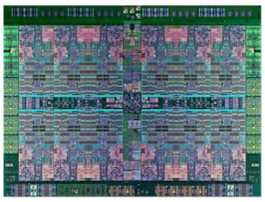
Excel@FIT 2016
http://excel.fit.vutbr.cz

Performance Analysis of IBM POWER8 Processors

Jakub Jelen*



Abstract
As computational strength of nowadays computers increase, the simple measurements using a clock frequency, number of cores or simple GFLOPS does not seem enough anymore. Most of the current computers have Intel processors, based on the `x86_64` architecture, which is well understood, described and proven by many applications. But, there are also other architectures, namely *PowerPC* architecture with the latest Power8 processor and I will investigate whether these two can compete.
In comparison to the previous generation, Power8 was significantly improved at the level of instructions, threads and cores, but also at the the level of the whole system. Various characteristics will be demonstrated on several algorithms and benchmarks in comparison to the recent *Intel Haswell-EP* architecture.
Power8 processor has impressively good memory and caches performance, reaching up to 145 GB/s of sustainable memory bandwidth between processor and the main memory on evaluated system. Also, measurements of simple algorithms exploiting the most of the computational power are comparable or better than the Intel processors, even though I do not have access to the fully equipped Power8 system configuration.
This is not only the first complex comparison of IBM Power8 against current Intel processors, but also a guide to understand the architecture and optimizations for current PowerPC platform. For the next year is the new Power9 is announced, which will build with NVIDIA GPUs again one of the TOP 500 supercomputers.

Keywords: PowerPC — Performance Analysis — Intel

Supplementary Material: Downloadable Code

*xjelen@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

These days, most of the TOP 500 supercomputers of the world are dominated by clusters powered by various Intel processors. They are frequently discussed and described in other publications. But there are curenly 26 clusters using *IBM Blue Gene/Q* (based on Power7 technology) or directly Power7 processors. Since the creation of Power7 processor, IBM has been doing a great job in improving general performance at thread level, core level, but also system level. The evaluation of the new Power8 can bring us some insights into the next year, when IBM probably introduces another fastest supercomputers in the world, Summit and Sierra, based on the *next generation Power9 processors*.

Generally speaking, there are many papers written about performance evaluation of every generation of *Intel Xeon* processors, about scientific applications running on them, about optimization techniques on x86_64 architecture and today also about GPUs and co-processors. There is almost nothing to be found about evaluation of the PowerPC architecture from instruction level to multiprocessor parallelization. Some of the optimization techniques from other architectures are suitable for Power8, but the others require deep understanding of the achitecthure, which is presented in only a few scientific papers.

Most of my information about Power8 architecture itself comes from a few publicly available articles directly from *IBM Journal of Research and Development*, describing Core micro-architecture[1], Cache and memory subsystem[2] and also the first Power8 benchmarking during last year[3], presented on conference in Canada. The first two of them are deeply focused on the architecture, the third is evaluating a specific complex financial algorithm on Power8.

In my work, I will summarize the Power8 architecture, development tools and applicable optimizations on this architecture. The results will be compared against recent Intel Xeon processors and presented on simple algorithms and scientific applications from real world. I will be focusing more on the basics of the architecture, describe in the detail optimization techniques of simple algorithms (provided in attachment), comment on difference in performance results according to compiler, used optimization technique or endianness and will do direct comparison to Intel Xeon processors.

The direct result of this work is a set of optimized algorithms for Power8, well tested and explained on recent Fedora 22, in both little and big endian mode and compared against the algorithm variants on recent Intel processors.

Furthermore it is a handy learning material for current Power8 architecture, for programming and troubleshooting performance gaps in existing applications.

2. Power8 Architectures highlights

The new processor from IBM was announced during the year 2013 and first systems equipped with Power8 processor were available in the middle of 2014. The design of the processor is available under openPOWER license, which means that also other vendors can design and build their own Power8 boards (for example Google did).

The processor chip is created using 22 nm Silicon on Insulator (SOI) process. There are various versions available from 4 to 12 cores per chip with several clock frequencies from 2.5 to 5 GHz. Every single core is able to execute instructions from 8 different software threads in real Simultaneous Multi-Threading (SMT). Every socket can be connected to 8 memory modules using fast memory lanes.

In addition to the local chip interconnect, there are memory lanes for connecting 16 processors together and creating even faster cluster with memory coherency.

The Power8 can operate in both little and big endian mode, which differs a bit in performance of some operations. This supports compatibility between older Power processors using little-endian and IBM z/Architecture systems using big-endian.

2.1 Simultaneous Multi-threading

As already pointed out, the Power8 processor supports instructions execution from up to 8 different software threads on every single core. The mode can be dynamically modified from SMT8 through SMT4 and SMT2 down to ST (Single Thread) where only one thread on a core is scheduled. But even in the SMT8 mode, one running thread can make use of most of the resources, if they would be left unused by the other threads.

The instructions from different threads are going from instruction cache through the decode stage and various issue queues into execution units. There are two 128b wide vector pipelines executing VSX and VMX instructions, two fixed point pipelines, two load/store pipelines, two load pipelines, four double precision FP pipelines, one cryptographic pipeline, one branch execution pipeline, one condition register logical pipeline and one decimal floating point-pipeline.

In every clock cycle, the processor can fetch up to eight instructions, decode and dispatch up to eight instructions, issue and execute up to ten instructions (in the above described 16 execution pipelines), and commit up to eight instructions. Together, there can be performed two load and two load/store operations per cycle, which is double the performance of POWER7 core[1].

2.2 Cache and Memory subsystem

As emphasized in the previous section, every processor core can do up to four memory operations in every cycle. This requires significant improvements all the way throughout the cache to the main memory.

The improvements in speed and throughput were achieved by introducing special memory modules, called *Centaur*. These modules are connected to the processor using 3 B wide (2 B read and 1 B write) lanes operating on frequency 9.6 GHz, providing over 28 GB/s per a module. Every processor can have up to 8 such modules connected, which makes a peak throughput around 230 GB/s per socket. This is a big step since POWER7 providing around 100 GB/s and also against Intel, presenting around 68 GB/s.

The Centaur chips are not only a connection between the processor and memory chips, but they employ also 16 MB L4 cache, which helps to decrease latencies of L3 misses, but also speeds up writes, because the processor does not have to wait for confirmation from the ”slow” memory[2].

2.3 Competitor: Intel

As a direct competitor to the above described Power8 processor, we can consider Intel Haswell-EP server architecture introduced in roughly the same time, based on same lithography size and providing a similar theoretical performance.

Intel Haswell-EP also introduced several improvements since last model, namely the support for Fused Multiply and Add (FMA) in the AVX2 unit, improvements in out-of-order execution, scheduling up to 8 micro-instructions per cycle[4]. Brief comparison of both systems is available in Table 1.

Table 1. Evaluated architectures comparison (per 1 socket). The value in **bold** marks theoretically better parameter.

	IBM Power8 8247-21L	Intel Xeon E5-2680 v3 Haswell-EP
Lithography	22 nm	22 nm
Launch date	Q2’14	Q3’14
Cores	10	**12**
Threads	**80**	24
L1 data cache	**64 kB**	32 kB
L2 cache (core)	**512 kB**	256 kB
L3 cache (sum)	**80 MB**	30 MB
L4 cache	**128 MB**	-
Bandwidth	**192 GB/s**	68 GB/s
Vector unit width	2×128 b (VSX)	256 b (AVX2)
Frequency [GHz]	**3.425**	2.5 Turbo: 3.3
Performance [GFLOPS] (single precision)	**548**	480

3. Benchmarking algorithms and applications

My master thesis describes in detail the difference in performance between various compilers provided by Intel, IBM, or the general-purpose ones (GCC) and also effect of various compiler options on different platforms. There is not enough space for this so I will add only short note about the differences to respective algorihtms.

3.1 STREAM: Memory bandwidth

There are several ways how to measure memory bandwidth, but STREAM[1] has become almost standard over last two decades. The benchmark consists of four operations handling arrays of size bigger than the largest cache level. The first one is a simple *Copy* operation moving values from one place to another, the second is *Scale* which also performs multiplication by constant. They require 1 read and 1 write for every array index. Later on, there is *Add* operation reading two values and storing their sum and *Triad* operation multiplying the first value with constant and adding the result to the third value. These operations require 2 reads and 1 write operation.

In the Table 1, there is a marginal difference between theoretical memory throughput of Power8 system visible, compared to the Intel system. Both of the testing machines have two sockets. But the real measurement is not fair, because the Power8 system I have access to, has only half of the memory modules occupied, and Intel cluster has every node with two sockets and full memory lanes.

In the Figure 1, we can observe very steep increase up to 8 threads for both architectures. The Intel does

[1]https://www.cs.virginia.edu/stream/

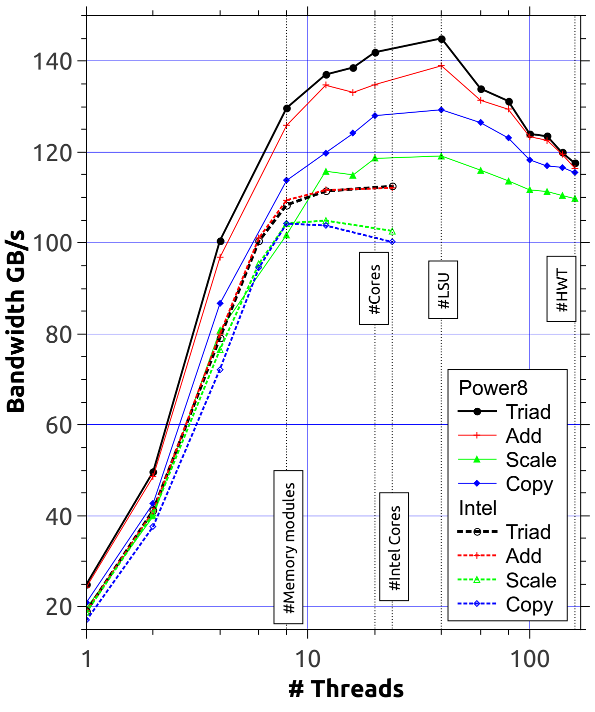


Figure 1. STREAM: Sustainable memory bandwidth for different amount of running threads on (half-filled) Power8 and Intel Haswell-EP architecture

not scale much over 8 threads, because the 4 threads per socket are able to saturate all 4 memory lanes with directly attached memory modules. But the peak in the Intel curve is with all employed cores reaching up to 122 GB/s.

The Power8 makes the maximum use of all the memory traffic, 144 GB/s, with 40 threads (2 thread per core), which corresponds to the amount of load/store units available. Using more than 2 threads per core does not bring any more improvements on this system because in SMT4 and SMT8 mode, data prefetch is highly limited and there are not enough load/store units to provide enough data in every cycle.

The algorithm is auto-vectorized by compiler. The only possible optimization is an explicit data prefetch. It is visible that Intel compiler is using this technique because GCC on Intel provides a bit worse results (graph not shown).

Running these tests also requires pinning the software threads to the hardware threads to prevent their migration or placing all of them on a single physical core. This was achieved using an OpenMP environment variables. The measurements were tested with 500 GB of data per array, which eliminates any impact of caches (except automatic hardware prefetches). The results are comparable to the results published in [3] and [5], but their results are from the fully equipped system. They report 320 GB/s and 300 GB/s respectively.

3.2 Matrix and vector multiplication

The second minimalistic benchmark is compute bound and it makes use of vector units available in Intel and Power8. As visible from the Table 1, Intel AVX2 is using wider vector registers, which makes it technically two times faster. But Power8 running in ST mode can use two VSX units, basically in the same time, which makes both cores "equal" in the specifications. But what about the real performance?

The benchmark is limited and optimized to a single thread to evaluate the performance of single core. The algorithm is simple enough and makes an opportunity to use of Fused Multiply-and-Add (FMA) operation available both in Power8 and Haswell-EP architecture. We can evaluate both the processor performance and the compiler ability to generate optimal code from a native algorithm, using OpenMP SIMD `pragmas` or a hand-vectorized variant. There is also a respective operation in BLAS library for comparison, provided by ATLAS and Intel Math Kernel Library (MKL). The measured results for matrix 1000×1000, repeated 10 000 times can be found in a Table 2. The hand-vectorized code on Power is reaching the best results,

Table 2. Matrix and Vector multiplication: 10 000 multiplications of a matrix 1024×1024. The table shows results of different algorithms, different optimization techniques on both architectures using different compilers. The BLAS library (Intel MKL and ATLAS) was evaluated for comparison, but is always faster than any other solution on both platforms.

	IBM Power8 (be)		Intel Haswell-EP	
Compiler:	GCC	AT	GCC	ICC
Standard order				
Native	18.38 s	18.15 s	10.63 s	1.42 s
Hand-vectorized	**1.24 s**	**1.24 s**	1.58 s	1.57 s
omp SIMD	7.49 s	7.45 s	1.67 s	1.77 s
cBLAS	1.15 s	1.08 s	1.42 s	1.42 s
Transposed matrix				
Native	1.90 s	1.78 s	1.55 s	**1.38 s**
Hand-vectorized	1.90 s	1.80 s	-	-
omp SIMD	1.94 s	1.99 s	1.58 s	1.77 s
cBLAS	1.12 s	1.11 s	1.29 s	1.34 s

but generally all the other different techniques on Intel results in better time.

The compiler names refer to GNU Complier Collection (GCC), IBM Advance Toolchain (AT)[2] and commercial Intel C++ Compiler (ICC). I also ran all test cases on Power8 machine in little endian mode and it showed slightly worse performance in all cases (not shown in the table).

3.3 N-body

N-body is a simulation of gravitation forces among particles in space (in our case 3D). The computation of all forces in one moment has a complexity of $O(n^2)$ and requires more different operations than just an FMA in the previous task. As the task is getting bigger, there is a good opportunity to evaluate parallelization on Power8, performance for more threads, data sharing and synchronization.

In this algorithm, we can again observe a difference between Intel and GCC compilers. The first one is able to successfully vectorize this complicated algorithm and data flows, but the other requires hand-vectorization to deliver some reasonable results (but still the automatically tuned vectorized version is faster). Also I was unable to run this algorithm on the Power8 machine with two sockets, because of some performance problems in the little endian mode. The results are obviously better for Intel even though intel executes

[2] http://ibm.co/AdvanceToolchain

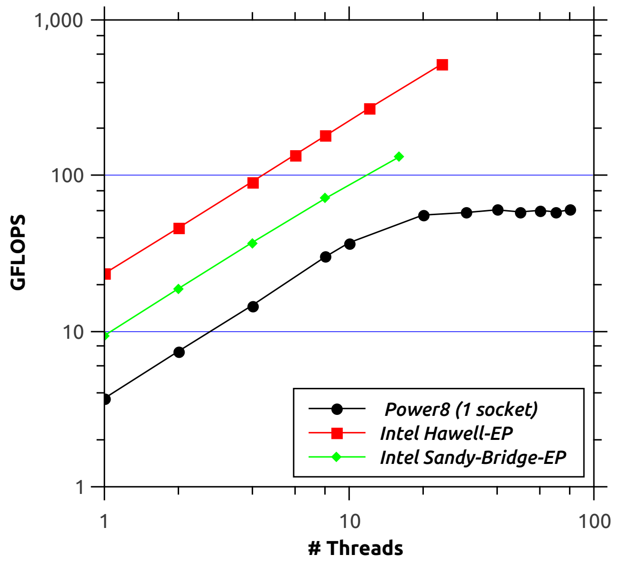


Figure 2. N-body: Evaluation of 10 000 particles over 1 000 steps in discrete time. Single-thread performance of Haswell architecture is roughly twice as fast as Sandy Bridge and it is roughly twice as fast as Power8 processor. However, single socket Power8 with 20 threads is reaching almost the same speed as single socket of Intel Sandy Bridge.

significantly more floating point operations. Also for the comparison, I added to the graph previous generation of Intel processors, Sandy Bridge, which has comparable performance in this case.

3.4 Steady state heat distribution

This algorithm requires less computing and more data from memory than the N-body problem and, therefore, it should suit more the Power8 processor. It is again an iterative simulation. In every iteration, we adjust temperature at every single grip point of the matrix based on its neighbors until the sum of changes is small enough. This algorithm is computed in double precision so we can expect significantly worse results on both platforms.

Figure 3 shows the performance on both machines for various matrix sizes. This algorithm scales on Power8 processor (1 socket) very well for all shown matrix sizes up to 10 threads and them fills the gaps (except for the smallest matrices struggling to divide between large amount of threads). The largest matrix (256 MB) is bigger than the L3 and L4 caches, but even though single socket Power8 processor provides better performance than two sockets with Haswell-EP.

The Intel Haswell-EP scales in similar manner very steady. We can also observe, that the matrix of size 2048 does not fit into the L3 cache anymore and there is significant drop of performance for 24 threads. It is even more visible on the larges matrix.

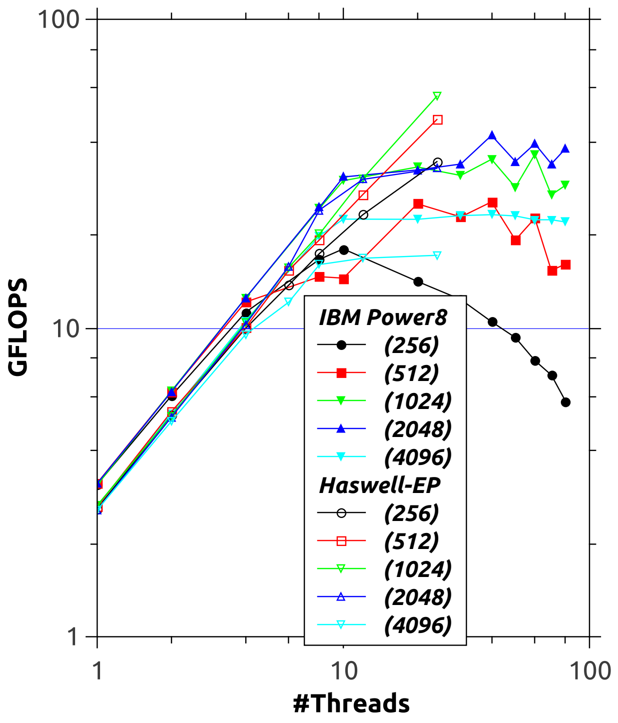


Figure 3. Steady state heat distribution: Scaling over different amount of threads for different matrix sizes (256 means matrix 256 $\times$ 256 which makes around 1 MB of memory). Intel maximum is at 56 GFLOPS for matrix of size 1024 $\times$ 1024. Single socket of Power8 can make over 40 GFLOPS for the matrix 2048 $\times$ 2048.

4. Conclusions

This paper describes an evaluation of Power8 architecture, providing massive simultaneous multi-threading and enormous memory and caches performance ready for today's cloud computing workload. It brings also insight into the next generation Power9 system aiming for the first place in TOP500 supercomputers in the world.

The measurements confirmed a very good memory performance reaching up to 144 GB/s, which beats the Intel concurrence in every aspect. The computation power is a bit disqualified by using a non-commercial compiler, but the scaling is steady in all tested cases. For N-body, we reached around 60 GFLOPS (single precision) on single socket and over 40 GFLOPS for Steady state heat distribution (double precision). Basically, tasks requiring more memory to operate are more suitable for this architecture at this time.

This is complex evaluation of the whole Power8 architecture in direct comparison to current Intel Haswell-EP processors. It is also evaluation of open-source development tools (GCC) and their suitability for HPC algorithms.

Another people interested in this topic can make use of attached algorithms or guides about programming and optimizing applications for Power8 architecture. It would be also interesting to re-evaluate the future Power9 architecture or investigate scalability of different algorithms in this environment.

Acknowledgements

I would like to thank my supervisor *Jiří Jaroš* for his guidance and valuable help and comments during the whole work on my master thesis.

I also thank Red Hat for providing me with an access to IBM Power8 servers, especially to *Dan Horák*.

The last but not least thanks go to the *IT4I*, the Czech National Supercomputing Center for providing me with an access to Intel clusters *Anselm* and *Salomon*.

References

[1] B. Sinharoy, J.A. Van Norstrand, R.J. Eickemeyer, H.Q. Le, J. Leenstra, D.Q. Nguyen, B. Konigsburg, K. Ward, M.D. Brown, J.E. Moreira, D. Levitan, S. Tung, D. Hrusecky, J.W. Bishop, M. Gschwind, M. Boersma, M. Kroener, M. Kaltenbach, T. Karkhanis, and K.M. Fernsler. Ibm power8 processor core microarchitecture. *IBM Journal of Research and Development*, 59(1):2:1–2:21, Jan 2015.

[2] W.J. Starke, J. Stuecheli, D.M. Daly, J.S. Dodson, F. Auernhammer, P.M. Sagmeister, G.L. Guthrie, C.F. Marino, M. Siegel, and B. Blaner. The cache and memory subsystems of the ibm power8 processor. *IBM Journal of Research and Development*, 59(1):3:1–3:13, Jan 2015.

[3] I. Z. Reguly, Abdoul-Kader Keita, and M. B. Giles. Benchmarking the ibm power8 processor. In *CASCON 2015 November 2-4, 2015, Toronto, Canada*, November 2015.

[4] Daniel Molka, Daniel Hackenberg, Robert Schöne, and Wolfgang E Nagel. Cache coherence protocol and memory performance of the intel haswell-ep architecture. 2015.

[5] A. V. Adinetz, P. F. Baumeister, H. Böttiger, T. Hater, T. Maurer, D. Pleiter, W. Schenck, and S. F. Schifano. Performance evaluation of scientific applications on power8. In *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation*, pages 24–45. Springer, 2014.