

Profit-Aware Server Allocation for Green Internet Services

Michele Mazzucco^{*†}, Dmytro Dyachuk[†] and Marios Dikaiakos^{*}

^{*}University of Cyprus, Cyprus

[†]University of Saskatchewan, Canada

[‡]University of Tartu, Estonia

Abstract—A server farm is examined, where a number of servers are used to offer a service to impatient customers. Every completed request generates a certain amount of profit, running servers consume electricity for power and cooling, while waiting customers might leave the system before receiving service if they experience excessive delays. A dynamic allocation policy aiming at satisfying the conflicting goals of maximizing the quality of users' experience while minimizing the cost for the provider is introduced and evaluated. The results of several experiments are described, showing that the proposed scheme performs well under different traffic conditions.

I. INTRODUCTION

In recent years large investments have been made to build data centers, or *server farms*, purpose-built facilities providing storage and computing services within and across organizational boundaries. A typical server farm may contain thousands of servers, which require large amounts of power to operate and to keep cool, not to mention the hidden costs associated with data centers' carbon footprint and water consumption for cooling purposes [1]. On the other hand, the increasing use of the Internet as a provider of services and a major information media have changed significantly, especially over the last ten years. Expectations in terms of performance and responsiveness have markedly grown. For example, Google reports that an extra 0.5 seconds in search page generation entails degraded user satisfaction, with a consequent 20% traffic drop, while trimming the page size of Google Maps by 30% resulted in a 30% traffic increase [2], [3]. Hence, the development of 'green' data centers, *i.e.*, data centers that are energy efficient, is a challenging problem for service providers as they operate under stringent performance requirements. Consequently, it is very important to devise strategies aiming at reducing the power consumption while maintaining acceptable levels of performance.

Unfortunately, despite considerable effort in designing servers whose power consumption is proportional to their utilization [4], the reality is that the amount of power consumed by an idle server is about 65% of its peak consumption [5], as existing hardware components offer only limited controls for trading power for performance. Thus, the only way to significantly reduce data centers' power consumption is to improve the server farm's utilization, *e.g.*, by tearing down servers in excess. Therefore, we propose and evaluate a strategy that aims at maximizing the overall performance while minimizing the number of required servers. Under suitable assumptions

about the nature of user demand, it is possible to explicitly evaluate the effect of a particular server allocation on the achievable revenue. Hence, we derive a numerical algorithm for computing the optimal number of servers required for handling a certain user demand. The model considers limited user patience time and the fact that servers energy consumption depends on servers' utilization. The computational costs of the decision making are extremely low and the algorithm can be used on-line as a part of a dynamic allocation policy.

The problem of reducing energy consumption of server farms can be approached from different angles. Improving energy efficiency for servers by means of dynamic scaling of the CPU frequency has been addressed in several papers, *e.g.*, [6], [7]. An alternative solution consists of switching off servers in excess. The most closely related work can perhaps be found in [8] and [9]. The former discusses a queuing model for controlling the energy consumption of service provisioning systems subject to Service Level Agreements. However, while Chen et al. take into account the cost for smaller mean time between failures (MTBF) when powering up/down some servers, the cost function they propose does not consider the time and energy wasted during state changes, nor the cost for failing to meet the promised quality requirements. Hence, the taken decisions could be either too performance oriented or too energy-efficiency oriented. [9], instead, discusses a problem similar to that we attack in this paper. However, in that paper the authors assume that clients have no patience, while they do not consider the fact that servers consume energy without producing any revenue during system reconfigurations. Finally, since running too many servers increases the electricity consumption while having too few servers switched on requires running those servers' CPUs at higher frequencies, some hybrid approaches have been proposed, *e.g.*, [6].

The rest of the paper is structured in the following way. In the next section we introduce the system model. Section III contains the mathematical analysis. The model for power consumption estimation is discussed in Section IV, while the resulting policy is introduced in Section V. A number of experiments are presented in Section VI. Finally, we conclude the paper in Section VII with a summary and some remarks.

II. THE MODEL

A server farm is a collection of servers interconnected by high-speed, switched LANs that hosts content and runs

applications (or services) accessed over the Internet. In this paper we focus on server farms designed according to the dedicated architecture, where a web application is hosted on a set of physical servers (see, for example, [10]) and the provider can change the number of servers allocated to run each service in order to react to traffic changes. Once a decision about how to partition the available servers has been made, it is possible to treat each subsystem (*i.e.*, service) in isolation of each other. Therefore, in the following we tackle the problem of maximizing the revenue of a single subsystem (service) only.

Now, assume that q (identical) servers have been allocated to queue i . Among those q servers, h are running and capable of serving incoming user demand (*jobs*, from now on), while the remaining $(q - h)$ are switched off in order to save energy. In this context, the term ‘switched off’ means that the server can not perform any useful work and it does not consume any electricity. A maximum of m jobs can be processed in parallel on each server without significant interference – this limit being imposed by the number of available threads or processes. This is modelled by assuming that there are m parallel servers on each physical machine or core, and thus a total of $S = qm$ servers are available, while $n = hm$ are running. If n jobs are currently in execution, further requests are temporarily parked into an external first-in-first-out (FIFO) queue whose size is assumed to be infinite. If, once a server has finished processing a request, the queue is empty, the server begins to idle (*i.e.*, it consumes energy without generating any profit). Otherwise, it removes the leading job from the waiting queue and starts processing it. Every processed request generates some profit. For example, it can be a profit coming from advertisements or from sales (in case of online merchants, such as Amazon). While in the first case advertising agencies usually pay for each impression (*i.e.*, display of an add), in the second case the profit from each request can be estimated as follows. Suppose that every 10,000 views generate 10\$ of revenue from sales. Hence, we can state that on average each request brings 0.01 cents. It is worth stressing that companies like eBay employ more complex revenue models. However, using simple transformations such as dividing gross income over the number of requests, one can easily estimate the average profit brought by every request.

Since running servers consume electricity, which costs r \$ per kWh, the provider dynamically decides how many servers to run by means of a ‘resource allocation’ policy. The objective is to find the optimal number of servers, n , that should be switched on in order to optimize the provider’s profit. The extreme values, $n = 0$ and $n = S$, correspond to switch respectively off, or on, all available servers. Given that the amount of running servers should change in response to changes in the user demand, the problem is how to estimate the best n . We assume that data is widely replicated, hence switching some servers off does not affect service availability. However, since lost jobs do not generate any revenue, the provider should ensure that the time users wait for their requests to be served does not exceed their patience. Otherwise

the clients will start aborting their requests, *e.g.*, by clicking a Stop button in their browsers, see Figure 1.

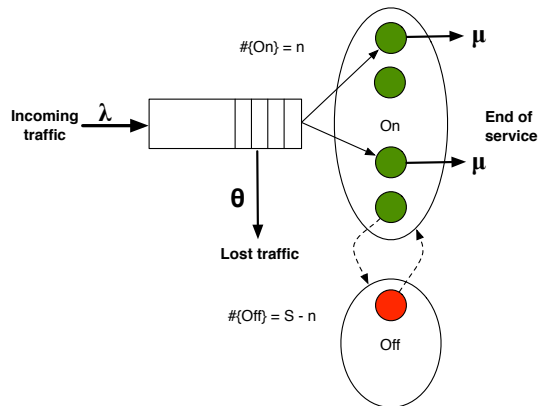


Fig. 1. System model. Jobs whose average job size is $1/\mu$ enter the system at rate λ and abandon the system at rate θ while waiting.

During the intervals between consecutive policy invocations, the number of running servers remains constant. Those intervals, which will be referred to as ‘observation windows’, are used by the controlling software to collect the traffic statistics used by the allocation policy at the next decision epoch.

While different metrics can be used to measure the performance of a computing system, as far as the service provider is concerned, the performance of the server farm is measured by the average revenue, R , earned per unit time. That value can be estimated as

$$R = cT - rP, \quad (1)$$

where c is the income generated by each completed job, T is the system’s throughput, and P is the total average power consumed by powered up servers. c is a parameter of the model, while the formula for computing T and P are discussed in Sections III and IV. For the following it will be convenient to indicate explicitly the dependency of Equation (1) on the parameter n by introducing the notation

$$R = r(n), \quad (2)$$

where $r(n)$ stands for the right-hand side of (1).

It is worth noting that although we do not make any assumption about the relative magnitudes of charge and cost parameters, the most challenging case is when they are close to each other. If the charge for executing a job is much higher than the provisioning cost, one could guarantee a positive, but not optimal, revenue by switching on all servers, regardless of the load. On the other hand, if the charge is smaller than the cost, than it would be better to switch all servers off.

III. ANALYSIS

Suppose that n servers have been allocated to serve incoming traffic. Jobs enter the system according to an independent Poisson process with rate λ . Operating servers accept one job at a time, with the service times, or ‘job size’, being

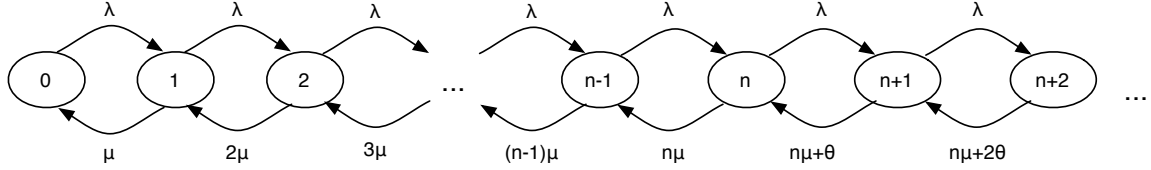


Fig. 2. State transition diagram.

exponentially distributed with mean $1/\mu$. Hence, one might try to model the system as an $M/M/n$ queue (see [11] for more details) with the offered load being $\rho = \lambda/\mu$ and with the stability condition being $\rho < n$. Unfortunately, this system, also called Erlang-C (or delay), does not acknowledge abandonment; thus it either distorts or completely fails to provide important information.

In this work we do allow customers abandonment by assuming that a time-out policy is in operation: if a job entering the system does not acquire a server before its time-out period expires, the job is terminated and leaves the system without generating any revenue. This can be used to model HTTP time-outs, as well as impatient customers. Impatient customers are of particular importance, as [12] reports that 75% of people would not go back to a web site that took more than 4 seconds to load. HTTP time-outs are in practice of fixed length, while users' patience is not. For the purposes of analytical tractability, we assume that both the user's patience and HTTP time-outs are i.i.d random variables distributed exponentially with mean $1/\theta$, with θ being referred to as the *abandonment rate*. The extreme values, $\theta = 0$ and $\theta = \infty$, correspond to jobs with no, or infinite patience. Hence, the appropriate queueing models would be $M/M/n/n$ and $M/M/n$ respectively [11]. Also, we assume that the patience variables are independent of all other model elements, namely arrival and service rates.

The first step to computing Equation (1) is the steady state analysis of the Markov chain associated with the model sketched in Figure 1, see Figure 2. A tractable way to model this system is the $M/M/n+M$ queue, also known as Erlang-A (where the 'A' stands for 'Abandonment') [13]. The main difference between the Erlang-A and Erlang-C models is that the queue never grows unbound in the Erlang-A model, as jobs are allowed to leave. Moreover, jobs abandonment reduces workload only when needed, *i.e.*, when the load is high. The implication is that fewer servers are needed to guarantee the same level of performance under Erlang-A, compared to the traditional Erlang-C. This observation is very important because in this paper we focus on highly loaded (and potentially overloaded) systems, as our goal is to switch off servers in excess while serving as many customers as possible.

Now, let L_t be the total number of jobs inside the system at time t , including both queueing and executing requests. Then, $L = \{L_t : t \geq 0\}$ is a state-dependent *Birth-and-Death* process [11]. The instantaneous transition rate from state j to state $(j+1)$ is equal to the arrival rate, $\lambda_j = \lambda$, $j = 0, 1, \dots$. The conditional departure rate, *i.e.*, the transition rate from state j to state $(j-1)$, depends instead on the number of operating servers as well as on the number of jobs present.

Hence, we should distinguish between two cases:

Case 1: $j \leq n$. The system behaves like an $M/M/\infty$ queue: all jobs in the system are being served without queueing, jobs leave the system at rate $\mu_j = j\mu$, and $(n-j)$ servers are idle.

Case 2: $j > n$. All servers are busy and $(j-n)$ jobs are queueing. The instantaneous completion rate does not depend on j anymore, while the abandonment rate depends on the current number of jobs in the queue.

Hence, the balance equations can be expressed in terms of p_0 , *i.e.*, the probability of the system being empty

$$p_j = \begin{cases} \frac{\rho^j}{j!} p_0 & \text{if } j \leq n \\ \frac{\rho^n}{n!} p_0 \left[\prod_{k=n+1}^j \frac{\lambda}{n\mu + \theta(k-n)} \right] & \text{if } j > n \end{cases} \quad (3)$$

The only unknown probability is p_0 . Steady state for this process exists only if Equation (3) can be normalized, *i.e.*, if $\sum_{j=0}^{\infty} p_j = 1$. From the normalization condition and from Equation (3) we obtain

$$p_0 = \left[\sum_{j=0}^n \frac{\rho^j}{j!} + \frac{\rho^n}{n!} \sum_{j=n+1}^{\infty} \prod_{k=n+1}^j \left(\frac{\lambda}{n\mu + \theta(k-n)} \right) \right]^{-1} \quad (4)$$

Thus, steady state for this Birth-and-Death process always exists, as jobs in excess eventually abandon, ensuring that the queue never grows unbound. To handle the series in Equation (4) it is convenient to introduce the function [13]

$$g(x, y) = 1 + \sum_{j=1}^{\infty} \frac{y^j}{\prod_{k=1}^j (x+k)} \quad (5)$$

Thus, by means of Equation (5) and the formula for computing the blocking probability in an Erlang-B system with n trunks and traffic intensity ρ , $B(n, \rho)$ [11], [14], after some algebraic manipulations we obtain

$$p_0 = \frac{n!}{\rho^n} \frac{B(n, \rho)}{1 + B(n, \rho) \left[g\left(\frac{n\mu}{\theta}, \frac{\lambda}{\theta}\right) - 1 \right]} \quad (6)$$

Having computed the steady state probability of the number of jobs present, it is possible to estimate the probability of abandonment, $P(Ab)$. Denote by $P_j(S)$ the probability that a job finding j other jobs inside the system will eventually get

served. Hence, the probability of abandonment of a job finding j other jobs ahead of it is simply $1 - P_j(S)$. Therefore, we can write

$$P_j(Ab) = \begin{cases} 0 & \text{if } j < n \\ \frac{(j+1-n)\theta}{n\mu + (j+1-n)\theta} & \text{if } j \geq n \end{cases} \quad (7)$$

Using Equation (7) and the PASTA property (Poisson arrivals see time averages, see [11] for more details), $P(Ab)$ yields to

$$P(Ab) = \sum_{j=0}^{\infty} p_j P_{j-n}(Ab). \quad (8)$$

Computing Equation (8) from its right-hand side is challenging. However, using Bayes' formula [15] we obtain

$$P(Ab) = P(W > 0)P(Ab|W > 0), \quad (9)$$

where $P(W > 0)$ is the delay probability, while $P(Ab|W > 0)$ is the conditional abandonment probability.

In order to compute the above probabilities it is useful to express the balance equations in terms of p_n , *i.e.*, the probability that all servers are busy and the queue is empty. Using Equations (3) and (6), p_n yields to

$$p_n = \frac{B(n, \rho)}{1 + B(n, \rho) \left[g \left(\frac{n\mu}{\theta}, \frac{\lambda}{\theta} \right) - 1 \right]}. \quad (10)$$

Hence, Equation (3) can now be written as

$$p_j = \begin{cases} \frac{n!}{j! \rho^{n-j}} p_n & \text{if } j \leq n \\ \frac{(\lambda/\theta)^{j-n}}{j-n} p_n & \text{if } j > n \\ \prod_{k=1}^n \left(\frac{n\mu}{\theta} + k \right) & \end{cases} \quad (11)$$

By means of Equation (11) and the PASTA property, it is now possible to estimate the delay probability

$$P(W > 0) = \sum_{j=n}^{\infty} p_j = p_n + \sum_{j=n+1}^{\infty} p_j. \quad (12)$$

Thus, using Equations (5), (11) and (12) we obtain

$$P(W > 0) = A \left(\frac{n\mu}{\theta}, \frac{\lambda}{\theta} \right) p_n, \quad (13)$$

while, after some manipulations, $P(Ab|W > 0)$ yields to

$$P(Ab|W > 0) = \frac{1}{\rho g \left(\frac{n\mu}{\theta}, \frac{\lambda}{\theta} \right)} + 1 - \frac{1}{\rho}. \quad (14)$$

N.B. As the size of the server farms grows, the system achieves economies of scale that make it more robust against traffic variability. Hence, while violating the Markovian assumptions about the arrival, patience and service processes

affects the average queue length, it does not substantially change the abandonment rate [16].

Having computed the stationary distribution of jobs present and the corresponding probability of abandonment, we can now compute the average number of requests served per unit time. That value can be expressed as

$$T = \min(n\mu, \lambda[1 - P(Ab)]). \quad (15)$$

N.B. The above expression successfully deals with two special cases, *i.e.*, $\theta = 0$, and thus the system would behave as an $M/M/n$ queue, and $\rho > n$.

IV. POWER USAGE ESTIMATION

A number of factors affect the amount of energy consumed by a server. However, the change in the power consumption of a server is mainly due to changes in the CPU utilization. In order to establish and quantify this relation we have measured the amount of energy drawn by a server in the presence of an increasing load. As for an application we have used Wordpress¹, a popular open source application implementing a blog and running on top of the LAMP² stack. The server had two Xeon Dual Core CPUs (2.8 Ghz) equipped with 2 Gb of RAM, 7200 RPM hard drive and 1 Gbps network card, while user demand was generated by Tsung³. The workload consisted of clients arriving over time, with each client simulating a typical behavior of a blog reader, such as checking the front page, navigating through the blog, searching for entries containing certain keywords, etc. The load was increased by reducing the interarrival intervals, which were generated according to an independent Poisson process. The measurements reported in

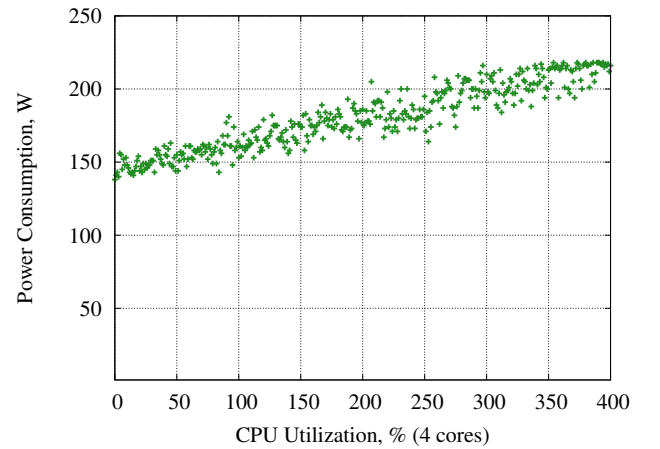


Fig. 3. Measured energy consumption.

Figure 3 show a linear dependency between the CPU utilization and power consumption. The experiment also confirms that idle servers consume a substantial amount of energy (140

¹<http://wordpress.org/>.

²Linux, Apache, MySQL and PHP.

³<http://tsung.erlang-projects.org/>.

Watts in our case). Hence, the average power consumed by a data center per unit time, P , can be estimated as

$$P = ne_1 + \tau(e_2 - e_1), \quad (16)$$

where e_1 is the energy consumed per unit time by idle servers, e_2 is the energy drawn by each busy server, and τ is the occupancy of the system ($\tau \leq n$)

$$\tau = \left\lceil \frac{T}{\mu} \right\rceil. \quad (17)$$

V. ALLOCATION POLICY

Consider now an allocation decision epoch. The state of the subsystem at that instant is defined by the number of servers which are not powered down and by the potential offered load. If the allocation does not change, the expected revenue for the next configuration interval is simply $r(n)$. The alternative is to power on/off some servers. Denote by n' the new number of servers allocated to the queue after reallocation:

Case 1: $n' > n$. Such a decision would increase the system throughput, thus increasing the potential revenue, but it would also increase the amount of energy consumed by the server farm.

Case 2: $n' < n$. Such a decision would decrease the expenditure for electricity, as $(n - n')$ servers would be switched off. However, it would also decrease the overall throughput, thus increasing the probability of jobs abandonment.

Powering servers on/off is not instantaneous, but takes on average k units time. Hence, at every state change there is some waste of time and money because servers changing their state can not serve user demand, but do consume electricity. Also, system's reliability is affected by state changes, as hardware components tend to degrade faster with frequent power on/off cycles than with continuous operation. For example, hard disks have an average lifetime of 40/50,000 on/off cycles [17]. Therefore, each state change involves the following cost

$$Q = \frac{|\Delta n|}{t} \left(\sum_{i=1}^l d_i + kre_{max} \right), \quad (18)$$

where t is the length of the observation windows, $|\Delta n|$ is the number of servers that are switched on/off, e_{max} is the power consumed per unit time during state changes, k is the average time required to switch a server on/off, d_i is the cost for a hardware component's state change, and l is the number of components.

N.B. One can easily relax the assumption that powering a server up and down takes the same amount of time and consume the same amount of energy.

The expected *change* in revenue resulting from a decision to change the number of running servers can be expressed as

$$\Delta r(n', n) = r(n') - r(n) - Q. \quad (19)$$

When R is computed for different values of n , it becomes clear that the revenue is a unimodal function of n , *i.e.*, it

has a single maximum. That observation implies that one can search for the optimal number of servers to run by evaluating R for consecutive values of n , stopping either when R starts decreasing or, if that does not happen, when the increase becomes smaller than some value ϵ . This can be justified arguing that R is a concave function with respect to n . Intuitively, the economic benefits of powering more servers on become less and less significant as n increases, while the loss of potential revenues gets bigger and bigger as n decreases. Such a behavior is an indication of concavity. Thus, a fast algorithm of the *binary search* variety suggested by the above observations works as follows:

- 1) Set with $n_l = 0$ and $n_u = S$;
- 2) Set $n' = \lceil \lambda/\mu \rceil$. If $n' = 0$, set $n' = 1$. Similarly, if $n' = S$, set $n' = S - 1$.
- 3) While $n_l < n_u$
 - a) Calculate $\Delta r(n' - 1, n)$, $\Delta r(n', n)$ and $\Delta r(n' + 1, n)$.
 - b) If $\Delta r(n' - 1, n) \leq \Delta r(n', n) \geq \Delta r(n' + 1, n)$, then n' is the best solution. Hence, go to 4.
 - c) If $\Delta r(n' - 1, n) \leq \Delta r(n', n) \leq \Delta r(n' + 1, n)$, then the optimal n is in the interval n', \dots, n_u . Hence, set $n_l = n' + 1$ and go to 3e.
 - d) We have $\Delta r(n' - 1, n) \geq \Delta r(n', n) \geq \Delta r(n' + 1, n)$. Hence, the search has to be carried out in the interval n_l, \dots, n' , so set $n_u = n' - 1$ and go to 3e.
 - e) Set $n' = \lceil (n_l + (n_u - n_l)/2) \rceil$.
- 4) If $\Delta r(n', n) > 0$, then set $n = n'$. Otherwise, leave the allocation as it is .

Since at every iteration the state space is reduced by a factor of two, $\log(S)$ iterations are required in order to find the 'best' n , *i.e.*, the number of servers that maximizes the profit. We have put quotation marks around the word 'best' because such choice might be slightly sub-optimal when the exponential assumptions are violated and S is small (see the remarks at the end of Section III). This policy will be referred to as the 'Adaptive' allocation policy.

Finally, since the state change is not instantaneous, the current window ends immediately if some of the servers are to be switched off (*e.g.*, if $n' < n$) or after all the servers are powered up if $n' > n$.

VI. RESULTS

Several experiments were carried out, with the aim of evaluating the effects of proposed scheme on the maximum achievable revenues. We assume the server farm has a Power Usage Effectiveness (PUE), the main metric used to evaluate the efficiency of data centers, of 1.7. That value is computed as the ratio between the total facility power and the IT equipment power. Also, we take indirect costs into account. These include the cost for capital as well as the amortization of the equipment such as servers, power generators or transformers, and account for twice the cost of consumed electricity. Finally, in order to reduce the number of variables, when not specified otherwise, the following features were held fixed:

- 250 servers, configured as described in Section IV, *i.e.*, $S = 1,000$.
- The power consumption of each four core server machine ranges between 140 and 220 W, see Figure 3. In other words, each core from now on server consumes between 35 and 55 W. Since the server farm has a PUE factor of 1.7, the minimum and maximum power consumption are approximately $e_1 = 59$ and $e_2 = 94$ W per server.
- The cost for electricity, r , is 0.1 \$ per kWh⁴.
- The average job size, $1/\mu$, is 0.1 seconds.
- Jobs are not completely CPU bound. Instead, when a server is busy, the average CPU utilization is 70%. In other words, busy servers draw 69.58 Wh, and thus each job costs, for electricity, 2×10^{-7} \$ on average.
- Each successfully completed job generates, on average, 6.2×10^{-6} \$.

It is worth noting that while the size of the server farm might look small, the application logic of the 10th busiest web site in the world, Wikipedia, is hosted on 350 servers having 856 cores, spread across three data centers [18].

A. Stationary Traffic

The first experiment, see Figure 4, is purely numerical. It shows how the number of running servers affects the average earned revenue under different loading conditions. The potential offered load is increased from 30% to 90% by increasing the rate at which new jobs enter the system, from 3,000 to 9,000 jobs per second.

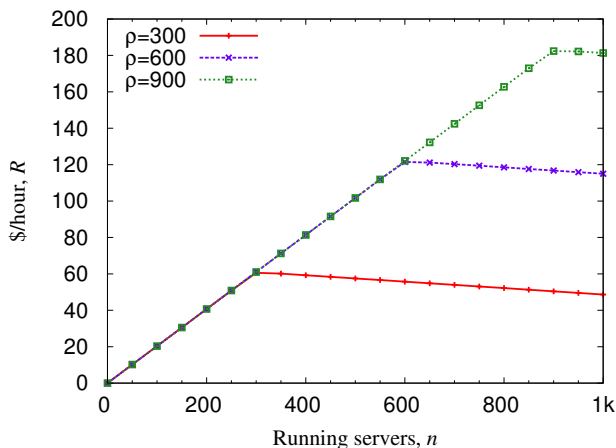


Fig. 4. Revenue as a function of the running servers.

Figure 4 shows that (i) in each case there is an optimal number of servers that should be switched on; (ii) the heavier is the load, the higher is the optimal number of servers to run, but higher is also the maximum achievable revenue; (iii) when $n > n_{opt}$, the system under-performs because the cost of running idle servers erodes revenues, while when $n < n_{opt}$, the system under-performs because it misses potential revenues.

⁴<http://www.neo.ne.gov/statshhtml/115.htm>.

Next, we evaluate the effectiveness of the dynamic allocation scheme via computer simulation. For comparison reasons, two versions of the ‘Static’ policy, a policy which runs always the same amount of servers, are also displayed. One runs $n = S/2 = 500$ servers, while the other $n = S = 1,000$. We vary the load between 10% and 110% (*i.e.*, the system would be over-saturated without job abandonment) by varying the arrival rate, *i.e.*, $\lambda = 1,000, \dots, 11,000$ jobs/second. Each point in the figure represents one run lasting 16.5 hours, while servers are reallocated every hour, *e.g.*, the policy described in Section V is invoked every hour. During each run, approximately between 119 (low load) and 653 (high load) million jobs enter the system, while samples of achieved revenues are collected every 1.5 hours and are used at the end of each run to compute the corresponding 95% confidence interval, which is calculated using the Student’s t-distribution.

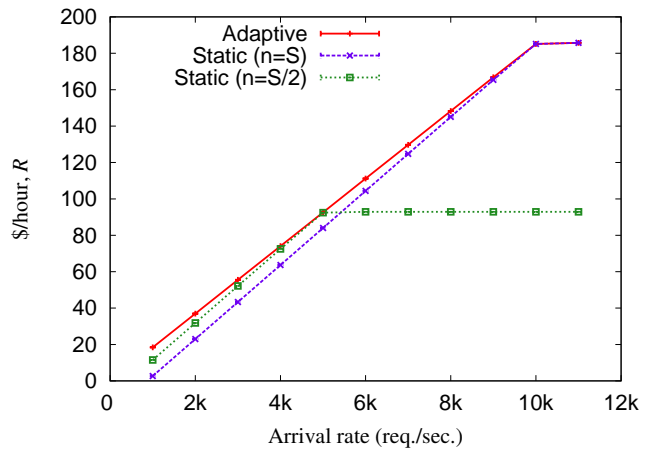


Fig. 5. Observed revenues for different policies.

The most notable feature of the graph plotted in Figure 5 is that the ‘Static’ policies do not perform well under light load (because of the servers running idle), while the one with parameter $n = S/2$ can not cope with high traffic, thus missing income opportunities. On the other hand, the ‘Adaptive’ heuristic produces revenues that grow with the offered load. Next, we report the average power consumption. Figure 6 shows that the ‘Adaptive’ heuristic runs servers only when necessary, thus reducing its carbon footprint as well as the provider’s electricity bill.

In the next experiment we evaluate the effects of increased variability on the performance of the ‘Adaptive’ heuristic by departing from the assumption that the interarrival and patience times are exponentially distributed. Now those parameters are Log-Normally distributed, with the squared coefficient of variation, *i.e.*, the variance divided by the square of the mean, of 2 (interarrival intervals on YouTube have a squared coefficient of variation of 1.7 [19]) and 5, respectively. These changes increase the variances of the interarrival and patience times while preserving the averages, thus making the system less predictable and allocation decisions more challenging.

In Table I we compare the achieved revenue (R), average

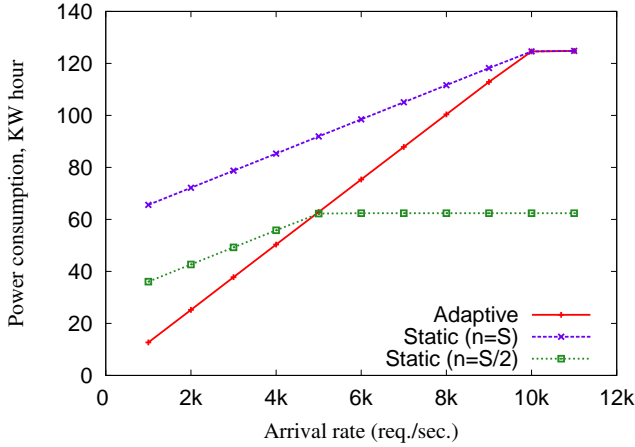


Fig. 6. Observed energy consumption for different policies.

S	$scv(\lambda) = scv(\theta) = 1$			$scv(\lambda) = 2, scv(\theta) = 5$		
	R	L	% Ab	R	L	% Ab
10	1.45	4.636	1.136	1.44	6.216	1.398
20	2.91	4.602	0.572	2.91	7.040	0.614
50	7.35	11.034	0.547	7.35	17.374	0.514
100	14.75	14.371	0.359	14.76	24.007	0.268
500	74.06	28.467	0.142	74.11	53.349	0.055
1000	148.26	40.651	0.101	148.38	82.370	0.033

TABLE I
PERFORMANCE OF THE ‘ADAPTIVE’ HEURISTIC FOR DIFFERENT PARAMETERS.

queue length (L) and the percentage of jobs abandoning the system under the ‘Adaptive’ policy for different values of S . The system is always loaded at 80% by scaling the arrival rate in proportion to S . From the experiment we observe that while L depends on the variability of the traffic parameters, the achieved revenue and the percentage of jobs leaving do not. Moreover, the system becomes less and less sensible to the distribution of the interarrival intervals and patience times as S increases. This can be explained by observing that large systems can achieve economies of scale which are simply not possible when the number of servers is small. In particular, the behavior of large server farms under heavy load differs from that of Kingman’s Law (*i.e.*, delays/job losses are very common under heavy load) in that service quality is carefully balanced with server efficiency.

B. Sensitivity Analysis

The previous experiments investigated how server allocation decisions and other parameters can affect the revenue. However, in real world scenarios the chances that service providers would have to deal with stationary traffic are extremely rare. In fact, studies of the Wikipedia traces show that throughout the day the incoming traffic can change by as much as 70% [21]. A logical solution to this problem would be reconfiguring the servers pool (by switching servers on/off) according the changes in the load, as we propose in Section V.

However, service providers need to estimate the arrival rate for the next configuration interval. Such a prediction might be hard to make if the load is non stationary, and even good forecasting tools might produce results which sometimes differ from the observed values. Forecasting tools range from simply using the last observed value of λ to very complex algorithms requiring significant periods of time for training and considering seasonal and trend components. Instead of benchmarking the proposed approach with various forecasting tools, which would lead to an explosion of the experimental space, in the following we investigate its sensitivity in respect to an error in load prediction. This also can help in choosing the right prediction mechanism in real-world scenarios, as the deployment of certain forecasters may require long training, while simpler ones might exhibit slightly worse prediction quality, while having little or no effect on the ultimate result.

In order to see how various error rates affect the revenue, we introduce an Oracle forecaster which tells the exact arrival rate for the next configuration interval. Then we introduce an error, first 5%, then 10% and finally 20%. Please note that the forecasting values are distributed according to a Laplace distribution with the mean value equal to the actual λ . Thus, for example 5% represents the mean of the absolute differences between actual and predicted values of λ . We have chosen a Laplace distribution because we have observed such a behavior when trying to predict the Wikipedia workload using the double exponential smoothing method. Unfortunately, due to the lack of space we cannot present this data. Also, it is important to note that by combining Oracle with the binary search from section V we obtain the optimal solution in terms of profit maximization.

In the next set of experiments we contrast five different cases: static policy (all servers are switched on), Adaptive heuristic (see Section V) with Oracle forecasting, and Adaptive heuristic with Oracle forecasting making a systematic error of 5%, 10% and 20%. In the simulation, a workload which represents a scaled version of the Wikipedia traces is employed. The arrival rate ranges between 2,688 and 5,729 jobs/second, while each simulation run lasts 240 hours (*e.g.*, 10 days).

TABLE II
PERCENTAGE OF LOST JOBS WITH RESPECT TO ERROR IN THE PREDICTION

Error	0%	5%	10%	20%
Lost jobs, %	0.01	2.19	4.46	10.54

As shown in Table II, the error in the forecasting has a significant impact on the number of lost jobs. The Oracle behaves almost as good as the static policy which massively over-provisions the system, which in its turn negatively reflects on the amount of consumed energy, see Figure 7, and consequently the revenue, see Figure 8.

At the same time 5% error in the prediction affects the number of lost jobs without significant impact on the other metrics. An increase of the error from 5% to 10% markedly reflects on the number of lost jobs, while the revenue values still stay very close. A jump of the error from 10% to 20%

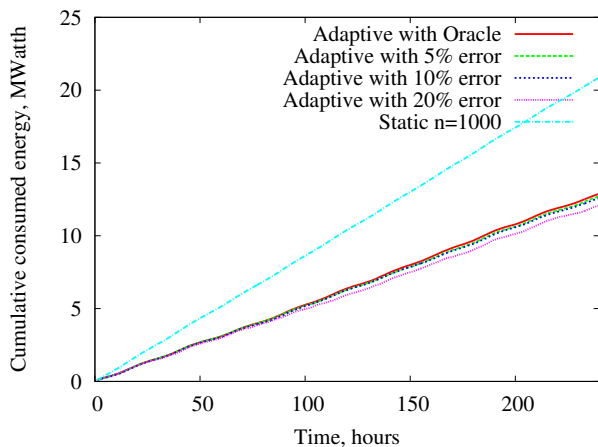


Fig. 7. Cumulative energy consumption.

not only increases the number of lost jobs more than twice due to frequent mistakes causing under-provisioning and thus adversely affecting the revenue. Despite the obvious advantage in the power consumption, using forecasting with 20% error is unacceptable due to its low revenue and high percentage of the lost jobs. From the above experiment we can conclude that 5% error in forecasting does not have significant impact on the revenue and energy consumption, while a 10% error still exhibits results which are markedly superior to the Static allocation policy.

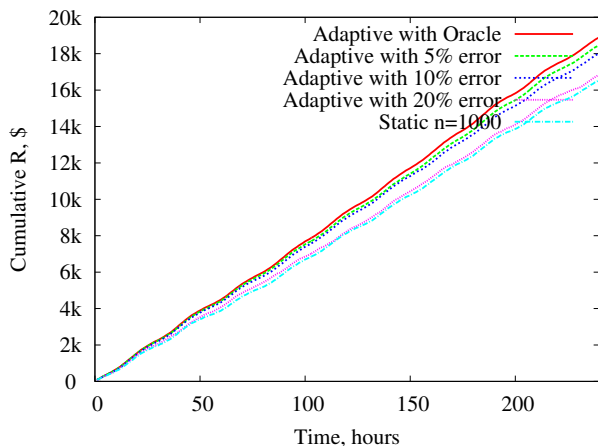


Fig. 8. Cumulative revenue.

VII. CONCLUSIONS

In this paper we have introduced and evaluated an easily implementable policy for dynamically adaptable Internet services. Under some simplifying assumptions, the numerical algorithm we propose can find the best trade off between consumed power and delivered service quality. We have demonstrated that the number of running servers can have a significant effect on the revenue earned by the provider. The experiments we have conducted show that our approach works

well under different traffic conditions, and that our policy is not very sensitive to errors in parameters estimation.

Possible directions for future research include taking into account the trade offs between the number of running servers, the frequency of the CPUs and the maximum achievable performance, as well as fault tolerance issues.

ACKNOWLEDGEMENTS

The authors would like to thank the European Commission (Marie Curie Action, contract number FP6-042467), the EU Cost Action IC0804, and the EUREKA Project 4989 (SITIO).

REFERENCES

- [1] J. Hamilton, "Data Center Efficiency Best Practices," Amazon Web Services, April 2009. [Online]. Available: www.mvdirona.com/jrh/TalksAndPapers/JamesHamilton_Google2009.pdf
- [2] G. Linden, "Marissa mayer at web 2.0 - <http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html>," November 2006. [Online]. Available: <http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html>
- [3] S. Shankland, "We're all guinea pigs in Google's search experiment," May 2008. [Online]. Available: http://news.cnet.com/8301-10784_3-9954972-7.html
- [4] L. A. Barroso and U. Hözl, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.
- [5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, January 2009.
- [6] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *In Proceedings of the 2nd Workshop on Power-Aware Computing Systems*, 2002, pp. 179–196.
- [7] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu, "Power-aware qos management in web servers," in *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS '03)*, 2003, p. 63.
- [8] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," *SIGMETRICS Perform. Eval. Rev.*, vol. 33, no. 1, pp. 303–314, 2005.
- [9] M. Mazzucco, D. Dyachuk, and R. Deters, "Maximizing Cloud Providers Revenues via Energy Aware Allocation Policies," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing (IEEE Cloud 2010)*, July 2010.
- [10] M. Mazzucco, "Towards Autonomic Service Provisioning Systems," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010)*, May 2010.
- [11] I. Mittrani, *Probabilistic Modelling*. Cambridge University Press, 1998.
- [12] J. McGovern, "Selfish, Mean, Impatient Customers," July 2008. [Online]. Available: <http://www.cmswire.com/cms/web-content/selfish-mean-impatient-customers-002891.php>
- [13] C. Palm, "Research on Telephone Traffic Carried by Full Availability Groups," *Tele (English Edition)*, vol. 1, 1957.
- [14] O. Hudousek, "Evaluation of the Erlang-B formula," in *Proceedings of RTT 2003*, 2003.
- [15] S. M. Ross, *Introduction to Probability Models*. Academic Press, 2000.
- [16] W. Whitt, "Efficiency-Driven Heavy-Traffic Approximations for Many-Server Queues with Abandonments," *Management Science*, October 2004.
- [17] P. M. Greenawald, "Modeling Power Management for Hard Disks," in *In Proceedings of 2nd IEEE MASCOTS*, 1994, pp. 62–66.
- [18] M. Bergsma, "Wikimedia Architecture," Wikimedia Foundation Inc., 2007. [Online]. Available: <http://www.networks.org/~mark/presentations/san/Wikimedia%20architecture.pdf>
- [19] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07)*, 2007, pp. 15–28.
- [20] W. Whitt, *Stochastic-Process Limits*. Springer-Verlag, 2002.
- [21] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Comput. Netw.*, vol. 53, no. 11, pp. 1830–1845, 2009.