int.eu.grid

http://www.interactive-grid.eu

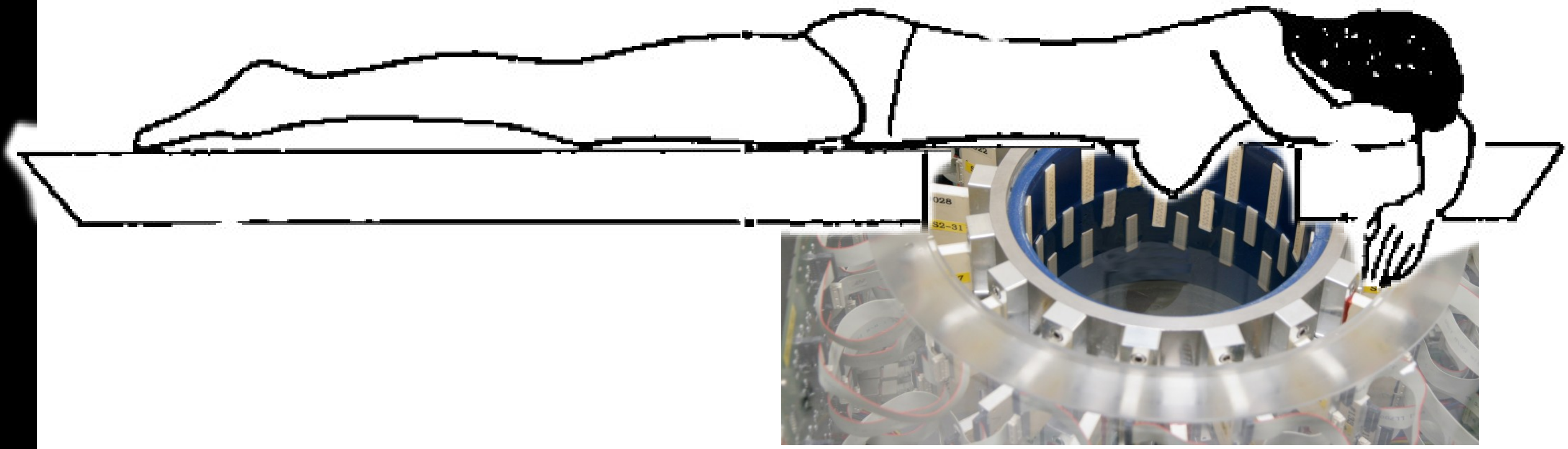
Gridcomputing for Ultrasound CT (USCT)

Marcus Hardt
SCC (Formerly **IWR**) @ FZK

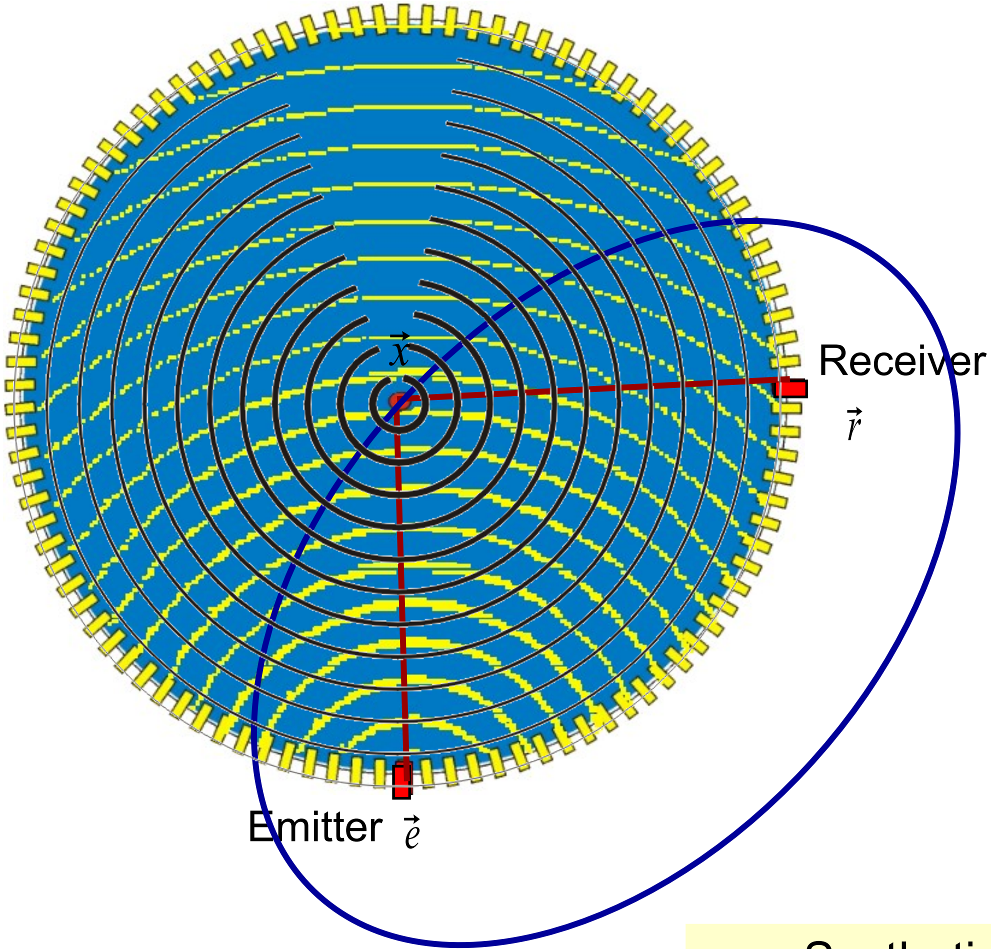
Ultrasound Computer Tomography (USCT)

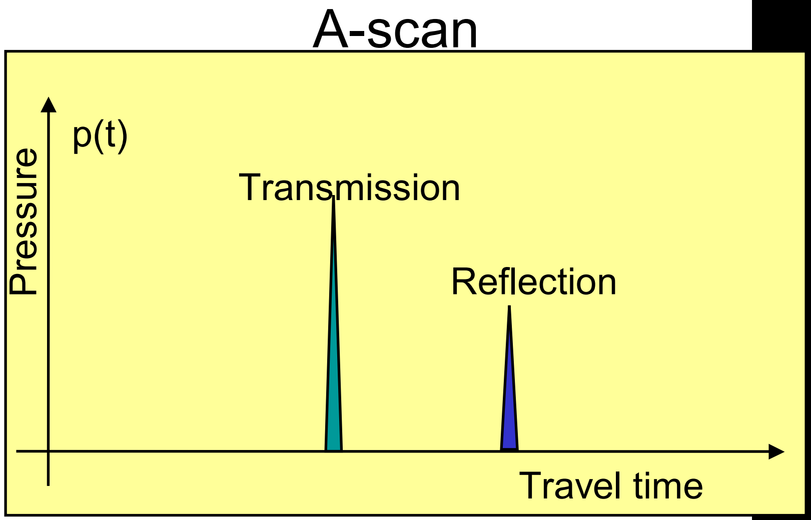
- New method for medical imaging
- Focus: Breast cancer diagnosis

USCT setup



USCT – Method





$$t=\frac{|\vec{e}-\vec{x}|+|\vec{x}-\vec{r}|}{c}$$

c sound speed ≅ constant

$$R(\vec{x})=\sum_{\vec{e},\vec{r}} p\left(\frac{|\vec{e}-\vec{x}|+|\vec{x}-\vec{r}|}{c}\right)$$

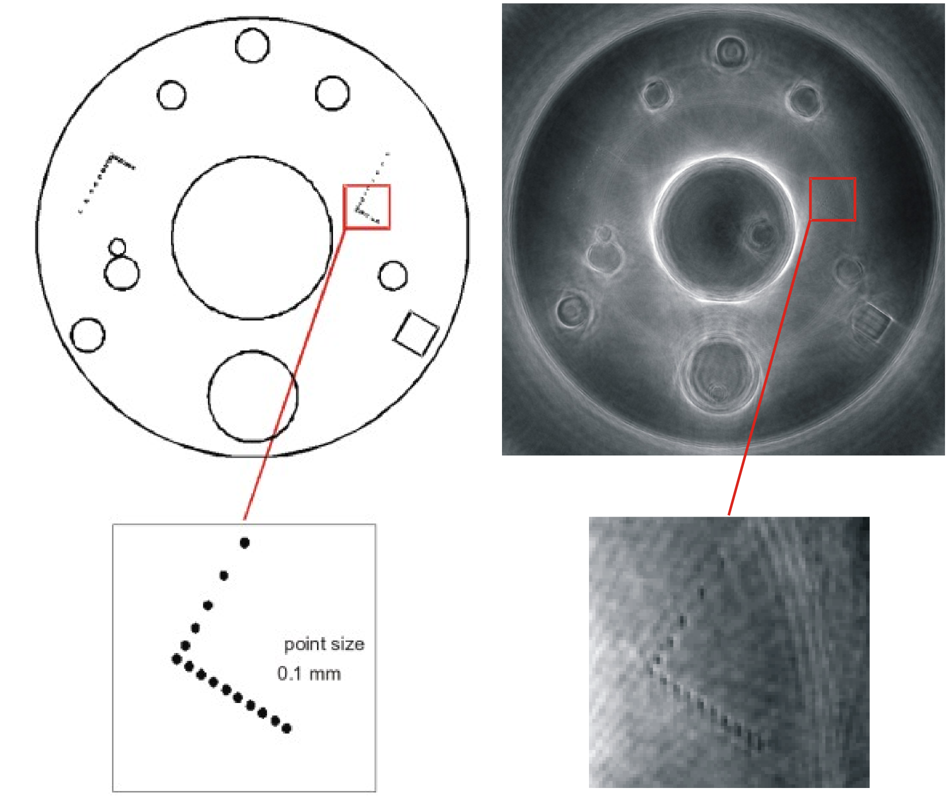
Mean frequency: 3 MHz,

Synthetic Aperture Focussing Technique

USCT Images

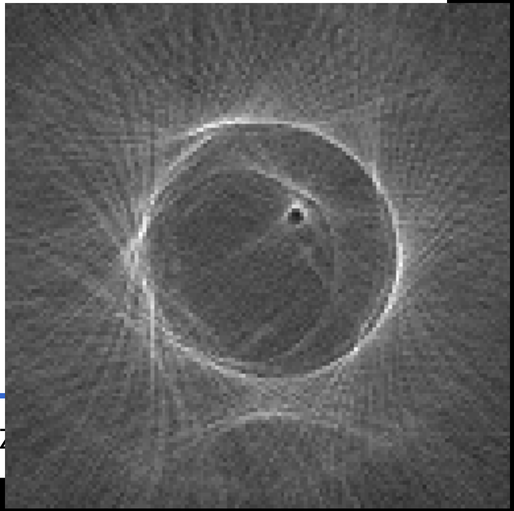
First results with 2D USCT:

- 0.1 mm nylon threads visible



Current results with new hardware:

- Egg and yolk visible
- 3D imaging



USCT Algorithm

- Characteristicts:
  - Input: 20 GB (full set)
  - Computing time depends
    - on output size / resolution
    - amount of input data

٣٥MB	٢٠GB	٢٠GB	Data
4096^2	128^2x100	4096^2x3410	Voxels
١ Hour	١.٥ Months	١٥٠ Years	Time

- Matlab
  - Strategic development platform (95% sourcecode)

USCT Algorithm

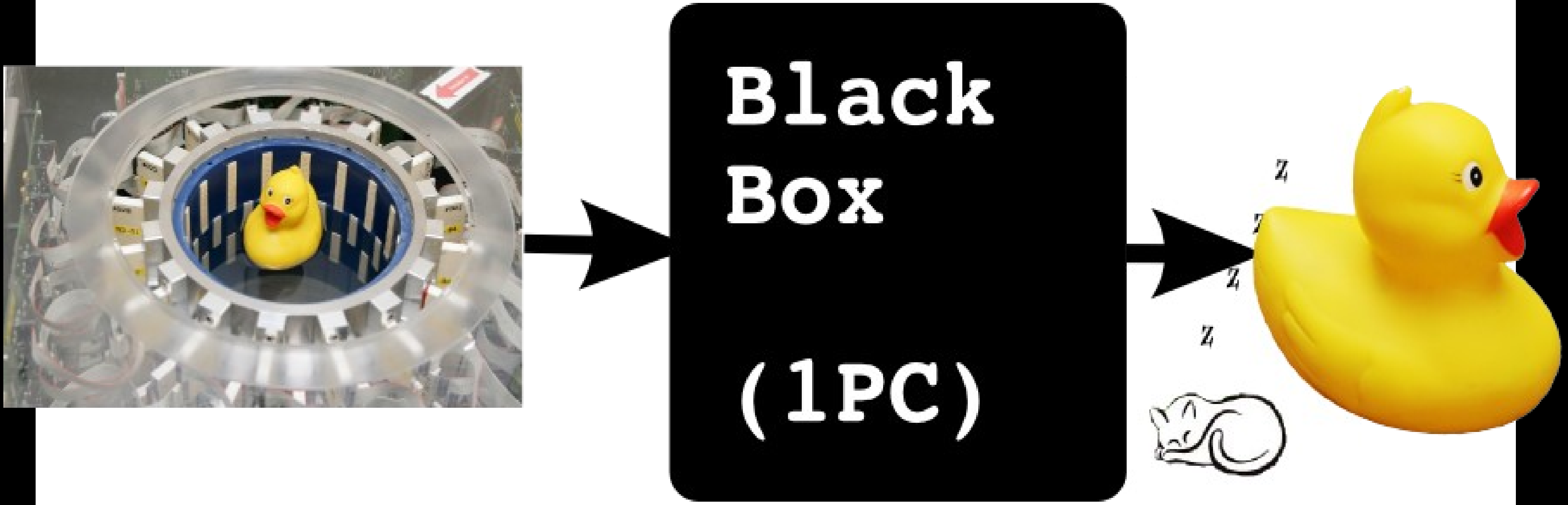
- Characteristicts:
  - Input: 20 GB (full set)
  - Computing time depends
    - on output size / resolution
    - amount of input data

٣٥MB	٢٠GB	٢٠GB	Data
4096²	128²x100	4096²x3410	Voxels
١ Hour	١.٥ Months	١٥٠ Years	Time

- Matlab
  - Strategic development platform (95% sourcecode)

- **Goals for grid access:**
  - **Seamless**
  - **Interactive**
  - **from Matlab**

USCT reconstruction := “Black Box”



Black Box

(1PC)

- Computation takes long (days, weeks, years)
- Grid in order to speed up

Grid Computing

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

From Electrical power grid => computational grid

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

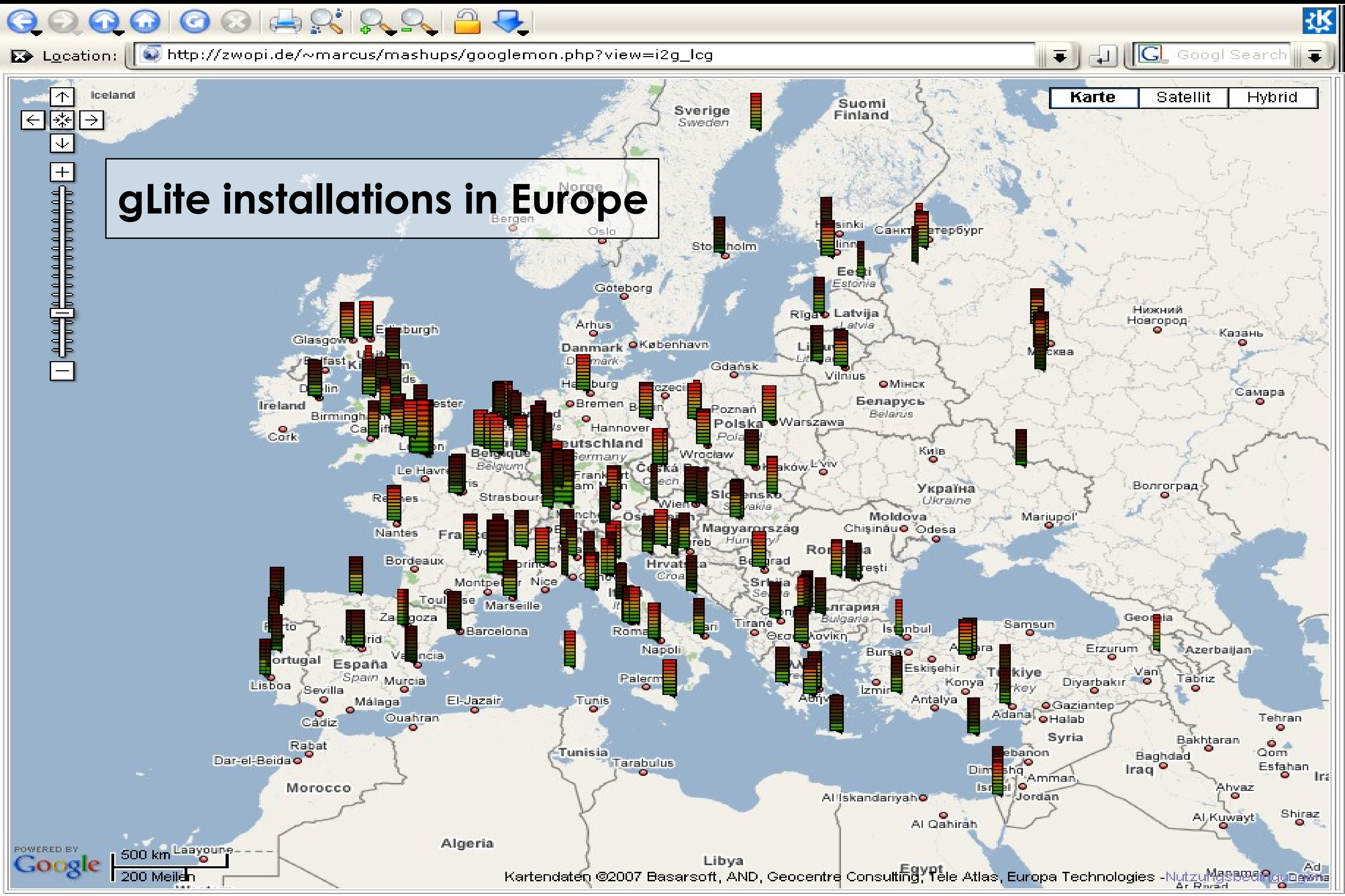
Grid middleware

- 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

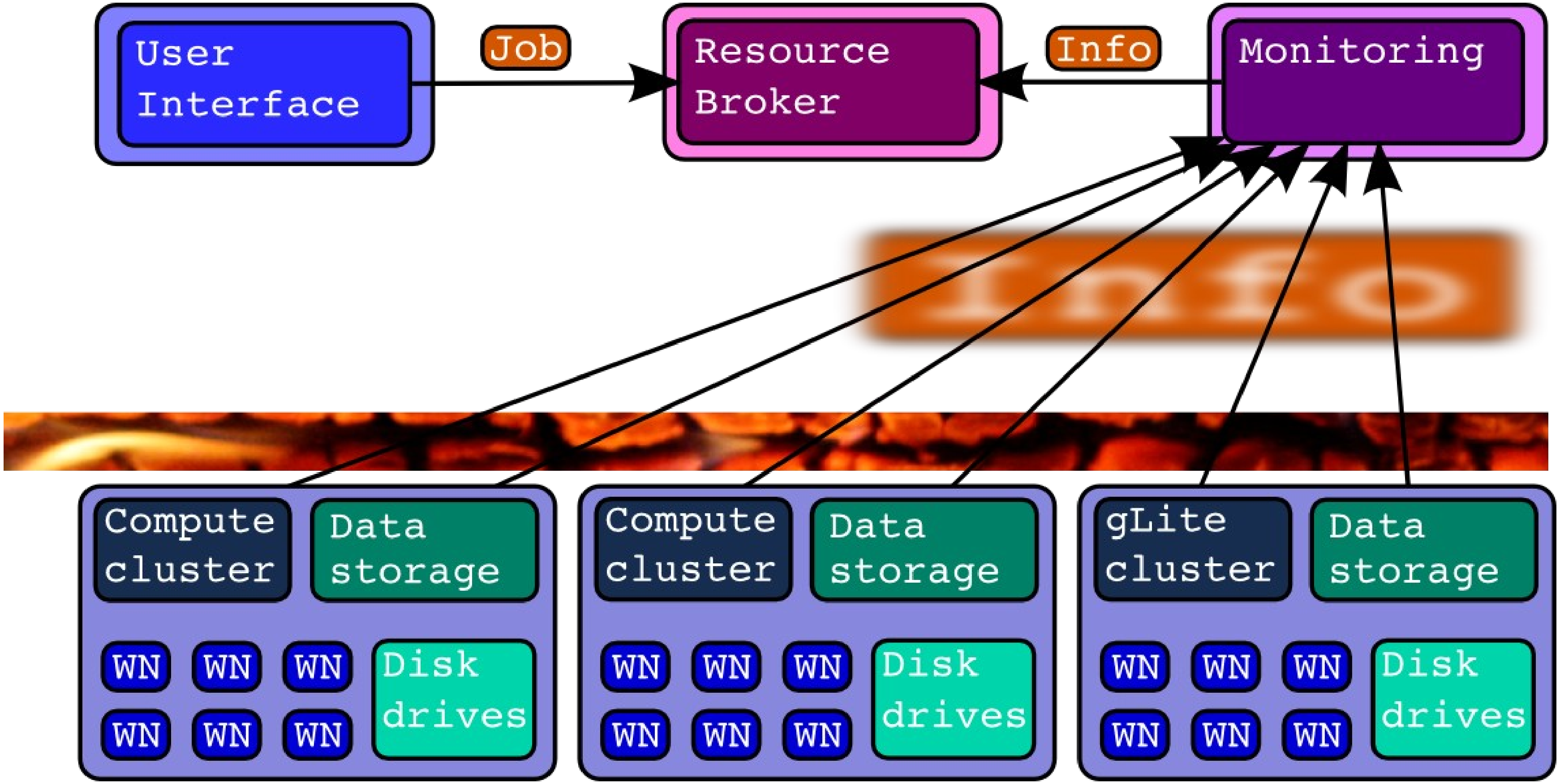


Location: http://zwopi.de/~marcus/mashups/googlemon.php?view=i2g_lcg

- 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

gLite architecture

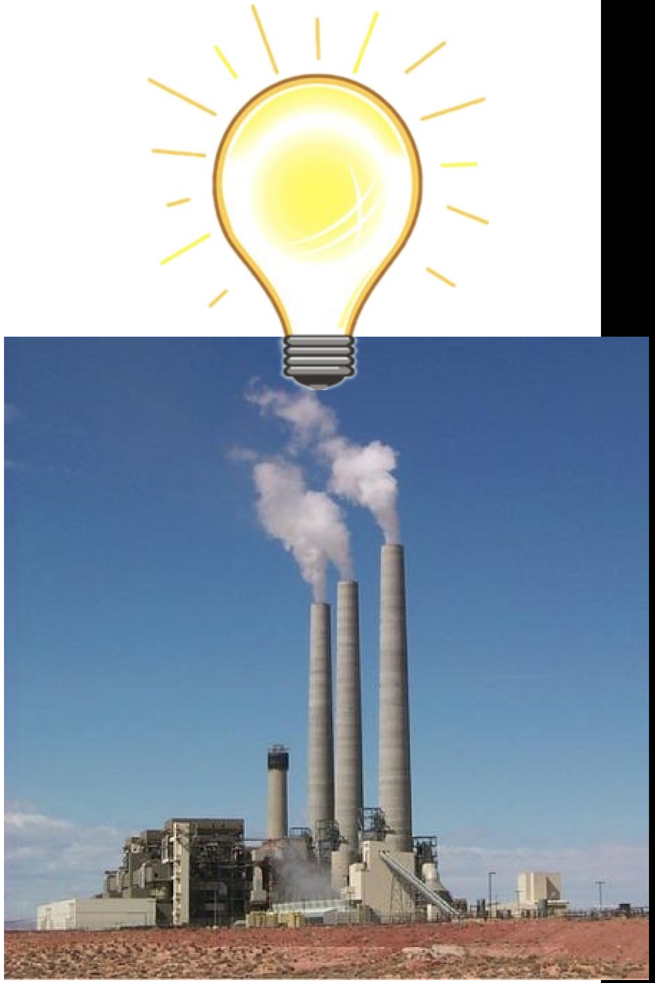


Using a lamp in the grid world

- Describe the lamp

```
Voltage, Watts, Number_Lamps
Hertz, Lighting_time, ...
```

- Submit request for electricity to broker
  - => Powerplant chosen for you
  - => Send lamp to powerplant
  - => Wait for electricity
  - **=> Lamp glows**
- Results come back
  - About 20% of the lamps broken

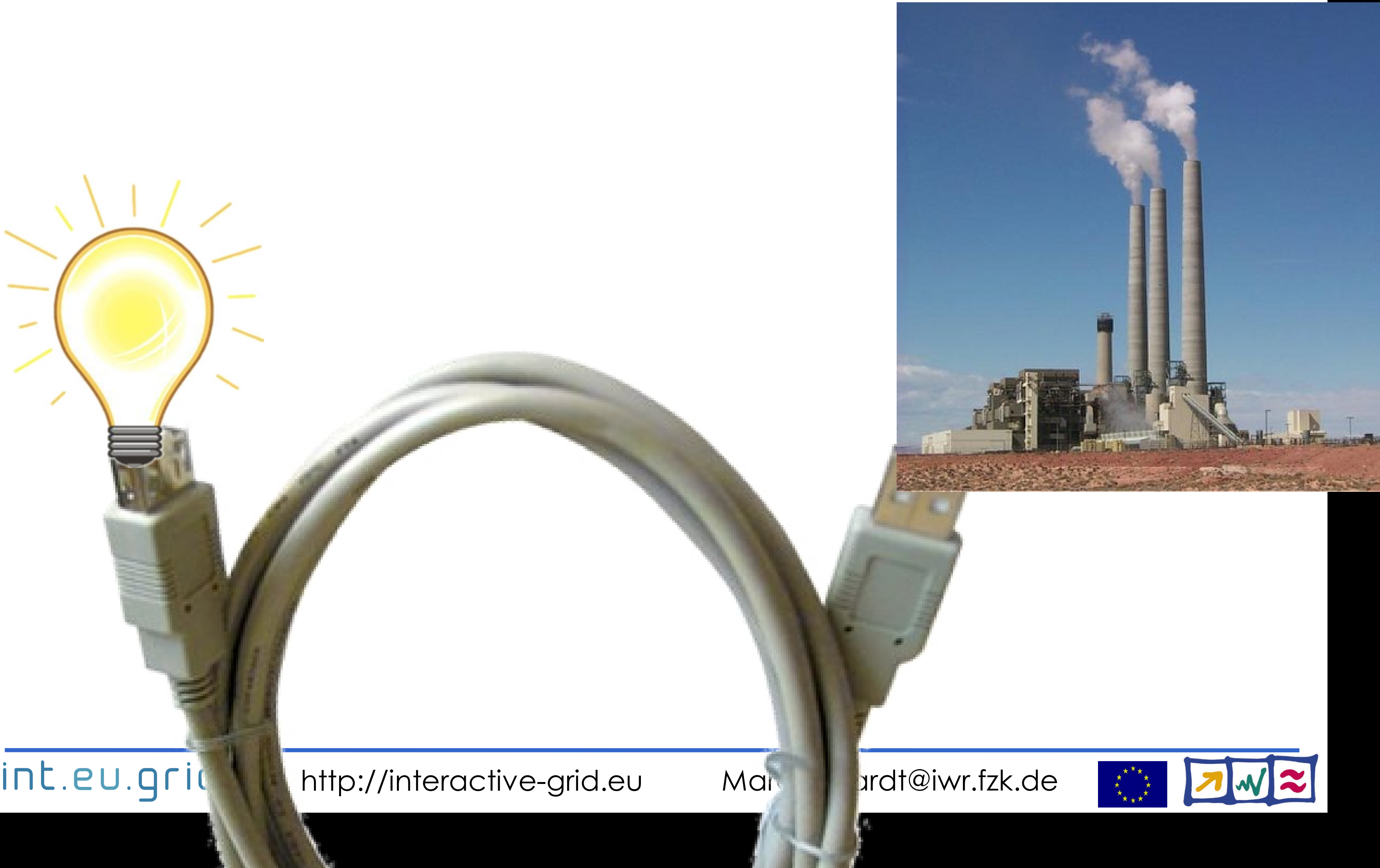


Is interactivity a solution?

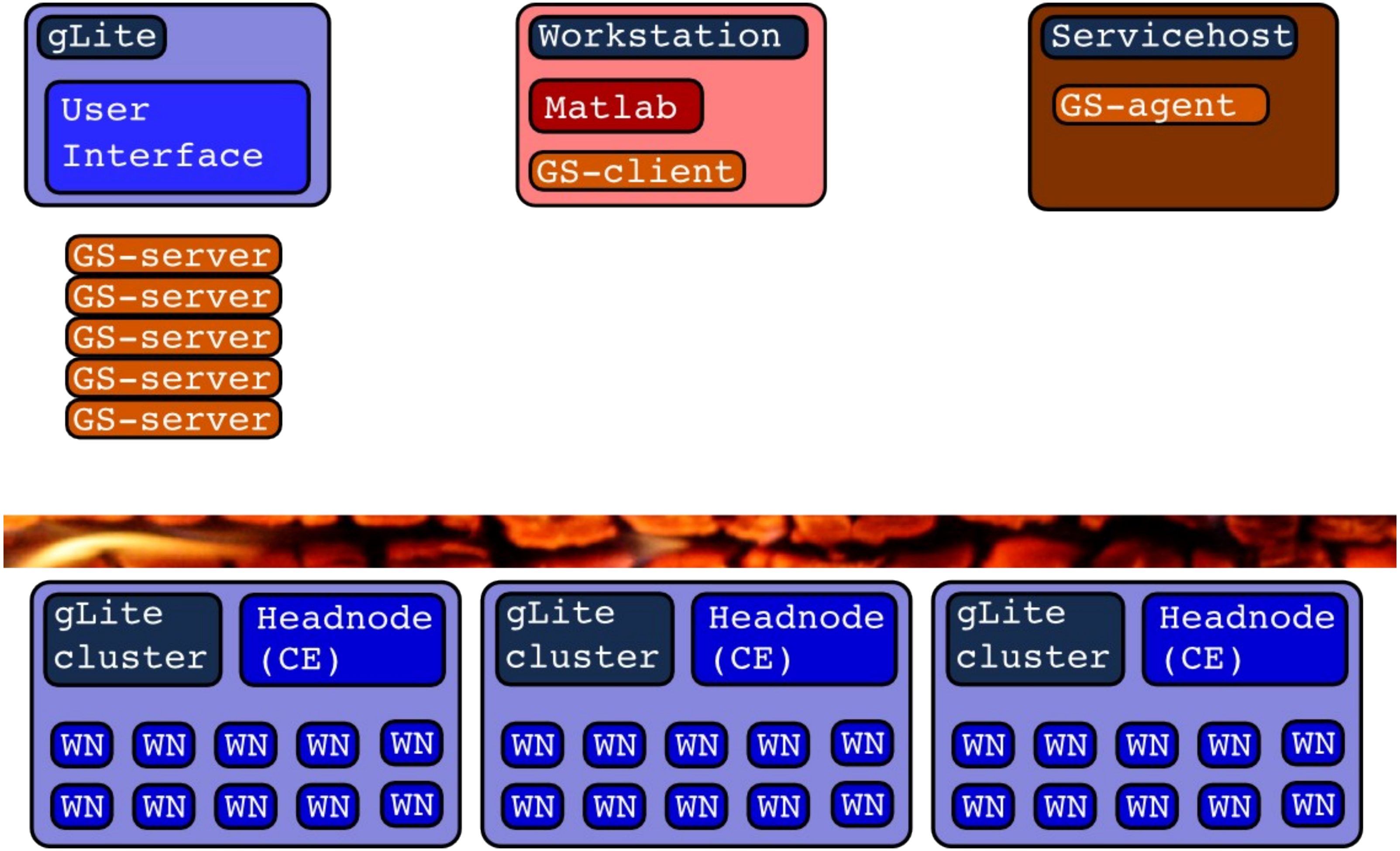
Is interactivity a solution?

Yes!
We submit a cable!!!

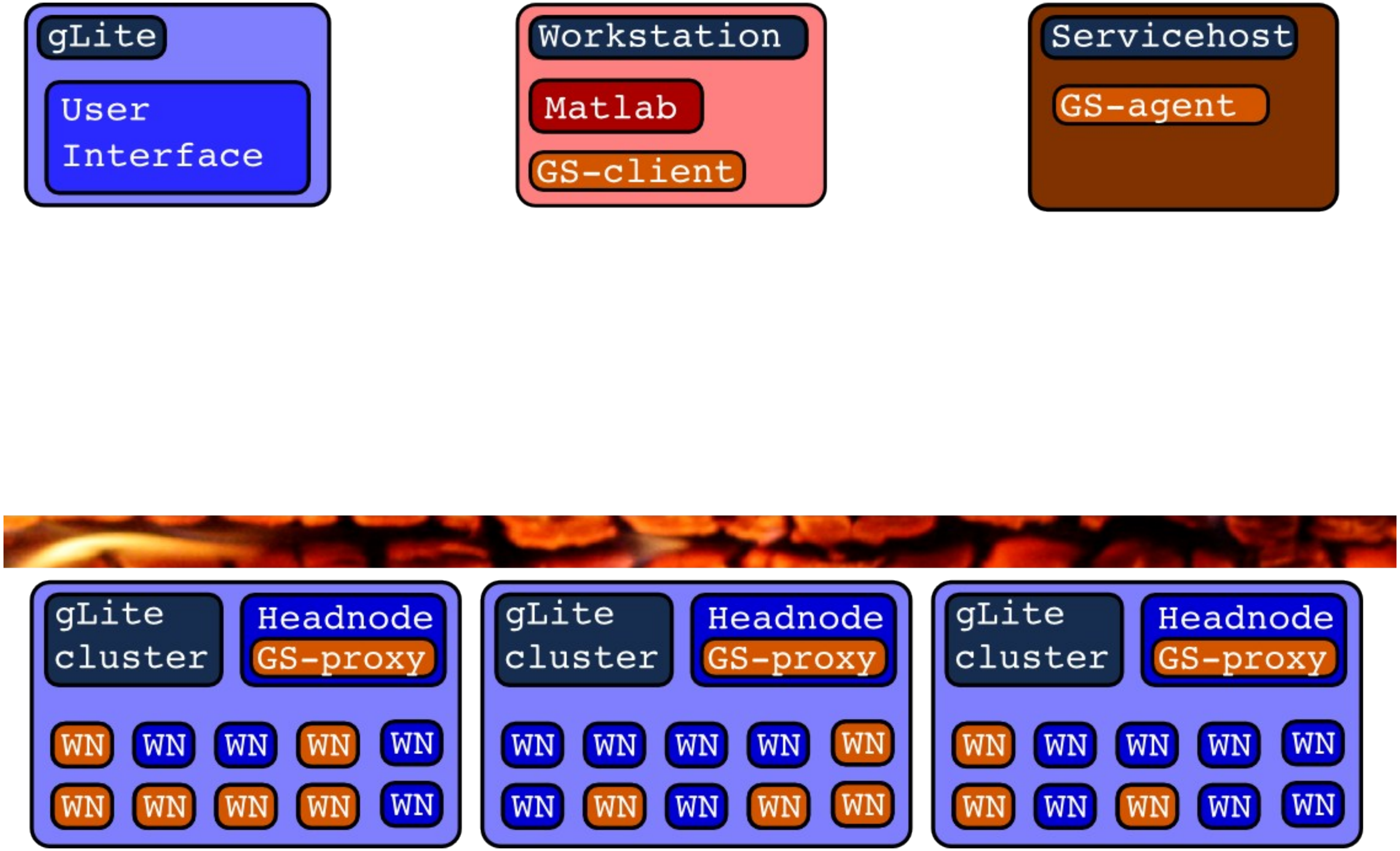
The interactive channel



Our cable: GridSolve



GridSolve ready for action



gLite

User Interface

Workstation

Matlab

GS-client

Servicehost

GS-agent

gLite cluster

Headnode

GS-proxy

WN WN WN WN WN

WN WN WN WN WN

gLite cluster

Headnode

GS-proxy

WN WN WN WN WN

WN WN WN WN WN

gLite cluster

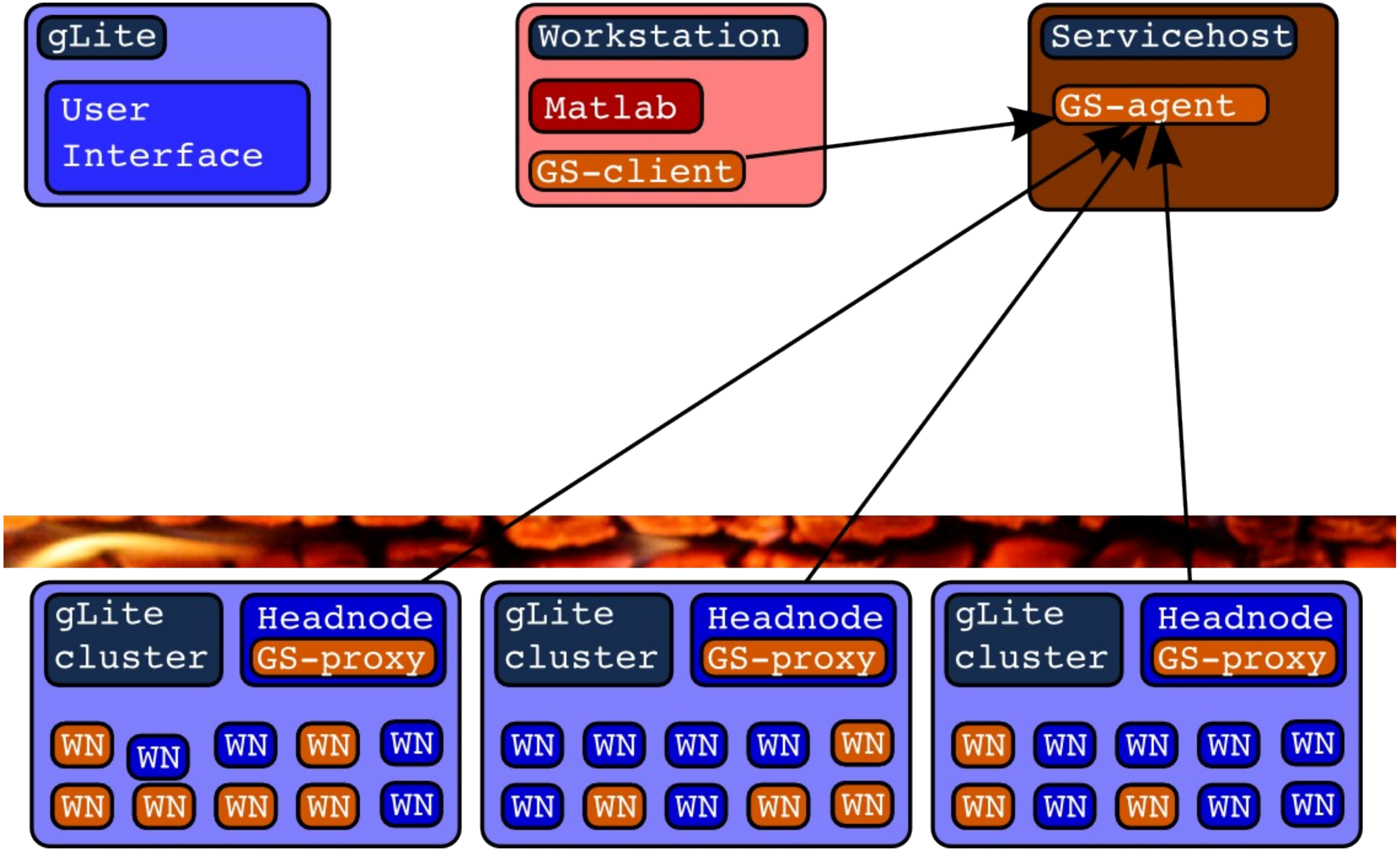
Headnode

GS-proxy

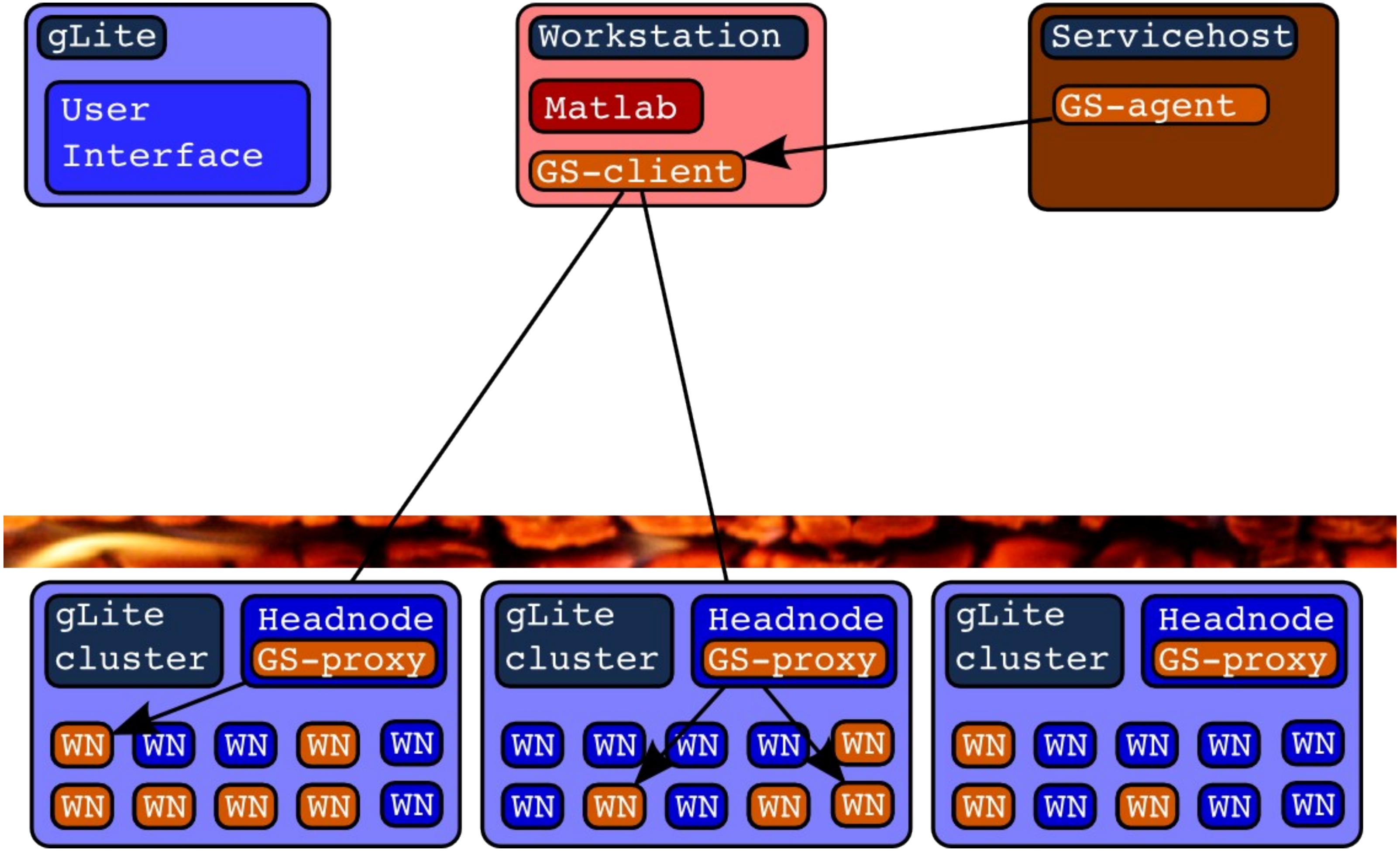
WN WN WN WN WN

WN WN WN WN WN

GridSolve in action



GridSolve in action



GridSolve interface

- Client interface for Java, C, Fortran, **Matlab**, Octave
- Easy to use:

```
y=problem(x) <=> y=gs_call('problem', x)
```

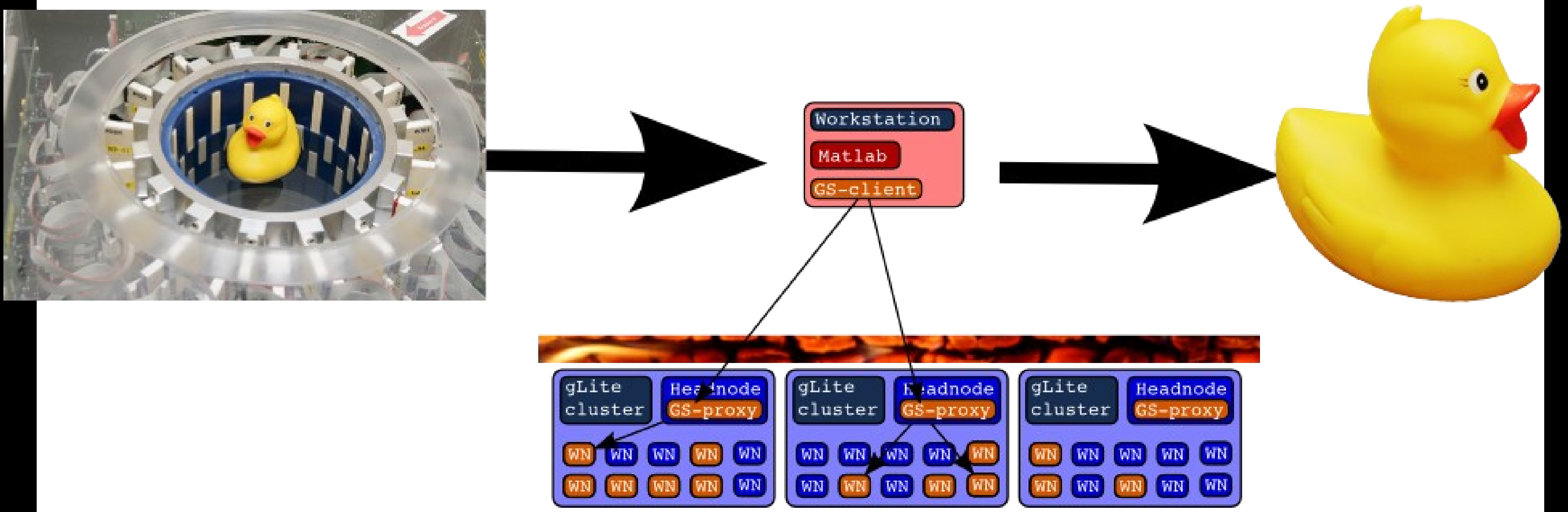
  - Transport input parameters to remote side
  - Execute “problem”
  - Transport result back
- Server limited to C and Fortran
  - Matlab compiler tested => works

=> Reduce complexity of the grid to one function call

GridSolve (GS)/gLite integration

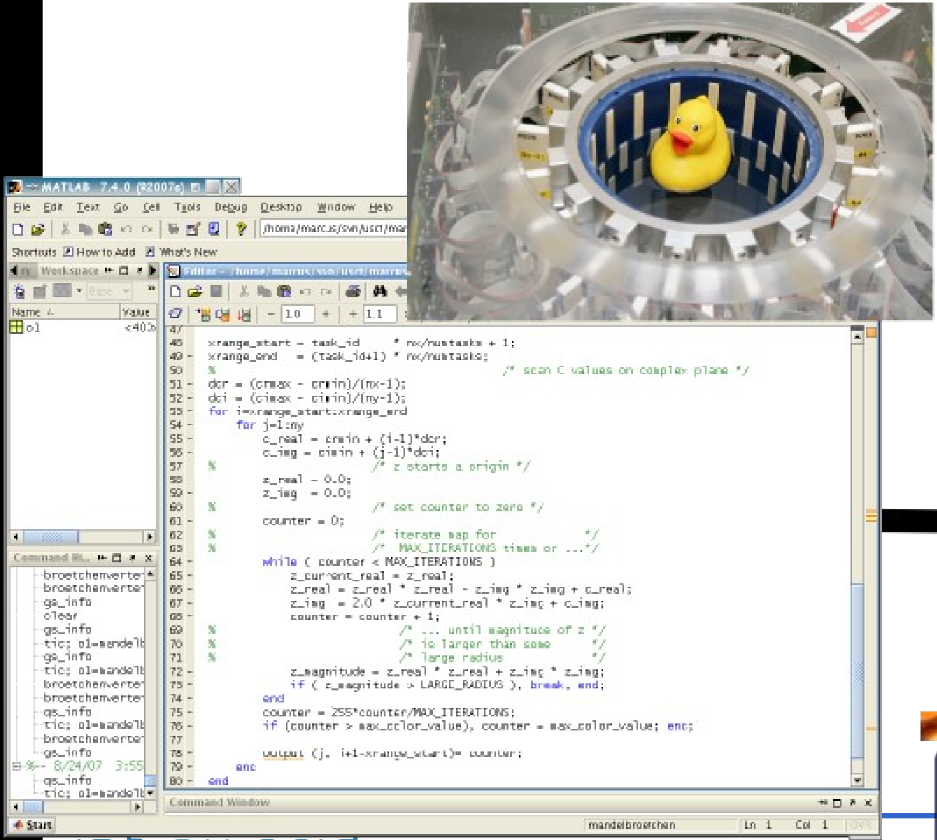
- Send GS-servers to gLite clusters
  - Package GridSolve + My software
  - Send packages into gLite jobs
  - Install packages on WorkerNodes (WN)
- Create GS-service hosts (GS-agent)
- Ensure network connectivity
  - GS-client, GS-agent, GS-proxy, GS-server
- All this happens behind the scenes!

Putting things together

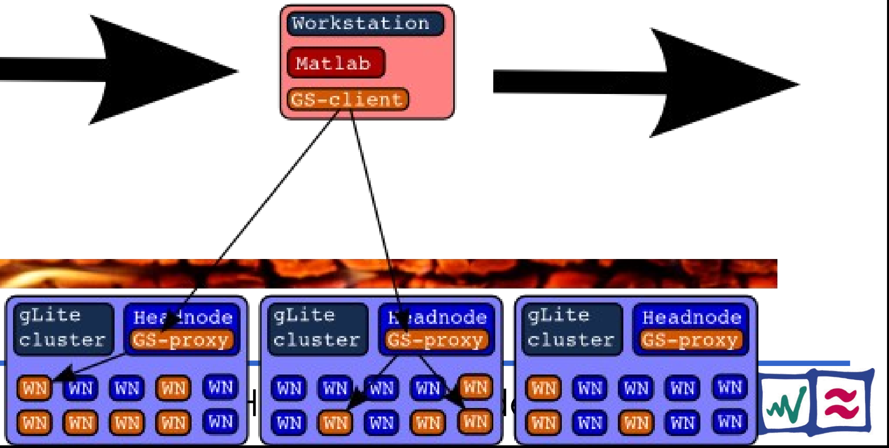


Demonstration

- Simulation: Mandelbrot fractal
- Using the same infrastructure



```
    xrange_start = task_id     * nx/numtasks + 1;
    xrange_end   = (task_id+1) * nx/numtasks;
%                                           /* scan C values on complex plane */
    dcr = (crmax - crmin)/(nx-1);
    dci = (cimax - cimin)/(ny-1);
    for i=xrange_start:xrange_end
        for j=1:ny
            c_real = crmin + (i-1)*dcr;
            c_img = cimin + (j-1)*dci;
%                           /* z starts a origin */
            z_real = 0.0;
            z_img  = 0.0;
%                           /* set counter to zero */
            counter = 0;
%                           /* iterate map for           */
%                           /*  MAX_ITERATIONS times or ...*/
            while ( counter < MAX_ITERATIONS )
                z_current_real = z_real;
                z_real = z_real * z_real - z_img * z_img + c_real;
                z_img  = 2.0 * z_current_real * z_img + c_img;
                counter = counter + 1;
%                               /* ... until magnitude of z */
%                               /* is larger than some      */
%                               /* large radius             */
                z_magnitude = z_real * z_real + z_img * z_img;
                if ( z_magnitude > LARGE_RADIUS ), break, end;
            end
            counter = 255*counter/MAX_ITERATIONS;
            if (counter > max_color_value), counter = max_color_value; end;

            output (j, i+1-xrange_start)= counter;
        end
    end
```

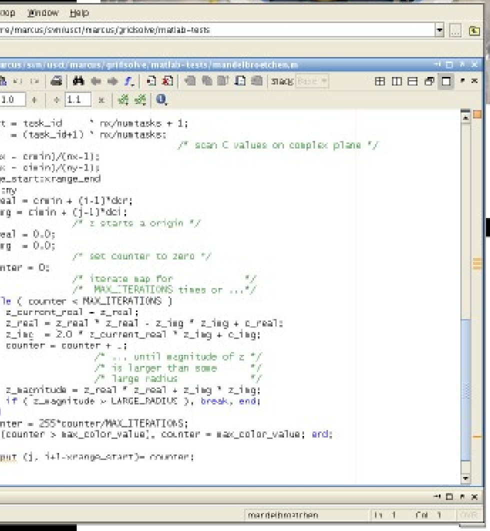


Life-Demo

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

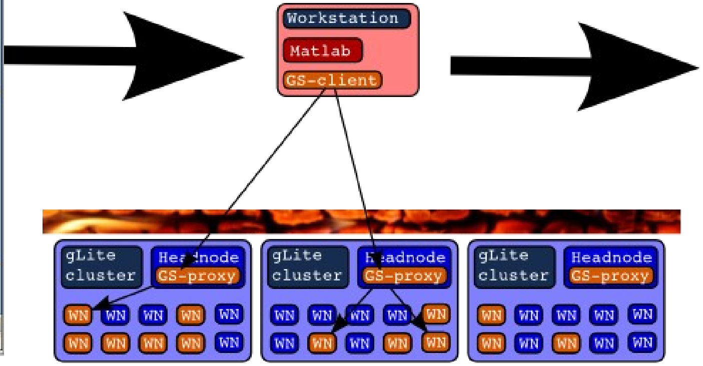
Result

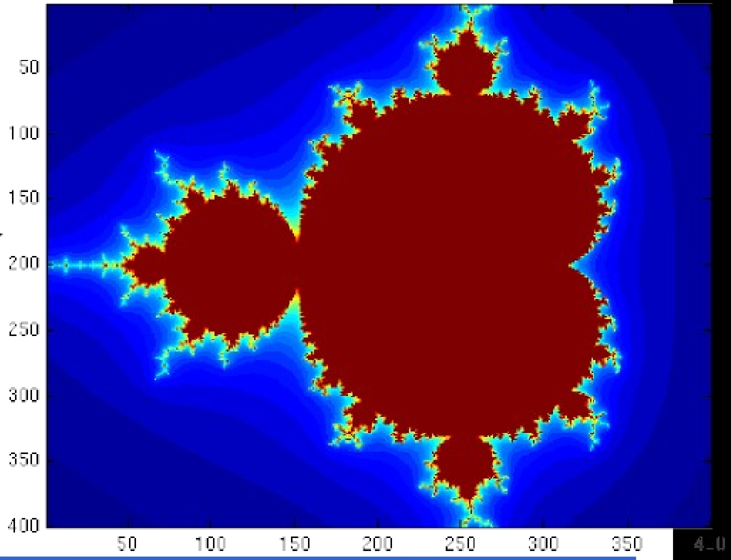
- Simulation works
- Reasonable speedup (4x on 8 machines)



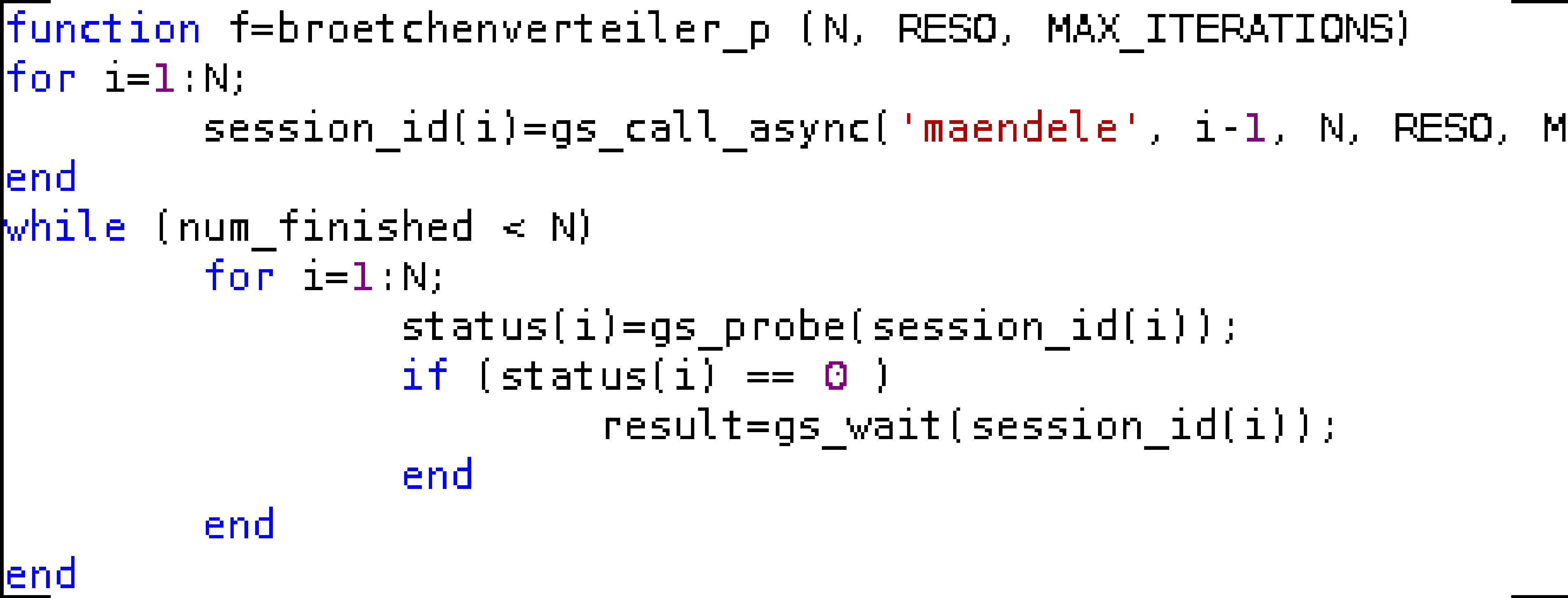
oop Window Help
re/marcus/svn/usct/marcus/gridsolve/matlab-tests
arcus/svn/usct/marcus/gridsolve/matlab-tests/mandelbroetchen.m
Stack: Base
1.0
1.1

```
t = task_id     * nx/numtasks + 1;
  = (task_id+1) * nx/numtasks;
                        /* scan C values on complex plane */
x - crmin)/(nx-1);
x - cimin)/(ny-1);
e_start:xrange_end
:ny
eal = crmin + (i-1)*dcr;
rg = cimin + (j-1)*dci;
         /* z starts a origin */
eal = 0.0;
rg  = 0.0;
         /* set counter to zero */
nter = 0;
         /* iterate map for              */
         /*  MAX_ITERATIONS times or ...*/
le ( counter < MAX_ITERATIONS )
 z_current_real = z_real;
 z_real = z_real * z_real - z_img * z_img + c_real;
 z_img  = 2.0 * z_current_real * z_img + c_img;
 counter = counter + 1;
            /* ... until magnitude of z */
            /* is larger than some      */
            /* large radius             */
 z_magnitude = z_real * z_real + z_img * z_img;
 if ( z_magnitude > LARGE_RADIUS ), break, end;

nter = 255*counter/MAX_ITERATIONS;
(counter > max_color_value), counter = max_color_value; end;

put (j, i+1-xrange_start)= counter;
```





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, M
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
```

Summary

- **Goals for grid access**
  - **Seamless** ✓
  - **Interactive** ✓
  - **From matlab** ✓

- We can
  - Use the grid from Matlab / Fortran
  - Run simple simulations in our infrastructure
- We want to...
  - Use real code
    - Cope with the data (20 GB in, 8 GB out)
  - Automatically send Matlab functions to the grid
  - Explore tighter connected code
    - Use OpenMPI support on interactive grid

interactive grid

What's missing?

- **Goal:**
  - **Seamless** ✗
  - **Interactive** ✗
  - **Grid access** ✓
  - **From matlab** ✗

- Seamless
  - Don't compile standalone application
- Interactive
  - No overhead (< 10 s)
  - No manual data movem
- From Matlab
  - Run Matlab-functions rer

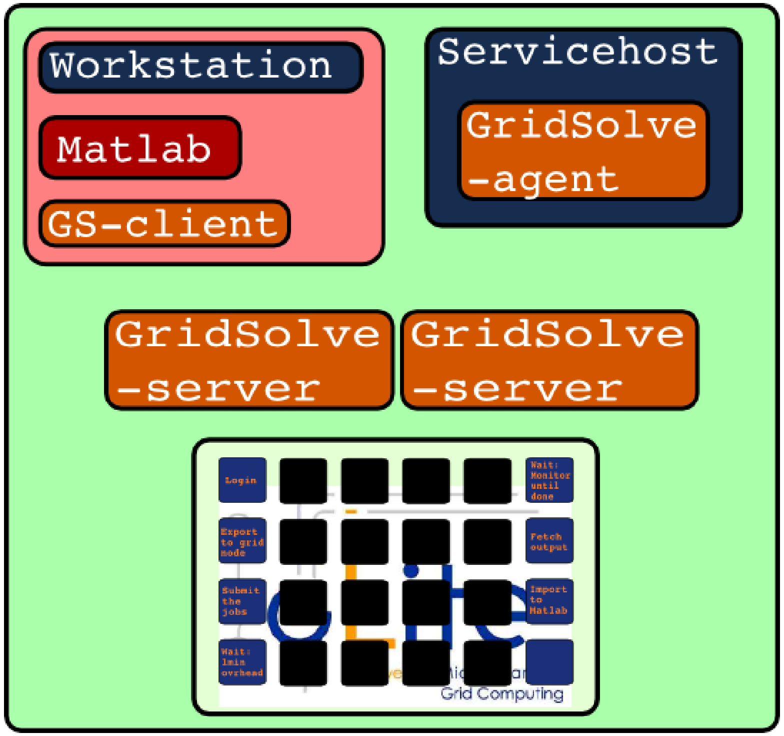
Example:

Large Excel Table

- **Excel must run locally**
- **Computation in the grid**

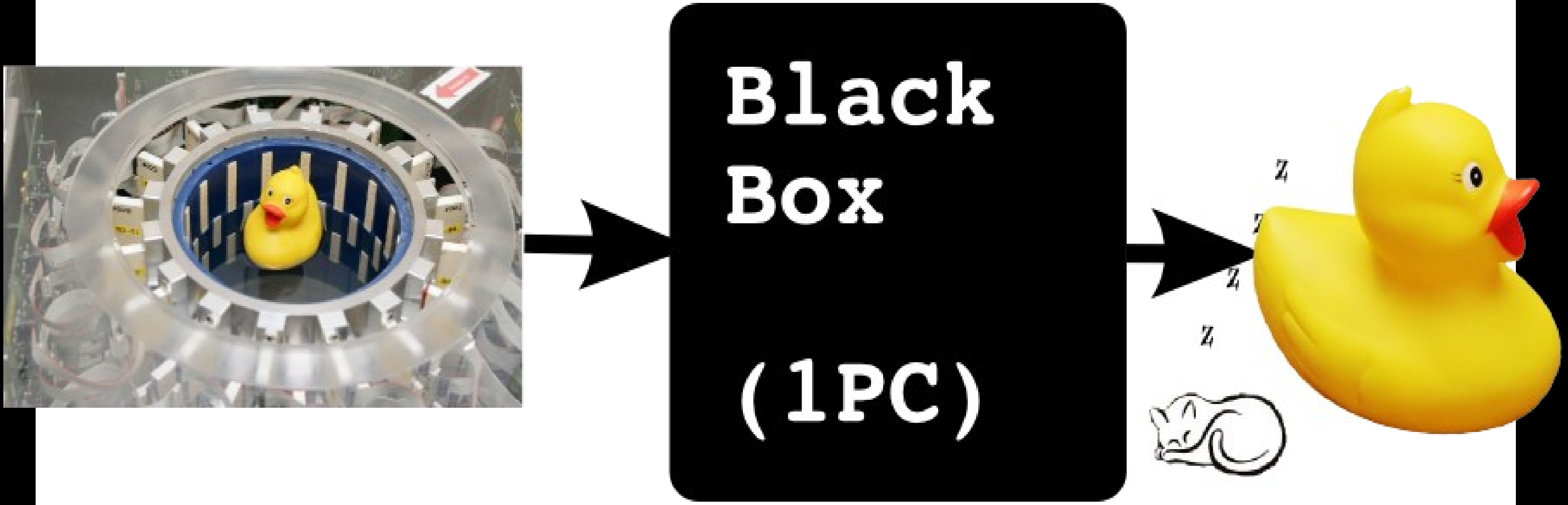
How to do it?

- 1. Make Matlab run on gLite
- 2. Integrate GridSolve with gLite



=> **G**rid **i**n **M**atlab using **G**ridsolv**e** & **R**PC
GIMGER

USCT reconstruction := “Black Box”



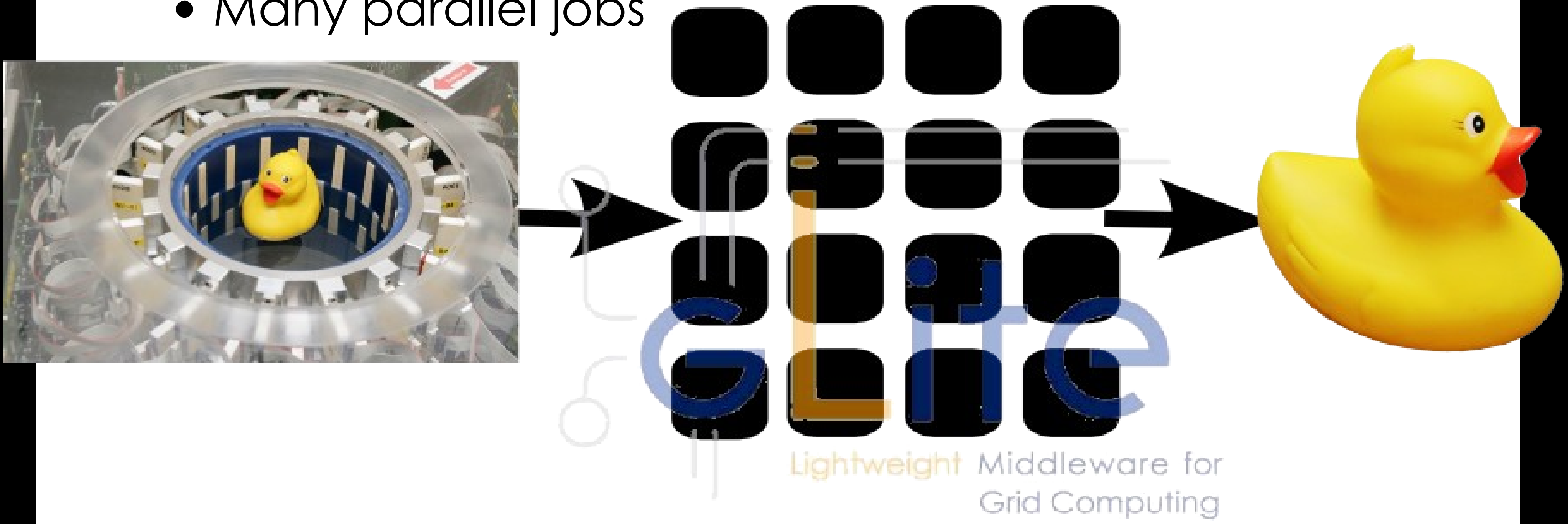
Black Box

(1PC)

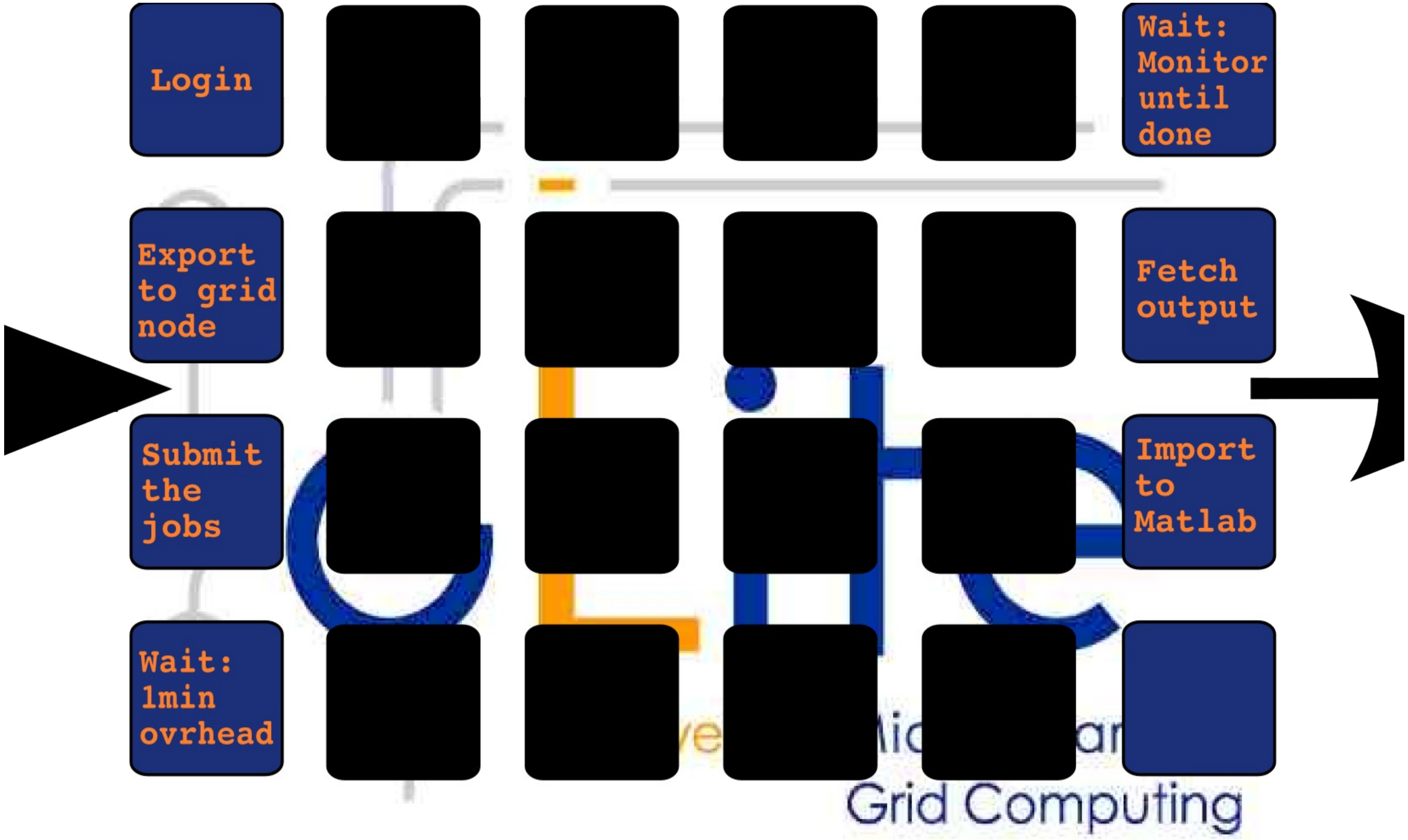
- Computation takes long (days, weeks, years)

Using gLite

- Initial approach to parallel execution:
  - Partitioning of data
  - Many parallel jobs



Using gLite in practise



- A lot of work is left to the user

Using gLite

- Goal:
  - Seamless ✗
  - Interactive ✗
  - Grid access ✓
  - From matlab ✗

What's missing?

- **Goal:**
  - **Seamless** ✗
  - **Interactive** ✗
  - **Grid access** ✓
  - **From matlab** ✗

- Seamless
  - Don't compile standalone application
- Interactive
  - No overhead (< 10 s)
  - No manual data movement
- From Matlab
  - Run Matlab-functions remotely