Thiao: fractional resource sharing experiment

This is Thiao, an experiment for fractional resource management, the objective is to use some well trusted software to do job management, in this case SLURM, and also some cloud computing management software to manage the virtual machine infrastructure which this time is OpenNebula.

With this you can allow the execution of many jobs at the same time in one node while letting them execute in their own isolated environment without realizing that there are other jobs in execution in the same physical node.

Installing:

Note: this **requires** a working SLURM and OpenNebula installation:

```
#create the home directory for thiao
sudo mkdir /opt/thiao
sudo chmod 777 /opt/thiao
cd /opt/thiao
#retrieve the latest update, you can also download the zip file
git clone git://github.com/scarmiglione/thiao.git .
#make the database R/W for everyone
chmod 666 thiao.db
#compile the suspend and resume programs
scons
#configure in case you plan to use the **tsub mode**
./bin/Setup
```

Set the **OneAuth** variable in **thiao.cfg** and **src/thiao.h** and make sure the user slurm can read it (it's needed if you want SLURM to call the tclean.py script every time a job finishes it's execution).

Power saving mode

It is possible to use SLURM's power saving functions to **create and stop the virtual machines**, this can give a fine grained control of how much load each of your physical nodes will take. For instance, you can jump from n to n+1 VMs running in each node and then back depending on your queue(s).

First, install Thiao, then you just need to set the appropriate Suspend and Resume programs in your *slurm.conf* like this:

```
SuspendProgram=/opt/thiao/bin/Suspend
ResumeProgram=/opt/thiao/bin/Resume
ResumeTimeout=120
SuspendTime=120
```

After that restart the service.

Now just send a job to the queue using sbatch and let Thiao launch the virtual machines to run it.

The first time it may be necesary to start the VMs **manually**, lets say that you have the hosts fg0 trough fg3 (like the ones in the examples directory), you can launch them all like this:

```
/opt/thiao/bin/Resume fg[0-3]
```

After SuspendTime seconds of idle time (no jobs in the queue?) SLURM should bring down the hosts automatically running SuspendProgram as the SlurmUser.

tsub mode:

First phase: use tsub.py in the bin directory to submit a job, it is a wrapper program that does the following:

1. Submit the job received as parameter using sbatch.
2. Retrieve the job id.
3. Retrieve the job requirements (at this time only required nodes).
4. Launch as many VMs as nodes required.

The VMs will then be contextualized, connect to the slurmctld daemon and execute the job submitted.

Second phase: you should have set tclean as the epilog program for SLURM like this:

```
EpilogSlurmctld="/opt/thiao/bin/tclean.py"
```

When the job finishes its execution, the slurmctld will call tclean which will:

1. Read the id of the finished job.
2. Shutdown the virtual machines that where created for it.

Troubleshooting

Both SLURM and OpenNebula are independent and so is the troubleshooting, if you are sure that both are working correctly, check doc/NodeStatus for some hints that may help you.

End notes

Don't mix the modes.

Remember that this is an experimental scenario.

Take a look at doc/one-slurm.pdf for more information on the experiment.

SLURM's power saving guide

https://computing.llnl.gov/linux/slurm/power_save.html

OpenNebula documentation:

http://opennebula.org/documentation:rel3.0

This software was tested with the environment described in the presentation that can be found in the /doc/one-slurm.pdf directory which includes:

```
OpenNebula 2.2.1
SLURM 2.2.7
1 Debian/testing guest VM (clonable)
2 RHEL6 vm hosts
1 Debian/testing server
```