Power Proportional Characteristics of an Energy Manager for Web Clusters

Simon Holmbacka, Sébastien Lafond, Johan Lilius
Department of Information Technologies, Åbo Akademi University
Joukahaisenkatu 3-5 FIN-20520 Turku
firstname.lastname@abo.fi

***Abstract*—Energy consumption is a major issue in data centers operating 24 hours a day, 7 days a week. The power dissipated by a web cluster is not proportional to the numbers of incoming requests if only DVFS (Dynamic Voltage Frequency Scaling) is used. This is because of the nonlinear power efficiency of DVFS, the large load fluctuation in web services and the typical CPU utilization rates of a server.**

This paper presents a system level controller making a cluster of low-power servers power proportional by managing the resources on the platform. Our controller uses sleep states to switch on or off CPUs in order to continuously match the current workload with the system capacity. Methods from control theory are used to drive the CPUs from and into sleep states. The power proportional characteristics of the proposed energy manager are studied for different workload patterns. Results from system simulation show that power proportionality is obtainable but only with appropriate parameters set on the controller.

I. Introduction

Energy efficiency and power density are key issues for data centers. These factors do not only affect the operational costs and ecological footprint, but have also an important impact on the possibility to construct or expend data centers.

The *Efficient Servers* project [1] evaluated the increase of electric power consumption of servers in Western Europe at 37% between 2003 and 2006 [2]. In 2007 the energy consumed in data centers in Western Europe was 56 TWh and is projected to increase to over 100 TWh per year by 2020 [3].

In current servers, there is a mismatch between the energy-efficiency characteristics and the behavior of server class workloads as their most common operating mode corresponds to the lowest energy-efficiency region [4]. When using DVFS as power management technique, an energy efficient server still consumes about half of its energy while idling.

With an average of 10 to 50 percent CPU utilization for servers [5] and the large load fluctuation found in typical web services [6], the use of slower but more energy-efficient cores could match the workload more efficiently with a much finer granularity than server-grade cores. A cluster of mobile processors can provide the same computational power as server-grade processors, but with a lower power density. The usage of mobile processors also aims at obtaining cheaper server facilities by minimizing the need of active cooling infrastructure.

Because switching on and off a CPU is orders of magnitude slower than changing its voltage and frequency, a cluster of such low-power CPUs needs an energy manager on system level i.e. a component controlling the whole cluster as one entity and continuously matching the current workload with the whole cluster capacity.

This paper analyses for different workload patterns the proportional characteristics of an energy manager that uses sleep states to dynamically adjust the system capacity according to the workload so that minimal performance penalty and maximum reduction in energy consumption is obtained.

II. Related work

Previous work has been done in the area of using sleep states to reduce the energy consumption of mobile processors. The authors in [7] are proposing a mixture of high-end Xeon servers combined with low-end mobile processors in order to achieve a fine granularity of system capacity in relation to the workload. All processing elements in the system uses sleep states to shut down the CPUs during low workload and thus reduce the energy consumption. Once the system recognizes an increase in workload, the system activates the processing elements in accordance with their different capacities and wake-up times. To determine the power proportionality, experiments were conducted on two different types of workload patterns, which results concluded in a power proportional system.

In our approach, the power management system uses control theory as basis for the capacity adaption. We argue that the use of the PID controller could, with correctly set parameters, create a near optimal adaption of system capacity to the workload. Moreover, we intend to use a cluster consisting only of low-end mobile processors to gain finer power granularity of the whole system.

The authors in [8] present a sleep state based power manager for server grade CPUs together with PSUs (Power Supply Units) in a so called *RAILS*-configuration (Redundant Array for Inexpensive Load Sharing). Smaller low-power PSUs are used instead of one powerful, since a RAILS-configuration will make the PSUs operate in their most energy efficient sweet spot. As the power need increases more PSUs are enabled to provide the sufficient power needed. Similarly to [7] the CPU cores are switched on and off according to the workload to give a better power proportionality of the system. The method of anticipating the workload curve was not mentioned, but an average of 74 % energy reduction was achievable according

to the authors [8]. The power proportionality was determined based on the wake-up time for the core, and would in best case result in a linear function.

We have used the idea from both of the previous works together with the implementation of a PID controller [9] to adjust the capacity of the system. By using a larger number of low power CPUs we argue that – by having a finer granularity – we could achieve a higher energy reduction while keeping the power proportionality constant and obtaining a sufficient performance.

III. Power Proportional Web Cluster

A. System Level Power Management

We created a power manager which adjusts the system capacity dynamically in order to save energy. The manager uses sleep states to switch on and off cores according to the current need and according to the anticipated future workload. The simulated cluster uses ARM Cortex-A8 processors used in the BeagleBoard and its wake-up time was measured by experiments to roughly 800 ms. By using the measured capacity of a Cortex-A8 the energy consumption for a many-core cluster was simulated. The basic processing element in this paper is referred to as a *core*, since embedded systems with multi-core configurations have recently been available.

1) **Overview:** The outline of the framework is shown in Figure 1. The framework is based on input in form of requests made to the service. The framework shows the output based on data from a PID-controller and a feedback loop, which sends information to the compare-block regarding the needed capacity of the system.

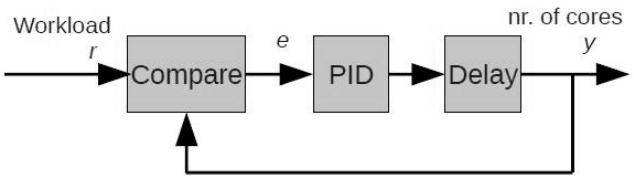


Fig. 1: Structure of the simulation framework

2) **System capacity:** System capacity is measured as the number of requests per second the system as a whole is able to handle. The compare block reads the workload with a certain sample rate and divides this number with the current capacity of the system. The ratio of this division determines the QoS (Quality of Service) output. When the capacity monitor in the compare-block notices a higher workload than the system is able to handle, it sends the ratio between the workload and the capacity to the PID controller in form of an error value. Similarly when the capacity of the system exceeds the workload, the monitor sends a negative error value to the controller which in turn switches off CPU cores.

Since switching on and off CPU cores is not instantaneous, the framework uses a delay to postpone the control signal to the workload comparator. For this function, the simulation framework implements a unit-delay block with a configurable delay length. This simulates the actual delay introduced in the change of CPU state.

The framework uses one *static* core. This core will constantly be active and is used to instantaneously handle the small amount of requests that are made between request peaks. The number of static cores is also configurable.

3) **QoS value:** A trade-off to energy consumption is the performance and response time of the system. By lowering the capacity of the system the performance will drop – this leads occasionally to an increased response time for certain requests. *Quality of Service* is the measurement on how well the system performs compared to a pre-defined value. Our simulations show a drop in QoS as soon as a request is delayed more than the selected deadline. The amount of delayed requests compared to non-delayed requests results in the QoS value. If every request is handled before their deadlines the QoS will be 100 %, if half of the requests are handled before the deadline the QoS will be 50 % etc.

4) **PID controller:** The controller block in Figure 1 includes a PID controller which, based on methods from control theory, adjusts the capacity of the system. The obtained difference between y and r is called the control error e, which is the *a priori* result from the capacity comparison in the previous block. The output y of the PID controller partly shows the amount of cores needed to achieve a sufficient performance and partly generates feedback data to the comparison block in the next time frame. The aim of the feedback loop is to minimize the control error and to thereby achieve equilibrium in the system.

The behavior of the PID controller is determined by setting P, I and D values in the controller. The value of P determines how fast the controller reacts on a change in the reference value r. The value I, which is the inverse time constant of the controller, determines the integral effect of the control error. The derivative part, which is adjusted by the parameter D, predicts the future input based on the previous input.

5) **Final energy consumption:** The system shows the power output as a multiple of the amount of active cores and their power dissipation. The cores are assumed to run on the highest possible clock frequency once activated, and retain this clock frequency until they are shut down. The final energy consumption is the sum over the power dissipation for all time frames in the simulation.

B. Power Proportionality

While the power manager shows a promising result in energy reduction, we need to investigate how well it scales in a growing web cluster. To be able to apply the power manager into a large cluster the proportionality of the workload compared to the power dissipated must be constant. This means that if the workload increases by a certain factor, the power dissipation should also increase with the same factor.

To measure the proportionality we created different workload patterns against which the power dissipation was compared. The behavior of the system was simulated by inserting the workload patterns into the simulation framework.

IV. Simulation data

Our simulations will be based partly on specially generated request patterns and partly on real web server data, which allows for a comparison of power proportionality in different situations. The first simulations are executed against trivial cases to evaluate and clearly illustrate the theory. Later the real web server data will show the obtained proportionality in a real-world scenario.

A. Request patterns

1) **Linear cone:** The first pattern to investigate energy proportionality is generated by requests made according to a linear cone as seen in Figure 2. Requests are made with certain increments and a selected step size. The increment determines how much the requests increase for each step, the length of which is selected by a step size.

For the energy to be proportional to the requests, the power dissipation for all time frames should increase linearly according to the workload curve. A linear increase in the power dissipation would scale the energy consumption well in a large web cluster.

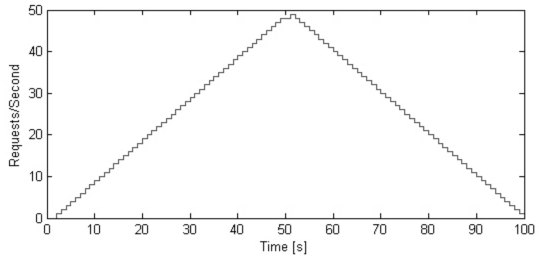


Fig. 2: Linear workload pattern

2) **Exponentially increasing cone:** The exponentially increasing cone in Figure 3 is created by incrementing the steps multiplied by a certain constant. The energy proportionality of an exponentially increasing pattern should be followed by a similar pattern in the power dissipation. By investigating different patterns, we will be able to determine the proportionality characteristics of the power management system.

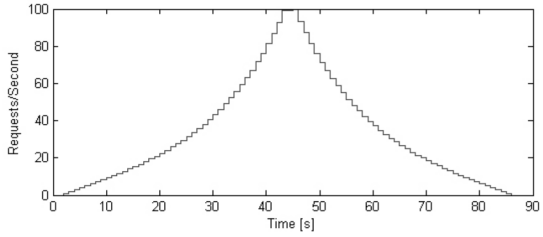


Fig. 3: Exponential workload pattern

3) **Web server requests:** We also used requests made to a Finnish web space provider [10] to compare the results of power proportionality in a real-world scenario. The request samples were obtained from 1 Nov. 2010 and simulated for 30 minutes and shown in Figure 4. The simulation data is freely available from [11].

B. PID-parameters

The values of the PID-parameters P, I and D determines how the controller should react to its input signal. The parameters were chosen based on a heuristic method, and tuned until desired result was achieved. The simulation framework supports currently only static PID-parameters, but could eventually be further developed to handle dynamic values. The value of the delay after the PID-block was chosen, based on conducted experiments, to 1000 ms in order to ensure the necessary delay of the wake-up time, which was measured to 800 ms. A sample time of 250 ms was selected as time frame for updating the output from the controller.

C. BeagleBoard power dissipation

To obtain values for the simulation framework and be able to run a proof-of-concept simulation, the power dissipation of one BeagleBoard revision C3 low-power platform was measured. The BeagleBoard is equipped with one ARM Cortex-A8 processor-based TI-OMAP3530 chip that does not require any forced cooling system or heat sinks. The system ran Ångström Linux kernel version 2.6.32 and was controlled through a remote serial console. The operating performance points (OPPs) of the TI-OMAP3530 chip were used to dynamically scale the clock frequency and voltage of the ARM subsystem. The OPPs were accessed through the Linux ACPI. To avoid unwanted energy consumption, the display subsystem of the TI-OMAP3530 was disabled. The BeagleBoard includes a resistor, which provides a way to measure the current consumption used by the board. The voltage drop across the resistor was measured for each OPP and the corresponding power was calculated. The obtained power values of the system running at respective voltage and clock frequency are displayed in Table I. To ensure that the load would remain constant during the measurements, the processor was stressed to 100 % utilization using a simple program that recursively counts Fibonacci numbers. The highest OPP (720 MHz) was used in the simulation framework to represent the power dissipation for the active core.

TABLE I: Measured power dissipation of the BeagleBoard

Frequency [MHz]	720	600	550	500	250	125
Voltage [V]	1.35	1.35	1.27	1.20	1.06	0.985
Power [W]	1.40	1.15	1.05	1.00	0.65	0.55

D. BeagleBoard load capacity

The system capacity is dependent on both the number of CPU cores in use and their capacity. We needed to determine the capacity of a BeagleBoard in order to run a realistic simulation.

Experiments were therefore conducted, which results defined the system capacity of one BeagleBoard. The test tool in use was Autobench [12] which generates requests to an Apache [13] server running on the BeagleBoard. The number of requests per second generated by Autobench started from a

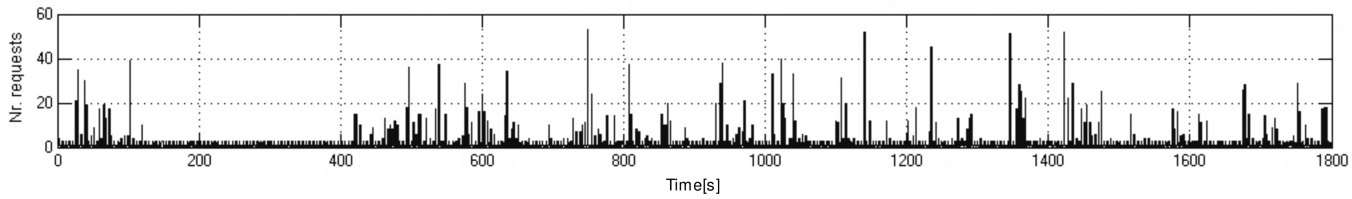


Fig. 4: Workload sample from [10] 1. November 2010

selected number and increased by specified increments. When the number of requests per second start to exceed the capacity of the host, delay-errors will start to occur as the selected deadlines for the requests are not met. The requests from Autobench are made to a selected file on the Apache server running on the BeagleBoard. Since a larger file will require more time to process, the capacity of the server is dependent on the size of the requested file. Our experiments show that a file size of 248 KB with a deadline of 1000 ms will result in the capacity of 5 requests per second – this number was used in the simulations. Related experiments in [7] result also in the capacity of 5 requests per second for the BeagleBoard which, as stated, would compare to a file size of 248 KB.

The file size was constant for each simulation with a maximum amount of 10 or 20 cores available. Using these numbers, the cluster would have a theoretical maximum capacity of 50 or 100 requests per second.

V. Simulation results

The framework for simulation was set-up according to the results from the experiments presented in the previous section. The simulations were run in three phases: using a linear cone, an exponential cone and real web server data.

A. Linear cone

The first simulation used the workload pattern of a linear cone. The step size of the cone determines how fast the request rate is climbing. A longer step size means that request rate will stay constant for a longer time. The step size for the first simulation was set to 5 seconds per step. Results from the first simulation is shown in Figure 5, which displays three graphs. The first graph (1) shows the power dissipation compared to the request rate. The second graph (2) shows the amount of cores switched on in the current time frame, and the last graph (3) shows the quality of service as a function of time.

The amount of cores in Figure 5 (2) shows to be spiking each time the step increases. This peak is a result of the delay (wake-up time) the cores introduce when switching from sleep state to active state. Because of the delay, the system will not react instantaneously to the increase in request rate (workload). As soon as the system notices the drop in QoS it needs to compensate for the delayed requests by switching on additional cores. After the system has processed the delayed requests, the need for the additional cores does not exist any more and they are shut off. This phenomenon occurs every time the step increases until a stable request rate is established.

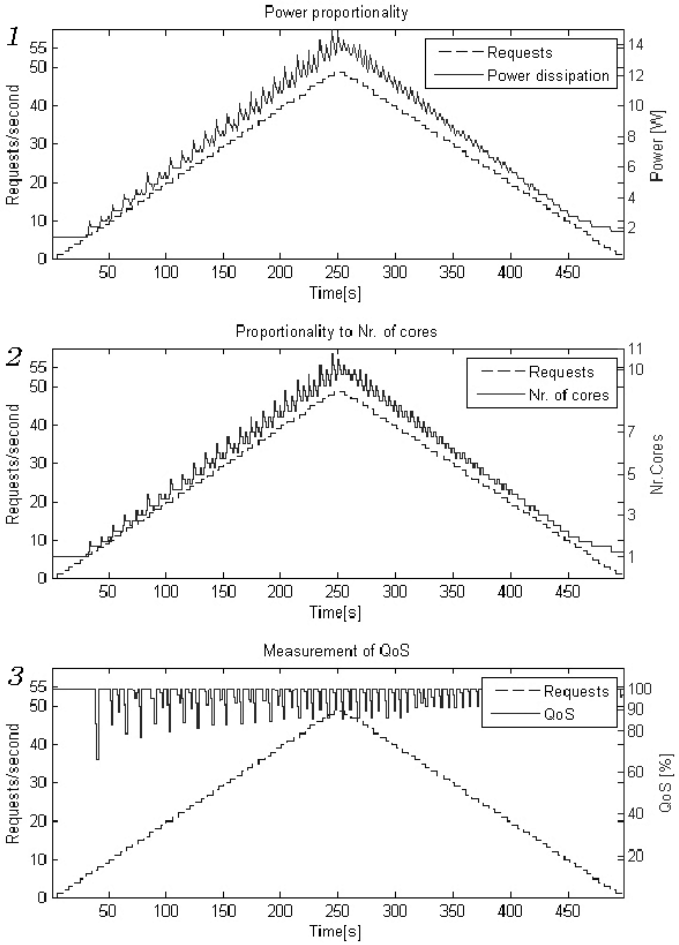
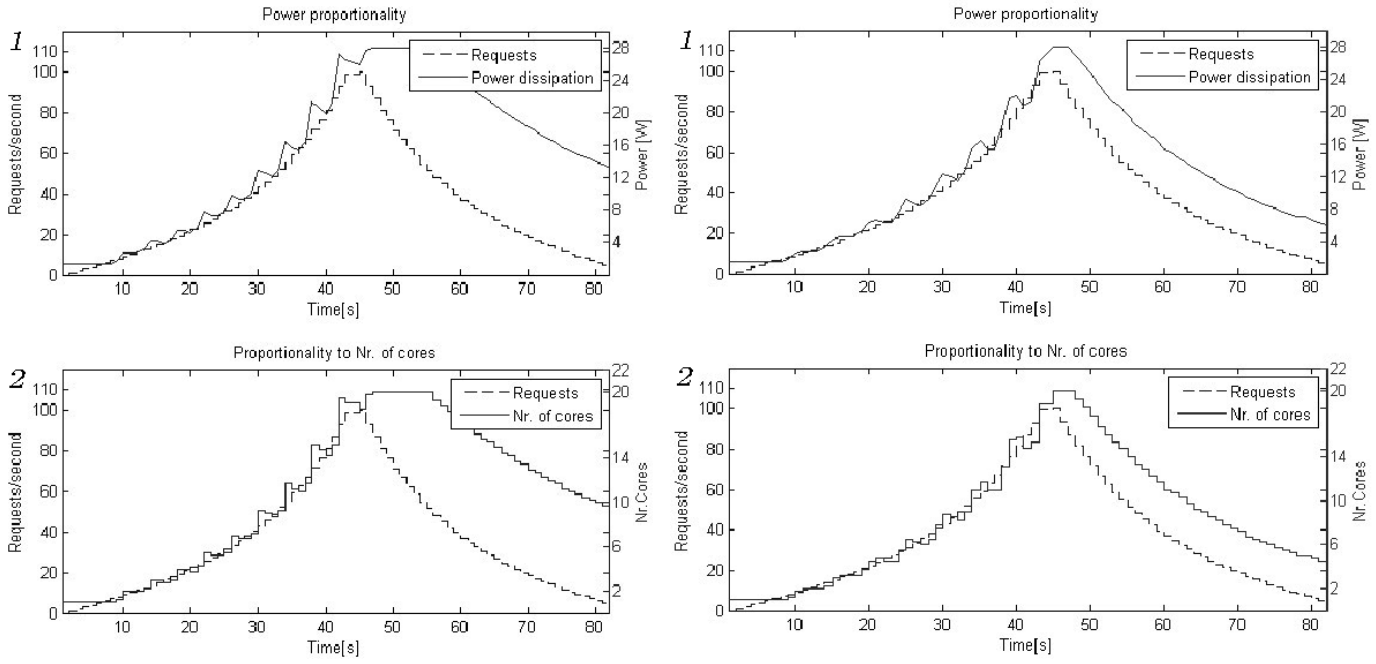


Fig. 5: Simulation of linear cone pattern

The power dissipation follows the amount of CPU cores since a CPU core is, in this model, either fully on or off. Similar peaks occurs therefore also in the power dissipation curve in Figure 5 (1) as the system compensates for the delayed requests.

Figure 5 (3) shows that the QoS value drops each time the step increases. This model, which strives to simulate a realistic case, cannot fully eliminate the QoS drop because of the delay of waking up cores. A strategy to reduce the QoS drop is to make the PID controller more QoS conservative by adjusting the parameters – this, however, also results in increased energy consumption.

Conclusions about the power proportionality can be drawn



(a) Simulation with initial parameters

(b) Simulation with improved parameters

Fig. 6: Simulation of exponential pattern with two different sets of parameters

based on the curve from Figure 5 (1). Aside from the fluctuation peaks when the steps increase, the power dissipation curve follows the workload in a linear fashion. By applying a low-pass filter on the power curve, we obtained the mean values for the fluctuating graph. The proportionality factor between power and workload is shown in Table II.

TABLE II: Measurement of power proportionality obtained from Figure 5 (1)

Time [s]	100	150	200	250	300	350	400
Req/sec	20	30	40	50	40	30	20
Power [W]	5.51	8.32	11.12	13.76	11.48	8.63	5.77
Prop. [Req/J]	3.63	3.60	3.60	3.63	3.48	3.47	3.47

Table II shows seven values derived from Figure 5 (1). The last row shows the final proportionality factor which is the ratio between the Req/sec and power, and thus uses the unit Requests/J. From the last row in the table we can see that the values of the proportionality does not fluctuate much – in fact the largest fluctuation, shown in Table II, results in a difference of 5 %.

B. Exponential cone

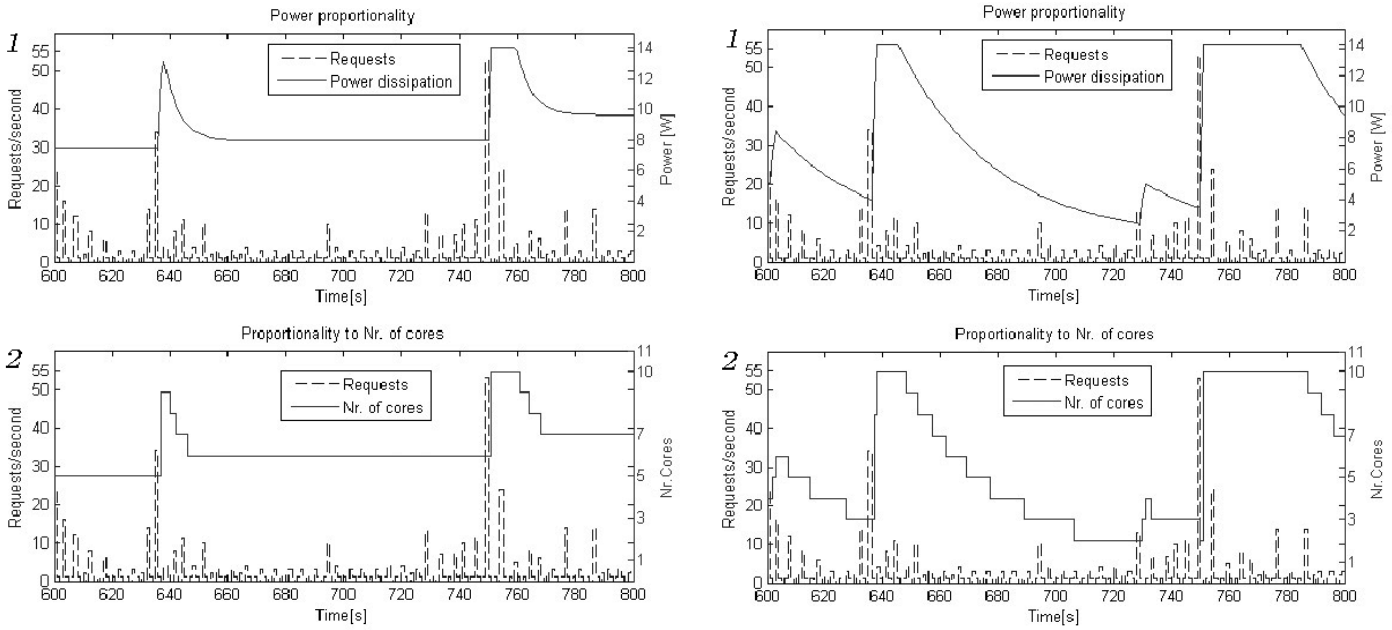
Secondly the framework was set-up with the same parameters as in the previous case, but with a different workload pattern. The second pattern was an exponentially growing request curve as shown in Figure 3. The workload used in this simulation has, in contrast to the previous simulation, a maximum value of 100 req/s to more clearly illustrate the behavior of the exponentially increasing pattern. To cope with 100 req/s we allow the system to use 20 cores instead of 10.

Figure 6a shows the result from the simulation. By using the same PID-parameters the controller fails to establish an effective output signal used for switching on and off the cores. The amount of cores in Figure 6a (2) shows to be insufficient as the curve increases. As the curve exponentially decreases, the control error remains high because of integrating property of the controller. The result is a slowly diminishing output which leads to wasted energy.

Because this simulation did not show a power proportional behavior we needed to alter the PID-parameters on the controller. After establishing a new set of parameters by experiments we run the simulation again. The new set of parameters achieved, with the same workload, better power proportionality as shown in Figure 6b. Table III shows both power dissipation and the power proportionality for both graphs in Figure 6. As seen in the table, case *b* (row 6) will show better proportionality than case *a* (row 4) because the power curve follows the workload more precisely.

TABLE III: Measurement of power proportionality obtained from Figure 6 (1)

Time [s]	15	25	35	45	55	65	75
Req/sec	40	60	80	100	80	60	40
Power 6a	4.20	7.70	15.75	27.65	26.95	20.30	15.4
Prop. 6a	3.65	4.08	3.76	3.38	1.79	1.22	0.69
Power 6b	4.20	7.70	15.40	28.00	18.55	11.90	8.40
Prop. 6b	3.65	3.82	3.61	3.34	2.60	2.08	1.55



(a) Simulation with exponential parameters

(b) Simulation with linear parameters

Fig. 7: Simulation of exponential pattern with two different sets of parameters

C. Web server data

The final simulation used the request log from a web server as workload. The PID-parameters for this simulation was chosen according to the previous simulation (exponential cone), but since the workload shows a maximum value of 50 req/s we allowed only 10 cores to be active simultaneously. The results from the simulation is shown in Figure 7a. The PID-parameters used for the exponential cone turned out to be unsuitable for controlling the workload from the web server, since the power dissipation will remain relatively constant and thus result in a poor energy management.

To achieve a better power proportionality we adjusted the PID-parameters according to the simulation with the linear cone. The results from this simulation is pictured in Figure 7b. By interpreting the curves in Figure 7 (1) we can see that the power proportionality of the system is highly dependent on the controller settings. The use of CPU cores pictured in Figure 7b (2) matches the workload better than the previous simulation. By using the new PID-parameters we achieved a better power proportionality as seen in Figure 7b (1) compared to the previous simulation showed in Figure 7a (1). The QoS was not included in Figure 7b and 7a since the numbers of cores remained high during the whole simulation and thus not resulted in any substantial QoS fluctuations.

To further improve the power proportionality factor we tuned the PID-parameters for this third case. The mentioned PID-parameters were selected to match the *spiky* workload obtained in a real-world scenario seen in Figure 4. A simulation using these parameters results in a greater energy reduction than the two previous simulations – this while keeping an acceptable QoS. The result from the last simulation is shown in Figure 9.

To calculate the power proportionality from Figures 7 and 9 we needed to time shift the power curve to accommodate for the delay introduced by the wake-up mechanism. Furthermore, we chose certain points in time where interesting measurement would take place. Table IV views the power proportionality for all three cases, with the power curve shifted one second to the left. This number displays the proportionality factor, meaning that a lower value is obtained when the system is using much energy [J] to serve few requests.

TABLE IV: Measurement of power proportionality obtained from Figure 7 (1) and 9 (1)

Time [s]	617	635	654	680	697	752	765
Req/sec	6	14	10	1	10	50	1
Prop(lin) 7b	1.07	3.43	0.89	0.17	2.45	3.57	0.07
Prop(exp) 7a	0.80	1.87	1.23	0.13	1.26	3.57	0.09
Prop(real) 9	2.54	1.31	1.18	0.66	1.30	3.57	0.61

Prop(lin) and *Prop(exp)* in Table IV represents the power proportionality of the first two simulations on real web server data. As seen in the table the fluctuations are large and close to zero in the last column. A value close to zero means that the power output of the system is much larger than the amount requests made to the system, i.e. the system is wasting much energy. The 7:th column (at time 752 s) shows equal values for all three cases. This happens due to the fact that the system is slightly overloaded, which happens if 50 or more requests are made during one second.

The results in *Prop(real)* show occasionally drops in proportionality such as at times 680 s and 765 s. This drop occurs

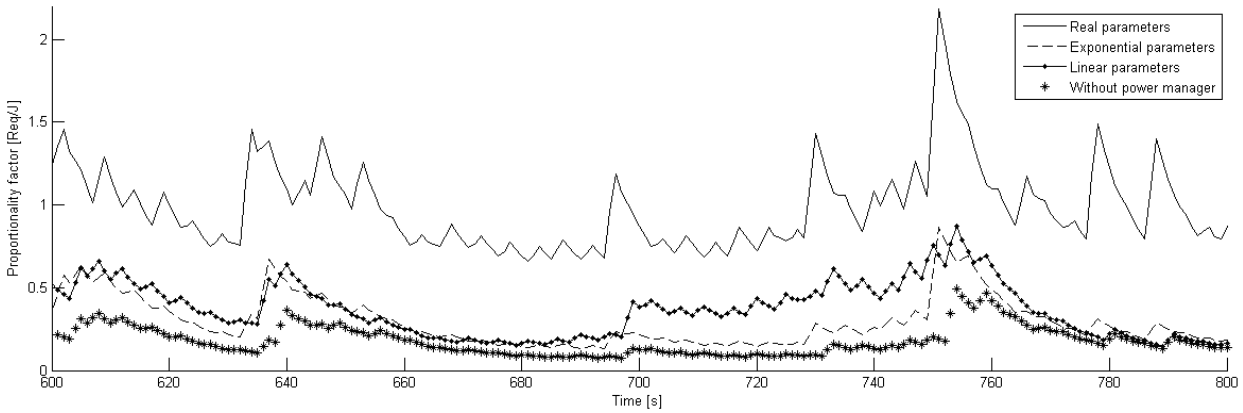


Fig. 8: Measurement of proportionality

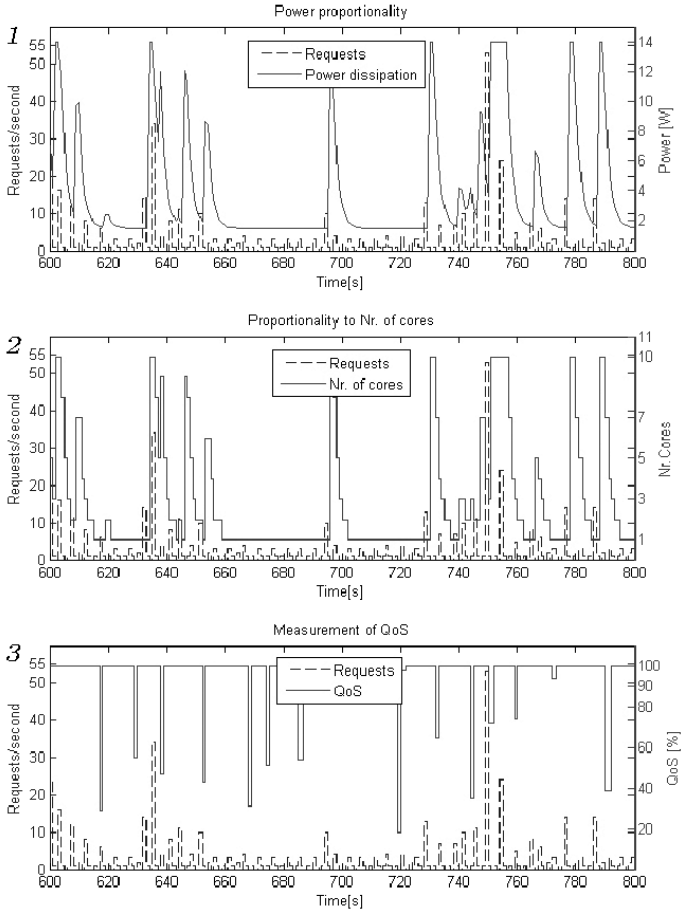


Fig. 9: Simulation of web server pattern

because of the static CPU core that will run even though the system only needs to process one request. Implementation of DVFS would in these cases be useful since the granularity of the power scaling would increase. Furthermore, in *Prop(real)* at times 635 s, 654 s and 697 s the power proportionality decreases even though the request rate is not minimal. This phenomenon is a temporal response from the system during a workload peak. The system compensates for delayed requests by temporary rising the capacity. Over time these workload peaks would not account substantially for the power proportionality of the system.

To illustrate the different proportionality factors in the three different cases, the drawn graph displays the whole time range from 600 s to 800 s (Figure 8). Furthermore, a low pass filter was used to filter the highly fluctuating output signal to better illustrate the average proportionality factor by using different PID parameters. The figure clearly shows that correct PID parameters will result in a higher proportionality factor, and thus less energy waste. The proportionality factor for a system without any power manager (all cores statically on) was also displayed in order to better compare the power proportionality of the manager.

D. Simulation summary

By observing the results from the three different simulations: *Linear cone*, *exponential cone* and *web server data*, we can state that power proportionality could be achieved by setting the appropriate PID control parameters for the workload in question. The outcome from using the power manager is a system with higher power proportionality, which means that most of the CPU power is actually used for real work rather than waiting for work to arrive.

The PID-controller reacts differently depending on the input of the controller, which means that the settings for one environment not necessarily support another environment. A run-time update of the PID variables would mean that the system should automatically accommodate the PID-parameters for not only the workload, but also the workload pattern. Another solution would be a model that reflects an *a priori* workload with sufficient precision. This model could use static PID parameters as long as the workload follows the model.

VI. Conclusions

A energy manager for a many-core web cluster was created in order for the system to show power proportional characteristics when serving alternating amount of work. The

power manager matches the system capacity, every time frame, according to the workload by using a PID-controller.

This paper has investigated the power proportionality characteristics of the power manager by simulations performed on determined workload patterns. The results from the simulations were used to determine the relationship between power dissipation and workload for selected sample points.

The simulations were divided up into three different workload patterns. These three patterns were individually simulated and their respective results were compared. The simulations show that power proportionality is achievable, but only with the correct parameters set on the controller. The controller parameters determine how the controller reacts on changes in the input signal.

In the most trivial case we used a linear cone as the workload pattern. The results from the controller showed a non fluctuating and constant relationship between the workload and the total power dissipation of the system. The second simulation used an exponentially increasing workload pattern which, with the same controller parameters, did not reach a sufficient proportionality. The parameters on the controller were adjusted, after which a better result was obtained. Lastly real data from a web server was used as workload pattern. The result showed that the controller parameters from neither of the two previous cases would give a power proportional system. The controller parameters were therefore adjusted to match the *spiky* nature of web server requests, which resulted in an increased power proportionality.

The parameters of the PID-controller need, as a conclusion, to match the workload pattern for the controller to be able to match the capacity of the system to the workload. Incorrect parameters will either result in poor QoS or unnecessary energy waste. The parameters need therefore a model from which the workload pattern is derived, or to dynamically change during run time. Based on these assumptions, our simulations show that the proportionality factor of a many-core system that uses a sleep state based power management is achievable.

VII. Future work

As concluded, future research is needed to determine if the system can reach power proportionality facing a general workload pattern. In order to adapt the system to such a pattern, the PID controller must adjust its control parameters during run-time. The run-time mechanism must therefore both analyze the previous workload pattern and anticipate the future workload pattern in order to make adjustment of the parameters. By recording the history of workload, CPU time, performance etc. the system could create certain models against which the parameter settings are set. After changes in the input variables occur, the models changes and thereby requires different controller settings.

Furthermore the power proportionality and energy reduction need to be compared to a system with both the current power manager and DVFS for each CPU core. The CPUs could thereby scale down their frequencies, and power dissipation (Table I) in accordance with the workload and thus increase the granularity of the system capacity further. Because scaling the frequencies is by orders of magnitude faster than switching on and off cores, the switching delay would not increase substantially.

References

[1] "Efficient servers, a project conducted within the eu-programme intelligent energy europe." [Online]. Available: http://www.efficient-server.eu

[2] B. Schäppi, F. Bellosa, B. Przywara, T. Bogner, S. Weeren, and A. Anglade, "Energy efficient servers in europe. energy consumption, saving potentials, market barriers and measures. part 1: Energy consumption and saving potentials," The Efficient Servers Consortium, Tech. Rep., November 2007.

[3] "Code of conduct on data centres energy efficiency, version 2.0," European Commission. Institute for Energy, Renewable Energies Unit, Tech. Rep., November 2009.

[4] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, pp. 33–37, December 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1339817.1339894

[5] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33 –37, 2007.

[6] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis," Vrije Universiteit, Amsterdam, The Netherlands, Tech. Rep. IR-CS-041, Sepember 2007 (revised: June 2008).

[7] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. H. Katz, "Napsac: design and implementation of a power-proportional web cluster," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, ser. Green Networking '10. New York, NY, USA: ACM, 2010, pp. 15–22. [Online]. Available: http://doi.acm.org/10.1145/1851290.1851294

[8] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," *SIGPLAN Not.*, vol. 44, pp. 205–216, March 2009. [Online]. Available: http://doi.acm.org/10.1145/1508284.1508269

[9] J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, *Feedback Control of Computing Systems*. Wiley and sons inc., 2004.

[10] K. Ab, "Kulturhuset," January 2011. [Online]. Available: http://kulturhuset.fi/start/

[11] Kulturhuset, "Request log november 2010 kulturhuset." [Online]. Available: https://research.it.abo.fi/projects/cloud/data/Request_log_kulturhuset_nov2010.zip

[12] J. T. J. Midgley, "Autobench," Xenoclast, May 2004. [Online]. Available: http://www.xenoclast.org/autobench/

[13] T. A. S. Foundation, "Apache," 2010. [Online]. Available: http://www.apache.org/