Grid Systems Deployment & Management using Rocks

Federico Sacerdoti[1], Sandeep Chandra[1], Karan Bhatia[1]

[1]San Diego Supercomputer Center, 9500 Gilman Drive, University of California, San Diego, La Jolla, CA 92093

{fds, chandras, karan}@sdsc.edu

Abstract

Computational clusters have become the dominant platform for a wide range of scientific disciplines. Cluster management software is available that allows administrators to specify and install a common software stack on all cluster nodes, and enable centralized control and diagnostics of components with minimal effort. While grid deployments have similar management requirements, they have faced a lack of available tools to address their needs. This paper describes the systems management solutions developed for one such grid deployment, the GEON Grid, a domain-specific grid of clusters for geological research. Machines in GEON grid are geographically dispersed, and must be connected with wide-area networks. GEON provides a standardized software stack across all sites, while allowing controlled local customization. This situation gives rise to a set of requirements that are difficult to satisfy with existing cluster tools. In this paper we describe extensions to the Rocks cluster distribution to satisfy several key goals of the GEON Grid, and show how these wide-area cluster integration extensions satisfy the most important of these goals.

1. Introduction

Computational clusters have become the dominant platform for a wide range of scientific disciplines. Due to this pressure, cluster management software has risen to the challenge; cluster tools exist that specify a common configuration base, install a software stack on all nodes, enable centralized control of components, and provide diagnostics for failure reporting, all with minimal effort. While cluster management toolkits have been successfully applied to large-scale clusters operating in tightly coupled LAN environments [1,2], current grid deployments with similar management requirements have faced a lack of available tools. These grids seek to offer a common operating environment for a community or scientific domain, and typically involve a diverse set of resources operating in a geographically dispersed environment. Examples of such grid deployments include GEON [3], BIRN [4], and GriPhyn [5] although many others exist [6, 7, 8].

We present grid design from the perspective of GEON, although its similarity to other grid efforts makes our results applicable to other projects. Figure 1 shows the GEON grid architecture. A set of physically distributed clusters are located within the administrative domain of each of sixteen participating sites. These clusters may have zero or more compute nodes and may consist of different hardware architectures. The operation of this *virtual organization* [9] requires the machines to run a common software stack that enables interoperability with other GEON resources. We restrict our attention to computational hardware resources. In addition to being geographically distributed, the clusters at each partner site build upon the common software base to provide site-specific applications and services. The challenge, therefore, is to manage the distributed set of

hardware resources, physically distributed at partner institutions and connected over the commodity Internet, in a way that minimizes system administration costs while achieving interoperability and local site autonomy. Although local cluster performance is important, the main objective of the GEON systems component is to efficiently manage the grid in the face of system upgrades, security patches, and new software components. A central tenant of the design is to achieve this level of management with minimum administration.

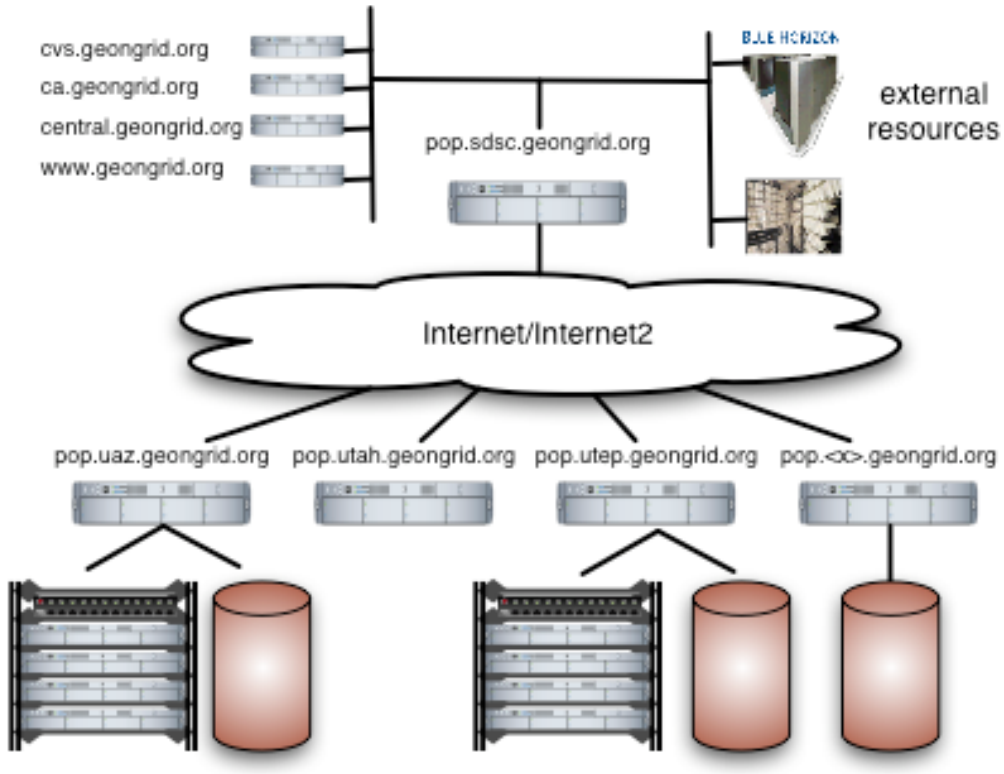

Figure 1: GEON grid consists of distributed resources, both internal and external, interoperating and connected over the commodity network and in some cases on Internet2.

We use the Rocks cluster distribution as a starting point for the GEON effort. Although it did not initially provide the functionality to manage a grid of machines, Rocks has been

successfully used to manage large clusters consisting of more than 500 nodes. With our direction, the Rocks toolkit has been modified to support wide-area deployment and management of systems in a fashion appropriate for GEON. In this paper we present our requirements for a grid management toolkit, and show how the new wide-area cluster initialization capability in Rocks satisfies a majority of these requirements. We hope the solutions explored for the GEON project can be directly applied to other grid deployments in addition to our own.

Section 2 describes the overall Rocks architecture and summarizes its capabilities along with other popular cluster management software. Section 3 discusses the specific requirements for wide-area grid deployments and management using GEON as an example. Section 4 describes the architecture and implementation changes made to Rocks to support wide-area grid management. Section 5 discusses some initial performance measurements and other feedback from using this system for real systems deployment and management in GEON and identifies open issues and future directions. Section 6 summarizes the paper.

2. Rocks Cluster Management

High-performance computing clusters have become the dominant computing platform for a wide range of scientific disciplines. Yet straightforward software installation, management, and monitoring for large-scale clusters has been a consistent and nagging problem for non-cluster experts. The free Rocks cluster distribution takes a fresh perspective on cluster installation and management to dramatically simplify version tracking, cluster management, and integration.

The toolkit centers around a Linux distribution based on the Red Hat Enterprise line, and includes work from many popular cluster and grid specific projects. Additionally, Rocks allows end-users to add their own software via a mechanism called *Rolls*. Rolls are a collection of packages and configuration details that modularly plug into the base Rocks distribution. In this paper, for example, we demonstrate injecting software into a domain-specific Grid via a GEON roll. Strong adherence to widely used tools allows Rocks to move with the rapid pace of Linux development. The latest release of Rocks, version 3.2.0, supports commodity architectures including x86, IA64 and x86_64 Opteron.

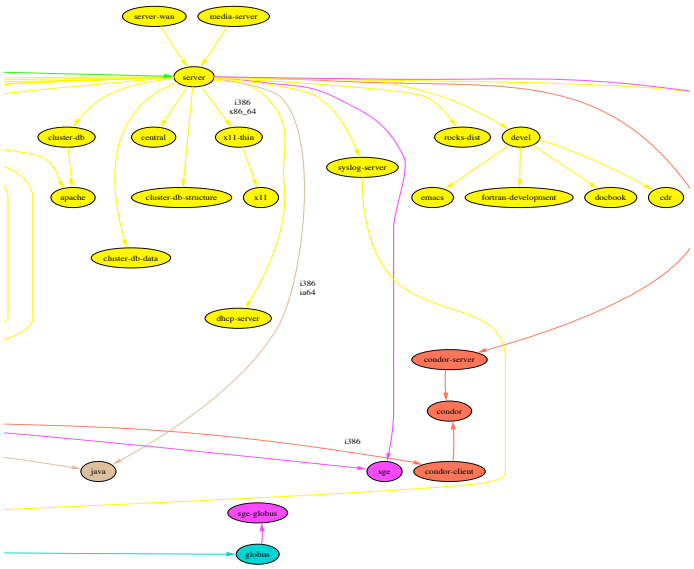

Figure 2: Rocks and Rolls. A node in the graph specifies a unit of packages and configuration; a graph traversal defines software for a cluster appliance. Yellow nodes are Rocks base, while colored nodes belong to various rolls.

2.1. Architecture

Figure 3 shows a traditional architecture used for high-performance computing clusters. This design was pioneered by the Network of Workstations [10], and popularized by the Beowulf project [11]. In this method the cluster is composed of standard high-volume

servers, an Ethernet network and an optional off-the-shelf performance interconnect (e.g., Gigabit Ethernet or Myrinet). The Rocks cluster architecture favors high-volume components that lend themselves to reliable systems by making failed hardware easy and inexpensive to replace.

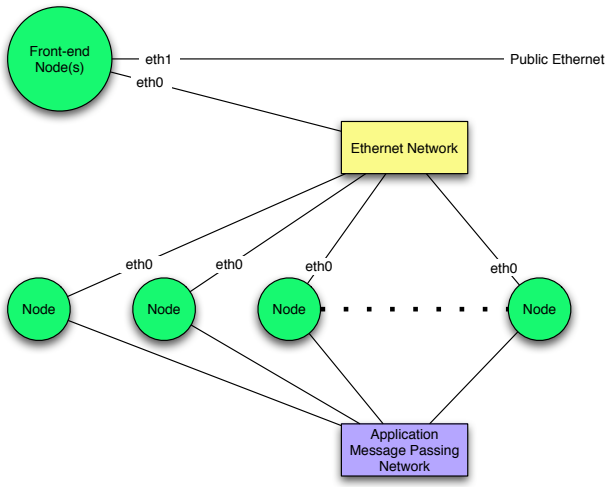

Figure 3: Rocks Cluster Hardware architecture. The Frontend node acts as a firewall and gateway between the private internal networks and the public Internet.

Rocks Frontend nodes are installed with the base distribution and any desired rolls. Frontend nodes serve as login and compile hosts for users. Compute nodes typically comprise the rest of the cluster and function as execution nodes. Compute nodes and other cluster *appliances* receive their software footprint from the frontend. Installation is a strong suit of Rocks; a single frontend on modern hardware can install over 100 compute nodes in parallel, a process taking only several minutes. Typically cluster nodes install automatically, using PXE to obtain a boot kernel from the frontend.

One of the key ingredients of Rocks is a robust mechanism to produce customized Linux distributions for a particular domain. When we build a cluster frontend with the GEON roll, all compute nodes will install a set of Geo specific software. This mechanism allows

us to easily inject domain-specific software into the Rocks integration system, enabling a Geo-specific grid of clusters. Specialized CDs can be generated from this custom distribution, which behave identically to those from RedHat. More importantly to this paper, the custom distribution may be transmitted to other cluster frontends over a standard wide-area network such as the Internet.

By leveraging the standard RedHat Anaconda installation technology, Rocks abstracts many hardware differences and automatically detects the correct configuration and hardware modules to load for each node (e.g., disk subsystem type: SCSI, IDE, integrated RAID adapter; Ethernet interfaces, etc). Although its focus is flexible and rapid system configuration (and re-configuration), the steady-state behavior of Rocks has a look and feel much like any other commodity cluster containing de-facto cluster standard services.

2.2. Related Work

We chose the Rocks cluster distribution to power the GEON scientific grid based on its fitness to our requirements. However several attractive clustering solutions exist, both in open source and commercial form.

2.2.1 SystemImager

SystemImager [12] performs automated installs, updates, and software and data distribution, to a network of Linux machines. It supports each hardware platform by storing a unique image of the directory structure for each hardware type. Cluster node configuration for a specific hardware configuration is managed by first installing one node in the cluster, configuring the software by hand, then taking a snapshot of the node (i.e., creating a master node). It is useful in environments with large numbers of identical

machines; any new hardware type in the cluster, however, requires its own full snapshot. Since there is no mechanism for sharing portions of the snapshots, SystemImager can degenerate into a collection of images, each with a large set of duplicated files. The concept of *data normalization* from database theory tells us this situation is particularly difficult to maintain: any change to the system must be repeated for each image copy, and great care must be exercised to avoid inconsistencies between images.

2.2.2 Local Configuration System (LCFG)

The LCFG project [2] has a similar installation philosophy to Rocks. It provides a configuration language and a central repository of configuration specifications, which are analogous to the Rocks kickstart nodes. The specifications can be combined to install and configure individual Unix machines over a network. Changes to the central specifications automatically trigger corresponding changes in the configuration of managed nodes. This incremental update is a feature missing from Rocks, but the lack of a packaged solution and the use of a homegrown installer and hardware detector make LCFG unsuitable for our purposes. The LCFG system is used in diverse configurations where even the Unix flavor is heterogeneous.

2.2.3 Scyld Beowulf

Scyld Beowulf [13] is a commercial, single-system-image cluster operating system. In contrast to SystemImager, LCFG, and Rocks, processes on a scyld cluster see a single PID space for all running processes. While this feature simplifies cluster operations, it relies on a heavily modified Linux kernel and GNU C library. On Scyld clusters, configuration is pushed to compute nodes by a Scyld-developed program run on the

frontend node. Scyld system has limited support for heterogeneous nodes, and does not support new hardware as readily as a distribution-only effort such as RedHat. Because of the deep changes required by the system, the Scyld company sits in the critical path of many bug and security fixes. These fundamental changes require Scyld to take on many (but not all) duties of the distribution provider.

2.2.4 YUM

The Yellow dog Updater, Modified (Yum) [14] is an automatic updater and package installer/remover for RPM-based distributions such as RedHat. While not a clustering solution, YUM can be used to keep a set of machines up to date with a central server. YUM automatically computes package dependencies, and essentially serves as an rsync facility at a package level.

YUM does not address the initial software installation, however, nor does it probe cluster hardware. In addition, it does not guarantee that updated packages will not disrupt currently running jobs, a matter of some significance for long-running scientific tasks. The updater is most suitable for users to keep their existing RPM-based systems up to date in an interactive fashion.

3. System Management Requirements

The GEON project is tasked with building the next generation cyberInfrastructure for the geosciences community. The GEON grid infrastructure is a *services-based* infrastructure providing decentralized data federation capabilities across heterogeneous databases. At the physical resource layer, the GEON project a wide-area grid, called the GEONgrid, with resources situated at each of the 16 partner sites (see Figure 1). At minimum, each

partner site runs a GEON pop node that acts as a point-of-presence and, optionally, coordinates additional machines used for providing large data capability or small-scale computation capability. The pop node at a particular partner site provides a standard set of services for its users. The set of services includes core system management services for determining resource availability and use, data management services for replication and caching of data across the grid, and high-level application services for users conducting data exploration, visualization and simulation. In order to ensure interoperability between nodes at different sites, SDSC provides a comprehensive standardized software stack definition that includes the core operating system, currently Linux, higher-level development environments such as Web and Grid Service libraries, and end user environments (portals). The software stack definition includes the necessary security information to link nodes to the GEON grid security infrastructure.

3.1. Consistency

The first major challenge for systems management of the GEON grid is to maintain a consistent software stack on all machines, currently 40 in total. These hosts are physically distributed among various administrative domains at different sites and connected through the commodity Internet[1]. Uniformity of the software stack is required to ensure interoperability between services running on different sites in the grid. The GEONgrid uses the NMI grid software release [15] as the base of its grid software stack. In addition, we provide higher-level software such as the portal environments, various web services libraries, an OGSI grid service container and libraries, and GEON-specific services. In order to deal with the complexity of so many interoperating components, we have chosen

[1] Few sites are connected using the higher bandwidth Internet2 backbone

to integrate and test the system as a whole unit, which when verified is pushed out to all nodes in the GEON grid.

3.2. Controlled Customization

The second major challenge of the project is to manage the constituent machines to allow controlled customization of the machines at each of the partner sites. There are three reasons for local customization: First, each site may have specific system requirements for its *configuration*. For example, software and security patches, firewall integrity, and other management tools specific to the site must be accommodated. Each partner site may also configure the GEON machines in such a way to leverage additional resources it may have. SDSC, for example, will provide gateways to large compute clusters (TeraGrid [16] and Datastar) and data archives (HPSS). Second, the partner sites may deploy additional grid or web services *applications* on the GEON machines beyond those built into the software stack. These services must be persistent across reinstallations and system updates. Third, partner sites may deploy *datasets* into the GEON network by hosting them on GEON machines. These datasets must also be preserved across software stack updates.

Unconstrained customization, however, leads to software incompatibilities that require significant administration overhead. Therefore the needs of customization must be balanced with the need for efficient system management.

3.3. Requirements

The following are specific requirements we have determined for the GEON deployment, which we believe are similar to other grid systems:

1. Centralized software stack definition. GEON central node will define the base software stack used to instantiate all other nodes.

2. The ability to push the software stack to GEON pop nodes over the Internet with little or no administration.

3. Enable local site autonomy by defining acceptable node types for compute, data, and site-specific customized nodes.

4. Ability to add site-specific constraints. Allow customized software with durable site-specific environment and settings. These software components should survive the process of upgrading or reinstalling the base distribution.

5. Update software and security patches. Use INCA framework [17] to monitor software versions of important middleware components. The ability to identify and incorporate changes to the base software stack.

6. Incremental and hierarchical base distribution updates. Updates to the pop frontend nodes must be automatically pushed to the compute and data nodes at the site.

7. Remotely update critical patches and the software stack. At the same time sites with varying administrative requirements and policies should be able to add additional rules to the basic update mechanism.

8. Nodes or clusters that join the grid should integrate easily and be immediately consistent with the base grid software stack.

While keeping in mind that the sites own their local resources and have full control of them, the GEON system must provide a robust, basic level of systems management that can be extended.

4. Wide Area Kickstart

To address the primary requirement of the GEON grid, a low management cost for grid-wide software installation and updates, we extended the Rocks distribution to perform full cluster installations over wide area networks. While compute nodes in Rocks always employed the LAN to install software from the frontend machine, the frontend itself had to be integrated with CD or DVD media. This strategy, while appropriate for cluster instantiations of basic Rocks software, is insufficiently flexible for the dynamic needs of the GEON grid. Specifically, we considered affecting grid-wide software with mailed disk media unacceptable.

4.1. Central

The wide area cluster integration involves a *Central server* that holds the software stack. Frontend pop nodes of the grid obtain a full Rocks cluster distribution from the central, this distribution is suitable to install local compute, data, and customized nodes. Since the software stack defines the whole of the base system, and because security patches and feature enhancements are common during updates, *any* part of the stack may be changed. The challenge is to retrieve all components necessary for integration from central over the network, including the installation environment itself. We require a small static bootstrap environment for the first pop initialization, which contains only network drivers

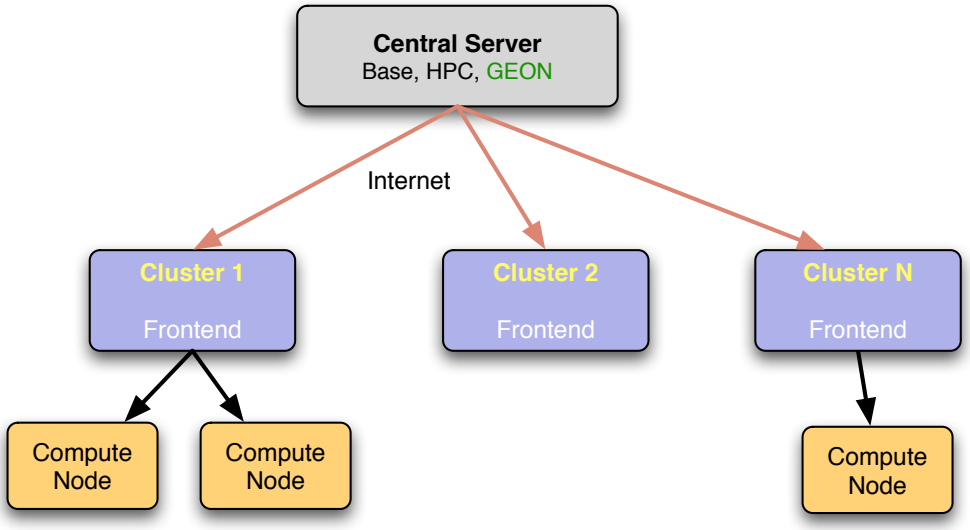

Figure 4: Wide Area Cluster Integration. Central server provides Rocks base and the GEON Roll to frontend cluster nodes in the Geongrid. Frontends then update the software on their respective compute nodes. Frontends obtain their software over wide area networks such as the Internet, while compute nodes install over a local LAN.

and hardware-probing features and fits onto a business card CD. This bootstrap environment is stored on the node for use in the upgrade process.

Figure 4 shows the wide-area kickstart architecture. A central server holds the base Rocks distribution, including the Linux kernel and installation environment, along with standard rolls such as the HPC (high performance computing), and the domain-specific GEON roll. We initialize GEON frontend pops over the wide-area Internet from this central. The software disbursement methodology is optimized for unreliable networks, and possesses

greater awareness and tolerance for failure than the local-area Rocks install process for between frontend and compute nodes. The frontend pop locally integrates the Rocks base and the various Rolls selected from the central server, and the combined software stack is rebound into a new Linux distribution. The pop then initializes cluster nodes with its combined distribution. If the pop is current it is easy to see any new compute, data, or

specialized nodes joining the grid immediately posses a consistent version of the base grid stack.

4.2. Controlled Customization

With the ability to bind the base and third party rolls into a distribution for local consumption comes the possibility of obtaining components from many sources. Figure 5 shows integration from several central servers. This feature allows a type of multiple inheritance, where a frontend's identity is composed from several sources. This inheritance model allows GEON's goal of controlled local customization. A site can maintain a central server with locally developed rolls for its use. However, rolls are highly structured combinations of software and configuration. Their framework allows quality control and verification of their constituent packages, to a level not available to traditional software-testing techniques [18]. With rolls, GEON supports controlled site customization suitable for complete software components, and allow these components to be easily incorporated into the base software stack.

However by the requirements, sites must be capable of affecting local configuration changes as well. The Rocks method of distributing updates reinstalls the root partition of the system. While this provides security assurances and software verification advantages, it requires all service configuration be relegated to other partitions. The GEON system places all local configuration and user environments on non-root partitions, which are durable across updates. No facility exists in the Rocks system to distribute incremental package updates, a shortcoming which is left to future work.

4.3. Multiple Architectures

Figure 5 illustrates the multiple-architecture aspects of the Rocks system as well, which extend to the wide-area kickstart facility. Contained within the bootstrap environment is a local architecture probe, which alerts the central or cluster frontend what architecture it is. This classification allows heterogeneous clusters to ensure each node runs most native software stack. It also requires frontend nodes to posses a distribution for each architecture they expect to encounter in their local cluster. For this reason, the GEON central provides the base distribution and rolls for several hardware types, notably x86 (Pentium, Athlon), ia64 (Itaniaum), and x86_64 (Opteron). The GEON central forms the repository of the base Grid DNA, with local central servers providing additional software resources.

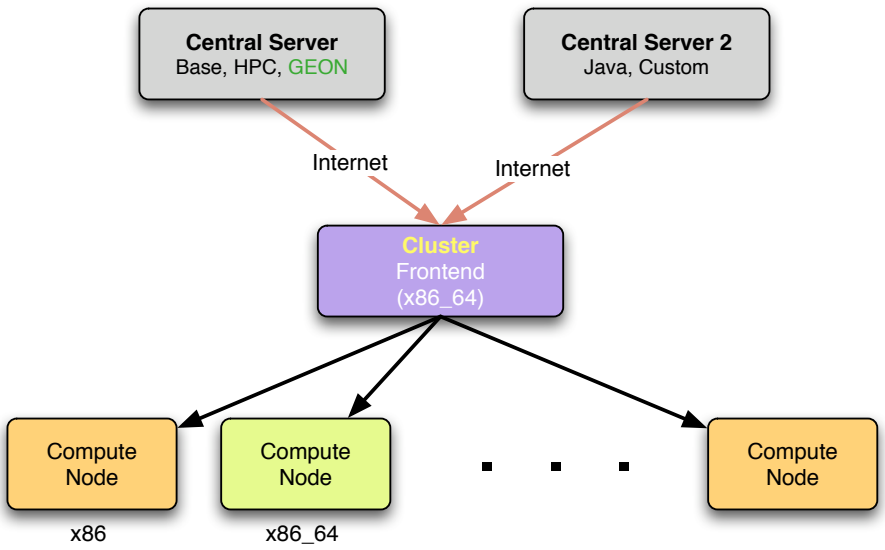

Figure 5: Multiple Central servers. A frontend may be composed of rolls from several sources, allowing course-grained local site customization suitable for large software components. Clusters can have heterogeneous hardware; base distribution and rolls are available for x86, ia64, and x86_64. Nodes automatically negotiate for the most native software available.

The Rocks wide-area cluster integration facility enables GEON to push its software stack to pop nodes over the Internet in an efficient manner with low administration overhead. Site-specific additions and customizations of software components are standardized with the roll structure, and are implemented with site-local central servers. Finally, customizations are maintained across software stack updates by the GEON system structure. The proceeding section provides an evaluation of the process and function of this wide-area integration capability.

5. Experience

To verify the effectiveness of the Rocks wide area kickstart technique we performed experiments in various relevant settings. In this section we show the wide-area cluster integration facility of Rocks satisfies our major management goals of a common software stack that is centrally disbursed to sites, a low-administration method of updates and patches, and the ability for local site customization. We present three experiments, each which model an important use case of our grid.

The objective of these trials is to measure the time needed to retrieve the base and Roll distributions over the public Internet, and to observe any failures and recovery during the process. In addition we wish to characterize the amount of administrative labor involved in the configuration before the installation process begins, and how much configuration is necessary for a local site to add their customized software Roll at the time of installation. Finally, we verify the consistency of the configuration between the base distribution and software stack installation on the remote site. The time taken to setup the node once it

retrieved the distribution is dependent on the hardware configuration of the node (e.g. processing speed, etc) and is not of great significance to the experiment.

5.1. Experimental Setup

All experiments involve a central server node and a frontend pop machine. The central node is a dual Pentium 4 server running Rocks 3.2.0 beta software with Linux kernel 2.4.21. Since all frontend nodes can function as central servers, we setup the as a Rocks frontend with modified access control. Software disbursement occurs using standard HTTP on port 80, access to which has been opened to our frontend pop node using Apache access control lists. The central server is connected to the network via copper-based Gigabit Ethernet capable of 1000mbps.

For the first experiment we kickstart a cluster frontend node from the central server located on the same local area network (LAN). A Gigabit Ethernet link connects the two machines. The purpose of this experiment is to verify operational correctness and ascertain the typical management overhead necessary for a cluster install. Note that once the frontend node has been initialized, standard and proven Rocks techniques can be used to install local compute nodes. Therefore we bias our report towards the integration of the cluster frontend nodes, which serve as pop nodes of the GEON grid.

In the second experiment we stress the system slightly by kickstarting a cluster frontend over the wide area network (WAN) in the same organizational domain. We define wide area in this case as a two network endpoints connected with one more IP routers between them. This scenario corresponds to obtaining a roll from a local site central server for the purposes of customization.

Finally, we initialize a frontend pop over the WAN from a central server in a different organizational domain. This experiment simulates the process of distributing the common Geon software stack to pop nodes on the grid. We expect to see the effects of increased packet loss and TCP congestion control when operating over this longer haul network.

The LAN testing was performed at SDSC between nodes on the same level 2 (switching) network segment. The WAN experiment within the same organizational domain was performed at SDSC using the central server from the first experiment. The frontend has a 100 mbps network link and therefore limits the bandwidth of this test. The third experiment we conduct using the central server at SDSC and a frontend node at Virginia Tech, roughly 4000 kilometers away. The maximum bandwidth between the sites is ~100mbps over the standard Internet link. The Rocks distribution is close to 660 MB, the HPC roll is 338 MB and the GEON roll is close to 98 MB. A series of 10 runs were conducted in each of these experiments and the results in Table 1 indicate the average time in each case.

Kickstart Type	*Rocks Base*	*Rolls (HPC+GEON)*
Local Area Network (Single level 2 segment)	65s	8s
Wide Area Network (Same Organization)	390s (6.5min)	25s
Wide Area Network (Different Organization)	2700s (45 min)	360s (6 min)

Table 1: Time (in seconds) to retrieve the Rocks base and Roll distributions over different types of network environments. Values represent average of several trials.

5.2. Discussion

Our observations show that we were successfully able to satisfy our most important requirements. When creating a node within the local area network, the distribution was retrieved in the shortest time as a result of the high bandwidth, and the node was consistent in software and configuration with the central server. The amount of personnel involvement was minimum as we would expect from the Rocks toolkit. Our experience with the wide area kickstart were the same, all the metrics were similar except the time taken to retrieve the distribution over the public Internet, a result dominated by the performance of the network over the geographic distance involved. The remote site was also able to successfully add their site specific Roll and it was consistent with the overall software installation. Our initial assessment is that the Rocks wide area kickstart extension successfully satisfies many of the GEON requirements and we believe similar principles can be applied to a grid consisting of many nodes. Limitations of the system are detailed in future work.

5.3. Future Work

Not all the requirements of GEON were met by the Rocks cluster distributions current release. We require a frontend node to be able to initiate an upgrade automatically. Another specification sought is the fine-grained incremental upgrade of critical RPMS. This feature is also absent from the Rocks system. In addition, nodes in wide area grid systems like GEON have the concept of user space and OS distribution space. Our ideal solution would posses the functionality to update or reinstall the OS distribution

completely without affecting the user environment in any way. We expect to address these requirements in future iterations of the Geon grid design.

6. Conclusion

This paper discusses the requirements, architecture and applicability of Rocks wide area grid systems deployment and management. We discussed some of the requirements of GEON grid deployment and management and how Rocks wide area functionality addresses those requirements. We also discussed some initial results and talk about possible extensions to the Rocks toolkit that would benefit GEON and similar grid systems.

We believe in GEON and other grid architectures the key is to manage an extremely heterogeneous hardware, computing, storage, and networking environment, rather than an "ultra-scale" environment. The complexity of management (e.g., determining if all nodes have a consistent set of software and configuration, incrementally updating patches, etc) of such grid systems is challenging. Rocks wide area kickstart and management toolkit establishes a rational software distribution and management plan so that the GEON network can easily expand with new clients, computational resources, services, middleware components, and other resources with minimal effort.

Acknowledgements

This work is funded by the NSF (GEON, Grant Number: 225673) and NPACI. We would like to thank Satish Tadepalli from Virginia Tech University for assisting us with some of the experiments.

References

1. NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters, Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno, October 2001, Cluster 2001
2. P. Anderson and A. Scobie. LCFG: The next generation. In UKUUG Winter Conference, 2002.
3. GEON, CyberInfrastructure for the Geosciences, http://www.geongrid.org/
4. BIRN, Biomedical Informatics Research Network, http://www.nbirn.net/
5. GriPhyn, Grid Physics Network, http://www.griphyn.org/index.php
6. An open platform for developing, deploying and accessing planetary-scale services. http://www.planet-lab.org/
7. Building the National Virtual Collaboratory for Earthquake Engineering http://www.neesgrid.org/index.php
8. Southern California Earthquake Center. http://www.scec.org/
9. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001.
10. Thomas E. Anderson, David E. Culler, and David A. Patterson. A case for NOW (networks of workstations). IEEE Micro, 15(1):54–64, February 1995.
11. T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In Proceedings of the 24th International Conference on Parallel Processing, volume I, pages 11–14, Oconomowoc, WI, 1995.
12. B. E. Finley. VA SystemImager. In Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA, Oct. 2000.
13. Scyld Software. Scyld Beowulf Clustering for High Performance Computing. Scyld Software Whitepaper, www.scyld.com.
14. Yellow dog Updater, Modifies http://linux.duke.edu/projects/yum/
15. NSF Middleware Initiative http://www.nsf-middleware.org/
16. Distributed Terascale Facility to Commence with $53 Million NSF Award. http://www.nsf.gov/od/lpa/news/press/01/pr0167.htm
17. The Inca Test Harness and Reporting Framework, Shava Smallen, Catherine Olschanowsky, Kate Ericson, Pete Beckman, Jennifer Schopf. Submitted to Supercomputing 2004
18. Rocks Rolls: A First-Class Method to Interact with Standard System Installer. Mason J. Katz, Greg Bruno. Submitted to Cluster2004.