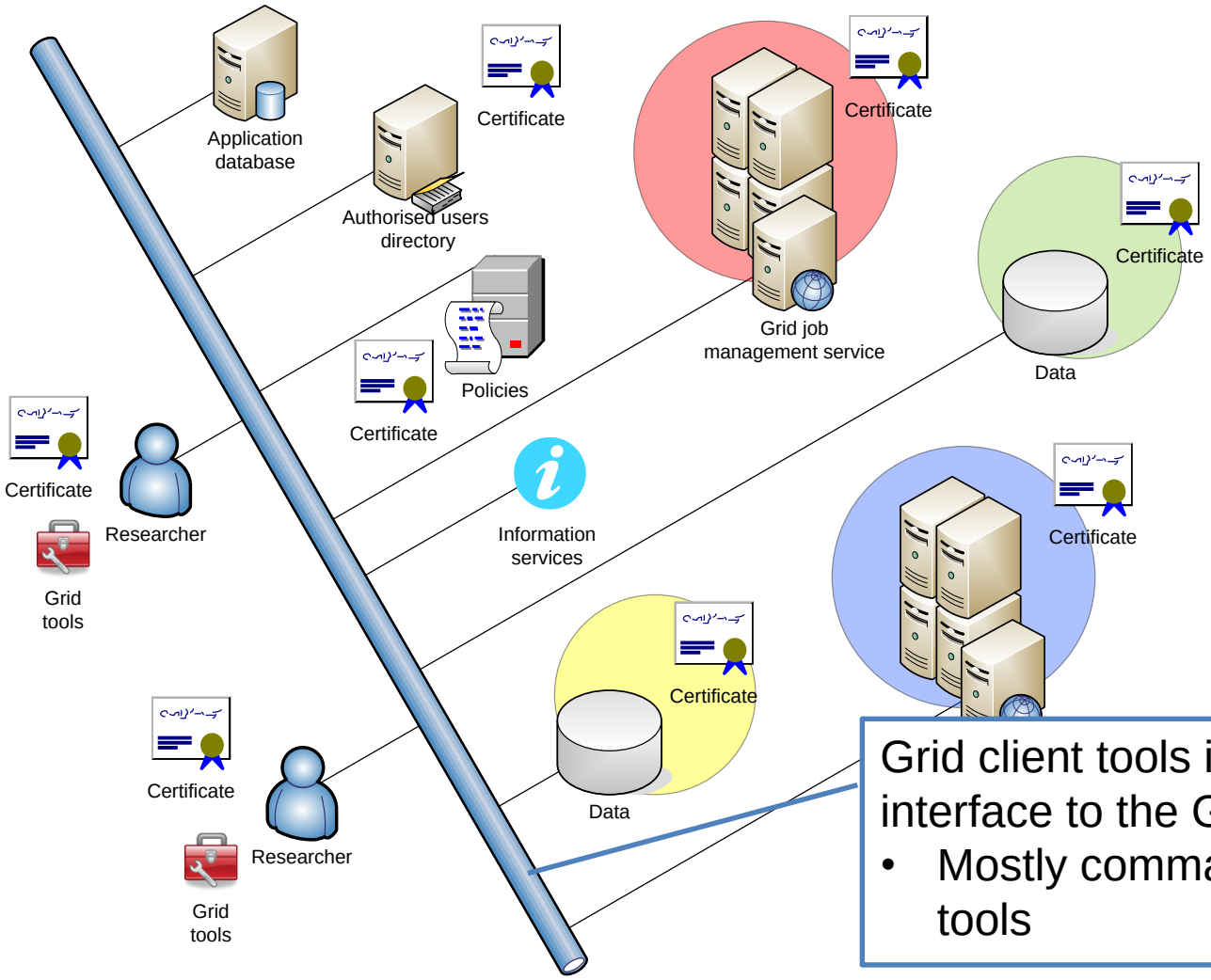
Scheduling, clients

Client tools know everything



Grid client tools is your interface to the Grid

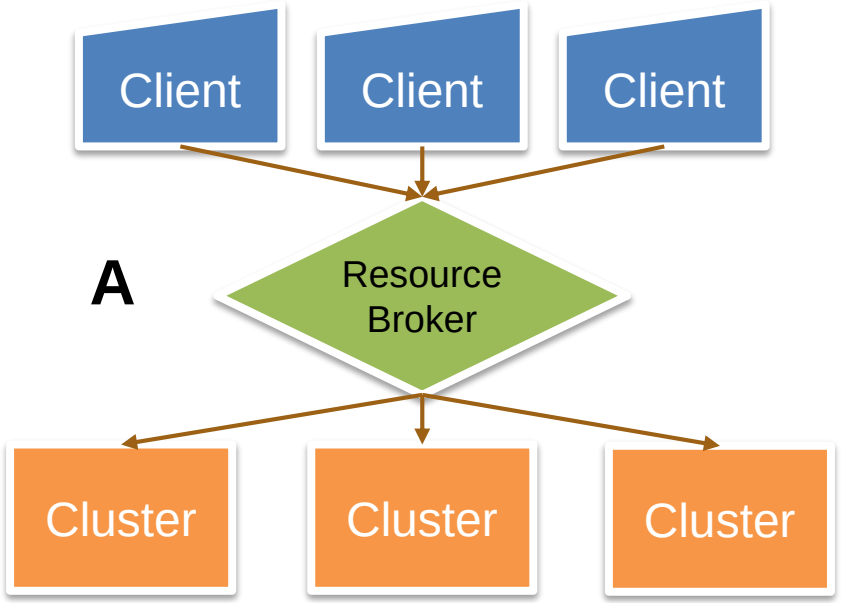
- Mostly command-line tools

LUND UNIVERSITY

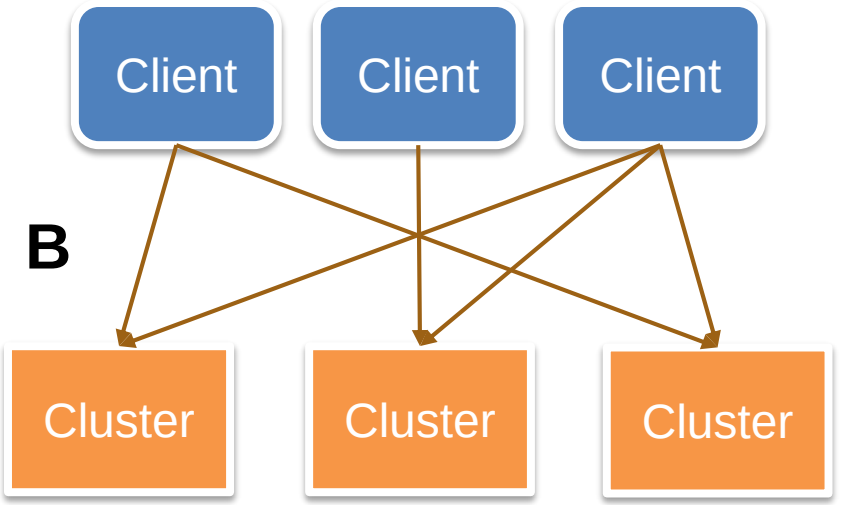
One client tool or many?

- Historically, most Grid tools are Command Line Interface, because:
  - It is similar to batch system tools
  - It is similar to many Unix/Linux tools
  - Unix/Linux users like to write own scripts on top of generic tools
  - There are too many parameters to make a useable graphical tool
- Some graphical tools do exist, focused on certain tasks
  - Like e.g. the storage explorers
- In general, there are many client tools
  - Each focussed on specific tasks
  - Some use Grid libraries, while others are wrapper scripts
  - Some overlap in functionalities
- We will focus on the set of ARC client tools

Grid workload management concepts



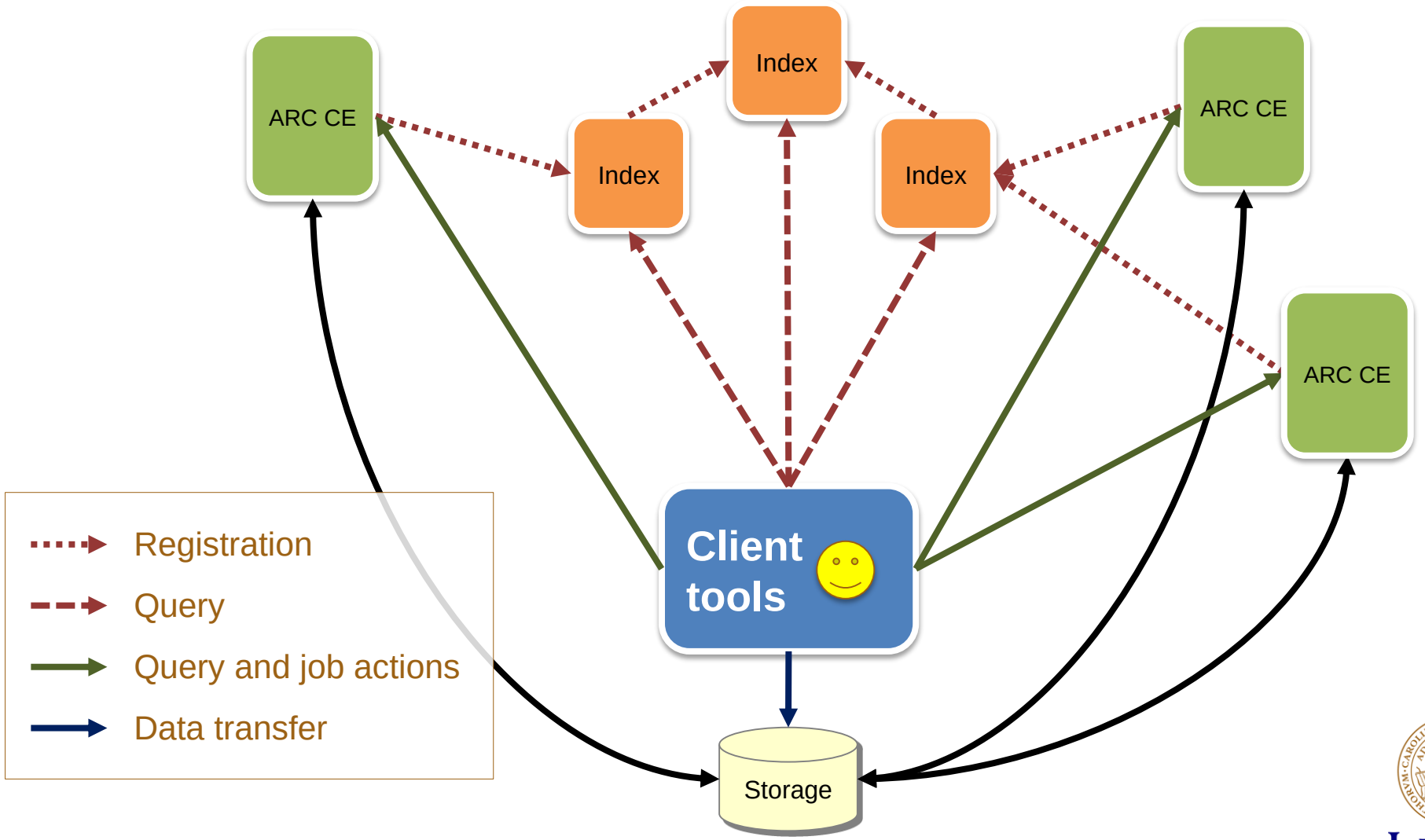
- Original idea **A**:
  - One central service to orchestrate the workload
  - Queue on top of other queues
- Problems:
  - Limited scalability
  - Single point of failure



- Alternative approach **B**:
  - Every client can submit jobs to any cluster
  - No single point of failure
- Problems:
  - Non-optimal workload
  - Rather complex clients
  - Slow interaction with users

LUND UNIVERSITY

ARC architecture overview



LUND UNIVERSITY

Tasks of Grid client tools

Security	• Create proxy certificates
Information	• Discover Grid resources
Computing	• Interpret job description and submit it to a matching resource
Data handling	• Copy files to/from the Grid

What should client tools do for me?

- Handle secure connections (proxies, delegation)
- Matchmaking and brokering
  - Find matching clusters (actually, queues) for my job description, and pick the best from the list of candidates
- Keep the list of my jobs
  - I actually can submit jobs from different computers
  - A client tool should discover them all and update my local list
- Do job and data manipulations on my request
  - Check status, get outputs, kill jobs, move files etc
- Do initial configuration
  - Most importantly: store starting points of the information system
  - Other configuration parameters: location of certificates, timeouts, etc

LUND UNIVERSITY

Recall: handling secure connections

- **`arcproxy`** client tool creates proxies
  - Needs an extra file to keep addresses of Virtual Organisations (e.g. **`~/.voms/vomses`**)
  - Needs your X509 keys (**`.p12`** or **`.pem`** files)
- All other client tools use proxies for secure communications
  - **`arcsub`** triggers creation of a delegated proxy on the cluster
    - » Also uses proxies for authentication
  - **`arccp`**, **`arcstat`** and others use proxies for authentication
    - » Authorisation check is performed by the server-side components
- Other Grid client tools can use your proxy
  - If it is in the default location and has a default name (**`/tmp/x509up_u<UID>`**)
  - Or if you define environment variable **`X509_USER_PROXY`** pointing to your proxy file
    - » ARC client tools can also find non-default proxy location from the configuration file
- All Grid tools need CA keys, e.g. in **`/etc/grid-security/certificates`**

Hold on, I can't remember all these defaults!

- No worry, nobody can!
- There are three ways to define default locations:
  - Use default **file names**, e.g. **`~/.globus/userkey.pem`**
  - Define **environment variables** in your logon scripts, e.g. **`X509_USER_KEY`**
  - Use ARC client **configuration file** (explained later in this lecture)
- All these three ways can be combined
- There can be up to 11 such default locations for e.g. the VO contact points file (**`vomses`**)
  - The client tool (e.g. **`arcproxy`**) will try them all, one by one
- So, check documentation, and pick whichever way is best for you
  - Some locations need superuser privileges

Matchmaking and brokering: `arcsub` tool

User submits a job: **`arcsub myjob.xrsl`**

- Even if option **`-c`** is specified, matchmaking and brokering still proceeds

Client tool (**`arcsub`**) looks up the list of default clusters and information indices in the configuration file

Matchmaking: the tool then polls information system to discover all queues that:

- Match **`myjob.xrsl`**
- Authorise the user

Brokering: from all matching queues, the tool selects one

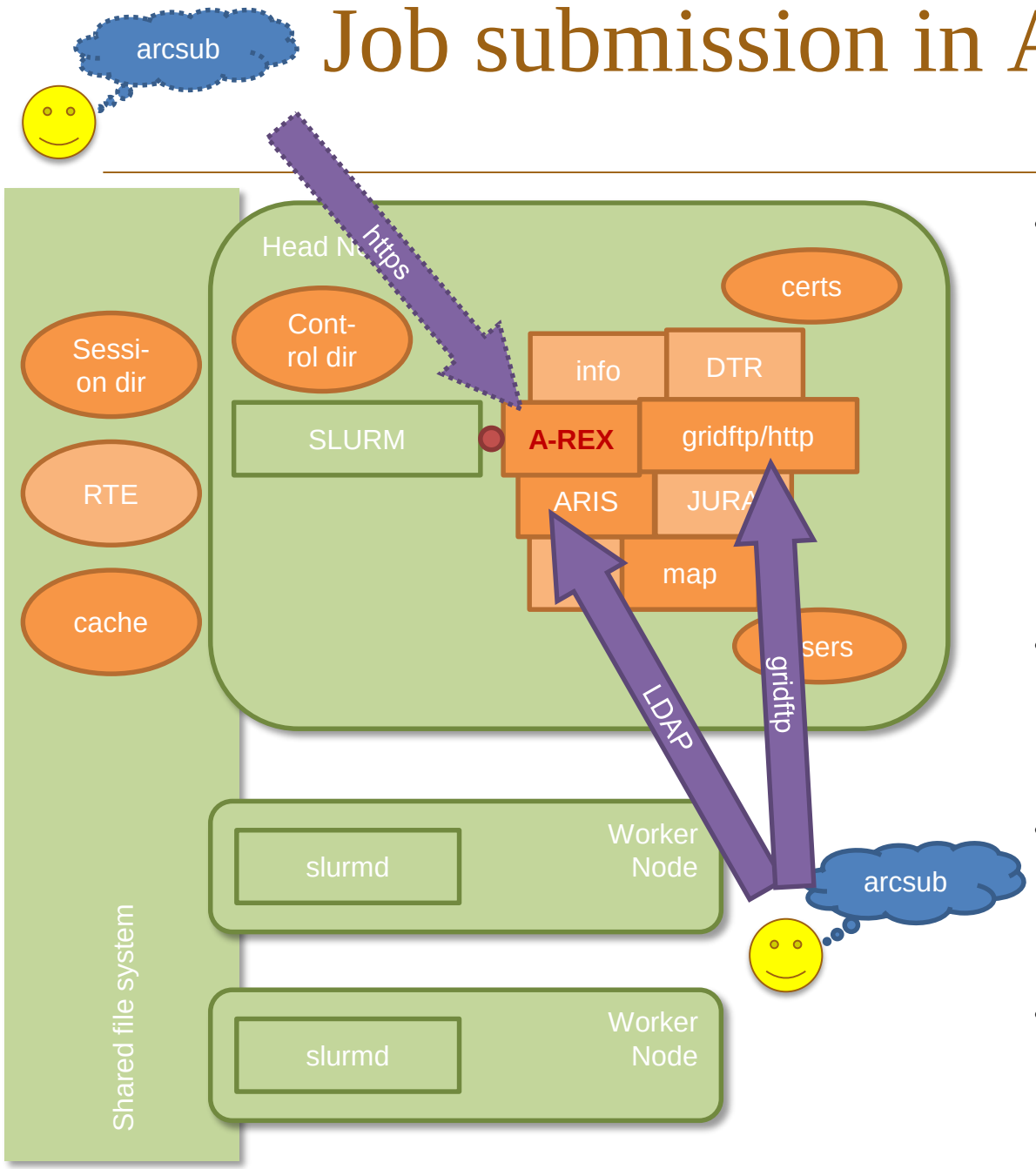
- By default – randomly
- A user can request a specific ranking algorithm

LUND UNIVERSITY

Actual job submission

- Once a target queue is selected, the ARC client tool does the following:
  - Signs the delegated **proxy** request using your own proxy
  - Changes your **XRSL** job description to match the target
    - » Converts expressions like `(memory>10)` to `(memory=20)`
    - » Adds extra attributes, like `(queue="long")`
    - » It can even convert your job description to another language
  - **Uploads** (securely) the following files to the target server:
    - » Job description document
    - » Executable files, if they are on your computer
    - » Other input files, if they are on your computer
  - Adds the submitted job ID to your local **list** (database) of jobs
- Actually, you can simply upload files by hand: it will trigger job submission anyway
  - But the job is likely to fail in a wrong queue and without proxy, leaving no trace

Job submission in ARC: summary



- **Client** tool must:
  - Query information
  - Match it to the job description document
  - Select the best site
  - Convert to a server document (deterministic)
  - Upload all the files
- **A-REX** discovers uploaded job files and launches job processing
  - Advance reservation is not possible
- Currently, information and upload use different protocols
  - https will be used in future for better consistency
- All steps require **authorisation**

LUND UNIVERSITY

Keeping the list of jobs: `arcsync`

- Keeping a list of jobs sent to the Grid is important
  - You don't want to lose track of your jobs
  - Useful reference when you want to operate on several jobs at once
    - » For example, kill all jobs sent to a bad cluster
- If you always use the same computer to send/kill jobs, it is easy to have the list on that computer – **`arcsub`** will do it for you
- If you use different computers, jobs list needs to be synchronised
  - But you can't synchronise with a notebook that is turned off!
  - Luckily, Grid information system knows about the jobs
    - » Provided the relevant clusters are up and running
- ARC keeps jobs list in a database: **`~/.arc/jobs.dat`**
  - In older versions it is an XML file containing job IDs, **`~/.arc/jobs.xml`**
- **`arcsync`** is the client tool that finds your jobs in the Grid information system and updates your **`jobs.dat`**

LUND UNIVERSITY

Checking job progress and getting results

- **`arcstat`** shows job status (taken from the information system)
  - Using A-REX states as described in the 4th lecture
- **`arccat`** prints out standard output/error/log of a job
- **`arcget`** downloads the job results
  - Only those files that you described in XRSL as:
    - » output files
    - » standard output, standard error and Grid log
  - Other files are removed when the job finishes
  - Job results are stored in the folder named after job ID
    - » This can be changed in the configuration
  - Once the results are downloaded, the job is cleaned from the cluster
    - » Option **`-k`** prevents clean-up

Manipulating jobs

- **`arckill`** triggers termination of job execution
  - Cleans up the job, unless option **`-k`** is given
- **`arcclean`** triggers clean-up of a job
  - Useful when you don't need to do **`arcget`**
- **`arcrenew`** triggers a new delegated proxy generation
  - Useful when the old proxy expires
- **`arcresume`** triggers restart of the job from the stage where it failed
  - Usually needed after **`arcrenew`**
- **`arcresub`** resubmits the job a-new
  - Useful when you don't have the original XRSL or input files any more

Working with files

- **`arcls`** prints out remote file or directory listing
- **`arccp`** copies files to/from Grid addresses
- **`arcrm`** removes files and directories on the Grid
  - Caution! Can even remove a job!
    - » Job ID is actually a valid URL!
- **`arcmkdir`** creates a new directory on the Grid
- **`arcrename`** renames files and directories
  - Not all protocols support it

Other ARC client commands

- **`arcinfo`** prints out information about clusters and queues
  - As taken from the information system
- **`arctest`** submits some test jobs
  - Very useful to check that everything works

- Other commands found in the distribution:
  - **`arcslcs`** creates short-living certificates using 3rd party identity providers
    - » Not known to be used in practice
  - **`arcmigrate`** moves jobs from one cluster to another
    - » Works only for a very specific interface
    - » Not known to be used in practice

Common command line options

- `-d` defines verbosity level of terminal output
  - use `-d VERBOSE` or `-d DEBUG` when you can't understand what is wrong
- `-v` prints out version number
- `-h` prints short help
- `-t` changes default timeout
- `-z` changes default configuration file

ARC client configuration

- ARC keeps relevant files in **`~/.arc/`** by default
  - Command line options overwrite defaults, as usual
  - Default configuration file: **`~/.arc/client.conf`**
    - » Command line option **`-z`** allows to specify some alternative configuration
  - Template can be found in **`/etc/arc/client.conf`**
- Most important is to configure the **information system entry points**!

Your **`client.conf`** must contain one or more blocks like this:

```
[registry/index2]
url = ldap://index2.nordugrid.org:2135/Mds-Vo-name=NorduGrid,o=grid
registryinterface = org.nordugrid.ldapegiis
default = yes
```

LUND UNIVERSITY

Configuration file details

- Read ARC Client User Manual for a complete description
  http://www.nordugrid.org/documents/arc-ui.pdf
- The file uses the **`INI`** format:
  - Plain text file
  - Consists of blocks corresponding to different groups of configurable parameters
  - Each block starts with a header in square brackets: **`[blockname]`**
  - Header names indicate hierarchy of blocks, e.g. **`[registry/index1]`** , **`[registry/index2]`**
  - Each block defines a set of parameters as **`attribute = value`** pairs
  - Commented lines start with #
  - Quotation marks are not allowed

Configuration blocks

- `[common]` – used to define many common parameters, such as e.g. timeout, proxy location, verbosity level etc
- `[registry/<alias>]` – used to define information indices
  - These are your entry points to the Grid!
- `[computing/<alias>]` – used to configure preferences for favourite computing clusters

Other configuration files

- `~/.voms/vomses` – can be used by native VOMS clients, too
- `~/.arc/srms.conf` – useful when you use SRM protocol for data transfer
  - Is created and populated automatically, normally needs no user attention
- Server has an own configuration file `arc.conf`, but we as users should never have to bother about it

Other client tools

- Some 3rd party ARC client tools exist:
  - ARC submission plug-in of HTCondor uses basic file upload
  - LUNARC's graphical tools use ARC libraries
    - » Simplify work with many jobs, but don't implement all possible functions
  - Several Web portals exist
    - » Some use ARC libraries
    - » Some call out to ARC command line tools
    - » Most are tailored for specific usage

Exercises

- Create your own **`client.conf`**, using the template from /etc/arc
  - **`cp /etc/arc/client.conf .arc/client.conf`**
- Configure your grid security settings (pathes for certificate files):
  - Create a new block **`[common]`**
  - Use the **`keypath`** and **`certificatepath`** attributes to specify that your certificate files are on the USB key

```
[common]
certificatepath=/media/your_USB_disk/your_dir/your_certfile
keypath=/media/your_USB_disk/your_dir/your_keyfile
```

- Use **`arcproxy`** to generate VOMS proxy, use the **`-d VERBOSE`** option to check which defaults, pathes are actually used
  - **`arcproxy -S nordugrid.org -d VERBOSE`**

Exercises

- Let's try to submit a job to the Grid and let the client find a suitable cluster for your job:
  - **`arcsub hello_grid.xrsl`** …. Most probably it will result in: `"Job submission aborted …"`
- The entry point(s) to the Grid must be specified in **`client.conf`** – add a couple of information index blocks, e.g:

```
[registry/topindex1]
url = ldap://index1.nordugrid.org:2135/Mds-Vo-name=NorduGrid,o=grid
registryinterface = org.nordugrid.ldapegiis
default = yes
```

- Re-run the **`arcsub hello_grid.xrsl`** and check which cluster was selected for your job
  - Use the **`-d DEBUG`** to see the various steps the client is doing "on the Grid"
  - Try to find "bad clusters" and exclude those from the submission attempts using the **`rejectdiscovery=host.name.of.the.cluster`** in the **`[common]`** block of the configuration file
  - Try to modify the **`[registry/...]`** blocks to use only a subset of the entire Grid, e.g., specify only Swedish EGIISes
  - Use the **`timeout`** parameter in the **`[common]`** block to speed up the job submission

LUND UNIVERSITY

Exercises

- Let's find all your jobs on the grid with **`arcsync`** (pretend that you start with a clean system, therefore remove your **`.arc/jobs.dat`** file if it exists):
  - Start with a known cluster: **`arcsync -c arc-iridium.lunarc.lu.se`**
  - Run **`arcstat -a`** to check the newly discovered jobs
  - Extend the search for jobs, synchronization for the entire Grid: **`arcsync -d VERBOSE`**
- After synchronization use **`arcstat`** to list:
  - All your jobs on a cluster: **`arcstat -c arc-iridium.lunarc.lu.se`**
  - All the deleted jobs on the Grid: **`arcstat -s DELETED`**
  - All the successfully completed jobs on the Grid: **`arcstat -s FINISHED`**
  - All the running jobs: **`arcstat -s Running`**
- Fetch the output of one of the FINISHED jobs with **`arcget`**
  - edit **`client.conf`** to change the default download directory (**`jobdownloaddirectory`** in the **`[common]`** block
  - Run **`arcget <jobid>`**
  - What happens if you run **`arcget`** against a not-yet-completed job or against a job you already "downloaded"?

Exercises

- Try other client tools:
  - Use the **`arctest`** utility to easily launch simple test jobs and the other **`arc*`** commands to manage jobs:
    - **`arctest -J 1`**
    - **`arcstat <jobID>`**
    - **`arccat <jobID>`**
    - **`arcclean <jobID>`**
  - Submit a longer job (arctest –J 1 is long enough) and kill it when it starts running:
    - **`arctest –J 1`**
    - **`arcstat <jobID>`**
    - **`arckill <jobID>`**
  - Submit a longer job and renew its proxy:
    - **`arctest –J 1`**
    - **`arcrenew <jobID>`**