Design Patterns of Scalable Cluster System Software

Bibo Tu[1,2] , Ming Zou[1,2], Jianfeng Zhan[1], Lei Wang[1] and Jianping Fan[1]

1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
2. Graduate School of Chinese Academy of Sciences, Beijing 100039, China
{tbb, zm, jfzhan, wl}@ncic.ac.cn, fan@ict.ac.cn

Abstract

The design pattern of cluster system software has an important influence on scalability of massive cluster system. The paper presents design patterns of scalable cluster system software, including scalable software topologies and optimized communication modes. These design patterns have been widely applied in Dawning series of supercomputers and some results of performance evaluation show their good scalability.

1. Introduction

According to recent TOP500 lists of supercomputer sites, massive cluster systems have become the trend in high performance computing [1], which benefits from high performance/cost ratio and innate scalability of cluster architecture. In order to provide a unified, virtual computing environment for high performance computing users, the typical cluster environment should have the following functions at least: a) monitoring service of node, network and all kinds of resources, b) load balancing among nodes, c) job scheduling and job management capabilities, d) providing transparent job execution environment [2]. The scalability of system software requires that above functions can work without incurring errors or reducing their functions when cluster scale increases. Moreover, the performance should not degrade. That is, the responding time and self overhead of the system software should merely increase within the tolerable range. However, with the expanding cluster scale, scalability problems of the system software have gotten more prominent and high overhead of system software has greatly affected the performance of cluster system.

Therefore, the scalability of cluster system software becomes more important in designing massive cluster system. This paper presents design patterns of scalable cluster system software, from two important directions, one is scalable software topologies, and the other is optimized communication modes, which have been widely applied in Dawning series of supercomputers

developed by institute of computing technology, Chinese academy of sciences, and shown good performance in scalability.

The rest of the paper is organized as follows: Section 2 analyzes different software topologies; Section 3 compares some optimized communication modes; some successful instances and their evaluations applied in Dawning series of supercomputers are presented in section 4, and conclusions are drawn in section 5.

2. Scalable Software Topologies

Above all, good scalability of cluster system software rests with its software topology. System designers have to evaluate the requirements for their particular area and pick a topology that matches their needs. This section explains and compares common software topologies in distributed system, and then puts forward several feasible, scalable software topologies based on partition technology in massive cluster system.

2.1. Taxonomy of Software Topologies

Four basic topologies are in use on distributed system or internet: centralized and decentralized, but also hierarchical and ring systems. These topologies can be used by themselves, or combined into one system creating hybrid systems [3].

Centralized systems are the most familiar form of topology, typically seen as the client/server pattern used by databases, web servers, and other simple distributed systems. Communication is completely centralized with many clients connecting directly to a single server. The primary advantage of centralized systems is their simplicity and manageability, while the drawback of centralization systems is that everything is in only one place. There is no fault tolerance, if the central server goes down, everything does. Moreover, scale is clearly limited by the capacity of the server,

COMPUTER SOCIETY

and so centralized systems are often thought of as unscalable.

A ring topology is a physical closed loop consisting of point-to-point links. Communication between the servers coordinates the sharing of the system status. This establishes a group of nodes that provide identical function to a single server but incorporate redundancy and load-balancing capabilities. Typically, ring systems consist of machines that are on the same intranet and which are owned by a single organization. The advantages of rings over centralized systems are fault tolerance and simple scalability. If a host goes down in a ring, failover logic makes it a simple matter to have another host cover the problem.

Hierarchical topologies have a tree-like structure and provide an extremely fast way of searching through information that is organized in a consistent fashion. Hierarchical systems are more fault-tolerant than centralized systems, but the root is still a single point of failure. The primary advantage of hierarchical systems is their incredible scalability -- new nodes can be added at any level to cover for too much load. This scalability is best demonstrated in DNS, the best-known hierarchical system on the Internet, which has scaled over the last 15 years from a few thousand hosts to hundreds of millions. The relative simplicity and openness of hierarchical systems, in addition to their scalability, make them a desirable option for large Internet systems.

Recently, the peer-to-peer trend has renewed interest in **decentralized systems**. Peers communicate symmetrically and have equal roles and have no single point of control. Gnutella is probably the most "pure" decentralized system used in practice today, with only a small centralized function to bootstrap a new host. Many other file-sharing systems are also designed to be decentralized, such as Freenet or OceanStore. A primary virtue of decentralized systems is their extensibility. For example, in Gnutella any node can join the network and instantly make new files available to the whole network. Decentralized systems also tend to be fault-tolerant. The failure or shutdown of any particular node does not impact the rest of the system. The scalability of decentralized systems is hard to evaluate. In theory, the more hosts you add, the more capable a decentralized network becomes. In practice, the algorithms required to keep a decentralized system coherent often carry a lot of overhead. Scalability of decentralized systems remains an active research topic.

Hybrid topologies combine multiple topologies into one system. Real-world distributed systems often have a more complex organization than any one simple topology. Systems could conceivably be built with three or more topologies combined. For example, a node might have a centralized interaction with one part of the system, while being part of a hierarchy in another part. Often, different topologies are chosen for different parts of a system to get the best of the strengths without the weaknesses.

2.2. Hybrid Topologies based on Partition Technology

For cluster management, there are some advantages to partition a big cluster into several small parts.

Partition technology may bring good scalability. The overhead from massive cluster management often limits the increasing of cluster scale, partitioning a cluster may balance it. In theory, when cluster scale enlarges, more partitions may be divided, which will make cluster system scales well. However, over many partitions will cause more complex configuration. So the tradeoff is needed how to partition a big cluster.

Partitions are also useful for establishing boundaries due to hardware constraints. For instance, a 128-node cluster may have two 64-port Ethernet switches (which are cheaper than a single 128-port switch), and as a result, running a code on nodes connected to a single switch will be more efficient than running on nodes spanning both switches.

Otherwise, partitions are beneficial to share different kinds of resources. A massive cluster system may be composed with heterogeneous nodes (different speed, OS, architecture). Usually parallel jobs will run more efficiently using homogeneous nodes. Partition technology facilitates allocating homogenous nodes to a parallel job.

Since it is an effective approach for cluster management to use partition technology, what topologies will be picked to implement it? In every partition, centralized, ring, decentralized, even hierarchical topology may all be used, but centralized topology prefers to be picked because of its simplicity and manageability. Among all partitions or on the top of them, these basic topologies can be combined arbitrarily to build a hybrid topology with more merits. Based on partition technology, we put forward three hybrid topologies as shown in Figure 1.

Centralized hybrid topology is given in Figure 1a. A massive cluster system is divided into several partitions; on the top of these partitions, a single server takes charge of controlling and coordinating them. This hybrid topology has a two-tiers architecture, every tier uses the same centralized topology. In fact, this hybrid topology may be regarded as hierarchical topology.
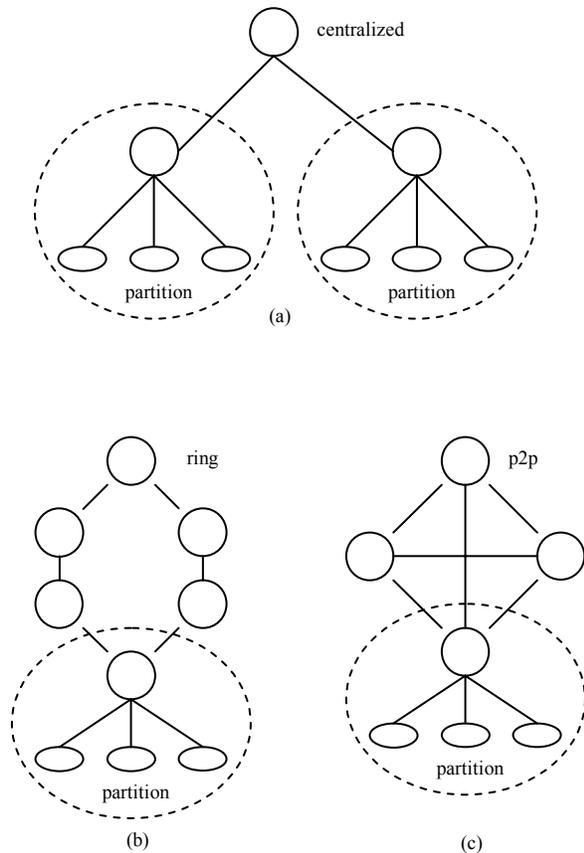
IEEE
COMPUTER
SOCIETY

Fig.1. Three hybrid topologies based on partition technology

Ring-based hybrid topology is given in Figure 1b. All partitions are organized into a closed loop. Concretely, a single server controls all nodes in the same partition, these servers establish a group with a ring fashion to provide and maintain identical information among partitions.

The last hybrid topology has a decentralized architecture with a peer-to-peer fashion among partitions as shown in Figure 1c. All partition servers are grouped into an interconnect network with a self-organizing protocol. This topology is especially suitable for passing information and sharing data among partitions.

Compared with three hybrid topologies, they more or less contribute to the extensibility, scalability and fault-tolerance of system; it is easy to add node resources or partitions to cluster systems. However, the scalability of centralized hybrid topology lies on the capacity of the root server, and the root server is still a single point of failure. But its simplicity and manageability are better than other topologies. Ring-based and peer-to-peer hybrid topologies have good ability of fault tolerance. However, to add partitions to a ring need coherence agreement among members, too

many partitions easily cause communication latency; in a sense, the scale of partitions in the peer-to-peer hybrid topology lies on the effectiveness of its self-organizing algorithm.

3. Optimized Communication Modes

Optimized communication modes can improve scalability of cluster system software. Poor communication system usually increases the response time of connection and causes congestion of too many connections. Furthermore, it may result in overload of system and failure of connection. So it is beneficial to choose suitable communication programming model and connection method for scalable cluster system software.

3.1. Communication Programming Model

There are the following communication programming models used to designing massive cluster software [4].

Blocking I/O Multiplexing (Model 1): Using select or poll system call to identify which blocking channels can be read or written. These channels with a blocking I/O fashion maybe block the process.

Non-blocking I/O Multiplexing (Model 2): Using select or poll system call to identify which unblocking channels can be read or written. This model can prevent blocking process in some situation which can happened in Model 1, for instance, too much data needs to be written into system buffer or read more data than that in system buffer.

Multiple Processes (Model 3): Forking more processes to accept and process connections or requests. Generally, multiple processes model can service requests concurrently to improve system's throughput, but forking a process needs to copy too much kernel data, which is a time-consume way and brings some overhead. Pre-forking a pool of processes is a reformative way. Moreover, synchronization efficiently among processes is also needed to consider.

Multiple Threads (Model 4): Create more threads to accept and process connections or requests. Compared with Model 3, It is a light-weight way and consumes less system resources for more threads, and synchronization among threads is more efficiency than that among processes.

The four models are often applied in distributed system, and that they can settle denial of service attacks. We measured the performance of four models by copying 2000 lines data to server host with a 175ms RTT. The results are shown in Table 1.

Table 1. Performance of four models (seconds)

Model	Model 1	Model 2	Model 3	Model 4
Time	12.3	6.9	8.7	8.5

Comparatively, apart from model 1, other three models have low overhead and better scalable performance, and the Non-blocking I/O Multiplexing demonstrates the best performance. In real world, it needs tradeoff and further performance evaluation to select suitable communication programming model.

3.2. Connection Method

In cluster monitoring system or job scheduling system, it is required to detect the status of system and resource periodically. Polling every cluster node is not wise method in massive cluster system. Thus, active and non-blocking connection methods are very feasible.

Active Connection: Polling periodically usually costs too long time and hampers the response of next event. Furthermore, if a node breaks down, polling procedure will be blocked until time-out occurs, which will induce service suspend, even crash. Compared with polling method (passive connection for every node), active connection method requires that every cluster node actively connects server host (service node) and reports its status. For service node, connections from every cluster node are sporadic and scattered, which will not result in much too long connection time and block next service, and that failed nodes will not affect continuous service of server host.

Non-blocking Connection: The use of a non-blocking connection avoids a long time-out if any cluster node is down or slow. Moreover, a non-blocking connection will get the most of processing cycles compared with a blocking connection.

Otherwise, it is beneficial for scalability of massive cluster system to modify some connection parameters such as buffer size, fd_size, listening backlog and so on and use reliable udp connection or broadcast technology.

4. Several Instances and their Evaluations

According to design patterns of scalable system software, several instances such as Phoenix Cluster Management System (Phoenix), Partitioned Workload Solution (PWS), Dawning MPI Loader (DMLoader) have successfully been deployed in Dawning 4000A supercomputer and have demonstrated good scalability. In Dawning series of supercomputers, Dawning 4000A supercomputer is the latest massive cluster system with 640 nodes. Each node is a 4-way NUMA with AMD Opeteron 64bit processors and SuseLinux operating

system. With about 11.2Tflops maximum peak performance and about 8Tflops real Linpack performance, Dawning-4000A was ranked No. 10 of TOP500 supercomputer sites in the list of June 2004.

4.1. Phoenix

Phoenix aims to provide a unified, scalable, high available cluster management system with the supports of heterogeneous nodes and self-management capacity. It provides a minimum set of cluster core functions with scalability and fault-tolerance support [5]. Group service is the kernel component in Phoenix to solve scalability and high availability as shown in Figure 2. For high scalability, ring-based hybrid topology with partition technology and active connection method are adopted. In Phoenix, the whole cluster is divided into several partitions. In every partition, there is only one group service daemon (gsd) which collects heartbeats periodically from watching daemons (wd) running in every cluster node. All group service daemons form a meta-group through a ring-based membership protocol. Thus, group service with partition support can not only support massive scale nodes, but also detect nodes' failures rapidly and recovery them as possible.
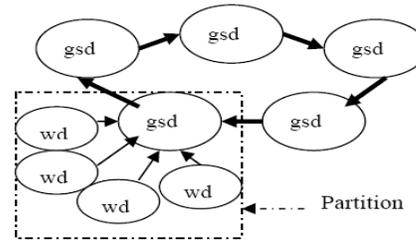

Fig.2. Scalable Group Service in Phoenix

Table 2. Phoenix's Impact on Linpack Benchmark Performance

Nodes(CPUs)	Without Phoenix running	With Phoenix running
64 (256)	79.83%	78.12%
128 (512)	78.74%	73.25%
256 (1024)	75.94%	70.38%
640 (2560)	71.56%	68.59%

We measure the performance impacts of Phoenix on cluster system under 64, 128, 256 and 640 nodes respectively using the Linpack benchmark program. From the Table 2, it can be concluded that Phoenix has little impact on cluster system with the support of software scalability.

4.2. PWS

Partitioned Workload Solution (PWS) is a job management system (JMS) with high scalability and manageability on Dawning 4000A supercomputer. Based on OpenPBS software (an open source project) [6], a new approach of partitioning cluster and enabling dynamic resource lease among different partitions is adopted by PWS as shown in Figure 3. This is a centralized hybrid topology, which facilitates the functional design of job scheduling and resource management. Moreover, instead for polling periodically and blocking connection method in OpenPBS software, PWS optimizes its communication modes through non-blocking I/O multiplexing model and active connection method. These design patterns make PWS support massive cluster systems with thousands of CPUs or nodes.
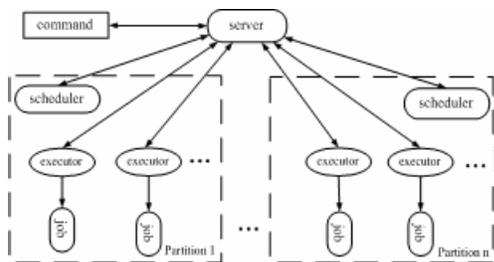

Fig.3. The topology of PWS

Table 3. Compared with PWS and OpenPBS (milliseconds)

JMS	Shortest Response Time	Longest Response Time	Average Response Time
PWS	12	1970	78
OpenPBS	11	30000 (timeout)	328

We measure shortest, longest and average response time of PWS and OpenPBS under the heaviest load on Dawning 4000A supercomputer. That is, to submit thousands of small jobs, each of which occupies only one CPU and runs between 30 seconds to 5 minutes. When the system is filled with those small jobs, there are more than 2000 jobs running simultaneously and thousands of jobs waiting for scheduling in queues. The measure results are shown in Table 3. The results make it clear that scalability of PWS gets improved more greatly.

4.3. DMLoader

Dawning MPI Loader, a software tool called DMLoader, is used to load DMPI tasks on Dawning Supercomputers. There are two different modes to load MPI tasks, one is based on remote commands, and the other is based on daemons. The former is used in most of MPI implementations which by default use rsh or ssh to provide remote process startup, such as MPICH and MPICH-GM, while the latter is used in DMLoader with some advantages such as task's scalability, short startup time and good task management abilities (control task's processes on remote nodes) [7]. In DMLoader, communication service daemon (CSD), only a daemon, runs in every cluster node shown in Figure 4. CSDs communicate with each other to load and manage the processes of DMPI tasks with a peer-to-peer topology and non-blocking connection method. Every CSD is an equal peer, and system overhead is scattered. Thus, DMLoader will not have service bottlenecks, and that load balance and scalability will be achieved.
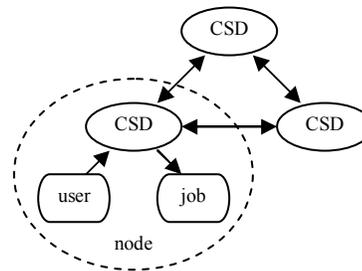

Fig.4. Peers of DMLoader

Figure 5 illustrates the basic performance and scalability characteristics of DMLoader on Dawning4000A supercomputer. It shows that the startup time (the initialization time plus the loading time) of the parallel task varies with the number of nodes that the parallel task demands. As shown in Figure 5, the increasing trend of the average startup time of DMPI tasks is by far lower than that of the number of nodes, that is, the performance degradation of loading large scale tasks is within a tolerant range. Therefore, the design patterns of DMLoader meet scalability requirement of MPI task loader.

5. Conclusions

Software topology and communication mode greatly affect scalability of massive cluster system. According to requirements of real world, it is a key problem how to select suitable design pattern of scalable cluster system software. This paper focuses on scalability issues on massive cluster system software and gives some feasible design patterns. In a sense, successful development of Dawning series of supercomputers has owed to design patterns in scalability of Dawning cluster software.
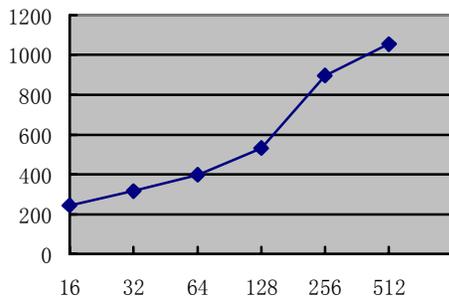
Fig.5. The startup time (milliseconds) of task on different number of nodes

References

[1] TOP500 supercomputer sites. http://www.top500.org.

[2] Srinivas Nimmagadda, Llan Harari, "Scalability Issues in Cluster Computing Operating Systems", Proceedings of the First Workshop on Cluster-Based Computing in conjunction with the 13th ACM-SIGARCH International Conference on Supercomputing, Greece, 20 June 1999.

[3] Ian J. Taylor, "From P2P to Web Services and Grids", ISBN 1852338695, Spring-Verlag London Limited, 2005.

[4] W.Richard Stevens, Bill Fenner and Andrew M. Rudoff, "UNIX Network Programming Vol. 1: The Sockets Networking API, Third Edition", ISBN 0-13-141155-1, Addison-Wesley, 2003.

[5] Jianfeng Zhan and Ninghui Sun, "Fire Phoenix Cluster Operating System Kernel and its Evaluation", IEEE International Conference on Cluster Computing (Cluster 2005), Boston, USA, September 27 - 30, 2005.

[6] OpenPBS Public Home web site, http://www-unix.mcs.anl.gov/openpbs

[7] Bibo Tu, Taoying Liu and Dan Meng, "DMLoader: a Scalable MPI Task Loader on Dawning Supercomputers", The Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, HPCAsia 2004, IEEE press, Tokyo, Japan, 2004, pp. 130-135.

IEEE
COMPUTER
SOCIETY