Optimal CPU Frequency Selection to Minimize the Runtime of Tasks Under Power Capping

Fanny Dufossé[1][0000−0002−2260−2200] and Rizos Sakellariou[2][0000−0002−6104−6649]

[1] Univ. Grenoble Alpes, Inria, CNRS, Grenoble, France
[2] The University of Manchester, Manchester, UK

Abstract. Data centers are increasingly becoming significant energy consumers worldwide. To reduce the amount of electricity they consume, power capping may be used to set a limit to the maximum power they can use at some given point in time. In this situation, an interesting problem is how to make best use of the available power by throttling the CPU frequency of different servers. As different tasks assigned to each of these servers may not be impacted the same way when changing a server's CPU frequency, one problem that arises is how to select CPU frequencies for each of the servers running tasks with specific characteristics in such a way that the total execution time of all these tasks is minimized while the overall power cap for all the servers is respected. The paper presents an approach that models this problem as an optimization problem and shows how to find an optimal solution in different cases. This work can provide the basis to find economical solutions to operate large data centers under power capping efficiently.

Keywords: Data Centers · Power Capping · DVFS · CPU frequency selection

1 Introduction

Energy consumption has become a serious concern for computing in recent years. Mechanisms such as *power capping* or *Dynamic Voltage Frequency Scaling* (DVFS) set limits to control power consumption: power capping sets an upper bound for the maximum power that can be consumed at any point in time, while DVFS scales down voltage and CPU frequency, hence power. Both mechanisms have the potential to lead to energy savings, however, there are various aspects and trade-offs that have to be considered for this to happen, also to avoid any adverse effects on performance and system Quality of Service [10,12,20]. In general, there appears to be a consensus that such techniques need to be carefully managed if they are to lead to energy savings.

In this paper, we present work that considers a set of DVFS-enabled servers which operate in an environment where a global power cap needs to be met. This could be, for instance, the servers of a cluster, a cloud provider or a data center, where the total power consumption of the cluster, provider or data center, respectively, should not exceed a certain limit. Each of the servers has been allocated some tasks, with specific CPU and I/O requirements; these tasks cannot

be reallocated and must be executed on the servers they have been allocated to. The problem that we address is how to choose frequencies for each of the servers in a way that the overall execution time of the tasks is minimized, yet the global power cap for all the servers is met. The key property to take into account is the task requirements: the execution time of tasks with high CPU requirements is affected most by a CPU frequency reduction whereas the execution time of tasks with high I/O requirements is affected less.

The rest of the paper describes the fundamentals of an approach to solve the problem. Section 2 provides some background and related work. Section 3 formulates the problem. Section 4 presents a solution along with an example of how this solution can be applied. Finally, Section 5 concludes the paper.

2 Background and Related Work

There has been lots of work in the literature that considers DVFS-enabled resources where the objective is to select appropriate CPU frequencies often to minimize energy consumption [1,2,11,14,16,19]. Other work has also considered how to optimize performance in the presence of a power cap [6,18,5]. In general, finding an appropriate configuration of frequencies to meet a power cap without overly damaging performance is not a trivial problem. Besides the optimization aspects, there are various trade-offs between power (hence CPU frequency too), energy and performance, which may also be affected by the characteristics of the specific applications that are running; see, for example, Figures 1-3 in [15].

Yet, as also noted in [13,14], some cloud providers price compute resources in terms of CPU frequency too, in such a way that a low CPU frequency costs less than a high CPU frequency. This means that cloud users would need to select CPU frequencies that optimize their use of cloud resources: clearly, going for the cheapest CPU frequency, which is the lowest, may not necessarily be a good option as, in this case, applications will take longer to complete. Furthermore, following the observations in [15], it is not the case that the same CPU frequency would be ideal for different sorts of tasks (applications). Generally, CPU-bound tasks (or tasks that do lots of CPU processing) would be affected more than I/O-bound tasks (or tasks that spend lots of time doing input-output and less time on the CPU) if they run at a lower CPU frequency. In other words, the performance drop of CPU-bound tasks would be more noticeable than the performance drop of I/O-bound tasks when running at a lower CPU frequency. Thus, as users typically have a maximum budget for the cloud resources they use, the problem is how to use their budget to select CPU frequencies appropriately in a way that optimizes the performance of a set of tasks with different characteristics that they need to execute on a cloud platform. It is this problem that motivated the work in this paper.

Assuming that CPU frequencies are priced linearly, the answer to this problem from the user's point of view is equivalent to finding an answer to the following practical question: *Given a cloud provider or a data center that should not exceed a certain power consumption limit (power cap), how this provider/center*

can lower frequencies of individual servers, each of which has been allocated a specific task with different CPU and I/O characteristics for execution, so that the provider/center's power cap is not exceeded while the total execution time of all these tasks is minimized?

The model that we develop in the paper to answer this question relies on two key sub-problems that have a significant history in the literature. The first is how to model power consumption in relation to CPU frequency. Generally, it is assumed that power consumption is proportional to frequency cubed [17], a relation that is generalized [2,3,4] to:

$$P_f = P_0 \cdot f^{\alpha}, \tag{1}$$

where P_0 and $\alpha > 1$ are hardware-dependent characteristics and P_f is the power consumption at frequency f. In this paper, we adopt Eq. (1) to model power.

The second sub-problem is how the reduction in frequency affects the execution time of a task. Adopting an approach initially proposed in [9] and also used in [7,8], we estimate the runtime $RT(i, f)$ of a task i at frequency f, as follows:

$$RT(i, f) = \left(\beta_i \cdot (\frac{f_{max}}{f} - 1) + 1\right) \cdot RT(i, f_{max}), \tag{2}$$

where $RT_{f_{max}}$ is the task runtime when running at the maximum CPU frequency f_{max} and β_i is a task-specific parameter that captures a task's CPU-boundedness, takes values between 0 and 1 and can be estimated through profiling [7,8]. Tasks with lots of CPU requirements have a value close to 1 whereas tasks with lots of I/O have a value close to 0.

3 Problem Formulation

We consider a problem with n tasks allocated on m identical machines. An allocation function *Alloc* indicates for each task i, the machine $Alloc_i$ on which it is allocated. In the following, we will use the notation $S_j = \{i, Alloc_i = j\}$, the set of tasks allocated on machine j. Each machine can operate at a frequency f between bounds $f_{\min}$ and $f_{\max}$. The frequencies are fixed once and for all on each machine, before beginning the execution of the tasks. The objective is thus to determine these frequencies in a way that minimizes runtime without exceeding a total power cap. We consider that each task consists of a part of I/O exchanges, and a part of pure CPU computation. The periods of I/O exchanges are not affected by CPU frequency. Both power consumption and runtime are correlated with CPU frequency, as shown in Eqs. (1) and (2). Thus, the runtime of tasks allocated on machine j is

$$RT_j(f) = \sum_{i \in S_j} RT(i, f). \tag{3}$$

For the sake of simplicity, we denote $C_j = \sum_{i \in S_j} \beta_i f_{\max} RT(i, f_{\max})$ the amount of computation of the tasks of S_j and $CT_j = \sum_{i \in S_j} (1 - \beta_i) RT(i, f_{\max})$ its I/O

duration. Eq. (3) can thus be written:

$$RT_j(f) = \frac{C_j}{f} + CT_j. \tag{4}$$

We consider a power capping constraint P for the consumption of the m machines. We make the hypothesis that $P \geq m \times P_0 f_{\min}^{\alpha}$, to guarantee the existence of a valid solution. The optimization criterion is the sum of the runtimes of all tasks. Then, the objective is to minimize the function:

$$RT = \min_{f_1,\ldots,f_m} \sum_{i=1}^{m} RT_j(f_j) \tag{5}$$

under the constraints:

$$\forall j, P_0 \sum_{j=1}^{m} f_j^{\alpha} \leq P, \tag{6}$$

and

$$\forall j, f_{\min} \leq f_j \leq f_{\max}. \tag{7}$$

Without loss of generality, we will consider in the following $P_0 = 1$.

3.1 Example

We consider as an example the execution of 5 tasks with the following parameters (C_j, CT_j): $[(1,5),(6,6),(7,3),(8,2),(40,20)]$. We consider the frequency limits $f_{min} = 2$ and $f_{max} = 5$ and $\alpha = 3$. This means that the minimum power consumption is 40 with resulting runtimes $[5.5, 9, 6.5, 6, 40]$; the sum of runtimes equals 67. The maximal power consumption is 625, with runtimes $[5.2, 7.2, 4.4, 3.6, 28]$ and sum of runtimes 48.4.

We consider a power cap of 200. A first possibility is to run the longest tasks at maximum frequency and the shortest tasks at minimum frequency. We may for example run tasks 1,2 and 3 at frequency 2, and task 5 at frequency 5; the remaining power 51 is for task 4 that can run at frequency $\sqrt[3]{51} \sim 3.7$. With this allocation, we obtain a runtime ~ 53.2. A second possibility is to use the same frequency for all tasks. We then run all tasks at frequency $\sqrt[3]{40} \sim 3.4$. Then, we obtain a runtime of around 54.1.

The optimal frequency values proven in Section 4 obtain a sum of runtimes around 53.1.

4 An Optimal Algorithm to Select Frequencies

4.1 Solution without frequency bounds

We first consider the problem where frequencies have no bounds, that is, there is no constraint as defined in Eq. (7).

Lemma 1. *Optimal frequencies with no bounds If no constraints are given for frequencies values, then, the minimum total runtime is reached for*

$$f_i = \left(\frac{P \cdot C_i^{\frac{\alpha}{1+\alpha}}}{\sum_{j=1}^{m} C_j^{\frac{\alpha}{1+\alpha}}} \right)^{\frac{1}{\alpha}}.$$

The corresponding total runtime is then

$$RT = P^{-\frac{1}{\alpha}} \left(\sum_{j=1}^{m} C_j^{\frac{\alpha}{1+\alpha}} \right)^{\frac{\alpha+1}{\alpha}} + \sum_{j=1}^{m} CT_j.$$

Proof. We demonstrate this lemma by induction on m. This property is trivially true for $m = 1$.

Suppose the result holds for $m-1$, let us prove it for m. We denote P_1 the power consumed by the $m-1$ first machines. By induction, we know that for i between 1 and $m-1$, $f_i = \left(\frac{P_1 \cdot C_i^{\frac{\alpha}{1+\alpha}}}{\sum_{j=1}^{m-1} c_j^{\frac{\alpha}{1+\alpha}}} \right)^{\frac{1}{\alpha}}$. In addition, it remains $P - P_1$ power available for machine m, thus $f_m = (P - P_1)^{\frac{1}{\alpha}}$.

We obtain:

$$\begin{aligned}
RT &= \min_{P_1} \left(\sum_{i=1}^{m-1} \frac{C_i}{\frac{P_1^{\frac{1}{\alpha}} \cdot c_i^{\frac{1}{1+\alpha}}}{\left(\sum_{j=1}^{m-1} c_j^{\frac{\alpha}{1+\alpha}} \right)^{\frac{1}{\alpha}}}} \right) + \frac{C_m}{(P-P_1)^{\frac{1}{\alpha}}} + \sum_{j=1}^{m} CT_j \\
&= \min_{P_1} P_1^{-\frac{1}{\alpha}} \left(\sum_{j=1}^{m-1} c_j^{\frac{\alpha}{1+\alpha}} \right)^{\frac{1}{\alpha}} \left(\sum_{i=1}^{m-1} \frac{C_i}{C_i^{\frac{1}{1+\alpha}}} \right) + \frac{C_m}{(P-P_1)^{\frac{1}{\alpha}}} + \sum_{j=1}^{m} CT_j \\
&= \min_{P_1} P_1^{-\frac{1}{\alpha}} \left(\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}} \right)^{\frac{\alpha+1}{\alpha}} + \frac{C_m}{(P-P_1)^{\frac{1}{\alpha}}} + \sum_{j=1}^{m} CT_j
\end{aligned}$$

We define the function $f(x) = x^{-\frac{1}{\alpha}} \left(\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}} \right)^{\frac{\alpha+1}{\alpha}} + \frac{C_m}{(P-x)^{\frac{1}{\alpha}}}$.

Then, $f'(x) = -\frac{1}{\alpha x^{\frac{\alpha+1}{\alpha}}} \left(\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}} \right)^{\frac{\alpha+1}{\alpha}} + \frac{C_m}{\alpha (P-x)^{\frac{\alpha+1}{\alpha}}}$. $f'(x)$ is an increasing function between 0 and P , that tends to $-\infty$ in 0 and $+\infty$ in P, so $f(x)$ is first decreasing then increasing, and its minimum is obtained in x_{opt} with

$f'(x_{opt}) = 0$. Thus,

$$\begin{aligned}
\frac{1}{\alpha x_{opt}^{\frac{\alpha+1}{\alpha}}} \left(\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}}\right)^{\frac{\alpha+1}{\alpha}} &= \frac{C_m}{\alpha (P-x_{opt})^{\frac{\alpha+1}{\alpha}}} \\
\left(\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}}\right)^{\frac{\alpha+1}{\alpha}} (P-x_{opt})^{\frac{\alpha+1}{\alpha}} &= C_m x_{opt}^{\frac{\alpha+1}{\alpha}} \\
\left(\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}}\right) (P-x_{opt}) &= C_m^{\frac{\alpha}{1+\alpha}} x_{opt} \\
x_{opt} &= P \frac{\sum_{i=1}^{m-1} C_i^{\frac{\alpha}{1+\alpha}}}{\sum_{i=1}^{m} C_i^{\frac{\alpha}{1+\alpha}}}
\end{aligned}$$

The optimal value for P_1 is, therefore, $P_1 = x_{opt}$. We can easily deduce the result for all values f_i and RT.

In the following, we denote

$$f(i,j,k,P) = \left(\frac{P \cdot C_i^{\frac{\alpha}{1+\alpha}}}{\sum_{l=j}^{k} C_l^{\frac{\alpha}{1+\alpha}}} \right)^{\frac{1}{\alpha}}$$

4.2 Main problem resolution

We now focus on the main problem with frequencies bounded between $f_{\min}$ and $f_{\max}$. First note that if frequencies given by Lemma 1 are all in the good interval defined in Eq. (7), the solution is optimal. We consider without loss of generality that machines are ordered by increasing C_i, that is $C_1 \leq C_2 \leq \cdots \leq C_n$. We first prove the following result.

Lemma 2. *For two machines i and j, if $C_i \leq C_j$, then in the optimal solution $f_i \leq f_j$.*

Proof. If we just consider the run times of machines i and j, we obtain $x = \frac{C_i}{f_i} + \frac{C_j}{f_j} + CT_i + CT_j$. By optimality, exchanging frequencies of i and j has lower runtime and same power consumption. Thus, if $C_i < C_j$,

$$\begin{aligned}
\frac{C_i}{f_i} + \frac{C_j}{f_j} + CT_i + CT_j &\leq \frac{C_i}{f_j} + \frac{C_j}{f_i} + CT_i + CT_j \\
\frac{C_i}{f_i} + \frac{C_j}{f_j} &\leq \frac{C_i}{f_j} + \frac{C_j}{f_i} \\
\frac{C_j - C_i}{f_j} &\leq \frac{C_j - C_i}{f_i} \\
f_i &\leq f_j
\end{aligned}$$

If $C_i = C_j$, Lemma 1 states that the optimal frequencies are equal for same duration tasks.

We can conclude that in the optimal solution, the frequencies f_j are of increasing values of j. We define $j_{\min}$ and $j_{\max}$ as the last index of a machine at frequency $f_{\min}$ (0 if no machine is at $f_{\min}$) and the first index of a machine at $f_{\max}$ ($n+1$ if no machine is at f_{max}). Then, the machines from

1 to $j_{\min}$ run at frequency $f_{\min}$, the machines from $j_{\max}$ to m run at frequency $f_{\max}$, and the machines between $j_{\min}+1$ and $j_{\max}-1$ operate at a frequency defined by Lemma 1. More precisely, for $j_{\min} < j < j_{\max}$, the optimal frequency of machine j is $f_j = f(j, j_{\min}+1, j_{\max}-1, P_{bound})$ with $P_{bound} = P - j_{\min} \cdot f_{\min}^{\alpha} - (m - j_{\max} + 1) \cdot f_{\max}^{\alpha}$. This corresponds to an optimal solution of the problem for this subset of machines.

In such a solution, the runtime can be computed as follows:

$$RT = \frac{\sum_{k=1}^{j_{\min}} C_k}{f_{\min}} + P_{bound}^{\frac{1}{\alpha}} \left(\sum_{k=j_{\min}+1}^{j_{\max}-1} C_k^{\frac{\alpha}{\alpha+1}} \right)^{\frac{\alpha+1}{\alpha}} + \frac{\sum_{k=j_{\max}}^{m} C_k}{f_{\max}} + \sum_j CT_j$$

We denote $S_{\min} = \sum_{k=1}^{j_{\min}} C_k$, $S_{\max} = \sum_{k=j_{\max}}^{m} C_k$, $S_{RT} = \sum_{k=j_{\min}+1}^{j_{\max}-1} C_k^{\frac{\alpha}{\alpha+1}}$ and $CT = \sum_j CT_j$. Thus, we obtain:

$$RT = \frac{S_{\min}}{f_{\min}} + P_{bound}^{\frac{1}{\alpha}} S_{RT}^{\frac{\alpha+1}{\alpha}} + \frac{S_{\max}}{f_{\max}} + CT$$

The problem now is to determine the optimal values for $j_{\min}$ and $j_{\max}$. The main constraint for these values corresponds to frequency bounds. The frequencies computed by Lemma 1 need to be contained in the bounds of Eq. (7). Formally,

$$\forall j_{\min} < j < j_{\max}, f_{\min} \leq f(j, j_{\min}+1, j_{\max}-1, P_{bound}) \leq f_{\max} \tag{8}$$

As C_j are in increasing order, we can simply verify

$$f(j_{\min}+1, j_{\min}+1, j_{\max}-1, P_{bound}) \geq f_{\min} \tag{9}$$

and

$$f(j_{\max}-1, j_{\min}+1, j_{\max}-1, P_{bound}) \leq f_{\max} \tag{10}$$

With the current notation, we have :

$f(j_{\min}+1, j_{\min}+1, j_{\max}-1, P_{bound}) = \left(\frac{P \cdot C_{j_{\min}+1}^{\frac{\alpha}{1+\alpha}}}{S_{RT}} \right)^{\frac{1}{\alpha}}$ and

$f(j_{\max}-1, j_{\min}+1, j_{\max}-1, P_{bound}) = \left(\frac{P \cdot C_{j_{\max}-1}^{\frac{\alpha}{1+\alpha}}}{S_{RT}} \right)^{\frac{1}{\alpha}}$.

We check all possibilities for $j_{\min}$ and $j_{\max}$ between 1 and m, as described in Algorithm 1, which enumerates all possible values for $j_{\min}$ between 0 and m, and for $j_{\max}$ between 1 and $m+1$. For each of these values, it updates the values of $S_{\min}$, $S_{\max}$ and S_{RT} (lines 3-5). Then, line 7 verifies if the current values $j_{\min}$ and $j_{\max}$ constitute a valid solution for constraints (6), (9) and (10). The last lines (8-11) compute the corresponding runtime and update the objective value $MinRT$ if necessary. The algorithm has a quadratic complexity $O(m^2)$, due to the constant time to update variables $S_{\min}$, $S_{\max}$ and S_{RT}.

Input: $f_{min}, f_{max}, \alpha, [(C_1, CT_1), \cdots (C_m, CT_m)]$
$MinRT = +\infty$; $S_{total} = \sum_{i=1}^{m} C_i$; $S_{\min} = 0$; $S_{\max} = S_{total}$; $S_{RT} = 0$;
for $j_{\min} = 0$ *to* m **do**
  $S_{\max} = S_{total} - S_{\min}$; $S_{\min} += C_{j_{min}}$; $S_{RT} = -C_{j_{\min}}^{\frac{\alpha}{\alpha+1}}$;
  for $j_{\max} = j_{\min} + 1$ *to* $m + 1$ **do**
    $S_{\min} -= C_{j_{\max}-1}$; $S_{RT} += C_{j_{\max}-1}^{\frac{\alpha}{\alpha+1}}$;
    $P_{bound} = P - j_{\min} * f_{\min}^{\alpha} - (m - j_{\max} + 1) * f_{\max}^{\alpha}$;
    if $P_{bound} \geq 0$ **and** $f(j_{\min} + 1, j_{\min} + 1, j_{\max} - 1, P_{bound}) \geq f_{\min}$ **and** $f(j_{\max} - 1, j_{\min} + 1, j_{\max} - 1, P_{bound}) \leq f_{\max}$ **then**
      $RT_{current} = \frac{S_{\min}}{f_{\min}} + P_{bound}^{\frac{1}{\alpha}} S_{RT}^{\frac{\alpha+1}{\alpha}} + \frac{S_{\max}}{f_{\max}}$;
      if $RT_{current} < MinRT$ **then**
        $MinRT = RT_{current}$;
      end
    end
  end
end
Return $MinRT$;

Algorithm 1: Algorithm for optimal frequencies without allocation

4.3 An Example

We use again the example in Section 3.1, with 5 tasks with parameters (C_j, CT_j): $[(1,5),(6,6),(7,3),(8,2),(20,20)]$, global parameters $f_{min} = 2$, $f_{max} = 5$ and $\alpha = 3$, and a power cap of 200. Obtaining optimal values according to Section 4.1 leads to frequency values between 2.05 and 4.33 that is valid for frequency limits. It corresponds to a total runtime of 47.4. This property holds for a power cap ranging between 186.8 (in which case the optimal frequency for task 1 is 2) and 308.6, with an optimal frequency for task 5 of 5. Below a power cap of 186.8, the optimal frequency for task 1 will be $f_{min} = 2$ and above a power cap of 308.6, the frequency of task 5 will be fixed at 5.

If we consider a variant with task parameters C_j: $[1, 6, 7, 8, 30]$, the optimal frequency for the first task without frequency limits is 1.96, below f_{min}. Applying frequency limits, this task will run at minimum frequency, and the remaining tasks will follow Lemma 1 applied on the 4 remaining tasks with power 192.

Using parameters C_j: $[1, 6, 7, 8, 40]$ and a power cap of 235, the optimal frequency values without bounds range from 1.99 to 5.01. The optimal solution in the general problem is then obtained with task 1 running at frequency 2 and task 5 at frequency 5.

5 Conclusion

This paper has presented a model and a solution to select CPU frequencies for a set of servers belonging to a cloud provider or a data center so that a global power cap constraint can be met while the execution time of the tasks allocated

to these servers is minimized. To achieve this, the paper relies on modelling tasks in terms of their CPU boundedness and exploiting the fact that CPU-bound tasks are more impacted by any frequency reduction than I/O-bound tasks. The preliminary work in this paper suggests that there is potential in producing energy-efficient solutions that could be used in practice to find economical solutions to operate large data centers under power capping. Additional work could evaluate our approach in real-world environments while it could evaluate and model the CPU-boundedness of tasks in different workloads more elaborately.

References

1. Arroba, P., Moya, J.M., Ayala, J.L., Buyya, R.: Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers. Concurrency and Computation: Practice and Experience **29**(10), e4067 (2017). https://doi.org/10.1002/cpe.4067
2. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: Proceedings of the 2009 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 693–701. https://doi.org/10.1137/1.9781611973068.76
3. Bansal, N., Kimbrel, T., Pruhs, K.: Dynamic speed scaling to manage energy and temperature. In: 45th Annual IEEE Symposium on Foundations of Computer Science. pp. 520–529. IEEE (2004). https://doi.org/10.1109/FOCS.2004.24
4. Chen, J.J., Kuo, T.W.: Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics. In: 2005 International Conference on Parallel Processing (ICPP'05). pp. 13–20. IEEE (2005). https://doi.org/10.1109/ICPP.2005.53
5. Conoci, S., Di Sanzo, P., Pellegrini, A., Ciciani, B., Quaglia, F.: On power capping and performance optimization of multithreaded applications. Concurrency and Computation: Practice and Experience **33**(13), e6205 (2021). https://doi.org/10.1002/cpe.6205
6. Costero, L., Igual, F.D., Olcoz, K.: Dynamic power budget redistribution under a power cap on multi-application environments. Sustainable Computing: Informatics and Systems **38**, 100865 (2023). https://doi.org/10.1016/j.suscom.2023.100865
7. Etinski, M., Corbalan, J., Labarta, J., Valero, M.: Optimizing job performance under a given power constraint in HPC centers. In: Proceedings of the International Green Computing Conference. pp. 257–267. IEEE (2010). https://doi.org/10.1109/GREENCOMP.2010.5598303
8. Etinski, M., Corbalan, J., Labarta, J., Valero, M.: Understanding the future of energy-performance trade-off via DVFS in HPC environments. Journal of Parallel and Distributed Computing **72**(4), 579–590 (2012). https://doi.org/10.1016/j.jpdc.2012.01.006
9. Hsu, C.H., Kremer, U.: The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction. ACM SIGPLAN Notices **38**(5), 38–48 (2003). https://doi.org/10.1145/780822.781137
10. Katal, A., Dahiya, S., Choudhury, T.: Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing **26**(3), 1845–1875 (2022). https://doi.org/10.1007/s10586-022-03713-0

11. Lin, W., Luo, X., Li, C., Liang, J., Wu, G., Li, K.: An Energy-Efficient Tuning Method for Cloud Servers Combining DVFS and Parameter Optimization. IEEE Transactions on Cloud Computing **11**(4), 3643–3655 (2023). https://doi.org/10.1109/TCC.2023.3308927
12. Petoumenos, P., Mukhanov, L., Wang, Z., Leather, H., Nikolopoulos, D.S.: Power Capping: What Works, What Does Not. In: IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS). pp. 525–534 (2015). https://doi.org/10.1109/ICPADS.2015.72
13. Pietri, I., Sakellariou, R.: Cost-Efficient CPU Provisioning for Scientific Workflows on Clouds. In: Altmann, J., Silaghi, G.C., Rana, O.F. (eds.) Economics of Grids, Clouds, Systems, and Services. pp. 49–64. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-43177-2_4
14. Pietri, I., Sakellariou, R.: A Pareto-based approach for CPU provisioning of scientific workflows on clouds. Future Generation Computer Systems **94**, 479–487 (2019). https://doi.org/10.1016/j.future.2018.12.004
15. Rauber, T., Rünger, G.: A scheduling selection process for energy-efficient task execution on DVFS processors. Concurrency and Computation: Practice and Experience **31**(19), e5043 (2019). https://doi.org/10.1002/cpe.5043
16. Rizvandi, N.B., Taheri, J., Zomaya, A.Y., Lee, Y.C.: Linear Combinations of DVFS-Enabled Processor Frequencies to Modify the Energy-Aware Scheduling Algorithms. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. pp. 388–397 (2010). https://doi.org/10.1109/CCGRID.2010.38
17. Sundriyal, V., Sosonkina, M.: Modeling of the CPU frequency to minimize energy consumption in parallel applications. Sustainable Computing: Informatics and Systems **17**, 1–8 (2018). https://doi.org/10.1016/j.suscom.2017.12.002
18. Zhang, H., Hoffmann, H.: Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques. SIGPLAN Notices **51**(4), 545–559 (2016). https://doi.org/10.1145/2954679.2872375
19. Zhang, W., Zhang, Z., Zeadally, S., Chao, H.C., Leung, V.C.M.: Energy-efficient Workload Allocation and Computation Resource Configuration in Distributed Cloud/Edge Computing Systems With Stochastic Workloads. IEEE Journal on Selected Areas in Communications **38**(6), 1118–1132 (2020). https://doi.org/10.1109/JSAC.2020.2986614
20. Zhao, D., Samsi, S., McDonald, J., Li, B., Bestor, D., Jones, M., Tiwari, D., Gadepally, V.: Sustainable Supercomputing for AI: GPU Power Capping at HPC Scale. In: Proceedings of the 2023 ACM Symposium on Cloud Computing. p. 588–596. SoCC '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3620678.3624793