<myJAM/> -

Sophisticated Accounting and Monitoring

Quick Start Manual

January 25, 2011

1 Introduction

<myJAM/> is a net-distributed job accounting and monitoring system for homogenous and heterogeneous compute clusters. It interacts with an existing batchsystem (e.g. Torque[1], PBSPro[2], etc.) to get information about then jobs running on a compute cluster.

All collected data are stored in the <myJAM/> database which is realized with MySQL[3]. We designed our database schema to extensively account for the Third Normal Form (3NF). All entities used in <myJAM/> like users, projects, queues, etc. are represented by the central tables in this concept.

The <myJAM/> web front-end is a highly interactive web application and the main interface for users and administrators. It generates W3C compliant XHTML and is written in object orientated PHP5 and JavaScript. For the visualization we make use of the glorious OpenFlashChart[4] library.

Intrinsically, we planned to make a fork of the open source project myPBS[5], which we found abandoned at version 0.8.4, dated from April, the 7th 2006. But then we basically completely redeveloped it and only parts of the look and some concepts (e.g. the *Service Units*) are left. Countless new features have been developed. We hope, you will find them useful.

We have been highly inspired by the concepts of the ancient myPBS. We even dared to name our project "myPBS-2" to show our respect to the original authors. But at the International Supercomputing Conference 2008[6], while presenting our project, we were informed by Altair[7] that we are not allowed to use the substring "PBS" anymore as it is a registered trademark of Altair Engineering, Inc.

So we renamed it to <myJAM/> for "Job Accounting and Monitoring".

[1] `www.clusterresources.com/products/torque-resource-manager.php`
[2] `www.pbsworks.com`
[3] `www.mysql.com`
[4] `http://teethgrinder.co.uk/open-flash-chart`
[5] `http://my-pbs.sourceforge.net`
[6] `www.supercomp.de/isc08/content`
[7] `www.altair.com`

2 Installing <myJAM/>

To use <my*JAM*/> you need a MySQL[1] database server and an apache web server[2]. Both of them can be run on separated systems. But of course the web server should be able to access the database! Also, the pro- and epilogue-scripts (which most likely run on the batch server) need to access the database.

We also have several installations of <my*JAM*/> running perfectly on the Zend Server Community Edition[3].

2.1 Getting <myJAM/> CE

You can always get the latest stable version of the community edition of <my*JAM*/> on SourceForge:

`http://sourceforge.net/projects/myjam`

If you need help, you can use the functions SourceForge offers to you or you can write an email to:

`myjam-support@uni-duesseldorf.de`

which is a direct way to our TroubleTicket system.

2.2 Extracting <myJAM/>

Extract the tgz-file you have got to a location, where your apache web server can access it. In a RedHat environment a good place could be `/var/www/html/myJAM_www`. Of course, your apache needs to know, that there is something that it has to serve. Just enter something like

[1] `www.mysql.com`
[2] http://httpd.apache.org
[3] `http://zend.com/de/products/server-ce`

```
<Directory /var/www/html/myJAM_www>
  Options None
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

to your apache configuration.

More detailed information about configuring apache can be found here: `http://httpd.apache.org/docs`

2.3 Setting Up The Database

Let's suppose you have a running MySQL installation. This should not be very hard to get as most of the Linux distributions come along with stable MySQL packages. We recommend MySQL 5.1.x.

Get onto the MySQL server and open a MySQL session as "root":

```
>$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 667
Server version: 5.1.49-log SUSE MySQL RPM

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

If the user "root" needs a password to login, you will be prompted to enter this. If not, hurry and give "root" a password! A very comfortable is using the script `mysql_secure_installation` which comes with any MySQL version. Just follow the dialog to secure your MySQL.

Next, create a database schema with any name you like, e.g. "myJAM":

```
mysql> create database myJAM;
Query OK, 1 row affected (0.00 sec)
```

Well done. Now you can terminate your MySQL session by either typing 'quit' or just press CTRL-C.

After this, you have to create all the tables for <my*JAM*/>. For this purpose, you will find a MySQL-creation script in the directory `DB_model`:

```
>$ mysql -u root -p myJAM < myJAM_DB2.4.sql
Enter password:
```

After entering the appropriate password all necessary tables and views will be created. This will take some seconds.

Last thing to do is, creating a database user and give him the rights to connect to the <my*JAM*/> database schema. MySQL can control the access of users to its databases very granular. For our purposes you need to create a user (e.g. "myjamd") and give him access coming from a specific host (*e.g.* your web server, where the PHP part of <my*JAM*/> is running). After this, you must give this tuple of user and host access to the database schema you created for <my*JAM*/> and allow this user to utilize the SQL statements SELECT, INSERT, UPDATE and DELETE (the usual *CRUD*-stuff).

Let's say we want to create a user "myjamd" with the password "secret" for accessing the database from "localhost". This may be the scenario if your webserver (running the <my*JAM*/> web frontend) is located on the very same machine as your MySQL server. In this case you need to do something like:

```
mysql> CREATE USER 'myjamd'@'localhost' IDENTIFIED BY 'secret';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON myJAM.* TO 'myjamd'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

Of course you need to create a (user, host)-tuple for every host that needs to access the database server. In our case these are the web server and the batch server on which the pro- and epilogue-scripts will run.

You can test your installation by going to all involved hosts and try

```
>$ mysql -u myjamd -p myJAM
```

If you are prompted for a password and after typing in the correct phrase you see the mysql-prompt, then everything should be just fine!

A more "fancy" way is to use the MySQL Administrator (Fig. 2.1, a quite useful GUI which is included in most of the Linux distributions. If not, you can get it here: `http://dev.mysql.com/downloads/gui-tools/5.0.html`

2.4 Configuring the Web Front-End

In the `config` directory of the <my*JAM*/> folder some configuration files can be found that need to be adapted to your needs.

2.4.1 Database Access

In `CFG_database.php` you can enter everything, that is necessary to connect to your database server (see below), like database user, the corresponding password, the IP or hostname of the database server and the name of the database schema.

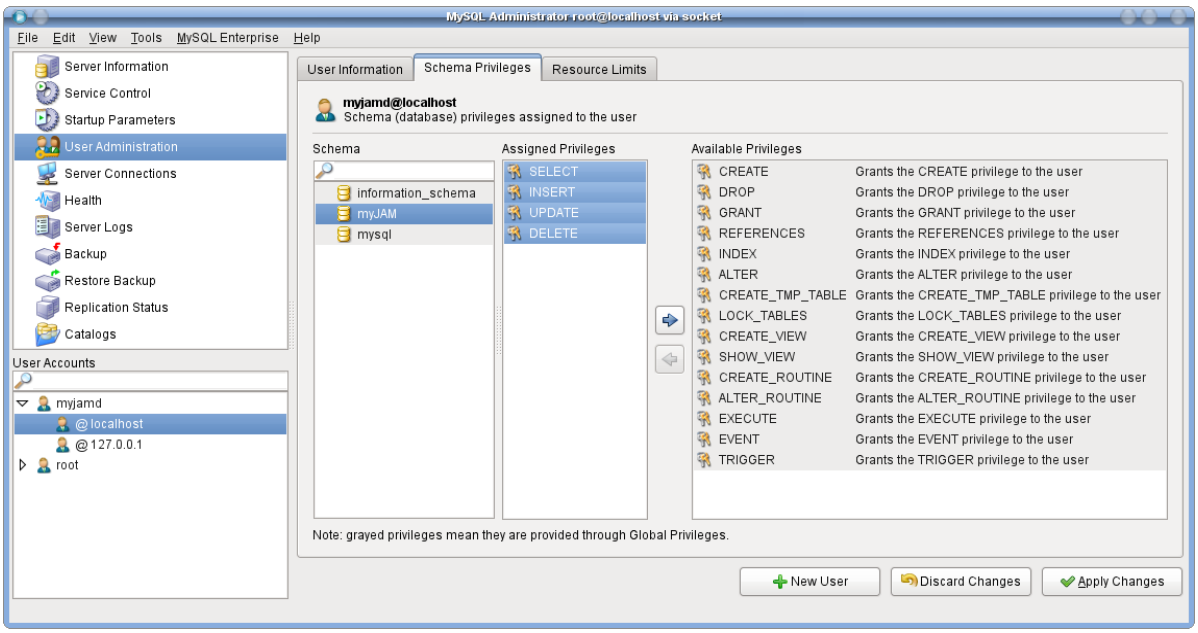

Figure 2.1: User Administration view of the MySQL Administrator.

For a local database server and a <my*JAM*/> database schema named "myJAM", `CFG_database.php` could look like (for all about username and password see below):

```
<?php
$_CFG_DATABASE_USER = 'myjamd';
$_CFG_DATABASE_PASSWORD = 'secret';
$_CFG_DATABASE_SERVER = 'localhost';
$_CFG_DATABASE_NAME = 'myJAM';
```

2.4.2 Batchsystem

<my*JAM*/> is designed to work with all kinds of batchsystems. Currently we have only implemented interfaces to pbs-oid systems, but we are working on interfaces for SGE and LSF, too! Nevertheless, <my*JAM*/> needs to know what batchsystem you are working with. This can be determined in `CFG_batchsystem.php`. The `_CFG_BATCHSYSTEM_SERVER` value is for planed extensions. Just let it set to "localhost" for the moment.

If the necessary executables of the batchsystem are not within your `PATH` you can also specify a directory where <my*JAM*/> will look for these in the variable `_CFG_BATCHSYSTEM_BINDIR`.

So, if you are using the widely-used Torque (with or without Maui), your `CFG_batchsystem.php` should be:

```
<?php
$_CFG_BATCHSYSTEM_TYPE = 'TORQUE';
```

```
$_CFG_BATCHSYSTEM_SERVER = 'localhost';
$_CFG_BATCHSYSTEM_BINDIR = '';
```

2.4.3 Authentication

There are two ways, to get access to the <my*JAM*/> web front-end: Firstly, via a HTML-based login-form on the first page of <my*JAM*/>. In this case login-name and a password (in MD5) are stored for every user in the <my*JAM*/> database.

A more elegant way is, to use an `.htaccess` file to use the same authentication method for the <my*JAM*/> web front-end that is used on your cluster anyways, *e.g.* a site-wide LDAP-Server, NIS, whatever. In this case only the username is stored in the <my*JAM*/> database.

An appropriate `.htaccess` file for using LDAP can look like:

```
AuthBasicProvider ldap
AuthType Basic
AuthzLDAPAuthoritative on
AuthName "Please login with your usual account data"
AuthLDAPURL ldap://myldapserver:389/[...]
require valid-user
```

Of course you have to complete the `AuthLDAPURL` regarding the structure of your LDAP data.

`CFG_auth.php` is the place to choose your way of authentication. The values can be either "DB" for the MySQL-based way or "HTACCESS" for using an authentication described in an `.htaccess` file:

```
<?php
$_CFG_AUTH_METHOD = 'DB';
//$_CFG_AUTH_METHOD = 'HTACCESS';
```

2.4.4 Mail Service

<my*JAM*/> can send emails to all users, if you want this. This is used for announcements. In `CFG_mail.php` you can set some variables to let <my*JAM*/> use your mail server. Just have a look at the example configuration; it is self-explanatory:

```
<?php
$_CFG_MAIL_FROM            = 'myJAM <myjam-support@uni-duesseldorf.de>';
$_CFG_MAIL_REPLY_TO        = 'myjam-support@uni-duesseldorf.de';
$_CFG_MAIL_ENVELOPE_SENDER = 'myjam-support@uni-duesseldorf.de';
$_CFG_MAIL_SIGNATURE       = [...]
```

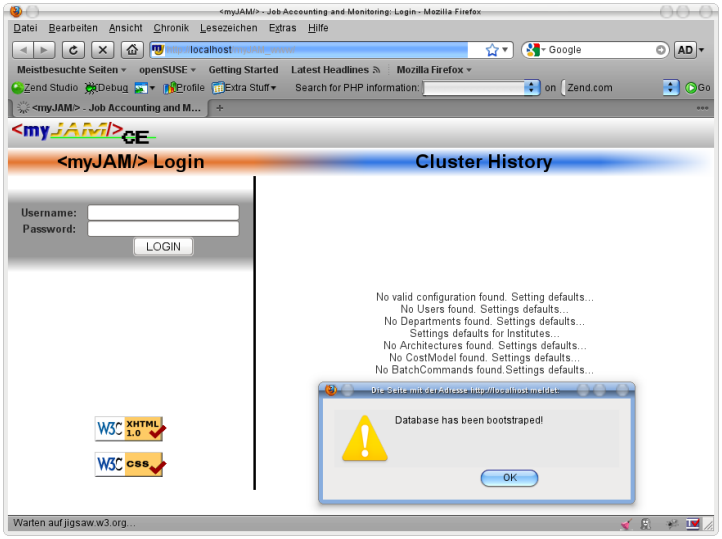

Figure 2.2: Bootstrapping the database.

2.5 Setting up the Pro- and Epilogue Scripts

The interaction of <my*JAM*/> with the batchsystem is done via pro- and epilogue-scripts which are called automatically by the batchsystem just before or right after a job respectively. The files are located in the `scripts` directory of <my*JAM*/> and need to run on the batchserver.

We have also included exemplary `prologue` and `epilogue` files for Torque which must be placed in the `mom_priv` directory on each compute node. These files need some editing: `HOST` should be set to the IP or the name of the batchserver, as can be accessed from the compute nodes (in most cluster environments the batchserver has several ethernet interfaces). `MYJAMDIR` is the location (absolute path) of <my*JAM*/> on the bathserver.

With these information the scripts just do an ssh to the batchserver and execute the `myJAM_prologue.php` and `myJAM_epilogue.php` respectively.

2.6 Discovering Your Cluster

The first time you call the web interface <my*JAM*/> will insert some defaults into the database. This will look like shown in Figure 2.2. After clicking "OK" the login page will be loaded (Fig. 2.3).

Now you can try to let <my*JAM*/> discover all the queues and computenodes you have in your batchsystem. Find below an exemplary output of a discover-run.

```
>/myJAM/scripts$ ./myJAM_Discover.php
myJAM_Discover>> Hostname: master-1
```

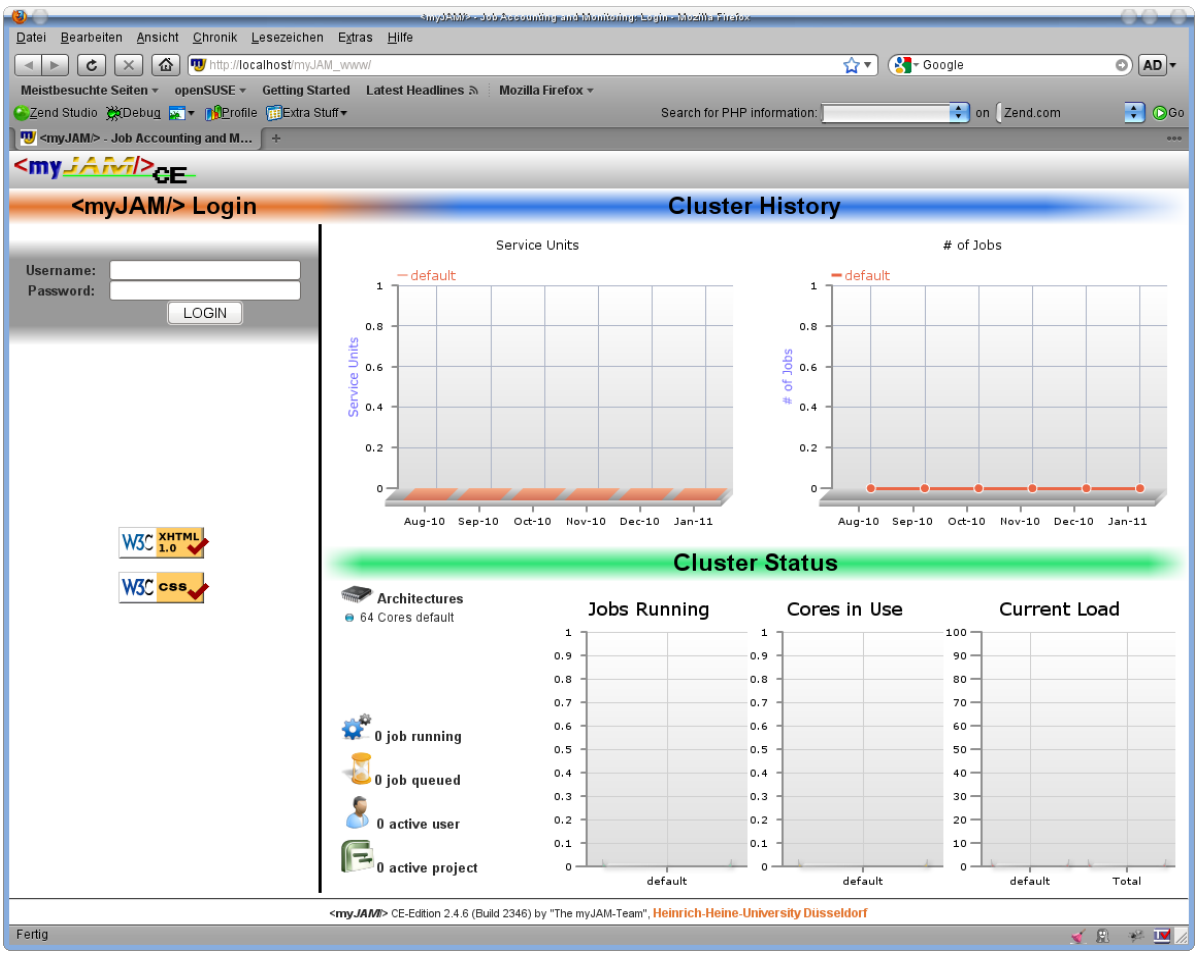

Figure 2.3: First call of the login page.

```
  >> No Host in database.
  >> Adding host "master-1"


myJAM_Discover>> Discovering Queues...
myJAM_Discover>> Queue: mpi
  >> no queues in database.
  >> Adding queue "mpi"
myJAM_Discover>> Queue: smp
  >> queue not in database.
  >> Adding queue "smp"
myJAM_Discover>> Queue: default
  >> queue not in database.
  >> Adding queue "default"
myJAM_Discover>> Queue: BenchMarking
```

```
  >> queue not in database.
  >> Adding queue "BenchMarking"
myJAM_Discover>> Queue: TheoPhys
  >> queue not in database.
  >> Adding queue "TheoPhys"

myJAM_Discover>> Discovering Nodes...
 >> node1 [online] [NO DATA]
 >> node2 [online]
              OS: Linux version 2.6.18-194.26.1.el5[...]
    Architecture: ia64
          # CPUs: 4
   Cores Per CPU: 2
          Memory: 33926447104 Bytes

 >> node3 [online]
              OS: Linux version 2.6.18-194.26.1.el5 [...]
    Architecture: ia64
          # CPUs: 4
   Cores Per CPU: 2
          Memory: 33926447104 Bytes

 >> node4 [OFFLINE]

[...]
```

If you don't experience any errors: Fine! Enjoy it!

3 The Concepts of <myJAM/>

Within <my*JAM*/> the central element is the "job", which is a representation of one job, that is known by the underlying batchsystem. <my*JAM*/> finds out about jobs by the prologue script, *i.e.* the very moment the job is placed on its execution host(s). From the ancient myPBS we have adopted the "Service Unit" (SU) which is just one core-hour, *i.e.* the used SUs of a job is the used walltime (in hours) multiplied by the used cores.

For each job the following values are recorded:

- Job ID
- Start date
- End date
- User
- Project
- Used Cores
- Requested SUs
- Used walltime, used SUs
- Used queue

Each user, who wants to start jobs on a cluster needs to be known to <my*JAM*/> and needs to be member of a <my*JAM*/> project (also something we found very good on myPBS). A project can have several users and each user can be member of more than one project! Each project can be granted access to an arbitrarily subset of the queues that are configured.

To determine the project under which a job should be accounted, an appropriate "accounting string" needs to be set. For Torque or PBSPro this can be done using the `-A` option, *e.g.*:

```
#!/bin/bash
#PBS -l walltime=167:00:00
#PBS -l nodes=1:ppn=2+31:ppn=1
#PBS -A H-Bonds
#PBS -N tm6_cc2

[do some calculations]
```

This would request 32 cores (of which at least the first two will share a compute node) for the project "H-Bonds" (which obviously is a project dealing with hydrogen bonds). For PBSPro the third line needs to be reformulated with a adequate select-statement.

The script `myJAM_prologue.php` will check if the user of this job is a member of this project and some other things. If only one prerequisite is not fulfilled, the script will terminate with a non-zero return value, which will result in the job being terminated by the batchsystem.

Calling `myJAM_epilogue.php` will set the state of the job to "finished" and will fill in missing metrics that can only be calculated after the job has terminated.