A job shaping strategy to accomodate workload traces under varying resource management policies

João Pedro M. N. dos Santos, Antônio Tadeu A. Gomes

[1]Laboratório Nacional de Computação Cientıífica (LNCC)
Petrópolis – RJ – Brasil

macleure@posgrad.lncc.br, atagomes@lncc.br

Abstract. *Supercomputers play a pivotal role in advancing research and development across diverse scientific and engineering domains. However, configuring resource management in these systems to ensure maximum productivity and cost-effectiveness is a challenge. Workload simulation emerges as a crucial tool in this context, offering a mechanism to explore resource management configurations in the presence of expected user behaviors. This paper focuses on a specific requirement for simulation-based optimization applied to tuning resource management configurations: the need for simulators that are both precise and efficient. This paper introduces a job shaping strategy to accommodate real workload traces under varying resource management policies in discrete-event RMS simulations. Our findings from evaluating the proposed strategy on a real-world case study suggest that job shaping allows effectively capturing changes in system behavior, regardless of whether some of the real workload traces used as input to the simulation are incompatible with the simulated policies.*

1. Introduction

High-Performance Computing (HPC) systems have become indispensable tools in a myriad of scientific and engineering disciplines. They enable researchers to conduct detailed numerical simulations and analyze vast amounts of data. As the demands for more sophisticated analyses and simulations grow, the efficient utilization of HPC systems becomes critical. Tuning the configuration of HPC systems aiming at optimal utilization is a challenging task. The focus of this paper is on the configuration of a key component in shared HPC systems: the resource management subsystem (RMS). Configuring an RMS includes the definition of policies for queue management, job prioritization, and exploration of backfill strategies.

One possible approach to tuning the configuration of an RMS is through *simulation-based optimization*, in which an objective function (e.g. minimize queue waiting time or maximize system utilization) is searched via simulations in which variations in policy are stochastic [Fu 2014]. A key point in the optimization via simulation is that the simulator invoked in each step of the optimization process be both *precise* and *fast*. In the case of tuning the configuration of an RMS, the simulators must be both: (i) faithful to the parameterization of the policy configuration; and (ii) fast enough to allow the solution to the optimization problem to be achievable in a time that is not prohibitive.

Many pieces of work have developed RMS simulators with various levels of support for queuing, prioritization and constraining, allowing different levels of trustworthiness in the modeling of policy configurations [Dutot et al. 2017,

Rodrigo et al. 2018, Jokanovic et al. 2018, Galleguillos et al. 2020, Klusáček et al. 2020, Simakov et al. 2022]. Fidelity in representing *user behavior* in RMS simulators should be as important as representing *system behavior* though. Using real workload traces poses difficulties in this context because variations in policy during an optimization process may lead to valid jobs in the workload traces being incompatible with the newly generated policy. Disregarding these jobs is not an option as it changes the actual workload imposed on the HPC system.

In this paper, we present a simple strategy for accommodating real workload traces to incompatible resource management policies in discrete-event RMS simulations. The strategy—so-called job *shaping*—is inspired by the traffic management technique of the same name used in computer networks to bring their packet flows into compliance with desired traffic profiles. In our case of RMS simulations, the shaping of a job occurs by means of a transformation of its *geometry*, i.e. the product of the requested CPU cores and the time limit (corresponding to the estimated wall-clock time) for the job. The idea is to keep the actual value of this product—and therefore the actual computational demand of the job in terms of CPU hours—unchanged, while changing the ratio of its factors.

To evaluate the job shaping strategy, we developed an RMS simulator and used as a case study the Santos Dumont supercomputer at LNCC in Brazil,[1] analyzing its workload traces from mid-2017 to mid-2019, in the midst of which (mid-2018) a policy change was carried out with the aim of reducing the long waiting times the users experienced then. Our main findings with this case study are that:

1. job shaping allows our RMS simulations to capture the change in the system behavior even when using workload traces recorded in the period before the actual policy change;
2. job shaping works even when exploring limited changes in the ratio of the geometry factors of the affected jobs; and
3. the queue where the shaped job is targeted at matters, even if the queues that are potential targets share a same resource pool, because of the geometry constraints each queue imposes.

We organized the remainder of this paper in the following way: Section 2 presents some related work; Section 3 describes the job shaping strategy; Section 4 shows the experimental methodology used for evaluating the proposed strategy; Section 5 presents the results of the proposed strategy applied to real data injected in simulated resource management policies; Section 6 presents some concluding remarks and a few topics for future work.

2. Related work

Many pieces of work have developed RMS simulators, including: (i) discrete-event simulators, either built from scratch [Galleguillos et al. 2020] or based on simulation frameworks such as GridSim[2], SimGrid [Casanova et al. 2014] and Batsim [Dutot et al. 2017], and (ii) time-accelerated simulators, which extend existing RMSs such as Slurm[3] [Rodrigo et al. 2018, Jokanovic et al. 2018, Simakov et al. 2022].

[1] `http://sdumont.lncc.br`
[2] `http://www.gridbus.org/gridsim`
[3] `https://slurm.schedmd.com`

Discrete-event RMS simulators are bound to be less reliable because their behavior is a necessary simplification of a real RMS, but are also generally less computationally expensive and therefore more likely to be plugged into an optimization process.

Recent developments in discrete-event RMS simulators support multiple queues with varying priorities and constraints, allowing complex policy configurations to be modeled more faithfully [Klusáček et al. 2020]. This fidelity in representing *system behavior* is an important part in an optimization process, but not the only one; capturing *user behavior* is also crucial. All the RMS simulators we identified in the literature allow the specification of workloads—either synthetic ones or real ones, the latter usually described as workload traces in the Standard Workload Format (SWF) [Chapin et al. 1999]. The problem with using real workload traces is that variations in policy during an optimization process may lead to valid jobs in the workload traces being incompatible with the newly generated policy. As far as we could explore the literature, there is no other piece of work that proposes an approach as ours to accommodate these "uncompliant" jobs.

Job shaping closely resembles the concept of *moldable jobs* [Cirne and Berman 2001]. There is an extensive body of research on moldable jobs focusing on novel scheduling algorithms [Sabin et al. 2007, Gupta et al. 2014, Prabhakaran et al. 2015, Posner et al. 2024]. Our concept of job shaping explores moldability only to jobs in a workload trace that do not fit within new policies, to accurately evaluate the system's behavior under different configurations while preserving the overall workload structure.

3. Job shaping

Job shaping is a strategy for bringing jobs from real workload traces into compliance with policies being tested as part of an optimization of an RMS configuration.

Consider the two different scenarios of Figure 1. Figure 1(a) shows an RMS configured with two different queues:

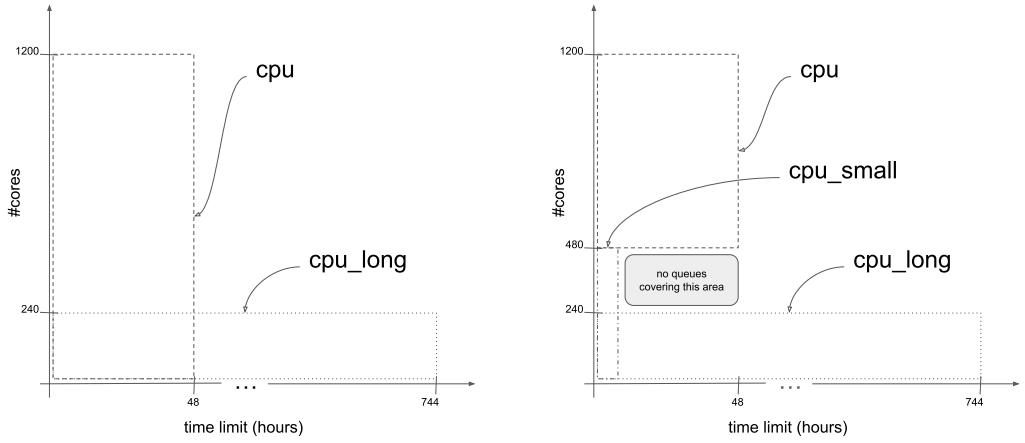
- `cpu`, which allows jobs allocating from 1 to $1,200$ CPU cores with a maximum time limit of 48 hours; and
- `cpu_long`, which allows jobs allocating from 1 to 240 CPU cores with a maximum time limit of 744 hours (31 days).

Figure 1(b) shows the result of a change in the policy adopted in Figure 1(a) with a modification of the minimum amount of allocatable CPU cores for the queue `cpu` to 480, as well as the creation of a third queue designed for jobs with small geometries:

- `cpu_small`, which allows jobs allocating from 1 to 480 CPU cores with a maximum time limit of 2 hours.

This change—which actually happened in a real-world case, as described in Section 4.1—resulted in a region of possible geometries unreachable by jobs. Therefore, those jobs in the workload traces that covered the unreachable region could not be used to evaluate the change in policy illustrated in Figure 1(b).

The shaping of a job occurs by means of a change in the ratio of its geometry factors, while keeping the actual value of the geometry unchanged. Figure 2 illustrates the idea, using the configuration depicted in Figure 1(b).



(a) Queues before change

(b) Queues after change

Figure 1. Example of a geometry area uncovered by a change in policy.

Mind that different shaping alternatives may be available, including the possibility of different *target queues* for the shaped jobs. Taking Figure 2 as an example, a job (represented by a black thin line) originally requested 360 cores and had an estimate of 48 hours for its wall-clock time. This was a possible configuration for a job in the previous `cpu` queue configuration (Figure 1(a)), but not in its new configuration (Figure 1(b)). The shaping of this job can produce different alternatives on either the (reconfigured) `cpu` queue or the (original) `cpu_long` queue. In an optimization process these alternatives should be taken into account, e.g. as part of a stochastic variation in the simulation inputs.

Presently, the change in this ratio considers that all jobs have linear speed-up, i.e. if we multiply the amount of available cores for a job by a factor of X, the estimated wall-clock time for this job is divided by X. The simulation results presented in Section 5 demonstrate that this is a fairly reasonable assumption in the average.

4. Experimental Methodology

4.1. About the Santos Dumont supercomputer

The Santos Dumont supercomputer comprises two HPC clusters, both of them with hybrid configurations including thin nodes, fat nodes, and nodes with GPUs. All of these nodes are managed by a single RMS (Slurm). The 1st HPC cluster started operating in 2015 and its thin nodes (total of 504) and nodes with GPUs (total of 198) have the following configuration: $2\times$ CPU Intel Xeon Ivy Bridge (12 cores each CPU) and 64GB RAM. The 2nd HPC cluster started operation in 2019 and its thin nodes (total of 246) and nodes with GPUs (total of 94) have the following configuration: $2\times$ CPU Intel Xeon Cascade Lake Gold (24 cores each CPU) and 384GB RAM. In this paper we focus on the usage of the thin nodes of the 1st HPC cluster between 2017 and 2019. The main reason for this choice is that during this time frame there was only a single change in the resource management policy, in 28th May 2018, which gives us a much larger sample of workload traces than in any other time frame throughout the operation of the Santos Dumont supercomputer.

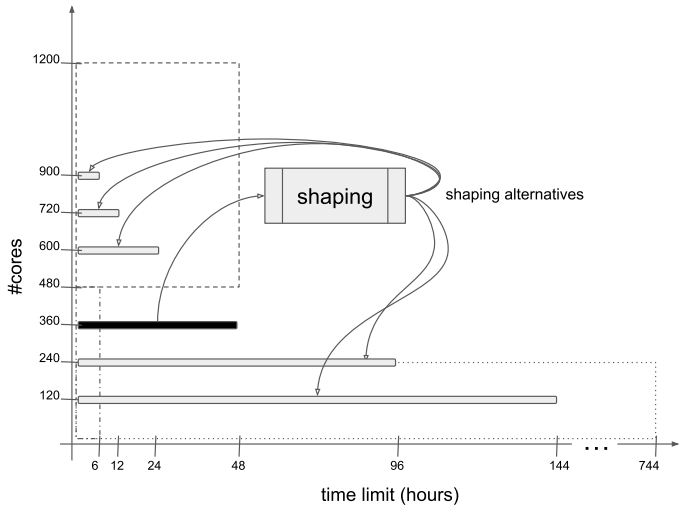


Figure 2. Examples of job shaping.

4.2. Data collection

The data used for this study were collected from the workload traces of the Santos Dumont supercomputer in the period between mid-2017 and mid-2019. Knowing that there was a change in the resource management policy on 28th May 2018, the data was divided into two periods of equivalent length, considering submissions prior to the policy change, in the period between 15th June 2017 and 15th May 2018—called the "before-change" (BC) period—, and after the policy change, in the period between 15th June 2018 and 15th May 2019—called the "after-change" (AC) period. The jobs submitted between 16th May 2018 and 14th June 2018 were disregarded to avoid the eventual effect of transient behavior just before and after the change in the policy.

Only job submissions to queues with thin nodes (`cpu`, `cpu_long`, `cpu_scal`, and `cpu_small`, this last one created in the AC period) were considered, disregarding the queues used for user training and development and system testing. Considering the whole period, a total of $68,845$ job records were initially considered in this study. In the BC period, there are $30,586$ (44%) job records, and in the AC period, $38,259$ jobs records (56%). Each period is composed by an interval of 334 days.

4.3. Simulation tool

We used SimPy[4] as our discrete event simulation engine. SimPy provides a framework for modeling complex processes and interactions. It is particularly well-suited for simulating scenarios such as resource allocation, queuing systems, and scheduling policies.

Our RMS simulator was designed based on entities that we have implemented over SimPy to compose a typical HPC system:[5] **Supercomputer**, **Partition**, **Account**, **User** and **Job**. Figure 3 shows a domain model of our RMS simulator.

[4] `https://simpy.readthedocs.io/`

[5] The code is available at `https://gitlab.com/itdf/hpc_sim`

The Supercomputer entity represents the source of resources that users need, i.e. nodes for job processing. Those nodes can be organized in different partitions—the equivalent of a queue—, each one with specific rules for resource requesting, and can be part of the same or of different resource pools. In the version used for this work we only consider thin nodes and disregard heterogeneity in resource capabilities.

The User entity represents the source of jobs. Each user in the simulator is associated with a job generator, which can be based on: (i) known probability distributions (Poisson, Exponential, etc.); (ii) real workload data, as is the case of our data described in Section 4.2; and (iii) histogram sampling, so that any unknown probability distribution may be approximated provided that we have sufficient data sampled from it. Users can also be organized into accounts, allowing priority levels based on projects, for example.

The Simulation entity organizes the interactions between the user job generation process and the supercomputer job execution process, in addition to capturing system measurements. The configurable parameters for job priority calculation were inspired by the Slurm Multifactor Priority Plugin,[6] and in the version used for this paper included the job age, job size, partition priority and quality of service (QoS) factors.

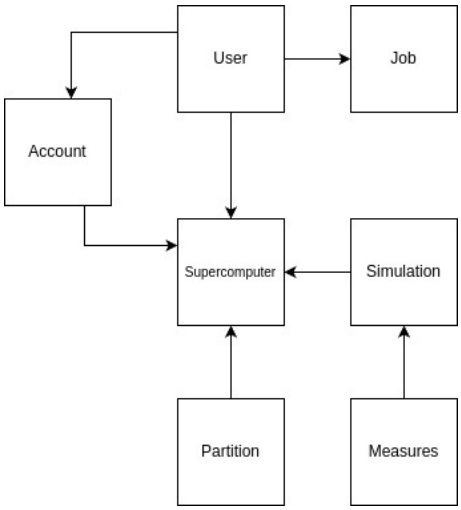


Figure 3. UML domain model of the simulator.

4.4. Analysis of real and simulated data

To evaluate the quality of our simulations, statistics regarding queue waiting time (QWT), requested nodes, requested time limit, elapsed time and inter-arrival time were compared between simulated results and real data.

We first compared the expected results of a theoretical model and the values obtained through simulation for a configuration equivalent to the compared model. Then, we observed the behavior of the system from real data and compared to the behavior of the simulation with a configuration equivalent to the real system.

Once the simulation is compatible with the expected behavior, the "what-if" scenarios explored in an optimization process can be simulated. When performing this type

[6] `https://slurm.schedmd.com/priority_multifactor.html`

of scenario, one must evaluate whether the processed workload is compatible with the simulated resource management policy. In other words, if the simulation configuration is restrictive, there will be jobs in the workload that will not be processed, impacting the result observed in the simulation. As an alternative to dealing with jobs that do not fit into simulated scenarios, jobs are shaped, as described in Section 3.

The relationship between statistics from the BC period compared to the AC period was also observed, evaluating whether the behavior of the real data was captured by the simulator. As presented in [Gomes 2018], the policy change in the middle of the observed period resulted in a reduction of the 95-percentile of the QWT from ≈ 54 hours to ≈ 18 hours. However, when evaluating the ECDFs of the QWT we observed that the policy change affected jobs with short and long waiting times differently. More specifically, in the second quartile (Q2) the ECDFs of the BC (ECDF_{1st}) and AC (ECDF_{2nd}) periods crossed at point $2,920$, indicating that jobs with QWT up to that point had an increase in QWT, whereas the jobs with QWT longer than $2,920$ secs benefited from a decreased QWT after this change, as shown in Figure 4.

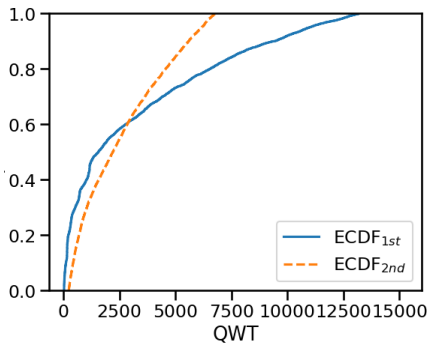


Figure 4. Q2 of the QWT in the BC (ECDF_{1st}) and AC (ECDF_{2nd}) periods.

5. Results

5.1. Theoretical validation

Queuing theory models play a crucial role in validating simulation results. By providing a mathematical framework to analyze and predict system behavior, queuing theory offers a benchmark against which simulation outcomes can be measured. These models help ensure that simulations accurately reflect real-world dynamics, such as waiting times, resource utilization, and system throughput.

Based on M/M/c queue models, we defined six different simulations varying inter-arrival rate (λ), service time (μ) and number of servers (c). The simulation was configured with a single supercomputer and partition with equivalent number of servers. For priority calculation only the job age parameter was set, thus rendering a classical First-Come-First-Served queue discipline. The user was configured with a job generator associated with a Poisson process for inter-arrival time, with an Exponential process for the requested runtime, and with 1 as a constant number for requested nodes. For comparing results, as the simulation was based on a random generator, we executed it 30 times, calculating a confidence interval (95%) for the obtained QWT average. Results are shown on Table 1 for each M/M/c variation.

Table 1. M/M/c validations on expected average QWT.

	λ=0.5 / μ=1		λ=0.7 / μ=1		λ=0.9 / μ=1	
	Expected	**Observed**	**Expected**	**Observed**	**Expected**	**Observed**
M/M/1	1.00	0.96 – 1.02	2.33	2.25 – 2.44	9.00	8.20 – 9.80
M/M/2	0.06	0.07 – 0.07	0.14	0.13 – 0.14	0.25	0.25 – 0.27

5.2. Real workload processing validation

Comparing real data analysis with simulated results is a critical step in validating the accuracy and reliability of computational models. That said, we carried out a simulation configured with approximated parameters for job priority calculation in the BC and AC periods with their respective workloads. As observed in Tables 2 and 3, the statistics about inputs show that it had no changes, as expected. Both workload periods were simulated in ≈ 10 minutes in a single-core python kernel.

Table 2. BC period descriptive statistics comparison.

	Req. nodes		**Req. timelimit**		**Elapsed time**		**Inter-arrival time**	
	real	**simulated**	**real**	**simulated**	**real**	**simulated**	**real**	**simulated**
Avg.	10	10	108,705	108,702	38,118	38,116	942	942
Std. dev.	16	16	79,539	79,539	115,033	115,029	4,447	4,446
Min	1	1	60	60	0	0	0	0
1Q	1	1	14,400	14,400	92	92	1	1
2Q	4	4	172,800	172,800	2,566	2,566	59	59
3Q	10	10	172,800	172,800	31,583	31,580	420	420
Max	128	128	1,314,000	1,314,000	2,678,429	2,678,429	434,656	434,656

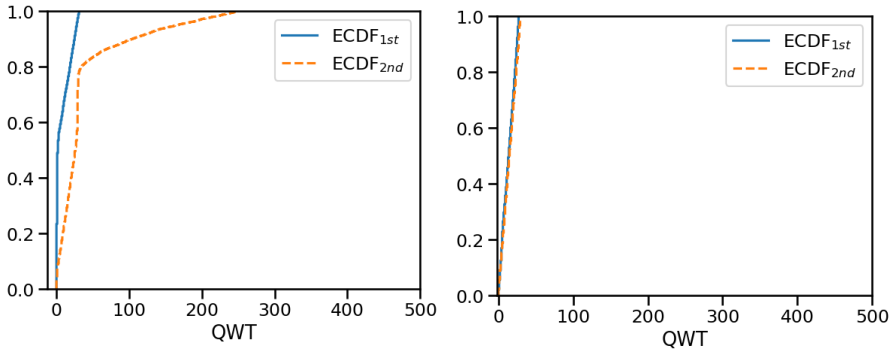
Table 3. AC period descriptive statistics comparison.

	Req. nodes		**Req. timelimit**		**Elapsed time**		**Inter-arrival time**	
	real	**simulated**	**real**	**simulated**	**real**	**simulated**	**real**	**simulated**
Avg.	10	10	36,301	36,300	24,378	24,377	754	754
Std. dev.	12	12	62,573	62,572	141,547	141,543	4,259	4,259
Min	1	1	60	60	0	0	0	0
1Q	1	1	5,400	5,400	58	58	0	0
2Q	4	4	7,200	7,200	884	884	32	32
3Q	16	16	32,400	32,400	6,000	6,000	461	461
Max	240	240	756,000	756,000	2,678,422	2,678,422	516,197	516,197

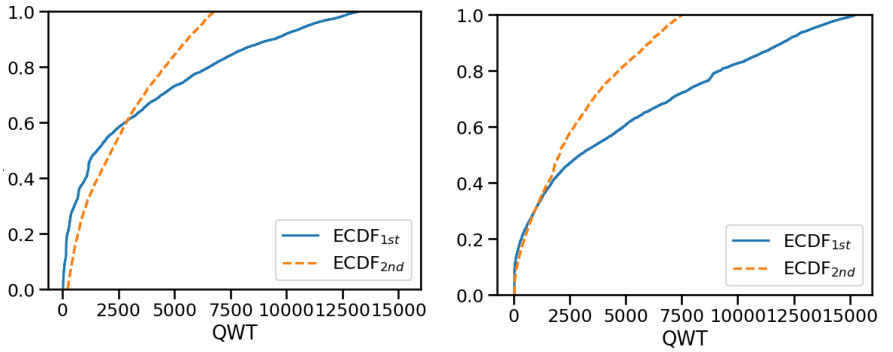
We then compare the ECDFs of the QWT obtained through real data and through the simulation, as depicted in Figure 5. We observe a fairly similar behavior, particularly in quartiles Q2, Q3 and Q4. We can even observe the simulation mimicking the different effect of the change in policy in jobs with shorter and longer QWT, as described in Section 4.4, albeit with a different crossing point.

5.3. BC workload into AC configuration

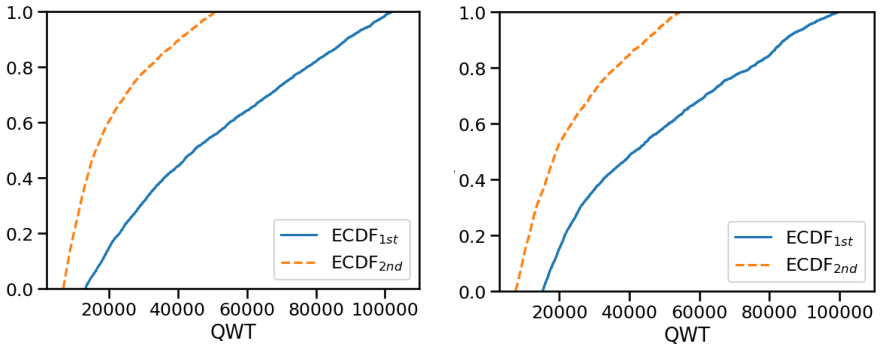
Assuming that the simulation approximately reflected the behavior of the system, we observed the processing of the BC period workload with the resource management policy of the AC period, thus imitating an iteration in an optimization process. We identified that the system behavior, measured by QWT, was not captured from the BC period workload simulation with the AC period configuration, as shown in Table 4, considering an average variation of 75% in the compared statistics. This was because some of the jobs did not fit the new policy. Therefore, it would be necessary to reshape the jobs and evaluate how the system would behave if the jobs had fit into the new policy.



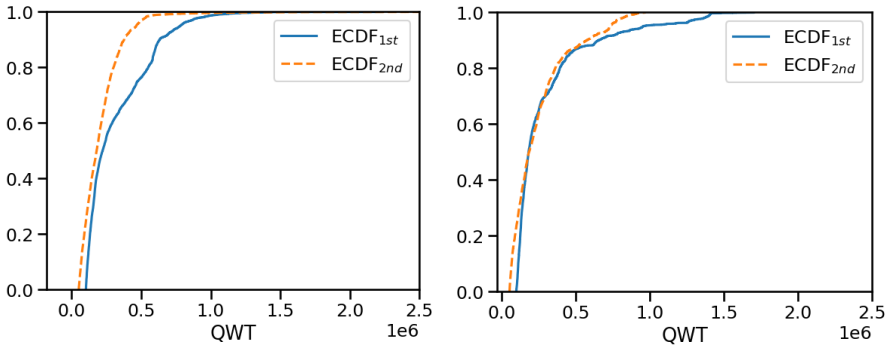
(a) Q1 – real (b) Q1 – simulated



(c) Q2 – real (d) Q2 – simulated



(e) Q3 – real (f) Q3 – simulated



(g) Q4 – real (h) Q4 – simulated

Figure 5. Comparison between the ECDF of real and simulated data.

Table 4. BC workload into AC configuration descriptive statistics comparison.

	QWT		Req. nodes		Req. timelimit		Elapsed time		Inter-arrival time	
	real	simulated	real	simulated	real	simulated	real	simulated	real	simulated
Avg.	94,259	33,157	10	9	108,705	104,548	38,118	36,483	942	910
Std. dev.	179,002	87,525	16	16	79,539	80,564	115,033	117,996	4,447	4,427
Min	0	0	1	1	60	60	0	0	0	0
1Q	31	12	1	1	14,400	14,400	92	68	1	1
2Q	13,197	25	4	4	172,800	172,800	2,566	2,197	59	55
3Q	102,076	16,537	10	8	172,800	172,800	31,583	27,309	420	397
Max	2,236,740	782,296	128	128	1,314,000	1,314,000	2,678,429	2,678,429	434,656	434,656

5.4. Workload processing with job shaping

We identified a set of jobs from the BC period that were not processed with the AC period configuration due to queue configurations that limited the geometry of the jobs to a certain interval. We then assessed two scenarios of shaping these unprocessed jobs considering new resource management policies:

1. Doubling the requested amount of cores and halving the time limit for the `cpu` queue;
2. Halving the requested amount of cores and doubling the time limit for the `cpu_long` queue.

This shaping ensured that in each scenario, jobs not processed in the workload injection were processed by the simulation in an alternative queue, maintaining the requested geometry. Tables 5 and 6 present the results of each scenario. As observed, when performing job shaping it was possible to better approximate the system behavior compared to workload injection without job shaping.

Nevertheless, when comparing scenarios 1 and 2, we observe that the choice of the target queue for job shaping impacts the simulation result, with an average variation in QWT statistics of 18% for the `cpu` queue and 16% for the `cpu_long` queue. Therefore, the importance of assessing different destination queues in the simulation of workload processing scenarios stands out.

Table 5. BC workload with job shaping to `cpu` queue into AC configuration descriptive statistics comparison.

	QWT		Req. nodes		Req. timelimit		Elapsed time		Inter-arrival time	
	BC simulated	simulated	BC simulated	simulated	BC simulated	simulated	BC simulated	simulated	BC simulated	simulated
Avg.	88,096	90,910	10	10	108,702	102,521	38,116	35,868	942	942
Std. dev.	189,948	199,150	16	17	79,539	77,876	115,029	113,696	4,446	4,447
Min	0	0	0	1	60	60	0	0	0	0
1Q	27	28	1	1	14,400	14,400	92	88	1	1
2Q	15,247	16,133	4	4	172,800	111,600	2,566	2,407	59	59
3Q	99,558	98,922	10	10	172,800	172,800	31,580	29,820	420	420
Max	1,704,118	1,904,918	128	128	1,314,000	1,314,000	2,678,429	2,678,429	434,656	434,656

6. Conclusion

In this paper, we presented a job shaping strategy that accommodates real workload traces to incompatible policies that may arise in optimization procedures based on discrete-event RMS simulations. Our main finding with a case study involving the Santos Dumont supercomputer at LNCC in Brazil was that job shaping allows capturing changes in system behavior simply and effectively, regardless of eventual incompatibilities between the real workload traces and the simulated policies. In this way, parameterizing the job shaping

Table 6. BC workload with job shaping to `cpu_long` queue into AC configuration descriptive statistics comparison.

	QWT		Req. nodes		Req. timelimit		Elapsed time		Inter-arrival time	
	BC simulated	**simulated**	**BC simulated**	**simulated**	**BC simulated**	**simulated**	**BC simulated**	**simulated**	**BC simulated**	**simulated**
Avg.	88,096	88,580	10	9	108,702	121,073	38,116	42,617	942	942
Std. dev.	189,948	189,171	16	15	79,539	98,771	115,029	121,473	4,446	4,447
Min	0	0	0	1	60	60	0	0	0	0
1Q	27	27	1	1	14,400	18,000	92	95	1	1
2Q	15,247	15,206	4	4	172,800	172,800	2,566	2,751	59	59
3Q	99,558	98,473	10	8	172,800	172,800	31,580	34,221	420	420
Max	1,704,118	1,453,906	128	128	1,314,000	1,314,000	2,678,429	2,678,429	434,656	434,656

strategy allows for an in-depth evaluation of optimization opportunities, offering insights into how configurations impact overall system performance.

It is important to highlight that the job shaping method used is simple and considers that all shaped jobs have linear speed-up behavior. The study was also limited to evaluating the application of only one job shaping method per scenario. As future work, there is a need to evaluate other system behavior metrics in addition to QWT, e.g. system utilization. It is also important to evolve the job shaping technique, allowing the evaluation of more complex scenarios, e.g. involving workflows.

References

Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

Chapin, S. J., Cirne, W., Feitelson, D. G., Jones, J. P., Leutenegger, S. T., Schwiegelshohn, U., Smith, W., and Talby, D. (1999). Benchmarks and standards for the evaluation of parallel job schedulers. In Feitelson, D. G. and Rudolph, L., editors, *Job Scheduling Strategies for Parallel Processing*, pages 67–90, Berlin, Heidelberg. Springer.

Cirne, W. and Berman, F. (2001). A model for moldable supercomputer jobs. In *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001*, pages 8 pp.–.

Dutot, P.-F., Mercier, M., Poquet, M., and Richard, O. (2017). Batsim: A realistic language-independent resources and jobs management systems simulator. In Desai, N. and Cirne, W., editors, *Job Scheduling Strategies for Parallel Processing*, pages 178–197, Cham. Springer International Publishing.

Fu, M. C. (2014). *Handbook of Simulation Optimization*. Springer Publishing Company, Incorporated.

Galleguillos, C., Kiziltan, Z., Netti, A., and Soto, R. (2020). AccaSim: a customizable workload management simulator for job dispatching research in HPC systems. *Cluster Computing*, 23(1):107–122.

Gomes, A. T. A. (2018). Assessing the behavior of HPC users and systems: The case of the Santos Dumont supercomputer. Lecture of the XIX Brazilian Symposium on High-Performance Computing Systems (WSCAD), São Paulo, Brazil.

Gupta, A., Acun, B., Sarood, O., and Kalé, L. V. (2014). Towards realizing the potential of malleable jobs. In *2014 21st International Conference on High Performance Computing (HiPC)*, pages 1–10.

Jokanovic, A., D'Amico, M., and Corbalan, J. (2018). Evaluating SLURM simulator with real-machine SLURM and vice versa. In *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pages 72–82. IEEE.

Klusáček, D., Soysal, M., and Suter, F. (2020). Alea – complex job scheduling simulator. In Wyrzykowski, R., Deelman, E., Dongarra, J., and Karczewski, K., editors, *Parallel Processing and Applied Mathematics*, pages 217–229, Cham. Springer International Publishing.

Posner, J., Hupfeld, F., and Finnerty, P. (2024). Enhancing supercomputer performance with malleable job scheduling strategies. In Zeinalipour, D., Blanco Heras, D., Pallis, G., Herodotou, H., Trihinas, D., Balouek, D., Diehl, P., Cojean, T., Fürlinger, K., Kirkeby, M. H., Nardelli, M., and Di Sanzo, P., editors, *Euro-Par 2023: Parallel Processing Workshops*, pages 180–192, Cham. Springer Nature Switzerland.

Prabhakaran, S., Neumann, M., Rinke, S., Wolf, F., Gupta, A., and Kale, L. V. (2015). A batch system with efficient adaptive scheduling for malleable and evolving applications. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 429–438.

Rodrigo, G. P., Elmroth, E., Östberg, P.-O., and Ramakrishnan, L. (2018). ScSF: A scheduling simulation framework. In Klusáček, D., Cirne, W., and Desai, N., editors, *Job Scheduling Strategies for Parallel Processing*, pages 152–173, Cham. Springer International Publishing.

Sabin, G., Lang, M., and Sadayappan, P. (2007). Moldable parallel job scheduling using job efficiency: An iterative approach. In Frachtenberg, E. and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, pages 94–114, Berlin, Heidelberg. Springer.

Simakov, N. A., Deleon, R. L., Lin, Y., Hoffmann, P. S., and Mathias, W. R. (2022). Developing accurate Slurm simulator. In *Practice and Experience in Advanced Research Computing 2022: Revolutionary: Computing, Connections, You*. Association for Computing Machinery.