NBCR NATIONAL BIOMEDICAL COMPUTATION RESOURCE
Conduct, catalyze and enable multiscale biomedical research

Providing Dynamic Virtualized Access to Grid Resources via the Web 2.0 Paradigm

Luca Clementi
clem@sdsc.edu

Zhaohui Ding
zhaohui.ding@email.jlu.edu.cn

Sriram Krishnan
sriram@sdsc.edu

Xiaohui Wei
weixh@jlu.edu.cn

Peter W. Arzberger
parzberger@ucsd.edu

Wilfred Li
wilfred@sdsc.edu


CRBS SDSC it² UCSD


Outline

- Our vision of grid computing
- Introduction to the Opal Toolkit
- Novel contributions
  - Automatic interface generation for Opal
  - Modeling applications as first class resources via CSF4 metascheduler
  - Opal CSF4 integration

Outline

- Our vision of grid computing
- Introduction to the Opal Toolkit
- Novel contributions
  - Automatic interface generation for Opal
  - Modeling applications as first class resources via CSF4 metascheduler
  - Opal CSF4 integration

The Problem

- Leveraging grid system is still too complex from scientific end-users:
  - High deployment and maintenance cost
  - User has to learn low-level grid related concepts:
    - grid credential management
    - staging data
    - job submission
    - etc.

Proposed Solution

- Application centric view of the grid
  - Applications as resources for scheduling
  - Applications wrapped as web services
- Multiple user interfaces (GUI)
  - Command-line description language
  - Web-based-customized submission form

Outline

- Our vision of grid computing
- Introduction to the Opal Toolkit
- Novel contributions
  - Automatic interface generation for Opal
  - Modeling applications as first class resources via CSF4 metascheduler
  - Opal CSF4 integration

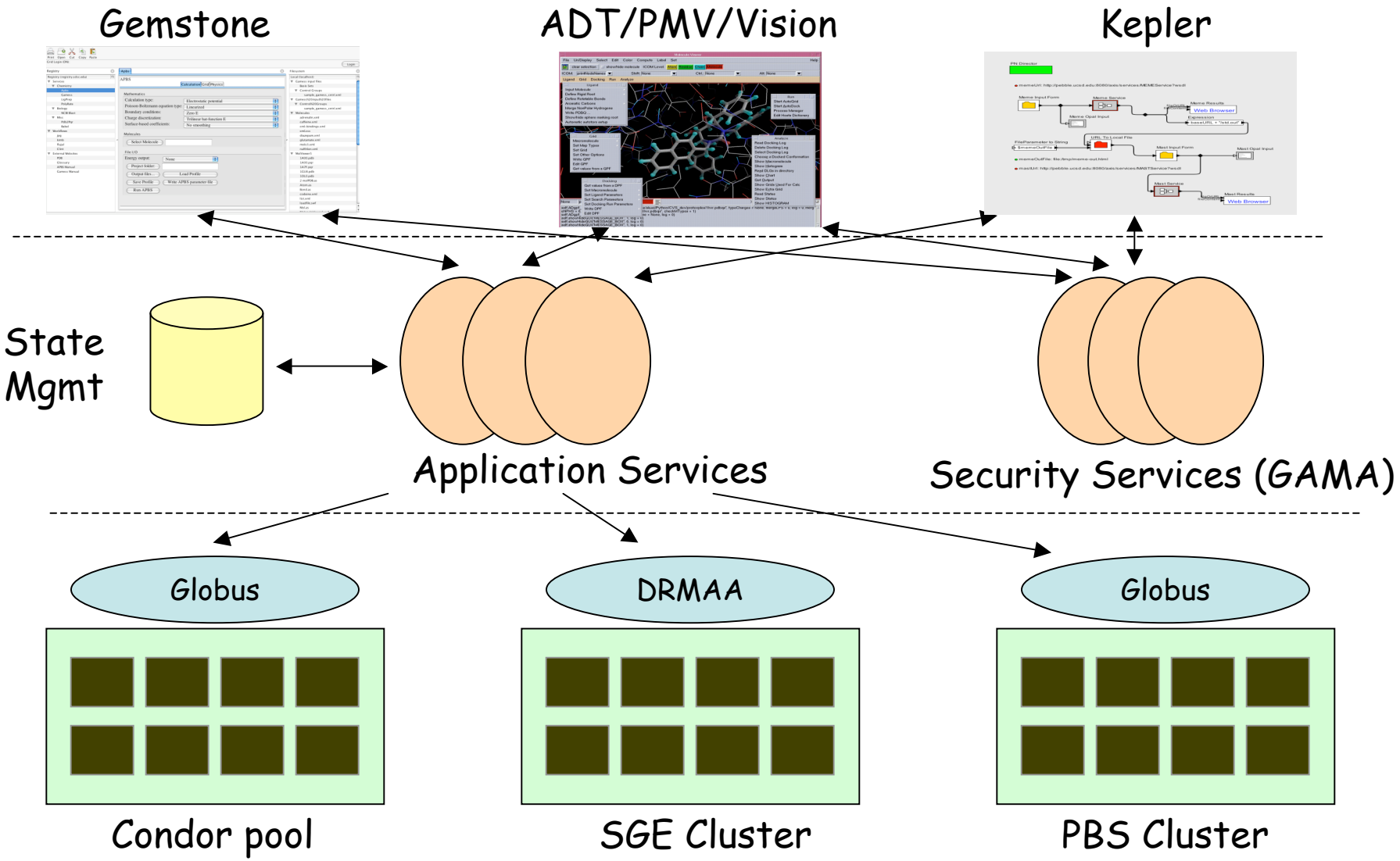
Opal Toolkit

- Opal allows application developers to publish command-line applications using Web services
  - Minimal deployment effort: no coding, only a simple configuration file (next slide)
  - Common interface: every application uses the same WSDL
  - It takes care of data staging
  - It supports submission via:
    - Fork
    - Globus GRAM
    - DRMAA
  - Used in NBCR, CAMERA, GLEON, among others

Publish PDB2PQR (appConfig file)

```
<appConfig xmlns="http://nbcr.sdsc.edu/opal/types"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <metadata appName="PDB2PQR">
    <usage><![CDATA[
            python pdb2pqr.py [options] --ff={forcefield} {path} {output-path}
              ]]>
    </usage>
    <info xsd:type="xsd:string">
    <![CDATA[
         The required arguments are as follows:
         <forcefield>
             The forcefield to use -- currently AMBER, CHARMM, PARSE and TYL06
             are supported.
       ...
       ]]>
    </info>
  </metadata>
  <binaryLocation>/usr/local/pdb2pqr-1.2.1/pdb2pqr.py</binaryLocation>
  <defaultArgs>--verbose</defaultArgs>
  <parallel>false</parallel>
</appConfig>
```

Opal Usage Scenario



Opal Toolkit (client)

- Several clients APIs available: Java, Python, PERL.
- Command line generic client:

```
# java edu.sdsc.nbcr.opal.GenericServiceClient
                    -l http://localhost:8080/axis/services/PDB2PQRServicePort
                    -r launchJob
                    -a "-ipdb sample.pdb -h -opdb output.pdb“
                    -f etc/sample.pdb
```

- Too complex for beginner users
- Graphical User Interface

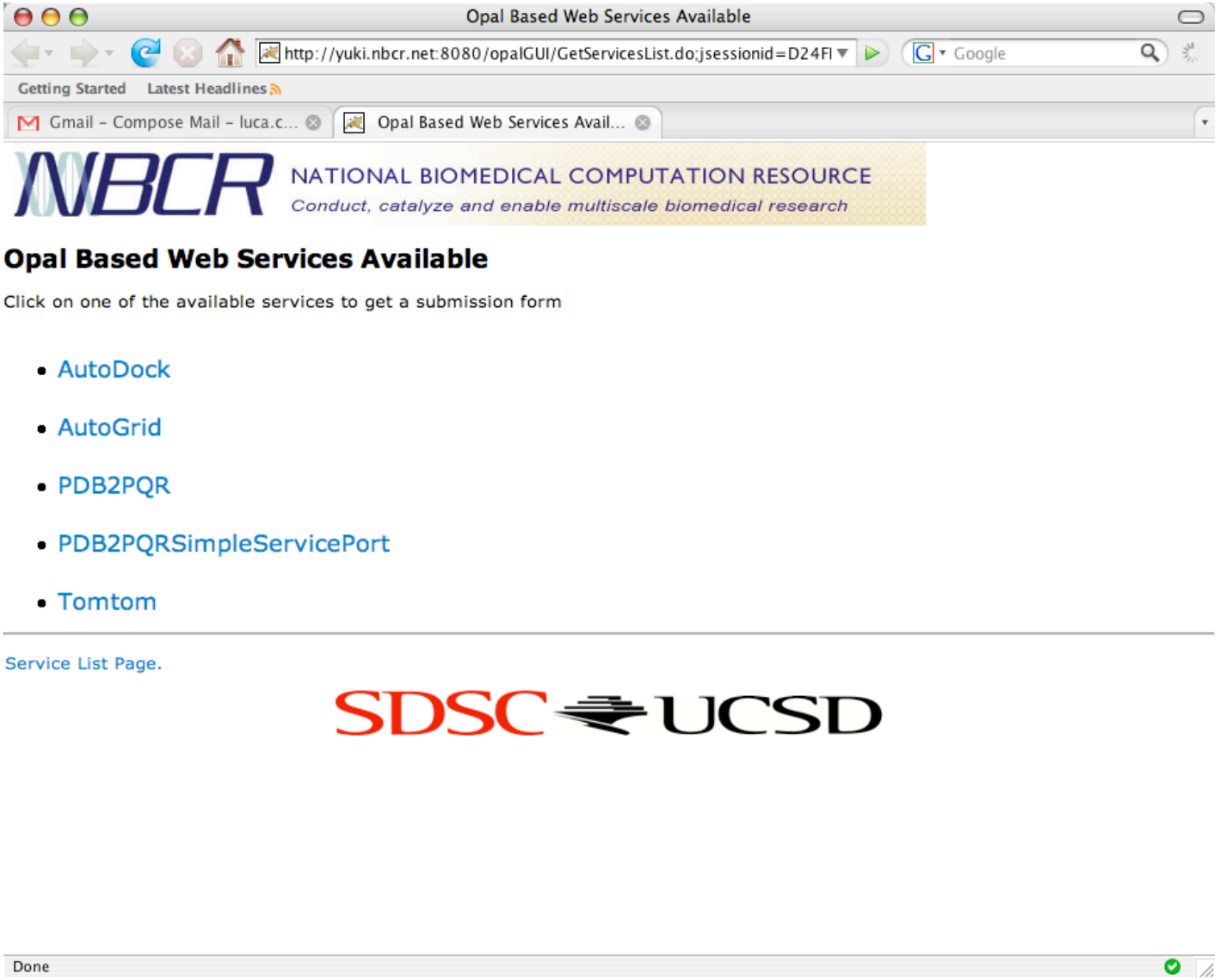
Outline

- Our vision of grid computing
- Introduction to the Opal Toolkit
- Novel contributions
  - Automatic interface generation for Opal
  - Modeling applications as first class resources via CSF4 metascheduler
  - Opal CSF4 integration

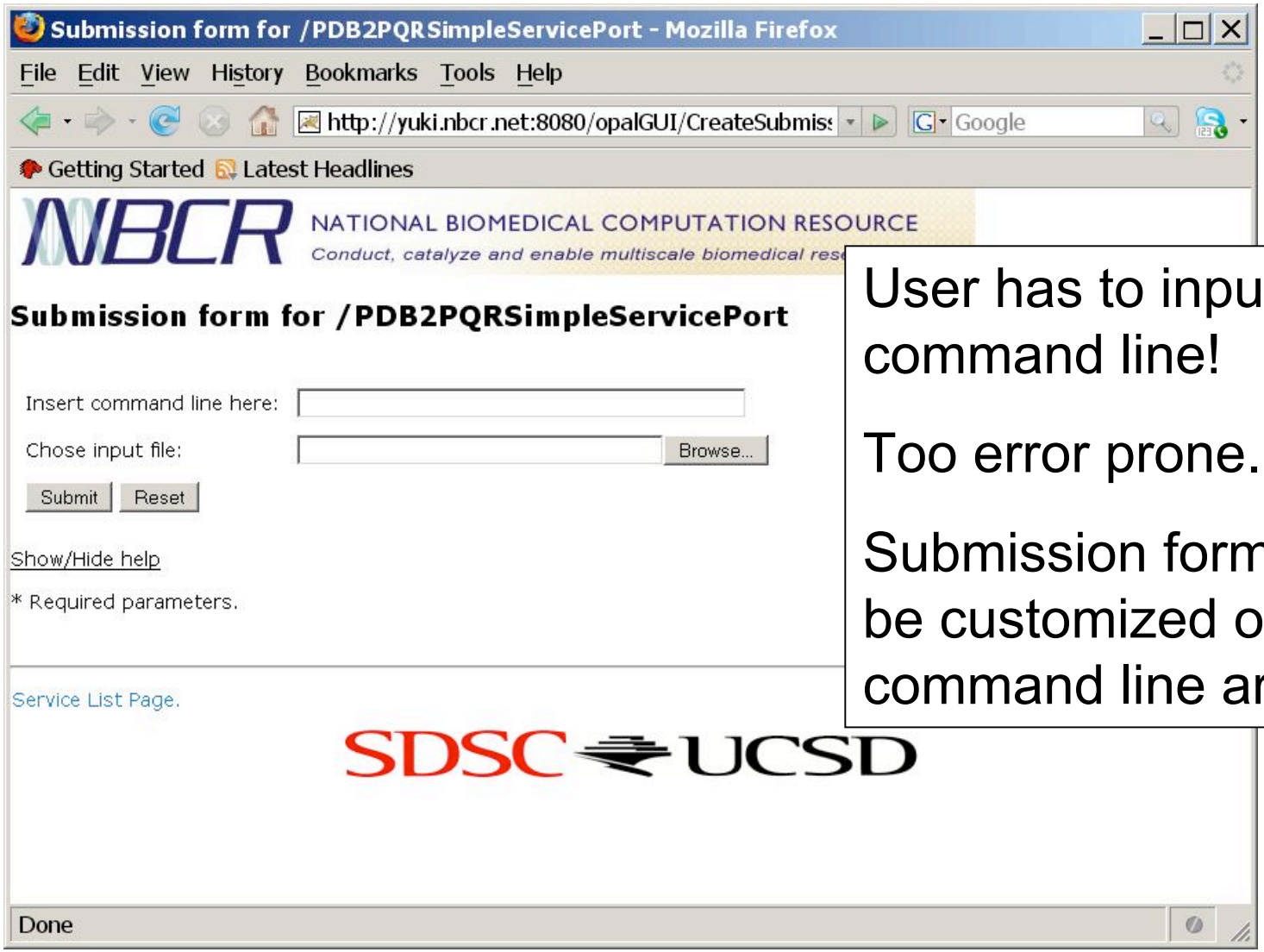
Opal GUI

- Motivations:
  - Richer end-user experience
  - Simpler for inexperienced user
- Main characteristics:
  - Working out of the box (no configuration)
  - Multiplatform -> Web interface
  - Implemented in Java
- Key features:
  - List of services
  - Simple submission form
  - Advanced submission form

List of Services



Simple Submission Form



User has to input command line!

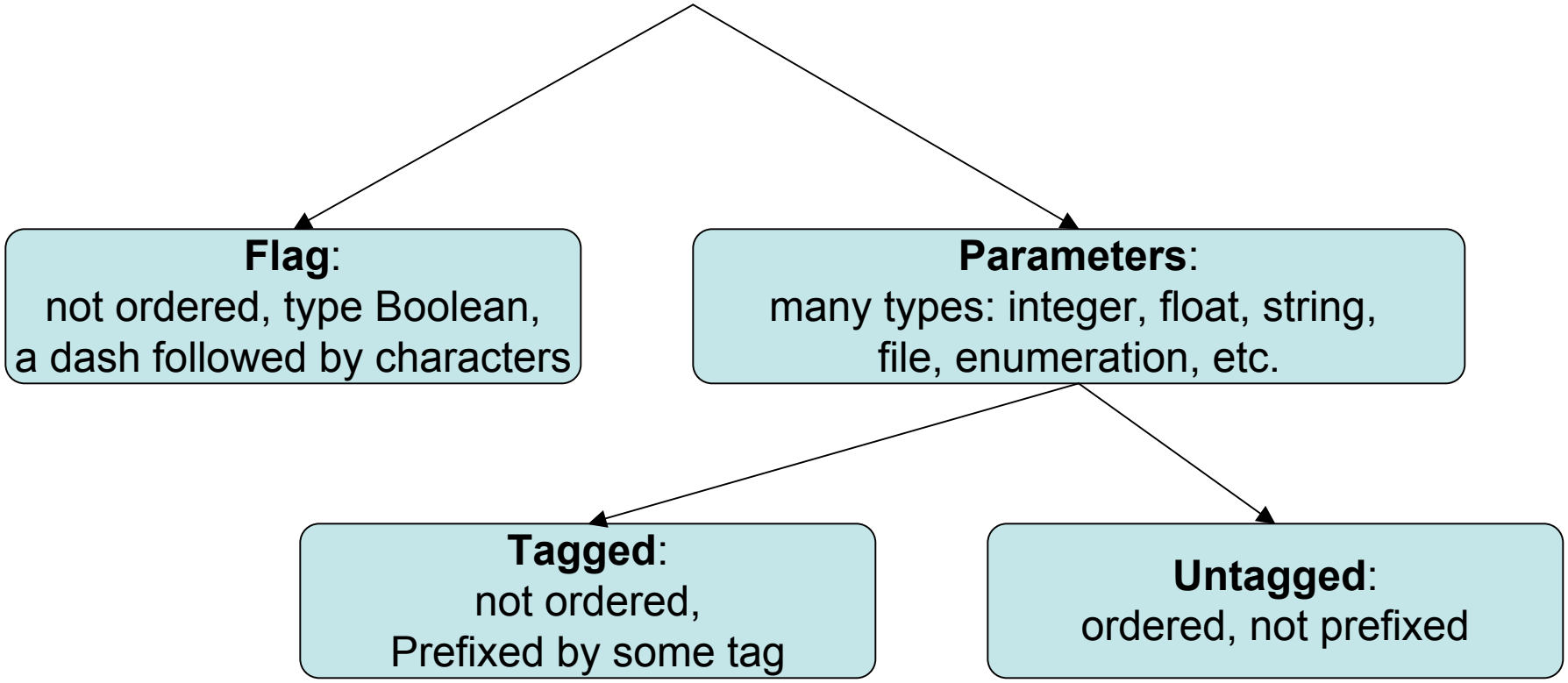
Too error prone.

Submission form should be customized on command line arguments

Advanced Submission Form

- An **optional** tag in the appConfig file to describe input parameters (`types`)
- It is a command line syntax description language

Command Line Input Arguments Taxonomy



Advanced Submission Form

- Grouping capability:
  - To group several parameters together
  - A group can be exclusive
- Default values
- An example of the command line syntax description language and of the form…

```
<types xmlns="http://nbcr.sdsc.edu/opal/types">
  <flags>
    <flag>
      <id>nodebump</id>
      <tag>--nodebump</tag>
      <textDesc>Do not perform the debumping operation</textDesc>
    </flag>
    …
  </flags>
  <taggedParams>
    <separator>=</separator>
    <param>
      <id>ffout</id>
      <tag>--ffout</tag>
      <paramType>STRING</paramType>
      <textDesc>Instead of using the standard canonical naming scheme for residue and atom names, use the names from
      the given forcefield</textDesc>
    </param>
    …
  </taggedParams>
  <untaggedParams>
    <param>
      <id>output-path</id>
      <paramType>FILE</paramType>
      <ioType>OUTPUT</ioType>
      <textDesc>The desired output name of the PQR file to be generated</textDesc>
    </param>
    …
  </untaggedParams>
  <groups>
    <group>
      <name>inputParam</name>
      <elements>inFile inId</elements>
      <required>true</required>
      <exclusive>true</exclusive>
      <textDesc>Input file to be used (choose one of the two options)</textDesc>
    </group>
    …
  </groups>
</types>
```

Flags

Tagged Parameters

Untagged Parameters

Groups

```
<flags>
  <flag>
    <id>nodebump</id>
    <tag>--nodebump</tag>
    <textDesc>Do not perform the debumping operation</textDesc>
  </flag>
  …
</flags>
<taggedParams>
  <separator>=</separator>
  <param>
    <id>ffout</id>
    <tag>--ffout</tag>
    <paramType>STRING</paramType>
    <textDesc>Instead of using the standard canonical naming scheme for
    residue and atom names, use the names from the given forcefield
    </textDesc>
  </param>
  …
</taggedParams>
```

Flags

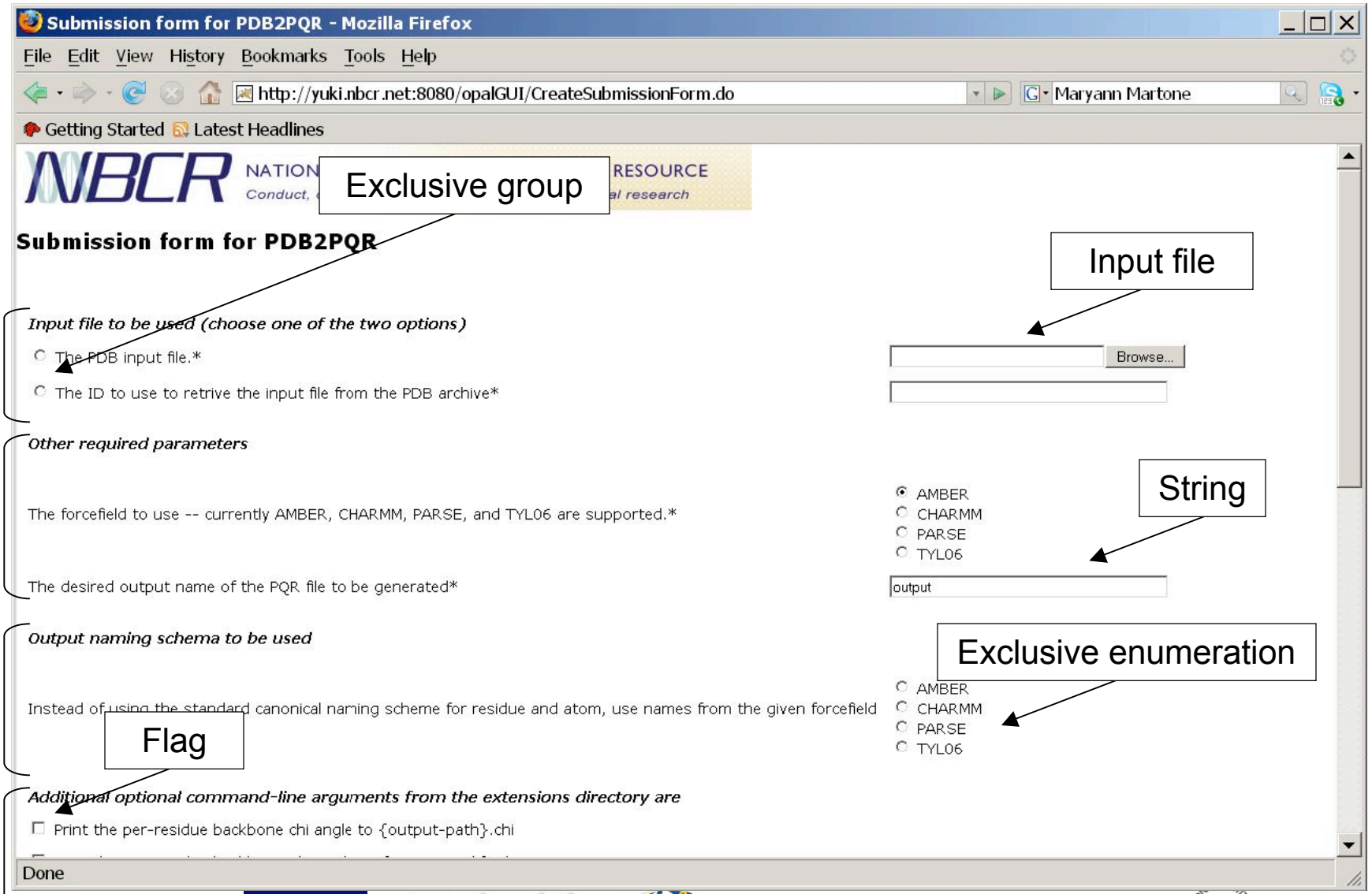
Tagged Parameters

```
<untaggedParams>
  <param>
    <id>output-path</id>
    <paramType>FILE</paramType>
    <ioType>OUTPUT</ioType>
    <textDesc>The desired output name of the PQR file to be generated
    </textDesc>
  </param>
  …
</untaggedParams>
<groups>
  <group>
    <name>inputParam</name>
    <elements>inFile inId</elements>
    <required>true</required>
    <exclusive>true</exclusive>
    <textDesc>Input file to be used (choose one of the two options)
    </textDesc>
  </group>
  …
</groups>
```

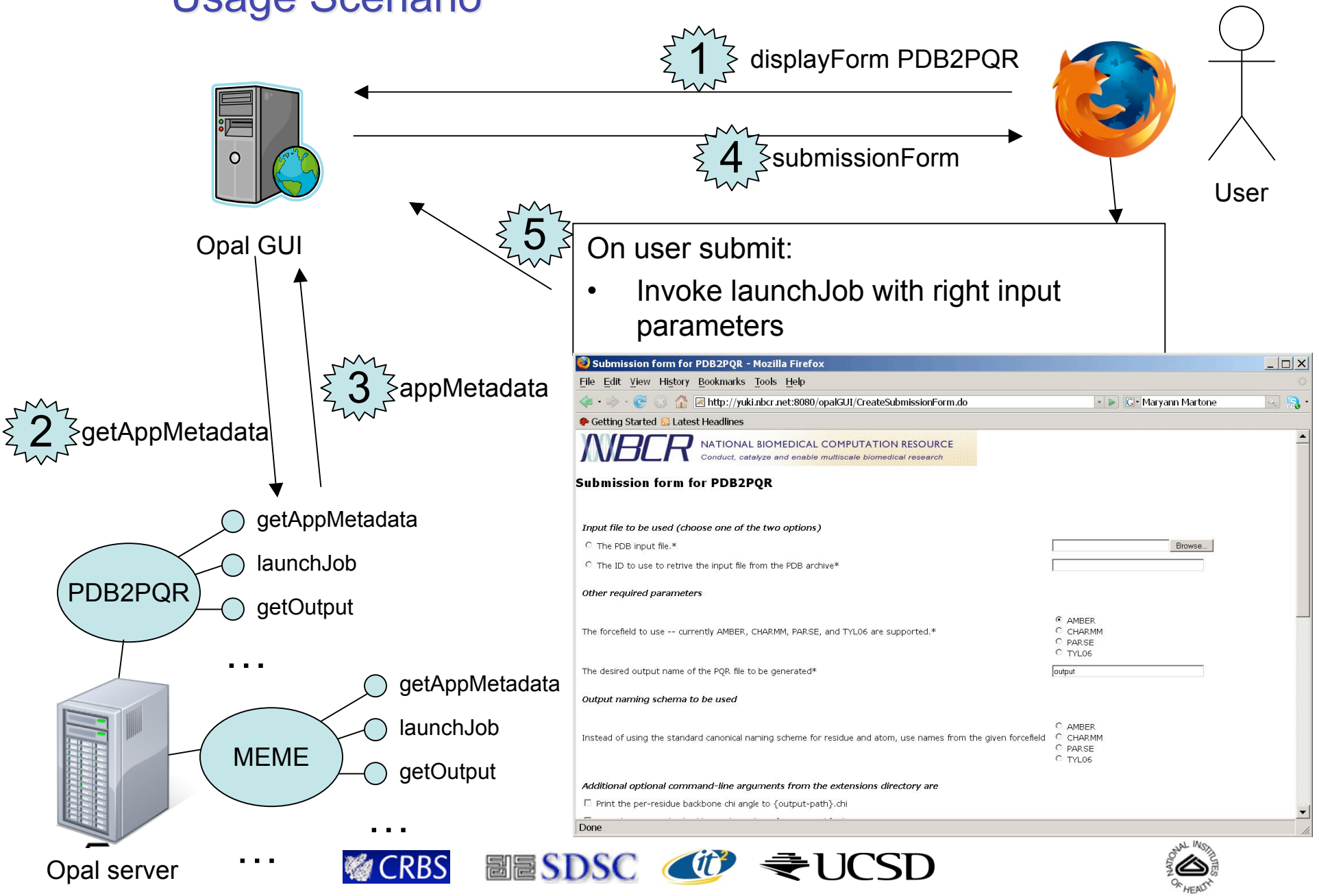
Untagged Parameters

Groups

PDB2PQR Advanced Submission Form



Usage Scenario



CRBS SDSC it2 UCSD NATIONAL INSTITUTES OF HEALTH

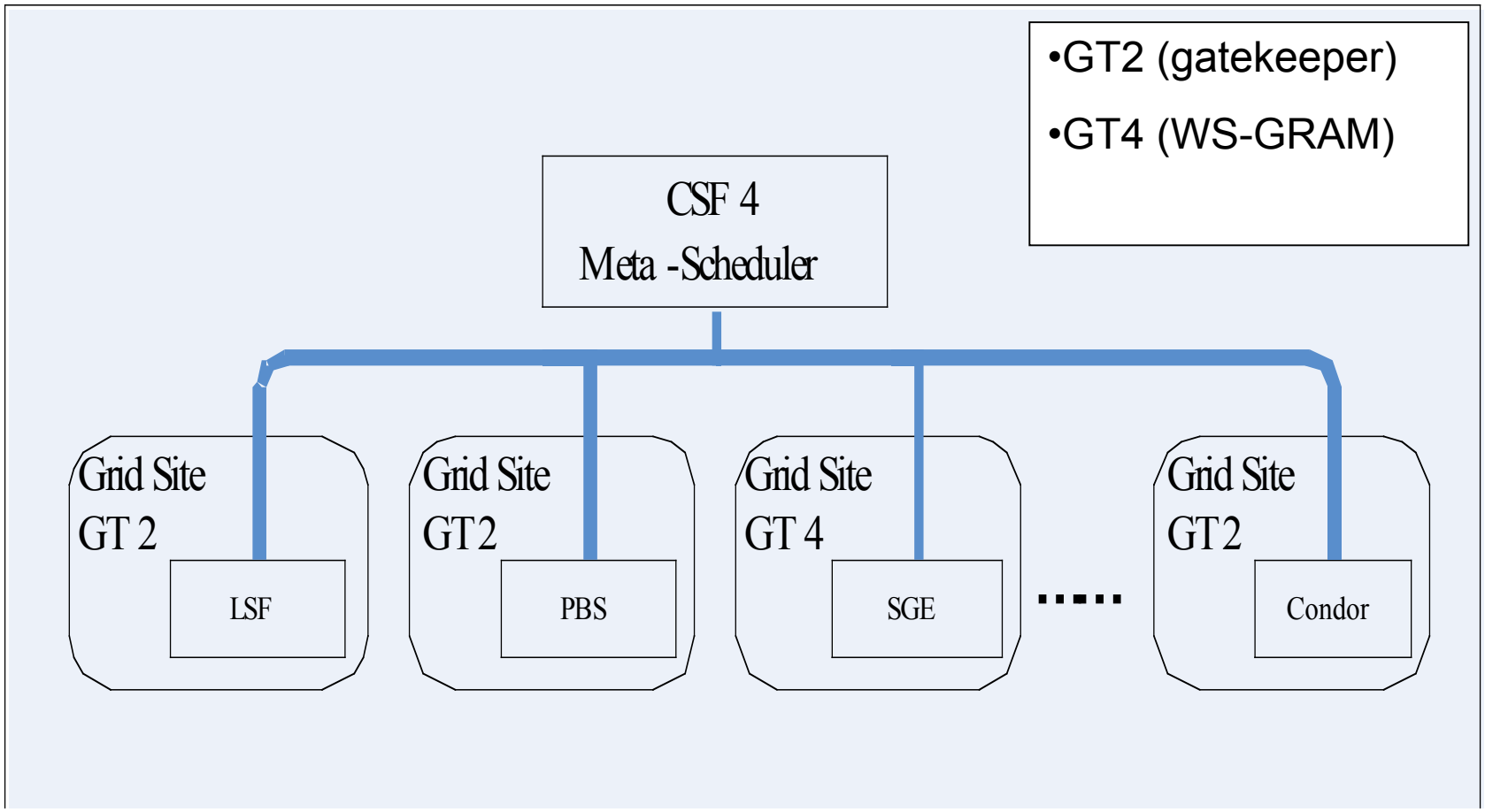
Outline

- Our vision of grid computing
- Introduction to the Opal Toolkit
- Novel contributions
  - Automatic interface generation for Opal
  - Modeling applications as first class resources via CSF4 metascheduler
  - Opal CSF4 integration

CSF4

- ***C***ommunity ***S***cheduler ***F***ramework:
  - Open Source project and can be accessed at http://sourceforge.net/projects/gcsf
  - Developed by Lab. of Distributed Computing and System Architecture, Jilin University, China
  - It is a metascheduler framework hosted as an Execution Component in GT4 container
  - It uses WSRF compliant services
  - It can submit jobs to Globus

CSF4 Typical Deployment



CSF4

- Functionalities
  - Submit jobs to Grid without Specifying Cluster
  - Monitor and Control Jobs
  - Support for Queues
  - Automatic data-staging
  - Extensible scheduling framework
    - Schedule jobs by custom-built polices
  - Command-line and Web based client (CSF4 Portlet)

CSF4 New Feature (1/2)

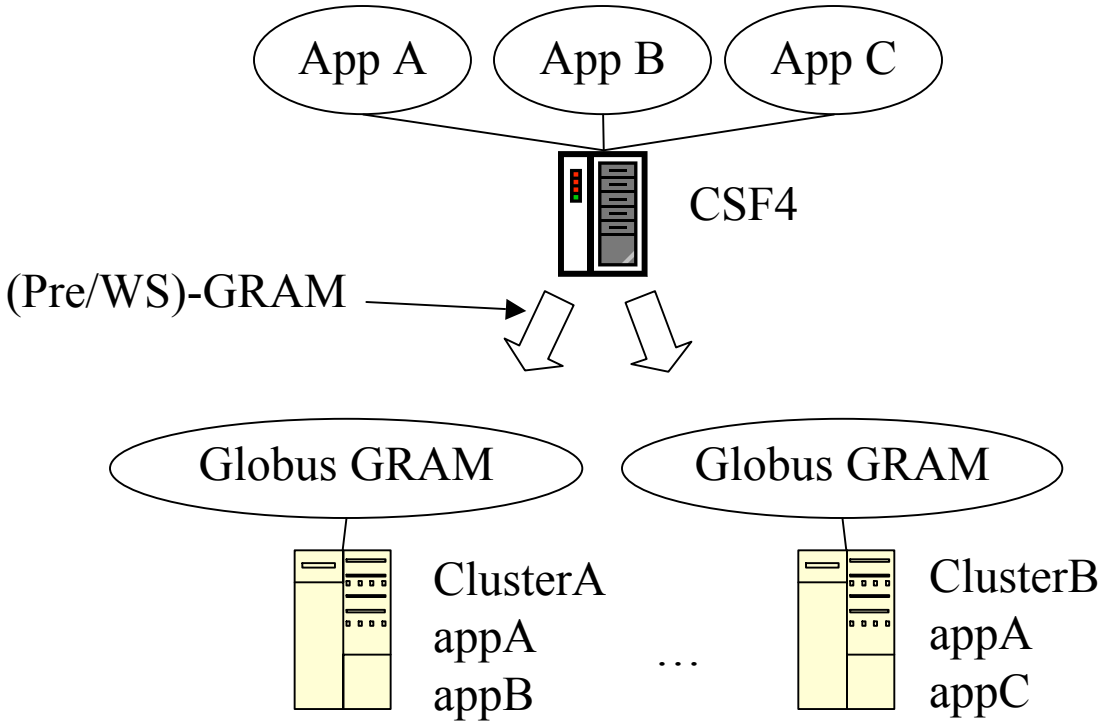
Users want to run applications

- Application based scheduling:
  - CSF4 keeps a table of available applications

Application name	Resources
appA	clusterA:/usr/local/appA
appA	clusterB:/usr/share/bin/appA
appB	clusterA:/some/path/appB
appC	clusterB:/some/path/appC

CSF4 New Feature (2/2)

- Virtualization of computational resource
  - Clients submit jobs specifying only applications name
  - Computational resources are hidden by the metascheduler

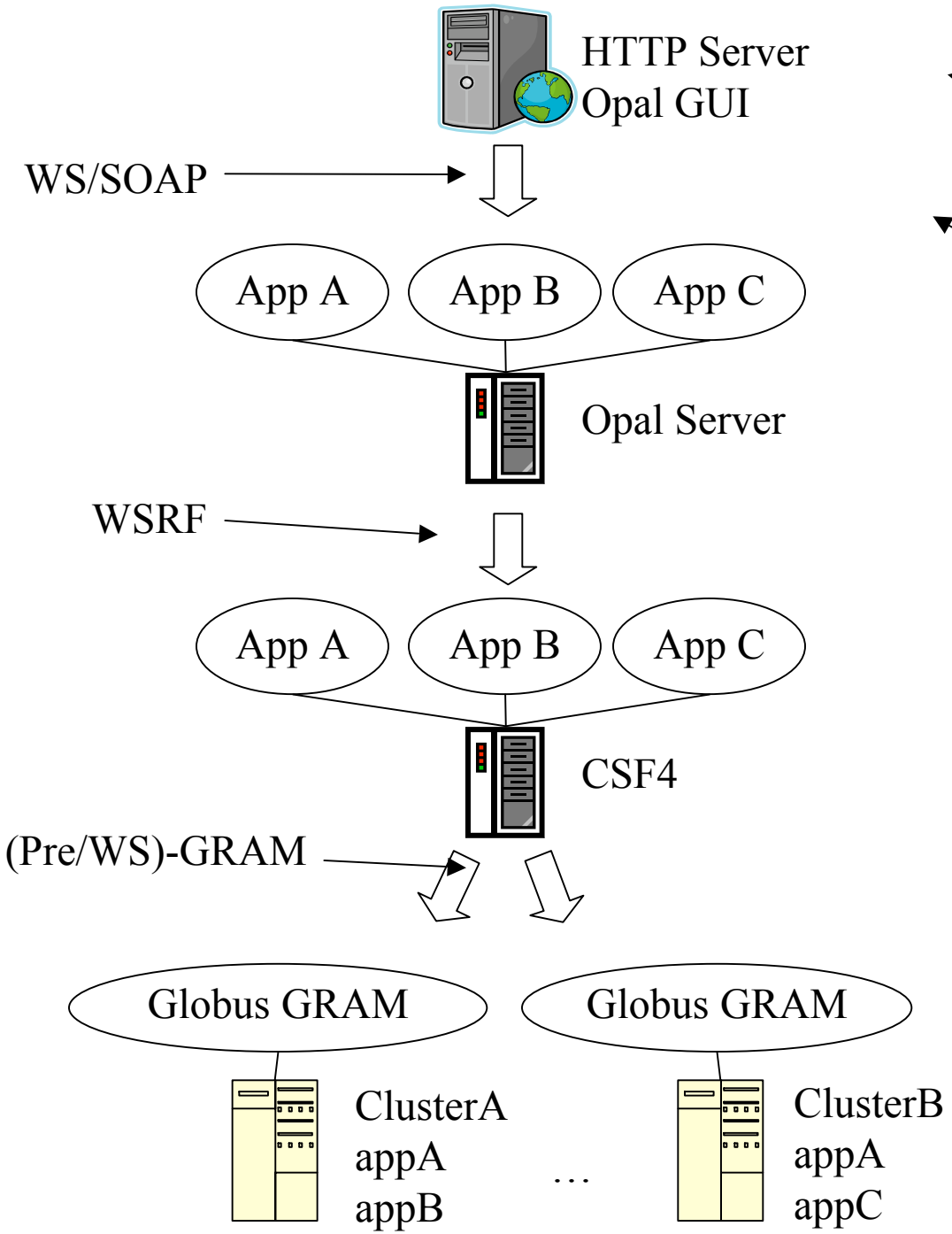


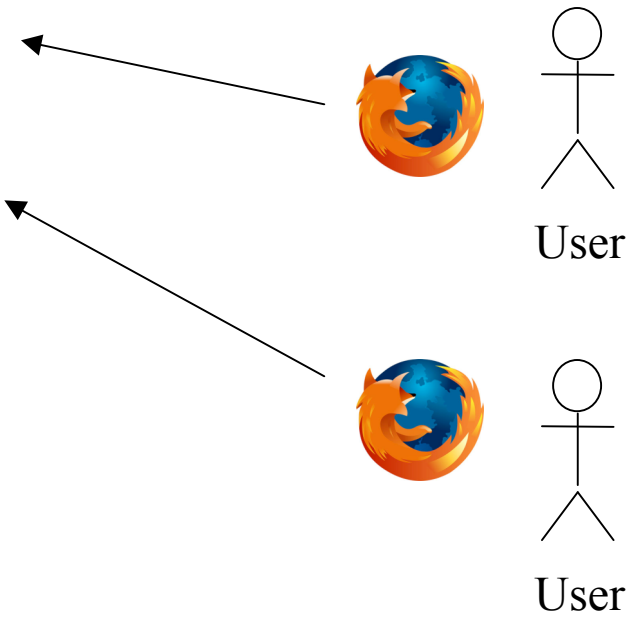
Outline

- Our vision of grid computing
- Introduction to the Opal Toolkit
- Novel contributions
  - Automatic interface generation for Opal
  - Modeling applications as first class resources via CSF4 metascheduler
  - Opal CSF4 integration

Opal CSF4 Integration

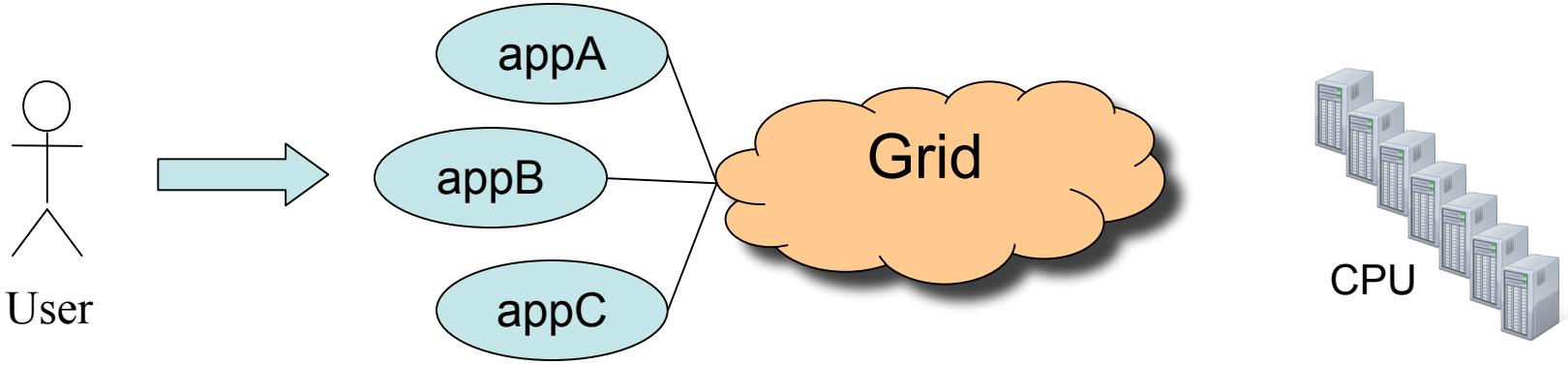
- Opal plug-in for submission to CSF4
- Deployment architecture:


HTTP Server
Opal GUI
WS/SOAP
App A
App B
App C
Opal Server
WSRF
App A
App B
App C
CSF4
(Pre/WS)-GRAM
Globus GRAM
Globus GRAM
ClusterA
appA
appB
…
ClusterB
appA
appC


User
User

Conclusion

- End-users prefer to deal with high-level concepts (applications)
- Web 2.0 (Web as a platform, service oriented, enable light-weight programming models, rich user experience)



Thank you…

…for your attention

- Visit booth #3055 for more information
- Ask for:
  - Luca or Sriram (Opal)
  - Zhaohui (CSF4)