Open MPI und ADCL

Kommunikationsbibliotheken für parallele, wissenschaftliche Anwendungen

Edgar Gabriel
Department of Computer Science
University of Houston
gabriel@cs.uh.edu

Is MPI dead?

New MPI libraries released in the last three years:

• LAM/MPI 7.x	re-implementation of LAM/MPI focusing on a component architecture
• MPICH2	all new version by ANL
• Open MPI	all new public domain implementation
• MVAPICH/2	(public domain) MPI libraries for InfiniBand interconnects

---

• Intel MPI	commercial cluster MPI library based on MPICH2
• HP-MPI	MPI library for clusters
• Voltaire MPI etc.	vendor specific derivatives of MPICH-1.2.x for InfiniBand

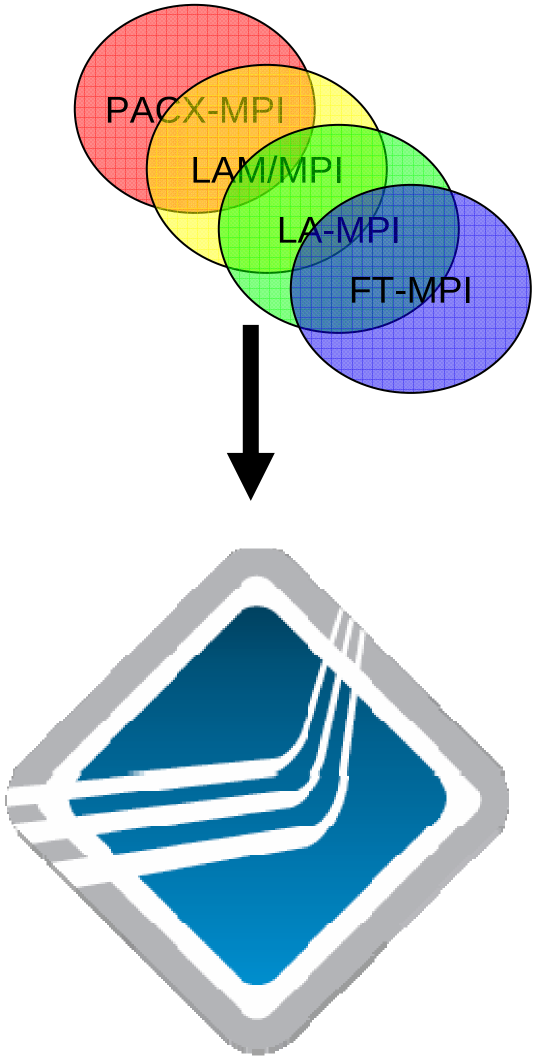
Open MPI Team

Los Alamos NATIONAL LABORATORY EST.1943

pervasivetechnologylabs AT INDIANA UNIVERSITY

ICL UT

H L R I S



CS@UH

CISCO SYSTEMS

VOLTAIRE The Grid Backbone

Mellanox TECHNOLOGIES

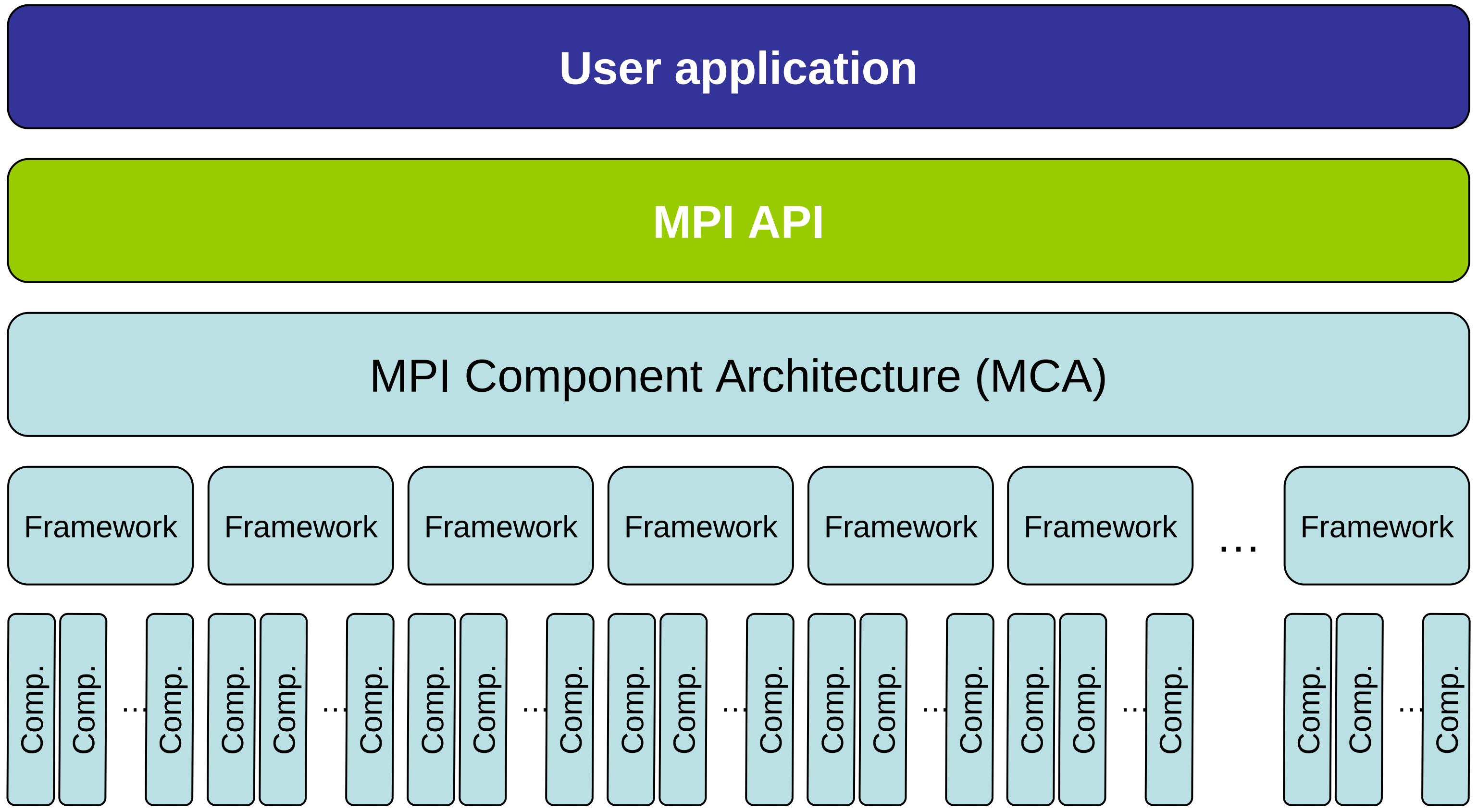
Sun microsystems

CS@UH

Goals

- All of MPI-2
- Thread safety (MPI_THREAD_MULTIPLE)
- Based on a component architecture
  - Flexible run-time tuning
  - "Plug-ins" for different capabilities (e.g., different networks)
- Optimized performance
  - Low latency and High bandwidth
  - Polling vs. asynchronous progres
- Production quality
- Open source
  - Vendor-friendly license (BSD)
  - Bring together "MPI-smart" developers
- Prevent "forking" problem
  - Community / 3rd party involvement

MPI Component Architecture (MCA)



MCA Component Frameworks

- Components divided into three categories
  - Back-end to MPI API functions
  - Run-time environment
  - Infrastructure / management
- Rule of thumb:
  - “If we’ll ever want more than one implementation, make it a component”

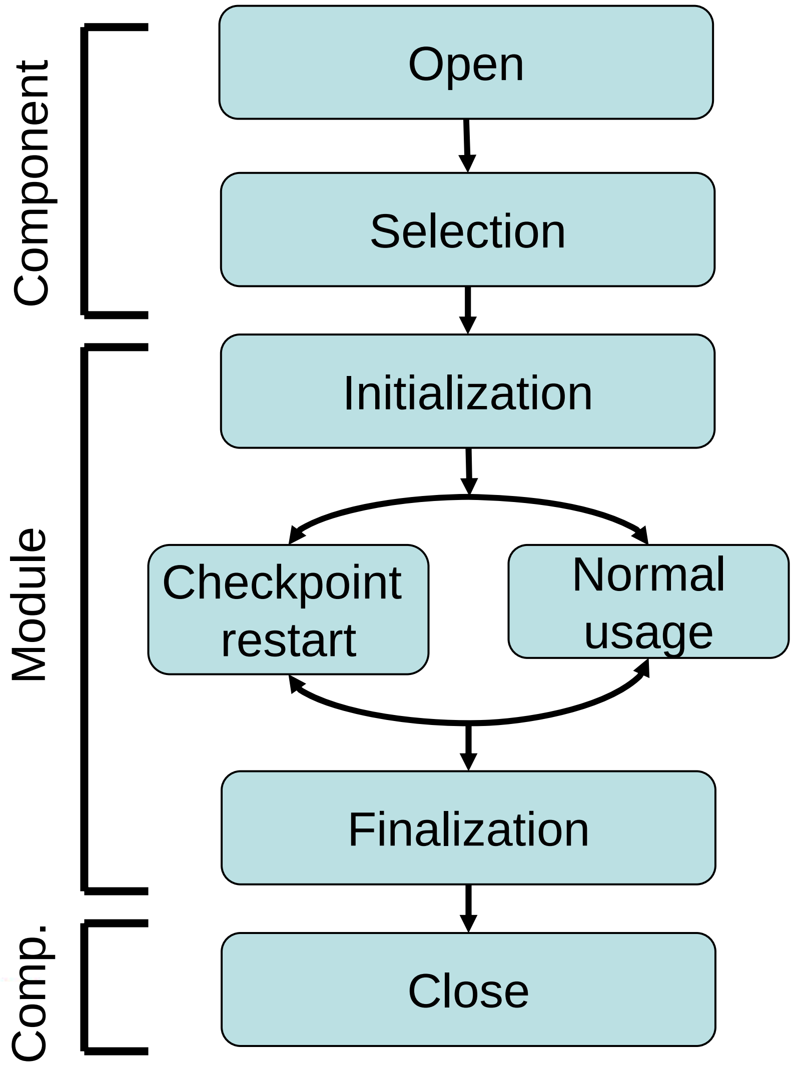
MCA Component Types

- MPI types
  - P2P management
  - P2P transport
  - Collectives
  - Topologies
  - MPI-2 one-sided
  - MPI-2 IO
  - Reduction Operations

- Run-time env. Types
  - Out of band communication
  - Process control
  - Global data registry

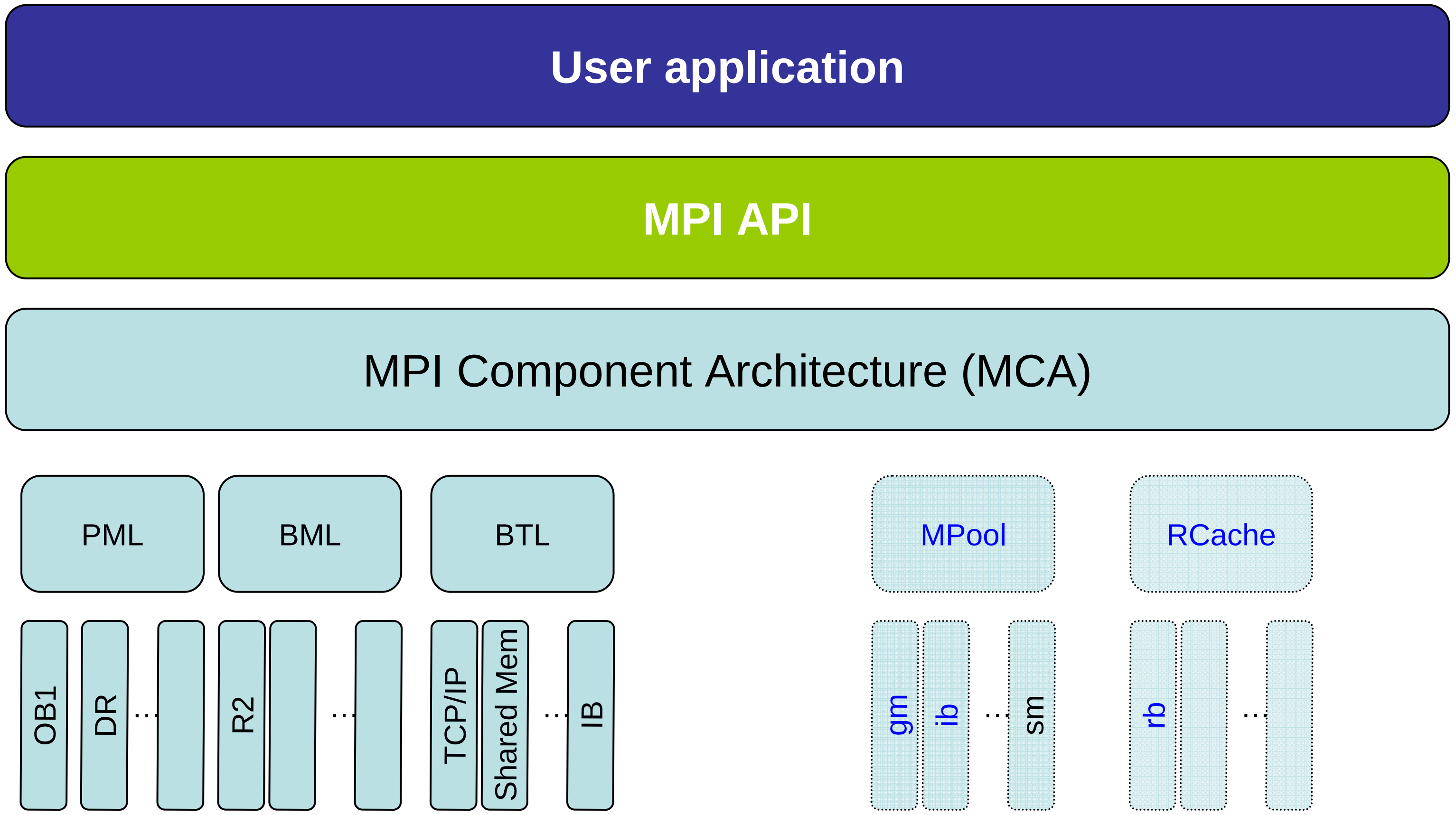
- Management types
  - Memory pooling
  - Memory caching
  - Common

Component / Module Lifecycle



- Component
  - Open: per-process initialization
  - Selection: per-scope determine if want to use
  - Close: per-process finalization
- Module
  - Initialization: if component selected
  - Normal usage / checkpoint
  - Finalization: per-scope cleanup

Point-to-Point



Pt-2-Pt Components

- **PML – P2P Management Layer**
  - Provides MPI Point-to-point semantics
  - Message Progression
  - Request Completion and Notification
- Internal MPI messaging protocols
  - Eager send
  - Rendezvous
- Support for various types of interconnect
  - Send/Recv
  - RDMA
  - Hybrids

- **BTL – Byte Transfer Layer**
  - Data mover
  - Message Matching
  - Responsible for own progress (polling or async)

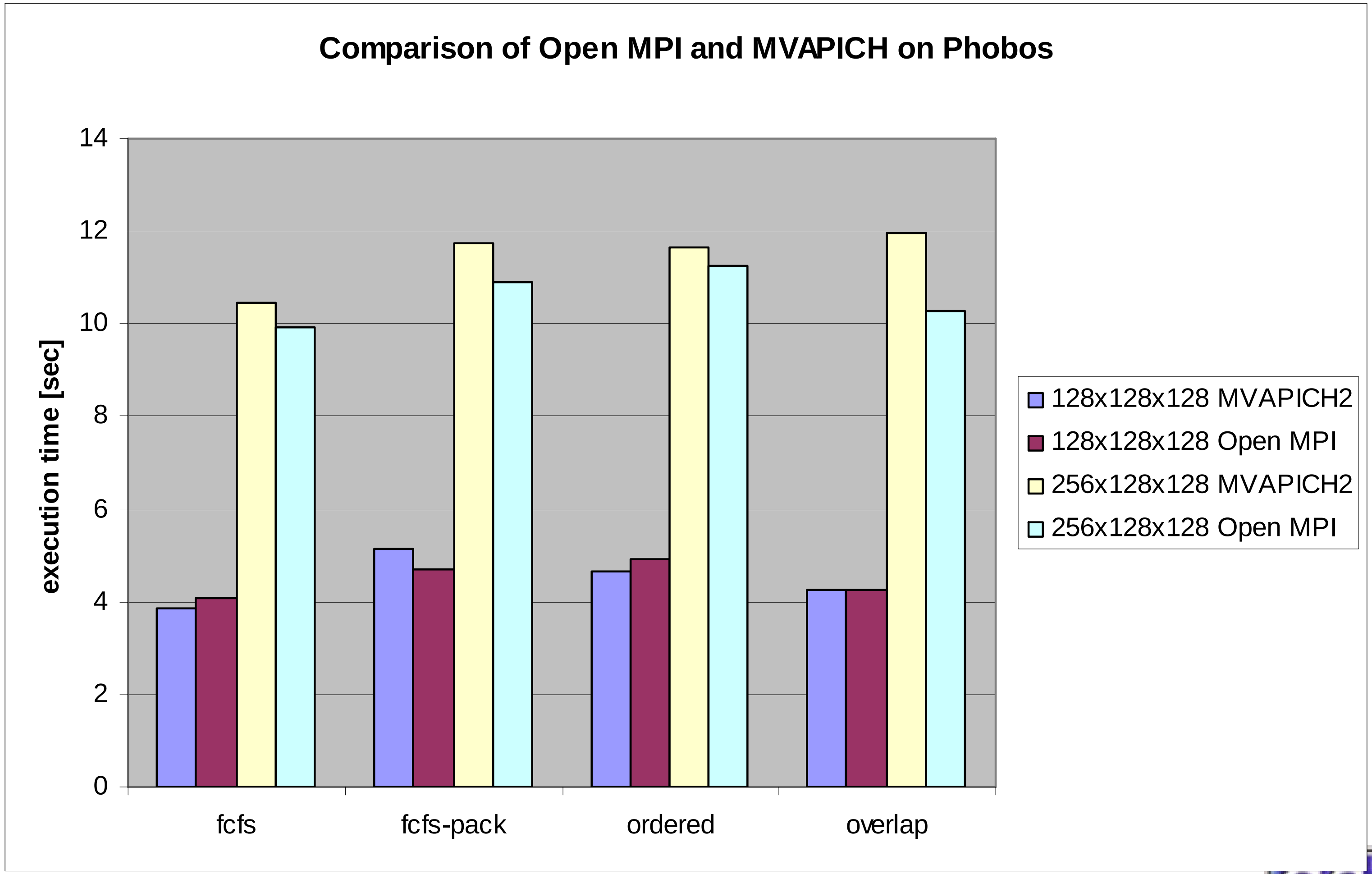
- **BML – BTL Management Layer**
  - Thin multiplexing layer over BTL's
  - Manages peer resource discovery

Performance Results

- 3-D Finite Difference with four different implementations of the occurring communication pattern evaluated:
  - **fcfs**: first-come first-serve using non-blocking communication and derived datatypes
  - **fcfs-pack**: first-come first-serve using non-blocking operations and pack/unpack
  - **overlap**: first-come first-serve using non-blocking operations, derived datatypes and overlapping communication and computation
  - **ordered**: using blocking Send/Recv operation with derived datatypes
- Tests executed on
  - cacau (HLRS): EM64T cluster using an InfiniBand and GEthernet
  - phobos (ZIH): Opteron cluster using InfiniBand
- Three different MPI libraries tested:
  - Open MPI v1.0.1
  - Intel MPI 1.0
  - MVAPICH 1.2.

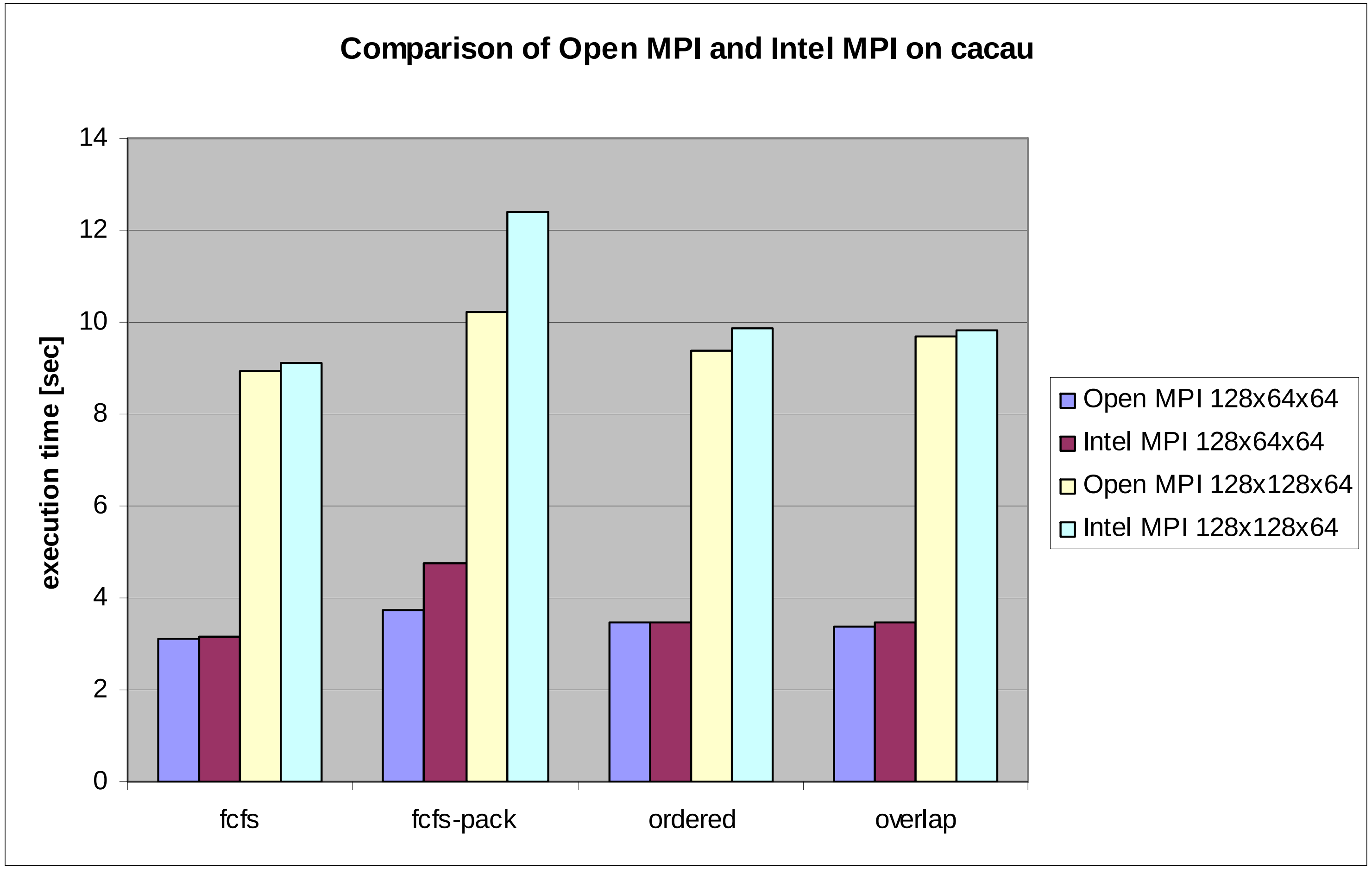
Performance Results (I)

Execution time for 200 iterations on 64 processes/ 32 nodes on phobos



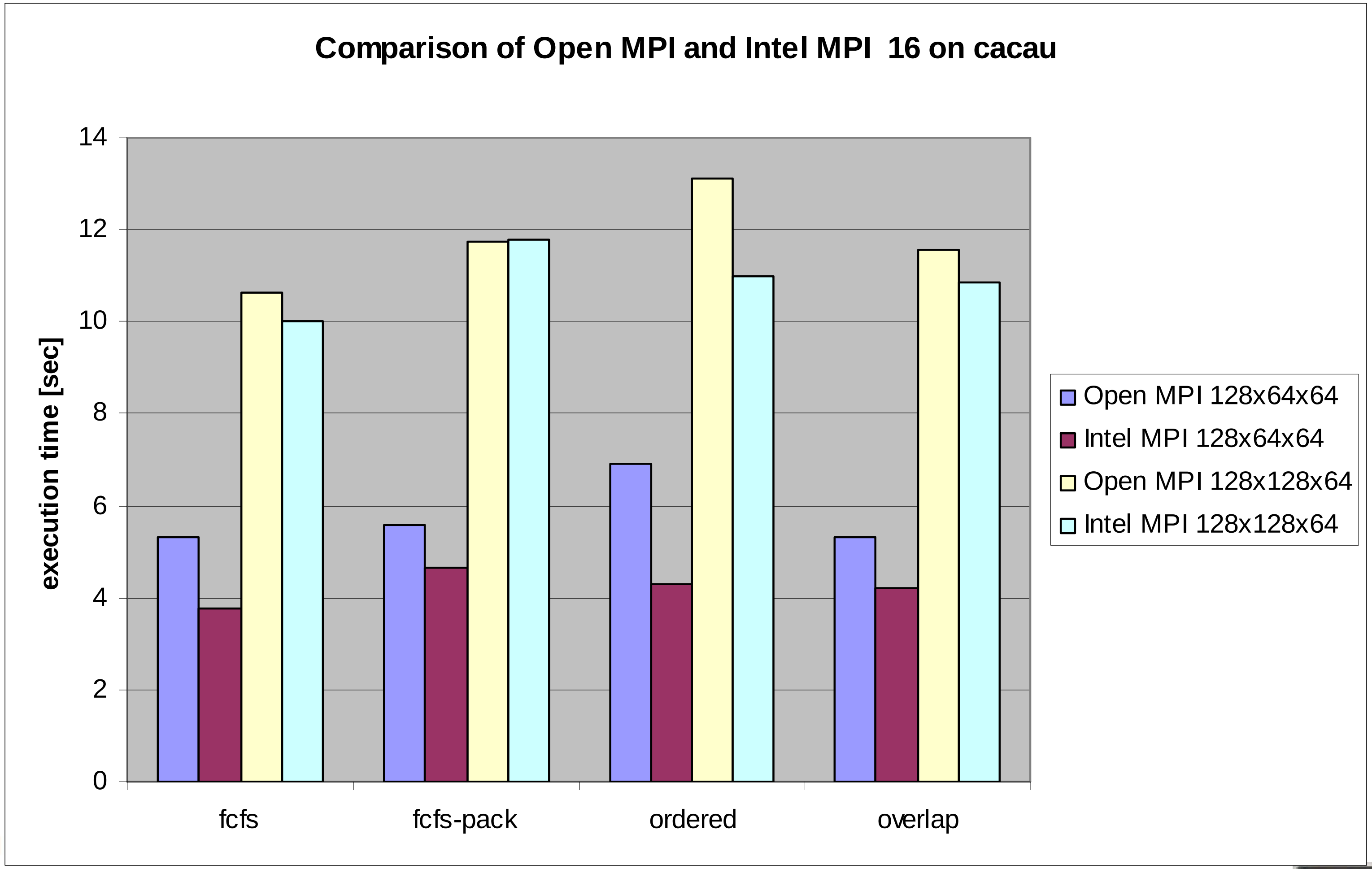
Performance Results (II)

Execution time for 200 iterations on 16 processors / 16 nodes over IB



Performance Results (III)

Execution time for 200 iterations on 16 processors / 16 nodes over GE



Edgar Gabriel

Current status (I)

- Current stable release: v1.0.2
- Last week branched for v1.1
  - Expected to be released May/June/July
  - Tuned collective communication component
  - One-sided communication component
  - Data reliability

- Supported operating systems
  - Linux
  - OS X (BSD)
  - Solaris *
  - AIX *

* Less frequently tested

Current status (II)

- Supported network interconnects
  - TCP
  - Shared memory
  - Myrinet
    - GM, MX
  - Infiniband
    - mVAPI, OpenIB
  - Portals

- Supported batch schedulers
  - rsh / ssh
  - BProc (current)
  - PBS / Torque
  - SLURM
  - BJS (LANL BProc Clustermatic)
  - Yod (Red Storm)

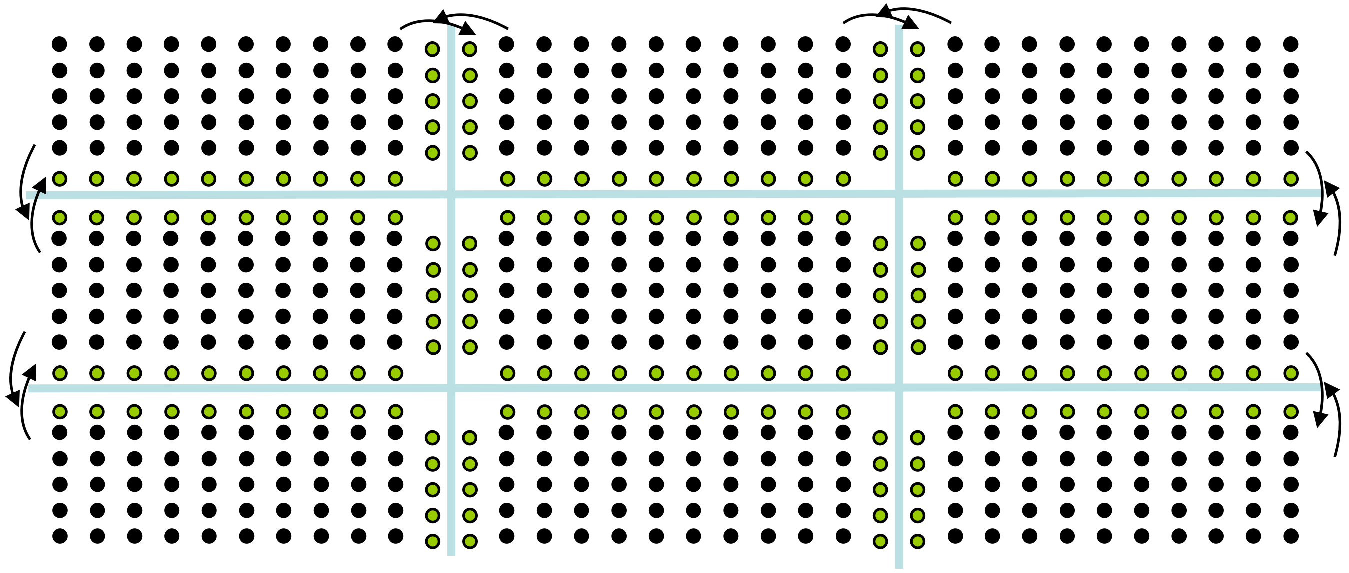
Currently ongoing work

- Data reliability for point-to-point operations
- Definition of new collective framework collv2
  - Selection on a per-function bases (instead of per component basis in v1)
- Coordinated checkpoint-restart capabilities

- ORTE v2
  - Relevant for dynamic process management

ADCL - Motivation (I)

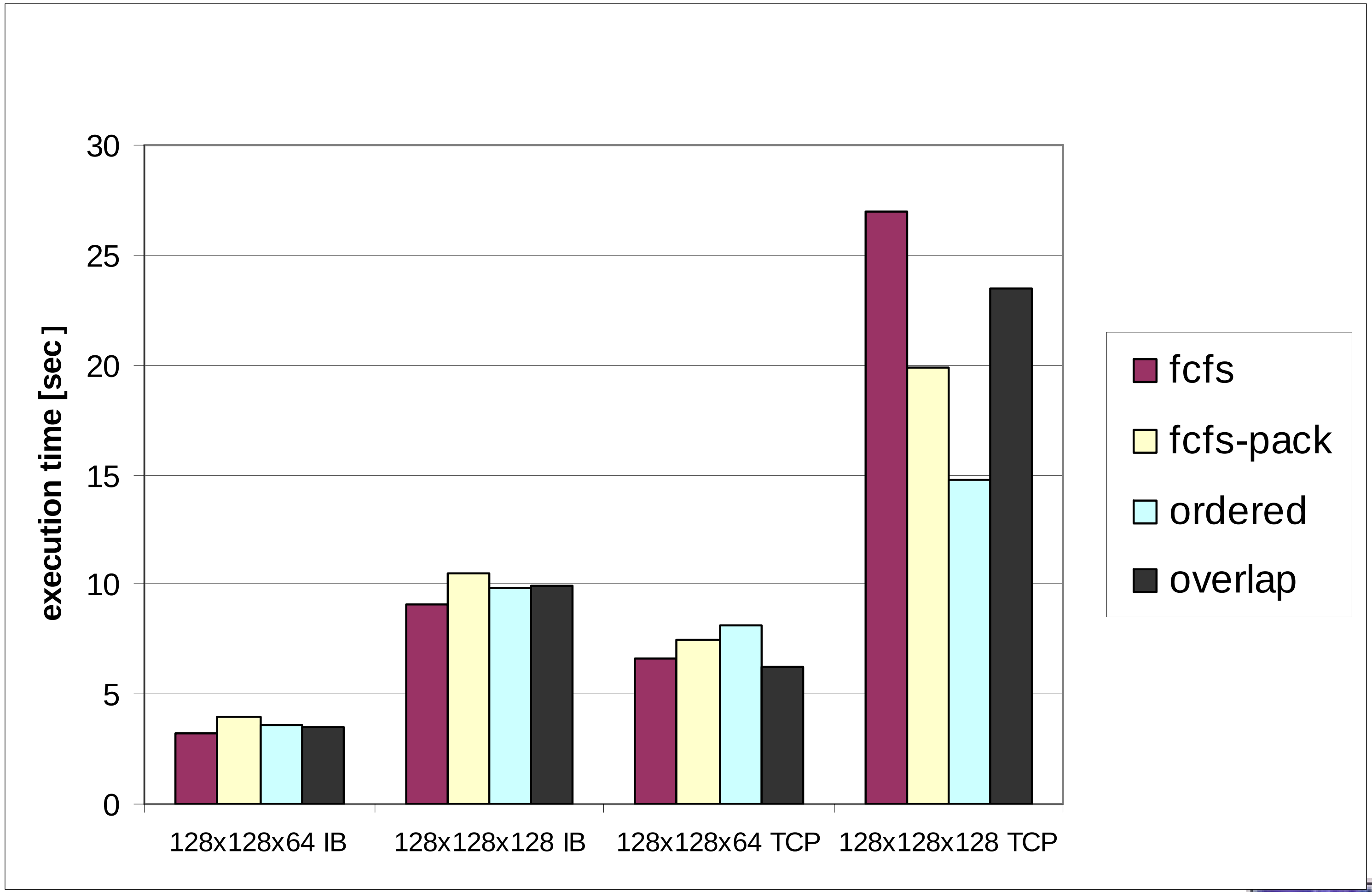
- Finite difference code using regular domain decomposition
  - Data exchange at process boundaries required in every iteration of the solver
  - Typically implemented by a sequence of point-to-point operations



Edgar Gabriel

Motivation (III)

Execution time for 200 iterations on 32 processes/processors



How to implement the required communication pattern?

- Dependence on platform
  - Some functionality only supported (efficiently) on certain/platforms or with certain network interconnects
- Dependence on MPI library
  - Does the MPI library support all available methods
  - Efficiency in overlapping communication and computation
  - Quality of the support for user defined datatypes
- Dependence on application
  - Problem size
  - Ratio of communication to computation

- **Problem**: How can an (average) user understand the myriad of implementation options and their impact on the performance of the application?
- **(Honest) Answer**: no way
  - Abstract interfaces for application level communication operations required $\longrightarrow$ ADCL
  - Statistical tools required to detect correlations between parameters and application performance

ADCL - Adaptive Data and Communication Library

- Goals:
  - Provide abstract interfaces for often occurring application level communication patterns
    - Collective operations
    - Not-covered by MPI specification
  - Provide a wide variety of implementation possibilities and decision routines which choose the fastest available implementation (at runtime)
- Not replacing MPI, but add-on functionality
  - Uses many features of MPI

ADCL – components (I)

1. Static (parallel) configure step
   - Exclude methods not supported by the MPI library
   - Determine characteristics of the MPI library, e.g.
     - `MPI_Send` vs. `MPI_Isend` vs. `MPI_Put` vs. `MPI_Get`
     - Effect of `MPI_Alloc_mem`
     - Derived Datatypes vs. `MPI_Pack/MPI_Unpack`
     - Efficiency of overlapping communication and computation
     - …
   - Characteristics stored as attributes of the library

ADCL – components (II)

2. ADCL Methods and Runtime library
   - Collection of all available implementations for a certain communication operation
   - Runtime decision routines
     - Matching of requirements of an implementation to the attributes set by the parallel configure step
     - Testing at runtime
   - Monitoring of the performance
     - Used for initiating re-evaluation of a decision
3. Historic learning
   - Input file
   - Usage of performance skeletons (cooperation with Jaspal Subhlok)

Classification of implementations

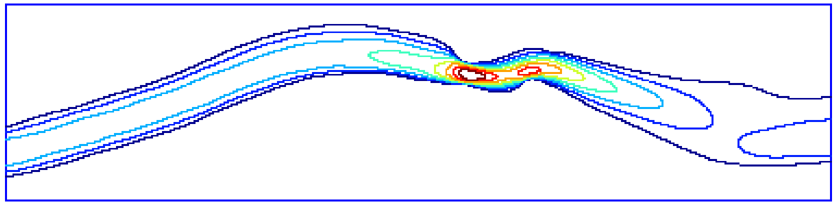
1. Data transfer primitives
   - Blocking point-to-point operations
   - Non-blocking point-to-point operations
   - Persistent request operations
   - One-sided operations
   - Collective operations
2. Mapping of the communication pattern to data transfer operations
   - Direct-transfer vs. Variable-transfer
   - Single-block vs. dual-block implementations
3. Handling of non-contiguous messages
   - Sending each element separately
   - Pack/unpack
   - Derived datatypes

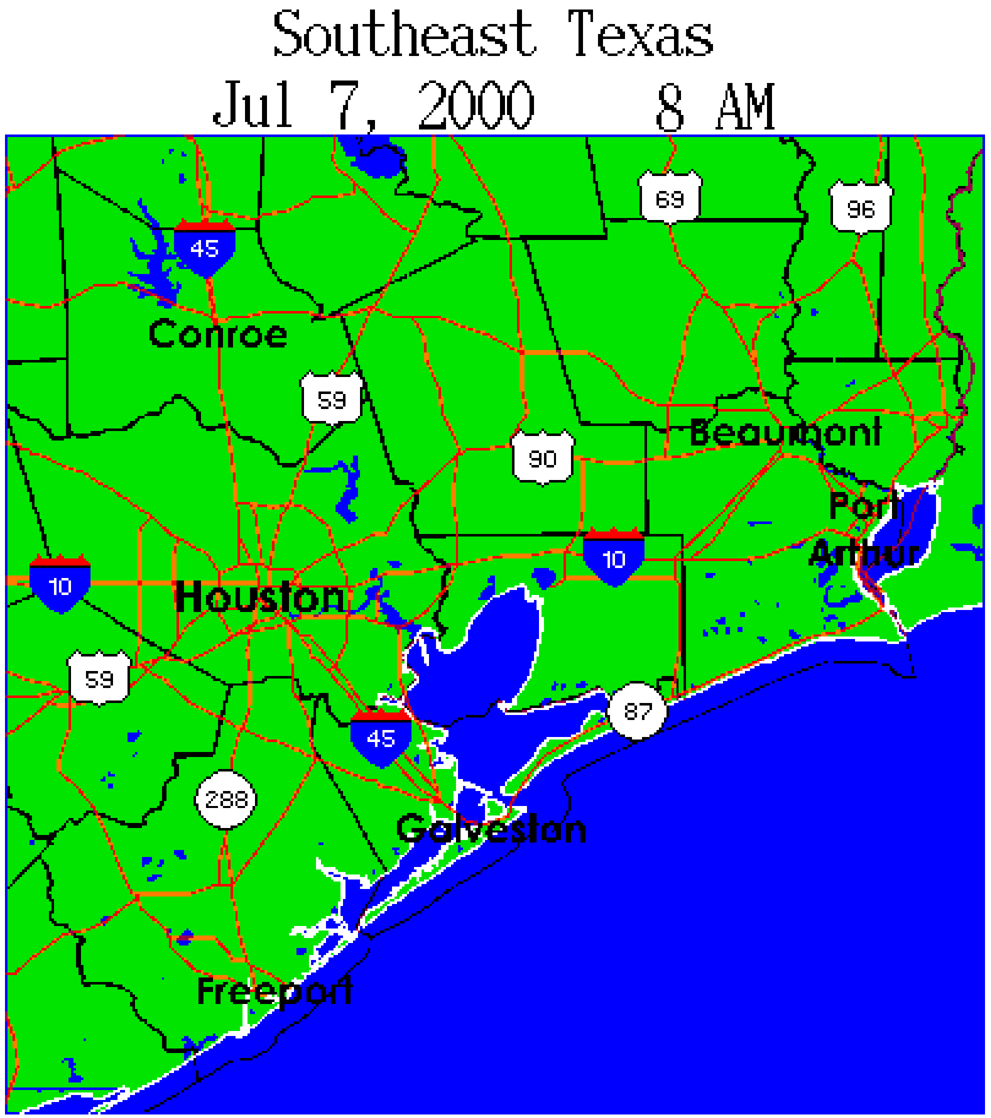
ADCL – code sample

```
/* describe neighborhood relations using Topology
   functions of MPI */
MPI_Cart_create ( comm, n, dims[n], period, reorder,
                  &cart_comm);

/* Register a data structure for communication operations*/
ADCL_Register_dense_matrix ( matrix, n, matrix_dims[n],
                             k, submatrix_dims[k],
                             num_ghostcells, distance,
                             cart_comm, &adcl_request);


…
/* Start a blocking communication for the registered matrix
   on the provided communicator */
ADCL_Start ( &adcl_request );
…
```

Current status of ADCL

- Application driven
  - CMAQ: air-quality code (Daewon Byun)
  - Multi-scale blood-flow simulation (Marc Garbey)





Available implementations for 3-D neighborhood communication

	Data transfer primitive	Communication structure	Handling of non-cont. messages
ordered	Blocking	Direct-transfer, single-block	Der. datatypes
fcfs	Non-blocking	Direct-transfer, single-block	Der. datatypes
fcfs-pack	Non-blocking	Direct-transfer, single-block	Pack/Unpack
overlap	Non-blocking	Direct-transfer, dual-block	Der. datatypes
alltoallw	Collective	Direct-transfer, single-block	Der. datatypes
get-fence	One-sided	Direct-transfer, dual-block	Der. datatypes
put-fence	One-sided	Direct-transfer, dual-block	Der. datatypes
get-start	One-sided	Direct-transfer, dual-block	Der. datatypes
put-start	One-sided	Direct-transfer, dual-block	Der. datatypes
topo	Non-blocking	Variable-transfer, single-block	Der. datatypes
topo-overlap	Non-blocking	Variable-transfer, dual-block	Der. datatypes

Summary

- Open MPI
  - a component based, flexible implementation of the MPI-1 and MPI-2 specifications
  - Resolves some of the issues seen on today’s cluster with other MPI libraries
- ADCL:
  - An adaptive communication library for abstracting often occurring application level communication operations
  - Simplifies the development of portable and performant code for scientific computing