TotalView / PMIx / MPIR2

John DelSignore (RWS)
Ralph Castain (Intel)

December, 2016

What's the Problem?

- MPIR1 is..
  - Not part of the MPI standard
  - Not an API
  - Not able to handle modern MPI features
  - Not scalable (supposedly)
- MPI Tools Working Group (MPIWG-Tools) would like to define MPIR2
  - https://github.com/mpiwg-tools
- To make progress, in December, 2016…
  - Ralph Castain (Intel) volunteered PMIx as a partial MPIR2 solution using the PMIx Reference Server
  - John DelSignore (RWS) volunteered to study/test PMIx/debugger integration

Overview

- MPIR1: What's right? What's wrong?
- MPIR2: What should it be?
- What is PMIx?
- PMIx's role in MPIR2?
- PMIx Work, December, 2016, status

MPIR1: What's right? What's wrong?

- **MPIR1 is a defacto-standard, not a standard. But…**
  - Has withstood the test of time (20+ years).
  - Widely implemented.

- **Process acquisition.**
  - MPIR1 is not an API
    - A "rendezvous" protocol for process acquisition (discovery of process pids, nodes, and executable)
  - Requires the tool debug the starter process
    - Supports only one tool at a time, limits tool composition (e.g., Spindle + TotalView)
  - Requires symbols and types be properly preserved in the starter process
    - Simple in theory, problematic in practice
  - Theoretically a scalability bottleneck
- **Tool daemon spawning.**
  - No standard tool daemon spawning support
  - Co-spawning at job launch with Blue Gene MPI and Open MPI
- **MPI task labeling.**
  - Limited to "index in proctable" == "rank in COMM_WORLD"
  - Does not support advanced MPI constructs
    - Threads as MPI tasks (CEA MPC)
    - Dynamic processes, sessions, etc.

MPIR2: What should it be?

- **MPIR2 *would* be part of the MPI Standard**

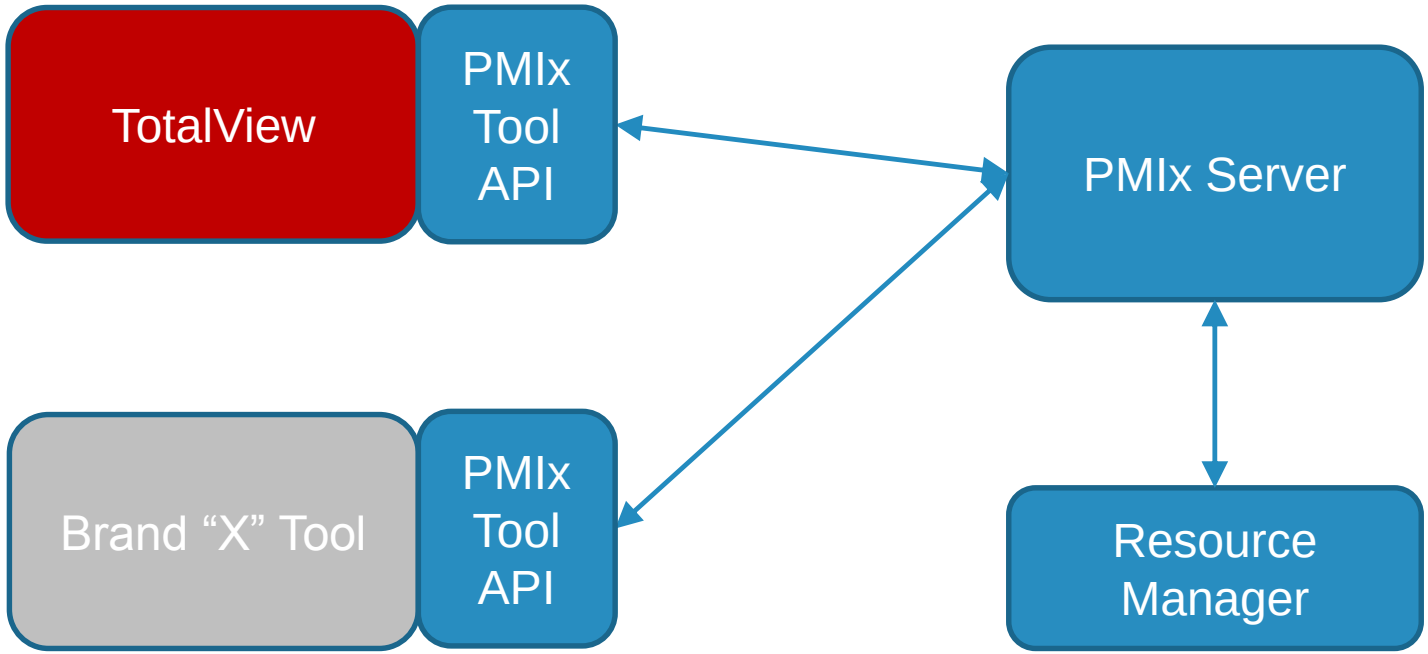
- **Process acquisition.**
  - Make it an API-based interface, not debugger/symbol table based.
  - A tool could discover the processes in a parallel job (but not necessarily an MPI job).
    - Allow query of proctable-like info of process pids, nodes, and executables.
    - Support for multiple tools (avoid tracing the starter process).
  - Support dynamic process creation (e.g., MPI2), sessions support.
- **Tool daemon spawning.**
  - A capability that allow a tool to spawn its daemon processes.
    - Co-spawn with application job: Launches the tool daemon processes along side the application processes, requires Resource Manager (RM) support. Currently Blue Gene and Open MPI only.
    - As a separate tool job: Independent of the application job, launch at any time.
- **MPI task labeling.**
  - A capability where any time after process acquisition, the debugger can discover a process's or thread's "role" as an MPI task, and "label" it accordingly.
  - PMIx has no knowledge of MPI structure, but could provide a vehicle for the MPI implementation to communicate it to the tool.
- **MPI Message Queue Dumping?**
- **MPI Opaque Handles?**

What is PMIx?

- PMIx is an API
- Ralph says:
  - PMIx itself doesn't actually do anything. It just communicates requests, and returns responses. PMIx doesn't launch anything, it doesn't map processes or assign ranks, or anything.
  - It just conveys your request to launch an application to the RM, and merrily transports the reply back to you.
- PMIx Reference Server implementation
  - https://github.com/pmix/pmix-reference-server
  - git clone git@github.com:pmix/pmix-reference-server.git
- Vendors (like IBM Spectrum MPI) can create their own implementations

PMIx Tool Basics

- Tool is compiled/linked with the PMIx Tool header/library (e.g., pmix_tool.h / libpmix.so)
- Tool “connects” to the PMIx server (Unix Domain Socket), uses separate “namespaces”
- Tool makes a request (e.g., “query the proctable”)
- PMIx Server asks the Resource Manager
- PMIx returns result to the tool

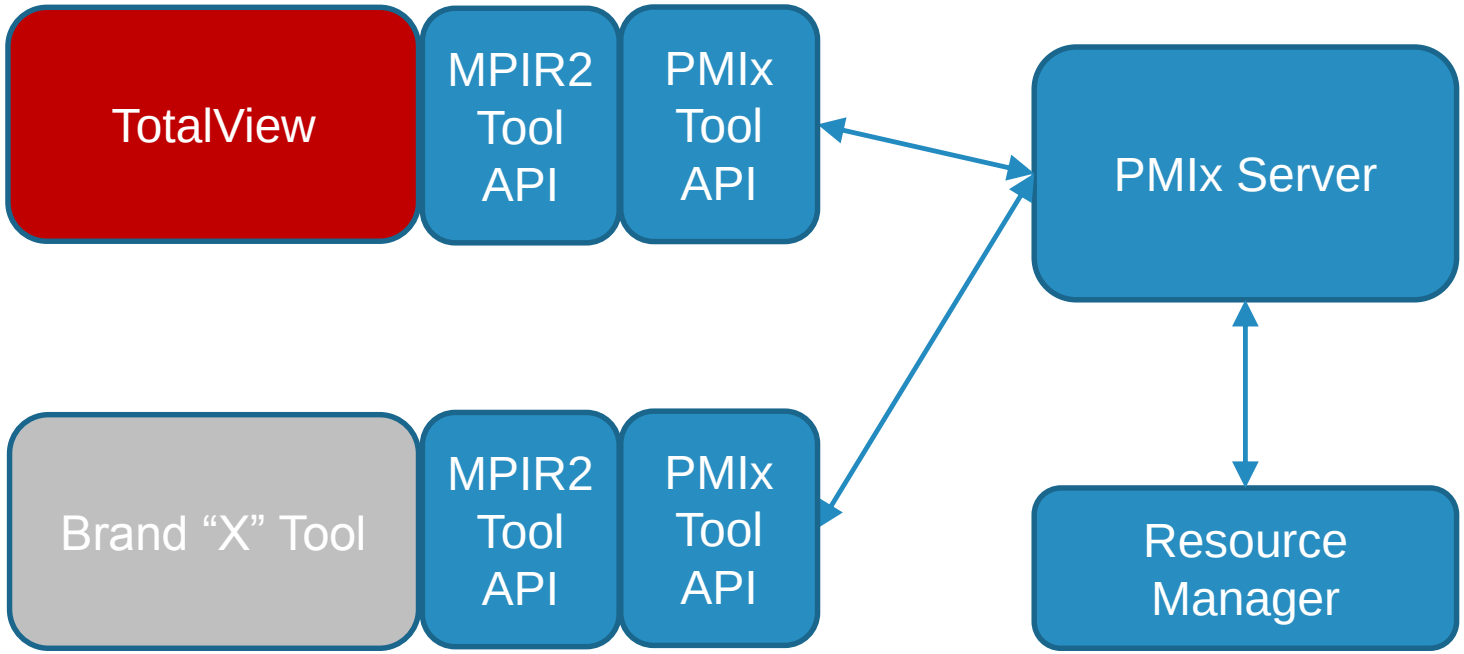


PMIx Tool Example

- PMIx_tool_init()
  - Initialize the PMIx library linked to the tool (e.g., in libpmix.so)
  - Connect to the PMIx server
  - Creates a PMIx thread in the tool
- PMIx_Register_event_handler()
  - Non-blocking PMIx calls result in a callback to the tool
  - Callback made from the PMIx thread
- PMIx_Query_info_nb()
  - Get RM capabilities, proctable, other info
- PMIx_Spawn()
  - Spawn debugger daemon processes
- PMIx_tool_finalize()
  - Cleanup and shutdown services

PMIx's Role in MPIR2

- TBD, but here's what I'm thinking…
- Use PMIx as a *test vehicle* for figuring out an official MPIR2 interface
- MPIR2 would be a PMIx-like API
- Making it simple to wrap PMIx with MPIR2 symbol names



Beyond the Basics

- PMIx can answer many questions and allow control, for example
  - Jobs on the system
  - Memory footprint for the nodes, procs, or job
  - Network integration
  - Job control (suspend, resume, kill)
  - Advantage: Enables tools to provide more information to the users
- PMIx's API is easily extended by adding tags, not functions
  - Allows flexibility as programming models change
  - Perhaps MPIR2 should be too
- Support for non-traditional MPI, as well as non-MPI, models desirable
  - Mostly, the tools don't care if the application is using MPI or not

PMIx Work, December 2016

- Ralph Castain worked on PMIx support for tools
  - Added tool support to PMIx Reference Server
  - Created examples of “debugger”, “debugger daemon”, and “client application”
- John
  - Cloned PMIx Reference Server git repository
  - Tested Ralph’s drops, provided feedback
  - Prototyped PMIx testing in the TV test rig
  - No changes to TV itself (yet)

What's Next?

- Ralph to
  - Finish support for "attach to running job"
- John to
  - Ascertain PMIx/MPIR2 interest at RWS
  - Communicate current progress back to MPIWG-Tools
  - More TV/PMIx integration? TBD.
- MPIWG-Tools
  - Interested in MPIR2/PMIx connection?
  - How to handle MPI labeling API?
  - What to do with existing MQD API? Part of MPIR2?
  - Does Jeff Squyres' MPI Opaque Handles API have a future?


RogueWave
SOFTWARE
Accelerating Great Code