Get Performance Boost without Code Modification
Use π supercomputer efficiently

Jianwen Wei
weijianwen@sjtu.edu.cn

Center for High Performance Computing, SJTU
http://hpc.sjtu.edu.cn

Jul 22, 2015

1. Part I: Run Your Applications on π
2. Part II: Application Monitoring and Naive Compiler Optimization
3. Part III: More Materials

Is it possible to get performance boost without code modification?

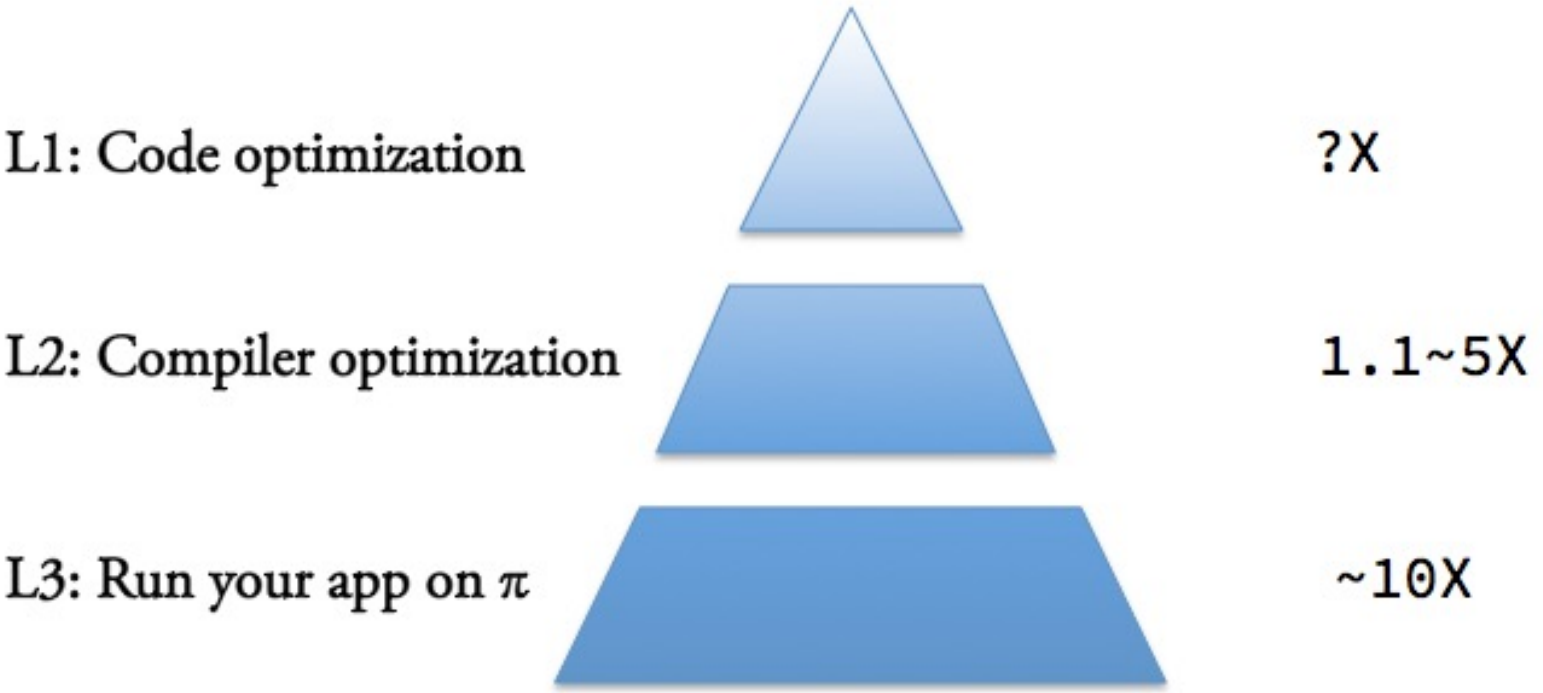
The short answer: *Yes*.

N-GenIC is an example from HPC users.

- Original version using ICC + Intel MPI + self-compiled FFTW2, takes 3 hours to complete
- Optimized version using GCC + MVAPICH2 + optimized FFTW2, takes 21 minutes to complete

9x speedup?!

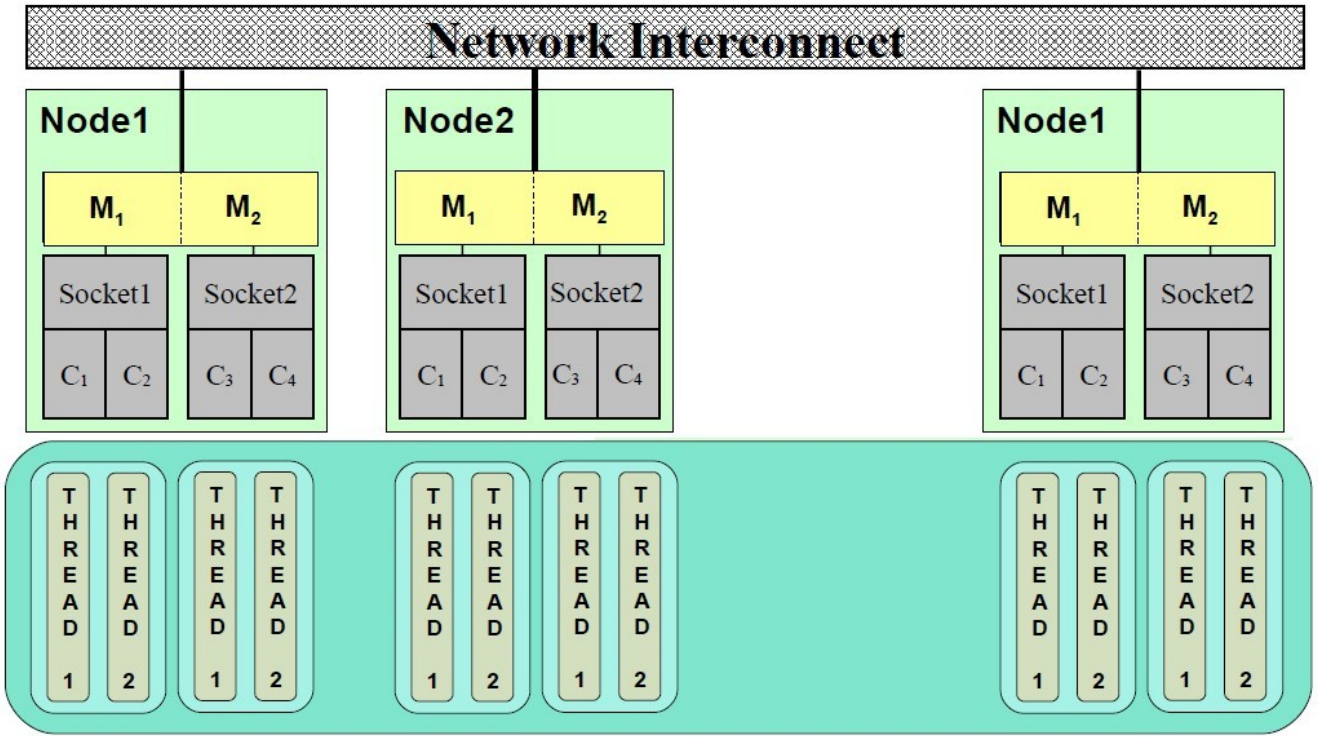
Levels of Optimization



Part I: Run Your Applications on π

SJTU π: A computer cluster

- Multiple nodes connected by ultra high speed networks
- A single computer in 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

How to Run a Job

1. Login to π via SSH
2. Pick a compiler + MPI lib combination via "module"
3. Build and install your application
4. Submit jobs via LSF
5. Log out, and wait your mail notification

Login via SSH

- Visit `http://vpn.hpc.sjtu.edu.cn` to get your IP addresss
- Wait the IP to be added into the white list
- Login via one of the followings:

```
ssh YOUR_USER@202.120.58.229
ssh YOUR_USER@202.120.58.230
ssh YOUR_USER@202.120.58.231
```

- Keep your passwords confidential
- Your IP will be banned if login fails too many times within short period. Try `ssh-copy-id` to login without password.
- Windows users: putty

Data Transfer via scp/rsync/sshfs

- `scp` copies data between your laptop and π

```
scp -r local/data YOUR_USERNAME@202.120.58.230:~/data
scp -r YOUR_USERNAME@202.120.58.230:~/data local/data
```

- `rsync` syncs data *changes*:

```
rsycn -azv local/data YOUR_USERNAME@202.120.58.230:~/data
rsync -azv YOUR_USERNAME@202.120.58.230:~/data local/data
```

- `sshfs`: Mount π's file system to your laptop
- Data transfer rate is limited by network, ~1MB/s – 15MB/s.
- Windows users: WinSCP and DeltaCopy

Use `module` to pick compilers and libraries

- *module* is a collection of environment variables.
- π has pre-compiler tools and libraries: icc, gcc, OpenMPI, GSL, FFTW, CUDA,...

```
module avail
```

- module is clean, easy to load/unload:

```
module load gcc/4.9.1 openmpi/gcc/1.6.5
module purge
module load icc/15.0.0 impi/5.0.1
```

- module can be extended in your $HOME
- Read more on http://pi.sjtu.edu.cn/docs/Module_ch

General Rules for Picking Compilers and MPI

- Always **try** the latest version (ICC 15.0, GCC 4.9.1, CUDA 7.0)
- ICC $>$ GCC
- Intel MPI $>$ MVAPICH2 $>$ OpenMPI
- **Need lots of tests.**

Build and Install Your Application

- Feel free to build and install your own application in $HOME.
- Tell us if this application is commonly used in your field.
- Unlicensed software is not supported.
- Set HTTP proxy before downloading packages within π

```
export http_proxy=http://mu04:3004
export https_proxy=http://mu04:3004
```

Referece for building software on π
http://pi.sjtu.edu.cn/docs/Installation_Guide_(zh)

Submit Jobs via LSF

π uses LSF to manage jobs.

- `bjobs`: Check job status, `RUN`, `PEND`, `EXIT`
- `bhist`: Check job history information
- `bsub`: Submit jobs
- `bkill`: Kill jobs
- `bpeek`: Check stdout and stderr
- Log files `*.out` and `*.err` will be generated when completed

More details are `http://pi.sjtu.edu.cn/docs/LSF_ch`

Limits on Users

- Users and research groups *share* compute resources.
- Disk quotas: ~1TB per user or 1M files (Soft limit).
- Max PEND jobs: adjusted dynamically, hundreds to thousands.
- Max RUN jobs: adjusted dynamically, hundreds to thousands.
- You can reserve resource for big jobs ($>$ 1024 cores).

DON'Ts on π

- NO `du` to calculate disk space.
- NO parallel jobs ($>$ 4 cores) on login nodes.
- NO generating large numbers of small files on Lustre. `/tmp` is recommended instead.
- NO Bitcoin, Dogecoin, Hacking...
- NO support from HPC team for unlicensed software.

Part II: Application Monitoring and Naive Compiler Optimization

Wait, is my application running well?

You can cofirm your applicaion's state by:

- Comparing performance between π and your laptop (No kidding!)
- 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.

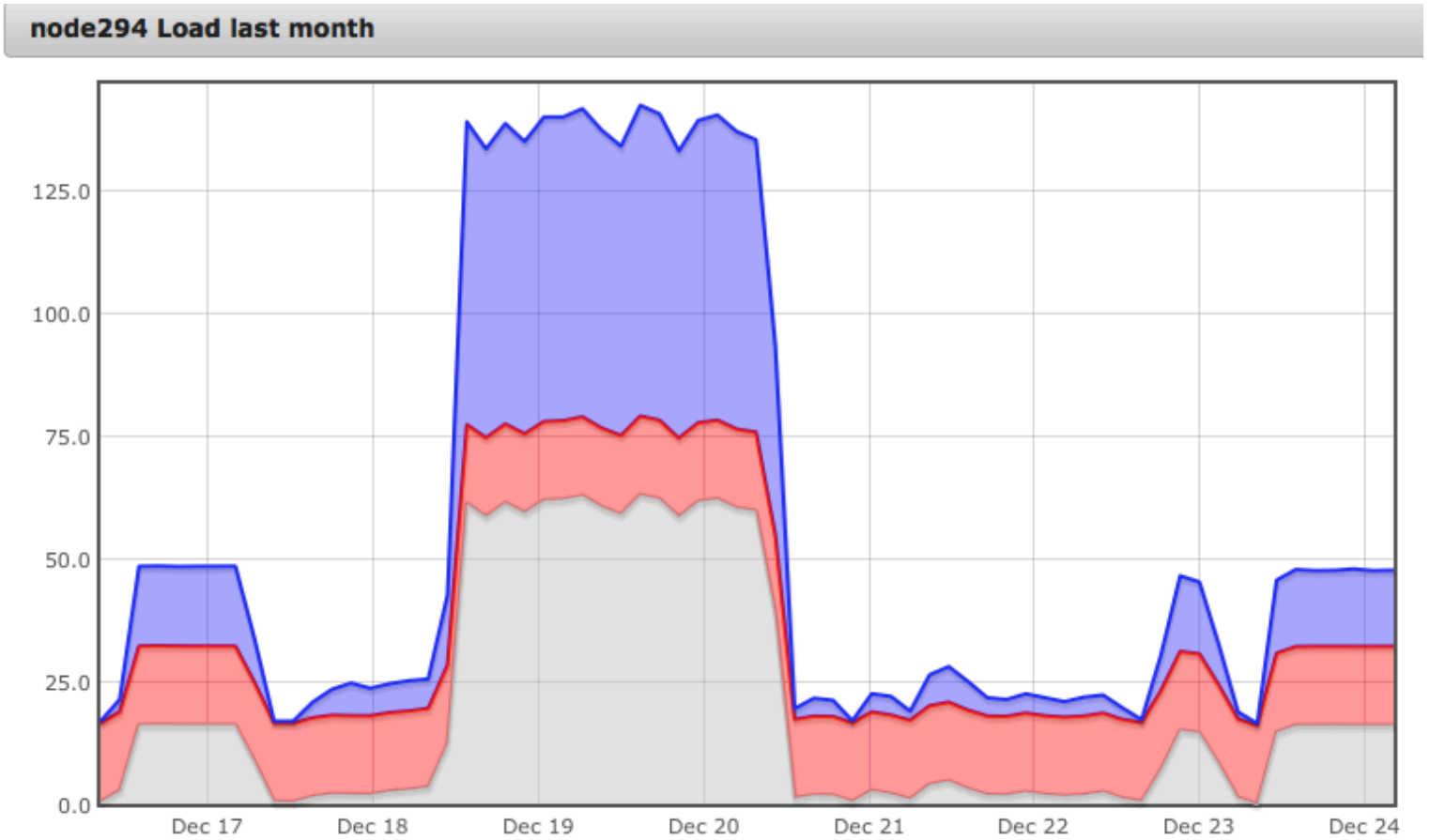
Monitor what? Load, CPU, Mem, Network

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.

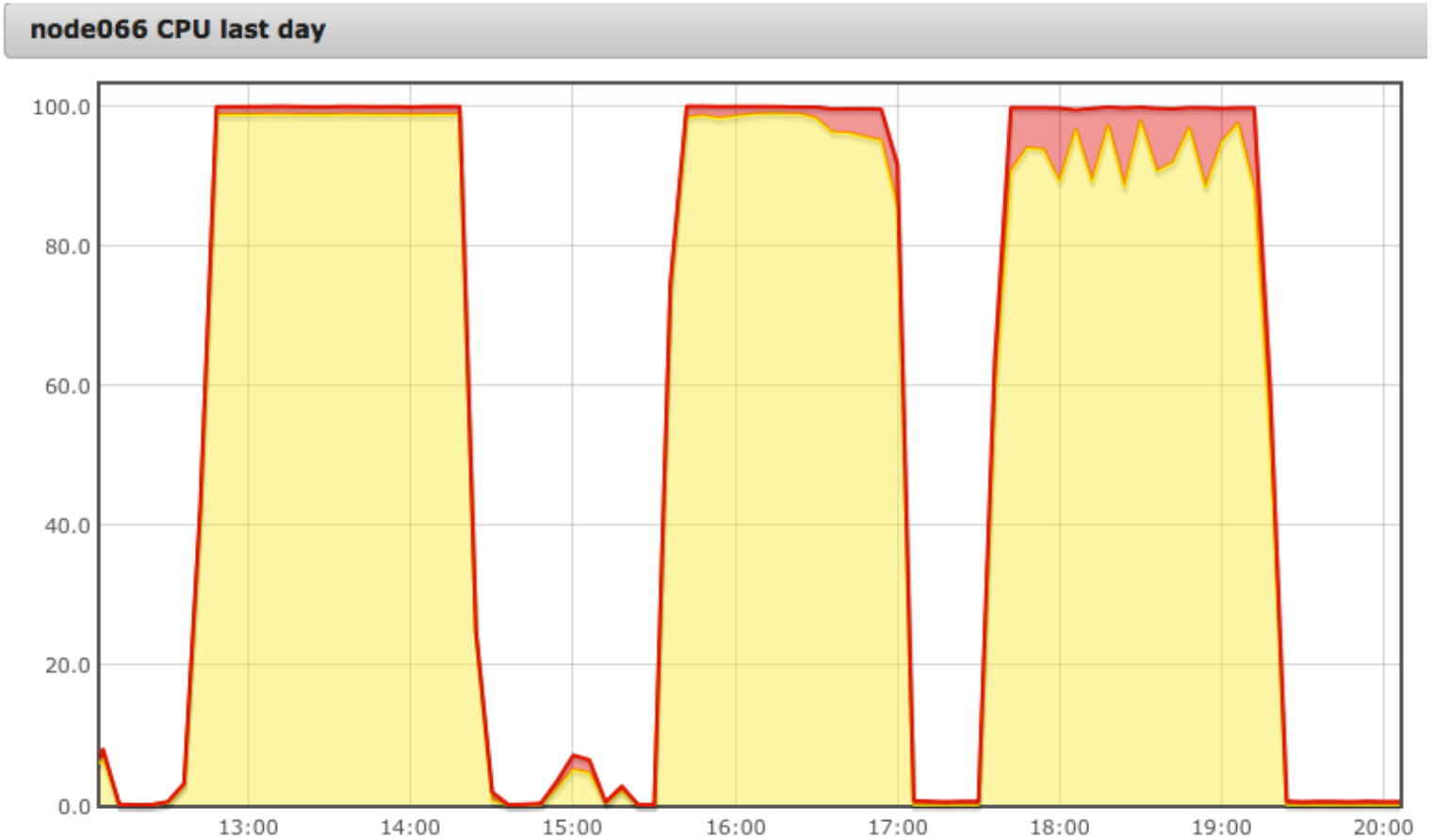
Case 1: Overload due to too many processes

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



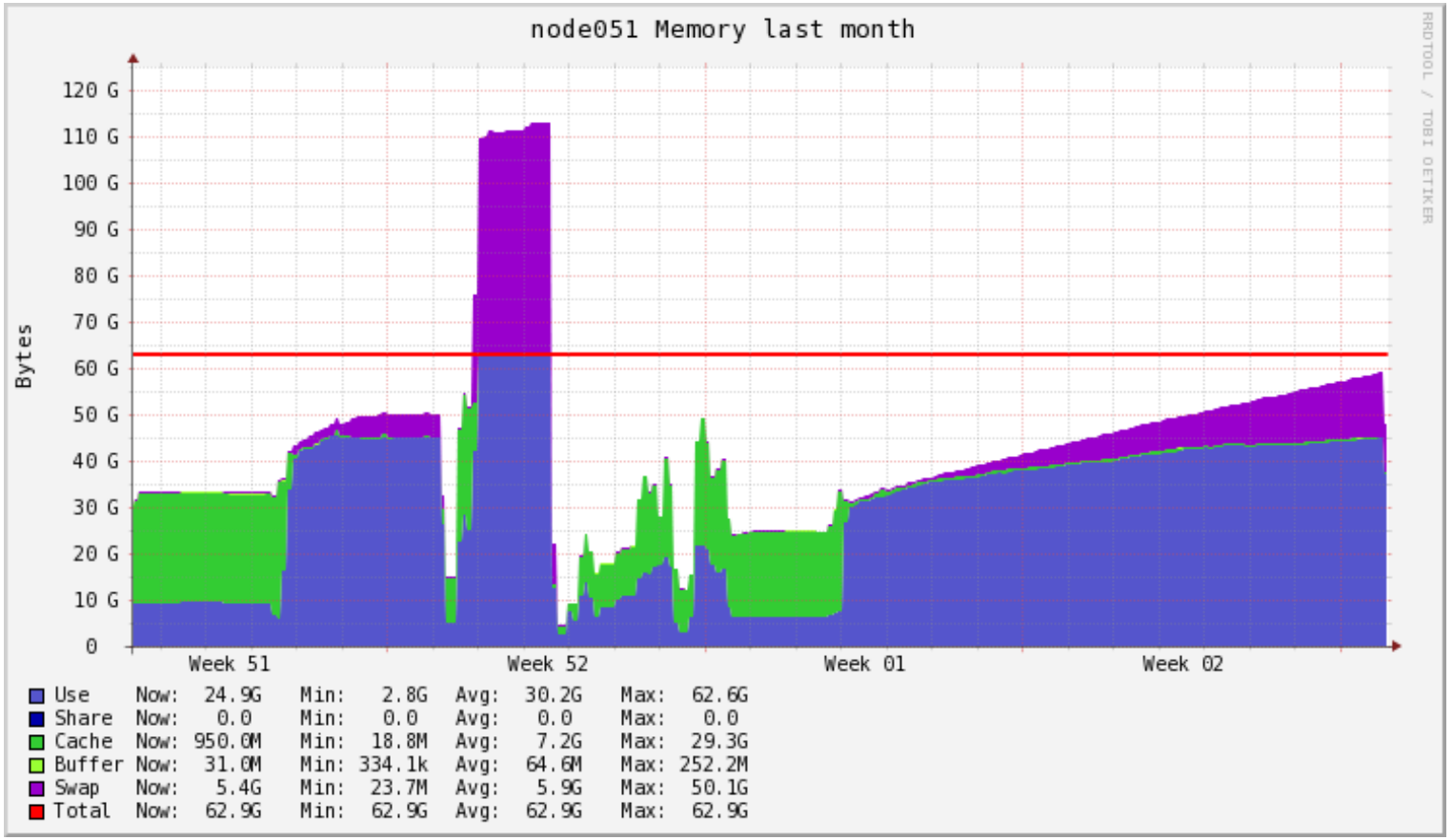
Case 2: Too high sys utilization

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



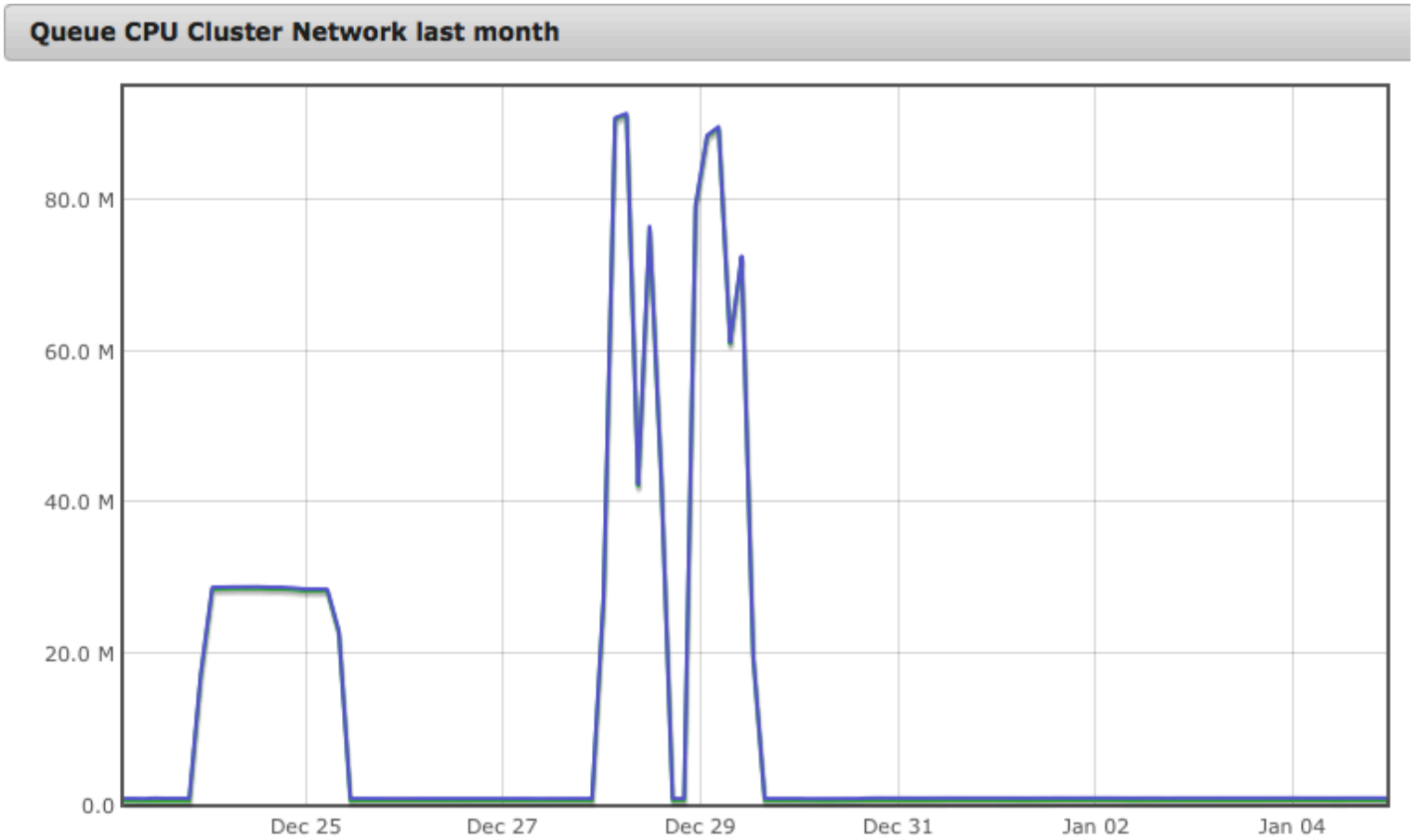
Case 3: Memory Usage Exceeding

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.



Naive Compiler Optimization Flags

Rule of thumb: Begin with simple one, raise optmization gradually, test every case.

- General optimization flag: `-O2` and `-O3`
- AVX Vectorization: `-xavx` for ICC, `-march=corei7-avx` for GCC
- Interprocedural Optimization for ICC: `-ipo`

Please refer to:

- "Quick-Reference Guide to Optimization with Intel Compilers"

Part III: More Materials

What's Next?

- Application Profiling
- Heterogenous Programming with CPU+GPU
- Data Processing

How to Work in Linux

- 鸟哥的 *Linux* 私房菜 *(*基础篇, 服务器篇*)*
  http://linux.vbird.org/
- *The Linux Programming Interface: A Linux and UNIX System Programming Handbook* http://www.amazon.com/The-Linux-Programming-Interface-Handbook/dp/1593272200/ref=pd_bxgy_14_img_y
- *RHCE Video Course*
  https://pt.sjtu.edu.cn/details.php?id=34260

How to Learn Parallel Programming

- "Using LLNL's Supercomputers"
  https://computing.llnl.gov/tutorials/agenda/index.html
- "Udacity: Introduction to Parallel Programming"
  https://www.udacity.com/course/cs344
- "Coursera: High Performance Scientific Computing"
  https://www.coursera.org/course/scicomp

π, the Hybrid Supercomputer of SJTU

- *News & Notices* http://hpc.sjtu.edu.cn/
- *Documents about How to Use* π http://pi.sjtu.edu.cn/docs/
- *Code Samples for* π
  http://pi.sjtu.edu.cn/docs/Codesample_ch
- *Maintenance Log of* π
  http://pi.sjtu.edu.cn/docs/Maintenance_Log
- *Ganglia Monitoring System* http://pi.sjtu.edu.cn/ganglia/

Center for HPC, SJTU

- *News & Notices* http://hpc.sjtu.edu.cn/
- *Apply for HPC Accounts* http://hpc.sjtu.edu.cn/Service_/GettingStarted.htm
- *Location* Room 205 @ Network & Information Center
- *Email to Us at* support@lists.hpc.edu.cn

More

- *Free lunch is over*
  http://www.gotw.ca/publications/concurrency-ddj.htm
- *Raspberry Pi Super Computer*
  http://www.raspberrypi.org/archives/tag/supercomputer
- *Setting up a Beowulf Cluster Using Open MPI on Linux*
  http://techtinkering.com/2009/12/02/setting-up-a-beowulf-cluster-using-open-mpi-on-linux/
- *Programming Massively Parallel Processors, Second Edition*
  http://www.amazon.com/Programming-Massively-Parallel-Processors-Edition/dp/0124159923