LLNL-PRES-409460

MPI Forum Tools WG

Status and Current Activities

Martin Schulz

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

MPI Tools WG

- Founded at MPI Forum Meeting in September
- Goals of the WG:
  - Define interfaces for tools
    - Correctness & Performance tools
    - Provide standardized access to internal information
  - Extend/Build on idea/concept of PMPI interface
- Current status:
  - Started regular Telcons (biweekly, Monday 8am PT)
  - Defined agenda and initial focus

Structure

- Two main groups of interfaces
  - Tool Deployment Interfaces
    - MPIR and extensions for dynamic process creation
    - DLL discovery
    - MPI handle query interface
  - Introspection Interfaces
    - Performance Information (PIMPI)
    - MPI State Information
    - Message Queue Debugging
- Issue should be orthogonal

Immediate Goals

- Formalize the MPIR “standard”
  - Originally developed for/with/by Totalview
  - Used by many MPI tools to find and attach to all processes of an MPI job
  - De facto standard with many variations and extensions
- Goals
  - Survey existing extensions
  - Provide single standard document
- Next steps:
  - Introduce new helper APIs for tool deployment
  - Discuss state/performance information queries

MPI Forum Tools WG

MPIR Interface

Lawrence Livermore National Laboratory, P. O. Box 808, Livermore, CA 94551

This work performed under the auspices of the U.S. Department of Energy by
Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344

Functionality

- Initially designed for/with/by Totalview
  (hence currently often referred to as debugger interface)
  - Automated way for TV to find MPI ranks
  - Useful for any third party MPI tool
- Provides host/PID pair for all MPI ranks
  - No need to interact with scheduler
  - Sufficient information to locate and attach to tasks
- Allows tools to control MPI codes
  - From startup
  - Using dynamic attach

Variables (from wiki)

- **MPIR_proctable**
  - This is an array containing hostname, executable name, and PID for every MPI rank
- **MPIR_proctable_size**
  - Number of entries in MPIR_proctable
- **MPIR_debug_state (rename?)**
  - This int has value 0 before MPIR_proctable is initialized and MPIR_DEBUG_SPAWNED after MPIR_proctable has been filled in.
- **MPIR_Breakpoint()**
  - This routine is called by the MPI implementation during MPI_Init after MPIR_proctable is setup. A debugger can set a breakpoint here to provide a point to stop after MPI_Init has set up all MPI processes but before exiting MPI_Init.

Additional variables needed

- **MPIR_being_debugged**
  - Set to 1 to notify MPI implementation that a tool is waiting for a call to MPI Breakpoint
- **MPIR_debug_gate**
  - Used in MPICH-based implementations
  - Release MPI after tool has attached
- **MPIR_debug_abort_string** (Open MPI only?)
- **MPIR_debug_state** (Open MPI only?)
  - 2 if MPI_Abort was called – notify debugger that something has happened, set string to error, then call to MPIR_Breakpoint
  - Reason why MPIR_Breakpoint is called
  - 0 (first call), 1 (commspawn), 2 (abort)

Workflow from tool's side (1)

- Tool attaches to target process and stops the process
  - COMM_WORLD/rank 0
  - Resource manager controlling the MPI job (e.g., srun, mpirun/mpiexec)
- Tool probes to determine which symbols are present
  - None is valid, but not desired
  - Defines level of support available by this MPI
- If (MPIR_debug_state is not set to 1)
  - Tool inserts breakpoint at MPIR_Breakpoint
  - Tool set MPIR_being_debugged to 1
  - Tool resumes process
  - Tool waits for breakpoint to be called

Workflow from tool's side (2)

- Assert (MPIR_debug_state==1)
- Tool reads MPI_proctable_size and MPI_proctable
- Extensions: co-locating daemons
- Start tool support infrastructure
  - Launch tool daemons on compute nodes (if nec.)
- Warning: some implementations require that tools attach to all processes to get them out of a suspended state
  - Can attach to all/subset of processes
    - In this case resume after setting MPIR_debug_gate to 1
- Tool sets MPIR_debug_gate to 1
- Resume target process
  - Returning from breakpoint
  - Resuming process
- Question: what happens if abort is called by a process that is not under debugger control -> forwarded to all processes (only if under debugger/tool control)

Portability Issues

- Location of API
  - Task 0 (MPICH-1)
  - MPI Daemon (OpenMPI, MPICH-2)
  - External resource manager (SLURM, POE, …)
- State of MPI task at MPIR_Breakpoint
  - All ranks not initialized (SLURM)
  - All tasks with initialized MPI hanging in MPI_Init
- Extensions
  - Tool daemon startup on Blue Gene Platforms
  - Others?