K-WfGrid Distributed Monitoring and Performance Analysis Services for Workflows in the Grid *

Hong-Linh Truong, Peter Brunner, Thomas Fahringer, Francesco Nerieri, Robert Samborski
Institute of Computer Science, University of Innsbruck, Austria
Email: {truong, brunner, tf, nero, robert}@dps.uibk.ac.at

Bartosz Baliś, Marian Bubak, Kuba Rozkwitalski
Institute of Computer Science, AGH, Poland
Email: {balis, bubak}@uci.agh.edu.pl, kubaroz@gmail.com

Abstract

Grid workflows for e-science are complex and prone to failures. However, there is a lack of performance monitoring and analysis tools for supporting the user as well as workflow middleware to monitor and understand the performance of complex interactions among Grid applications, middleware and resources involved in workflow executions. In this paper, we present a novel integrated environment which supports online performance monitoring and analysis of service-oriented workflows. Performance monitoring and analysis of Grid workflows and infrastructure is conducted through a Web portal. Performance overheads of Grid workflows are analyzed in a systematic way, and performance problems can be detected during runtime. Moreover, we present several languages that alleviate the interaction among performance monitoring and analysis services and their clients. Our system has been integrated into the K-WfGrid knowledge-based workflow system. It plays a key role in supporting the user and developer to analyze their workflows and in providing performance knowledge for constructing and executing workflows.

1 Introduction

Recently, Grid workflows have been increasingly used for solving large scale problems in e-science, e.g., biology, molecular science, astrophysics and environmental simulations. Therefore, many workflow systems have been developed for e-science, as surveyed in [18]. However, the work on developing workflow systems for e-science is still at early stages as most efforts are spent on developing workflow languages and execution environments. Users and developers still have to do a daunting task when developing and executing Grid workflows due to the lack of supporting tools, notably performance monitoring and analysis tools.

High complexity is inherent in Grid systems, thus requiring performance monitoring and analysis tools to monitor and capture relevant monitoring data at multiple levels of abstraction and to provide insights into the performance of Grid applications and resources as well as their interactions. Monitoring and analysis tools not only assist users to verify and validate the execution of their workflows, and to decide when to steer their applications, but also provide insightful, detailed information about Grid services and resources to resource management, workflow scheduling, workflow construction and execution tools.

In this paper, we present the implementation of a novel monitoring and performance analysis system for workflows in the K-WfGrid project [3]. The K-WfGrid distributed monitoring and analysis environment integrates various services for performance instrumentation, monitoring and analysis of Grid workflows into a single system. It provides a single Web portal for the user to conduct any performance monitoring and analysis tasks. Moreover, we also introduce languages that alleviate the interaction among various services and clients involved in performance monitoring and analysis. Using these languages, any clients can easily obtain monitoring data, specifying performance problems and retrieving performance results.

The rest of this paper is organized as follows: Section 2 discusses the motivation and contribution of the paper. Section 3 presents the architecture of the distributed performance monitoring and analysis system. Performance monitoring is discussed in Section 4, followed by performance search techniques presented in Section 5. The monitoring and analysis portal is described in Section 6. We present experiments in Section 7. Section 8 discusses related work

*The work described in this paper is supported by the European Union through the IST-2002-511385 project K-WfGrid.

and Section 9 summarizes the paper.

2 Motivation and Contribution

Our work is motivated by the challenges of performance monitoring and analysis of hierarchical concepts of Grid/Web services-based workflows. We observed that workflows imply multiple levels of abstraction, ranging from workflow, workflow region, workflow activity to invoked application and code region. Therefore, depending on which level we want to examine, the performance monitoring and analysis tool for Grid workflows has to use different techniques to deal with different types of services because Grid workflows for e-science are commonly composed of different applications and are executed on different computational services. Moreover, the execution of Grid workflows is dynamic, involves many services and resources, implies complex dependencies, and normally lasts a long time. Therefore, techniques to monitor, analyze and detect performance problems during runtime are important. In addition, just for monitoring and analyzing performance of Grid workflows, various services have to be developed and deployed in different Grid sites. But these services have to be integrated with each other and to serve many different types of clients such as end-users and Grid middleware. As a result, the services must provide well-defined, extensible interfaces and data representation to facilitate the service interoperability and integration in the Grid.

This paper tackles the above-mentioned challenges. The main contributions of this paper are (i) the design and implementation of a comprehensive, unified, extensible performance monitoring and analysis of Web services-based workflows; (ii) novel request representations alleviate the interaction between performance services and their clients; and (iii) performance of Grid workflows is systematically analyzed according to a classification of workflow overheads, and performance problems can be detected online by interpreting performance metrics at runtime.

3 Integrated Architecture for Workflow Performance Monitoring and Analysis

Figure 1 depicts the architecture of performance monitoring and analysis services in K-WfGrid that includes GEMINI, DIPAS (DIstributed Performance Analysis Service) Gateway, and DIPAS Portal. GEMINI is responsible for monitoring workflows executed by GWES (Grid Workflow Execution Service) [6] which supports Petri-net based workflow representations. GEMINI instruments GWES and Grid services, capturing the execution status of workflows and Grid services as well as Grid resources. The monitoring data provided by GEMINI is used by DIPAS

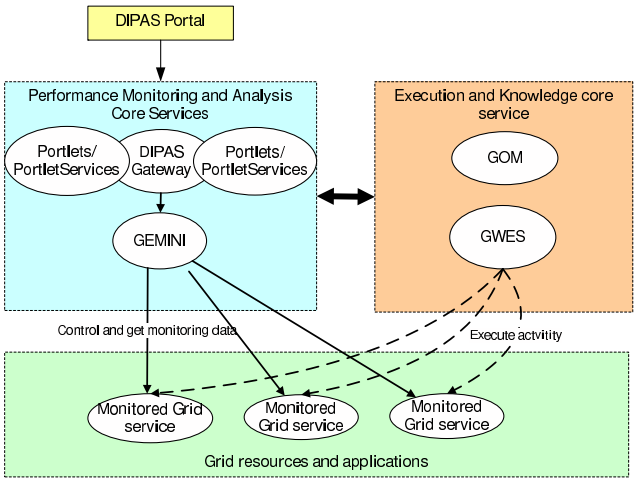

Figure 1. The K-WfGrid performance monitoring and analysis system

Gateways which determine workflow overheads and performance problems. Moreover, monitoring data can also be processed and visualized at DIPAS Portal because we apply various other performance monitoring and analysis techniques to Grid workflows during runtime. In the latter case, monitoring data is processed by an applet in the portal and by portlets/portlet services. Both GEMINI and DIPAS Gateways consist of many components/services which are distributed on various Grid sites.

GEMINI and DIPAS use GOM (Grid Organizational Memory) [15] to publish and search information about monitoring data of workflows. Since the monitoring and analysis is conducted through a web portal, DIPAS Portal, which interacts with, requests and retrieves data from many services, e.g., GEMINI and GWES, a set of replicated DIPAS Gateways is also responsible for shielding all the complex interactions from the portal to other services. In this case, DIPAS Gateways handle requests from the portal and forward the requests to corresponding services, and act as a proxy from which the portal can retrieve the requested data.

4 Instrumentation and Online Monitoring

4.1 Instrumentation

GEMINI consists of (i) an instrumentation system to control the performance measurement, and (ii) application sensors to collect monitoring data and to deliver the data to the monitoring service. GEMINI supports dynamically enabled instrumentation of Grid workflow applications:

- *Activity-level monitoring*: to instrument and monitor invoked applications, which perform the real task of workflow activities, and code regions within invoked

applications. In this level, the instrumentation is currently done before the execution of applications but the measurement is dynamically activated at runtime.

- *Workflow-level monitoring*: to statically instrument GWES and to monitor workflow executions. This provides monitoring data for analyzing performance of workflows, workflow regions and workflow activities.

Since the instrumentation involves several services and multilingual applications, we have to develop a neutral means to allow different clients to understand the structure of applications, to select code regions of interest, and to control the instrumentation of multilingual applications. Our approach is that we use XML-based representations for describing application structures and specifying instrumentation requests. Detailed instrumentation techniques and control mechanism can be found in [8].

4.2 Publishing and Retrieving Monitoring Data

GEMINI collects monitoring data from various sources. To ensure that many clients and services can seamlessly process and utilize various monitoring data types of different monitored resources as well as to increase the dissemination of monitoring data, all monitoring data is represented in XML format. In order for the client to locate monitoring services which provide monitoring data, GEMINI publishes information about available monitoring data into GOM, an ontology-based knowledge repository. Information about monitoring data is described in OWL (Web Ontology Language) and OWL-based information is published into GOM. From GOM, any client knows to which GEMINI service it should contact in order to obtain the monitoring data. From ontological descriptions, any client can build requests which are sent to GEMINI in order to retrieve the monitoring data.

We have developed a generic performance data query and subscription (PDQS) language for querying and subscribing various types of monitoring data. Figure 2 shows the schema of PDQS. Basically, every monitoring data type and monitored resource is associated with a unique *dataTypeID* and *resourceID*, respectively. Based on that, the client can specify PDQS requests which include *(dataTypeID,resourceID)* together with other information like XPath-based data filters (denoted by *dataFilter*) and subscription time (denoted by *subscriptionTime*). A PDQS request can be used to *query* or *subscribe* the same monitoring data type of various resources.

PDQS requests can be built based on published information in GOM. For example, Figure 3(a) presents an OWL description for workflow events of the workflow whose workflow ID is

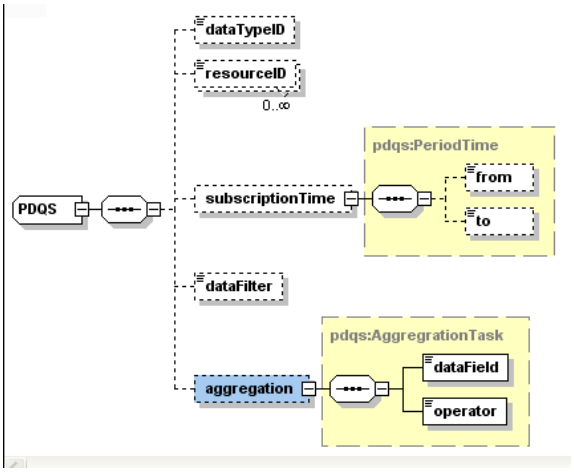

Figure 2. Performance data query and subscription (PDQS) language visualized with the `XMLSpy` tool

`truong_810cf130-eb24-11da-8ebd-a46bfd55290e`. Figure 3(b) presents a PDQS request used to get all workflow events generated during the workflow's execution.

5 Performance Search for Workflows

One of the goals of DIPAS Gateways in Figure 1 is to search for performance problems and to inform clients about detected problems. In order to search for performance problems, we have developed a novel classification of performance overheads for workflows that includes middleware overheads (e.g., due to scheduling or resource management), loss of parallelism overheads (e.g., due to load imbalance), etc. The performance of K-WGrid workflows is systematically analyzed according to that classification. Based on performance overheads, we define performance severity as the ratio of performance overheads to the total execution time. Performance severity indicates the importance of a performance overhead with respect to the performance of the workflow. Performance problems can then be determined based on conditions established on the basis of appropriate performance metrics (for example, overheads and severities), and their thresholds. Given a predefined threshold for a performance metric, a performance problem occurs when the value of the metric is greater than the threshold. During the execution of the workflow, any clients of DIPAS can specify requests to obtain performance overheads, severities and problems.

To simplify the interaction between clients and DIPASGateways, we design a novel workflow analysis request language (WARL) which is used to specify analysis requests. Figure 4 presents the cur-

```
<dg:DataObject rdf:ID="MD1148476305524_DO">
 <dg:contains>
      <dg:MonitoringData rdf:ID="MD1148476305524">
        <dg:hasDataType rdf:datatype="...">wfa.event</dg:hasDataType>
        <dg:ofResource rdf:datatype="..."
        >truong_810cf130-eb24-11da-8ebd-a46bfd55290e
        </dg:ofResource>
        <dg:validFrom rdf:datatype="...">1148476305524</dg:validFrom>
        <dg:validTo rdf:datatype="...">0</dg:validTo>
      </dg:MonitoringData>
    </dg:contains>
 <dg:isStoredIn rdf:resource="http://gom.kwfgrid.net/gom/ontology/
      ServiceRegistry/CMN#MSa6240bba-3c48-4cc6-ad31-648e9b60124b"/>
</dg:DataObject>
```

(a)

```
<?xml version="1.0"?>
<pdqs xmlns="http://net.kwfgrid/dr/pdqs">
   <dataTypeID>wfa.event</dataTypeID>
   <resourceID>truong_810cf130-eb24-
   11da-8ebd-a46bfd55290e</resourceID>
   <subscriptionTime>
      <from>0</from>
      <to>0</to>
   </subscriptionTime>
</pdqs>
```

(b)

Figure 3. Example of (a) OWL description (simplified) and (b) corresponding PDQS

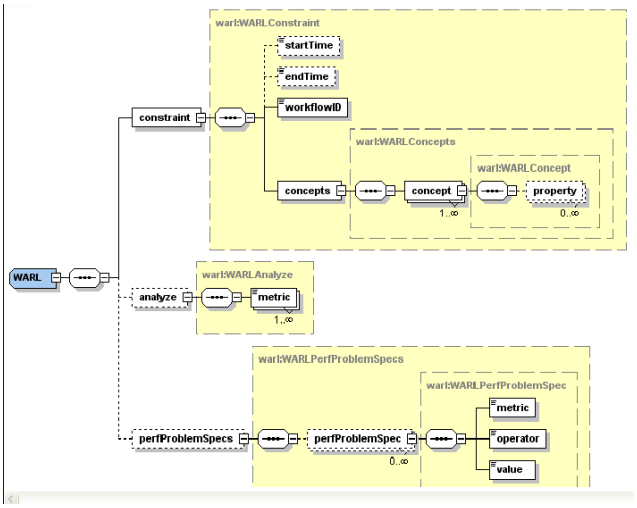

Figure 4. Workflow analysis request language (WARL) visualized with the `XMLSpy` tool

```
<?xml version="1.0" encoding="UTF-8"?>
<warl>
<constraint>
  <startTime>0</startTime><endTime>0</endTime>
  <workflowID>truong_3d6c4330-eb2a-11da-8ebd-a46bfd55290e
  </workflowID>
  <concepts>
   <concept name="truong_3d6c4330-eb2a-11da-
                8ebd-a46bfd55290e" type="Workflow"/>
   <concept name="computeStartZonePolyg"
                                    type="Activity"/>
   <concept name="computeEndZonePolyg" type="Activity"/>
   <concept name="computeStartNodes" type="Activity"/>
  </concepts>
</constraint>
<analyze>
   <metric>LoadIm</metric><metric>TotalOverhead</metric>
   <metric>QueuingTimeSeverity</metric>
</analyze>
 <perfProblemSpecs>
   <perfProblemSpec><metric>SynDelaySeverity</metric>
       <operator>GE</operator><value>0.2</value>
   </perfProblemSpec>
   <perfProblemSpec><metric>QueuingTimeSeverity</metric>
       <operator>GE</operator><value>0.1</value>
   </perfProblemSpec>
 </perfProblemSpecs>
</warl>
```

Figure 5. Examples of WARL request for obtaining performance overheads and specifying performance conditions

rent version of WARL. A WARL request includes three parts: constraints (element `constraint`, type `WARLConstraint`), performance metrics to be analyzed (element `analyze`, type `WARLAnalyze`), and performance conditions (element `perfProblemSpecs`, type `WARLPerfProblemSpecs`). Constraints include information about hierarchical workflow concepts (e.g., `Workflow` and `Activity`) to be analyzed and their properties (e.g., the name of `Activity`). Each concept is identified by a name and a type: the name indicates the identifier of the concept in the workflow description, e.g. activity name, while the type determines whether the concept is a workflow or an activity or a code region, etc. A WARL `analyze` request specifies a list of performance overheads that should be analyzed and provided. Performance problems can be checked by specifying a set of performance conditions, each condition includes a metric name, an operator (e.g., greater than or less than), and a value (e.g., indicating a threshold). Given WARL requests, the performance analysis service will conduct corresponding analyses and send to the requesters the analysis result described in XML. Figure 5 presents an example of a WARL request which is used to analyze performance metrics and performance problems of three activities `computeStartZonePolyg`, `computeEndZonePolyg`, `computeStartNodes` and the workflow `truong_3d6c4330-eb2a-11da-8ebd-a46bfd55290e`.

6 Integrated Performance Monitoring and Analysis Portal

In K-WfGrid, the user conducts performance monitoring and analysis through a Web portal. All features for mon-

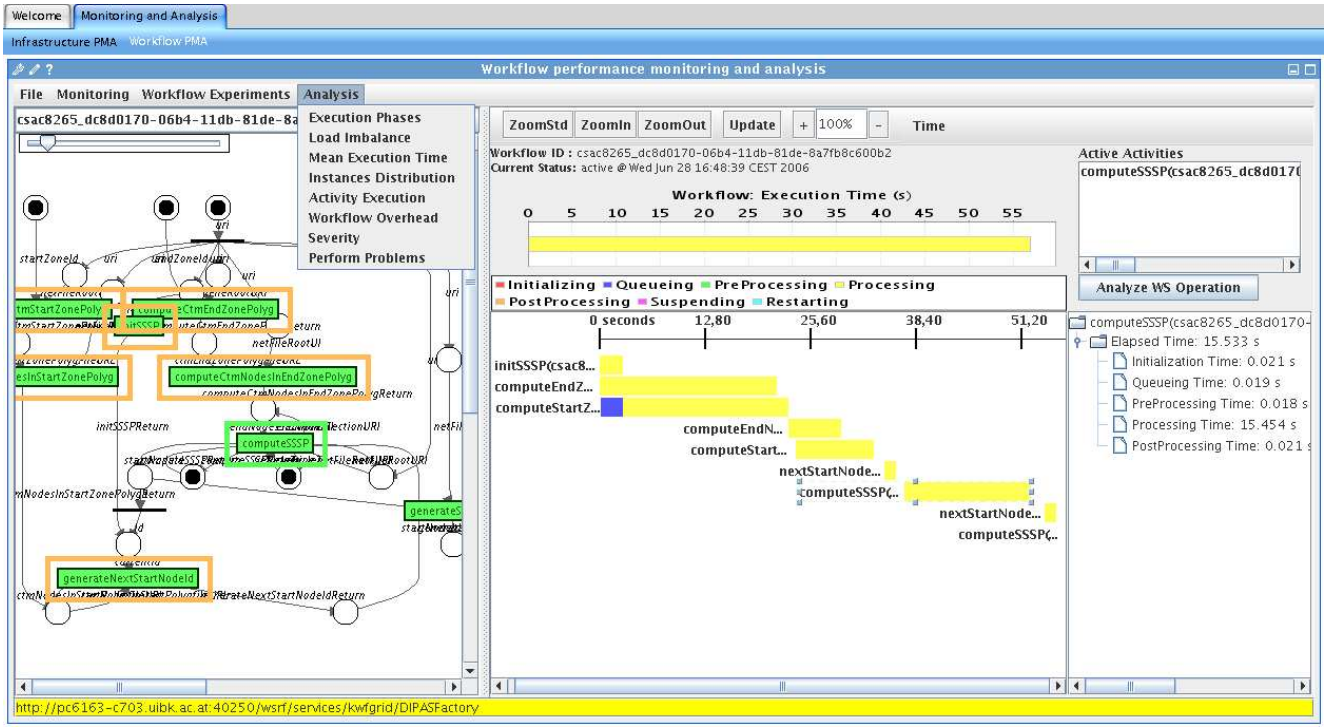

Figure 6. Workflow performance monitoring and analysis portal.

itoring and analyzing the performance of Grid workflows as well as Grid resources are integrated into DIPAS Portal, thus providing a unique place for the user to carry out any monitoring and analysis tasks. In the portal, the user basically can (i) conduct the online monitoring of the execution of workflows together with Grid resources on which the Grid services are executed, (ii) request the analysis service to determine the overhead associated with workflows, and (iii) control the instrumentation and measurement process of Grid services. Currently, the DIPAS Portal includes:

- a set of portlets that allows the user to configure the monitoring service, to select existing monitored resources, to specify PDQS requests, and to query and/or subscribe monitoring data and events.
- a Java applet for conducting performance visualization and analysis of Grid workflows.

Figure 6, for example, depicts our performance monitoring and analysis portal for Grid workflows. From the portal the user can select existing workflows, currently being executed by or submitted to GWES, and start the monitoring and analysis. The workflow representation is shown in the left-pane whereas online execution progress of the workflow and its activities, together with performance metrics, are shown in the right-pane. During runtime, the user can conduct any analysis by selecting activities or activity instances and features in the menu. For example, the user can examine the execution phase of the whole workflow or performance metrics of activities/instances. Performance analysis of completed workflows is also supported.

7 Experiments

We have implemented our prototype and integrated it into the K-WfGrid system. Monitoring and analysis services are WSRF-based, powered by GT4 [10], whereas the portal is based on Gridsphere [11]. However, the dynamically-enabled instrumentation GUI has not been integrated into the portal. In this section, we present a few selected experiments conducted through the portal to demonstrate the usefulness of our tool.

7.1 Infrastructure Monitoring

Figure 7 presents an example of using DIPAS Portal to monitor a QoS parameter of `Availability` of K-WfGrid services. The portal `Data Query and Subscription` provides interfaces for selecting monitored resources, specifying PDQS requests, and performing data query and subscription. In the portal `Display Data`, the performance status of monitored resources is visualized and updated over the time. For example, the portal `Display Data` shows an example about the availability of GOM.

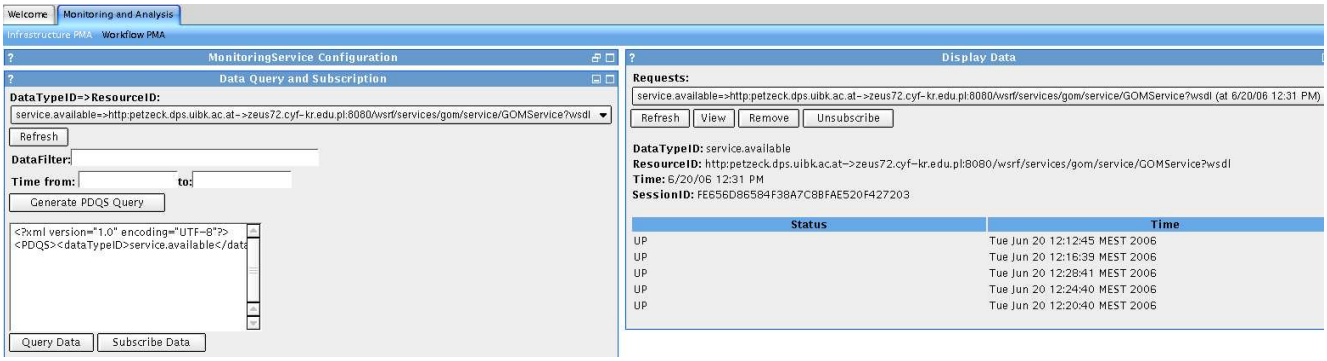

Figure 7. Example of monitoring availability of K-WfGrid services

7.2 Workflow Performance Monitoring and Analysis

Our experimental workflow is a K-WfGrid pilot application named CTM (Coordinated Traffic Management) workflow, developed by Softeco Sismat S.p.A. This workflow is used to compute emissions for the main pollutants in an urban area. Figure 8 presents the CTM workflow visualized in DIPAS Portal using the K-WfGrid GWUI (Grid workflow user interface) library. In CTM, the topology of the urban area, divided into different zones, is described in `.NET` file. Moreover, an origin/destination zone matrix with traffic flows data per different vehicle types is provided. Based on that the CTM will compute the emission. Currently, Web services of the CTM are deployed in Innsbruck and Genoa. In our experiments, GWES is a centralized service deployed in Innsbruck whereas GOM is a distributed service deployed in a cluster in Cracow, Poland. We deploy DIPAS Portal, DIPAS Gateway and GEMINI on different machines in Innsbruck.

Through the portal the user can observe the execution of the CTM. When the execution of an activity changes status, the color of the outer rectangle enclosing the activity node in the visualization of workflow representation will be changed accordingly. For example, in Figure 8, the execution of most activities, except activity `generateSVGFile`, are `completed`. Figure 9 shows the execution trace of activity instances of the CTM. The tree in left-pane shows performance metrics associated with an instance of activity `computeEndZonePolyg` which takes a large portion of processing time. The window `Load Imbalance` shows that the load balance among instances of activity `computerSSSP` is poor. The scheduler decides to invoke more service operations on a Genoa machine, `grid02.softeco.it` than that on a machine in Innsbruck, `kwfgrid.dps.uibk.ac.at`, as shown in the window `Distribution of Activity Instances`. Overall only two activities named `computeSSSP` and `computeEndZonePolyz` dominate the execution time, as presented in the window `Execution Phases` which displays summarized execution time of phases for individual activities.

During runtime, we can obtain performance overheads and problems of workflows. In Figure 10, the window `Overhead Analysis` shows an instance of the overhead tree which is updated when the performance search produces new results whereas the window `Performance Problems Analysis` displays any performance problems detected based on severities and thresholds, identified by user-defined specifications of performance conditions.

8 Related Work

Many techniques have been introduced to study quality of service and performance models and to monitor the execution of business Web services and workflows [14, 9, 7]. The WebLogic Process Integrator [5] allows the user to examine status of workflow instances. However, its monitor is limited to the activity level. Web Service Navigator [16] provides visualization techniques for Web service based applications. ARM defines means for obtaining monitoring data of business transactions through instrumentation [12]. However, ARM agents are monitoring sensors that could be integrated into our framework. Performance search and analysis at multiple levels of abstraction are not supported in the business domain.

In the Grid domain, there is a lack of monitoring and analysis tools for workflows of Grid/Web services. P-GRADE [13] supports tracing of workflow applications. In contrast to K-WfGrid, it does not support cyclic and Web service-based workflows. Taverna Workbench [17] also monitors status of activities within Grid workflows, but provides information in simple tables. The OntoGrid project [4] uses knowledge gained from monitoring data to debug workflows, but not to monitor and analyze the performance. The ASKALON [1] and the K-WfGrid share many monitoring and analysis methods and concepts in common. However, performance analysis techniques in ASKALON sup-

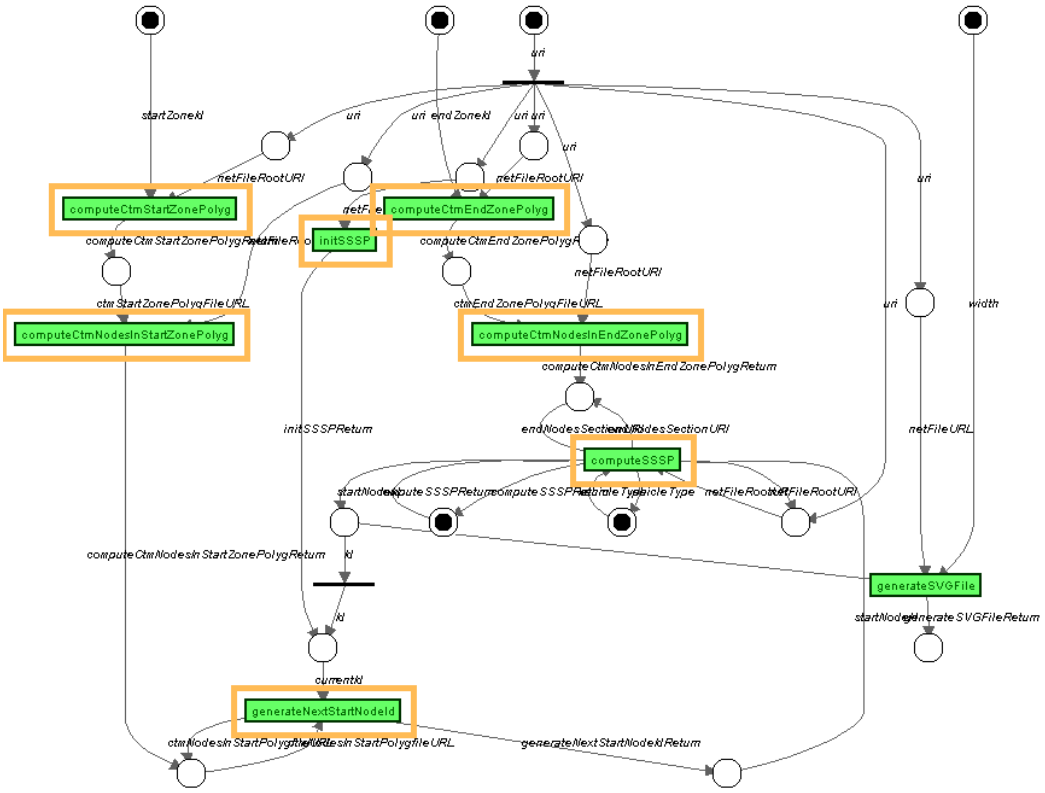

Figure 8. The CTM workflow representation visualized in DIPAS Portal

port workflows of C/Fortran-based scientific applications.

K-WfGrid performance monitoring and analysis services differ from these tools in many aspects. Firstly, our services are WSRF-based services. Most existing performance monitoring and analysis tools for workflows are limited to the activity level and do not support the selection of abstraction levels to which the analysis should be limited. Existing performance analysis tools for workflows neither provide a systematic classification of overheads nor support online detection of performance problems, and are not integrated with infrastructure monitoring and analysis.

9 Conclusions and Future Work

In this paper, we have presented a novel unified and extensible system for performance monitoring and analysis of Grid workflows. We have described a prototype and demonstrated the usefulness of our system. The main contributions of the paper are centered on novel techniques to support performance monitoring and analysis of multilingual workflow-based applications at multiple levels. Although designed for the K-WfGrid workflow system, the performance framework presented in this paper is interoperable and extensible, and can easily be adapted to any other workflow systems, e.g, those supporting BPEL [2].

Currently, the dynamically enabled instrumentation has not been fully achieved and we are working on integrating it into our system. We plan to store performance results into GOM for conducting multiple-experiment analysis.

References

[1] ASKALON: Grid Application Development and Computing Environment, http://www.askalon.org.

[2] Business Process Execution Language for Web Services, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

[3] K-WF Grid Project. http://www.kwfgrid.net.

[4] Ontogrid project, http://www.ontogrid.net.

[5] WebLogic Process Integrator Overview, http://edocs.beasys.com/wlpi/wlpi11/studio/index.htm.

[6] Grid Workflow Execution Service (GWES), http://www.gridworkflow.org/kwfgrid/gwes/docs/, 2006.

[7] A. F. Abate, A. Esposito, N. Grieco, and G. Nota. Workflow performance evaluation through wpql. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 489–495. ACM Press, 2002.

[8] B. Balis, H.-L. Truong, M. Bubak, T. Fahringer, K. Guzy, and K. Rozkwitalski. An Instrumentation Infrastructure for Grid Workflow Applications. In *International Symposium*

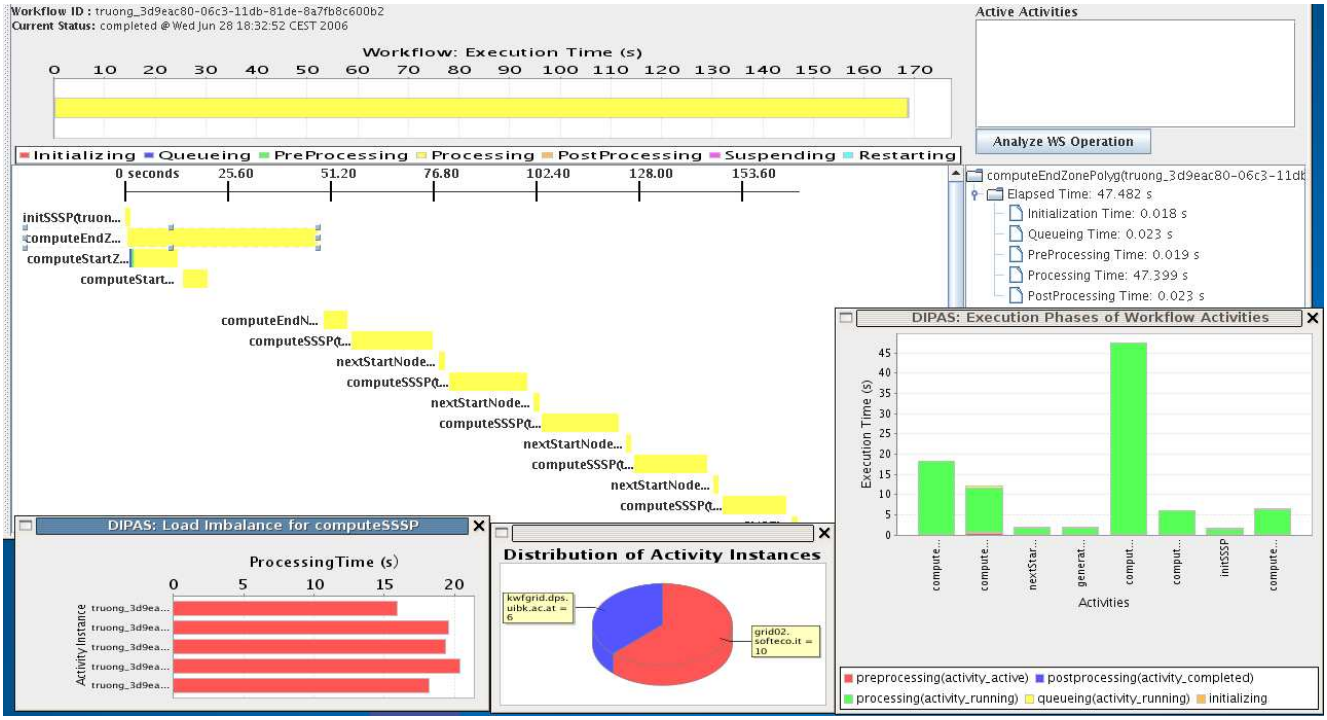

Figure 9. Execution trace of activity instances in the CTM workflow

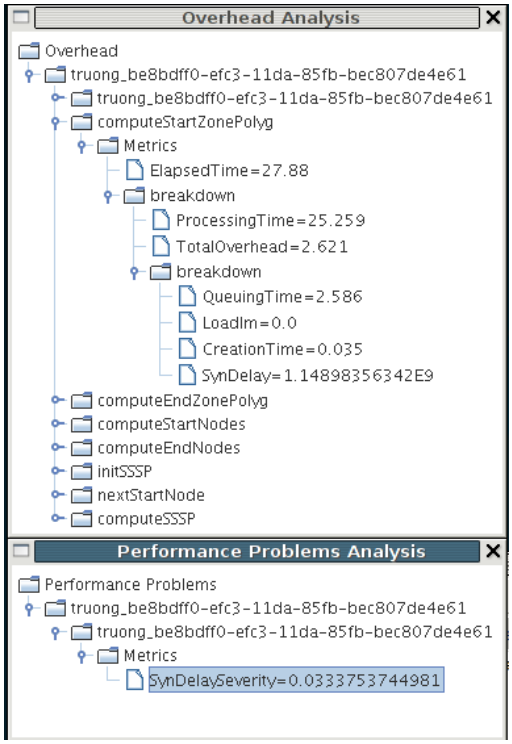

Figure 10. Online analysis of performance overheads and problems

on Grid Computing, High-Performance and Distributed Applications (GADA06), LNCS, 2-3 November 2006.

[9] J. Cardoso, A. P. Sheth, and J. A. Miller. Workflow quality of service. In K. Kosanke, R. Jochem, J. G. Nell, and A. O. Bas, editors, *ICEIMT*, volume 236 of *IFIP Conference Proceedings*, pages 303–311. Kluwer, 2002.

[10] Globus Project. http://www.globus.org.

[11] GridSphere Portal Framework. http://www.gridsphere.org.

[12] M. W. Johnson. Monitoring and diagnosing applications with arm 4.0. In *Int. CMG Conference*, pages 473–484. Computer Measurement Group, 2004.

[13] P. Kacsuk, G. Dozsa, J. Kovacs, R. Lovas, N. Podhorszki, Z. Balaton, and G. Gombas. P-GRADE: a Grid Programming Environment. *Journal of Grid Computing*, 1(2):171–197, 2003.

[14] K.-H. Kim and C. A. Ellis. Performance analytic models and analyses for workflow architectures. *Information Systems Frontiers*, 3(3):339–355, 2001.

[15] B. Kryza, M. Majewska, R. Slota, and J. Kitowski. Unifying grid metadata representations through ontologies. In *Proc. of the 6th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005*, volume 3911 of *LNCS*, Poznan, Poland, 11-14 September 2006. Springer.

[16] W. D. Pauw, M. Lei, E. Pring, L. Villard, M. Arnold, and J. F. Morar. Web services navigator: Visualizing the execution of web services. *IBM Systems Journal*, 44(4):821–846, 2005.

[17] Taverna, http://taverna.sourceforge.net/, 2006.

[18] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3):44–49, 2005.