Use SLURM job scheduling system on π supercomputer

SJTU HPC Center
hpc@sjtu.edu.cn

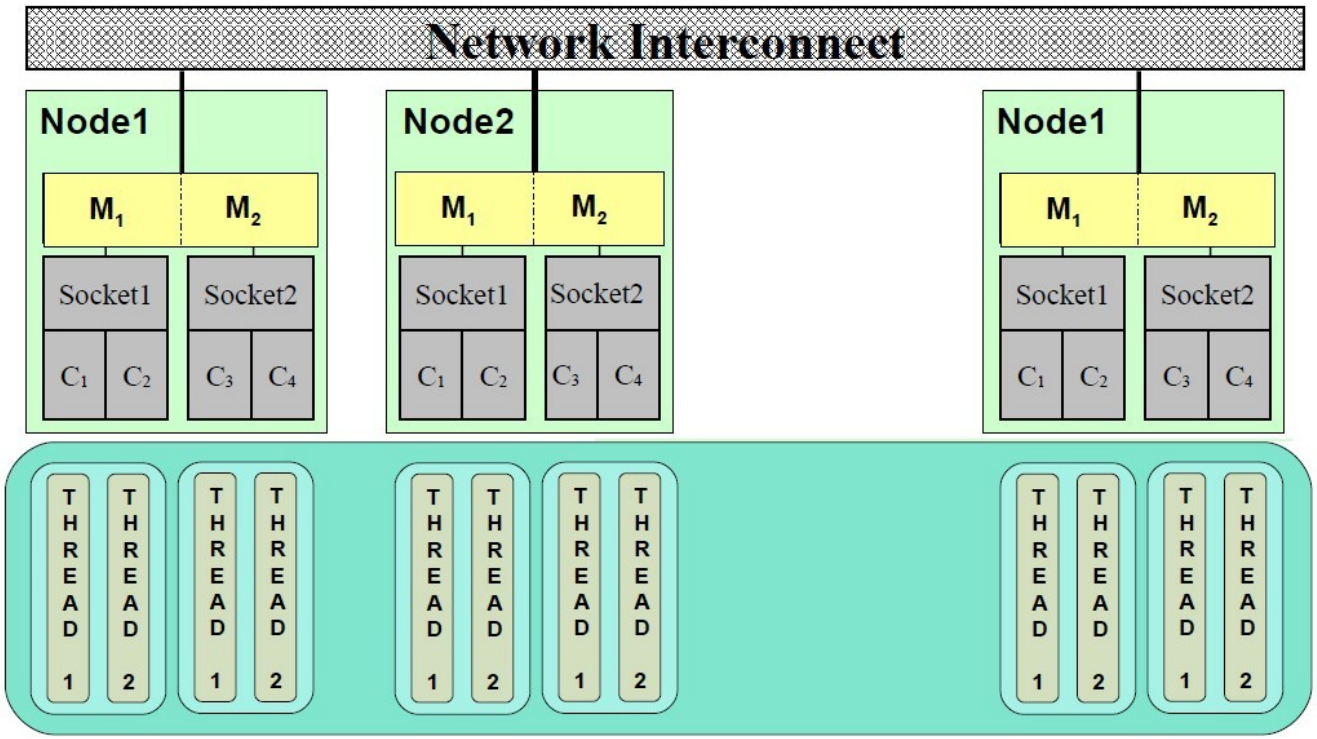
Jan 7th, 2016

1. Part I: A brief introduction to SJTU π supercomputer
2. Part II: Job submission via SLURM scheduling system
3. Part III: Smart environment modules
4. Part IV: Tips for monitoring your jobs

Part I: A brief introduction to SJTU π supercomputer

SJTU π: A computer cluster

- Multiple nodes connected by ultra high speed networks
- A virtual computer under programming abstraction (OpenMP, MPI)
- CPUs with low clock frequency, high parallelism, high aggregated computer power



Different Compute Queues on π

Queues are pools of compute nodes on *Pi*:

- `cpu`: 2xCPUs with 16 cores in total, 64GB Mem
- `fat`: 2xCPUs with 16 cores in total, 256GB Mem
- `gpu`: 2xCPUs with 16 cores in total, 2xK20m GPUs, 64GB Mem
- `k40`: 2xCPUs with 16 cores in total, 2xK40 GPUs, 64GB Mem

Assitant Programs for HPC users

- SLURM: the job scheduling system
- Environment Modules: Load and unload libraries with ease
- Ganglia monitoring page: http://pi.sjtu.edu.cn/ganglia

Part II: Job submission via SLURM scheduling system

Why another job scheduling system (SLURM)?

SLURM just works:

- Free and opensource
- Proven scalability and reliability
- Fast
- Debug friendly: SSH-login to compute hosts at running

Known issues of SLURM on π

- NO QoS or job quotas yet
- Unexpected library behaviors on SLURM
- A hard limit of 24 hours on walltime per job
- Job privacy is NOT enabled yet

Resource on SLUMR will be available until Jan 31th, 2016.

SLURM Overview

LSF	SLURM	Function
	sinfo	Cluster status
bjobs	squeue	Job status
bsub	sbatch	Job submission
bkill [JOB_ID]	scancle [JOB_ID]	Job deletion

`sinfo`: check cluster status

Host state: `drain`(something wrong), `alloc`(in use), idle, down.

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
cpu*         up 1-00:00:00      1  drain node001
cpu*         up 1-00:00:00     31  alloc node[002-032]
gpu          up 1-00:00:00      4  alloc gpu[47-50]
fat          up 1-00:00:00      2  alloc fat[19-20]
k40          up 1-00:00:00      2  alloc mic[01-02]
k40          up 1-00:00:00      2   idle mic[03-04]
fail         up 2-00:00:00      1  down* node222
```

squeue: check job status

Job status: R(Running), PD(Pending).

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
 2402       fat   add_upc  hpctheo PD       0:00      2 (Resources)
 2313       cpu    hbn310    physh  R   23:49:00      2 node[003,008]
```

Prepare to submit a job

Make sure you know:

- which partition or queue to use.
- how many CPU cores in total
- how many CPU cores on each host
- whether GPUs are requried
- expected runtime at max

`sbatch` usage

SLURM

```
sbatch jobsript.slurm
```

v.s. LSF

```
bsub < jobscript.lsf
```

`sbatch` options

LSF	SLURM	Meaning
`-n [count]`	`-n [count]`	Total processes
`-R "span[ptile=count]"`	`--ntasks-per-node=[count]`	Processes per host
`-q [queue]`	`-p [partition]`	Job queue/partitio
`-J [name]`	`--job-name=[name]`	Job name
`-o [file_name]`	`--output=[file_name]`	Standard output fi
`-e [file_name]`	`--error=[file_name]`	Standard error file
`-W [hh:mm:ss]`	`--time=[dd-hh:mm:ss]`	Max walltime

`sbatch` options (continued)

LSF	SLURM	Meaning
`-x`	`--exclusive`	Use the hosts exclusive
	`-mail-type=[type]`	Notification type
`-u [mail_address]`	`--mail-user=[mail_address]`	Email for notification
	`--nodelists=[nodes]`	Job host preference
	`--exclude=[nodes]`	Job host to avoid
	`--depend=[state:job_id]`	Job dependency

A `sbatch` example (CPU)

```
#SBATCH --job-name=LINPACK
#SBATCH --partition=cpu
#SBATCH -n 64
#SBATCH --ntasks-per-node=16
#SBATCH --mail-type=end
#SBATCH --mail-user=YOU@EMAIL.COM
#SBATCH --output=%j.out
#SBATCH --error=%j.err
#SBATCH --time=00:20:00
```

A `sbatch` example (GPU)

```
#SBATCH --job-name=GPU_HPL
#SBATCH --partition=k40
#SBATCH -n 4
#SBATCH --ntasks-per-node=2
#SBATCH --exclusive
#SBATCH --mail-type=end
#SBATCH --mail-user=YOU@MAIL.COM
#SBATCH --output=%j.out
#SBATCH --error=%j.err
#SBATCH --time=00:30:00
```

Part III: Smart environment modules

Fresh new modules for SLURM: `/lustre/usr/modulefiles/pi`

- Loaded automatically on `mu07` login node
- Smart enough to derive the combanation of compilers and MPI libs
- The NO. of modules is still growiing
- Sampels for job submission: `/lustre/utility/pi-code-sample`

Simplified module loading

```
$ module load gcc/4.9 openmpi/gcc49/1.8 fftw3/gcc49/openmpi8/3.3
```

v.s.

```
$ module load gcc/4.9 openmpi/1.8 fftw3/3.3
```

v.s.

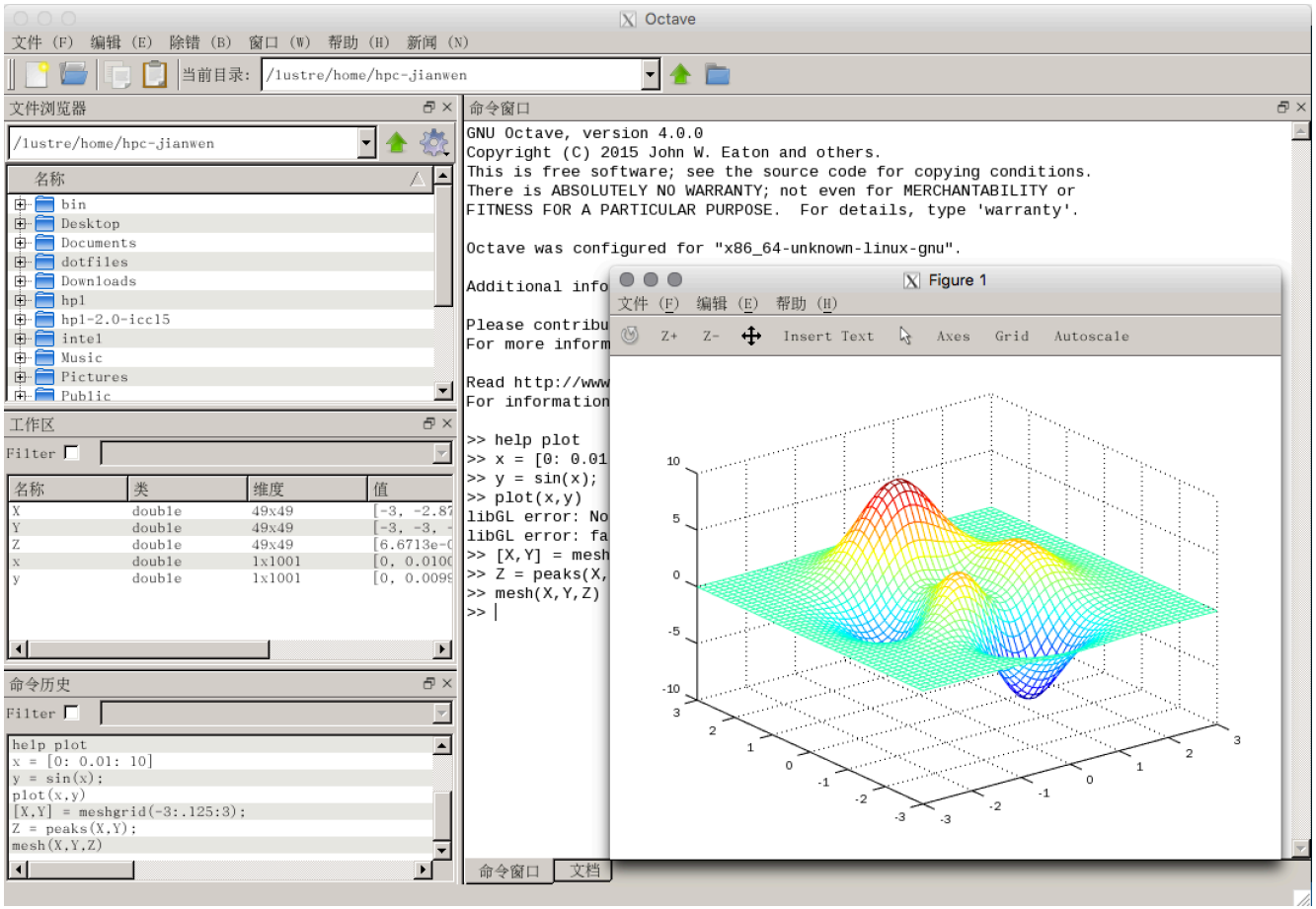
```
$ module load gcc openmpi fftw3
```

Optimizedly built libraries and software

```
gcc icc jdk perl python R pgi
impi openmpi mvapich2 mpich
mkl atlas lapack openblas mpc gmp mpfr gsl eigen
abysss samtools smufin gatk maq bwa bowtie
openfoam cgal gromacs
hdf5 netcdf scotch ffmpeg szip
cuda(cublas included) cudnn caffe
```

Opensource alternatives of MATLAB and IDL

- GNU Octave `https://www.gnu.org/software/octave/`
- GNU Data Language (GDL)



Part IV: Tips for monitoring your jobs

Wait, is my application running well?

You can cofirm your applicaion's state by:

- Comparing performance between π and your laptop.
- Comparing performance betwen π and existing traces or benchmarks.
- Monitor the applicaiton, compute nodes more exactly, via `http://pi.sjtu.edu.cn/ganglia`.
- Asking administrators `support@lists.hpc.sjtu.edu.cn` for help.

Communicate with HPC administrators

Yelling "XXX is slow" doesn't help. Please report the following information:

- Job ID;
- Website of your application;
- Expected results and what actually happened;

Email `support@lists.hpc.sjtu.edu.cn` is always preferable over phone.

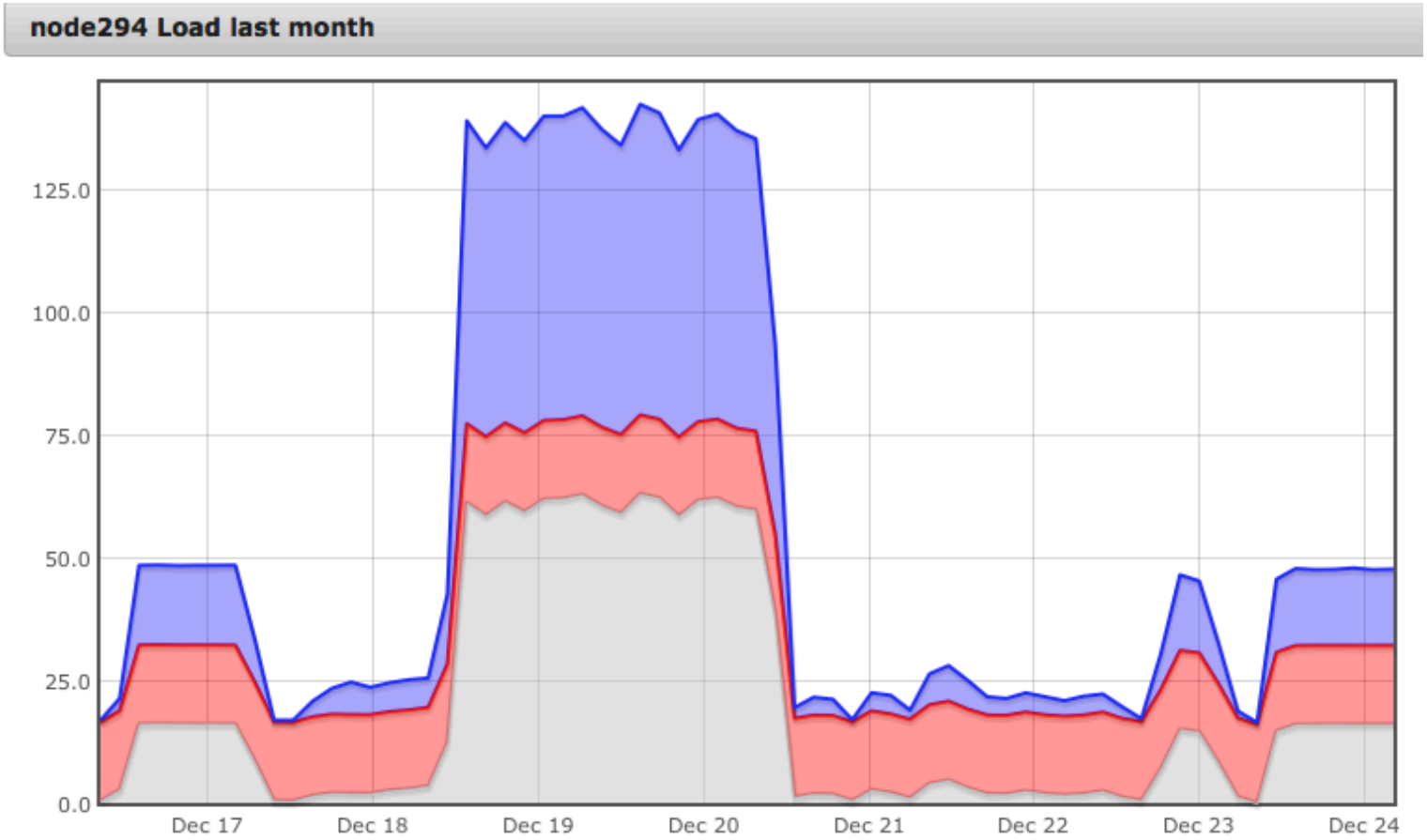
Monitor what? Load, CPU, Mem, Network

Please check `http://pi.sjtu.edu.cn/ganglia`:

- Load: The number of "threads", should be approximately 16 – the number of CPU cores
  - Below 16: starving
  - Above 16: overload
- CPU report: Charts in yellow color are good
  - `sys` and `wait` should be less than 5%.
- Mem: Do NOT exceed the physical capapcity
- Network: Ethernet traffic should be less than 1MB/s.

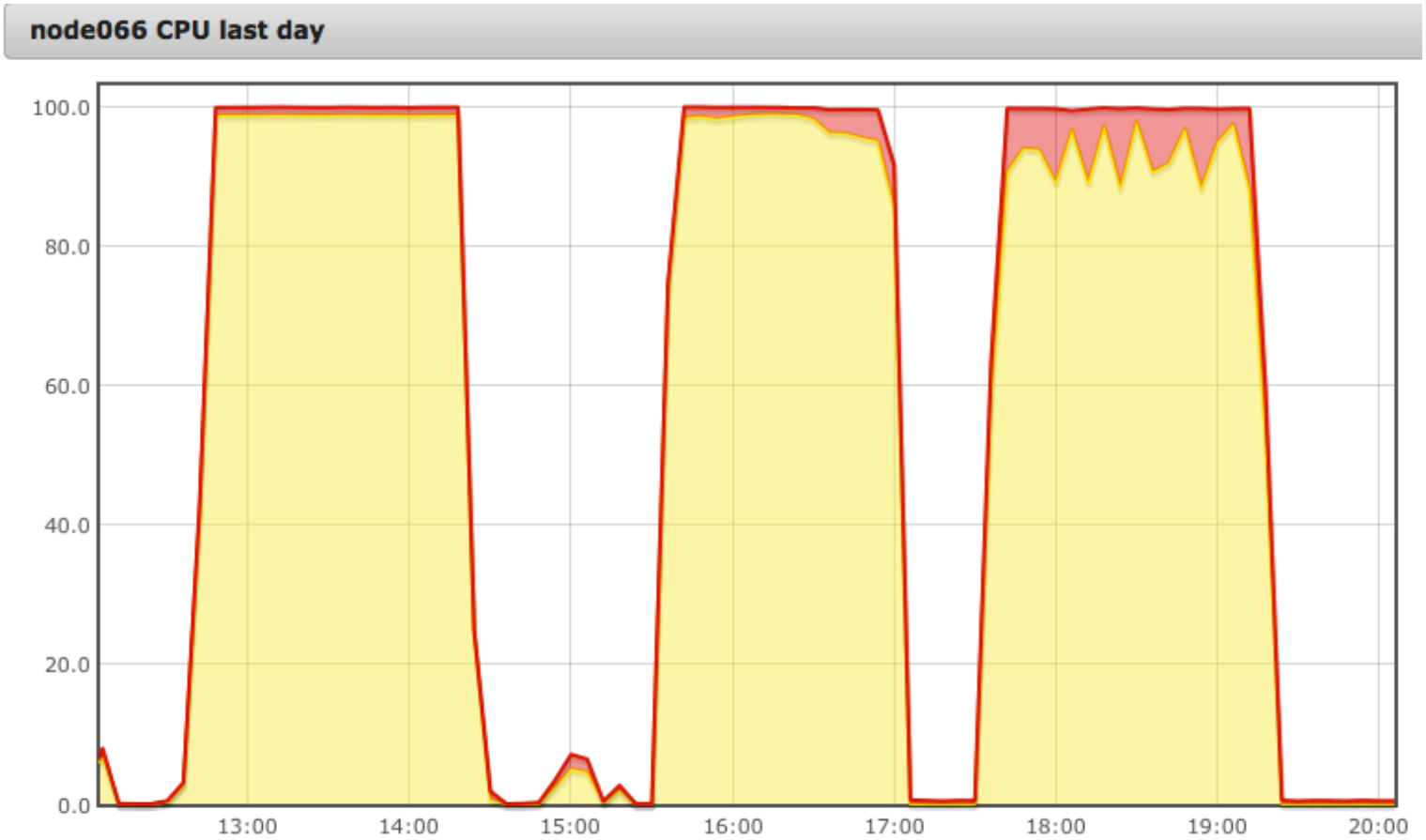
Case 1: Overload due to too many processes

Caused by incorrect setting of NO. of cores, or inbalanced load between nodes.



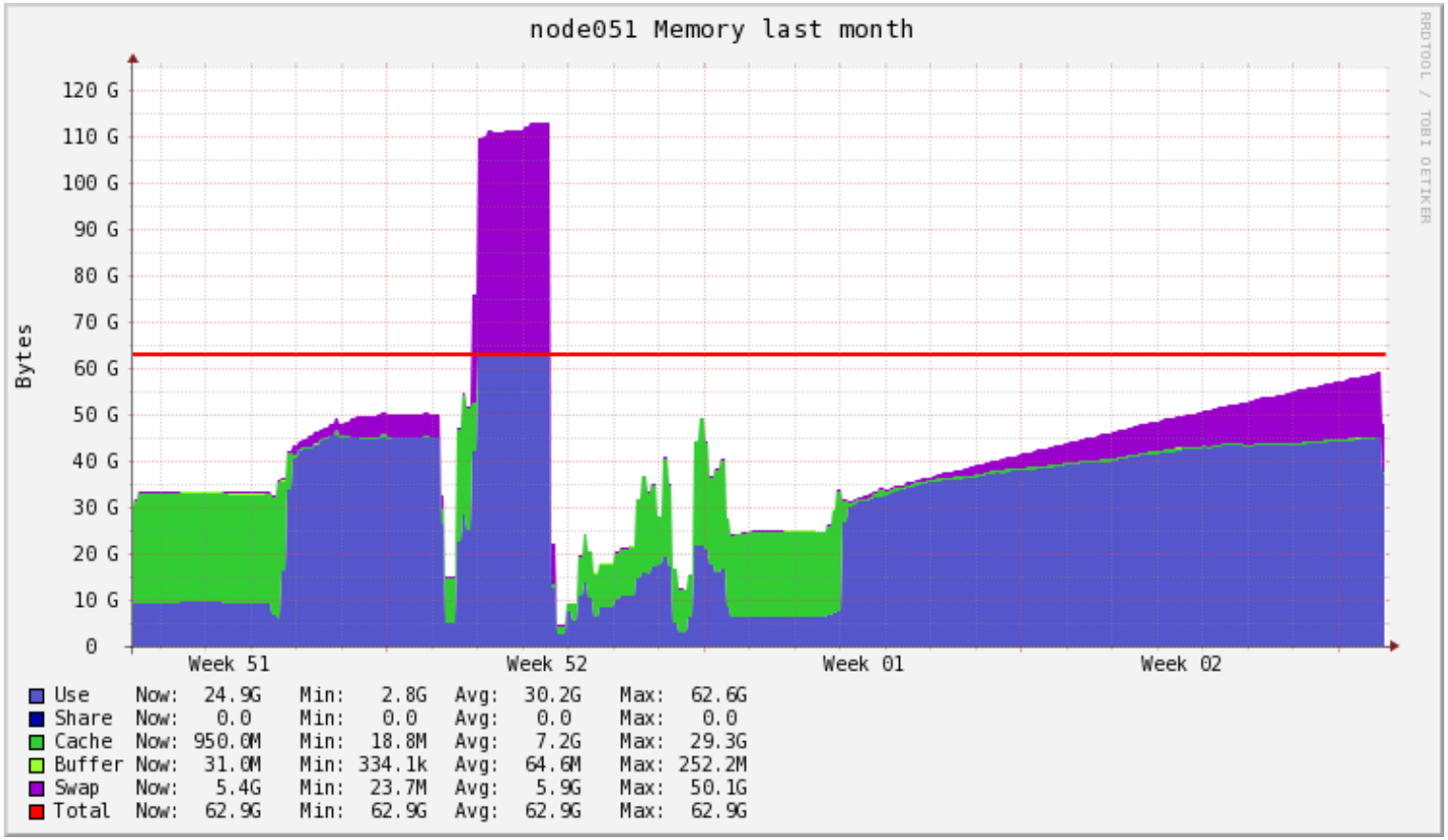
Case 2: Too high `sys` utilization

Caused by linking or loading incorrect MPI libraries, or hardware issue.



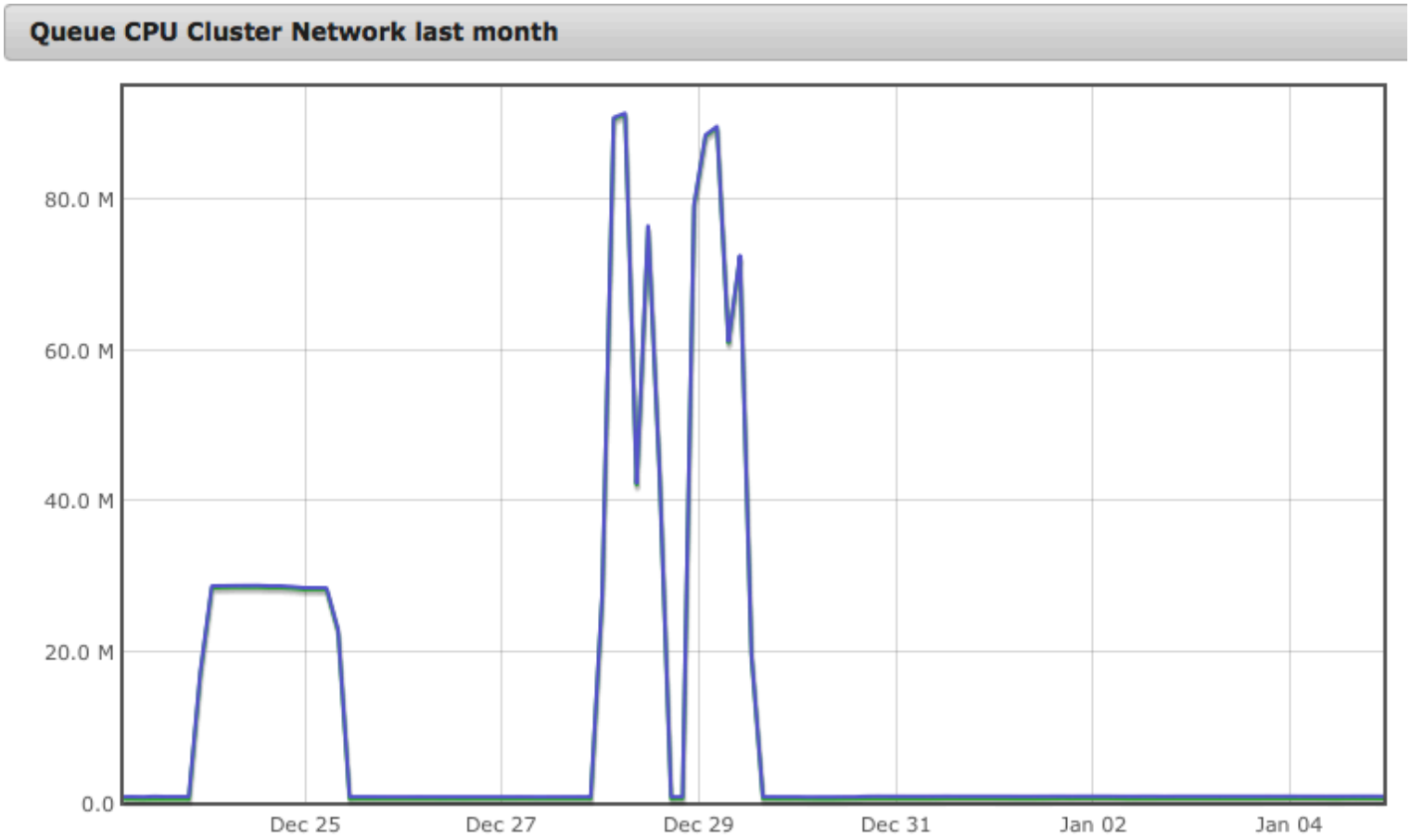
Case 3: Memory Usage Exceeding

The data is just too *fat*. Try the `fat` queue please.



Case 4: Inefficient Use of Ethernet

Caused by linking or loading incorrect MPI libraries, or Infiniband driver issue.



A workaround: *export I_MPI_FABRICS=shm:dapl*

Reference

- *SJTU π documents* `http://pi.sjtu.edu.cn/pi`
- *ACCRE's SLURM Documentation*
  `http://www.accre.vanderbilt.edu/?page_id=2154`
- *Job samples for Pi supercomputer*
  `http://pi.sjtu.edu.cn/doc/samples/`
- *Remote Desktop via NoMachine*
  `http://pi.sjtu.edu.cn/doc/rdp/`
- *Environment Module on Pi* `http://pi.sjtu.edu.cn/doc/module/`