Predicting the efficiency of job scheduler actions

Ana Eloina Nascimento Kraus[1], **Guilherme Diel**[1], **Guilherme Piêgas Koslovski**[1]

[1]Santa Catarina State University (UDESC) – Joinville – SC – Brazil

Abstract. *The order in which tasks are executed in High Performance Computing (HPC) infrastructures is fundamental to the efficiency of the virtual environment. This article covera a amachine learning- and polynomial regression-aided effort of predicting and thus, drawing out a better understanding of, the individual performances of various job scheduling algorithms.*

1. Introduction

Establishing an order to the execution of important tasks in a High Performance Computing (HPC) system calls for the power of job scheduling algorithms to sort out long queues. Specifically, job scheduling algorithms depend on the system administrator and on the workload [Brucker 1999]. There is a multitude of algorithms commonly used by the specialized literature: Shortest Job First (SJF), First Come - First Serve (FCFS), Smallest Area First (SAF), EASY Backfilling, F1, F2, F3 and F4, non-exhaustively. In the case of F1 through F4, proposed by [Carastan-Santos and de Camargo 2017], the job scheduling heuristics are statistically-backed nonlinear functions obtained through simulation of scheduling of generalistic workloads with state-of-the-art ad-hoc algorithms.

Applying the correct strategy for job scheduling in a processing-intensive environment such as a HPC datacenter is no trivial task and the concerns in this choice range from environmental to financial. For example, choosing the most appropriate algorithm based on the profile and patterns of the workflow being processed will bring considerable energy usage benefits [Casagrande et al. 2022]. In this context and with such objective, this work focused on using mathematical regression tools to obtain regression models consistently capable of predicting the outcome of SJF-, FCFS-, SAF-, EASY-, F1-, F2-, F3- and F4-scheduled workloads. For our study we focused on the score, in three ways for each job: energy, average slowdown, and execution time. The score of each job represents its contribution, or influence, in the workload's general performance regarding a particular variable. By preprocessing and, finally, by working with polynomial regression, we aimed to predict and study the overall performance of job schedulers. Fitting a line to the data was the pivot point of this work.

This paper is organized in the following manner: a brief going over the data and code work involved is presented in Section 2. Experimental results are discussed in Section 3, and finally, Section 4 concludes the work.

2. Data, tools and implementation

2.1. Tasks and jobs

Distributed applications are scheduled, that is, lined up for execution task by task, in a hierarchy devised by the chosen scheduling algorithm. In an HPC datacenter, hosts, that is, the computational resources of a physical machine, take on jobs one at a time from the scheduler. Scheduling jobs and tasks in HPC is to ensure that the machines run at

the edge of their capacities in speed and parallelism and the scheduling algorithm plays a fundamental role in this requirement. Every task has key attributes as follows: time of arrival, requested resources, requested time, execution time, and slowdown. The last two, which make up great part of a task's relevance in its workflow, can only be precisely determined after scheduling and after proper execution [Brucker 1999].

In this research, the Standard Workload Format [Chapin et al. 1999] was used and specifically the first phase of research consisted in simulating the scheduling of the SDSC-SP2, KTH-SP2, SDSC-BLUE and HPC2N raw workloads. Each workload has between 28000 to 250000 jobs waiting do be scheduled. In parallel, the SimGrid simulation framework [Casanova 2001] and its Python API for batch scheduling, pybatsim, were selected for composing the prototype. The workloads were scheduled in the way of the 7 aforementioned heuristics, seeking to simulate a HPC infrastructure of 32 and of 64 hosts. Each server can host up to a single task, without preemption. Our focus was on the numbers of energy consumption of each task, average slowdown of the entire workload at the moment of scheduling, and the estimated execution time of each task.

2.2. Score functions

After every combination of workload, platform and scheduling algorithm was calculated, we started assigning scores to the jobs. The function used to assign a score for energy consumption places the wattage value for each job before the average energy consumption of the workload. The less energy it takes to schedule this job, the better a score for the job. In turn, the function used to assign a slowdown score consists on evaluating the turnaround time and the execution time of a job, putting them against each other to assert the job's contribution to the overall fluidity of the workload. The less a job slows down the workflow, the better a score this job will get. Last but not least, the function used to assign a score for the execution time simply considers the time taken to run the task or job after it was scheduled.

2.3. Implementation

The forward step consisted in applying specifically polynomial regression, as this kind of curve would be able to best interpolate the points in our dataset. Our intention was to compose statistic models that would enable the generation or improvement of scheduling policies. The preprocessing phase was composed of: reducing the operation overhead while maintaining statistical relevance of the results, thus, the extraction of a pseudo-randomized excerpt of the original dataset; then, conversion of the data into a treatable format; after that, using each sample's Local Outlier Factor to clean the dataset of statistical outliers; and, finally, normalizing the data to a range of 0 to 1.

Our final dataset contained 15 rows of data for each job, including the submission time, the waiting time and, most important to us, the execution time, the slowdown and the energy consumption of each job. The energy consumption, for example, is listed in watts, and does not vary with the scheduling algorithm used, seeing how the scheduling algorithms do not interfere with the core of the job itself. An opposite example among those listed would be the waiting time which does vary accordingly to the scheduling algorithm's priorities. On to the machine learning, the imperative train-test-splitting of the dataset was carried out on a proportion of 70/30, respectively. With the X and Y sets ready, we applied polynomial regression to the 30 percent set aside for testing of the

models, and then observed how successful their training had been with use of the RMSE (Root Mean Squared Error, a classic prediction quality measurement variable; the smaller, the better). In each of the three models, we made use of degree 4 polynomial regression, which proved to be a successful approximation to the tendencies shown by our dataset and the original tasks, without substantial additional load of operations. Interpolation by linear, quadratic and cubic polynomials remain available for comparison and development purposes. The data manipulation was handled by Pandas, the numerical operations were orchestrated by NumPy, the plotting of curves for polynomial regression was done by Matplotlib, the training of regression models was made by Sklearn, and then the pickle lib was used to serialize these models.

3. Experimental results

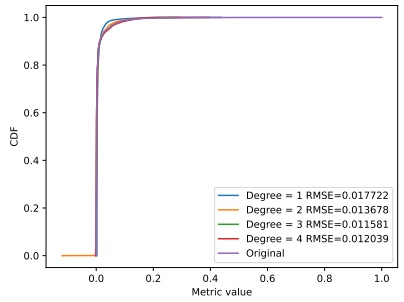
The Cumulative Distribution Functions (CDF) of all three key attributes are displayed in Figure 1. Each non-purple line represents the model with its respective degree polynomial and with its RMSE. The x axis corresponds to the possible values for the predicted variable mentioned in each subtitle, and the y axis is a percentage of values to lay in the specific range that x entails. The plotted curve of the chosen model should compare exactly to the plotted curve of the actual distribution of values for the dataset in question. That is, the two plotted curves should overlap. Increasing deviation from the purple plotted curve should mean the trained model has not correctly predicted values in the dataset and is not reliable.

All of the trained models were able to predict values very closely to what the dataset shows in the Average energy case. The percentage of samples that should fall in the 0.0 to 0.2 range of metric value, to illustrate, is shown and indeed corresponds to the most part (if not entirety) of the original dataset in this regard. The Execution time CDF is in a similar situation, with an even smaller variation of the RMSE. And from what Figure 1, (b) shows, the training of the four models in the chosen attributes to predict the Average slowdown of a workload at the time of scheduling one specific task resulted in few intersection points with the purple curve, which places the slowdown as the most clearly distinguishing score from one workload to another.

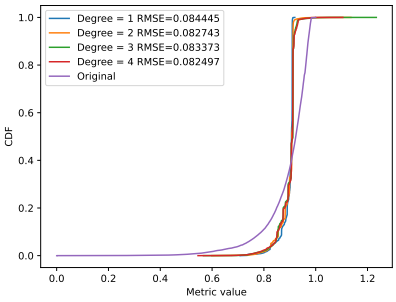
4. Conclusions

The job scheduling problem has been thoroughly studied, described, and has seen many optimized solutions ever since. Our research contributes to this by adding to the verifiable sum of knowledge and stress testing of the main job scheduling algorithms in the literature, making a scientifically backed, more straightforward decision [Shyalika et al. 2020] out of running a specific algorithm to schedule tasks in a real life system. With a combination of scheduling policies, job workloads and HPC DC infrastructures, we put together a dataset that would allow us to analyze the workflow idiosyncrasies of each algorithm and their behavior facing robust, heavy parallel workloads. All that we accomplished was via simulation, machine learning and regression, having yielded three trained models able to guide sysadmins into understanding the behavior of their scheduling policies.

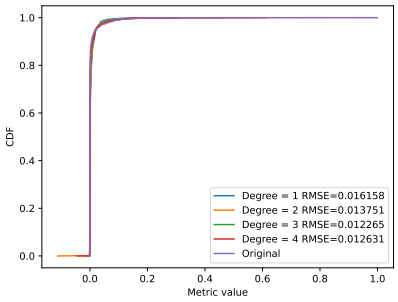
Acknowledgment: This work was funded by the National Council for Scientific and Technological Development (CNPq), the Santa Catarina State Research and Innovation Support Foundation (FAPESC), Santa Catarina State University (UDESC), and developed at Laboratory of Parallel and Distributed Processing (LabP2D).



(a) Execution time (CDF).



(b) Average slowdown (CDF).



(c) Average energy (CDF).

Figure 1. Polynomial regression for scheduling parameters.

References

Brucker, P. (1999). Scheduling algorithms. *Journal-Operational Research Society*, 50:774–774.

Carastan-Santos, D. and de Camargo, R. Y. (2017). Obtaining Dynamic Scheduling Policies with Simulation and Machine Learning. In *SC'17 -2 International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing)*, Denver, United States.

Casagrande, L., Koslovski, G., Miers, C.C., P., M.A., and Gonzalez, N. (2022). Don't hurry be green: scheduling servers shutdown in grid computing with deep reinforcement learning. In *International Journal of Grid and Utility Computing*. Inderscience Publishers.

Casanova, H. (2001). Simgrid: A toolkit for the simulation of application scheduling. In *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 430–437. IEEE.

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 Berlin Heidelberg.

Shyalika, C., Silva, T., and Karunananda, A. (2020). Reinforcement learning in dynamic task scheduling: A review. *SN Computer Science*, 1:306.