Exploring Simplicity and Efficiency: Regression-based Scheduling Heuristics in HPC

Lucas Rosa[1], **Danilo Carastan-Santos**[2], **Alfredo Goldman**[1]

[1]Institute of Mathematics and Statistics, Department of Computer Science
University of São Paulo (USP) — São Paulo/SP — Brazil

[2]Laboratoire d'Informatique de Grenoble, CNRS, Inria, Grenoble INP
Univ. Grenoble Alpes (UGA) — Grenoble/FR — France

roses.lucas@usp.br, danilo.carastan-dos-santos@inria.fr,

gold@ime.usp.br

Abstract. *This research examines the interplay between resource management in high-performance computing systems and the application of machine learning techniques in developing scheduling heuristics. The potential for improved performance, through scheduling heuristics based on linear regression and polynomial job characteristics, was explored. Larger polynomials caused instability due to multicollinearity effects, but the simplest polynomial delivered stable and efficient scheduling performance. The study also evaluates the long-term resilience of these regression-based heuristics.*

1. Introduction

High-Performance Computing (HPC) has become a key tool for advances in science and industry, generating discoveries and bringing innovative solutions. Enhancing the efficiency of current HPC platforms can be achieved through several strategies, with resource management standing out. Resource competition occurs when a large number of users compete to execute their jobs on shared platforms. Solving the complex problem of parallel job scheduling is an important part of optimizing resource allocation.

Although the theoretical side of parallel job scheduling has been well studied [Dutot et al. 2016, Lucarelli et al. 2018], implementation considerations, varying assumptions, and the inherent unpredictability of the problem in HPC have motivated both practitioners and researchers to implement and study simple heuristics. EASY Backfilling is the most widely used heuristic on HPC platforms [Carastan-Santos et al. 2019]. A number of resource and job management software solutions, most notably SLURM [Yoo et al. 2003], make use of this heuristic.

Collecting information about the jobs executed on the platform is a common practice among HPC platform maintainers. These workload logs contain details about the jobs, such as processing times, processor requests, wait times, and so on. Given the increasing amount of data generated by HPC platforms and the need for efficient scheduling strategies, we aimed to understand the relationship between simplicity and efficiency in the design of scheduling heuristics using a simple approach involving simulation and regression.

2. Extracting Scheduling Knowledge

To simulate, we considered an HPC platform represented by a homogeneous cluster composed of m processors. This approach uses a pair of job sets, S and Q, to represent the current state of the platform and the waiting queue, respectively. The job sets are derived from a workload log file. The simulation starts with jobs from set S being processed in a first-come, first-served (FCFS) order. Once the submission of jobs from set S is complete, jobs in set Q begin to branch out into trials Q_i^*, $i = 1, 2, 3, \ldots$, which are random permutations representing possible scheduling scenarios. The jobs are then scheduled based on the particular order given by each permutation. We define $\mathcal{P}$ as being the set containing all sampled permutations.

At the end of the scheduling simulation for all branches, each job j in Q is assigned a score. This score is given by Equation 1. The score represents the impact of scheduling a job to execute first, as measured by the average bounded slowdown of all the jobs in Q. A lower score indicates a more positive impact on the overall slowdown of executing that job first.

$$score(j) = \frac{\sum_{Q_i^* \in \mathcal{P}(j_0=j)} \mathrm{AVGbsld}(Q_j^*)}{\sum_{Q_k^* \in \mathcal{P}} \mathrm{AVGbsld}(Q_k^*)} \tag{1}$$

By combining the generated samples from multiple independent tuples (S, Q), we produce a distribution $score(j)$. We decided to represent a job j based on certain attributes: Estimated processing time (p_j), number of processors required for execution (q_j), and submission time (r_j).

3. Regression Analysis

By applying a weighted multiple linear regression to the score distribution, we can derive a generalized, smoother representation that serves as a job sorting function. This function could be used to make decisions about the scheduling of jobs on an HPC platform.

We have defined four functions (Equation 2 through 5) to derive the scheduling heuristics. A linear combination of the job attributes p, q, and r is represented by the function Lin. The remaining functions, Qdr, Cub, and Qua, gradually increase the degree of the job attributes and includes a multiplicative factor pq, referred to in the literature as the job area. Each factor correspond to a well-known scheduling policies such as First-Come, First-Served ($FCFS(j) = r_j$), Shortest Processing Time first ($SPT(j) = p_j$), Shortest Number of Processors first ($SQF(j) = q_j$), and Shortest Area first ($SAF(j) = p_j q_j$).

$$Lin(j) = \theta_0 + \theta_1 p_j + \theta_2 q_j + \theta_3 r_j, \tag{2}$$

$$Qdr(j) = Lin(j) + \theta_4 p_j^2 + \theta_5 q_j^2 + \theta_6 r_j^2 + \theta_7 p_j q_j, \tag{3}$$

$$Cub(j) = Qdr(j) + \theta_8 p_j^3 + \theta_9 q_j^3 + \theta_{10} r_j^3 + \theta_{11} p_j^2 q_j + \theta_{12} p_j q_j^2, \tag{4}$$

$$Qua(j) = Cub(j) + \theta_{13} p_j^4 + \theta_{14} q_j^4 + \theta_{15} r_j^4 + \theta_{16} p_j^3 q_j + \theta_{17} p_j^2 q_j^2 + \theta_{18} p_j q_j^3. \tag{5}$$

4. Results and Discussion

The scheduling simulations initially produced three datasets. The synthetic workload trace generated using the Lublin and Feitelson workload model and two real-world work-

loads from the Parallel Workloads Archive [Feitelson et al. 2014], namely CTC-SP2 and SDSC-Blue, were used to generate the job characteristics.

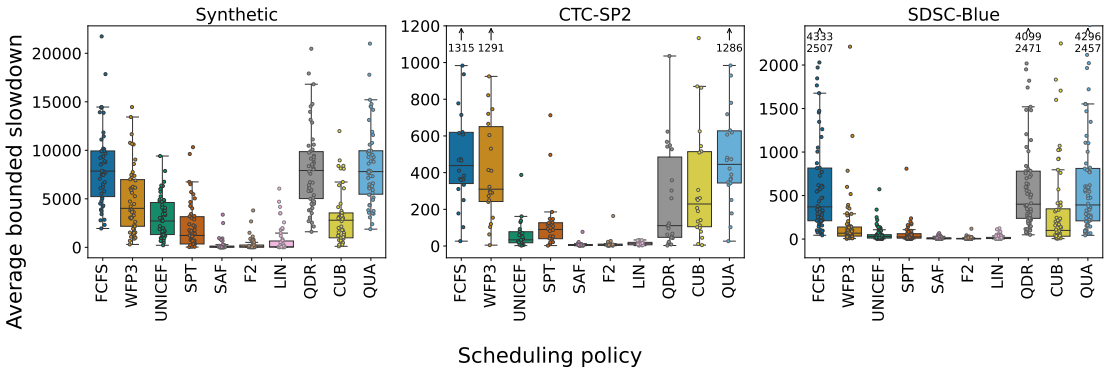

Figure 1. Schedule performance results by scheduling jobs derived from the same workloads used in the simulation phase, and by incorporating actual processing time when making scheduling decisions.

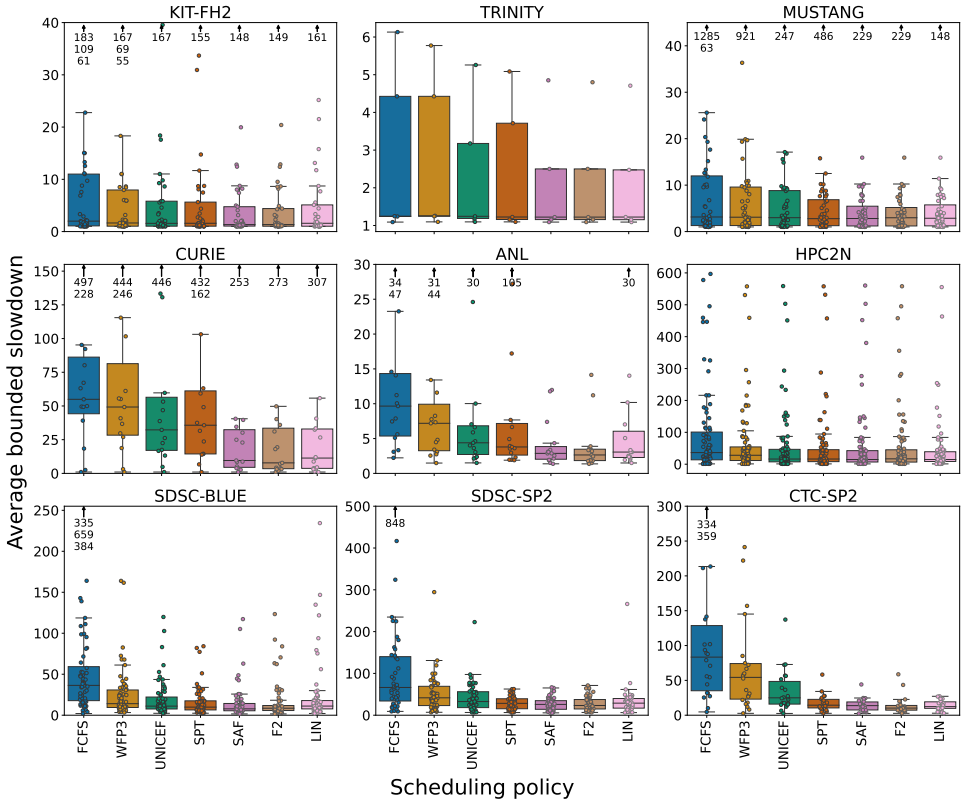

Figure 2. Computed average bounded slowdown for different scheduling policies. Experiments based on user-estimated processing times, with backfilling algorithm.

We then used these datasets to determine the optimal parameters for the proposed scheduling heuristics. The Mean Absolute Error (MAE) values resulting from the regression were relatively small and consistent for the same function across different workloads, indicating that the functions are representing the score distributions. However, the addition of derivative features and their correlation with the original job attributes resulted in

high values for the Variance Inflation Factor [García García et al. 2022] (VIF), suggesting a multicollinearity problem. We found that the simplest function was the most effective. While more complex functions resulted in poorer scheduling performance (Figure 1).

We then evaluated the efficiency of the heuristics obtained through regression on various high-performance computing platforms and workloads, as shown in Figure 2. This involved a simulation campaign using workload data collected over 19 years, spanning multiple platform and workload generations. By evaluating the efficiency of Lin and introducing $F2$, another machine learning-derived scheduling heuristic, against other policies, our results highlighted that regression-based scheduling heuristics can provide stable and efficient performance across diverse high-performance computing environments without the need for coefficient adjustments over time.

5. Future Work

In future studies, we intend to improve our regression methodology by including more features related to workload attributes and platform utilization, including elements such as platform utilization levels, remaining job processing time, and time of day. In addition, to address the growing concern about energy consumption in high performance computing, we intend to incorporate energy-aware considerations into our heuristics. Our goal is to reduce the overall power consumption of the platform while achieving comparable scheduling efficiency.

References

Carastan-Santos, D., De Camargo, R. Y., Trystram, D., and Zrigui, S. (2019). One Can Only Gain by Replacing EASY Backfilling: A Simple Scheduling Policies Case Study. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 1–10, Larnaca, Cyprus. IEEE.

Dutot, P.-F., Saule, E., Srivastav, A., and Trystram, D. (2016). Online Non-preemptive Scheduling to Optimize Max Stretch on a Single Machine. In Dinh, T. N. and Thai, M. T., editors, *Computing and Combinatorics*, volume 9797, pages 483–495. Springer International Publishing, Cham.

Feitelson, D. G., Tsafrir, D., and Krakov, D. (2014). Experience with using the Parallel Workloads Archive. *Journal of Parallel and Distributed Computing*, 74(10):2967–2982.

García García, C., Salmerón Gómez, R., and García Pérez, J. (2022). A review of ridge parameter selection: Minimization of the mean squared error vs. mitigation of multicollinearity. *Communications in Statistics - Simulation and Computation*, pages 1–13.

Lucarelli, G., Moseley, B., Thang, N. K., Srivastav, A., and Trystram, D. (2018). Online Non-preemptive Scheduling on Unrelated Machines with Rejections. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 291–300, Vienna Austria. ACM.

Yoo, A. B., Jette, M. A., and Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. In Goos, G., Hartmanis, J., Van Leeuwen, J., Feitelson, D., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862, pages 44–60. Springer Berlin Heidelberg, Berlin, Heidelberg.