v. 4, n. 1, 2024

Journal of
Engineering
Research

APPROACH TO LEARNING THE GEM5 SIMULATOR FOR BEGINNING RESEARCHERS

Pedro Corrêa Rigotto
Has scientific initiation scholarship - FIP PUC Minas.

Henrique Cota de Freitas
Computer Architecture and Parallel Processing Team (CArT)
Department of Computer Science – Pontifícia Universidade Católica de Minas Gerais Belo Horizonte – MG – Brazil

Abstract: The gem5 simulator is a feature-rich tool for development of research in computer architecture. It supports simulations of complete systems, the most widely used Instruction Set Architectures (ISAs) on the market and different types of system configurations. It is mainly used by researchers, but can also be used to teach computer architecture to students. In this article, a simple approach is proposed for beginners to start using the simulator in a quick and easy to understand way.

INTRODUCTION

The rapid evolution of computers demands alternatives to support design and development. From tube and transistor computer architectures to quantum computers, researchers make use of analytical models and simulations in the initial phases of design and testing. The study in undergraduate courses focused on computer architecture focuses on scalar and superscalar pipelines, which in a way, corresponds to what is most present in the processor industry. Support for multiple threads and the development of multiple cores considerably increases the complexity of studying, researching and developing architectures from the point of view, mainly, of students new to research. With a very extensive range of architecture options, it is necessary to create conditions for a scientific initiation to quickly overcome a learning barrier so that results do not continue to emerge.

Therefore, research in computer architecture requires highly accurate simulators, so that innovations can be tested quickly and reliably. As said by Akram and Sawalha [2019], the choice of which simulator to use depends largely on the purpose of the research, however there is an option that meets several different demands.

This simulator is gem5, which is an application that can simulate different architectures and configurations [Binkert et al., 2011], and which will be the simulator to be used for the purposes of this article.

The gem5 simulator enables the simulation of complete or basic systems, and this allows us to use it with the desired granularity of precision. For this article, full-system simulation will not be used, as it is not relevant for beginners to learn how to use the simulator, despite being widely used in real research [Carmo et al, 2016; Penna and Freitas, 2013; Vasconcelos et al., 2014; Mishra, 2019].

The problem addressed by this article is that despite supporting several configurations, gem5 presents a barrier for beginners, as it is so complex. The existing documentation lacks necessary updates and important information. This means that the beginner must open files present in the simulator's source code to discover the correct way to use a function. Furthermore, the step-by-step instructions available on the gem5 website for beginners[1], in addition to being very complex, are very extensive. It must only be used for reference after the user has already obtained prior knowledge, so that initial learning is quick, and specifics can be studied when necessary.

The objective of this article is to present a more didactic and easy way to start studying the simulator, so that researchers starting their research with this tool can accelerate their adaptation process, and so that it is possible to teach computer architecture to students in a simple way, without that they spend precious time learning functions that will not be used in a didactic environment.

The approach for this learning will be using example configuration scripts that already exist in the simulator's source code, instead of creating new scripts for each configuration to be tested. This way, the functionality of the simulator can be learned simply and

1 https://www.gem5.org/documentation/learning gem5/introduction/

quickly, without going into details of how to build your own objects or configuration scripts. Thus, gem5's functionality of executing different architectures, processor types, and configurations of the simulated computer can be explored, in addition to allowing the comparison of results between these simulations, using the simulator's own statistics system.

The remainder of this article will be presented as follows: In Section 2, related articles that explore computer system simulators are discussed, and where this article differs from the existing literature. The simulations performed and the steps necessary to repeat them are covered in Section 3. Section 4 presents the evaluation of the results found and the knowledge that was possible to be extracted from the statistics found from the simulations carried out. The conclusions found in Section 5 are exposed, as well as what can be explored in future work.

RELATED WORK

Computer system simulators are very important for research and learning. They provide insights that would be unobtainable in other ways. The market is full of these simulators, each with its own resources and limitations.

Nikolic et al. [2009] carried out a research in which several computer system simulators were compared in 2009. The results found show that no simulator existing at the time was able to cover all the topics of a computer architecture course. However, among those that had the best results is the M5 simulator, which was one of the two simulators that served as the basis for the development of gem5 [Binkert et al., 2011]. This indicates that gem5, as it has more capabilities than M5, is more suitable for teaching computer architecture than the other simulators explored by the authors of that article.

Computer system simulators are used to teach computer architecture and organization [Ristov et al., 2013; Garcia et al., 2009; Soares et al., 2016; Radivojevic et al., 2018; Penna and Freitas, 2013], and there are benefits to its use [Prasad et al., 2016]. Many of these simulators being used are visual and simplified. Gem5, despite not having a graphical interface, presents system simulations accurately. It can simulate complete systems, including the operating system, and obtain performance statistics that reflect the behavior of real systems. Program execution statistics on physical systems are not easy to obtain, and, by using gem5, students can have contact with these numbers that would normally be out of reach.

Although there is literature available that explores simulators, and gem5 is one of the most used in the industry [Lowe-Power et al., 2020], no article was found that aims to facilitate scientific initiation. information and its exploration. Therefore, this article proposes an approach that is more accessible and faster for setting up the simulator. Thus, students interested in computer architecture and researchers beginning in the field will be able to explore this tool in an easy and understandable way.

PERFORMED SIMULATIONS APPROACH

For the tests carried out, three different architectures and three types of processor were used. The architectures chosen were the most present on the market, being ARM, x86 and RISC-V. Among the types of processor available, the following were tested:

Simple CPU Timing, Atomic Simple CPU and *O3CPU*.

Timing Simple CPU and Atomic Simple CPUs are simple CPU types. This means that they are not very detailed during their execution, therefore, they are used in moments where a more detailed model is not necessary. They execute instructions in just one cycle, except memory access instructions, therefore there are no CPI statistics for this type of CPU.

The Timing Simple CPU is the most detailed of the simple CPUs, and its purpose is to execute instructions in the most realistic time, for greater precision in statistics. Atomic Simple CPU is a much faster simulation, with time estimates for requests to speed up the simulation. Therefore, the Atomic Simple CPU is used for fast-forwarding to a pre-defined checkpoint, or to "warm up" the caches, instead of being the workhorse of the simulation.

The other CPU used, O3CPU, is a much more detailed model of a processor. It allows the execution of instructions out of order, and tries to simulate the timing of requests as accurately as possible. It is used when the simulation demands precision in aspects such as pipeline stages.

The data collected were the total simulated time, the number of hits and the number of missions in the cache, the mission rate, and the total CPI of the simulation. The chosen system uses the default values for all aspects of the processor, which are number of CPUs equal to one, one thread, cache line size equal to 64 bytes, cache associativity equal to 2, two numbers cache levels, 64kB dcache, 32kB icache, 512MB DRAM. All these values can be changed in the simulator configuration.

The configuration file used was "se.py", available in the example configurations folder present in the simulator source code. It uses Syscall Emulation (SE) mode, which does not simulate a complete system. The simulator was developed for Linux, therefore, the operating system used for testing was Ubuntu 22.04.2 LTS.

STEP-BY-STEP GUIDE TO REPEATING THE SIMULATIONS

The first step to running these simulations is to install gem5. The developers provide a guide for this[2], however the command for installing the prerequisites present on the website in August 2023 is out of date. There is a need to change the Python installation command from python and python-dev para python3 and python3-dev, respectively. The final command is:

```
sudo a p t i n s t a l l b u i l d – e s s e n t i a
l g i t m4 s c o n s z l i b 1 g
z l i b 1 g –dev l i b p r o t o b u f –dev p r
o t o b u f – c o m p i l e r l i b p r o t o c –dev
l i b g o o g l e – p e r f t o o l s –dev python
3 –dev python 3
```

After installing the prerequisites, it is necessary to obtain the simulator code, which can be done using the command:

git clone https://github.com/gem5/gem5, executed in the folder chosen for the simulator. To compile gem5 with the desired architecture, the following commands are used in the folder where the simulator was installed:

```
s c o n s b u i l d /ARM/ gem5. o p t
s c o n s b u i l d / X86 / gem5. o p t
s c o n s b u i l d / RISCV / gem5. o p t
```

It is important to note that the name of the chosen architecture must be written in capital letters. The simulator offers other options: *Instruction Set Architectures* (ISAs), however, for this article only those presented above were used.

The program being executed by the simulated system is the

"hello" file, present in the gem5 source code in the folder

"g e m 5 / t e s t s / t e s t - p r o g s / h e l l o / bin/<arquitetura escolhida> /linux". The path differs for each test, so we must change the text:

2 https://www.gem5.org/documentation/learning gem5/part1/building/

<``*arquitetura escolhida*``> by the name of the installed architecture, using lowercase letters. This file just prints: "*Hello world!*" on the screen. Despite being simple, it is enough for the purposes of this article.

After compiling the simulator with the desired architectures, it is possible to carry out tests. To do this, the following command was used:

```
b u i l d /ARM/ gem5. o p tc o n f i g s / example / se. py –c t e s t s / t e s t – p r o g s / h e l l o / b i n / arm / l i n u x / h e l l o
––cpu – t y p e =TimingSimpleCPU –– c a c h e s
```

It is necessary to change the name of the architecture and the type of CPU to repeat the tests, keeping the capital letter, after: build/ and lowercase letter, after: /bin/. An example with other types of architecture and CPU is the following:

```
b u i l d / X86 / gem5. o p t c o n f i g s / example / se. py –c t e s t s / t e s t – p r o g s / h e l l o / b i n / x86 / l i n u x / h e l l o
––cpu – t y p e =X86O3CPU –– c a c h e s
```

This command contains important information for configuring the simulator. With it we define the architecture, the configuration file, the binary to be executed, and the type of CPU desired, in addition to the fact that the system contains caches with standard configuration. It is possible to include several other arguments in this command to define other system parameters, and obtain more information with debug flags, however, these options are more advanced, and are beyond the scope of this article. More information is available on the simulator website[3].

To analyze the results, some statistics are necessary. They can be found in the file: gem5/ m5out/stats.txt.

In the tests carried out, the values present in the lines that begin with the following names were verified:

```
s im Seconds
system. cpu. dcache. o v e r a l l H i t s: t o t a l
system. cpu. dcache. o v e r a l l M i s s e s: t o t a l
system. cpu. dcache. o v e r a l l M i s s R a t e: t o t a l system. cpu. c p i
```

If there is a need to check something that is not present in the standard statistics, it is possible to add your own values or debug flags[4].

RESULTS ASSESSMENT

With the data present in Figure 1, we can verify some characteristics of each architecture and type of CPU. As expected, Atomic Simple CPU runs much faster than the other two. The two CPUs Timing Simple CPU and Atomic Simple CPU present exactly the same number of hits and misses in the cache in all tests, and it is notable that the miss rate is much higher than the others in the O3CPU, except when the x86 architecture is used. It is also visible that there is a difference in Cycles Per Instruction (CPI) between ISAs, with the RISC-V architecture being the most efficient in this regard.

CPU Type	Architecture	Hits	Misses	*Miss rate* (%)	Time (in µs)	CPI
	ARM	1833	142	7,18	29	-
Timing	x86 RISC-V	1890	135	6,66	31	-
		1935	238	6,52	30	-
	ARM	2292	519	18,46	14	5.764
O3	x86 RISC-V	2313	200	7,95	16	5.655
		2204	527	19,29	15	5.258
	ARM	1833	142	7,18	3	-
Atomic	x86 RISC-V	1890	135	6,66	6	-
		1935	135	6,52	3	-

Figure 1. Test results with three types of CPU and architecture

3 https://www.gem5.org/documentation/learning gem5/part1/example configs/

4 https://www.gem5.org/documentation/learning gem5/part2/debugging/

The results indicate that there is knowledge to be acquired through these statistics, even when only systems with already existing and known components and ISAs are tested. This can be used by computer architecture teachers to teach their students concepts such as the importance of caches and the difference in CPI between architectures or programs. Researchers with more experience testing new technology can speed up their introduction to the simulator by carrying out the tests presented here to get used to it.

CONCLUSIONS

The gem5 simulator is a tool that promotes several benefits to students and researchers who use it. It presents several functionalities that encompass most aspects of computer architecture, however, as it has so many features, it becomes difficult to get used to and use, for beginners. This article proposes a complementary and more simplified approach for the first use of the simulator, with the aim of lowering the barrier to entry. Instead of following the step-by-step instructions provided by the simulator developers, it was proposed to use pre-assembled configurations, so that it is not necessary to delve into the specific aspects of the simulator during the first use.

The tests carried out indicate that there is knowledge to be obtained even without the construction of customized configuration files. Using the simulator this way can be useful for teaching computer architecture and organization, in addition to providing a first experience in gem5 with fewer barriers than its traditional use. Therefore, the proposed objectives were successfully achieved.

Future work may explore more complex aspects of the simulator, to facilitate the use of the more advanced resources it makes available. The simulation functionality of complete systems, the creation of new components to be simulated, and the introduction of new technologies can be explored.

THANKS

The authors thank FAPEMIG, CNPq, PUC Minas, for carrying out this research.

REFEREENCES

Ayaz Akram and Lina Sawalha. A survey of computer architecture simulation techniques and tools. *IEEE Access*, 7:78120–78145, 2019. doi: 10.1109/ACCESS.2019.2917698.

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, and et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, aug 2011. ISSN 0163-5964. doi: 10.1145/2024716.2024718. URL https://doi-org.ez93. periodicos.capes. gov.br/10.1145/2024716.2024718.

Daniel Carmo, Matheus Souza, and Henrique Freitas. Avaliac¸a˜o de topologias de redes-em-chip usando simulac¸a˜o de sistemas completos e aplicac¸o˜es paralelas. In *Anais do XVII Simpo´sio em Sistemas Computacionais de Alto Desempenho*, pages 109– 120, Porto Alegre, RS, Brasil, 2016. SBC. doi: 10.5753/wscad.2016.14252. URL https://sol.sbc.org.br/index.php/wscad/article/view/14252.

M. Isabel Garcia, Santiago Rodriguez, Antonio Perez, and Antonio Garcia. p88110:A graphical simulator for computer architecture and organization courses. *IEEE Transactions on Education*, 52(2):248–256, May 2009. ISSN 1557-9638. doi: 10.1109/TE.2008.927690.

Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, and et al.

The gem5 simulator: Version 20.0. *arXiv.org*, 2020. ISSN 2331-8422.

Debadatta Mishra. gemos: Bridging the gap between architecture and operating system in computer system education. In *Proceedings of the Workshop on computer architecture education*, WCAE'19, pages 1–8. ACM, 2019. ISBN 1450368425.

Bosko Nikolic, Zaharije Radivojevic, Jovan Djordjevic, and Veljko Milutinovic. A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization. *IEEE Transactions on Education*, 52(4):449–458, 2009. doi: 10.1109/ TE.2008.930097.

PH Penna and Henrique C Freitas. Ana´lise e avaliac¸a˜o de simuladores de sistemas com- pletos para o ensino de arquitetura de computadores. *Int. Journal of Computer Archi- tecture Education*, 2(1):13–16, 2013.

P. W. C. Prasad, Abeer Alsadoon, Azam Beg, and Anthony Chan. Using simulators for teaching computer organization and architecture. *Computer applications in engineer- ing education*, 24(2):215–224, 2016. ISSN 1061-3773.

Zaharije Radivojevic, Zarko Stanisavljevic, and Marija Punt. Configurable simulator for computer architecture and organization. *Computer applications in engineering educa- tion*, 26(5):1711–1724, 2018. ISSN 1061-3773.

Sasko Ristov, Marjan Gusev, Blagoj Atanasovski, and Nenad Anchev. Using edu- cache simulator for the computer architecture and organization course. *Interna- tional Journal of Engineering Pedagogy (iJEP)*, 3(3):pp. 47–56, Jun. 2013. doi: 10. 3991/ijep. v3i3.2784. URL https://online-journals.org/index.php/ i-jep/article/view/2784.

Jorge Fernando Maxnuck Soares, Lu´ıs Tadeu M. Raunheitte, and Takato Kurihara. The use of marie cpu simulator in computer architecture course: A case study of student's perception of learning and performance. *Journal of systemics, cybernetics and infor- matics*, 14(7):7–13, 2016. ISSN 1690-4524.

Leonardo BA Vasconcelos, Max V Machado, and Henrique C Freitas. Ambiente para estudo de computac¸ao paralela baseado no simulador completo gem5 e em algoritmos de ordenac¸ao escritos com openmp. *International Journal of Computer Architecture Education (IJCAE)*, 3(1):1–4, 2014.