An Ontology-based Approach to Performance Analysis, Data Sharing and Tools Integration in Grids*

Hong-Linh Truong[‡], Thomas Fahringer[§]

[‡]Institute for Software Science, University of Vienna
Liechtensteinstr. 22, A-1090 Vienna, Austria
truong@par.univie.ac.at

[§]Institute for Computer Science, University of Innsbruck
Technikerstr. 25/7, A-6020 Innsbruck, Austria
Thomas.Fahringer@uibk.ac.at

Last revised: February 9, 2004

Abstract

In this paper, we propose a new approach to performance analysis, data sharing and tools integration in Grids that is based on ontology. We devise a novel ontology for describing the semantics of monitoring and performance data that can be used by performance monitoring and measurement tools. We introduce an architecture for an ontology-based model for performance analysis, data sharing and tools integration. At the core of this architecture is a Grid service which offers facilities for other services to archive and access ontology models along with collected performance data, and to conduct searches and perform reasoning on that data. We also discuss how ontology can be used in automatic performance analysis. Using an approach based on ontology, performance data will be easily shared and processed by automated tools, services and human users, thus helping to leverage the data sharing and tools integration, and increasing the degree of automation of performance analysis.

Key words: Performance analysis, performance data model, Grid, ontologies.

1 Introduction

The recent emerging Grid computing raises many challenges in the domain of performance analysis. One of these challenges is how to understand and utilize performance data where the data is diversely collected and no central component manages and provides semantics of the data. Performance monitoring and analysis in Grids differ from that in conventional parallel systems in terms of no single tool providing performance data for all Grid sites and the need of conducting monitoring, measurement and analysis across multiple Grid sites at the same time. Normally users run their applications in multiple Grid sites, each is equipped with different computing capabilities, platforms, libraries that require various tools to conduct performance monitoring and measurement. Without the central component, performance monitoring and measurement tools have to provide a means for seamlessly utilizing the data they collect

*A short version of this report is published in Proceedings of International Conference on Computational Science 2004 (ICCS 2004), Krakow, Poland, June 7-9, 2004.

and provide, because many tools and services atop them need the data for specific purposes such as performance analysis, scheduling and resource matching. Current Grid performance tools focus on monitoring and measurement, but neglect data sharing and tools integration.

We take a new direction on describing the semantics of performance data and establishing performance data sharing and tools integration by investigating the use of *ontology* in performance analysis domain. Basically, ontologies provide a *shared and common* understanding of a domain that can be communicated between people and heterogeneous and widely spread application systems; ontology is developed to facilitate *knowledge sharing and reuse*. Based on sharable and extensible ontologies in the domain of performance analysis, an analysis tool, service or user is able to access multiple sources of performance and monitoring data provided by a variety of performance monitoring and measurement tools, understanding the data and making use of that data. With the expressive power provided by ontology which can describe concepts, resources in sufficient detail, supporting automated performance analysis will also be enhanced. In this paper, we describe a proposed ontology for performance data that defines semantics of performance data. We introduce an architecture for an ontology-based model for performance analysis, data sharing and tools integration. At the core of this architecture is a Grid service of ontology-based performance data repository which archives and provides ontology models with accompanied instance performance data.

The rest of this paper is organized as follows: Section 2 discusses the motivation. In Section 3 we present our proposed ontology for describing performance data. We describe an architecture for an ontology-based model for performance analysis, data sharing and tools integration in Section 4. In Section 5, we outline how ontology can be applied in automatic performance analysis. Section 6 overviews our prototype implementation. The related work is discussed in Section 7 followed by the Conclusion and Future work in Section 8.

2 Motivation

Most monitoring and measurement tools collect raw data (e.g. performance metrics) of objects monitored but they do not directly model and clarify the relevant aspects of these objects. As a result, the collected data lacks semantics and is difficultly correlated with the appropriate domain knowledge, and only skilled user is able to interpret the data provided. The software program hardly understands that data, thus not fostering to automatically detect, correct and predict behavior of applications and computing systems at runtime. Lack of detailing model of resources monitored also prevents moving performance analyzers as close to the source monitored as possible. As a result, the performance analysis still is very centralized even though the monitoring and measurement are quite distributed in Grid.

Currently, several data representations with different capabilities and expressiveness, e.g. XML, XML schema and relational database schema, are employed by Grid performance monitoring and measurement tools. However, little effort has been done to standardize the semantics of performance data as well as the way performance tools collaborate. In Grids, data is diversely collected and no central component manages and provides its semantics. Each Grid site may be equipped with its own performance monitoring and measurement tool. Thus, the end user or the high-level tool in Grids has to interact with a variety of tools offering monitoring and measurement service. Performance monitoring and measurement tools should not simply offer well-defined operations for other services calling them, e.g. based on Open Grid Services Architecture (OGSA) [7], but they have to provide a means for adding semantics to the data as Grid users and services require seamless integration and utilization of (performance) data provided by different tools.

Existing approaches on performance data sharing and tools integration that mostly focus on building wrapper libraries for directly converting data between different formats, making data available in relational database with specific data schema, or exporting data into XML, have several limitations. For example, building a wrapper requires high cost of implementation and maintenance; wrappers convert data between representations but not always between semantics. Although XML and XML schemas are sufficient for exchanging data between parties that have agreed in advance on definitions, its use and meaning, they mostly are suitable for one-to-one communication and impose no semantic constraints on the meaning of the data. Everyone can create his own XML vocabularies with his own definitions for describing his data. However, such vocabularies and definitions are not sharable and do not establish a common understanding about the data, thus preventing semantic interoperability between various parties which is an important issue that Grid monitoring and measurement tools have to support. Utilizing relational databases to store performance data [27, 25] makes data more sharable and accessible. However, data models represented in relational database are still very tool-specific and inextensible. Moreover, integrating different relational tables by using SQL join would generate nonsense information as relational database checks the data type, instead of the semantics, of the data. Notably, XML and relational database schemas do not explicitly express meanings of data they encode. Since all above-mentioned techniques do not provide enough capability to express the semantics of performance data and to support tools integration, they might not be applicable in Grids due to the autonomy and diversity of performance monitoring and measurement tools.

We investigate whether the use of ontology can help to solve the above-mentioned issues. Ontologies are a popular research topic in various communities such as knowledge engineering, natural language processing, cooperative information systems. An ontology is a formal, explicit specification of a shared conceptualization [11]. An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept. There are several reasons that ontology should be used [19]. One of key features of ontology is that it provides a shared and common understanding of some domains that can be communicated between people and application systems. Another feature is that a set of ontology statements by itself can allow to conclude another facts, e.g. via description logics [2], while that can not be achieved with XML or database schema. To employ XML or relational database to generate new data, we have to embed additional knowledge in some procedural code, while that knowledge is explicitly stated in ontology.

There are many ways in using ontology for addressing the above-mentioned issues. Firstly, ontology can be used to directly *describe and model the data* collected, thus allowing to share a common understanding of performance data and easily correlating the data with the knowledge domain. Secondly, ontology can be used to *define mappings between different representations* employed by different Grid monitoring and measurement tools. This would allow a high-level service to transparently access different types of data in a homogeneous way. This paper works on the first direction. By describing and modeling performance data in ontological representation, the performance data is made smarter, tool-independent.

3 PERFONTO: Ontology for Describing Performance Data

While initial work on using ontology for system and network management has been introduced, e.g in [5], to date we are not aware any ontology for describing performance data of applications in the field of performance analysis. Our starting point is that we try to propose an ontology for describing monitoring and performance data of both applications and systems. In this section, we describe PERFONTO[1], an ontology used to describe and represent performance data.

3.1 Selecting Ontology Language

Although various languages can be used to represent ontology such as RDFS (Resource Description Framework Schema) [22], DAML (DARPA Agent Markup Language) [15], OIL (Ontology Inference Layer) [17], DAML+OIL [4], we choose OWL (Web Ontology

[1]PERFONTO is a shorthand for **ONTO**logy for **PERF**ormance data

Language) [20] for developing PERFONTO due to several reasons. OWL is developed as a vocabulary extension of RDF (Resource Description Framework) [21] and is derived from DAML+OIL. OWL has the ability to describe classes in more interesting and complex ways than RDFS. For example, in RDFS, properties can be related via a property hierarchy. OWL extends this by allowing properties to be denoted as transitive, symmetric or functional, and allow to declare one property to be the inverse of another. OWL also makes a distinction between two main categories of properties: *Data Property* and *Object Property*. Data properties have data-values (a.k.a literals in RDF terminology) as their range whereas object properties have individuals. OWL somehow is similar to DAML+OIL because the W3C's Web Ontology Working Group, who designed OWL, took DAML+OIL as their starting point. OWL is currently being standardized by W3C.

OWL takes object-oriented approach, with the structure of the domain being described as a set of definitions of *classes* and *properties*. An ontology consists of a set of *axioms* including *class axioms* and *property axioms* that assert subsumption relationship between classes and properties. Class axioms typically contain additional components that specify necessary and/or sufficient characteristics of a class, e.g. subclass, equivalent class whereas a property axiom defines (additional) characteristics of a property such as range, domain, relations to other properties.

3.2 PERFONTO Design

PERFONTO comprises two parts that describe *experiment-related concept* and *resource-related concept*. Here we briefly discuss main classes in current version of PERFONTO.

Experiment-related concept describes experiments and their associated performance data of applications. The structure of the concept is described as a set of definitions of *classes* and *properties*. Figure 1 demonstrates several classes and a few properties of experiment-related concept in PERFONTO. (The figure does not contain all formalisms to represent PERFONTO. It is only for illustrative purpose.) **Application** describes information about the application. **Version** describes information about versions of an application. The property *ofApplication* of a version identifies the application to which the version belongs. (The property *hasVersion*, which is an inverse of *ofApplication*, of an application describes versions of the application). **SourceFile** describes source file of a version. **CodeRegion** describes a static (instrumented) code region. Code regions are classified into subclasses that are programming paradigm-dependent, e.g. message passing code region described by **MPCodeRegion**, shared memory code regions described by **SMCodeRegion**, and paradigm-independent, e.g. loops, arbitrary code regions; a sequential code region is described by **SeqCodeRegion**. **Experiment** describes an experiment which refers to a sequential or parallel execution of a program. The relationship between an experiment with a program version is described by property *ofVersion*. **RegionInstance** describes a region instance which is an execution of a static (instrumented) code region at runtime. A code region instance is associated with a processing unit (property *inProcessingUnit*) and has events (*hasEvent*) and subregion instances (*hasChildRI*). A processing unit, represented by class **ProcessingUnit**, describes the context in which the code region is executed; the context includes information about grid site, compute node, process, thread. **RegionSummary** describes the summary of code region instances of a static (instrumented) code region in a processing unit. A region summary has performance metrics (*hasMetric*) and subregion summaries (*hasChildRS*). **PerformanceMetric** describes a performance metric, each metric has a name and value (*hasMetricName, hasMetricValue*). **Event** describes an event record. Subclasses of Event are *EnterEvent* and *ExitEvent* that describe enter and exit event of a region instance, respectively. An event happens at a time (*atEventTime*) and has event attributes (*hasEventAttr*). **EventAttribute** describes an attribute of an event which has an attribute name and value (*hasAttrName, hasAttrValue*). An excerpt of OWL for *RegionSummary* is shown in Figure 2.

Resource-related concept describes static, benchmarked, and dynamic (performance) information of computing and network systems. In the current ver-

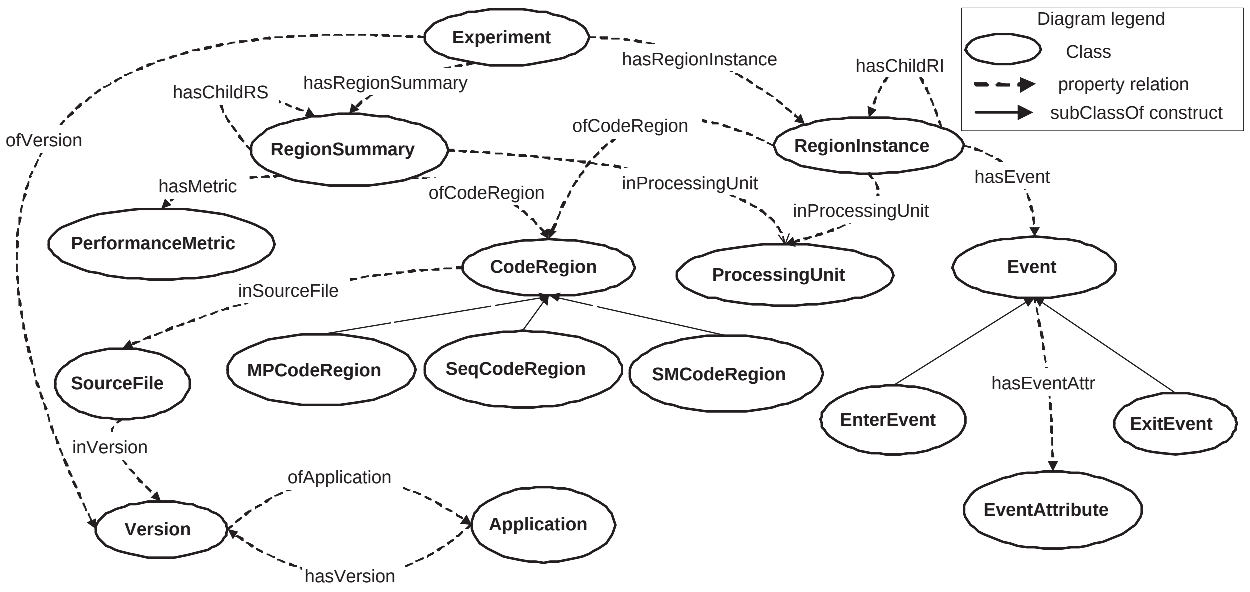


Figure 1: Illustrative classes and properties of experiment-related concept. *subClassOf* construct allows to define a class as a subclass of another class. Property relation represents the relation between classes.

sion, resource-related concept provides classes to describe static and benchmarked data of computing and network resources. For example, **Site** describes information of (grid) computing site. **Cluster** describes a set of physical machines (compute nodes). Cluster has subclasses such as *SMPCluster* represented a cluster of SMP. **ComputeNode** describes information about physical machine. ComputeNode has subclasses, e.g. *SMPComputeNode* represented an SMP machine. **Network** describes an available network. Subclasses of Network can be *EthernetNetwork*, *MyrinetNetwork*, etc. **NodeSharedMemoryPerf** describes performance characteristics of shared memory operations of a compute node. **NetworkMPColPef** and **NetworkMPP2PPerf** describe performance characteristics of message passing collective and point-to-point operations of a network, respectively. Other classes describing performance of high-level protocols (e.g. HTTP, SOAP) are also being proposed.

The proposed ontology is largely based on our previous work on developing data schema for expressing experiment data in relational database [27] and based on on APART experiment model [6]. The development of PERFONTO should be considered as the investigation of using ontology for describing performance data, not establishing a standard for all tools. However, one of the main key advantages of ontology is that different ontologies can be reconciled [18]. Therefore, one can employ or extend PERFONTO, others may develop their own ontologies. Finally, proposed ontologies could be merged.

4 An Architecture for An Ontology-based Model for Performance Analysis, Data Sharing and Tools Integration

Figure 3 presents a three layers architecture for an ontology-based model for performance analysis, data sharing and tools integration in Grids. At the core

```
<owl:Class rdf:ID="RegionSummary">
  <rdfs:label>RegionSummary</rdfs:label><rdfs:comment>RegionSummary Class</rdfs:comment>
</owl:Class>
<owl:ObjectProperty rdf:ID="inProcessingUnit">
  <rdfs:label>inProcessingUnit</rdfs:label><rdfs:comment>processing unit</rdfs:comment>
  <rdfs:domain rdf:resource="#RegionSummary"/>
  <rdfs:range rdf:resource="#ProcessingUnit"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ofCodeRegion">
  <rdfs:label>ofCodeRegion</rdfs:label> <rdfs:comment>code region</rdfs:comment>
  <rdfs:range rdf:resource="#CodeRegion"/>
  <rdfs:domain rdf:resource="#RegionSummary"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasChildRS">
  <rdfs:label>hasChildRS</rdfs:label> <rdfs:comment>child region summary</rdfs:comment>
  <rdfs:domain rdf:resource="#RegionSummary"/>
  <rdfs:range rdf:resource="#RegionSummary"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasMetric">
  <rdfs:label>hasMetric</rdfs:label> <rdfs:comment>performance metric</rdfs:comment>
  <rdfs:range rdf:resource="#PerformanceMetric"/>
  <rdfs:domain rdf:resource="#RegionSummary"/>
</owl:ObjectProperty>
```

Figure 2: Description of *RegionSummary* has four object properties namely *inProcessingUnit, ofCodeRegion, hasChildRS, hasMetric* that specify associated processing unit, code region, subregion summary and performance metric of the region summary, respectively.

of this architecture is a performance data repository service which includes:

- *PERFONTO* is ontology for representing performance data discussed in Section 3.
- *Ontological database* is a relational database which is used to hold ontologies (e.g PERFONTO) and performance data (instance data).
- *ONTO APIs* are interfaces used to store and access data in ontological database.
- *Query Engine* provides searching and querying functions on ontological data.
- *Inference Engine* provides reasoning facilities to infer, discover and extract knowledge from ontological performance data.

The performance data repository service is designed as a Grid service. The *Performance Data Collector* of performance monitoring and measurement tools can store collected data (instance data) along with corresponding ontology model (e.g. PERFONTO) into the ontological database. Via service operations, any clients needed performance data such as performance analysis tools, schedulers or users can easily request the ontology model and then retrieve instance data from ontological database. The key difference from approaches of using XML or relational database is that performance data is either described by a common ontology (e.g. PERFONTO) or by a tool-defined ontology, thus, with the presence of ontology model, these clients can easily understand and automatically process retrieved data. Via *Per-*

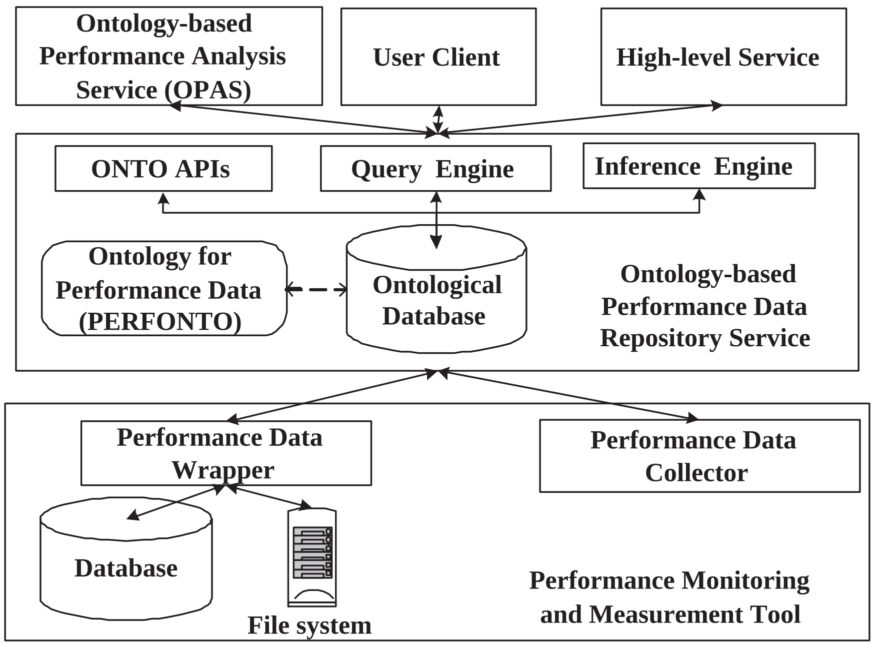

Ontology-based Performance Analysis Service (OPAS)
User Client
High-level Service
ONTO APIs
Query Engine
Inference Engine
Ontology for Performance Data (PERFONTO)
Ontological Database
Ontology-based Performance Data Repository Service
Performance Data Wrapper
Performance Data Collector
Database
File system
Performance Monitoring and Measurement Tool

Figure 3: Three layers architecture for an ontology-based model for performance analysis, data sharing and tools integration.

formance Data Wrapper, data in tool-defined non-ontology format also can be extracted an transformed into ontological representation and vice versa.

To implement this architecture, we select Jena [13] for processing ontology-related tasks. Jena is open source and grown out of work with the HP Labs Semantic Web [12]. It is an Java framework for building Semantic Web applications and provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine. The performance data repository service is developed based on OGSA [7]. In the following subsections, we detail components of ontology-based performance data repository service.

4.1 Ontological Database

Ontological database is used as a persistent storage for ontology descriptions and instance data. Given an application, for each experiment, performance data is a set of instances (individuals) following a specific ontology model (e.g. PERFONTO). Similarly, for each Grid site, system performance data is collected and archived. We can store both ontology model and instance data in persistent storage. Note that the ontological database is used as a storage. The task of storing and retrieving ontological data is done by other components (e.g. sensors, measurement and monitoring tools). To store ontology model and instance data into persistent storage, we use Jena API for persistent database models. The ontology model and associated instance data are kept in separately although this is not required.

4.2 Search on Ontological Data

A search engine is developed to support clients finding interesting data in the ontological database. At the initial step, we use a search engine provided by Jena that supports RDQL (RDF Data Query Language) query language [16].

RDQL syntax is similar to SQL. In RDQL, the *SE-*

LECT clause identifies the variables to be returned to the application; variables are introduced with a leading *'?'*. The *FROM* clause specifies the model by URI (Uniform Resource Identifiers) whereas the *WHERE* clause specifies the graph pattern as a list of triple patterns. The *AND* clause specifies the Boolean expressions. RDQL introduces *USING* clause which provides a way to shorten the length of URIs.

The use of RDQL in combining with ontology can simplify and provide a high-level model of search in performance analysis in which searching query is easily understood and defined by end-user, not only by the tool developer. Let us show a simple example: a client wants to find any region summary executed in compute node *gsr*410 that its wallclock time (denoted by metric name *wtime*) is greater than or equal to 3E8 microsecond. The corresponding RDQL query based on PERFONTO is presented in Figure 4. Line l_1 selects variable *regionsummary* via *SELECT* clause. In line l_2 information about processing unit of *regionsummary*, determined by property *perfonto:inProcessingUnit*, is stored in variable *processingunit*. The compute node of *processingunit* must be *"gsr410"* as stated in line l_3. In line l_4, performance metric of *regionsummary* is stored in variable *metric* and line l_5 states that the name of *metric* must be *"wtime"*. In line l_6, the value of *metric* is stored in variable *value* which must be greater than or equal to 3E8 as specified in line l_7. Line l_8 specifies the URI for the shortened name *perfonto*.

In comparison with searching performance data based on relational database [27], we found that ontology-based searching query is more simple, user-oriented and easy to be implemented.

4.3 Reasoning on Ontological Data

The use of ontology for representing performance data allows additional facts to be inferred from instance data and ontology model by using axioms or rules. Based on ontology, we can employ inference engine to capture knowledge via rules.

Let us analyze a simple rule for detecting all MPI point-to-point communication code regions of which the average message length is greater than a predefined threshold [6]. As presented in Figure 5, line l_1 defines the name of the rule. In line l_2, a term of triple pattern specifies link between a region summary and its associated code region. Line l_3 states the code region is an instance of *MPCodeRegion* (message passing code region) and is an MPI point-to-point communication region (denoted by mnemonic CR_MPIP2P) as specified in line l_4. Line l_5, l_6 and l_7 are used to access the average message length of the region summary. Line l_8 checks whether the average message length is greater than a predefined threshold (BIG_MESSAGES_THRESHOLD) by using a built-in function. In line l_9, the action of this rule concludes and prints the region summary having big message. This example shows how using ontology helps simplifying the reasoning on performance data.

5 Enhanching Automatic Performance Analysis

An automatic performance analysis system typically consists of various components based on different tools and specifications [9]. The use of PERFONTO particularly and ontology generally can help increasing the degree of automation of performance analysis. For example, to automate basic performance analysis experiments, performance analyzers must be able to process information (including static and dynamic) about the application; that information is mostly collected by different tools. The idea of using ontology (e.g. PERFONTO) fits well for that purpose as ontology can describe the semantics of information about applications. Moreover, by using ontology, high-level components such as analysis agents [8] could be able to understand data provided by many sources in heterogenious environment even if they do not know the exact type of that data in advance. Another potential of ontology that can be applied in automatic performance analysis is reasoning capability. As simply demonstrated in Section 4.3, we can define rules for automatic performance analysis. Based on predefined patterns, the action of rules can automatically determinate and perform appropriate tasks without or with little any human intervention. By doing so,

```
SELECT ?regionsummary
WHERE
     (?regionsummary perfonto:inProcessingUnit ?processingunit)
     (?processingunit perfonto:inNode "gsr410")
     (?regionsummary perfonto:hasMetric ?metric)
     (?metric perfonto:hasMetricName "wtime")
     (?metric perfonto:hasMetricValue ?value)
AND (?value >=3E8)
USING perfonto FOR <http://www.par.univie.ac.at/project/scalea/perfonto#>
```

Figure 4: An example of RDQL query based on PERFONTO.

```
[rule_detect_bigmessages:
     (?regionsummary perfonto:ofCodeRegion ?codeRegion),
     (?codeRegion rdf:type perfonto:MPCodeRegion),
     (?codeRegion perfonto:hasCrType "CR_MPIP2P"),
     (?regionsummary perfonto:hasMetric ?metric),
     (?metric perfonto:hasMetricName "AvgMessageLength"),
     (?metric perfonto:hasMetricValue ?length),
     greaterThan(?length, BIG_MESSAGES_THREADHOLD)
->       print(?regionsummary,"Big message hold!")]
```

Figure 5: An example of Rule-based Reasoning based on PERFONTO.

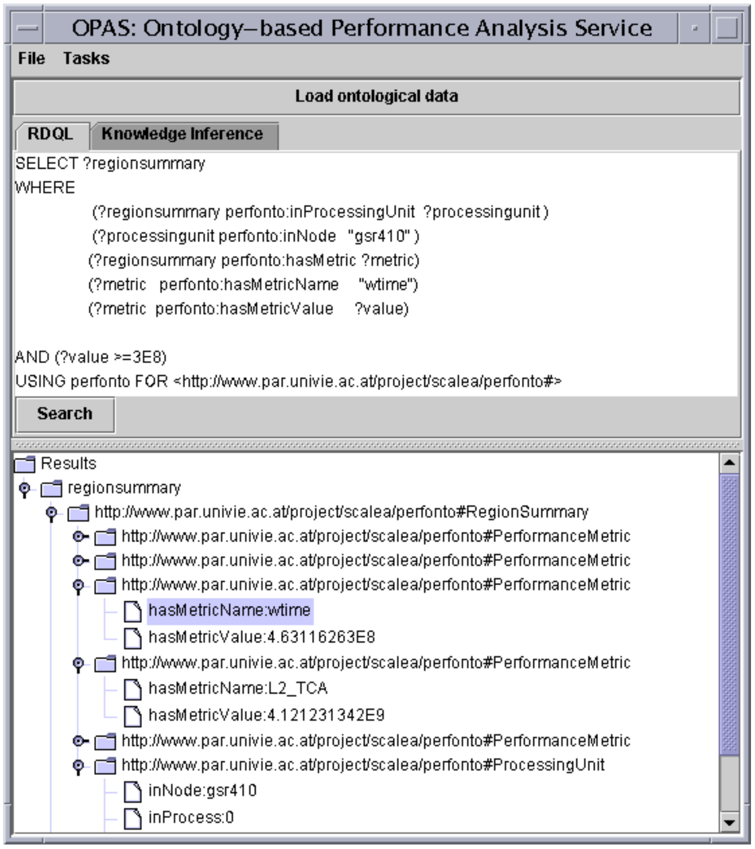
we also can move analysis components as close to resources monitored as possible.

Ontology can also be used for ensuring semantically interactions between tools of an automatic performance analysis system, e.g. instrumentor, experiment planner. The ontology can represent features of available tools by classifying their main components and specifying the relationships and contraints among them, thus allowing users/services to select and configure the most suitable solution for automatically executing a performance analysis process.

6 Prototype Implementation

We are currently implementing the proposed ontology and ontology-based service. The ontology-based performance data repository is an OGSA-based service of which the ontological database is based on PostgreSQL. However, in current prototype this service supports only operations for retrieving and storing ontology descriptions and instance data; searching and reasoning have to be done at client side. We are working on providing searching and reasoning operations.

We develop an Ontology-based Performance Analysis Service (OPAS) which supports ontology-based searching and reasoning. Figure 6 presents an user interface for performing searches in OPAS. In the top window the user can specify queries whereas the result will be shown in the bottom window. For example, we conducted a search with the query presented in Section 4.2 with a 3D Particle-In-Cell application.



```
SELECT ?regionsummary
WHERE
            (?regionsummary perfonto:inProcessingUnit ?processingunit )
            (?processingunit perfonto:inNode  "gsr410" )
           (?regionsummary perfonto:hasMetric ?metric)
           (?metric  perfonto:hasMetricName   "wtime")
           (?metric  perfonto:hasMetricValue   ?value)

AND (?value >=3E8)
USING perfonto FOR <http://www.par.univie.ac.at/project/scalea/perfonto#>
```

```
Results
  regionsummary
    http://www.par.univie.ac.at/project/scalea/perfonto#RegionSummary
      http://www.par.univie.ac.at/project/scalea/perfonto#PerformanceMetric
      http://www.par.univie.ac.at/project/scalea/perfonto#PerformanceMetric
      http://www.par.univie.ac.at/project/scalea/perfonto#PerformanceMetric
        hasMetricName:wtime
        hasMetricValue:4.63116263E8
      http://www.par.univie.ac.at/project/scalea/perfonto#PerformanceMetric
        hasMetricName:L2_TCA
        hasMetricValue:4.121231342E9
      http://www.par.univie.ac.at/project/scalea/perfonto#PerformanceMetric
      http://www.par.univie.ac.at/project/scalea/perfonto#ProcessingUnit
        inNode:gsr410
        inProcess:0
```

Figure 6: Graphical User Interface for conducting searches.

In the bottom window, under the subtree of variable *regionsummary*, list of region summaries met the condition will be shown. The user can examine performance metrics in details. Other information, such as source code and machine, can be visualized as needed.

7 Related work

The Pablo Self-Defining Data Format (SDDF) [1] is a data description language that specifies both data record structures and data record instances. The method of using ontology provides more facilities to describe the semantics of performance data.

Database schemas are proposed for representing performance data such as in SCALEA [27], Prophesy [25]. However, these approaches are tool-specific rather than widely-accepted data representations. It is difficult to extend database schema structure to describe new resources. The relational database schema does not explicitly express semantics of data whereas the ontology does. As a result, building knowledge discovery via inference in ontological data is less intensive work, hard and costly.

CIM [3] is a model for describing overall management information in a network/enterprise environment. Our work is different as we are focusing on describing performance data by using OWL.

The Global Grid Forum (GGF) Network Measurements working group has created an XML schema which provides a model for network measurement data [10]. Similarly, GLUE schema [26] defines a conceptual data model to describe computing and storage elements and networks. In [5], ontology has been applied for improving the semantic expressiveness of network management information and the integration of information definitions specified by different network managements. None of these schemas models concepts of application experiments. However, the modeled objects in GGF and GLUE schema are similar to that in our resource-related ontology thus vocabularies and terminologies of these schemas can be incorporated into our resource-related ontology.

There are tools for publishing and managing ontologies on the Web such as Joseki [14], TAP [24]. Our repository service differs from these tools as it is a Grid service intentionally designed for publishing and archiving ontological performance data.

Recent work in [23] describes how ontology can be used in Grid resource matching. Our work differs as we try to propose an ontology to describe performance data of not only compute resources but also applications. Our framework can provide data for matching resources in the Grid.

8 Conclusion and Future Work

In this paper, we have investigated how ontology can help to overcome the lack of semantics description possessed by current techniques that are used in existing performance monitoring and measurement tools to describe performance data in Grids. We have developed an ontology for representing performance data and introduced an architecture for an ontology-based model for performance analysis, data sharing and tools integration. The core of this architecture is a Grid service for archiving and providing ontology models and performance data. Initial results show that ontology is a promising solution in the domain of performance analysis because it not only provides a means for seamlessly utilizing monitoring and performance data but also increases the degree of automation of performance analysis.

Besides working toward the full prototype, we are currently enhancing and reevaluating our proposed ontology. We are extending resource-related concept to cover dynamic data of compute and network systems at runtime, and advancing the experiment-related ontology to describe performance properties, performance data of workflow applications, etc. Another future work is to study the use of ontology for mapping between different representations of performance data. In addition, we plan to develop a task-based ontology that describes conceptualizations of tasks and processes along with their interrelationships and properties for an automatic performance analysis system described in [9].

Acknowledgements

This research is supported by the Austrian Science Fund as part of the Aurora Project under contract SFBF1104. We thank Dr. Ian Glendinning at the Institute for Software Science, University of Vienna, for his useful comments and suggestions.

References

[1] R. A. AYDT. SDDF: The pablo self-describing data format. Tech. rep., Department of Computer Science, University of Illinois, April 1994.

[2] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.

[3] Common Information Model (CIM). http://www.dmtf.org/standards/standard_cim.php.

[4] DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html.

[5] J.E. Lopez de Vergara, V.A. Villagra, J.I. Asensio, and J. Berrocal. Ontologies: Giving semantics to network management models. *IEEE Network*, 17(3):15–21, May-June 2003.

[6] T. Fahringer, M. Gerndt, Bernd Mohr, Felix Wolf, G. Riley, and J. Träff. Knowledge Specifi-

cation for Automatic Performance Analysis, Revised Version. APART Technical Report, Workpackage 2, Identification and Formalization of Knowledge, Technical Report http://www.kfa-juelich.de/apart/result.html, August 2001.

[7] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid Services for Distributed System Integration. *IEEE Computer*, pages 37–46, June 2002.

[8] M. Gerndt, A. Schmidt, M. Schulz, and R. Wismueller. Performance Analysis for Teraflop Computers - A Distributed Automatic Approach. In *Proceedings of 10th Euromicro Workshop on Parallel, Distributed, and Network-based Processing (EUROMICRO-PDP 2002)*, Canary Islands, SPAIN, 2002.

[9] Michael Gerndt and Bernd Mohr. Automatic Performance Analysis Roadmap Report. APART Deliverable, http://www.kfa-juelich.de/apart/result.html, January 2001.

[10] GGF Network Measurements Working Group. http://forge.gridforum.org/projects/nm-wg/.

[11] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[12] HP Labs Semantic Web Research. http://www.hpl.hp.com/semweb/.

[13] Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net.

[14] Joseki. http://www.joseki.org/.

[15] DAML: The DARPA Agent Markup Language. http://www.daml.org/.

[16] RDQL: RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm.

[17] OIL: Ontology Inference Layer. http://www.ontoknowledge.org/oil/.

[18] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge, Colorado, USA, April 2000.

[19] N. Noy and D. L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report KSL-01-05, Knowledge Systems Laboratory, March 2001.

[20] OWL Web Ontology Language Reference. http://www.w3.org/tr/owl-ref/.

[21] Resource Description Framework (RDF). http://www.w3.org/rdf/.

[22] RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/tr/rdf-schema/, January 2003. W3C Working Draft.

[23] H. Tangmunarunkit, S. Decker, and C. Kesselman. Ontology-based Resource Matching in the Grid—The Grid meets the Semantic Web. In *Proceedings of the Second International Semantic Web Conference*, Sanibel-Captiva Islands, Florida, USA, October 2003.

[24] TAP: Building the Semantic Web. http://tap.stanford.edu/.

[25] V. Taylor, X. Wu, J. Geisler, X. Li, Z. Lan, R. Stevens, M. Hereld, and Ivan R.Judson. Prophesy:An Infrastructure for Analyzing and Modeling the Performance of Parallel and Distributed Applications. In *Proc. of HPDC's 2000*, Pittsburgh, August 2000. IEEE Computer Society Press.

[26] The Grid Laboratory Uniform Environment (GLUE). http://www.cnaf.infn.it/ sergio/datatag/glue/index.htm.

[27] Hong-Linh Truong and Thomas Fahringer. On Utilizing Experiment Data Repository for Performance Analysis of Parallel Applications. In *9th International Europar Conference(EuroPar 2003)*, Lecture Notes in Computer Science, Klagenfurt, Austria, August 2003. Springer-Verlag.