Distributed Symmetric Multi-Processing (DSMP)

A New Paradigm for High Performance Computing

Symmetric Computing

Venture Development Center
University of Massachusetts
100 Morrissey Boulevard
Boston, MA 02125
USA

This page is intentionally blank.

Introduction

Today, the de-facto standard for high performance computing (HPC) is distributed memory clusters connected using the Message Passing Interface (MPI) protocol. However, to achieve the performance that HPC clusters promise, applications must be tailored for the architecture. Over time, a library of cluster-ready HPC applications has been developed. However, there are many important end-user applications left unaddressed. Problems with big data-sets are awkward at best for the MPI model. There is no shared memory for storing large data structures. Even after a successful port, many programs suffer poor performance due to MPI hierarchy and message latency, and/or reliance on file systems as a working global memory. Entire fields of scientific endeavor that could benefit from high performance computing have not, due primarily to the programming complexity of HPC clusters.

Most scientists, researchers, engineers and analysts like to focus on their specialty, get their computations executed quickly, and avoid becoming entangled in the programming complexities that supercomputing clusters demand. They typically develop their work on Symmetric Multi-Processing (SMP) workstations -- SMP computers aligning more closely with their computer skills, and only need supercomputers for their more demanding applications and data sets. Scientists, researchers, engineers and analysts would rather not re-write their applications for a supercomputing cluster and would prefer to use SMP supercomputers. However, SMP supercomputers have been out of reach economically for many. They have been too expensive due to their reliance on costly proprietary hardware and proprietary interconnects, and they have had limited scalability.

What is needed is the ability to make computing clusters function like large SMP machines. There are two approaches that have attempted to achieve this goal. The first effort tried to mimic the architecture of mainframe SMP machines by building custom boards that could be added to each node in a cluster that would effectively enforce cache line coherence and shared memory across all nodes. The second effort capitalized on the techniques of virtualization, and built a hypervisor that ran on each node creating a virtual SMP machine. The performance of both of these approaches was somewhat disappointing.

Symmetric Computing's patented distributed symmetric multi-processing (DSMP) takes a different approach. By recognizing the limitations of the mainframe cache line coherency model, and implementing our algorithms as extensions to the Linux kernel, we are able to deliver the performance of mainframe supercomputers at the cost of computing clusters. Our model maintains the programming simplicity of SMP. The DSMP technology can potentially bring the benefits of supercomputing to heretofore unreached fields of research, development and analysis. Furthermore, this technology has the potential to replace traditional mainframes in many HPC and enterprise applications.

Limitations of MPI Supercomputing Clusters

Computations assigned to a MPI cluster must be carefully structured to accommodate the limitations of the individual server nodes that make up the cluster. In many cases, highly skilled programmers are needed to modify code to accommodate the clusters hierarchy, per-node memory limitation and messaging scheme. Once the data-sets and program are ready to run, they must first be propagated onto each and every node within the cluster, usually by means of a cluster file system such as Lustre. Only then can actual work begin.

Besides complexity, there are entire classes of applications and data sets that are inappropriate for MPI clusters. Many high performance computing applications (e.g., genomic sequencing, coupled engineering models) invoke large data sets and require large shared memory (≥512-GB RAM). Addressing these problems is awkward using the MPI-1 model, for it has no shared memory concept, and MPI-2 has only a limited distributed shared memory concept with a significant latency penalty. Hence a significant restructuring of the application and associated data sets is required in order to use MPI.

Data-sets in many fields (e.g., bioinformatics & life sciences, Computer-Aided Engineering (CAE), energy, earth sciences, financial analyses) are becoming too large and too computationally intensive for single commodity SMP servers. In many cases, it is impractical and inefficient to rewrite the application to use an MPI cluster. The alternatives are to:

- Restructure the problem (to fit within the memory limitations of the nodes and suffer inefficiencies)
- Wait for and purchase time on a University or National Labs SMP supercomputer.

Each of these options has their own drawbacks, ranging from latency & performance to lengthy queues for time on a government (e.g., NSF, DoE) supercomputer. What HPC users really want is unencumbered access to an affordable, large shared-memory SMP supercomputer.

Distributed Symmetric Multi-Processing (DSMP)

Symmetric Computing DSMP architecture provides affordable SMP supercomputing. It enables Distributed Shared Memory (DSM), or Distributed Global Address Space (DGAS), across an Infiniband connected cluster of homogeneous Symmetric Multiprocessing (SMP) nodes. The cluster is converted into a DSMP supercomputer that can service very large data-sets or accommodate MPI applications with increased efficiency and throughput running on shared memory. DSMP provides an alternative to the MPI protocol for a wide range of memory intensive applications because of its ability to service economically a wider class of problems with greater efficiency.

DSMP creates a large, shared-memory software architecture at the operating system level. It supports distributed multi-threading and synchronization across all processor cores on all of the nodes using the standard POSIX thread model (Pthreads). From the programmer's perspective, there is a single software image and one Linux operating system for a DSMP cluster. Since DSMP runs on clusters built with industry standard severs, it delivers large shared memory, many-core SMP mainframe computing with both economy and performance.

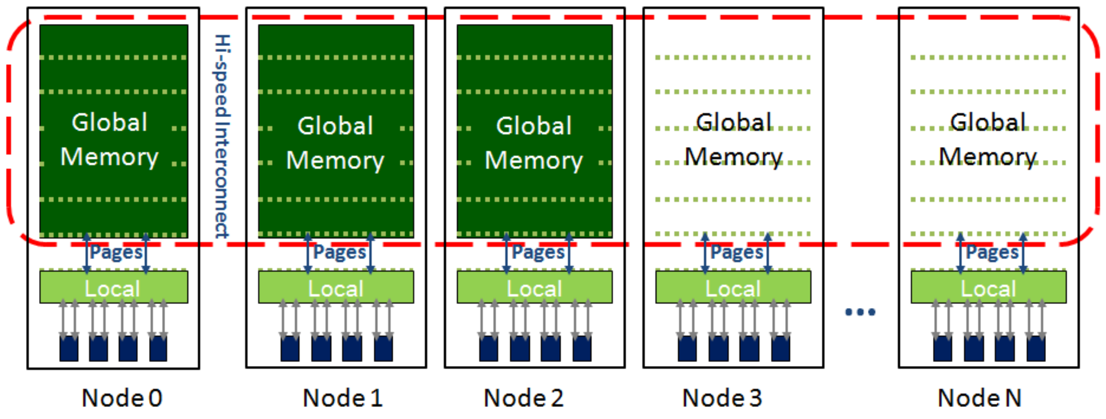
The key features of the DSMP architecture are:

1. A transactional distributed shared-memory architecture
2. A kernel based RDMA inter-node communication driver using Infiniband
3. An application driven, memory-page coherency scheme
4. A new kernel based distributed POSIX threads implementation supporting localization
5. Support for process execution and System V IPC across all nodes
6. Distributed buffered file I/O
7. Single system image via head node

How DSMP Works

DSMP is implemented as a Linux kernel enhancement. The DSMP Linux kernel is installed on every node in the Infiniband connected cluster. These kernels coordinate their activities, effectively operating as a single cluster operating system. One node is designated as the head node and the others as worker nodes. An application program begins execution on the head node, but has the ability to allocate global memory from a combined pool taken from multiple nodes. It also has the ability to launch execution threads running on multiple worker nodes, but referencing the same global address space. Currently, DSMP will support up to 16 nodes that host global memory. Additional compute only worker nodes that can access but not host global memory are supported. The generalized DSMP architecture is shown as follows. Nodes that have a green shaded global memory region are global memory hosts, whereas nodes with white shaded global memory regions are compute only and access global memory.

Single SMP Server comprised of tightly coupled cluster nodes



Node 0 is the head node providing a single system image. Nodes 1-N are worker nodes. Worker nodes can be compute only, with no hosted global memory.

Interconnect supports RDMA. Can be Infiniband, PCI-E, or other.

Global Shared Memory
🟩 Local Memory
■ Multi-Core Processor

Transactional Distributed Shared-Memory System[1]

The centerpiece of the DSMP architecture is its transactional distributed shared-memory architecture, which is based on a two-tier memory organization. DSMP divides physical memory into two partitions: local working memory and global shared memory. The global partitions on each node are combined to form a single global shared memory that is linearly addressable by all nodes in a consistent manner. The global memory maintains a reference copy of each 4096-byte memory page used by a program running on the system at a fixed address. The local memory contains a subset of the total memory pages used by the running program. Memory pages are copied via hardware based demand-paging from global memory to local memory when needed by the executing program. Any changes made to local memory pages are written back to the global memory. At the same time, a page invalidation message is sent to all nodes in order to force any node that has a copy of the page, to update that copy. In the event that a node receives a page invalidation request for a page that has been locally modified, a page invalidation fault is generated.

DSMP sets the size of local memory at boot time, typically 64 GB. When there is a page-fault in local memory, the DSMP kernel finds an appropriate not recently used (NRU) 4096-byte memory page in local memory and swaps in the missing global memory page. The large local memory (cache) provides all the performance benefits (STREAMS,

[1] The use of the word "*transactional*" here is drawn from the standard programming model used in database development, where the programmer is responsible for managing record locks in order to ensure data coherency.

RandomAccess and Linpack) of local memory in a legacy SMP server, with the ability to service a page fault from the large globally shared memory in less than 5µ-seconds. Not only is this architecture unique and extremely powerful, it can scale to hundreds of nodes with no appreciable loss in performance.

Kernel based RDMA Infiniband Driver: DSMP is made possible with the advent of a low-latency, commercial-off-the-shelf network fabric. Today Infiniband is the fabric of choice for most supercomputing clusters due to its low latency and high bandwidth. In order to squeeze every last nanosecond of performance out of the fabric, DSMP bypasses the Linux Infiniband protocol stack with its own low-level driver. The DSMP kernel based Infiniband driver leverages the native RDMA capabilities of the Infiniband host channel adapter (HCA). This allows the HCA to service and move memory-page requests without processor intervention. Hence, RDMA eliminates the overhead for message construction and deconstruction, reducing system-wide latency.

Application-Driven, Memory Page Coherency Scheme: All proprietary shared memory mainframe computers maintain memory consistency and/or coherency via a hardware extension of the host processors cache-line coherency scheme. DSMP, which utilizes local and global memory resources, takes a different approach. Coherency within the local memory of each of the individual SMP servers is maintained by the x86-64 Memory Management Unit (MMU) on a cache-line basis. Memory consistency between a page in global memory and all copies of the page in local memory is maintained by the DSMP Linux kernel. This is further supported by a set of system calls that perform memory page lock operations similar to those used in database transactions:

- Allow a global memory page to be locked for exclusive access by a node
- Release the lock
- Force a local memory page to be immediately updated to global memory

This Symmetric Computing API allows programs with multiple execution threads, possibly running on multiple nodes, to maintain global memory page consistency across all nodes. This API, combined with some simple intuitive programming rules, makes porting an application to a multi-node DSMP platform simple and manageable. Those rules are as follows:

- Be sensitive to the fact that memory-pages are swapped into and out of local memory (cache) from global memory in 4096-byte pages.
- Since the granularity of a DSMP global memory lock is a 4096-byte page, it is important not to map data structures that need to be locked independently on the same memory page. A new malloc( ) option is provided to force alignment on a 4096-byte boundary when your application requires it. This ensures that data structures that need to be accessed independently by the program are on separate pages.
- Identify the cause of any page invalidation faults, and add locks to handle access to the affected pages by multiple threads.

- If there is a data-structure that can be accessed and modified by multiple threads, then the three new system calls can be used to maintain memory-consistency:
  1. msync( ) forces immediate synchronization of a data structure with its reference copy in global-memory
  2. mlock( ) prevents any other process thread from accessing and subsequently modifying the noted data-structure. mlock( ) also invalidates all other copies of the data structure (memory-pages) within the computing-system. If a process thread on another node accesses a memory-page associated with a locked data structure, execution is blocked until the structure (memory page) is released
  3. munlock( ) unlocks a previously locked data structure

Distributed POSIX Threads: The standard for parallelizing shared-memory C/C++ or Fortran programs is by using OpenMP and the POSIX thread library Pthreads. The DSMP implementation of Pthreads is within the kernel and operates transparently across all the nodes in the system. In addition, support is provided for localizing a thread on a particular node. Mutex and other synchronization primitives function across all nodes.

Distributed Process Execution and System V IPC: DSMP supports the launching of standard Linux processes on any selected node from the head node. In addition, System V IPC features such as shared memory segments are supported across all nodes.

Distributed Buffered File I/O: DSMP supports a distributed file I/O feature, where a process on the head node can open a file with a distributed file descriptor that will allow buffered file read and writes by threads or processes running on any node in the system.

DSMP Price-Performance

The table below compares MPI Clusters, SMP Mainframes and DSMP-enabled clusters:

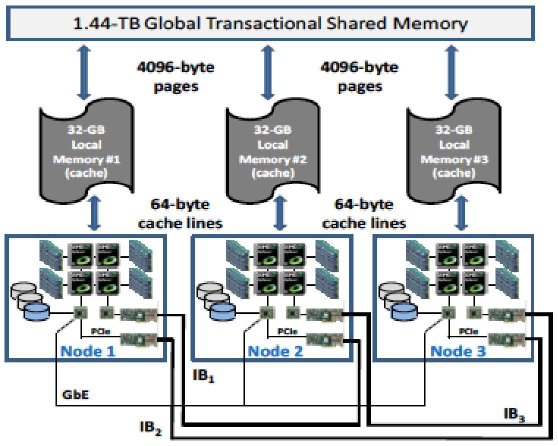
	MPI Cluster	SMP Mainframe	DSMP Cluster
Proprietary Hardware	No	Yes	No
Affordability	$	$$$	$
Shared Memory	No	Yes	Yes
Single Software Image	No	Yes	Yes
IPMI	Yes	Yes	Yes
RAS	Yes	Yes	Yes
Scalable	Yes	No	Yes

Performance of technical computing applications is largely a function of two metrics:

1. Processor performance (computational throughput) and;
2. Global Memory Read/Write performance (particularly random access).

Currently, DSMP cannot match mainframe performance in either of these metrics. However, due to the faster pace of development of industry standard processors and interconnects, both processors performance and global memory access performance are rapidly improving. The performance of latest generation industry standard processors already rivals that of proprietary mainframe processors. Over time the performance gap between a DSMP cluster and a proprietary SMP mainframe computer of equivalent processor/memory density will continue to narrow and eventually disappear. Given that the cost of the cluster is one-tenth the cost of an equivalent mainframe, it already dramatically excels in price / performance.

Today, Symmetric Computing is offering its direct connect family of departmental supercomputers. These are small DSMP clusters directly connected via infiniband without the need of a switch. They include two, three, four and five node system configurations, with shared memory capacities of 2 to 5 TB, and 128 to 320 processor cores. The following diagram shows a Trio Departmental Supercomputer with 192 processor cores and 3 TB of RAM. It is made with three homogeneous 4P (four processor sockets) servers each with 64 cores (using 16-core AMD Opteron™ 6380 series processors) and 1 TB of physical memory per node.


1.44-TB Global Transactional Shared Memory
4096-byte pages
4096-byte pages
32-GB Local Memory #1 (cache)
32-GB Local Memory #2 (cache)
32-GB Local Memory #3 (cache)
64-byte cache lines
64-byte cache lines
PCIe
Node 1
Node 2
Node 3
GbE
IB1
IB2
IB3

Looking forward, Symmetric Computing plans to introduce a multi-node Infiniband switch-based system delivering up to 2048 cores and 32 TB of RAM in a single 42U rack. In addition, we are working with our partners to deliver turnkey platforms optimized for application specific missions.

About Symmetric Computing

Symmetric Computing is a Boston based software company with offices at the Venture Development Center on the campus of the University of Massachusetts. We design software to accelerate the use and application of shared-memory computing systems for bioinformatics and life sciences, computer-aided engineering, energy, earth sciences, financial analyses and related fields. Symmetric Computing is dedicated to delivering standards-based, customer-focused technical computing solutions for users, ranging from Universities to enterprises.

For more information, please visit www.SymmetricComputing.com