SLURM Bank
A collection of wrapper scripts giving “GOLD-like capabilities” for managing resources to SLURM

J. Tang  P. Doyle

Trinity Centre for High Performance Computing
Talk Outline

1. Introduction to SLURM Bank (sbank)
2. Design and Implementation
3. Limitations and Our Experiences
4. Future Work and Conclusions
5. Walk-Through
**SLURM Bank** is a collection of wrapper scripts to give simple banking capability to SLURM

- Create SLURM Associations; add users
- Deposit time with *GrpCPUMins*
- Simple interface for admins and users
Developers

- Jimmy Tang, Digital Repository of Ireland (DRI)
  jtang@tchpc.tcd.ie
- Paddy Doyle, Trinity Centre for High Performance Computing
  paddy@tchpc.tcd.ie
Historical Background

- Funding agencies require **accounting** for justifying use of resources
- Alternative software solutions:
  - Moab (commercial)
  - Sun Grid Engine (forks?)
  - Torque + Maui + GOLD
- TCHPC used SLURM + Maui + GOLD:
  - Issues: **reliability**, performance, scalability
- Didn’t need many features from GOLD: **simple accounting**
- Discussions on the [slurm-dev] mailing list (Apr 2011)
- Not an official work-sponsored project — driven by local devops
We already collect info about “projects” running on our clusters

- title
- abstract
- project leaders and members
- funding stream
- project start/end dates
- requested CPU hours

High-level details published online
Goals of SLURM Bank

- **Simple** banking system for admins and users
- Well-defined workflow using existing slurm features/tools
  - targeted at simplifying existing tools
Design

- Flat hierarchy of Associations
- Hard time limits per Association
- No half-life decay, no usage reset
- Set `GrpCPUMins` on an Association
  - The is the “account balance”
Design

- Use info provided by SLURM tools
  - `sacctmgr`, `sshare`, `sinfo`, `sacct`
- But have a single command (similar in spirit to `git` and its sub-commands)
  - Self-documenting
- Fix “CPU hour” as the unit (rather than minutes or seconds)
Rapid prototyping a proof-of-concept
  - Shell / Perl wrappers

Single command `sbank` similar to `git`
  - Wrapper around `sacctmgr`, `sshare`, `sinfo`, `sacct`
  - Use parsable output

Terminology: SLURM Association == `sbank` Project
Single wrapper script `sbank` to allow the following:

- admin creates projects (SLURM associations)
  
  ```bash
  sbank project create -c mycluster -a myproject
  ```

- admin adds users, add/refund hours
  
  ```bash
  sbank project useradd -c mycluster -a myproject -u someuser
  sbank deposit -c mycluster -a myproject -t 1000
  sbank refund job -j 5345
  ```

- tools for users to check balance, query, estimate, submit
  
  ```bash
  sbank balance statement -u
  ```
Limitations

- Reads usage info from `sshare`
  - No half-life decay is possible, for hard limits
- No lifetime/expiry of Associations / Projects
- Untested for multi-cluster
- No hierarchy of Associations
- No per-user limits within an Association
Our Experiences

- In production for over a year at TCHPC on 3 clusters
- `sbank balance statement` written to `slurm.out` file by SLURM Epilog
  - not always noticed by users
- No overdrafts!
  - run out of hours == people problem
- Heavy users vs light users
  - no usage decay with fairshare
Future Work

- Re-factor the implementation using SLURM API
- Investigate if `sreport` can be queried for usage info, instead of `sshare`
  - could re-enable half-life decay, and let fairshare work as intended!
- **Feature request**: add Association lifetimes/expiry to `slurmdbd`
  - similar to start- and end-times for reservations
Conclusions

- Currently deployed on SL5.x, SLURM 2.4 (also worked with 2.2 and 2.3)
- Funding agencies don’t care too much what software, so long as they get a report
- Users haven’t complained about changes in workflows, learning new command
  - users are silent mostly!
- As a sysadmin, much happier! Much more reliable than slurm+maui+gold
Admin Walk-Through: Install

- On a RHEL 5.x clone (with bash/Perl, and slurm):
  - `rpmbuild -ta -without docs slurm-bank-1.0.tar.gz`
- On a generic Linux system:
  - `make install`
- Docs: `man sbank` or `sbank help`
- Simple tests: `make test`
Admin Walk-Through: Setup

- Set the parameters in `slurm.conf`

```
AccountingStorageEnforce=limits
PriorityType=priority/multifactor
PriorityUsageResetPeriod=NONE
PriorityDecayHalfLife=0
```

- If you haven’t registered the cluster with `sacctmgr`, there’s a wrapper:

```
sbank cluster create mycluster
```
Admin Walk-Through: Create Projects

- Create SLURM Associations with:
  ```
sbank project create -c mycluster -a myproject
  ```
- Can also delete:
  ```
sbank project delete -c mycluster -a myproject
  ```
- Associate users with the project:
  ```
sbank project useradd -c mycluster -a myproject
  -u someuser
  ```
- And remove:
  ```
sbank project userdel -c mycluster -a myproject
  -u someuser
  ```
Admin Walk-Through: Policy

Decide on a local policy

- Figure out how many CPU hours are available on the cluster
- Decide on how many projects to support and how many hours to allocate to each project
- Decide on how much to over-subscribe
- Create associations for each project or group, perhaps setup a hierarchy of projects
- Allocate hours to the projects/groups
- Review projects and usage
- Go to start
Admin Walk-Through: Deposit Hours

- Deposit hours to a project:
  sbank deposit -c mycluster -a myproject -t 1000
- Remove hours:
  sbank deposit -c mycluster -a myproject -t -500
If a job has failed you may want to refund the hours that the job has used, to do this you need to know the job id.

`sbbank refund job -j 5345`

The refund command will look up slurmdbd, look up the association and the elapsed time. The elapsed time will be deposited back to the association where it originally ran from.

In general this should be left as a people issue.
To check your balances:

- `sbank balance statement -u`

<table>
<thead>
<tr>
<th>User</th>
<th>Usage</th>
<th>Account</th>
<th>Usage</th>
<th>Account Limit</th>
<th>Available (CPU hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>paddy</td>
<td>24</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>paddy</td>
<td>13</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
</tbody>
</table>
To check the balances of your associations, including other members of the associations:

```
sbank balance statement
```

<table>
<thead>
<tr>
<th>User</th>
<th>Usage</th>
<th>Account</th>
<th>Usage</th>
<th>Account Limit</th>
<th>Available (CPU hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>adamssl</td>
<td>0</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>jose</td>
<td>38</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>paddy *</td>
<td>24</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>darach</td>
<td>0</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
<tr>
<td>jtang</td>
<td>17</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
<tr>
<td>paddy *</td>
<td>13</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
</tbody>
</table>
To see the unformatted balance in a single association:

- `sbank balance statement -a tchpc`

315330
To see the balances of all associations in the cluster:

`sbank balance statement -A`

<table>
<thead>
<tr>
<th>User</th>
<th>Usage</th>
<th>Account</th>
<th>Usage</th>
<th>Account Limit</th>
<th>Available (CPU hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>0</td>
<td>ROOT</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>adamssl</td>
<td>0</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>jose</td>
<td>38</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>paddy *</td>
<td>24</td>
<td>MSCHPC</td>
<td>62</td>
<td>315,360</td>
<td>315,298</td>
</tr>
<tr>
<td>darach</td>
<td>0</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
<tr>
<td>jtang</td>
<td>17</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
<tr>
<td>paddy *</td>
<td>13</td>
<td>TCHPC</td>
<td>30</td>
<td>315,360</td>
<td>315,330</td>
</tr>
<tr>
<td>tom</td>
<td>113</td>
<td>HPC-03</td>
<td>30,030</td>
<td>100,000</td>
<td>69,970</td>
</tr>
<tr>
<td>fred</td>
<td>10,220</td>
<td>HPC-03</td>
<td>30,030</td>
<td>100,000</td>
<td>69,970</td>
</tr>
<tr>
<td>bob</td>
<td>19,697</td>
<td>HPC-03</td>
<td>30,030</td>
<td>100,000</td>
<td>69,970</td>
</tr>
</tbody>
</table>
How many CPU hours will a given number of nodes+cores for a given wall-time take?

```
sbank time estimate -N 64 -c 2 -t 72
```

9216

Or check how many hours a given script would require:

```
sbank time estimatescript -s sample-job1.sh
```

3072
Print expected balance when submitting a script:

```
sbank submit -s sample-job1.sh
```

```
log: Getting balance for jtang
User      Usage | Account      Usage | Account Limit | Available (CPU hrs)  
---------- ----------- + --------------- + ---------------- ----------- + - ------------ ----------- + - ------------  ----------- + - ------------  ----------- + - ------------  ----------- + - ------------  ----------- + - ------------
jtang 20 | TCHPC 32 | 315,360 | 315,328 |
```

log: Checking script before submitting
warn: no account specified in the script, using default: tchpc

Current balance = 315,328
Requested hours = 3,072
Expected balance = 312,256

log: sbatch’ing the script
Additional Commands

- Not banking per-se, but a few useful utilities we added
- Display CPU hours per cluster per period
  
  
  ```
  sbank cluster cpuhrs
  ```

  Cluster = lonsdale  Cores = 1216  Period = year  Avail = 10,652,160 hrs
  Cluster = lonsdale  Cores = 1216  Period = month Avail = 875,520 hrs
  Cluster = lonsdale  Cores = 1216  Period = week  Avail = 204,288 hrs
  Cluster = lonsdale  Cores = 1216  Period = day   Avail = 29,184 hrs

- Display max core count (or min)
  
  ```
  sbank cluster cpupernode
  ```
  ```
  sbank cluster cpupernode -m
  ```

- Convert SLURM time to hours
  
  ```
  sbank time calc -t 1-03:00:00
  27
  ```
  ```
  sbank time calc -t 4-01:00:00
  97
  ```
Questions ?
TCHPC: http://www.tchpc.tcd.ie/
GitHub: https://github.com/jcftang/slurm-bank
Docs: http://jcftang.github.com/slurm-bank/
Jimmy Tang, Digital Repository of Ireland (DRI)
  jtang@tchpc.tcd.ie
Paddy Doyle, Trinity Centre for High Performance Computing
  paddy@tchpc.tcd.ie