An Empirical Evaluation of HPC Processors for a Coupled Scientific Simulation

Jelle van Dijk[1][0000-0003-3005-9890], Okba Hamitou[2], Gabor Zavodszky[1][0000-0003-0150-0229], Ana-Lucia Varbanescu[1,3][0000-0002-4932-1900], Andy D. Pimentel[1][0000-0002-2043-4469], and Alfons Hoekstra[1][0000-0002-3955-2449]

[1] Informatics Institute, FNWI, University of Amsterdam, The Netherlands
[2] Eviden, Bezons, France
[3] EEMCS Faculty, University of Twente, The Netherlands

Abstract. Scientific simulations grow increasingly complex, and require massive compute and energy resources. To address these requirements, new HPC processors emerge regularly. Quantifying the performance and energy-efficiency impact of new HPC processors is non-trivial, yet critical for next generation (super)computers development. In this work, we demonstrate how comprehensive benchmarking contributes to quantifying the performance and energy efficiency impact of five different processors on HemoCell, a complex coupled scientific simulation. We constructed representative benchmarks and ran hundreds of benchmarks to provide both coarse-grained data, focusing on metrics such as utilization and speed-up, and fine-grained data, based on hardware performance counters. Our detailed data analysis provides insights into the impact of higher memory bandwidth, the need for higher cache capacity, the relevance of high core-density, and the effect of low-power designs. Our results show that cache capacity is the *highest-impact* feature for both the performance increase and energy consumption reduction of HemoCell.

Keywords: Energy Efficiency · Performance · Coupled Scientific Simulation · Benchmarking · Performance Analysis

1 Introduction

Scientific computing applications are increasingly complex. In turn, they require substantial computational power and energy; these developments lead to increasing concerns regarding supercomputing's energy bill [9]. To address this, new processors deliver higher peak compute throughput (FLOP/s), memory bandwidth (GB/s), and energy efficiency (FLOPs/Watt). Processors achieve these improvements with features such as higher core counts, larger caches [8], and power optimizations, all of which *should* help improve the performance and efficiency of high-performance computing (HPC).

In this work, we show that improvement is not guaranteed, as we investigate the impact of different HPC processors on a coupled scientific simulation. Specifically

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

we show the methodology and results of a benchmarking campaign to characterize the performance and energy efficiency impact of five HPC processors on a coupled scientific simulation. We analyze five processors spanning high-bandwidth memory (*Intel SPR ± HBM*), high core density (*AMD Bergamo*), low-power ARM design (*Ampere Altra*), and two AMD Zen architectures (*Rome, Genoa*).

Our representative scientific application is HemoCell [12], a coupled large-scale scientific framework that simulates cell-resolved blood flow. In prior work, performance predictive and descriptive models have been built for Hemocell. [1, 11] However, none of these models are suitable for the evaluation of new processors, as these require calibration and are not designed for cross-processor evaluation.

Thus, we propose the first benchmarking strategy that enables performance and energy impact assessment of new processors on HemoCell. To understand the processors' impact on the performance and energy efficiency of HemoCell, we propose a three-step strategy: (1) peak-performance benchmarking, (2) application benchmarking, (3) fine-grained application benchmarking. Combining this data shows how these processors have different impact on HemoCell, and highlights the large cache per core of *AMD Genoa* as the highest-impact feature.

2 Methodology

Benchmarking The goal of benchmarking is to collect data to characterize the impact of the processor for the application of interest. In this work, we propose two types of benchmarks: (1) hardware-specific, to validate the processors' peak performance, and (2) application-specific, to determine the ability of the application to utilize this peak. Peak performance is the highest achievable FLOP/s and memory bandwidth of a hardware configuration. For the application-specific data we run a set of single-node HemoCell benchmarks. Here we collect high-level performance metrics and detailed performance data in the form of hardware events, such as the number of cycles, instructions, and cache misses.

Performance analysis We propose two levels of performance analysis: (a) a high-level performance comparison, and (b) a fine-grained performance-pattern identification. With high-level analysis we compare the main performance metrics, e.g., runtime and energy consumption. These metrics enable objective performance comparisons between workloads and/or systems. However they are insufficient to explain the observed performance because of their lack of detail. For fine-grained analysis, we use the hardware event data, which improves our understanding of the performance. Hardware performance counters keep track of metrics related to code execution and hardware behavior, e.g., cache hits/misses, number of cycles, and number of retired instructions.

3 Experimental Setup

This section describes the application, hardware, and tools we used for data collection and analysis. In Table 1 we describe all processors used in this study.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

3.1 Application under test: Hemocell

HemoCell couples separately modeled blood plasma and red-blood cells (RBCs) [12]. The domain is decomposed into N equally sized subdomains (one MPI process per CPU core). During the simulation neighboring processes exchange the outer fluid layer and RBCs crossing subdomain boundaries. Domain size is expressed in lattice units (LU).[4] Each benchmarks is a cubic domain, with 18% of the volume occupied by RBCs. We benchmark two domain sizes, S1 ($200 \times 200 \times 100$ LU, 20,000 iterations) and S2 ($200 \times 200 \times 200$ LU, 10,000 iterations), chosen to provide equivalent computational workloads across processors , while respecting the 25×25×25 LU minimum subdomain constraint, imposed by HemoCell, for the highest-core-count processor (AMD Bergamo, 256 cores).

Table 1: The processors under study, characterized by Lithography (L), Thermal design power (TDP), Frequency, Number of cores, Sockets per Node (S), Cache sizes (L1 and L2 per core, L3 shared), Channels per socket (C), memory technology (DIMMs), throughput, and size.

Processor	TDP [W]	Base/Max Freq. [GHz]	Cores (Sockets)	L1/L2/L3 Cache size	C	DRAM	Size [GB]
AMD Epyc Bergamo 9754	400	2.25/3.1	128 (2)	32 kB / 1 MB 256 MB	12	DDR5 4800 MT/s	384
AMD Genoa 9654	360	2.4/3.55	96 (2)	32 kB / 1 MB 384 MB	12	DDR5 4800 MT/s	384
AMD Rome 7H12	280	2.65/3.3	64 (2)	32 kB / 512 kB 256 MB	8	DDR4 3200 MT/s	256
Ampere Altra Q8030	210	3.0	80 (1)	64 kB / 1 MB 32 MB	8	DDR4 3200 MT/s	256
Intel Sapphire Rapids 9480	350	1.9/3.5	56 (2)	48 kB / 2 MB 112.5 MB	8	DDR5 4800 MT/s	512
Intel Sapphire Rapids HBM 9480 (Cache mode)	350	1.9/3.5	56 (2)	48 kB / 2 MB 112.5 MB	8	HBM2e 3200 MT/s	128

3.2 Compilers and monitoring tools

For all experiments, we use the gcc compiler (versions between 11.3 and 13.2 for all AMD and Intel CPUs, and version 8.5.0 for the Ampere CPU), and OpenMPI 4.1.5. Hemocell is compiled with *-O3*,*-march=native*, and *-std=c++11* flags. For each runtime or energy data point, we run 3 repetitions and report the average. For hardware events measurements, two repetitions are performed.

Runtime and Performance Counters Execution time is captured by internal HemoCell timers. For detailed measurements we use Score-P [5], an automatic code instrumentation and profiling tool. Hardware counters are read using `perf`.

Peak Performance For the *AMD Bergamo*, *Intel SPR*, and *Ampere Altra* processors we use Stream [7] for memory bandwidth and HPL, an implementation of the LINPACK [3], for FLOP/s. On the other processors we collect the peak performance results through the empirical roofline model toolkit (ERT) [6].

Energy Consumption For the *AMD Bergamo*, *Intel SPR*, and *Ampere Altra* processors, we measure the power consumption of the node at a frequency

[4] 1 LU = 0.5 µm

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

of 0.7Hz, using processor-specific tools. For the *AMD Rome* and *AMD Genoa* processors, we measure the total energy consumption using EAR [2].

4 Evaluation

The benchmarking results are analyzed in the following five sections [5].

4.1 Peak performance

Figure 1 shows the peak performance for all processors. The results reveal the significant influence of the *AMD Bergamo* and *Intel SPR* processor features. Firstly, the high number of cores on the *AMD Bergamo* processors provides peak FLOP/s. Secondly, the *HBM* has a significant impact on the peak memory bandwidth. Enabling *HBM* increases the peak memory bandwidth to 1400 GB/s.

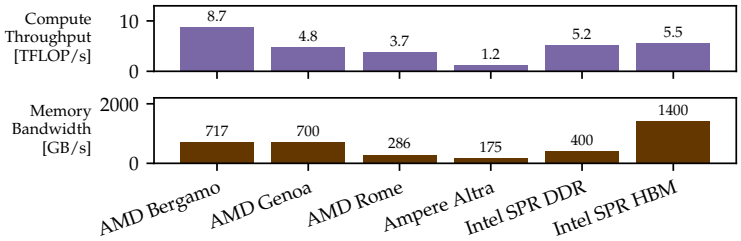

Fig. 1: Performance results for main memory bandwidth and double-precision FLOP/s.

4.2 Runtime and Energy

In Figure 2, we show time, energy, and power results for the S1 benchmark. We observe the best performance on the *AMD Genoa* processor, with both the shortest time per iteration, 14 ms, and the highest energy efficiency, 11 J/iteration. Even though it does not have the highest peak performance or bandwidth. This confirms that the performance of a complex application is hard to predict based only on peak performance measurements.

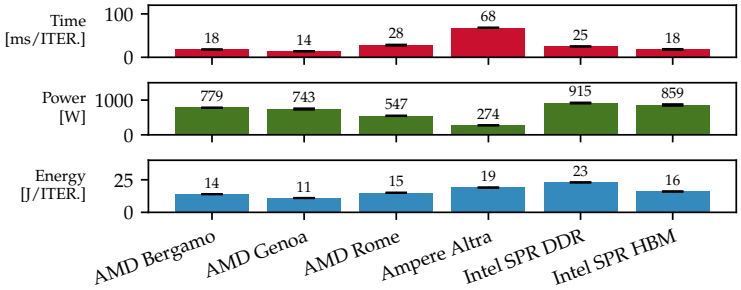

Fig. 2: Time, power, and energy per iteration for S1 (mean and Std. dev. from 3 runs).

Figure 2 confirms the advertised low power consumption of the *Ampere Altra*. However, because of the low peak performance the energy efficiency is worse than all other processors except for the *Intel SPR DDR*.

[5] Benchmarks, results and plot generation scripts are available at, `https://doi.org/10.5281/zenodo.15003101` and `https://doi.org/10.5281/zenodo.18701790`

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

On the *Intel SPR*, enabling *HBM* leads to a speedup of around 1.4, and improves the energy efficiency. The effect of *HBM* indicates a memory bottleneck. However, the improvement is not as high as the peak memory bandwidth improvement, which is 3.5 times higher than *DDR*. This suggests that HemoCell is unable to utilize all the available memory bandwidth.

In Figure 3 we show time and energy consumption per iteration for a varying number of MPI processes. On a two socket node, using half the cores only utilizes a single socket. For all but one of the processors, we observe a reduction in runtime and energy consumption with a decrease in MPI processes. However, on the *Ampere Altra* processor, there is no performance loss if we do not utilize all cores. Due to this and a higher power consumption when using more cores, increasing the number of MPI processes above 32 decreases energy efficiency. This result indicates a memory bottleneck, because adding more compute power does not reduce the runtime - we investigate this further in Section 4.4.

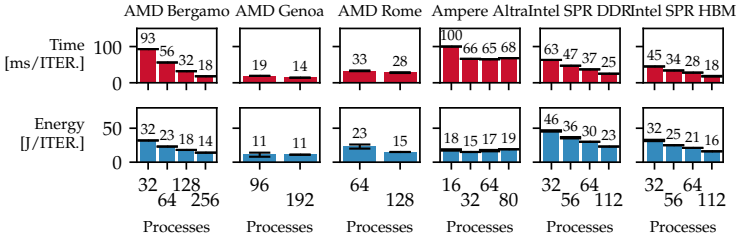

Fig. 3: Time and energy per iteration for S1, for a varying number of processes.

4.3 Cycles and Instructions

Table 2 presents the number of instructions, cycles, cycles per instruction (CPI), and data requests per instruction (MOPI), as indicators of how the processors handle the instruction mix of the application. Strikingly, there is a large deviation in the instruction count, both between processors and when changing the number of processes. The difference in the instruction count likely stems from the differences in ISAs, compilers, and even different levels of auto vectorization.

Table 2: Number of instructions (# Inst.), cycles (# Cycles), cycles and memory operations per instruction per processor (CPI and MOPI), when running S1 with a varying number of processes (NP) per processor. CPI = (Cycles/Instructions) × processes, MOPI = L1 accesses/Instructions.

Processor	NP	# Inst. $\times 10^{12}$	# Cycles $\times 10^{12}$	CPI	MOPI	NP	# Inst. $\times 10^{12}$	# Cycles $\times 10^{12}$	CPI	MOPI
AMD Bergamo	128	278	1.56	0.72	0.51	256	358	0.86	0.61	0.52
AMD Genoa	96	188	0.78	0.40	0.51	192	309	0.67	0.42	0.51
AMD Rome 7H12	64	211	2.27	0.69	0.50	128	303	1.34	0.57	0.51
Ampere Altra	64	199	3.15	1.01	0.37	80	178	3.25	1.46	0.37
Intel SPR DDR	56	192	1.85	0.54	[a]0.35	112	204	0.95	0.52	[a]0.35
Intel SPR HBM	56	174	1.31	0.42	[a]0.36	112	185	0.69	0.42	[a]0.35

[a] For the *Intel SPR* processor we can only measure the L1 loads.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

Both the *AMD Genoa* and *Intel SPR* with *HBM* processors have a CPI of 0.42. However, their performance and energy efficiency differ significantly. The *Ampere Altra* shows a noticeably high CPI of 1.46. But, due to the completely different instruction set architecture (ISA), this comparison does not necessarily translate into a corresponding performance drop.

On all *AMD* processors, we observe that the number of L1 requests per executed instruction is between 0.50 and 0.52. On the *Ampere Altra* processors this ratio is lower, at 0.35. This higher ratio can be explained by difference in ISA: the *ARM*-based *Ampere Altra* favors simple instructions and uses a load-store architecture, resulting in fewer L1 accesses per instruction.

4.4 Cache and Memory Bandwidth Utilization

The observed L1 miss-rate across all processes is between 1.1% and 2.4%. In Table 3 we show last-level-cache (LLC) requests and misses. We note that, due to restricted access, we only have this data for the *Intel SPR* and *Ampere Altra* processors. Table 3 reveals a clear difference in the number of LLC requests between the two processors. Specifically, there are approximately 8× and 4× more LLC misses on *Ampere Altra* than on *Intel SPR*. We expect the larger L2 cache of the *Intel SPR* to cause this difference. Furthermore, this difference translates to memory bandwidth utilization, which we show in Table 4.

Table 3: Total requests (req.), request misses and miss rate (MR) for Last-level cache (LLC) per processor, for the S1 benchmark with a varying numbers of processes (NP).

Platform	NP	# LLC req. $\times 10^{12}$	# LLC misses $\times 10^{12}$	MR	NP	# LLC req. $\times 10^{12}$	# LLC misses $\times 10^{12}$	MR
Ampere Altra	64	4.12	2.07	0.50	80	4.03	2.03	0.50
Intel SPR DDR	56	0.45	0.41	0.91	112	0.47	0.41	0.87
Intel SPR HBM	56	0.49	0.43	0.88	112	0.53	0.45	0.84

Table 4: Memory bandwidth utilization for the S1 and S2 benchmarks (B), for varying numbers of processes (NP) and two different processors. Data = L3 misses × 64 B, Bandwidth = Data/Runtime.

Platform	NP	B	Time [s]	Data [GB]	Bandwidth [GB/s]	NP	B	Time [s]	Data [GB]	Bandwidth [GB/s]
Ampere Altra	64	S1	1315	132421	102	64	S2	1180	116921	100
	80	S1	1358	130136	96	80	S2	1392	135757	98
Intel SPR DDR	56	S1	962	26276	28	56	S2	983	30752	32
	112	S1	505	26229	53	112	S2	527	30943	60
Intel SPR HBM	56	S1	684	27779	41	56	S2	689	31915	47
	112	S1	365	28775	79	112	S2	365	32960	92

On the *Intel SPR*, we notice an increase in memory bandwidth utilization with the increase in either the number of processes or the problem size. This is explained by cache availability. Increasing the number of processes reduces the cache per process, while an increase in problem size increases the demands on the memory bandwidth. Even this higher demand on memory bandwidth is only

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

6% or 15% of the peak bandwidth. Based on this, we conclude that the cache size is large enough that the main memory bandwidth is not the bottleneck.

On the *Ampere Altra*, the behavior is different. Here we observe a consistent bandwidth utilization across all experiments. This is around 58% of the peak memory bandwidth. We do not observe a similar increase in memory bandwidth utilization as on the *Intel SPR* processor, which suggests that the memory bandwidth is a bottleneck. In Section 4.2, we observed a lack of scaling when using more cores on the *Ampere Altra*. The results in Table 4 indicate that this poor scaling emerges because performance is limited by the main memory bandwidth.

4.5 Roofline analysis

To explore the observed memory bottleneck we construct a Roofline plot on the best performing processor, i.e., *AMD Genoa*, in Figure 4. A Roofline model is designed to analyze if an application is compute or memory bound, and, if the latter, at which memory level. This is done by exploring the relation between memory bandwidth, at each memory level, and the peak performance of the processor [4]. We capture the operational intensity (OI) of HemoCell in GB/sec and the memory requests per numerical method, i.e., fluid and particle computation. We show the OI in relation to L1 and L2 requests.

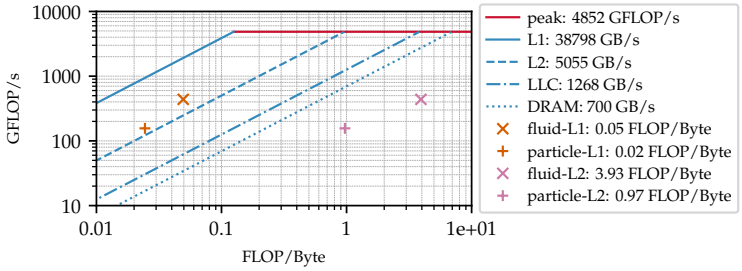

Fig. 4: Roofline model for S1 on AMD Genoa. Data is collected with ERT [6] and LIKWID [10]. Bytes processed is calculated as L1 requests × 8.

Figure 4 shows that HemoCell has a low OI, i.e., a high number of memory operations per FLOP, for both of the numerical methods. The model clearly shows that both kernels should be bound by the bandwidth of L2, LLC and main memory. However, due to the efficient utilization of L1, shown in Section 4.4, only 2.5% of the memory requests are serviced by L2, LLC, or main memory. This is confirmed by the higher OI in relation to the L2 requests. For these kernels, the OI is the number of FLOPs divided by the number of bytes requested from L2. We observe that these kernels are not bound by the L2 bandwidth anymore. However, depending on the miss rate on L2, performance can still be limited by LLC or main memory bandwidth. Thus, a small cache will cause the performance of HemoCell to be limited by the main memory bandwidth. This reaffirms that cache-size per process is the most influential feature. However, since the two numerical methods react differently to features, no single feature is sufficient to predict the ranking of the tested processors.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44

5 Conclusion

This work investigated the impact of different HPC processors on the performance of HemoCell. We found that our analysis can capture the diverse impact these processors' features have on performance. Based on the results, we conclude that (1) the CPU with the best performance and energy efficiency for HemoCell is the *AMD Genoa*, (2) the feature with the highest impact on HemoCell performance is the cache size per core, and (3) since energy use closely tracks runtime, cutting runtime remains key to reducing energy.

Beyond HemoCell Even with built-in instrumentation, and highly configurable setups, this study still required 869 benchmarks, over 116 hours, using six profiling tools producing data in ten file formats. The lack of a unified benchmarking framework remains a significant barrier for large benchmarking studies.

Acknowledgment We thank Eviden for providing access to HPC resources on their BullSequana supercomputer.

References

[1] Alowayyed, S., et al.: Load balancing of parallel cell-based blood flow simulations. Journal of Computational Science **24** (2018). https://doi.org/10.1016/j.jocs.2017.11.008
[2] Corbalan, J., et al.: Energy Optimization and Analysis with EAR. In: Cluster'20 (2020). https://doi.org/10.1109/CLUSTER49012.2020.00067
[3] Dongarra, J.J., Luszczek, P., Petitet, A.: The LINPACK Benchmark: Past, present and future. CCPE **15**(9) (2003). https://doi.org/10.1002/cpe.728
[4] Ilic, A., Pratas, F., Sousa, L.: Cache-aware Roofline model: Upgrading the loft. IEEE Comput. Archit. Lett. **13** (2014). https://doi.org/10.1109/L-CA.2013.6
[5] Knüpfer, A., et al.: Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope,Scalasca, TAU, and Vampir. In: Tools High Perform. Comput. 2011. Springer (2012). https://doi.org/10.1007/978-3-642-31476-6_7
[6] Lo, Y.J., et al.: Roofline Model Toolkit: A Practical Tool for Architectural and Program Analysis. In: High Perform. Comput. Syst. Perform. Model. Benchmarking Simul. LNCS, Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-17248-4_7
[7] McCalpin, J.: Memory bandwidth and machine balance in high performance computers. IEEE Technical Committee on Computer Architecture Newsletter (1995)
[8] McCalpin, J.D.: Bandwidth Limits in the Intel Xeon Max (Sapphire Rapids with HBM) Processors. In: High Perform. Comput. LNCS, Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-40843-4_30
[9] Patel, T., et al.: What does Power Consumption Behavior of HPC Jobs Reveal? In: IPDPS'20 (2020). https://doi.org/10.1109/IPDPS47924.2020.00087
[10] Treibig, J., et al.: LIKWID: Lightweight Performance Tools. In: Competence in High Performance Computing 2010. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24025-6_14
[11] van Dijk, J., et al.: Building a Fine-Grained Analytical Performance Model for Complex Scientific Simulations. In: PPAM'23. LNCS, Springer International Publishing (2023). https://doi.org/10.1007/978-3-031-30442-2_14
[12] Zavodszky, G., et al.: Hemocell: A high-performance microscopic cellular library. Procedia Computer Science **108** (2017). https://doi.org/10.1016/j.procs.2017.05.084

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_44