Parallel Execution of Workflows driven by Distributed Database Techniques

Renan Souza, Marta Mattoso

COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

{renanfs,marta}@cos.ufrj.br

Abstract. *Many-Task Computing (MTC) workflow executions involve thousands of parallel tasks that consume and produce large amounts of data and are scheduled on multiple nodes in a large HPC cluster. A complete run may last for weeks. Users have to analyze and steer the dataflow at runtime. This introduces several challenges for efficient data management without jeopardizing performance. This dissertation combines distributed data management techniques (ACID transactions, concurrency control, and database design) to propose a scalable solution for MTC workflows. Domain data, dataflow provenance, and workflow execution data are managed together in an in-memory distributed DBMS. As a result, a distributed scheduling via transactions in this database attains high scalability in a 1,008-cores HPC cluster, while maintaining runtime data analytical capabilities.*

1. Introduction

Large-scale scientific computations in a wide variety of domains are often modeled as Many-Task Computing (MTC) workflows, with thousands of parallel tasks that run on a High Performance Computing (HPC) cluster. The tasks produce data elements to be consumed by other tasks in a coherent dataflow. A complete workflow execution may last for days in a large cluster and process tera or petabytes of complex scientific data. Meanwhile, users cannot wait for the workflow to finish so their result data analysis can start. They need to correlate input with output, check correctness of their hypotheses, modify simulation parameters, debug, visualize the flowing data elements, and steer the computation at runtime, maintaining high parallel efficiency of the HPC simulation. Since these processes are data-centric, an efficient data management that allows for data parallelism in MTC and runtime data analysis without jeopardizing performance is of utmost importance.

Parallel Scientific Workflow Management Systems (SWMSs) have been employed to orchestrate workflow executions in HPC [1]. Pegasus [3] and Swift [16] are two well-known SWMSs and are the most scalable ones. To allow for result data analysis, they collect distributed data provenance and store in multiple typically unstructured log files, which are much harder to query at runtime. Alternatively, they load the log files (through an ETL process) into a database for *post-mortem* analytical queries. Additionally, task scheduling data (information about each task, which node each task ran, CPU and memory consumed by each task) are managed completely separated from dataflow provenance and domain data stores. This highly limits analytical capabilities because users need to analyze the data integrating domain, execution, and provenance [10,14]. To cope with this, a data-oriented SWMS solution, called Chiron, was built [6]. Chiron adopts a DBMS to manage its scheduling data by updating it while processing parallel tasks and storing provenance data in this same database, which we call Workflow Database (wf-Database). Thus, users can query the same database being used by the SWMS scheduler, which has been shown to be very beneficial [7,13].

Nevertheless, all this runtime analytical support comes with a price. This data-oriented SWMS solution relies on a centralized task scheduling data management, using a master-workers design where the centralized master node is the only able to access the centralized DBMS. This introduces a significant performance overhead at the master node [14] highly limiting the system scalability to about 300 cores. Additionally, two single points of failure are introduced: the master node and the centralized DBMS, limiting the solution's fault tolerance. Therefore, we are not aware of a SWMS solution that both allows for analytical queries integrating domain data, provenance, and execution at runtime and is highly scalable.

In this dissertation, we make extensive use of distributed data management techniques (ACID transactions, concurrency control, and database design) to propose SchalaDB: a scalable data-oriented distributed task scheduling solution for SWMSs. In SchalaDB, all workers directly query and update the wf-Database through SQL. There is no centralized master to which workers need to communicate via message passing during scheduling. To accommodate multiple workers concurrently querying the wf-Database, the solution uses an in-memory distributed DBMS (DDBMS) with distributed ACID transactions. We implemented SchalaDB by completely redesigning Chiron's traditional centralized scheduling, and we call it d-Chiron. We evaluated it using synthetic benchmarks, and real case studies in oil and gas and bioinformatics in Grid5000 (www.grid5000.fr) clusters with up to 1,008 cores. As a result, this is the first SWMS that achieves high scalability in an HPC cluster of this size while maintaining support for rich data analysis at runtime. These are the main contributions of this dissertation:

- A scalable design for MTC workflow scheduling driven by a DDBMS. We specify how task scheduling and parallel data placement are done to maximize system performance, improve availability, and reduce load imbalance.
- A concrete implementation of this design in d-Chiron SWMS and performance tests on a 1,008-cores cluster.
- For reproducibility, all executables, instructions to use d-Chiron on an HPC cluster, pre-configured workflows, and sample analytical queries are on GitHub [4].

2. A Scalable Architecture for Scheduling MTC Workflows Driven by Distributed Data Management

This dissertation aims at providing high scalability while maintaining runtime data analytical support in a SWMS. Data analysis is supported via queries in the wf-Database. It follows PROV-Wf, an entity-relationship diagram that models workflow general concepts relevant for provenance data collection. PROV-Wf is based on a W3C recommendation, which facilitates integration and queries using the representation for provenance. For example, data from the wf-Database can be published on the semantic web to be consumed by different research groups, using different SWMSs [2]. An implementation of PROV-Wf in a concrete database schema is on [4]. Since the wf-Database is continuously populated at runtime, it can potentially be queried for data generation tracking, monitoring, and other advanced data analyses [7,10,13,14]. Also, a SWMS engine can use this data-oriented approach for runtime optimizations [6] or adaptivity [8].

To allow for these features, the SWMS engine updates the wf-Database as the tasks are created, scheduled, executed, and completed. It has to store all fine-grained task-related data in the wf-Database for each task. Thus, both users and the SWMS engine have the most up-to-date possible data available for structured queries. However, in MTC, there can be thousands of parallel tasks running, each taking few seconds to minutes. When the task scheduler is centralized, there are points of contention and failure, negatively impacting performance. SchalaDB, however, has a decentralized scheduling design and uses an in-memory DDBMS. Although some DDBMSs, like MySQL Cluster, are well-known for their efficiency in processing OLTP while being able to run OLAP queries [9], their use in MTC scheduling has not been experienced before.

Using a DDBMS in an MTC scheduler has several advantages beyond ACID transactions controlling multiple concurrent updates during task scheduling. Particularly, DDBMSs that allow for ACID transactions are useful to facilitate data consistency control when multiple concurrent updates occur in the task-related data during task scheduling. Moreover, DDBMSs enable robust parallel cache memory data management. Also, there are DDBMSs that run exclusively in the cluster main memory, avoiding intense disk I/O operations. Data replication and partitioning into multiple nodes are also well studied and implemented in many DDBMSs. Considering that a DDBMS already implements most of these mechanisms usually very efficiently [9], it can be an integrant part of an SWMS architecture and alleviate the effort on developing such complex controls inside the SWMS engine's source code. In this way, the SWMS developers can focus on specific concerns related to dataflow management (*e.g.*, data dependencies between tasks) instead of implementing distributed data management algorithms and dealing with sophisticated distributed concurrency issues and contention at scheduling queues. As centralized DBMSs, DDBMS also has query interfaces through which users can query the continuously populated wf-Database. Therefore, SchalaDB controls the parallel execution of workflows with a distributed task scheduling driven by a DDBMS.

Architecture details. For execution control, the main supporting relation is the Work Queue (WQ), which has the list of tasks to be scheduled and their data. SchalaDB distributes the WQ data across *D data nodes*. The data nodes are responsible for managing the distributed data partitions playing the role of multiple masters. SchalaDB uses database drivers to implement the *connectors*. Figure 1 illustrates SchalaDB.

It is the main data provider for the SWMS engine's distributed scheduler so that worker nodes can submit queries to retrieve the data to be used in a scheduling decision and a task execution. Instead of having workers requesting tasks to a master through regular message passing, like in traditional task scheduling implementations, workers send structured queries to the DDBMS. Instead of having a master to receive the worker request; get the next ready tasks; and send them to the worker, the DDBMS uses its distributed data nodes to respond the multiple concurrent requests from the workers, diminishing contention. Regarding availability, the DDBMS can use replication to replicate all relations, including the WQ relation. Since the wf-Database stores workflow control data, input and output domain data composing the dataflow, and paths to large scientific files stored on disk [6], it is not large and replication in the main-memory is viable considering a cluster with multiple nodes, each with at least few gigabytes of RAM. If a node hosting a WQ partition fails, there is still at least one extra replica that may be utilized. To increase availability in the system, each worker may connect and query the DDBMS via two different database connector communications: the main communication, represented by full gray lines in Figure 1 and the secondary communication by dashed gray lines. If one connector fails, workers connected to it just need to connect to their secondary database connector. In addition, the secondary supervisor node removes the single point of failure at the supervisor node. Data node's availability is outsourced to the DDBMS.

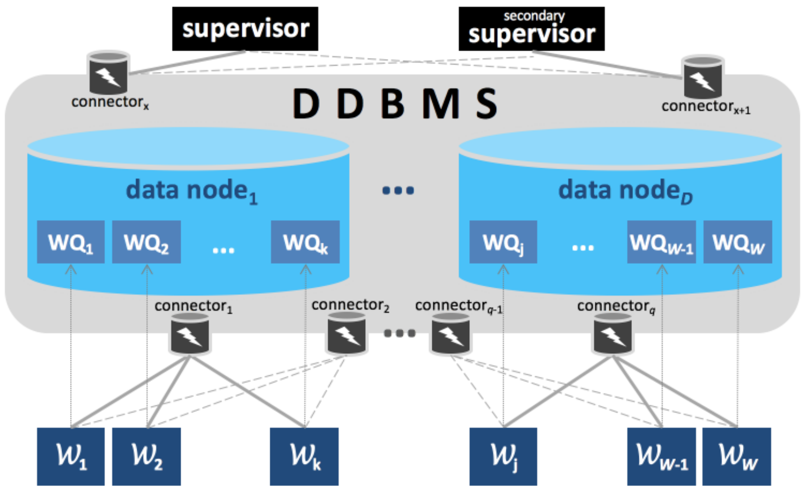

Figure 1. SchalaDB design. *W* workers directly accessing the DDBMS composed of *D* data nodes.

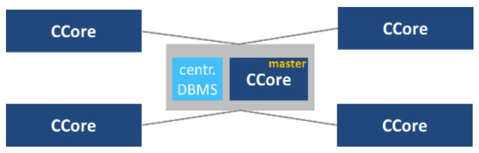

Figure 2. Chiron's centralized architecture relying on a centralized DBMS.

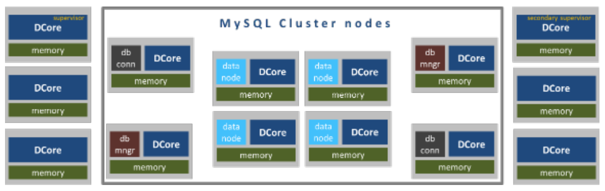

Figure 3. d-Chiron architecture. The gray boxes represent physical nodes in an HPC cluster.

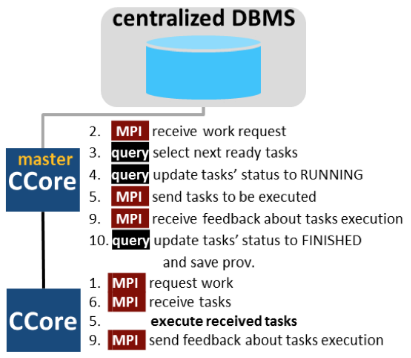
centralized DBMS
master CCore
CCore
2. MPI receive work request
3. query select next ready tasks
4. query update tasks' status to RUNNING
5. MPI send tasks to be executed
9. MPI receive feedback about tasks execution
10. query update tasks' status to FINISHED and save prov.
1. MPI request work
6. MPI receive tasks
5. execute received tasks
9. MPI send feedback about tasks execution

Figure 4. Centralized scheduling with a centralized DBMS.

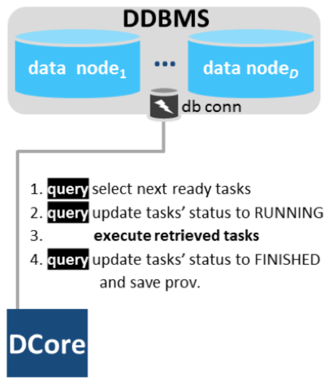
DDBMS
data node$_1$
data node$_D$
db conn
1. query select next ready tasks
2. query update tasks' status to RUNNING
3. execute retrieved tasks
4. query update tasks' status to FINISHED and save prov.
DCore

Figure 5. DDBMS-driven scheduling.

Distributed Data Design. The WQ is typically the largest data structure for the scheduler in terms of number of elements. A distributed database design requires a partitioning strategy that matches the desirable number of partitions, and the placement of the partitions [9]. To reduce load imbalance, SchalaDB distributes the WQ partitions across the data nodes. The number of partitions is equal to the number W of workers. Thus, each worker has its own WQ partition to improve data parallelism by using different memory spaces in parallel for each partition. Local processing is also improved because task lookup for each worker goes straight to its partition instead querying a large WQ. This also reduces race condition among workers, which otherwise would be competing for the entire WQ. Each worker w_i only accesses its own WQ_i partition using queries like "select/update the next ready tasks in the WQ where $partition = WQ_i$".

With respect to implementation, after trying multiple DDBMSs, we found that MySQL Cluster would be the best fit, since SchalaDB needs OLTP for scheduling and OLAP for runtime queries. In addition, it is open-source, scalable, can run fully in cluster memory, and implements ACID transactions. Further details about why MySQL Cluster was chosen over other options are discussed in [12]. In Figure 2, we show the centralized architecture in Chiron. In Figure 3, we show how we implemented SchalaDB in d-Chiron using MySQL Cluster. Figures 4 and 5 show a comparison of centralized and distributed task scheduling driven by a centralized and distributed DBMS, respectively. We can see that a DDBMS-driven scheduling eases scheduling and reduces the overhead caused by message passing between master and workers.

3. Contributions and Concluding Remarks

Therefore, we are not aware of a SWMS solution that both allows for analytical queries integrating domain data, provenance, and execution at runtime and is highly scalable. We proposed a decentralized MTC task scheduler, which is the core of an HPC system, using distributed data management techniques aiming at high performance for a SWMS. This is the first work that frequently attains over 80% of parallel efficiency running on a 1,008 cores cluster while managing workflow data in a same database available for runtime queries. Observing the tendency of the curves plotted in a comprehensive set of performance tests [12], we could see that the solution could still scale if we had access to an even larger cluster with at least few thousands of CPU cores. In addition to benchmarking workflows, we successfully ran two real workflow case studies: one in oil and gas and other in bioinformatics domains. Finally, we show that our implementation runs at least two orders of magnitude faster than the implementation that uses a centralized data management and scheduling [12,15]. Besides the performance gains, by using SchalaDB's scheduling, a complex part of the SWMS engine source code can be outsourced to a specialized system, *i.e.*, the DDBMS. This dissertation was developed in the context of a set of published works:

- The core ideas of SchalaDB and some results in d-Chiron were presented in the prestigious ACM/IEEE Supercomputing conference as a poster [15].
- An approach to analyze performance data integrated with domain dataflow and provenance was presented in [14]. We could quantify the performance bottlenecks that a centralized scheduling was causing in Chiron SWMS. It motivated this work and won the second-best paper award in the workshop. It also derived in [10], presented in a workshop held in conjunction with ACM/IEEE Supercomputing.
- A strategy to publish data stored in the wf-Database on the semantic web using ontology, RDF, and triple stores was presented in [2], which followed a linked data publication strategy presented in [11]. It ran for best poster award. It was derived from an undergraduate dissertation that I co-supervised.
- A DDBMS-driven approach for fault tolerance in SWMSs [5]. It is part of an undergraduate dissertation that I co-supervised.

- The results of the dissertation have contributed to a new direction of research related to data reduction in scientific workflows, presented in a workshop in conjunction with Supercomputing [8]. Even though it has additional work, developed after the dissertation, we consider it a derived result from the scalability of d-Chiron.

References

[1] Atkinson, M., Gesing, S., Montagnat, J., Taylor, I. Scientific workflows: past, present and future. *FGCS*, 75:216–227, 2017.

[2] Castro, R., Souza, R., Sousa, V.S., Ocaña, K.A.C.S., Oliveira, D. de, Mattoso, M. Uma abordagem para publicação de dados de proveniência de workflows científicos na web semântica. *Simpósio Brasileiro de Banco de Dados*, 1–6, 2015.

[3] Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Ferreira da Silva, R., et al. Pegasus, a workflow management system for science automation. *FGCS*, 46(C):17–35, 2015.

[4] GitHub. d-Chiron Repository. Available at: github.com/hpcdb/d-Chiron

[5] Miranda, P. *Um mecanismo de tolerância a falhas em execuções paralelas de workflows apoiadas por banco de dados*. BSc dissertation, UFRJ, 2015.

[6] Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M. An algebraic approach for data-centric scientific workflows. *PVLDB*, 4(12):1328–1339, 2011.

[7] Oliveira, D., Costa, F., Silva, V., Ocaña, K., Mattoso, M. Debugging scientific workflows with provenance: achievements and lessons learned. *Simpósio Brasileiro de Banco de Dados*, 1–10, 2014.

[8] Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M. SciCumulus: a lightweight cloud middleware to explore many task computing paradigm in scientific workflows. *IEEE Int. Conf. on Cloud Computing*, 378–385, 2010.

[9] Özsu, M.T., Valduriez, P. *Principles of distributed database systems*. 3 ed. New York, Springer, 2011.

[10] Silva, V., Neves, L., Souza, R., Coutinho, A.L.G.A., Oliveira, D. de, Mattoso, M. Integrating domain-data steering with code-profiling tools to debug data-intensive workflows. *WORKS*, 59–63, 2016.

[11] Souza, R., Cottrell, L., White, B., Campos, M.L., Mattoso, M. Linked open data publication strategies: Application in networking performance measurement data. *ASE International Conference on BigData/SocialCom/CyberSecurity*, 1–7, 2014.

[12] Souza, R. *Controlling the parallel execution of workflows relying on a distributed database*. MSc dissertation, COPPE/UFRJ, 2015.

[13] Souza, R., Silva, V., Coutinho, A.L.G.A., Valduriez, P., Mattoso, M. Online input data reduction in scientific workflows. *WORKS*, 44–53, 2016.

[14] Souza, R., Silva, V., Neves, L., De Oliveira, D., Mattoso, M. Monitoramento de desempenho usando dados de proveniência e de domínio durante a execução de aplicações científicas. *Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, 1–14, 2015.

[15] Souza, R., Silva, V., Oliveira, Daniel, Valduriez, P., Lima, A.A.B., Mattoso, M. Parallel execution of workflows driven by a distributed database management system. *Poster in IEEE/ACM Supercomputing*, 1–3, 2015.

[16] Wozniak, J.M., Armstrong, T.G., Wilde, M., Katz, D.S., Lusk, E., Foster, I.T. Swift/T: large-scale application composition via distributed-memory dataflow processing. *IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing*, 95–102, 2013.