Implementing Endogenous and Exogenous Connectors with the Common Component Architecture

Gisele Freitas, Francisco H. Carvalho Junior and Ricardo C. Corrêa

CBHPC09 Nov. 15-16, Portland, OR, USA
Universidade Federal do Ceará
Fortaleza-CE, Brasil

November 2009

Summary

Introduction

- Components and connections are the basis of the CCA model;
- But the way which ports are connected to each other is left unspecified;
- CCA is seen as a low-level component model in a including models that support sophisticated connections.

Introduction - Our suggestion

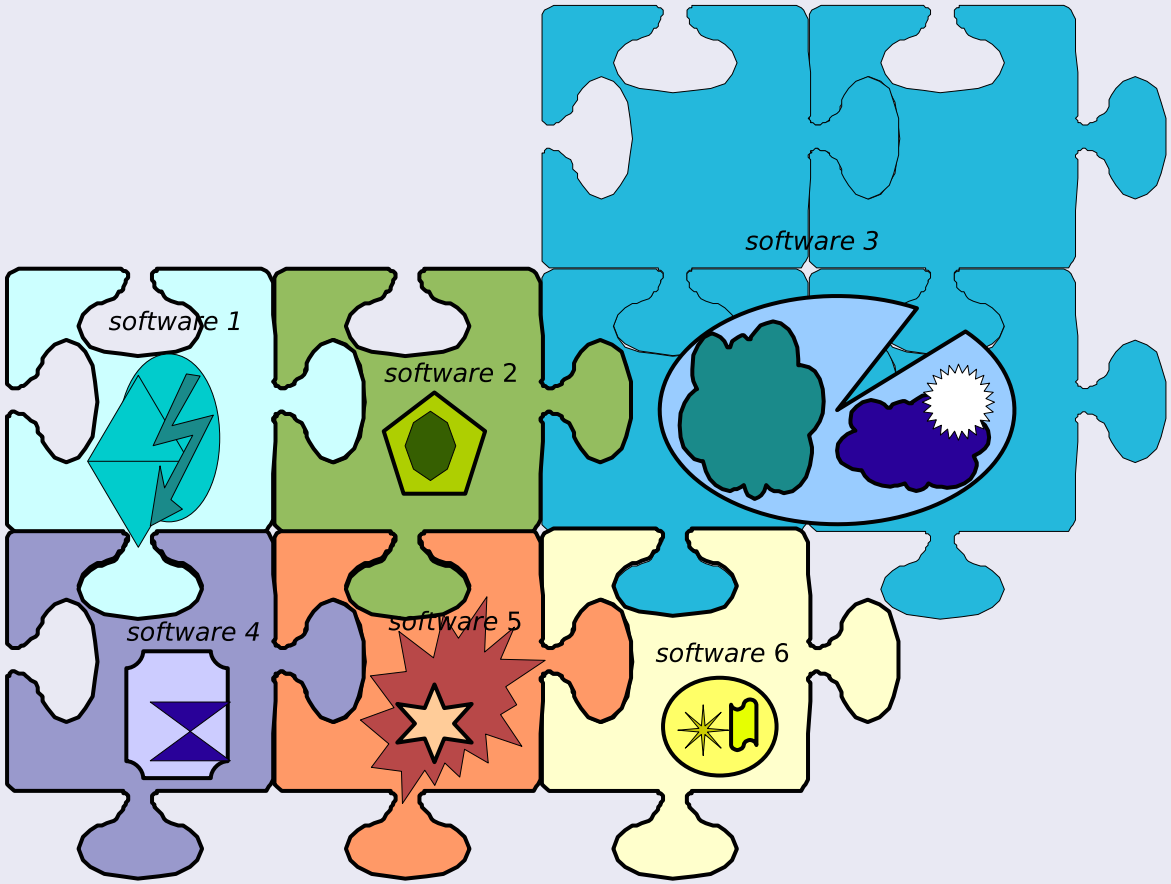
- Endogenous and exogenous connectors:
  - Goal: encapsulate interaction among components at high-level of the component models;
  - Implementations: CCA components;
  - Use: some high performance computing applications.

Introduction

Question

- How to connect these different *softwares*?

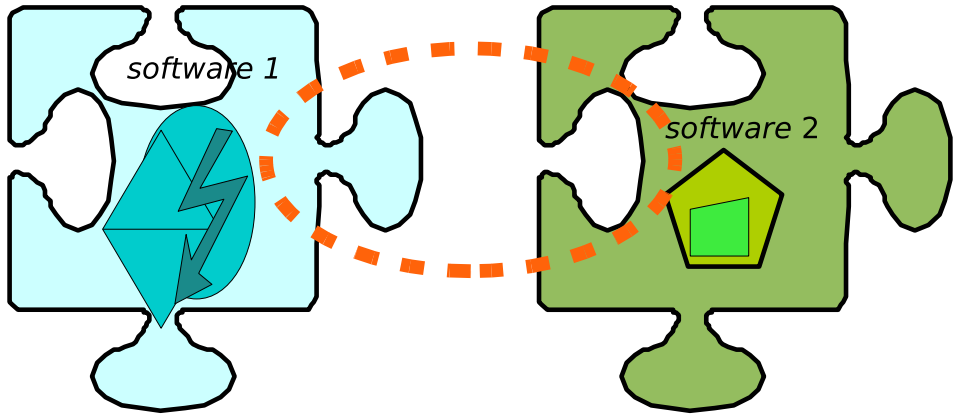
Possible solution



Connectors

Connector

An unit that intermediate the interaction among two or more components [Smeda et ali 2004].



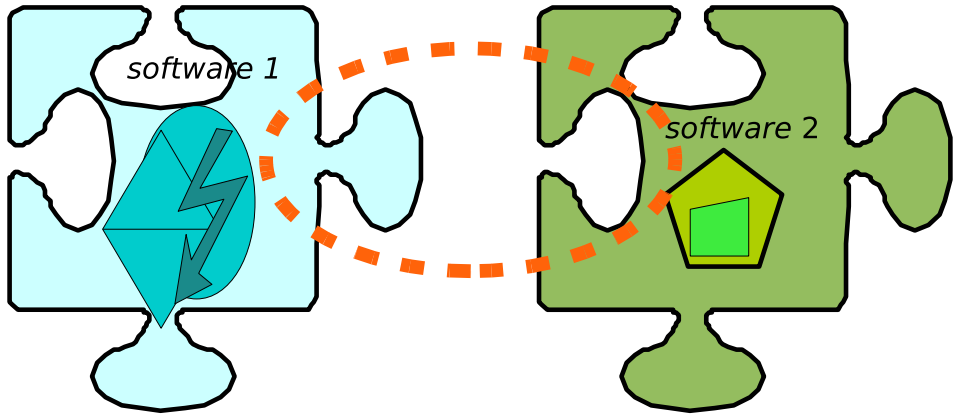
Role of Connector

Perform and control the communication among components in a connection.

Connectors

Connector

An unit that intermediate the interaction among two or more components [Smeda et ali 2004].



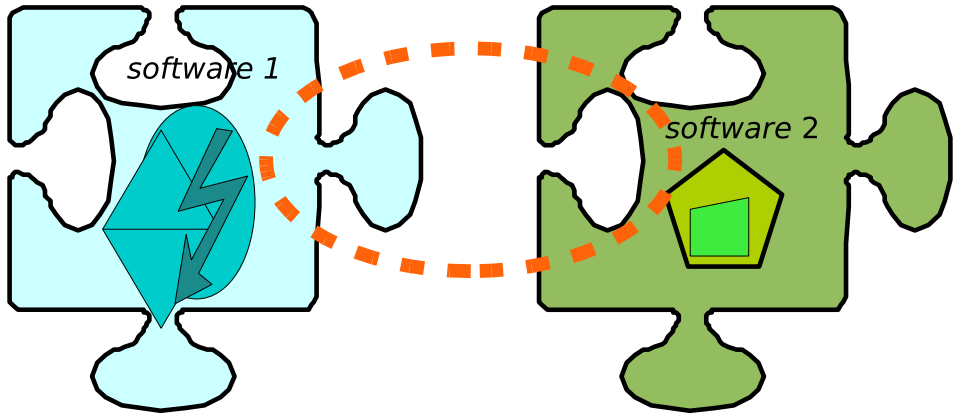
Role of Connector

Perform and control the communication among components in a connection.

Connectors

Connector

An unit that intermediate the interaction among two or more components [Smeda et ali 2004].



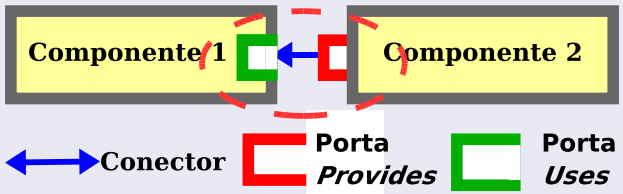
Role of Connector

Perform and control the communication among components in a connection.

Connectors Classification

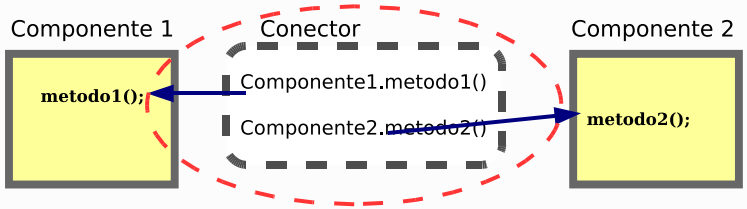
Endogenous Connector

- Connection between a port of a component and a port of other component.



Exogenous Connector

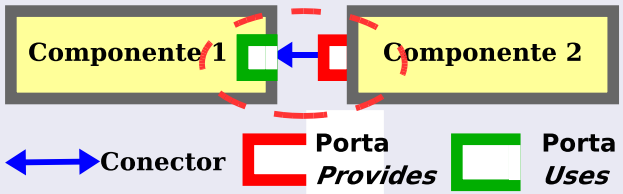
- Provides the control flow.
- Methods calls in the connector itself.



Connectors Classification

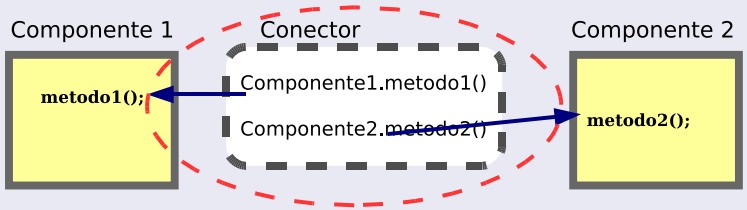
Endogenous Connector

- Connection between a port of a component and a port of other component.



Exogenous Connector

- Provides the control flow.
- Methods calls in the connector itself.



Deploy Model of the Connections

Connector Direct

Perform a connection directly between *provides* ports.

- Ex: invocation of an method directly in a port.

Connector Indirect

Perform indirectly the connections, for a calls sequence of a function or a method.

- Compatible to endogenous or exogenous connectors;
- It can use: directs or indirects connectors or deploying components.
- Ex: invocation of a method through of a *link*.

Deploy Model of the Connections

Connector Direct

Perform a connection directly between *provides* ports.

- Ex: invocation of an method directly in a port.

Connector Indirect

Perform indirectly the connections, for a calls sequence of a function or a method.

- Compatible to endogenous or exogenous connectors;
- It can use: directs or indirects connectors or deploying components.
- Ex: invocation of a method through of a *link*.

Deploy Model of the Connections

Connector Direct

Perform a connection directly between *provides* ports.

- Ex: invocation of an method directly in a port.

Connector Indirect

Perform indirectly the connections, for a calls sequence of a function or a method.

- Compatible to endogenous or exogenous connectors;
- It can use: directs or indirects connectors or deploying components.
- Ex: invocation of a method through of a *link*.

Deploy Model of the Connections

Connector Direct

Perform a connection directly between *provides* ports.

- Ex: invocation of an method directly in a port.

Connector Indirect

Perform indirectly the connections, for a calls sequence of a function or a method.

- Compatible to endogenous or exogenous connectors;
- It can use: directs or indirects connectors or deploying components.
- Ex: invocation of a method through of a *link*.

Models

- CCA model constraints:
  1. there are no intra-component communications; and
  2. component is responsible for internal thread creation, scheduling and termination.
- Models of parallel or distributed computations is a distinguishing feature of component models for HPC.
- Ex: components in nodes of a cluster communicate with each other using a message-passing platform to ensure high efficiency during execution of a parallel fragment of simulation.

Implementing indirect connectors

- First element - communication infra-structure (duct).
- Duct separates connector functionalities from the communication infra-structure.

Duct

Intermediary between the distributed system and the framework that should be able to expose the set of computational resources of the distributed system.

- Assuming point-to-point message-passing ducts.
- Two major reasons:
  - Most models of parallel/ distributed computations - message-passing models.
  - Message-passing infra-structures are used in practice - high performance application.

Implementing indirect connectors

- First element - communication infra-structure (duct).
- Duct separates connector functionalities from the communication infra-structure.

Duct

Intermediary between the distributed system and the framework that should be able to expose the set of computational resources of the distributed system.

- Assuming point-to-point message-passing ducts.
- Two major reasons:
  - Most models of parallel/ distributed computations - message-passing models.
  - Message-passing infra-structures are used in practice - high performance application.

Implementing indirect connectors

- First element - communication infra-structure (duct).
- Duct separates connector functionalities from the communication infra-structure.

Duct

Intermediary between the distributed system and the framework that should be able to expose the set of computational resources of the distributed system.

- Assuming point-to-point message-passing ducts.
- Two major reasons:
  - Most models of parallel/ distributed computations - message-passing models.
  - Message-passing infra-structures are used in practice - high performance application.

Implementing indirect connectors

- Numerous HPC legacy parallel codes for distributed memory systems - communications $\longrightarrow$ via message exchange.
- Embedding such applications into components are possible at different levels of abstractions.
- The lowest level consists in performing communications directly through a message-passing duct.
- Making legal communications: indirection between a component and a message-passing duct.

Implementing indirect connectors

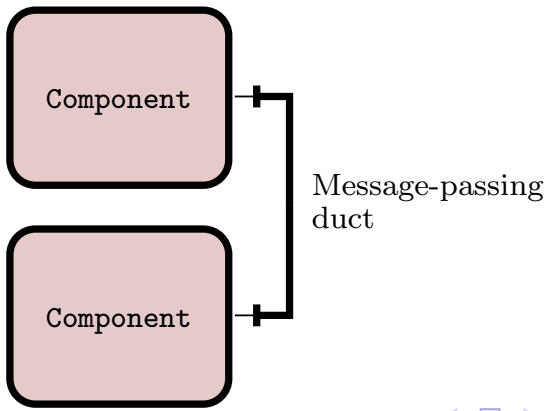
- Indirection: if one or more CCA components of special type, called *connector*, implementing `Connector` that extends `Component`.
- Connector is parameterized by duct.
- `Link`: providing access to functionalities of duct of connector.
- `Connector` includes methods for setting its duct and creating links.
- `Connector` is extended with `ExogenousConnector`, which provides a `GoPort`.

Implementing indirect connectors

Connecting space

set of locations with disjoint address spaces and a set of links connecting pairs of locations and sharing the same duct.

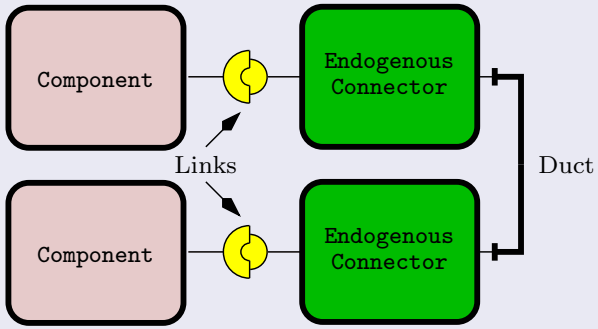
- Attribute of a connecting space: timing characteristics of the communications (Ex: synchronization, global states recordings).
- Lowest level consists in performing communications directly through a message-passing duct:



Method Invocation Applications

Simple Indirect Connection

- Use of an endogenous connector and link ports to implement such an indirection.



- Link port used is specialized to message-passing communications - methods with 1:1 correspondence with communic. functions;
- Endogenous connectors connect uses ports, since communication functionalities are accessed by function calls in legacy code (Ex: MPI).

Event-driven applications

- Model used to design and analyze parallel algorithms in message-passing systems is the *event-driven* model;
- Computation is carried out in system composed by set of nodes, interconnected by set of point-to-point bidirectional communication channels;
- Node is able to perform computations and to interact with neighbors sending or receiving messages through channels;
- Distributed computation is described by initial global state (comprising an initial state for each node and no messages in transit);

Event-driven applications (cont.)

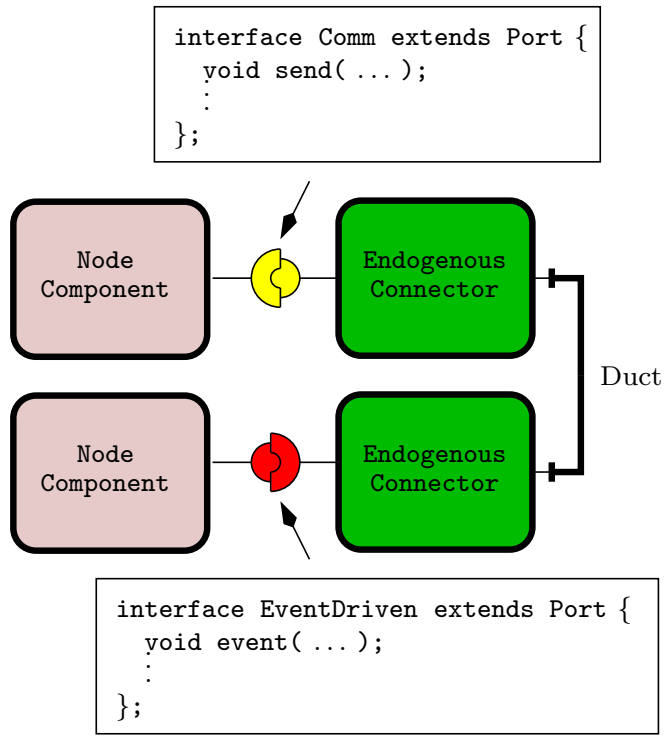
- Local Computations (LC) performed by node, and interactions between nodes;
- LC form sequence of *events*, whose first member may be spontaneous (LC - does not depend on the reception of any message);
- Non-spontaneous event is node's reaction when it receives a message.

Event-driven applications

- Receiving of message affects node's local state - execution of method `event()`;
- It encapsulates actions of particular computation associated with this node;
- This method take message as input;

Event-driven applications

- Besides changing the local state of its node, execution of `event()` produces as result set of messages that are sent in order to induce events at other nodes.



A Sample Application

- Application: Coloring Problem (minimum number of colors);
- Way to improve the efficiency: distributed memory parallel branch and bound implementation;
- Our implementation combines two communication infrastructures;
- one is used to connect branch-and-cut components;
- Exogenous connector control the execution by requesting a subproblem

Concluding Remarks

- Our solution is CCA compliant framework for HPC applications;
- It acts as a platform that is able to support higher-level component models.
- It enriches existing component architectures with the support for several models of computation in a single application;
- It addresses the problem of high heterogeneity of contemporary distributed architectures that has emerged for application in scientific domain.

End

Suggestions?