HaDeS: a Scalable Service Oriented Deployment System for Large Scale Installations

Massimo Gaggero[1], Gianluigi Zanetti[2]
[1]CRS4, Edificio 1 - Pula (CA), Italy, massimo.gaggero@crs4.it
[2]CRS4, Edificio 1 - Pula (CA), Italy, gianluigi.zanetti@crs4.it

***Abstract*—Building large computational facilities requires scalable and flexible deployment tools that can cope with massive loads. Classical installation methods are not very flexible, since they are usually limited in the number of OS supported, rely on transfer solutions that impose constraints on network topology, and do not scale very well. Here we describe HaDeS (Hardware Deployment System), a new deployment system for large scale installation designed to be agnostic with respect to the network topology and the OS deployed and to scale with the number of nodes being deployed.**

***Index Terms*—Deployment, scalability, operating system, hades, cybersar, metalink.**

I. Introduction

CYbersar [1] is a cyber–infrastructure that connects and supports all the research organizations in Sardinia that heavily employ computational facilities to support their research activities. Together with the “classical” use of HPC facilities, Cybersar supports research focused on the study and development of new computational paradigms, e.g., infrastructure as a service for scientific applications, that require continuous development of the entire software stack, from the OS, to libraries, middle-ware and applications. Cybersar infrastructure thus needs to support, in a scalable manner, heavy customization of Operating Systems and frequent cluster reconfiguration in a spirit similar to what is done in the project GRID5000 [2].

Building large computational facilities require scalable and flexible deployment tools that can cope with massive loads. Classical installation methods are usually limited in the number of OS supported and, typically, not very flexible. Moreover, they usually rely on broadcast transfers or remote OS mounting solutions that are not very robust, are network topology dependent, and do not scale very well with the number of nodes to be installed. On the other hand, to support cybersar research we needed a hardware deployment system that satisties the following requirements: it should be

- independent from both the OS to be installed and the packaging system;
- fast and scalable with respect to the number of nodes;
- support a fully automatized and unattended deployment process;
- able to support incremental changes in large scale installation;
- integrable in complex system, e.g., so that it can be used as service component in SOA based work-flow applications,

Since the deploy system is supposed to be orchestrated by higher order entities [3], [4], it does not need to be a complete cluster management system, but just an intelligent and flexible deployment tool.

To satisfy these requirements we developed HaDeS. HaDeS image distribution mechanism is based on standard transfer protocols used in common p2p file-sharing applications. HaDeS can, therefore, work across LAN boundaries and it is compatible with multi-site geo-

graphical installations. HaDeS development has been kept strictly independent from the OS deployed and no assumptions were made about the images transferred. The system is built following SOA principles and the controlling interface is developed as Web Service. This approach allows for an easy integration of HaDeS as a module in complex management infrastructures while keeping it also usable as a standalone application. HaDeS is used within Cybersar as a component of the VIDA [4] infrastructure to manage fast whole cluster redeployment, e.g., to reconfigure clusters as hadoop (an open source implementation of map reduce [5]) machines, and to deploy test installations of cloud computing middleware such as OpenNebula [6].

A. Flexibility: OS Image Deployment Vs Package Installation

The automatic and unattended tools commonly used for cluster setup and configuration are packaging-system oriented and often strictly related to the OS installed [7] [8] and to the packaging system [9] used; albeit few, [10], are more flexible and can handle different OS and packaging system. While this approach is appropriate for the usual handling of computational facilities where the setup is done basically only once and the update is incremental, it is not particularly efficient in a quickly varying research environment focusing on the computational infrastructure itself.

Package installation is usually performed as a sequence of steps of widely varying duration: a target disk is selected, partitioned and formatted; software is transferred from the network to the target host and installed; the post-inst phase performs specific customizations and the installation of software that cannot be automatized. Since it is dependent on the amount of configuration and customization done, post-install can take longer than the installation phase and require manual intervention.

A more flexible approach is the deployment of a pre-built and pre-configured image of the operating system as a single file. As shown in fig 1 the mandatory steps for image deployment

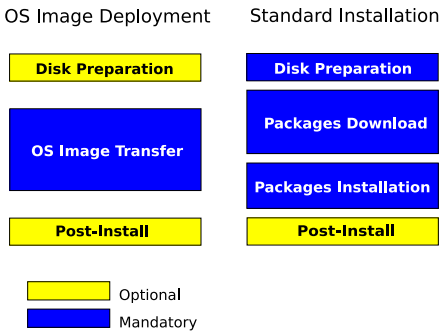

Fig. 1. Deployment and installation phases

are reduced to one. As the image comes with the whole file-system structure on it, partition formatting is no more required. Packages download and installation are replaced by the only operation of image file transfer from network to disk. A quick post-inst phase is possible, if required, but an accurate customization can avoid it. With current network and disk technology, the downloading of the disk images is bound by the transfer to disk bandwidth. Thus, with standard hardware, the time needed for a single image transfer is in the few minutes range.

B. Scalability: P2P and multi-protocol file transfer

Scalability is an issue related to the transfer protocol. Some tools deal with scalability using different file transfer protocols. SystemImager [11], for example, offers three different protocols with three different modules: rysnc transfer, multicast transfer and BitTorrent. Each one has advantages and disadvantages: Rsync is able to transfer only file deltas but it does not scale, multicast is more scalable but it requires specific network topology, BitTorrent can take advantage of multi-servers and downloaded chunk sharing but forces the images to be provided as torrent file. Moreover, since many ISPs and firewalls drop p2p protocols, geographical deployment cannot be done.

A flexible solution that overcomes these limitations is the Metalink File Format [12]. With Metalink it is possible to indicate different

sources and different transfer protocols pointing to the same file. Compatible clients can select between protocols and source to maximize the incoming bandwidth. Mixing together HTTP sources and P2P sharing for example, overcomes strict network policies while allowing for local chunks sharing.

C. Full Automation: controlling the hardware

Installation or deployment typically requires two reboots:

- machine power-on/reset;
- boot of the installation system;
- installation/deployment process;
- machine reboot.

In order to be really automatic and unattended, deployment has to manage also the hardware power switching. A standard embedded methods often available as standard is the IPMI protocol. As an alternative, many main-board manufacturer provide proprietary solution like HP Onboard Manager and iLO and Sun Ilom. One tool able to deal with hardware and issue power switching command is xCAT [13]: it uses IPMI standard to control power and boot device. However, it does not handle other Platform Control (PFC) methods and it does not provide full support for image deployment.

D. Integration

The usual tools for cluster management and provisioning tend to have an autocratic approach, i.e., the assume to be the only system controlling the computational infrastructure, do not usually adhere to interfacing standards and are very difficult to integrate in more complex systems. A more modern approach is to think to the deployment system as a component, that can work standalone, but it can also be readily integrated in a larger control-plane infrastructure, e.g., orchestrated as a SOA [3], [4].

II. Discussion

HaDeS is mainly a flexible tools for research: its development has been kept strictly independent from the OS deployed and no assumptions were made about the images transferred.

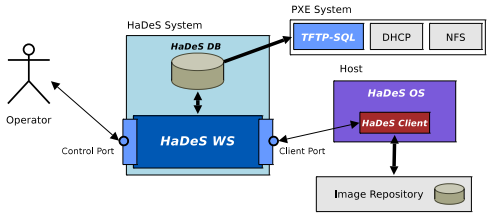

Fig. 2. HaDeS components

HaDeS system is composed by 5 different software components:

- the HaDeS Scout Image;
- the HaDeS Client;
- the HaDeS Web Service along with the HaDeS Database;
- the SQL-TFTP Server;
- the PFC Web Service.

A. HaDeS Scout Image and HaDeS Client

HaDeS works on the target host atop of a minimal disk-less operating system, the HaDeS Scout Image; as the target machine is booted, the OS is loaded via PXE. We are using for the Scout Image a Gentoo [14] LiveCD image, since it is small and contains a wide selecton of the disk and file system management tools needed by the HaDeS client. Moreover, the Gentoo LiveCD is able to detect a large quantity of different hardware devices and it can be very easily upgraded and customized using Gentoo tools such as Genkernel and Catalyst. The Scout Image is mounted from a NFS server by the target system and the root file-systems contained is loaded in memory, to reduce the network impact of NFS accesses. The NFS mounting of the Scout Image does not affect scalability, since only small and infrequent NFS data transfers are required during the boot and deployment processes. Moreover, the performance impact of NFS could be reduced, if needed, with server redundancy.

Once loaded, the Scout Image starts all the services of a common Linux system such as network configuration, SSH and Syslog daemons; the last service started is the HaDeS client. The latter collects information about the target host

and use them to query the HaDeS Web Service on the actions to be performed.

Then deployment starts following configuration information provided by the Web Service: the related schema is called *host setup* and it indicates the target partitions, the type of file-system to be created/deployed and the remote and local location of the image to be deployed. The host is then rebooted from a boot device at the end of deployment as per setup instructions.

During the process, HaDeS client talks to the HaDeS Web Service, reporting process status and errors occurred.

In order to adhere to well known standards, the setup is a XML file and communication between client and Web Service is implemented as SOAP messages. The client is written in Python 2.5 [15] and run atop of an instance of the Twisted networking engine [16].

B. HaDeS Web Service and Database

All the informations about host managed by HaDeS are stored in the HaDeS Database. Hosts are described by their name, the setup to use for deployment, the next boot device, the collection of the network interfaces with MAC and IP address and all the Platform Control (PFC) methods provided to the host and which one is currently in use. The database is used by the HaDeS Web Service to provide clients with all the instructions needed to perform the deployment. Details about PFC methods are used, instead, by the PFC Web Service to control deployment targets hardware.

The HaDeS Web Service provides two ports:

- *Control Port:* it is intended to be used by the user, by means of the HaDeS tools, to communicate with the database requests such as list and details of hosts, setups, boot types, PFC details etc...;
- *Service Port:* it is the port used by the HaDeS clients to ask for instructions and to report process status.

When a client sends data gathered about a host that is not already in the database, and HaDeS is setup for automatic node discovery, the Service Port issues to the Control Port an add command for that host.

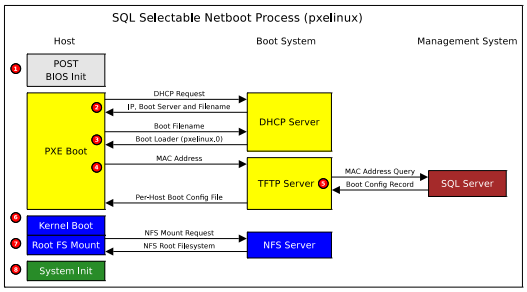

Fig. 3. SQL selectable PXE boot

HaDeS Web Service is developed in Python using the PyGridWare [17] library, a python implementation of the WS-Resource framework and part of the Globus [18] Toolkit. It is WSDL 1.1 [19] compliant and uses SOAP [20] messages. The database engine used is MySQL.

C. SQL-TFTP Server and PFC Web Service

Two important HaDeS software components not strictly related to deployment are the SQL-TFTP Server and the PFC Web Service. They control the network boot process when PXE is used by the nodes. They are crucial components of HaDeS since the deployment process requires hardware power control and automatic boot selection. The PXE boot system relies on a TFTP server to retrieve the boot loader and related configuration files, being able to programmatically select the boot file provided by the TFTP server makes it possible to control the nodes booting sequence.

We have developed a patch for the TFTP server that interface it to the SQL Database allowing boot selection just as a matter of changing the relevant boot record in the database when a deployment is issued to HaDeS, the Database makes the node point to the HaDeS scout image and, when finished, restores it back to the default device. Figure 3 illustrates the role of the SQL-TFTP server in selectable boot process.

The PFC Web Service instead is in charge of handling the way the hardware is turned on or

off or reset. The most reliable way is to issue power command using IMPI messaged to the node’s Board Management Controller (BMC) or the equivalent for the board manufacturer (HP iLO for example or Sun iLOM). PFC Web Service acts as a proxy to these protocols: depending of the data provided by the HaDeS Database, it issues to the hardware the appropriate message when a reset is required to start the deployment. It is developed as a modular system allowing new protocols and new platform control methodologies to be added later.

D. Scalability

Scalability has been reached in HaDeS with the adoption of the *aria2* [21] utility developed by Tatsuhiro Tsujikawa. Aria2 is an application for the parallel download of files from multiple sources/protocols such as HTTP(S)/FTP and BitTorrent. While the data is downloaded using HTTP(S)/FTP, it is uploaded to the BitTorrent swarm. Thus, In this way just a small number of nodes directly downloads from the image repository while the others refer to them and one to each other using BitTorrent. This spreads the transfer to the internal lan making it possible to use all the available aggregate bandwidth of the network infrastructures leaving a reasonable high and constant transfer speed per client instead of a $\frac{1}{N}$ drop, where N is the number of nodes.

III. Preliminary results

Several proofs of concept and functional tests have been carried out on HaDeS, while intensive efficiency and scalability tests are currently being performed. Here we describe some of the preliminary results thus far collected. Figure 5 illustrates results from a scalability test done deploying an image of about 3 GBytes over 16 nodes. The image is hosted in a single HTTP server with a 1Gbps link to the node’s switch. Aria2 is used to download the image and share available chunks with BitTorrent. The ”HTTP/BT” curve show the average transfer

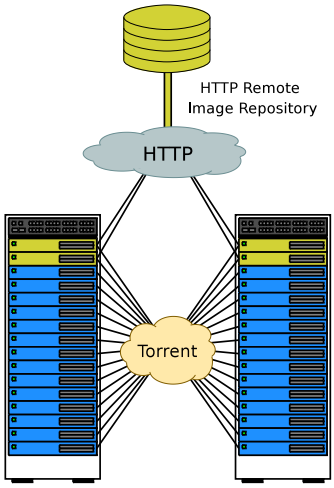

Fig. 4. Combinated use of HTTP and Bittorrent protocols

speed per client as the number of nodes increases, while the ”HTTP” represents the estimated per-client speed using just HTTP. The use of the p2p sharing has led to an effective per node transfer rate of about 20 MBytes/sec, that is essentially constant with respect to the number of nodes deployed. As a reference, the effective disk bandwidth on the same node hardware is of about 40 Mbytes/sec.

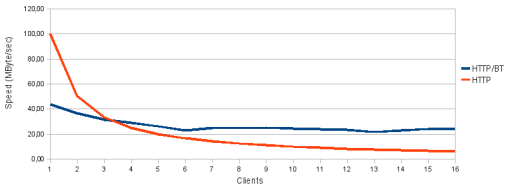

Fig. 5. Per-client transfer speed

The second test described is the deployment over 72 nodes of a 4,5 GBytes image during a Hadoop (an open source Map Reduce implementation [5]) evaluation test. As above, the image is downloaded by a single HTTP server and available chunks are shared using the BitTorrent protocol. Table I shows that the deployment time with hades is 323 sec, about 5 minutes, while the transfer only takes about 4 minutes; the difference of 75 seconds is the time spent to create the accessories partition required

by Hadoop to store data. The average transfer speed is about 19 MByte/sec, definitely better than the estimated 1,38 MByte/sec of a simple HTTP download from 72 clients.

TABLE I
DEPLOYMENT OF A 4,5 GBYTES IMAGE OVER 72 NODES

HaDeS Time	Aria2 Time	Transfer Speed
323,49 secs	248,47 secs	19,12 MByte/sec

IV. CONCLUSION

HaDeS has been developed initially with the aim to provide a flexible tool to allow the rapid deployment of heavily customized Operating System images and the fast switching of clusters between production and development environments. Scalability results from the use of p2p technologies such as BitTorrent protocol and Metalink file format. The Web Service interface allows HaDeS to be integrated as component in complex system SOA based. From this point of view, HaDeS has been successfully used in the VIDA [4] system for virtual cluster deployment. Other experiments and evaluations in the Cybersar project has been carried out using HaDeS as deployment tool and configuration switcher for the Cybersar cluster (72 nodes) hosted at CRS4. During Hadoop framework evaluations, HaDeS has switched the whole cluster from production to development and back to its initial state. The whole evaluation took less than two days, with just six minutes for the complete deployment of a pristine Hadoop environment and a minute to revert back the nodes into production. In a similar manner, HaDeS has been used to deploy a test installation of the OpenNebula cloud computing middle-ware [6]. Feedback from these tests has suggested a quantity of improvements and a new 2.0 version is under development. Changes include the development of new drivers for different OnBoard Control Systems, improvements in system robustness and resilience, deployment process monitoring and so on. At the same time, new test are planned to investigate scalability performance of HaDeS in different network topologies and its behavior at the geographical scale.

REFERENCES

[1] P. Anedda, C. Guidi, G. Zanetti, A. Bosin, A. Masoni, D. Mura, N. D'Amico, and I. Porceddu, "Cybersar: a new computational infrastructure for research in sardinia," in *3rd IEEE International Conference on e-Science and Grid Computing, Bangalore, India, 10-13 Dec. 2007*, December 2007.

[2] 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: IEEE/ACM, November 13-14 2005.

[3] P. Anedda, S. Manca, M. Gaggero, and G. Zanetti, "Soa based control plane for virtual clusters," in *Euro-Par 2007 Workshops Parallel Processing - HPPC 2007, UNICORE Summit 2007, and VHPC 2007*, vol. 4854, 2007, pp. 154–163.

[4] P. Anedda, C. Guidi, and G. Zanetti., "Vida: A new virtual images deployment architecture," in *Workshop finale dei Progetti Grid del PON "Ricerca" 2000-2006 - Avviso 1575*, 2009.

[5] A. Bialecki, M. Cafarella, D. Cutting, and O. O'Malley, "Hadoop: a framework for running applications on large clusters built of commodity hardware, 2005," *Wiki at http://lucene.apache.org/hadoop*.

[6] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity leasing in cloud systems using the opennebula engine," in *CCA08: Cloud Computing and its Applications, Chicago Oct. 22/23 2008*, October 2008.

[7] C. Winter and D. Layfield, "Using solaris jumpstart with the solaris 10 os for x86/x64 platforms," *BigAdmin System Administration Portal*, March 2007. [Online]. Available: http://www.sun.com/bigadmin/features/articles/jumpstart_x86_x64.jsp

[8] A. Nashif and U. Gansert, *AutoYaST - Automatic Linux Installation and Configuration with YaST2*, SUSE Linux Products GmbH. [Online]. Available: http://www.suse.com/~ug/autoyast_doc/index.html

[9] "Anaconda/kickstart." [Online]. Available: http://fedoraproject.org/wiki/Anaconda/Kickstart

[10] T. Lange, "Fai - fully automatic installation." [Online]. Available: http://www.informatik.uni-koeln.de/fai/

[11] [Online]. Available: http://systemimager.org

[12] "Metalink - open standard/framework/file format." Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Metalink

[13] "xcat - extreme cluster administration toolkit." [Online]. Available: http://xcat.sourceforge.net/

[14] "Gentoo linux." [Online]. Available: http://www.gentoo.org/

[15] "Python programming language," Python Software Foundation. [Online]. Available: http://www.python.org/

[16] "Twisted matrix labs," WWW. [Online]. Available: http://twistedmatrix.com/trac

[17] “Python/pygridware,” June 2007. [Online]. Available: http://dev.globus.org/wiki/PyGridWare
[18] “Globus toolkit.” [Online]. Available: http://www.globus.org/toolkit/
[19] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, “Web services description language (wsdl) 1.1,” W3C, March 2001. [Online]. Available: http://www.w3.org/TR/wsdl
[20] “Simple object access protocol (soap) 1.2,” W3C. [Online]. Available: http://www.w3.org/TR/soap12-part1/
[21] T. Tsujikawa, *aria2c - The ultra fast download utility*, 1st ed., 07 2009. [Online]. Available: http://aria2.sourceforge.net/