Performance Metrics and Ontology for Describing Performance Data of Grid Workflows

Hong-Linh Truong, Thomas Fahringer, Francesco Nerieri  
Distributed and Parallel Systems Group  
Institute for Computer Science, University of Innsbruck  
{truong,tf,nero}@dps.uibk.ac.at

Schahram Dustdar  
Information Systems Institute, Vienna University of Technology  
dustdar@infosys.tuwien.ac.at

http://dps.uibk.ac.at/projects/pma

1st Performability Workshop, CCGrid05, Cardiff 09 May, 2005
Outline

- Motivation
- Grid workflows and workflow execution model
- Performance metrics of Grid workflows
- WfPerfOnto: Ontology for describing performance data of Grid workflows
- Utilizing WfPerfOnto
- Conclusion and Future work
Motivation

- Lack of comprehensive study of useful performance metrics for Grid workflows
  - A few metrics are studied and supported
  - Most of metrics are being limited to the activity (task) level.
    *study performance metrics at multiple levels of abstraction*

- Describing and sharing performance data of Grid workflows
  - Highly heterogeneous, inter-related and dynamic
  - Inter-organizational
  - Multiple types of performance and monitoring data provided by various tools
  - an ontology for performance data
    - Can be used to describe concepts associated with workflow executions
    - Will facilitate the performance data sharing
Hierarchical Structure View of a Workflow

Workflow

Workflow Construct n

Activity m

Invoked Application m

Code Region 1

Code Region ...

Code Region q

<parallel>

<activity name="mProject2">
  <executable name="/home/truong/mProject2"/>
</activity>

<activity name="mProject1">
  <executable name="/home/truong/mProject1"/>
</activity>

</parallel>

mProject1.c

int main() {
  A();
  while () {
    ...
  }
}

Performance Metrics and Ontology for Describing Performance Data of Grid Workflows, CCGrid 05
Workflow Execution Model (Simplified)

- **Workflow execution**
  - Spanning multiple Grid sites
  - Highly inter-organizational, inter-related and dynamic

- **Multiple levels of job scheduling**
  - At workflow execution engine (part of WfMS)
  - At Grid sites
Interesting performance metrics associated with multiple levels of abstraction

- Metrics can be used in workflow composition, for comparing different invoked applications of a single activity, etc.

Five levels of abstraction

- Code region, Invoked application
- Activity, Workflow construct, Workflow

Performance metrics of a lower level can be used to construct similar metrics for the immediate higher-level

- By using aggregate operator
- Based on metric definition and structure of workflows
### Performance Metrics at Code Region Level

<table>
<thead>
<tr>
<th>Category</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>ElapsedTime, UserCPUTime, SystemCPUTime, SerialTime, EncodingTime</td>
</tr>
<tr>
<td>Counter</td>
<td>L2_TCM, L2_TCA, etc., (hardware counters)</td>
</tr>
<tr>
<td></td>
<td>NCalls, NSubs, RecvMsgCount, SendMsgCount</td>
</tr>
<tr>
<td>Synchronization</td>
<td>CondSynTime, ExclSynTime</td>
</tr>
<tr>
<td>Data Movement</td>
<td>TotalCommTime, TotalTransSize</td>
</tr>
<tr>
<td>Ratio</td>
<td>MeanElapsedTime, CommPerComp, MeanTransRate, MeanTranSize</td>
</tr>
<tr>
<td></td>
<td>CachMissRatio, MFLOPS, etc.</td>
</tr>
<tr>
<td>Temporal overhead</td>
<td>temporal overhead of parallel code regions</td>
</tr>
</tbody>
</table>

- Most existing conventional performance tools provide these metrics
- Existing workflow monitoring and analysis tools normally do not
- Challenging issues
  - Integrate conventional performance monitoring tools into workflow monitoring tools
Most metrics can be constructed from metrics at code region level

<table>
<thead>
<tr>
<th>Category</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>ElapsedTime</td>
</tr>
<tr>
<td></td>
<td>FailedTime</td>
</tr>
<tr>
<td>Counter</td>
<td>NCallFailed</td>
</tr>
<tr>
<td></td>
<td>NCalls</td>
</tr>
<tr>
<td>Ratio</td>
<td>FailedFreq</td>
</tr>
<tr>
<td>Performance Improvement</td>
<td>SpeedupFactor</td>
</tr>
</tbody>
</table>
### Performance Metrics at Activity Level

<table>
<thead>
<tr>
<th>Category</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>ElapsedTime, ProcessingTime, QueuingTime, SuspendingTime</td>
</tr>
<tr>
<td></td>
<td>FailedTime, SharedResTime</td>
</tr>
<tr>
<td>Counter</td>
<td>RedandantActivity, Nlteration, PathSelectionRatio, ResUtilization</td>
</tr>
<tr>
<td>Ratio</td>
<td>Throughput, MeanTimePerState, TransRate</td>
</tr>
<tr>
<td>Synchronization</td>
<td>SynDelay, ExecDelay</td>
</tr>
<tr>
<td>Performance Improvement</td>
<td>SlowdownFactor</td>
</tr>
</tbody>
</table>

- Metrics can be defined for both activity and activity instance
- Aggregate metrics of an activity can be defined based on its instances and the execution of instances at runtime
- **Challenging problems**
  - How to monitor and correlate metrics when a resource is shared among applications
## Performance Metrics at Workflow Construct Level

<table>
<thead>
<tr>
<th>Category</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>ElapsedTime, ProcessingTime</td>
</tr>
<tr>
<td>Counter</td>
<td>RedundantActivity,</td>
</tr>
<tr>
<td></td>
<td>NIteration, PathSelectionRatio, ResUtilization</td>
</tr>
<tr>
<td>Load balancing</td>
<td>LoadIm (Load imbalance)</td>
</tr>
<tr>
<td>Performance Improvement</td>
<td>SpeedupFactor</td>
</tr>
<tr>
<td>Resource</td>
<td>RedundantProcessing</td>
</tr>
</tbody>
</table>

- Aggregate metrics of a workflow construct/workflow construct instance are defined based on the structure of the construct. E.g.,
  - LoadIm (load imbalance) is for parallel construct
  - ElapsedTime/ProcessingTime is defined based on critical path
# Performance Metrics at Workflow Level

<table>
<thead>
<tr>
<th>Category</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>ElapsedTime, ProcessingTime, ParTime, SeqTime</td>
</tr>
<tr>
<td>Ratio</td>
<td>QueuingRatio, MeanProcessingTime, MeanQueuingTime, ResUtilization</td>
</tr>
<tr>
<td>Correlation</td>
<td>NAPerRes, ProcInRes, LoadImRes</td>
</tr>
<tr>
<td>Performance Improvement</td>
<td>Speedup</td>
</tr>
</tbody>
</table>
Performance Metrics Ontology

- **WfMetricOnto**
  - OWL-based performance metrics ontology

- **Metrics ontology**
  - Specifies which performance metrics a tool can provide
  - Simplifies the access to performance metrics provided by various tools
Monitoring and Measuring Performance Metrics

- Performance monitoring and analysis tools
  - Operate at multiple levels
  - Correlate performance metrics from multiple levels

- Middleware and application instrumentation
  - Instrument execution engine of WfMS
    - Execution engine can be distributed or centralized
  - Instrument applications
    - Distributed, spanning multiple Grid sites

- Challenging problems: Performance tool and data complexity
  - Integrate multiple performance monitoring tools executed on multiple Grid sites
  - Integrate performance data produced by various tools
Ontology Describing Performance Data of Grid Workflows

**Objectives**
- Understanding basic concepts associated with performance data of Grid workflows
- Performance data integration for Grid workflows
- Towards distributed/intelligent performance analysis

**WfPerfOnto (Ontology describing Performance data of Grid Workflows)**
- Basic concepts
  - Concepts reflects the hierarchical view of a workflow
  - Static and dynamic performance and monitoring data of workflow
- Relationships
  - Static and dynamic relationships among concepts
Ontology for Describing Performance Data of Grid Workflows
Utilizing WfPerfOnto

- Describing Performance Data and Data Integration
  - Different monitoring and analysis tools can store/export performance data in/to ontological representation
  - High-level search and retrieval of performance data

- Knowledge base performance data of Grid workflows
  - Utilized by high-level tools such as schedulers, workflow composition tools, etc.
  - Used to re(discover) workflow patterns, interactions in workflows, to check correct execution, etc.

- Distributed Performance Analysis
  - Performance analysis requests can be built based on WfPerfOnto
Utilizing WfPerfOnto: Describing Performance Data

```xml
<rdf:Description rdf:about="http://dps.uibk.ac.at/wfperfonto#mImgtbl21">
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#ElapsedTime78"/>
  <rdf:type rdf:resource="http://dps.uibk.ac.at/wfperfonto#ActivityInstance"/>
  <wfperfonto:instanceName>mImgtbl21</wfperfonto:instanceName>
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#QueuingTime80"/>
  <wfperfonto:ofActivity rdf:resource="http://dps.uibk.ac.at/wfperfonto#mImgtbl2"/>
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#ProcessingTime79"/>
</rdf:Description>
```
Utilizing WfPerfOnto: Checking Correct Execution

<rdf:Description rdf:about="http://dps.uibk.ac.at/wfperfonto#Seq4ForkJoin5">
  <wfperfonto:hasActivityInstance rdf:resource="http://dps.uibk.ac.at/wfperfonto#tRawImage4"/>
  <wfperfonto:hasActivityInstance rdf:resource="http://dps.uibk.ac.at/wfperfonto#tProjectedImage4"/>
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#ElapsedTime57"/>
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#QueuingTime59"/>
  <wfperfonto:hasActivityInstance rdf:resource="http://dps.uibk.ac.at/wfperfonto#mProject14"/>
  <wfperfonto:hasActivityInstance rdf:resource="http://dps.uibk.ac.at/wfperfonto#mImgthb14"/>
  <wfperfonto:ofWorkflowConstruct rdf:resource="http://dps.uibk.ac.at/wfperfonto#WorkflowConstructInstance"/>
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#ProcessingTime58"/>
  <wfperfonto:hasPerfMetric rdf:resource="http://dps.uibk.ac.at/wfperfonto#SeqForkJoin"/>
  <wfperfonto:instanceName>Seq4ForkJoin5</wfperfonto:instanceName>
</rdf:Description>
Utilizing WfPerfOnto: Distributed Performance Analysis

DIPAS
- Grid analysis agent
- Grid analysis agent
- Grid analysis agent

GOM

Monitoring Service

Resources Applications
Utilizing WfPerfOnto: Analysis Request

Requests based on WfPerfOnto

Analysis agent

Ontological data

Monitoring agent

Grid analysis agent

To the Monitoring Service
Conclusion and Future Work

- Performance metrics of Grid workflows that characterize the performance and dependability of Grid workflows; metrics associated with multiple levels of abstraction

- Ontology describing performance data of Grid workflows

Current implementation

- OWL-based ontologies, Jena toolkit for processing ontology-related task
- Store and export performance data in/to WfPerfOnto representation

Future work

- Extend and revise performance metrics and WfPerfOnto
- Distributed performance analysis
- Reasoning performance data

Shared conceptualization → community work?