ADEM: An Automation Tool for Application Software Deployment and Management on OSG

Zhengxiong Hou[*], Mike Wilde[+], Xingshe Zhou[*], Ian Foster[+#], Jing Tie[#]
[*]*Center for High Performance Computing, Northwestern Polytechnic University, Xi'an, China*
[+]*Computation Institute, University of Chicago and Argonne National Laboratory, Chicago, IL, USA*
[#]*Department of Computer Science, University of Chicago, Chicago, IL, USA*
houzhx@hotmail.com, zhouxs@nwpu.edu.cn, {wilde, foster}@mcs.anl.gov, tiejing@gmail.com

Abstract

In the grid environment, the problem of application software deployment and management is a major practical challenge for the end-users. Manual operation is error-prone and not scalable to large grids. In this work, we propose an automation tool for Application software DEployment and Management on Open Science Grid: ADEM. On the basis of the grid middleware Globus, it is integrated with pacman. Currently, it can be adaptive to pre-build and dynamic build approaches. NMI B&T system is adopted for the pre-build function. After the application software packaging, ADEM is mainly for automatic deployment, update or removing. The automatic workflow includes automatically getting the available grid sites with their signatures, site signature based automatic deployment or management on a set of grid sites in parallel, automatic dependencies check and integration, automatically preventing some possible errors, automatically getting the results. And the provenance tracking is helpful for the troubleshooting of potential exceptions. Some experiment results on Open Science Grid (OSG) show that ADEM is easy to use, more successful and efficient than manual operation.

1. Introduction

Production grids have emerged as a new generation computational infrastructure, with resource sharing and cooperative work for federated distributed high performance computing. An application is said to be grid enabled when it is able to run on multiple heterogeneous resources comprising a grid. [1] Grid enabled application usually has a coarser level of granularity, needs to have minimal inter-node communication. [2] Most of the grid application needs the support of application software, especially for science and engineering computing. There are many kinds of application software for different disciplines, such as astronomy, astrophysics, cosmology, computational chemistry, computational biology, computational fluid dynamics, computational structural or solid mechanics, materials science, electromagnetism computing, etc. In fact, grid resources are mainly the aggregation of all kinds of traditional computing resources. So, almost all kinds of application software (including the traditional intra-machine or intra-cluster) can be deployed, managed and executed in the grid environment.

Whether open source or commercial application software, the software engineering for the grid usually includes: development, deployment, test and debug, execution. [3] In these phases, execution has attracted the most attention, which usually means scheduling and coordinating execution, using a variety of resources concurrently. But, the application software needs deployment before execution.

The lack of middleware support for application deployment presents a challenge to scientists utilizing computational grid infrastructure. [4] Because, in the grid environment, operating systems, grid middleware and some application software are usually deployed and managed by system administrators. But in many cases, end-users or application administrators have to deploy and manage their domain specific application software on the grid by themselves. In this paper, we define deploying an application on grid as installing and testing the software on a range of different platforms comprising a grid. With the intrinsic distributing, heterogeneous, dynamic and autonomous or across administrative domains characteristics, it is non-trivial for the application software deployment and management in the grid environment. The main problems are as follows:

(1) There is a wide disparity in operating systems, system architectures and runtime environments (CPU, instruction set, file system, software libraries, etc.). Application software should have the right versions and the related dependencies to support different platforms in the grid sites.

(2) The aggregation number of grid sites is usually large and dynamic: dozens, hundreds or even thousands. The grid sites properties are also dynamic. And some grid sites may be uncertain.

(3) The application software has to be deployed and managed remotely, for the machines or clusters are geographically distributed among the different grid sites.

(4) The application software should be deployed and managed by end-users or application administrators with limited resource authority (not a super-user with root permission) in different autonomous administrative domains.

(5) Without an account and permission for a Virtual Organization (VO), the grid sites are unavailable at all.

Manual deployment and management of the application software usually require expertise both about the underlying system platforms and the application. However, to date, this phase is often performed manually, which is both error-prone and does not scale to large grids. For example, in order to deploy an application across 20 grid sites, a user would typically need to login each of the 20 sites sequentially, compiling, linking, installing and testing the software or just do it by batch. [3] And the deployment may raise some licensing and maintenance issues at each site.

So, it is important to conduct the research on how to enable the application software automatic deployment and management for the grid.

In this paper, we propose an automation tool for Application software DEployment and Management on OSG [5]: ADEM. The users can trigger an automatic workflow by ADEM. Currently, it can be adaptive to pre-build and dynamic build approaches. The automatic workflow includes automatically getting the available grid sites with their signatures, site signature based automatic deployment or management on a set of grid sites in parallel, automatic dependencies check and integration, automatically preventing some possible errors, automatically getting the results. And the provenance tracking is helpful for the troubleshooting of potential exceptions.

The rest of this paper is organized as follows: In section2, we introduce the related work in this area. Section 3 is the design and architecture of ADEM for OSG. Section 4 is the implementation for the automatic workflow of ADEM. Then, in section 5, some experiment results on OSG are presented. Section 6 concludes this paper.

2. Related work

Some tools are specifically designed for the installation and management of linux clusters, such as Rocks [8, 6], Cfengine [6], OSCAR [6], LCFGng [6], etc. And there are also some research works about the application software installation, deployment, or management in the distributed or grid computing environment.

On the LHC computing grid, Tank&Spark [7] is a server-client solution that extends the LHC grid application software system. It allows for centrally triggering and controlling software installations at many remote sites within a virtual organization. And gLite Packman [9] can be an alternative to Tank&Spark. But it does not tackle the problem of automatic installation on a farm of grid sites.

For the ATLAS experiment, [10] the main work was to interface CMT [11] and pacman [12] with ATLAS software. Additional tools were developed to extract files in common package formats and write the pacman files.

Quattor [6] is a system administration toolkit for the installation, configuration and management of operating systems and application software for computing fabrics. The configuration management modules are responsible for describing the desired configuration of all the nodes, and the installation management modules are responsible for changing the actual configuration of nodes to the desired state. It mainly supports unix derivatives, such as linux and solaris.

Distributed ant [4] is a system to support application deployment in the grid. It extends the ant build file environment to provide a flexible procedural deployment description and implements a set of deployment services.

Adage [13] is claimed for the automatic application deployment on grids, the main steps include: general application description, deployment planner and deployment plan execution, and application configuration. Currently, the following programming models are supported: MPI, JXTA, CCM (CORBA Component Model) applications.

In the work of gabor kecskemeti and peter kacsuk [14], based on virtual machine, they extended the globus workspace service, including creating virtual appliances for grid services, service deployment from a repository, and influencing the service schedules by altering execution planning services, candidate set generators or information systems.

From industry, rBuilder [15] enables application vendors to combine their application with just enough operating system to create a software or virtual appliance. The electric cloud [16] company provides a distributed build system that re-factors an application's makefiles into parallel workloads executed on dedicated clusters.

Unlike the related work, our work is focused on the automatic workflow for real domain specific application software deployment and management on OSG. It can be adaptive to different build approaches, such as pre-build and dynamic build. With the pre-build function, it is unnecessary to compile the source code on every grid site. There is no additional daemon program. And it can support the execution of grid workflow application by swift [17].

3. The design of ADEM

To solve the above problems for the application software deployment and management in the grid environment, the design of ADEM is described as follows.

On OSG, the automatic deployment and management of application software are based on globus [18] and pacman [12]. They are suitable for the remote task execution and data transfer in the distributed grid environment.

Heterogeneity is a main concern in the grid environment. In ADEM, grid site signature is used for the identification of heterogeneous platforms. Here, we define grid site signature as the configuration features of a platform, which are usually important for the successful installation and running of application software, such as CPU architecture, operating system and its kernel version, glibc, gcc version, and MPI. "Linux-2.6.9-i686-glibc2.3.4-gcc3.4.6-mpich1.2.7" is an example. This is important for choosing the right pre-built application binary packages for the various grid sites. In some cases, the pre-built application software on a given machine can not be migrated to other machines, although they have the same signature. So, we also support the dynamic build approach. For the dynamic build, it is also important for choosing the right version of source codes for the various grid sites. On the basis of pacman and the software repository (including dependency software, library, etc.), the dependencies can be checked and deployed automatically if they are unavailable.

Dynamicity is another main concern in the grid environment. The dynamic grid sites information is fetched on the basis of VORS (Virtual Organization Resource Selector) [19] or ReSS [20]. We adopt a grid sites cache file for the grid sites information. For the old grid sites in the grid sites cache file, ADEM detects them for the detailed information, updates it if it is changed. If an old grid site can not be used, it will be removed from the grid sites cache file. On the contrary, once a new grid site is found, the new information about site name, gateway, application work directory, the path of pacman, site signature, data directory, local resource manager, the path of MPI and available disk space will be automatically fetched and added into the grid sites cache file. Especially, if there is no available pacman or MPI on the new grid site, it will be automatically installed during the updating process. Usually, there is enough disk space for these kinds of application software.

Security or trust model is of particular concern. On OSG, we use the standard globus grid services, there is no additional daemon, no additional port for a user's globus container or opening a host based firewall. Each grid site has a delegated account for each VO, which it belongs to. When executing deployment or management scripts by globus GRAM on the OSG grid sites, an end-user is mapped as the delegated user for the particular VO, which the end-user's certificate belongs to. As described in Figure 1, if some end-users are belonged to the same VO X, we create an individual work directory for every end-user under the same delegated user account on each grid site. The work directories for application deployment and execution are also separated.

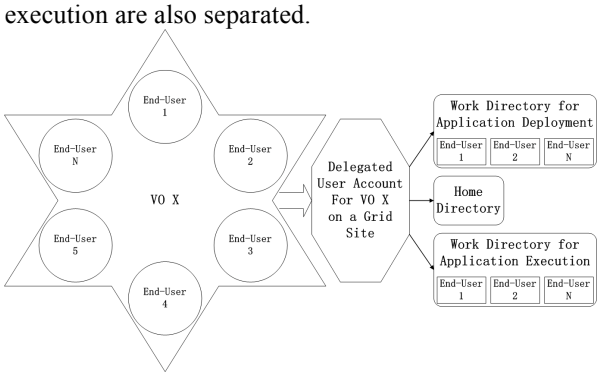

Figure1.The mapping from end-users to the delegated user for a Virtual Organization

According to the design concepts, the logical architecture of ADEM is described in Figure 2.

(a) NMI B&T System [21]. It is for the pre-build function, including the compilation, linking, installation, test and packaging of application software for the heterogeneous platforms. For a new application, it needs to prepare the software source code tarball, and the description about how to obtain, compile, link, install and test it on the NMI B&T system. The end-users or application administrators login the submitting machines of NMI B&T system to submit the software build and test jobs. To improve the efficiency, the users can just prepare the application software for the specific heterogeneous platforms representing the available OSG grid sites which are to be used.

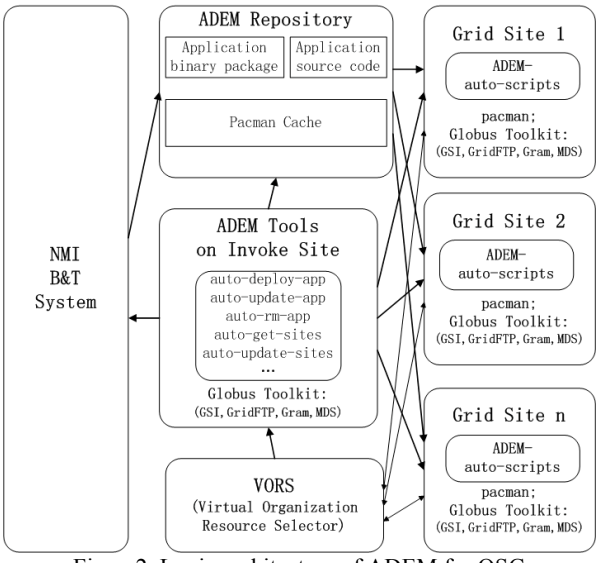

Figure2. Logic architecture of ADEM for OSG

(b) A repository for application software and the possible dependencies packages. The application software package can be source code tarball or pre-built binary package from NMI B&T system. So, there is a matched package for each grid site signature in this repository. A pacman cache can also be placed in it, which contains pacman files describing how to download and install the application software for different grid sites signatures. It is decided by the pacman file to choose the pre-build or dynamic build approach. With the pre-built binary package, the installation of application software just needs an unpacking to the defined work directory in the grid sites.

(c) An invoke site to trigger application software automatic deployment or management to the available grid sites in parallel. Globus toolkit 4+ should be installed on it. It can be any machine with a certificated user for the VOs. The ADEM script tools are downloaded and executed on the invoke site. Then, the automatic workflow for application software deployment or management is triggered on a set of available grid sites within a given VO.

(d) VORS [19] (Virtual Organization Resource Selector). It is the source for the dynamic grid sites information. ReSS [20] can be another option.

(e) The grid sites. Globus toolkit 4+ should be installed on them. GSI is used for the common grid user's certification and security. GridFTP is adopted to transfer the automatic scripts for the deployment and management. GRAM is used for the execution of the scripts on the distributed remote grid sites. If there is no available pacman, it will be automatically prepared by ADEM. The automatic scripts for the deployment or management are generated and transfered from the invoke site to the grid sites in run-time. As described above, besides a specific delegated user's home directory for each VO, there is an individual user space as a work directory for each end-user in the grid sites. And each application has an individual space.

4. Implementation for the automatic workflow of ADEM

Based on globus, ADEM is integrated with pacman for the application software automatic deployment and management on OSG. Currently, it mainly supports unix derivatives. We provide a few commands for the automatic deployment and management functions, mainly including automatic deployment, update and removing. They were implemented by scripting languages, such as Bash and Perl.

We propose a site signature based application software packaging with binary codes or source codes. Only application administrators or a subset of end-users should be entitled to prepare the application software packaging and manage the default application software repository and pacman cache, including adding, removing, modifying or updating application software packages and pacman files. The users can also create their own repositories for application software and pacman cache. If it is not setup while installing the ADEM tool, the default application repository will be used.

```
# Description of the package
description=' Dock Binary for Molecular Dynamics on
x86_64_linux_2.6.9 platform'
# What to download?
platformGE( 'unix' )
downloadUntarzip( 'http://www.***.***.edu/~houzx/pac-cache/
dock6-x86_64_rhas_4.tar.gz' ,' DOCK_TAR' )
# How to install
# Just unpack for install
# How to test
cd('$DOCK_TAR/install/test')
shell('make test > make-dock-test.log')
cd()
```

Figure3. dock-linux_2.6.9-x86_64.pacman file

The instructions on how to fetch and install software are described in a pacman file, i.e. dock-linux_2.6.9-x86_64.pacman is an example for dock application with a binary codes package (Figure 3). The operating system and CPU features are taken from the grid sites signature to name the pacman file. When necessary, the glibc or gcc can be dealt with as dependencies.

On the invoke site, after the application software packaging, any certificated common user can trigger an automatic workflow to deploy or manage a given application software to the available grid sites in parallel, and get the results automatically. The specific

commands include “auto-deploy-app”, “auto-update-app” and “auto-rm-app”. They need some parameters, such as VO, sites file, application software name. The VO and sites file can be automatically fetched. And ADEM will automatically check the repository to show the packaged application software.

The general automatic workflow for the application software deployment and management on OSG is described in Figure 4.

For example, the key steps of the automatic workflow for application software deployment are as follows:

Step1: Get the available grid sites automatically and dynamically.

Based on the dynamic grid sites information from VORS [19] or ReSS [20], it can discover the available grid sites and automatically generate the available grid sites file for a given VO, which the end-user belongs to. A cache file for the grid sites information is adopted to improve the performance. And it will be updated after every usage. If it’s the first time to get the available grid sites for an end-user, an individual work directory for application deployment and execution will be created respectively. To support the execution of grid workflow by swift [17], a sites.xml file will be automatically generated for the specific users.

Step2: Execute the application software automatic deployment on the selected grid sites in parallel.

With a loop for launching deployment, it automatically generates the deployment script for each grid site and transfers them to the grid sites by GridFTP. Then they are executed in parallel in the background. Pacman is integrated in the automatic deployment script, which is used to transport, and install application software to the grid sites. With the grid sites signatures, ADEM can choose the right application software packages for different grid sites platforms from the repository. For the pre-build approach, it just needs to unpack the binary codes package for the installation. For the described dependencies in the pacman file, they will be automatically checked in the grid sites. If there are no available dependencies, they will be installed automatically. The experiences of dealing with some errors can also be added into the pacman file. So, it can automatically prevent some possible errors. And the standard output and standard error information for the deployment on each grid site will be returned to the invoke site.

Step3: Check the results automatically.

The deployment results on all of the available grid sites will be automatically fetched. If there are some errors on some grid sites, the specific sites can be automatically cleaned and re-deployed from scratch for one time.

To support the execution of grid workflow by swift [17], a tc.data file will be automatically created to record the successfully deployed application software and the path of the executables.

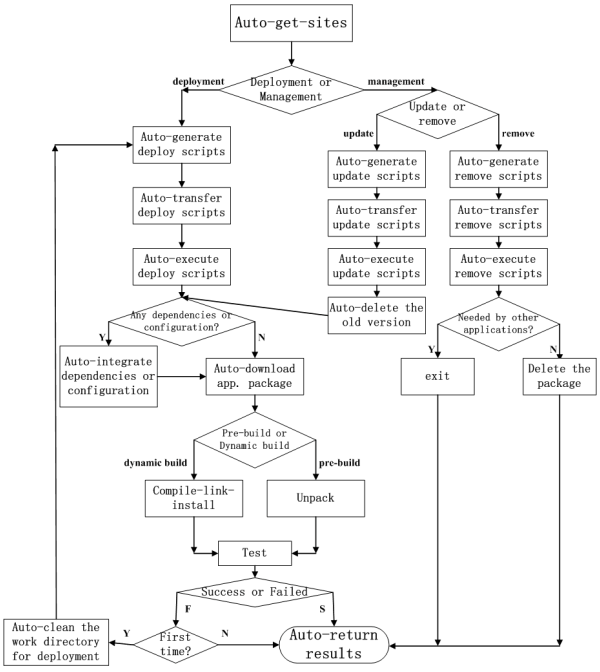

Figure 4.The automatic workflow of ADEM

The automatic workflow for application software management is similar with the deployment. In the invoke site, a common certificated user can use the script tools “auto-rm-app” and “auto-update-app” to trigger automatic removing and update for a given application software on a set of grid sites. And they are also executed in parallel in the background. The management results can also be checked and fetched automatically.

After the completion of automatic deployment or management for the application software, all of the executed commands will be automatically recorded in a log file. All of the results of the dynamic available grid sites, application deployment or management will be automatically recorded in related log files. And all of the standard output, standard error information will also be automatically recorded. So, the logs can be used for the provenance tracking and are helpful for the troubleshooting of some potential exceptions in the grid environment.

5. Experiments

Open Science Grid [5] is a distributed computing grid for data-intensive research in US and beyond.

Altogether, there are more than 70 grid sites and 30 Virtual Organizations. We did the experiments mainly within the osg and osgedu VO.

In the experiments, more than 10 applications were tested to be automatically deployed and managed on about 20 available grid sites. Almost all of the grid sites adopt linux as the operating system. And the main two CPU architectures are i686 and x86-64. The application domains included molecular dynamics, biology, computer science, and so on. Dock [22], Nab [23], Blast [24], etc. were our typical application software. In the invoke site, the application software automatic deployment and management on a set of grid sites were triggered by a certificated common grid user. The machine for the invoke site provided dual core Intel Pentium® CPU 2.80GHz with 3GB of RAM.

5.1 Success rate and Scalability

Manual deployment of the application software is error-prone and not scalable to large grids. In fact, if the application software has to be built on each grid site, there are some errors with different extent for most of the application software. Most of the installation failures are due to the dependencies or compiling errors.

With the site signature based pre-build approach, we do not need to compile the source code on the grid sites. The error-prone problem was solved almost perfectly. The success rate was greatly improved because we could get the right version of the application's binary code package for the specific platform, which was pre-built on the well prepared machine with the same signature from NMI B&T system [21].

For the available 20 grid sites within OSG VO, Figure 5 shows the comparison of success rate between pre-build based application software automatic deployment with ADEM and manual deployment. Several failures were due to the problem of pacman. Especially, because Blast application package was already a binary package, it did not need a pre-build. So, the success rates were the same with ADEM and manual deployment, which were both 100%.

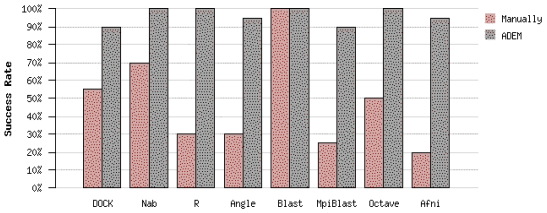

Figure5. Comparison of success rate between ADEM and manual deployment for 20 grid sites

For a set of available grid sites, no matter what size it is, the automatic deployment or management for application software is launched one by one by a loop and executed in the grid sites in parallel in the background. In theory, it is easily scalable to large scale grid sites. But, it will be overloaded if there are hundreds or even thousands of grid sites accessing the application software repository concurrently. So, to ensure the scalability, we tend to use repository mirrors and caches to share the workloads, if there are very large scale grid sites.

5.2 Time Cost

The automatic deployment of application software to a set of grid sites includes a launching and an execution process. In the launching process, it generates the automatic deployment scripts for each grid site. Then the automatic deployment scripts are transported to the remote grid sites and executed in parallel in the background.

Figure 6 is the average launching time for application automatic deployment on 20 grid sites. It was just about 2 seconds. For the launching time, there was a very little difference for different application software. Due to the execution loop for a set of grid sites, the launching time of automatic deployment for application software to a set of grid sites usually increases with the number of grid sites. But it is relatively very fast, i.e. for the simulated 100 grid sites it was just about 8 seconds.

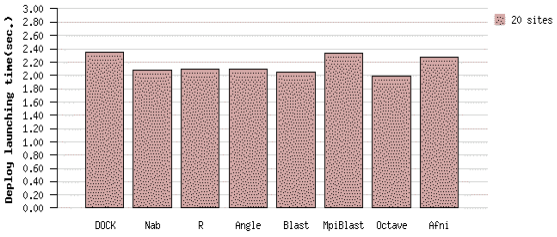

Figure6. Application automatic deployment average launching time for 20 grid sites

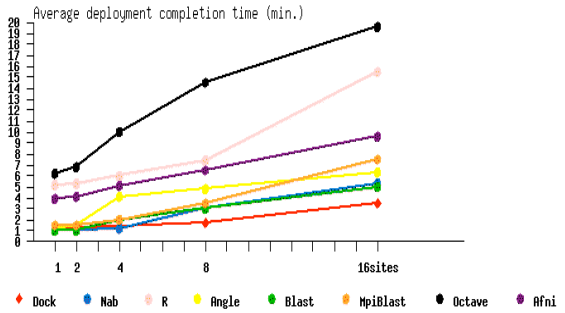

Figure7. Pre-build based application automatic deployment average completion time statistics

For the deployment completion time to a set of grid sites, it includes the launching time, the time for the execution of installation and tests. The average completion time of the pre-build approach is shown in Figure7. For the pre-build approach, it usually just needs to unpack the binary package for installation. Sometimes, it needs to install dependencies when necessary. There was an obvious variation among different grid sites. Although we could deploy the application on all of the available grid sites in parallel, the whole deployment completion time was decided by the slowest grid site. So, the average deployment completion time increased with the grid sites number to some different extent. For different application software, the average deployment completion time is usually different. It is mainly dependent on the unpacking time and the network performance.

However, compared to the manual deployment or dynamic build on each grid site, the pre-build based automatic deployment gains a much better performance. For example, if a user login each grid site to deploy DOCK, it costs about 292.8 minutes for the 16 grid sites, provided that there was no error. With the dynamic build on each grid site, for the successful deployment of DOCK application to 16 grid sites, the average time cost was about 45 minutes. With the pre-build approach, the average pre-build time was 15 minutes, and the average successful deployment to 16 grid sites was 3.5 minutes, so the whole time cost was just about 18.5 minutes.

For the application software automatic management functions, such as removing and update, the launching process is similar with that of the deployment. The completion time for removing is also very fast if compared with the deployment completion time. And the completion time for update is about the sum of removing and deployment. For example, the time cost statistics on 16 grid sites is revealed in Figure 8. It took a comparatively long time for the pre-build of octave application, which generated a binary package with the size of more than 200MB. While blast did not need a pre-build, because it was already a binary package.

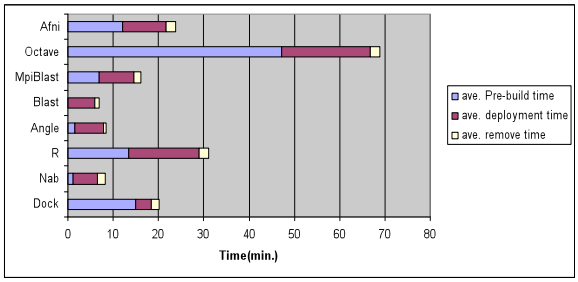

Figure8. Pre-build based time cost statistics on 16 grid sites

After the deployment, the application software is utilized for the real grid applications. We have executed large scale parameters sweep applications on the basis of the deployed applications, such as DOCK, and Blast.

6. Conclusion

We proposed an automatic workflow schema for Application software DEployment and Management on OSG: ADEM. Although we prefer to use the pre-build approach, in some cases, the dynamic build approach is also necessary. The pre-build function is implemented on the basis of NMI B&T system. The globus grid middleware provides the basic grid services, and we integrate ADEM with pacman. It can get the grid sites automatically and dynamically, trigger automatic deployment or management of a given application software on a set of available grid sites in parallel. And the results are automatically fetched. Our experiment results on OSG show that ADEM is feasible and has good effect. It significantly reduces manual efforts for the deployment and management of application software. It is easy to use, much more successful and faster than manual operation. And it can be easily scalable to large grid sites.

The future works may include implementation for large scale grid sites, virtual machine based application automatic deployment and management, and a web interface for the users.

7. Acknowledgement

This research is supported in part by NSF grant OCI-721939 and the U.S. Dept. of Energy under Contract DE-AC02-06CH11357. It was made possible by the resources of the Open Science Grid, the TeraGrid, and the Build and Test System of the National Middleware Initiative. We thank Glen Hocky for testing and feedback.

The following government license should be

removed before publication.

8. References

[1] P.V. Coveney, R.S. Saksena, S.J. Zasada, et. al. The Application Hosting Environment: Lightweight Middleware for Grid-based Computational Science. [J] Computer Physics Communications 176(2007) 406-418.

[2] Arun Krishanan. A Survey of Life Sciences Application on the Grid. [J] New Generation Computing, 22(2004) 111-126. 2004.

[3] David Abramson. Applications Development for the Computational Grid. Frontiers of WWW Research and Development (APWeb 2006), LNCS 3841, pp. 1-12. 2006.

[4] Wojtek Goscinski and David Abramson. Distributed Ant: A System to Support Application Deployment in the Grid. Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04). 2004.

[5] OSG (Open Science Grid). http://www.opensciencegrid.org/, 2009.

[6] R. Garc´ıa Leiva, M. Barroso L´opez, et al. Quattor: Tools and Techniques for the Configuration, Installation and Management of Large-Scale Grid Computing Fabrics. [J] Journal of Grid Computing (2004) 2: 313–322.

[7] Roberto Santinelli, Flavia Donno. Installing and Configuring Application Software on the LHC Computing Grid. Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05). 2005.

[8] Rocks. http://www.rocksclusters.org/wordpress/, 2009.

[9] Alien/gLite packman. http://glite.web.cern.ch/glite, 2009.

[10] Simon George1, Christian Arnault, Michael Gardner, et al. Automated Software Packaging and Installation for the ATLAS Experiment. 2003.

[11] CMT. http://www.cmtsite.org/, 2009.

[12] Pacman. http://physics.bu.edu/~youssef/pacman/, 2009.

[13] Adage. http://www.irisa.fr/paris/ADAGE/, 2009.

[14] Gabor Kecskemeti, Peter Kacsuk. Automatic Service Deployment Using Virtualization. 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing. 2008.

[15] rBuilder. http://www.rpath.com/corp/products, 2009.

[16] Electric cloud. http://www.electric-cloud.com, 2009.

[17] Zhao Y., Hategan, M., Clifford, B., Foster, I., Wilde, M. et al. Swift: Fast, Reliable, Loosely Coupled Parallel Computation. IEEE International Workshop on Scientific Workflows 2007.

[18] Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. [J] Journal of Computer Science and Technology, 21(2006) 513-520. 2006.

[19] VORS (Virtual Organization Resource Selector). http://vors.grid.iu.edu/cgi-bin/index.cgi,2009.

[20] Resource Selection Service (ReSS). http://twiki.grid.iu.edu/bin/view/ResourceSelection/WebHome, 2009.

[21] Andrew Pavlo, Peter Couvares, Rebekah Gietzel, et. al. The NMI Build&Test. Laboratory: Continuous Integration Framework for Distributed Computing Software. 20th Large Installation System Administration Conference (LISA '06). 2006.

[22] Dock. http://dock.compbio.ucsf.edu/DOCK_6/index.htm, 2009.

[23] Nab. http://www.scripps.edu/mb/case/casegr-sh-3.2.html, 2009.

[24] Blast. http://blast.ncbi.nlm.nih.gov/Blast.cgi, 2009.