








Why High-Performance Modelling and Simulation for Big Data Applications Matters

Clemens Grelck¹ , Ewa Niewiadomska-Szynkiewicz² ,
Marco Aldinucci³ , Andrea Bracciali⁴ , and Elisabeth Larsson⁵ 

¹ University of Amsterdam, Amsterdam, Netherlands
`c.grelck@uva.nl`

² Warsaw University of Technology, Warsaw, Poland
`ens@ia.pw.edu.pl`

³ University of Turin, Turin, Italy
`marco.aldinucci@unito.it`

⁴ University of Stirling, Stirling, UK
`abb@cs.stir.ac.uk`

⁵ Uppsala University, Uppsala, Sweden
`elisabeth.larsson@it.uu.se`

Abstract. Modelling and Simulation (M&S) offer adequate abstractions to manage the complexity of analysing big data in scientific and engineering domains. Unfortunately, big data problems are often not easily amenable to efficient and effective use of High Performance Computing (HPC) facilities and technologies. Furthermore, M&S communities typically lack the detailed expertise required to exploit the full potential of HPC solutions while HPC specialists may not be fully aware of specific modelling and simulation requirements and applications.

The COST Action IC1406 *High-Performance Modelling and Simulation for Big Data Applications* has created a strategic framework to foster interaction between M&S experts from various application domains on the one hand and HPC experts on the other hand to develop effective solutions for big data applications. One of the tangible outcomes of the COST Action is a collection of case studies from various computing domains. Each case study brought together both HPC and M&S experts, giving witness of the effective cross-pollination facilitated by the COST Action.

In this introductory article we argue why joining forces between M&S and HPC communities is both timely in the big data era and crucial for success in many application domains. Moreover, we provide an overview on the state of the art in the various research areas concerned.

1 Introduction

The big data era poses a critically difficult challenge for high-performance computing (HPC): how to efficiently turn massively large and often unstructured or

semi-structured data first into valuable information and then into meaningful knowledge. HPC facilities and technologies are effectively required in a rapidly increasing number of data-intensive domains from life and physical sciences to socioeconomic systems. Thus, the big data era likewise offers striking opportunities for HPC to widen its scope and to strengthen its societal and economic impact.

High-performance Computing (HPC) and high throughput computing underpin the large scale processing of grand challenge problems with data-intensive requirements in order to enable complex applications in distinct scientific and technical fields such as high-energy physics, genomics, systems and synthetic biology, industrial automation, social and economic data analytics and medical informatics. This has led to a substantial improvement in the understanding of diverse domains ranging from the evolution of the physical world to human societies. Application performance in HPC systems is nowadays largely dominated by remote and local data movement overhead (network messages, memory and storage accesses). This poses new challenges to HPC modelling and programming languages, which should enhance data locality where possible and enable fast data transition where needed.

When investigating the behaviour and complexity of abstractions for large-scale big data systems, one employs a series of technologies that have their roots in well-funded large compute cluster environments. With the advent of hardware accelerators (GPU, FPGA), pay-by-use cloud services, and the increased performance of general-purpose processors, HPC has become an option for many scientific disciplines.

The COST Action IC1406 *High-Performance Modelling and Simulation for Big Data Applications* facilitates cross-pollination between the HPC community (both developers and users) and M&S disciplines for which the use of HPC facilities, technologies and methodologies still is a novel, if any, phenomenon. Data-intensive domains make the issue of efficiency particularly relevant for problems such as multi-dimensional and multi-level integration and model state explosion. Furthermore, these complex systems do not straightforwardly lend themselves to modular decomposition, a crucial prerequisite for parallelisation, and, hence, HPC support. They often require a significant amount of computational resources with data sets scattered across multiple sources and different geographical locations.

Modelling and Simulation (M&S) are widely considered essential tools in science and engineering to substantiate the prediction and analysis of complex systems and natural phenomena. Modelling has traditionally addressed complexity by raising the level of abstraction and aiming at an essential representation of the domain at hand. This has resulted in a complicated trade-off between accuracy and efficiency. That is to say, the properties of a system can be studied by reproducing (i.e., *simulating*) its behaviour through its abstract representation. Arguably, the context of the application level should be reconsidered. For instance, Monte Carlo simulations must be fed with input data, store intermediate results, and filter and merge output data in an adjusted and reliably robust

manner. Thus, M&S approaches are particularly affected by the data deluge phenomenon since they need to use large data sets to enhance resolution and scale and distribute and analyse data in the different phases of the simulation-analysis pipeline.

Both HPC and M&S are well established research areas each by themselves. However, a better integration of the two, aimed at applications from various domains, will bring substantial progress in addressing big data problems.

On the one hand, domain experts need HPC for simulation, modelling, and data analysis, but are often unaware of performance and parallelism exploitation pitfalls in their designs. On the other hand, designers of HPC development tools and systems primarily focus on absolute performance measures, by definition the *raison d'être* for HPC. However, MIPS, FLOPS, and speedups need not be the only measures. Domain-specific considerations may put some more or even almost all emphasis on other factors, such as usability, productivity, economic cost and time to solution. By further improving collaboration with domain experts HPC architects ought to be able to develop programming models and architectures better tailored to specific problems. Likewise, analysis and validation tools ought to be improved for a better understanding of HPC systems by domain experts.

The COST Action IC1406 *High-Performance Modelling and Simulation for Big Data Applications* is based on the idea that key aspects of HPC-enabled M&S must be jointly addressed by considering the needs and issues posed by the two communities together. When multidimensional, heterogeneous, massive data sets need to be analysed in a specific big data application domain, the methods required to suitably process the data are necessarily determined by the kind of data and analysis to be performed.

Consequently, the features of a programming language, library or execution machinery supporting the efficient implementation of the analysis should not be thought of as independent of the specific data and analysis themselves. Analogously, data characteristics must drive the design and implementation of data storage systems enabling efficient storage, access, and manipulation. Within this vision, the COST Action addresses the specific challenges of both M&S and HPC in a unified way.

The participants of the COST Action jointly work towards a unified framework for the systematic advancement of M&S and big data endeavours supported by leading HPC-enabled models and tools through a coordinated effort of HPC and M&S experts. The main objective is to create a long-lasting, sustainable, reference network of research links between the HPC community on the one hand and the multiple M&S research communities addressing big data problems on the other hand. Such links enable a novel and persisting collaboration framework across HPC and M&S communities, covering both academia and industry across Europe and beyond with a common agenda: turning huge amounts of raw data into useful knowledge.

The remainder of this paper is organised as follows: We first illustrate the background of our work and review the current state of the art in Sect. 2.

Following this introductory part, we have a closer look at the subjects relevant to the four working groups that make up the COST Action IC1406. We focus on *Enabling Infrastructures and Middleware for Big-Data Modelling and Simulation* in Sect. 3, *Parallel Programming Models for Big-Data Modelling and Simulation* in Sect. 4, *HPC-enabled Modelling and Simulation for Life Sciences* in Sect. 5, *HPC-enabled Modelling and Simulation for Socio-Economical and Physical Sciences* in Sect. 6, respectively. Last, but not least, we draw some conclusions in Sect. 7.

2 Background and State of the Art

High-Performance Computing is currently undergoing a major change with *exascale systems* expected for the early 2020s. They will be very different from today's HPC systems and pose a number of technological challenges, from energy consumption to the development of adequate programming models for millions of computing elements. Several current exascale research programmes, therefore, span a 10 to 20-year period. Major experiments depend on HPC for the analysis and interpretation of data and the simulation of models.

Modelling and Simulation have traditionally been used where the complexity of the problem makes more direct analytical approaches unsuitable or impossible. This is particularly true for big data problems where the support of HPC infrastructures and programming models is essential. The design and optimisation of HPC-enabled big data experiments and large scale HPC systems require the realistic description and modelling of the data access patterns, the data flow across the local and wide area networks and the scheduling and workload presented by hundreds of jobs running concurrently and exchanging very large amounts of data. Data-intensive (*big data*) HPC is arguably fundamental to address *grand-challenge* M&S problems.

In fact, several M&S approaches are based on discrete-event frameworks due to their efficiency and scalability. M&S have addressed problems such as scheduling in distributed, heterogeneous environments, economy-driven resource allocation, big data access in distributed environments and more generic HPC concurrent, distributed and cloud architecture. As described in the CERN Big Data HPC infrastructure, stochastic data traffic, management of virtual machines, and job allocation in data centers represent *grand-challenge* HPC-related problems, which require extensive use of M&S and HPC itself. Attempts to describe and analyse hardware, middleware and application co-design, an important development direction for HPC, have been made, but they currently appear too costly. The complexity can be reduced by means of coarse-grained models, which need precise measures of uncertainty and associated errors and statistical inference. Simulations have been run in this context for systems with one million cores. Recent trends aim to empower programmers to more easily control the hardware performance. Examples include the embedding of HPC facilities in standard OS distributions.

From an application perspective, HPC-enabled M&S has started to play a crucial role in a number of diverse knowledge domains. Preliminary proposals

with direct porting of existing techniques in HPC (e.g. in climate modelling) and further developments are being sought. In computational electromagnetics modelling problems with up to one billion variables have been addressed with both memory- and CPU-intensive algorithms, solving major longstanding problems. More structured approaches based on pattern-based parallel programming effectively cater for the design and development of parallel pipelines for M&S in systems biology and next generation sequencing [1,2], providing developers with portability across a variety of HPC platforms, like clusters of multi-cores [3,4] as well as cloud infrastructures [5].

However, HPC-enabled M&S has still not reached a fully satisfactory maturity, facing relevant problems in terms of computational efficiency and lack of generality and expressiveness when addressing data-intensive scenarios. The development of new complex HPC-enabled M&S applications requires collaborative efforts from researchers with different domain knowledge and expertise. Since most of these applications belong to domains within the life, social and physical sciences, their mainstream approaches are rooted in non-computational abstractions and they are typically not HPC-enabled.

Recent surveys of the use of HPC in life sciences illustrate possible new scenarios for knowledge extraction and the management of large-scale and heterogeneous data collections with numerous applications in medical informatics. Valuable big data diagnostic applications are being developed with the aim of improving diagnosis by integrating images and large multi-source data sets. These come at the extraordinary price of HPC-level infrastructure and suffer from the lack of standard protocols for big data representation and processing. Once computational results are obtained, large amounts of information need domain-specific validation. For instance, in bio-medical studies, wet-lab validation typically involves additional resource-intensive work that has to be geared towards a statistically significant distilled fragment of the computational results, suitable to confirm the bio-medical hypotheses and compatible with the available resources.

Big data is an emerging paradigm whose size and features are beyond the ability of the current M&S tools [6]. Datasets are heterogeneous, i.e., they are produced by different sources and are of a large size with high in/out rates. big data accessibility and the capability to efficiently bring and combine data together will be extremely valuable. Currently, many HPC-enabled M&S efforts have been proposed in several big data contexts, as diverse as performance evaluation and the management of HPC frameworks, research on blood anti-coagulants, the numerical evaluation of quantum dynamics, computational social network analysis (e.g. the relationship between Internet use and specific emotions, human obesity or happiness) and genomic sequence discovery. Some approaches have been successful, leading to potential industrial impact and supporting experiments that generate petabytes of data, like those performed at CERN for instance.

Furthermore, there are a growing number of new implementations of memory-demanding applications that have not yet been adapted for HPC environments, mainly because of limited communication between field experts and those with

suitable skills for the parallel implementation of data-intensive applications. Therefore, another natural objective of our work is to intelligently transfer the heterogeneous workflows in M&S to HPC, which will boost those scientific fields that are essential for both M&S and HPC societies [7]. Benefits will be reciprocal. M&S experts are supported in their investigations by properly-enabled HPC frameworks, currently sought but missing. HPC architects in turn obtain access to a wealth of application domains by means of which they will better understand the specific requirements of HPC in the big data era. Among others, we aim at the design of improved data-center oriented programming models and frameworks for HPC-enabled M&S.

3 Enabling Infrastructures and Middleware for Big-Data Modelling and Simulation

From the inception of the Internet, one has witnessed an explosive growth in the volume, speed and variety of electronic data created on a daily basis. Raw data currently originates from numerous sources including mobile devices, sensors, instruments (e.g., CERN LHC, MR scanners, etc.), computer files, Internet of Things, governmental/open data archives, system software logs, social networks, commercial datasets, etc. The challenge is how to collect, integrate and store, with less hardware and software requirements, tremendous data sets generated from distributed sources.

The so-called big data problem requires the continuous improvement of servers, storage, and the whole network infrastructure in order to enable the efficient analysis and interpretation of data through on-hand data management applications, e.g. agent-based solutions in Agent Component in Oracle Data Integrator (ODI). The main challenge in big data modelling and simulation is to define a complete framework which includes intelligent management and communication, data fusion, mapping algorithms and protocols. The programming abstractions and data manipulation techniques must, therefore, be designed for (a) the seamless implementation of application solutions with efficient levels of virtualisation of computational resources (communications, storage, and servers) and (b) the effective normalisation and merging of data with dissimilar types into a consistent format (wide class of data services).

Energy-awareness is an important aspect of big data computing and simulation. The goal is to reduce the gap between the capacity provided by distributed computing environments and application requirements, especially during low workload periods. Various efforts are undertaken to develop energy efficient task scheduling and balancing of loads [8–11] and frequency scaling techniques [12–14].

Infrastructure and Middleware for Big Data Processing. Numerous algorithms, computing infrastructures and middleware for HPC and big data processing have been developed during previous decades [15, 16]. In general, current middleware

for parallel computing focuses on providing powerful mechanisms for managing communication between processors and environments for parallel machines and computer networks. High Performance Fortran (HPF), OpenMP, OpenACC, PVM, and MPI were designed to support communications for scalable applications. The application paradigms were developed to perform calculations on shared memory machines and clusters of machines with distributed memory. However, the easy access to information offered by the internet led to a new idea: extending the connection between computers, so that distributed resources, including computing power, storage, applications, etc. could be accessed as easily as information on web pages. The idea was implemented in many forms, but lately it has grown into three main computing environments: computing clusters, grids and clouds. A survey of software tools for supporting cluster, grid and cloud computing is provided in [15, 17, 18]. Examples of commonly known kernels for cluster computing are MOSIX [19], OpenSSI [20] and Kerrighed [21]. Uniform interfaces to computing resources in grids and toolkits for building grids (e.g. UNICORE [22] or Globus Toolkit [23]) are described in literature [24, 25]. Cloud computing infrastructures consisting of services delivered through common centers and built on servers are discussed in [18].

An alternative to supercomputers and computing clusters—the (General-Purpose) Graphics Processing Unit (GPGPU)—is widely used in HPC simulation [26]. Using both CPU and GPU through CUDA or OpenCL many real-world applications can rather easily be implemented and run significantly faster than on multi-processor or multi-core systems.

Tools and Platforms for Big Data Processing. Job scheduling, load balancing and management play a crucial role in HPC and big data simulation [27, 28]. TORQUE [29] is a distributed resource manager providing control over batch jobs and distributed compute nodes. Slurm [30] is an open source, fault-tolerant and highly scalable cluster management and job scheduling system for large and small Linux clusters. MapReduce [31] is a framework that simplifies the processing of massive volumes of data through using two subsequent functions, i.e. the Map function that sorts and splits the input data and the Reduce function that is responsible for processing the intermediate output data. Resource management and job scheduling technology like YARN [32] allows multiple data processing engines such as batch processing, real-time streaming, interactive SQL and data science to handle data stored in a single platform. The Apache Hadoop software library [32] supports the distributed, scalable batch processing of large data sets across clusters of computers using a simple programming model. The power of the Hadoop platform is based on the Hadoop Distributed File System (HDFS), a distributed and scalable non-relational database HBase, MapReduce, YARN and many other open source projects. Some of the best-known include: Spark, Pig, Hive, JAQL, Sqoop, Oozie, Mahout, etc. Apache Spark [33], a unified engine for big data processing, provides an alternative to MapReduce that enables workloads to execute in memory, instead of on disk. Thus Spark avoids the resource-intensive disk operations that MapReduce requires. It processes data in RAM utilizing a data model based on the Resilient Distributed Dataset (RDD)

abstraction. Apache Storm [34] is a scalable, rapid, fault-tolerant platform for distributed computing that has the advantage of handling real time data processing downloaded from synchronous and asynchronous systems. Apache Flink [35] can be used to batch and stream processing, event-time processing and stateful computations, well-suited for discrete-event simulation.

Platforms for Big Data Visualisation and Machine Learning. Numerous tools for big data analysis, visualisation and machine learning have been made available. RapidMiner Studio [36], Orange [37] and Weka [38] belong to this group. New software applications have been developed for browsing, visualizing, interpreting and analyzing large-scale sequencing data. Some of them have been designed specifically for visualisation of genome sequence assemblies, including Tablet [39]. Other tools, such as BamView [40] have been developed specifically to visualise mapped read alignment data in the context of the reference sequence. Artemis [41] is a freely available integrated platform for visualisation and analysis of large-scale experimental data. The survey of platforms and packages for social network analysis, simulation and visualisation that have wide applications including biology, finance and sociology is presented in [42].

Frameworks for Big Data Systems Simulation. Another issue is concerned with large-scale systems simulation. The combination of efficient and reliable simulation software and purpose-built hardware optimized for simulation workloads is crucial to fully exploit the value of simulation and big data. Synchronous and asynchronous distributed simulation have been one of the options that could improve the scalability of a simulator both in term of application size and execution speed, enabling large scale systems to be simulated in real time [43, 44]. ScalaTion [45] provides comprehensive support for discrete-event simulation and big data analytics. A software framework for federated simulation of WSN and mobile ad-hoc networks is presented in [46]. The paper [47] reviews several large-scale military simulations and describes two frameworks for data management based on layered and service oriented architectures. GPU-based simulation platforms are mainly dedicated to massive data processing, e.g. high-performance neural network simulators [48, 49], simulation of P systems [50], large-scale volume of data simulation and visualisation [51].

Numerous software platforms have been designed to simulate large-scale distributed data centers and computer networks. JADE [52] is the heterogeneous multiprocessor design simulation environment that allows to simulate network-on-chips, inter-chip networks and intra-rack networks using optical and electrical interconnects. SimGrid [53] can be used to simulate grids, clouds, HPC or P2P systems and evaluate heuristics or prototype applications. CloudSim [54] is one of the most popular open source framework for modeling and simulation of cloud computing infrastructures and services. Multi2Sim [55] is a software platform for simulation of new CPU and GPU technologies.

4 Parallel Programming Models for Big-Data Modelling and Simulation

A core challenge in modelling and simulation is the need to combine software expertise and domain expertise. Even starting from well-defined mathematical models, manual coding is inevitable. When parallel or distributed computing is required, the coding becomes much harder. This may impair time-to-solution, performance, and performance portability across different platforms. These problems have been traditionally addressed by trying to lift software design and development to a higher level of abstraction.

In the domain-specific languages (DSL) approach abstractions aim to provide domain experts with programming primitives that match specific concepts in their domain. Performance and portability issues are ideally moved (with various degrees of effectiveness) to development tools. Examples include Verilog and VHDL hardware description languages, MATLAB for matrix programming, Mathematica and Maxima for symbolic mathematics, etc.

In general-purpose approaches such as model-driven engineering (MDE), general-purpose programming concepts are abstracted into high-level constructs enforcing extra-functional features by design, e.g. compositionality, portability, parallelisability. In this regard, the number and the quality of programming models enabling the high-level management of parallelism have steadily increased and, in some cases, these approaches have become mainstream for a range of HPC, data-intensive and big data workloads: streaming (e.g. Storm [56] and Spark [57]), structured parallel programming and MapReduce [58] (e.g. Hadoop [59], Intel TBB [60], OpenMP [61], MPI [62]), SIMD (e.g. OpenACC [63]). This list can be extended by various academic approaches, including ones proposed and advocated by members of the COST Action (e.g. FastFlow [64, 65]), SkePU [66], SAC [67], S-Net [68]).

A sensible result achieved by the working group on *Parallel Programming Models for Big-Data Modelling and Simulation* has been the assessment of the state of the art. A selection of the mainstream approaches in this area are presented in Sect. 4.1, namely Google MapReduce, Apache Spark, Apache Flink, Apache Storm and Apache Beam. In Sect. 4.2 we describe a systematic mapping study, aimed to capture and categorise non-mainstream DSLs.

4.1 Languages and Frameworks for Big Data Analysis

Boosted by big data popularity new languages and frameworks for data analytics are appearing at an increasing pace. Each of them introduces its own concepts and terminology and advocates a (real or alleged) superiority in terms of performance or expressiveness against its predecessors. In this hype, for a user approaching big data analytics (even an educated computer scientist) it is increasingly difficult to retain a clear picture of the programming model underneath these tools and the expressiveness they provide to solve some user-defined problem.

To provide some order in the world of big data processing, a toolkit of models to identify their common features is introduced, starting from data layout.

Data-processing applications are divided into *batch* vs. *stream* processing. Batch programs process one or more *finite* datasets to produce a resulting finite output dataset, whereas stream programs process possibly unbounded sequences of data, called *streams*, in an incremental manner. Operations over streams may also have to respect a total data ordering, for instance to represent time ordering.

The comparison of different languages for big data analytics in terms of the expressiveness of their programming models is a non-trivial exercise. A formalised approach requires to map them onto an unifying (and lower-level) computation model, i.e. the *Dataflow model* [69]. As shown in [70], it is able to capture the distinctive features of all frameworks at all levels of abstraction, from the user-level API to the execution model. In the Dataflow model, applications as a directed graph of actors. In its “modern” macro-data flow version [71], it naturally models independent (thus parallelizable) kernels starting from a graph of true data dependencies, where a kernel’s execution is triggered by data availability.

The Dataflow model is expressive enough to describe batch, micro-batch and streaming models that are implemented in most tools for big data processing. Also, the Dataflow model helps in maturing the awareness that many big data analytics tools share almost the same base concepts, differing mostly in their implementation choices. For a complete description of the Dataflow model we refer back to [6, 70], where the main features of mainstream languages are presented.

Google MapReduce. Google can be considered the pioneer of big data processing, as the publication of the MapReduce framework paper [72] made this model mainstream. Based on the *map* and *reduce* functions, commonly used in parallel and functional programming [73], MapReduce provides a native key-value model and built-in sorting facilities. These made MapReduce successful for several big data analytics scenarios.

A MapReduce program is built on the following user-defined functions:

1. a **map** function that is independently applied to each item from an input key-value dataset to produce an *intermediate* key-value dataset;
2. a **reduce** function that combines all the intermediate values associated with each key (together with the key itself) into lists of reduced values (one per key);
3. a **partitioner** function that is used while *sorting* the intermediate dataset (i.e., before being reduced), so that the order over the key space is respected within each partition identified by the partitioner.

Parallel Execution. A simple form of data parallelism can be exploited on the flat-map side, by partitioning the input collection into n chunks and having n executors process a chunk. In Dataflow terms this corresponds to a graph with n actors, each processing a token that represents a chunk. Each flat-map executor

emits R (i.e. the number of intermediate partitions) chunks, each containing the intermediate key-value pairs mapped to a given partition.

The intermediate chunks are processed by R reduce executors. Each executor:

1. receives n chunks (one from each flat-map executor);
2. merges the chunks into an intermediate partition and partially sorts it based on keys, as discussed above;
3. performs the reduction on a per-key basis.

Finally, a downstream collector gathers R tokens from the reduce executors and merges them into the final result.

A key aspect in the depicted parallelisation is the *shuffle* phase in which data is distributed between flat-map and reduce operators, according to an *all-to-all* communication schema. This poses severe challenges from the implementation perspective.

Run-time Support. The most widespread implementation (i.e. *Hadoop*), is based on a *Master-Workers* approach, in which the master retains the control over the global state of the computation and informs the workers about the tasks to execute.

A cornerstone of Hadoop is its distributed file system (HDFS), which is used to exchange data among workers, in particular upon shuffling. As a key feature HDFS exposes the *locality* for stored data, thus enabling the principle of moving the computation towards the data and to minimise communication. However, disk-based communication leads to performance problems when dealing with iterative computations, such as machine learning algorithms [74].

Apache Spark. *Apache Spark* [75] was proposed to overcome some limitations in Google’s MapReduce. Instead of a fixed processing schema, Spark allows datasets to be processed by means of arbitrarily composed primitives, constructing a directed acyclic graph (DAG). Moreover, instead of exclusively relying on disks for communicating data among the processing units, in-memory caching is exploited to boost performance, in particular for iterative processing.

Parallel Execution and Runtime Support. From the application DAG, Spark infers a parallel execution dataflow, in which many parallel instances of actors are created for each function and independent actors are grouped into *stages*. Due to the Spark batch-oriented implementation, each stage that depends on some previous stages has to wait for their completion before execution commences, equivalent to the classical Bulk Synchronous Parallelism (BSP) approach. Thus, a computation proceeds in a series of global *supersteps*, each consisting of:

1. concurrent computation, in which each actor processes its own partition;
2. communication, where actors exchange data between themselves if necessary (the *shuffle* phase);
3. barrier synchronization, where actors wait until all other actors have reached the same barrier.

Similar to the MapReduce implementation, Spark’s execution model relies on the master-Workers model: a cluster manager (e.g. YARN) manages resources and supervises the execution of the program. It manages application scheduling to worker nodes, which execute the application logic (the DAG) that has been serialized and sent by the master.

Apache Flink. *Apache Flink* [76] is similar to Spark, in particular from the API standpoint. However, Flink is based on *streaming* as a primary concept, rather than a mere linguistic extension on top of batch processing (as Spark). With the exception of iterative processing, stream parallelism is exploited to avoid expensive synchronizations among successive phases when executing both batch and stream programs.

Parallel Execution and Runtime Support. Flink transforms a JobGraph into an ExecutionGraph, in which the JobVertex contains ExecutionVerteces (actors), one per parallel sub-task. A key difference compared to the Spark execution graph is that, apart from iterative processing (that is still executed under BSP), there is no barrier among actors or verteces. Instead, there is effective pipelining.

Also Flink’s execution model relies on the master-workers model: a deployment has at least one job manager process that receives Flink jobs and coordinates checkpointing and recovery. The job manager also schedules work across the task manager processes (i.e. the workers), which usually reside on separate machines and in turn execute the code.

Apache Storm. *Apache Storm* [56, 77] is a framework that exclusively targets stream processing. It is perhaps the first widely used large-scale stream processing framework in the open source world. Whereas Spark and Flink are based on a declarative data processing model, i.e., they provide as building blocks data collections and operations on those collections, Storm, in contrast, is based on a “topological” approach in that it provides an API to explicitly build graphs.

Parallel Execution and Runtime Support. At execution level, each actor is replicated to increase the inter-actor parallelism, and each group of replicas corresponds to the Bolt/Spout in the semantics Dataflow. Each of these actors represents independent data-parallel tasks, on which pipeline parallelism is exploited. Eventually, tasks are executed by a master-workers engine, as in the previously discussed frameworks.

Google Cloud Platform and Apache Beam. Google Dataflow SDK [78] is part of the Google Cloud Platform. Google Dataflow supports a simplified pipeline development via Java and Python APIs in the Apache Beam SDK, which provides a set of windowing and session analysis primitives as well as an ecosystem of source and sink connectors. Apache Beam allows the user to create pipelines that are executed by one of Beam’s supported distributed processing back-ends, which are called *runners*. Currently, they include, among others, Apache Flink, Apache Spark and Google Cloud Dataflow.

Parallel Execution and Runtime Support. The bounded (or unbounded) nature of a PCollection also affects how data is processed. Bounded PCollections can be processed using batch jobs, that might read the entire data set once and perform processing as a finite job. Unbounded PCollections must be processed using streaming jobs—as the entire collection will never be available for processing at any one time—and bounded subcollections can be obtained through logical finite size windows.

As mentioned, Beam relies on the *runner* specified by the user. When executed, an entity called *Beam Pipeline Runner* (related to execution back-end) translates the data processing pipeline into the API compatible with the selected distributed processing back-end. Hence, it creates an execution graph from the Pipeline, including all the Transforms and processing functions. That graph is then executed using the appropriate distributed processing back-end, becoming an asynchronous job/process on that back-end. Thus, the final parallel execution graph is generated by the back-end.

The parallel execution data flow is similar to the one in Spark and Flink. Parallelism is expressed in terms of data parallelism in Transforms (e.g. ParDo function) and inter-actor parallelism on independent Transforms. In Beam's nomenclature this graph is called the Execution Graph. Similar to Flink pipeline parallelism is exploited among successive actors.

4.2 The Systematic Mapping Study on Parallel Programming Models for Big-Data Modelling and Simulation

A major challenge undertaken within the working group on *Parallel Programming Models for Big-Data Modelling and Simulation* was that of trying to understand and classify the state of the art in this area and to better understand the lines of future development. In order to minimize the bias, given that many Action participants actively design programming models and tools, the working group refined and adopted a systematic methodology to study the state of the art, called systematic mapping study (SMS). The mapping study focused on the main paradigms and properties of programming languages used in high-performance computing for gig data processing.

The SMS started from the definition of a workflow based on the methodology proposed in [79] that is organised in five successive steps:

Research Questions aiming at formulating the research questions the SMS should answer;

Search of Primary Studies aiming at detecting the largest number of primary articles related to the proposed research questions;

Selection of Primary Studies aiming at sieving false positive by a human-driven abstract inspection;

Quality Assessment aiming at validating the fitness of the articles against the aims of the SMS;

Data Extraction and Synthesis which aims at answering each research question for all selected articles.

Specifically, the SMS focused on domain-specific languages and explicitly excluded general-purpose languages, such as C, C++, OpenMP, Fortran, Java, Python, Scala, etc., combined with parallel exploitation libraries, such as MPI.

Quantitatively, in the SMS, the initial literature search resulted in 420 articles; 152 articles were retained for final review after the evaluation of initial search results by domain experts. Results of our mapping study indicate, for instance, that the majority of the used HPC languages in the context of big data are text-based general-purpose programming languages and target the end-user community. To evaluate the outcome of the mapping study, we developed a questionnaire and collected the opinions of domain experts. A comparison of mapping study outcome with opinions of domain experts reveals that the key features of HPC programming languages for big data are portability, performance and usability. We identified the language learning curve and interoperability as the key issues that need more attention in future research.

5 HPC-Enabled Modelling and Simulation for Life Sciences

Life Sciences typically deal with and generate large amounts of data, e.g., the flux of terabytes about genes and their expression produced by state of the art sequencing and microarray equipment, or data relating to the dynamics of cell biochemistry or organ functionality. Some modelling and simulation techniques require the investigation of large numbers of different (virtual) experiments, e.g., those addressing probabilistic, noise and robustness aspects [80–84], or based on statistical approaches. Curation and mining of large, typically multimedia, medical datasets for therapeutic and analytics purposes, are computationally expensive. Recent and future developments, such as personalised medicine need to integrate a mix of genomics, Systems and Synthetic Biology and medical information in a systemic description of a single individual. A surge of large-scale computational needs in these areas spans from the BBMRI (Biobanking and Biomolecular Resources Research Infrastructure) and the flagship effort Human Brain Project, which targets simulating the behaviour of a human brain, to FP7 projects like PD-HUMMODEL and TRANSFORM. In fact, this COST Action integrates well with the goals of the ESFRI Roadmap, promoted by the EC. Requirements go from pure computational efficiency, to large data file management and storage capabilities and vast memory-bound computational power.

This section focuses on the much-needed integration of HPC architects and Life Sciences modellers, with the goal of letting them develop and diffuse a coordinated, mature and productive use of HPC facilities. In order to bridge these two communities, some big data problems, applications and modelling techniques in the broad context of live sciences are discussed. We will consider approaches for modelling healthcare and diseases as well as problems in systems and synthetic biology. We will survey some themes on genomics and metabolic networks, then discuss efficient modelling and learning techniques, and finally consider also the modelling of the management of healthcare.

Healthcare and Disease Modelling. Understanding disease complexity is the definite scientific challenge of the 21st century medicine [85,86]. Using computational models is the path to a technological medical revolution, where modelling will have a truly catalytic effect in biomedical big data by bridging the large body of knowledge produced by next generation genomic data with the clinical quantities and the functional observable (for instance through self monitor and implantable sensor devices). Biomedicine is essentially a big data field, and modelling is superior to the data-mining correlative approach in transforming data into knowledge.

Taking into account only the DNA sequencing data, its rate of accumulation is much larger than other major generators of big data, such as astronomy, YouTube and Twitter. Recent estimates show that the total amount of sequencing data produced is doubling approximately every seven months. The growth is driven by three main factors:

1. Biomedicine is heavily interdisciplinary and e-Healthcare requires physicians, bioinformaticians, computer scientists and engineers to team up. Although they continuously produce results that are underutilised in medical practice, such interdisciplinarity generates the need for large-scale data integration. Areas such as systems medicine, clinical informatics, systems biology and bioinformatics have large overlaps with classical fields of medicine, and extensively use biological information and computational methods to infer new knowledge towards understanding disease mechanism and diagnosis.
2. Many acute and chronic diseases originate as network diseases. A patient's condition is characterised by multiple, complex and interrelated conditions, disorders or diseases [87,88]. A state of health can be defined as the capacity of absorbing accidents and showing metabolic flexibility, and is altered by infections and ageing, that cause comorbidities to emerge. Therefore, a good-quality stratification of a patient requires lots of information. The bridge between the characterisation of a disease mechanism and the stratification of a patient would require a data-driven computational model. Current successful approaches focus on resource-intensive hybrid modelling approaches including cellular automata, (stochastic) differential equations and agent-based models. The more effective the diagnostic and prognostic markers are, the less information will be needed to correctly stratify a patient. This aspect makes precision medicine highly computation-resource intensive. In particular, complex disease management is mostly based on electronic health records collection and analysis, which are expensive processes. Analyses are presented in a rather empirical and sometimes simplistic way, completely missing the opportunity of uncovering patterns of predictive relationships and meaningful profiles. Our chances to make the data the drivers of paths to cures for many complex diseases depends in a good percentage on extracting evidences from large-scale electronic records comparison and on models of disease trajectories. The medical approach to comorbidities represents an impressive computational challenge, mainly because of data synergies leading to the integration of heterogeneous sources of information, the definition of deep phenotyping

- and markers re-modulation; the establishment of clinical decision support systems. Computational model development is further complicated by aspects of geo-differentiation and ethnic balance, protocols for sharing of digital information, interoperability between different record types (structured and non-structured) to optimize the process of decision making in an actionable way.
3. A third important factor is the multi-scale nature of the biomedical information. The genome sequence is only the first level of understanding of the human biology. Bioinformatics data resources should be much more populated with longitudinal information gathered at intracellular, cell, intercellular and tissue levels. The longitudinal sampling could happen for important clinical events, such as hospitalisation or routinely perhaps a few times from uterine to elderly age. At the bioinformatics level, genome wide information of all the different levels of biological information will be integrated and this may include: Genomic sequence variations (haplotypes), levels of gene functioning for different tissues and conditions (circadian and longitudinal data) (gene expression), Epigenetic changes for different tissues (methylations and histonic modifications), information on chromatin conformation for different cell types and conditions (FISH, HI-C, 5C, microscopy), protein and metabolites abundances for different cell types and conditions, protein-protein interaction variations (longitudinal data).

For instance by using the Next Generation Sequencing technology approaches cancer clones, subtypes and metastasis could be appropriately traced. Microbiome data (number, type and multi Omics) for different part of the body and different conditions. Furthermore, gut microbiome could be regularly sampled, monitoring the diets and nutritional shifts. This could be of great importance for epigenetic data, which shows alteration with ageing, inflammatory diseases, obesity, cardiovascular and neurodegenerative diseases. Gene expression may vary in relation to the circadian cycle or ageing. Sampling may focus on determining the actual level of inflammation that is related to ageing rate (inflammaging). Large number of high-resolution images of the different parts of the patient's body such as MRI, PET, CT scan, including intravital microscopy techniques, can be used. The images will tend to be progressively enriched with genomics and proteomics data information. A disease first emerges as a dysfunction at the nucleus level, then metabolic and signalling, cell level and is then translated at the tissue level due to a change in the cell response. The tissue level is central to stem cells organisation in maintaining the mechanical properties of the tissue: the current thinking is that the dominant effect of reduced stem cell activity and failing tissue maintenance is due to changes in the niches that support and control stem cell activity. Therefore, tissue modelling can be thought as the missing link between basic research and clinical practice and will require a conceptual framework for an efficient multi-scale analysis between the cell and tissue levels. The cell level will be represented with agent-based or ODE models that will be specifically developed to handle millions of single cells. The tissue level will be represented using image-based finite element modelling (partial differential equation, PDE).

An important example is the bone system which is also related to the immune and endocrine systems. The osteocytes in the bone act as sensitive mechanosensors so they react to microdamages that alter the tension; with their flattened morphology and long processes, they form a sensory network which allows the detection of abnormal strain situations such as generated by microcracks. Normal locomotion is thought to cause microdamage to bone material and, thus, stimulate osteoclasts and osteoblasts to remove and then replace damaged tissue. They can be modelled as agents driven by signals and could reflect concentrations and velocities. Osteocytes are connected to one another and to surface osteoblasts via gap junctions. In general, mechanical forces are experienced by many osteoprogenitor cells which are present in the bone marrow and in the soft mesenchymal tissues subjected to mechanical strain. Dependant on the magnitude of mechanical stress osteoprogenitors differentiate or transdifferentiate into osteoblastlike cells that express characteristic proteins and can form bone matrix. Under physiological mechanical stimuli osteocytes prevent bone resorption by changing the RANKL/osteoprotegerin (OPG) ratio. By communicating these signals to bone lining cells (the second terminally differentiated osteoblast cell type) or secrete factors that recruit osteoclasts, osteocytes initiate the repair of damaged bone. The functional behaviour of bone tissues is primarily described in term of physical quantities such as pressures and forces to reflect deformation, loading, stress, strain, etc. Such quantities, are usually considered to vary across space and time, in a continuous fashion, and can be thus represented using fields, and systems of partial differential equations (PDE). The transition between a continuous representation and a discrete representation makes the coupling of the models across the cell-tissue scale particularly difficult. Conventional homogenisation approaches, frequently used as relation models to link to component models defined at different scales, are computationally resource demanding [89–92].

Modelling Problems in System and Synthetic Biology. Systems Biology approaches and methodologies are also very interesting in Synthetic Biology pipelines: in semi-synthetic minimal cells, for instance, liposomes are synthesized with some metabolic networks entrapped inside [93]. These devices, called protocells, share some properties in common with real biological cells, and can perform some biological action [94]. In wet-lab the problem is which metabolic component to choose, among the several different ones that can perform the same biological action. A combinatorial experimental approach is not affordable, since it requires a lot of time, budget and lab resources. A computational approach, instead, is very useful, as it can score the different hypotheses about the protocell to synthesize, sorting out the best theoretically performing. Along this research line, several papers have been published, based on computer simulation of the metabolic networks entrapped in the protocells [95], to understand the solute distribution and enrichments processes [96,97] and the energetic balance of complex biological processes like DNA transcription and RNA translation [98].

Genomics. In recent years, thanks to faster and cheaper sequencing machines, a huge amount of whole genomic sequences within the same population has become available (e.g. [99]). Modern genomes analyses workflows have thus to face new challenges to perform functional annotations and comparative analysis, as there is no longer just *a* reference genome, but rather many of them that can have to be used all together as a control sequence. A collection of genomic sequences to be analysed jointly, or to be jointly used as a reference, is called *pangenome* [100]. The reference genome is a representative example of the whole genomic sequence of a species, acting as a control sequence against which fragments of a newly sequenced individual are mapped to be located, or against which another whole genome is compared. A single well annotated reference genome was - and mostly still is - traditionally used as a control sequence, as it could provide a good approximation of any individual genome. However, in loci where polymorphic variations occur (a *polymorphism* is a genetic variation of an individual or a population), such mappings and comparisons are likely to fail: this is where a multiple reference genome—a reference pan-genome—would be a better control [101].

In the data structure literature, several different compressed representations have been considered for sets of similar texts [102, 103], as well as algorithmic methods for their investigation [104]. We present here a natural representation of pan-genomes (whole genomes or their fragments): *elastic-degenerate texts*. An *elastic-degenerate text* (*ED-text*) is a sequence compactly representing a multiple alignment of several closely-related sequences: substrings that match exactly are collapsed, while those in positions where the sequences differ (by means of substitutions, insertions, and deletions of substrings) are called *degenerate*, and therein all possible variants observed at that location are listed [105]. Actually, *ED-texts* correspond exactly to the Variant Call Format (.vcf), the *standard* for files storing genomic variations [106]. As an example, consider the following three closely-related sequences, where their similarity is highlighted by their alignment, and where the symbol ‘-’ represents a deletion:

```
CAATGTGTGAC
CAGTCAAT-AC
C--T-ACTGAC
```

These sequences can be compacted into the single *ED-text*:

$$\tilde{T} = C \cdot \left\{ \begin{array}{c} \text{AA} \\ \text{AG} \\ \varepsilon \end{array} \right\} \cdot T \cdot \left\{ \begin{array}{c} \text{GTG} \\ \text{CAA} \\ \text{AC} \end{array} \right\} \cdot T \cdot \left\{ \begin{array}{c} \text{G} \\ \varepsilon \end{array} \right\} \cdot \text{AC} \quad (1)$$

where ε is the empty string. The *length* n of \tilde{T} is the total number of segments, and its *size* N is the total number of letters, that all belong to an alphabet Σ . Due to biotechnologies limitations, sequencing (that is, giving as input the *in vitro* DNA and getting out an *in silico* text file) can only be done on a genome fragment of limited size. For this reason, before the sequencing process, a genome is actually broken into many fragments of such limited size and then, whenever a

reference is available, the resulting *in silico* fragments (named *reads*) are mapped onto it. This mapping is a critical step and there is an ample literature aiming at making as efficient as possible. When the reference is an *ED*-text, the reads mapping problem translates into the problem of finding all matches of a deterministic pattern P (that is, $P \in \Sigma^*$) in text \hat{T} . We call this the *EDSM* problem.

In [107] the problem has been solved for the simplest case of *ED*-text, in which a degenerate segment can only contain single letters. In [108] the problem has been efficiently solved for the more general case of *ED*-texts introducing (i) an algorithmic framework that has been conserved also by more recent papers, and (ii) adding a very fast bit-vector based version of the same algorithm that requires the pattern to have size no longer than the machine word. In [109] the algorithmic framework has been extended to find *approximate* occurrences of P , under both the Hamming and the edit distance model. In other words, occurrences of P are detected allowing up to a given amount of mismatches (Hamming distance model), or even insertions or deletions (edit distance model). In [110] the bit-vector algorithm of [108] has been extended to work with a *collection* of patterns P_1, P_2, \dots, P_h rather than a single string P , and in [111] a step of the algorithm presented in [108] has been improved by a factor $\sqrt{|P|}$. Another natural problem that arises is the comparison of two *ED*-texts and, in particular, whether the sets of strings they actually represent has a non empty intersection. This problem has been efficiently solved in [112] with a linear time algorithm for the case of non-elastic *D*-texts (a degenerate segment can only contain strings of the same size).

Once that a set of DNA fragments of an individual have been aligned, haplotype phasing is an important problem in the analysis of genomics information. It consists of determining which one of the possible alleles (alternative forms of a gene) each fragment comes from. Haplotype information is relevant to gene regulation, epigenetics, genome-wide association studies, evolutionary and population studies, and the study of mutations. Haplotyping is currently addressed as an optimisation problem aiming at solutions that minimise, for instance, error correction costs, where costs are a measure of the confidence in the accuracy of the information acquired from DNA sequencing. Solutions have typically an exponential computational complexity. WHATSHAP [113] is a framework returning exact solutions to the problem of haplotyping which moves computational complexity from DNA fragment length to fragment overlap, i.e., coverage, and is hence of particular interest when considering sequencing technology's current trends that are producing longer fragments. Nonetheless, the combinatorial nature of the problem makes larger coverages quickly intractable. An interesting experiment, paradigmatic of a HPC-supported modelling solution, is pWHATSHAP [1, 114], i.e. a freely-available, multicore parallelisation of WHATSHAP, based on the FastFlow parallel programming framework [65]. This parallel implementation on multi-core architectures allows for a relevant reduction of the execution time for haplotyping, while the provided results enjoy the same high accuracy as that provided by WHATSHAP, which increases with coverage.

Metabolic Network Robustness Analysis. Many functional modules are linked together in a Metabolic Network for reproducing metabolic pathways and describing the entire cellular metabolism of an organism. An enormous interdisciplinary interest has grown for metabolic networks robustness studies in terms of errors and attacks tolerance. Scale-free networks based approaches suggest that metabolic networks are tolerant to errors, but very vulnerable to targeted attacks against highly connected nodes. An integrated approach based on statistical, topological, and functional analysis allows for obtaining a deep knowledge on overall metabolic network robustness. With more details, several software frameworks were developed to model a metabolic network and perform the Topological Analysis, the Flux Balance Analysis, and the Extreme Pathways Analysis over it [115,116]. The simulation trials have demonstrated that metabolic network robustness is not simply associated to the network local properties (low-connectivity-degree node or high-connectivity-degree node) but also to functional network properties. So, ultra-peripheral non-hub nodes can assume a fundamental role for network survival if they belong to network extreme pathways, while hub nodes can have a limited impact on networks if they can be replaced by alternative nodes and paths [115,116].

The same approach have been applied as a bio-inspired optimisation method to different application domains. In [117] the use of the previous bio-inspired techniques allows for analysing the structural aspects of a road network, finding its extreme pathways, and outlining the balanced flow combinations. The approach optimises traffic flows over a road network, minimises road congestion and maximises the number of vehicles reaching their destination target. In [118], the bio-inspired methodology has been applied to a class of digital ecosystems based on a scale-free architecture for both maximum information flow and fault/error tolerance detection. Highly connected nodes, inter-module connectors and ultra-peripheral nodes can be identified by evaluating their impact on digital ecosystems behavior and addressing their strengthen, fault tolerance and protection counter-measures.

Modelling Methodologies. The computational analysis of complex biological systems can be hindered by three main factors:

1. modelling the system so that it can be easily understood and analysed by non-expert users is not always possible;
2. When the system is composed of hundreds or thousands of reactions and chemical species, the classic CPU-based simulators could not be appropriate to efficiently derive the behaviour of the system. To overcome these first two limitations, [119] proposes a novel approach that combines the descriptive power of Stochastic Symmetric Nets, a graphical mathematical formalism, with LASSIE, a GPU-powered deterministic simulator that offloads onto the GPU the calculations required to execute many simulations by following both fine-grained and coarse-grained parallelisation strategies. The effectiveness of this approach was showed on a case study aimed at understanding the role of possible malfunctions in the cross-balancing mechanisms that regulate

peripheral tolerance of self-reactive T lymphocytes in case of a relapsing-remitting multiple sclerosis. From our experiments, LASSIE achieves around 97 speed-up with respect to the sequential execution of the same number of simulations;

3. The determination of model structure and model parameters is difficult. Due to economical and technical reasons, only part of these details are well characterised while the rest remains unknown. To deal with this aspect, many parameter estimation and reverse engineering methods were developed. However, these methods often need an amount of experimental data that not always is available.

An alternative approach to deal with situations in which insufficient experimental data hamper the application of PE and RE methods was proposed in [120]. To overcome the lack of information concerning undetermined reactions an empirical biological knowledge was exploited to overcome model indetermination solving an optimisation problem (OP) with an objective function that, similarly to Flux Balance Analysis, is derived from empirical biological knowledge and does not require large amounts of data. The system behaviour is described in detail by a system of ordinary differential equations (ODE) while model indetermination is resolved selecting time-varying coefficients that maximize/minimize the objective function at each ODE integration step. As discussed by the authors, in this context approximation techniques in which OP is not solved at every integration step and/or parallelisation strategies are mandatory to speed-up the solution process.

Learning-Based Modelling Approaches. Some interesting applications in this context are based on the study of integrated biological data and how they are organised in complex systems. In particular, these approaches focus on multi-omic spaces and multi-view analysis. They are very complex applications that require high-throughput analysis based on advanced machine learning (ML) and, more recently, deep learning (DL). One of the several applications in this field is described by Bardozzo et al. [121], where high throughput omic analysis (HTO) is adopted with the aim to the end of describing the antibiotics efficacy with respect to the bacterial adaptive mechanisms. Moreover, a specific survey on HTO is introduced by Suravajhala et al. [122]. Nevertheless, a general survey oriented to high throughput biomedical data analysis with ML and DL is widely described in the work of Serra et al. [123].

Healthcare Management Modelling. Globally healthcare faces many challenges that result in increasing healthcare costs [124] and poor outcomes [125] (morbidity or mortality) depending on the setting and demographic. These challenges have been traced to weak health systems whose symptoms can manifest in: low productivity, poor financial management, inadequate information for decision making, insufficient strategic management, issues with managed care and other systems dynamics [126].

The persistent challenges in the healthcare sector call for urgent review of strategies. Several industry application of operations management have been

documented [127]. There has also been diverse application of operations management techniques in several domains including the health sector. While there is still room for health use-case modification, adoption of innovations used in other domains has been slow. A major classification identified resource and facility management, demand forecasting, inventory and supply chain management, and cost measurement as application groupings to prioritise [126].

An area of increasing interest is human resource planning that captures recruitment, rostering, scheduling and management of clinical and non-clinical staff, their retention, training, payment and incentives as well as performance appraisal. Challenges do also arise around patient workflow: admission, scheduling, and resource allocation. To model solutions to these process and workflow challenges, simple statistics, stochastic [128], mathematical [129], artificial intelligence [130], lean [131], agile [132], six-sigma [131] and total quality management [133] based models have been variously proposed and used. Sometimes, inadequate data may warrant simulation to fill in deterministic and non-deterministic data gaps [134, 135]. This obviously comes with the need for adequate computing and storage capabilities.

The optimum framework for modelling and simulating a particular use-case depends on the availability, structure and size of data [126]. Other considerations will be if the system should be automated or not, if they are sophisticated, deterministic or not. The choice of model and/or simulation technique can ultimately be influenced by available computing power and storage space. How user-friendly such a system is, will be a major consideration as well. Opportunities for application of one or more of these modelling, simulation and prediction techniques to address some of the lingering healthcare challenges is huge.

6 HPC-Enabled Modelling and Simulation for Socio-Economical and Physical Sciences

Many types of decisions in society are supported by modelling and simulation. Some examples are political decisions based on predictive simulations of future climate changes, evacuation planning based on faster-than-real-time simulation of tsunamis, and financial market decisions based on mathematical models emulating current market conditions. In all of these situations, large amounts of data such as global geographical information, measurements of the current physical or financial state, and historical data are used both in the model building and model calibration processes.

We can roughly divide the applications within the large and diverse area, that we here call socio-economical and physical sciences, into two groups. Classical HPC applications, where we build a large-scale complex model and simulate this in order to produce data as a basis for decisions, and Big data applications, where the starting point is a data set, that is processed and analyzed to learn the behaviour of a system, to find relevant features, and to make predictions or decisions.

In classical HPC applications, the need for HPC arises from the fact that we have a large-scale model or a computationally heavy software implementation, that needs to make use of large-scale computational resources, and potentially also large-scale storage resources in order to deliver timely results.

Some particularly challenging problem features are high-dimensionality (e.g., in finance or quantum physics) where the computational costs grow exponentially with the dimension, multi-scale physics (e.g., in climate and tsunami simulations) where scales that differ in orders of magnitude need to be resolved to capture the relevant physical processes, and computations under uncertainty, where the impact of uncertain measurements, parameters and models is quantified through multiple evaluations or extended models leading to an increased computational cost (e.g., in safety critical decision problems).

Highly advanced algorithms and implementations for many different application areas have been developed over decades. A huge challenge is that these legacy codes are not optimized for modern computer architectures and cannot efficiently exploit massively parallel systems [136]. HPC knowledge and innovation is needed to merge the software and hardware state-of-the-art into highly efficient application simulation tools. An opportunity that is brought by the increase in available computer power is instead that the limits of what can be simulated are expanding outwards. The recent increase in research on uncertainty quantification [137] is one example of how this has changed the computational research landscape.

Big data processing as opposed to classical HPC simulation is a relatively young field. The amount of data that is being harvested is following an exponential trend, while hardware development, often in relation to cloud environments, and software development with a specific focus on machine learning and AI is struggling to keep up. The opportunities for using data in new ways are endless, but as is suggested in [138], data and algorithms together can provide the *whats*, while the innovation and imagination of human interpreters is still needed to answer the *whys*. Areas where we see a rapidly growing need for HPC solutions is the internet of things [139], where the expected vast amounts of data provides new challenges for the extraction of knowledge, as well as in the social media context [140], where all kinds of real world events or personal preferences provide footprints that can be tracked and exploited.

In the following paragraphs of this section, we highlight some of the work and contributions of the participants in this Action within the diverse subfields in the wider physical and socio-economical application area. Some of the topics are also represented as individual chapters later in this volume.

Classical HPC Applications. In this sub-field, the interplay of the algorithms with the parallel implementation is crucial, and we provide two examples, both with industrial design applications.

Wing design is one of the essential procedures of aircraft manufactures and it is a compromise between many competing factors and constraints. Efficient numerical optimization methods are important to speed-up the design procedure, especially for design parameters of $\mathcal{O}(10-100)$. In wing shape optimization,

necessary derivatives can easily be calculated by applying finite-difference methods. However, finite difference methods are in general significantly more expensive, requiring at least one additional flow solution per parameter. By using the method of modular analysis and unified derivatives (MAUD), we can unify all methods for computing total derivatives using a single equation with associated distributed-memory, sparse data-passing schemes. Moreover, the wing design requires a set of benchmark cases for the shape optimization to find solutions of many candidate shapes by applying computational fluid dynamics (CFD) analysis with turbulence models. High-fidelity CFD simulations must be carried out in parallel to reduce the total run-time using HPC resources [141].

An application problem that is also discussed later in an individual chapter is electromagnetic scattering, with applications to, e.g., aircraft antenna design. These problems have millions or billions of unknown variables, and the code needs to run on a cluster due to the memory requirements. However, few of the existing (legacy) implementations are parallelised for multicore-based computational nodes. We show results from a pilot implementation using a task-parallel programming model [142], and discuss how to develop this further into a complete distributed implementation.

HPC in Computational Intelligence. As a thriving application platform, HPC excels in supporting execution and it's speedup through parallelisation when running Computational Intelligence (CI) algorithms. The likes of CI algorithms supported by this action includes development of some of most efficient optimization algorithms for continuous optimization as defined with benchmark functions competition framework from Congress on Evolutionary Computation (CEC) 2017 [143, 144]. Specifically useful, in [144] a Differential Evolution (DE) algorithm is enhanced with a new mechanism, the distance based parameter adaptation in the context of Success-History based DE (SHADE), the winner strategy of several previous CEC competitions. An important contribution of an expert system for underwater glider path planning using DE was published in [145], where the application of SHADE strategy enabled significant advances in improved path planning over mesoscale ocean current structures. Another CI technique in learning pipeline is Stability Selection (SS), yet another computationally demanding technique like DE, and SS was improved through a discrete optimization algorithm [146]. In [147], a recent whole pipeline survey overview for black-box discrete optimization benchmarking (BB-DOB) is provided, defining taxonomy, evaluation, and ranking for BB-DOB algorithms. Also, in the case of EU project RIVR (Upgrading National Research Structures in Slovenia) supported by European Regional Development Fund (ERDF), an important side-effect of cHiPSet COST action was leveraging it's experts' inclusiveness to gain capacity recognition at a national ministry for co-financing HPC equipment¹. In the view of future possibilities for modelling and simulation in CI context, gain from HPC is clearly seen in improving upon techniques with DE like in

¹ <https://www.rtv slo.si/znanost-in-tehnologija/v-mariboru-vzpostavljajo-superracunalski-center/475543>.

energy applications [148], constrained trajectory planning [149], artificial life of full ecosystems [150] including HPC-enabled evolutionary computer vision in 2D [151, 152] and 3D [151], many other well recognized real-world optimization challenges [153], or even insight to deep inner dynamics of DE over full benchmarks, requiring large HPC capacities [154].

IoT, Smart Cities, and Big Data Applications. Monitoring the real-world environment is a big challenge given the number of variables that can be sensed nowadays in IoT environments, as for example GPS-position, temperature, humidity, presence, people location, ultraviolet radiation, air quality, hazardous gases, pressure, proximity, acceleration. IoT assumes that multiple sensors can be used to monitor the real-world and this information can be stored and processed, jointly with information from soft-sensor (RSS, web, etc.) [155], to for example assist elderly people in the street [156], develop intelligent interfaces [157] or detect anomalies in industrial environments [158]. In any case, the developed global system needs to fuse heterogeneous data for obtaining a complete view of actual situation and inferring future dangerous situations. These two tasks need Cloud capabilities and HPC.

One of the critical aspects of management within the “smart city” concept is Intelligent Transport Systems, and in particular road traffic control. Off-line traffic simulation is perfect for designing road systems and planning traffic light timings, but does not allow to tackle unexpected or rare situation in real time. Short-term traffic forecasting [159], especially using data-driven (i.e. *learning*) methods, provides a complementary approach. With the availability of smart sensing technologies, like automatic vehicle counting from standard surveillance cameras, it is possible to devise decentralised solutions that measure the current situation of traffic flow on each road, perform local communication between nodes, and forecast the conditions for the immediate future using machine learning algorithms [160]. These may be augmented with evaluations of unexpectedness and per-node traffic jam prediction. Concentration of these data at a decision-making location may also allow travel time estimation, exploitation of network locality information, as well as comparison with the estimates provided by a traffic management system, which can be evaluated for effectiveness on the medium term and possibly tuned accordingly.

Further topics that are discussed in later chapters of this volume look at such diverse questions as how to use data from mobile cellular networks for applications such as urban sensing and event detection, and how sentiment analysis can be applied to forecast the value of cryptocurrencies.

Small Data Applications. The growing big data processing field is well known, but in parallel, there is also a growing interest in a specific type of small data applications. With the increasing instability of the financial and political systems, and of the global climate, there is an increased occurrence of extreme events. Within the big data sets, there are small data sets, that sample the extreme events. To understand their behaviour has applications, e.g., in financial

risk management, in insurance, and in prediction of catastrophic climate events. In a later chapter, methods for extreme value estimation are surveyed.

7 Summary and Conclusion

HPC and M&S form two previously largely disjoint and disconnected research communities. The COST Action IC1406 *High-Performance Modelling and Simulation for Big Data Applications* brings these two communities together to tackle the challenges of big data applications from diverse application domains. Experts from both communities jointly study these applications and application scenarios and cooperatively develop solutions that benefit from the cross-pollination of expertise. Different perspectives on the same topic lead to creative solutions and ultimately to the common goal of HPC-enabled M&S.

The purpose of this paper is to set the scene for individual applications bringing together HPC and M&S. We have argued why high-performance modelling matters for big data applications. Following this line of reasoning we looked at the subject matter from the four angles of the four working groups into which the COST Action is organised. Throughout the previous two sections we have presented a myriad of application opportunities and technological challenges for HPC-enabled modelling and simulation in life, socio-economical and physical sciences. These are complemented by comprehensive surveys of the current state of the art with respect to HPC technology and tools, both from the perspective of programming models as well as from middleware solutions.

Bringing together specialists from all these communities is the central contribution of the COST Action. Having set the scene in this paper, the other papers of this volume exemplify the achievements of the COST Action. Each addresses a specific application or application scenario from the life, socio-economical or physical sciences and explores how the application of state-of-the-art HPC tools and technologies may lead to superior solutions in the near future.

Acknowledgments. This work was supported by the ICT COST Action IC1406 *High-Performance Modelling and Simulation for Big Data Applications*. We would like to thank all members of the COST Action for their direct or indirect contributions to this work and the success of the COST Action in general.

We particularly thank Christoph Kessler, Linköping University, Sweden; Salvatore Vitabile, University of Palermo, Italy; Marco Beccuti, University of Torino, Italy; Lalit Garg, University of Malta, Malta; Jing Gong, KTH Royal Institute of Technology, Stockholm, Sweden; Aleš Zamuda, University of Maribor, Slovenia; José Manuel Molina Lopez, Universidad Carlos III de Madrid, Spain; Amr Abdullatif, Alberto Cabri, Francesco Masulli, and Stefano Rovetta, University of Genova and Vega Research Laboratories, Genova, Italy, for their contributions to this chapter.

References

1. Bracciali, A., et al.: PWHATSHAP: efficient haplotyping for future generation sequencing. *BMC Bioinform.* **17**, 342 (2016)
2. Misale, C., Ferrero, G., Torquati, M., Aldinucci, M.: Sequence alignment tools: one parallel pattern to rule them all? *BioMed. Res. Int.* **2014**, 12 p. (2014). Article ID 539410. <https://doi.org/10.1155/2014/539410>
3. Aldinucci, M., Ruggieri, S., Torquati, M.: Porting decision tree algorithms to multicore using FastFlow. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) *ECML PKDD 2010. LNCS (LNAI)*, vol. 6321, pp. 7–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15880-3_7
4. Buono, D., Danelutto, M., Lametti, S., Torquati, M.: Parallel patterns for general purpose many-core. In: 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 131–139 (2013)
5. Aldinucci, M., et al.: Parallel stochastic systems biology in the cloud. *Brief. Bioinform.* **15**, 798–813 (2014)
6. Aldinucci, M., Drocco, M., Misale, C., Tremblay, G.: Languages for big data analysis. In: Sakr, S., Zomaya, A. (eds.) *Encyclopedia of Big Data Technologies*, pp. 1–12. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-63962-8_142-1
7. Tordini, F., Aldinucci, M., Viviani, P., Merelli, I., Liò, P.: Scientific workflows on clouds with heterogeneous and preemptible instances. In: *Proceedings of the International Conference on Parallel Computing, ParCo 2017*, 12–15 September 2017, Bologna, Italy. *Advances in Parallel Computing*. IOS Press (2018)
8. Akhter, N., Othman, M.: Energy aware resource allocation of cloud data center: review and open issues. *Cluster Comput.* **19**, 1163–1182 (2016)
9. Mastelic, T., Oleksiak, A., Claussen, H., Brandic, I., Pierson, J.M., Vasilakos, A.V.: Cloud computing: survey on energy efficiency. *ACM Comput. Surv.* **47**, 33 (2015)
10. Niewiadomska-Szynkiewicz, E., Sikora, A., Arabas, P., Kamola, M., Mincer, M., Kołodziej, J.: Dynamic power management in energy-aware computer networks and data intensive systems. *Future Gener. Comput. Syst.* **37**, 284–296 (2014)
11. Antal, M., et al.: MoSiCS: modeling, simulation and optimization of complex systems - a case study on energy efficient datacenters. *Simul. Model. Pract. Theory* (2018). <https://doi.org/10.1016/j.simpat.2018.12.004>
12. Karpowicz, M.: Energy-efficient CPU frequency control for the Linux system. *Concurrency Comput.: Pract. Exp.* **28**, 420–437 (2016)
13. Karpowicz, M.P., Arabas, P., Niewiadomska-Szynkiewicz, E.: Energy-aware multilevel control system for a network of Linux software routers: design and implementation. *IEEE Syst. J.* **12**, 571–582 (2018)
14. Cotes-Ruiz, I., Prado, R., Garcia-Galan, S.: Dynamic voltage frequency scaling simulator for real workflows energy-aware management in green cloud computing. *PLoS ONE* **12**, e0169803 (2017)
15. Healy, P.D., Lynn, T., Barrett, E., Morrison, J.P.: Single system image: a survey. *J. Parallel Distrib. Comput.* **90–91**, 35–51 (2016)
16. Oussous, A., Benjelloun, F., Lahcen, A.A., Belfkih, S.: Big data technologies: a survey. *J. King Saud Univ. - Comput. Inf. Sci.* **30**, 431–448 (2018)
17. Berman, F., Fox, G., Hey, A.: *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, Hoboken (2003)
18. Sehgal, N., Bhatt, P.C.P.: *Cloud Computing. Concepts and Practices*. Springer, Heidelberg (2018). <https://doi.org/10.1007/978-3-319-77839-6>

19. MOSIX. www.mosix.org
20. OpenSSI. www.openssi.org/cgi-bin/view?page=openssi.html
21. Kerrighed. www.kerrighed.org
22. UNICORE. www.unicore.eu
23. Globus Toolkit. toolkit.globus.org
24. Cannataro, M.: Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine, and Healthcare. IGI Global, Hershey (2009)
25. Walters, R.J., Crouch, S., Bennett, P.: Building computational grids using ubiquitous web technologies. In: Camarinha-Matos, L.M., Xu, L., Afsarmanesh, H. (eds.) PRO-VE 2012. IAICT, vol. 380, pp. 254–261. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32775-9_26
26. Hwu, W.H. (ed.): GPU Computing Gems, Emerald edn. Morgan Kaufman, Waltham (2011)
27. Singh, A., Bhat, J., Raju, R., D’Souza, R.: Survey on various load balancing techniques in cloud computing. *Adv. Comput.* **7**, 28–34 (2017)
28. Zhang, J., Yu, F.R., Wang, S., Huang, T., Liu, Z., Liu, Y.: Load balancing in data center networks: a survey. *IEEE Commun. Surv. Tutor.* **20**, 2324–2352 (2018)
29. Staples, G.: Torque resource manager. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. ACM, New York (2006)
30. Slurm Workload Manager. slurm.schedmd.com
31. Mohamed, E., Hong, Z.: Hadoop-MapReduce job scheduling algorithms survey. In: 2016 7th International Conference on Cloud Computing and Big Data (CCBD), pp. 237–242 (2016)
32. White, T.: Hadoop: The Definitive Guide. O’Reilly Media, Newton (2015)
33. Apache Spark. spark.apache.org
34. Apache Storm. storm.apache.org
35. Apache Flink. flink.apache.org
36. RapidMiner Studio. rapidminer.com
37. Orange. orange.biolab.si
38. Frank, E., Hall, M.A., Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Burlington (2016)
39. Milne, I., et al.: Tablet-next generation sequence assembly visualization. *Bioinformatics* **26**, 401–403 (2010)
40. Carver, T., Böhme, U., Otto, T., Parkhill, J., Berriman, M.: BamView: viewing mapped read alignment data in the context of the reference sequence. *Bioinformatics* **26**, 676–677 (2010)
41. Rutherford, K., et al.: Artemis: sequence visualization and annotation. *Bioinformatics* **16**, 944–949 (2000)
42. Desale, D.: Top tools for social network analysis and visualisation (2018). www.kdnuggets.com/2015/06/top-30-social-network-analysis-visualization-tools.html/3
43. Sikora, A., Niewiadomska-Szynkiewicz, E.: A federated approach to parallel and distributed simulation of complex systems. *Int. J. Appl. Math. Comput. Sci.* **17**, 99–106 (2007)
44. Inostrosa-Psijas, A., Gil-Costa, V., Marin, M., Wainer, G.: Semi-asynchronous approximate parallel DEVS simulation of web search engines. *Concurrency and Computation: Pract. Exp.* **30**, e4149 (2018)
45. Miller, J., Cotterell, M., Buckley, S.: Supporting a modeling continuum in scalability: from predictive analytics to simulation modeling. In: Proceedings of Winter Simulation Conference: Simulation: Making Decisions in a Complex World, pp. 1191–1202. IEEE Press (2013)

46. Niewiadomska-Szynkiewicz, E., Sikora, A.: A software tool for federated simulation of wireless sensor networks and mobile ad hoc networks. In: Jónasson, K. (ed.) PARA 2010. LNCS, vol. 7133, pp. 303–313. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28151-8_30
47. Song, X., Wu, Y., Ma, Y., Ciu, Y., Gong, G.: Military simulation big data: background, state of the art, and challenges. *Math. Probl. Eng.* **2015**, 1–20 (2015). <https://doi.org/10.1155/2015/298356>
48. Fidjeland, A.K., Roesch, E.B., Shanahan, M.P., Luk, W.: NeMo: a platform for neural modelling of spiking neurons using GPUS. In: 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors, pp. 137–144 (2009)
49. Szynkiewicz, P.: A novel GPU-enabled simulator for large scale spiking neural networks. *J. Telecommun. Inf. Technol.* **2**, 34–42 (2016)
50. Martínez-del-Amor, M.A., Macías-Ramos, L.F., Valencia-Cabrera, L., Riscos-Núñez, A., Pérez-Jiménez, M.J.: Accelerated simulation of P systems on the GPU: a survey. In: Pan, L., Păun, G., Pérez-Jiménez, M.J., Song, T. (eds.) BIC-TA 2014. CCIS, vol. 472, pp. 308–312. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45049-9_50
51. Beyer, J., Hadwiger, M., Pfister, H.: A survey of GPU-based large-scale volume visualization. In: Proceedings of the Eurographics Conference on Visualization (Eurovis 2014), pp. 1–19 (2014)
52. Maeda, R., et al.: Jade: a heterogeneous multiprocessor system simulation platform using recorded and statistical application models. In: Proceedings of HiPEAC Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems, pp. 1–6 (2016)
53. Casanova, H., Giersch, A., Legrand, A., Quinson, M., Suter, F.: Versatile, scalable, and accurate simulation of distributed applications and platforms. *J. Parallel Distrib. Comput.* **74**, 2899–2917 (2014)
54. Calheiros, R., Ranjan, R., Beloglazov, A., Rose, C.D., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **41**, 23–50 (2011)
55. Multi2Sim Workload Manager. www.multi2sim.org
56. Nasir, M.A.U., Morales, G.D.F., García-Soriano, D., Kourtellis, N., Serafini, M.: The power of both choices: practical load balancing for distributed stream processing engines. *CoRR* abs/1504.00788 (2015)
57. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2010, p. 10. USENIX Association, Berkeley (2010)
58. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. *CACM* **51**, 107–113 (2008)
59. Apache Software Foundation: Hadoop. <http://hadoop.apache.org/>. Accessed 2018
60. Intel Corp.: Threading Building Blocks. Accessed 2018
61. Park, I., Voss, M.J., Kim, S.W., Eigenmann, R.: Parallel programming environment for OpenMP. *Sci. Program.* **9**, 143–161 (2001)
62. Pacheco, P.S.: Parallel Programming with MPI. Morgan Kaufmann Publishers Inc., San Francisco (1996)
63. Khronos Compute Working Group: OpenACC Directives for Accelerators. <http://www.openacc-standard.org>. Accessed 2018

64. Aldinucci, M., Danelutto, M., Meneghin, M., Torquati, M., Kilpatrick, P.: Efficient streaming applications on multi-core with FastFlow: the biosequence alignment test-bed. In: *Advances in Parallel Computing*, vol. 19. Elsevier, Amsterdam (2010)
65. Aldinucci, M., Danelutto, M., Kilpatrick, P., Torquati, M.: FastFlow: high-level and efficient streaming on multi-core. In: Pllana, S., Xhafa, F. (eds.) *Programming Multi-core and Many-core Computing Systems. Parallel and Distributed Computing*. Wiley, New York (2017)
66. Enmyren, J., Kessler, C.W.: SkePU: a multi-backend skeleton programming library for multi-GPU systems. In: *Proceedings of the Fourth International Workshop on High-level Parallel Programming and Applications, HLPP 2010*, pp. 5–14. ACM, New York (2010)
67. Grelck, C., Scholz, S.: SAC: a functional array language for efficient multithreaded execution. *Int. J. Parallel Program.* **34**, 383–427 (2006)
68. Grelck, C., Scholz, S., Shafarenko, A.: Asynchronous stream processing with S-net. *Int. J. Parallel Program.* **38**, 38–67 (2010)
69. Lee, E.A., Parks, T.M.: Dataflow process networks. *Proc. IEEE* **83**, 773–801 (1995)
70. Misale, C., Drocco, M., Aldinucci, M., Tremblay, G.: A comparison of big data frameworks on a layered dataflow model. *Parallel Process. Lett.* **27**, 1740003 (2017)
71. Aldinucci, M., Danelutto, M., Anardu, L., Torquati, M., Kilpatrick, P.: Parallel patterns + macro data flow for multi-core programming. In: *Proceedings of International Euromicro PDP 2012: Parallel Distributed and Network-Based Processing*, pp. 27–36. IEEE, Garching (2012)
72. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: *Proceedings of 6th USENIX Symposium on Operating Systems Design & Implementation*, pp. 137–150 (2004)
73. Cole, M.: *Algorithmic Skeletons: Structured Management of Parallel Computations*. Research Monographs in Parallel and Distributed Computing. Pitman, London (1989)
74. Chu, C.T., et al.: Map-reduce for machine learning on multicore. In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pp. 281–288 (2006)
75. Zaharia, M., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation* (2012)
76. Carbone, P., Fóra, G., Ewen, S., Haridi, S., Tzoumas, K.: Lightweight asynchronous snapshots for distributed dataflows. *CoRR* abs/1506.08603 (2015)
77. Toshniwal, A., et al.: Storm@twitter. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 147–156 (2014)
78. Akidau, T., et al.: The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proc. VLDB Endowment* **8**, 1792–1803 (2015)
79. Kitchenham, B., Brereton, O.P., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering - a systematic literature review. *Inf. Softw. Technol.* **51**, 7–15 (2009)
80. Calzone, L., Pages, F., Soliman, S.: BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics* **22**, 1805–1807 (2006)
81. Zechner, C., Seelig, G., Rullan, M., Khammash, M.: Molecular circuits for dynamic noise filtering. *Proc. Natl. Acad. Sci.* **113**, 4729–4734 (2016)

82. Fages, F., Soliman, S.: On robustness computation and optimization in BIOCHAM-4. In: Proceedings of Computational Methods in Systems Biology - 16th International Conference, CMSB 2018, Brno, Czech Republic, 12–14 September 2018, pp. 292–299 (2018)
83. Nasti, L., Gori, R., Milazzo, P.: Formalizing a notion of concentration robustness for biochemical networks. In: Mazzara, M., Ober, I., Salaün, G. (eds.) STAF 2018: Software Technologies: Applications and Foundations. LNCS, vol. 11176, pp. 81–97. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04771-9_8
84. Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies. In: BIOINFORMATICS 2019 (2019, in Press)
85. Sansom, C., Castiglione, F., Lio, P.: Metabolic disorders: how can systems modelling help? *Lancet Diabetes Endocrinol.* **4**, 306 (2016)
86. Bartocci, E., Liò, P.: Computational modeling, formal analysis, and tools for systems biology. *PLOS Comput. Biol.* **12**, 1–22 (2016)
87. Capobianco, E., Liò, P.: Comorbidity networks: beyond disease correlations. *J. Complex Netw.* **3**, 319–332 (2015)
88. Capobianco, E., Liò, P.: Comorbidity: a multidimensional approach. *Trends Mol. Med.* **19**, 515–521 (2013)
89. Bartocci, E., Liò, P., Merelli, E., Paoletti, N.: Multiple verification in complex biological systems: the bone remodelling case study. In: Priami, C., Petre, I., de Vink, E. (eds.) Transactions on Computational Systems Biology XIV. LNCS, vol. 7625, pp. 53–76. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35524-0_3
90. Liò, P., Paoletti, N., Moni, M., Atwell, K., Merelli, E., Viceconti, M.: Modelling osteomyelitis. *BMC Bioinform.* **13**(Suppl. 14), S12 (2012)
91. Paoletti, N., Liò, P., Merelli, E., Viceconti, M.: Multilevel computational modeling and quantitative analysis of bone remodeling. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **9**, 1366–78 (2012)
92. Liò, P., Merelli, E., Paoletti, N., Viceconti, M.: A combined process algebraic and stochastic approach to bone remodeling. *Electron. Notes Theor. Comput. Sci.* **277**, 41–52 (2011). The Second International Workshop on Interactions Between Computer Science and Biology
93. Luisi, P.L., Ferri, F., Stano, P.: Approaches to semi-synthetic minimal cells: a review. *Naturwissenschaften* **93**, 1–13 (2006)
94. Kuruma, Y., Stano, P., Ueda, T., Luisi, P.L.: A synthetic biology approach to the construction of membrane proteins in semi-synthetic minimal cells. *Biochimica et Biophysica Acta (BBA)-Biomembranes* **1788**, 567–574 (2009)
95. Lorenzo, C., Ospri, L.L., Marangoni, R.: On fine stochastic simulations of liposome-encapsulated pure systems. *Commun. Comput. Inf. Sci.* **587**, 146–158 (2016)
96. Lazzerini-Ospri, L., Stano, P., Luisi, P., Marangoni, R.: Characterization of the emergent properties of a synthetic quasi-cellular system. *BMC Bioinform.* **13**, S9 (2012)
97. Fanti, A., Gammuto, L., Mavelli, F., Stano, P., Marangoni, R.: Do protocells preferentially retain macromolecular solutes upon division/fragmentation? A study based on the extrusion of POPC giant vesicles. *Integr. Biol.* **10**, 6–17 (2017)
98. Calviello, L., Stano, P., Mavelli, F., Luisi, P.L., Marangoni, R.: Quasi-cellular systems: stochastic simulation analysis at nanoscale range. *BMC Bioinform.* **14**, S7 (2013)
99. The 1000 Genomes Project Consortium: A global reference for human genetic variation. *Nature* **526**, 68–74 (2015)

100. The Computational Pan-Genomics Consortium: Computational pan-genomics: status, promises and challenges. *Brief. Bioinform.* **19**, 118–135 (2018)
101. Huang, L., Popic, V., Batzoglu, S.: Short read alignment with populations of genomes. *Bioinformatics* **29**, 361–370 (2013)
102. Bille, P., Landau, G.M., Raman, R., Sadakane, K., Satti, S.R., Weimann, O.: Random access to grammar-compressed strings. In: 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 373–389 (2011)
103. Navarro, G.: Indexing highly repetitive collections. In: 23rd International Workshop on Combinatorial Algorithms (IWOCA), pp. 274–279 (2012)
104. Gagie, T., Gawrychowski, P., Puglisi, S.J.: Faster approximate pattern matching in compressed repetitive texts. In: 22nd International Symposium on Algorithms and Computation (ISAAC), pp. 653–662 (2011)
105. Iliopoulos, C.S., Kundu, R., Pissis, S.P.: Efficient pattern matching in elastic-degenerate texts. In: Drewes, F., Martín-Vide, C., Truthe, B. (eds.) *LATA 2017*. LNCS, vol. 10168, pp. 131–142. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-53733-7_9
106. Danecek, P., et al.: The variant call format and VCFtools. *Bioinformatics* **27**, 2156–2158 (2011)
107. Holub, J., Smyth, W.F., Wang, S.: Fast pattern-matching on indeterminate strings. *J. Discret. Algorithms* **6**, 37–50 (2008)
108. Grossi, R., et al.: On-line pattern matching on a set of similar texts. In: *CPM. LIPIcs* (2017)
109. Bernardini, G., Pisanti, N., Pissis, S.P., Rosone, G.: Pattern matching on elastic-degenerate text with errors. In: Fici, G., Sciortino, M., Venturini, R. (eds.) *SPIRE 2017*. LNCS, vol. 10508, pp. 74–90. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67428-5_7
110. Pissis, S.P., Retha, A.: Dictionary matching in elastic-degenerate texts with applications in searching VCF files on-line. In: 17th International Symposium on Experimental Algorithms (SEA), pp. 16:1–16:14 (2018)
111. Aoyama, K., Nakashima, Y., Inenaga, S., Bannai, H., Takeda, M.: Faster online elastic degenerate string matching. In: 29th Annual Symposium on Combinatorial Pattern Matching, (CPM). *LIPIcs*, pp. 9:1–9:10 (2018)
112. Alzamel, M., et al.: Degenerate string comparison and applications. In: 18th International Workshop on Algorithms in Bioinformatics (WABI). *LIPIcs*, pp. 21:1–21:14 (2018)
113. Patterson, M., et al.: WhatsHap: weighted haplotype assembly for future-generation sequencing reads. *J. Comput. Biol.* **22**, 498–509 (2015). PMID: 25658651
114. Aldinucci, M., Bracciali, A., Marschall, T., Patterson, M., Pisanti, N., Torquati, M.: High-performance haplotype assembly. In: di Serio, C., Liò, P., Nonis, A., Tagliaferri, R. (eds.) *CIBB 2014*. LNCS, vol. 8623, pp. 245–258. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24462-4_21
115. Vitabile, S., Conti, V., Lanza, B., Cusumano, D., Sorbello, F.: Metabolic networks robustness: theory, simulations and results. *J. Interconnection Netw.* **12**, 221–240 (2012)
116. Vitabile, S., Conti, V., Lanza, B., Cusumano, D., Sorbello, F.: Topological information, flux balance analysis, and extreme pathways extraction for metabolic networks behaviour investigation. *IOSPress*, no. 234, pp. 66–73 (2011)
117. Vitello, G., Alongi, A., Conti, V., Vitabile, S.: A bio-inspired cognitive agent for autonomous urban vehicles routing optimization. *IEEE Trans. Cogn. Dev. Syst.* **9**, 5–15 (2017)

118. Conti, V., Ruffo, S., Vitabile, S., Barolli, L.: BIAM: a new bio-inspired analysis methodology for digital ecosystems based on a scale-free architecture. *Soft Comput.* **23**(4), 1133–1150 (2019)
119. Beccuti, M., et al.: GPU accelerated analysis of treg-teff cross regulation in relapsing-remitting multiple sclerosis. In: Mencagli, G., et al. (eds.) *Euro-Par 2018. LNCS*, vol. 11339, pp. 626–637. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10549-5_49
120. Totis, N., et al.: Overcoming the lack of kinetic information in biochemical reactions networks. *ACM SIGMETRICS Perform. Eval. Rev.* **44**, 91–102 (2017)
121. Bardozzo, F., Lió, P., Tagliaferri, R.: A study on multi-omic oscillations in *Escherichia coli* metabolic networks. *BMC Bioinform.* **19**, 194 (2018)
122. Suravajhala, P., Kogelman, L.J.A., Kadarmideen, H.N.: Multi-omic data integration and analysis using systems genomics approaches: methods and applications in animal production, health and welfare. *Genet. Sel. Evol.* **48**, 38 (2016)
123. Serra, A., Galdi, P., Tagliaferri, R.: Machine learning for bioinformatics and neuroimaging. *Wiley Interdisc. Rev. Data Min. Knowl. Discov.* **8**, e1248 (2018)
124. McClean, S., Gillespie, J., Garg, L., Barton, M., Scotney, B., Kullerton, K.: Using phase-type models to cost stroke patient care across health, social and community services. *Eur. J. Oper. Res.* **236**, 190–199 (2014)
125. WHO: World Health Statistics 2018: Monitoring the SDGs. Technical report (2018)
126. Garg, L., McClean, S., Barton, M.: Can management science methods do more to improve healthcare? (2014)
127. Jahangirian, M., et al.: A rapid review method for extremely large corpora of literature: applications to the domains of modelling, simulation, and management. *Int. J. Inf. Manag.* **31**, 234–243 (2011)
128. Garg, L., Barton, M., Meenan, B.J., Fullerton, K.: Intelligent patient management and resource planning for complex, heterogeneous, and stochastic healthcare systems. *IEEE Trans. Syst. Man Cybern. - Part A Syst. Hum.* **42**, 1332–1345 (2012)
129. Garnett, G., Cousens, S., Hallett, T., Steketee, R., Walker, N.: Mathematical models in the evaluation of health programmes. *Lancet* **378**, 515–525 (2011)
130. Shanmugam, S., Garg, L.: Model employee appraisal system with artificial intelligence capabilities. *J. Cases Inf. Technol.* **17**, 30–40 (2015)
131. Aleem, S.: Translating 10 lessons from lean six sigma project in paper-based training site to electronic health record-based primary care practice: challenges and opportunities. *Qual. Manag. Health Care* **22**, 224 (2013)
132. Kannan, V., Fish, J.C., Willett, D.L.: Agile model driven development of electronic health record-based specialty population registries (2016)
133. Heidari Gorji, A., Farooque, J.: A comparative study of total quality management of health care system in India and Iran. *BMC Res. Notes* **4**, 566 (2011)
134. Garg, L., Dauwels, J., Earnest, A., Leong, K.P.: Tensor-based methods for handling missing data in quality-of-life questionnaires. *IEEE J. Biomed. Heal Inform.* **18**, 1571–1580 (2014)
135. McClean, S., Barton, M., Garg, L., Fullerton, K.: A modeling framework that combines Markov models and discrete-event simulation for stroke patient care. *ACM Trans. Model. Comput. Simul.* **21**, 25 (2011)
136. Herlihy, M., Luchangco, V.: Distributed computing and the multicore revolution. *SIGACT News* **39**, 62–72 (2008)
137. Smith, R.C.: Uncertainty Quantification: Theory, Implementation, and Applications, vol. 12. SIAM (2013)

138. John Walker, S.: *Big Data: A Revolution that will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt, Boston (2014)
139. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things: a survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **17**, 2347–2376 (2015)
140. Kennedy, H.: *Post, Mine, Repeat: Social Media Data Mining Becomes Ordinary*. Springer, London (2016). <https://doi.org/10.1057/978-1-137-35398-6>
141. Zhang, M., Melin, T., Gong, J., Barth, M., Axner, L.: Mixed fidelity aerodynamic and aero-structural optimization for wings. In: 2018 International Conference on High Performance Computing and Simulation, pp. 476–483 (2018). QC 20180808
142. Zafari, A., et al.: Task parallel implementation of a solver for electromagnetic scattering problems (2018). <https://arxiv.org/abs/1801.03589>
143. Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., Zamuda, A.: Distance based parameter adaptation for differential evolution. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–7. IEEE (2017)
144. Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T., Zamuda, A.: Distance based parameter adaptation for success-history based differential evolution. *Swarm Evol. Comput.* (2018)
145. Zamuda, A., Sosa, J.D.H.: Success history applied to expert system for underwater glider path planning using differential evolution. *Expert Syst. Appl.* **119**, 155–170 (2019)
146. Zamuda, A., Zarges, C., Stiglic, G., Hrovat, G.: Stability selection using a genetic algorithm and logistic linear regression on healthcare records. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2017)*, pp. 143–144 (2017)
147. Zamuda, A., Nicolau, M., Zarges, C.: A black-box discrete optimization benchmarking (BB-DOB) pipeline survey: taxonomy, evaluation, and ranking. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2018)*, pp. 1777–1782 (2018)
148. Glotić, A., Zamuda, A.: Short-term combined economic and emission hydrothermal optimization by surrogate differential evolution. *Appl. Energy* **141**, 42–56 (2015)
149. Zamuda, A., Sosa, J.D.H., Adler, L.: Constrained differential evolution optimization for underwater glider path planning in sub-mesoscale eddy sampling. *Appl. Soft Comput.* **42**, 93–118 (2016)
150. Zamuda, A., Brest, J.: Environmental framework to visualize emergent artificial forest ecosystems. *Inf. Sci.* **220**, 522–540 (2013)
151. Zamuda, A., Brest, J.: Vectorized procedural models for animated trees reconstruction using differential evolution. *Inf. Sci.* **278**, 1–21 (2014)
152. Zamuda, A., Mlakar, U.: Differential evolution control parameters study for self-adaptive triangular brushstrokes. *Informatica - Int. J. Comput. Inform.* **39**, 105–113 (2015)
153. Zamuda, A., Brest, J.: On tenfold execution time in real world optimization problems with differential evolution in perspective of algorithm design. In: 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 1–5. IEEE (2018)
154. Zamuda, A., Brest, J.: Self-adaptive control parameters' randomization frequency and propagations in differential evolution. *Swarm Evol. Comput.* **25**, 72–99 (2015)
155. Blázquez Gil, G., Berlanga, A., Molina, J.M.: InContexto: multisensor architecture to obtain people context from smartphones. *Int. J. Distrib. Sens. Netw.* **8** (2012)

156. Cristina-Bicharra, A., Vivacqua, C., Sanchez-Pi, N., Martí, L., Molina, J.M.: Crowd-based ambient assisted living to monitor elderly health in outdoor settings. *IEEE Softw.* **34**, 53–57 (2017)
157. Griol, D., Molina, J.M., Sanchis, A.: Integration of context-aware conversational interfaces to develop practical applications for mobile devices. *J. Ambient Intell. Smart Environ. (JAISE)* **9**, 561–577 (2017)
158. Marti, L., Sanchez-Pi, N., Molina, J.M., Bicharra, A.C.: Anomaly detection based on sensor data in petroleum industry applications. *Sensors* **15**, 2774–2797 (2015)
159. Vlahogianni, E.I., Karlaftis, M.G., Golias, J.C.: Short-term traffic forecasting: where we are and where we're going. *Transp. Res. Part C: Emerg. Technol.* **43**, 3–19 (2014)
160. Abdullatif, A., Masulli, F., Rovetta, S.: Tracking time evolving data streams for short-term traffic forecasting. *Data Sci. Eng.* **2**, 210–223 (2017)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

