Screencast: Basic Architecture and Tuning

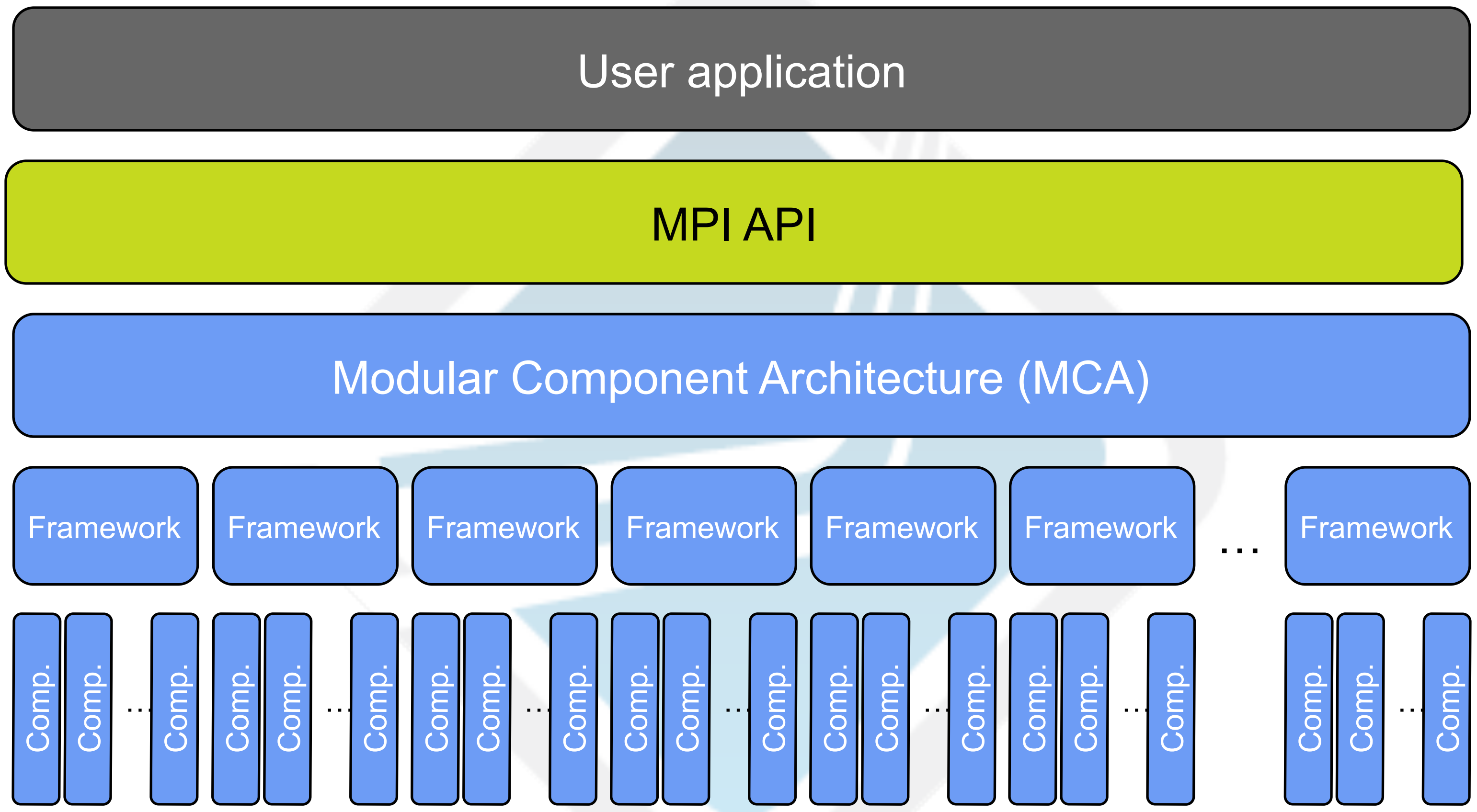
Jeff Squyres

May 2008

Open MPI Architecture

- Modular component architecture (MCA)
  - Backbone plugin / component system
  - Finds, loads, parameterizes components
- Hierarchy
  - MCA: foundation
  - Framework: functionality specification
  - Component: code for specific functionality
  - Module: “instance” of a component

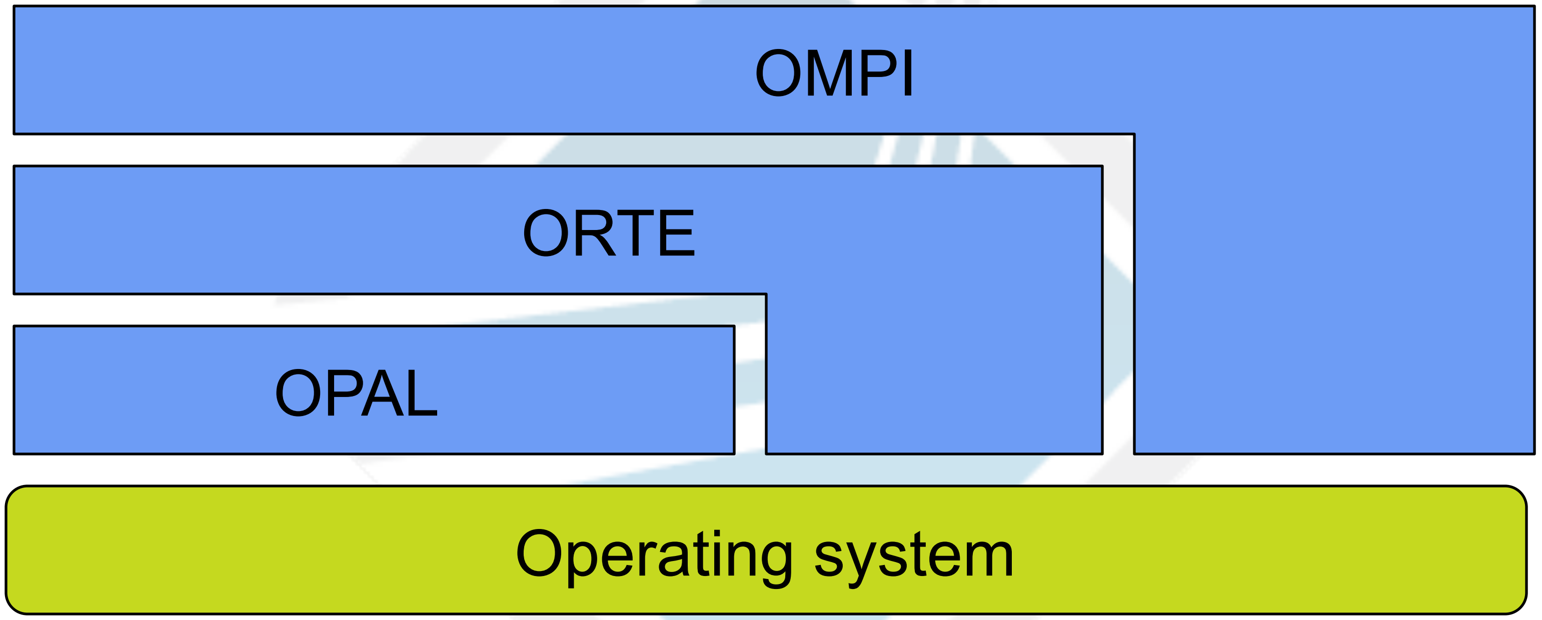
Open MPI Architecture



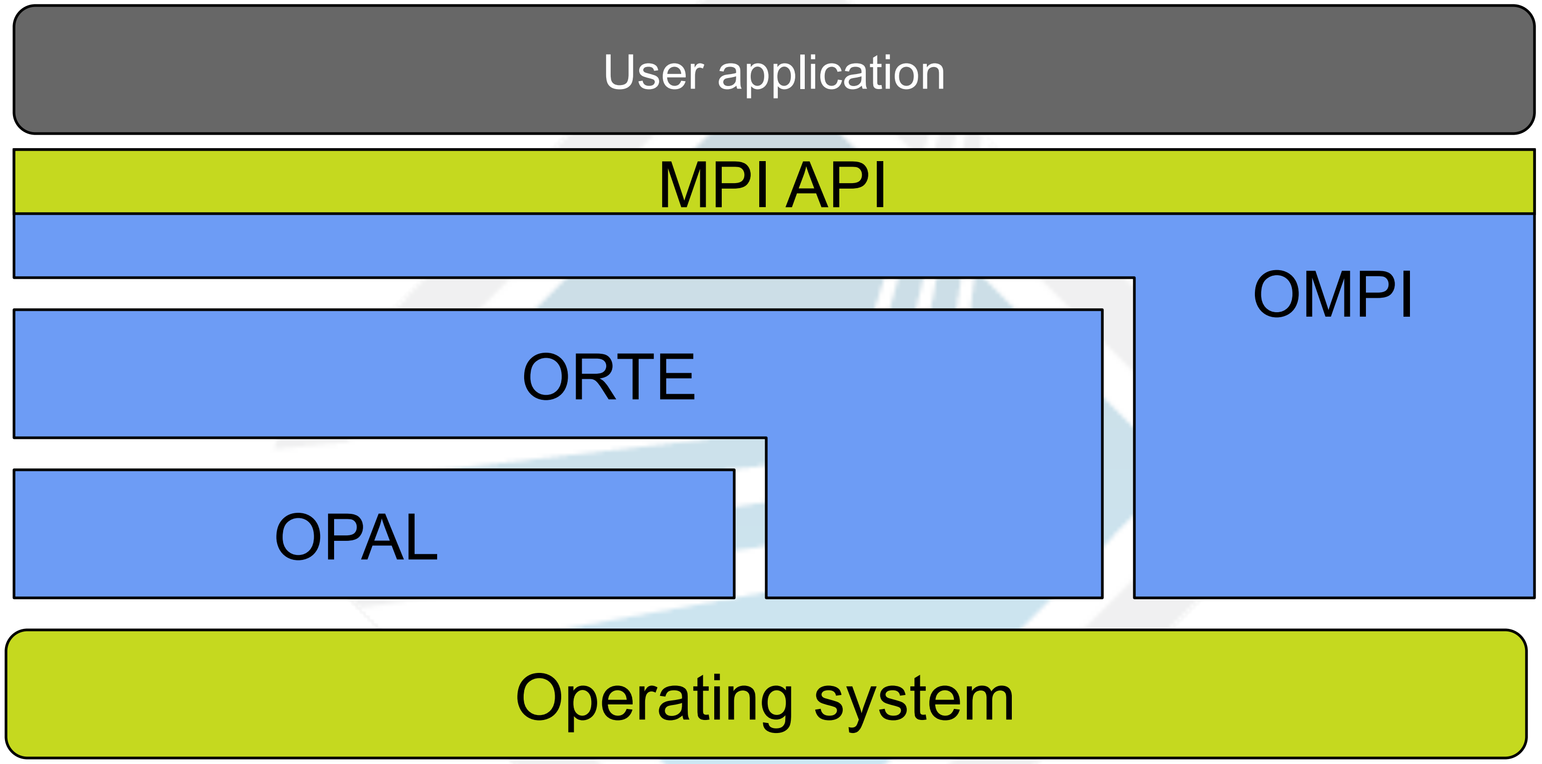
Three Main Code Sections

- Open MPI layer (OMPI)
  - Top-level MPI API and supporting logic
- Open Run-Time Environment (ORTE)
  - Interface to back-end run-time system
- Open Portability Access Layer (OPAL)
  - OS / utility code (lists, reference counting, etc.)
- Dependencies - not layers
  - OMPI → ORTE → OPAL

Three Main Code Sections



Three Main Code Sections



Tuning

MCA Parameters

- Run-time tunable values
  - Per layer
  - Per framework
  - Per component
- Change behaviors of code at run-time
  - Does *not* require recompiling / re-linking
- Simple example
  - Choose which network to use for MPI communications

MCA Parameter Lookup Order

1. mpirun command line

```
mpirun --mca <name> <value>
```

2. Environment variable

```
export OMPI_MCA_<name>=<value>
```

3. File
   - $HOME/.openmpi/mca-params.conf
   - $prefix/etc/openmpi-mca-params.conf

   (these locations are themselves tunable)

4. Default value

So Much Information…

- Open MPI has:
  - ~30 frameworks
  - 100+ components
  - Each component has run-time tunable parameters

- How to know what to use / how to use it?

ompi_info Command

- Tells everything about OMPI installation
  - Finds all components and all params
  - Great for debugging
- Can look up specific component

```
ompi_info --param <type> <plugin>
```

  - Shows parameters and current values
  - Can also use keyword “all”
- “--parsable” option

Example: Specify BTL

- BTL: Byte Transfer Layer
  - Framework for MPI point-to-point communications
  - Select which network to use for MPI communications

```
mpirun --mca btl tcp,self \
    -np 4 ring_c
```

- Framework-level MCA parameter
  - Specifies which components to load

Example: Specify TCP BTL

```
mpirun --mca btl tcp,self -np 4 ring_c
```

- Components
  - tcp: TCP sockets
  - self: Process loopback (send-to-self)

Example: Specify openib BTL

```
mpirun --mca btl openib,self -np 4
 ring_c
```

- Components
  - openib: OpenFabrics verbs
  - self: Process loopback (send-to-self)

Example: Specify sm+openib BTLs

```
mpirun --mca btl sm,openib,self -np 4
 ring_c
```

- Components
  - openib: OpenFabrics verbs
  - self: Process loopback (send-to-self)
  - sm: Shared memory (on-host communication)

What Does This Do?

```
mpirun -np 4 ring_c
```

What Does This Do?

```
mpirun -np 4 ring_c
```

- Use all available components
  - tcp, sm, openib, …
- TCP too?
  - Yes -- and no
  - TCP will automatically disable itself in the presence of low latency components (e.g., openib)

What Does This Do?

```
mpirun -np 4 ring_c
```

- More specifically:
  - Open each BTL component
  - Query if it wants to be used
  - Keep all that say “yes”
  - Rank by bandwidth and latency rating

What Does This Do?

```
mpirun -np 4 --mca btl ^tcp ring_c
```

What Does This Do?

```
mpirun -np 4 --mca btl ^tcp ring_c
```

- Use all available components except tcp
- More specifically:
  - Open each BTL component except tcp
  - Query if it wants to be used
  - Keep all that say “yes”
  - Rank by bandwidth and latency rating

openib BTL Parameters

```
ompi_info --param btl openib
```

- Shows all openib BTL MCA parameters
  - …there are a lot!
- Also try:

```
ompi_info --param btl openib \
    --parsable
```

- What do they all mean?

CISCO