Parallel Execution of Workflows Driven by a Distributed Database Management System

Renan Souza
Vítor Silva
Computer Science – COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
{renanfs,silva}@cos.ufrj.br

Daniel de Oliveira
Institute of Computing
Fluminense Federal University
Niterói, Brazil
danielcmo@ic.uff.br

Patrick Valduriez
Zenith Team, Inria and LIRMM
Montpellier, France
patrick.valduriez@inria.fr

Alexandre A. B. Lima
Marta Mattoso
Computer Science – COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
{assis,marta}@cos.ufrj.br

ABSTRACT

Scientific Workflow Management Systems (SWfMS) that execute large-scale simulations need to manage many task computing in high performance environments. With the scale of tasks and processing cores to be managed, SWfMS require efficient distributed data structures to manage data related to scheduling, data movement and provenance data gathering. Although related systems store these data in multiple log files, some existing approaches store them using a Database Management System (DBMS) at runtime, which provides powerful analytical capabilities, such as execution monitoring, anticipated result analyses, and user steering. Despite these advantages, approaches relying on a centralized DBMS introduce a point of contention, jeopardizing performance in large-scale executions. In this paper, we propose an architecture relying on a distributed DBMS to both manage the parallel execution of tasks and store those data at runtime. Our experiments show an efficiency of over 80% on 1,000 cores without abdicating the analytical capabilities at runtime.

1. INTRODUCTION

Large-scale scientific simulations are complex, involve huge amounts of data, and demand parallel execution on High Performance Computing (HPC) environments. These simulations frequently handle many program invocations, each of which is treated as a task – alluding to the Many Task Computing (MTC) paradigm [1] –, generating data to be consumed by a next task, forming a dataflow. Thus, they may be modeled as workflows and orchestrated by Scientific Workflow Management Systems (SWfMS). Keeping the information of which tasks will be scheduled to which machines composing the HPC environment as well as which data will be consumed by the tasks is essential for a parallel execution engine. SWfMS are also expected to collect provenance data, which represent both workflow specification and execution results [2]. Storing provenance data is essential for reproducibility, sharing, and knowledge reuse. Besides provenance and workflow execution data (for parallel execution management), SWfMS must manage domain-specific data, e.g. wave propagation velocities (seismic domain). These data may provide powerful insights when adequately correlated and analyzed. Altogether, SWfMS should manage three types of data, along the dataflow generation: (i) performance execution data – mainly related to MTC, (ii) provenance data, and (iii) domain data.

Typically, SWfMS manage these kinds of data by storing them in separate data structures or files. During execution, provenance data is usually registered in log files. Database Management Systems (DBMS) are made for managing data in large scale. They are specialized in providing advanced storage techniques, concurrent data access control, and query strategies with powerful analytical capabilities. Appending data onto log files is easier to implement and potentially delivers better performance than using DBMS. However, to allow for provenance queries, these log files have to be loaded into a DBMS, after the execution, disregarding analytical capabilities at runtime. It has been shown that using a DBMS at runtime delivers powerful analytical capabilities, such as execution monitoring, user steering [3], and anticipated results discovery, more evident in long executions [4].

The problem we want to tackle can be synthetized as a trade-off between runtime analytical capabilities and performance. Centralized DBMS efficiently manage a large amount of data, but introduce performance issues because they struggle when dealing with simultaneous requests – especially writes – from many clients, which is critical in large environments. A way to alleviate this is to use a single central node that accesses the DBMS and schedules tasks to the other worker nodes. Contention at the DBMS is alleviated, but it introduces a point of contention at the central node. Moreover, the central node needs to communicate with the worker nodes through message passing (e.g., MPI) for tasks scheduling, making the application code more complex.

To deliver good performance without abdicating analytical capabilities at runtime, we use a Distributed DBMS (DDBMS) to support both parallel execution management and provenance data storage. A central node for tasks scheduling is no longer needed; rather, such responsibility is transferred to the DDBMS, which is specialized in distributed concurrency control for multiple simultaneous requests [5]. All nodes can directly access the database to retrieve their tasks. Consequently, it reduces the application code complexity related to messages passing, which are, in great part, outsourced to a specialized system. Provenance data continues to be gathered and available for queries during execution (our main motivation), since DDBMS have the same abilities as centralized DBMS.

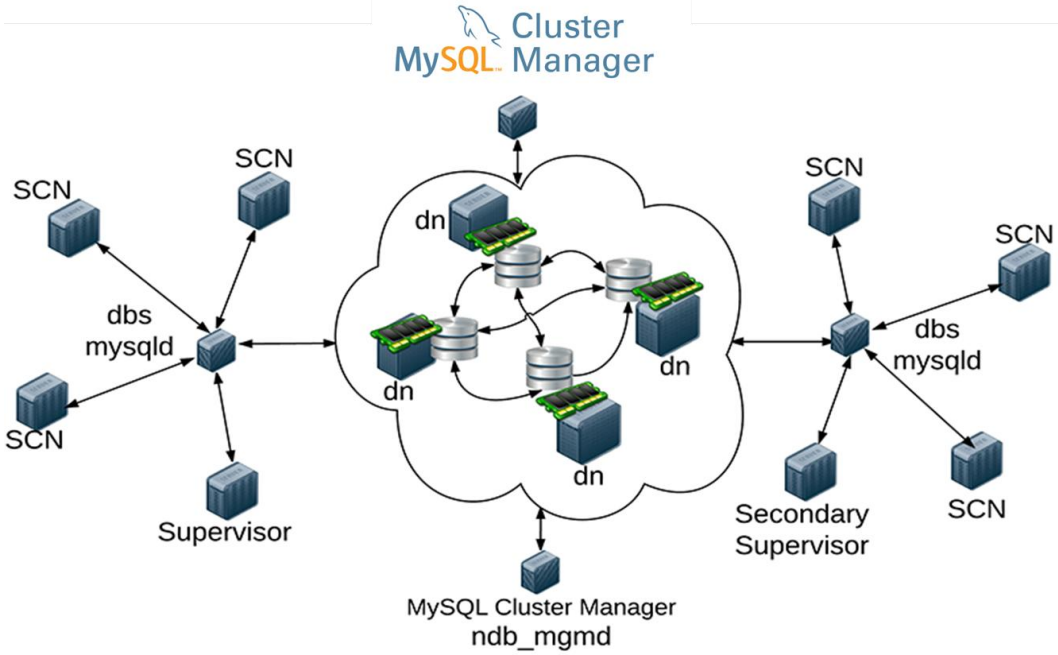

Figure 1 – This is d-SCC's architecture. The database is distributed into ***d*** data nodes (dn). A supervisor node is responsible for initial tasks distribution by inserting tasks in the database and for coordinating load balance and hardware failures. SCN are responsible for processing tasks. Database servers (dbs) act as ports through which nodes connect to the DDBMS. All slaves have same privileges for accessing the database and concurrency issues related to tasks scheduling are outsourced.

Figure 1 illustrates our architecture. The system that runs on top of it is called d-SCC. We have integrated the codes of Chiron [6] and SciCumulus [7], which are SWfMS for cluster and clouds, into a system named SCC. Since SCC continues to use a centralized DBMS to manage data execution, this work contributes with a distributed architecture, while keeping all the benefits from both SWfMS.

2. EXPERIMENTAL EVALUATION

The experimental evaluation was on a 1,008 cores cluster on Grid5000 [8]. We used d-SCC with MySQL Cluster 7.4.6, a distributed in-memory DBMS while running a synthetic workflow with three map/reduce activities. Evaluation results show: execution times obtained from 120 to 960 cores (Figure 2); scalability (Figure 3); and a comparison between SCC and d-SCC (Figure 4).

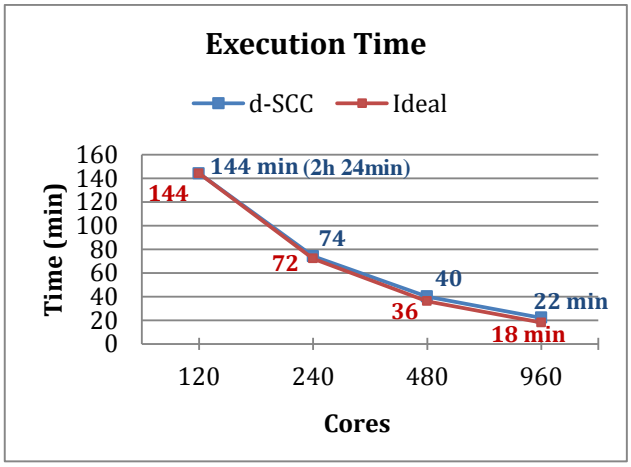

Figure 2 – The execution time of a workflow with 30k tasks (each of which costing 32s on average), varying the hardware configuration (120 to 960 cores). We achieve a very close to ideal execution time (only 4 minutes from the ideal on 960 cores). The theoretical sequential time is approximately 11.5 days.

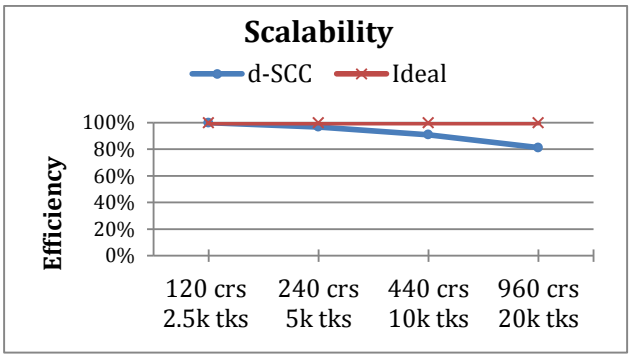

Figure 3 – Scalability when doubling both problem size and hardware configuration. We varied from 120 cores executing 2.5k tasks to 960 cores executing 20k tasks. In all configurations, we attained an efficiency of over 80%.

3. CONCLUSION

We presented a distributed architecture for workflow engines relying on a DDBMS to support both parallel execution management and provenance gathering at runtime. Our purpose is allowing for queries on provenance data, performance and domain data at runtime, so workflow data can be analyzed, improving interactivity between scientists and execution in a simulation run. Preliminary results on d-SCC show that relying on a DDBMS for tasks scheduling delivered efficiencies frequently over 80% and over 90% of gains in relation to SCC, which uses a centralized database and MPI for tasks scheduling. A fine-tuning on our architecture is planned as future work, to take further advantages of both the parallel hardware and the DDBMS.

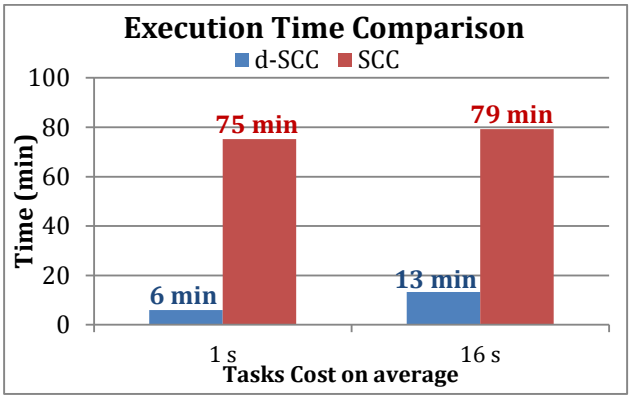

Figure 4 – Results for a synthetic workflow composed of 3 activities, 10k tasks per activity, and executed on 1008 cores. d-SCC attains significant gains (more than 90% for 1s tasks cost on average) in relation to the architecture that relies on a centralized DBMS (SCC).

4. ACKNOWLEDGMENTS

Work partially funded by CNPq, FAPERJ and Inria (MUSIC and HOSCAR projects) and performed (for P. Valduriez) in the context of the Computational Biology Institute (www.ibc-montpellier.fr). The experiments were carried out using the Grid'5000 testbed (https://www.grid5000.fr).

REFERENCES

[1] I. Raicu, I.T. Foster, and Y. Zhao. Many-Task Computing for Grids and Supercomputers. *IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08)*, 2008.

[2] S.B. Davidson and J. Freire. Provenance and Scientific Workflows: Challenges and Opportunities. *ACM SIGMOD International Conference on Management of Data*, 1345–1350, 2008.

[3] Mattoso, J. Dias, K. Ocaña, E. Ogasawara, F. Costa, F. Horta, V. Silva, and D. de Oliveira. Dynamic steering of HPC scientific workflows: A survey, *Future Generation Computer Systems*, v. 46, p. 100–113, 2015.

[4] J. Dias, G. Guerra, F. Rochinha, A.L.G.A. Coutinho, P. Valduriez, M. Mattoso. Data-centric iteration in dynamic workflows. *Future Generation Computer Systems*, v. 46, p. 114–126, 2015.

[5] M.T. Özsu, P. Valduriez, Principles of Distributed Database Systems, third ed., Springer, New York, 2011.

[6] E. Ogasawara, J. Dias, D. Oliveira, F. Porto, P. Valduriez, M. Mattoso, An algebraic approach for data-centric scientific workflows. *37th International Conference on Very Large Data Bases, PVLDB*, vol. 4(12), 1328–1339, 2011.

[7] D. Oliveira, E. Ogasawara, F. Baião, and M. Mattoso. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. *Proceedings of the 3rd International Conference on Cloud Computing*, 378–385, 2010.

[8] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, et al. Adding Virtualization Capabilities to the Grid'5000 Testbed. *Cloud Computing and Services Science*, I. Ivanov, M. Sinderen, F. Leymann, and T. Shan, eds., Springer International Publishing, 3–20, 2013.