

**Scalable Cache
Coherent Shared
Memory at
Cluster Prices**



White Paper

SMP Redux: You *Can* Have It All

Large scale plug-and-play SMP delivers mainframe performance at commodity prices

By: Douglas Eadline

About the Author: Douglas Eadline, Ph.D. is the Senior HPC Editor for Linux Magazine and has been writing and working with high performance computers and Linux for over twenty years. He is also the Editor of Cluster-Monkey.net.

NW3V1

ABSTRACT

When High Performance Computing (HPC) is mentioned, one often envisions a large expensive mainframe built by a company like Cray, IBM, SGI, or Sun (now Oracle). These machines are highly parallel and employ a Symmetric Multi-Processing (SMP) design that provides global memory and process spaces. Delivering this level of performance has always been expensive due to the custom engineering required to integrate a large number of processors into a shared memory environment. Indeed, the added expense of large-scale SMP systems has pushed the market to use a “cluster approach,” where a large number of commodity server machines connected together are used as a single resource. It is well known that clusters are harder to manage and less efficient than SMP mainframe systems.

The cost of large-scale SMP systems has been an impediment to their widespread adoption, when compared to the cost of commodity server clusters. In this paper we will contrast current SMP and cluster designs in the context of HPC. We will also introduce a breakthrough technology from Numascale that allows commodity hardware to be combined into a cost-effective and scalable SMP system that delivers the best of both worlds – the low cost of clusters and the efficiency of SMP.

What is SMP And Why Should I Care?

Anyone who uses a multi-core processor is almost certainly taking advantage of SMP, which is defined as two or more identical processors connected to a single shared main memory bank and managed by a single operating system (OS) instance. Standard software will run as is on an SMP system, even as more processors are added. The software will only use one processor, however.

For an end user, an SMP system allows for more work to be done at the same time. With SMP, the OS can improve performance by assigning programs to various processors (or cores within a processor) to improve overall throughput. The OS normally has “the right” to relocate programs in order to keep the cores evenly loaded. Thus, SMP provides a simple and effective way to increase the throughput of simultaneous programs running on a computer.

Most programs are designed to use a single processor or core. SMP allows programs to use multiple cores at the same time. These programs are often called “parallel” and can increase performance by employing more than one processing core. This mode of operation requires software to be modified from its original single processor/core design.

One key advantage of any SMP system is the ability of all processing cores to see all memory in the system. This feature allows for easier “parallel programming” (i.e., creating parallel SMP programs) and management of the machine. These advantages have prompted companies like IBM, Sun, HP, and Silicon Graphics to produce SMP mainframe systems with hundreds of processing cores. While these systems offer many advantages to both users and administrators, they can cost up to 30 times more than an equivalent cluster of commodity servers.

SMP systems offer better resource utilization than clusters because there are no hard boundaries between processors, memory,

and the file system. Process loads on an SMP can be managed at much finer grain than on a cluster. In addition, there is also no need for a cluster file system on an SMP machine. Since there is only one OS image running, there is only one view of the file system for all processes. Virtually all clusters employ a distributed file system such as NFS (Network File System). Large clusters often require a more sophisticated parallel file system, which means substantially more administration. Less administration translates directly into a lower cost of ownership for SMP systems.

The need for many parallel processing cores in the HPC market sector should make SMP mainframes a natural choice for many users. But interestingly, the low cost of commodity servers allows users to “cluster” a large quantity of smaller servers in order to achieve higher overall core counts at a lower cost than purchasing a large SMP mainframe. The servers are connected by high-speed interconnects such as 10 Gigabit Ethernet or InfiniBand. Users and administrators give up many of the advantages of a true SMP computer in exchange for more processing cores, which exist as a collection of discrete servers (or nodes) in a large “networked” system. The inability of the OS to see all processors and memory creates a system that is often harder to manage and program. Thus, most users would prefer an SMP environment, but in the end are content to suffer cluster headaches in exchange for a larger number of processing cores at an acceptable cost.

Another advantage SMP systems have over clusters is the ability to operate on a single large data set. These types of problems are of two basic types. The first is the predictive simulation (i.e., quantum chemistry, CFD, etc.) where they can use as much RAM as possible in order to allow larger problems to be solved. The typical cluster node can only provide up to 64 GBytes of RAM, and thus forces large problems to be broken up between nodes, which can drastically reduce efficiency. Another type of application is a complex database query searching for a

piece of data in a large in-core database that will help steer the next simulation. Many of these problem sets are not possible (or are very cumbersome) on a traditional cluster.

A final advantage of SMP systems is their ease of management. A single OS image provides global memory and program views that make administration simple and consistent. It also allows for easier job scheduling synchronization and storage management, and results in less cost. The traditional cluster often has many hidden costs that are not present in SMP systems (see: The True Cost of HPC Cluster Ownership,

<http://www.clustermonkey.net/content/view/262/33>).

At a more technical level, an SMP system with a single synchronized OS does not suffer from an effect known as “OS-jitter,” which is due to multiple non-synchronous real-time OS clocks running on each node. In a large cluster, this effect can reduce performance significantly by slowing down collective operations or other synchronizing functions by as much as 400% (see: Operating System Jitter, http://domino.research.ibm.com/comm/research_projects.nsf/pages/osjitter.index.html).

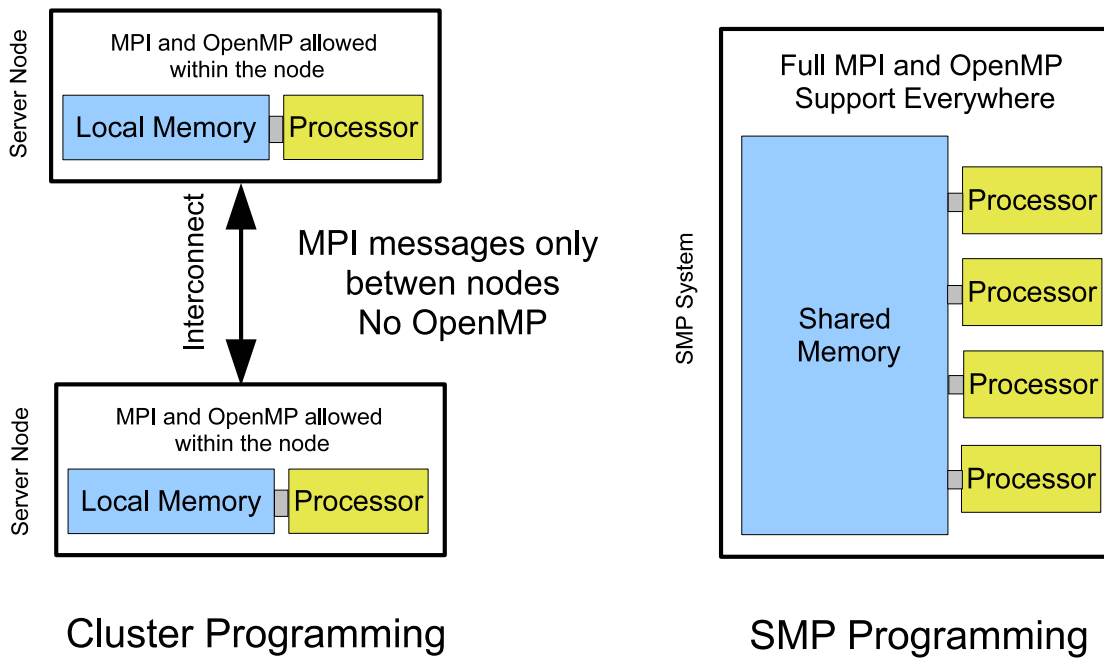


Figure One: An idealized design illustrating the differences between a cluster and an SMP machine. Each cluster node has a local processor and memory with a separate OS instance. Data must be transported between nodes over the interconnect. The SMP system has a single OS and shared memory (there is no need to copy data).

Sending Messages or Creating Threads

Perhaps the biggest difference between SMPs and clusters is the programming methodology. Ignoring for a moment the additional administrative overhead required by clusters, creating programs is often the biggest hurdle for anyone wanting to create parallel HPC applications.

Parallel programming for an SMP system is based on the “thread” model. In the thread model, the main program creates many sub-programs that share the same memory view

as the main program. That is, each sub-program can “see” the all the memory allotted to the program from the OS. Shared memory is easier to program and easier for compilers to exploit via the loop-level parallelism found in many HPC applications. Each “thread” can therefore read and write the same memory as every other thread. Memory access can be protected and threads can communicate through various mechanisms. The most common thread model is based on the IEEE Pthreads standard (Pthreads is short for POSIX threads, Portable Operat-

ing System Interface for Unix).

While Pthreads has provided a successful and standard interface, many users find the model cumbersome to use. In an effort to make thread programming easier, the OpenMP standard was introduced. The OpenMP model is not a programming language, but a specification for supplying compiler directives as part of the source code. An OpenMP-compliant compiler will then utilize the user-placed directives to produce a parallel “threaded binary” of the source code. The pay-off for threaded binaries is their ability to run in parallel on an SMP system. In most cases, the number of threads (parallel parts) is tunable and determined at run-time based on the number of available cores. Users like running codes in this type of environment because the parallelism is flexible and largely transparent.

In a cluster environment, migrating threads across nodes is very difficult to do efficiently. For this reason, the MPI (Message Passing Interface) is used to send memory data between servers. MPI is implemented as a library and is used to provide messaging functionality to existing programming languages (C, C++, Fortran). At its core, MPI is a memory copying protocol, because message data must exist on two or more machines. While this may seem ineffective, it does allow for scaling programs across large numbers of nodes. Due to this fundamental difference (thread-shared vs. memory-copied), MPI and OpenMP programs are designed using different approaches or paradigms. MPI programs are often a bit more difficult to run than SMP programs because they must start programs on multiple servers at the same time.

Users are often faced with the OpenMP vs. MPI decision. On the one hand OpenMP is easier to program but does not scale, while MPI is harder to program but can scale beyond a single server node. This decision is becoming much more important because the typical commodity SMP node can have

as many as 48 processing cores.

Connecting Servers

Digging a little deeper into how clusters operate, a typical cluster is essentially a collection of individual islands connected by boats that must traverse the water between the systems. Cargo (or data) must be moved from one island to another before it can be used. As mentioned, in a cluster data is actually copied (vs. moved) from one location to the next. This feature is fundamental to all types of cluster interconnects.

For the fastest clusters, the interconnect of choice is InfiniBand (IB). As a high-end networking standard, IB provides high throughput and low latency communication between servers. It also adds to the cost of each server node in terms of an extra interface card and switch hardware. These costs can often account for as much as one third of a cluster hardware budget.

From the administrative side, large server farms represent a difficult management problem. In many cases, each node has its own hard disk and OS image. Provisioning and managing these images is not a trivial task and usually requires additional software overhead to do properly (i.e., upgrading can be especially difficult). In addition, a cluster-wide batch scheduler must be employed to allow for shared access of these systems. Each node must “report in” to a central server so that resources can be allocated in consistent fashion. Since each node represents a separate “process environment” (i.e., each node only “knows” about its own running programs), tracking and managing multiple programs across a cluster can, at times, be a difficult task. These tools are available both commercially and as open source tools. In either case, they incur an additional purchase price or time investment beyond the cost of the hardware.

The Commodity SMP Proposition

In one sense, clusters and SMP are an either/or orthogonal approach to comput-

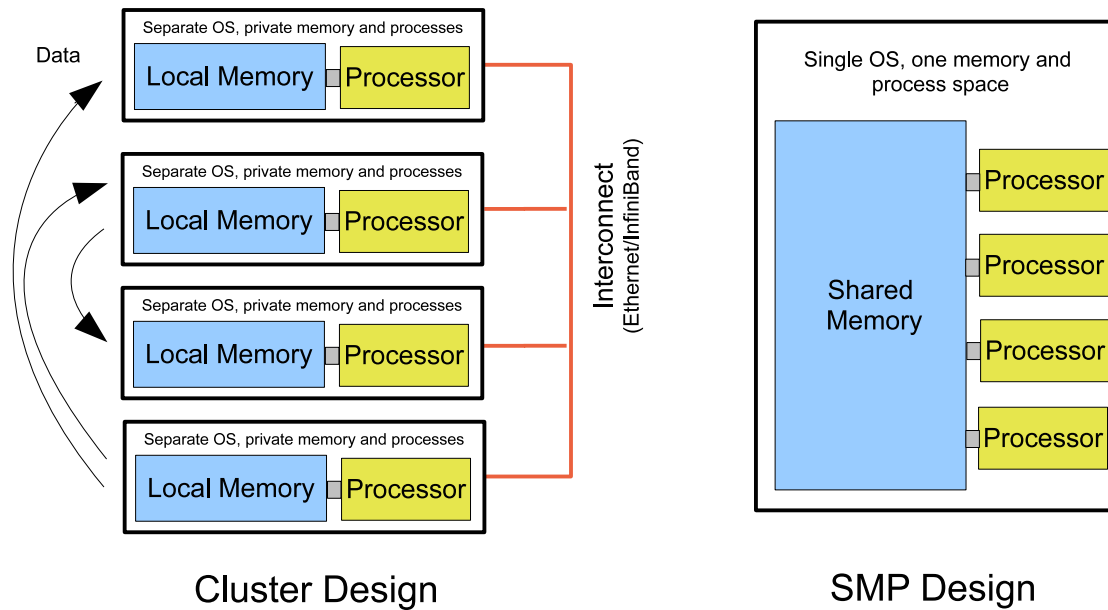


Figure Two: The Cluster Programming Dilemma. Clusters allow MPI programs to run efficiently between server nodes, but there is no easy method for OpenMP programs to operate across a cluster. In an SMP system both MPI and OpenMP can run anywhere on the machine.

ing. Indeed, the programming models are so different that moving between architectures may often require large re-writes of existing code.

Traditionally, threaded (OpenMP) applications cannot be scaled beyond a single server node that exists within a cluster. Thus, users often chose MPI programming as a means to scale beyond a single server. On the other hand, while MPI codes can run in an SMP environment, they may not be optimal for a shared memory environment. Both methodologies represent solutions that are directed by the underlying hardware.

Until recently, the only hardware solution that supports both programming environments for large numbers of processing cores is the expensive SMP mainframes. These systems are often seen as prohibitively expensive for most HPC users and do not offer the flexibility of commodity solutions.

The desire for SMP features has been the driving force for many cluster software projects as well. Indeed, CLOMP, Mosix, bproc, Kerrighed, and ScaleMP all attempt

to bring many of the SMP features (i.e., unified memory and process view/migration) to the commodity cluster. These projects have seen some success, but still operate on top of the island-to-island transport mechanism mentioned above.

The optimal solution would be to connect commodity servers to form a true SMP machine. Such a system would address many of the issues mentioned above. Users and administrators could expect the following:

- Mainframe SMP performance at commodity prices
- Support for both OpenMP (threads) and MPI (messages) in the same environment
- Single memory and process space
- Large amounts of memory
- Single OS image and simple management

In essence, virtually all of the management headaches and programming issues would simply not exist in a scalable SMP environment.

Introducing Plug-and-Play SMP

Numascale's NumaConnect™ Technology is literally a plug-and-play solution that provides a scalable SMP environment out of commodity components based on the industry standard HyperTransport. The technology combines all the processors, memory, and IO resources of the system in a fully virtualized environment controlled by standard operating systems. NumaConnect is not an emulation technology that makes multiple servers look like an SMP, nor does it use Direct Memory Access (DMA) techniques to move memory around.

Due to the limitations of shared memory technology, all large-scale SMP systems employ what is known as Non-Uniform Memory Access (NUMA) to achieve a global memory view. Memory, while globally accessible, is not always local. It may exist in another bank of memory on the same motherboard, or on a motherboard somewhere else in the system. Access times will vary depending upon location.

Another feature of global memory is cache coherence. Briefly, data may exist in either main memory or in processor cache memory. Changes that occur in cache memory are not reflected in the main memory until the cache memory is synchronized with the main memory. To make efficient use of the processor caches, NUMA systems need to contain automatic cache coherence mechanisms. In order to fully support global memory access with full cache coherence, NumaConnect provides cache coherent NUMA (ccNUMA) capabilities to all connected systems.

The NumaConnect technology uses a single chip that combines the ccNUMA control logic with an on-chip seven-way switch; the on-board switch eliminates the need for a separate central switch and long cables. The details of the NumaConnect directory-based ccNUMA technology are available in previous papers. The NumaConnect SMP adapter has the following features:

- Attaches to coherent HyperTransport (HT) through a standard HTX connector found on commodity motherboards
- Provides up to 4 Gbytes of Remote Cache for each node
- Full 48 bit physical address space, providing a total of 256 Tbytes of memory
- Support for up to 4,096 nodes
- Sub-microsecond MPI latency between nodes (ping-pong/2)
- An on-chip distributed switch fabric

The NumaConnect™ SMP Adapter literally transforms AMD multi-core servers into a fully functioning SMP system. The solution does not require software to be modified or rely on any OS extensions.

A Better Connection Value

All high-end clusters include a leading edge message passing interconnect. Now that scalable commodity SMP is a reality, why not ask the question:

If you could buy both an SMP and a cluster for the same price, would you?

Programmers can utilize the best of both worlds, and there is no penalty or limitation imposed by either methodology. The user is not constrained by the hardware and can now choose the best solution for their problem.

In addition, from an IT manager's standpoint:

How much money are you spending managing a cluster that could otherwise be saved by using an SMP?

Because the NumaConnect SMP adapter includes integrated switching, there are no additional switches or rack space required beyond the adapter and cables. The scaling cost is constant and there is automatic redundant routing built into the fabric.

Linux, Solaris, and Windows Server will run unmodified over the NumaConnect fabric. A bootstrap loader performs initialization and configuration so that the system appears to the OS as a large unified shared memory system.

Conclusion

Numascale has made the return of SMP to the HPC mainstream possible. The cost barrier to large-scale SMP deployments has been broken by an interconnect that provides true shared memory (ccNUMA) across commodity servers. Based on the industry standard AMD HyperTransport, the Numascale NumaConnect™ adapter provides the cost savings and convenience of a mainframe SMP system at commodity prices.

The advantages of SMP systems are well known and do not preclude running existing cluster applications (i.e., MPI) on an SMP machine. Indeed, the NumaConnect technology represents the best of both worlds (mainframe SMP and low cost clusters) and literally provides a “plug-and-play” SMP environment that does not require emulation software or specialized drivers. A single operating system image provides a unified memory, process, and I/O space in which both threaded (OpenMP) and message passing (MPI) applications can execute, eliminating the “either/or” scenario found on today’s clusters.

For the first time in the history of commodity HPC, a high performance interconnect can actually reduce administration costs while at the same time improve performance and utilization above that of a traditional cluster. Scalable commodity-based SMP systems can provide measurable reductions in both administration costs and time-to-solution for your HPC needs.