Minimum intrusion GRID

Build one to throw away

...
So, in a given time frame, plan to achieve something worthwhile in half the time, throw it away, then rebuild what has to be a better version, given the advantages of hindsight.

MiG

- Minimum intrusion GRID
- This really is the philosophy
  - We want to develop a GRID middleware that makes as few requirements as possible

What's wrong with the GLOBUS model?

- Single point of failure
  - There may be redundancy
  - But algorithmically these collapse onto one component
  - We need a model where liveliness scales with the system-size

What's wrong with the GLOBUS model?

- Lack of scheduling
  - Parallel submissions end up on the same resource
  - No means of addressing changes to timing that originates from the local queuing system
  - We need real scheduling with online properties

What's wrong with the GLOBUS model?

- Poor scalability
  - Paradox: The more resources Grid has the faster your job can be completed – but the time to submit your job grows linearly
  - If you assume that the number of users grow with the size of the Grid, the users actually looses from larger Grid's

What's wrong with the GLOBUS model?

- No means of implementing privacy
  - Users need to authenticate themselves to the resource
  - Users contact the resources directly
  - Some users are concerned with exposing what they are working on
  - We need to allow anonymity

What's wrong with the GLOBUS model?

- No means of utilizing 'cycle-scavenging'
  - The Globus model expect the machine to be dedicated
  - And be available for a predefined period
  - Condor + NorduGrid do address this
    - But not with deadlines
  - We would like a model that uses cycle-scavenging but with deadlines

What's wrong with the GLOBUS model?

- Require a very large installation on each resource and on the user site
  - The resource software is huge
  - The client software is pretty large
    - Though ARC-lib limits the client side enormously
  - We need a model that require **no** software to be installed on either resource or client side

What's wrong with the GLOBUS model?

- Firewall dependency
  - Globus requires a number of ports to be opened in the firewall
    - 2135, 2811-14, 9000-9300
  - System administrators are very reluctant to do this
  - We need a setup that exists within the confines of a standard setup

What’s wrong with the GLOBUS model?

- Highly bloated middleware
  - The middleware has become huge
  - There are large amounts of dead code
  - Much replicated code
  - We need middleware that is more compact and maintainable

What's wrong with the GLOBUS model?

- Complex implementation using multiple languages
  - Mutiple languages are used in Globus-2
  - None (few) seems to be well chosen
    - Much C code – but the functions are not system related or performance limited
    - Some Perl code – actually well chosen for the functions but unreadable and hard to maintain

MiG Idea

- MiG should address all the previous concerns
- GRID should be a system
  - not just a protocol between sites
- By having a set of servers that **are** GRID we get full control of GRID
  - we upgrade all machines at the same time
- Users and resources do not have to maintain anything

The desired user view of MiG

- MiG should be a virtual computer to the user
- Files are stored within MiG – each user has his own home-directory
  - In job descriptions files are relative to the users home-directory
- Eventually we'd like interactive application support
- Economically MiG should be fair

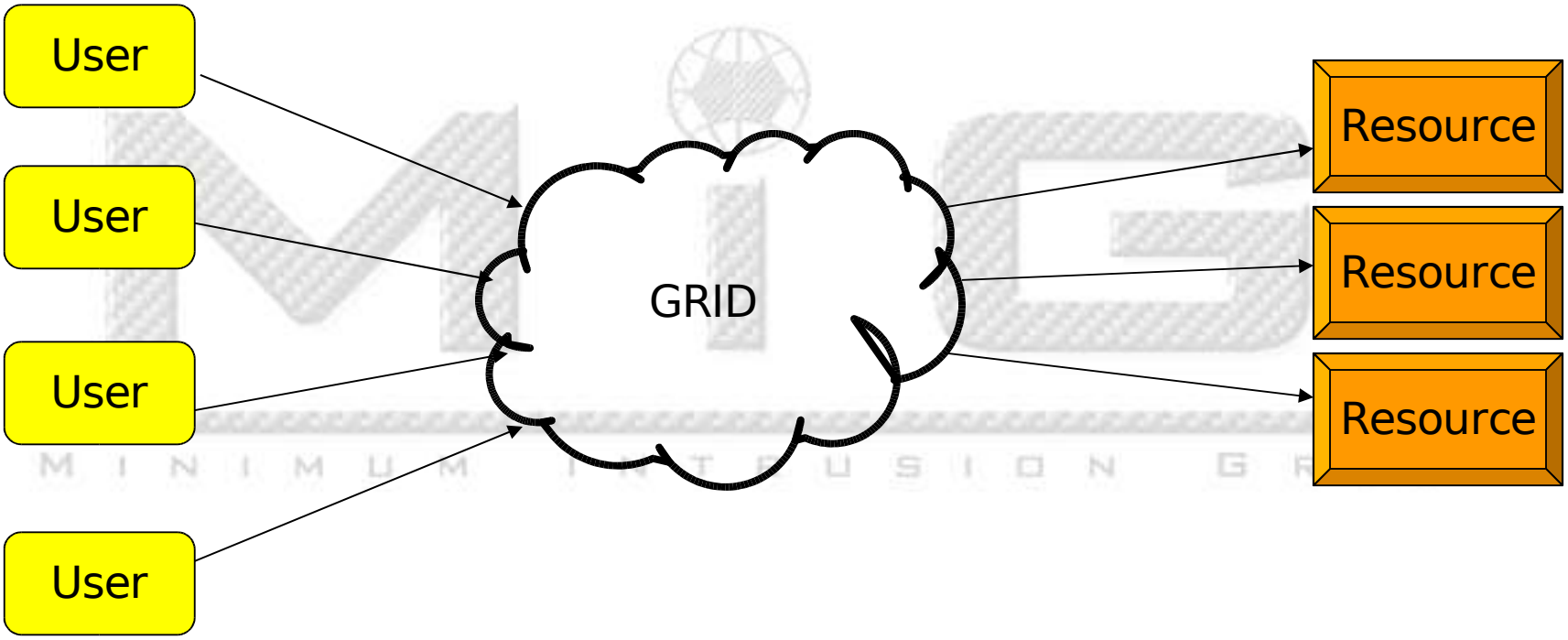
The desired resource view of MiG

- MiG is just another user
  - or set of users
- Economically MiG should be fair

MiG Rules

- **Nothing** produced by MiG can be required to be installed on either the resource or client end
- Everything within MiG must be implemented in Python unless another language is absolutely required
- Any design and implementation decision must optimize towards transparency for the users
- Anything that is not right must be thrown away

The abstract MiG model



MiG requirements - User

- Users should depend on **no** installed SW
  - A web-browser should be enough
- In addition to a browser the users need only a certificate
  - A scriptable user-interface is also available – but not required

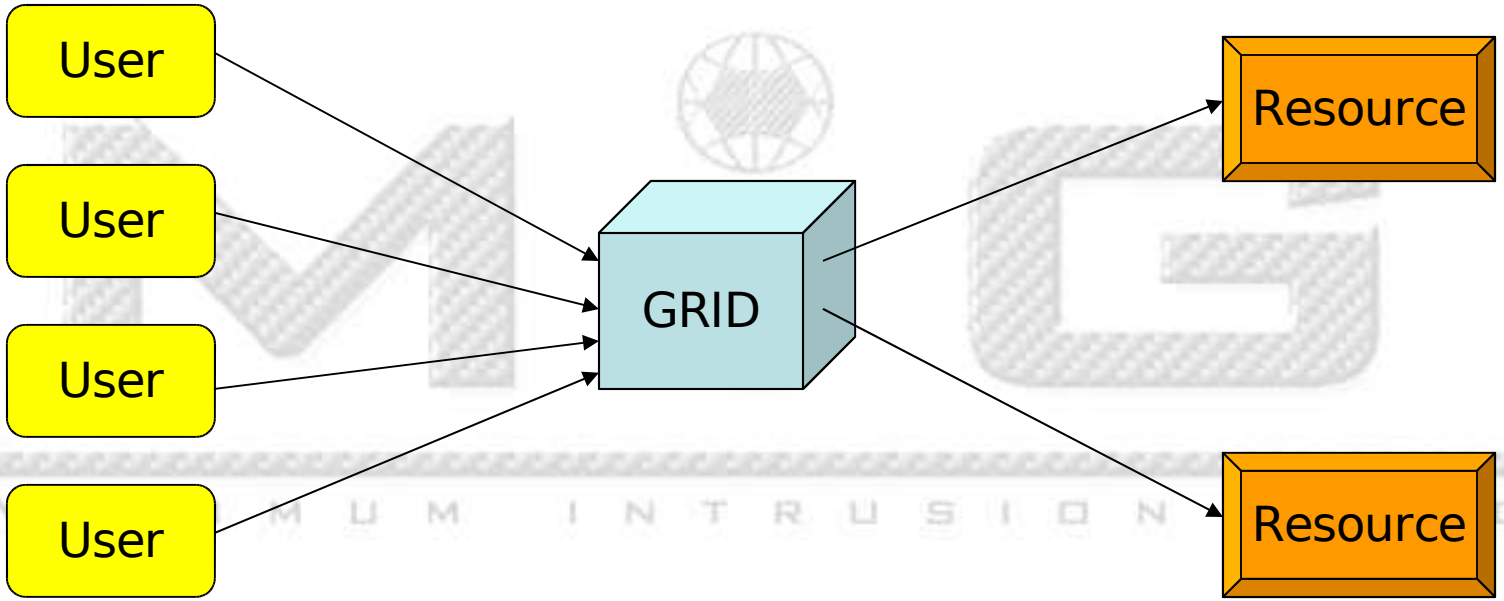
MiG requirements - Resource

- Resources should only create a MiG user
- As few requirements to the resource as possible
  - support ssh inbound
  - support https outbound
- The system administrator also needs a certificate to register the resource

The simple MiG model

- The user sends a job to the GRID machine
- The GRID submits the job to a free resource
  - When one is available
  - Otherwise the job will be stored
- The users can then retrieve the result

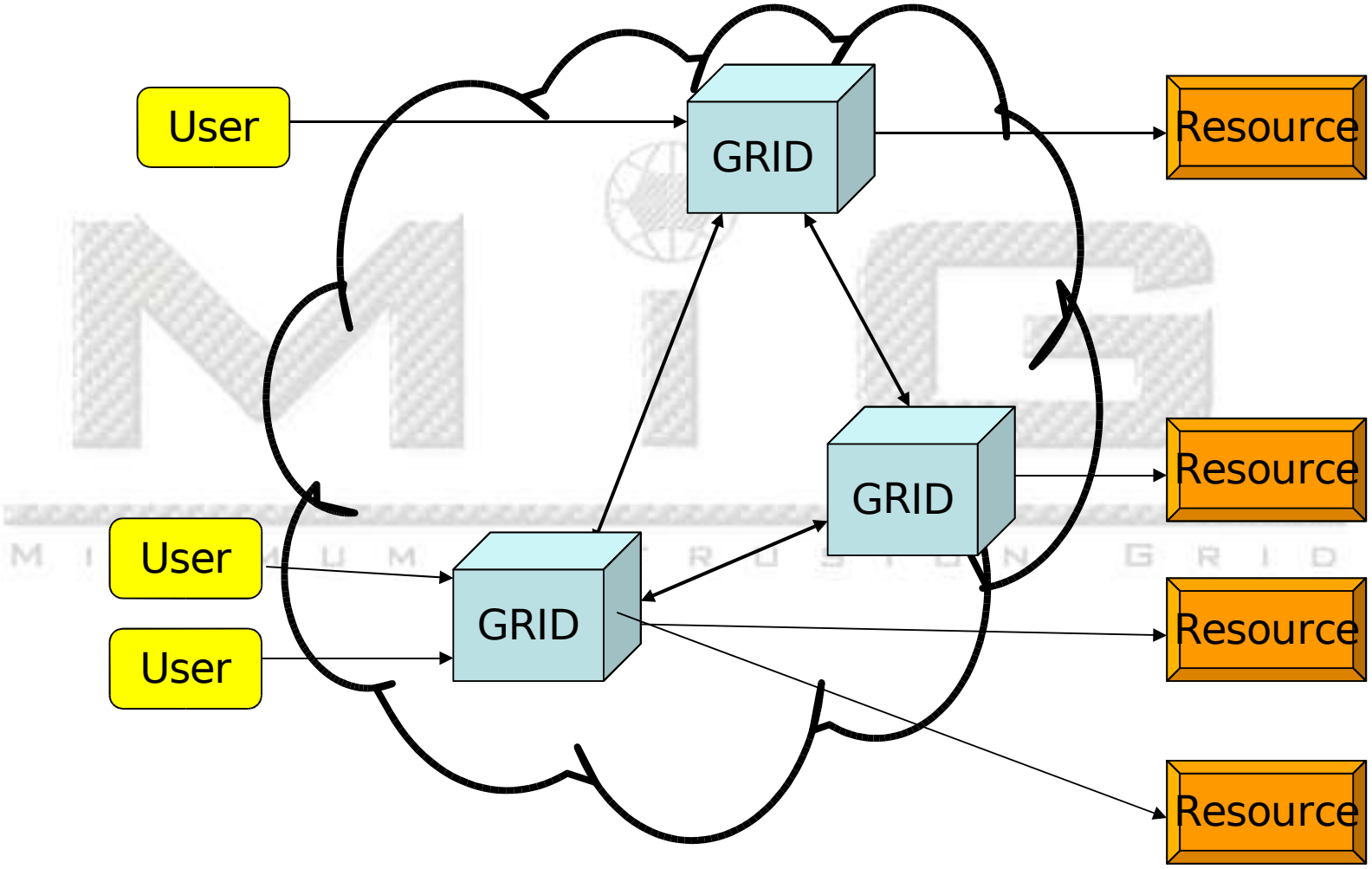
The simple MiG model



Considering the simple model

- Single point of failure
  - and bottleneck
- Solution
  - add more GRID machines

The full MiG model



Considering the full model

- How do we survive the crash of a GRID machine?
  - make redundant copies of jobs and results
- How do we schedule on multiple GRID servers?
  - Envy based load-balancing
- How is the user informed of GRID machines?
  - Users automatically get new hostlist when contacting GRID
  - But he only needs to access one

Future work

- Shared data-structures for MiG
- Fault-tolerance
- Load balancing
- Accounting and Job Pricing
- User defined scheduling
- Grid File System
- Occam like job description language

Future work

- Interfacing with other Grid Implementations
- Programmers API
- Stand-alone http server in Python
- Monitoring
- Statistics
- Application User interfaces
- Graphics rendering on GRID
- Command line interface
- Screensaver model

MiG and DCGC

- MiG is available to DCGC – naturally
- We would like more systems to join as we mature
  - MiG is non intrusive so it will co-exist with any other Grid system as long as ordinary users are supported
- MiG is already being used for our “bio-user-case”
- Anybody within DCGC is very welcome to join the MiG effort

Next

- The simple model – Implementation and status
  - Henrik
- MiG Remote File Access
  - Rasmus
- Economics and Loadbalancing
  - Jonas