Introduction/Tutorial on the Linux Ecosystem

Hyperion cluster, SLURM scheduler, Measuring execution time (part 3)

Alexandru Iulian Orhean
aorhean@hawk.iit.edu

Illinois Institute of Technology
Computer Science Department
Data-Intensive Distributed Systems Laboratory

Table of contents

1. Hyperion cluster
2. SLURM scheduler
3. Measuring execution time
4. Hyperion upgrade

Hyperion cluster

Hyperion cluster

- The platform where you will run PA1;
- The platform where we will grade PA1;

```
localhost$ ssh <username>@129.114.33.105
hyperion$ passwd
```

- SLURM - free & open-source job scheduler for Linux and UNIX*;
- allocates computer resources to users in an **exclusive** and/or non-exclusive mode, for a limited amount of time;
- provides a framework for starting, executing and monitoring work on a set of allocated nodes;
- arbitrates contention for resources by managing a job queue;

SLURM scheduler

SLURM commands

```
$ sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
interactive    up    1:00:00     10   idle bluecompute-[1-10]
compute*       up      15:00     50   idle redcompute-[1-50]

$ squeue
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST
   46   compute     bash  aorhean  R       0:40      1 redcompute-1

$ scancel 46

$ srun -n 1 -p interactive --pty /bin/bash

$ sbatch run.slurm
```

SLURM job script

```
#!/bin/bash

#SBATCH --nodes=2
#SBATCH --output=main.out
#SBATCH --wait-all-nodes=1

echo $SLURM_JOB_NODELIST

./myprogram /tmp/input.txt /tmp/output.txt
```

Measuring execution time

User time vs System time vs Wall time

```
$ time du -sh /home
8.4G    /home/

real    0m17.233s
user    0m0.350s
sys     0m1.850s
```

- Wall (Real) time -> total time from start to finish, including wait time (end of process quanta or waiting for I/O to complete);
- User time -> total time spent on the CPU in user space (other processes and time the processes spends blocked do not count);
- Sys time -> total time spent on the CPU in kernel space (other processes and time the processes spends blocked do not count);

C example - CPU time

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    clock_t start, end;
    start = clock();
    sleep(10);
    end = clock();

    printf("elapsed: %fs\n", (((float) end - start)
        / CLOCKS_PER_SEC));
    return 0;
}

$ gcc -Wall main.c
$ ./a.out
elapsed: 0.000034s
```

C example - high precision Wall time

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    struct timeval  start, end;
    gettimeofday(&start, NULL);
    sleep(10);
    gettimeofday(&end, NULL);

    printf("elapsed: %fs\n",
        (float) (end.tv_usec - start.tv_usec) / 1000000 +
        (float) (end.tv_sec - start.tv_sec));
    return 0;
}

$ gcc -Wall main.c
$ ./a.out
elapsed: 10.000120s
```

Hyperion upgrade

Hyperion upgrade

Friday 2nd of Feb scheduled upgrade!

- not accessible between 6pm-12am;
- upgrade the login node
  (more ram & more cores)
- add more compute nodes
  (aiming for a total of 90 compute nodes)