# Operational Grid tools developed at SCL

V. Slavnić, B. Acković, D. Vudragović, A. Balaž, A. Belić
Scientific Computing Laboratory
Institute of Physics Belgrade
Pregrevica 118, 11080 Belgrade, Serbia
http://www.scl.rs/

*Abstract* — **A number of tools are used in everyday Grid operations and operational support by site administrators in order to allow efficient and reliable management of Grid services and their full availability to end users. Such tools perform various functions, from monitoring and problem diagnostics to site installation, configuration and management. In the Scientific Computing Laboratory (SCL) of the Institute of Physics Belgrade we have developed several tools useful for maintenance and management of gLite-based Grid sites. Some of these tools can be also used on general purpose Linux-type clusters. Here we report on the SCL tools and scripts most used in everyday Grid operations.**

***Keywords- Grid; Operations; Scripting; SCL; gLite; Linux; DWARF; gFinger;***

## I. INTRODUCTION

The EGEE [1] project and many of the regional Grid projects and communities use the gLite [2] middleware as a basis for distributed research e-Infrastructures maintained and used by the corresponding projects. Thus, gLite represents one of the major middleware stacks used today. In particular it is used by the SEE-GRID-SCI [3] project. Although widely used, installation, maintenance and everyday Grid operations of a successful gLite resource center (Grid site) are not an easy task due to the complexity of its structure, interdependencies of many of its components and services, as well as the fact that still some components are not ready for large-scale production use, requiring continuous overseeing and occasional manual interventions.

Site administrators are responsible to maintain the committed Grid resources available to supported user communities, to resolve all operational problems identified by the deployed monitoring tools, or diagnosed by the users. In fact, site administrators managing large clusters are usually deploying customized or home-made tools for monitoring their resources, and write custom scripts or use other specific mechanisms to automatize the tasks that have to be performed regularly on many nodes, or during some operational tasks (e.g. installation of updates on all nodes, execution of a specific command on all nodes). The good collaborative practice is to make such custom tolls available to other interested site administrators.

The Scientific Computing Laboratory (SCL) [4] of the Institute of Physics Belgrade has developed several very useful tools and scripts for managing a gLite-based Grid site. All of them are used locally at SCL for managing two large Grid sites (AEGIS01-IPB-SCL and AEGIS07-IPB-ATLAS) and all core Grid services deployed. In this paper we present all such tools and scripts, which are available for download from our SVN and RPM repositories.

## II. SCL TOOLS

Section *A* presents scl-scripts [5], developed and used for simplifying large Linux cluster management. In Section *B* we present gFinger [6], tool for extracting information about the local mapping to Unix pool accounts of Virtual Organization Management Service (VOMS) users on various Grid services. The package repository manager DWARF [7] is described in the Section *C*, while in Sections *D* through *L* give short descriptions of the custom scripts used in everyday operations: scl-bdii-conf [8], scl-wms [9], scl-clean-scratch [10], scl-generate-pool-accounts [11], scl-generate-users [12], scl-network-sleep [13], scl-sensors [14], scl-jobs [15], and scl-tests-status [16].

### A. *scl-scripts*

The script set of scl-scripts [5] is developed in the BASH [17] scripting language. It is intended to simplify the management of medium or large computer clusters. Using these scripts it is possible to execute the same command on all nodes or to distribute the files on many nodes in an automated way. The set scl-scripts consists of five scripts and one configuration file per group of nodes or servers.

The configuration file scl-nodes is a list of fully qualified domain names (FQDN) of nodes that will be used for scl-scripts, i.e. it represents the set of nodes where the scripts will perform their operations, one by one.

The first script, scl-ssh, can be used to generate public and private RSA [18] and DSA [19] keys, and to exchange public keys among the nodes specified in the configuration file. These keys are used for Secure Shell [20] (SSH) communication on all nodes listed in scl-nodes configuration file. Administrators are supposed to start this script only once, when the scl-scripts are installed. After that, all other scripts can work without prompting for password each time one tries to access each node in a cluster. It is possible to use scl-scripts even without running scl-ssh, but in that case a user will be asked for a password each time any of scripts is executed. Not only that – without properly configured key-based authentication, password is required for each node separately, which would

make practical use of scripts exceedingly slow for larger number of nodes.

Second script, scl-exec, is able to execute a specified command using SSH on all nodes listed in scl-nodes files. The syntax is the following:

```
[root@ce-atlas ~]# scl-exec date

wn01.ipb.ac.rs:
Tue Nov 10 13:23:15 CET 2009

wn02.ipb.ac.rs:
Tue Nov 10 13:23:15 CET 2009

wn03.ipb.ac.rs:
Tue Nov 10 13:23:15 CET 2009
```

It is possible to run more than one command in the line using the quotes (“”), like in this example:

```
scl-exec “cat /var/log/messages | grep error”
```

This command will show all lines that contain word “error” from /var/log/messages on all nodes listed in scl-nodes. Note that the use of quotes is essential here, since in this way the specified command will be executed on each node separately, and we will see the output sorted by the node, like in the example with the "date" command. Without the use of quotes, the command "cat /var/log/messages" would be executed on all nodes, this huge output would be transferred to the node where the scl-exec command is initiated, and only then the grep would be performed. On one hand, this would result in a huge network traffic, and on the other hand one would not get the formatted output as expected, i.e. the information on the node where the specific error is found would be lost.

The other three scl-scripts perform operations with files. The basic one, scl-scp, copies a specified file to all nodes listed in the scl-nodes configuration file using secure copy (secure transfer of files using the SSH protocol). The command:

```
[root@ce-atlas ~]# scl-scp site-info.def  /root/

site-info.def     100%     21KB     20.8KB/s 00:00
site-info.def     100%     21KB     20.8KB/s 00:00
site-info.def     100%     21KB     20.8KB/s 00:00
```

would copy local siteinfo.def file from the current directory to /root directory on all nodes listed in scl-nodes. With scl-scp command it is not possible to copy folders, just individual files.

Two remaining scripts, scl-pull and scl-push, are designed to copy different files to or from nodes listed in scl-nodes configuration file. In the example

```
scl-push /root/nodes-network/ /etc/sysconfig/network
```

the specified command will copy the file from the stated folder /root/nodes-network/ matching the node name listed in scl-nodes file to each node. If the file scl-nodes contains entry wn01.ipb.ac.rs, and the corresponding file named wn01 (with the domain stripped) exists in the directory /root/nodes-network, it will be copied to wn01.ipb.ac.rs node and saved there as /etc/sysconfig/network, and so on for each node present in the configuration file. The pull command,

```
scl-pull /etc/sysconfig/network /root/node-network/
```

will collect files /etc/sysconfig/network from all nodes listed in the scl-nodes file and store them into the specified directory /root/node-network/, and name each file according to the node name, with the domain name stripped.

For larger and more complex computer systems, set of scl-scripts could be easily modified to handle more than one group of nodes. Administrators can simply copy the configuration file and scripts scl-nodes, scl-exec, scl-scp, scl-push and scl-pull into new instances with different names, and modify the new scripts for use with the new configuration files. For example, one can have two clusters, one with 32bit nodes and one with 64bit nodes (which is our use case). One set of scl-scripts can be used for all nodes (32bit, 64bit, as well as all servers and core services), a second set could contain only 32bit nodes, third set only 64bit nodes, and fourth set only core service nodes.

All scl-scripts are provided as an RPM [21] package which will install scripts in /opt/scl/. Configuration files are located in /opt/scl/etc, while scripts are located in /opt/scl/bin, which has to be added to the $PATH in order to be easily used.

### *B. gFinger*

gFinger [6] is a command-line tool developed at SCL, providing information on local VOMS mapping of users authenticated by digital certificates on various Grid services, such as Computing Element (CE), Workload Management System (WMS), Storage Element (SE), etc.

When a request from a client for access to some Grid service is received, the Globus gatekeeper (or another Grid-aware daemon) attempts to map the user to the corresponding local Unix pool account in /etc/grid-security/grid-mapfile file. This file is combination of /etc/grid-security/dn-grid-mapfile and /etc/grid-security/voms-grid-mapfile files. The first one is regularly updated by edg-mkgridmap cron job and contains pairs of certificate subjects. e.g.

```
/C=RS/O=AEGIS/OU=Institute of Physics Belgrade/CN=Dusan Vudragovic
```

and pool account entries (e.g. aegis) allocated for user's Virtual Organization (VO). The second file is static and contains pairs of VO groups/roles (e.g. "/aegis/Role=VO-Admin") and pool account entries. Since VDT Globus [22] distribution used by the EGEE and SEE-GRID-SCI projects incorporates Pool Accounts patch [23] for Globus [24], in both cases the pool account entries are followed by a dot (".") that requires the function gridmapdir_userid to be invoked, either to allocate a Unix username from the pool of currently non-allocated accounts; or to return the username already allocated to this certificate subject.

Information on account mapping is stored in /etc/grid-security/gridmapdir directory. For each Unix username in the pool allocated for the user's VO, an empty file exists in this directory, with the same name as Unix pool username. When one of Unix accounts is allocated to a Grid user, a hard link with the same name as the certificate subject is created in /etc/grid-security/gridmapdir directory. The link points to the appropriate file according to the Unix username to which the subject is mapped. Certificate subject names are stored using

the "URL encoding" method - numbers are unchanged; letters are transformed to lowercase, while other characters are replaced by %HH where HH is their hexadecimal code. This is necessary since subjects can include slashes and other characters, "unfriendly" for naming files.

To find which username corresponds to which subject and vice versa, gFinger searches directory for links pointing to the same inode.

Example of usage on the gLite WMS service:

```
[root@wms ~]# gfinger seevo002
DN: /c=rs/o=aegis/ou=institute of physics belgrade/cn=aleksandar bogojevic:seevo
Login: seevo002          Name: mapped user for group ID seevo
Directory: /home/seevo002          Shell: /bin/bash
User ID: 23002          Group ID: 2300
or

[root@wms ~]# gfinger vudragovic

DN: /c=rs/o=aegis/ou=institute of physics belgrade/cn=dusan vudragovic:atlas:atlas
Login: atlas001          Name: mapped user for group ID atlas
Directory: /home/atlas001          Shell: /bin/bash
User ID: 20001          Group ID: 2000

DN: /c=rs/o=aegis/ou=institute of physics belgrade/cn=dusan vudragovic:aegis:aegis
Login: aegis002          Name: mapped user for group ID aegis
Directory: /home/aegis002          Shell: /bin/bash
User ID: 26002          Group ID: 2600
```

This script is very useful in diagnosing the problems related to the use of Grid services, when site administrator has to track the individual user through log files on different hosts.

### *C. DWARF*

DWARF [7] is a framework used for authorized Advanced Packaging Tool [25] (APT) and Yellow dog Updater Modified [26] (YUM) repositories management. It is developed at the Scientific Computing Laboratory of the Institute of Physics Belgrade.

In user/development communities where large number of partners/collaborators from different institutions jointly contribute to applications and RPMs built from applications' sources, it is useful to create a unique software repository that collects all such RPM. However, for security, scalability and reliability reasons, authentication and authorization of submitters should be established and closely checked.

DWARF allows uploading of RPM packages and creation of APT and YUM repositories, with the authentication and authorization based on digital certificates using Public Key Infrastructure [27] (PKI). This framework is implemented as a web application and it is composed of the DWARF web portal, DWARF modules and DWARF database. Architecture of the DWARF framework is shown in Figure 1.

DWARF web portal, the frontend of the DWARF framework is implemented as a PHP script under Apache HTTP server [28] on top of Secure Sockets Layer. Integration at the server level allows the server to retrieve the authentication parameters negotiated by SSL, and SSL achieves authentication via public-key cryptography in digital certificates.

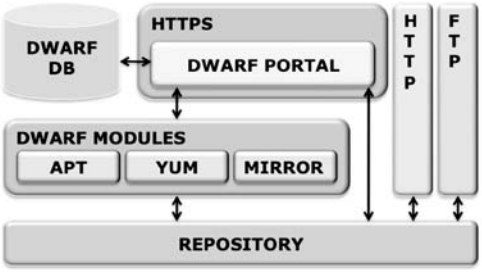


Figure 1: Overview of the DWARF architecture.

From the DWARF web portal, an authenticated and authorized user can perform following operations on the repository:

- Create and change repository structure – Users are free to create paths to new distributions and components, by specifying chosen names. In the current implementation of the DWARF framework, the users are able to create APT and YUM repositories, as well as to create a mirror to an existing remote repository.
- Package uploading – Users can upload different software packages, but only to sections of the repository to which they are authorized as contributors.
- Build repository – After each RPM upload, a user should build the repository structure. If not, a system will do it automatically, through a cron job.

DWARF modules are implemented as bash scripts that handle build action on repositories.

After an appropriate repository structure is created from the DWARF portal, all RPMs must be indexed in order to create the corresponding APT or YUM database. This is done by the APT DWARF and YUM DWARF modules, which analyze the RPM packages in a directory tree and build information files so that that directory tree can be used as an APT or YUM repository.

The MIRROR DWARF module is responsible for mirroring some existing software repository locally. Through the DWARF web portal, a user can specify a set of command-line switches that should be used to control the repository synchronization process. For example, files in the local repository that are not present in the remote directory can be deleted, or DWARF can download only newer files than the existing ones. Also, files and directories that should be skipped during the synchronization can be specified. By default, each

mirror repository will be synchronized six times a day (each 4 hours) via a cron job.

The DWARF database is realized using MySQL database technology, and contains information on security, repositories type, repositories metadata, mirror repositories, and logging information. All information about users and user's permissions for different repository sections are also stored here. In addition to that, the DWARF database contains metadata repository information on build's timestamps, contexts, and descriptions of the repositories, as well as repository types.

The DWARF framework provides configurations that must be included in the local HTTP and FTP servers' configuration files in order to provide the context of repositories once they are constructed.

SCL is using DWARF framework for managing self-produced RPMs and mirroring repositories of different versions of Scientific Linux, gLite software and some frequently used community repositories (e.g. dag). DWARF is currently deployed by the SEE-GRID-SCI Grid e-Infrastructure. Repositories can be accessed through the location given in Ref. [29].


Figure 2: Screen shot of the DWARF web portal.

*D. scl-bdii-conf*

Top level BDII [30] (Berkeley Database Information Index) server republishes information from site BDIIs. In order to do this, top level BDII server uses a list of LDAP [31] (Lightweight Directory Access Protocol) URIs (Uniform Resource Identifiers) pointing to site BDIIs. Top level BDII downloads this list periodically from an HTTP location. The scl-bdii-conf script prepares lists of LDAP URIs relevant for SEE (South Eastern Europe) Grid BDII instances.

Current implementation of the scl-bdii-conf [8] script prepares lists of LDAP URIs using the information from GOCDB [32], HGSM [33], and static files. The script itself is executed by the cron job each hour on the regional top-level BDII bdii.ipb.ac.rs.

*E. scl-wms*

WMS/LB processes are controlled by the gLite OS-level system daemons. However, we have observed that sometimes gLite service cannot be properly restarted, nor can be cleanly shut down when an administrator issues the standard "/etc/init.d/gLite stop" command. In order to resolve this until the corresponding issues are dealt with by the appropriate developers, we have developed scl-wms [9] script that is able to cleanly stop, start and restart all WMS-related daemons. For example, in order to stop all gLite services, the script first uses the OS-level daemon stop command, waits for some (configurable) amount of time, then checks if there are some remaining processes owned by the glite user, and kills them. The script accepts one of three possible options:

- start
- stop
- restart

*F. scl-clean-scratch*

A cluster worker nodes usually use shared home directory on a storage server in order to support MPI jobs. In the gLite middleware it is possible to define the start folder for jobs other than user's home, and this folder is designated as a scratch directory. In the case of MPI job, start directory is always user's shared home, but for other types of jobs it is usual to use local disk space on each Worker Node (WN) in order to avoid slow-down due to load of the storage server whenever possible.

When a job is finished, the jobmanager (PBS [34] in our case) is supposed to remove all job data from the scratch directory. However, this fails to be done from time to time due to a variety of reasons. Although it does not happen very frequently, the job data are unavoidably piling up in the scratch directory, which has to be cleaned regularly by a cron job setup by the site administrator.

In order to clean scratch folder, we have developed a scl-clean-scratch [10] script. This script is executed from the Computing Element on each WN. It first checks which Unix users have currently running jobs on each node, by querying the jobmanager. Then, script deletes all files and folders in each WN scratch directory owned by users other than those having currently running jobs on a given WN. The script uses the node list from the installed scl-scripts set of scripts (scl-nodes), described in section *A*.

*G. scl-generate-pool-accounts*

To generate configuration files for pools of Unix accounts necessary for the installation of gLite services, we have developed a script scl-generate-pool-accounts [11]. gLite installer YAIM [35] uses a file users.conf to create pool of

accounts for each Virtual Organization specified in the site-info.def configuration file. The script scl-generate-pool-accounts allows easy generation of the users.conf file, and simplifies their maintenance for site administrators. Apart from the initial installation of each Grid service, this script can be used whenever support for a new VO has to be added to a given Grid service. The definitions of users for users.conf wil be generated by this script using the command line options, specifying VO name, base user ID (UID), group name, group ID (GUID), group size (number of pool accounts to be created), prd size (number of special Grid production users for a given VO to be created in a pool) and sgm size (number of special software Grid manager pool accounts to be created). The default value for group size is 200, and default prd and sgm sizes are 10.

### *H. scl-generate-users*

On the occasion of organizing a Grid training event or in some similar case, administrators have to create a pool of accounts for participants of such an event on User Interface (UI). It can be easily done using the scl-generate-users [12] script. The script generates users on Linux system and sets random passwords for each user. Administrator can choose name for the pool and length of the initial set random password.

### *I. scl-network-sleep*

We have observed that in some cases network service is not operational immediately after the network daemon starts it. It could be up to the network adapter, or due to a switch behavior, which could take some time to recognize new NIC and configure a routing table for it. If this happens, some system services that need network to operate properly cannot be started. This is specifically the case with mounting of file system over network (e.g. NFS), which might fail if the network is not operational when the corresponding daemon is invoked. To ensure that network is running, we have created a small script scl-network-sleep [13] that is executed just after the network daemon, which sleeps enough (configurable) to allow network adapter to establish network connectivity through the switch.

### *J. scl-sensors*

A carefully developed set of scl-sensors [14] is intended to be executed on each worker node in a given cluster and provide the data about the node status. It obtains the CPU and motherboard temperature from the IPMI interface [36], as well as various other information from the operating system interface to the underlying hardware devices. The data are saved on a shared disk rather than sent through the network service. The data are later accessed and published by the http server that has access to the shared disk. We are using scl-sensors for Cumulative Grid Monitoring Tool [37] (CGMT) described in our contributed paper on “Grid Site Monitoring tools developed and used at SCL”, presented at this conference.

### *K. scl-jobs*

The script scl-jobs [15] is used to extract information about the running and waiting jobs from the jobmanager on specific Computing Element. This information is provided per supported Virtual Organisation. The script uses the PBS command qstat to get information about running and waiting jobs on CE. The script is easily extended to print not only numbers of running and waiting jobs, but also numbers of running and waiting processes, taking into account multiple processes initiated for parallel (MPI) jobs.

### *L. scl-tests-status*

The set of scripts scl-tests-status [16] is designed to access SAM [38], BBSAM [39] and GStat [40] tools to get information about the Grid site services status. In general, in order to access SAM database, one has to present a valid user certificate in a browser, and it is not possible to access SAM web portal directly via the script. For this reason we are using SAM Programmatic Interface (SAM PI), which allows access to SAM DB, provided that site administrator requests SAM firewall to be opened for the IP number of the host(s) that will execute the scl-tests-status script. BBmSAM and GStat monitoring tools can be accessed without certificates so script has only to find information on particular pages of these monitoring tools.

Scripts scl-sensors, scl-jobs and scl-tests-status are all integrated and developed for the use with the CGMT tool.

## III. Conclusions

We have presented a set of operational tools and scripts developed by the Scientific Computing Laboratory of the Institute of Physics Belgrade. The presented tools and scripts are used for various Grid operations of the two gLite-based sites at SCL. All of them are developed in order to ease and simplify Grid operations and manual tasks that have to be performed on many nodes, as well as for easier monitoring of the status of medium and large clusters. All presented tools and scripts are available for download from our SVN and WPM repository.

## Acknowledgment

This work is supported in part by the Ministry of Science and Technological Development of the Republic of Serbia through research grant No. OI141035, and by the European Commission through projects CX-CMCS (FP6), SEE-GRID-SCI (FP7) and EGEE-III (FP7).

## References

[1] EGEE, http://www.eu-egee.org/

[2] gLite, http://glite.web.cern.ch/glite/
[3] SEE-GRID-SCI, http://www.see-grid-sci.eu/
[4] Scientific Computing Laboratory, Institute of Physics Belgrade, http://www.scl.rs
[5] scl-scripts, http://http.ipb.ac.rs/tools/scl-scripts/
[6] gFinger, http://gfinger.scl.rs/
[7] DWARF web portal, https://dwarf.scl.rs/
[8] scl-bdii-conf, https://http.ipb.ac.rs/tools/scl-bdii-conf/
[9] scl-wms, http://http.ipb.ac.rs/tools/scl-wms/
[10] scl-clean-scratch, http://http.ipb.ac.rs/tools/scl-clean-scratch/
[11] scl-generate-pool-accounts, http://http.ipb.ac.rs/tools/scl-generate-pool-accounts/
[12] scl-generate-users, http://http.ipb.ac.rs/tools/scl-generate-users/
[13] scl-network-sleep, http://http.ipb.ac.rs/tools/scl-network-sleep/
[14] scl-sensors, http://http.ipb.ac.rs/tools/scl-sensors/
[15] scl-jobs, https://http.ipb.ac.rs/tools/scl-jobs/
[16] scl-tests-status, https://http.ipb.ac.rs/tools/scl-tests-status/
[17] BASH, http://www.gnu.org/software/bash/
[18] RSA, http://www.rsa.com/rsalabs/node.asp?id=2125
[19] DSA, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf
[20] SSH, http://tools.ietf.org/html/rfc4252
[21] Red Hat RPM Guide, http://docs.fedoraproject.org/drafts/rpm-guide-en/index.html
[22] VDT Globus, http://vdt.cs.wisc.edu/edg_lcg.html
[23] Pool Accounts patch for Globus, http://www.gridsite.org/gridmapdir/
[24] Globus toolkit, http://www.globus.org/toolkit/
[25] Wikipedia:Advanced Packaging Tool, http://en.wikipedia.org/wiki/Advanced_Packaging_Tool
[26] Yum website, http://yum.baseurl.org/
[27] CA, PGP and SKIP, The Black Hat Briefings '99, http://www.securitytechnet.com/resource/rsc-center/presentation/black/vegas99/certover.pdf
[28] The Apache HTTP Server Project, http://httpd.apache.org/
[29] SCL Repository Service, http://rpm.scl.rs
[30] BDII, https://twiki.cern.ch/twiki//bin/view/EGEE/BDII
[31] LDAP, http://www.openldap.org/
[32] GOCDB, https://goc.gridops.org/
[33] HGSM, https://hgsm.grid.org.tr/
[34] PBS Torque, http://www.clusterresources.com/products/torque-resource-manager.php
[35] YAIM, http://yaim.info/
[36] IPMI, http://www.intel.com/design/servers/ipmi/
[37] CGMT, http://cgmt.scl.rs/
[38] SAM, https://lcg-sam.cern.ch:8443/sam/sam.py
[39] BBmSAM, https://c01.grid.etfbl.net/bbmsam/
[40] GStat, http://goc.grid.sinica.edu.tw/gstat/.