Introduction to Programming and Computing for Scientists

Oxana Smirnova

Lund University

Lecture 4: Distributed computing

Most common computing: personal use – PCs, workstations

- Everybody likes to have one or two
- Powerful enough for many scientific tasks

- Strictly personal
- Heavily customized

Customized shared service – clusters, supercomputers

There is always demand and supply

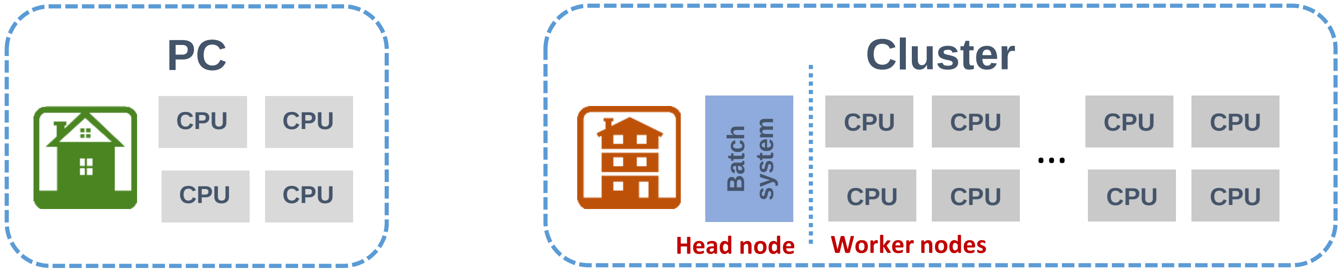
- Systems are customized, but each can serve different users
- Disparate systems can be federated: create computing <u>Grids</u>

Generic service for rent – Clouds

Now exist for computing, data storage, databases etc

- Each Cloud is different, but each can be seemingly infinite because of virtualization
- Users can customize their rent
- No high performance

Big machines for big data: clusters



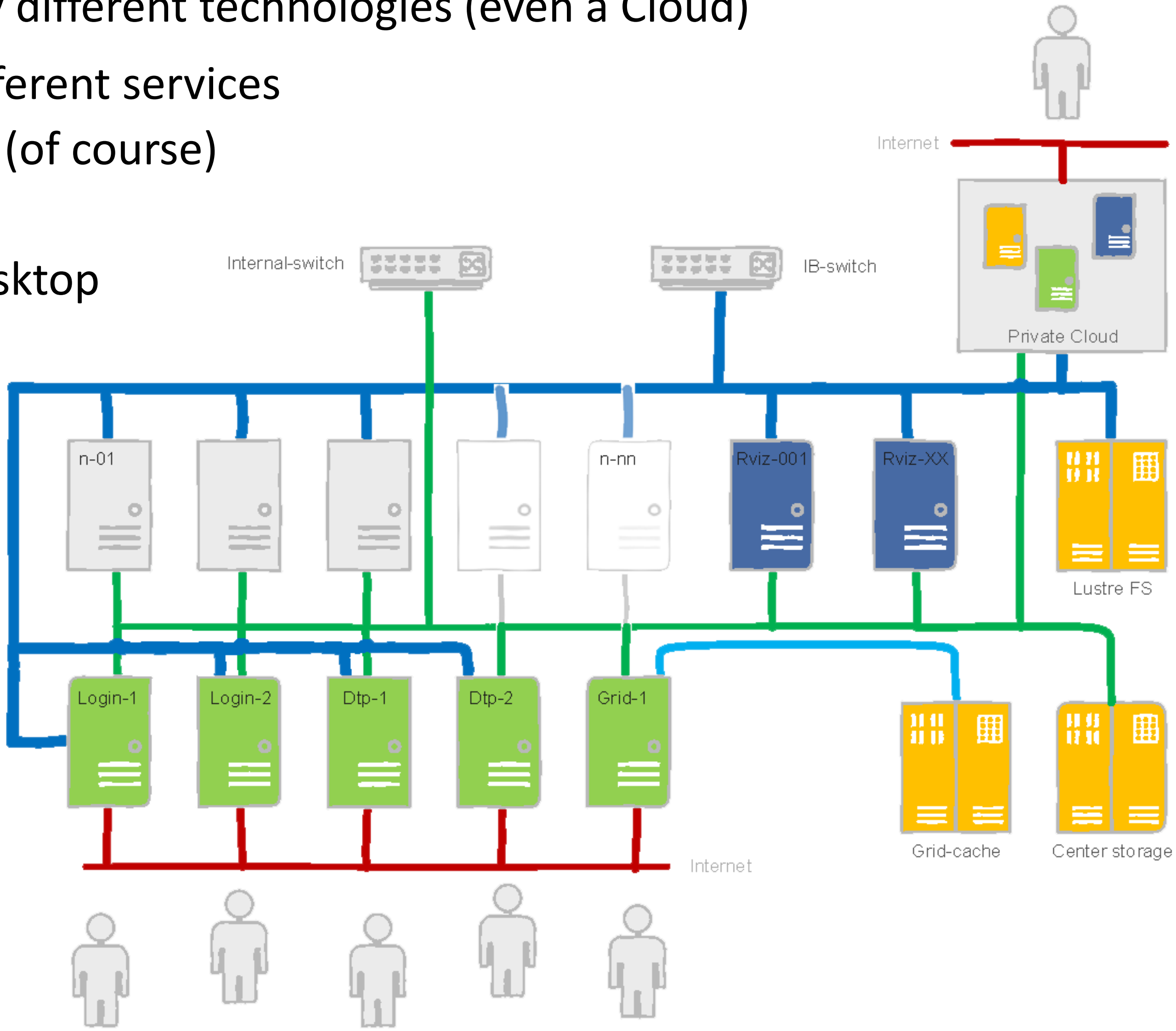
- Computing facilities in universities and research centers usually are Linux **clusters**
  - Some supercomputers are actually clusters, too
- A cluster is a loosely coupled set of computing systems
  - Presented to users as a single resource
  - A typical cluster has a **head node** and many **worker nodes**
    - A *node* is a unit housing processors (CPUs, cores) and memory
  - Distribution of load to worker nodes is orchestrated by **batch systems**
    - Batch system is a software that schedules tasks of different users
    - Many batch systems exist on the market: *PBS*, *SLURM*, *LSF*, *SGE/OGE* etc
- Every cluster is a heavily customised system built for a range of specific applications

Clusters in the LUNARC center

Photo: Gunnar Menander, LUM

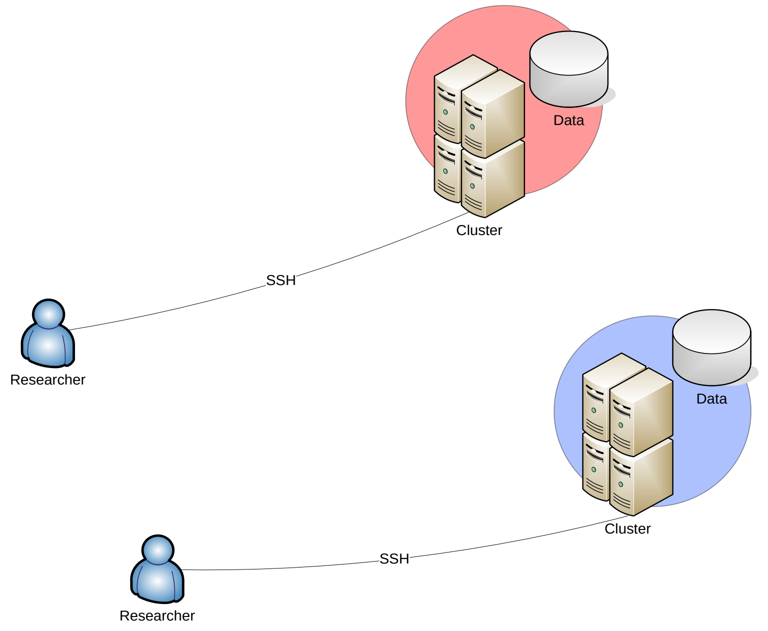
The upcoming cluster at LUNARC

- Combines many different technologies (even a Cloud)
- Offers many different services
    - Computing (of course)
    - Storage
    - Remote desktop
    - etc



Graphics by J. Lindemann

Typical workflow on clusters



- Users connect to the **head node**
  - Typically, using **Secure Shell - SSH**
- Necessary software is installed
  - For example, your own code
    - Either centrally by admins, or privately by yourself
- Specialised scripts are used to launch tasks via **batch systems**
  - A single task is called a **job**
- Data are placed in internal storage
- Scientists often have access to several clusters
  - Different accounts
  - Different passwords
  - Even different operating systems
  - And different sysadmins!

Jobs and queues

- A **batch system** is a software that schedules jobs to worker nodes
  - Called "*batch*" because they are designed to handle batches of similar jobs
- Batch system relies on requirements specified by the users, such as:
  - A job should use a single core (**serial job**), or several cores at once (**parallel job**)
  - Necessary CPU **time** and astronomic (*wall-clock*) time
    - A well-parallelized job will consume less wall time, but CPU time will be similar to a serial one
  - Necessary **memory** and **disk** space
  - Intensive input/output operations (**data** processing)
  - Public **network** connectivity (for example, for database queries)
- When there are more jobs than resources, **queue** management is needed
  - A cluster may have several queues for different kinds of jobs (long, short, parallel, Grid etc)
  - A queue is actually a persistent **partition** of a cluster
    - Queues exist even if there are no jobs – like cashiers in supermarkets

Distributed computing motivations

- More scientific data need more computing and storage
- How to deal with increasing computing power and storage requirements?
  - For parallel jobs: buy larger clusters/supercomputers - $$$
    - Normally, supercomputers are designed for simulation, and not for data processing
      - Disk read/write speed is often lower than processing speed
  - For serial jobs: distribute them across all the community resources
    - For smaller clusters it is easier to match processing and input/output speeds
    - We would like to use the same **access credentials**
    - The results must be collected in one place
    - Progress needs to be monitored
    - Uniform software environment is also needed
  - This community computing is called **Grid**

Some Grid precursors

Distributed file systems: AFS, NFS4

- First implementation in ~1984
- Allow different systems to have common storage and software environment

Condor/HTCondor pools

- High Throughput Computing across different computers
- Started in ~1988 by pooling Windows PCs
- A variant often used as a cluster batch system

Networked batch systems: LSF, SGE

- Can use single batch system on many clusters since ~1994
- Variants of regular cluster batch systems

Volunteer computing: SETI@HOME, BOINC

- Target PC owners since ~1999
- Supports only a pre-defined set of applications

Grid concept – formulated in ~1999

- Grid is a software technology that:
  - creates <u>federations</u> of different computing systems
  - provides <u>single sign-on</u> and <u>delegation</u> of user's access rights
  - relies on <u>fast networks</u>

Abstracts interfaces from systems

- No need for common batch systems or common file systems

Introduces security infrastructure

- Single sign-on
- Digital certificate based authentication and authorisation

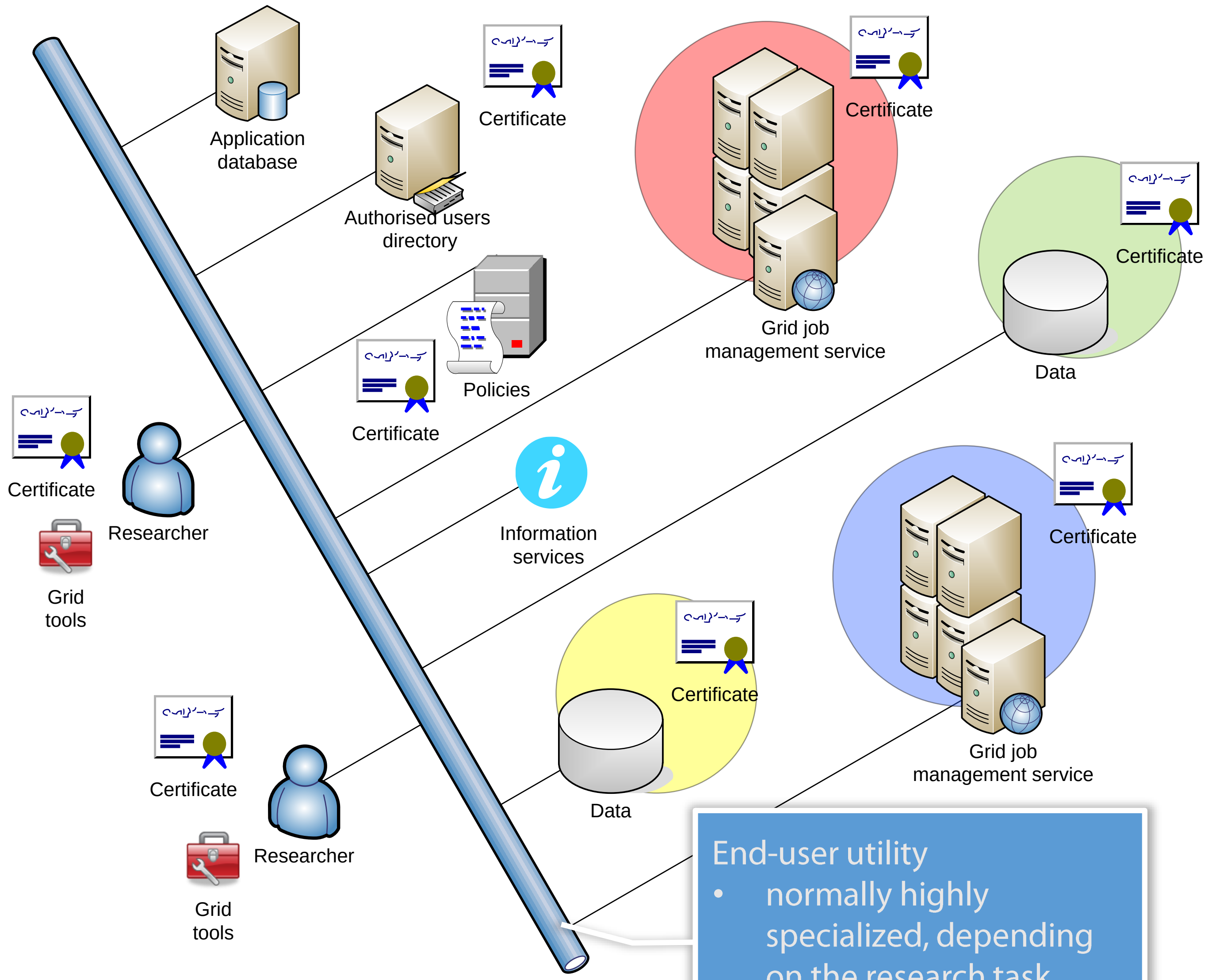
Introduces resource information system

- Necessary for job management across different systems

Ideal for distributed serial jobs

- Initially was thought to be suitable even for parallel jobs

Overview of generic Grid components



End-user utility

- normally highly specialized, depending on the research task

Some Grid software providers

The first: Globus Toolkit (stems from the USA)
http://toolkit.globus.org/toolkit

- Provides computing capacity, basic storage capacity, and corresponding client tools
- Comes with extensive libraries and API, used by other providers
  - Especially for the Grid Security Infrastructure

Used for CERN-related computing and in this course: ARC by NorduGrid
http://www.nordugrid.org/arc

- Provides computing capacity and client tools for job and file operations
- Developed in Lund and many other places

Used in Europe for CERN-related computing: EMI
http://www.eu-emi.eu

- Includes many different components and services for storage, accounting, information, security etc
- Getting old and unmaintained, unfortunately

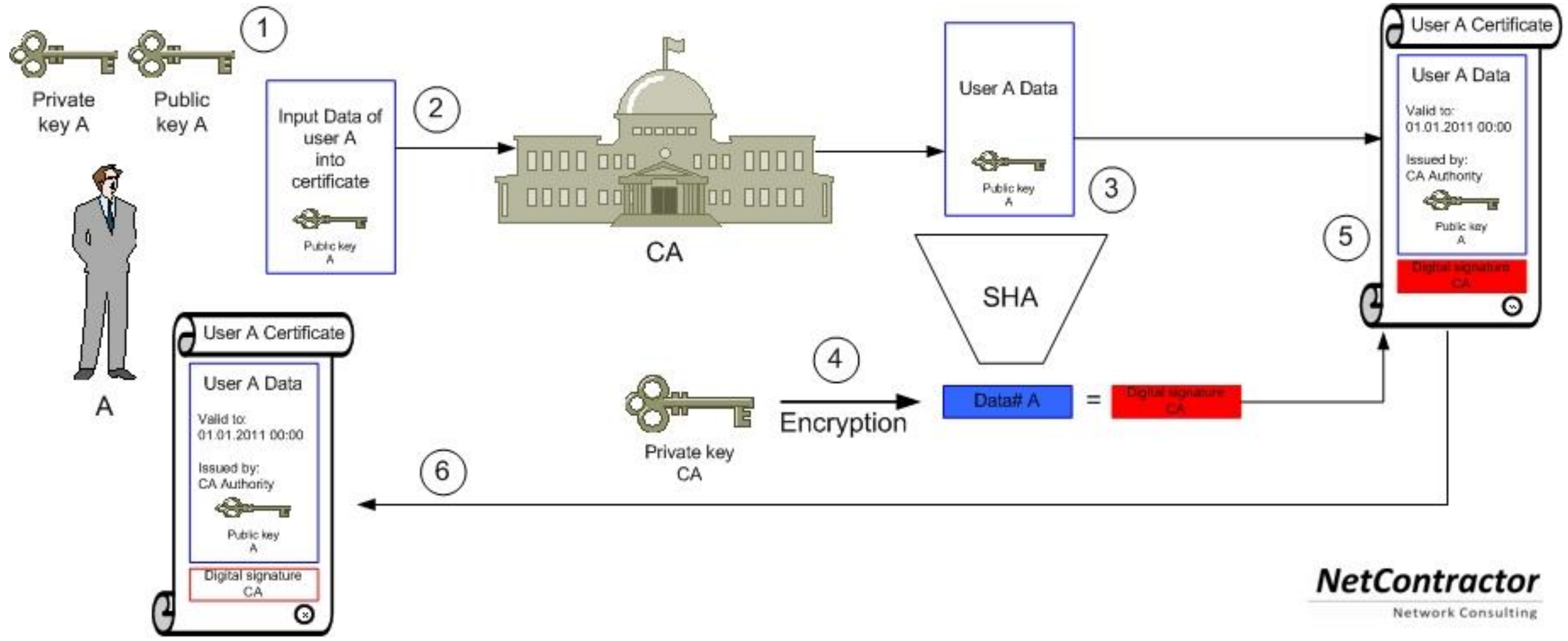
To access community resources, you need a permission

- To access one computer (or one cluster) you need a password
  - You also have a personal user space (account)
- Now scale it up 100+ clusters and 1000+ users
  - You can't quite remember 100+ passwords
  - Sysadmins can't quite manage 1000+ user accounts
- Solution: use **Public-Key Infrastructure** (PKI)
  - Each user has a digital certificate
  - Each service also has a certificate
    - Service is anything you can connect to: e-mail service, Web service, database service, bank service etc
    - Sometimes you need services to act on your behalf: delegate your rights to them

Principles of PKI

- Goals:
  - reliably verify identity of users and authenticity of services by means of digital signatures
  - communicate securely over public networks
- There are trusted **Certificate Authorities** (CA) that can vouch for:
  - identities of users
  - trustworthiness of services
- Each actor (user, service, CA) has a public-private **pair of keys**
  - Private keys are kept secret, **off-line**; public keys are **shared**
  - Keys are used for both authentication and communication encryption/decryption
    - For our purposes, authentication is most important
- CAs digitally validate ("sign") **public certificates** of eligible users and services
  - Public certificate contains owner information and their public key
  - Each CA has a set of policies to define who is eligible

Obtaining a personal certificate



Beware: words “certificate” and “key” are often used interchangeably!

Private key

- Private key is a cryptographic key – essentially, a sufficiently long random number
  - Longer it is, more difficult it is to crack; 2048 bit is good (as of today)
- Purposes:
  - **Create** digital signature
    - to sign letters, contracts etc
  - **Decrypt** encoded information
    - when encrypted by someone using your *public* key
- There are many softwares that create private keys
  - Even your browser can do it
  - Keys come in many different formats
- **Important**: private key must **never** travel over public unprotected network
  - **Don't store them in Dropbox!**
    Don't send them by e-mail!

Public key

- Mathematically linked to the private key
  - It *should* be impossible to derive private key from the public one
    - Different public-key algorithms exist
    - Benefit: no need to securely exchange private keys, as public keys are enough and can travel unprotected
- Purposes:
  - **Verify** digital signature
    - use sender's public key
  - **Encrypt** plain information
    - use your addressee's public key
- Usually, software tools create both public and private key in one go
  - They can even be stored in one file
    - **this file must not travel then!**

Protocols using public key cryptography

- Some examples:
    - SSH
    - SSL and TLS (used e.g. in https, Gmail)
    - GridFTP: a variant of FTP tailored for Grid
    - PGP and GPG (used e.g. to sign software packages or sign/encrypt e-mail)
    - Bitcoin
    - ZRTP (used by secure VoIP)

Grid flavour of PKI

- Historically, Grid makes use of the **X.509** PKI standard (so do Nordea, Skatteverket and many others)
  - Defines public certificate format
    - Certificate must include subject's **Distinguished Name** (DN):

      ```
      /C=UK/O=Grid/OU=CenterA/L=LabX/CN=John Doe
      ```

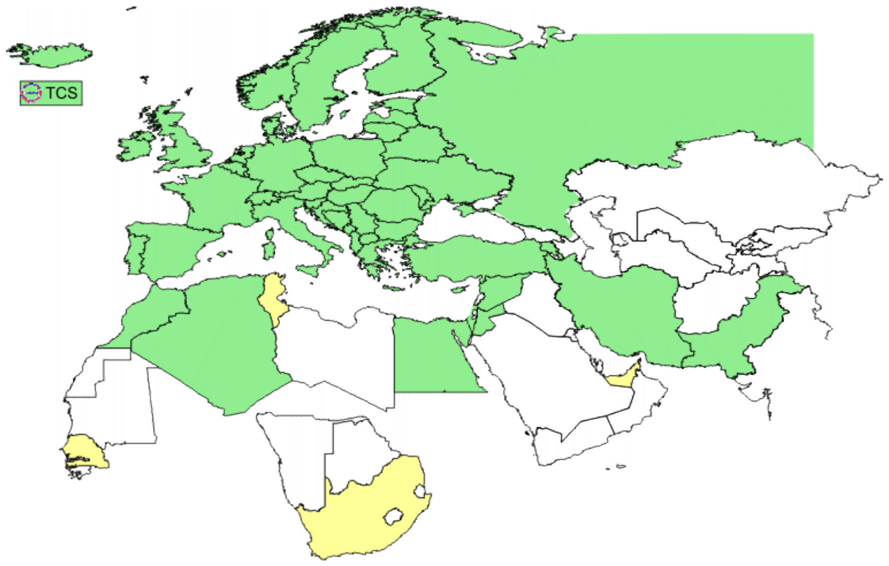
    - Certificate has **limited** validity period
      - Usually, one year or so
  - Assumes strict hierarchy of trusted CAs
    - Unlike PGP, where anyone can vouch for anyone
    - Check your browser for pre-defined list of root CAs
  - Requires certificate revocation status checks
  - Public certificate is **password-protected**
    - You can not reset the password; if forgotten, a new certificate must be requested
- One can convert X.509 certificates into SSH ones

Certificate Authorities

- Web browsers and even operating systems come with a set of trusted root CA certificates
  - You can always add own trusted CAs, or remove untrusted ones
    - When you remove a CA, you won't be able to securely connect to a server certified by that CA
- Grid has an own set of trusted CAs: the **International Grid Trust Federation** (IGTF), http://www.igtf.net/
- In order to use Grid, you **must** keep the IGTF CA certificates up-to-date!
  - Several releases per year
  - Each CA is represented by a separate package
    - You can always uninstall a CA package you don't like or don't trust
  - Packages are available from IGTF and two Grid projects: EGI, NorduGrid
    - RPM, deb, tar

IGTF

- European part of IGTF: EUGridPMA
  - https://www.eugridpma.org/
- Each country used to have an own CA
  - CERN also has a CA
  - Nordic countries have one CA
- Nowadays, there is a single European CA: TERENA
  - Lund students and employees should use TERENA certificates
  - Relies on national network operators to confirm identities
    - National operators rely on universities and such



You still need all the IGTF CA certificates!

Certificate revocation lists (CRL)

- Certificates of people and services can be revoked
  - If they are compromised, or if some information in the certificate is changed
    - If your affiliation changes, you must get a new certificate, and the old one must be revoked
- For security reason, before connecting to a service, software must check whether its certificate is revoked or no
- Certificate revocation lists (CRLs) are published by CAs
  - They are regularly updated
  - You must regularly refresh your local copy of CRLs
    - A cron-based tool exist
- Other technologies exist – e.g. Online Certificate Status Protocol (OCSP) – but in the Grid world CRLs rule

Mutual authentication

- **Authentication** is establishing validity of person's (or service) identity
  - Not to be confused with authorisation: established identity may still lead to denied access
- Users and services on the Grid must mutually authenticate:
  - Both parties must have valid certificates
  - Both parties must trust the CAs that signed each other's certificates
    - "Trusting a CA" means having the CA's public certificate stored in a dedicated folder/store
    - Removing a CA certificate breaks trust
    - Removing your own signing CA certificate breaks everything
- Technically, authentication process involves exchange of encrypted messages, which parties can decrypt only if they are who they claim to be

Delegation: Why act on behalf of users?

A “normal” cluster usually has local storage

Same user identity is used for jobs and data access

On the Grid, storage is all over the World

Every time a job needs to read or write data, authorised remote connection is required

A Grid cluster’s own certificate is not enough

Users want to protect their data from unauthorised access

Users also don’t want everybody to write to their storage share

So each cluster needs a document from a user, delegating access rights

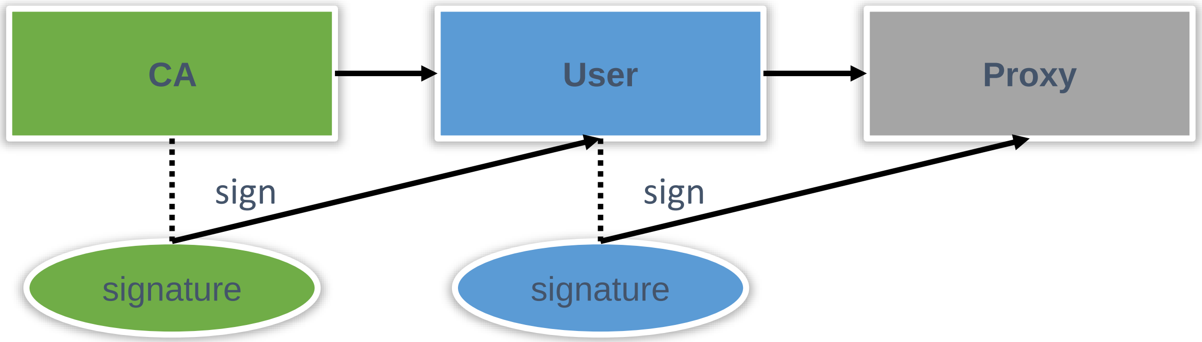
Delegation: Act by proxy

- In real life, you sign a **proxy** document and certify it by a notary
  - Document says what actions can be performed on your behalf
- On the Grid, a proxy document is a X.509 certificate signed by you
  - Since your certificate is in turn signed by a CA, proxy is also a trusted document
  - Proxy may contain a lot of additional information

PROXY AND WAIVER

IRVING NATIONAL BANK

Know all Men by These Presents, [illegible]



Proxy certificate

- Proxy is an extension of the SSL standard
- Proxy contains both public and private keys
  - Not the same as users' keys, but derived from them
- Proxy needs no password (unlike usual PKI certificates)
- Proxy can not be revoked
- Proxies are used by Grid services, to act on behalf of the proxy issuer

There is no need to transfer proxy: it is created by the service

Proxies must have **very short lifetime:**

- Reduces the chance of getting stolen
- Minimizes the damage

Authorisation

- Authentication = passport; authorisation = visa
  - Having a valid passport is not enough to enter a country
  - Having a valid proxy is not enough to access computing or storage resources

- Authorisation can be by person or by group
  - By person: a person with Swedish visa can enter Sweden
  - By group: everybody with a EU/EEA/US passport can enter Sweden
- Authorisation on the Grid:
  - By person: your DN is in the list on a cluster (matched to your proxy DN)
  - By group: your DN is in the **Virtual Organisation** *(VO)* list
    - Your proxy has this VO's *Attribute Certificate*

Virtual Organisation

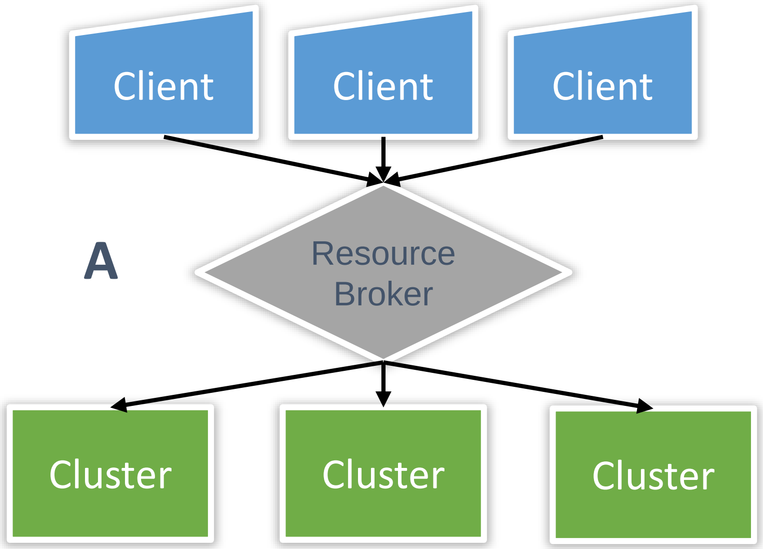
- A Grid **Virtual Organisation** (VO) is simply a group of people
- VO attributes:
  - VO must have a manager who approves membership
  - VO must have a set of rules – policies – regulating the membership
  - VO must have means of providing an up-to-date list of members' DNs to Grid services
  - VO may have groups and roles
    - Useful to define shares and privileges
  - VO may run a service that issues Attribute Certificates (AC)
    - An AC asserts VO membership of a user, as well as their role, group, or other attributes
    - An AC is digitally signed by the issuing VO
    - An AC is included into the proxy

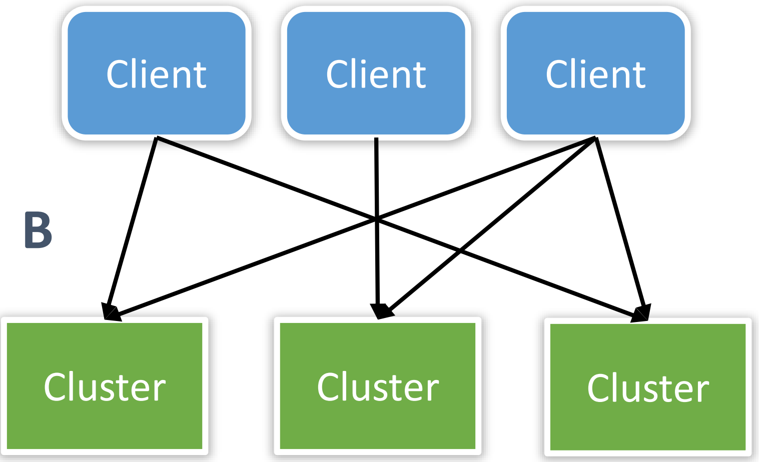
The core of the Grid: Computing Service

- Once you got the certificate and joined a VO, you can use Grid services
- Grid is primarily a distributed **computing** technology
  - It is particularly useful when **data** is distributed
- The main goal of Grid is to provide common layer on top of different computing resources
  - Common authorization, **single sign-on** – by means of proxies
  - Common task specification (**job description**)
  - Common protocols and interfaces for job management
  - Common accounting and monitoring
- All this is provided by Grid **Computing Services**
  - A single instance of such service is called a **Computing Element** (CE)
  - You also need a Grid **client** software to communicate to Grid services (CE, storage etc)

Grid workload management concepts

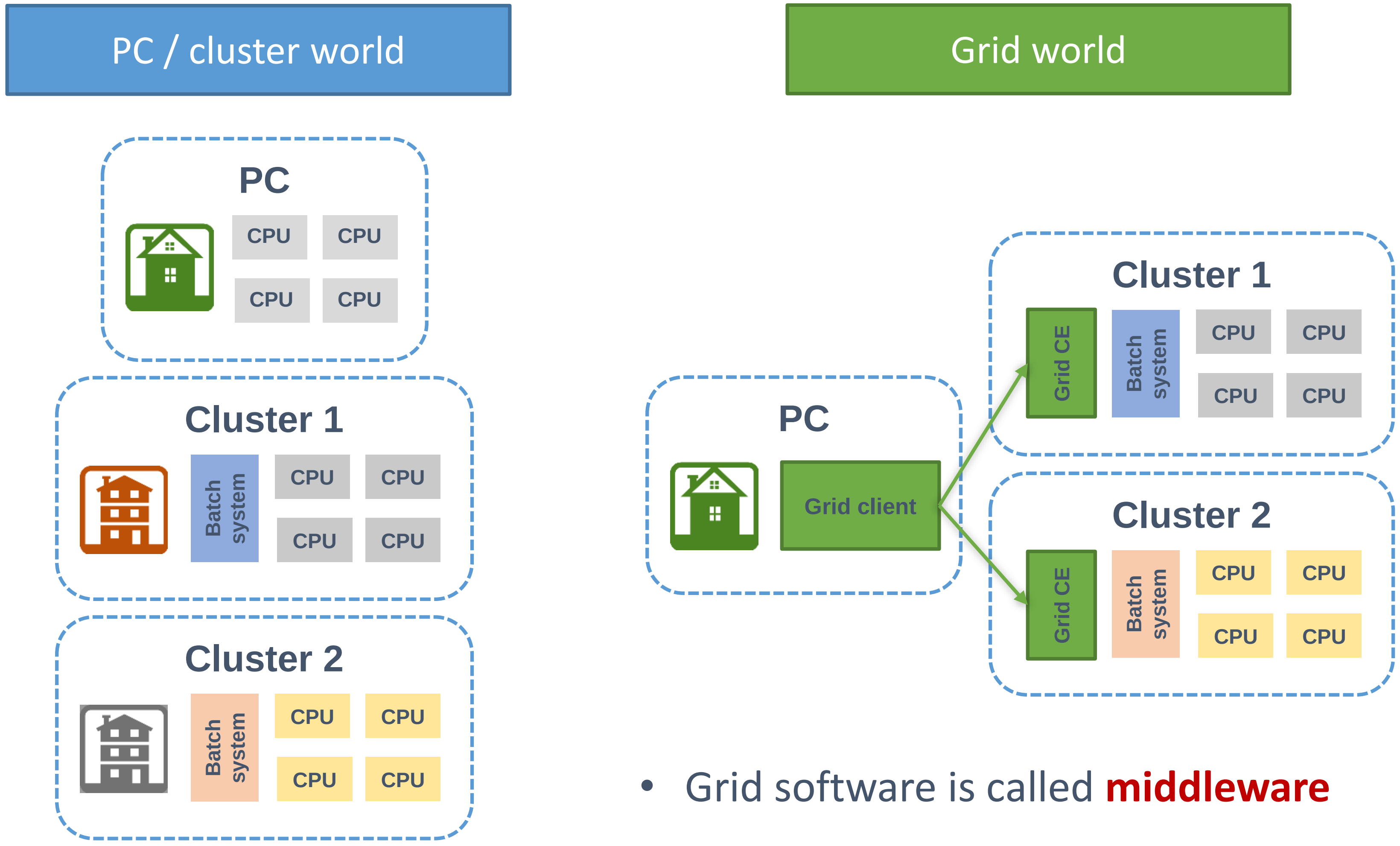
- Original idea **A**:
  - One central service to orchestrate the workload
  - Queue on top of other queues
- Problems:
  - Limited scalability
  - Single point of failure





- 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

Grid as abstraction layer for computing



- Grid software is called **middleware**
  - A layer between the system and applications

Key differences between normal and Grid computing

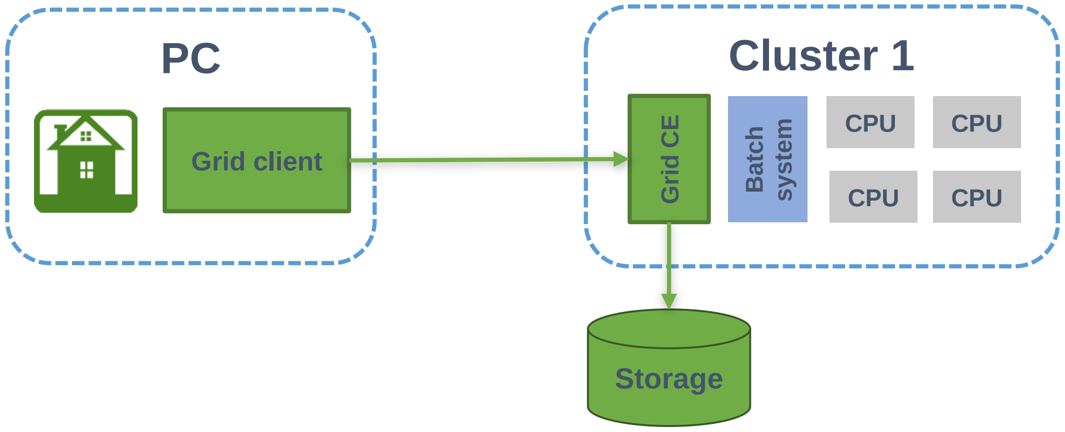
Operation	PC/Cluster	Grid
Log in	Interactive SSH session	No actual log in – proxies are used
	Different passwords	Single sign-on
Job description	Shell script with batch-system-specific variables	Specialized language
	Different scripts for different batch systems	Same document for all systems
Environment	Can be personalized	Pre-defined, generic
	Can be explored in detail	All details can not be known
Job monitoring and management	Requires log in	Remote
Data management	Manual	Can be automatic

Grid job description

- For the purposes of this course, Grid job description is a document prepared by the user
- Job description has a twofold purpose:
  - Specify the **workflow**
    - Executable (your program), input/output files, notifications etc
  - Express job **requirements** such that a matching resource can be found
- Job description can express requirements as a range, or as a condition
  - E.g., at least 1 GB of memory, or use different input if there is little disk space
    - Description received by batch systems must be deterministic, no ambiguities
      - This is why Grid client software may modify job description made by you, by substituting actual available parameters
- All batch systems are different, so a common language never covers all the features

Simplest Grid job submission

- Your Grid client would:
    - Create a proxy
    - Find a cluster on the Grid that matches your job description
    - Submit the job description document to that cluster
- The Computing Element (CE) on the cluster should:
    - Check whether you are authorised
    - Fetch input files (not all CEs can do it)
    - Convert job description to a batch script and start a batch job
    - Upload output files (not all CEs can do it)



Reducing Grid to one cluster, for illustration

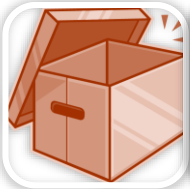
Grid client tools: functionality overview

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

- There are several Grid client softwares around
  - Not all have all the functionalities above
- Most are command-line (CLI) tools
  - Graphical tools are usually too simplistic
  - Scientists like to write own scripts using CLI

Data management: More than just storage

Scientific data management is much more than just storage


Data need to be stored

Data need to be made available for processing

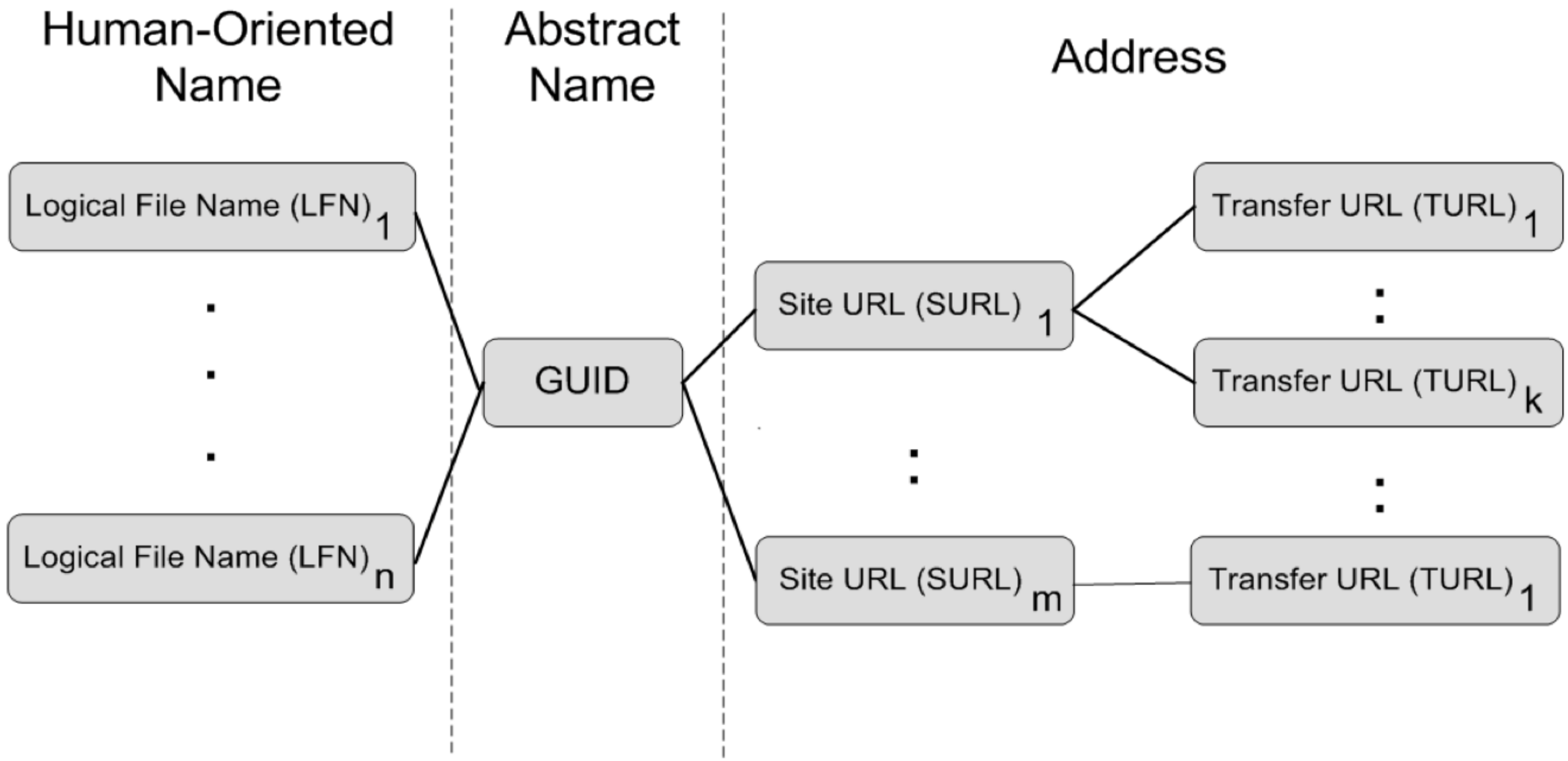
Data have to be preserved for future re-processing

Data need to be shared with other researchers

What does Grid offer today for data management

- **Storage Elements** (SE)
  - Disk/tape storage pools managed by storage middleware
    - Internal space management, shares, some backup etc
    - Grid access control
    - Storage federation (common name space across different SEs)
    - Accounting and information
- File transfer service
  - Designed to transfers millions of files between hundreds of source/destination points
- Data and metadata indexing services
  - Simple file catalogues
  - Application-specific metadata catalogues
    - Attempts to create generic catalogues keep failing
- Client tools for all of the above

Grid file names



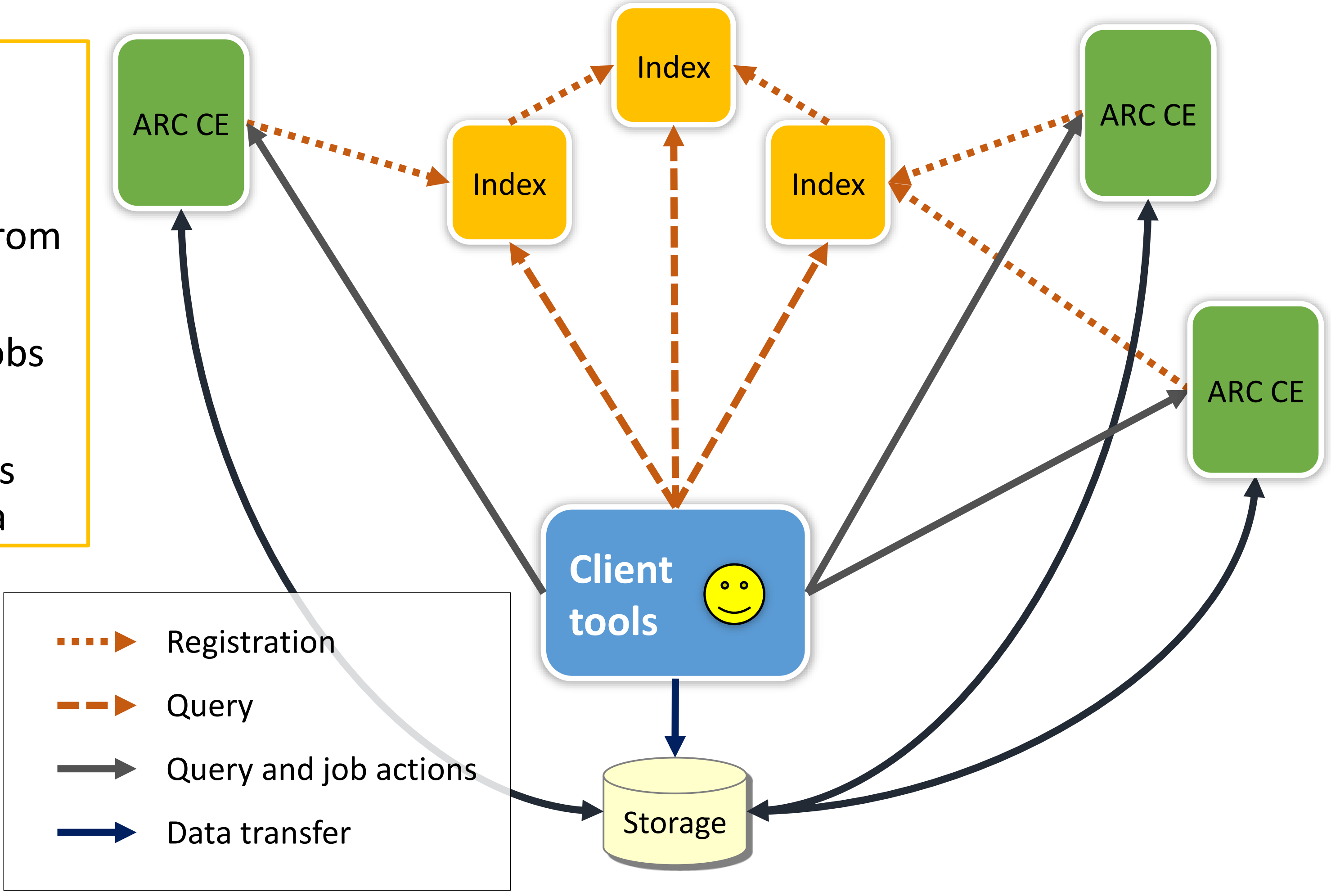
Graphics by StoRM

- LFN example: “data2014-1-raw”
- GUID: 26851250-b9f8-11e3-a5e2-0800200c9a66
    - **GUID: Globally Unique Identifier**; can denote a dataset or a single file
- SURL: **srm**://dcache.swegrid.se/lund/astro/data2014-1-raw.xls – a meta-protocol
- TURL: https://server5.liu.se/pool3/12nsd3/data2014-1-raw.xls

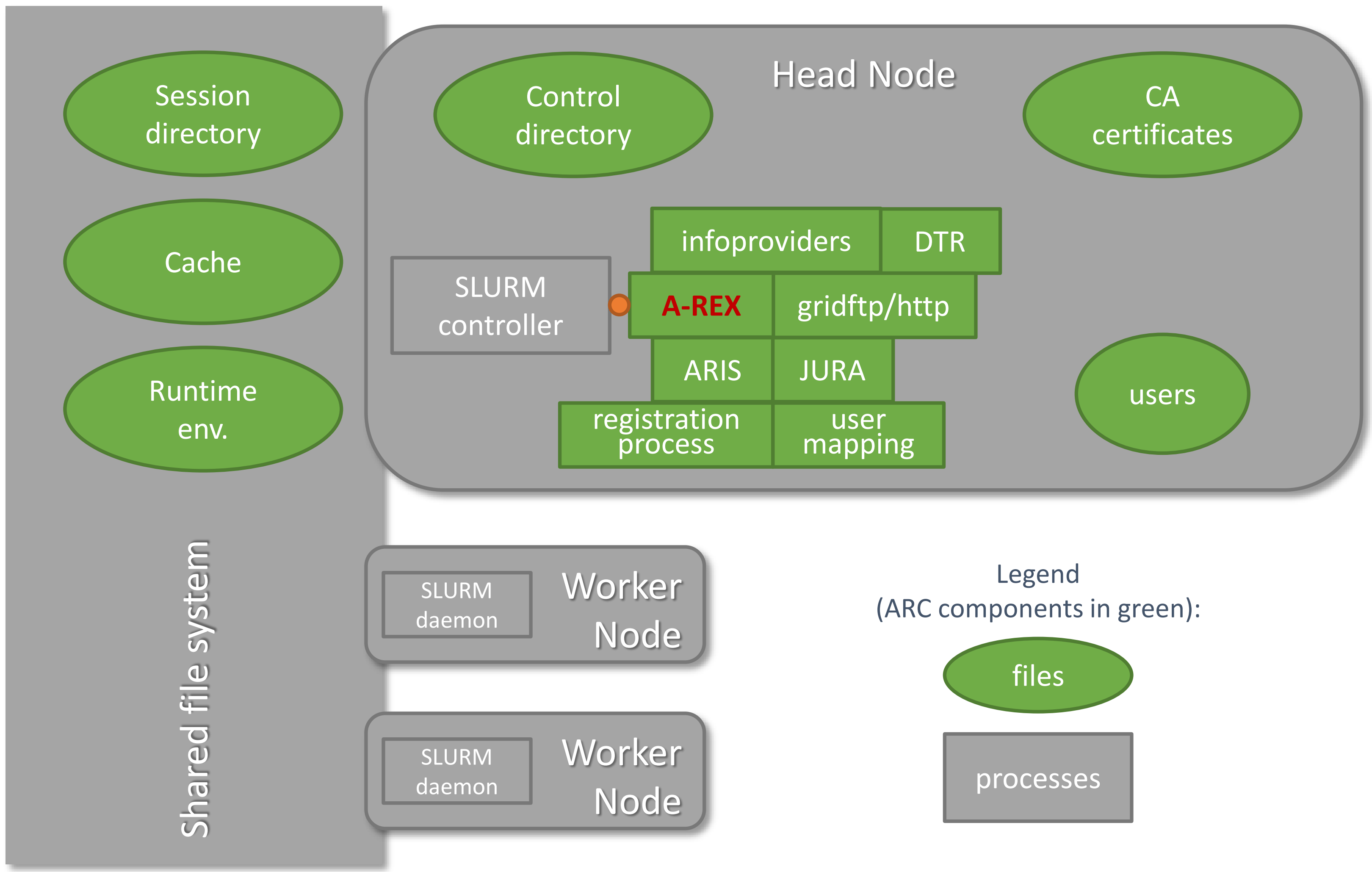
Grid software made in Lund: ARC

- ARC stands for *Advanced Resource Connector*
- Provides a Computing Element, client tools and indexing services

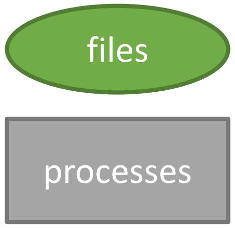
- CEs register to index servers
- Client gets a matching CE from an index
- Client sends jobs to CEs
- Clients and CEs can move data



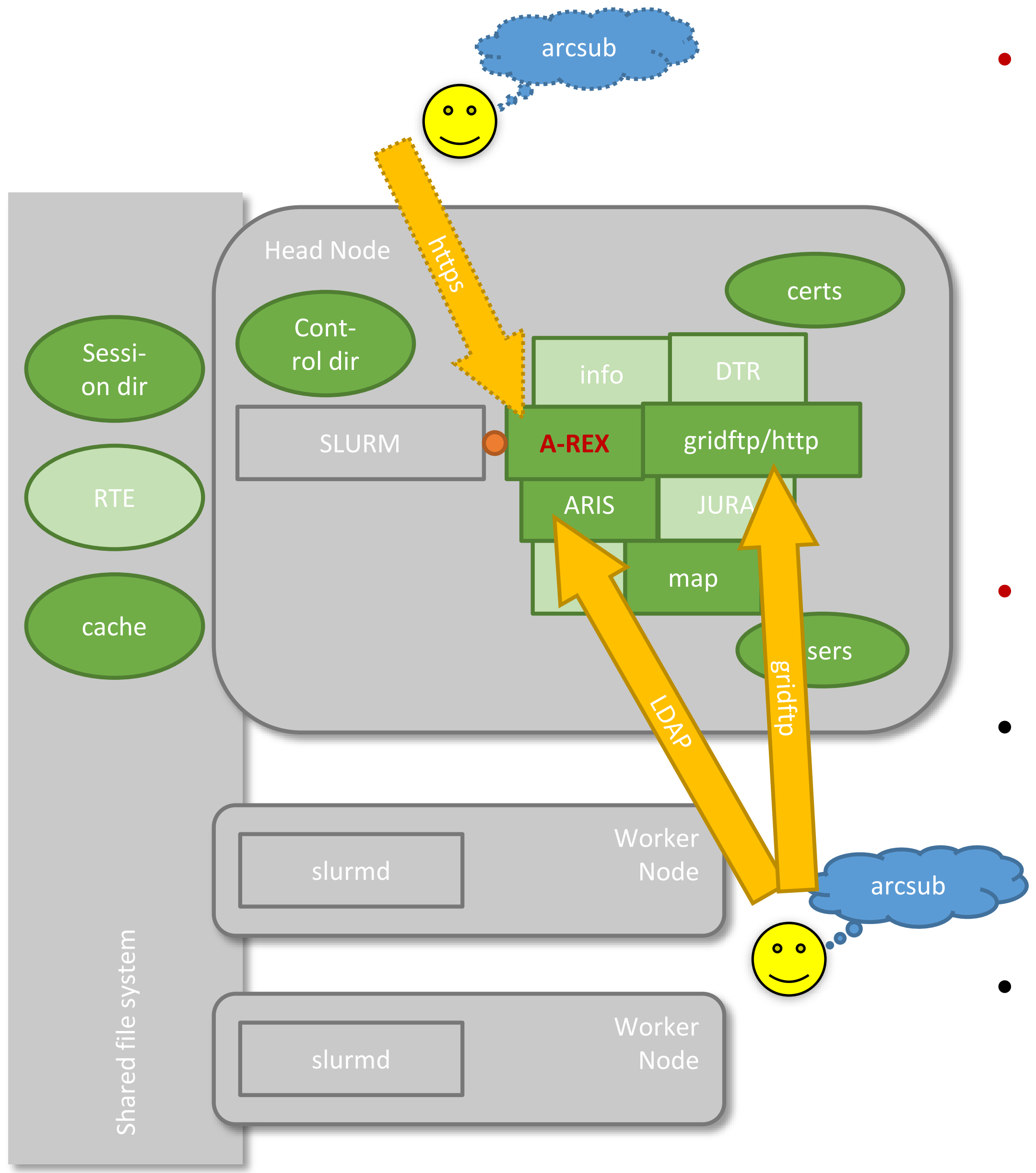
ARC CE components on a cluster



Legend
(ARC components in green):



Job submission in ARC: an overview



- **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
- Currently, information and upload use different protocols
  - https should be used in future for better consistency
- All steps require **authorisation**

Conclusion

- When you have many similar jobs to execute, use batch systems
- When you have way too many similar jobs, use Grid
  - Especially if it is about data analysis
- Grid is designed to provide access to many different computers
- Grid security is a very difficult concept to grasp
  - Even though it is the same technology as e.g. used by banks
- There are many different Grid clients
  - Each organisation – virtual or real – comes with own set of tools
  - There is no one established Grid language
- Once you learn how to handle proxies and describe jobs, you can get access to community resources all over the world