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 tasks 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 specific files, some existing approaches store them using a Database Management System (DBMS), which provides powerful analytical capabilities, including execution monitoring, anticipated result analyses, and user steering, when available at runtime. 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 support controlling tasks parallelism and manage those data at runtime. Our experiments show an efficiency of over 80% on 1,008 cores without abdicating the analytical capabilities at runtime.

1. DDBMS TO MANAGE TASKS, PROVENANCE AND DOMAIN DATA

Large-scale scientific simulations 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, which forms a dataflow that can be modeled as a workflow to be orchestrated by a Scientific Workflow Management System (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 metadata on both workflow specification and execution results [2]. Storing provenance data is essential for reproducibility, sharing, analysis and knowledge reuse. Besides provenance and workflow execution data (for parallel execution management, *e.g.*, tasks status), SWfMS must manage domain-specific data, *e.g*. wave propagation velocities (seismic domain). 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 and very limited analysis at runtime [2,3]. During execution, performance execution data is registered in structures local to the MTC scheduler, while provenance data is usually registered in log files and loaded in Database Management Systems (DBMS) only when execution finishes. Conversely, it has been shown that using a DBMS to manage these data, jointly, at runtime, delivers powerful analytical capabilities, such as execution monitoring, user steering [4], and anticipated results discovery, more evident in long executions [5]. In addition, it avoids data redundancy between these data.

The problem we tackled can be synthetized as a trade-off between runtime analytical capabilities and performance. To deliver good performance without abdicating analytical capabilities at runtime, we use a Distributed DBMS (DDBMS) to support both parallel execution management data and provenance data storage.

Figure 1 shows a central node responsible for initial tasks distribution by inserting, in the database, the tasks with the identifier of each worker that will execute each task. After that, all workers know exactly which tasks to execute and retrieve them through queries to the DDBMS, which is specialized in distributed concurrency control for multiple simultaneous requests [6]. Consequently, there is no message passing between the central node and workers for tasks scheduling, which both improves performance and reduces the application code complexity, outsourcing distributed concurrency issues to a specialized system. Provenance data continues to be gathered and available for queries during execution (our main motivation).

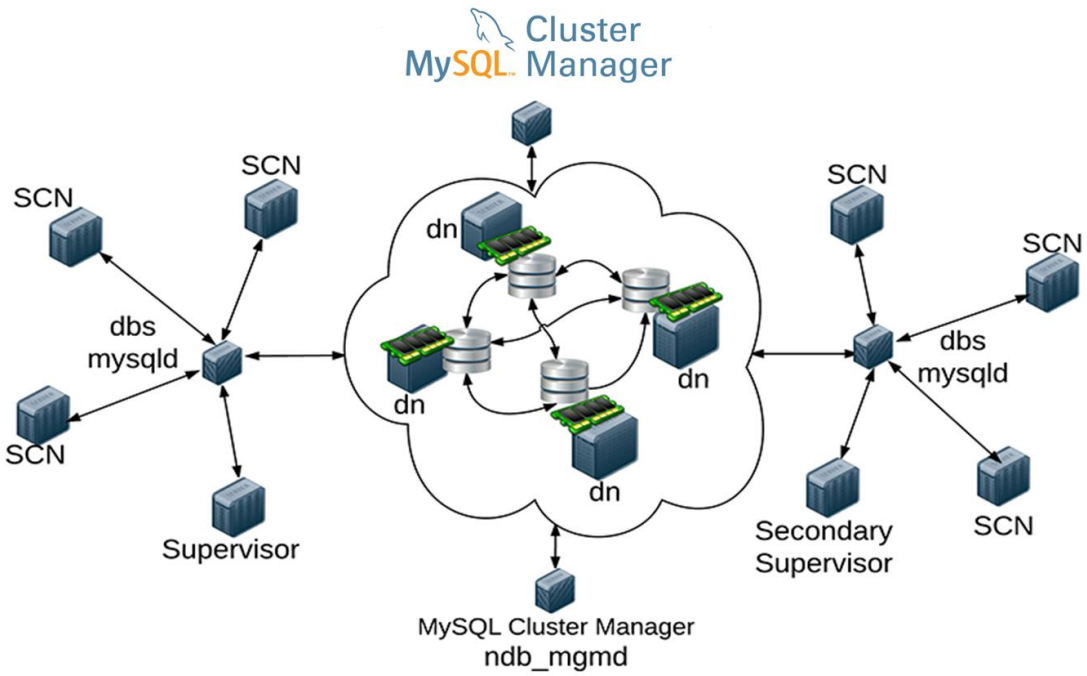

Figure 1 – This is d-SCC's implemented architecture.

The system that runs on top of it is called d-SCC (Figure 1). It integrates the codes of two SWfMS: SciCumulus [8] specialized in clouds and, Chiron [7] for clusters, into SCC (centralized control). To implement d-SCC we used MySQL Cluster 7.4.6, a distributed in-memory DBMS while running a synthetic workflow with three map/reduce activities. 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 that is outsourced.

The experimental evaluation was on a 1,008 cores cluster on Grid5000 [9]. 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).

Preliminary results on d-SCC show that relying on a DDBMS for task scheduling achieves efficiencies frequently over 80% and over 90% in relation to SCC, which uses a centralized database and MPI for tasks scheduling.

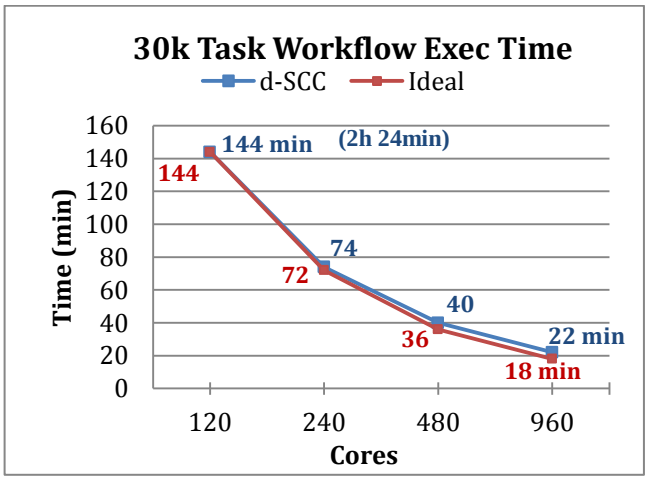

Figure 2 – Execution time with each task costing 32s on average is very close to ideal execution time (only 4 minutes from the ideal on 960 cores with relation to 120 cores first measure. The theoretical sequential time is approximately 11.5 days.

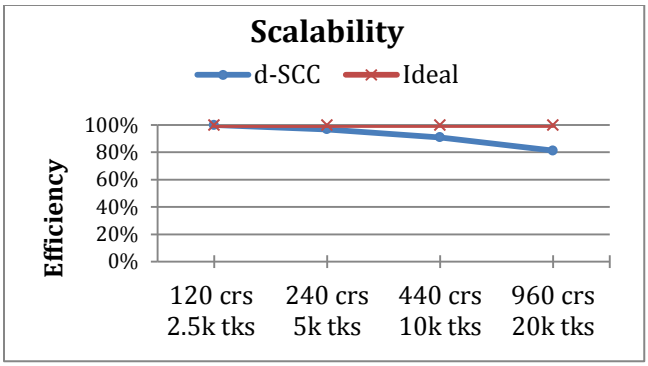

Figure 3 – Scalability when doubling both problem size (tasks) and hardware configuration (cores).

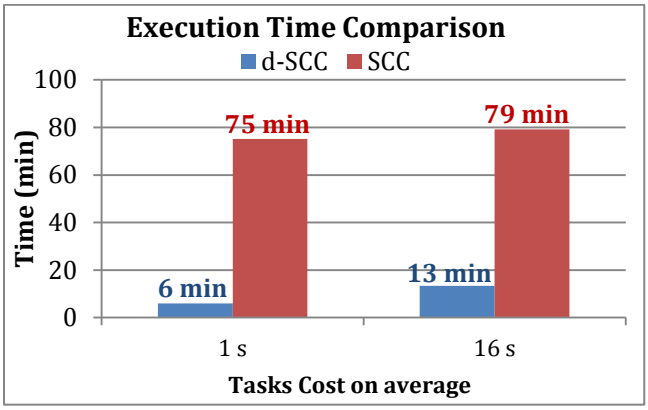

Figure 4 – Results for a synthetic workflow composed of 3 activities, 10k tasks per activity, and executed on 1008 cores. d-SCC attains significant gains (more than 90% for 1s as cost of task on average) compared to its centralized DBMS in SCC.

2. ACKNOWLEDGMENTS

This work is 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] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, et al. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015.

[3] J.M. Wozniak, T.G. Armstrong, M. Wilde, D.S. Katz, E. Lusk, and I.T. Foster. Swift/T: Large-Scale Application Composition via Distributed-Memory Dataflow Processing. *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 95–102, 2013.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, 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.