Multi-Class Queuing Networks Models for Energy Optimization

D. Cerotti
DEIB Politecnico di Milano
Via Ponzio 34/5 20133
Milano, Italy
davide.cerotti@polimi.it

M. Gribaudo
DEIB Politecnico di Milano
Via Ponzio 34/5 20133
Milano, Italy
marco.gribaudo@polimi.it

P. Piazzolla
DEIB Politecnico di Milano
Via Ponzio 34/5 20133
Milano, Italy
pietro.piazzolla@polimi.it

R. Pinciroli
DEIB Politecnico di Milano
Via Ponzio 34/5 20133
Milano, Italy
riccardo.pinciroli@polimi.it

G. Serazzi
DEIB Politecnico di Milano
Via Ponzio 34/5 20133
Milano, Italy
giuseppe.serazzi@polimi.it

ABSTRACT

The increase of energy consumption and the related costs in large data centers has stimulated new researches on techniques to optimize the power consumption of the servers. In this paper we focus on systems that should process a peak workload consisting of different classes of applications. The objective is to implement a policy of load control which allows an efficient use of the power deployed to the resources. The proposed strategy controls the workload mix in order to achieve the maximum utilization of all the resources allocated. As a consequence, the power provision will be fully utilized and the throughput maximized. Thus, the costs to execute a given workload will be minimized, together with its energy consumption, since the required processing time is decreased.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Performance, energy optimization, energy-aware scheduling, queuing network, multi-class workload, load control policies

1. INTRODUCTION

The energy management of computer systems has become a key issue in the design and implementation of IT infrastructures. The cost of powering computers is increased along with their improvement of performance. Typically, users of data centers expect the quality of service they require to be continuously available to their high performance requests. For this reason, and for the highly fluctuations in the flow of incoming requests, data centers are usually sized to meet peak workload with the performance required. This produce an outlay of massive amounts of energy since several servers undergo long periods of low utilization during which their energy consumptions is still significant.

The search for green IT has inspired a wide spectrum of techniques for power management that exploit, albeit in different ways, two types of basic mechanisms: the dynamic scaling of a given component's performances (Dynamic Speed Scaling) and the dynamic hibernation of components (Dynamic Resource Sleeping), see e.g.[1]. Among the techniques developed for energy efficiency in data center, there are some works on the system load, providing algorithms/policies either to balance or schedule it among the servers. Simple Linux Utility for Resource Management (SLURM) is described in [2] and is a resource management and job scheduling system for Linux based clusters of all sizes. It provides a power management facility for idle nodes that can be placed in a lower power state. In [3] the authors propose a scheduling technique to reduce the energy consumption by allocating jobs to a node based on rack information. In [4], the authors propose a load balancing policy that turns on or off cluster nodes dynamically according to the system workload. Their work is continued in [5], where the Service Level Agreement (SLA) is used to adjust active resources by making a trade-off between service quality and power consumption. By using multiple-class production workloads, in [6], the authors to quantify how power usage patterns are affected by workload choice. To our knowledge this is the first power usage study at the scale of datacenter workloads, and the first reported use of model-based power monitoring techniques for power provisioning in real production systems. In [7] several high-level full-system power models for a wide range of workloads and machines are generated and their accuracy evaluated with experimental data. Results show the limitations of previously proposed models based solely on OS-reported component utilization for systems where the CPU is either aggressively power-managed or is not the dominant consumer of dynamic power. The work in [8] focused on energy-efficient parallel application job allocation for heterogeneous architectures. In heterogeneous architectures, the system power model becomes much

VALUETOOLS 2014, December 09-11, Bratislava, Slovakia

DOI 10.4108/icst.valuetools.2014.258214

more complicated because the power consumption patterns, power state spaces, and power management mechanisms are different.
In [9] a model is proposed to make a trade off between performance of the system and its power consumption, by dynamically powering down servers and powering them up again according to demand. Several heuristic policies that try to optimise the behaviour of the system are also proposed.

In this paper we apply queuing networks to derive the values of the parameters for the load control so that the energy consumption of servers processing a multi-class workload is minimized. It is known that the performance of systems providing services to different classes of users depends on the population mix of the workload that the systems process. More precisely, given a set of resources characterized by their service demands, it is possible to identify a set of population mixes corresponding to which some of the resources are saturated simultaneously and the system achieves its best performance. It is known that one of these population mixes allow two resources to be equiutilized regardless the population sizes. This operational condition is optimal since it maximizes the utilization of the resources and the system throughput. Thus, with this optimal mix in execution the power provisioned is fully utilized and, consequently, the energy consumption per job executed is minimized.
Since the arriving flow of requests of the various classes is subject to fluctuations, in this work we propose strategies to shape and redirect the incoming workload to allow one or more interconnected systems to work at their optimal conditions. As a result, for a larger range of population mixes the interconnected systems work as much as possible at their optimal conditions, spending less energy to execute each job and thus the total energy consumption is reduced. The strategies and their variations are analyzed to evaluate the achieved gain in terms of both performance and energy reduction. Such approach takes advantage in presence of high heterogeneity among the requests of each system characterized by large differences between their optimal mixes.
The remainder of the paper is organized as follows. Section 2 presents the model used to characterize energy consumption using queuing networks. In Section 3 we extend the model to include multi-class, multiple stations systems to show that the energy consumption is minimized when the system works with an optimal population mix. In Section 4 we add an energy-aware mechanism that changes the incoming population mix workload, while in Section 5 results are presented to show the validity of our approach. Section 6 concludes the paper presenting some future directions

2. ENERGY CONSUMPTION CHARACTERIZATION

As seen in Section 1, energy consumption has been analyzed in several works, and it has been related to several parameters of the considered systems. In this work we will focus on energy consumption models that can be related to classical performance indexes, such as utilization, throughput and system response time, provided by queuing network models.

2.1 Behavior

A data center machine is characterized by several components: CPUs, storage units, networking, as well as other devices to control the infrastructure. In many works, see e.g. [6] [7], the instantaneous power consumption of a server is shown to have a linear relation to the CPU utilization. This linear relation seems to still hold true with today's high-end systems, as shown in Figure 1, where results from a set of measurement collected on an *Intel i7* are presented. The estimated utilization of several workloads vs the power consumption in *Watts* are shown. To perform the tests we used a 4 cores, 2 simultaneous multi-threaded (SMT) machine running at 2.4GHz. This architecture provides 8 CPUs for running the applications. The machine's operating system was Ubuntu 14.04.1(Trusty Tahr)[10] and we used the `powerstat`[11] command as the energy consumpion meter. To simulate an increasing CPU-intensive workload, we run the Sunflow benchmark of the DaCapo suite[12] increasing from 1 up to 8 the number of simultaneous threads generated by the application at each run. Since each thread fully utilizes a CPU, the total utilization of the machine is approximately 12.5% times the number of used threads. This explains the small clusters of data visible in Figure 1.

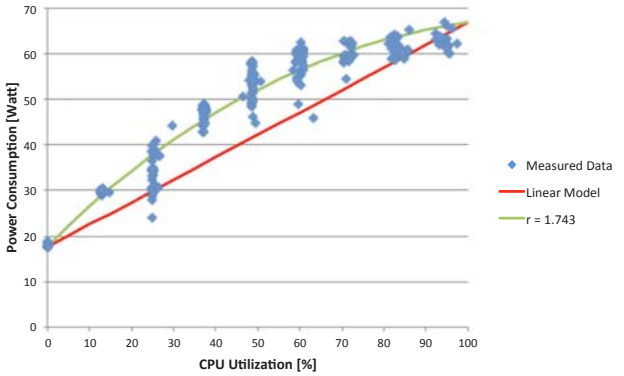

Figure 1: Power consumption models vs utilization

Let U be the CPU utilization, the power consumption $PC(U)$ of a server can be described as [6]:

$$PC(U) = P_{idle} + U \cdot (P_{max} - P_{idle}) \quad (1)$$

where P_{idle} is the power consumed when no user application is running, and P_{max} is the maximum power consumption when the server is 100% utilized. Note that both $PC(U)$, P_{idle} and P_{max} are measured in $Watt$ since they represent the instantaneous amount of energy consumed by the application. In the example of Figure 1, parameters have been estimated as $P_{max} = 66.85\ Watt$ and $P_{idle} = 17.59\ Watt$.

Although in this paper we focus on linear models only, the proposed results are still valid for any expression of the power consumption that increases monotonically with respect to the utilization. For example, a slightly more complex expression for $PC(U)$, as defined in [13], can be the following:

$$PC(U) = P_{idle} + (2U - U^r) \cdot (P_{max} - P_{idle}) \quad (2)$$

where r must be experimentally determined by data collected from the considered system. In the example shown in Figure 1, this shape parameters has been estimated as $r = 1.743$.

2.2 Metrics

From Eq.1, it seems that in order to reduce the power consumption we must reduce the utilization of a server. However, reducing the utilization means also decrease the productivity of the server. A better characterization of a server energy consumption is the *average energy consumed per job* $EJ(U)$. Let us focus on a time interval T: if $PC(U)$ is the average power required by the server, then the total energy required during T is $PC(U) \cdot T$. The number of jobs completed by the server in the interval T is $C = X \cdot T$, where X represents the throughput of the server. Thus, $EJ(U)$ is:

$$EJ(U) = \frac{PC(U) \cdot T}{C} = \frac{PC(U) \cdot T}{X \cdot T} = \frac{PC(U)}{X} \tag{3}$$

Since, for the *utilization law* we have that $U = X \cdot D$, where D represents the average service demand, we can express the energy consumption per job as a function of the utilization:

$$EJ(U) = D \cdot \frac{PC(U)}{U} \tag{4}$$

If we insert Eq. 1 into Eq. 4 we obtain:

$$\begin{aligned} EJ(U) &= D \cdot \frac{P_{idle} + U \cdot (P_{max} - P_{idle})}{U} \\ &= D \cdot \left(P_{idle} \cdot \frac{1-U}{U} + P_{max} \right) \end{aligned} \tag{5}$$

Figure 2 shows the average energy per job for $D = 1sec$ and $P_{max} = 1Watt$ for different values of P_{idle}. As $U \to 0$, although the power consumption is minimized, the energy per job increases, while as $U \to 100\%$ it tends to its minimum value $D \cdot P_{max}$. So, in order to reduce the energy required per job, the utilization of the system must be maximized, even if this can result in a higher instantaneous power consumption.

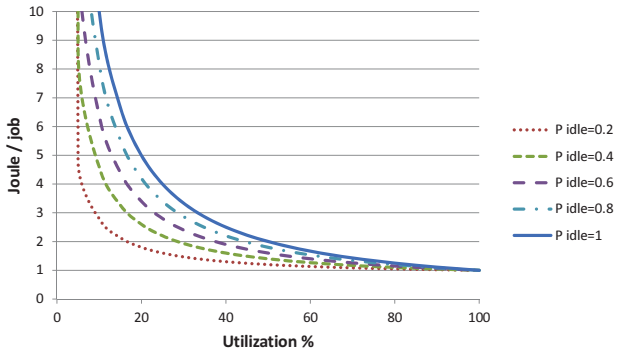

Figure 2: Energy consumption per job, for $D = 1$ *sec.*, $P_{max} = 1$ *Watt* and different P_{idle}.

Let us now consider a system composed by R different resources, each characterized by a specific power consumption $PC_r(U_r)$. We can define a *system-wise* power consumption PC_S and energy consumption per job EJ_S as:

$$PC_S = \sum_{r=1}^{R} PC_r(U_r) \qquad EJ_S = \frac{PC_S}{X} \tag{6}$$

where X is the *system throughput*. As observed in Eq. 5, each resource works with the minimum energy consumption per job when it is fully utilized. We characterize this condition with the index EE_S that we refer to as *Energy Efficiency*. In particular, we want that $EE_S \to 1$ as the system utilizes all its resources in the most efficient way, and that $EE_S \to 0$ when all resources are used at their worst. We define EE_S as:

$$EE_S = \frac{1}{R} \cdot \sum_{r=1}^{R} \frac{PC_r(U_r) - PC_r(0)}{PC_r(100\%) - PC_r(0)} \tag{7}$$

It can be easily seen that Eq.7 has the two desired properties of $EE_S \to 0$ if $U_r \to 0, \forall r$ and $EE_S \to 1$ if $U_r \to 100\%, \forall r$. Moreover, in the special case in which all the resources are identical and follow the linear behavior of Eq.1, the Eq.7 can be simplified to:

$$EE_S = \frac{1}{R} \cdot \sum_{r=1}^{R} \frac{P_{idle} + U_r \cdot (P_{max} - P_{idle}) - P_{idle}}{P_{max} - P_{idle}} = \frac{\sum_{r=1}^{R} U_r}{R} \tag{8}$$

3. ENERGY OPTIMIZATION IN SYSTEMS WITH MULTI-CLASS REQUESTS

Given the significantly different service demands exhibited by nowadays web applications, multi-class models are the most appropriate technique to characterize their workload. For example, in a site for e-commerce some requests heavily load the DB server, e.g., search operations on the catalog, while others stress particularly the application server, e.g., operations to finalize the purchase. The load generated by the various requests can be modeled accurately by multiple classes of users, each class representing a different type of requests.

In this paper we consider large-scale data centers comprising thousands of servers processing millions of requests. Typically, in this type of computing environment Quality of Service considerations suggest to limit the number of requests in execution in each server. Several studies are reported in literature addressing the problem of energy consumption minimization of a set of servers subject to fluctuating arrival traffic. Consolidation policies, reduction of active servers, and decrease of clock frequency are among the actions suggested. In this paper we consider the same problem focusing only on periods of peak load. In this condition, the flow of arriving requests is much higher than usual, and the limitation of the maximum number of requests in execution on each server is active.

According to the considerations described above, in the following we use closed multi-class queuing networks to model the servers and the workload considered. More precisely, we consider two classes of requests and two queue stations for each system, representing the computation and the storage servers, respectively.

Some of the considerations apply for two-station models only while others can be extended to models with a higher number of resources and with a higher number of classes. More generally, we consider that for each class of requests there is a resource that is utilized the most, i.e., the *bottleneck* of that class. Such station must be different for each class.

Note that when the classes have the same bottleneck, the technique discussed in this work is still valid, but less effective.

Let N be the *total number of requests* in execution in a system consisting of two resources modeled with two fixed rate queue stations. Since the model is closed and the workload comprises two classes of requests, it will be: $N = N_1 + N_2$ where N_c is the number of requests of class c in execution. Let $\underline{\beta} = (\beta_1, \beta_2)$ be the vector describing the proportion of the requests of each class that are in execution, referred to as *population mix*. Thus, it is $\beta_1 = N_1/N$ and $\beta_2 = N_2/N$ with $\beta_1 + \beta_2 = 1$. Therefore, the population of the system is given by the vector $\underline{N} = (\beta_1 N, \beta_2 N)$.
For simplicity, we parameterize our model using the service demands $D_{r,c}$ of the requests, i.e., the total amount of time that a request of class c requires in service at resource r for a complete execution. The characteristics of the classes are described by a *demand matrix* $\underline{D}$. For example, consider the following demand matrix where per-class demands are by column and per-resource demands are by row:

	Class 1	*Class* 2
Resource 1	0.75	0.64
Resource 2	0.48	1.25

(9)

According to matrix (9), class 1 service demands on resources 1 and 2 are 0.75 *sec* and 0.48 *sec*, respectively. Class 2 service demands on resources 1 and 2 are 0.64 *sec* and 1.25 *sec*, respectively. The bottleneck of class 1 is resource 1, while the bottleneck of class 2 is resource 2. The workload of the system is completely described by the vector $\underline{N}$, or alternatively by the *total number N of requests* in execution and the *population mix* vector $\underline{\beta}$, and the *service demand* matrix $\underline{D}$.

3.1 Optimal operational mix

In the following, we will use some peculiar properties of two-class models to approach the energy consumption minimization problem. It is known that in systems with multi-class workloads having at least two distinct class bottlenecks, i.e., the resource most loaded by a class is different from that most loaded from the other classes, two or more resources may saturate concurrently as a function of the population mixes. In this case, varying the population mix, keeping constant the total number of requests in execution, it is possible to observe that the saturation condition moves among different resources. In [14] it is shown that when the bottleneck migrates between two resources there exists a set of population mix, referred to as *common saturation sector*, corresponding to which the two resources saturate simultaneously. The population mixes that belong to a common saturation sector are interesting for our purposes since they maximize the sum of the utilization of the two resources that saturate concurrently, and thus maximize the Energy Efficiency EE_S of Eq.8. We will see that this condition corresponds to the minimization of the energy consumption per request EJ_S. Let us focus now on a system with two resources and two class of requests. In [15] it is shown that in such a system there exists a combination of requests of the different classes such that the resources are equally utilized regardless of the total number of requests in execution. The corresponding mix $\underline{\beta}^*$, referred to as *equiutilization mix*, is within a common saturation sector and maximizes the sum of the resource utilizations for all population sizes. So, with this mix of classes the condition of saturation with very high population size can be relaxed. The equiutilization mix, as a function of the service demands $D_{r,c}$, in [15] is given by:

$$\underline{\beta}^* = \left(\beta_1 = \frac{\ln \frac{D_{22}}{D_{12}}}{\ln \frac{D_{11} D_{22}}{D_{12} D_{21}}} \quad , \quad \beta_2 = 1 - \beta_1 \right) \tag{10}$$

It is shown that the equiutilization mix provides an optimal operational point of the system where the system power, a metric defined in [16] as the ratio of system throughput to system response time, is maximized. This means that the system operates with the best performance since the global utilization of the resources is maximum and so is the ratio of throughput to response time. This condition corresponds to having the maximum throughput with the minimum response time, and thus to have the minimum energy consumption per job EJ_S.
Let us remark that with multi-class workload the system throughput must be properly weighted to take into account the differences among the service demands of the various classes. Indeed, classes with high total service demand, i.e., the total service demand of class c is $\sum_r D_{r,c}$, are slower and have smaller throughput than fast classes, i.e., those with small total service demands. The weight should balance the amount of work completed per time unit, i.e., throughput, with the service time it requires. According to [15], we normalize system throughput by multiplying the per-class components by the ratio of the total service demand of each class to the total service demand of all the classes. Thus, the *normalized system throughput* $X^{'}$ is:

$$X^{'} = \frac{D_{11} + D_{21}}{D} X_1 + \frac{D_{12} + D_{22}}{D} X_2 \tag{11}$$

where the total service demand of all the classes is $D = \sum_r \sum_c D_{r,c}$. Using the utilization law $U_{r,c} = X_c D_{r,c}$, and considering that $U_{r1} + U_{r2} = U_r$, where U_r is the total utilization of resource r , Eq. 11 becomes

$$X^{'} = \frac{U_1 + U_2}{D} \tag{12}$$

showing that the normalized throughput $X^{'}$ is equal to the sum of the utilizations of all the resources scaled with the total service demand of the workload. Eq.12 shows that the normalized throughput is directly proportional to the sum of resource utilizations. Since in multi-class workload both PC and normalized throughput $X^{'}$ depend on the utilizations, which in turn depend on the population mix $\underline{\beta}$, we use the notations $PC(\underline{\beta})$ and $X'(\underline{\beta})$. With the new notation, Eq. 6 is extended for multi-class workload in:

$$PC_S(\underline{\beta}) = \sum_{r=1}^{R} PC_r(\underline{\beta}) \qquad EJ_S = \frac{PC_S(\beta)}{X'(\underline{\beta})} \tag{13}$$

In the closed model with constant population N the utilizations of the stations with respect to all the possible population mixes are maximized with the optimal population

mix $\underline{\beta}^*$. With this mix, as N grows the utilizations tend to 100%, thus the power consumption PC_S is maximized and the energy consumption per job EJ_S is minimized. As a consequence, the optimal population mix provides the best result in term of both performance and energy consumption.

4. THE ENERGY-AWARE SCHEDULER

Section 3 shows that with multi-class workload the energy consumption is minimized when the system works in its optimal condition, i.e. when the population mix is equal to $\underline{\beta}^*$. However, the workload of the system is often variable in unpredictable ways and thus the system rarely works at its optimal mix. In this Section, we propose to add an energy-aware component that changes the incoming population mix workload.

4.1 Architecture

We consider a collection of several sub-systems with heterogeneous resources that may have different optimal population mixes. For simplicity we assume that each of them is composed by two resources. Moreover, we assume that the *capacity* W of each sub-system is limited, i.e., the number of jobs admitted in each sub-system is bounded. Then, we add to the global system the energy-aware component to control the incoming workload, as shown in Figure 3.

Since each sub-system has a limited capacity, when all of them are full, the further incoming jobs will wait in the queue of the energy controller. Such component monitors all the sub-systems in order to evaluate their actual population mixes. According to these values the controller implements two strategies that aim to preserve as much as possible the actual population mix of each sub-system equal to the corresponding target mix value.

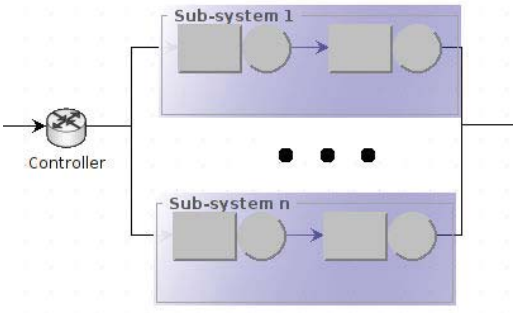

Figure 3: Architecture of the energy optimizer

4.2 Energy-aware strategies

The *scheduling strategy* identifies the next job, and in particular its class, that will be selected from the queue of the controller and directed towards the sub-systems. The *routing strategy* decides to which sub-system the selected job will be forwarded. The destination sub-system will be selected so that the distance between the new mix and its optimal target mix is minimized with respect to all the sub-systems.

The entire process is tuned by the following parameters.
Window - It identifies the portion of the controller queue from which the jobs are selected. A larger window increases the chance to have enough jobs of the proper class to preserve the target mix.
Age - In case of a very unbalanced target mix, jobs of one class will be very frequently forwarded to the sub-systems, whereas the jobs of the other class will wait for a very long time. The *Age* parameter prevents starvation: any job waiting longer than the Age value will be always forwarded ignoring its effect on the population mix.
Drop - When a job has waited longer than the Age parameter, but all sub-systems are full, the controller must take a decision. It will drop the job, in case of a Drop policy; it will keep the job in its queue, otherwise.

Let $\underline{\beta}^*_{Sys_i}$ the optimal mix of the sub-system i and $\underline{\beta}^{act}_{Sys_i}$ the actual population mix of the sub-system i as computed by the controller. The two strategies are described by Algorithms 1 and 2. First, the scheduler strategy inspects for a job in the front of the queue of the controller older than the Age parameter (Algorithm 1: line 1-10). When all subsystem are full, such job will be dropped or it will stay in the queue according to the Drop policy chosen. If at least a subsystem is not full, such job will be forwarded. In absence of too old jobs, the strategy computes for all pair of classes and sub-system (c, Sys_i) the mix that will result from forwarding a class c job to sub-system i (Algorithm 1: line 15-17). The scheduling strategy forwards the first job in the queue of the class that minimizes the distance between the computed population mixes and the corresponding target ones (Algorithm 1: line 18-19). Analogously, the routing strategy forwards the selected job to the sub-system that minimizes the same measure (Algorithm 2: line 1-5).

Notice that the scheduling strategy must consider together both the class of the job and the sub-system at which it will be forwarded, to take the proper choice. Typically, the scheduling and routing decisions are logically separated and so they have been described. Clearly, the algorithms can be integrated together in order to avoid duplicated computations.

5. RESULTS

We evaluate the scheduler proposed in Section 4, by initially applying the strategy to a system consisting of a single sub-system and then to a most complete case with two subsystems. In these experiments we do not consider the age parameter. We have extended the simulator JSIMgraph of the JMT tool [17] to implement the energy optimizer scheduler.

5.1 Single system

We start by applying the energy optimizer to a system composed by a single set of two resources, characterized by the demands defined in Eq. 9, with a global population of $N = 50$ jobs and W of them in the sub-system, as shown in Figure 4. In this case, the model can be studied analytically. After an initial period, as soon as the population mix in the sub-system reach a value close to β^*, whenever a job of a given class exits from the sub-system, it is replaced by another one of the same class selected from the jobs waiting in the energy controller queue. Thus, the system can be analyzed as a closed model with just the jobs inside the sub-system. The resulting closed model is shown in Figure

Algorithm 1 Scheduling strategy

Input: Set of $\underline{\beta}^*_{Sys_i}$, Window, Age, Drop option

FJ=First job in Window of the controller queue
if FJ too old **then**
  if All Sub-system full **then**
    if Drop option **then**
      Drop FJ
    end if
    return NULL
  else
    return FJ
  end if
else
  if All Sub-system full **then**
    return NULL
  end if
  for all pairs (c, Sys_i) **do**
    $\underline{\beta}^{act}_{Sys_i}(c) = ComputeMix(c, Sys_i)$
  end for
  $c^* = \arg\min_{(c,Sys_i)} |\underline{\beta}^{act}_{Sys_i}(c) - \underline{\beta}^*_{Sys_i}|$
  return the first job in Window of class c^*, when it is present, of the other class, otherwise
end if

Algorithm 2 Routing strategy

Input: Set of $\underline{\beta}^*_{Sys_i}$, c^* computed by scheduling strategy

for all Sys_i **do**
  $\underline{\beta}^{act}_{Sys_i}(c) = ComputeMix(c^*, Sys_i)$
end for
$Sys^* = \arg\min_{Sys_i} |\underline{\beta}^{act}_{Sys_i}(c^*) - \underline{\beta}^*_{Sys_i}|$
return Sys^*

4 enclosed by the inner box and where the dashed arrow represents the new path of the jobs.

Let us call $W_1 = \beta^*_1 W$ and $W_2 = \beta^*_2 W$ (with $W_1 + W_2 = W$) the number of jobs in the sub-system for the two classes. We apply the *Mean Value Analysis* (MVA) to solve a system with W_1 and W_2 jobs. Since we have only one sub-system, the utilization and the throughput computed with the MVA also apply to the complete model. In order to compute the global response time, we add to the response time of the sub-system the time spent in the queue of the scheduler computed using Little's law: in particular, if we call X_1 and X_2 the throughput of the two classes, the time spent in the scheduler can be computed as $(N_1 - W_1)/X_1$ and $(N_2 - W_2)/X_2$.

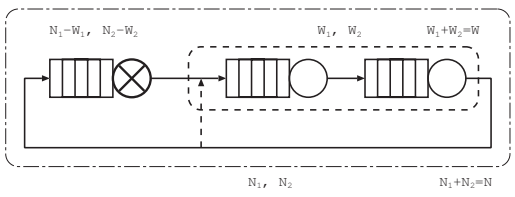

Figure 4: Single system closed model.

We start by validating the analytical solution with the results obtained by the simulator as shown in Figure 5a. In particular it can be seen that the analytical model perfectly matches the results provided by the simulation when the capacity of the sub-system is $W = 20$. The figure also shows that with $W = N$, the analytical model matches the simulation of the system without the energy optimizer.

We then analyze the effect of the size of the capacity W on the energy efficiency (Figure 5b) and on the energy consumption per job (Figure 5c). As the capacity decreases, the maximum efficiency tends to become smaller, but it is reached for a larger set of input mixes. This allows to optimize the energy consumption by selecting the best capacity W for a given input mix. As shown in Figure 5d, for a fixed input mix the energy consumption per job has a minimum that represents the best capacity for the considered β. This optimum tends to the maximum capacity (which means that the energy optimization is not useful) only if the input mix corresponds to the optimal operational point of the sub-system. The "*Optimal W*" curve in Fig.5c represents the minimum energy consumption per job that can be obtained when the optimal capacity corresponding to each input mix is used.

Note that as N increases, the energy optimizer becomes more effective, by making the energy efficiency with optimal capacity closer to 1, as it can be seen in Figure 5e. Reducing the energy consumption has however a negative effect on the throughput of the class with more jobs than the one at the optimal mix, that is slowed down to allow the sub-system working at its optimal point (see Figure 5f). This suggests that the optimization can be further tuned to provide the best tradeoff between the energy consumption and the per-class performances.

5.2 Two heterogeneous sub-systems

We have also applied the proposed procedure to two subsystems, as shown in Figure 6. They are characterized by different demand matrix, in particular we have chosen them to be opposite:

$$\underline{D}_1 = \begin{vmatrix} 0.75 & 0.64 \\ 0.48 & 1.25 \end{vmatrix} \qquad \underline{D}_2 = \begin{vmatrix} 0.64 & 0.75 \\ 1.25 & 0.48 \end{vmatrix} \tag{14}$$

Figure 7 shows the resulting energy efficiency computed via simulation using the extended version of JMT. The total number of jobs is $N = 100$, and both sub-systems have a capacity $W = 20$. As it can be seen, the energy optimizer obtains the highest energy efficiency for almost all the input mixes between 0.2 and 0.8.

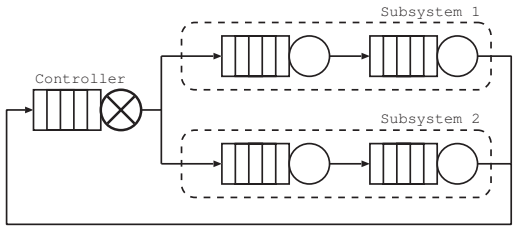

Figure 6: Closed model with two heterogenous subsystems.

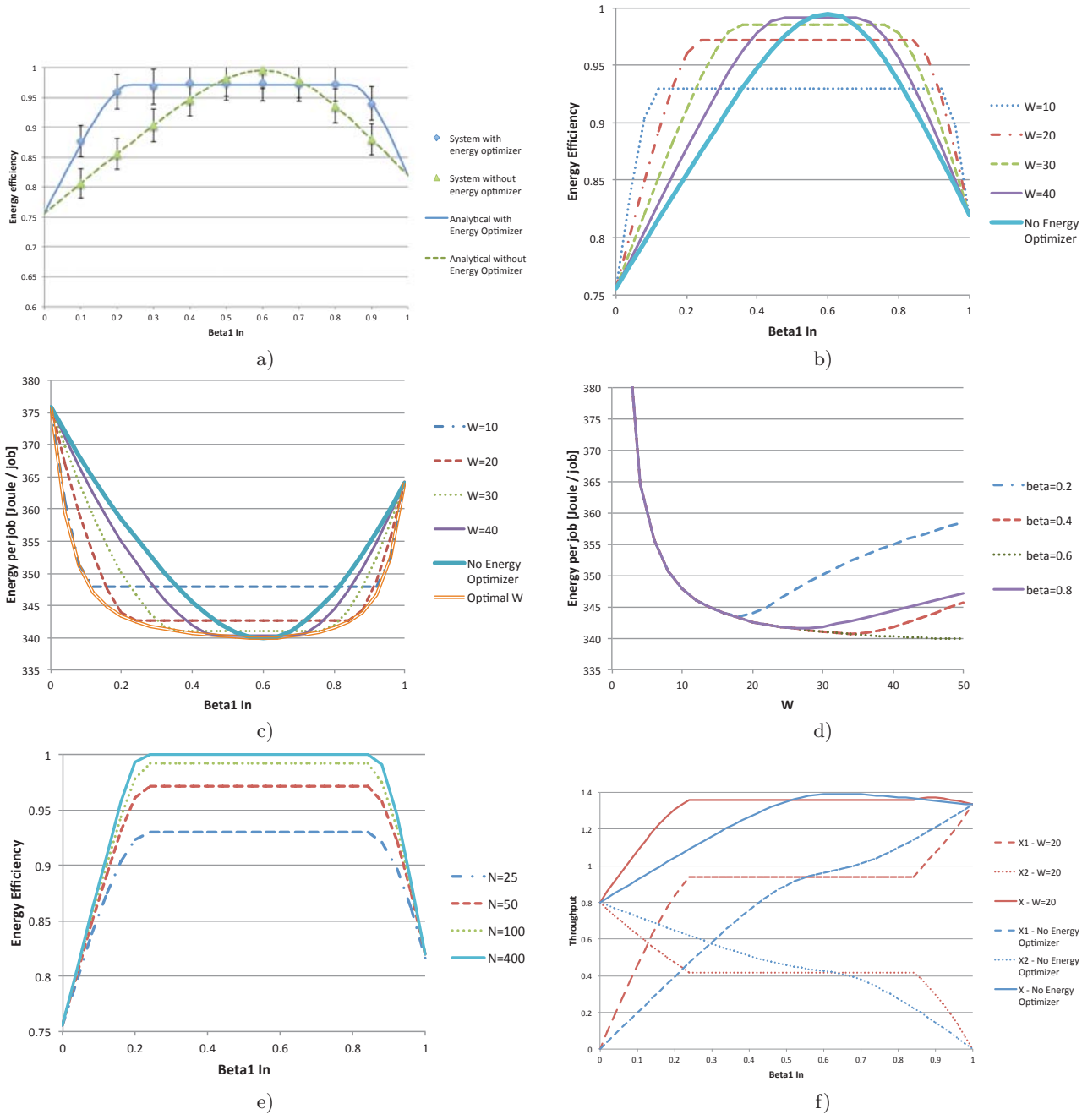

Figure 5: Single system closed mode performance indices and validation: a) Energy efficiency validation, b) Energy efficiency as function of the capacity, c) Energy per job, d) Optimization of the capacity, e) Energy efficiency as function of the total population, f) System Throughput

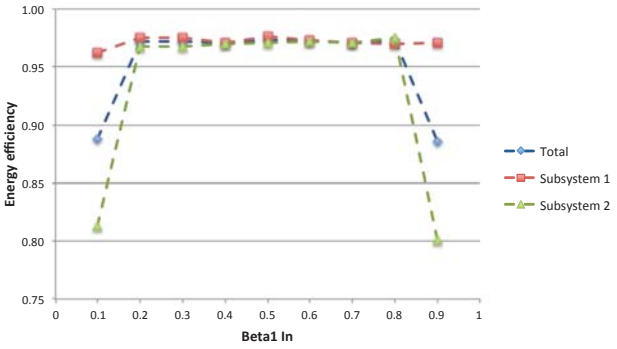

Figure 7: Energy efficiency of a model with two heterogeneous sub-systems

6. CONCLUSIONS

In this paper we have explored the possibility of using known results on optimal operational point of multi-class queueing networks to reduce the energy consumption per job. A new type of energy-aware scheduler has been proposed. Simulations with a two-class workload have shown that reduction of energy per job can be obtained as a function of the proportions of the two classes. We are currently implementing the proposed algorithm to experimentally validate the results presented in this paper. Future works will include evaluating the effectiveness of the technique with bursty arrivals, and the study of its effectiveness in a very large scale system using fluid approximations and mean-field techniques.

7. REFERENCES

[1] Yongpeng Liu and Hong Zhu. A survey of the research on power management techniques for high-performance systems. *Software: Practice and Experience*, 40(11):943–964, 2010.

[2] Bull LLNL, H.P. Simple linux utility for resource management. Available at `https://computing.llnl.gov/linux/slurm/`.

[3] V.A Patil and V. Chaudhary. Rack aware scheduling in HPC data centers: An energy conservation strategy. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 814–821, May 2011.

[4] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. Load balancing and unbalancing for power and performance in cluster-based systems. Available at `http://www2.ic.uff.br/~julius/stre/pinheiro01load.pdf`, 2001.

[5] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centers. *SIGOPS Oper. Syst. Rev.*, 35(5):103–116, October 2001.

[6] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.

[7] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower'08, pages 3–3, Berkeley, CA, USA, 2008. USENIX Association.

[8] Ziliang Zong, Xiao Qin, Xiaojun Ruan, K. Bellam, M. Nijim, and M. Alghamdi. Energy-efficient scheduling for parallel applications running on heterogeneous clusters. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, pages 19–19, Sept 2007.

[9] Joris Slegers, Nigel Thomas, and Isi Mitrani. Dynamic server allocation for power and performance. In Samuel Kounev, Ian Gorton, and Kai Sachs, editors, *Performance Evaluation: Metrics, Models and Benchmarks*, volume 5119 of *Lecture Notes in Computer Science*, pages 247–261. Springer Berlin Heidelberg, 2008.

[10] Release notes for ubuntu 14.04.1. `https://wiki.ubuntu.com/TrustyTahr/ReleaseNotes/`.

[11] Colin Ian King. Ubuntu powerstat package. `https://launchpad.net/ubuntu/+source/powerstat`.

[12] Stephen M. Blackburn, Robin Garner, Chris Hoffmann, Asjad M. Khang, Kathryn S. McKinley, Rotem Bentzur, Amer Diwan, Daniel Feinberg, Daniel Frampton, Samuel Z. Guyer, Martin Hirzel, Antony Hosking, Maria Jump, Han Lee, J. Eliot B. Moss, Aashish Phansalkar, Darko Stefanović, Thomas VanDrunen, Daniel von Dincklage, and Ben Wiedermann. The dacapo benchmarks: Java benchmarking development and analysis. *SIGPLAN Not.*, 41(10):169–190, October 2006.

[13] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*, volume 35(2), pages 13–23. ACM, 2007.

[14] Gianfranco Balbo and Giuseppe Serazzi. Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Performance Evaluation*, 30(3):115–152, 1997.

[15] Emilia Rosti, Francesco Schiavoni, and Giuseppe Serazzi. Queueing network models with two classes of customers. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1997. MASCOTS'97., Proceedings Fifth International Symposium on*, pages 229–234. IEEE, 1997.

[16] Leonard Kleinrock. Power and deterministic rules of thumb for probabilistic problems in computer communications. In *Proceedings of the International Conference on Communications*, volume 43, pages 1–43, 1979.

[17] Marco Bertoli, Giuliano Casale, and Giuseppe Serazzi. JMT: performance engineering tools for system modeling. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):10–15, 2009.