A Module Extension to OpenPBS: Principles, Implementation and Experiences

Mike Becher[a] and Wolfgang Rehm[b]

[a] Computing Center of the University, Chemnitz University of Technology,
Str. der Nationen 62, 09111 Chemnitz, Germany, `mibe@hrz.tu-chemnitz.de`

[b] Department of Computer Science, Chemnitz University of Technology,
Str. der Nationen 62, 09111 Chemnitz, Germany, `rehm@informatik.tu-chemnitz.de`

OpenPBS [8] was developed within a research project of NASA Ames Research Center, Lawrence Livermore National Laboratory, and Veridian Information Solutions, Inc. It is a rife, widely used, and portable Batch-System, which complies with many of the requirements of such a system. Belonging to these among others is the management of computer resources ('nodes'), which also covers the monitoring of their static operating states, e.g. processor, operating system or hard disk space as well as their dynamic operating states, for instance disk level, processor load or memory usage rate. Furthermore the system permits the administration of jobs with the minimally necessary limiting conditions, such as authentication of users and administration of groups of users. Furthermore OpenPBS provides features, such as the provision of relevant data before a job starts, ('stage-in') the data return after the completion of a job, ('stage-out'), a Job-Execution-Functionality and a Distributed Shell Implementation.

If the application of OpenPBS is planned in an AFS [7] environment, further characteristics are necessary, which are not provided by the Batch-System. In order to achieve an access permission on the AFS file system for instance, a so-called AFS-Token, which is user-referred, is necessary. For safety reasons a PAG environment is needed additionally, which protects the AFS-Token of one user from being accessed by other users. Another example is the impact on setting or removing of access permissions when applications make use of SSH [11] or RSH. These Remote Shells are often used by MPI-Implementations, such as MPICH [6] or LAM-MPI [4], for providing the MPI-Environment on cluster nodes. A third example may be if a site needs support not only for accounting provided by the Batch-System but also for traditionally accounting based on `utmp/wtmp`.

The availability of the Source-Code, provided by OpenPBS, maintains a large flexibility for code changes. However a restart of services is often necessary for the activation of such a change. Therefore it might be more favorable if instead the code could be exchanged during run-time. This problem is specifically decisive for the 'site specific' code of Daemon-Processes. Referring to this the challenge is the implementation of the named cases using a proper set of modular extensions in order to be able to fulfill the requirements.

This Paper describes the Design of a modular extension of OpenPBS and describes the solutions [9] mainly with the help of a Module-Implementation for the safe (encrypted) forwarding and usability of an AFS-Token. Nonetheless further modules have been implemented on the basis of a uniform interface and it exists the possibility of the realization of further modules with this interface if necessary. Moreover it should be possible to transfer the Design also to other Batch-Systems with some adaptations.

1. Introduction

In 1999 a project was started to install a large Linux cluster with more than 500 nodes at Chemnitz University of Technology (Technische Universität Chemnitz, TUC). Today it is called 'Chemnitz Linux Cluster' or simply 'CLiC' [1]. It consists of 528 nodes and two cluster servers which are connected by two Fast-Ethernet networks. During process of evaluation of how to integrate such a large system into an existing computing environment some problems became obvious. One of them is each cluster node should behave to a user like a local Linux workstation.

Another problem is to manage access to nodes for a large group of different users in a fair manner. For that problem in most cases a solution is the use of a Batch-System. The main requirement on a Batch-System is to provide interactive and normal batch access to computing resources, in this case a set of nodes. When process of evaluation of different Batch-Systems was finished, OpenPBS[1][8] was the only one that met most of our requirements. For instance it provides interactive and batch access, as well as being available as open source software. Also, fact of on going development process of OpenPBS has played a role to prefer OpenPBS against other Batch-Systems at that time.

However, OpenPBS does not support the required feature to work in an AFS environment. Source code of OpenPBS does not provide hook code to implement full AFS support except for some empty structures and functions for forwarding some kind of credentials. Because of the availability of OpenPBS in source code and its good documentation, a project was started to provide AFS support in PBS. The question was at that moment what is the best point to start with that enhancement? Should AFS support be implemented as an AFS password forwarding mechanism or is there another way to serve this problem in a more flexible fashion?

The idea to forward a password is rejected due to of two reasons. Firstly, forwarding a password might lead to a security problem because in practice users mostly use single password for many access points. And secondly, independence from password changing is desired. Moreover, computing center of TUC does not use NIS to provide information about users and groups needed by hosts. It uses traditional mechanism based on local files like `/etc/passwd` or `/etc/group`. In such an environment an update of local password files may take some times till all hosts get notice of that action. Analysis was done to find out how an AFS token is handled by the AFS server and clients, as well as how you can achieve independence from problems arising in conjunction to password handling. Results of this analysis led to a basis for a concept to enhance OpenPBS with the needed features.

In some circumstances there may be the need to provide site specific code for example to change behavior of standard queue routing algorithm provided by OpenPBS or to map a user account from a foreign site to a user account at the Batch-System site. For these reasons OpenPBS provides a few hook functions that can be adopted by a site. Unfortunately, these codes are statically linked into the executables. After a substitution of code is done, a restart of the Batch-System or parts of it is mostly required in order to activate new functionality. This may lead to problems in a very busy batch environment where long running job are the normal case. The use of dynamic loadable code would minimize the number of restart procedures of the system and it would make it easier to substitute code at runtime.

As a conclusion of all these issues this paper shows

- an analysis of how OpenPBS handles a job,
- the principle of AFS token forwarding and extension,
- the use of dynamically loadable modules at runtime in OpenPBS,

and leads to a modular extension that fulfills our requirements.

2. Analysis of OpenPBS

As fore mentioned, before a concept could be designed, internals of OpenPBS have to be studied. One of the challenges is to enhance OpenPBS to transport an AFS token and to make it available for a job of a user when it is needed to authenticate the job as a valid AFS user process. Analysis was required, on how OpenPBS handles a job from its submission to its execution.

From submission to execution a job passes mostly six major steps. Only some special exceptions require some more steps. Example shown in figure 1 explains all major steps of lifetime of a job.

1. **job creation by user**
   The user wants to create a job that requires two nodes.

[1]'OpenPBS', 'Portable Batch System', and the PBS Juggler Logo are trademarks of Altair Grid Technologies, LLC.

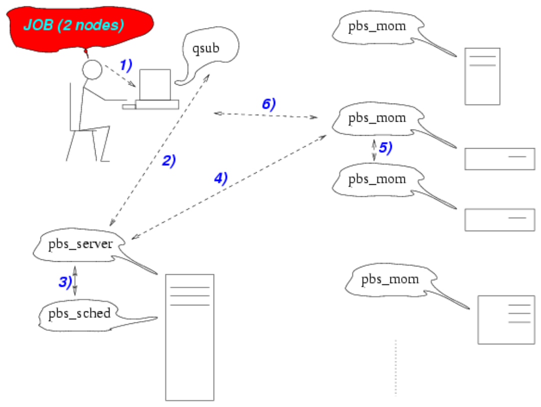

Figure 1. Steps followed by a job requiring two nodes

2. **job submission**
   If the user has created a job description with all needed information like number of nodes, maximum job wall time, etc., he submits his job with `qsub` command or with `xpbs`, which makes use of `qsub` internally. `qsub` performs syntactical checks and if these checks are passed then it contacts `pbs_server`, the batch daemon.

   Before `pbs_server` will take this job, it checks a few points of interest like, for instance, has the user permission to execute a job in the given batch environment, are there generally enough resources available to serve this job, etc. When all checks are successfully passed it takes the job and saves job script and its resource description. Job is now in state 'queued'.

3. **job scheduling**
   After 'queuing' is done the batch server sends a message to the scheduler, `pbs_sched`, to notify it about this recently arrived job. The scheduler evaluates the resource requirements of the job. If required resources are available it informs the batch server. It also provides the batch server with a hint of the nodes it can reserve for the job. To get valid information about debit balance and actual balance of batch environment it talks with `pbs_mom`, the Machine Oriented Mini Daemons (MOM).

4. **job stage in**
   Each node of the batch environment is running on instance of MOM. Its functionality is analogous to `execd` of DQS. Batch server will transfer the job to a master MOM (or superior MOM, as it is called in PBS). And this superior MOM gets the job.

5. **job execution**
   In case of a job that requires multiple nodes, superior MOM also gets a list of sister MOM's from the batch server. It will then contact all its sister MOM's to tell them to join this job. If all requirements are still valid this job will be executed. At that moment job is in state 'execution'. But, if there are not enough resources available the job will be 'queued' again, however, only if it is flagged as 'rerunable'.

6. **interactive job**
   In case of an interactive job, the `pbs_mom` opens a bidirectional connection to the waiting `qsub` command of the user to establish a console connection. Otherwise, if the job is a non interactive job (a simple batch job) this step is skipped.

additional comments:
Additional actions can be performed

- **before job execution**: delivery of relevant data ('stage in'), running administrative prologue script,
  or sending a mail to the user to announce the start of execution if wanted, and
- **after job execution**: sending a mail to user to announce the end or abortion of execution if wanted, running administrative epilogue script, and data return after the completion ('stage out')

3. The concept

Analysis has shown that in principle it is possible to identify and authenticate a user against AFS with a valid AFS token. So handling of passwords is not necessary. A problem of using AFS tokens is thier limited lifetime. It has to be extended from time to time. So, additional processes are required to perform this action. The 'credential templates' provided by OpenPBS sources may not fulfill this requirement. A proposed solution is described in next section on how to perform AFS token extension of lifetime.

3.1. AFS token extension

Figure 2 provides an overview about the whole process of AFS token forwarding and extension. Before `qsub` submits data for a job it should obtain a valid AFS token. If there is not a valid AFS token available job submission process should be canceled. Otherwise `qsub` sends the valid AFS token along with other job data to the batch server. At batch server site AFS token should be stored like

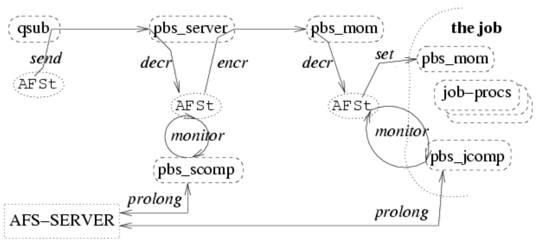

Figure 2. AFS token forwarding and extension scheme

all other relevant job data with the exception of an additional data file ('credential data file' or 'CDF'). Furthermore, the Batch-System should only know the data file name but not its size because its size may vary over time. At `pbs_server` site the AFS token lifetime should be monitored, and if its lifetime comes to an end it should be extended.

Extension of the AFS token must be done at the AFS server. It is the only host that has permission to do this action. So AFS token must be transferred from the batch server to the AFS server (perhaps with use of a secure channel that may be provided by SSH). At this moment the AFS token must be still valid. At the AFS server site, the token's lifetime will be extended and retransmitted to the batch server site over the same channel. The calling process at the batch server site stores the new AFS token into a local 'CDF'. Monitoring of the lifetime of AFS tokens will go on. As mentioned before, this action must be performed when the old AFS token is still valid.

Whenever the job and its data are transferred from the batch server to the superior MOM in case of execution, the AFS token should also be transferred. Superior MOM stores the job data at its site into a 'CDF' in the same way as the batch server. But superior MOM receives only a copy of the AFS token from the batch server.

MOM must set the AFS token of a job before it tries to receive date at time of 'stage-in' or before it begins with the process of job execution. Afterwared, administrative actions can be taken and/or job can be started. But when a job takes more time than the AFS token is valid, because of its limited lifetime, there must be an additional process as a companion for this job. This process should monitor AFS token lifetime at MOM site as it is done by the `pbs_server` process or the additional process at batch server site. Also, the process of AFS token extension may be done in the same way as at batch server site.

3.2. A module extension to OpenPBS

Analysis of OpenPBS made in section 2 and the proposed AFS token extension process described in section 3.1 lead to the following results:

1. One or several additional processes may be required at batch server site to monitor attributes like lifetime of the AFS token and than to extend this token. A design with additional processes is a good decision because actions, that must be performed, may take long time. Batch server cannot process those actions without latency because of its monolithic design ('single process'). So it would not answer any other batch requests meanwhile. Due to this fact, a 'server companion' should be implemented at the batch server site. It will be called `pbs_scomp`.

2. At least one additional process may be required at the superior execution daemon site to monitor and to perform repetitive actions. The decision was made because a running job, in reality, the user's application, cannot perform the required actions because it doesn't know anything about this need. So there should be a job companion, call him `pbs_jcomp`, which is not a companion for superior MOM. `pbs_jcomp` should be a companion of one and only one job. Figure 3 shows the process.

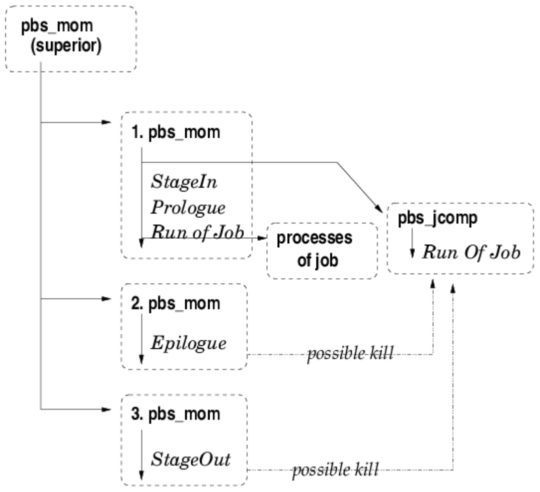

Figure 3. steps that will done by MOM to execute a job (module version)

3. All processes involved with job handling should be able to load code at runtime. In practice this should be done by loading a shared library, or a set of shared libraries, as 'module'. In conjunction with loading of shared libraries all processes should export their internal data, or symbols, such that a loaded shared library can take advantage of objects provided by the running processes.

4. An additional idea is that OpenPBS should spawn a child process of `pbs_mom` on each node of a sister MOM of a job. Then it can handle activities, such as AFS token extension, in the same way node of the superior MOM does. But this may require a complete redesign of the code of `pbs_mom`. It may be an idea for future work.

5. Some points in the code, where module actions should be performed, may be defined. Additionally source code of OpenPBS must be enhanced to enable loading of shared libraries and to support loadable modules.

6. Providing a module interface gives the possibility of development of third party modules. So an API should be defined and documentation should be prepared.

4. Implementation

The implementation of the design described in sections 3.1 and 3.2 was done in two development cycles. From 1999 to 2000 a first approach of implementation was made. It produced:

- a `pbs_jcomp` that runs besides each job
- a `pbs_scomp` that spawns a child for each arrived 'CDF' at batch server site
- a first version of module 'AFS_token' that ships token in clear text
- a first version of module 'Login_Control' to provide and manage a user access file for SSH or RSH (mostly used in MPI implementation)
- a first version of module 'PAG_tools' to provide a secure PAG environment in conjunction with the use of AFS token

All processes involved with job handling are aware of the action tags listed in Table 1 Source code of

Table 1
Action tags for module processing (version 0)

action tag	point in time	process
-BJC-	before job will be collected	`qsub`
-BJS-	before job will be submitted	`qsub`
-AJS-	after job was submitted	`qsub`
-WJIQ-	while job is queued	`pbs_scomp`
-BSIOJD-	before stage-in of job data	`pbs_mom`
-ASIOJD-	after stage-in of job data	`pbs_mom`
-BSOOJD-	before stage-out of job data	`pbs_mom`
-ASOOJD-	after stage-out of job data	`pbs_mom`
-BJST-	before prologue/job will be started	`pbs_mom`
-AJWF-	after job was finished	`pbs_mom`
-BES-	before epilogue will be started	`pbs_mom`
-BESJ-	before epilogue jcomp will be started	`pbs_mom`
-AEF-	after epilogue was finished	`pbs_mom`
-ASMWJJ-	after sister mom was joining job	`pbs_mom`
-BSMILJ-	before sister mom is leaving job	`pbs_mom`
-WJIR-	while job is running	`pbs_jcomp`

all the programs of OpenPBS was enhanced to provide loading of modules. Communication protocol between clients (`qsub`) and batch server (`pbs_server`), and between batch server and superior MOM (`pbs_mom`) was also enhanced. to provide transport of additional data along with other job data, and store it in a 'CDF'.

Some clusters with this implementation are working without complications at the TUC site. This version was also used on 'CLiC' until march 2003.

But this first approach has shown two security problems

- AFS token forwarding between clients and batch server, and batch server and superior MOM is done in clear text.
- The control over which module should be loaded is done by the 'CDF' that is created by clients.

In summer 2002, when this module extension to OpenPBS was in use for two years, an analysis of the first approach was done. It produced the following results:

- More action tags must be available to replace clear text forwarding of AFS tokens by a more secure version. Then, an encryption software, such as GNUPG[2] can be used. The batch server must be involved in actions to decrypt a received AFS token and to encrypt it when it should be transferred to superior MOM.
- To have control over module loading, a local configuration file should be preferred instead of a 'CDF' delivered by a client.
- The `pbs_scomp` 'process-per-file' ('CDF') approach does not scale to a large number of jobs. A concept that may follow 'master-slave-scheme' must be implemented.
- Due to the fact of the large number of installations of the first version of module extension to OpenPBS, the new design must provide a backward compatibility.
- New modules are required to support local accounting, for instance.
- `Libsite`, provided by OpenPBS in server code (`pbs_server`, `pbs_mom`), should be also modularized so restarts of batch services can be avoided.

Based on these results a complete redesign of the module interface was done in Fall 2002. The new design is easier to understand, and it provides additional action tags. All action tags require less arguments. Table 2 shows the additional action tags. Also, site specific functions provided by

Table 2
Additional action tags for module processing (version 1)

action tag	point in time	process
-WJIAOS-	when job is arrived on server	`pbs_server`
-WJWBMBS-	when job will be moved by server	`pbs_server`
-WJWMBS-	when job was moved by server	`pbs_server`
-SCTEST-	test tag can be used by scomp	`pbs_scomp`
-WJCU-	when job comes up	`pbs_jcomp`
-WJGD-	when job goes down	`pbs_jcomp`
-SAUA-	site allow user access	`pbs_server`
-SAR-	site alternate route	`pbs_server`
-SCUM-	site check user map	`pbs_server`
-SAC-	site acl check	`pbs_server`
-SMU-	site map user	`pbs_server`
-SMU-	site map user	`pbs_mom`
-SMCU-	site mom check user	`pbs_mom`
-SMPC-	site mom post checkpoint	`pbs_mom`
-SMPR-	site mom pre restart	`pbs_mom`
-SJS-	site job setup	`pbs_mom`

OpenPBS were changed to provide module extension support.

Additionally, the server companion was reimplemented. Now it works in accordance to the 'master-slave-scheme'. Number and behavior of master and slaves can be controlled by a configuration file.

Since the beginning of 2003 there are six modules available and in use. These modules are

1. Module ‘AFS_token’
   It is designed to let an Batch-System work in an AFS environment. It reads a valid AFS token on job submission site (`qsub`), stores and extends it at `pbs_server` and superior MOM’s sites. This version of the module sends AFS tokens in clear text. Extension is done over a secure SSH channel between the AFS token server and the batch server or MOM.

2. Module ‘AFS_token_gpg’
   It has the same functionality as module ‘AFS_token’, except this version sends an AFS token in encrypted text. Encryption and decryption is done by GNUPG [2].

3. Module ‘Mod_utmp_wtmp’
   It is designed to provide support for local accounting facility via `utmp` and `wtmp`. When a job starts a user is inserted into `utmp` and `wtmp` and when it finished these entries are cleared.

4. Module ‘Login_Control’
   It is designed to provide an additional user access file (`node-user.access`) similar to `login.access` file. With this file a host has the opportunity to configure its PAM (‘Pluggable Authentication Modules’ [12,3]) enhanced TELNET-, SSH- or RSH dynamically by the Batch-System to give a user permissions to access this node.

5. Module ‘PAG_tools’
   It is designed for setting up a PAG environment where an when it is required and ‘unlog’ it when it is not required anymore. This is used mainly to provide a secure AFS token environment. Implementation of shared libraries makes use of either KRB4 only or ‘krbafs’ and ‘krb4’ from KRB5 [5]. This could be configured at time when module is compiled. Use of KRB5 is advised.

6. Module ‘SACtable’
   This module provides two routines to decide whether a user will get access to the Batch-System or not based on a table that content ‘project groups’ (account) and user accounts. This table is managed by an administrator of the Batch-System.

4.1. An example of usage

Because there is not enough space to describe the usage of ‘AFS_token’ or ‘AFS_token_gpg’ modules in detail we try to show the usage based on module ‘Mod_utmp_wtmp’.

As previiously mentioned, a module consists in most cases of a shared library or a set of shared libraries. In this special case the module consists only of one library named `libuwtmp.so.1`. Loading of shared library (name and version) and calling of functions provided by this library can be configured through the configuration file `pbs_modules.conf`, shown in example below. The library by itself must be installed in custom PBS module path
`$LIBDIR/pbs_custom/<module name>` because processes are searching only in this path. To get a full description of syntax of configuration file, please read documentation at [10].

In the case of module ‘Mod_utmp_wtmp’ there is little work to be done to configure this module. The only thing to do is to activate the module at MOM site in the configuration file `pbs_modules.conf`. An adminstrator has to insert the following lines:

```
##PBS_MOD_BEGIN Mod_utmp_wtmp
##PBS_MOD_TAG -BJST-   -L- -v 1 uwtmp do_insert_user
##PBS_MOD_TAG -AEF-    -L- -v 1 uwtmp do_remove_user
##PBS_MOD_TAG -ASMWJJ- -L- -v 1 uwtmp do_insert_user
##PBS_MOD_TAG -BSMILJ- -L- -v 1 uwtmp do_remove_user
##PBS_MOD_END Mod_utmp_wtmp
```

which have the following meaning:

- Before a job will be started activate entry about this job in `utmp` with pseudo terminal `pbs/<pseudo i>` and remote hostname `pbs-<jobid>`. Also, insert such an entry into `wtmp` (load `libuwtmp.so.1` and execute function `do_insert_user()`).

- After jobs epilogue script finished deactivate entry in `utmp` and insert an entry into `wtmp` that notifies that the job finished (load `libuwtmp.so.1` and execute function `do_remove_user()`).
- Do activation process at sister MOM site (load `libuwtmp.so.1` and execute `do_insert_user()`).
- Do deactivation process at sister MOM site (load `libuwtmp.so.1` and execute `do_remove_user()`).

Now an administrator can use, for instance, the `last` command to get an overview about usage statistics in traditional manner. The output of `last` command may look like this:

```
mibe@clic2b11:mibe: last
mibe     pbs/2380     pbs.5472.clic0a1 Mon Jun 30 15:54   still logged in
mibe     pbs/2380     pbs.5471.clic0a1 Mon Jun 30 15:53 - 15:54  (00:00)

wtmp begins Mon Mar 10 15:53:55 2003
```

5. Conclusions

At TUC OpenPBS with module extension is used on nearly all clusters. It works properly on clusters like ‘CLiC’ [1] installed at computing center, as well as on ‘Roulette’ or ‘Riesen’ installed at Department of Physics.

We successfully migrated from the existing system to the secure version of AFS token forwarding and extension module to solve our security problem with clear text token forwarding. Furthermore, support for local `utmp/wtmp` accounting was enabled.

Redesign of `pbs_scomp` leads to a more stable and scalable working environment. Now it can serve hundreds of jobs which are queued concurrently at the batch server site.

The feature of backward compatibility to prior versions of module extension of OpenPBS allows a updating to the newer version. But the update must be done on execution hosts first. Afterwards, the client software can be updated as well. This process was realized at time of updating of ‘CLiC’ during first week of march 2003.

REFERENCES

[1] *CLiC – Chemnitzer Linux Cluster*, URL: http://www.tu-chemnitz.de/urz/anwendungen/CLIC/

[2] *The GNU Privacy Guard*, URL: http://www.gnupg.org

[3] *Linux-PAM* (Pluggable Authentication Modules for Linux), URL: http://www.kernel.org/pub/linux/libs/pam/

[4] *LAM / MPI Parallel Computing*, URL: http://www.lam-mpi.org

[5] *The Network Authentication Protocol*, URL: http://web.mit.edu/kerberos/www/

[6] *MPICH-A Portable Implementation of MPI*, URL: http://www-unix.mcs.anl.gov/mpi/mpich/

[7] *OpenAFS Homepage*, URL: http://www.openafs.org

[8] *OpenPBS v2.3: The Portable Batch System Software*, Altair Grid Technologies, LLC, Mountain View, CA, September 2000, URL: http://www.openpbs.org

[9] *Module Extension to OpenPBS*, URL: http://www.tu-chemnitz.de/˜mibe/sw/OpenPBS/

[10] *Documentation for Module Extension to OpenPBS*, URL: http://www.tu-chemnitz.de/˜mibe/sw/OpenPBS/lang/en/doc.php3

[11] *OpenSSH Homepage*, URL: http://www.openssh.org

[12] *PAM modules* at Openwall Projects site, URL: http://www.openwall.com/pam/

[13] *Portable Batch System - External Reference Specification*, NASA Ames Research Center and National Energy Research Center, December 1997

[14] *PBS Pro Homepage*, URL: http://www.pbspro.com

[15] The PSCHED API Working Group: *PSCHED: An API for Parallel Job/Resource Management*, Version 0.1, 1996

[16] Karl Czajkowski, Ian Foster, Nicholas Karonis, Carl Kesselman, Stuart Martin, Warren Smith, Steven Tuecke: *A Resource Management Architecture for Metacomputing Systems*, Proc. IPPS/SPDP ’98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, 1998.