InstantGrid: A Framework for On-Demand Grid Point Construction

Roy S.C. Ho, K.K. Yin, David C.M. Lee, Daniel H.F. Hung, Cho-Li Wang, and Francis C.M. Lau

Department of Computer Science
The University of Hong Kong, Hong Kong*
{scho,kkyin,cmlee,hfhung,clwang,fcmlau}@cs.hku.hk

Abstract. This paper proposes the InstantGrid framework for on-demand construction of grid points. In contrast to traditional approaches, InstantGrid is designed to substantially simplify software management in grid systems, and is able to instantly turn any computer into a grid-ready platform with the desired execution environment. Experimental results demonstrate that a 256-node grid point with commodity grid middleware can be constructed in five minutes from scratch.

1 Introduction

To build platforms for grid computing is nontrivial. First, there is the problem of potential mismatch between the platforms and what is expected or required by the applications. Therefore, grid systems need to build upon a custom model that can manage multiple execution environments (EE's) in order to provide flexible support matching the applications. Secondly, the installation and configuration of contemporary OS'es or middleware are usually a time-consuming process. As a result, it could take a long while to construct an EE, or to switch from one EE to another, which complicates system administration and results in poor user experience. Finally, it is difficult to construct grid platforms with customized EE's in production systems—even if there are idle computing resources. This is because the installation of additional software might affect the data originally stored in the permanent storage, an effect that is certainly not to be desired. While the current R&D efforts have been focusing on how to aggregate (e.g., [1][4]) and make use of (e.g., [2][3]) distributed computing resources, few addressed the above issues. In this paper, we report on the *InstantGrid* framework, which is the result of our effort in creating a tool to facilitate the management of EE's and their efficient dissemination to networked machines on-demand according to the requirements of the target applications. In response to the last problem mentioned above, InstantGrid supports an *in-memory execution* mode for an EE to be executed entirely in the physical memory without affecting the data stored in the permanent storage.

* This research is supported in part by HKU Foundation Seed Grant 28506002, HKU Large Equipment Grant 01021001, and a HKU grant for the HKU Grid Point.

2 The InstantGrid Framework

InstantGrid works in the client/server mode, where all EE's are stored in and managed by an InstantGrid server from which the compute nodes obtain their EE's on-demand to form the grid platform (Figure 1). The framework allows for replication of the InstantGrid service for better performance and reliability. The framework consists of an *EE management model* and an *EE dissemination service*, as shown in Figure 2.

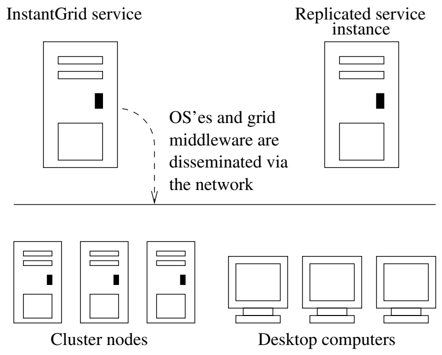

InstantGrid service
Replicated service instance
OS'es and grid middleware are disseminated via the network
Cluster nodes
Desktop computers

Fig. 1. InstantGrid Servers and Clients

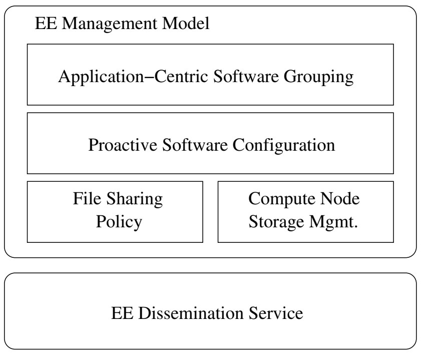

EE Management Model
Application–Centric Software Grouping
Proactive Software Configuration
File Sharing Policy
Compute Node Storage Mgmt.
EE Dissemination Service

Fig. 2. The InstantGrid Framework

Application-centric software grouping. In InstantGrid, an EE is a collection of software components, including an OS, the supporting libraries and applications, grid middleware, cluster middleware, user applications, and the user data. Essentially, we can see what is in an EE by taking a snapshot of all software components in a running system. Each EE is associated with an *EE specification*, which contains a list of software required by a specific application. This application-centric requirement specification facilitates EE management as different users and their applications may require different platforms over time. Once the administrators have defined the list of software in each EE, the system can conveniently switch from one EE to another according to the user requirements, without having to deal with the individual software components.

Proactive software configuration. Traditionally, the OS and system software are installed and configured incrementally. In InstantGrid, by contrast, software belonging to the same EE have to be configured in the central server before being disseminated to the compute nodes. In other words, InstantGrid creates and maintains ready-to-run versions of various EE's. This arrangement saves installation and configuration time during the dissemination process. Nevertheless, some software must perform *local* configuration. For instance, some grid middleware (e.g., Globus Toolkit) require host credentials such as certificates, which have to be handled locally at the respective nodes. In any case, Instant-

Grid takes a "greedy" approach and carries out as many configuration tasks in the central server as possible, leaving minimum work to the compute nodes.

Discriminative file sharing mechanisms. Disseminating the entire EE from the InstantGrid server to the compute nodes is challenging as the size of a typical EE could be in the order of gigabytes, and full replication is therefore impractical. However, to access all files through NFS all the time is also inefficient since many files in an EE are frequently updated—retrieving them through the NFS would result in poor runtime performance as well as heavy loading at the file server. InstantGrid adopts a hybrid approach to the problem: it only replicates files that are likely to be modified (e.g., the files in `/etc` or `/dev`), and leaves the others in the file server for sharing via a network file system.

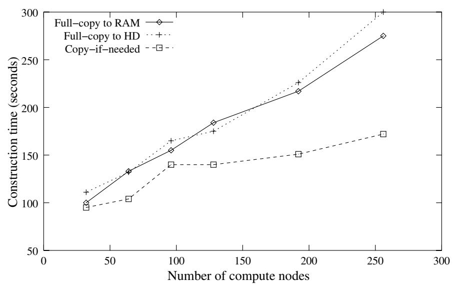
Compute node storage management. InstantGrid allows the files being replicated to a compute node to be stored in the hard disk ("Full-copy to HD") or entirely in the physical memory ("Full-copy to RAM"). The in-memory execution mode enables any computer to participate in a grid without affecting the data stored in its local storage. If the files are stored in the hard disk, InstantGrid supports an additional option of I/O caching ("Copy-if-needed"): before a file is transferred from the InstantGrid server to a compute node, InstantGrid would first check if there is a local version of that file; if so, it would verify whether it is up-to-date, and file transfer would only take place if the file is missing or outdated.

EE dissemination service. This service is offered through a DHCP server, a TFTP server, and an NFS server. When a client machine boots up, it obtains its IP address and the kernel (a Linux kernel in our reference implementation) from the DHCP and TFTP servers, respectively. When the booting process finishes, InstantGrid constructs the pre-defined EE by replicating writable files to local storage and mounting the read-only directories through the NFS.

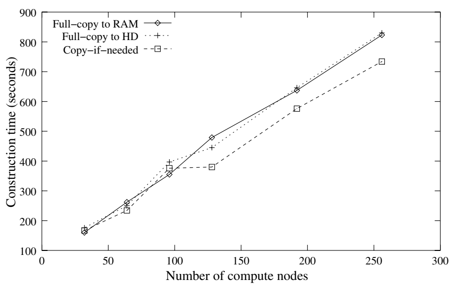
3 Experiments

We evaluated the performance of InstantGrid using the HKU CS Gideon cluster which consists of 300 Pentium IV machines (we used 256 in the experiments). The EE's are disseminated through a hierarchical Ethernet network in which 13 24-port Fast Ethernet switches are interconnected by a Gigabit Ethernet switch. The InstantGrid server, which connects to the Gigabit Ethernet, is a Pentium IV machine with 512MB RAM and an IDE hard disk. In the experiments, InstantGrid disseminated the Fedora Core 1 OS to the cluster nodes by sharing the `/bin`, `/lib`, `/sbin`, and `/usr` directories through the NFS, and replicating the remaining directories to the nodes' local storage.

Figure 3(a) presents the time to construct a grid point which is a cluster consisting of a frontend node (Fedora OS, Globus Toolkit 2.4, Ganglia, and PBS) and a number of compute nodes (Fedora OS, Ganglia, PBS, and MPICH-G2). The results show that a 256-node grid point can be constructed from scratch in three ("Copy-if-needed") to five ("Full-copy to HD") minutes. The good performance mainly attributes to the proactive software configuration in InstantGrid,



(a) A cluster–based grid point



(b) Standalone grid points

Fig. 3. Construction Time of Grid Points

which shortens substantially the dissemination time. Figure 3(b) shows the construction time of standalone service grid points (Fedora OS, Globus Toolkit 3.2, and Ganglia). The reason for the longer construction time compared to that of the previous test is that each compute node is treated as a standalone grid point, i.e., each node requires a unique host certificate; and that the certificates have to be generated *sequentially* in a central certificate authority server. This result suggests that some software configuration processes are indeed time consuming, which should either be avoided or redesigned to allow for more efficient deployment.

4 Conclusion

We have proposed the InstantGrid framework for on-demand construction of grid points. The experimental results show that InstantGrid is able to efficiently construct Linux-based grid points with commodity grid middleware. Future work will be conducted along two dimensions. First, we will devise standard protocols for communicating EE specifications between the InstantGrid servers and compute nodes. Secondly, we will look into possible performance optimizations for InstantGrid in WAN, which could enable remote construction of grid points through broadband networks.

References

1. A. Grimshaw and A. Ferrari and A. Knabe and M. Humphrey. Legion: An Operating System for Wide-Area Computing. *IEEE Computing*, 32(5):29–37, May 1999.
2. I. Foster, C. Kesselman, J.M. Nick, and S. Tueckel. The Physiology of the Grid - An Open Grid Services Architecture for Distributed Systems Integration. In *White Paper, The Globus Project. http://www.globus.org/.*
3. J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, 5:237–246, 2002.
4. S. Zhou. LSF: Load Sharing in Large-scale Heterogeneous Distributed Systems. In *Workshop on Cluster Computing*, 1992.