CERN School of Computing

Data Technologies

Alberto Pace
alberto.pace@cern.ch
CERN Data and Storage Services Group

Agenda

- **Introduction to data management**
  - Data Workflows in scientific computing
  - Storage Models
- **Data management components**
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- **Summary**

1st lecture

2nd lecture

3rd lecture

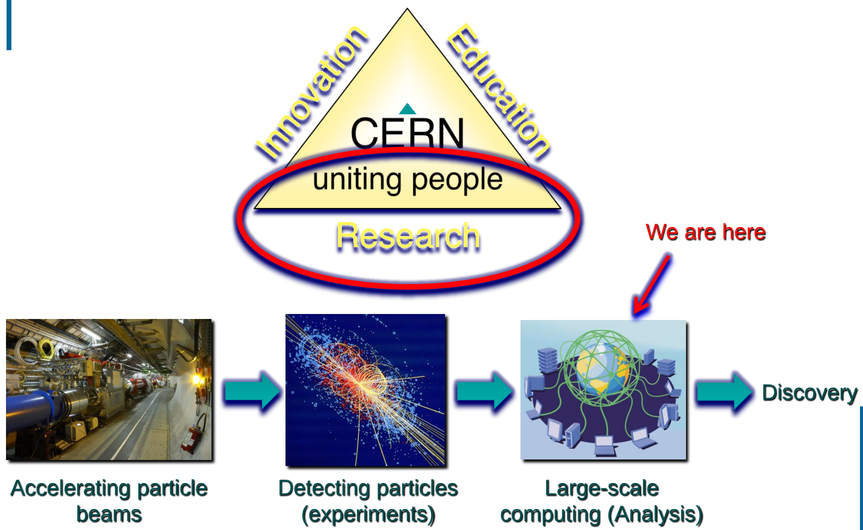
4th lecture

5th lecture

CERN School of Computing

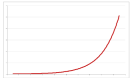
Introduction to data management

The mission of CERN



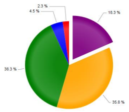
The need for computing in research

- **Scientific research in recent years has exploded the computing requirements**
  - Computing has been the strategy to reduce the cost of traditional research



At constant cost, exponential growth of performances

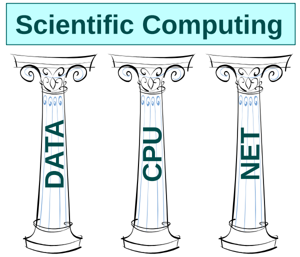
  - Computing has opened new horizons of research not only in High Energy Physics



Return in computing investment higher than other fields: Budget available for computing increased, growth is more than exponential

The need for storage in computing

- **Scientific computing for large experiments is typically based on a distributed infrastructure**
- **Storage is one of the main pillars**
- **Storage requires Data Management…**

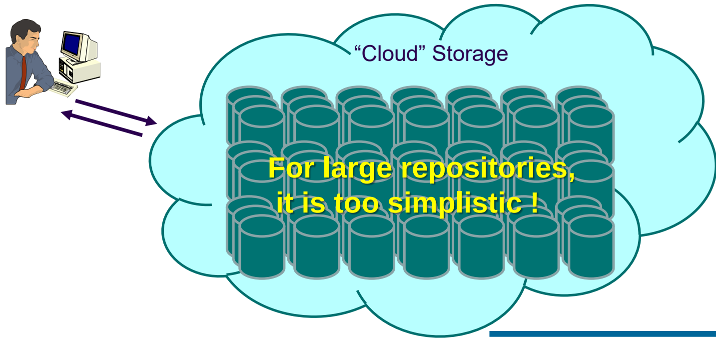


"Why" data management ?

- **Data Management solves the following problems**
  - Data reliability
  - Access control
  - Data distribution
  - Data archives, history, long term preservation
  - In general:
    - Empower the implementation of a workflow for data processing

Can we make it simple ?

- **A simple storage model: all data into the same container**
  - Uniform, simple, easy to manage, no need to move data
  - Can provide sufficient level of performance and reliability

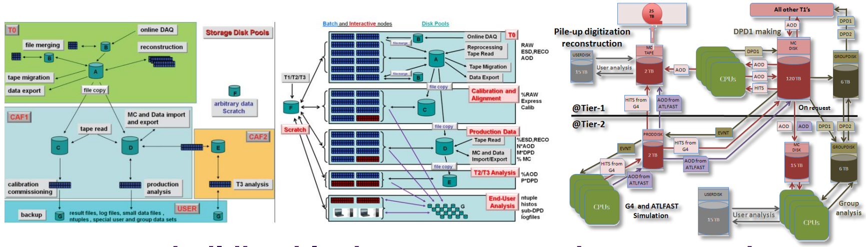


Why multiple pools and quality ?

- **Derived data used for analysis and accessed by thousands of nodes**
  - Need high performance, Low cost, minimal reliability (derived data can be recalculated)
- **Raw data that need to be analyzed**
  - Need high performance, High reliability, can be expensive (small sizes)
- **Raw data that has been analyzed and archived**
  - Must be low cost (huge volumes), High reliability (must be preserved), performance not necessary

So, … what is data management ?

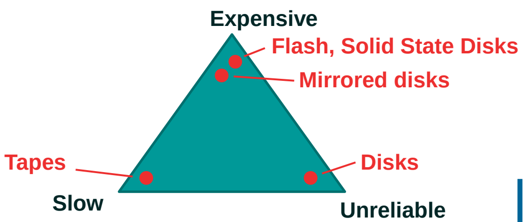
- **Examples from LHC experiment data models**



- **Two building blocks to empower data processing**
  - Data pools with different quality of services
  - Tools for data transfer between pools

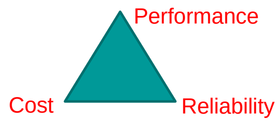
Data pools

- **Different quality of services**
  - Three parameters: (Performance, Reliability, Cost)
  - You can have two but not three

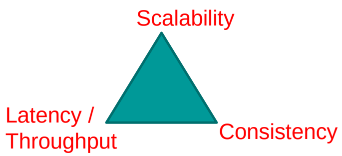


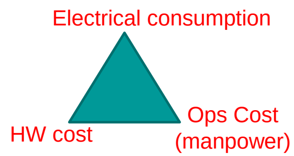
But the balance is not as simple

- **Many ways to split (performance, reliability, cost)**



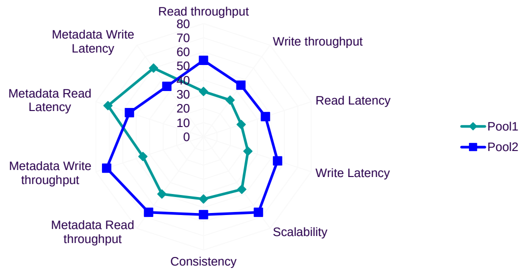
- **Performance has many sub-parameters**
- **Cost has many sub-parameters**
- **Reliability has many sub-parameters**





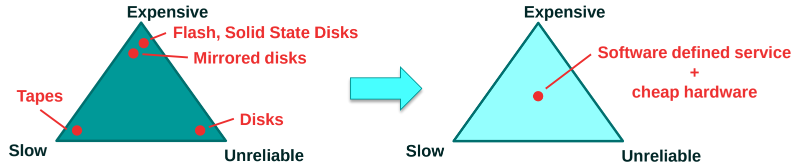
And reality is complicated

- Key requirements: Simple, Scalable, Consistent, Reliable, Available, Manageable, Flexible, Performing, Cheap, Secure.
- Aiming for “à la carte” services (storage pools) with on-demand “quality of service”
- And where is scalability ?



Where are we heading ?

- Software solutions + Cheap hardware



CERN School of Computing

Data Management Components

Agenda

Name Server

- **The name server is "the" database of a managed storage system which contains the catalogue of all data (typically all files)**
- **It is a simple lookup-based, single-key, database application for which several implementation exists**
  - DNS (domain name server) software
  - LDAP databases
  - Hash tables / Object databases
  - Relational Databases
- **Name server reliability is critical**
  - Name server failure brings down the whole storage system
- **Name server performance is critical**
  - See next slide …

Criticality of the name server performance

- **Every meta-data operation requires a database transaction.**
- **It is essential to understand where the "name server" approach is placed ...**
  - The name server lookup time dictates the performance of the whole storage system
  - The database becomes the bottleneck of the entire storage process: low performances are a symptom of major architectural mismatch
  - *Comment: Cloud storage ? An architecture that replaces the name server DB lookup with a "calculated" name resolution (… more to come …)*

Short digression on ...
Uniform Resource Identifiers (URI)

- **Example from the web ...**
  - http://csc.cern.ch/data/2012/School/page.htm

protocol   **host / domain**   **volume**   **folder / directory**   **file**

- **Where is the database lookup when accessing a web page ?**
  - **at the host / domain level.**
  - **Every host has its own namespace, managed locally.**
- **Excellent example of "federated" namespace**
  - **Extremely efficient, but some limitations**

http://www.ietf.org/rfc/rfc2396.txt

Similar problem in storage systems

- **Example from storage ...**
  - storage://cern.ch/data/2012/School/page.htm

protocol   **host / domain**   **volume**   **folder / directory**   **file**

- **In several implementation, the database lookup is placed at the "file" level**
  - Impacts all operations, including most popular open() and stat()
- **Great flexibility but huge performance hit, which implies more hardware and constant database tuning**

Between extremes…

- **There are intermediate solutions between**
  - File systems (no database)
  - Storage systems with database lookup placed the “file” level
- **Examples of high performance scalable solutions**
  - AFS / DFS
    - Database placed at the domain / host level (same as the web)
    - Very scalable
    - But within the domain (eg: “cern.ch”), identical to a file system with physical files directly mapped to logical filenames
  - XROOTD / Scalla / NFS / clustered storage / cloud storage
    - Database (somehow) placed at the volume level (this is a simplified statement)
    - Similar scalability, with more flexibility in terms of data management with physical files “calculated” from logical filenames not requiring database lookup below the “volume” level
    - Federated storage

Agenda

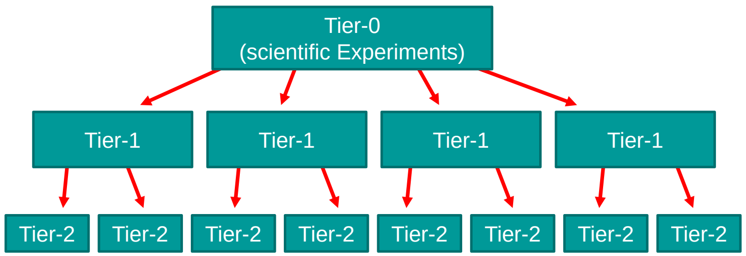
- Introduction to data management
  - Data Workflows in scientific computing
  - Storage Models
- Data management components
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- Summary

Access protocols

- **File-level (posix) access is the starting point**
  - Open, Stat, Read, Write, Delete, ...
- **Several extensions that are “implementation specific” which cannot be mapped to posix calls**
  - Pre-staging data from slow storage into fast storage
  - Managing pool creation / sizes
  - Reading or changing access permissions
  - Interpretation of extended data attributes and meta data
- **Some parts of the posix standard are not scalable (ls, chdir, …)**
  - Not all storage systems implements posix entirely
- **Various protocols offering file access**
  - rfio (Castor, DPM, ...), dcap (dCache), xroot (Scalla, Castor, DPM, EOS ...), NFS, AFS, S3, …
- **Various protocols for bulk data movements over wide area networks**
  - GridFTP, ...

Dataflows across sites

- **Storage in scientific computing is distributed across multiple data centres**
- **Data flows from the experiments to all datacenters where there is CPU available to process the data**



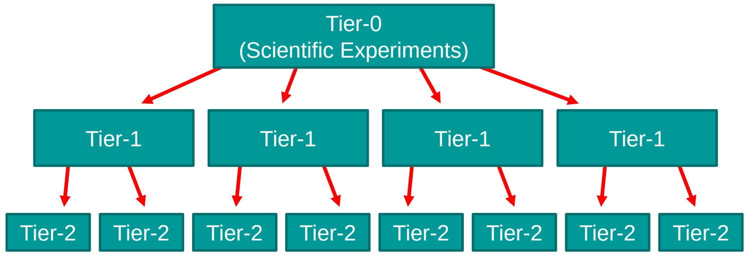
Efficiency

- **A key parameter in distributed scientific computing is the efficiency**
  - High efficiency requires to have the CPUs colocated with the data to analyze, using the network.
- **Whenever a site has …**
  - idle CPUs (because no data is available to process)
  - or excess of Data (because there is no CPU left for analysis)
  - Idle or saturated networks
- ***… the efficiency drops***

Data Technologies – CERN School of Computing 2019

Data distribution

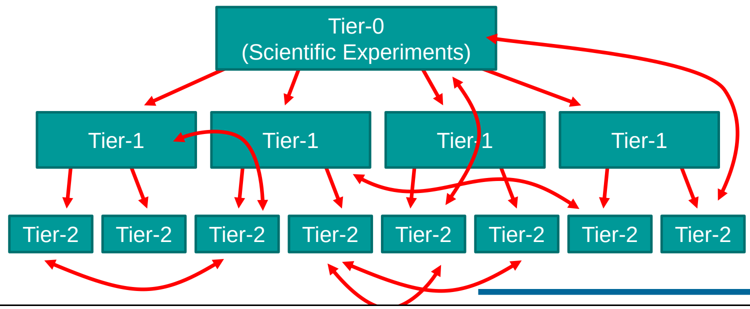
- **Analysis made with high efficiency requires the data to be pre-located to where the CPUs are available**



Data Technologies – CERN School of Computing 2019

Data distribution

- **Analysis made with high efficiency requires the data to be pre-located to where the CPUs are available**
- **Or to allow peer-to peer data transfer**
  - This allows sites with excess of CPU, to schedule the pre-fetching of data when missing locally or to access it remotely if the analysis application has been designed to cope with high latency



Data Technologies – CERN School of Computing 2019

Data distribution

- **Both approaches coexists in High Energy Physics**
- **Data is pre-placed**
  - This is the role of the experiments that plans the analysis
- **Data is globally accessible and federated in a global namespace**
  - The middleware always attempt to take the local data and uses an access protocol that redirects to the nearest remote copy when the local data is not available
  - All middleware and jobs are designed to minimize the impact of the additional latency that the redirection requires
- **Using access protocols that allows global data federation is essential**
  - http, xroot

Agenda

- Introduction to data management
  - Data Workflows in scientific computing
  - Storage Models
- Data management components
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- Summary

Data Technologies – CERN School of Computing 2019

Storage Reliability

- **Reliability is related to the probability to lose data**
  - Def: “the probability that a storage device will perform an arbitrarily large number of I/O operations without data loss during a specified period of time”
- **Reliability of the “service” depends on the environment (energy, cooling, people, ...)**
  - Will not discuss this further
- **Reliability of the “service” starts from the reliability of the underlying hardware**
  - Example of disk servers with simple disks: reliability of service = reliability of disks
- **But data management solutions can increase the reliability of the hardware at the expenses of performance and/or additional hardware / software**
  - Disk Mirroring
  - Redundant Array of Inexpensive Disks (RAID)

Data Technologies – CERN School of Computing 2019

Hardware reliability

- **Do we need tapes ?**
- **Tapes have a bad reputation in some use cases**
  - Slow in random access mode
    - high latency in mounting process and when seeking data (F-FWD, REW)
  - Inefficient for small files (in some cases)
  - Comparable cost per (peta)byte as hard disks

- **Tapes have also some advantages**
  - Fast in sequential access mode
    - > 2x faster than disk, with physical read after write verification
  - Several orders of magnitude more reliable than disks
    - Few hundreds GB loss per year on 80 PB tape repository
    - Few hundreds TB loss per year on 50 PB disk repository
  - No power required to preserve the data
  - Less physical volume required per (peta)byte
  - Inefficiency for small files issue resolved by recent developments
  - Nobody can delete hundreds of PB in minutes
- **Bottom line: if not used for random access, tapes have a clear role in the architecture**

Data Technologies – CERN School of Computing 2019

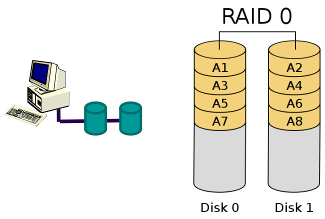
Reminder: types of RAID

- **RAID0**
  - Disk striping
- **RAID1**
  - Disk mirroring
- **RAID5**
  - Parity information is distributed across all disks
- **RAID6**
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss

http://en.wikipedia.org/wiki/RAID

Reminder: types of RAID

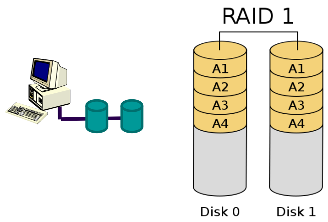
- **RAID0**
  - Disk striping
- **RAID1**
  - Disk mirroring
- **RAID5**
  - Parity information is distributed across all disks
- **RAID6**
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss



http://en.wikipedia.org/wiki/RAID

Reminder: types of RAID

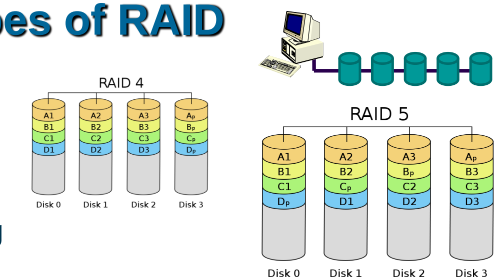
- **RAID0**
  - Disk striping
- **RAID1**
  - Disk mirroring
- **RAID5**
  - Parity information is distributed across all disks
- **RAID6**
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss



http://en.wikipedia.org/wiki/RAID

Reminder: types of RAID

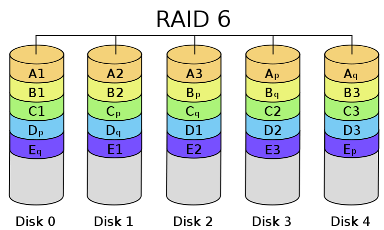
- **RAID0**
  - Disk striping
- **RAID1**
  - Disk mirroring
- **RAID5**
  - Parity information is distributed across all disks
- **RAID6**
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss



http://en.wikipedia.org/wiki/RAID

Reminder: types of RAID

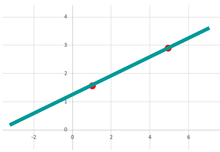
- **RAID0**
  - Disk striping
- **RAID1**
  - Disk mirroring
- **RAID5**
  - Parity information is distributed across all disks
- **RAID6**
  - Uses Reed–Solomon error correction, allowing the loss of 2 disks in the array without data loss



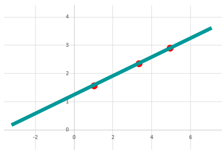
http://en.wikipedia.org/wiki/RAID

Understanding error correction

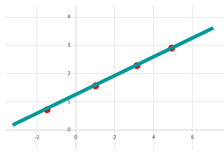
- **A line is defined by 2 numbers: a, b**
  - (a, b) is the information
  - y = ax + b
- **Instead of transmitting a and b, transmit some points on the line at known abscissa. 2 points define a line. If I transmit more points, these should be aligned.**



2 points



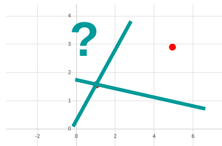
3 points



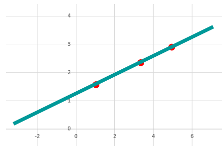
4 points

If we lose some information …

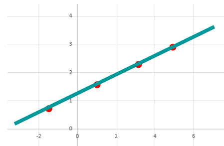
- **If we transmit more than 2 points, we can lose any point, provided the total number of point left is >= 2**



1 point instead of 2
information lost



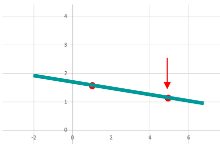
2 points instead of 3



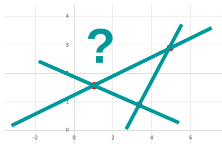
2 or 3 points instead of 4

If we have an error …

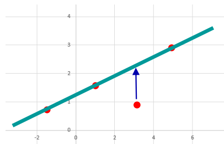
- **If there is an error, I can detect it if I have transmitted more than 2 points, and correct it if have transmitted more than 3 points**



Information lost
(and you do not notice)



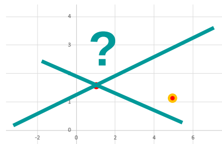
Error detection
Information is lost
(and you notice)



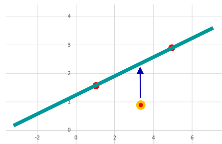
Error correction
Information is recovered

If you have checksumming on data …

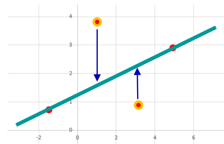
- **You can detect errors by verifying the consistency of the data with the respective checksums. So you can detect errors independently.**
- **… and use all redundancy for error correction**



Information lost
(and you notice)



Error correction
Information is recovered



2 Error corrections possible
Information is recovered

Reed–Solomon error correction …

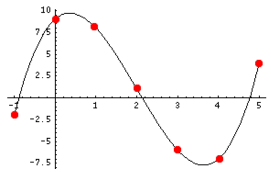
- .. is an error-correcting code that works by oversampling (by n points) a polynomial constructed from the data
- Any m distinct points uniquely determine a polynomial of degree, at most, m − 1
- The sender determines the polynomial (of degree m − 1), that represents the m data points. The polynomial is "encoded" by its evaluation at n+m points. If during transmission, the number of corrupted values is < n the receiver can recover the original polynomial.
- Implementation examples:
  - n = 0 no redundancy
  - n = 1 is Raid 5 (parity)
  - n = 2 is Raid 6 (Reed Solomon, double / diagonal parity)
  - n = 3 is … (Triple parity)

http://en.wikipedia.org/wiki/Reed-Solomon

Reed–Solomon (simplified) Example

- 4 Numbers to encode: { 1, -6, 4, 9 }   (m=4)
- polynomial of degree 3 (m − 1):

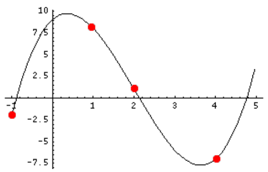
$y = x^3 - 6x^2 + 4x + 9$



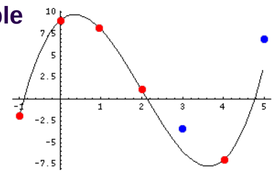
- We encode the polynomial with n + m = 7 points
  { -2, 9, 8, 1, -6, -7, 4 }

Reed–Solomon (simplified) Example

- To reconstruct the polynomial, any 4 points are enough: we can lose any 3 points.



- We can have an error on any 2 points that can be corrected: We need to identify the 5 points “aligned” on the only one polynomial of degree 3 possible



http://kernel.org/pub/linux/kernel/people/hpa/raid6.pdf

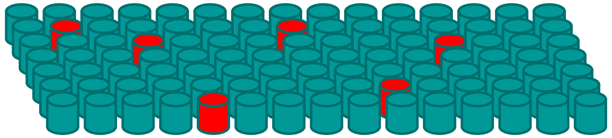
Error detection vs Error correction

- With Reed-Solomon:
  - If the number of corrupted values is **=** n we can **only detect** the error
  - If the number of corrupted values is **<** n we can **correct** the error
- However, by adding a checksum or hash on each point, we can individually identify the corrupted values
  - If checksum has been added, Reed-Solomon can **correct** corrupted values **≤** n

Reliability calculations

- **With RAID, the final reliability depends on several parameters**
  - The reliability of the hardware
  - The type of RAID
  - The number of disks in the set
- **Already this gives lot of flexibility in implementing arbitrary reliability**

Raid 5 reliability



- Disk are regrouped in sets of equal size. If c is the capacity of the disk and n is the number of disks, the sets will have a capacity of

c (n-1)

example: 6 disks of 1TB can be aggregated to a “reliable” set of 5TB

- The set is immune to the loss of 1 disk in the set. The loss of 2 disks implies the loss of the entire set content.

Some calculations for Raid 5

- **Disks MTBF is between 3×10^5 and 1.2×10^6 hours**
- **Replacement time of a failed disk is < 4 hours**
- **Probability of 1 disk to fail within the next 4 hours**

$$P_f = \frac{Hours}{MTBF} = \frac{4}{3\times10^5} = 1.3\times10^{-5}$$

Some calculations for Raid 5

- **Disks MTBF is between 3×10^5 and 1.2×10^6 hours**
- **Replacement time of a failed disk is < 4 hours**
- **Probability of 1 disk to fail within the next 4 hours**

$$P_f = \frac{Hours}{MTBF} = \frac{4}{3\times10^5} = 1.3\times10^{-5}$$

- **Probability to have a failing disk in the next 4 hours in a 15 PB computer centre (15’000 disks)**

$$P_{f15000} = 1-(1-P_f)^{15000} = 0.18$$

Some calculations for Raid 5

- Disks MTBF is between 3 x 10^5 and 1.2 x 10^6 hours
- Replacement time of a failed disk is < 4 hours
- Probability of 1 disk to fail within the next 4 hours

p(A and B) = p(A) * p(B/A)

if A,B independent : p(A) * p(B)

$$P_f = \frac{Hours}{MTBF} = \frac{4}{3\times10^5} = 1.3\times10^{-5}$$

- Probability to have a failing disk in the next 4 hours in a 15 PB computer centre (15'000 disks)

$$P_{f15000} = 1-(1-P_f)^{15000} = 0.18$$

- Imagine a Raid set of 10 disks. Probability to have one of the remaining disk failing within 4 hours

$$P_{f9} = 1-(1-P_f)^9 = 1.2\times10^{-4}$$

Some calculations for Raid 5

- Disks MTBF is between 3 x 10^5 and 1.2 x 10^6 hours
- Replacement time of a failed disk is < 4 hours
- Probability of 1 disk to fail within the next 4 hours

p(A and B) = p(A) * p(B/A)

if A,B independent : p(A) * p(B)

$$P_f = \frac{Hours}{MTBF} = \frac{4}{3\times10^5} = 1.3\times10^{-5}$$

- Probability to have a failing disk in the next 4 hours in a 15 PB computer centre (15'000 disks)

$$P_{f15000} = 1-(1-P_f)^{15000} = 0.18$$

- Imagine a Raid set of 10 disks. Probability to have one of the remaining disk failing within 4 hours

$$P_{f9} = 1-(1-P_f)^9 = 1.2\times10^{-4}$$

- However the second failure may not be independent from the first one. There is no way to calculate this probability ! We can arbitrarily increase it by two orders of magnitude to account the dependencies (over temperature, high noise, EMP, high voltage, faulty common controller, ...)

$$P_{f9corrected} = 1-(1-P_f)^{900} = 0.0119$$

Some calculations for Raid 5

- Disks MTBF is between 3 x 10^5 and 1.2 x 10^6 hours
- Replacement time of a failed disk is < 4 hours
- Probability of 1 disk to fail within the next 4 hours

p(A and B) = p(A) * p(B/A)

if A,B independent : p(A) * p(B)

$$P_f = \frac{Hours}{MTBF} = \frac{4}{3\times10^5} = 1.3\times10^{-5}$$

- Probability to have a failing disk in the next 4 hours in a 15 PB computer centre (15'000 disks)

$$P_{f15000} = 1-(1-P_f)^{15000} = 0.18$$

- Imagine a Raid set of 10 disks. Probability to have one of the remaining disk failing within 4 hours

$$P_{f9} = 1-(1-P_f)^9 = 1.2\times10^{-4}$$

- However the second failure may not be independent from the first one. There is no way to calculate this probability ! We can arbitrarily increase it by two orders of magnitude to account the dependencies (over temperature, high noise, EMP, high voltage, faulty common controller, ...)

$$P_{f9corrected} = 1-(1-P_f)^{900} = 0.0119$$

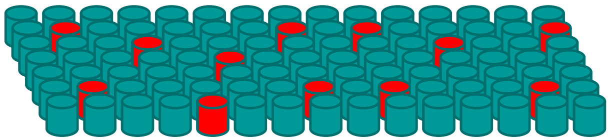
- Probability to lose computer centre data in the next 4 hours

$$P_{loss} = P_{f15000} \times P_{f9corrected} = 6.16\times10^{-4}$$

- Probability to lose data in the next 10 years

$$P_{loss10yrs} = 1-(1-P_{loss})^{10\times365\times6} = 1\text{-}10^{-21} \cong 1$$

Raid 6 reliability



- Disk are regrouped in sets of arbitrary size. If c is the capacity of the disk and n is the number of disks, the sets will have a capacity of

c (n-2)

example: 12 disks of 1TB can be aggregated to a “reliable” set of 10TB

- The set is immune to the loss of 2 disks in the set. The loss of 3 disks implies the loss of the entire set content.

Same calculations for Raid 6

- **Probability of 1 disk to fail within the next 4 hours**

$$P_f = \frac{Hours}{MTBF} = \frac{4}{3\times10^5} = 1.3\times10^{-5}$$

- **Imagine a raid set of 10 disks. Probability to have one of the remaining 9 disks failing within 4 hours (increased by two orders of magnitudes)**

$$P_{f9} = 1-(1-P_f)^{900} = 1.19\times10^{-2}$$

- **Probability to have another of the remaining 8 disks failing within 4 hours (also increased by two orders of magnitudes)**

$$P_{f8} = 1-(1-P_f)^{800} = 1.06\times10^{-2}$$

- **Probability to lose data in the next 4 hours**

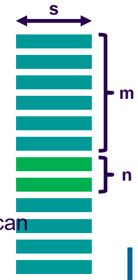
$$P_{loss} = P_{f15000}\times P_{f99}\times P_{f98} = 2.29\times10^{-5}$$

- **Probability to lose data in the next 10 years**

$$P_{loss10yrs} = 1-(1-P_{loss})^{10\times365\times6} = 0.394$$

Arbitrary reliability

- **RAID is “disks” based. This lacks of granularity**
- **For increased flexibility, an alternative would be to use files ... but files do not have constant size**
- **File “chunks” (or “blocks”) is the solution**
  - Split files in chunks of size “s”
  - Group them in sets of “m” chunks
  - For each group of “m” chunks, generate “n” additional chunks so that
    - For any set of “m” chunks chosen among the “m+n” you can reconstruct the missing “n” chunks
  - Scatter the “m+n” chunks on <u>independent</u> storage



Arbitrary reliability with the “chunk” based solution

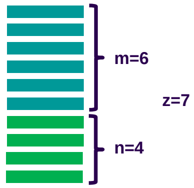
- **The reliability is independent form the size “s” which is arbitrary.**
  - Note: both large and small “s” impact performance
- **Whatever the reliability of the hardware is, the system is immune to the loss of “n” simultaneous failures from pools of “m+n” storage chunks**
  - Both “m” and “n” are arbitrary. Therefore arbitrary reliability can be achieved
- **The fraction of raw storage space loss is n / (n + m)**
- **Note that space loss can also be reduced arbitrarily by increasing m**
  - At the cost of increasing the amount of data loss if this would ever happen

Analogy with the gambling world

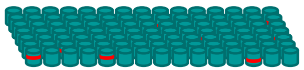
- We just demonstrated that you can achieve “arbitrary reliability” at the cost of an “arbitrary low” amount of disk space. This is possible because you increase the amount of data you accept loosing when this rare event happens.
- In the gambling world there are several playing schemes that allows you to win an arbitrary amount of money with an arbitrary probability.
- Example: you can easily win 100 Euros at > 99 % probability ...
  - By playing up to 7 times on the “Red” of a French Roulette and doubling the bet until you win.
  - The probability of not having a “Red” for 7 times is $(19/37)^7$ = 0.0094)
  - You just need to take the risk of loosing 12’700 euros with a 0.94 % probability

Amount		Win		Lost	
Bet	Cumulated	Probability	Amount	Probability	Amount
100	100	48.65%	100	51.35%	100
200	300	73.63%	100	26.37%	300
400	700	86.46%	100	13.54%	700
800	1500	93.05%	100	6.95%	1500
1600	3100	96.43%	100	3.57%	3100
3200	6300	98.17%	100	1.83%	6300
6400	12700	99.06%	100	0.94%	12700

Practical comments



- **n can be …**
  - 1 = Parity
  - 2 = Parity + Reed-Solomon, double parity
  - 3 = Reed Solomon, ZFS triple parity
- **m chunks of any (m + n) sets are enough to obtain the information. Must be saved on independent media**
  - Performance can depend on m (and thus on s, the size of the chunks): The larger m is, the more the reading can be parallelized
  - Until the client bandwidth is reached
- **For n > 2 Reed Solomon has a computational impact affecting performances**
  - Alternate encoding algorithms are available requiring z chunks to reconstruct the data, being m < z < m+n (see example later on with LDPC).
  - These guarantees high performance at the expenses of additional storage. When m=z we fall back in the "optimal" storage scenario



 http://blogs.sun.com/ahl/entry/triple_parity_raid_z

Data Technologies – CERN School of Computing 2019

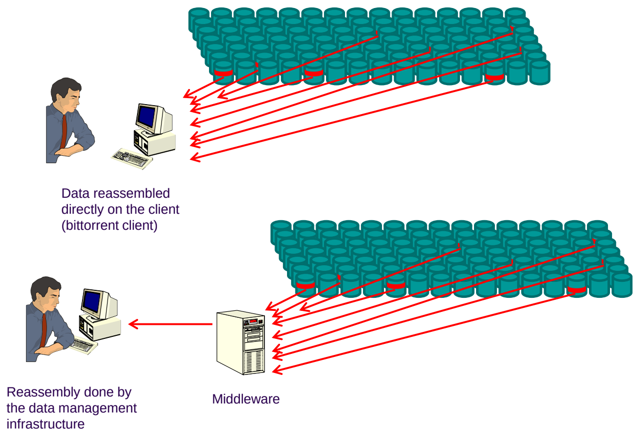
Chunk transfers

- **Among many protocols, Bittorrent is the most popular**
- **An SHA1 hash (160 bit digest) is created for each chunk**
- **All digests are assembled in a "torrent file" with all relevant metadata information**
- **Torrent files are published and registered with a tracker which maintains lists of the clients currently sharing the torrent's chunks**
- **In particular, torrent files have:**
  - an "announce" section, which specifies the URL of the tracker
  - an "info" section, containing (suggested) names for the files, their lengths, the list of SHA-1 digests
- **Reminder: it is the client's duty to reassemble the initial file and therefore it is the client that always verifies the integrity of the data received**

http://en.wikipedia.org/wiki/BitTorrent_(protocol)

Data Technologies – CERN School of Computing 2019

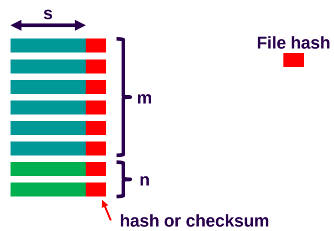
Reassembling the chunks



Data Technologies – CERN School of Computing 2019

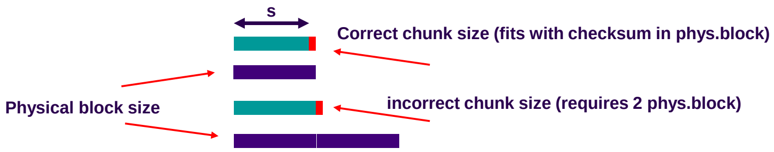
Ensure integrity, identify corruptions

- **You must be able to identify broken files**
  - A hash is required for every file.
- **You must be able to identify broken chunks**
  - A hash for every chunk (example SHA1 160 bit digest) guarantees chunks integrity.
- **It tells you the corrupted chunks and allows you to correct n errors (instead of n-1 if you would not know which chunks are corrupted)**



Chunk size and physical blocks

- **The storage overhead of the checksum is typically of few hundred bytes and can be easily neglected compared to the chunk size that is of few megabytes.**
- **To guarantee a high efficiency in transferring the chunks is essential that the sum of the chunk size with its checksum is an exact multiple or divisor of the physical block size of the storage**
- **Avoid at all cost is to choose a chunk size equal to the physical disk block size leaving no space to save the checksum in the same physical block.**



Types of arbitrary reliability (summary)

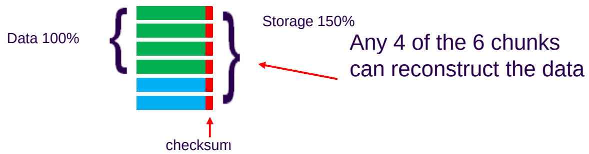
- **Plain (reliability of the service = reliability of the hardware)**

Types of arbitrary reliability (summary)

- **Plain (reliability of the service = reliability of the hardware)**
- **Replication**
  - Reliable, maximum performance, but heavy storage overhead
  - Example: 3 copies, 200% overhead

Types of arbitrary reliability (summary)

- **Double parity / Diagonal parity**
  - Example 4+2, can lose any 2, remaining 4 are enough to reconstruct, only 50 % storage overhead



Types of arbitrary reliability (summary)

- **Plain (reliability of the service = reliability of the hardware)**
- **Replication**
  - Reliable, maximum performance, but heavy storage overhead
  - Example: 3 copies, 200% overhead
- **Reed-Solomon, double, triple parity, NetRaid5, NetRaid6**
  - Maximum reliability, minimum storage overhead
  - Example 10+3, can lose any 3, remaining 10 are enough to reconstruct, only 30 % storage overhead

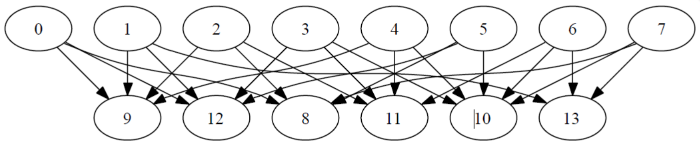
Any 10 of the 13 chunks are enough to reconstruct the data

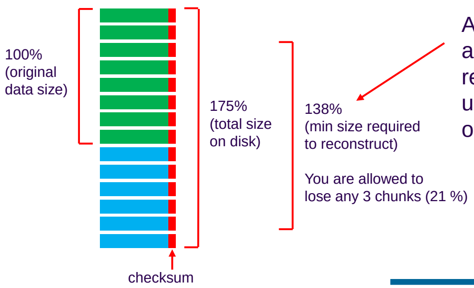
Types of arbitrary reliability (summary)

- **Plain (reliability of the service = reliability of the hardware)**
- **Replication**
  - Reliable, maximum performance, but heavy storage overhead
  - Example: 3 copies, 200% overhead
- **Reed-Solomon, double, triple parity, NetRaid5, NetRaid6**
  - Maximum reliability, minimum storage overhead
  - Example 10+3, can lose any 3, remaining 10 are enough to reconstruct, only 30 % storage overhead
- **Low Density Parity Check (LDPC) / Fountain Codes / Raptor Codes**
  - Excellent performance, more storage overhead
  - Example: 8+6, can lose any 3, remaining 11 are enough to reconstruct, 75 % storage overhead (See next slide)

Example: 8+6 LDPC

0 .. 7: original data
8 .. 13: data xor-ed following the arrows in the graph





Any 11 of the 14 chunks are enough to reconstruct the data using only XOR operations (very fast)

Types of arbitrary reliability (summary)

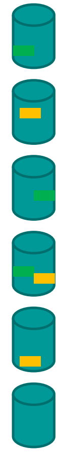
- **Plain (reliability of the service = reliability of the hardware)**
- **Replication**
  - Reliable, maximum performance, but heavy storage overhead
  - Example: 3 copies, 200% overhead
- **Reed-Solomon, double, triple parity, NetRaid5, NetRaid6**
  - Maximum reliability, minimum storage overhead
  - Example 4+2, can lose any 2, remaining 4 are enough to reconstruct, 50 % storage overhead
  - Example 10+3, can lose any 3, remaining 10 are enough to reconstruct, 30 % storage overhead
- **Low Density Parity Check (LDPC) / Fountain Codes**
  - Excellent performance, more storage overhead
  - Example: 8+6, can lose any 3, remaining 11 are enough to reconstruct, 75 % storage overhead
- **In addition to**
  - File checksums (available today)
  - Block-level checksums (available today)

Availability versus Reliability

- **Reliability relates to the probability to loose data**
  - However, you can have service interruptions and temporary unavailability of data without data loss
- **Example of a real incident:**
  - “The file system corruption of lxfsrc2206 castorcms/default affected users in several ways. At first tape recalls were attracted to the broken filesystem. Afterwards, once the machine was put in draining mode, replications from the machine were timing out. We have put machine in maintenance while it is repaired so things look OK but for 3 CANBEMIGR files in the broken filesystem that are still unavailable”
- **Consequences**
  - High data reliability does not imply high service availability
  - When hardware fails, the data on it becomes unavailable. It is a service failure
  - When experiencing service failure, an intervention must happen as fast as possible.
    - Requires a piquet with engineers on call
    - Draining and restore operations take lot of time, affecting also availability

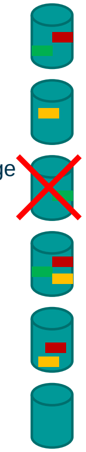
Example: High Availability with replication

- **We have “sets” of T independent storage**
  - This example has T=6
- **The storage pool is configured to replicate files R times, with R < T**
  - This example: R=3 every file is written 3 times on 3 independent storage out of the 6 available
  - When a client read a file, any copy can be used
  - Load can be spread across the multiple servers to ensure high throughput (better than mirrored disks, and much better than Raid 5 or Raid 6)



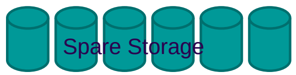
Example scenario: hardware failure

- **The loss of a storage component is detected. The storage component is disabled automatically**
- **File Read requests can continue if R>1 (at least 1 replica), at reduced throughput**
  - The example has R=3
- **File Creation / Write requests can continue**
  - New files will be written to the remaining T – 1 = 6 – 1 = 5 storage components
- **File Delete request can continue**
- **File Write / Update requests can continue**
  - Either by just modifying the remaining replicas or by creating on the fly the missing replica on another storage component
- **Service operation continues despite hardware failure. (remember: independent storage)**



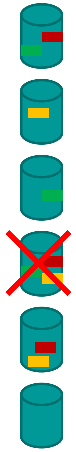
Example scenario: failure response

- **The disabled faulty storage is not used anymore**
- **There is “Spare Storage” that can be used to replace faulty storage**
  - manually or automatically
- **The lost replicas are regenerated from the existing replicas**
  - Manually or automatically



Example scenario: draining a server

- To drain a server, just power it off
- Will be seen as faulty and disabled (it will not used anymore)
- The available "Spare Storage" will be used to replace faulty storage
  - manually or automatically
- The lost replicas are regenerated from the existing replicas
  - Manually or automatically

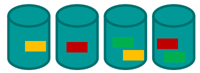


Do you really want to waste spare storage ?

- Why would you keep spare storage unused ?
- Act in advance to deal with an expected failure or replacement
  - Proactive replication of blocks/files in the spare strorage
- The number of replica (R=3) becomes a minimum.
  - Some files have R=3, others R>3
- When a server fails, some files have R=2, others will have already R≥3
  - you need to replicate only the files that have R=2

Number of Replicas

(yellow)	3	5	4	4	4	4
(red)	3	5	4	3	2	3
(green)	3	5	4	3	3	3



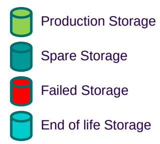
Spare Storage

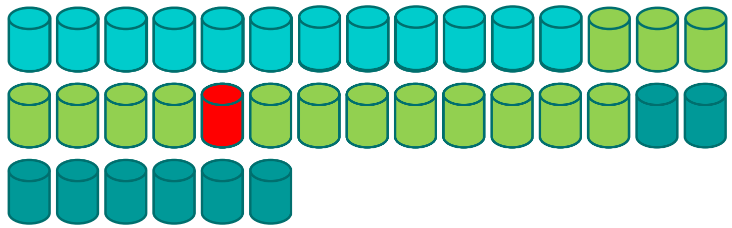
Reliability and availability calculations

- We have redundancy. Service failure, requires multiple failures
  - Prob(Service fail) = Prob (Failure 1 AND Failure 2)
- Reminder
  - P(A and B) = P(A) x P(B/A)
    [P(B/A) is higher than P(B) and difficult to calculate in the case of storage]
  - if A,B are independent events: P(A and B) = P(A) x P(B)
- We must reduce all sources of dependencies across storages: disk controller, server, network, data centre, power grid, etc …
- With storage dispersed on different servers (or data centres), we can reach independent storage (within the service responsibility) and a major reliability increase
  - Prob(Service fail) = Prob (Failure 1) x Prob (Failure 2)
- And we have a parameter R (nb of replicas) that can be adjusted arbitrarily
  - Prob(Service fail) = Prob (Storage Failure)^R
  - Example: Prob (Storage Failure) = 10^{-4}
    - Prob(Service fail)$^{R=1}$ = 10^{-4}
    - Prob(Service fail)$^{R=2}$ = 10^{-8}
    - Prob(Service fail)$^{R=3}$ = 10^{-12}
    - Prob(Service fail)$^{R=4}$ = 10^{-16}

Service operation eased …

- Production cluster, 15 Server with 9 spare
- Server Failure (servers)
- New HW delivery (6 servers)
- Out of warranty (6 servers)
- End of life





Service operations

- **Ensure that there is enough spare storage to cope with:**
  - Hardware failures
  - Planned replacement
- **No need to intervene timely when Hw fails**
  - Asynchronous interventions only
  - No engineers on call or piquet needed
- **Create and configure data pools**
- **Arbitrary level of services: Monitor reliability, availability and performance and adapt the various parameters (storage size, T, R, …) to optimize the operational experience and meet the service level agreement**
- **Identify what responses can be automated and what needs to be handled manually**

Agenda

A quick introduction to Cryptography

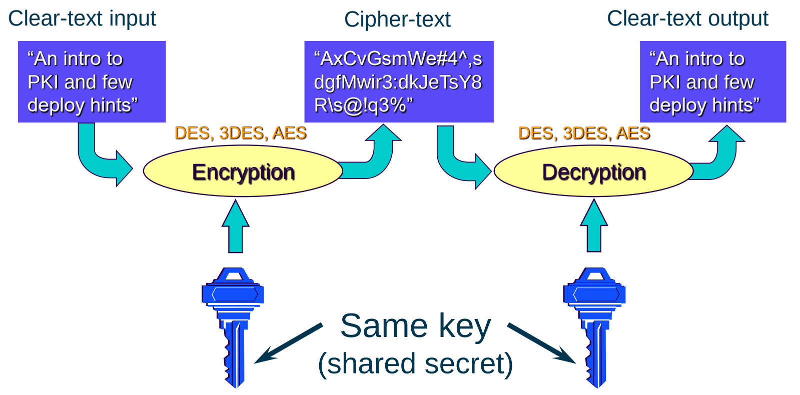
Understanding cryptography is essential for data management

It is essential to understand concepts like “Hash”, “Digital signature”, “Digitally signed data”, “Encrypted data”, “Certificate”, “Kerberos Ticket”, “Ticket Granting Ticket”, …

What does Cryptography solve?

- **Confidentiality**
  - Ensure that nobody can get knowledge of what you transfer even if listening the whole conversation
- **Integrity**
  - Ensure that message has not been modified during the transmission
- **Authenticity, Identity, Non-repudiation**
  - You can verify that you are talking to the entity you think you are talking to
  - You can verify who is the specific individual behind that entity
  - The individual behind that asset cannot deny being associated with it

Symmetric Encryption



Example: XOR function

XOR	0	1
0	0	1
1	1	0

Encryption

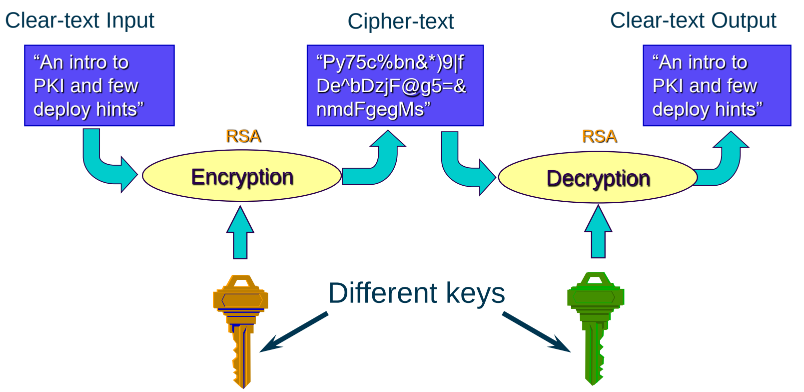
Message	1	1	0	1	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1
Key	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0
Cipher	1	0	1	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1

Decryption

cipher	1	0	1	0	1	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1
Key	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0
Message	1	1	0	1	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1

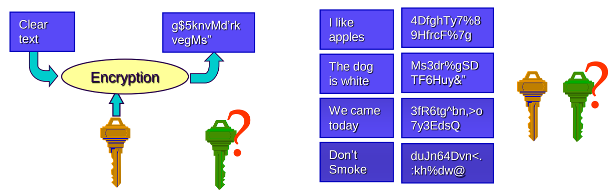
Secure if Key length = Message Length

Asymmetric Encryption



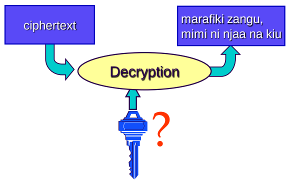
Asymmetric Encryption

- **Things to remember**
  - The relation between the two keys is unknown and from one key you cannot gain knowledge of the other, even if you have access to clear-text and cipher-text
  - The two keys are interchangeable. All algorithms make no difference between public and private key. When a key pair is generated, any of the two can be public or private (in theory but not in practice)

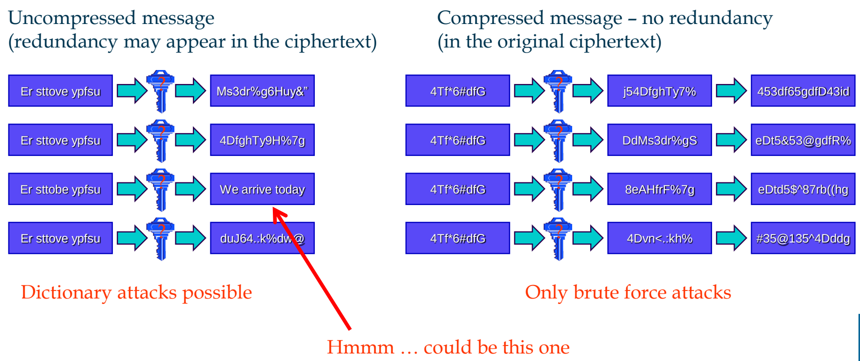


What "Cracking" means ...

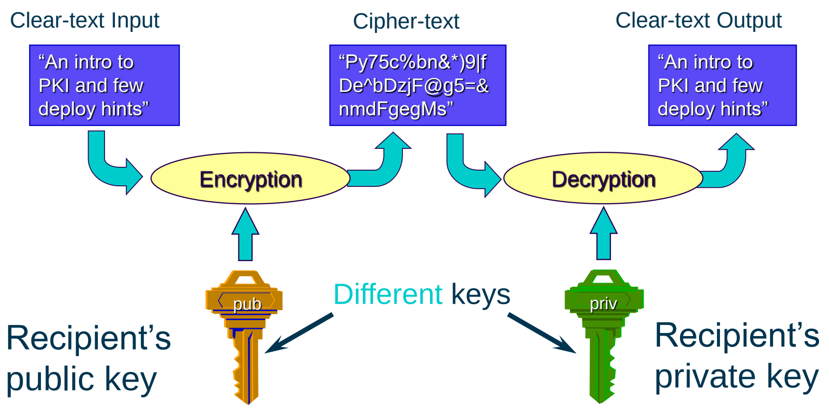
- **Cracking Asymmetric encryption is like solving a (difficult) mathematical problem that is entirely defined**
  - Find x, y so that
    x * y = 5549139029240772017554613865259030307060771696148489
  - (Answer: 2833419889721787128217599, 1958459827775699263024005111)
- **Cracking Symmetric encryption requires a way to verify that your "supposed" decryption is correct**
  - Guessing the message may rely on supposed redundancy
  - Compression is important because it removes redundancy



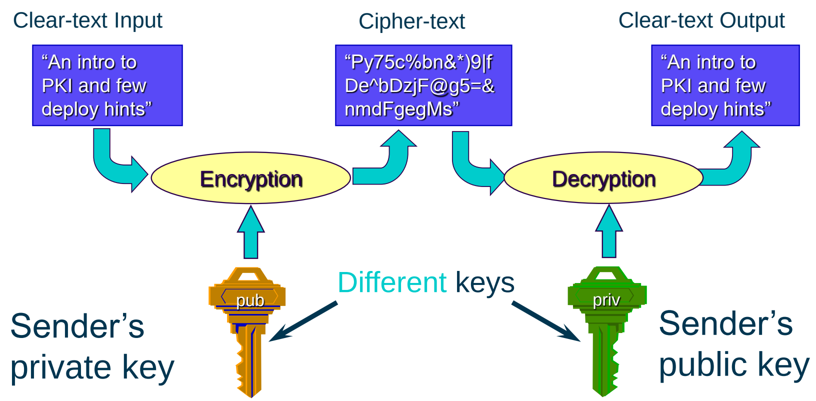
Cracking symmetric encryptions



Example: Confidentiality

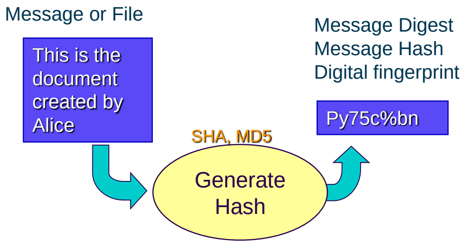


Example: Authenticity

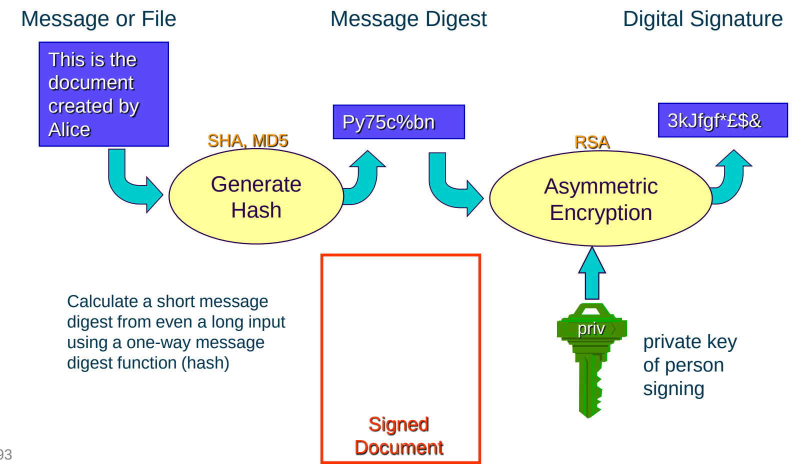


Cryptographic Hash Functions

- **A transformation that returns a fixed-size string, which is a short representation of the message from which it was computed**
  - Any (small) modification in the message generates a modification in the digest
- **Should be efficiently computable and impossible to:**
  - find a (previously unseen) message that matches a given digest
  - find "collisions", wherein two different messages have the same message digest

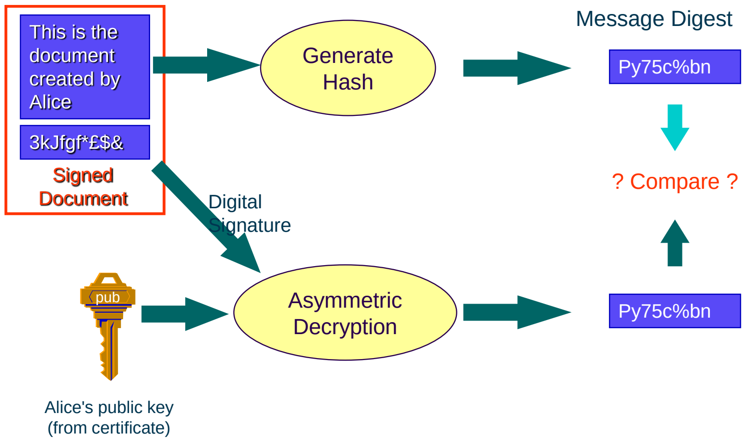


Example: Integrity Creating a Digital Signature



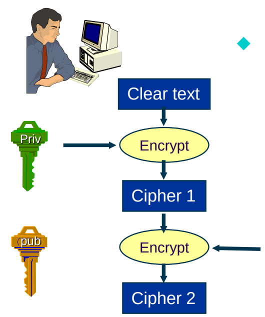
Calculate a short message digest from even a long input using a one-way message digest function (hash)

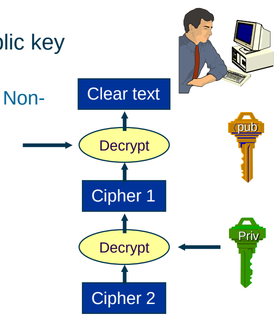
Verifying a Digital Signature

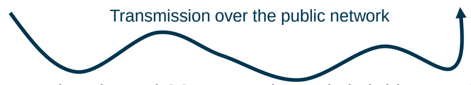


Example: SSL (simplified)

- Ensures confidentiality
  - And integrity if digitally signed
- depending on how public key are exchanged
  - Authenticity, Identity, Non-repudiation

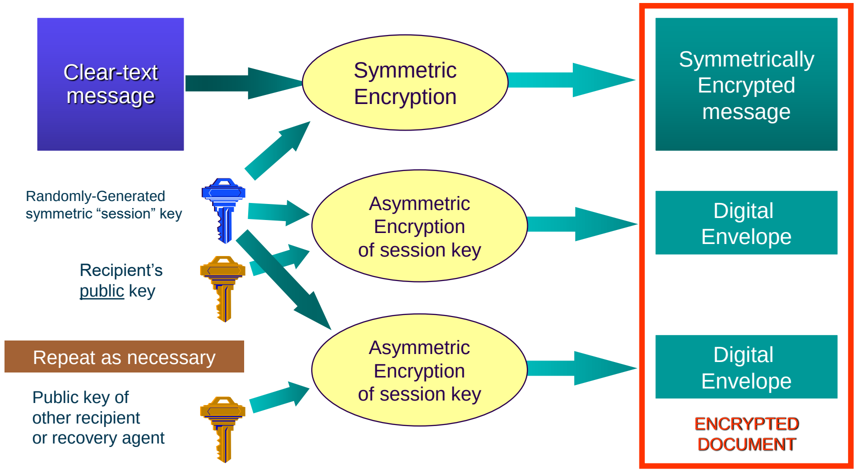




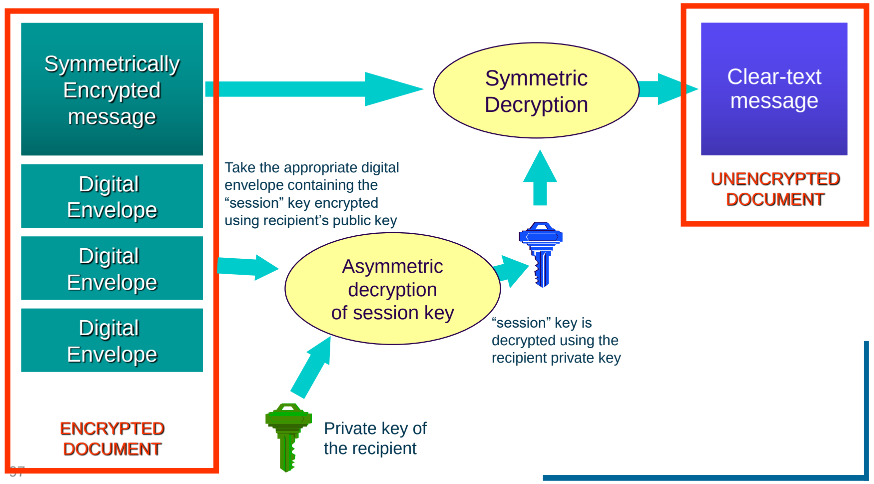


- In practice, the real SSL protocol uses in hybrid encryption
  - Why ?

Real World: Hybrid Encryption (typical for encrypted data storage)



Real World: Hybrid Decryption



Cryptography Security

- **Kerckhoff's Principle**
  - The security of the encryption scheme must depend only on the secrecy of the key and not on the secrecy of the algorithms
- **The algorithms should be known and published**
  - They should have resisted to hacking for quite some time
  - They are all based on the fact that some calculations are difficult to reverse (probabilistic impossible)
- **But design and key length matter (brute force attacks)**
  - This means that DES, 3DES, AES , RSA, ECC, MD5, SHA are not immune to attacks
  - They all have a certain strength you should be aware of

Agenda

- **Introduction to data management**
  - Data Workflows in scientific computing
  - Storage Models
- **Data management components**
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- **Summary**

Is cryptography enough ?

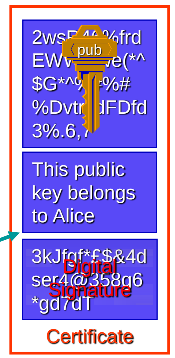
- **We just showed that cryptography solves the problem of confidentiality, Integrity, (Authenticity, Identity, Non-repudiation)**
  - How do we share secrets (symmetric encryption) and public keys (asymmetric encryption) safely on the internet ?
- **Problem …**
  - Michel creates a pair of keys (private/public) and tells everyone that the public key he generated belongs to Alice
  - People send confidential stuff to Alice
  - Alice cannot read as she is missing the private key to decrypt …
  - Michel reads Alice's messages
- **Except if people have met in some private place and exchanged a key, they'll need help from a third party who can guarantee the other's identity.**
  - PKI is one technology to share and distribute public keys (asymmetric encryption)
  - Kerberos another technology to share and distribute shared secrets (symmetric encryption)

PKI = Public Key Infrastructure

- **"A technology to implement and manage E-Security" A. Nash, "PKI", RSA Press**
- **My definition of PKI**
  - "Public Key Infrastructure provides the technologies to enable practical distribution of public keys"
  - Using CERTIFICATES
- **PKI is a group of solutions for :**
  - Key generation
  - key distribution, certificate generation
  - Key revocation, validation
  - Managing trust

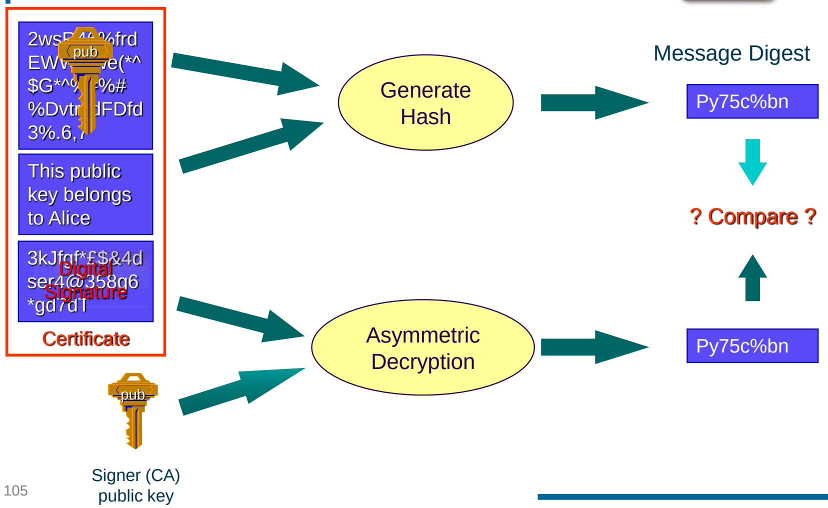
What is a Certificate ?

- **The simplest certificate just contains:**
  - A public key
  - Information about the entity that is being certified to own that public key
- **… and the whole is**
  - Digitally signed by someone trusted (like your friend or a CA)
  - Somebody for which you ALREADY have the public key

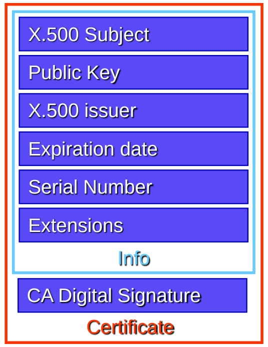


Can be a person, a computer, a device, a file, some code, anything …

Verifying a Certificate



X.509 Certificate (simplified)



Who is the owner, CN=Alice,O=CERN,C=CH

The public key or info about it

Who has signed, O=CERN,C=CH

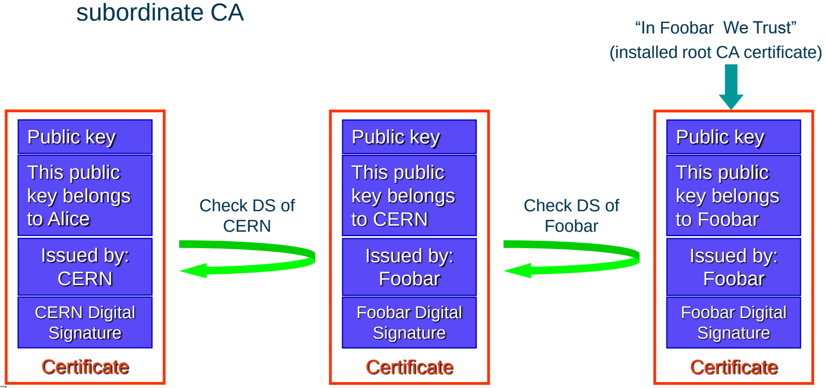
See later why expiration date is important

Additional arbitrary information

… of the issuer, of course

Certificate Validation

- When checking the digital signature you may have to "walk the path" of all subordinate authorities until
  - you reach the root ... or ... you reach an explicitly trusted subordinate CA



Authentication with Certificates

- **Owning a Certificate of Alice does not mean that you are Alice**
  - Owning a Certificate does not imply you are authenticated
- **How would you verify that the person who comes to you pretending to be Alice and showing you a certificate of Alice is really Alice ?**
  - You have to challenge her !
  - Only the real Alice has the private key that goes in pair with the public key in the certificate.

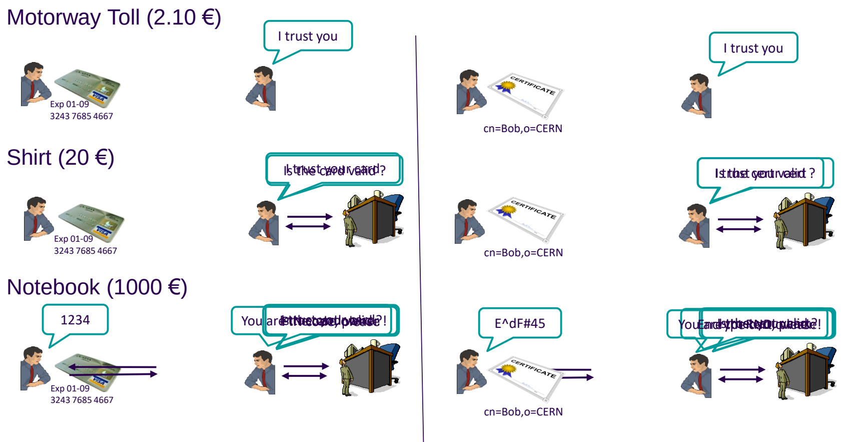
Authentication with Certificates

- **Bob gets Alice's certificate**
- **He verifies its digital signature**
  - He can trust that the public key really belongs to Alice
  - But is it Alice standing if front of him, or is that Michel ?
- **Bob challenges Alice to encrypt for him a random phrase he generated ("I like green tables with flowers")**
- **Alice has (if she is the real Alice) the private key that matches the certificate, so she responds ("deRf35D^&#dvYr8^*$@dff")**
- **Bob decrypts this with the public key he has in the certificate (which he trusts) and if it matches the phrase he just generated for the challenge then it must really be Alice herself !**

Traditional Human authentication versus Certificate authentication

- High Security Entrance
- Immigration
- Authentication for payments

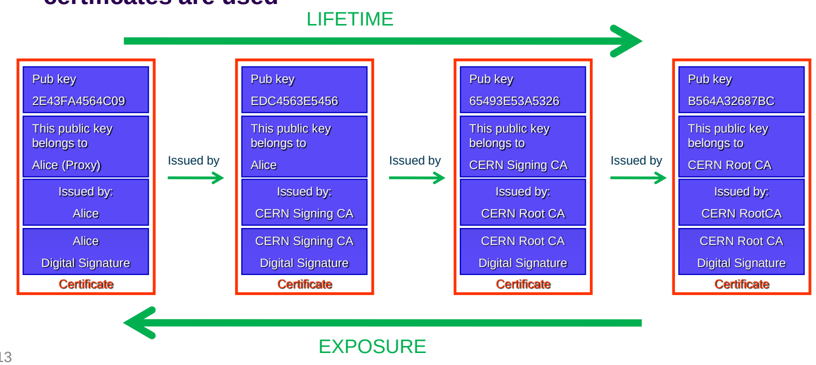
A comparison with real life



It is all about managing risks !

Certificate Key Hierarchy

- Every time the private key is used to encrypt something, it is somehow indirectly exposed.
- To limit the exposure, and to limit the impact and the cost of an eventual compromise, key hierarchies and proxy certificates are used



Where should certificates be stored

- Certificates can be stored anywhere
  - Store them in a file or a “dumb” memory-only smartcard
  - You can publish them in your LDAP directory
- No need to protect Certificates
  - ... from being tampered as they are digitally signed
  - ... them from being read as they contain only public information
- Private keys that match the public key are confidential
  - Loosing the private key = Loosing the identity
- Private keys should be stored in (at least) ...
  - computers files, protected by pass phrases
  - OS protected storage
  - smartcards

Certificates on Smartcards

- A “bad” smartcard is only a dumb memory chip
  - Containing the Certificate and the private key
  - Both readable: You must trust the machine reading your smartcard
  - Better than saving everything to a file
- A “good” smartcard is more than a memory chip
  - Contains the Certificate, readable
  - Contains the private key but not readable from outside. However it exposes a mechanism to challenge the knowledge of the private key by allowing the encryption of random strings using the private key
- A “very good” smartcard
  - May request the user to know a PIN code to execute any encryption request
  - (of course, now you have to protect the PIN code)
  - May support biometric recognition instead of the pin code

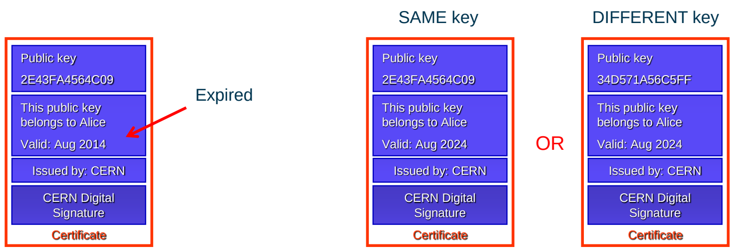
Increased cost

Certificate Revocation

- (Private) keys get compromised, as a fact of life
- You or your CA issue a certificate revocation certificate
  - Must be signed by the CA, of course
- And you do everything you can to let the world know that you issued it. This is not easy
  - Certificate Revocation Lists (CRL) are used
  - They require that the process of cert validation actively checks the CRL and keep it up-to-date
  - It is a non scalable process
  - Many people disable this function
- This explains why
  - Every certificate has an expiration date
  - Short expiration policies are important

Certificate Renewal

- **When the certificate is expired, the Certificate authority has two options when issuing a new certificate:**
  - create a new certificate with a new expiration date using the same public key (so that the user can continue to use the same private key he was using in the past)
  - OR, force the new certificate to have a different public key
- **The choice between the two options may depends on the “intended purpose” of the certificate (which is written in the certificate)**
  - Example: Authentication, Signing or Encrypting

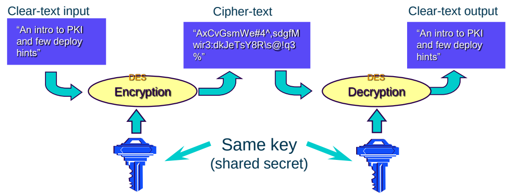


Kerberos: An alternative to PKI

- Identical goals of PKI
- Advantages:
  - Simpler to manage, keys managed automatically, Users understand it better
  - Forwardable authentication easier to implement
- Disadvantages
  - Cross Domain Authentication and Domain Trusts more difficult to implement
  - Offline authentication difficult to implement

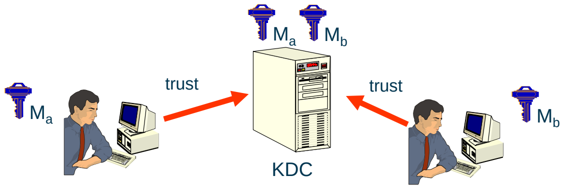
Kerberos Basics

- Kerberos is an authentication protocol based on conventional cryptography
- it relies on *symmetrical* cryptographic algorithms that use the same key for encryption as for decryption
  - Different from PKI !



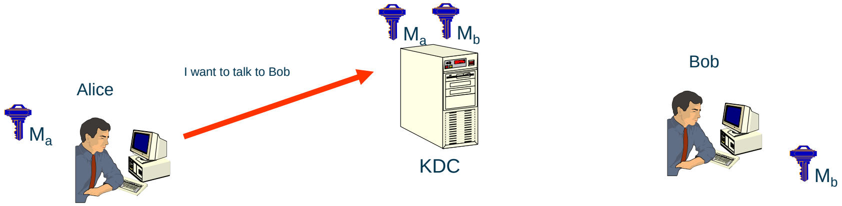
Basic principles

- There is an authority known as the Key Distribution Center (KDC). Every user shares a secret key with the KDC, which allow him to communicate securely with the KDC
- Everybody trusts the KDC
- The secret master key is different for each user
  - Two users have no direct way of verifying each other's identity
- The job of the KDC is to distribute a unique session key to each pair of users (security principals) that want to establish a secure channel.
  - Using symmetric encryption



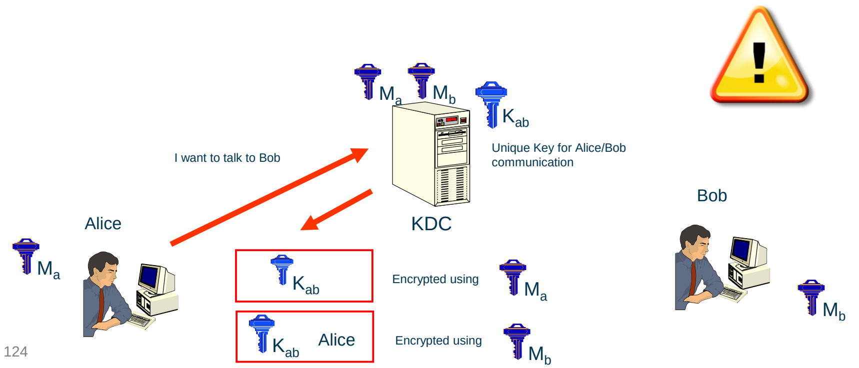
A (simplified) Kerberos session

- Alice wants to communicate with Bob
  - bob could be a server or a service
- Alice can communicate securely with the KDC, using symmetric encryption and the shared secret (Master Key)
- Alice tells the KDC that she wants to communicate with Bob (known to the KDC)



(simplified) Kerberos session 2

- The KDC generates a unique random key for Alice and Bob (K_{ab})
- Two copies of K_{ab} are sent back to Alice.
  - The first copy is sent encrypted using Alice's master key
  - The second copy of K_{ab} is sent with Alice's name encrypted with Bob's master key. This is known as the “kerberos ticket"

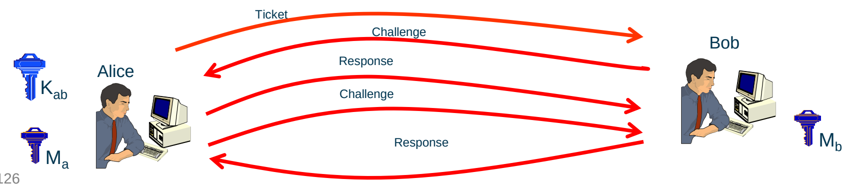


What is the ticket ?

- The ticket is a message to Bob that only Bob can decrypt
  - "This is your KDC. Alice wants to talk to you, and here's a session key that I've created for you and Alice to use. Only you, me and Alice know the value of K_{ab}, since I've encrypted it with your respective master keys. If your peer can prove knowledge of this key, then you can safely assume it is Alice."

How authentication *could* be done

- Alice sends the ticket to Bob. Bob takes the ticket and decrypts K_{ab}
- Bob generates a “random phrase” (the challenge) that is sent back to Alice
- Alice encrypts the “random phrase” using K_{ab} and send the results to Bob
  - She proves that she knows K
- Bob has authenticated Alice
- The whole could be repeated to have Alice authenticating Bob (are you sure ? Security !)
- But Kerberos doesn’t work that way ! It is smarter and anticipates the challenge by encrypting the “current time”

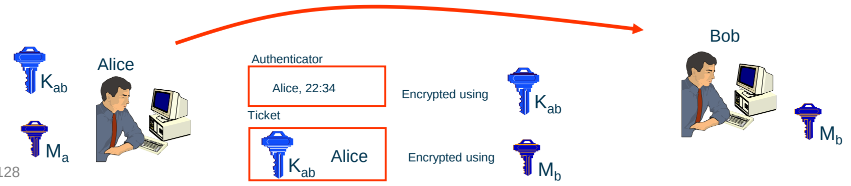


Reflection vulnerability

- Consider a normal session between a client and server:
  1. Client connects to Server.
  2. Server sends a challenge to Client.
  3. Client computes the response to the challenge and sends it to Server.
  4. Server performs the same calculation as the Client using the credentials it has stored.
  5. Server compares the response to its own calculated value. If the two match, the connection is a success.
- During a Reflection attack, the session proceeds as follows:
  1. The client (victim) initiates a connection to the server (attacker).
  2. Here, the attacker's instead of sending a challenge to the victim it initiates a new connection to the victim.
  3. The victim generates a challenge for the inbound connection from the attacker.
  4. The attacker takes the challenge received in Step 3 and sends it to the victim as the challenge for the connection the victim initiated in Step 2.
  5. The victim computes the response to the challenge and sends it to the attacker.
  6. The attacker takes the response received in Step 5 and returns it to the victim as the response to the connection initiated to the victim in Step 2.

Kerberos authentication

- **Alice sends the ticket to Bob**
- **Alice must also proof that she knows K_{ab}**
  - She also sends her name and the current time, all encrypted with the session key K_{ab} (this is called the authenticator)
- **Bob takes the ticket, decrypts it, and pulls K_{ab} out. Then decrypts the authenticator using K_{ab}, and compares the name in the authenticator with the name in the ticket**
  - If the time is correct, this provides evidence that the authenticator was indeed encrypted with K_{ab}
  - Bob can also detect replays from attackers listening on the network where Alice, Bob, and the KDC are conversing
    - By rejecting authenticators with time already used
    - By rejecting authenticators with the wrong time (question for the students: Why ?)

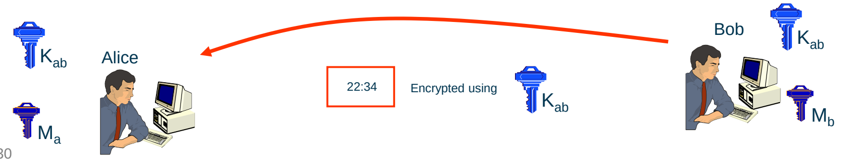


Kerberos authentication

- **Why encrypting the current time ?**
  - As the goal is to prove the knowledge of the shared secret (K_{ab}), you can see this action as a "challenge" on the knowledge of the shared secret (K_{ab}):
  - "prove that you know K_{ab} by encrypting the current time for me"
- **Why does time need to be synchronized ?**
  - To defeat replaying an earlier attempt, Bob needs to remember all past times that have been previously used. As this approach does not scale, Bob remembers only all times used in the past within a certain time interval (example: "within five minutes") around his own time.
  - If the time encrypted using the shared secret (K_{ab}) is within his time interval, Bob can be sure that this time has never been used before by comparing it with all the recorded past times
  - If the time encrypted using the shared secret (K_{ab}) is outside his time interval, Bob cannot be sure that this time has never been used before and therefore he rejects the request … with a hint of what his time is (Bob's time isn't a secret)

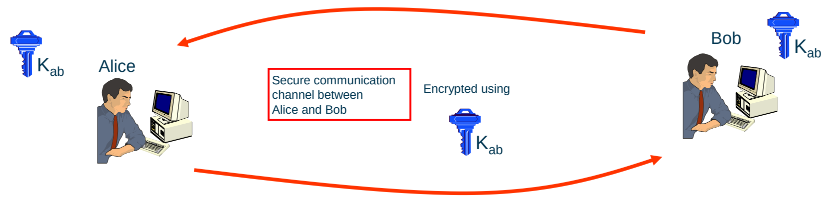
Mutual authentication

- **Alice has proved her identity to Bob**
- **Now Alice wants Bob to prove his identity as well**
  - she indicates this in her request via a flag.
- **After Bob has authenticated Alice, he takes the timestamp she sent, encrypts it with K_{ab}, and sends it back to Alice.**
- **Alice decrypts this and verifies that it's the timestamp she originally sent to Bob**
  - She has authenticated Bob because only Bob could have decrypted the Authenticator she sent
  - Bob sends just a piece of the information in order to demonstrate that he was able to decrypt the authenticator and manipulate the information inside. He chooses the time because that is the one piece of information that is sure to be unique in Alice's message to him



Kerberos Secure Communication

- **Alice and Bob share now a unique secret K_{ab} that they use to communicate**

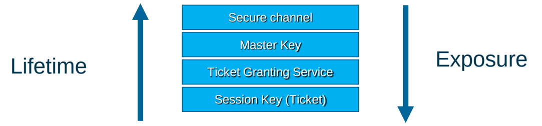


But real life is more complicated

- **Real Kerberos includes an extra step for additional security**
- **When Alice first logs in, she actually asks the KDC for what is called a "ticket granting ticket", or TGT.**
- **The TGT contains the session key (K_{ak}) to be used by Alice in her communications with the KDC throughout the day.**
  - This explains why when the TGT expires you have to renew it
- **So when Alice requests a ticket for Bob, she actually sends to the KDC her TGT plus an authenticator with her request.**
- **The KDC then sends back the Alice/Bob session key K_{ab} encrypted with K_{ak}**
  - as opposed to using Alice's master key as described earlier
  - Alice doesn't even need to remember her master key once she receives the TGT (unless she wants automatic TGT renewal).

Kerberos Key Hierarchy

- The session key (or short-term key). A session key is a secret key shared between two entities for authentication purposes. The session key is generated by the KDC. Since it is a critical part of the Kerberos authentication protocol, it is never sent in the clear over a communication channel: It is encrypted using the Ticket Granting Services key
- The Ticket Granting Services key (medium-term key). A secret key shared between each entities and the KDC to obtain session keys. It is never sent in the clear over a communication channel: It is encrypted using the master key.
- The master key (or long-term key). The master key is a secret key shared between each entity and the KDC. It must be known to both the entity and the KDC before the actual Kerberos protocol communication can take place. The master key is generated as part of the domain enrollment process and is derived from the creator's (user, machine, or service) password. The transport of the master key over a communication channel is secured using a secure channel.
- The secure channel. The secure channel is provided by the master key shared between the workstation you're working on and the KDC. In this case the master key is derived from the workstation's machine account password.



Kerberos ticket in real life

	Field name	Description
	tkt-vno	Version number of the ticket format. In Kerberos v.5 it is 5.
	Realm	Name of the realm (domain) that issued the ticket. A KDC can issue tickets only for servers in its own realm, so this is also the name of the server's realm
	Sname	Name of the server.
🔑	Flags	Ticket options
🔑	Key	Session Key
🔑	Crealm	Name of the client's realm (domain)
🔑	Cname	Client's name
🔑	Transited	Lists the Kerberos realms that took part in authenticating the client to whom the ticket was issued.
🔑	Starttime	Time after which the ticket is valid.
🔑	Endtime	Ticket's expiration time.
🔑	renew-till	(Optional) Maximum endtime that may be set in a ticket with a RENEWABLE flag.
🔑	Caddr	(Optional) One or more addresses from which the ticket can be used. If omitted, the ticket can be used from any address.
🔑	Authorization-data	(Optional) Privilege attributes for the client. Kerberos does not interpret the contents of this field. Interpretation is left up to the service.

🔑: Fields encrypted using the session key of the recipient's TGT

Auth: Various scenarios possible

- **Authentication is "delegated" to the operating system**
  - A local account exist for every potential user connecting to the service
  - local accounts could be "created on the fly" when missing (example: Grid applications)
  - The daemon process impersonates the user account when executing the requests on behalf of the user
  - The Authorization can be delegated to the operating system
- **The Authentication is managed by the application**
  - The user is authenticated and the identity of the user is attached as an attribute to the request
  - The daemon process runs under full privileges and has to verify the permissions on every request (Authorization)
- **Both approaches are valid**

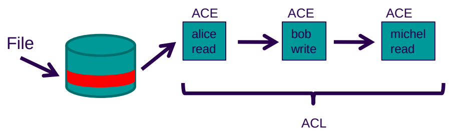
Agenda

Authorization

- **Implements the access control**
  - The information describing what end-user can do on computing resources. It is the association of a right (use, read, modify, delete, open, execute, …), a subject (person, account, computer, group, …) and a resource (file, computer, printer, room, information system, …)
  - The association can be time-dependent
- **Authorization process**
  - Verification that the connected user has the permission to access a given resource

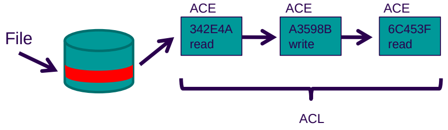
Authorization in practice

- **Every resource has, among its metadata, a linked list called ACL (Access Control List)**
- **The ACL is made of ACEs: Access Control Entries**
  - ACE contains the "subject" (person, account, computer, group, …) and the "right" (use, read, modify, delete, open, execute, …) that the subject has on the "resource". Eventually also the "time" when the ACE is valid



Authz: How to identify users

- **Two approaches**
- **Users can be identified by “login name” of by the “subject” found in the certificate**
  - Easier to understand, but changing login name or changing the “subject” in the certificate means changing the identity
- **Users can be identified by a unique GUID or Virtual ID. The login name or the certificate “subject” become only an account attribute**
  - Extremely more flexible, no performance penalty
  - All serious security implementations should follow this way ...
  - But GUID or Virtual ID are specific to an instance. Authorization information need to be recalculated if the data is moved from one site to another

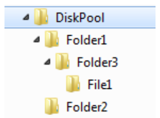


342E4A : alice
A3598B : bob
6C453F: michel

Authz: Implementation choices

- **Where ACL should be stored ?**
  - Usual dilemma: database or with the resource ?
  - It is an additional DB lookup “in the line of fire”
- **How ACL should be verified ?**
  - Authorization is “delegated” to the operating system: The ACL are set on the file system and the process accessing the file impersonates the credential of the user
  - The Authentication is managed by the application: The permissions of the request owner must be verified by the Data Management software on every request.

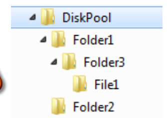
Authz: Some complications



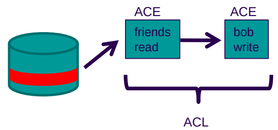
- **Inheritance**
  - ACL must be supported on every node (folders) of the file structure. Inheritance flags must be foreseen
  - Another dilemma:
    - Calculate the “resultant permissions” in real time when the file is accessed
      - Possible but require low level code optimization, otherwise heavy performance hit
      - Very efficient when permissions need to be changed
    - Compile the “resultant permissions” (File permissions + the Inherited permissions) with the file.
      - Very efficient when resolving permissions (one DB lookup)
      - Very inefficient when changing permission
- **Choice between optimize read or write speed**
- **In real life ... A mix of the two is implemented**
  - And many file systems do this

What about cloud storage ?

Authz: more complications

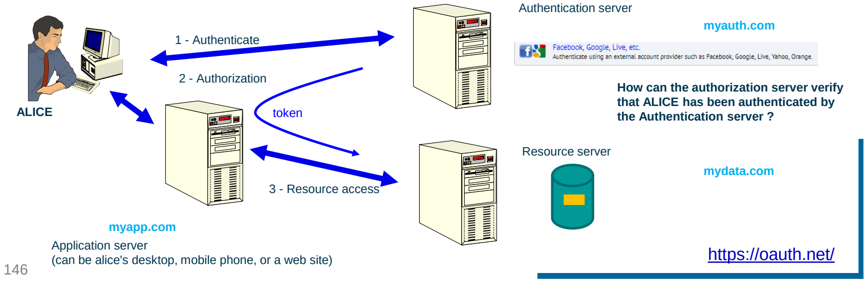


- **Support for “groups” of users and “roles”**
- **Allows ACLs can be granted to aggregate groups of users**
- **Another dilemma: When should be “group” resolution be done ?**
  - At runtime, when the resource is accessed ? Possible but inefficient (another DB lookup)
  - At “Authentication” time. The Authentication token contains the login name and all groups the user belongs to. The ACL is then compared against the users and all groups he belongs to. Much better but changes in group membership require re-authentication to be effective.
- **What is the scope of the “group” ?**
  - Local to the Storage Element ?
  - Local to the Site ?
  - Global as the grid users ?



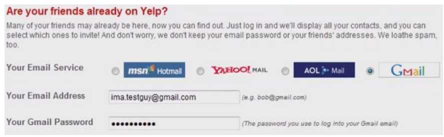
OpenID Connect, Oauth-2

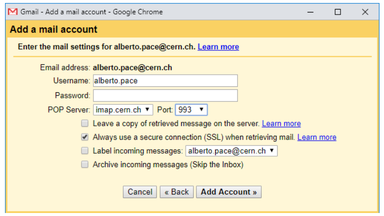
- **In all scenarios so far, the server that authenticates is also the server that authorizes accesses. What if …**
  - The authorization server wants to delegate the authentication to another (trusted) server
  - The authentication server wants to allow (untrusted) servers to use its authentication credentials
  - Alice wants to access a resource without giving her password to the application or resource server …
    - Example: Authorization using a Google account, Facebook, Ms Live, …



https://oauth.net/

The wrong way …

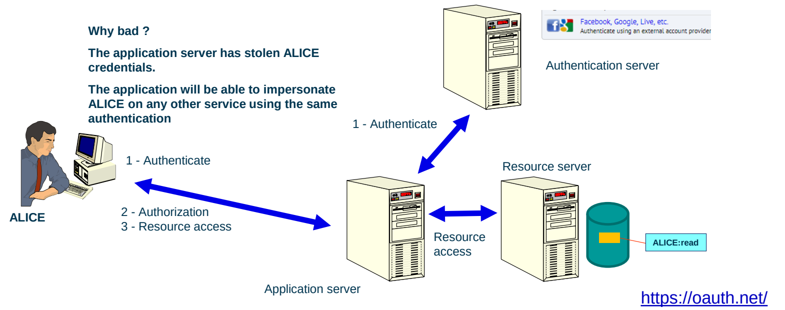




Why bad ?

The application server has stolen ALICE credentials.

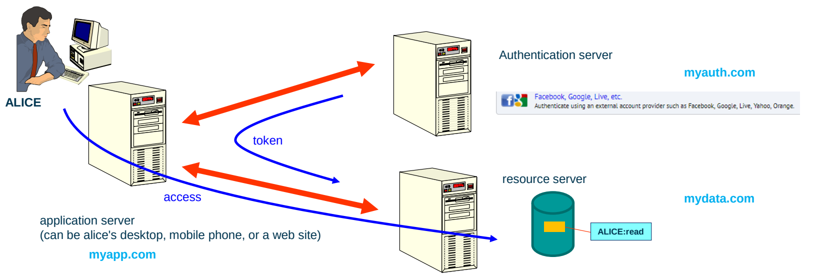
The application will be able to impersonate ALICE on any other service using the same authentication



https://oauth.net/

OpenID Connect, Oauth

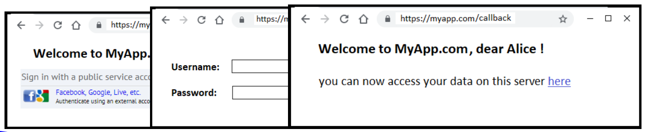
- **To generalize Oauth, lets consider 3 actors**
  - The application, the authentication and the resource servers
  - The application server wants to access the resource on behalf of Alice
  - Alice needs to be authenticated by the authentication server



https://www.youtube.com/watch?v=GyCL8AJUhww
https://www.youtube.com/watch?v=996OiexHze0&t=1468s
https://www.youtube.com/watch?v=CPbvxxslDTU

Authentication with OpenID Connect

- **This is the simplest case, there is no resource server**
  - All resources are owned by the application server which also handle the authorization. The application only needs the identity of the person connecting to it.



ALICE

1 - access application

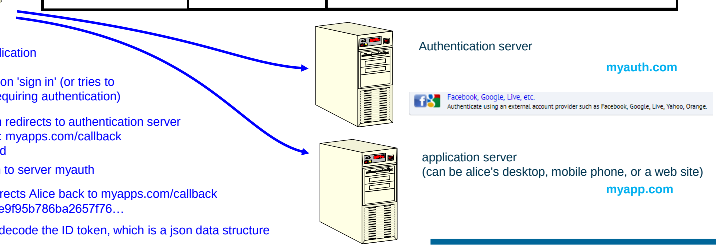
2 - Alice clicks on 'sign in' (or tries to access page requiring authentication)

3 - myapp.com redirects to authentication server
- Redirect URI: myapps.com/callback
- Scope: openid

4 - Alice logs in to server myauth

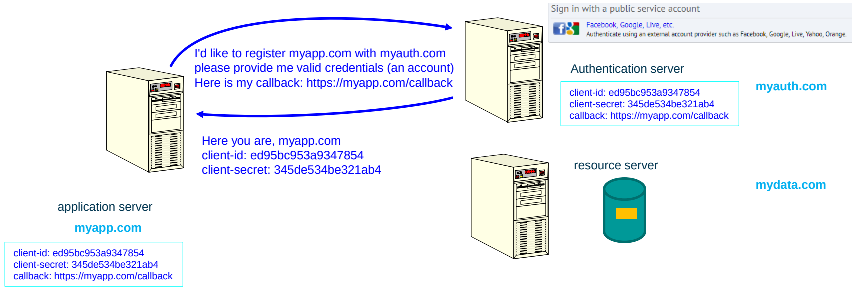
5 - myauth redirects Alice back to myapps.com/callback
- ID token: 34de9f95b786ba2657f76…

6 - myapp can decode the ID token, which is a json data structure



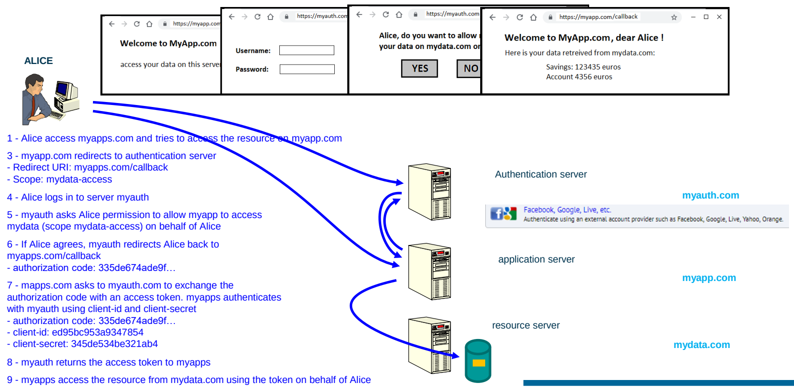
Authentication and Authorization with Oauth

- **Alice needs to agree to have his data accessed by myapp.com**
- **The authentication server needs to ensure that tokens are only given to myapp.com**
  - therefore myapp.com needs to have an Oauth client account in myauth.com
  - The resource server trusts the authentication server
- **The account allows the application server to authenticate the application**
  - This will allow to create the access token only for mydata.com



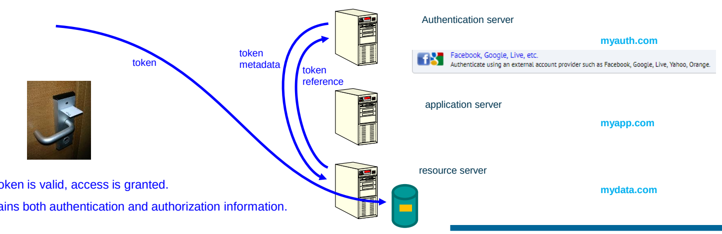
Accessing resources with Oauth-2

- **There are several scenario possible.**
- **Here is the most generic one, with both application and resource servers**



Validating the token

- **How does the resource server validate the token ?**
- **Two possibilities:**
  - The token is a self contained
    - Need to decode the token which contains all info in json format
    - Important: need to verify the digital signature (integrity check)
    - cannot be revoked
  - The token is a reference
    - Need to verify the token with the myauth server
    - myauth server retains full control on its validity. Allows token revocation



If the token is valid, access is granted.

It contains both authentication and authorization information.

Agenda

- Introduction to data management
  - Data Workflows in scientific computing
  - Storage Models
- Data management components
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, **Accounting**
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- Summary

Accounting, Transactions and Undo

- **List of actions (who, when, what, where) that enables traceability of all changes and transactions rollback**
- **Multiple levels of accounting possible**
  - Simple logging: Allows you to know the history
  - Logging + journal of all transaction: Allows you to know the history and rollback in time
- **A good accounting is a valid alternative to a strict authorization scheme**
  - Users are “empowered” but held responsible of their actions

Agenda

- **Introduction to data management**
  - Data Workflows in scientific computing
  - Storage Models
- **Data management components**
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- **Summary**

Cloud Storage

- **Storage generally hosted by third parties on the Internet which offers a model of virtual pools.**
  - Highly scalable
  - Pay “a la carte” as you use / as you store
- **Simple interfaces to access storage**
  - HTTP Put/Get
  - Amazon S3 (Simple Storage Services) API
- **Not posix compliant**
  - The lack of a posix interface allows the deployment of scalable infrastructures
- **Various Pro and Cons using cloud storage**
  - See next slide

http://en.wikipedia.org/wiki/Cloud_storage

Why cloud storage ?

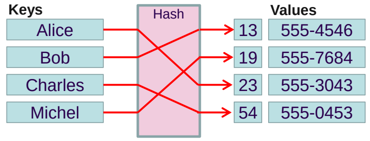
- **It is simpler …**
  - Single pool where all data go. Reliable, Fast and outsourced.
  - Unique quality of service
  - Economically interesting for small / medium data sizes as only variable costs are exposed
- **… but can be simplistic**
  - The single pool with unique quality of service is very far from the requirements of scientific data analysis: it can become expensive or inefficient

Technologies used by Cloud storage

- **Distributed Hash Table (DHT)**
  - Empowers scalability. Storage spans multiple clusters / multiple data centers and is federated under a unique name space
- **Keywords and technology in DHTs**
  - Hash Tables
  - Hash algorithm and collisions
  - ***Distributed*** Hash Tables and their challenges

Hash Table

- **A hash table is a data structure that uses a hash function to map keys (ex: a person's name) to their values (ex: the telephone number). A hash table implements an associative memory.**



- **The hash function is used to transform the key into the index (the hash) of an array element where the corresponding value is stored.**
- **In a hash table, the cost (number of instructions) for a lookup is independent of the number of elements stored in the table: perfect scalability.**
- **Also insertions and deletions of key-value pairs can be done at constant cost per operation**

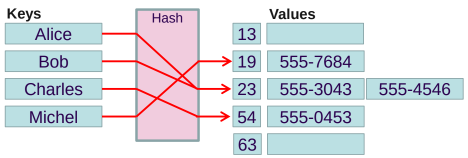
http://en.wikipedia.org/wiki/Hash_table

Hash Algorithms and Collisions

- **Note that the hash function defines the maximum number of entries in the hash table (new entries are never added after the table is created):**
  - CRC16 – 2^{16} entries (65536)
  - SHA1 – 2^{160} entries (> 10^{48})

Hash	Algorithm
8450fdfa6e8e47313920cd0c877c73e8	MD5
68205ce7	CRC32
cb6c	CRC16
bcb2bb1b03d2f9d0c32561d59f0301e382dc56fd	SHA1

- **All hash functions have collisions. These must be handled**



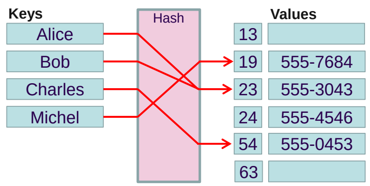
http://en.wikipedia.org/wiki/Hash_table

Hash Algorithms and Collisions

- **Note that the hash function defines the maximum number of entries in the hash table (new entries are never added after the table is created):**
  - CRC16 – 2^{16} entries (65536)
  - SHA1 – 2^{160} entries (> 10^{48})

Hash	Algorithm
8450fdfa6e8e47313920cd0c877c73e8	MD5
68205ce7	CRC32
cb6c	CRC16
bcb2bb1b03d2f9d0c32561d59f0301e382dc56fd	SHA1

- **All hash functions have collisions. These must be handled**



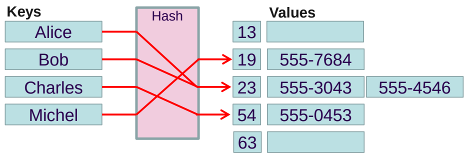
http://en.wikipedia.org/wiki/Hash_table

Hash Algorithms and Collisions

- **Note that the hash function defines the maximum number of entries in the hash table (new entries are never added after the table is created):**
  - CRC16 – 2^{16} entries (65536)
  - SHA1 – 2^{160} entries (> 10^{48})

Hash	Algorithm
8450fdfa6e8e47313920cd0c877c73e8	MD5
68205ce7	CRC32
cb6c	CRC16
bcb2bb1b03d2f9d0c32561d59f0301e382dc56fd	SHA1

- **All hash functions have collisions. These must be handled**



- **The Load factor (or Fill factor) is equal the ratio between the number of stored entries (4) and the size (00 - 99) of the table's array.**
- **The probability of collisions and the cost of handling them increases with the load factor that must be kept low.**
- **Note that as load factor approaches 0, the proportion of unused areas in the hash table increases resulting in wasted memory.**

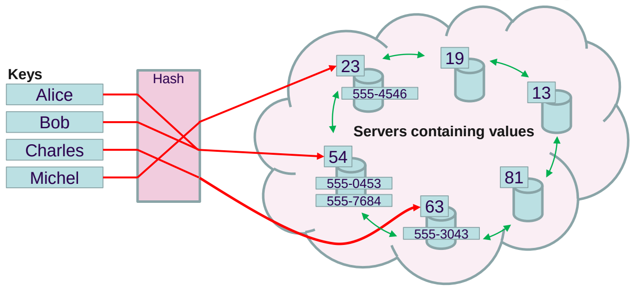
http://en.wikipedia.org/wiki/Hash_table

Resizing Hash Table

- **When the load factor is close to 0 (waste of memory) or 1 (too many collisions), you may need to resize the hash table**
  - This requires a full or incremental rehash of all keys.
  - This breaks scalability …
- **Workaround:**
  - Choose a hash function that preserve the key hashes when the table is resized. This approach, called **consistent hashing**
  - Essential in Distributed Hash Tables when load balancing between servers.
    - if a server is added or removed and every object is hashed to a new location, this would require rewriting all stored data

http://en.wikipedia.org/wiki/Consistent_hashing

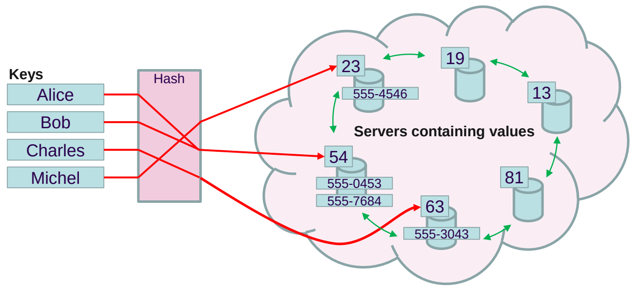
Distributed Hash Tables (1/2)



- A **keyspace partitioning** scheme distributes hash values among the multiple servers. (See more on partitioning)
- An **overlay network** (⟷) connects the nodes, allowing any of them to route a data request (put, get) to the server owning any key in the keyspace.
- Must be able to add / remove nodes: **Consistent Hashing** is essential (allows removal or addition of one server by affecting only adjacent nodes). (See more on partitioning)

http://en.wikipedia.org/wiki/Distributed_hash_table

Distributed Hash Tables (2/2)



- **Reliability** and **Fault Tolerance** can be implemented by replicating values (or error correction information) on n adjacent nodes.
- If nodes are distributed over wide area, reliability can be calculated (because probabilities of failures are independent)
- **Highly scalable**, reading and writing data is independent of the size of the hash table.

http://en.wikipedia.org/wiki/Distributed_hash_table

Keyspace partitioning (1/2)

- **We mentioned “Range” vs “Hash” partitioning**
  - Range partitioning requires a table that maps objects into storage instances – Easy to add instances
  - Hash partitioning requires all data to be moved when adding a new instance, unless you use consistent hashing
    - If you do not use consistent hashing you can also start with a large number of storage instances that you deploy on few servers. When data grows, you move the self contained instances to another server (Q: how do you know on which server is an instance ?)

https://redis.io/topics/partitioning

Keyspace partitioning (2/2)

- **Where is the partitioning job done ?**
  - In the client
    - The client knows or calculates which servers owns a key
  - In a middle-layer server (proxy assisted partitioning)
    - An intermediate database knows or calculates which servers owns a key
  - In the server instances
    - Only the servers know or calculate who owns a key. The clients send queries to a random server which ensures to either forward or redirect the request to the appropriate server (cfr: overlay network).

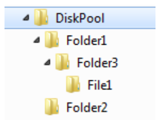
http://oldblog.antirez.com/post/redis-presharding.html

Cloud storage in practice

- **The DHT implements storage elements called buckets**
- **The programming interfaces to (API) the bucket allows you read/write any portion of the bucket storage.**
  - You can store a file system in a bucket
  - You can store a database in a bucket
  - You can store a disk image in a bucket
  - In general, you store objects in a bucket
- **Every functionality limiting scalability is dropped:**
  - Hierarchical storage in folders / directories
  - Permission inheritance
  - Authorization based on groups / Roles
- **Example of Amazon S3 API:**

DELETE Object	PUT Object acl
Delete Multiple Objects	PUT Object - Copy
GET Object	Initiate Multipart Upload
GET Object ACL	Upload Part
GET Object torrent	Upload Part - Copy
HEAD Object	Complete Multipart Upload
POST Object	Abort Multipart Upload
PUT Object	List Parts

Hashes vs Checksums



- **C (a + b) = C (a) + C (b)**
  - C(Diskpool) = C (Folder1) + C (Folder2)
  - C(Folder1) = C (Folder3) = C (File1)
- Checksums just need to be different when the input is different (as far as possible), but it's almost as important that they're fast to compute.
- Hash codes (for use in hashtables) have the same requirements, and additionally they should be evenly distributed across the code space, especially for inputs that are similar.
- Cryptographic hashes have the *much* more stringent requirement that given a hash, you cannot construct an input that produces this hash. Computation times comes second. **Cannot be reversed.**

Note: Does simple checksumming guarantee data integrity ?

Checksumming and data integrity

- You want to hack digitally signed document that uses a (inappropriate) hash/checksum that supports C (a + b) = C (a) + C (b)
- The original document has a hash of H_O
- Your tampered document has a hash of H_H
- You only need to find an arbitrary small document D whose hash H_D satisfies the equation
  - $H_O = H_H + H_D$
- if you are using 32 bit checksums (like adler32), this can be as simple as adding 4 bytes to the document to manipulate its content while preserving the checksum !

Agenda

Block Storage

- Block storage is a level of abstraction for storage organized in “blocks”, where a block is data having a nominal (fixed) length (a block size).
- When the size exceeds one block, data is stored in multiple blocks.
- Whenever data size is not an exact multiple of the block size, the last block is only partially filled.

http://en.wikipedia.org/wiki/Distributed_hash_table

Choosing the block size

- Block storage leads to space inefficiency due to internal fragmentation as file lengths are rarely exact multiples of the block size
  - Small block sizes reduces the amount of storage wasted, but increases the number of blocks that must be read for a constant amount of data
  - Large block sizes reduces the number of blocks that must be read for a constant amount of data, but increases the amount of storage wasted

Why block-based storage ?

- **Variable-size storage becomes fixed-size storage**
- **More effective load balancing of storage**
  - Independent from file sizes
- **Allows implementation of error correction algorithms**
  - Even across multiple servers
  - Less wasted storage compared to data replication
- **Performance is proportional to the resources deployed and less dependent from access patterns**
  - Blocks of “hot files” are scattered on multiple servers
- **Software abstraction identical to traditional hard disk**
  - Blocks = Disk sectors

Usage block based storage

- **Block based storage can be mounted and seen as virtual hard disks**
  - Typical usage for virtual machines
  - The data on the virtual hard disk is striped and replicated across the various nodes of the storage cluster
  - Better than the physical hard disks:
    - The underlying striping can provide significantly higher performance than physical hard disks
    - Disk images and partition can be easily resized, exported, copied/renamed/duplicated.
    - Virtual disk snapshot is possible, with journal and rollback

Agenda

- **Introduction to data management**
  - Data Workflows in scientific computing
  - Storage Models
- **Data management components**
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- **Summary**

Analytics

- **A generic term, an abstraction from several (innovative) attempt to optimize the discovery of meaningful data patterns in large data sets**
- **Used in many disciplines**
  - Market analysis, Customer Analysis, Finance, Security, Monitoring, (Software) Accounting, History analysis, High Energy Physics …
- **Typically operates in a large computer network, with lot of data (Big Data) and can use the CPU in the storage server**

Many concepts

- **Several techniques**
  - Data mining
  - Machine learning
  - Artificial intelligence
- **Several technologies**
  - Databases (Sql / NoSql)
  - Structured / Unstructured data
  - Map reduce
  - Hadoop + its ecosystem
    - Hive, HBase, Spark, Impala, Flume, …

Dealing with unstructured information

- **Data that does not have a predefined data model or is not organized. It is typically text based (documents, email, messages) and contains dates, numbers, and assertions.**
- **Contains irregularities, ambiguities, and errors, every information has a probability of being incorrect**
- **It is difficult to process/analyze using traditional programs, compared to structured (and verified) data.**

In this course: Map-Reduce exercise

- **In a distributed computing environment, the separation of "Data servers" from the "Computing nodes" can lead to severe inefficiencies**
  - Due to round trip times and high latencies
- **Map – Reduce provides an architecture to have some data processing or analysis directly on the "Data servers".**
  - Easy to implement when the type of processing is unique / uniform and well defined for all data
  - Difficulties appears when interferences appears between the processing load on the data servers and the strong requirement of serving data with high efficiency

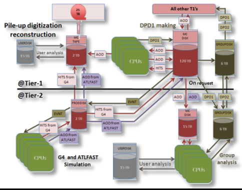
Map – Reduce

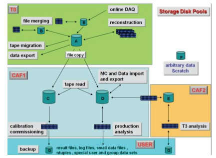
- **Often based on the open-source implementation of the Hadoop ecosystem**
- **The analytic consists of**
  - a design that splits the processing between parallel data processing and an aggregation algorithm
  - a "map" program that is distributed and running on every data node. This map program performs filtering and data analysis
  - a "reduce" program that aggregates the results coming from all "map" jobs
- **This requires orchestrating (parallelizing) the processing among multiple servers, managing all status, communications and data transfers between the various components of the system, including providing redundancy and fault tolerance.**

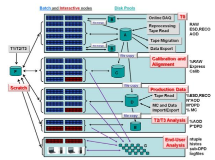
Agenda

Data replication (and syncing)

- **Data replication consists of transferring information between pools**
- **Under many different scenarios**
  - WAN and/or LAN
  - One way (master / slave), Two ways (multiple masters)
  - Aggregated / file-based transfers
  - Synchronous, scheduled, manual replications







Data replication: One / Two ways

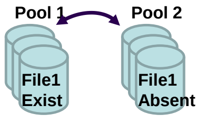
- **One Way (Master / Slave)**
  - The slave copy is read/only. Only the replication process can modify its content
  - Each create/modify/delete operation can be intercepted and replicated in real-time to the slave
  - To ensure consistency, a scheduled process scans and replicates differences found. The scan can be at different detail levels:
    - Scan meta information / file descriptor in the name server
    - Scan for the existence of every file
    - Scan the content of every file

Data replication: One / Two ways

- **Two Way (Multiple Masters)**
  - All pools are read/write. Each write operation can be intercepted and replicated to the other masters
  - Conflicts are possible and must be handled
    - Manually (raise alarm and stop)
    - Automatically (based on policy)
  - Several possible policies to resolve conflicts
    - Define which master is the "super-master"
    - Last modification wins
    - Duplicate data on conflict
  - To ensure consistency, a scheduled process scans and replicates differences found. The scan can be at different detail levels:
    - Scan meta information / file descriptor in the name server
    - Scan for the existence of every file
    - Scan the content of every file

Multi Master replication issues

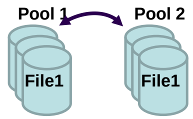
- **Handling the difference between**
  - file creation (copy file1 to pool2)
  - file deletion (delete file1 from pool1)



- **Options**
  - Remember the last sync date and compare to file's creation/modified dates
  - Keep a database of files of last sync and log all requests
- **Must be aware of pool errors**
  - File 1 is absent because “deleted” or because of failure of “Pool2”

Handling replications errors

- **Similar as handling multi-master conflicts**
  - Manually (raise alarm and stop)
  - Automatically (based on policy)
- **Many possible policies to re-synchronize**
  - Define the reference pool used to resolve errors
  - Use a third copy containing
    - File catalogue and metadata
    - File content hashes or checksums
    - File content



Same datetime
Same/Different size
Different content

Unique file identification

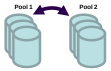
- **How should files be identified / recognized ?**
  - This allows identify file move and rename
  - Sync can be much faster …
- **Many Options**
  - Filename 👎
  - Filename + Size + Last Write Time 👉
  - Filename + Size + Last Write Time + Create Time 👉
  - Hash of file content 👍
  - Hash of file content + Size + Last Write Time 👍
  - Server-side File ID 👍 👎

Data Replication: Aggregated transfers

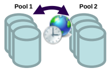
- **Data transfer efficiency relies on the “size of data”**
  - Not true for all media, but important for any storage requiring sequential access / lead-in / lead-out / tape marks / database access / ...
  - The “atom” of the transfer can be
    - A file
    - A directory
    - A Volume
    - An arbitrary set of files in a pool (independent of the file structure)
- **Concept of “Data Sets”**
  - The datasets predefines the “access pattern” to the data
  - When you recall multiple files in the same dataset, you are guaranteed maximum efficiency
- **Physical implementation may be very different**
  - Co-locate dataset files on the same media (tapes)
  - Scatter dataset files on multiple media (disks, for max throughput)

Data Replication: Schedule

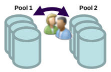
- **Real-time, triggered by data change**
  - data are always synchronized
  - High throughput, WAN replicas, redundancy



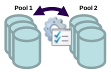
- **At scheduled intervals**
  - “pool 1” contains “pool 2” data from the past ...
  - Backup, Archives, History, Rollback



- **“Manual replication” (ex: the experiment recalls on disk pool data from 3 years ago from tape pool)**
  - Archives, Checkpoints, Baseline, Data recovery, bring online



- **“Custom” (the experiment scripts his transfer policy)**
  - Everything else
  - Includes data processing

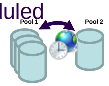


Data Technologies – CERN School of Computing 2019

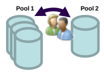
Data Replication: Transfer Type

- **Reliability can be achieved**
  - Within the pool (as we have already discussed)
  - With data replication
- **If one of the pool is reliable* there is no need to transfer all data to the destination pool**
  - Complete / differential / incremental content that allows all types of movement schedule using smaller pools.
  - Introduces the concept of “snapshot” and “journal”
  - Does not requite multiple pools: Can be within one pool !

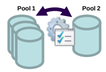
scheduled



manual



custom



*: the probability of the pool to lose data is below the accepted risk of the data owner

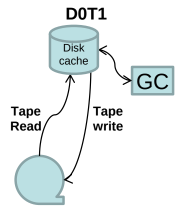
Data Technologies – CERN School of Computing 2019

Agenda

- **Introduction to data management**
  - Data Workflows in scientific computing
  - Storage Models
- **Data management components**
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- **Summary**

Data Technologies – CERN School of Computing 2019

Data Caching



- **The cache is a “fast” storage area where frequently accessed data can be temporary stored for rapid access.**
- **The cache algorithm tries to guess which data may be used in the future and to avoid re-fetching or re-computing the requested data, it keeps it in the cache**
- **A cache is effective whenever the correct guesses (Hits) outnumber the amount of data that need re-fetching (Misses)**

http://en.wikipedia.org/wiki/Cache

Cache considerations

- **Adding cache memory**
  - Increases the cost without increasing the capacity
  - Can increase the performance (depending on the efficiency)
- **Efficiency = “number of hits” / “number of requests”**
- **The efficiency of the cache depends on the data access pattern**
  - Streaming data: Efficiency = 0%
    - no data is ever requested again, the cache is useless
  - True Random Access: Efficiency = Cache Size / Total Size
    - future data requests cannot be predicted from past data operations. The cache performance is only proportional to its size
  - In real life, the efficiency varies depending on how access patterns can be predicted

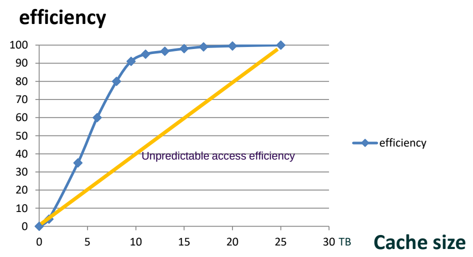
Cache predictions and algorithms

- **Predicting future data requests from past access patterns**
  - Temporal: recently accessed data has higher probability to be accessed again
  - Spatial: data close to recent requests has higher probability to be accessed
  - Read Ahead: data ahead of recent requests has higher probability to be accessed than data behind
- **Algorithms**
  - LRU – Least Recently Used
  - MRU – Most Recently Used
  - LFU – Least Frequently Used
  - ...

http://en.wikipedia.org/wiki/Cache_algorithms

Cache size

- **Efficiency also depends on the cache size**
  - Important non linear behaviour
    - Cache is useless below a minimum size (only misses)
    - Cache is useless beyond a maximum size (hits/miss ration does not improve anymore)
- **Example for a total storage of 25 TB:**



Additional complications

- **If the cache is “volatile”, write operation must be flushed to the “slow” memory**
  - Synchonously
  - Asyncrhonously, Scheduled, but within a max amount of time
    - Thread pool with fixed number of thread (constant throughput)
    - Thread pool with dynamically changing number of thread (max amount of time)
    - This affects reliability of the service
- **Dynamically changing the algorithm**
  - Depending on the length of the read and write queues
- **There may be a fixed cost per I/O operation (ex: tape mount, seek & rewind)**
  - Cache algorithm depends on the “size” of the I/O requests
  - Aggregation of small I/O requests

How to tune the cache ?

1. **Start from some theoretical model, which will define the algorithm**
2. **Start your system and measure in real time hits, misses and latencies**
3. **Measure the sensitivity of the caching efficiency towards all adjustable parameters (cache size, cache algorithm, requests size, etc)**
4. **Tune the parameters to the desired compromise between efficiency and cost**

- This is an difficult, multi goal, non linear problem

Agenda

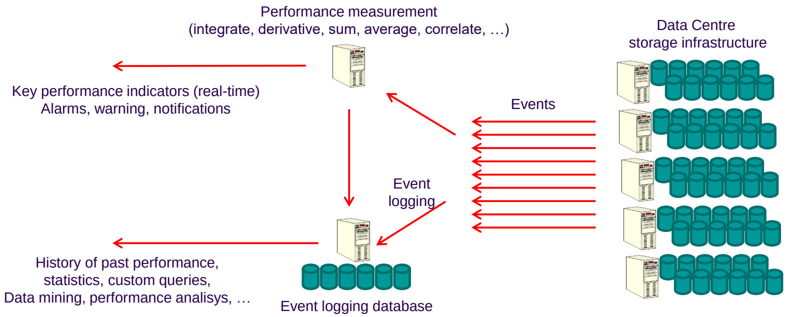
Monitoring and Performances

- **Essential component of a Data Management infrastructure**
- **A mean to measure the key performance indicators of the service.**
- **Several parameters need to be monitored, one for every potential bottleneck of the service**
  - Network Performance, Memory and CPU load of every server
  - I/O rates for every disks
  - Popularity of every file, to identify hot spots
  - In HSM or in front of a cache: hit and miss analysis
  - I/O rates per user
  - Length of the pending request queue
  - Access latency
- **Multiple views**
  - Service viewpoint: Aggregated performance, statistics, global indicators
  - User viewpoint: Performance's measured from the user's perspective, queues waiting time, latency, throughputs, …

Alarms

- **Monitoring data must be aggregated, in order to provide, in real time, the maximum, the minimum, the average, the current value in time.**
- **All the aggregated values must be constantly compared with "Service Level Agreement" parameters and linked to the Alarm system.**

Monitoring and Alarms



Agenda

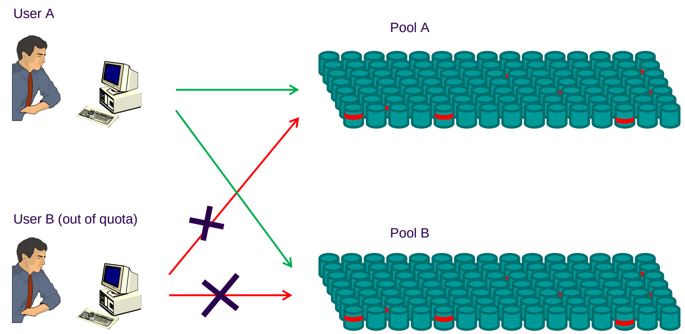
- Introduction to data management
  - Data Workflows in scientific computing
  - Storage Models
- Data management components
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Analytics
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- Summary

Quota

- Quota is a limit set by an administrator on the usage of (storage) resources
- Resources can be “Storage Space”, “Transfer bandwidth”, “Requests per seconds”, …
- The quota can be applied on “Accounts”, “Volumes”, “Pools”, “Directory”, ...
- The quota can be “Hard” of “Soft”
- Quotas can be mapped to physical resources or arbitrary limits independent of physical resources
- Tools are necessary to manage quota

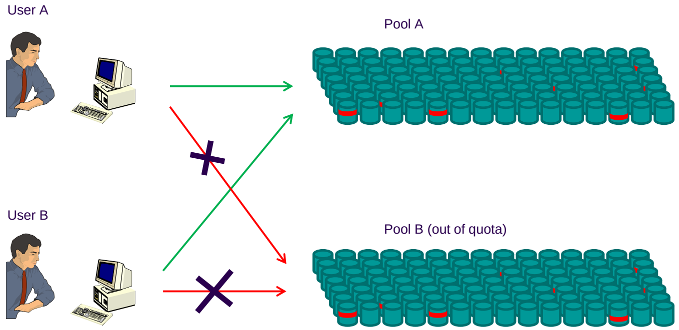
Quota per user

- User B cannot write to pool B because he owns a large file in pool A



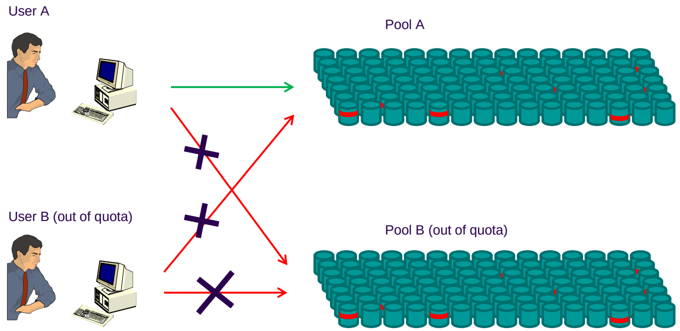
Quota per pool

- User B cannot write to pool B because User A has put a large file in it



Quota per user and per pool

- All is possible, more difficult for end-user to understand



Hard and soft quotas

- In hard mode, the quota verification is done synchronously with the request. Another operation "in the line of fire"
- With hard quota the system is guaranteed to be consistent and it is impossible for a user to abuse the available resources
- Soft quota allows resource accounting to be calculated asynchronously, allowing increased performances

Quotas and physical resources

- Can be linked
  - In this scenario quota is a mechanism to guarantee the availability of resources to a particular owner, avoiding any possible shared resources
  - Unused resources cannot be reused. Inefficient use of resources
  - Equivalent to a "resource reservation" implementation
- Can be independent
  - Quota is a mechanism to "control" usage and keep it under a reasonable level
  - The sum of quotas allocated can be orders of magnitude higher than the resources available
  - Potential "overbooking" problem
    - Overbooking is not necessarily a problem, provided that procedures are defined to handle it

Summary

- **Several components, many of them independent**
  - Name Servers and databases
  - Data Access protocols
  - Reliability
    - Availability
  - Access Control and Security
    - Cryptography
    - Authentication, Authorization, Accounting
  - Scalability
    - Cloud storage
    - Block storage
  - Data Replication
  - Data Caching
  - Monitoring, Alarms
  - Quota
- **Allow to build an architecture to transform “storage” into “data management services”**