ARC Tutorial

Aleksandr Konstantinov
Vilnius University/Lithuania and
University of Oslo/Norway

Contents

- What is Grid?
- What changes?
- Misunderstandings.
- Applications
- Middleware
  - Legion
  - Unicore
  - Globus
- Grid users
  - Authentication
  - Authorization
- NorduGrid
- ARC
  - How it works
  - Job's definition
  - Preparation
  - Simple job
  - Sophisticated job

NORDUGRID
Grid Solution for Wide Area Computing and Data Handling

What is Grid?

- Relatively new term – definition is fuzzy.
  - Many users, many definitions
  - Common misunderstandings
  - Standardization just http://www.globalgridforum.org/.
- Old idea
  - Uniform and safe access to geographically remote and inhomogeneous computing resources.
  - Dynamic pool of users and resources.
  - Distributed management.
  - Resources belonging to different institutions are linked into system.
  - Collaboration and social networking are as important as technology
- Grid environment is usually formed a layer over operating system installed on participating resources and other services.
  - Term used - "**middleware**".
- *I. Foster and C. Kesselmann, The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufman Publishers, 1998*

What changes?

- Physical location of computing and data resources not important.
- Unification is important
  - Usernames and passwords are not used
  - Unification of resources: virtualization or unified interfaces.
    - A lot of unsolved problems.
    - Standardization just started.
    - Maybe virtual computers can help.
    - What should adapt, user's application or Grid environment?

Common misunderstandings

- Grid increases resources
  - One computing resource won't be able to compute more if put on a Grid.
  - But resources can be used more effectively.
- Grid is black box and everything inside it happens automatically.
  - So far in future plans only
  - In reality users still have to learn about peculiarities of implementation. Possibility to monitor job becomes very important.

NORDUGRID
Grid Solution for Wide Area Computing and Data Handling

Applications for Grid

- Should users' applications be adapted to be able to execute in a Grid environment?
  - 1) No changes. Grid is like big cluster and Grid middleware takes care of transferring job and date to execution location.
  - 2) Application is installed on resources in advance or on demand and is contacted through Grid interface. Usually this approach is problematic if application is interactive.
  - 3) Application is modified so it can take advantage of Grid. That is a proper approach if Grid provides own OS-like interface. Especially useful for applications which are able to divide tasks into loosely coupled sub-tasks.

Middleware - Legion

Number of base Grid implementations is not high. Most others are built on top of them.

- **Legion** – Worldwide Virtual Computer - http://legion.virginia.edu
  - An example of middleware which creates own operating system.
  - For user whole system looks like one computer
    - Everything is an object – computing resource, application, device.
    - For new kind of resources new classes are created.
    - It defines interfaces and functionality But not implementations. Only basic objects are implemented.
    - Users can create own classes and even own implementations of existing classes.
  - Commercial.

Middleware - Unicore

- **UNICORE** - “Uniform Interface to Computing Resources” - http://www.unicore.org
  - Hides differences of computing resources behind unified interfaces.
  - Easy to use because it has own graphical user interface for job creation and control.
    - Job is abstract and complex. Job can depend on each other.
    - For every important application new user interface being developed.
    - There is an interface for legacy jobs. But specialized interface are preferred
  - Recently became Open Source.

Middleware - Globus

- **Globus Toolkit™** - “fundamental technologies needed to build computational grids” - http://www.globus.org.
  - Initially set of services and libraries.
  - Now significant part is standards and interface definitions based on “Web Service Resource Framework” (WSRF) http://www.globus.org/wsrf/.
  - Not a single system
    - Services can be used separately.
    - Own services may be created.
  - There are many Grid projects which produce middleware based on Globus Toolkit™.
  - Open Source.

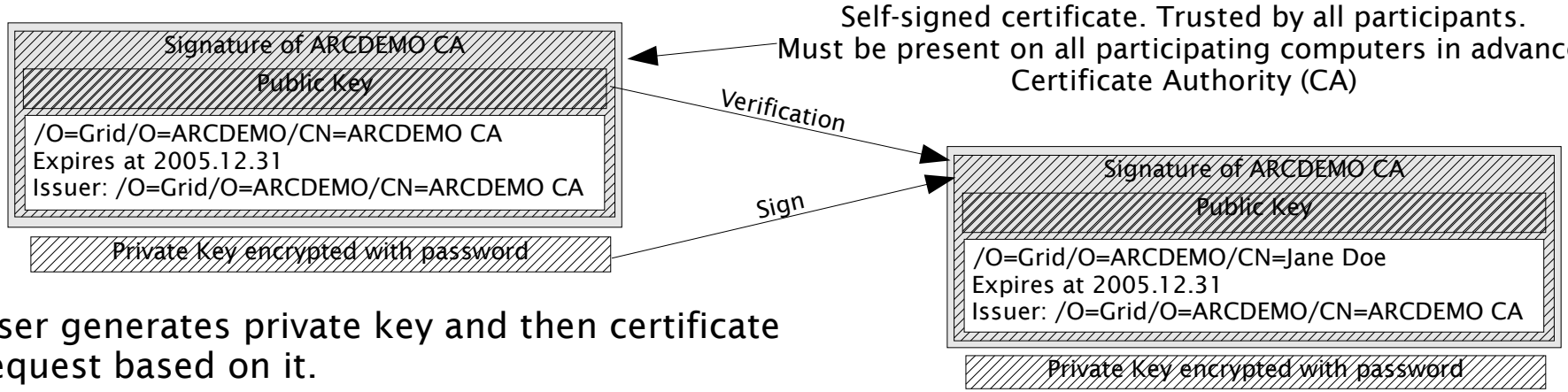
Grid users

If number of users grows high (10^3,10^6,...). And number of resources too (10,100,...). And all handled under different policies. It becomes difficult to manage and synchronize usernames and passwords everywhere and foe everyone. Another method is needed:

- Secure and reliable identification of user - authentication
- Control over users' possibilities - authorization

Grid users - authentication

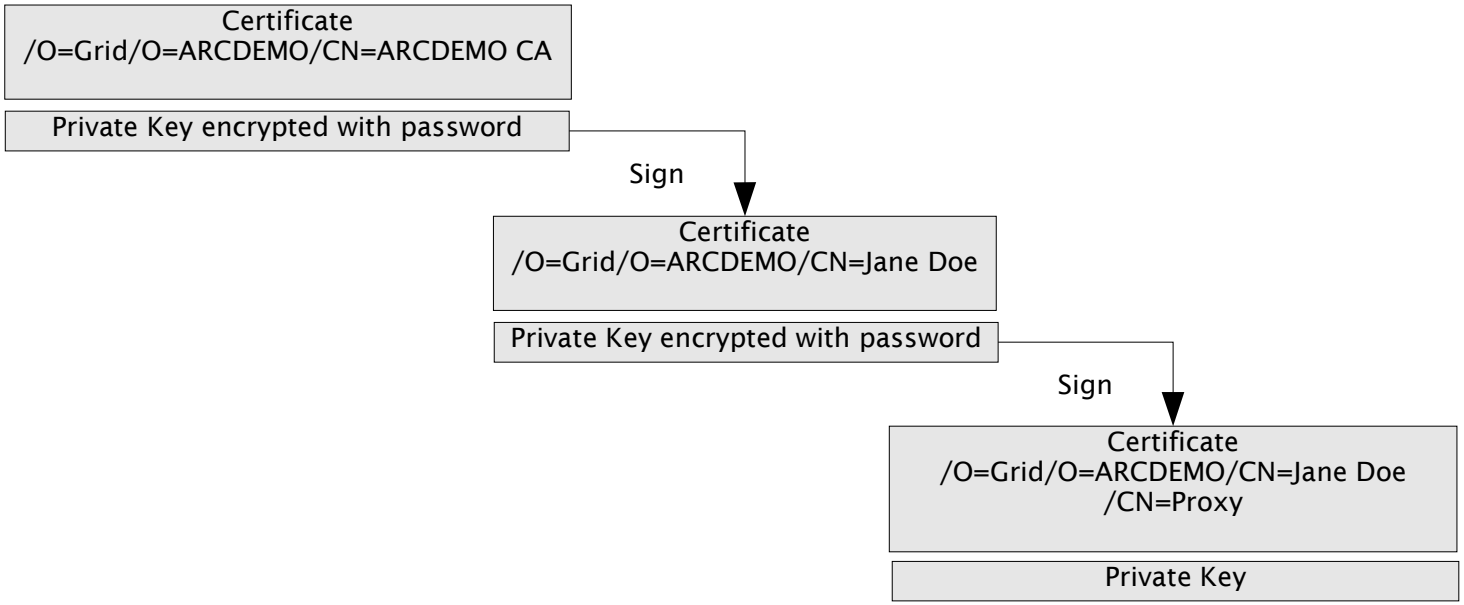
- Many Grid projects use Public Key Infrastructure (PKI) for authentication of all participating objects. Usually X.509.
  - Key – private big enough set of random bits (usually from 1024 to 4096 bits).
  - Certificate – public key bound to private through special algorithm with attributes describing participant attached and digitally signed by private key of higher level participant.
  - Non-symmetric encryption – modification of information through encryption algorithm (cypher). Algorithm uses one the keys. For extracting initial information another key from the pair must be used.



- User generates private key and then certificate request based on it.
- Request is sent to CA.
- CA uses own methods to check authenticity of the user.
- All Grid participants trust CA and it's methods of checking users.

Grid users - authentication

- Proxy
  - Globus uses additional pair of certificate and key created automatically.
  - They are not protected by password.
  - Protected only by means of operating system.
  - Have short validity period – usually 12 hours since creation.
  - Single log in concept – user has to enter password only once.



Grid users - authorization

- It would be quite problematic for owners of resources to decide for every Grid user what he/she is allowed to do on their resources.
- Users are gathered into groups called Virtual Organizations (VO).
- Inside VO roles/capabilities are managed by VO administrators.
- Resources trust VOs and set restrictions per VO.
- Resources can also be part of VO and delegate decisions to VO administrators.

- Globus uses simple text list of users for authorization.
  - There are applications which can manage those files dynamically.
  - But there are more flexible solutions being developed and deployed.

NorduGrid

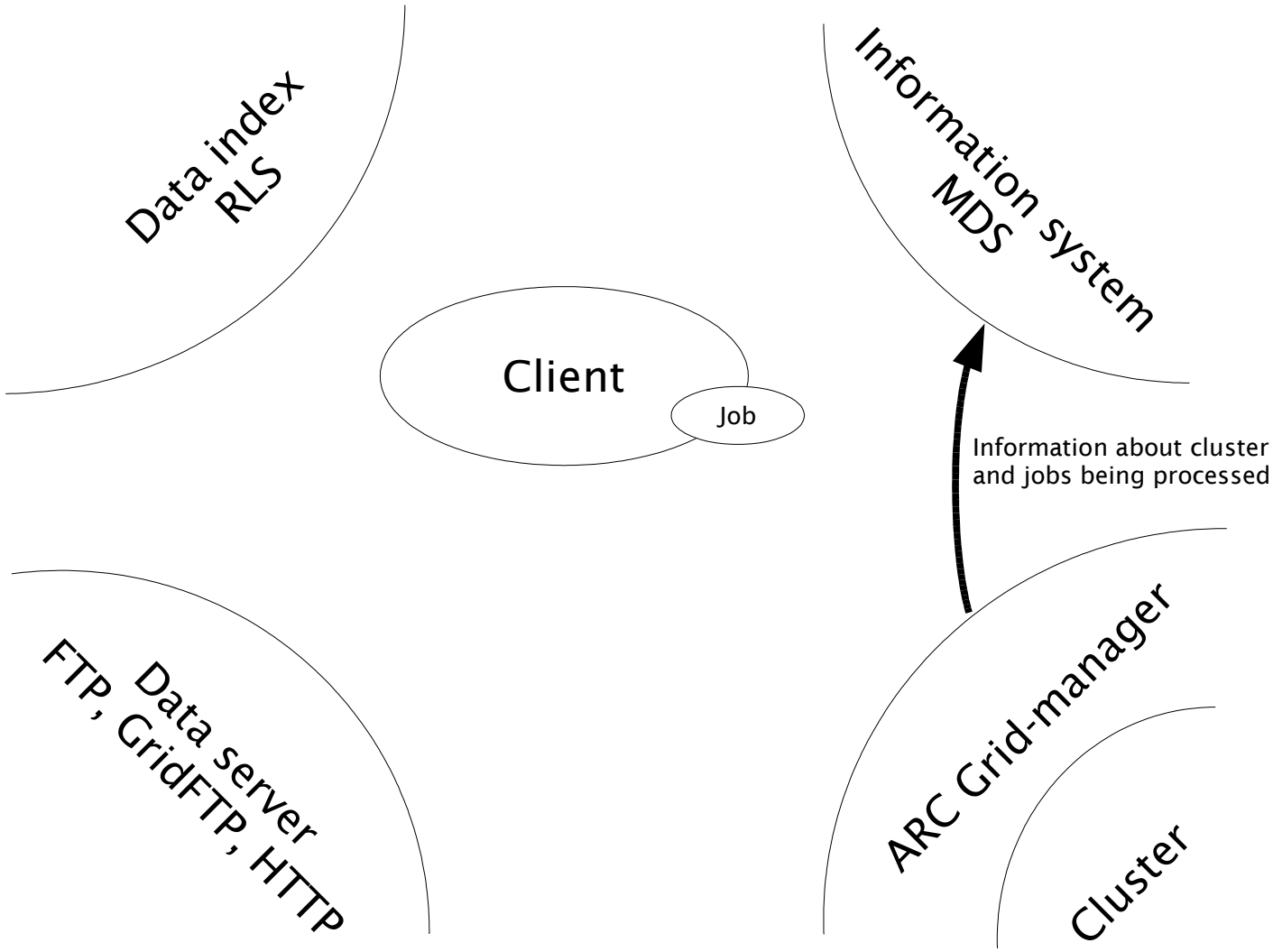
- NorduGrid is a research collaboration established by universities in Denmark, Estonia, Finland, Norway and Sweden
  - Focuses on providing production-capable Grid-like middleware for academic researchers
  - Currently supports one of the largest Grid production systems
    - 10 countries, 40+ sites, ~4000 CPUs, ~30 TB storage

ARC

- ARC (Advanced Resource Connector) is Grid middleware developed by NorduGrid project.
  - It consists of set of services and user interface utilities
  - Based on Globus Toolkit™
- Basic requirements during development
  - Simple
  - Stable
  - Non-intrusive

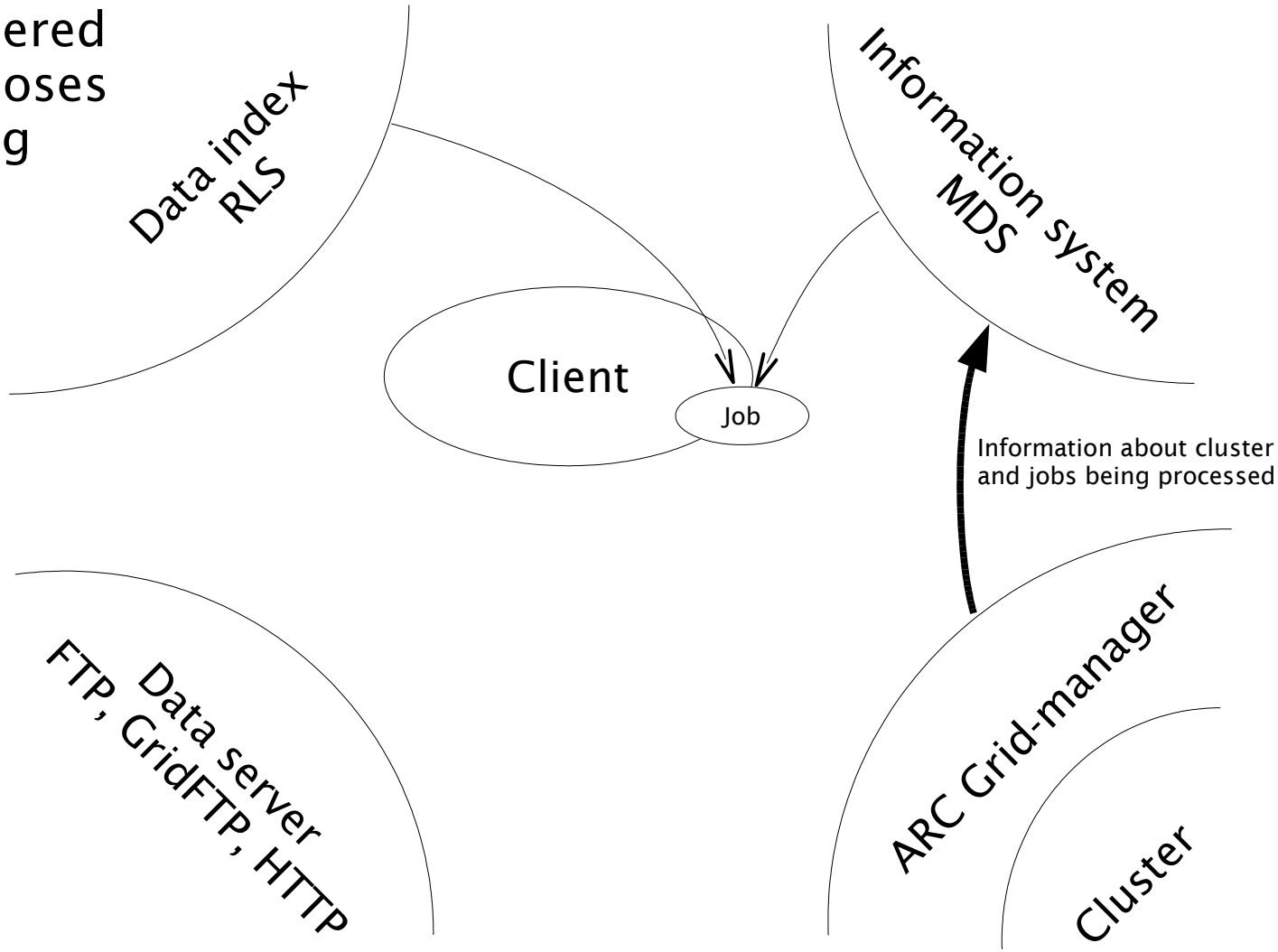
ARC – how it works

- User creates job's description and pass to client application



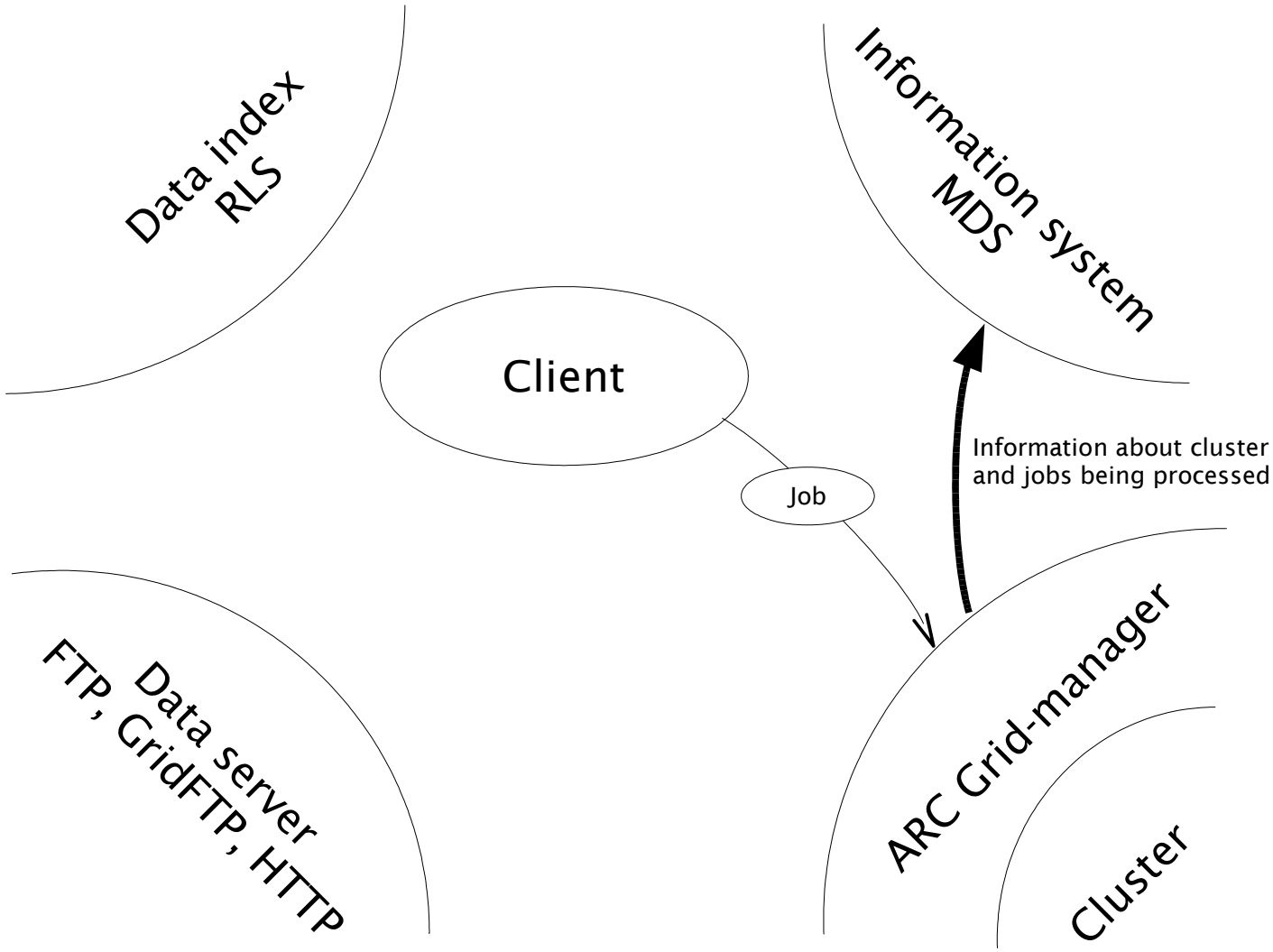
ARC – how it works

- Client gathers information about resources and data storage locations
- According to gathered information it chooses suitable computing resource (cluster)



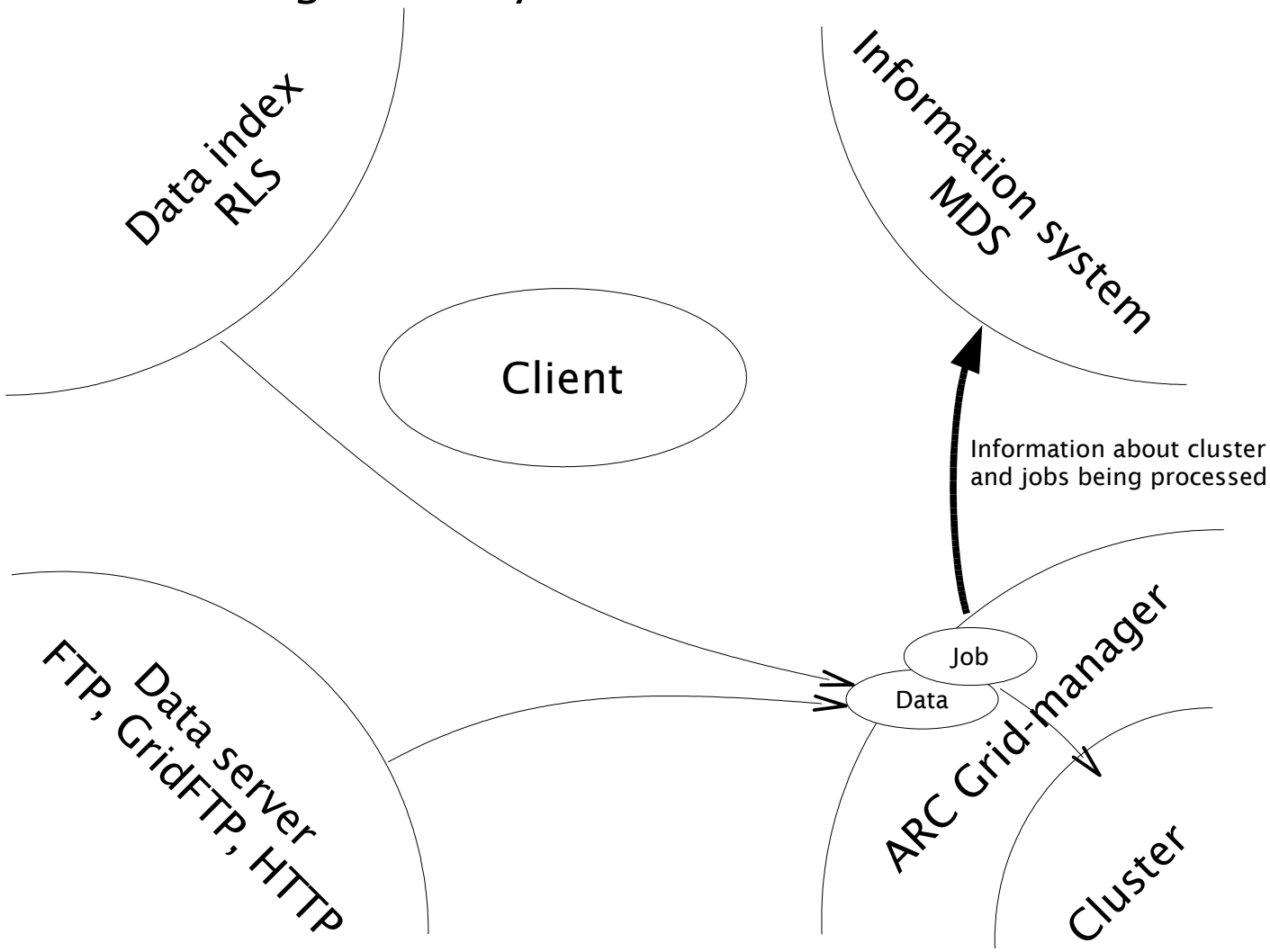
ARC – how it works

- Job is directed to cluster



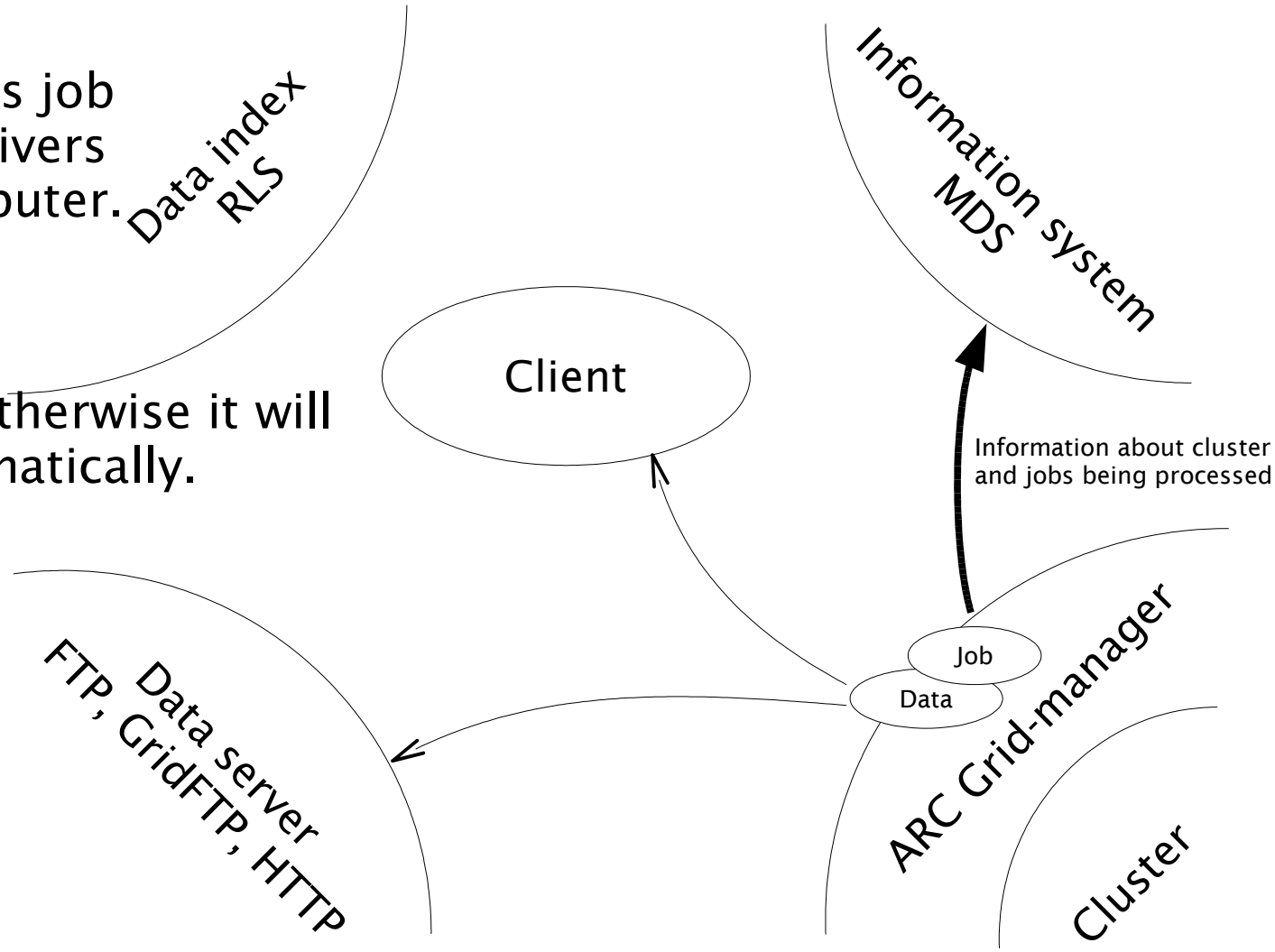
ARC – how it works

- Data is gathered to job
- Job is directed to cluster management system



ARC – how it works

- After execution data generated by job is partially delivered to data servers and partially stored at cluster.
- After client detects job has finished it delivers data to local computer.
- Job is removed. Otherwise it will be removed automatically.



ARC – job description -xRSL

ARC accepts jobs written in xRSL (Resource Specification Language).

&(attribute=value)(attribute=value)...

Mostly used attributes are

- ***executable*** – main executable
- ***arguments*** – executable's arguments
- ***cputime*** – how much time job needs
- ***memory*** – how much memory job can take
- ***jobname*** – job's name
- ***runtimeenvironement*** – application/runtime environment (RE) required by job
- ***stdin***, ***stdout***, ***stderr*** – UNIX standard streams' redirection to files
- ***inputfiles*** – list of input files
- ***outputfiles*** – list of output files

Every file is described by 2 elements

- name of job during job's execution
- source/destination

ARC – environment preparation

This step is already performed

- Installation of User Interface
  - Standalone User Interface is available at http://ftp.nordugrid.org/download/ - *releases - standalone*.
  - Unpack it

```
$ tar -zxvf nordugrid-standalone-0.5.24-1.i386.tgz
```

  - Initialize it

```
$ cd nordugrid-standalone-0.5.24
$ source setup.sh
$ cd ..
```

Working environment is ready. But user must be identified.

ARC – Obtain certificate

This step is already performed

- User needs certificate which identifies his/her authenticity Usually users are supposed to contain a CA of own country.
  - Create certificate request

    $ grid-cert-request -int

  - Answer question:

    password

    organization

    name
  - New directory *.globus* will be created in your home directory. It will contains private key protected by password and certificate request . globus*/usercert_request.pem* .
  - Sent certificate request to your CA with your personal data included.
  - Obtained response store in *globus/usercert.pem* . That can take few days.

ARC – Become a member

This step is already performed

- User must be allowed to use resources
  - Simplest way is to join some VO
  - Contact your VO's administrator
  - NorduGrid Guest VO can be used for testings.
    - Almost everyone is allowed
    - Only very limited amount of resources accept members of that VO.

ARC – at beginning of work

- Start your work from setting up an environment

```
$ cd nordugrid-standalone-0.5.24
$ source setup.sh
$ cd ..
```

- Create proxy certificate and key

```
$ grid-proxy-init
```

- If your jobs require more that 12 hours to finish

```
$ grid-proxy-init -valid 24:00
```

- There few prepared jobs in a directory *examples*

```
$ cd examples
```

ARC – Hello Grid

- Simplest job

```
$ cat hello.rsl
& (executable="/bin/echo")
(arguments="Hello Grid")
(stdout="out.txt")
```

  - No own executable. Using */bin/echo*, available inmost UNIX-like operating systems.
  - No input and output data. Standard output is redirected to a file *out.txt* and automatically appended to the list of output data.
- Submit job

```
$ ngsub -f hello.rsl
Job submitted with jobid
   gsiftp://farm.hep.lu.se:2811/jobs/16294111066557347803 4132
```

  - This can take some time because client must contact all resources and find suitable one.
  - Message from command contains identifier of a job.

ARC – Hello Grid

- Job's state can be monitored using *ngstat* command

```
$ ngstat gsiftp://farm.hep.lu.se:2811/jobs/16294111066557347803 4132
Job gsiftp://farm.hep.lu.se:2811/jobs/16294111066557347803 4132
  Status: FINISHED
```

- Most possible results
  - ACCEPTED – job is accepted and waiting
  - PREPARING – job's input data is being processed
  - INLRMS – job is being executed
  - FINISHING – output data is being processed
  - FINISHED – job is finished
- While job is being run You can look at it's standard output with

```
$ ngcat gsiftp://farm.hep.lu.se:2811/jobs/16294111066557347803 4132
Hello Grid
```

- After job finished all results can be obtained by command

```
$ ngget gsiftp://farm.hep.lu.se:2811/jobs/16294111066557347803 4132
ngget: downloading files to /home/user/examples/16294111066557347803 4132
ngget: download successful - deleting job from gatekeeper.
```

ARC – Hello Grid - Enhancements

- Try to enhance this example
  - Try to use command *ngsub -d 1 -f hello.rsl*. This should show resource ngsub *ngsub* is talking to and how it choses one.
  - Add *(jobname="Test")* to job's description (hello2.rsl) and try to use this name instead of job's identifier.
  - Add *(gmlog="log")* to job's description (hello3.rsl) and look what new appears in job's results.
  - Try commands mentioned before with *-h* argument.

ARC – useful job - GAMESS

Let's try job with some sense.

- GAMESS - The General Atomic and Molecular Electronic Structure System – quantum chemistry application.
- RSL – gamess.rls

```
&(executable="$GAMESS_LOCATION/rungms")
(arguments=gamess.inp)
(stdout="stdout")(stderr="stderr")(gmlog="gmlog")
(inputfiles=("gamess.inp" ""))
(runtimeenvironment="APPL/CHEM/GAMESS")
(cputime="1h")
(jobname="GAMESS")
```

Path to executable is defined thorough variable provided by RE

Input file without source will be transferred to session directory from user's computer

Job requires application/RE GAMESS

Let's request 1 hour of CPU. That is too much but in this way we are on a safe side.

- Input file gamess.inp consists of GAMESS command which form standard test task.
- This a testing job. Hence all we need are standard streams. Make sure generated files do not contain any messages about processing errors.

ARC – Sophisticated job

- Lets try a job with input and output data, own executable. And it also requires preinstalled application
  - Executable file – *Bash script* povray.sh

```
#! /bin/sh
povray +H600 +W800 +omonolith.png monolith.pov
```

Runs main application with defined arguments

  - RSL - povray.rsl

```
&(executable=runpov.sh)
(runtimeenvironment=POVRAY)
(inputfiles=
 ("monolith.pov"
    "http://www.nordugrid.org/tutorial/dapsys_tutorial/povraydemo/monolith.pov")
 ("front.png"
    "http://www.nordugrid.org/tutorial/dapsys_tutorial/povraydemo/front.png"))
(outputfiles=("monolith.png" ""))
(stdout=out.txt)(stderr=err.txt)
(gmlog=logs)
(cputime=1h)
(jobname=povray)
```

Executable to be run. It will be delivered to session directory from user's computer

Job requires runtime environment POVRAY

Input file will be delivered into session directory from defined source URL

Output file will be stored on a cluster in session directory

ARC – Grid Monitor

- Whole system and every job can be monitor using *Web* interface.
  http://www.nordugrid.org/monitor/
- Main window shows a list of all computing resources and their usage.
- Find yourself
  - Use *gridcert-info -subject*, *to find out your name*
  - Choose  to get list of all users
  - Find your name in that list
  - New window will show list of available resources and your jobs

Try it. Almost every element can be clicked.

ARC – Sophisticated job

- After job finished obtain results
  ngget povray
- Result is made of 3 files
- out.txt – output stream
- err.txt – errors' stream
- monolith.png – generated picture

This example was derived from one by Leif Nixon which also shows how to split task into multiple jobs by rendering image in slices.

http://www.nsc.liu.se/~nixon/ng-povray/