Introduction to Programming and Computing for Scientists

Oxana Smirnova

Lund University

Lecture 4: Distributed computing, security

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

A *supercomputer* or a *cluster* is a <u>system</u> of many (thousands) processors. In supercomputers, CPUs share memory, in clusters – usually not.

- One system serves many users
- One user can use many systems
- Systems are typically provided as public service (by universities and labs)

- Systems are customized, but each can serve many different users
- When many different systems jointly offer common services (login, storage etc), they create a computing <u>Grid</u>

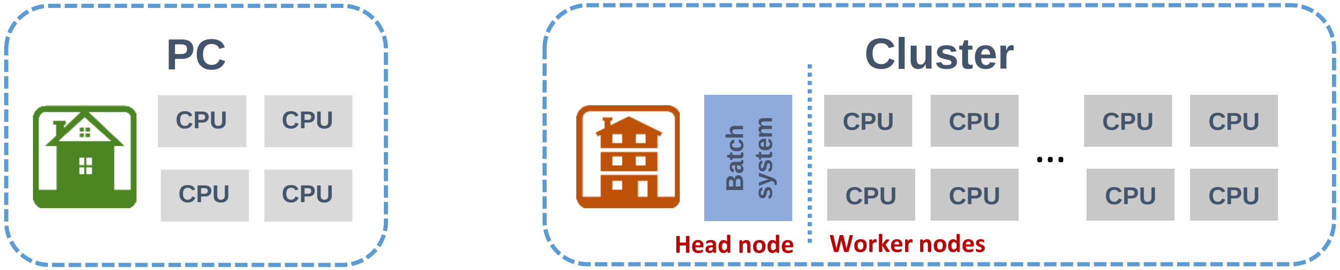
Generic service for rent – Clouds

A system built of virtual machines is called a <u>Cloud</u>

- There are clouds for computing, data storage, databases etc
- Originally appeared as a business concept, but can be used as a public service

- Each Cloud is different, but each can be (seemingly) infinite because of virtualization: “elasticity”
- Users can customize their “rent”
- No high performance

Big machines for big data: clusters



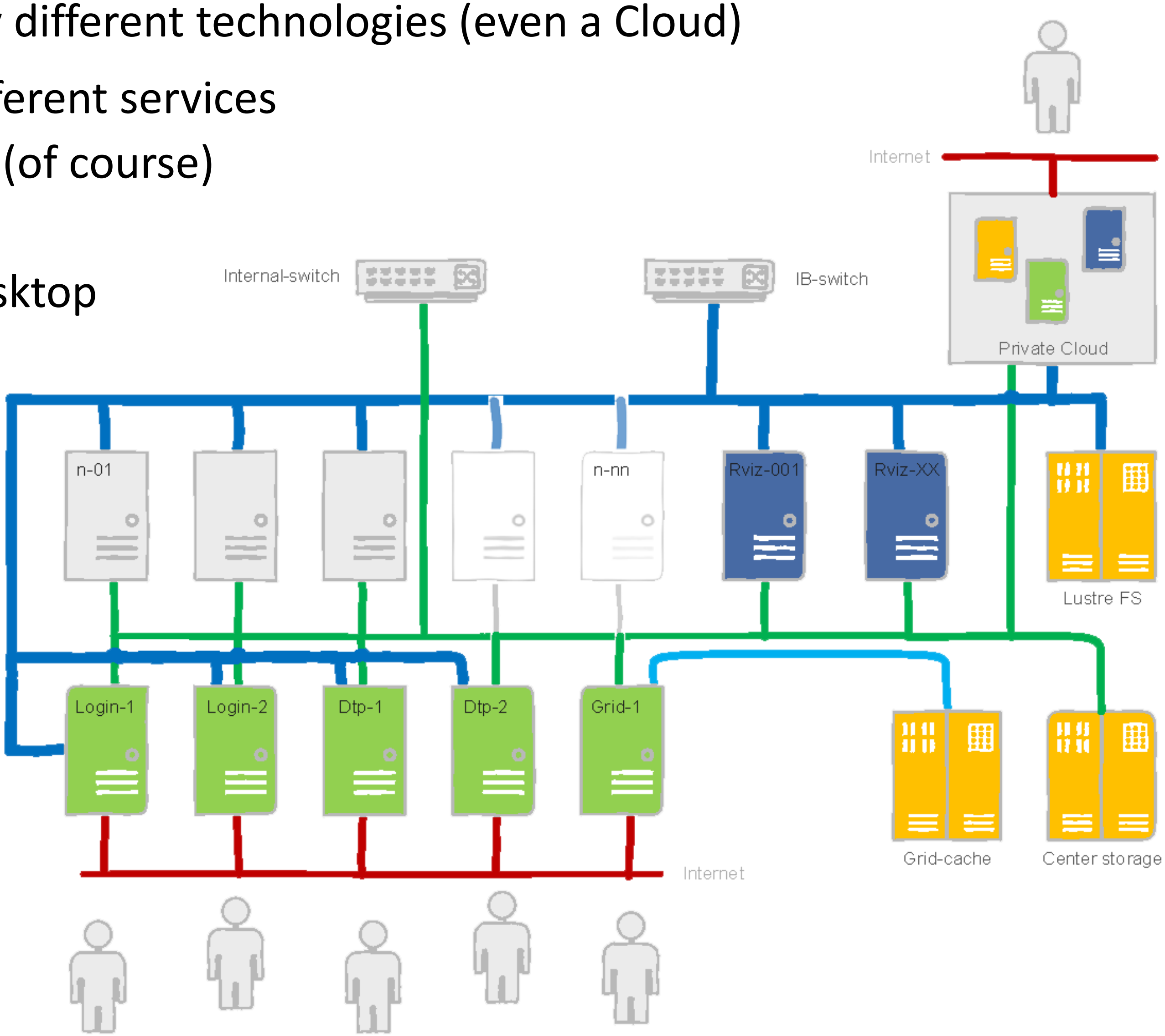
- Computing facilities in universities and research centers usually are Linux **clusters**
- A cluster is a loosely coupled computing system
  - Users see it as a single computer
  - A typical cluster has a **head node** and many **worker nodes**
    - A *node* is a unit housing processors (CPUs, cores) and memory – basically, a PC box on steroids
  - Distribution of work 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* etc
- Every cluster is a heavily customised system built for a range of specific tasks

Clusters in the LUNARC center

Photo: Gunnar Menander, LUM

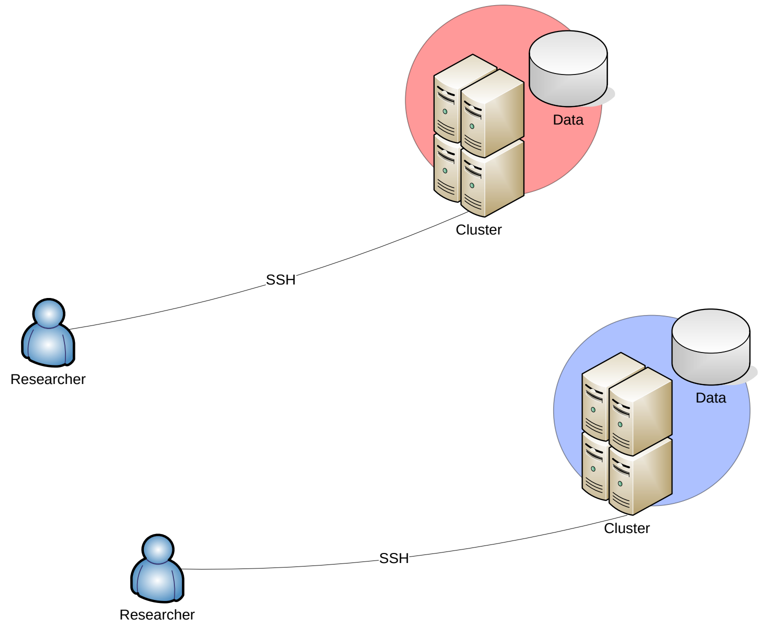
AURORA cluster at LUNARC

- 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 and supercomputers



- Users connect to the **<u>head node</u>**
  - 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 **<u>batch systems</u>**
  - A task can be anything, from adding 2 and 2, to bitcoin mining
  - 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, for example:
  - A job can use a single CPU 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 that of a serial job
  - 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 CPU resources, jobs are waiting in a **queue**
  - A cluster may have several queues for different kinds of jobs (long, short, parallel, etc)
  - Queues exist even if there are no jobs – like cashiers in supermarkets

Different tasks use different jobs

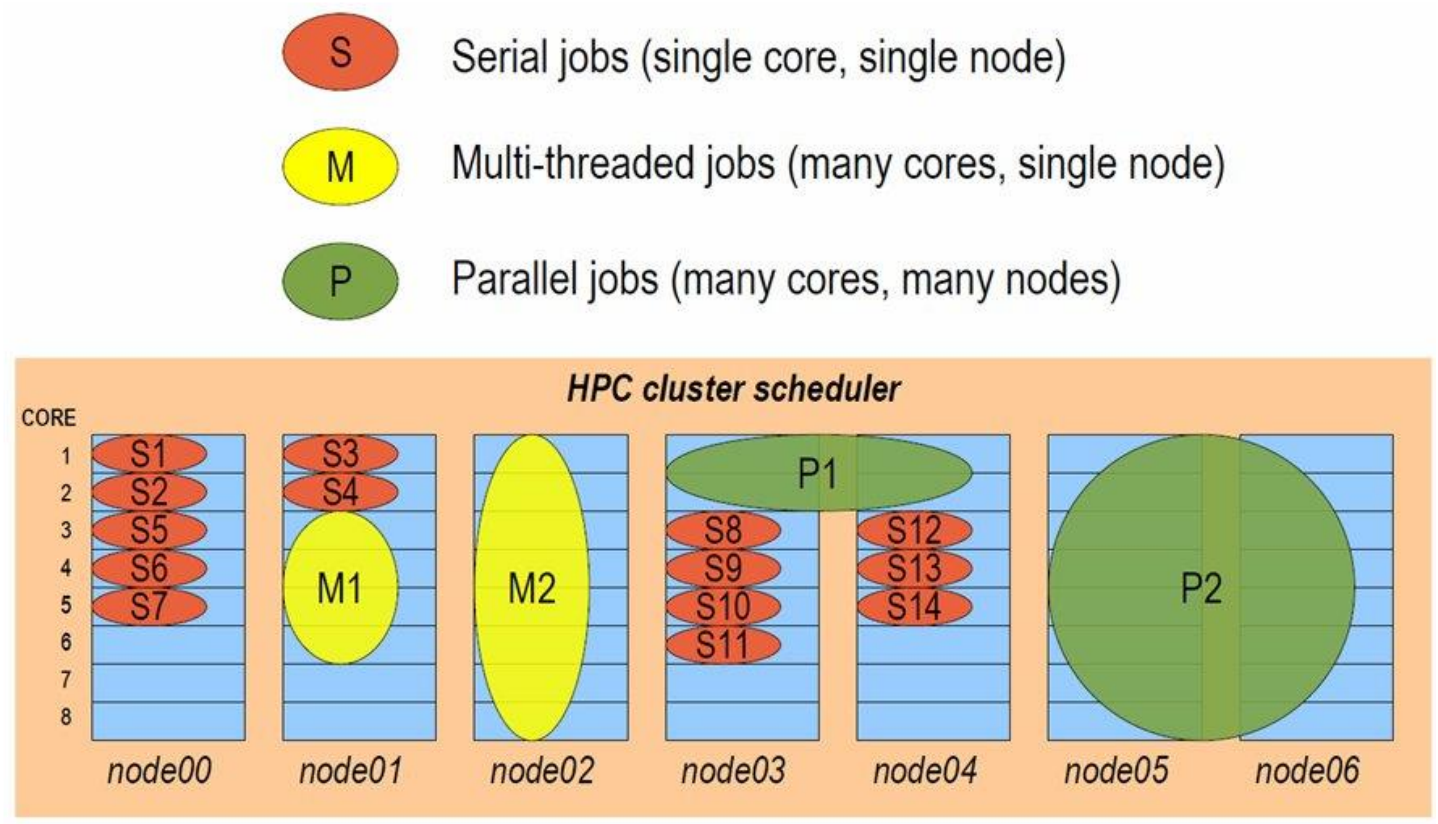
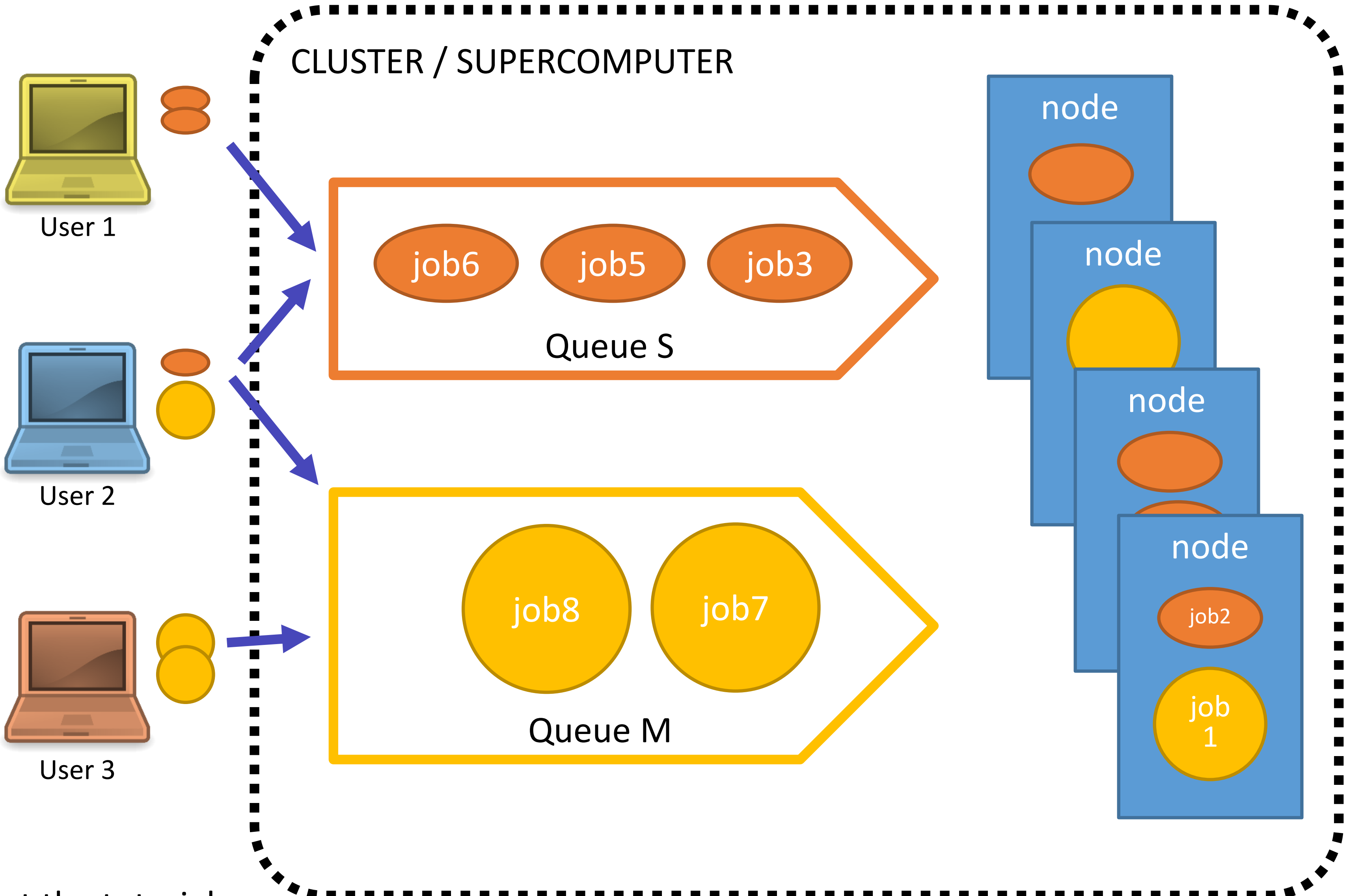


Image from https://alces-flight.com

- Batch scheduling is like playing Tetris – only you can't rotate pieces *(says Gonzalo Rodrigo)*

Scheduling usually relies on queues



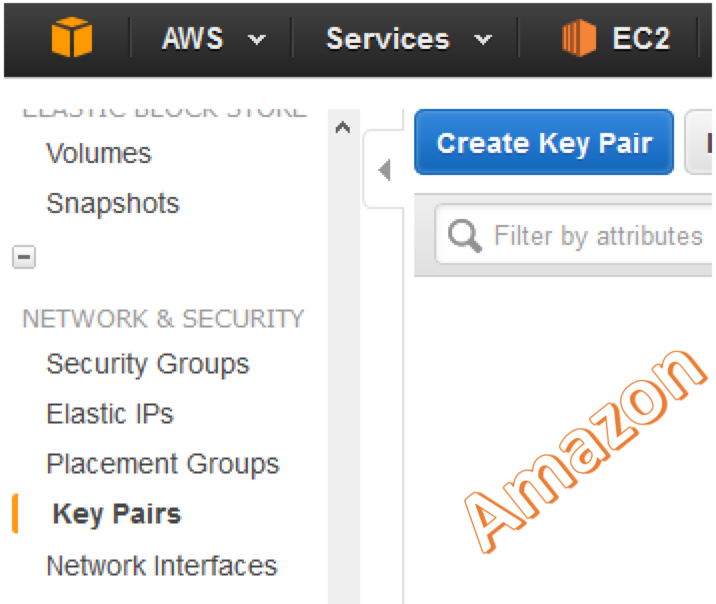
More at the tutorial...

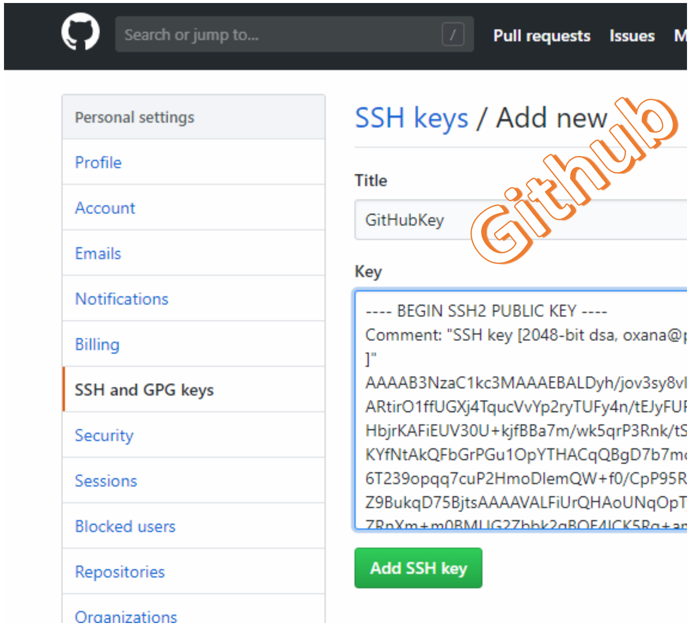
To access computers, you need a permission

- To access one computer (or one cluster) you need a password
  - You also have a personal user space (account)
- Now scale it up 100+ computers, clusters, clouds, and 1000+ users
  - You can’t quite remember 100+ passwords
  - Sysadmins can’t quite manage 1000+ user accounts

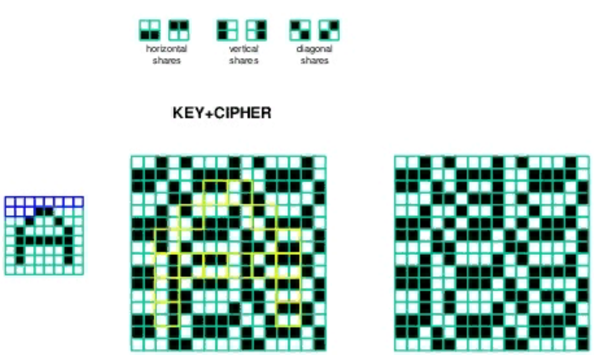
- Cryptography to the rescue!
  - Many different ways to securely access remote services exist, all based on cryptography methods
  - We will explain only a few

Many cloud services and clusters use SSH key pairs





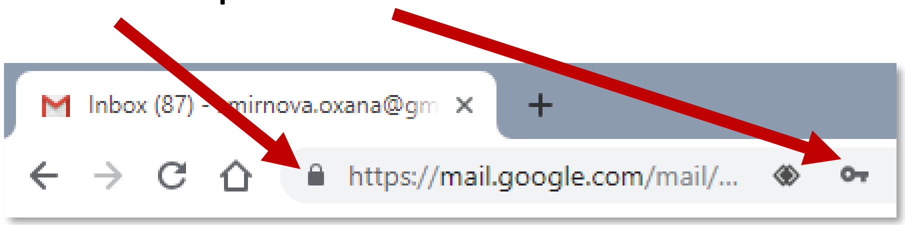
- Secure Shell (SSH) is when your bash (tcsh, zsh etc) session is on a <u>remote</u> machine connecting via an **encrypted connection**
- To encrypt anything, you need **encryption keys**



Can we trust all keys?

- Anybody can create as many SSH keys as they wish: no protection from rogue actors!
- 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, internet shop, Web service, database, online bank etc
    - Sometimes you need services to act on your behalf: delegate your rights to them
      - For example, delegate to Twitter rights to post on Facebook on your behalf

- All secure Web sites are protected by PKI

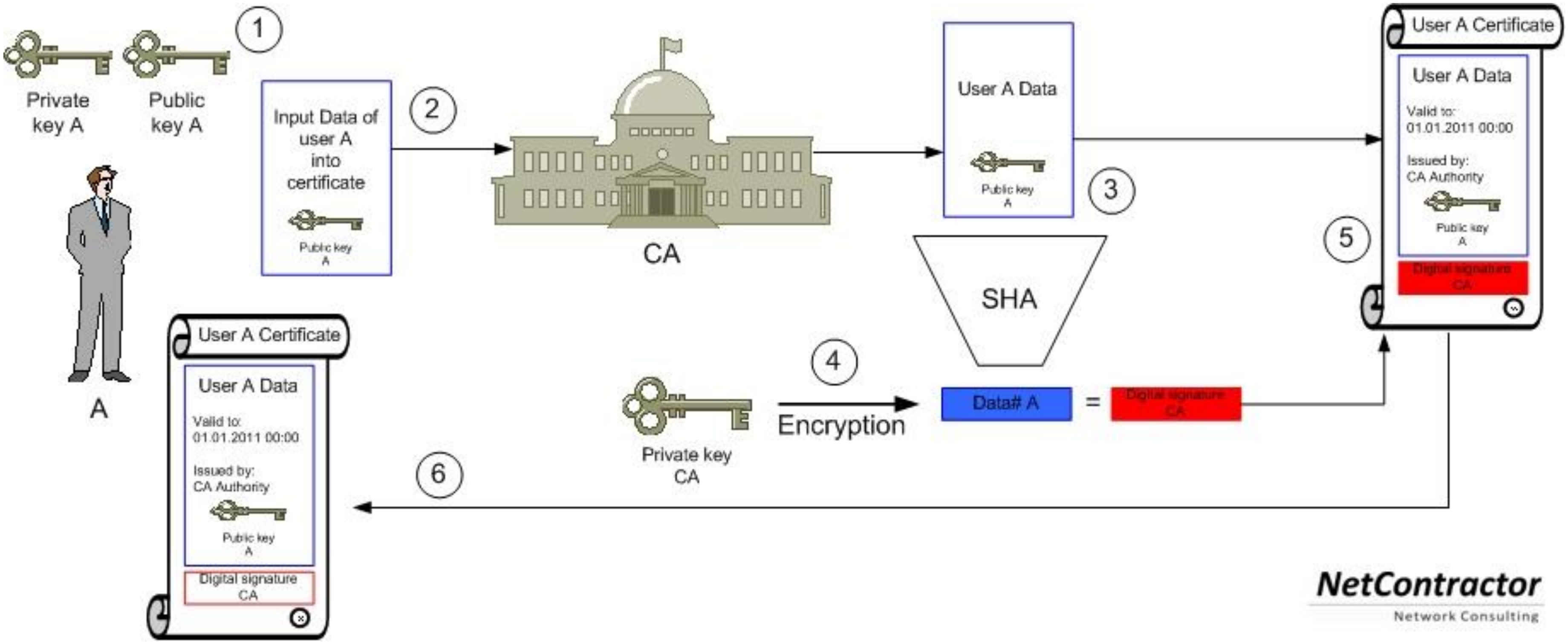


Principles of PKI

- 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

A CA is just a group of trusted people who have a procedure to check who you are (for example, check your passport)

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
  - Tools like Telegram store them in your device
  - **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
    - Browsers do it
    - **This file must not travel!**

Protocols and systems using public key cryptography

- A *protocol* in our context is a formal procedure of information exchange; it can be insecure (plain data exchange), or secure – involving cryptography
- Some examples:
  - SSH: used to login remotely to computers
  - SSL and TLS: used e.g. in https, Gmail
  - GridFTP: a secure variant of FTP
  - ZRTP: used by secure VoIP
  - PGP and GPG: used e.g. to sign software packages or sign/encrypt e-mail
  - Bitcoin and other cryptocurrencies
    - Used to ensure authenticity of transactions and individuals
    - Proof of mining work

X509 flavour of PKI

- Several implementations of PKI exist
- Arguably the most secure is the **X.509** PKI standard (used e.g. by 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 13 months
  - Assumes strict hierarchy of trusted CAs
    - Unlike PGP, where anyone can vouch for anyone
    - You can check your browser for a 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, revocation lists

- Web browsers and even operating systems come with a list of trusted root CAs
  - It means the browser has their public certificates included
  - You can always remove untrusted CAs, or add own trusted ones
    - When you remove a CA, you won't be able to securely connect to a server certified by that CA
    - You can even establish an own CA – if anybody trusts you…
- Certificates of people and services can be revoked
  - If they are compromised, or if some information in the certificate is changed
- 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 and are regularly updated

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 that want to establish a secure connection 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: Acting on behalf of users

A “normal” computer / cluster usually has local storage

Same user identity is used for jobs and data access

Scientific data are stored all over the World

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

A computer’s own certificate is not enough

Users want to protect their data from unauthorised access

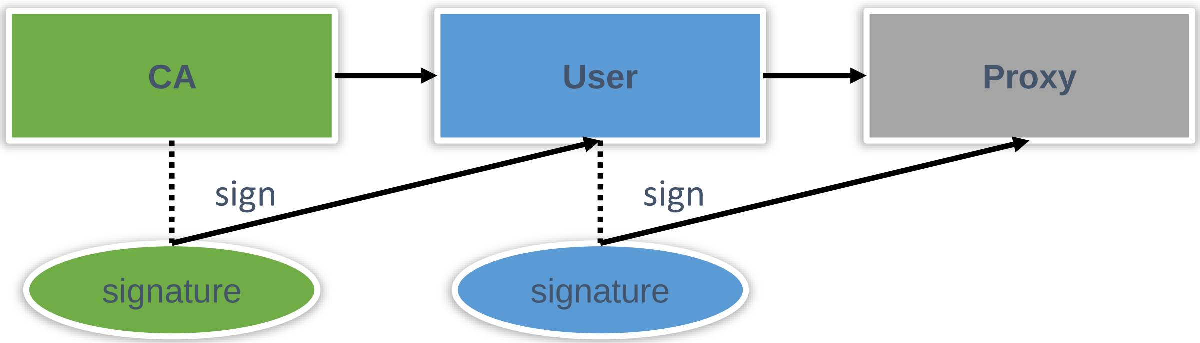
Users also don’t want everybody to write to their storage share

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

Delegation in X.509: 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
- In the PKI context, 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



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: you just sign the one created by the service

Proxies must have **very short lifetime:**

- Reduces the chance of getting stolen
- Minimizes the damage

How is a X.509 proxy created?

- A **<u>new</u>** private/public key pair is created for each proxy
  - When a proxy <u>expires</u>, a new one must be created to continue working
    - Default expiration time is 24 hours
- A proxy is then constructed of:
  1. <u>Public certificate</u> (with public key embedded)
     - Certificate contains modified owner's Distinguished Name (has "*proxy*" appended to the name)
       - Owner's DN:
         `/C=UK/O=Grid/OU=CenterA/L=LabX/CN=john doe`
       - Proxy DN:
         `/C=UK/O=Grid/OU=CenterA/L=LabX/CN=john doe/CN=proxy`
     - Certificate is signed by the proxy owner's **real** private key
     - Certificate contains validity period
  2. <u>Private key</u>
  3. Optionally, <u>Attribute Certificates</u> – extensions containing additional information

Delegation: The tale of two proxies

- A user always has to create a proxy certificate **P1**
  - Technically, it can be sent to the server, but it is a **security breach**
- A server creates itself a **delegated** **proxy P2** upon every user request:
  1. Server generates a **new** private/public key pair (yes, that's a 3rd one...)
  2. Server returns the generated public key as a certificate sign request to the user
  3. User's tool signs that public key and inserts user information (DN etc), thus generating a public certificate. It uses the private key of proxy **P1** for performing signing operation.
     - It can also use the actual private key, but that will require entering password every time!
  4. User's tool sends the signed public certificate back to the server
  5. Server adds generated private key to that certificate and creates a delegated proxy **P2** and now can act on behalf of users without compromising their private keys

Sounds complicated, but it never been compromised

It is used for Large Hadron Collider computing

Authentication is not enough: we need 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 services

- 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 in X509:
  - By person: your DN is in the trusted list on a cluster (matched to your proxy)
  - By group: your DN is in the **Virtual Organisation** *(VO)* list
    - Your proxy has this VO's *Attribute Certificate*
- Unfortunately, Virtual Organisations are not well defined and difficult to work with
  - They are not supported by browsers either

Another way of delegating: OAuth2

- Did you encounter "Log in with your Facebook account" in Twitter or suchlike?
  - Facebook, Google and others rely on delegation protocol OAuth2
- OAuth2 is Open Authorisation 2.0
  - Free and open standard protocol
  - Designed to delegate <u>authorisation</u>
  - Instead of using proxies, it uses <u>tokens</u>


```
{
  "sub": "e1eb758b-b73c-4761-bfff-adc793da409c",
  "aud": "iam-client test",
  "iss": "https://iam-test.indigo-datacloud.eu/",
  "exp": 1507726410,
  "iat": 1507722810,
  "jti": "39636fc0-c392-49f9-9781-07c5eda522e3"
}
```


A token body example by A.Ceccanti

- OAuth2 actors:
  - User is a *Resource Owner* (you own your identity info and other data)
  - User's data are in the *Resource Server* (e.g. Facebook)
  - User uses a *Client* to act on his behalf (e.g., use Twitter to post images to FB)
  - Authorisation is handled by *Authorisation Server* – it is the one issuing access tokens to Clients
    - Resource Server and Authorisation Server can be the same, as in FB

Basic OAuth delegation process

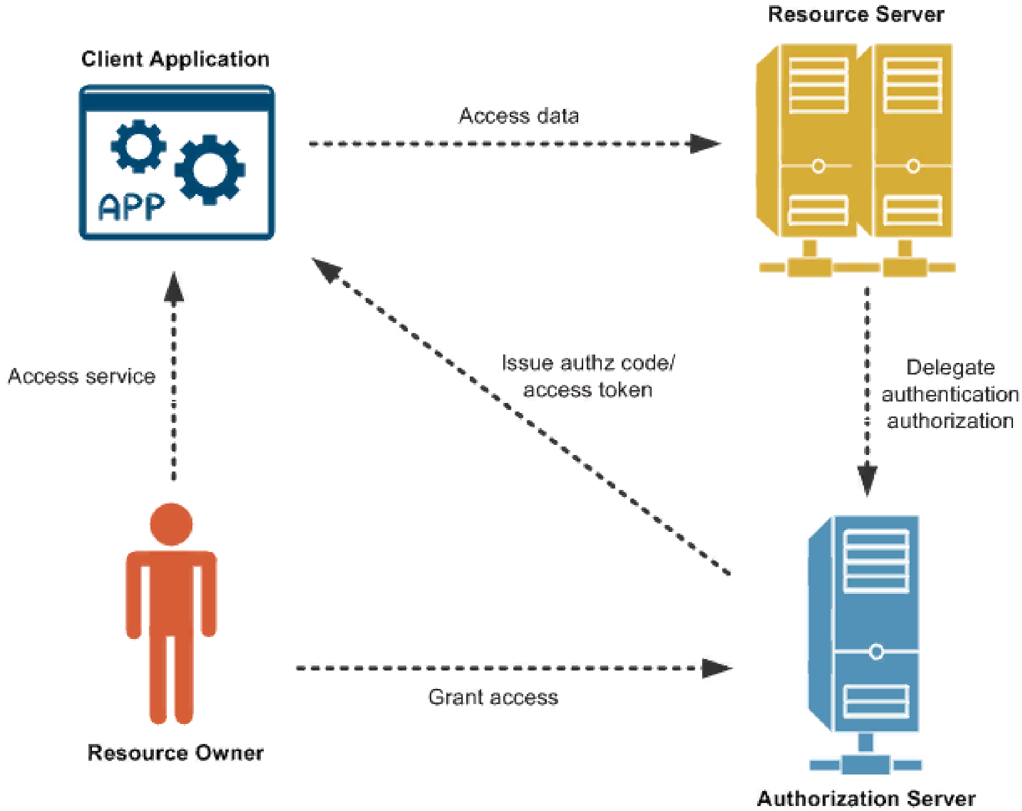


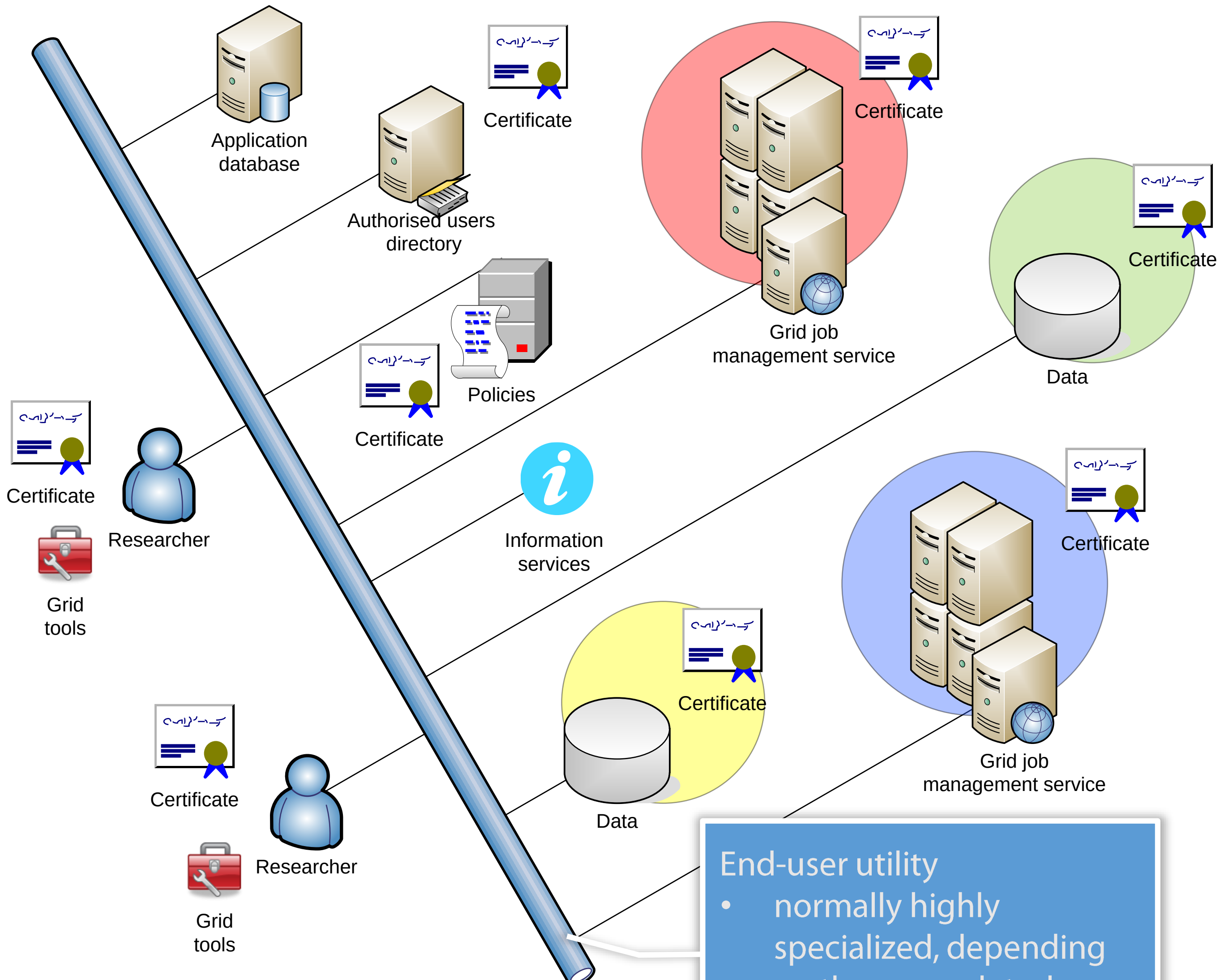
Diagram by ORACLE

- OAuth2 actually does not require cryptography client-side (but needs https)
- OAuth2 access tokens are short-lived
  - One can use long-lived refresh tokens to obtain new access tokens

Why scientific computing needs all this?

- More scientific data need more computing and storage than exist in one lab
  - Nobody likes to wait in a queue!
- 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
  - Two types of community computing exist:
    - Volunteer computing (google for BOINC): individual PCs
    - Grid computing: jointly working resources of scientific communities, workhorse of CERN

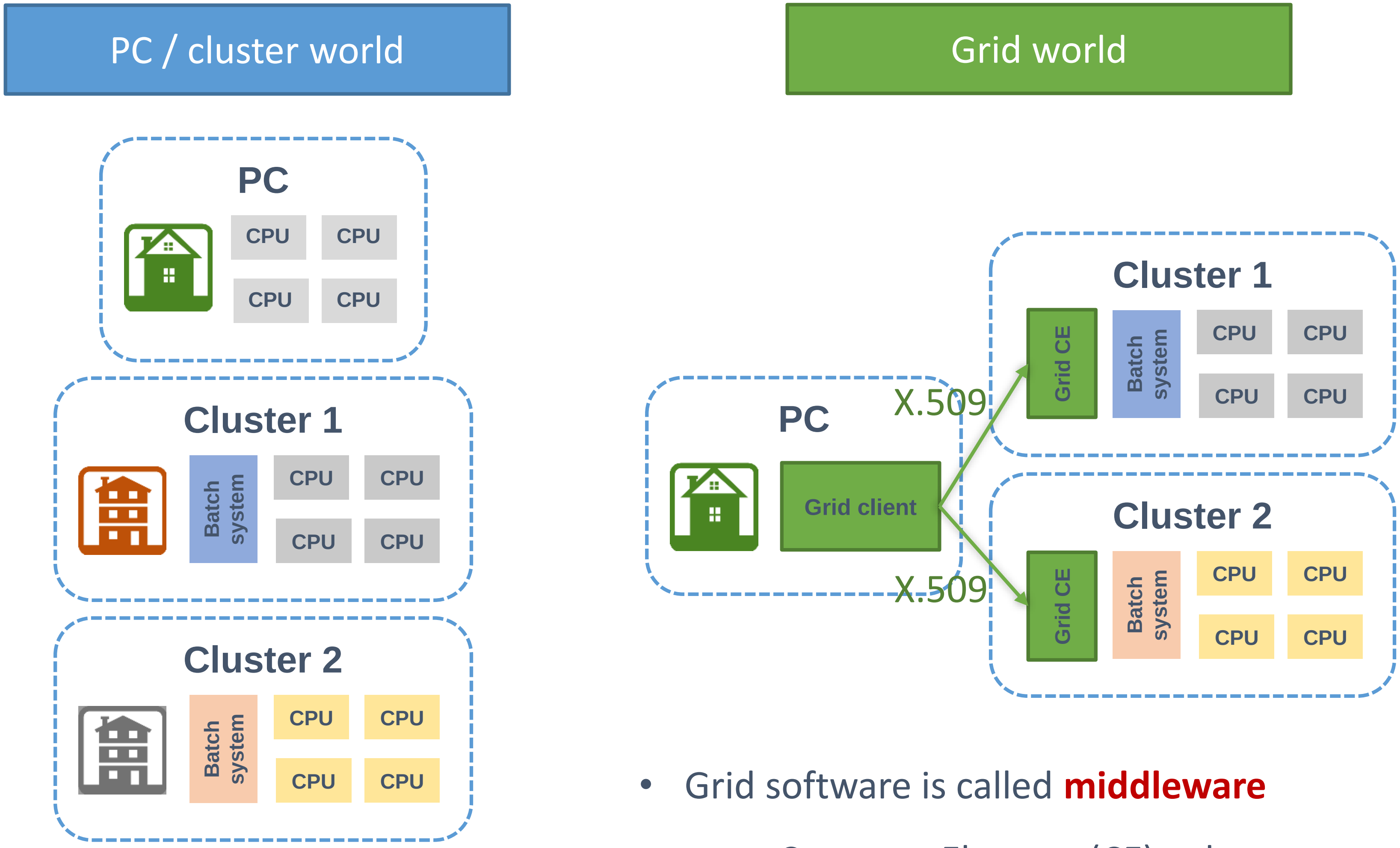
Overview of generic Grid components



End-user utility

- normally highly specialized, depending on the research task

Grid as abstraction layer for computing



- Grid software is called **middleware**
  - Compute Element (CE): a layer between the system and applications

Summary

- When you have too many similar jobs to execute, use a batch system: High Performance Computing (HPC)
- When you have too many HPC systems, use distributed computing solutions: CERN uses Grid
- With great power comes great responsibility: always use secure access to powerful computing resources and valuable data storage
- Never EVER share your passwords and private keys!