Trade-Offs Between Energy and Quality of Service

Erol Gelenbe and Ricardo Lent
Intelligent Systems & Networks Group, EEE Dept.
Imperial College, London SW7 2BT, UK
{e.gelenbe, r.lent}@imperial.ac.uk

***Abstract*—Energy consumption in ICT is growing annually by 4% despite efficiency gains in technology, and it already has a carbon impact that is equivalent to air travel. It is therefore important to study ways to reduce energy consumption in ICT while preserving acceptable levels of quality of service (QoS). Energy consumption in computer systems is often related to the operating load and will be increasing as systems become more energy efficient. We examine the choice of system load that offers the best trade-off between energy consumption and QoS, and use measurements to validate the results.**

***Index Terms*—System Load; Measurements; QoS-Energy Trade-Off**

I. Motivation and Problem Setting

The simplest means of energy savings in ICT is to turn off computers and network units that are not being used, provided one can restart them rapidly when requests for computation or communication services do arrive. In the ideal case, a service system would only be consuming energy during its busy periods. However, turning a system on and off will in itself consume energy, as we will see in the sequel. Thus this paper attempts to take a holistic view so as to find the best system operating point with regard to a composite metric that includes both energy consumption and QoS. For a system with a load factor (or processor utilisation) of $\rho = \lambda E[S]$, where λ is the average arrival rate of jobs and $E[S]$ is the average service time, the average power consumption in watts, under ideal operation when energy is only used when jobs are being processed, is $\Pi = \omega\rho$ and the energy consumption per job in Joules would then be $J_{job} = \Pi/\lambda = \omega E[S]$. However, it is not easy to wake up a server instantaneously as soon as a job arrives for processing, and then to turn it off right after the processing ends. There will always be some energy consumption even if the processor is idle, and both putting the system to sleep and waking it up will cause additional energy expenditure.

Thus we first consider the case when the processor is constantly on, and use a cost function that includes both the response time to jobs, and the energy that is consumed per job on average. We obtain the optimum value of load which minimises the cost function. We will also use measurements on a system with a synthetic workload to estimate the power consumption parameters of the system, the average processing time per job, and we validate the theoretical results regarding the optimum load that minimises the cost function. Then we study the case when multiple such systems are being operated in parallel and we need to share a flow of jobs to the system so as to optimise a composite cost function similar to the previous one, and derive the optimum share of load that must be assigned to each of the sub-systems. We also consider a system which can be tuned off and on intermittently with a specific cost in power consumption associated with the each off-on operation and with no power consumed when the system is off. We again derive the value of the system load which minimises the composite cost function that includes both average response time and energy consumed per job.

II. Combining Energy and QoS

The simple formulae given above do not take into account the fact that the power consumption will depend on the load [2], and that putting a server to sleep and waking it up will take time and consume additional energy. Without taking into account the power needs of complex cooling equipment that is needed for large systems, a simple but fairly realistic power consumption relation for current processing units is $\Pi = A + B\rho$, where A is the power consumption of the processing unit when it is idle. A very efficient processor might have a very small value of A, and B would correspond to the rate of increase in power consumption as more more cores are turned on as the load increases. Unfortunately, for much of the current equipment A is still a significant part (often more than 50%) of the total processor power consumption when it is idle. This includes the fact that the memory system and the peripheral equipment and network connections need to be powered even when no jobs are being processed, and that the operating system can remain active (and hence contributes to the energy consumption) even when there are no external jobs that need to be processed. From Π we obtain an expression for the energy consumption per job:

$$J_{job} = \frac{A}{\lambda} + BE[S], \tag{1}$$

which would justify the principle of concentrating computation on a small number of processing units in order to minimise the power consumption per job. However if one also wishes to consider the resulting quality of service (QoS) then it would be reasonable to examine the simple

978-3-901882-46-3

cost function:

$$C_{job} = aE[S]\left[1+\frac{\rho(1+C_S^2)}{2(1-\rho)}\right]+bJ_{job}, \quad (2)$$
$$= aE[S]\left[1+\frac{\rho(1+C_S^2)}{2(1-\rho)}\right]+\frac{bA}{\lambda}+bBE[S],$$

where a and b are the relative importance that is being placed on the QoS for a job, and the energy consumption per job, and the QoS represented in (3) is the average response time computed from the well known Pollaczek-Khintchine formula in (for instance) [4] per job, assuming Poisson arrivals and general (experimentally measured) service times in a single server queue.

A. Experimental Validation

To validate the energy-QoS metric and optimum load model, we conducted a series of experiments using jobs executing on a server class system having a quad-core Intel Xeon 3430 (8M cache, 2.4 GHz), 2 GB RAM, single 150 GB SATA hard drive, and 2 on-board Gigabit Ethernet interfaces. The system runs Linux (Ubuntu) with CPU throttling enabled with the *on demand governor*, which dynamically adjust the cores' frequency depending on load. A client machine is attached to the server through a fast Ethernet switch to generate the workload, and the client machine also measures the system's power consumption.

We measured power consumption when it is idle, i.e. when it has no external jobs to execute, to be $A = 69.5$ Watts, which corresponds to the value of A in (1). We then launched a sequence of synthetic jobs, where each job consisted in calculating the real number π, using Machin's formula, to a desired level of precision. This level of precision was used to provide a wide range in the execution time and workload from one job to the next by choosing the precision at random with a uniform distribution in the range of 10 to 50 thousand digits. The job also included sending the results back to the client through the network connection. By recording the start and completion times of each job at the server, we measured the *average* job processing time to be 6.4235s, exclusive on any waiting or queueing times at the server. We varied the job arrival rate λ and the measured average response time for a job is shown in Figure 1 as a function of load (ρ), together with the average response time R predicted using a Poisson arrival process and an exponentially distributed service time. Figure 2 depicts the job service times.

Then we measured the average energy consumed by a single job from observations obtained from serving a large number of jobs (1000), the average power consumption and the total running time of the experiment. The value of B was measured to be 13.32 Watts per job on average. The measured value of J_{job} and the calculated results from (1) using the experimentally estimated values of A and B are shown in Figure 3.

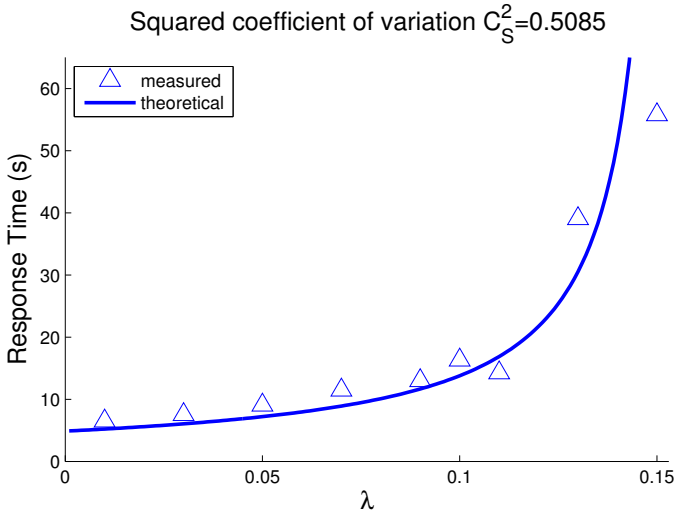


Fig. 1. Comparison of measured response time as a function of load $\rho = \lambda E[S]$, against theoretical predictions that assume Poisson arrivals.

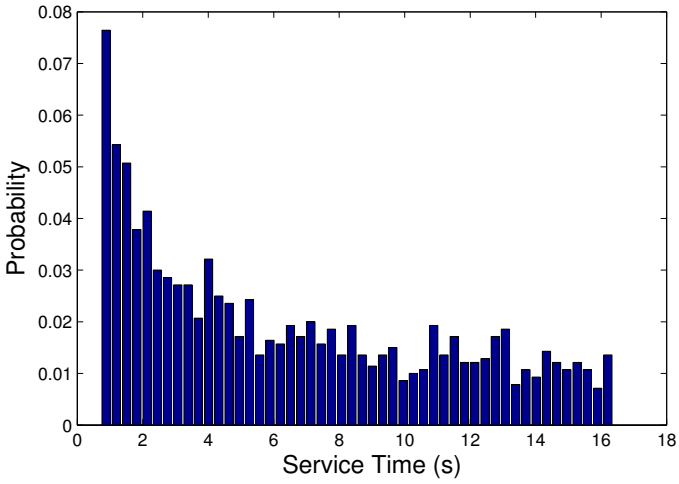


Fig. 2. Normalized histogram of service times.

B. Optimising Energy and QoS

A simple analysis allows us to compute the value of the arrival rate λ that minimises the composite cost function C_{job}. As a consequence, the optimum setting of the load $\rho^* = \lambda^* E[S]$ will depend on A (the idle power consumption) and on the ratio b/a which is the relative importance of energy consumption with respect to the average response time of jobs:

$$\rho^* = \sqrt{\frac{2bA}{a(1+C_S^2)}}\left(1+\sqrt{\frac{2bA}{a(1+C_S^2)}}\right)^{-1} \quad (3)$$

The expression (3) gives us a simple rule of thumb for selecting system load for optimum operation, depending on how we weigh the relative importance of energy consumption with respect to average response time or how fast we are getting the jobs done.

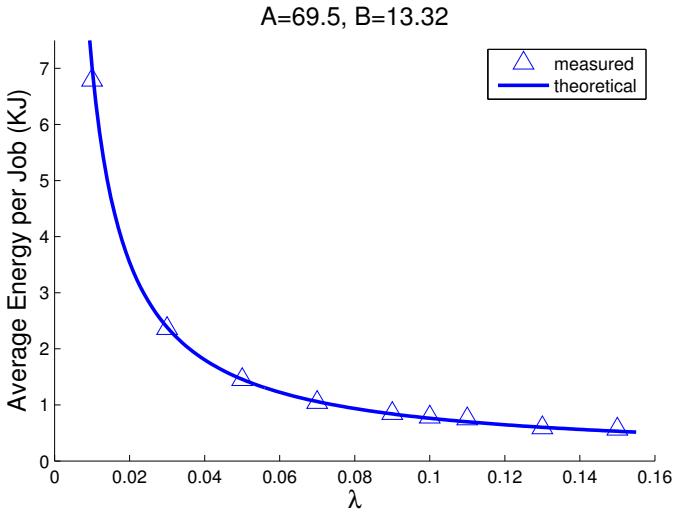


Fig. 3. Measured energy consumption per job (in KJoules) as a function of load ρ compared to the value predicted by the expression (1) using the experimentally measured values $B = 13.32$ and $A = 69.5$ Watts. The parameter b in (1) has been set to $b = 1.4725 \times 10^{-}04$ which is the inverse of the maximum energy consumption per job (in Joules) that was measured during the experiments.

We also see that ρ^* is an increasing function of the ratio $bA/a(1 + C_S^2)$. This tells us how the optimum load should increase as a function of the system's idle power consumption, and/or the relative importance that we place on energy.

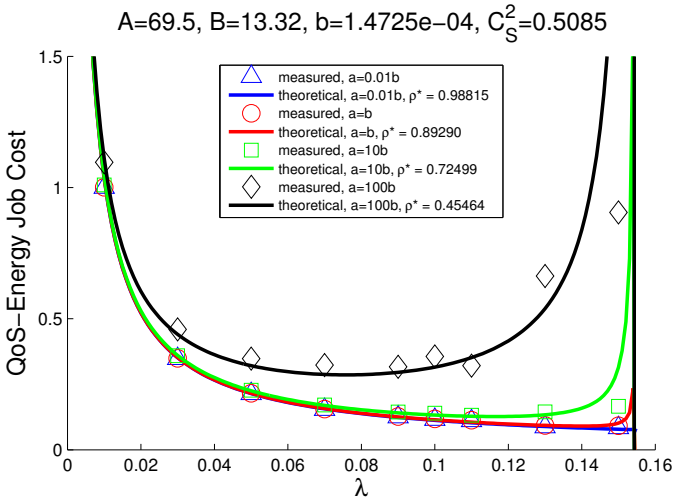


Fig. 4. The overall cost function as a function of load, when a varies between $0.01 \times b$ to $100 \times b$, for $b = 1.4725e - 04$ which is the inverse of the maximum energy measured during the experiments.

III. Load Sharing in N Sub-Systems

In many cases, a data centre is composed of N heterogeneous sub-systems, and our analysis can provide guidelines about how to share load among them, where each one has an energy profile described by the parameters A_i and B_i and with different processing capacity. The base system will be system 1 in the sense that $B_1 \leq B_i$ for $i \neq 1$. The execution of a job on system i will take on average time $E[S_i] = E[S_1]/\sigma_i$ where σ_i is the speed-up factor for system $i \neq 1$ with respect to the base system, and we *do not* imply that the base system is the slowest one. When λ jobs arrive per unit time and we assign a fraction p_i to the $i - th$ sub-system, denote $\lambda_i = p_i\lambda$ with $\sum_{i=1}^{N} p_i = 1$. The cost function:

$$\begin{aligned} C_{job} &= \sum_{i=1}^{N} p_i \{ \frac{aE[S_i]}{1 - \lambda_i E[S_i]} + bJ^i_{job} \} \\ &= \sum_{i=1}^{N} p_i \{ \frac{aE[S_i]}{1 - \lambda_i E[S_i]} + \frac{bA_i}{\lambda_i} + bB_i E[S_i] \} \end{aligned} \quad (4)$$

is then minimised with regard to the flows assigned to each sub-system, by computing:

$$\begin{aligned} \frac{\partial C_{job}}{\partial p_i} &= E[S_i](bB_i + \frac{a}{(1-\rho_i)^2}) \\ &\quad - E[S_1](bB_1 + \frac{a}{(1-\rho_1)^2}), 2 \leq i \leq N \end{aligned} \quad (5)$$

where $\rho_i = p_i \lambda E[S_i]$, so that to minimise the cost function we need to set:

$$\rho_i = 1 - \sqrt{\frac{a}{\frac{a\sigma_i}{(1-\rho_1)^2} + b[B_1\sigma_i - B_i]}} \quad (6)$$

where $\sigma_i = E[S_1]/E[S_i]$ is the speed-up factor of running a job on system i with respect to system 1. As a numerical example, consider three systems with speed-up factors $\sigma = 1,\ 1.5,\ 2.0$ (i.e, the second system is 50% faster than the first and the third system twice as fast). Assume that these systems have idle power consumption $A = 70,\ 85,\ 100$ watts respectively and let the B values be $B = 10,\ 12.5,\ 15$ watts; in this case higher computing power also implies higher power consumption. From (6) we obtain the optimum routing probabilities for the three systems as a function of the job arrival rate λ, as shown in Figure 5. Despite its higher power consumption, the faster system is preferred at lower loads because it can produce the lowest energy consumption per job. By assigning a greater weight to the response time via the parameter a in (5), the routing probabilities to the other two systems will increase as would be expected.

IV. Systems that are Turned On and Off

In all the previous analyses we assume that the systems are kept on all the time, and that power consumption varies with load. Based on these assumptions, and using a composite cost model that combines energy consumed per job, and average response time to a job we have tried to compute the load value (or operating point) that minimises this cost function.

In this section we assume that the system is turned on and off independently of its workload; when it is off we assume that it does not consume energy, but the on-off switching itself consumes an amount of energy γ in Joules, and it will also degrade the QoS. Furthermore, when the

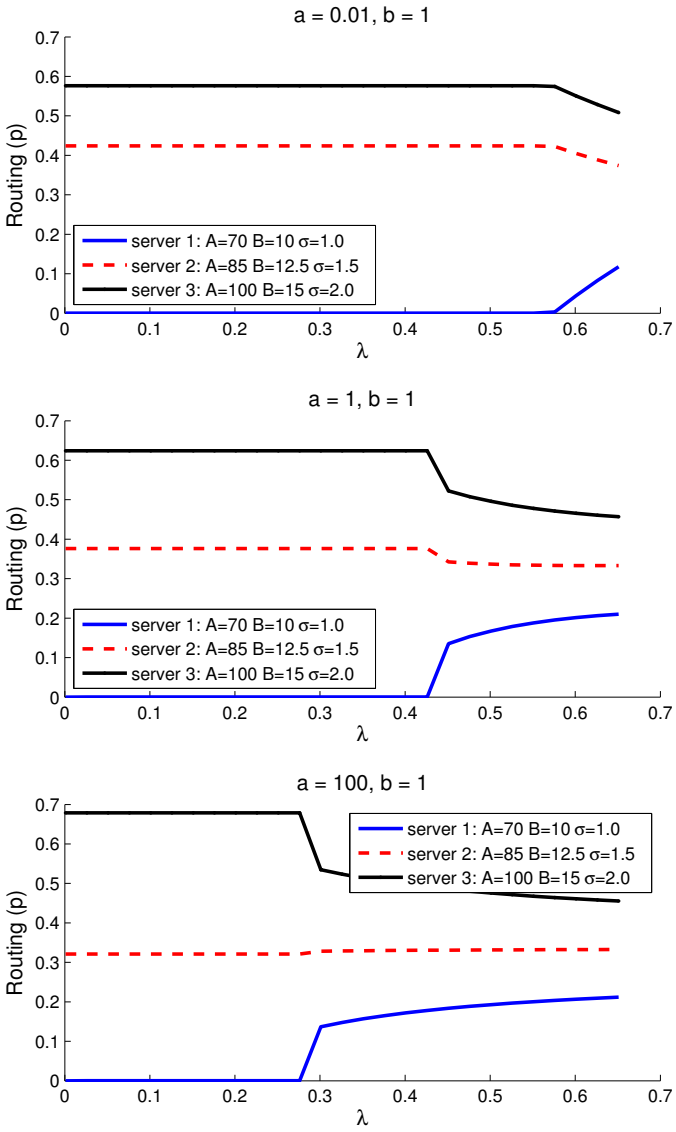


Fig. 5. Optimum load-sharing probabilities that minimize the Energy-QoS cost function for different values of a and b.

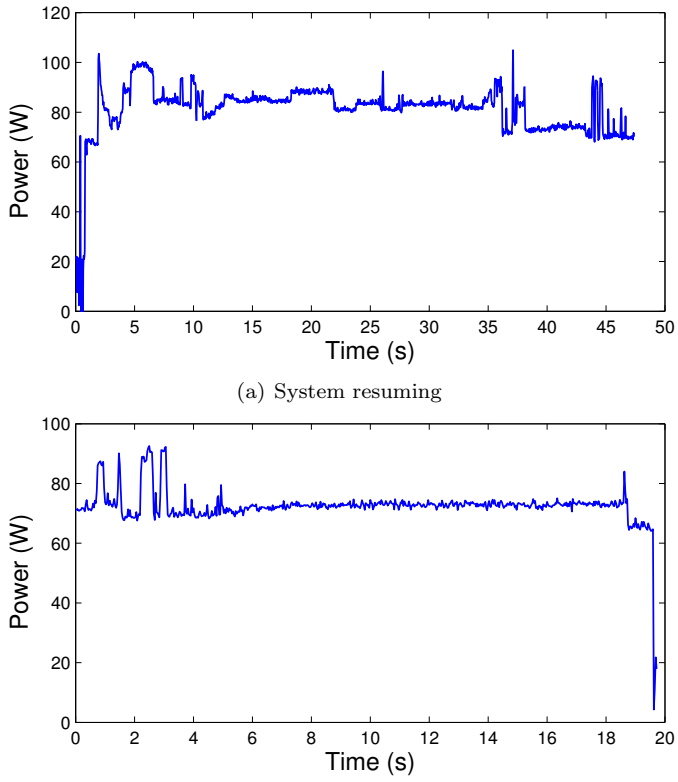
system is "asleep", it still consumes some power at D watts. Let F be the probability that this system is turned ON and let f be the rate at which it is being turned off and then on again. Using results such as [1] we obtain new cost function per job:

$$C^I_{job} = \frac{aE[S]}{F - \lambda E[S]} + b\frac{F(A-D) + \gamma f + D}{\lambda} + bB\frac{E[S]}{F}. \quad (7)$$

As a consequence, the optimum system load factor ρ^*_I becomes:

$$\rho^*_I = \frac{\sqrt{\frac{b[F(A-D)+\gamma f+D)}{a}}}{1+\sqrt{\frac{b[F(A-D)+\gamma f+D)}{a}}} \quad (8)$$

To experimentally evaluate the performance of this scheme, we use system hibernation, rather than hard ON-OFF system switching, because the system state which is in RAM during normal operation must be stored to hard disk during hibernation so that interrupted jobs may resume execution after a system resumption. Although the sleep mode is preferable to hibernation because putting the system to sleep would be faster, the sleep mode was not available in the system that we use for experiments, as is the case with many server-class systems. The measured power consumption of the server over time, while being put into hibernation and during system resumption, is shown in Figure 6. The measured time needed to hibernate the system was $19.719sec$, while system resumption took $47.375sec$. These measurements allow us to estimate the value of γ as being $5.268\ KJoules$.



(a) System resuming

(b) System hibernating

Fig. 6. System power consumption during (a) system resumption and (b) hibernation.

To control the system state, we use a separate system as a controller that issues *ssh* commands (`pm-hibernate`) and wake-on-lan packets to the server under study. In the experiments, the complete cycle of hibernation and normal operation is set to 200 *sec* so that $f = 0.005$. The probability that the system is in operation for job execution (i.e., excluding the hibernation-resumption times) was measured to be $F = 0.38631$, and the power consumption during hibernation was measured as being $D = 15.17\ W$. Figure 7 shows the measured (average) response time of jobs compared to the model predictions with the same set of jobs as in the previous experiment, except that the result of each job is written to disk instead of being

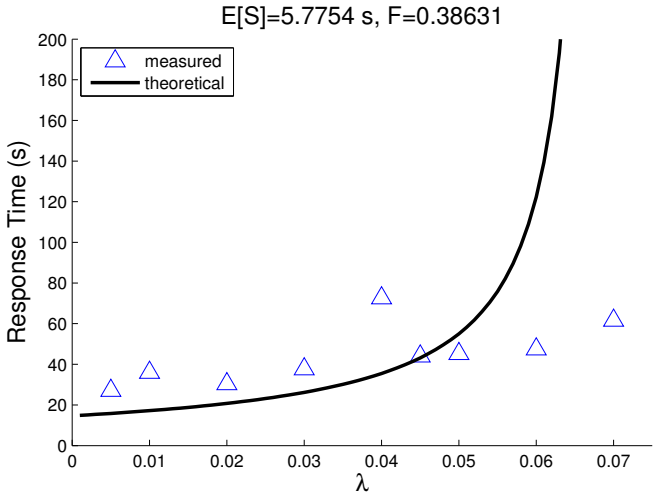


Fig. 7. Measured and theoretical average response time versus load, for the system with ON-OFFs with $f = 0.005$.

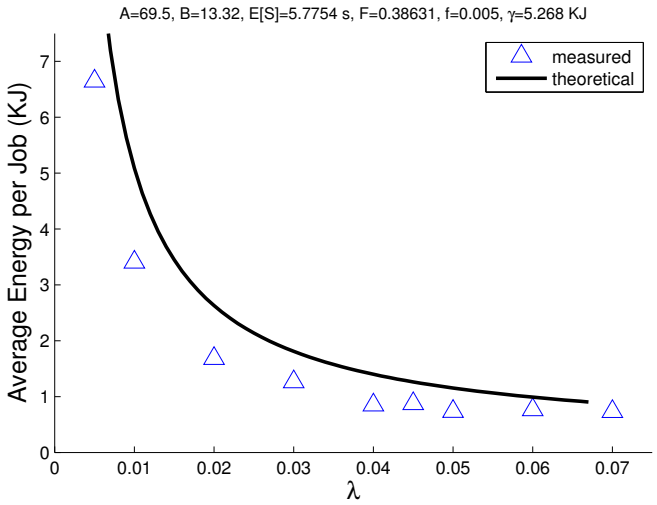


Fig. 8. Theoretical and measured energy consumption per job versus load, in the system with ON-OFFs for $f = 0.005$.

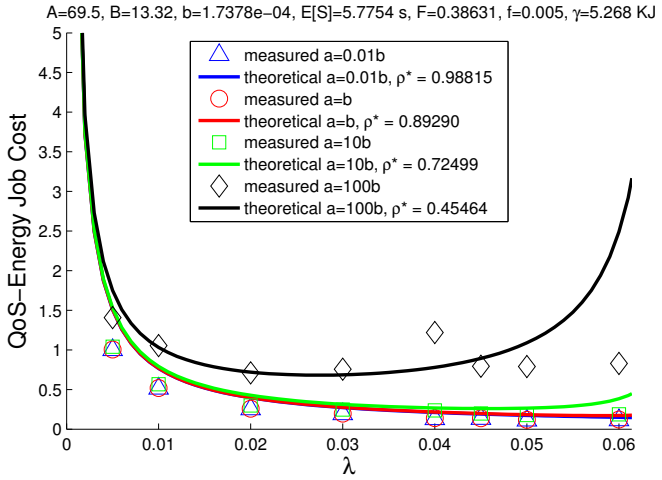


Fig. 9. Composite Energy-QoS cost metric versus load in the system with ON-OFFs for $f = 0.005$.

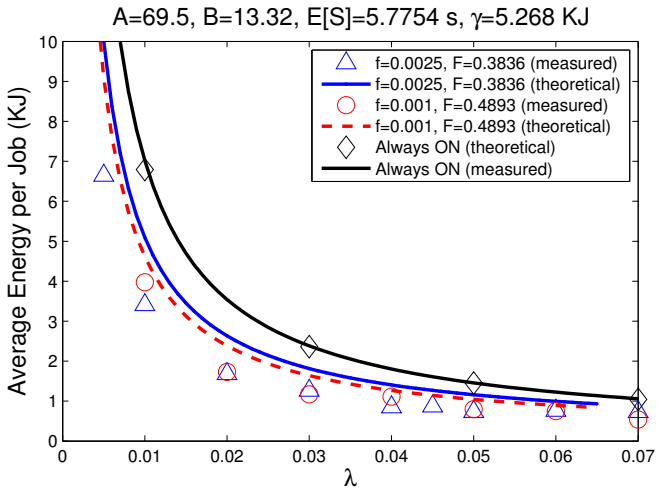


Fig. 10. Theoretical and measured energy consumption per job versus load, in the system with ON-OFFs for different values of f. We see that some energy can be saved if f is small and the "off" cycle is long.

sent over the network to a client. The average service time was measured to be 5.7754 *secs*. Figure 8 shows the average energy consumption per job and Figure 9 shows the measurements and theoretical predictions, for the overall cost function including average response time and energy per job.

V. Conclusions

We have shown that an optimum system load can be selected to minimise a cost function that includes both energy and QoS. Furthermore, putting a system intermittently to sleep can reduce overall energy consumption as shown in Figure 10, despite the energy wasted in switching the system on and off, and the reduction in QoS.

References

[1] E. Gelenbe, S. Tripathi, D. Finkel "Performance and reliability of a very large distributed system", *Acta Informatica*, Vol. 23, 643-655, 1986.

[2] A. Berl, E. Gelenbe, M. di Girolamo, G. Giuliani, H. de Meer, M.-Q. Dang, and K. Pentikousis "Energy-Efficient Cloud Computing", The Computer Journal, 53(7), September 2010, doi:10.1093/comjnl/bxp080.

[3] E. Gelenbe, D. Derochette "Probabilistic behaviour of a computer system under intermittent failures", *Comm. ACM*, Vol. 21, No. 6, pp. 493-499, June 1978.

[4] E. Gelenbe, I. Mitrani (2010), Analysis and Synthesis of Computer Systems. Imperial College Press, London.

[5] R. Lent, A sensor network to profile the electrical power consumption of computer networks, in: GLOBECOM Workshops (GC Wkshps), 2010 IEEE, 2010, pp. 1433–1437.

[6] Bolla, R.; Bruschi, R.; Carrega, A.; Davoli, F.; , "An analytical model for designing and controlling new-generation green devices," GLOBECOM Workshops (GC Wkshps), 2010 IEEE , vol., no., pp.1388-1393, 6-10 Dec. 2010