Buffered Coscheduled (BCS) MPI
A New Approach in the System Software Design for Large-Scale Parallel Computers

Juan Fernández1,2, Fabrizio Petri1 and Eitan Frachtenberg1
{juan,fabrizio,eitanf}@lanl.gov
1Modeling, Algorithms and Informatics Group (CCS-3)
Los Alamos National Laboratory
2Computer Engineering Department
University of Murcia, SPAIN

Motivation

BCS-MPI introduces a new approach to design system software for large-scale parallel machines. The goal is to reduce the complexity, non-determinism and redundancy of the main components of the system software with a minimal performance penalty.

BCS-MPI globally organizes all the system activities at a very fine granularity. Both computation and communication are scheduled at regular intervals, in a real-time fashion, and the scheduling decisions are taken after a global exchange of control information.

BCS-MPI is a lightweight MPI implementation that represents a trade-off between simplicity and performance. It is designed on top of a minimal set of communication primitives that are almost entirely implemented in the network interface card.

BCS-MPI has been successfully validated with several scientific codes representative of the ASCI workload.

Goals

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

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 activities are organized in micropipelines 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.

Performance Evaluation

Cluster Configuration
- 32 HP rx8600 compute nodes
- 128-port Quadrics switch

Compute Node Configuration
- Dual Itanium-II processor
- 2 GB of ECC RAM
- 123 Mbps LX8/64-bit PCI-X buses
- 2 Quadrics QM-400 Eitan3 NIC
- 100 Mbit Ethernet NIC

Software Configuration
- Red Hat Linux 7.2
- Intel C/Fortran 7.1.17
- SWEEP3D (50x50x50)

Results
- Comparison between BCS-MPI and Quadrics MPI for different numbers of processors.

BCS-MPI provides similar performance of a production-level MPI, with a much simpler design and implementation!!!