On-Line Analysis of Hardware Performance Events for Workload Characterization and Processor Frequency Scaling Decisions

Robert Schöne
Technische Universität Dresden
Center for Information Services and HPC (ZIH)
robert.schoene@tu-dresden.de

Daniel Hackenberg
Technische Universität Dresden
Center for Information Services and HPC (ZIH)
daniel.hackenberg@tu-dresden.de

ABSTRACT

Energy efficiency optimizations of computational resources continue to be of growing importance for both classical datacenter workloads as well as high performance computing environments. New hardware generations introduce more and more energy efficiency features, resulting in a power consumption variation by at least a factor of four between idle and full load. Even the power consumption of different full-load workloads can vary substantially, clearly showing that there is energy saving potential apart from the traditional "race to idle". In this paper we present a configurable CPU frequency governor that adapts processor frequencies based on performance counter measurements instead of processor load. We use the SPEC OMP benchmark suite to determine the potential of our approach and present governor configurations for two up-to-date x86_64 microarchitectures. Moreover we show that substantial follow-up work is required to assess further efficiency optimization potential in this field.

Categories and Subject Descriptors

D.4.0 [**Operating Systems**]: General—*Linux*; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Performance, Measurement

Keywords

energy efficiency, DVFS, frequency governor, performance counter

1. INTRODUCTION

The past decade has brought some disruptive changes to the HPC community. While in previous years we focused strongly and often solely on performance, energy efficiency has now become a second and even equally important metric. The ever-increasing power consumption of commodity computer hardware forced vendors to build more sophisticated power saving capabilities into their hardware. Operating system developers are steadily adapting to these features in order to use hardware resources more efficiently. Most recently, middlewares and even applications are exposed to aspects that are relevant with respect to energy efficiency optimizations.

While these developments allow to run software more energy efficient, more progress needs to be made and newly emerging hardware features are continuously offering new opportunities. Among the power-saving techniques used in current processors is Dynamic Voltage and Frequency Scaling (DVFS), one of the major attempts to increase energy efficiency of computing systems by adapting the voltage and frequency of compute cores or other processor parts to the current system load. As the consumed power is proportional to the frequency and the squared voltage, a reduction in both significantly reduces the processors power consumption.

The Linux operating system offers two so-called *CPU frequency governors* that reduce the frequency of idling processor cores automatically [8]. The strategy of the *ondemand* governor is to adapt the processor frequency abruptly (i.e. from lowest to highest) while the *conservative* governor changes frequencies step-by-step. However, both base their decisions solely on the load of the processor core. The load is determined by the amount of time that the core is actually performing work (as opposed to running the kernel's idle loop). We argue that this approach is neither precise enough nor sufficient, as the load of a processor core is likely not an adequate measure to determine the general needs for a specific hardware set-up and the optimal configuration in terms of energy efficiency.

In this paper we introduce a more sophisticated governor. The frequency decisions are based on the more informative metric *instructions per memory access*. The governor uses the Linux kernel *perf events* to access hardware performance counters, hence the name *pe-Governor*. To evaluate this governor we use SPEC OMP, a thread parallel benchmark suite for shared memory systems [1]. SPEC OMP consists of eleven different benchmarks which can be executed on different working set sizes. These benchmarks represent real world applications from different scientific fields. Moreover, the benchmarks of the SPEC high performance group are currently under consideration for an extension that targets power efficiency [4].

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

There are three key contributions of this work:

- A Linux CPU frequency governor that can control the frequency of each processor core in a shared memory system based on standard Linux performance events without need of hardware or software modifications and on various processor architectures.
- A metric as basis for the frequency scaling decisions along with a tuned parameter setting for two major x86_64 implementations by AMD and Intel.
- Measurement results of two x86_64 systems using the thread parallel SPEC OMP suite to validate the effectiveness of the governor, the metric and our tuned parameter settings.

The paper is organized as follows: Section 2 highlights our approach for a more sophisticated frequency governor and Section 3 describes our software and hardware test environment. Section 4 lists performance and energy efficiency results for SPEC OMP benchmarks on two different x86_64 systems. Section 5 topics previous work in this field. Finally, Section 6 points out strength and weaknesses of our approach and details ongoing and future work.

2. THE PE-GOVERNOR

The basic idea behind the pe-Governor is twofold:

1. use hardware performance counters for an on-line characterization of the workload that is running on a processor core, and
2. optimize the configuration of the processor core in order to run the workload as power efficient as possible.

One example for this approach would be the detection of an inefficient use of the hardware prefetchers based on performance counters and the subsequent (temporarily) deactivation of the prefetcher. Other usage scenarios are possible. However, we currently focus on the characterization of workloads in terms of memory boundedness. If the tasks on one CPU are memory bound, its frequency (and thereby voltage) can be decreased to a certain level without sacrificing main memory performance. If implemented properly, the performance impact on the application can be minimal while the energy savings are substantial.

The pe-Governor is implemented as a kernel module that can be installed on Linux systems running kernel 2.6.33 or higher. It uses the *perf events* system interface to add performance counters for each logical processor core in order to measure the number or rate of variable hardware performance events. In our case we measure the number of executed instructions and the number of last level cache (llc) misses in a certain time interval. Based on these numbers we determine the rate *instructions per llc-miss* that presumably allows us to estimate how strongly the current CPU load is bound in its performance due to main memory accesses (we refer to this as *memory boundedness*).

Consistent to other loadable modules, the pe-Governor is installed in the running kernel using `insmod`. It can then be activated using the sysfs interface for CPU frequency governors (`/sys/devices/system/cpu/cpufreq/`). Upon activation the module adds the hardware performance counters and establishes a daemon that is activated every 10 milliseconds. At this rate, the daemon retrieves the performance counter data for the last measurement interval, determines the memory-boundedness of the current workload, and adjusts the core frequency accordingly. To avoid a frequently changing frequency, the actual frequency is determined as the highest proposed frequency for the last two measurement intervals.

The governor configuration includes several threshold values that determine when frequency changes should occur. As the optimal configuration is highly system-dependent, the thresholds as well as the target frequencies can be adapted through the sysfs interface. Four `threshold` files can be used to define intervals for different levels of memory boundedness. Using the `frequency` files, each memory boundedness level can be mapped to user-defined processor clock frequencies. In this example configuration

- `threshold_4` = 200
- `threshold_3` = 100
- `frequency_4` = 3000000
- `frequency_3` = 2000000

the system would change a core's clock frequency to 3 GHz when there are at least 200 instructions per llc-miss, to 2 GHz for at least 100 instructions per llc-miss and to the minimal frequency for all other (<100) rates. Another sysfs file allows users to redefine the sampling interval of the governor.

Figure 1 shows an eight second section of an event trace gained from the SPEC OMP benchmark fma3d. We use the VAMPIR performance suite [2] to visualize the event trace. The illustration depicts the timeline of the program execution with the OpenMP threads, the power consumption while running the benchmark, and the frequency of the processor core that runs the master thread. The power consumption is measured using the Dataheap infrastructure [4] while the clock frequency is recorded using the kernel tracepoint events. These two additional event streams are merged into the program trace using the VampirTrace plugin counter interface [9].

3. TEST SETUP

We use two different dual-socket machines with state-of-the-art x86_64 processors as test systems to evaluate the pe-Governor. One system is powered by Intel Westmere processors, while the other machine has AMD Istanbul CPUs. Table 1 provides a detailed lists of the configuration of both systems. The test workload is provided by the thread parallel SPEC OMP benchmark suite in version 3.2, compiled with the Intel compiler suite version 11.1. Each OpenMP thread is pinned to one physical processor core.

The Intel Westmere processors provide the architectural event *Last-Level-Cache-Misses* for each logical processor of the test system. The *unit mask* of this event is 0x41, the *event code* is 0x2E. We disable the TurboBoost technology for our measurement runs. Hyperthreading (HT) is enabled but not used when running the benchmarks, therefore only one thread is running on each physical processor core.

AMD processors do not support a native per-core last level cache-miss event, as the last level cache (L3) is part of the Northbridge. The event code for last level cache-misses is 0x4E0. This event can be counted per core by using the unit

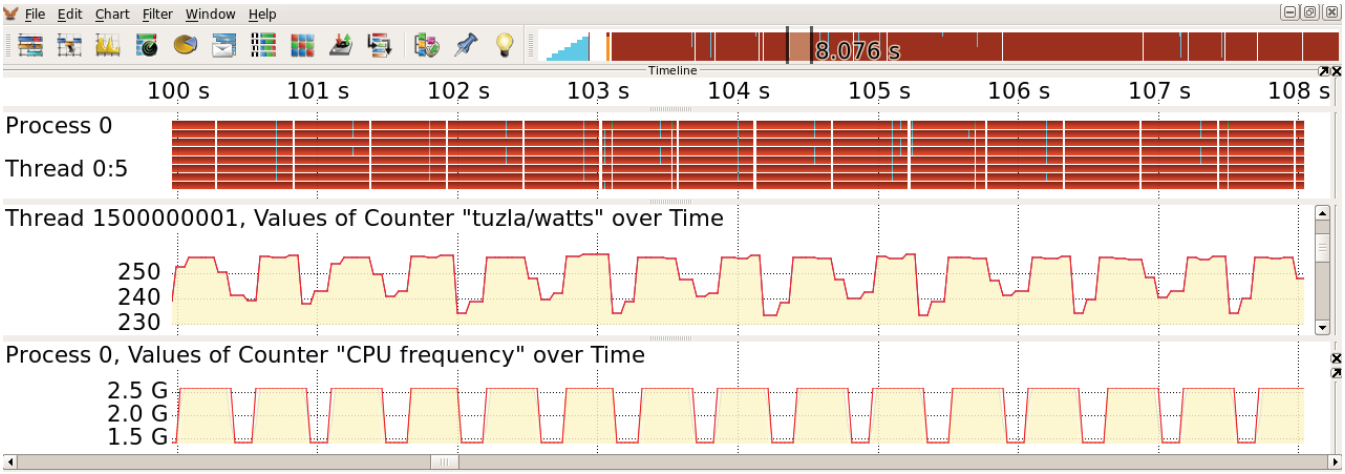

Figure 1: pe-Governor scaling frequency for fma3d benchmark

mask 0x?7, in which the first half-byte encodes the processor core whose events are counted. However, one cannot use the same performance counter register on multiple cores of one processor to measure Northbridge events. Although this is ensured by the Linux perf events infrastructure, the hardware limitation of only four available performance counter registers cannot be circumvented. This limits the number of cores that can measure Northbridge events simultaneously and we therefore only use four of the available six cores per AMD processor for our benchmark runs.

Table 1: Configuration of our test systems

System	Intel Software Evaluation System	Sun Fire X4140
Operating System	Linux 2.6.35.23	Linux 2.6.35.4
Processor	2x Intel Xeon X5670	2x AMD Opteron 2435
Number of used Cores	2x 6 Cores	2x 4 Cores
Core clock	2.93 GHz	2.6 GHz
Uncore/ NB clock	2.66 GHz	2.2 GHz
TDP	95 W	115 W
Interconnect	QPI 4.8 GT/s (19.2 GB/s)	coherent HT3 2.2 GHz (17.6 GB/s)
Codename	Westmere-EP	Istanbul
Technology	32 nm HKMG	45 nm SOI
L1 cache	2x32 KiB per core	2x64 KiB per core
L2 cache	256 KiB per core	512 KiB per core
L3 cache	12 MiB per chip	6 MiB per chip
IMC channels	3x RDDR3	2xRDDR2
Memory type	PC3-10600R	PC2-5300R
Memory size	12 GiB (6x 2 GiB)	16 GiB (8x2 GiB)
Chipset	Intel 5520	nVidia MCP55
Power supply	Ablecom PWS-801P-1R 885W	Delta Energy Systems A221 658W

Table 2 lists our configuration of the pe-Governor. The thresholds for the memory boundedness rate and the corresponding processor clock frequencies have been optimized for the two test systems through a series of benchmark runs. Especially for the AMD system it is likely that further improvements of the settings could be achieved by testing more thoroughly.

Table 2: pe-Governor settings

	Intel Test System	SUN X4140
threshold_4	175	50
frequency_4	2926000	2600000
threshold_3	100	0[1]
frequency_3	2128000	1400000[1]

Figure 2 depicts the design of our power measurement infrastructure. Attached to the test systems are ZES Zimmer LMG95 or LMG450 power analyzers that record the power consumption of the systems at a rate of 10 samples per second. A power measurement daemon on a separate system (the *Collector*) is used to configure the power analyzers and read the measurement data from the devices. The Collector forwards the recorded power values to the so-called *Dataheap* server that stores all measurement data along with their individual timestamps for later analysis. We use a command line client to query the average power consumption for a certain interval from the Dataheap server. This client is used to perform an (automated) post mortem merging of benchmark results and power measurement data. One of the advantages of this approach is that the power measurement itself does not influence the benchmarks in any way [9].

4. TEST RESULTS

We use our test systems to determine the runtime, total energy consumption, and average power consumption for the SPEC OMP benchmarks. The ondemand governor serves as a reference and we quantify the difference between the ondemand results and the pe-Governor results in percent. Our

[1]This setting prevents the system to fall back to 800 MHz, which would reduce the memory bandwidth significantly.

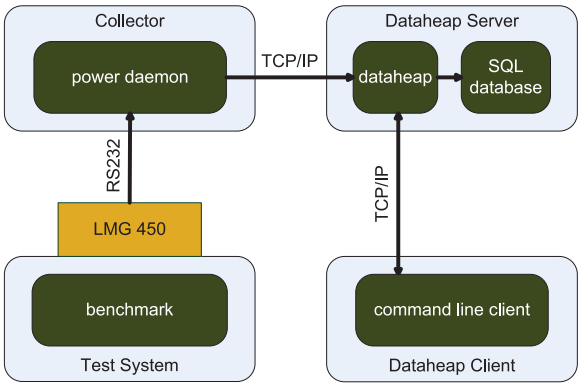

Figure 2: System setup for power measurements

experiments have shown that the results for the ondemand and the performance governor are very similar as both rarely reduce the clock frequency due to the very high computational load generated by the benchmarks.

4.1 Intel Xeon X5670 Test System

Table 3 presents the results of our pe-Governor running on an Intel Westmere system. The energy-saving potential is substantial, as we can achieve a power reduction of up to 11%. However, the runtime for single benchmarks is increased by up to 5.3 % compared to the ondemand governor. This shows that either the frequency decisions of the pe-Governor are inappropriate for some benchmarks or that the induced overhead due to slow p-state transitions or inert frequency adaption is too high. One possible reason for wrong frequency assignments are prefetchers that influence the number of memory references. In average, the pe-Governor increases the runtime by 1.58 % while at the same time providing a 3.88 % and 2.37 % reduction in power and energy consumption, respectively.

Table 3: Changes due to usage of pe-Governor on Intel X5670 Test System

benchmark	runtime (s)	energy (kJ)	power (W)
310.wupwise	0.51 %	-1.58 %	-2.08 %
312.swim	-0.46 %[2]	-11.50 %	-11.09 %
314.mgrid	0.78 %	0.43 %	-0.35 %
316.applu	1.20 %	-5.77 %	-6.90 %
318.galgel	5.11 %	-1.36 %	-6.15 %
320.equake	1.96 %	-3.51 %	-5.37 %
324.apsi	5.33 %	2.80 %	-2.40 %
326.gafort	1.46 %	-4.21 %	-5.58 %
328.fma3d	2.14 %	-0.89 %	-2.97 %
330.art	-0.89 %[2]	-0.87 %	0.02 %[2]
332.ammp	0.21 %	0.43 %	0.22 %
Min	-0.89 %	-11.50 %	-11.09 %
Max	5.33 %	2.80 %	0.22 %
Average	1.58 %	-2.37 %	-3.88 %

4.2 AMD Opteron X4140 Test System

Table 4 presents the results of our pe-Governor running on an AMD Istanbul test system. Compared to the results for the Intel system, the governor does not perform equally well on the Opteron. The maximum runtime increase is 7.12 %, mostly due to a governor configuration that is not yet optimal for the AMD test system. Still, the pe-Governor performs better then the ondemand governor in some cases and also on average. The average energy consumption per benchmark is reduced by 1.36 %. The pe-Governor reduces the energy consumption for eight out of eleven benchmarks. It is interesting to note that the benchmark with the biggest performance degradation is now *applu*, as opposed to *galgel* on the Intel system. Moreover, *applu* shows a 5.77 % reduced energy consumption on the Intel platform compared to a 2.84 % increase on the AMD system. This clearly demonstrates how different the individual characteristics of the two x86_64 microarchitectures are.

Table 4: Changes due to usage of pe-Governor on Sun Fire X4140

benchmark	runtime (s)	energy (kJ)	power (W)
310.wupwise	2.12 %	-0.29 %	-2.36 %
312.swim	1.51 %	-8.87 %	-10.22 %
314.mgrid	0.51 %	0.26 %	-0.25 %
316.applu	7.12 %	2.84 %	-4.00 %
318.galgel	2.94 %	-1.61 %	-4.42 %
320.equake	-0.26 %[2]	-0.33 %	-0.07 %
324.apsi	1.76 %	0.08 %	-1.65 %
326.gafort	1.39 %	-4.64 %	-5.94 %
328.fma3d	1.99 %	-1.89 %	-3.81 %
330.art	0.00 %	-0.15 %	-0.15 %
332.ammp	-0.60 %[2]	-0.41 %	0.19 %[2]
Min	-0.60 %	-8.87 %	-10.22 %
Max	7.12 %	2.84 %	0.19 %
Average	1.68 %	-1.36 %	-2.97 %

5. RELATED WORK

Keller et al. propose a performance event driven CPU frequency governor called VAMOS [5]. Although they do not explicitly name the metric they base the frequency decision on, the paper implies that it is the number of floating point operations per second (FLOPS). FLOPS however is unsuitable as a base for a frequency decision, as integer based workloads do not generate floating point events but can also be memory- or cpu-bound. The authors use *perfmon* as interface to measure FLOPS, which implies the need for a kernel-patch to run the governor. The BLAS routines SMXV and DGEMM are used to verify the result of this work.

Free et al. [3] define two metrics that can predict the energy-time trade-off. They propose that system software designers use them for frequency scaling decisions. These metrics are *misses-per-operation* (MPO) and *slack* (slack is used to predict communication bottlenecks). In our work we use *misses-per-instruction* and thereby implement parts of their suggestions. However, our approach uses a CPU frequency governor to adapt the frequency at runtime instead of being set for a whole application based on its average properties. This has the important consequence, that our approach does not require any workload characterization or user interaction. Instead, only the system administrator is

[2]Runtime decreases and power increases should not occur and can be put down to influences by the operating system.

responsible to setup the CPU frequency governor. However, although being challenging to set up, individual workload characterization is also by design more effective in terms of energy savings.

Long et al. [7] use DVFS to slow down parts of parallel programs to establish a higher compute balance while at the same time saving energy. They determine critical paths of an MPI parallel application during its runtime. In those parallel fractions, which are not part of the critical path, the frequency of the executing processors is reduced to a point at which the program execution is just in time to not slow down blocked communication with other ranks. They additionally use Dynamic Concurrency Throttling (DCT) to alter the number of OpenMP threads in hybrid-parallel applications for memory bound OpenMP regions to further reduce the power consumption.

Kotla et al. [6] propose a scheduling algorithm which measures the instructions per cycle rate as well as the miss rates of the different cache levels. With the given data they determine the reason for low IPC rates and the performance loss for any given frequency. These results are compared to a factor for the acceptable performance loss and frequencies are scaled appropriately. While this approach is comparable to ours, the authors focus on IBM POWER 4 processors, while we propose results for Intel and AMD x86_64 processors. They also used a single-threaded privileged user-level daemon, while we use a multi-threaded system-level daemon that should reduce the overhead significantly. Moreover, we actually implement a CPU frequency governor that can coexist with other tools that rely on hardware performance counters provided by Linux perf events such as PAPI.

6. CONCLUSION AND FUTURE WORK

In this paper we propose a new CPU frequency governor that bases its frequency scaling decisions on performance counter data rather than the load of a specific CPU. It thereby reduces the power consumption of memory-bound tasks. Its major advantages are:

1. there is no need for recompiling, profiling or tracing applications,
2. the governor can be loaded at runtime without disturbing the operating system or workloads,
3. the governor configuration can be optimized for different processor architectures and system characteristics.

Our benchmark results from two different x86_64 test systems show that the average power consumption while running a real-life-like workload can be reduced by up to four percent or even more for very memory intensive applications. Although the average runtime increases slightly, the energy consumption (power consumption times runtime) can be reduced by up to two percent on average. However, some workloads still do not respond well to the pe-Governor and show a runtime increase of up to seven percent. Further experiments and improvements in this area are necessary and planned.

Our future work will cover several areas in which further improvements appear to be achievable. In order to reduce the effort needed to find good governor configurations we plan to implement an auto-tuning algorithm that will determine reasonable threshold values automatically. Moreover, the sysfs interface should be extended in order to allow programmers to have their applications write individual threshold and frequency configurations. In a further step, a compiler-wrapper or library-wrapper may improve the quality of frequency adaption for specific workload phases. We also plan to address the issues that occur when we run the governor on AMD processors with more than four cores. Finally, the governor could be used to adapt more sophisticated hardware features as for example uncore or GPU frequencies, memory prefetchers, or cache configurations. These adaptions are even more system-specific and will certainly require an advanced workload characterization.

7. ACKNOWLEDGEMENT

This work has been funded by the Bundesministerium für Bildung und Forschung via the Spitzencluster CoolSilicon (BMBF 13N10186). We would also like to thank Intel Germany for providing us with the Intel Xeon X5670 evaluation system.

8. REFERENCES

[1] V. Aslot, M. Domeika, R. Eigenmann, G. Gaertner, W. B. Jones, and B. Parady. Specomp: A new benchmark suite for measuring parallel computer performance. In *In Workshop on OpenMP Applications and Tools*, pages 1–10, 2001.

[2] H. Brunst, D. Hackenberg, G. Juckeland, and H. Rohling. Comprehensive Performance Tracking with Vampir 7. In M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, editors, *Tools for High Performance Computing 2009*, pages 17–29. Springer Berlin Heidelberg, 2010.

[3] V. W. Freeh, F. Pan, D. K. Lowenthal, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal. Analyzing the energy-time tradeoff in high-performance computing applications. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):835–848, June 2007.

[4] D. Hackenberg, R. Schöne, D. Molka, M. S. Müller, and A. Knüpfer. Quantifying power consumption variations of hpc systems using spec mpi benchmarks. *Computer Science - Research and Development*, 25:155–163, 2010. 10.1007/s00450-010-0118-0.

[5] V. Keller and R. Gruber. One joule per gflop for blas2 now! In *ICNAAM 2010*, pages 1321–1324, 2010.

[6] R. Kotla, S. Ghiasi, T. Keller, and F. Rawson. Scheduling processor voltage and frequency in server and cluster systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11 - Volume 12*, IPDPS '05, pages 234.2–, Washington, DC, USA, 2005. IEEE Computer Society.

[7] D. Li, D. S. Nikolopoulos, K. Cameron, B. R. de Supinski, and M. Schulz. Hybrid mpi/openmp power-aware computing. In *IPDPS 2010*, 2010.

[8] V. Pallipadi and A. Starikovskiy. The ondemand governor: past, present and future. In *Proceedings of Linux Symposium, vol. 2, pp. 223-238*, 2006.

[9] R. Schoene, R. Tschueter, D. Hackenberg, and T. Ilsche. The VampirTrace Plugin Counter Interface: Introduction and Examples. In *Proceedings of the EuroPar 2010 - Workshops (accepted)*, 2010.