

Adapting and Evaluating Commercial Workflow Engines for e-Science

Sharanya Eswaran, David Del Vecchio, Glenn Wasson, and Marty Humphrey
Department of Computer Science, University of Virginia, Charlottesville, VA USA
{se6d, dad3e, wasson, humphrey}@cs.virginia.edu

Abstract

Numerous Grid workflow engines exist, each generally specialized for a single application domain such as protein folding. Although the underlying purpose and functionality of the Grid workflow engines are similar, and they make use of a common set of Grid protocols, the implementations vary vastly, making interoperation among them nearly impossible. On the other hand, the concept of workflows of Web services is very popular in the enterprise domain and many engines exist for business workflows. These mostly revolve around the Business Process Execution Language (BPEL), an emerging standard for Web services workflow description. We study three different business workflow engines - Microsoft's BizTalk Server, Microsoft's Windows Workflow Foundation and Oracle's Business Process Manager and analyze how each of these engines can be adapted to e-science and the Grid environment. We have implemented widely-used grid workflows on each of these engines and show quantitatively and qualitatively that each business engine has positive and negative aspects for large-scale e-science.

1. Introduction

Many excellent workflow engines for grids and/or e-science exist today [1-10]. They differ from one another because they are generally tailored toward different application domains and different user requirements. While the engine authors have often painstakingly met a specific set of requirements driven by a particular application, and thus have provided something of deep value to a particular user community, these systems can be difficult to adapt to a new domain. As each tends to utilize its own language for representing the workflow (both statically and dynamically), little can be shared between different grid workflow engines.

Arguably, the commercial community is more advanced than the Grid/e-science community in defining and standardizing workflow languages that at least hold the potential of interoperability. Typical workflow scenarios include document lifecycle management, internal application workflow and business process management. A number of engines have been developed which revolve around well-

established protocols such as BPEL [11], Microsoft's XLANG [12], and IBM's WSFL [13].

A number of researchers have identified that there are issues that need to be addressed before business-oriented workflows can be seamlessly utilized in e-science. For example, Slomanski [14] and Rana [8] independently identify the key challenges: support for large data flows; the need to perform parameterized execution of a large number of jobs; and the need to execute in dynamic environments in which resources come and go. If these commercial workflow engines could be adapted to support grid workflows without an overwhelming effort, then the key advantage would be that the users could find support for workflows already "built into the platform". This would be analogous to the way in which recent Grids have begun to leverage Web services as relatively ubiquitous foundational "plumbing".

The contribution of this paper is a qualitative and quantitative evaluation of three widely-available commercial workflow systems for their suitability to perform general-purpose e-science. The workflow engines studied are: Microsoft's BizTalk Server [18], Microsoft's Windows Workflow Foundation (WF) [19] and Oracle's BPEL Process Manager (OraBPM) [20]. WF is particularly important because it is going to be a foundational component of the next version of the Windows operating system (Vista) --- if Windows comes with, in effect, support for e-science workflow, arguably, the potential numbers of users of these tools could increase dramatically.

To evaluate the three systems, we implemented representative Grid workflows on these engines in the context of the University of Virginia Campus Grid (UVaCG [21]), which includes both Globus services [22] and WSRF.NET services [23]. We find that implementing Grid workflows on these engines is almost as straight-forward as implementing business workflows, provided, the client components for the required Grid services are available in a format that can be easily integrated using these engines. These formats in general include executables, dynamic libraries, Web services and high level code snippets. But the ease of integration for each format varies with the engine. Run-time overhead is minimal.

The rest of this paper is organized as follows. Section 2 introduces the three engines and their

general capabilities. Section 3 identifies the core requirements for a Grid workflow engine. Section 4 describes the steps we took to utilize the business workflow engines in performing e-science. Section 5 provides a qualitative evaluation of the engines from a Grid perspective. Section 6 provides quantitative evaluation and Section 7 concludes this paper.

2. Features of the Workflow Engines

Before discussing the three commercial workflow engines, it is important to describe BPEL [11], which is the core of these commercial systems. BPEL is emerging as the standard XML-based workflow language for defining and executing business processes using XML Web services. Without this standardization, the environment of the commercial systems would be not unlike the current Grid workflow engine landscape. After describing BPEL, and giving an overview of each of the three commercial workflow engines in our study, we describe how to use the workflow engines (Section 2.1) and describe the support for expressiveness and extensibility (Section 2.2).

BPEL enables the composition, orchestration and coordination of web services. A business process described in BPEL can itself be treated as an XML web service. BPEL converged from two other workflow description languages – Microsoft’s XLANG [12] and IBM’s WSFL [13]. BPEL provides constructs for invoking a web service and exchanging messages with a web service, both synchronously and asynchronously. It also has other primitive constructs which include constructs for manipulating data variables, indicating faults and exceptions, terminating a process and, waiting for some time. It also supports compensation blocks for exception handling. BPEL also has control constructs, such as looping, if-then-else and switch-case activities. BPEL supports both sequential and parallel execution of activities. Since BPEL is XML-based, it is extensible, which means that we can add our own constructs and also provide our own implementation of these extensions.

Oracle’s Business Process Manager is a BPEL engine. Microsoft’s BizTalk Server supports BPEL to the extent that BPEL documents can be imported into the BizTalk environment and the workflows developed using BizTalk can be exported as BPEL scripts. At the time of this writing, Windows Workflow Foundation (WF) does not directly support BPEL, but it may be noted that the constructs provided by WF are very similar to those in BPEL and many people in the community believe that direct support for BPEL is planned for the future.

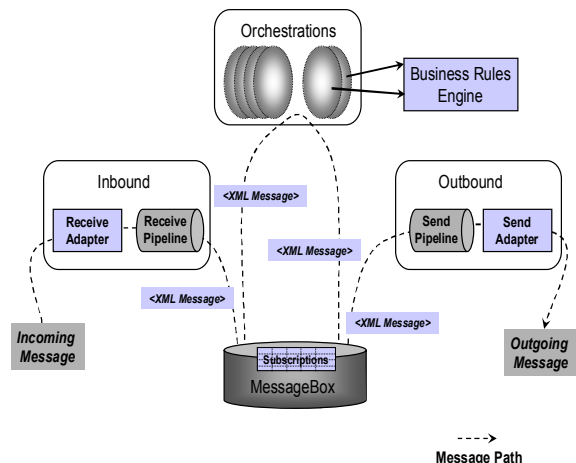


Fig.1 – BizTalk Server 2006 engine architecture (Src: [18])

BizTalk Server. Microsoft’s BizTalk server (see Figure 1) is a separate commercial product that is currently in its fourth version (2006). It makes use of the Microsoft .NET technology. It is integrated with management tools, support for Service-Oriented Architecture paradigm, through use of Web services and WS-* specifications and a business activity monitoring portal. The orchestration capabilities in BizTalk Server 2006 are focused on system-to-system communication, supporting business processes that depend on integrating diverse software.

Windows Workflow Foundation (WF). Windows Workflow Foundation (see Figure 2) is designed to provide a framework that lets any Windows application use workflow technology. Unlike BizTalk Server, with its focus on integrating independent systems, WF provides a general framework for creating applications that are themselves built around workflows. It aims to provide a singular engine for workflow execution for all applications built on the Windows platform. Over time, WF will become the common workflow technology used by Microsoft products, including the Microsoft Office System and others. In fact, the BizTalk release that follows BizTalk Server 2006 will include the ability to create WF workflows alongside its current orchestration capabilities. It is currently in its beta version and it is a core component of the next generation of the .NET Framework (.NET Framework 3.0).

BPEL Process Manager (OraBPM). Oracle’s Business Process Manager (OraBPM, see Figure 3) is a BPEL engine that provides a framework for designing, deploying, monitoring, and administering processes based on BPEL standards. It is developed in Java and hence is agnostic of operating system

platform. While it is a commercial product, it has a freely available developer license. It supports fault handling and exception management during both design time and run time. The engine consists of design components – JDeveloper BPEL designer and Eclipse BPEL designer plug-ins, deployment component, which is the BPEL server and management component – Oracle BPEL Console.

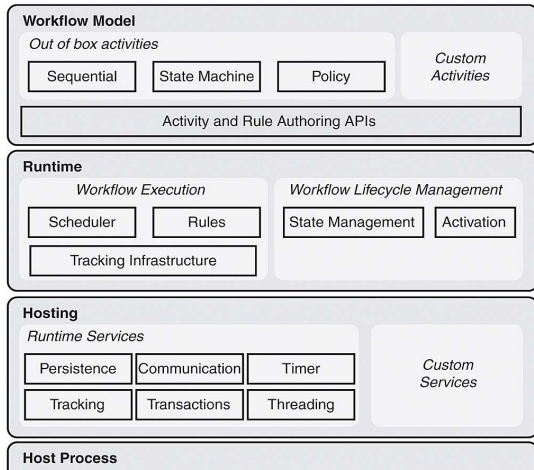


Fig. 2 – Windows Workflow Foundation engine architecture (Src:[19])

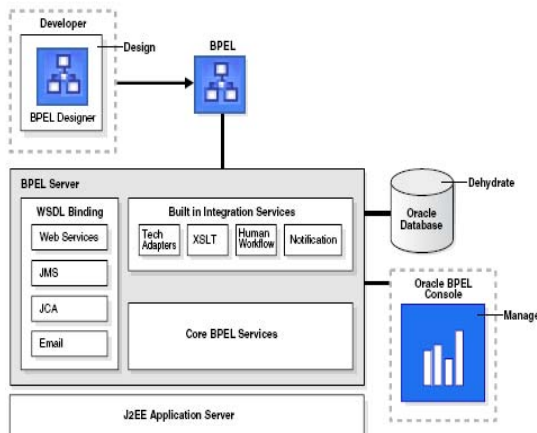


Fig. 3 – Oracle BPEL Process Manager architecture (Src:[20])

2.1 Using the Workflow Engines

All the three engines support a graphical user interface. BizTalk Server uses Microsoft’s XLANG internally to represent the workflow that is constructed using the GUI. It supports XPATH expressions and it also allows .NET components to be used within the workflow. It has a .NET programming interface too but using that requires knowledge about XLANG.

Workflow in OraBPM can be developed in two ways – one is using BPEL scripts directly, the other is using the graphical designer interface (added on to both JDeveloper and Eclipse software). The workflow developed using this drag-and-drop GUI, is translated into BPEL script. OraBPM also allows Java code snippets to be embedded within the workflow (using <bpel:exec> activity). In addition to the capability to communicate with Web services, support is provided for EJBs (Enterprise Java Beans).

Workflow in Windows Workflow Foundation enables reusable component development through extensibility points. Workflow in WF can be developed in any .NET language. Any .NET application can host a workflow, by creating an instance of the WorkflowRuntime class. Also, any .NET code can be included within a workflow.

2.2 Expressiveness and Extensibility

A particularly important aspect of "usability" is the ease at which a workflow concept or requirement can be represented -- which we refer to as expressiveness and extensibility. All three engines allow arbitrary code to be embedded in the workflow. WF allows the use of .NET code as well as the use of .NET libraries within a workflow. OraBPM allows Java snippets to be embedded, and also invocation of EJBs. BizTalk allows .NET components and code snippets to be used but it’s not as straightforward as in WF, since BizTalk has restrictions on variable definitions and requires the .NET assemblies used to be in the Global Assembly Cache (GAC). While the inclusion of custom code also allows workflows to be extensible to a certain degree, the custom code essentially defines the internal operation of a single activity (a “node” within the workflow). It would also be useful to create new workflow concepts – i.e. extensions to the workflow language. It should be noted that the BPEL language is extensible because new namespaces and elements can be added, but it would be difficult to capture all the capabilities of a programming language in a scripting language. None of the engines provide any specific support for extending the workflow language.

3. Requirements of Grid Workflow Engines

The discussion on the usability, expressiveness and extensibility in Section 2 has given the readers a general understanding of the engines. In this section, we enumerate the requirements of a Grid Workflow Engine. These requirements are based in part on and are refinements to the requirements identified by Slomanski [14] and Rana [8]. In later sections of this paper, we qualitatively (Section 5) and quantitatively

(Section 6) assess the degree to which each of the three engines satisfies these requirements.

Data Management. A workflow typically requires movement of (input or output) data from one task or process within the workflow to the other. Grid workflows are characterized by large amounts of data and data streaming. A Grid workflow engine must have the capability to efficiently and reliably handle the data flow within the Grid.

Job Submission and Management. A Grid user submits the jobs that he/she wants to run and specifies what to do with the results, in the form of a workflow. Hence, as a basic functionality, the workflow engine must be able to support protocols for remote execution such as GRAM or the new GGF OGSA Basic Execution Services (BES)/Job Submission Description Language (JSDL) [16][17].

Security. Security is a basic requirement for Grids because of their heterogeneous and collaborative nature. The user must have the necessary credentials to run a Grid workflow (and the jobs within the workflow). Also, the workflow engine must be able to communicate with secure services. The engine itself must be secured, to prevent malicious manipulation of information within a workflow.

User Interface with the Grid. A workflow is typically submitted by a user, and so the engine must be able to provide a seamless interface with the Grid. This could be via a Grid portal, Grid client application, or a Web service.

Scalability. The workflow engine must be able to scale, either the number of workflows running within an engine or the number of tasks within a workflow (or the number of nested workflows).

Heterogeneity and Interoperability. Heterogeneity is common in Grids and so Grid workflow engines must provide the same functionality on multiple platforms (ideally these would be interoperable). One of the important motivations behind adapting business workflow engines to Grids is to achieve this.

Tracking Capability (Monitoring and Logging). Tracking provides the ability to capture data and events that occur during the execution of a workflow. The user must be able to trace how a particular result has been obtained during any stage of the execution of a workflow.

Fault Tolerance. Grids are highly dynamic environments, and so the Grid workflow engines

must be able to handle a certain class of errors and faults. While transactional execution of workflows, i.e. having the workflow engine handle issues like check-pointing and rollback, may be desirable, it may be difficult in the Grid context because of activities that span multiple sites. Other types of fault tolerance include multiple attempts to communicate with a service and the ability to dynamically select the available service from a class of services (should some failure occur). Although fault tolerance is more of a useful feature than a requirement per se, it is something that the users will greatly benefit from and to that end, it arguably can be stated as a requirement.

Persistence. Grid workflows are usually long-running. A workflow engine that retained its state in volatile memory for that period would be susceptible to machine power outages. Hence, a persistent state architecture is desired where the workflow state will persist to a store while it waits for a response that may take some time.

4. Executing Grid Workflows on the Three Workflow Engines

Because Grid computing is increasingly relying on Web services, and the three commercial systems studied here are well-integrated with Web services, the amount of software that we had to write in order to adapt these commercial workflow engines for e-science was relatively minimal. To ensure a proper match with the requirements of an e-scientist, our methodology was that we chose to be driven by a small set of workflows that we desired to utilize Grid services, and then design and implement the necessary "glue" code to integrate the commercial workflow engines with the University of Virginia Campus Grid (UVaCG).

We implemented three representative grid workflows on the three workflow engines considered. One is a simple workflow as shown in Figure 4, where the input data is staged-in, a job is executed and the output data is staged out. This fundamental grid workflow was chosen to understand how easy or difficult it is to achieve basic Grid functionality. The second workflow is the parameter-space workflow shown in Figure 5. This workflow was chosen to test the scalability of the engines. The third workflow is shown in Figure 6. In this workflow, a job is run on the machine with the least CPU load, and the output from that job is moved to the machine with the most available disk space. This workflow was chosen to assess the ease of implementing workflows with dynamic requirements involving non-functional properties.

For each workflow type, we found that the amount of "glue code" was minimal and involved GridFTP and GRAM. The stage-in and stage-out operations invoke a GridFTP client program and the 'run job' operation invokes a GRAM client. Both clients were written in C#, and interoperate with both Windows and Linux servers. The machines used consisted of 7 windows boxes and 23 Linux boxes. Windows machines have GridFTP.NET and GRAM.NET servers [24] (.NET implementation of GridFTP and GRAM server) running on them. The Linux machine uses GT4. The "dynamic" workflow communicates with GT4's Monitoring & Discovery System (MDS) [25] to determine the least loaded machine and the machine with the most available disk space among the Linux boxes. UVaCG's own information service was used for determining the same information on the windows machines.

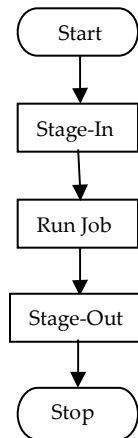


Fig. 4 – Simple workflow

On all the three engines, the graphical designer was used to develop these workflows. This involved dragging and dropping and connecting the necessary constructs. The client programs were invoked as executables on BizTalk Server and OraBPM, while they were invoked as libraries on WF. For the parameter-space workflow, all the three engines support the 'parallel' construct. For the dynamic workflow, all the three engines invoked the information services as they would invoke any Web service. The workflow also contains activities (code) for parsing the XML document returned by the information services and determining the machines that satisfy the non-functional properties during runtime. All the three engines support dynamic binding, i.e., all the three engines can dynamically decide which machine to talk to.

In summary, we found that we did not have to write code specifically to utilize the commercial engines to execute these e-science workflows: our group's pre-existing clients for GridFTP and GRAM

and/or the Globus Toolkit's java clients were sufficient to integrate these commercial engines with the Grid.

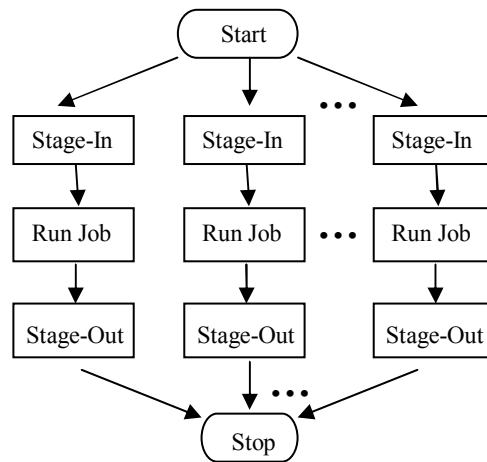


Fig. 5 – Parameter space workflow

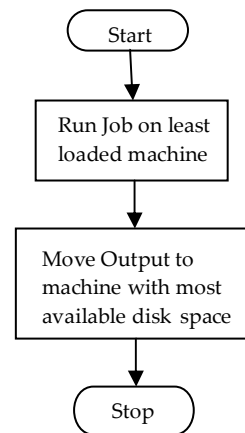


Fig. 6 – Dynamic workflow

5. Qualitative Evaluation

The qualitative evaluation is based on the requirements of Grid workflow engines discussed in the previous section.

Data Management. While workflow engines typically cause data to flow through the machine running the engine, this is highly inefficient for the typically large data movements used in Grids. However, the GridFTP [15] protocol allows this to be avoided via third-party transfer. The role of the workflow engine in handling data is to be able to invoke the GridFTP client with the appropriate source and destination addresses.

Job Submission and Management. Similar to data management, the engines invoke the GRAM client in

order to handle job submission and management. While all the three engines have the option of invoking the client as either an executable or as a web service, WF has an additional option of invoking it as a library, since the client program and WF have the .NET platform in common.

Security. The user credentials and verification are handled by the GRAM and GridFTP client programs, and so the workflow engine is relieved of that burden. However, all three engines are capable of communicating with secure services. The complexity of the process varies with each engine. In BizTalk Server, data integrity and authentication can be ensured (using digital certificates) by adding encoding or decoding components to the send or receive pipelines. OraBPM allows WS-Security authentication headers to be passed to WS-Security secured web services. WF is the simplest to use by leveraging all the capabilities of WSE 3.0, including the security features, just as any client or service on the .NET platform. Authorization models also differ between the engines. BizTalk Server uses Windows authorization, along with an Enterprise Single Sign-on system. Authentication and authorization to OraBPM engine are handled by an 'Identity Service' which is a thin web service layer on top of the Oracle Application Server 10g security infrastructure. WF requires that the runtime engine be hosted within another program. Hence authorization in WF must come from the host program.

Interface with the Grid. Since browser-based Grid portals are the most convenient entry point to the grid, it would be useful to have a similar interface to workflows. This is feasible with all the three engines, as they have can be accessed from a browser. OraBPM provides a console in the form of a Web portal, where we can submit already deployed workflows. WF workflow can be hosted in any .NET application, so ASP.NET web applications can submit, control and monitor WF workflows. Similarly, instances of workflows deployed on the BizTalk Server can be triggered by a range of activities including dropping an XML file in a folder, or via HTTP or SOAP. However, a 'Web interface' is not available for developing the workflows and deploying the workflows into these engines.

Heterogeneity and Interoperability. BizTalk Server and WF engines can only run on Windows platform. OraBPM has both Windows and Linux versions available. However, this does not restrict their application in Grid as we observe from our implementation. This is because the GRAM and GridFTP client programs are capable of

communicating with any platform. One important feature that paves way for interoperability is that, the workflows developed using any of these three engines can be published as a Web service. Hence, workflows from any engine can be nested in workflows from another engine.

Tracking Capability (Monitoring and Logging). BizTalk Server allows querying the created workflows, the messages that were exchanged, and their status. The logging and monitoring facilities are available only on the site where BizTalk is running.

Oracle BPEL Process Manager provides a portal console with information about previously submitted workflows. The portal also provides auditing, a visual log and debugging information. OraBPM also supports 'sensors', which allow the specification of BPEL activities, variables and faults to be monitored during runtime.

WF does not come with any separate monitoring or auditing component. However, it includes several runtime services that can be plugged into the WorkflowRuntime. Of these, the "tracking services" allow the tracking of certain workflow events (whether the workflow has been loaded, persisted, terminated with exception, completed, etc), activity events (indicating the status of an activity within the workflow), and user events, i.e., any customized events that a user may want to track (such as the result from calling a web service).

Fault Tolerance. All the three engines support transactions in workflows. Long-running transactions, which span different remote partners and for which the engine has no control over the remote resources, are supported via "Compensation Blocks". Compensation blocks allow the user to explicitly specify what operations need to be performed on roll-back. Different parts of the workflow can have different compensation blocks, by using a scoping operator to define the boundaries of a transaction. In addition to this, BizTalk Server and WF support atomic transactions. Atomic transactions are more rigorous than long-running transactions and preserve the four classic transactional properties: atomicity, consistency, isolation and durability. Atomic transactions are similar to those handled by the Windows Distributed Transaction coordinator. In OraBPM, the server can execute any embedded Java code within its JTA (Java Transaction API) transaction context and if the java code calls any other resource, such as EJBs, the transactional context is automatically propagated.

Persistence. All the three engines allow the state of the workflow to be persisted in backend databases.

6. Quantitative Evaluation

The timing information for the execution of the simple workflow of Fig. 4 on each of engines is shown in Fig. 7. The baseline is a simple C# program which implements the three activities (stage-in, run job and stage-out), using the same client applications used by the engines. The fastest engine was WF at 3.417 seconds (0.2% slower than the baseline), and the slowest was OraBPM at 4.005 seconds (17% slower than the baseline). WF incurs the least overhead as it invokes the GRAM and GridFTP clients as libraries, while the other two engines are only able to engage the Grid functionality by invoking them as executables. Overall, The differences in the times can be attributed to the cost of instantiating the workflow, varying efficiency of execution, varying logging and bookkeeping functionalities and the overhead of the engines themselves.

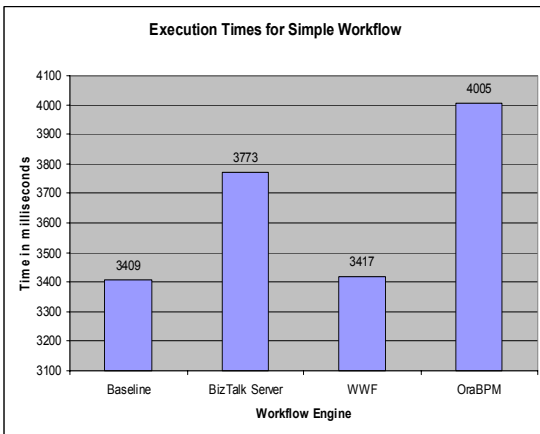


Fig. 7 – Execution time for simple workflow

The timing information for the execution of the “dynamic workflow” is shown in Fig. 8. We can observe that it follows the same trend as Figure 7, implying that there is no difference in the trend among the engines with regard to communicating with Web services and dynamic selection of Web service instances.

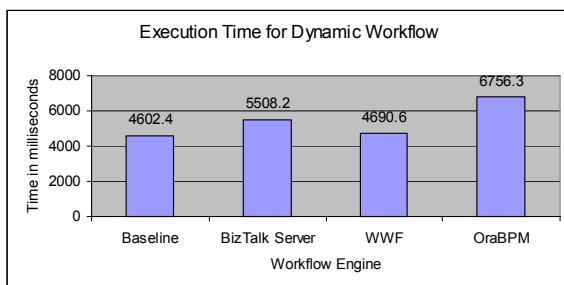


Fig. 8 – Execution time for dynamic workflow

The timing information for the execution of the parameter space workflow is shown in Fig. 9. The number of parallel branches of the parameter space was increased up to 500 and they were distributed over 30 nodes. The baseline again is a C# program which implements the parallel branches using threads. The workflow is implemented in each of the three engines using the ‘parallel’ construct available. This was to test the scalability and robustness of the engine. We observed creating these workflows using the GUI designer of both the WF and OraBPM engines failed around 400 parallel branches, while BizTalk was more robust. The 500 branch versions of these workflows were created “in-code”. We can observe from the figure that there is a lot of deviation from the baseline. This suggests that the ‘parallel’ construct provided by these engines do more than just spawn a thread. From what we observed from the order of output, perceived synchronization of all the parallel branches contribute to the overhead. From the data, WF appears to be more scalable than the other two. Also, once the GUI for BizTalk Server fails, it is more difficult and tedious to directly write the code for this engine when compared to WF and OraBPM.

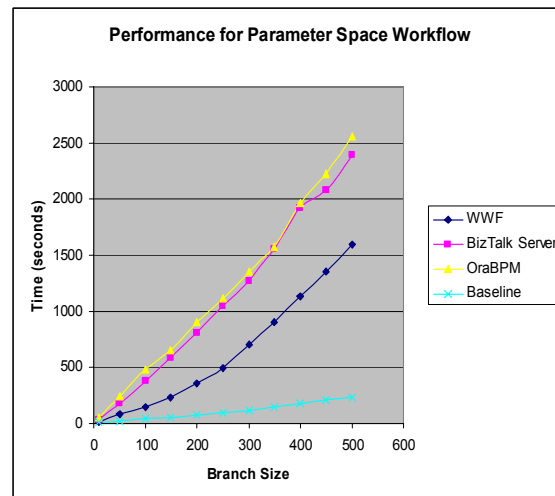


Fig.9 – Execution times for parameter space workflow

7. Conclusion

Workflows are increasing in importance for e-scientists, and many workflow engines are being developed specifically for the Grid. In this paper, we studied three commercial workflow engines and determined experimentally that they each satisfy the requirements of Grid workflow engines for e-science. Each system has its advantages: OraBPM is free and applicable for those systems in which Java/Linux is desired; BizTalk Server is a mature, proven product

with a robust feature set; WF is free and Microsoft's vision for workflow looking forward across much of the product suite. One of the perceived biggest disadvantages is the cost of these systems (and/or the HW/SW systems on which they run), although we observe that e-scientists are often academics, who can generally acquire this software at greatly reduced prices or no cost at all. Overall, we believe that each of these three systems will continue to be developed for e-business into a full-featured, supported, and highly-optimized workflow engine that can be readily utilized for e-science.

References

- [1] A. Paventhan, K. Takeda, S.J. Cox, and D.A. Nicole. "Leveraging Windows Workflow Foundation for Scientific Workflows in Wind Tunnel Applications", IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow'06), Atlanta, GA.
- [2] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock, "Kepler: An Extensible System for Design and Execution of Scientific Workflows", 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04)
- [3] Shalil Majithia, Matthew S. Shields, Ian J. Taylor, and Ian Wang, "Triana: A Graphical Web Service Composition and Execution Toolkit", Proceedings of the IEEE International Conference on Web Services (ICWS'04), pages 514-524. IEEE Computer Society, 2004
- [4] K. Amin, Gregor von Laszewski, M. Hategan, N.J. Zaluzec, S. Hampton, A. Rossi, "GridAnt : A Client-Controllable Grid Workflow System", Proceedings of the 37th Hawaii International Conference on System Sciences, 2004
- [5] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10) San Francisco, California, August 7-9, 2001
- [6] Ali Shaikh Ali, Omer F. Rana and Ian J. Taylor, "Web Services Composition for Distributed Data Mining", Workshop on Web and Grid Services for Scientific Data Analysis (WAGSSDA), Oslo, Norway, 2005.
- [7] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, Miron Livny, "Pegasus : Mapping Scientific Workflows onto the Grid", Across Grids Conference 2004, Nicosia, Cyprus
- [8] Omer Rana, "Creating and Managing Distributed Scientific Workflows: Techniques and Tools", tutorial, EuroPAR 2005 Conference
- [9] I. Foster, J. Voeckler, M. Wilde, Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation", 14th International Conference on Scientific and Statistical Database Management (SSDBM'02)
- [10] Jia Yu and Rajkumar Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005.
- [11] WS-BPEL specification, "www.software.ibm.com/software/developer/library/ws-bpel.pdf"
- [12] Satish Thatte, XLANG specification, "www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm"
- [13] Frank Leymann, WSFL specification, "www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf"
- [14] A. Slomanski, "http://www.extreme.indiana.edu/swf-survey/IntroductionToWorkflowsInGridsAndPortals_GGF9_2003.ppt"
- [15] W. Allock, GridFTP Protocol Specification Global Grid Forum Recommendation GFD.20, March 2003.
- [16] JSDL, "www.ggf.org/documents/GFD.56.pdf", GGF recommendation GFD-R-P.056
- [17] OGSA-BES specification, www.ggf.org/mail_archive/ogsa-bes-wg/2005/07/pdf00000.pdf
- [18] S. Woodgate, et al, "Microsoft BizTalk Server 2004 Unleashed", SAMS Publishing.
- [19] P. Andrew et al., "Presenting Windows Workflow Foundation Beta Edition", SAMS Publishing
- [20] Oracle BPEL Process Manager (OraBPM) Developer's guide. "http://download-west.oracle.com/docs/cd/B14099_16/integrate.1012/b14448.pdf"
- [21] M. Humphrey and G. Wasson, "The University of Virginia Campus Grid: Integrating Grid Technologies with the Campus Information Infrastructure", 2005 European Grid Conference (ECG 2005), Feb 2005.
- [22] Foster, I. and Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications, 11 (2). 115-128. 1997.
- [23] M. Humphrey and G. Wasson. Architectural Foundations of WSRF.NET. International Journal of Web Services Research. 2(2), pp. 83-97, April-June 2005.
- [24] J. Feng, L. Cui, G. Wasson, and M. Humphrey, "Toward Seamless Grid Data Access: Design and Implementation of GridFTP on .NET", The 6th IEEE/ACM International Workshop on Grid Computing, Nov 2005
- [25] Jennifer M. Schopf, Mike D'Arcy, Neill Miller, Laura Pearlman, Ian Foster, and Carl Kesselman, "Monitoring and Discovery in a Web Services Framework: Functionality and Performance of the Globus Toolkit's MDS4", Argonne National Laboratory Tech Report ANL/MCS-P1248-0405, April 2005.