

A Grid-enabled Workflow System for Reservoir Uncertainty Analysis

Emrah Ceyhan, Gabrielle Allen, Christopher White and Tevfik Kosar
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA 70803, USA
{eceyhan,gallen,cwhite,kosar}@cct.lsu.edu

ABSTRACT

Reservoir uncertainty analysis is significant for petroleum engineers for predictions of reservoir performance. However, analysis of reservoir performance uncertainty is challenging because large amounts of data must be transferred efficiently, reliably and securely between sites, and thousands of simulations are executed across different resources. There are several steps in conducting reservoir performance prediction, including: (a) transferring input files to remote resources, (b) running thousands of simulations in different scheduler systems, (c) monitoring the jobs, (d) transferring output files from remote sites to the local system, and (e) post-processing to determine whether simulations have resolved uncertainties adequately. This whole process may have to be repeated as new data are obtained, or if uncertainty thresholds change. Therefore, it is essential to automate end-to-end processing for this complex, composite application with many tasks that are executed in a specific order. We implemented an end-to-end automated system for reservoir uncertainty analysis using Grid technologies such as Condor-G, DAGMan, and Stork. This paper describes the requirements, design and implementation of such a system.

Categories and Subject Descriptors

D.1.3 [Software]: Programming Techniques - *Concurrent Programming*

General Terms

Performance, Design, Reliability, Experimentation

Keywords

Reservoir modeling, uncertainty analysis, end-to-end-processing, workflows, data management, Stork, distributed systems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CLADE'08, June 23, 2008, Boston, Massachusetts, USA.
Copyright 2008 ACM 978-1-60558-154-5/08/06 ...\$5.00.

1. INTRODUCTION

Performing large-scale science has become increasingly complex as scientists have started using distributed computing tools to enable end-to-end analysis and to automate the computational experiments. As the complexity of scientific applications increases, computational experiments have large scale data sets, require more computational resources, and scientists may be distributed geographically. Therefore, managing large datasets across distributed resources is needed to enable end-to-end analysis for many scientific applications.

End-to-end analysis aims to decrease human intervention during execution, and to automate the end-to-end processing for scientific applications. As Grid computing has evolved over the past few years, it has become easier to solve complex problems on distributed resources, as well as manage and process large datasets. Data generated across different resources, for example, may comprise terabytes or even petabytes. For many scientific applications using distributed resources, input data must be transferred to the computation sites and the results sent back to the investigators; the investigators can then extract summary data, visualize results, and in general analyze data from the computational experiment. Commonly, component tasks must be executed in a certain order as part of an application. For example, data must be staged in before computation starts; once computation completes, results are moved back to local system to be analyzed. As another example, we may have to repeat a simulation (which may consist of a sequence of tasks) until we get an error within desired limits. This kind of end-to-end processes need to be managed and coordinated to process and analyze large datasets generated across different resources. Grid computing technologies [1] are considered to be very efficient in providing tools to utilize resources distributed geographically.

These issues have arisen and been addressed in a project (UCoMS) to integrate sensors, computation, and modeling for uncertainty prediction and characterization of oil and gas reservoirs. Background information on UCoMS project is presented in Section II. The components of the required reservoir modeling workflow are outlined in Section III. Relevant research and applications in grid computing are discussed in Section IV. The details of the workflow implementation are then enumerated in Section V, before presenting the results and conclusions.

2. UCOMS BACKGROUND

Institutions from different universities in Louisiana devel-

oped UCoMS project whose main objective is to develop and deploy a Ubiquitous Computing and Monitoring System (UCoMS) for oil/gas exploitation and management.

This project helps researchers at different institutions in Louisiana understand issues in the areas such as wireless networked systems, and grid computing, and their application for large scale petroleum engineering applications. The solutions obtained from this project will help researchers facilitate drilling and operational data logging and processing, on-platform information distribution and displaying, infrastructure monitoring/intrusion detection, seismic processing and inversion, and management of complex surface facilities and pipelines [13].

Developed in collaboration of Louisiana State University (LSU), University of Louisiana at Lafayette (ULL), and Southern University (SUBR), UCoMS aims to address issues such as:

- The computational challenges faced by compute-intensive simulations for reservoir uncertainty analysis that requires thousands of simulations and deals with terabytes, and even petabytes of data.
- The development of a prototype wireless sensor network (WSN) infrastructure to collect and process real-time data from production locations.

Many sensors have been considered in the UCoMS project for different uses on the oil platforms/rigs and along the pipeline. The data acquisition rate depends on the types of sensors. For example, ultrasonic sand detection sensors have a lower data rate while the DWD sensors yield a very high data rate. The sensor deployment and data management also vary according to sensors and applications. The sensors used during the well drilling process in this project may take up to 2000 measurements per second and generate 350 GB data per day.

One of the major objectives of this project is to run compute-intensive simulations and use of a large sets of data and real-time processing. Since large-scale data management is involved for reservoir uncertainty analysis, ResGrid is used to address this issue and execution support. The ResGrid makes use of the Grid Application Toolkit (GAT) [14] to get accesses to grid services. With the help of GAT API, data grid tools are used so that data distributed geographically can be handled efficiently.

3. STEPS INVOLVED IN UCOMS END-TO-END PROCESSING

Reservoir uncertainty analysis via simulation is used to improve reservoir performance prediction. There are four steps that are commonly performed for uncertainty analysis. These steps are [17]:

- Seismic inversion
- Reservoir modeling
- Flow simulation
- Post processing

The first step in reservoir uncertainty analysis is to combine seismic and well data (perhaps using a sparse spike inversion and correlated wavelet extraction) to estimate reservoir

geometry and compute a preliminary estimate of net rock volume and fluid contents. The required seismic, petrophysical, and well data sets may be very large.

The second step is reservoir modeling at a scale appropriate for flow modeling, called a geomodel. This step down-scales the seismic inversion output into simulation grid and enforces spatial correlations and decorating models with stratigraphic subseismic structure so that reservoir simulation can evaluate flow effects of subseismic heterogeneity. After model-based inversion, there is an ensemble of models created for each trace on seismic grid.

Flow simulation is used to predict performance of these geomodels. There are several steps involved in a flow simulation, which consist of [17]:

- Generation of geostatistical realizations to sample the uncertainty of geological parameters; in uncertainty analysis there are many realizations
- Combination of geology, geostatistical realizations, fluid and flow parameters, along with well locations and other engineering factors to constitute a base model, which must be “loadable” into the flow simulator; there are many loadable models
- Simulation of the ensemble of loadable models to obtain production predictions and recovery factors for a chosen recovery process or development program
- Computation of economic performance indicators, uncertainty, risk, or technical measures

Each model is a combination specified by the base model, a geostatistical realization, and uncertain factors (rock–fluid properties, well descriptions, etc.) sampled at different levels. Each loadable model requires a flow simulation to predict performance. After flow numerical simulation, post processing is performed. Post processing may include response models, sensitivity analysis, optimization, and many other steps.

3.1 Challenges for UCoMS End-to-end Processing

There are several challenges in grid-enabling end-to-end processing of UCoMS. One of the challenges to manage simulation data efficiently. The loadable models comprise large scale datasets, such as geological and geophysical data as well as well-logging data. These datasets could be terabytes or even petabytes and they could be distributed geographically across different sites. In a reservoir uncertainty analysis process, it is common to use ensembles of dozens to hundreds of loadable models. These models might be updated dozens to hundreds of times. The results dataset of a single reservoir simulation (with 10^7 cells, and 3 state variables and 2 model variables to be passed per cell — as used in an Ensemble Kalman Filter) is 400 MB. Even with only 100 ensemble members and 100 data assimilations, the required transfer is 4 TB. In practice, larger models and ensembles and more frequent data assimilation may be needed, pushing the data transfer requirements upward. Meanwhile, drilling processing and real-time monitoring are also data-intensive. During drilling processing, there are multiple gigabytes of data produced even in a short period. Petroleum engineers would view and analyze the drilling processing through these data, and then make decisions in real time. It is challenging to

store, share, retrieve, analyze, and visualize these streaming data.

Another challenge for this project is to run thousands of simulations distributed across multiple heterogeneous clusters numerous times, with different loadable models to assess the impacts of uncertainty.

Finally, we must implement the end-to-end system. Since there are many tasks involved in predicting reservoir performance, reservoir engineers have to run these tasks manually to get the results. For example, these tasks could be transferring input files to different remote sites, submitting jobs and monitoring them as well as getting the results from remote sites and finally perform a post processing. However, it could be very tedious to run these tasks manually [17]. Therefore, we need an automated end-to-end system to enable engineers to run these tasks in a more efficient and reliable way.

4. RELATED WORK

In this section, we present the work related to the UCoMS project. We first discuss several uses of Grid Computing in Sensor Networking Applications. Afterwards, we explain data management approaches in Sensor Networks and then applications that employ the approach we have used for end-to-end analysis for the UCoMS project. Finally, we explain several workflow management systems.

4.1 Grid-Enabling Sensor Networking Applications

Dany Huges et al [2] at Lancaster University in UK developed a wireless sensor network system for flooding monitoring and warning system. This system has several characteristics such as connecting remote Grids for computation intensive flood modeling and performing onsite flood modeling. Their system called GridStix [3] utilizes powerful embedded hardware, heterogeneous wireless networking and Grid middleware to implement an adaptable wireless sensor network. This Grid middleware has the capability of allowing nodes to transfer data to remote fixed Grid and perform local Grid computations.

Another project was developed in College London by Richards et al [4] for urban air pollution monitoring using Sensor Grids within the Discovery Net. The Discovery Net allows users to connect and to use data sources efficiently. The main goal of the Discovery Net is to develop an advanced generic computing infrastructure so that it can support real-time processing, interpretation, integration, visualization and mining of large-scale data that have been produced from distributed high throughput devices. Much of these data are generated by GUSTO high throughput pollution monitoring sensors. Data is generated every two seconds and then uploaded to remote Grid storage resources through Grid services for post-analysis.

4.2 Data Management Approaches in Sensor Networks

There are several data management approaches developed in sensor networks such Directed Diffusion [5], Cougar [6], and TinyDB [9]. These approaches use a declarative language to specify queries on data. They also support in-network processing in order to decrease data within the network. These approaches give the functionality to be able to move filters for continuous queries into the network so

that processing data can be handled closer to data source, thereby saving energy significantly. This way, there would not be too much communication traffic, as the end result needs to be communicated. Directed Diffusion and TinyDB adopted the notion of the sensor network as a database. These two approaches assumed that intelligence is placed at the sensors and that queries are moved into the network. Direct querying for these approaches is very efficient since processing data is handled closer to data source, thereby saving energy significantly. In contrast, TinyDB assumed that intelligence is placed at the edge of the network, while maintaining the sensors within the center of the network.

4.3 Similar End-to-end Processing Systems

Kosar et al [7] designed and implemented a fault-tolerant system so that astronomers can process large amount of images utilizing idle CPUs, commodity clusters and grid resources. Their system handles transferring the images to the computation sites, processing them, and transferring the output data to the local system to analyze it. The system is also capable of handling all the failures that may arise during the process. They used Condor as their computation scheduler to schedule the jobs on the compute cluster and Stork for scheduling the data transfer. Since there are dependencies between tasks, they used DAGMan to handle dependencies. Using this system, they processed three terabytes of Digital Palomar Sky Survey images using idle grid resources across three different clusters.

Another similar application was developed by Kola et al [8] for distributed video processing and off-site replication. In different areas such as bio-medical engineering, educational research and geology, it is essential to process large amounts of video and make them available at different locations for post-analysis. Since these applications involve large scale data, it brings several challenges. In this sense, they designed and implemented a fully automated system that processes videos and handles off-site replications in a fault-tolerant way. The system transfers the videos to the computation sites, processing them and transferring the encoded videos to the destination. Similar to the previous application, they used Stork for transferring data, Condor as the computation scheduler and DAGMan to handle dependencies.

4.4 Other Workflow Management Systems

Pegasus [20] is a workflow management system designed to enable an application so that it can utilize resources at multiple locations efficiently. Pegasus allows scientists to build workflows in an abstract form without worrying about the details of the underlying CyberInfrastructure. Today, Pegasus is used in a number of applications such as astronomy, biology, earthquake science, and gravitational-wave physics. Pegasus performs a mapping from an abstract workflow to concrete workflow. An abstract workflow portrays the scientific analysis along with the data used and generated, but does not provide information about the resources needed for execution. A concrete workflow is an executable workflow, which includes details of the execution environment.

Triana [18] is a workflow system developed at Cardiff University. It has a graphical workflow-based environment which handles a set of tasks distributed across several resources. The distributed components considered in Triana is based on the notion of group units and two distribution com-

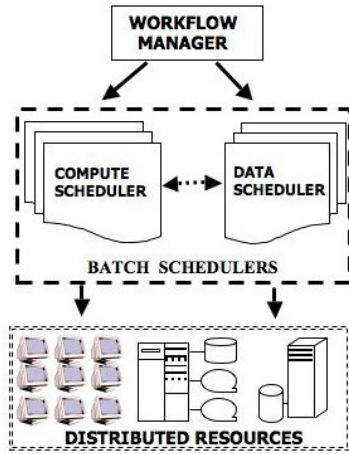


Figure 1: Model for Workflow Execution

ponents have been implemented: parallel and peer-to-peer. Tasks in parallel component in Triana can be dynamically allocated to available resources in an efficient manner. The peer-to-peer system depends on data being passed between hosts. Triana is a two layered application which comprises of Triana GUI and Triana Service. Triana GUI provides an easy way to use graphical user interface. It is very simple for users to create workflow in Triana for their purposes. Triana Service, on the other hand, represents Grid operations such task allocation, job submission and data transfer.

Taverna [19] is the workflow manager for the MyGrid [21]. Taverna and the MyGrid middleware are designed to support silico experiments in biology. Taverna is primarily designed to help scientist develop and execute bioninformatics workflows on the Grid. Taverna is capable of providing users a multi-window environment so that users can create workflows, select available resources, and execute and monitor the workflows.

Kepler [22] is similar to Taverna workflow system. It is a very popular workflow system for composing scientific applications and execute them efficiently in different resources. Kepler is derived from Ptolemy [23] and has the capability to invoke web services for accessing to remote resources and services.

5. END-TO-END SYSTEM FOR THE UCOMS PROJECT

We designed and developed an automated end-to-end system for the UCoMS project utilizing Grid technologies. Our system for the UCoMS project consists of several characteristics such as:

- Automation
- Reliability
- Separation of computing and data tasks
- Running jobs on heterogeneous batch systems
- Visually monitoring workflow progress
- Increased performance

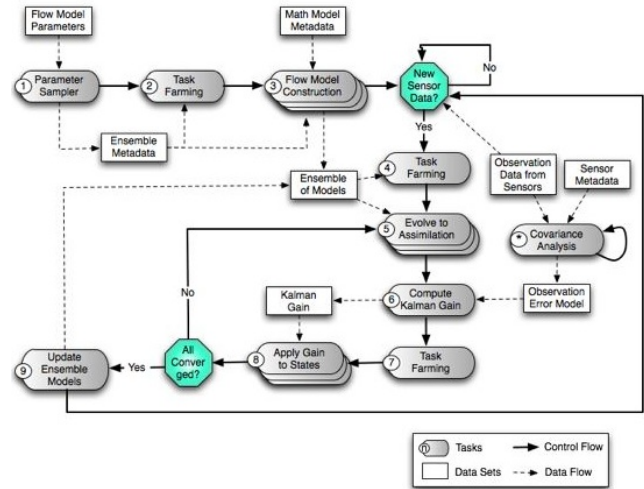


Figure 2: Abstract UCoMS Workflow

Since there are many tasks that need to be implemented in a specific order for the UCoMS project, the main goal of this project was to automate end-to-end processing so that tasks can be run without human intervention in a fault-tolerant manner. We used DAGMan as our workflow manager to implement a system in an automated manner. Another objective of this project was to have a reliable system. Our system is also reliable. In the case of failures, the workflow does not fail as a whole. Instead, it tries to execute only the failed tasks. It is also important to treat data placement jobs differently from computational jobs because they may have different semantics and different characteristics. We have used Condor and PBS for computational jobs and Stork for data placement jobs. Our system also allows a user to utilize heterogeneous resources at multiple locations. A user, for example, may want to run jobs in different batch systems such as Condor, PBS, and LoadLeveler. We used Condor-G to submit tasks to different batch schedulers which run Globus on their head nodes. A user may want to visualize the workflow while it is running. We implemented a web interface using a modified version of DAGMan that allows the user to see the status of each task on a web browser while the workflow is executed.

Figure 1 illustrates the workflow execution for the UCoMS project in an abstract model. In an abstract model, the workflow is represented without referring to specific Grid resources for task execution. As can be seen in this figure, a user starts the workflow manager and then the workflow manager separates the computing and data tasks. After that, jobs are distributed to remote resources to be run on different batch systems.

In our implementation, we used DAGMan as the workflow manager. The workflow manager then separates computing and data tasks. Computing tasks are distributed to remote sites through Condor-G and data tasks are managed through Stork. When a user initiates the dag file, jobs are distributed to different batch systems such as PBS and Condor through Condor-G.

Figure 2 illustrates an abstract workflow for the UCoMS project. This workflow is independent of Grid resources. It simply gives the task ordering. As can be seen from this

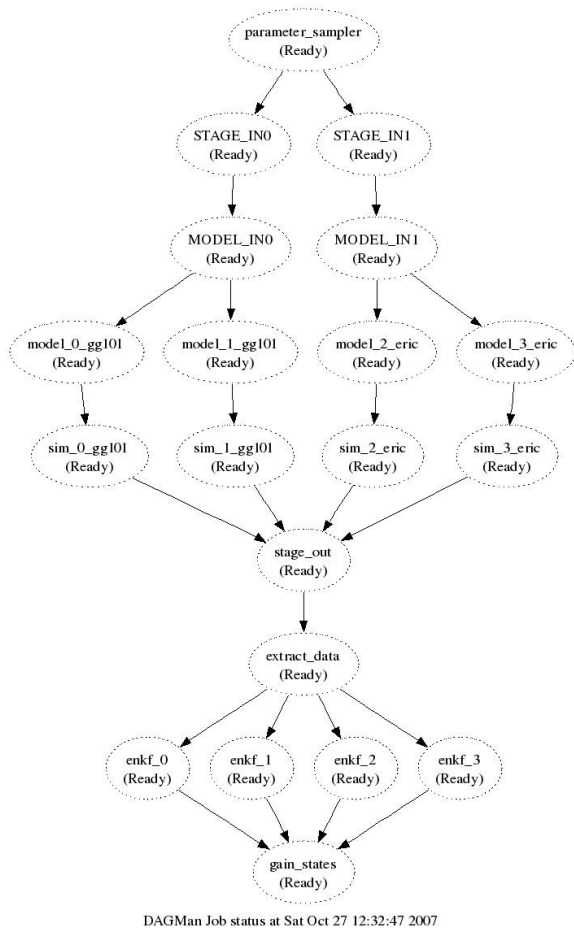


Figure 3: Concrete UCoMS Workflow

figure, there are a number of tasks that need to be executed in sequence. We used DAGMan as our workflow system to automate these tasks. This way, user does not have to intervene to run these tasks. In the case of failures, the workflow is capable of recovering. Figure 3 shows a simple concrete workflow for UCoMS project. In the concrete workflow, tasks are mapped to Grid resources and executed in the order specified. To be able to run the workflow for the UCoMS project, the user has to run the dag generator. The dag generator is written in Perl for flexibility and portability. We used task farming to distribute jobs on different resources. Based on the availability of machines, the dag generator will create all the Condor and Stork submit files as well as dag file to be submitted. Once the dag file is submitted, DAGMan will run the jobs in the given order, monitor them, and recover in the case of failures. It is important to note that the user does not have to intervene at all during the phase of execution.

We separated computing and data tasks. In other words, computing jobs are managed through Condor-G and data tasks are managed through Stork.

Some of the tasks might fail for some reason while they are executing. In these cases, the workflow should not fail as a whole. In our case, if any of the tasks fail for some reason, our workflow system will recover that task by either re-submitting the task or try a different protocol via Stork.

After the dag file is submitted to queue, tasks may need to be run in different resources. We used Condor-G to distribute tasks to different batch systems such Condor and PBS. Once the jobs are run successfully, the results will be transferred to local system to perform a post-processing.

6. RESULTS

6.1 Automation

In this work, we designed and developed a system that can process all the tasks for predicting reservoir performance in a fault-tolerant manner. Our system takes care of transferring the input files to the computation sites, running thousands of simulations across distributed resources, and transferring the generated results to the local system for post-analysis and handles all the failures that may occur during the process. This way, researchers can manage all the steps for predicting reservoir performance in an automated manner without any human interaction.

6.2 Reliability

There could be many problems that may occur during the execution such as hardware, software and network failures. Since both Condor and Stork have persistent job queues, crash or restart of the stork server and condor server is not a big issue. Stork and DAGMan have a retry mechanism and are both fault-tolerant. For data transfer jobs, if a data-transfer fails, stork tries to use an alternative protocol to carry out the transfer. If, for example, there are transient failures, Stork uses retries to guarantee that data transfers get completed. DAGMan also provides fault-tolerance feature. In DAGMan, the number of retries is specified in the dag file. After that, it creates a rescue DAG which indicates the jobs that have not been executed successfully and this can be forwarded to DAGMan so that it can retry the jobs again.

6.3 Improvement of Data Transfer Speed

In predicting reservoir performance, there are a number of input files involved for the simulations and output files generated after the simulations are run. Since we utilize resources distributed at multiple locations, transferring files to remote sites in an efficient manner is an important step. Because simulations are run across different clusters, input files need to be transferred to different remote sites and then data generated in remote sites have to be transferred back to local host in order to perform post-analysis.

Previous research showed that if there are many small files to be transferred as part of an application, transferring each file individually gives too much overhead. What we have come up with in this case to increase the transfer rate, we used the data fusion method to get a better performance for transferring files. The approach we used to transfer files is that we merge smaller files into larger packages to be transferred in our local system. Afterwards, we sent the merged package to remote host. Once the merged package is transferred successfully, we unpackage files in remote host. In doing so, we gained a significant performance rate.

Table 1 illustrates how many simulations need to be run depending on the number of factors, levels, and replicas. It also shows the size of input files and the time it takes to transfer input files to remote host using regular methods along with transfer speed of these files. The number of

Factors	Levels	Replicas	# of Sims	Data size (KB)	Transfer time regular (sec)	Transfer speed regular (KB/s)	Transfer time with data fusion (sec)	Transfer speed with data fusion (KB/s)	% Speedup
1	3	2	6	72	20	3.6	10	7.2	200 %
1	4	3	12	144	30	4.8	19	7.57	157 %
1	5	5	25	300	50	6	19	15.7	261 %
2	4	3	48	576	90	6.4	19	30.3	473 %
2	4	5	80	960	150	6.4	20	48	750 %
3	4	2	128	1536	230	6.7	20	76.8	1146 %
3	4	3	192	2304	295	7.81	20	115.2	1475 %
3	4	4	256	3072	445	6.9	21	146.2	2120 %
4	4	2	512	6144	895	6.86	25	245.76	3582 %
4	4	3	768	9216	1350	6.82	28	329.14	4826 %
4	4	4	1024	12288	1790	6.86	42	292.57	4264 %

Table 1: Comparison of data transfer speeds with and without data fusion

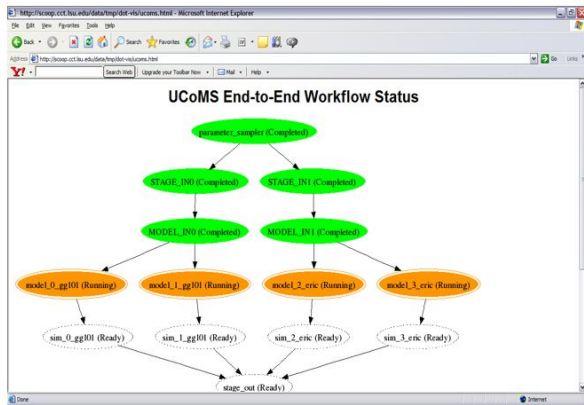


Figure 4: Example of how the workflow looks like after a task is completed

simulations is determined as $n^m * Replicas$ where n is the number of levels and m is the number of factors. For example, if there are two factors each of which has four levels with 2 replicas, then the simulations runs would be $4^2 * 3 = 48$. The table also shows the time it takes in seconds to transfer files to remote resources using the data fusion method and the speedup gained. As the number of datasets gets larger, there is a significant increase in transfer rate using the data fusion method.

6.4 Job Monitoring

It is also possible for a user to monitor the status of the jobs while the system is executed. This is very important feature of the system because the user can understand easily what goes on behind the scene. We implemented a web interface to visualize the system on a web browser. The user can start the workflow either from a portal or command line.

The user needs to be authorized to be able to use the portal. Once the user logs in with the username and password, he can start the workflow by pressing Start button. After that, the user can check the status of the jobs by using a web browser. Figure 4 shows how the components look like in a web browser.

As can be seen from this figure, when the status changes for a task, it will turn a different color depending on the status of the task. The job could be either Completed, Running

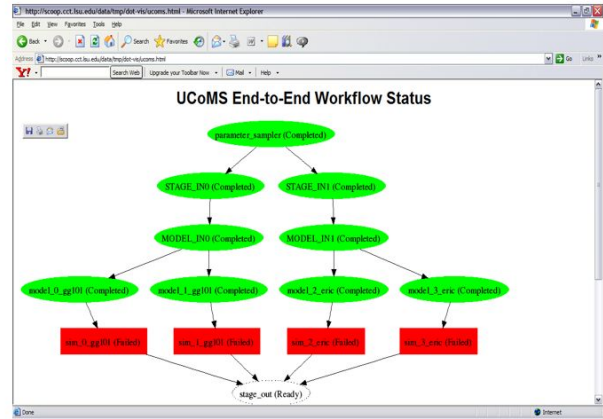


Figure 5: Example of how the workflow looks like in the case of failures

or Failed. Figure 5 shows an example of how the workflow looks like in the case of failures.

Another option to initiate the workflow for visualization could be from the command line. In this case, the user needs to run the dag file manually from the command line to be able to visualize the workflow on a web browser.

7. CONCLUSION

As the complexity of scientific applications increases, the need for using workflow management systems to automate end-to-end processing of applications on distributed resources increases as well. In today's scientific applications, computational experiments will involve large scale data sets and more computational resources and scientists distributed geographically. This report describes the requirements, design and implementation of a system for conducting reservoir performance prediction in an efficient and reliable way. The system has several main characteristics such as automation, reliability, being able to run jobs in different batch systems and visualization the status of the tasks on a web browser.

As our future goals, we would like to use Pegasus for mapping complex, large-scale scientific workflows with thousands of tasks processing TeraBytes of data onto the Grid. Another future goal is to use Petashare for solving data handling problems, where terabytes of data will be moved from sensors to processing nodes and then to archival sites.

8. ACKNOWLEDGMENTS

This project is in part sponsored by the National Science Foundation (NSF) under Award Number CNS-0619843 (PetaShare), by the U.S. Department of Energy (DOE) under Award Number DE-FG02-04ER46136 (UCoMS) and by the Board of Regents, State of Louisiana, under Contract Numbers DOE/LEQSF(2004-07) and LEQSF(2006-09)-RD-A-06. Joint support and access to Grid infrastructure has been provided by the Center for Computation & Technology at Louisiana State University.

9. REFERENCES

- [1] I. Foster, C. Kesselman, S. Tuecke. The anatomy of the Grid: enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200-222, 2001.
- [2] D. Hughes, P. Greenwood, G. Blair, G. Coulson, F. Pappenberger, P. Smith, and K. Beven. An intelligent and adaptable Grid-based flood monitoring and warning system. *UK eScience All Hands Meeting*, 2006.
- [3] D. Hughes, P. Greenwood, G. Coulson, and G. Blair. Gridstix: Supporting flood prediction using embedded hardware and next generation Grid middleware. In *WOWMOM 06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 621-626, Washington, DC, USA, 2006. IEEE Computer Society.
- [4] S. AlSairafi, F.S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopoulos, M. Osmond, A. Rowe, J. Syed, and P. Wendel. The design of discovery net: Towards open Grid services for knowledge discovery. *International Journal of High Performance Computing Applications*, 17(3):297-315, 2003.
- [5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM/IEEE Mobicom*, pages 56-67, Boston, MA. Aug, 2000.
- [6] Y. Yao and J. E. Gehrke.
- [7] T. Kosar, G. Kola, R. J. Brunner, M. Livny, and M. Remijan Reliable, Automatic Transfer and Processing of Large Scale Astronomy Datasets. In *Proceedings of 14th Astronomical Data Analysis Software & Systems Conference (ADASS 2004)*, Pasadena, CA.
- [8] G.Kola, T. Kosar and M. Livny. A Fully Automated Fault-tolerant System for Distributed Video Processing and Off-site Replication. In *Proceedings of 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (Nossdav 2004)*, Kinsale, Ireland, June 2004. The Cougar Approach to In-Network Query Processing in Sensor Networks. In *SIGMOD Record*, Vol 31 Number 3, Sept 2002.
- [9] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM TODS*, 2005.
- [10] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. Lee, J. Tao, Y. Zhao. Scientific workflow management and the Kepler system.
- [11] D. Hollinsworth *The Workflow Reference Model, Workflow Management Coalition, TC00-1003*, 1994.
- [12] D. P. Spooner, J. Cao, S. A. Jarvis, L. He and G. Nudd. Performance-Aware Workflow Management for Grid Computing
- [13] The UCoMS Project Home Page. November, 2007, <http://www.ucoms.org/>
- [14] G. Allen, K. Davis. The Grid Application Toolkit: Towards Generic and Easy Application Programming Interfaces for the Grid. *Proceedings of the IEEE*, Vol. 93, No. 3, pp. 534-550, March 2005.
- [15] K. Aziz, A. Settari. "Petroleum Reservoir Simulation". *Applied Science Publishers Ltd.*, London, 1979.
- [16] J. Yu, and R. 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, 2005.
- [17] Z. Lei, D. Huang, A. Kulshrestha, S. Pena, G. Allen, X. Li, R. Duff, S. Kalla, C. D. White, J. R. Smith. , Leveraging Grid Technologies For Reservoir Uncertainty Analysis. , in the *proceeding of High Performance Computing Symposium (HPC 2006)*, April 3- 6, 2006, Huntsville, AL, 2006.
- [18] I. Taylor, M. Shields, and I. Wang. Resource Management of Triana P2P Services. Grid Resource Management, Kluwer, Netherlands, June 2003.
- [19] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver and K. Glover, M.R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045-3054, Oxford University Press, London, UK, 2004.
- [20] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, M. Livny. Pegasus: Mapping Scientific Workflow onto the Grid. *Across Grids Conference 2004*, Nicosia, Cyprus, 2004.
- [21] R. D. Stevens, A. J. Robinson, and C. A. Goble. myGrid: Personalized Bioinformatics on the Information Grid. *Bioinformatics*, 19(Suppl. 1):i302-i304, Oxford University Press, London, UK, 2003.
- [22] I. Altintas, A. Birnbaum, K. Baldrige, W. Sudholt, M. Miller, C. Amoreira, Y. Potier, and B. Ludaescher. A Framework for the Design and Reuse of Grid Workflows. *International Workshop on Scientific Applications on Grid Computing (SAG'04)*, LNCS 3458, Springer, 2005.
- [23] X. Liu, J. Liu, J. Eker, and E. A. Lee. Heterogeneous Modeling and Design of Control Systems, *Software-Enabled Control: Information Technology for Dynamical Systems*, Tariq Samad and Gary Balas (eds.), Wiley- IEEE Press, April 2003.
- [24] M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *8th International Conference of Distributed Computing Systems (ICDCS)*, IEEE CS Press, Los Alamitos, CA, USA, June 1988; 104-111.
- [25] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. Condor - A Distributed Job Scheduler. *Beowulf Cluster Computing with Linux*, The MIT Press, MA, USA, 2002.

- [26] J. Frey, I. Foster, M. Livny, T. Tannenbaum, and S. Tuecke, S. Condor-G: A Computation Management Agent for Multi-Institutional Grids, University of Wisconsin Madison, 2001.
- [27] M. Papakhian. Comparing Job-Management Systems: The User's Perspective. *IEEE Computational Science and Engineering*, (April-June) 1998. See also <http://pbs.mrj.com>.
- [28] Loadleveler accessed September 2007. [Online]. Available: <http://www.mhpcc.edu/training/workshop/loadleveler/>.
- [29] T. Kosar and M. Livny Stork: Making Data Placement a First Class Citizen in the Grid. *To appear in Journal of Parallel and Distributed Computing*, (2005)
- [30] Graphviz - Graph Visualization Software. [Online]. Available: <http://www.graphviz.org/>.
- [31] T. Kosar. PetaShare: an innovative distributed data archival, analysis and visualisation instrument for data intensive collaborative research. <http://www.petashare.org>
- [32] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Lamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle : A Framework for Constructing Scalable Replica Location Services. *In Supercomputing (SC2002)*, Baltimore, USA: IEEE Computer Society, Washington, DC, USA, November 16-22, 2002.
- [33] E. Deelman, C. Kesselman, and G. Mehta. Transformation Catalog Design for GriPhyN. Technical Report GriPhN-2001-17, 2001.
- [34] Pegasus. [Online]. Available: <http://pegasus.isi.edu/index.php>.