Shared Computing Resource for Advancing Science and Engineering Using the Cray XD1 at Rice University[1]

Jan E. Odegard, B. Kim Andrews, Franco Bladilo,
Kiran Thyagaraja, Roger Moye, Keith Schincke
Rice University, Houston, Texas

ABSTRACT: *In this paper we discuss the recent procurement and deployment of one of the largest XD1 systems in the world. The system deployed at Rice University by the Computer and Information Technology Institute in collaboration with the IT Division represents a major addition (~3× increase in theoretical compute capacity) to the existing shared HPC infrastructure. The Cray XD1 will serve as the primary HPC resource supporting cutting edge research in engineering, the physical sciences and the social sciences. This paper focuses on three aspects of the system: (1) the system procurement process, (2) the system deployment and acceptance process, and (3) system operation and management.*

KEYWORDS: XD1, procurement, deployment, acceptance, management

1 Introduction

Rice University is a private, coeducational institution of higher education located in the center of Houston, Texas. Rice is a member of and is located across the street from the Texas Medical Center, the world's largest medical complex, with a total of 42 institutions treating over 5 million patients per year. The first president of Rice University, Edgar Odell Lovett, envisioned building an institution of higher learning that "*…aspires to university standing of the highest grade.…For the present it is proposed to assign no upper limit to its educational endeavor*." inaugural address, 1912.

Since 1912, Rice has grown steadily and today has about 540 faculty members distributed across four academic schools and five professional schools, 2900 undergraduate students and 1900 graduate students.

Rice is still a small institution; in fact, it is one of the smallest research universities in the US. Recognizing that in many cases this was a handicap, Rice created a number of cross disciplinary research institutes in the 1980s focused on areas of research strength where the faculty felt that Rice could be far bigger than the size would lead one to believe. These institutes (in IT, Energy, Biotechnology, Quantum Chemistry and Nanotechnology) were created as virtual organizations with a mission to serve as a catalyst stimulating far reaching collaborative projects supporting the ambitions of the faculty to reach beyond its size. The Computer and Information Technology Institute (CITI) was created to support research in High Performance Computing, Information Technology, and Computational Science and Engineering (all broadly conceived). CITI's mission is to "build a community of scholars that engages in collaborative research and education covering virtually every aspect of information technology and computing."

1.1 Shared Computing Infrastructure

CITI has historically been focused on supporting faculty members seeking to engage in basic research in information technology. This mission expanded in 2002 when it became clear that our continued success in research would be severely limited without a focused effort aimed at providing large scale, campus wide,

[1] This work was supported in part by a Major Research Infrastructure grant from the National Science Foundation (CNS-0421109), Rice University and partnerships with AMD and Cray.

computational resources to an increasing number of faculty members whose research was either primarily computational or, as was increasingly the case, a strong combination of computation and experimentation. At that time CITI, under the leadership of the Director and Executive Director of the institute, coordinated a team of faculty members that wrote a successful proposal to the NSF Major Resource Infrastructure (MRI) program for a 1 TeraFLOP Intel Itanium2 shared computing resource. With this first grant Rice procured a 280-processor HP Itanium2 system. Recognizing that additional capacity would soon be needed, CITI coordinated a second proposal to the same NSF program in 2004, leading to the procurement of the current 3 TeraFLOP Cray XD1.

2 Cray XD1 Site Overview

The Cray XD1 deployed at Rice is configured in five racks. Three racks are populated with 28 XD1 chassis (12+12+4), one rack contains two 144-port switches, and the final rack is populated with about 20TB of usable storage (6TB Lustre and 15TB NFS). Additional details of the system can be found by visiting http://www.citi.rice.edu.

The system was intentionally designed with additional core infrastructure that we can expand by adding eight more XD1 chassis without increasing port count on the spine switch. The fully populated configuration can house 48 XD1 chassis in 4 racks. This spare infrastructure was added to accommodate near term needs for capacity growth as well as to support groups and individuals that might have special needs but could still benefit from the economy of scale and leveraged system administration support. Growth can either be in the form of system-wide large scale shared investments or through the investments of individual research groups for "private partitions", much like a condominium.

2.1 Procurement Process

An open invitation for vendors to present technology solutions that would be available and satisfy broad guidelines was made in the form of a written RFI (request for information) document. The RFI loosely specified technology needs, targets size and time line. A number of vendors opted to participate in this RFI round giving in-depth NDA technology and roadmap presentations to the Rice procurement team. Based on technologies presented, we issued a thoroughly detailed request for proposals (RFP) asking vendors to prepare bids on a substantial system built around the AMD OpteronTM processor. Vendors were given a cost target and were generally asked to bid a solution that they felt could be deploy by late 2005. While dual cores were not a requirement, we did expect many vendors to propose them.

The system was required to run 64-bit Linux and the bid required vendors to include end-to-end hardware and software maintenance for three years with options for additional years of support. Each vendor was required to prepare a comprehensive written bid providing extensive benchmarking results and addressing each of the specifications outlined in a meticulous Hardware and System Software Requirements section.

Questions by vendors during the RFP process were required to be submitted in writing. Questions and corresponding answers (without the vendors' name) were posted to a password protected shared web page that was accessible by all vendors. This web page was a natural extension of the RFP document and served as the only interaction between vendor and customer during the RFP process. This provided a very insulation

2.2 Benchmarking

The benchmarking requirements consisted of two types: (1) system components benchmarks and (2) customer code benchmarks. A system configured with proposed hardware (or largely equivalent hardware) with at least 36 processors was required for benchmarking. A detailed report of all benchmarks should, in addition to reporting performance results, also provide enough information about the system and all software for the results to be reproduced. We also requested that if large scale benchmarking could only be done on single core systems, then dual core performance must be predicted based on limited tests of early access systems, which each of the vendors had indicated would be accessible during the RFP process. Vendors were free to use their preferred compiler suite, but must specify versions (with any patches applied) and all compiler flags that were used for each of the benchmarks. Benchmark code changes of any kind were forbidden.

2.3 System Component Benchmarks

To illustrate the scope of the benchmarking requirements, we are providing a brief summary of the benchmarking requirements. All benchmarking results were required to be presented in a spreadsheet. Networking configurations; OS kernel versions, drivers and patches; and application versions were required for all benchmarks.

HPC Challenge

- Run benchmark on a 4, 16, and 32 processor dual socket configured system with proposed hardware
- If available please report results obtained by running on larger systems

- Run MPICH, and if applicable a vendor preferred commercial MPI

NPB (NAS Parallel Benchmarks)

- Run mg, cg, and sp in the NPB 2.3 benchmarks suite with Class C size over the proposed computational network with dual socked nodes
- Run MPICH, and if applicable a vendor preferred commercial MPI
- Report results and provide graphs for the following cluster configurations
- For mg
  - 4, 8, 16, 32
  - Recommended: 64, 128
- For cg
  - 4, 8, 16, 32
  - Recommended: 64, 128
- For sp
  - 4, 9, 16, 25, 36
  - Recommended: 49, 64, 81, 100

If possible please report results on file system using Bonnie++ as follows

- Bonnie++ (hard drive and file system performance)
- Run Bonnie++ on the proposed compute node hardware, master node hardware, and file server hardware
- Run with file sizes that are 5 times the physical memory
- Report all results (throughput and CPU usage)
- Run Bonnie++ over NFS (hard drive and file system performance)
- Run Bonnie++ over NFS from proposed file server to two proposed clients over the proposed I/O network
- NFS server is the proposed file server
- NFS client is the proposed compute node
- Export file system using the following options:
  - sync, rw, no_root_squash
- Client mount options are variable with the following exceptions that must be set:
  - hard, intr, tcp
- Report NFS server configuration used including export options
- Report NFS client mount options used
- Report OS version including kernel, kernel patches, GigE NIC drivers, version, and options for the NFS file server and NFS clients.

2.4 Customer Code Benchmarks

This section lists the vendor-specified application codes that we asked them to benchmark. The list consists of a mix of commercial (most vendors had the appropriate licenses), open source and internally-developed codes. Some of Rice codes were subject to executing an NDA. Additional details on problem size for each code and all the necessary information for running benchmarks on Rice code was provided to the vendors through the RFP web page extension.

2.5 Scientific Code

- Gaussian
- Amber 8
- NAMD
- In-house developed CFD
- In-house developed molecular modeling code

2.6 Selection Process

Although making final procurement decisions are a complicated and time consuming task, we guided this process using a small set of criteria that was supported by the RFP specifications. Since the main objective of the procurement was to support Rice's specific computational needs, the first level ranking was heavily biased towards benchmarking performance. However, this in and of itself was not sufficient since benchmarking can only cover one subset of the codes we expected the system to support. Based on this complexity, we used the combination of criteria listed below.

Primary criteria:

- Benchmark performance as a predictor for capability
- Peak performance (CPU count) as a measure of capacity
- Cost

Secondary criteria:

- Partnership opportunities
- Power and cooling requirements
- Weight

2.7 Procurement Decision

A total of six vendors responded to the full RFP. While each system had its unique features and was roughly identical in peak performance (about the same number of compute cores), there were significant differences in designs and capabilities. Using the above criteria for selection, however, the XD1 was a clear first on most of the benchmarks as well as a clear winner on each of the secondary criteria.

3 Deployment and Acceptance Criteria

We formalized the acceptance of the Cray with a Memorandum of Understanding. This document basically

provided detailed definitions of several terms relating to the system's state and outlined the necessary steps for the system to be accepted.

The acceptance test had two phases, a Site Install Test and a Production Test. The purpose of the Site Install Test was to reproduce the performance of the system before & after shipping it to Rice from Cray's facility in Chippawa Falls (an extensive subset of the site install test had to be executed while the system was in the Cray facility). The Site Test consisted of reproducing the pre-shipping LINPACK results as well as a few key tests within the HPCC benchmarking suite. After the completion of the Site Test, a 60-day Performance Test period was began. The purpose of this period was to accumulate 30 consecutive days of Operational Use. We linked the definition of Operational Use to an "Effectiveness Level".

For the purpose of the Production Test, the "Effectiveness Level" of the System was computed as follows:

$$Effectivness\ Level = \left(\frac{Operational\ Use\ Time\ (hours)}{Scheduled\ Use\ Time\ (hours)} \times 100 \right) \times \delta(t)$$

and $\delta(t)$ is a binary function with the value 1 or 0. The function except during any time period when the following is true:

- The Lustre file system is not available (this includes all hardware and software associated with the operation of the Lustre file system)
- The NFS file system is not available and working (this includes all hardware and software associated with the operation of the NFS file system)
- Both master nodes are down
- All login (development) nodes are down
- The job scheduler is down
- Switch fails

Operational Use was defined as a system Effectiveness Level of at least 96.5%, corresponding to 25 hours of downtime out of 30 days. In this way, we hoped to avoid any ambiguities as far as whether the system was "up" or "down" from an operational viewpoint.

3.1 Measurement & Tracking of Operational Use

Maintaining a high (>96.5%) Operational Use was the main goal of the Performance Acceptance phase. We met daily to track the progress of the testing and to log the accumulated Effectiveness Level, annotating problems as they arose. Rice was in charge or providing compute load on the system during the production test. This was accomplished with repeated and extensive benchmark runs as well as a small group of heavy-hitting friendly users.

4 Problem Diagnosis and Resolution

4.1 Hardware

We saw only a very small number of hardware issues, and the hardware failures we observed were primarily caused by an abnormal level of power harmonics generated due to the high load drawn from one single PDU to power the entire system.

Cray engineers said: "Our experience [had learned this after the system had left the Cray facility] shows that when more than a dozen or so XD1 three phase power supplies are fed by a single transformer (PDU), the accumulated current harmonics create a voltage harmonic above 5%. And we've found that when ambient voltage harmonics exceed 5%, our power supplies begin to fail."

We measured and found only very limited harmonics on the primary power feed and hence the harmonics problem was in fact created by the three-phase power supplies putting a significant amount of harmonics back into the power system, in some cases causing 10-12% power harmonics. A single transformer (i.e., a PDU) cannot mitigate this harmonic load sufficiently. As a result, eight of the three-phase power supplies ultimately failed during the Acceptance Test, well above statistical expectations.

Resolution Options

Replace all power supplies with single-phase power supplies. However, while this would have solved the problem, single-phase power supplies were not as reliable as their three-phase supplies and the MTBF would likely be higher.

The second alternative was to add two more PDUs and distribute the load equally across all PDUs. While this would not eliminate power harmonics it would reduce the harmonics level to an acceptable level.

Implemented Solution

Working with Cray Rice was able to, in a short amount of time, to install two PDUs and distribute the load across all three PDUs. This reduced the harmonics and solved the problem for the system at its current size. Future growth will, however, either require additional PDU capacity or use of single-phase power supplies.

4.2 Software

LDAP proxy randomly crashed

The first issue we encountered with LDAP on the XD1 was that Active Manager's supported LDAP configuration, unlike Cray's NIS implementation, required an external LDAP server instead of providing the option of functioning as an LDAP server itself.

Our first approach was to install SuSE's openLDAP RPMs and set up our server on the base node. This solution didn't scale well due to the way SuSE built the RPMs. The symptom we observed was that large jobs failed to start. Further investigation proved that the SuSE's threaded LDAP server had been built with a MAX FD table (maximum number of concurrent connections or open files) of 1024. This limit was easily overrun when PBS tried to lookup gecos information on large parallel jobs. (Cray was notified with a suggested fix)

The second attempt to address the issue was to use an LDAP proxy. This limited the number of outgoing connections to the server as well as decreased the query request time. Unfortunately, while this looked like the perfect technical solution, the openLDAP proxy backend generated bugs that we traced along with Cray for weeks without success. At one point, we decided to compile our own openLDAP package, but it exhibited the same behavior.

We homed in on the current configuration of an external threaded openLDAP server running on Linux Redhat Enterprise 4 with a MAX FD table of 8192. This server is located outside the cluster Ethernet network and some client tuning parameters (e.g. NSCD) were done on the Cray XD1 system to decrease the request latency. This solution has been working flawlessly since its implementation.

PBS died randomly

This issue was likely caused by the ongoing LDAP issues we had during that timeframe.

A user code ran across a max FD limit (1024) in the PBSPro moms. Cray provided a patched PBSPro rpm with this FD limit increased to 8192.

IMB could not be run consistently

IMB running with large processor counts (512 and beyond) required more memory buffers than could be stored in the RapidArray registration table. When the table became full, the IMB application received a "buffer registration failed" error and terminated. Unfortunately, this triggers a bug when the IMB application is terminated. The result of this bug is that some memory corruption occurs and all subsequent application runs (of any MPI application, not just IMB) fail to start with a RapCreateCQ error.

The registration table simply wasn't large enough to run an IMB job at high processor counts. To run the job successfully, Cray increased the size of the registration table from 4000 to 128000 entries. This allowed IMB to execute successfully and avoids the above-mentioned bug as well.

Cray support and R&D worked a patch for 1.3GA that included the following changes:

- kernel SP3 updates
- code changes to fix the IMB issue
- Lustre load that works with kernel SP3

The 1.3 patch load was deployed end February 2006 and this fix would be part of 1.4GA. The 1.4 load would also correct other problems; most notably support a 144 port switch.

Pros: The workaround allowed us to continue working without an interruption of service

Cons: The workaround does consume some memory. Not sure what effects this workaround has on the stability or performance of the rapid array network.

LVM at reboot requires manual intervention

The fileserver nodes were equipped with fiber channel adapters that had no built-in support in Cray's supplied kernel. The driver for these adapters was being loaded at a later stage of the system start-up. The problem arises because SuSE, like many other Linux distributions, expects that hardware is "ready" before trying to initialize services such as LVM, quotas, etc. In our case, hardware support wasn't present to initialize these necessary services and the result was a non-functional fileserver.

We suggested using a kernel init RAMdisk which is a typical way of solving these issues but it conflicted with AM/LSS.

The current solution is to use a boot local script that runs at the very end of the start-up process and ensures that hardware support and services are loaded in the correct chronological order.

The permanent solution requires kernel fix from Cray

HPCC died after reboot, runs OK after resubmission

This never happened again after we moved into the current LDAP solution. However, we will test this carefully as we move to 1.4.

Current System Issues

- We are unable to move nodes between partitions due an issue with L2F route calculations. This has been addressed on AM 1.4.

- System provisioning scalability is unacceptably slow. We can only provision 8 blades at a time. This issue has been brought to Cray's attention.

- Lustre crashes sporadically. We have two tickets open with Cray for this issue. There is a lustre patched kernel that Alabama is currently running, as they had similar issues. We are waiting for Cray to confirm the stability of the fix.

- The base node crashes after approx. 2 weeks of uptime. The kernel on base logs numerous errors relating to the AMD IOMMU controller and it eventually crashes the system. We have recently spoken with Cray Engineers about this issue and they may soon be able to provide a patched kernel, but without lustre support. We shall wait until their first testing site has validated this fix and Cray moves forward with the lustre version of it.

- cfengine is a resource hog, it puts a high load on base without doing too much work. Cray may have a version of cfengine in AM 1.4 that is supposed to be much more relaxed.

5 System Management

5.1 Research Computing Support Group

The Research Computing Support Group (RCSG) works for Kamran Kahn, the Vice-Provost of Information Technology, in the Academic and Research Computing Department directed by Rick Peterson. The group is managed by Dr. B. Kim Andrews, Manager of Research Computing, and is staffed by four highly-trained technologists with an opening for Research Computing Application Developer. The goal of the RCSG is to increase the productivity of Rice's research faculty. There are 4 main areas of service that the RCGS provides to the Rice campus: system administration, application support, user support, and data management.

5.2 Commissioning for Production

After the acceptance of the machine, we completely re-commissioned the machine and wiped out all accounts. We then quickly ran through some of the more stressful benchmarks and confirmed that the machine was properly configured.

In order to control the workload generated from porting activities, we planned to move users onto the machine in three phases. We first brought in all users who had participated in the Acceptance Test and the faculty members who were lead investigators of the MRI grant proposal. We then quickly opened up the machine to the other 30+ co-authors of the MRI proposal. The system will be opened for general access by Rice faculty mid May 2006 pending finalization of a revised online account application system.

5.3 Operational Overview

We use several industry standard packages to manage and monitor both the Cray as well as other large shared resources. For resource management and scheduling we use maui on top of PBSPro. We use Active Manager for internal component availability monitoring and alerts. A separate nagios server integrated into IT's campus Operations Support is used for external availability monitoring. We have set up a ganglia grid for resources managed by the RCSG to provide real-time monitoring of utilization and performance.

We monitor system availability using Active Manager. Nagios is also used to monitor availability (ping & ssh) of the 4 login nodes. We employ Change Management procedures for all modifications made by the RCSG. Any non-RCSG changes that might impact the system are also maintained in the Forward Schedule of Changes to avoid "Change Collisions."

Utilization is measured at several levels in order to maintain high system efficiency. To detect hardware problems/inefficiencies at their onset, we watch the individual real-time node utilization patterns (CPU, memory, network, etc.) using Ganglia. Queue statistics (e.g. # of jobs submitted & average wait time as a function of nodes requested, CPU usage by user, group, project, etc.) are gathered using internal Perl scripts that post-process a database generated in maui on a daily basis. We use this data to monitor the users' overall throughput and queue submission strategies. We also use Perl scripts to analyze individual user application performance/efficiency (CPU, memory, i/o, etc.) to detect inefficient code. Users are brought in for consultation whenever we find queuing or performance anomalies. This has provided an excellent opportunity for mutually beneficial collaboration and relationship building between support staff and users.

Problem resolution is handled using Rice's IT-wide helpdesk software, Request Tracker. The software helps greatly to keep track of the complex user problems that are typical of supporting research on a shared resource. For problem avoidance (which is much more effective than problem resolution!) we are implementing a formal

partner relationship with principal investigators called a Service Assurance Partnership. One of the outcomes of this is that we meet regularly with individual research groups to discuss their end-to-end computing environment, present their usage statistics, problem resolution status, upcoming relevant infrastructure changes, etc.

Our queuing policy is currently very simple. Fair Share is implemented with a four-hour maximum runtime and no limit on the number of CPUs requested. We now heavily favor the Quality of Service for multi-node job requests to prevent the queue fragmentation caused by high numbers of single-CPU jobs, which we have observed on other systems and diagnosed it on the Cray before the bias was in place. Requests for changes in or exceptions to the queuing policy may be made through Request Tracker. During the porting phase of an application we have made several exceptions for timing, parallelizing and optimizing code. Prior experience indicates that there will be the need to accommodate large scalability studies and time-sensitive requests as well.

The partitioning of the machine for dedicated use continues to be an area of focus and interest. Currently, we have dedicated nodes running under a separate maui partition that allows the principal investigator to set and manage their own queues within some reasonable bounds. There is no sharing of unused nodes (and there has not been a free cycle since it was set up!), but we are planning to incorporate preemption into any future shared resource in order to assure timely access to sub-system owners.

5.4 Policy Overview

There are several policy documents either completed or at various stages of review. All users must read and agree to the campus-wide Acceptable Use and Security policies. Last fall, Rice University endorsed a new Core Services Service Level Agreement (SLA) describing IT services that are provided as a matter of course to the entire Rice community. An analogous SLA pertaining to research computing support has already been reviewed by the IT Advisory Committee and awaits further review before being finalized. The Research SLA describes services and associated costs of computer support beyond that provided through the Core SLA.

CITI and Rice's Information technology are working together to create a Shared Research Computing Resource (SRCR – pronounced *ShareCore*) policy framework that will apply to all shared computer resources. This not only provides a framework for granting agencies to use while leveraging resources, but also offers an optimal working model to faculty who are looking to invest as a SRCR partner.

There are also several policies documents that address operational issues and user support. The Change Management Policy outlines the workflow and requirements of making changes to the research computing infrastructure. The queuing policy document provides the user with a high-level description of a given resources queuing policies as well as technical details on how PBSPro/maui implements these policies, including tools to assist users in developing a submission strategy.

At a user group level, the Service Assurance Partnership MOU details the relationship between the RCSG and scientific investigators working on projects through their research group. Designed to increase productivity by increasing the synergy between the RCSG and Rice's research groups, centers, and institutes, it is basically a central repository of information for meeting the specific research computing needs of our faculty partners.

6 Summary

This paper describes the process of procuring, deploying and operating a 3 TeraFLOP Cray XD1 at Rice University. The process involved a competitive bid process with six vendors. The bid process involved an extensive benchmarking request that was designed to stress capability and capacity requests, as well as IO performance for engineering and science computing needs. Of the six vendors that submitted bids only one vendor, Cray, delivered a complete set of results as per our request. The timeframe for these benchmarks (late spring 2005) did not permit large scale benchmarking on Dual Core AMD Opteron. Cray did, however, provide extensive scaling analysis for the different benchmarks where this was relevant and while such numbers may always be questioned the documentation provided for the scaled performance was extensive and backed up with data from limited experiments on sample availability of dual core technology. We were truly impressed by Cray's benchmarking team and our overall experience with the Cray installation team and field support was exemplary.

Based on our current experience we do have a number of recommendations related to the Active Manager (AM) that we feel Cray might want to consider in light of the XD1 or future products where AM might be considered.

Active Manager by default is very intrusive and makes many assumptions it shouldn't, such as forcefully associating queues with hardware partitions.

We believe this software would be more acceptable if it were modularized. The monitoring and system management components need to be separated. The

customer should be able to choose what method of system provisioning to use as well as how to distribute and maintain configurations, networking, etc.

We understand that the current AM model may be good for smaller clusters, but it does not scale well on large shared computing systems.

Active Manager does not provide information comparable to the industry standard, nagios. In fact, Active Manager is not capable of providing historical resource availability information at all. The alarm and fault response features are not as flexible as nagios and the ability to create new "components" to be monitored doesn't seem to be documented or available.

7 Acknowledgments

The authors would like to thank colleagues, users and Cray for working together on the procurement and deployment of the system at Rice University.

8 About the Authors

Jan E. Odegard is the Executive Director of the Computer and Information Technology Institute at Rice University. He can be reached at Rice University, Computer and Information Technology Institute, 6100 Main St., Houston TX 77005, E-Mail: odegard@rice.edu. **B. Kim Andrews** is the Manager of Research Computing with the IT Division at Rice University. He can be reached at Rice University, Information Technology, 6100 Main St., Houston TX 77005, E-Mail: kimba@rice.edu **Franco Bladilo** is a Linux Architect with the IT division at Rice University. He can be reached at Rice University, Information Technology, 6100 Main St., Houston TX 77005, E-Mail: bladilo@rice.edu **Kiran Thyagaraja** is a Linux Integrator with the IT Division at Rice University. He can be reached at Rice University, Information Technology, 6100 Main St., Houston TX 77005, E-Mail: kiran@rice.edu **Roger Moye** is a Linux Cluster Administrator with the IT Division at Rice University. He can be reached at Rice University, Information Technology, 6100 Main St., Houston TX 77005, E-Mail: moye@rice.edu **Keith Schincke** is a Research Computing Support Specialist with the IT Division at Rice University. He can be reached at Rice University, Information Technology, 6100 Main St., Houston TX 77005, E-Mail: kschin@rice.edu