Parallel Execution of Workflows Driven by a Distributed Database Management System

Renan Souza[1], Vítor Silva[1], Daniel de Oliveira[2], Patrick Valduriez[3], Alexandre A. B. Lima[1], Marta Mattoso[1]
[1]Federal University of Rio de Janeiro, Brazil - [2]Fluminense Federal University, Brazil - [3]INRIA, France
{renanfs,silva,assis,marta}@cos.ufrj.br, danielcmo@ic.uff.br, patrick.valduriez@inria.fr

1. INTRODUCTION

SCENARIO

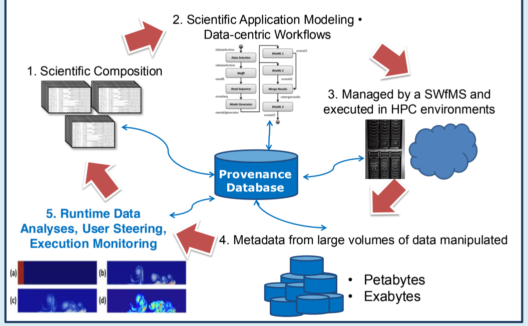
- Large-scale scientific simulations are complex • Huge amounts of data are manipulated • Executions take long time • HPC is required
- Simulations are composed of chained applications with a dataflow in between
- Modeled as data-centric workflows • Managed by a Scientific Workflow Management System (SWfMS)
- Large solution space is explored by varying parameters (Parameter Sweep)
- Each computation of parameters is represented by a task • Many Task Computing (MTC) parallelism

DATA MANAGEMENT

- Runtime data analyses are extremely important • Data management is complex
- Three types of data are expected to be managed by a SWfMS:
  - ✓ Provenance Data • Performance Data • Domain-specific Data

MOTIVATION

- Storing these data using a Database Management System (DBMS) at runtime enables powerful **runtime analytical capabilities**
  - ✓ Execution Monitoring • Anticipated result analyses • User steering



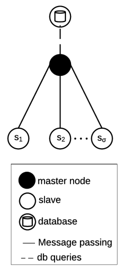
2. PROBLEM & OBJECTIVE

- SWfMS that use a DBMS at runtime rely on a Centralized DBMS, jeopardizing performance
- Trade-off: Analytical capabilities vs. Performance in a large scenario
- **Our objective is to deliver good performance without abdicating runtime analytical capabilities in large-scale simulations**

3. RESEARCH DESIGN & METHODOLOGY

Performance issues: Centralized Data Management

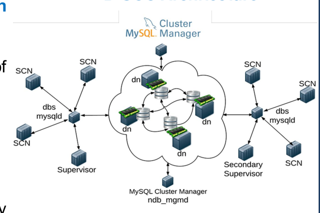
- C. DBMS struggles when dealing with simultaneous requests (mainly writes) from many clients
- A central (master) node is needed so it will be the only node able to access the database
- Contention at the DBMS is alleviated, but not at the master node
- The master is responsible for scheduling tasks among all slaves via message passing
- Application code becomes more complex for handling concurrency issues



SOLUTION: Parallel Execution Driven by a Distributed DBMS (DDBMS)

- Utilization of a DDBMS to support **both parallel execution management** and **runtime provenance data gathering**
- Specialized in distributed concurrency control in the presence of multiple simultaneous requests
- A central node for tasks scheduling is no longer necessary
- Such responsibility is transferred to the DDBMS
- Application code related to message passing is reduced and mostly **outsourced to a specialized system**
- Regarding provenance data gathering during execution, DDBMS have the same abilities as centralized DBMS, **maintaining analytical capabilities at runtime**

D-SCC Architecture

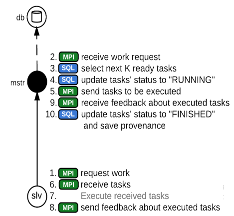
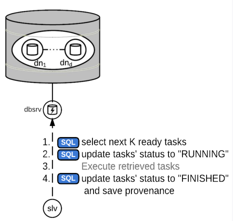


Composing Nodes:

- Supervisor & Secondary
- SCN – Worker machine
- Dbs (mysqld) –db server
- Dn – Data node
- MySQL Cluster Manager

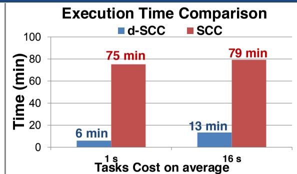
Scan it!

4. SCHEDULING COMPARISON

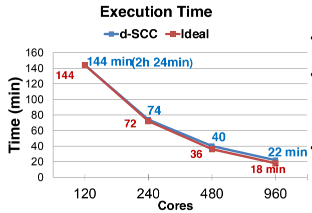
Relying on a Centralized DBMS + MPI	Relying on a DDBMS only
	
• DBMS being used to mainly store provenance • Application code needs message passing	• DDBMS being used to both support parallelism and store provenance • Application code is simpler; complexity is outsourced

5. EXPERIMENTAL EVALUATION

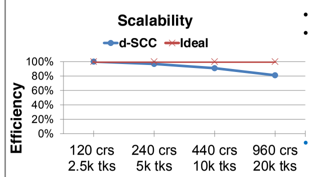
- We modified the architecture of SciCumulus (**SCC**), a SWfMS that uses a Centralized DBMS to store provenance at runtime
- MySQL Cluster, an in-memory DDBMS, was utilized
- We ran the experiments on Grid5000
- We call our solution as **d-SCC**.



- 30k tasks
- 1008 cores in parallel
- More than 90% of gains for 1s tasks cost on average in relation to SCC



- 30k tasks (32 s on average each)
- Very close to ideal execution time (4 minutes from the ideal on 960 cores).
- Theoretical sequential time is approximately 11.5 days.



- Scalability
- Varying 120 cores executing 2.5k tasks to 960 cores executing 20k tasks
- Efficiency of over 80% in all cases

6. CONCLUSIONS

- Utilization of an in-memory DDBMS to support parallel execution and runtime provenance gathering in workflow systems
- **Good performance** on a 1,000-cores cluster maintaining **enhanced analytical capabilities at runtime**
- Over **80% of efficiency** and 90% of gains comparing with an architecture that relies on a Centralized DBMS.

FUTURE WORK

- Fine-tuning of d-SCC exploring core memory
- Improve load balance and hardware failure features

REFERENCES

- 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.
- M. 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.
- 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.
- 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.

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). Experiments were carried out using the Grid'5000 testbed (https://www.grid5000.fr).