Euro PVM / MPI

Bonn, Germany
Sept 2006

An Interface to Support the Identification of Dynamic MPI 2 Processes for Scalable Parallel Debugging

Chris Gottbrath, Brian Barrett, Bill Gropp, Rusty Lusk, and Jeff Squyres

Organization

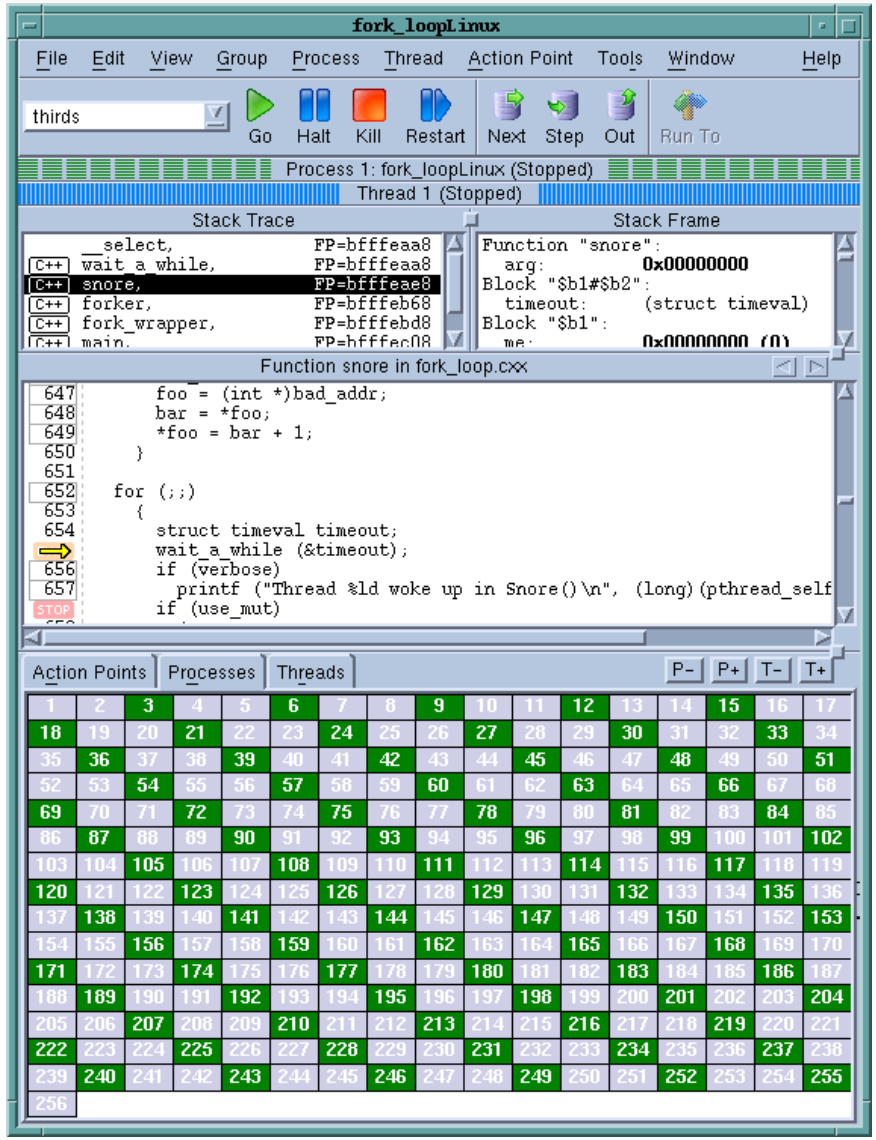
- Context
- Challenge
- Solution
- Status

Organization

- **Context**
  - What is TotalView?
  - State of MPI support now
  - Relevant Debugger Characteristics
- **Challenge**
- **Solution**
- **Status**

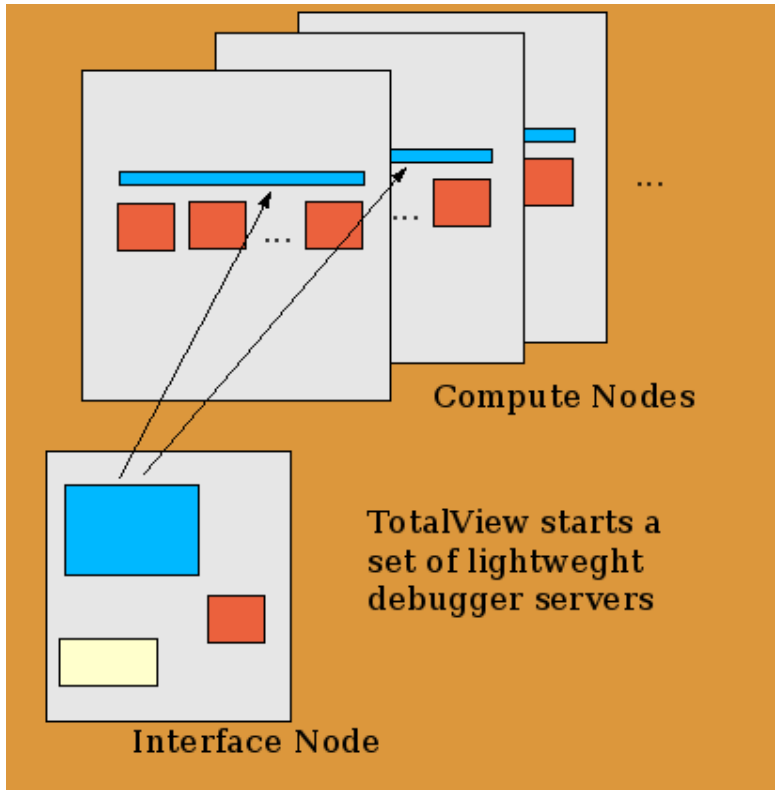
What is TotalView?

- **Source Code Debugger**
  - Proprietary
  - C, C++, Fortran 77, Fortran90, UPC
  - Wide compiler and platform support
  - Multi-threaded Debugging
  - Parallel & Remote Debugging
  - Memory Debugging Capabilities
  - Powerful and Easy GUI
  - CLI for Scripting
- **Used**
  - on ~ 90% of the top 100 supercomputers
  - by thousands of others!



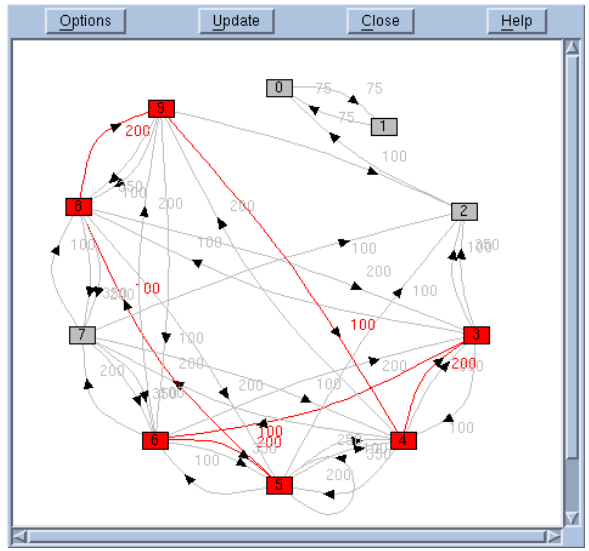
Architecture for Cluster Debugging

- **Cluster Architecture**
  - Single Front End (TotalView)
    - GUI and debug engine
  - Debugger Agents (tvdsvr)
    - Low overhead, 1 per node
    - Traces multiple rank processes
  - TotalView communicates directly with tvdsvrs
    - Not using MPI
    - Optimized Protocol
- **Provides: Robust, Scalable, Minimal Interaction**



MPI Support in TV

- **Process Acquisition**
  - Way to find the processes
  - Interface with the MPI library
  - Subset attach, Attach to hung jobs
  - TV uses MPI ranks to identify nodes



- **Bulk Launch**
- **MPI Message Queue Display**
  - Interface with the MPI library
  - Graphical display
- **Parallel Debugger**
  - Process Control, Breakpoints, Data Display, etc..

Requirements for Process Acquisition

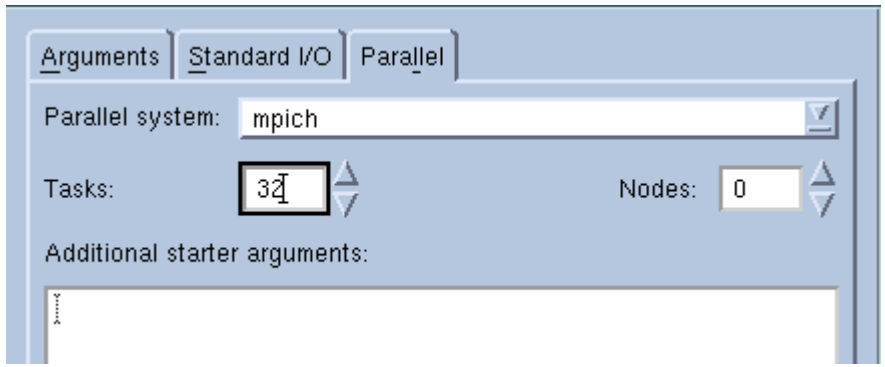
- **TV needs info**
  - MPI Rank
  - Host
  - PID
  - Executable Name
- **TV needs help**
  - Keep the MPI processes from running away
- **The MPI library**
  - Knows this info
  - Is a more constant part of the environment than the resource manager

MPI 1 Process Acquisition Interface

- **MPI defines a process table**
  - Global variable
  - Structure type with specified fields
- **TotalView reads & writes data**
  - Directly in the target program using ptrace()
- **Synchronization**
  - TV notified target being debugged
  - MPI notifies TV that the table is ready
  - MPI processes pause and wait to be attached
  - TV releases the process
- **Widely Adopted**
  - Reference implementation in MPICH

Process Acquisition in TV

- **Simple Start Up**



- **Restarts**
- **Subset attach**
  - Helps with information overload
  - Scalability

Organization

- **Context**
- **Challenge**
  - What is MPI 2
  - MPI 2 Dynamic
  - Goal
- **Solution**
- **Status**

MPI 2

- **Dynamic Process Operations**
  - Spawn(), Connect()/Accept(), Join()
  - Allows MPI applications to change the fundamental set of processes at runtime
  - MPI Services
- **MPI RMA (1 sided) Operations**
- **MPI I/O**
- **Obscure Parts**
  - Language bindings
  - User Defined Collectives

MPI 2 Dynamic Processes

- **MPI 1**
  - MPI processes are static
  - MPI COMM WORLD uniquely identifies each process
- **MPI 2**
  - Processes Can Change
    - Spawn() adds processes
    - Connect()/Accept() and Join() merge separate jobs
  - MPI_COMM_WORLD no longer provides a unique ID

Goal for this work

- **MPI 2**
  - Provides new flexibility to the user
  - Takes away comfortable notions for the debugger
  - Users can't debug if you can't attach
- **An MPI 2 aware proctable**
  - Easy for the user
  - Performance
  - Portability
  - Functionality
  - Reliability

Organization

- Context
- Challenge
- Solution
  - Requirements
  - Architecture
  - Structure
- Status

Requirements for MPI 2 Process Acquisition

- **All dynamic operations**
  - Spawn(), Connect()/Accept(), Join()
- **Launching the job**
- **Attaching to the job**
  - Even a hung job
- **Subset Attach**
- **Naming of MPI tasks**
  - Uniquely, Repeatably
- **Low Performance Impact**
  - On by default has to be reasonable

MPI 2 Process Acquisition

- **Process table**
  - A bit more info in each entry to allow for naming
- **Distribute the process table**
  - The MPI does not have to maintain a central table with all rows.
    - That could turn into a choke-point
    - Have the MPI do just enough communication but no more
  - The debugger marshals the data when needed.
    - Minimize runtime cost that is payed even with no debugging
- **Synchronization**
  - Allow for debugging of newly spawned processes

Process Table Entries

- **Each Row Encodes**
  - System Level Info
    - Host Name
    - Executable Name
    - Pid
  - MPI Level Info
    - unique id for the MPI_COMM_WORLD to which the process belongs
    - Rank in MPI_COMM_WORLD of the process
    - Spawn Description
      - parent communicator, rank, and sequence

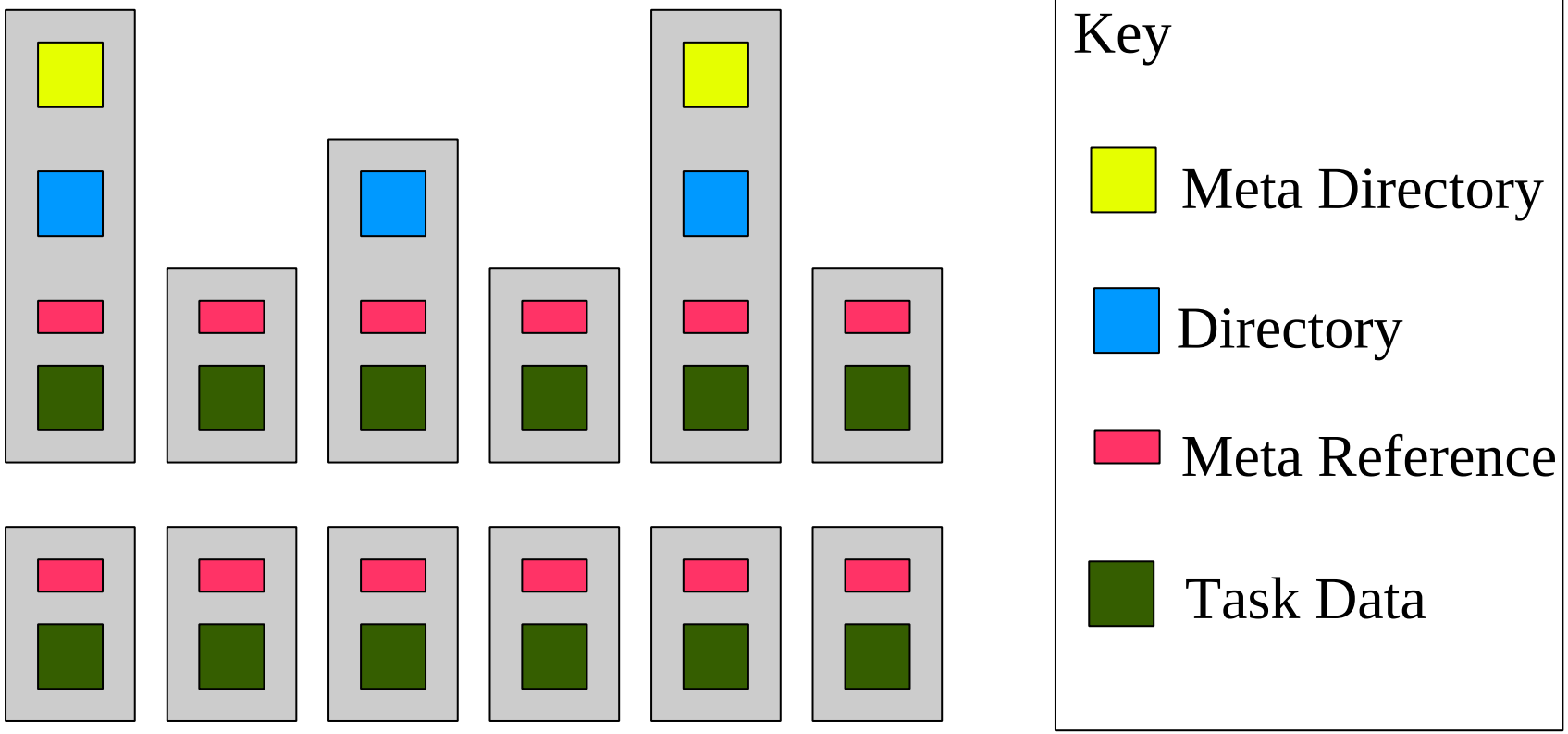
ETNUS TOTALVIEW

Distribution of the table

- **MPI task**
  - Has a reference to a Meta directory
- **MPI directory task**
  - Has a portion of the process table
- **MPI meta directory task**
  - Knows about directory processes
  - Knows about other meta directories

ETNUS
TOTALVIEW

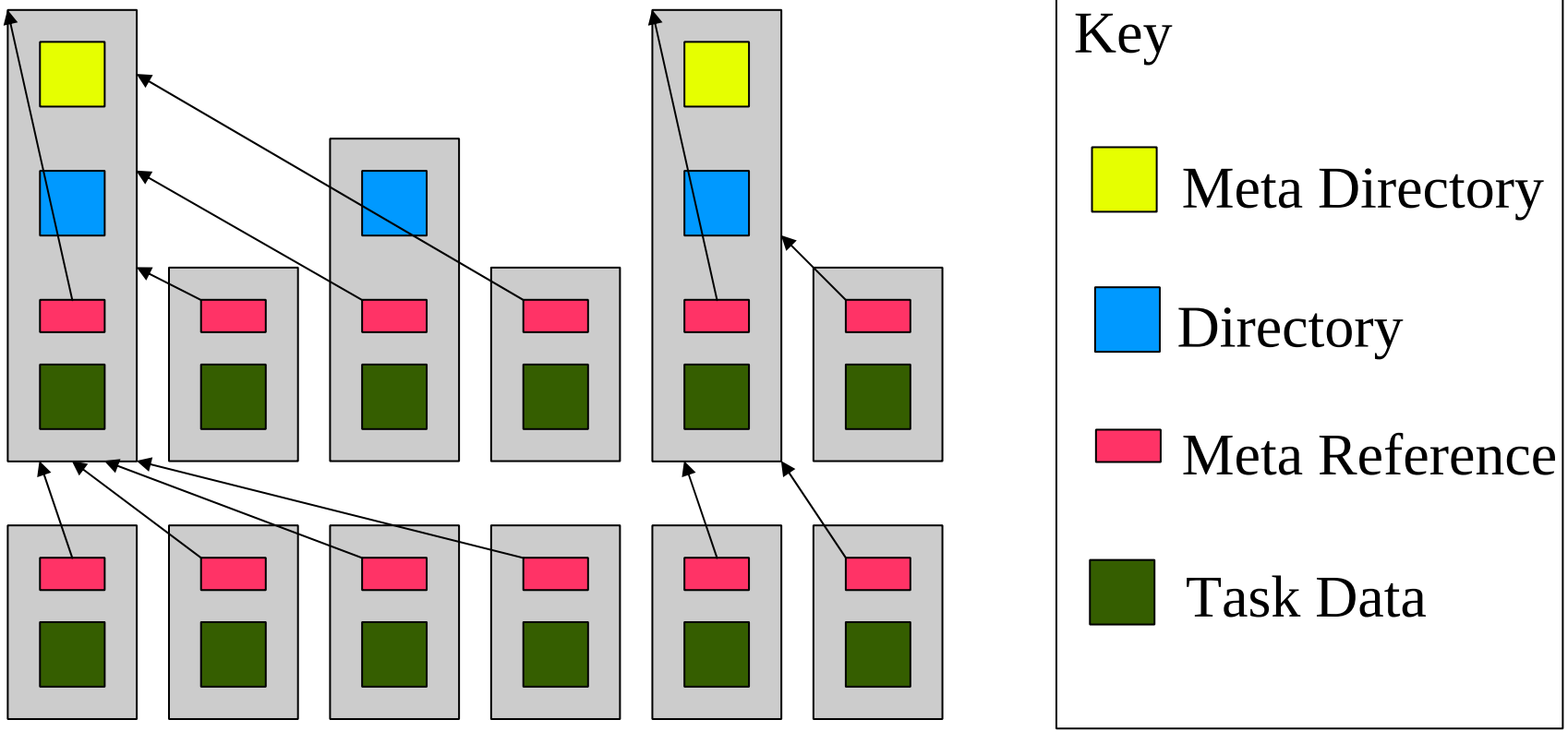
MPI 2 Process Acquisition



ETNUS TOTALVIEW

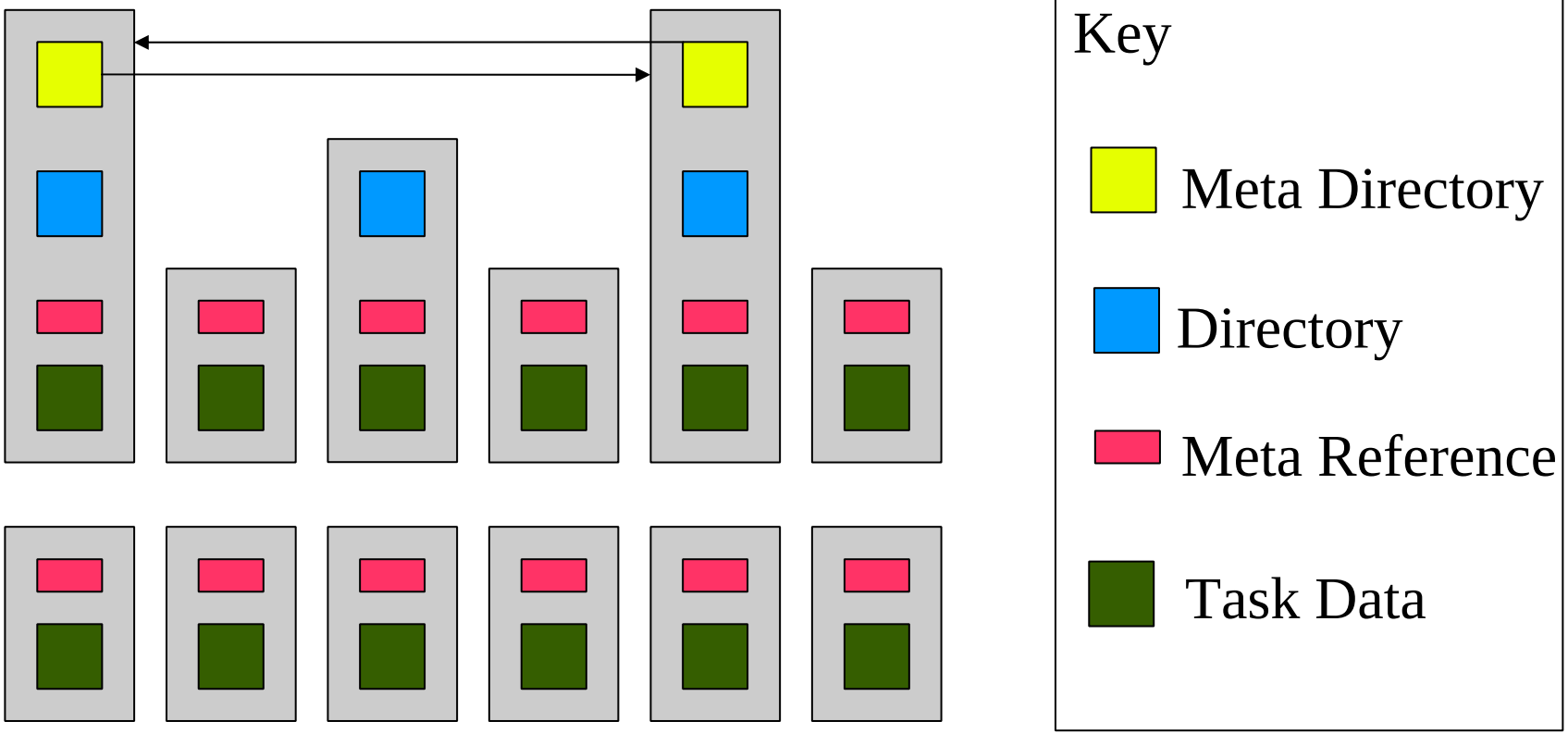
MPI 2 Process Acquisition

Finding the Meta Directory



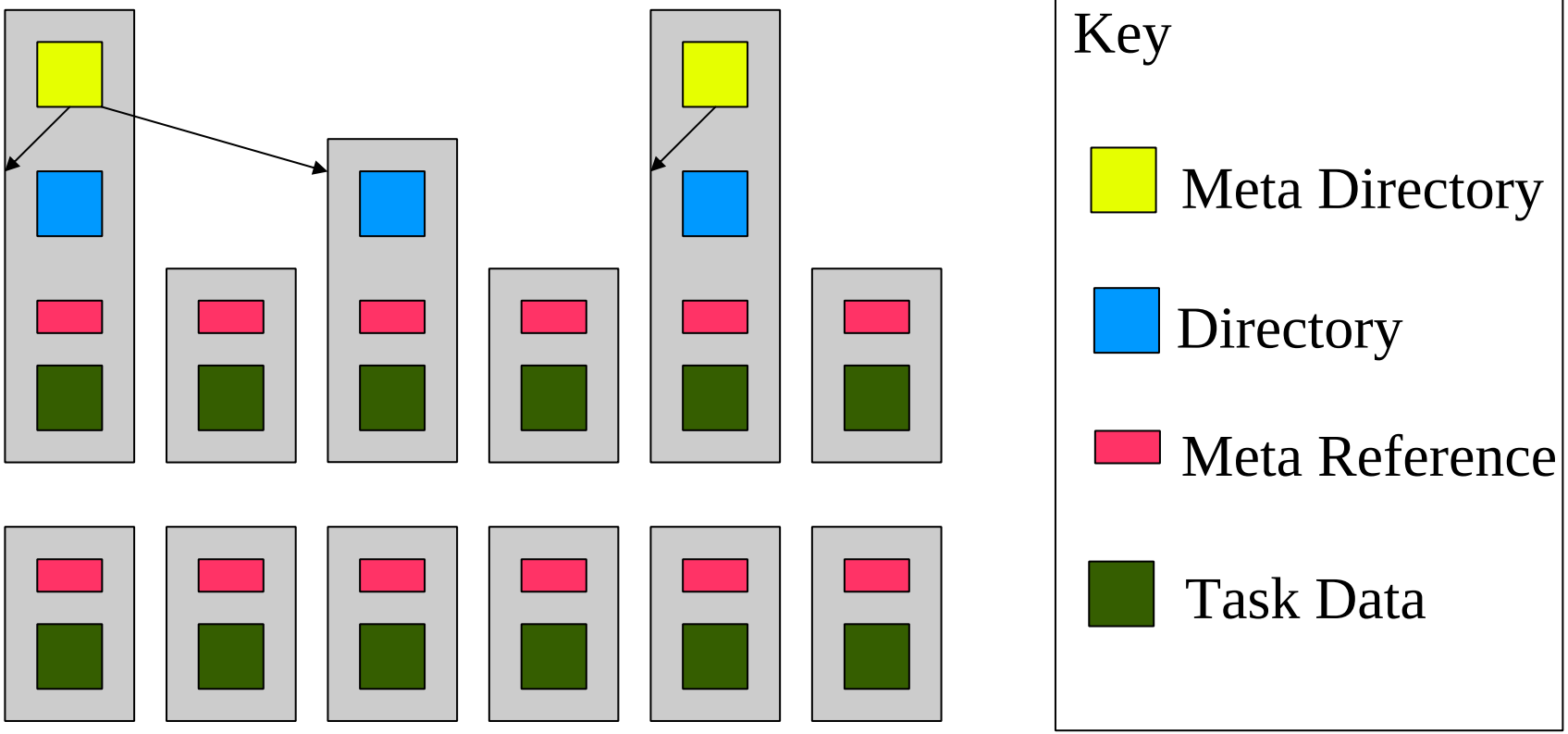
MPI 2 Process Acquisition

Finding Meta Directory Peers



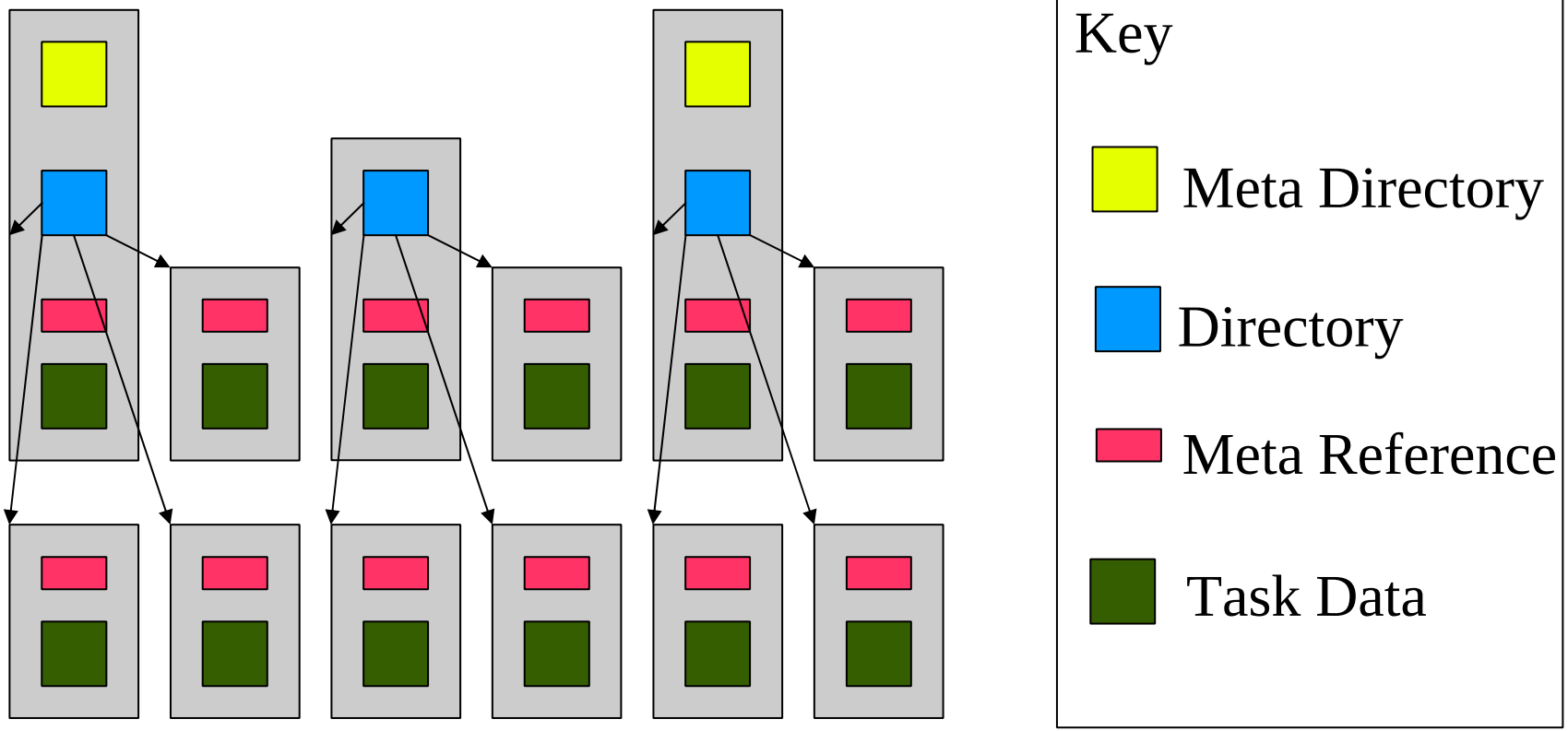
MPI 2 Process Acquisition

Finding Directory Processes



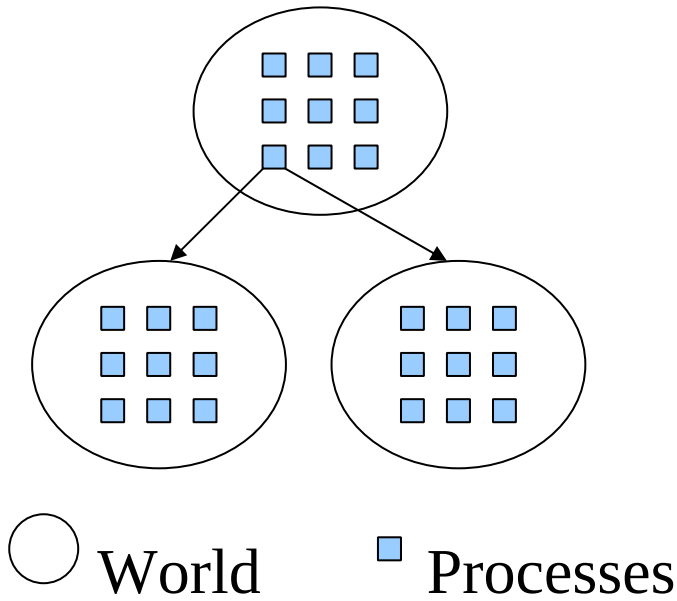
MPI 2 Process Acquisition

Finding the users MPI tasks



MPI 2 Process Naming

- **Clear**
  - Based on what happens in the MPI
  - Based on MPI_COMM_WORLD rank id
- **Repeatable**
  - As much as possible
- **Name**
  - World Communicator
    - parent communicator
    - rank of parent process
    - sequence
  - Process rank id



Notification and Synchronization

- **Breakpoint Function**
  - Used for notification and synchronization
  - Stub function
    - TotalView, if present, will set a breakpoint on this function
  - Called on meta directory process to notify the debugger that a change has occurred
    - Called once the data is updated
  - Called on spawning process to notify the debugger that new processes are being created
    - Gives the debugger the opportunity to attach to the newly created processes
- **Some status variables**

Organization

- **Context**
- **Challenge**
- **Solution**
- **Status**
  - Implementation Plan
  - Current Status

Implementation Plan

- **Collaborative Design**
  - Representatives from Open MPI, MPICH 2, and Etnus co-develop
- **Develop a Prototype**
  - Reference Implementation
  - Proof of concept
  - Work out kinks
- **Work with larger community**
  - Will other MPI vendors adopt this?
  - Is more formal validation desired and/or required?

Current Status

- **Working on reference implementation / prototype**
  - Mailing list
  - Design mostly done
  - Implementation in TV in progress
  - Implementation in MPICH 2 and Open MPI to start soon

The End

- **Thanks for listening!**
  - Stop by and see Etnus in the lobby
    - tell us about your site
    - let us know what you think about this paper
    - let us know what you think about TotalView
    - let us know if there are other tools you need
  - Contact me via email: chris.gottbrath@etnus.com
  - Tell your friends about TotalView
    - They can try it out for free for 15 days
      - www.etnus.com