LUND UNIVERSITY

Distributed Computing Concepts and Tools

COMPUTE RESEARCH SCHOOL COURSE NTF004F

Outline of the course

- 8 lessons:
  - Lectures followed by active learning hands-on exercises
- Home assignments
  - Bring your work home
- Final project
  - Writing a toy proposal for a research computing infrastructure
  - Students are expected to demonstrate understanding of basics of distributed computing and research data management

Lessons

1. Introduction: from traditional computing to distributed. Security considerations, certificates, authorisation, tokens and delegation
2. Distributed computing services: cluster grids, HPC systems, clouds
3. Active learning: LUNARC/Iridium, batch system
4. Virtualisation, containerisation (Docker/Singularity)
5. Principles of scientific data management, big data workflows
6. Active learning: Hands-on tutorial: Rucio tutorial
7. Active learning: ARC-CE installation
8. Project team work

Introduction

BASIC CONCEPTS

Which is your computer?

	Processor cores	Storage per core, TB	Operating system (typical)	Real	Virtual ("cloud")
Personal computer (workstation)	$10^0 – 10^1$	$10^0 – 10^1$	Windows, MacOS, Linux	✓	✓
Cluster (farm)	$10^2 – 10^3$	$10^0 – 10^1$	Linux	✓	✓
Supercomputer	$10^4 – 10^6$	$10^{-3} – 10^{-1}$	Linux	✓	

- Deviations exist, boundaries are sometimes blurred
  - Fast interconnect between processors is a distinct property of supercomputers
- For the purposes of this course, this classification is sufficient

LUND UNIVERSITY

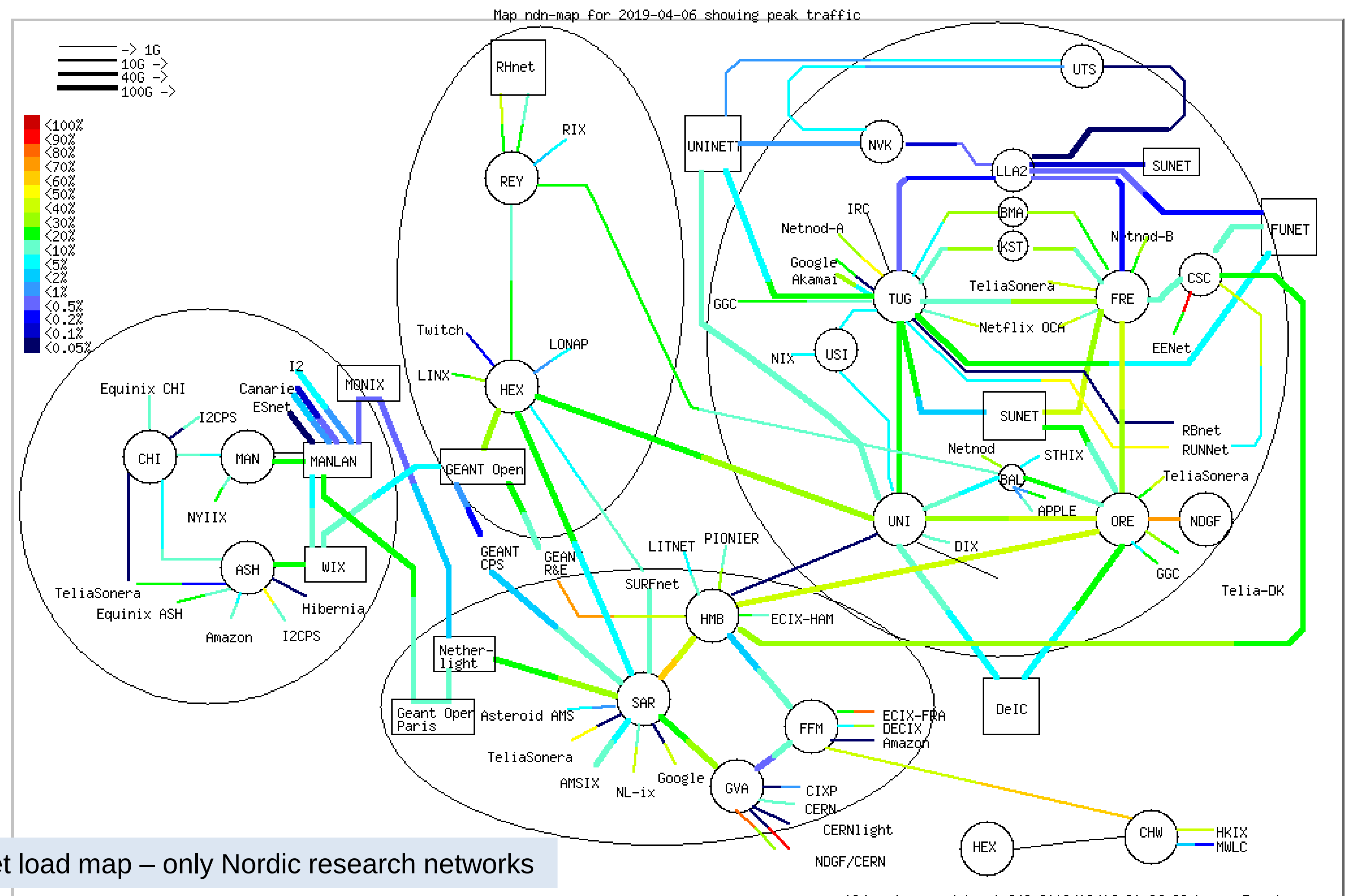
Big data need big computers

- Even the most advanced desktop workstation will take years to process Petabytes of data
  - And will require a dedicated network connection to transfer all that
- Simulation of a statistically significant sample on a workstation will take years

But we need our Nobel prize tomorrow!

- It took ~2 weeks of massive data processing to find a hint of the Higgs boson – the fastest discovery of this kind
- Solution: use supercomputers or large computer clusters, with large attached storage and very fast network
  - Network: 10 Gbps is the baseline now, 1 Tbps in some near future
  - Computers: custom systems, not any two alike
  - Storage: a real issue, is never enough

The invisible world of research networks

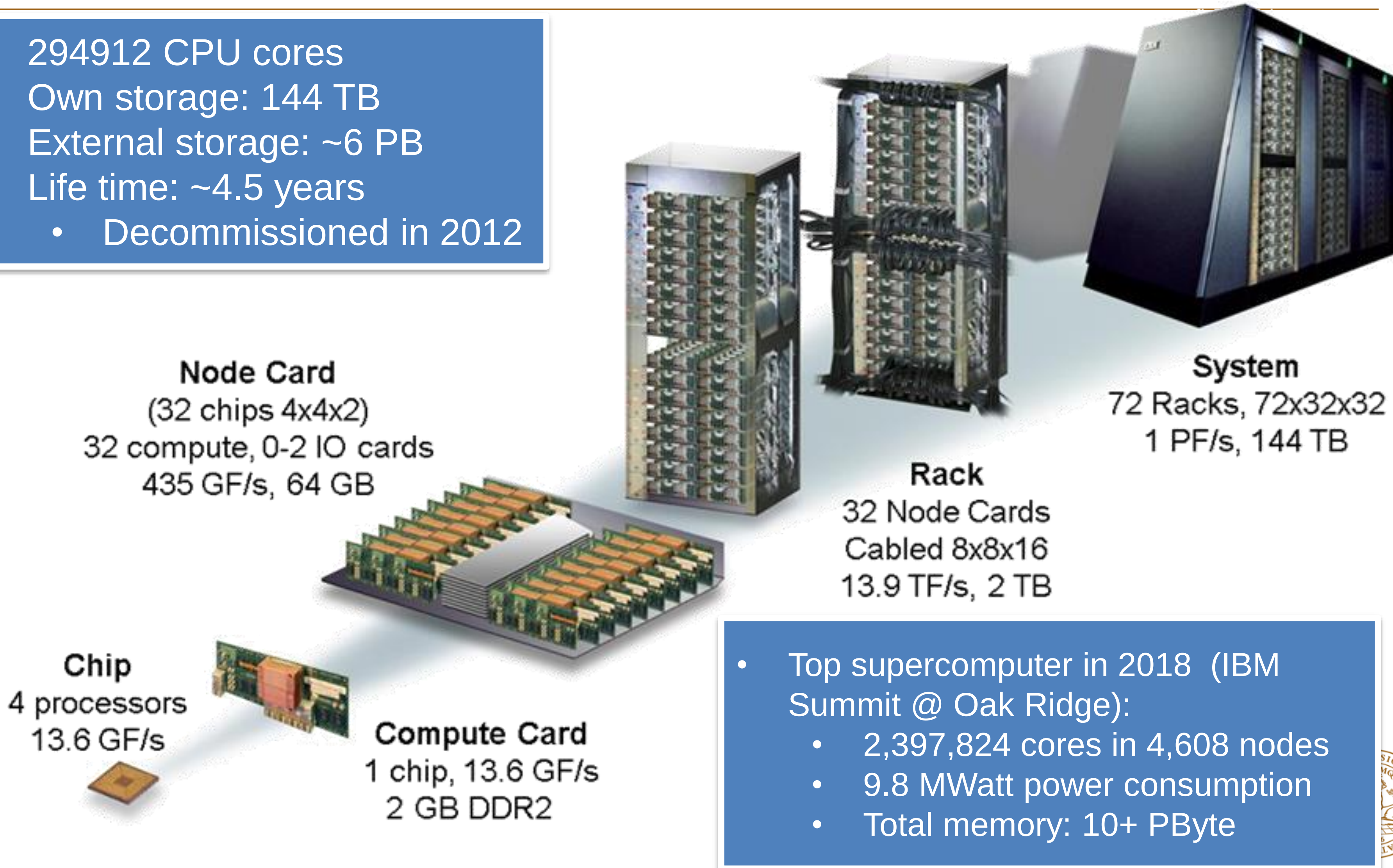


NORDUnet load map – only Nordic research networks

LUND UNIVERSITY

An (old) supercomputer: Blue Gene/P

- 294912 CPU cores
- Own storage: 144 TB
- External storage: ~6 PB
- Life time: ~4.5 years
    - Decommissioned in 2012



Graphics by IBM

- Top supercomputer in 2018 (IBM Summit @ Oak Ridge):
    - 2,397,824 cores in 4,608 nodes
    - 9.8 MWatt power consumption
    - Total memory: 10+ PByte

Linux clusters

A very old traditional Linux cluster

Aurora Linux cluster in Lund (LUNARC center)

Computer memory vs storage

- In what follows, "memory" normally means primary memory – volatile (non-permanent), high-speed access
  - Non-volatile storage is often slower (esp mechanical), can be referred to as secondary memory, but we will call it **storage** (disks, flash memory etc)
  - Primary memory written to secondary memory is called virtual memory
- PCs and clusters have similar architectures memory-wise, while supercomputers share memory globally between cores
  - This is why supercomputers can not be virtualized
  - Memory in PCs and clusters can still be shared programmatically

LUND UNIVERSITY

Memory modules

Memory in an HP server

Memory for a SUN blade server

High-performance memory for professionals

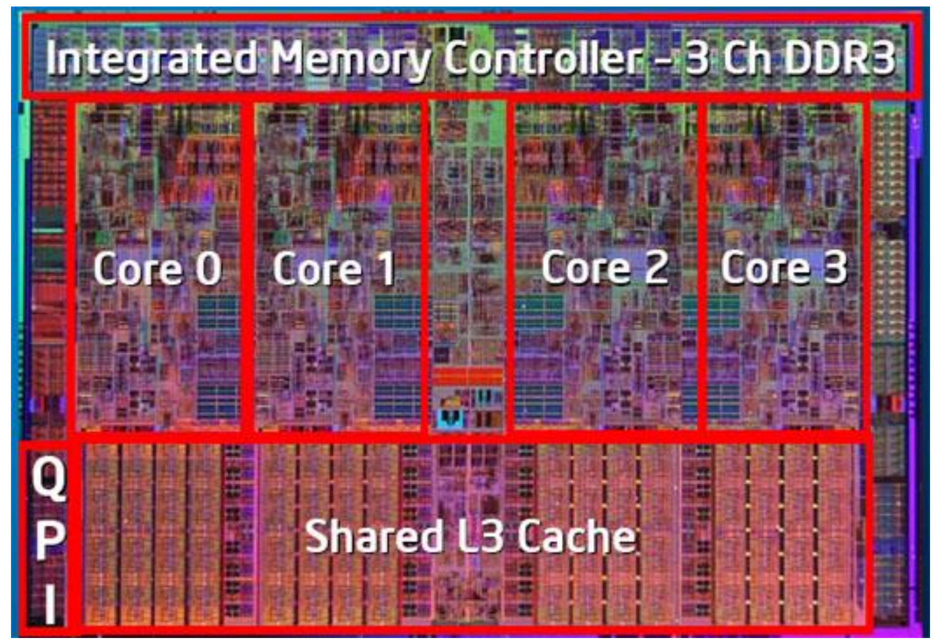
CPUs and cores

- CPU – Central Processing Unit – a chip that performs arithmetic, logical and input/output operations (computing and decision-making)
  - Programmable device, reads binary instructions and data, processes the data and outputs results
- Modern chips contain several units and are referred to as multicore processors
  - Terminology is still confused: some call each processor a core, and the multicore chip – a CPU. Others call each core a CPU.
  - Cores on one chip usually share memory and input/output channel.
- GPU – Graphical Processing Unit – chips optimized to process graphics, good for parallel data processing
  - GPUs are normally optimal for low-precision repetitive operations, as in e.g. neural networks training

Traditional Processing Units

CPUs

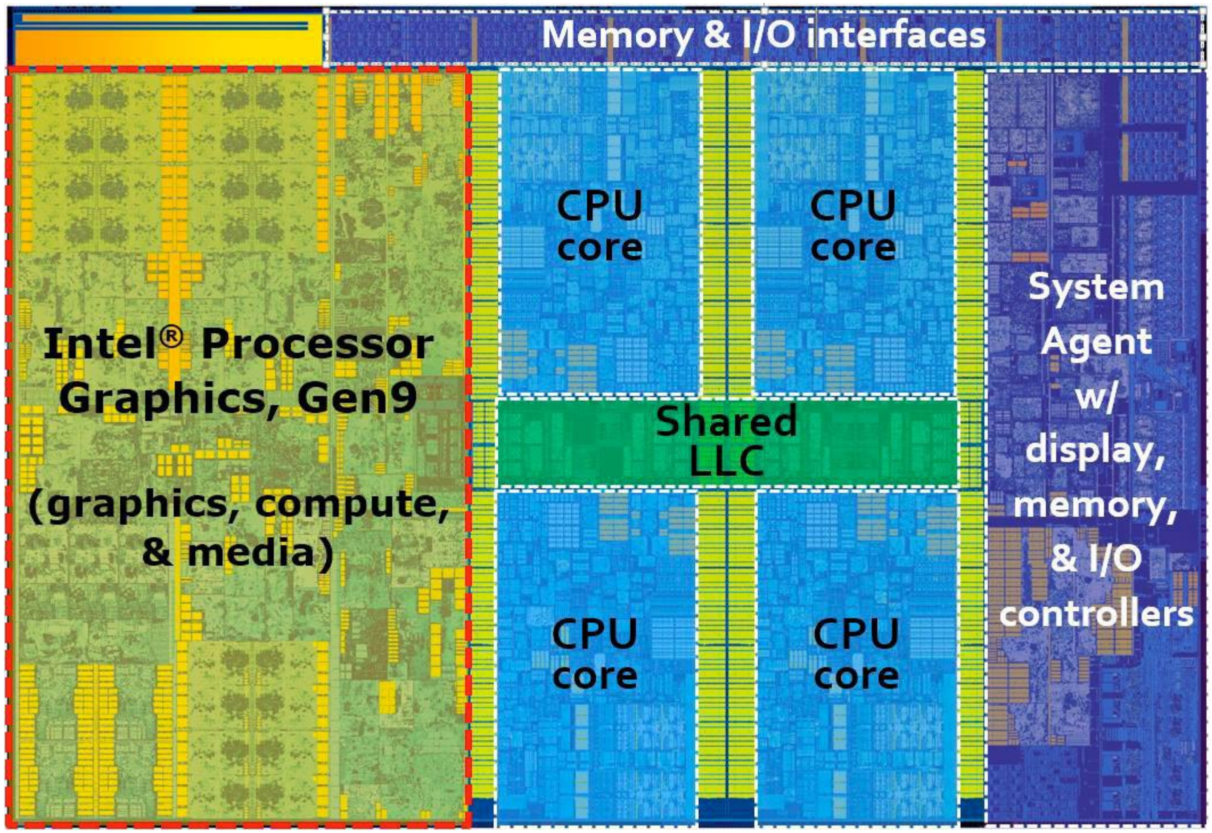
Die map of a multicore CPU (old Intel's Nehalem)



GPU

LUND UNIVERSITY

Modern approach: integrated CPU and GPU

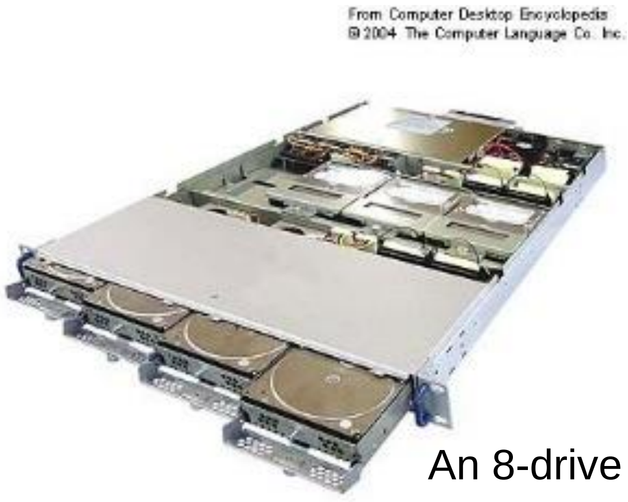


Intel's Skylake processor die layout

Storage

- Storage necessary for operating system, software and processing usually comes as disks close to CPUs
  - Diskless servers are also possible, though rare
- For permanent storage, dedicated disk servers are manufactured
  - Computing servers with very large storage capacity (dozens of Terabytes), optimized for fast access and back-up: low latency or on-line storage
- For archival, tape servers are used
  - Slow to access: serial read, require the tape to be fetched and inserted into the reading device: high latency or off-line storage

Storage servers



An 8-drive rack unit

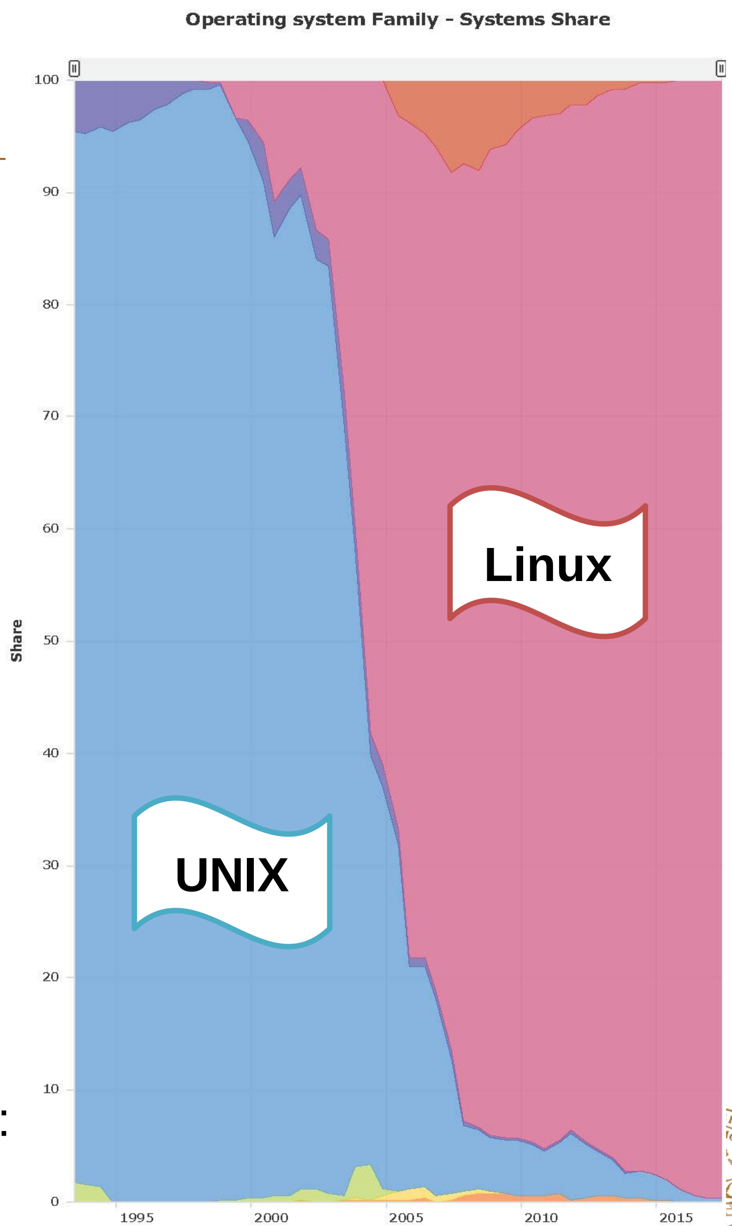
Tape robot at FNAL

A disk storage rack fragment

Operating systems (OS)

- On PCs, Microsoft Windows and MacOS dominate
  - For scientific computing, Linux and sometimes MacOS are used as well
- On clusters and supercomputers, Linux is by far dominant
  - Comes in many flavors
  - Often – RedHat Linux or its derivatives

Graph from Top 500:



Virtualization and Clouds

- Modern processors and operating systems allow full emulation of a computer
  - Such emulation is called virtualization
  - Everything is virtualized: CPU, network cards, disk partitions etc
  - Practical use: if your program works in one OS, and your PC uses another, you can simply emulate the computer with the necessary OS
    - » System to emulate is encapsulated in virtual images
    - » One real machine can host several virtual ones
- One can rent a virtual PC or even a virtual cluster from Cloud providers
  - Cloud servers are very large clusters, optimized to host virtual machines
  - Other Cloud services also exist: storage, databases etc

Scientific computing

WORKFLOWS, SECURITY

Scientific computing scenarios

A Infrequent tasks with moderate resource consumption

- E.g. Excel macros, simple image processing etc

B Large batches of similar (simple) independent tasks: <u>serial</u> jobs

- Processing of many images, analysis of accelerator collision events etc

C Lengthy resource-consuming tasks: often <u>parallel</u> jobs

- Pattern recognition, complex system simulation, parameter scanning, lattice QCD etc
  - Parallel jobs share memory, or exchange information by other means

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

- 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 *Grid*

Generic service for rent – Clouds

- *Cloud* typically refers to systems of virtual machines
- 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”
- Usually, no high performance
  - Unless you pay

LUND UNIVERSITY

Computer security considerations

Using info by CERN Security Officer

- Don’t think nobody is interested in your sad computer!
- Interpol report from 2012:
  “*Organised international gangs are behind most internet scams and that cyber crime’s estimated cost is more than that of cocaine, heroin and marijuana trafficking put together*”
- Don’t worry about hacker kids: 80% of crime committed online is now connected to organised gangs operating across borders
  - State-sponsored, not for money, sophisticated
  - For-profit organisations, usually up for big cash
  - Hacktivists, sabotage seeking publicity

Dangers for research organisations

- Financial: fake invoices, fraudulent transfers
- Reputation or legal impact: leaking confidential research documentation, proprietary technologies, tenders
- Personal information: collecting medical, travel details, personal contacts for social engineering
- Infrastructure damage: data loss, access to equipment control, blackouts

LUND UNIVERSITY

Protecting ourselves, our tools and our results

- Be extra vigilant with links and attachments in e-mails
- Use Chrome or Firefox (better on Linux)
  - Do not install fancy extensions, player add-ons, bars etc
- Always install security updates
  - Use a good antivirus
  - Install software from vendors themselves or authorised stores, not from aggregators
- And what about passwords?.. This is what the rest of the lecture is about!

LUND UNIVERSITY

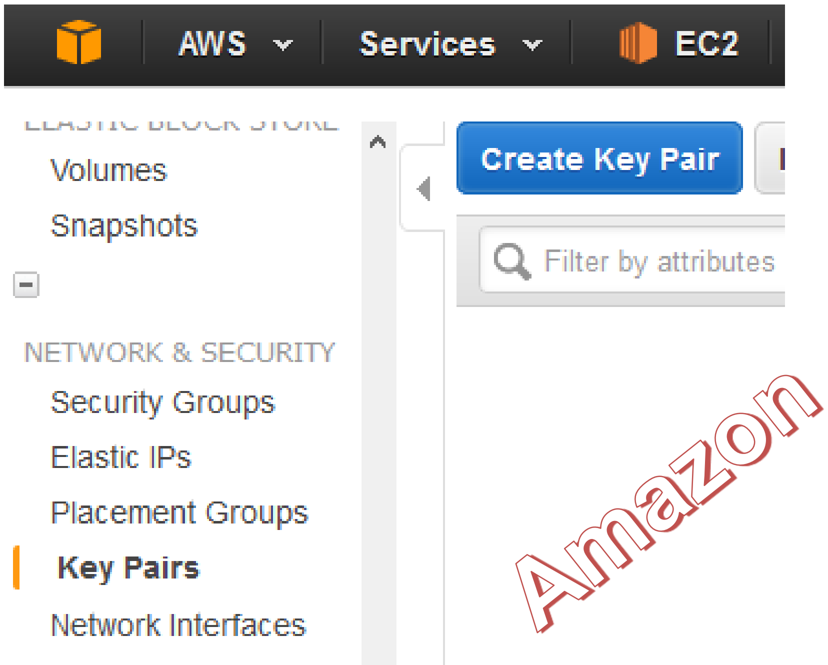
To access computers, storage, or a cloud, you need permission

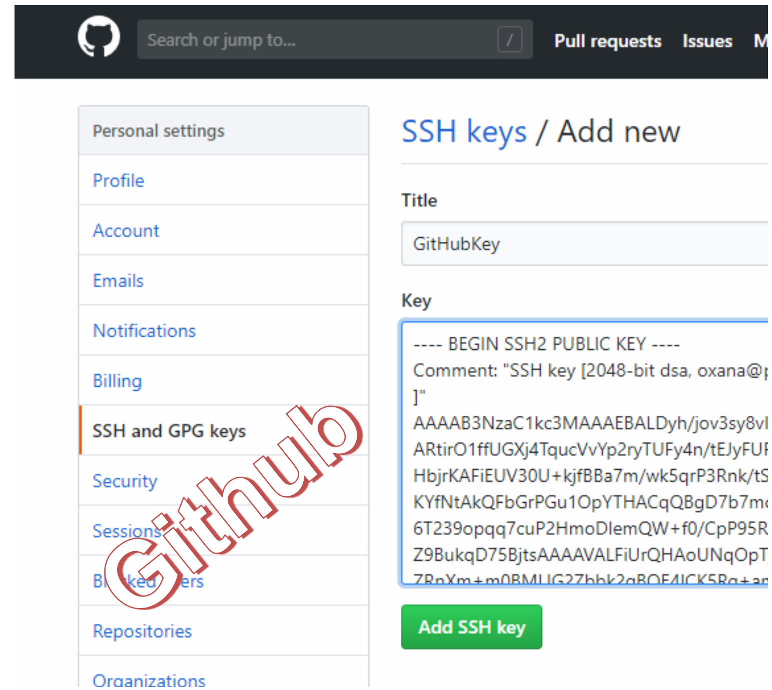
- To access one computer (or one cluster) you need a password
  - You also have a personal user space (account)
- Now scale it up 100+ computers, clusters, clouds, and 1000+ users
  - You can't quite remember 100+ passwords
  - Sysadmins can't quite manage 1000+ user accounts

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

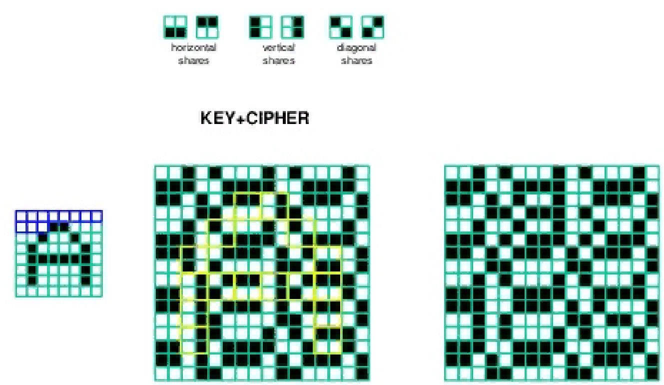
LUND UNIVERSITY

Many cloud services and clusters use SSH key pairs





- Secure shell (SSH) is when your bash (tcsh, zsh etc) session is on a remote machine and you work through an **encrypted connection**
- To encrypt anything, you need **encryption keys**

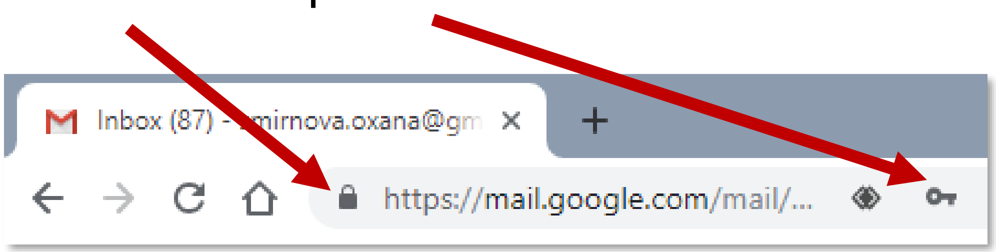


LUND UNIVERSITY

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, Web service, database service, bank service etc
    - Sometimes you need services to act on your behalf: delegate your rights to them
      - For example, if your job needs to access a password-protected database

- All secure Web sites are protected by PKI



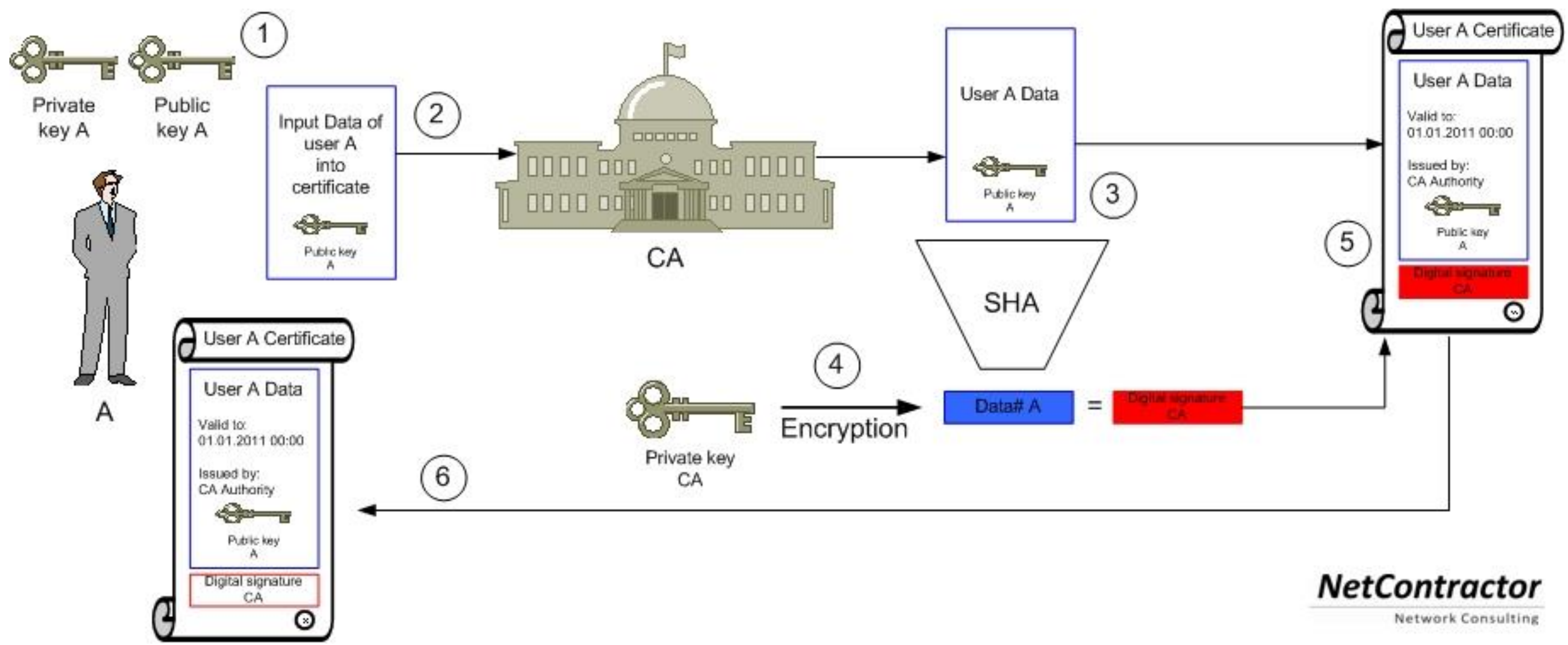
Principles of PKI

- 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

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

- 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
  - 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 (exported as .p12 files)
    - » **File exported from browser must not travel!**

LUND UNIVERSITY

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

LUND UNIVERSITY

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

LUND UNIVERSITY

Mutual authentication

- **Authentication** is establishing validity of person's (or service) identity
  - Not to be confused with <u>authorisation</u>: 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 <u>breaks everything</u>
- 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 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

↓

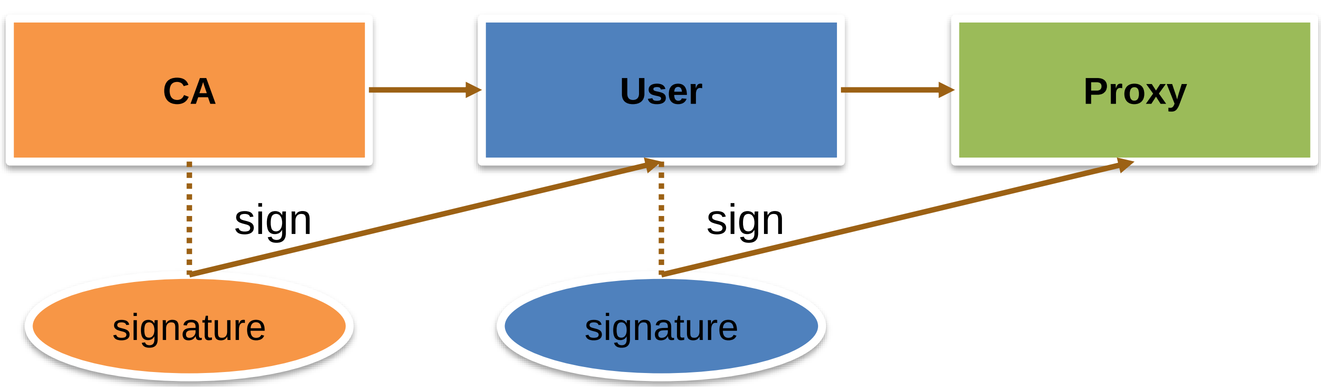
So each server needs a document from a user, delegating access rights

LUND UNIVERSITY

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
- 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 X509 proxy created?

- A **new** private/public key pair is created for each proxy
  - When a proxy expires, a new one must be created to continue working
    - » Default expiration time is 24 hours
- A proxy is then constructed of:
  1. Public certificate (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. Private key
  3. Optionally, Attribute Certificates – 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

LUND UNIVERSITY

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 <u>delegate authorisation</u>
  - Instead of using proxies, it uses tokens


```
{
  "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

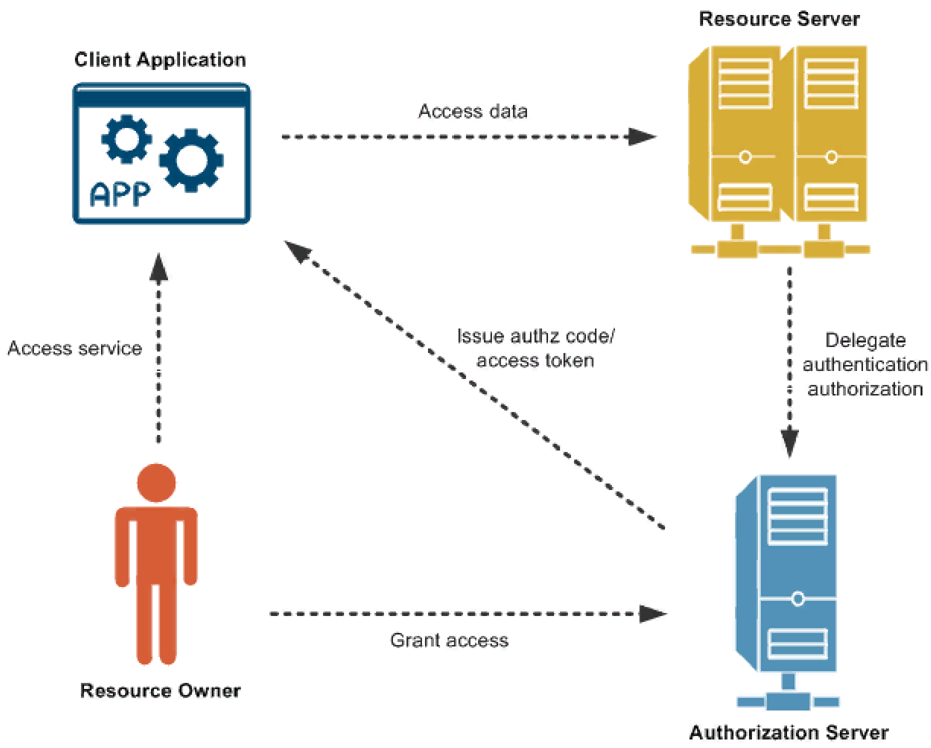


Diagram by ORACLE

- 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
    - » 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

…more on distributed computing next time!

Homework

- Get yourself a personal X509 certificate signed by LU
- Hints:
  - Google for “lund digicert certifikat”
  - Read instructions for personal certificate
  - Request “Grid Premium” certificate
  - Export the certificate from the browser as a .p12 file
- If you have access to Linux, extract private and public keys:

Private key:

```
openssl pkcs12 -nocerts -in mycert.p12 -out userkey.pem
```

Public key:

```
openssl pkcs12 -clcerts -nokeys -in mycert.p12 -out usercert.pem
```


LUND UNIVERSITY