Grid-enabling a vibroacoustic analysis toolkit

Brian Bentow, Jonathan Dodge, Aaron Homer, Christopher D. Moore and Robert M. Keller

Harvey Mudd College,
Claremont, California, USA
E-mail: bbentow@cs.hmc.edu
E-mail: jdodge@cs.hmc.edu E-mail: ahomer@cs.hmc.edu
E-mail: cdmoore@cs.hmc.edu E-mail: keller@cs.hmc.edu

Craig Lee*, Mark Thomas, Matthew Presley, Jorge Seidel, Robert Davis and Joseph Betser

The Aerospace Corporation,
El Segundo, California, USA
E-mail: lee@aero.org E-mail: mathomas@aero.org
E-mail: presley@aero.org E-mail: seidel@aero.org
E-mail: robdavis@aero.org E-mail: betser@aero.org
*Corresponding author

Abstract: A vibroacoustic analysis toolkit for launch vehicle telemetry was refactored as a service architecture using Globus Toolkit 4.0. We grid-enabled two tools in the analysis flow, managing their remote execution workflow from a desktop client. We examine the performance of the tools and the processing time for the entire tool chain. The results indicate that coordinated communication and execution scheduling as part of workflow management can significantly improve overall efficiency. Our experience also shows that the service-oriented architecture approach, using grid computing tools, can provide a more flexible system design, in addition to improved performance and increased utilisation of resources.

Keywords: vibroacoustic telemetry analysis; grid service performance.

Reference to this paper should be made as follows: Bentow, B., Dodge, J., Homer, A., Moore, C.D., Keller, R.M., Lee, C., Thomas, M., Presley, M., Seidel, J., Davis, R. and Betser, J. (2008) 'Grid-enabling a vibroacoustic analysis toolkit', *Int. J. High Performance Computing and Networking*, Vol. 5, No. 3, pp.168–178.

Biographical notes: Brian Bentow received the BS in Computer Science from Harvey Mudd College in 2005. He is currently Manager of Product Development at InstaMed in Irvine, California, which develops software for the healthcare industry.

Jonathan Dodge received the BS in Computer Science from Harvey Mudd College in 2005.

Aaron Homer received the BS in Computer Science from Harvey Mudd College in 2005. He is a software developer at Laserfiche in Long Beach, California.

Christopher D. Moore received the BS in Physics from Harvey Mudd College in 2005. He has since worked at Northrop Grumman's Navigation Systems Division in Woodland Hills, CA developing embedded systems for sensor fusion applications.

Robert M. Keller received the BS and MS Degrees from Washington University, and the PhD from the University of California, Berkeley. He has been on the faculty at Princeton University, the University of Utah, and the University of California, Davis prior to joining Harvey Mudd College, where he is a Professor and Director of the Computer Science Clinic. His research includes contributions to parallel computing, languages, and music software, among others.

Craig Lee is a Senior Scientist at The Aerospace Corporation and has worked in parallel and distributed computing for the last 25 years. He has conducted DARPA and NSF sponsored research in the areas of grid computing, optimistic models of computation, active networks, and distributed simulations, in collaboration with USC, UCLA, Caltech, ANL, and the College of William and Mary. He is on the steering committees for Grid XY and CCGrid and has served as a panelist for the NSF, DOE, NASA, and INRIA. He has published over 50 technical works, including four book chapters and seven edited volumes and issues. He is currently serving as

President of the Open Grid Forum. He has a PhD in Computer Science from the University of California, Irvine.

Mark Thomas is a member of the Technical Staff at the Aerospace Corporation, and is currently pursuing a combined MS in Computer Science and MBA at the University of California, Los Angeles. Mark is an expert on Grid Computing using the Globus tools and established the hardware and software infrastructures for the Enterprise Source Software environment at the Aerospace Corporation. His other interests include high performance computing using heterogeneous processor architectures. He holds a BS in computer science from the University of California at Berkeley.

Matthew Presley is a Senior Project Leader at the Aerospace Corporation researching distributed computing, including service-oriented architectures and transparent distributed execution of applications. He received a BS Mathematics from Harvey Mudd College and a PhD in Computer Science from UCLA where he worked on the verification of parallel discrete event simulation engines. He has worked at Jet Propulsion Laboratories and Computer Sciences Corporation developing simulations and simulation technology. As Chief Scientist of Agari Mediaware, he led a team creating distributed middleware for integrating rich media applications.

Jorge Seidel is an Engineering Manager in the Computer Systems Research Department at the Aerospace Corporation. His research interests include high performance computing, including the use of FPGAs nodes in computing clusters; computational wireless networks; and open source development. In addition to presenting a tutorial on open source at the *Ground Systems Architecture Workshop* he has taught the introduction to logic design course at USC. He holds a BS in Electrical Engineering from the University of Utah and an MS in Computer Engineering from the University of Southern California.

Robert Davis is a Project Leader in the Advanced Information Systems Technology Department at The Aerospace Corporation. He spent the 15 years of his career performing research in information systems and distributed computing and developing software applications for telemetry analysis, project management, and an Enterprise Service Bus. He holds a BS in Mathematics from Harvey Mudd College and a MS in Computer Science from UC Davis.

Joseph Betser is a Senior Project Leader for Strategic Planning, Knowledge Management, and Business Development with the Aerospace Corporation. He served as a DARPA PI for networking and information assurance and authored multiple publications in these areas. He received multiple commendations including the GPS Program Recognition Award, and awards for serving as a Program Chair and General Co-Chair of the *Ground System Architectures Workshop (GSAW)*. He served as Program Chair for multiple IFIP/IEEE Symposia , and received international citations from the IEEE and IFIP for leading global activities. He holds a BS in Aerospace Engineering, MS and PhD Degrees in Computer Science, and an executive MBA from UCLA.

1 Introduction

Space launches are costly, high-risk, multi-discipline endeavors, and mission payloadsmust be carefully engineered to ensure success. Simply surviving the launch is an important hurdle since satellite payloads are subjected to intense mechanical vibration and acoustic noise. In order to monitor, understand, and better predict this environment, The Aerospace Corporation is developing a Java-based tool suite called Vibroacoustic Intelligent System for Predicting Environments, Risk, and Specifications (VISPERS) that has tools to allow analysts to clean-up, filter and analyse vibroacoustic data (Bradford et al., 2002). Certain components of VISPERS are compute-intensive, however, and performance could be improved by parallelising those applications and running them on whatever resources are available. It would also be very useful to allow multiple telemetry analysts to use the tools in an interactive fashion and process data from different launches simultaneously. Hence, it was decided to investigate the implementation of VISPERS as a gridenabled *service-oriented architecture* to achieve these goals.

This paper reports on the continuing project to gridenable VISPERS and achieve enhanced performance and functionality by using the Globus Toolkit (http://www-unix.globus.org/toolkit/docs/3.2) to produce *gVISPERS*. We focus on two major service components, gTACT and gVAIL, which will be presented in detail shortly. Earlier work on gVAIL was reported in Bentow et al. (2005). That work was based on Globus 3.2.1 since Globus 4.0 was not available when the project was started. Now, however, all gVISPERS code, including gVAIL, is based on Globus 4.0.

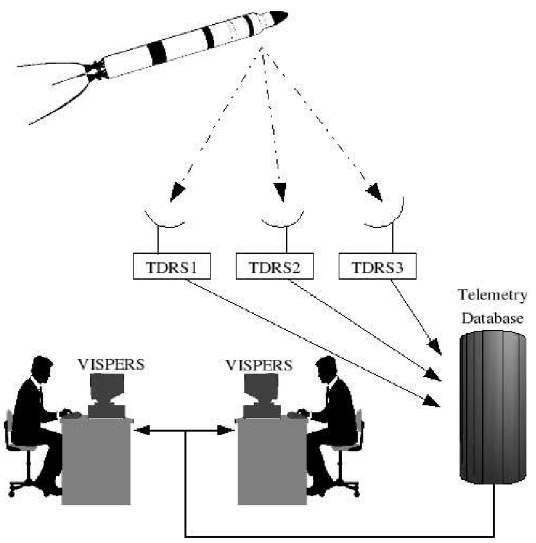
We begin by discussing telemetry collection and the general VISPERS toolkit. We then present the gTACT and gVAIL services. This gives us the opportunity to make some observations and record some lessons learned about

Globus 3.2.1 vs. 4.0. We then present performance data for gVAIL, gTACT, and the combined toolkit prior to making some concluding remarks.

2 Telemetry collection and VISPERS

When a spacecraft is launched it contains many sensors, including accelerometers to measure vibration and microphones to measure acoustic noise. The data provided by these sensors is converted from analog to digital, then consolidated into a single stream of digital data and sent by radio to Telemetry Data Receiving Stations (TDRS) along the path of the spacecraft, as illustrated in Figure 1. This telemetry stream will be picked up by the nearest receiving station, but will likely include static and other anomalies. For example, the signal may briefly disappear (drop out) due to the antenna on the spacecraft rotating away from the receiving station.

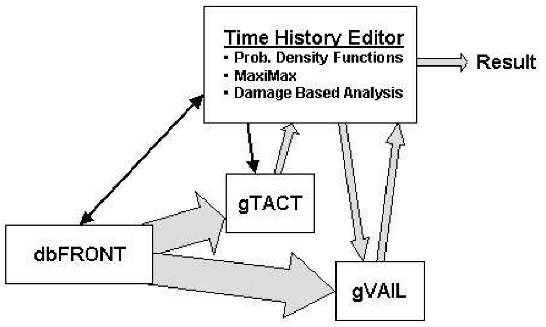
Figure 1 Telemetry collection



The data that VISPERS is used to analyse is provided by all of the telemetry stations along the flight path, both fixed stations and temporary stations on aircraft or ships. The multiple telemetry streams are placed in files in a telemetry database. Once there, they need to be scrubbed for noise, and consolidated into a single, best data stream prior to final processing. It is possible that hundreds of telemetry streams may have to consolidated and analysed to identify episodes of excessive vibration during launch.

This sequence of steps is illustrated in Figure 2. (Note that simple line arrows are used to represent interactions that involve primarily small amounts of control information, while block arrows are used to indicate interactions with significant amounts of data.) For historical reasons, the user client is called the *Time History Editor* which has a graphical user interface allowing the user to view and modify telemetry data files, known as *time histories*, to look for anomalies. From this desktop client, an analyst can select telemetry streams from the telemetry database front-end. Typically many telemetry streams are sent to Telemetry Alignment and Consolidation Tool (gTACT) and combined into a single, continuous waveform. A relatively small amount of alignment information is returned to the client. The client then uses this information to run VISPERS Artificial Intelligence Laboratory (gVAIL) and analyse possible episodes of interest. We note that once the best alignments are determined, less data must be retrieved from the telemetry database and realigning it is trivial. More importantly, however, we note that *gTACT and gVAIL can be run in either order.* An analyst may want to run gVAIL first to roughly identify areas of interest prior to getting an exact alignment with gTACT. Hence, for gVISPERS to be most useful in the general case, it must be possible to run gTACT and gVAIL interactively.

Figure 2 The gVISPERS workflow



The ultimate goal is, of course, to help engineers design parts that are unlikely to break in this severe environment. In addition to gTACT and gVAIL, tools are available to look for maximum vibrational episodes, analyse damage to the telemetry stream edit streams by hand, and construct Probability Density Functions (PDFs) for later comparison. Since these additional functions are not as compute-intensive, they still reside within the Time History Editor. We now describe gTACT and gVAIL in more detail.

3 gVISPERS components

3.1 gTACT

As noted above, telemetry is received at many different ground stations at different times depending on a launch vehicle's trajectory. The harsh launch environment may also cause severe noise and data drop-outs in the telemetry signal. The result is a set of fragmented telemetry streams that somehow need to be consolidated into a single waveform for analysis. Hence, the purpose of TACT is to align a series of waveforms with some amount of chronological overlap and consolidate them into a single, contiguous waveform.

The TACT algorithms provide excellent opportunities for parallelisation. gTACT, the grid-enabled version of TACT, was implemented to take advantage of some of these opportunities. There are two main algorithms TACT uses to align waveforms called Diff and Slide. Only the more compute-intensive algorithm, Slide, was implemented for gTACT and is discussed here.

Given a region of overlap between a reference waveform *ref* and the chronologically next waveform *next*, the Slide algorithm takes a small region of points from ref *refRegion* and compares it to a region of points of the same length in *next.* Starting from some index x within the chronological overlap, Slide will calculate the absolute difference between *refRegion* and the region of length *refRegion.size* starting at index x in *next.* This process will be repeated for some range of indices $[x, y]$ in the overlap region. Slide then returns the index at which the absolute difference was minimal. One can think of this process as taking *refRegion* and sliding it along a track region in *next*, storing the index in *next* at which the minimum absolute difference is located as it goes. At the end of this process, the index in *next* is converted into a time offset from *ref* based on the sample rate, among other factors, and the waveforms can then be aligned accordingly.

Each of the absolute difference calculations are completely independent of one another: each calculation can produce an absolute difference and an offset. The results of these independent calculations need only be compared at the end of analysis to find the lowest absolute difference. As such, it is possible to spread these potentially very numerous calculations across several analysing nodes.

3.2 gVAIL

Given a telemetry stream (aligned or not), an analyst wants to look for anomalies which may represent excessive vibroacoustic events, or may represent simple errors introduced in the telemetry collection process. If the analyst decides an anomaly is an error, it may be possible to 'fix' the error. VISPERS AI Lab (VAIL) uses artificial intelligence techniques, specifically neural nets, to locate anomalies in the telemetry streams and suggest ways to fix them to the human analyst. VAIL looks at the data stream by doing a DSP analysis, and one or more *neurons* fire when an anomaly is detected. If the human analyst agrees on the location and type of anomaly, the behaviour of those neurons is reinforced as necessary. VAIL then suggests a fix for the anomaly from several possible tools based on the choice the human analyst has made for similar anomalies in the past.

VAIL is extremely compute-intensive. Not only does it look at (do a DSP analysis of) every point in the waveform and the neighbourhood around that point, but different neurons look for different anomalies (noise spikes, data dropout, DC drift or signal bleed-through to name just a few) so it is useful to split VAIL's computation across multiple machines. This task is facilitated by VAIL's highly parallelisable algorithm. Because VAIL's analysis on one part of a file is independent of its analysis on another part of the file, a task can be divided so that each machine analyses a separate section of a data file. To leverage the computational power of machines across heterogeneous networks, it is advantageous to employ grid computing, distributed computing that allows computational resources to be shared across disparate networks and organisations (Foster and Kesselman, 2004).

3.3 Service scheduler

Both gTACT and gVAIL are available to the analyst through the gVISPERS Time History Editor client. To run instances of these services, however, the client requires a scheduler service, an analyser factory service, a node information publisher service, and Globus' index service (http://www-unix.globus.org/toolkit/docs/3.2/infosvcs/ws/key/index.html). The client submits analyses to the scheduler service, which queries Globus' index service to find out which nodes are available. The scheduler (or resource broker) then decides which node should perform which part of the analysis. Currently, it makes this decision using a round-robin algorithm in which each node is assigned an equal amount of work, defined by the number of data points the node is to analyse. Each node runs both the analyser service, which provides an interface into the TACT or VAIL code, as well as the node information publisher service.

In the earlier Globus 3.2.1 version of gVAIL, the node information publisher exposed a Service Data Element (SDE) called `gVAILNodeInfo` and caused the index service to subscribe to this SDE. The scheduler would use the SDE's `originator` attribute to determine the node's IP address and its `goodUntil` attribute to determine whether the node is available. In the Globus 4.0 version, which is based on the Web Services Resource Framework (WSRF) (http://www.oasis-open.org/com-mittees/tchome.php?wgabbrev=wsrf), these SDEs have become *resource properties.* In the future, more information will be published through resource properties, such as the node's load, which the scheduler could use to implement a more effective resource brokering algorithm.

The sequence of steps taken when the user submits an analysis is depicted in Figure 3. The steps are as follows:

1 when the user tells the client to begin an analysis, the client talks to the scheduler, telling it which data file the user wants to analyse

2 the scheduler queries the index service, asking for a list of machines available to perform analyses

3 the index service responds with the list of nodes

4 the scheduler decides which node should perform which part of the analysis and passes this information on to the client

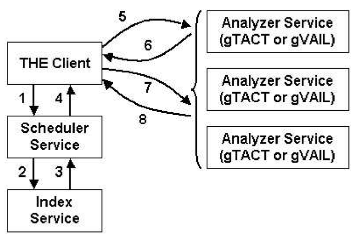
5 the client tells the analyser service on each node to perform its assigned portion of the analysis

6 each node spawns off an instance of the analyser service to perform the analysis, sending status messages to the client to keep it informed of the node's progress

7 when the node indicates that it has finished, the client requests the results

8 the node sends the results to the client.

Figure 3 gVISPERS service interactions during an analysis


THE Client
Scheduler Service
Index Service
Analyzer Service (gTACT or gVAIL)
Analyzer Service (gTACT or gVAIL)
Analyzer Service (gTACT or gVAIL)
1
2
3
4
5
6
7
8

4 Design considerations

4.1 Service model vs. job model

The primary objective of the project was to determine the feasibility of using the Globus Toolkit to grid-enable applications in a production environment. Consequently, the authors looked at both the job (http://www.globus.org/toolkit/docs/3.2/gram/key/index.html) and service models (The Open Grid Services Architecture, 2005) for grid-enabling an application. Eventually, however, the service model was chosen.

The models differ in a number of ways, but most important for this project are their inter-node communication mechanisms. In the job model, the process sending information outputs it to stdout, which is sent to the receiving node. This node must then parse the stream of text. Using the Web Service Resource Framework (http://www.oasis-open.org/committees/tchome.php?wgabbrev=wsrf) and Web Services Notification (WSN) (http://www.oasis-open.org/committees/tchome. php?wgabbrev=wsn), nodes can communicate through either grid service function calls or through resource properties. Grid services may publish resource properties, and other services can subscribe to them. Subscribers may be notified when the property's value changes.

The service model has many advantages over the job model that make it a more attractive candidate for grid-enabling VAIL. It is desirable to have nodes send status messages to the client as they are performing an analysis, and property notifications work well for sending such small messages. As discussed in the next section, properties are also useful when the client retrieves results. Also, the service model is more object-orientated and can be used to grid-enable existing Java applications more seamlessly than the job model.

As an aside, it is important to note that the job model does have some advantages over the service model. It has been used more extensively, so more third party schedulers and tools are available for it. Also, the job component of Globus has gone through many more stable releases and has been more thoroughly tested.

4.2 Grid communication

Using grid services, Globus provides support for communication between different parts of the grid through three different mechanisms: function calls, property exposure, and property-based notifications. Each has advantages and disadvantages, and gVISPERS utilises all three.

4.2.1 Function calls

Globus provides support for function calls according to the standard grid service model: function (method) names for the grid service are exposed and can be invoked directly by the client. When the client makes a remote function call on the service, the service performs some computation and returns a value. The primary limitation of function calls is that the essential nature of most grid applications calls for a high degree of concurrency, while function calls block. This can be dealt with by using threading either on the client, the service, or possibly some combination of the two. The Time History Editor client uses function calls to invoke the scheduler and accepts the blocking behaviour, since it needs the scheduling results before it can proceed. It needs to communicate with multiple analyser services in parallel, however, so threading must be used for this. For the sake of simplicity, gTACT and gVAIL use threading exclusively on the service end. The client makes a function call that returns immediately to start the analysis. Once the analysis is started on the grid nodes, the other two mechanisms are utilised for further communication.

4.2.2 Property exposure

Globus 3.2.1 supported the notion of *SDEs* which became *resource properties* in GT4. Resource properties behave essentially like public fields in a object. In this case, the service itself is the object. The service can store data in them, and the client can retrieve that data. Clients can also set the values of properties, although gTACT and gVAIL do not utilise this feature. An analyser service stores the results it generates in a resource property. Once the client determines that a service has completed its analysis (see next section for how), it retrieves the results from the appropriate property associated with the service.

4.2.3 Notifications

In addition to simply storing and exposing data, resource properties form the basis for notifications in Globus 4.0.

To receive notifications, the client subscribes to individual properties on a specific service. The service can, at any time, notify all listeners of a change to its properties, even if no change has actually taken place. These are *push* notifications in that the data stored in the property is sent to the client with the notification. *Pull* notifications differ in that the only data sent is a notice indicating that a change has occurred. gTACT and gVAIL use notifications to keep the client up-to-date on the status of the services, including completion.

4.3 Reliable index service

The difficulties originally faced with a reliable index service in GT3 have been resolved by the new index service implementation in GT4 (http://www-unix.globus.org/

toolkit/docs/4.0/ common/key/index. html). GT3's index service required the implementers of the gVAIL code to add a dynamic resubscription method to their *gvail_nodeinfopublisher* service in order to ensure accurate service creation and destruction. This was necessary since, once a service was published to the GT3 index service, there were issues in removing it even after the service had died. The gVAIL team implemented this by using the SDE *gooduntil* attribute, which any querying service would check to see if that service had yet expired and which the dynamic resubscription mechanism would periodically refresh.

Combining the use of Virtual Organisations (VOs) with the WSRF-compliant GT4 index service, this was no longer a problem in the updated code. Each analyser node had its own Globus container running and, therefore, its own index service. Local grid services (i.e., the *NodeInfoPublisher Services* and the *AnalyzerServices*) were published first to these local index services which were, in turn, themselves subscribed to a central index service (http://www.globus.org/toolkit/docs/4.0/info/WSMDSSamp les.html). The central index service could then be configured to refresh these index service subscriptions periodically (the default being ten minutes), at which point all non-existent services would be weeded out and removed.

5 gVISPERS performance

Since gVISPERS is a continuing development project involving different teams, its components have been built and tested in different environments as feasible. The earliest work was done on gVAIL (Bentow et al., 2005) with subsequent work being done on gTACT. For completeness, we present basic performance information for gVAIL done under GT 3.2.1 which focuses on the performance of the gVAIL function rather than that of the Globus Toolkit. For gTACT, however, we investigate its performance in the context of being a GT4 service. We also present data for the use of the gTACT and gVAIL services together under GT4.

5.1 gVAIL performance

To test gVAIL's performance, it was deployed on four 2.8 GHz Pentium 4 machines with 640 MB RAM running Red-hat Linux 9. Each machine ran Globus Toolkit 3.2.1 (web service base only) from a local drive. After this initial deployment, another 16 machines (2.8 GHz Pentium 4's with 1 GB RAM) that form part of a cluster computer at Harvey Mudd College were added. These machines used Redhat Enterprise Linux and the same versions of Globus mentioned above. All 20 machines were connected via 100 Mbps ethernet switches to a 1 Gbps ethernet backbone. Globus was placed on a network drive shared among all 16 machines, and data files were distributed on the shared drive. (When running these experimental systems, we cannot read data directly from the operational telemetry database. Hence, for our experimental goals here, we assume that the client has staged data close to the service hosts since we wish to focus on the performance of the grid services and ignore for now any front-end database operations.)

Software testing was performed using gVAILBatchMod-eClient, a special client designed to perform many consecutive gVAIL analyses. gVAILBatchModeClient reads an XML configuration file that describes the analyses to be run. It then executes each analysis sequentially.

A few factors that affect the runtime of gVAIL should be noted. First, VAIL uses short circuit evaluation to determine if a particular point can be identified as an anomaly, so the overall runtime is dependent on the number of anomalies in a file. Thus, if one node receives an unfair number of anomalies, it will take much longer than the rest of the nodes, delaying completion of the analysis. Finally, in order to minimise the systematic error that may be introduced by repeatedly using the same machine, the scheduler randomly selected which nodes to use.

Four data files with varying total points and anomalous points were analysed using a variable number of nodes. The characteristics of these files are shown in Table 1. The number of nodes ranged from one to 20 in increments of one, and for each number of nodes, each file was analysed 20 times. Finally, a serial VAIL analysis was performed on each file 40 times to establish a performance baseline.

Table 1 Characteristics of the sample files

	Total points	*Anomalous points*
File 1	3,200,000	72
File 2	1,784,000	640
File 3	1,783,993	629
File 4	219,071	813

The speedup curves for all files using 1–20 nodes are shown in Figure 4. Speedup was computed by using the serial VAIL performance. For each file, this serial performance was 150, 81, 210 and 17 s, respectively.

As expected, analysing the smaller file using gVAIL on one node was slower than analysing it using serial VAIL.

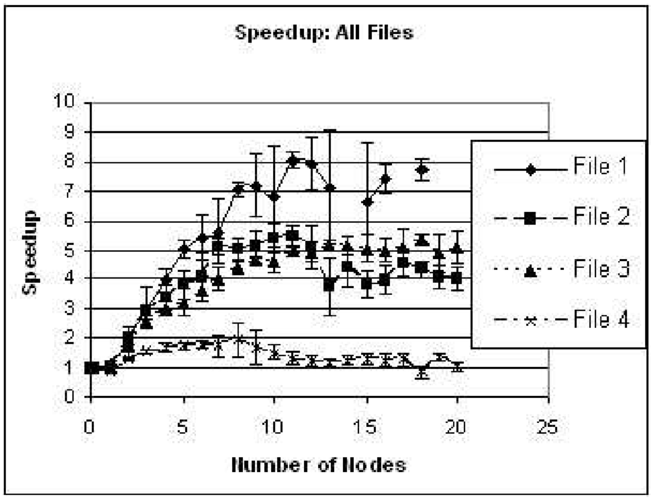
However, analysing any other file using gVAIL on one node was slightly *faster* on average than using serial VAIL. This is not statistically significant, however, because the baseline measurement's standard deviation in all cases nearly encompasses the standard deviation of the one node point.

Files 2 and 3 show similar results. This is expected because they contain comparable numbers of both total data points and anomalous points. However, note that File 2's absolute runtime is much lower due to the type and distribution of anomalies. For testing purposes, File 2 was generated artificially so that it would have an even distribution of anomalies. Thus, when the scheduler divides up the file based on number of points, it more evenly divides the actual work.

During the analyses of File 4, it started to behave erratically as the number of nodes increased beyond eight.

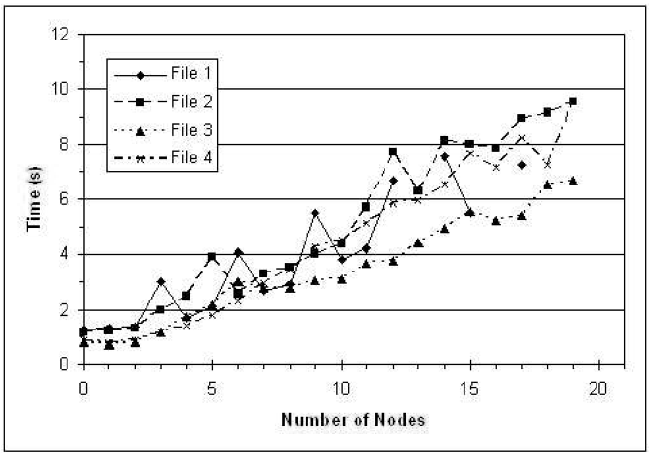
This behaviour was also seen, to a lesser degree, in the other waveforms. Part of this is due to the large number of back-to-back analyses performed. The erratic analyses were performed after 200 back-to-back analyses, and the performance of the grid degraded noticeably over time unless the nodes were restarted periodically. The period required varied with the size of the file. For instance, with the large file, services begin to fail at the file load stage after approximately 80 analyses. With the smaller file, services began to fail at this stage after approximately 500 analyses. The dependence on file size suggests that garbage collection is not working perfectly. It is also likely that some of the erratic behaviour was caused by load on the cluster nodes from other projects. Load on the nodes would produce more pronounced aberrations as the number of nodes utilised increased.

Figure 4 gVAIL speedup



However, most of the contribution to increasing runtime seems to be communication costs, as demonstrated by Figure 5. Also, from Figure 5 it is possible to observe that the communications time is file size independent. Thus, a more advanced scheduler for future work in this project might consider how to balance communication cost with analysis cost for more efficiency.

Figure 5 gVAIL communication times



Overall, grid-enabling VAIL increased its performance. Moreover, the number of nodes at which speedup reaches a plateau varies with file size and anomaly distribution, as expected.

5.2 *gTACT performance*

The gTACT service uses essentially the same structure as gVAIL but our interest here is examining the end-to-end performance of the service call chain from the client's perspective, rather than a service's internal performance. Hence, we used *netlogger* (Tierney. and Gunter, 2003) from Lawrence Berkeley Lab to instrument both the client and server machines and collect timestamped event log files to do this analysis. The causal chains of events in these log files can be plotted as *lifelines* that graphically illustrate the relative performance of each step of the call chain.

The gTACT experiments were run on seven machines. One was a 2.4 GHz Pentium 4 with 1 GB RAM and a 75 GB hard drive running Red Hat 3.4.4-2. The six other machines were 2.4 GHz Celerons, each with 512 MB of RAM and a 40 GB hard drive running Red Hat Linux 3.3.3–7. These machines were connected by 100 Mbs ethernet. Reasonable clock synchronisation was accomplished using the Network Time Protocol.

We examine the gTACT service invocation performance using three data sets as shown in Table 2. These data sets are of different sizes and are called small, medium, and large. (These designations are only relevant for this paper.) We give the total number of points in each file and also the number of *overlap points* since this is what will actually drive the TACT analyser performance.

Table 2 Characteristics of additional sample files

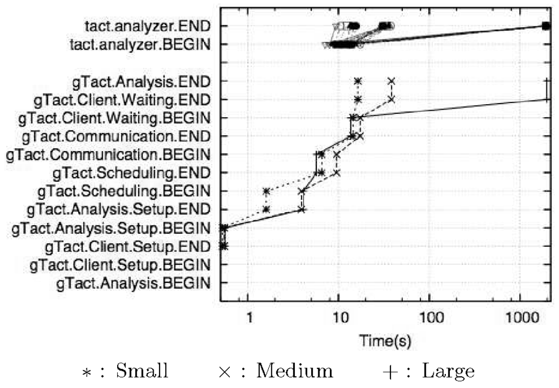
	Total points	*Overlap points*
Small	122,412	5,000
Medium	604,348	10,000
Large	3,023,999	730,000

The lifelines for gTACT service invocations on these three files are shown in Figure 6. All events occur in `BEGIN-END` pairs which define time epochs that have the following meanings:

- *Analysis*. This epoch spans the entire gTACT service invocation from the client side including the return of results, i.e., the the time offset between the waveforms.
- *Client.setup*. During this epoch, local client-side variables are set and the Globus notification facility is initiated.
- *Analysis.setup*. Here two waveforms are read from disk and the overlap region between them is found.
- *Scheduling*. During this epoch, the client contacts the SchedulerService and retrieves a list of available analyser nodes and assigns a range of points in the overlap region for each to analyse.

- *Communication*. Here the client contacts the AnalyzerService on each node and initiates an actual analysis, i.e., the AnalyzerService starts a thread.
- *Client.Waiting*. Finally the client waits for notifications from the analyser threads that were initiated in the Communication phase. This epoch is complete when all analysers have notified the client that they are done and have returned their results.

Figure 6 gTACT lifelines



We note that the events `gTact.Analysis.BEGIN` and `gTact.Client.Setup.BEGIN` are not actually plotted since these event times are essentially zero and we are using a log scale on the time axis.

Along with the invocation event chain, individual TACT analyser executions are also shown at the top of the chart with the `tact.analyzer.BEGIN` and `tact.analyzer.END` events. For each data set, seven analysers are run. Also, rather than letting the client machine become idle during analyses, this machine was made available for scheduling one analyser.

Clearly, for the small data set, the service invocation overhead dominates the execution time. Over a 16.3 s epoch, the analysers are running for less than seven seconds. While each analysers runs about two seconds, it takes about 8 s to get the first one started. The scheduling and communication functions take the most time, at about 4.9 and 7.9 s, respectively.

For the medium and large data sets, however, the real work being done by the analysers comes to dominate the end-to-end execution time. The analysis setup, scheduling and communication times are all relatively constant, while the analysers run much longer. The first analysers start up between eight and ten seconds and continue to run for approximately 30 and 1978 s, respectively.

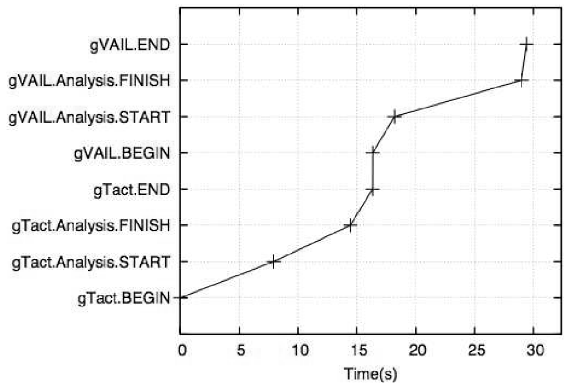
While service invocation overhead is non-trivial, we see that it is quickly dominated by the analyser execution times when the data size exceeds roughly half a million points. We also note that the TACT function can be parallelised by both decomposing a single alignment over several nodes, and by performing multiple alignments simultaneously. Hence, while the service invocation overhead is non-trivial, there is clearly sufficient parallelism in the TACT function to amortise the overhead with suitably sized data sets. More to the point, though, TACT (and VAIL) are suitable for use in a service architecture allowing for tremendous flexibility with regards to user environments.

5.3 *Concatenated service performance*

Another important aspect of service-oriented computing is the concatenation of service invocations. gTACT and gVAIL will be run successively with data being transferred between them through the client. While this data will be relatively small, it is still important to quantify and understand the overhead of transferring it through the client and the possible benefits of doing a 'third-party' transfer directly between the gTACT and gVAIL hosts.

To investigate this issue, we will run both services on the data sets identified in Table 2. We will, however, simplify the event chain we are examining as shown in Figure 7 giving the lifeline for a typical service concatenation on the small data set. In this plot, each service has a `BEGIN-END` event pair with an analysis `START-FINISH` event pair within it. The `BEGIN-END` event pair is the total service execution time on the client side while the analysis `START-FINISH` event pair brackets the start of the first and the completion of the last of seven analysers. Hence, one can easily determine how much start-up and completion overhead each service has, and the time between the completion of the TACT analysers and the start of the VAIL analysers.

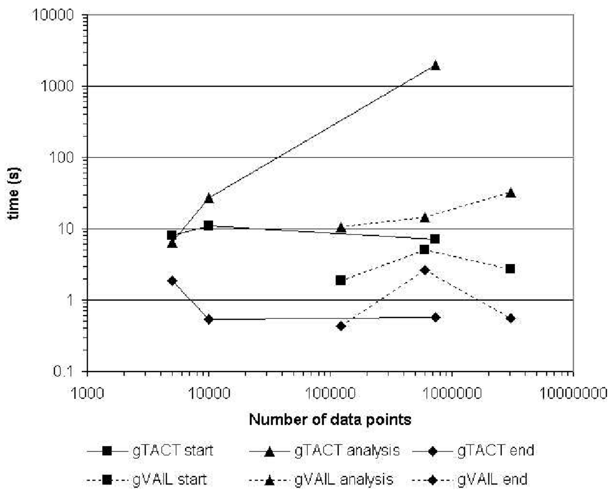
Figure 7 A concatenated gTACT and gVAIL lifeline for the small data set



When examining service concatenation for the larger data sets, however, such lifeline plots become difficult to read since they can span several orders of magnitude with long and short intervals interspersed. Hence, we plot the data as shown in Figure 8 giving the `start`, `analysis`, and `end` time epochs for each data set. We plot the gTACT analysers against the number of overlap points, while we plot the gVAIL analysers against the total number of points. We see that for both service start-up and completion, there are no trends related to the number of data points involved. Some variability is seen which is reasonably due to network delays. For both of the analysers, however, the execution does increase, and quite dramatically in the case of gTACT. Hence, for the larger data sets, the analysis time dominates the service overhead. While the service overhead is larger than expected, it is still tolerable.

When interpreting these results, we must remember several gVISPERS design choices. The start-up time includes the SchedulerService time for the discovery and scheduling of analysers. Assuming that the output of one analyser does not affect the selection and scheduling requirements of a subsequent analyser, it should be possible to schedule the gVAIL analyses without waiting for the gTACT analysers to complete, thus improving performance. We note that it may also be advantageous to create a scheduling dependency where the gTACT and gVAIL analysers hosts are very 'close' together over the network, or are even the same hosts. Observing such a dependency, however, would create a *co-scheduling* issue which carries its own overhead. We must also remember that the raw telemetry data is read by the analyser hosts while primarily control information is issued by the client. This greatly reduces the client-server communication requirements. In the scenario where multiple gVISPERS clients are requesting many data sets, the bandwidth demand for raw telemetry data could become an issue.

Figure 8 Concatenated gTACT and gVAIL performance



6 Related work

This paper has focused on the support of telemetry processing using grid technology. We point out, however, that the term *grid telemetry* has been applied to the monitoring of grids themselves, such as Grid2003 (which has become the Open Science Grid) (Mambelli et al., 2004). The 'distant measuring', or telemetry, of a distributed infrastructure, such as a grid, is certainly not unreasonable but is clearly not the meaning intended here. However, given that the traditional uses of telemetry involve monitoring remote and possibly distributed systems, the application of grid technology to this field is completely appropriate. This has already been recognised by organisations with significant telemetry needs such as NASA (Hinke, 2004).

A fundamental design issue for such monitoring systems is how 'far up' the telemetry chain can grid technology be deployed. This project has dealt with telemetry that has already landed in a telemetry database. It is not unreasonable that 'the grid' could be pushed closer to the system, and sensors, of interest. The DAME project (Austin et al., 2005), for instance, collects sensor data from in-service jet engines for maintenance and reliability analyses. While this data is currently downloaded after the aircraft has landed, the fact that network connectivity during flight is becoming commonplace means that such engine sensors could actually be on-line (even though it will probably remain more attractive for the airline to sell the available bandwidth to passengers to check their email and surf the web). In general, the notion of integrating grids, instruments, sensors and sensor networks offers significant benefits and has always been a prominent concept in the grid community. The Common Instrument Middleware Architecture (CIMA) is endeavouring to develop a web services-based architecture along with a sensor ontology to facilitate grid-sensor interactions (McMullen et al., 2005). The notion of integrating grids and sensors can also be extended to grids and *remote sensing* using on-orbit, satellite-based sensors (Gasster et al., 2006).

We mentioned that VISPERS services will, in practice, be used interactively, even though such interactive use was not the focus of this paper. For investigations into interactive grid use, we can point to projects involving interactive graphics (Kumar et al., 2003) and computational steering (Ali et al., 2004; Pickles et al., 2005).

This paper was also interested in the performance of the Globus Toolkit from the application level. We were primarily interested in how long it takes to initiate a service and a pair of services. While this did give us insight into how the Globus 'overhead' would affect end-to-end application performance, it is certainly possible to perform a more detailed analysis of the entire Globus service invocation sequence. Such as analysis is reported in Torres (2005) where an early version of GT4 was integrated with a Java Instrumentation Suite allowing detailed timelines to be recorded for Java client and server threads in the SOAP interaction sequence. This level of analysis would be useful for tuning GT4 implementations but is generally too detailed for evaluating impact on applications.

Besides analysing individual service invocations, it is also possible to evaluate aggregate throughput performance. Such throughput performance is thoroughly investigated in this Masters Thesis (Raicu, 2005). This work developed the Distributed Performance (DiPerf) analysis tool and was used to compare performance between multiple versions of the Globus Toolkit for operations including the GRAM, GridFTP, and grid services creation and invocation. Such throughput performance analysis would be very interesting for gVISPERS, especially when multiple gVIS-PERS clients are invoking multiple analysis on shared cluster resources. We will consider such work when the gVIS-PERS user base requires it.

This project was also forced to consider the differences between GT3 and GT4 in both performance and functionality. The DiPerf study cited above examined throughput performance for several Globus versions.

The performance and functionality of other Globus services, such as the MDS, have also been systematically examined (Schopf et al., 2006). Other experiences at the application level in moving from GT3 to GT4 have also been reported. One example is given by the Grid Execution Management for Legacy Code Architectures (GEMLCA) project (Delaitre, 2005). Since many projects were facing this issue, the UK e-Science project issued a report (Harmer et al., 2005) looking at specific issues for specific parts of the toolkit. While some parts were relatively unaffected, application code re-factoring could be necessary if certain built-in OGSI classes were used that changed under the WSRF structure. In some cases, scripts could be used to convert between the GT3 and GT4 service forms. A basic how-to guide for such conversions is available (Harmer and McCabe, 2005) and a book is now available for GT4 programming (Sotomayor and Childers, 2005).

7 Future work

Clearly from the human analyst's perspective, the goal is to provide a simple-to-use toolkit that has superior capabilities and performance. They need to flexibly select sensor data, process it using a variety of tools, and quickly arrive at the desired analytical conclusions. Besides the two computationally intensive services discussed here, other tools are in use and in development that will facilitate with data clean-up. Hence, the goal is to make the entire data clean-up process run seamlessly from data acquisition using dbFront, through anomaly removal and the consolidation of multiple telemetry streams into one good stream. Once that stream is created, individual analysis tools can be invoked, the results analysed, and, if necessary, the data can be re-scrubbed based on those results.

This user scenario clearly points to the need to integrate a end-user workflow engine into the gVISPERS client. This would enable analysts to compose their own tool chain executions to serve their specific requirements. This would also make it more feasible for an analyst to include more telemetry streams in their analyses and use the shared corporate resources more effectively. Each computational service could be allocated to appropriate resources depending on the associated computational intensity and desired time-to-solution. With a service architecture approach, multiple gVISPERS clients could be serviced simultaneously. At this point, throughput analysis would become much more necessary.

8 Conclusion

Grid-enabling VISPERS using Globus Toolkit is a viable way to improve performance and provide enhanced capabilities. We have shown, for example, that using as few as 11 grid nodes, gVAIL's runtime improved by a factor of eight. The service architecture approach also provides valuable flexibility with good performance when used with suitably sized data sets.

Since the start of this project, the Globus Alliance officially migrated to Globus Toolkit 4 which adopts the WSRF standards family in order to implement grid services. Though the differences were primarily syntactical, a significant effort was put forth updating the Globus Toolkit 3 OGSI code to its WSRF analog. As was noted in an earlier paper (Bentow et al., 2005), programming a Globus grid service still has a significant learning curve, especially when attempting to utilise Globus-provided tools from within the service's code (e.g., the index service). This difficulty continues to be caused by a lack of adequate documentation.

In the larger context, however, our experience also shows that the service-oriented architecture approach, using grid computing tools, can provide a more flexible system design, in addition to improved performance and increased utilisation of resources. With proper communication and execution scheduling, performance and utilisation could be improved even further. We also wish to emphasise that a workflow management engine is being integrated with the Time History Editor that will allow the flexible control of all toolkit services, include gTACT and gVAIL. Once this is done, we will thoroughly investigate the use of integrated communication and execution scheduling. Clearly we expect to have a powerful yet flexible system that can process more data more quickly than previously possible, enabling a greater understanding of the launch vehicle environment.

References

Ali, A., Anjum, A., Bunn1, J., Cavanaugh, R., van Lingen, F., McClatchey, R., Newman, H., Steenberg, C., Thomas, M., Willers, I. and Zafar, M.A. (2004) 'Job interactivity using a steering service in an interactive grid analysis environment', *Computing in High Energy and Nuclear Physics (CHEP 2004)*.

Austin, J., Davis, R., Fletcher, M., Jackson, T., Jessop, M., Liang, B. and Pasley, A. (2005) 'DAME: searching large data sets within a grid enabled engineering application', in Parashar, M. and Lee, C.A. (Eds.): *Proceedings of the IEEE*, Vol. 93, No. 3, March, pp.496–509.

Bentow, B., Dodge, J., Homer, A., Moore, C., Keller, R., Presley, M., Davis, R., Seidel, J., Lee, C. and Betser, J. (2005) 'Grid-enabling a vibroacoustic analysis application', *6th International Workshop on Grid Computing*, Seattle, Washington, USA, 13–14 November, pp.33–39.

Bradford, K.B., Wong, D. and Bartosisk, J. (2002) 'The vibroacoustic intelligent system for predicting environments, reliability and specifications (VISPERS)', *Proceedings of the Ninth International Congress on Sound and Vibration*, Orlando, Florida, USA, July, pp.486–493.

Delaitre, T. (2005) *Experiences with Migrating GEMLCA from GT3 to GT4*, UK Globus Week, 4–8 April, http://www.nesc.ac.uk/esi/events/519/talks.cfm.

Foster, I. and Kesselman, C. (Eds.) (2004) *The Grid 2*, Morgan Kaufmann, USA.

Gasster, S., Lee, C. and Palko, J. (2006) 'Grid computing for remote sensing data and data analysis', in Plaza, A.J. and Chang, C. (Eds.): *High-Performance Computing in Remote Sensing*, Chapman & Hall/CRC, October 2007.

Harmer, T. and McCabe, J. (2005) *GT3.2 to GT4 Migration: A First HowTo*, Belfast e-Science Center, http://www.qub.ac.uk/escience/howtos

Harmer, T., Stell, A. and McBride, D. (2005) *UK Engineering Task Force, Globus Toolkit Version 4 Middleware Evaluation*, UK Technical Report UKeS-2005-03, http://www.nesc.ac.uk/technicalpapers/UKeS-2005-03.pdf

Hinke, T. (2004) 'Grid technology as a cyberinfrastructure for earth science applications', *NASA Fourth Earth Science Technology Conference*, Palo Alto.

Kumar, R., Talwar, V. and Basu, S. (2003) 'A resource management framework for interactive grids', *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, Seattle, Washington, USA.

Mambelli, M., Kim, B., Legrand, I., Fisk, I., Weigand, J., Grundhoefer, R., Quick, R., Hicks, J., Green, M., Gardner, R., Zahn, A., Prescott, C., Rodriguez, P. and Avery P. (2004) 'Grid2003 monitoring, metrics, and grid cataloging system', *Computing in High Energy and Nuclear Physics (CHEP 2004)*, http://griddev.uchicago.edu/download/grid3/ doc.pkg/presentations/chep04-490-g3mon.pdf

McMullen, D., Bramley, R., Chiu, K., Huffman, J., Devidithya, T., Huffman, K., Lu, W., Tilak, S. and Peralman, L. (2005) *Instruments and Sensors on the Grid: Issues and Challenges*, GlobusWORLD, Boston, MA, USA.

Pickles, S., Haines, R., Pinning, R. and Porter, A. (2005) 'A practical toolkit for computational steering', *Phil. Trans. R. Soc.*, Vol. 363, No. 1833, pp.1843–1853, http://www.realitygrid.org

Raicu, I. (2005) *A Performance Study of the Globus Toolkit and Grid Services via DiPerf, an Automated Distributed Performance Testing Framework*, Masters Thesis, University of Chicago, Chicago, Illinois, USA, May.

Schopf, J., D'Arcy, M., Miller, N., Pearlman, L., Foster, I. and Kesselman, C. (2006) *Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4*, Argonne National Laboratory MCS Preprint # ANL/MCS-P1315-0106, Submitted to HPDC-15.

Sotomayor, B. and Childers, L. (2005) *Globus Toolkit 4 Programming Java Services*, Morgan Kaufman/Elsevier, December, Burlington, MA, USA, http://www.gt4book.com

The Open Grid Services Architecture (2005) *Version 1.0*, Global Grid Form Document GFD-I.030, 29 January, http://www.ggf.org/documents/GWD-I-E/GFD-I.030.pdf

Tierney, B. and Gunter, D. (2003) 'NetLogger: a toolkit for distributed system performance tuning and debugging', in Goldszmidt, G.S. and Schönwälder, J. (Eds.): *Proceedings of the IFIP/IEEE Eighth International Symposium on Integrated Network Management (IM 2003)*, Vol. 246 of IFIP Conference Proceedings Kluwer, pp.97–100.

Torres, J. (2005) *GT4 Instrumentation and Evaluation Project*, http://personals.ac.upc.edu/torres/GT4.pdf

Websites

Globus Toolkit 3.2.1 Documentation, http://www-unix.globus.org/toolkit/docs/3.2

GRAM: Key Concepts, http://www.globus.org/ toolkit/docs/3.2/gram/key/index.html

GT4.0 Common Runtime Components, http://www-unix.globus.org/toolkit/docs/4.0/ common/key/index.html

OASIS Web Services Notification (WSN) TC, http://www.oasis-open.org/committees/tchome. php?wgabbrev=wsn

OASIS Web Services Resource Framework (WSRF) TC, http://www.oasis-open.org/com-mittees/tchome.php?wgabbrev=wsrf

The Globus MDS, http://www.globus.org/toolkit/docs/4.0/info/WSMDSSamples.html

WS Information Services: Key Concepts, http://www-unix.globus.org/toolkit/docs/3.2/infosvcs/ws/key/index.html