Managing a grid, Part 4: Day-to-day tasks for a grid administrator

Level: Intermediate

David Medinets (david@affy.com), Freelance Writer, Consultant
David A. Cafaro (dac38@georgetown.edu), Computational Researcher, Consultant

23 Jan 2007

Managing a grid involves many elements, from the network and hardware you use to deploy your grid to the security, job management, and job metrics and statistics generated during the execution of your grid, enabling you to more effectively manage the work. In this four-part "Managing a grid" series, we look at key elements of the grid management process, such as identifying hardware and network fundamentals that affect your grid process, and how to use metrics information as a scheduling, prediction, and expansion tool. In this final article, we cover the day-to-day management of a grid.

Introduction

If you've been following along with this series, you should have a good understanding of what's involved in designing, deploying, and monitoring a grid infrastructure. We've looked at what to consider when looking at network and grid hardware and software and how you must look at what services you will be providing and in what environment you will be operating. We've looked at what should be considered during the design phase in implementing security to provide a controlled means of managing who and what has access to your grid resources. And we have looked at what you should be monitoring during the operation of your grid to help you maintain the operational status of your resources.

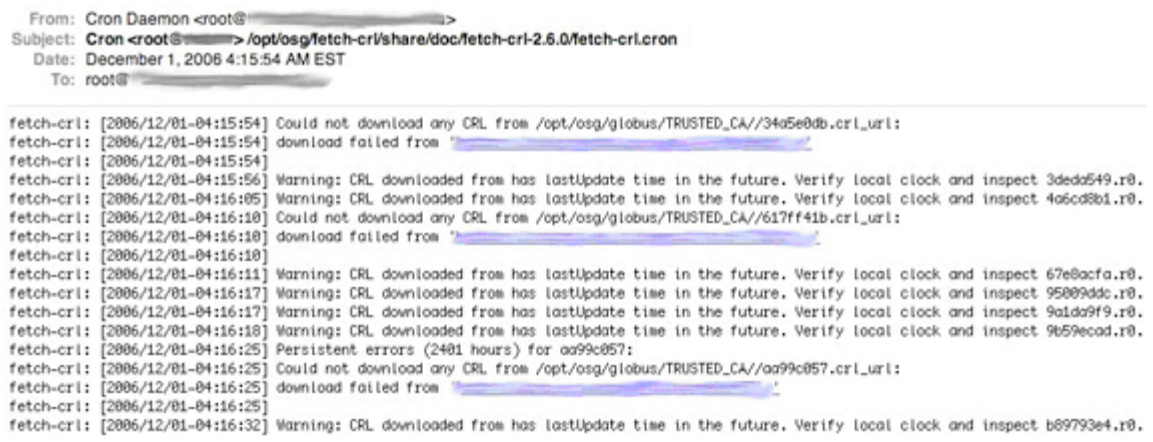
Part 4 will take what we have learned and present what a typical grid administrator's work schedule might look like. We'll look at daily tasks, issue handling, job maintenance, and planning.

Daily tasks

The day-to-day operations of a grid administrator are not much different from that of any good systems administrator, as long as you did your homework. A well-designed grid infrastructure will generally remain stable. The only issues usually result from hardware failures, maintenance updates, and user error (either faulty applications or simple mistakes). Here are some daily issues:

- **Logs and monitoring** -- One of the first and most important things you will do during the day is scan your logs and look at your metrics reports. You may use software to help you scan these items and highlight keywords and issues to make it easier to spot issues. You'll be looking for anything that isn't within the norm, such as errors and warnings in log files and unusual activity in your metrics or lack of activity, which is more likely. The following screenshot shows what a log-based software error might look like.

Figure 1. Error on authentication services



```
   From: Cron Daemon <root@[redacted]>
Subject: Cron <root@[redacted]> /opt/osg/fetch-crl/share/doc/fetch-crl-2.6.0/fetch-crl.cron
   Date: December 1, 2006 4:15:54 AM EST
     To: root@[redacted]

fetch-crl: [2006/12/01-04:15:54] Could not download any CRL from /opt/osg/globus/TRUSTED_CA//34a5e0db.crl_url:
fetch-crl: [2006/12/01-04:15:54] download failed from '[redacted]'
fetch-crl: [2006/12/01-04:15:54]
fetch-crl: [2006/12/01-04:15:56] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect 3deda549.r0.
fetch-crl: [2006/12/01-04:16:05] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect 4a6cd8b1.r0.
fetch-crl: [2006/12/01-04:16:10] Could not download any CRL from /opt/osg/globus/TRUSTED_CA//617ff41b.crl_url:
fetch-crl: [2006/12/01-04:16:10] download failed from '[redacted]'
fetch-crl: [2006/12/01-04:16:10]
fetch-crl: [2006/12/01-04:16:11] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect 67e8acfa.r0.
fetch-crl: [2006/12/01-04:16:17] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect 95009ddc.r0.
fetch-crl: [2006/12/01-04:16:17] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect 9a1da9f9.r0.
fetch-crl: [2006/12/01-04:16:18] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect 9b59ecad.r0.
fetch-crl: [2006/12/01-04:16:25] Persistent errors (2401 hours) for aa99c057:
fetch-crl: [2006/12/01-04:16:25] Could not download any CRL from /opt/osg/globus/TRUSTED_CA//aa99c057.crl_url:
fetch-crl: [2006/12/01-04:16:25] download failed from '[redacted]'
fetch-crl: [2006/12/01-04:16:25]
fetch-crl: [2006/12/01-04:16:32] Warning: CRL downloaded from has lastUpdate time in the future. Verify local clock and inspect b89793e4.r0.
```

- **Education and alerts** -- It's always important to stay current with the technology and tools you're using. It's recommended that you join mailing lists and other forms of communication to get alerts to new developments in the software you're using (see Resources). Often, by the time you notice something wrong in the applications you're using, someone else has noticed the same problem and has started a discussion on it. It may even have been fixed or a workaround discovered. These lists can also alert you to future problems and help you plan for them before they become a real issue.
- **System maintenance** -- Maintenance tasks will often be divided into online and offline tasks. Online tasks are usually simple things like deleting stale temporary files, adding or removing users, or any other task that doesn't interrupt running processes. Offline tasks generally consist of system upgrades that require notification to users that a given system cannot be used during a given time period. This should be scheduled during your normal maintenance windows if possible.
- **Scheduling job requests** -- This depends if you are using an automated scheduler or manual scheduling system. With an automated scheduler, you may be checking that jobs are running and that new jobs are queuing. With manual scheduling, you need to consult your metrics and match job requests with available resources.
- **Planning** -- If everything is going fairly smoothly, one big task you'll face is planning for expansion and upgrades. As with anything in the technology field, once something is used and deemed beneficial, there is a constant call for improvement.

Monitoring logs will often lead to working on issues that present themselves in the logs (system failures, emergency maintenance, security responses). The same can be said of things you learn from mailing lists and RSS feeds. System maintenance may include tasks such as recycling Java™ VMs, restarting hung services such as HTTP, or simply dealing with new grid users. Depending upon the type of jobs and your grid infrastructure, scheduling jobs will most likely be one of the smaller chunks of time out of your day. The following sections look at some of these related tasks in more detail.

System failure and recovery

From time to time, you'll look through your logs or check your metrics or e-mail and discover a system that has failed. This can be a minor annoyance or a major issue, depending on which system has failed. Core components such as nonredundant routers and switches, master nodes on computing resources, storage, and authentication systems can bring grid resources to a screeching halt. When one of these key components fails, you will likely be forced to perform emergency maintenance, taking your resources offline. If you were able to build some redundancy into your architecture, you may not have to take the resources offline for long periods of time or at all. In cases of compute nodes or storage elements failing, it will most likely not bring down all of your resources and can sometimes be dealt with best during regularly scheduled maintenance windows or when usage is light. No hardware failure should be left for long because these resources are part of the redundancy built into any grid system. There are also software failures that need to be addressed quickly. Software failures tend to take an entire resource offline.

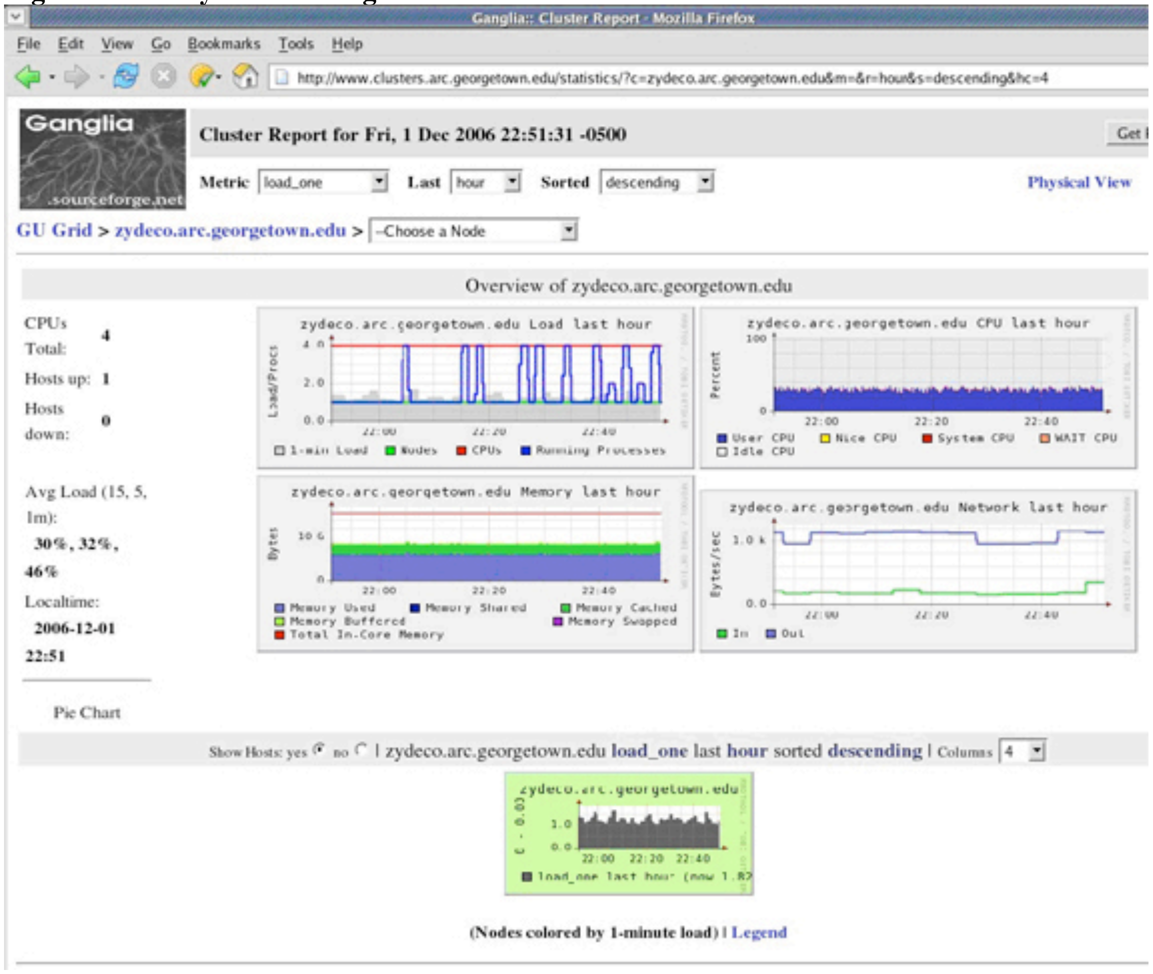
- **Master node failures** -- This is one type of failure that will bring your grid or resources to a standstill and will need to be addressed immediately. Since a master node generally controls all work being done on your grid, a loss of the master node cuts off all scheduling and workflow control. At this point, it's probably easiest to go ahead and bring down the rest of the affected grid while you repair or replace the master node. All compute and storage nodes need to resynchronize once the master node is back online, and this is often easier to do from a cold start. You can avoid some of these issues by having hot standby nodes ready to take the place of a failed master node. You may still have issues with some failed jobs, but new jobs can continue, and jobs that for some reason failed during the switchover can be rerun while the failed master node is rebuilt and restarted.
- **Authentication failures** -- Though not as serious as a master node failure because it won't usually affect running jobs, it is serious in that no new jobs can be submitted and the results of completed jobs can't be retrieved. Most often, this doesn't involve taking down the entire grid or resources and can be dealt with immediately. However, new jobs probably won't start because they can't authenticate, which means they can't allocate resources. Again, backup or redundant authentication servers can reduce or eliminate downtime for new jobs.
- **Network equipment failures** -- Generally speaking, network equipment failure occurs less often than other types of failures. When equipment failure does occur, it can often bring an entire grid to a stop unless enough redundancy is built in. If the network equipment links separate grids within your organization, it is usually not necessary to shut down each grid while the repair is taking place. They will simply resynchronize when the network recovers. If the network equipment provides backbone communications between grid nodes, it may be necessary to bring down the affected grid to allow for a clean restart once the repairs have been made.
- **Application and job failures** -- These are the most common failures. They usually do not require bringing the grid offline, but they need to be addressed immediately before they affect other jobs running on your grid. Most often, the run logs from the jobs or applications will point to areas of interest and you can work with your software developers or application providers to diagnose the issue.
- **Grid node failures** -- This is probably the least disruptive type of failure. Usually, if a master grid node detects a failed grid node, work can be reassigned and the failed node marked. At this point, it's a task of repairing the failed node. There are occasions where a grid node can fail yet still appear to be working to the master grid node. In these circumstances, you will need to look for long-running jobs taking longer than expected or producing results with errors. Again, this is not the usual issue, but something to be aware of.

Spotting trends and issues

As your grid gets used and becomes more popular and more productive, you are bound to notice that your resources have a finite limit. Luckily with grid systems, you don't really have to worry about that finite limit on your current grid as you can easily expand your grid by adding new resources. In some instances, it may just be a matter of modifying where and when jobs are scheduled. The most difficult task becomes keeping track of when and how much you need to expand or how to reschedule. This is where your metrics come into play. By analyzing long-term metrics, you can spot resource usage trends. You should look for which resource types are most popular and when they are most used.

- **When to redistribute work** -- It's important to look at snapshots of overall load, as well as averages of your grid load. This may show your grid at 50-60 percent use, yet it may feel closer to 100 percent. Often, you will notice "bursty" work that occurs regularly all at the same time. Your system may go unused evenings and weekends, or be heavily used for a week, then go dormant the next. Depending on your needs, you can work to better schedule your workloads to better take advantage of these downtimes. Before upgrading hardware, it's good to check and see that you are making the best use of current resources. Figure 2 shows an example of bursty work on the top-left graph. Such a system might benefit from better job scheduling.

Figure 2. Bursty work on a grid resource


Ganglia:: Cluster Report - Mozilla Firefox

http://www.clusters.arc.georgetown.edu/statistics/?c=zydeco.arc.georgetown.edu&m=&r=hour&s=descending&hc=4

Cluster Report for Fri, 1 Dec 2006 22:51:31 -0500

Metric load_one Last hour Sorted descending

GU Grid > zydeco.arc.georgetown.edu > --Choose a Node

Overview of zydeco.arc.georgetown.edu

CPUs Total: 4

Hosts up: 1

Hosts down: 0

Avg Load (15, 5, 1m): 30%, 32%, 46%

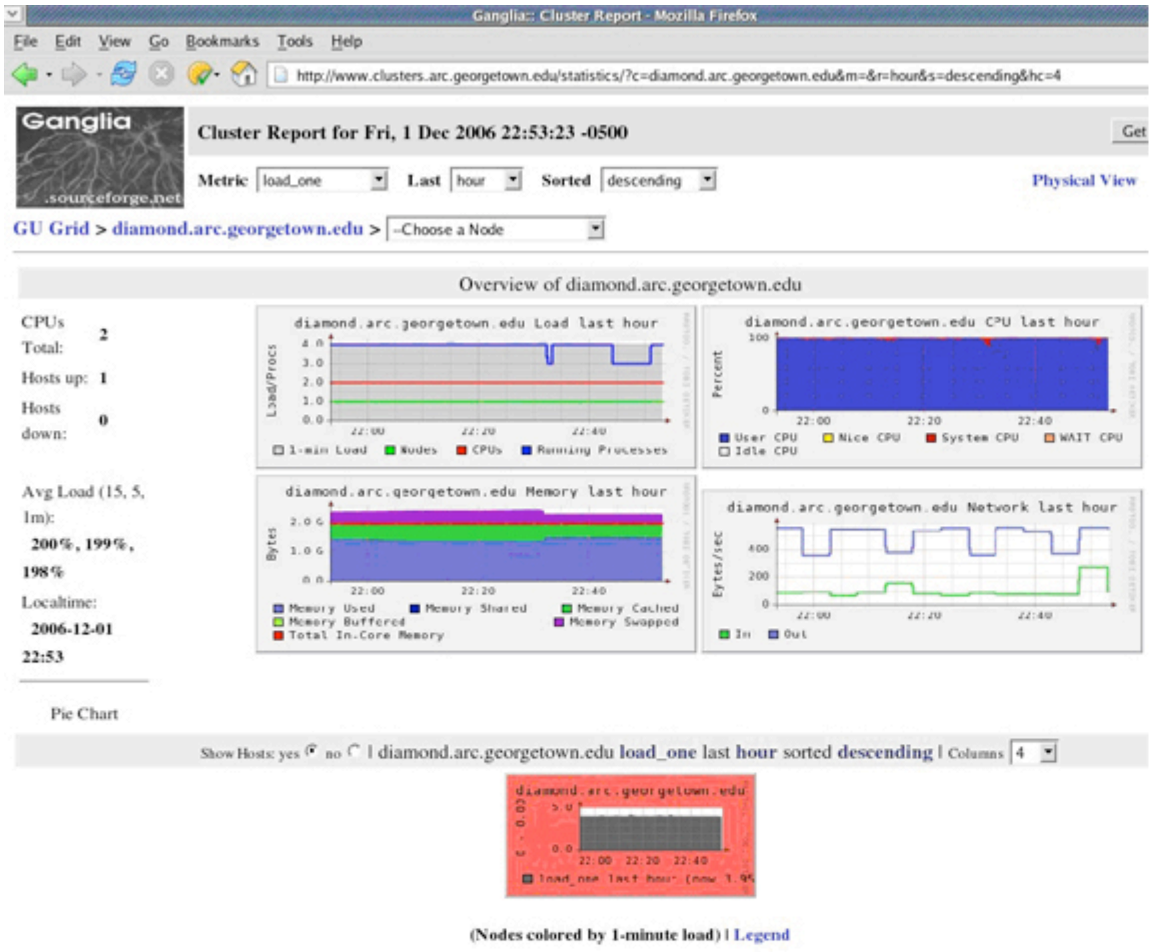
Localtime: 2006-12-01 22:51

Show Hosts: yes [✓] no | zydeco.arc.georgetown.edu load_one last hour sorted descending | Columns 4

(Nodes colored by 1-minute load) | Legend

- **When to plan for expansion** -- In some cases, you see a trend toward a grid with constant 75-percent utilization, or rescheduling is not an option because of the nature of the work. In these cases, you need to decide when an upgrade or expansion is needed. You want to avoid having your system reach 75-percent capacity usage. By looking at your metrics and taking into account your organization's future plans and implementation speed, you can plan to install upgrades and expansions when your system is reaching that mark. This gives you some slack in the case of unforeseen issues and rush work. It's important to plan well in advance by taking advantage of the vast amounts of metrics data your grid can produce for you. Figure 3 gives an example of a grid resources likely in need of an expansion or upgrade. Notice top-left graph that the system is nearly maxed in terms of usage.

Figure 3. Heavily loaded system


Ganglia:: Cluster Report - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Ganglia
sourceforge.net
Get
Physical View
diamond.arc.georgetown.edu Load last hour
Load/Procs
1-min Load
Nodes
CPUs
Running Processes
diamond.arc.georgetown.edu CPU last hour
Percent
User CPU
Nice CPU
System CPU
WAIT CPU
Idle CPU
diamond.arc.georgetown.edu Memory last hour
Bytes
Memory Used
Memory Shared
Memory Cached
Memory Buffered
Memory Swapped
Total In-Core Memory
diamond.arc.georgetown.edu Network last hour
Bytes/sec
In
Out
Pie Chart
diamond.arc.georgetown.edu
load_one last hour (now 1.95)

http://www.clusters.arc.georgetown.edu/statistics/?c=diamond.arc.georgetown.edu&m=&r=hour&s=descending&hc=4

Cluster Report for Fri, 1 Dec 2006 22:53:23 -0500

Metric load_one Last hour Sorted descending

GU Grid > diamond.arc.georgetown.edu > --Choose a Node

Overview of diamond.arc.georgetown.edu

CPUs Total: 2

Hosts up: 1

Hosts down: 0

Avg Load (15, 5, 1m): 200%, 199%, 198%

Localtime: 2006-12-01 22:53

Show Hosts: yes [✓] no | diamond.arc.georgetown.edu load_one last hour sorted descending | Columns 4

(Nodes colored by 1-minute load) | Legend

Updates and upgrades

The updating of a grid is vital, but you can do it in different ways. A complete shutdown/startup will take your grid offline, but will ensure that all machines are upgraded simultaneously. A phased update takes blocks of machines out of action and upgrades them. This approach may reduce the effectiveness of your grid, but doesn't take it out of action and is, therefore, more appealing in situations where a single update phase could mean shutting down the grid for days or weeks.

- **Scheduling** -- Ideally, you want to schedule when you will have the smallest impact on your users. You need to be sure to take a couple of things into account, such as access to external assistance in case of unforeseen problems and extra time in case of work overruns. Though the least impact is often in the evening or on weekends, these are also the times when getting outside assistance can be difficult. It's usually easier and safer to schedule on a light-loaded work day by providing users with warning of the scheduled downtime. You should take into account the amount of disruption, the chances of unforeseen problems, and access to outside assistance if needed when deciding on when work should take place.
- **Full shutdown/update** -- These are usually best done during low-use periods, as you will be taking your grid offline. It is also necessary to make sure all parties are aware well in advance. It will make it easier to perform the updates and upgrades as you will be able to verify each component as it comes back online. This often leads to the longest downtimes, so it should generally be done

only during major hardware and software upgrades or because of serious security issues.

- **Staged/phases shutdown/update** -- This is probably one of the favored ways of performing updates and upgrades. This allows you to take certain components down while leaving the rest up and running. This provides most of the benefits of a full shutdown without causing a complete loss of service. The downside is that it does not work as well for time-sensitive updates, such as security issues or critical software flaws that may threaten your entire grid infrastructure.
- **Live expansion** -- Adding grid nodes or grid components can sometimes be done without the need for a shutdown of any type. Adding more compute elements or another grid storage system can be as simple as bringing the equipment online and updating the configurations on the other master grid nodes to alert them of the new resources.

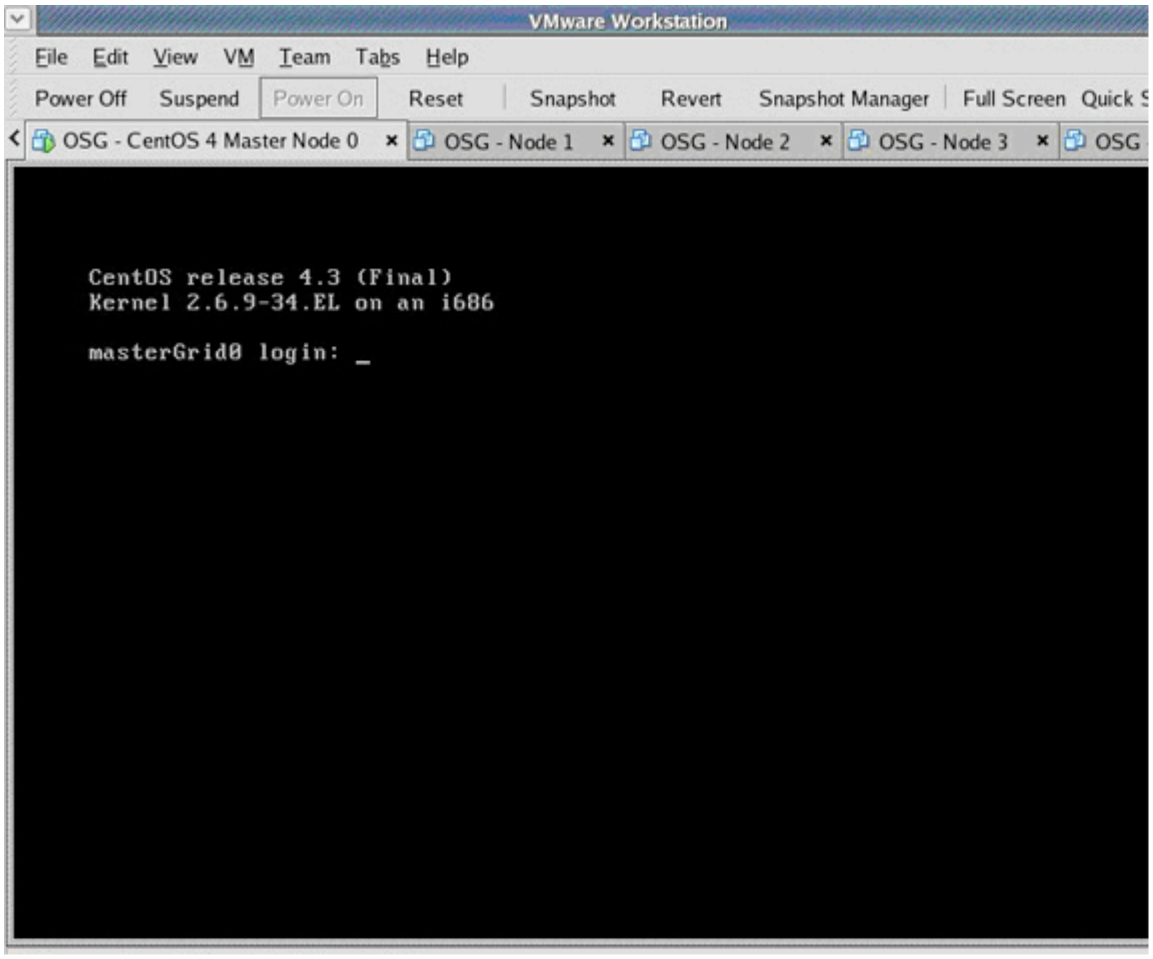
With any upgrades or expansions, it's important to test and prepare well in advance. Scheduling should always include time to do tests with nonproduction equipment and to verify that operating procedures will work during the real event.

Testing and staging

Here's a quick overview of some techniques for providing testing and staging services so updates and changes to software can be handled and organized before being applied to a grid. Virtual machines are a great tool here, but even a small 3- or 4-node hardware setup can be useful.

- **Hardware testing grids** -- One way to test is to create a small sample grid in your test lab. This can consist of a few samples of your deployed or planned-to-be-deployed equipment. This provides a great way to test your architectural design, as well as how different pieces of hardware will work together. It can also provide a good platform for software testing as it will most closely resemble your actual grid hardware. Disadvantages include expense, space, and software failures and system restores/images.
- **Virtual testing grids** -- The use of virtualized servers and networks can provide a great environment for testing software for your grid before deployment. Many virtualization software packages provide easy access to saved system states, easy rollback of system images, and flexibility in dealing with multiple configurations. A primary disadvantage is that you may be limited in your networking design and performance testing. Figure 4 shows an example of a grid test environment running on a virtualized environment.

Figure 4. Virtual test grid


VMware Workstation
File Edit View VM Team Tabs Help
Power Off Suspend Power On Reset Snapshot Revert Snapshot Manager Full Screen Quick S
OSG - CentOS 4 Master Node 0
OSG - Node 1
OSG - Node 2
OSG - Node 3
OSG

```
CentOS release 4.3 (Final)
Kernel 2.6.9-34.EL on an i686

masterGrid0 login: _
```

- **Staging** -- New equipment should be brought up and tested in a testing environment before being added to the live grid infrastructure. This can simply be turning the system up in place while having the network connection to the production grid unplugged. Make sure you are given a chance to see how the system behaves before it's in a position to interact with the rest of your grid. You can also test how it may interact with your production grid by testing it against your testing grid beforehand.

Summary

You should now have an excellent idea of what is involved in keeping a grid operating safely and smoothly. We conclude the "Managing a grid" series, having discussed:

Share this...

Digg this story

Post to del.icio.us

Slashdot it!

- System failure and recovery -- Which surfaces important issues to think about before a widespread failure happens to you.
- Trends and issues -- Every system has peaks and valleys. Know the trends so you can prepare your grid.
- Updates and upgrades -- As you spot trends, you'll be able to update/upgrade your grid accordingly to prepare for the worst and to ensure that you stay below the 75-percent utilization mark.
- Testing and staging -- It's wise to test your grid in a sandbox-like environment that doesn't damage

sensitive data. If it fails, at least your customers won't be affected. You can use your skills to fix and update the grid, and bring it live when it's ready.

Resources

Learn

- The TeraGrid is a National Science Foundation (NSF)-funded project to provide an integrated computational and data infrastructure for research scientists within the United States. Read a three-part series titled "Lessons learned from the TeraGrid."

- Read "Building a unified grid," a four-part series of articles describing the grid-based system architecture developed in the Telescience Project at the National Center for Microscopy and Imaging Research.

- For documents that provide information and specifications to developers and others involved with grid computing, check out the Open Grid Forum (OGF) Document Series.

- The screenshots in this tutorial were provided by Advanced Research Computing (ARC) at Georgetown University.

- The Globus Alliance is a community of organizations and individuals developing fundamental technologies behind the grid, which lets people share computing power, databases, instruments, and other online tools securely across corporate, institutional, and geographic boundaries without sacrificing local autonomy.

- The Gridbus project is engaged in the design and development of open source cluster and grid middleware technologies for service-oriented computing. Gridbus emphasizes the end-to-end quality of services driven by computational economy at various levels -- clusters, peer-to-peer networks, and the grid -- for the management of distributed computational, data, and application services.

- Check out GridsWatch for grid-related news, links to articles and tutorials, interviews, and training.

- International Science Grid This Week provides information and news about people and projects involved in grid computing and the science that relies on them. It's supported by the U.S. Department of Energy, National Science Foundation and the European Commission's Information Society.

- The Globus Consortium is a nonprofit organization formed by global computing leaders who support the Globus Toolkit, the de-facto standard for open source grid computing infrastructure.

- "Comparing traditional grids with high-performance computing" looks at network topologies, including specialized network hardware and mesh structures, traditionally used in an HPC environment that are being employed in grids, and how commodity systems are replacing the need for networking and infrastructure requirements.

- "Grid computing -- moving to a standardized platform" discusses how simplifying your deployment environment can make your grid easier to manage and develop.

- "Grid in action: Managing the resource managers" looks at resource management and how it's used to distribute and manage workload in a grid.

- "Building a grid using Web services standards" builds an entire resource-sharing grid for storing movies based on the principles of the Web services standards in six parts.

- "You've got grid" demonstrates how easy it is to build a grid using existing tools and environments by building a work distribution system based on email.
- Need a primer? See developerWorks' "New to grid computing."
- Find more grid resources in "Recommended reading list for grid developers."
- Browse all of the Grid computing content on developerWorks.
- Visit the developerWorks Grid computing zone for information to help you develop with grid technologies.
- Stay current with developerWorks technical events and webcasts.
- Visit Safari Books Online for a wealth of resources for grid computing technologies.
- developerWorks podcasts include interesting interviews and discussions for software developers.

Get products and technologies

- Ganglia is an open source project providing a free scalable distributed monitoring system for high-performance computing systems, such as clusters and grids.
- Innovate your next open source development project with IBM trial software, available for download or on DVD.

Discuss

- Check out the Grid computing forums.
- Get involved in the developerWorks community by participating in developerWorks blogs.

About the authors

David Medinets is an independent software consultant specializing in Java technology, SQL, and XML. In 1999, he designed and coded most of ToysRs.com. Since then, he has been involved in a variety of startups and established companies in many industries. For Cordiem, a versioned airline parts database was designed and created. For WWRE, an XML validation and processing engine was architected and developed. He also has written three books and several articles covering technical topics.

David A. Cafaro is currently employed at Georgetown University's Advanced Research Computing (ARC) group, where he supports computational research needs through the use of Linux and open source technologies and is a Red Hat Certified Engineer. In addition, he sits on the Program Committee for LinuxWorld Expo and Summit ,where he helps develop track content and technology focus. He has worked as a security analyst at Tresys Technology with a focus on SELinux policy work. He is active in the open source and Linux communities as a member of the Tux.org Board of Directors.