# Stochastic analysis of energy consumption in pool depletion systems

D. Cerotti, M. Gribaudo, R. Pinciroli, G. Serazzi

Dip. di Elettronica, Informazione e Bioingegneria, Politecnico di Milano,
via Ponzio 34/5, 20133 Milano, Italy,
[davide.cerotti,marco.gribaudo,riccardo.pinciroli,giuseppe.serazzi]@polimi.it

**Abstract.** The evolutions of digital technologies and software applications have introduced a new computational paradigm that involves initially the creation of a large pool of jobs followed by a phase in which all the jobs are executed in systems with limited capacity. For example, a number of libraries have started digitizing their old books, or video content providers, such as YouTube or Netflix, need to transcode their contents to improve playback performances. Such applications are characterized by a huge number of jobs with different requests of computational resources, like CPU and GPU. Due to the very long computation time required by the execution of all the jobs, strategies to reduce the total energy consumption are very important.
In this work we present an analytical study of such systems, referred to as *pool depletion systems*, aimed at showing that very simple configuration parameters may have a non-trivial impact on the performance and especially on the energy consumption. We apply results from queueing theory coupled with the absorption time analysis for the depletion phase. We show that different optimal settings can be found depending on the considered metric.

**Key words:** stochastic models, energy efficiency, performance evaluation

## 1 Introduction

In this paper we focus on systems in which there is a fixed and huge number of jobs, referred to as a *pool*, waiting to be admitted for execution in a set of service centers with limited capacity. Many current real life problems require models with this structure. For example, video content providers, such as YouTube or NetFlix, often need to transcode a huge pool of videos [6] to multiple formats suitable to be sent and playback by several different devices (e.g. smart-phone, smart-TV, tablet, ...). Similarly, several big data applications generate during the map phase a huge pool of data that can subsequently be split and executed in parallel on different systems with limited capacity for performance reasons.
The behavior of this system can be regarded as divided into two phases. An initial phase, in which the system is loaded with the maximum number of jobs

allowed, and every job completed is immediately replaced by another one admitted from the pool. Then, when the pool empties a new phase starts, referred to as *depletion* phase, and the number of jobs in execution continues to decrease until all jobs are completed.
Since the jobs may have very different resource demands, in our analysis we consider *multi-class* workloads. To study the behavior of this type of systems, that are not at the equilibrium, we applied the stochastic analysis implementing the CTMC of the different cases.
The problem approached is: given a workload, i.e., pool size and characteristics of the classes of jobs (service demands and fractions in execution), and a system with finite capacity study a scheduling admission policy so that the global amount of time to execute the complete pool of jobs, i.e., the duration of the full capacity phase plus the depletion phase, is minimized.
To reach this objective, the load admission policy that schedule the sequence of executions must be able to fully exploit the capacity of *all* the resources of the system. In other words, the saturation of the resources must be reduced as much as possible controlling the bottlenecks. Let us remark that minimizing the time required from the execution of the complete workload is equivalent to minimize the energy required for this task. Thus, we may say that our ultimate objective is to minimize the energy needed to execute a workload through a suitable admission policy based on the bottleneck control.
We adopt known results of queuing networks for the full capacity phase, and the absorption time analysis for the depletion phase. By queuing theory, it is known that the performance of systems with multi-class workloads depends on the fraction of the classes of jobs in execution (referred to as *population mix*). More precisely, given the service demands of the classes, it is possible to identify a set of population mixes that saturate more than one resource concurrently. Moreover, one of these population mixes allow resources to be equiutilized regardless the population sizes. This operational condition is optimal since it maximizes the utilization of all the resources and thus the system throughput [16].
The main objectives are to study the impact of both the size of the job pool and the maximum processing capacity, in term of maximum number of jobs in execution, to the depletion time. Moreover, in multi-class workload, we want to examine the effect of the population mix to optimize depletion time, energy consumption, and response time in order to identify an optimal trade-off between them.
The remainder of the paper is structured as follows. In Section 2 we review some metrics used for energy consumption measurement. Section 3 presents in detail the pool depletion models both with single and multi-class workload and the Markov Chains utilized. In Section 4 we investigate the behavior of the model and show that the energy consumption is minimized when the system works with an optimal population mix. Section 5 concludes the paper and presents some future directions of work.

## 2 Energy consumption

Several works, e.g. [7], [15], show the existence of a linear relationship between the power consumption of a server and the utilization of its CPU. For such reason, a widespread used approximation of the power consumption $P(U)$ of a server is given by:

$$P(U) = P_{idle} + U\ (P_{max} - P_{idle}) \tag{1}$$

where $P_{idle}$ is the power consumed when no user applications are running, $P_{max}$ is the power drawn by the fully utilized server and $U$ is the CPU utilization. Many improvements of this model have been proposed to take into account other devices likes memory [12] or disks [5], and consider non-linearity measured in some real applications.

Since the global energy consumed by a task of duration $T$ can be computed as $E = P \cdot T$, there is a trade-off between two factors. On one hand, Eq. 1 suggests to reduce the utilization to decrease the power consumption; on the other, a low utilization yields a low server productivity, increasing the time $T$ required to complete the given task and thus also the energy consumed. In addition, there is a related trade-off between the energy consumed and the performance provided by the system. Energy-Response time Product (ERP), also known as Energy-Delay Product (EDP), and Energy-Response time Weighted Sum (ERWS) are two metrics widely used to evaluate the performance-energy trade-off of a system. Both of depend on the total energy consumption ($E$) and the response time ($R$). The index ERP [8, 9, 11, 13] is defined as their product:

$$ERP = E\ R, \tag{2}$$

whereas ERWS [1–3, 10] is defined as their weighted sum:

$$ERWS = w_1\ R + w_2\ E, \quad w_1, w_2 \geq 0. \tag{3}$$

The *average energy consumed per job* $EJ$ is a further metric to compute such trade-off [4]. It is defined as:

$$EJ = \frac{E}{C} = \frac{P \cdot T}{C} = \frac{P}{X}, \tag{4}$$

where $C$ is number of jobs completed during a time interval of length $T$ and $X$ is the *system throughput*. Equation 4 holds for a resource processing a single-class workload, but it is not fair with a workload composed by jobs of different classes, especially when the time required to complete a job varies significantly according to its class. To overcome such problem, exploiting *the utilization law* a multi-class extension of Eq. 4 has been proposed as:

$$EJ = D\ \frac{P}{U}, \tag{5}$$

where $D$ is the aggregate demand (i.e. the total service demand of all the classes) and $U$ is the resource utilization. The details of the Eq. 5 derivation and its extension to take into account systems composed of several resources can be found in [4].

## 3 Model description

Let us consider the depletion model of a system composed by two resources $r_1$ and $r_2$ as shown in Figure 1. Resources can represent important parts of a computing architecture: in the following we will use one resource to model a single-core CPU, and the other to model a GPU. The system executes two classes of jobs $A$ and $B$. Each class requires an exponentially random distributed amount of execution time at each resource, and it is characterized by its average $D_{rc}$, with $r \in \{1, 2\}$ and $c \in \{A, B\}$. The two resources satisfy the classical BCMP assumptions: they either work in processor sharing, or in first-come-first-served with all the requests of identical size, but possibly with different visit ratio. The total number of jobs that must be executed in the two classes are respectively $N_A$ and $N_B$. However, only $K = k_A + k_B$ jobs are executed in parallel, with $k_A$ jobs of class $A$ and $k_B$ jobs of class $B$. We call this constraint as *Finite Capacity Region* (FCR).Whenever a class $A$ job completes and leaves the system, another class $A$ jobs is started. If all the class $A$ jobs are finished, but there are still class $B$ jobs to be executed, class $B$ jobs enter the system in place of class $A$ jobs, to maintain its workload to $K$ jobs. If there are no more jobs waiting to be executed, as soon as a job finishes, it is not replaced by other activity until all the $N_A + N_B$ jobs have been completed. Class $B$ jobs behaves in the symmetrical way.

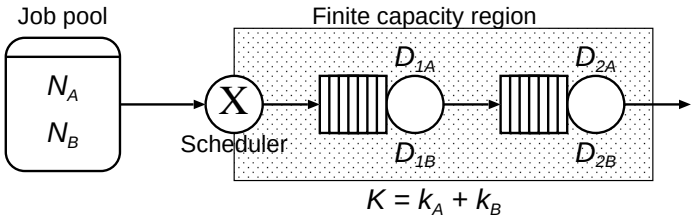



Fig. 1: A pool depletion model with two class and two resources.

Figure 2 shows the temporal evolution of the system. First jobs are loaded from the pool into the first resource, until the size of the FCR is reached (*Phase 0*). To simplify the presentation, we will consider the duration of this phase to be negligible, and we consider the system starting from a state in which there are $k_A$ class $A$ jobs and $k_B$ class $B$ jobs in execution in the first resource $r_1$. During normal execution, as soon as one job finishes, another one of the same class immediately starts (*Phase I*): this is the time in which the system works at regime, and it is also the moment in which optimization can take place. As soon as the jobs of one class in the pool finishes, the system moves to *Phase II*, where the scheduler cannot really perform a decision since it can only start jobs of the remaining class to fill the number of tasks in concurrent execution.

Finally, when there are no more new jobs that can be started, the *depletion* phase begins (*Phase III*).In this case the number of jobs in execution reduces progressively until all the tasks have been completed. Note that both in *Phase II* and *Phase III*, one class of jobs might finish much earlier than the other, reducing the system to a single class behavior.

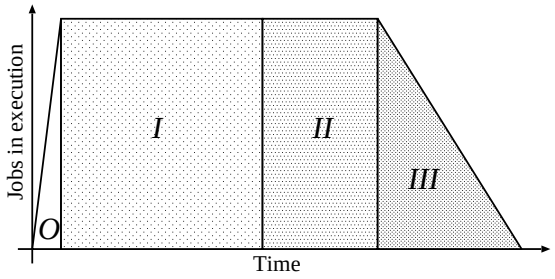



Fig. 2: Temporal evolution of the system.

For each model we construct the corresponding underlying CTMC. Even if the proposed model seems to be very simple, the underlying Markov process is characterized by lot of asymmetries that makes its description a bit involved. To simplify the presentation, we start by presenting a simple single-class example with fixed parameters, and then we extend it to the two-classes general case.

### 3.1 Single-class model

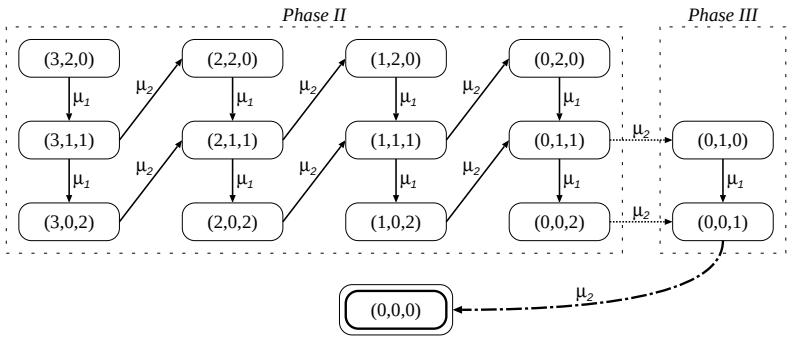



Fig. 3: The CTMC corresponding to a single class system with $N_A = 5$ and $k_A = 2$.

Let us consider a single-class model with $N_A = 5$ jobs to be completed, in which $K = k_A = 2$ jobs at a time are executed in parallel by the system. The corresponding CTMC is shown in Figure 3, and its state is identified by a tuple: $(n_{OA}, n_{1A}, n_{2A})$, where $n_{OA}$ is the number of jobs that are still waiting to be started, $n_{1A}$ is the number of jobs in resource $r_1$ and $n_{2A}$ is the number of jobs in resource $r_2$.
Since we ignore the loading phase, all jobs that can be immediately executed starts in resource $r_1$. For this reason the initial state of the CTMC is $(n_{OA} - n_{1A}, n_{1A}, 0) = (3, 2, 0)$. Let us call $\mu_1 = 1/D_{1A}$ the rate at which jobs leaves resource $r_1$, and $\mu_2 = 1/D_{2A}$ the rates at which jobs complete their execution. Jobs always leave from $r_1$ to $r_2$ at rate $\mu_1$, producing a transition from state $(n_{OA}, n_{1A}, n_{2A})$ to state $(n_{OA}, n_{1A} - 1, n_{2A} + 1)$. The effect of the end of service at resource $r_2$ is different depending on whether there are jobs waiting to start ($n_{OA} > 0$ - *Phase II* in Figure 2). If this is the case, the system performs a transition from state $(n_{OA}, n_{1A}, n_{2A})$ to state $(n_{OA} - 1, n_{1A} + 1, n_{2A} - 1)$ at rate $\mu_2$ corresponding to the fact that whenever a job exits the system from resource $r_2$, one of the waiting job is immediately started at $r_1$. If instead the jobs waiting to be started are finished ($n_{OA} = 0$ - *Phase III* in Figure 2), the system starts working on one less job performing a transition from state $(0, n_{1A}, n_{2A})$ to state $(0, n_{1A}, n_{2A} - 1)$, always at rate $\mu_2$. When the last job ends, the system jumps in the absorbing state $(0, 0, 0)$.

### 3.2 Multi-class model

Figure 4 shows the basic transition structure of the CTMC underlying a two-class model. To simplify the presentation, only outgoing arcs are shown. In the two class case, the state is characterized by a six components tuple:

$$(n_{OA}, n_{OB}, n_{1A}, n_{1B}, n_{2A}, n_{2B})$$

which contains the count of jobs waiting outside, being executed at $r_1$ or at $r_2$ for both classes. If $n_{1A} + n_{1B} > 0$, jobs can complete their service at resource $r_1$. In this case we can have a transition either to state $(n_{OA}, n_{OB}, n_{1A} - 1, n_{1B}, n_{2A} + 1, n_{2B})$ or to state $(n_{OA}, n_{OB}, n_{1A}, n_{1B} - 1, n_{2A}, n_{2B} + 1)$ at rate $\mu_{1c}$ (with $c \in \{A, B\}$):

$$\mu_{1c} = \frac{n_{1c}}{n_{1A} + n_{1B}} \frac{1}{D_{1c}}. \tag{6}$$

The first part of the equation represents the processor sharing policy used by the resource. The end of service of a job at resource $r_2$ can instead trigger four different types of behaviors, each leading to a different pattern for the next state. Let us focus on a class $A$ job: the case for class $B$ will be symmetrical.
If the there are still class $A$ jobs waiting to be started ($n_{OA} > 0$ - *Phase I* in Figure 2), the system will allow a new class $A$ job to start its execution. This leads the system to state $(n_{OA} - 1, n_{OB}, n_{1A} + 1, n_{1B}, n_{2A} - 1, n_{2B})$ and it is represented in the figure by arrows drawn with a continuous line.
If there are no more class $A$ jobs waiting to be started ($n_{OA} = 0$) but still class

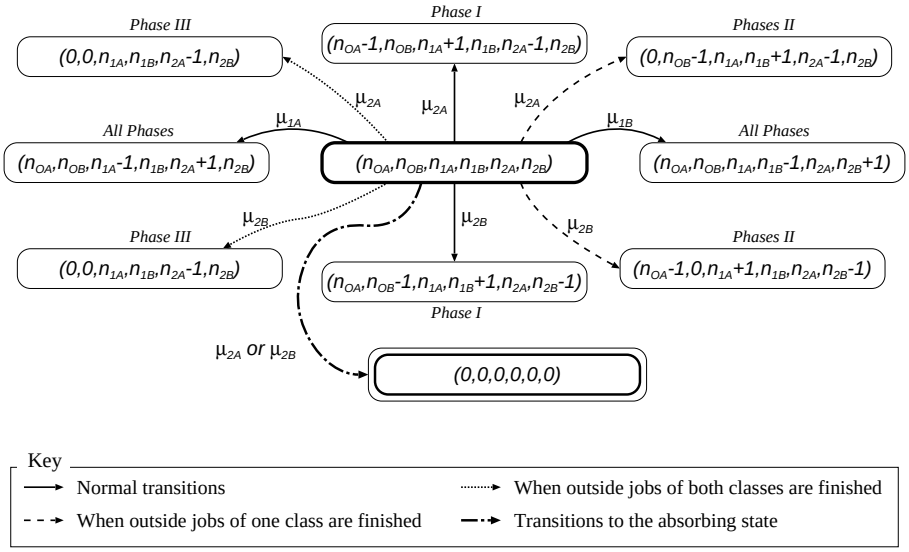



Fig. 4: Portion of the CTMC corresponding to a two class system.

$B$ jobs ($n_{OB} > 0$ - *Phase II* in Figure 2), then the end of a class $A$ job triggers the start of a class $B$ job to exploit the maximum parallel running capacity $K$ of the system. This is represented in Figure 4 as a dashed arrow, and leads the system to state $(0, n_{OB} - 1, n_{1A}, n_{1B} + 1, n_{2A} - 1, n_{2B})$.
If there are no more jobs to be started of either classes ($n_{OA} = 0$ and $n_{OB} = 0$ - *Phase III* in Figure 2), then the system starts working with less than $K$ jobs in parallel, by jumping to state $(0, 0, n_{1A}, n_{1B}, n_{2A} - 1, n_{2B})$. This is the depletion phase, which is represented in the figure by a dotted line.
Finally, when the last job ends the system jumps to the absorbing state $(0, 0, 0, 0, 0, 0)$. This is represented with a dash-dotted line in the figure. Again, due to the processor sharing nature of the system, ending of jobs at resource $r_2$ occurs at rate $\mu_{2c}$ (with $c \in \{A, B\}$):

$$\mu_{2c} = \frac{n_{2c}}{n_{2A} + n_{2B}} \frac{1}{D_{2c}}. \tag{7}$$

### 3.3 Model Analysis

In order to compute the depletion time, we apply the well-known technique for the evaluation of the upto-absorption time. Let us consider the CTMC of the general model with absorbing state (0,0,0,0,0,0) and infinitesimal generator matrix $\mathbf{Q} = [q_{ij}]$ and let us call $B$ the set of non-absorbing states. We define the mean time spent by the CTMC in state $i$ until absorption as $z_i = \int_0^\infty \pi_i(\tau) d\tau$, where $\pi_i(\tau)$ is the unconditional probability of the CTMC being in state $i$ at

time $\tau$. The row vector $\mathbf{z} = [z_i]$ satisfies the following equation:

$$\mathbf{z}\,\mathbf{Q}_B = -\pi_B(0), \tag{8}$$

where $\pi_B$ and $\mathbf{Q}_B$ are the transient probability vector and the infinitesimal generator matrix restricted to the non-absorbing states only. Following [14], the mean time to absorption $T$ of the CTMC can be computed from the solution of Eq. 8:

$$T = \sum_{i \in B} z_i \ .$$

If we call $P_i$ the average power consumed in state $i$, then the average total energy consumed by the system is:

$$E = \sum_{i \in B} z_i \cdot P_i \ .$$

In a similar way, if we call $u_{ri}$ an indicator function that tells us if a resource $r$ is used in state $i$, $\phi_i(X)$ an indicator function that tells us if state $i$ belongs to phase $X \in \{I, II, III\}$, and $m_i$ the number of jobs in the FCR in state $i$, then we can compute the average utilization $U_r$ of resource $r$, the average time $\Phi(X)$ spent in phase $X$, and the average number of jobs in the FCR as:

$$U_r = \frac{1}{T} \sum_{i \in B} z_i \cdot u_{ri} \qquad \Phi(X) = \sum_{i \in B} z_i \cdot \phi_i(X) \qquad M = \frac{1}{T} \sum_{i \in B} z_i \cdot m_i \ . \tag{9}$$

## 4 Results

We have implemented the model described in Section 3 and run several analytical experiments both with single-class and multi-class workloads. Models are analyzed by generating their underlying CTMC and solving it according to Section 3.3 using a linear algebra library implemented in C language. Performance indices can be computed in few minutes on a standard Linux laptop even for the cases with the largest state space. In particular, the size of the state space can vary from 201 states when we work with a single-class model and $K = 1$, to 470 771 states when we are considering the multi-class model with $K = 40$.

### 4.1 Single-class model

In the first set of experiments, we analyze the pool depletion system working with single-class workloads. In particular, we want to characterize the behavior of the model as a function of the number of jobs simultaneously admitted into the FCR.

The total number of jobs in the system is $N_A + N_B = 100$, and the service demands $D_r$ used in the experiments for the two resources are given in Table 1. The number of jobs that can enter the FCR at the same time varies from $K = 1$

Table 1: Service demands used for the single-class model.

| | Conf. 1 | Conf. 2 | Conf. 3 | Conf. 4 |
|---|---|---|---|---|
| $D_1$ | 0.75 | 0.64 | 1.95 | 1.2 |
| $D_2$ | 0.48 | 1.25 | 0.6 | 1.6 |

to $K = 100$. In case of $K = 1$, only one job can be processed at once. When $K = 100$, all the jobs that are in the system can enter the FCR and they are concurrently serviced with a processor sharing policy.

Fig. 5 shows the performance indexes of the pool depletion systems with single-class workloads as a function of the FCR size $K$. In order to emphasize the results for small values of $K$, a base-10 log scale is used on the $x$-axis.

The energy consumption is computed setting the idle power consumption of system $P_{idle} = 70\ Watt$, the maximum power of the system when only resource $r_1$ is used $P_{busy1} = 160\ Watt$, the maximum power when only resource $r_2$ is used $P_{busy2} = 130\ Watt$ and the maximum power of the system when both resources are used $P_{busy} = 210\ Watt$. As shown in Fig. 5a, larger values of $K$ reduce the energy consumption, since they reduce the total completion time.

Fig. 5b shows the average response time to complete a job: the average time a job is running. This index does not account for the time spent outside the FCR, and it is computed using Little's law as:

$$R = \frac{M}{X} = \frac{M}{(N_A + N_B)/T}$$

where $M$ is the average number of jobs in the FCR defined in Eq. 9. As it can be seen, $R$ increases with $K$ since resources are shared by a larger number of jobs. ERP and ERWS are plotted in Fig. 5c and Fig. 5d, respectively. For ERWS, we define $w_1$ and $w_2$ in order to normalize the values of response time and energy consumption. Thus, for each configuration, $w_1$ is set to $1/max(R_k)\ \forall k$ and $w_2$ is set to $1/max(E_k)\ \forall k$, where $k$ is the number of considered jobs into the FCR.

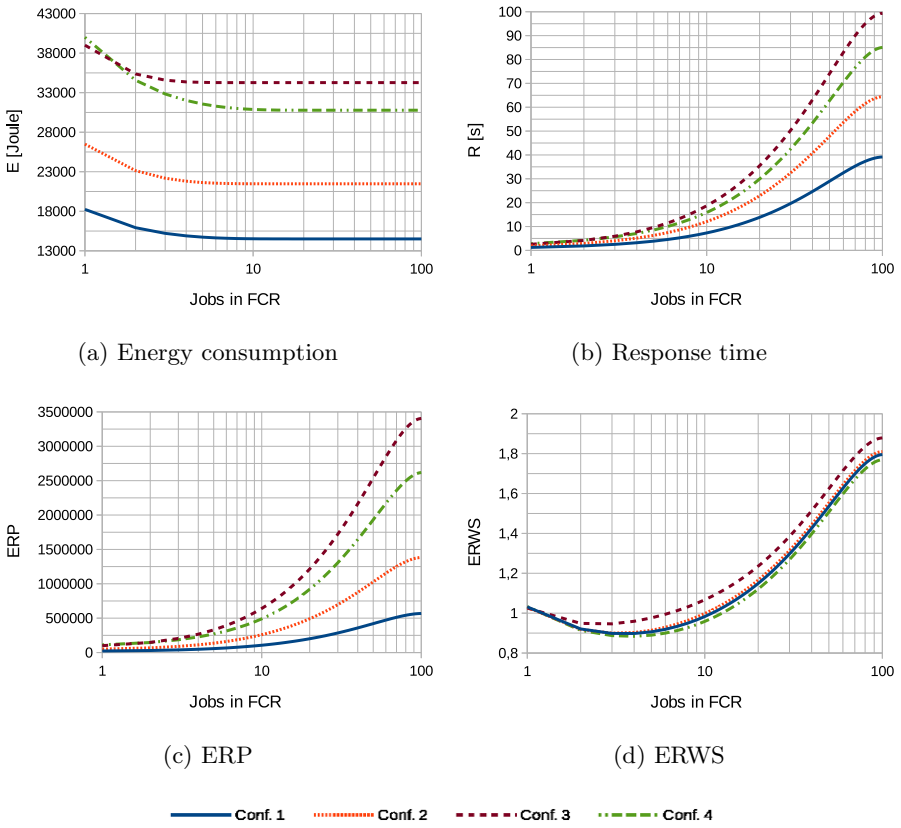
ERP identifies Conf. 1 as the best configuration and the optimal point is when only a job is in the FCR. Instead, for ERWS the best configuration is Conf. 4 and the minimum coincides with four jobs concurrently executed by the system[1].

### 4.2 Multi-class model

Next, we consider a multi-class model where the total number of jobs in the system is $N_A + N_B = 80$ and the number of jobs admitted in the FCR $K = 20$. We consider the following service demands:

$$\begin{array}{ll} D_{1A} = 0.26 & D_{1B} = 0.01 \\ D_{2A} = 0.08 & D_{2B} = 0.19 \end{array}.$$

[1] Providing evidence about which is the best metric between ERP and ERWS is out of the purposes of this paper.




(a) Energy consumption

(b) Response time

(c) ERP

(d) ERWS

Fig. 5: Performance indexes for the single-class system.

In Fig. 6, we plot the main performance indexes as a function of the inner population mix $k_A$ and $k_B$. Each curve corresponds to a different outer population mix $N_A$ and $N_B$. The dashed line is the outer population mix for which the system can provide the best result. Note that, the optimal outer population mix can slightly change based on the considered index. We now consider two more indexes (i.e. depletion time $T$ and the energy per job $EJ$ ) since, with multi-class workloads, they behave differently from the energy consumption $E$. In particular, we evaluate $EJ$ using Eq. 5:

$$EJ = D\frac{P}{U} = \left(\sum_{r,c} D_{rc}\right) \frac{\frac{E}{T}}{U_1 + U_2},$$

where $U_r$ is computed according to Eq. 9 and the average power consumption $P$ is computed dividing the average energy $E$ by the average total time $T$.
We plot energy consumption in Fig. 6a. We used the same values of power as

for the single-class case. With multi-class workloads, it is possible to identify a set of inner population mixes $k_A$ and $k_B$ where the energy consumption is lower. Moreover, the lower is the number of class A jobs, the lower is the energy consumption of the system. This is due to the power consumption values that we used and to the time the jobs of class A spend into the system.

Depletion time can reach the minimum value when $N_A = 37$ and $N_B = 43$ (i.e. the dashed line). Nonetheless, configuration with $N_A = 30$ and $N_B = 50$ can provide better results when the inner population mix is highly unbalanced.

The main difference between depletion time in Fig. 6b and average response time in Fig. 6c is between configurations 20-60 and 50-30. In particular, the system has always a lower depletion time with $N_A = 20$ and $N_B = 60$. Instead, the average response time of that configuration is slightly greater than the one computed with $N_A = 50$ and $N_B = 30$.

ERP, ERWS and EJ are plotted in figures 6d, 6e and 6f respectively. All the three metrics indicate a different configuration as the best one. More generally, ERWS and EJ agree on the good performance of the 40-40 configuration, whereas ERP shows an improvement with $N_A = 30$ and $N_B = 50$.

Focusing in the 40-40 configuration (that seems to provide good results for most of the considered cases) we also analyze how a different size of the FCR $K$ can affect the performance indexes. Results for ERP, ERWS and EJ are shown in Fig. 7. Each curve corresponds to a different value of $K$ (i.e. $K = \{10, 20, 30, 40\}$) and they are plotted as a function of the inner population mix.

The ERP metric shown in Fig.7a indicates that the smaller is the FCR size, the better will be the energy-response time trade-off of the system. In Fig. 7b, ERWS seems to depends on both the FCR size and the considered inner population mix. For example, when the workload is composed for the 60% by class A jobs, it is better to work with $K = 10$. Instead, when there are only jobs of class B the system should run with $K = 30$. EJ is shown in Fig. 7c. As opposed to the ERP metric, the larger is the FCR size, the better are the performance of the system. It is true especially when the system is strongly unbalanced.

Fig. 7d compares the minimum value that the previous analyzed metrics (i.e. ERP, ERWS and EJ) can achieve for different FCR sizes $K$. In order to provide a fair comparison of the considered metrics, all values are normalized in the $[0, 1]$ range according to the following rules. First we compute:

$$\tau(K) = min_\beta(V_\beta(K)), \tag{10}$$

where $\beta$ represents the inner population mix (i.e. $k_A = \beta \cdot K$ and $k_B = K - k_A$), and $V_\beta(K)$ is the value of the metric computed for specific $\beta$ and $K$. From Eq. 10, we compute the normalized value of each metric with the following formula:

$$\alpha(K) = \frac{\tau(K) - min_K(\tau(K))}{max_K(\tau(K)) - min_K(\tau(K))}. \tag{11}$$

To that purpose, ERP has been defined as a function of both $R$ (i.e. $ERP(R) = \mathbb{E}[R] \cdot \mathbb{E}[E]$) and $T$ (i.e. $ERP(T) = \mathbb{E}[T] \cdot \mathbb{E}[E]$). It is interesting to see that the

four metrics have different trends. Since ERP(R) is defined on $R$ and ERP(T) on $T$, the two metrics have different behaviors with respect to the number of jobs in the FCR: the former is increasing and the latter is decreasing. Also EJ depends on $T$ (see eq. 4), thus it is decreasing too. Instead, ERWS has a parabolic shape.

Finally, in Fig. 8, we plot the length of the phases $\Phi(I)$, $\Phi(II)$, $\Phi(III)$ described in Eq. 9. The considered configuration is the 40-40, with 20 jobs admitted at the same time in the FCR. We divide *Phase II* and *Phase III* in three subphases in order to distinguish among all the available possibilities (i.e. both class A and class B jobs, only class A jobs or only class B jobs are in execution). Note that, the sum of the duration of all the phases is equal to the depletion time for that specific configuration. In general, it would be arguable that the longer is the *Phase I*, the shorter is the depletion time of the system, since the scheduler can only work in that phase. In Fig. 8, this can be seen when there are 12 jobs of class A and 8 jobs of class B in the FCR. Unfortunately, this is not true for all the configurations; for example, when there are 50 jobs of class A and 30 jobs of class B in the whole system, the longest *Phase I* and the shortest depletion time do not meet the same inner population mix. In that case, the system has the longest *Phase I* when 95% of jobs in the FCR belong to class A, whereas the shortest depletion time is reached when only the 70% of jobs in the FCR are of class A.

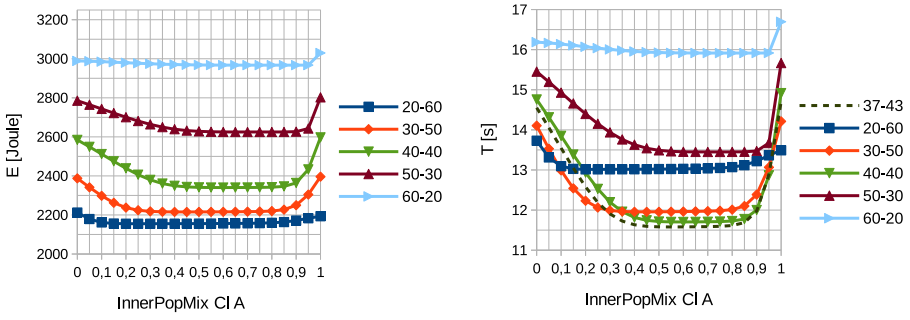
## 5 Conclusion

In this paper we investigated the performance of models of a computational paradigm consisting of a given pool of jobs of known size that must be executed by a system having a limited capacity. The objective is to optimize the performance so that the energy consumption required to execute a complete workload is minimized. To this aim, with a multi-class workload we have considered a scheduling policy that try to optimize the mix of jobs of the different classes in concurrent execution. Future works will investigate different policies and will focus on the analytical computation of the optimal point for a given metric. We are also implementing specific benchmarks to validate our theoretical approach against measurements.

## References

1. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms (TALG) 3(4), 49 (2007)
2. Andrew, L.L., Lin, M., Wierman, A.: Optimality, fairness, and robustness in speed scaling designs. In: ACM SIGMETRICS Performance Evaluation Review. vol. 38, pp. 37–48. ACM (2010)
3. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms. pp. 693–701. Society for Industrial and Applied Mathematics (2009)

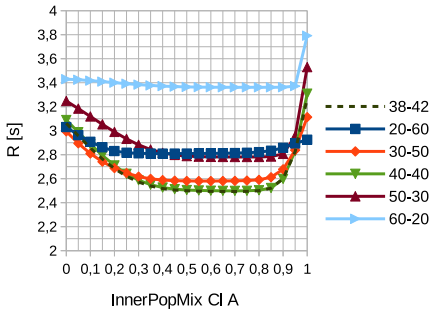
4. Cerotti, D., Gribaudo, M., Piazzolla, P., Pinciroli, R., Serazzi, G.: Multi-class queuing networks models for energy optimization. In: Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools. pp. 98–105. VALUETOOLS '14, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium (2014), http://dx.doi.org/10.4108/icst.Valuetools.2014.258214
5. Chen, D., Goldberg, G., Kahn, R., Kat, R., Meth, K.: Leveraging disk drive acoustic modes for power management. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. pp. 1–9 (May 2010)
6. Diaz-Sanchez, D., Marin-Lopez, A., Almenarez, F., Sanchez-Guerrero, R., Arias, P.: A distributed transcoding system for mobile video delivery. In: Wireless and Mobile Networking Conference (WMNC), 2012 5th Joint IFIP. pp. 10–16 (Sept 2012)
7. Fan, X., Weber, W.D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th Annual International Symposium on Computer Architecture. pp. 13–23. ISCA '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1250662.1250665
8. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.A.: Optimality analysis of energy-performance trade-off for server farm management. Performance Evaluation 67(11), 1155–1171 (2010)
9. Gonzalez, R., Horowitz, M.: Energy dissipation in general purpose microprocessors. Solid-State Circuits, IEEE Journal of 31(9), 1277–1284 (1996)
10. Hyytiä, E., Righter, R., Aalto, S.: Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure. Performance Evaluation 75, 17–35 (2014)
11. Kang, C.W., Abbaspour, S., Pedram, M.: Buffer sizing for minimum energy-delay product by using an approximating polynomial. In: Proceedings of the 13th ACM Great Lakes symposium on VLSI. pp. 112–115. ACM (2003)
12. Kant, K.: A control scheme for batching dram requests to improve power efficiency. In: Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems. pp. 139–140. SIGMETRICS '11, ACM (2011)
13. Kaxiras, S., Martonosi, M.: Computer architecture techniques for power-efficiency. Synthesis Lectures on Computer Architecture 3(1), 1–207 (2008)
14. Muppala, J., Malhotra, M., Trivedi, K.: Markov dependability models of complex systems: Analysis techniques. In: Ozekici, S. (ed.) Reliability and Maintenance of Complex Systems, vol. 154, pp. 442–486. Springer Berlin Heidelberg (1996), http://dx.doi.org/10.1007/978-3-662-03274-9_24
15. Rivoire, S., Ranganathan, P., Kozyrakis, C.: A comparison of high-level full-system power models. HotPower 8, 3–3 (2008)
16. Rosti, E., Schiavoni, F., Serazzi, G.: Queueing network models with two classes of customers. In: Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1997. MASCOTS'97., Proc. Fifth Int. Symp. on. pp. 229–234. IEEE (1997)



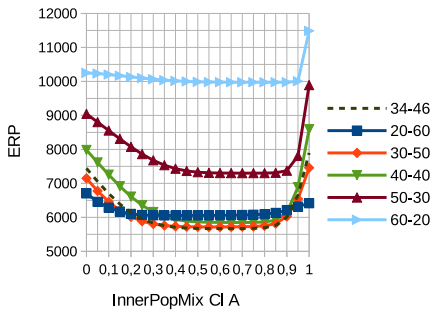

(a) Energy consumption



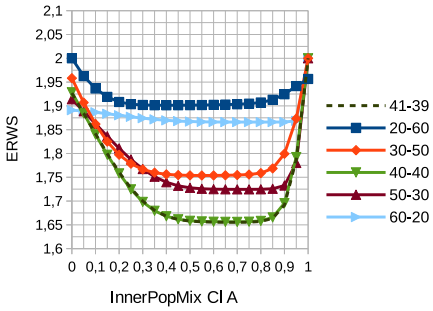

(b) Depletion time



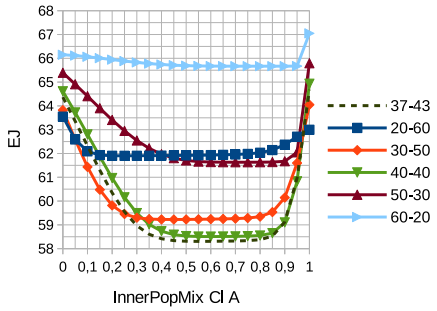

(c) Response time




(d) ERP

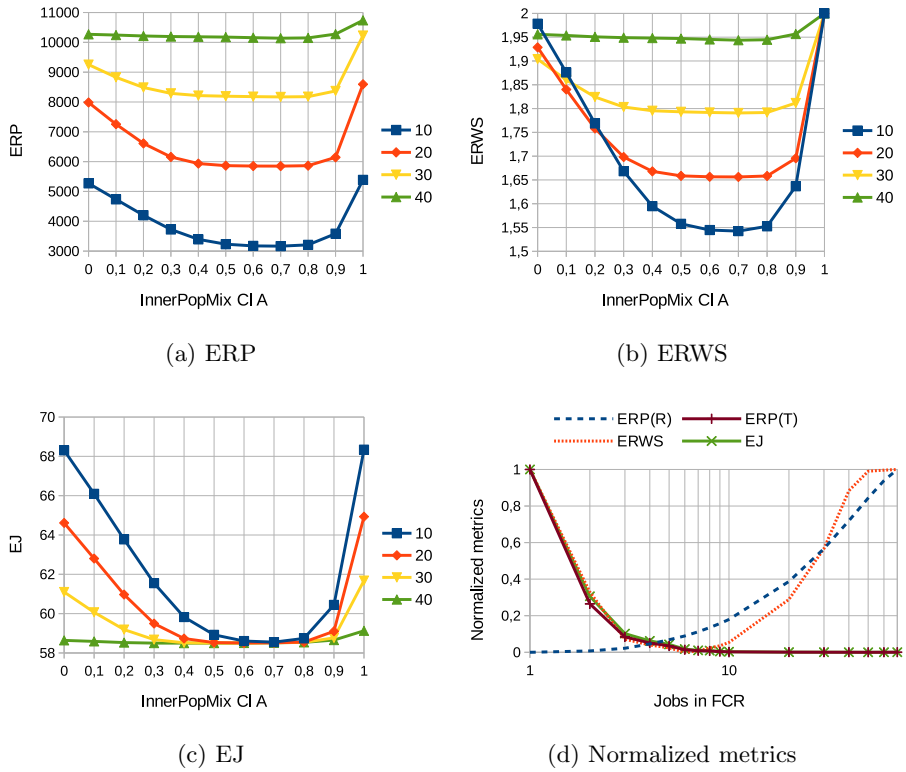



(e) ERWS




(f) EJ

Fig. 6: Performance indexes for the multi-class system with different outer population mixes. X-Y means $N_A = X$ and $N_B = Y$.




(a) ERP (b) ERWS

(c) EJ (d) Normalized metrics

Fig. 7: Performance indexes for the multi-class system with different size of the FCR.

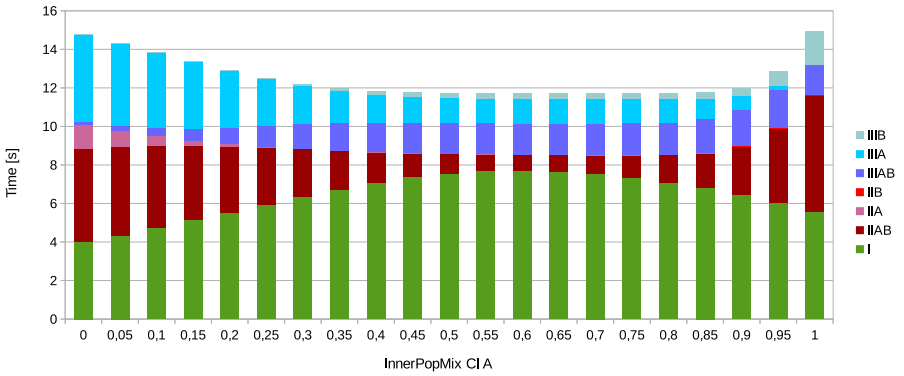



Fig. 8: Length of each phase for the execution of a two-class workload with $n_A = 40$, $n_A = 40$ and with $K = 20$ jobs in the FCR.