

# Welcome

## Deploying the Large Scale Air Pollution Model as a Grid Service

**Cihan Sahin, Ashish Thandavan,  
Vassil Alexandrov**

Centre for Advanced Computing and Emerging Technologies  
(ACET)

School of Systems Engineering  
University of Reading, UK

# Overview

- Who are we?
- OGSA Testbed Project
- Danish Eulerian Air Pollution Model (DEAP Model)
- Porting DEAP Model to our IBM resource
- System Design
- System Components
- Conclusions & Future Work

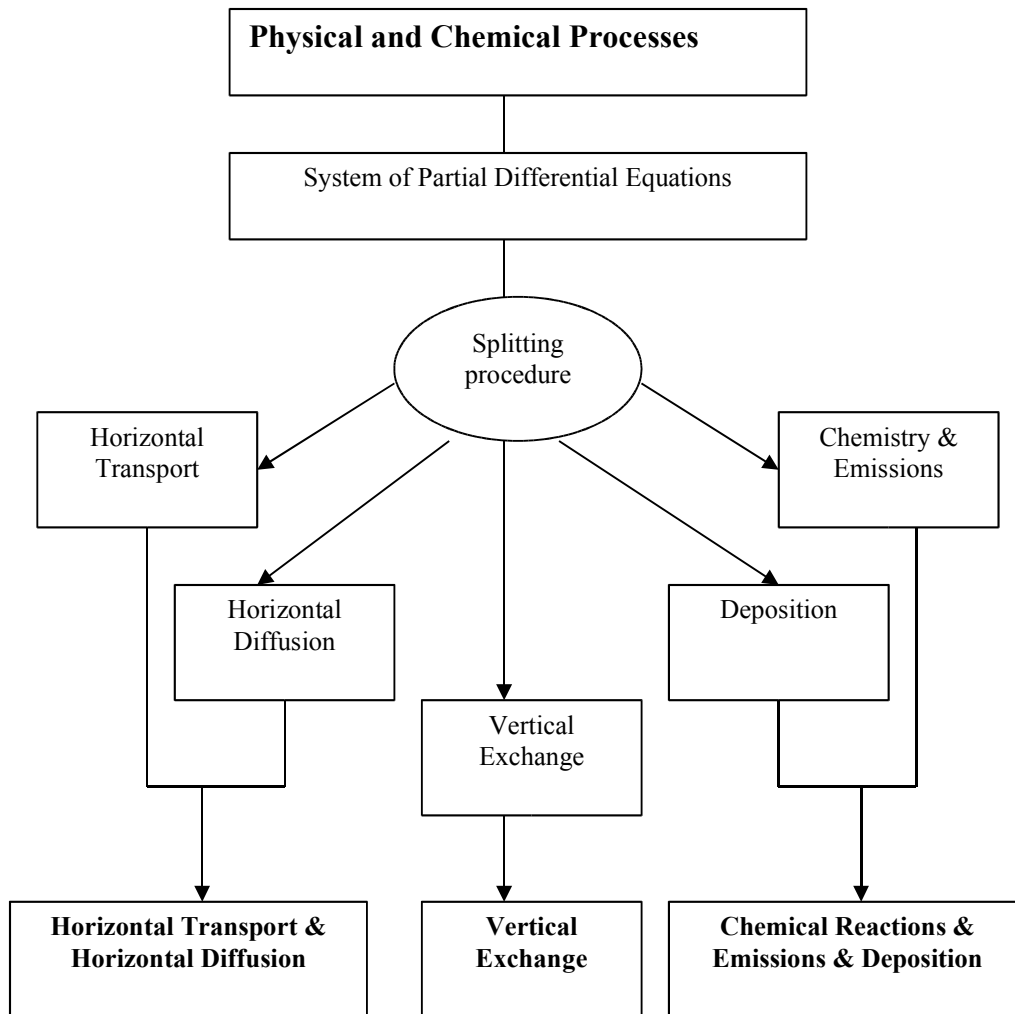
# Who are we ?

- The Centre for Advanced Computing and Emerging Technologies (ACET) located at School of Systems Engineering, University of Reading.
- One of ACET's computing resources consists of a four-node IBM p-655 cluster. Each node has eight 1.5 GHz POWER4 processors with 16 GB RAM and 72 GB internal storage.
- OS is AIX 5.2
- IBM's LoadLeveler and Parallel Operating Environment

# OGSA Testbed Project

- One year e-Science testbed project funded by EPSRC
- Install and evaluate middleware based on OGSI on partners' resources forming a UK-wide testbed
- Document experiences with deploying applications on this testbed.
- Partners – Universities of Portsmouth, Westminster, Manchester, Southampton & Reading, Daresbury Laboratories and MTA-SZTAKI
- The University of Reading's contribution – the IBM pSeries cluster and the DEAP Model

# DEAP Model (Physics)



# DEAP Model Details

- Space domain is 4800x4800 km, Europe at the centre.
- Coarse resolution: 96x96 grids, Fine resolution: 480x480 grids.
- 2-D: One vertical layer, 3-D: 10 vertical layers.
- Time step: 900 sec for transport sub-models & 150sec for chemical sub-model.
- Parallelisation: MPI and OpenMP
- A configuration file “initial\_input” defines number of grids on X,Y,Z coordinates and the location of the input data.

# Porting DEAP Model to IBM resource –I

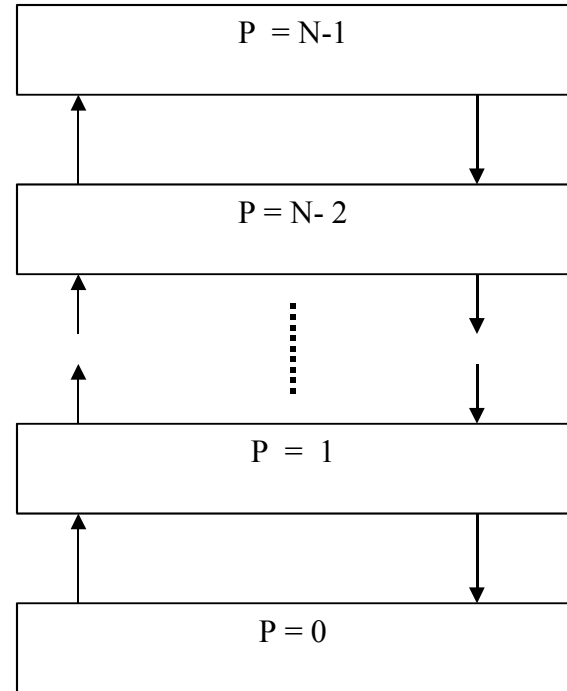
- MPI deadlock problem

## BEFORE (Sun cluster)

```
call MPI_WAITALL(4, request, status, ierror)
    if(ierror.ne.MPI_SUCCESS) then
        ierror = -13
        return
    endif
```

## AFTER (IBM resource)

```
if (nproc > 1) then
    bottom(1)=request(1)
    bottom(2)=request(2)
    top(1)=request(3)
    top(2)=request(4)
    if (me .eq. 0) then
        call MPI_WAITALL(2, bottom, status, ierror)
    else if (me .eq. nproc-1) then
        call MPI_WAITALL(2, top, status, ierror)
    else
        call MPI_WAITALL(4, request, status, ierror)
    endif
    if(ierror.ne.MPI_SUCCESS) then
        ierror = -13
        return
    endif
endif
```



# Porting DEAP Model to IBM resource –II

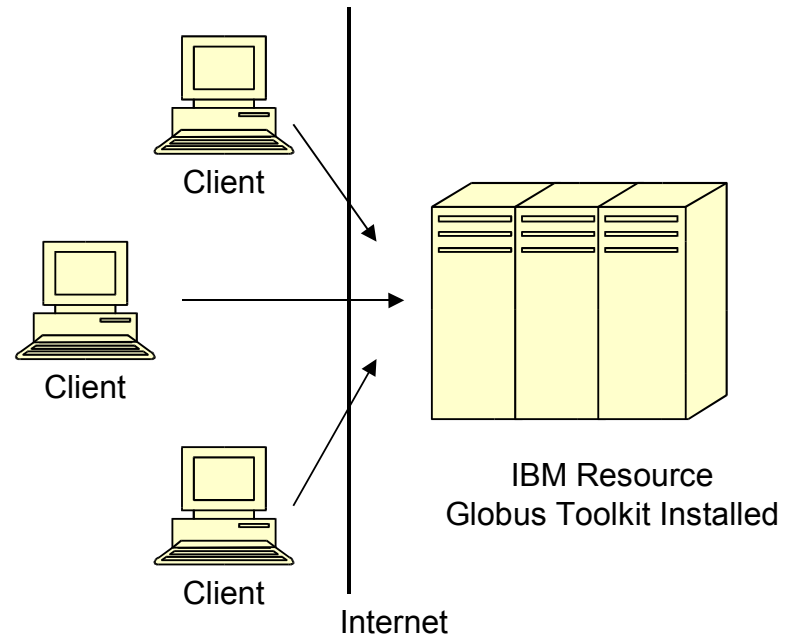
- DEAP Model runs on our IBM resource and scales almost linearly.

SUB-MODEL	N=1	N=2	N=4	N=8	N=16
WIND+SINKS	120.6	61.5	34.4	18.1	19.8
ADVECTION	1362.5	654.1	317.8	165.7	84.9
CHEMISTRY	5884	2971.2	1483.7	778.1	411.4
OUTSO2	375.9	305.2	280.2	319.3	343.2
COMM	0.08	51.9	45.1	105.1	283.9
TOTAL	7749.4	4051	2168.2	1388.5	1143.2

*Run results for 2-D coarse grain DEAP model over a year.*

# Ideal Case Design

- Globus Toolkit v3 is installed on the IBM resource.
- Clients invoke the grid service deployed on the IBM resource.
- The grid service invokes WS-GRAM service which interfaces with Loadleveler.

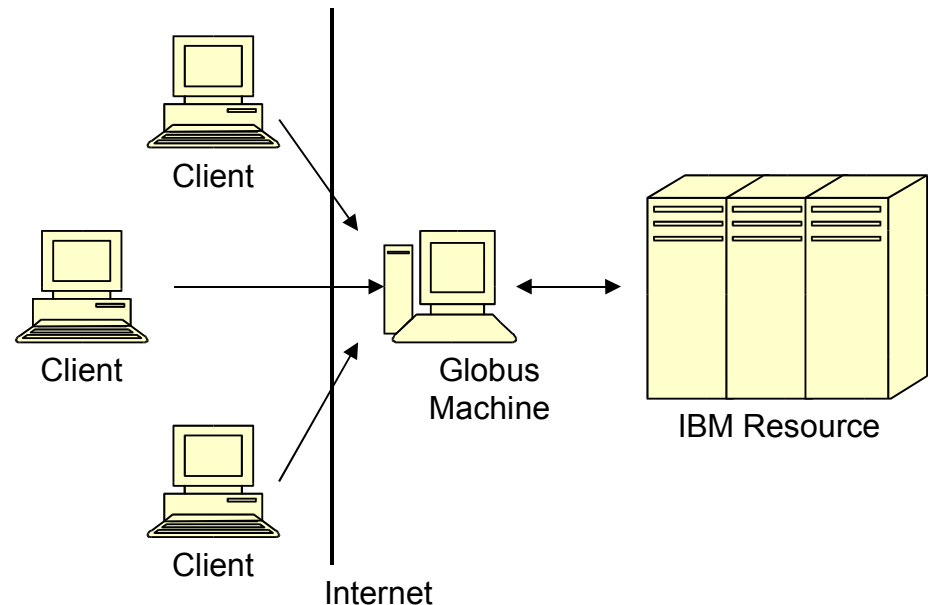


**BUT**

**Globus Toolkit v3 could not be installed!**

# Alternative Design

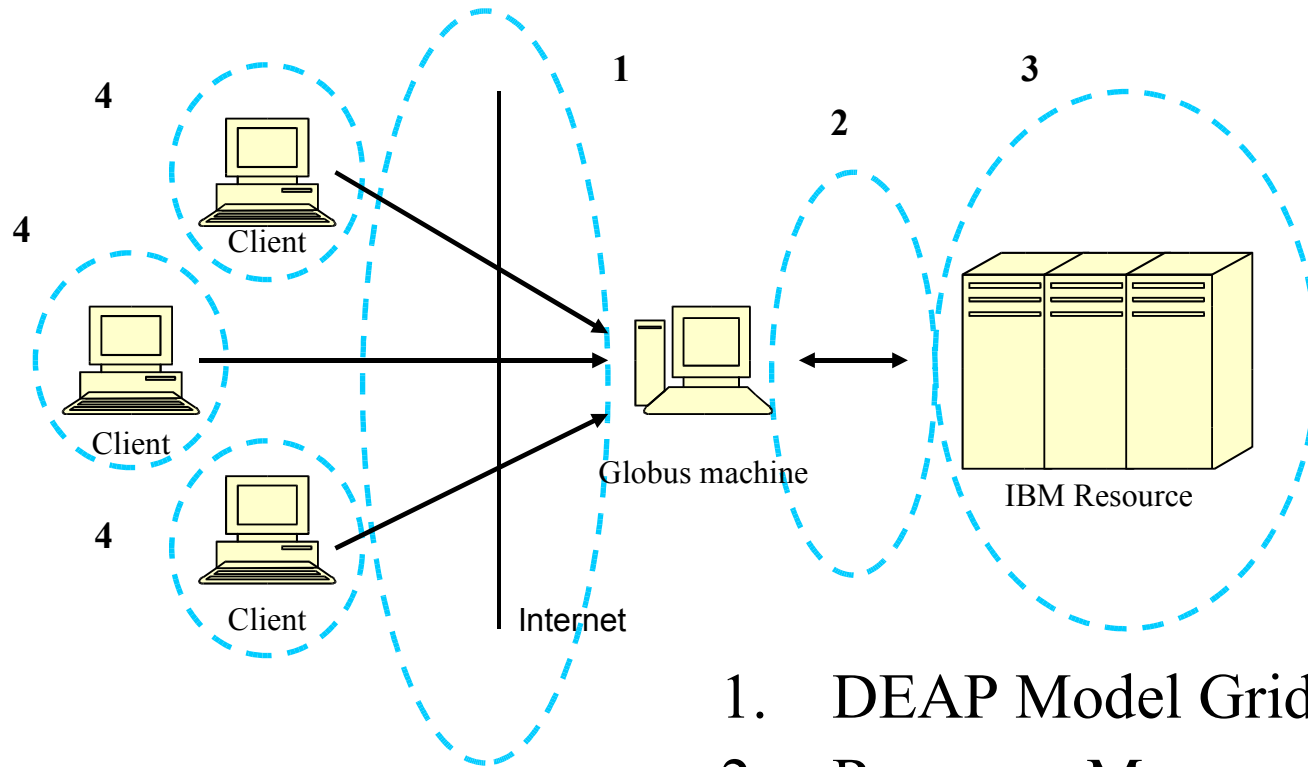
- The Globus Toolkit v3 is installed on a machine between clients and the IBM resource.
- Globus machine handles essential functionality like security.



## What needs to be developed ?

- A resource management mechanism
- A grid service deployed on the globus machine

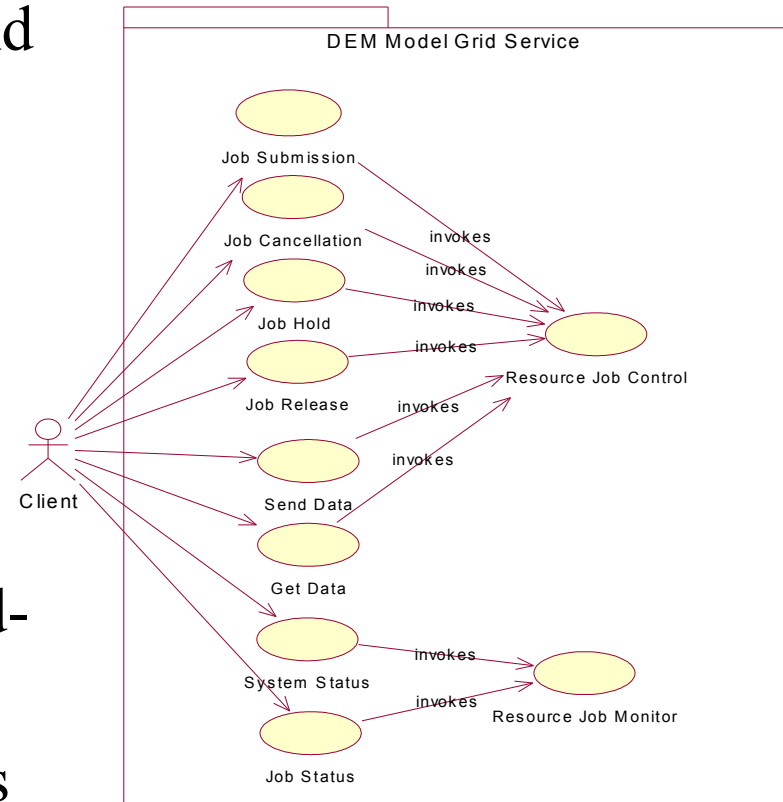
# Alternative Design Components



1. DEAP Model Grid Service
2. Resource Management
3. Interface to Loadleveler
4. Client Application

# DEAP Model Grid Service

- Remote method specifications and implementation
- Submitted jobs' status retrieval (use of notifications)
- Authentication of the users by proxy certificates
- Authorization of the users by mapping to local usernames (grid-map file)
- Message level security. All RMIs are encrypted.



# Resource Management

- Client / Server pairs for job control and job status retrieval.

*A* : Job Control Client

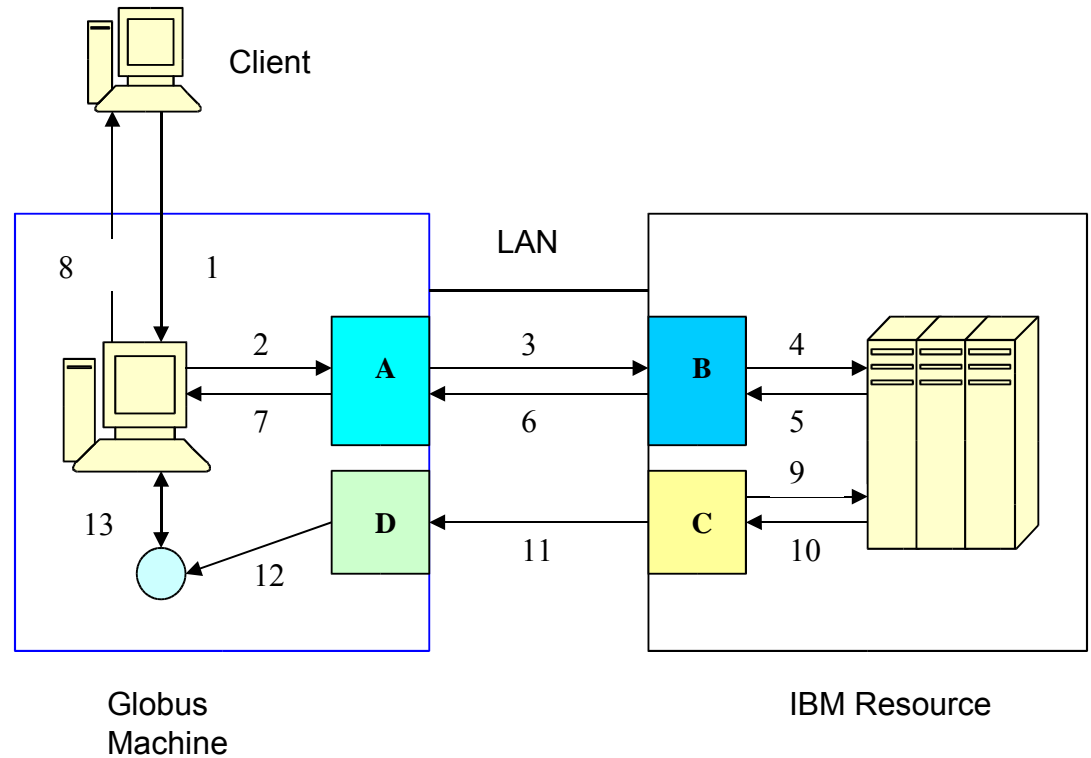
*B* : Job Control Server

*C* : Job Status Client

*D* : Job Status Server

2-7 : Represents a job control data / message flow.

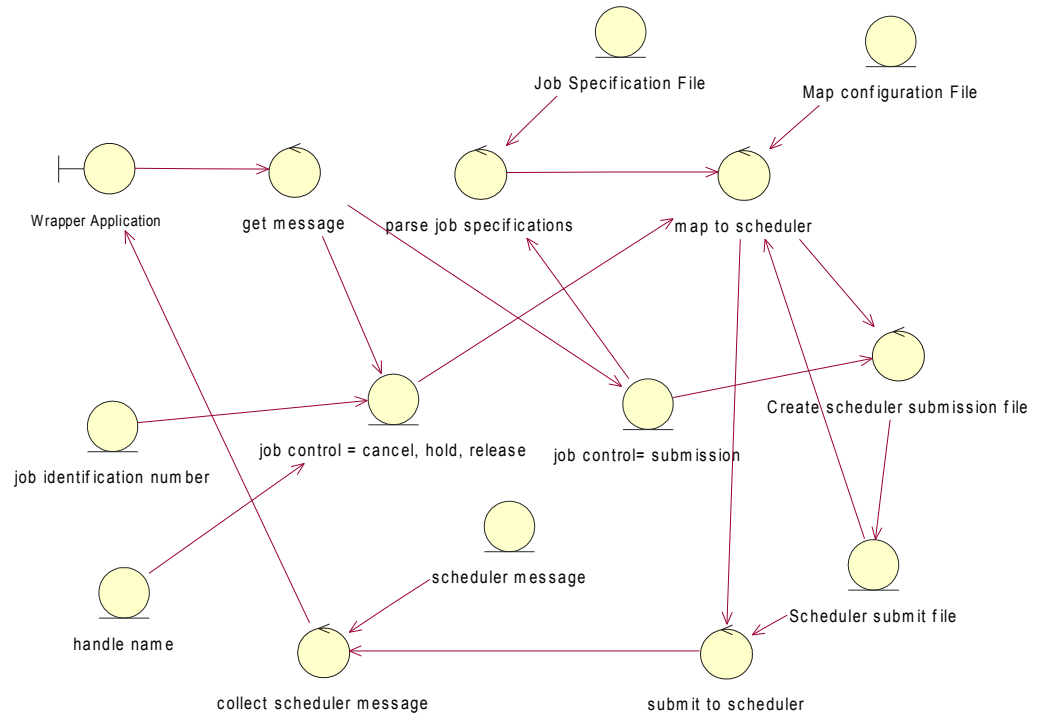
9-13: Represents a job status data / message flow.



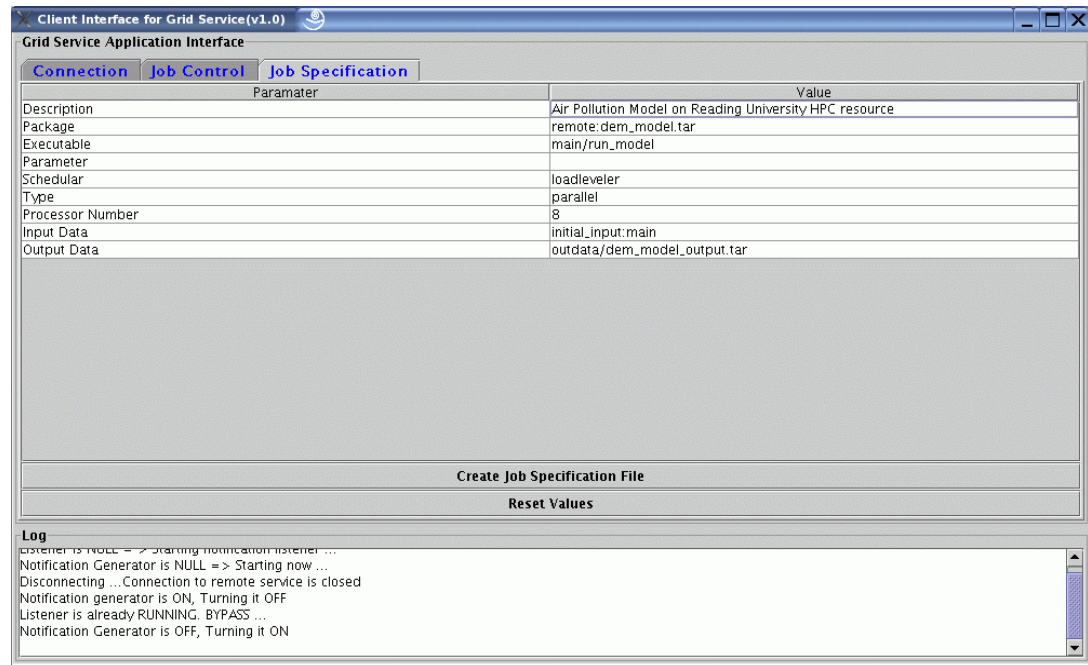
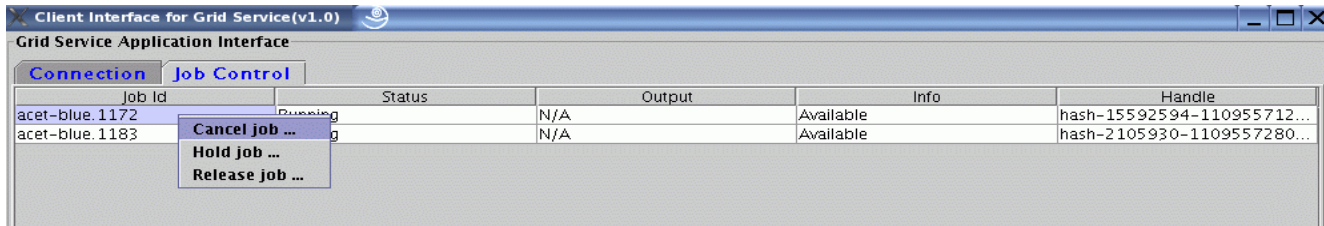
- A, B* : Forwards job control requirements and local scheduler messages.
- C, D* : Handles job status retrieval

# Interface to LoadLeveler

- A Perl script wrapped in *B* and *C* of the resource management programs.
- Parses the job specification file and prepares LoadLeveler job submission file
- Uses a mapping file to map specifications to LoadLeveler commands.
- Submits to LoadLeveler and returns job identification number
- Or LoadLeveler message in case cancel, hold or release are used.



# Client Application



# Conclusions & Future Work

- Development of a resource management system, can easily be extended for other architectures and schedulers.
- Proposed design is generic enough to deploy other applications as well.
- Client application can be extended for generic use or replaced by a portal interface.
  
- **OGSI is dead, Long Live WSRF!**  
OGSI is dead which means WSRF replaces the grid middleware with Web Services.

**BUT**

The functionality of all high level components stays the same!

# Thank you!

---

## Any Questions?

[c.sahin,a.thandavan,v.n.alexandrov]@rdg.ac.uk