Homework 2 - The Beowulf cluster

- Course : **Operating Systems**
- Deadline : **2017/04/13 23:59**
- Demo : **2017/04/14 14:30 - 16:30**
- Contact : Che-Pin Chang (張哲彬), TA, jalex.cpc@gmail.com

Overview

In this assignment, we are goning to practice to build the Beowulf cluster, and use mpi to execute a program in parallel. Finally, implemmenting a simple web service to demonstrate its functionalities.

Tutorials

The Beowulf cluster

In this part, we are going to build the Beowulf cluster, which use mpi as communication channel.

1. Create 3 nodes(VMs) as **master**, **slave1**, and **slave2**.
2. Install **Network File System** in each nodes. Install nfs-server in master, and install nfs-client in slaves.
3. Create a user, named **mpiuser** in master. And then, share his home directory to slaves.
4. Setup ssh configurations in each nodes to make sure that master can access slaves without password-checking.
5. Install **MPICH** in each nodes.
6. Validation

    1. Download attached sample code - mpi_hello.c.
    2. Compile the code with the custom MPI C compiler.

    ```
    $ mpicc mpi_hello.c -o mpi_hello
    ```

    3. Run the program by mpiexec.

    ```
    $ mpiexec -n <total cpu number> -host master,slave1,slave2 ./mpi_hello
    // -n means the number of cores to ask for.
    // This should not be more than the sum of cpu cores in all nodes.
    ```

    4. You would see some message like this:

```
// if you have 6 cpus in total.
Hello from processor 0 of 6
Hello from processor 1 of 6
Hello from processor 2 of 6
Hello from processor 3 of 6
Hello from processor 4 of 6
Hello from processor 5 of 6
```

Octave and mpi package

In this part, we are goning to see more examples aboout mpi and its applied senarios.

1. Install octave.

```
$ sudo add-apt-repository ppa:octave/stable
$ sudo apt-get update
$ sudo apt-get install python-software-properties
$ sudo apt-get install software-properties-common
$ sudo apt-get install octave
```

2. Install octave-mpi.

   - If success, you can see the mpi in package list.

```
$octave
octave> pkg list
Package Name  | Version | Installation directory
--------------+---------+-----------------------
   mpi    |   1.2.0 | /usr/share/octave/packages/mpi-1.2.0
```

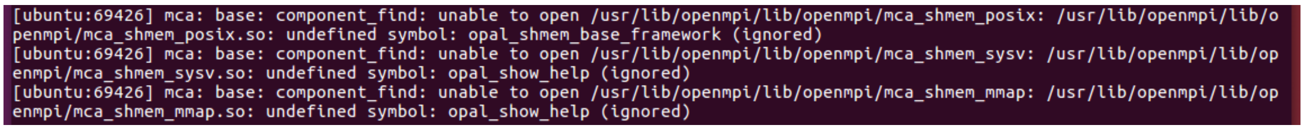
3. Simple validation

   1. Run example - Pi()

```
$ mpirun —np <number> -host master,slave1,slave2 octave --eval  'pkg load
mpi; Pi()'
...
results =
scalar structure containing the fields:

pi =  3.1416
err =   -6.2172e-14
time =  0.19877
```

   2. If you meet the problem like this:



```
[ubuntu:69426] mca: base: component_find: unable to open /usr/lib/openmpi/lib/openmpi/mca_shmem_posix: /usr/lib/openmpi/lib/openmpi/mca_shmem_posix.so: undefined symbol: opal_shmem_base_framework (ignored)
[ubuntu:69426] mca: base: component_find: unable to open /usr/lib/openmpi/lib/openmpi/mca_shmem_sysv: /usr/lib/openmpi/lib/openmpi/mca_shmem_sysv.so: undefined symbol: opal_show_help (ignored)
[ubuntu:69426] mca: base: component_find: unable to open /usr/lib/openmpi/lib/openmpi/mca_shmem_mmap: /usr/lib/openmpi/lib/openmpi/mca_shmem_mmap.so: undefined symbol: opal_show_help (ignored)
```

- Export **LD_PRELOAD** path in .bashrc

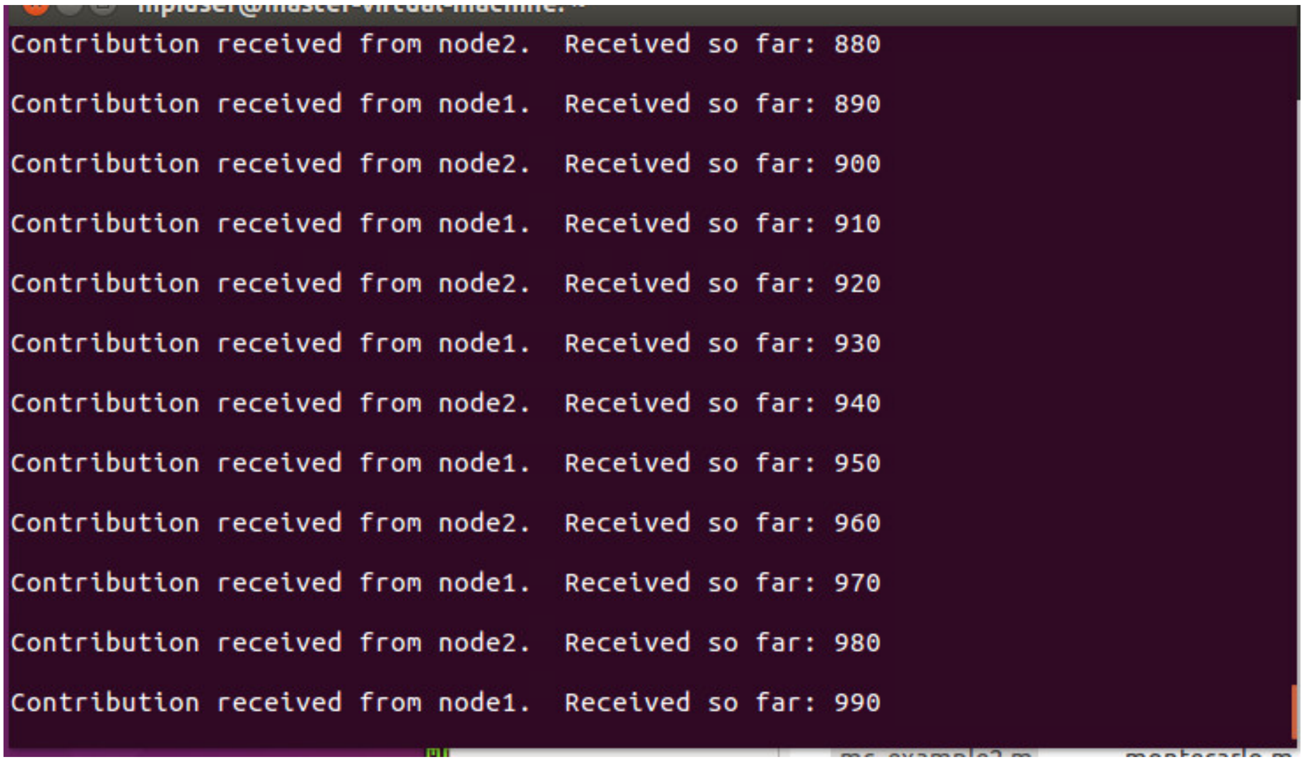
```
$ vim ~/.bashrc
export LD_PRELOAD=/usr/lib/openmpi/lib/libmpi.so.12.0.2
$ source ~/.bashrc
```

4. Example of Monte Carlo Algo.

   1. Demonstrates doing Monte Carlo with mpi.

```
$ mpirun —np <number> -host master,slave1,slave2 octave --eval  'pkg load mpi; mc_example()'
```

   2. You will see result like this:



```
Contribution received from node2.  Received so far: 880
Contribution received from node1.  Received so far: 890
Contribution received from node2.  Received so far: 900
Contribution received from node1.  Received so far: 910
Contribution received from node2.  Received so far: 920
Contribution received from node1.  Received so far: 930
Contribution received from node2.  Received so far: 940
Contribution received from node1.  Received so far: 950
Contribution received from node2.  Received so far: 960
Contribution received from node1.  Received so far: 970
Contribution received from node2.  Received so far: 980
Contribution received from node1.  Received so far: 990
```

   3. And you also find that the cpuloading of 3 nodes all raise up. After program was finished, the cpuloading descend to original status.

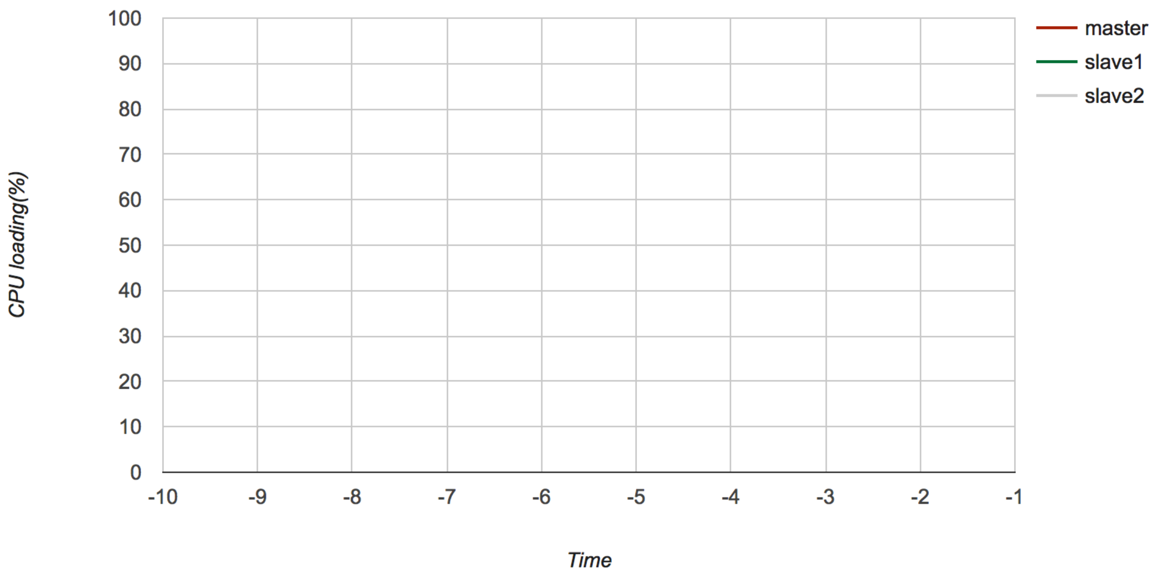
Web-based CPU loading monitor

In last homework, we had build cpuloading controller, which can generate cpu loading and list vms' status. In this homework, we are goning to visualize CPU loading of VMs.

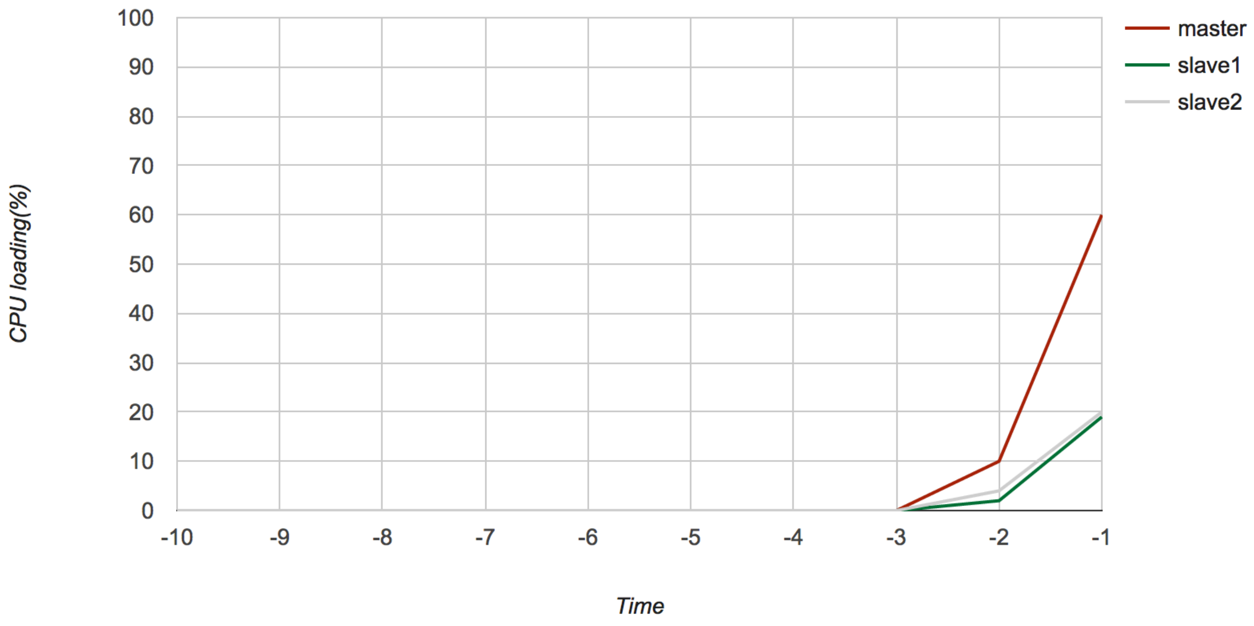
1. Install Apache Http Server in Master. **Notice that**, if you don't want to use PHP to build your backend

server, you can skip this step.

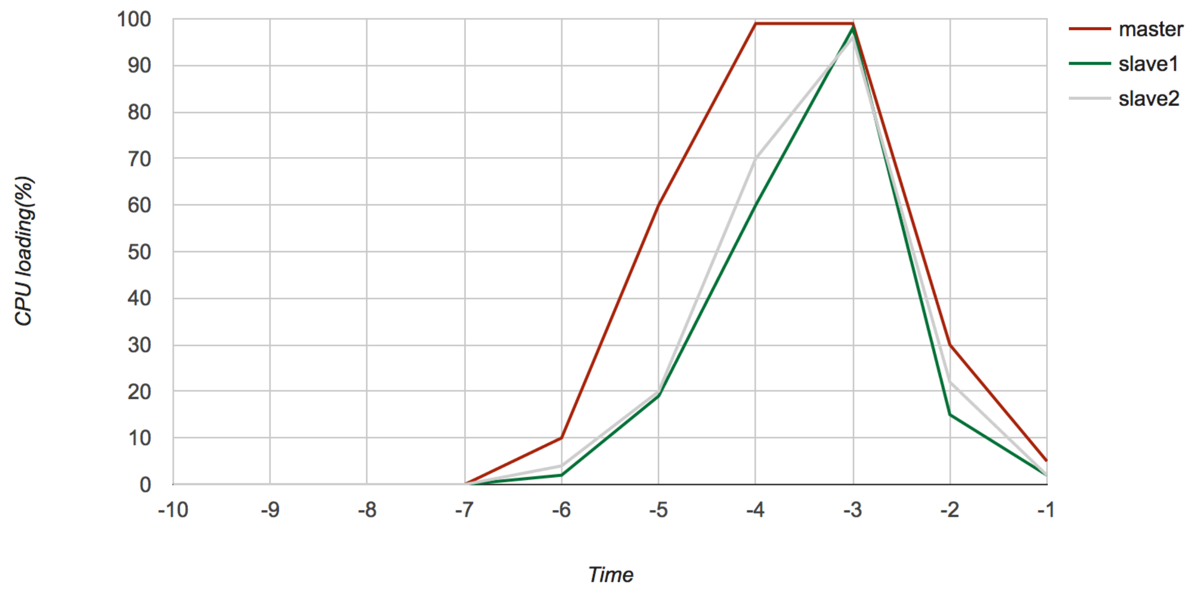
2. Build your own web service to manage VMs. This service provides following features:
   1. Dispalys cpu loading of 3 nodes in latest 10 seconds.
   2. Fix x-axis and y-axis. The range of x-axis is [-10,-1] and y-axis is [0,100].
   3. Periodically updates info without reloading the webpage.



3. You can use Google Chart to achieve visualization. Also, you can use any other visualization tools you prefer.
4. Demonstration
   1. Executing mc_example
   2. Begining



   3. Finished



Grades

In each part, we have prepared several oral questions to make sure that you finished the homework by yourself. If you meet the requirements and pass oral defense, you can get the following grades:

- The Beowulf cluster **(60%)**
- Octave and mpi package **(20%)**
- Web-based CPU loading monitor **(20%)**

References

- The Beowulf cluster
  - https://www-users.cs.york.ac.uk/~mjf/pi_cluster/src/
  - https://nixingaround.blogspot.tw/2017/01/a-homemade-beowulf-cluster-part-2.html

- Octave
  - http://www.linuxdiyf.com/linux/22034.html
  - http://askubuntu.com/questions/645600/how-to-install-octave-4-0-0-in-ubuntu-14-04

- Octave-mpi
  - https://octave.sourceforge.io/mpi/overview.html
  - https://octave.sourceforge.io/mpi/overview.html

- Google chart
  - https://developers.google.com/chart/