Towards Green Big Data at CERN

Tapio Niemi[a,d], Jukka K. Nurminen[a,b,c], Juha-Matti Liukkonen[e], Ari-Pekka Hameri[a,d]

[a]*Helsinki Institute of Physics, CERN, Switzerland*
[b]*VTT Technical Research Centre of Finland*
[c]*Department of Computer Science, Aalto University, Finland*
[d] *Faculty of Business and Economics, University of Lausanne, Lausanne, Switzerland*
[e]*Reaktor, Helsinki, Finland*

Abstract

High-energy physics studies collisions of particles traveling near the speed of light. For statistically significant results, physicists need to analyze a huge number of such events. One analysis job can take days and process tens of millions of collisions. Today the experiments of the large hadron collider (LHC) create 10 GB of data per second and a future upgrade will cause a ten-fold increase in data. The data analysis requires not only massive hardware but also a lot of electricity. In this article, we discuss energy efficiency in scientific computing and review a set of intermixed approaches we have developed in our Green Big Data project to improve energy efficiency of CERN computing. These approaches include making energy consumption visible to developers and users, architectural improvements, smarter management of computing jobs, and benefits of cloud technologies. The open and innovative environment at CERN is an excellent playground for different energy efficiency ideas which can later find use in mainstream computing.

Keywords: Scientific computing, green computing, CERN, energy efficiency

1. Introduction

The Large Hadron Collider (LHC), which was used to discover the famous Higgs boson, started to run at CERN in 2010. During the first run, the CERN computing center stored up to 6 GB of data per second. The total need of computing resources was around 200,000 CPUs and 40 PB disk space. The second run of LHC started in June 2015. During this run, data is stored at a maximum rate of 10 GB per second and CERN alone has allocated 140 PB disk space divided between its data centers in Switzerland and Hungary [1]. The required computing resources for LHC data analysis are divided among 11 tier-1 and 155 tier-2 globally distributed computing centers by using the grid computing paradigm [2]. Efficient management of these computing resources is vital for the success of the project, which is foreseen to be active for the next 20 years.

From the physicist point of view computational speed is of prime importance to efficiently analyze the data and to enable progress on particle physics. It is therefore important to consider both user needs and cost-efficient use of resources when managing the computing infrastructure and training and encouraging users.

The energy efficiency of computing is receiving increasing attention (see e.g., [3] for a survey). For example, Van Heddeghem et al. [4] reported that data centers worldwide consumed 270 TWh of energy in 2012 and this consumption had a Compound Annual Growth Rate (CAGR) of 4.4% from 2007 to 2012. Besides the expenses related to the data center energy consumption, environmental aspects are also relevant. Therefore, the reduction of electricity consumption for computing is important both from cost and environmental point of view.

The increased computational speed enabled by Moore's law has for a long time contributed to increased energy efficiency. If the same task can be completed faster and the power consumption remains unchanged, this obviously results into energy savings. However, we are now in a situation where the computational speed will no longer increase. While Moore's law still increases the number of transistors on chips, the computational speed of individual cores has stopped growing [5]. As a result developers need to be able to better distribute their workloads to take advantage of the increasing number of cores in networked systems. However, according to Amdahl's law [6] the distribution is not a panacea, because the speed improvement of parallelism is limited by the parts of the software that cannot be parallelized. Normally this has a negative effect on energy efficiency but in some cases, due to the dynamic voltage and frequency scaling (DVFS) [7], increased parallelism can even have a positive effect on energy efficiency [8]. One conclusion of this is that instead of simply trying to make the software run faster, and, as a result, become more energy-efficient, we need to consider other ways to run the software in a more energy-efficient fashion.

In this article, we take a holistic view of energy efficiency by introducing three main roles, which can be recognized in the value-chain of scientific computing: user, software developer, and data center operator. These three groups are connected through the computing system they use/develop/operate even though their aims and goals are often quite different and orthogonal. This also naturally affects energy consumption. Therefore, we look at energy efficiency from these three perspectives and form a holistic view of the scientific computing ecosystem. We do this by combining the roles with a set of intermixed approaches, which we have studied in our Green Big Data project (`https://twiki.cern.ch/twiki/bin/view/Main/GreenBigData`) to improve the energy efficiency of CERN computing. Philosophically, we can say that our research methodology is based on decomposing the whole system into independent subsystems, which can be optimized separately [9]. The result of optimization is usually Pareto optimal, meaning that improving the system from the view point of one role, would reduce its optimality for another role. For example, allowing longer queueing times can make it possible to improve the energy efficiency of the data center but it also reduces the service level of the user.

The rest of this article is organized as follows: First, we give a review of related work in energy efficiency in Section 2 and then introduce the reader to scientific computing by reviewing the CERN computing problem (Section 3). In Section 4 we introduce the three roles/stakeholder groups in computing and present possible technology solutions to improve energy efficiency related to the functions of these groups. In Section 5, we discuss future possibilities and identify potential targets for future research. Finally, conclusions are given in Section 6.

2. Related Work

A large part of research in energy efficiency has focused on cloud data centers. For example, Dayarathna et al. [10] give a large survey of state-of-art techniques in energy consumption modeling and prediction for Internet data centers, and Shuja et al. [11] present several case studies demonstrating methods and techniques for sustainable data centers. Moreover, Mazumdar and Pranzo [12] study server consolidation in cloud data centers by proposing a formal formulation for the server consolidation problem and showing that using a snapshot-based method it is possible to find efficiently near optimal server allocations.

Another significant part of research focuses on hardware. For example Karpowichz et al. [13] study on energy-aware design in hardware, middleware and software layers. They note that to get benefit on hardware development, a holistic view to the whole system must be taken. This includes, for example, developing power consumption models, measuring methods for energy efficiency, modeling computing and network dynamics, multi-level control systems, energy-aware scheduling and software development techniques.

There are also many studies on energy efficiency in high-performance computing. For example, Rong et al. [14] review energy optimization technologies in high-performance computing and propose a set of strategies to maximize the efficiency and minimize the impact for environment. Further, Zakarya and Gilliam [15] focus on energy efficiency on scientific computing systems. They key findings are: 1) using system level technologies may actually increase energy consumption in clusters; 2) optimizing scheduling and resource allocation in clouds can offer better results than consolidation using migrations; and 3) turning off idle resources works well in clusters but may cause performance issues in cloud when demands fluctuates.

There are also many other studies on resource management and scheduling in scientific computing. Uddin et al. [16] evaluate three scheduling algorithms to find the most energy-efficient one. The algorithms were implemented using the CloudSim software to simulate IaaS cloud infrastructure. The results indicates that the two phases power convergence (TPPC) algorithm [17] is the most energy-efficient of the tested algorithms. Zhao et al. [18] propose an energy and deadline aware scheduling method for data intensive applications. The method is based on the idea of modeling data sets as a binary tree based on correlations among them. This helps reducing data transmission. The second step of the method is based on energy-aware scheduling minimizing the number of active servers. Finally, Madni et al. [19] study resource allocation methods in their review article. Their conclusion is that not all important parameters are taken into account in current methods and improvements would be needed.

Energy-aware algorithms have been received a lot of attention during the last years. Many of these algorithms aim at optimizing resource selection or scheduling problems [20, 21, 22], while others focus on cloud computing [23, 24, 25] or networking [26, 27, 28].

Although most of the research on energy efficiency have focused on hardware, infrastructure, or algorithms there are many studies on software development, too. For example, Jagroep et al. [29] study how to make software developers aware on energy efficiency. In their case study they followed two software development projects and gave feed back to the developers on energy and performance issues. The results indicate that increased awareness makes the developers consider more on energy efficiency.

Energy-efficient operation is naturally highly important for major cloud service providers (e.g. [30] for Google,[31] for Facebook). Although there are commonalities, the key difference between big cloud operators and scientific computing community is that in cloud providers are hosting services which often require low latency to keep the interactive users happy. In scientific computing single jobs can run for hours or even days and thus high throughput is often far more important than shorter runtime. Therefore the architectural concepts are not directly transferable between the two camps but a lot of the learnings can still be beneficial for both.

Moreover, scientific computing has been slow in adopting virtualization, containers, and other techniques which form the basis of commercial cloud services. One reason for the slow adoption has been the belief that all kinds of additional layers waste computing resources. However, some studies indicate that the performance difference especially with container technology is not very significant [32]. Moreover, the resource isolation of containers allows new ways to manage the computation and the possibility to store the entire computational environment in the container provides new opportunities for reproducible research [33]. Therefore it is likely that we see increasing adoption of container technologies in the future in scientific computing following the initial steps already taken [34, 35, 36].

3. Computing Challenge at CERN

In the 27 km long LHC ring particles are accelerated close to the speed of light. Two particle beams traveling in opposite directions collide at the detectors of experiments. There are three main experiments at LHC: CMS [37], Atlas [38], and LHCb [39]. Each experiment has a custom made particle detector consisting of multiple layers of different sensors, which measure speed, charge, and other characteristics of the particles that are created in the collisions. While the number of monitored collisions is large, only some of them are selected for further studies. The first selection is made in real time by dedicated trigger hardware [40] in the experiment and the second filtering using a cluster of powerful computing servers [41]. The data from the selected events is permanently stored on the computing center. The raw event data is processed and converted into a reconstructed event, which is more compact and more suitable for analysis. The actual analysis phase then selects a set of events matching according to some criteria with the simulated events, which, based on theoretical models, try to anticipate what should be happening in the collisions. In order to understand physics phenomena and get statistically significant results a huge number of measured events must be compared with simulated events.

A single high-energy physics analysis job processes millions of events. The work can be parallelized easily because each event can be analyzed independently. Normally, each analysis job starts around 600 - 1,000 separate processes each of which analyzes a subset of total events. Physics events are stored in database like repositories called ROOT files [1]. A typical ROOT file is about 100 - 300 MB in size and contains 700 - 2,000 events. An average analysis job would pass through 15 million events that are stored in 35,000 files.

Originally, the huge computing needs of LHC were handled by the grid computing model (see e.g., [42] for an overview). For this purpose, large R&D projects were launched in early 2000 [43, 44]. Initially, achieving the required computing power was the only target and the importance of minimizing energy consumption was not understood yet. Some years later, however, offering enough power started to limit the number of servers that could be installed in the CERN data center. Outsourcing a part of capacity to another data center to Hungary using cloud technologies was found as a working solution. Moving from grid to cloud offers new possibilities for energy optimization such as consolidation of computing [45] either automatically or through human expertise.

So far the increasing need for computing power has been satisfied by purchasing more hardware and by performance improvements. Moreover the computing has been distributed to multiple locations with grid and cloud technologies. Until recently Moores law has handled the increasing computing needs quite well and the power efficiency has improved at least at the same rate as computing power. But as computational needs continue to grow and keeping up with Moores law is increasingly difficult, new ideas and technologies are needed.

Scientific computing at CERN is an interesting case since the problems are real, the amount of data massive, and the atmosphere at CERN like in the science community in general, is receptive to new ideas. An idea tested in the challenging environment of CERN has a good possibility of being useful elsewhere. The most known example of this is the World Wide Web [46], which has its roots in CERN. However, it is important to keep in mind that there are differences between scientific and general computing environments. For instance, power proportional computing is useful in many areas in the computing industry [47], but its impact on scientific computing at CERN is small since there are enough computing tasks to keep the batch processing queues full and the servers busy all the time. Obviously, this does not alone mean that CPU utilisation would be 100%, since there can be latencies, for example, because of waiting for I/O. However, high CPU utilisation can be guaranteed by slightly over committing CPU resources [48]. Moreover, long computing times spanning hours or days are a norm in scientific computing while interactive applications naturally require rapid responses.

4. Three Roles in Energy Efficiency

Generally, green computing has been seen as a technical solution and there are plenty of methodologies and technologies for improving the energy efficiency of super computing clusters [49, 50]. However, in many cases, the motivation for investing in energy efficient technologies remains quite low. For example, quite often hosting charges in data centers are based on the amount of electricity used. This clearly demotivates the company running the data center in the effort to save energy in its computing infrastructure. However, improving other parts of the infrastructure such as cooling still makes sense. Partially for this reason, the PUE index [51] is a commonly used key performance indicator for energy efficiency. Or, in some other organizations, the electricity bill for computing is still very small compared to other costs such as rent, personnel, etc. Optimizing computing systems for energy efficiency can also increase risks in operational failures or make those risks more pronounced. For example, a hardware failure of a virtualized server running several services can have a negative impact for many customers.

In scientific computing, like at CERN, we can recognize three different groups of actors: 1) Users, that is, scientists solving their problems by running scientific software packages on computing clusters, 2) Developers of scientific software packages, and 3) Operators of the computing clusters and the data center. Based on this classification, we can form three different and mostly orthogonal views of energy efficiency:

End user view How can a scientist using computational methods in his/her research improve energy efficiency and avoid wasting resources or energy (Section 4.1)?

Software developer view How can energy-efficient software be written for complex multi-core-multi-user systems? What can the software developer do to improve the energy-efficiency of the code (Section 4.2.2) and what kind of tool support would help in this (Section 4.2.1)?

Data center management view How can resources be managed in an optimal but still fair way? How can we influence energy spending by better scheduling and allocating work loads in a data center locally (Section 4.3), and also globally by dividing the work between multiple data centers in an energy optimal way (Section 4.4)?

A way to study the efficiency of scientific computing is to look at the ratio between computing outcomes and expenses. Computing expenses can be classified into hardware, electricity, and personnel costs, as well as the impact on the environment. While the three first costs are easy to measure, the environmental impact is a more hidden factor. Fortunately, through electricity pricing the common objectives of minimizing energy expenses and minimizing the environmental impact have strong synergies.

The outcomes of computing are more difficult to measure than its costs. Obviously computing is done for a goal but quantifying the value of computing results is difficult as well as comparing the values of different computing outcomes. Therefore it is difficult to come up with useful measures of efficiency by simply dividing the value of computing output with the computing costs. CPU time is often used as a substitute measure of computing output although it does not indicate how efficiently the implemented algorithms work and use the available resources. There are also a lot of trade-offs: for example, between the computing speed and the cost (i.e., electricity or CO2 emissions).

Keeping in mind the limitations that arise from the difficulties of exact metering, in the following subsections we study how different solutions can help the actors in the roles of end users, developers, and data center managers to improve the energy efficiency in the parts of the computing system they can influence.

4.1. User View for Energy Efficiency

While technical solutions in computing hardware form the basis of energy efficiency, the way computers are used also plays an important role in energy efficiency. Efficiency, meaning optimal usage of available resources and avoiding generating waste, is a widely-studied topic in operations and production management [52] but it is usually understood in a narrower sense in scientific computing, where the main target is optimising the computing speed. One particular challenge is to motivate users to write and run their programs in an energy-efficient way. The users are focused on getting the results they need and not on looking at the bigger picture. For example, creative ways to greedily use the computing resources for gaining an advantage over other users, such as running multiple parallel tasks for solving an optimization problem and picking-up the best solution, or bypassing the cluster's scheduling system by using so called pilot jobs [53] for reserving resources [54], are clearly not be the most energy-efficient way.

It is however natural for a single user to minimize both the set-up and run time of his/her own application. However, this seldom results into the best total (energy) efficiency for the whole user community. If every user behaves in this way, everyone has less resources available. In closed communities, such as among CERN scientists, these issues could be alleviated by better training and also by introducing tools giving feed-back to the users on the energy and computational efficiency of their applications. Usually people want to reduce the environmental impact if it is not too difficult or costly for them [55].

How can the users be motivated to save energy? The following could possibly provide some incentives:

- Moving to energy based accounting and billing.
- Offering benefits to energy-efficient users. For example allocating higher priority for their jobs or giving them the most powerful hardware.
- Promoting environmental awareness and making it possible to schedule jobs based on the availability of green electricity.

4.2. Software Developers View

For the software developers we can identify at least two methodologies for improving energy efficiency: 1) making energy consumption visible and 2) advising them to use optimal methods for the architecture at hand. These are studied in the following sub sections.

4.2.1. Energy Profiling

As far back as in the 19th century Lord Kelvin observed that "...when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind" [56]. Tom DeMarco applied this to software development by famously stating that "You cannot control what you cannot measure" [57].

The observation also applies well to the energy consumption. Most developers do not have any idea how much electricity it takes to run their application. In comparison to computing speed, the energy-spending of computing is hard to detect. In mobile devices consumers are sensitive to power consumption, because of the frequent need to recharge influences the comfort of using the device. However, in data centers this is not the case and therefore, the energy consumption of computing is generally beyond the interest of a typical software developer or end user. Obviously the electricity bill of the data center is in the interest of the IT managers, but the link to the developer is typically weak.

An obvious way to make developers consider the energy efficiency of their code is to make energy spending visible. There are multiple ways to achieve this. First, tools like RAPL (Intel technology for programmatic access to counters giving the energy spending of the processor chip [58]) or smart power sources allow programmatic access to counters that keep track of spent electricity [59]. In this way we can easily monitor the electricity consumption of the computing server (or even some smaller units like CPU or memory) but it is hard to know the share of a single job's electricity consumption relative to the overall energy consumption of the computing facility. Furthermore it is difficult to identify how different parts of the software use the energy.

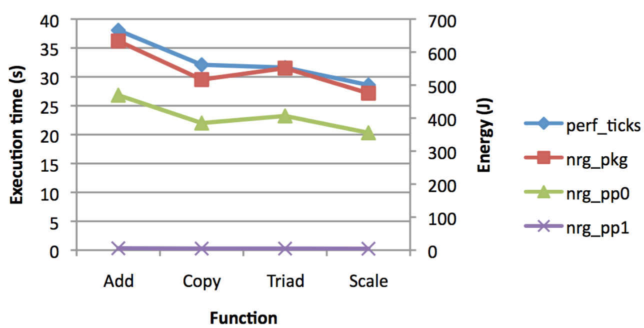

Figure 1: An example of measurements taken by the performance and energy profiling tool. [60]

To make the energy usage more visible to the software developers, we implemented an energy profiling functionality to IGProf [60]. IGProf is an open source profiler developed initially at CERN [61]. With its energy profiling features, it is able to report the amount of energy consumed in each function [62] in addition to the processing time. The essence of the energy profiler is the RAPL measurements, which tell how much energy has been consumed between consecutive readings, and a mechanism to allocate the energy reading to different functions.

While the energy profiler is operational and the concept works, it still has two main challenges. First, the granularity of energy measurements is rather coarse, around 1 ms. Therefore, it is difficult to allocate the spent energy to the proper functions because a program typically executes thousands of function calls before RAPL updates the energy counters. An option used by Hähnel et al. [63] is to artificially spend time in a function until the next reading is available. However, this has the drawback of making the execution of the software extremely slow. An alternative option is to rely on statistics: if the same code is executed multiple times, the energy readings should over time accumulate to those functions that are responsible for the consumption.

The second problem with energy profiling concerns the value of separate energy readings. In many cases there is a very strong correlation between energy-spending and processing time and therefore it is reasonable to ask what is the added value of energy profiling. However, while the correlation is strong we can still see cases where the time and energy do not exactly correlate. In those cases the energy profiler is able to give additional insight beyond the regular performance profiles.

In summary, there is still no way to accurately estimate the energy consumption of each function or line of code. The present trade-off between accuracy of energy estimation and the effect on the execution time needs to be replaced with a viable solution. Furthermore, accurate energy estimates are also difficult to achieve because the energy consumption depends on what other applications are running at the same time. Also, another program competing for the same memory and cache resources will influence the energy consumption. However, even indicative estimates are likely to be helpful.

4.2.2. Software architecture implications on energy-efficiency

The key to computational efficiency is the quality of used algorithms. However, even with an optimal algorithm, its implementation must follow the constraints of the run-time environment to maximize performance and energy efficiency. Especially, the way in which the software accesses data stored in the memory has a significant impact on performance and energy spending.

At the core level, the processor executes calculations on values stored in its registers. Fetching a value from the RAM to a register takes approximately 100 clock cycles. Therefore multiple levels of faster cache memory are utilized to prefetch data from the slow RAM to a faster cache, so that the register load can happen in only around 10 clock cycles. To produce this speed-up, cache prefetching algorithms must correctly predict future data access needs. If a value that is not in cache is needed, the cache needs to prefetch the contents of memory in the accessed region and the processor will stall waiting for the memory load to complete.

The speed disparity between registers, cache, and RAM has a major effect on the computing speed, and, through speed, on energy consumption. While the effect of object-oriented programming to memory use has already been investigated and well understood in the 1990s [64], the problem is becoming even more visible today, since the difference between the computing speed and memory access speed is significantly larger in modern processors than it was previously. Therefore, the efficient use of the cache is even more important today [65].

Optimizing cache prefetching involves designing data layout in memory to match processor cache design. When a cache miss occurs, the cache controller fetches a block of memory around the accessed location. The size and alignment of the block, called a cache line, depends on the processor. A typical cache line is 32 words, or full register sized data items, starting from the preceding memory location whose address is divisible by the cache line size. Data layout in memory should therefore be consecutive words, packaged in cache line size sequences that are also referenced with high locality by the functional instructions in temporal proximity.

To quantify the importance of proper order of accessing memory, we performed a simple test with a large table of C++ instances [66]. When we accessed the instances (to perform a XOR operation) in the same order as they were created, the speed was around 4x faster than if we accessed the same instances in a random order. Accessing the instances in exactly the reverse order from their creation fell between these two extremes.

To further minimize computing effort, we notice that some functional instructions are more efficient than others, but these instructions can only be used when data is in specific registers and in a specific format. For example, a matrix multiplication can be implemented using traditional looping over a two-dimensional array, or by preloading the array to suitable registers and using a single-instruction-multiple-data (SIMD) instruction. Compared to classical looping over array, SIMD instructions have some set-up cost as multiple data items need to be loaded in suitable registers, but their execution can be an order of magnitude more efficient both in time and energy consumed. When data is suitably prepared so that SIMD instructions can be utilized, multiple results can be calculated per clock cycle.

There are also some special challenges related to scientific computing. For example, a problem with floating point numeric computing is that the results can be different depending on compiler version and operating system. For instance, Baloolan [67] compared codes for electromagnetic wave simulation and found code compiled with the same compiler produced different results when running in Windows and Linux environments or in different Windows XP versions. The underlying reason is the floating point calculations way they have been implemented [68]. Even if there are standards for floating point encodings and functionality [69] subtly differences in e.g. computation order or processor use can still cause different numeric results from the same computing [70]. Hayes et al. noticed a similar problem when running the same simulation software in two different hardware [71]. The unfortunate consequence of this is that taking into use newer, more optimal compilers is not possible. The big investment over the years to ensure that the physics computing codes produce the correct answers would be partially wasted and the results of the new calculations could not be trusted without extensive analysis and verification. Therefore the key physics software is stuck with the old compiler versions which are not able to take advantage of advanced possibilities of modern processor architectures. This is one instance of risk avoidance, which is also visible in other software improvement actions. The threshold is high to modify old, thoroughly tested and validated code only for better performance.

In the CERN case, a lot of the code is old and developed at a time when the processor architecture did not support the features of modern processors described above. In a typical use case, the data in ROOT format files is used to construct C++ objects in RAM [72]. The object oriented design of the low-level data manipulation software enables high-level abstractions in application code, provides clear interfaces between components, and thereby supports cooperative work. Unfortunately, it also distributes the data items in non-consecutive locations in memory, in a format that is not directly suitable for use by SIMD instructions. The data items are not packed in cache-efficient blocks and, due to the data hiding pattern, the algorithm implementations can not consider locality of reference for data item accesses. The unfortunate result is that the processor spends a lot of time waiting for memory accesses and efficient SIMD instructions can not be used. The challenge is to write the software providing a high-level abstraction interface for the users, while still handling data efficiently in memory, and adapting the handling to the type of processor where it is scheduled to run.

It is well understood that software tends to increase in complexity as a result of late-lifecycle changes making it harder to modify and forcing an architectural restructuring over time. Williams et al. [73] review the work on this phenomena sometimes called code decay, software aging, or architectural degeneration. Most of the work deals with how new and changing functional requirements cause code complexity to increase. In early and influential work on software evolution Lehman [74] observes that hardware change is one of the reasons for code decay. However, hardware change and how to develop software which is able to improve efficiency by using the novel opportunities of new hardware technologies has received less attention that changes arising from functional requirements. The study by Kuusela [75] is an attempt to consider how to create an architecture that is adaptive to hardware changes. However, in most cases the adaptation to hardware is seen as a way to make the system work in another hardware platform.

Our observation is that bringing the software to work on a new platform can sometimes be easy but more work is needed to get the software work efficiently with the opportunities and characteristics of new hardware platforms. Therefore, an ideal architecture should be resilient both to new functional requirements as well as to new hardware opportunities. Brown at al. [76] discuss how changes done for improvements in one dimension create an architectural debt in other dimensions. The debt, as seen in additional complexity or non-optimal performance, has to be "paid back" as a later development step. We think that this theory applies well to the CERN case where the functional and other requirements lead the development. The performance problems initially felt to be secondary but as time goes by they become more visible. The same is even more true for the energy spending of the software. It is only recently that the electricity spending has become a bottleneck.

In conclusion, an energy efficient software architecture must provide the interfaces that users need to define their application level logic, while providing an efficient data manipulation and processing layer underneath the high-level interface. This data manipulation layer either needs to be sufficiently adaptable to runtime environment, or variants of the layer needs to be available for a smart scheduler that is able to choose the correct software version for the current runtime environment.

In the case of CERN, adapting the current software towards a more energy efficient architecture is a major challenge because of the large amount of code and the long legacy in software development. Even if we understand what the energy-optimal architecture would be, reaching that goal is a slow process. From the software design perspective the direction of the required changes is clear, but from a software engineering perspective it is not obvious how the task could be achieved. Although software re-engineering has been widely studied, for example in several EU funded projects such as [77], it is a time consuming and complex process. Empirical study of industrial software refactoring [78] indicates that developers tend to fix concrete coding issues and improve the maintainability of their code. Because energy-efficiency is largely invisible to the developers, it is unlikely that it would be considered in ordinary refactoring. Explicit targets and resources for its improvement would therefore be needed.

4.3. Data Center Management View

In this section, we change our perspective beyond the efficiency of a single piece of software and investigate the problem of how to process a large set of jobs efficiently. While most of the existing work on high performance computing focuses on optimizing the processing time of individual computing jobs, we now try to optimize the energy consumption and the total processing time of the set of jobs by choosing an optimal scheduling policy. In what follows, we assume that jobs (the programs to be executed) can be treated as black boxes without detailed understanding of the internals of them. This means we also change the perspective from what software developer can do, to ideas, which can be applied by the workload management at the data center.

The computing clusters can be seen as production resources processing jobs consisting of numerous tasks [79]. The tasks can be processed by different resources, and finally the jobs are assembled together to be delivered back to the cluster customers. Operating such a cluster has its own cost structure related to capital invested, energy consumed, maintenance work, and facility related costs. Based on this, we can apply operations management principles used in manufacturing such as minimizing the lead time, waste, and inventory, and maximizing the utilization and output to improve productivity and minimize the energy consumption.

Following the operations management approach [52], the efficient management of computing resources is based on the following principles:

1. Modeling the computing system as a manufacturing unit applying well-known operations management theory (e.g. [80, 79]).
2. Measuring performance using meaningful indicators (e.g. [81]).
3. Managing the resources in a way that maximizes the energy efficiency and output (e.g. [48, 82]).

In the following sub sections, we discuss these principles in more detailed.

4.3.1. Modeling Energy Efficiency of Scientific Data Center

Since a data center or a cluster closely resembles an industrial production unit, we can apply operations management principles, such as minimizing the lead time, waste, and inventory and maximizing the utilization and output, used in manufacturing to improve computing cluster productivity and minimize its energy consumption. However, not all principles from manufacturing directly fit in computing clusters. For example the law of variation, stating that all variability reduces efficiency, is not actually true in computing [79].

In practice, our approach is based on collecting and analyzing log data on a computing cluster at CERN [79]. When analyzing the data, we noticed, for example, that wall clock processing times of jobs are 15 to 50% longer than the actual CPU times. Since the cluster configuration was set to process one job per CPU core, this means that there is a bottleneck slowing down the computing process. The most obvious bottleneck is I/O access to the disk or network. Furthermore, the memory utilization of jobs was only about half of the CPU utilization rate. One reason for this is an irregular memory utilization curve of particle physics jobs. Based on this observation, we assumed that reasonable overloading can help and throughput can be improved and electricity consumption reduced by increased multitasking, i.e. processing more than one task per CPU core in parallel, and mixing heterogeneous tasks in parallel while multitasking.

4.3.2. Technologies for Measuring Energy Efficiency

There are several indicators for measuring the (energy) efficiency of data centers [83, 84]. Probably the most commonly used is the Power Usage Effectiveness (PUE), measuring the ratio of power consumed by ICT equipment compared to the total power of the data center [85]. It measures the energy efficiency of the overall infrastructure, but it does not indicate whether ICT services run efficiently. Therefore, other performance indicators have been suggested, for example for measuring useful work per unit of energy [86].

Since all components of the computer use energy even when idle, and they also have an optimal utilization rate, the optimal case would be using all components in their energy-optimal rate all the time. For most of the components the highest energy efficiency is reached when utilization is high but not too close to 100%. For this purpose we [81] have developed the energy-efficient utilization indicator (EEUI). This indicator gives a weighted average of energy efficiency for all components of a computing unit (single server to data center). Using EEUI as a guide to run computing resources optimizes both throughput and energy efficiency. However, a challenge using the EEUI indicator is the need for power measurements in a server or even server component level. Fortunately, measuring, or estimating, power values is becoming possible through technologies such as RAPL [87], which was discussed more thoroughly in Section 4.2.1.

4.3.3. Technologies for Energy-efficient Management of Computing Resources

This final step in our approach is probably the most difficult one answering to the following question: How should workloads be scheduled for different computing servers so that the processing remains energy-efficient and gives the optimal throughput? Good allocation involves several practical issues. For example, the number of parallel tasks in a server is limited by physical resources, especially by the main memory. Another challenge is that the workload is not stable but constantly changes. Thus, static scheduling would lead to a non-optimal solution [82, 48].

For scheduling, we can use the following criteria: 1) a task uses a minimal amount of energy, and 2) the total throughput is maximal. These goals can be contradictory but usually the maximal or near maximal throughput also gives the minimal energy consumption per processed task. As far as the clusters are concerned, the problem can be divided into two independent steps assuming that cluster nodes are homogeneous:

1. Finding the optimal load combination for the computing node.

2. Scheduling jobs to the computing nodes in such a way that all computing nodes are as close as possible to the optimum state.

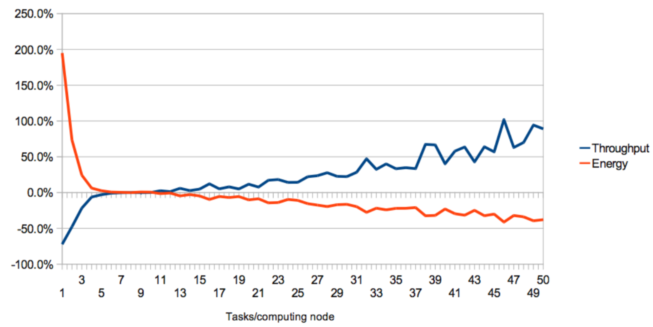

Figure 2: An example how the number of parallel jobs affects throughput and energy efficiency [48].

The efficiency and energy consumption of computing servers depends on the load level as illustrated in Figure 2 when CMS data analysis application [88] was run. If the workload is heavy enough to keep all servers busy, the optimal number of parallel jobs running on a computing server can be estimated, for example, by the fuzzy logic-based algorithm [82]. The algorithm dynamically adapts the memory threshold for submission of jobs based on the overall load. In this way, it is possible to keep memory consumption stable with different workloads while achieving significantly higher throughput and energy-efficiency than using traditional fixed number of jobs or fixed memory threshold approaches. The test results showed that at best, the memory-based scheduling with fuzzy tuning improved throughput by over 150% and reduced energy consumption around 50% compared to one job per CPU core scheduling. Mixing CPU and I/O intensive jobs improved both throughput by 10-20% and energy efficiency by 5-10% compared to running single types of jobs only.

For situations where the intensity of the workload fluctuates a lot, we can apply dynamic workload management, meaning that the allocation of computing tasks running on virtual machines (VM), to servers can be changed during processing. For this purpose, we developed a methodology and prototype system for the load based management of virtual machines in an OpenStack cluster [89]. The novelty of this load balancer is to use specialized, energy-efficient servers (park servers) for storing idle virtual machines in addition to the active monitoring of resources and the load based active management of VMs in the physical cluster. Since the park server is used to store idle VMs its over-commit ratio can be high while actual computing servers used by active VMs have one-to-one mapping of virtual to physical resources. The load balancing algorithm is based on live migrated VMs as follows: 1) move idle VMs to the park server; 2) move a VM from the park server to a computing node when it becomes active; 3) consolidate active VMs within the cluster to minimize the number of active servers. Our tests with real hardware indicated potential energy savings of 9% to 48% and the simulation results showed that in large installations the energy efficiency could be improved by up to 40%. Naturally, the energy savings of the active management greatly depend on the workload and its fluctuations.

4.4. From Energy Saving to Proper Timing of Energy Use

An important observation in energy-efficient computing is that saving energy is not the only goal. Often it is more important to use the energy at the right time in the right place. Demand management [90] is a hot topic in the electricity sector and scientific computing could be a component that brings additional elasticity to the demand. In electricity grids the supply and demand have to meet at each time point and scientific computing could be one way to very rapidly react to imbalances in supply and demand.

Especially with renewable energy sources the energy supply and price can vary a lot. There have been extreme cases for example in Germany and Denmark where the price of electricity has been negative. The obvious but not very useful way to take advantage of this is to time the computing so that it is done when the electricity is free or cheap. The problem with this approach is that scientists are eager to get results and therefore they are not willing to tolerate very long wait times. Moreover, it is questionable whether an investment in computer hardware that is only used when cheap energy is available, makes enough economic sense these days when the electricity price is generally low.

A set of more interesting ideas is based on the observation that often it is easier to move computing tasks than to move electricity. Therefore the goal could be to move the computing tasks to places which have abundant energy. Researchers have already explored cases of optimally moving the computing to different countries or regions depending on the variable electricity prices (see e.g. [91, 92, 93]).

In our own work, we have studied the idea of spending the spare energy of a house equipped with renewable energy sources for computing as described in [94]. The development is based on the fact that it is not straightforward to decide what to do with the extra energy generated, for example by solar panels. This issue can exist for example on a sunny day when the home is empty. The conventional possible alternatives include selling spare energy to an electricity company, storing it locally, or in the extreme, wasting it completely. However, all of these have they shortcomings. Therefore, we have developed a computing server which uses the spare electricity in a home to perform computing. This allows the homeowner to earn money by selling computing services instead of selling electricity back to the grid. Scientific computing workloads could be excellent candidates for such calculations since they can often tolerate the delays that arise because of the unpredictable availability of the computing services.

5. Towards a More Energy-Efficient Future

In this section we discuss ideas and observations for improving energy efficiency in scientific computing and computing in general. First, it is important to have a holistic view. Different actors, as we have described in this paper, have different viewpoints and different sets of options to influence the energy-efficiency of computing. In addition to focusing on improving a single perspective it is essential to understand how the different views coexists and influence each other. It is also essential to analyze the computing system as a whole and to find the most promising opportunities to be implemented. Taking advantage of expected benefits requires work: it includes making changes to the existing legacy systems, developing new ways of working, and inventing new methods to manage computing. A large part of this work can be done in parallel. Only through gradual progress on multiple frontiers, we can gain concrete advances, since any single breakthrough alone is unlikely to have a dramatic effect but accumulating the effects of several improvements will be important.

In this paper, we introduced three groups of methods for improving energy efficiency of scientific computing presented in Section 4: 1) promoting green values to the users, 2) offering tools such as energy profiling and energy-aware architectures for developers, and 3) monitoring and optimizing tools and methods for data center operators. Although these methods have similarities to proven techniques used in traditional manufacturing such as the continuous improvement paradigm [95] emphasizing visual control and the theory of constraints focusing on process bottlenecks [96],our examples also show that a scientific computing system should not be managed as an automated factory but it is an interactive system among the shareholders. For example by providing the user with feedback on the resource usage and energy consumption along with the yield (processed jobs in time per available processing time) may change user behavior towards a more resource intensive and energy efficient use of the facility. At the same time, a developer who realizes through visual reporting which functions of the software are most used, may direct his efforts to further develop those key functions to be more efficient and faster. This follows closely the bottleneck approach, which aims to remove process constraints to reach a swift, even flow [97] in the value adding process. Moreover, data center management could be directed, for example, by showing key performance indicators on resource utilization. According to operations management principles, the practical version of the law of utilization indicates that lead times and work in process will increase radically when a threshold level is passed [52].

On the other hand, we must remember that, as operational entities, data centers radically differ from the traditional supplier-customer relationships in manufacturing and service industries. In these industries partnerships are critical to success, meaning that in supply chains suppliers and customers share information, plans and the additional business environment related knowledge to improve efficiency and responsiveness. In the computing center context the triangle of stakeholders, namely the users, developers, and data center operators seldom discuss with each other. This can reduce efficiency, since scientific computing systems are often custom-made systems which would benefit information sharing among the stakeholders. There is no genuine partnership or unity, which would on its own improve the situation like in supplier-customer quality circles [98], where different parties systematically and with objective methodology improve the overall system performance.

As we have discussed above there are still many open issues in the quest for greater energy efficiency. While our work and that of other scientists in the area is touching some of these issues, a fair number of unsolved problems exist. One important outcome of our work in this area is a clearer understanding of important research problems to tackle. We have collected the major open technical challenges covered in this article in Table 1. The first column of the table describes the challenge, the second one explains its importance, the third one indicates possible difficulties and the last one gives examples on studies towards possible solutions.

Besides the somewhat well understood possibilities there are naturally a multitude of other future possibilities such as photonic, biomolecular, and quantum computing (see e.g. [116, 117, 118]).

While each of the ideas discussed in this article works separately the best results can of course be attained if they would be used in an integrated fashion. However, this is very challenging to achieve in practice. The massive amount of legacy software would need to be adapted to the new environment. Considering the different aspects would make software development harder, and in scientific computing most of the software is developed by scientists who are experts in physics but not necessarily in software development. Furthermore, considering the amount of time that is needed to develop and deploy integrated solutions the world may change and new, unanticipated options, may become available.

Therefore a perfect future solution is an illusion. It is good to have a vision and take steps in that direction but at the same time admit that in real-life reaching the ideal target may be challenging.

6. Conclusions

Improving the energy efficiency of computing systems is becoming more and more difficult since simple technical solutions and replacing the computing hardware with next generation models does not offer as much improvement it used to. Therefore, other solutions are needed and, for example, optimizing parts other than hardware of computing systems still offer a large potential for improvement. In this paper, we studied how the computing system can be seen as a value chain of three main roles: users, software developers, and data center administrators. Using this grouping, we introduced a set of potential technologies and methodologies for improving the energy efficiency of each of these roles. This makes it possible to recognize the bottlenecks of the system and focus on removing those bottlenecks.

The high energy physics community is a global family of researchers and experts working on the experiments to detect new particles generated by the colliding beams in the LHC. To reveal the elementary levels of matter from these collisions requires massive computing infrastructure and multitude of skills to master. Therefore, information technology has always played an essential role [119]. The community works in distributed manner through globally spread institutes, which all share their own political and computational habits, but still striving towards common scientific goals. The spirit is similar to open source societies, where transparency and rough consensus drives the collaboration. Complex system development requires architectural control, which is balanced by technological memorandum of understanding to which developers commit and which is continuously scrutinized through a social network and frequent collaboration meetings. This facilitates also global and fast dissemination of different results produced by the community.

Table 1: Summary of selected technical challenges towards more energy-efficient future

Challenge	**Motivation**	**Difficulty**	**Studies towards possible solutions**
Accurate and non-intrusive way to estimate energy spending at function or line-of-code level	Making it possible for the developer to know where in the program code the energy is consumed.	Too course granularity of energy measurements	[60, 99]
Methodology to see how energy consumption varies when code is modified	In this way, alternative designs could be tested and 'energy bugs' could be recognized	Software energy consumption is influence by external factors like other applications running at the same time	[100]
Measuring energy consumption of a computing job	Users would be more motivated for saving energy if its usage is visible	Other applications and utilization of the resources make it difficult to map the energy to a single application	[101]
Automatic methods for taking benefit on heterogeneous hardware	Code could benefit energy optimizations of different architectures and applications run in the optimal hardware	Although relatively easy for similar architectures, using e.g. GPUs requires writing memory management etc.	[102, 89, 103, 104, 105, 106]
Appropriate software architectures, which have proper balance between efficiency and other needs	Software systems tend to live and evolve for decades. Easy adaptation to new needs is essential. At the same time efficiency is important when amount of analyzed data grows	Anticipating the aspects that are relevant for future is hard.	[107, 108, 109]
Incentives for scientists to develop energy efficient code	Prices of ICT components are continuously decreasing while electricity production remains a bottleneck	Little tradition in teaching and developing energy efficient code and systems in scientific computing	[110, 111, 112]
Large-scale energy driven distributed computing	Bring computing to locations with plentiful energy (with renewable production energy availability will increasingly fluctuate)	Grid and cloud systems not designed for easy geographic migration of computing	[113, 93, 114, 115, 94]

References

[1] I. Bird, F. Carminati, R. Mount, B. Panzer-Steindel, J. Harvey, I. Fisk, B. Kersevan, P. Clarke, M. Girone, P. Buncic, et al., Update of the computing models of the wlcg and the lhc experiments, Tech. rep. (2014).

[2] J. Shiers, The worldwide LHC computing grid (worldwide LCG), Computer physics communications 177 (1) (2007) 219–223.

[3] I. Dumitru, I. Fagarasan, S. Iliescu, Y. H. Said, S. Ploix, Increasing energy efficiency in data centers using energy management, in: Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on, IEEE, 2011, pp. 159–165.

[4] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, P. Demeester, Trends in worldwide ict electricity consumption from 2007 to 2012, Computer Communications 50 (2014) 64–76.

[5] H. Sutter, The free lunch is over: A fundamental turn toward concurrency in software, Dr. Dobbs journal 30 (3) (2005) 202–210.

[6] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, spring joint computer conference, ACM, 1967, pp. 483–485.

[7] S. Herbert, D. Marculescu, Analysis of dynamic voltage/frequency scaling in chip-multiprocessors, in: Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on, IEEE, 2007, pp. 38–43.

[8] S. Cho, R. Melhem, Corollaries to amdahl's law for energy, IEEE Computer Architecture Letters 7 (1) (2008) 25–28.

[9] R. T. Haftka, L. T. Watson, Multidisciplinary design optimization with quasiseparable subsystems, Optimization and Engineering 6 (1) (2005) 9–20. `doi:10.1023/B:OPTE.0000048534.58121.93`.
URL `http://dx.doi.org/10.1023/B:OPTE.0000048534.58121.93`

[10] M. Dayarathna, Y. Wen, R. Fan, Data center energy consumption modeling: A survey, IEEE Communications Surveys & Tutorials 18 (1) (2016) 732–794.

[11] J. Shuja, A. Gani, S. Shamshirband, R. W. Ahmad, K. Bilal, Sustainable cloud data centers: a survey of enabling techniques and technologies, Renewable and Sustainable Energy Reviews 62 (2016) 195–214.

[12] S. Mazumdar, M. Pranzo, Power efficient server consolidation for cloud data center, Future Generation Computer Systems 70 (2017) 4–16.

[13] M. Karpowicz, E. Niewiadomska-Szynkiewicz, P. Arabas, A. Sikora, Energy and power efficiency in cloud, in: Resource Management for Big Data Platforms, Springer, 2016, pp. 97–127.

[14] H. Rong, H. Zhang, S. Xiao, C. Li, C. Hu, Optimizing energy consumption for data centers, Renewable and Sustainable Energy Reviews 58 (2016) 674–691.

[15] M. Zakarya, L. Gillam, Energy efficient computing, clusters, grids and clouds: A taxonomy and survey, Sustainable Computing: Informatics and Systems 14 (2017) 13–33.

[16] M. Uddin, Y. Darabidarabkhani, A. Shah, J. Memon, Evaluating power efficient algorithms for efficiency and carbon emissions in cloud data centers: A review, Renewable and Sustainable Energy Reviews 51 (2015) 1553–1563.

[17] P. Jatesiktat, P. Uthayopas, Efficient power management on a cloud system using two phases power convergence algorithm, in: Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on, IEEE, 2012, pp. 399–404.

[18] Q. Zhao, C. Xiong, C. Yu, C. Zhang, X. Zhao, A new energy-aware task scheduling method for data-intensive applications in the cloud, Journal of Network and Computer Applications 59 (2016) 14–27.

[19] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, et al., Recent advancements in resource allocation techniques for cloud computing environment: a systematic review, Cluster Computing (2016) 1–45.

[20] H. Kataoka, S. Nakamura, T. Enokido, M. Takizawa, Simple energy-aware algorithms for selecting a server in a scalable cluster, in: Advanced Information Networking and Applications Workshops (WAINA), 2017 31st International Conference on, IEEE, 2017, pp. 146–153.

[21] A. Sawada, S. Nakamura, D. Duolikun, T. Enokido, M. Takizawa, Simple energy-aware algorithms to selecting a server for storage and computation processes in a cluster, in: International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Springer, 2017, pp. 98–109.

[22] H. Kataoka, A. Sawada, D. Duolikun, T. Enokido, M. Takizawa, Energy-aware algorithms to select servers in scalable clusters, in: Complex, Intelligent, and Software Intensive Systems (CISIS), 2016 10th International Conference on, IEEE, 2016, pp. 308–315.

[23] K. Gai, M. Qiu, H. Zhao, L. Tao, Z. Zong, Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing, Journal of Network and Computer Applications 59 (2016) 46–54.

[24] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi, F. Li, Minimizing sla violation and power consumption in cloud data centers using adaptive energy-aware algorithms, Future Generation Computer Systems.

[25] F. Tao, Y. Feng, L. Zhang, T. W. Liao, Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling, Applied Soft Computing 19 (2014) 264–279.

[26] G. A. Beletsioti, G. I. Papadimitriou, P. Nicopolitidis, Energy-aware algo-

rithms for ip over wdm optical networks, Journal of Lightwave Technology 34 (11) (2016) 2856–2866.

[27] R. Wang, Z. Jiang, S. Gao, W. Yang, Y. Xia, M. Zhu, Energy-aware routing algorithms in software-defined networks, in: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a, IEEE, 2014, pp. 1–6.

[28] J. Sung, D. Lee, Y. Bang, J. Lee, J.-K. K. Rhee, Energy-aware algorithms for network-assisted device-to-device content delivery networks, in: Information and Communication Technology Convergence (ICTC), 2015 International Conference on, IEEE, 2015, pp. 469–471.

[29] E. Jagroep, J. Broekman, J. M. E. van der Werf, S. Brinkkemper, P. Lago, L. Blom, R. van Vliet, Awakening awareness on energy consumption in software engineering, in: Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Society Track, IEEE Press, 2017, pp. 76–85.

[30] L. A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer 40 (12).

[31] Y. Chen, S. Alspaugh, D. Borthakur, R. Katz, Energy efficiency for large-scale mapreduce workloads with significant interactive analysis, in: Proceedings of the 7th ACM european conference on Computer Systems, ACM, 2012, pp. 43–56.

[32] J. Higgins, V. Holmes, C. Venters, Orchestrating docker containers in the hpc environment, in: International Conference on High Performance Computing, Springer, 2015, pp. 506–513.

[33] C. Boettiger, An introduction to docker for reproducible research, ACM SIGOPS Operating Systems Review 49 (1) (2015) 71–79.

[34] D. Salomoni, I. Campos, L. Gaido, G. Donvito, M. Antonacci, P. Fuhrman, J. Marco, A. Lopez-Garcia, P. Orviz, I. Blanquer, et al., Indigo-datacloud:

foundations and architectural description of a platform as a service oriented to scientific computing, arXiv preprint arXiv:1603.09536.

[35] D. Hufnagel, The CMS Tiero goes cloud and grid for LHC run 2, in: Journal of Physics: Conference Series, Vol. 664, IOP Publishing, 2015, p. 032014.

[36] T. Bell, B. Bompastor, S. Bukowiec, J. C. Leon, M. Denis, J. van Eldik, M. F. Lobo, L. F. Alvarez, D. F. Rodriguez, A. Marino, et al., Scaling the cern openstack cloud, in: Journal of Physics: Conference Series, Vol. 664, IOP Publishing, 2015, p. 022003.

[37] S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A. Sirunyan, W. Adam, T. Bauer, T. Bergauer, H. Bergauer, M. Dragicevic, J. Erö, et al., The cms experiment at the cern lhc.

[38] G. Aad, E. Abat, J. Abdallah, A. Abdelalim, A. Abdesselam, O. Abdinov, B. Abi, M. Abolins, H. Abramowicz, E. Acerbi, et al., The atlas experiment at the cern large hadron collider, Journal of Instrumentation 3 (8) (2008) S08003–S08003.

[39] A. A. Alves Jr, L. Andrade Filho, A. Barbosa, I. Bediaga, G. Cernicchiaro, G. Guerrer, H. Lima Jr, A. Machado, J. Magnin, F. Marujo, et al., The lhcb detector at the lhc, Journal of instrumentation 3 (08) (2008) S08005.

[40] A. Tapper, C. Collaboration, et al., The CMS level-1 trigger for LHC run ii, in: Proceedings of the 38th International Conference on High Energy Physics (ICHEP2016). 3-10 August 2016. Chicago, USA. Online at http://pos. sissa. it/cgi-bin/read er/conf. cgi? confid= 282, id. 242, 2016.

[41] V. Gori, The CMS high level trigger, in: International Journal of Modern Physics: Conference Series, Vol. 31, World Scientific, 2014, p. 1460297.

[42] M. Lamanna, The lhc computing grid project at cern, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 534 (1) (2004) 1–6.

[43] H. Stockinger, F. Donno, E. Laure, S. Muzaffar, P. Kunszt, P. Millar, Grid data management in action: Experience in running and supporting, in: in the EU DataGrid Project, in: Computing in High Energy Physics (CHEP 2003), La Jolla, Citeseer, 2003.

[44] E. Laure, B. Jones, Enabling grids for e-science: The egee project, Grid computing: infrastructure, service, and applications (2009) 55.

[45] R. P. Taylor, J. Hover, C. Cordeiro, P. Love, A. Mcnab, T. Kouba, A. Di Girolamo, R. Sobie, J. Schovancova, Consolidation of cloud computing in atlas, Tech. rep., ATL-COM-SOFT-2016-071 (2016).
URL http://cds.cern.ch/record/2216950/files/ATL-SOFT-SLIDE-2016-657.pdf

[46] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, A. Secret, The world-wide web, Commun. ACM 37 (8) (1994) 76–82. doi:10.1145/179606.179671.
URL http://doi.acm.org/10.1145/179606.179671

[47] L. A. Barroso, J. Clidaras, U. Hölzle, The datacenter as a computer: An introduction to the design of warehouse-scale machines, Synthesis lectures on computer architecture 8 (3) (2013) 1–154.

[48] T. Niemi, J. Kommeri, A.-P. Hameri, Improving energy-efficiency of scientific computing clusters, Energy-aware Systems and Networking for Sustainable Initiatives.

[49] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, et al., A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing 98 (7) (2016) 751–774.

[50] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, et al., An overview of

energy efficiency techniques in cluster computing systems, Cluster Computing 16 (1) (2013) 3–15.

[51] C. L. Belady, C. G. Malone, Metrics and an infrastructure model to evaluate data center efficiency, in: ASME 2007 InterPACK Conference collocated with the ASME/JSME 2007 Thermal Engineering Heat Transfer Summer Conference, American Society of Mechanical Engineers, 2007, pp. 751–755.

[52] W. J. Hopp, M. L. Spearman, Factory physics, Waveland Press, 2011.

[53] I. Sfiligoi, T. Martin, B. Bockelman, D. Bradley, F. Würthwein, Minimizing draining waste through extending the lifetime of pilot jobs in grid environments, in: Journal of Physics: Conference Series, Vol. 513, IOP Publishing, 2014, p. 032089.

[54] P. Love, Analysis of empty atlas pilot jobs, Tech. rep., ATL-COM-SOFT-2016-122 (2017).

[55] L. Steg, J. W. Bolderdijk, K. Keizer, G. Perlaviciute, An integrated framework for encouraging pro-environmental behaviour: The role of values, situational factors and goals, Journal of Environmental Psychology 38 (2014) 104–115.

[56] B. W. T. Kelvin, Electrical units of measurement (May 1889).
URL `https://todayinsci.com/QuotationsCategories/M_Cat/Measurement-Quotations.htm`

[57] T. DeMarco, Controlling software projects: Management, measurement, and estimates, Prentice Hall PTR, 1986.

[58] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, S. Moore, Measuring energy and power with papi, in: Parallel Processing Workshops (ICPPW), 2012 41st International Conference on, IEEE, 2012, pp. 262–268.

[59] C.-H. Hsu, S. W. Poole, Power measurement for high performance computing: State of the art, in: Green Computing Conference and Workshops (IGCC), 2011 International, IEEE, 2011, pp. 1–6.

[60] D. Abdurachmanov, P. Elmer, G. Eulisse, R. Knight, T. Niemi, J. K. Nurminen, F. Nyback, G. Pestana, Z. Ou, K. Khan, Techniques and tools for measuring energy efficiency of scientific software applications, in: Journal of Physics: Conference Series, Vol. 608, IOP Publishing, 2015, p. 012032.

[61] G. Eulisse, L. Tuura, Igprof profiling tool, Computing in High Energy Physics and Nuclear Physics (2004) 665.

[62] K. N. Khan, F. Nybäck, Z. Ou, J. K. Nurminen, T. Niemi, G. Eulisse, P. Elmer, D. Abdurachmanov, Energy profiling using igprof, in: Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, IEEE, 2015, pp. 1115–1118.

[63] M. Hähnel, B. Döbel, M. Völp, H. Härtig, Measuring energy consumption for short code paths using rapl, ACM SIGMETRICS Performance Evaluation Review 40 (3) (2012) 13–17.

[64] B. Calder, D. Grunwald, B. Zorn, Quantifying behavioral differences between c and c++ programs, Journal of Programming languages 2 (4) (1994) 313–351.

[65] T. Albrecht, Pitfalls of object oriented programming, Proceedings of Game Connect: Asia Pacific (GCAP).

[66] O. Kiljunen, Memory performance of object-oriented programming, Tech. rep., Aalto University (2017).

[67] S. Baboolal, Java/c++/fortran cross-platform performance and consistency issues for large simulation codes, in: Proceedings of the 10th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2010, CMMSE, 2010, pp. 27–30.

[68] D. Goldberg, What every computer scientist should know about floating-point arithmetic, ACM Computing Surveys (CSUR) 23 (1) (1991) 5–48.

[69] D. Zuras, M. Cowlishaw, A. Aiken, M. Applegate, D. Bailey, S. Bass, D. Bhandarkar, M. Bhat, D. Bindel, S. Boldo, et al., Ieee standard for floating-point arithmetic, IEEE Std 754-2008 (2008) 1–70.

[70] N. Whitehead, A. Fit-Florea, Precision & performance: Floating point and ieee 754 compliance for nvidia gpus, rn (A+ B) 21 (1) (2011) 18749–19424.

[71] M. Hayes, F. Schmidt, E. McIntosh, The influence of computer errors on dynamic aperture results using sixtrack, Tech. rep., CERN-LHC-Project-Note-309 (2003).

[72] R. Brun, F. Rademakers, Rootan object oriented data analysis framework, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 389 (1-2) (1997) 81–86.

[73] B. J. Williams, J. C. Carver, Characterizing software architecture changes: A systematic review, Information and Software Technology 52 (1) (2010) 31–51.

[74] M. M. Lehman, Programs, life cycles, and laws of software evolution, Proceedings of the IEEE 68 (9) (1980) 1060–1076.

[75] J. Kuusela, Architectural evolution, in: Software Architecture, Springer, 1999, pp. 471–478.

[76] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, et al., Managing technical debt in software-reliant systems, in: Proceedings of the FSE/SDP workshop on Future of software engineering research, ACM, 2010, pp. 47–52.

[77] REPARA reengineering and enabling performance and power of applications, http://repara-project.eu/, accessed: 2017-09-01.

[78] G. Szőke, G. Antal, C. Nagy, R. Ferenc, T. Gyimóthy, Empirical study on refactoring large-scale industrial systems and its effects on maintainability, Journal of Systems and Software 129 (2017) 107–126.

[79] F. Abaunza, V. Chavez-Demoulin, A.-P. Hameri, T. Niemi, Do flow principles of operations management apply to computing centres?, Production Planning & Control 26 (4) (2015) 249–264.

[80] A.-P. Hameri, T. Niemi, Applying operations management principles on optimisation of scientific computing clusters, in: Rapid Modelling and Quick Response, Springer, 2011, pp. 105–117.

[81] F. Abaunza, A.-P. Hameri, T. Niemi, EEUI: A new measure to monitor and manage energy efficiency in data centers, International Journal of Productivity and Performance Management In press.

[82] T. Niemi, A.-P. Hameri, Memory-based scheduling of scientific computing clusters, The Journal of Supercomputing 61 (3) (2012) 520–544.

[83] A. Kipp, T. Jiang, M. Fugini, I. Salomie, Layered green performance indicators, Future Generation Computer Systems 28 (2) (2012) 478–489.

[84] L. Wang, S. U. Khan, Review of performance metrics for green data centers: a taxonomy study, The journal of supercomputing 63 (3) (2013) 639–656.

[85] C. Belady, A. Rawson, J. Pfleuger, T. Cader, Green grid data center power efficiency metrics: Pue and dcie, Tech. rep., Technical report, Green Grid (2008).

[86] B. Schaeppi, T. Bogner, A. Schloesser, L. Stobbe, M. D. de Asuncao, Metrics for energy efficiency assessment in data centers and server rooms, in: Electronics Goes Green 2012+(EGG), 2012, IEEE, 2012, pp. 1–6.

[87] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, S. Moore, Measuring energy and power with papi, in: 2012 41st International Conference on Parallel Processing Workshops, 2012, pp. 262–268. doi:10.1109/ICPPW.2012.39.

[88] F. Fabozzi, C. D. Jones, B. Hegner, L. Lista, Physics analysis tools for the CMS experiment at LHC, IEEE Transactions on Nuclear Science 55 (6) (2008) 3539–3543.

[89] J. Kommeri, T. Niemi, J. K. Nurminen, Energy efficiency of dynamic management of virtual cluster with heterogeneous hardware, The Journal of Supercomputing (2016) 1–23.

[90] P. Palensky, D. Dietrich, Demand side management: Demand response, intelligent energy systems, and smart loads, IEEE transactions on industrial informatics 7 (3) (2011) 381–388.

[91] N. Buchbinder, N. Jain, I. Menache, Online job-migration for reducing the electricity bill in the cloud, in: International Conference on Research in Networking, Springer, 2011, pp. 172–185.

[92] S. Ren, Y. He, F. Xu, Provably-efficient job scheduling for energy and fairness in geographically distributed data centers, in: Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on, IEEE, 2012, pp. 22–31.

[93] A. Khosravi, R. Buyya, Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art, Advancing Cloud Database Systems and Capacity Planning With Dynamic Applications (2017) 27.

[94] J. Nurminen, T. Niemi, J. Strandman, K. Ruokosuo, Sunburnusing excess energy of small-scale production for distributed computing, Energy Efficiency (2017) 1–23.

[95] M. Imai, Kaizen (vol. 201), Random House Business Division New York. Iyengar, Sheena S., & Lepper, Mark R.(1999). Rethinking the value of choice: a cultural perspective on intrinsic motivation. Journal of Personality and Social Psychology 76 (3) (1986) 349.

[96] E. M. Goldratt, J. Cox, The goal: a process of ongoing improvement, Routledge, 2016.

[97] R. W. Schmenner, M. L. Swink, On theory in operations management, Journal of operations management 17 (1) (1998) 97–113.

[98] H. Kaynak, J. L. Hartley, A replication and extension of quality management into the supply chain, Journal of Operations Management 26 (4) (2008) 468–489.

[99] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, S. Moore, Measuring energy and power with papi, in: 2012 41st International Conference on Parallel Processing Workshops, 2012, pp. 262–268. doi:10.1109/ICPPW.2012.39.

[100] A. J. Oliner, A. P. Iyer, E. Lagerspetz, S. Tarkoma, I. Stoica, Collaborative energy debugging for mobile devices., in: HotDep, 2012.

[101] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, M. Hautreux, Energy accounting and control with slurm resource and job management system., ICDCN 8314 (2014) 96–118.

[102] D. Cheng, P. Lama, C. Jiang, X. Zhou, Towards energy efficiency in heterogeneous hadoop clusters by adaptive task assignment, in: Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on, IEEE, 2015, pp. 359–368.

[103] M. Guzek, D. Kliazovich, P. Bouvry, Heros: Energy-efficient load balancing for heterogeneous data centers, in: Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on, IEEE, 2015, pp. 742–749.

[104] D. Bán, R. Ferenc, I. Siket, Á. Kiss, Prediction models for performance, power, and energy efficiency of software executed on heterogeneous hardware, in: Trustcom/BigDataSE/ISPA, 2015 IEEE, Vol. 3, IEEE, 2015, pp. 178–183.

[105] D. del Rio Astorga, R. Sotomayor, L. M. Sanchez, J. G. Blas, A. Calderon, J. Fernandez, Assessing and discovering parallelism in C++ code for heterogeneous platforms, The Journal of Supercomputing.doi:10.1007/s11227-016-1794-8.

[106] Z. Ou, H. Zhuang, J. K. Nurminen, A. Ylä-Jääski, P. Hui, Exploiting hardware heterogeneity within the same instance type of amazon ec2, in: Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing, HotCloud'12, USENIX Association, Berkeley, CA, USA, 2012, pp. 4–4.
URL http://dl.acm.org/citation.cfm?id=2342763.2342767

[107] G. Procaccianti, H. Fernandez, P. Lago, Empirical evaluation of two best practices for energy-efficient software development, Journal of Systems and Software 117 (2016) 185–198.

[108] I. Manotas, L. Pollock, J. Clause, Seeds: a software engineer's energy-optimization decision support framework, in: Proceedings of the 36th International Conference on Software Engineering, ACM, 2014, pp. 503–514.

[109] H. S. Zhu, C. Lin, Y. D. Liu, A programming model for sustainable software, in: Proceedings of the 37th International Conference on Software Engineering-Volume 1, IEEE Press, 2015, pp. 767–777.

[110] A. Noureddine, A. Rajan, Optimising energy consumption of design patterns, in: Proceedings of the 37th International Conference on Software Engineering-Volume 2, IEEE Press, 2015, pp. 623–626.

[111] J. Michanan, R. Dewri, M. J. Rutherford, Greenc5: An adaptive, energy-aware collection for green software development, Sustainable Computing: Informatics and Systems 13 (2017) 42–60.

[112] R. Pereira, T. Carção, M. Couto, J. Cunha, J. P. Fernandes, J. Saraiva, Helping programmers improve the energy efficiency of source code, in: Proceedings of the 39th International Conference on Software Engineering Companion, IEEE Press, 2017, pp. 238–240.

[113] E. Baccarelli, N. Cordeschi, A. Mei, M. Panella, M. Shojafar, J. Stefa, Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study, IEEE Network 30 (2) (2016) 54–61.

[114] A. Khosravi, A. Nadjaran Toosi, R. Buyya, Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers, Concurrency and Computation: Practice and Experience.

[115] M. Kimmerlin, M. Plauth, S. Heikkila, T. Niemi, P. Sarolahti, Practical evaluation of a network expansion mechanism in an openstack cloud federation, in: Proceedings of IEEE CloudNet 2017, forthcoming, IEEE, 2017.

[116] A. Di Meglio, A. Purcell, M. Gaillard, Cern openlab whitepaper on future it challenges in scientific research, Tech. rep. (2014).

[117] A. Nowak, Future computing technology (2015).
URL `https://indico.cern.ch/event/404677/`

[118] A. Wright, Big data meets big science, Commun. ACM 57 (7) (2014) 13–15. `doi:10.1145/2617660`.
URL `http://doi.acm.org/10.1145/2617660`

[119] A.-P. Hameri, M. Nordberg, From experience: linking available resources and technologies to create a solution for document sharingthe early years

of the www, Journal of Product Innovation Management 15 (4) (1998) 322–334.