The Journal of Supercomputing (2019) 75:6178–6193
https://doi.org/10.1007/s11227-019-02844-3

How good is the OpenPOWER architecture for high-performance CPU-oriented weather forecasting applications?

R. Moreno[1] · **E. Arias**[2] · **A. Navarro**[1] · **F. J. Tapiador**[1]

Published online: 4 April 2019

Abstract
Performance, i.e., execution times, is one of the most important features of HPC software, but energy consumption is also growing in importance if we intend to extend application to Exascale. This is the case of HPC software used in weather forecasting, in which every ounce of performance is critical in order to increase the accuracy and precision of its results. In this work, we study the performance-energy balance of an OpenPOWER processor, which is designed for the high workloads typically seen on data servers and HPC environments. Our results show that the OpenPOWER processor is superior in performance in weather forecast workloads compared to other processors commonly used in HPC, but at the expense of consuming more energy. Furthermore, the highest hyperthreading modes available on OpenPOWER processors do not perform well with HPC workloads and are even detrimental to performance.

Keywords OpenPOWER · Performance · Energy consumption · Compilers · Weather research and forecasting

1 Introduction

Processors evolved over recent years, following different development paths related to different goals. Many processor architectures have focused on providing computational capacity to the widely known big data [19], a paradigm in which peak performance is not as important as the data throughput needed. These targets do not always fit well with the capabilities of a general-purpose processor and need

✉ R. Moreno
raul.moreno@uclm.es

[1] University of Castilla-La Mancha, Toledo, Spain

[2] University of Castilla-La Mancha, Albacete, Spain

the better-fitted capabilities offered by accelerators such as GPUs, driving enhancements in processor architectures as host processors for these accelerators.

High-performance computing (HPC) is another of the previously mentioned paths, which has not attracted as much attention as its big data counterpart. HPC seeks peak performance for software that is not always well-fitted to an accelerator and needs all the computing capacity that the processor can provide. Weather forecasting software is an example of these high demanding types of HPC software, in which every ounce of performance is crucial to increase the resolution and accuracy of its results.

Furthermore, HPC is a strategic resource for Europe in order to improve enterprise competitiveness and science, dealing with complex computational problems[1] such as: accelerating genome sequencing by two orders of magnitude, enabling scientists to defeat cancer diseases, develop new drugs or study the human brain.[2]

The European HPC strategy plans to reach Exascale by 2022, which means having supercomputers capable of executing one trillion (10^{18}) operations per second. However, entering the Exascale era has a price [2], this mainly being energy consumption. Exascale platforms will consist of thousands of processors, meaning that a slight reduction in energy consumption on each of them would substantially reduce the energy consumption of the whole platform. Therefore, the need to reduce energy consumption is even more important than computing performance, not only from an economic point of view, but also to preserve our planet. More than ever, we need to squeeze every drop of performance per Watt consumed in order to advance in HPC-related fields. For this reason, HPC-focused processor architectures are evolving to increase their performance without increasing their energy footprint, rather even reducing it.

The IBM OpenPOWER[3] architecture is partially designed for HPC workloads, and its performance is indeed good, but at the expense of relatively high energy consumption compared to other architectures. In the present work, we summarize the problems that arose when we used the POWER8 architecture, described by Sinharoy et al. [28], to execute our reference weather forecasting software. Summarizing, we:

- Studied the balance between performance and energy consumption achieved by a POWER8 processor and its different hyperthreading (SMT) modes.
- Analyzed how good this balance is compared to another Intel-based processor.
- Two different compilers for the POWER8 are also compared.
- Identified ways in which the OpenPOWER architecture could be improved to achieve better performance-energy balance in order to reach the Exascale on the weather forecasting field.

In Sect. 2, we summarize some of the works that have used an OpenPOWER architecture as their target platform for solving big data and HPC problems. In

[1] https://ec.europa.eu/digital-single-market/en/policies/high-performance-computing.
[2] https://ec.europa.eu/digital-single-market/en/news/high-performance-computing-best-use-examples.
[3] https://openpowerfoundation.org/.

Sect. 3, we present the POWER8 architecture and the weather forecast simulations we performed on it. The performance/energy balance when using two different compilers is studied in this section, including the power saving method implemented in the POWER8 architecture. Moreover, we offer some thoughts about how the simulations would improve when using the POWER9 architecture. Finally, in Sect. 5, we provide some insights into the results of this work.

2 Precedents and related work

In this section, we first review part of the literature on the great suitability of OpenPOWER architectures for accelerators and for Big Data. However, there are not so many works that study how good the OpenPOWER architecture is for non-accelerated HPC, i.e., multiprocessor platforms or supercomputers. In the last part of this section, we review some works that studied the behavior of hyperthreading in HPC environments with different processor architectures. However, none of these works studies whether the hyperthreading capabilities of the OpenPOWER architecture could be beneficial in an HPC environment. Besides, we could not find any reference that characterized the performance-energy balance of the OpenPOWER architectures.

2.1 OpenPOWER benefits when using accelerators

Köhler and Saak [18] studied how different clock frequencies in a POWER8 influence the Gauss–Jordan elimination scheme when using General Purpose GPU (GPGPU) programming. The vast majority of the work was performed on the GPUs attached to the system, and the POWER8 processor role was to compute the first part of the algorithm, whose resulting matrices were feed to the GPUs on the second part. The algorithm is iterative, so they overlapped the second part of the algorithm (performed in the GPUs) in parallel with the first part of the next iteration (performed in the CPU). The authors observed that the CPU finished earlier than the GPUs and had to wait for the GPUs to finish, and thus they lowered the POWER8 clock frequency. The result was lower power consumption (14.2%) without increasing the overall wall time of the whole algorithm.

Wei et al. [32] studied portability issues when using accelerators in HPC programs. They proposed the use of OpenACC to increase the portability of their case study software based on a Particle-In-Cell (PIC) algorithm. They ported their PIC algorithm to OpenACC and executed it successfully on GPUs, ×86 and OpenPOWER processors. This work shows there are no inherent architectural problems when porting software from an x86 architecture to an OpenPOWER architecture, even when using a GPGPU configuration. Some problems may arise if the target software is hard-coded to a specific architecture, i.e., using *tailored-to-architecture* compiler intrinsics.

In Zenker et al. [33], the OpenPOWER memory architecture, along with the improved CPU-GPU communications, is used to complement the memory

limitations of the GPUs, in a GPGPU environment. The authors focused on the portability of a PIC algorithm when using multiple CPU architectures and NVIDIA GPUs, finally creating an abstraction of the CUDA programming language. Algorithm performance was not improved although portability was enhanced.

2.2 OpenPOWER benefits in big data processing

Lu et al. [21] illustrated how the OpenPOWER memory architecture can improve the performance of the latest generation of Remote Direct Memory Access (RDMA) software over Infiniband networks. In their study, they achieved a 2.73× performance improvement when using Hadoop's remote call procedures with RDMA over OpenPOWER processors. These results show how future big data processing can benefit from OpenPOWER.

2.3 Other OpenPOWER works

Adinetz et al. [1] explored the POWER8 architecture features over three different scientific programs. Their methodology is based on the performance hardware counters implemented in the OpenPOWER processor. In summary, their conclusion is that the POWER8 is a processor that is designed to be the CPU of a GPU-accelerated system, tackling the weaknesses present in a GPU such as integer performance, out-of-order execution and memory bandwidth. Additionally, they were unable to efficiently use the higher SMT modes of POWER8 CPU with the tested scientific software. The results showed that the SMT2 mode is the ideal case, the SMT4 is equal to the SMT2 in terms of performance, and the SMT8 is always inferior to the others.

Few works exist in the literature that seek an efficient usage of the high SMT modes provided by the OpenPOWER architecture. Sudheer and Srinivasan [30] highlight the importance of synchronization barriers in POWER8 when there is a large amount of thread parallelism, i.e., higher SMT modes. In their study the MPI, POSIX and OpenMP barriers are compared. They leveraged the memory hierarchy present in the OpenPOWER architecture and proposed a k-ary tree based barrier, greatly improving synchronization latencies.

Feliu et al. [9] proposed a novel scheduling method based on process symbiosis. They pack complementary processes from the point of view of processor core elements in such a way that the interference between them is minimal. This scheduling is useful for multiple process placement such as that used with Cloud containers (e.g., Docker), but is at application level, so cannot be used to improve the efficient placement of the threads of the same process, e.g., an OpenMP program.

2.4 High SMT modes in HPC

Problems with high SMT modes when using HPC applications are not only related to OpenPOWER architectures, which usually use low SMT modes for achieving better performance on highly loaded cores [13, 17]. Leng et al. [20] studied the effects

of Intel Hyper-Threading technology on different HPC applications. They demonstrated that computationally intensive applications may not benefit from SMT because the resources of each CPU core are already being used to the maximum by a single thread.

Kaliszan et al. [16] tested several HPC applications with different computational requirements (time, memory and I/O) on different state-of-the-art HPC architectures (Intel, AMD, ARM and POWER8). Their results showed that the best times are obtained when the number of processes coincides with the number of physical cores, i.e., without SMT. In fact, the use of SMT would decrease the performance of these applications. WRF was one of the applications tested, but due to problems with compilation, they could not test it on the ARM and POWER8 systems. Therefore, we do not know how it could behave with the high SMT modes of POWER8.

3 POWER8: OpenPOWER for HPC weather forecasting

The POWER8 is a Reduced Instruction Set Computer (RISC) processor commercialized by IBM, with 22 nm integration technology. It can achieve a frequency of 4.5 Ghz, having up to 12 processing cores with 8MB of L3 cache memory each. It also features four multithreading modes with 1 thread (SMT1), 2 threads (SMT2), 4 threads (SMT4) and 8 threads (SMT8) per core. IBM concentrated their efforts on the branch prediction and out-of-order execution features of this processor.

Most of the works summarized in the previous section focused on using the OpenPOWER architecture as a CPU host for a GPGPU (or accelerator-based) system. This is because its memory architecture is well suited to hosting accelerators and their energy management features, along with the improved communication network between the accelerator and the processor. Therefore, and based on the literature, we can state that the OpenPOWER architecture is highly appropriate for a GPGPU environment.

Nonetheless, the 8-way hyperthreading capabilities in previous works remain mostly ignored. At first sight, hyperthreading might be a desirable feature to increase the number of processing units and therefore increase computing performance in a non-GPU-supported HPC applications. It might also be interesting to know how the different hyperthreading levels affect the energy consumption of the processor.

Energy consumption is also a current concern. For a processing unit, it is not only important to achieve the best performance when processing data, but also to do so with the minimal energy footprint, especially in the case of an HPC system which could have thousands of these processing units. There are some ways to measure how energy efficient a system is, e.g., using the inverse (J^{-1}) of the energy consumption [25]. Also, it can be possible to compare the relative energy consumption between two different systems [12]. However, these metrics do not tell us about how the performance-energy balance of an individual system is. Other works compare performance efficiency and energy efficiency separately and then evaluate the balance by hand (using graphical methods) [3, 4], but these works do not provide a simple metric to measure that balance.

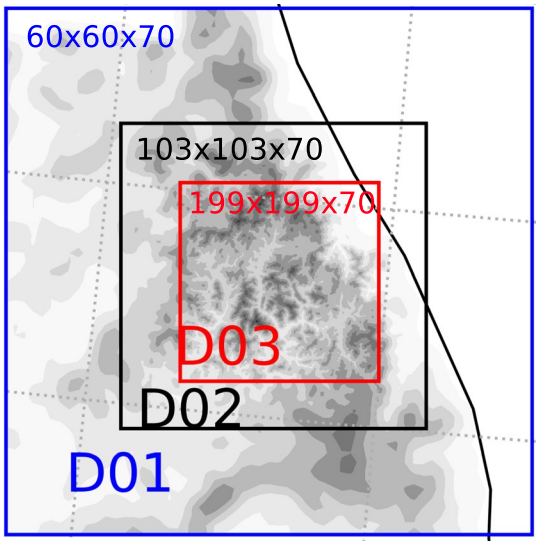

Fig. 1 Graphic representation of WRF domains used in this study. The D01, D02 and D03 are, respectively, the low-, middle- and high-resolution domains

A simple metric for evaluating the performance-energy balance of a system is the Energy-Delay-Product (EDP) concept proposed by Freeh et al. [11] and defined as:

$$\mathrm{EDP} = J \cdot t \tag{1}$$

in which J is the energy consumed in Joules and t the total time of simulation. The lower the value of the EDP, the better is the balance between the execution time and energy consumption. For the sake of clarity, all the EDPs collected in this work are represented divided by 10^9.

3.1 Collected data

Our case study is based on the WRF [29] software, a commonly used HPC software for weather forecasting and atmospheric research. WRF is an example of an HPC application that is CPU-bounded, so high-resolution simulations run on a supercomputer [8, 31]. For parallel performance, WRF implements a hybrid MPI-OpenMP implementation for the calculations. Many distributed computing adaptations of WRF exist in the literature, including an adaptation to Grid computing by Davidovic et al. [6], or another adaptation by Fernandez et al. [10] to hybrid Cluster-Grid-Cloud distributed systems. GPU support could also be added to some WRF routines to achieve greater parallelism [7, 27], which could benefit from the GPGPU capabilities of OpenPOWER architectures.

We used a high-resolution configuration for our simulations, using three nested grids with dimensions ($199 \times 199 \times 70$), ($103 \times 103 \times 70$) and ($60 \times 60 \times 70$), and cell resolutions of 300, 900 and 2700 meters, respectively. A graphic representation of the domains is shown in Fig. 1.

Nesting in WRF [5, 15, 29] allows resolution to be focused over a region of interest, by using additional high-resolution domains (child domains) inside other

Table 1 High-performance compilation options used in our simulations for every compiler

System	Compiler	Compiler options
Intel	Intel compilers 18	`-O3 -ipo -xHost -qopenmp -fpp -auto`
POWER8	IBM XL Community edition 16.1	`-O5 -q64 -qhot -qaltivec -qsmp=omp`
POWER8	IBM Advanced toolchain 11 (GNU v7.3.1)	`-Ofast -flto -mtune=power8 -mcpu=power8 -mpowerpc64 -maltivec -mvsx -ftree-vectorize -funroll-loops -fvect-cost-model -fopenmp`

low-resolution domains (parents) that maintain the boundary conditions for their children domains. This approach yields similar results in the region of interest than the alternative, i.e., simulating the whole boundary region with a single high-resolution domain, without greatly increasing computational requirements. Therefore, it is common in WRF to nest high-resolution domains, such as the `D03` domain, inside other lower resolution domains, e.g., `D02` and `D01` domains.

Also, note the difference in the size of the problem between the high-resolution domain `D03` with approximately 3 million cells and the low-resolution domain `D01` with approximately 250k cells. The complexity of the physics used in the simulation combined with the size of the problem of the high-resolution domain is directly proportional to the amount of processing power needed for the simulation. The size of the problem not only demands computing speed, but also determines the amount of memory required to host the simulation.

This study utilized the WRF-ARW version 3.9.1, using P3 microphysics [23], the Rapid Radiative Transfer Model [22] scheme and the Noah Land Surface Model [24]. WRF simulations were performed on a POWER8 platform (8335-GCA model, 4.5 GHz), having one processor with 10 cores and another processor with 9 cores on the same motherboard. The operating system was Ubuntu Linux and the platform mounted 128GB of RAM. On this platform, two versions of WRF were compiled, one with the IBM XL 16.1 Community Edition compilers and the other with the IBM Advanced Toolchain 11 (AT), the latter based on GNU v7.3.1 compilers. Additionally, simulations for every SMT mode of the POWER8 processor were performed in order to profile the behavior of the system with different hyperthreading levels. For best performance, we used the compilation options summarized in Table 1. Note that best performance does not mean better forecast results, so we attempted to use the best compilation options to obtain performance without sacrificing too much precision.

As a reference point, we also executed the same simulations on an Intel-based platform with two 8-core processors (Xeon E5-2650, 2.2 GHz) on the same motherboard, a widely used setup in HPC systems. To be specific, this system is a computing node of a private supercomputer that we isolated to perform the simulations for our study. On this platform, we used the Intel Compilers (v18) to compile the software, Red Hat Enterprise Linux 6 and 64GB of RAM. A WRF simulation with

Table 2 Time and energy consumption values for POWER8 and Intel platforms with different compilers and hyperthreading (SMT) modes

System	Cores	Thrs	t (s)	Avg. (W)	Energy (MJ)	Sp	EDP ($/10^9$)
Intel (IC)	16 (8 + 8)	16	3043	235	0.70	1.00	2.13
SMT1 (XL)	19 (10 + 9)	19	1669	814	1.36	1.82	2.27
SMT2 (XL)	19 (10 + 9)	38	1624	916	1.49	1.87	2.42
SMT4 (XL)	19 (10 + 9)	76	1952	764	1.49	1.56	2.91
SMT8 (XL)	19 (10 + 9)	152	2683	800	2.15	1.13	5.77
SMT1 (AT)	19 (10 + 9)	19	1887	745	1.40	1.61	2.64
SMT2 (AT)	19 (10 + 9)	38	1644	799	1.31	1.85	2.15
SMT4 (AT)	19 (10 + 9)	76	1648	723	1.19	1.85	1.96
SMT8 (AT)	19 (10 + 9)	152	1876	743	1.39	1.62	2.61

Speedup respect to the Intel case (Intel time divided by POWER8 time) and the EDP are shown too

the selected configuration uses about 4GB of RAM so, even though the available memory of the Intel-based platform is half that of the POWER8 platform, it was enough for the simulations and no disk swapping occurred.

In order to avoid outliers, the computing times and energy consumption were obtained from the average of ten executions in each case. Table 2 shows all the data obtained from the executions, including wall time, energy and power consumption, and EDP.

3.2 Energy-performance balance

Regardless of the compiler, if we compare the Intel platform with the best cases of the POWER8, we can see that the EDP is very similar, meaning that both platforms are equally efficient in "squeezing" performance from the energy they consume. From the perspective of energy-performance balance, EDP for the POWER8 processors with AT compilers was very similar to the Intel processors. The SMT4 mode, when used with the AT compilers, was the only case in which the EDP of the POWER8 was slightly lower than that of the Intel. In this case, the execution time was similar to the other cases, but the energy consumed was significantly lower, reducing the EDP to 1.96.

The POWER8 processor is well suited to handle petitions from 2 threads at the same time. This is supported by the slight increase in performance and energy consumption of the SMT2 mode with respect to the SMT1, meaning that the processor is being used more extensively by two threads. In addition, an interesting phenomenon emerges when using the SMT4 mode. In this mode, the energy footprint was the lowest of all the SMT modes, and without loss of performance in the case of using the AT compilers.

Note the increased energy consumption when using the XL compilers, meaning that these compilers were using more processor components and functionality than the AT compiler. However, as we can see, more did not mean better, as the AT compiler produced simpler code that significantly reduced power consumption with a

Table 3 Time and energy consumption values for POWER8 (AT compilers) and Intel platforms with different hyperthreading (SMT) modes

System	Cores	Thrs	t (s)	Avg. (W)	Energy (MJ)	Sp	EDP ($/10^9$)
Intel	16 (8 + 8)	16	3043	235	0.70	1.00	2.13
SMT1 (PS-AT)	19 (10 + 9)	19	2765	420	1.16	1.10	3.21
SMT2 (PS-AT)	19 (10 + 9)	38	2304	434	1.00	1.32	2.30
SMT4 (PS-AT)	19 (10 + 9)	76	2229	445	0.99	1.37	2.21
SMT8 (PS-AT)	19 (10 + 9)	152	2585	453	1.17	1.18	3.02

similar performance to the XL compilers. Therefore, increasing the SMT mode with the XL compilers meant increasing energy consumption without a clear gain in the performance for SMT2, and a substantial decrease in performance for higher SMT modes. Moreover, the AT compilers exhibited a lower energy footprint which, combined with their overall performance, reduced their EDP values and made them a better option than the XL cases.

All in all, and despite their energy consumption, the best times were achieved when using the XL compilers with SMT2.

3.3 Power saving mode

Compared to the Intel processors, the POWER8 processors used more energy for the same simulation, but, in exchange, they yielded higher computing performance. The OpenPOWER architecture features power saving and management modes in which their unused processors and cores are put to sleep to save energy when not in use. The clock frequency of the processor can be easily set from the operating system. The simulation times previously shown were obtained with the maximum frequency available in the cores used, while those not used were adjusted in energy saving mode. This is equivalent to setting the processor power management policy to `on demand` mode.

In order to compare the Intel and IBM processors, we set up simulations in the POWER8 processors with a lower power consumption and performance. The AT compiler was chosen for these tests due to its reduced energy consumption in the previously conducted tests. The objective was to equal simulation times and compare their energy consumption. We lowered the power consumption of the POWER8 processor by reducing its frequency to the minimum (2.5 GHz), i.e., setting all the processor cores to `power saving` (PS) mode, regardless of their usage. This action had the effect of reducing the simulation energy consumption at the expense of increasing the computing times, obtaining the results in Table 3.

Even in their lowest frequency, the POWER8 achieved better performance (1.37X) than its Intel counterpart, which was running at a comparable clock frequency. The power consumption was also reduced by a 1.19 × factor with respect to its analogous full frequency case, even though the Intel processor consumption was once again lower.

Table 4 Performance stats obtained from the POWER8 SMT modes (19 cores) and the Intel platform. The values corresponding to branching and cache loads were divided by 10^9

	CPU%	CPI	GBranch	/	GMisses (%)	L3 GLoads	/	GMisses(%)
Intel	0.98	1.22	9989	/	044 (0.45%)	1948	/	855 (44%)
SMT1	0.99	1.02	11144	/	197 (1.77%)	145	/	021 (15%)
SMT2	0.88	1.45	16671	/	220 (1.32%)	176	/	020 (12%)
SMT4	0.82	2.33	27578	/	239 (0.87%)	304	/	062 (21%)
SMT8	0.83	4.35	44944	/	280 (0.62%)	482	/	140 (29%)

The POWER8 has more cores per processor than the Intel, so we compared the execution time divided by the number of cores. The POWER8 obtained 117 s per core, while the Intel processor took 190 s per core. Again, the POWER8 cores achieved higher performance per core even at the same clock frequency.

From the perspective of energy-performance balance, the EDP with the power saving mode was slightly higher for the best cases (SMT2 and SMT4) than the EDP from Table 2. This meant a significant reduction in the power consumption with a proportional decrease in performance.

3.3.1 CPU and cache efficiency

In the previous sections, we looked at the achievable performance and power consumption of POWER8 architecture compared to an Intel one. The results showed much better performance of the POWER8 architecture. We examined processor performance statistics in order to determine the reason for this better performance.

Both processor architectures used in this work had hardware counters at their disposal for profiling processor events. We used the Linux utility `perf` for handling these counters and measured some important metrics for performance such as the CPU usage, cycles, branch prediction misses and last level cache (L3) misses. The performance metrics obtained are summarized in Table 4.

A look at the Intel architecture metrics shows that the CPU usage was near 100%, so the processor was busy all the time. Moreover, the Cycles Per Instruction (CPI), which gives an estimation of how costly it is to execute an instruction in this simulation, was very efficient. The problematic metric with this architecture was the high usage of the L3 cache coupled with nearly half the petitions being a miss. The amount of L3 cache available in each of the Intel processors was 20 MB (2.5 MB per core), which represents a great disadvantage compared to the POWER8 processor (8 MB per core). Despite this high cache-missing rate, the CPI was relatively low, which means that the Intel architecture exhibited a low penalty on accessing main memory.

In the case of the POWER8 architecture, the CPU usage depended on the SMT mode. As expected, the SMT1 and SMT2 modes kept the CPU busy most of the time, while in the SMT4 and SMT8 modes the CPU was used at approximately 80% of capacity. The SMT1 mode achieved a similar CPI to the Intel processor (2.2 GHz), but doubled the processor clock frequency (4.5 GHz), doubling

the performance. The POWER8 had 8MB of L3 cache for every core, resulting in 80MB in the processor with 10 cores and 72MB in the 9-core processor. Compared to the Intel processor's cache, the POWER8 had four times more cache memory, including the first two cache levels. This high amount of cache was reflected in the significantly fewer accesses to L3 cache and even fewer misses, 15% compared to the Intel's 44%. Similar metrics were obtained with the SMT2 mode.

When using the SMT4 mode, the CPI rose significantly, making the instructions more costly in terms of processor cycles. This was due to the increase in the overall branch and cache misses, especially the latter. The POWER8 has an advanced branch predictor, but the percentage of branch prediction misses is still much higher than Intel. Even though all the metrics were worse compared to the SMT1 mode, 4 threads are processing in parallel, which ultimately makes the execution as fast as the SMT2 mode (see Table 2). In fact, the reduced CPU usage of the SMT4 mode while achieving the same simulation time was causing this reduction in power consumption. Furthermore, the decrease in performance per thread observed in the SMT4 mode was magnified in the SMT8 mode. In this mode, the metrics were significantly lower, doubling the cache misses with respect to the SMT4 mode. In this case, the number of threads could not compensate for the performance degradation produced by the misses in the cache. From these results, we can see that the POWER8 performance was very susceptible to cache misses, i.e., accessing main memory, unlike its Intel counterpart which maintained a low CPI.

From these results, we can state that the OpenPOWER memory architecture performed well under SMT1 and SMT2 modes, but was not sufficient for higher SMT modes. Many HPC programs (such as WRF) make use of nearly 100% of the thread computing time, while the SMT4 and SMT8 modes in OpenPOWER are currently designed for light threads with variable workload, i.e., throughput focused.

We propose two design options to improve the OpenPOWER architecture and achieve higher HPC efficiency:

- Increasing the duplicity of the intra-core hardware elements to access the memory hierarchy, reducing cache missing and its associated penalty.
- Reducing the core complexity by removing the SMT4 and SMT8 modes and removing other features related to accelerators. Simpler cores would allow an increase in the number of cores with their expanded cache and a reduction in the energy consumption.

The latter option is partially achieved in certain variants of the current generation of OpenPOWER architecture: the POWER9, although it is still focused on being a host CPU for GPUs in all the variants.

3.4 POWER9 expectations

The OpenPOWER POWER9 processor is the successor of the POWER8, offering superior specifications and four different processor architectures depending on the intended usage. A comparison between the POWER8 and the different POWER9

Table 5 POWER8 (P8) and POWER9 (P9) specifications

	Max sockets	Max cores	Max SMT	L3 (MB)	Mem buffer	Mem bandwidth (GB/s)
P8	2	10	SMT8	96	L4	230
P9 SU PowerVM	16	12	SMT8	120	L4	230
P9 SO PowerVM	2	12	SMT8	120	Direct	120
P9 SU Linux	16	24	SMT4	120	L4	230
P9 SO Linux	2	24	SMT4	120	Direct	120

variants is presented in Table 5. The *Scale-Up* (SU) processors are designed to be installed with up to 15 other processors in the same node, and due to their higher memory bandwidth requirements, they provide more memory bandwidth than their *Scale-Out* counterparts. The *Scale-Out* (SO) processors are designed for dual-socket nodes, with both processors working together seamlessly. In this case, the access to memory is direct with a reduced memory bandwidth. The *PowerVM* variants are designed to work with virtual machines, where they need very high concurrency which is supported by the `SMT8` mode. Moreover, there are also the *Linux* variants with more hardware cores than their *PowerVM* counterparts, but with reduced `SMT` modes.

The POWER9 processor is even more focused toward being the host processor of an accelerator-based system due to its high performing communications based on *NVLink 2.0*, the OpenCAPI protocol and the available PCIe Gen4 ports. These specifications make the POWER9 an excellent processor for managing multiple NVIDIA GPUs.

Considering our results, we expect that the best performing POWER9 variant in our WRF simulations would be the *Scale-Up Linux* because of its high number of hardware cores (24) and their near-optimal `SMT4` mode. This option could present memory bottlenecks when using 16 processors on the same node due to the reduced amount of L3 cache per core (5 MB), so the *Scale-Out Linux* (10 MB of L3 per core) could also be an alternative if these bottlenecks occur. Ultimately, it is difficult to know *a priori* which of both configurations would perform better.

4 Discussion

The energy consumption of the POWER8 is significantly higher compared to the reference processor, even when reducing the clock frequency to the minimum possible. This consumption is higher when using the IBM XL compilers than when using other compilers, due to a more intensive use of the hardware elements in the processor. This extra complexity is unneeded in the HPC software used, leading to an increase in power consumption without an increase in performance. For these reasons, this processor is not recommended for environments where energy consumption is critical.

In terms of performance, the POWER8 is significantly faster compared to the reference processor, even when reducing the clock frequency to the minimum to save energy. Therefore, this processor is highly recommended for HPC environments where energy consumption is secondary.

The POWER8 includes high-performance memory hierarchy, which is excellent for hosting an accelerator (GPGPU). This is one of the main advantages of the POWER8 compared to other processors such as the reference processor used in this study, but it could still be better. The SMT4 and SMT8 modes are devised for light workload threads, so the memory hierarchy becomes a bottleneck when using all the threads intensively. Moreover, a miss in the L3 last-level cache entails a higher penalty in performance than with other processor architectures.

Another question is how these two architectures might behave when more than one node is used. The main factor affecting the MPI scalability in HPC architectures, when more computing nodes are used, is the communication overhead [14, 26]. Two main factors affect the injection of data into the network:

- The number of nodes.
- The data communications generated by these nodes.

Having more nodes means that MPI processes must communicate data to more targets, especially when using MPI collective operations [14]. Also, allocating multiple MPI processes per node could be detrimental for performance, as more communication is generated per node [26].

The POWER8 node had better simulation times than the Intel node, and it would take approximately 2 Intel nodes to match that performance. Therefore, if we increase the number of POWER8 nodes used, we would need to at least double the number of Intel nodes to match the same performance.

On the other hand, when the number of nodes is too high for the network to manage, the scalability limit is reached, where a further increase in nodes is detrimental to performance. Therefore, Intel nodes are expected to reach their limit before POWER8 nodes, or in other words, POWER8 nodes could get more performance at their scalability limit.

5 Conclusions

A performance-energy study based on weather forecasting software executed over POWER8 processors is presented in this work. Several simulations of the HPC weather forecasting WRF software were performed for the characterization of the processor with this kind of CPU intensive load.

In terms of energy-performance balance, the EDP is very similar to other processors used in HPC, although the POWER8 obtains significantly higher performance at the cost of more energy consumption.

As a conclusion, the POWER8 SMT2 and SMT4 modes are the best-fitted hyperthreading modes for HPC workloads because of their superior performance and proportional energy consumption, the latter being slightly better with SMT4. However,

the SMT1 mode leaves the processor resources underused, while the SMT8 mode collapses the memory hierarchy when using threads at full throttle. In order to improve performance for non-accelerator-based HPC workloads, we suggest an increase in the duplicity of the intra-core hardware to improve parallel access to the memory hierarchy, and a reduction in the complexity of the hardware cores to allow processors with a larger number of cores.

As future work, the scalability of the OpenPOWER architecture, from the point of view of the EDP, should be studied with an increasing number of nodes. It is difficult to predict beforehand what behavior will be observed with a high number of nodes. On the other hand, the current generation of OpenPOWER processors, the POWER9 architecture, is very promising for improving the performance of weather forecasts due to the improved memory hierarchy and the increased number of hardware cores.

In the future, we want to obtain the EDP for the POWER9 architecture and study whether the performance-energy tradeoff is better or worse than POWER8. We expect the energy consumption per core of the POWER9 architecture to be lower or at least equal to that of its predecessor. In addition, some POWER9 models are optimized for HPC and in fact only implement up to SMT4 hyperthreading, so it would be interesting to test the performance obtained by the SMT4 mode in these processor models.

Acknowledgements Funding from projects CGL2013-48367-P and CGL2016-80609-R (Spanish Ministry of Economy and Competitiveness, Science and Innovation) is gratefully acknowledged. RM acknowledges an FPI grant EEBB-I-17-12253. AN acknowledges Grant FPU13/02798

References

1. Adinetz AV, Baumeister PF, Böttiger H, Hater T, Maurer T, Pleiter D, Schenck W, Schifano SF (2014) Performance evaluation of scientific applications on POWER8. In: International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, Springer, pp 24–45
2. Bergman K, Borkar S, Campbell D, Carlson W, Dally W, Denneau M, Franzon P, Harrod W, Hill K, Hiller J et al. (2008) Exascale computing study: technology challenges in achieving exascale systems. Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Technical Report, p 15
3. Bermejo B, Juiz C, Guerrero C (2018) On the linearity of performance and energy at virtual machine consolidation: the cis2 index for cpu workload in server saturation. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp 928–933
4. Bermejo B, Juiz C, Guerrero C (2018) Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance. J Supercomput 75:1–29
5. Daniels MH, Lundquist KA, Mirocha JD, Wiersema DJ, Chow FK (2016) A new vertical grid nesting capability in the weather research and forecasting (wrf) model. Mon Weather Rev 144(10):3725–3747
6. Davidović Davor, Skala Karolj, Belušić Danijel, Prtenjak Maja Telišman (2010) Grid implementation of the weather research and forecasting model. Earth Sci Inf 3(4):199–208
7. Denham Mónica, Lamperti Enrico, Areta Javier (2018) Weather radar data processing on graphic cards. J Supercomput 74(2):868–885

8. Farguell A, Cortés A, Margalef T, Miró JR, Mercader J (2018) Scalability of a multi-physics system for forest fire spread prediction in multi-core platforms. J Supercomput. https://doi.org/10.1007/s11227-018-2330-9
9. Feliu Josue, Eyerman Stijn, Sahuquillo Julio, Petit Salvador, Eeckhout Lieven (2017) Improving IBM POWER8 performance through symbiotic job scheduling. IEEE Trans Parallel Distrib Syst 28(10):2838–2851
10. Fernández-Quiruelas V, Blanco C, Cofiño Antonio S, Fernández J (2015) Large-scale climate simulations harnessing clusters, grid and cloud infrastructures. Future Gener Comput Syst 51:36–44
11. Freeh Vincent W, Lowenthal David K, Feng Pan, Nandini Kappiah, Rob Springer, Rountree Barry L, Femal Mark E (2007) Analyzing the energy-time trade-off in high-performance computing applications. IEEE Trans Parallel DistribSyst 18(6):835–848
12. Goel B, Titos-Gil R, Negi R, McKee SA, Stenstrom P (2014) Performance and energy analysis of the restricted transactional memory implementation on haswell. In: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, pp 615–624
13. Jeffers James, Reinders James, Sodani Avinash (2016) Intel Xeon phi processor high performance programming Knights landing edition. Morgan Kaufmann, Burlington
14. Jin Haoqiang, Jespersen Dennis, Mehrotra Piyush, Biswas Rupak, Huang Lei, Chapman Barbara (2011) High performance computing using MPI and OpenMP on multi-core parallel systems. Parallel Comput 37(9):562–575
15. Jones Robert W (1977) A nested grid for a three-dimensional model of a tropical cyclone. J Atmos Sci 34(10):1528–1553
16. Kaliszan D, Fürst S, Gienger M, Gogolenko S, Meyer N, Petruczynik S (2019) Comparative benchmarking of HPC systems for GSS applications: GSS applications in the HPC ecosystem. In: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, ACM, pp 43–52
17. Kim R, Choi J, Lee M (2019) Optimizing parallel GEMM routines using auto-tuning with intel AVX-512. In: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, ACM, pp 101–110
18. Köhler M, Saak J (2018) Frequency scaling and energy efficiency regarding the Gauss–Jordan elimination scheme with application to the matrix-sign-function on OpenPOWER 8. Concur Comput Pract Exp 31:e4504
19. Alexandros Labrinidis, Jagadish Hosagrahar V (2012) Challenges and opportunities with big data. Proc VLDB Endow 5(12):2032–2033
20. Leng T, Ali R, Hsieh J, Mashayekhi V, Rooholamini R (2002) An empirical study of hyper-threading in high performance computing clusters. In: Proceedings of LCI International Conference on Linux Clusters: Linux HPC revolution, 45
21. Lu Xiaoyi, Shi Haiyang, Shankar Dipti, Panda Dhabaleswar K DK (2017) Performance characterization and acceleration of big data workloads on OpenPOWER system. In: 2017 IEEE International Conference on Big Data, pp 213–222
22. Mlawer Eli J, Taubman Steven J, Brown Patrick D, Iacono Michael J, Clough Shepard A (1997) Radiative transfer for inhomogeneous atmospheres: RRTM, a validated correlated-k model for the longwave. J Geophys Res Atmos 102(D14):16663–16682
23. Hugh Morrison, Milbrandt Jason A (2015) Parameterization of cloud microphysics based on the prediction of bulk ice particle properties. part i: scheme description and idealized tests. J Atmos Sci 72(1):287–311
24. Niu GY, Yang ZL, Mitchell KE, Chen F, Ek MB, Barlage M, Kumar A, Manning K, Niyogi D, Rosero E et al (2011) The community noah land surface model with multiparameterization options (Noah-MP): 1 model description and evaluation with local-scale measurements. J Geophys Res Atmos. https://doi.org/10.1029/2010JD015139
25. Park Jinsu, Baek Woongki (2019) Analyzing and optimizing the performance and energy efficiency of transactional scientific applications on large-scale NUMA systems with HTM support. J Parallel Distrib Comput 127:1–17
26. Shainer G, Liu T, Lui P, Graham R (2011) Accelerating high performance computing applications through mpi offloading. HPC Advisory Council–HPC Scale Special Interest Group, Sunnyvale, CA
27. Pablo Silva Juan, José Hagopian, Marcel Burdiat, Ernesto Dufrechou, Martín Pedemonte, Alejandro Gutiérrez, Gabriel Cazes, Pablo Ezzatti (2014) Another step to the full GPU implementation of the weather research and forecasting model. J Supercomput 70(2):746–755

28. Balaram Sinharoy, Van Norstrand JA, Eickemeyer Richard J, Le Hung Q, Jens Leenstra, Nguyen Dung Q, Konigsburg B, Ward K, Brown MD, Moreira José E et al (2015) IBM POWER8 processor core microarchitecture. IBM J Res Dev 59(1):1–2
29. Skamarock WC, Klemp JB, Dudhia J, Gill DO, Barker DM, Wang W, Powers JG (2005) A description of the advanced research wrf version 2. Technical report, National Center For Atmospheric Research Boulder Co Mesoscale and Microscale Meteorology Div
30. Sudheer CD, Srinivasan A (2015) Efficient barrier implementation on the POWER8 processor. In: 2015 IEEE 22nd International Conference on High Performance Computing (HiPC), pp 165–173
31. Wang Yuzhu, Jiang Jinrong, Zhang Junqiang, He Juanxiong, Zhang He, Chi Xuebin, Yue Tianxiang (2018) An efficient parallel algorithm for the coupling of global climate models and regional climate models on a large-scale multi-core cluster. J Supercomput 74(8):3999–4018
32. Wei Y, Wang Y, Cai L, Tang W, Wang B, Ethier S, See S, Lin J (2016) Performance and portability studies with Open ACC accelerated version of GTC-P. In: 2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp 13–18
33. Zenker E, Widera R, Huebl A, Juckeland G, Knüpfer A, Nagel WE, Bussmann M (2016) Performance-portable many-core plasma simulations: porting PIConGPU to OpenPOWER and beyond. In: International Conference on High Performance Computing, Springer, pp 293–301