Optimal population mix in pool depletion systems with two-class workload

Davide Cerotti Marco Gribaudo Riccardo Pinciroli

Giuseppe 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 the concurrent processing of jobs taken from a large pool in systems with limited capacity. The definition of admission control policies that choose which jobs to process is crucial to improve the overall performance especially in systems with multiclass workload. In a previous work we show that in such systems, hereinafter called *pool depletion systems*, few parameters have a non-trivial impact on the processing time of the whole pool. Other performance indices, such as the energy consumption, are also deeply affected. In the present work, we further investigate such phenomenon by applying results from queueing theory, absorption time analysis and by performing discrete event simulations. We propose different techniques in order to identify the optimal or near-optimal setting. We analyze their complexity and provide guidelines to choose which of them adopt according to the application scenario characteristics.

Keywords

Stochastic models; pool depletion system; performance evaluation; optimization; multiclass models

1. INTRODUCTION

The replication concept is extensively applied by the big data technologies to achieve the expected performance improvements. The key feature of this computing framework is the parallel execution of a high number of processes, that may be, for example, replicas of the same request operating on different data sets or functions that summarize huge amount of data. A complete execution of a big data application consists of a sequence of parallel computations and their following synchronizations according to some user defined rules. Typically, the computing framework orchestrates the executions of the processes taking into consideration the configuration of the available architecture. The optimization of the distribution of the executions plays a fundamental role in the performance gains that can be obtained.

In this paper we approach one of the problems related to the optimization of the execution time of big data applications with different resource demands. Given a high number of processes (referred to as *pool*) belonging to two different applications, we study the problem of the minimization of the execution time required by their complete executions, referred to as *depletion time*. More precisely, knowing the resource demands of the two classes of applications, we are able to derive analytically the mixes of processes in concurrent execution in a system with finite capacity that minimize the global execution time of all the processes in the pool.

In the study presented we consider one system with two resources, e.g., CPU and storage. However, it will be easy to show that the optimization results achieved in this case, which is the least favorable, will be emphasized with an increased number of systems. We study the depletion system with different load conditions and we derive the analytical solutions in both cases with asymptotic and non-asymptotic load. To compute the results we use queueing networks, discrete event simulations and Markov analysis techniques.

The paper is organized as follows. Sections 2 and 3 provide a review of the background results and a description of the system architecture, software and hardware, used. Section 4 reports the derivation of the analytical results in the asymptotic case. The non-asymptotic load is studied in Section 5, where the results obtained with different techniques are also presented. Section 6 concludes the paper.

2. BACKGROUND

In this Section we emphasize some results on product-form closed queuing networks [1] with two-class workload and fixed rate single server stations that will be exploited in the analysis of depletion systems developed in the following Sections. Two-class workloads are enough simple to analitically deal with, while being representative of several realistic workload of actual systems. In this case, the workload of models with M stations can be characterized by a matrix of service demands $\mathbf{D} = [D_{rc}]$ where D_{rc} is the time required by station $r \in \{1, \ldots, M\}$ to completely process a class $c \in \{A, B\}$ request. Hereinafter, we will consider a sys-

ValueTools '16 October 26–28, 2016, Taormina, Italy

DOI:

tem with two resources and the following service demands:

$$\mathbf{D} = \begin{bmatrix} 0.75 & 0.64 \\ 0.48 & 1.25 \end{bmatrix} \quad (1)$$

According to matrix (1), class A service demands on stations 1 and 2 are 0.75 and 0.48 time units, respectively, whereas class B demands on the two stations are 0.64 and 1.25 time units, respectively.

The performance of systems with multi-class workloads depends on the fraction of requests in execution of each class, referred to as *population mix*. Let β represents the fraction of requests of class A in execution. In the example of matrix D, when $\beta = 0$ we have only class B requests and the bottleneck (i.e., the saturated resource) is station 2, whereas with $\beta = 1$ (only class A requests) the bottleneck becomes station 1. Thus, varying the value of β a *bottleneck switch* occurs. In these conditions, there exists a value β^* such that both stations are equally utilized for any number of requests inside the system; such value is called the *equi-utilization point*. In addition, providing that the service demand matrix values allow the occurrence of a bottleneck switch[1] and when the number of requests is sufficiently large, it is possible to identify an interval of values of the population mix that concurrently saturate the stations. Indeed, for the values of β belonging to such *common saturation sector* (CSS) the system throughput achieves the maximum and the system response time is minimum.

In [6] it is shown that the equi-utilization point is given by:

$$\beta^* = \frac{log\frac{D_{22}}{D_{12}}}{log\frac{D_{11}D_{22}}{D_{12}D_{21}}} \quad (2)$$

and it is proved that the equi-utilization point always belongs to the CSS of edges:

$$\begin{cases} \beta^L &= D_{2A}\frac{D_{2B}-D_{1B}}{D_{1A}D_{2B}-D_{2A}D_{1B}} \\ \beta^U &= D_{1A}\frac{D_{2B}-D_{1B}}{D_{1A}D_{2B}-D_{2A}D_{1B}} \end{cases} \quad (3)$$

where β^L and β^U denote the lower and upper edges, respectively. Moreover, in the same paper the behavior of the per-class throughputs are analytically computed. In particular, we have that for values of β inside the CSS the per-class throughputs are:

$$\begin{array}{rcccl} X_A^{CSS} &=& X_A(\mathbf{K}) &=& \frac{D_{2B}-D_{1B}}{D_{1A}D_{2B}-D_{2A}D_{1B}} \\ X_B^{CSS} &=& X_B(\mathbf{K}) &=& \frac{D_{2A}-D_{1A}}{D_{1B}D_{2A}-D_{1A}D_{2B}} \end{array} \quad (4)$$

where $\mathbf{K} = [K_A, K_B]$ denotes the number of requests of each class. For values of β at the right of the saturation sector (i.e. with $\beta > \beta^U$) we have:

$$X_A(\mathbf{K}) = \frac{\beta}{D_{1A}} \quad X_B(\mathbf{K}) = \frac{1-\beta}{D_{1B}} \quad (5)$$

Similar equations can be derived for values of β at the left of the saturation sector, thus obtaining the overall trend of the per-class throughputs as function of the population mix.

3. SYSTEM DESCRIPTION

We consider a system where each job consists of several independent tasks. Examples of such type of workloads are video transcoding/analysis, applications of business analytics, NoSQL queries and so on. In particular, we focus on jobs composed by two types of tasks, defined as class A and class B. For example, in multimedia stream applications each chunk is processed by a single task and the two classes may represent the computation of audio and video chunks, respectively.

We assume that the tasks of a job are executed sequentially by two resources, denoted as Res_1 and Res_2, that, for example, may represent the CPU and the storage of a server. The global service time required by resource r for a complete execution of a class c task is referred to as *service demand* D_{rc}. The service demands characterize the workload in terms of total processing requirements to the resources, and their values are considered exponentially distributed. Each resource executes the concurrent tasks according to a *processor sharing* queuing discipline: all tasks are processed by the resources with a service rate proportional to the current number of tasks in service.

We call N_A the number of class A tasks, N_B the number of class B and $N = N_A + N_B$ the total number of tasks of a job. The system can execute at most K tasks concurrently. This limitation is used to model constraints on memory occupancy that prevents all the N tasks to be executed in parallel. In particular, the system is allowed to execute K_A tasks of class A and K_B of class B, with $K = K_A + K_B$. As soon as one task is completed, another task of the same class is admitted in execution, if available. When all the tasks of one class are completed, the system allows the tasks of the other class to be processed.

Figure 1 gives a visual representation of the considered scenario. We are interested in studying the total time T required to complete the execution of all the N tasks of a job that initially are in the pool. With the type of applications considered in this study we have $N_A \geq K_A$ and $N_B \geq K_B$.

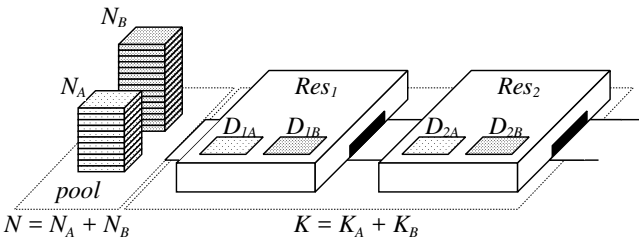


Figure 1: The considered scenario.

Figure 2 shows the temporal evolution of the number of tasks in the system. Initially, K out of N tasks immediately starts being serviced by the first resource of the system. As long as there are tasks of both classes in the pool waiting to be executed, the number of tasks in the system is constantly kept to $K = K_A + K_B$. We address this phase as Φ_{I}: after a short initial transient period, denoted with $T_{\tau 0}$, the system behaves as a closed queueing model with K_A and K_B customers, since as soon as one of the task of a class leaves the system, it is immediately replaced by another one of the same class. At time $T_{\Phi 1}$ the tasks in the pool of one class are finished and no new tasks of that class may enter the system (in Figure 2 this happens for class A tasks). At this time the system starts replacing the tasks of the exhausted class with the ones of the other class: phase Φ_{II} begins. After an initial

[1]It can be verified by checking if $D_{1A} > D_{1B}$ and $D_{2A} < D_{2B}$; see [6] for further details.

transient in which all the remaining tasks of the exhausted class are executed, the system behaves as a closed queuing model with K customers (in Figure 2 these customers are of class B). We denote the end of this transient part as $T_{\tau 12}$. After some time, the tasks in the pool that still need to be executed will end, and the system will begin to execute a decreasing number of tasks. We denote this instant of time by $T_{\Phi 2}$, and the period of time in which the server is working with less than K tasks as phase Φ_{III}. The job completes its execution when all its tasks are terminated at time T, referred to as *depletion time*.

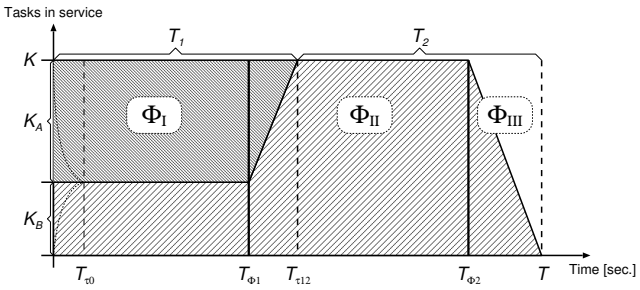


Figure 2: Behavior of the number of tasks in execution with phases characterization.

4. ASYMPTOTIC RESULTS

In this Section we describe the model used to study the asymptotic behavior of the system. This assumption is motivated by the actual big data applications in which the number of tasks that generated by a job is definetly larger than the number of tasks that can be executed concurrently in the system. Two examples of problems that can be solved through our analysis are proposed.

4.1 Asymptotic model

In order to study the asymptotic behavior of the system, let us assume that the number N of tasks in the pool is much larger than the number K of tasks admitted in execution, i.e., $N \gg K$. The region containing the resources of the system where $K = K_A + K_B$ tasks are admitted in execution is identified as Finite Capacity Region (FCR). If K is large enough to saturate the system, the per-class throughputs can be derived with closed formulas [6]. Due to the asymptotic assumption, the transient phases shown in Figure 2 are negligible and the depletion time may be computed as $T = T_1 + T_2$, since Φ_{I} and Φ_{II} are the dominant ones.

First phase Φ_{I} lasts until there is at least one task of each class in the system. Let us define the fraction of class A tasks initially in the pool, and that could be executed concurrently in the system, when available, as:

$$\alpha = \frac{N_A}{N}, \quad \beta = \frac{K_A}{K},$$

respectively. The Φ_{I} length can be derived as:

$$T_1(N, \alpha, \beta) = min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) \tag{6}$$

where $X_c(\beta)$ is the asymptotic throughput for tasks of class $c \in \{A, B\}$ when there are $K_A = \beta K$ and $K_B = (1-\beta)K$ tasks in the FCR. Φ_{I} ends when system runs out of tasks of one class.

In phase Φ_{II}, in the asymptotic case, the system always executes a single class workload (e.g., class B in Figure 2). It starts with less than N_C tasks – where C is the remaining class – since part of them have been already served during Φ_{I} and it ends when there are no more tasks in the system. Φ_{II} phase length, that is T_2, is computed as:

$$\begin{aligned} T_2(N, \alpha, \beta) &= \frac{N_A - min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) X_A(\beta)}{X_A(1)} \\ T_2(N, \alpha, \beta) &= \frac{N_B - min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) X_B(\beta)}{X_B(0)} \end{aligned} \tag{7}$$

where $X_A(1)$ ($X_B(0)$) is the class A (B) throughput when there are only tasks of class A (B) in the system. One of the two equations is considered based on the remaining class. For example, if Φ_{I} ends when tasks of class A are exhausted (i.e., $\frac{N_A}{X_A(\beta)} < \frac{N_B}{X_B(\beta)}$), the second equation in (7) must be used, since the first one is 0. Instead, if system runs out of class B tasks when Φ_{I} finishes, first equation in (7) is considered.

To derive the asymptotic depletion time of the system we use the definition previously given and we sum equations (6) and (7):

$$\begin{aligned} T(N, \alpha, \beta) &= T_1(N, \alpha, \beta) + T_2(N, \alpha, \beta) \\ &= min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) \\ &\quad + \frac{N_A - min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) X_A(\beta)}{X_A(1)} \\ &\quad + \frac{N_B - min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) X_B(\beta)}{X_B(0)} \end{aligned} \tag{8}$$

We may sum both equations in (7) since one of them is null based on the exhausted class. Collecting the minimum in equation (8) we have:

$$\begin{aligned} T(N, \alpha, \beta) &= \frac{N_A}{X_A(1)} + \frac{N_B}{X_B(0)} \\ &\quad + min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) \cdot \left(1 - \frac{X_A(\beta)}{X_A(1)} - \frac{X_B(\beta)}{X_B(0)}\right) \end{aligned} \tag{9}$$

Finally, due to the asymptotic assumption $N \gg K$, we divide equation (9) by the initial number of tasks in the pool:

$$\begin{aligned} \frac{T(N, \alpha, \beta)}{N} &= T(\alpha, \beta) \\ &= \frac{\alpha}{X_A(1)} + \frac{1-\alpha}{X_B(0)} \\ &\quad + min\left(\frac{\alpha}{X_A(\beta)}, \frac{1-\alpha}{X_B(\beta)}\right) \cdot \left(1 - \frac{X_A(\beta)}{X_A(1)} - \frac{X_B(\beta)}{X_B(0)}\right) \end{aligned} \tag{10}$$

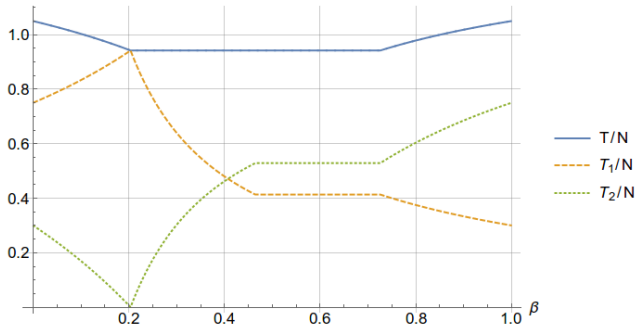
where $N_A/N = \alpha$ and $N_B/N = 1 - \alpha$.

4.2 Relative minimum depletion time

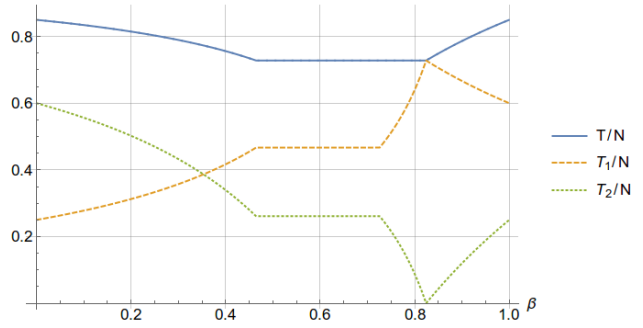
Initially we consider a configuration where all the parameters of the system (i.e., N_A, N_B, K and the D matrix) are

given by the user and only the fraction of the tasks of the two classes in the FCR (i.e., K_A and K_B) varies. Referring to equation (10), only the β value can be set by the scheduler, whereas α is given as input parameter that is kept constant. In this situation, the scheduler must search for the β value such that the depletion time computed with equation (10) is minimized.

Let us assume that the input parameters given by the user are matrix (1) and $\alpha \in [0, 1]$. The scheduler must analyze all the possible $\beta \in [0, 1]$ that can be adopted by the FCR and identify the one that minimize the depletion time. Figure 3 shows the behavior of T/N, T_1/N and T_2/N as functions of β with two values of α, 0.4 and 0.8 respectively. In Figure



(a)



(b)

Figure 3: Asymptotic depletion time as a function of parameter β for $\alpha = 0.4$ (a) and $\alpha = 0.8$ (b).

3a T is minimum when $\beta \in [0.204, 0.725]$. The common saturation sector (CSS) [6] of the considered system (i.e., $(\beta_L, \beta_U) = (0.465, 0.725)$) is included in the interval that minimize the depletion time and both of them have the same β_U. In order to minimize T, we look for the value of β that makes the executions of all class A and class B tasks end at the same time. Let us denote such value with β^+. Thus, with β^+ we have $N_A/X_A(\beta^+) = N_B/X_B(\beta^+)$. Indeed, since the system completes the execution of both classes at the same time, we have:

$$T_1(N, \alpha, \beta) = min\left(\frac{N_A}{X_A(\beta)}, \frac{N_B}{X_B(\beta)}\right) = \frac{N_A}{X_A(\beta)} = \frac{N_B}{X_B(\beta)}$$

and $T_2 = 0$. The depletion time is minimized when $K_A/K = \beta^+$, since we defined $T = T_1 + T_2$ and T_1 and T_2 are both positive, and T_1 is maximum since it is defined as the minimum value of two functions: one of them is monotonically increasing and the other one is monotonically decreasing.

Similar considerations can be applied to results of Figure 3b. In this case, the minimum depletion time is obtained for $\beta \in [0.465, 0.824]$ and this interval is sharing its minimum value with the CSS.

It is interesting to note that when $\beta^+ < \beta_L$ (i.e., the minimum value of the CSS) the depletion time is minimum for $\beta \in [\beta^+, \beta_U]$ and if $\beta^+ > \beta_U$ (i.e., the maximum value of the CSS) the depletion time is minimum for $\beta \in [\beta_L, \beta^+]$.

As said, the value of β^+ in the asymptotic condition can be computed imposing that:

$$\frac{X_A(\beta)}{X_B(\beta)} = \frac{N_A}{N_B} \tag{11}$$

and properly using equations (4-5). In particular, if substituting equation (4) in (11) we obtain an identity, then for all the values of β in the CSS the depletion time is minimum. Otherwise, assuming that β^+ belongs to the right side of the CSS, using equation (5) and after some algebraical manipulations, we have:

$$\beta^+ = \frac{N_A D_{1A}}{N_A D_{1A} + N_B D_{1B}} \tag{12}$$

where the solution is valid only if $\beta^+ > \beta^U$. Similar equation can be derived when β^+ is on the left side of the CSS.

Since depletion time is minimum for several values of β, the scheduler can consider also other performance metrics to make better choices about the value of β to use. For example, it may consider the per-class response times shown in Figure 4. In particular, it may constraint the system to operate with the minimum T and privileging one of the two classes making its tasks exit the system faster than the tasks of the other class.

Finally, note that the minimum depletion times identified in this Section are relative to the given values of α. A more general problem is studied in next Section, where an absolute minimum is identified.

4.3 Absolute minimum depletion time

In the second problem that we study, the parameters defined by the user are the initial number of tasks N in the pool, the capacity K of the FCR, and the matrix D. In this case, the scheduler chooses the proportion of tasks both inside the FCR and in the pool. In other words, it searches for the minimum depletion time varying both parameter α and parameter β in equation (10). The scheduler operates according to the following steps: *i)* it computes the equi-utilization point $\beta^* \in [\beta_L, \beta_U]$ for the given matrix D as defined by equation (2); *ii)* it derives $X_A(\beta^*)$ and $X_B(\beta^*)$ through equations (4); *iii)* it computes the optimal initial population mix of the pool α^*. Starting from equation (11) and making some derivations we obtain:

$$1 + \frac{N_B}{N_A} = 1 + \frac{X_B(\beta^*)}{X_A(\beta^*)} \tag{13}$$

that is

$$\frac{N_A + N_B}{N_A} = \frac{X_A(\beta^*) + X_B(\beta^*)}{X_A(\beta^*)} \tag{14}$$

and inverting equation (14):

$$\alpha^* = \frac{X_A(\beta^*)}{X_A(\beta^*) + X_B(\beta^*)} \tag{15}$$

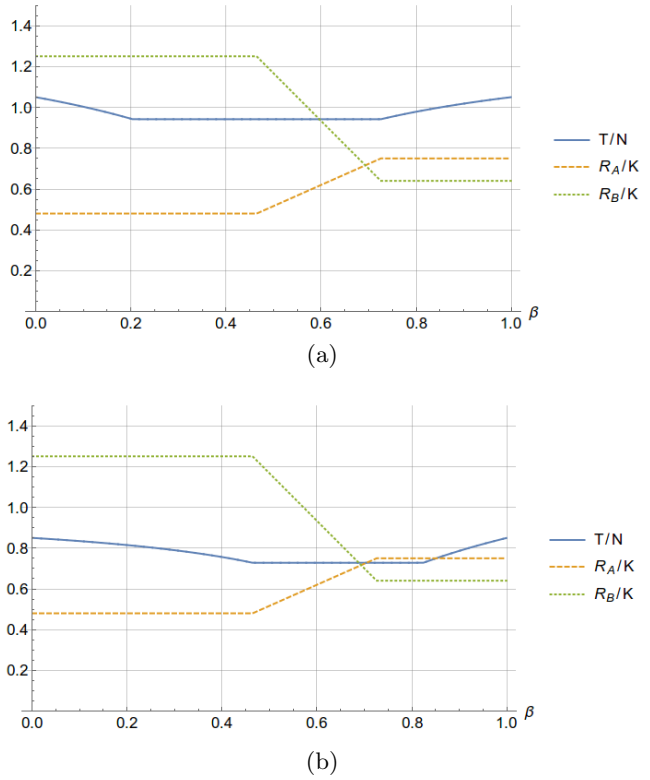


Figure 4: Depletion time and asymptotic response time as a function of β for $\alpha = 0.4$ (a) and $\alpha = 0.8$ (b).

since $\alpha^* = N_A/N$; *iv)* it forces the system work with $N_A/N = \alpha^*$ and with any number of tasks in the FCR such that $K_A/K \in [\beta_L, \beta_U]$.

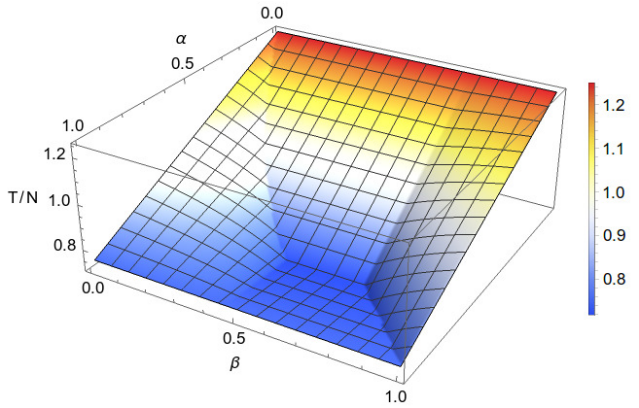


Figure 5: Depletion time as a function of α and β.

Figure 5 represents depletion time as a function of α and β, and matrix (1) has been used for the service demands of the system. It shows that an absolute minimum can be find in α^* (i.e., 0.693 for the given matrix) and for any value of $\beta \in [0.465, 0.725]$, that is, the CSS for the matrix in equation (1). If $\alpha \neq \alpha^*$ it is only possible to find a relative minimum depletion time, as seen in Section 4.2.

We also compute the gain G of the system as a function of α. It is the percentage of saved time using the best $\mathbf{K}$ – defined as $(\beta K, (1-\beta)K)$ – in the FCR for the considered value of α. The maximum and the minimum depletion times for each value of the variable α have been collected and the gain has been computed as:

$$G = \frac{T_{max} - T_{min}}{T_{min}} \tag{16}$$

Figure 6 shows the gain of the system for different proportions of the tasks in the pool, when it works with a value of β that minimizes depletion time. The system achieves the largest G when it works with α^*.

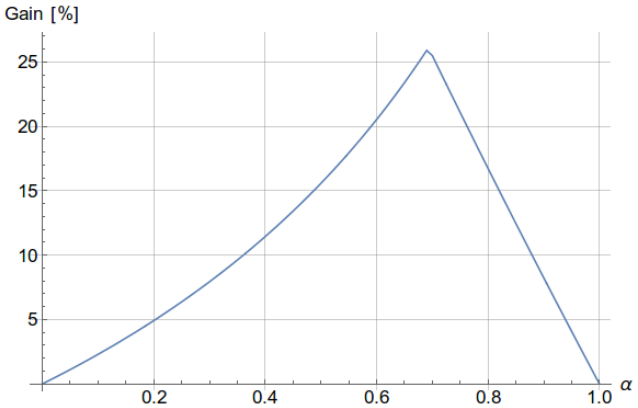


Figure 6: Gain of the system for different values of α, when it works with the best value of β.

5. NON-ASYMPTOTIC RESULTS

When the assumptions of the asymptotic results are not satisfied, the system have a slightly different behavior. In order to study the system in non-asymptotic regime, we perform both analytical evaluation using technique presented in [3], and discrete event simulation using the *JMT: Java Modeling Tool* [2]. The main differences from the asymptotic behavior come from the fact that the transient durations $T_{\tau 0}$, $T_{\tau 12} - T_{\Phi 1}$ and $T - T_{\Phi 2}$ are not negligible with respect to the total depletion time T.

5.1 Simulation model

In order to study the system under configurations with $N \gg 0$ or $K \gg 0$ that would have led to state space sizes for which the analytical technique could have not been applied, we have performed discrete event simulation using a specifically patched version of JMT. The system has been modeled by the fork/join queuing network shown in Figure 7. The model has just one job, starting in the delay station `Delay` with zero service time: the purpose of this node is to provide the reference station for computing throughput and response time, and to restart the system after one execution run. This job is immediately split into N_A class A tasks and N_B class B tasks by the fork node named `Fork`. The same node produces also one class C task, which acts as a marker to allow us to compute the duration of phase Φ_I in a simulation run. The job finishes when all tasks rejoins into the originating job in node `Join`. The two resources are modeled by nodes `Res 1` and `Res 2`, that act as single server processor sharing queues with exponential service times characterized by demands D_{1A}, D_{1B}, D_{2A} and D_{2B}. The service time

for the marker class C is zero in both stations. The policy of the scheduler is implemented in a patched version of the finite capacity region that includes both resources. In particular, we have modified the admission policy inside FCR to: *i)* allow at most K_A and K_B jobs if there are still some jobs of both classes waiting to be served in the FCR queue (phase Φ_{I}); *ii)* allow at most K jobs if one class as finished its jobs (phase Φ_{II}); *iii)* let the marker class C task enter and pass through the system when phase Φ_{I} is ended. The average system response time corresponds to the depletion time T, and the average waiting time of the class C task in the queue of the FCR defines $T_{\Phi 1}$, the total length of phase Φ_{I}.

5.2 Validation

In order to validate the results, a continuous-time Markov chain (CTMC) of the depletion system has been developed to compare the values obtained for the depletion time with those obtained by the simulator described in Section 5.1. In the CTMC the state of the system is described by a tuple counting for each class the current number of tasks in the pool, in station Res_1 and in station Res_2. The completions of tasks change the system state and are represented in the CTMC as transitions from a value of the tuple to another. Assuming that all such events follow an exponential distribution, the whole state space of the system is built and the resulting CTMC can be analysed by standard numerical techniques. In particular, the depletion time and the duration of each phase, including the transient ones, are analytically computed through absorption time analysis. Further details can be found in [3] and [5].

The analytic analysis of the CTMC suffers of the the well-known state explosion problem: large values of K makes the resulting state space grow exponentially, thus making the solution computationally unfeasible. For such reasons, we restrict the validation to scenarios with small values of K, in particular we compared the two models for different configurations with $N = 100$ and $K = 10$. With 99% confidence intervals and 3% maximum relative error the measures obtained through simulation fit those obtained with the stochastic model.

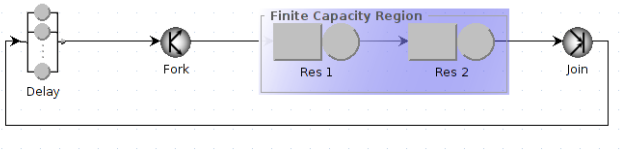


Figure 7: JMT model of the considered system.

5.3 Depletion times in non-asymptotic condition

In order to study the behavior of system in non-asymptotic regime, in this Section we use different finite values for N and K (e.g., $(N, K) = \{(100, 10),$
$(1000, 10), (1000, 100), (2000, 200)\}$). When we need to choose a value for α, we assume user is passing the closest value to α^*. For the system with matrix (1) $\alpha^* = 0.693$, as computed in Section 4.3. Let us call that system *System1*. We also analyze another system with two resources and two classes and the following service demands:

$$\mathbf{D} = \begin{bmatrix} 0.8 & 0.496 \\ 0.2 & 1.25 \end{bmatrix} \tag{17}$$

With equation (15) we derive $\alpha^* = 0.557$ for the second system, *System2*.

First of all, we study the non-asymptotic version of the *relative minimum* problem presented in Section 4.2. Figure 8 shows the length of each phase for both the systems when $N = 100$ and $K = 10$. *System1* has been analyzed for $\mathbf{N} = (70, 30)$, whereas *System2* for $\mathbf{N} = (55, 45)$. In both cases all possible $\mathbf{K}$ are taken into consideration. Note that, the sum of all the phases is the depletion time, and it is also represented in that figure. *System1* has the minimum

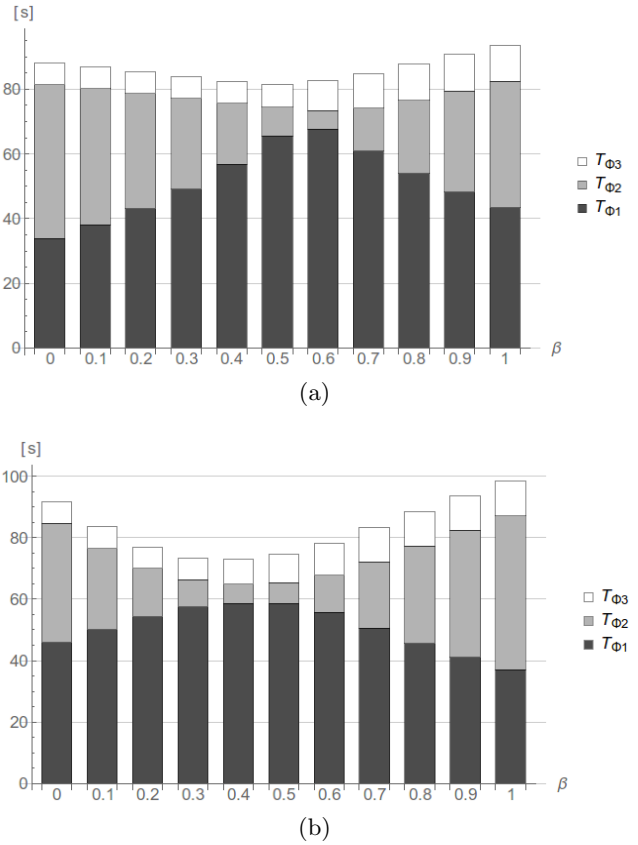
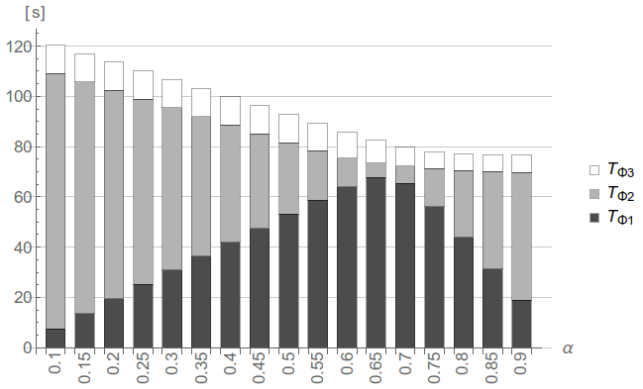


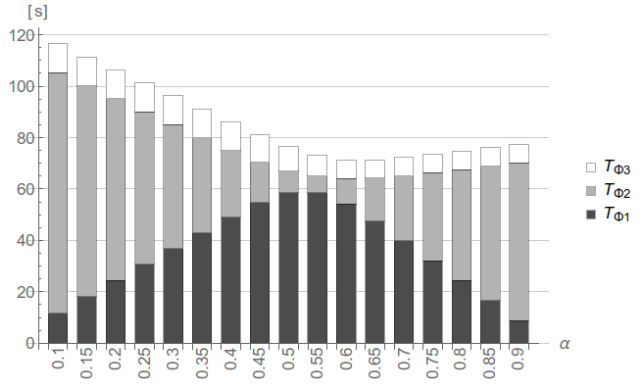
Figure 8: Length of each phase and of depletion time for *System1* when $\alpha = 0.7$ (a) and *System2* when $\alpha = 0.55$ (b).

depletion time for $\beta = 0.5$, whereas the maximum $T_{\Phi 1}$ is for $\beta = 0.6$. We have that discrepancy since now N is not asymptotically bigger than K. For this reason, transient phases (i.e., $T_{\tau 0}$, $T_{\tau 12}$ and $T_{\Phi 3}$) are now not negligible and they may affect the general system behavior. Nevertheless, this difference with respect to the asymptotic regime is not seen in all the systems. For example, in *System2* both the minimum T and the maximum $T_{\Phi 1}$ are measured for $\beta = 0.4$.

We also study the non-asymptotic behavior of the *absolute minimum* problem. For this purpose we derive β^* for both the systems (i.e., $\beta^* = 0.6$ for *System1* and $\beta^* = 0.4$ for *System2*). Then we vary α, looking for the one that make the system empty in the shortest time. The results are shown in Figure 9. Differently from the asymptotic behavior, for both



(a)



(b)

Figure 9: Length of each phase and of depletion time for *System1* when $\beta = 0.6$ (a) and *System2* when $\beta = 0.4$ (b).

the considered systems the minimum depletion time is not measured in α^*. In particular, *System1* has the minimum T for $\alpha = 0.90$, and *System2* has its minimum depletion time for $\alpha = 0.65$. Also the maximum $T_{\Phi 1}$ of the two systems is not measured for α^*, even if in both cases they are closer to the asymptotic optimal than the minimum T (i.e., $\alpha = 0.65$ for *System1* and $\alpha = 0.5$ for *System2*).

Due to the discrepancies found in previous results with respect to what has been shown in Section 4, we analyze the percentage error done by the scheduler when it uses asymptotic strategies for non-asymptotic regime. It is computed for different values of α and for the β chosen by the scheduler. The percentage error is defined as:

$$Error = \frac{|T_{min} - T_{estimated}|}{T_{min}} \tag{18}$$

where T_{min} is the real minimum value of the depletion time that can be reached with the passed parameters, and $T_{estimated}$ is the depletion time that is reached using the scheduler. For both the systems, error is never larger than 5% and for several values the asymptotic strategy used by the scheduler makes the system empty in the shortest time and the percentage error is 0%.

Also the gain in using the scheduler has been studied. In

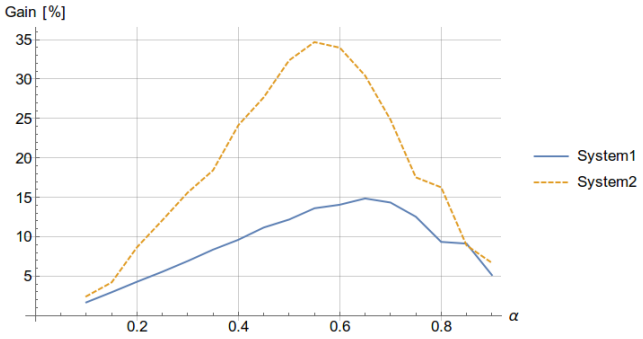


Figure 10: System gain when the scheduler is used to work with the best population mix in non-asymptitic regime.

the non-asymptotic case, it is computed as:

$$G = \frac{T_{max} - T_{estimated}}{T_{estimated}} \tag{19}$$

Results are shown in Figure 10 for both the systems as a function of α, when the β is chosen by the scheduler. *System2* has a greater gain in using the scheduler than *System1*. It depends on the service demands of the resources. Furthermore, comparing the asymptotic gain presented in Figure 6 with the non-asymptotic one for *System1* in Figure 10, we see that in asymptotic regime we have a greater gain for the same values of α. It is due both to the error done by the scheduler in a non-asymptotic regime and to the differences between the asymptotic and the non-asymptotic behaviors described at the beginning of Section 5.

Finally, in Figure 11 results obtained through simulation are shown. Here, we considered three different configurations of *System1*: $(N, K) = \{(1000, 10), (1000, 100), (2000, 200)\}$ when $\alpha = 0.7$. The confidence interval of each measure is 99% and the maximum relative error is 3%. Confidence intervals are shown in the graph. From these results we note that the larger is K, the flatter is the optimal region (i.e., the interval for which the depletion time is minimum). Indeed, if the FCR size is larger, then the system can saturate. This result supports our initial decision in considering asymptotic behavior of the system, and we expect that the larger is the value of K, the lower will be the error done by the scheduler. Since the simulation time for configuration $(N, K) = (2000, 200)$ was long, we focused only on the most interesting values of β.

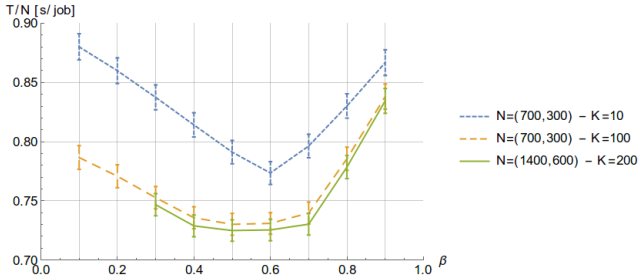


Figure 11: Simulation of different configurations of *System1* as a function of β.

5.4 Computation of the optimal class mix β

As we have seen in Section 5.3, we have verified that an appropriate choice of $\beta = \beta^+$, $K_A = \beta K$ and $K_B = (1-\beta)K$ is the one for which:

$$\frac{X_A(\beta)}{N_A} \approx \frac{X_B(\beta)}{N_B} \tag{20}$$

This is because, when $\frac{X_A(\beta)}{N_A} = \frac{X_B(\beta)}{N_B}$ the length of phase Φ_{II} tends to zero, minimizing the effects of the transients. In the following we will see two techniques for determining β depending on the concurrent execution capacity K of the system.

If K is of moderate size (say $K < 100$), we can find β^+ in quadratic time complexity $O(K^2)$, and linear space complexity $O(K)$ using a modified version of the Mean Value Analysis (MVA) algorithm for multi-class models [4]. In particular, we can insert the exhaustive search of β^+ in the MVA algorithm, and re-use the measurements of the system performances when working with one job less to compute the throughput for all the values of β at the same time. The algorithm is sketched in 1.

Algorithm 1 Modified MVA algorithm to determine β

1: $Q_1[0][0] \leftarrow 0$, $Q_2[0][0] \leftarrow 0$, ..., $Q_M[0][0] \leftarrow 0$
2: $d \leftarrow \infty$,
3: **for** $k \leftarrow 1$ *to* K **do**
4: **for** $k_A \leftarrow 0$ *to* k **do**
5: $k_B \leftarrow k - k_A$,
6: **for** $r \leftarrow 1$ *to* M **do**
7: $R_{rA} \leftarrow D_{rA}\,(1 + Q_r[k_A - 1][k_B])$
8: $R_{rB} \leftarrow D_{rB}\,(1 + Q_r[k_A][k_B - 1])$
9: **end for**
10: **for** $c \in \{A, B\}$ **do**
11: $X_c \leftarrow \dfrac{k_c}{\sum_i^M R_{ic}}$
12: **end for**
13: **for** $r \leftarrow 1$ *to* M **do**
14: $Q_r[k_A][k_B] \leftarrow \sum_{c \in \{A,B\}} R_{rc} \cdot X_c$,
15: **end for**
16: **if** $\left|\frac{N_A}{X_A} - \frac{N_B}{X_B}\right| < d \wedge k = K$ **then**
17: $\beta^+ \leftarrow \frac{k_A}{K}$, $\quad d \leftarrow \left|\frac{N_A}{X_A} - \frac{N_B}{X_B}\right|$,
18: **end if**
19: **end for**
20: **end for**

Lines 1-2 initialize the average queue length for the two resources Res_1 and Res_2 for the empty system, and the current measure of equality d for $\frac{N_A}{X_A}$ and $\frac{N_B}{X_B}$ to ∞. As for standard MVA, loops in lines 3-5 cycles respectively on the total population k, and on the class A population k_A, defining the corresponding class B population to $k_B = k - k_A$. Loop 6 computes the average residence time R_{rc} for each resource r and class c. Then the throughput X_c for each class c is computed in loop 10, and the average number of jobs in the stations for the current task population $Q_r[k_A][k_B]$ is computed in loop 13. When the total population K is reached, lines 16-18 updates the current values of β and d, if the considered class mix provides a better approximation for equation (20).

If $K \gg 0$, we can use the same approximation used in Section 2 to determine the throughputs $X_A(\beta)$ and $X_B(\beta)$, and find the β that satisfy equation (20). In particular, from equations (3-5) and (20), we have:

$$\beta^+ \approx \begin{cases} \frac{N_A D_{2A}}{N_A D_{2A} + N_B D_{2B}} & if\ N_B X_A^{CSS} > N_A X_B^{CSS} \\ \forall \beta \in [\beta^L, \beta^U] & if\ N_B X_A^{CSS} = N_A X_B^{CSS} \\ \frac{N_A D_{1A}}{N_A D_{1A} + N_B D_{1B}} & if\ N_B X_A^{CSS} < N_A X_B^{CSS} \end{cases} \tag{21}$$

where the conditions specify the position of β^+ with respect to the CSS.

6. CONCLUSIONS

In this paper we have considered the problem of determining an optimal scheduling algorithm that can reduce total time required to execute a job composed of a set of tasks belonging to two different classes, by simply determining how many task per class initially allow to be concurrently executed. We have proven that this simple policy can obtain gains in the order of 30%.
Future works will extend the model to include multiple servers resource: this will allow to use the proposed framework to optimize data analytics applications based on frameworks such as Apache Spark, in which a job is split into a large number of tasks that are concurrently executed in parallel on a number of nodes. Other studies will focus on the analysis of the impact of the proposed scheduling technique on other important metrics such as power consumption.

Acknowledgment. This work was partially funded by the European Commission under the grant ANTAREX H2020 FET-HPC-671623.

7. REFERENCES

[1] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, Apr. 1975.

[2] M. Bertoli, G. Casale, and G. Serazzi. Jmt: performance engineering tools for system modeling. *SIGMETRICS Perform. Eval. Rev.*, 36(4):10–15, 2009.

[3] D. Cerotti, M. Gribaudo, R. Pinciroli, and G. Serazzi. Stochastic analysis of energy consumption in pool depletion systems. In *Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems - 18th International GI/ITG Conference, MMB & DFT 2016, Münster, Germany, April 4-6, 2016, Proceedings*, pages 25–39, 2016.

[4] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance.* Prentice-Hall, 1984.

[5] J. Muppala, M. Malhotra, and K. Trivedi. Markov dependability models of complex systems: Analysis techniques. In S. Ozekici, editor, *Reliability and Maintenance of Complex Systems*, volume 154, pages 442–486. Springer Berlin Heidelberg, 1996.

[6] E. Rosti, F. Schiavoni, and G. Serazzi. 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*, pages 229–234. IEEE, 1997.