Comput Sci Res Dev (2012) 27:265–275
DOI 10.1007/s00450-011-0189-6

SPECIAL ISSUE PAPER

# Energy-aware job scheduler for high-performance computing

**Olli Mämmelä · Mikko Majanen · Robert Basmadjian · Hermann De Meer · André Giesler · Willi Homberg**

Published online: 31 August 2011


**Abstract** In recent years energy-aware computing has become a major topic, not only in wireless and mobile devices but also in devices using wired technology. The ICT industry is consuming an increasing amount of energy and a large part of the consumption is generated by large-scale data centers. In High-Performance Computing (HPC) data centers, higher performance equals higher energy consumption. This has created incentives on exploring several alternatives to reduce the energy consumption of the system, such as energy-efficient hardware or the Dynamic Voltage and Frequency Scaling (DVFS) technique. This work presents an energy-aware scheduler that can be applied to a HPC data center without any changes in hardware. The scheduler is evaluated with a simulation model and a real-world HPC testbed. Our experiments indicate that the scheduler is able to reduce the energy consumption by 6–16% depending on the job workload. More importantly, there is no significant slowdown in the turnaround time or increase in the wait time of the job. The results hereby evidence that our approach can be beneficial for HPC data center operators without a large penalty on service level agreements.

**Keywords** HPC · Energy-efficiency · Simulation · Testbed · Scheduling · Power consumption

O. Mämmelä (✉) · M. Majanen
VTT Technical Research Centre of Finland, Oulu, Finland
e-mail: olli.mammela@vtt.fi

M. Majanen
e-mail: mikko.majanen@vtt.fi

R. Basmadjian · H. De Meer
University of Passau, Passau, Germany

R. Basmadjian
e-mail: robert.basmadjian@uni-passau.de

H. De Meer
e-mail: hermann.demeer@uni-passau.de

A. Giesler · W. Homberg
Jülich Supercomputing Centre, Jülich, Germany

A. Giesler
e-mail: a.giesler@fz-juelich.de

W. Homberg
e-mail: w.homberg@fz-juelich.de

## 1 Introduction

In 1990s the energy-aware computing started with wireless mobile and embedded devices since they had only limited power available in batteries. Recently, energy-awareness has also become a major issue in wired devices. As summarized in [28], in 2006, U.S. servers and data centers consumed around 61 billion kilowatt hours (kWh) at a cost of about 4.5 billion U.S. Dollars. This is about 1.5% of the total U.S. electricity consumption or the output of about 15 typical power plants. High energy consumption naturally causes huge environment pollution. It has been estimated that ICT as a whole covers 2% of world's $CO_2$ emissions [23], a number equivalent to the emissions of the aviation.

HPC data centers do not make any exception. In fact, the ever-growing demand for higher performance seems to increase the total power consumption, even though more flops per watt are achieved. Table 1 presents recent power consumption numbers from the TOP500 list of HPC systems [26]. The most power consuming HPC system is the Jaguar (No. 2 in Nov 2010) with a total power consumption of 6.95 MW.

In this paper we introduce an energy-aware scheduler for HPC data centers that communicates with the data center's resource management system. The performance of the

**Table 1** Power consumption of HPC systems [26]

| Year | Nov 2008 | Nov 2010 |
|---|---|---|
| Avg. power consumption in TOP10 systems | 2.48 MW | 3.2 MW |
| Avg. power efficiency in TOP10 systems | 228 $\frac{\text{Mflops}}{\text{W}}$ | 268 $\frac{\text{Mflops}}{\text{W}}$ |
| Avg. power consumption in TOP500 systems | 358 kW | 447 kW |
| Avg. power efficiency in TOP500 systems | 132 $\frac{\text{Mflops}}{\text{W}}$ | 219 $\frac{\text{Mflops}}{\text{W}}$ |
| Systems using more than 1 MW in TOP500 | 14 | 25 |

scheduler was studied by simulation and real testbed experiments. The results show that energy savings of 6–16% can be achieved without significant increase in job wait or turnaround times.

The rest of the paper is organized as follows: Sect. 2 presents the related work in energy-aware HPC, Sect. 3 presents the developed energy-aware scheduler, Sect. 4 introduces the developed simulation framework including the simulation model and the power consumption models of the data center. Section 5 presents the simulation scenario and Sect. 6 shows the simulation and testbed results. The paper is concluded and future work is presented in Sect. 7.

## 2 Related work

Research in energy-aware HPC has been active in recent years. In order to reduce energy consumption, research has mainly focused on the following topics:

- Energy-efficient or energy proportional hardware
- DVFS (Dynamic Voltage and Frequency Scaling) technique
- Shutting down hardware components at low system utilizations
- Power Capping
- Thermal Management

By designing energy-efficient hardware, the components themself are energy-efficient, i.e., they consume less energy than the standard ones. [5] argues that the power consumption of a server should be proportional to its workload, i.e., it should consume no power in idle state, almost no power when the workload is very low, and eventually more power when the workload is increased. Ideally, an energy proportional server could save half of the energy used in data center operations.

Since the processor power consumption is a significant portion of the total system power (roughly 50 % under load [14]), the DVFS technique is used for controlling the CPU power. By running a processor at lower frequency/voltage energy savings can be achieved, but the job execution time is increased. Thus, DVFS should be applied at a period of low system activity, since user's SLAs (Service Level Agreements) must be respected. Some research efforts on this topic can be found in [10, 17], and [13].

In a typical HPC data center, servers consume nearly as much energy in idle state as when running an application. At phases of low system utilization, some servers or their components could be shut down or switched to a low-power state. This strategy thus tries to minimize the number of active servers of a system while still satisfying incoming application requests. Since this approach is highly dependent on the workload, the challenge is when to shut down components and how to provide a suitable job slowdown value.

In [16] the authors performed an empirical 3.75 year study by implementing a energy-aware scheduler to their HPC system. The operation of the scheduler is simple: if a node is inactive for 30 minutes, it is powered off. When the node is required for job execution, it is powered on and moved to active state. Powering up a node on their system takes approximately 45 minutes, which is a substantially large time. The strategy described can nevertheless reach power efficiency improvement of 39% at best.

Pinheiro et al. [31] also developed an approach that dynamically turns cluster nodes on and off. The approach uses load concentration to concentrate workload in fewer nodes, so that idle nodes can be turned off. The experimental tests on a static 8-node cluster point out 19% savings in energy. In their testbed, it takes about 100 seconds to power on a server and 45 seconds to shut it down. The degradation in performance is approximately 20%. However, all the experiments start with a single-node configuration, which includes having all nodes except one powered off. When the demand of the system rises, additional nodes are powered on.

By using *Power Capping*, the data center operator can set a threshold of power consumption to control that the actual power of the data center does not exceed it [11]. It prevents sudden rises in power supply and keeps the total power consumption under predefined budget. Basically the power consumption can be reduced by descheduling tasks or CPU throttling, for example.

Thermal management techniques are similar as power capping techniques, except that the heat of the data center is monitored instead of the power consumption [28]. Higher temperatures have a large impact on system reliability and can also increase cooling costs. The workload of the system is adjusted according to a predefined temperature threshold: if the temperature on a server rises above the threshold, its workload is reduced.

In this paper we also introduce power consumption models for the data center. Some related work on this topic can be found in [34, 35], and [32].

## 3 HPC energy-aware scheduler

Typically, HPC cluster consists of a resource management system (RMS) and several compute nodes. Users submit jobs to the queue(s) inside RMS. The job scheduler decides, which jobs are run in which compute nodes and when. Thus, the scheduler is a natural place for making energy-aware decisions. There are several algorithms for job scheduling. The developed scheduler supports three commonly used scheduling algorithms with additional energy-saving features. This section describes the functionalities of the supported scheduling algorithms.

### 3.1 First In, First Out (FIFO)

FIFO (or First Come, First Served (FCFS)) is the simplest scheduling algorithm. In FIFO, new jobs are inserted in the end of the queue. Whenever there are enough resources available for the first job in the queue, it is dequeued and executed. When there are not enough resources for the first job in the queue, all the jobs in the queue have to wait. This creates possibilities for energy savings. In the energy-aware FIFO (E-FIFO), we power off idle nodes, if there is more than $T$ seconds time remaining before the estimated start time of the first job in the queue. $T$ is considered as a power off threshold.

### 3.2 Backfilling (first fit and best fit)

The backfilling algorithm works like FIFO, but when there are not enough resources for the execution of the first job in the queue, the rest of the queue is checked for finding the jobs that could be executed with the available resources. The execution of that kind of backfill job should not cause any delay for the first job (or $n$ first jobs, typically $n < 3$) in the queue. So the backfill job execution is limited to the available resources and the available time before the expected start of the first job in the queue. The expected start time of the first job in the queue is calculated based on the walltimes of the jobs that are currently running. The walltime estimations are given by the users and are inaccurate [4, 8], which may sometimes cause some delay for the first job in the queue when the running job finishes earlier than expected.

There are several ways to choose the backfill job. In the *backfill first fit* (BFF) strategy, the first job in the queue that meet the resource and time constraints is chosen. In the *backfill best fit* (BBF) strategy, all the potential backfill jobs are searched and the selection is made based on certain criteria. For example, the shortest or longest backfill job can be chosen, or the one that requires most or less processors.

In the energy-aware backfilling algorithms (E-BFF and E-BBF), we use the same methods for energy saving as in FIFO: the idle nodes are powered off if there are more than $T$ seconds before the estimated start of the first job in the queue. Since the backfill algorithms try to exploit all the idle nodes, there are less possibilities to turn off idle nodes than in FIFO. On the other hand, backfilling is more efficient in running the jobs, so the wait times of the jobs are shorter than in FIFO.

## 4 Simulation framework

This section describes the developed simulation and energy consumption models.

### 4.1 Simulation model

The HPC simulation model is built by using OMNeT++ [20] and the INET Framework [19]. Figure 1 presents the network topology used in the simulations that consists of three backbone routers, a gateway router, a data center module and a fixed number of client modules.

As can be seen in Fig. 2, the data center module contains a RMS module, a fixed number of servers and a router between them. Clients, servers and the RMS are all derived from the StandardHost module of the INET Framework. The StandardHost module has transport, network and physical layer protocols already available. The client, server and RMS functionalities were developed as an application layer program to all these modules. Clients send job request messages and input data to the RMS. The RMS module handles all incoming job requests arriving to the data center module and allocates the jobs to the servers for execution according to the specified scheduling algorithm. Thus, the RMS also functions as a scheduler in the simulation. Servers receive job requests delivered by the RMS and execute the jobs.

The application programs also include models of the server components and their respective energy consumption formulas. These formulas are described in Sect. 4.2. For example, the application module on a server specifies also the details of the CPUs, cores, memory, fans, etc. of the server. The status of each of these is updated every time something changes on the server, for example a new job starts its execution or a job completes and the server goes to idle state. Between the two events, the state of the server is considered as constant. The energy consumption between the two events is calculated using the developed models (equations) for CPU, RAM, fan, etc.

### 4.2 Power consumption models

In this section, we introduce the models used to model the power consumption of servers. It was shown in [11] that the main contributors in power consumption of a server are the processor (37%), memory (17%), mainboard (12%),

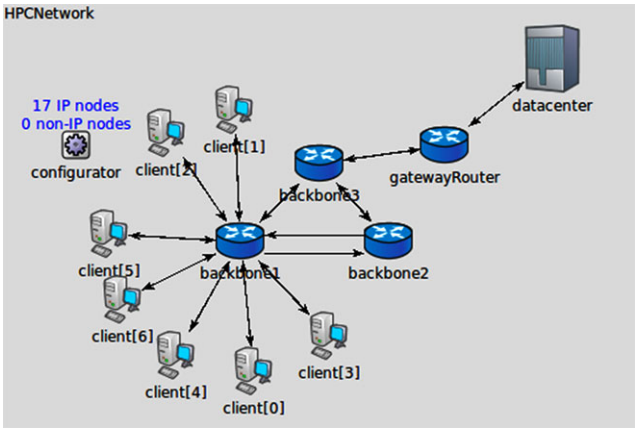



**Fig. 1** Network topology

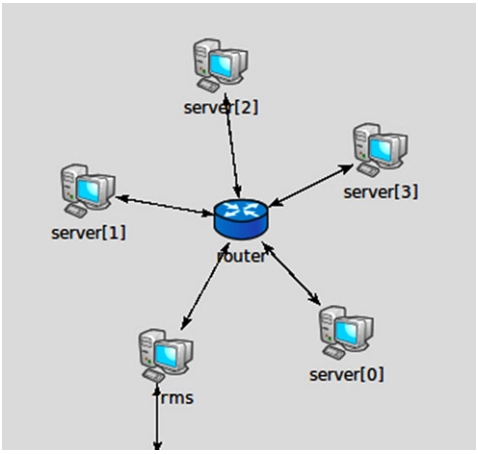



**Fig. 2** Data center module

hard disk drive (6%), fans (5%) and PCI slots (23% ). Next, we provide the power consumption models for each of the above-mentioned components and then give the generic model of a server.

### *4.2.1 Processor*

Unlike the traditional single-core processors, modern ones are equipped with multiple cores which have a significant impact on the power consumption. On the other hand, the multi-core processors possess energy-efficient features and mechanisms which trade performance with reduced overall power consumption. Based on our observations, we noticed that individual core of multi-core processors has the same power consumption behavior as that of the single-core ones: the power consumption increases linearly with the utilization. As a matter of fact, inspired by the linear utilization-based power consumption model for single-core processors of [11], we give the power consumption of individual cores by the following equation:

$$P_c = P_{max} \times \frac{L_c}{100}, \tag{1}$$

where $L_c$ denotes the utilization (load) of the corresponding core and $P_{max}$ indicates the maximum power of the processor computed by the following well known equation:

$$P_{max} = v^2 f C_{eff},$$

such that $v$ and $f$ denote respectively the voltage and frequency at the corresponding utilization ($L_c$), whereas $C_{eff}$ indicates the effective capacitance. Hence, the power consumption of multi-core processors is given by the following equation:

$$P_{CPU} = P_{idle} + \sum_{i=1}^{n} P_{c_i}, \tag{2}$$

such that $P_{idle}$ denotes the power consumption of the processor (considered constant in this paper) in the idle state whereas $P_{c_i}$ represents the power consumption of each core given by (1). It is worthwhile to note that due to (1) energy-efficient mechanisms such as INTEL's SpeedStep and AMD's Quiet'n'Cool as well as (2) resource sharing such as the L2-cache, some energy reduction factors are introduced to (2) whose modeling is out of the scope of this paper.

### *4.2.2 Memory*

Random Access Memories (RAM) can be classified into different categories based on their technology (e.g. DRAM, SDRAM, etc). Furthermore, there exist several types of memories (e.g. DDR, $DDR_2$, $DDR_3$) where each can be either buffered or unbuffered. All the above mentioned characteristics of memory play a role in its power consumption. In this paper, we focus on unbuffered Synchronous Dynamic RAM $DDR_2$ technology since the servers of our real-world testbed are equipped with such type of memories.

Given an unbuffered SDRAM of type $DDR_2$, then its power consumption at the idle state is given by:

$$P_{RAM\_idle} = \sum_{i=1}^{n} s_i \times p, \tag{3}$$

where $n$ denotes the number of installed memory modules and $s$ indicates the size (in GB) of each individual memory. For an unbuffered $DDR_2$ SDRAM, Table 2 gives the values of $p$ for different vendors, where $f$ denotes the *frequency* (in MHz) of the memory module. Note that the *Generic* vendor type provides a rough estimation of the idle power consumption for vendors other than those mentioned in Table 2.

**Table 2** Values of $p$ for different unbuffered $DDR_2$ SDRAMs

| Vendor | Value |
|---|---|
| Kingston [2] | $\frac{f}{1000}$ |
| Samsung [3] | $0.95 \times \frac{f}{1000}$ |
| Hynix [1] | $1.9 \times \frac{f}{1000}$ |
| Generic | $1.45 \times \frac{f}{1000}$ |

In order to derive a model for the dynamic power consumption of RAMs, we performed observations using the RAMspeed benchmark [18]. RAMspeed is a Linux-, DOS- and Windows-based benchmark that tests the performance of the memory subsystem. The main objective of the experiments was to allocate a given part (different sizes) of the physical RAM and perform read and write operations on this part of the RAM. Hence, we derive the dynamic power consumption model of an unbuffered $DDR_2$ SDRAM in the following manner:

$$P_{RAM} = P_{RAM\_idle} + \beta, \tag{4}$$

such that $P_{RAM\_idle}$ is given by (3) and $\beta = 7.347$. Based on the aforementioned observations, we noticed that $\beta$ is constant due to the fact that there is always only 1 active operating rank per channel regardless of the number of memory modules or module ranks in the system. The remaining other ranks and other memory modules are in idle mode drawing less power.

#### 4.2.3 Hard disk

In general, the hard disk drive can be in one of the following three states: accessing, idle, or startup. Each of the above states has a different power consumption behavior due to the involved mechanical and electrical interactions. We noticed that the startup and accessing mode power consumptions are respectively 3.7 and 1.4 times greater than that of the idle mode power consumption. Then the power consumption of the hard disk is given by:

$$P_{HDD} = a \times 1.4 \times P_{idle} + b \times P_{idle} + c \times 3.7 \times P_{idle}, \tag{5}$$

where $P_{idle}$ is the idle power consumption of the hard disk provided by the manufacturer's data sheet, whereas $a, b, c \in [0, 1]$ denote respectively the probability that the disk is in accessing, idle and startup modes. Values for $a$, $b$ and $c$ are chosen based on the frequency of read and write operations performed on the disk, which depend on the job characteristics. Note that the details on choosing values of $a$, $b$ and $c$ as well as the standby and sleep mode power consumptions of the hard disk can be found in [6].

#### 4.2.4 Network interface card

For the Network Interface Card (NIC) power consumption, we consider the linear model [7, 33], in which the power consumption depends linearly on the traffic load. Thus, the power consumption of a NIC device is given by the following equation:

$$P_{NIC} = P_{idle} + (P_{max} - P_{idle}) \times pps, \tag{6}$$

where $P_{idle}$ is the power consumption of the NIC device in an idle state, $P_{max}$ is the maximum power consumption of the device and $pps$ is the packets per second value of the device. The values used in the simulations for $P_{idle}$ and $P_{max}$ can be found in Sect. 5. These values are derived from the results reported in [27]. In this paper we consider the NIC power consumption of the servers, RMS and the router. For the router we also compute the power due to switching operations in addition to the NIC power. The formula is similar as in (6) except that $pps$ is the total pps of the router through all of its NIC devices.

#### 4.2.5 Mainboard

The mainboard being the central printed circuit board that holds many of the crucial components of the server, then its power consumption is given by the following equation:

$$P_{Mainboard} = \sum_{i=1}^{l} P_{CPU} + P_{RAM} + \sum_{j=1}^{m} P_{NIC} + \sum_{k=1}^{n} P_{HDD} + c, \tag{7}$$

where $l$ denotes the number of processors whose power consumption is $P_{CPU}$ of (2), $P_{RAM}$ is the memory power consumption of (4), $m$ indicates the number of network interface cards whose power consumption is $P_{NIC}$ of (6), $n$ denotes the number of attached hard disk drives whose power consumption is $P_{HDD}$ of (5), whereas $c$ is constant having a value of 40 W.

#### 4.2.6 Fan

The power consumption of fans was measured by directly attaching the cabling to the measurement device (power meter). The benchmark "SpeedFan" [21] was used to detect the current revolutions per minute (RPM) of the fan in measurement. The fan's RPM were then manually adjusted using a variable resistor. For a set of around 8 RPM settings per fan the power consumption was measured (10 minutes average). Based on the above described observations, the power consumption of a fan is given by the following 4th order polynomial:

$$P_{Fan} = 8.33068 \times 10^{-15} \times a^4 + 8.51757 \times w^4 - 2.9569 \times d^4 - 1.10138 \times 10^{-10} \times a^3 + 54.6855 \times w^3 - 76.4897 \times d^3 + 4.85429 \times 10^{-7} \times a^2 + 258.847 \times w^2 - 1059.02 \times d^2 - 6.06127 \times 10^{-5} \times a + 32.6862 \times w + 67.3012 \times d - 5.478 \quad (8)$$

where $w$ denotes the width (in mm), $d$ indicates the depth (in mm), and $a$ presents the revolutions per minute.

#### 4.2.7 Power supply unit

The power supply unit (PSU) being the only means of supplying power to the numerous components of the server, then its power consumption is given by the following equation:

$$P_{PSU} = \frac{P_{server}}{PSU_{count} \times e} \times 100 - \frac{P_{server}}{PSU_{count}}, \quad (9)$$

such that $P_{server}$ indicates the power consumption of all the components of the mainboard as well as fans of the server that have been calculated using previously introduced formulas, $e$ denotes the efficiency of the PSU provided by the manufacturer's data sheet, and $PSU_{count}$ represents the number of PSUs providing power to the server.

#### 4.2.8 Server power

The total power consumption of the server is a sum of the power consumptions of its constituent components. The power consumption of the data center is a sum of total consumption of each server, RMS, and network router/switch. When the above mentioned power consumption formulas are multiplied by the time, we get the corresponding energy consumption formulas ($E = P \times t$).

## 5 Simulation scenario

The basic network and data center topologies are depicted in Figs. 1 and 2. We used 20 clients and 32 servers in the simulations. Each client generated 20 job requests, i.e. totally 400 jobs, and the simulation was ended after all the jobs were finished.

Table 3 shows the parameters for each server. Servers are considered to be homogeneous, with equal characteristics. Parameters for the router and network are in Table 4. The NIC max and idle power values are derived from the results reported in [27]. For the power off threshold presented in Sect. 3, we chose to use value of 50 s. The power off threshold is used for defining when it is appropriate to power off servers. If there are more than 50 s before the estimated start of the first job in the queue, idle servers are powered off. Table 5 presents the job requirement parameters. A job can require randomly 1, 2 or 4 cores. Other requirement param-

**Table 3** Server parameters

| Parameter | Value |
|---|---|
| RAM size | $4 \times 2$ GB $= 8$ GB |
| RAM vendor | Kingston |
| RAM type | $DDR_2$ 800 MHz, un-buffered |
| Number of CPUs | 2 |
| CPU idle power | 15 W |
| CPU Architecture | AMD |
| Cores per CPU | 2 |
| Core frequency | 2.4 GHz |
| Core voltage | 1.2 V |
| Operating System | Linux |
| Number of PSUs | 1 |
| PSU efficiency | 70% |
| NIC power max | 0.09093495 W |
| NIC power idle | 0.090807 W |
| HDD power idle | 4.0 W |
| HDD power loaded | 5.6 W |
| HDD power standby | 0.5 W |
| HDD power start-up | 14.8 W |
| HDD start-up time | 10 s |
| Fan max rpm | 12000 |
| Fan actual rpm | 6000 |
| Fan power idle | 0.5 W |
| Fan power max | 2.0 W |
| Fan width | 600 mm |
| Fan depth | 300 mm |
| Power off threshold $T$ | 50 s |

**Table 4** Router and network parameters

| Parameter | Value |
|---|---|
| Data center network | 10 GB/s Ethernet |
| NIC power max | 69.316105370000002 W |
| NIC power idle | 69.316 W |

**Table 5** Job requirement parameters

| Parameter | Value |
|---|---|
| Cores | Random (1, 2, 4) |
| Core load (%) | Uniform (30, 99) |
| Nodes | Uniform (1, 5), Uniform (1, 10), Uniform (1, 32) |
| Memory | Uniform (100 MB, 2 GB) |
| Wall time | Uniform (600 s, 86400 s) |
| HDD Loaded time | Uniform (5%, 100%) |
| Idle interval | Uniform (100 s, 300 s) |

eters are based on uniform distributions. Core load is the amount of load the job puts on the processor cores it requires on the server. After a job is finished, clients stay idle until the period specified by the *idle interval* is reached. *Loaded time* is the fraction of the job execution time that the HDD spends in a loaded state.

## 6 Results

This section goes through the results from the simulation experiments and the results obtained by testing the scheduling algorithms in a real world testbed.

### 6.1 Simulation results

The metrics of interest for the simulation evaluation are the energy savings achieved with the developed energy-aware scheduler, the average wait time for jobs, and the average simulation duration to execute all the 400 jobs. All the metrics are evaluated with the 6 different scheduling algorithms (FIFO, BFF, BBF, E-FIFO, E-BFF, E-BBF), and 3 types of jobs depending on the node requirements (only small jobs requiring 1–5 nodes, medium size jobs requiring 1–10 nodes, or mixture of small and big jobs requiring 1–32 nodes). Each simulation run was repeated 10 times with a different seed.

Figure 3 shows the energy savings when comparing the standard scheduling algorithms to their energy-aware versions. The highest energy saving of 16% can be achieved with energy-aware FIFO (E-FIFO) when the clients send job requests requiring 1–32 nodes. The cause for this is that there are more idle servers during a simulation run than with the backfilling algorithms, which try to fill in jobs to be executed from the queue. Other energy savings with different configurations are about 6–10%. Consequently, the savings in energy are notably dependent on the system utilization. The higher the system utilization is, the less energy can be saved. Subsequent work on this topic would be to consider the option of employing the DVFS technique at a low system activity, in addition to the energy-aware scheduling algorithms described in this paper.

In Figs. 4 and 5 we can see the energy consumption with the different scheduling algorithms. It is clear from

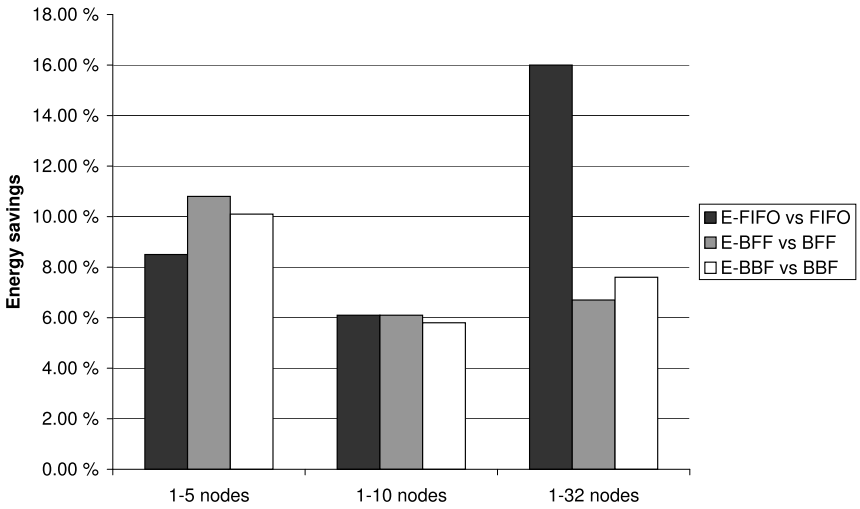



**Fig. 3** Energy savings for different job sizes and algorithms

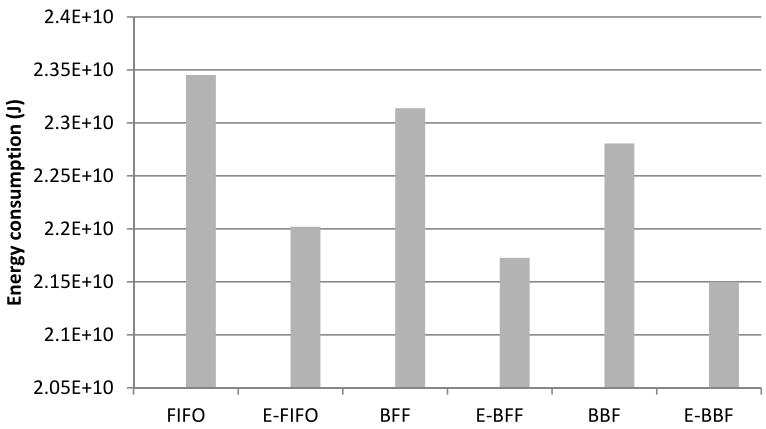



**Fig. 4** Energy consumption (J) with 1–10 node requirements

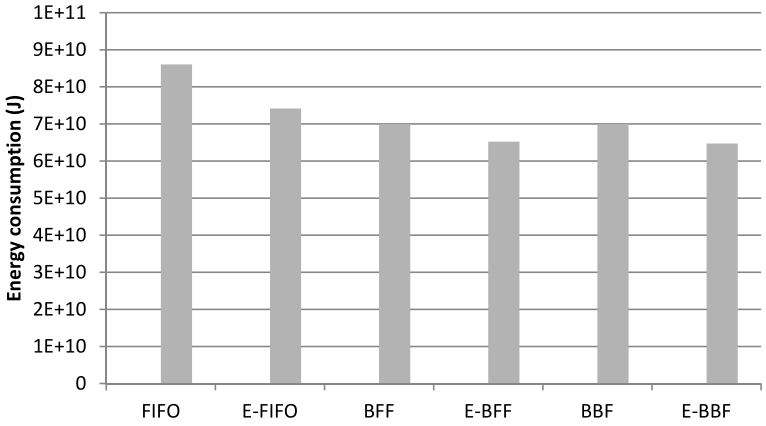



**Fig. 5** Energy consumption (J) with 1–32 node requirements

**Table 6** Wait time (s)

| Scheduler | 1–5 nodes | 1–10 nodes | 1–32 nodes |
|---|---|---|---|
| FIFO Average | 10 406 | 62 140 | 374 550 |
| FIFO Min | 8362 | 54 386 | 349 972 |
| FIFO Max | 13 111 | 71 742 | 387 628 |
| E-FIFO Average | 10 748 | 62 267 | 377 609 |
| E-FIFO Min | 8076 | 56 868 | 351 774 |
| E-FIFO Max | 13 802 | 73 963 | 400 638 |
| BFF Average | 10 256 | 55 789 | 259 616 |
| BFF Min | 8607 | 50 708 | 241 919 |
| BFF Max | 12 052 | 64 011 | 279 104 |
| E-BFF Average | 10 120 | 56 037 | 260 162 |
| E-BFF Min | 7635 | 48 677 | 242 725 |
| E-BFF Max | 12 854 | 69 220 | 279 998 |
| BBF Average | 10 102 | 54 953 | 256 751 |
| BBF Min | 8380 | 50 359 | 243 001 |
| BBF Max | 12 052 | 61 507 | 274 724 |
| E-BBF Average | 10 120 | 55 625 | 256 674 |
| E-BBF Min | 7635 | 51 211 | 245 694 |
| E-BBF Max | 12 854 | 62 980 | 280 636 |

**Table 7** Simulation duration (s)

| Scheduler | 1–5 nodes | 1–10 nodes | 1–32 nodes |
|---|---|---|---|
| FIFO Average | 1 251 524 | 2 263 359 | 8 569 376 |
| FIFO Min | 1 178 433 | 2 148 559 | 8 129 291 |
| FIFO Max | 1 346 262 | 2 488 887 | 8 871 941 |
| E-FIFO Average | 1 263 672 | 2 263 112 | 8 624 335 |
| E-FIFO Min | 1 165 143 | 2 127 663 | 8 173 664 |
| E-FIFO Max | 1 389 054 | 2 525 291 | 9 073 891 |
| BFF Average | 1 265 439 | 2 228 446 | 6 760 934 |
| BFF Min | 1 222 715 | 2 028 882 | 6 371 523 |
| BFF Max | 1 322 370 | 2 400 136 | 7 212 451 |
| E-BFF Average | 1 249 461 | 2 241 027 | 6 838 423 |
| E-BFF Min | 1 122 534 | 2 057 437 | 6 435 711 |
| E-BFF Max | 1 333 313 | 2 547 580 | 7 302 678 |
| BBF Average | 1 272 054 | 2 193 420 | 6 600 031 |
| BBF Min | 1 222 715 | 2 021 550 | 6 357 118 |
| BBF Max | 1 322 370 | 2 501 352 | 7 232 223 |
| E-BBF Average | 1 249 461 | 2 207 088 | 6 756 830 |
| E-BBF Min | 1 122 534 | 2 087 254 | 6 298 219 |
| E-BBF Max | 1 333 313 | 2 410 314 | 7 288 894 |

the figures that the energy-aware BBF (E-BBF) algorithm is the most energy-efficient scheduling algorithm and standard FIFO consumes the most amount of energy. The energy savings with E-BBF compared to FIFO are 9.1% (Fig. 4) and 33% (Fig. 5). However, backfilling itself can also reduce the energy consumption of the system compared to FIFO. This is because backfilling exploits the idle nodes for running shorter jobs while with FIFO they are left in an idle state.

Table 6 indicates the minimum, maximum and average wait times. As the node requirements increase, so does the average wait time. It can also be noted that backfilling decreases the wait time: with 1–32 node requirements and using backfill best fit the decrease is approximately 31% compared to standard FIFO. The table points out that there is no significant increase in wait time when using the energy-aware scheduling algorithms. The increase is at highest only 3.2% for FIFO with 1–5 nodes requirements.

The minimum, maximum and average simulation duration for executing all the 400 jobs requested by the clients can be seen in Table 7. Here we can see that the energy aware scheduling algorithms do not significantly increase the duration (2.3% on average at highest). On average, backfilling can decrease the duration by 23% (with 1–32 node requirements) compared to FIFO.

**Table 8** Juggle testbed parameters

| Parameter | Value |
|---|---|
| Number of nodes | 4 |
| CPUs per node | 2 |
| Cores per CPU | 2 |
| Core frequency | 2.4 GHz |
| CPU Architecture | AMD Opteron F2216 |
| Operating System | Linux |
| CPU Idle Power | 95 W |
| RAM size | $4 \times 8 \times 1$ GB $= 32$ GB |
| RAM Vendor | Kingston |
| RAM type | $DDR_2$ 667 MHz, unbuffered |

### 6.2 Testbed results

The energy-aware scheduler was also implemented and tested on the 'Juggle' (Table 8) cluster at Jülich Supercomputing Centre (JSC) [15]. In typical HPC scenarios simulation programs are developed which need a lot of processing power. Thus, the programs need to be parallelized in order to exploit multiple CPUs simultaneously. The testing environment at JSC tries to simulate such a typical usage of a supercomputer by providing a workload generator that submits characteristic user jobs. The test jobs are particularly CPU intensive LINPACK-based [24] simulations solving linear equations and linear least-square problems. Additionally, there are RAM- and IO-intensive test jobs stressing the available memory and file system performance.

The workload can be configured using several stress levels for each type of job (CPU, Memory, IO), the resources to be used (number of nodes, number of cores, and the required amount of memory), the expected time for completion and the number of job instances to be created.

From the LINPACK package a C-version of a linear equation solver is adopted. In a first step the coefficient matrix is factored by Gaussian elimination with partial pivoting which represents the most time consuming part of order $n^{**}3$ where $n$ is the matrix size. Then, the system is solved by using the computed matrix factors (complexity of $n^{**}2$). The linpack-based job is adapted to the configured stress level by adjusting the dimension of the corresponding matrices and vectors. Using OpenMP [30], the number of threads is set according to the configured number of cores to be used per node and MPI [12] is applied for the setup of multi-node jobs.

For the RAM stressing jobs the STREAM benchmark [25] is adopted which measures the sustainable memory bandwidth and the corresponding computation rate for vector kernels like copy, scale, add, and the triad. Again, OpenMP and MPI is employed for the creation of multi-thread and multi-node jobs.

**Table 9** Testbed results

| | Torque Scheduler | E-BFF |
|---|---|---|
| Elapsed time | 2049 s | 2062 s |
| Energy consumption | 1600 kJ | 1500 kJ |
| Avg. power consumption | 781 W | 729 W |

For the I/O part the Bonnie++ benchmark [22] is used which aims at performing a number of tests of hard drive and file system performance.

The generated workload takes about 35 minutes computing time and utilizes constantly 75–100% of the available cores of the test cluster. This stress test can be considered as adequate to get reliable results in respect of relative load average and appropriate power consumption in HPC environments.

The installed default scheduler of Torque RMS [29] processes the incoming jobs in the queue by requesting the statuses of the compute nodes and submitting them to the appropriate resources. With the default configuration, the used scheduling algorithm is closer to BFF than strict FIFO. A measurement script has been developed to capture the elapsed time from starting the test jobs until the completion of all jobs. Additionally, the average power consumption of the test cluster is measured during this period. A Raritan [9] smart power strip connected with the power cables of the compute nodes allows to measure the power consumption of the nodes. The current wattage of each connected node is polled in intervals of three seconds.

For the comparison, the same stress test was run with the developed energy-aware job scheduler including an HPC communication plugin to control the Torque RMS. The HPC communication plugin requests the RMS periodically on a pre-defined interval about statuses of nodes and jobs in the queue, and as soon as the plugin detects new jobs in the queue, or the status of jobs and nodes changes, it contacts the scheduler for performing an energy-aware resource management. In the stress tests an interval of 30 seconds was used. The strategy for power savings is based on setting idle nodes in low-power standby mode in case there are no jobs queued which could make use of them. The standby mode consumes 50 W less power compared to the idle state. The resume from standby to idle mode takes 5 seconds. Table 9 compares the results from the standard scheduler with the developed scheduler with E-BFF algorithm. Using the same workload results in a slightly increased elapsed time (+0.63%), however contrasted by an energy saving of 6.3%.

## 7 Conclusions and future work

This work presented an energy-aware scheduler that can be applied to HPC data centers without any changes in hard-

ware.With the simulations we achieved an energy saving of 6–16% depending on the system utilization and scheduling algorithm. Moreover, the energy-aware scheduling algorithms did not increase the average wait time or the average simulation duration dramatically.

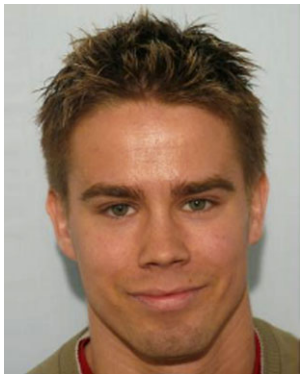
By testing the energy-aware scheduler in a real HPC test cluster we accomplished an energy saving of 6.3% with only a moderate increase in completion time (+0.63%). In the testbed experiments the utilization of the available cores of the test cluster was approximately 75–100%. The experiments in the test cluster confirm that our energy-aware scheduler is applicable in providing energy savings.

On the whole, the results look promising and our future work includes investigating the possibility of applying DVFS technique when appropriate, input/output data compression, and different varieties of the backfill best fit algorithm with regards to energy. We also plan to try out different low power states, such as standby or hybernated.

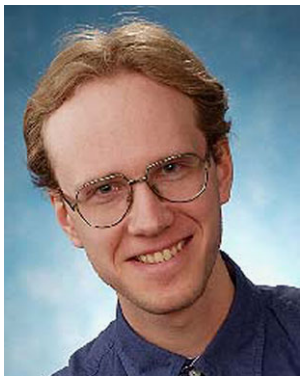
**Acknowledgements** This work was supported by EU FP7 project FIT4Green (www.fit4green.eu). The authors would like to thank all the colleagues working in the project. The comments from the anonymous reviewers are also gratefully acknowledged.

## References

1. http://www.hynix.com/products/consumer/consumer_sub.jsp?menuNo=1&m=2&s=1&menu3=01&RK=03&RAM_NAME=DDR2%20SDRAM&SUB_RAM=1Gb
2. http://www.kingston.com/hyperx/products/khx_ddr2.asp
3. http://www.samsung.com/global/business/semiconductor/productList.do?fmly_id=696&xFmly_id=695
4. Bailey Lee C, Schwartzman Y, Hardy J, Snavely A (2005) Are user runtime estimates inherently inaccurate? In: Job scheduling strategies for parallel processing. Lecture notes in computer science, vol 3277, pp 253–263. Springer, Berlin. http://dx.doi.org/10.1007/11407522_14
5. Barroso L, Holzle U (2007) The case for energy-proportional computing. Computer 40(12):33–37. doi:10.1109/MC.2007.443
6. Basmadjian R, Ali N, Niedermeier F, de Meer H, Giuliani G (2011) A methodology to predict the power consumption of servers in data centers. In: Proceedings of the 2nd international conference on energy-efficient computing and networking 2011 (e-Energy). ACM, New York
7. Bianzino A, Chaudet C, Larroca F, Rossi D, Rougier J (2010) Energy-aware routing: a reality check. In: GLOBECOM workshops (GC Wkshps). IEEE Press, New York, pp 1422–1427. doi:10.1109/GLOCOMW.2010.5700172
8. Cirne W, Berman F (2001) A comprehensive model of the supercomputer workload. In: IEEE international workshop on workload characterization (WWC-4), pp 140–148. doi:10.1109/WWC.2001.990753
9. http://www.raritan.de/px-5000/px-5528/tech-specsde/: Raritan
10. Etinski M, Corbalan J, Labarta J, Valero M (2010) Utilization driven power-aware parallel job scheduling. Comput Sci Res Dev 25:207–216. http://dx.doi.org/10.1007/s00450-010-0129-x. 10.1007/s00450-010-0129-x
11. Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th annual international symposium on computer architecture (ISCA'07). ACM, New York, pp 13–23. http://doi.acm.org/10.1145/1250662.1250665
12. http://www.mpiforum.org/docs/: Mpi
13. Freeh VW, Lowenthal DK, Pan F, Kappiah N, Springer R, Rountree BL, Femal ME (2007) Analyzing the energy-time trade-off in high-performance computing applications. IEEE Trans Parallel Distrib Syst 18:835–848. doi:10.1109/TPDS.2007.1026
14. Ge R, Feng X, Song S, Chang HC, Li D, Cameron K (2010) Powerpack: energy profiling and analysis of high-performance systems and applications. IEEE Trans Parallel Distrib Syst 21(5):658–671. doi:10.1109/TPDS.2009.76
15. Girolamo MD, Giuliani G, Egea JCL, Homberg W, Giesler A, Lent R, Mahmoodi T, Sannelli D, Salden A, Georgiadou V, Dang MQ, de Meer H, Basmadjian R, Klingert S, Schulze T (2011) Pilot evaluation of energy control plug-in inside single data centers. Deliverable D-6.2 v8.0, FIT4Green. FP7-ICT-2009-4-249020–FIT4Green/D-6.2, http://www.fit4green.eu/
16. Hikita J, Hirano A, Nakashima H (2008) Saving 200 KW and $200 k/year by power-aware job/machine scheduling. In: IEEE international symposium on parallel and distributed processing (IPDPS 2008), pp 1–8. doi:10.1109/IPDPS.2008.4536218
17. Hsu Ch, Feng Wc (2005) A power-aware run-time system for high-performance computing. In: Proceedings of the 2005 ACM/IEEE conference on supercomputing (SC'05). IEEE Computer Society, Washington, p 1. http://dx.doi.org/10.1109/SC.2005.3
18. http://alasir.com/software/ramspeed/: Ramspeed
19. http://inet.omnetpp.org/: Inet framework
20. http://omnetpp.org/: Omnet++ network simulation framework
21. http://www.almico.com/speedfan.php: Speedfan
22. http://www.coker.com.au/bonnie++/: Bonnie++
23. http://www.gartner.com/it/page.jsp?id=503867: Gartner newsroom—press release
24. http://www.netlib.org/linpack/: Linpack
25. http://www.streambench.org/: Stream
26. http://www.top500.org/lists/2010/11/highlights: Top500 supercomputer sites
27. Lent R (2010) Simulating the power consumption of computer networks. In: 15th IEEE international workshop on computer aided modeling, analysis and design of communication links and networks (CAMAD), pp 96–100. doi:10.1109/CAMAD.2010.5686955
28. Liu Y, Zhu H (2010) A survey of the research on power management techniques for high-performance systems. Softw Pract Exp 40:943–964. http://dx.doi.org/10.1002/spe.v40:11
29. http://www.clusterresources.com/pages/products/torque-resourcemanager.php/: Torque
30. http://www.openmpi.org/: Openmp
31. Pinheiro E, Bianchini R, Carrera EV, Heath T (2001) Load balancing and unbalancing for power and performance in cluster-based systems. In: Proceedings of the workshop on compilers and operating systems for low power (COLP)
32. Ranganathan P, Rivoire S, Moore J (2009) Models and metrics for energy-efficient computing. Adv Comput 75:159–233
33. Restrepo J, Gruber C, Machuca C (2009) Energy profile aware routing. In: IEEE international conference on communications workshops, pp 1–5. doi:10.1109/ICCW.2009.5208041
34. Rivoire S, Ranganathan P, Kozyrakis C (2008) A comparison of high-level full-system power models. In: Proceedings of the conference on power aware computing and systems (HotPower'08). http://portal.acm.org/citation.cfm?id=1855610.1855613
35. Rivoire S, Shah M, Ranganatban P, Kozyrakis C, Meza J (2007) Models and metrics to enable energy-efficiency optimizations. Computer 40(12):39–48. doi:10.1007/s00450-011-0189-6



**Olli Mämmelä** is a Research Scientist at the VTT Technical Research Centre of Finland in Oulu, Finland. He received his M.Sc. in information technology from the Department of Electrical and Information Engineering at Oulu University in 2010. His research interests include mobile computing, future Internet, energyefficient computing, and network simulations and modelling.



**Mikko Majanen** joined VTT Technical Research Centre of Finland in 2000 and he is currently working there as a research scientist in the Seamless networking team. He received his M.Sc. in physics from the University of Oulu, Finland in 2003. He is currently studying towards another M.Sc. degree in information engineering in the same university. His research interests include mainly network simulations.

**Robert Basmadjian** received his Ph.D. degree in 2008 on the topic of "An Arbitrary Tree-Structured Replica Control Protocol" from University of Toulouse, France. Since 2009, he has been in postdoc position at the Chair of Computer Networks and Communications headed by Professor Hermann De Meer in University of Passau. His main research interests include replication in large-scale distributed systems as well as energy efficiency of distributed systems. Within the context of energy efficiency, he is an active member of Energy efficiency in large scale distributed systems community as well as he is involved in EU project FIT4Green.

**Hermann De Meer** received his Ph.D. in 1992. He had been an Assistant Professor at Hamburg University, Germany, a Visiting Professor at Columbia Universityin New York City, USA, and a Reader at University College London, UK. Heis currently appointed as Full Professor at the University of Passau, Germany, and as Honorary Professor at University College London, UK. Heis director of the Institute of IT Security and Security Law (ISL) atthe University of Passau. His main research interests include IT security and resilience, virtualization and energy efficiency, complex and self-organizing systems, peer-to-peer systems, quality of service and performance modeling, Internet protocols, home networking, and mobile computing. Hermann de Meer has led several nationally and internationally funded projects on Performance Modeling and Computer Networking. He currently holds several research grants funded by the Deutsche Forschungsgemeinschaft (DFG) and by the EU (FP6 and FP7). Prof. H. de Meer is co-authoring a textbook on "Queueing Networks and Markov Chains–Modeling and Performance Evaluation with Computer Science Applications", published by John Wiley in 1998 and 2006.

**André Giesler** studied Computer Science in Civil Engineering and received his diploma degree from the Technische Universität of Darmstadt. He is working since 2006 as a scientific assistant at the Jülich Supercomputing Centre (JSC) of Forschungszentrum Jülich (FZJ). He developed several components to enhance the functionality of the UNICORE Grid Middleware during the European project DEISA and was involved in the ETICS project as well. His research interests are in the field of distributed software architectures, data management, and energy efficient HPC computing.

**Willi Homberg** started in 1978 at the Central Institute for Applied Mathematics (ZAM) of Research Centre Jülich where he was primarily engaged in the education of mathematical-technical assistants. In 1990, he moved to data center operations. He was responsible for UNIX servers and Linux compute clusters. Since 2006 he works in the department Technolology at Jülich Supercomputing Centre (JSC). Currently he is leader of the work package energy efficiency in "Exascale Innovation Center", a cooperation of IBM and JSC aiming at the development of future exascale supercomputer systems.