BUILDING APPLICATIONS FROM A WEB SERVICE BASED COMPONENT ARCHITECTURE

D. Gannon, S. Krishnan, L. Fang,
G. Kandaswamy, Y. Simmhan,
A. Slominski

What this talk is about

- How to build secure, reliable applications composed from distributed components and web services.
- A Motivating Application
  - LEAD Project
    - Tools to allow research meteorologists to compose powerful applications that predict mesoscale weather events in better than real time.

Anvil of large cumulonimbus thunderhead during early stages of developing storm. Credit: NOAA Photo Library

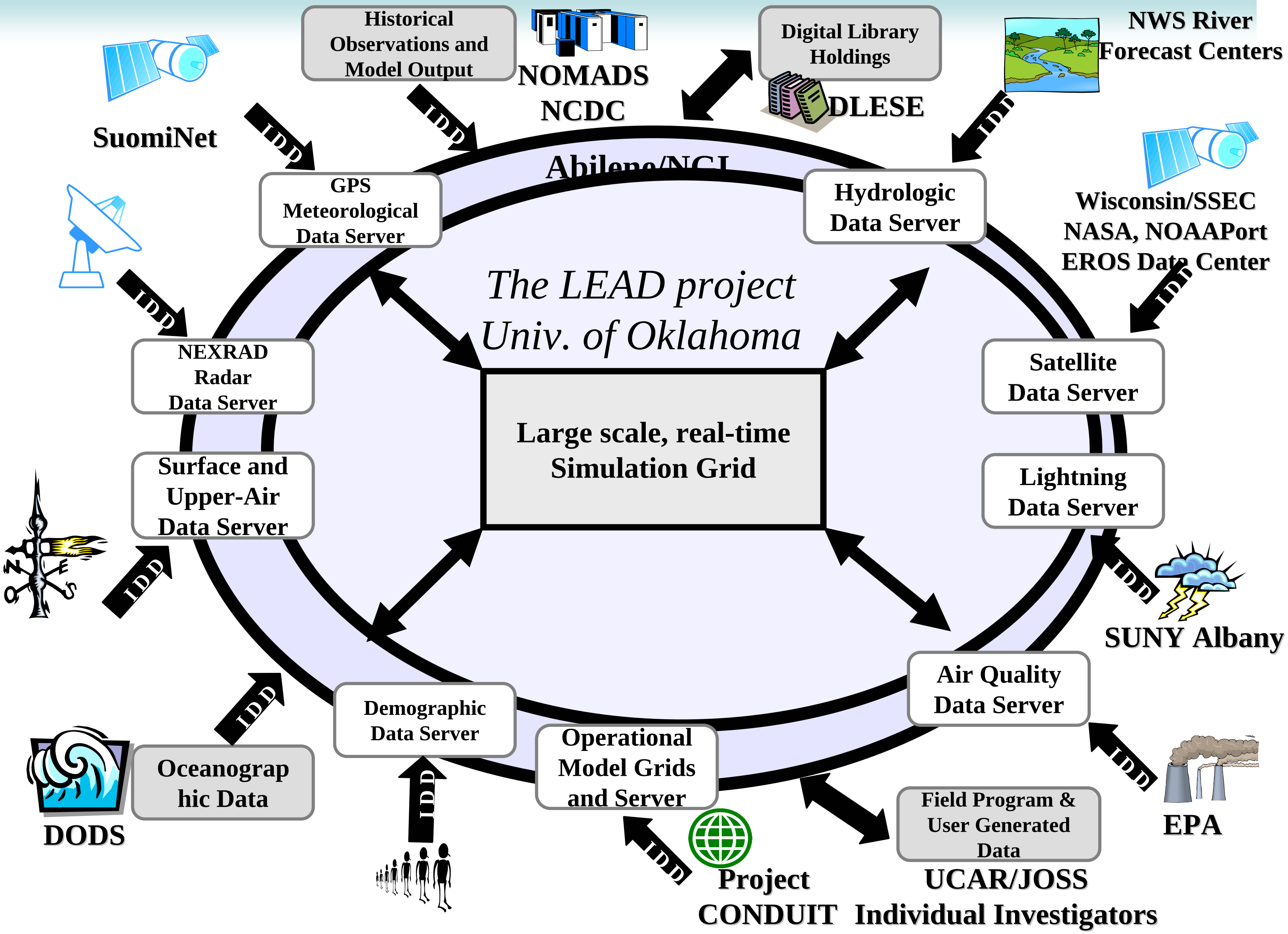
Multiple cloud-to-cloud and cloud-to-ground lightning strokes caught using time-lapse photography during a night-time thunderstorm. Credit: NOAA Photo Library

Seymour, TX
April 10, 1979
Photographer: D. Burgess
Credit: NOAA Photo Library

Predicting Severe Storms

- To deliver better than real-time predictions
  - Data mining of live instrument streams and historical storm metadata
  - Requisition large computational resources on demand to start a large number of simulations
    - Mine simulation outputs to see which track real storm evolution.
    - Refine scenarios that match incoming data.
  - May Need to requisition bandwidth to make the needed data analysis possible.
  - May require real-time re-alignment of instruments.
  - Workflows may run for a long time and they must be adaptive and very dynamic

Predicting Severe Storms



Historical Observations and Model Output

NOMADS
NCDC

Digital Library Holdings

DLESE

NWS River Forecast Centers

SuomiNet

Abilene/NCI

GPS Meteorological Data Server

Hydrologic Data Server

Wisconsin/SSEC
NASA, NOAAPort
EROS Data Center

The LEAD project
Univ. of Oklahoma

NEXRAD Radar Data Server

Large scale, real-time Simulation Grid

Satellite Data Server

Surface and Upper-Air Data Server

Lightning Data Server

SUNY Albany

Air Quality Data Server

Demographic Data Server

Operational Model Grids and Server

Oceanograp hic Data

DODS

Field Program & User Generated Data

EPA

Project CONDUIT

UCAR/JOSS
Individual Investigators

Typical, Very Simple, LEAD Scenario

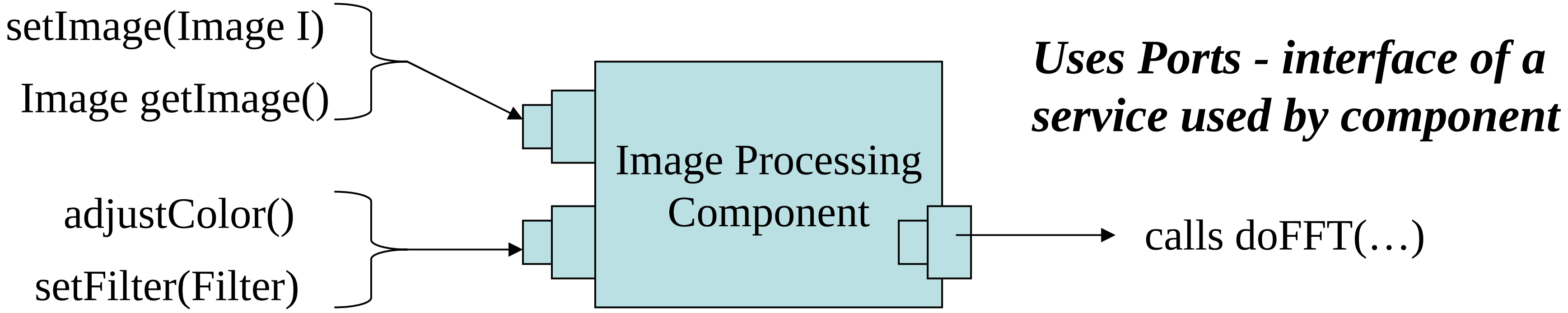
- Search for data, run a simulation and catalog results.
  - Query metadata catalog for a dataset
  - Use the result for a large WRF simulation
  - Allocate storage on a remote resource
  - Move the WRF output to that allocated space
  - Record the output location and computation history in a metadata catalog.
- How does a user describe such a scenario as a workflow or distributed application?
- How do we free the user from details of distributed computing in a service oriented architecture?
- What does a service architecture mean in this context?
- Can it be done by a component composition approach?

Common Comp. Architecture (CCA)

- Started in mid 90s.
  - The Common Component Architecture
  - Four different implementations exist
    - SciRun II
    - Caffene (Sandia)
    - Decaf (Livermore)
    - XCAT (Indiana/Binghamton)
  - A specification for component design for parallel and distributed applications
- A Few words about the architecture and applications

CCA Concepts

- Ports: the public interfaces of a component
  - – defines the different services provided by a component and the ways the component uses other services and components.



Uses Ports - interface of a service used by component

Provides Ports - interfaces functions (services) provided by component

Building Applications by Composition

- Connect uses Ports to Provides Ports.

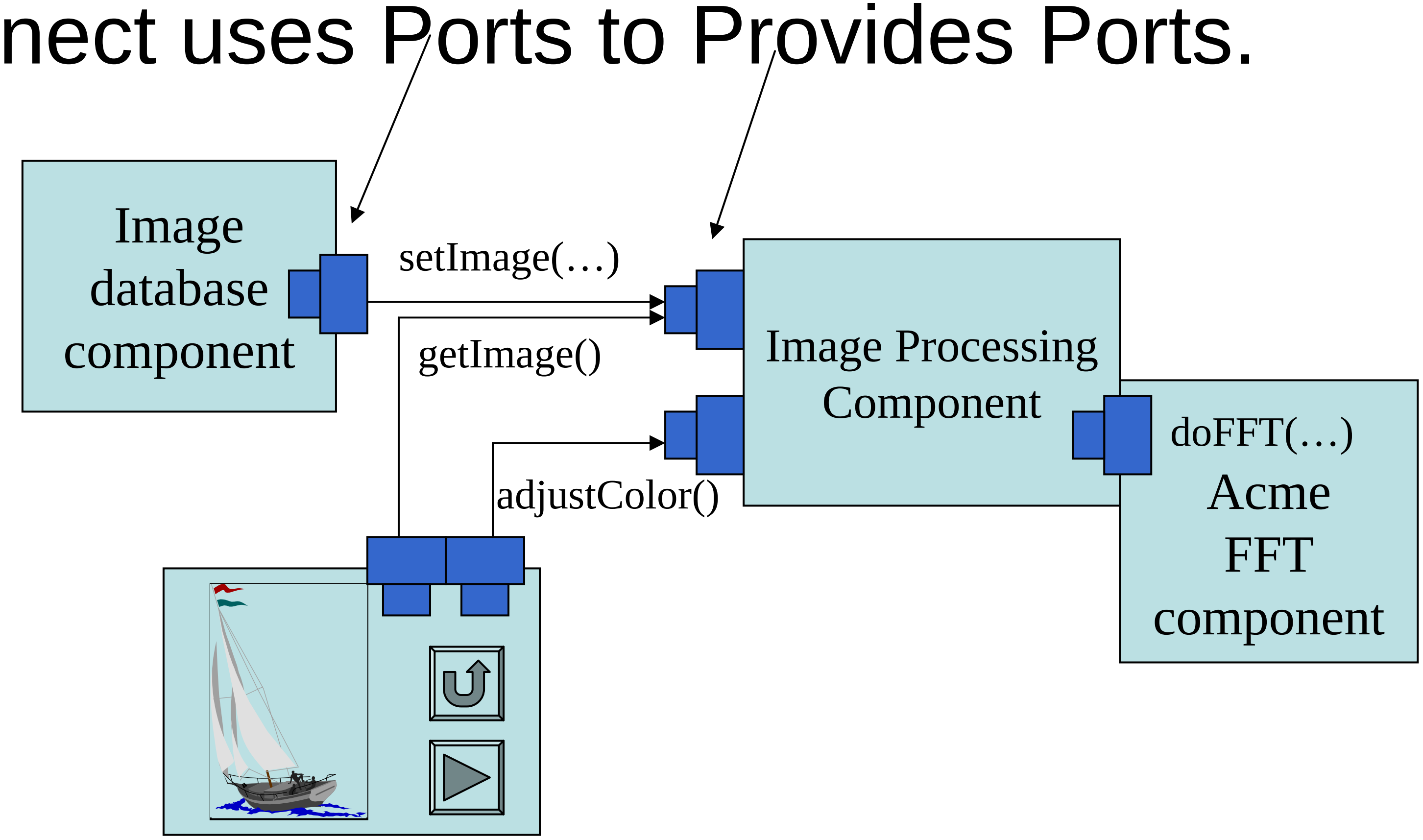
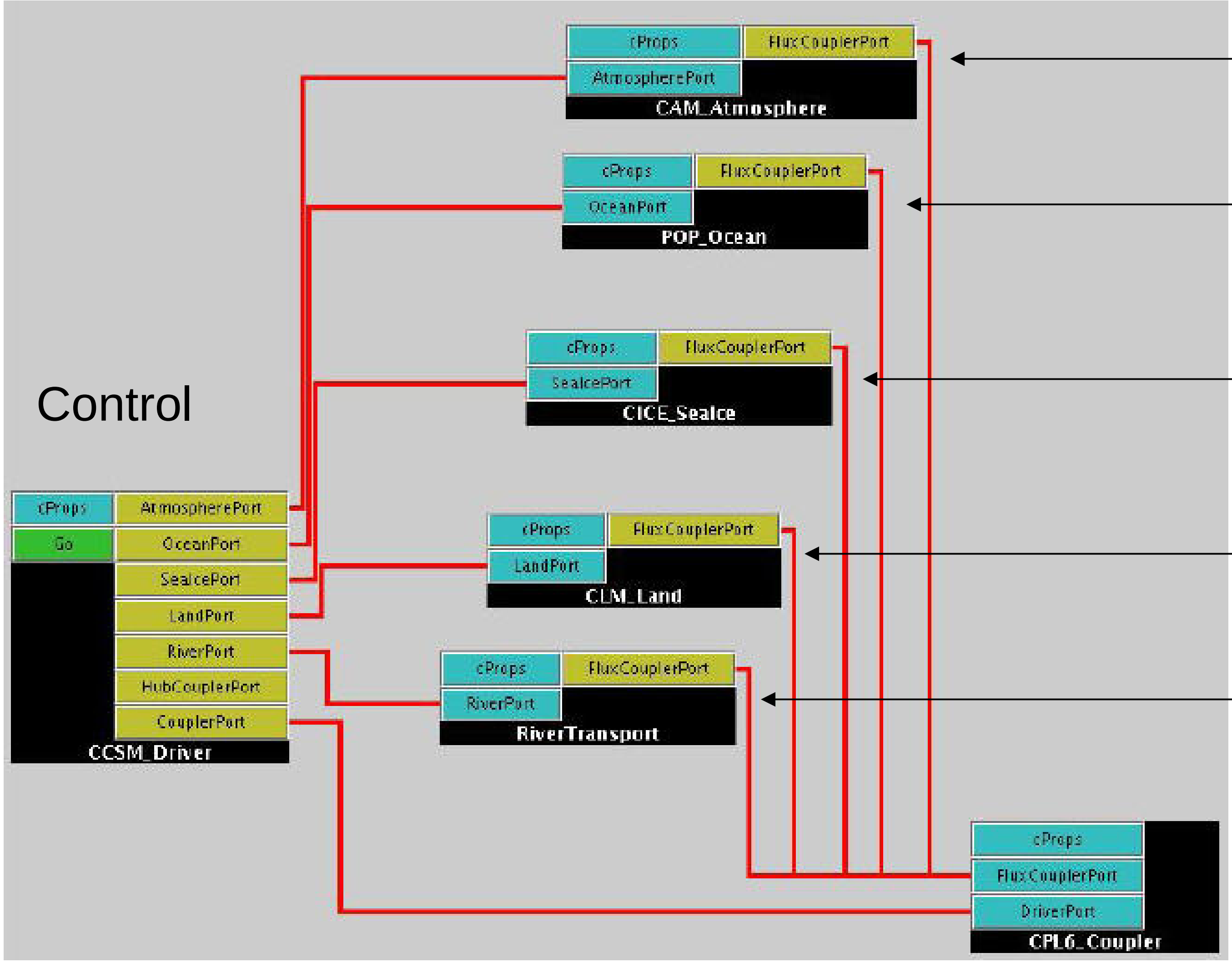


Image tool graphical interface component

Community Climate System Model



Control

Atmosphere

Ocean

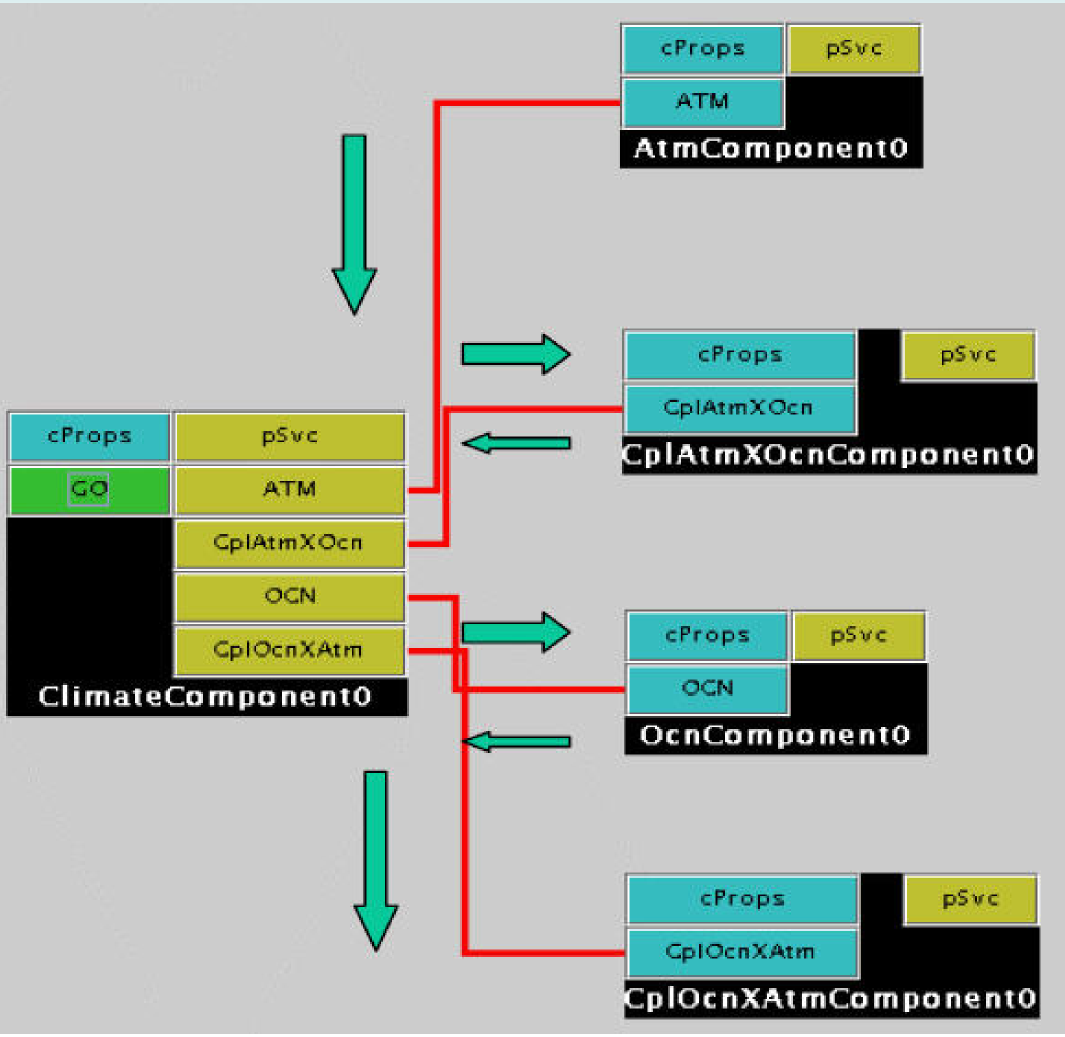
Sea Ice

Land

River

Flux Coupler

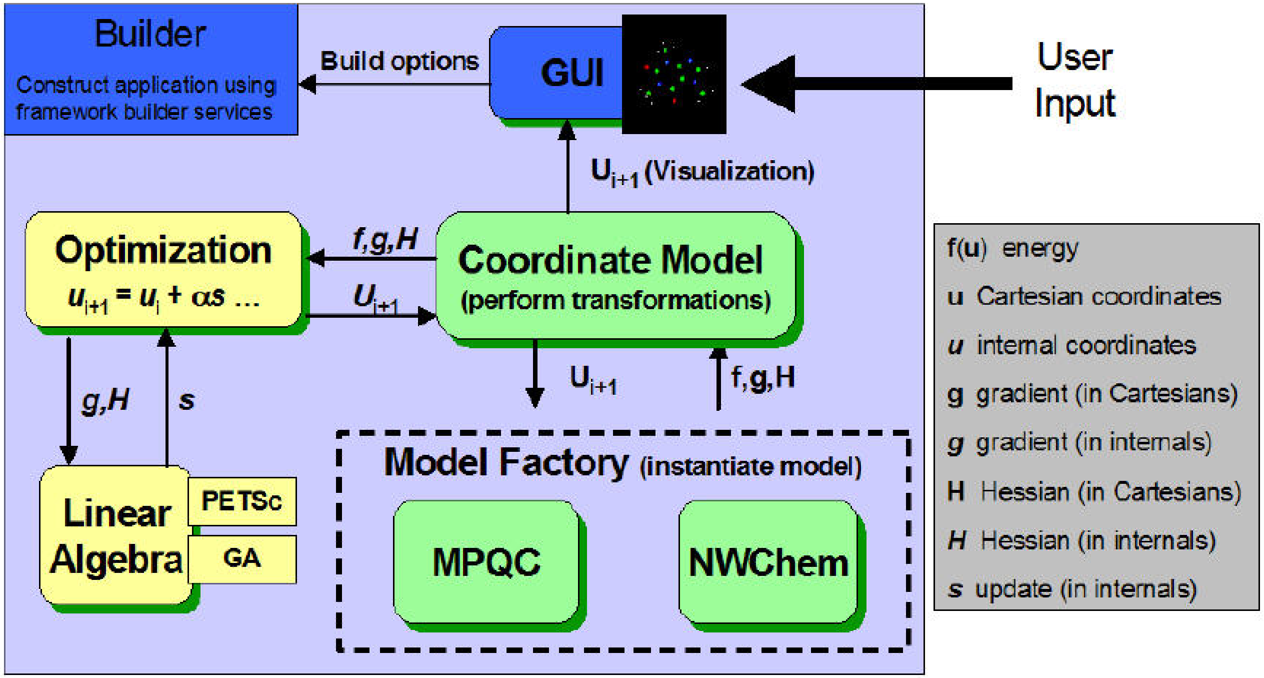
Earth System Modeling Framework CCA Prototype



- The Climate Component is Control
  - Atmosphere Component
  - Ocean Component
  - Atmosphere to Ocean Transformer
  - Ocean to Atmosphere Transformer

Quantum Chemistry

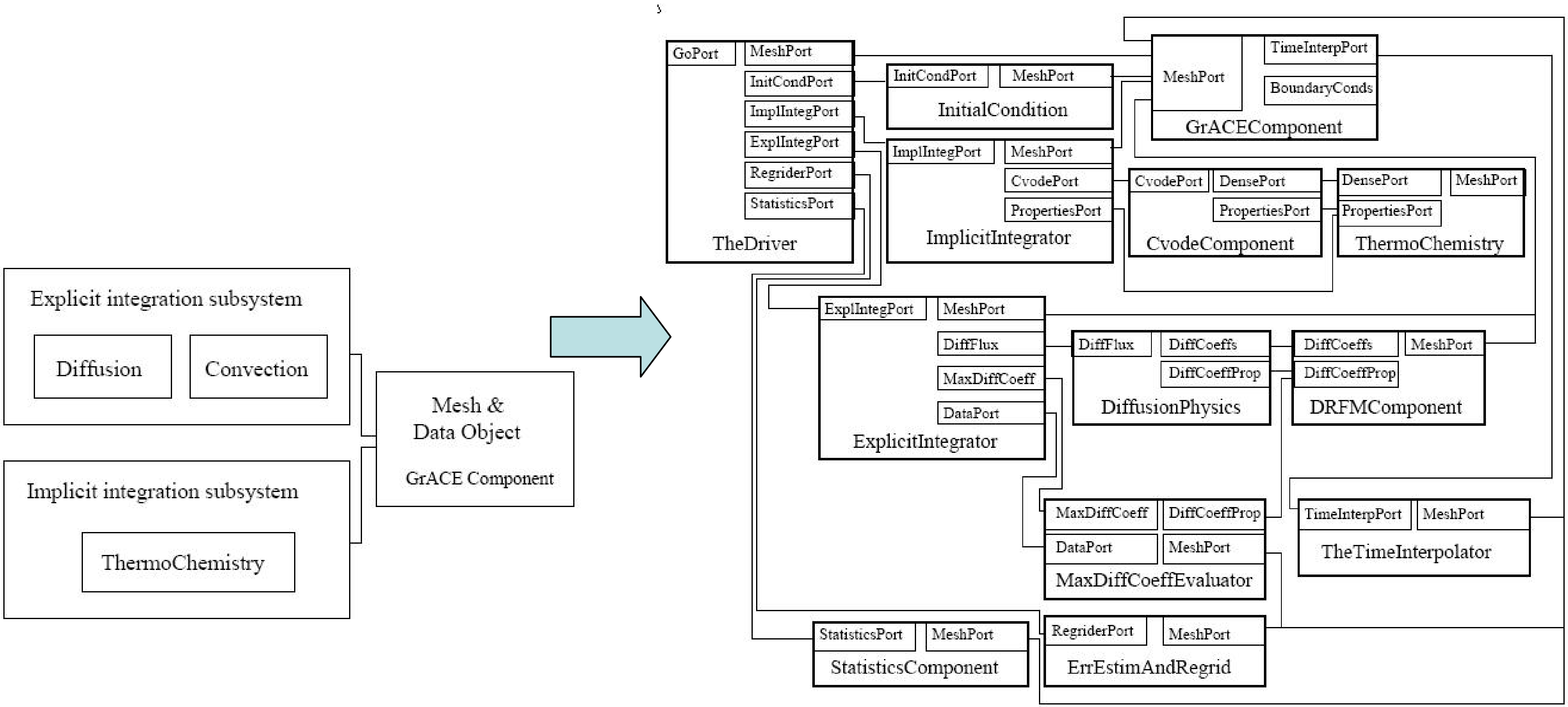
- Design of CCA integration of NWChem and MPQC. (A work in progress)
  - One is Fortran and the other C++
  - One standard interface using CCA/Babel/SIDL.



Builder
Construct application using framework builder services
Build options
GUI
User Input
U_{i+1} (Visualization)
Optimization
$u_{i+1} = u_i + \alpha s$ ...
f,g,H
U_{i+1}
Coordinate Model
(perform transformations)
g,H
s
U_{i+1}
f,g,H
Linear Algebra
PETSc
GA
Model Factory (instantiate model)
MPQC
NWChem
f(u) energy
u Cartesian coordinates
u internal coordinates
g gradient (in Cartesians)
g gradient (in internals)
H Hessian (in Cartesians)
H Hessian (in internals)
s update (in internals)

Combustion Modeling

- The High level model of the system integration is refined into the component composition



Experience with CCA

- The effort to “re-factor” applications can be very difficult.
  - Where are the component boundaries?
  - Who owns large data structures?
    - Make the data structure a component.
  - What are the correct port Interfaces?
    - Can’t interoperate unless they share the same interface type.
  - No quantitative results yet.
- The positive
  - Production codes are just becoming mature.
  - A serious library of components is starting to emerge.
- Now beginning to understand components for distributed apps

CCA components as Web Services

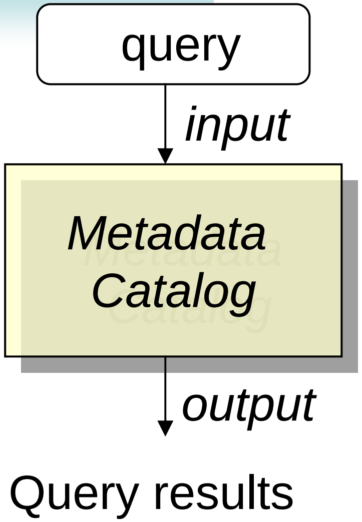
- Each Provide port can be a complete web service
  - Web service with more than one port are not very well defined
- Uses ports become web service “client stubs”.
- Connection is then a binding between a client stub and a provided service.
- XCAT3 implements this feature.
  - Uses python as the scripting language.
- What about using web/grid services as components?

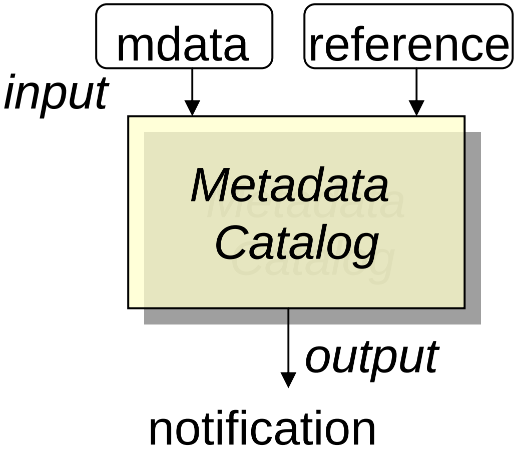
Working with Web Services

- Web Service are not the same as CCA
- Message oriented and not RCP based.
  - Send a message to the service
    - You may get a response or you may not.
      - Depends upon the service semantics.
- No concept of “uses port”.
  - However some serves generate messages in response to messages sent.

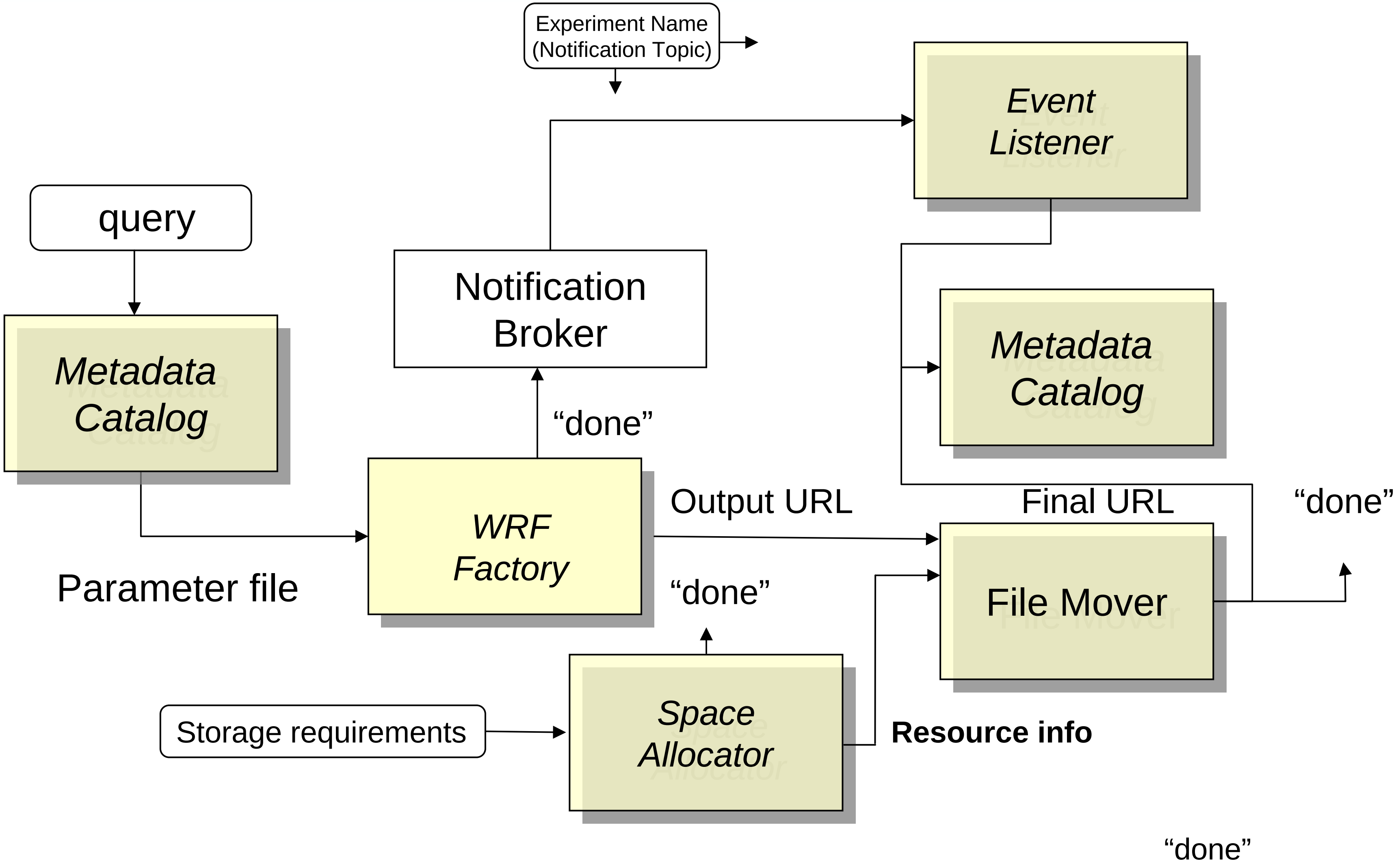
Component Programming with Services

- Services in our example are
  - Metadata catalog
  - Storage Allocator
  - WRF Simulation Engine
  - Execution history recorder
- The services are assumed to be stateless or, if stateful, they are transient.
- Services have input messages and output messages.
  - Each message may have multiple parts
  - An input message may have its parts come from different sources
  - Outputs may be sent to multiple sources
  - A Notification event is often generated as an additional output.



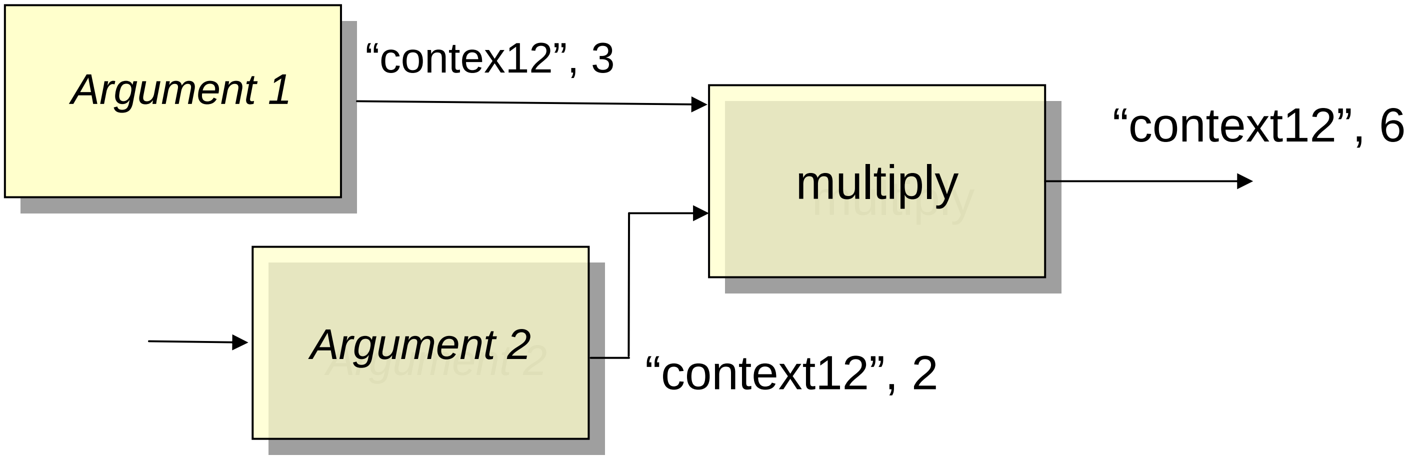


The Workflow



Services, state and context

- How do you manage services with multiple “related clients”?



Questions

- Can we compile such a “graph picture” into a running distributed application?
  - What type of application is it?
    - Workflow? Statically connected distributed components?
  - How is synchronization handled?
  - How is failure managed?
- How and when are specific resources allocated for the computational parts?
- Can the resulting workflow be turned into another service to be used by another?
- What are the security implications?

Several Possible Solutions

- Triana, Kepler, Taverna
  - All excellent examples of tools to compose workflows using graphical tools.
- Each is based on an approach where the workflow engine is based on an interpretation of the execution graph.
- Triana has been extended to incorporate web services by means of a component proxy.
- We need something that is more appropriate for dynamic, long running workflows
  - BPEL4WS is the most powerful workflow language, but it is not very “friendly”.
  - Can we compile graphical specs into a BPEL spec?

A Three Level Design

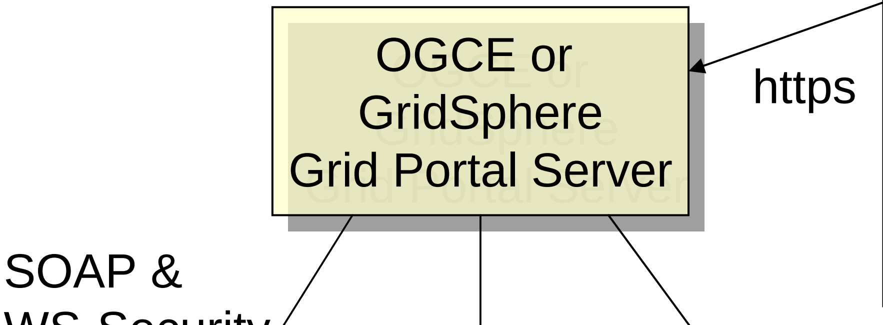
- Front End
  - A Grid portal with tools to build and launch distributed application from remote component services.
    - Allow scientists to compose workflow scenarios
- The Middle
  - Application factories and security services
- The Back End
  - Composing workflow from distributed web service components. Compiles workflow into a BPEL extension we call GPEL.

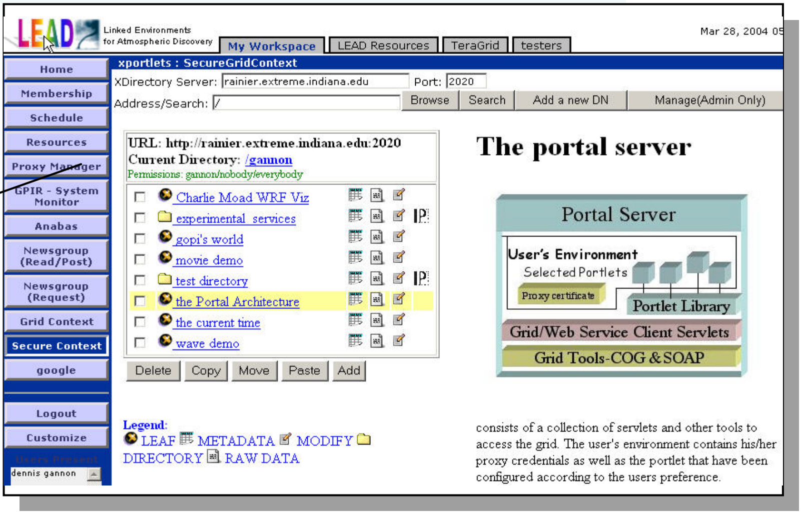
Front End: The Portal

The User’s View of the Grid

The Portal as a Grid Access Point

- The Portal Server provides the users Grid Context.





Mar 28, 2004 05

xportlets : SecureGridContext

XDirectory Server: rainier.extreme.indiana.edu Port: 2020

Address/Search: /

URL: http://rainier.extreme.indiana.edu:2020
Current Directory: /gannon
Permissions: gannon/nobody/everybody

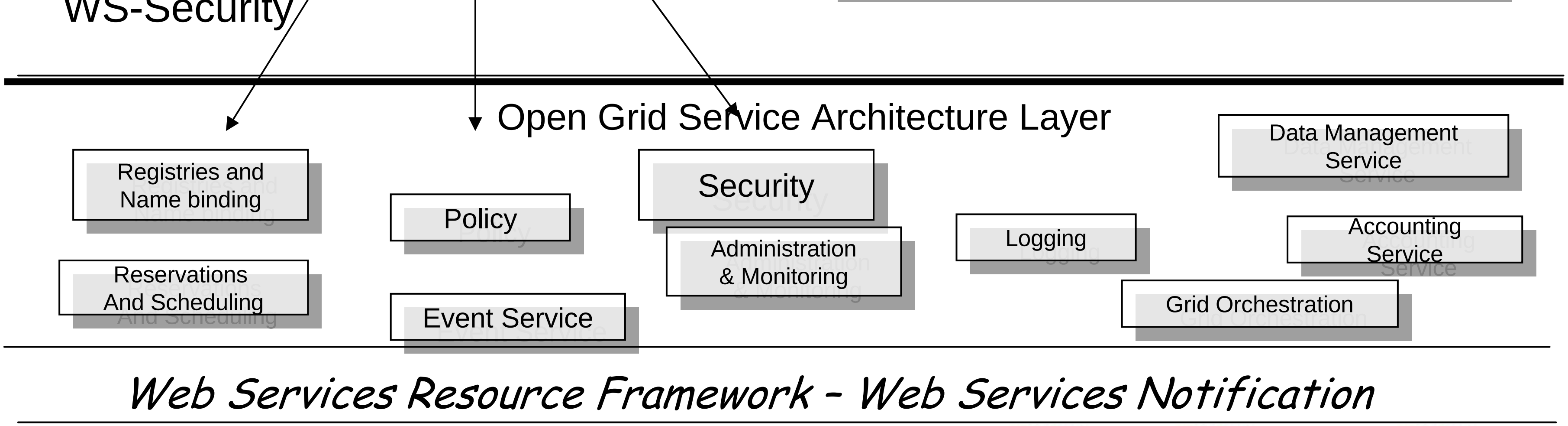
- □ Charlie Moad WRF Viz
- □ experimental_services
- □ gopi's world
- □ movie demo
- □ test directory
- □ the Portal Architecture
- □ the current time
- □ wave demo

Legend: LEAF METADATA MODIFY DIRECTORY RAW DATA

dennis gannon

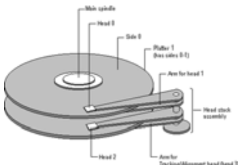
The portal server

consists of a collection of servlets and other tools to access the grid. The user's environment contains his/her proxy credentials as well as the portlet that have been configured according to the users preference.



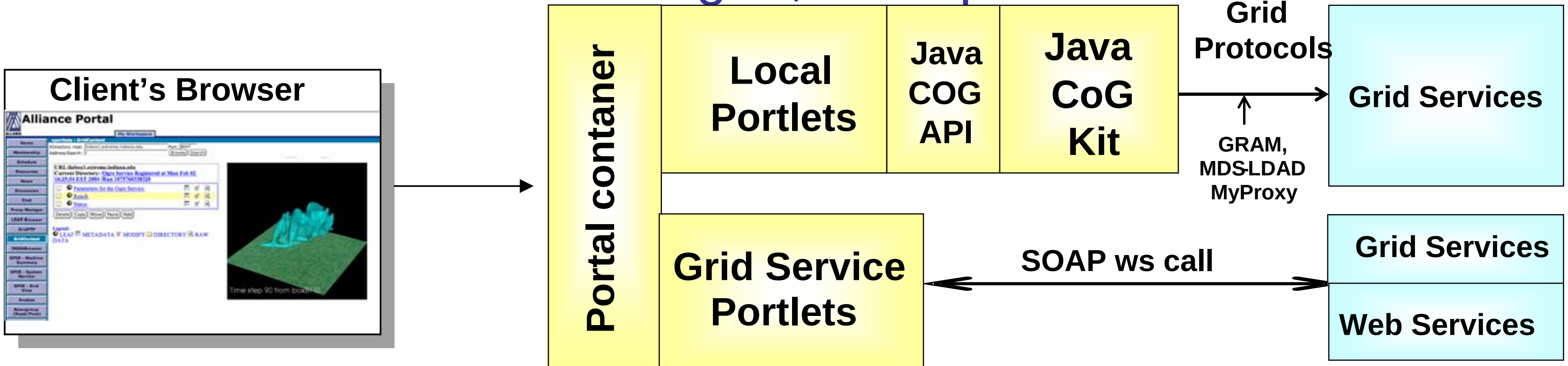
Open Grid Service Architecture Layer
Registries and Name binding
Policy
Security
Data Management Service
Reservations And Scheduling
Administration & Monitoring
Logging
Accounting Service
Event Service
Grid Orchestration
Web Services Resource Framework – Web Services Notification

Physical Resource Layer



Portal Architecture

- Building on Standard Technologies
  - Portlet Design (JSR-168) IBM, Oracle, Sun, BEA, Apache
  - Grid standards: Java CoG, Web/Grid Services
- User configurable, Service Oriented
- Based on Portlet Design
  - A portlet is a component within the portal that provides the interface between the user and some service
  - Portlets can be exchanged, interoperate



Client's Browser

Alliance Portal

Portal contaner

Local Portlets

Java COG API

Java CoG Kit

Grid Protocols

GRAM, MDS-LDAD MyProxy

Grid Services

Grid Service Portlets

SOAP ws call

Grid Services

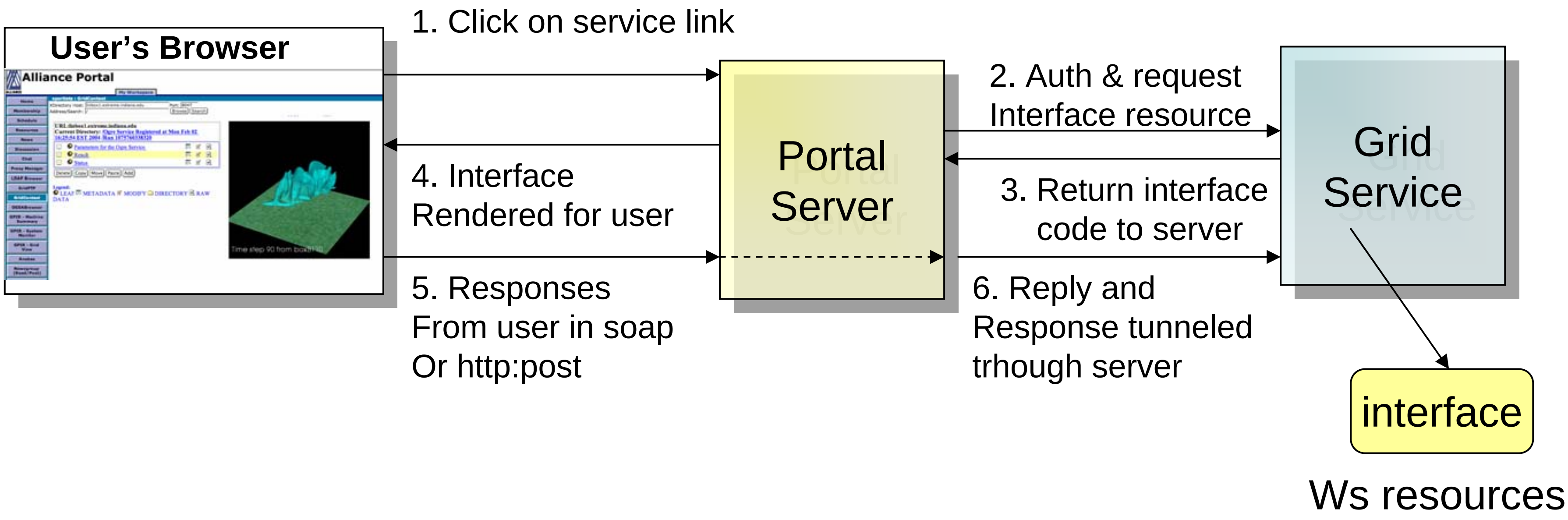
Web Services

The Middle Tier: Making Applications into Services Visible to the Portal & User

How do users interact with Grid Services?
How do we maintain reliable Grid Services?
What is the security model?

User-Portal-Service Interaction

- Grid Services with user interfaces are mediated by the portal.
  - The Grid service can keep an interface client to itself as a WS resource which can be loaded by the server and presented to the client.
  - Allows security to be https from browser and ws-security from portal server to Grid service.



1. Click on service link

User's Browser

Alliance Portal

Portal Server

2. Auth & request Interface resource

Grid Service

3. Return interface code to server

4. Interface Rendered for user

5. Responses From user in soap Or http:post

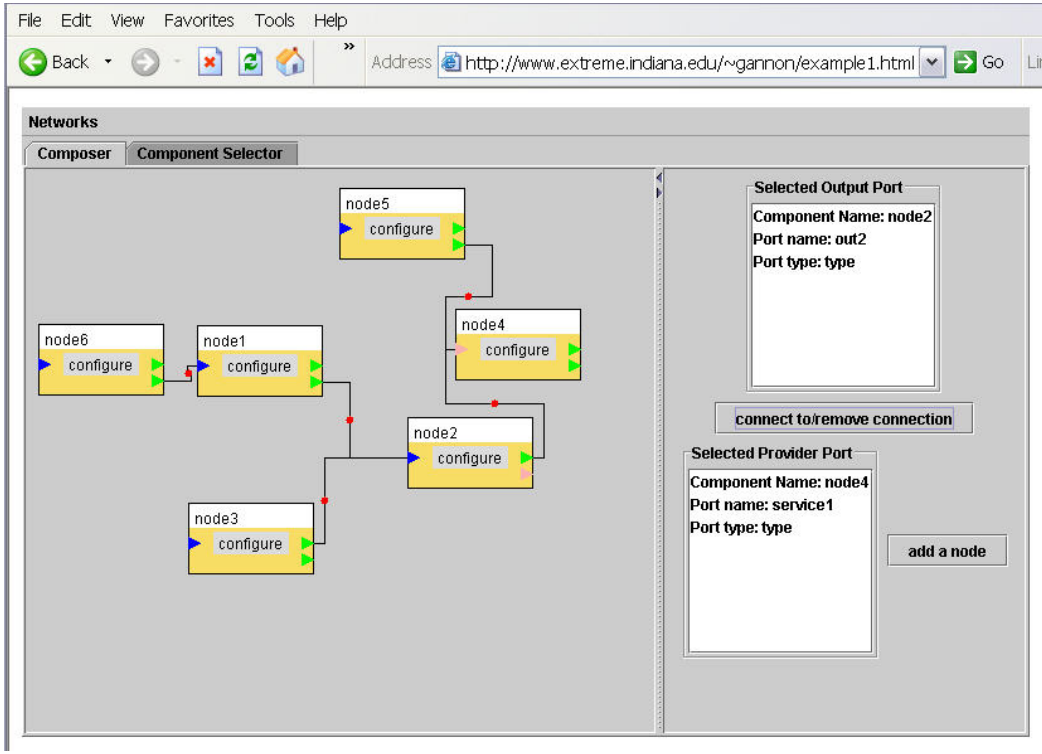
6. Reply and Response tunneled trhough server

interface

Ws resources

For example: Component Composer

- An interactive workflow composer.
  - Component database and workflow compiler is provided by the grid service
    - which also provides the interface tool.
  - MVC pattern.
  - Composer allows
    - Component selection from library
    - Drop and drag placement and connection establishment
    - Save and load graph functions.



http://www.extreme.indiana.edu/~gannon/example1.html

Selected Output Port

Component Name: node2
Port name: out2
Port type: type

Selected Provider Port

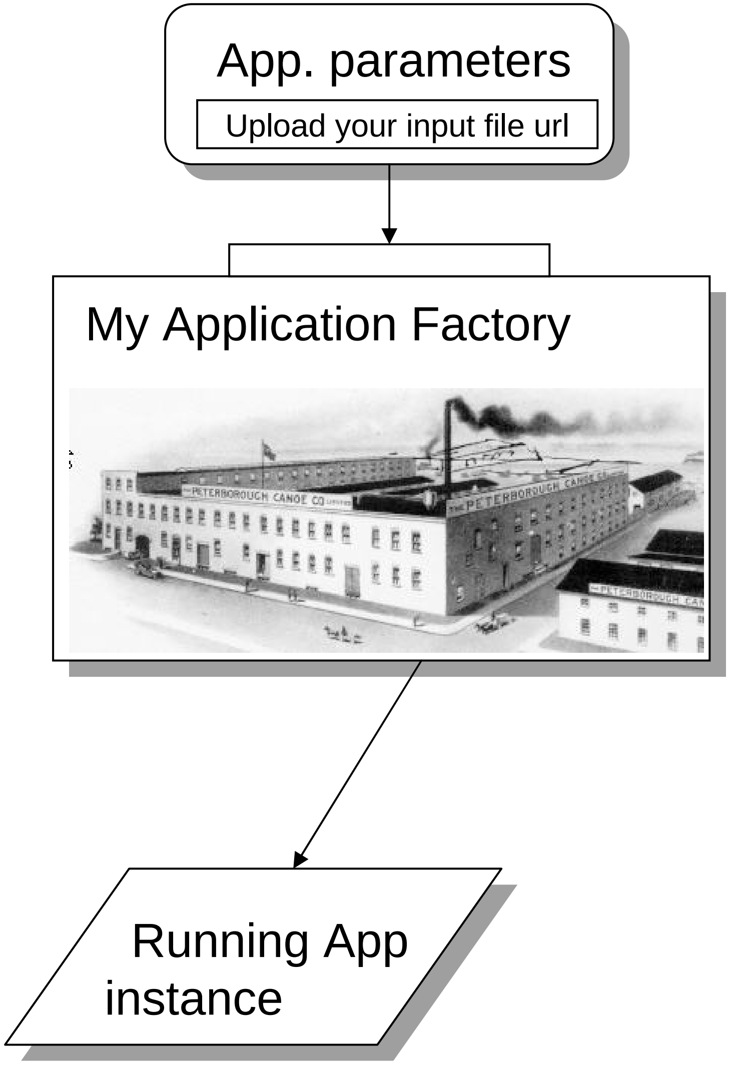
Component Name: node4
Port name: service1
Port type: type

The Most Common Question

- How can I turn my entire application into a component or grid service?
- I want to provide my application as a service for others to use.
  - But I don’t want too many others to use it.
  - I can’t get my friends accounts?

Wrapping Science Apps as Services

- The Factory Pattern
  - A Factory is a web service that creates a running instance of an application for authorized users.
  - A factory client allows app user to:
    - Specify needed input files and other parameters
    - Indicate choice among known execution hosts where app is deployed.



App. parameters

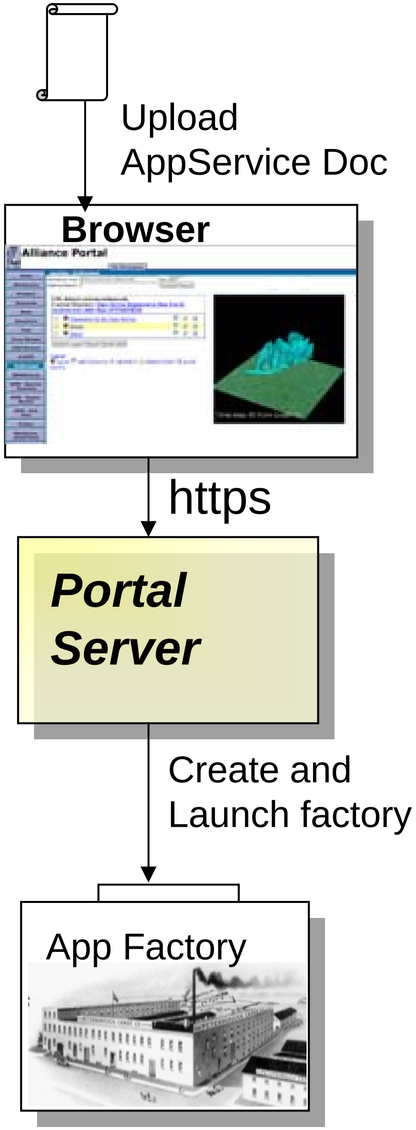
Upload your input file url

My Application Factory

Running App instance

The Portal Factory Service Generator

- Start with
  - A Deployed Application
    - A script to run it.
    - A list of all needed input files
    - A list of all generated output files.
- Write a AppService Document
  - Upload this to the portal Factory generator in the portal.
- A new Factory is started for you.
  - A portal client interface to the factory is also automatically generated.



Upload AppService Doc

Browser

https

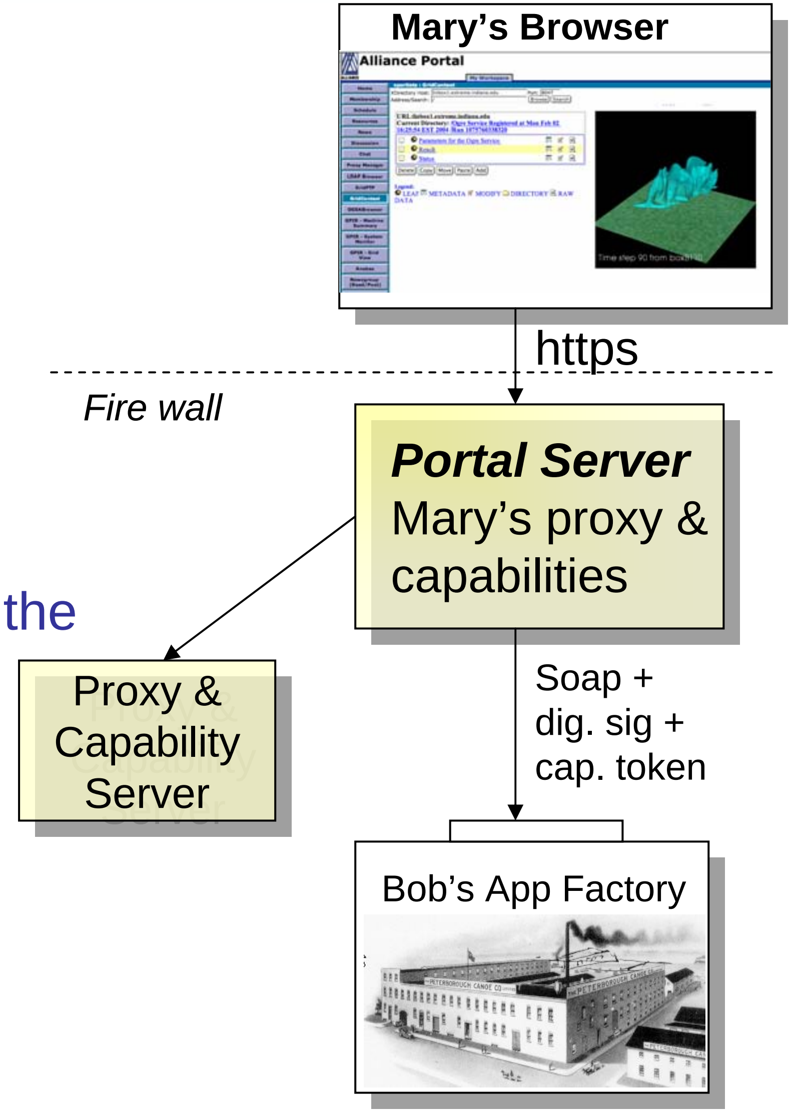
Portal Server

Create and Launch factory

App Factory

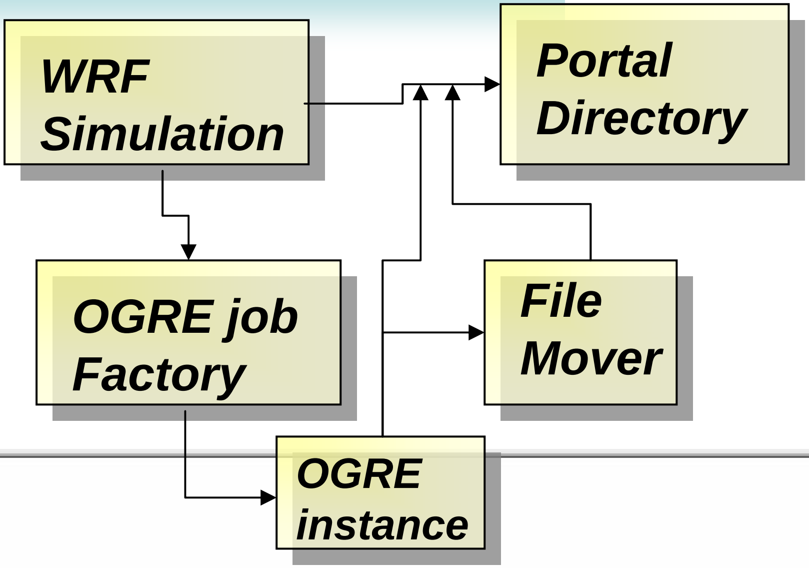
The Security Model

- The parties:
  - The service provider
    - Usually the application scientist in charge of the app.
  - The user
    - Usually an associate or client of the provider.
    - Is provided a capability token by the provider to run the application.
- The capability token
  - An xml document (SAML) signed by the provider that says the user has permission to access the service.
- The factory service
  - Only accepts requests signed by the user and containing the required capability token.

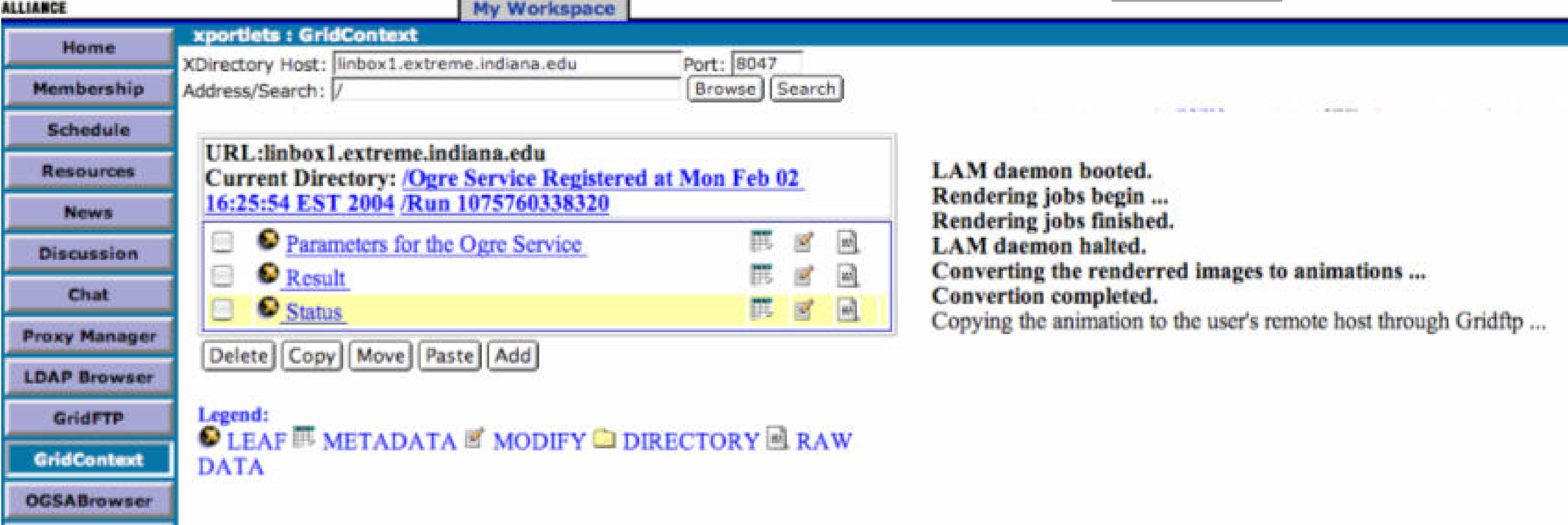


Example: Rendering a Storm

- Take WRF output,
- move it to a cluster,
- Launch an “OGRE” script to render it,
- move movie to users directory
- (Work by C. Moad, B. Plale, G. Kandaswami, L. Fang)



Alliance Portal



xportlets : GridContext

XDirectory Host: linbox1.extreme.indiana.edu Port: 8047

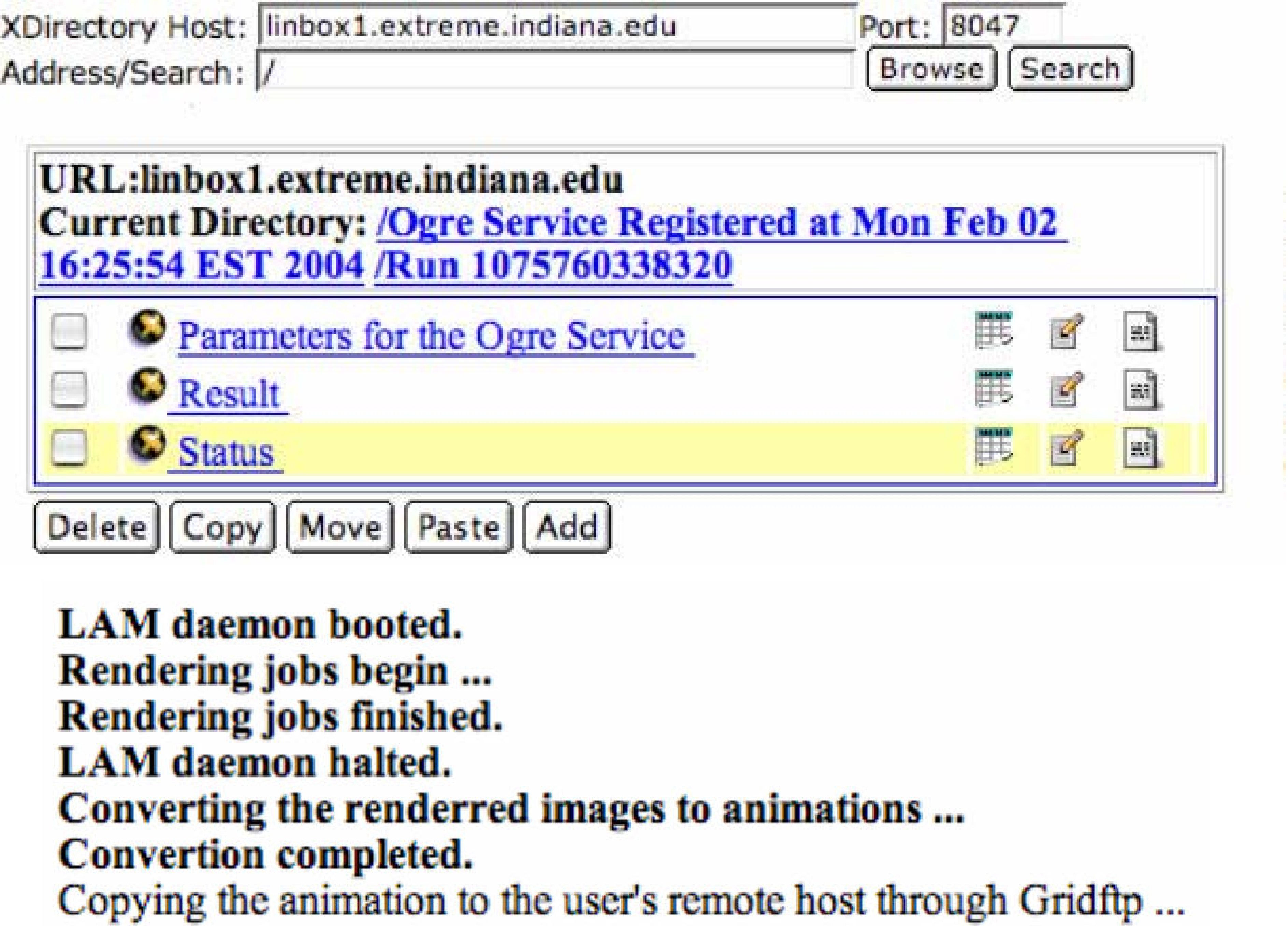
Address/Search: /

URL:linbox1.extreme.indiana.edu
Current Directory: /Ogre Service Registered at Mon Feb 02 16:25:54 EST 2004 /Run 1075760338320

□ Parameters for the Ogre Service
□ Result
□ Status

```
LAM daemon booted.
Rendering jobs begin ...
Rendering jobs finished.
LAM daemon halted.
Converting the renderred images to animations ...
Convertion completed.
Copying the animation to the user's remote host through Gridftp ...
```

Legend:
LEAF METADATA MODIFY DIRECTORY RAW DATA



XDirectory Host: linbox1.extreme.indiana.edu Port: 8047

Address/Search: /

URL:linbox1.extreme.indiana.edu

Current Directory: /Ogre Service Registered at Mon Feb 02 16:25:54 EST 2004 /Run 1075760338320

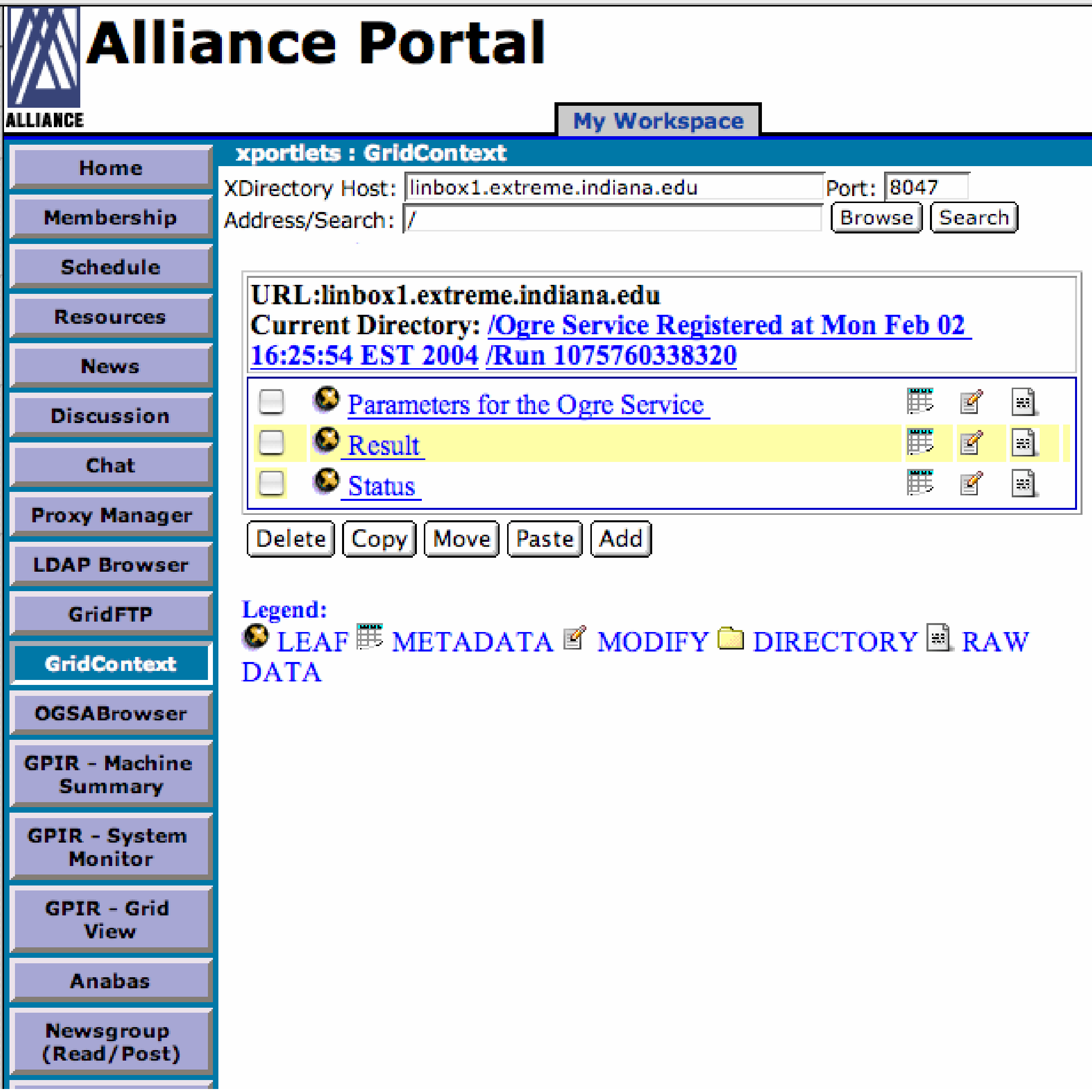
□ Parameters for the Ogre Service

□ Result

□ Status

LAM daemon booted.
Rendering jobs begin ...
Rendering jobs finished.
LAM daemon halted.
Converting the renderred images to animations ...
Convertion completed.
Copying the animation to the user's remote host through Gridftp ...

View the Results



xportlets : GridContext

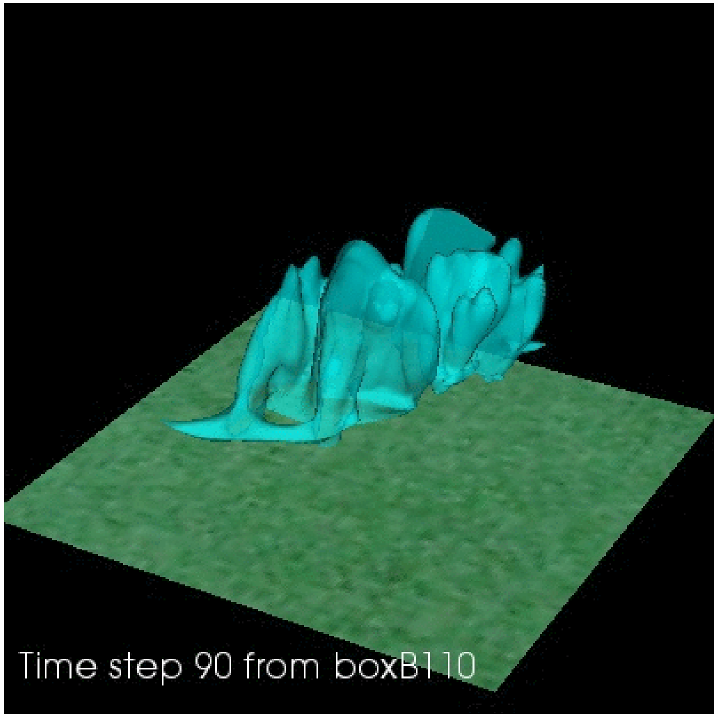
XDirectory Host: linbox1.extreme.indiana.edu Port: 8047

Address/Search: /

URL:linbox1.extreme.indiana.edu
Current Directory: /Ogre Service Registered at Mon Feb 02 16:25:54 EST 2004 /Run 1075760338320

- □ Parameters for the Ogre Service
- □ Result
- □ Status

Legend:
LEAF METADATA MODIFY DIRECTORY RAW DATA



Component Models

- Frameworks used here
  - Portlets – the component model for the Portal
    - JSR 168 industry standard
  - CCA – Common Component Model
    - XCAT3 distributed computing version
  - Web Services composed by GPEL
    - GPEL is a grid version of BPEL4WS by Alek Slominski

Conclusions

- It is possible to integrate the CCA model with web/Grid services.
    - Each cca provides port is a web service (OGSA)
    - Web services are cca components \
        - Either one provides port with returned values, or
        - One input provides port and one output uses port.
        - Notification an important standard “uses port”.
- Security must be built-in from day one.
    - It is the single most difficult part of making this work.
- It is possible to wrap legacy applications as a web-service-based component using a factory pattern.
- Interoperability between component frameworks is an important goal. Web service standards help.