Online Tarema: Learning Task and Infrastructure Characteristics at Workflow Runtime

Friedrich M. Tschirpke

November 17, 2023

Scientific data analysis applications continually grow in complexity and data sizes. This growth increases the need for automation, reproducibility, and scalability in scientific applications. Scientific Workflow Management Systems (SWMSs) such as Nextflow [1], Pegasus [2], and Snakemake [3] provide a solution by allowing users to abstractly describe their complex applications as data analysis workflows consisting of several abstract tasks. The SWMSs handle the data management and execution of workflows on typically heterogeneous, distributed clusters.

Usually, the heterogeneity of the cluster resources extends to I/O speeds, computational speeds, and memory access speeds. However, resource managers like Kubernetes[1], YARN [4], or Slurm [5] do not consider resource differences beyond the number of required CPU cores and memory when scheduling the execution of tasks. Additionally, tasks in a workflow may differ greatly in resource usage as some are computationally intensive while others are I/O-heavy. For a reduction in workflow execution makespan, it is therefore of interest to consider tasks' resource usage and clusters' heterogeneity when making scheduling decisions.

For resource-aware scheduling decisions, the scheduler requires historic runtime data or needs to deduce the resource properties of the tasks and the system during the execution of the workflow. Using historic runtime data is impractical as it makes the scheduling method only possible if historic data is available and the input data, workflow, and cluster do not change. In contrast, the deduction of resource properties during runtime allows resource-aware scheduling for every execution without storing historical data including cold start executions.

[1] `https://kubernetes.io/`

Tarema [6] determines tasks' resource usage during runtime and node resource characteristics through benchmarks. Each node and each task receives labels for their resource characteristics. Based on these labels, the scheduler maps computationally expensive tasks to nodes with faster CPUs and I/O-heavy tasks to nodes with higher I/O speeds. However, additional benchmarks need to be executed for this approach.

1 Related Work

The cluster scheduling problem deals with assigning jobs to heterogeneous computing resources while considering the multidimensionality of resources, the heterogeneity of jobs, the highly fluctuating resource availability, and an optimization goal such as wall clock time [7]. Resource scheduling research has focused on designing scalable and heterogeneity-aware architectures and algorithms to best deal with these characteristics [7]. However, workflow scheduling remains a difficult problem due to its NP-hardness [8].

In this chapter, we will discuss heterogeneous cluster scheduling in general and then for clusters specifically. After that, we will discuss Tarema's [6] heterogeneity-aware workflow scheduling and CherryPick's [9] approach to learning cluster resources.

Researchers have developed many different workflow scheduling techniques for heterogeneous clusters [7]. The popular HEFT [10] algorithm and its dynamic adaptation P-HEFT [11] both require knowledge about task execution times. This is not feasible in many real-world applications.

Runtime systems like StarPU [12] for task scheduling on heterogeneous architectures have been proposed but are usually designed for high-performance architectures and not for scientific workflow executing in commodity clusters. Resource-aware scheduling approaches for clusters such as R-Storm [13] and RAS [14] focus on maximizing resource utilization without considering task characteristics or resource heterogeneity. In contrast, Rupam [15] is a heterogeneity-aware scheduling approach designed for Spark [16], an engine for task execution on clusters.

Rupam considers multiple aspects of heterogeneity beyond different numbers of CPU cores or different memory amounts. Like Rupam, the Tarema system [6] considers different computation, reading, and writing speeds. Tarema groups cluster nodes with similar performance based on benchmarks and ranks each node for each soft resource characteristics under investigation. Task monitoring data allow tasks to similarly be ranked into percentiles for each resource usage characteristic. Tarema labels tasks according to these percentiles allowing it to schedule tasks in the highest percentiles regarding a resource characteristic to nodes with the highest ranks for that resource characteristic and tasks in lower percentiles to nodes with lower ranks. The scheduler chooses the node such that the mean absolute error between node and task labels is minimal. Thus, the chosen node provides resources with similar characteristics to what the task uses. This approach is SWMS-independent, unlike Rupam which is explicitly designed for Spark executions.

CherryPick [9] is not a scheduling approach but learns cluster configurations for

cloud computing using downsampled test executions. CherryPick optimizes the cluster configuration for a given workload using Bayesian optimization while executing test runs allowing the user to choose a fitting cluster configuration for their needs. Bayesian optimization allows CherryPick to find a near-optimal solution after a few runs of the workload without providing a pre-defined format for the performance model. The fast convergence and adaptable model format make the method attractive for workflow applications because the scheduler usually does not know the format of tasks and having many unoptimized runs before finding the best solution is expensive on a cluster. However, CherryPick finds the optimal cluster configuration before workload execution and includes cloud provider prices in the calculation instead of optimizing task allocation on a provided cluster making the approach unsuitable for workflow scheduling.

2 Goals

This work aims to investigate the Tarema scheduling approach without running node benchmarks but instead learning the node characteristics at runtime while still dynamically labeling tasks according to their resource usage.

The Tarema system deduces task labels for resource characteristics well enough to make better scheduling decisions than round-robin, fair, fill nodes, and shortest-job-fastest-nodes approaches [6]. But Tarema's scheduling decisions rely on accurate node characteristics provided by benchmarks. It is our goal to investigate whether we can learn node characteristics at runtime with enough accuracy to perform similarly to offline Tarema. We aim to analyze the inaccuracies that might occur when learning node characteristics at runtime in combination with Tarema's dynamic task labeling as these may influence each other.

3 Implementation

We will implement our resource-aware scheduling approach for Nextflow and Kubernetes. Lehmann *et al.* propose a common workflow scheduler interface [17] and provide an implementation of their interface as a Nextflow extension[2] and Kubernetes scheduler[3]. We will extend them to implement our approach.

We will use a machine learning model to estimate the nodes' resources by observing the execution of different tasks. This model will use Bayesian optimization to produce good results within a few task runs and to allow for uncertainty. Uncertainty is high on a cluster especially with this approach because Tarema's dynamic task labels and the new model for the cluster resources will change simultaneously. Having two dynamic components will inevitably cause inaccuracies and uncertainty.

The node characteristics model will consider all abstract tasks with enough task instances to derive characteristics from their execution. Besides the node and the task,

[2] `https://github.com/CommonWorkflowScheduler/nf-cws`
[3] `https://github.com/CommonWorkflowScheduler/KubernetesScheduler`

we plan to use the uncompressed input size of each task instance as a parameter for the Bayesian optimization. The model's predictions for the node resources will inform task-node mappings using Tarema's labeling approach.

The execution order of the tasks and their node placements will influence the amount and kind of training data that will be available to the model. As long as the model's uncertainty for node resources is high, our scheduler will assign tasks to nodes randomly in order to collect data without any bias. Our scheduling approach will prioritize task instances whose abstract task has the least number of finished instances (LLF). Witt *et al.* propose LLF as the best task prioritization technique when learning resource properties during runtime [18].

4 Evaluation

We will evaluate our scheduling approach by comparing the runtimes of different workflows on a university-internal cluster with Tarema and the round-robin task ranking approach Rank Min-RR.

The university-internal cluster gives us control and reduces the noise and inconsistencies usually present when using virtual machines in the cloud and removes the financial costs associated with cloud services. Cloud services would allow choosing different cluster configurations whereas we will need to create different kinds of heterogeneity by running noise and stressing tools.

Tarema makes scheduling decisions based on more accurate data than we will be able to because the Tarema approach runs benchmarks instead of learning infrastructure characteristics at runtime. We want to ascertain if our approach is on par with the Tarema approach and outperforms the round-robin approach Rank Min-RR. Thus, we will investigate our scheduler's performance when learning the cluster resources during execution and its performance when we hard-code the resource information. With hard-coded resource information, our approach should perform like the original Tarema. Additionally, we will compare our approach to the task ranking strategy Rank Min-RR which prefers smaller input size tasks and assigns nodes using the round-robin principle. Rank Min-RR is one of the best-performing scheduling strategies evaluated by Lehmann *et al.* [17].

To evaluate the scheduling approaches we will execute nf-core [19] workflows. We will choose workflows that differ in structure in order to test as many behaviors as possible. We will execute each workflow with each scheduling strategy a minimum of three times to gain significant data.

References

[1] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature*

Biotechnology, vol. 35, no. 4, pp. 316–319, Apr. 2017, ISSN: 1087-0156, 1546-1696. DOI: 10.1038/nbt.3820.

[2] E. Deelman, K. Vahi, M. Rynge, *et al.*, "The Evolution of the Pegasus Workflow Management Software," *Computing in Science & Engineering*, vol. 21, no. 4, pp. 22–36, Jul. 2019, ISSN: 1521-9615, 1558-366X. DOI: 10.1109/MCSE.2019.2919690.

[3] J. Köster and S. Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, Oct. 2012, ISSN: 1367-4811, 1367-4803. DOI: 10.1093/bioinformatics/bts480.

[4] V. K. Vavilapalli, A. C. Murthy, C. Douglas, *et al.*, "Apache Hadoop YARN: Yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, Santa Clara California: ACM, Oct. 2013, pp. 1–16, ISBN: 978-1-4503-2428-1. DOI: 10.1145/2523616.2523633.

[5] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., red. by G. Goos, J. Hartmanis, and J. Van Leeuwen, vol. 2862, Berlin, Heidelberg: Springer Berlin Heidelberg, Jun. 2003, pp. 44–60, ISBN: 978-3-540-20405-3 978-3-540-39727-4. DOI: 10.1007/10968987_3.

[6] J. Bader, L. Thamsen, S. Kulagina, J. Will, H. Meyerhenke, and O. Kao, "Tarema: Adaptive Resource Allocation for Scalable Scientific Workflows in Heterogeneous Clusters," in *2021 IEEE International Conference on Big Data (Big Data)*, Orlando, FL, USA: IEEE, Dec. 2021, pp. 65–75, ISBN: 978-1-66543-902-2. DOI: 10.1109/BigData52589.2021.9671519.

[7] W. Khallouli and J. Huang, "Cluster resource scheduling in cloud computing: Literature review and research challenges," *The Journal of Supercomputing*, vol. 78, no. 5, pp. 6898–6943, Apr. 2022, ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-021-04138-z.

[8] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: A survey," *The Journal of Supercomputing*, vol. 71, no. 9, pp. 3373–3418, Sep. 2015, ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-015-1438-4.

[9] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, "CherryPick: Adaptively unearthing the best cloud configurations for big data analytics," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, Boston, MA: USENIX Association, Mar. 2017, pp. 469–482, ISBN: 978-1-931971-37-9.

[10] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, Mar. 2002, ISSN: 10459219. DOI: 10.1109/71.993206.

[11] J. G. Barbosa and B. Moreira, "Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters," *Parallel Computing*, vol. 37, no. 8, pp. 428–438, Aug. 2011, ISSN: 01678191. DOI: 10.1016/j.parco.2010.12.004.

[12] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures," in *Euro-Par 2009 Parallel Processing*, H. Sips, D. Epema, and H.-X. Lin, Eds., vol. 5704, Berlin, Heidelberg: Springer Berlin Heidelberg, Aug. 2009, pp. 863–874, ISBN: 978-3-642-03868-6 978-3-642-03869-3. DOI: 10.1007/978-3-642-03869-3_80.

[13] B. Peng, M. Hosseini, Z. Hong, R. Farivar, and R. Campbell, "R-Storm: Resource-Aware Scheduling in Storm," in *Proceedings of the 16th Annual Middleware Conference*, Vancouver BC Canada: ACM, Nov. 2015, pp. 149–161, ISBN: 978-1-4503-3618-5. DOI: 10.1145/2814576.2814808.

[14] J. Polo, C. Castillo, D. Carrera, *et al.*, "Resource-Aware Adaptive Scheduling for MapReduce Clusters," in *Middleware 2011*, F. Kon and A.-M. Kermarrec, Eds., red. by D. Hutchison, T. Kanade, J. Kittler, *et al.*, vol. 7049, Berlin, Heidelberg: Springer Berlin Heidelberg, Dec. 2011, pp. 187–207, ISBN: 978-3-642-25820-6 978-3-642-25821-3. DOI: 10.1007/978-3-642-25821-3_10.

[15] L. Xu, A. R. Butt, S.-H. Lim, and R. Kannan, "A Heterogeneity-Aware Task Scheduler for Spark," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, Belfast: IEEE, Sep. 2018, pp. 245–256, ISBN: 978-1-5386-8319-4. DOI: 10.1109/CLUSTER.2018.00042.

[16] M. Zaharia, R. S. Xin, P. Wendell, *et al.*, "Apache Spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2934664.

[17] F. Lehmann, J. Bader, F. Tschirpke, L. Thamsen, and U. Leser, "How Workflow Engines Should Talk to Resource Managers: A Proposal for a Common Workflow Scheduling Interface," in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, May 2023, pp. 166–179. DOI: 10.1109/CCGrid57682.2023.00025.

[18] C. Witt, D. Wagner, and U. Leser, "Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows," in *2019 International Conference on High Performance Computing & Simulation (HPCS)*, Dublin, Ireland: IEEE, Jul. 2019, pp. 761–768, ISBN: 978-1-72814-484-9. DOI: 10.1109/HPCS48598.2019.9188055.

[19] P. A. Ewels, A. Peltzer, S. Fillinger, *et al.*, "The nf-core framework for community-curated bioinformatics pipelines," *Nature Biotechnology*, vol. 38, no. 3, pp. 276–278, Mar. 2020, ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-020-0439-x.