Implementing a Full Single System Image UnixWare Cluster: Middleware vs Underware

Bruce Walker
Compaq Computer Corporation,
Los Angeles, CA, USA.
Tel: +1 (1) 310-414-3102

Douglas Steel
SCO Ltd., Watford, U.K.
Tel: +44 (0) 1923-813516
Fax: +44 (0) 1923-813804

ABSTRACT *A Single System Image (SSI) cluster is a collection of computers that appear as a single large computer. SSI can be implemented in two main ways - as Middleware (i.e. the application or subsystem level) or as Underware (i.e. the kernel level). In this paper we pesent the argument that implementing SSI as underware provides several distinct advantages over implementing SSI as middleware.*

We describe UnixWare NonStop® Clusters (NSC), which provides a full SSI clustering environment implemented as underware, whilst maintaining UnixWare functionality and semantics, and providing advanced system scalability and tolerance to node failures. Particular attention is given to the process management subsystem which is one of the main areas that many attempts at SSI leave out.

1. WHAT IS A SINGLE SYSTEM IMAGE ?

Greg Pfister [1] defined a Single System Image as "*the illusion, created by software or hardware, that a collection of computing elements is a single computing resource*". Pfister also defined several "*levels*" of SSI:

- application and subsystem (middleware)
- operating system kernel (underware[1])
- hardware

The SSI concept means that a collection of resources is presented as one, more powerful resource [2]. The SSI concept can be applied to applications, specific subsystems, or an entire server cluster.

[1]The term "underware" refers to the infrastructure hidden below the user/kernel interface [3]

Application or middleware SSI presents multiple, cooperating components of an application to the administrator as a single application. This is often combined with a single management image which lets the administrator work with familiar interfaces and commands to manage the entire cluster as a single entity. SSI also can be applied to subsystems like the file system. This occurs when every node in the cluster has the same view of the data.

A full clusterwide SSI allows all physical resources and kernel resources to be visible and accessible from all nodes within the system. Full SSI can be achieved as underware - in other words by each node's operating system kernel cooperating to present the same view from all kernel interfaces on all nodes.

Such a full SSI has several advantages. If at the kernel level, SSI can save time and money because programs and applications used throughout the cluster do not have to be rewritten to work in a distributed environment. In addition, programs will run on any node without administrative setup, and processes can be migrated to balance the load between the node's in the cluster while they are running.

1.1 SSI AS MIDDLEWARE

Parallel applications (e.g. parallel databases such as Oracle Parallel Server or Informix XPS) are a valuable addition to clustering, and provide a single system view of that application and its data. However, without a full SSI or standard programming interface, application developers must build and rebuild capabilities on an application-by-application basis.

Several attempts have been made to extend the generality of middleware level SSI, such as Beowulf [4] and PVM [5]. These systems provide a set of libraries, daemons and administrative commands to implement a middleware level SSI. However, they largely provide non-native programming models such as message passing or distributed shared memory and thus tend to make the clustered nature of the underlying system visible and are therefore at best provide only a partial SSI. Achieving a full SSI at the middleware level would involve either adopting a new programming interface (and rewriting applications) or by intercepting all system calls at the libc (or similar) level. We believe that implementing an SSI as underware provides a greater level of hiding of the underlying clustered nature of the hardware and also allows the kenrel itself to make use of the SSI (i.e. sending signals to process which may be remote and accessing remote files or devices).

Having the underlying system provide more complete SSI lets application developers avoid the extra effort. It can simplify cluster management and the user and programmer models while also greatly simplifying application development for availability and scalability.

1.2 SSI AS UNDERWARE

There are several important partial SSI cluster features that, when combined, can result in a full SSI offering. The base level functionality consists of several largely independent partial SSI features such as :

- clusterwide system management
- clusterwide device access
- cluster file system
- cluster Internet Protocol (IP) addressing
- clusterwide IPC
- clusterwide process management
- clusterwide load levelling

which can be added on to an existing operating system at the middleware level, underware level or a combination of both.

SSI management can include a whole set of add-on cluster-management functions or more extensions to existing single-system management capability. An SSI device or filesystem capability means a single namespace and SSI syntax and semantics.

Providing the cluster with a single external name (IP address) has proven difficult because external clients cannot be modified. One approach has been to designate one IP address per service and use IP address failover to provide transparent availability. This approach can get quite unmanageable. Other approaches usually have either a special cluster IP address or cluster name with optional capability to distribute the networking load to various cluster members.

Other partial SSI cluster features often require at least a cluster file system to be effective. Batch-load leveling enables the administrator to chose background tasks to run on the least-loaded node. SSI IPC is a single clusterwide namespace and the sharing of standard single-system IPC capabilities like pipes, message queues, semaphores, and shared memory. SSI IPC is included usually only as part of a full SSI cluster. The other two pieces of a full SSI cluster are process management and dynamic load balancing. SSI process management means a single namespace for processes across the cluster and full access to the processes from all nodes. With this feature, creation of processes on other nodes is transparent and semantically identical to local process creation. With full SSI, processes can be rescheduled on other nodes (process migration), similar to rescheduling a process on another processor in an SMP system.

Compared to simple failover or middleware clusters, partial and full SSI cluster offer simplified system management, more effective resource usage, and a simplified programming model. This allow the programmer or administrator to concentrate on availability, sharing, and scalability.

2. UNIXWARE NONSTOP CLUSTERS - SSI AS UNDERWARE

UnixWare® NonStop® Clusters (NSC6] is the result of many years research and development based on the original Locus project [7]. It has undergone revision and iteration by Locus, Platinum, Tandem, Compaq and SCO. Implementations have been developed based on different base UNIX® kernel technologies including AIX, OSF/1 AD, SVR4.x and SVR5. UnixWare NonStop Clusters builds on this base to provide a robust and scalable SSI clustering environment implemented as Underware.

NSC provides a full SSI by making all operating system resources transparently usable cluster wide. This in contrast to middleware level SSI's such as Beowulf, or partial SSI such as GOLDRUSH[8] and NFS[9]. Middleware level SSI's tend to concentrate on one specific area e.g. filesystem, distributed shared memory, system management.

By making all operating system resources, such as processes, files and devices, usable cluster wide in a transparent manner whilst maintaining UnixWare semantics, NSC allows standard UnixWare applications to run unmodified on the cluster. Applications automatically benefit from the system's inherent scalability and resilience. Since NSC's underware level SSI makes use of the cluster transparent, system management is achieved by trivial extensions to the core UnixWare system management facilities. System resilience can be administered by operations such as the ability to specify alternative routes to multiported disks.

To make full use of NSC's facilities, additional API's are provided to allow applications to control placement and migration of processes, and to monitor and react to changes in the cluster's configuration. An application can make use of the inherent parallelism and scalability by using the standard UNIX fork mechanism to create multiple processes and using standard UNIX IPC mechanisms to communicate between the processes, e.g. sockets, pipes, signals, SYSV IPC.

2.1 FULL SSI REQUIREMENTS

In order to provide a full SSI as underware, NSC supports all standard UnixWare API's and system calls. These are location transparent such that access to any resource (device, file, process, semaphores, pipes, etc.) can be made from any node in the cluster.

If a cluster operating system is to provide a full SSI, all operating system resources must be accessible clusterwide without needing to know where the resources are physically located. Such resources include:

- processes
- TCP/IP networking
- STREAMS
- IPC objects (including semaphores, message queues and

shared memory)

- files
- devices

NSC provides location transparent access to all of these resources. For example it is possible for a process running on node 1 to open a device located on node 2, read from it, copy the data to a file hosted by a filesystem on node 3 and to signal process on node 4 to say that it has finished, without any knowledge of where those resources are located.

In NSC many resources, for example processes and filesystems, can migrate between nodes in the cluster to support loadbalancing, process eviction prior to node shutdown, etc. Core system resources are resilient to node failure and can be transparently migrated, rebuilt or restarted on another node.

NSC supports a full SSI through additional kernel modules which are invoked via a small number of "hooks" located at strategic points inside the base UnixWare kernel code (see Figure 1). Hooks exist in areas such as process management, file system framework, virtual memory, STREAMS and TCP/IP stack. Additional kernel modules include distributed process management, cluster membership, file token management, cluster file system and distributed STREAMS.

The underware level SSI is built upon a low-level low-latency kernel-to-kernel communications layer currently based on top of ServerNet, but designed to be portable to other Server Area Networks (SANs) compliant with the Virtual Interface (VI) Architecture [10]). Kernel modules use RPC, request/response (async RPC) and one way message mechanisms to talk to each other in contrast to SolarisMC which uses a CORBA based remote method invocation mechanism [11].

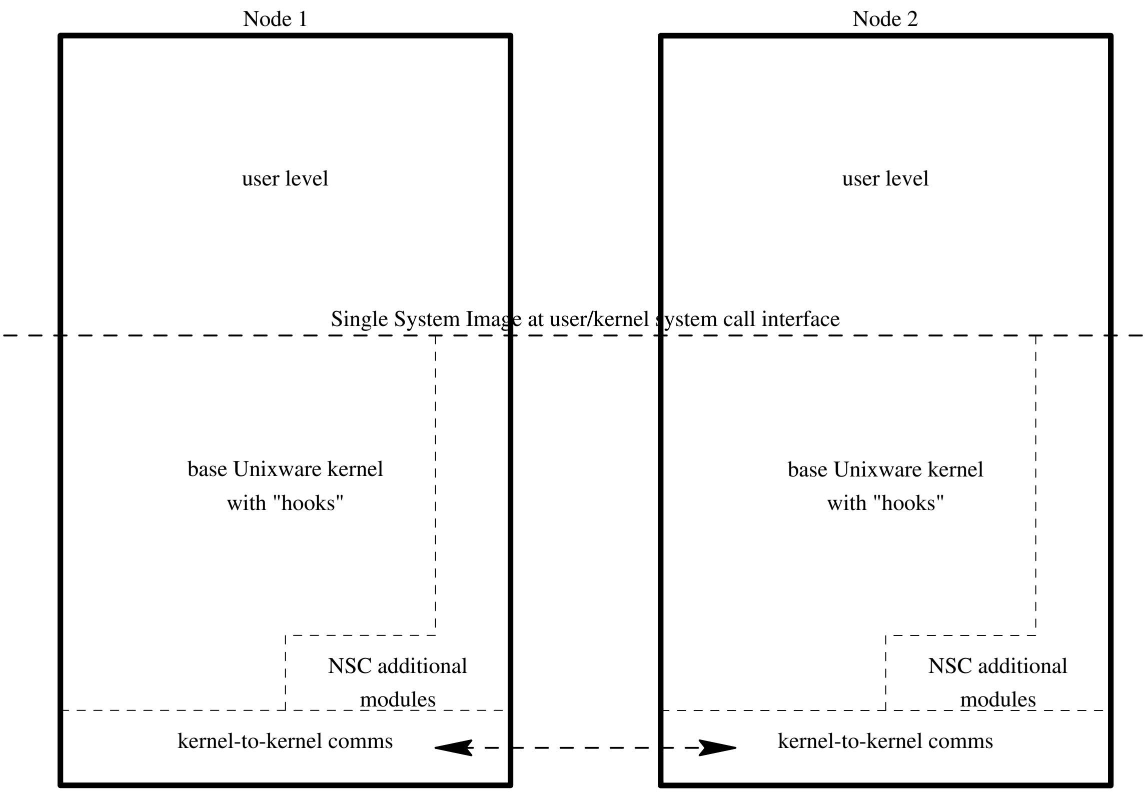

Figure 1 – relationship of kernel components to SSI

3. PROCESS MANAGEMENT

Processes and IPC objects are often not covered by many cluster operating systems and hence they cannot be considered to provide a full SSI. In this section we will describe how the NSC underware implements full location transparency for processes and IPC objects, allowing processes executing on a collection of nodes to cooperate and communicate as they would on a single operating system instance. NSC also allows processes to migrate between nodes whilst they are executing in contrast to some cluster OS's require processes to shutdown and restart from a saved state.

In order to support location transparency for the process management subsystem NSC provides a virtual process (vproc) abstraction above the normal UNIX process [12]. The vproc is analogous to the UNIX filesystem virtual node (vnode) abstraction [13] which sits over a filesystems inode and provides filesystem type transparency. The vproc abstraction hides from the system call layer of the process management subsystem the physical implementation of the process which may be either on the local node or on a remote node and the distributed implementation of process relationships. This is invisible to the user and all standard UNIX process management interfaces can be used cluster wide e.g. ps, kill, /proc. There is no restriction on placement of processes, for example a child process may exist on a different node from its parent and they may form part of a process group which spans the whole cluster.

NSC's implementation of process management involves splitting the base UnixWare process management kernel software into an upper half which handles system calls (and calls from other parts of the kernel) by performing operations on vprocs, and a lower half which performs the physical implementation of a local process (access to user and proc structures). In between these section of base kernel software is inserted the vproc layer which is responsible for the distributed process management and all process relationships by invoking either the local low level process code or using RPC's to talk to a remote nodes vproc code.
Migrating a process is in some ways like job control. Processes and process groups can be migrated by sending them a new SIGMIGRATE signal which an unmodified process doesn't even see. In kernel mode, the process migrates itself by suspending, copying its state to the target node (argument to the SIGMIGRATE signal), making iteslf runnable on the target node and destroying the image on the originating node. A sucessful migration results in the process resuming execution at the point the signal was processed. Standard signal permissions are needed to migrate a process and the process can ignore or catch the signal. If the migration failed (e.g. the target node crashed) the process can be resumed on the node where it was running when the migration request was received.

4. CONCLUSIONS

In addition to ensuring high availability, the ideal cluster environment provides scalability and systemwide manageability (which translates into lower cost of ownership). Clusters also provide an easy migration path for scaling to more powerful hardware and software, without disrupting service for users. This is possible only if the cluster has been designed with both scalability and continuous availability in mind. Such an approach maximizes the hardware capabilities with the implementation of a highly available operating environment and clusterwide load balancing, creating a system that handles both planned and unplanned outages. Load balancing allows the distribution of work across the nodes of a system in keeping all nodes equally busy, while meeting the work-performance goals of the user.

Ideally, the cluster provides a single administration and security domain. SSI provides a simple, straightforward view of all system resources and activities, from any node of the cluster.

UnixWare NonStop Clusters implements a full Single System Image as underware. This enables standard applications to run unaware that they are running on a cluster, and while they are running they may be transparently migrated. Operating services may become temporarily unavailable due to hardware failure and then recovered, without losing the whole cluster and with minimal disruption to services. Additional utilities and APIs allow applications and administrators to make full use of the salability and availability provided by NSC.

Future enhancements include support for cross node disk mirroring, more APIs including user level VI architecture support, support for additional cluster interconnects using kernel level VI architecture, migration of more kernel objects such as socketsand IPC objects, and online system upgrade.

NSC is available now on UnixWare 2.1.2 in configurations of up to six nodes with a scalability of 97% [14], it has also been demonstrated on a configuration of 30 nodes (120 processors) with over 80% scalability. NSC is currently being ported to UnixWare 7.1 and this version is expected to be released in Summer 1999.

5. REFERENCES

[1] Greg Pfister, "In Search of Clusters", 2nd Edition, Prentice Hall 1998

[2] B. Walker, "Clusters: Moving Beyond Failover", Performance Computing, August 1998

[3] J. Boettcher, J. Ogden, K. Klingensten,"Middleware - Authentication, Authorization, and Directory Services", June 1998, http://www.cren.net/know/exptrans/middleware_1.html

[4] D. Becker, T. Sterling, D. Saverese, J. Droband, U. Ranawak, C. Packer, "Beowulf: A Parallel Workstation for Scientific Computing", Proceedings International Conference on Parallel Processing 1995

[5] V. Sunderam, "PVM: A Framework for Parallel Distributed Computing", Concurrency: Practice and Experience, December 1990, pp315-339

[6] D. Steel, "Inside UnixWare NonStop Clusters", to appear in "The UNIX System Today", published by The Open Group 1999.

[7] G. Popek and B. Walker, "The LOCUS Distributed System Architecture", MIT Press, 1985

[8] S. Kittur, D. Steel, F. Armand, J. Lipkis,"Fault Tolerance in a Distributed CHORUS/MiX System", USENIX 1996 Annual Technical Conference, pp219-228

[9] R. Sandberg, D. Goldberg, D. Walsh, B. Lyon, "The Design of the Sun Network File System", Proceedings of the Summer USENIX 1985, pp119-131

[11] J. Kahlidi, J. Bernabeu, V. Matena, K. Sherriff, M. Thadani, "Solaris MC: A Multi Computer OS", USENIX 1996 Annual Technical Conference, pp191-203

[10] VI Architecture group, "Virtual Interface Architecture Specification Version 1.0", December 1997, available from http://www.viarch.org (registration required)

[12] B. Walker, R. Zajcew, G. Thiel, "VPROCS: A Virtual Process Interface for POSIX Systems", Technical Report LA-0920, Locus

Computing Corp., 1992
[13] S. Kleiman, "Vnodes: An Architecture for Multiple File System Types in Sun UNIX", USENIX Summer 1986 Conference, pp238-247
[14] Standard Performance Evaluation Corporation, "SPECint_rate95 results", available from http://www.spec.org/osg/cpu95/results/res98q4