Scalability Issues in Cluster Computing Operating Systems

Srinivas Nimmagadda
Intel Corporation
M/S SC12-605, 3600 Juliette Lane,
Santa Clara, CA 95052
+1 (408) 765-5137
Srinivas.Nimmagadda@intel.com

Ilan Harari
Intel Corporation
M/S AG1-107, 1525 NW Amberglen Ct,
#100, Beaverton, OR 97006
+1 (503) 696 8460
Ilan.Harari@intel.com

ABSTRACT

Network of Workstations, popularly known as NOW [6], is quickly becoming a cost-effective and scalable alternative for high-end engineering computing. In large computing environments such as Intel chip design EDA environment, global optimization across several clusters is necessary to satisfy the large scale computing needs of several CPU design teams working on multiple generations of chips. Interconnection of clusters involves different aspects such as resource management, global scheduling, and network bandwidth considerations. In particular it becomes complex when different organizational units support and control different parts of the computing resources. We focused on the topology of the system as the key part of the entire system architecture while addressing the scalability issues. Over the past three years we have been using a multi-cluster batch system and our conclusions are based on this experience. This paper presents different approaches and associated challenges in interconnecting tens of thousands of machines, and shares some of our experiences in scaling and global optimization across multiple clusters. We conclude that multi-level clustering is essential and practical for large scale global distributed computing across several co-operative and geographically dispersed user groups.

Keywords

Cluster computing, job scheduling, load balancing, multi-cluster, distributed batch processing, scalable batch systems.

1. INTRODUCTION

A typical cluster environment gives a virtual supercomputer view of its networked workstations facilitating the following functions: i) load balancing among nodes, ii) scheduling: match making of user jobs based on some priority schemes and node availability, iii) providing transparent job execution environment, and iv) cluster and job management capabilities. Generally, these capabilities are provided in a local cluster using master-slave topology. Where slaves provide transparent job execution environment and masters (either centralized or distributed) control the load balancing, scheduling, and job management capabilities. Typical local cluster contains 100-1500 machines. From our experiences with NetBatch (an in-house cluster computing system), we found that scaling of load balancing and scheduling beyond this range presents significant reliability, reduced user responsiveness, and performance degradation challenges. The natural extension to solve this problem is to distribute resources across several clusters and group them using some topology to provide virtual super-cluster (or super supercomputer) view without compromising the four functional areas mentioned earlier. Apart from scalability, multiple clusters may be needed for other reasons such as, geographically separate sites preferring their own cluster but willing to borrow or share idle resources with others. In all these cases, interconnecting clusters to provide virtual user and administrator interface, global load balancing, and scheduling is essential.

The rest of the paper is organized as follows. Section 2 describes different topologies we had considered and experimented to address scaling issues, Section 3 presents scalability issues, Section 4 presents our experiences, and Section 5 gives an overview of relevant work in this area. Section 6 summarizes our conclusions.

2. DIFFERENT INTERCONNECT TOPOLOGIES

Considering the above four key functional scalability aspects for clustering, we have investigated the following four interconnect topologies. They are, i) Virtual Cluster approach, ii) Multilevel Virtual Cluster approach, iii) Virtual Machine-borrow approach, and iv) Hierarchical Global Master approach.

2.1 The Virtual Cluster Approach

In virtual cluster topology we add a new layer above the classic batch cluster (which will be noted a “physical cluster” from now on). As mentioned before, each physical cluster has a master responsible for load balancing within the cluster. The new layer consists of virtual cluster masters. Each virtual cluster master is connected to all or to some of the different physical cluster masters. Virtual cluster masters are not connected to each other (see Figure 1). The end user interacts with one virtual cluster master only. The virtual cluster master accepts new tasks, and later migrates them to one of the physical clusters. The virtual cluster master tracks the state and location of a migrated task throughout its lifecycle.

In order for the virtual cluster master to make the right decision it must receive information form the physical cluster masters about their queues, tasks and resources. It then tries to load-balance the queues of the physical clusters according to the user preferences and the task requirements. Tasks that were not executed within a given period of time can be migrated to another physical cluster in order to prevent starvation. The main role of the virtual cluster master is to allow the user to see entire system as one big cluster. For that, the master needs to balance the queues in all physical clusters in a way that the resource quotas will be honored and maximal throughput will be achieved. The shares are allocated in the physical clusters themselves and if the virtual cluster master is putting the right mixture of tasks in all the physical cluster queues, the shares will be honored.

This approach best fits when a significant number of tasks will not be executed in one cluster.

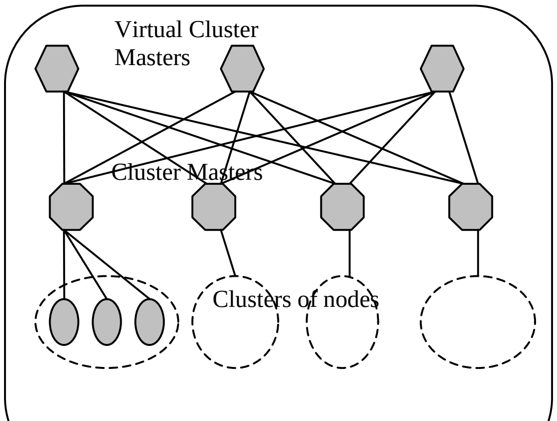

Figure 1 Virtual Cluster Topology

Pros:

- Can be applied on top of an existing batch system. For best results we recommend that the Cluster masters need to be aware of their clients.
- No single point of failure. If any of the clusters or virtual clusters fail the rest of the system will continue to function properly.
- Robust to networks partitioning. In each partition the virtual cluster will use the resources from the clusters reachable by it.
- Scalable to a high degree. In section 4 we describe our experience with it.
- Allows distributed system administration of the system. Each site supports its own clusters and virtual cluster(s). Collaborations is needed only in terms of shares allocation, permission, users and tasks environments.

Cons:

- Complex scheduling algorithms. The virtual cluster balances the queues based on statistics and behavior predictions.
- Temporary imbalance of cluster queues may occur when clusters load change or when users submit tasks in bursts. In the later situation one user may flood the queues and get a bigger share than was allocated. Shortly later, another user may submit jobs and will not be compensated in other clusters for that.

2.2 The Multi-Level Virtual Cluster Approach

This is an extension of the virtual cluster approach. It is to be used when the limits of the former do not allow it to scale any further. The idea is adding more hierarchies to the system. Virtual cluster masters are still the entry points for the tasks. One or more layers may be added between these masters to the physical cluster masters. Any virtual cluster master is connected to some of the servers in the new layer(s). Which in turn is connected to some of the servers in the next layer (see Figure 2). There must be at least one route from every virtual cluster master to every physical cluster. More routes can be used in order to increase robustness of the system. The middle layer(s) servers accept tasks from the virtual clusters, which then try to forward the tasks to the next level so the tasks will wait for the shortest wait time. Most of the internal decision-making algorithms are similar to that of the virtual cluster servers.

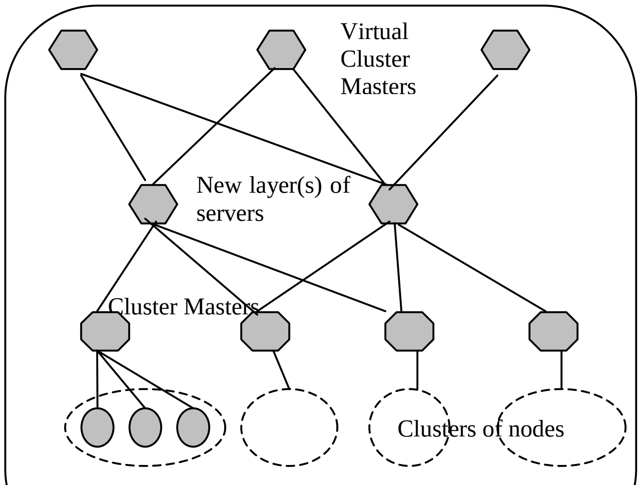

Figure 2 Multi-level Virtual Cluster Toplogy

Pros relative to the virtual clusters approach:

- Highly scalable: More levels can be added as needed.
- Very robust: no single point of failure.
- Very flexible.

Cons:

- Complex decision making algorithms.
- Temporary imbalance may be greater.

2.3 Virtual Machine-Borrow Approach

In this approach, the base level (local) clustering mechanism remains similar to the earlier approaches. Inter-cluster resource sharing happens by virtual transfer of machines between clusters as opposed to transfer of jobs between clusters as described in other approaches. A machine that has been borrowed will start reporting its availability indicators to the new (borrower) cluster master. In this approach, a machine can be borrowed for certain time, or till a return-call from home cluster occurs. This borrow method can be negotiated between the two cluster masters directly. A central daemon process can act as banker between lenders and borrowers, implementing any predefined machine borrow (commerce) rules, and optimizing the global scheduling and network throughput. This approach is suitable in multi-cluster environments where some clusters reach their peak demand while other clusters have relatively low utilization and willing to lend machines. While this approach imposes no minimum or maximum limits on borrow times, this configuration is suitable for borrow times ranging from several hours to several weeks. Virtual machine transfer (borrow-lend) protocol can be implemented as an add-on module on top of the base cluster masters. Administrators can define thresholds for initiating the machine loan protocol when the demand (could be defined as average wait times of jobs in the cluster) reaches certain peak points. Once the machine is added to the borrowing cluster, the entire job scheduling decisions and priority schemes of the cluster will apply for using the new additional resources.

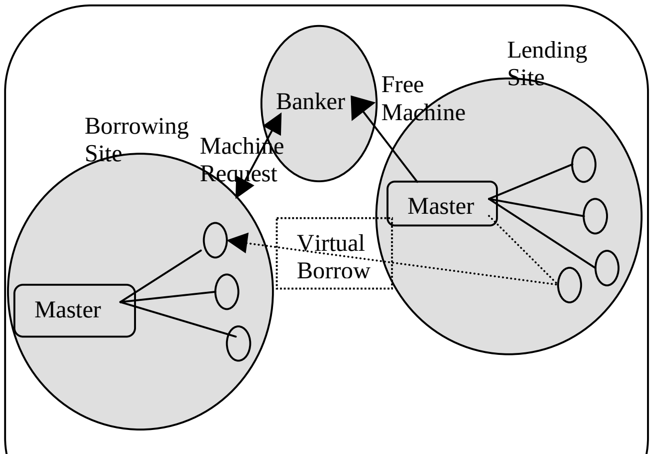

Figure 3. Virtual Machine Borrow Approach

Pros:

- Less overhead in cross-cluster scheduling for multiple jobs
- Useful for coarse-grained resource sharing across clusters. Borrow a chunk of machines, use them for a while as local resources and return back to lender.
- Provide uniform user interface for local and cross-cluster job submission, monitoring, and administration. Provides transparent remote batch environment with local batch environment semantics.

Cons:

- Scalability issues arise when a cluster needs to borrow too many machines to offset the peak demand. A single cluster master may not be able to control too many local and borrowed machines for job scheduling and load balancing.
- Difficult to achieve globally optimized resource utilization. For example, certain clusters may be willing to share machines when they don't have any work, but the lender wants the machines back as soon as they have jobs to execute in their own cluster.

2.4 Hierarchical Global Master Approach

In this approach, machines are pooled together to form base clusters, typically each site/project having one base cluster. Each cluster has its own master and makes autonomous load balancing and job scheduling decisions using predefined policies. The master of the base cluster provides a single uniform interface for accessing local and global resources. Users submit their jobs and monitor the progress using the interface provided by the masters. Administrators define policies for resource allocation in the base cluster for local and remote jobs, as well as the criteria for migrating jobs from base clusters to remote clusters. In each base cluster, servers report their availability indicators to the master. Master acts as a switch for finding the right machines based on each job resource requirements.

A group of base masters can be networked to form super-cluster controlled by a super-master as shown in figure 4. The super-master is responsible for load balancing among the clusters and making inter-cluster job scheduling decisions. Analogous to servers in a base-cluster, each master acts as a server to the super-master. Each master periodically reports the availability of free resources to the super-master. Based on administrator defined thresholds, each base-master selects the jobs to be sent to remote clusters and queues the jobs to the super-master. Using the cross-site resource sharing policies, and other global scheduling data (such as network load, locality), the super-master can find a best remote cluster to dispatch the jobs for execution. Once the scheduling decision is made at the super-cluster, then the job will be migrated directly from the originating cluster to the remote cluster. Then onwards both base masters work together directly to manage the job. The originating cluster provides any further user or administrator interface to the job.

This hierarchy mechanism can be extended to additional levels (such as super super-master) based on the number of machines to be clustered together. The advantage with this method is scalability without having additional software components. It requires configuring the masters to perform actions according to their place in the global cluster hierarchy.

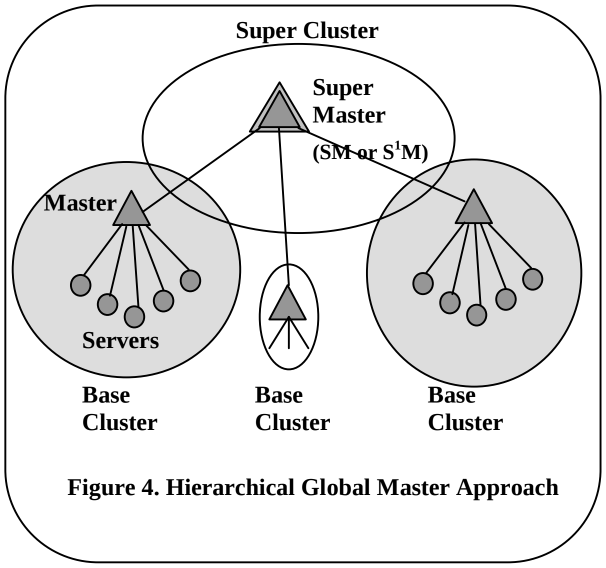

Figure 4. Hierarchical Global Master Approach

Pros:

- Provides capabilities for load balancing at each level based on administrator defined scheduling (priorities at local and global level) parameters.
- Easy administration and control of job execution across clusters. Super-master can act as a central point for decision making based on cross-site policies, while the base-masters have complete control on how many resources they can offer for remote jobs or what and when they can transfer jobs to remote sites.
- Easy software development and scaling based on the need.

Cons:

- Single master at each level could pose fault-tolerant issues. Having backup masters can solve this.
- The super-masters could become bottleneck if there are too many jobs from too many clusters to be executed remotely at the same time.

3. SCALABILITY ISSUES

Load balancing involves finding the least loaded and best suitable machine in the network to execute a job. In a local cluster environment this can easily be achieved by centralized match making algorithms such as the one used in Condor [3]. Centralized global load information pooling and making load-balancing decisions will not be practical and scale well in global cluster environment. Partitioning the global cluster system with some interconnect topologies (such as the ones described in section 2) is the key for building scalable cluster computing systems. Well-coordinated distributed load balancing algorithms and protocols across clusters is necessary to make optimal resource allocation and usage in a global cluster environment. Even with such a scheme, finding the best machine to execute every job in global scale is not desirable. One has to look at the trade off between local and global optimization considering the cost of moving a job to remote sites for execution. Poor cross-cluster execution decisions might result in network congestion, and keeping machines idle for long time while making global placement decisions and migration of jobs.

Job scheduling involves finding most eligible job to run from among jobs belonging to several users and groups. In a local cluster environment this is achieved by fair-share and priority based scheduling schemes. Extending this notion to departments and projects spread across several geographic regions is the key for global job scheduling optimization. Using these global scheduling schemes one must be able to control and enforce project resource priorities with competing projects. This is necessary for “good” throughput in global scale. We refer good throughput as best utilization of resources for most important tasks as defined by the user community. Taking the distributed scheduling state snapshots and construction of virtual global state is required for making such decisions.

The load-balancing algorithms that choose the execution location should also consider network load, availability of user data, and secure execution environment with same user credentials across sites. These problems can be solved with distributed file systems and computing environment utilities such as AFS, DCE/DFS, and Kerberos authentication mechanisms.

Uniform user interface refers to providing capabilities for users to submit and track the progress of local or remote job execution, and administrators to configure any component of the system from anywhere. Providing uniform user interface in a multi-cluster environment requires jobs to be tracked with universally unique identifiers and allowing users to have a single entry point into the system. Cross-domain administration and policy controls may be implemented with multiple hierarchical (administrative) domains. This can be used for tuning various global parameters while having full control on individual local sites.

Allocation of resources can be done locally at each site and by a centralized committee for the global system. When each site allocates its own resources, it may allow users from other sites to share its excess resources but gives first priority to the local users. When doing global resource management the fair share should be kept at the system (usage across all clusters) as a whole. When using a mixture of the two approaches it may be possible to have some of the resources guaranteed for local users while remaining resources shared among all other sites. The system should be able to enforce these resource allocation policies when the global allocation and utilization snapshot is available at all levels.

4. OUR EXPERIENCES

In the past three years, we had used an earlier version of the Virtual Cluster approach (described in section 2) to integrate as many as 8000 machines into one virtual supercomputer. During that time we had identified several cross-cluster computing issues such as providing transparent data access for job execution across clusters, cross-site cluster management, network bandwidth optimized scheduling, and resource allocation across several geographically dispersed user groups. At present we are experimenting with *Hierarchical Global Master* and *Multilevel Virtual Cluster* approaches to address several scheduling and scalability issues uncovered during earlier version of Virtual Cluster implementation.

We had used the underlying file system to provide transparent data sharing and file access needs of the batch jobs in local as well as in remote clusters. Uniform user execution environment is an essential for cross-cluster job execution. Common tools, such as interpreters and shells are preferred and need to be available across all sites. Different participating clusters are scattered across different parts of the world and sometimes connected via a limited bandwidth network. High I/O intensive tasks can easily saturate the network. It is essential to have intelligent load balancing algorithms to take task execution location and computing needs while considering jobs for remote execution. Users are encouraged to mirror their environment and data to all potential sites in order to improve performance and minimize network traffic. Our multi-cluster scheduler daemon at each site monitors the network bandwidth to different sites and considers this information in job routing decisions.

In the current version, cluster tuning is based on configuration files. Each site administrator maintains two types of files, one for local scheduling, and the other for cross-site (global) scheduling. Local site administrator has full control over local configuration files, while global allocation settings are configured across sites by means of group communication. In future releases we expect this to be automated by making our masters more intelligent in propagating the changes to other clusters.

Resource allocation in local clusters is done on a group utilization and demand basis. Resource share is allocated at weekly meetings and enforced by the administrators by tuning the queues appropriately. Users at any site also decide what percent of their resources, if any can be used for cross-site needs. This information is manually synced up in bi-weekly cross-site resource allocation meetings, where all the global resources are pooled and shared among the projects that have computing needs excessive of their local capacity. Participating sites need to allocate a minimum amount of their resources to off-site projects, based on a cross-site committee decision. Cross-cluster tuning is done and policies are enforced using the methods described earlier.

Clustered machines are in most cases different flavors of Unix[1] machines. In our environment we have created cross-site cluster consisting of approximately 8000 machines. Average Utilization of machines in the cluster depends on the specific site policies, configuration, and stage of project in its lifecycle. Utilization typically ranged between 50-90% for compute server farms and 40-80% for interactive workstation farms.

5. RELEVANT WORK

There have been several academic and commercial cluster-computing systems, such Condor[1] [1][2], LSF[1] [4], and CODINE[1] [5], have been developed to address these needs. But most of them have focused on and optimized for local clustering, typically providing load balancing and queuing in clusters of less than 1500 machines. LSF [4] uses virtual machine pooling approach, where free machines in all connected clusters are put together in one pool and job scheduling is done in this cluster. Multi-clustering methods used in LSF are mainly focused on addressing interconnectivity between clusters for load balancing, but fall short of providing a complete virtual view of the global environment with respect to system management, seamless execution, and global resource allocation. Other approaches, such as used by Condor, may fail to scale and perform in a demanding environment due to the complexity associated with their gateway approach for global load balancing and completely distributed nature for job queuing and scheduling decisions

6. CONCLUSIONS

We presented the challenges associated with scaling traditional cluster computing to a multi-site global environment. Global load balancing, resource sharing policies, transparent execution environment, and remote administration are some of them. We have described four interconnect topologies for networked clusters, presented associated scaling issues, and shared our experiences with an earlier implementation of virtual cluster approach. We believe that apart from scaling multi-level clustering is beneficial for co-operative computing between geographically isolated groups.

7. REFERENCES

[1] D.H.J. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A Worldwide Flock of Condors: Load Sharing among Workstation Clusters, Journal on Future Generations of Computer Systems, Volume 12, 1996.

[2] M.J. Litzkow, M. Livny, and M.W. Mutka. Condor – A Hunter of Idle Workstations, In Proc 8th International Conference on Distributed Computing Systems, San Jose, California, June 1988.

[3] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing, *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, July 28-31, 1998, Chicago, IL.

[4] S. Zhou, J. Wang, X. Zheng, and P. Delisle. Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems, Technical Report CSRI-257, University of Toronto, Toronto, Canada.

[5] Computing in Distributed Network Environment (CODINE), Technical Product Description, http://www.genias.de/ welcome.html.

[6] T.E. Anderson, D.E. Culler, D. Patterson, and the Now Team. A case for Network of Workstations: NOW, IEEE Micro, Feb 1995.

[1] Third party trademarks and brands are the property of their respective owners.