Comparison of AMD Zen 2 and Intel Cascade Lake on the Task of Modeling the Mammalian Cell Division

Maxim A. Krivov(✉) , Nikita G. Iroshnikov, Andrey A. Butylin, Anna E. Filippova, and Pavel S. Ivanov

Lomonosov Moscow State University, Moscow, Russia
m_krivov@cs.msu.su

Abstract. Modern architectures of central processors, in particular, AMD Zen 2 and Intel Cascade Lake, allow one to build shared memory systems with more than 100 computational cores. This paper presents the results of comparing the performance of these architectures shown on numerical modeling of mitosis in eukaryotes. The MiCoSi software that was developed by the authors and previously demonstrated a linear scalability when executed on cluster systems was used as a benchmark. The testing was performed on Amazon EC2 cloud nodes with 96 logical cores each. It is shown that, in relation to the problem of mitosis modeling, the two architectures under study have similar performance, while in some cases, Intel Cascade Lake bypasses its competitor AMD Zen 2 by 5–23%.

Keywords: Manycore processors · Shared memory · Parallel programming · Cell division · Mitosis · Prometaphase

1 Introduction

Modern processor architectures such as AMD Zen 2 and Intel Cascade Lake allow one to build computing systems with dozens or even hundreds of shared-memory computing cores. As a result, many theoretical and applied scientific problems that previously could be numerically solved on small clusters now can run on individual workstations.

This brings up the question of whether modern software packages can run on such systems in their natural SMP mode without any cluster emulation or separation of calculations by processes. This problem can relate both to architecture of the package itself, which does not allow to effectively use an increased number of cores, and to the specifics of memory organization, access to which is complicated due to the chiplet layout and multiprocessor configurations. As a result, the authors of numerous recently published papers assessed separately the

This work was supported by Russian Foundation for Basic Research (RFBR), grant 19-07-01164a (to M.K., N.I., A.F. and P.I.).

It is the author manuscript.
The final version of the paper is available at:
https://doi.org/10.1007/978-3-030-78759-2_27

scalability of various algorithms, libraries, and packages when they are executed on systems with a really large number of cores. We pursue similar goals considering the problem of modeling cell division which is typical of such disciplines as biophysics and computational biology.

This paper has the following structure. Section 2 contains a comparison of the 48-core Amazon EC2 cloud nodes with processors based on AMD Zen 2 and Intel Cascade Lake architectures. Section 3 provides a brief description of the studied process of a eukaryotic cell division (mitosis) which results in the formation of two daughter cells. Section 4 is dedicated to the structure of the numerical algorithm and the chosen parallelization scheme. The specific settings of the virtual experiments and the characteristics of the MiCoSi package we are developing are given in Sect. 5. Finally, Sect. 6 contains conclusions about the performance of both architectures in relation to the considered subject area.

2 Computer Systems

2.1 Technical Specifications

The calculations were performed on the c5ad (AMD) and c5d (Intel) nodes of the Amazon EC2 cloud, which allow to create a computing system with 48 physical cores and a total memory of 192 GB. It is worth explaining that in Amazon EC2, one can only select the number of cores for which a certain level of performance is guaranteed. The specific processor models and even their architecture may change depending on the configuration of the virtual node and probably the selected data center.

Our measurements have shown that AMD EPYC 7R32 processors based on Zen 2 are always allocated for c5ad nodes. As for the c5d family, the architecture does vary. The maximum configuration with 48 cores (c5d.24xlarge) uses Cascade Lake processors, but when the number of cores decreases to eight (c5d.4xlarge), the former are replaced by a processor model of previous generation, based on a very similar, but still different Skylake-SP architecture.

Information about the technical characteristics of the nodes presented in Table 1 was obtained by summarizing the official documentation and the results of third-party tests. Since the processors in question were created specifically for Amazon, the values of similar models are given for unknown parameters.

It should be also noted that Zen 2 and Cascade Lake architectures support multiprocessor configurations, and the number of cores provided by Amazon EC2 cloud does not correspond to their uppermost capabilities. According to the documentation, for AMD EPYC processors, the maximum possible number of cores per node equals 128 (2 processors with 64 cores) while for Intel Xeon this figure reaches 224 (8 processors with 28 cores).

2.2 Comparison of Zen 2 with Cascade Lake

Due to their relative novelty, the Zen 2 and Cascade Lake architectures have not yet received significant coverage in the scientific literature. As a result, the

Table 1. Technical characteristics of the computer systems used.

Amazon EC2 node	c5ad.24xlarge	c5d.4xlarge	c5d.24xlarge
Processor	AMD EPYC 7R32	Intel Xeon Platinum 8124M	Intel Xeon Platinum 8275CL
Architecture	Zen 2	Skylake-SP	Cascade Lake
Year	2019*	2017	2019**
Technology, nm	7+14	14	14
TDP, W	280	240	240×2
Number of cores/threads	48/96	8/16 from 18/36	$24/48 \times 2$
Base frequency, GHz	2.2*	3.0	3.0
Dynamic frequency, GHz	3.3	3.5	3.9
L1/L2 caches, KB/core	32+32/512	32+32/1024	32+32/1024
L3 cache, MB	192	24.75	35.75×2
Type of memory	DDR4-3200*	DDR4-2666	DDR4-2933**
Number of memory channels	8	6	6×2
Memory size, GB	192	32	192

**AMD EPYC 7552*
***Intel Xeon Platinum 8270*

comparison that follows is based on the information from technical presentations and documentation provided by their manufacturers, AMD and Intel.

AMD EPYC processors [1,2] are based on individual chiplets (Fig. 1), referred to as Core Complex Die (CCD). They are completely independent chips produced by 7 nm technology and subsequently combined into a single processor. Each such CCD chiplet, in turn, is divided into two groups of four cores (Core Complex, CCX), equipped with 16 MB of L3 cache memory. Another independent IO-chiplet is responsible for working with RAM and I/O interfaces. It is manufactured using 12 or 14 nm technology and adapted to a specific processor model. For example, the EPYC 7R32 processor uses an IO-chiplet manufactured by 14 nm technology and equipped with eight memory channels, which results in a total bandwidth of about 200 GB/s.

All CCD chiplets (and even CCX groups) cannot communicate with each other directly or address RAM explicitly. Corresponding requests must go through IO-chiplet to which they are connected by a high-speed bus. Depending on the model, the Zen 2-based processor can contain up to 8+1 chips, which in total provides up to 64 physical cores visible to the operating system. In addition to them, the server models contain another specialized ARM core responsible for implicit encryption of the contents of RAM.

Intel solutions [3] use a more classic monolithic architecture (Fig. 1), and the increase in the number of cores is provided by multiprocessor configurations. They may contain 10, 18, or 28 computing cores per chip but some of them may

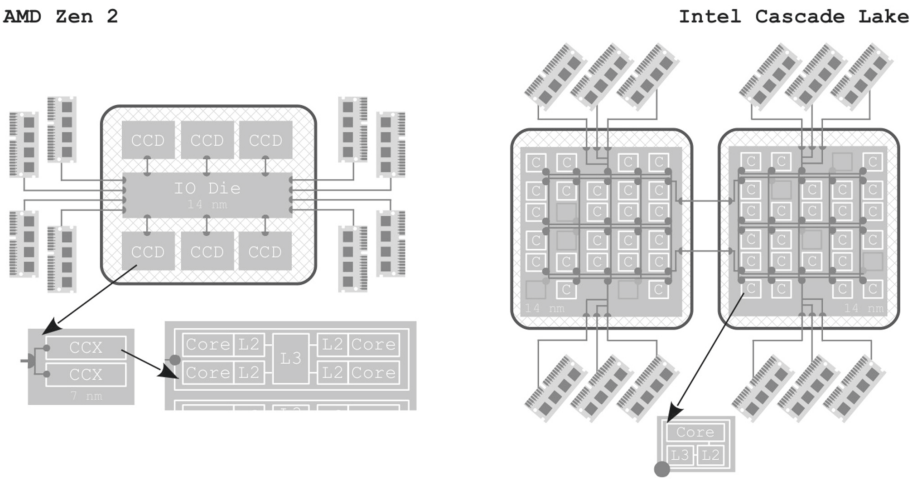

Fig. 1. The architecture of processors used in the c5ad.24xlarge (AMD Zen 2) and c5d.24xlarge (Intel Cascade Lake) nodes of Amazon EC2 cloud.

be disabled. For example, c5d.24xlarge nodes are based on processors with 28 cores, while only 24 of them are in operation. Each core has its own personal L2 cache of 1 MB, and also contains a portion of the total L3 cache (1.375 of 35.75 MB) available to all cores. Also, unlike Zen 2, the Cascade Lake architecture supports AVX-512 vector extensions, allowing, for example, the core to perform eight FP64 operations per CPU cycle.

All cores of a single processor are connected to each other by a high-speed on-chip interconnect with a topology in the form of rows and columns. Two memory controllers are implemented on each chip, providing a total of six channels and a bandwidth of about 128 GB/s per socket. Inter-chip interconnect referred to as Ultra Path Interconnect (UPI) and characterized by low latency and a bandwidth of about 60 GB/s, is used to communicate with cores and memory controllers from another socket.

It is almost impossible to determine which of these architectures is more preferable comparing only their technical characteristics. The main advantage of the chiplet approach is much larger number of transistors. For example, it allowed to significantly increase the total size of L3 cache (192 MB for Zen 2 versus 71.5 MB for Cascade Lake, see Table 1). However, due to the chiplet layout, L3 cache is no longer shared and a single core can use no more than 16 of 192 MB cache.

The situation with memory access looks quite similar. In Zen 2, all memory controllers are placed on a separate IO-chiplet, so the memory is unified, but at the cost of additional latency. In Cascade Lake, the access time depends on the exact location of memory controller that is processing the request, namely on the same chip as the core or in a neighboring socket. In the latter case, an additional data transfer is performed over the UPI links, which increases the latency and

potentially can even reduce the throughput. On the other hand, this architecture allowed for higher peak throughput (256 GB/s for Cascade Lake versus 200 GB/s for Zen 2).

2.3 Related Works

Usually, manycore processors are considered only as a tool for solving scientific problems. This explains the comparative scarcity papers dedicated to a full–fledged comparison of the performance of SMP systems from Intel and AMD. For example, the authors of [4] evaluate the scalability with the number of cores for the modified NAS Parallel Benchmark Suite, but the measurements were carried out on the already outdated architectures Intel Haswell (18 cores $\times$ 4 sockets) and AMD Piledriver (16 cores $\times$ 4 sockets). It is also worth noting similar estimates for single-socket systems based on Intel Haswell/Broadwell and AMD Zen, but for calculations using the Vienna Ab initio Simulation Package [5].

A larger-scale comparison of the Intel Skylake and AMD Zen architectures was undertaken in [6] by performing hydrodynamic calculations on 100 nodes of two supercomputers using an in-house software Hydro3D as a benchmark. According to their results, the cluster based on Intel processors had better scalability with the number of nodes. However, the AMD Zen architecture, due to the larger number of cores (64 versus 40 per node), still provided the shortest calculation time.

Of particular interest is the work [7], dedicated to the analysis of data with CERN LHCb detector in real time, which used processors based on Intel Skylake (2×16 cores) and AMD Zen 2 (1×64 cores). The transition from AVX2 to AVX-512 vector extensions, which are supported only by Intel processors, resulted only in a slight (about 10%) increase in performance, while in general, the AMD Zen 2 architecture appeared to be more preferable. For the same number of cores, it beat Intel Skylake by about 16%, but when all 64 cores were used, the gap increased to 93%.

3 The Simulated Biological Process

The cell cycle of the vast majority of eukaryotic cells consists of four repeated stages, the key of which is mitotic division that results in the formation of two genetically identical daughter cells. Violation of this process can lead to cell death or, even worse, to the appearance of aneuploid daughter cells with an altered set of chromosomes, which in some cases can lead to the development of malignant neoplasms [8]. There are various protective mechanisms that suppress possible “errors” of cell division, and currently the understanding of their work is an important fundamental task [9].

The complexity of studying mitosis arises from the small physical sizes of the objects under study (fractions of a micrometer), as well as the variety of biochemical reactions involved in the process of cell division. As a result, biophysicists actively use the apparatus of mathematical modeling to quantitatively

supplement and expand the experimental data. For example, authors [10] discussed the potentially exact formula for the polymerization rate of microtubules consisting of tubulin proteins when they start to interact with chromosomes during the first ten minutes of mitosis. An example of another area of research is the paper [11], in which a computer model of a yeast cell was used to evaluate such characteristics of living cells that have not yet been experimentally established.

The authors of this paper attempt to build their own complex mathematical model of mammalian cells division [12] which will holistically describe the course of three consecutive stages of mitosis, namely prometaphase, metaphase, and anaphase (Fig. 2). During the first stage, the nuclear membrane is destructed and chromosomes are released into the cytoplasm. Prometaphase is followed by metaphase with growth of tubulin microtubules and the appearance of their attachments to the chromosomes to pull the latter to the corresponding spindle poles. After some time, a balance of exerted forces aligns attached chromosomes in the equatorial plane of the cell. This stage is followed by anaphase, in which the centromere that binds the sister chromosomes breaks leading to the chromosomes distribution between two new daughter nuclei.

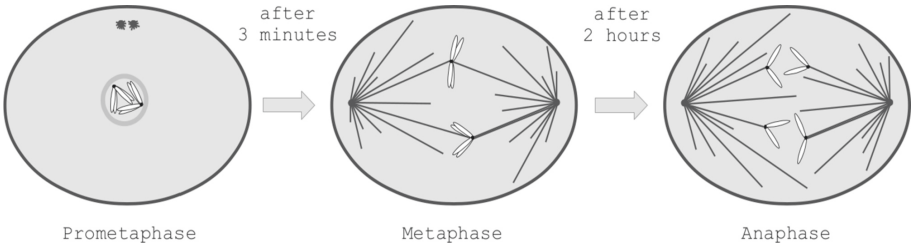

Fig. 2. Schematic representation of the stages of cell division accounted in the proposed model.

It is worth noting that there are numerous papers dedicated to the construction of mathematical models of the dividing cell and its individual parts, often based on mutually exclusive approaches to the description of mitosis. Many well-known models that were relevant as of 2012 are described and classified in the review [13].

4 Numerical Algorithm

The model of eukaryotic cell under consideration belongs to the class of the mechanical ones. Specifically, the chromosomes are represented by the closest geometric shapes (Fig. 3), which move both under the action of forces from the microtubules and according to some artificial laws that depend on the model settings. If the viscosity of cytoplasm is accounted for, the acceleration of chromosomes can be neglected. Knowing the forces acting on the chromosome at

a particular time, one can calculate its linear and angular velocities. Further, by numerically integrating the velocities using the one-step Adams method, one can reproduce the trajectory of the chromosome motion during the modeled time interval $[t_{start}, t_{end}]$.

The structure of the numerical algorithm is shown in the listing in Table 2. From the point of view of biophysics, it is interesting to study not individual cells, but their ensembles consisting of 100 to 500 identical cells. Thus, the algorithm by its nature has a fairly good potential for parallelization (lines 2 and 5). To simulate a single cell, three loops at each time step should be performed. In the first (lines 7–14), the state of microtubules is updated and their collisions with chromosomes are searched. If the microtubule comes into contact with the chromosome, it can attach to its kinetochore (line 13), and also, depending on the model settings and probabilistic events, break off or begin to shorten.

The second loop (lines 16–20) is responsible for updating the position of the chromosomes. Each of them is affected by three types of forces (line 17) that originate from the dynamics of microtubules, Brownian motion of molecules, and the influence of the sister chromosome. Knowing the sum of these forces, one can construct a SLAE with a 6×6 matrix, the solution of which is the values of linear and angular velocities (line 18). After that, it is possible to update the position and orientation of the pair of sister chromosomes (lines 19–20). Finally, the last loop (lines 22–23) consists of only two iterations and is required to move the spindle poles artificially.

The main computational complexity arises when the loop for microtubules is executed (lines 7–14). The attempts to adapt it to the SIMD architecture were not very successful due to the large number of branches within a single iteration. For example, for the scenarios discussed in Sect. 5, the CUDA version of the algorithm provided an acceleration of about 2- to 6-fold relative to a single processor core. The application of techniques such as reordering iterations and using a hardware rasterizer allowed to increase the speed up to dozens of times, but significantly complicated the code and hindered the development of the mathematical model.

On the other hand, a more elegant solution turned out to be the parallelization of the loop by cells for the MIMD architecture (line 5), for which OpenMP+MPI technologies were used. Our measurements have shown [14] that when the ensembles of 1000 cells were simulated on ten nodes of the Lomonosov-2 cluster, an ideal scalability across cores was observed. Thus, this algorithm is a fairly illustrative example of the fact that manycore architectures such as Zen 2 or Cascade Lake can be in demand, including their usage in scientific modeling. The reason lies in the fact that some problems of computational biology are difficult to adapt to graphics accelerators while the computing power of classical clusters turns out to be redundant for them.

Table 2. Pseudocode of an algorithm that implements one of the versions of the cell division model proposed by the authors.

```
1  t ← t_start
2  cells[1..N] ← InitializeCells()
3
4  while t < t_end
5    parallel for each cell in cells
6
7      for each mt in cell.mtubules
8        ProbabilisticEvents(mt)
9        GrowOrShrink(mt, Δt)
10       if not mt.bound
11         for each chr in cell.chromosomes
12           if HasCollision(mt, chr) and MayAttach(mt, chr)
13             Bind(chr, mt)
14             break
15
16     for each chr in cell.chromosomes
17       force, torque ← ComputeForces(chr.bound_mtubules)
18       v, w ← ComputeVelocities(force, torque)
19       chr.pos ← chr.pos + v * Δt
20       chr.orient ← chr.orient + w * Δt
21
22     for each pole in cell.poles
23       MovePole(pole)
24   t ← t + Δt
```

5 Testing Methodology

To evaluate the performance of computing systems, we used the open source package MiCoSi [12]. It contains the implementation of the three-dimensional mathematical model of a dividing cell described in Sects. 3 and 4, and also has interfaces to flexibly configure numerical experiments and to measure physical quantities of interest. The solver version corresponded to the git revision eb11432, the package was built using the Visual C++ 2017 compiler and successfully passed all functional and unit tests.

The choice of the Amazon EC2 cloud and the Windows platform for testing was caused by the fact that MiCoSi software package is primarily addressed to users with modest programming skills. It is assumed that, using the user-friendly C# language, they will be able to create and debug a simple program on a local machine and then export the data they are interested in, for example, in the form of CSV tables for subsequent analysis using Excel or Python. In this case, to conduct large-scale computational experiments, it will be enough to transfer

several files to the Amazon EC2 node, run calculations there, and then download tables with the results.

As a specific numerical experiment used as a benchmark, a program was developed to simulate prometaphase, a stage of cell division that lasts about 180 s and is characterized by the divergence of spindle poles in diametrically opposite parts of the cell. Such divergence generates a large number of interactions between tubulin microtubules and pairs of chromosomes that are freely floating in cytoplasm.

Two basic scenarios were considered. In the first, hereinafter referred to as 'Compute-bound' (Fig. 3a), the virtual cell consists of three pairs of chromosomes and 3000 microtubules. The simulation is performed in 0.1-s increments, and only the parameters of the final state of the cell are saved to the disk. This scenario generates a substantial computational load for performing geometric checks, as well as drawing up and solving SLAE for the subsequent determination of velocities. The second scenario, designated as 'IO-bound' (Fig. 3B), describes a simpler cell with one pair of chromosomes and 1000 microtubules. In addition, the states are unloaded every 0.01 s, resulting in about 628 MB of output data. As a result, the operations of serialization and packaging of information in the binary stream format *.cell begin to prevail.

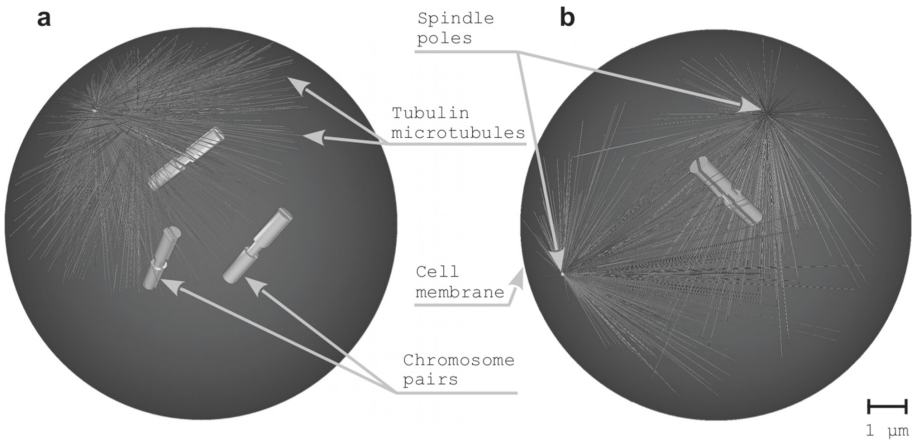

Fig. 3. Visualization of cells used for numerical simulation of prometaphase. (a) 'Compute-bound' scenario, 3 pairs of chromosomes and 3000 microtubules; (b) 'IO-bound' scenario, 1 pair of chromosomes and 1000 microtubules.

The size of the ensemble, depending on the type of test, was 48 and 384 virtual cells. The number of processor cores involved in the simulations was constrained by limiting the tasks, i.e. by subdividing the cells into groups of cells with exactly predetermined size. The total time spent executing the Launcher.StartAndWait() method was used as a metric. Testing was accomplished in the Amazon EC2 cloud on c5d (Intel Skylake-SP and Cascade Lake)

and c5ad (AMD Zen 2) nodes with Windows Server 2019/20H2 installed. The software was launched from local NVMe disks that have a physical connection to the nodes.

6 Results and Discussion

6.1 Support for a Large Number of Cores Is Required Not Only from Applications, But Also from Operating Systems and Compilers

In a list of changes made over the past three years to the Visual C++ and gcc compilers, one can notice references to a lot of improvements in the architectures under study. In addition to the expected progressive enhancements, a support for chiplet processors is also declared, which should result in a noticeable performance gain for some patterns of memory operations. To evaluate this effect, the testing was performed using two versions of Visual C++ 2017 compiler, namely VC++ 14.1 (May 2017) and VC++ 14.16 (January 2021).

The situation with Windows Server operating system looks similar. According to the documentation from AMD, it is required to use versions that are based on at least the update 1903 (May 2019), as the new scheduler is better aware of the structure of caches, which increases the final speed by dozens of percent. As our tests have shown, these limitations are much more severe. The version based on the previous update 1809 (October 2018), installed in the Amazon EC2 cloud by default, forcibly sets the affinity mask and does not allow a single process to run on all the cores. Only 24 physical cores were used on the c5d.24xlarge (Cascade Lake) nodes, while only 32 physical cores were available on the c5ad.24xlarge (Zen 2) nodes. When the system was upgraded to version 20H2 (October 2020), these problems disappeared.

The total effect of the simultaneous update of the operating system and the compiler was indeed significant. When using 8–24 cores in the 'Compute-bound' scenario, the computation time was reduced by about 7–12% (Cascade Lake) and 9–18% (Zen 2). The greatest acceleration was observed for the 'IO-bound' scenario on the Zen 2 processor with the gain in the range 27–50%, which significantly improved the scalability with the number of cores.

6.2 For Intensive Computations on a Small Number of Cores, Cascade Lake and Zen 2 Demonstrate Similar Performance

Figure 4a shows the measurements at the c5d.4xlarge (Cascade Lake or Skylake-SP, depending on the launch) and c5ad.4xlarge (Zen 2) nodes, with eight physical cores each. In such scenarios, Intel Xeon processors can run at slightly higher clock speeds (3.9 GHz vs. 3.3 GHz), which may explain their better performance, although the gain was insignificant. In both cases, the scalability was near-linear (6-fold and 6.9-fold on eight cores). After switching to virtual cores, the Intel technology, which allows for the division of one physical core into two logical

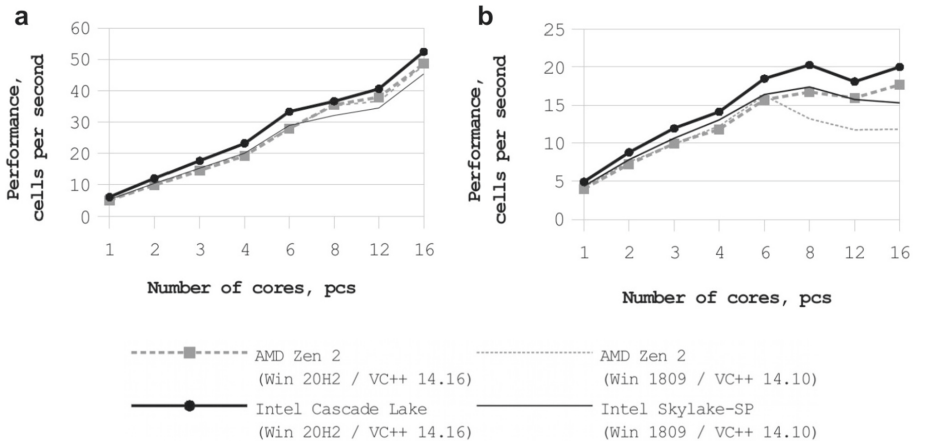

Fig. 4. Performance comparison when using c5ad.4xlarge (AMD Zen 2) and c5d.4xlarge (Intel Cascade Lake) nodes on an ensemble of 48 cells, more is better. (a) 'Compute-bound' scenario, without data unloading; (b) 'IO-bound' scenario, uploading data in 0.01 s increments.

ones, demonstrated slightly better results. For example, 16 Cascade Lake logical cores were 43% faster than 8 physical cores. For Zen 2, the speed up was 36%.

Unfortunately, the Cascade Lake and Skylake-SP architectures can hardly be compared objectively, since the nodes with Windows Server 20H2 have always been created only on the basis of Cascade Lake. Neglecting these differences, although it is not quite correct, the performance gain from switching to a new platform can be estimated as 13–17%.

6.3 On Active Memory Operations, Cascade Lake Is Slightly Ahead of Zen 2

The 'IO-bound' scenario (Fig. 4b) predictably demonstrated that from a certain point on, adding computational threads lead only to a decrease in performance. In both cases, this threshold turned out to be 6–8 cores. It is important to note that when operations with memory dominate over computations, the Cascade Lake architecture shows better results, consistently beating Zen 2 by 13–23%. These observations correlate well with technical specifications of these two architectures. At full load, Cascade Lake provides a larger number of memory channels per core (0.25 vs. 0.167 for Zen 2) and higher bandwidth per core (5.34 vs. 4.16 GB/s for Zen 2). It is also worth to account for the fact that the chiplet architecture of Zen 2 processors always causes additional latency, while Cascade Lake has cores and memory controllers both located on the same chip.

6.4 When the Second Socket Cores Are Used, Cascade Lake Sometimes Loses to Zen 2

The most interesting results have been obtained when the MiCoSi package was executed on all 96 logical cores of the c5d.24xlarge and c5ad.24xlarge nodes (Fig. 5). As noted in Sect. 6.1, these tests could only be performed on Windows Server 20H2, since older systems artificially limited the number of cores available to the process.

When the absolute performance is compared (Fig. 5a), the graph is clearly divided into two parts. In the range of 8–24 cores, Cascade Lake is the winner, consistently outperforming Zen 2 by 5–15%. However, for 32–96 cores, episodic failures begin to occur, thus allowing Zen 2 to significantly bypass its competitor. Such a picture is very characteristic of multi-socket systems which require a separate optimization of the executed program with special account for the physical heterogeneity of memory.

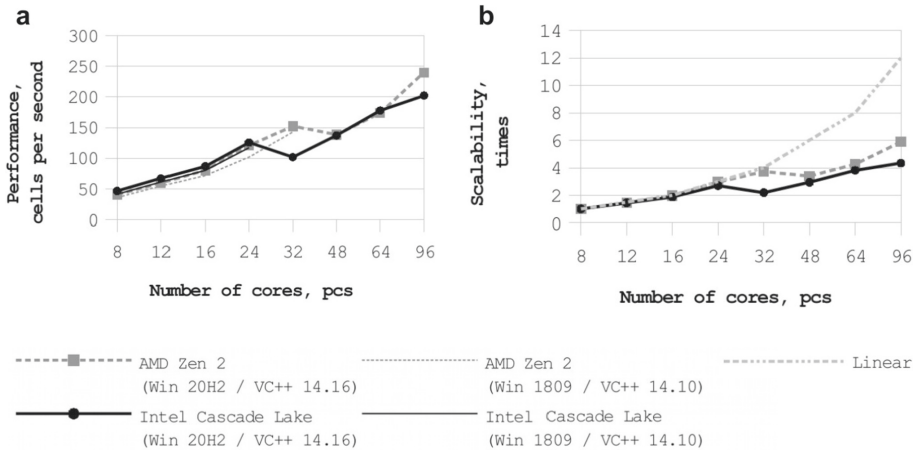

Fig. 5. Comparison of the performance of c5ad.24xlarge (AMD Zen 2) and c5d.24xlarge (Intel Cascade Lake) nodes with a predominance of computing load on an ensemble of 384 virtual cells, more is better.

Modern processors dynamically adjust the frequency even for a group of cores, remaining within the acceptable TDP level, which can somewhat "worsen" the scalability curve at their maximum load. Accordingly, the slight performance degradation observed for Zen 2 when switching from 32 to 48 cores may be of a similar nature. The base frequency of this processor is only 2.2 GHz, while for a group of cores it can increase up to 3.3 GHz.

It is also worth noting that in the relation to the effect of using logical cores, in the analogy with Sect. 6.2, a system based on AMD processors gets a greater benefit. For Zen 2, the acceleration brought about by the transition from 48 physical cores to 96 logical ones was 73%, while for Cascade Lake it appeared to be just 47%.

7 Conclusion

The use of synthetic benchmarks such as Linpack and HPCG is the most objective method of hardware platforms comparison, since they make it possible to measure the performance in absolute terms and approach the theoretically possible limits. Nevertheless, applied benchmarks that estimate the relative performance when solving a problem from an applied discipline are of particular interest. They are the ones that allow for making preliminary conclusions about the speedup one could expect when migrating to a new hardware platform.

The results of the comparison of Zen 2 and Cascade Lake in solving the considered problem of computational biology are an illustrative example of such an applied benchmark. The architectures considered are based on two alternative approaches: a chiplet layout and a large monolithic die. They can be easily compared based exclusively on technical characteristics or by extrapolating synthetic tests, while the latter is not quite correct. For example, Cascade Lake processors are produced using the formally outdated 14 nm technology and contain several times less L3 cache memory. Meanwhile, with the same number of cores, they have higher frequencies and support the new AVX-512 instruction set. Thus, it is quite difficult to make a full and objective comparison without solving applied problems from significantly different subject areas.

If one focuses only on the cost of renting the appropriate hardware, for the MiCoSi package, the Cascade Lake architecture is a preferred choice. The c5d and c5ad nodes presented in the Amazon EC2 cloud are positioned as interchangeable ones with the same rental price per core. At the same time, Intel solutions provide better performance in the scenario of intensive work with memory (13–23%) and slightly higher speed when accomplishing computations on the cores of a single die (5–15%). The only advantage of Zen 2-based processors is a much higher speed when running on 32 and 96 cores (19–50%). However, due to the lack of linear scalability with the number of cores, there are all reasons to believe that the observed 'performance spike' is caused by insufficient optimization of the MiCoSi package or by the chosen testing method.

References

1. Suggs, D., et al.: AMD "ZEN 2". In: 2019 IEEE Hot Chips 31 Symposium (HCS), pp. 1–24. IEEE Computer Society (2019)
2. Suggs, D., Subramony, M., Bouvier, D.: The AMD "Zen 2" processor. IEEE Micro **40**(2), 45–52 (2020)
3. Arafa, M., et al.: Cascade Lake: next generation Intel Xeon scalable processor. IEEE Micro **39**(2), 29–36 (2019)
4. Cho, Y., Oh, S., Egger, B.: Performance modeling of parallel loops on multi-socket platforms using queueing systems. IEEE Trans. Parallel Distrib. Syst. **31**(2), 318–331 (2019)
5. Stegailov, V., Vecher, V.: Efficiency analysis of Intel and AMD x86_64 architectures for Ab initio calculations: a case study of VASP. In: Voevodin, V., Sobolev, S. (eds.) RuSCDays 2017. CCIS, vol. 793, pp. 430–441. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71255-0_35

6. Ouro, P., Lopez-Novoa, U., Guest, M.: On the performance of a highly-scalable Computational Fluid Dynamics code on AMD, ARM and Intel processors. arXiv preprint arXiv:2010.07111 (2020)
7. Hennequin, A., et al.: A fast and efficient SIMD track reconstruction algorithm for the LHCb Upgrade 1 VELO-PIX detector. J. Instrum. **15**(06), P06018 (2020)
8. Ben-David, U., Amon, A.: Context is everything: aneuploidy in cancer. Nat. Rev. Genet. **21**, 44–62 (2020)
9. Cimini, D.: Detection and correction of merotelic kinetochore orientation by Aurora B and its partners. Cell Cycle **6**, 1558–1564 (2007)
10. Wollman, R., et al.: Efficient chromosome capture requires a bias in the 'search-and-capture' process during mitotic-spindle assembly. Curr. Biol. **15**, 828–832 (2005)
11. Edelmaier, C., et al.: Mechanisms of chromosome biorientation and bipolar spindle assembly analyzed by computational modeling. Elife **9**, e48787 (2020)
12. Krivov, M.A., Ataullakhanov, F.I., Ivanov, P.S.: Evaluation of the effect of cell parameters on the number of microtubule merotelic attachments in metaphase using a three-dimensional computer model. In: Panuccio, G., Rocha, M., Fdez-Riverola, F., Mohamad, M.S., Casado-Vara, R. (eds.) PACBB 2020. AISC, vol. 1240, pp. 144–154. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-54568-0_15
13. McIntosh, R., et al.: Biophysics of mitosis. Q. Rev. Biophys. **45**, 147–207 (2012)
14. Krivov, M.A., et al.: Modeling the division of biological cells in the stage of metaphase on the "Lomonosov-2" supercomputer. Numer. Methods Program. **19**, 327–339 (2018). (in Russian)