Pricing schemes for energy-efficient HPC systems: Design and exploration

Pricing Schemes for Energy-Efficient HPC Systems: Design and Exploration

Andrea Borghesi[a,b], Andrea Bartolini[b], Michela Milano[a], Luca Benini[b,c]

[a] *DISI, University of Bologna. Viale Risorgimento 2, 40123, Bologna, Italy*
[b] *DEI, University of Bologna. Viale Risorgimento 2, 40123, Bologna, Italy*
[c] *Integrated Systems Laboratory at ETH Zurich, Switzerland*

Abstract

Energy efficiency is of paramount importance for the sustainability of HPC systems. Energy consumption limits the peak performance of supercomputers and accounts for a large share of total cost of ownership. Consequently, system owners and final users have started exploring mechanisms to trade off performance for power consumption, for example through frequency and voltage scaling.

However, only a limited number of studies have been devoted to explore the economic viability of performance scaling solutions and to devise pricing mechanisms fostering a more energy-conscious usage of resources, without adversely impacting return-of-investment on the HPC facility. We present a parametrized model to analyze the impact of frequency scaling on energy and to assess the potential total cost benefits for the HPC facility and the user. We evaluate four pricing schemes, considering both facility manager and the user perspectives. We then perform a design space exploration considering current and near-future HPC systems and technologies.

Keywords: High Performance Computing, Energy-Efficiency, Power Consumption, Pricing Schemes

1. Introduction

Energy consumption poses a great challenge for the growth of worldwide HPC installations. As supercomputers increase their peak performance, so do their power consumption, leading in turn to increased energy costs. Hence, the last few years saw a shift from a "performance at all costs" mentality to a more balanced and energy efficient perspective [1, 2, 3].

Several methods aim at curtailing the power consumption through a trade-off between the computing performance and power consumption, for example via frequency and/or voltage scaling (*DVFS*) [4]. The main drawback of this technique is the decreased computing speed that leads to increased application run times. This issue is partially mitigated because several HPC applications and benchmarks are not CPU-bound but present a memory and I/O bottleneck [5, 6, 7]; reducing the frequency of the computing units used by these jobs does not impact severely their time-to-solution (TtS)[8]. For instance, memory or I/O bound application are less sensitive to power reduction. See differences between CPU-heavy benchmarks such as HPL[9] and the memory bandwidth constrained HPCG[10]. While in the rest of the paper we will refer explicitly to frequency scaling, our conclusions can also be applied Intel's *Running Average Power Limit* (RAPL) [11], that does not directly change the computing nodes clock frequency but indirectly does so by enforcing a socket-level power cap. This technique is analogous to DVFS since the power bound leads to increased run times [12, 13].

While applications of DVFS in power capped contexts have been widely studied, very little attention has been dedicated to the economic aspect of the frequency scaling. For example, a very common accounting scheme in HPC centers consists in linking the price paid by final users to the time-to-solution of their application multiplied by the requested resources [14]; this scheme is therefore directly affected by techniques altering the applications run time. The rapid depreciation of computing facilities pushes against any policy that stretches job execution time. Moreover, decreasing the computing unit performance leads to lower power consumption, but this does not guarantee lower energy consumption, due to the longer durations.

In this paper we take steps to address these issues. We introduce a parameterized model representing a HPC system, based on a real Top 500 supercomputer on the tier-0 *Fermi* supercomputer, hosted at the CINECA computing center[15]. We use the model to understand the economic impact of frequency scaling, from the point of view of both the facility manager (maximizing the overall gain and re-

Email addresses: andrea.borghesi3@unibo.it (Andrea Borghesi), a.bartolini@unibo.it, barandre@iis.ee.ethz.ch (Andrea Bartolini), michela.milano@unibo.it (Michela Milano), luca.benini@unibo.it,luca.benini@iis.ee.ethz.ch (Luca Benini)

ducing the total cost of ownership – also called TCO) and of users (minimizing the costs paid for resource per hour). We present four different pricing schemes and we evaluate their economic viability, given the parameters characterizing the *Fermi* supercomputer and the hosting facility. We consider how DVFS impacts both the energy costs (The electricity cost paid by the facility to operate the IT infrastructure plus the cooling system) and the generated income; we explore mechanisms that can be used to foster a reduction in energy costs while maintaining a profitable condition for both users and owners. We also extend our parametric analysis considering how the pricing schemes could generate different outcomes with different systems and operating conditions.

The rest of the paper is organized as follows: Section 2 provides an overview on the related works in the area of frequency scaling in HPC and a brief discussion on energy-aware pricing schemes found in the data center literatures. Section 3 describes the parameterized models and evaluates the proposed pricing schemes. Section 4 discusses the alternative scenarios and explores the design space. Finally, Section 5 summarizes the paper and provides the concluding remarks.

2. Related Works

In this section we briefly describe the state-of-the-art techniques aiming at energy efficiency (in particular frequency scaling). We then present an overview of the literature regarding pricing schemes found in data centers and targeted at fostering energy efficient solutions.

2.1. Power/Energy Efficiency

Since the HPC community widely recognizes the need to reduce power consumption in supercomputers, several research avenues have been explored for this purpose. Many techniques have been proposed to bound the power consumption of HPC machines, ranging from Dynamic Voltage and Frequency Scaling (DVFS) [16], energy proportional systems [17], over-provisioning [18], turning off idle resources [19], exploiting components variability [20]. In this paper we are going to focus on frequency scaling and socket-level power capping (RAPL) because they are well-known solutions that have been adopted in several HPC systems [21, 22, 4, 23, 24, 25].

Nowadays, many supercomputers employ some form of DVFS[22, 26], i.e. they exchange processor performance for lower power consumption. With DVFS, a processor can run at one of the supported frequency/voltage pairs lower than the maximum one. The main issue with DVFS-based approaches is the trade-off between power savings and decrease in performance: reducing the clock frequency clearly increases the TtS of the applications. To overcome this issue, several methods try to apply DVFS only in periods of low system activities or in particular phases of a job execution. For example, in [27], Freeh et al. study the energy-time trade-off of high performance cluster nodes with several power states available. They conclude that applying DVFS to applications with memory or communication bottlenecks does not imply large time penalties. This strategy strongly relies on the nature of the running applications, which must be known and modeled in advance, before their actual execution. In [28], Hsu et al. propose to solve this problem through a power-aware *adaptive* algorithm which does not employ any application-specific information a priori, but implicitly gathers such information at run-time.

Etinski et al. [29] extend the well-known EASY-backfilling scheduling policy to limit a supercomputer power consumption through DVFS. Their results are promising in terms of energy savings and also a better utilization of the system and reduced waiting time for the users, thanks to the possibility to execute more jobs concurrently if their frequency (thus power) is reduced. The same authors introduce also another approach in [30]: in the latter work they propose a novel scheduling policy based on integer linear programming (ILP). This method offers better performance in terms of average job wait time over various power budget. These two works focus exclusively on the effect of frequency scaling on applications run times while we are mainly concerned with the energy consumption and its economic impact.

RAPL provides a software configurable and hardware enforced power cap. Instead of setting a specific frequency, this mechanism takes as input the power budget for a socket and subsequently forces the power consumption to be within the limit. For instance, Ellsworth et al. [31] present a scheme to decide the power allocated to each node in a supercomputer (*Dynamic Power Sharing*). Initially the overall available power budget is uniformly divided among all nodes; periodically the algorithm adjusts the allocated power depending on actual consumptions, i.e. if a node consumes less power than the allocated one the exceeding capacity can be transferred to a different node which needs it. RAPL is used to enforce the node power limit at run time. The main drawback of RAPL is the same that troubles DVFS mechanisms, namely the indiscriminate power reduction implies an increase in TtS (performance loss).

The main limitation of the related works in the research literature is that they focus (almost) exclusively on the energy-savings and time-to-solution considerations while discounting the cost aspects. All the considered approaches can influence the HPC system revenues exclusively through the reduction of energy/power spending and therefore overlook a critical component of the facility costs, the non negligible depreciation costs. In our paper we consider both elements that determine the supercomputer TCO.

Real HPC applications have different sensitivities towards frequency & voltage scaling; memory or I/O bound application are less sensitive to frequency reduction. For instance, see differences between CPU-heavy benchmarks

such as HPL[9] and the memory bandwidth constrained HPCG[10]. We consider for simplicity an "average" job sensitivity and sweep it as a parameter.

2.2. Pricing Schemes

Another important area of research deals with the problem of finding optimal pricing schemes for the resources composing a supercomputer. The current state-of-the-art for pricing schemes in HPC systems is somewhat lacking, whilst researchers in the data center community investigated this issue in a more thorough manner[32, 33, 34]. Generally speaking, data centers operate with a slightly different set of assumption w.r.t. HPC facilities and therefore they are not directly comparable to the method proposed in this paper.

Chase et al. [35] present a new architecture to manage resources in a data center, with the goal of energy efficiency. The main idea is to implement a bidding mechanism where the services running in the system bid for resources as a function of delivered performance. Afterwards, resource prices are regulated through a greedy algorithm to balance supply and demand, allocating resources to their most efficient use.

Zhang et al. [36] consider the issue of minimizing the electricity bill of a network of data centers; for this purpose they devise an approach that leverage the different electricity prices in different geographical locations to distribute workloads among those locations. Their work explicitly models the effects of the power demands induced by cloud-scale data centers on electricity prices and the power consumption of cooling and networking in the minimization of electricity bill. Although the proposed solution is very interesting, the vast majority of nowadays HPC systems do not have a distributed nature similar to the one considered in this work.

Wang et al. [37] tackle the problem of optimizing data center electric utility bill under uncertainty in workloads and real-world pricing schemes. They consider a data center where the power consumption of the IT equipment can be modulated via control knobs. The key assumption of the model they propose is that the power effects of most IT control knobs can be seen as dropping and/or delaying a portion of the power demand, i.e. through dynamically modulating the workload. They propose a hierarchical infrastructure to manage system resources and workload; the hierarchical structure allows to separate the abstract layer specifying the optimization policy from the lower level that implements the actual power-modulation knobs. The main drawback back of this work (and several others found in the literature) is that it disregards the total cost of ownership and the depreciation costs.

3. The HPC System Model

In this section we introduce the parameterized model, used to describe the cost, energy, performance trade-off in a generic supercomputer. The parameters configuration considered in this section is based on the *Fermi* supercomputer[38]. The proposed model abstracts the ensemble of computing resources as a composition of allocable elements. As the considered system was composed of multi-cores, we referred to them as "core". This is done to simplify the analysis but nothing prevents the addition of different resources to the model to extend our approach. In our analysis we do assume that scheduling and allocation decisions have been taken by a higher-level scheduler. This is normal in supercomputer infrastructures[39, 40, 41, 42].

We assume that the considered machine is capable of decreasing the power consumption of computing units in exchange for reduced performance through frequency & voltage scaling, which may lead to an increased run-time of the involved applications, accordingly to their properties. We model the power consumption of each computing resource with two contributions: the idle power and the active power. The idle power is a constant power term needed to keep the resource on, the active power is only consumed when the resource is active and executes a job. The absolute value is proportional to the clock frequency. The dependency of the active power to the frequency is monotonic and superlinear with an exponent alpha dependent on the technology[43].

3.1. Model Description

The key parameters composing the model are listed in Table 1. From these base parameters we compute the values of a set of intermediate variables, presented in Table 2. In Table 3 we report the output, or target, parameters. The main output parameters are used to evaluate the pricing schemes discussed in Section 3.3. We chose two main parameters: 1) the system gain (the difference between the income obtained by the system owner and the operating costs); 2) the price paid by users for their application (measured as the price paid per hour and per single resource usage). These outcomes are relative to the considered time frame θ. The model parameters are linked through the mathematical expressions exposed in the tables. Some parameters are self explaining; we give here details to illustrate the less obvious ones. Part of the parameters presented in Table 1 describe the HPC facility. In our case, their values depend on the supercomputer we took as example; different configurations can model different systems. Other parameters are instead used to represent the applications.

There are two main parameters that define the behaviour of the system (how system gain and job price are affected) when frequency scaling is applied:

- the scaling factor, φ, indicates how much power consumptions are decreased (the same factor is applied to each slowed down jobs);
- the job sensitivity, σ, modulates the duration increase due to the power scaling (again, same value applied to all slowed jobs)

The system might be not fully used (not enough jobs, resource bottlenecks, SLAs constraints..) but its cores are occupied only up to a certain percentage U. In the proposed model we consider also the case where the power consumption is scaled down only for a fraction of the jobs; β tells the percentage of jobs that are not subject to slow down (conversely $1-\beta$ indicates the fraction of jobs with a reduced power consumption). The alpha factor α is a technology-dependent parameter and affects the reduction in power consumption following a frequency reduction. Given a core base power consumption at maximum frequency, the idle power percentage ι indicates the proportion of consumption due to the idle power (when the core is not used). Lower values of ι indicate a more energy proportional system, i.e. systems where power tends to near-zero values when frequency tends to zero.

The scaling factor, φ, specifies the ratio between the maximum and reduced frequency and it directly modulates the power consumption variation (decrease). It is a real number and, given a maximum frequency f_{max} and the scaled one f_{scaled}, is computed as $\varphi = f_{max}/f_{scaled}$. The job sensitivity, σ, modulates the time-to-solution increase due to the power scaling. The job sensitivity embeds both the nature of the application (ranging between CPU-bound or memory-bound) and the fact that a HPC job can be composed by several sub-tasks with relative dependencies: an application with many intertwined tasks may experience higher performance degradation when subject to frequency scaling.

The idle and active power consumed by each core at maximum frequency (P^I and P^A) are obtained by dividing the total energy consumed by the IT infrastructure – derived from the yearly energy cost C^Y_{EI} – by the total number of core and the hours of utilization. The power consumption of a job at maximum frequency is computed as the sum of idle and power consumption for each core (at maximum frequency) multiplied by the number of requested cores (ν_j). In Table 2 we also observe how the time-to-solution and the power consumption of a job change if the power is scaled down; the scaling factor φ and the job sensitivity σ are the only parameters affecting the outcome – we assume that α remains constant in the whole time frame (besides being the same for all cores).

The parameter R^a indicates the number of resources (only cores in our model) that are used in the system by the running applications; it is computed as the number of total resources available in the system multiplied by the system utilization U.

The table contains also the derived parameters which are directly involved in the computation of the final output variables, in particular the total cost, depreciation payment plus energy consumption, per time frame. We assume that the depreciation cost is constant in the time period C^θ_S, the energy cost for the cooling is proportional to the IT energy cost, the latter being the sum of the energy consumption of each job. As discussed earlier, only a percentage of jobs undergo a slow down, therefore the energy consumption of each job is a combination of TtS $\times$ power at maximum frequency (non-slowed down jobs) and TtS $\times$ power at scaled frequency (slowed down jobs). The sum of all job energies is multiplied by the electricity cost to infer the energy costs ($E_\epsilon/1000$). We assume that the energy costs are going to be identical for each pricing scheme presented in Section 3.3 (the pricing scheme influences only the system income and not its expenses).

The ROI is an input parameter representing the expected Return-On-Investment desired by the system owner. κ_T stands for the baseline hourly cost per resources, derived from ROI, depreciation and estimated energy cost. κ_E is defined similarly but discarding the energy cost. The maintenance costs and the value of money are embedded in the depreciation costs and Return-Of-Investment.

3.2. Energy Saving Potential

A fundamental aspect impacting the system cost – hence system gain and price paid by users – is the energy cost. We must consider two issues: 1) does decreasing speed (clock frequency) actually reduce energy consumption? if that is the case, certainly the energy cost would go down; 2) even if the above is true, does this lead to reduced system TCO? This may not happen because of depreciation costs. In this section we are going to answer to the first question, while Sec. 3.3 deals with the second issue.

In general, when we decrease the power consumption of a set of computational resources the HPC jobs that are using them will suffer a performance loss and thus they might require more time to complete. The power decrease and time-to-solution increase are clearly intertwined and their relation strongly depends on the nature of the application; for instance, a memory-bound application would experience a smaller TtS increase. This may lead to an actual energy consumption increase since the energy E associated to a job is computed as: $E = \pi \times \delta$, where π is the power consumption of the job and δ is its time-to-solution.

To answer the question we can analyze the ratio between the energy consumed by a job at maximum frequency and the energy consumed at the reduced frequency. The energy ratio value is expressed by the following equation:

$$
\begin{aligned}
E_{ratio} &= \frac{\pi_M \times \delta_M}{\pi_S \times \delta_S} = \frac{\nu_j(P^I + P^A) \times \delta_M}{\nu_j(P^I + \frac{P^A}{\varphi^\alpha}) \times (\delta_M + \delta_M(\varphi-1)\sigma)} \\
&= \frac{(\iota P + (1-\iota)\cdot P) \times \delta_M}{(\iota P + \frac{(1-\iota)\cdot P}{\varphi^\alpha}) \times (\delta_M + \delta_M(\varphi-1)\sigma)} \\
&= \frac{1}{(\iota + \frac{(1-\iota)}{\varphi^\alpha}) \times (1 + (\varphi-1)\sigma)}
\end{aligned} \tag{1}
$$

The numerator and the denominator represent, respectively, the energy consumed by an application at maximum frequency (TtS multiplied by power, $\pi_M \times \delta_M$) and the energy consumed at the reduced frequency ($\pi_S \times \delta_S$). The rest of the equation is obtained by substituting the TtS and power values with their corresponding expressions, as

Name	*Symbol*	*Unit*
Time frame	θ	Days
Number of cores in the system	NC^T	NA
Power Usage Efficiency	PUE	NA
Electricity cost	E_ϵ	€/ KWh
System lifetime	LF	Years
System installation cost	C_S^T	€
Estimated energy cost (IT) per year	C_{EI}^Y	€
Return On Investment (≥ 1)	ROI	NA
Percentage of system utilization	U	NA
Idle power as % of power at max. frequency	ι	NA
Alpha factor	α	NA
Job TtS (Time-to-Solution) at maximum frequency (estimate)	δ_M	Hours
Number of requested cores per job	ν_j	NA
Frequency scaling factor	φ	NA
Job sensitivity	σ	NA
Percentage of non-slowed jobs	β	NA

Table 1: Model Base Parameters

Name	*Symbol*	*Expression*	*Unit*
System cost per year (depreciation)	C_S^Y	C_S^T/LF	€
Cooling Energy Cost per Year	C_{EC}^Y	$C_{EI}^Y \cdot (PUE-1)$	€
IT energy cost - Lifetime	C_{EI}^T	$C_{EI}^Y \cdot LF$	€
Cooling energy cost - Lifetime	C_{EC}^T	$C_{EC}^Y \cdot LF$	€
Total energy cost - Lifetime	C_E^T	$C_{EI}^T + C_{EC}^T$	€
System cost (depreciation) - Time frame	C_S^θ	$C_S^Y/365 \cdot \theta$	€
Coefficient - Total	κ_T	$\frac{ROI \cdot C_S^T + C_E^T}{NC^T \cdot LF \cdot 24 \cdot 365}$	NA
Coefficient - System Only	κ_E	$\frac{ROI \cdot C_S^T}{NC^T \cdot LF \cdot 24 \cdot 365}$	NA
Core Power (max frequency)	P	$\frac{1000 \cdot C_{EI}^Y/E_\epsilon}{NC^T \cdot 365 \cdot 24}$	W
Core Idle Power (max frequency)	P^I	ιP	W
Core Active Power (max frequency)	P^A	$(1-\iota) \cdot P$	W
Job power consumption at max frequency	π_M	$\nu_j(P^I + P^A)$	W
Job TtS at scaled frequency	δ_S	$\delta_M + \delta_M(\varphi - 1)\sigma$	Hours
Job power consumption at scaled frequency	π_S	$\nu_j(P^I + \frac{P^A}{\varphi^\alpha})$	W
Number of resources active	R^a	$NC^T \times U$	# Cores

Table 2: Model Derived Parameters

described in the Tables 1 and 2. We assume that the parameters that do not appear in Eq. 1 have fixed values.

We observe two facts: 1) values ≥ 1 are better since they imply that the energy of the job decreases when we scale down its power; 2) the only involved variables are the alpha factor α, the idle power expressed as percentage of the total core power (at maximum frequency) ι, the scaling factor φ and the job sensitivity σ. To further simplify our analysis we now assume that the scaling factor is fixed to a particular value $\varphi > 1$ (it must be greater than one if we want to study the power savings effect); as we are going to see, setting the scaling factor to a constant value does not invalidate our conclusions.

In Figure 1 we have a three-dimensional plot representing the isosurface of value 1 corresponding to the energy ratio described in Eq. 1. The x-axis, y-axis and z-axis contain, respectively, the idle power percentage ι, the the job sensitivity σ and the alpha factor α. An *isosurface* is a surface that represents points of constant target value (it is the 3-d analog of an isoline or contour line); points above the isosurface have values larger than the target one, points below the surface have value smaller than the target. The red arrow indicates the volume of space formed by points above the isosurface. For example, in the graph of Fig. 1 the point with coordinates $(0.2, 0.2, 2.0)$ is above the isosurface, hence its corresponding energy ratio is larger than 1, which basically means that reducing the clock speed is convenient energy-wise; conversely, the point with coordinates $(0.8, 0.8, 1.5)$ is situated below the isosurface and corresponds to an energy ratio lower than 1.

Figure 1 reveals that indeed there are some combination of values for which reducing the job's power consumption

Name	*Symbol*	*Expression*	*Unit*
Income per time frame	I^θ	Sec. 3.3	€
IT energy cost per time frame	C^θ_{EI}	$[(1-\beta)\pi_S\delta_S + \beta\pi_M\delta_M] \cdot \frac{E_\epsilon}{1000}$	€
Cooling energy cost per time frame	C^θ_{EC}	$C^\theta_{EI} \cdot (PUE - 1)$	€
Total cost per time frame	C^θ_T	$C^\theta_S + C^\theta_{EI} + C^\theta_{EC}$	€
System gain per time frame	γ^θ	$I^\theta - C^\theta_T$	€
Average job price	χ^θ	$\frac{I^\theta \nu_j}{R^a}$	€

Table 3: Model Output Parameters

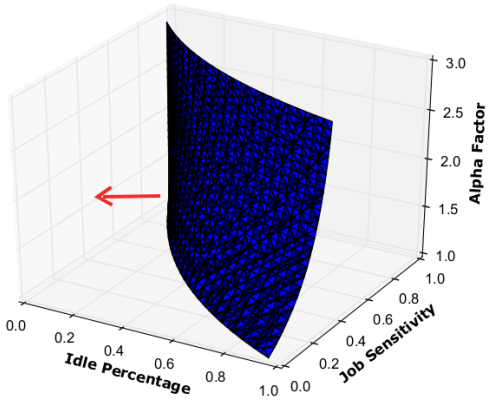


Figure 1: Energy Savings: isosurface with energy ratio = 1

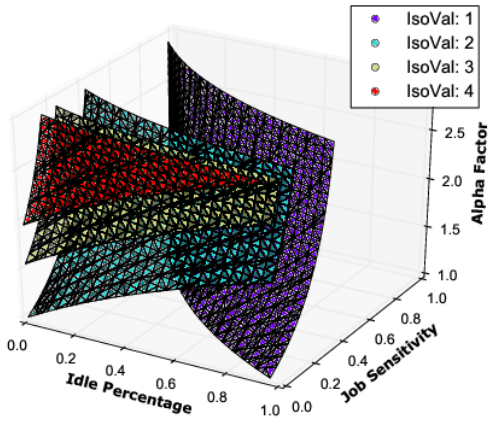


Figure 2: Energy Savings: multiple isosurfaces with different ratio values

leads to energy savings. For example, as one could have expected, low values of job sensitivity imply a larger energy ratio (saving): if the job TtS increases only marginally when the power is reduced the outcome is an energy saving. We can also notice that better (higher) energy ratios are associated to lower values of ι: this happens because if the idle power component has a relatively smaller influence, decreasing the operating frequency of the computing nodes leads to greater power savings – the idle power consumption is not affected by the scaling-down action. However, it is also clear that there are many configurations where frequency (and power) reduction does not reduce energy.

As a first result of the proposed model: cost reduction policies based on performance scaling make sense only if the system is operated in the area above the isosurface, defined by (ι, σ, α). σ depends on the application slack which is defined based on the target architecture and applications set. ι and α are instead technological parameters: α is determined by the technology while ι depends on the system architecture and on the leaking components present in the compute node (i.e. Fans, HDDs, NIC, etc).

In Figure 2 we displayed different isosurfaces along with the one corresponding to an energy ratio of 1. The additional isosurfaces correspond to energy ratios of 2, 3 and 4; as noted before, a higher energy ratio means more potential energy saving and thus combinations of (ι, σ, α) leading towards the new isosurfaces are preferable.

Figure 3 shows what happens if we also change the value of the scaling factor parameter φ; the figure presents again isosurfaces of value 1. As we anticipated before, the scaling factor influences the energy ratio as revealed by the different gradients of the surfaces but the overall shape of the isosurfaces remain similar. One thing that can be noticed is that when the scaling factor increases the alpha factor impact slightly decreases – the surface varies less along the z-axis.

From Fig. 1, 2 and 3 we can draw a positive conclusion. Reducing the power consumption of the application in a HPC system can lead to energy savings, depending on some the parameters characterizing the system and the application. As a general rule, we can say that facility owner as well as user should target the reduction of power consumption of the less sensitive jobs, i.e. those jobs whose time-to-solution will not be too affected by the power reduction (for example memory, I/O and communication bound applications). This conclusion is more prominent in installations in which the idle power is a large component of the total power consumption; In this case reducing the operational frequency can increase the consumed en-

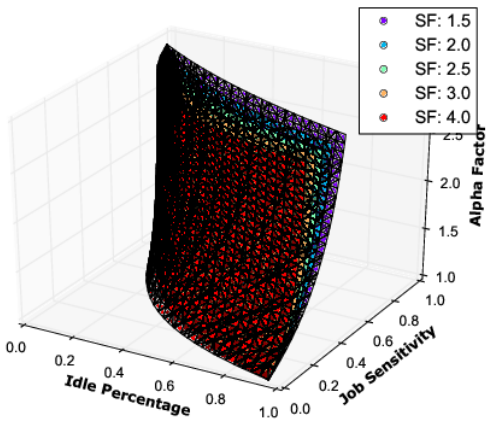


Figure 3: Energy Savings: isosurfaces with ratio = 1, different scaling factor φ values

ergy.

3.3. Pricing Schemes

The results of the previous section suggest that, depending on the characteristics of applications (jobs) and of the supercomputer infrastructure, it is possible to decrease the energy consumption of the HPC system by slowing it down. We have not determined yet: 1) if the energy reduction leads to lower costs for the facility manager and for the users; 2) how to perform accounting in order to foster the adoption (by the facility manager and users) of the energy-efficient operating condition.

We will now discuss four different pricing scheme to see how they impact the TCO and the system total gain and the average job price. In addition to the variables introduced in Table 3, we are also going to consider normalized values for the two most interesting variables: 1) normalized system gain γ_N^θ and 2) normalized job price χ_N^θ. The normalized gains and costs are computed w.r.t. to the *Pricing Scheme 1* (see 3.3.1), with a scaling factor equal to 1, a situation that we assume is our baseline. The normalized gain (cost) for any given combination of parameters and pricing scheme is obtained by dividing the resulting gain (cost) by the baseline gain (cost). Since in all the remaining discussion we are going to focus on system gains and average job prices (and related parameters) computed in time frame θ we are going to omit the time frame reference from the mathematical notation, for the sake of clarity (for example $\gamma^\theta \rightarrow \gamma$).

In Table 4 the different ways to compute the system time frame income implied by the different pricing schemes are summarized. The table final three columns serve to quickly summarize the scheme features. *Coeff.* indicates the cost coefficient used to give a price to resource per hour; it can include both the depreciation costs (derived from the system installation cost) and the energy cost ("Depreciation+Energy") or consider only the depreciation cost ("Depreciation"). The *TtS* column specifies the time-to-solution used in the price formula; allowed values are: the real TtS, the oracle TtS (the time-to-solution at maximum frequency) and the scaled time-to-solution (the real TtS divided by the scaling factor).

Finally, the *Energy* columns tells how the energy is taken into account; "explicit" means that the energy costs is directly covered by the users, "implicit" means that the cost is included in the price coefficient (see the numerator of κ_T in Table 2).

Since we are interested in understanding the influence of frequency scaling, we begin by focusing our analysis on the parameters that mostly impact its effect, namely the scaling factor (φ) and job sensitivity (σ).

We then observe the target output as a function of these two variables, keeping all remaining parameters fixed. The scaling factor is the main variable the system manager and the users can use as a knob to regulate the power consumption; in our analysis we consider values ranging from 1 (no scaling) to 5 (aggressive power reduction). As an example in today high end CPUs it is common to see the clock frequency ranging from 3.6 GHz (Turbo mode) to 1.2GHz. The job sensitivity has a big influence on the outcome due to the direct impact on the job time-to-solution when the power is reduced; we let the job sensitivity vary from 0, that is an idealized case where reducing the power consumption does not entail a TtS increase, to 1, when the TtS increase is proportional to the power reduction. Job sensitivity values closer to 0 represent memory or I/O bound jobs while moving closer to the opposite end of the range the application are getting more CPU-bound.

When looking at the normalized system gain values larger than one indicate that the considered price model with the specified scaling factor and job sensitivity (tuple $< price\_model, \varphi, \sigma >$) leads to larger gains w.r.t. to the baseline. Conversely, normalized system gains smaller than 1 and negative values indicate that the baseline produces better results; negative values are possible because for some pricing scheme and parameters combination the system gain can actually be negative – the system is losing money due to the fact that the cost is higher than the income. With the fixed parameters configuration used in the following subsections the baseline does produce positive net gain for the system. The same discussion can be applied to the normalized job price, with the exception that the latter can never be negative – the minimum value for the average cost of a job is zero.

One last point to address before introducing the pricing schemes is the issue of the TtS increase. Users might not accept the fact that the TtS of their application is stretched over a certain point due to the frequency scaling. This is mitigated by the fact that when users submit their job, they typically provide estimated TtS that are longer than the actual TtS; stretching their application but maintaining them under their estimated TtS would generate no complaints. Using historical data from a tier-0 su-

Scheme	*Expression*	*Coeff.*	*TtS*	*Energy*
Scheme 1	$\kappa_T R^a\big((1-\beta)\delta_S + \beta\delta_M\big)$	Depreciation + Energy	Real	Implicit
Scheme 2	$\kappa_T R^a\big(\delta_M\big)$	Depreciation + Energy	Oracle	Implicit
Scheme 3	$\kappa_T R^a\big(\frac{(1-\beta)\delta_S}{\varphi} + \beta\delta_M\big)$	Depreciation + Energy	Scaled	Implicit
Scheme 4	$R^a\kappa_E\big((1-\beta)\delta_S + \beta\delta_M\big) +$ $+\ \big((1-\beta)\pi_S\delta_S + \beta\pi_M\delta_M\big)\cdot\frac{E_\epsilon}{1000}$	Depreciation	Real	Explicit

Table 4: Income functions with different pricing strategies

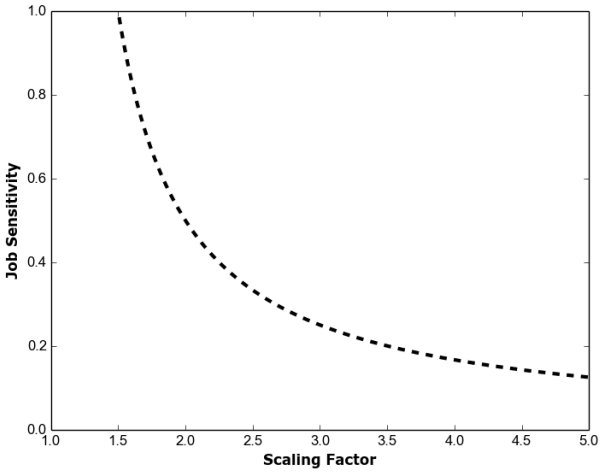


Figure 4: Scaling Factor VS Job Sensitivity for acceptable TtS increases

percomputer we discovered that the average ratio between estimated TtS and real TtS is 1.5 (considering only jobs which run longer than 1 hour to exclude very short application that would skew the mean value). This acceptable TtS increase corresponds to the values of scaling factor φ and job sensitivity σ displayed as dashed black lines in the following two-dimensional figures and as a black line in the three-dimensional ones. Points below the line correspond to acceptable TtS increase. This information can be used while devising pricing scheme in order to account also for the user satisfaction (for instance, not selecting scaling factor values that would exceedingly slow down an application).

This acceptable TtS increase corresponds to the values of scaling factor φ and job sensitivity σ displayed in Figure 4; points below the line correspond to acceptable TtS increase. This information can be used while devising pricing scheme in order to account also for the user satisfaction (for instance, not selecting scaling factor values that would exceedingly lengthen an application).

3.3.1. Scheme 1

This is the pricing model employed in most HPC facilities. Users pay a price based on the amount of requested resources and the real time-to-solution (wall time) of their job multiplied by the coefficient κ_T. The total income for the HPC facility is therefore given as the sum of the prices of all jobs that run during the time frame.

In this case (as in the two following ones discussed in Sections 3.3.2 and 3.3.3) the energy costs are entirely covered by the facility managers, energy savings or increase do not modify the job price for the user which only depends on the TtS. The system owners address this issue by including worst-case estimated energy costs in the cost coefficient κ_T.

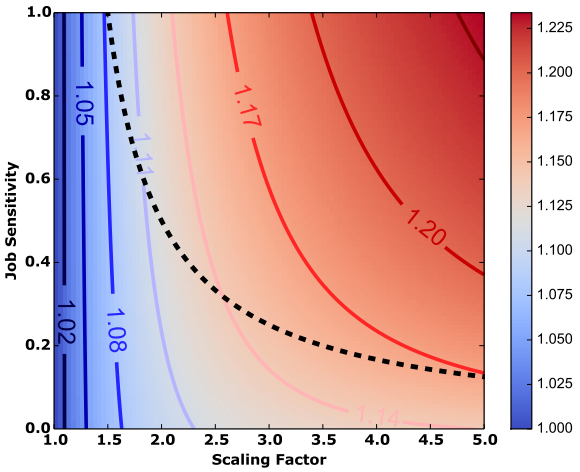
In Figure 5 we observe the normalized system gain for the *Pricing Scheme 1*. Fig. 5a shows in the x-axis the scaling factor φ and the job sensitivity σ in the y-axis; the different colored contours (the lines of points with the same value) indicate the normalized system gain. The same information is presented in three dimensions in Fig. 5b; here the x-axis and y-axis indicate again the scaling factor and job sensitivity while the z-axis shows the normalized system gain. This kind of coupled plots is used also to look at the normalized job price (Figure 6) and for the remaining models (see corresponding figures in Sections 3.3.2, 3.3.3 and 3.3.4).

The dotted black line plotted in the two-dimensional graphs is the same line seen in seen in Fig. 4; combinations of (φ, σ) above that line represent conditions where the frequency scaling would make the job TtS longer beyond the point where the user notice the difference (and loss of quality of service – QoS).

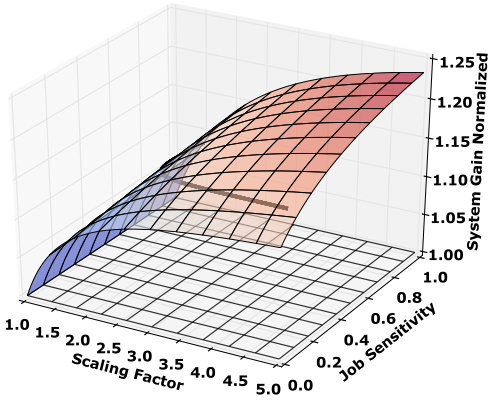
It is quite straightforward to see that with *Pricing Scheme 1* the system owner gains more when the scaling factor increases, especially with higher job sensitivity. This happens because the price paid by the users increases due the longer TtS of the jobs. This is clearly shown by Fig. 6, where the normalized (average) job price rises rapidly together with the scaling factor. If the scaling factor is set to one, the job sensitivity loses its influence and the system gain and job price do not differ from the baseline. This happens with all pricing models. Although this pricing scheme is very enticing from the facility owner point of view, the steep price rises facing the users make its actual implementation almost impossible.

3.3.2. Scheme 2

In this strategy the price paid for each job is given by multiplying number of requested cores by the same coefficient of Sec. 3.3.1 and by the job time-to-solution at maximum or nominal frequency. Clearly, the latter quantity can be only known a posteriori or by means of an

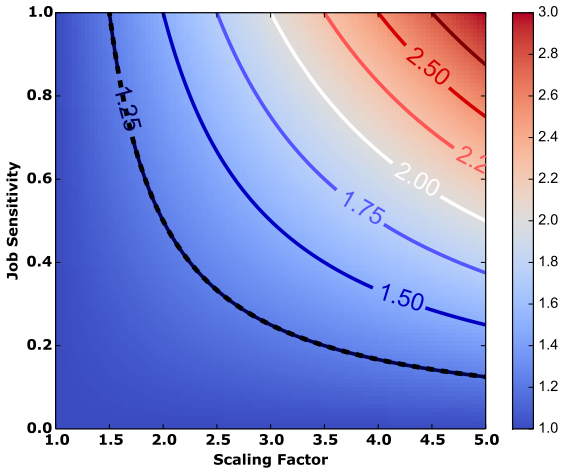


(a) 2d Contour

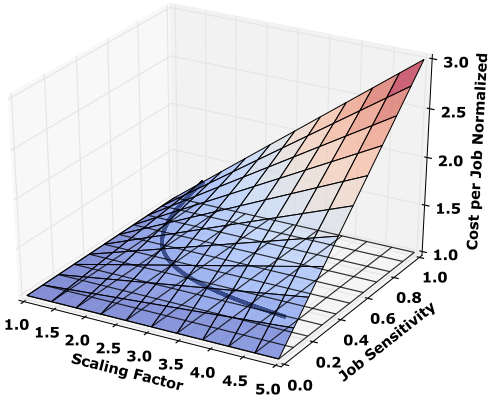


(b) 3d Surface

Figure 5: *Pricing Scheme 1*: System Gain Normalized



(a) 2d Contour



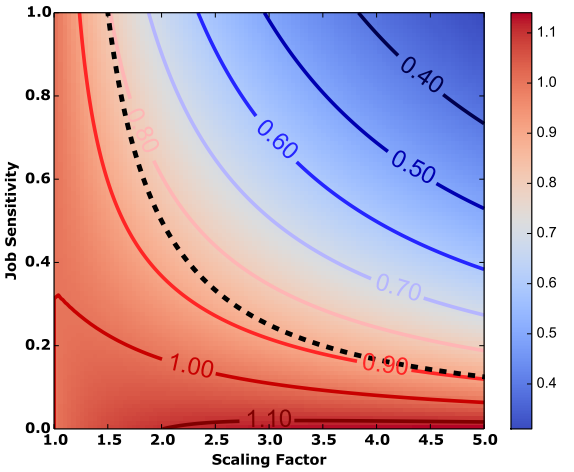
(b) 3d Surface

Figure 6: *Pricing Scheme 1*: Job Price Normalized

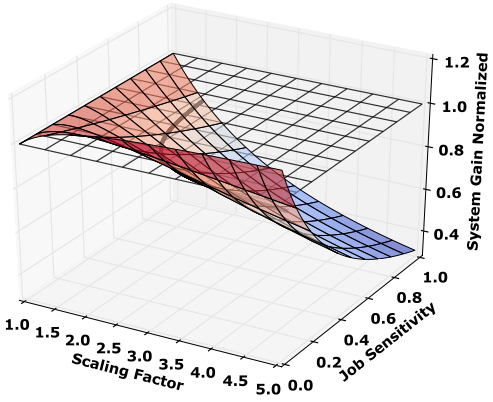
oracle, a priori. Very precise application and architectural models and monitoring tools could be used to obtain an accurate estimate. The results in this section motivate that this technology would enable power management solutions leading to a win-win situation for the system owner and final users. The income is computed as the sum of all jobs prices. In this case the price per job remains constant, i.e. it is not affected by the reduction in power consumption; for this reason we did not include the corresponding figure. When compared with the default pricing (*Pricing Scheme 1*) this scheme benefits the supercomputer users while the gains from the system owner's point of view depend on the application scaling factor and job sensitivity.

In Figure 7 we can observe the normalized system gain for *Pricing Scheme 2*. As previously noted, with this scheme the price paid by users for each job does not change with the scaling factor because it depends only on the application's estimated TtS while running at maximum frequency. The job price is therefore equal to the baseline one, hence the normalized job price is equal to one in every point. Aside from this relatively trivial consideration, it is worth to note that while the job price remain constant, the system gain drastically changes: when the scaling factor and job sensitivity are relatively low *Pricing Scheme 2* leads to a larger gain compared to the baseline. This happens because in this case the real job time-to-solution is not too different from the estimated ones and therefore the income loss is lower than the cost saved on energy consumption thanks to the reduced power consumptions. Conversely, when the scaling factor increases the system gain drops since the energy savings does not balance the loss of income relative to the baseline.

As a final remark, it must be noted from Fig. 7a that the area where the system owners achieve a gain (under the red-line with 1.00 marker) is below the user noticeable level (black dashed line). Meaning that the system owner

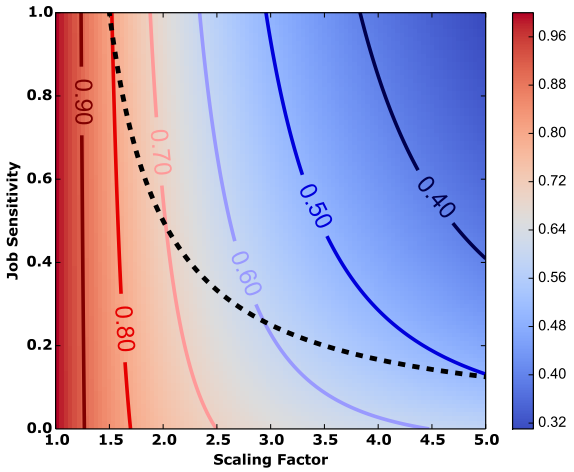


(a) 2d Contour

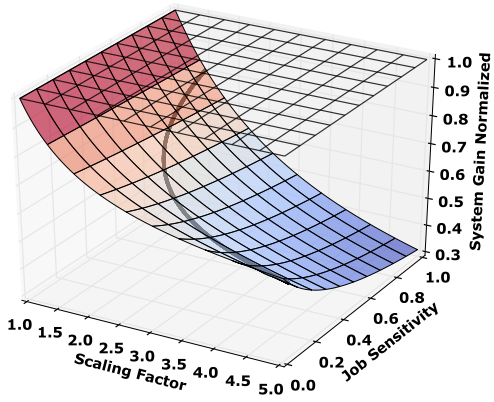


(b) 3d Surface

Figure 7: *Pricing Scheme 2*: System Gain Normalized



(a) 2d Contour



(b) 3d Surface

Figure 8: *Pricing Scheme 3*: System Gain Normalized

can achieve a gain without inducing QoS loss. In this scheme it is essential for the system owner to identify the area delimited by combinations of application sensitivity (σ) and scaling factor (φ) leading to a gain. The system owner assumes the risks for failing it. To summarize, the actual implementation of this price scheme requires the development of tools for identifying job sensitivity and estimating the application time-to-solution at the maximum frequency.

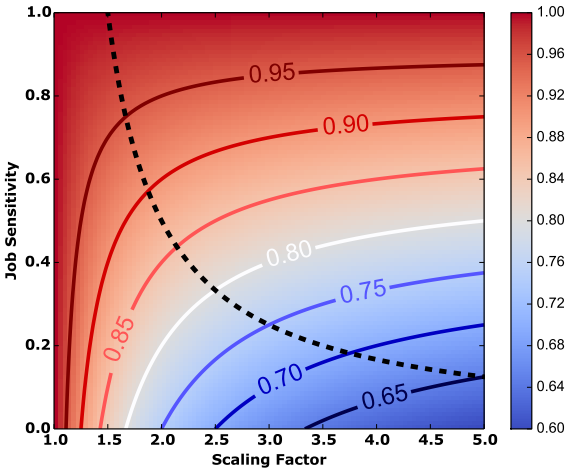
3.3.3. Scheme 3

This pricing model closely resembles the one of Sec. 3.3.2 but tries to solve the problem of estimating the jobs duration at maximum frequency by employing the real job TtS at a scaled frequency with scaling factor φ. This is done taking advantage of the observation that when reducing a processor frequency of a scaling factor φ, the time-to-solution can increase at maximum of a factor φ. For this reason the price of jobs with reduced frequency is discounted by the scaling factor ($\frac{(1-\beta)\delta_S}{\varphi}$).

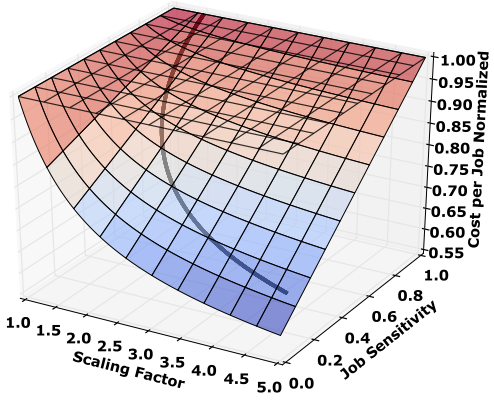
From Figure 9 we can notice that this approach is highly favourable from the users point of view, since it leads to markedly diminishing cost when the scaling factor and the job sensitivity increase. The smaller average job price is due to the division by the scaling factor applied to the price of the slowed down jobs. However, for the considered system configuration, this causes a lower system gain w.r.t. the baseline (*Pricing Scheme 1*) since the energy-related savings are much smaller than the decrease of revenues (see Fig. 8).

3.3.4. Scheme 4

With this last pricing schemes, in opposition to the previous ones, the energy cost is not paid by the system owner but it is directly included in the job price. Also in this case the income is given as the sum of all job prices and now

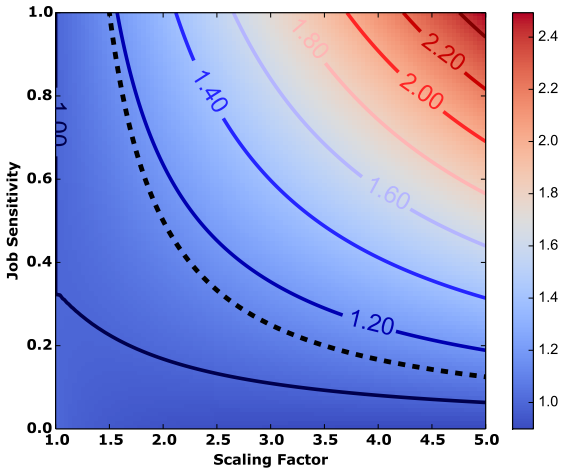


(a) 2d Contour

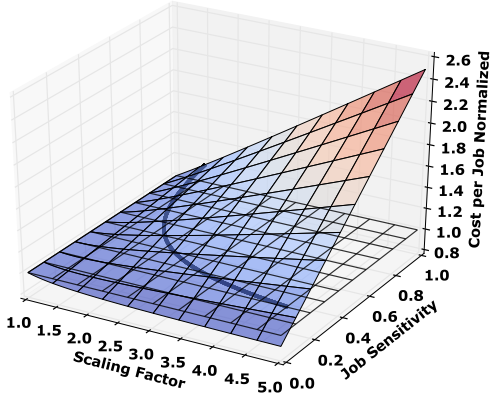


(b) 3d Surface

Figure 9: *Pricing Scheme 3*: Job Price Normalized



(a) 2d Contour



(b) 3d Surface

Figure 10: *Pricing Scheme 4*: Job Price Normalized

each price is composed by two components. The first one depends on the number of requested cores times the TtS (scaled and not scaled) multiplied by the cost coefficient κ_E; this coefficient is computed excluding the estimated energy costs – users would not agree to cover the energy costs *twice*. The second component is the cost of the energy of the job, given as the TtS multiplied by the power consumption times the price of the energy (E_ϵ).

The system gain with *Pricing Scheme 4* is constant and therefore also the normalized system gain does not change and it is always equal to the baseline (hence the corresponding figures are not displayed). The possible benefits deriving from the adoption of this pricing scheme stems from the reduction of average job price, as revealed by Figure 10. With lower values of scaling factor and job sensitivity the normalized job price is smaller than the baseline; when these parameters start rising, the job price follow them accordingly and therefore it surpasses the baseline. Differently from *Pricing Scheme 2* , this approach shifts the gains and the risks to the final user.

It does not require estimating the jobs TtS at maximum frequency but only needs a per job energy accounting system. Clearly, users would need tools for selecting and applying the right power reduction to their applications.

3.3.5. Pricing Schemes Comparison

In Table 5 we can see an example of the results of the pricing schemes. Starting from the previous configuration – based on *Fermi* – we modified a subset of the input parameters (idle percentage ι, scaling factor ϕ and job sensitivity σ); we also varied the amount of cost (per θ) due to system depreciation – expressed as percentage of the total cost. As output we present the difference w.r.t. the baseline, showing both system owner gain and price paid by users, for each pricing scheme. The values in bold highlight the pricing schemes that, under the given condition,

manage to bring benefits for both owners and users. From the point of view of the system owner positive values are preferable (increased gain), while users prefer negative values (price decrease).

Considering a set point resembling a memory bound application (TtS increase of 20% as effect of a 2x in frequency reduction) we notice that: 1) *Pricing Scheme 1* increases the system gain but penalizes the final user; 2) *Pricing Scheme 3* is beneficial for the user (who gets a discount of 20%) but generates significant revenue loss for the system owner; 3) *Pricing Scheme 2* and *Pricing Scheme 4* instead lead to noticeable saving without harming the counterpart – favouring, respectively, the facility manager and the final user. Lowering the idle power improves the savings of 2/3 while reducing the depreciation cost of 1/3 doubles the revenues and price reductions achievable by power management strategies. This can reach the 10% of the total revenues in case of low idle power and long machine turnaround.

The challenge in implementing the *Pricing Scheme 2* is the need to predict what would have been the real application TtS if no power management strategy had been applied; *Pricing Scheme 4* only requires the support for accurate per job energy accounting.

4. Future HPC Scenarios

So far, we focused on an existing HPC system with its particular parameters. In this section we are going to explore different scenarios that can be envisioned as near-future evolutions of current supercomputers. As we have seen in Section 3 two of the main factor impacting the costs faced by system owners are idle power aspects hindering the convenience of frequency scaling, namely the non-null percentage of power consumed by computing units in idle state (the idle power consumption remains constant even if the operating frequency is reduced) and the depreciation costs. The depreciation costs is not influenced by the frequency scaling: if the energy savings are not big enough to compensate the lost income the system owner will face an overall loss. In the system considered as a case study for this work the depreciation costs have a notable impact and they correspond to the 67% of the total per-time frame expenses. We consider two cases: 1) energy proportional systems (where the idle power consumption is very low) and 2) low depreciation costs.

Since the behaviours of the pricing schemes *Pricing Scheme 1*, *Pricing Scheme 2* and *Pricing Scheme 4* in the new scenarios are not substantially different than those observed in Sec. 3.3 we concentrate on *Pricing Scheme 3*. Now we want explore the design space to understand if under different conditions this scheme can generate profit also for the system owners; as we have seen before this is the best scheme from the user point of view because it lowers the price paid per job. In the following sections we are going to evaluate the economic viability of *Pricing Scheme 3* in the case of alternative HPC systems, with low idle power consumption (4.1) and low depreciation costs (4.2).

4.1. Energy Proportional Systems

Several research works have pointed in the direction of energy proportional systems as a possible solution towards improvements in terms of energy efficiency [44, 17, 45]. In an energy proportional system the power consumed by its computing nodes scales down proportionally with the load. In our model, this kind of system can be simulated by setting a very low percentage of idle power consumption ι. We analyze the profitability for the system owner using *Pricing Scheme 3*; the scheme generates profit if the income for time frame is larger than the expenses (energy costs plus depreciation). We are going to consider the isosurface corresponding to the points where the function C_T^θ / I^θ (total costs divided by income) is equal to 1. Points *below* the surface represents parameters combinations that are profitable for the system.

Figure 11 considers the system profitability with varying depreciation costs, while maintaining a fixed (very low) value for the idle power percentage ($\iota = 0.01$). In the x and y axis we have the alpha factor α and the scaling factor φ; the z-axis presents instead the system life time LF. This parameter is a very good proxy for the depreciation costs impact, since a shorter life time means that the installation costs must be recovered more quickly, hence higher depreciation costs. In the figure, the life time varies in a range of $[1, 50]$ years, with a corresponding percentage of depreciation costs (w.r.t. the total time frame costs) of $[88\%, 13\%]$. We observe that, with a negligible idle power, the depreciation costs strongly impacts the system gain: with lower life time values is much harder for the system to be profitable. This happens because if the depreciation costs are the biggest expense source the energy saved through frequency scaling gets negligible while the income loss – due to dividing the price paid by users by the scaling factor – becomes preponderant.

4.2. Low Depreciation Costs

The second parameter strongly influencing the feasibility of a pricing scheme is the depreciation cost, or more precisely the fraction of the total time frame costs that serve to cover the initial investment expenses. The depreciation costs are regulated by the system installation cost C_S^T and by the expected life time LF, that is generally a few years. The continuous quest towards maximum computing performance tends to increase the system installation costs and to squeeze the machines lifetime, but as more nuanced approaches more focused on energy efficiency are gradually taking hold, it is possible to envision slightly different systems where the installation costs decrease and the life time increases. This shift would lead to systems where the depreciation costs impact is less predominant w.r.t. to the energy expenses sustained to operate the machine.

Depreciation	ι	φ	σ	*Scheme 1*		*Scheme 2*		*Scheme 3*		*Scheme 4*	
				Gain	*Price Dif.*	*Gain*	*Price Dif.*	*Gain*	*Price Dif.*	*Gain*	*Price Dif.*
67%	20%	2.0	0.2	16%	10%	**4%**	**0%**	-21%	-20%	**0%**	**-4%**
	10%	2.0	0.2	19%	10%	**6%**	**0%**	-19%	-20%	**0%**	**-5%**
47%	20%	2.0	0.2	27%	10%	**9%**	**0%**	-26%	-20%	**0%**	**-6%**
	10%	2.0	0.2	30%	10%	**12%**	**0%**	-23%	-20%	**0%**	**-8%**

Table 5: Example: Pricing Schemes Results; the *Gain* and *Price Dif.* columns represent, respectively, the system gains and the price difference, expressed as percentage

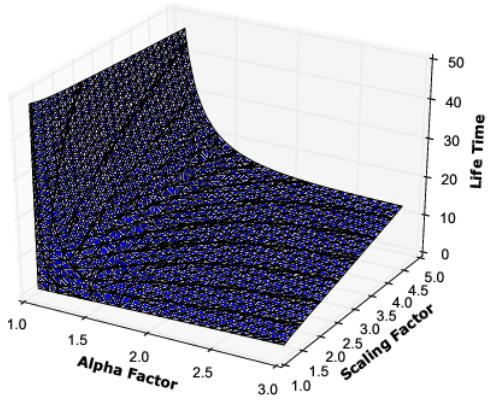


Figure 11: System Profitability with low idle power %

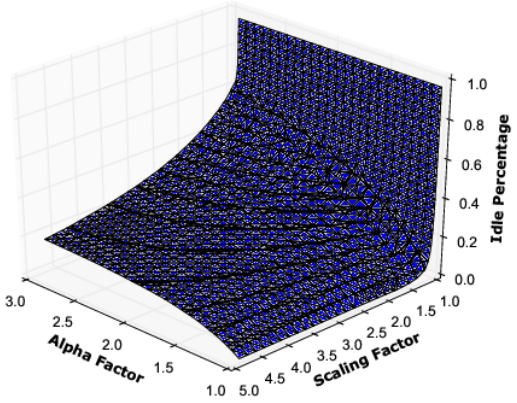


Figure 12: System Profitability with low depreciation costs

In Figure 12 we see the system profitability surface in case of depreciation costs close to zero ($\leq 0.01\%$ of the time frame costs). The three axes x, y and z represent, respectively, the alpha factor α, the scaling factor φ and the idle power percentage ι. The points below the surface (i.e. $< 2.5, 3.0, 0.2 >$) form the region where the system gain is positive (the costs are smaller than the income); as we proceed further from the surface the gain gets higher. With no frequency scaling ($\varphi = 1$) the system is always gaining, due to the remaining model parameters being configured to assure a net profit at maximum frequency (as a baseline). As it was expected, low idle power percentage leads to bigger benefits for the system owner since it allows to consume less power if the frequency is scaled down. We can also notice that higher α values are better for the system owners; this happens because a larger alpha factors means that scaling down the frequency leads to greater energy savings. Finally, we notice an asymptotic behaviour w.r.t. scaling factor: the benefits of decreasing the frequency tend to get thinner and thinner.

5. Conclusion

In this paper we tackled the issue of understanding the impact of energy-aware mechanisms in HPC machines. More precisely, we considered frequency scaling as a technique to exchange the power performance of computing nodes in exchange for lower power consumption. Frequency scaling has a clear impact on the energy expenses sustained by a supercomputing facilities and at the same time it strongly influence the accounting mechanism (the price paid by users for using system resources). Our goal was then to provide an instrument capable to analyse the costs and benefits obtained through frequency scaling in a HPC system.

We then devised a parametric model inspired by a real supercomputer to simulate the impact of frequency scaling on the system revenue and energy-related costs. We proposed four different pricing schemes and evaluated their effectiveness including the perspectives of both the facility owner and the system users. Our preliminary results indicate that is indeed possible to save energy and curb operational costs via frequency scaling and, at the same time, not to penalize users from a economic point of view.

As a final takeaway the most valuable strategy to push towards green computing is to shift the cost of the energy consumption to the final user while at the same time providing her instruments for accounting her job energy consumption and scaling the performance level. Letting the system owner play this knob still requires research progress in order to estimate the TtS of applications not perturbed by frequency scaling. In future energy proportional sys-

tems, with a longer turn-around, simpler estimation methods will start to pay off as well.

Acknowledgements

This work was partially supported by the FP7 ERC Advance project MULTITHERMAN (g.a. 291125). We also want to thank CINECA for granting us the access to their systems.

[1] W.-c. Feng, K. Cameron, The Green500 List: Encouraging Sustainable Supercomputing, IEEE Computer 40 (12).

[2] K. Bergman, S. Borkar, D. Campbell, et al., ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems (September 2008).

[3] A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo, W. Lintner, United States Data Center Energy Usage Report (Jun. 2016).

[4] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. De Supinski, M. Schulz, Bounding energy consumption in large-scale MPI programs, in: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, ACM, 2007, p. 49.

[5] D. Zivanovic, M. Pavlovic, M. Radulovic, H. Shin, J. Son, S. A. Mckee, P. M. Carpenter, P. Radojković, E. Ayguadé, Main Memory in HPC: Do We Need More or Could We Live with Less?, ACM Trans. Archit. Code Optim. 14 (1) (2017) 3:1–3:26. doi:10.1145/3023362.
URL http://doi.acm.org/10.1145/3023362

[6] V. Marjanović, J. Gracia, C. W. Glass, Performance modeling of the HPCG benchmark, in: International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, Springer, 2014, pp. 172–192.

[7] M. Radulovic, D. Zivanovic, D. Ruiz, B. R. de Supinski, S. A. McKee, P. Radojković, E. Ayguadé, Another Trip to the Wall: How Much Will Stacked DRAM Benefit HPC?, in: Proceedings of the 2015 International Symposium on Memory Systems, MEMSYS '15, ACM, New York, NY, USA, 2015, pp. 31–36. doi:10.1145/2818950.2818955.
URL http://doi.acm.org/10.1145/2818950.2818955

[8] A. Auweter, A. Bode, M. Brehm, L. Brochard, N. Hammer, H. Huber, R. Panda, F. Thomas, T. Wilde, A Case Study of Energy Aware Scheduling on SuperMUC, Springer International Publishing, Cham, 2014, pp. 394–409. doi:10.1007/978-3-319-07518-1_25.
URL http://dx.doi.org/10.1007/978-3-319-07518-1_25

[9] J. J. Dongarra, P. Luszczek, A. Petitet, The LINPACK benchmark: past, present and future, Concurrency and Computation: practice and experience 15 (9) (2003) 803–820.

[10] J. Dongarra, M. A. Heroux, Toward a new metric for ranking high performance computing systems, Sandia Report, SAND2013-4744 312.

[11] H. David, E. Gorbatov, U. R. Hanebutte, et Al., RAPL: Memory Power Estimation and Capping, in: Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED '10, ACM, New York, NY, USA, 2010. doi:10.1145/1840845.1840883.

[12] Y. Inadomi, T. Patki, K. Inoue, et Al, Analyzing and Mitigating the Impact of Manufacturing Variability in Power-constrained Supercomputing, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, ACM, New York, NY, USA, 2015, pp. 78:1–78:12. doi:10.1145/2807591.2807638.

[13] A. Langer, H. Dokania, L. Kale, et Al., Analyzing Energy-Time Tradeoff in Power Overprovisioned HPC Data Centers, in: Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International, 2015, pp. 849–854. doi:10.1109/IPDPSW.2015.129.

[14] Cineca accounting policy, https://wiki.u-gov.it/confluence/pages/viewpage.action?pageId=64201371, accessed: 2017-03-30 (2017).

[15] Cineca inter-university consortium, http://www.cineca.it//en.

[16] M. Etinski, J. Corbalan, J. Labarta, M. Valero, Understanding the future of energy-performance trade-off via DVFS in HPC environments, Journal of Parallel and Distributed Computing 72 (4) (2012) 579 – 590. doi:http://dx.doi.org/10.1016/j.jpdc.2012.01.006.

[17] G. Varsamopoulos, S. K. Gupta, Energy proportionality and the future: Metrics and directions, in: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, IEEE, 2010.

[18] T. Patki, D. K. Lowenthal, B. Rountree, et Al., Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing, in: Proceedings of the 27th International ACM Conference on International Conference on Supercomputing, ICS '13, ACM, New York, NY, USA, 2013, pp. 173–182. doi:10.1145/2464996.2465009.

[19] J. Hikita, A. Hirano, H. Nakashima, Saving 200kW and $200 K/year by power-aware job/machine scheduling, in: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, 2008, pp. 1–8. doi:10.1109/IPDPS.2008.4536218.

[20] H. Shoukourian, T. Wilde, A. Auweter, A. Bode, Power variation aware Configuration Adviser for scalable HPC schedulers, in: High Performance Computing Simulation (HPCS), 2015 International Conference on, 2015, pp. 71–79. doi:10.1109/HPCSim.2015.7237023.

[21] V. W. Freeh, D. K. Lowenthal, Using Multiple Energy Gears in MPI Programs on a Power-scalable Cluster, in: Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '05, ACM, New York, NY, USA, 2005, pp. 164–173. doi:10.1145/1065944.1065967.
URL http://doi.acm.org/10.1145/1065944.1065967

[22] M. Y. Lim, V. W. Freeh, D. K. Lowenthal, Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs, in: SC 2006 conference, proceedings of the ACM/IEEE, IEEE, 2006, pp. 14–14.

[23] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, T. Bletsch, Adagio: Making DVS Practical for Complex HPC Applications, in: Proceedings of the 23rd International Conference on Supercomputing, ICS '09, ACM, New York, NY, USA, 2009, pp. 460–469. doi:10.1145/1542275.1542340.
URL http://doi.acm.org/10.1145/1542275.1542340

[24] P. E. Bailey, D. K. Lowenthal, V. Ravi, et Al., Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems, in: Proceedings of the 2014 Brazilian Conference on Intelligent Systems, BRACIS '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 371–380. doi:10.1109/ICPP.2014.46.

[25] T. Patki, D. K. Lowenthal, A. Sasidharan, et Al., Practical Resource Management in Power-Constrained, High Performance Computing, in: Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing, HPDC '15, ACM, New York, NY, USA, 2015, pp. 121–132. doi:10.1145/2749246.2749262.

[26] N. Kappiah, V. W. Freeh, D. Lowenthal, Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs, in: Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, 2005, pp. 33–33. doi:10.1109/SC.2005.39.

[27] V. W. Freeh, D. K. Lowenthal, F. Pan, et Al., Analyzing the Energy-Time Trade-Off in High-Performance Computing Applications, IEEE Trans. Parallel Distrib. Syst. 18 (6). doi:10.1109/TPDS.2007.1026.

[28] C. Hsu, W. Feng, A power-aware run-time system for high-performance computing, in: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, IEEE Computer Society, 2005.

[29] M. Etinski, J. Corbalan, J. Labarta, M. Valero, Optimizing job performance under a given power constraint in HPC centers, in: Green Computing Conference, 2010 International, 2010. doi:10.1109/GREENCOMP.2010.5598303.

[30] M. Etinski, J. Corbalan, J. Labarta, M. Valero, Parallel job scheduling for power constrained HPC systems, Parallel Computing 38 (12). doi:http://dx.doi.org/10.1016/j.parco.2012.08.001.

[31] D. A. Ellsworth, A. D. Malony, B. Rountree, M. Schulz, Dynamic Power Sharing for Higher Job Throughput, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, ACM, New York, NY, USA, 2015, pp. 80:1–80:11. doi:10.1145/2807591.2807643.

[32] P. Samadi, A.-H. Mohsenian-Rad, R. Schober, V. W. Wong, J. Jatskevich, Optimal real-time pricing algorithm based on utility maximization for smart grid, in: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, IEEE, 2010, pp. 415–420.

[33] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, R. Buyya, Pricing cloud compute commodities: A novel financial economic model, in: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on, IEEE, 2012, pp. 451–457.

[34] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, F. C. Lau, Dynamic pricing and profit maximization for the cloud with geo-distributed data centers, in: INFOCOM, 2014 Proceedings IEEE, IEEE, 2014, pp. 118–126.

[35] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, R. P. Doyle, Managing energy and server resources in hosting centers, ACM SIGOPS operating systems review 35 (5) (2001) 103–116.

[36] Y. Zhang, Y. Wang, X. Wang, Electricity bill capping for cloud-scale data centers that impact the power markets, in: Parallel Processing (ICPP), 2012 41st International Conference on, IEEE, 2012, pp. 440–449.

[37] C. Wang, B. Urgaonkar, Q. Wang, G. Kesidis, A hierarchical demand response framework for data center power cost optimization under real-world electricity pricing, in: Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on, IEEE, 2014, pp. 305–314.

[38] Fermi supercomputer, https://www.cineca.it/it/news/fermi-il-nuovo-supercomputer-del-cineca, accessed: 2017-06-19 (2017).

[39] D. Feitelson, Job scheduling in multiprogrammed parallel systems (extended version), IBM Research Report RC19790 (87657) 2nd Revision 16 (1997) 104–113. doi:10.1145/1007771.55608.

[40] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, Job Scheduling Strategies for Parallel Processing: 10th International Workshop, JSSPP 2004, New York, NY, USA, June 13, 2004. Revised Selected Papers, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, Ch. Parallel Job Scheduling — A Status Report, pp. 1–16. doi:10.1007/11407522_1.
URL http://dx.doi.org/10.1007/11407522_1

[41] J. Cao, A. Chan, Y. Sun, S. Das, M. Guo, A taxonomy of application scheduling tools for high performance cluster computing, Cluster Computing 9 (3) (2006) 355–371. doi:10.1007/s10586-006-9747-2.
URL http://dx.doi.org/10.1007/s10586-006-9747-2

[42] H. You, H. Zhang, Comprehensive Workload Analysis and Modeling of a Petascale Supercomputer, in: W. Cirne, N. Desai, E. Frachtenberg, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing, Vol. 7698 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 253–271. doi:10.1007/978-3-642-35867-8_14.
URL http://dx.doi.org/10.1007/978-3-642-35867-8_14

[43] T. Sakurai, A. R. Newton, Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas, IEEE Journal of solid-state circuits 25 (2) (1990) 584–594.

[44] L. A. Barroso, U. Holzle, The Case for Energy-Proportional Computing, IEEE Computer 40.
URL http://www.computer.org/portal/site/computer/index.jsp?pageID=computer_level1&path=computer/homepage/Dec07&file=feature.xml&xsl=article.xsl

[45] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, C. Kozyrakis, Towards Energy Proportionality for Large-scale Latency-critical Workloads, SIGARCH Comput. Archit. News 42 (3) (2014) 301–312. doi:10.1145/2678373.2665718.
URL http://doi.acm.org/10.1145/2678373.2665718