EPCSAC - Extensible Platform for Cloud Scheduling Algorithm Comparison

Tiago José de Oliveira Toledo Junior[1], **Sarita Mazzini Bruschi**[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
Avenida Trabalhador São-carlense, 400, 13566-590 - São Carlos - SP

tiago.toledo@usp.br, sarita@icmc.usp.br

Abstract. *When developing a new cloud scheduling algorithm, simulation is the most used approach to test the algorithm, mainly due to the impossibility of controlling all the cloud variables and also because of the costs involved. However, setting up the simulation environment can be a difficult task and each environment can be configured its own way, resulting in no easy way of reproducing the results with other published algorithms neither comparing both under the same circumstances. To solve both of these problems, we propose the EPCSAC, an online open-source platform that allows researchers to worry only about the creation of their algorithm. They are able to create a set of simulation parameters, test their algorithm under these parameters, and compare their results with other algorithms on the platform. This way, there is an improved reproducibility and a reduced overhead of development. Researchers may have access to the platform at* `epcsac.lasdpc.icmc.usp.br/`.

1. Introduction

The development of new solutions in cloud computing faces several challenges that can decrease the rigor and reproducibility of experiments reported in scientific papers. Most of these problems rely on the non-deterministic behavior of a cloud infrastructure, in a way that is not guaranteed to the researcher to receive a similar result given the same input.

To solve this problem, most researches in the cloud computing area rely on simulation softwares that simulate, inside a single machine or in a cluster, the functionalities of a real cloud architecture [Calheiros et al. 2009]. This approach allowed researchers to propose new functionalities and strategies, among them, different scheduling algorithms, the focus of this study.

However, even with simulation environments, the comparison between two different proposed scheduling algorithms is difficult. This happens for reasons like the unavailability of the source code for the algorithm proposed by a peer, the lack of data on the inputs asserted into this algorithm and the configurations of the simulation software itself. Also, in several cases, these researchers focus on only one aspect of the algorithm, like time efficiency or energy consumption and each new research requires the researcher to implement by itself the infrastructure of the simulation, reducing the time available to the focus of the research: studying the scheduling algorithm [Tsai et al. 2017].

This leads to several papers where new scheduling algorithms proposed are compared only to algorithms developed by the same researcher or some classic algorithms from the literature, like the Round-Robin algorithm.

This reduces the potential for direct comparison between these algorithms since one cannot truly know if some algorithm performs better than the other, and if so, how and why does that happen.

To solve this problem and make the research process easier for the researchers, we propose the EPCSAC (Extensible Platform for Cloud Scheduling Algorithm Comparison), a web service based on the SaaS model that provides the simulation infrastructure required for researchers to test their algorithms.

With this platform users will be able to focus only on developing their algorithm, and then easily compare the proposed algorithm with other ones previously implemented, reducing time overhead for development and improving both the reproducibility of their experiments and the quality of research.

This paper is structured as follows: Section 2 discusses the research developed to create the EPCSAC, Section 3 describes the system and its development process, Section 4 presents the results obtained after the development and Section 5 concludes the paper and indicates next steps to improve the platform.

2. Literature Review

Several new cloud scheduling algorithms were proposed, with different metrics of comparison. [Elhady and Tawfeek 2015] have used makespan as a metric for the performance of the algorithms. They varied the number of tasks and compared genetic algorithms, such as ABC (Artificial Bee Colony), PSO (Particle Swarm Optimization) and ACO (Ant Colony Optimizations) with some standard ones like FCFS (First-Come First-Served) and a Random Scheduler. The authors concluded that the ABC stands out as the better algorithm in terms of makespan.

[Shu et al. 2014] used makespan, response time and energy consumption as performance metrics. The number of tasks and the parameters of genetic algorithms were the non-fixed parameters used when testing their algorithms. After the simulations, they concluded that their proposal offers significant improvement in terms of average makespan and energy consumption.

[Sonkar and Kharat 2016] cited as metrics of performance: throughput, overhead, migration time, response time and scalability when dealing with resource allocation. They defended that an effective resource allocation, to maximize the server and resource utilization, is essential for achieving user satisfaction in cloud computing.

[Duan et al. 2018] used makespan as a performance metric, and varied the number of tasks and genetic algorithms features like mutation and crossover. The comparison was made using AIGA (Adaptative Incremental Genetic Algorithm), Max-Min, Min-Min, SGA (Standard Genetic Algorithm), ABC and SA (Simulated Annealing). They concluded that the AIGA performs better than the other algorithms and also converges faster than the other genetic options tested.

Research was also developed on the area that tries to improve the research workflow for researchers. The state of the art platform on this area, named Pewss, was proposed by [Tsai et al. 2017] as a platform to provide the simulation infrastructure required to test workflow scheduling algorithms and to improve the research process.

Pewss uses a client/server approach to develop its infrastructure. The client side implements two main components: the Simulation UI and the Editor. The Simulation UI allows the researcher to set up the configuration, such as parameter values and input workloads. The editor allows new algorithms to be inserted and tested in Pewss. The server side has a user authentication system and a file manager that keeps track of all algorithms. A database controller is responsible for the communication with the database that holds information about the users and the conducted simulations.

3. Project Development

Inspired by the development made by [Tsai et al. 2017], the EPCSAC is a web platform that provides a transparent simulation environment to the user, that needs to worry only about developing his scheduling algorithm.

When running a simulation on EPCSAC, all the data required to reproduce the simulation is stored on the database. This way, the user can be confident that results will be reproducible between different researches.

3.1. Architecture

The EPCSAC is composed of 6 modules. Figure 1 presents those modules and their connections.

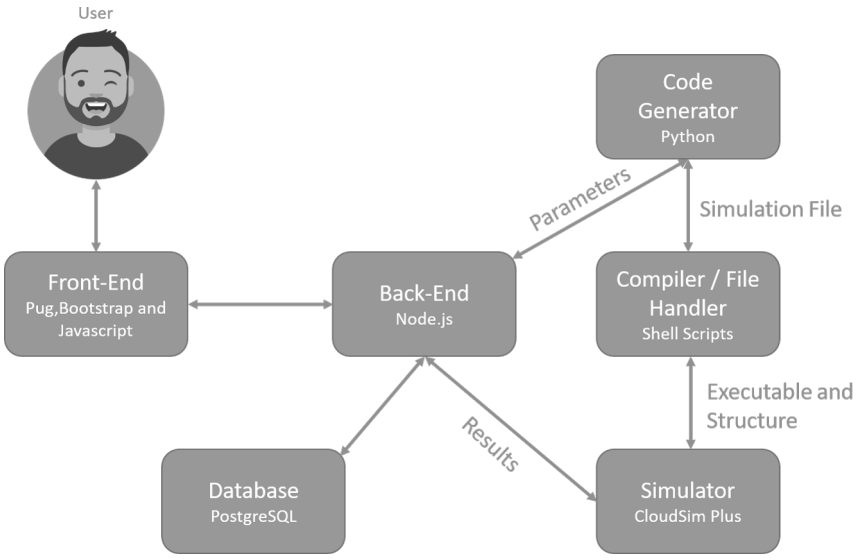

Figure 1. EPCSAC Architecture

Front-End: The Front-End was developed using HTML, CSS, Javascript, Bootstrap and Pug. It has a responsive layout allowing the website to be accessed from mobile devices. All inputs from the user are made by HTML forms.

Back-End: Developed using a Node.js server, this module is for the connection to the database, and the page routing.

Database: A PostgreSQL database where all the historical and user-related data is stored.

Simulator	Available	Update	Documentation	Usage	Functions	Expansion
Green Cloud	No	NA	NA	NA	NA	NA
CloudSched	Yes	4+ Years	1	1	2	1
iCanCloud	No	4+ Years	NA	NA	NA	NA
CloudSim	Yes	Frequent	3	5	5	3
CloudSim Plus	Yes	Frequent	5	4	5	5

Table 1. Simulator comparison summary

Code Generator: This module generates the simulation code which requires the name of the class of the algorithm to be simulated and the parameters of the simulator. The code is generated by a shell script that receives the data and passes it to a Python script that creates the file.

Compiler / File Handler: When a new simulation is requested, several steps must happen: the algorithm file has to be copied from the user folder to the simulator folder, the simulation file that was generated must be moved to the right folder and then the simulation must run. This module takes care of these steps.

Simulator: The simulator has its own folder inside the project and is used, by a shell script, to run the simulations requested by the user.

3.2. Simulator Selection

To select the simulator to be used on the platform, a comparison framework was required. The first step, therefore, was to develop suitable metrics for comparison. The following criteria were used:

- **Availability** - Is the simulator available to be downloaded? In case of a negative answer, it is not viable to be used.
- **Update Frequency** - How often the simulator is updated and improved? If it does not receive frequent updates, it should not be used.
- **Documentation** - Is there documentation available? The better the documentation, the easier will be for researchers to develop their algorithms.
- **Usage by the community** - As the scheduling algorithm should be implemented in a way to work on the simulator, it has to be a simulator that the community knows how to use and accepts.
- **Functionalities** - Does the simulator implement a handful amount of functionalities? If the simulator is too basic, it may not achieve the necessities of the researcher.
- **Ability to expand** - Is it possible to expand the simulator and add more features to it?

All these criteria were summarized on Table 1 and the CloudSim Plus was the selected simulator. The numerical values range from 1 (worst) to 5 (better).

3.3. Database Design

Four main entities have their own tables on the database:

- **Researchers**: Information about the users of the platform, such as name, e-mail address and encrypted password are held in this table.

- **Parameters**: The parameters of the simulation, such as the number of physical machines and characteristics of each one such as RAM are stored on this table.
- **Algorithms**: Name, version and the path to the location of the algorithm on the server are kept on this table.
- **Simulations**: An aggregation that keeps, for each simulation, which parameters, algorithms and researchers were involved, promoting reproducibility.

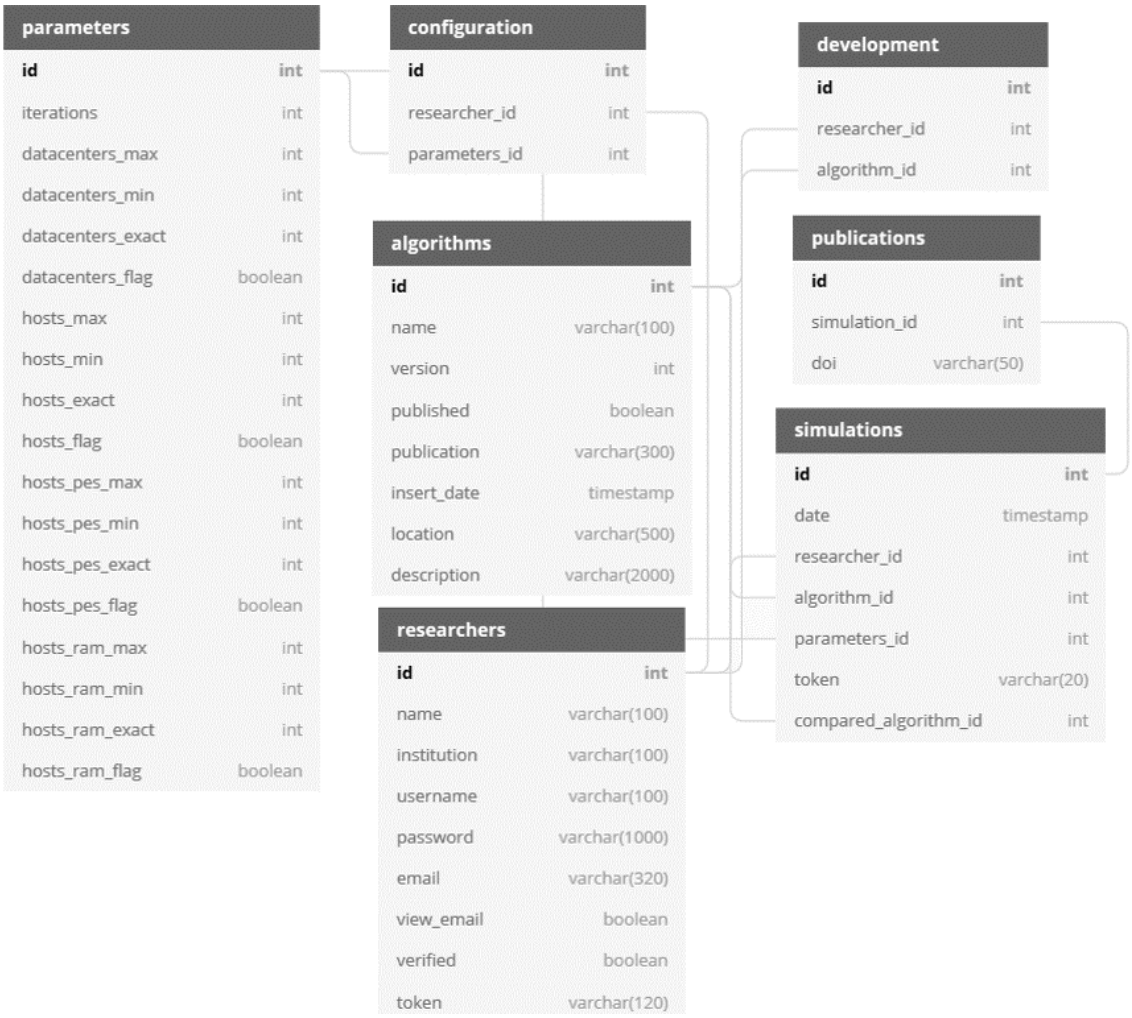

Figure 2. EPCSAC Database Simplified Entity-Relationship Diagram.

The full entity-relationship diagram can be found on Figure 2. The parameters table was truncated because of its size.

3.4. Website Development

The website was made with the following steps:

- User creation and authentication
- Parameters creation
- Algorithm upload
- Algorithm simulation
- Retrieval of simulation results

- Algorithm comparison

Each of these steps had a design choice that was relevant to the development of the project as a whole.

The user creation and authentication were done because it is possible to have a multi-user platform that allows non-volatile storage of the platform usage. While it may be obvious at this point, a platform could indeed be developed without an user creation system.

The algorithm upload brought the following question: how should the files be stored in the platform? Two possibilities were found: insert the algorithm file directly on the database as a binary stream, or keep the file on the server file system and store only the algorithm file path on the database. The later was chosen because it allowed a better organization of the algorithms, having a separate folder for each user and all files related to that user inside his folder, and to reduce overhead on the database.

For the algorithm simulation and the retrieval of simulation results, one issue was found: the simulation can take a very long time to run. In this case, should the user be locked on the screen while the simulation was running? Simulations that take longer, days even, would be unreliable as the user may close the window. The solution proposed was to send an e-mail to the user with a link to a results dashboard when the simulation finishes, allowing the user to keep navigating on the website.

Finally, during the development of the comparison between two algorithms, the question was: how this should be done? To reuse the code that was developed on the single algorithm simulation, it was decided that the comparison would be made with two simulation runs: the user algorithm and the published algorithm. After these runs, the user receives the results from both.

3.5. Simulation Workflow

The simulation starts when the user clicks the form button on the simulation page. From this point, the simulation module retrieves the parameters and the algorithm to be simulated and the simulation file is generated.

Next, it copies the algorithm file to the CloudSim Plus path where it needs to be, inside the schedulers package, for the program to compile.

Finally, a call is made to execute the script named *run_simulation.sh*. This script will build the CloudSim Plus using Maven, to generate the .jar file needed for the simulation to run. Then, this script will also use the built-in script of CloudSim Plus that runs an example from the list of examples.

To use this script, the simulation file is copied to the example folder inside the cloudsimplus.examples.schedulers package. With this, the shell script that was implemented by [Campos da Silva Filho et al. 2017] can be reused.

Once the compilation and simulation are finished, an e-mail is sent to the user, and the files that were copied to the folders are deleted to not clutter the simulation folder.

4. Results

As results for the development of the project, we have a full functional system with an web interface, capable of running simulations and of presenting the results of the simulation for the user.

The user receives two types of results: a standardized .csv file for each algorithm, containing the results of the execution of each task, and an online dashboard with comparative analysis between the algorithms in terms of average makespan, success rate and average response time, which can be seen in Figure 3.

To illustrate a simulation result, Table 2 presents the comparison between 3 scheduling algorithms and their results obtained from the platform.

Table 2. Obtained results after the simulation

Algorithm	Success Rate	Average Makespan (s)	Average Response Time (s)
Fair Scheduler	100%	1.03	0.64
Round Robin	100%	1.84	0
Batch Process	100%	1	0.55

Some inference can be made upon those results, for example: why the Round-Robin algorithm has the highest average makespan while having the lowest response time? Well, this can be answered because of the preemptive approach of this algorithm. The quantum value defined for it, most probably, is very low, making the processes switch frequently inside the VMs. This way, the response time is zero, because all tasks started on the first second of the simulation, however, as they keep switching context, the overall time required to finish all tasks is higher.

This is interesting because shows the potential of this algorithm to be used on systems where the interaction with the user is important and the response time has more relevance than the overall time required to finish the tasks.

Another analysis that can be made is to notice that the Batch Process outperforms the Completely Fair Scheduler on both, average makespan and average response time. If these are the only features of interest for the researcher, then the Batch Process can be chosen over the Completely Fair Scheduler.

This shows how easy it is to perform a simulation and start coming up with hypothesis to be answered during the research process. Also, as all of this data is saved and the simulation has deterministic characteristics for non-random inputs, it can be retrieved and simulated again. Therefore we have ease of development and improved reproducibility.

5. Conclusion and Future Work

As an user-friendly platform that allows researchers to propose their own improvements to the system, the EPCSAC can become a standard tool in the process of researching and developing new cloud scheduling algorithms.

With the simulation analysis, we show that in fact the EPCSAC can reduce the time consuming aspect of developing a simulation environment and improve the way the

results are reproduced. The results returned to the researcher by the platform allow for quick and easy hypothesis generation for the research process, making it easier. This way, the EPCSAC attends the requirements to be an useful tool for the researching process.

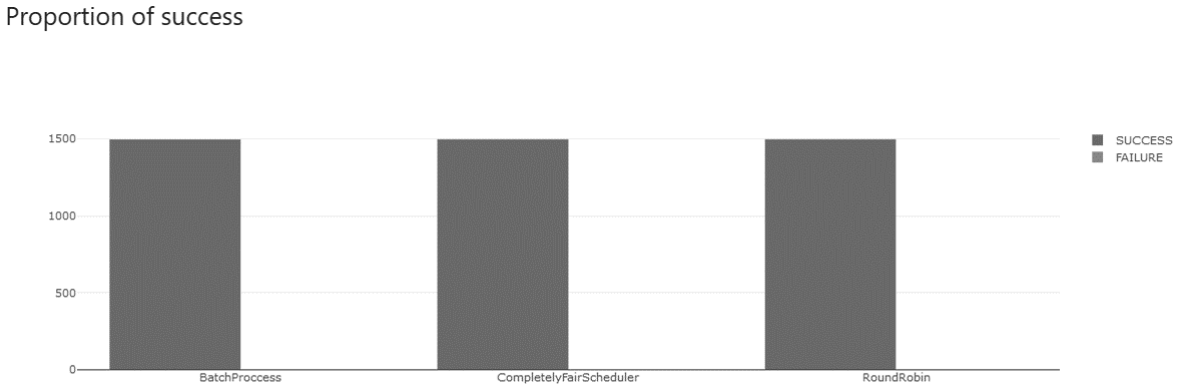

Figure 3. First Chart of EPCSAC Dashboard

Several next steps can be asserted. However, as the top priorities, the next steps of development will be:

- Allowing researchers to create their personal pages, and the creation of pages for algorithms, so others can find and choose the algorithms to compare.
- Creating more options of metrics, such as power consumption.
- Improving the quality of the software introducing unit and integration tests.

References

Calheiros, R., Ranjan, R., De Rose, C., and Buyya, R. (2009). Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services.

Campos da Silva Filho, M., Oliveira, R., Monteiro, C., Inácio, P., and Freire, M. (2017). Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. pages 400–406.

Duan, K., Fong, S., Siu, W., Song, W., and Guan, S.-U. (2018). Adaptive incremental genetic algorithm for task scheduling in cloud environments. *Symmetry*, 10:168.

Elhady, G. and Tawfeek, M. (2015). A comparative study into swarm intelligence algorithms for dynamic tasks scheduling in cloud computing. pages 362–369.

Shu, W., Wang, W., and Wang, Y. (2014). A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2014.

Sonkar, S. and Kharat, M. (2016). A review on resource allocation and vm scheduling techniques and a model for efficient resource management in cloud computing environment. pages 1–7.

Tsai, M.-H., Lai, K.-C., Chang, H.-Y., Chen, K., and Huang, K.-C. (2017). Pewss: A platform of extensible workflow simulation service for workflow scheduling research. *Software: Practice and Experience*, 48.