SANDIA REPORT

SAND2008-4460
Unlimited Release
Printed July 2008

iSIGHT-FD Scalability Test Report

Max S. Shneider, Robert L. Clay

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2008-4460
Unlimited Release
Printed July 2008

iSIGHT-FD Scalability Test Report

Max S. Shneider
Informatics and Decision Sciences Department
Sandia National Laboratories, CA
P.O. Box 969
Livermore, CA 94551-0969
msshnei@sandia.gov

Robert L. Clay
Advanced Software Research and Development Department
Sandia National Laboratories, CA
P.O. Box 969
Livermore, CA 94551-0969
rlclay@sandia.gov

Abstract

The engineering analysis community at Sandia National Laboratories uses a number of internal and commercial software codes and tools, including mesh generators, preprocessors, mesh manipulators, simulation codes, post-processors, and visualization packages. We define an analysis workflow as the execution of an ordered, logical sequence of these tools. Various forms of analysis (and in particular, methodologies that use multiple function evaluations or samples) involve executing parameterized variations of these workflows. As part of the DART project, we are evaluating various commercial workflow management systems, including iSIGHT-FD from Engineous. This report documents the results of a scalability test that was driven by DAKOTA and conducted on a parallel computer (Thunderbird). The purpose of this experiment was to examine the suitability and performance of iSIGHT-FD for large-scale, parameterized analysis workflows. As the results indicate, we found iSIGHT-FD to be suitable for this type of application.

Acknowledgments

Special thanks to Shyam Kumar, Mark Ondracek, and Jon Long of Engineous for their help throughout the test. Their support throughout this experiment was invaluable. Also, thanks to Mike Eldred for his help with DAKOTA and Matt Kerschen for providing us with our example problem.

Contents

Figures

Nomenclature

APC Analysis Process Coordinator

DAKOTA Design Analysis Kit for Optimization and Terascale Applications

DART The Design through Analysis Realization Team

DTA Design Through Analysis

UQ Uncertainty Quantification

This page intentionally left blank.

DART Overview

The engineering and analysis community at Sandia National Laboratories uses a number of internal and commercial software codes and tools to perform their work. These include mesh generators, pre-processors, mesh manipulators, simulation codes, post-processors, and visualization tools.

DART is integrating these codes and tools into a complete DTA toolset. This will reduce the time required to create, generate, analyze, and manage the data that is generated in the computational analysis process.

DART/UQ Overview

While DTA tools may be executed only once, they are increasingly being used to generate sample-based studies and answer optimization and uncertainty quantification (UQ) questions. In these cases, the tools are executed multiple times as parameterized sets of simulation runs. Often, this is done in a parallel environment to reduce the turnaround time. At Sandia, these types of studies typically involve DAKOTA [1], a parallel framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Because it is so widely adopted at Sandia, our DART/UQ solution must be integrated with DAKOTA.

There are two main objectives to the DART/UQ project. First, we want to simplify the setup of these large, parameterized simulations in the DART environment. In talking with the analyst community, we found that they typically spend more time setting up the problem than they do actually running it and analyzing the results. Our goal is to reverse that time allocation. The second objective is to provide data management so that inputs and results are automatically and systematically committed to repositories as the simulation runs execute. There is currently no standard mechanism for easily sharing and backing up information in these types of studies. In addressing this second objective, we will provide data management in terms of sets of runs, consistent with the natural structure associated with parametric and UQ studies.

We have identified a number of solutions to meet these objectives, including a new DAKOTA user interface to easily create, modify, and submit jobs, and integration with DART data management capabilities (embodied by the APC). However, the piece that's relevant to this particular document is using commercial workflow management systems to build and manage workflows, which execute ordered, logical sequences of Sandia's engineering analysis tools.

iSIGHT-FD Overview

Currently, when Sandia analysts want to use a DART tool in conjunction with DAKOTA, they write a script to setup work directories, parse input parameters from DAKOTA parameter files (using a tool such as grep [2]), substitute the input parameters into the tool's input deck (using another tool like APREPRO [3]), execute the tool, parse the output parameters from the tool's output file, substitute the output parameters into DAKOTA result files, and cleanup the work directories. Writing these scripts can take days to months depending on the level of complexity; and, there is no inherent mechanism for sharing them, short of passing on the scripts to others. Also, extending and debugging the scripts can be very time consuming, again depending on the level of complexity. To alleviate the burden on the user, we are exploring commercial workflow systems which allow users to layout workflows in a drag-and-drop, reusable fashion, wrap tools so that they can be used in the workflow, and conveniently manage input and output parameter mappings so that they can easily be passed between tools.

We evaluated a number of commercial workflow systems, including Samtech's BOSS quattro [4], Phoenix Integration's ModelCenter [5], MSC's SimManager [6], and Engineous' iSIGHT-FD [7]. Through a formal down-selection process, we decided to explore iSIGHT-FD in more detail and test it in the scalability test described below. iSIGHT-FD has a Gateway client that enables users to create workflows as described above. The workflows may be run from the Gateway, or through a command line utility called fipercmd. Both methods use the same underlying Java libraries, so the choice is up to the user. The fipercmd utility was very useful in our case because it allowed the installation and execution of iSIGHT-FD on a parallel cluster (Thunderbird, described below), where graphical applications were not an option. A separate installation of iSIGHT-FD was used on a standard workstation to create the workflows which were then transferred over when the scalability test was ready to run. This ability to create a workflow on one system (e.g., the user's workstation) and then run it on another machine (e.g., a parallel cluster) was one of the reasons iSIGHT-FD was selected as a leading candidate for further evaluation. All work described in this report was conducted using iSIGHT-FD version 3.0.0.080911452.

Purpose of the Scalability Test

Every software package has a certain amount of associated overhead. For instance, with iSIGHT-FD, the time that it takes to wrap tools, layout the workflow, and execute the workflow all must be considered. In addition, iSIGHT-FD uses Macrovision's FLEXlm license server [8] which requires that each time you run an instance of iSIGHT-FD, whether it's through the Gateway or fipercmd, the user must obtain a new license from the license server. This performance penalty presented a big question since we wanted to make sure that there weren't any critical performance hits in the parallel domain, where we'd be requesting multiple (and perhaps a large number of) licenses at the same time.

Additional questions arose since we'd be using iSIGHT-FD in a much different way than it was intended to be used. Most users design complex workflows that are executed once either directly in the Gateway, or through a web interface (which is provided as a convenient way to share workflows with multiple users). If the workflows are executed multiple times, it's usually a small number that can be done directly on the user's workstation serially (one run starts after the previous one completes). At Sandia, however, much of our analysis is done on parallel clusters. For instance, at the time of the test, we were using two of the top 11 supercomputers in the world [9]). This is both because of the size of the models that we use in our simulations, and the sheer number of runs that are often required. Running on a parallel cluster means that multiple (and possibly many) fipercmd instances would be utilizing the same Java libraries concurrently – albeit on different compute nodes. In addition, the workflows would be executed with fipercmd which is provided in the iSIGHT-FD distribution, but has not seen heavy usage in the user community.

Scalability Test Specifics

Because of the questions above, we designed a scalability test to examine the system behavior and performance. For the experiment, we obtained an electrical analysis study that uses the Xyce electronic simulator [10]. Since the study was used on a previous Sandia project (instead of being concocted simply to test the system), it naturally provides a realistic example of the work at Sandia National Labs.

As shown in the scalability test workflow (Figure 1) below, DAKOTA drives the simulation and is responsible for managing the individual runs. In other words, only one job is submitted to the PBS queue, upon which and DAKOTA starts a batch of runs on the reserved compute nodes, waits for these runs to complete (indicated by the existence of result files), and then repeats this process until the job is complete. This is standard practice when using DAKOTA on parallel machines, and avoids the delay incurred by having every iteration (i.e., individual simulation run) work through the queuing system on the parallel machine.

The workflow itself is simple: we extract the input parameters from the DAKOTA parameter file, substitute the parameters into the Xyce input deck, run Xyce, extract the output parameters from the Xyce output file, and substitute the parameters into the DAKOTA result file.

The test was run on Sandia's Thunderbird cluster, which at the time of the test consisted of 8 login nodes and 4,480 compute nodes. Each node had a dual 3.6 GHz Intel EM64T processor, 6 GB of RAM, and Red Hat Enterprise Linux WS release 4 for the OS. Thunderbird has a shared file system which enabled the test to be self-contained; iSIGHT-FD, DAKOTA, and Xyce were all installed locally. DAKOTA ran on the head node and spawned Xyce workflow runs to the compute nodes. The iSIGHT-FD license server was also installed on the shared file system, and fipercmd licenses were obtained by putting "localhost" in the configuration files.

Originally we intended to execute a 200,000 run job, using 1,000 processors at a time. However, we soon learned that a job of that size would take days to get through the Thunderbird job queue. Also, the job would take another day or two to complete, putting the total turnaround time for a single job at around a week, at considerable cost of machine time. Because of this, we decided to shorten the test to a 5,000 run job, using 50 processors at a time.

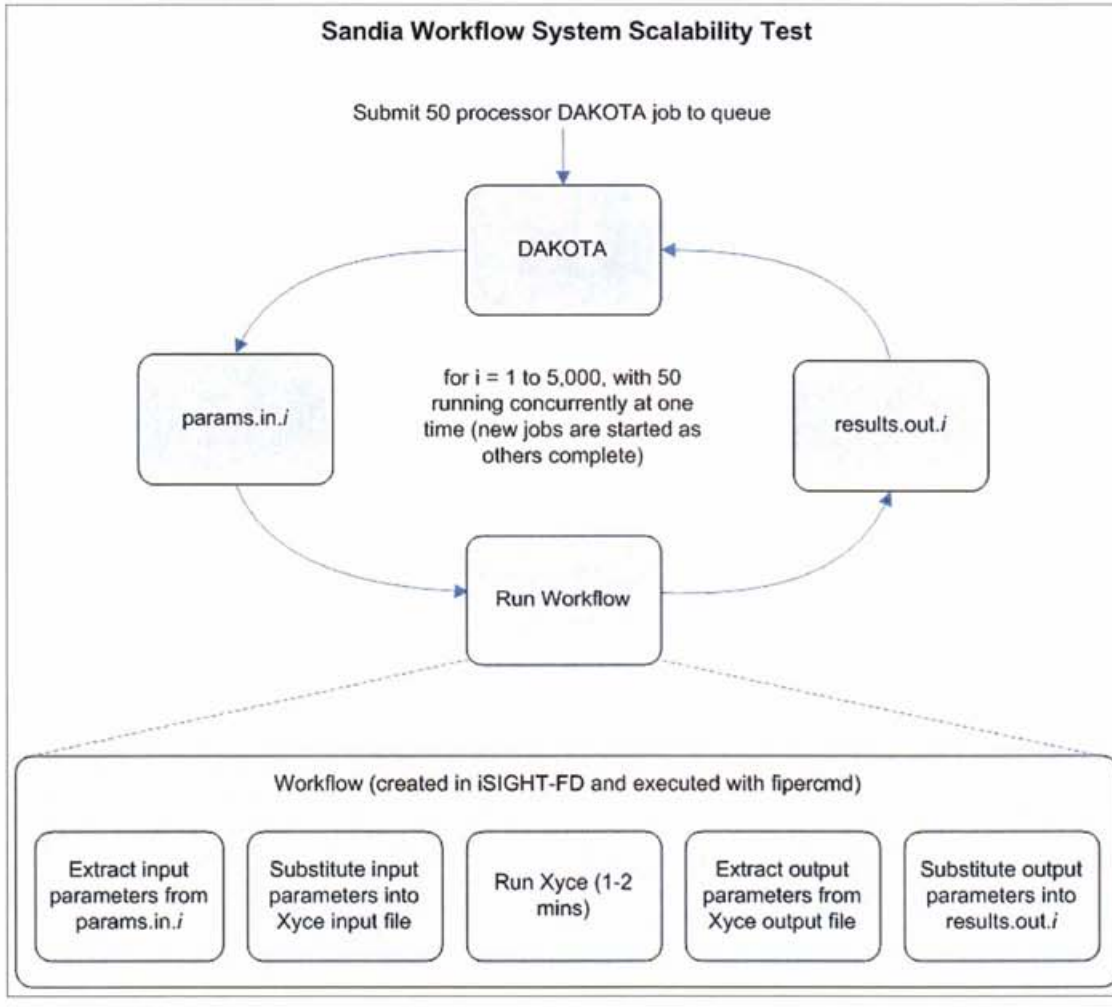

Figure 1. Scalability Test Workflow

Scalability Test Observations

We began by participating in a day and a half of iSIGHT-FD training with an Engineous representative. During the training, we worked through examples in order to become familiar with the software which helped tremendously in our initial development. We were able to wrap a couple of our tools within a day which was far quicker than expected.

It took a couple of additional days to complete our workflow since it entailed some extensive parsing and parameter substitution. We also had to develop some solutions for running each fipercmd instance in a separate working directory. These took some time to resolve with the Engineous developers, but iSIGHT-FD always proved flexible enough to allow us, in some way or another, to do what was needed. For instance, we parsed the current working directory with a Unix command, and then set the run number as a global variable so that it could be accessed by subsequent components.

However, there were some errors that even after weeks of investigation were never resolved. The most significant of these was an error that occurred in the DAKOTA parameter file Data Exchanger, in which we were parsing over 100 input parameters. One factor that made this hard to debug was that the file was dynamic. It wasn't available until DAKOTA created it; and, it was different for each run. We worked with the Engineous team to significantly reduce the frequency of these errors to under one percent of the runs, but were never able to eliminate them completely.

We did see some other errors in the results (for instance, the fipercmd utility sometimes failed to load the required iSIGHT-FD libraries), but these were limited, so we did not feel the need to investigate them further. Overall, it took a couple of solid weeks to develop the test. This can be attributed to a couple of factors. First, it simply took some time to become familiar with how to work in iSIGHT-FD, which is true of any software package. For instance, we assigned the Data Exchanger output files and the component run directories to the wrong places, both of which the Engineous team had to correct. We view this as a one-time cost that comes with learning any new software package, so further development would presumably be much quicker. All in all, the product was relatively straightforward to install, learn, and use.

The second main factor affecting development time was the fact that we were using iSIGHT-FD in non-standard ways. For example, the fact that we had 50 fipercmd instances executing at once could have caused the parsing errors that the Engineous team had not seen before. We were also calling fipercmd from a DAKOTA job on a massively parallel cluster. The unique environment could have played a part in the parsing errors as well. Also, being new DAKOTA users ourselves, it turns out that we were originally invoking it with the wrong syntax. That caused some other errors and tracking it down added a couple of days to the process.

Scalability Test Results

We ran three separate yet identical scalability test instances, with the following success/failure rates:

- **Job 1** – 4,991 successful runs and 9 failures
- **Job 2** – 4,991 successful runs and 9 failures
- **Job 3** – 4,992 successful runs and 8 failures

On each run, we gathered the following timing statistics:

- **fipercmd-start** – the time (in seconds) when the fipercmd starts, recorded in the script immediately before we call fipercmd.
- **workflow-start** – the time (in seconds) when the workflow starts, recorded in the first component in the workflow.
- **workflow-end** – the time (in seconds) when the workflow completes, recorded in the last component in the workflow.
- **fipercmd-end** – the time (in seconds) when the fipercmd completes, recorded in the script immediately after we call fipercmd.

The statistics that were used to calculate the time intervals are as follows:

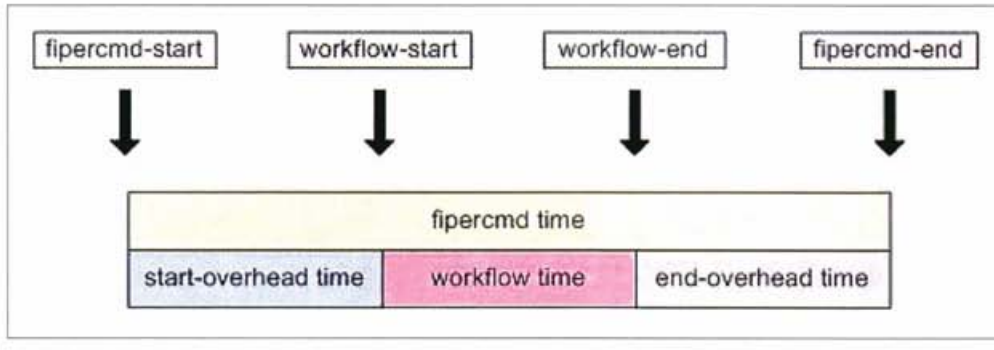

Figure 2. Time Interval Calculations

- **fipercmd time** – the time between **fipercmd-start** and **fipercmd-end**, which includes launching fipercmd, obtaining a license, running the workflow, returning the license, and exiting fipercmd. Results reflect only successful runs, not failures.

- **workflow time** – the time between **workflow-start** and **workflow-end**, which does not include obtaining/returning a license, or launching/exiting fipercmd. Results reflect only successful runs, not failures.
- **start-overhead time** – the time between **fipercmd-start** and **workflow-start**, which includes launching fipercmd and obtaining a license. This value is reported even for failed runs, since all failures occurred during the workflow execution.
- **end-overhead time** – the time between **workflow-end** and **fipercmd-end**, which includes returning a license and exiting fipercmd. Since this value was always zero or one second, we have removed it from the results in this report.

Figures 3-5 show the **fipercmd**, **workflow**, and **start-overhead** times for each job:

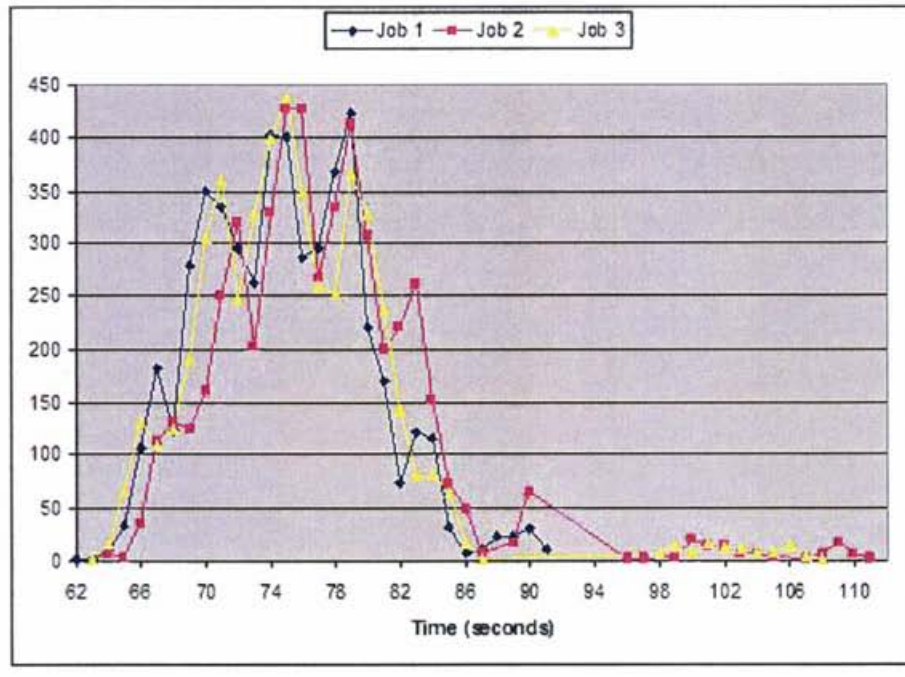

Figure 3. Fipercmd Result Times

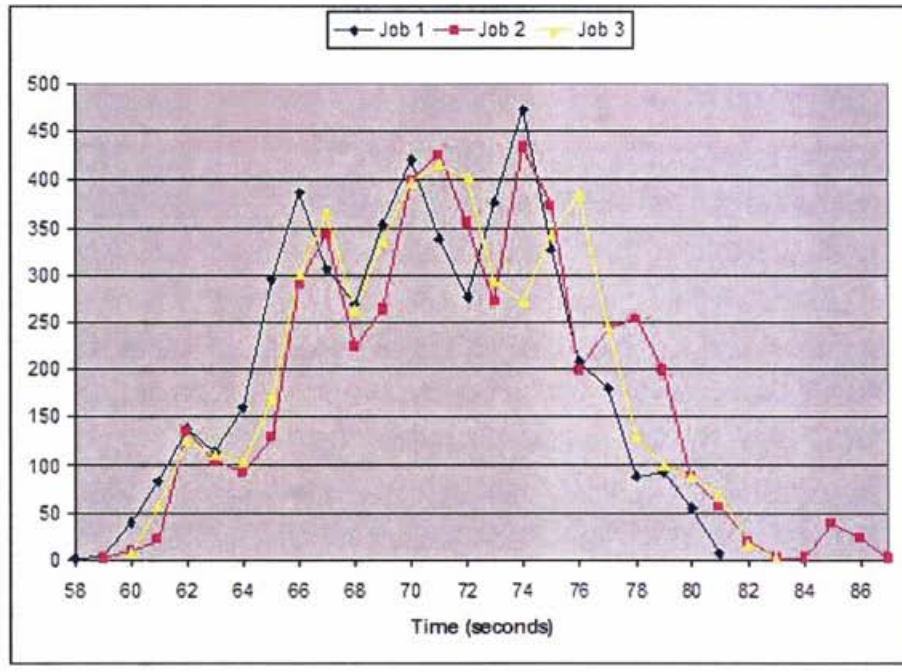

Figure 4. Workflow Result Times

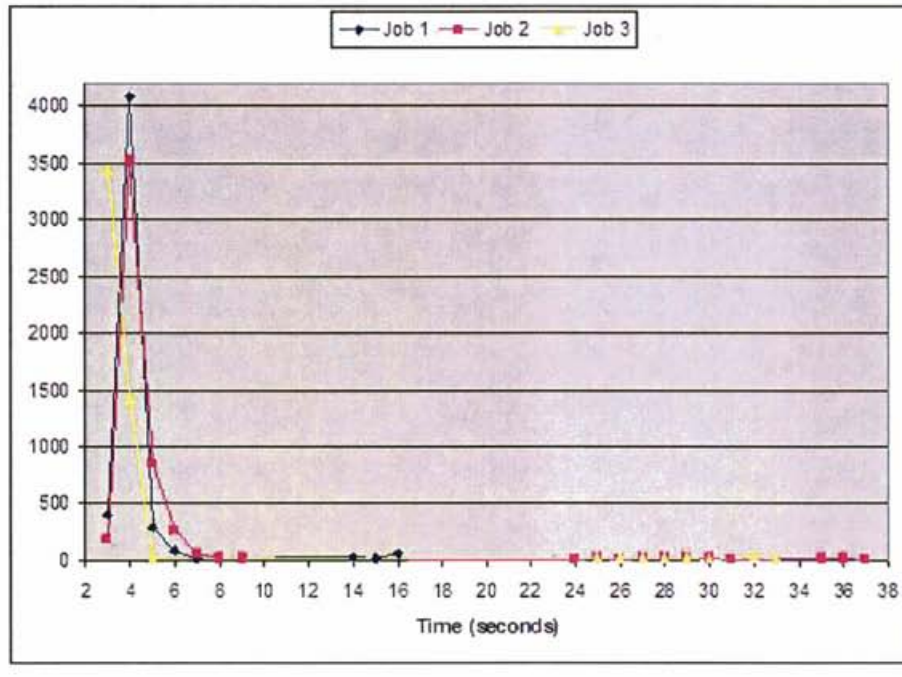

Figure 5. Start-Overhead Result Times

Scalability Test Discussion

As shown in the results, all three sets of **workflow** times have a mean of about 70 seconds, with a Gaussian-like distribution and a range from around 60 seconds on the low end to 85 seconds on the high end.

The **start-overhead** times are predominantly around 3-4 seconds with a few outliers in the 30 second range. This is in stark contrast to the previous version of iSIGHT-FD that we tested which had numerous outliers around or even exceeding 5,000 seconds (83 minutes) on the high end. The few outliers on the results above are not viewed as significant, but we remain curious as to the exact cause. The typical 3-4 second overhead from the fipercmd startup should be measured in terms of the typical time to run the workflow. In general, the simulations embedded in the workflow are expected to take significantly longer than a few seconds, so this startup time isn't viewed as problematic for the typical case. If however, the workflow times were small in comparison, then the startup overhead would begin to dominate the overall time, which is not a desirable outcome.

Another consideration is the overall success rate of the runs submitted. As shown above, the 3 jobs of 5,000 runs each had failure rates of 9, 9, and 8, or 0.18, 0.18, and 0.16 percent, respectively. All three jobs were submitted to the queue with the exact same command, and they executed the same workflows and scripts. The different results are due to the aforementioned parsing bug, which causes runs to fail approximately 0.17 percent of the time. We are continuing to examine this issue in conjunction with the Engineous staff. It is somewhat disconcerting, but not a showstopper since 'random' failure modes are often seen on large jobs on our big clusters. Nevertheless, we would be more confident in recommending this tool if we understood the source of these failures and eliminated iSIGHT-FD as the source.

Conclusions and Recommendations

Overall, we were very impressed with iSIGHT-FD and with the support we received from the Engineous staff. We were able to wrap our tools and create our workflow in a short amount of time. We were also able to execute our workflow in a non-traditional way, as compared to other users, with concurrent fipercmd instances on a parallel machine.

While there are still a few open questions about using this tool, we believe it could be used today at Sandia National Labs for parallel execution (i.e., execution of multiple workflow instances on a parallel machine). A more exhaustive set of experiments would be advisable before a large-scale rollout for production use. Yet, we have no issues recommending early adopters to try iSIGHT-FD in lieu of their traditional scripting approaches.

References

[1] DAKOTA – http://www.cs.sandia.gov/DAKOTA/software.html

[2] grep – http://www.gnu.org/software/grep/

[3] APREPRO – http://endo.sandia.gov/SEACAS/Documentation/Aprepro.SAND.pdf

[4] Samtech's BOSS quattro – http://www.samcef.com/en/pss.php?ID=3&W=products

[5] Phoenix Integration's ModelCenter – http://www.phoenix-int.com/products/modelcenter.php

[6] MSC's SimManager – http://www.mscsoftware.com/products/simmanager.cfm?Q=131&Z=288

[7] Engineous' iSIGHT-FD – http://www.engineous.com/iSIGHTFD.cfm

[8] Macrovision's FLEXlm license server – http://en.wikipedia.org/wiki/FLEXlm

[9] Top Supercomputers as of June, 2007 – http://www.top500.org/list/2007/06/100

[10] Xyce – http://xyce.sandia.gov/

DISTRIBUTION:

1 Jon Long, Engineous Software, Inc., 825 Bantam Way, Petaluma, CA 94952
1 Kevin Harris, Engineous Software, Inc., 2000 CentreGreen Way Suite 100, Cary, NC 27513
3 Mark Ondracek, Engineous Software, Inc., 77 Milford Dr. Ste. 237, Hudson, OH 44236
1 Shyam Kumar, Engineous Software, Inc., 1175 Nelrose Ave., Venice, CA 90291

1	MS 0341	Steven Wix, 1734
1	MS 0376	Teddy Blacker, 1421
1	MS 0447	Matthew Kerschen, 2111
1	MS 1004	John Linebarger, 6344
1	MS 1004	Michael Skroch, 6344
1	MS 1004	Stephenson Tucker, 6344
1	MS 1318	Brian Adams, 1411
1	MS 1318	James Stewart, 1411
1	MS 1318	Michael Eldred, 1411
1	MS 9151	Howard Hirano, 8960
1	MS 9152	Edward Hoffman, 8964
1	MS 9152	Ernest Friedman-Hill, 8962
1	MS 9152	Marcus Gibson, 8964
1	MS 9152	Joe Shelton, 8964
3	MS 9152	Max Shneider, 8962
1	MS 9152	Mike Hardwick, 8964
3	MS 9152	Robert Clay, 8964
1	MS 9159	Heidi Ammerlahn, 8962
1	MS 9159	Joseph Ruthruff, 8964
1	MS 9159	Pamela Williams, 8962
1	MS 9159	Patricia Hough, 8962
1	MS 0899	Technical Library, 4536
2	MS 9018	Central Technical Files, 8944