Interactivity on the Grid

Marcus Hardt
SCC
(The insitute formerly known as **IWR**)
@FZK

Grid Computing – the dream

Ideal: **Computer power <=> Electrical power**

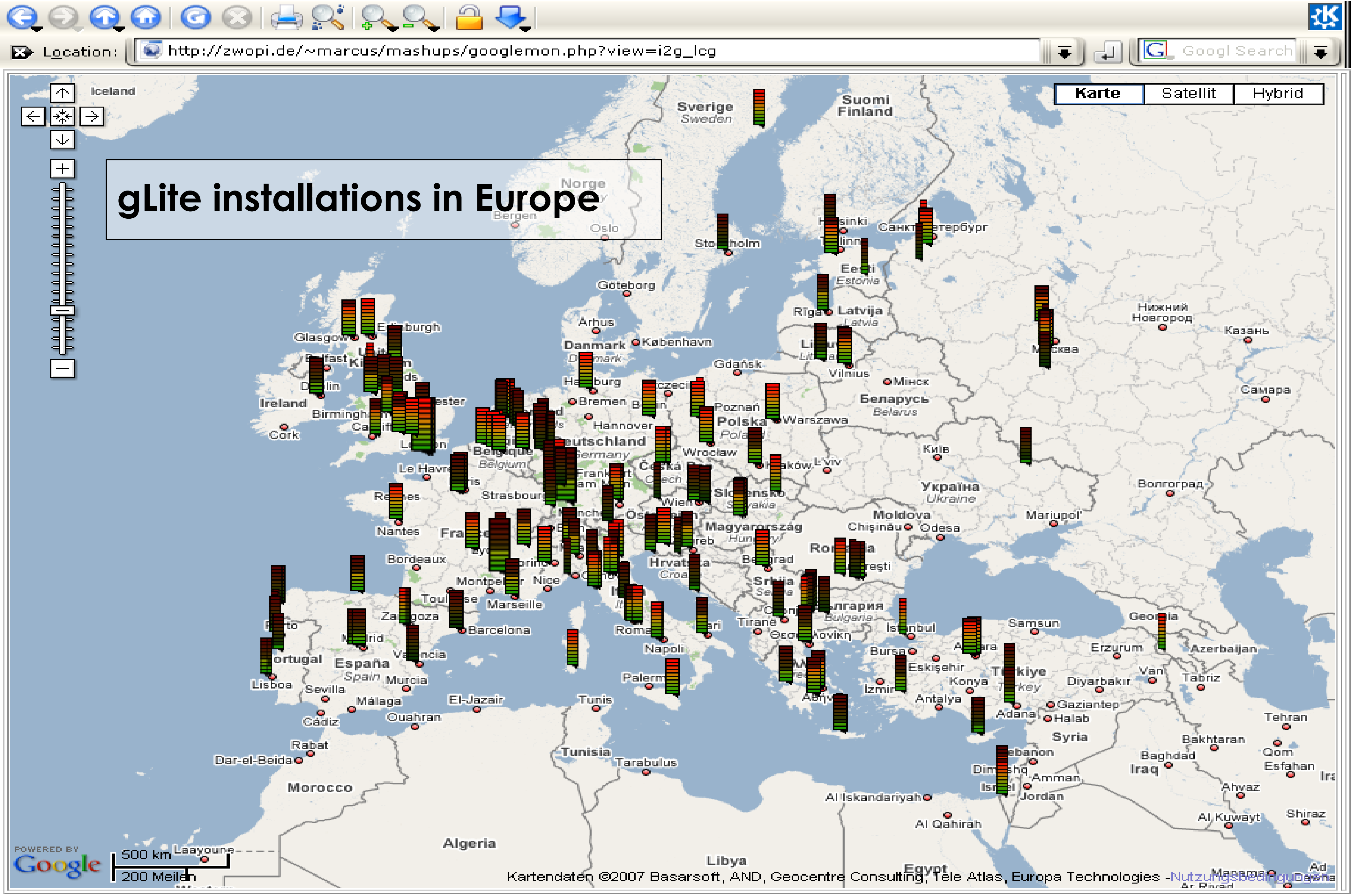
From Electrical power grid => computational grid

- Across organisationsal domains / countries
- Transparent access to
  - Computing
  - Data
  - Network
- Large scale installations

Before int.eu.grid started

- Middleware
  := Layer between application and operating system
- **gLite:** one grid middleware
  - Development driven by CERN
  - Tools for data+computing of new accelerator
  - 10 TB/year * 20 years, random access
- Paradigm: **Send job to where the data is**
- Job: Self contained application

gLite installations in Europe



- Sites: 243 (in 49 countries)
- CPUS: 42798 (176 per site)
- RAM: 19TB
- RAM/CPU: 468MB
- DISK [Tot / Avail]: [8042TB / 5408TB] ([33892GB / 22792GB] per site)

0-9 CPUs --- 10-99 CPUs --- 100-999 CPUs --- > 1000 CPUs

Using a lightbulb in the glite world

- Describe the lightbulb
  Voltage, Watts, Amount
  Lighting_time, ...
- Submit request for electricity to broker

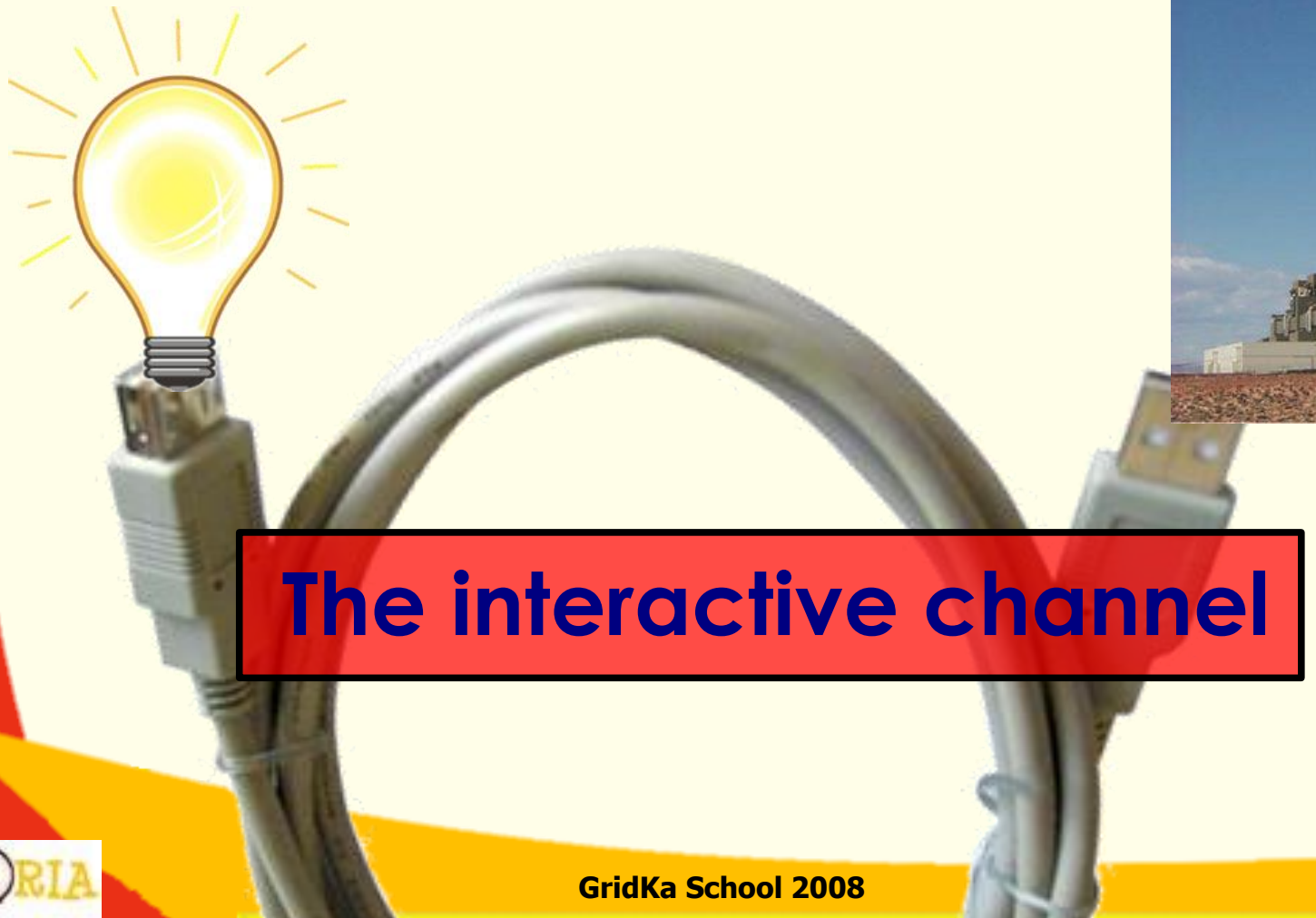
  => Powerplant automatically chosen for you

  => Send lightbulb to powerplant

  => Wait for electricity

  => Lightbulb glows
- Results come back
  - About 20% of the bulbs broken

Our idea for a solution



The interactive channel

The team

Interactive European Grid Project

- ***24 Months***
- ***2,5 ME***
- ***35 People***

http://www.interactive-grid.eu

PARTNERS:

CSIC-IFCA Coord - Spain	LIP Portugal	PSNC Poland	FHK Germany	UAB Spain	CYFRONET Poland	GUP Austria	TCD Ireland	CESGA Spain	II SAS Slovakia	ICM Poland	BIFI Spain	HLRS Germany

Key achievements

- Established interest from research communities
  - Fusion, medicine, environment, HEP, astrophysics
- MPI
  - Open MPI (incl. Infiniband support) & PACX-MPI
    - Collaboration with EGEE
- Interactivity:
  - GVid (& steering through Glogin)
  - CrossBroker
  - Integration in Migrating Desktop (user and developer friendly!!!)

The challenges of int.eu.grid

- From the **middleware point of view**
  - **Parallel Computing** (MPI)
    - Support intracluster Jobs with OpenMPI
    - Support intercluster Jobs with PACX-MPI
  - **Advanced visualization tools** allowing simulation steering
    - GVid, glogin
  - A **Job scheduler** that supports it all
  - **User friendly interface** to the grid supporting all this features
    - Integrating in the Migrating Desktop all the features

- From the **Infrastructure point of view**
  - **Operate** a production level infrastructure 24/7
  - **Support Virtual Organizations** at all levels
    - Running the VO (user support)

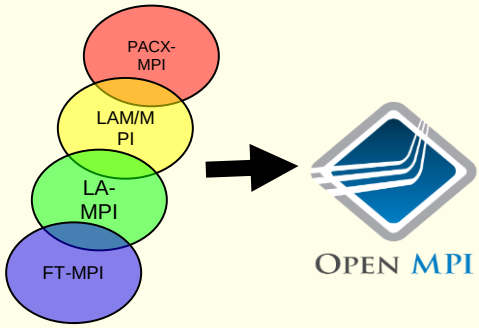
- From the **Applications point of view**
  - Analyze requirements of **reference applications**
    - Ensure that middleware copes the reference applications demands
  - **Application Porting** Support
  - Promote collaborative environments like **AccessGrid**

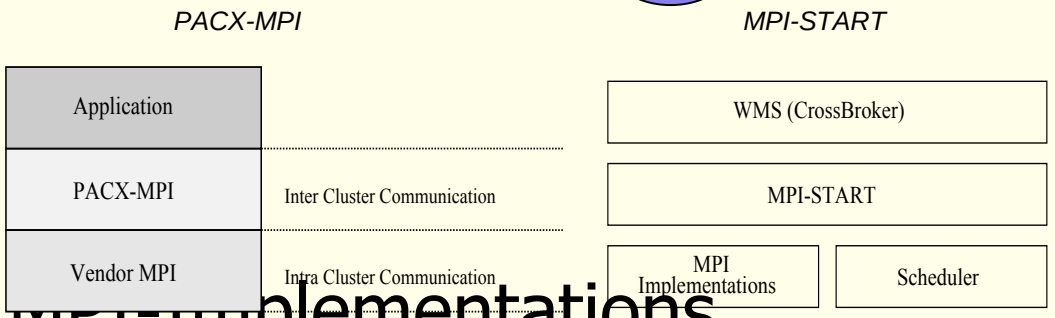
Middleware Requirements

- Provide computing resources
  - **MPI support**
- The job should be started inmediately on the user desktop
  - **Glide-in mechanism**
  - **or… use an Interactive Session**
- The graphical interface should be forwarded to the user desktop
  - **Graphical interface to the grid → Migrating Desktop**
  - **Supporting Visualization → GVid**
- The user should be able to steer the simulation
  - **Real Time steering → glogin**

MPI types supported

- OpenMPI
  - Joint Effort / OSS
  - Best of FT-MPI, LA-MPI, LAM-MPI, PACX-MPI
- PACX-MPI
  - Inter-Cluster / Site
  - Supports Vendor-MPI
- MPI-Start
  - Shell Scripts
  - Abstraction Layer WMS / Schedulers and MPI-Implementations
  - Flexibility (relocateable Shellscripts)

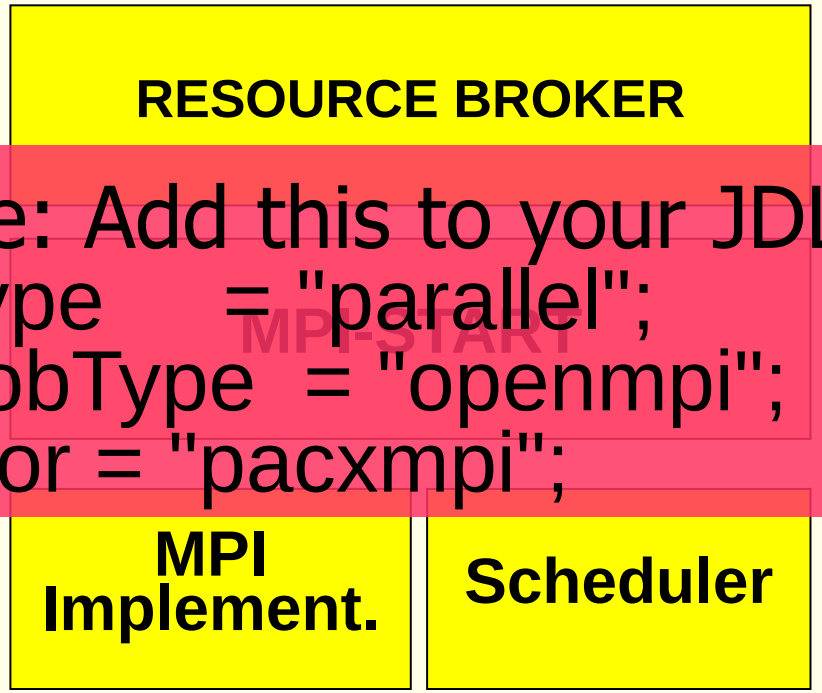




MPI Support on the Grid

Our solution, an intermediate layer:

mpi-start



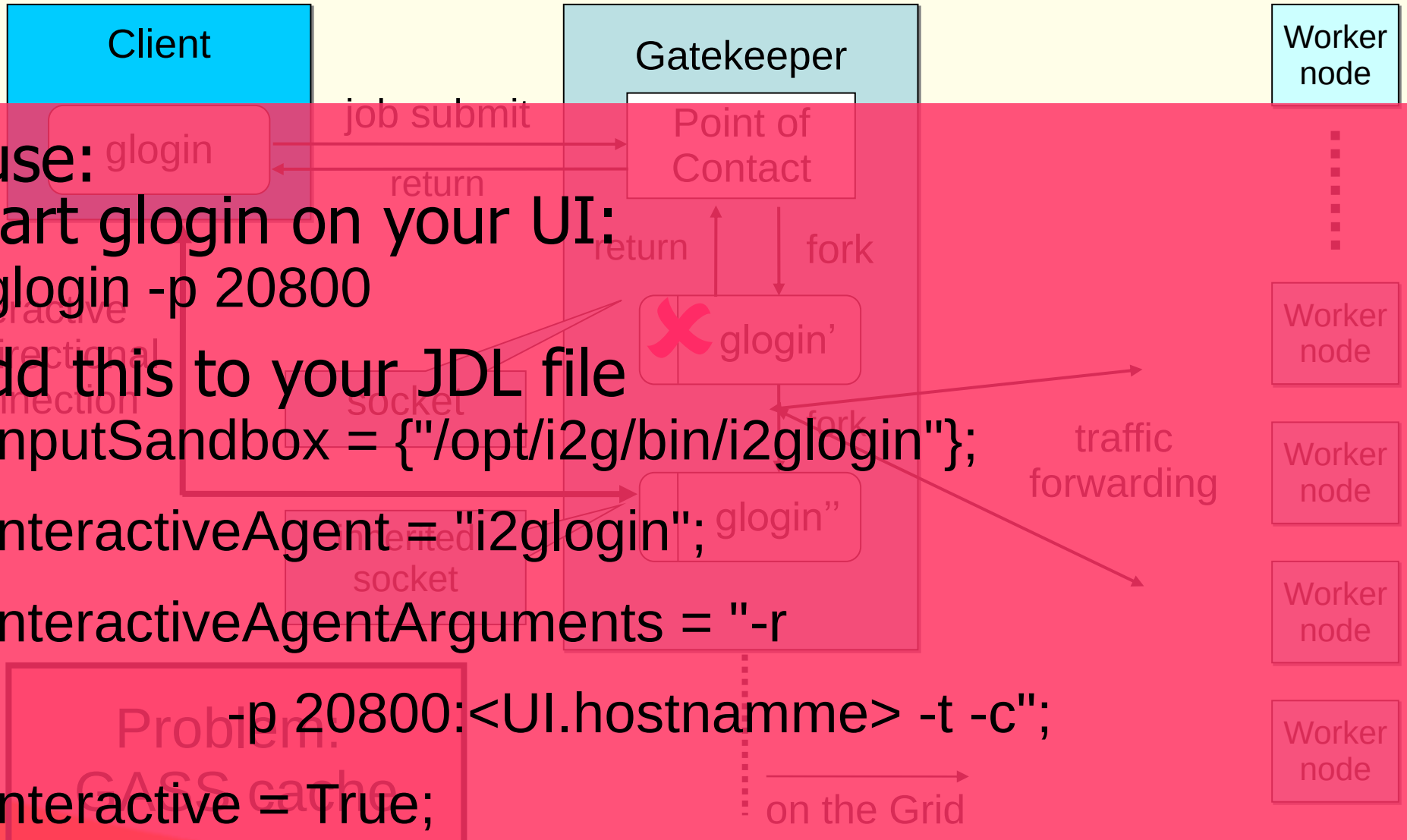
To use: Add this to your JDL file
JobType = "parallel";
SubJobType = "openmpi";
or = "pacxmpi";

Middleware for Visualization & Steering

- **Glogin**
  - **Lightweight** tool for support of **interactivity** on the grid
  - Grid authenticated shell access "glogin host"
  - **No dedicated daemon** needed such as *sshd*
  - **TCP Port Forwarding** enables access to grid worker nodes with **private IPs**.
  - **X11** Forwarding

- **GVid**
  - **Grid Video Service**
  - Visualization can be executed remotely on a grid resource
  - **Transmits the visualization output to the user desktop**
  - Communication of the interaction events back to the remote rendering machine
  - Uses **Glogin** as bi-directional communication channel

(i2)glogin



To use:

- Start glogin on your UI:

  glogin -p 20800

- Add this to your JDL file

  InputSandbox = {"/opt/i2g/bin/i2glogin"};

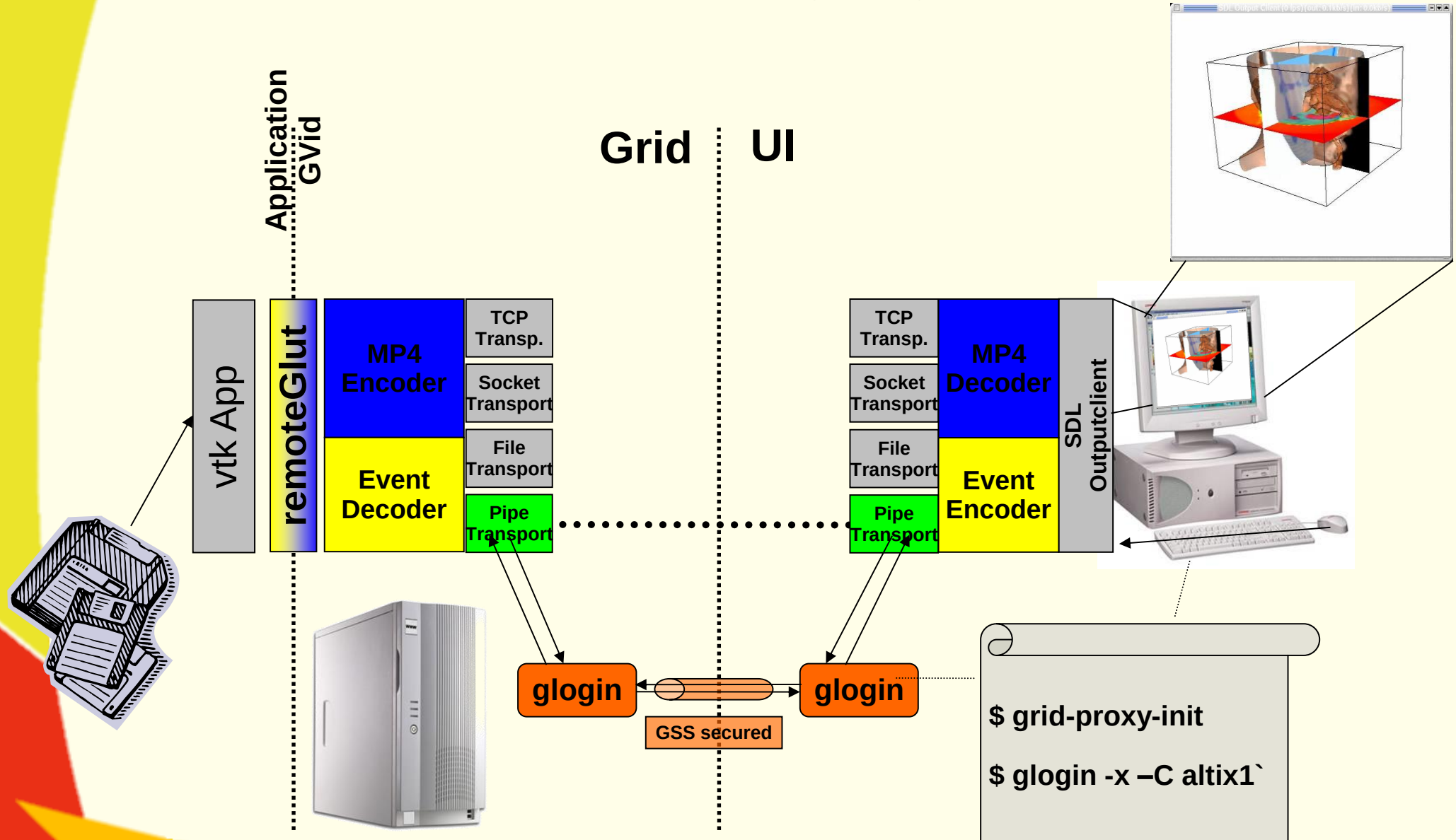
  InteractiveAgent = "i2glogin";

  InteractiveAgentArguments = "-r

  -p 20800:<UI.hostnamme> -t -c";

  Interactive = True;

GVid

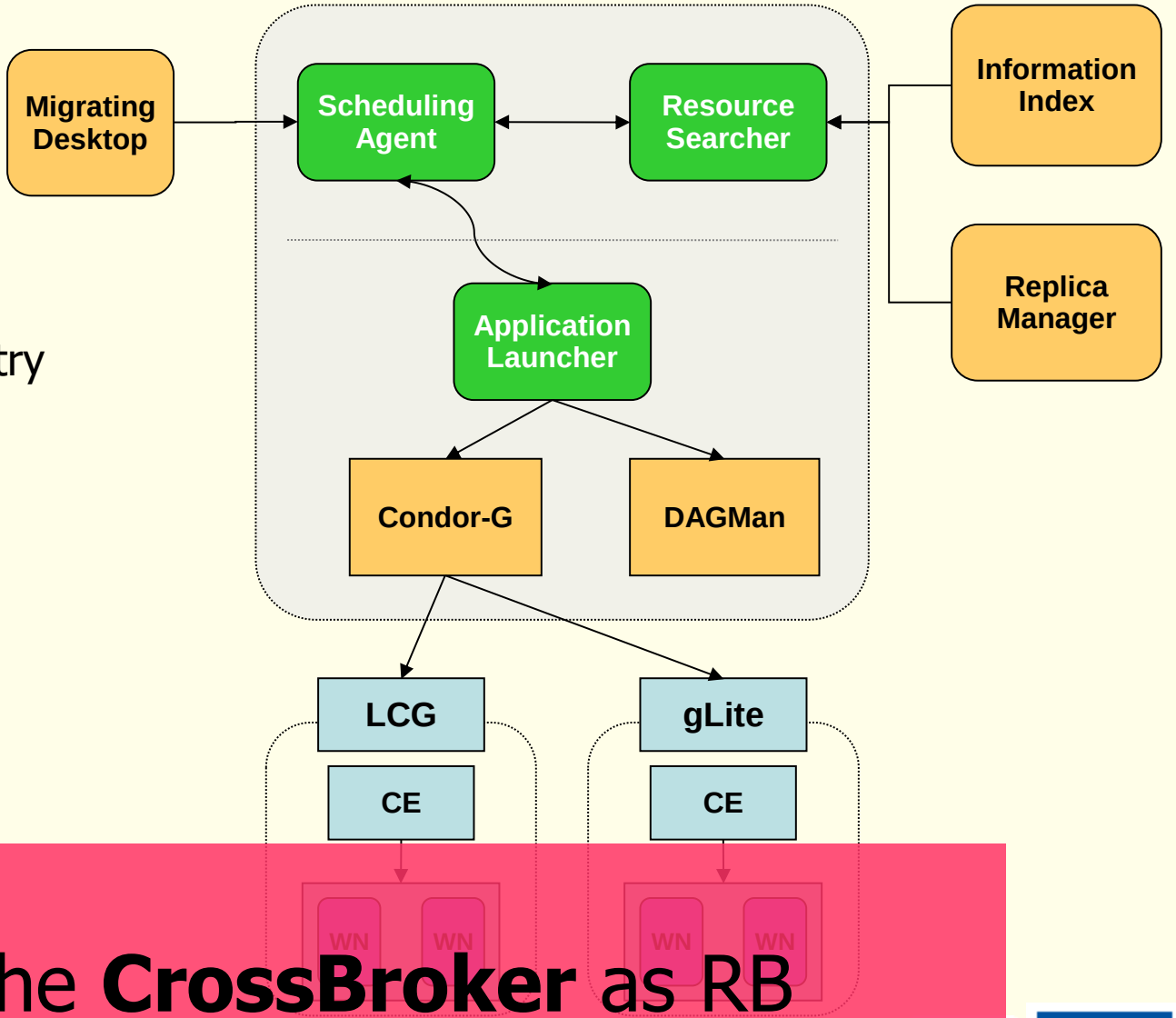

Application
GVid
Grid
UI
vtk App
remoteGlut
MP4
Encoder
Event
Decoder
TCP
Transp.
Socket
Transport
File
Transport
Pipe
Transport
TCP
Transp.
Socket
Transport
File
Transport
Pipe
Transport
MP4
Decoder
Event
Encoder
SDL
Outputclient
glogin
glogin
GSS secured
$ grid-proxy-init
$ glogin -x –C altix1`

Concurrency: glide-in

- Glide-in is a different way to submit job
  - Uses condor glide-in
- Resource Broker keeps a handle to job
  - Change priority of jobs
  - Submit ”high-priority-jobs”
  - VO policies respected

CrossBroker

- Interactive Job Broker
  - Automatic Job Management for Parallel Applications
    - Resource Searching
    - Job Conditioning
    - Launching, Monitoring, Retry
    - Result Retrieval
  - Workflows, Interactive & Batch Jobs, MPI Support
    - JDL Extensions
    - Compatible
  - Best Effort Approach for Failures / Problems
  - Improved Job Startup Time

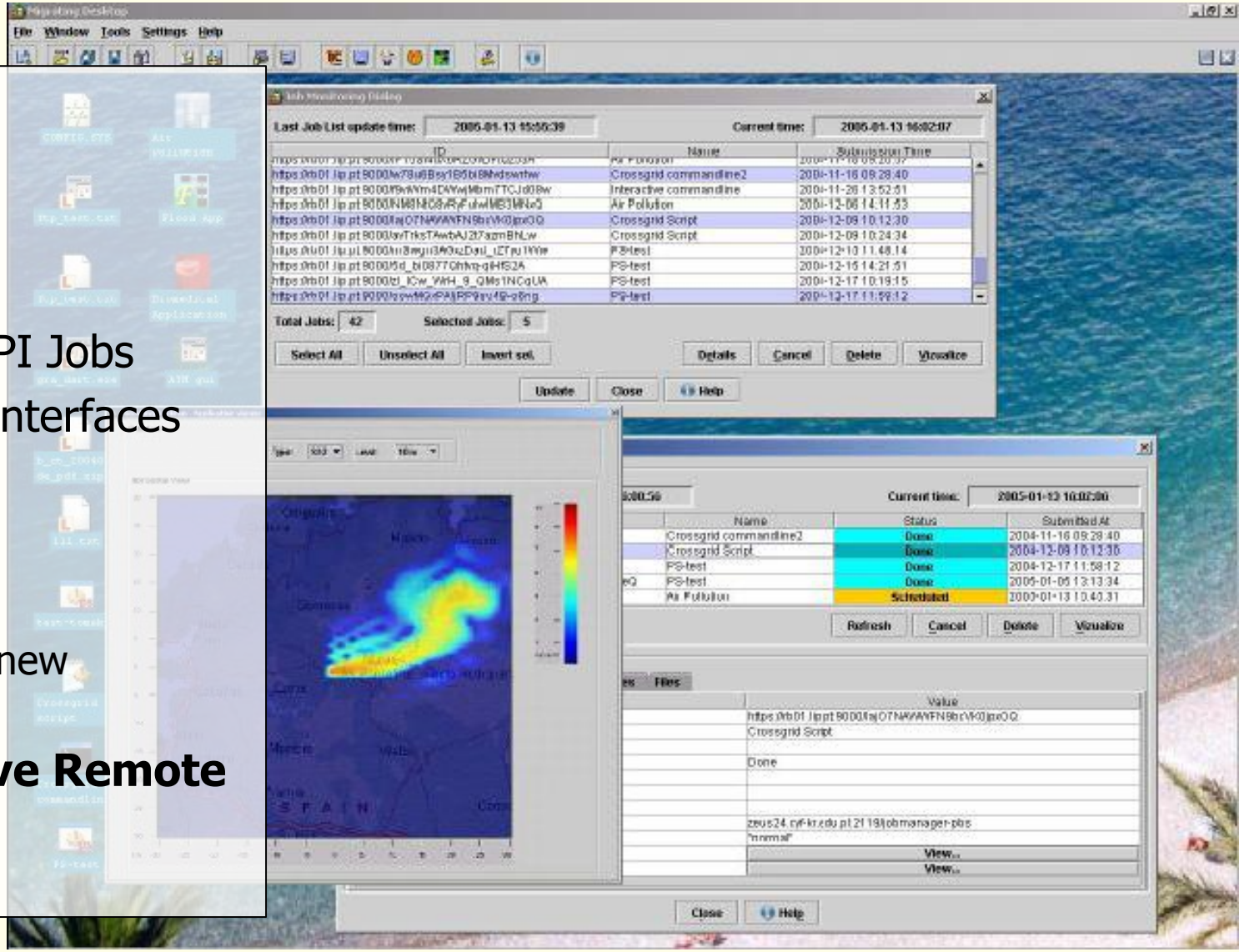


To use:

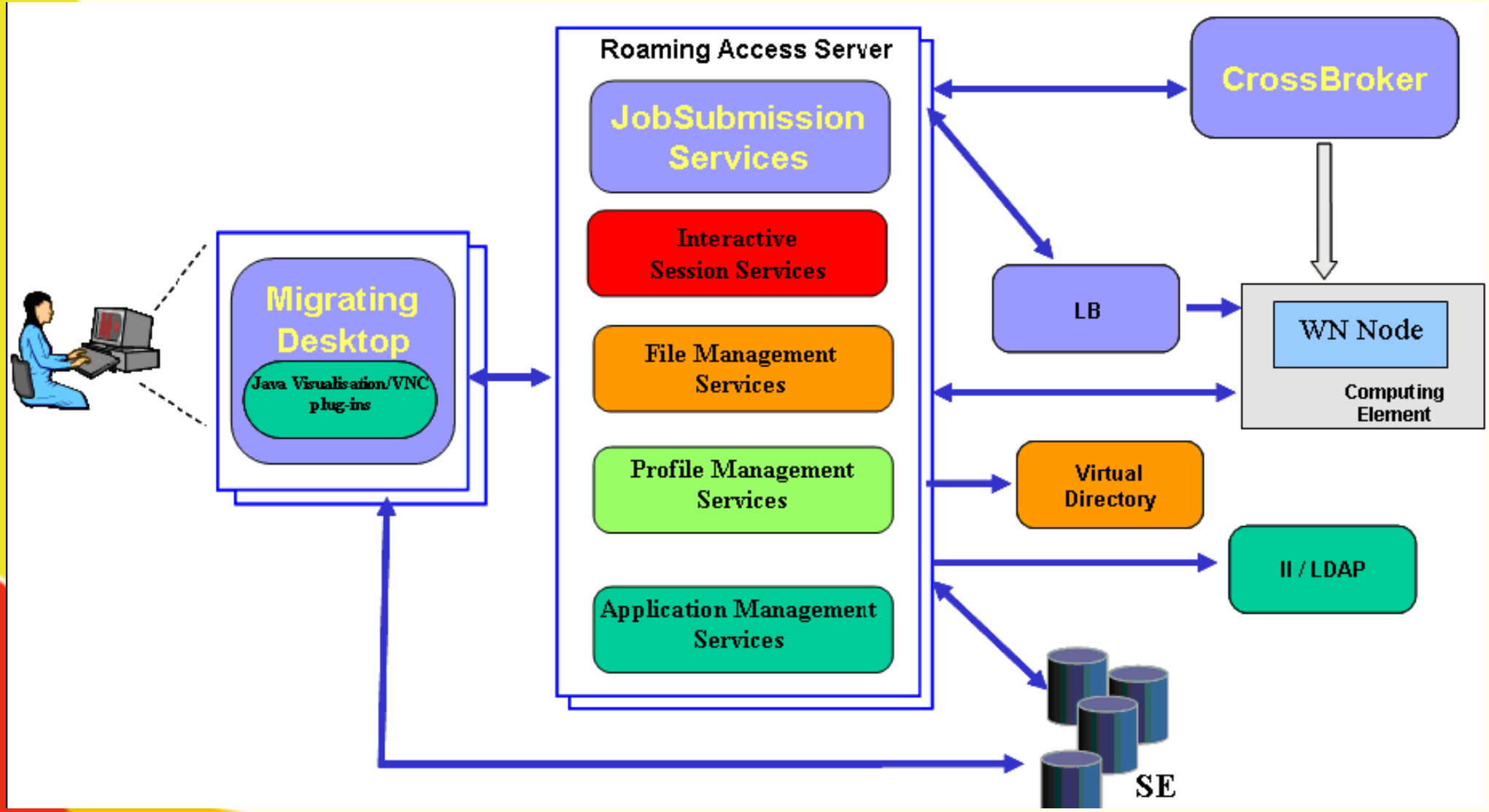
- Use the **CrossBroker** as RB
- Use an **interactive CE**

GUI – Migrating Desktop + Backend

- User-Friendly
- Platform Independent
  - **Java**
- Running Sequential & MPI Jobs
- Roaming Access Server interfaces with the UI
- Data Management
- Easily Extendable
  - **Plugin Mechnism** for new Applications
- Extension for **Interactive Remote Visualization**



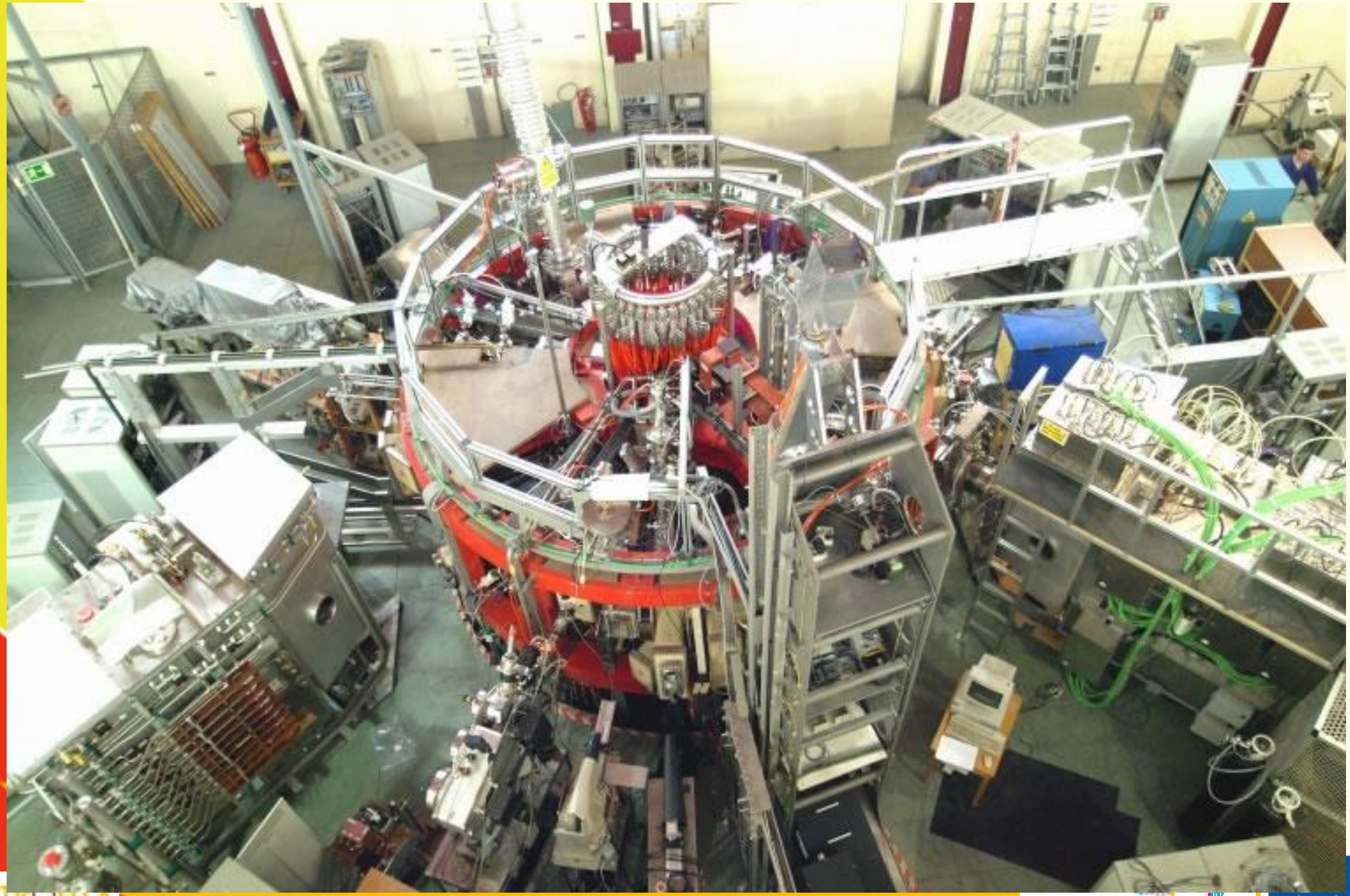
GUI – Migrating Desktop + Backend



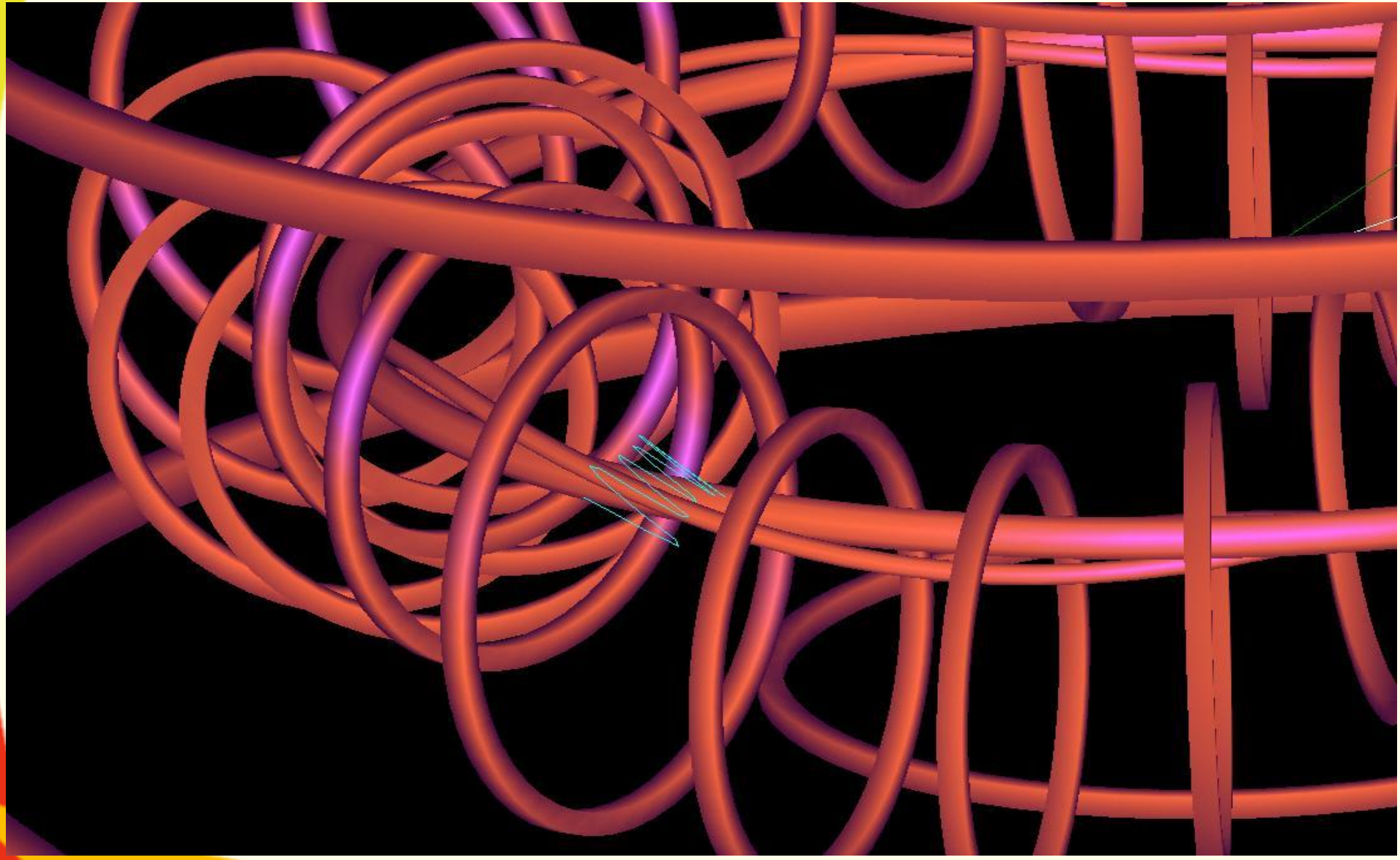
Example: Fully integrated Application

- **Fusion Application** – It integrates:
  - MPI Support
  - Crossbroker
  - Visualization
  - Videostreaming
  - GUI Support
  - Interactivity

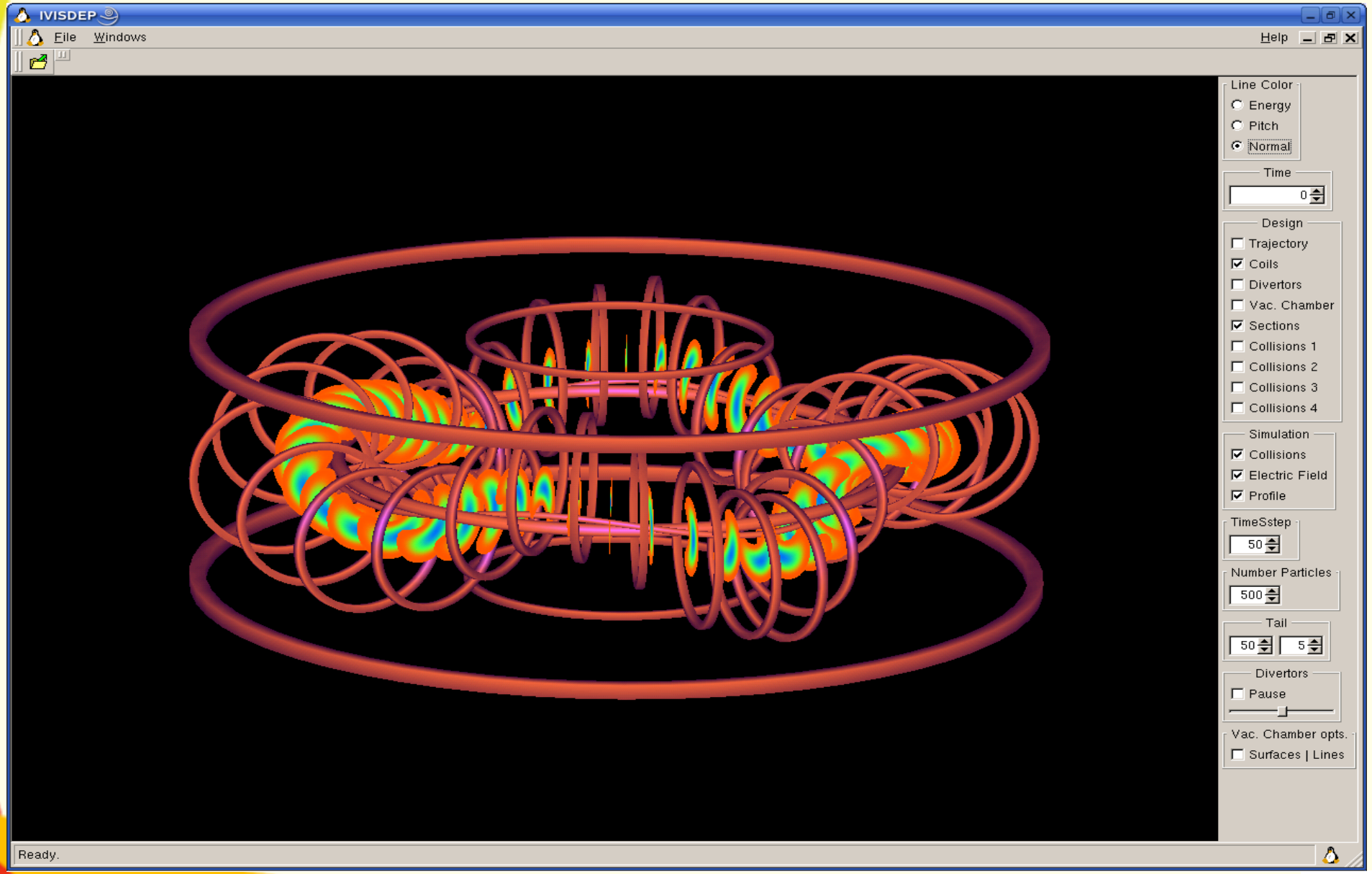
Fusion Reactor "TJ-II" in Zaragoza



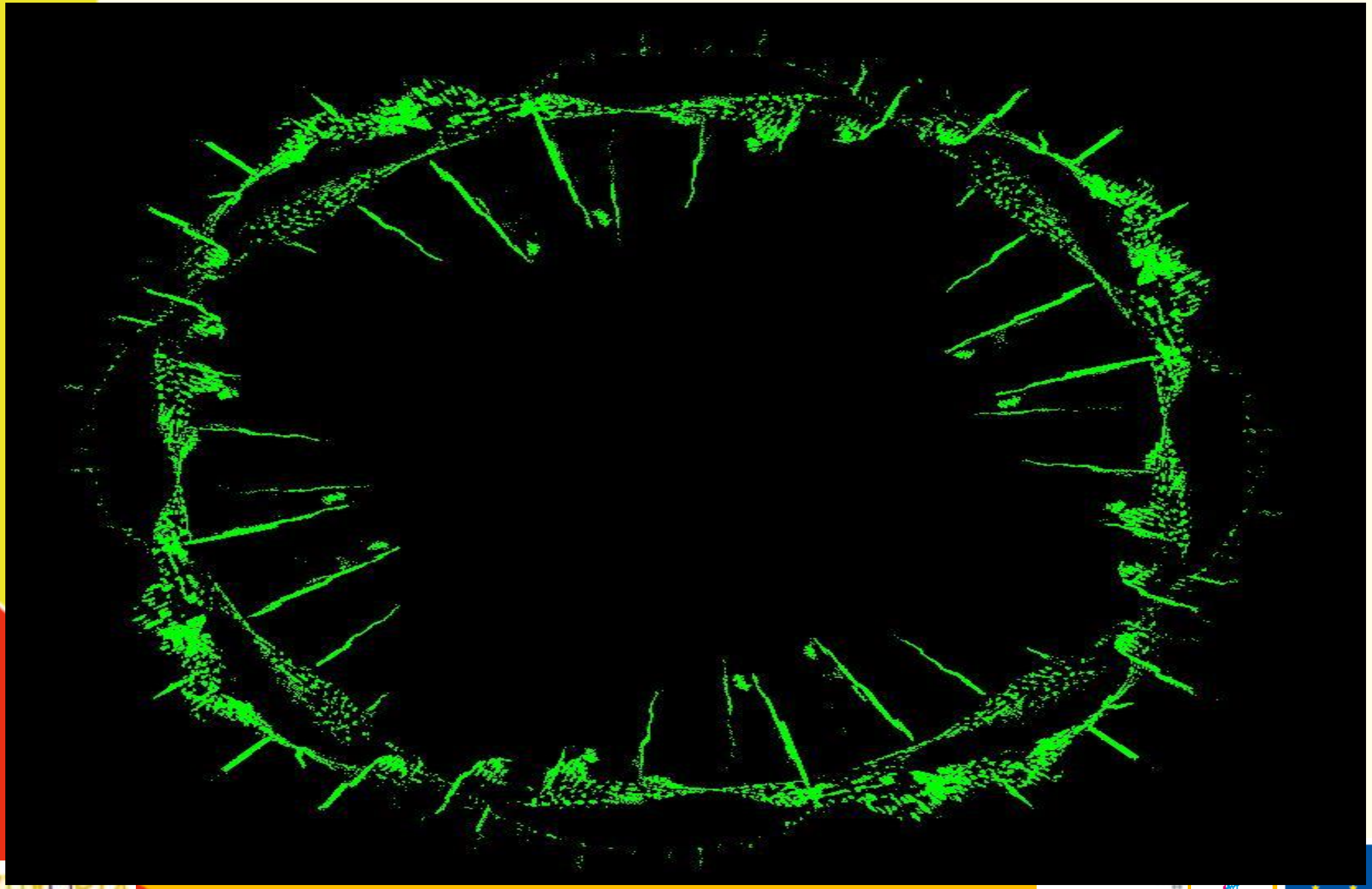
Individual trajectories computed on the grid



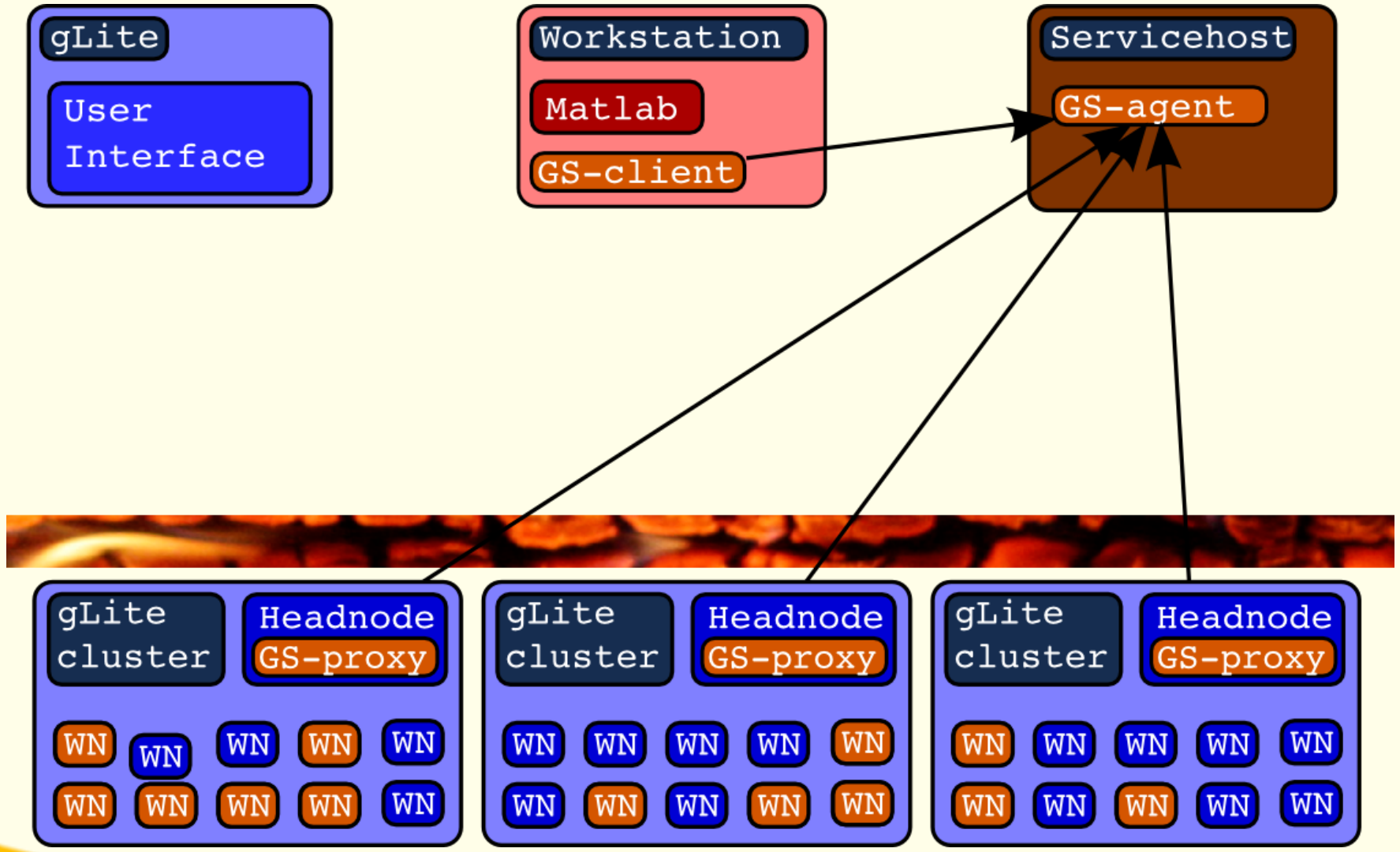
Interactive Visualizator for ISDEP



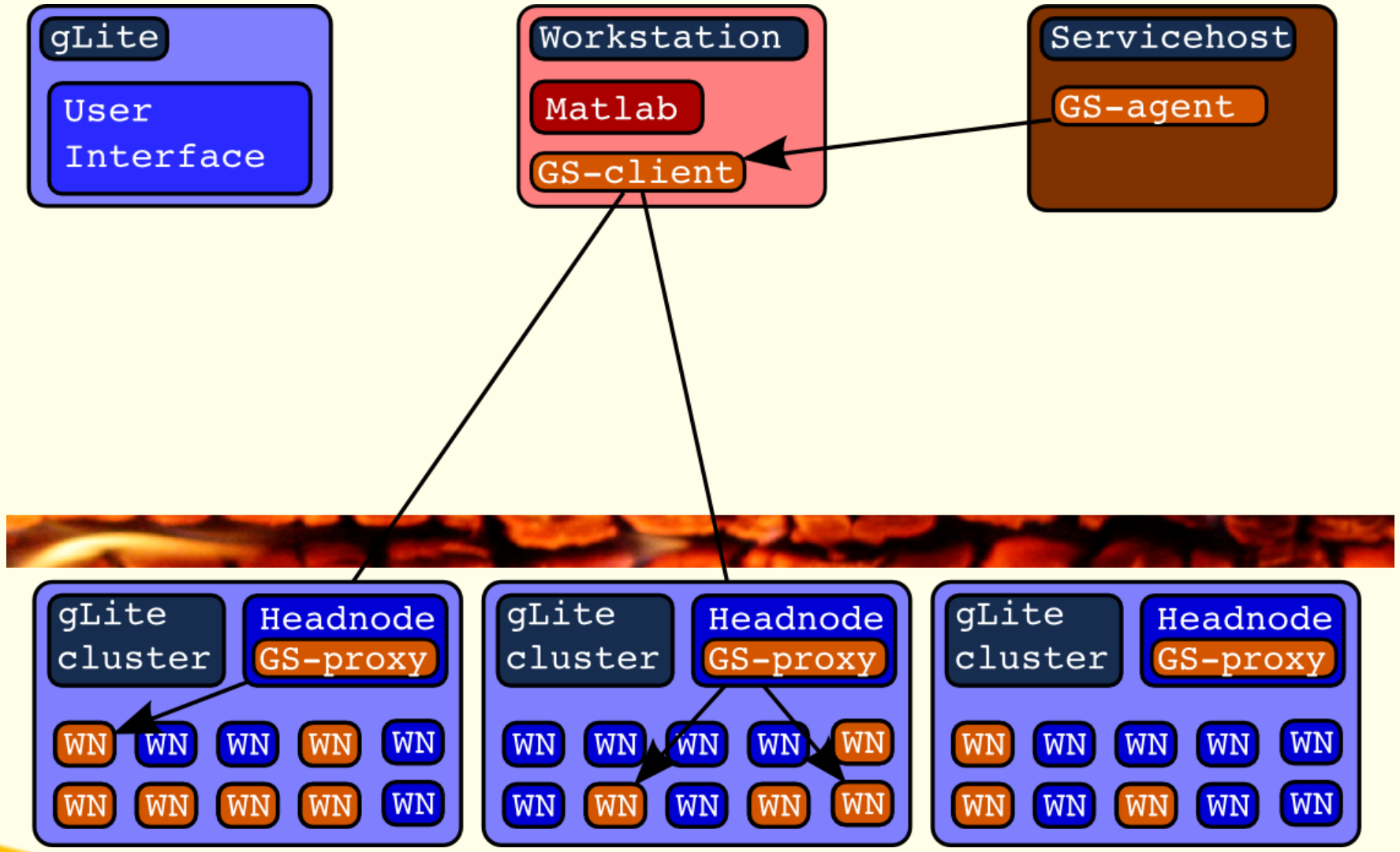
Particles hit the vacuume vessel



Another interactive cable: GridSolve



Another interactive cable: GridSolve



GridSolve interface

- **Interface**
  - API for Java, C, Fortran, **Matlab**, Octave, ...
- Easy to use:

  y=problem(x) <=> y=gs_call('problem', x)
  - Transport input parameters to remote side
  - Execute "problem"
  - Transport result back

=> Reduce complexity of the grid to one function call

Life-Demo

- Movie of the life demonstration:
  - **http://marcus.hardt-it.de/grid4matlab**
- **Life demo** on int.eu.grid

Source code

```
function f=broetchenverteiler_p (N, RESO, MAX_ITERATIONS)
for i=1:N;
        session_id(i)=gs_call_async('maendele', i-1, N, RESO, MA
end
while (num_finished < N)
        for i=1:N;
                status(i)=gs_probe(session_id(i));
                if (status(i) == 0 )
                        result=gs_wait(session_id(i));
                end
        end
end
```


Google
interactive grid
I'm Feeling Lucky

Interactive Job Support

```
Type = "Job";
VirtualOrganisation = "imain";
JobType = "Parallel";
SubJobType = "openmpi";
NodeNumber =  11;
Interactive = TRUE;
InteractiveAgent = "glogin";
InteractiveAgentArguments = "-r -p 195.168.105.65:23433";
Executable = "test-app";
InputSandbox = {"test-app", "inputfile"};
OutputSanbox = {"std.out", "std.err"};
StdErr = "std.err";
StdOutput = "std.out";
Rank = other.GlueHostBenchmarkSI00 ;
Requirements =
   other.GlueCEStateStatus == "Production";
```