High Performance Computing on Low Power Devices

Vojtech Nikl
2^{nd} year, full-time study
Supervisor Jiri Jaros

Brno University of Technology
Faculty of Information Technology
Bozetechova 2, Brno, Czech Republic
inikl@fit.vutbr.cz

***Abstract*—Nowadays, the power efficiency of modern processors is becoming more and more important next to the overall performance itself. Many programming tasks and problems do not scale very well with higher number of cores due to being memory or communication bound, therefore it is often not beneficial to use faster chips to achieve better runtimes. In this case, employing slower low power processors or accelerators may be much more efficient, mainly because it is possible to get the same results using much less energy. Dynamic runtime adjustments applied to the system based on the properties of a given algorithm, such as frequency and voltage scaling or switching off unneeded parts, may further enhance power efficiency. This paper describes the benefits of using low power chips for building an HPC cluster, the group of algorithms where this approach can be useful, possible system adjustments towards better power efficiency, results achieved so far and future plans.**

***Keywords*—HPC, parallelism, low power, processor architecture, supercomputers, k-Wave, MPI, OpenMP, performance evaluation, numerical methods, clocking**

I. Introduction and Motivation

Even though computer processors have come a long way in terms of performance, there are still many tasks and problems which require large amounts of computing power to be successfully solved. For some time, hardware engineers haven't been purely focusing on raw performance, but the energy consumption has also become a very important factor. Using low powered processors can be much more efficient for certain kinds of algorithms, mainly for memory and communication bound problems. Unfortunately, this is where algorithms' scalability come into play. Underclocking processsor's cores or switching them off during these computationally non-intensive events may bring further energy benefits.

Today's supercomputers are usually based on the x86 architecture, specifically the Intel Xeon Sandy Bridge or newer. One of many examples is the Anselm cluster[1], located in Ostrava, Czech Republic. It consists of 209 2×8-core Intel Xeon E5-2665 2.6 GHz nodes, each with atleast 64 GB of RAM. Each node requires approximately 230 W of energy under full load, while providing about 400 GFlop/s of theoretical performance in 64-bit double precision. Most of that energy is dissipated into heat and therefore requires very intensive and expensive cooling system. Some systems chose a little bit different approach towards higher power effectiveness. One of them is the Fermi cluster[2], located at the CINECA consortium, Bologna, Italy. This cluster, on the other hand, consists of 10,240 nodes, each integrating 16-core IBM PowerA2 1.6GHz processor. While the overall performance per node is about half of the Anselm one's, the peak power consumption is only 55W. This results in almost twice as good performance per Watt ratio. *The Green 500 list*[3] provides a ranking of the most energy-efficient supercomputers in the world and Fermi is close to the top at the 59th place, having over 2 GFlop/s per Watt. The most efficient supercomputer has about 7 GFlop/s per Watt (January 2016). Current estimates indicate that processor efficiencies will have to evolve from the current 5 GFlop/s per Watt to 50 GFlop/s per Watt for exascale machines to be viable [1], mainly because a realistic power budget for an exascale system is 20 MW.

The Mont-Blanc project[4] [2][3] is aiming to design a new type of computer architecture capable of setting future global HPC standards, built from energy-efficient solutions used in embedded and mobile devices. The project is run by the Barcelona Supercomputing Center[5] and is funded mainly by the European Commision. The main focuses of development are the *OmpSs* parallel programming model to automatically exploit multiple cluster nodes, transparent application check pointing for fault tolerance, support for ARMv8 64-bit processors, and the initial design of the Mont-Blanc Exascale architecture. The main goal is to design a new high-end HPC platform that is able to deliver a new level of performance/energy ratio when executing real applications, that should provide exascale performance using 15 to 30 times less energy.

Unfortunately, the approach of using ARM based kits has a

[1]https://docs.it4i.cz/anselm-cluster-documentation/hardware-overview

[2]CINECA consortium, IT, http://www.hpc.cineca.it/content/ibm-fermi-user-guide

[3]http://www.green500.org/

[4]https://www.montblanc-project.eu/

[5]https://www.bsc.es/

few downsides. Since the low power processors are generally less powerful, it is necessary to employ much more of them to reach the same level of overall system performance, however algorithms may have trouble scaling that high.

Another problem may be the amount of system memory, where ARM based kits offer only 1–4 GB, which can quickly become limiting for extensive simulations.

For communication-intensive algorithms, the mostly equiped 1 Gbit/s network cards are also going to be quite limiting.

Last but not least, one more important reason to focus more on power effectiveness is the resource allocation policy of supercomputing centers. Currently, resources are distributed between users based on *core-hours*, however in the future, users will most likely be billed based on consumed kWhs, which is going to put much more pressure on algorithms' efficiency.

II. Power Consumption Analysis

In this section, frequency scaling benchmarks on x86 Intel Xeon Haswell architecture are presented, as the first step of understanding algorithm's power consumption properties. The purpose of these tests is to show theoretical performance impact on both compute and memory-bound problems, the scaling of power consumption of both the processor and memory modules in relationship to performance, define power efficiency of different benchmarks in relation to frequency scaling and make a conclusion about suitability of these tests in regards to low power architectures.

The system on which all the benchmarks were run is shown in Table I.

Server	Supermicro 7048GR-TR
Motherboard	Supermicro X10DRG-Q
Processor	2x Intel Xeon E5-2620v3
RAM	DDR4-2133 64GB (4 channels)
SSD	Crucial 250GB

TABLE I: System hardware overview.

The operating system is *Ubuntu 14.04* with *3.19.0-51-generic* kernel version.

The energy measurements were taken using the PAPI library[6] and its RAPL framework [4], which can directly access the hardware counters of the CPU. PAPI measures the energy consumption of three main components of each CPU - *package*, *powerplane* and *dram*. Package measures the whole socket including the memory controller, powerplane measures only the cores themselves and dram measures the corresponding dram module. Powerplane measurements were not supported on our system (always returned 0 Joules), so only packages and drams were taken into account.

Our Haswell CPU supports frequencies ranging from 1.2 to 2.4 GHz, excluding Turbo mode. To be able to manually set chosen frequency, the Intel driver had to be replaced with the ACPI driver and the governor was set from *balanced* to *userspace* using the system's `cpupower` utility.

[6]http://icl.cs.utk.edu/papi/

All the benchmarks were compiled using the `GNU GCC 5.3.1` compiler. The flags used were

```
gcc -std=c99 -O3 -mavx -ffast-math
```

Three main frequencies for all the cores were chosen to be benchmarked, 1.2, 1.8 and 2.4 GHz. Turbo was turned off. Voltages for each of the frequencies were set automatically based on the default CPU stepping provided by Intel.

The total energy consumed of each benchmark was calculated by adding package0, package1, dram0 and dram1 power consumptions up.

The first set of tests is focusing on memory subsystem performance, using the `lmbench` [5] tool.

The lmbench memory bandwidth (see Fig. 1) benchmark shows that while every level of CPU cache scales almost linearly with the cpu frequency, the main memory bandwidth is affected very little, not more than 5%.

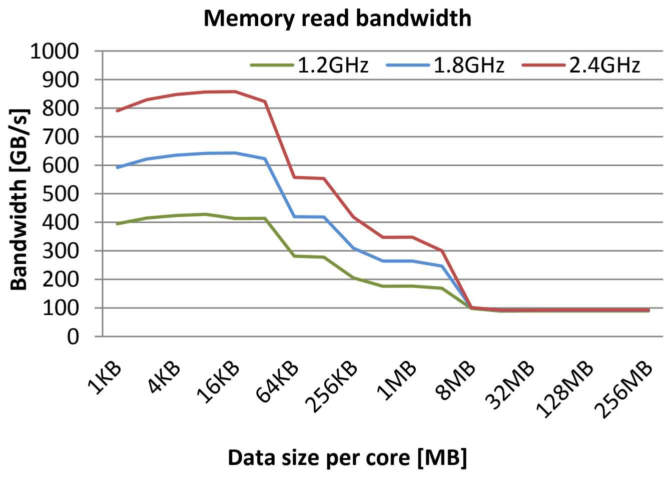

Fig. 1: Memory read bandwidth benchmark using lmbench.

When combined with the energy measurements, Fig. 2 presents the amount of GBs transfered per Watt during the memory read bandwidth benchmark. When the data fits into caches, 25% frequency drop results in 10–30% increased GBs per Watt ratio, in the main memory the increase is only 4–5%.

Combining the bandwidth and power consumption, Fig. **??** presents the amount of GBs transfered per Watt during the memory read bandwidth benchmark. When the data fits into caches, 25% frequency drop results in 10–30% increased GBs per Watt ratio, in the main memory the increase is 4–5%.

While previous benchmarks were focusing on the memory, the Linpack benchmark [6] is focusing on the raw cpu performance. Linpack solves a set of 2D equations based on factorization with $O(n^3)$ operation complexity. While the Flops per second metric should scale directly with the cpu frequency, it may not always be the case, mainly due to the cache and memory performance not scaling linearly with the frequency, so the slower running cpu may not wait for the data as long.

	Package0 [W]	Package1 [W]	Dram0 [W]	Dram1 [W]	Gflop/s	GFlops/W
1.2GHz idle	16.1	16.4	14.0	9.7		
1.2GHz burn	37.2	37.0	35.1	29.2	201.2	2.44
1.8GHz idle	17.0	17.0	14.0	9.7		
1.8GHz burn	48.6	46.7	36.1	30.1	250.6	2.39
2.4GHz idle	17.1	17.1	14.2	9.8		
2.4GHz burn	73.5	57.0	41.6	31.2	299.7	2.07

TABLE II: Linpack.

	Total time [s]	Simulation time [s]	Total energy [J]	Simulation energy [J]
1.2GHz	96.21	44.08	10135	5636
1.8GHz	82.56	38.88	9435	5023
2.4GHz	67.97	32.1	9636	5113

TABLE III: Power consumption of one k-Wave simulation, 576^3, 10 timesteps

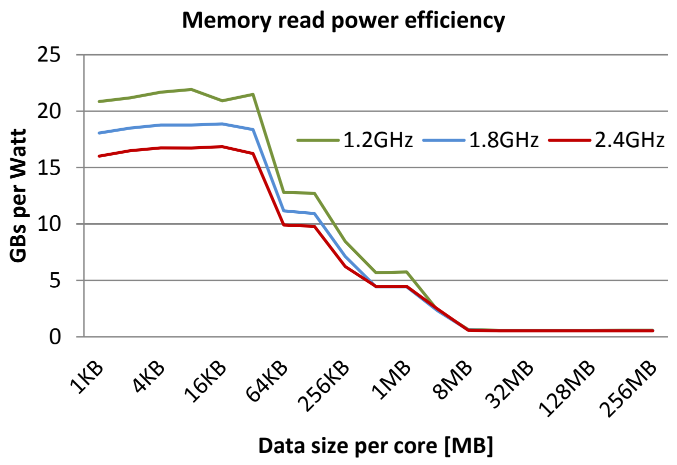

Fig. 2: Amount of data transfered from memory per one Watt.

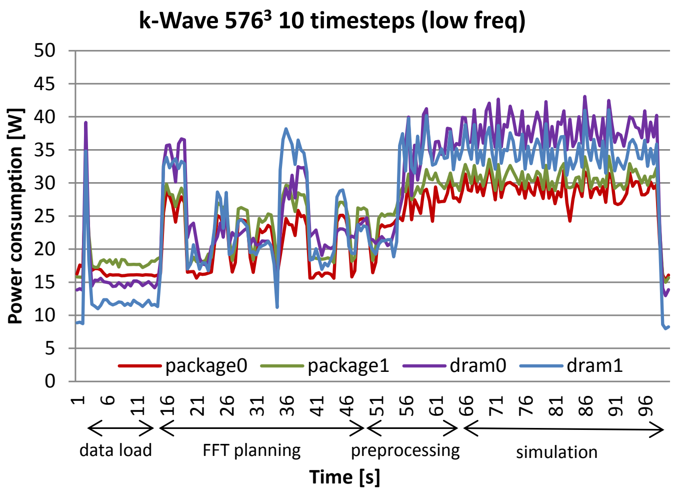

Fig. 3: k-Wave simulation, 576^3, 10 timesteps (1.2 GHz)

Table II shows the average power consumption during a single run (problem size is 25 000, leading dimension is 25 000 and memory alignment is set to 1 KB), overall performance in GFlop/s and power efficiency in GFlop/s per Watt. The best efficiency is achieved running on low and middle frequency, while the top frequency is slightly behind.

As a final practical application, the OpenMP version of k-Wave [7] was benchmarked. In [8] and [9], the scalability of Fast Fourier transforms (FFT) and whole k-Wave simulations, which implement these FFTs, was shown to reach 16 384 and 8 192 cores, respectively, with over 50% efficiency. In table III, total time and consumed energy of one k-Wave simulation of size 576^3 with 10 timesteps is measured. Fig. 3 shows the consumption progress of different parts of the system during the whole simulation. While the frequency drop from 2.4 GHz to 1.8 GHz slightly improved the overall power-efficiency, further drop to 1.2 GHz worsened the results. This is mainly because the dram consumption starts to dominate and the cpu is running too slowly and stalls the simulation. While previous benchmarks showed higher effectivity with lower frequencies, k-Wave's scaling ends when the cpu consumption drops below the dram consumption.

Overall these tests showed, that lowering the frequency of the cpu can bring significant improvements in performance-to-power ratio, mainly for tasks that are memory-bound and cannot benefit as much from fast CPUs and are therefore more likely suitable for low power architectures.

III. Goals of the PhD thesis

Goals

Show that certain classes of extreme-scaling algorithms used in HPC, mainly

- memory-bound,
- interconnect network-bound and
- I/O-bound

can objectively benefit from the use of low power achitectures and **dynamic power consumption optimization techniques** in

- total power consumed during the computation,
- lower cooling requirements, less expensive infrastructure at the massive deployment,
- total financial expenses,

while other important factors, such as performance, programming complexity or reliability are affected very little or not at all, compared to the current HPC clusters.

Achieving the goals

- Runtime algorithm analysis and profiling
  Analyze given algorithms and their runtime behaviour (power consumption, performance, level of utilization of different parts of the system, network communication,...) using hardware counters, profiling tools, etc. Based on the data obtained, a model describing the behaviour of a given algortihm and hardware setup is automatically created. This paper briefly describes the analysis run on the Intel Haswell architecture, where the frequency scaling showed only small improvements in terms of energy demands. Next step is to move to an ARM based kits, namely nVidia Tegra and Samsung Odroid.
- Dynamic on-the-fly optimizations
  Runtime automated measurements of performance (memory bandwidth, nop operations, utilization of cores,...) and power consumption, making power optimizations based on a decision model, such as dynamic on/off thread switching, over or underclocking of specific cores, etc.
- Network communication optimizations
  Automated runtime decisions whether it is beneficial to switch off or underclock parts of the system during intensive MPI communications, based on message sizes and quantities, considering the latencies of such operations.

IV. Conclusion

This paper described the motivation behind the suitability and the use of low power architectures for solving specific tasks, mainly the ones that are memory and/or communication bound, instead of the common architectures used today, mainly for overall power-efficiency and the use as basic building blocks for future exascale clusters. Power consumption analysis was presented on the Haswell architecture, which showed roughly 5% energy savings for 30% reduced frequency. Next step is to move to ARM-based kits, namely Samsung Odroid and nVidia Tegra.

Acknowledgment

This work was supported by the FIT-S-14-2297 Architectures of Parallel and Embedded Computer Systems project.

References

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, K. Yelick, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Keckler, D. Klein, P. Kogge, R. S. Williams, and K. Yelick, "Exascale computing study: Technology challenges in achieving exascale systems peter kogge, editor & study lead," 2008.

[2] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with commodity cpus: are mobile socs ready for hpc?" in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–12.

[3] N. Rajovic, N. Puzovic, L. Vilanova, C. Villavieja, and A. Ramirez, "The low-power architecture approach towards exascale computing," in *Proceedings of the Second Workshop on Scalable Algorithms for Large-scale Systems*, ser. ScalA '11. New York, NY, USA: ACM, 2011, pp. 1–2. [Online]. Available: http://doi.acm.org/10.1145/2133173.2133175

[4] V. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring energy and power with papi," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, Sept 2012, pp. 262–268.

[5] L. McVoy and C. Staelin, "Lmbench: Portable tools for performance analysis," in *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '96. Berkeley, CA, USA: USENIX Association, 1996, pp. 23–23. [Online]. Available: http://dl.acm.org/citation.cfm?id=1268299.1268322

[6] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003. [Online]. Available: http://dx.doi.org/10.1002/cpe.728

[7] B. E. Treeby and B. T. Cox, "k-wave: Matlab toolbox for the simulation and reconstruction of photoacoustic wave-fields," *J. Biomed. Opt.*, vol. 15, no. 2, p. 021314, 2010.

[8] V. Nikl and J. Jaros, "Parallelisation of the 3d fast fourier transform using the hybrid openmp/mpi decomposition," in *Mathematical and Engineering Methods in Computer Science*, ser. LNCS 8934. Springer International Publishing, 2014, pp. 100–112.

[9] J. Jaros, V. Nikl, and E. B. Treeby, "Large-scale ultrasound simulations using the hybrid openmp/mpi decomposition," in *Proceedings of the 3rd International Conference on Exascale Applications and Software*. Association for Computing Machinery, 2015, pp. 115–119.