Light-weight Grid Computing for Small Group Use Cases

Aiqiang Zhang, Qimin Zhou, Xu Deng, Benda Xu

Department of Engineering Physics, Tsinghua University

zhangaq19@mails.tsinghua.edu.cn

CHEP, 4-8 November 2019

1. Introduction

In modern physics experiments, data analysis need considerable computing capacity. While several largescale grid solutions exist, for example DiRAC (Distributed Infrastructure with Remote Agent Control) [1], there are few schemes devoted to solve the problem at small-scale.

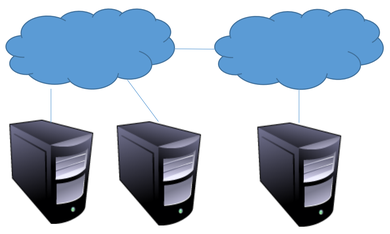


Figure 1: small scale group

Requirement:
1.distribute filesystem
2. job management
3. maintainable without support team

limitation:
1. several hosts, number range 2~4
2. different hosts in private isolated network

For the cases when light-weight grid computing is more desirable, our project provides a simple solution in connecting and managing the distributive computing resources of a small group. The components are all freely available as official Debian packages.

2. Overview

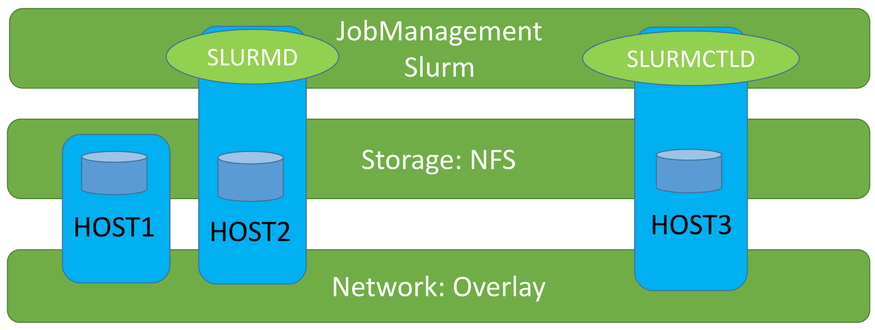
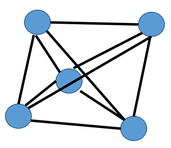


Figure 2: Concepts maps of our solution

The system is mainly consisted by Network layer, Storage layer, Management layer. The network layer ensures connectivity of different hosts. Storage layer supplies as distributed storage. Management layer controls the job schedule and allocation.
This system is designed for small grid computing. All components could be installed easily via repository and configuration is simple.

2.1 Network: Overlay



Tinc + Babel

Tinc is a VPN daemon that use tunneling to create private network.
Babel is a loop-avoiding distance-vector routing protocol for IPv6 and IPv4 with fast convergence properties[2]. It limits the duration of routing pathologies.

Figure 3: The overlay network

We use tinc to build the backbone network. Each node in the network is equal and the network is decentralized, which means the whole network system is fault-tolerant.
All nodes measure the RTT with each other to decide the best routing strategy.
The packets delay via the overlay network is lower than public network.

We connect the hosts in private network to the backbone network, which means the more computing resources in different networks could be used.

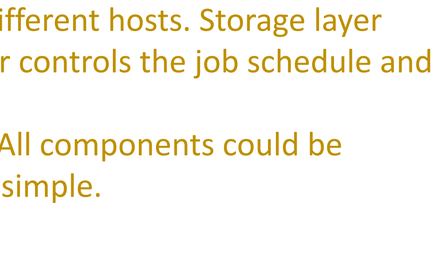
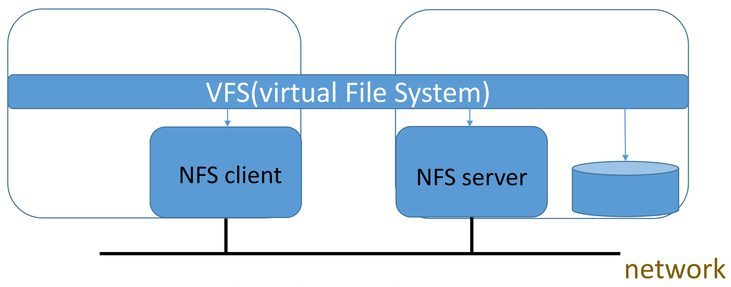


Figure 4: private and backbone network

- tinc and babel are installed in every nodes.
- network is fault-tolerated and has low latency.
- hosts in private network could be connected.

2.2 Storage: NFS

NFS (Network File System) was developed to allow machines to mount a disk partition on a remote machine as if it were a local disk.[4]



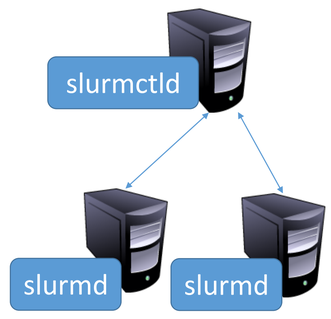
NFS supports a same virual File System for all hosts in the distributed system.

Figure 5: The schema of NFS

- run NFS server on the storage sharing host
- run NFS client on the hosts which can mount the shared filesystem

2.3 JobManager: Slurm

Slurm is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters.[3]



- Slurm can allocate jobs in different cores and nodes.
- It is easy to monitor the process of each jobs in slurm.

Figure 6: The schema of Slurm

- slurmd daemon runs on each computing node
- slurmctld daemon runs on management node

3. Performance

ENVIRONMENT:
- The system is setup with 2 hosts in different private network.
- The NFS server and Slurm controller run on the same node.
- Server CPU: 32 cores, Intel Xeon E5-2630
- Client CPU: 16 cores, Intel Xeon E5-620

3.1 Network Performance

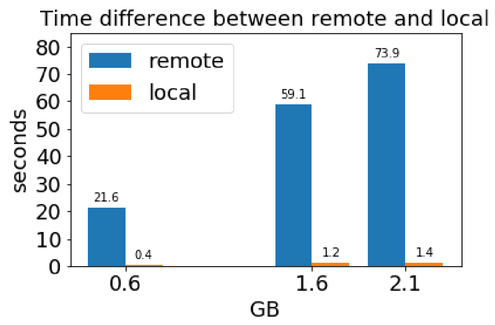


Figure 6 shows a time cost comparison between local file copy and remote file copy. The different size of files cost different time to copy.

Copying file use NFS need network transport, which leads to more time delay.

Figure 7: Time delay between local and NFS

When a file is first used, the file will cached.[4] This feature ensures the network delay influence will reduce when using proper job allocate strategy.

3.2 System Performance

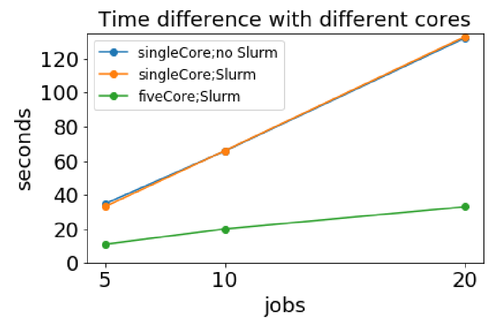


Figure 7 shows an experiment to measure the data processing time using single core without Slurm, single core with Slurm, five cores with Slurm.
In the experiment, the job is to process many waveform files (the waveform generated from PMT voltage readout).

Figure 8: Performance in difference circumstances

The same job cost similar time whether using Slurm or not, which means the job allocation time is not significant when handling few jobs.
Splitting job into different independent tasks on Slurm accelerates the speed of jobs.

4 Conclusions

- The NFS cost a lot of time in the whole system.
- It is faster to use Slurm on more cores than single core.
- A unified interface is achievable with a few available packages from popular GNU/Linux distributions.

5 Upcoming Research

- Compare NFS with other distribute filesystem.
- Find the best strategy to allocate jobs to reduce the delay from network.
- Compare performance with other grid solutions, such as DIRAC.

References

[1] Fifield, Tom, et al. "Integration of cloud, grid and local cluster resources with DIRAC." *Journal of Physics: Conference Series*. Vol. 331. No. 6. IOP Publishing, 2011.
[2] Juliusz Chroboczek. The Babel Routing Protocol. RFC 6126. Apr. 2011. doi: 10.17487/RFC6126. url: https://rfc-editor.org/rfc/rfc6126.txt.
[3] ”Slurm Workload Manager”. 2019. Web. Oct 2019. https://slurm.schedmd.com/overview.html.
[4] Ming Chen, Dean Hildebrand, Geoff Kuenning, Soujanya Shankaranarayana, Bharat Singh, and Erez Zadok. 2015. Newer Is Sometimes Better: An Evaluation of NFSv4.1. SIGMETRICS Perform. Eval. Rev. 43, 1 (June 2015), 165-176. DOI=http://dx.doi.org/10.1145/2796314.274584

Acknowledgements

This work is support by SRT (Student Research Training) of Tsinghua University. The whole project is mentored by Professor Benda Xu. Qimin Zhou and Xu Deng did the main part of this project with me.