

When Workflow Management Systems and Logging Systems Meet: Automatically Analyzing Large-Scale Execution Traces

Problem Statement Scientific research using today's cyberinfrastructure often requires very large-scale calculations. In order to achieve scientifically meaningful results, many research projects require thousands and even millions of coordinated tasks to be executed efficiently and reliably. Scientific workflow management systems, such as the Pegasus Workflow Management System (Pegasus-WMS, <http://pegasus.isi.edu>) have been proven to provide a reliable and efficient platform for the execution of complex scientific workflows on the the TeraGrid and the Open Science Grid. Among these workflows are applications in astronomy, bioinformatics, earthquake sciences, gravitational-wave physics, and others. Although a workflow management system can handle the tasks' execution, it is still challenging to analyze the results of the computations: which tasks were executed and where, how long did they take, etc., especially when a million interdependent tasks form a single analysis and the execution environment is widely distributed. At the same time logging technologies have been maturing and providing insights into system behavior. More detail is becoming available from the systems themselves, e.g. via *ganglia* or *nagios*. Middleware components are also beginning to log more detailed and precise information: For example, the XML records provided by Kickstart, or the adoption of the CEDPS "Best Practices" logging methodology by all the major components of the Globus Toolkit and auditing components within EGEE. Tools for collection and correlation are becoming better as well: distributed versions of *syslog*, such as *syslog-ng*, are becoming standard parts of most Unix operating systems. To normalize and correlate the resulting flood of information, the NetLogger Toolkit (<http://acs.lbl.gov/NetLoggerWiki>) uses a relational database back-end. In order to analyze large amounts of data, often consisting of a million records, one needs to leverage existing log processing capabilities. This poster explores the integration of workflow management with logging capabilities and shows the results using a real-world application from the earthquake science domain.

Approach In this work we integrated two mature systems, each focused on a different area of the problem: Pegasus-WMS for workflow management and NetLogger for logging and log mining. We applied the combined system to the analysis of the SCEC CyberShake application running on the TeraGrid (www.teragrid.org).

Pegasus-WMS is composed of the Pegasus workflow mapper, the DAGMan workflow executor, and the Condor schedd. In order to use the system, the scientist needs to describe the workflow at a high-level, capturing the computations involved and the data that need to be processed. This description is usually independent of the underlying cyberinfrastructure. The workflow along with information about the execution environment (including data and computational resources) is used by Pegasus-WMS to generate an executable workflow and to run it in an efficient and reliable fashion. Each job in the executable workflow is wrapped by kickstart, a shell script that captures information about the job: input/output data, parameters used, runtime, etc. If errors occur, kickstart captures the error codes. The output file of the kickstart wrapper are sent back to the "submit host" the resource which manages the overall workflow execution on remote resources.

NetLogger is a set of software tools and a methodology for troubleshooting and performance analysis of complex applications and middleware. The software tools include a multi-language instrumentation library and a log "pipeline". To use the log pipeline, the logs of interest first need to be collected by the middleware, as with Pegasus, or with a distributed log collection tool like *syslog-ng*. Then the log parser converts the logs from their native format to NetLogger's standard format. The database loader loads NetLogger-formatted log files into a pre-defined schema. Troubleshooting and performance analysis proceeds from these database tables, e.g., through direct SQL queries, Python programs, or with the "R" statistical language.

Example application: The CyberShake research project at the Southern California Earthquake Center (SCEC) uses a set of scientific numerical modeling codes to simulate earthquakes based on physics-based models of earthquake processes. These simulations can then be used for seismic hazard analysis and risk management. The ultimate goal of the SCEC research program is to transforming seismology into a predictive science with forecasting capabilities similar to those available today in the areas of climate modeling and weather forecasting.

To characterize the earthquake hazards in a region, seismologists and engineers utilize the Probabilistic Seismic Hazard Analysis (PSHA) technique. PSHA attempts to quantify the peak ground motions from all possible earthquakes that might affect a particular site (geographic area) and to establish the probabilities that the site will experience a given ground motion level over a particular time frame. The goal of the CyberShake project is to bring the physics-based modeling for PSHA calculation into the mainstream. CyberShake calculations consist of two main parts. First, an MPI-code is run to calculate volumetric datasets called strain Green tensors

(SGT's). Second, large-scale post-processing calculation is done in which hundreds of thousands of serial jobs process the SGT's. The serial jobs often have short run-times. SCEC uses Pegasus-WMS to run CyberShake workflows on the TeraGrid. A single CyberShake workflow can consist of as much as one million jobs.

Figure 1, on the right, shows the system architecture. Pegasus-WMS dispatches workflow execution tasks to the execution environment. The kickstart wrapper generates the execution logs which are then parsed and uploaded to the Log DB. Currently the logs are processed "offline" after the workflow completes execution. Subsequent queries to the database provide statistics and graphs of the overall workflow performance.

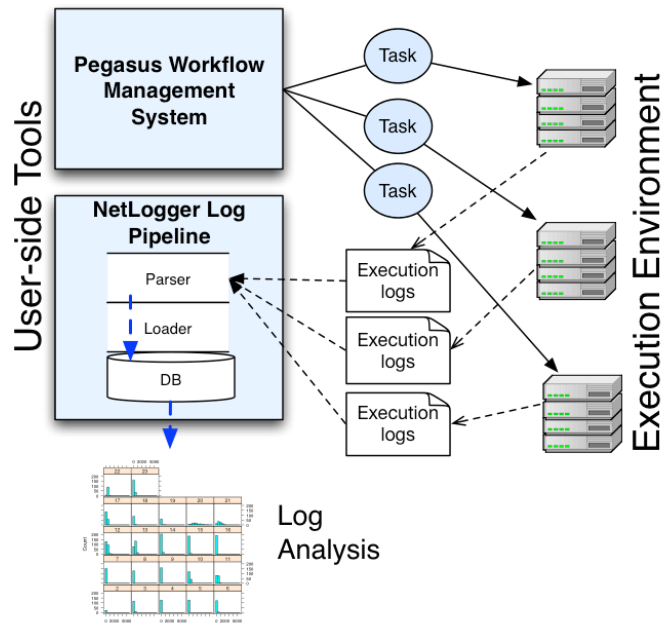


Figure 1: System Architecture

Results

System scalability Initial results show that the NetLogger pipeline to load the database scales well, and that the analysis capabilities are quite powerful and flexible. For very large datasets, queries may be too slow to be fully interactive, but more work on database optimization and setting up derived tables is expected to resolve this. Results shown here use log data from a CyberShake run on November 30, 2007. The run spanned four distinct clusters and contained roughly 800,000 kickstart invocation records. Kickstart logs were parsed into the NetLogger "Best Practices" (BP) format, and then these logs loaded into a MySQL database.

Analysis Initial analysis consisted of six queries. Together, they provided answers to some basic questions: what, if anything, failed; how many resources were consumed; and how well did the workflow perform. The queries and results are shown in Table 2, below. The run times are from a Macintosh Intel Core 2 Duo laptop.

Table 1: Initial Queries on the Kickstart Log Data

Query	Result size	Result value	Run time
(1) How many jobs failed	1 row	0	9.4 seconds
(2) How many jobs succeeded	1 row	754063	10.2 seconds
(3) How many jobs ran on a given day	1 row	754063	0.8 seconds
(4) What was the cumulative runtime of these jobs	1 row	5270.39 hours	25.8 seconds
(5) How many jobs ran on given hosts	361 rows	Count by host	35.0 seconds
(6) How many jobs of a given type ran on a given day	3 rows	Count by type	24.5 seconds

Further analysis focused on the load balance in space and time. This turned out to be a relatively simple extension of the query (5) above. A single SQL query aggregated Pegasus "invocations" by hour, and then an R script plotted a histogram, shown in Figure 2, for the number of jobs per host in each hour-long period.

A natural next step was to "drill down" and see how individual jobs contributed to the overall time for the clusters of jobs submitted by Pegasus-WMS. To do this, we parsed the output of a Pegasus tool called *tailstatd*, which provided a simplified version of the Condor DAGMan logs. Correlations with other logs could be added the same way to incrementally deepen and broaden the analysis capabilities.

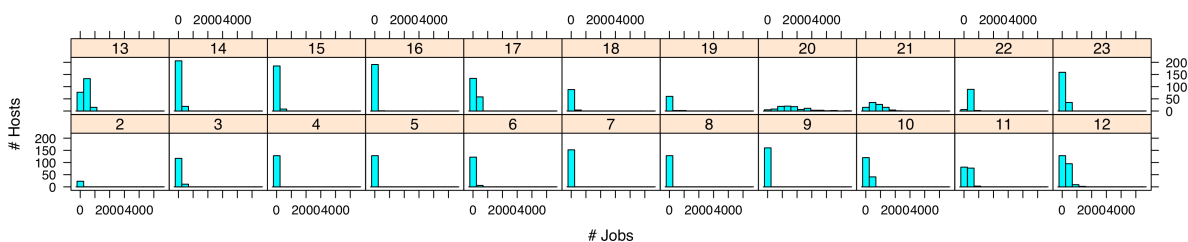


Figure 2: Histogram of jobs-per-host broken down by hour