Grid-Based PDE.Mart

A PDE-Oriented PSE for Grid Computing

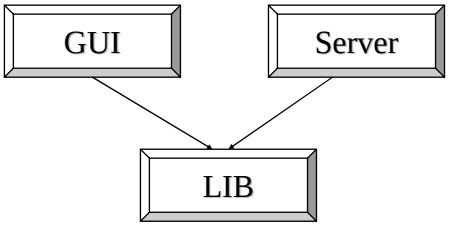
GY MAO, M. MU, Wu ZHANG, XB ZHANG

School of Computer Science and Engineering, Shanghai University, CHINA

Department of Mathematics, Hong Kong University of Science and Technology, HK

Introduction of PDE.Mart

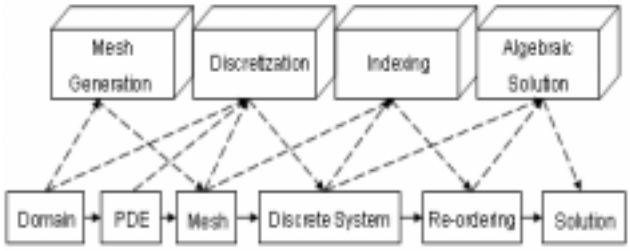
- Network-based PDE.Mart is a problem-solving environment (PSE) for solving partial differential equations.



Introduction of PDEMart

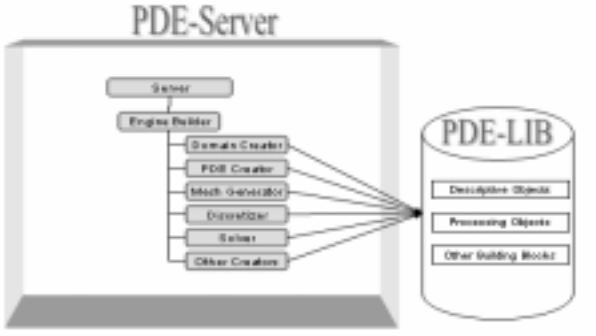
- Grid-based PDE.Mart is a service-oriented system. It solves a user-supplied PDE problem by using PDE solving services available on distributed grid nodes instead of a single server.
- The PDE solving objects in PDE-LIB of network-based PDE.Mart are converted to corresponding services and deployed to different grid nodes
- Grid-based PDE.Mart extends network-based PDE.Mart from object integration to service integration

Flow chart for a typical PDE computation



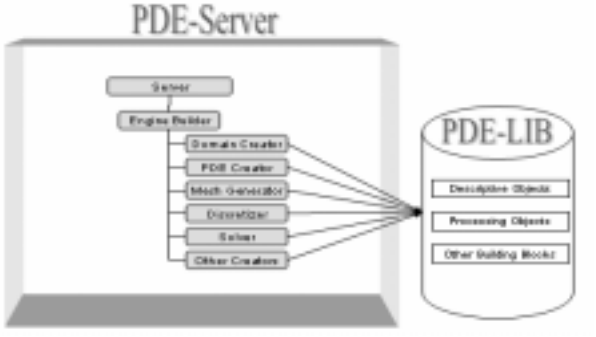
The computational process starts with a geometric domain and a PDE model, goes through a mesh, a discrete system, a re-ordering index mapping, and finally arrives at a solution. At each step, a numerical procedure shown as a box, such as mesh generation, discretization, indexing, and algebraic solution, is executed to generate an output state shown as a rectangle by taking one or more previous states as the input

Overall structure of network-based PDE-Server



The server class is mainly responsible for managing the client-server protocol. Once running, it keeps listening to clients on the network to provide PDE solution services. Upon each client's connection, a socket is created on the server to establish the private communication channel between the client and the server. The engine builder is then invoked to create a computational session for building the client's own computational engine

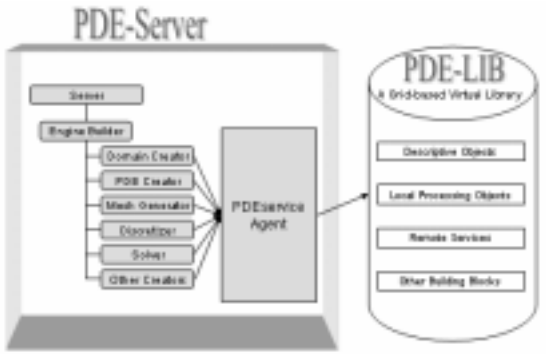
Overall structure of network-based PDE-Server



The engine builder class is responsible for creating a computational session. Computational sessions are multi-threaded and concurrent. Each client owns an individual computational session that communicates with the instance of PDE-GUI through the assigned socket. The major components of the engine builder include Domain Creator, PDE Creator, Nonlinear PDE Solution Creator, Mesh Generator, Discretizer, Indexer, Solver, and Blackbox Creator. The invocation of these creators is interactively controlled by the client via a request query

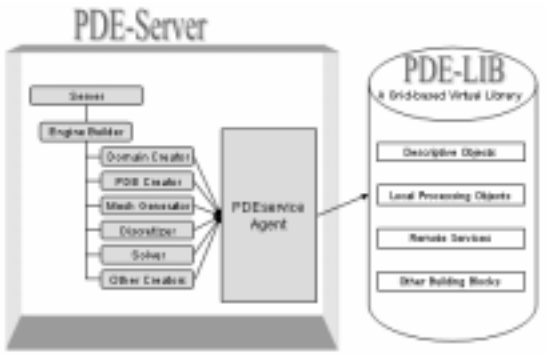
Grid-Based PDE.Mart

In a grid environment, the processing objects are usually distributed among different grid nodes. Therefore, PDE-LIB now turns to be a virtual library in grid-based PDE.Mart. Correspondingly, those processing objects available on the remote nodes are converted to services



Grid-Based PDE.Mart

PDEServiceAgent is a centralized agent system, where a resource database is maintained for all the registered processing objects. Each invocation to a numerical procedure is now directed to this agent system. The agent then determines whether and how a remote service is invoked in order to generate the user-requested target object



Introduction of Web service

- Web service is one of the most popular distributed technologies to support service-oriented software development
- A web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another web service
- The web services technology uses SOAP (Simple Object Access Protocol) for the XML payload and uses a transport such as HTTP to carry the SOAP messages back and forth

Apache AXIS

- Apache AXIS is an implementation of the SOAP submission to W3C. It is the third generation of Apache SOAP
- Apache AXIS supports the Web Services Description Language (WSDL), which allows users to easily build stubs to access remote services, and to automatically export machine-readable descriptions of deployed services from AXIS
- Therefore, the Apache AXIS implementation can still keep grid-based PDE.Mart object-oriented, where all the arguments in a service invocation are encapsulated descriptive objects

Implementation using Apache AXIS

Step 1 Generation of WSDL document for a JAVA class

Step 2 Deploying a service on a grid node

Step 3 Application side implementation

Generation of WSDL document for a JAVA class

Suppose we would like to convert a Java class *FivePSDiscretizer* in PDE-LIB to a web service. This processing class contains a method *discretizePDE* that implements a numerical procedure Five-Point-Star for discretizing an elliptic PDE. The method takes a list of input arguments and returns an object of type *BeanDiscretization* as the generated discrete algebraic system

```
package Disserviceclient;
public class FivePSDiscretizer {
    public BeanDiscretization dis;
    ......
    public FivePSDiscretizer() {}
    public BeanDiscretization discretizePDE(BeanGlobalControl gc,
                BeanMesh mes,BeanNonrectdom nrdom,
                String  pdestring, boolean isdomrect )
    {
      ......
      return dis;
    }}
```

The AXIS tool *JAVA2WSDL* takes this class as the input and generates a WSDL document that describes the operation of the service, including its name, arguments from the request, and output from the response

WSDL Segment

```
<wsdl:message name="discretizePDERequest">
<wsdl:part name="in0" type="impl:BeanGlobalControl" />
<wsdl:part name="in1" type="impl:BeanMesh" />
<wsdl:part name="in2" type="impl:BeanNonrectdom" />
<wsdl:part name="in3" type="soapenc:string" />
<wsdl:part name="in4" type="xsd:boolean" />
</wsdl:message>
<wsdl:message name="discretizePDEResponse">
<wsdl:part name="discretizePDEReturn"
      type="impl:BeanDiscretization"/>
</wsdl:message>
<wsdl:portType name="FivePSDiscretizer">
<wsdl:operation name="discretizePDE"
      parameterOrder="in0 in1 in2 in3 in4">
<wsdl:input message="impl:discretizePDERequest"
      name="discretizePDERequest"/>
<wsdl:output message="impl:discretizePDEResponse"
      name="discretizePDEResponse"/>
</wsdl:operation>
</wsdl:portType>
```

Deploying a service on a grid node

- For deploying a service, say FivePSDiscretizer, on a grid node, by using the WSDL document *FivePSDiscretizer.wsdl* above, the *WSDL2JAVA* tool generates the corresponding WSDD (Web Services Deployment Descriptor) files deploy.wsdd and undeploy.wsdd
- The generated WSDD file deploy.wsdd describes the deploy information of the implementation class, and contains extra meta data describing the operations and parameters of the implementation class

WSDD (Web Services Deployment Descriptor)

```
<service name="FivePSDiscretizer" type="" regenerateElement="true"
 provider="java:RPC" style="rpc" use="encoded" validate="true">
<parameter name="scope" value="Request" regenerateElement="false"/>
<parameter name="className"
 value="Disserviceclient.FivePSDiscretizer"
 regenerateElement="false"/>
<parameter name="allowedMethods" value="*"
 regenerateElement="false"/>
<namespace>http://Disserviceclient/</namespace>
<typeMapping
 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 qname="ns4:BeanDiscretization"
 languageSpecificType="java:Disserviceclient.BeanDiscretization"
 serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
 deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
 name="BeanDiscretization" regenerateElement="true"
 xmlns:ns4="http://Disserviceclient/"/>
```

Application side implementation

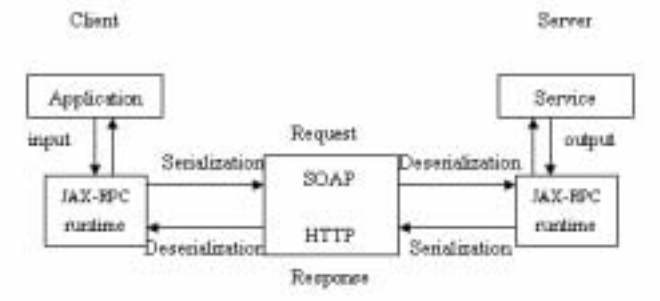
- To consume a web service in grid-based PDE-LIB, several Java classes must also be generated on the application side

  FivePSDiscretizerSoapBindingStub.java
  FivePSDiscretizerServiceLocator.java
  FivePSDiscretizerService.java
  FivePSDiscretizerPort.java

- An application program does not instantiate a stub directly. It instantiates a service locator and calls a get method which returns a stub. The Service Definition Interface (SDI) is an interface derived from a WSDL's portType. This interface is used to access the operations on the service. A stub class implements the SDI. It contains the code that turns the method invocations into SOAP calls using the AXIS Service and Call objects. It stands in as a proxy for the remote service, letting the application program call it exactly as if it were a local object

- This application program is the centralized agent system PDEServiceAgent. It accepts service requests from various creators in PDE-Server; instantiates appropriate service locators according to the resource database, receives service responses, and finally pass them back to the requesting creators

Workflow of Grid-based PDE-Server



To call a service and get the result, all input parameters are serialized into the XML format at the client side and passed to the server via HTTP. On the server side, the XML document is deserialized to retrieve the original information

SOAP Request and Response Information

Further Work

- Enhance the functions of PDEServiceAgent
- Explore other Web Service techniques besides Apache AXIS and make comparisons.
- Explore other grid techniques besides Web Service
- Introduce an open interface for free integration of third-party services