System Level Power Management for Many-Core Systems

Simon Holmbacka [#1] Jens Smeds [#2] Sébastien Lafond [#3] Johan Lilius [#4]
[#] *Department of Information Technologies, Åbo Akademi University*
Joukahaisenkatu 3-5 FIN-20520 Turku
firstname.lastname@abo.fi

***Abstract*—This paper introduces a framework simulating a system level power management for many-cores targeting server cards used in warehouse-sized data centers. The energy consumption for different operating performance points of a TI-OMAP3530 chip was measured and is used by the framework to estimate the system energy consumption. The proposed system level power management shows a typical 34 % energy saving for randomly generated requests compared to a system lacking any power management.**

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. A cluster of mobile CPUs can provide the same computational power as server-grade CPUs, but with lower total energy consumption. The usage of mobile CPUs also aims at obtaining cheaper and physically smaller individual server cards by minimizing the need of cooling infrastructure. With an average of 10 to 50 percent CPU utilization for servers [1] and a large load fluctuation found in typical web services [2] the use of slower but more energy-efficient cores enables system level power management, which dynamically matches the load fluctuation to the computational capacity, at a much finer granularity than server-grade cores.

This paper presents a system level power management for many-core server cards consisting of mobile CPUs. Compared to related works [3][4], the presented power management system uses both sleep states and dynamic voltage frequency scaling (DVFS) to continuously match the work-load while minimizing the system power consumption. Moreover the presented framework uses control theory and the features of the PID controller as basis for anticipating the work-load and switching between sleep states as well as CPU frequencies.

II. Simulation framework

A simulation framework was created to simulate and measure the total energy reduction induced by using the power management. The framework will minimize the total energy consumption by disabling cores while maintaining a QoS defined by the user. The framework consists of three blocks shown in Figure 1.

a) ***System monitor:*** The monitor block is the source of information on which the other blocks are applying functionality. The input to the monitor, shown in Figure 2(A), is randomly generated requests to the service. The system will dynamically adapt to handle all the incoming requests by monitoring the quality of service (QoS), presented in Figure 2(B). The QoS is determined by calculating the difference between the incoming requests and the current capacity of the system. The requests that are not being handled due to insufficient capacity will be delayed and added to the job queue of the next iteration.

b) ***Dynamic core allocation:*** The core allocation block in this framework will during run-time add or subtract cores to or from the system. As the QoS value describes the performance, one or more cores are added to the system according to the QoS value which is the result of the control error provided by the feedback loop in Figure 1. The allocation block can determine the trend of both the incoming request curve and the QoS curve. The trends of both curves are determined by the derivative of the functions with respect to time. By following the trend and enabling cores before receiving a large set of requests, the QoS will not suddenly drop. If the QoS value remains high while the request curve drops the system reduces the amount of cores and thus reduces the energy consumption. This action is taken if the QoS curve remains high for a longer time, which is determined by integrating the QoS curve over a time window. Depending on the received integral value, the system disables a certain amount of cores. Additionally the allocation block monitors the trend of the request curve to get an insight into the demand of future performance.

c) ***System capacity:*** The capacity of the system is dependent on the amount and the capacity of the cores in use. The capacity block sends the current capacity of the system as feedback to the monitor to use in the next iteration of the power management system.

The framework supports frequency scaling. As the framework is assuming an optimal allocation of tasks on the cores, the load of all cores, except one, is 100 %. The last core enabled, which is eventually not fully loaded, can scale down its frequency to reduce the energy consumption.

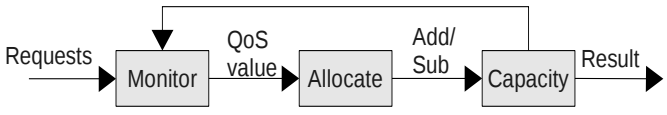

Fig. 1: Main blocks in simulation framework

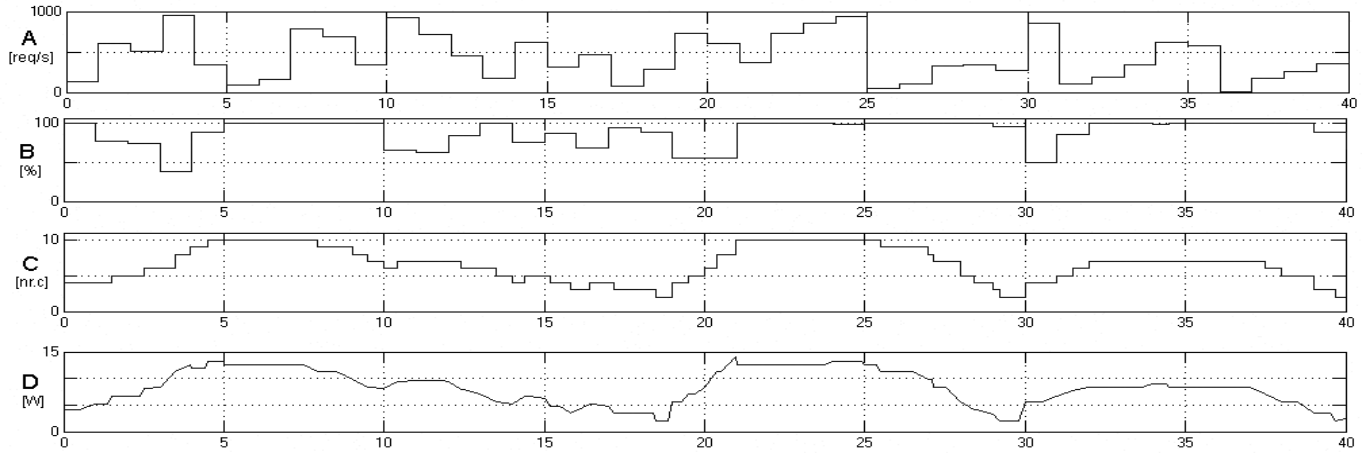

Fig. 2: Results from simulation. A: Incoming requests, B: QoS value, C: Number of cores in use, D: Energy consumption

III. Simulation parameters

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. The system ran Angstrom 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 idle power for 720 MHz was measured to be 1.2 W.

IV. Results

By generating requests to the system, the energy consumption for the many-core system can be estimated. The result in Figure 2(C) shows that the number of cores changes dynamically according to the variance in load and QoS during the simulation of 40 seconds. Figure 2(D) shows that dynamic core allocation will reduce the overall energy consumption of the system. The QoS can occasionally drop to low level, as seen in Figure 2(B), even though the system is anticipating the trends both in terms of requests and QoS. By increasing the P, I and D values in the controller, the system is set to prioritize higher performance instead of low energy consumption and vice versa by decreasing the values. In order to estimate the energy reduction, we calculated the energy consumption for the system without any power management applied and run the simulation several times. As a core uses 1.40 W of power during full load and 1.2 W of power idling, the total energy reduction for the incoming requests is on average 34 % lower than the energy consumed by a system without power management. The system without power management is assumed to have 10 cores statically allocated and no DVFS.

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

V. Conclusion and future work

We have shown that it is possible to reduce the energy consumption while maintaining the needed system performance in a many-core system by using dynamic core allocation. The strategy of dynamic core allocation is to enable and disable CPU cores during run-time in order to reduce energy consumption. The simulation framework uses a control system which uses features from the PID controller to anticipate the incoming request curve and match the system capacity accordingly. We intend to further improve the system by creating a more intelligent core allocator and include migration of tasks between cores during run-time. Frequency scaling for every core in the system would also improve the conservation of energy during a non-optimal distribution of tasks between the cores.

References

[1] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33 –37, 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] "Feedback edf scheduling exploiting dynamic voltage scaling," in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE Computer Society, 2004, pp. 84–.

[4] 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. ACM, 2010, pp. 15–22.