

Towards Proxy Workflow Execution in Environmental Research: Application to Vortex2

Beth Plale, Chathura Herath, and Eran Chinthaka
School of Informatics and Computing
Data To Insight Center of Pervasive Technologies Institute
Indiana University Bloomington

Workflow systems embody a programming model whose strength is in reducing the complexity of non-uniform, computationally rich scientific investigation. The class of workflows that we study, which are fairly characteristic of model driven analysis in the geosciences, are fairly squat in length and mapspan, but are computationally intense and generate and consume large amounts of data (Ramakrishnan and Plale 2010). A pattern of one such workflow is shown in Fig. 1, where megabyte (MB) inputs at the beginning of the workflow turn into gigabyte (GB) inputs and outputs then are followed by MB outputs as the workflow progresses. CPU consumption starts at 64 cores, ramps to 512 cores at its most computationally intense phase, then drops down to a low computation phase as analysis products are produced. This is accomplished by a workflow containing 7-10 tasks, no loops, and localized (within node) parallelization.

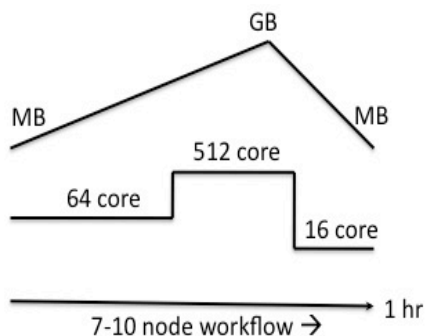


Fig 1. Pattern of atmospheric forecast workflow.

Workflow systems usually assume a particular platform for execution, often TeraGrid or a dedicated cluster. The Trident Scientific Workflow Workbench (Barga et al. 2008) was developed to work with Windows Workflow Foundation on Windows servers, while Pegasus (Deelman et al. 2005) and Swift (Wilde et al. 2009) execute on HPC

resources. Swift's back-end load balancer, Falkon, and custom OS, ZeptoOS, for instance work together to launch new applications on a BlueGene supercomputer. The LEAD workflow system has used a BPEL-based workflow engine, GFAC (Kandaswamy et al. 2006) and Globus Gram (Globus 2010) to submit workflow tasks to schedulers on HPC resources. With the introduction of solid performing Windows cluster operating system, Windows HPC Server 2008, we undertook a study to execute parts of a Linux-based workflow on an HPC Server 16-core (duo core, quad processor) machine. There are several ways to run Linux applications on Windows: port the application, invoke the Linux application executable through a Linux emulator such as Cygwin, and finally, invoke Linux applications from logic on the Windows side that supports what we call workflow execution proxying.

At Supercomputing 2009 we demonstrated workflow execution of the Weather Research Forecast model (WRF), version 3.1 through the Trident Scientific Workflow Workbench, a Windows desktop workflow tool. WRF results were passed to a set of NCAR NCL scripts that were executed through Cygwin. LEAD II takes this one step further in its spring support for the NSF-funded Vortex2 field experiment to study tornadoes. Over the six weeks in late Spring 2010, the LEAD II system generated 252 short-term forecasts and created 9000+ visualization products. In support of this effort we added the feature of delegating a sub workflow that can be passed to and invoked by a Linux workflow orchestrator, Apache ODE, which executes the subworkflow on TeraGrid. Using the Trident scientific workflow workbench as our driving workflow platform, we are developing an approach to scheduling and resource management across cloud and non-cloud

resources to **demonstrate the feasibility of a proxy (hybrid) workflow execution model** that seamlessly utilizes cloud resources including computation, application-specific data sets, and data storage.

The specific question we examine is one of: *How do we build a proxy workflow model using an existing workflow system as the entry point?* There are two approaches: the subworkflow approach, and the component-as-workflow approach. The *subworkflow approach to the proxy workflow model* has the primary workflow tool handing off a subworkflow to a subordinate workflow orchestration tool. Using this approach eases the integration of existing system functionality with the Trident workflow system. A benefit of this approach is that there are a limited number of extensive workflow systems, and if you solve for one, chances are such solution may be applicable for many gateways that may use that workflow system. That means we can capitalize on the domain specific issues that were already solved in the existing gateway infrastructure. There are also certain scientific workflow systems that might be hard to capture using Windows workflow systems. For example the parametric sweeps for millions of parameters required for earthquake simulation requires managing millions of small jobs and could overwhelm a system like Trident. For such cases a system like Pegasus that hands over the parametric sweep to a Condor system would be very efficient.

The component-as-workflow approach, which proxies each component individually, would give fine-grained control over resource selection so our proxy may appear to be a resource selection proxy. This has features in common with the actor model of Kepler and in a certain sense where specialized activities may select the underlying resources and build the contextual information necessary to launch the job. This approach requires different proxy activities for handling each unique resource (e.g., TeraGrid, cloud) and service components (e.g., web service, cloud components). Alternatively, we might have one big proxy activity that is configurable and captures all these scenarios. We will

study the options in more detail before choosing an approach.

A key component in our suite of tools is Sigiri (Chinthaka et al. 2009), a resource management tool for deploying jobs (i.e., tasks or subworkflows) to heterogeneous platforms. It is designed to extend to new job descriptions languages and currently supports JSDL (Job Submission Description Language) (Anijomshoaa et al., 2004) and RSL (Resource Specification Language) (Globus URL). A Trident activity designed to interact with Sigiri enables scientists to submit jobs to different computational resources within a Trident workflow. Once the required computational resource is selected (either by the user or by a quality of service optimization algorithm) this activity will pass this information, together with job descriptions and credentials, to Sigiri which will use appropriate daemon to submit this job to the computational resource selected. This activity can also be used to continuously monitor the progress of the submitted job, using the Sigiri Web services API, and report the state transitions.

Scientific workflow systems provide a means to execute a complex sequence of activities without intensive user intervention and in ways that support flexible reordering or reconfiguration of the workflow. Often these workflows encapsulate one or more compute intensive jobs and require, large-scale systems to execute in an efficient and timely manner. As large-scale compute resources become more abundant (e.g., Amazon Web Services, Azure, TeraGrid, Open Science Grid), workflow systems should be capable of working with all possible and available resources. These multiple options maximize turnaround time but throw challenges at workflow adoption by presenting multiple non-interoperable access interfaces. Even though providing a higher-level abstraction to cover all possible options will be an ideal solution, it will be quite a challenge and a time consuming task as the underlying technologies are still emerging.

References

Altintas, I., Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludäscher and Steve Mock 2004: Kepler: An Extensible System for Design and Execution of Scientific

Workflows, *SSDBM*, pp. 21-23.

Anjomshoaa, A., F et al. 2004: Job Submission Description Language (JSDL) Specification v0.3, *Global Grid Forum*.

Barga, R. Jackson, J. Araujo, N. Guo, D. Gautam, N. Simmhan, Y. 2008: Trident Scientific Workflow Workbench, *IEEE Fourth International Conference on eScience*, IEEE Computer Society Press, pp. 317-318

Chinthaka, Eran, Suresh Marru, and Beth Plale 2009: Sigiri: Towards A Light-Weight Job Management System for Large Scale Systems, *Indiana University Computer Science Technical Report TR681*, Aug 2009.

Deelman, E. et al. 2005. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems, *Scientific Programming Journal*, Vol 13(3), pp 219-237

Globus, The Globus Resource Specification Language (RSL), Globus v3.2 Specification,

http://www.globus.org/toolkit/docs/3.2/gram/ws/developer/mjs_rsl_schema.html.

Globus Toolkit, <http://www.globus.org/toolkit>

Kandaswamy, G., L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon 2006: Building Web Services for Scientific Grid Applications. *IBM Journal of Research and Development*, 50(2/3) pp. 249-260

Ramakrishnan, Lavanya and Beth Plale 2010: Multidimensional Classification Model for Scientific Workflow Characteristics 2010: *1st Int'l Workshop on Workflow Approaches for New Data-Centric Science, co-located with ACM SIGMOD Int'l Conference on Management of Data*, Jun 2010.

Wilde, M. et al. 2009. Parallel scripting for Applications at the Petascale and Beyond, *Computer*, IEEE Computer Society Press, Nov.