Grid Appliance – On the Design of Self-Organizing, Decentralized Grids

David Wolinsky and Renato Figueiredo
Advanced Computing and Information Systems Laboratory
University of Florida
[davidiw,renato]@acis.ufl.edu

Keywords: grid computing, cloud computing, virtual private networks, P2P computing, peer-to-peer

Draft of 2010/05/18 12:42

"Give a man a fish, feed him for a day. Teach a man to fish, feed him for a lifetime" – Lau Tzu

Grid computing projects such as TeraGrid [5], Grid'5000 [4], and OpenScience Grid [7] enable researchers to access vast amounts of compute resources, but in doing so, they force the researcher to adapt their workloads to the environments these systems provide. This does not leave many alternatives for researchers as creating these types of systems requires coordination and expertise in networking, operating systems, security, and grid middleware. This results in many research groups creating small, in-house compute clusters where scheduling is often ad-hoc (and resource utilization is low), and aggregation of resources across multiple groups is hindered by the complexities in constructing federated systems. This paper describes the "Grid Appliance", the first system of its kind that enables researchers to efficiently deploy their own compute clusters, and to seamlessly extend their systems across network domains to create small to large scale computing grids. The paper details the design of the Grid Appliance and reports on experiences and lessons learned over four years of development and deployment of wide-area Grid appliance pools.

Our approach begins with a P2P infrastructure enabling peers to discover each other and coordinate the organization of a grid. We use a P2P system based upon a distribute hash table (DHT), which enables peers to efficiently query the system with a key and discover one or more values that map to that key. We also layer on top of the P2P overlay a virtual private network (VPN), which provides all-to-all connectivity and transparently handles configuration and organization through the DHT. The grid middleware can use either the DHT or the VPN to bootstrap various grid middleware stacks, such as Condor, Hadoop, MPICH, or PBS (Torque). The entire system has been packaged into a repository to automatically configure physical, virtual, and even cloud resources. The resulting "Grid Appliance" is a preconfigured environment emphasizing trivial installation, enabling user interaction with the system to focus on the tools provided and not configuration details, providing researchers with a plug-and-play tool to create ad-hoc virtual compute clusters for their own groups, local or federated.

To justify our techniques, consider the difficulty in combining resources across disparate networks, which may or may not involve multiple research groups. Challenges such as safety, security, connectivity, and efficiency require an information technology (IT) expert. Furthermore, network constraints present another complexity beyond configuration and organization of distributed resources. An individual group may have resources behind different network address translators (NATs) and firewalls, preventing direct communication with each other. Even assuming that an institution's network administrator is willing to make exceptions for the grid, additional rules may be required for each new cluster or resource added internally and externally, quickly becoming unmanageable.

The grid systems considered in this paper consist of three fundamental components: execute nodes, resource managers, and submission nodes. The execute nodes run the tasks submitted from the submission sites. Users access a submission site, craft task description files, and submit them to a scheduler or resource manager, which will queue tasks to the various execute nodes.

Existing work that falls under the general area of desktop grids/opportunistic computing include Boinc [2], BonjourGrid [1], and PVC [8]. Boinc, used by many "@home" solutions, focuses on adding execute nodes easy; however, job submission and management rely on centralization and all tasks must use the Boinc APIs. BonjourGrid removes the need for centralization through the use of multicast resource discovery; the need for which limits its applicability to local area networks. PVC enables distributed, wide-area systems with decentralized job submission and execution through the use of VPNs, but relies on centralized VPN and resource management.

Each approach addresses a unique challenge in grid computing, but none addresses the challenge presented

as a whole: easily constructing distributed, cross-domain grids. Challenges that we consider in the design of our system are ensuring that submission sites can exist any where and are not confined to complex configuration or highly available, centralized locations; ability to dynamically add and remove resources by starting and stopping an appliance; and the ability for individual sites to share a common server or to have one or more per site so that no group in the grid is dependent on another. We emphaize these points, while still retaining the ease of use of Boinc, the connectivity of PVC, and the flexibility of BonjourGrid. The end result is a system similar in organization to OurGrid [3], though with self-configuration and organization and a VPN to transparently handle network constraints.

"Grid Appliance" systems can be constructed in one of two ways. Local grids can be constructed by booting the appliances, which will then use multicast self-discovery to find other resources, create the DHT overlay, and then form VPN links. Alternatively, a wide area grid construction is managed through a web site and bootstrapped from a P2P overlay, either our public shared deployments or private systems easily deployed through a virtual appliance. The web site uses a group interface, similar to online social networking groups, as a wrapper around a certificate authority based public key infrastructure. Users can join a group or create their own. To support collaboration across groups, groups are divided into two categories: VPN groups (used to determine nodes authorized to connect to the VPN) and institution groups (used to determine priority in accessing resources). In practice, each collaborative group represents a unique VPN and each institutional group represents a collective subset of the grid.

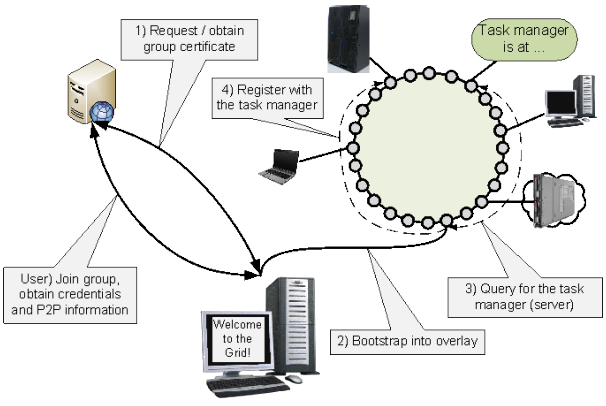

Figure 1: An example deployment scenario: obtaining configuration files, to starting the appliance, and connecting with a resource manager.

As presented in Figure 1, users join a resource group for VPN purposes and then an institutional group to obtain priority for resources shared by their group. Upon joining both groups, the user can download one of three different types of configurations: server, worker, and client (mapping to the three types of typical grid nodes, resource manager, execution site, and submission site). In addition to the configuration data, a first time user can download the generic appliance, or one extended with additional software customized for their community. In the case of a virtual machine appliance, the configuration data can be added to the system as a virtual floppy image, automatically configuring the appliance for a specific behavior. In the case of the client, it begins searching for an appropriate resource manager by querying the DHT. Upon finding an entry in the DHT, the client communicates with the resource manager via the VPN. The same procedure can be repeated many times to add additional resources into the grid. Resources booted using the "server" image configure and advertise themselves through the DHT or VPN. Each individual grid can support multiple servers for fail-over or load balancing, depending on the capabilities of the grid middleware (e.g. Condor flocking). Beyond the downloading of the appliance and the configuration data, all other steps are performed transparently from the user.

As of this date, we have focused on two uses for this system: high throughput computing (HTC) systems for research and self-configuring modules for education. To this end, we have used the Condor grid middleware for our HTC, which we will discuss shortly. On the other spectrum, techniques that we describe enable easy construction of educational clusters/grids, which would otherwise be significant time sinks for students who need to learn more about how a grid works and less about how to configure one across domains. As of this date, we have implemented Hadoop and MPI educational packages. For HTC, we chose Condor because, through simple configuration, it enables resources to join dynamically, submit from any location, enforce priorities based upon groups, and allow the sharing of resources across multiple resource managers. In the paper, we will discuss why we chose Condor and what separates it from other solutions in addressing our challenges.

	50 - EC2	50 - NEU	50 - UF	150 - All
Start	2:44	10:21	20:23	21:14
Connect	2:27	11:36	3:53	17:13
Run	7:15	6:35	5:53	21.19

Table 1: Time in minutes:seconds to start and connect execute nodes from various sites, Amazon EC2, Northeastern University, and University of Florida, to an already online resource manager, and then the time to run a 5 minute job from a freshly connected submission node.

As proof to the validity of our approach, we present an evaluation of our system across various resources as presented in Table 1 and describe several deployments of our described system. Over the past 2 years, we have had an active grid deployed for computer architecture research, Archer [6]. Archer currently spans four seed universities with 500 resources with over hundreds of students and researchers submitting jobs totaling over 150,000 hours of job execution in the past year alone. Groups at the Universities of Florida, Clemson, Arkansas, and Northwestern Switzerland have used it as a tool to teach grid computing. Clemson and Purdue are constructing campus grids using the underlying VPN to connect resources together. Recently, two private small-scale systems have come online using our shared system available at `www.grid-appliance.org`. Feedback from users through surveys have shown that non-expert users are able to connect to our public Grid appliance pool in a matter of minutes by simply downloading and booting a plug-and-play VM image that is portable across VMware, VirtualBox, and KVM.

References

[1] H. Abbes, C. Cérin, and M. Jemni. Bonjourgrid: Orchestration of multi-instances of grid middlewares on institutional desktop grids. In *IPDPS*, pages 1–8, 2009.

[2] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *the International Workshop on Grid Computing*, 2004.

[3] N. Andrade, L. Costa, G. Germglio, and W. Cirne. Peer-to-peer grid computing with the ourgrid community. In *in 23rd Brazilian Symposium on Computer Networks (SBRC 2005) - 4th Special Tools Session*, 2005.

[4] F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jegou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, and O. Richard. Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform. In *Grid'2005 Workshop*, Seattle, USA, November 13-14 2005. IEEE/ACM.

[5] C. Catlett. The philosophy of teragrid: Building an open, extensible, distributed terascale facility. In *CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 8, Washington, DC, USA, 2002. IEEE Computer Society.

[6] R. J. Figueiredo, P. O. Boykin, J. A. B. Fortes, T. Li, J. Peir, D. Wolinsky, L. K. John, D. R. Kaeli, D. J. Lilja, S. A. McKee, G. Memik, A. Roy, and G. S. Tyson. Archer: A community distributed computing infrastructure for computer architecture research and education. In *Collaborative Computing: Networking, Applications and Worksharing*, volume 10, pages 70–84. Springer Berlin Heidelberg, 2009.

[7] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Wrthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick. The open science grid. volume 78, page 012057, 2007.

[8] A. Rezmerita, T. Morlier, V. Neri, and F. Cappello. Private virtual cluster: Infrastructure and protocol for instant grids. In *Euro-Par*, 2006.

Outline

- Introduction
  - Background / motivation
  - Related work
- Choosing a Grid Middleware, comparing:
  - Condor
  - PBS (Torque)
  - Sun Grid Engine
  - State of the art BonjourGrid, etc
- Connecting the Grid, VPNs
  - Need for VPNs
    - LAN / Multicast
    - NATs / firewalls
    - IPv4 Address space
  - P2P VPN
    - Bootstrapping
    - Organization
- Constructing an individual resource
  - Packaging
    - Virtual Appliances
    - Cloud Instances
    - Physical resources (generic deployment)
  - Making it easy for users to extend and redistribute
  - Security considerations
- Constructing the Grid
  - Organizing Grids
    - DHT approach
    - IP Multicast
  - Designing and interacting with a collaborative website
- Evaluation
  - User feedback
    - Surveys
    - Correspondence with remote deployments
    - Use in courses
  - Deployment across clouds
  - Bootstrapping local independent pools
- Conclusions
  - Overview of how our approach addresses challenges
  - Future work