Dynamic Runtime Environments for Grid Computing

D. Bayer[1], T. Bhimdi[1,2], G. Oechsler[1], F. Orellana[3], A. Wäänänen[3], B. Kónya[4], and S. Möller[1]

[1] University of Lübeck, Institute for Neuro- and Bioinformatics, Lübeck, Germany
[2] George Mason University, Fairfax, Virginia, United States
[3] Copenhagen University, Niels Bohr Institute, Copenhagen, Denmark
[4] University of Lund, Department of Physics, Lund, Sweden
email: [bayer,moeller]@inb.uni-luebeck.de
phone: +49 451 500 5514, *fax:* +49 451 500 5502

Abstract

In a grid computing context the execution of jobs on remote machines of collaborating institutes often requires the provisioning of more than mere CPU time. Besides libraries and utilities distributed with the operating system in question, further software may be required by the jobs and in such cases need to be installed prior to the job's execution. In grid computing this problem is aggravated by decreased personal contacts among collaborators.
Currently, Virtual Organisations of existing production grids typically agree on a set of runtime environments that participating sites install manually. The challenge is to automate this process, thus easing the burden of the site's maintainers and allowing grid-wide software updates with immediate effect. This paper presents an RDF based schema for the description of runtime environments and the implementation of a service for their *automated* and *dynamic* installation.

Keywords: grid computing, runtime environment, automated installation

Abbreviations: ARC (Advanced Resource Connector), GIS (Grid Information System), LRMS (Local Resource Management System), RDF (Resource Description Framework), RE (Runtime Environment), RER (Runtime Environment Registry), VO (Virtual Organisation)

Availability: `http://dre.knowarc.eu`

1 Introduction

Computing has become an integral part of several fields of science and engineering, comprising *in silico* simulations, data mining or general data analysis. Other progress has been made in data acquisition. Databases beyond giga- and terabytes in size are no longer an exception. For some research questions, e. g., in the biological sciences, it is especially the combination of information sources

from which new insights are gained. This led to the employment of agent [12] and workflow technologies [16].

For a grid to be successful, jobs should find well-equipped hosts easily. The provision of runtime resources, i. e., utilities, libraries, databases or tools for their accession should be prepared and tested before a job is executed. The Advanced Resource Connector (ARC) middleware [4] of NorduGrid [3] allows the specification and seamless usage of so called Runtime Environments (REs) in grid jobs. Grid jobs are directed to exactly those sites at which the requested REs are available.

For today's grid sites, the actual selection and installation of the software is still left to the human maintainers. In the NorduGrid, the information needed for installation is provided by the so-called Runtime Environment Registry (RER) [11]. With an increased acceptance of Grid computing across scientific disciplines and the associated diversification of software to be installed and maintained. This manual process imposes an unbearable amount of work on the site's maintainers. Thus missing or outdated installations of REs often render resources unusable.

We speak of *automated* installations if the RE installation runs unattended after being initiated. An installation is *dynamic* if it is not started by a human but by the acceptance of a grid job requesting the particular RE. This paper describes a generic approach for an automated installation of software or its updates, implementing the dynamic provision of runtime environments for the ARC grid middleware.

Using this scheme a Virtual Organisation (VO) will no longer need to contact the grid site's administrators to install their software and its updates. They will simply manage their own machine-readable software catalogue which is used by the middleware to deploy the needed REs. Virtualisation, although not strictly required, is an important part of this concept, as it allows to easily set up an environment that is specifically tailored for the job. It also provides extra security for the site providing the resources.

2 Methods

The here laid out principle was implemented in Perl. No ARC specific libraries were used. We are using the the resource description framework (RDF) technology [17] to describe REs. This description is read with help of the Redland RDF library[1]. It is the only required non-standard Perl module. The RDF files themselves are distributed by an http server. For the human observer the same data is offered reformatted as HTML. None of the changes to the middleware are visible to ordinary clients.

[1] http://librdf.org/

3 Results

The ARC middleware does not contain a special broker service. Instead the brokering is done by the user's grid client. The user specifies which REs the job needs and the client searches in the Grid Information System (GIS) [8] for sites announcing the requested REs. Upon submission, the job's specification is sent to the selected grid site's Grid Manager [7].

With the introduction of the dynamic installation of runtime environments, the Grid Manager needs to distinguish between REs that are *installed*, those that are *installable* and those that are *not eligible*. To decide on the eligibility of a runtime environment for installation on a site, the Grid Manager first requests details about that runtime environment. This comprises technical information, i. e., the installation means and dependencies on other runtime environments. But also the field of application or the maintainer of the package may be important. Site maintainers can constrain the dynamically installable packages by these RE's attributes.

3.1 The RDF Schema used to describe REs

The automation of the installation process requires its formal description. Most importantly this comprises the dependencies of the particular RE. The collection of these formal descriptions is called the *Catalogue*. It is represented as an RDF document. The RDF schema used contains four different types of entries. These are referred to as *MetaPackage*, *Package*, *Tag* and *BaseSystem*.

The MetaPackage nodes describe the REs. In the previous system, informal specifications of each commonly used RE within the ARC community were collected in the RER. Its entries contain the RE's ID, its version information and a short description. For installation instructions it links to regular web pages describing the manual installation procedure. In the Catalogue this information is stored in MetaPackage nodes. To ease the specification of constraints, MetaPackage nodes can have an arbitrary number of associated Tag nodes, which describe the MetaPackage in more detail.

The BaseSystem nodes are used to distinguish the operating systems of the worker nodes. The last type of nodes is the Package node. The Package nodes describe how to deploy a MetaPackage on a system with a specific BaseSystem. So each MetaPackage links to at least one Package which in turn links to exactly one BaseSystem.

An example is given in Fig. 1. It describes how to deploy the RE named APPS/BIO/WEKA on Debian Sid using the tar packages `weka-3.4.8.tar.gz` [18] and `jre_1.5-1.tar.gz`. The figure only shows the most basic nodes. Some descriptions and constraints are omitted. The *TarPackage* node in the figure is a special subclass of Package.

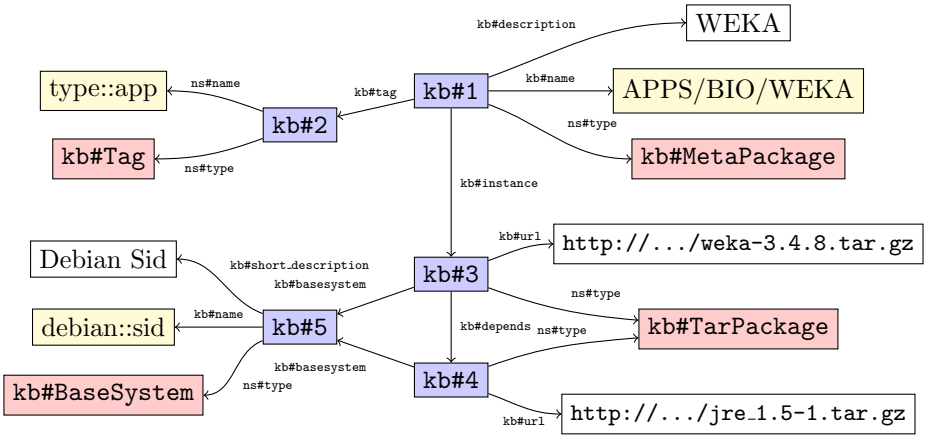

Fig. 1: Subset of RDF triplets of a Catalogue describing the runtime environment WEKA and its installation. `kb#1` represents the software (MetaPackage). `kb#3` offers its installation from a tar file on Debian Sid (`kb#5`), if Java (`kb#4`) is also installed. These tar files can be downloaded from the given URLs.

3.2 Deployment

The Catalogue is used by a service called the *Janitor*. It manages REs on behalf of the Grid Manager or the site's administrator and provides an information system for the states of the REs.

The Janitor has three different interfaces. For the

Grid Manager: An interface used to register jobs and to trigger the automatic installation of REs just before job execution.

Grid Infosystem: An interface used to query the list of supported REs.

Site Admin: An interface used for removing unused REs and retrieving state information.

The site's administrator determines which Catalogues the Janitor is allowed to access. He also configures which base system the worker nodes are using. Additionally the administrator can specify for each RE individually if it is eligible or not.

With this information given, REs are automatically installed by the Janitor before a job is executed. An interaction diagram for this case is given in figure 2: After a job's submission, the Grid Manager registers the job with the Janitor (figure 2, step 2). The Janitor checks if all requested REs are installed, deployable or unavailable. If they are deployable the Grid Manager asks the Janitor to do so (step 3); otherwise the job fails. Upon successful installation, the Grid Manager calls the LRMS helper (step 4) which prepares the job for submission to the LRMS. For this it needs to know how to use the REs. This information is retrieved from the Janitor (step 5). Finally, when job execution has finished, the job is unregistered from the Janitor (step 7).

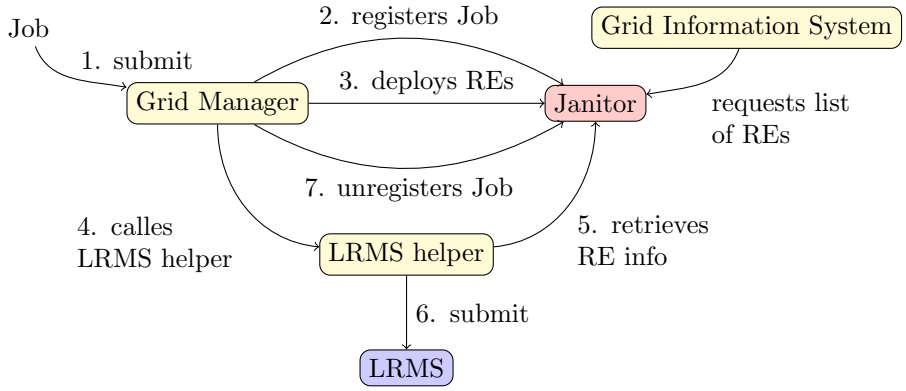

Fig. 2: Interaction of the Janitor with the ARC middleware. Upon submission (1) a job is registered with the Janitor (2), and the needed REs are installed if necessary (3). The LRMS helper is started (4) which retrieves information on the used REs (5) and submits the job to the local batch system (6). If the job is done it is unregistered (7). The Grid Informations System periodically requests the list of available REs.

Another component interacting with the Janitor is the Grid Information System. It periodically polls the Janitor to get the current list of available REs.

The RE's installation process itself highly depends on the kind of Package which is to be installed. A well-suited approach for small grid sites is to use a shared directory for deploying software to the worker nodes. In this case simple TAR packages can be used. To install such a package, the Janitor (i) creates a subdirectory on the shared storage, (ii) extracts the tar-file containing the software into it and (iii) executes a bundled `install` script.

3.3 Overhead introduced by the Janitor

Basically, the operations the Janitor performs can be categorised into two groups. A first group of operations only touches the metadata on installed REs and registered jobs. A second changes the REs themselves. Operations which belong to the latter can take an arbitrarily[2] long time. For each job three operations of group one and maybe one operation of group two are needed.

Often an RE will be used by many jobs. So the vast majority of operations performed by the Janitor belong to the first group. The simplest way to use the Janitor is to start a new process for each operation. Implemented this way and running on a recent system[3] the Janitor needs about 900 ms for operations of this group. Most of this time is spent during startup. Another possibility is to

[2] E. g., to install an RE, files from remote sites must be downloaded.

[3] AMD Athlon X2 4200+ processor

run the Janitor as a daemon and fork a child for each operation. Implemented this way, it takes about 40 ms to perform a single operation.

4 Discussion

Applicability for other grid middlewares and site configurations The concept presented is not dependent on ARC-specific features. The Janitor has a generic interface which can be easily integrated with other grid middleware stacks.

The ARC middleware is known as light-weight and non-invasive, supporting many different batch systems and network setups. Thus, a framework for dynamic REs used by ARC has to be very flexible, too. The formal descriptions presented consequently allow to describe dynamic REs for very different site setups, ranging from small-scale grid sites using a shared network folder for software deployment to large sites using virtualisation technologies and job-specific images.

Package management The concept to have libraries and tools added dynamically with respect to dynamically specified dependencies has been implemented for many years in the build daemon infrastructure of the Debian Linux distribution [14]. Any Debian developer can submit new software packages that are eventually built automatically for the 11 supported platforms. The here presented work extends this principle for arbitrary grid computation. The Debian distribution is much respected for this achievement and to indicate such may foster the acceptance of dynamic runtime environments in the grid community.

Software Catalogues Our approach is related to the Configuration Description, Deployment, and Lifecycle Management (CDDLM) specification [2] of the Open Grid Forum. But being designed to deploy whole grid services, this specification is considerably more complex then the Catalogue, which only describes how to install a selection of libraries or programs.

Virtualisation The use of virtualisation provides the job with an environment which is completely different from the installed operation system on the worker node. The most basic form of virtualisation is the usage of a chroot environment. But in this case the separation of the job-specific environment from the system of the worker node is weak. The job directly uses the same kernel and only the filesystem is virtualised. So it can see and interact with processes running outside the chroot environment. Easy ways to break this kind of virtualisation are well known [15]. Consequently, a chroot environment is usefull for jobs which only need specific libraries and are trusted not to try to break into the system environment. E. g., the Debian build daemons use chroot environments.

Over the last years multiple new virtualisation technologies were developed. They are superior to chroot as they create a complete virtual system. These can be used to create sandboxes which reduce potential harm by malicious jobs.

This is especially important if dynamic REs are used and third party software is installed automatically. One of these virtualisation technologies is Xen [1], which uses paravirtualisation, thus yielding a negligible overhead. These technologies were already used to provide grid jobs with hand-crafted virtual environments (e. g., [5, 6]).

In general, images for virtual machines contain the installation of the operating system and the needed additional software. Creating them is possible by remote-controlling the bootstrap procedure of the operating system. This has been achieved by projects like, e. g., FAI [10], several years ago. For the Janitor, this was implemented in a prototype.

An anticipated future scenario is to let grid sites use virtualisation technologies to automatically create job-specific images. In the Catalogue this can be described by a special subclass of Package and the BaseSystem entry linking to a basic disk image. To create a specific image for, e. g., Debian, a copy of the basic image is mounted in a chroot environment and packages provided by the Debian distribution are installed by Debian-specific means. As redoing this for every job is much to expensive, mechanisms for caching [9] should be used.

5 Conclusion

By using the here presented framework, a VO after being accredited at a grid site will no longer be required to ask the site's administrators to install software but simply manage their own Catalogue. Thus, the process of deploying updates to all grid sites is simplified, the availability of grid resources to the user improved and the potential userbase increased [13].

Acknowledgements

Many thanks go to Alexandr Konstantinov, Péter Stefán and Ferenc Szalai for their comments. This work is funded by the EU FP6 project "KnowARC".

References

1. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.
2. CDDLM Working Group. Configuration description, deployment, and lifecycle management – smartfrog-based language specification, 2005. GFD 51, http://www.ogf.org/documents/GFD.51.pdf.
3. P. Eerola, B. Kónya, O. Smirnova, T. Ekelof, M. Ellert, J. R. Hansen, J. L. Nielsen, A. Waananen, A. Konstantinov, and F. Ould-Saada. Building a Production Grid in Scandinavia. *IEEE Internet Computing*, 7(4):27–35, 2003.
4. M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. Advanced Resource Connector middleware for lightweight computational Grids. *Future Generation Computer Systems*, 23(2):219–240, 2007.

5. K. Keahey, K. Doering, and I. Foster. From Sandbox to Playground: Dynamic Virtual Environments in the Grid. In *5th International Workshop on Grid Computing (Grid 2004), Pittsburgh, PA, November 2004*, pages 34–42, 2004.
6. K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid. *Scientific Programming*, 13(4):265–275, 2005.
7. A. Konstantinov. *The NorduGrid Grid Manager and GiridFTP Server: Description and Administrator's Manual*, 2007. [NORDUGRID-TECH-2].
8. B. Kónya. *The NorduGrid/ARC Information System: Technical Description and Reference Manual (v0.9)*, 2007. [NORDUGRID-TECH-4].
9. I. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo. Vmplants: Providing and managing virtual machine execution environments for grid computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 7, Washington, DC, USA, 2004. IEEE Computer Society.
10. T. Lange. Fully automatic installation. http://www.informatik.uni-koeln.de/fai/.
11. J. Lento and O. Tourunen. Runtime environment registry. http://www.csc.fi/grid/rer/concept.phtml.
12. E. Merelli, G. Armano, N. Cannata, F. Corradini, M. d'Inverno, A. Domse, P. Lord, A. Martin, L. Milanesi, S. Möller, M. Schroeder, and M. Luck. Agents in bioinformatics, computational and systems biology. *Briefings in Bioinformatics*, 8(1):45–59, 2007.
13. S. Möller, D. Bayer, D. Vernazobres, A. Gebhardt, and D. Eddelbuettel. Scientific Grid Computing via Community-Controlled Autobuilding of Software Packages Across Architectures. In *NETTAB*, Pisa, Italy, 2007.
14. R. Murray. buildd: Debian package auto-builder. http://buildd.debian.org/.
15. Simes. How to break out of a chroot() jail, 2002. http://www.bpfh.net/simes/computing/chroot-break.html.
16. R. D. Stevens, H. J. Tipney, C. J. Wroe, T. M. Oinn, M. Senger, P. W. Lord, C. A. Goble, A. Brass, and M. Tassabehji. Exploring Williams-Beuren Syndrome Using $^{\text{my}}$Grid. *Bioinformatics*, 20:i303–i310, 2004.
17. W3 Consortium. Resource Description Framework, 2004. http://www.w3.org/RDF/.
18. I. H. Witten and E. Frank. *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.