



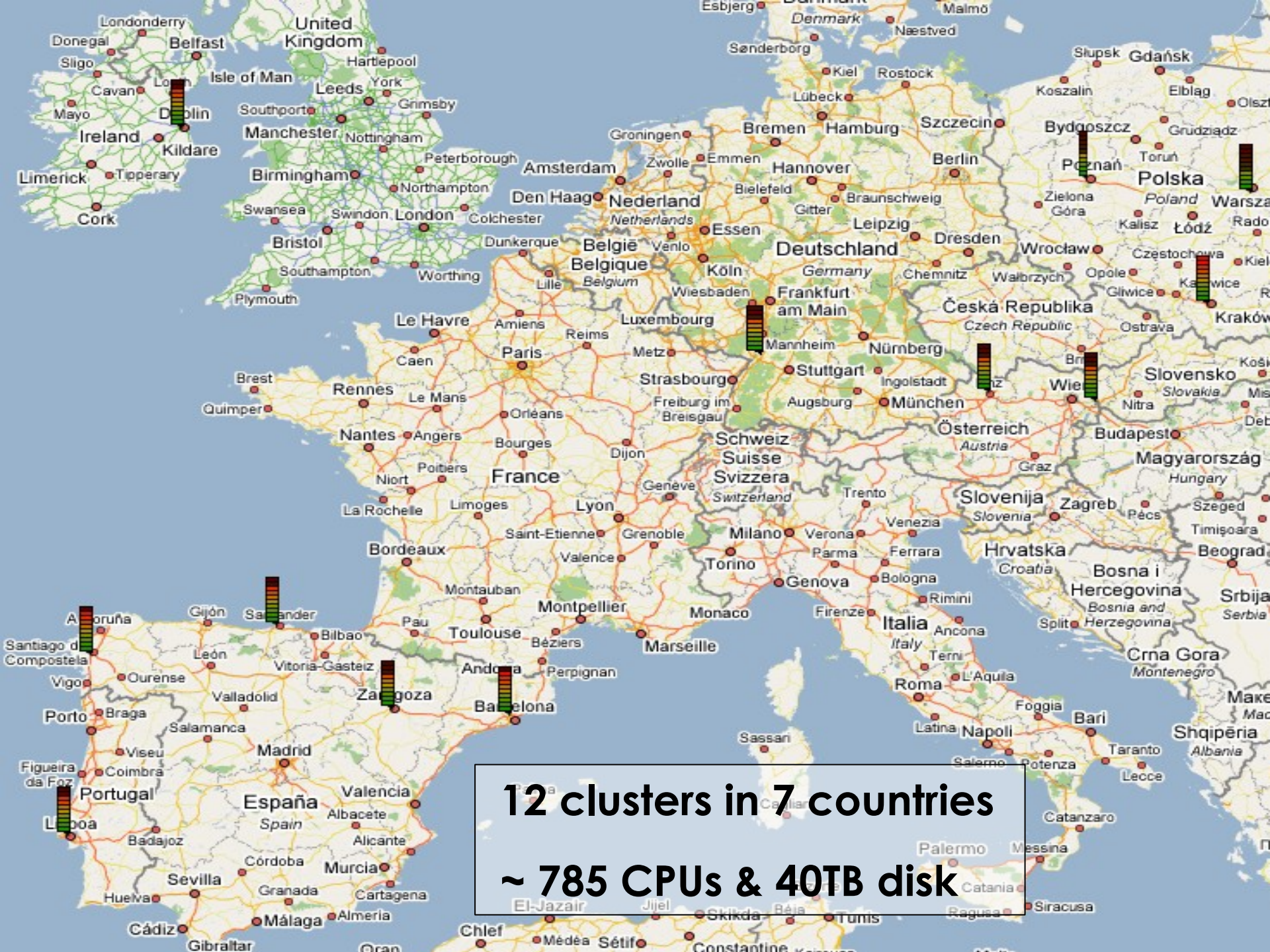
int.eu.grid

<http://www.interactive-grid.eu>



Interactive grid-access for Ultrasound-CT

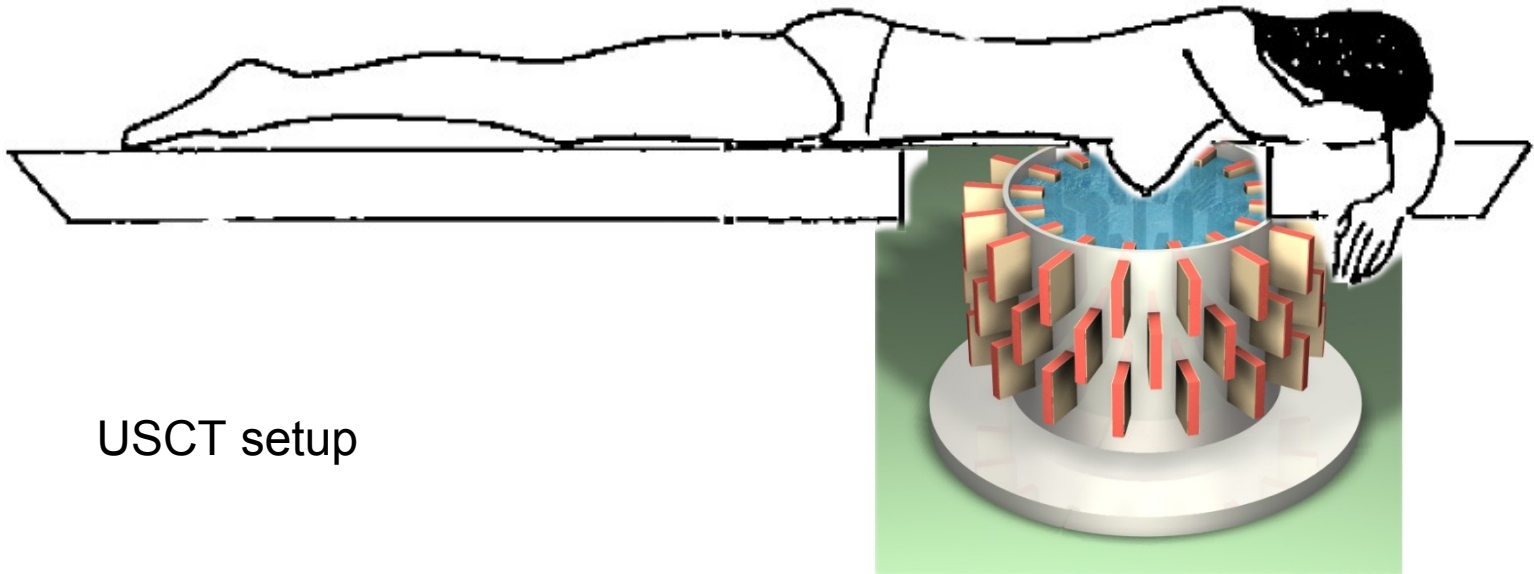
Marcus Hardt
Forschungszentrum Karlsruhe



12 clusters in 7 countries
~ 785 CPUs & 40TB disk

The application

- The application: Ultrasound CT (USCT)
 - New method for medical imaging
 - Application: Breast cancer diagnosis

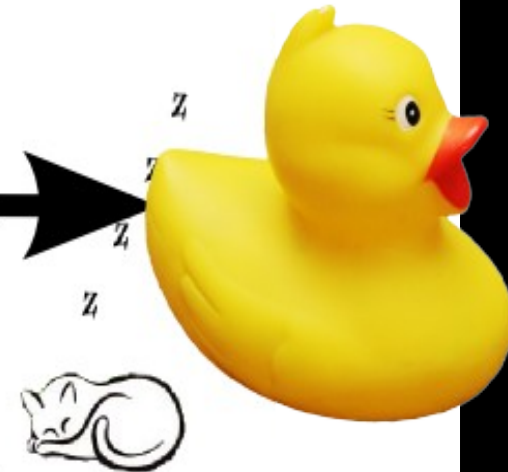
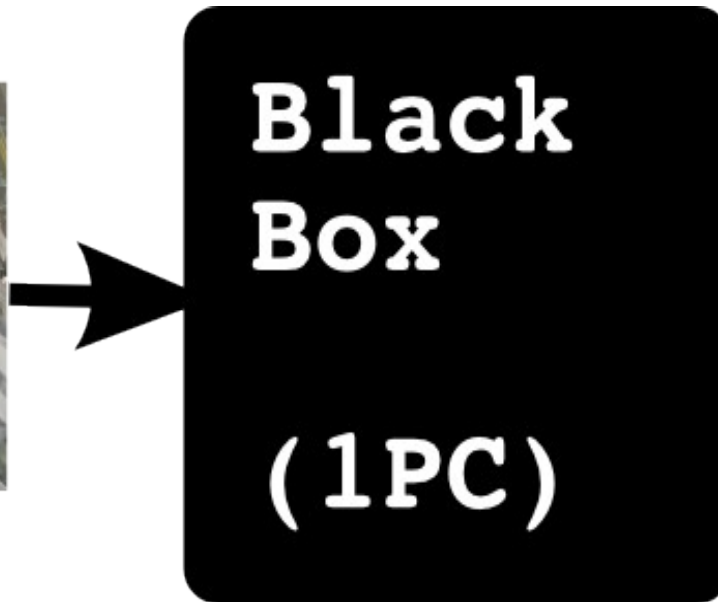
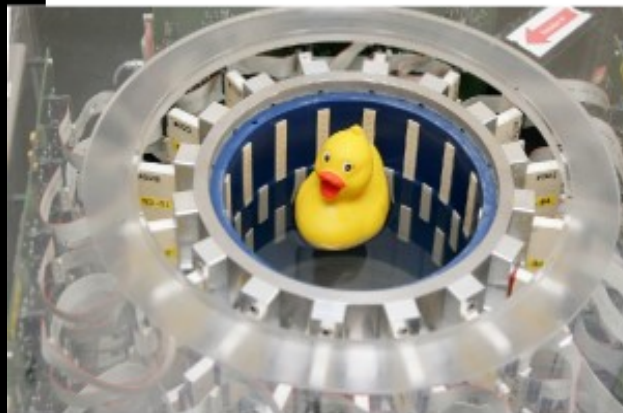


USCT setup

USCT Reconstruction := “Black Box”

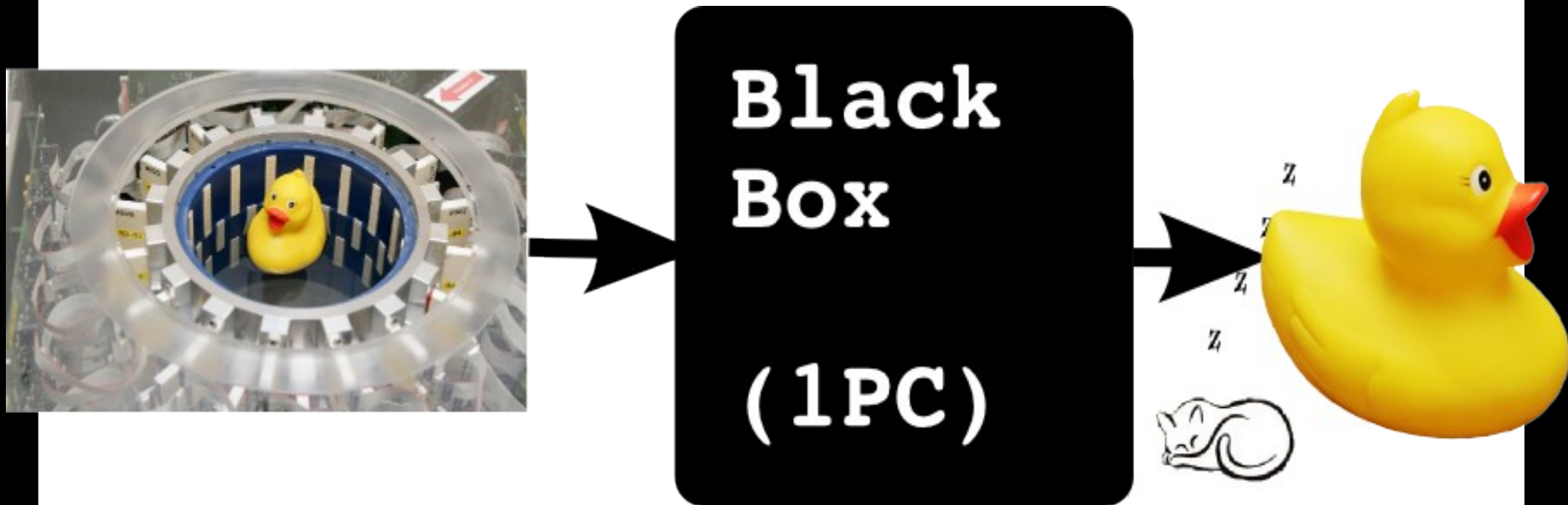
- Algorithm:
 - Based on ellipsoidal backprojection (SAFT)
 - Converts ultrasound signals to 3D volume graphics
 - Input: ~ 20GB
 - Output: ~ 8GB
 - Computing time:
 - $4096^2(2D)$... $128^2 \times 100$... $4096^2 \times 3410$
<=> 1 hour ... 1.5 Months... 150 Years
- Matlab
 - Problem solving environment
 - similar to Maple, Mathematica, Scilab ...
 - Strategic development platform
 - Not easy to “submit matlab to the grid”

USCT Reconstruction



- Computation takes long (days, weeks, years)

USCT Reconstruction



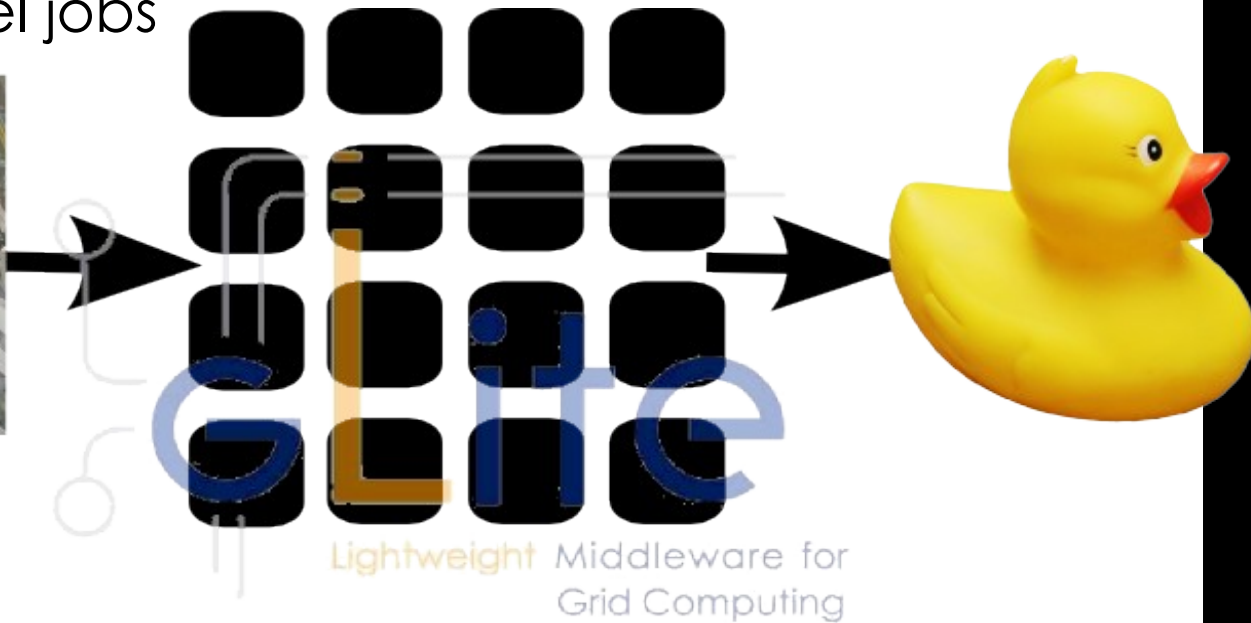
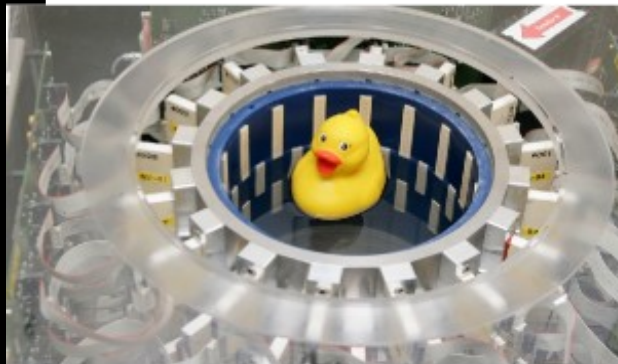
- Computation takes long (days, weeks, years)

- **Goal:**

- Seamless, interactive, grid access
- from Matlab

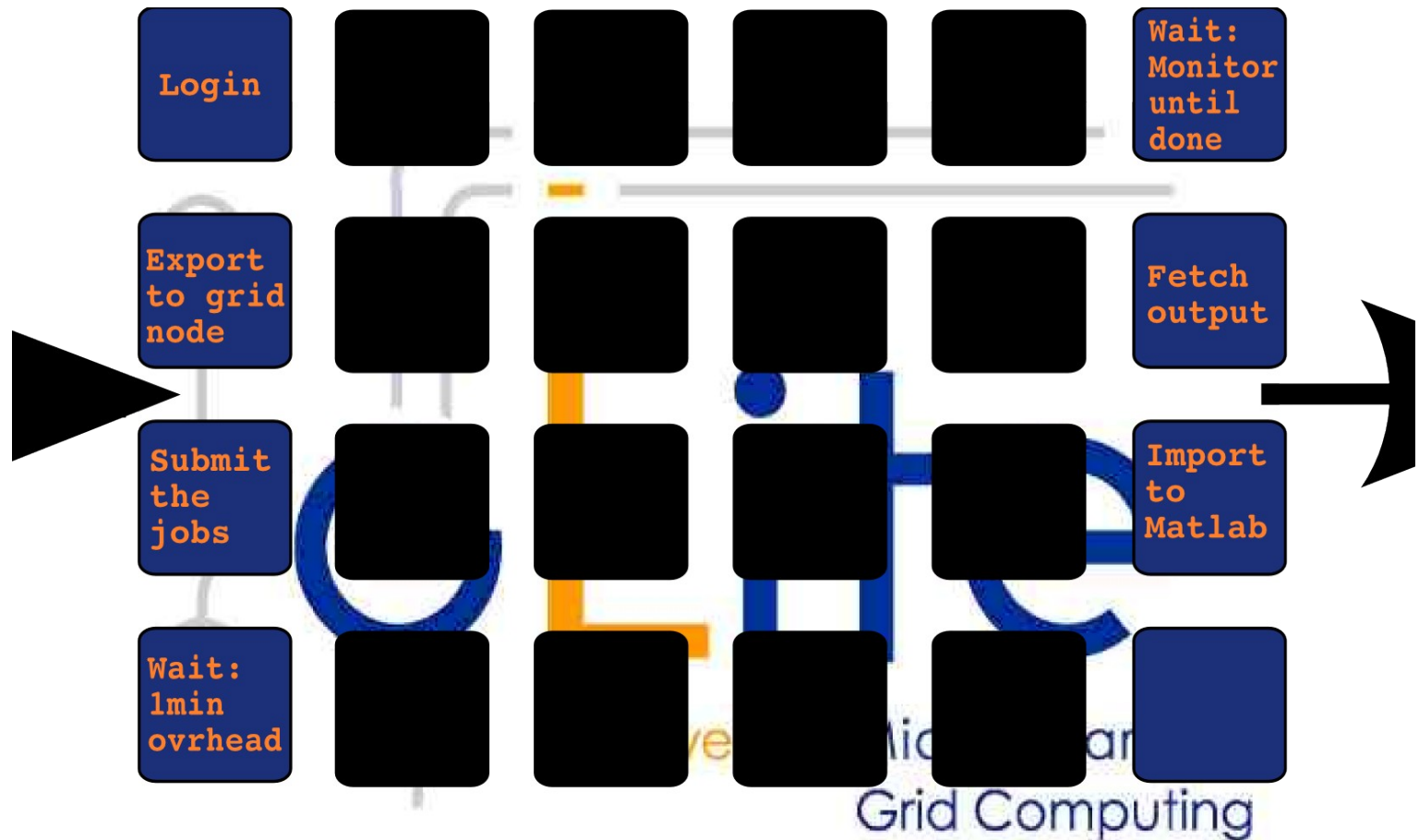
Using the grid

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



Using the grid

- Lets take a close look



Using the grid

- **Goal:**

- Seamless 
- interactive 
- grid access 
- from Matlab 

Using the grid

- **Goal:**
 - Seamless 
 - interactive 
 - grid access 
 - from Matlab 

Usability-test:

=> The users will run away



What's missing?

- **Goal:**

- Seamless 
- interactive 
- grid access 
- from Matlab 

- Seamless

- User might not know if he uses the grid

- Interactive

- No overhead (< 10 s)
- No manual data movement

- From Matlab

- Run Matlab-functions remotely

Improving Grid Access with RPC



- GridSolve

- Developed at ICL, University Tennessee, Knoxville
- Implements an RPC client/server solution
- 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

=> Reduce complexity of the grid to one function call

How to do it?

- **Goal:**

- Seamless 
- Interactive 
- Grid access 
- For scientists 

- 1. Integrate GridSolve with gLite
 - **GridSolve** integration with **gLite**
 - **GIGGLE**
- 2. Make Matlab run on gLite
 - **Grid** in **Matlab** using **Gridsolve** & **RPC**
 - **GIMGER** (speak: ginger)

GridSolve(GS)/gLite Integration

- Create GS-Service hosts (GS-agent + GS-proxy)
- Send 100s of GS-servers to gLite infrastructure
 - Build infrastructure
 - Package GridSolve + Matlab Runtime
 - Install GridSolve + Matlab on WNs
- Ensure network connectivity
 - GS-Server \Leftrightarrow GS-Proxy \Leftrightarrow GS-agent \Leftrightarrow GS-client

Matlab/gLite integration

- Matlab Compiler (toolbox)
=> Matlab Compiler Runtime (MCR)
 - Install on the fly (as part of glite-job)
 - Fix linux glibc version incompatibility
 - Install new glibc on the fly
- Usability enhancement
 - Access GridSolve from Matlab
 - Point Matlab to service hosts
 - Support for RPC creation
 - Compilation/Linking/Deployment

GridSolve startup on gLite

gLite

User Interface

Workstation

Matlab

GS-client

Servicehost

GS-agent

- GS-server
- GS-server
- GS-server
- GS-server
- GS-server



gLite CE

WN WN WN WN WN

WN WN WN WN WN

gLite CE

WN WN WN WN WN

WN WN WN WN WN

gLite CE

WN WN WN WN WN

WN WN WN WN WN

GridSolve ready for action

gLite

User Interface

Workstation

Matlab

GS-client

Servicehost

GS-agent



gLite CE

GS-proxy

WN WN WN WN WN

WN WN WN WN WN

gLite CE

GS-proxy

WN WN WN WN WN

WN WN WN WN WN

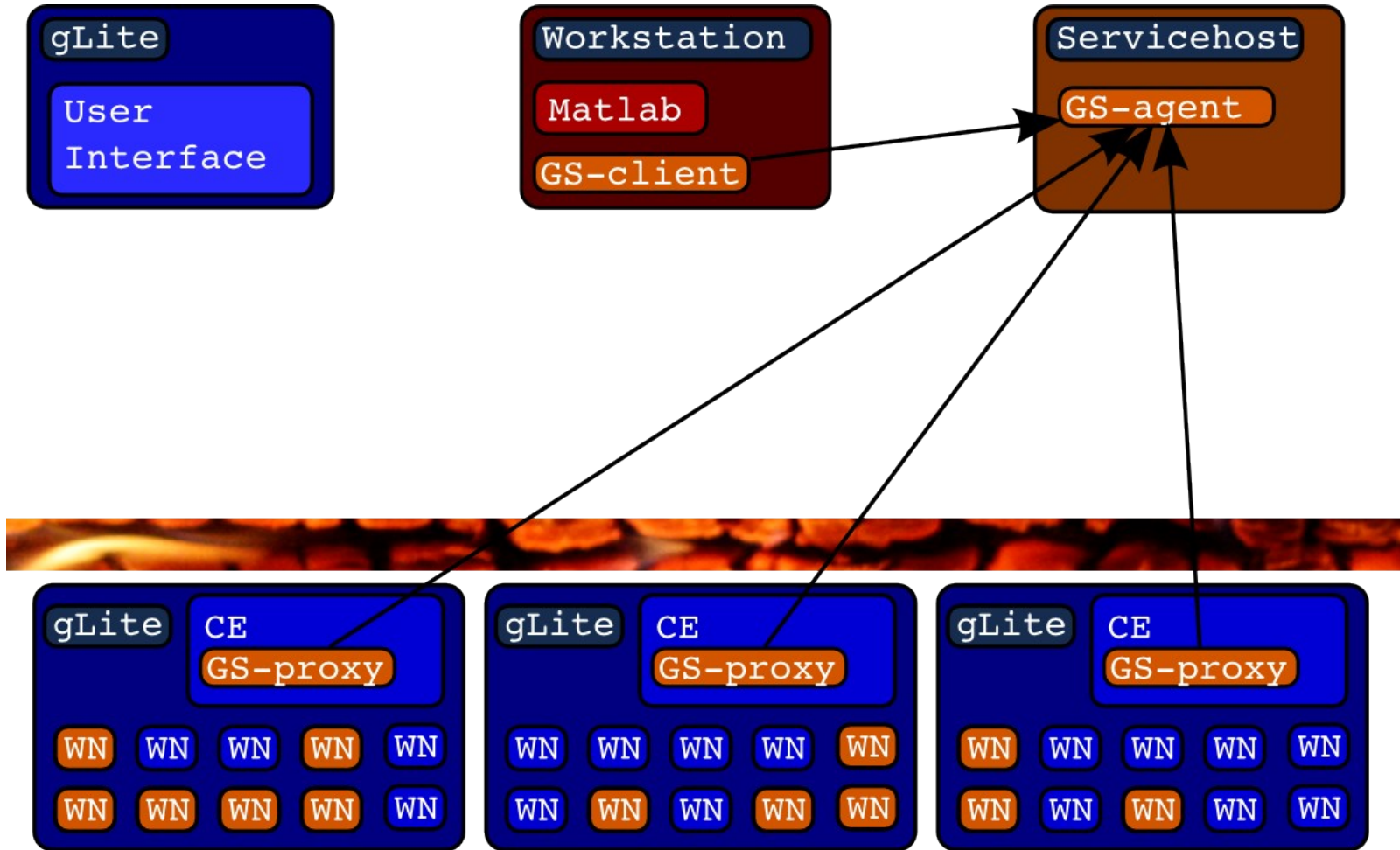
gLite CE

GS-proxy

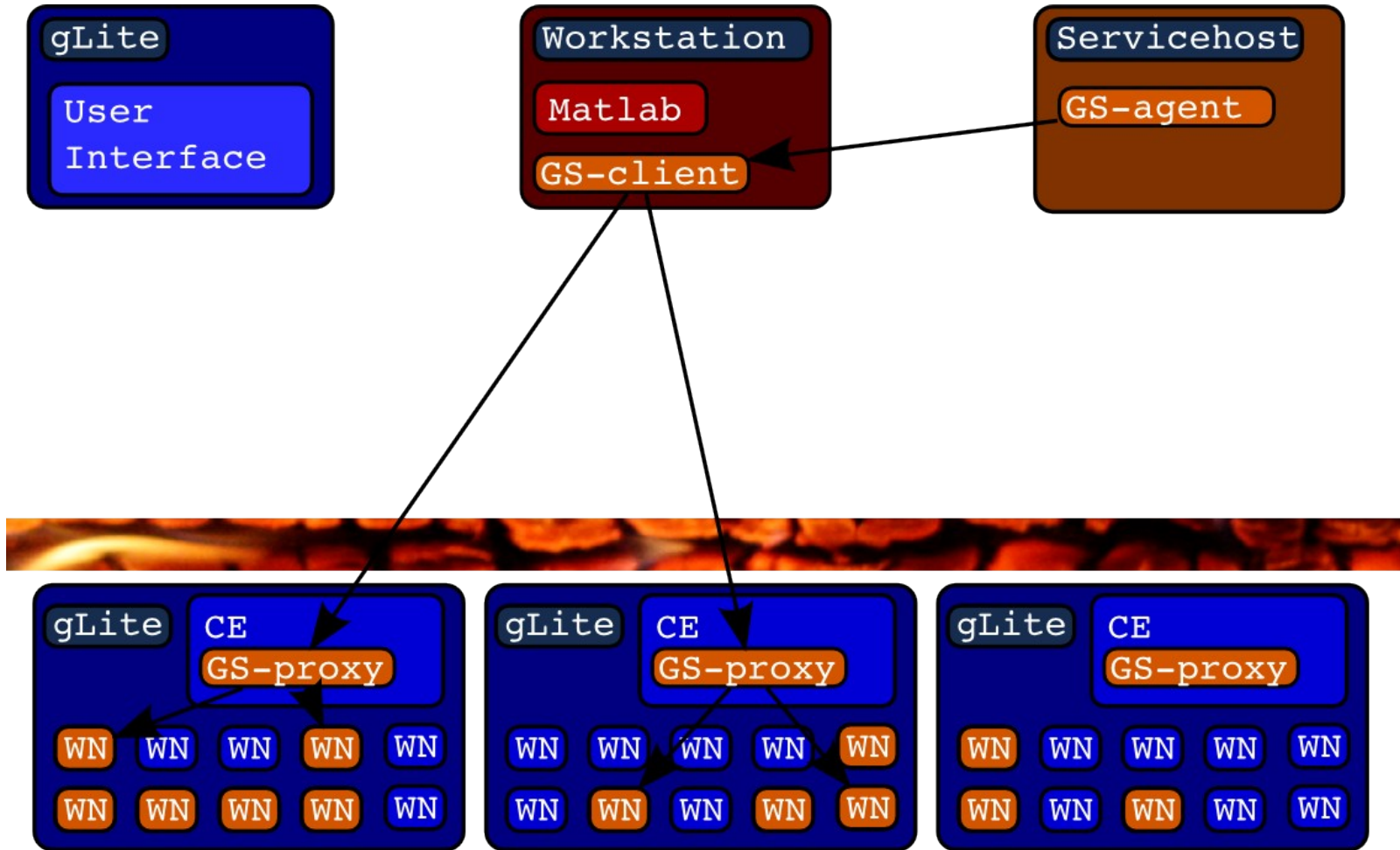
WN WN WN WN WN

WN WN WN WN WN

GridSolve ready in action

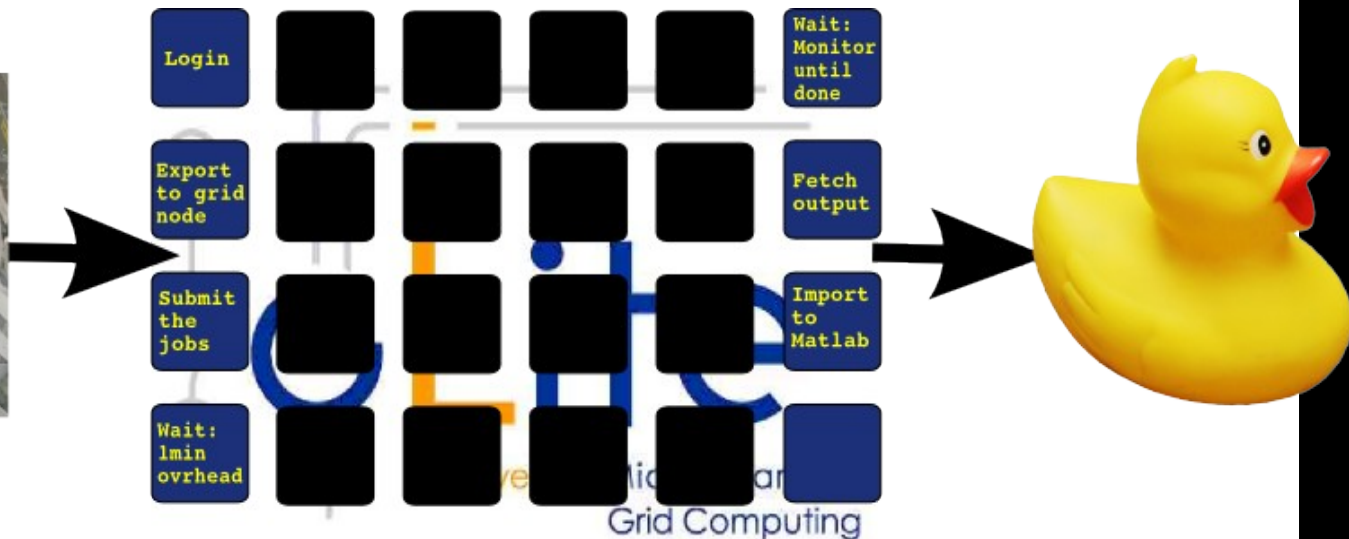
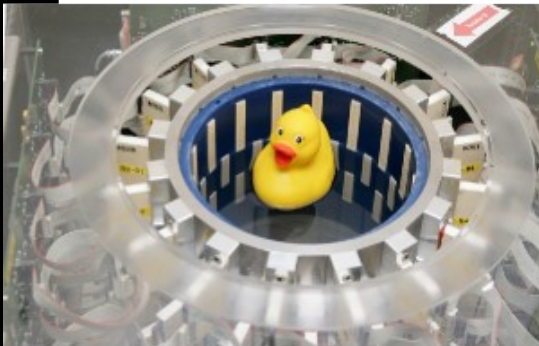


GridSolve ready in action



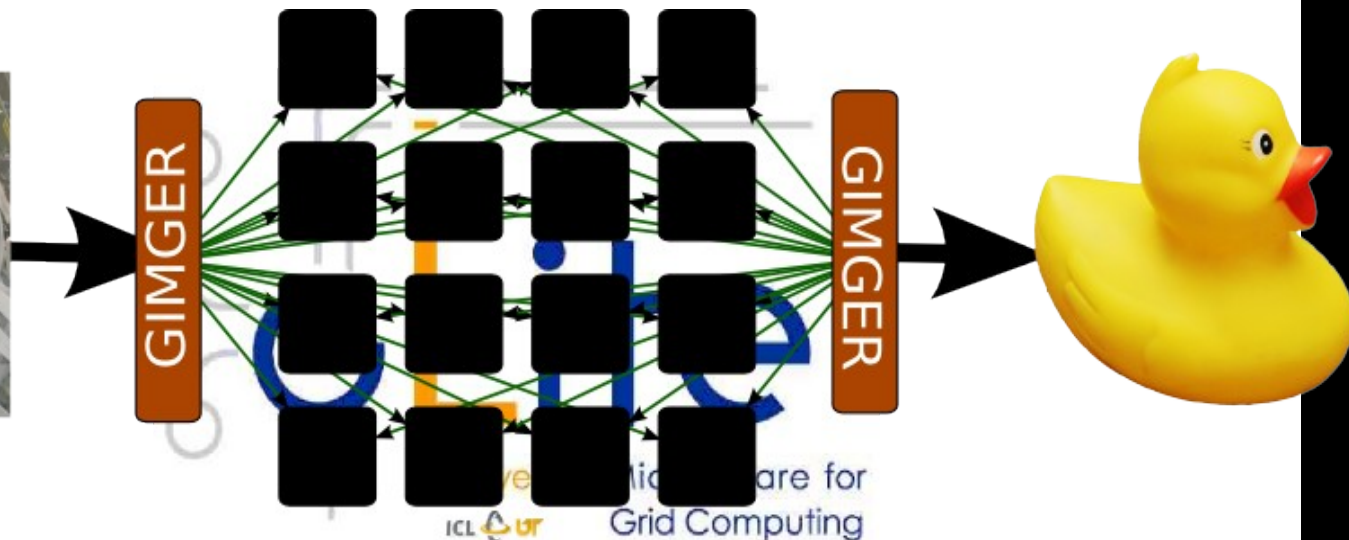
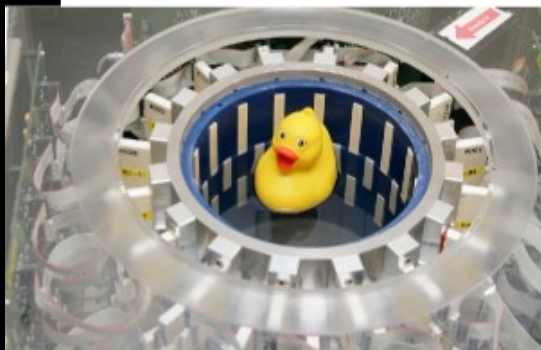
Putting things together

- RPC with GridSolve
- On top of int.eu.grid/gLite
- Using Matlab functionality
- GIMGER



Putting things together

- RPC with GridSolve
- On top of int.eu.grid/gLite
- Using Matlab functionality
- GIMGER



Demonstration

- Simulation: Mandelbrot fractal
- Using the same infrastructure

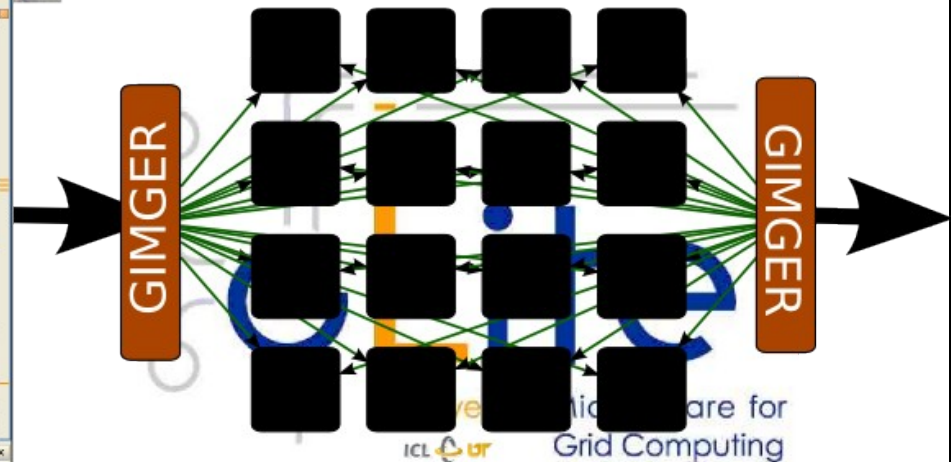


```

MATLAB 7.4.0 (R2007a)
File Edit Text Go Cell Tools Debug Desktop Window Help
/home/marcus/softst/marcus/gridsoe/matlab-tests

Workspace:
Name Value
o1 <400>

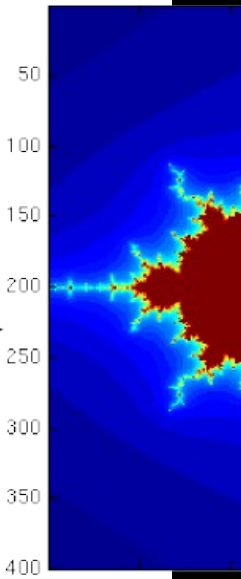
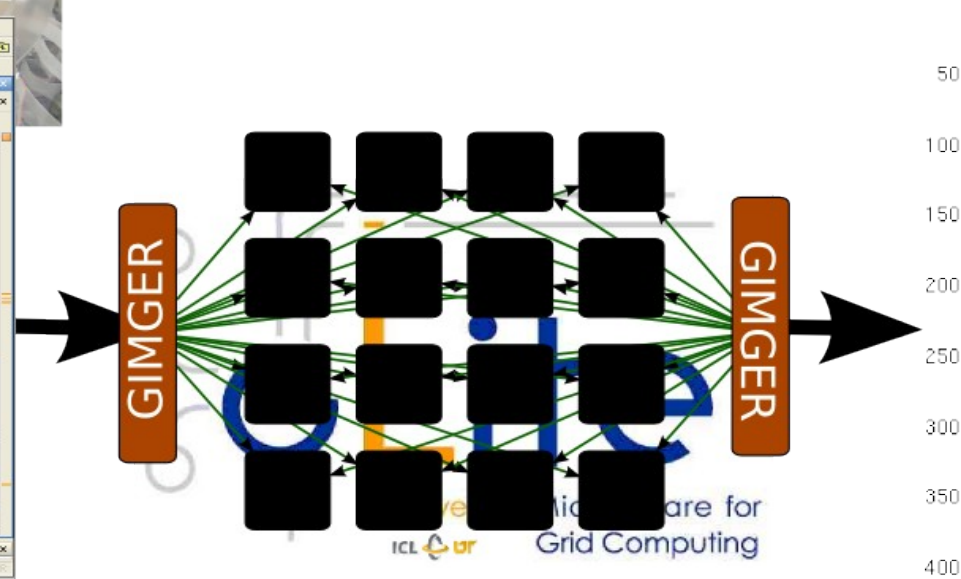
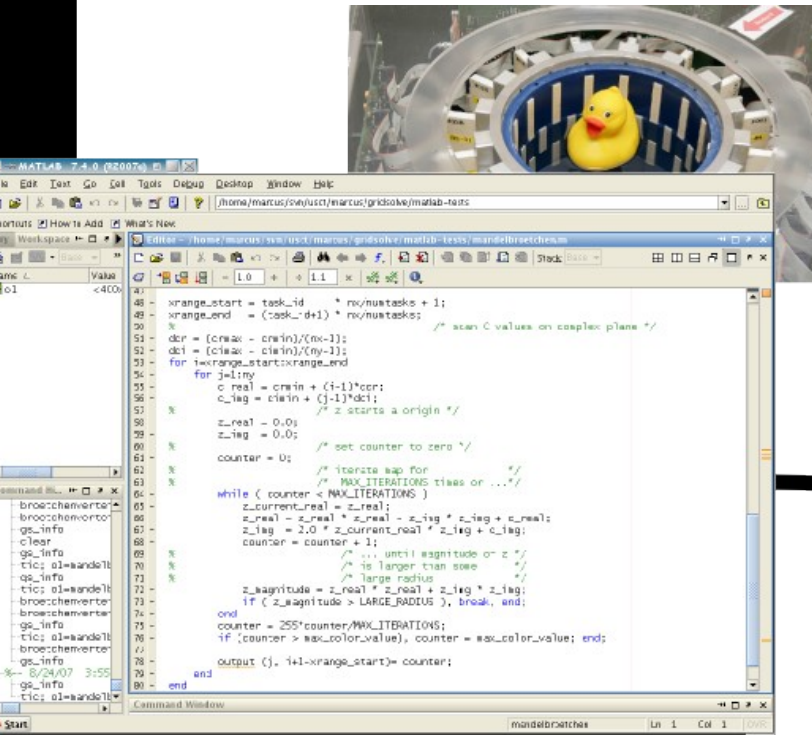
Command Window:
48 xrange_start = task_id \ n\ n\ ntasks + 1;
49 xrange_end = (task_id+1) \ n\ n\ ntasks;
50 %
51 rcr = (cmax - cmin)/(nx-1); /* scan C values on complex plane */
52 dci = (ci_max - ci_min)/(ny-1);
53 for i=xrange_start:xrange_end
54     for j=1:ny
55         c_real = cmin + (i-1)*rcr;
56         c_img = ci_min + (j-1)*dci; /* z starts a origin */
57 %
58 z_real = 0.0;
59 z_img = 0.0;
60 %
61 counter = 0; /* set counter to zero */
62 %
63 /* iterate esp for
64 /* MAX_ITERATIONS times or ... */
65 while ( counter < MAX_ITERATIONS )
66     z_current_real = z_real;
67     z_real = z_real * z_real - z_img * z_img + c_real;
68     z_img = 2.0 * z_current_real * z_img + c_img;
69     counter = counter + 1;
70 % ... until magnitude of z */
71 % is larger than some
72 % large radius
73 z_magnitude = z_real * z_real + z_img * z_img;
74 if ( z_magnitude > LARGE_RADIUS ), break, end;
75 end
76 counter = 255*counter/MAX_ITERATIONS;
77 if (counter > max_color_value), counter = max_color_value; end;
78 output (j, i-1*xrange_start)= counter;
79 end
80 end
Command Window
mandelbrotschen
  
```



- Movie of the life demonstration:
 - <http://marcus.hardt-it.de/grid4matlab>
- **Real** life demo on int.eu.grid
 - Talk to me (any time during the breaks)

Result

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



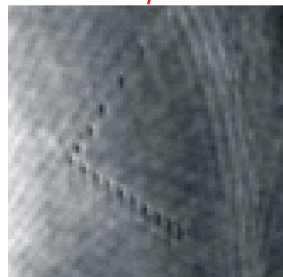
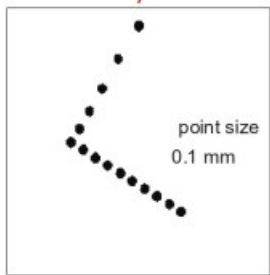
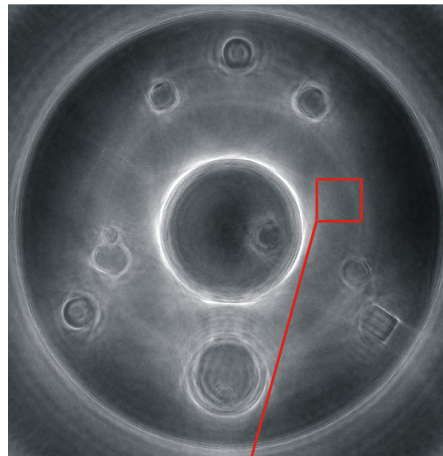
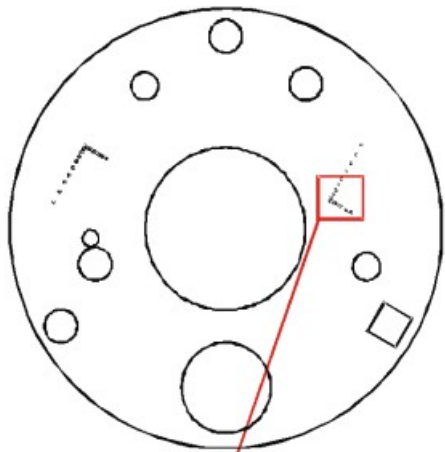
- We can
 - Convert Matlab functions to run on the grid
 - Involved hands-on work
 - Automatic: current work
 - Run simple simulations in our infrastructure
- We want to...
 - Use real code
 - Automatically send Matlab functions to the grid
 - Reduce hands-on work
 - Data Handling (GFAL)
- Interactive Grid
 - Submit from MD (better: from Matlab via RAS)
 - Interactive channel for debugging
 - MPI for collecting output data



USCT Images

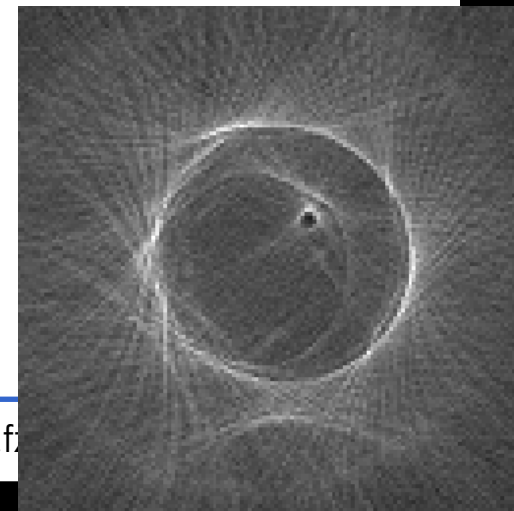
First results with old USCT:

- 0.1 mm Nylon threads visible



Current results with new hardware:

- EGG & Yolk visible
- 3D imaging



- Idea: Remote Procedure Calls (RPC)
 - Submit daemon(s) as glite job(s)
 - Integrate client into Matlab
 - Connect to daemon(s) from client
 - Call remote procedures from client
 - Transfer input/output parameters
- Advantages:
 - “glite-submit-penalty” only for startup
 - Interactive answer via direct network connection
- Disadvantages:
 - Implement an RPC solution....