Simulation-Driven Portfolio Scheduling for Scientific Workflows

João Antonio Soares[1], **Gerson Geraldo H. Cavalheiro**[1]

[1] Centro de Desenvolvimento Tecnológico (CDTec) – Universidade Federal de Pelotas (UFPel)

{jansoares,gerson.cavalheiro}@inf.ufpel.br

Abstract. *This paper proposes portfolio scheduling approach for scientific workflow management systems (WMS). The goal is to improve scheduling decisions by dynamically selecting suitable scheduling policies during workflow execution. The portfolio integrates multiple task scheduling and resource allocation strategies, considering both computational and network characteristics.*

1. Introduction

Distributed systems form the backbone of modern scientific research, where workloads are predominantly heterogeneous workflows that combine data-intensive and compute-intensive jobs. WMS automate application deployment and execution, inherently making scheduling decisions. Scheduling policies help WMS efficiently distribute workloads and minimize execution time. However, managing highly heterogeneous workflows with fluctuating resource requirements and potential system failures remains a complex challenge.

One approach to address this challenge is portfolio scheduling, described by [Deng et al. 2013]. Portfolio scheduling dynamically selects and applies a scheduling policy at runtime from a predefined set of policies, called a portfolio. This work builds upon the Simulation-Driven Portfolio Scheduling (SDPS) approach [Casanova et al. 2023]. While their work successfully implemented SDPS in a simulated environment, it did not address key challenges related to monitoring runtime resources in production WMSs. Additionally, it did not consider the constraints imposed by the static planning phase of such systems, which often limits the feasibility of dynamic scheduling in real-world scenarios.

The objective of this work is to present the design and implementation of a SDPS component integrated into a WMS, with a focus on runtime system integration. The primary challenge addressed is integrating the online simulator with the core components of the WMS to enable dynamic scheduling adjustments. Section 2 details the high-level description of the architecture and integration.

2. Structure of Proposed Solution

A portfolio scheduling mechanism, utilizing discrete event simulation to determine optimal scheduling policies, is proposed. This mechanism will be integrated into Pegasus WMS (`https://pegasus.isi.edu`) and invoked during workflow execution via the `pegasus-plan` command. The implementation will leverage Python and HTCondor (`https://htcondor.org`) services, the underlying middleware behind Pegasus, through its Python bindings. The system comprises the following key components:

- **System Monitoring**: The system state is continuously updated using HTCondor's command line monitoring tools (e.g., `condor_q` and `condor_history`)

at fixed checkpoint intervals. This querying data feeds into a simulation module that evaluates various scheduling policies based on workload conditions.

- **Online Simulator**: This component evaluates the performance of different scheduling policies under current workload conditions. It receives the system status and user-defined workload as input, executing simulations for each policy in the portfolio. The primary evaluation metric initially focuses on minimizing makespan, although future iterations may incorporate additional user-defined performance metrics. The SimGrid toolkit is employed for discrete event simulation, though alternatives may be considered.
- **Policy Portfolio**: The portfolio of scheduling policies is structured into two categories: task selection and resource selection. Network infrastructure characteristics are also considered into resource selection for task submission.
- **WMS Middleware Interfacing**: The SDPS module operates as a command line program executed when the WMS launches a workflow. Integration with Pegasus is planned, potentially through a pre-execution script during the planning phase. Real-time scheduling modifications require interaction with the HTCondor middleware. This will be achieved by interfacing with the `schedd` job queue daemon and modifying job scheduling properties via HTCondor's `ClassAds` properties.

The portfolio scheduling mechanism activates before the submission of the first workflow task. The simulation predicts the most efficient scheduling policy based on current system conditions, as described in [McDonald et al. 2024]. Subsequent portfolio scheduling rounds are triggered at predefined execution thresholds (e.g., 25% workflow completion) to recalibrate the strategy and mitigate prediction errors.

3. Future Work

Next efforts will focus on expanding the policy portfolio with more sophisticated approaches, such as nature-inspired meta-heuristic optimization techniques. Benchmark evaluations will be conducted to compare the proposed system against other scheduling algorithms. Additionally, extensions to incorporate multi-objective optimizations, including energy efficiency and fault tolerance, are planned.

References

Casanova, H., Wong, Y. C., Pottier, L., and da Silva, R. F. (2023). On the feasibility of simulation-driven portfolio scheduling for cyberinfrastructure runtime systems. In Klusáček, D., Julita, C., and Rodrigo, G. P., editors, *Job Scheduling Strategies for Parallel Processing*, pages 3–24, Cham. Springer Nature Switzerland.

Deng, K., Song, J., Ren, K., and Iosup, A. (2013). Exploring portfolio scheduling for long-term execution of scientific workloads in iaas clouds. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, New York, NY, USA. Association for Computing Machinery.

McDonald, J., Dobbs, J., Wong, Y. C., Ferreira da Silva, R., and Casanova, H. (2024). An exploration of online-simulation-driven portfolio scheduling in workflow management systems. *Future Generation Computer Systems*, 161:345–360.