DSS

Data & Storage Services

CERN IT Department

Handling Big Data

an overview of mass storage technologies

Łukasz Janyst

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

GridKA School 2013
Karlsruhe, 26.08.2013

CERN

What is Big Data?

A buzzword typically used to describe data sets that are too big to be stored and processed by conventional means.

What can we do with it?

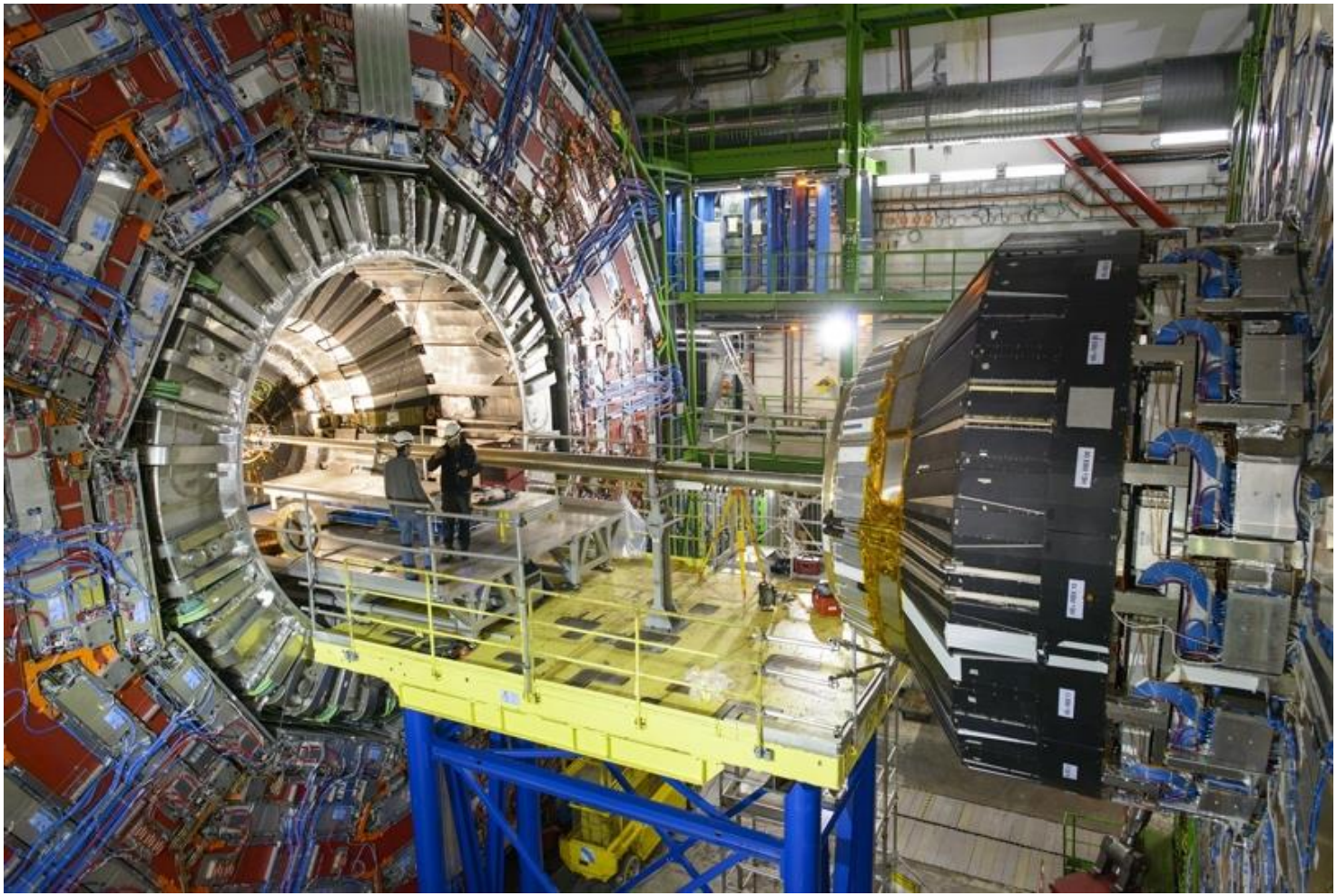
- Analyze anonymous GPS records from 100 million drivers to help home buyers determine optimal property locations
- Analyze billions of credit card transactions to protect from fraud
- Find trends in the stock market moves
- Decode human genome

What can we do with it?

Copy, store, and analyze the internet traffic for more or less questionable reasons

The NSA's Data Center in Utah - where all the PRISM data is supposedly handled

What can we do with it?



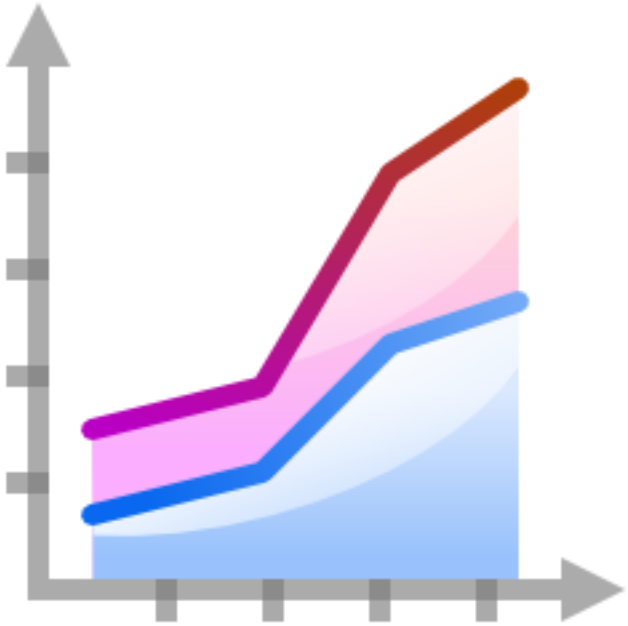
Process data from over 150 million sensors to find the Higgs boson

How big is it now?

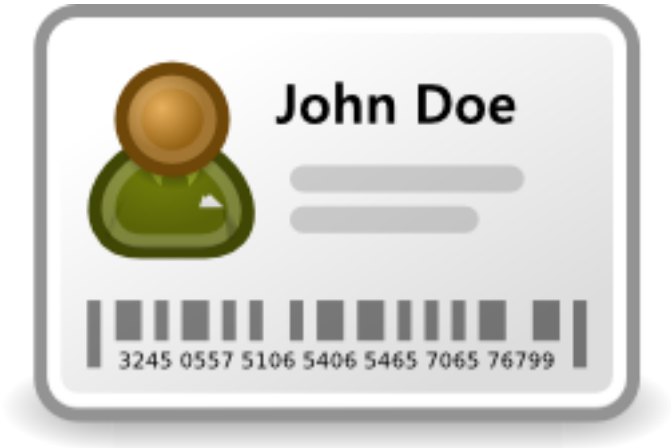
- CERN alone currently stores over 100 petabytes of data, with the experiments producing around 30 PB annually
- Facebook stores around 300 billion photos
- Walmart processes 1 million client transactions per hour and has 2.5 PB
- NSA builds a data center capable of handling 12 exabytes of data

How big is it going to be?

International Data Corporation forecasts the digital universe to grow up to 40ZB (40 trillion gigabytes) by 2020.



Grow by 50% each year



5200 GB/per person in 2020

What are the challenges?

Capture

Store

Transmit

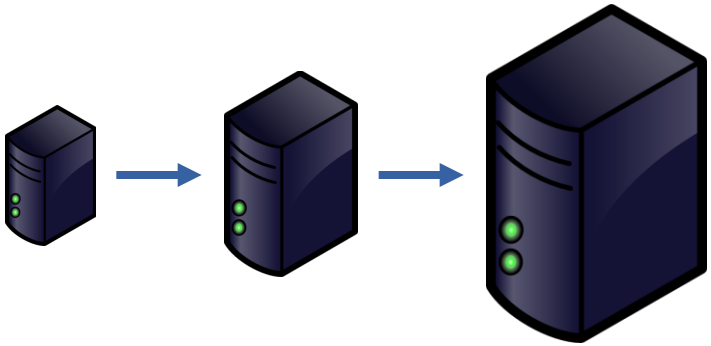
Process

Scope of this presentation

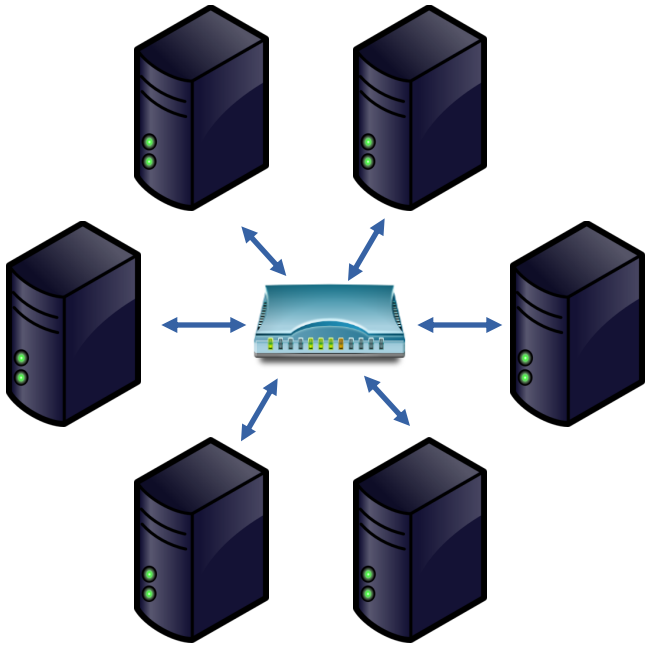
Multitude of solutions

Scaling

Storage systems need to be able to grow with the growing amount of data they handle.



Scaling up



Scaling out

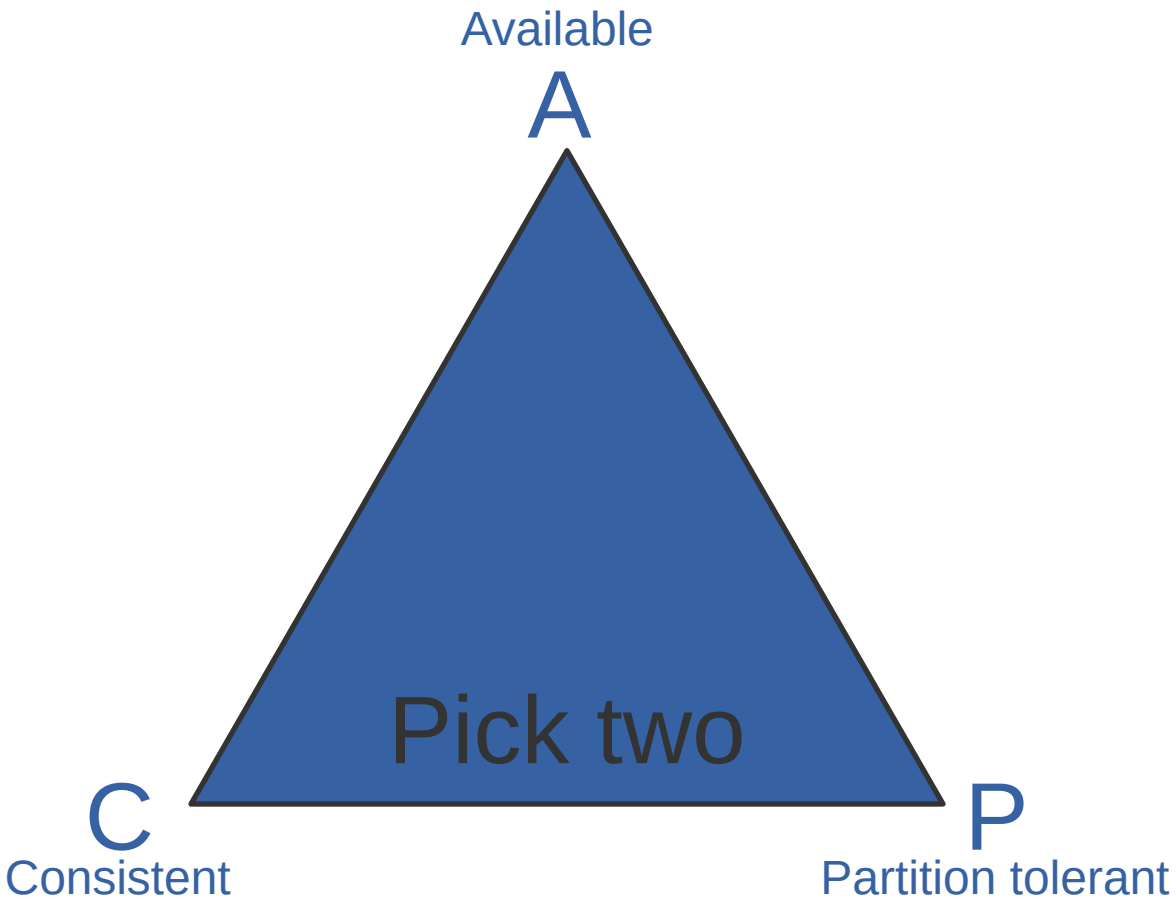
Ideal properties

Ideally all distributed systems should be:

- Consistent
  - commits are atomic across the entire system, all clients see the same data at the same time
- (Highly) Available
  - remains operational at all times, requests are always answered (successfully or otherwise)
- Tolerant to partitions
  - network failures don’t cause inconsistencies, the system continues to operate correctly despite part of it being unreachable

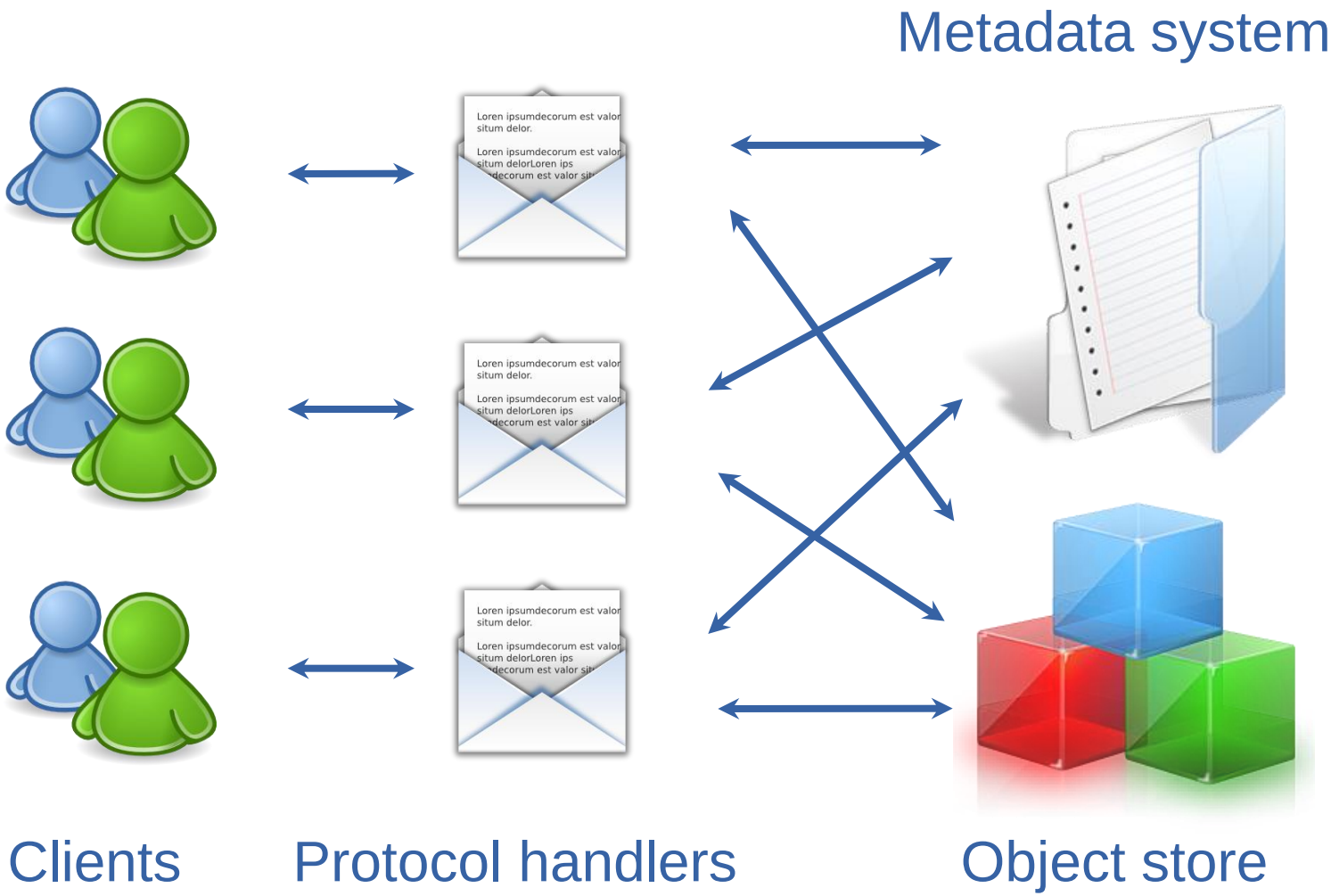
Ideal properties - CAP

In reality however:



Brewer's CAP theorem

Typical components



Caveat: not necessarily logically separate - may be tightly coupled and interleaved

Object stores

10c39527b893c798a93e8997772f65a8

(Hashed) key

Data Blob

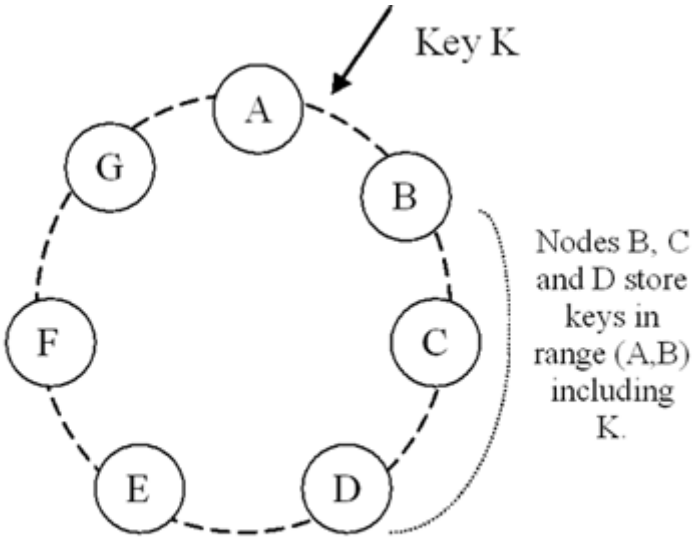
Distributed Object Store - typically, a collection of uncorrelated flexible-sized data containers (objects) spread across multiple data servers

Object-node mapping

- Algorithmic
  - object location can be computed by the client or server using object name (key) and other inputs (cluster state)
  - Dynamo, CEPH
- Manager/Cache
  - manager node asks storage nodes for an object and caches the location for future reference (XRootD)
- Index
  - central entity (database) knows all the objects and their locations - most of “traditional” storage systems

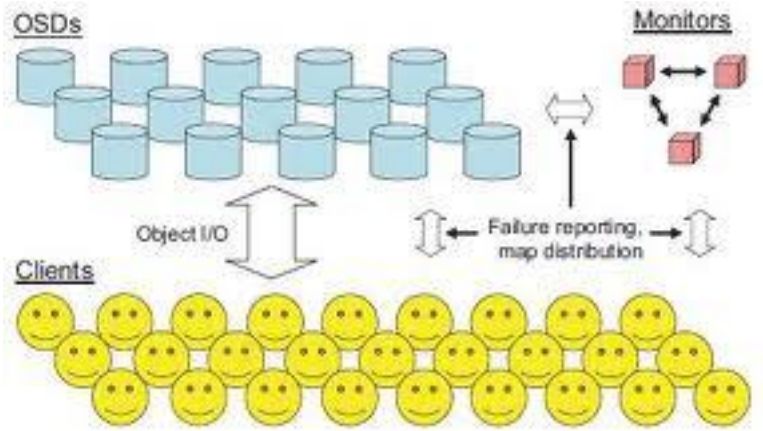
Amazon Dynamo

- The output space of the hash function is treated like a ring
- A node is assigned a random value denoting it's position in the ring
- An object is assigned to a node by hashing the key and walking the ring clockwise to find a node with a position larger than the key.
- Replicas are stored to the subsequent nodes



CEPH - RADOS

- Each object is first mapped to a placement group depending on the key and replication level



- Placement groups are assigned to nodes and disks using a stable, pseudo random mapping algorithm depending on cluster map (CRUSH).

- Cluster map is managed by monitors and replicated to storage nodes and clients.

Chunks, stripes, replicas

For performance, space and safety reasons, the data may be distributed in many different ways

- Replicas
  - fairly simple, little metadata, performance
  - space issues: knapsack problem, expensive for archiving
- Chunks
  - solves the knapsack problem, distributes the load
  - still requires replicating for safety, much more metadata
- Stripes
  - relatively cheap archiving
  - more metadata, knapsack problem

RAIN - Erasure codes

- RAIN - redundant array of inexpensive nodes (RAID implementation across nodes instead of disks)

- Used to increase fault tolerance by adding extra stripes correlating the info contained in the base stripes. Multiple techniques:

  - Hamming parity
  - Reed-Solomon error correction
  - Low-density parity-check

System topology

Data placement needs to take into account system topology.

- Spread replicas/chunks/stripes between failure domains:
  - Different disks, nodes, racks, switches, power supplies, or entire data centers if possible
- There is even some research on reducing heat production by appropriately scheduling disk writes.

Data locality

- Computation is most efficient when executed close to data it operates on
- Core concept of Hadoop, where nodes are typically both storage and computation nodes
- HDFS exposes interfaces allowing job schedulers to dispatch jobs close to data: often the same node or rack

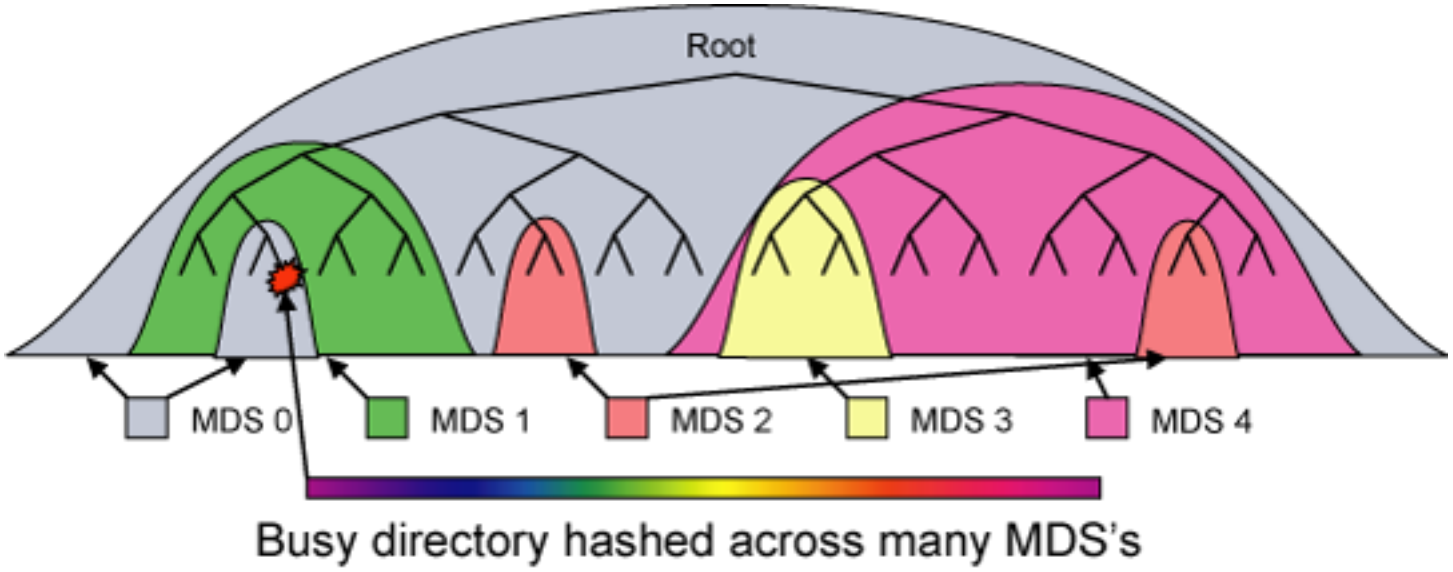
Metadata services

Group and organize objects into human-browsable groups, manage quotas, ownership, group attributes...

- POSIX-like trees
  - familiar, used since decades
  - very hard to scale out

- Accounts/Containers/Objects
  - trivially scalable
  - may be hard to adjust legacy software

CEPH Filesystem



- Runs on top of RADOS
- Maps files and directories hierarchies to RADOS objects
- Does dynamic tree partitioning
- Metadata cluster may grow or contract - nodes are stateless facades for accessing data in RADOS

Amazon S3 approach

- Proprietary technology
- Most likely it's Dynamo with:
  - HTTP interface
  - accounting system for billing
  - user authentication/authorization mechanisms
- User accounts consist of buckets
- Buckets are sets of files
- account-bucket-file tuples are likely used as keys of Dynamo objects

Backups-Archiving

Some data may need to be moved to cheaper or more reliable media.

- Back up - copy important data to a different kind of media - cheaper, more resilient to some natural phenomena
- Archive - move inactive data to a cheaper but safer and possibly less available system

Backups and archives of big data are likely even bigger data!

HSM and Tiers

- Hierarchical Storage Manager - transparently move data files between media types depending on how soon and how often they are accessed
- Tier Storage - assigning different categories of data (more/less critical, active/inactive, ...) to different kind of storage technologies, often manually

Clients

- APIs
  - direct use
  - integrating into commonly used tools as plug-ins

- Mount points
  - through widespread protocols (NFS, CIFS/Samba, ...)
  - dedicated drivers (typically FUSE)

- Commandline and GUIs
  - through widespread software (web browsers)
  - custom tools

Access requirements

- User authentication
  - is the system exposed to multiple users?
  - X.509, Kerberos, user/password, etc.
- Transmission encryption
  - are the channels secure or data sensitive
  - symmetric/asymmetric

- Access patterns
  - Is put/get enough?
  - Do we need partial reads, vector reads?
  - What about updates?
- Filesystem/bucket operations
  - list, stat, chown, etc.

Efficiency considerations

- Latency
  - support for logical streams and priorities
  - allow for multiple queries at once and provide a way of disambiguating responses
- Bandwidth
  - protocol overhead
  - compression (both headers and payload)

- Server-side CPU intensiveness
  - Do requests need to be decompressed?
  - Does it need to parse a ton of text/XML?

HTTP

- HTTP is indisputable king of the cloud communication protocols
  - not because it's particularly efficient, but because clients are built into pretty much every computer
- There's problems with it, mainly:
  - does not allow out-of-order or interleaved responses
    - reasonable performance only for big, one-shot downloads
  - protocol overhead:
    - many headers sent with each request, most of which are redundant

HTTP 2.0 (a.k.a SPDY)

An effort to fix the most important issues with HTTP

- SPDY kind of a virtual transport protocol for HTTP.
  - It does not really change the request or response format, just the way they are transported over the wire
- Prioritization and multiplexing
  - introduce multiple logical streams within one connection
  - responses may interleave each other
- Header dictionary
  - only changing headers are sent over the wire and they are compressed

Outlook

Prediction is very difficult, especially about the future.

- Niels Bohr

- Need for handling bigger and bigger data will likely push out POSIX completely
  - write-once read-many, put/get/remove
  - the main cost is in moving applications to use the new semantics
- Metadata services are bottlenecks
  - likely replaced by deterministic data placement

Questions

Thanks for your attention!

Questions? Comments?

Notes

- Some interesting reading:
  - Dynamo: Amazon's Highly Available Key-value Store
  - RADOS: A Scalable, Reliable Storage Service for Petabyte-scale Storage Clusters
- Most of the artwork in this presentation comes from: Open Icon Library