Lightweight Grid Computing for Small Group Use Cases

Aiqiang Zhang[1,], *Xu* Deng[1,], *Qimin* Zhou[1,], and *Benda* Xu[1,2,3,*]

[1]Department of Engineering Physics, Tsinghua University, Beijing 100084, China
[2]Key Laboratory of Particle & Radiation Imaging (Tsinghua University), Ministry of Education, Beijing 100084, China
[3]Center for High Energy Physics, Tsinghua University, Beijing 100084, China

Abstract. Grid computing provides important data analysis infrastructure for many physics experiments. Small groups may only have several hosts that may exist in different private isolated networks. The existing solutions are complex or heavy to run for small groups. Our project focuses on small-scale grid computing, using several official Debian packages, to construct the whole system. The system is designed to be lightweight and scalable.
Keyword: grid computing, lightweight, small group

1 Introduction

In modern physics experiments, data analysis needs considerable computing capacity. Computing resources of a single site are often limited and distributed computing is usually cost effective and flexible. While several large-scale grid solutions exist, for example, DIRAC (Distributed Infrastructure with Remote Agent Control) [1], there are few schemes devoted to solving the problem at small-scale.

There are some key components to small-scale grid solutions. The solutions should include a distributed filesystem and job management. The whole system should be easy to use so that the grid can be maintained without a dedicated support team. In addition, the solution should work when several different hosts scatter in many private isolated networks.

For the cases when lightweight grid computing is more desirable, our project provides a performant 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

As illustrated in Figure 1, the system mainly consists of the network layer, the storage layer and the management layer. The network layer ensures the connectivity of different hosts. The storage layer offers a remote filesystem. The management layer controls the job schedule and allocation.

This system is designed for small grid computing. All components should be installed from repositories of GNU/Linux distributions and configuration should be simple. Fine-grained authentication is not necessary.

*e-mail: orv@tsinghua.edu.cn

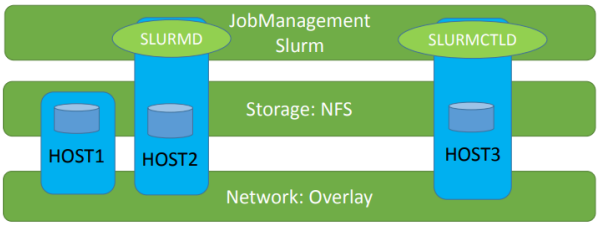

Figure 1. Schematics of the solution of three layers of the whole system are shown.

2.1 Network: Overlay

The network layer is constructed with two packages, Tinc and Babel. Tinc is a daemon that uses tunneling to create private networks [2]. Babel is a loop-avoiding distance-vector routing protocol for IPv6 and IPv4 with fast convergence properties [3]. It limits the duration of routing pathologies.

Our project uses Tinc to build the backbone network. Each node in the network is equal and the network is decentralized, making the whole network system fault-tolerant. All nodes measure the RTT with each other to decide the best routing paths. The packets delay via the overlay network is lower than the public network.

Hosts in private networks are connected to the backbone network, making the computing resources in different networks point-to-point reachable without NAT (network address translation). In our experimental network, there are two servers named *bernoulli* and *nova* selected to be the nodes of the computing grid.

2.2 Storage: NFS

NFS is the abbrivation of Network File System [4] [5]. NFS was developed to allow machines to mount a disk partition on a remote machine as if it were local. The resources on the remote hosts can be transparently accessed via NFS as if on the same disk. This system acts as a virtual storage layer. It is useful when doing tasks on different systems at the same time.

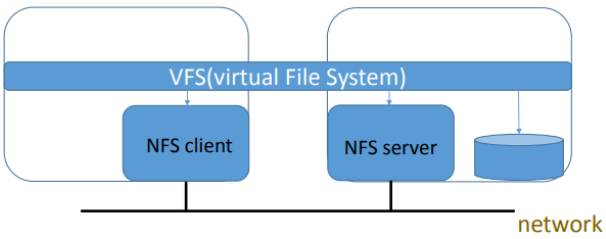

Figure 2. Illustration of NFS offering a shared view of vitual file system to client and server.

Figure 2 show the architecture of NFS. VFS handles local and remote files via a common interface. The RPC (remote procedure call) transport the data through the network. Both *bernoulli* the server and *nova* the client can access the vitual filesystem transparently [6].

2.3 JobManager: Slurm

Slurm is an open-source, fault-tolerant, and highly scalable cluster management and job scheduling system. It provides a framework for starting, executing, and monitoring work on the allocated nodes. Besides, Slurm manages a queue of pending work to balance the allocation of resources [7]. The features of Slurm satisfy the requirements of small-scale grid solutions(see Sect. 1).

As depicted in Figure 3, Slurm consists of a slurmd daemon running on each compute node and a central slurmctld daemon running on a management node [7]. Slurm uses server-client mode to communicate among the slurmd daemons and the slurmctld daemon. All of the user commands can run anywhere in the cluster.

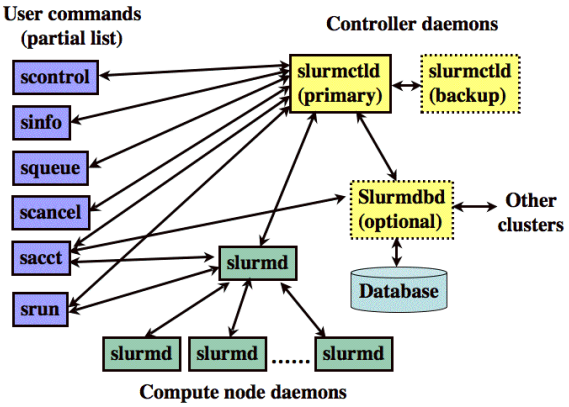

Figure 3. Slurm schemas and communication process [7]. Slurm uses server-client mode to manage the job schedule.

For our setup, slurm provides us high performance, scalability and fault tolerance. Members of our group can submit and correct tasks anytime and anywhere from the Internet.

3 Perfomance

The small-scale grid solution is tested on two hosts in different private networks. NFS server and Slurm controller both run on the host named *bernoulli* which acts as a server, while NFS client and Slurm worker daemon both run on another host named *nova* which act as a client. There are 16 Intel Xeon E5-2630 cores on the server, and 8 Intel Xeon E5-620 cores on the client.

3.1 Network Performance

Figure 4 shows the time consumption comparison between local and remote file copying. The different sizes of files cost different amounts of time to copy.

Copying files in NFS need network transportation, which leads to additional time delay. As illustrated by Figure 4, time consumption on network transportation is about 30 seconds per gigabyte. Computing grid always encounters the difficulty of copying large file among different hosts which is unbearable if network transportation costs too much time.

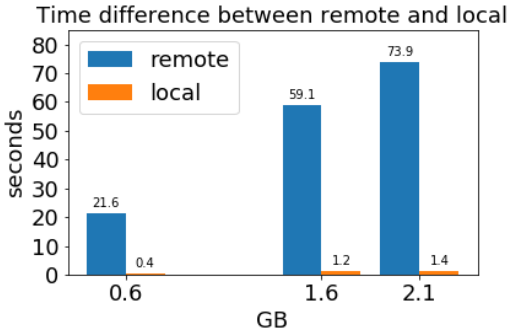

Figure 4. NFS performance. The orange bars show the time consumption of local filesystem reading, while the blue bars illustrate the time consumption of remote filesystem reading. The x-axis gives different file sizes.

Fortunately, when a file is first used, the file can be cached automatically [8]. This feature ensures the network delay influence to be decreased with some proper job allocation strategy.

3.2 System Performance

Figure 5 shows an experiment to measure the data processing time using a single core without Slurm, single core with Slurm, and five cores with Slurm.

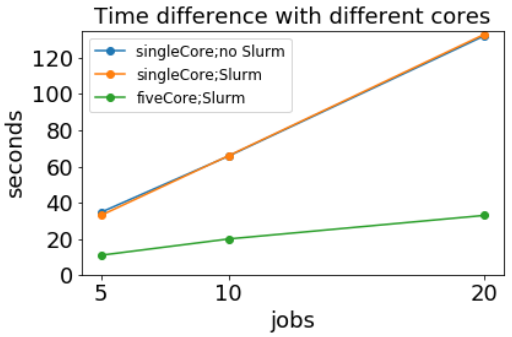

Figure 5. Illustration of Slurm performance. Overlapping blue and orange lines show that there is nearly no noticeable overhead cause by Slurm.

The test case is composed of jobs to process PMT voltage waveform readouts from a liquid scintillator experiment. The same job costed similar time whether using Slurm or not, which means the job allocation time is not significant when handling a few jobs. Comparing the orange line and green line in Figure 5, we can find that splitting jobs into different independent tasks on Slurm will accelerate the overall processing speed.

4 Summary

The NFS costs a lot of time in the whole system when files are copied at the first time. This overhead could be reduced by the NFS cache strategy. It is faster to use Slurm on more cores

than a single core. A unified interface is achievable with just a few available packages from popular GNU/Linux distributions.

More work will be done in the future, such as comparing NFS with other distributed filesystems, finding the best strategy to allocate jobs to reduce the delay in the network, and comparing performance with other grid solutions, such as DIRAC.

Acknowledgement

This work is supported by SRT (Student Research Training) of Tsinghua University.

References

[1] Fifield, Tom, et al, Journal of Physics: Conference Series **331**, 062009 (2011)

[2] Guus Sliepen, *Tinc*, `https://tinc-vpn.org` (2020)

[3] J. Chroboczek, *The Babel Routing Protocol*, `https://rfc-editor.org/rfc/rfc6126.txt` (2011)

[4] R. Sandberg, D. Goldberg, S. Kleiman, *Design and implementation of the Sun network filesystem*, (1985)

[5] T. Haynes, *Network File System (NFS) Version 4 Protocol*, `https://tools.ietf.org/html/rfc7530` (2015)

[6] Anderson, Thomas E., et al, Proceedings of the fifteenth ACM symposium on Operating systems principles, 109-126 (1995)

[7] SchedMD LLC, *Slurm Workload Manager*, `https://slurm.schedmd.com` (2020)

[8] Ming Chen, Dean Hildebrand, et al, SIGMETRICS Perform **43**, 165-176 (2015)