to create benders cuts. Additionally, it can be expected that due to the need of an iterative execution of master and sub-problems the total computing time would significantly increase. Taking into account the best achievable speed-up of 10 of temporal zooming compared to simply solving the monolithic model, there is only a little room for improvements which may be disproportionate to the implantation effort required.

Finally, the combination of both improved performance and maintaining the accuracy requires iterative methods as well as the utilization of distributed memory computing hardware. However, effective implementations of such performance enhancement approaches require efficient communication between the processes that are executed in different computing nodes. Parallelization should therefore not only be thought at the conceptual level but also on the technical layer. This goes hand in hand with the parallelization of solvers which is realized with the PIPS-IPM++ solver (Breuer et al., 2018). This solver provides a HPC-compatible implementation of the interior point method for LPs that are characterized by linking variables and linking constraints.

3.4 Conclusions

Energy systems analysis highly depends on modeling tools such as Energy System Optimization models (ESOMs). To fulfill their purpose to provide insights into complex energy systems for decision support they need to be solvable within acceptable time spans.
For the broad spectrum of existing measures to improve the performance of ESOMs, we provided a detailed classification of conceivable approaches. Furthermore, we gave examples on easy-to-use adaptions that already improve computing performance, especially for ESOMs formulated in GAMS. These measures were accompanied by comprehensive benchmark analyses for a set of frequently applied speed-up techniques. The conducted examination included model aggregation approaches on different scales as well as strategies for heuristic decomposition. Both were applied to a spatially (488 regions) and temporally (8760 time steps) highly resolved ESOM of Germany for an energy scenario of the year 2030. While conventional computing with commercial solver software required more than two days for optimal solutions of certain model instances, selected speed-up approaches obtained sufficient solutions after less than six hours.
In particular, the novelty of this report is the systematic evaluation of a broad set of approaches assessed for an applied ESOM focusing on achievable performance improvements. This allowed statements concerning possible speed-up factors and implied accuracy losses that went far beyond existing, methodologically focused assessments of single approaches with generic model setups.
In this context, 3.5 shows the final overview of the deeply analyzed speed-up approaches of the current study. Here, the "sufficient speed-up" indicates how many times faster a model instance could be solved compared to the total time required to solve the same model in the conventional way. As our analyses emphasized model reduction and heuristic decomposition, "accuracy" was quantified by using a set of pre-defined accuracy indicators (see 3.1.4.3). In 3.5, the deviation from 100% accuracy is listed for both, the average over all assessed accuracy indicators and the accuracy indicator that showed the greatest error.

According to Table 3.5, within our evaluation framework, temporal down-sampling turned out to be the most efficient speed-up approach. The usefulness of this approach is strongly related to the "effectiveness of model reduction". In other words, the larger and more difficult to solve a particular ESOM becomes, the greater the achievable speed-up by already minor model reductions is. Taking into account that solving of linear ESOMs with mixed-integer variables is more complicated than for the model instances considered in this study, we suppose that the presented speed-up approaches are especially effective for such use cases.
As far as only specific model outcomes such as additional transmission capacities are of interest and extensive multi-threading is possible, the presented heuristic decomposition approaches with grid computing (temporal zooming) are also promising as they allow additional speed-ups without increasing loss of accuracy. Moreover, they offer the possibility for executing an ESOM on multiple shared memory computers even though parallelization is only applied to the conceptual layer of the optimization model.
Nevertheless, we showed that the appropriate gains in performance are limited depending on the size of a certain model. In this case, the down-side of "effectiveness of model reduction" comes into play: Since the idea behind decomposition is based on solving multiple reduced sub-models, such approaches reach their speed-up limit when the decrease of computing time by model reduction becomes negligible for very small sub-models.
Restrictively, the examined speed-up approaches were implemented and evaluated for a single ESOM framework. In this regard, further systematic evaluations are conceivable where variations of both input data

Table 3.5: Overview of analyzed performance enhancement approaches: observed speed-up and accuracy.

Speed-up approach	**Sufficient speed-up** (model instance)	**Accuracy** Average	Worst (indicator)
Spatial aggregation			
"REMix Dispatch"	>4 (100 regions)	>95%	>70% (power expansion)
"REMix Expansion"	>8 (150 regions)	>95%	>70% (transmission expansion)
Down-sampling			
"REMix Dispatch"	>6 (2190 time steps)	>97%	>81% (storage utilization)
"REMix Expansion"	>10 (2190 time steps)	>97%	>87% (storage utilization)
Rolling horizon dispatch	$\approx$ 2.5 (16 intervals)	>96%	>87% (storage utilization)
Temporal zooming (sequential)	>8 (1095 time steps/16 intervals)	>93%	>69% (storage expansion)
Temporal zooming (grid computing)	>10 (1095 time steps/16 intervals)	>92%	>68% (storage expansion)

and model specific source code need to be done. This especially applies to the latter because, based on our findings, we suppose that differing input data affect the accuracy of an ESOM rather than the computing performance.

In conclusion, the capability to solve very large ESOMs much faster is a pre-condition for best-practice studies in the field of energy systems analysis. Rather than spending time on solving models only for a hand full of scenarios and parameter sets, broad parameter scans become possible for which plenty of model solutions are required. In this manner, the application of effective speed-up approaches highly contributes to the generation of robust and well-founded model-based analyses for the development of decarbonization strategies of the energy system.

Part II

Technical Performance Enhancement

Chapter 4

Overview of technical performance enhancement

Technical energy system performance enhancement refers to massively parallel computing power, either on institute-level clusters or in high performance computing centers. The architectures of high-performance computers require the models to be as highly parallelized as possible, i.e. structured in individual, independent sub-problems. Therefore, this performance enhancement strategy requires not only the high-performance computing hardware but a specialized solution algorithm and an adapted model formulation at the same time. These three components of technical performance enhancement are described in this chapter.

4.1 A hand-tailored parallel algorithm

While commercial generic LP and MIP solvers are often extremely powerful, they usually cannot run (efficiently) on distributed systems. Moreover, for special problems (such as perhaps most famously the traveling salesman problem (Cook, 2018)), hand-tailored solvers may significantly outperform general-purpose commercial MIP software. As large ESM instances of the BEAM-ME project have proven to be intractable within a time frame of several hours for commercial solvers, much effort has been spent on developing specialized parallel algorithms that exploit the structure of the ESM instances at hand.

4.1.1 Exploiting the problem structure within an interior-point algorithm

Since interior-point methods usually prove to be more successful for large problems, in particular for ESM, see Section 1.2, this method was chosen for the LPs in the BEAM-ME project. Mathematically, a salient characteristic of these LPs is their block-diagonal structure with both linking constraints and linking variables, as depicted below:

$$
\begin{array}{lllll}
\min & c^T x & & & \\
\text{s.t.} & T_0 x_0 & & & = h_0 \\
& T_1 x_0 + W_1 x_1 & & & = h_1 \\
& T_2 x_0 + & W_2 x_2 & & = h_2 \\
& \vdots & & \ddots & \vdots \\
& T_N x_0 + & & W_N x_N & = h_N \\
& F_0 x_0 + F_1 x_1 + & F_2 x_2 & \cdots \quad F_N x_N & = h_{N+1},
\end{array}
\tag{4.1}
$$

with $x = (x_0, x_1, ..., x_N)$. The linking variables are represented by the vector x_0, whereas the linking constraints are described by the matrices $F_0, ..., F_N$, and the vector h_{N+1}. The approach to solve this LP is based on the parallel interior-point solver PIPS-IPM (Petra, Schenk, & Anitescu, 2014), that was developed for solving stochastic linear programs. It has already successfully been applied to solve large-scale problems in massively parallel environments. The form of its designated problem class also exhibits block-diagonal structures, although only with linking variables and without linking constraints. In this way, PIPS-IPM in its original form cannot handle problems of the form shown above. Within the BEAM-ME project PIPS-IPM has been extended in order to handle LPs with both linking constraints and linking variables. The resulting solver is called PIPS-IPM++ or short PIPS++. The exact mathematical details are beyond the scope of this guide, but the pivotal ingredient of the algorithm is the parallelization of the linear system that has to be solved in each iteration of an interior point algorithm. Below, a short overview over the most impactful extensions and their schematic mathematical description are given.

PIPS-IPM and its new extension make use of the Message Passing Interface (MPI) for communication between their (parallel) processes, which in the following will be referred to as *MPI-processes*. Without going into too much technical detail, an important feature is that the whole LP can be distributed among the MPI-processes, with no process needing to store the entire problem. This allows tackling problems that are too large to even be stored in the main memory of a single desktop machine. The main principle is that for each index $i \in \{0, 1, ..., N\}$ all x_i, h_i, T_i, and W_i (for $i > 0$) need to be directly available to only one MPI-process; h_{N+1} needs to be assigned to the MPI-process handling $i = 0$. Moreover, each MPI-process needs access to the current value of x_0. Below, the distribution is exemplified for the case where the information to both $i = 0$ and $i = 1$ is being assigned to the same MPI-process (in gray). The vectors and matrices that need to be processed together are marked in gray, black, and bold, respectively.

$$
\begin{array}{lllllll}
\min & c_0^T x_0 + c_1^T x_1 & + & c_2^T x_2 + & \cdots & \mathbf{c_N^T x_N} & \\
\text{s.t.} & T_0 x_0 & & & & & = h_0 \\
& T_1 x_0 + W_1 x_1 & & & & & = h_1 \\
& T_2 x_0 + & & W_2 x_2 & & & = h_2 \\
& \vdots & & & \ddots & & \vdots \\
& \mathbf{T_N x_0} + & & & & \mathbf{W_N x_N} & = \mathbf{h_N} \\
& F_0 x_0 + F_1 x_1 & + & F_2 x_2 & \cdots & \mathbf{F_N x_N} & = h_{N+1}
\end{array}
$$

The maximum of MPI-processes that can be used is N; in the opposite border case the whole LP is assigned to a single MPI-process.

4.1.2 Improving performance and scalability

The prototype of PIPS++ showed deficits in both, convergence and run time: While it was possible to solve some ESMs to optimality, for others the algorithm could not find an optimal solution—it also could not outperform the interior point algorithm of the commercial solver CPLEX, which was used as the reference (ran with the empirically most profitable number of threads on a large shared-memory machine). Several steps have been taken to improve performance and convergence.

4.1.2.1 Solving the Schur complement

To begin with, interior point algorithms rely on the successive solution of LPs. In the extended solver, these systems have the form

$$
\begin{bmatrix}
K_1 & & & B_1 \\
& \ddots & & \vdots \\
& & K_N & B_N \\
B_1^T & \cdots & B_N^T & K_0
\end{bmatrix}
\begin{bmatrix}
\Delta z_1 \\ \vdots \\ \Delta z_N \\ \Delta z_0
\end{bmatrix}
=
\begin{bmatrix}
b_1 \\ \vdots \\ b_N \\ b_0
\end{bmatrix},
\tag{4.2}
$$

with matrices K_i, B_i, and some right-hand side vector. These matrices each are composed out of the submatrices shown in 4.1 and we do not go into detail about their structure here. The distribution of system 4.2 follows the one described earlier and thus the solving of the LP can be done (mostly) in parallel. The solver applies a Schur complement approach to the system which decomposes into the following procedure:

1. Multiply each row $i = 1, ..., N$ of (4.2) by $-B_i^T K_i^{-1}$.
2. Sum up all rows.
3. Solve $\big(K_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} B_i\big)\Delta z_0 = b_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} b_i$.
4. For each row $i = 1, ..., N$ insert Δz_0 and compute Δz_i.

The matrix

$$C = K_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} B_i, \tag{4.3}$$

is called the Schur complement of the above system and its computation, or rather, the solution of the associated linear system is most crucial to the interior point algorithm. While most of the steps can be done locally on each MPI-process, the formation of the actual Schur complement and the solution of the resulting linear system would require storing the matrix on a single process. This poses a problem, since for many energy system models there are more than 300 000 linking constraints and variables, making it intractable to compute and store the Schur complement as a dense matrix, the common procedure for the depicted algorithm. However, in many linear energy system models, the majority of linking constraints and variables are local, meaning, they only connect two different building blocks i, j of the original problem. These structures lead to an increased sparsity of the resulting global Schur complement. One of the major improvements implemented is the exploitation of these local linking structures. Knowing beforehand about the locality of certain linking constraints and variables allows the prediction of the non-zero pattern in the resulting Schur complement. This reduces the amount of storage needed for the matrix and also enables one to use sparse LP solvers for the solution of the Schur complement. Both advantages increase the performance of the original version of PIPS greatly allowing it to efficiently solve systems with larger linking parts. For a detailed description of the (actually, even more sophisticated) implemented mechanisms for exploiting the locality of linking constraints and variables in the extended solver refer to (Rehfeldt et al., 2019).

Despite the previously mentioned improvements, a high number of linking constraints and variables can still slow down the solution process of the algorithms significantly. This is because factorization and solution of the global Schur complement can still be prohibitively time-consuming. One possible remedy is to not compute the Schur complement explicitly but to use an iterative approach, e.g. a Krylov subspace method (Saad, 2003). These methods only require matrix-vector products with the system matrix but often depend highly on the effective preconditioning of the linear system. For this purpose, PIPS-IPM has been extended by a distributed preconditioner, details of which can be found in (Rehfeldt et al., 2019). The implementation of such a preconditioner in the context of interior-point methods being quite a novelty, one achieved significant performance improvements.

4.1.2.2 Improving the solution algorithm

Most of the algorithmic efforts described above have concentrated on efficiently solving the linear system arising from an interior-point algorithm. However, also the interior point algorithm itself is of high importance to achieve fast and robust convergence. To improve robustness, the originally implemented predictor-corrector algorithm of Mehrotra has been replaced by the multiple centrality correctors scheme of Colombo and Gondzio (Colombo & Gondzio, 2008). Moreover, since PIPS-IPM has been developed to solve quadratic problems, the same step length (central part of an interior-point algorithm) for primal and dual variables has been used originally. As the project is only concerned with LPs, different primal and dual step lengths, see e.g. (Wright, 1997), have been implemented into PIPS++ to achieve faster convergence.

4.1.2.3 Preconditioning

Two additional points in implementing robust interior-point algorithms that are not directly tied to the algorithm itself are scaling and presolving. Both preconditioning techniques modify the original system matrix in a way that makes the application of an interior point method to solving it more reliable. For the next paragraphs we consider the LP in a slightly different and simplified form than (4.1), namely with explicit variable bounds:

$$\min\{c^T x : Ax = b,\ \ell \leq x \leq u\}. \tag{4.4}$$

This notation allows for a better demonstration of the following scaling and presolving techniques. Scaling of (4.4) can be described by means of two diagonal matrices $R = (r_{i,j})$ and $C = (c_{i,j})$ with positive diagonal elements. The diagonals correspond to the row and column scaling factors respectively. Defining

$$\tilde{A} = RAC, \quad \tilde{b} = Rb, \quad \tilde{c} = Cc, \quad \tilde{\ell} = C^{-1}\ell, \text{ and } \tilde{u} = C^{-1}u$$

one obtains the scaled linear program

$$\min\{\tilde{c}^T x : \tilde{A}x = \tilde{b},\ \tilde{\ell} \leq x \leq \tilde{u}\}. \tag{4.5}$$

Each solution $\tilde{x}$ to (4.5) corresponds to a solution $x = C\tilde{x}$ to (4.4) with the same objective value. PIPS-IPM has been extended by geometric scaling, equilibrium scaling, and a combination of both (Elble & Sahinidis, 2012). While equilibrium scaling divides all coefficients in each nonzero row and column of the constraint matrix by the absolute largest entry within this vector, geometric scaling uses a simplified geometric mean of the absolute vector entries as divisor: For each column A_j of the constraint matrix A the divisor is $\sqrt{\max_{i:a_{ij}\neq 0} |a_{ij}| \cdot \min_{i:a_{ij}\neq 0} |a_{ij}|}$, for each row a_i. it is $\sqrt{\max_{j:a_{ij}\neq 0} |a_{ij}| \cdot \min_{j:a_{ij}\neq 0} |a_{ij}|}$. Geometric scaling is computationally more expensive than equilibrium scaling, since it is applied iteratively (up to 10 times in the extended solver); equilibrium scaling on the other hand always converges in one step. However, the arrowhead structure allows to perform all scaling methods efficiently in parallel. Therefore, the scaling run times are neglectable, and geometric scaling is used as default.

4.1.2.4 Presolving the problem

As mentioned, another important ingredient of state-of-the-art linear programming solvers is presolving (Achterberg, Bixby, Gu, Rothberg, & Weninger, 2019). LPs resulting from a modeled application often contain redundant information. Thus it is advisable to apply presolving routines to the LP to eliminate redundant parts. Presolving techniques for LPs aim to reduce the number of variables, constraints, and non-zeroes while keeping the presolved and original problem equivalent (similar to 4.5 and the unscaled problem 4.4). Currently, PIPS++ newly implements four different presolving methods. Each incorporates one or more of the techniques described in (Achterberg et al., 2016; Andersen & Andersen, 1995; Gondzio, 1997): singleton row elimination, bound tightening, parallel and nearly parallel row detection, and a few methods summarized under the term model cleanup. The latter includes the detection of redundant rows as well as the elimination of negligibly small entries from the constraint matrix.

The presolving methods are executed consecutively in the order listed above. Model cleanup is additionally called at the beginning of the presolving. A presolving routine can apply certain reductions to the LP: deletion of a row or column, deletion of a system entry, modification of variable bounds and the left- and right-hand side, and modification of objective function coefficients. These reductions are applied as equivalence operations not changing the means of the linear program. This allows it to recover an original solution (a solution of the original system matrix) from the solution computed by the solver in a so-called postsolve step. It is important to note, that none of the presolving techniques described above are allowed to alter the general block structure of the system since this would lead to the non-applicability of the structure-specific solution method. Thus, the set of implemented methods is highly tailored to the general structure of the problem (Gleixner, Kempke, Koch, Rehfeldt, & Uslu, 2019).

4.1.3 Further improvements

While the work described above already yields notable computational improvements, there are several paths for further development of PIPS-IPM++. A natural one is the implementation of additional presolving

methods. Promising candidates are substitution of variables and elimination of linearly dependent rows. To improve robustness, also more aggressive scaling methods could be implemented, as well as the homogeneous self-dual interior-point method (Vanderbei, 2008).

Yet another extension, that is currently being implemented, is a hierarchical approach, which splits the Schur complement decomposition (and thus also the Schur complement) into several layers to handle energy system models with even stronger linkage.

4.2 High Performance Computing

The first part of this chapter will give a brief overview of common HPC architectures. A detailed overview of HPC is provided, for example, by Hager et al. (Hager & Wellein, 2016) and Sterling et al. (Sterling, Anderson, & Brodowicz, 2018). Further sources with additional information are referenced in the following sections. The second part will briefly describe the supercomputers at HLRS ans JSC that were used during the BEAM-ME project.

4.2.1 High performance computing architectures

The theoretical concept of modern computers has been developed by Turing in 1936 (Turing, n.d.) and the implementation of this concept was named a stored-program digital computer (see figure 4.1). The program is a stream of instructions and is stored together with the data in the same memory. The control unit reads the instructions and the arithmetic logic unit reads the data from memory and performs the actual computations. A well know bottleneck of this concept, the von Neumann bottleneck, is the influence of the memory interface speed on the computational performance. If the instructions and data cannot be transfered fast enough to the Central Processing Unit (CPU), the CPU cannot continue with the computations. This inherently sequential workflow can only process one instruction on one data element in one cycle. According to Flynn's taxonomy, this architectural design belongs to the group "Single Instruction Single Data" (SISD). The following section lists several modifications and extension to improve the performance of this architecture and to enable parallelism.

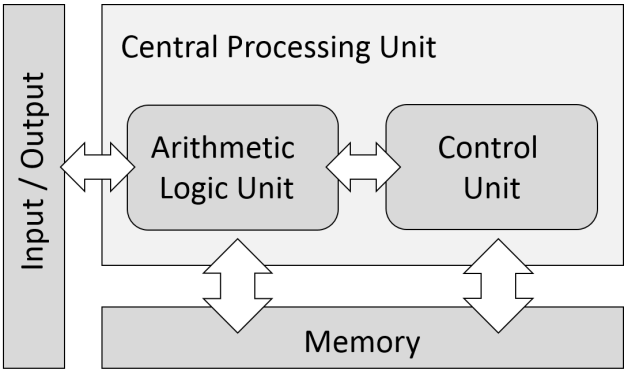


Figure 4.1: stored-program digital computer

Around 1970 Gordon Moore postulated that the number of transistors on a chip doubles every two years (see figure 4.2). As it turned out that concomitant with doubling of the number of transistors the compute performance has been growing equivalently, this was called *Moore's law* and is still valid. Along with this development, the clock frequency increased as well. This engineering progress has led to further innovations which includes among others:

- Pipelining: Instruction Level Parallelism (ILP)
- Superscalarity
- Hyper-Threading / Simultaneous Multithreading (SMT)
- Out-of-oder execution
- Single Instruction Multiple Data (SIMD); Multiple Instruction Multiple Data (MIMD)
- Memory hierarchy - Caches: Registers, prefetching, coherence

In the early 2000s there was a change in the hardware development because the increase of transistors and clock frequency have resulted in higher temperatures and power consumptions so that cooling became a

major issue. This was the motivation to design multicore CPUs with multiple cores in a CPU instead of increasing the frequency while keeping the same power envelope as before (see figure 4.2).

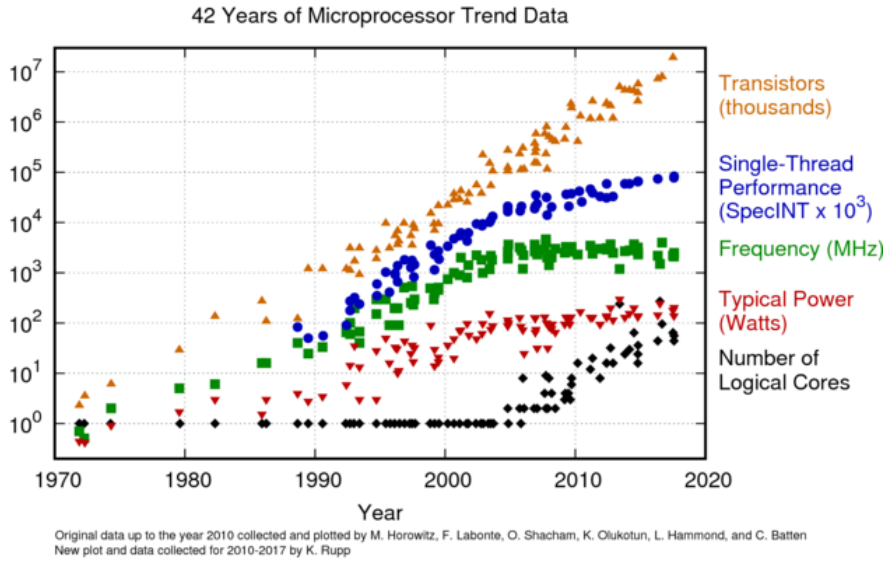


Figure 4.2: 42 years processor trend

Modern supercomputers are usually equipped with two or more multicore CPUs per compute node. One compute node is a shared memory system. But the strength of an HPC system is coming from the fast network which connects the different compute nodes so that applications are able to communicate and synchronize their workflow. The most important performance indicators of such an interconnect are bandwidth and latency.

A supercomputer is a distributed-shared memory system. To use such a hardware efficiently, parallel programming languages like the widely used MPI and OpenMP (OMP) are neccessary. Apart from the described processor archictures, other computing hardware exists, e.g. GPUs (graphics processing units), FPGAs (field-programmable gate arrays) and vector processors. Specialized parallel file systems are available to deal with huge demand on storage capacity and to reduce the influence of the Input and Output (I/O) bottleneck.

Although the following section Distributed-Shared-Memory achitecture and Message Passing Interface is not mandatory for the users who want to solve linear optimization models on an HPC system using the strategy developed in BEAM-ME, we believe that they serve the general understanding of the HPC architecture and its advantages, disadvantages and future development. It also helps readers to understand terms in HPC that may appear in other areas of science but with a different meaning, such as a "node" or "load balancing".

4.2.2 Distributed-Shared-Memory achitecture and Message Passing Interface

The strength of HPC lies in the connection of numerous computing nodes via a high-performance connection. Therefore, it is not surprising that algorihm parallelisation and network performance are the keys to application efficiency on HPC systems. In the following, the most important features of distributed systems and their interrelationship to parallel applications are described.

A process is a serial program in execution on one core and has its own virtual address space, which is mapped to the physical main memory of a compute node. If one wants to use multiple cores for a computation, threads are used. The only difference to a process is that multiple threads of same process have the same virtual address space. OMP is an application program interface (API) that automatically implements shared memory parallelism.[1] It has become a successfull parallelization model and is the de facto standard for

[1] OMP is used to fork additional threads to carry out the work enclosed in a OMP construct in parallel.

shared-memory parallel programming (OpenMP, 2018).

In addition to threads, one also uses Message Passing Interface (MPI) processes to code and run a parallel application on multiple cores. The additional effort compared to OpenMP is that the data must be distributed accross the processes, because each of MPI processes has its own address space. This method is called domain decomposition. The advantage is that an application can run on multiple compute nodes. If two processes are running on two different compute nodes, the data is copied over a network. If two processes are running on the same compute node, the data is copied inside the processor or via QPI links. In both cases, the same MPI API are used. Only the performance can differ greatly. Although this enables distributed parallel programs to be written exclusively with MPI, the Hybrid Programming Paradigm is becoming more and more important. The reason for this is that the number of cores per computer node is constantly increasing and thus memory interfaces and QPI links becomes a bootleneck. The hybrid programming model MPI/OpenMP is also used in PIPS-IPM++.

The domain decomposition consists of splitting the overall computational domain into as many tasks as desired, and assigning each subdomain to an MPI process, which can then be solved at the same time. Figure 4.3 shows a distribution of a two dimensional domain to four MPI processes. If the parallel algorithm needs the data from the neighboring subdomains to be exchanged, the halo data is sent via the so-called MPI messages. The MPI standard (MPI-Forum, 2015) defines an interface for the message exchange between the processes. The standardization in this case is very important because the high performance networks can be equipped very differently depending on the model and manufacturer.

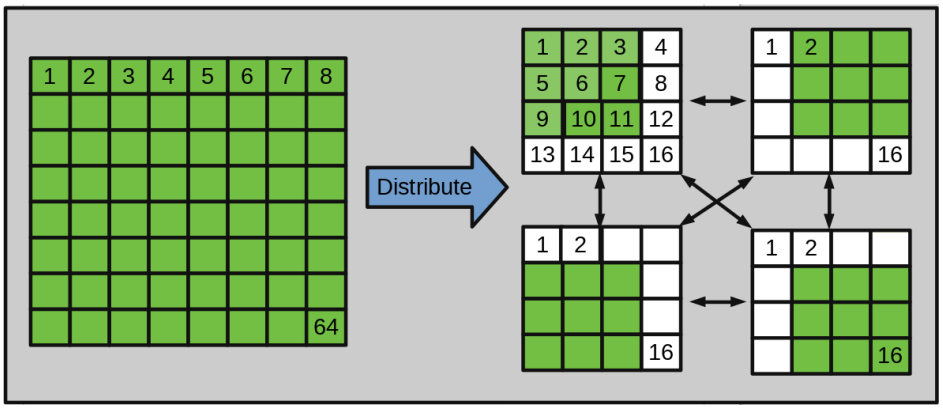


Figure 4.3: Domain decomposition in distributed memory

In the example shown, the amount of work is evenly distributed between the MPI processes. This makes load balancing ideal. An MPI/OpenMP program differs from a pure MPI program in that each subdomain can be processed with several threads instead of one.[2]

4.2.2.0.1 Load balancing: In real world applications it is often almost impossible to divide the area into exactly the same subdomains. Since the parallel solvers are often iterative algorithms, the parallel processes must wait in each iteration until all processes have completed their calculation on the subdomain. If the decomposition is inappropriate, all processes must wait for one that has been assigned to the largest subdomain. In addition, not all serial algorithms can be ideally parallelized. A part of the processing has to be done with a single process, while the other MPI processes are waiting.

4.2.2.0.2 Amdahl's law: Amdahl's law shows how much a computation can be sped up by running part of it in parallel. The speedup $Speedup(p)$ is defined as the ratio of the serial runtime of a sequential algorithm for solving a problem to the time taken by a parallel algorithm to solve the same problem with p

[2]In HPC one starts one thread per core.

processes. Since the parallel algorithms need the communication between the processes, Amdahl's Law can be further extended with the communication time T_{comm}. If we assume that the communications portion γ increases linearly with the number of processors, equation 4.6 predicts the theoretical maximum speedup for a parallel program.

$$
\begin{aligned}
T_{tot} &= T_{ser} + T_{comm} + T_{par}; \\
Speedup(p) &= \frac{T_{tot}}{T_{ser} + \gamma * p + \frac{T_{par}}{p}}; \\
\\
T_{tot} &\quad \textit{- Total time to solve a problem;} \\
T_{ser} &\quad \textit{- Time of not parallizable serial part of algorithmus;} \\
T_{comm} &\quad \textit{- Communication time;} \\
T_{par} &\quad \textit{- Time of paralliezable part of algorithmus;} \\
p &\quad \textit{- Number of parallel processes;} \\
comm &\quad \textit{- Communication share per process;}
\end{aligned}
\tag{4.6}
$$

Figure 4.4 on the left side illustrates the limitation of the speedup without taking communication into account. The figure on the right side shows the speedup considering the communication time. As you can be see, If too many processes are used, the solution time may even increase.

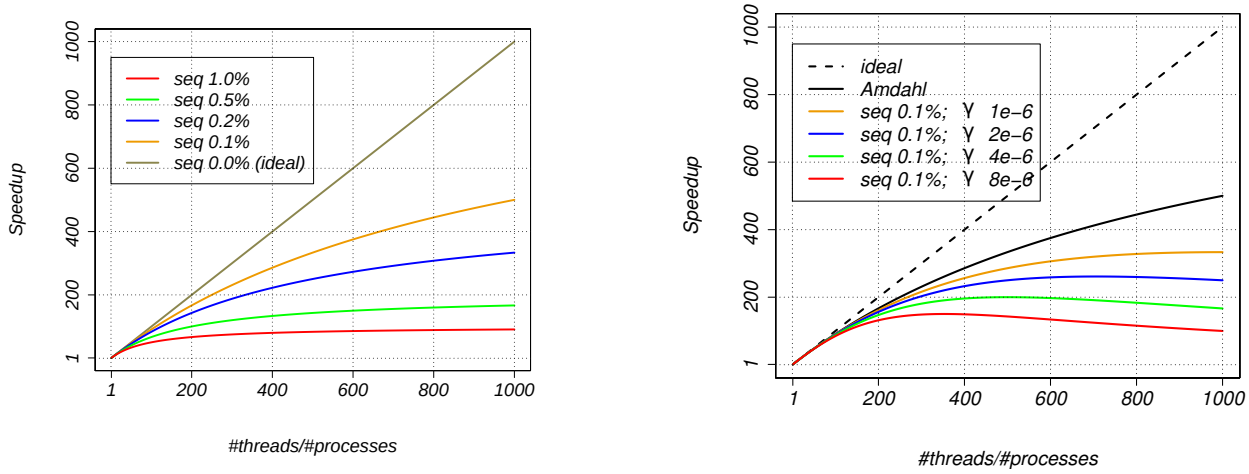


Figure 4.4: Amdahl's law without (left) and with consideration of the communication (right)

4.2.2.0.3 PIPS-IPM++ and Amdahl's law: Figure 4.5 illustrates a timeline for three of 512 MPI processes for two iterations of the interior-point method, which is implemented in PIPS-IPM++. The diagram shows the exclusive time per function group, namely MPI and USER (labeled as Application in the figure). The red color shows the MPI part and the green color represents the rest of the computation (USER). The yellow color indicates a particular function that is not MPI parallelized and is a part of the USER group. The corresponding function solves the schur complement of the LP system (see section 5 for more details). The MPI part contains not only the communication time, but also the waiting time if load balancing is not perfect, because the essential part of communication can begin only when all involved processes are ready to send and receive the messages. If more than 512 parallel processes are used for the computation, the duration of the sequential part will even increase due to the block structure of the LP task. It will lead to the reduction of parallel efficiency and, consequently, of speedup.

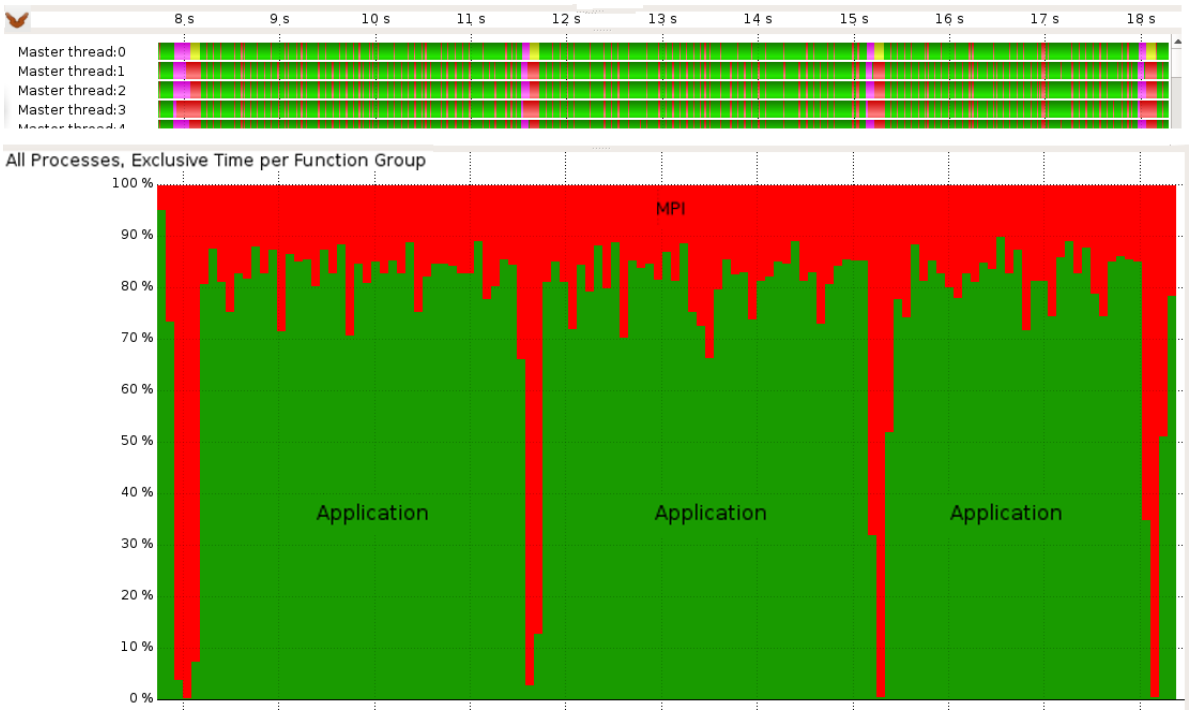


Figure 4.5: Timeline diagram (Top) and exclusive time per function group (Bottom) of a PIPS-IPM++ run with 512 MPI processes

4.2.3 Computers used at HLRS

Figure 4.6: Supercomputer „Hazel Hen“ (Cray XC40) at HLRS und its blade with four compute nodes

HLRS offers a variety of supercomputing systems reflecting the different needs of its scientific and industrial customers (HLRS-Systems, 2019). However, PIPS-IPM++ was mostly started on Hazel Hen. Supercomputer Hazel Hen (see Figure 4.6), a Cray XC40 system, is at the heart of the high performance computing system infrastructure at HLRS. With a peak performance of 7.42 Petaflops (quadrillion floating point operations per second), Hazel Hen is one of the most powerful HPC systems in the world (11/2018 rank 30 in the TOP500, 11/2017 rank 19, 11/2016 rank 14, 11/2015 rank 8). Hazel Hen entered operation in October 2015, is based on the Intel Haswell Processors and the Cray Aries network technologies, and is designed for sustained application performance and high scalability (11/2018 rank 20 in HPCG List, 11/2017 rank 14, 11/2016 rank 12, 11/2015 rank 6). Figure 4.6 on the right side shows one of the 1928 blades of Hazel Hen with four compute nodes and its shared Aries network chip.

Table 4.1: Main components of Supercomputer „Hazel Hen“ (Cray XC40)

Components	Quantity
Cray Cascade Cabinets	41
Dual socket compute node	7712
Processors E5-2680v3 (Haswell)	$2 \times 7712 = 15424$
Compute cores	$12 \times 15424 = 185088$
DDR4 memory per CPU	64 GB
DDR4 memory in total	15424×64 GB= 964 TB
Disk capacity	10 PB
Interconnect	Aries
Operating system	Cray Linux Environment
Power consumption	$\sim$ 3.2 MW

The upcoming system, which HLRS has named Hawk, will have very probable the world's highest sustained performance for science and industrial applications in the fields of energy, climate, mobility, and health. Hawk, based on the Hewlett Packard Enterprise next-generation HPC platform running a next generation AMD EPYC processor code named Rome (Shilov, 2018), will have a theoretical peak performance of 24 petaFLOPs, and consist of a 5,000-node cluster. Hawk is expected to go into full service in the beginning of the year 2020.
Even if the present machine Hazel Hen will be replaced in the year 2020 with a new one, the steps of the below described workflow remain mostly unchanged. Appendix B provides step-by-step configuration guide of PIPS and GAMS software on Hazel Hen. In the following, the key issues for using HLRS resources and their impact on the GAMS/PIPS-IPM WorkFlow are addressed.

4.2.4 Use of Supercomputer at HLRS

Once the proposal for compute time is approved, the new users will receive their login data (username and password). For security reasons[3], the HPC resources of HLRS (inclusive Hazel Hen) are accesible withhin internal HWW network, protected by a firewall and other security measures (see Figure 4.7). To enter this network from public internet, the users have two alternatives:

- Access to HLRS compute platforms requieres a registration of the clients static public IP address in the firewall. If this requirement is met, one can directly login to one of the login nodes of Hazel Hen.
- The second alternative is the usage of VPN service at HLRS. For more information, see HLRS's Knowledge Base section about VPN (HLRS-VPN, 2019).

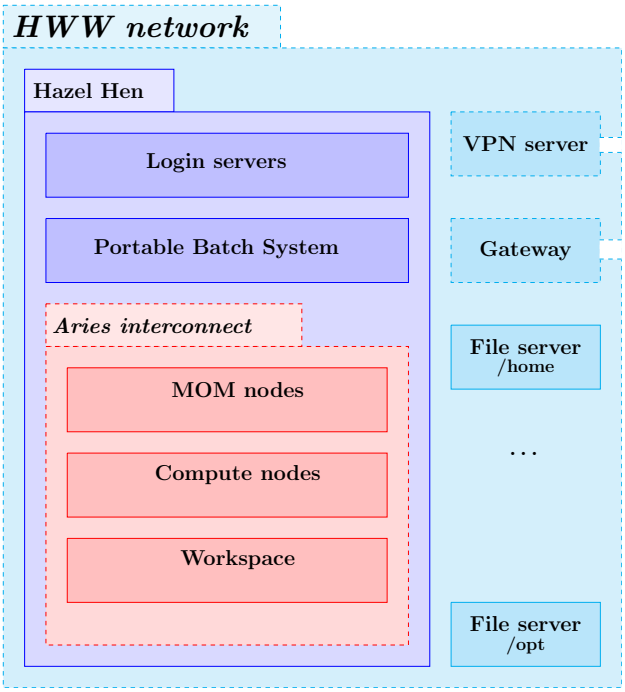
After an HLRS account has been created and, if required, a VPN connection is established, the user can open a Secure Shell (SSH) connection to one of the login nodes of Hazel Hen[4]. A detailed description of Secure Shell at HLRS can be found in Knowledge Base section about SSH (HLRS-SSH, 2019).

4.2.4.1 File Systems at HLRS

In HLRS we distinguish the file systems in their purpose, performance, capacity and durability. There are four main storage systems: HOME, OPT, WORKSPACE and High Performance Storage System (named as HPSS Archive in the following):

[3]HLRS offers a variety of HPC services to industry and Small and Medium-sized Enterprises (HLRS-SICOS, 2019), which have especially high demands for security.

[4]SSH is the exclusive way to get into the secure environment of HWW. If you are on Linux or Mac open terminal and type the command „ssh username@hazelhen.hww.hlrs.de and if you are on windows computer use PuTTY SSH client (putty.org, 2019) or another suitable terminal emulator. After the password prompt you are on front end of Hazel Hen. The best practice to avoid the password query on every connection is use of SSH Key Pair. Most likely the ssh host address for the new machine will be „hawk.hww.hlrs.de“.



- Hazel Hen is in HWW network.
- Login nodes are used for login, compiling, edit and serial execution of the scripts and submission of the computational jobs (PBS-jobs).
- Home-directory is suitable for serial access (e.g. from Login nodes): File system is not parallel.
- Aries network is a high performance network.
- Workspase is a parallel file system.
- A PBS-job script is executed on one of MOM nodes (serial).
- aprun is the ALPS (Application Level Placement Scheduler) application launcher. aprun is started on a MOM node and distribute the application between reserved compute nodes.

Figure 4.7: HWW network for HPC resources at HLRS

- *Home Directory*: Each time you log in, your current directory is set to home. This storage type is available on all compute resources within HWW network[5] Users should store e.g. profiles, script files for workflow tasks, sources for program development in home. The filesystem space of home directory is limited by a small quota in the order of less dozen of GiB and several hundreds thousands of files. This file system is not parallel and its resources are shared between all users in HWW network. Therefore, do not use the home directory to crack numbers (especially for large parallel jobs) - it can paralyze the work of all users on the system.
- *Opt Directory:* The „/opt“ directory is for reading only. Although this directory is available on all HPC resources, the content may be different. It collects the software libraries and the tools, for example compilers and profiler: Unlike standard PC, only a minimal system is installed in standard directories (for example „/usr“). Because of a large number of software packages it may take some time to find specific items in this file system using standard tools such as „find“. Therefore, the software packages are categorized with the Environment Module tool (see section 4.2.4.2).
- *Workspace:* For large files and fast I/O Lustre (Lustre, 2019) based file system are available which make use of the high speed network infrastructure (Aries). Workspace directories are accessible on all compute and login nodes via the workspace mechanism, which allows you to keep data outside your home not only during a run, but also after a run. The idea is to allocate disk space for a number of days, and giving it a name, which allows you to identify a workspace, and to distinguish several workspaces. To get the best performance on parallel file systems, such as Lustre, its always advisible to use MPI-IO or other Higher-level I/O libraries, such as HDF5. The Knowledge Base at HLRS contains more details in section about Workspace (HLRS-WORKSPACE, 2019). If you consider to produce and temporally store more than hundreds of gigabytes or even of terabytes of data its better to make mention of this in the application for compute time due to the quota limit mechanism on HLRS's HPC systems.

[5]Users' HOME directories are mounted via NFS on all login and compute nodes.

- *HPSS Archive:* As mentioned above, the workspaces doesn't provide the storage space for a long period of time and the home directory is essentially limited by the available space. HPSS is a Hierarchical Storage Management System (HSM) and it is designed to manage petabytes of data stored on disks and in tape libraries. HLRS provides more than 20 Petabytes of storage (one tape storage capacity is currently 16 TB). The storaged data are mirrowed on the second tape storage facility in the dedicated server room. This ensures that the data is securely available for many years. It is also recommended to make mention in the application form for computing time the requirements on HPSS Archive.

4.2.4.2 Module Environment at HLRS

All HPC systems at HLRS (incl. Hazel Hen) use the Module Environment tool. With some simple commands of it the user can change the current software development environment (SDE) and load the required external libraries. The tool fulfills its purpose by setting environment variables, such as „$PATH“, „$LD_LIBRARY_PATH“ and others[6]. This ensures that the required software can be found by the Operating System (OS). A detailed description of Module commands can be found in Knowledge Base section about Module Environment (HLRS-MODULE, 2019).

4.2.4.3 Batch system on Hazel Hen

Also certain well-scalable applications require the entire HPC system, in most cases the users execute their programs on a part of the machine. They will be required to define a job script for a fair and efficient distribution of the available hardware resources. The job script can be submitted to one of the queues of Portable Batch System (PBS)[7] (see example on slide 21 in appedix B and example 2 in section 4.3.5.4). PBS performs the job scheduling by allocating the required number of the compute nodes for each of the submitted jobs. Figure 4.8 shows the working principle of PBS.

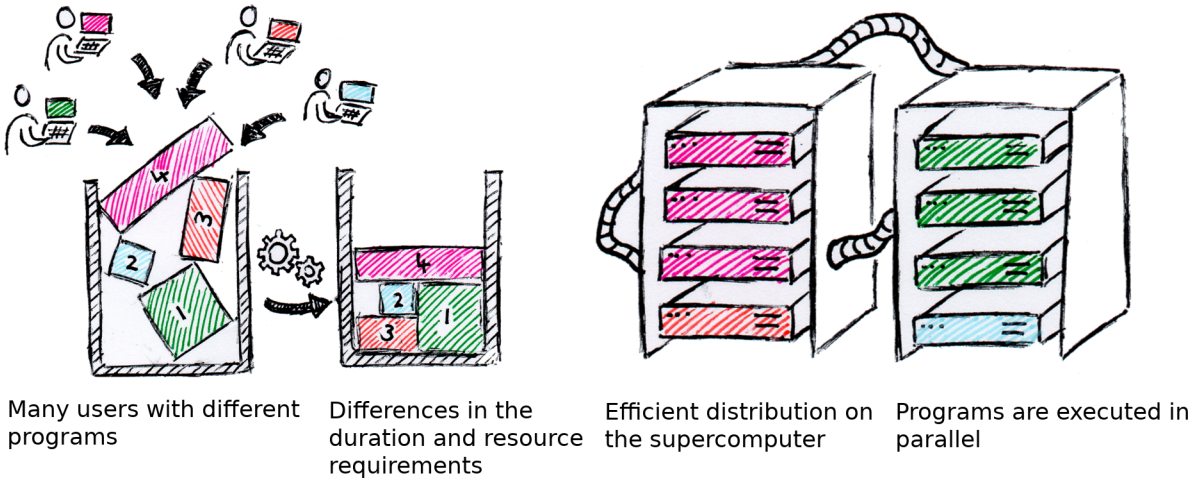


Figure 4.8: Distribution of Compute Jobs in a Supercomputer @Diagram by Philipp Offenhäuser (HLRS)

As a PBS job placed in one of the queues receives at its disposal the computing resources, the script will be executed on the HPC system. A Cray system differs from many other systems on one key point, namely that, the job script is executed not on the first of the allocated compute nodes but on the particular „MOM“ node. Thus one must use in the script the command „aprun“ [8] to start the parallel program on the allocated compute nodes. The „aprun“ comand is very flexible and lets you run the program with almost any topology of pinning the processes and threads to the cores of the allocated nodes. In addition, you can set the frequency

[6]The $PATH variable specifies a list of directories separated by colon (:). During the execution of a bash command, the bash interpretator searches those directories (from left to right) for the first occurrence of the appropriated file and tries to execute it. The $LD_LIBRARY_PATH variable specifies the directories, which are searched by OS for the files with shared libraries needed by execution of the programs.

[7]PBS script is an extension of Linux bash script with a PBS header.

[8]This is analogous to mpirun or srun on JURECA supercomputer.

of the processors for the time of the job execution. Unfortunately, the frequency is then fixed for the entire time of execution, so you can use this property rather only for performance optimization: For example, to determine if your program is memory or compute-bound. A detailed description of Batch system at HLRS can be found in Knowledge Base section about PBS (HLRS-PBS, 2019).

4.2.5 Computers used at JSC

4.2.5.1 JURECA

During the first phase of the BEAM-ME project the supercomputer JURECA (figure 4.9) at the Jülich Supercomputing Centre (JSC) has been used as the primary test platform for the development of PIPS. JURECA has been installed in 2015 and is currently (June 2019) listed on the 52th rank in the Top500 list with a peak performance of 6.56 PFLOPS. The compute nodes are based on 1872 Intel Haswell CPUs. 75 of these node are equipped with two NVIDIA K80 GPUs. Due to the different node memory demand of the applications 1605 of the computes nodes are equipped with 128 GiB DDR4 memory, 128 nodes with 256 GiB and 64 with 512 GiB. Those are connected via a Mellanox EDR InfiniBand high-speed network with non-blocking fat tree topology. For special visualization requirements JURECA has additional 12 nodes dedecated to those purposes. In 2017 JURECA has been extended with a Booster module. It contains 1640 compute nodes with one Intel Xeon Phi 7250-F Knights Landing CPUs per node which are connected via a Intel Omni-Path Architecture high-speed network with non-blocking fat tree topology. As a workload mangement system Slurm is used. Futher information about the hardware configuration and usage can be found here: http://www.fz-juelich.de/ias/jsc/jureca.

Figure 4.9: Supercomputer JURECA (left) and its Booster module (right) at JSC. Copyright: Forschungszentrum Jülich

4.2.5.2 JUWELS

JUWELS (figure 4.10) has entered operation in July 2018 and is currently (June 2019) listed on the 30th rank in the Top500 list with a peak performance 10.4 PFLOPS on CPUs plus 1.6 PFLOPS on GPUs. This system became the main production and development platform for the BEAM-ME project and MEXT participants since November 2018. JUWELS is based on 2271 Intel Skylake CPUs with 96GB memory. In addition there are 240 Skylake node with 192GB memory and 56 Intel Xeon Gold nodes accelerated with 4 NVIDIA V100 GPUs. Those are connected via a Mellanox EDR InfiniBand high-speed network. For special visualization requirements JUWELS has additional 4 nodes dedecated to those purposes. As a workload mangement system Slurm is used. Futher information about the hardware configuration and usage can be found here: http://www.fz-juelich.de/ias/jsc/juwels.

4.2.5.3 JUST

The configuration of the Jülich Storage Cluster (JUST) (figure 4.11) is continuously under movement and expansion to integrate newly available storage technology in order to fulfill the evergrowing capacity and I/O bandwidth demands of the data-intense simulations and learning applications on the supercomputers. Currently the 5th generation of JUST consists of 22 Lenovo DSS systems (Lenovo Distributed Storage Solution) and three older IBM GSS systems (GPFS Storage Server). The software layer of the storage cluster is based on the Spectrum Scale (GPFS) from IBM. JUST provides in total a gross capacity of 75

Figure 4.10: Supercomputer JUWELS at JSC. Copyright: Forschungszentrum Jülich / R.-U. Limbach

PB and a theoretical bandwidth of 400 GB/sec. Additionally in Mid-2018 the storage cluster JUST-DATA started production which realized a large disk based capacity (52 PB gross) and a moderate bandwidth of 20 GB/s. To match the growing data requirements yearly 12-28 PB will be added.

At the lowest level in our hierarchical storage approach there are tape libraries installed with an actual capacity of around 281PB. They are the most cost-efficient technology in terms of TCO and capacity, but with the drawback of a very high latency. It's designation is to store cold data, which will be read very seldom or may be never and is used in the storage hierarchy for three central services: backup and Restore of data, long term archival of data and migration of active (online) data to less expensive storage media. Futher information about the hardware configuration can be found here: http://www.fz-juelich.de/ias/jsc/just.

4.3 Preparing Energy System Models for High Performance Computing with GAMS

To be able to use novel solution approaches like PIPS-IPM that exploit certain problem structures, a processable representation of a model's block structure is required. This section outlines how users can annotate their GAMS models to provide such block structure information to the solution algorithm and how to run PIPS-IPM on model instances generated by GAMS.

4.3.1 Motivation

Many algorithms, as for example the parallel interior point method implemented in PIPS-IPM, exploit certain problem characteristics like e.g. block-diagonal structures. Automatic detection of block structures in models has its limitations and hence a processable block structure information based on the user's deep understanding of the model is often preferable. It is important to note that there is no unique block structure in a model but there are many of them, depending on how rows and columns of the corresponding matrix are permuted. For ESMs blocks may for example be formed by regions and/or time steps.

4.3.2 Model Annotation

For a model with linking variables and linking constraints that has n blocks of non-linking variables the annotation scheme can be summarized as follows (see also figure 4.12).

Figure 4.11: Filesystem JUST at JSC. Copyright: Forschungszentrum Jülich

Variables:

- Linking variables go into block 1 (i.e. `<variable name>.stage = 1`)
- Other variables go into blocks 2, ..., n+1 (i.e. `<variable name>.stage = 2, ..., n+1`)

Equations/Constraints:

- Constraints that contain only linking variables go into block 1 (i.e. `<equation name>.stage = 1`)
- Linking constraints go into block n+2 (i.e. `<equation name>.stage = 2`)
- Other constraints that contain only variables out of one of the blocks 2, ..., n+1 go into blocks 2, ..., n+1 (i.e. `<variable name>.stage = 2, ..., n+1`)

Note that one of the known limitations of the current (as of June 2019) GAMS/PIPS-IPM-Link is that model annotation via the `.stage` attribute does not work in combination with GAMS' Model Scaling Facility.

4.3.3 Model Generation and Solution Reporting

Once a model has been annotated, a corresponding model instance can be generated in a format that fits the requirements of the PIPS-IPM API and is understood by the GAMS/PIPS-IPM solver link. That process is called **model generation**.
A model with n blocks of non-linking variables is represented by n+1 gdx files that hold all the required information about the block structured problem. The details of the model generation are explained below. **Note that the following instructions require GAMS version 25.1.1 or newer.** The latest GAMS version is available at `https://www.gams.com/download/`. A model can be generated either

a) all at once

or

b) in a distributed block-wise fashion.

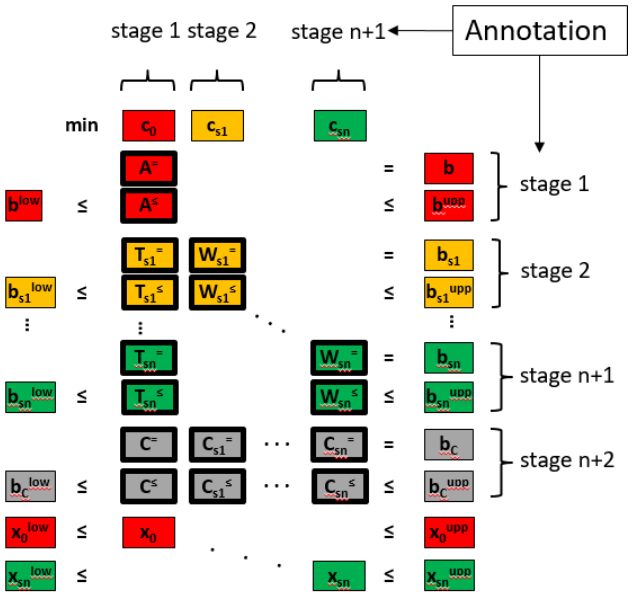


Figure 4.12: Block structure required by the PIPS-IPM API and annotation scheme in GAMS

Note that currently (as of June 2019) the GAMS/PIPS-IPM Link in combination with all-at-once model generation has some known limitations (see section 4.3.3.5).

4.3.3.1 All-at-Once Model Generation

If an entire model is generated at once, we first obtain one large gdx file that contains the entire problem information including the user's annotation of the block structure. That file can be split into the required n+1 gdx files with a tailor-made tool called gmschk. To generate such an all-in-one gdx file, let's call it blockAll.gdx, the following steps need to be taken:

- Complete annotation of the model as described in 4.3.1.
- For LPs the LP solver must be set to convertd, e.g. via a statement

  ```
  option lp=convertd;
  ```

  which must be placed prior to the solve statement.
- A convertd option file with the instruction to create the desired gdx file must be created, e.g. via

  ```
  $echo jacobian blockAll.gdx > convertd.opt
  ```
- convertd has to be instructed to actually use the option file, e.g. via statement

  ```
  <modelName>.optfile = 1;
  ```

  which must be placed prior to the solve statement.
- convertd is a utility which can transform a GAMS model instance to various formats, however, it is invoked just like a solver, e.g. via

  ```
  solve <modelName> use lp min <objVariableName>;
  ```

If the gdx file blockAll.gdx has been created successfully, it needs to be split into several block files. The tool gmschk checks for consistency of the annotation and can also be used to split such a file. Details on how to compile and run that tool are given in section 4.3.5.1. To split blockAll.gdx into n+1 gdx files as required by PIPS-IPM, run gmschk for example as follows

```
<path/to/gmschk>/gmsschk -T <n+1> blockAll.gdx
```

Note that `<n+1>` must be replaced by the proper number that represents the number of (variable) blocks of the model instance at hand. As a result, n+1 gdx files called blockall0.gdx, blockall1.gdx, ..., blockall`<n>`.gdx are created.

4.3.3.2 Speeding up All-at-Once Model Generation

For large-scale instances, all-at-once model generation can require significant resources regarding both, time and memory. This section provides a brief summary of the recommended steps to increase the efficiency of all-at-once model generation. Note that even if these steps are entirely followed, there are known limitations that can currently (as of June 2019) not be surpassed with all-at-once model generation.
In addition to the steps described in 4.3.3.1, efficiency of all-at-once model generation can be improved as follows:

- As usual, the solver option file should contain the instruction to write a GDX version of the Jacobian Matrix. If the string "noVEnames" is added to the filename, original equation and variable names are suppressed. This is extremely useful for the further processing of the gdx files, especially for large-scale model instances. Having the original equation and variable names can be useful for debugging purposes but can make the GDX file significantly larger and slow to write and read. The solver option file `convertd.opt` could for example look as follows:

  ```
  jacobian fileName_noVEnames.gdx
  ```

- To further reduce the size of the GDX file and to improve performance of writing the GDX file, the string "noUELs" can be added to the file name. This results in unique elements (UELs) not being written to the GDX file. In GAMS jargon, a "UEL", also known as "label", refers to the elements of one-dimensional sets, a fundamental data structure in GAMS. The solver option file `convertd.opt` could for example look as follows:

  ```
  jacobian fileName_noVEnames_noUELs.gdx
  ```

  Model instances captured in GDX files created without UELs can be split with gmschk and solved with PIPS-IPM++. The solution reportingworks as well. **However, it should be noted that the tools introduced in section 4.3.4 cannot be applied on that type of reduced GDX file.**

- When splitting the gdx file (e.g. `fileName_noVEnames.gdx`), the choice of gmschk options can have a significant impact on the time it takes to split. For large-scale instances, it is recommended to split with option

  ```
  -T split GDX file into multiple GDX files without uels and strings
  ```

  A gmschk call could for example look as follows:

  ```
  <path/to/gmschk>/gmsschk -T <n+1> fileName_noVEnames.gdx
  ```

4.3.3.3 Distributed Model Generation

While in the previous chapter a large gdx file containing the entire model has been created first and then sliced into smaller block gdx file, distributed model generation aims to generate these small block gdx files immediatey in a distributed way. The main challenge with distributed model generation is that all linking variables must always be registered in the same order in every block, even though the corresponding variable may not appear in any of the constraints of the block. Similar logic applies for the linking constraints. This is crucial for the GAMS/PIPS-IPM interface since the information what joins different blocks is retrieved under the assumption that all linking variables and constraints are registered in every block in the same order. // Distributed model generation for the stochastic SIMPLE version without linking constraints has been implemented and is explained in Example 1: SIMPLE-SP without linking constraints. Distributed model generation for model instances with linking constraints is available via `simple_v2/simple4pips.gms` and is explained in Example 2: SIMPLE4PIPS with linking constraints and distributed model generation.

4.3.3.4 Solution Reporting

Note that the solution can only be reported if the model is generated via all-at-once model generation. To import the solution from PIPS-IPM back to GAMS, the following steps are required.

- After generating the model, a dictionary file that holds information about the variable and equation names of the GAMS Model needs to be saved. By default this dictionary file is created in the process directory (225a, 225b,...) of the current GAMS run. The process directory is temporary and by default it gets deleted after a (successful) GAMS run. Hence, it is recommended to retrieve the dictionary file via system call

  ```
  execute 'mv -f \%gams.scrdir\%gamsdict.dat <filename>_dict.gdx';
  ```

  which moves the dictionary out of the process directory into the working directory. Note that `<filename>` has to correspond to the name of the gdx file created via all-at-once model generation.

- To instruct PIPS-IPM to return the solution gmspips has be started with option `printsol`.

- The solution provided by PIPS-IPM is mapped back to the original GAMS namespace (which is why the dictionary is needed) and stored in a gdx file called `<filename>_sol.gdx` which can be loaded back into GAMS via

  ```
  ...
  execute_loadpoint '<filename>_sol.gdx';
  ...
  ```

The entire GAMS/PIPS-IPM WorkFlow with solution reporting is visualized in figure 4.13

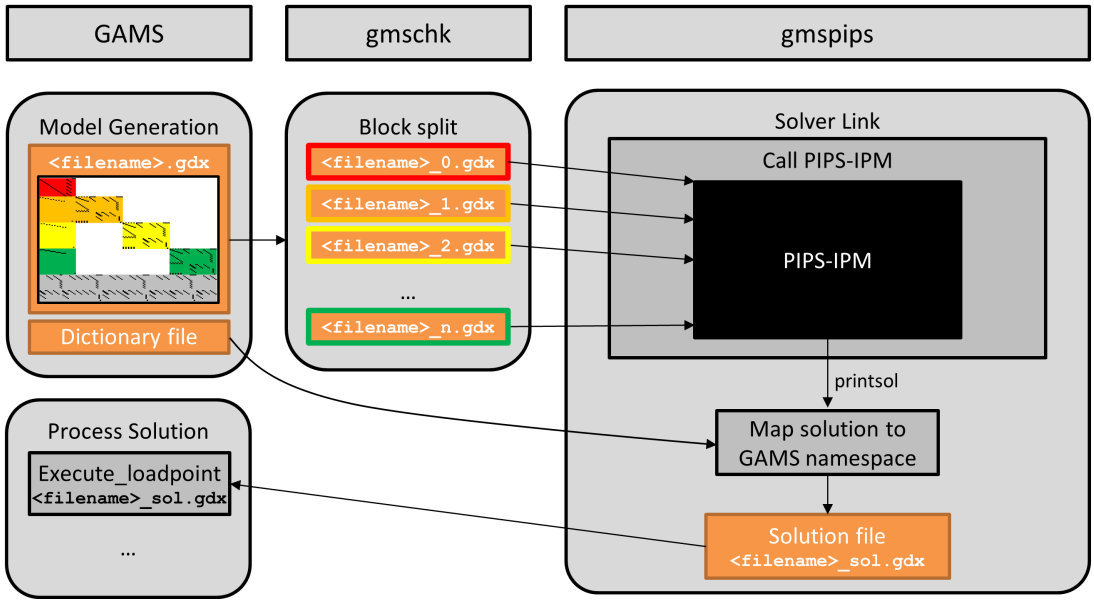


Figure 4.13: Overview of GAMS/PIPS-IPM WorkFlow with Solution Reporting

The workflow is also summarized in the following code snippet where the number of tasks and the number of blocks is assumed to be computed and stored in compile time variables `%NBTASKS%` and `%NBBLOCKS%`.

```
...
option lp=convertd;
$echo jacobian allblocks_noVEnames.gdx > convertd.opt
solve simple min OBJ use lp;
execute 'mv -f %gams.scrdir%gamsdict.dat allblocks_noVEnames_dict.gdx';
execute 'gmschk -T -g "%gams.sysdir%" %NBBLOCKS% allblocks_noVEnames.gdx';
execute 'srun -n %NBTASKS% gmspips %NBBLOCKS% allblocks_noVEnames "%gams.sysdir%" printsol';
execute_loadpoint 'allblocks_noVEnames_sol';
display '### OBJECTIVE FUNCTION VALUE:', OBJ.l;
...
```

4.3.3.5 Known Limitations

The GAMS/PIPS-IPM-Link has initially been implemented with the research project BEAM-ME and with the end of that project, it still has a few limitations worthwhile to mention.

1. There is a hard maximum of 2,147,483,647 non-zeroes that can be contained in a model instance generated by GAMS. The same limitations exists in many solver APIs. Reason is that the array of non-zeroes is indexed with a 32 bit signed integer. This limitation can be surpassed with distributed model generation.
2. All-at-once model generation does not benefit from multiple cores. Currently (as of June 2019), distributed model generation has to be implemented by the user which is challenging. For models where the solution process can be sped up dramatically by hundreds or thousands of cores, model generation can become a bottleneck. Note that there are plans to implement generic and user friendly block sharp model generation in a potential follow-up project.
3. The `.stage` attribute used for the model annotation overloads another variable attribute, namely the `.scale` attribute. Hence, if displayed or looked at in a gdx file, the annotation defined via `.stage` will be shown as `.scale`. Note that this prevents usage of the Model Scaling Feature for annotated models that should be solved with PIPS-IPM.

4.3.3.6 Using Different Platforms for Model Generation and Solution Process

HPC Systems are usually Linux based while many energy system modelers rely on Windows as an operating system. While GAMS code is in general platform independent, ESMs frequently pull data out of various sources (e.g. MS Excel) or use system calls to execute external programs and scripts that tie the ESM to a particular platform.
This section discusses ways to handle such platform dependencies and to carry out certain steps of the GAMS/PIPS-IPM work flow on different machines.

1. Generate model locally and upload to HPC: If the modeler has a local machine available that has sufficient resources to generate the model, all-at-once model generation can be carried out on that machine and the resulting gdx file can be transferred to the HPC where it is splitted and PIPS-IPM is called. If needed, the solution file can be transferred back to the local machine once gmspips has terminated.
2. If resources on the locally available machine are insufficient to execute the GAMS program, the so-called dump file is an convenient way to capture an instance with its associated data and algebraic model representation on the local machine but to generate the actual LP instance on a different machine. To do so, add `dumpopt=11` to the list of command line parameters when running GAMS. To create the dump file it suffices to compile but not execute a GAMS program. This can be accomplished by instructing GAMS to compile only via command line parameter `action=c`. An exemplary command line call of GAMS to create a dump file in compile only mode looks as follows:

   ```
   gams model.gms action=c dumpopt=11
   ```

   That would create a file model.dmp. That file can then be transferred to a suitable machine and can

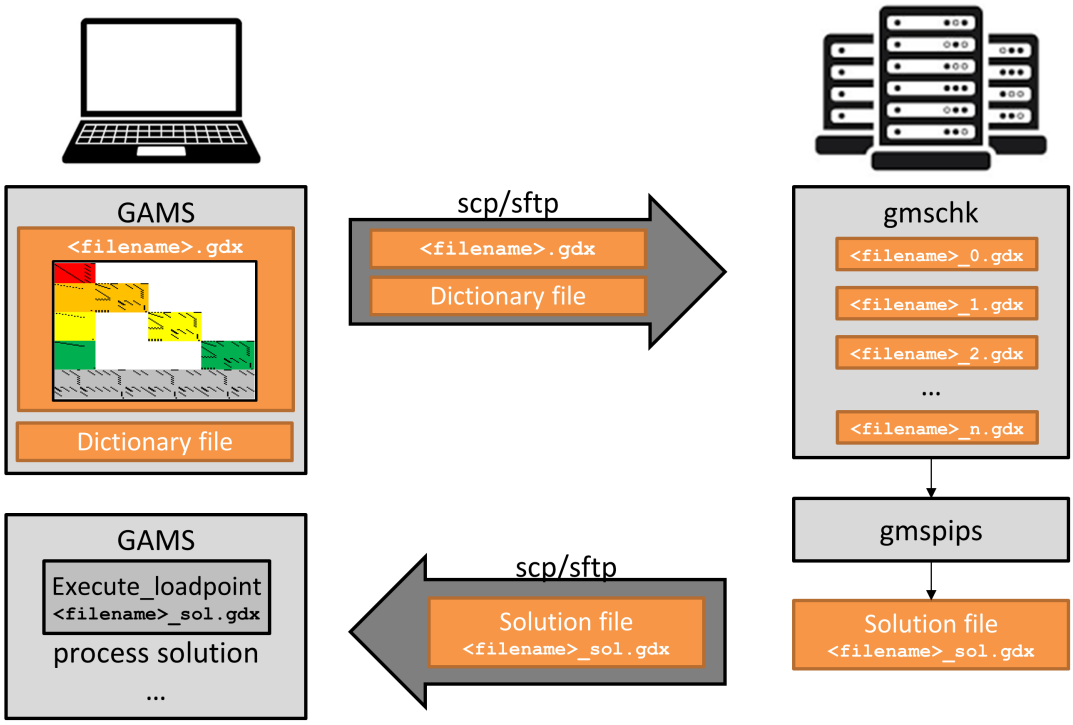


Figure 4.14: GAMS/PIPS-IPM: Generate Locally and solve on HPC

be executed by GAMS via

```
gams model.dmp
```

Sometimes, the execution of dump files may not work right away, for example because output files (e.g. when the put writing facility or commands like execute_unload are used) should be created in particular folders , assuming a certain folder structure which only existed on the local machine that was used to create the dump file. Those errors can usually be fixed or ignored easily.

4.3.4 Useful Tools

Several useful tools that support the process of annotating and generating model instances for PIPS-IPM are available. **Note:** The following tools can be found in the BEAM-ME repository.

4.3.4.1 checkanno

With `checkanno.gms` the equation annotation can be computed from the variable annotation. If the equation annotation already exists, it will be checked for consistency and if errors are detected they will be fixed. The program can be executed via

```
gams checkanno --jacfilename=<filename.gdx>
```

The resulting file will be named `filename_novenames.gdx`.

4.3.4.2 stripjac

With `stripjac.gms` variable and equation names can be removed from jacobian GDX files that have been created without string "novenames" in their filename The program can be executed via

```
gams stripjac.gms --jacfilename=<filename.gdx>
```

The resulting file will be named `filename_novenames.gdx`.

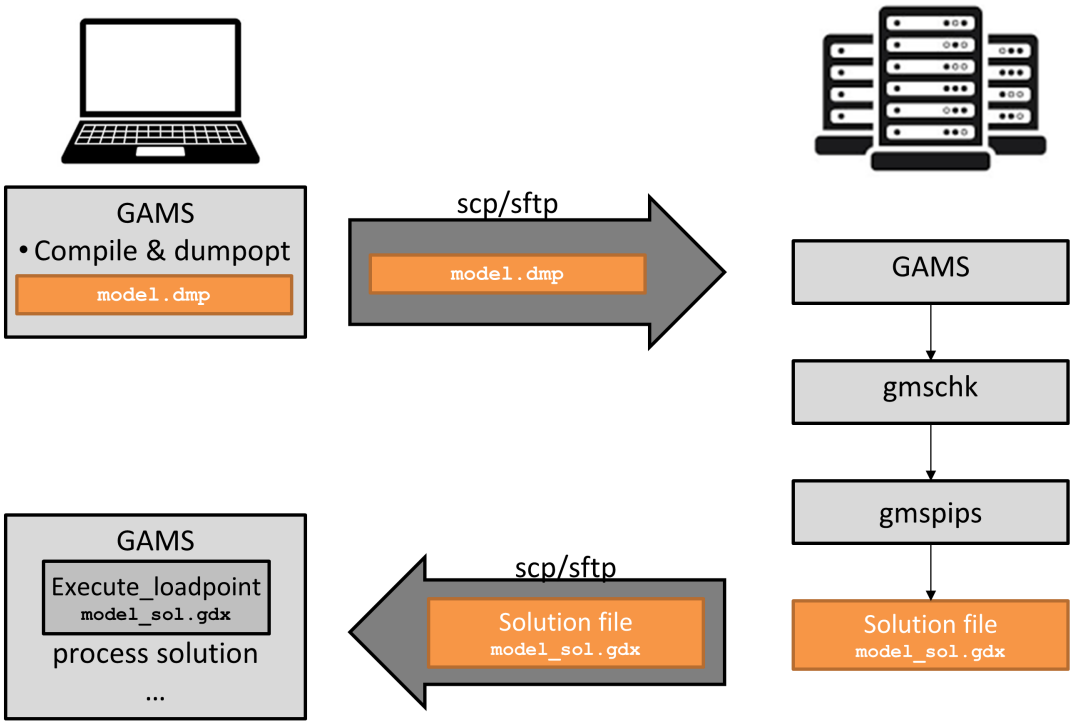


Figure 4.15: GAMS/PIPS-IPM: Create dump file locally, generate and solve on HPC

4.3.4.3 solveJacobian

With `solveJacobian.gms`, model instances stored in a single gdx file can be solved with all LP solvers linked with GAMS. The program can be executed via

```
gams solveJacobian --jacfile=<filename.gdx>
```

If no other options are set, the model instance will be solved with the default LP solver with default settings. If double dash parameter `--TARGET=<string>` is set, the model instance is not solved but converted into another file format. Possible options are

`LP`	Pass the model to CONVERT to convert it to the `CplexLP` format.
`MPS`	Pass the model to CONVERT to convert it to the `CplexMPS` format.
`SOPLEXLP`	Pass the model to SOPLEX and let SOPLEX convert it to the `lp` format.
`CPLEXLP`	Pass the model to CLEX and let CPLEX convert it to the `lp` format.

4.3.5 Solving a GAMS Model Instance using the GAMS/PIPS-IPM Solver Link

The main purpose of the model annotation is to produce input files for PIPS-IPM. Currently, the model generation and running PIPS-IPM on the generated model are two separate processes.

For the following explanations we assume that a suitable branch (e.g. "linking-zib" or "linkin-zib-pardiso") from the git repository [9] has been cloned. For the sake of simplicity we assume the clone is located in `$HOME/PIPS`.

[9] gitlab.version.fz-juelich.de/breuer1/PIPS_beamme

4.3.5.1 Compiling gmschk and gmspips

To compile gmspips and gmschk the lines for compilation instructions have been added to CMakeList.txt in the `$HOME/PIPS/PIPS-IPM` folder. Now change the directory to the `$HOME/PIPS/build` folder and run `cmake .. -DBUILD_ALL=OFF -DBUILD_PIPS_IPM=ON && make`. This should produce an executable gmspips and gmschk in the build folder.

4.3.5.2 Running gmschk

For a brief summary on how to use gmschk run

```
<path/to/gmschk>/gmschk -h
```

The following should be written to output:

```
Usage: [-dhtTwWx] [-b actBlock] [-g GAMSSysDir] [-o n] numBlocks file[Stem]
  -h          print usage

  Splitting operation:
    -t        split GDX file into multiple GDX files
    -T        split GDX file into multiple GDX files without uels and strings
    -g        specify GAMS system directory
    -b        specify single block
    -o        specify stage offset (default 1)
    numblocks total number of blocks
    file      GDX file

  Analysis operation:
    -d        debugging mode (unmatched vars, equs, and matrix elements with good names)
    -w        output of block structure counts to stdout
    -W        output of block structure to stdout
    -x        fileStem is GDX file stem
    -g        specify GAMS system directory
    -b        specify single block
    -o        specify stage offset (default 1)
    numblocks total number of blocks
    fileStem  GDX file or file stem
```

gmschk provides several options and has two general modes of operation, *splitting* and *analysis*. When used in splitting mode (i.e. with `-t` or `-T`), options that are for analysis only (like `-d`, `-w`, `-W` and `-x` are ignored. To split a file, e.g. blockAll_novenames.gdx into n+1 gdx files as required by PIPS-IPM run

```
<path/to/gmschk>/gmsschk -T <n+1> blockAll_novenames.gdx
```

Note that `<n+1>` must be replaced by the proper number that represents the number of (variable) blocks of the model instance at hand. As a result, n+1 gdx files called blockall_novenames0.gdx, blockall_novenames1.gdx, ..., blockall_novenames`<n+1>`.gdx are created.
Also note that in an earlier gmschk version, the option `-X` had to be used to tell gmschk that the input is a GDX file. At an early stage of development, this was a necessary option to distinguish between different input formats. Other input formats have vanished in the meantime, such that this option has become redundant. While it is not necessary to specify `-X` anymore, the option is still accepted but does not cause any non-default behavior.

4.3.5.3 Running gmspips

Assuming that the n+1 gdx block files (`blockall0.gdx, blockall1.gdx, ..., blockall<n+1>.gdx`) that serve as PIPS-IPM input have been created successfully and gmspips has been compiled as explained in Compiling gmschk and gmspips, we can run PIPS-IPM via the driver gmspips as follows:

```
mpirun -n <n> $HOME/PIPS/build/gmspips <n+1> </path/to/gdx>/blockAll $GAMSSYSDIR
```

Please note that the first argument of `mpirun <n>` determines the number of parallel MPI tasks while the first argument of `gmspips <n+1>` is the total number of block files. If we generate for example the stochastic problem version with 5 scenarios (`--NBSCEN=5`) we obtain an instance with 6 blocks in total. However, the maximum number of MPI tasks is the total number of blocks minus 1. For 5 scenarios this would again correspond to 5 as in `mpirun -n 5 [...]`. The environment variable `$GAMSSYSDIR` points to the GAMS system directory location. You can start gmspips in any directory. If the GDX files are not located in your current directory, you need to specify the location to the GDX file as part of the file name stem.

4.3.5.4 Examples

The purpose of the following examples is to guide the reader through the process of generating input files from different variants of the SIMPLE model to run the GAMS/PIPS-IPM solver link. For the experiments, it might be best to add the GAMS system directory to the PATH environment variable or to remember it in a corresponding environment variable, e.g `GAMSSYSDIR`. All experiments are done from the command line/shell.

Note:

- Some examples make use of recent developments and require GAMS version 25.1.1 or newer!
- The SIMPLE model(s) used in the examples can be found in folder `simple_v2` which is contained in the repositories `PIPS_beamme` and `best_practice_guide`.

Example 1: SIMPLE-SP without linking constraints

1. Clone a suitable branch (e.g. "linking-zib" or "linking-zib-pardiso" from the git repository[10]. For sake of simplicity we assume the clone is located in `$HOME/PIPS`. The relevant folders for the GAMS/PIPS-IPM link are in `$HOME/PIPS/PIPS-IPM/Drivers`. You will find the sub-directories: simple, simple_v2, gmspips, and gams. simple_v2 has all the GAMS and data files required to run GAMS to produce input files for GAMS/PIPS. gmspips contains the custom source for the GAMS/PIPS link while gams contains some C header and source files that are required to build the GAMS/PIPS link but are also shipped with every GAMS distribution.

2. The first experiment is to run a small simple model and solve with Cplex. For this and all other experiments change directory to `$HOME/PIPS/PIPS-IPM/Drivers/simple_v2`

   ```
   cd $HOME/PIPS/PIPS-IPM/Drivers/simple_v2
   ```

   and run

   ```
   gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=999
   ```

   This will terminate with

   ```
   Optimal solution found.
   Objective :           8.897090
   ```

   and some more GAMS lines.

3. Second experiment is to create a single GDX file that contains the entire simple model. Run

   ```
   gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=-2
   ```

   This produces a file called blockall.gdx. Check for the existence and contents of blockall.gdx by running

   ```
   gdxdump blockall.gdx
   ```

   This command should also list the different block numbers. Run

   ```
   gdxdump blockall.gdx symb=x | grep SCALE
   ```

[10] gitlab.version.fz-juelich.de/breuer1/PIPS_beamme

and you should see a couple of lines with `SCALE 2` to `SCALE 6` in the output.

4. Third experiment to run simple with sequential block model generation. Run

```
gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=0
gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=1
gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=2
gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=3
gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=4
gams simple --TO=0.0003 --nbregions=5 --method=spexplicitde --nbscen=5 --CPLEXBENDERS=0 --SCENBLOCK=5
```

   This produces files called block0.gdx to block5.gdx. Check for the existence of these files and that the SCALE field has the appropriate values.

5. On a HPC (Linux) machine with mpirun prepare a submission script for the parallel block model generation (generateSimpleBlock.sh):

```
#!/bin/bash
mkdir blk$PMI_RANK
SIMPLEPATH=$HOME/PIPS/PIPS-IPM/Drivers/simple_v2
ARGS="--TO=0.0003 --NBREGIONS=5 --nbscen=5 --CPLEXBENDERS=0 "
ARGS+="fileStem=simple$PMI_RANK lo=2 sd=blk$PMI_RANK optdir=blk$PMI_RANK"
echo Generating block $PMI_RANK
gams $SIMPLEPATH/simple $ARGS --METHOD=spexplicitde --SCENBLOCK=$PMI_RANK
if [ $? != 0 ] ; then
echo Error generating block $PMI_RANK
else
echo Done generating block $PMI_RANK
fi
rm -rf blk$PMI_RANK
```

   Execute the generation script (from anywhere):

```
mpirun -n 6 $HOME/PIPS/PIPS-IPM/Drivers/simple_v2/generateSimpleBlock.sh
```

   After the run make sure that the files block0.gdx to block5.gdx exist. The run can be done from any directory and it will deposit the GDX block files in this directory. All temporary and permanent files will be written in this directory. The generateSimpleBlock.sh script knows about the location of simple.gms and other files via the SIMPLEDIR shell variable.

6. Generating bigger instances: The GAMS SIMPLE model is highly parameterized. The decomposition in the examples above is done by price scenarios for generation cost (not by regions or time as discussed previously). This decomposition has the advantage to have equal block sizes plus the ability to produce any number of blocks. You can increase the number of blocks by increasing the number after `--nbscen`. This number corresponds to the number of blocks. You can also increase the size of the individual bock by

   a. increasing the time horizon. For this, one needs to increase the number after the `--TO`. The maximum number is 1 corresponding to a time horizon of 8760 hours. 0.5 means e.g. 4380 hours.
   b. increasing the number of regions. For this, one needs to increase the number after `--nbregions`.

7. When generating bigger instances, it may be useful to have in mind how to organize the solution process with GAMS/PIPS. An instance with n scenarios (meaning n+1 blocks) can be solved with at most n MPI processes. We can also solve with fewer MPI processes, e.g. $m < n$. The distribution of blocks to MPI processes follows a strict logic in PIPS. MPI process 0 to m-2 gets floor(n/m) scenario blocks plus block 0. The last MPI process m-1 gets the remaining (n-floor(n/m)*(m-1)) scenario blocks plus block 0. Here is an example with n=100 and m=30:

```
MPI rank 0: blk: 0,1,2,3
```

```
MPI rank 1: blk: 0,4,5,6
MPI rank 2: blk: 0,7,8,9
...
MPI rank 28: blk: 0,85,86,87
MPI rank 29: blk: 0,88,89,90,...,100
```

Knowing the distribution, we can already generate the blocks in a parallel fashion in appropriate subdirectories using the following script (gensub.sh):

```
#!/bin/bash
if [ x$1 = x ]; then
   echo "Usage gensub.sh nScenarios"
else
   mkdir rank$PMI_RANK
   pushd rank$PMI_RANK
   SIMPLEPATH=$HOME/PIPS/PIPS-IPM/Drivers/simple_v2
   ARGS="--TO=0.0003 --NBREGIONS=5 --nbscen=$1 --CPLEXBENDERS=0 "
   ARGS+="lo=2 --SCENBLOCK=0 --METHOD=spexplicitde"
   rankp1=$(($PMI_RANK + 1))
   nSize=$(($1 / $PMI_SIZE))
   nStart=$((($nSize * PMI_RANK) + 1))
   if [ $rankp1 -eq $PMI_SIZE ] ; then # last process
     nEnd=$1
   else
     nEnd=$(($nSize * (PMI_RANK + 1)))
   fi
   echo Generating blocks $nStart to $nEnd for $PMI_RANK
   gams $SIMPLEPATH/simple $ARGS --SCENLISTSTART=$nStart --SCENLISTEND=$nEnd
   if [ $? != 0 ] ; then
       echo Error generating blocks for $PMI_RANK
   else
       echo Done generating blocks for $PMI_RANK
   fi
   popd
fi
```

This script creates permanent directories ranki that contain the GDX files of the blocks required by PIPS MPI process i:

```
mpirun -n 30 $HOME/PIPS/PIPS-IPM/Drivers/simple_v2/gensub.sh 100
```

Again, the script can be run from anywhere and creates the ranki directories in the current directory. With the example from above (n=100 and m=30) we get:

```
[beamme@anton tmp]$ ls
rank0   rank11  rank14  rank17  rank2   rank22  rank25  rank28  rank4  rank7
rank1   rank12  rank15  rank18  rank20  rank23  rank26  rank29  rank5  rank8
rank10  rank13  rank16  rank19  rank21  rank24  rank27  rank3   rank6  rank9

[beamme@anton tmp]$ ls rank0 rank1 rank2 rank28 rank29
rank0:
block0.gdx  block2.gdx  convertd.opt  simple.lst
block1.gdx  block3.gdx  simple.log
rank1:
block0.gdx  block5.gdx  convertd.opt  simple.lst
```

```
block4.gdx  block6.gdx  simple.log

rank2:
block0.gdx  block8.gdx  convertd.opt  simple.lst
block7.gdx  block9.gdx  simple.log

rank28:
block0.gdx   block86.gdx  convertd.opt  simple.lst
block85.gdx  block87.gdx  simple.log

rank29:
block0.gdx    block89.gdx  block92.gdx  block95.gdx  block98.gdx   simple.log
block100.gdx  block90.gdx  block93.gdx  block96.gdx  block99.gdx   simple.lst
block88.gdx   block91.gdx  block94.gdx  block97.gdx  convertd.opt
```

8. To run gmspips in the folder where we have placed all GDX block files we can enter the following command:

   ```
   mpirun -n <n> $HOME/PIPS/build/gmspips <n+1> /path/to/gdx/block $GAMSSYSDIR
   ```

   In the more complicated setup where the block GDX files are placed in the ranki directories gmspips needs to be started by a script (runsub.sh) because the location of the GDX files depends on the PMI_RANK variable:

   ```
   #!/bin/bash
   if [ x$1 = x ]; then
      echo "Usage runsub.sh nScenarios"
   else
      if [ x$GAMSSYSDIR = x ]; then
         GAMSSYSDIR=$HOME/PIPS/PIPS-IPM/Drivers/gams25.0_linux_x64_64_sfx
      fi
      echo $GAMSSYSDIR
      pushd rank$PMI_RANK
      nScenp1=$(( $1 + 1 ))
      $HOME/PIPS/build/gmspips $nScenp1 block $GAMSSYSDIR
      popd
   fi
   ```

   This script runsub.sh needs to be started in the directory that contains all ranki directories, eg:

   ```
   mpirun -n 30 $HOME/PIPS/PIPS-IPM/Drivers/simple_v2/tmp/runsub.sh 100
   ```

Example 2: SIMPLE4PIPS with linking constraints and distributed model generation

1. Clone a suitable branch (e.g. "linking-zib-pardiso") from the git repository[11]. For sake of simplicity we assume the clone is located in `$HOME/PIPS` and that environment variable `$GAMSSYSDIR` points to the GAMS system directory location. The relevant folders for the GAMS/PIPS-IPM link are in `$HOME/PIPS/PIPS-IPM/Drivers`. There, you will find the subdirectories: simple, simple_v2, gmspips, and gams. simple_v2 has all the GAMS and data files required to run GAMS to produce input files for GAMS/PIPS. gmspips contains the custom source for the GAMS/PIPS link while gams contains some C header and source files that are required to build the GAMS/PIPS link but are also shipped with every GAMS distribution.

[11] gitlab.version.fz-juelich.de/breuer1/PIPS_beamme

2. The first experiment is to run a small simple model and solve with CPLEX. For this and all other experiments change directory to `$HOME/PIPS/PIPS-IPM/Drivers/simple_v2`

   ```
   cd $HOME/PIPS/PIPS-IPM/Drivers/simple_v2
   ```

   and run

   ```
   $GAMSSYSDIR/gams simple4pips --NBREGIONS=5 --TO=0.0191 lp=cplex
   ```

   This will terminate with

   ```
   Optimal solution found.
   Objective :        2071.650401
   ```

   and some more GAMS lines.

3. Second experiment is to create a single GDX file that contains the entire simple model with annotation:

   ```
   $GAMSSYSDIR/gams simple4pips --NBREGIONS=5 --TO=0.0191 --METHOD=PIPS --TBSIZE=24
   ```

   The instance created via parameters `--NBREGIONS=5 --TO=0.0191` contains 5 regions and 168 hours (=1 week). Parameter `--TBSIZE=24` (default) means that 24 time steps form a time block, i.e. we obtain 7 time blocks plus the block of linking variables. If TBSIZE is no divisor of the number of time steps the last time block will be smaller than TBSIZE time steps. The call above produces a file called allblocks_noVEnames.gdx. Check for the existence and contents of allblocks_noVEnames.gdx by running

   ```
   $GAMSSYSDIR/gdxdump allblocks_noVEnames.gdx noData
   ```

   The following command should also list the different block numbers. Run

   ```
   $GAMSSYSDIR/gdxdump allblocks_noVEnames.gdx symb=x | grep SCALE
   ```

   and you should see a couple of lines with SCALE 2 to SCALE 8 in the output that indicate the block membership of the variables.

4. The next experiment is to split allblocks_noVEnames.gdx into several block files. The tool gmschk checks for consistency of the annotation and can also be used to split such a file. How to compile that tool is explained in section 4.3.5.1. The following command splits the gdx file created in the previous step:

   ```
   $HOME/PIPS/build/gmschk -T -g $GAMSSYSDIR 8 allblocks_noVEnames.gdx
   ```

   This produces files called allblocks_noVEnames0.gdx to allblocks_noVEnames7.gdx. Check for the existence of these files and that the SCALE field has the appropriate values. For example, with

   ```
   $GAMSSYSDIR/gdxdump allblocks_noVEnames3.gdx FilterDef=N symb=x | grep SCALE
   ```

   you should see only variables with SCALE=4 and SCALE=1 (linking variables).

5. Generating bigger instances: The GAMS model simple4pips is highly parameterized. The decomposition in the examples above is done by time blocks. However, simple4pips also supports regional decomposition as well as a combination of regional and time decomposition. As regional decomposition does not appear to be very promising, the default is to decompose (annotate) by time blocks only. By default, a time block contains 24 time steps. The size can be varied via parameter `--TBSIZE`. The number of time blocks can be increased by

   a. increasing the time horizon. For this, one needs to increase the number after the `--TO`. The maximum number is 1 corresponding to a time horizon of 8760 hours. 0.5 means e.g. 4380 hours.
   b. decreasing the resolution. For this, one needs to set the number after optional parameter `--RESOLUTION` to a smaller value. Default is 1 which corresponds to a time discretization of 1 hour. Setting `--RESOLUTION=0.5` results in 30 minutes steps etc.

c. decreasing the time block size. For this, one needs to set the number after `--TBSIZE` to a smaller value.

To increase the size entire model one could also increase the number of regions. For this, one needs to increase the number after `--NBREGIONS`.

6. To run gmspips in the folder where we have placed the GDX block files we can enter the following command:

   ```
   mpirun -n <n> $HOME/PIPS/build/gmspips <n+1> /path/to/gdx/filename $GAMSSYSDIR
   ```

   where n determines the number of MPI processes and n+1 is the number of block files.
   The following command will do this for our running example with 7 MPI processes:

   ```
   mpirun -n 7 $HOME/PIPS/build/gmspips 8 allblocks_noVEnames $GAMSSYSDIR
   ```

7. To generate the block files of our running example in parallel (with sliced data reading), prepare the following submission script (genSimple4pipsBlocks.sh).

   ```
   #!/bin/bash
   mkdir blk_$PMI_RANK
   ARGS="--TO=0.0191 --NBREGIONS=5 --METHOD=PIPS --SLICE=2 gdxcompress=1 lo=2 "
   ARGS+="fileStem=simple4pips$PMI_RANK procdir=blk_$PMI_RANK optdir=blk_$PMI_RANK keep 1"
   echo Generating block $PMI_RANK
   /home/beamme/PIPS/PIPS-IPM/Drivers/gams25.0_linux_x64_64_sfx/gams simple4pips $ARGS --BLOCK=$PMI_RANK
   if [ $? != 0 ] ; then
    echo Error generating block $PMI_RANK
   else
    echo Done generating block $PMI_RANK
   fi
   rm -rf blk_$PMI_RANK
   ```

   Note that in this script we also use parameter `--SLICE=2` which activates sliced data reading and requires the input data to be pregenerated once and stored in a gdx files via command

   ```
   $GAMSSYSDIR/gams simple_data_gen --NBREGIONS=5 --WRITEGDX=1
   ```

   After this has been done, parallel model generation can be started with the following command:

   ```
   mpirun -n 8 genSimple4pipsBlocks.sh
   ```

   This should create files block_noVEnames0.gdx to block_noVEnames7.gdx. Afterwards the corresponding model instance can be solved with PIPS-IPM via the following command:

   ```
   mpirun -n 7 $HOME/PIPS/build/gmspips 8 block_noVEnames $GAMSSYSDIR
   ```

8. In general, the distributed model generation works from any location and does not need to be run from the simple_v2 directory. This example illustrates how to generate a large scale model instance in parallel from a different location. For this example we want to create a large-scale model instance with

   - 20 regions
   - a 30 minute time resolution (i.e. $8750 \cdot 2 = 17520$ time steps)
   - a block size of 50 (i.e. every block contains 50 time steps, i.e. there will be 351 time blocks plus block 0 for the linking variables).

   First, we navigate to our home directory and create a folder `parGen` with a sub directory nbreg20_res30minutes_blocksize50_blocks351 where we will run this experiment.

   ```
   cd $HOME
   mkdir parGen
   cd parGen
   mkdir nbreg20_res30minutes_blocksize50_blocks351
   cd mkdir nbreg20_res30minutes_blocksize50_blocks351
   ```

similar to the previous example 2.7, we create a submission script genSimple4pipsBlocks.sh.

```
#!/bin/bash
mkdir blk_$PMI_RANK
GAMSSYSDIR=$HOME/PIPS/PIPS-IPM/Drivers/gams25.0_linux_x64_64_sfx
SIMPLEPATH=$HOME/PIPS/PIPS-IPM/Drivers/simple_v2
ARGS="--NBREGIONS=20 --TO=1 --RESOLUTION=0.5 --TBSIZE=50 --METHOD=PIPS "
ARGS+="lo=2 gdxcompress=1 --SLICE=2 keep 1 "
ARGS+="fileStem=simple4pips$PMI_RANK procdir=blk_$PMI_RANK optdir=blk_$PMI_RANK "
echo Generating block $PMI_RANK
$GAMSSYSDIR/gams $SIMPLEPATH/simple4pips $ARGS --BLOCK=$PMI_RANK
if [ $? != 0 ] ; then
 echo Error generating block $PMI_RANK
else
 echo Done generating block $PMI_RANK
fi
rm -rf blk_$PMI_RANK
```

Note that this time we set environment variables `GAMSSYSDIR` and `SIMPLEPATH` in the submission script. If your GAMS system directory and/or the simple_v2 folder are stored in a different location, these lines have to be adjusted accordingly Again, this script also activates sliced data reading via parameter `--SLICE=2` and requires the input data to be pregenerated once and stored in a gdx files via commands

```
SIMPLEPATH=$HOME/PIPS/PIPS-IPM/Drivers/simple_v2/
$GAMSSYSDIR/gams $SIMPLEPATH/simple_data_gen --NBREGIONS=20 --WRITEGDX=1
```

Afterwards, we can create the block files block_noVEnames0.gdx to block_noVEnames351.gdx. via the following command:

```
mpirun -n 352 genSimple4pipsBlocks.sh
```

9. In this example we show how to submit the job(s) from the previous example 2.7 for parallel model generation to the JURECA Supercomputer at the Juelich Supercomputing Centre (JSC). For the submussion we use 2 files. `jobscript.sh`, which is similar to the `genSimple4pipsBlocks.sh` scripts used before. In addition, we use `submissionscript.sh` to request particular computing resources and also to pregenerate the GDX input data file.
`submissionscript.sh`:

```
#!/bin/bash -x
#SBATCH --output=mpi-out.%j
#SBATCH --error=mpi-err.%j
#SBATCH --time=01:00:00
#SBATCH --mail-user=name@mail.com
#SBATCH --mail-type=ALL
#SBATCH --partition=batch
#SBATCH --nodes=15
#SBATCH --ntasks=352
SIMPLEPATH=$HOME/simple_v2
GAMSSYSDIR=$HOME/gams/gams25.1_linux_x64_64_sfx
$GAMSSYSDIR/gams $SIMPLEPATH/simple_data_gen --NBREGIONS=20 --WRITEGDX=1
srun jobscript.sh
```

`jobscript.sh`:

```
#!/bin/bash
```

```
mkdir blk_$PMI_RANK
SIMPLEPATH=$HOME/simple_v2
GAMSSYSDIR=$HOME/gams/gams25.1_linux_x64_64_sfx
ARGS="--NBREGIONS=20 --TO=1 --RESOLUTION=0.5 --TBSIZE=50 --METHOD=PIPS "
ARGS+="lo=2 gdxcompress=1 --SLICE=2 keep 1 "
ARGS+="fileStem=simple4pips$PMI_RANK procdir=blk_$PMI_RANK optdir=blk_$PMI_RANK "
echo Generating block $PMI_RANK
$GAMSSYSDIR/gams $SIMPLEPATH/simple4pips $ARGS --BLOCK=$PMI_RANK
if [ $? != 0 ] ; then
 echo Error generating block $PMI_RANK
else
 echo Done generating block $PMI_RANK
fi
rm -rf blk_$PMI_RANK
```

to execute the parallel model generation, we type the following command:

`sbatch submissionscript.sh`

The parallel model generation creates the files `block_noVEnames0` to `block_noVEnames351`.

Chapter 5

Performance analysis of technical performance enhancement

During the duration of the project, PIPS-IPM++ has been continuously tested and debugged on the two HPC platforms at HLRS and JSC (section 4.2). The goal was to achieve better performance than any of the leading commercial solvers, both with respect to shorter run times and the ability to solve large-scale energy system LPs. Specialized tools for large-scale computational systems have been used to perform an in-depth analysis of the code. For instance, they can identify bugs like data races or memory leaks in the code or indicate potential performance and I/O bottlenecks which are limiting the scalability of the code. Those tools are being developed at the HPC centers itself or provided by companies like, among others, Intel or Cray. Due to the high complexity of the supercomputers and the parallel source code, the HPC experts and code developers had to work closely together to use the performance tools in a pertinent way and to interpret the output of the measurements.

Hereinafter it will be described exemplarily how Scalasca (Geimer et al., 2010) and Vampir (`https://vampir.eu/`) have been used to identify a performance bottleneck in PIPS-IPM++.

5.1 Performance and Scalability of PIPS-IPM++

For simplicity, this section only reports on results obtained on the JUWELS supercomputer at JSC; see Section 4.2.5.2 for details on this system.

To demonstrate the scalability of PIPS-IPM++, a (relatively) small-scale SIMPLE instance with 5.6 million variables, 5.1 million constraints, 20.4 million non-zeroes, and 512 diagonal blocks is used. This instance can be handled with a single MPI process in a reasonable time by PIPS-IPM++. Also, the instance allows for a customary power of 2 scaling plot. To achieve good load-balancing, the number of MPI processes should divide the number of blocks—in this way each MPI process is assigned the same number of blocks. The scaling behavior of the commercial LP solvers and PIPS-IPM++ is shown in Figure 5.1. Furthermore, the figure shows the scaling behavior of a reduced version of PIPS-IPM++ using only the Schur complement decomposition, but none of the algorithms additionally implemented and described in Section 4.1.2. For each MPI configuration of PIPS-IPM++, including the reduced version, 2 OpenMP threads were used.

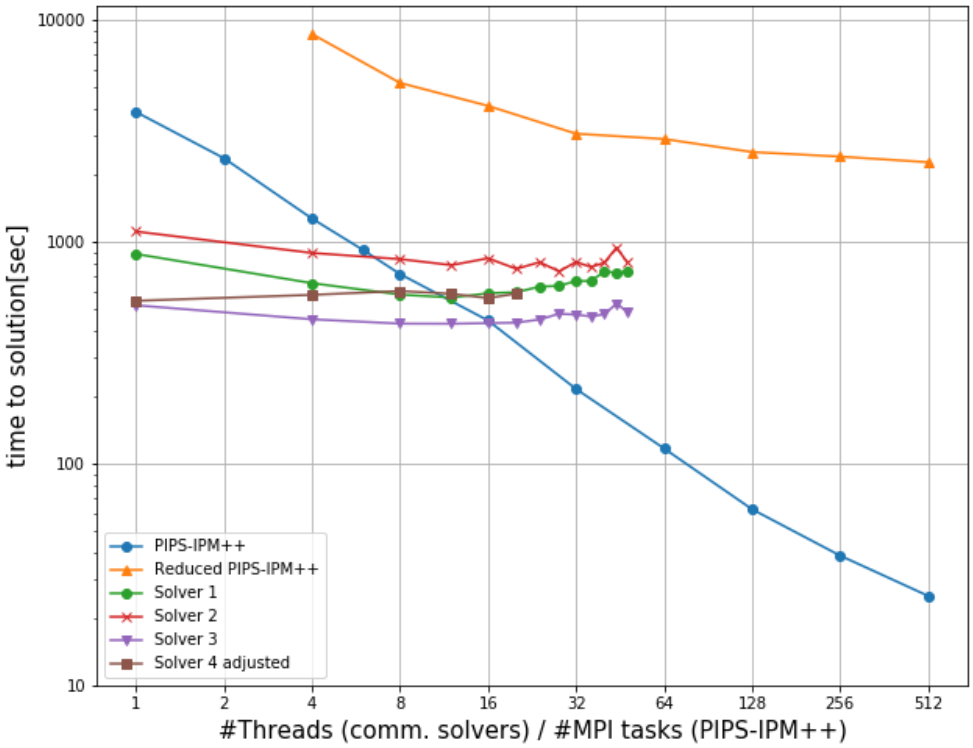


Figure 5.1: Scaling results of leading commercial LP solvers and PIPS-IPM++ on a SIMPLE instance. PIPS-IPM++ was run with 2 OpenMP threads per MPI process.

In the following, some results for large-scale SIMPLE instances as well as for the energy system models REMix and ELMOD are given. The ELMOD instances are used to analyze the impact of growing shares of renewable energies on the operation of the European transmission grid and the dispatch of conventional power plants. The first two ELMOD instances (*CWE*) comprise the entire transmission grid of the Central Western European region. For the other two instances (*EU*) this configuration was further extended to a set of 19 European countries with detailed transmission grid representation. This is the largest configuration currently possible for the ELMOD model and no instance of this configuration could previously be solved. For comparison we use the state-of-the-art (commercial) LP solvers CPLEX 12.8[1], Gurobi 8.1[2], MOSEK 8.1[3] , and Xpress 8.4.7[4]. We use 16 threads for each of the commercial solvers—more threads usually lead to performance degradation. Note that none of the commercial solvers allows for the distributed parallel solution of LPs. Table 5.1 shows results on JUWELS of the new solver (with 2 OpenMP threads per MPI process) and the respective best commercial solver. The measured times include I/O. *OOM* signifies that the commercial solvers ran out of memory, even on the large nodes of JUWELS. Some of the instances intractable on JUWELS could be solved by two of the four commercial solvers on a large shared-memory machine at ZIB with 2 TB RAM and 88 cores. However, the solution required more than one day and incurs

[1] https://www.ibm.com/products/ilog-cplex-optimization-studio
[2] https://www.gurobi.com
[3] https://www.mosek.com/
[4] https://www.fico.com/en/products/fico-xpress-optimization

Table 5.1: Computational results for large-scale instances.

	Size			PIPS-IPM++ resources		Run time (seconds)	
Instance	Variables	Constraints	Non-zeroes	MPI processes	Nodes	PIPS-IPM++	Best commercial
SIMPLE1	336 385 255	304 849 086	1 210 195 025	1024	64	959	OOM
SIMPLE2	1 150 014 619	1 044 894 025	4 174 953 472	1024	128	13 170	–
YSSP_exp	107 555 441	93 902 968	286 477 612	250	25	2311	OOM
YSSP_disp	94 965 730	81 314 846	225 032 421	350	35	931	OOM
ELMOD_CWE15	85 646 554	98 646 274	271 875 064	438	19	181	6 201
ELMOD_CWE16	85 883 074	98 909 074	272 602 144	438	19	216	6 111
ELMOD_EU15	224 677 686	254 304 961	712 452 541	876	38	1 245	OOM
ELMOD_EU16	226 061 766	256 284 723	717 436 984	876	38	1 119	OOM

even far longer waiting times due to limited availability of the machine. The larger SIMPLE instance could not even be generated sequentially due to its size. Thus there was no comparison with any commercial solver possible (since they cannot read in LPs in a distributed way). As to the instances solvable on JUWELS, one notes that the run time per computing resources of the new solver is better than the best result of the commercial solvers (19 compute nodes versus 1 compute node.

5.2 Systematic testing of PIPS++ with REMix instances

For a comparison between the enhanced PIPS-IPM++ solver versus the solvers used prior to the BEAM-ME project for solving the energy system models, a systematic comparison was performed for a large number of different REMix instances. These computations were performed on the hardware acquired within the BEAM-ME project, which is representative for smaller computing clusters at scientific institutions.

There are several factors impacting the performance of the parallel solver PIPS++. To address the different elements in the systematic comparison several instances were chosen for the benchmark runs. These instances can be distinguished in four model dimensions: the spatial scale, the mathematical complexity of investment decisions, the modelling approach for electrical grids and interconnectedness via electrical energy storages.

With respect to the spatial scope of the study we consider three different instances of the same energy system model containing 30, 120 and 488 model regions, which are derived on the basis of the spatial aggregation of the high-resolution input data.

Investment decisions in new grid capacities as well as in new electrical energy storage systems represent an additional degree of freedom. While this additional degree of freedom has little influence on the structure of the underlying mathematical optimization problem, the effort for the optimization algorithm increases significantly. This can be explained by the influence of the investment variables on the development of the company. corresponding hourly deployment decisions.

For the consideration of spatial networking, two different model formulations for electrical transmission networks are compared: A DC transport model (DC-trans) with symmetrical line capacities and losses based on line lengths as well as a DC optimal power flow (DC-OPF) formulation, which takes into account the node-specific feed-in or withdrawal as well as the system-wide voltage angles at each model node. The OPF formulation tends to significantly increase the wall-clock times for the solver due to the very high spatial interconnectedness.

The consideration of tear-and-wear (TaW) of conventional power plants represents the last considered complexity factor for the problem structure. The calculation of the costs associated with TaW requires the hourly difference of the generation variables compared to their previous time step, which leads to significantly more linking constraints in the model.

The results of this systematic evaluation are currently incomplete due to ongoing publication in academic journals. These passages will subsequently included after their corresponding publications are completed.

5.3 Performance Analysis of PIPS-IPM++ with HPC tools

This section describes exemplarily how Scalasca (Geimer et al., 2010) and Vampir (`https://vampir.eu/`) have been used to identify a performance bottleneck in PIPS-IPM++ and how the performance has improved after the distributed preconditioner for the Schur complement has been implemented (see Section 4.1.2).

Test scenario:

- **HPC system**: JUWELS
- **#nodes**: 19
- **#tasks per node**: 16
- **#threads per task**: 2
- **input data**: ELMOD_CWE15
- **code**: version 14th Oct 2019; WITH_DIST_PRECOND (distributed preconditioning) set to OFF or ON
- **compile software**: GCC/8.3.0, ParaStationMPI/5.2.2-1, Boost/1.69.0-Python-3.6.8, imkl/2019.3.199, MUMPS/5.1.2

Before the actual measurement runs could be executed on the HPC system JUWELS the source code had to be instrumented by Score-P (Knüpfer et al., 2012) in the compilation process. To reduce the runtime overhead and the disk space necessary to store the collected data a filter file has been used to filter out functions irrelevant for the analysis.

The GUI Cube (Saviankou, Knobloch, Visser, & Mohr, 2015), which is developed at, among others, JSC, has been used to display the performance data retrieved by Scalasca. Identifying the main bottleneck of PIPS-IPM++ at this stage of the development has been an easy task.

Collective communication operations that send data from all processes to all processes (i.e., n-to-n) exhibit an inherent synchronization among all participants, that is, no process can finish the operation until the last process has started it. The first column in Figure 5.2 shows that around 75% of the total runtime was spent in n-to-n MPI collective operations waiting for other processes to reach the synchronization point. 95% of the waiting time could be traced back to one `MPI_Allreduce` call (second column in Figure 5.2).

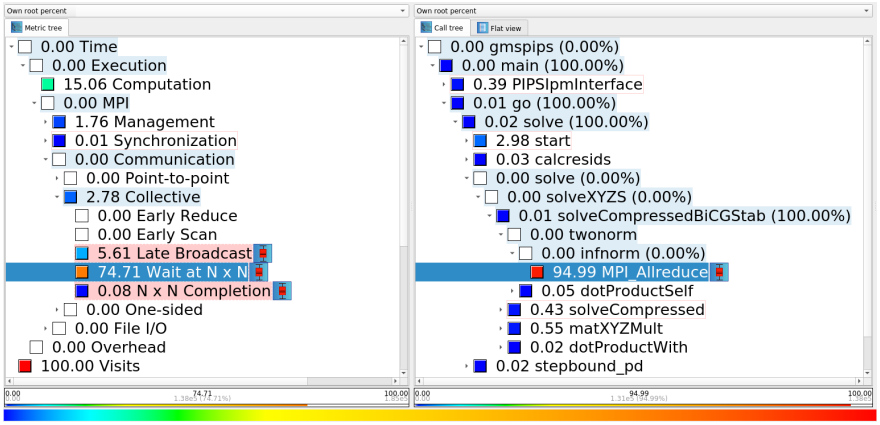


Figure 5.2: Left and middle column of the Cube GUI showing that 75% of the total runtime can be ascribed to 95% to a `MPI_Allreduce` function call.

Besides, there was a massive imbalance in the time the individual process spent within that function call as it can bee seen in the violin plot in Figure 5.3. Switching the view from the violin plot to the system tree view it became obvious that all processes apart from process 0 were having that problem. This means that all processes were waiting for process 0 at this point.

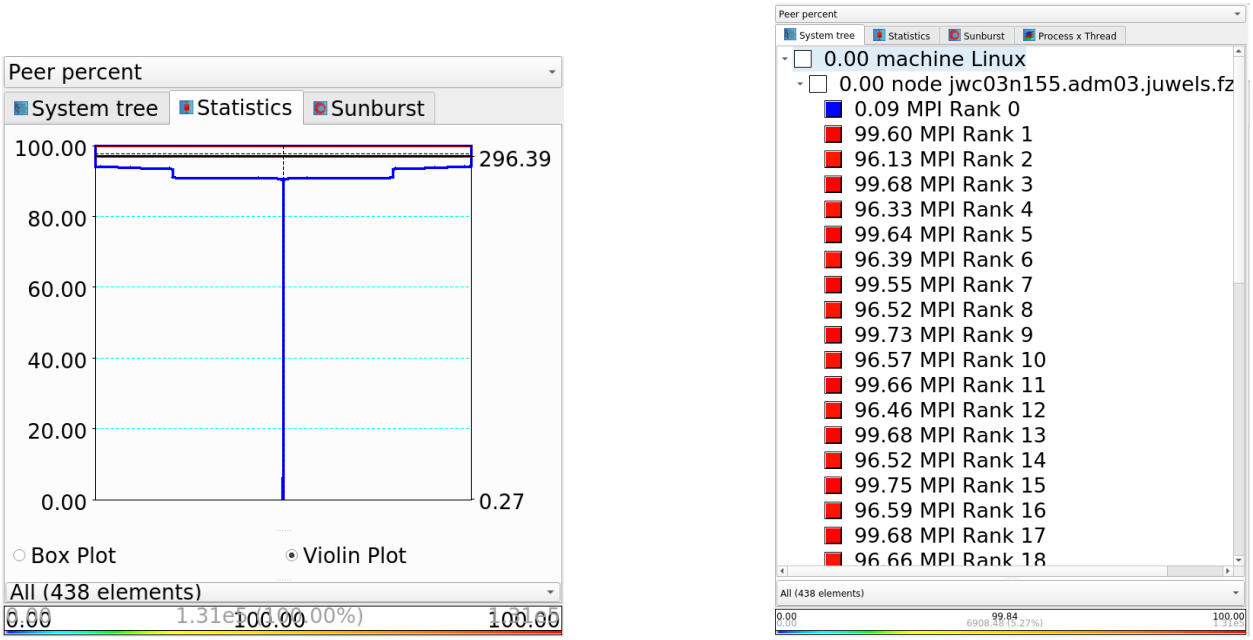


Figure 5.3: Third column of the Cube GUI which belongs to Figure 5.2. The violin plot (left) shows the time distribution of the `MPI_Allreduce` function call among all processes. The system tree view on the right shows all ranks apart from rank 0 experiencing the problem.

The delay costs metric highlights the root causes of wait states. Whereas short-term costs reflect the direct effect of load or communication imbalance on wait states in MPI n-to-n collective communication operations (first column in Figure 5.4). The second column in Figure 5.4 shows that the function `factorize()`, which solves the Schur complement of the LP system, was almost exclusively responsible for the above described

massive imbalance. Looking at the computation time for this function (Figure 5.5) it was is called on MPI rank 0.

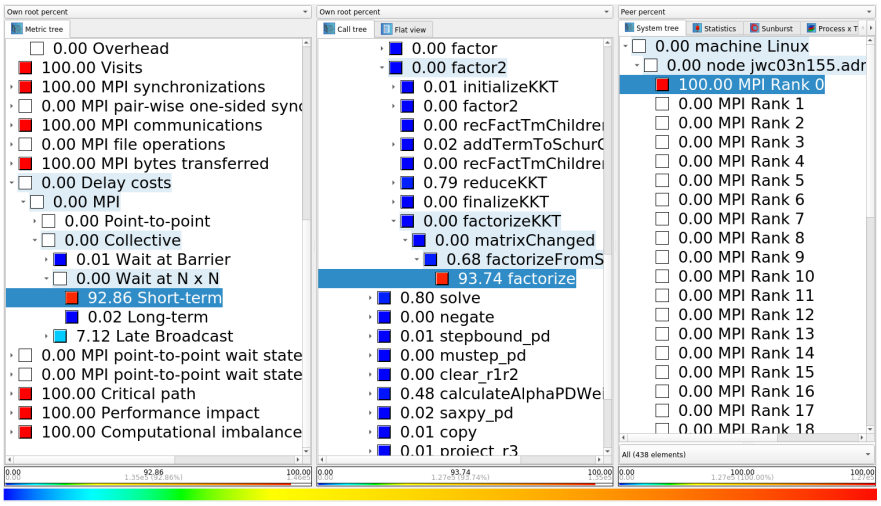


Figure 5.4: The delay costs metric to analyze the imbalance shows the function `factorize()` is the root cause.

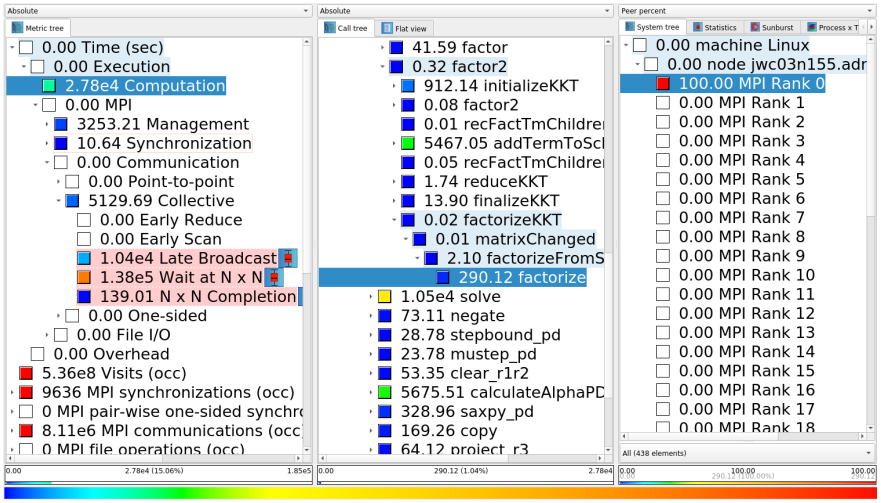


Figure 5.5: Function `factorize()` shows computation time on rank 0 exclusively.

Furthermore the visit count confirms that this function is exclusively called on rank 0 at a total count of 33 (Figure 5.6). PIPS-IPM++ solved this instance within 34 iterations. The first iteration is required to calculate a starting point for the interior point method so that the function `factorize()` is called once in each of the remaining iterations.

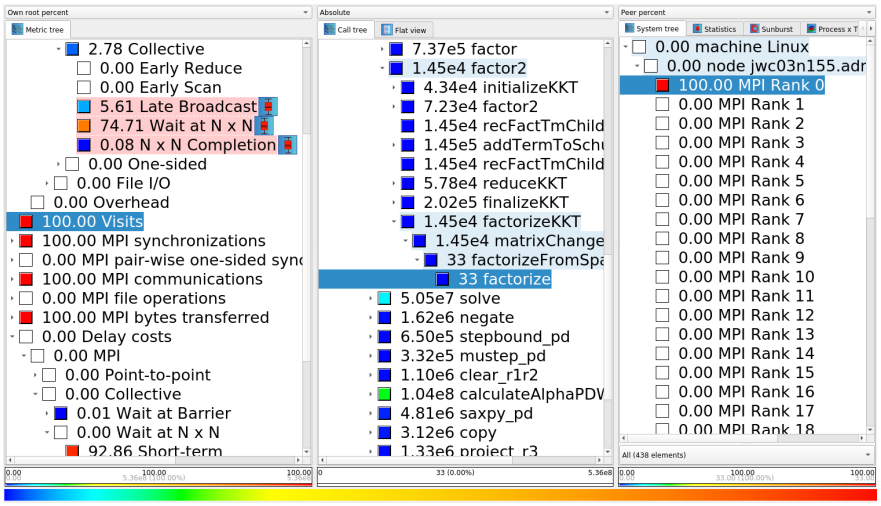


Figure 5.6: Function `factorize()` is called for 33 times (middle column) exclusively on rank 0 (right column).

Last but not least the critical path metric highlights parallel performance bottlenecks. In essence, the critical-path imbalance is the positive difference of the time a call path occupies on the critical path and the call path's average runtime across all CPUs. Thus, a high critical-path imbalance identifies call paths that spend a disproportionate amount of time on the critical path. The function `factorize()` appears significantly on the critical path indicating an imbalance (Figure 5.7).

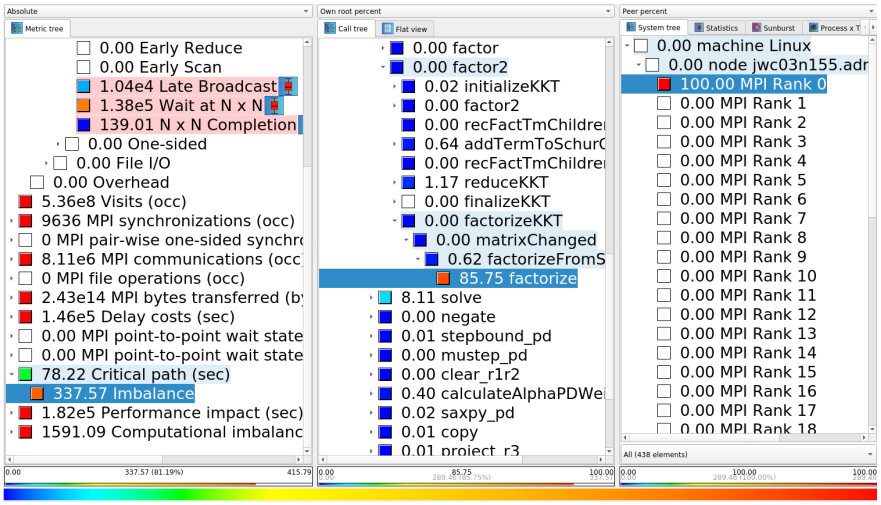


Figure 5.7: The critical path metric indicates that the function `factorize()` contributes significantly to the imbalance.

Summarizing it can be noted that the function `factorize()`, which is called exclusively on rank 0, was causing a big imbalance because all other processes were waiting in a `MPI_Allreduce` call until rank 0 had reached this synchronization point as well. In that, it could be concluded that any optimization applied to this function or any algorithmic change to prevent the described behavior would have an immediate positive effect on the total runtime of the code.

5.3.1 Performance Analysis after optimization

After implementing the distributed preconditioner for the Schur complement the performance of PIPS-IPM++ has improved a lot so that the runtime reduced from 415 to 203 seconds.

The time spent waiting in MPI collective functions has decreased to around 44% of the total runtime (first column in Figure 5.8). As can be seen in the middle column of the same figure it is still the same `MPI_Allreduce` call that is responsible for this delay. It appears again on all ranks apart from rank 0.

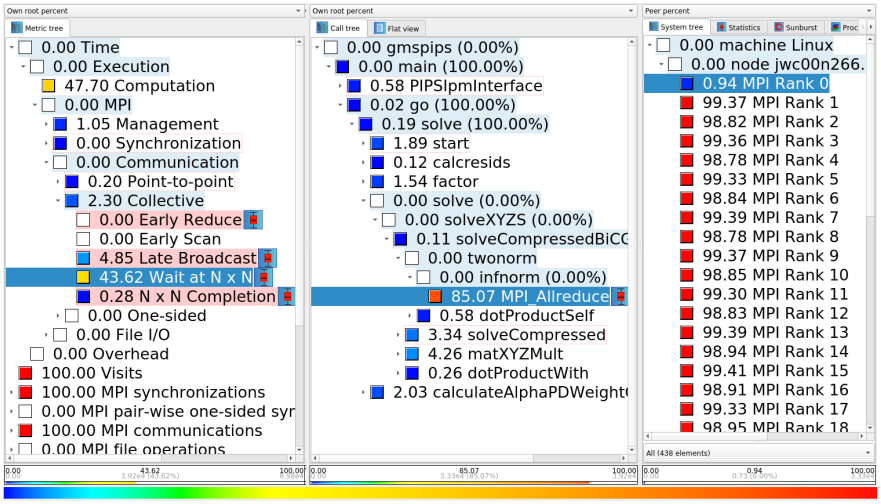


Figure 5.8: Left and middle column showing that around 44% of the total runtime after the optimization can be ascribed to 85% to a `MPI_Allreduce` function call. The system tree view on the right shows that all ranks apart from rank 0 are waiting in this function call.

The critical path imbalance metric indicates that the function `factorize()` is still the dominating part of the code that reduces the overall parallel efficiency, and therefore the speedup (see Amdahl's Law 4.2.2.0.2), of PIPS-IPM++ (Figure 5.9).

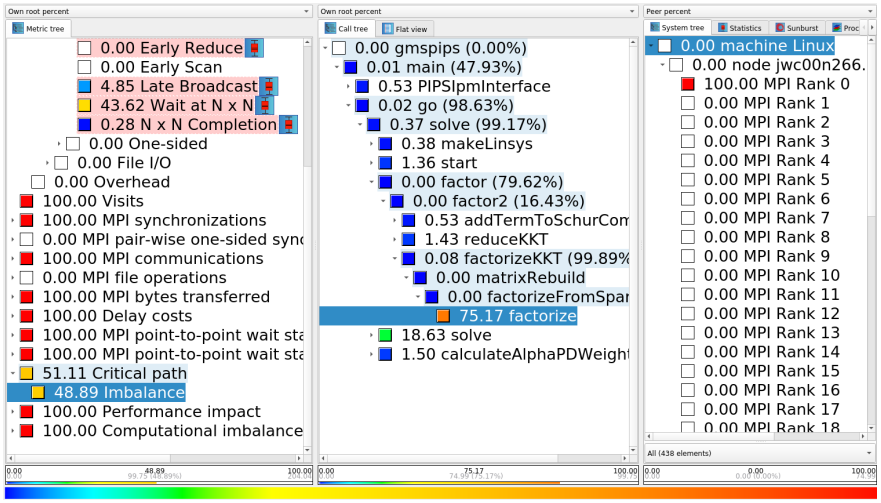


Figure 5.9: The critical path metrics after code optimization still indicates that the function `factorize()` contributes significantly to the imbalance.

5.3.1.0.1 Performance comparison using Vampir

To finalize the analysis Vampir has been used to compare both measurements. In the following plots, the white background shows the data of the optimized version and the grey background the code version without the distributed preconditioner for the Schur complement. On the x-axis, time is plotted. A clustering of the processes with respect to the time spent per function is shown in Figure 5.10. Most of the clusters are quite similarly spending most of their time in a `MPI_Allreduce` function call. Just one process shows a completely different behavior: Little time spend in the `MPI_Allreduce` call and most of the time spend in function `factorize()`.

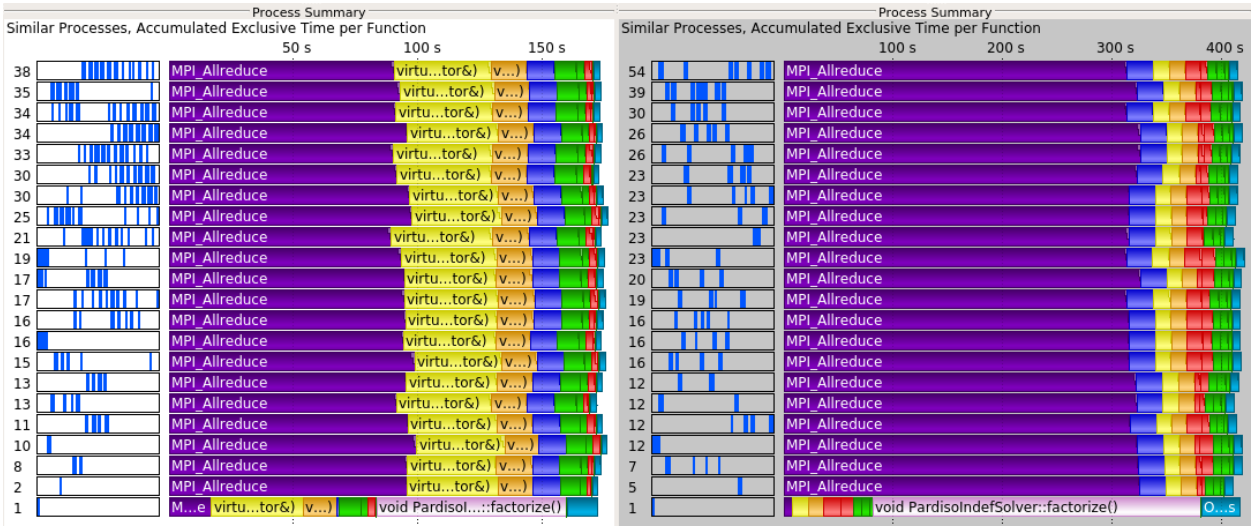


Figure 5.10: Using Vampir clustering the processes with respect to the time spent per function. White background shows the optimized code version. Grey shows the code without the distributed preconditioner for the Schur complement.

Representatively visualizing the first 9 MPI ranks out of 438 MPI ranks in total it can be confirmed that rank 0 is the process that spends most of its time in calling the function `factorize()` while all the processes are waiting for rank 0 to reach the `MPI_Allreduce` function call. This finding applies to both versions of the code under the observation that the runtime of the optimized code version has reduced by a factor of more than 2 (Figure 5.11).

In addition, the ratio between time spent in MPI functions and computation time spent in the application function calls has drastically improved in the optimized version (Figure 5.12). Thus the parallel efficiency has improved as well such that better scalability of PIPS-IPM++ can be expected.

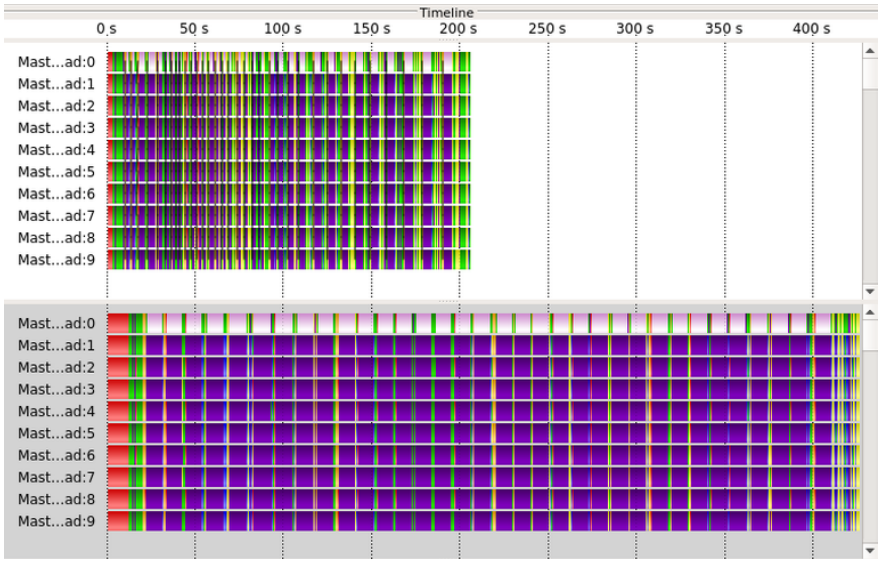


Figure 5.11: Comparing the time line of the optimized code version (white background) against the code version without distributed preconditioner (grey) for the Schur complement representatively for the first 9 ranks.

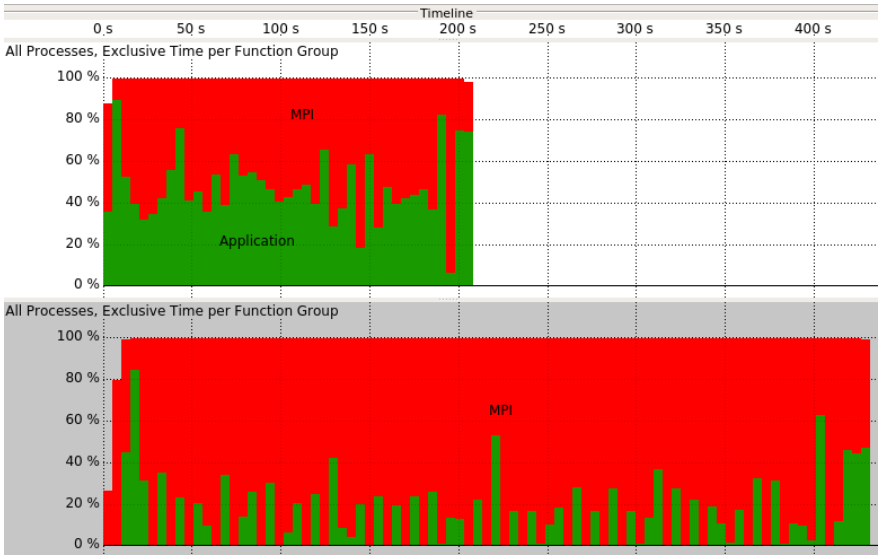


Figure 5.12: Comparing the runtime spent in the functions of the application (green) against the time spent in MPI calls (red).

5.3.1.0.2 Energy efficiency of PIPS-IPM++

As the size of high-performance computing systems continues to grow, energy consumption becomes a serious limitation problem. This challenge is becoming more and more demanding in HPC: Already now costs over lifetime of an HPC facility often exceed acquisition costs. The efficient planning and operation of such large-scale systems depends crucially on the HPC software developers and HPC users developing an understanding of it.

This section shows how the choosing of execution parameters for PIPS-IPM++ affects the energy efficiency of the HPC systems. Unlike the previous scaling experiment, the instance is distributed to the MPI processors so that each process receives only one block. For this, the block size must be varied. We performed this kind of scaling experiment on HPC System „Hazel Hen“. „Hazel Hen“ provides the power consumption of used compute nodes. Hence, in addition to the solution time, the energy costs are shown in Figure 5.13. As you can see, the considered instance is solved with 240 MPI processors and 2 OMP threads (480 cores, 20 compute nodes) in almost the same time as with 522 MPI processes and 2 OMP threads (1044 cores, 44 compute nodes). However, the solution with 522 MPI processes consumes twice as much resources (number of compute nodes and energy).

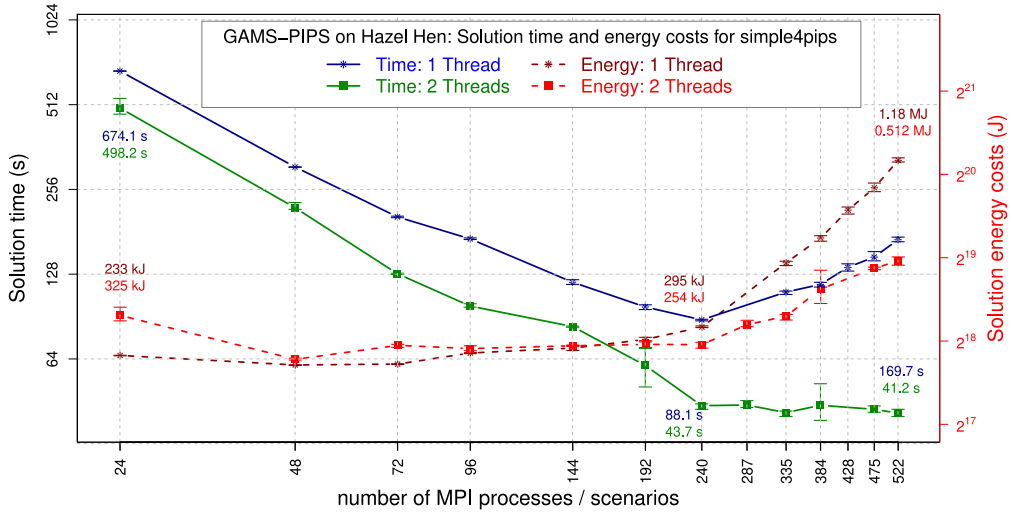


Figure 5.13: Solution Time and energy costs of the simple4pips problem with parameters –TO=0.992 (–TBSIZE=51) –NBREGIONS=20 –RESOLUTION=0.33333 OMP_THREADS=1 and 2. Im contrast to Figure 5.1 each of the processes owns only one block. The problem was solved on HPC System „Hazel Hen“.

5.3.1.0.3 Conclusion of performance comparison

With the assistance of different HPC performance measurement tools, the HPC experts and code developers gained an in-depth understanding of the runtime behavior of PIPS-IPM++ so that the functionality and the parallel performance of the application could be improved at the same time.

It is also necessary for users to spend time for scaling experiments with their ESOM to find the optimal executing configuration for their models. Using the optimal configuration will help to save expensive HPC resources, which can be used by the user for further modeling and finding the optimal solutions, as well as by other users to solve their problems.

Depending on the performance of the supercomputer network and compute nodes, we recommend that users use a minimum number of blocks per process, such as one as shown in this section. It is also important to find the optimal block size and number of OMP threads. Unfortunately, we don't have any equation that could provide these parameters, because it depends on too many factors, from ESOM properties to supercomputer performance. However, we believe that the information contained in the Best Practice Guide will help the users of PIPS-IPM++ to overcome this challenge.

Part III

Results of the model experiment

Chapter 6

Performance analyses in the model experiment

The chapter of the performance analysis of model experiment is currently incomplete due to ongoing publication in academic journals. These passages will be subsequently included after their corresponding publications are completed.

6.1 Concept of the model experiment

A key objective of the project BEAM-ME was to ensure the transferability of the project results. A „model experiment “was set-up to meet this objective. The methods and solvers developed in this project were tested on REMix and six state of the art energy system models currently used by renowned research institutions. The models differ in scope and focus and are applied to various different research questions: Models by some partners focus on single scenario years, investigate the operation and dispatch. These models have a strong emphasis on grid as well as power plant operation. Other models focus on long term investments in the energy system, with a focal point on the transition path and the long term development of different technologies. The model experiment partners contributed their models and know-how to the project in two fields. On the one hand they helped to develop best practice on model based speed-up methods. Different ways to reduce the complexity of the model on the modeler's side were compared. On the other hand the models adapted their models for application on distributed computing systems. The group learned to apply the concept of “model annotation”. All models were successfully annotated. Then models and respective scenarios were then solved on the high performance computing facility in Juelich (JURECA). Some models showed significant speed-up when solved on an HPC machine. Some models however could not be used with the newly developed PIPS IPM++ solver, due to the inherent structure of the models.

6.2 Lessons learned from annotation

Model annotation was introduced as a new feature in the modeling language developed by the project partner GAMS. In the annotation phase all model experiment partners were able to use the new methods developed in this project. In order to decompose the power system models for parallel computing, the models had to be annotated in a ways the resulting different blocks/sub-problems were suitable for computation on the distributed nodes. The six participants had to learn to develop models with a high number of blocks (to be distributed on a high number of cores) and at the same time not to increase the complexity of the sub-problems as well as the number of linking variables between the sub-problems, so that the new algorithms can still solve these problems. The annotation phase showed that annotating a model itself is not a complex task. Within a short time frame of a few days complex energy system models could be annotated. It turned

out that the partners were able to perform a first annotation of the models after a short training. While the implementation of the annotation is rather straight forward, finding a good structure for HPC is a difficult task. Structuring the model in a way it can be divided in numerous subproblems is not yet automated. To perform this task it requires a lot of knowhow by the modeler about the model itself. In this project it took months to eliminate the errors and to make the annotation generic enough to allow a variation of the annotated block structure. At first problems arose from identifying equations and variables that were not properly annotated. Further it was difficult to split the model in ways such that all resulting blocks would be of similar size. The tools developed within this project, the graphical representation and the CheckAnnot tool, proofed to be of great help when developing and testing the annotation. Especially the graphical representation provided a good intuitive understanding of the model structure and allowed to easily find errors in the annotation of the models. The lessons learned from the annotation phase in Chapter xx aim to facilitate the future work of fellow modelers, when adopting their models for HPC.

6.3 Solving energy system models on HPC

Overall, the model experiment showed that energy system models in general tend to be "very interconnected" and showed a large number of linkage variables between the divided blocks. Further the partners learned, that slicing the model into different timesteps seems to be the best way forward and is much easier, then dividing the model by regions or by technologies. Some partners were successfully able to divide the model both by time and regions. The model experiment partners provided different instances of their models to the project. The instances differed in size and structure. Model instances were executed on a fat node on the JSC computer JURECA to provide a basis for model comparison and benchmark analysis with distributed computation (based on PIPS) and conceptual acceleration methods. Model runs on fat nodes showed significant problems in scaling the model variables, which varied over several decades. This had a negative impact on the numerical stability. This led to long computing times for solving the models and limited validity of the results. With an improved formulation the solver was able to find faster and more reliable solutions. These scaling problems had a significant impact on distributed computing runs with the PIPS solver. Therefore, the model experiment partners undertook considerable efforts to analyze all variables in their energy system models and to adapt them if necessary. The high number of linking constraints in the model and to some extend scaling issues prevented some models from successful model runs on HPC. PIPS IPM++ still faces difficulties coping with these problems for some model instances. Other partners showed however significant improvements of computation time. For the large EU instances ELMOD model runs with PIPS-IPM++ could be solved in less than 20 minutes. Three out of four solvers could not solve these model instance on large memory machines. Other conventional solvers required between 18 and 24 hours to solve the problem. All in all acceleration of speed-up by a factor of 50-70 could be observed. This is the case already for considerable small number of computing nodes (38). Further it could be observed that the problem scales well. By increasing the number of computing run time of pips does not increase significantly for more complex problems. The model runs on ELMOD showed that the models scale very well on HPC. The size of the model instances can increase significantly. By increasing the blocks and distributing the individual subproblems to more nodes, they can also be calculated comparatively quickly. However, the model runs also showed that large instances are very time-consuming to create. The considerations of a distributed generation of model instances, which were previously presented in the report, can make an important contribution in the follow-up to the project to further accelerate the overall process of solving energy system models on HPC computers. The investment model EUSTEM was able to reduce the computation time by a magnitude of 2.5 for their model runs.

6.4 Performance analyses of modeling based strategies in the model experiment

Intensive effort has been done by the modelling community to bridge the gap between model complexity, computational effort and model accuracy. All in all the model experiment shows that for model based speed-

up methods – concepts to reduce number of time steps in the model or concepts to divide (rolling horizon and rolling planning approaches) seem to be very suitable for application in energy system models. The developed methods aim to reduce the drawbacks of reducing complexity with regard to model accuracy to a large extend. If no HPC is available for detailed simulation time based methods for reduction of model complexity seem to be a promising compromise.

6.5 Conclusions of the model experiment

The BEAM-ME project and the model experiment aimed to improve future energy system models, by helping to reduce computation time and allow for more complex energy system models, facilitating and improving the work of energy system modelers. A series of tools and methodologies were developed and existing methods refined with the aim to reduce the computation time and simplify the work required for energy system modelers. The project gained significant insights and skills from the model experiment partners. Continuous feedback on the developed solvers, methods and tools helped to refine and improve the results of the project. The model experiment allowed conducting the praxis-test right during the project time. By including the six MEXT partners knowledge could be disseminated at an early stage. One feedback from the model experiment partners presented at the EURO 2019 in Dublin states: "Solving the model with PIPS-IPM needs a high degree of parallelization but the expected reduction in the solution time is worth the effort of annotation".

Part IV

Conclusion

Chapter 7

Overall Conclusions from the BEAM-ME Project

Energy systems analysis highly depends on modeling tools such as Energy System Optimization models (ESOMs). To fulfill their purpose to provide insights into complex energy systems for decision support they need to be solvable within acceptable time spans. Against this background, the central goal of the BEAM-ME project was the significant reduction of solution times for energy system models formulated as linear optimization problems in GAMS (General Algebraic Modeling System) and the solution of previously unsolvable such problems. To achieve this goal, different methods were evaluated and their effects were compared in a model experiment for differently formulated and focused models.

The two central pillars of model acceleration are on the one hand the application of modelling based methods and on the other hand the use of solution algorithm based methods. The modelling methods essentially comprise different approaches of spatial and temporal aggregation, as well as heuristic and mathematically exact problem decomposition. They represent the main part of the acceleration strategies used so far for energy system optimization models and offer the advantage that they can usually be developed and implemented by model users themselves by adapting source codes or pre-processing input data. Within BEAM-ME they were implemented and evaluated mainly in the energy system model REMix, but also in other models. These measures were accompanied by comprehensive benchmark analyses for a set of frequently applied speed-up techniques. The conducted examination included model aggregation approaches on different scales as well as strategies for heuristic decomposition. In addition to the reduction of the solution time, the effect on the accuracy of the results was of essential importance. As a central result of the evaluation of model-based methods it could be shown that with reasonable restrictions regarding the accuracy, heuristics can reduce the computing time of large models up to a factor of 10. Within our evaluation framework, temporal down-sampling turned out to be the most efficient speed-up approach. The usefulness of this approach is strongly related to the "effectiveness of model reduction". In other words, the larger and more difficult to solve a particular ESOM becomes, the greater the achievable speed-up by already minor model reductions is. Taking into account that solving of linear ESOMs with mixed-integer variables is more complicated than for the model instances considered in this study, we suppose that the presented speed-up approaches are especially effective for such use cases.

As far as only specific model outcomes such as additional transmission capacities are of interest and extensive multi-threading is possible, the presented heuristic decomposition approaches with grid computing (temporal zooming) are also promising as they allow additional speed-ups without increasing loss of accuracy. Moreover, they offer the possibility for executing an ESOM on multiple shared memory computers even though parallelization is only applied to the conceptual layer of the optimization model.

Nevertheless, we showed that the appropriate gains in performance are limited depending on the size of a certain model. In this case, the down-side of "effectiveness of model reduction" comes into play: Since the

idea behind decomposition is based on solving multiple reduced sub-models, such approaches reach their speed-up limit when the decrease of computing time by model reduction becomes negligible for very small sub-models.
The core of the investigation of algorithm based methods was the development of the solver PIPS-IPM++, which allows the parallel solution of linear optimization problems on high performance computer architectures. It is based on the open-source solver PIPS-IPM and has been extensively developed within BEAM-ME to enable its use for typical problems of power system optimization. This problem category is characterized by many so-called linking constraints, which are necessary for modeling networks, storage and CO_2 emission constraints. For this purpose, the solver has been adapted in such a way that it can process not only linking variables but also a large number of linking constraints in parallel. For a better integration an interface between GAMS and PIPS-IPM++ was developed within the project. The basis for this is the also newly created possibility of annotating block structures in GAMS models, which allows a decomposition of very large problems into many small blocks, which in turn is indispensable for a parallel application of the solution algorithms. With the development of PIPS-IPM++ and the adaptation of the energy system models, the use of High Performance Computers (HPC) for energy system analysis was opened up.

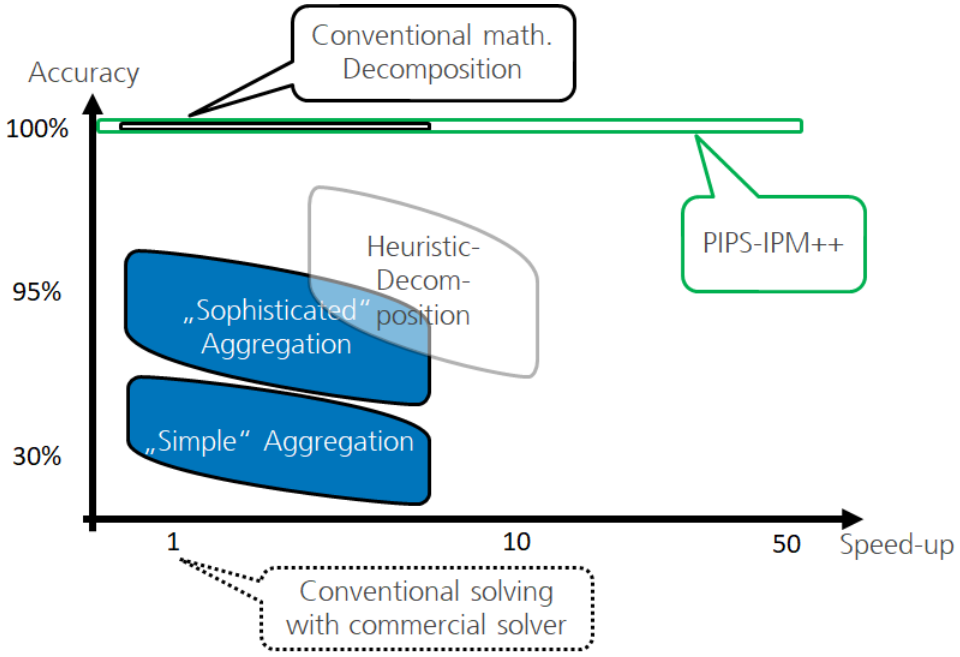


Figure 7.1: Relationship between accuracy and computing time for different acceleration approaches investigated

The progress achieved within BEAM-ME allows a reduction of the solution time of linear optimization problems of energy system analysis up to a factor of 26. Furthermore, previously unsolvable problems have become solvable by efficient problem decomposition and the use of PIPS-IPM++. The achieved progress on the technical side can be illustrated by a typical scientific computing cluster. On such systems a reduction of the runtime by 76% and a reduction of the required memory per node by 96% could be achieved by extending PIPS. In comparison, on the HPC mainframe Juwels at FZJ the runtime was reduced by 96% for one of the largest REMix models. These advances allow a significant expansion of the analysis depth of energy system models. For REMix, the accessible spatial resolution was increased from about 50 to more than 1000 model nodes, the optimization of transformation paths instead of individual reference years was realized, the consideration of further technologies and sectors was made possible, and the possibility of large-scale

parameter space analyses was created. In this manner, the application of effective speed-up approaches highly contributes to the generation of robust and well-founded model-based analyses for the development of decarbonization strategies of the energy system.
The BEAM-ME project met with broad interest in the scientific community, which was manifested in particular by a great interest in participating in the model experiment, manifold inquiries and lively participation in the final workshop.

Acknowledgement

The authors thank the Bundesministerium für Wirtschaft und Energie (BMWi) for funding the BEAM-ME project (FKZ 03ET4023). We are also thankful for the technical and administrative support provided by Projektträger Jülich (PTJ).
Furthermore, the authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC) and at the High Performance Computing Centre Stuttgart (HLRS).

Bibliography

Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., & Weninger, D. (2019). Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*. (accepted for publication 2018-08-31)

Achterberg, T., et al. (2016). *Presolve reductions in mixed integer programming* (ZIB-Report No. 16-44). Zuse Institute Berlin.

Agner. (2017). *Test results for broadwell and skylake.* Retrieved from https://www.agner.org/optimize/blog/read.php?i=415

Alguacil, N., & Conejo, A. (2000). Multiperiod optimal power flow using Benders decomposition. *IEEE Transactions on Power Systems*, *15*(1), 196-201. Retrieved from http://dx.doi.org/10.1109/59.852121 doi: 10.1109/59.852121

Andersen, E. D., & Andersen, K. D. (1995). Presolving in linear programming. *Mathematical Programming*, *71*(2), 221–245.

Angela Flores Quiroz, Rodrigo Palma Behnke, Golbon Zakeri, & Rodrigo Moreno. (2016). A column generation approach for solving generation expansion planning problems with high renewable energy penetration. *Electric Power Systems Research*, *136*, 232 - 241. Retrieved from http://www.sciencedirect.com/science/article/pii/S0378779616300177 doi: http://dx.doi.org/10.1016/j.epsr.2016.02.011

Babrowski, S., Heffels, T., Jochem, P., & Fichtner, W. (2013). Reducing computing time of energy system models by a myopic approach. *Energy Systems*, 1-19. Retrieved from http://dx.doi.org/10.1007/s12667-013-0085-1 doi: 10.1007/s12667-013-0085-1

Baños, R., Manzano-Agugliaro, F., Montoya, F., Gil, C., Alcayde, A., & Gómez, J. (2011). Optimization methods applied to renewable and sustainable energy: A review. *Renewable and Sustainable Energy Reviews*, *15*(4), 1753-1766. Retrieved from https://EconPapers.repec.org/RePEc:eee:rensus:v:15:y:2011:i:4:p:1753-1766

Barth, R., Brand, H., Meibom, P., & Weber, C. (2006, June). A stochastic unit-commitment model for the evaluation of the impacts of integration of large amounts of intermittent wind power. In *2006 international conference on probabilistic methods applied to power systems* (p. 1-8). doi: 10.1109/PMAPS.2006.360195

Bethany A. Corcoran, Nick Jenkins, & Mark Z. Jacobson. (2012). Effects of aggregating electric load in the United States. *Energy Policy*, *46*, 399 - 416. Retrieved from http://www.sciencedirect.com/science/article/pii/S0301421512002844 doi: http://dx.doi.org/10.1016/j.enpol.2012.03.079

Bethany A. Frew, Sarah Becker, Michael J. Dvorak, Gorm B. Andresen, & Mark Z. Jacobson. (2016). Flexibility mechanisms and pathways to a highly renewable {US} electricity future. *Energy*, *101*, 65 - 78. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360544216300032 doi: http://dx.doi.org/10.1016/j.energy.2016.01.079

Binato, S., Pereira, M., & Granville, S. (2001, May). A new Benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems*, *16*(2), 235-240. Retrieved from http://dx.doi.org/10.1109/59.918292 doi: 10.1109/59.918292

Brady Stoll, Gregory Brinkman, Aaron Townsend, & Aaron Bloom. (2016). *Analysis of Modeling Assumptions used in Production Cost Models for Renewable Integration Studies* (Tech. Rep.). {National Renewable Energy Laboratory (NREL)}. Retrieved from https://www.nrel.gov/docs/fy16osti/65383.pdf

Breuer, T. (2019). *Contribution of hpc to the beam-me project. in the beam-me project. implementation of acceleration strategies from mathematics and computational sciences for optimizing energy system models.*

Breuer, T., Bussieck, M., Cao, K.-K., Cebulla, F., Fiand, F., Gils, H. C., ... Wetzel, M. (2018). Optimizing large-scale linear energy system problems with block diagonal structure by using parallel interior-point methods. In N. Kliewer, J. F. Ehmke, & R. Borndörfer (Eds.), *Operations research proceedings 2017* (pp. 641–647). Cham: Springer International Publishing.

Brouwer, A. S., van den Broek, M., Zappa, W., Turkenburg, W. C., & Faaij, A. (2016). Least-cost options for integrating intermittent renewables in low-carbon power systems. *Applied Energy*, *161*, 48 - 74. Retrieved from http://www.sciencedirect.com/science/article/pii/S0306261915012167 doi: http://dx.doi.org/10.1016/j.apenergy.2015.09.090

Bryan Stephen Palmintier. (2013). *Incorporating operational flexibility into electric generation planning: impacts and methods for system design and policy analysis* (Unpublished doctoral dissertation). Massachusetts Institute of Technology.

Bussar, C., Stöcker, P., Cai, Z., Luiz, M., Leuthold, M., Sauer, D., ... Moser, A. (2015, 03). Large-scale integration of renewable energies and impact on storage demand in a european renewable power system of 2050. In (Vol. 73). doi: 10.1016/j.egypro.2015.07.662

Cao, K.-K., Gleixner, A., & Miltenberger, M. (2016). Methoden zur reduktion der rechenzeit linearer optimierungsmodelle in der energiewirtschaft - eine performance-analyse. In *Eninnov 2016: 14. symposium energieinnovation 2016.*

Cao, K.-K., Metzdorf, J., & Birbalta, S. (2018, Juni). Incorporating power transmission bottlenecks into aggregated energy system models. *Sustainability*, *10*(6). Retrieved from https://elib.dlr.de/120656/

Cebulla, F., & Fichter, T. (2017, May). Merit order or unit-commitment: How does thermal power plant modeling affect storage demand in energy system models? *Renewable Energy*, *105*, 117 - 132. Retrieved from http://www.sciencedirect.com/science/article/pii/S0960148116310990 doi: http://dx.doi.org/10.1016/j.renene.2016.12.043

Cerebras. (2019, August). *The wse is the largest chip ever built.* Cerebras Systems. Retrieved from https://www.cerebras.net/ (Accessed: 2018-08-20)

Chandrakasan, A. P., Sheng, S., & Brodersen, R. W. (1995). Low power cmos digital design. *IEEE JOURNAL OF SOLID STATE CIRCUITS*, *27*, 473–484.

Christian Bussar, Philipp StÃ¶cker, Zhuang Cai, Luiz Moraes Jr., Dirk Magnor, Pablo Wiernes, ... Dirk Uwe Sauer (2016). Large-scale integration of renewable energies and impact on storage demand in a European renewable power system of 2050â€"Sensitivity study. *Journal of Energy Storage*, *6*, 1 - 10. Retrieved from http://www.sciencedirect.com/science/article/pii/S2352152X16300135 doi: http://dx.doi.org/10.1016/j.est.2016.02.004

Chulwoo Kim, Hyun-Woo Lee, & Junyoung Song. (2014). *High-bandwidth memory interface.* Springer. Retrieved from http://dx.doi.org/10.1007/978-3-319-02381-6 doi: 10.1007/978-3-319-02381-6

ColfaxInternational. (2017). *Capabilities of intel avx-512 in intel xeon scalable processors (skylake).* Retrieved from https://colfaxresearch.com/skl-avx512/

Colombo, M., & Gondzio, J. (2008, Dec 01). Further development of multiple centrality correctors for interior point methods. *Computational Optimization and Applications*, *41*(3), 277–305. Retrieved from https://doi.org/10.1007/s10589-007-9106-0 doi: 10.1007/s10589-007-9106-0

Conejo, A. J., Castillo, E., Mínguez, R., & García Bertrand, R. (2006). *Decomposition Techniques in Mathematical Programming.* Springer Science + Business Media. Retrieved from http://site.ebrary.com/lib/dlr/detail.action?docID=10130102 doi: 10.1007/3-540-27686-6

Cook, W. (2018). *The traveling salesman problem.* Retrieved from http://www.math.uwaterloo.ca/tsp/

Corporation, I. (2017). *Ibm ilog cplex optimization studiocplex user's manual.*

Daniel Molka, Daniel Hackenberg, Robert Schöne, & Wolfgang E. Nagel. (2015). Cache coherence protocol and memory performance of the intel haswell-ep architecture. *2015 44th International Conference on Parallel Processing*, 739-748.

Deane, J., Drayton, G., & Gallachóir, B. Ó. (2014). The impact of sub-hourly modelling in power sys-

tems with significant levels of renewable generation. *Applied Energy*, *113*, 152 - 158. Retrieved from http://www.sciencedirect.com/science/article/pii/S030626191300593X doi: http://dx.doi.org/10.1016/j.apenergy.2013.07.027

Deckmann, S., Pizzolante, A., Monticelli, A., Stott, B., & Alsac, O. (1980, Nov). Studies on power system load flow equivalencing. *IEEE Transactions on Power Apparatus and Systems*, *PAS-99*(6), 2301-2310. doi: 10.1109/TPAS.1980.319798

Dorfler, F., & Bullo, F. (2013, Jan). Kron Reduction of Graphs With Applications to Electrical Networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *60*(1), 150-163. doi: 10.1109/TCSI.2012.2215780

Egerer, J., Gerbaulet, C., Ihlenbur, R., Kunz, F., Reinhard, B., von Hirschhausen, C., ... Weibezahn, J. (2014). *Electricity Sector Data for Policy-Relevant Modeling: Data Documentation and Applications to the German and European Electricity Markets* (Data Documentation No. 72). DIW Berlin, German Institute for Economic Research. Retrieved from https://ideas.repec.org/p/diw/diwddc/dd72.html

Elble, J. M., & Sahinidis, N. V. (2012). Scaling linear optimization problems prior to application of the simplex method. *Computational Optimization and Applications*, *52*(2), 345–371. doi: 10.1007/s10589-011-9420-4

ENTSO-E. (2012). *Transparency platform cross-border commercial schedule and cross-border physical flow.* Retrieved from https://transparency.entsoe.eu/content/static_content/Static%20content/legacy%20data/legacy%20data2012.html

Esmaili, M., Ebadi, F., Shayanfar, H. A., & Jadid, S. (2013). Congestion management in hybrid power markets using modified benders decomposition. *Applied Energy*, *102*(Supplement C), 1004 - 1012. Retrieved from http://www.sciencedirect.com/science/article/pii/S030626191200462X (Special Issue on Advances in sustainable biofuel production and use - XIX International Symposium on Alcohol Fuels - ISAF) doi: https://doi.org/10.1016/j.apenergy.2012.06.019

Eurostat. (2017, 10). *European commission eurostat. nuts - nomenclature of territorial units for statistics. overview.* Retrieved from http://ec.europa.eu/eurostat/web/nuts/overview

Fallah, F., & Pedram, M. (2005, 04). Standby and active leakage current control and minimization in cmos vlsi circuits. *IEICE Transactions*, *88-C*, 509-519. doi: 10.1093/ietele/e88-c.4.509

Frank, S., Steponavice, I., & Rebennack, S. (2012, Sep 01). Optimal power flow: a bibliographic survey i. *Energy Systems*, *3*(3), 221–258. Retrieved from https://doi.org/10.1007/s12667-012-0056-y doi: 10.1007/s12667-012-0056-y

GA. (2019). *Gauß-allianz.* Retrieved from https://gauss-allianz.de

GCS. (2019). *Gauss center for supercomputing.* Retrieved from https://www.gauss-centre.eu

Geimer, M., Wolf, F., Wylie, B. J. N., Ábrahám, E., Becker, D., & Mohr, B. (2010, April). The Scalasca performance toolset architecture. *Concurrency and Computation: Practice and Experience*, *22*(6), 702–719. doi: 10.1002/cpe.1556

Georg Hager, Jan Treibig, Johannes Habich, & Gerhard Wellein. (2012). Exploring performance and power properties of modern multicore chips via simple machine models. *CoRR*, *abs/1208.2908*. Retrieved from http://arxiv.org/abs/1208.2908

Gils, H. C. (2015). *Balancing of Intermittent Renewable Power Generation by Demand Response and Thermal Energy Storage* (Doctoral dissertation, Universität Stuttgart). doi: http://dx.doi.org/10.18419/opus-6888

Gils, H. C., Bothor, S., Genoese, M., & Cao, K.-K. (2018). Future security of power supply in germany—the role of stochastic power plant outages and intermittent generation. *International Journal of Energy Research*, *42*(5), 1894-1913. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/er.3957 doi: 10.1002/er.3957

Gils, H. C., Scholz, Y., Pregger, T., de Tena, D. L., & Heide, D. (2017). Integrated modelling of variable renewable energy-based power supply in europe. *Energy*, *123*, 173 - 188. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360544217301238 doi: https://doi.org/10.1016/j.energy.2017.01.115

Gils, H. C., & Simon, S. (2017). Carbon neutral archipelago – 100energy supply for the canary islands.

Applied Energy, *188*, 342 - 355. Retrieved from http://www.sciencedirect.com/science/article/pii/S0306261916317871 doi: https://doi.org/10.1016/j.apenergy.2016.12.023

Gils, H. C., Simon, S., & Soria, R. (2017). 100brazil—the role of sector coupling and regional development. *Energies*, *10*(11). Retrieved from https://www.mdpi.com/1996-1073/10/11/1859

Gleixner, A., Kempke, N.-C., Koch, T., Rehfeldt, D., & Uslu, S. (2019). *First experiments with structure-aware presolving for a parallel interior-point method* (Tech. Rep. No. 19-39). Takustr. 7, 14195 Berlin: ZIB.

Gondzio, J. (1997). Presolve analysis of linear programs prior to applying an interior point method. *INFORMS Journal on Computing*, *9*(1), 73-91.

Green, R., Staffell, I., & Vasilakos, N. (2014, May). Divide and Conquerk-Means Clustering of Demand Data Allows Rapid and Accurate Simulations of the British Electricity System. *IEEE Transactions on Engineering Management*, *61*(2), 251 - 260. Retrieved from http://dx.doi.org/10.1109/TEM.2013.2284386 doi: 10.1109/tem.2013.2284386

Haas, J., Cebulla, F., Cao, K., Nowak, W., Palma Behnke, R., Rahmann, C., & Mancarella, P. (2017). Challenges and trends of energy storage expansion planning for flexibility provision in low-carbon power systems – a review. *Renewable and Sustainable Energy Reviews*, *80*, 603 - 619. Retrieved from http://www.sciencedirect.com/science/article/pii/S1364032117308377 doi: https://doi.org/10.1016/j.rser.2017.05.201

Habibollahzadeh, H., & Bubenko, J. A. (1986, Feb). Application of Decomposition Techniques to Short-Term Operation Planning of Hydrothermal Power System. *IEEE Transactions on Power Systems*, *1*(1), 41 - 47. doi: 10.1109/TPWRS.1986.4334842

Hager, G., & Wellein, G. (2016). *Introduction to high performance computing for scientists and engineers, second edition.* Taylor & Francis.

Haikarainen, C., Pettersson, F., & Saxen, H. (2016, Feb). A decomposition procedure for solving two-dimensional distributed energy system design problems. *Applied Thermal Engineering*. Retrieved from http://dx.doi.org/10.1016/j.applthermaleng.2016.02.012 doi: 10.1016/j.applthermaleng.2016.02.012

Haydt, G., Leal, V., Pina, A., & Silva, C. A. (2011). The relevance of the energy resource dynamics in the mid/long-term energy planning models. *Renewable Energy*, *36*(11), 3068 - 3074. Retrieved from http://www.sciencedirect.com/science/article/pii/S096014811100142X doi: http://dx.doi.org/10.1016/j.renene.2011.03.028

Hendrik Kondziella, & Thomas Bruckner. (2016). Flexibility requirements of renewable energy based electricity systems - a review of research results and methodologies. *Renewable and Sustainable Energy Reviews*, *53*, 10 - 22. Retrieved from http://www.sciencedirect.com/science/article/pii/S1364032115008643 doi: http://dx.doi.org/10.1016/j.rser.2015.07.199

HLRS-MODULE. (2019). *Module command.* Retrieved from https://kb.hlrs.de/platforms/index.php/Module_command

HLRS-PBS. (2019). *Cray xc40 using the batch system.* Retrieved from https://kb.hlrs.de/platforms/index.php/CRAY_XC40_Using_the_Batch_System

HLRS-SICOS. (2019). *Enterprises & sme.* Retrieved from https://www.hlrs.de/solutions-services/enterprises-sme

HLRS-SSH. (2019). *Secure shell ssh.* Retrieved from https://kb.hlrs.de/platforms/index.php/Secure_Shell_ssh

HLRS-Systems. (2019). *Systems.* Retrieved from https://www.hlrs.de/systems/

HLRS-VPN. (2019). *Access hlrs compute service using vpn.* Retrieved from https://kb.hlrs.de/platforms/index.php/VPN

HLRS-WORKSPACE. (2019). *Workspace mechanism.* Retrieved from https://kb.hlrs.de/platforms/index.php/Workspace_mechanism

Hofmann, F., Hörsch, J., & Gotzens, F. (2018). *Fresna/powerplantmatching: python3 adjustments 2018.*

Hörsch, J., & Brown, T. (2017, June). The role of spatial scale in joint optimisations of generation and transmission for European highly renewable scenarios. In *2017 14th international conference on the european energy market (eem)* (p. 1-7). Retrieved from http://ieeexplore.ieee.org/document/

7982024/ doi: 10.1109/EEM.2017.7982024
HyungSeon Oh. (2011, Aug). Optimal Planning to Include Storage Devices in Power Systems. *Power Systems, IEEE Transactions on*, *26*(3), 1118 - 1128. doi: 10.1109/TPWRS.2010.2091515
IBM. (2013). *Ibm deterministic time, release notes for ibm cplex optimizer for z/os v12.5.*
ICL. (2018). *Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers.* www.netlib.org. Retrieved from http://www.netlib.org/benchmark/hpl/
Intel. (2016, Juni). Intel 64 and ia-32 architectures optimization reference manual. In (p. 2-29). Retrieved from http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html
Intel. (2019a, April). Intel 64 and ia-32 architectures optimization reference manual. In (p. 2-9 and 2-17). Retrieved from https://software.intel.com/sites/default/files/managed/9e/bc/64-ia-32-architectures-optimization-manual.pdf
Intel. (2019b, April). Intel 64 and ia-32 architectures optimization reference manual. In (p. 2-31). Retrieved from https://software.intel.com/sites/default/files/managed/9e/bc/64-ia-32-architectures-optimization-manual.pdf
Intel. (2019c, May). *Intel xeon processor scalable family, specification update; reference number: 336065-008us.* Retrieved from https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-scalable-spec-update.pdf
James H. Merrick. (2016). On representation of temporal variability in electricity capacity planningÂ models. *Energy Economics*, *59*, 261 - 274. Retrieved from http://www.sciencedirect.com/science/article/pii/S0140988316302018 doi: http://dx.doi.org/10.1016/j.eneco.2016.08.001
Jan Abrell, Friedrich Kunz, & Hannes Weigt. (2008). *Start Me Up: Modeling of Power Plant Start-Up Conditions and their Impact on Prices, working paper* (Tech. Rep.). {Universität Dresden}. Retrieved from https://mpra.ub.uni-muenchen.de/65661/1/wp_em_28_Abrell_Kunz_Weigt_Startup.pdf
Jianhui Wang, Shahidehpour, M., & Zuyi Li. (2008, Aug). Security-Constrained Unit Commitment With Volatile Wind Power Generation. *IEEE Transactions on Power Systems*, *23*(3), 1319-1327. Retrieved from http://dx.doi.org/10.1109/TPWRS.2008.926719 doi: 10.1109/tpwrs.2008.926719
Jorge MartÃnez Crespo, Julio Usaola, & JosÃ© L. FernÃ¡ndez. (2007). Optimal security-constrained power scheduling by Benders decomposition. *Electric Power Systems Research*, *77*(7), 739 - 753. Retrieved from http://www.sciencedirect.com/science/article/pii/S0378779606001532 doi: http://dx.doi.org/10.1016/j.epsr.2006.06.009
Jülich Supercomputing Centre. (2019). JUWELS: Modular Tier-0/1 Supercomputer at the Jülich Supercomputing Centre. *Journal of large-scale research facilities*, *5*(A135). Retrieved from http://dx.doi.org/10.17815/jlsrf-5-171 doi: 10.17815/jlsrf-5-171
Kagiannas, A. G., Askounis, D. T., & Psarras, J. (2004). Power generation planning: a survey from monopoly to competition. *International Journal of Electrical Power and Energy Systems*, *26*(6), 413 - 421. Retrieved from http://www.sciencedirect.com/science/article/pii/S0142061503001315 doi: https://doi.org/10.1016/j.ijepes.2003.11.003
Katrin Schaber, Florian Steinke, & Thomas Hamacher. (2012). Transmission grid extensions for the integration of variable renewable energies in Europe: Who benefits where? *Energy Policy*, *43*(0), 123 - 135. Retrieved from http://www.sciencedirect.com/science/article/pii/S0301421511010469 doi: http://dx.doi.org/10.1016/j.enpol.2011.12.040
Khodaei, A., Shahidehpour, M., & Kamalinia, S. (2010, Aug). Transmission Switching in Expansion Planning. *IEEE Transactions on Power Systems*, *25*(3), 1722-1733. Retrieved from http://dx.doi.org/10.1109/TPWRS.2009.2039946 doi: 10.1109/tpwrs.2009.2039946
Knüpfer, A., Rössel, C., Mey, D. a., Biersdorff, S., Diethelm, K., Eschweiler, D., ... Wolf, F. (2012). Score-p: A joint performance measurement run-time infrastructure for periscope,scalasca, tau, and vampir. In H. Brunst, M. S. Müller, W. E. Nagel, & M. M. Resch (Eds.), *Tools for high performance computing 2011* (pp. 79–91). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-31476-6_7
Kris Poncelet, Erik Delarue, Daan Six, Jan Duerinck, & William D'haeseleer. (2016). Impact of the level of temporal and operational detail in energy-system planning models. *Applied Energy*, *162*, 631 - 643. Retrieved from http://www.sciencedirect.com/science/article/pii/S0306261915013276 doi:

http://dx.doi.org/10.1016/j.apenergy.2015.10.100
Langrene, N., van Ackooij, W., & Breant, F. (2011, Aug). Dynamic Constraints for Aggregated Units: Formulation and Application. *Power Systems, IEEE Transactions on*, *26*(3), 1349-1356. doi: 10.1109/TPWRS.2010.2089539
Leuthold, F. U., Weigt, H., & von Hirschhausen, C. (2012, Mar 01). A large-scale spatial optimization model of the european electricity market. *Networks and Spatial Economics*, *12*(1), 75–107. Retrieved from https://doi.org/10.1007/s11067-010-9148-1 doi: 10.1007/s11067-010-9148-1
Linderoth, J., & Wright, S. J. (2003). Decomposition algorithms for stochastic programming on a computational grid. *Comp. Opt. and Appl.*, *24*(2-3), 207–250. Retrieved from https://doi.org/10.1023/A:1021858008222 doi: 10.1023/A:1021858008222
Loulou, R., & Labriet, M. (2008, 02). Etsap-tiam: the times integrated assessment model part i: Model structure. *Computational Management Science*, *5*, 7-40. doi: 10.1007/s10287-007-0046-z
Lustre. (2019). *Lustre filesystem.* Retrieved from http://lustre.org/
Marquant, J., Evins, R., & Carmeliet, J. (2015, 12). Reducing computation time with a rolling horizon approach applied to a milp formulation of multiple urban energy hub system. *Procedia Computer Science*, *51*, 2137-2146. doi: 10.1016/j.procs.2015.05.486
Matousek, J., & Gärtner, B. (2007). *Understanding and using linear programming.* Springer Berlin Heidelberg. Retrieved from https://books.google.de/books?id=6MO_RS4z0w8C
McCarl, B. (2017). *Bruce mccarl's gams newsletter number 41.*
McCarl, B. A. (2000). *Speeding up gams execution time.* GAMS. Retrieved from https://www.gams.com/mccarl/speed.pdf
Mitra, S., Sun, L., & Grossmann, I. E. (2013). Optimal scheduling of industrial combined heat and power plants under time-sensitive electricity prices. *Energy*, *54*(Supplement C), 194 - 211. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360544213001448 doi: https://doi.org/10.1016/j.energy.2013.02.030
MogonWiki. (2017). *Vectorization.* Retrieved from https://mogonwiki.zdv.uni-mainz.de/dokuwiki/vectorization
MPI-Forum. (2015, June). *Mpi: A message-passing interface standard, version 3.1.* Retrieved from https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf
Munoz, F. D., Sauma, E. E., & Hobbs, B. F. (2013). Approximations in power transmission planning: implications for the cost and performance of renewable portfolio standards. *Journal of Regulatory Economics*, *43*(3), 305 - 338. Retrieved from http://dx.doi.org/10.1007/s11149-013-9209-8 doi: 10.1007/s11149-013-9209-8
Nima Amjady, & Mohammad Reza Ansari. (2013). Hydrothermal unit commitment with {AC} constraints by a new solution method based on benders decomposition. *Energy Conversion and Management*, *65*, 57 - 65. Retrieved from http://www.sciencedirect.com/science/article/pii/S0196890412003081 (Global Conference on Renewable energy and Energy Efficiency for Desert Regions 2011) doi: http://dx.doi.org/10.1016/j.enconman.2012.07.022
Nolden, C., Schönfelder, M., Eßer Frey, A., Bertsch, V., & Fichtner, W. (2013). Network constraints in techno-economic energy system models: towards more accurate modeling of power flows in long-term energy system models. *Energy Systems*, 1 - 21. Retrieved from http://link.springer.com/article/10.1007/s12667-013-0078-0
O'Dwyer, C., & Flynn, D. (2015). Using energy storage to manage high net load variability at sub-hourly time-scales. *IEEE Transactions on Power Systems*, *30*(4), 2139-2148.
Open power system data data package time series [Version 2017-07-09. (Primary data from various sources, for a complete list see URL)]. (2017). GAMS. Retrieved from https://data.open-power-system-data.org/time_series/2017-07-09
Oree, V., Hassen, S. Z. S., & Fleming, P. J. (2017). Generation expansion planning optimisation with renewable energy integration: A review. *Renewable and Sustainable Energy Reviews*, *69*(Supplement C), 790 - 803. Retrieved from http://www.sciencedirect.com/science/article/pii/S1364032116308711 doi: https://doi.org/10.1016/j.rser.2016.11.120
Paltsev, S. (2016). *Energy scenarios: The value and limits of scenario analysis* (EcoMod2016 No. 9371).

EcoMod. Retrieved from https://EconPapers.repec.org/RePEc:ekd:009007:9371
Pandzzic, H., Dvorkin, Y., Yishen Wang, Ting Qiu, & Kirschen, D. (2014, July). Effect of time resolution on unit commitment decisions in systems with high wind penetration. In *Pes general meeting — conference exposition, 2014 ieee* (p. 1-5). doi: 10.1109/PESGM.2014.6939548
Papavasiliou, A., Oren, S. S., & Rountree, B. (2015). Applying high performance computing to transmission-constrained stochastic unit commitment for renewable energy integration. *IEEE Transactions on Power Systems*, *30*(3), 1109–1120.
Paul Nahmmacher, Eva Schmid, Lion Hirth, & Brigitte Knopf. (2016). Carpe diem: A novel approach to select representative days for long-term power system modeling. *Energy*, *112*, 430 - 442. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360544216308556 doi: http://dx.doi.org/10.1016/j.energy.2016.06.081
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011, November). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, *12*, 2825–2830. Retrieved from http://dl.acm.org/citation.cfm?id=1953048.2078195
Petra, C. G., Schenk, O., & Anitescu, M. (2014). Real-time stochastic optimization of complex energy systems on high-performance computers. *Computing in Science and Engineering*, *16*(5), 32-42. Retrieved from doi.ieeecomputersociety.org/10.1109/MCSE.2014.53 doi: 10.1109/MCSE.2014.53
Pfenninger, S., Hawkes, A., & Keirstead, J. (2014). Energy systems modeling for twenty-first century energy challenges. *Renewable and Sustainable Energy Reviews*, *33*, 74 - 86. Retrieved from http://www.sciencedirect.com/science/article/pii/S1364032114000872 doi: http://dx.doi.org/10.1016/j.rser.2014.02.003
putty.org. (2019). *Putty.* Retrieved from https://putty.org
Quintero, J., Zhang, H., Chakhchoukh, Y., Vittal, V., & Heydt, G. T. (2014, July). Next generation transmission expansion planning framework: Models, tools, and educational opportunities. *IEEE Transactions on Power Systems*, *29*(4), 1911-1918. doi: 10.1109/TPWRS.2014.2317590
Raichur, V., Callaway, D. S., & Skerlos, S. J. (2016). Estimating Emissions from Electricity Generation Using Electricity Dispatch Models: The Importance of System Operating Constraints. *Journal of Industrial Ecology*, *20*(1), 42 - 53. Retrieved from http://dx.doi.org/10.1111/jiec.12276 doi: 10.1111/jiec.12276
Ramos, A. (2018). *Good optimization modeling practices with gams.*
Rehfeldt, D., Hobbie, H., Schönheit, D., Gleixner, A., Koch, T., & Möst, D. (2019). *A massively parallel interior-point solver for linear energy system models with block structure* (Tech. Rep. No. 19-41). Takustr. 7, 14195 Berlin: ZIB.
Rippel, K. M., Preuß, A., Meinecke, M., & König, R. (2017). *Netzentwicklungsplan 2030 Zahlen Daten Fakten* (Data Documentation). German Transmission System Operators.
Roh, J. H., Shahidehpour, M., & Fu, Y. (2007, Nov). Market-Based Coordination of Transmission and Generation Capacity Planning. *IEEE Transactions on Power Systems*, *22*(4), 1406-1419. Retrieved from http://dx.doi.org/10.1109/TPWRS.2007.907894 doi: 10.1109/tpwrs.2007.907894
Romero, R., & Monticelli, A. (1994, Feb). A hierarchical decomposition approach for transmission network expansion planning. *IEEE Transactions on Power Systems*, *9*(1), 373-380. doi: 10.1109/59.317588
Saad, Y. (2003). *Iterative methods for sparse linear systems* (2nd ed.). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.
Saviankou, P., Knobloch, M., Visser, A., & Mohr, B. (2015, June). Cube v4: From performance report explorer to performance analysis tool. *Procedia Computer Science*, *51*, 1343–1352. doi: 10.1016/j.procs.2015.05.320
Scholz, A., Sandau, F., & Pape, C. (2016). A European Investment and Dispatch Model for Determining Cost Minimal Power Systems with High Shares of Renewable Energy. *Operations Research Proceedings 2014*, 515-521. Retrieved from http://dx.doi.org/10.1007/978-3-319-28697-6_72 doi: 10.1007/978-3-319-28697-6_72
Scholz, Y., Gils, H. C., & Pietzcker, R. C. (2017). Application of a high-detail energy system model to derive power sector characteristics at high wind and solar shares. *Energy Economics*, *64*, 568 - 582. Retrieved from http://www.sciencedirect.com/science/article/pii/S0140988316301682 doi:

https://doi.org/10.1016/j.eneco.2016.06.021
Shayesteh, E., Hamon, C., Amelin, M., & Söder, L. (2014). REI method for multi-area modeling of power systems. *International Journal of Electrical Power & Energy Systems*, *60*(Supplement C), 283 - 292. Retrieved from http://www.sciencedirect.com/science/article/pii/S0142061514001045 doi: https://doi.org/10.1016/j.ijepes.2014.03.002
Shi, D., & Tylavsky, D. J. (2015). A Novel Bus-Aggregation-Based Structure-Preserving Power System Equivalent. *Power Systems, IEEE Transactions on Power Systems*, *30*(4), 1977 - 1986.
Shilov, A. (2018, 10). *Amd previews epyc rome processor: Up to 64 zen 2 cores.* AnandTech. Retrieved from https://www.anandtech.com/show/13561/amd-previews-epyc-rome-processor-up-to-64-zen-2-cores
Silvente, J., Kopanos, G. M., Pistikopoulos, E. N., & Espuña, A. (2015). A rolling horizon optimization framework for the simultaneous energy supply and demand planning in microgrids. *Applied Energy*, *155*, 485 - 501. Retrieved from http://www.sciencedirect.com/science/article/pii/S0306261915007230 doi: https://doi.org/10.1016/j.apenergy.2015.05.090
Slurm workload manager. (2019). Retrieved from https://slurm.schedmd.com
Spiecker, S., Vogel, P., & Weber, C. (2013). Evaluating interconnector investments in the north european electricity system considering fluctuating wind power penetration. *Energy Economics*, *37*(Supplement C), 114 - 127. Retrieved from http://www.sciencedirect.com/science/article/pii/S0140988313000170 doi: https://doi.org/10.1016/j.eneco.2013.01.012
Stefan Pfenninger. (2017). Dealing with multiple decades of hourly wind and PV time series in energy models: A comparison of methods to reduce time resolution and the planning implications of inter-annual variability. *Applied Energy*, *197*, 1 - 13. Retrieved from http://www.sciencedirect.com/science/article/pii/S0306261917302775 doi: http://dx.doi.org/10.1016/j.apenergy.2017.03.051
Sterling, T., Anderson, M., & Brodowicz, M. (2018). *High performance computing : modern systems and practices.* Cambridge, MA: Elsevier Morgan Kaufmann Publishers. (englisch)
Supercomputing-Akademie. (2019). *Mit der supercomputing-akademie zum hpc-anwender!* Retrieved from https://www.supercomputing-akademie.de
Sylvie Ludig, Markus Haller, Eva Schmid, & Nico Bauer. (2011). Fluctuating renewables in a long-term climate change mitigation strategy. *Energy*, *36*(11), 6674 - 6685. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360544211005512 doi: http://dx.doi.org/10.1016/j.energy.2011.08.021
Teruel, A. G. (2015). *Perspestective of the energy transition: Technology development and investments under uncertainty* (Unpublished master's thesis). Technical University of Munich.
Thomas Anderski, Yvonne Surmann, Simone Stemmer, Nathalie Grisey, Eric Momo, Anne-Claire Leger, ... Peter Van Roy (2014). *Modular Development Plan of the Pan-European Transmission System 2050 - European cluster model of the Pan-European transmission grid* (Tech. Rep.). Retrieved from http://www.e-highway2050.eu/fileadmin/documents/Results/D2_2_European_cluster_model_of_the_Pan-European_transmission_grid_20072015.pdf
TOP500.org. (2018). *Top500 description.* Retrieved from https://www.top500.org/project/top500_description/
Tuohy, A., Denny, E., & O'Malley, M. (2007, Jul). Rolling Unit Commitment for Systems with Significant Installed Wind Capacity. *2007 IEEE Lausanne Power Tech*, 1380 - 1385. Retrieved from http://dx.doi.org/10.1109/PCT.2007.4538517 doi: 10.1109/pct.2007.4538517
Turing, A. M. (n.d.). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, *s2-42*(1), 230-265. Retrieved from https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s2-42.1.230 doi: 10.1112/plms/s2-42.1.230
Vanderbei, R. J. (2008). The homogeneous self-dual method. In *Linear programming: Foundations and extensions* (pp. 361–381). Boston, MA: Springer US. Retrieved from https://doi.org/10.1007/978-0-387-74388-2_22 doi: 10.1007/978-0-387-74388-2_22
Ventosa, M., Baillo, A., Ramos, A., & Rivier, M. (2005). Electricity market modeling trends. *Energy Policy*, *33*(7), 897-913. Retrieved from https://EconPapers.repec.org/RePEc:eee:enepol:v:33:y:2005:i:7:p:897-913

Virmani, S., Adrian, E. C., Imhof, K., & Mukherjee, S. (1989, Nov). Implementation of a Lagrangian relaxation based unit commitment problem. *IEEE Transactions on Power Systems*, *4*(4), 1373-1380. doi: 10.1109/59.41687

Wang, Q., McCalley, J. D., Zheng, T., & Litvinov, E. (2016, Feb). Solving corrective risk-based security-constrained optimal power flow with Lagrangian relaxation and Benders decomposition. *International Journal of Electrical Power & Energy Systems*, *75*, 255-264. Retrieved from http://dx.doi.org/10.1016/j.ijepes.2015.09.001 doi: 10.1016/j.ijepes.2015.09.001

Wang, S., Shahidehpour, S., Kirschen, D., Mokhtari, S., & Irisarri, G. (1995). Short-term generation scheduling with transmission and environmental constraints using an augmented Lagrangian relaxation. *IEEE Transactions on Power Systems*, *10*(3), 1294-1301. Retrieved from http://dx.doi.org/10.1109/59.466524 doi: 10.1109/59.466524

Weigt, H., Jeske, T., Leuthold, F., & von Hirschhausen, C. (2010). "take the long way down": Integration of large-scale north sea wind using hvdc transmission. *Energy Policy*, *38*(7), 3164-3173. Retrieved from https://EconPapers.repec.org/RePEc:eee:enepol:v:38:y:2010:i:7:p:3164-3173

Wiegmans, B. (2016). *Gridkit extract of entso-e interactive map 2016.*

Wikichip. (2018). *Skylake (server) - microarchitectures - intel.* Retrieved from https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)

Wogrin, S., Dueñas, P., Delgadillo, A., & Reneses, J. (2014, Sep.). A new approach to model load levels in electric power systems with high renewable penetration. *IEEE Transactions on Power Systems*, *29*(5), 2210-2218. doi: 10.1109/TPWRS.2014.2300697

Wright, S. J. (1997). *Primal-dual interior-points methods.* Philadelphia, Pa, USA: SIAM.

Wu, F., Zheng, F., & Wen, F. (2006). Transmission investment and expansion planning in a restructured electricity market. *Energy*, *31*(6), 954 - 966. Retrieved from http://www.sciencedirect.com/science/article/pii/S0360544205000538 (Electricity Market Reform and Deregulation) doi: https://doi.org/10.1016/j.energy.2005.03.001

Zerrahn, A., & Schill, W.-P. (2017). Long-run power storage requirements for high shares of renewables: review and a new model. *Renewable and Sustainable Energy Reviews*, *79*, 1518 - 1534. Retrieved from http://www.sciencedirect.com/science/article/pii/S1364032116308619 doi: https://doi.org/10.1016/j.rser.2016.11.098

Zhao, F., Litvinov, E., & Zheng, T. (2014, Jan). A Marginal Equivalent Decomposition Method and Its Application to Multi-Area Optimal Power Flow Problems. *IEEE Transactions on Power Systems*, *29*(1), 53-61. Retrieved from http://dx.doi.org/10.1109/TPWRS.2013.2281775 doi: 10.1109/tpwrs.2013.2281775

Zhu, J. (2015). *Optimization of power system operation* (Vol. 47). John Wiley & Sons. Retrieved from http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118854152.html

Part V

Appendices

Appendix A

The SIMPLE Model - A Simplified ESM

SIMPLE is a simplified energy system model that has multiple variants. This chapter introduces a basic version of SIMPLE as well as many variants plus different decomposition approaches, heuristics and stochastic extensions that were built on top of SIMPLE.

A.1 Motivation

State-of-the-art Energy System Models are often very complex. Many have been developed over years or even decades by multiple people. Novel solution approaches usually involve complex algorithms from the field of Mathematical Programming. Apparently, implementing complex algorithms for complex models is a very challenging and time consuming task. That motivates the need for a simplified ESM that maintains relevant parts of the model structure that can be found in many ESMs but is at the same time compact and comprehensive. Apparently such a simplified model lacks many details that are considered in "'real"' ESMs. However, it allows for rapid prototyping of different solution approaches that would be very time consuming to implement for "'real"' ESMs. SIMPLE was developed to allow convenient preliminary evaluation of complex methods to identify promising solution approaches.

A.2 Automated Input Data Generation

A core concept of SIMPLE is the convenient and automatic generation of input data. The SIMPLE models come with a data generator `simple_data_gen.gms` that can be parametrized to generate data instances of different size. The data generator is called automatically from all SIMPLE models and takes only one argument, the number of regions. The basic principle of the data generator is to create all the data that goes into the model like for example demand per region and hour, available power plants and their capacity, or the amount of available renewable energy per region and hour. All the data is computed by randomizing standard basic time series that provide the corresponding data for a "'standard"' region. In addition, a simple algorithm will place the regions on a 1000x1000 km grid and compute a network of transmission links that connects different regions under consideration of distances between the regions. Note that even though there is a lot of randomization involved in the automated data generation, the process is deterministic.

A.2.1 Input Data vs. Model Data

The data generator always produces data instances that cover an entire year in hourly resolution, ergo 8760 hours. What comes out of the data generator is denoted *input data*.
In Addition, the SIMPLE models allow the user to work with further customized *model data*. This includes

the opportunity to change the time resolution which means that hourly time steps from the input data can be aggregated or disaggregated for the actual model data. Furthermore, the time horizon to be considered can be controlled via parameters `--FROM` and `--TO`.
The following example illustrates how to solve the basic SIMPLE model with 12 regions for the first half of a year with a time resolution of 6 hours.

```
gams simpleBase.gms --NBREGIONS=12 --FROM=0 --TO=0.5 --RESOLUTION=6
```

A.3 The SIMPLE Models - How to run?

The following sections introduce different versions of the SIMPLE model:

- Basic SIMPLE Model: `simpleBase.gms`
- Extensions to the SIMPLE Model: `simple.gms`
- SIMPLE Model for PIPS-IPM: `simple4pips.gms`

In Addition, to run the SIMPLE models the data generator `simple_data_gen.gms` is required. Assuming that GAMS has been properly installed, the SIMPLE models can be run from the command line as follows:

```
gams <modelname>.gms [list of parameters]
```

A.3.1 Generic Parameters

The different SIMPLE Models share several parameters which are summarized in this section. Other model or even method specific parameters are explained in the following sections about the different model versions. The SIMPLE parameters can be set as so called Double Dash Parameters:

```
gams <modelname>.gms --parameter1=value --parameter2=value ...
```

Generic parameters are:

Parameter	Type	Range	Default	Description
FROM	float	$[0, 1]$	0	This parameter determines the start of the time horizon that should be considered in the model data
TO	float	$[0, 1]$	1	This parameter determines the end of the time horizon that should be considered in the model data
RESOLUTION	float	$[0, 8760]$	1	This parameter determines the resolution in hours for the model data
NBREGIONS	integer	$\{1, 2, ..., \text{inf}\}$	4	This parameter determines the number of regions

Table A.1: Generic SIMPLE Parameters

A.4 Basic SIMPLE Model - `simpleBase.gms`

A.4.1 Symbols

A.4.1.1 Sets

Name	Domains	Description
rr, rr1, rr2	*	regions
p	*	plants
s	*	storages
e	*	emissions

Name	Domains	Description
type	*	plant type
rp	rr, p	region to plant mapping
ptype	rr, p, type	plant type mapping
rs	rr, s	region to storage mapping
net	rr1, rr2	transmission links
tt	*	time steps
r	rr	active region
t	tt	subset of active time steps

A.4.1.2 Parameters

Name	Domains	Description
plant_emission	rr, p, e	plant emission [tons/GWh]
cost_power_generation	rr, p	electricity production cost [MEUR/GWh]
cost_emission	e	emission costs [MEUR/ton]
storage_cap	rr, s	storage capacity [GWh]
storage_efficiency	rr, s	efficiency factor of storage
storage_efficiency_in	rr, s	efficiency factor of storage inflow
storage_efficiency_out	rr, s	efficiency factor of storage outflow
cost_plant_add	rr, p	cost for additional plant capacity [MEUR/GW]
cost_storage_add	rr, s	cost for additional storage capacity [MEUR/GWh]
cost_link_add	rr1, rr2	cost for additional link capacity [MEUR/GWh]
plant_cap	tt, rr, p	plant capacity in time step t [GWh]
total_plant_cap	rr, p	plant capacity over total time span [GWh]
total_emission_cap	e	emission cap for total time span [tons]
cost_unserved_demand	tt	price for unserved demand [MEUR/GWh]
link_cap	tt, rr1, rr2	transmission link capacity per time step[GWh]
link_efficiency	tt, rr1, rr2	transmission link efficiency factor
demand	tt, rr	demand for region per time step [GWh]
avail	tt, rr, p	
resolution		model resolution in hours. 1 means that a model time step corresponds to one hour, 3 means it correspnds to 3 hours etc

A.4.1.3 Variables

Name	Domains	Description
POWER	tt, rr, p	power production [GWh]
FLOW	tt, rr1, rr2	power flow [GWh]
LINK_ADD_CAP	rr1, rr2	arc capacity expansion [GWh]
SLACK	tt, rr	uncovered demand [GWh]
STORAGE_LEVEL	tt, rr, s	storage level [GWh]
STORAGE_INFLOW	tt, rr, s	power entering storage [GWh]
STORAGE_OUTFLOW	tt, rr, s	power taken from storage [GWh]
PLANT_ADD_CAP	rr, p	plant capacity expansion [GW]
STORAGE_ADD_CAP	rr, s	storage capacity expansion [GWh]
EMISSION_SPLIT	rr, e	emission allowance split [fraction]
EMISSION_COST	rr, e	emission cost [MEUR]
ROBJ	rr	total region cost [MEUR]
OBJ		total [MEUR]

A.4.1.4 Equations

Name	Domains	Description
eq_robj	rr	total cost in region
eq_power_balance	tt, rr	power balance
eq_plant_capacity	tt, rr, p	respect plant capacity
eq_total_plant_capacity	rr, p	respect total plant capacity
eq_storage_balance	tt, rr, s	storage balance
eq_storage_capacity	tt, rr, s	respect storage capacity
eq_emission_region	rr, e	calculate regional emissions
eq_emission_cost	rr, e	calculate regional emission costs
eq_emission_cap	e	respect emission cap
eq_link_capacity	tt, rr1, rr2	respect link capacity
eq_obj		total cost

A.4.2 Equation Definitions

eq_obj - objective function

$$\text{OBJ} = \sum_{r} \text{ROBJ}_{r} + \sum_{rr1,rr2,net_{rr1,rr2}} (\text{LINK\_ADD\_CAP}_{rr1,rr2} \cdot \text{cost\_link\_add}_{rr1,rr2})$$

The objective function sums over the regional objectives plus inter-regional costs for network expansion.

---

eq_robj$_r$ - total cost in region

$$\begin{aligned}\text{ROBJ}_{r} = & \sum_{t,p,type,ptype_{r,p,type},rp_{r,p}} (\text{POWER}_{t,r,p} \cdot \text{cost\_power\_generation}_{r,p}) \\ & + \sum_{t} (\text{SLACK}_{t,r} \cdot \text{cost\_unserved\_demand}_{t}) \\ & + \sum_{p,rp_{r,p}} (\text{PLANT\_ADD\_CAP}_{r,p} \cdot \text{cost\_plant\_add}_{r,p}) \\ & + \sum_{s,rs_{r,s}} (\text{STORAGE\_ADD\_CAP}_{r,s} \cdot \text{cost\_storage\_add}_{r,s}) \\ & + \sum_{e} \text{EMISSION\_COST}_{r,e} \qquad \forall r\end{aligned}$$

Total costs per region are computed as the sum of power generation costs, penalty for unserved demand, plant expansion costs, storage expansion costs and emission costs.

---

eq_power_balance$_{t,r}$ - power balance constraint

$$\begin{aligned} & \sum_{p,rp_{r,p}} \text{POWER}_{t,r,p} \\ + \sum_{rr2,net_{rr2,r}} (\text{FLOW}_{t,rr2,r} \cdot \text{link\_efficiency}_{t,rr2,r}) & - \sum_{rr2,net_{r,rr2}} \text{FLOW}_{t,r,rr2} \\ + \sum_{s,rs_{r,s}} (\text{STORAGE\_OUTFLOW}_{t,r,s} & - \text{STORAGE\_INFLOW}_{t,r,s}) \\ & + \text{SLACK}_{t,r} \geq \text{demand}_{t,r} \qquad \forall t, r \end{aligned}$$

The power balance constraint ensures for every time step and every region that the demand is satisfied by the sum of:

- generated power
- inbound power flow minus outbound power flow
- withdrawal from storage minus filling up of storage
- unsatisfied demand

---

eq_plant_capacity$_{t,rp_{r,p}}$

$$\text{POWER}_{t,r,p} \leq (\text{plant\_cap}_{t,r,p} + \text{PLANT\_ADD\_CAP}_{r,p} \cdot \text{resolution}) \cdot \text{avail}_{t,r,p} \qquad \forall t, rp_{r,p}$$

In every time step for every plant the power generation has to be less than the initially installed capacity [GWh per time step] plus the additionally installed capacity [GW] multiplied with the resolution [hours per time step].

---

eq_total_plant_capacity$_{rp_{r,p}}$

$$\sum_t \text{POWER}_{t,r,p} \leq \text{total\_plant\_cap}_{r,p} \qquad \forall rp_{r,p}$$

Over the entire time horizon, a power plant's maximum production is limited to a fixed value. This constraints mimics the scarceness of fuel types required to operate certain plant types.

---

eq_storage_balance$_{t_{tt},rs_{r,s}}$

$$\begin{aligned}\text{STORAGE\_LEVEL}_{tt,r,s} = {} & \text{STORAGE\_LEVEL}_{tt--1,r,s} \cdot \text{storage\_efficiency}_{r,s} \\ & + \text{STORAGE\_INFLOW}_{tt,r,s} \cdot \text{storage\_efficiency\_in}_{r,s} \\ & - \text{STORAGE\_OUTFLOW}_{tt,r,s} \cdot \frac{1}{\text{storage\_efficiency\_out}_{r,s}} \qquad \forall t_{tt}, rs_{r,s}\end{aligned}$$

The storage balance constraint ensures for every time step and every storage that the current storage level is equal to:

- the storage level from the previous time step minus some factor that mimics for example evaporation in water reservoirs of pump storages
- plus storage inflow in the current period scaled with a corresponding efficiency factor
- minus withdrawal from the storage in the current period scaled with a corresponding efficiency factor

---

eq_storage_capacity$_{t,rs_{r,s}}$

$$\text{STORAGE\_LEVEL}_{t,r,s} \leq \text{storage\_cap}_{r,s} + \text{STORAGE\_ADD\_CAP}_{r,s} \qquad \forall t, rs_{r,s}$$

In every time step, the storage level cannot exceed the initial storage capacity plus the additionally installed storage capacity.

---

eq_emission_region$_{r,e}$

$$\sum_{p,t,rp_{r,p}} (\text{POWER}_{t,r,p} \cdot \text{plant\_emission}_{r,p,e}) \leq \text{total\_emission\_cap}_e \cdot \text{EMISSION\_SPLIT}_{r,e} \qquad \forall r, e$$

For every emission type and region, the emissions produced by the plants in that region have to be less than or equal to an emission cap.

---

eq_emission_cost$_{r,e}$

$$\sum_{p,t,rp_{r,p}} (\text{POWER}_{t,r,p} \cdot \text{plant\_emission}_{r,p,e}) \cdot \text{cost\_emission}_e = \text{EMISSION\_COST}_{r,e} \qquad \forall r, e$$

For every emission type and region, the emission costs are computed.

---

eq_emission_cap$_e$

$$\sum_{rr} \text{EMISSION\_SPLIT}_{rr,e} \leq 1 \qquad \forall e$$

For every emission type, the emission has to be split between the different regions.

---

eq_link_capacity$_{t,net}$

$$\text{FLOW}_{t,net} \leq \text{link\_cap}_{t,net} + \text{LINK\_ADD\_CAP}_{net} \cdot \text{resolution} \qquad \forall t, net$$

In every time step, the amount of power flow transferred on a particular link of the power grid, cannot exceed the initial capacity of that link [GWh per time step] plus the additionally installed capacity [GW] multiplied with the resolution [hours per time step].

$$\begin{aligned}
\text{POWER}_{tt,rr,p} &\geq 0 && \forall tt, rr, p \\
\text{SLACK}_{tt,rr} &\geq 0 && \forall tt, rr \\
\text{PLANT_ADD_CAP}_{rr,p} &\geq 0 && \forall rr, p \\
\text{STORAGE_ADD_CAP}_{rr,s} &\geq 0 && \forall rr, s \\
\text{FLOW}_{tt,rr1,rr2} &\geq 0 && \forall tt, rr1, rr2 \\
\text{STORAGE_OUTFLOW}_{tt,rr,s} &\geq 0 && \forall tt, rr, s \\
\text{STORAGE_INFLOW}_{tt,rr,s} &\geq 0 && \forall tt, rr, s \\
\text{STORAGE_LEVEL}_{tt,rr,s} &\geq 0 && \forall tt, rr, s \\
\text{EMISSION_SPLIT}_{rr,e} &\geq 0 && \forall rr, e \\
\text{LINK_ADD_CAP}_{rr1,rr2} &\geq 0 && \forall rr1, rr2
\end{aligned}$$

A.4.3 Model Overview

$$\text{OBJ} = \sum_r \text{ROBJ}_r$$

$$+ \sum_{rr1,rr2,net_{rr1,rr2}} (\text{LINK\_ADD\_CAP}_{rr1,rr2} \cdot \text{cost\_link\_add}_{rr1,rr2})$$

$$\begin{aligned}
\text{ROBJ}_r = & \sum_{t,p,type,ptype_{r,p,type},rp_{r,p}} (\text{POWER}_{t,r,p} \cdot \text{cost_power_generation}_{r,p}) \\
& + \sum_t (\text{SLACK}_{t,r} \cdot \text{cost_unserved_demand}_t) \\
& + \sum_{p,rp_{r,p}} (\text{PLANT_ADD_CAP}_{r,p} \cdot \text{cost_plant_add}_{r,p}) \\
& + \sum_{s,rs_{r,s}} (\text{STORAGE_ADD_CAP}_{r,s} \cdot \text{cost_storage_add}_{r,s}) \\
& + \sum_e \text{EMISSION_COST}_{r,e} && \forall r
\end{aligned}$$

$$\begin{aligned}
\text{demand}_{t,r} \leq & \sum_{p,rp_{r,p}} \text{POWER}_{t,r,p} \\
& + \sum_{rr2,net_{rr2,r}} (\text{FLOW}_{t,rr2,r} \cdot \text{link_efficiency}_{t,rr2,r}) \\
& - \sum_{rr2,net_{r,rr2}} \text{FLOW}_{t,r,rr2} \\
& + \sum_{s,rs_{r,s}} (\text{STORAGE_OUTFLOW}_{t,r,s} \\
& - \text{STORAGE_INFLOW}_{t,r,s}) \\
& + \text{SLACK}_{t,r} && \forall t, r
\end{aligned}$$

$$\text{POWER}_{t,r,p} \leq (\text{plant\_cap}_{t,r,p} + \text{PLANT\_ADD\_CAP}_{r,p} \cdot \text{resolution}) \cdot \text{avail}_{t,r,p} \qquad \forall t, rp_{r,p}$$

$$\sum_t \text{POWER}_{t,r,p} \leq \text{total\_plant\_cap}_{r,p} \qquad \forall rp_{r,p}$$

$$\begin{aligned}
\text{STORAGE_LEVEL}_{tt,r,s} = & \text{STORAGE_LEVEL}_{tt--1,r,s} \cdot \text{storage_efficiency}_{r,s} \\
& + \text{STORAGE_INFLOW}_{tt,r,s} \cdot \text{storage_efficiency_in}_{r,s} \\
& - \text{STORAGE_OUTFLOW}_{tt,r,s} \cdot \frac{1}{\text{storage_efficiency_out}_{r,s}} && \forall t_{tt}, rs_{r,s}
\end{aligned}$$

$$\text{STORAGE\_LEVEL}_{t,r,s} \leq \text{storage\_cap}_{r,s} + \text{STORAGE\_ADD\_CAP}_{r,s} \qquad \forall t, rs_{r,s}$$

$$\sum_{p,t,rp_{r,p}} (\text{POWER}_{t,r,p} \cdot \text{plant\_emission}_{r,p,e}) \leq \text{total\_emission\_cap}_e \cdot \text{EMISSION\_SPLIT}_{r,e} \qquad \forall r, e$$

$$\text{EMISSION\_COST}_{r,e} = \sum_{p,t,rp_{r,p}} (\text{POWER}_{t,r,p} \cdot \text{plant\_emission}_{r,p,e}) \cdot \text{cost\_emission}_e \qquad \forall r, e$$

$$\sum_{rr} \text{EMISSION\_SPLIT}_{rr,e} \leq 1 \qquad \forall e$$

$$\begin{aligned}
\text{POWER}_{tt,rr,p} &\geq 0 && \forall tt, rr, p \\
\text{SLACK}_{tt,rr} &\geq 0 && \forall tt, rr \\
\text{PLANT_ADD_CAP}_{rr,p} &\geq 0 && \forall rr, p \\
\text{STORAGE_ADD_CAP}_{rr,s} &\geq 0 && \forall rr, s
\end{aligned}$$

$$\text{FLOW}_{tt,rr1,rr2} \geq 0 \qquad \forall tt, rr1, rr2$$
$$\text{STORAGE\_OUTFLOW}_{tt,rr,s} \geq 0 \qquad \forall tt, rr, s$$
$$\text{STORAGE\_INFLOW}_{tt,rr,s} \geq 0 \qquad \forall tt, rr, s$$
$$\text{STORAGE\_LEVEL}_{tt,rr,s} \geq 0 \qquad \forall tt, rr, s$$
$$\text{EMISSION\_SPLIT}_{rr,e} \geq 0 \qquad \forall rr, e$$
$$\text{LINK\_ADD\_CAP}_{rr1,rr2} \geq 0 \qquad \forall rr1, rr2$$

A.5 Extensions to the Basic SIMPLE Model - `simple.gms`

As mentioned in the motivation above, SIMPLE was designed as a compact and comprehensive simplification that allows for rapid prototyping of decomposition methods and extensions. The following sections provide an overview of solution approaches that have been implemented on top of the basic SIMPLE model.

In Addition to the generic parameters, there are some parameters that are dedicated to the extended SIMPLE version.

Parameter	Type	Range	Default	Description
METHOD	string		`standard_lp`	This parameter specifies the method/extension to apply. Possible values are: • `standard_lp` • `rolling_horizon` • `lagrange_relaxation` • `spExplicitDE` • `stochasticEMP` • `spBendersSeq` • `spBendersAsync` • `spBendersMPI`
SCALING	binary	$\{0, 1\}$	0	This parameter determines if a scaled version of the model with better numerical properties should be generated (1) or not (0)
NOSLACK	binary	$\{0, 1\}$	0	This parameter determines whether the $SLACK$ variables should be removed from the model (1) or not (0)

Table A.6: Parameters for SIMPLE Extensions

A.5.1 Rolling Horizon

If `simple.gms` is called with parameter `--METHOD=rolling_horizon`, the model will be partitioned into (overlapping) time horizons that are solved sequentially. Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
NBINTERVALSRH	integer	$\{1, 2, ..., n\}$	4	This parameter specifies the number of time horizons. The number of time steps n can be at most the number of model time steps
OVERLAPSIZERH	float	$[0, \text{inf}]$	0.1	This parameter determines the relative overlap of one time horizon with the subsequent time horizon.

Table A.7: Parameters for the Rolling Horizon Approach

The Rolling Horizon approach is implemented in `rolling_horizon.gms`.

A.5.2 Benders Decomposition

If `simple.gms` is called with parameter `--METHOD=benders`), the SIMPLE model will be solve via BENDERS Decomposition . Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
BENDERSMAXITER	integer	[0, inf]	5000	This parameter specifies the maximum number of Benders iterations

Table A.8: Parameters for the Benders Decomposition Approach

The Benders Decomposition approach is implemented in `benders.gms`.

A.5.3 Lagrangian Relaxation

If `simple.gms` is called with parameter `--METHOD=lagrange_relaxation`), the SIMPLE model will be solved by Lagrangian Relaxation . Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
ITERLIM	float	[0, 1e6]	10	This parameter specifies the maximum number of iterations
RESLIM	float	[0, inf]	20	This parameter specifies the time limit
RGAP	float	[0, inf]	0.1	This parameter specifies the relative termination tolereance

Table A.9: Parameters for the Lagrangian Relaxation Approach

The Lagrangian Relaxation approach is implemented in `lg.gms`.

A.5.4 Stochastic Program - Explicit Deterministic Equivalent

If `simple.gms` is called with parameter `--METHOD=spExplicitDE`, the deterministic equivalent of a stochastic problem version will be generated explicitly. The stochasticity arises from uncertainty in the power generation costs. Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
NBSCEN	integer	[1, inf]	5	This parameter specifies the number of scenarios
CPLEXBENDERS	integer	{0, 1, ..., 999}	1	This parameter specifies the CPLEX algorithm to solve the model. Possible options are 0 Do not use CPLEX at all. Instead use the conversion tool CONVERTD to create a gdx file that contains (one block of) a representation of the problem that is understood by parallel interior point solver PIPS-IPM. (see also option SCENBLOCK) 1 CPLEX will use its internal implementation of Benders Decomposition. The partition into master and subproblems is done via dot option BendersPartition 999 CPLEX will use the barrier algorith without crossover N CPLEX will look for a solver option file `cplex.opN` (or `cplex.oNN` or `cplex.NNN` if N has 2 or 3 digits). If there is no such CPLEX option file, CPLEX will solve the problem with default settings.

Parameter	Type	Range	Default	Description
SCENBLOCK	integer	$[-2, \texttt{NBSCEN}+1]$	-2	This parameter is only relevant if parameter CPLEXBENDERS is equal to 0. In that case a gdx file with a block structured representation of (one block of) the problem is created. options for this parameter are: -2 generate one large gdx file with the entire model -1 generate NBSCEN+1 block files sequentially n for $0 \leq n \leq$ NBSCEN the corresponding block file will be created

Table A.10: Parameters for the Explicit Deterministic Equivalent Approach

The Explicit Deterministic Equivalent approach is implemented in `spexplicitde.gms`.

A.5.5 Stochastic Program - Extended Mathematical Programming Framework

The Extended Mathematical Programming (EMP) framework is an extension to GAMS that facilitates the automatic reformulation of new model types as models in more established mathematical programming classes, allowing them to be solved by mature solver algorithms. If `simple.gms` is called with parameter `--METHOD=stochasticEMP`, the EMP framework will be used to create a two-stage Stochastic Problem version via some annotation file and the deterministic SIMPLE model. The stochasticity arises from uncertainty in the power generation costs. The annotation is provided in the EMP Info File and specifies a partition of variables and equations into first and second stage. Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
EMPMETHOD	string		DE	This parameter specifies the EMP Method that should be used to create/solve the two-stage Stochastic Problem. Possible options are: DE The tool DE is used to build the Deterministic Equivalent of the Stochastic Program which can then be solved with any LP Solver. LINDO The deterministic model plus the EMP Info file are passed to the solver LINDO which forms and solves the Determinstic Equivalent of the Stochastic problem internally. LINDOBENDERS The deterministic model plus the EMP Info file are passed to the solver LINDO which forms the Determinstic Equivalent of the Stochastic problem internally and solves it with LINDO's internal implementation of Benders Decomposition.

Table A.11: Parameters for the EMP Approach of Extended SIMPLE Model

The EMP approach is implemented in `stochasticEMP.gms`.

A.5.6 Stochastic Program - Benders Decomposition (Sequential)

If `simple.gms` is called with parameter `--METHOD=SpBendersSeq`, a two-stage Stochastic Problem with uncertain power generation costs will be solved with a Benders Decomposition approach. In this approach, the sub-problems are solved sequentially. Note that this approach is the basis for the more sophisticated parallel implementation of the Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
USESTARTINGPOINT	binary	$\{0,1\}$	1	If set to 1, the deterministic problem is solved to obtain a starting point
USETRUSTREG	binary	$\{0,1\}$	1	If set to 1, trust region approach will be used
DYNAMICTRUSTREG	binary	$\{0,1\}$	1	If set to 1, dynamic trust region approach will be used
NBSCEN	integer	$\{1,\infty\}$	5	This parameter specifies the number of scenarios
NBCLUSTER	integer	$\{1,\infty\}$	%NBSCEN%1	Number of cluster cuts
BENDERSITERLIM	integer	$\{1,\infty\}$	300	Benders iteration limit
CONVERGENCECRIT	float	$[0,\infty]$	10^{-5}	Convergence criterion (difference between best found solution and best bound)

Table A.12: Parameters for the sequential stochastic Benders Decomposition approach for the extended SIMPLE Model

The sequential stochastic Benders Decomposition approach that is implemented in `spBendersSeq.gms`.//

A.5.7 Stochastic Program - Benders Decomposition (Asynchronous)

If `simple.gms` is called with parameter `--METHOD=spBendersAsync`, a two-stage Stochastic Problem with uncertain power generation costs will be solved with a Benders Decomposition approach based on the asyncronous trust region method for stochastic programming by (Linderoth & Wright, 2003). Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
USESTARTINGPOINT	binary	$\{0,1\}$	1	If set to 1, the deterministic problem is solved to obtain a starting point
USESCENARIOCUTS	binary	$\{0,1\}$	1	If set to 1, scenario cuts will be used
USETRUSTREG	binary	$\{0,1\}$	1	If set to 1, trust regions will be used
DYNAMICTRUSTREG	binary	$\{0,1\}$	1	If set to 1, dynamic trust regions will be used
NBSCEN	integer	$\{1,\infty\}$	5	This parameter specifies the number of scenarios
NBCLUSTER	integer	$\{1,\infty\}$	%NBSCEN%	Number of cluster cuts
BENDERSITERLIM	integer	$\{1,\infty\}$	300	Benders iteration limit
REQUIREDSCENSHARE	float	$[0,1]$	1	Share of evaluated clusters to start subsequent iteration
CONVERGENCECRIT	float	$[0,\infty]$	10^{-5}	Convergence criterion (difference between best found solution and best bound)

Table A.13: Parameters for asynchronous stochastic Benders Decomposition approach for the extended SIMPLE model

The asynchronous stochastic Benders Decomposition approach is implemented in `spBendersAsync.gms`.

A.5.8 Stochastic Program - Benders Decomposition (parallel using MPI)

If `simple.gms` is called with parameter `--METHOD=SpBendersMPI`, a two-stage Stochastic Problem with uncertain power generation costs will be solved with a Benders Decomposition approach. The solution Process is parallelized via MPI. Note that in this approach, simple.gms has to be used like an MPI program, i.e. for n scenarios, we have to run $n+1$ GAMS jobs of simple.gms in parallel. Each of the $n+1$ knows its *rank* where the rank goes from 0 to n. The GAMS job with rank 0 solves the master problem while the GAMS jobs with rank $1, 2, ..., n$ solve the subproblems that belong to the different scenarios. That means data exchange between master ans sub-problems does require inter process communication which is implemented via MPI. To be precise, The Embedded Code facility to convert the GAMS data structures that need to be exchanged into Python data structures which are then communicated via Python package mpi4py and its broadcast and gather commands. Additional Double Dash Parameters can be set:

Parameter	Type	Range	Default	Description
USESTARTINGPOINT	binary	$\{0,1\}$	1	If set to 1, the deterministic problem is solved to obtain a starting point
USETRUSTREG	binary	$\{0,1\}$	1	If set to 1, trust region approach will be used
DYNAMICTRUSTREG	binary	$\{0,1\}$	1	If set to 1, dynamic trust region approach will be used
NBSCEN	integer	$\{1,\infty\}$	5	This parameter specifies the number of scenarios
NBCLUSTER	integer	$\{1,\infty\}$	%NBSCEN%1	Number of cluster cuts
BENDERSITERLIM	integer	$\{1,\infty\}$	300	Benders iteration limit
CONVERGENCECRIT	float	$[0,\infty]$	10^{-5}	Convergence criterion (difference between best found solution and best bound)

Table A.14: Parameters for Stochastic Benders Decomposition Approach for the extended SIMPLE Model

The Stochastic Benders Decomposition approach that uses MPI is implemented in `spBendersMPI.gms`.// Below we show an exemplary submission script to solve a 2-stage Stochastic Problem version of SIMPLE with 100 Scenarios on the JURECA Supercomputer at the Juelich Supercomputing Centre (JSC). Note that this example results in 101 tasks where we can (and do) use multiple threads (4) for the solution of every single LP. The number of threads to use by the LP Solver (CPLEX) is for this example hard coded in an automatically generated solver option file. Altogether, the example runs 101 GAMS Jobs and uses 17 nodes, 404 cores, and 4 cores per solve statement.

```
#!/bin/bash -x
#SBATCH --nodes=17
#SBATCH --ntasks=101
#SBATCH --cpus-per-task=4
#SBATCH --output=mpi-out.%j
#SBATCH --error=mpi-err.%j
#SBATCH --time=01:00:00
#SBATCH --mail-user=<name>@<mail>.com
#SBATCH --mail-type=ALL
#SBATCH --partition=batch

module purge
module load Python/3.6.5
module load GCC/5.5.0  ParaStationMPI/5.2.1-1
module load GCC/7.3.0  ParaStationMPI/5.2.1-1
module load Intel/2018.2.199-GCC-5.5.0  IntelMPI/2018.2.199
module load Intel/2018.2.199-GCC-5.5.0  ParaStationMPI/5.2.1-1
module load mpi4py/3.0.0-Python-3.6.5

srun gams simple --to=0.25 --method=SPBENDERSMPI --NBREGIONS=10 --NBSCEN=100
        fileStem=SPBENDERS_MPI_ fileStemApFromEnv=PMI_RANK lo=2 reslim 999999
```

A.6 SIMPLE Model for PIPS-IPM - `simple4pips.gms`

`simple4pips.gms` is another standalone version of the SIMPLE model, specifically tailored for the usage with parallel interior point solver PIPS-IPM. In addition to minor changes in the model structure compared to `simpleBase.gms` , it implements various model annotations and several ways to generate the model in a format readable by the GAMS/PIPS-IPM-Link. In addition, load shifting can be activated and parameterized for simple4pips. With load shifting activated, a certain portion of the demand in a particular time step does not have to be satisfied in this time step but demand satisfaction can be *shifted* to subsequent time steps. Since load shifting links different time steps, it results in additional (sparse) linking constraints.
In Addition to the generic parameters, there are some parameters that are dedicated to `simple4pips.gms`.

Parameter	Type	Range	Default	Description
NBSHIFTS	integer	$\{0, 1, ..., n\}$	0	The number of target time periods to which demand satisfaction can be shifted. If for example `--NBSHIFTS=2 --SHIFTSTEPSIZE=1` are set, demand that is not satisfied in time step t can be satisfied in $t+1$ and/or $t+2$. If `--NBSHIFTS=0` (default), no load shifting is allowed and the value of `SHIFTSTEPSIZE` is irrelevant.
SHIFTSTEPSIZE	integer	$\{1, 2, ..., n\}$	2	The number of time periods demand satisfaction is shifted per shift step. If for example `--NBSHIFTS=2 --SHIFTSTEPSIZE=3` are set, demand that is not satisfied in time step t can be satisfied in $t+2$ and/or $t+4$ and/or $t+6$. If `--NBSHIFTS=0` (default), no load shifting is allowed and the value of this parameter is irrelevant.
METHOD	string		standard_lp	This parameter specifies the method to apply. Possible options are: `standard_lp` The model is solved with CPLEX' barrier algorithm without crossover. `PIPS` The model is generated in PIPS-IPM input format. Note that depending on parameter `BLOCK`, one or more gdx files will be created which changes the process of passing the model instance to PIPS-IPM).
BLOCK	integer	$\{-2, -1, 0, 1, ..., n\}$	-2	This parameter is only relevant if `--METHOD=PIPS` is set. It controls the generation of block files. Possible options are: `-2` Generation of a single gdx files that contains the entire (annotated) model instance. `-1` Sequential generation of all the small block files. `0,1,2,...,n` Generation of a single small block file. This is mainly of interest for distributed model generation.
SCALING	binary	$\{0, 1\}$	0	This parameter determines if a scaled version of the model with better numerical properties should be generated (1) or not (0)
NOSLACK	binary	$\{0, 1\}$	0	This parameter determines whether the $SLACK$ variables should be removed from the model (1) or not (0)

Parameter	Type	Range	Default	Description
KEEPVENAMES	binary	$\{0,1\}$	0	This parameter determines whether the block gdx files are generated with full variable and equation names (1) or if those names are squeezed out (0). Squeezing out the names leads to significant speedups (and reduction in file size) in the further process of passing on a model instance to PIPS-IPM. GDX files with squeezed out names contain the string "noVE-names" in their file name. In fact, adding this string to the filename of the Jacobian created by ConvertD is what triggers the variable and equation names to be squeezed out.
KEEPUELS	binary	$\{0,1\}$	1	This parameter determines whether the block gdx files are generated with GAMS UELs (1) or not (0). This further increase the efficiency of the GAMS/PIPS-IPM-Link but is currently turned off by default. Model instances stored without UELs can only be solved via GAMS/PIPS-IPM but not via solveJacobian.gms. GDX files with squeezed out UELs contain the string "noUELs" in their file name. In fact, adding this string to the filename of the Jacobian created by ConvertD is what triggers the UELs to be squeezed out.
SUPPRESSDM	binary	$\{0,1\}$	1	This parameter determines whether at creation of gdx files an additional dictionary gdx file is created which allows to map original GAMS variale and equation names to variables and equations in the gdx file(s). This is mainly useful for debugging purposes
SLICE	integer	$\{0,1,2\}$	0	This parameter is only relevant in the context of distributed model generation when `BLOCK` $\geq$ 0. It controls the degree of sliced data reading and sliced model annotation. Possible options are: `0` The entire data is read, the entire model is annotated, single block is generated. `1` The entire data is read, relevant model slices (time steps) are identified and annotated, single block is generated. `2` For time indexed data, only the relevant slices are read, relevant model slices (time steps) are identified and annotated, single block is generated. Note that for this approach, the input data must be pregenerated once and stored in a gdx file. This can be done by running `gams simple_data_gen.gms --NBREGIONS=n --WRITEGDX=1`.

Table A.15: Parameters for `simple4pips`

Appendix B

Installation and Solving of SIMPLE Model on Hazel Hen

This appendix describes the installation steps for PIPS-IPM and GAMS software on Hazel Hen. You can also find instructions about how to write a job script for SIMPLE Model and submit it for execution on Hazel Hen. The quick start quide can be also found in the branch „hazelhen“ of PIPS-IPM Git repository under the directory „CRAYXC40/docs“.

::::: H L R S

BEAM-ME:

Installation and Solving of SIMPLE Model on Hazel Hen with GAMS-PIPS

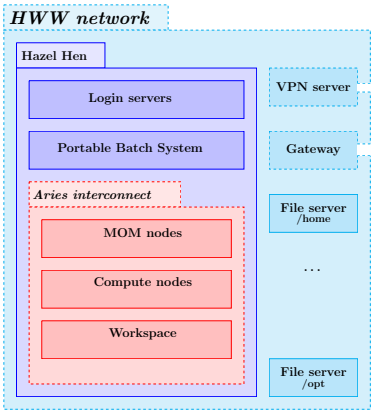
Thomas Breuer JÜLICH Forschungszentrum Dmitry Khabi H L R S

Frederik Fiand GAMS Daniel Rehfeldt ZIB

7. Juni 2019

1/28 :: *BEAM-ME:*Installation and Solving of SIMPLE Model on Hazel Hen with GAMS-PIPS :: 7. Juni 2019 ::

Hww Network and Hazel Hen



← HWW network

- Hazel Hen is in hww network.
- Login nodes are used for login, compiling, edit and serial execution of the scripts and submitting of the computational jobs (PBS-jobs).
- Home-directory is suitable for serial an access (e.g. from Login nodes): File system is not parallel (NFS).
- Aries network is a high performance network.
- Workspases are managed by a parallel file system (Lustre).
- A PBS-job script is executed on a MOM node (serial);
- *aprun* is the ALPS (Application Level Placement Scheduler) application launcher. *aprun* is started on a MOM node and distribute the application between reserved compute nodes. For details about ALPS see the link wiki.hlrs.de.

Installation Requirements of GAMS-PIPS on Cray XC40 Hazel Hen

- User account on gitlab.version.fz-juelich.de at JSC and get an access to the repository „PIPS_beamme“;
- User account on Cray XC40 Hazel Hen at HLRS;
- Acces to hww.hlrs.de:
  - Static IP Address;
  - Fortivpn Connection;
  - ssh client ($ssh -l username hazelhen.hww.hlrs.de -X);
- The „GAMS“ package and it's license are installed globally for the users of the group zib44088[1] (project BEAM-ME).
- Third Party Software Licenses for used libraries:
  - The Academic Licence for the „Paradiso“ library can be required under Pardiso homepage.

[1] see related bash command: *man groups*; *man newgrp*

Third-Party Components an Module Environments

GAMSPIPS makes use of third-party components:

- *Pardiso ASL*,*CBC*,*ConicBundle*,*Metis*,*MA27*,*MA57*,*HSL* . . .
  - BEAM-ME project partners have read acces to the global installation containing the third-party libraries:
    `/opt/hlrs/numlib/beam-me/`;
- *cray-libsci (BLAS, LAPACK, SCALAPACK*, . . .)
  - Avalaible among others on Hazel Hen through Environment Modules package wickie.hlrs.de
    - $module av -S *keyword* #search available modules for a keyword, e.g. libsci
    - $module load *module name* #load module
    - $module show *module name* #print parameters of module
    - $module list #print current loaded modules
    - $module swap PrgEnv-cray PrgEnv-gnu # change Cray programming environment (compiler, std-libraries . . .) to GNU programming environment

Note: No actions are required on this slides due to the global installation of third party libraries. If you want to use your own installation, the appropriate actions will be explained below.

Adaptation of Bash environment on Hazel Hen (Cray XC40)

To work more efficiently, you need to make the bash environment on the login node more user friendly. The bash environment is set by the bash script „${HOME}/.bashrc“ at the start phase of each login. You can also force the new changes with the command „source“(details are in a man: *man source*). [2]

Activity 1: Customize your bashrc on the login node „esloginXXX“to your requirements and taste. For example:

```
@eslogin:~$ vim ~/.bashrc
# for setting history length see HISTSIZE and HISTFILESIZE in man pages of bash
export HISTSIZE=-1
export HISTCONTROL=ignoredups:ignorespace
...
# check the window size after each command
shopt -s checkwinsize
:wq
@eslogin:~$ source ~/.bashrc
```

Note: Bashrc example matching the Hazel Hen can be found under the link `https://gitlab.version.fz-juelich.de/breuer1/PIPS_beamme/blob/hazelhen/CRAYXC40/scripts/hazelhen_bashrc`. Among other things, the title of the command line is adapted to the format „user@host:dir$ *usercommand*“.

[2] Bash properties are not to be confused with the terminal program configuration on your local pc (e.g. „MobaXterm “ for Windows OS or “Xfce Terminal Emulator“ for Linux OS).

Bypassing HWW-Firewall for Git

In order to check out the sources of the GAMS-PIPS, you must go throuh the firewall of hww network. This can be done with SSH port forwarding. The example below shows how port forwarding can be done if the client is a linux pc. For more details about the port forwarding see the link wickie.hlrs.de

Activity 2: Open SSH Tunnel on your local pc.

```
@localpc:~$ ssh -N -D 1080 localhost
```

Activity 3: Open the second terminal and connect to Hazel Hen.

```
@localpc:~$ ssh -R 7777:localhost:1080 hazelhen.hww.de -X
@eslogin:~$ module load tools/git/2.8.0
```

Activity 4: Setup .gitconfig file in your home directory on Hazel Hen. Alternatively, the special command line parameters can be used for the git.

```
@eslogin:~$ vim ~/.gitconfig
[http]
       proxy = socks5://localhost:7777
[https]
       proxy = socks5://localhost:7777
[user]
       name = NAME
       email = EMAIL
:wq
```


Clone Repository with source code

All scripts and sources of „GAMS-PIPS“ (not the third-party components) are available in a Git repository:

Activity 5: Clone the PIPS_beamme repository and check out the branch „hazelhen“

```
@eslogin:~$ module load tools/git/2.8.0
@eslogin:~$ mkdir BEAM-ME #create project directory
@eslogin:~$ cd BEAM-ME #change into the project directory
@eslogin:~/BEAM-ME/$ module load tools/git/ #load the newest av. git
@eslogin:~/BEAM-ME$ git clone \
       https://gitlab.version.fz-juelich.de/breuer1/PIPS_beamme.git
@eslogin:~/BEAM-ME$ cd PIPS_beamme/
@eslogin:~/BEAM-ME/PIPS_beamme$ git checkout hazelhen
@eslogin:~/BEAM-ME/PIPS_beamme$ git branch
* hazelhen
  master
@eslogin:~/BEAM-ME/PIPS_beamme$ cd .. #back to the project directory
```

GAMS-PIPS Setup (1/2)

There are predefined setups to compile the „GAMS-PIPS“ sources.

- HAZELHEN_CRAY: Production version of „GAMS-PIPS“
- HAZELHEN_CRAY_CRAYPAT: Developer version (can be unstable). „GAMS-PIPS“ will be automaticaly instrumented by CRAY-PAT. The compiler is cray;

Note: Not to be confused with the „GAMS“ package. The „GAMS“ package and all needed additional third parties libraries are installed globally for the users of the group zib44088[3](project BEAM-ME). „GAMS“ can also be downloaded from the Internet GAMS Wiki and installed in the user's home or a workspace directory.

[3] see related man pages: *man groups*; *man newgrp*

GAMS-PIPS Setup (2/2)

The setup scripts offer several options to adjust the installation to fulfil the user requirements. For example, the user can change

- **source directories:** the path to the sources (clone directory) and to the third parties libraries.
- **build directories:** the path to the installation directory of „GAMS-PIPS“.
- **compiler directives:** the compiler directives, so called macros, which switch the various components of the software (e.g. used solvers).
- **debug/production mode:** the level of compiler optimizations.

Activity 6: Change to the top directory of the BEAM-ME project and view or edit if needed the installation script.

```
@eslogin:~$ vim PIPS_beamme/CRAYXC40/install_HAZELHEN_CRAY.sh
 1 #!/bin/sh
 2 #Project BEAM-ME (see details on http://www.beam-me-projekt.de)
  ...
17 export PROJECT_PATH=$HOME/BEAM-ME/PIPS_beamme #Directory with the sources
  ...
21 export OPT_LEVEL=3 #0(debug) or 1(release) optimization level;
:wq
```

GAMS-PIPS Compiling

Activity 7: Grant the execution right to the installation script and execute it.

```
#change premission of the installation script:
@eslogin:~/BEAM-ME$ chmod 755 PIPS_beamme/CRAYXC40/install_HAZELHEN_CRAY.sh
@eslogin:~/BEAM-ME$ PIPS_beamme/CRAYXC40/install_HAZELHEN_CRAY.sh
...
~ 30 seconds
INSTALLATION(PRODUCTION) DONE!!!
#verify if you compilation was successful:
@eslogin:~/BEAM-ME$ ls ./PIPS_beamme_HAZELHEN_NEW/
... ... gmspipschk ... gmspips ...
```

Note:A quick reference to the parameters of the bash command „chmod“ can be found in the man pages and under `https://ss64.com/bash/chmod.html`.

Data Structure for SIMPLE model on Hazel Hen (1/3)

The GAMS-PIPS computational jobs will be submited from the workspace MODELJOB.

```
/lustre/cray/ws8/ws/$USER-MODELJOB
└─ simple_v2
    ├─ job_simple_v2.sh
    ├─ job_simple_v2_template.sh
    ├─ cmd_aprun_simple_lc
    ├─ start_scale_exp.sh
    ├─ job_scripts
    │   └─ ..
    └─ messages
        └─ ..
```

MODELJOB - *Workspace for GAMS-PIPS.*

simple_v2 - *Top directory for the SIMPLE model.*

job_simple_v2.sh - *Bash script to generate the PBS[a] job scripts for SIMPLE with various parameters.*

job_simple_v2_template.sh - *Template of PBS job scripts for SIMPLE.*

cmd_aprun_simple_lc - *Support utility used in „job_simple_v2.sh“* (compute arguments of aprun command).

start_scale_exp.sh - *Bash script utility to start a parameter study for various sizes of time blocks of the SIMPLE model.*

job_scripts - *Subdirectory for the generated PBS job scripts.*

messages - *Subdirectory for the standard and error output of the executed PBS jobs.*

[a]PBS - Portable Batch System is a job scheduler mechanism to allocate needed computing r

Data Structure for SIMPLE model on Hazel Hen (2/3)

The global schared directory under „/opt/ “ contains, among others, various librtaries of the GAMS package and the model's data.

```
/opt/hlrs/numlib/beam-me/
├── extlibs
│   └── pardiso
├── gams
│   └── gams25.1_linux_x64_64_sfx
│       ├── gams
│       └── ...
├── PIPS_beamme
│   └── HPC_Workshop_2018
│       ├── gmspips
│       ├── gmspipschk
│       └── ...
└── models
    └── simple_v2
```

extlibs - *Global installation of the third parties libraries.*

gams - *Global installation of the GAMS package.*

PIPS_beamme - *Global installation of the GAMS-PIPS (version HPC-Workshop2018).*

models - *Global subdirectory with the SIMPLE model.*

Data Structure for SIMPLE model on Hazel Hen (3/3)

The second workspace „data“ contains the data, which will be generated during the execution of the PBS job scripts for the SIMPLE models.

```
/lustre/cray/ws8/ws/$USER-data
├── data
│   └── ..
├── performance
│   └── PERFORMANCE_REG_10
│       ├── max_performance_rank.dat
│       └── ...
└── tmp
    └── ...
```

data - *The subdirectories with the output of GAMS-PIPS.*

performance - *The subdirectories with the performance data of GAMS-PIPS.*

tmp - *The subdirectories to store the generated blocks of the models.*

Note: The name of the subdirectory containing the solution and performance data depends on the parameters of the SIMPLE model, which was solved with GAMS-PIPS.

Prepare Workspaces for GAMS-PIPS (1/3)

Activity 8: Create your personal workspaces on Hazel Hen (case sensitive) :

```
@eslogin:~/BEAM-ME/PIPS_beamme/$ ws_allocate MODELJOB 31 #case sensitive
@eslogin:~/BEAM-ME/PIPS_beamme/$ ws_allocate data 31 #case sensitive
@eslogin:~/BEAM-ME/PIPS_beamme/$ ws_list #output is reformatted
Filesystem: default #for a better overview
Workspace ID Workspace location Filesystem Remaining time
--------------------------------------------------------------------------
id: MODELJOB /lustre/cray/ws8/ws/... ws8 30 days 23 hours
...
id: data /lustre/cray/ws8/ws/... ws8 30 days 23 hours
available extensions:3
#E-mail with a cal. entry reminder about expiration of the remaining time
@eslogin:~/BEAM-ME/PIPS_beamme/$ s_send_ical MODELJOB -m your_email
```

Note: Both the output and the input of a parallel program should basically done in parallel file system. This also applies to GAMS-PIPS.
Hazel Hen uses a parallel file system called „Lutre“[4]. That filesystem saves files into the so-called “workspaces“. The workspace „ws8“ has been assigned to the group „zib44088“.
For details about the workspace mechanism on Hazel Hen see the link wickie.hlrs.de

[4] http://lustre.org/

Prepare Workspaces for GAMS-PIPS (2/3)

Activity 9: Create the subdirectories in the assigned workspaces necessary to run „GAMS-PIPS“. The subdirectories names are case sensitive.

```
@eslogin:~/BEAM-ME/$ mkdir /lustre/cray/ws8/ws/${USER}-data/data
@eslogin:~/BEAM-ME/$ mkdir /lustre/cray/ws8/ws/${USER}-MODELJOB/simple_v2
@eslogin:~/BEAM-ME/$ cd /lustre/cray/ws8/ws/${USER}-MODELJOB/simple_v2
#copy dirs with the support tools from the sources into the workspace
@eslogin:../simple_v2$\
 cp -r ~/BEAM-ME/PIPS_beamme/CRAYXC40/workspace/simple_v2 ./
#grant execution premission to the tools
@eslogin:../simple_v2$ chmod 755 cmd_aprun_simple_lc
@eslogin:../simple_v2$ chmod 755 job_simple_v2.sh
#create a sub-directory for the standard output of GAMS-PIPS
@eslogin:../simple_v2$ mkdir messages
#copy the compiled GAMS-PIPS files into the workspace
@eslogin:../simple_v2$ cp ~/BEAM-ME/PIPS_beamme_HAZELHEN_NEW/gmspips ./
@eslogin:../simple_v2$ cp ~/BEAM-ME/PIPS_beamme_HAZELHEN_NEW/gmspipschk ./
```

Note: The bash environment variable „$USER“ stores the username of the user. This can of course be replaced by your certain username.

Prepare Workspaces for GAMS-PIPS (3/3)

If you are going to use the global installation of GAMS-PIPS software, you must adjust the environment variable „GAMSPIPS_BIN_PATH“ in „job_simple_v2.sh“ or copy the executables „gmspips“ and „gmspipschk“ into the workspace directory.
If you are going to use the self compiled GAMS-PIPS software, you must copy the executable files from your own GAMS-PIPS installation directory.
Also, you must create the license file for your pardiso license.

Activity 10: Copy the compiled GAMS-PIPS files and create the license file.

```
#copy the compiled GAMS-PIPS files into the workspace
#not needed if you are going to use the global version of GAMS-PIPS
@eslogin:../simple_v2$ cp ~/BEAM-ME/PIPS_beamme_HAZELHEN_NEW/gmspips ./
@eslogin:../simple_v2$ cp ~/BEAM-ME/PIPS_beamme_HAZELHEN_NEW/gmspipschk ./
#Set the execution permision
@eslogin:../simple_v2$ chmod 755 ./job_simple_v2.sh
@eslogin:../simple_v2$ chmod 755 ./cmd_aprun_simple_lc
#Create a new file for the pardiso license
#and paste your license key matching your username.
#Prevent any additional symbols in the file.
@eslogin:../simple_v2$ vim pardiso.lic
186ED67575582D93...
:wq
```


Generate a Job Scripts for the SIMPLE Model (1/2)

The bash script „job_simple_v2.sh“ will generate the correct PBS job script for a parameterized SIMPLE model. The script must be called with certain parameters.
The parameters and their sequence can be looked up in the script or listed in the terminal by executing the script without parameters.

Activity 11: View the bash script “job_simple_v2.sh“ or print out the help message.

```
@eslogin:../simple_v2$ vim job_simple_v2.sh
  1 #!/bin/sh -f
  2 #SOLVE SIMPLE MODEL WITH GAMS AND PIPS
  ...
  16 echo "=="
  17 echo "Usage: ./job_simple_v2.sh NUM_NBREGIONS ..."
  ...
:q!
#print out the help message
@eslogin:../simple_v2$ ./job_simple_v2.sh
========
Usage: ./job_simple_v2.sh NUM_NBREGIONS ...
...
```

Generate a Job Script for the SIMPLE Model (2/2)

The execution of the script „job_simple_v2.sh“ will generate a new PBS job script for solving a SIMPLE model mapping the specified parameters.
The last command in the script is to issue the PBS job script submission command.

Activity 12: View the bash script “job_simple_v2.sh“ or print out the help message.

```
@eslogin:../simple_v2$ ./job_simple_v2.sh 15 1.0 381 1.0 0 1 scaleGeo you_email
....
===SUBMIT COMMAND===
qsub /lustre/...PROCS_23_NODE_1.sh
```

Note: The global installation of the SIMPLE Model contains the data for the models with 10, 15 and 20 regions.

Submitting the Job Script for the SIMPLE Model (1/2)

The generated job script is configured to execute GAMS-PIPS on one node and will be submitted into the queue „test“. The maximal walltime of the job is 25 minutes. If you need more times or more than 16 nodes, you must change the queue or let the field empty. This can be done also in the file „job_simple_v2.sh„ for all further generated job scripts.

Activity 13: Open the job script and check the parameters

```
@eslogin:/lustre/../simple_v2$ vim job_scripts/job_simple_v2...23_NODE_1.sh
#!/bin/sh -f
#PBS -o /lustre/cray/../messages/$PBS_JOBID.out
#PBS -e /lustre/cray/../messages/$PBS_JOBID.err
#PBS -m bea #mail alert at start, end and abortion of execution
#PBS -M youremail #send mail to this address
#PBS -l walltime=00:25:00
#PBS -l nodes=1:ppn=24
#PBS -q test
...
export PARDISO_LIC_PATH=...
...
```

Submitting a Job Script for SIMPLE Model (2/2)

Activity 14: Submit the job script to the queue of Haezel Hen:

```
@eslogin:/lustre/../simple_v2$ qsub job_scripts/job_simple_v2_..._NODE_1.sh
@eslogin:..-PIPS-JOBS/PRODUCTION$ qstat -u $USER

hornet-batch.hww.de:

Job ID Username Queue Jobname SessID NDS TSK S ...
-----------------------
922175.hazelhen-batch.hww.hlrs.de username small job.pbs -- 1 24 Q ...
```

The standard and error output of the job will be saved in the sub-directory „messages“.
The command -u $USER“ chekcs the state of the job:

- Q - The job is in the queue.
- R - The job is running.
- C - The job is completed.
- H - The job is halted (if the job were submitted with a dependency on another job)

For details about the batch system see the link wickie.hlrs.de;

Data Output of GAMS-PIPS on Hazel Hen

The solution of the SIMPLE models will be stored in the subdirectories of „data“. The temporal files of the SIMPLE models are stored in the subdirectories of „tmp“(see slide 15).
Please, delete these subdirectories if not needed:
@eslogin:..._vs2$ rm /lustre/cray/ws8/ws/$USER-data/tmp/ -rf*
Note: The GAMS-PIPS version „HPC-Workshop2018“ doesn't support the data output.

Performance Counters of GAMS-PIPS on Hazel Hen (1/2)

There are performance files „max_performance_rank.dat“ in the sub-diretories of „performance“. Each of the files stores the csv spreadsheet with the performance statics. The statistics for the SIMPLE models parameterized with different number of regions are saved in the separated subdirectories. The most important counters are

- **JOBID** - PBS job id is the unique number assigned before job execution by PBS system.
- **NUM_NBREGIONS,TIME_HORIZONT,NUM_NBSCEN,NUM_BLOCKS,...** - Used SIMPLE model parameters.
- **NUM_THREADS** - Current version of „GAMS-PIPS“ on Hazel Hen uses 1 Thread.
- **MPI_SIZE** - Number of MPI-Processes used by GAMS-PIPS. One block is assigned to each of the mpi processes. Block 0 contains the linking constrains und is shared.
- **NUM_NODES** - Number of compute nodes used by execution of GAMS-PIPS.
- **CREATE_GDX_TIME** - Time in seconds to generate a parameterized SIMPLE model with „gams“.
- **GDX_SPLIT_TIME** - Time in seconds to split the model into the blocks in parallel with „gams“.
- **TOTAL_GMSPIPS_TIME** - Time in seconds to find and store the solution of the model with „GAMS-PIPS“.

Performance Counters of GAMS-PIPS on Hazel Hen (2/2)

- **MKDIR_TIME** - Time to prepare the structure of the directories at the beginning of the job script.
- **PIPSINTERFACE_TIME** - „GAMS-PIPS“ Time to read in the blocks of the matrix.
- **0-LINK_VARIABLES, 2-LINKS_EQUALITY, ..., LINKED_RATIO** - Properties of the model's linear system.
- **SUCCESSFUL** - „GAMS-PIPS“ status, 0 -success.
- **DUALITY_GAP,OBJECTIVE,MU,RESIDUAL_NOR** - Properties of the solution.
- **TOTAL_IPM_SOLVER_NUM_ITER** - Number of iteration in interior point method.
- **TOTAL_SCHUR_INDEF_SOLVER_TIME** - Time in seconds to solve the schur complement.

Command line arguments of GAMS-PIPS (1/2)

„The parallel „GAMS-PIPS“ processes will be distributed and launched on the compute nodes by ALPS with a command „aprun“(See a job script for SIMPLE model).
The number of needed mpi processes and other parameters will be calculated in the script „job_simple_v2.sh“(line 120) and used wenn assembling the command line of GAMS-PIPS(line 262).
The main arguments:

- $-j < int >$: PBS Job-ID;
- $-n < int >$: number of blocks;
- $-f < string >$: GDX filename;
- $-d < string >$: GDX dirname;
- $-o < string >$: Output filename;
- $-a < string >$: Output dirname;
- $-s < 0/1 >$: Use scale property in solver (scale,scaleEqui,scaleGeo,scaleGeoEqui);
- $-l < 0/1 >$: Use stepDiffLp property in solver (default:1);
- $-i < 0/1 >$: Use Presolver (default:1);
- $-w < 0/1 >$: Write out the solution (default:0);
- $-c < int >$: Maximum number of iterations (default: 200);

Command line arguments of GAMS-PIPS (2/2)

The further parameters are optional and used to gather the statistics of the entire workflow in a single performance file „max_performance_rank.dat“.
Profiling arguments :

- $-t < float >$: time horizont;
- $-r < int >$: number of regions;
- $-z < string >$: How many scenarios has the instance;
- $-k < int >$: Number of omp threads (fixed to one);
- $-1 < int >$: number of nodes (optional);
- $-2 < float >$: time of mkdir (optional);
- $-3 < float >$: time of GAMS (optional);
- $-4 < float >$: number of produced GAMS files (optional);
- $-5 < float >$: size produced GAMS files in Bytes (optional);
- $-6 < string >$: directory for the performance statistic;
- $-7 < float >$: time of GAMSPIPSCHK (optional);

Outlook

The presented procedure can also be applied to other modles than SIMPLE.
The corresponding filenames and other parameters can easily be replaced in the script „job_simple_v2.sh“.
However, if your model can not be splited into the blocks in parallel with „gams“, you'll need to comment this step in the job script (or job template). The blocks must be generated separately.
The reason for this is that running GAMS PIPS need more than one compute node. On the other hand, generating the blocks with gamspipschk can only be done with one process. Thus, many nodes would be empty during block generation, which is very inefficient. For small models you can generate the blocks on a login node and use the „tmp“ subdirectory in the workspace „data“ to save the data.
If you are going to split the large models, use an additional PBS job script and split the models also on a compute node.

The End

Appendix C

HPC Node-Level Architecture

The following section describes the operating principles of CPU components (incl. main memory) and its mikroarchitecture limitations. To demonstrate the potential bottlenecks of an HPC system, two processor models are considered, namely „Haswell Extrem Platform“ and „Skylake Scalable Performance“ (hereinafter referred to as Haswell and Skylake).

C.1 Semiconductor Technology and its Limitations

A modern processor consists mostly of several hundred million to billions of tiny switches realized with two types of transistors: N-channel and P-channel MOS field effect transistors (MOSFET). These transistors, wired in groups, are used as building blocks for the Metal-Oxide-Semiconductor Elements (CMOS), which form the basis for the construction of all the logical gates of a processor. Thus, a processor can also be considered as an integrated circuit. An integrated circuit, often mentioned as chip or die, is built up on a substrate of semiconductor, so-called wafer. Photolithography, the key of the production process, projects the patterns with the processor's circuits throught the masks on a piece of the substrate.

The size in nanometers (nm) indicates the smallest edge length of the mask pattern. The smaller the mask details, the more transistors can be integrated into the same chip area. However, the maximum size of a chip is strongly limited by the high error rate during the manufacturing process, as otherwise the manufacturing costs would increase significantly. In addition, a larger chip must be clocked lower, since the maximum permissible temperature of a larger chip is generally lower than that of a smaller chip due to material fluctuations.[1] It is generally true to say that the larger the area of a chip is, the more likely errors in manufacturing are to occur and the slower the supported operating frequency of the chip is.

Figure C.1 shows a simple CMOS inventor with two transistors that implements a non-logic gate. Because the electrical components are unable to use the logic values, these are represented by two voltage levels in digital binary coded signals: the high level and the low level. The high level almost corresponds to the operating voltage ($V_{dd}{}^{+}_{-}$) and the low level is close to 0 volts ($V_{ss}{}^{+}_{-}$).

C.2 CPU Frequency and Power consumption

To execute a machine command, certain CMOS elements of the processor must be concatenated and synchronized. A clock signal is used for the synchronization purpose.

Such synchronization prolongs the latency time of the machine commands. A higher operating voltage V_{dd} can be used to shorten the response time of the transistors and thus the latency time of the machine commands: The higher the operating voltage, the faster the transistors switch over. But it also means a much higher electrical power

[1] During the writing of this document, a new world largest processor chip of 2.1 trillion transistors was reported (Cerebras, 2019). The chip is more than 100 times larger than all previous processor chips and 56 times larger than the largest GPU. The price and other technical details are not during the writing of this document.

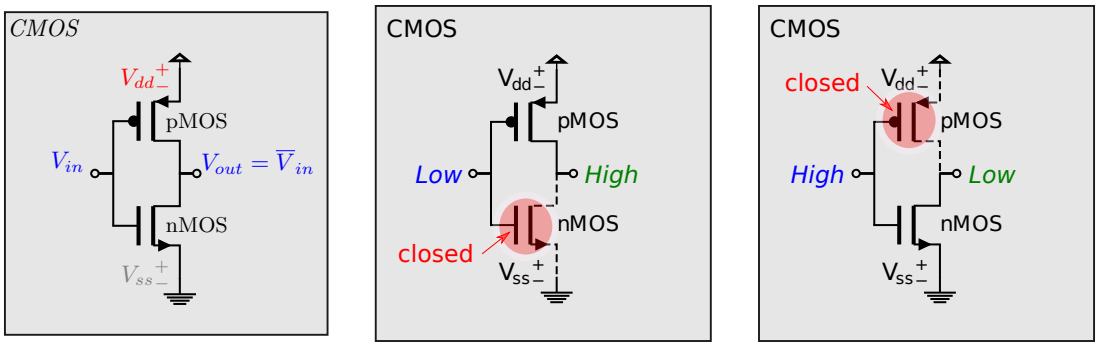


Figure C.1: Logisches Gatter „Inverter“ aus zwei MOSFETs

In figure C.2 that synchronization signal is marked with „Clk“. If, for example, the output signal V_{out} is used as an input signal for another, much more complex CMOS element, this second element should only be active on the rising slope in the following cycle. Otherwise it may happen that the resulting output signal is in an undefined state at the beginning of the next cycle (see red area in Figure on the left). Thus the bits - in the form of high and low signals - travel through the signal lines from one CMOS element to another and are transformed accordingly. Such chains of CMOS elements are dynamically linked according to the executing micro-operations. Unlike a processor, the corresponding chains of CMOS elements are mostly fixed in FPGA boards or ASICs. This may higly improve both latency and power dissipation of the computational units.

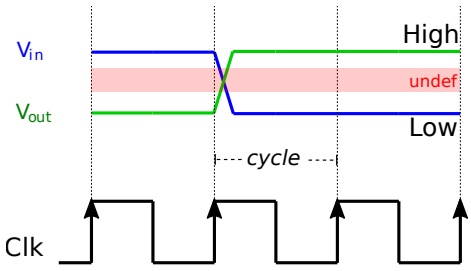


Figure C.2: Clock, input and output signals of a CMOS inverter

consumption of the processor. This can be critical as all the electrical power is converted into heat. The heat must be dissipated to avoid damaging the processor.

Equation (C.1) shows an approximation of power comsumption for a CMOS (Chandrakasan, Sheng, & Brodersen, 1995).The power approximation consist of three components: static, dynamic and short-circuit.

$$
\begin{aligned}
P_{cmos} &= P_{static} + P_{dynamic} + P_{shortcircuit}; \\
P_{static} &= I_{leakage} \times V_{dd}; \\
P_{dynamic} &= \alpha \times C_L \times {V_{dd}}^2 \times f_{clk}; \\
P_{shortcircuit} &= I_{sc} \times V_{dd}; \\
I_{leakage}, I_{sc} &: \textit{ Leakage and short-circuit current;} \\
f_{clk} &: \textit{ Operating frequency of the integrated circuit;} \\
V_{dd} &: \textit{ Operating voltage;} \\
\alpha &: \textit{ Probability of change of state;} \\
C_L &: \textit{ Load capacity of the CMOS element;}
\end{aligned}
\tag{C.1}
$$

The static parts refers to the leakage current, which flows within the closed transistors. The strength of the current increases exponentially as the distance between the drain and the source of the transistors decreases (Fallah & Pedram, 2005). Introduced in 2011, Tri-gate transistors reduced the leakage compared to planar transistors. This and the improvement of the substrate properties allow further refinement of the Semiconductor Technology. It should be mentioned here that the corresponding resistance properties of the substrat material degrades with the increase of the operating temperature. This raises the question of an optimal temperature of the cooling.

The state-switching of the CMOS element causes a short circuit between the line of V_{dd} and the ground line of V_{ss} during the short period when both transistors are opened (see Figure C.2): the higher the voltage, the steeper the pulse edge and the shorter the short circuit.

The dynamic part refers to the power which must be applied to change the state of the transistors. $P_{dynamic}$ increases quadratically with the operating voltage and linearly with the operating frequency. To manage the rapidly increasing dynamic part of power consumption, many CPU manufacturers add a DVFS extension to the processor architecture. DVFS stands for Dynamic Voltage and Frequency Scaling. Figure C.3 indicates the dependency between the core voltage and core frequency during execution of a simple stream kernel Add ($(a[i] = b[i] + c[i]$, see section C for details) on all cores of Haswell processor E5-2680v3[2]. The CPU frequency was set from user space for each of the measurement points.

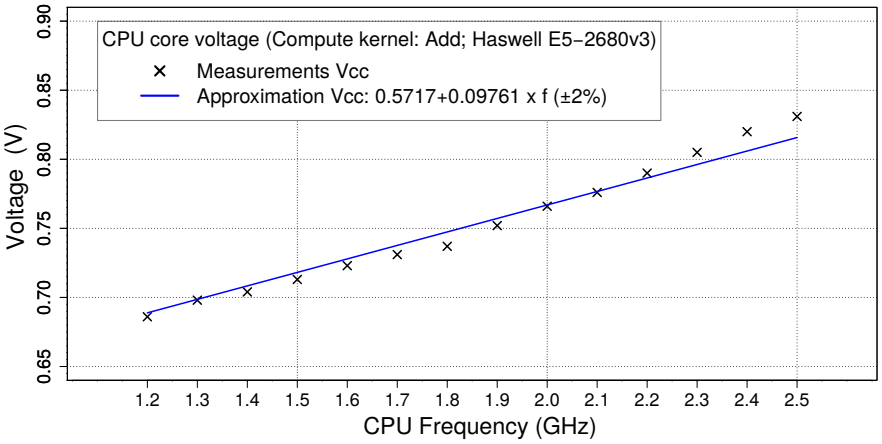


Figure C.3: Dependence between the core voltage V_{cc} and the CPU frequency during the execution of the kernel Add $(a[i] = b[i] + c[i])$ on the processor Haswell E5-2680v3

As the measurements show, the electrical power of an active processor $P_{cpu}(f)$ increases approximately with the power between 2.0 and 2.5 of the CPU frequency.

C.3 Peak Performance and Vectorization

As suggested in the introduction section about High Performance Computing 1.3, HPC addresses various kinds of algorithms. The so-called General-purpose CPU is particularly efficient for the computation with a low $Byte/Flop$ rate. The peak performance is achived at the rate where $Byte = 0$. This shall be the case if all operands and results of the floating point operatons are stored in the particular vector registers and no data dependencies exist.

The width of the vector registers in x86 architecture has not grown significantly in the past 20 years. Intel's first streaming SIMD extension (SSE) introduced in 1999 supports vector registers with two 64-bit double-precision floating-point numbers (*doubles*). The Advanced Vector Extensions (AVX) introduced in 2008 by Intel and 2011 by AMD support four doubles. Figure C.4 shows a schematic representation of one of the two operand AXV instructions, namely addition (*vadd*). The next AVX2 extension was extended with a three operands Fused-Multiply-Add (FMA) instruction set, which can be used to perform two operations, namely addition and multiplication, in one turn.

One Haswell processor of supercomputer Hazel Hen at HLRS (see section 4.2.3) supports AVX2 and can execute up to two FMA instructions simultaneously. Consequently, the Haswell with 12 cores and base frequency of 2.5GHz[3] has a peak performance of $12 \times 2 \times 2 \times 4[\text{Flop}] \times 2.5[\text{GHz}] = 480[\text{GFlops}]$.

The next generation server processors Skylake support Intel Advanced Vector Extensions 512 (AVX-512), whose vector registers can store up to 8 doubles. Thus, one processor of supercomputer JUWELS (Jülich Supercomputing Centre, 2019) at JSC has a peak performance of $28 \times 2 \times 2 \times 8[\text{Flop}] \times 2.7[\text{GHz}] = 2.074[\text{TFlops}]$.

[2]These processors are installed in HLRS supercomputer Hazel Hen. In the production of Haswell processors, the 22 nm lithography are used.

[3]Sometimes one uses a manufacturer-specified maximum turbo frequency, for example here (Intel, 2019c), although many core processors work at such high frequency for a short time. The main reason for it is a high power consumption, which results in a high temperature impact on the chip and the need to lower the frequency.

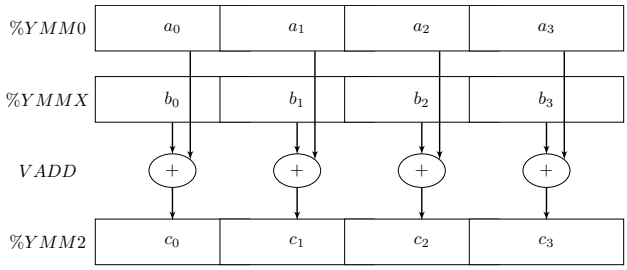


Figure C.4: Execution of a Single-Instruction-Multiple-Data (SIMD) addition on AVX regesters

Not for nothing such a performance is also called theoretical. It is not so easy to find any algorithm with such properties that is useful for real-world applications. The problem must also be large enough to hide the latencies of the floating point operations through pipelining.

For example, an instruction that has a latency of 5 processor clocks will have its data available for another instruction after so many cycles from the beginning of the execution. The throughput of an instruction is the number of cycles required to execute a instruction. For this many cycles it is also prevented that the execution unit, e.g. FMA, is used for one of the next instructions in the pipeline. According to Agner's CPU blog (Agner, 2017), the latency of an FMA on Haswell is 5 cycles and only 4 cycles on Skylake. The throughput of Fused-Multiply-Add instruction is one cycle per instruction for both processors.

Vectorization is a parallel hardware-assisted paradigm, whose approach is automatically supported by the compiler. However, not every loop can be vectorized and many loops cannot be recognized as vectorizable. Except general rules, such as no recursive dependencies in a loop, every compiler and every language has its features, which a developer must consider in order to write a vectorizable code. For example, a vectorization guide for GCC compiler kann be found in (MogonWiki, 2017) and information for Intel Compiler on Intel machines is here (ColfaxInternational, 2017). Attention should also be paid to the compiler options, which are partly dependent on the version of the compiler and processor architecture.

C.4 Core Mikroarchitecture

Each core of a modern General-purposed server processsors is equipped with more than one hundred of the vector registers.[4] These registers are part of the Execution Engine. In addition to the registers, the execution engine consists of pipeline, instruction scheduler, branch prediction and other units for the control of both instruction and data streams within the core and the entire CPU.

Execution of a program consists of many instructions – so-called micro-operations that are processed one after the other in the pipeline. In Out-of-Order unit of the pipeline a micro-operation is divided into the suboperations, which are then executed step by step. If the code to be executed consists of repeating and partially independent micro-operations, as is often the case in loops, then the corresponding suboperations can overlap each other. This increases the overall throughput compared to instruction execution without pipelining. The longest delays in the pipeline, known as bubbles, arise due to loading and store commands.

The available number of the registers is far from sufficient for solving even small problems without storing the data in the memory. In fact, most data is stored outside of the CPU in the main memory (often referred to simply as RAM), which is based on synchronous dynamic random access memory (SDRAM) technology. Due to the low performance of this type of memory compared to the performance of the execution engine, the memory hierarchy has been introduced with different types and sizes of cache levels.

C.5 Cache Mikroarchitecture

A chunk of data handled by the cache is called cache line. This is applied even if only one byte of the data needs to be accessed, the entire cache line must be stored and transfered between the caches and the memory. 64 byte is

[4] A Vector Physical Register File of Skylake and Haswell have 168 registers.

usually the size of a cache line.

The private cache consists mostly of L1 and L2 caches. If the data requested by CPU core is not in the L1 cache, it must be transported from L2 to L1. After that it can be loaded into the registers in the cores. The cache lines associated with the private cache can only be processed by the nearest core.

The static random access cache (SRAM) takes up a significant area of a CPU chip, although even the size of the cache is much smaller than the main memory.[5] The reason for this is that a flip-flop SRAM cell has quite a high fixed cost and occupies a lot more space than an SDRAM cell (see section C.8). What matters is that SRAM is much better in performance than SDRAM.

For the reduction of cache fixed costs one is also forced to introduce „set associative cache“. Fully associative cache means any 64 byte, for example eight consecutively stored doubles from memory, can be copied to any cache line. While such cache microarchitecture is highly flexible in terms of the cache reusing, the costs to lookup for a particular cache line are very high. For this, the entire lookup table of the cache must be searched. For example, the lookup table of 32-KByte fully associative cache has 512 entries. The opposite is 1-way associative cache which means a memory address can be mapped to only a single cache line. That is easier to implement and would make the search in the lookup table fast (e.g. with a simple hash function), but the hit rate would be very low due to the cache thrashing. The reason for that is multiple main memory locations competing for the same cache line.

The L1 and L2 caches of Haswell are 8-way set associative. The L2 cache of Skylake is 16-way.[6] The fastest latency of the L1 cache of both processors Haswell and Skylake is 4 cycles. The fastest latency of the Haswell's L2 cache is 11 [cycles] and of the Skylake's L2 cache between 14 [cycles] and 22 [cycles] (Intel, 2019a).

Figure C.5 shows a simplified scheme of Haswell Processor (E5-280v3) and its memory hierarchy. As can be seen, the interfaces between L1 caches and their cores has a width of 96 byte. At once the data can be read for eight doubles and written for four doubles into and out from the registers. This corresponds to one and a half AVX registers and a bandwidth of 96 byte/cycle. However, a number of additional conditions needs to be met, which we will not address here. The L1 cache interface of Skylake was doubled according to the length of AVX-512 registers. This also means that an FMA operation can be performed without delay only if at least one AVX register is reused as an operand. Otherwise, the FMA operation can not be performed in each clock.

C.6 Last Level Cache

Unlike the private cache, the Last Level Cache (LLC, pictured as L3 cache) is shared between all cores and is used for the data transfer between the local caches and the memory. The entire L3 cache is divided into L3 segments, the so-called multiple cache slices, between all processor cores. A cache line is stored in one of the L3 segments depending on its memory address, associativity of the cache and possibly other parameters. The corresponding unknown relation is called a hash function. According to „Intel 64 and IA-32 Architectures Optimization Reference Manual“ (Intel, 2016), the address space is distributed uniformly between the segments. If a processor core didn't find a requested cache line in L1 and L2 caches, the request will be processed by the local and home cache agents (CA and HA). If the cache line is not present in the nearest L3 segment but in one other, it will be pushed to the recipient via the so-called ring bus. If the data is not stored in any of the cache lines of the L3 cache, the data is loaded from the main memory into one of the L3 segments via the internal memory controllers (IMC).

All this is done while preserving cache coherency through CA and HA. The cache coherence protocol prevents the individual caches from returning different (inconsistent) data for the same memory address. More details about the cache and ring bus can be found, for example, in the paper "Cache Coherence Protocol and Memory Performance of the Intel Haswell-EP Architecture" (Daniel Molka, Daniel Hackenberg, Robert Schöne, & Wolfgang E. Nagel, 2015).

The Skylake processors increased their L2 cache from 256 KB to 1 MB. This comes at the expense of the L3 cache, which is reduced from $\sim$ 2.5MB to 1.375MB per core. Due to the higher core count, Intel has replaced also the ring bus with a mesh interconnect. A detailed description of Skylake architecture can be found in (Wikichip, 2018). Not

[5] There are processors that consist of several chips. In such a case, the chips are tightly interconnected and placed in a package. However, the execution engine and the private cache should be in the same chip for performance reasons.

[6] If the memory access pattern includes long jumps of power two, caution should be exercised to ensure than no cache thrashing occurs.

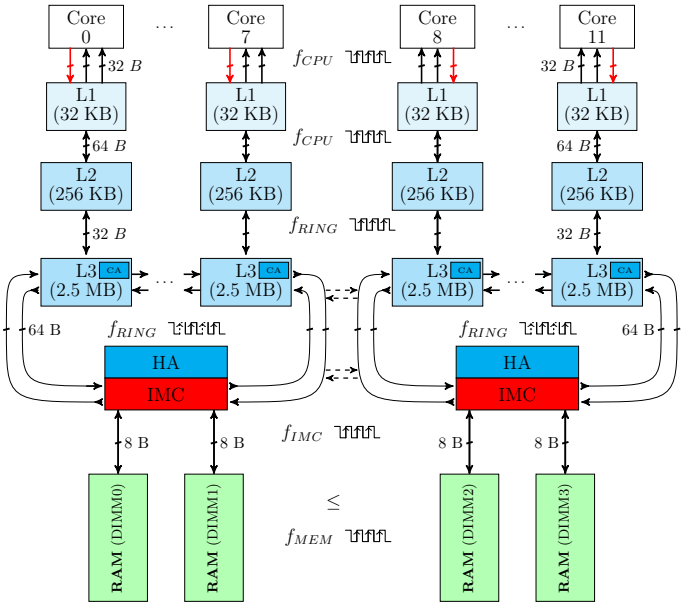


Figure C.5: Haswell Processor (E5-280v3) and its memory hierarchy

only L3 segments and IMC are connected to the mesh and ring bus, but also the other controllers, such as QuickPath Interconnect (QPI)[7] and PCIe.

The next section provides an example of the performance of LLC.

C.7 Cache Performance

Let us consider an example: compute kernel *Add*. The kernel operation "Add" calculates the sum of two fields $b[]$ and $c[]$ and writes the result in the third field $a[]$:

$$a[i] = b[i] + c[i]; 0 \leqslant i < n; \tag{C.2}$$

The computation is started first on one core then on two and so on. The length of the arrays grows linearly with the number of cores, so each core always has the same amount of work to be done. Such experiments are called „weak scaling". In contrast to weak scaling, strong scaling means that the problem size does not increase with the number of active cores. In addition, the CPU frequency is varied for each configuration-number of cores and CPU frequency of the experiment.

The experiment was performed on Skylake SP 6148 with 20 cores and the array length per core was 131072. Accordingly, a core needs enought memory to store $\frac{3 \times 131072 \times 8[B]}{1024 \times 1024} = 3\text{MB}$.[8] This is considerably more than what could be stored in an L3 segment: the processor needs as much cache as two full L3 segments and a little more.

To perform one plus operation two times 8 bytes must be read and 8 bytes must be written. In total, 24 bytes have to be transported between the memory and the registers. However, the caches and memory of Haswell and Skylake SP feature a wire-back policy. When a core writes data to a memory location with a write back policy, the processor first ensures that it has the cache line containing this memory location in the L1 cache (Intel, 2019b). This means that 8 bytes must be loaded additionaly. Since we know that 32 bytes must be transported to and from the AVX registers for one plus operation, we can calculate the average bandwidth of the kernel *Add*.

Figure C.6 shows the results. As expected, performance and therefore the bandwidth increase linearly with the number of cores and CPU frequency until the data fits in theL3 cache. However, when some of the data needs to be

[7]The Intel QuickPath Interconnect is a point-to-point processor interconnect. This allows the multiple processes to share their memory between each other. AMD uses Infinity Fabric to support multi sockets systems.

[8]The hard disk manufacturers calculate 1 KB with 1000 bytes. In HPC the conversion with 1024 bytes is widespread. Accordingly, one megabyte has 2^{20} and one gigabyte 2^{30}bytes.

written to the main memory or read from it, performance degrades rapidly. In addition, the bandwidth for the lowest frequency (drawn in red color) scales linearly for $\sim$ 14 cores while the bandwidth for Turbo frequency scales linearly only up to 10 cores. If only a small part of the data has to be transported between L3 and the main memory, the interface between IMC and the main memory DIMMs (see figure C.5) fulfills the requirements of the cores for the data. This means that the fewer cores and the smaller the frequency, the easier it is for the memory interface to meet the requirements. Unlike the ring and core frequencies, the frequency of the IMC and RAM does not vary (according additional test results). It should be noted that the CPU frequency of Haswell's ring bus as well as Skylake's mesh depend on the load. This reduces the electric power and slightly increases the energy efficiency of the processors when few cores are active.

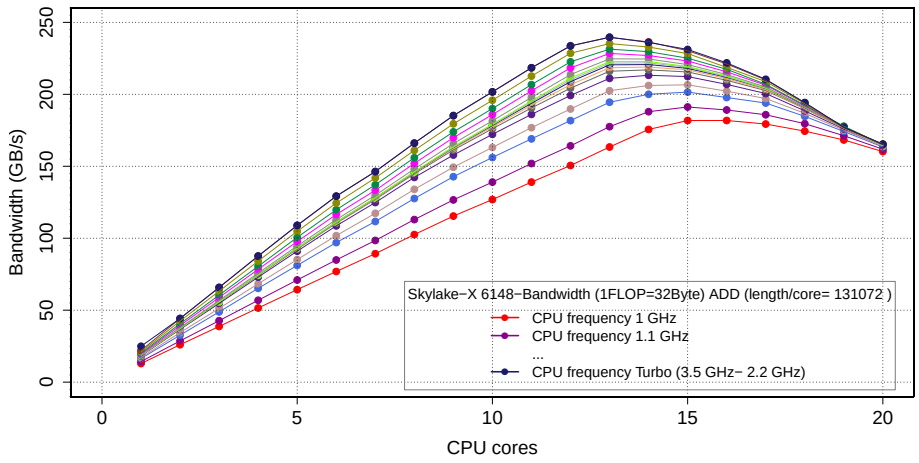


Figure C.6: Bandwidth of Kernel „Add“ on Skylake SP 6148 if the data for the arrays $a[]$,$b[]$ und $c[]$ are stored in the L3 cache and partly in the main memory if more than 12 threads are used.

C.8 Main Memory

While semiconductor technology and processor architecture have advanced very rapidly, main memory performance hasn't. Figure C.5 shows four memory channels connecting two IMCs of the processor with four main memory modules (DIMMs). The width of a memory channel is 8 bytes. The memory frequency is 2.133GHz (2133MHz). Thus, the theoretical memory bandwidth of the Haswell processor is $\sim$ 64 GB/s:

$$
\begin{aligned}
B_{MEM}[B/s] \quad &= N_{mem\_channel} \times W_{bus}[B] \times f_{clock\_mem}[MHz]; \\
\\
N_{mem\_channel} \quad &= 4 \textit{ (Number of memory channels);} \\
W_{bus} \quad &= 8 \text{ B } \textit{(Memory channel bus width);} \\
f_{clock\_mem} \quad &= 2133 \text{ MHz } \textit{(Memory interface frequency);}
\end{aligned}
\tag{C.3}
$$

The Skylake processors have six memory channels clocked with 2.666GHz, which increases the bandwidth up to $\sim$ 119 GB/s. If Skylake has an even better rate per core with 20 cores, from 24 cores upwards a Skylake core has less bandwidth than a Haswell core.

The Execution-Cache-Memory (ECM) performance model (Georg Hager, Jan Treibig, Johannes Habich, & Gerhard Wellein, 2012) makes very clear why the theoretical memory bandwidth couldn't be achieved in most cases. The model is described by equation C.4. Even though no prefetcher[9] is modeled, the model has a high prediction accuracy for performance of different computational kernels, particulary when many cores are active and higher CPU frequency

[9]Prefetching is used to „boost“ the performance. A prefetcher tries to preload the data for the future requests. The decision which data to load is made by looking up the history of the last several requsted addresses.

is set.

$$
\begin{aligned}
T_{ECM} &= \max(T_{OL}, T_{nOL} + T_{L1\leftrightarrow L2} + T_{L2\leftrightarrow L3} + T_{L3\leftrightarrow MEM});\\
\\
T_{ECM} &\text{ - } \textit{Execution time of a kernel;}\\
T_{OL} &\text{ - } \textit{Execution time in the core with the overlap for data transfer;}\\
T_{nOL} &\text{ - } \textit{Execution time in the core without the overlap for data transfer;}\\
T_{A\leftrightarrow B} &\text{ - } \textit{Execution time of the data transfer between the memory levels A and B;}
\end{aligned}
\tag{C.4}
$$

The next section C.9 shows the results of the kernel Add. During execution the data must be transfered between cores and memory.

It should be noted that SDRAM main memory architecture is optimized to delivery the data in the streams. As a rule, in HPC, the mapping between the memory address space and the memory modules (DIMMs) is set to interleaving mode. In this mode, a cache line is evenly distributed over the DIMMs and can be read in parallel from the memory channels. The successive cache lines can be read in streams without any noticeable interruption in boost mode. However, if the code is not vectorized, IMC may read the data piece by piece. Due to the high latencies of the memory access $\sim$ 100ns the bandwidth drops rapidly in such a case. You can find out more about the memory architecture and in particular about the SDRAM technology, for example, in the book „High-Bandwidth Memory Interface“ (Chulwoo Kim, Hyun-Woo Lee, & Junyoung Song, 2014).

C.9 Memory Wall

The kernel Add was executed on Skylake SP 6148 with 20 cores and the array length per core was 42949664. Accordingly, a core needs enought memory to store $\frac{3\times 42949664)\times 8[B]}{2^0} = 0.96$GB. The size of a Last Level Caches of Haswell and Skylake SP is significantly smaller. Since we know that 32 bytes must be transported between AVX registers and the main memory for one plus operation, we can calculate the average bandwidth of the kernel *Add* processing the data in main memory.

Figure C.7 on the left shows the bandwidth of *Add*, if the array length per core was set to 42949664. In this case, the data has to be stored in the main memory even if only one core is active. Each byte of data must be transported between the main memory and AVX registers across all cache levels, and the data transport between all hierarchical levels of memory can not overlap. The execution of the compute instruction, when the data is already in the register, can be executed in parallel (overlapped) with the further data transport. Figure C.7 on the rigth shows the energy

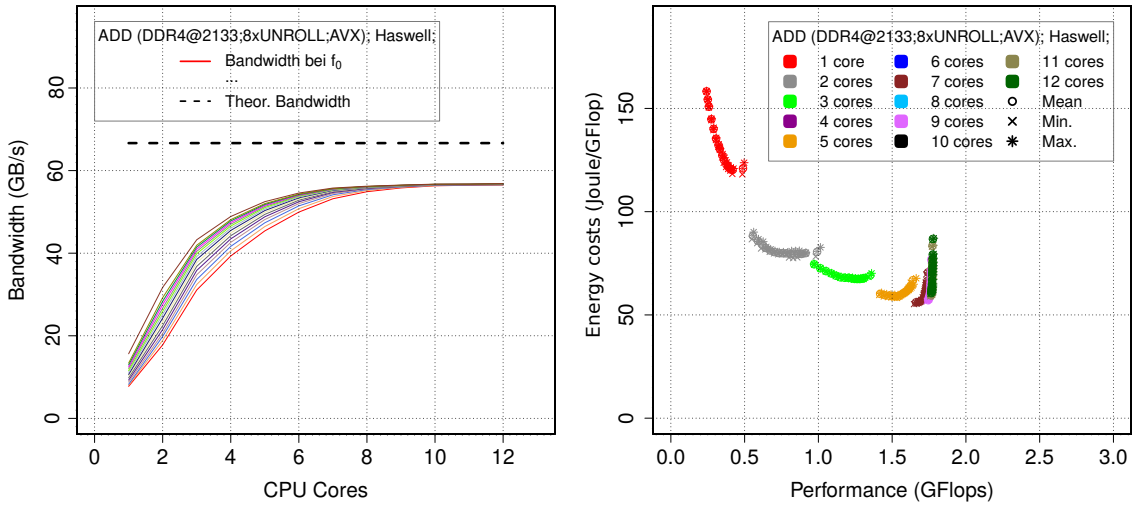


Figure C.7: Bandwidth of Kernel „Add “ on Haswell E5-5680v3 (left) and dependency between the energy costs and the performance (right) if the data for the arrays $a[]$,$b[]$ und $c[]$ are stored in the main memory

costs of the processor depending on the performance of the kernel *Add*-how many joules was spent by CPU to execute 10^9 floating point operations. This is equivalent to the transport of $10^9 \times 32$ bytes between the main memory and a

core. The energy costs are defined by equation C.5.

$$\frac{\mathrm{W}}{Flop/s} = \frac{\mathrm{Joule}}{Flop} \tag{C.5}$$

A you can see in figure C.7 on the right, increasing the CPU frequency increases the performance only if few cores are active. As soon as the memory bandwidth is fully utilized, there is no advantage to use more cores or to increase the CPU frequency. On the contrary, the power dissipation continues without increasing the performance. Equation C.6 shows a power approximation of the Haswell processor with 64 GB main memory during the execution of *Add* if data is stored in the main memory. The reason for different coefficients in the approximation is that during the kernel execution with one or two active cores, the frequency of the ring bus (see figure C.5) is noticeably lower compared to the execution with more cores. In addition, the power consumption of memory increases rapidly when more than two cores are active and more data is transferred from and to the memory modules.

Add on Haswell E5-2680v3 (data in RAM); Approximation for power of CPU+RAM(64GB):

$$\begin{aligned} P(f, p \in (1,2))_{ADD,RAM}[\mathrm{W}] &= (25.46588 + 8.26171 \times p) + (0.49423 + 2.39527 \times p) \times f^{1.91}; \\ P(f, p \in (3..,12))_{ADD,RAM}[\mathrm{W}] &= (65.52072 + 2.02131 \times p) + (2.02131 + 0.32525 \times p) \times f^{2.11}; \\ P &\text{ - Power of CPU and main memory;} \\ p &\text{ - Number of active cores;} \\ f &\text{ - CPU frequency;} \\ \epsilon_{rel} &< 0.077; \end{aligned} \tag{C.6}$$

For completeness, figure C.8 shows the bandwith of the Skylake SP processor for the same experiment.

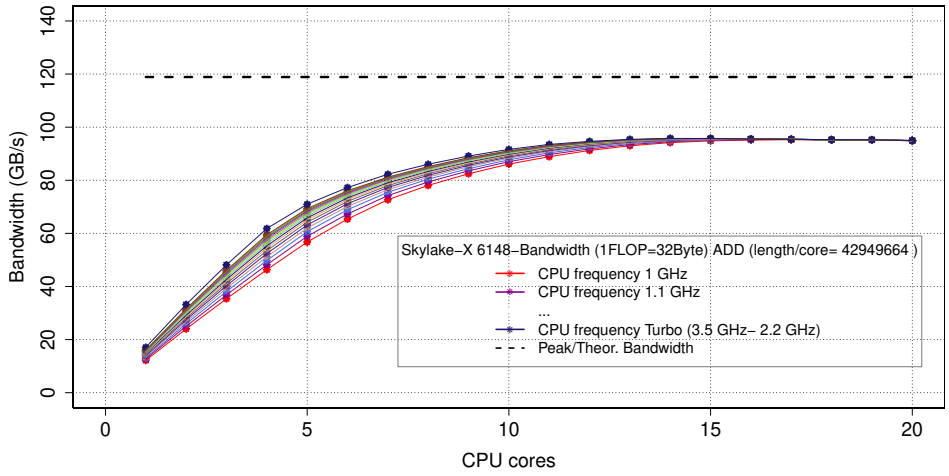


Figure C.8: Bandwidth of Kernel „Add “ on Skylake SP 6148 if the data for the arrays a[],b[] und c[] are stored in the main memory

According to the ECM model (see eqution C.4) and the results of the kernel ADD, a Haswell processor loses about 10% of its bandwidth compared to the theoretical bandwidth of the main memory and reaches 64 GB/s. A Skylake processor loses 20% of the theoretical bandwidth. Only because of the faster main memory and more memory channels, the kernel on Skylake can reach a bandwidth of 95 GB/s.