Open MPI for Cray XE/XK Systems

Samuel K. Gutierrez – LANL
Nathan T. Hjelm – LANL
Manjunath Gorentla Venkata – ORNL
Richard L. Graham - Mellanox
Cray User Group (CUG) 2012
May 2, 2012
A Collaborative Effort
First Things First – Open MPI Overview

- Open-Source Implementation of the MPI-2 Standard
- Developed and Maintained By
  - Academia
  - Industry
  - National Laboratories
- Supports a Range of High-Performance Network Interfaces
  - Infiniband
  - Cray SeaStar
  - … and Now Cray Gemini
The Gemini System Interconnect[^3] – An Overview

- Network Used by the Cray XE and XK System Families
- Successor to the Cray SeaStar* Network Interconnect
- 3D Torus Network Built of Gemini ASICs

**Gemini ASIC**
- Provides 2 NICs and a 48-port Router
- Connects 2 Opteron Nodes
- Provides 10 Torus Connections – 2 x (+X, -X, +Z, -Z) – 1 x (+Y, -Y)
Open MPI’s Plugin Architecture – A High-level Overview

User Application

MPI API

Modular Component Architecture (MCA)
Open MPI’s Plugin Architecture – A High-level Overview

- **MPI API**
  - E.g. MPI_Send, MPI_Recv, MPI_Bcast
Open MPI’s Plugin Architecture – A High-level Overview

- **Modular Component Architecture (MCA)**
  - Backbone of Open MPI
  - Plugin System
  - Finds, Loads, and Parameterizes Components

- **Open MPI Hearts MCA Parameters**

## Modular Component Architecture (MCA)

<table>
<thead>
<tr>
<th>Framework</th>
<th>Framework</th>
<th>...</th>
<th>Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Component</td>
<td>...</td>
<td>Component</td>
</tr>
<tr>
<td>Component</td>
<td>Component</td>
<td>...</td>
<td>Component</td>
</tr>
<tr>
<td>Component</td>
<td>Component</td>
<td>...</td>
<td>Component</td>
</tr>
<tr>
<td>Component</td>
<td>Component</td>
<td>...</td>
<td>Component</td>
</tr>
<tr>
<td>Component</td>
<td>Component</td>
<td>...</td>
<td>Component</td>
</tr>
</tbody>
</table>
Open MPI’s Plugin Architecture – A High-level Overview

- **Frameworks**
  - Functionality Specification
  - E.g. Resource Manager, Point-to-Point, Collective Algorithm
Open MPI’s Plugin Architecture – A High-level Overview

- Components
  - Implementation of a Framework Type – A Plugin
  - E.g. SLURM RAS Component, Open IB BTL Component

- What a Developer Typically Creates to Support New Functionality

- Module: an Instance of a Component
Open MPI’s Plugin Architecture – Main Code Sections

- **Open MPI Layer (OMPI)**
  - MPI API and Support Logic

- **Open Run-Time Environment (ORTE)**
  - Run-Time System

- **Open Portability Access Layer (OPAL)**
  - OS-Specific/Utility Code
The Port - ORTE

- **Environment-Specific Services (ESS)**
  - Run-Time Environment (RTE) Setup
  - Messaging, Routing, Module Exchange (ModEx)
  - Process Naming – Job Size and Locality Information

- **Process Lifecycle Management (PLM)**
  - Central Switchyard for All Process Management

- **Resource Allocation Subsystem (RAS)**
  - Job Resource Availability and Allocation

- **RML Routing Table (ROUTED)**
  - “Next Hop” Routing Services – De Bruijn
OMPI Point-to-Point Overview

MPI API

PML

BML

BTL₁

BTL₁ MPool

RCache

...

...

BTLₙ

BTLₙ MPool

RCache
Byte Transfer Layers (BTLs)

- **Transport Interface Support Plugins**
  - Think: Byte Transfer Driver

- **Thin Abstraction Layer Above Target Device**
  - Source/Destination Preparation
  - Protocol Definition – Short, Medium, Long
  - Send, SendI, Put, Get

- **No Notion of MPI Semantics**
The Port: New BTLs

- Kernel-Assisted (Single Copy) Shared Memory BTL
  - Used Exclusively for Intra-Node Communication
  - Currently Named vader in Development Trunk

- Gemini BTL
  - Used Exclusively for Inter-Node Communication
  - Leverages Cray’s Generic Network Interface (uGNI)
  - Currently Named ugni in Development Trunk
BTL Management Layer

- Manages Multiple BTLs Within in Single Process
- No Modifications Needed for Port
Point-to-Point Management Layer

- Provides Point-to-Point Functionality Required by the MPI Layer
- Minor Modification Required for Port
More About the XPMEM BTL - Vader

- **MPICH Nemesis-like Design**
  - Lock-Free Message Queues
  - “Fast Boxes” – I.e. Per-Peer Receive Queues for Short Messages

- **Copy Backend Changes Based on Message Size**
  - E.g. `bcopy [a,b)` - `memcpy` Otherwise
  - User Tunable with Good Defaults

- **Cross-Process Memory Mapping Allows for RDMA-Like Semantics**
  - Copy-In/Copy-Out (CICO) Avoided
  - No Backing Store Required
  - Heavy Use of Registration Cache

- **XPMEM Support Requires Kernel Patch and User-Level Library**
  - Already Available and Leveraged by Cray’s Native MPI Implementation
More About the uGNI BTL

- **Protocols**
  - Short Message – Fast Memory Access (FMA) Short Messaging (SMSG)
  - Medium Message – FMA RDMA
  - Long Message – Block Transfer Engine (BTE) RDMA

- **Lazy Connection Establishment**
  - Resource Utilization Directly Related to Application Communication Characteristics
Improved Collectives: Cheetah²

- **ORNL’s Cheetah – A Framework for Collective Operations**
  - Collectives Implemented with Collective Primitives
  - Each Primitive is Optimized for a Particular Communication Path
  - Progressed Asynchronously and Independently When Semantics Permit
Improved Collectives: Cheetah

- Base Collectives (BCOL) – Implements Collective Primitives
- Subgrouping (SBGP) – Provides Process Grouping Rules
- Multilevel (ML) – Coordinates Collective Primitive Execution
- For Design and Implementation Details: See Cheetah Publications
**Improved Collectives: uGNI BCOL Barrier**

- Implemented uGNI Cheetah Barrier
  - Fan-In/Fan-Out Algorithm
  - Atomic Barrier Leverages Atomic Operations Provided by the uGNI Library
  - Currently Only Supports MPI_Barrier
Performance Evaluation - Setup

- **Test Beds**
  - Cielo - 142,304 Core XE6
  - Enhanced Jaguar – 299,008 Core XK6

- **Point-to-Point Latency**
  - OSU’s MPI Mico-Benchmark Suite – osu_latency & osu_multi_lat

- **Point-to-Point Bandwidth**
  - OSU’s MPI Mico-Benchmark Suite – osu_bibw & osu_mbw_mr

- **Barrier Latency**
  - MPI_Barrier in a Tight Loop – Average Latency Reported
Vader Latency on AMD Magny-Cours

OSU Latency: Shared-Memory - Open MPI versus Native MPI

- Latency: Native MPI - 2 Processes
- Multi Latency: Native MPI - 16 Processes
- Latency: Open MPI - 2 Processes
- Multi Latency: Open MPI - 16 Processes

Message Length (Bytes) vs. Latency (Microseconds)
Vader Bandwidth on AMD Magny-Cours
uGNI BTL Latency on XE6

OSU Latency: uGNI - Open MPI versus Native MPI

- Latency: Native MPI - 2 Processes (1 Process per Node)
- Multi Latency: Native MPI - 8 Processes (1 Process per Node)
- Latency: Open MPI - 2 Processes (1 Process per Node)
- Multi Latency: Open MPI - 8 Processes (1 Process per Node)
uGNI BTL Bandwidth on XE6
Performance of Cheetah Barriers on XK6
Ongoing/Future Work

- **Point-to-Point Stabilization/Optimization**
  - Already Tested at 128k Processors (Cielo)
  - Investigating New Protocols

- **Continue Collectives Work**
  - Evaluate Performance and Scalability Characteristics of the Atomic Collective Operations at Larger Scales
  - Evaluate the Potential for Implementing Other Collective Operations Using the Atomic Collective Operations

- **Work with Friendly Testers**

- **Prepare for General Release**
Thanks!
Questions?

- Questions?
- Comments?
References

