Improving Energy-Efficiency of Scientific Computing Clusters

Tapio Niemi¹, Jukka Kommeri¹, and Ari-Pekka Hameri²

¹Helsinki Institute of Physics, Technology Programme
CERN, CH-1211 Geneva 23, Switzerland
tapio.niemi@cern.ch, kommeri@cern.ch

²Faculty of Business and Economics (HEC),
University of Lausanne, CH-1015 Lausanne, Switzerland
ari-pekka.hameri@unil.ch

Abstract

We applied operations management principles on scheduling and allocation to scientific computing clusters to decrease energy consumption and to increase throughput. We challenged the traditional one job per one processor core scheduling method commonly used in scientific computing with parallel processing and bottleneck management. We tested the effect of increased parallelism by using different test applications related to high-energy physics computing. The test results showed that at best our methods both decreased energy consumption down to 40% and increased throughput up to 100% compared to the standard one task per CPU core method. The trade-off is that processing times of individual tasks get longer, but in scientific computing the overall throughput and energy-efficiency are often more important.

Introduction

Scientific computing clusters are widely used in many research fields, especially in experimental physics, astronomy, and bio-sciences. Computing intensive research easily deploys thousands, even hundreds of thousands, of CPUs to analyze various data sets and models. These clusters can be viewed as production resources processing jobs consisting of numerous tasks. 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. In all, a computing cluster closely resembles an industrial production unit, thus the working hypothesis for this paper is to apply operations management principles used in manufacturing to improve computing cluster productivity and overall efficiency.

Green computing, as it is now often referred to, seeks to improve efficiency of computing centers. It is a wide topic incorporating issues like data centre locations near cheap energy sources (Brown and Reams, 2010), minimizing so called e-waste (Hanselman and Pegah, 2007), designing optimal cooling infrastructure and running the centre in an optimal way (Marwah et al., 2009). Generally, energy and resource optimization in scientific computing has mostly focused on hardware and infrastructure issues, for example the development of more efficient hardware or optimizing cooling of computer centers, and has not focused as much on operational methods such as workload management and even less on operating systems and application software optimization for energy-efficiency.

The most well-known method for comparing energy efficiency of data centers is power usage effectiveness (PUE) metrics (Belacy, 2008). This is a ratio of the total facility power / IT equipment power. It indicates how much of the energy is lost in cooling, power distribution, and other infrastructures. However, it does not indicate how efficiently IT resources are operated. Limitations of PUE metrics have been recognized by the Green IT Promotion Council (2010), which proposes three additional metrics: ITEU (IT Equipment Utilization) - IT equipment usage in data center; ITEE (IT Equipment Energy Efficiency) - total rated capacity of IT equipment / total rated energy consumption of IT equipment; and GEC (Green Energy Coefficient), - Green (natural energy) energy / total energy consumption of data center. Based on these metrics it is possible to calculate Datacenter Performance Per Energy used (DPPE) metrics as follows: DPPE = (ITEU × ITEE × 1/PUE) / (1-GEC). Further, the report gives four methods for improving energy efficiency: 1) Operating the data center in an efficient way,
i.e. reducing amount of hardware and increasing its utilization; 2) Installing energy efficient hardware; 3) Improving energy efficiency of non IT infrastructure; and 4) Using renewable energy.

In this work we focus on a typical computing problem in high-energy physics: How to process a large set of jobs efficiently. While most of the existing work on high performance computing focuses on optimizing processing time of individual computing jobs, we try to optimize energy consumption per computing job and the total processing time of the set of jobs by choosing an optimal scheduling policy.

We study this problem by using real experimental physics data, computing jobs and dedicated computing clusters. CERN, the European Organization of Nuclear Research in Geneva, provided us with a unique possibility to experiment and test the hypothesis. The Large Hadron Collider (LHC) experiment at CERN produces about 15 petabytes of data in a year. The overall computing infrastructure comprises numerous computing clusters of alternative sizes, yet the total amount of CPUs is over 100,000 in over 140 computing centers. 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.

Basically, the problem is similar to production management in any manufacturing facility. In manufacturing this kind of optimization problem can lead to a trade-off situation: improving energy-efficiency can weaken throughput. However, within the context of the computing center our tests indicated that these two aims are not necessarily contradictory, meaning that optimizing system throughput also improves energy efficiency. Our approach is based on a conclusion that computers should run full-power or be turned off, since the fixed power consumption is around 50% of the full-power of the server. This chapter is based on our earlier papers (Hameri & Niemi 2010), (Niemi et al., 2009a), and (Niemi et al., 2009b).

Background

The theoretical background of our research comes from production optimisation research. Production engineers search ways to increase throughput of a facility with limited capacity. The principles of production, much like the law of capacity dictating that, in a steady state, all plants will release work at an average rate that is strictly less than average capacity, sets a challenge to get the best out of the facility by avoiding waste, unnecessary waiting times, shut downs and disrupting events. This means that the organization and allocation of work in production systems affects the performance of the system.

As a review of related research, we first present methods to improve efficiency at the whole data centre or at least pertaining to computing clusters. For example, Lefurgy et al. (2007) suggested a method to control peak server power consumption. The method is based on power measurement information on each computing server. Controlling peak power makes it possible to use smaller and more cost- and energy-effective power supplies. Moreover, Kappiah et al. (2005) developed a method to decrease power consumption of parallel computation in power-scalable clusters, while Conner et al. (2006) studied how energy can be saved by dynamically disabling network links in super computing clusters.

Another group of studies focuses on servers. Venkatachalam and Franz (2005) gave a detailed overview on techniques that can be used to reduce energy consumption of computer systems. Li et al. (2005) studied performance guaranteed control algorithms for energy management of the disk and main memory. Ge et al. (2005) studied methods based on the Dynamic Voltage Scaling technology of microprocessors and created a software framework to implement and evaluate these methods. Yuan and Nahmstedt (2002) studied the same issues in mobile devices. Essary and Amer (2008) optimized disk arm movements for saving energy, while Zhu et al. (2005) proposed a disk array energy management system. Finally, Zhang et al. (2004) gave compiler-based strategies to optimize cache energy consumption of microprocessors.

One relevant field of research for our study is scheduling methods on how the order and allocation of computing tasks to computing nodes affects the efficiency of the system. The way individual computers schedule their processes is beyond the scope of this paper. Scheduling problems can be classified according to the following properties:

- on-line / offline
• knowledge on jobs
• knowledge on computing resources

More formally (e.g. following (Pinedo, 2008) or (Brucker, 2007)) the scheduling problem can be defined as follows: We have m machines (i.e. computing nodes in our case) and j jobs to be processed. A schedule S is an allocation of time intervals from machines for each job. The challenge is to find an optimal schedule for jobs when taking into account certain constraints. A schedule is optimal if it minimizes a given optimality criteria, for example time, cost, or usage of some resources. The optimality criteria can be defined in several ways, such as the completion time of the last job or the total completion time i.e. sum of all completion times, or in our case, energy efficiency. The overall objective can also be a (weighted) sum of the sub-objectives, which often leads to a Pareto-optimal schedule.

Scheduling is a widely studied topic but most of the work focuses on finding optimal schedules when jobs have proceeding constraints and/or strict time limitations. Optimizing total throughput or energy efficiency in high throughput computing has received less research interest. Instead some works suggest clearly opposite approaches: For example, Koole and Righter (2008) suggest a scheduling model in which tasks are replicated to several computers. However, the authors do not estimate how much more resources are needed when the same tasks (or at least parts of them) are computed several times. Fu et al. (2007) present a scheduling model capable to restart batch jobs. They give an efficient algorithm to solve the problem but they do not mention resource usage. Further, Da Silva et al. (2003) presented another model using replication.

Most of the research on scheduling focuses on theoretical aspects but some practical studies do exist. For example, Etsion and Tsafrir (2005) compared commercial workload management systems focusing on their scheduling systems and default settings. According to the authors, the default settings are often used by the administrators, or they are just slightly modified. Goes et al. (2005) studied scheduling of irregular I/O intensive parallel jobs. They noted that CPU load alone is not enough but all other system resources (memory, network, storage) must be taken into account in scheduling decisions.

More on the theoretical side, Prasanna and Musiecs (1996) define a theoretical scheduling model in which the number of processors allocated to a task can be a continuous variable, which makes it possible to allocate all processors for one task if needed. Edmonds (2000) studied non-clairvoyant scheduling in multiprocessor environments. In his model, the jobs can have arbitrary arrival times and execution characteristics can change. Santos-Neto et al. (2004) studied scheduling in case of data-intensive data mining applications, while Wang et al. (2009) developed optimal scheduling methods in a case of identical jobs and different computers. Their on-line algorithm aims to maximize the throughput and to minimize the total load. Finally, self-learning scheduling models also exist, such as that of Shivam et al. (2006).

One group of scheduling research focuses on grid computing. Medernach (2005) studied workload in a grid computing cluster in order to be able to compare different scheduling methods. The idea of the work was to find ways to group cluster users to characterize their usage. The scheduling was based on one job per CPU core idea. Kurowsk et al. (2008) studied two-level hierarchical grid scheduling taking into account all stakeholders of grid computing systems. The approach does not require time characteristics of jobs being known. Aziz and El-Rewini (2008) studied online scheduling algorithms based on evolutionary algorithms in the grid context. Ni et al. (2005) developed a heuristic scheduling algorithm for grid environments. The algorithm is based on the concept of changing tasks between computers. Some studies apply fuzzy logic and other novel optimization methods. For example, Cao et al. (2003) applied fuzzy time functions to a large scale grid workflow management, Liouane et al. (2008) studied multi-objective scheduling with a fuzzy controller, and Abraham et al. (2009) used a fuzzy particle swarm algorithm. Some evolutionary approaches, such as (Moallem and Ludwig, 2009) and (Grosan et al., 2007) have also been studied. Finally, there are some studies on energy-aware scheduling. For example, Rajan and Yu (2008) and Mukherjee et al. (2009) studied this topic. Further, Bunde (2006) developed power aware scheduling methods for minimizing energy consumption and not reducing system performance by applying dynamic voltage scaling technologies.

Other related work (e.g. Mu'alem and Feitelson, 2001; Tsafrir et al., 2005, 2007) investigates the accuracy of user time estimates and noticed that they are inaccurate. To solve the problem, Tsafrir et al. developed a model to
simulate user run time estimates. Piro et al. (2009) studied a similar problem in the Grid context. Yom-Tov and Aridor (2008) presented a self-learning algorithm to estimate resource requirements of batch jobs. Moreover, Karlsson et al. (2005) introduced a method to control storage access in a data centre. The method is based on adaptive controllers and does not assume prior knowledge of the system.

Virtualization has also been studied as a novel method to improve energy efficiency of computing centers. Virtualization makes it possible to combine services from several physical machines on one physical machine. This is helpful when the utilization of the machines is minimal and the services are running on dedicated hardware. With virtualization, it is possible to use excess resources and downsize the hardware pool. However, consolidation of jobs to fewer servers is not directly relevant to our problem since we assume that there are a large number of jobs to be processed. This is a realistic assumption in physics computing. Increased security provided by virtualization is not usually needed in scientific computing.

The allocation and management of virtualized resources is a widely studied area. Both dynamic and static solutions have been proposed. For example, Bobroff et al. (2007) developed a dynamic system that forecasts future load and in this way can minimize the number of physical servers. Further, Viswanathan et al. (2008) studied energy efficiency of virtual servers with predefined loads. They succeeded in improving the energy-efficiency by scheduling predefined virtual machine loads to minimize the resource contention and maximize resource usage. However, running several virtual machines on one physical computer does not come without penalties. Apparao et al. (2008) studied the effects of consolidation running several benchmarks in a virtualized environment. This study indicates that sharing physical resources and contention for resources clearly reduces the performance of individual virtualized applications. Pradeep et al. (2007) carried out a performance study of two different virtualization technologies. They noticed that some overhead always exists and the size of it depends on the used virtualizing technology. Like Pradeep et al., also Regola et al. (2010) evaluated different virtualization technologies and concluded that full virtualization technologies still pose heavy overheads and are not viable in high throughput computing that use I/O excessively. Though virtualization would allow scientists to use more heterogeneous environments and send their jobs to computing clusters in a form of a virtual machine.

As shown above, research on improving efficiency in computing systems focuses on hardware optimization, power management, various scheduling models etc. However, there is little research on how to apply production management principles to the management of computing resources, i.e. viewing a computing centre as a factory with limited capacity and bottleneck resources. According to the manufacturing and production management point of view the key issues are related to the optimization of throughput and minimizing operating costs; these are also desirable objectives when computing resources are managed efficiently. Our aim is to increase the utilization rate of bottleneck components (usually I/O or CPU) and in therefore increase throughput, which decreases average power consumed by a computing job.

**Problem and Methods**

The main research question of our study is: How can computing centers improve throughput and energy-efficiency by scheduling jobs based on their estimated properties and load requirements of all components (CPU, memory and network) in computing nodes. A computing system is much more flexible than a manufacturing system since, in a computer, it is possible to run several tasks in parallel, and since tasks are not dedicated to any specific resource, set-up times between different jobs are also practically non-existent. Another difference with manufacturing is that it is difficult to see what is happening inside the computer, thus all measures are based on the in- and output of the system, as actual work in progress takes place literally in a black-box.

The motivation for our work comes from the CERN’s LHC experiment. The Worldwide LHC Computing Grid (WLCG1) is used to analyze the data produced by the Large Hadron Collider at the CERN, and it consists of hundreds of terabytes of data storage and tens of thousands CPU cores. On this scale, even small system optimization can offer noticeable energy and cost savings and performance improvements. Since high-energy physics computing has many special characteristics, common industry practices are not always the best. The main characteristics of LHC computing are: jobs contain large sets of similar tasks, data-intensive computing, processing time of an individual task is not critical, no preceding conditions among tasks, and little intercommunication between tasks, i.e. high parallelisms.
We use the following basic terminology in this work: A task is the smallest elementary entity when work is processed. The task starts, retrieves/reads its possible input file, processes the data and possibly writes its output file. A job is a collection of tasks. In our current work we assume that the processing order of the tasks inside a job is irrelevant. By energy efficiency, we mean the number of similar tasks or jobs that can be processed by using the same amount of electricity while throughput means the number of similar tasks or jobs that can be processed in a time unit.

We started our study by collecting and analyzing log data for a physics computing cluster. When analyzing the data we noticed that lead 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 and network. When estimating memory utilization of jobs compared to their CPU utilization, the memory utilization rate was about half of the CPU utilization rate. One reason for this is an irregular memory utilization curve of physics jobs. We assume that the other reason is related to I/O waiting times. Based on this analysis we assumed that reasonable overloading and processing different kinds of tasks in parallel can improve throughput by keeping bottleneck resources busy. The following two hypotheses document this:

1. Throughput can be improved and electricity consumption reduced in data intensive high-energy physics (HEP) computing compared to the traditional single task per CPU core processing by multitasking, i.e. processing more than one task per CPU core in parallel.
2. The performance can further be increased by mixing heterogeneous tasks while multitasking.

The hypotheses above do not give any upper limit for the number of parallel tasks. In practice, the number is limited by physical resources, especially by memory. The practical challenge in multitasking is to find the optimum number of simultaneous tasks. We used the following criteria: 1) a task uses a minimal amount of energy, and 2) the total throughput is maximal. These goals could be contradictory but in all our tests we noticed that the 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 (i.e. Step 1).

In an important special case in which all tasks are identical, the problem becomes: How many tasks must be processed simultaneously in a computing node.

We tested the hypotheses by building a realistic test environment, and then developing, and finally testing different scheduling methods. In practice we did this by processing a job, i.e. a large set of tasks and measuring the time and electricity consumed during the test run. As explained above, we assume having a large set of independent tasks, and we are interested in the total processing time and electricity consumption of this set. By the total processing time we mean the total time from the submission of the first task in the set to the computing cluster to the end of processing the last task in the same set. Our aim is to minimize both the amount of electricity and time needed to process the set. Minimizing the processing time is equal to maximizing the throughput.

Our experimental system was comprised of one front-end computer running the workload management system and one computing node and 1 Gb local area network. To make sure that results are not valid for one particular computer, we used two different architectures: 1) a Dell PowerEdge SC1435 server with two 4-core AMD Opteron 2376 2.3GHz and 32 GB of memory, and 2) a Dell PowerEdge R410 server with two Intel Xeon E5520 quad core 2.27 GHz processors and 16 GB DDR3 memory. The first server was used in our first test and the second server in tests two and three. We used a Sun Grid Engine (SGE) (Sun, 2008) as a workload management system, as it is also commonly used in grid computing clusters. It has various features to control scheduling. The
scheduling is based on the load of the computing nodes and resource requirements of the jobs. There are also various other batch scheduling systems such as Torque\(^1\), OpenPBS\(^2\), LSF\(^3\), and Condor\(^4\). These systems have different features but their basic functionality is very similar.

Electricity consumption of the computing nodes was measured with the Watts Up Pro electricity meter. The accuracy of the meter is around \(+\,-1\%\). The operating system used with Xeon and Opteron was Rocks 5.0 with kernel version 2.6.18.

We used five different test applications. The first three were implemented to test usage of only one resource and the fourth and fifth were real physics applications:

- The I/O test application writes and read 300 MB files multiple times. After generating a file, the content is copied to another file 20 times. Each time the file is a bit different (small shift in numerical values) to avoid buffering.

- The CPU test application contains a long loop calculating floating point multiplication. A reminder of the index variable is also used to make compiler optimization harder.

- The memory test application reserves memory (200 MB) and fill it with numbers. After that it read a part of the memory and writes it to another part. This is done multiple times.

- The physics data analysis application is based on the CMSSW framework. Input data for the test is from the CRAFT (CMS Running At Four Tesla) experiment that used cosmic ray data recorded with CMS detector at the LHC during 2008 (Acosta and Camporesi, 2008). This detector was used much like current LHC experiments and the data was very close to the final data analysis. The analysis software reads the input file (94 MB or 360 MB), performs the analysis, and writes a small summary file on the local disk. Input data is stored in ROOT data container files (Antcheva et al, 2009). Physics data is stored in binary format to save space and one root file usually contains hundreds of events captured by the detectors. The disk I/O during the application processing is shown in Figure 1 and the memory usage in Figure 2.

- The CPU intensive physics application is also based on the CMSSW framework (Fabozzi et al., 2008) including event generation with Pythia\(^6\) (Sjostrand et al., 2006) and full detector simulation with Geant4 (Agostinelli et al., 2003). The Pythia program is a standard tool for the generation of high-energy collisions using monte carlo methods.

Each of these test sets was run several times to get stable results. The input data were different for each run to eliminate the effect of the disk cache. Additionally, the disk cache was cleared between the test sets to provide a similar environment for all the test sets. Since differences in averages among the different test sets were relatively high and standard deviation inside the sets low, this test setting gave us large enough sample size to look for statistically significant results.

---

3. www.platform.com/Products/Workload-Management
4. www.cs.wisc.edu/condor
Tests and Results

We tested our hypotheses by using simplified test applications and high-energy physics applications. In the first test we compared the performance of different applications when changing the amount of parallelism. The second test shows how the performance of the I/O application varies when it is processed with a different amount of parallel tasks. The final test demonstrates that mixing different tasks (I/O and CPU oriented) can further increase performance.

In the first test, running 3-4 tasks per CPU core yielded the best results, but the test also showed that benefit of multitasking heavily depends on the application used. If an application does almost only memory operations such as our memory test application, the benefit is almost zero. Luckily, few real applications are like this. For computing oriented CPU applications improvements are quite clear. Our first test case produced 13% more throughput, while 7% less energy was consumed. Much larger improvements are related to I/O, by processing our I/O test application using 4 tasks per CPU core setting instead of 1 task per core, improved throughput over 50% and decreased energy consumption around 25%. However, the biggest improvement gave our real physics analysis application 100% more throughput while 30% less energy was consumed. The test results are shown in Table 1 and percentile changes when comparing the best case to one task per CPU core case, are shown in Figure 3. These results validate Hypothesis 1.

The reason for the improvements is not straightforward. Intuitively one could assume that especially overloading
I/O could decrease throughput. However, according to our tests, it seems to be that overloading I/O improves performance. We assume two possible reasons: 1) Overloading keeps the bottleneck busy and eliminates the total I/O waiting time in the system, and/or 2) overloading improves the efficiency of disk caches since it is more likely that some other task has already fetched the required data.

Figure 3. Improvements of running 3-4 tasks in parallel compared to 1 task/core
Table 1. Test results of the first test

<table>
<thead>
<tr>
<th>Test type</th>
<th>Jobs/core</th>
<th>Jobs/hour</th>
<th>Wh/job</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>1</td>
<td>320</td>
<td>0.79</td>
<td>4</td>
</tr>
<tr>
<td>CPU</td>
<td>1</td>
<td>407</td>
<td>0.6</td>
<td>3</td>
</tr>
<tr>
<td>Physics analysis</td>
<td>1</td>
<td>188</td>
<td>1.1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>11</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>15</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>19</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>23</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>27</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>31</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>35</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>39</td>
<td>39</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>43</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>47</td>
<td>49</td>
<td>49</td>
</tr>
</tbody>
</table>

Figure 4. Comparing performance and energy-efficiency when processing physics analysis jobs using different numbers of tasks per CPU core settings to 1 task per core setting.

Figure 4 shows the variation of throughput and energy-consumption when processing 1 to 50 I/O tasks in an 8-core server. The difference is calculated comparing the result to 1 task per core, i.e. 8 tasks per server, setting. Each measurement was taken twice and the test application was standard physics data analysis. The figure shows that after reaching 1 task per core situation, the performance and energy-efficiency still improve almost linearly until the physical memory is full.
Our final test was related to mixing different applications in parallel processing. Since the physics analysis seemed to give the biggest benefit on multitasking, we tested the effect of mixing I/O and CPU intensive tasks by using two different physics applications: an I/O intensive data analysis and a CPU intensive simulation. We compared first processing CPU-oriented tasks and then I/O-oriented physics tasks to processing the same jobs mixed together. Figure 5 illustrates the improvements in energy consumption and throughput in these tests. The results indicated that mixing tasks using different resources clearly improved both throughput and energy-efficiency. This validates Hypothesis 2.

Discussion

In scientific computing clusters, the single job (task) per core-scheduling method is mostly used. This simply means that each CPU core can run a maximum of one job. Usually the jobs are distributed equally to all computing nodes of the cluster. Instead of this common practice, we tested variations of different scheduling methods based on the idea that it is necessary to fully load computing nodes. Our tests showed that running multiple tasks simultaneously, or mixing tasks that utilize different resources, can decrease energy usage per computing task and improve throughput of the computing node when running high-energy physics (HEP) applications. The trade-off is that processing times of individual tasks are longer but in cases, such as HEP computing, in which the tasks are not time critical, usually only the total throughput is important. We assume that the reason for the improvement is that some component within the system becomes a bottleneck making the processor wait for some operations, causing CPU utilization to drop from the maximum 100%. Based on our tests, we can assume that I/O traffic is an obvious bottleneck.

The results are also in line with Schmenner (2010), who proposed that companies emphasizing swift even flow, i.e. they focus on speed and variability reduction, would outperform slower companies. This principle also implies that companies should focus on value adding tasks and removing of non-value adding tasks, while at the same time trying to eliminate bottlenecks in order to introduce even flow and short lead times.

Our work showed that multitasking improves efficiency in HEP data analysis but it also easily makes disk access a bottleneck. While the number of CPU cores continuously increases, I/O remains a bottleneck. Faster disks like Solid State Drives (SSD) can be used to partially remove this bottleneck, since SSD disks are clearly faster than hard disks. Our future research will also focus on this issue, and our preliminary results already confirmed this assumption (Niemi et al., 2011). However, our results depended highly on the applications used.

Generally, value adding production systems with high throughput and short lead times have been proven to generate benefits other than pure output performance. Statistically these systems also produce better quality and less waste, and thus have a better overall environmental efficiency. Systems which perform better operationally, also have more satisfied customers and tend to be more competitive in the market. Based on this, an optimized computing system should, in addition to reduced electricity consumption, also work more reliably and offer more computing power. When applying new optimizing methods to large scale computing resources, the results
can bring remarkable savings, especially in large computing installations.

There are still several unsolved issues concerning the loading of computing resources. This chapter presented a method based on the processing of more than one task per CPU core. In this method an obvious challenge is to decide upon the number of tasks to run in parallel. Therefore we have started to develop automated methods for dynamically finding the right settings. One method based on fuzzy control is introduced in (Niemi and Hameri, 2011). Another approach uses the simulation approach to test different scheduling strategies much faster than with contemporary test runs. We have developed simulator software for this purpose (Kommeri et al., 2011).

Conclusions

We studied different scheduling settings with different hardware for high-energy physics computing to minimize electricity usage and maximize performance. Instead of the common practice of one task per CPU core scheduling used in many computing clusters, we tested variations of different scheduling methods based on the idea to process more tasks in parallel.

The results showed that optimizing the configuration of a workload management system by using our methods improved throughput and decreased electricity consumption. However, improvements heavily depended on the application. In a modern multicore environment running 3-4 tasks per CPU core gave the best results. The measured improvements of our methods increased throughput up to 100% and decreased energy consumption down to 40-50% compared to the standard one-task-per-core practice in scientific computing. The trade-off is that processing times of individual tasks become longer. Though in cases, such as HEP computing, in which the tasks are not time critical, the total throughput and energy-efficiency are more important. Instead, we did not find a trade-off between throughput and energy-efficiency, as in our tests the most efficient method also minimized energy consumption per processed task. As a general conclusion our results show that information on all components of the computer, including memory usage, processor load, and I/O traffic, should be used when making scheduling decisions.

References


Green IT Promotion Council (2010). Concept of New Metrics for Data Center Energy Efficiency Introduction of Datacenter Performance per Energy [ DPPE ], Japan.


