Analysis of Many-core CPUs Simulators

Daniel Jorge Rodrigues Ferreira,
Instituto Superior Técnico, Universidade de Lisboa

Abstract—Current trends predict many-core processors with thousands of cores by the end of the decade. Yet currently used interconnection networks and cache coherence protocols fall short in harvesting all their promised performance. Furthermore, programming on these systems poses a new set of challenges, but also opportunities.
These problems motivate research on the topic of many-core processors, trying to provide a set of insights about the best options on interconnection networks, on cache coherency, on the memory architecture, and on the parallelization strategy. All this with the goal of providing the maximum throughput of the processor.
To perform this research an accurate testing platform is necessary. The available information depicts gem5 as one of the most accurate simulators, while Sniper provides a good compromise between performance and accuracy. These claims are not entirely supported by the tests performed, existing significant discrepancies between what was expected and the actual results, and between the simulators' results themselves.

Index Terms—Many-core processors, gem5, Sniper, Graphite, PARSEC

1 Introduction

TODAYS multicore processors are the result of the pursue of more computational power. These are part of the tendency to increase the number of cores per processor, that has already lead to some sixty-four core high-end processors. Also, it is predicted that by the end of the decade, there will be thousand core processors [1].

This new paradigm brings a set opportunities and challenges, namely on the internal processor network and on the programming model, existing multiple solutions that try to extract the highest possible performance. The internal processor network must be prepared to handle the high number of cores, by providing the adequate inter-core communication and cache coherence. On the other hand it is necessary to prepare todays programs for this new paradigm, by taking advantage of program parallelisation and optimisation. Both topics are on-going research topics.

The research of both this topics is desirable, but a problem emerges: *How to perform tests on non-existing processors?* The answer is many-core simulators. The use of simulators allows the test of possible future architectures through the execution of programs on them, thus enabling the adaptation of programs to this new paradigm. Furthermore, this type of simulations allows the identification of the most promising architectures without actually having to build them, decreasing the time and cost needed to research them.

2 Many-core Processors

2.1 Interconnection Networks

Today's common off-the-shelf processors have a global bus that connects all the available cores. This is a good solution for low core counts but, as there can not be concurrent accesses to the bus, it does not scale gracefully.

The alternative is using an interconnection network, or Network-on-Chip(NoC) [2], as shown on Figure 1. Here the tiles are connected by a 2D mesh interconnection network.

Each tile, as shown on the right side of Figure 1, is usually constituted by the core itself with L1 instruction and data caches, both of them private, a L2 cache that can be shared or private, and a router able to send, and forward messages.

To connect the cores on a many-core Chip Multiprocessor there are multiple available

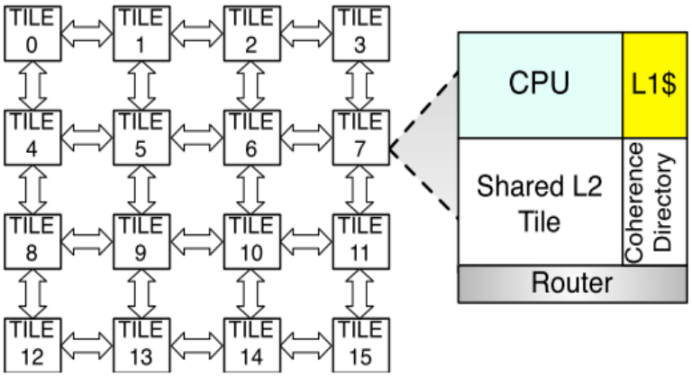

Fig. 1: 16-core tiled CMP, using a 2D mesh interconnection network with a distributed directory [3].

technologies: electrical networks; on-chip optical nanophotonic networks [4]; 3D interconnects [5]; and on-chip wireless communication [6]. The electrical networks are the only type of networks discussed here due to them having the smallest set of design and manufacturing challenges.

The electrical networks [7], [8], [9] consist in having each tile connect by one or more buses to other tiles. These are simple and only require available manufacturing methods. However these networks usually have have a big hop count from one end to another [10].

Each electrical network makes a different set of trade-offs. To evaluate and compare them the following set of key aspects should be considered:

- diameter: the greatest number of hops from one end to the other of the network;
- average inter-node distance: the average number of hops between any two tiles;
- bisection width: the minimum number of wires need to cut to split the network into two;
- cost: the number of links needed to build the network;
- router degree: the number of ports on the routers.

The electrical networks can be divided in Direct and Indirect. The former is characterised by having every router attached to a tile, while the latter may have routers that are not connected to tiles. Notwithstanding, the tiles maintain their structure.

Amongst the most discussed Direct interconnection networks are: 2D Mesh, 2D Torus (Fig. 2), Spidergon, and Spidergon-Donut (Fig. 3). As for the most common Indirect interconnection networks, they are: Fat Tree, Butterfly, Flattened Butterfly, and Star.

Table 1 contains information about the key aspects mentioned for each one of the interconnection networks presented before.

From the analysis of Table 1 the Spidergon-Donut(SD) [10] stands out. This network has an acceptable router degree of five, and has a low diameter.

2.2 Cache Coherence

When dealing with multiple cores, containing individual caches, the problem of cache coherence emerges. In order to produce the correct results it must be guaranteed that every core is able to read the last value written to any memory position. This problem is already addressed in current single and multiple core processors, with strategies also valid for many-core processors. There are multiple variations of cache protocols that assure this, with different performances on many-core processors.

This section gives an overview of the existing cache coherence protocols, and what is their viability on many-core processors, divided in: Snooping, and Directory.

2.2.1 Snooping

The snooping protocol [11] consists in having each core monitoring a bus shared by all the cores. It carries the information about each read and write of all the cores. So, using a Write

Invalidate protocol or a Write Update protocol it is possible to assure cache coherence.

Both of these protocols are part of the possible solutions, for today's common two and four cores multicore processors. These have the advantage of being straightforward solutions that provide a good performance. Nevertheless, they have the major disadvantage that they do not scale properly. To implement them a shared bus must be guaranteed between all the cores, so that the operations made by each one can be snooped by the other ones. This is manageable between a few cores but it does not work for hundreds or thousands of cores.

As a many-core processor should have an interconnection network, see 2.1, this option must be discarded.

The only option available that respects these restrictions is the Directory based cache coherence protocols.

2.2.2 Directory

The directory based cache coherence protocols [11] consist in having a shared memory that explicitly controls the coherence between the caches. The directory receives information about each cache copy, updating or invalidating cache lines accordingly.

This bypasses the problem of the snooping protocol that requires all the cores to share a bus. So, despite it having severe performance penalties [11], it constitutes the best solution for many-core processors.

To implement a directory, it can be either one single memory block, or a distributed memory block. The former has the advantage of constituting a simple solution, however it creates a bottleneck. Also, it may be built based either in a fast on-chip SRAM, or on an off-chip DRAM which imposes a slowdown to the caches. There is also the in-cache directory [11], Figure 1, that takes advantage of the L2 shared caches by using them as a distributed directory.

There are multiple directory based cache coherence protocols that provide different tradeoffs, some of the most discussed are: Full-map directories [11], Duplicate-tag-based directories [11], Sparse directory [11], In-cache directory [11], and ACKwise [14].

3 SIMULATORS

Simulators provide a simple mean of test and allow the execution of as many simulations of different configurations as needed. However, their performance is a bottleneck, limiting both the depth and breadth of the experiments that can be performed.

Also, most of them do not execute in parallel parallel [15], [16], [17], [18], [19].

Ideally, a simulator should: be accurate, fast, execute a wide range of workloads, and be easy to use and modify [20].

The following simulators are analysed: gem5, Graphite and Sniper.

3.1 gem5

The gem5 [21] is an emulation-based, user-level and full-system, PDES. This simulator

TABLE 1: Network comparison for 1024 cores.

Network	Diameter (inter-router hops)	Average Distance (hops)	Bisection Width (links)	Link Cost (links)	Router Degree (ports)
2D Mesh (2^5x2^5)	62	21.33	32	1984	4
2D Torus (2^5x2^5)	32	16	64	2048	4
32-ary 2-flat Flattened Butterfly	1	1	512	1.5K	63
8-ary 16-fly Butterfly	15	15	512	17K	16
Fat Tree (32 nodes/leaf router)	11	9.07	1024	6144	33
Hypercube HC(2^{10})	10	5	512	5120	10
Star HC(2^{10})	0	0	N/A	1024	1024
HR3(N2=2^3, N1=2^3, N0=2^4)	28	14	4	1154	4
Spidergon SG(2^{10}x)	256	128.62	514	1536	3
SD(N1=2^4,N0=2^6)	24	12.89	128	2056	5
4xconcentrated SD(16,16)	12	6.87	32	640	9

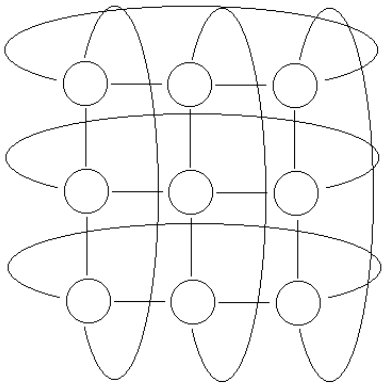

Fig. 2: 2D Torus layout [12].

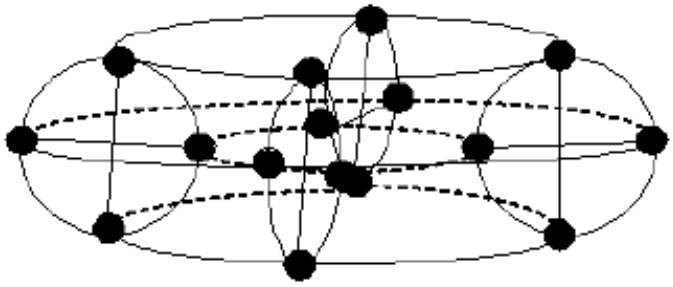

Fig. 3: Spidergon-Donut layout [13].

was originated by two previous simulators the M5 [22] and GEMS [23]. The former provided an "extremely configurable framework, multiple ISA and multiples CPU models" [21], and the latter complements "this capacities with a detailed and flexible memory model, including support for multiple cache coherence protocols and interconnection models" [21]. However, it has a very poor performance.

3.2 Graphite

Graphite [24] is an instrumentation based, user-level, PDES simulator. The purpose of this simulator is to simulate multicores with hundreds or thousands of cores using one or more hosts, one of its strong points. The instrumentation is carried out with Intel's Pin accordingly to a Dynamic Binary Translation (DBT) scheme, translating the instructions as they are executed.

3.3 Sniper

Sniper [25] is also an instrumentation based, user-level, PDES simulator. This simulator was built upon Graphite but it implements interval simulation. This type of simulation brings advantages to the precision of the simulation reducing the error on a single thread simulation, from 114% to 19.8%, and on a multi-thread simulation from 59.3% to 23.8%.

3.4 Simulator Comparison Conclusion

Having the information about the most well-known multicore simulators, it is possible to compare them with the goal to chose the one that provides the best performance, and results achievable.

Table 2 provides a comparison of multiple simulators, the described specifications are explained next:

- Engine: identifies in what way the binary code is executed by the simulator;
- Parallelisation: specifies the name of the technology used to manage the simulation;
- Detailed Core: identifies the simulators that support out-of-order execution (OOO), a characteristic necessary to simulate more complex modern processors that can execute instructions ahead, while waiting for the end of a memory access;
- Detailed Uncore: identifies the simulators that can include in their simulation process other elements exterior to the core, that can be essential to the core performance, as for example a L3 cache or a memory controller;
- Full-System: indicates which simulators support this type of simulation;
- Multiprocess apps: identifies the simulators that support applications that can start, not only more than one simulated thread but more than one simulated process, which can be useful in some tests;
- Managed Apps: identifies the simulators that support applications that are based upon virtual machines, as Java, Scala, or Python;
- Multi-host: identifies the simulators in which it is possible to distribute the sim-

ulation effort through a network of machines (grid, cluster, etc).

Concluding, the simulator that presents itself as the best option is the ZSim. However, it is not made publicly available by the authors. So, to perform the tests gem5 and Sniper will be used. The former provides more detailed results while the latter can perform simulations with a higher number of cores, but with less accurate results. This provides a more accurate study of the architecture, allowing double check of the results. Furthermore, Graphite makes part of the choice due to its exclusive ability to use multiple hosts to execute a simulation, and also to provide another way to double check some results.

4 Simulators' Results Comparison

Using three different simulators brings forward the issue of their validity. Although each one of them has been proved to have some degree of correction, it is still an open question the similarity of the results obtained with either one.

So, to gain more confidence in the results of the previously chosen simulators they are analysed, tested and cross examined.

The effective use of gem5, Sniper, and Graphite presents some limitations in the broadness and scope of the performed tests, fact that enforced a series of constraints. Next are described the trade-offs made to achieve a similar test execution procedure.

4.1 Compatible Simulator Settings

Sniper, Graphite, and gem5 present significant different characteristics and capabilities, as shown on Table 2. This is also visible by not being possible to achieve the exact same configuration in every simulated aspect, due to multiple constraints that the simulators impose.

The key factors specified for each configuration simulator are depicted in Table 3.

Due to Sniper constituting a Graphite's derivative project they have a very similar configuration, only differing in the type of the CPU. Despite their similarity the in-order core out-of-order memory (IOCOOM) CPU type shipped with Sniper, present in Graphite, does not function. So, an OOO CPU must be used. This difference makes it likely to observe faster executions on the simulated time in Sniper.

On the other hand, gem5 presented more challenges. To achieve the obtained configuration, first the closest type of cache coherence was chosen and here the issues began, no MSI scheme was available. The closest cache coherence mechanism available was MOESI, due to it including a directory, which was not available in no other configuration. But even this directory was different from the other. This one was localised in the last level of cache instead of in DRAM. Also, to have a 2D mesh it is enforced the use of an in-order CPU. So, gem5's results are expected to be significantly slower.

4.2 PARSEC

To provide a standardised mean of comparison between the three simulators two workloads of the PARSEC [26] benchmark suite were executed and analysed. These results can be easily reproduced due to PARSEC's availability online.

The two chosen workloads are *blackscholes*, *swaptions*, and *fluidanimate*. These were picked because they were possible to execute on all three simulators and they presented consistent results. Furthermore, due to Sniper and gem5 only supporting the older *2.1* version of PARSEC, Graphite was adapted to support this version. Nevertheless, the results for both *2.1* and *3.0* versions of PARSEC on Graphite are included in the results displayed.

4.2.1 blackscholes

The first workload tested across all three simulators was *blackscholes*. The following figures ease the effort to analyse its behaviour in the multiple simulators.

In Figure 4 it can be observed that, notwithstanding the different versions of PARSEC, the simulated execution time of Graphite with its two different versions is very similar, while being much higher than Sniper's. So, it seems to

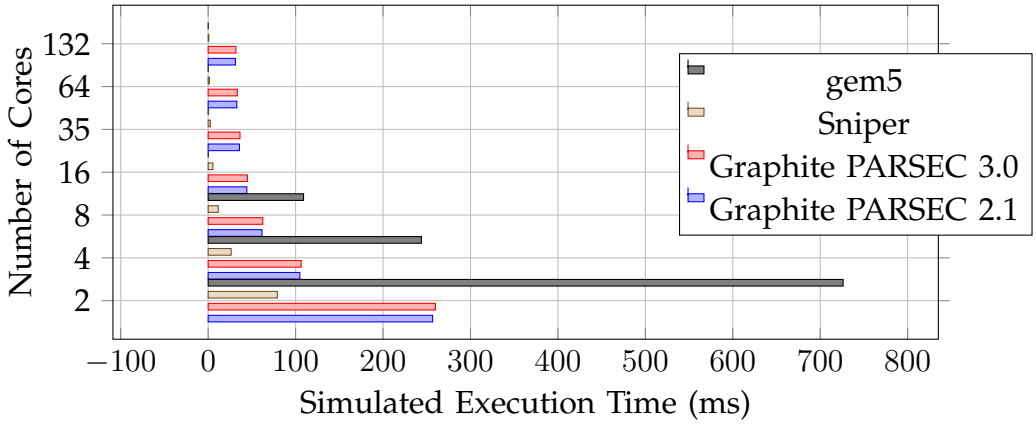

Fig. 4: blackscholes simulated execution time.

indicate that the version of *blackscholes* included in PARSEC *3.0*, and in PARSEC *2.1* has the same performance on Graphite, proving some similarity.

Sniper is the simulator with smallest simulated execution times. This is due to its faster OOO CPU that is able to improve the speed of this workload because it relies more on computation than in communication.

gem5's results are the highest by a large margin due to its simpler, and slower CPU type. However an interesting effect is present. As the number of cores increases its different cache coherence mechanism allows it to reduce the gap to the other simulators.

To bypass the problem of analysing results from different configurations, these can be put in proportion.

Figure 5 allows two different conclusions to be drawn. First, in Sniper and in Graphite if the same version of *blackscholes* is executed then the same proportion of simulated execution

TABLE 2: Simulator comparison.

Simulator	Gem5	CMPSim	Graphite	Sniper	HORNET	SlackSim	ZSim
Engine	Emulation	DBT	DBT	DBT	Emulation	Emulation	DBT
Parallelization	Sequential	Limited Skew	Limited Skew	Limited Skew	PDES	PDES	Bound-Weave
Detailed Core	OOO	No	No	Aprox. OOO	No	OOO	DBT-based OOO
Detailed Uncore	Yes	MPKI only	Approx. contention	Approx. contention	Yes	Yes	Yes
Full-System	Yes	No	No	No	No	No	No
Multiprocess apps	Yes	Yes	No	Trace-driven only	Trace-driven only	No	Yes
Managed apps	Yes	No	No	No	No	No	Yes
Multi-host	No	No	Yes	No	No	No	No

TABLE 3: Architectures simulated by the simulators.

Characteristic	**gem5**	**Sniper**	**Graphite**
Interconnect Network	2D mesh	2D mesh	2D mesh
Cache Coherence	MOESI	MSI	MSI
Cache Levels	2	2	2
Directory Location	LLC	DRAM	DRAM
Directory Type	fullmap	fullmap	fullmap
CPU Type	in-order	out-of-order	in-order complex out-of-order memory

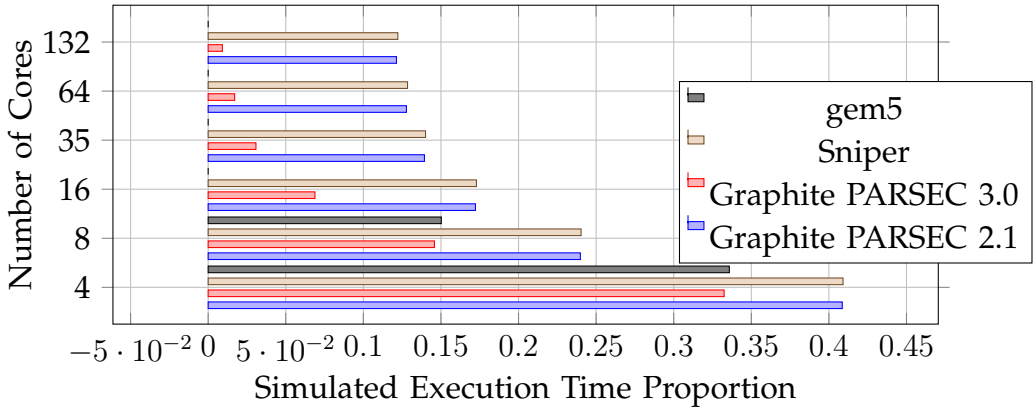

Fig. 5: blackscholes simulated execution time proportion relative to two cores.

time is observed. This indicates that, despite the differences in the simulator and in the configuration of the architecture, it is possible to assure that the differences in the simulated time are only due to this differences and not to erratic behaviour of the simulator. Meaning that the results are consistent.

Second, although gem5 is executing the same *2.1* version of PARSEC's *blackscholes* that Graphite and Sniper are executing, its results are extremely close to the *3.0* version. So, either this similarity is due to the inherent differences in the architecture and in the simulation mode of gem5 or this indicates that version *3.0* of PARSEC has indeed some differences.

Here it is possible to observe that the execution time proportions of Sniper and Graphite with PARSEC *2.1* are extremely similar. On the other hand, Graphite with PARSEC *3.0* has a completely different behaviour. It shows a significantly higher proportion in all the tests performed.

Therefore, it can be concluded that in fact the *blackscholes* workload in PARSEC *2.1* and PARSEC *3.0* must have indeed some differences. This contradicts the information made available on the changes of the benchmark [27].

4.2.2 swaptions

Another workload that was possible to execute in all the simulators was *swaptions*. Next are Figures 6 and 7 that depict the behaviour of the simulators when executing this workload.

Unlike the afore observed workload, in this the simulated execution time of Sniper and of Graphite with PARSEC *2.1* do not line up. The results of Graphite are very close with exception of the simulation with 64 cores. On the other hand, gem5 has a much higher execution time. Nevertheless, it decreases over time. This set of result seems to indicate that the *swaptions* workload included in both *2.1* and *3.0* versions of PARSEC are the same or very similar.

The simulated time proportion matches perfectly between all three simulators, a signal of consistency of the workload across them.

The only result that steps aside from this, is the result for Graphite with PARSEC *2.1* with 64 cores, which can be attributed to experimental error of this particular instance.

4.2.3 fluidanimate

Figure 8 depicts the simulated execution time.

In line with the results obtained in the previous workloads, Sniper was the simulator with the smallest values and gem5 with the highest results. Nevertheless, it was observed across all simulators that as the number cores was increased up to 64 cores the simulated execution time decreases. As for when the number of cores reaches 132 it is observed a slight increase on the execution time on Sniper and on Graphite with both versions of PARSEC.

Unlike in the previous workloads, Graphite was not able to perform this workload of PARSEC *2.1* with two, four and eight cores, although it was possible to execute the same workload but of the *3.0* version of PARSEC. So, to analyse the obtained data two figures were created. Figure 9 depicts the behaviour up to

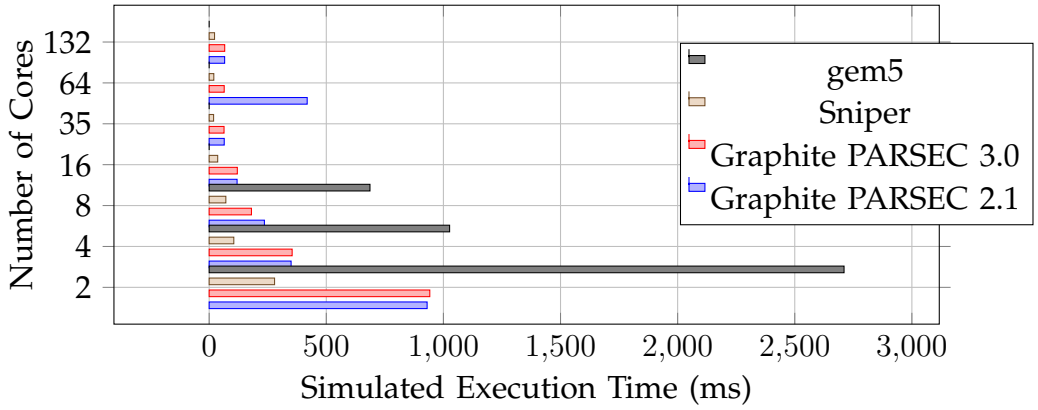

Fig. 6: swaptions simulated execution time.

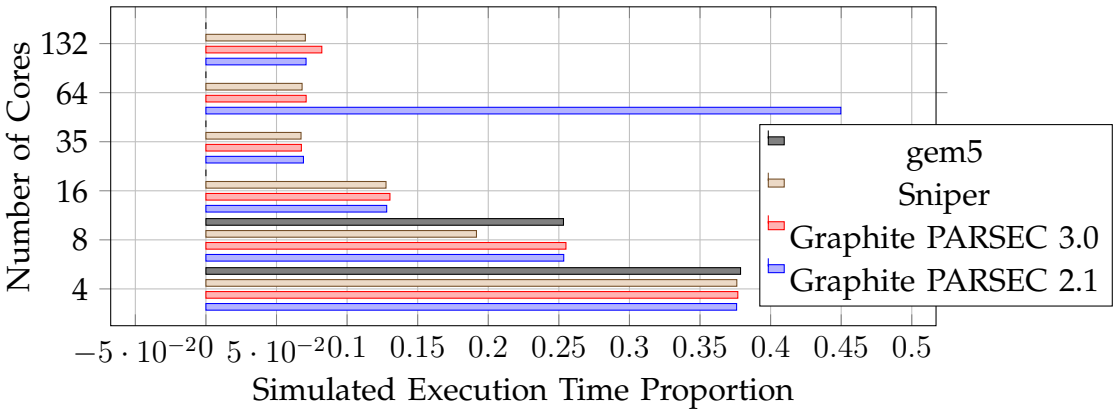

Fig. 7: swaptions simulated execution time proportion relative to two cores.

eight cores, as for Figure 10 it allows to observe the behaviour from 16 cores up to 132.

The former figure only allows to observe that Sniper and Graphite with different versions of *fluidanimate* have the same proportion. In the latter figure Sniper and Graphite with PARSEC *2.1* have very similar results, but with 132 cores they diverge more. On the other hand Graphite with PARSEC *3.0* has a much higher proportion than either Sniper or Graphite with the *2.1* version.

5 Conclusion

The chosen simulators simulate different types of CPU, interconnection networks, cache coherence mechanisms and other elements. Even when the same component is simulated its simulation is never carried out in the same way. All these aspects made it unlikely to obtain the same results when simulating the same system.

The key issue in the use of simulators is the consistency and reproducibility of results, for the same system and workload. The lack of capacity to specify the exact same configuration in all the simulators made the process of comparison difficult and less trustworthy. Nevertheless, the results complied with the expectations for each individual architecture. So, it is possible to assure some consistency, thus increasing the confidence of the results.

Another question is if this results obtained with the simulators would actually map to the results of an actual system with the simulated architecture. This is difficult to confirm. For instance, gem5, a simulator built to be very accurate, as been found to have discrepancies ranging between 5% [28] and 50% [29]. Also, Sniper has been validated against the Intel Core 2 microarchitecture [30], but differs greatly from the results obtained with gem5.

Concluding, the available simulators present many issues, and still face many issues. How-

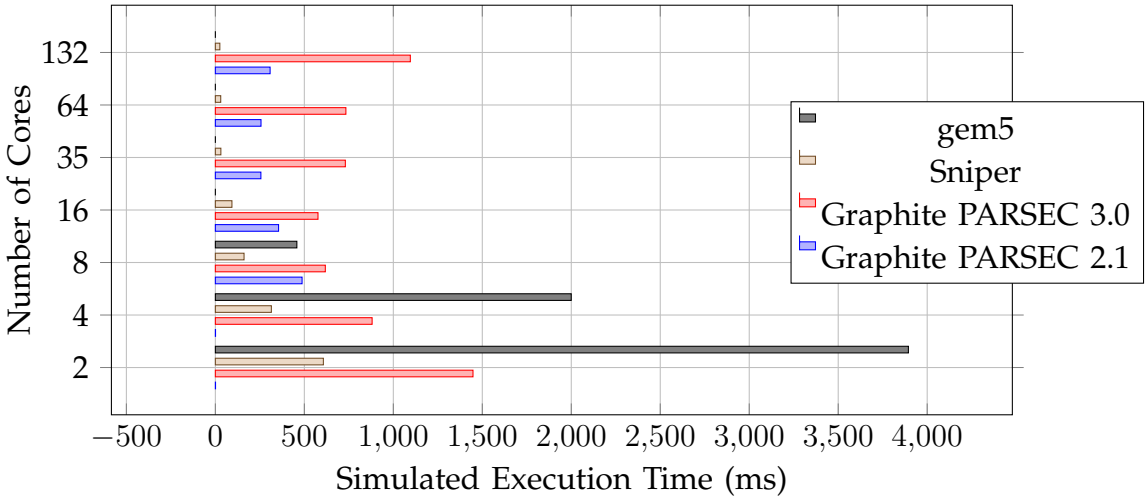

Fig. 8: fluidanimate simulated execution time.

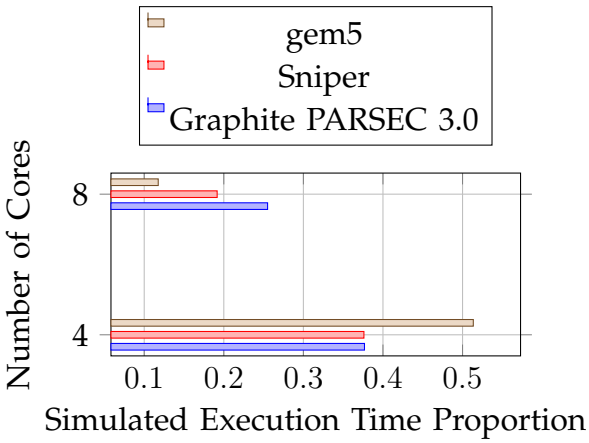

Fig. 9: fluidanimate simulated execution time proportion relative to 2 cores.

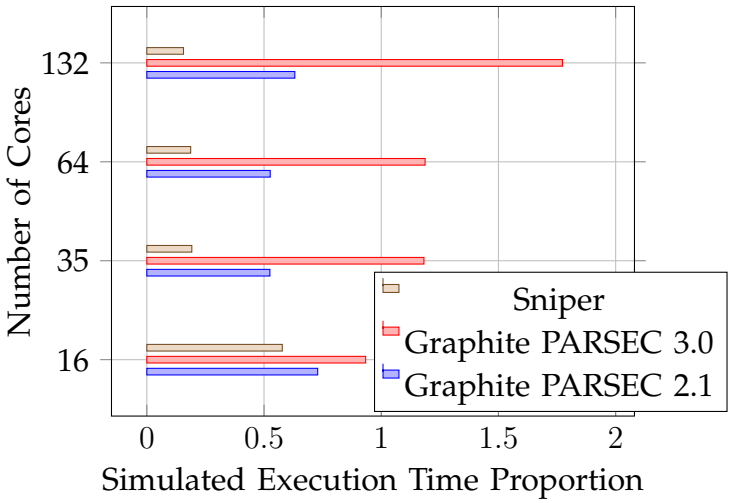

Fig. 10: fluidaniamte simulated execution time proportion relative to 8 cores.

ever, they can already provide some insights for the thousand-core era.

REFERENCES

[1] S. Borkar, "Thousand core chips: A technology perspective," in *Proceedings of the 44th Annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 746–749. [Online]. Available: http://doi.acm.org/10.1145/1278480.1278667

[2] A. H. M. Zaytoun, H. A. H. Fahmy, and K. M. F. Elsayed, "Implementation and evaluation of large interconnection routers for future many-core networks on chip," in *Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, ser. HPCC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 524–531. [Online]. Available: http://dx.doi.org/10.1109/HPCC.2012.77

[3] H. Zhao, A. Shriraman, S. Dwarkadas, and V. Srinivasan, "Spatl: Honey, i shrunk the coherence directory," in *Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 33–44. [Online]. Available: http://dx.doi.org/10.1109/PACT.2011.10

[4] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic, "Silicon-photonic clos networks for global on-chip communication," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, May 2009, pp. 124–133.

[5] V. F. Pavlidis and E. G. Friedman, "3d topologies for networks-on-chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 15, no. 10, pp. 1081–1090, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1109/TVLSI.2007.893649

[6] S.-B. Lee, S.-W. Tam, I. Pefkianakis, S. Lu, M. F. Chang, C. Guo, G. Reinman, C. Peng, M. Naik, L. Zhang, and J. Cong, "A scalable micro wireless interconnect structure for cmps," in *Proceedings of the 15th Annual International Conference on Mobile Computing*

and Networking, ser. MobiCom '09. New York, NY, USA: ACM, 2009, pp. 217–228. [Online]. Available: http://doi.acm.org/10.1145/1614320.1614345

[7] R. Singhal, "Inside intel next generation nehalem microarchitecture," *Hot Chips 20*, 2008.

[8] M. Zhang and K. Asanovic, "Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors," in *Proceedings of the 32Nd Annual International Symposium on Computer Architecture*, ser. ISCA '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 336–345. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2005.53

[9] P. Sweazey and A. J. Smith, "A class of compatible cache consistency protocols and their support by the ieee futurebus," in *Proceedings of the 13th Annual International Symposium on Computer Architecture*, ser. ISCA '86. Los Alamitos, CA, USA: IEEE Computer Society Press, 1986, pp. 414–423. [Online]. Available: http://dl.acm.org/citation.cfm?id=17407.17404

[10] F. N. Sibai, "A two-dimensional low-diameter scalable on-chip network for interconnecting thousands of cores," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 193–201, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2011.160

[11] L. Han, J. An, D. Gao, X. Fan, X. Ren, and T. Yao, "A survey on cache coherence for tiled many-core processor," in *Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on*, Aug 2012, pp. 114–118.

[12] F. D. M. Tassetto, G. Manzini. Reti di interconnessione: architetture. [Online]. Available: http://www.dsi.unive.it/\~arcb/AA96-97/Lezioni/Networks/reti_arch.html

[13] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2008.

[14] G. Kurian, J. E. Miller, J. Psota, J. Eastep, J. Liu, J. Michel, L. C. Kimerling, and A. Agarwal, "Atac: A 1000-core cache-coherent processor with on-chip optical network," in *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '10. New York, NY, USA: ACM, 2010, pp. 477–488. [Online]. Available: http://doi.acm.org/10.1145/1854273.1854332

[15] "Cmp memory modeling: How much does accuracy matter?" *MoBS-5*, 2009.

[16] C. Hughes, V. Pai, P. Ranganathan, and S. Adve, "Rsim: simulating shared-memory multiprocessors with ilp processors," *Computer*, vol. 35, no. 2, pp. 40–49, Feb 2002.

[17] M. Rosenblum, S. A. Herrod, E. Witchel, and A. Gupta, "Complete computer system simulation: The simos approach," *IEEE Parallel Distrib. Technol.*, vol. 3, no. 4, pp. 34–43, Dec. 1995. [Online]. Available: http://dx.doi.org/10.1109/88.473612

[18] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *Computer*, vol. 35, no. 2, pp. 50–58, Feb 2002.

[19] F. Bellard, "Qemu, a fast and portable dynamic translator," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 41–41. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247401

[20] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 475–486, Jun. 2013. [Online]. Available: http://doi.acm.org/10.1145/2508148.2485963

[21] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[22] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The m5 simulator: Modeling networked systems," *Micro, IEEE*, vol. 26, no. 4, pp. 52–60, July 2006.

[23] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 92–99, Nov. 2005. [Online]. Available: http://doi.acm.org/10.1145/1105734.1105747

[24] J. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "Graphite: A distributed parallel simulator for multicores," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, Jan 2010, pp. 1–12.

[25] T. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, Nov 2011, pp. 1–12.

[26] C. Bienia, S. Kumar, and K. Li, "Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors," in *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, Sept 2008, pp. 47–56.

[27] M. Authors. (2014) The parsec benchmark suite. [Online]. Available: http://parsec.cs.princeton.edu/

[28] A. Gutierrez, J. Pusdesris, R. Dreslinski, T. Mudge, C. Sudanthi, C. Emmons, M. Hayenga, and N. Paver, "Sources of error in full-system simulation," in *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*, March 2014, pp. 13–22.

[29] P. Abad, P. Prieto, L. G. Menezo, A. Colaso, V. Puente, and J.-A. Gregorio, "Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers," *Networks-on-Chip, International Symposium on*, vol. 0, pp. 99–106, 2012.

[30] M. Authors. (2014, May) The sniper multi-core simulator - sniper. [Online]. Available: http://snipersim.org/w/The_Sniper_Multi-Core_Simulator