A PID-Controlled Power Manager for Energy Efficient Web Clusters

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

***Abstract*—Large data centers using high-end processors operating continuously around the clock are major energy consumers. Long periods of idling due to low workload will cause a waste in energy because the processors are active but not doing any useful work.**

A cluster of low-end embedded processors could continuously match its computational capacity with the workload at a much finer granularity than a server-grade processor by changing the power states of the CPUs. This paper introduces a framework simulating a system level power manager for many-core clusters targeting server cards used in warehouse-sized data centers. The power management system uses sleep states to switch on or off processing elements in a cluster of low power boards to match the capacity of the whole system with the workload, and thus save energy. A PID-controller is implemented in the system; a component already well known with established methods in the industrial control domain. We intend to use this component to effectively determine the number of active processing elements in the used many-core cluster.

The proposed power manager can save up to 62 percent in energy compared to a system which only uses dynamic voltage and frequency scaling as power management.

***Keywords*-Power Management; Web Clusters; PID-controller; Low Power Processors;**

I. Introduction

Energy efficiency and physical size have become key issues for server cards used in warehouse-sized data centers. These factors do not only affect the operational costs and ecological footprint, but also have an impact on the possibilities to construct or expand data centers. With an average of 10 to 50 percent CPU utilization for servers [1] and the large load fluctuation found in typical web services [2], the use of slower but more energy-efficient cores could match the workload more efficiently with a much finer granularity and higher power proportionality [3] than server-grade cores.

A cluster of mobile processors can provide the same computational power as server-grade processors, but with a lower total energy consumption. Such a cluster can reduce the energy consumption efficiently by switching off elements according to the current need of service. For this purpose the cluster needs a power management on system level i.e. a component controlling the whole cluster as one entity.

This paper proposes a system level power manager for a cluster consisting of low-power nodes. The power manager uses sleep states to dynamically adjust the system capacity according to the workload. The monitored workload is matched so that minimal performance penalty and maximum reduction in energy consumption is obtained.

The PID-controller used in the industrial domain contains well established methods for obtaining stability and equilibrium in a dynamic system. We intend to exploit the theory of the PID-controller, and implement it into our power manager for matching the capacity of the system to the incoming workload.

Simulation parameters and workload data have been obtained by conducting experiments on real hardware, and by collecting statistics from a web space provider. The evaluated cluster is constructed of BeagleBoards [4] equipped with the ARM Cortex-A8 CPU. The chosen platform was selected based on its low price, energy efficiency and performance. By running several simulations on data samples containing 30 minutes of web statistics, we obtained a potential energy reduction of up to 62 percent compared to a similar system that only uses DVFS.

II. Related Work

The authors of [5] suggests a computational environment consisting 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 match the system capacity with the 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. Four different control algorithms for adapting the capacity to the workload were presented and evaluated in the paper.

The authors in [6] also uses per-core sleep states to reduce the energy consumption in high-end server CPUs. The control algorithm used a simple high/low watermark on each CPU to decide which CPU core should be active. The obtained energy reduction for the system was claimed to be 40 % higher than a system with only DVFS available.

We intend to create fast, scalable and efficient capacity controller with control theoretic methods as basis. We argue that the use of the PID-controller could create a near optimal adaption of system capacity to the workload.

To determine the needed capacity of the system Bertini et. al [7] used tardiness for setting the needed performance by altering the CPU frequencies in a multi-tier cluster. Furthermore, the work in [8] presents an energy manager for server clusters based on a power model combined with a closed-loop controller to control the performance with sleep states and DVFS. In this work high-end CPUs were used evaluated with different energy policies and wake-up times were not considered.

Our sleep state-based system in contrast operates with a granularity of seconds and the wake-up time of cores highly influences the system. Our manager and architecture consist only of low-end embedded processors to give a distributed view of the system, and to adapt the manager to future many-core architectures. A combination of using sleep states to reduce energy consumption, the theory of the PID-controller to drive the capacity and the distributed architecture could decrease energy consumption without a substantial performance penalty.

III. Simulation Framework

A. Overview

The dominant consumer of energy on the aforementioned board is the CPU core [9], which our research focuses on. A simulation framework was created in Simulink to simulate and calculate the total energy reduction of the boards induced by using the power manager. The framework minimizes the total energy consumption by deactivating cores while maintaining the required QoS (Section III-B).

The basic structure of the framework is illustrated in Figure 1. The structure consists of a closed-loop system with an input, a PID-controller that controls the system capacity, and an output. The basic processing element in this paper is referred to as a *core*, since embedded systems with multi-core configurations have recently been available. The output of the system is used to determine the amount of cores needed to serve all requests.

Since a sleeping node will not be able to act as the power manager, all state changes will be based on decisions from a monitor node in the cluster i.e. the system level power manager. By running the manager on system level, decisions for power management will benefit the whole cluster instead of only a local node and thus reach closer to a global energy optimum.

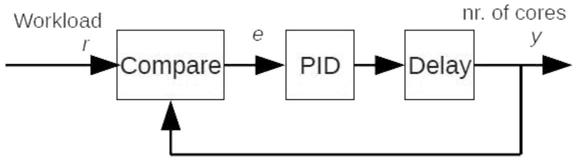

Figure 1. Basic structure of the power management system

B. Performance and quality

The simulation framework compares system capacity, i.e. how many requests the system can handle in a certain time, and the current workload. This comparison is taking place in the *Compare* block (Figure 1), which calculates difference between these two values. Incoming request are being spread out and processed in the web cluster in certain *time frames*. The granularity of the framework is therefore the length of one time frame.

QoS is a metric that is fully implementation dependent. The term describes how well an application is performing compared to its specification. QoS is usually used in soft real-time systems, in which the deadlines are set based on the human usability (or other subjective matters) of the system. Our system uses QoS to give a notion of latency of the request sent to the web service. A QoS drop occurs when a request in a certain time frame is not handled before the end of the time frame. This/these requests are then added to the next time frame and the QoS drops with a certain factor. Our definition of QoS states that as the workload exceeds the system capacity in a time frame, the QoS will drop. Eq. 1 shows the relation between QoS, capacity and workload.

$$QoS = \left(1 - \frac{W - C}{W}\right) \cdot 100 \qquad (1)$$

where W is the current workload and C is system capacity. The magnitude of the QoS drop is simply based on how many of the incoming requests were not handled in one time frame. The QoS is shown as a percentage. The maximum QoS value of 100 % means that the system provides the capacity to handle the whole workload in the measured time frame.

C. Switching delay

Our power manager works within the granularity of seconds. Since switching on cores is not instantaneous, the simulation must contain a delay for changing the CPU states. The algorithms in the PID-controller as well as measurements of the output signal also adds to the overhead of adapting the system capacity to the incoming workload. This overhead is represented in the simulation by inserting a delay block after the output of the PID-controller as shown in Figure 1. The delay can be adjusted in the simulation framework to represent different system configurations.

D. PID-controller

The PID-controller (Figure 1) is a common module in many control systems. It controls an output signal y depending on the input signal r and the controller settings. The difference between r and y is called the error value e, and is measured by using a feedback loop. The goal of the PID-controller is to minimize the error value and achieve equilibrium in the system.

The behavior of the PID-controller is determined by setting P, I and D values in the controller. These values

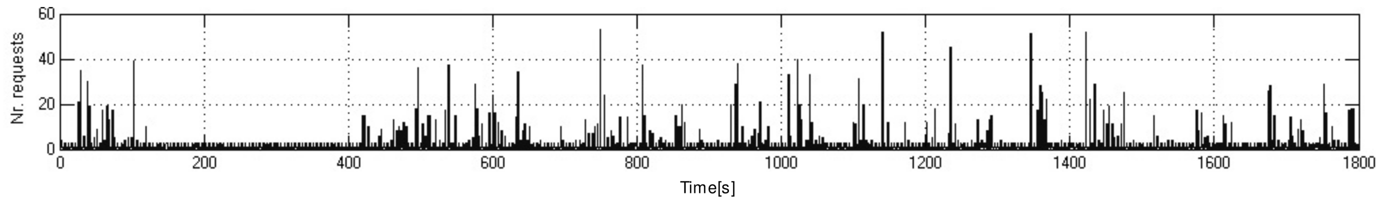

Figure 2. Workload sample from [10] 1. November 2010

choose how the output signal should react to changes in the input signal. The proportional part of the controller is set by the P parameter, which determines how fast or aggressive the controller reacts to changes in the input signal. I is the integral part of the controller. The main function of the integral value is to ensure that the process output agrees with the set point value in steady state. The derivative value D determines how the system reacts to changes in the reference value. By using a derivative term, the future of the reference curve is predicted based on previous values. The derivative term also enhances stability in the system [11].

The PID-controller is used in our power manager to select the amount of active CPU cores needed to process all requests in a time frame – this means that the PID-controller strives to activate only the minimum amount of cores and therefore minimize the energy consumption.

E. System capacity

The output of the system shown in Figure 1 is the current number of active CPU cores per time frame, which is determined as the output from the PID-controller.

Furthermore, the simulation framework supports the usage of *statically* active cores. These cores will be active and run on highest frequency completely independent of the control system. A high number of statically active cores allows the system to instantaneously being able to process the work between workload peaks. Workload peaks will decrease the QoS because of the delay the power management system introduces before it accommodates to the work peak. A high number of static cores will therefore slow down the QoS decrease during such a period, but will increase the average power dissipation of the cluster.

F. Final energy consumption

The simulation framework calculates the energy consumption for each time frame. The energy consumption is derived from the amount of active cores multiplied by the power dissipation of a core. We make the assumption that each core has two different states: *running* or *sleeping*. Since the board itself (with the CPU excluded) dissipates a small amount of energy, the power dissipation of the whole board is included in the output of one core for simplicity. The obtained power dissipation values were measured on the BeagleBoard with the DSP and the display subsystems disabled.

IV. Simulation Data

In order to simulate a realistic situation we conducted experiments to determine the parameters and settings for the simulation framework.

A. Web server requests

The web server requests used in the simulations were derived from [10] which is a Finnish web space provider. These http requests were addressed to over 750 websites and 510 domain names. By using data from an existing web space provider, we created a realistic situation for simulation. The workload curve pictured in Figure 2, relates to the number of http requests in a daytime sample from 1. November 2010. The curve shows, on average, a low workload with high peaks concentrated into certain time intervals. 30 minute samples were collected on the same date from the aforementioned server, and used as workload in the simulations. The data is freely available from [12].

B. PID parameters

As mentioned, the PID parameters determine how the controller reacts to changes in the input signal. This means that finding the appropriate parameters for the PID-controller is essential for having good regulation.

Several control methods for tuning PID parameters exist, and we will here focus on two common methods based on the frequency response of the closed loop system. Frequency response-based methods define the PID-parameters by determining the critical gain k_c in the closed-loop system. k_c is determined by increasing the controller gain until the output is on the border to instability, after which the period of the output signal t_c can be estimated. When these two parameters are determined, design recommendations are used to calculate the PID-parameters.

C. Ziegler-Nichols' frequency response-based recommendation

Ziegler-Nichols methods [11] were designed to give a good output response to load disturbances. This design recommendation is considered to give an aggressive controller with the risk of heavy overshoots, which means that our power manager will strive to quickly adjust the output resulting in fast reaction time and high overall QoS. Overshoots are a result of the control signal reaching over the desired set value to a certain amount before the controller stabilizes to the set value. This effect can cause slight energy waste because of unnecessary resource allocation.

Values for the PID-parameters, based on k_c and t_c can be obtain from Equation 2.

$$\begin{aligned} P &= 0.6 \cdot k_c \\ I &= \frac{1}{0.5 \cdot t_c} \\ D &= 0.12 \cdot t_c \end{aligned} \tag{2}$$

D. Åström-Hägglund's frequency response-based recommendation

The Åström-Hägglund method [13] also uses the parameters k_c and t_c obtained from the critical gain experiments to define the controller parameters. Furthermore, a constant κ has been defined through experiments and optimizations and is considered to give the system more robustness. κ is defined as :

$$\kappa = \frac{1}{K_p \cdot k_c} \tag{3}$$

where K_p is the process gain and k_c is the critical gain. This design suggests PID-parameters defined as:

$$\begin{aligned} P &= (0.3 - 0.1 \cdot \kappa^4) \cdot k_c \\ I &= (\frac{0.6}{1 + 2 \cdot \kappa})^{-1} \cdot t_c \\ D &= \left(\frac{0.15(1-\kappa)}{1 - 0.95 \cdot \kappa}\right) \cdot t_c \end{aligned} \tag{4}$$

The integral part in a PID-controller can cause problems when the input signal has great disturbances as the case shows in Figure 2. *Integral windup* is a phenomenon where the integral term accumulates a significant error during an overshoot. We have chosen to neglect the I-term completely to solve this problem. The nature of the power manager makes it possible to ignore the static control error that would otherwise have been eliminated with the I-term. This is due to the fact that the web cluster uses a discrete amount of cores and is not disturbed by a steady state value that is slightly off the set value. The implementation of the controller without an I-term will also be simpler with less calculation overhead. The result is actually a controller of PD-type, which is equal to a PID-controller with the I-term set to zero.

E. Static cores

The simulations were run with different configurations of static cores in order to measure the impact on the result. We used one to four static cores in different simulations. All four combinations were also simulated together with the different PID tuning methods to give a result on the energy and QoS relation between methods and static cores.

F. 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 [4]. 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 values from this experiment will be used to simulate the energy reduction using DVFS as power manager compared to the proposed PID-controlled power manager and a system without power management. The OPPs were accessed through the Linux ACPI. To avoid unwanted energy consumption, the display and DSP subsystems of the TI-OMAP3530 were 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. Furthermore, the power dissipation of a board with a sleeping core was

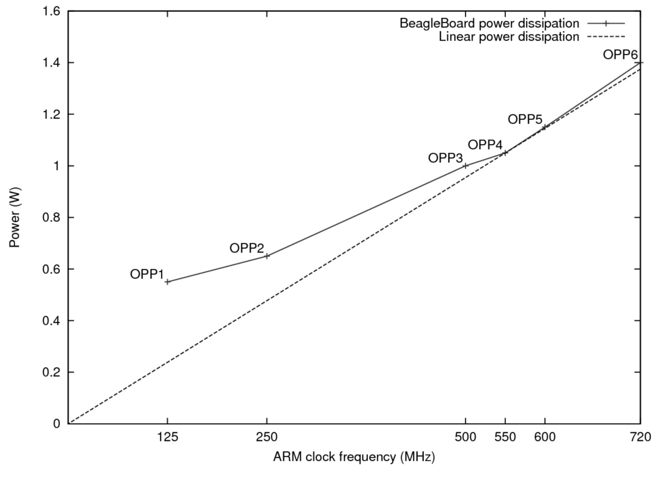

Figure 3. Non-linear power scaling by using DVFS

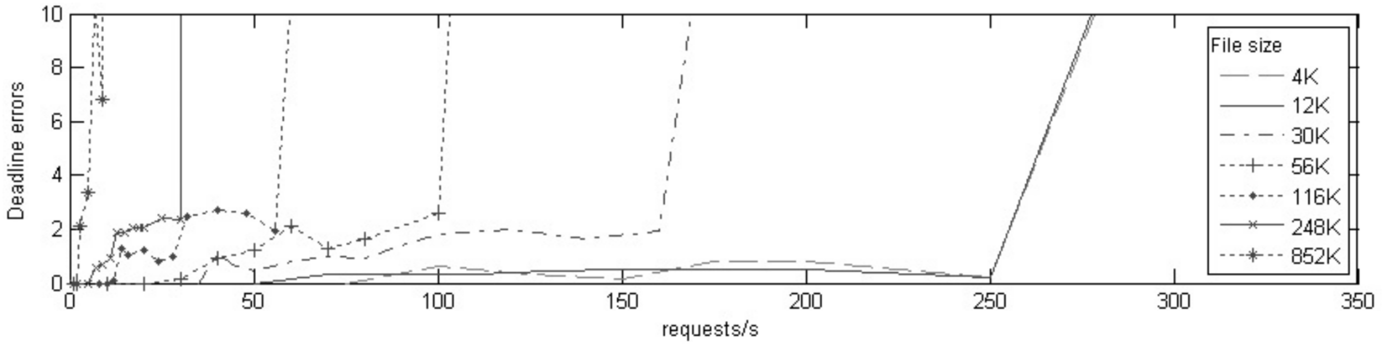

Figure 4. Capacity test for the BeagleBoard using Autobench

Table I
POWER DISSIPATION OF THE BEAGLEBOARD

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

measured to dissipate 0.2 W. Detailed information of this experiment can be found in [14].

The Table I and Figure 3 clearly show that the power dissipation does not drop linearly according to the clock frequency. Therefore, we intend to explore the possibility of using sleep states instead of DVFS as power management.

G. BeagleBoard wake-up time

To measure the wake-up latency we configured the system as illustrated in Figure 5. The expansion pin 23 of the BeagleBoard was set to alternate between logic ’1’ and logic ’0’. To initiate the wake up the system, the voltage on expansion pin 8 was set high. This will cause an interrupt that wakes up the system. The oscilloscope was connected to expansion pins 23 and 8. A transition from ’0’ to ’1’, i.e. the wake-up signal, on pin 8 was set to trigger the oscilloscope. The wake-up time for the BeagleBoard was on

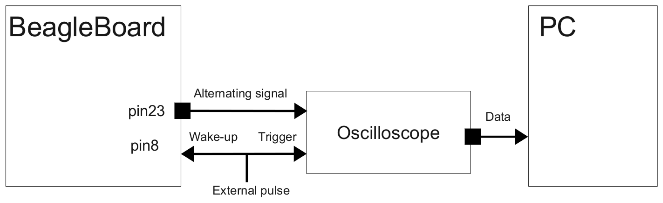

Figure 5. Schematic of wake-up tests

average measured to be 650 ms – with a standard deviation of 50 ms. Based on this measured wake-up time we set the transition delay in the simulation framework to 1000 ms to accommodate for overhead related to other eventual factors.

H. 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 conducted on a BeagleBoard to give the number of requests per second a BeagleBoard could handle. The test tool in use was Autobench [15] which generates requests to an Apache server running on the BeagleBoard. The number of requests per second generated by Autobench started from a selected number and increased by specified increments. When the number of requests per second start to exceed the capacity of the host, deadline *errors* will start to occur as the selected deadlines for the requests are not met. The selected deadline in our experiments was one second, in order to match the time frame of the workload described in section III. Moreover a range of files each request needed to process was chosen and shown in Table II. The table also shows at what point deadline errors start to occur for the different file sizes. The file sizes were selected based on typical file sizes used in a web server.

The result of the experiment showed a certain error rate produced when the requests were not processed within the given time interval of one second.

Table II shows that a BeagleBoard in general can handle between 75 and 2 requests per second without errors, when using file sizes between 4 KB and 852 KB. A large file size such as 2.4 MB will produce large errors already after one request per second; this implies that experiments with larger file sizes are not needed. Figure 4 illustrates how the errors increase according to the increasing request rate. The curves represent the outcome of different file sizes.

Table II
MEASURED CAPACITY OF THE BEAGLEBOARD

File size [KB]	4	12	30	56	116	248	852	2400
Max [req/s]	75	50	35	20	10	5	2	0

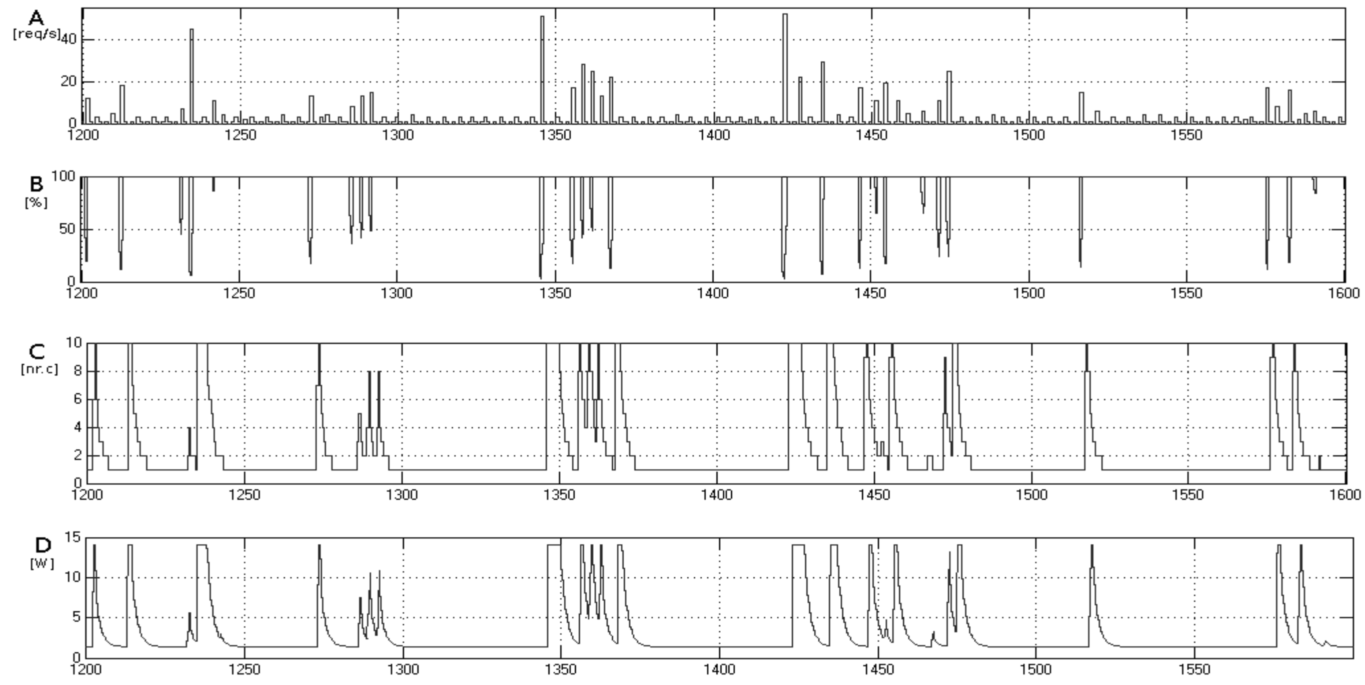

Figure 6. Results from simulation with Åström-Hägglund recommendations and 1 static core. A: Incoming requests, B: QoS value, C: Number of cores in use, D: Energy consumption

Related experiments in [5] result in a capacity of 5 requests per second for the BeagleBoard, which in our test setup would compare to a file size of 248 KB. The selected file size for our simulations was therefore chosen to be 248 KB. The file size is constant for each simulation with a maximum amount of 10 cores available.

Altering file sizes is a typical real-world scenario for web servers – this case is a general load balancing problem [16] and it has not been focused on in our simulations. Future research is needed to determine the impact of altering file sizes on the system.

V. Simulation Results

A. Comparisons

In order to draw a conclusion about the efficiency of our power management, our simulations should be compared to other power management systems.

Since existing systems implement power management such as DVFS, we also need to compare the final results with a 10 core system which is able to dynamically scale down its voltage and frequency. We created a framework for this purpose. Our simulations used the OPPs and the corresponding power dissipations presented in Table I. The simulation results show a typical 45 % energy reduction with DVFS enabled, compared to a system without power management. During a 30 minute run using a 10-core cluster, a system without power management would consume:

$$E_{full} = 10 \cdot 1.4W \cdot 30 \cdot 60s = 25200J \tag{5}$$

When enabling DVFS the energy consumption was reduced to $13558J$.

B. Our simulations

Given the values from the measurements we set-up the simulation framework as a 10-core system. The simulations were run for both tuning methods: *Ziegler-Nichols* and *Åström-Hägglund* – using the file size of 248 KB. This simulation was run four times, to use all combinations of static cores (1,2,3 and 4). The average QoS and total energy consumption was calculated and stored for all combinations of settings.

Figure 6 shows results from a simulation, which used the *Åström-Hägglund* method and one static core. The graphs in the figure are only showing the time interval [1200 1600]s for illustrative reasons. The graph in Figure 6(A), shows the incoming requests to the service. The corresponding QoS is presented as percentage in Figure 6(B), the number of currently active cores is shown in Figure 6(C) and the final power dissipation in Figure 6(D).

Table III shows the result for all simulations. The result consists of two important values: QoS and energy consumption. The values change depending on the used tuning method and amount of static cores. As seen in the table, both the QoS and energy consumption will steadily rise when switching on more static cores or using the more aggressive Ziegler-Nichol method.

Table IV compares the energy reduction between Ziegler-Nichols' and Åström-Hägglund's frequency response rec-

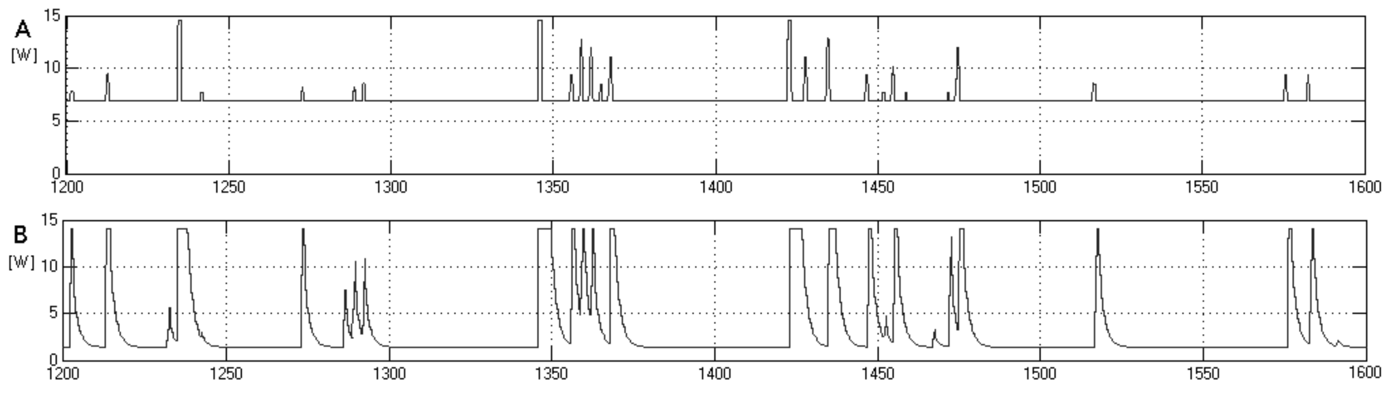

Figure 7. Energy graph comparing A: DVFS and B: sleep states

Table III
RESULTS FROM SIMULATIONS USING ZIEGLER-NICHOLS' AND ÅSTRÖM-HÄGGLUND'S FREQUENCY RESPONSE RECOMMENDATIONS. QOS [%] / ENERGY [J]

Static cores	Z-N	Å-H
1	96.8 / 6304	96.3 / 5190
2	98.1 / 7508	97.8 / 6746
3	98.9 / 9075	98.8 / 8612
4	99.2 / 11105	99.2 / 10787

ommendations for PID tuning. As expected, the more aggressive tuning method will result in less energy reduction, but as was seen in Table III the QoS was overall higher. By comparing the results with the reference value $13558J$ (obtained by only using DVFS), one can clearly state that a power management system that uses sleep state could reduce the energy consumption significantly compared to a system without a power management or to a system using DVFS as power management.

Table IV
TOTAL ENERGY REDUCTION FOR DIFFERENT CONTROL METHODS COMPARED TO A SYSTEM USING DVFS. THE TABLE SHOWS THE ENERGY SAVINGS IN %

Static cores	Z-N	Å-H
1	53.5%	61.7%
2	44.6%	50.2%
3	33.1%	36.5%
4	18.1%	20.4%

Figure 7 displays clearly the reason why sleep states will reduce the energy consumption more than DVFS. Part (A) in figure 7 shows the power dissipation for system in time range [1200 1600]s using DVFS as power management. In comparison, part (B) shows how sleep state-base power management allows the system to drop the power dissipation much more than DVFS, and therefore resulting in lower energy consumption.

Figure 8 shows a chart comparing the tuning methods to DVFS in terms of energy consumption and QoS. Naturally

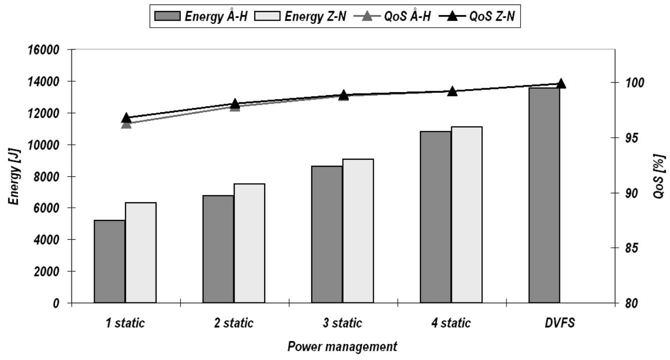

Figure 8. Energy chart comparing DVFS and sleep states

the sleep state-based power manager behaves more like DVFS when adding more static cores since more cores will then be constantly active. The most energy is saved when using the Åström-Hägglund method with one static core.

The lowest QoS will arise using a more conservative tuning method such as *Åström-Hägglund* and few static cores. Despite these constraints the simulations show that the QoS will only drop by approximately 4 % as seen in Table III. We assumed here that DVFS has 0 % drop in QoS. The table also shows that switching to a more aggressive tuning method (such as Ziegler-Nichols) and increasing the number of static cores will increase the QoS to over 99 % if requested.

VI. CONCLUSIONS

We have presented a power management system for many-core clusters. The power manager uses sleep states to match the capacity of the system with the workload to minimize the energy consumption while keeping the system performance on an acceptable level.

We have developed a simulation framework to evaluate the efficiency of the power management system. The framework reads a workload as input and shows, for each time frame, the results namely: number of cores in use, the QoS, and the energy consumption. Finally the framework shows the total energy consumption and the average QoS over the whole simulation.

The amount of active cores in the cluster is determined by a PID-controller that based on the well defined recommendations from Ziegler-Nichols and Åström-Hägglund methods drives the capacity of the system to just the necessary minimum.

Our simulation framework is based on parameters from the BeagleBoard equipped with an ARM Cortex-A8 processor. The capacity, power dissipation and wake-up time of the BeagleBoard was measured by experiments – this gives a realistic simulation and comparison of efficiency.

The results of our simulations show that our power management has the potential to reduce the energy consumption by 18 to 62 percent depending on the desired QoS, compared to a system that uses DVFS as power management. The reduction in energy arises from the fact that typical web servers have long idle times during which the full capacity of the system is not needed. Energy consumption can be reduced by replacing high-end server CPUs with clusters of low-end boards. The achieved granularity of low-end boards gives arise to extensive power management on system level which scales in a large data center.

VII. Future work

The power manager is currently being implemented on real hardware that uses the aforementioned CPU configuration. The real implementation will show the relation in energy consumption and an exact value of the introduced overhead of communication, algorithms etc.

We intend to further improve the power management for more intelligent control. Our simulations presented in [3] shows that static PID parameters is not sufficient to control all environments. Self-tuning regulators is described in [17] and could provide the necessary properties to adapt the PID-controller to heavily changing workload patterns.

Currently the framework does not support any tools to ensure a maximum latency for a request. By ensuring the maximum latency, a user can be guaranteed to receive a reply from the server in a certain time interval. PID-parameters can be influenced of this additional requirement and eventually self-tune to provide sufficient CPU power.

The file sizes used in the simulations have so far been constant. To compare the simulation to a further realistic case, the file sizes should change dynamically during the simulation according to an appropriate distribution function. The simulation framework can also be extended to explore situations of stochastic file sizes.

References

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

[2] 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).

[3] S. Holmbacka, S. Lafond, and J. Lilius, "Power proportional characteristics of an energy manager for web clusters," in *Proceedings of the 11th International Conference on Embedded Computer Systems: Architectures Modeling and Simulation*. IEEE Press, July 2011.

[4] *OMAP35x Technical Reference Manual*, Texas Instruments Incorporated, July 2010.

[5] A. Krioukov and P. Mohan, "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.

[6] J. Leverich and M. Monchiero, "Power management of datacenter workloads using per-core power gating," *IEEE Computer Architecture Letters*, vol. 8, pp. 48–51, 2009.

[7] L. Bertini, J. Leite, and D. Mosse, "Siso pidf controller in an energy-efficient multi-tier web server cluster for e-commerce," in *Second IEEE International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*, Munich, Germany, June 2007.

[8] T. Horvath and K. Skadron, "Multi-mode energy management for multi-tier server clusters," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*. New York, NY, USA: ACM, 2008, pp. 270–279.

[9] S. Madhavapeddy and B. Carlson, *OMAP 3 Architecture from Texas Instruments Opens new Horizons for Mobile Internet Devices*, Texas Instruments Incorporated, 2008.

[10] "Kulturhuset," http://kulturhuset.fi/start/, January 2011, [Online; accessed 31-May-2011].

[11] K. J. Åström and T. Hägglund, *Automatic tuning of PID controllers*. Instrument Society of America, 1988.

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

[13] K. J. Åström and T. Hägglund, *Advanced PID control*. Research Triangle Park, 2006.

[14] J. Smeds, "Evaluating power management capabilities of low-power cloud platforms," Master's thesis, Åbo Akademi University, Finland, 2010.

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

[16] E. Musoll, "Hardware-based load balancing for massive multicore architectures implementing power gating," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 29, pp. 493–497, March 2010.

[17] J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, *Feedback Control of Computing Systems*. IEEE Press, 2004.