Stochastic modeling and optimization for power and performance control in DVFS systems

Youssef Ait El Mahjoub, Hind Castel-Taleb, Léo Le Corre

STOCHASTIC MODELING AND OPTIMIZATION FOR POWER AND PERFORMANCE CONTROL IN DVFS SYSTEMS

Youssef Ait El Mahjoub ✉
Léo Le Corre
Efrei Research Lab
Université Paris-Panthéon-Assas
30 Av. de la République, 94800, Villejuif, France
youssef.ait-el-mahjoub@efrei.fr
leo.le-corre@efrei.net

Hind Castel-Taleb
SAMOVAR
Télécom SudParis, Institut Polytechnique de Paris
91120 Palaiseau, France
hind.castel@telecom-sudparis.eu

KEYWORDS

Stochastic modeling; DVFS; Multi-Objective Optimization; Markov Chain; Performance; Power consumption analysis

ABSTRACT

The paper addresses the problem of performance-energy trade-off in DVFS (Dynamic Voltage Frequency Scaling) systems. We propose a stochastic hybrid model between hysteresis models and server block models. We provide a closed form for the steady-state distribution probability and we establish a "st" type order to compare the performance measures. The fast computation of power and performance measures leads to a multi-objective optimization analysis in two forms: a scalarization method and a Pareto based method. For the two approaches, we propose fast and efficient approximate algorithms that construct progressively an optimal solution. To discuss results, the model is used to simulate a physical server hosting several VMs (Virtual Machines) where we investigate optimal thresholds for the performance-energy trade-off.

INTRODUCTION

High power consumption remains an issue in data centers, cloud systems, and more generally in information and communication technology (ICT). Moreover, the advent of Artificial Intelligence, Data Mining, IoT, High Performance Computing, Crypto-mining brings the issue of power consumption into focus. Server virtualization provides a major reduction impact on energy consumption due to on demand resource allocation through many techniques such as consolidation, virtual machine migration, scheduling, load balancing, dynamic frequency scaling. In this work, we focus on the Dynamic Voltage Frequency Scaling (DVFS) in virtualized context for power monitoring and Quality of Service (QoS) preservation. Many manufacturers of CPUs and GPUs (Intel, AMD, ...) implement the DVFS approach in their computing devices as processors or controllers. This mechanism operates the system at several frequencies and voltages denoted as "Pstates". The frequency and power increase in higher Pstates. The highest Pstate runs the system at full rate and full voltage. But during off-peak periods, the clock can go down considerably, saving a significant portion of the power at full speed ([1]).

This work is composed of two major parts. The first one consists on modeling the DVFS system by a Markov chain approach where we propose a closed form formula for steady-state distribution probability. Then we proceed with a numerical multi-objective analysis by proposing two fast and approximate algorithms for the performance and power consumption trade-off. To model DVFS system, most analytical methods are based on stochastic modeling as: in ([16]) a Petri Net model is proposed for dynamic scaling and VM migration in Energy-Aware cloud system; in ([4]) a random variable analysis is set for estimating the relationship between workload and power consumption in multi-core processor; while in ([22]) a Hidden Markov Model is used with a predicting algorithm for hidden states of the system applied to a multi-core DVFS for energy-time trade-off; Markovian Decision Process (MDP) also is proposed ([2]), for models with tasks deadlines and unbounded state space and speed rates. These models often suffer from the explosion of the set of states. However, queueing theory offers a range of models that can address this type of problems and solves systems efficiently with less concern for the quality of the solution or the scalability of the problem. In this regard one notes Ghandi's works, in particular, for the optimization of the distribution of energy in k-server Farm (using DVFS levels) ([12]). Some works also model the DVFS system as an $M/M/1$ ([3]) or $GI/G/K$ ([17]) queue for power management purpose. System modeling we propose is based on a threshold policy that is similar to Hysteresis models ([14], [20]) but adapted to DVFS systems. Hysteresis models are based on two threshold vectors. When the number of tasks exceeds an activation threshold, a new server is added (instantaneously: [20], or with non-negligible time: [14]) and when the number of tasks falls below an inverse threshold, a server is removed from the system. Similar approach can be found in Mitrani's model ([21]), where system is limited to two thresholds to manage a group of block reserved servers with exponential activating time. Note

that in the last two models, the servers are homogeneous and therefore have the same service rate. The DVFS model that we propose is a hybrid model between the classical hysteresis model and the block model of Mitrani. Indeed, we have a single vector of thresholds that instantaneously switches UP (with non-negligible power cost) or switches DOWN (with negligible power cost) frequency of a group of activate servers. We represent the system by a birth-death process from which we obtain a closed form for the steady state probability distribution, we derive the mean response time, tasks dropping rate probability and mean power consumption which we will optimize in the second phase.

The second part of this work consists in a Multi-Objective Optimization (MOO). This process seeks to find optimal threshold vector(s) that minimizes both response time and mean power consumption. This matter is often related to a decision problem. Many hysteresis studies address this problem with optimal control models through MDPs (Markovian Decision Process). However, convergence of classical MDP algorithms is subject to the hysteresis property of optimal policy. This property is proven for $M/M/1$ queue in ([19]) and $M/M/C$ queue in ([25]). An interesting comparative study is proposed in ([26]) where authors compare a panel of heuristic Markov chain resolution approaches (using states aggregation) with an MDP decision approach, stating that this later is more efficient for high scale simulations. One notes that all the discussed methods consider a scalarized global optimization reward function and provide a single solution to service provider to ensure a given level of QoS with reasonable energy consumption. In our study, we address the non-convex problem of optimal thresholds with: (i) A scalarization Weighted Sum Approach (WSM); (ii) A non-dominated Pareto approach that consists in proposing a set of optimal (i.e. compromise) solutions instead of a single one. In both approaches, we propose a fast and greedy approximate method to construct optimal vector(s).

This work is an extension and enhancement of our previous study ([27]), precisely, this paper is not limited to systems with $N = 2$ Pstates. The stochastic modeling and optimization approach considers vector of thresholds rather than a single threshold value. This makes the optimization process more complex with both a factorial decision space and an objective space that is neither linear nor convex. Moreover, we treat the modeling phase in the case of finite systems (thus more realistic), and infinite denumerable systems.

The rest of the paper is organized as follows. In the next section, we describe the Markov chain model. We present the proof of the closed form of the steady state distribution dealing with vector of $N \geq 1$ Pstates. We derive the performance and power consumption formulations. We also provide an "st" (i.e. strong stochastic) comparison between the steady-state probability distributions derived from processes with comparable sequence of thresholds. This result induces the comparison of the performance measures. In Section III, we focus on the numerical multi-objective optimization, proposing two approximate numerical algorithms either for the scalarization approach or the Pareto front approach. Finally, we present numerical results by comparing the proposed algorithms to some well-known MOO methods and analyze in more detail the scenario of a physical server hosting multiple VMs under DVFS policy.

MODEL DESCRIPTION AND MARKOV CHAIN ANALYSIS

We first recall the Pstates support for a DVFS system (Table I). The table represents the frequency f_i (in GHz) from the lowest Pstate P_1 to the highest one P_N. Power consumption p_i increases with the frequency f_i and $0 < f_1 < f_2 < \cdots < f_N$ and $0 < p_1 < p_2 < \ \ldots \ < p_N$. In this work, we model the Pstate

TABLE I: Pstates support for a DVFS system

Pstates	P_1	P_2	$\ldots$	P_N
Frequency (GHz)	f_1	f_2	$\ldots$	f_N
Power (W)	p_1	p_2	$\ldots$	p_N

system as a multi-server queue. We assume that

- The system contains C computing units (or VMs).
- External arrivals of tasks follow independent Poisson process with rate λ.
- Service rates are distributed according to exponential distributions.
- Task scheduling discipline is FCFS (First Come, First Served).
- Frequency is the number of cycles per second. CPI is the number of Cycles per Instruction. Tasks may require "b" instructions (with $b > 0$). Hence service rate of a single computing unit is $\mu_i = \frac{f_i}{b*CPI}$.

We represent the system as a birth-death process with N Pstates and a capacity B that could be either finite or infinite denumerable. Service rate increases (or decreases) dynamically and progressively according to the amount of tasks in the system. The latter is controlled by a vector of thresholds $\mathbf{\Gamma} = [th_1, th_2, \ldots, th_{(N-1)}]$ with $(0 < th_1 < th_2 < \ \ldots \ < th_{(N-1)} < B)$.

TABLE II: Main notations for Markov process modeling

Notation	**Description**
$\mathscr{P}$	Entry Pstates vector of size N
$\mathbf{\Gamma}$	Thresholds vector of size $N-1$
B	Maximum number of tasks in the system
C	Total number of computing units (i.e. VMs)
λ	Tasks arrival rate
μ_i	Service rate of a unit in Pstate i
p_i	Power consumption of a unit in Pstate i
$p_{i,Id}$	Power consumption of an idle unit in Pstate i
s_i	Power consumption of a switching unit to Pstate i
S_Γ	System described by $\{X^t_{\mathbf{\Gamma}},\ t \geq 0\}$ CTMC and steady-state vector $\Pi_{\mathbf{\Gamma}}$
$\mathbb{E}[X_{\mathbf{\Gamma}}]$	Mean number of tasks in the system
$\overline{T}_i$ or $\overline{T}_{\mathbf{\Gamma}}$	Mean response time of a mono-Pstate system in Pstate i or multi-Pstates system with thresholds $\mathbf{\Gamma}$
$PW_{\mathbf{\Gamma}}$	Mean power consumption of a multi-Pstates system with thresholds $\mathbf{\Gamma}$

Model and closed form for the steady-state distribution

Let (x) be the number of tasks in the system, then DVFS system instantly and dynamically adjusts its frequency and power to the corresponding Pstate. In next, we analyse the

system in both finite $B < \infty$ and infinite case $B \to \infty$. The existence of a steady-state distribution in infinite models is subject to the stability condition $\lambda < C\mu_N$. We suppose that all systems we study are stable. The system contains C computing units. Therefore $min(C,x)$ tasks are in service, and $(x - min\{C,x\})^+$ are queued. Under the classical assumptions mentioned previously, $\{X^t_{\mathbf{\Gamma}},\ t \geq 0\}$ is a Continuous Time Markov Chain (CTMC) with transitions

$$\begin{array}{rcl}(x) & \to & (\min\{x+1,B\}), \text{ with rate } \lambda,\\ (x) & \to & (\max\{0,x-1\}), \text{ with rate } min\{x,C\}\cdot\mu(x)\end{array}$$

and service rate

$$\mu(x) = \begin{cases} \mu_1 = \frac{f_1}{b*CPI} & if\ (x \leq th_1),\\ \mu_2 = \frac{f_2}{b*CPI} & if\ (th1 < x \leq th_2),\\ . & \\ . & \\ \mu_N = \frac{f_N}{b*CPI} & if\ (th_{(N-1)} < x \leq B). \end{cases} \tag{1}$$

***Theorem** 1:* The Markov chain is a birth-death process. Then, under stability condition $\lambda < C\mu_N$ if the system is infinite, we can derive $\Pi_{\mathbf{\Gamma}}(x)$ the steady-state probability.

$$\begin{array}{lll} Let & \Psi(x) = \prod_{k=1}^{x} min\{k,C\}\mu(k), & \\ and & R = max\{C, th_{(N-1)}\}, & then \end{array} \tag{2}$$

$$\Pi_{\mathbf{\Gamma}}(x) = \begin{cases} \frac{\lambda^x}{\Psi(x)}\Pi_{\mathbf{\Gamma}}(0) & \forall\ \ 0 < x \leq R,\\ \lambda^x \left[\Psi(R)(C\mu_N)^{x-R}\right]^{-1}\Pi_{\mathbf{\Gamma}}(0) & \forall\ \ x > R. \end{cases} \tag{3}$$

where $\Pi_{\mathbf{\Gamma}}(0) =$

$$\begin{cases} \left(1 + \frac{\lambda^{R+1}}{(C\mu_N - \lambda)\Psi(R)} + \sum_{x=1}^{R}\frac{\lambda^x}{\Psi(x)}\right)^{-1}, & if\ B \to \infty,\\ \left(1 + \frac{\lambda^{R+1}}{(C\mu_N - \lambda)\Psi(R)}\left(1 - \left(\frac{\lambda}{C\mu_N}\right)^{B-R}\right) + \sum_{x=1}^{R}\frac{\lambda^x}{\Psi(x)}\right)^{-1}, & \\ if\ B < \infty. \end{cases} \tag{4}$$

Proof: In Equation (5), we present the classical birth-death equations for the steady-state distribution of that model.

$$\Pi_{\mathbf{\Gamma}}(x) = \begin{cases} \frac{\lambda}{min\{x,C\}\mu(x)}\Pi_{\mathbf{\Gamma}}(x-1) & \forall\ \ 0 < x \leq R,\\ \left(\frac{\lambda}{C\mu_N}\right)^{x-R}\Pi_{\mathbf{\Gamma}}(R) & \forall\ \ x > R. \end{cases} \tag{5}$$

After simple substitutions (i.e. to express $\Pi_{\mathbf{\Gamma}}(x)$ as a function of $\Pi_{\mathbf{\Gamma}}(0)$ for all $x > 0$), we get Equation (3). Finally, after the normalization of probabilities $\sum_{x=0}^{B}\Pi_{\mathbf{\Gamma}}(x) = 1$, either in finite of infinite stable system, we obtain Equation (4). ■

Mean performance measures and power consumption

Let $S_{\mathbf{\Gamma}}$ be the queuing model of a DVFS system defined by thresholds vector $\mathbf{\Gamma}$.

***Corollary** 1:* In infinite system $S_{\mathbf{\Gamma}}$, we express the mean response time as :

$$\overline{T}_{\mathbf{\Gamma}} = \Pi_{\mathbf{\Gamma}}(0)\left[\frac{(Cu_N)^{R+1}}{(Cu_N - \lambda)\Psi(R)} + \sum_{x=1}^{R} x\lambda^{x-1}\left(\frac{1}{\Psi(x)} - \frac{(Cu_N)^{R-x}}{\Psi(R)}\right)\right] \tag{6}$$

Proof: The proof derives directly from Theorem 1. From Little's law, mean response time is expressed as :

$$\overline{T}_{\mathbf{\Gamma}} = \frac{E[X_{\mathbf{\Gamma}}]}{\lambda} = \frac{1}{\lambda}\sum_{x=1}^{+\infty} x\Pi_{\mathbf{\Gamma}}(x)$$

$$\Rightarrow\ \overline{T}_{\mathbf{\Gamma}} = \frac{1}{\lambda}\left[\sum_{x=1}^{R} x\Pi_{\mathbf{\Gamma}}(x) + \sum_{x=R+1}^{+\infty} x\Pi_{\mathbf{\Gamma}}(x)\right]$$

$$\Rightarrow\ \overline{T}_{\mathbf{\Gamma}} = \frac{1}{\lambda}\Pi_{\mathbf{\Gamma}}(0)\left[\sum_{x=1}^{R}\frac{x\lambda^x}{\Psi(x)} + \frac{\lambda(Cu_N)^{R-1}}{\Psi(R)}\sum_{x=R+1}^{+\infty} x\left(\frac{\lambda}{Cu_N}\right)^{x-1}\right]$$

$$\Rightarrow\ \overline{T}_{\mathbf{\Gamma}} = \frac{1}{\lambda}\Pi_{\mathbf{\Gamma}}(0)\left[\sum_{x=1}^{R}\frac{x\lambda^x}{\Psi(x)} + \frac{\lambda(Cu_N)^{R-1}}{\Psi(R)}\left(\sum_{x=1}^{+\infty} x\left(\frac{\lambda}{Cu_N}\right)^{x-1}\right.\right.$$

$$\left.\left. - \sum_{x=1}^{R} x\left(\frac{\lambda}{Cu_N}\right)^{x-1}\right)\right].$$

Using the derivative of the converging geometric summation, and after simplification of terms, we obtain Equation (6). ■

Note that, in finite model, mean response time formula is

$$\overline{T}_{\mathbf{\Gamma}} = \frac{E[X_{\mathbf{\Gamma}}]}{\lambda(1 - \Pi_{\Gamma}(B))} = \frac{\sum_{x=1}^{B} x\Pi_{\mathbf{\Gamma}}(x)}{\lambda(1 - \Pi_{\Gamma}(B))}.$$

It becomes harder to simplify and much more verbose. Therefore, we decide to only present the equation in infinite denumerable case.

The power consumption model we consider is as follows: we recall that DVFS technique allows a system to remain switched ON while changing frequency to match the traffic. This avoids successive shutdown and restoration costs of big systems that can result in waste additional power and further latency to restore the same regime ([11]). Thus, the proposed system is initially in the lowest Pstate and then progressively adjusts its speed to workload. Power formula we propose supports three forms $PW^{(a)}_{\mathbf{\Gamma}}$, $PW^{(Id)}_{\mathbf{\Gamma}}$ and $PW^{(s)}_{\mathbf{\Gamma}}$ which are, respectively, the mean power consumption of the system in activity states, in Idle states and in switching UP frequency states from a lower Pstate to a higher one. Switching DOWN of Pstates is assumed to be negligible (refer to nap mode in [23]).

The power consumed by each active unit while the system hosts x tasks is :

$$\begin{cases} p_1 & if\ (x \leq th_1),\\ p_2 & if\ (th_1 < x \leq th_2),\\ . & \\ . & \\ p_N & if\ (th_{(N-1)} < x < B). \end{cases} \tag{7}$$

We note $p_{i,Id} = \alpha p_i$ with $0 \leq \alpha \leq 1$ the idle power and s_i the extra power consumed when switching to a higher Pstate i. A system that scales to a higher Pstate requires all computing units to be placed at a higher frequency, resulting in a total additional cost of $C * s_i$ times the probability of being in switching states.

***Lemma** 1:* Let $PW_{\mathbf{\Gamma}}$ be the total power consumption of a stable DVFS system $S_{\mathbf{\Gamma}}$. For the simplicity of the power consumption equation, we fix $th_0 = -1$ and $th_N = B$ (effective

thresholds are $[th1,\ th2,\ \ldots\ ,\ th_{(N-1)}]$). Then

$$PW_{\mathbf{\Gamma}} = \sum_{j=0}^{N-1}\left[p_{j+1}\sum_{x=(th_j)+1}^{th_{j+1}}\left(\alpha C+(1-\alpha)min(x,C)\right)\Pi_{\mathbf{\Gamma}}(x) + s_{j+1}C\Pi_{\mathbf{\Gamma}}(th_j)1_{j>0}\right]. \quad (8)$$

Proof: We first decompose the mean power formula as :

$$PW_{\mathbf{\Gamma}} = PW_{\mathbf{\Gamma}}^{(a)} + PW_{\mathbf{\Gamma}}^{(Id)} + PW_{\mathbf{\Gamma}}^{(s)}\ , \quad (9)$$

where

$$\begin{cases} PW_{\mathbf{\Gamma}}^{(a)} = \sum_{j=0}^{N-1}\left[\sum_{x=(th_j)+1}^{th_{j+1}}\Pi_{\mathbf{\Gamma}}(x)\left(min(x,C)p_{j+1}\right)\right], \\ PW_{\mathbf{\Gamma}}^{(Id)} = \sum_{j=0}^{N-1}\left[\sum_{x=(th_j)+1}^{th_{j+1}}\Pi_{\mathbf{\Gamma}}(x)\left(C-min(x,C)\right)\right. \\ \left. p_{j+1,Id}\right], \\ PW_{\mathbf{\Gamma}}^{(s)} = C\sum_{j=1}^{N-1}s_{j+1}\Pi_{\mathbf{\Gamma}}(th_j). \end{cases} \quad (10)$$

After simplification of terms, we obtain Equation (8). Note that $PW_{\mathbf{\Gamma}} \geq 0$ since $0 \leq \alpha \leq 1$ and all other terms are positive. The last summation uses s_{j+1} as the switching UP power is supposed to be related to the arriving Pstate $j+1$. ■

Comparison of performance measures

We first recall the "el" comparison between vectors of thresholds.

***Definition** 1:* Let $\mathbf{\Gamma}1$ and $\mathbf{\Gamma}2$ be two thresholds vectors, then: $\mathbf{\Gamma}1 \leq_{el} \mathbf{\Gamma}2 \ \Rightarrow\ \mathbf{\Gamma}1(j) \leq \mathbf{\Gamma}2(j)\ \ \forall j \in \{1,\ldots,N-1\}$.

In the next corollary we prove the existence of "st" (i.e. strong stochastic comparison) between steady-state probability distribution vector for two systems.

***Corollary** 2:* Assuming that, either finite or infinite systems, $S_{\mathbf{\Gamma}1}$ and $S_{\mathbf{\Gamma}2}$ has comparable thresholds, then :

$$\mathbf{\Gamma}1 \leq_{el} \mathbf{\Gamma}2 \quad \Rightarrow \quad \Pi_{\mathbf{\Gamma}1} \leq_{st} \Pi_{\mathbf{\Gamma}2}. \quad (11)$$

Proof: Let $\mu_x^{(S_{\mathbf{\Gamma}1})} = min\{x,C\}\mu(x)$ (the same for $\mu_x^{(S_{\mathbf{\Gamma}2})}$) be the services rate transition generated at state x in system $S_{\mathbf{\Gamma}1}$ (resp. $S_{\mathbf{\Gamma}2}$) then

$$\mathbf{\Gamma}1 \leq_{el} \mathbf{\Gamma}2 \ \Rightarrow\ \ \mu_x^{(S_{\mathbf{\Gamma}1})} \geq \mu_x^{(S_{\mathbf{\Gamma}2})}\ \ \forall\ state\ x,$$

also for all x, we have $\lambda_x^{(S_1)} = \lambda_x^{(S_2)} = \lambda$. Thus, from Stoyan theorem of birth-death processes (in [24], page 196-197, Theorem 5.2.21), we deduce that $\{X^t_{\mathbf{\Gamma}1},\ t \geq 0\} \leq_{st} \{X^t_{\mathbf{\Gamma}2},\ t \geq 0\}$. Since both systems are stable, then steady-state probability distributions are comparable. We get $\Pi_{\mathbf{\Gamma}1} \leq_{st} \Pi_{\mathbf{\Gamma}2}$. ■

***Lemma** 2:* (Comparison of mean number of jobs and response time, [27])
Let $\mathbb{E}[X_1]$, $\mathbb{E}[X_2]$ (resp. $\overline{T_1}, \overline{T_2}$) be the mean number of jobs (resp. the mean response time) for two infinite stable DVFS systems, then : $\Pi_1 \leq_{st} \Pi_2 \ \Rightarrow\ \mathbb{E}[X_1] \leq \mathbb{E}[X_2]$ *and* $\overline{T_1} \leq \overline{T_2}$.

***Corollary** 3:* In finite systems $S_{\mathbf{\Gamma}1}$ and $S_{\mathbf{\Gamma}2}$ with comparable thresholds, rejection rate is higher in $S_{\mathbf{\Gamma}2}$:

$$\mathbf{\Gamma}1 \leq_{el} \mathbf{\Gamma}2 \quad \Rightarrow \quad \lambda\Pi_{\mathbf{\Gamma}1}(B) \leq \lambda\Pi_{\mathbf{\Gamma}2}(B) \quad (12)$$

Proof: As a consequence of Corollary 2 we have :

$$\mathbf{\Gamma}1 \leq_{el} \mathbf{\Gamma}2 \quad \Rightarrow \quad \Pi_{\mathbf{\Gamma}1} \leq_{st} \Pi_{\mathbf{\Gamma}2}.$$

Note that "st" comparison between vector of probabilities is expressed as :

$$\Pi_{\mathbf{\Gamma}1} \leq_{st} \Pi_{\mathbf{\Gamma}2}\ \ iff\ \ \forall k \in \{1,2,\ldots,B\},\ \ \sum_{j=k}^{B}\Pi_{\mathbf{\Gamma}1}(j) \leq \sum_{j=k}^{B}\Pi_{\mathbf{\Gamma}2}(j)$$

Hence taking $k = B$, we deduce that $\lambda\Pi_{\mathbf{\Gamma}1}(B) \leq \lambda\Pi_{\mathbf{\Gamma}2}(B)$. ■

From the last two proposed corollaries and Lemma 2, we deduce the Corollary 4 that we consider in the optimization process when one only intends to optimize the response time.

***Corollary** 4:* In finite or infinite systems $S_{\mathbf{\Gamma}1}$ and $S_{\mathbf{\Gamma}2}$ with comparable thresholds, mean response time and mean number of jobs are higher in $S_{\mathbf{\Gamma}2}$.

$$\mathbf{\Gamma}1 \leq_{el} \mathbf{\Gamma}2 \quad \Rightarrow \quad \mathbb{E}[X_{\mathbf{\Gamma}1}] \leq \mathbb{E}[X_{\mathbf{\Gamma}2}]\ \ and\ \ \overline{T}_{\mathbf{\Gamma}1} \leq \overline{T}_{\mathbf{\Gamma}2} \quad (13)$$

Proof: In finite or infinite system, from Corollary 2 we have $\Pi_{\mathbf{\Gamma}1} \leq_{st} \Pi_{\mathbf{\Gamma}2}$. For infinite systems we derive from Lemma 2 that $\mathbb{E}[X_{\mathbf{\Gamma}1}] \leq \mathbb{E}[X_{\mathbf{\Gamma}2}]$ and $\overline{T}_{\mathbf{\Gamma}1} \leq \overline{T}_{\mathbf{\Gamma}2}$. However, for finite systems, response time formula (Little's law) is different since rejected tasks are not considered. But, mean number of jobs in the system formula remains the same and is bounded by the limit capacity of the system (B tasks). Hence it is straightforward, from Lemma 2 that $\mathbb{E}[X_{\mathbf{\Gamma}1}] \leq \mathbb{E}[X_{\mathbf{\Gamma}2}]$. It remains to demonstrate the response time comparison in the finite case. From Corollary 3 we have :

$$\Pi_{\mathbf{\Gamma}1}(B) \leq \Pi_{\mathbf{\Gamma}2}(B) \quad \Rightarrow \quad \frac{1}{\lambda(1-\Pi_{\mathbf{\Gamma}1}(B))} \leq \frac{1}{\lambda(1-\Pi_{\mathbf{\Gamma}2}(B))},$$

$$\Rightarrow \frac{E[X_{\mathbf{\Gamma}1}]}{\lambda(1-\Pi_{\mathbf{\Gamma}1}(B))} \leq \frac{E[X_{\mathbf{\Gamma}1}]}{\lambda(1-\Pi_{\mathbf{\Gamma}2}(B))} \leq \frac{E[X_{\mathbf{\Gamma}2}]}{\lambda(1-\Pi_{\mathbf{\Gamma}2}(B))},$$

$$\Rightarrow \overline{T}_{\mathbf{\Gamma}1} \leq \overline{T}_{\mathbf{\Gamma}2}.$$

Hence, Equation (13) is proved in finite and infinite case. ■

In the following, we propose an upper bound and lower bound for performance measures. These bounds will be used for the normalization of objectives in optimization phase.

***Corollary** 5:* Let S_1 (resp. S_N) be the mono-Pstate DVFS system, i.e, system that considers only Pstate1 (resp. Pstate$N$). Let $\{X^t_1,\ t \geq 0\}$ and Π_1 (resp. $\{X^t_N,\ t \geq 0\}$ and Π_N) be CMTC description and steady-state probability distribution for, either both finite or both infinite systems S_1 (resp. S_N). Then for any multi-Pstates system $S_{\mathbf{\Gamma}}$:

$$\begin{cases} \mathbb{E}[X_N] \leq \mathbb{E}[X_{\mathbf{\Gamma}}] \leq \mathbb{E}[X_1] \\ \overline{T}_N \leq \overline{T}_{\mathbf{\Gamma}} \leq \overline{T}_1 \\ \Pi_N(B) \leq \Pi_{\mathbf{\Gamma}}(B) \leq \Pi_1(B) \end{cases} \quad (14)$$

The last inequality only holds for finite models, due to task dropout from a full system.

Proof: First, notice that when the system considers only one Pstate, then S_1 and S_N are reduced to an $M/M/C$ (resp. $M/M/C/B$) queue in infinite (resp. finite) system with service rate μ_1 and μ_N. By applying Stoyan theorem between S_1 and

$S_{\mathbf{\Gamma}}$, also between $S_{\mathbf{\Gamma}}$ and S_N and knowing that $\mu_1 \leq \mu_N$, then we obtain :

$$\{X_N^t,\ t \geq 0\} \leq_{st} \{X_{\mathbf{\Gamma}}^t,\ t \geq 0\} \leq_{st} \{X_1^t,\ t \geq 0\}$$
$$\Rightarrow \quad \Pi_N \leq_{st} \Pi_{\mathbf{\Gamma}} \leq_{st} \Pi_1.$$

For infinite or finite models, mean number of jobs and response time are comparable since we establish the "st" comparison between S_N and $S_{\mathbf{\Gamma}}$, and between $S_{\mathbf{\Gamma}}$ and S_1, hence using the same justification approach of Corollary 4, we obtain the two first equations of (14). Finally, the last equation is a direct consequence of the "st" comparison in finite systems for the state $k = B$. ■

MUTLI-OBJECTIVE OPTIMIZATION ANALYSIS

The threshold vector $\mathbf{\Gamma}$ is an inherent element of the system dynamics. This vector drastically affects the performance and consumption. The optimization process seeks to find the optimal thresholds (or compromise thresholds) between performance (mean response time and reject probability of tasks) and power consumption. Note that, these three objectives derive from the closed form proposed in Theorem 1. As mentioned in our first work ([27]), the analysis of a system with one or two Pstates remains very reasonable. However, the behavior of the system becomes more challenging when the number of Pstates is increasing ($N > 2$). We, nevertheless, provide Corollary 4 and 5 to compare performance metrics only based on the parameters of the system. On the other hand, the mean power consumption is difficult to analyse as it is neither monotonous (in the sense of "st" comparison) neither a convex function.

First, we can reduce the optimisation space to two objectives: power consumption and response time. Probability of rejecting tasks is positively correlated to the response time (i.e fast servers leads to high consumption, low response time and low reject probability). We can also state that a system with high threshold values switches tardily to higher Pstates, thus consumes less but shows deteriorated performance in terms of response time and rejection rate. In the following, we study this balanced behavior between power consumption and response time and provide optimal thresholds for this trade-off.

We express the non-convex optimization problem as :

$$\left[\begin{array}{l} \min \quad PW_{\mathbf{\Gamma}} \\ \min \quad \overline{T}_{\mathbf{\Gamma}} \\ s.t. \quad \mathbf{\Gamma} = [th_1, th_2, \ldots, th_{(N-1)}], \\ \ \forall\ i \in \{1, N-2\},\ 0 < th_i < th_{i+1} < B, \\ \ \forall\ i \in \{1, N-1\},\ th_i \in \mathbb{N}^*. \end{array}\right. \tag{15}$$

Let $\mathbb{D}$ be decision space of this problem, then $|\mathbb{D}|$ is a function of N and B, with $B \geq N \geq 2$:

$$|\mathbb{D}| = \frac{(B-1)!}{(N-1)!(B-N)!} = \frac{\Pi_{i=1}^{N-1}(B-N+i)}{(N-1)!}. \tag{16}$$

It is, therefore, necessary to consider fast resolution algorithms. To solve this problem, we analyse two main approaches ([15]) in MOO. In each one we propose a greedy approximation algorithm and we compare its results to some classical algorithms.

TABLE III: Main notations for MOO analysis

Notation	Description
$\mathbf{\Gamma}^*$	Optimal vector of thresholds obtained with a WSM method
$\mathbb{D}$	Decision space set
$\mathbb{H}(th,k)$	Set of all feasible integer values at position k having th as a fixed value in the last constructed vector Γ_{k-1}^*
$\mathbb{V}(\Gamma,k)$	Set of feasible neighbors values at position k obtained by adding (or emitting) $1,2,...,\Delta$ to $\Gamma(k)$
$\mathbb{S}_k$	Non-dominated set obtained at iteration k in approximate Pareto Algorithm
$\mathbb{S}$	Exact Pareto set obtained with Kung algorithm

Single solution approach

Here, we consider the Weighted Sum Method (WSM) which consists on the scalarization ([15]) of the set of objectives into a single one. This method is based on the additive utility assumption ([9]). It requires a high-level information that estimates the relative importance of each objective. Let $w \in [0,1]$ (resp. $1-w$) be the weight assigned to the response time (resp. mean power consumption). We also need to normalize objectives. Let $T_{\Gamma}^{'}$ (resp. $PW_{\Gamma}^{\prime}$) be the normalized objectives. Hence Equation (15) becomes :

$$\left[\begin{array}{l} \min \quad wT_{\Gamma}^{\prime} + (1-w)PW_{\Gamma}^{\prime} \\ s.t. \quad \mathbf{\Gamma} = [th_1, th_2, \ldots, th_{(N-1)}], \\ \ \forall\ i \in \{1, N-2\},\ 0 < th_i < th_{i+1} < B, \\ \ \forall\ i \in \{1, N-1\},\ th_i \in \mathbb{N}^*. \end{array}\right. \tag{17}$$

Where

$$T_{\Gamma}^{\prime} = \frac{\overline{T}_{\mathbf{\Gamma}} - \overline{T}_{\mathbf{\Gamma},min}}{\overline{T}_{\mathbf{\Gamma},max} - \overline{T}_{\mathbf{\Gamma},min}} \quad and \quad PW_{\Gamma}^{\prime} = \frac{PW_{\mathbf{\Gamma}} - PW_{\mathbf{\Gamma},min}}{PW_{\mathbf{\Gamma},max} - PW_{\mathbf{\Gamma},min}}.$$

Lemma 3: Normalization approach we consider consists in bringing all objectives to the range $[0,1]$, then for a system with N Pstates :

$$\overline{T}_{\mathbf{\Gamma},min} = \overline{T}_N \quad and \quad \overline{T}_{\mathbf{\Gamma},max} = \overline{T}_1,$$
$$PW_{\mathbf{\Gamma},min} = Cp_{1,Id} \quad and \quad PW_{\mathbf{\Gamma},max} = Cp_N + C\sum_{j=1}^{N-1} s_{j+1}$$

Proof: For response time objective, we use Corollary 5 to derive the upper and lower bound. However, strong stochastic "st" comparison does not hold for power consumption (Equation (8)). We, then, propose that $PW_{\mathbf{\Gamma},min}$ is the power consumption when all servers are in idle state in the lowest Pstate, and inversely $PW_{\mathbf{\Gamma},max}$ is the power consumption when all servers are active in the highest Pstate plus the switching UP power of the intermediate Pstates. ■

Lemma 4: If one wants to optimize only the response time $\overline{T}_{\mathbf{\Gamma}}$ (i.e. $w = 1$), then optimal solution is $\mathbf{\Gamma}^* = [1,2,\ldots,N-1]$.

Proof: According to Corollary 4. A lower threshold vector (in the sense of $\leq_{el}$ comparison), presents a lower response time. Hence, vector $[1,2,\ldots,N-1]$ which is the lowest feasible solution provides the best performance measures. In this case, higher Pstates are quickly attainable. Otherwise, if one wants to optimize only power consumption (i.e. $w = 0$) or both objectives (i.e. $0 < w < 1$). One should proceed with an optimization method. ■

The approximate greedy approach we propose (details in Algorithm 2) is based on a local resolution from a two Pstates system and we progressively add a new Pstate level until achieving the N Pstates system: we solve the problem with $N=2$ Pstates with a naive method, it is not time consuming since we only have one threshold (i.e. a single decision variable). Let $\mathbf{\Gamma}_1^* = [th_1]$ be the optimal solution at iteration 1. Then, for the next iteration, we construct the new threshold vector as $\mathbf{\Gamma}_2^* = \mathbf{\Gamma}_1^* \boxplus th_2$ where $th_2 \in \{th1+1, \ldots, B-N+2\}$. We append (i.e. $\boxplus$ operator) the new th_k to threshold vector $\mathbf{\Gamma}_{k-1}^*$. We continue this process until reaching N Pstates. We also add an exploration step in this algorithm. In order to improve the quality of the solution, we explore the neighborhood of the last element of the previous iteration. Let $\Delta \geq 0$ be the exploration factor, then we define the two sets as: $\mathbb{H}(th,k)$ is the set of possible values in the position k knowing that at position $k-1$ we have the value th, this set derives from the constraint of strict inferiority between thresholds. The second set $\mathbb{V}(\Gamma,k)$ represents the neighborhood possibilities of a given threshold value in position k.

$$\begin{cases} \mathbb{H}(th,k) = \{th+1,\ th+2,\ \ldots,\ B-N+k\}, \\ \mathbb{V}(\Gamma,k) = \{m\ /\ \forall i \in \{0,1,\ldots,\Delta\}, \\ \mathbf{\Gamma}(k-1) < m = \mathbf{\Gamma}(k) \mp i < B-N+k\ \}. \end{cases}$$

Δ parameter improves the quality of the solution but also increases time complexity. In many cases $\Delta = 0$ is sufficient to reach the optimal solution (see Tables V, VI and VII).

***Lemma** 5:* Let $b_k \leq B$ and $c_k \leq B$ then temporal complexity of the proposed greedy approximate algorithm is $B\sum_{k=1}^{N-1} b_k c_k$ in average case and $O((N-1)B^2)$ in best case.

Proof: Solve() method's (i.e. Algorithm 1) complexity is $O(B)$ (one can refer to the latest version of the XBorne tool; [10] to solve numerically classical birth-death processes). And, for each iteration k, we calculate the minimum of the objective functions in the first set $\mathbb{H}(th,k)$ which is upper bounded by B (i.e. $b_k = |\mathbb{H}(th,k)| \leq B$) that depends on the neighborhood set $c_k = |\mathbb{V}(\Gamma,k)| \leq B$. In best case, one considers $\Delta = 0$ hence $c_k = |\mathbb{V}(\Gamma,k)| = 1$, therefore complexity reaches $B\sum_{k=1}^{N-1} b_k$ which is in the order of $O((N-1)B^2)$. Note that a naive research method is in the order of $O(B|\mathbb{D}|)$. ■

This algorithm, indeed, does not ensure a global optimal solution, but besides being faster, it benefits from an optimal sub-structure property that produces an optimal solution in several investigated cases.

Algorithm 1: Solve($\mathbf{\Gamma}$,$\mathscr{P}$, w)

Input : Vector of thresholds $\mathbf{\Gamma}$, vector of Pstates $\mathscr{P}$, weight w.

Output: Calculating rewards given a vector of thresholds

1 Calculate the steady-state distribution (Theorem 1)
2 Derive the mean response time $\overline{T}_\Gamma$ and mean power consumption PW_Γ (Lemma 2)
3 Normalize the objectives (Lemma 3)
4 Derive the cost function using weight w.

Although the weighted sum approach is simple to analyse and reaches an optimal solution, it suffers from many difficulties ([13]): we need to know the weights (w), the inability to find some Pareto-optimal solutions (those in non-convex region), also each algorithm execution provides a single solution. In next, we analyse the Pareto-Optimal approach which consists in proposing a set of non-dominated solutions (i.e set of compromise solutions).

Algorithm 2: Approximate greedy algorithm for optimal thresholds

Input : Vector of Pstates $\mathscr{P}$, number of servers C, system capacity B, arrivals rate λ, preferred objective w.

Output: Optimal vector of thresholds $\mathbf{\Gamma}^*$ of the system

if $w = 1$ **then**
  $\mathbf{\Gamma}^* = [1, 2, \ldots, N-1]$ ▷ Lemma 4
else
  $\mathbf{\Gamma}_0^* \leftarrow [\,]$ ▷ Initialize with an empty vector
  $th_1^* \leftarrow \min_{y \in \mathbb{H}(0,\ 1)} Solve(\mathbf{\Gamma}_0^* \boxplus y,\ \mathscr{P}(1:2), w)$
  $\mathbf{\Gamma}_1^* \leftarrow \mathbf{\Gamma}_0^* \boxplus th_1^*$
  for $k \leftarrow 2$ **to** $N-1$ **do**
    $x \leftarrow \mathbf{\Gamma}_{k-1}^*(k-1)$ ▷ Last element in $\mathbf{\Gamma}_{k-1}^*$
    $th_k^* \leftarrow \min_{x \in \mathbb{V}(\mathbf{\Gamma}_{k-1}^*,\ k-1)} \left[\min_{y \in \mathbb{H}(x,\ k)} Solve(\mathbf{\Gamma}_{k-1}^* \boxplus y,\ \mathscr{P}(1:k+1), w) \right]$
    $\mathbf{\Gamma}_k^* \leftarrow \mathbf{\Gamma}_{k-1}^* \boxplus th_k^*$ ▷ Extend the optimal vector
  end
end

Pareto-Optimal approach

This approach consists in finding a set of solutions in which an objective cannot be improved without deteriorating at least one of the other objectives "Vilfredo Pareto". This set is composed of non-dominated solutions.

***Definition** 2:* A point $\Gamma \in \mathbb{D}$ dominate $\Gamma' \in \mathbb{D}$ iff :

$$(\overline{T}_{\mathbf{\Gamma}} \leq \overline{T}_{\mathbf{\Gamma'}} \wedge PW_\Gamma \leq PW_{\Gamma'}) \wedge (\overline{T}_{\mathbf{\Gamma}} < \overline{T}_{\mathbf{\Gamma'}} \vee PW_\Gamma < PW_{\Gamma'}).$$

Otherwise, Γ' is not dominated by Γ.

***Definition** 3:* Let $\mathbb{S}$ be the set of non-dominated points (i.e Pareto set). Then

$$\forall\ \Gamma \in \mathbb{S},\ \ \forall\ \Gamma' \in \mathbb{D},\ \ \Gamma \textit{ is not dominated by } \Gamma'.$$

Several algorithms exists in the literature to find the Pareto set. They are mainly grouped in two sections. Exact algorithms and the approximate (or approached) algorithms: (i) The exact algorithms consists in searching the whole Pareto set. The naive approach consists in checking the dominance property for each solution, which is highly time-consuming with $O(|\mathbb{D}|^2)$ and particularly inefficient for our case (see Equation (16)). There are also many fast and efficient methods, the most widely used one is Kung's method ([18]). This algorithm is mainly driven by sorting and dividing the population which results on a time-complexity of $O(|\mathbb{D}|log(|\mathbb{D}|))$ in the case of two objectives. Many algorithms has been proposed to improve this complexity as Ding's method ([7]) based on the scoring definition and Jun Du ([8]) based on ranking and indexing sets. (ii) For approximate approaches, genetic algorithms are the most widely used, the algorithms consists on improving the quality of a random population until approaching the real

Pareto Set. In genetic algorithms one aims to converge to the optimal front and to maintain as diverse as possible the population set. Most common ones are Deb's methods ([6]) as NSGA "Non-dominated Sorting Genetic Algorithm" and NSGA II "Elitist NSGA" ([5]). Although there exists many other genetic algorithms (VEGA, MOGA ...).

In next, we propose Algorithm 4 that merges our greedy approach with Kung's method (Algorithm 3). One should note that Kung's algorithm is indeed very fast, but it requires to have the decision space and the objective space. Yet we have $|\mathbb{D}|$ solutions where each one costs $O(B)$ to compute it's performance and power consumption. So the preliminary step of the Kung method costs $O(|\mathbb{D}|B)$. Which results with a total complexity of $O(|\mathbb{D}|log(|\mathbb{D}|)+|\mathbb{D}|B)$. Main idea of the approximate Pareto algorithm we propose, is to not explore all the decision space. We construct the Pareto set progressively from the system with two Pstates to the given system. The assumption here is quite similar to the one used in NSGA II where the goal is to emphasize non-dominated solutions from a generation to another: from non-dominated solutions of the sub-system with $N-1$ Pstates (i.e. $\mathbb{S}_{k-1}$), we create a new domain space $\mathbb{S}'_k$ that will be resolved by a Kung iteration. Based on the assumption of sub-optimal structure this algorithm provides a valuable approximation of the Pareto set in a reasonable time.

***Lemma** 6:* For an iteration k in Algorithm 4, complexity depends on the size of the non-dominated set obtained at iteration $k-1$. Hence, we have a complexity of $|\mathbb{S}_{k-1}||\mathbb{H}(x,k)|(B+log(|\mathbb{S}_{k-1}||\mathbb{H}(x,k)|)$. The worst case is when, at each iteration, the whole decision set is a non-dominated set. Complexity in this particular case tends to an exhaustive method. Inversely, with reasonable non-dominating sizes dominated solutions are disengaged earlier which makes the algorithm faster than a classical Kung's method. In best case, when non-dominated set is reduced to only one solution complexity is reduced to $O((N-1)(B^2+Blog(B))$ which is much equivalent to Algorithm 2 complexity in best case.

Algorithm 3: Kung's recursive Pareto algorithm

Input : Decision space $\mathbb{D}$
Output: Pareto set $\mathbb{S}$

Sort $\mathbb{D}$ according to descending order of the first objective. Let P be that set and let $\mathbb{S}=\emptyset$.
Front(*P*) : ▷ Recursive function
if $|P|=1$ **then**
  return P
else
  Sort P according to second objective
  s1 = **Front**($P(\ 1:\ \lfloor\ |P|/2\ \rfloor)$)
  s2 = **Front**($P(\ \lfloor\ |P|/2\ \rfloor+1:\ |P|)$)
  if *an element x in s2 is not dominated by any element in s1* **then**
    $\mathbb{S}=\mathbb{S}\cup x$
  end
  return $\mathbb{S}$
end

Algorithm 4: Approximate Pareto algorithm

Input : Vector of Pstates $\mathscr{P}$, number of servers C, system capacity B
Output: Pareto set $\mathbb{S}$

$\mathbb{S}_1 = \mathbf{Kung}(\mathbb{H}(\mathbf{0},\ \mathbf{1}))$ ▷ Kung's method for $N=2$
for $k \leftarrow 2$ **to** $N-1$ **do**
  $\mathbb{S}'_k = \emptyset$
  for $\Gamma \in \mathbb{S}_{k-1}$ **do**
    $x \leftarrow \Gamma(k-1)$
    for $th \in \mathbb{H}(x,\ k)$ **do**
      $\mathbb{S}'_k = \mathbb{S}'_k \cup (\Gamma \boxplus th)$
    end
  end
  $\mathbb{S}_k = \mathbf{Kung}(\mathbb{S}'_\mathbf{k})$
end
return $\mathbb{S}_{N-1}$

NUMERICAL RESULTS

DVFS systems are widely used in cloud computing. Especially in virtualized environments. Physical servers can host many virtual machines (VMs). Each VM can be assigned one or more virtual processor. In the following, we use our model to evaluate the trade-off between performance and energy consumption considering C VMs where each VM has 1 vCPU and a capacity queue of size B. The hypervisor of the physical server receives task requests with rate λ and assigns them to VMs on a first-come, first-served policy and a DVFS strategy across all VMs.

In Table IV we present a Pstate support inspired from AMD Opteron processor ([1]). We analyse three scenarios where we increase progressively the scalability of the problem. As we can see in the last column, decision space increases exponentially with N and B (refer to Equation (16)).

We have considered several instances for each of the three scenarios. Each instance of the problem is based on the combination of parameters $1 \leq \lambda \leq B$, $w \in \{0, 0.1, 0.2, \ldots, 1\}$ also different possibilities of Pstates. For instance, scenario A supports $N=4$ Pstates, hence we can derive the following possibilities of systems $[P_1,P_2,P_3,P_4]$, $[P_2,P_4,P_5,P_6]$ ... also, without loss of generality, we fixed: the switching UP power to Pstate i as equal to the active power $s_i = p_i$, $\alpha = 0.25$ (i.e. a power gain of 75% in idle state), number of cycles per instruction to $CPI=1$ and number of instructions per tasks to $b=10^9$ so that mean response time is expressed in seconds.

TABLE IV: Pstates support for a system supporting 6 Pstates and parameters of three different scenarios

Pstates	P_1	P_2	P_3	P_4	P_5	P_6
Frequency (GHz)	1	1.8	2	2.2	2.4	2.6
Power (W)	32	55	65	76	90	95

Parameters	B	C	N	$\lvert\mathbb{D}\rvert$
Scenario A	45	20	4	13244
Scenario B	80	30	4	79079
Scenario C	100	40	5	3764376

Comparison of algorithms

For weighted sum methods, we compare the proposed greedy approximation (Algorithm 2) with a local search method and a Tabu search, and also an exhaustive method to evaluate the success rate. The local search and Tabu search are classical neighborhood methods. The local search starts with a random seed solution and improves this latter locally until there is no more possible improvement in the neighborhood. However, it can get stuck in a local optimum. To avoid this behavior, we have opted for the Tabu search method; a method that gets out of a local optimum. In the Tabu method, we store all previous local optimums in a Tabu list of reasonable size (that we fixed to $4BN$). The Tabu list is updated progressively and keeps the most recent local optimums as long as possible in order to avoid cycling phenomena. The algorithm stops when no improvement occurs during several iterations. Note that, to compare results, we fixed the same randomized seed for both search methods.

Contrary to WSM methods, in Pareto methods, we need to compare two sets of vectors instead of two vectors. Therefore, we use Jaccard similarity index between the exact and the approximated Pareto front. The proposed success rate is the mean Jaccard index for all investigated instances. We also show $|\mathbb{S}_{avg}|$ the average size of Pareto set for all instances. Here, we no longer need preferency parameter w, so we have less instances to compare, but more computing time for each instance (refer to algorithms complexity in previous section). Due to the scalability of scenarios B and C, in WSM Methods we perform one simulation only for the exhaustive search to keep track of the computation time. Thus, we use the %Min rate instead of %Success rate. %Min rate is the percentage of optimality of each, non-exhaustive, method among the others. The simulations have been performed on a laptop with 10 cores (8 of them at 3.2 GHz peak frequency and two others at 2 GHz peak frequency) with 16GB RAM.

TABLE V: Scenario A, Comparison of 2024 instances

WSM Methods	**Success rate**	**Time (s)**
Exhaustive search	100%	54.18
Greedy approx ($\Delta = 0$)	96%	0.49
Greedy approx ($\Delta = \Delta_{Max}$)	99%	7.50
Local search	59%	0.81
Tabu search	71%	3.16
Kung Pareto Method	**Pareto set size**	**Time (s)**
Light load	$\|\mathbb{S}_{avg}\| = 9515$	59.24
Moderate load	$\|\mathbb{S}_{avg}\| = 3454$	56.10
Heavy load	$\|\mathbb{S}_{avg}\| = 475$	55.24
Approximate Pareto	**Success rate**	**Time (s)**
Light load	100%	49.23
Moderate load	100%	17.38
Heavy load	100%	3.24

The numerical results (Table V, VI, VII) show that:

- The proposed greedy approaches are the most efficient methods among those studied, notably on the quality of the solutions and the execution time. Whether in scalarization method or in Pareto method. In many cases it is not required to add an exploratory step ($\Delta = 0$) to improve the quality. Also, as observed in Scenario B and C, greedy methods are scalable. WSM greedy approach solves a system of 40 VMs and $3.7x10^6$ feasible solution in approximately 3 seconds with an optimality rate of 95%.
- The classical methods of exploration, in particular the local search or the meta-heuristic Tabu search, show weaker results as they strongly depend on the shape of the scalarized function, particularly in our model where rewards formulas are complex and decision space is factorial.
- Pareto methods require more computation time. But they are much more valuable to provide a panel of disparate compromise solutions instead of a single one (see Fig. 1 and Fig. 2). The proposed Approximate Pareto method is efficient for moderate and heavy load traffic. It can reach 100% (resp. 99%) of the Pareto front in 3 (resp. 54) seconds instead of 55 (resp. 601) seconds for scenario B (resp. scenario C).

TABLE VI: Scenario B, Comparison of 1540 instances

WSM Methods	**%Min rate**	**Time (s)**
Exhaustive search	100%	604.16
Greedy approx ($\Delta = 0$)	96%	1.88
Greedy approx ($\Delta = \Delta_{Max}$)	99%	48.02
Local search	65%	2.20
Tabu search	73%	7.50
Kung Pareto Method	**Pareto set size**	**Time (s)**
Light load	$\|\mathbb{S}_{avg}\| = 72863$	870.02
Moderate load	$\|\mathbb{S}_{avg}\| = 33846$	653.11
Heavy load	$\|\mathbb{S}_{avg}\| = 5368$	601.51
Approximate Pareto	**Success rate**	**Time (s)**
Light load	100%	845.20
Moderate load	100%	339.99
Heavy load	99%	54.40

TABLE VII: Scenario C, Comparison of 1012 instances.

WSM Methods	**%Min rate**	**Time (s)**
Exhaustive search	100%	33848
Greedy approx ($\Delta = 0$)	95%	3.32
Greedy approx ($\Delta = \Delta_{Max}$)	98%	122.52
Local search	53%	4.12
Tabu search	61%	28.51
Approximate Pareto	**Pareto size**	**Time (s)**
Moderate load	$\|\mathbb{S}_{avg}\| = 89661$	1756.43
Heavy load	$\|\mathbb{S}_{avg}\| = 29543$	677.39

System dynamic analysis

To analyze the system dynamics, we consider scenario A. In a first step (Fig. 1), we investigate the influence of the system workload on the response time and the power. Then (Fig. 2), we fix the load of the system and then extend the number of Pstates (from $N = 3$ to $N = 5$) for a case with negligible switching UP power of Pstates and a case considering switching UP power. We notice that : a) The higher the system is loaded, the more likely it is to reach the highest Pstates and thus the objective space becomes more disparate unlike unloaded systems where not all Pstates are explored. b) In order to minimize the response time, the system tends to move

towards high Pstates (i.e. with small thresholds) which would consume more, however to reach them the different switching phases must be handled (arcs patterns in the graphics). The system also has to avoid successive switch UP (and switch DOWN) effects. This makes the optimal choice more difficult. c) The model with negligible switch UP power, generates a much simpler behavior as it is not subject to switching power overheads or successive UP/DOWN effects. d) The exact (resp. Approx) Pareto method i.e. red points (resp. blue points) allows to clearly distinguish the equilibrium solutions between response time and power consumption. The Greedy method also approximately reaches the Pareto zone, however the distribution of points (yellow ones) remains a matter of discussion as some Pareto points are difficult to reach. Finally, we show the trajectory of the Tabu search promoting the response time (case of $w = 0.8$) and power consumption (case of $w = 0.05$), where we observe the convergence to different points of the Pareto front. For instance, in Fig. 2 with $N = 5$, weights $w \in \{0, 0.01, 0.02\}$, $w \in \{0.03, 0.04\}$, $w \in \{0.05, 0.08\}$, $w = 0.09$ and $w \geq 0.1$ provides, respectively, optimal thresholds vector $[41,42,43,44]$, $[1,42,43,44]$, $[1,2,43,44]$, $[1,2,3,44]$ and $[1,2,3,4]$ that is to remain for a long time in a single Pstate 2, 3, 4, 5 and 6, thus avoiding switching UP overheads. While in the same model with negligible switching costs (the case of [1] Opteron processor), system changes Pstates more frequently: for $w \in \{0, 0.01, 0.02\}$, $w = 0.03$, $w = 0.04$, $w = 0.05$ and $w = 0.06$ optimal thresholds are, respectively, $[41,42,43,44]$, $[40,42,43,44]$, $[32,37,43,44]$, $[27,31,34,35]$, $[24,27,29,30]$ until reaching $[1,2,3,4]$ with $w = 1$. Hence, switching costs s_i has an important impact on the system dynamism and must be chosen appropriately in order to benefit from the various Pstate levels avoiding the successive switch UP/DOWN changes.

CONCLUSION

In this paper, we proposed a new model to analyse DVFS systems with $N \geq 1$ Pstates. We propose a "st" comparison of system's performance measures and a closed formula to compute rapidly rewards of the system thus to proceed with a MOO analysis. In the optimization phase we considered the two main approaches in MOO. The scalarization of objectives and the Pareto front method. By analyzing the results, we observe that a greedy heuristic approach generates accurate results in a very convenient time. However, for WSM methods, preferency parameter and distribution of optimal point in objective space remain an issue. This will lead us to analyze Kung's approach, which is the most commonly used exact Pareto front method. This latter method suffers from the scalability of decision and objective space, by adjusting it with our greedy approach we can get fast results of a non-dominated set that is remarkably close to Pareto front, especially when dealing with systems with moderate and heavy load. Finally we compare two DVFS systems: one with switching UP power of the Pstates, and one with absence of switching power. We naturally observe an increase of the average consumption in the system with switching power. But more specifically, a disparate and complex decision space is formed where the system does not necessarily consume the most in the highest Pstates that we can observe in the model without switching power, thus making the optimal decision further challenging. Several perspectives can emerge from this work: adding a non-negligible delay for the Pstate change, considering a network of DVFS systems. This will lead to investigate further fast optimization methods or reinforcement learning methods to analyse the performance and power consumption trade-off.

SOURCE CODE

The proposed DVFS analysis is available in GitHub framework.

REFERENCES

[1] Advanced Micro Devices AMD. Power and cooling in the data center - addressing today's and tomorrow's challenges with the amd opteron tm processor and amd powernow tm technology with optimized power management (opm). Technical report, 3 2005.

[2] Jonatha Anselmi, Bruno Gaujal, and Louis-Sébastien Rebuffi. Optimal speed profile of a dvfs processor under soft deadlines. *Performance Evaluation*, 152:102245, 2021.

[3] Robert Basmadjian, Florian Niedermeier, and Hermann de Meer. Modelling performance and power consumption of utilisation-based dvfs using m/m/1 queues. In *Proceedings of the Seventh International Conference on Future Energy Systems*, e-Energy '16. Association for Computing Machinery, 2016.

[4] Waltenegus Dargie. A stochastic model for estimating the power consumption of a processor. *IEEE Transactions on Computers*, 64(5):1311–1322, 2015.

[5] Kalyanmoy Deb and Tushar Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, page 67–81. Springer-Verlag, 2001.

[6] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, page 42–50. Morgan Kaufmann Publishers Inc., 1989.

[7] Lixin Ding, Sanyou Zeng, and Lishan Kang. A fast algorithm on finding the non-dominated set in multi-objective optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 4, pages 2565–2571 Vol.4, 2003.

[8] Jun Du, Zhihua Cai, and Yunliang Chen. A sorting based algorithm for finding a non-dominated set in multi-objective optimization. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, pages 436–440, 2007.

[9] Peter C. Fishburn. Letter to the editor—additive utilities with incomplete product sets: Application to priorities and assignments. *Operations Research*, 15(3):537–542, 1967.

[10] Jean Michel Fourneau, Youssef Ait El Mahjoub, Franck Quessette, and Dimitris Vekris. Xborne 2016: A brief introduction. In Erol Gelenbe, Tadeusz Czachórski, Krzysztof Grochla, and Ricardo Lent, editors, *Computer and Information Sciences*, 2016.

[11] Anshul Gandhi and Mor Harchol-Balter. How data center size impacts the effectiveness of dynamic power management. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1164–1169, 2011.

[12] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. Optimal power allocation in server farms. *SIGMETRICS Perform. Eval. Rev.*, 37(1):157–168, 2009.

[13] Odu Godwin. Review of multi-criteria optimization methods – theory and applications. *IOSR Journal of Engineering*, 3:01–14, 10 2013.

[14] L. Golubchik and J.C.S. Lui. Bounding of performance measures for threshold-based queuing systems: theory and application to dynamic resource management in video-on-demand servers. *IEEE Transactions on Computers*, 51(4):353–372, 2002.

[15] Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018.

[16] Hua He, Yu Zhao, and Shanchen Pang. Stochastic modeling and performance analysis of energy-aware cloud data center based on dynamic scalable stochastic petri net. *COMPUTING AND INFORMATICS*, 39(1-2):28–50, 2020.

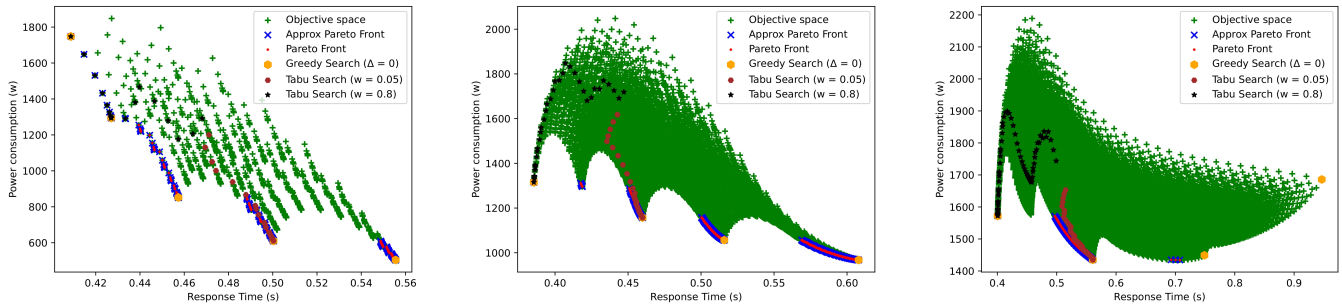

Fig. 1: Response time Vs Power consumption for a $N = 5$ Pstates DVFS system $[P_2, P_3, P_4, P_5, P_6]$) with: Light ($\lambda = 10$ in left figure), Moderate ($\lambda = 30$ in middle figure) and Heavy load ($\lambda = 40$ right figure). Other parameters come from scenario A

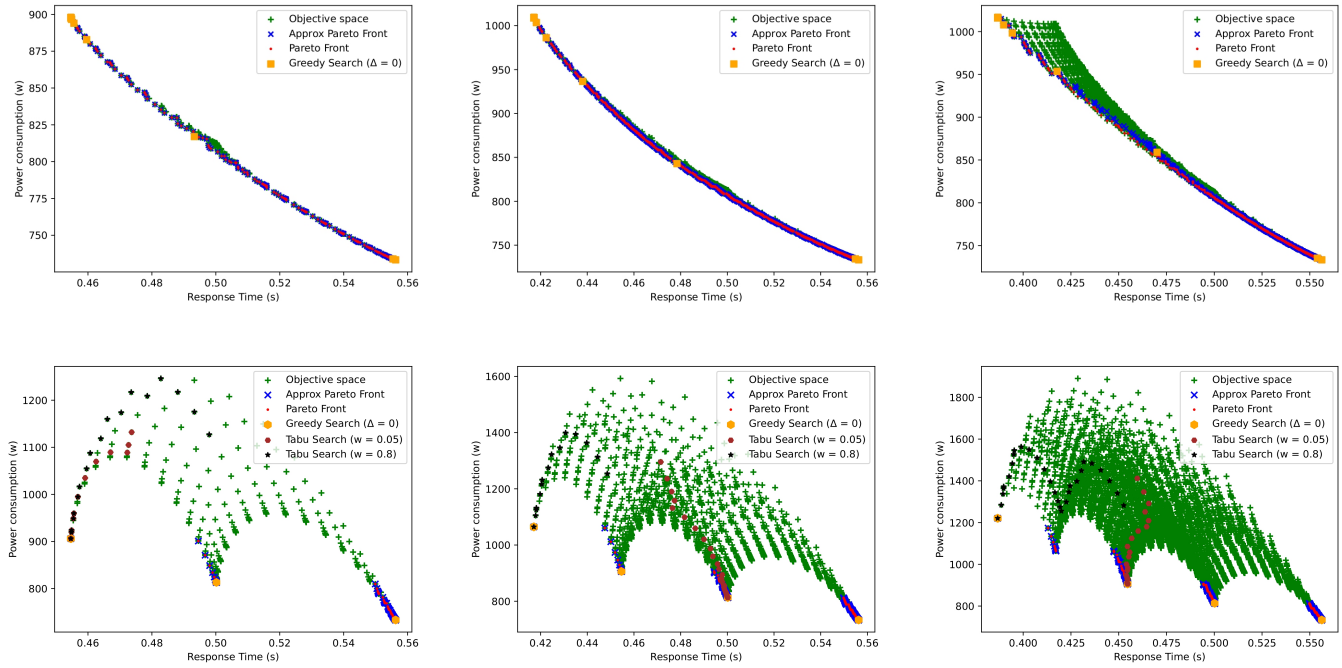

Fig. 2: Response time Vs Power consumption for three DVFS systems $[P_2, P_3, P_4]$, $[P_2, P_3, P_4, P_5]$, $[P_2, P_3, P_4, P_5, P_6]$ in moderate mode $\lambda = 20$. The first (resp. second) row represents models with negligible (resp. non-negligible) switch UP power

[17] Paul J. Kühn and Maggie Ezzat Mashaly. Dvfs-power management and performance engineering of data center server clusters. *2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 91–98, 2019.

[18] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22:469–476, 1975.

[19] F. V. Lu and R. F. Serfozo. M/m/1 queueing decision processes with monotone hysteretic optimal policies. *Operations Research*, 32(5):1116–1132, 1984.

[20] John C.S. Lui and Leana Golubchik. Stochastic complement analysis of multi-server threshold queues with hysteresis. *Performance Evaluation*, 35(1):19–48, 1999.

[21] Isi Mitrani. Service center trade-offs between customer impatience and power consumption. *Performance Evaluation*, 68(11):1222–1231, 2011.

[22] Quang Nguyen, Shervin Hajiamini, and Behrooz Shirazi. An improved stochastic-based approach for optimizing energy-time tradeoff in multicore systems. In *2020 11th International Green and Sustainable Computing Workshops (IGSC)*, pages 1–8, 2020.

[23] Altino M. Sampaio and Jorge G. Barbosa. Chapter three - energy-efficient and sla-based resource management in cloud data centers. In *Energy Efficiency in Data Centers and Clouds*, volume 100, pages 103–159. Elsevier, 2016.

[24] D. Stoyan. *Comparaison Methods for Queues and Other Stochastic Models*. John Wiley and Sons, Berlin, 1983.

[25] Donald S. Szarkowicz and Thomas W. Knowles. Optimal control of an m/m/s queueing system. *Operations Research*, 33:644–660, 1985.

[26] Tournaire. Thomas, Castel-Taleb. Hind, and Hyon. Emmanuel. Optimal control policies for resource allocation in the cloud: comparison between markov decision process and heuristic approaches. *arXiv:2104.14879*, 2021.

[27] Ait El Mahjoub. Youssef, Fourneau. Jean-Michel, and Castel-Taleb. Hind. Performance evaluation and energy consumption for dvfs processor. *26th International Conference on Analytical & Stochastic Modelling Techniques & Applications, ASMTA*, 2021.