Provenance Query Patterns for Many-Task Scientific Computing

Luiz Gadelha, Marta Mattoso
Federal University of Rio de Janeiro

Michael Wilde, Ian Foster
University of Chicago/Argonne National Laboratory

3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP '11)
Heraklion, Crete, Greece

June 20, 2011

Introduction

- Problem definition: provenance modeling, gathering and querying for many-task computing (MTC).
  - Data model for MTC provenance (OPM specialization).
  - Identification of query patterns for MTC provenance.
  - Creating support for the identified query patterns.
  - Implemented in the Swift parallel scripting system.

Provenance Model

- Requirements:
  1. Gather consumption and production relationships between datasets and processes.
  2. Gather hierarchical relationships between datasets.
  3. Allow for the users to enrich their provenance records with annotations.
  4. Gather versioning information about scientific workflows and their component applications.
  5. Gather runtime information about external applications invoked from a Swift script.
  6. Provide a usable and useful query interface for provenance.

Provenance Model

- The following entities are part of this data model:
  - *Process*. Can take artifacts as input, perform some computation, and produce artifacts as output.
  - *Data set*. Are given by artifacts that are consumed or produced by processes.
  - *Application invocation*. A type of process that is given by an invocation of a component applications of a scientific workflow.
  - *Application execution*. Are given by execution attempts of an external application.
  - *Script run*. Refers to the execution (successful or unsuccessful) of a Swift script.
  - *Annotation*. A name-value pair associated with either a dataset, process, or workflow.

Provenance Query Patterns

Queries patterns identified in the Provenance Challenge series and in Swift's provenance system usage:

- *Entity Attribute* (EA). Attributes of an entity of the data model.
- *One-step Relationship* (R). Entities involved in a relationship of the data model.
- *Multiple-step Relationship* (R^*). Entities involved in the transitive closure of a relationship of the data model.
- *Lineage Graph Matching* (LGM). Similarity between lineage graphs.

Provenance Query Patterns

- *Run summary* (RS). Application specific attributes.
  - *Run resource-level performance* (RRP). Runtime behavior of scientific computations.
  - *Run science-level performance* (RSP). Input and output scientific parameters.
- Run comparisons (RCp). Comparisons between multiple runs with respect to some attribute.
- Run correlations (RCr). Correlation between multiple runs with respect to a set of attributes.

Provenance Query Patterns

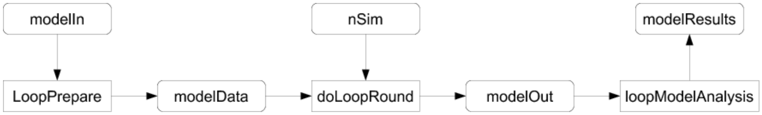
Table: Provenance Challenge Query Patterns.

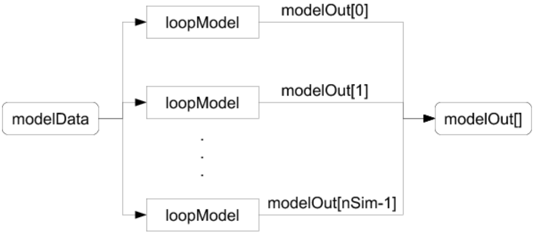
Pattern	PC1/PC2									PC3				PC3 (Optional Queries)														
	1	2	3	4	5	6	7	8	9	1	2	3	5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EA	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
R	×	×	×		×	×	×	×	×	×	×	×	×		×	×	×	×	×	×	×	×	×	×	×	×	×	×
R*	×	×	×			×	×			×	×	×	×		×		×				×	×			×	×	×	×
LGM							×								×													
RS	×	×	×							×	×	×	×	×		×	×						×	×	×	×	×	
RCp					×	×	×	×	×						×			×	×	×	×	×						×
RCr				×																								

Implementation

- Provenance extracted from Swift's log files and stored in a relational database.
- To abstract common provenance query patterns, we wrote SQL functions and stored procedures to hide the complexity of the database schema by encapsulating frequently used relational joins.
- RCp and RCr query patterns are abstracted with stored procedures:
  - compare_run($\langle$ list of parameters and annotation keys$\rangle$) returns a table with the values of the annotations and parameters per run.
- $R*$ query pattern can be abstracted with functions that use `WITH RECURSIVE` recursive queries.

Case Study: Open Protein Simulator





Case Study: Open Protein Simulator

What was the correlation between *root mean square distance* (RMSD) and the number of simulation (loopModel) steps for a given protein? (RCr pattern)

```
SELECT run_id, r.value as nSim, t.value as rmsd
FROM   compare_run_by_param('proteinId') as r
       INNER JOIN
       compare_run_by_param('nSim') as s USING (run_id)
       INNER JOIN
       compare_run_by_annot('rmsd') as t USING (run_id)
WHERE  r.value='TR567' and run.id LIKE 'psim.loops%';

             run_id                | nSim |  rmsd
-----------------------------------+------+---------
 psim.loops-20100604-2215-cdifsnb3 |  256 | 3.33123
 psim.loops-20100613-0125-keyyyc35 |  512 | 0.76274
 psim.loops-20100616-1512-h6q4g4ja | 1024 | 0.68426
 ...
```

Case Study: Open Protein Simulator

Common Table Expressions can be used to define functions supporting the R^* pattern:

```
CREATE OR REPLACE FUNCTION ancestors(varchar)
RETURNS SETOF varchar AS $$
  WITH RECURSIVE anc(ancestor,descendant) AS
  (
       SELECT parent AS ancestor, child AS descendant
       FROM   prov_graph
       WHERE child=$1
     UNION
       SELECT prov_graph.parent AS ancestor,
              anc.descendant AS descendant
       FROM   anc, prov_graph
       WHERE  anc.ancestor=prov_graph.child
  )
  SELECT ancestor FROM anc
$$ LANGUAGE SQL;
```

Where `prov_graph` is a database view that defines the edges of the provenance graphs stored in the database.

Case Study: Open Protein Simulator

An invocation of the previous function returns:

```
SELECT *
FROM   ancestors('dataset:20100618-0402-ia0bqb73:72000045');

                   ancestor
--------------------------------------------------
 execute:psim.loops-20100618-0402-qhm9ugg4:451006
 dataset:20100618-0402-ia0bqb73:72000039
 ...
```

Concluding Remarks

- We identified provenance query patterns from:
  - Provenance Challenge series.
  - Users of the Swift parallel scripting system.
- We implemented functions and stored procedures to support query patterns in Swift's provenance management system.
- Future work:
  - Compare how different data/storage models perform for provenance queries.
  - Provenance query language.

Concluding Remarks

Thank you!