Keeping accounts consistent across clusters using LDAP and YAML

Christian Clémençon, Ewan Roche, Ricardo Silva

École Polytechnique Fédérale de Lausanne

École Polytechnique Fédérale de Lausanne

Swiss Federal Institute of Technology

SCientific **IT** and **A**pplication **S**upport

- Using SLURM since 2014

- Using SLURM since 2014
- Between 2 and 4 clusters

- Using SLURM since 2014
- Between 2 and 4 clusters
- 800+ active users

Dear support,
We have a new postdoc (Bob Smith) and would like to move our shares to the GPU cluster.

Thanks
Professor Jones

ASTRO Lab

Can you add Bob Smith to the Astro account and move their shares to the GPU cluster?

```
[admin@cpu] $ sacctmgr
create user bsmith account=astro
update account astro set share=1

[admin@cpu] $ su -
$ mkdir /scratch/bsmith
$ chown bsmith:astro /scratch/bsmith
```

```
[admin@gpu] $ sacctmgr
update account astro set share=120
```

Dear support,
thanks for moving the shares but our new postdoc still can’t use the GPU cluster. Did you create his account?

Thanks
Professor Jones

ASTRO Lab

```
[admin@gpu] $ sacctmgr
create user bsmith account=astro

[admin@gpu] $ su -
$ mkdir /scratch/bsmith
$ chown bsmith:astro /scratch/bsmith
```

We need a way to manage this

Validation

Validation

Only group leaders have the right to add people to their account

Validation

Only group leaders have the right to add people to their account

We have to ask for their permission

Shares

Shares

Our boss has a spreadsheet

Shares

Our boss has a spreadsheet

If she changes something in it we need to figure out what to do

Multiple clusters

Multiple clusters

We have between 2 and 4 active clusters at any one time

Multiple clusters

We have between 2 and 4 active clusters at any one time

The probability of having consistent account information is almost zero

Other people have the same problem

Scripts to the rescue?

Scripts to the rescue?

https://github.com/OleHolmNielsen/Slurm_tools

Scripts

Scripts

require super user rights

Scripts

require super user rights

are not inherently multi-cluster

Scripts

require super user rights

are not inherently multi-cluster

need input

Shared SLURMDB?

SLURMDB is great but

SLURMDB is great but

it's not a source of truth

SLURMDB is great but

it's not a source of truth

one shared instance implies a single point of failure

SLURMDB is great but

it's not a source of truth

one shared instance implies a single point of failure

one shared instance implies synchronising SLURM updates

We would like to achieve:

We would like to achieve:

- Delegation of tasks

We would like to achieve:

- Delegation of tasks
- Delegation of power

We would like to achieve:

- Delegation of tasks
- Delegation of power
- Responsiveness

We would like to achieve:

- Delegation of tasks
- Delegation of power
- Responsiveness
- Abstraction

We would like to achieve:

- Delegation of tasks
- Delegation of power
- Responsiveness
- Abstraction
- Unification

Delegation of tasks

Delegation of tasks

Adjusting shares

Delegation of tasks

Adjusting shares

Creating Accounts

Delegation of tasks

Adjusting shares

Creating Accounts

Adding users

Delegation of Power

Delegation of Power

Why can't the professor add users?

Delegation of Power

Why can't the professor add users?

Let the professor decide who else has the right to do this

Responsiveness

Responsiveness

It shouldn't take 5 days

Responsiveness

It shouldn't take 5 days

- User requests an account

Responsiveness

It shouldn't take 5 days

- User requests an account
- We ask the Professor

Responsiveness

It shouldn't take 5 days

- User requests an account
- We ask the Professor
- Professor says yes (eventually)

Responsiveness

It shouldn't take 5 days

- User requests an account
- We ask the Professor
- Professor says yes (eventually)
- Find a sysadmin

Responsiveness

It shouldn't take 5 days

- User requests an account
- We ask the Professor
- Professor says yes (eventually)
- Find a sysadmin
- Correct mistakes

Abstraction

Abstraction

A SLURM account is not a Unix group

Abstraction

A SLURM account is not a Unix group

Unix groups change

Abstraction

A SLURM account is not a Unix group

Unix groups change

Many to one mapping

Unification

Unification

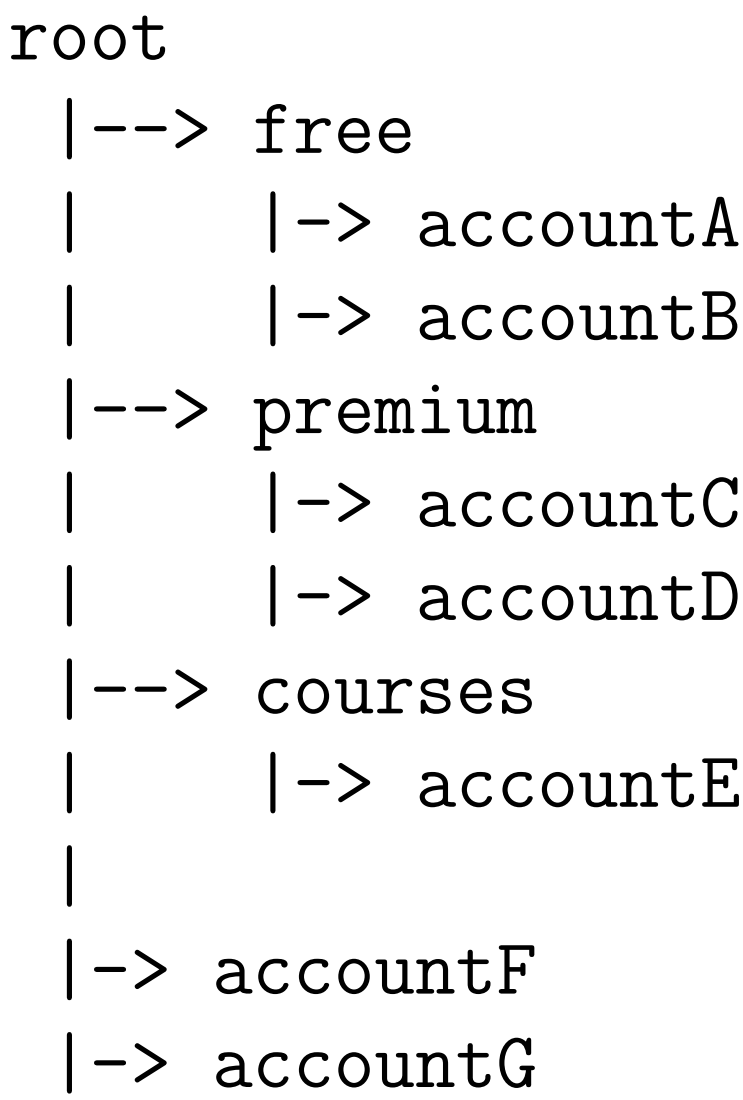
One configuration source

Unification

One configuration source

Consistent account information

We need a hierarchy


root
|--> free
| |-> accountA
| |-> accountB
|--> premium
| |-> accountC
| |-> accountD
|--> courses
| |-> accountE
|
|-> accountF
|-> accountG

We need a structured data format

YAML

We need a way to manage groups of users

Most large organisations have some form of LDAP/AD group service.

groups.epfl.ch

How can we put all this together?

PERL

PERL

- PERL 5

PERL

- PERL 5
- Net::LDAP

PERL

- PERL 5
- Net::LDAP
- YAML

PERL

- PERL 5
- Net::LDAP
- YAML
- sacctmgr

Design Principles

Design Principles

- Distributed

Design Principles

- Distributed
- Separate Data and Code

Design Principles

- Distributed
- Separate Data and Code
- Modular and extendable

accounts.yaml

`accounts.yaml`

- the source of truth

`accounts.yaml`

- the source of truth
- define account hierarchy

`accounts.yaml`

- the source of truth
- define account hierarchy
- map LDAP groups to accounts

`accounts.yaml`

- the source of truth
- define account hierarchy
- map LDAP groups to accounts
- set shares per account/cluster

`accounts.yaml`

- the source of truth
- define account hierarchy
- map LDAP groups to accounts
- set shares per account/cluster
- set some limits

groups.epfl.ch

`groups.epfl.ch`

- manage groups of users

`groups.epfl.ch`

- manage groups of users
- the professor is the administrator

`groups.epfl.ch`

- manage groups of users
- the professor is the administrator
- the professor can add administrators

`groups.epfl.ch`

- manage groups of users
- the professor is the administrator
- the professor can add administrators
- adding a user is the validation step!

Algorithm

Algorithm

- Calculate desired state

Algorithm

- Calculate desired state
- Check actual state

Algorithm

- Calculate desired state
- Check actual state
- Diff the two

Algorithm

- Calculate desired state
- Check actual state
- Diff the two
- Apply "rules" to transition

Algorithm

- Calculate desired state
- Check actual state
- Diff the two
- Apply "rules" to transition

See the code for the gory details

Where does this happen?

Where does this happen?

- accounts.yaml stored in GIT repo

Where does this happen?

- accounts.yaml stored in GIT repo
- git pull accounts.yaml on shared filesystem

Where does this happen?

- accounts.yaml stored in GIT repo
- git pull accounts.yaml on shared filesystem
- cron triggers the code on each cluster

Where does this happen?

- accounts.yaml stored in GIT repo
- git pull accounts.yaml on shared filesystem
- cron triggers the code on each cluster
- each cluster queries the LDAP service

accounts.yaml

```
SLURM Account:
    cluster1:
        parent:
        share:
    cluster2:
        parent:
        share:
    groups:
        - LDAP group 1
        - LDAP group 2
```

```
astro:
    castor:
        parent: premium
        share: 1
    deneb:
        share: 1036
    fidis:
        share: 1652
    groups:
        - hpc-astro
        - hpc-cosmo
```

Types of information

Types of information

- specific

Types of information

- specific
- generic

Types of information

- specific
- generic
- inherited

Types of information

- specific
- generic
- inherited
- implicit

Specific Values

Specific Values

Per account/cluster

Specific Values

Per account/cluster

- Shares

Specific Values

Per account/cluster

- Shares
- Walltime

Generic Values

Generic Values

Applicable to all clusters for a given account

Generic Values

Applicable to all clusters for a given account

- groups

Generic Values

Applicable to all clusters for a given account

- groups
- Walltime

Inherited Values

Inherited Values

We climb the tree to find the value

Inherited Values

We climb the tree to find the value

- Walltime

Inherited Values

We climb the tree to find the value

- Walltime
- MaxNodes

Implicit Values

Implicit Values

These are hardcoded

Implicit Values

These are hardcoded

- if cluster and share

  `parent : root`

Implicit Values

These are hardcoded

- if cluster and share

  `parent : root`

- if cluster not specified

  `parent : free`

Implicit Values

These are hardcoded

- if cluster and share

  `parent : root`

- if cluster not specified

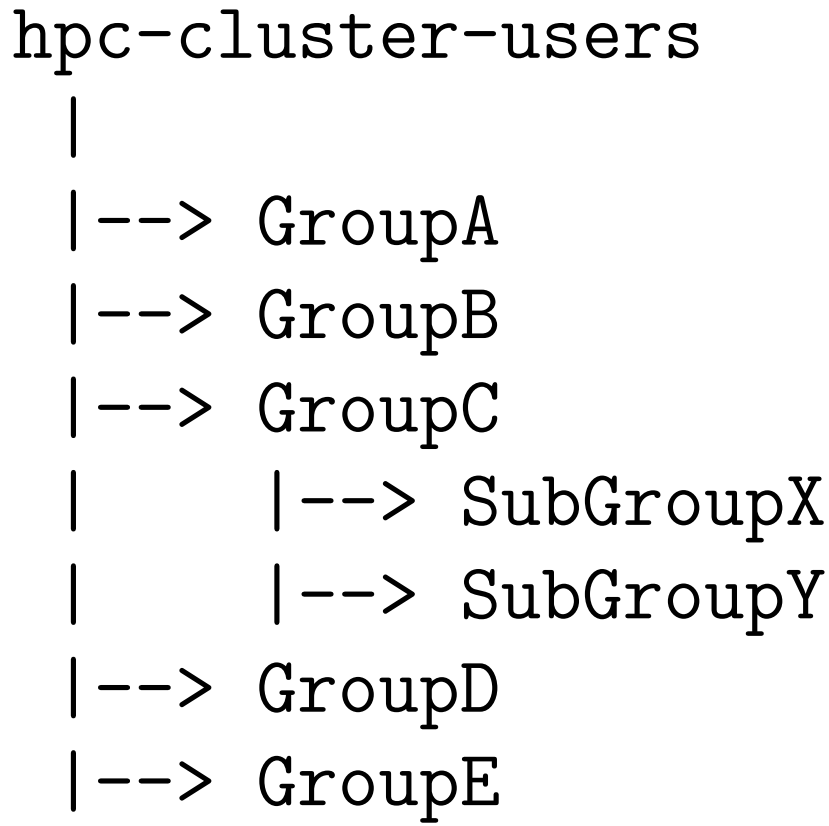
  `parent : free`

root also has a predefined MaxWall

Implementation detail

All the LDAP groups must be subgroups of the hpc-cluster-users group.

This matches with who can log on.


hpc-cluster-users
|
|--> GroupA
|--> GroupB
|--> GroupC
| |--> SubGroupX
| |--> SubGroupY
|--> GroupD
|--> GroupE

There is no structure defined here

We expand nested groups

Who changed what and when?

Logging

Logging

- One log file per cluster

Logging

- One log file per cluster
- Record changes

New user

```
2018-09-11@02:30:13 \
new user bsmith (user=bsmith group=phys sciper=123456 )

2018-09-11@02:30:13 \
/usr/bin/sacctmgr -i create user bsmith account=astro

2018-09-11@02:30:14 \
mkdir /scratch/bsmith;

2018-09-11@02:30:14 \
chown bsmith:10001 /scratch/bsmith; chmod 750 /scratch/bsmith
```

Update shares

```
2017-04-26@10:31:53 \
/usr/bin/sacctmgr -i update account astro set parent=root

2017-04-26@10:31:53 \
/usr/bin/sacctmgr -i update account astro set share=420
```

Update user

```
2018-01-16@02:30:11 \
modify user fred (user=fred group=pcsg sciper=123457 \
accounts=free,pcsg)

2018-01-16@02:30:11 \
/usr/bin/sacctmgr -i update user fred set defaultaccount=pcsg

2018-01-16@02:30:11 \
/usr/bin/sacctmgr -i delete user fred account=free
```

The LDAP service doesn't have logs

- The LDAP service doesn't have logs
  - Query group membership once per day

The LDAP service doesn't have logs

- Query group membership once per day
- Data in ElasticSearch

The LDAP service doesn't have logs

- Query group membership once per day
- Data in ElasticSearch

This records the state but not who did what

Who changed the shares?

Before: spreadsheet on NAS

Before: spreadsheet on NAS

Now: YAML file in GIT repo

git blame accounts.yaml

```
^84a0e68 (Clemencon Christian   62) csea:
^84a0e68 (Clemencon Christian   63)     deneb:
^84a0e68 (Clemencon Christian   64)         share: 576
902a9844 (Ewan Roche            65)     fidis:
902a9844 (Ewan Roche            66)         share: 1092
^84a0e68 (Clemencon Christian   67)     groups:
^84a0e68 (Clemencon Christian   68)         - hpc-csea
```

What happened when LDAP failed?

The code worked as designed

The code worked as designed

An empty group implies no users so let's delete all users/accounts

The code worked as designed

An empty group implies no users so let's delete all users/accounts

Deletion is now a separate task!

If we can do something with sacctmgr we can define it in YAML

How about managing the QoS per association?

QoS for time and resources

QoS for time and resources

- week

QoS for time and resources

- week
- fortnight

QoS for time and resources

- week
- fortnight
- month

QoS for time and resources

- week
- fortnight
- month
- gpu

`accounts.yaml`

or

`qos.yaml`

```
astro:
    fidis:
        share: 1652
    groups:
        - hpc-astro
        - hpc-cosmo
    qos:
        - week
        - gpu:
            - bsmith
```

```
week:
    - scitas
    - astro
    - chem
gpu:
    - scitas
    - astro:
        - bsmith
```

To be decided.

Once the structure is defined the implementation is easy.

What next?

Automatic git pull?

Automatic git pull?

Is it a good idea to have a manual step?

Default Account?

Default Account?

Currently alphabetical...

Default Account?

Currently alphabetical...

We could try and guess

Default Account?

Currently alphabetical...

We could try and guess

Requires a self service interface to be accurate

Managing accounts has gone from being painful and error prone to almost invisible.

Get the code and tell us what you think

c4science.ch/source/slurm-accounts/

`slurmtools@groupes.epfl.ch`

`scitas.epfl.ch`