Los Alamos NATIONAL LABORATORY

HIGH SPEED COMPUTING - Salishan, Gleneden Beach, OR, April 2003

Buffered Coscheduled (BCS) MPI

A Lightweight Deterministic Implementation of MPI

Juan Fernandez[1,2], Fabrizio Petrini[1] and Eitan Frachtenberg[1]
{juanf,fabrizio,eitanf}@lanl.gov
[1]Modeling, Algorithms and Informatics Group (CCS-3)
Los Alamos National Laboratory
[2]Computer Engineering Department
University of Murcia, SPAIN

Motivation

The development of parallel MPI programs for large-scale parallel machines is still a very time- and resource-consuming task. MPI programs are very difficult to develop, debug and maintain mainly due to their non-deterministic nature. These programs must produce the same results for the same inputs. However, the steps towards the solution are not necessarily taken in the same order since their messages may be exchanged in different sequences between executions.

BCS-MPI is a lightweight MPI implementation that represents a trade-off between simplicity and performance. It constitutes a new approach in facing the complexity of MPI code development for large-scale parallel machines. BCS-MPI allows the developer to control the level of non-determinism in a parallel application, for example, by sending all messages in the same order.

BCS-MPI has been succesfully validated with several applications that represent the ASCI workload.

Goals

Goals	Current Status	Future Work
-Target: large-scale parallel machines	-NIC-based implementation on state-of-the-art hardware (low level of intrusion)	-Improved Functional Debugging
-Simplify the design of the communication library and its implementation	-Integrated Monitoring and Debugging System which provides different levels of non-determinism	-Job Prioritization
-Minimize/eliminate non-determinism during the execution of MPI programs	-Most existing scientific codes run efficiently with BCS-MPI (based on MPICH)	-μKernel Implementation
-Automatic functional and performance debugging of MPI programs		-Checkpointing
-Minimal performance penalty		-Fault Tolerance

Design

Intuition: a SIMD communication library runs MIMD MPI programs.

Hierarchical design based on a basic set of communication/synchronization primitives.

Global scheduling of computation, communication and synchronization operations for MPI user code: Global Heartbeat (500µsec time slices).

System activitities are organized in microphases within every time slice.

NIC-based OS-bypass implementation.

Scalability is facilitated by tightly coupling the collective communication operations with the collective primitives provided by the hardware.

Integrated Monitoring and Debugging Mode which provides selectable level of non-determinism (in the strictest mode, the system is able to rerun an arbitrary large parallel program in a completely deterministic way).

Integration as a plugin in a resource management system for parallel jobs.

Figure 1
Blocking Communication

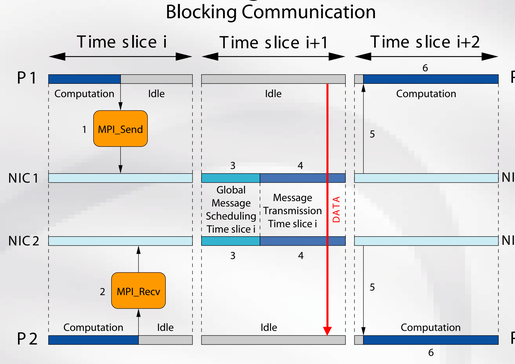

Figure 1. Blocking Send/Receive Scenario.
Process P1 sends a message to process P2 and P2 receives a message from P1:
1) P1 posts a send descriptor to the NIC and blocks.
2) P2 posts a receive descriptor to the NIC and blocks.
3) The transmission of data from P1 to P2 is scheduled since both processes are ready (all the pending communication operations posted before time slice i are scheduled if possible).
4) The communication is performed (all the scheduled communication operations are performed before the end of time slice i+1).
5) P1 and P2 are restarted at the beginning of time slice i+2.
6) P1 and P2 resume computation.

Note that the delay per blocking primitive is 1.5 time slices on average.

Figure 2
Non-Blocking Communication

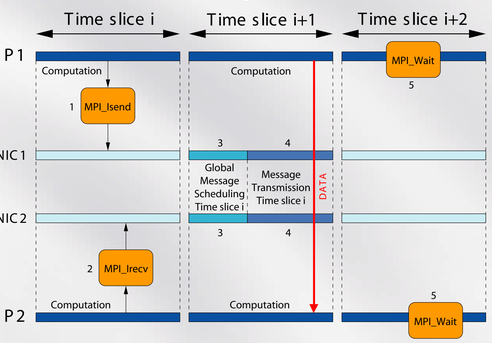

Figure 2. Non-Blocking Send/Receive Scenario.
Process P1 sends a message to process P2 and P2 receives a message from P1:
1) P1 posts a send descriptor to the NIC.
2) P2 posts a receive descriptor to the NIC.
3) The transmission of data from P1 to P2 is scheduled since both processes are ready (all the pending communication operations posted before time slice i are scheduled if possible).
4) The communication is performed (all the scheduled communication operations are performed before the end of time slice i+1).
5) P1 and P2 verify that the communication has been performed and continue their computation.

In this case, the communication is completely overlapped with the computation with no performance penalty.

Performance Evaluation

Cluster Configuration
- 32 Dell 1550 compute nodes,
- Dell 2550 management node
- 128-port Quadrics switch

Compute Node Configuration
- two 1GHz Pentium-III processors
- 1GB of ECC RAM
- 2 independent 66MHz/64-bit PCI buses
- Quadrics QM-400 Elan3 NIC
- 100Mbit Ethernet NIC

Software Configuration
- Red Hat Linux 7.3 with Quadrics kernel
- Sweep3D
- SAGE (timing_h.input)
- NAS Parallel Benchmarks 2.4 (Class C)
- All experiments run on 16 PEs

Figure 3
Comparison between BCS-MPI and Quadrics MPI (based on MPICH)

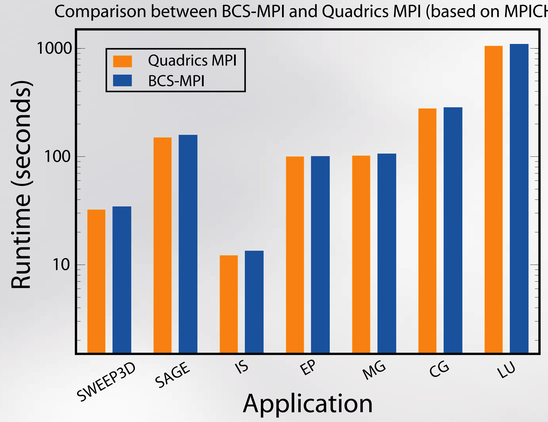

Figure 4
Slowdown BCS-MPI vs. Quadrics MPI

Application	Slowdown
SWEEP3D	7.14%
SAGE	5.64%
IS	10.10%
EP	1.16%
MG	4.35%
CG	2.66%
LU	4.00%