MPI Service Update

Benjamin A. Allan[1] Kosta Damevski[2]

[1]Sandia National Laboratories, Livermore, CA, USA, baallan@ca.sandia.gov
[2]Virginia State University, Petersburg, VA, USA, kdamevski@vsu.edu

CCA Forum Fall 2008

Outline

Case I: Simple SPMD

- 80% solution.
- Application includes all SPMD (SCMD) components on all processors.
  - Each component gets its own communicator, same size as frame.
  - No message tag conflicts.
- Any component using port type "gov.cca.ports.MPIService" autoconnected if a framework user so chooses (via ServiceRegistry configuration).
- From ccaffeine.MPIService, basically unchanged. SIDL later.

Case I: Simple SPMD

- 80% solution.
- Application includes all SPMD (SCMD) components on all processors.
  - Each component gets its own communicator, same size as frame.
  - No message tag conflicts.
- Any component using port type "gov.cca.ports.MPIService" autoconnected if a framework user so chooses (via ServiceRegistry configuration).
- From ccaffeine.MPIService, basically unchanged. SIDL later.

Case I: Simple SPMD

- 80% solution.
- Application includes all SPMD (SCMD) components on all processors.
  - Each component gets its own communicator, same size as frame.
  - No message tag conflicts.
- Any component using port type "gov.cca.ports.MPIService" autoconnected if a framework user so chooses (via ServiceRegistry configuration).
- From ccaffeine.MPIService, basically unchanged. SIDL later.

Case II: Shared communicator SPMD

- A 5-10% solution.
- Application includes all SPMD (SCMD) components on all processors.
  - Components can choose to share a single communicator.
  - Some implementations may have a hard time cloning a communicator at scale.
  - Each component can get an independent set of message tag numbers.
  - Encoding extra information in tag values not supported.
- Any component using port type "gov.cca.ports.MPIBorrow" autoconnected if a framework user so chooses (via ServiceRegistry configuration).
- From ccaffeine.MPIBorrow, basically unchanged. SIDL later.

Case II: Shared communicator SPMD

- A 5-10% solution.
- Application includes all SPMD (SCMD) components on all processors.
  - Components can choose to share a single communicator.
  - Some implementations may have a hard time cloning a communicator at scale.
  - Each component can get an independent set of message tag numbers.
  - Encoding extra information in tag values not supported.
- Any component using port type "gov.cca.ports.MPIBorrow" autoconnected if a framework user so chooses (via ServiceRegistry configuration).
- From ccaffeine.MPIBorrow, basically unchanged. SIDL later.

Case II: Shared communicator SPMD

- A 5-10% solution.
- Application includes all SPMD (SCMD) components on all processors.
    - Components can choose to share a single communicator.
    - Some implementations may have a hard time cloning a communicator at scale.
    - Each component can get an independent set of message tag numbers.
    - Encoding extra information in tag values not supported.
- Any component using port type "gov.cca.ports.MPIBorrow" autoconnected if a framework user so chooses (via ServiceRegistry configuration).
- From ccaffeine.MPIBorrow, basically unchanged. SIDL later.

Case III: MCMD Single Frame

- 5% solution.
- Application includes one frame on all processors.
  - Some component G gets and splits its own communicator from frame-scoped MPIService.
  - G sets up via BuilderService an MPIService component for each sub-communicator.
  - G sets up via BuilderService app components for each sub-communicator.
  - These other components are not auto-connected.
- Anyone smart enough to split a communicator is expected to be smart enough to manage port connections to separate MPIService provider instances.

Case III: MCMD Single Frame

- 5% solution.
- Application includes one frame on all processors.
  - Some component G gets and splits its own communicator from frame-scoped MPIService.
  - G sets up via BuilderService an MPIService component for each sub-communicator.
  - G sets up via BuilderService app components for each sub-communicator.
  - These other components are not auto-connected.
- Anyone smart enough to split a communicator is expected to be smart enough to manage port connections to separate MPIService provider instances.

Case IV: MCMD Multi-Frame

- 5% solution.
- Application creates separate framework instances.
    - Application driver configures each frame-scoped MPIService.
    - Components within each frame behave (autoconnect) as SPMD case I.

Tabled extension 1

- 80% solution.
- Component provides a "setCommunicator" interface (MPIConsumer)
- Why it is tabled
  - Typically communicator this is part of a more complicated, nonstandardizable initialization.
  - It is extremely trivial to define and implement yourself.

Tabled extension 1

- 80% solution.
- Component provides a "setCommunicator" interface (MPIConsumer)
- Why it is tabled
  - Typically communicator this is part of a more complicated, nonstandardizable initialization.
  - It is extremely trivial to define and implement yourself.

Tabled extension 2

- 80% solution.
- Component provides a "getCommunicator by string name" interface
- Why it is tabled
  - No specifically needed prior art.
  - It is extremely specific case of a generalized object Registry, not unlike a treating a MPI communicator as a component.

Tabled extension 2

- 80% solution.
- Component provides a "getCommunicator by string name" interface
- Why it is tabled
  - No specifically needed prior art.
  - It is extremely specific case of a generalized object Registry, not unlike a treating a MPI communicator as a component.

Tabled extension 2

```
// just the method signatures in PPT
interface MPIService extends gov.cca.Port {

  long getComm() throws gov.cca.CCAException;

  void releaseComm(in long comm) throws gov.cca.CC

}
```

MPIBorrow

```
interface MPIBorrow extends gov.cca.Port
{

  long borrowComm(in int tagsRequested,
                  inout array<int> tagList,
                  inout int key) throws gov.cca.CC

  void returnComm(in long comm,
                  in int tagsRequested,
                  inout array<int> tagList,
                  in int key) throws gov.cca.CCAEx
}
```

MPIServer (configuration interface for implementations)

```
interface MPIServer extends gov.cca.Port {

    bool isInitialized();

    // Set up a service to a frame.
    // Useful in at least the static linking case.
    void initAsService(in long borrowComm,
                       in long dupComm,
                       inout gov.cca.AbstractFramework

    // Useful, e.g after a comm split.
    void initComponent(in long borrowComm,
                       in long dupComm);

    void finalize(in bool reclaim) throws gov.cca.CCAE
}
```

Amendments: ServiceRegistry semantics

- Components need to be able to opt-out of auto-connect of services.
  - Currently not possible.
  - Easy to support statically (code-time) by a port property IGNORE_SERVICEREGISTRY.
  - Support dynamically (override hardcoding) by a component property "port-configure:%PORTNAME:%PROPERTYNAME:%PROPERTYVALUE"
- Changes required:
  - Add comments and framework-side behavior to gov.cca.Services.
  - Add comments and framework-side behavior to BuilderService port.

Amendments: ServiceRegistry semantics

- Components need to be able to opt-out of auto-connect of services.
  - Currently not possible.
  - Easy to support statically (code-time) by a port property IGNORE_SERVICEREGISTRY.
  - Support dynamically (override hardcoding) by a component property "port-configure:%PORTNAME:%PROPERTYNAME:%PROPERTYVALUE"
- Changes required:
  - Add comments and framework-side behavior to gov.cca.Services.
  - Add comments and framework-side behavior to BuilderService port.

Implementation

- ◇ Target is a Ccaffeine-free C reference component.
  - ○ To run in any MPI-tolerant framework.
- ◇ Still in progress.

Implementation

- Target is a Ccaffeine-free C reference component.
  - To run in any MPI-tolerant framework.
- Still in progress.

Feedback

- ◇ Anybody with an opinion they want heard needs to speak now or join the MPIService working group.
- ◇ Should we drop MPIBorrow from the standard process?
- ◇ Should we add one of the tabled issues to the standard process?
- ◇ Current draft is in cca-spec-babel/mpi/gov.cca.ports.mpi.sidl.

Feedback

- Anybody with an opinion they want heard needs to speak now or join the MPIService working group.

- Should we drop MPIBorrow from the standard process?
- Should we add one of the tabled issues to the standard process?

- Current draft is in cca-spec-babel/mpi/gov.cca.ports.mpi.sidl.

Feedback

- ◇ Anybody with an opinion they want heard needs to speak now or join the MPIService working group.
- ◇ Should we drop MPIBorrow from the standard process?
- ◇ Should we add one of the tabled issues to the standard process?
- ◇ Current draft is in cca-spec-babel/mpi/gov.cca.ports.mpi.sidl.

Feedback

- Anybody with an opinion they want heard needs to speak now or join the MPIService working group.
- Should we drop MPIBorrow from the standard process?
- Should we add one of the tabled issues to the standard process?
- Current draft is in cca-spec-babel/mpi/gov.cca.ports.mpi.sidl.