Cluster Middleware

An Essential Building Block for Stable and Reliable Compute Clusters

Abstract
Distributed compute clusters gain an ever larger share of the available and installed solutions within the area of high performance computing. Because of their excellent price-performance ratio, these platforms become a particularly attractive option.

To avoid the risk of failure and frustration of users, however, some specific aspects regarding selection and daily usage of the system components must be taken into consideration.

This paper discusses a number of possible problems and the requirements for a stable runtime environment of a compute cluster. Solutions are presented using ParaStation 4 as an example.

Compute Clusters – a Real Alternative
Driven by cost-effective hardware components found in the conventional PC environment, accompanied by previously unmatched computing efficiency and increasing demand for performance, many companies and users – especially in the technical/industrial area – are going to migrate their applications to harness clusters.

Meanwhile, the parallel compute clusters experienced a rapid evolution since the times of the first "Beowulf"-type system: the performance of the processors multiplied, and, in addition to Ethernet, many more capable interconnects are available, e.g. Myrinet or Infiniband. Even the classic Ethernet-based solutions see a revival utilizing cost effective Gigabit Ethernet components. A huge number of software tools simplify installation and administration of clusters. MPI provides a standard programming interface for parallel applications. And, last but not least, this was all made possible by using Linux, a stable and cost-effective operating system for the individual cluster nodes. Linux can be easily adapted to cluster-specific requirements. The large variety of freely available software for Linux, like open source or public domain, made the success of Linux possible. Open standards facilitate the cooperation and further development of particular components. Solutions are available for almost any imaginable problems. However, this brings up some new questions:

• In many cases it is hard to decide which public domain software package will finally match the requirements.
• Are additional packages required?
• If so, how do these packages interact?
• How stable or deficient is this software?

Clusters can be operated in two different modes: typically, the access to the cluster is controlled by a batch system. This implies that the user
does not want to decide and even cannot decide by himself, when and on which nodes his application will run. Presuming "filled up batch queues", efficient utilization of the cluster can be achieved.

Alternatively, access to the cluster can be interactive, i.e. every user logs on to a cluster node, e.g. the front-end node, and launches his application. In this case the users more or less have to coordinate their interaction among themselves. However, this results in very good response times, because there is no overhead associated with waiting time spent in batch queues. This type of usage is suitable, for example, for software development or simulation and immediate visualization. Hybrid usage types are also quite common.

**Compute Cluster Requirements: Compute Power**

One of the key figures of modern compute clusters is the performance that can be achieved. It is determined by a number of different factors:

- The performance of the individual compute node: figures such as pure CPU performance, cache size, or memory bandwidth play a major role. All these parameters can be influenced by selecting the proper hardware components when buying a cluster. Today, typically dual CPU systems are deployed, as these systems are reasonably priced and provide sufficient single CPU performance.

  ParaStation currently runs on all common platforms; with the new version 4, the emerging 64-bit systems from Intel and AMD will be supported as well.

- The performance of the (data) network: one of the major problems of parallel applications is the ability to transfer data very fast and efficiently between the compute nodes. It is common to use a dedicated high-speed network. The two key figures for this network type are the achievable minimal latency\(^1\) and the maximum throughput. Apart from the pure hardware performance, the software protocol overhead has to be taken into account.

  ParaStation 4 comes with a highly optimized communications protocol, especially designed for exchanging data within a cluster. This provides an essential performance advantage for Ethernet-based communication networks. Compared to the standard TCP/IP-based solution, considerably better latencies are even measured using Fast Ethernet, and higher throughput is achieved for Gigabit Ethernet. Performance results for latency can be found in the box "ParaStation 4 Performance" on the next page.

  ParaStation, of course, tolerates occasional transmission errors. Remedy for non-fatal errors is transparent for the applications.

  At the same time, ParaStation 4 significantly lowers the system overhead caused by the network protocol.

- Parallel library implementation: Today, the Message Passing Interface (MPI) represents the library standard for parallel applications. The implementation of the partially complex operations has to be considered very carefully with respect to performance issues. This is also true for the linkage to the underlying network.

  ParaStation, based on MPIch, offers a stable platform for parallel applications. Additional internal optimizations of the standard MPIch version result in much better throughput for large packets.

- Number of nodes: In theory, larger compute clusters provide more performance. Large parallel applications, however, might suffer from scalability issues. Depending on the specific user problem, the implementation within the application, the libraries and the network hardware, these scalability issues may place an upper limit on the number of usable nodes. Beyond this number, no performance gain can be achieved, the performance might even drop. The more efficient a protocol implementation, like in ParaStation, the better the scalability to be expected for the application.

A combination of the above issues determines the overall performance of the compute cluster. Optimum performance can only be achieved by tuning and optimizing all individual cluster components, including the application. Good programming practice includes the use of highly optimized compilers, processor-optimized libraries, tuning of the network layer and protocols, and the parallelism of the code, which enables the distribution of the computation across the optimum number of nodes. Paying attention to the specifics of data transfer, e.g. by using asynchronous communication, may also yield

---

\(^1\) This is the time between the transmission of data on one node and the time this data is received on the other node. Typically, this is measured on application level.
much better results. By using such asynchronous communication calls, data exchange and computation can overlap, thus avoiding idle times of the processor during I/O. The MPI library of ParaStation handles this mechanism very efficiently.

**Compute Cluster Requirements:**

**Availability**

Besides pure compute performance of a cluster, availability is another important aspect. Is the cluster still usable if a problem occurs? If not, how fast can the problem be found and solved? What other steps must be taken by the user or administrator?

A typical scenario is the breakdown of a compute node. This implies that all applications using this node must be terminated. A common problem are processes belonging to the terminated application, which are still running on some nodes.

In worst case, these processes still consume CPU cycles, e.g. while polling new data. Moreover, they could block precious resources such as contexts for user level communication, which are very limited. The system administrator has to check for such processes and manually terminate them. Without doubt, this is a time-consuming and error-prone task.

Based on its built-in process management capabilities, ParaStation detects these problems and eliminates such “left-over” processes. The compute cluster instantaneously offers the full compute power, resources are automatically freed without any intervention by the administrator.

Can the application be restarted? Many common implementations use static lists to determine which process should run on which node. Before starting an application, the user has to know the current status of the compute cluster. If the cluster configuration changes between runs, and the list is not modified manually by the user, the

---

**ParaStation4 Core Performance:**

The new communication protocol of ParaStation4 – specially designed for parallel applications – improves the performance significantly:

- The latency decreases from 27.5 µsec (with TCP) to **11** µsec with ParaStation4.
- The bandwidth increases from 100 MByte/sec (via TCP/IP) to more then **180** MBytes/sec with ParaStation4.
- The time for an MPI barrier call decreases from 31 µsec to **11** µsec.

Both scales show the measured latencies with small packet sizes and the achieved throughput with large packet sizes, each measured with ParaStation4 and MPIch (TCP/IP).

The values were measured on MPI between two dual-processor Xeon systems via Gigabit Ethernet. The bandwidths were measured with parallel send/receive.
program will terminate with an error message, which is usually received after a comparatively long timeout. Then, the user has to investigate which node is not available and remove this node from the list.

ParaStation does this automatically. Instead of using a host list, the user only has to provide the number of processes. The mapping to the nodes available at runtime is executed by ParaStation. Nodes that are currently unavailable are skipped without intervention by the user or administrator.

Another problem that might arise after a node crash, is cluster fragmentation into two logically independent segments. This is due to the fact that some solutions can only use consecutive nodes for the execution of parallel applications. If one of those nodes is not available, only a smaller number of processes can be started. As a result, the cluster is split up into two systems. Even worse, applications using only a moderate number of processes will "use up" the available nodes, leaving unused parts of the cluster inaccessible.

ParaStation does not have these limitations. The distribution of processes is based on different criteria, e.g. the current load. Processes need not be placed on (logically) consecutive nodes. After a node crash, the application only has to be restarted - the processes are automatically placed on the currently available nodes. Potentially available spare nodes are used automatically. They are even used while the system is in perfect condition, not idling, and waiting for a failure.

Many different software environments follow the approach of a central front-end node, which is used for application start-up and administration. Typically, this node also acts as gateway for the cluster nodes. The user has to login to this front-end or gateway node; direct access to the compute nodes is not possible. Of course, this is a critical point of failure. If this particular node crashes, the entire cluster is no longer useable.

ParaStation features an entirely symmetrical view of the cluster. Every node can perform any task, provided the necessary hardware is installed. For example, processes are only placed on nodes connected via a fast network. But, applications can be started on every node. If a dedicated front-end node is required, any cluster node with additional network connection to the external network can be configured with the attribute "do not run compute processes on this node". Then it is possible to logon to this node from external systems, run parallel applications or administrative tasks. The system does not run any computing processes on this node. If redundant configuration is required, simply configure an additional node in the same way or just install an additional network connection from one of the existing nodes to the external network.

**Compute Cluster Requirements: Manageability**

As for any other computer system, the point of administration is an important aspect of compute cluster operation:

- Which effort must be taken to keep the cluster up and running?
- How difficult is it to analyze and isolate problems?
- What kind of support is available for the administrator?

ParaStation offers efficient tools for monitoring and analyzing the current state of the cluster. For example, a single command shows all currently running applications and processes for the entire cluster. The process IDs are unique across the entire cluster. Non-local actions, such as terminating a job or process it on another node, can be initiated on every cluster node. If necessary, the action will be forwarded to the other node, transparently to the administrator. In the same way, a single call shows the current load of the compute cluster.

Besides the command-line based tools, which are preferred by many administrators, a graphical user interface is available, too. Integrating ParaStation-based clusters into overall management environments can be easily achieved using the SNMP interface of ParaStation.

**Integration with Batch Systems**

It is quite common to run ParaStation together with a batch system. On the one hand, this batch system controls the equitable distribution of resources and scheduled job order. On the other hand, a batch system cannot guarantee starting and monitoring of a parallel application. This is efficiently and securely performed by using ParaStation. Additional features are still available, such as I/O redirection, process monitoring, or optimized data transfer using the ParaStation protocol stack.
ParaStation 4: The Efficient Cluster Middleware Solution

As shown in the examples, ParaStation offers a flexible, high-performance and easy to administrate solution for compute clusters. The currently released version 4 features numerous improvements:

• **Support for additional interconnects:** In addition to Myrinet and Ethernet (using TCP/IP), both already available with the previous version, ParaStation 4 now supports Ethernet (including Gigabit Ethernet) through the optimized ParaStation protocol. Soon, Infiniband will also be available. Additional interconnect technologies are on the roadmap.

• **Optimized protocol for Ethernet:** Avoiding TCP/IP saves a lot of time-consuming protocol overhead. Very short latencies and high bandwidth are even achieved by using Fast Ethernet. The bandwidth measured with Gigabit Ethernet is much higher than for TCP/IP.

• **Support for mixed cluster environments:** Using ParaStation 4, it is possible to run cluster systems with different network interconnects such as Myrinet and Gigabit Ethernet as a single cluster. When running a parallel application, the desired interconnect is chosen automatically.

• **Redundant interconnects** For the compute nodes: if the communication via the primary network, e.g. Myrinet, is no longer available, the communication automatically fails over to an additional interconnect, e.g. Gigabit Ethernet (if available). This is fully transparent to the application.

• **Support for arbitrary applications:** The process management of ParaStation 4 enables the start and control of parallel applications, even if they are not linked to the ParaStation libraries. Applications communicating via TCP/IP automatically use the optimized ParaStation protocol and, therefore, benefit from the increased performance without any modification.

• **Enhanced process placement strategies:** Process placement strategies such as the selection of the node with the lowest load or lowest number of processes are already implemented. Additionally, dedicated processors or nodes can be reserved exclusively for a special process or user. Moreover, resources such as memory or disk space can be selected.

• **Start of serial processes:** It is possible to start and run serial processes which are not parallelized with MPI.

• **Improved scalability:** The start mechanism for parallel applications was improved in ParaStation 4. Large applications with numerous processes now are launched even faster and use even less resources. Special provisions are made in the event the program code has to be loaded from a network drive.

• **Standard input/output and signals:** The standard output of all processes is reported back to the calling process, this is the same for standard error messages. By default, the standard input is forwarded to the first process (rank 0). It can be redirected to any other process, but this is user-definable. Signals are forwarded transparently to all processes. This allows, for example, a secure way to suspend and restart an application and prevents these processes from “wasting” CPU time.

• **Simplified user administration:** With the start of an application, ParaStation does not only provide specific application information to the dedicated nodes, but also detailed user information. For clusters using a dedicated front-end or login node it is therefore sufficient to set-up the user only on this node, either local or via systems like NIS+. This makes administration tasks really easy, as the databases on the individual nodes – such as /etc/passwd and /etc/group – need not be synchronized.

In addition, ParaStation avoids concurrent requests from compute nodes to central services such as yellow pages or NIS, and – as a result – prevents network overload or other problems.

• **Extensive configuration options:** The administration interface of ParaStation offers a multitude of configuration features: access to the cluster or part of the cluster can be restricted to users or certain user groups. It is possible to modify protocol parameters during operation. This allows the communication protocol to be adjusted and optimized according to the needs of the application. Nodes can be removed or added to the cluster.

ParaStation supports common debugging and analysis tools such as, for example, TotalView or Vampir.
ParaStation 4 is available for all common Linux distributions. As no modification of the standard kernel functionalities are necessary, the ParaStation protocol can be adapted very quickly and easily to new versions. The user can choose his favorite Linux distribution, as special versions of the kernel or distribution are not necessary.

**Conclusion**

ParaStation is a cluster middleware solution with excellent performance and stability. It is the perfect combination of a high-performance protocol stack with a sophisticated process management. The new version ParaStation 4 supports a variety of interconnects, thus allowing the user to pick the optimum solution.

For more information refer to our website at www.par-tec.com.