Big Data Bases - Technologies and Applications

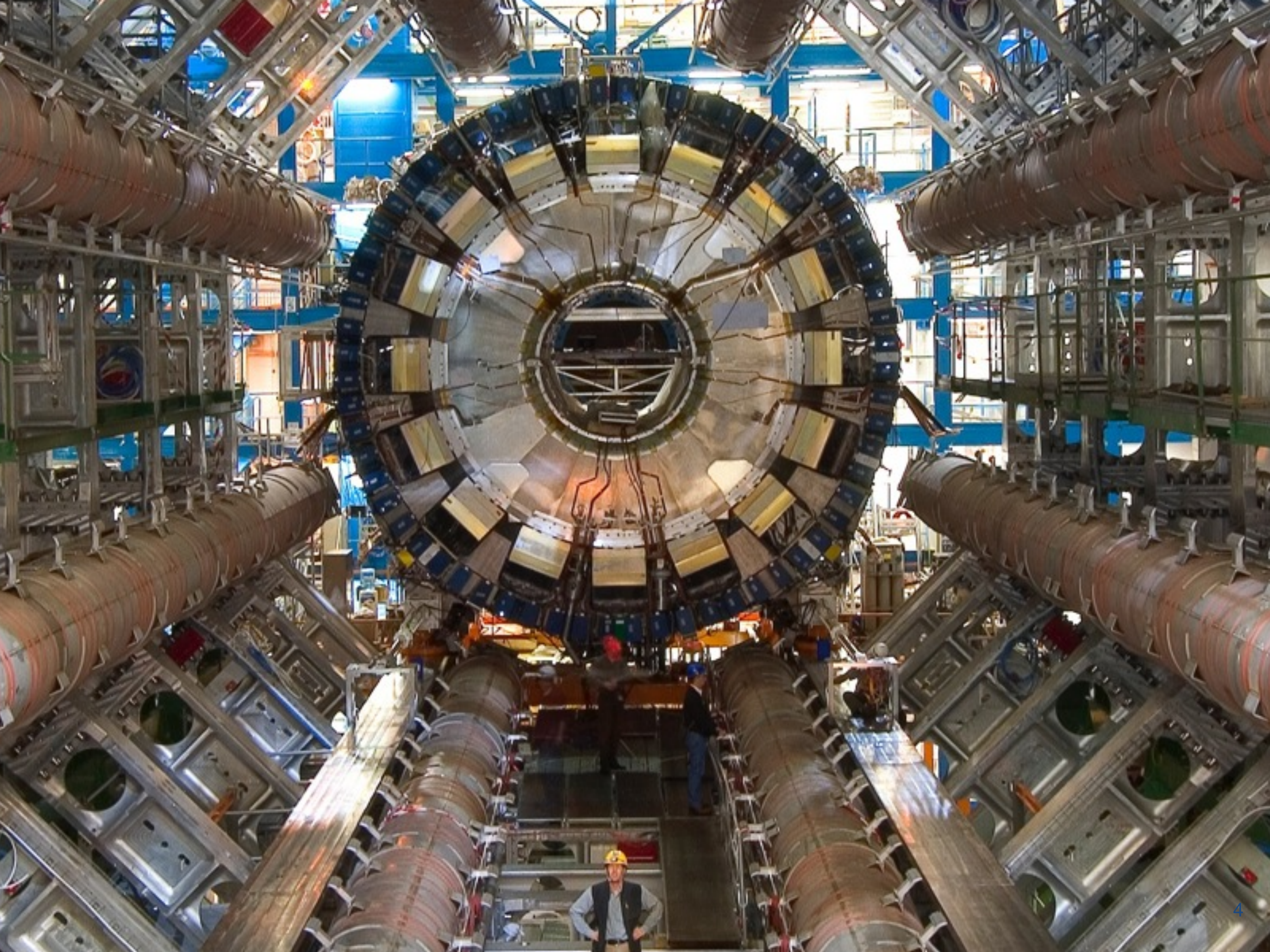
Dirk Duellmann, CERN

Outline

- Big Data - more than a buzzword?
  - new market for methods used in science since decades (eg analytics)
  - but - also new methods, which can be applied in science
- Storage media developments
- New approaches and technologies

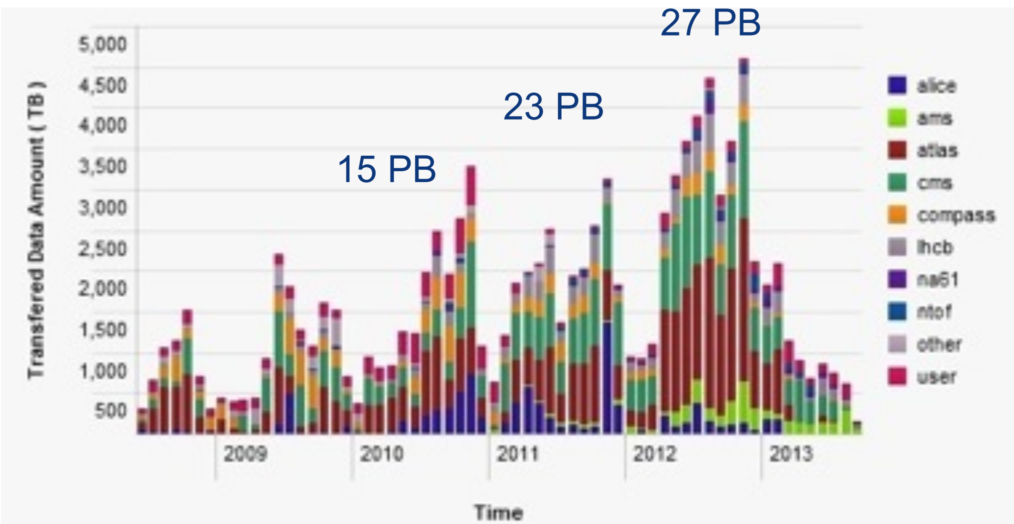
Big Data

- An estimated 35 zettabytes will be stored by 2020 (worldwide)
    - growing exponentially
- Why? Because…
    - …it is technically possible
        - Moore's & Kryder's law
    - …it is commercially relevant
        - data volume is proportional to budget
        - many new digital service providers
        - and foremost(?) digital marketing

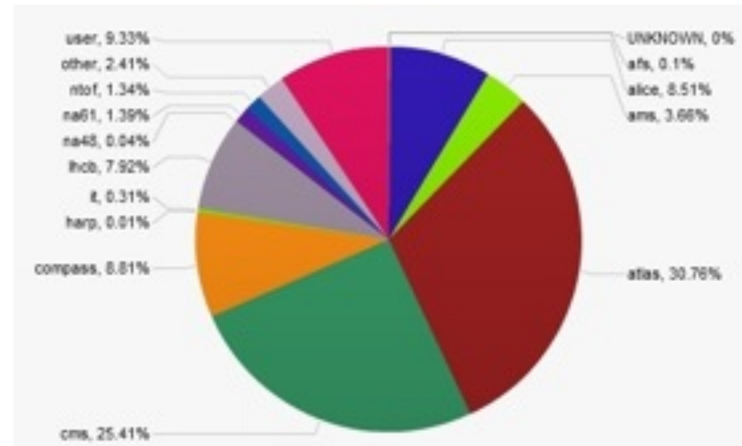


Data 2008-2013

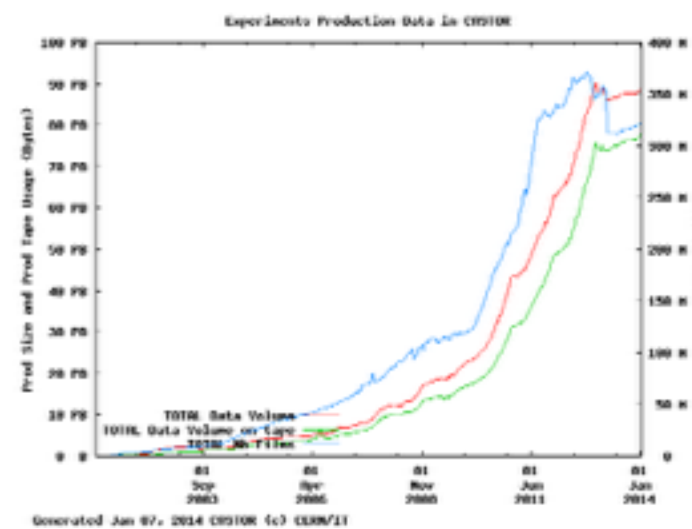
CERN Tape Writes



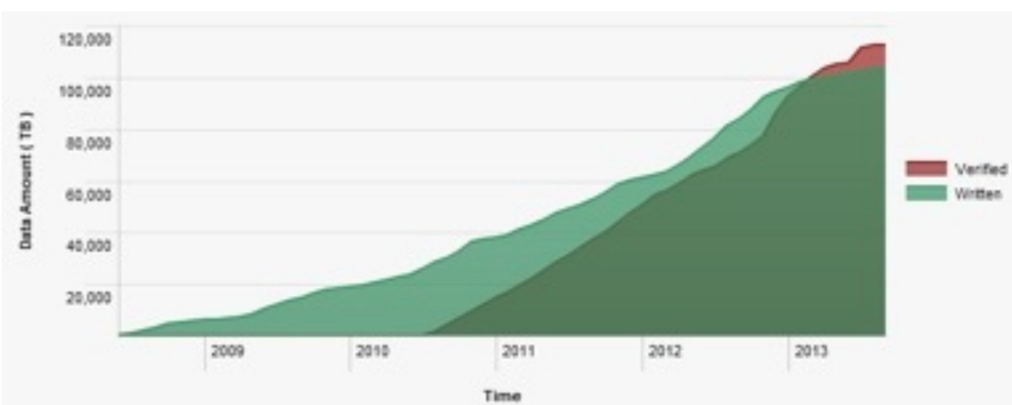
Tape Usage Breakdown



CERN Tape Archive

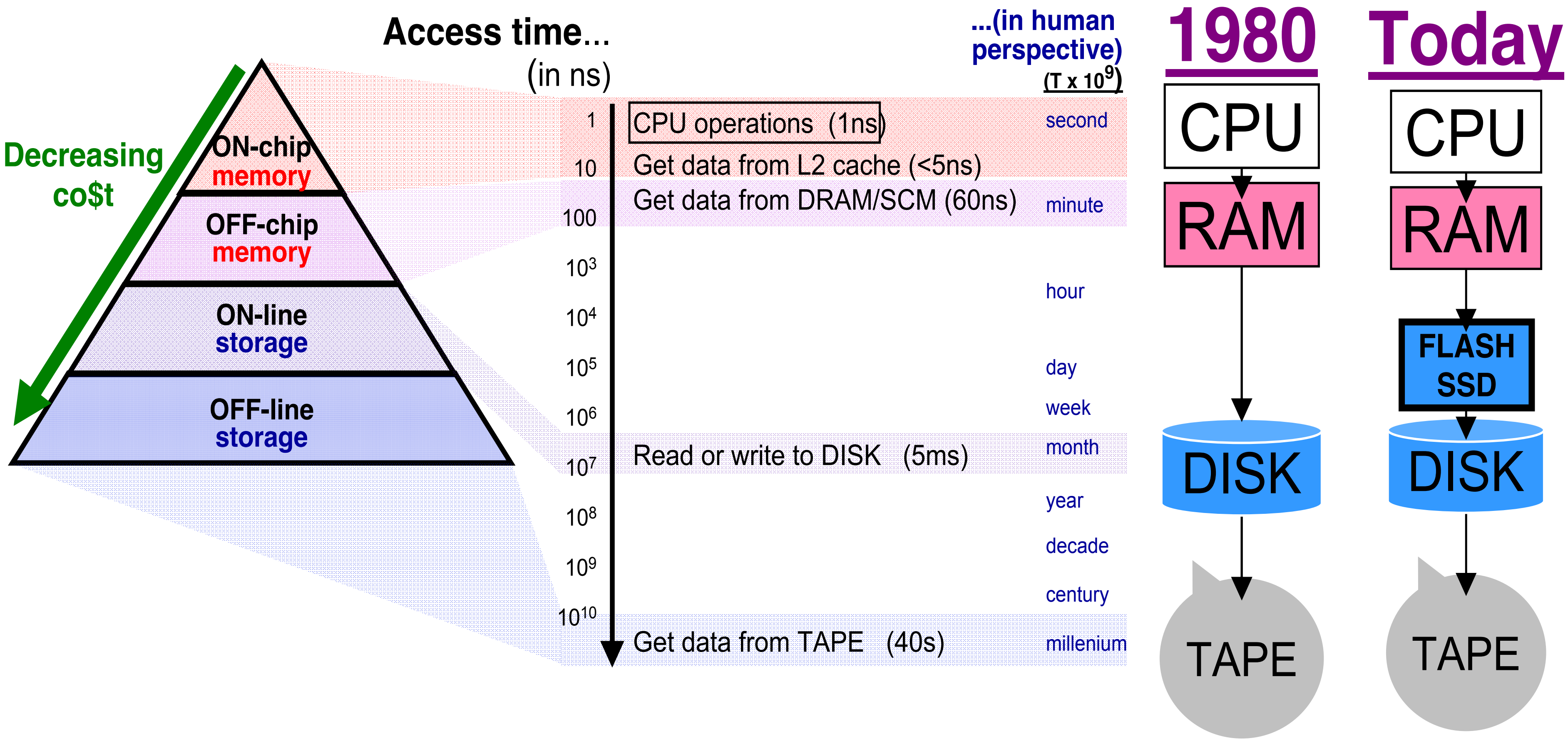


CERN Tape Verification



Data Loss: ~65 GB over 69 tapes
Duration: ~2.5 years

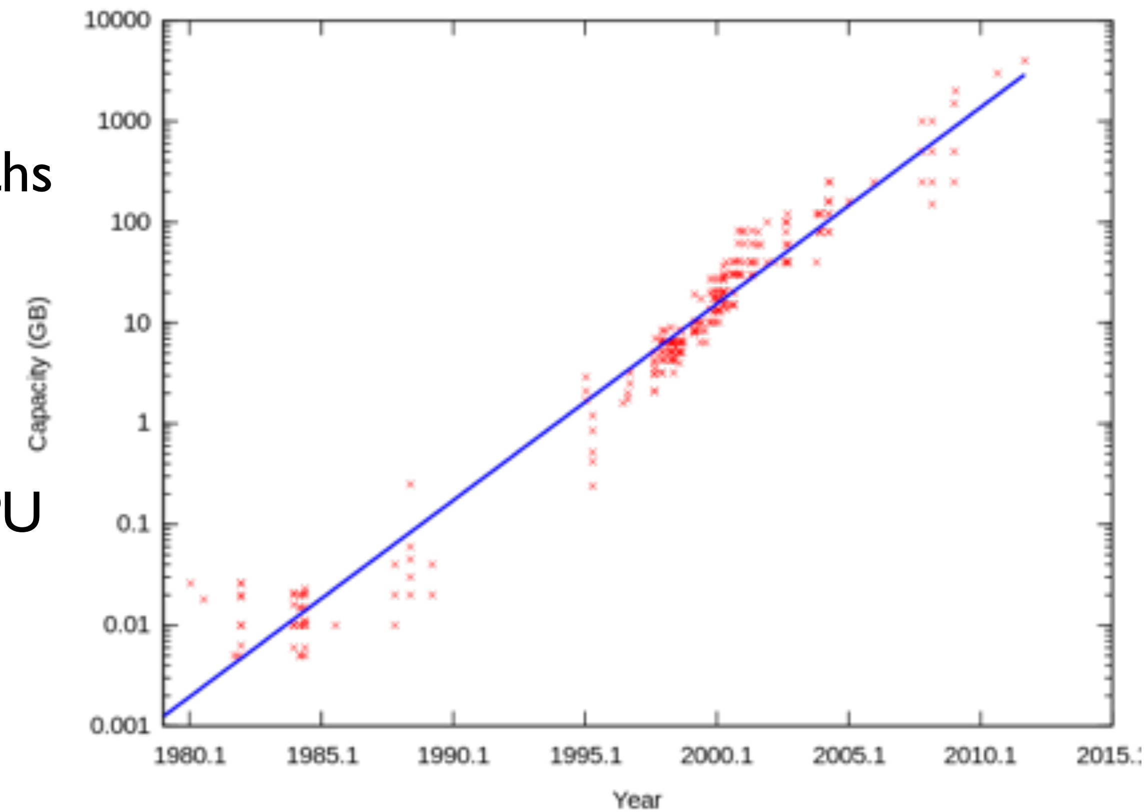
Storage Media Hierarchy



picture adapted from: "Storage class memory", IBM Almaden research centre, 2013

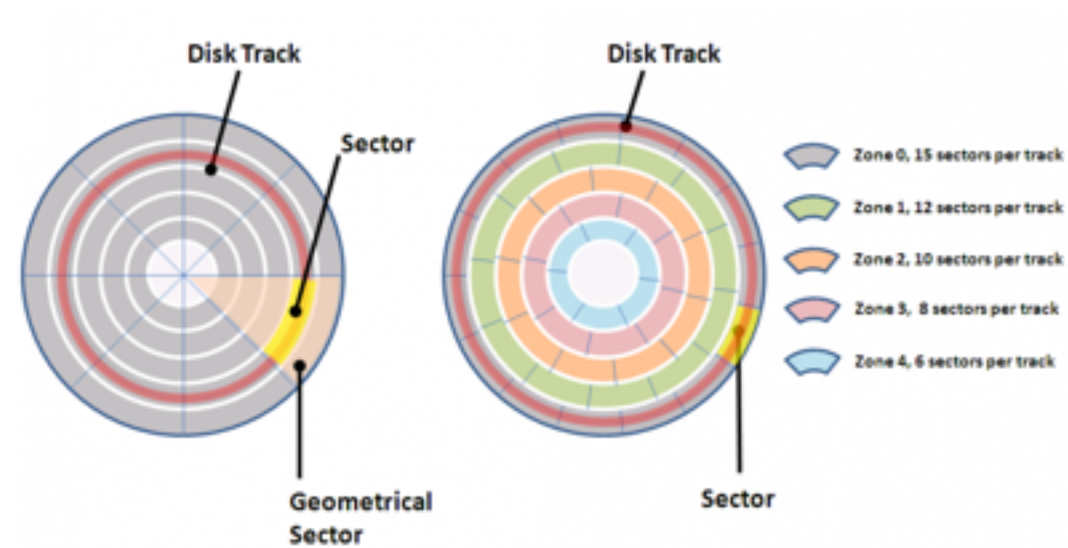
Magnetic Disk

- Kryder's "law" (observation)
  - magnetic disk areal storage density doubles every 13 months
  - compare to Moore's "law": silicon performance doubles "only" every 18 months
- Storage volume outperformed CPU
  - in other words: stored data volume is "cooling down"
  - finding relevant data is getting more important / difficult



Volume and IOPS

- Storage access time is governed mainly by two components
  - seek time - positioning time of the read head
    - eg 3-10 ms (average)
  - rotational delay of the disk
    - eg 7200rpm disk: 4.2 ms
- Both evolved due to mechanical constraints only within a “small” range - O(10)
- …only storage density has been growing exponentially.



Power Consumption

- Storage systems account often for 40% of power consumption
  - magnetic disks have improved, but still show relatively low power efficiency (defined as: power consumed per work done)

- empirically:

$$\text{Power} \approx \text{Diameter}^{4.6} \times \text{RPM}^{2.8} \times \text{Number of platters}$$

=> disks shrink and don’t increase in rotational speed

Sequential vs random access

- How does the simple mechanics of rotating disks affect different access patterns?
  - read time = seek time + rotational latency + transfer time
    - sequential: few seeks and rotational waits with long transfers
    - random: one seek and wait per I/O => O(10-100) slower

The secret to making disks fast is to treat them like tape
(John Ousterhout)

Tape is Dead, Disk is Tape, Flash is Disk, RAM Locality is King
(Jim Gray)

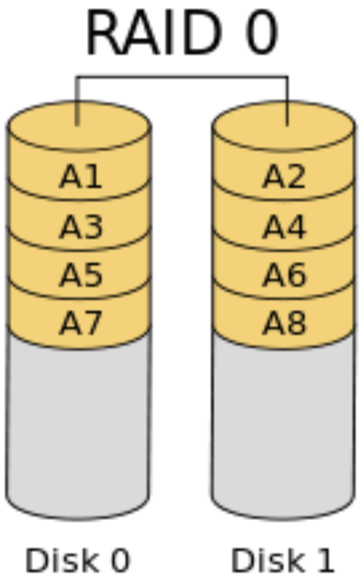
- Gap between sequential and random access is large and increasing with density
  - many concurrent sequential clients sharing storage create random pattern
- For many database and analysis applications only the lower random rate (or IOPS) is relevant
  - and single client benchmarks fail to deliver good performance estimates

Media Aggregation

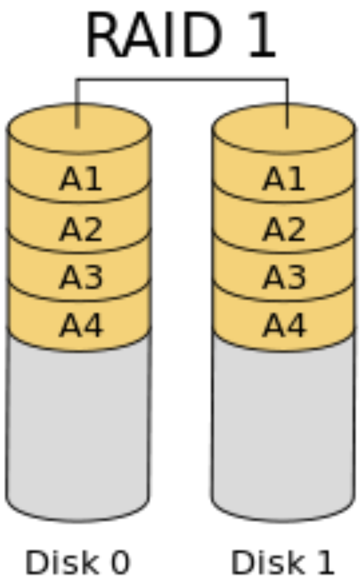
- Goals:
  - virtualise / cluster / federate many individual drive units into a single larger logical unit
  - provide more performance than a single drive
  - provide a higher reliability than the one of a single unit
- Redundant Array of Inexpensive Disks (RAID)
  - sometimes inexpensive => independent
  - initially implemented in dedicated disk controllers and disk arrays - later in software

(Simple) RAID Levels

- RAID 0 - Striping (to n stripes)
  - *failure rate r* and capacity c unchanged
  - potentially: n • disk throughput
  - fault tolerance: none



- RAID 1 - Mirroring (to n copies)
  - failure rate = $1-(1-r)^n$
    (*assuming independence!*)
  - capacity = 1/n • c
  - potentially: n • disk throughput
  - fault tolerance = n -1 drives



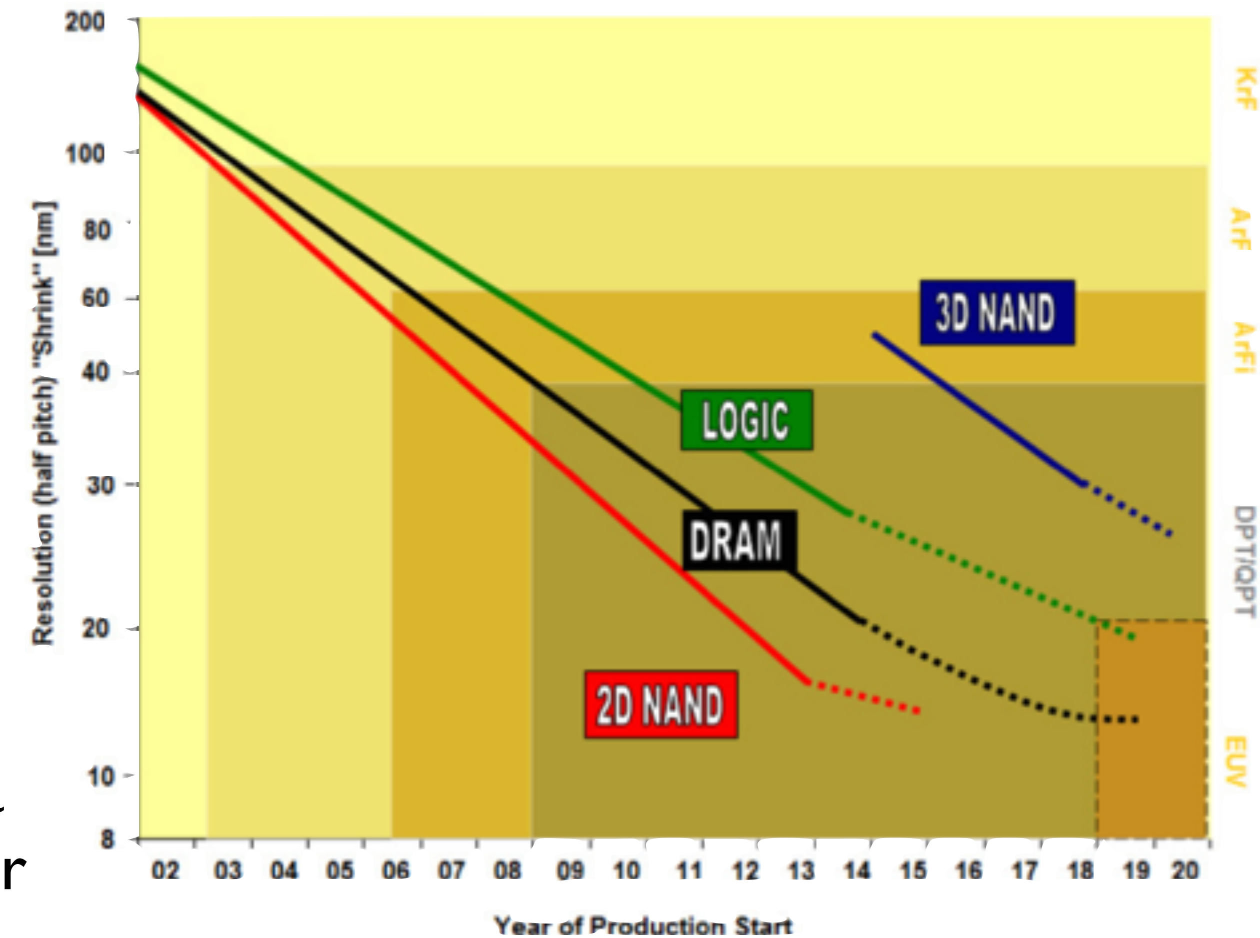
RAID Issues

- Assumption of independent drive errors does not hold
  - eg during recovery
  - drives often share also other common failure sources (power supplies, fans, network etc)
- Drive capacity increase and localised (=long) recovery result in probability for 2nd fault during recovery => data loss

- Most large scale systems departed from drive level RAID aggregation
  - but use similar concepts on a different level
    (eg file or chunk replication)

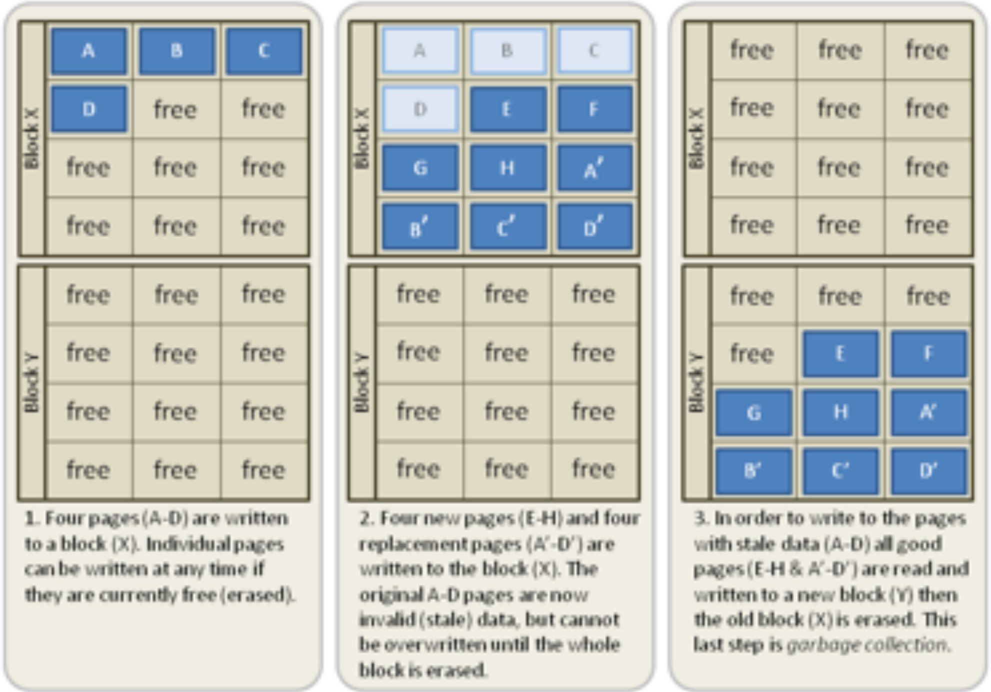
Flash: Basic Properties

- Density ~ Moore's law
  - no moving parts
  - power efficient
  - small form factor
- limited endurance
  - usually 5-100 k erase/write cycles
  - complex internal data management and wear levelling



Flash: undesired side-effects

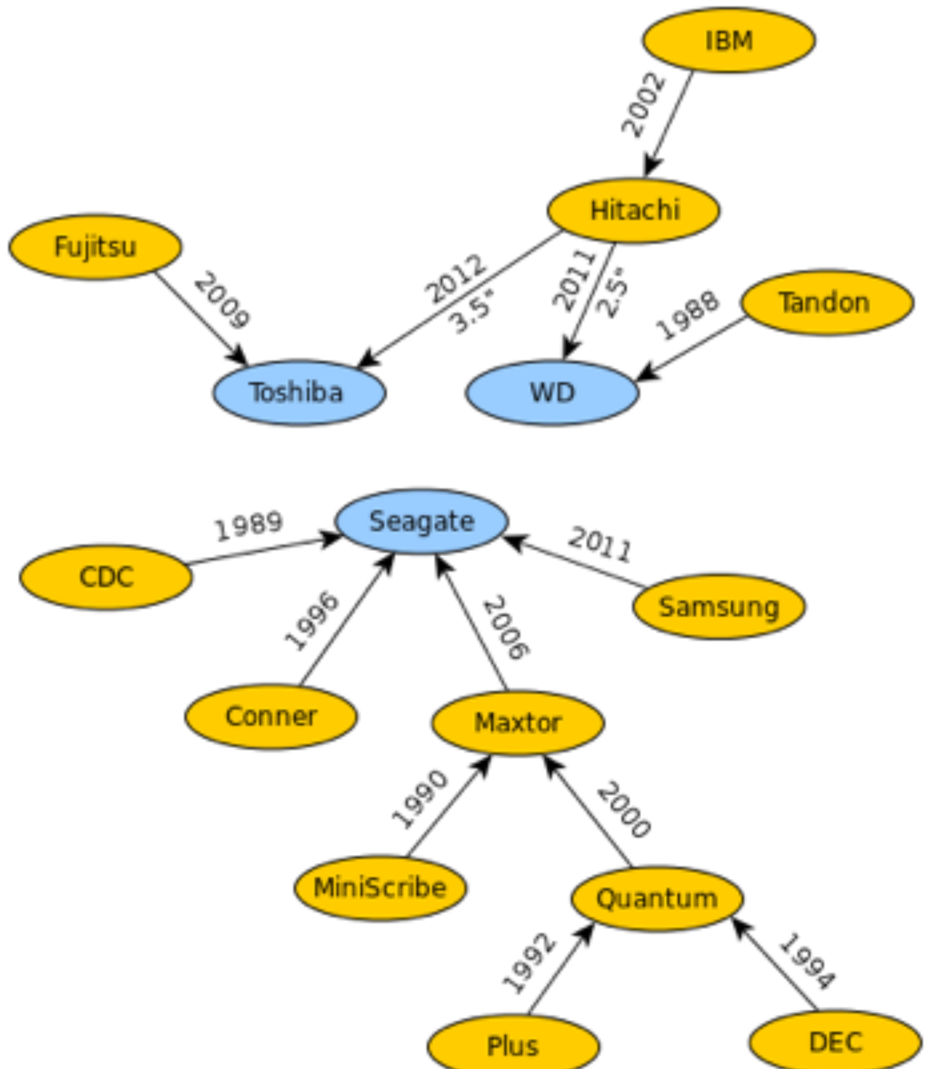
- asymmetric read/write performance
- write amplification : factor between user data and resulting flash memory changes
- block recycling : large internal trafic limits client transfers
- past writes influence future performance :
  eg benchmarks on new SSDs have only limited value
- limited durability (!= endurance)



SSD vs HDD

- SSD is less well defined and fragmented market
  - Large (factor 20) spread in performance and price
  - Several orders of magnitude more IOPS
    - current consumer SSDs reach 100k IOPS
  - Still O(10) higher price/GB
  - Better power efficiency - in particular for idle storage
- Still a niche solution in the data centre context
  - “Hot” transactional logs from databases or storage system metadata
- BUT - all the mobile market has gone to flash memory
  - and the magnetic disk market is consolidating…

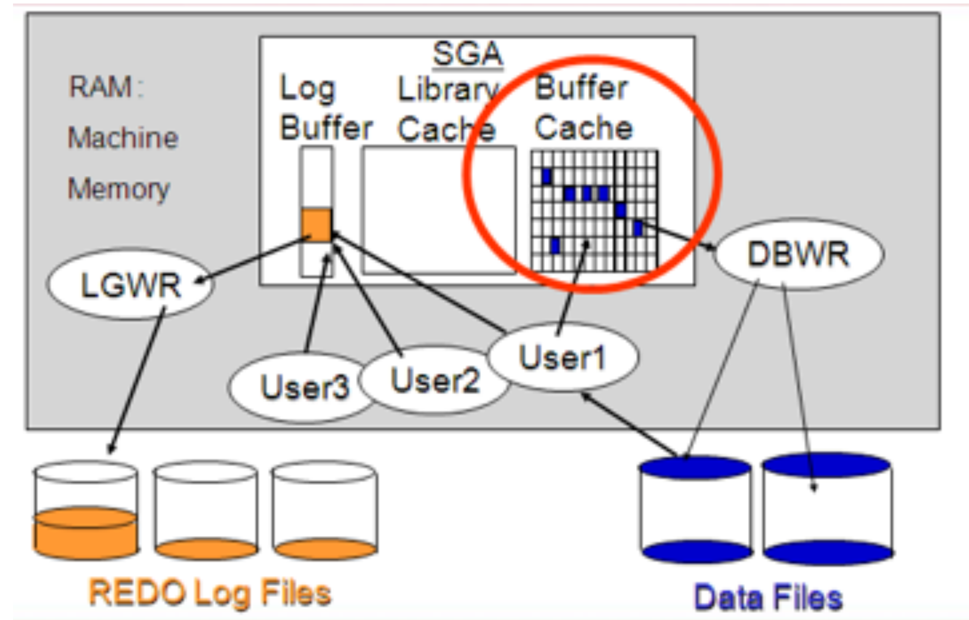
Disk Market Consolidation

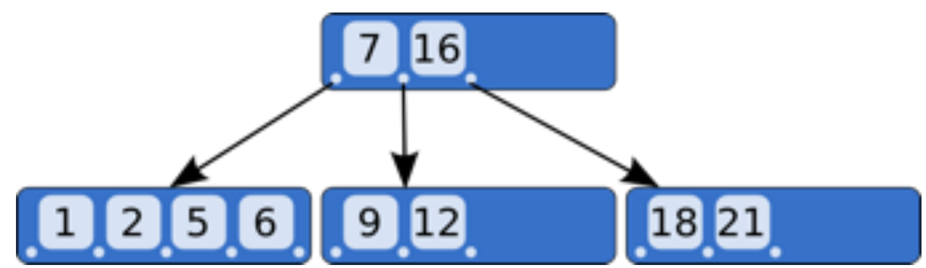


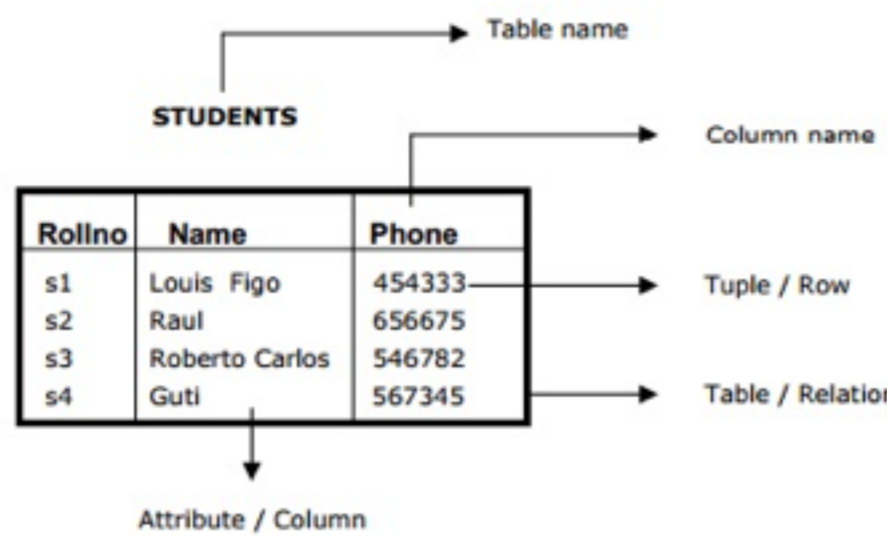
Relational Databases

- Consistent data changes for concurrent users
    - ACID transactions
- Indexed (fast) access to disk-resident data by key
    - eg Bayer-Trees (B-Trees)
- Structured Query Language
    - exploit the constraint tabular data model
    - generalised, logical development language

- All three main functions are under increasing pressure from simpler (= more specialised) solutions
    - ACID scaling & transactional development skills
    - Increased memory availability allows to access data much faster than B-Trees
    - Tabular model is too constraining for some applications or problems







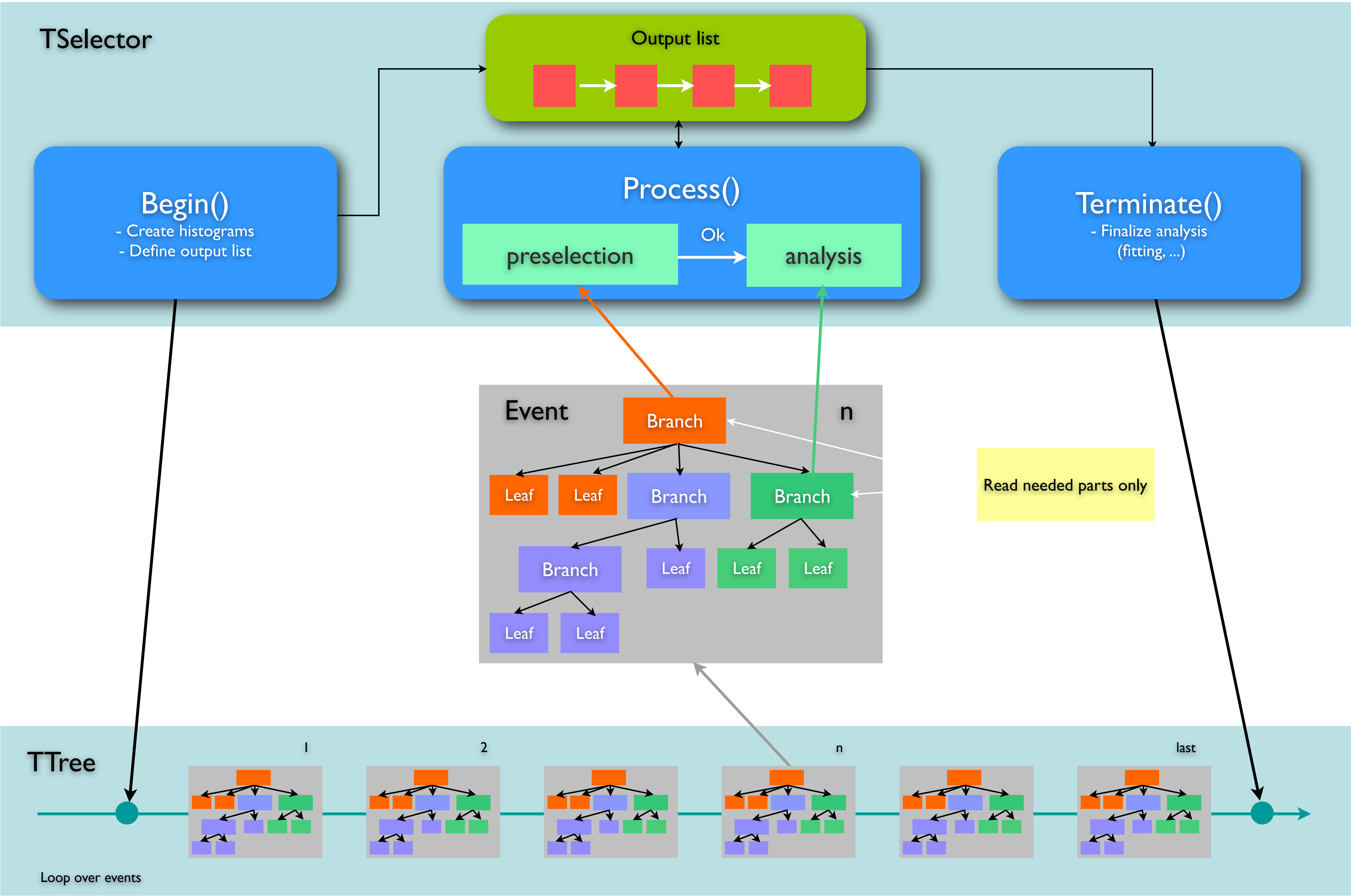
How to store/retrieve LHC data models? A short history…

- 1st Try - All data in an commercial Object Database (1995)
  - good match for complex data model and C++ language integration
  - used at PB scale for BaBar experiment at SLAC
  - but the market predicted by many analysts did not materialise!
- 2nd Try - All data in a relational DB - object relational mapping (1999)
  - Scale of PB deployment was far for from being proven
  - Users code in C++ and rejected data model definition in SQL
- Hybrid between RDBMS and structured files (from 2001 - today)
  - Relational DBs for transactional management of meta data (TB scale)
    - File/dataset meta data, conditions, calibration, provenance, work flow
    - via DB abstraction (plugins: Oracle, MySQL, SQLite, Frontier/SQUID)
    - see XLDB 2007 talk for details
- Home-grown persistency framework ROOT (180PB)
  - Uses C++ "introspection" to store/retrieve networks of C++ objects
  - Configurable column-store for efficient sparse reading

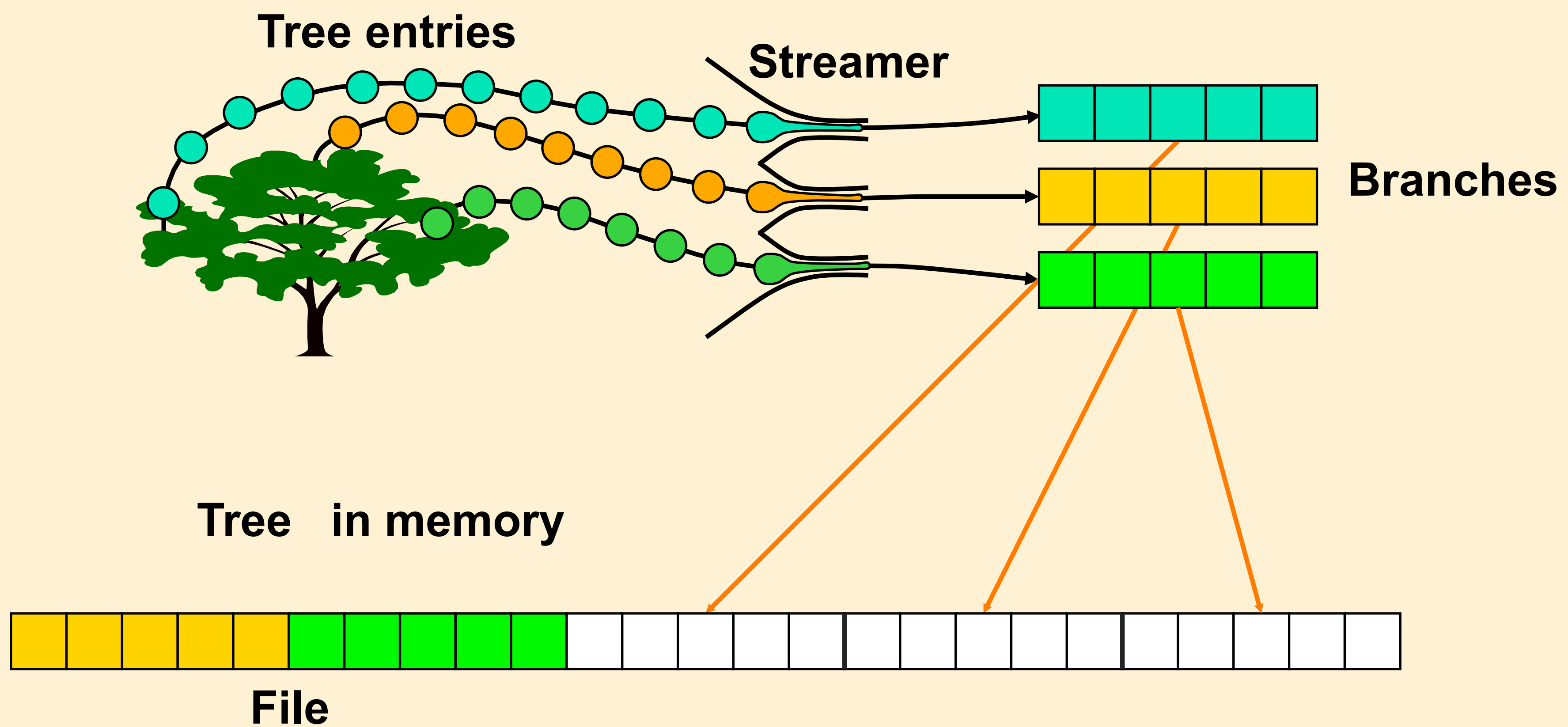
WLCG Worldwide LHC Computing Grid

LCG

Processing a TTree

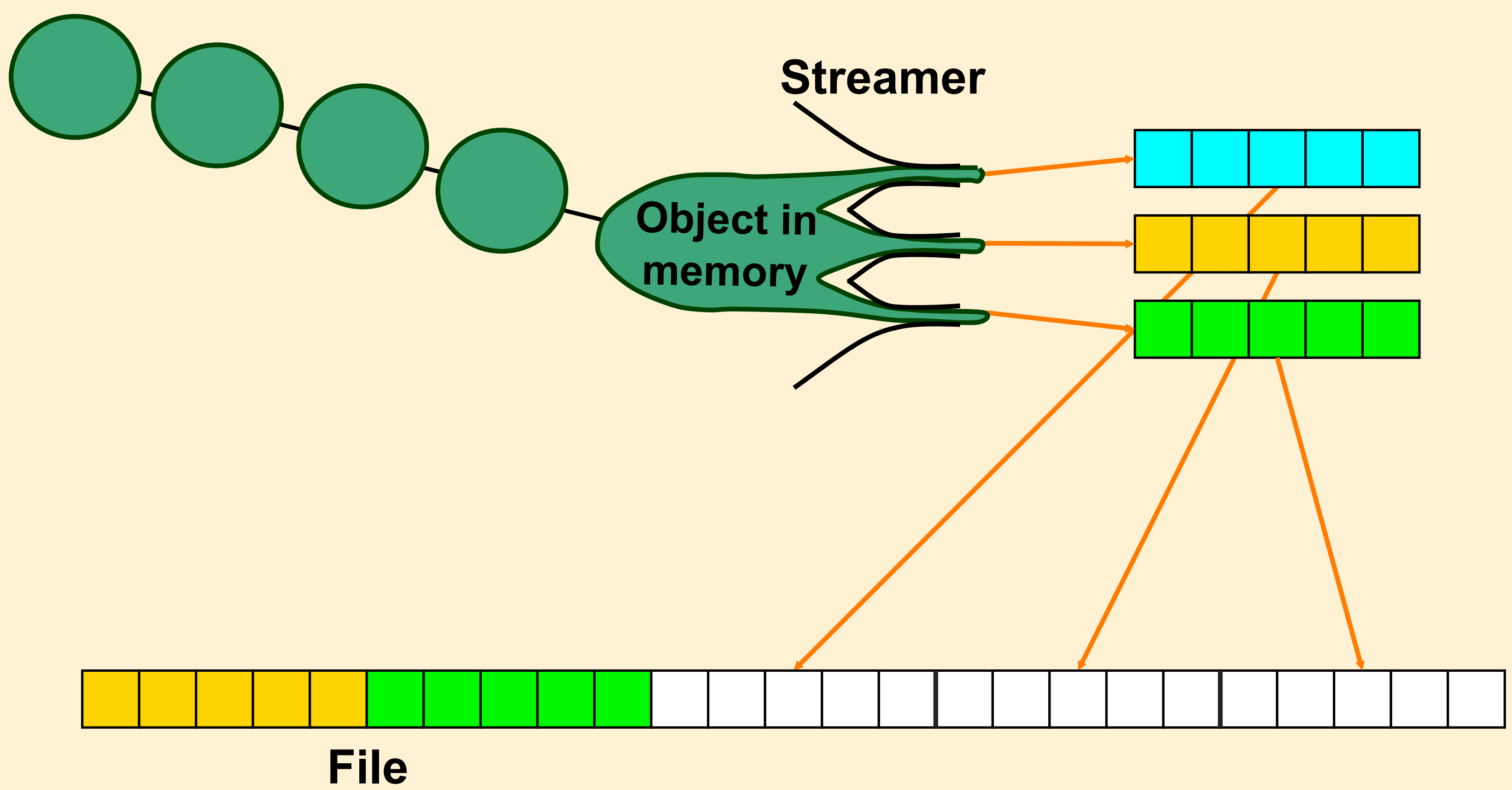


TTree - Clustering per Object



TTree = container for an arbitrary set of independent event trees

TTree - Clustering per Attribute



tuneable: mix of row, column storage is possible within an object tree

Michael Hausenblas - Chief Data Engineer @ MapR

in his bog at https://medium.com/large-scale-data-processing/3da34e59f123

[…]

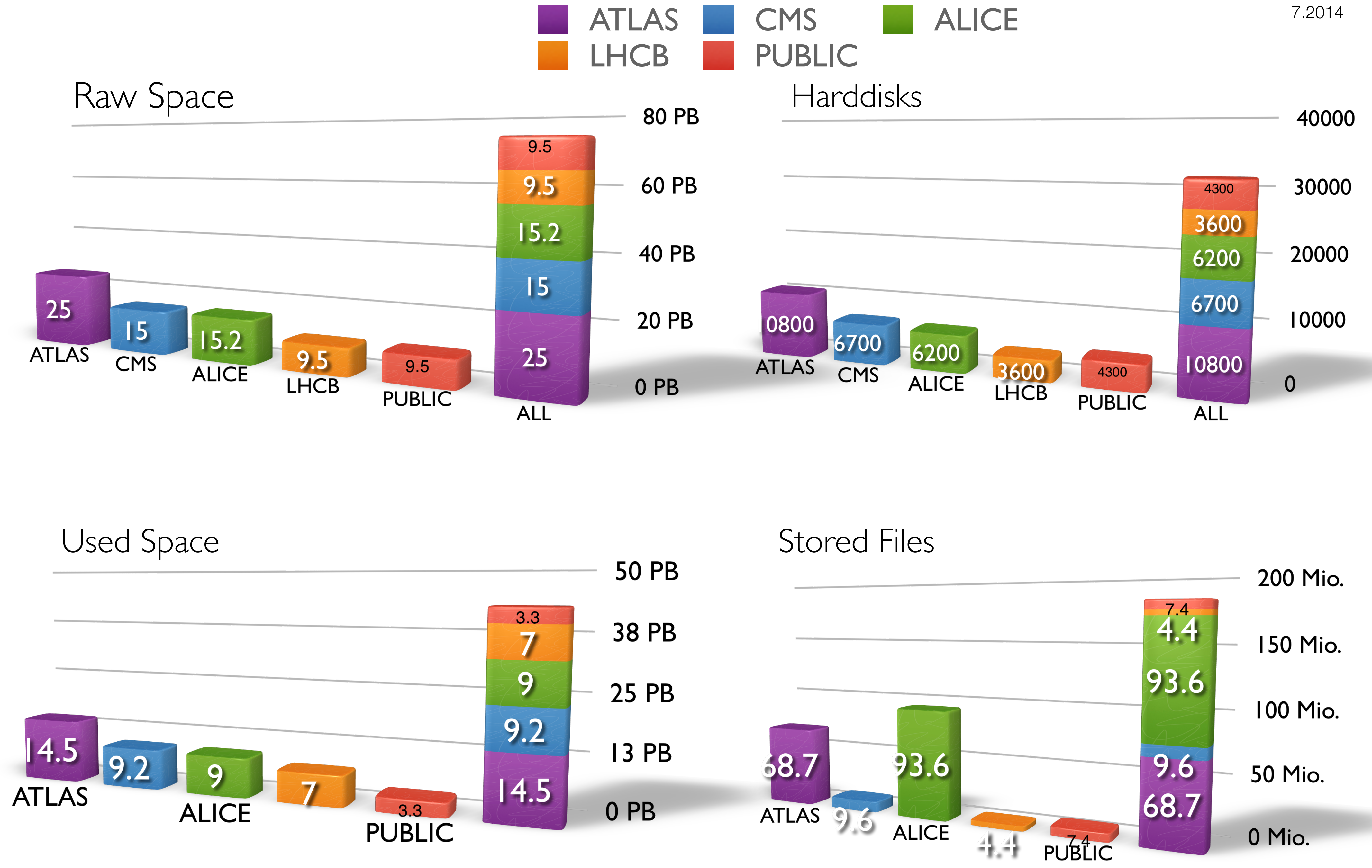
I was flabbergasted and went like: OMG, there is a group of people who have been doing this for almost 20 years now. While I think the Google engineers deserve the credits for the engineering innovations they introduced in their 2010 paper on Dremel I also believe Fons and his team deserve at least the same attention and credit.

[…]

EOS Deployment at CERN

CERN IT Department

7.2014



Deployment Simplifications and Development Targets

- Follow trend in many other large storage systems
  - server, controller, disk, file system failures need to be transparently absorbed by storage s/w
  - key functionality: file level replication and rebalancing
- Decouple h/w failures from data accessibility
  - data stays available (for some time at reduced performance) after a failure
  - this could change current approach wrt h/w lifecycle
- Fine grained redundancy options on top of a standardised h/w setup
  - eg choose redundancy level (and hence the storage overhead) for individual data rather than globally
- Support bulk deployment operations like retirement and migration building on lower level rebalancing
  - eg retire hundreds of servers at end of warranty period

Cloud Storage

CAP Theorem

- The CAP theorem (Brewer, 2000) states that any networked shared-data system can have at most two of three desirable properties
  - consistency (C) equivalent to having a single up-to-date copy of the data
  - high availability (A) of data (incl. for update)
  - tolerance to network partitions (P)
- “two of three” should rather be seen as exclusion of all three at the same time
  - This means
    - eg distributed ACID databases can not scale
    - but eventual consistency can

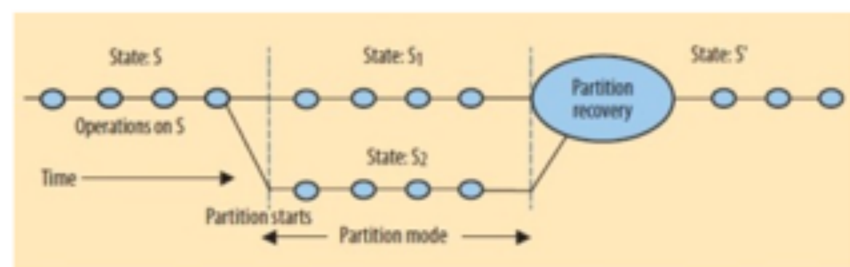
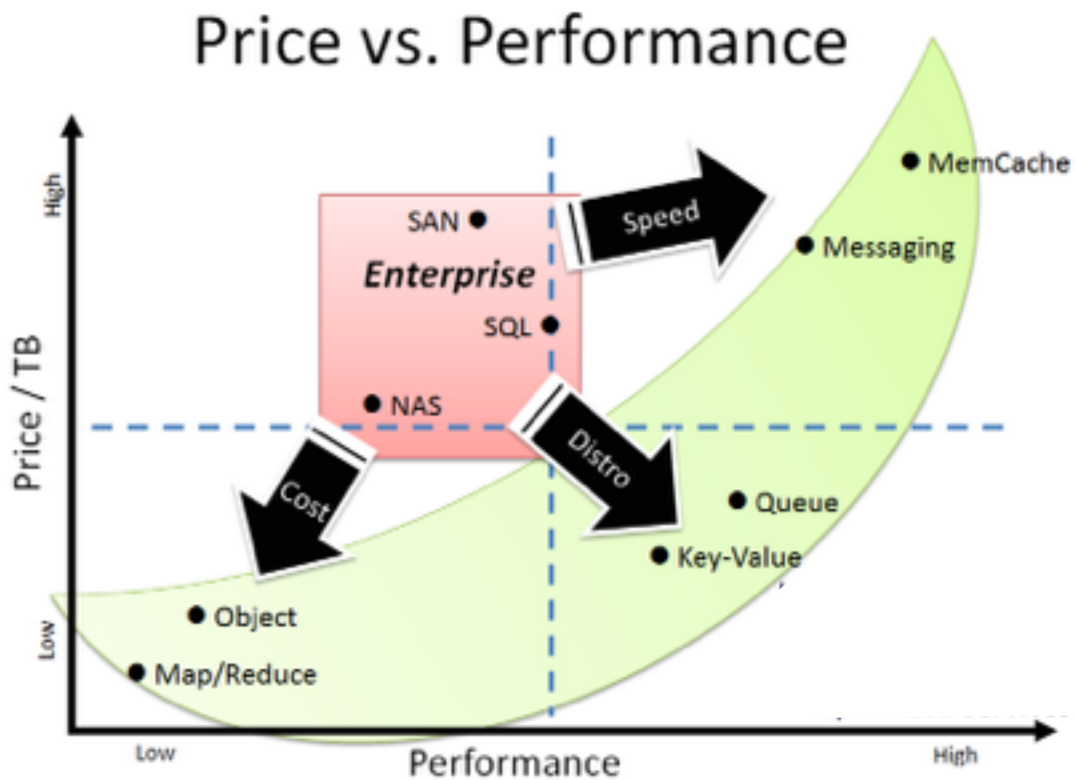


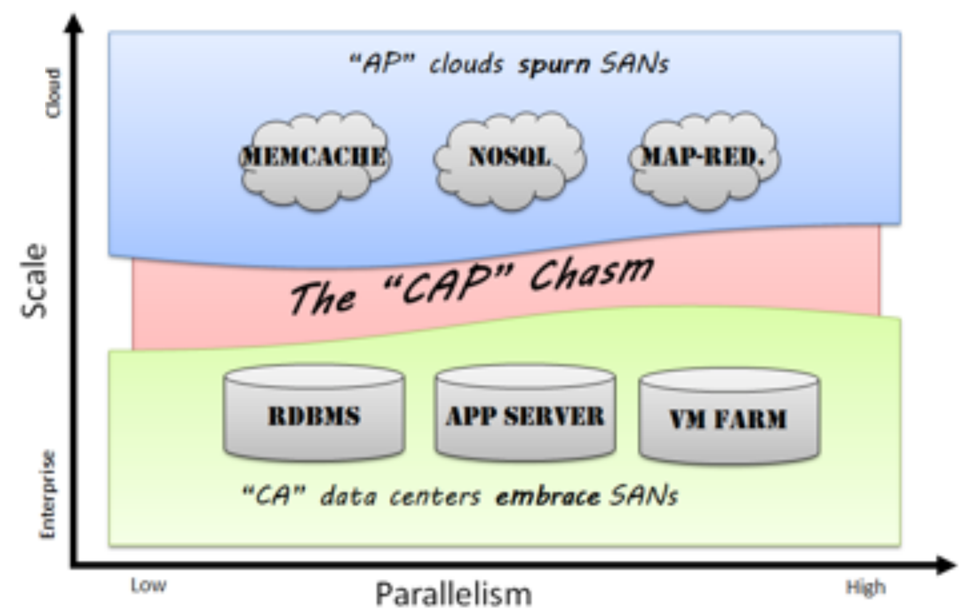
Figure 1. The state starts out consistent and remains so until a partition starts. To stay available, both sides enter partition mode and continue to execute operations, creating concurrent states S_1 and S_2, which are inconsistent. When the partition ends, the truth becomes clear and partition recovery starts. During recovery, the system merges S_1 and S_2 into a consistent state S' and also compensates for any mistakes made during the partition.

Cloud Storage



Cloud storage breaks one-size-fits-all model into optimized services

1. Legacy applications tried to eliminate faults to achieve Consistency with **physically redundant scale up** designs.
2. Cloud applications assume faults to achieve Partitioning Tolerance with **logically redundant scale out** design.



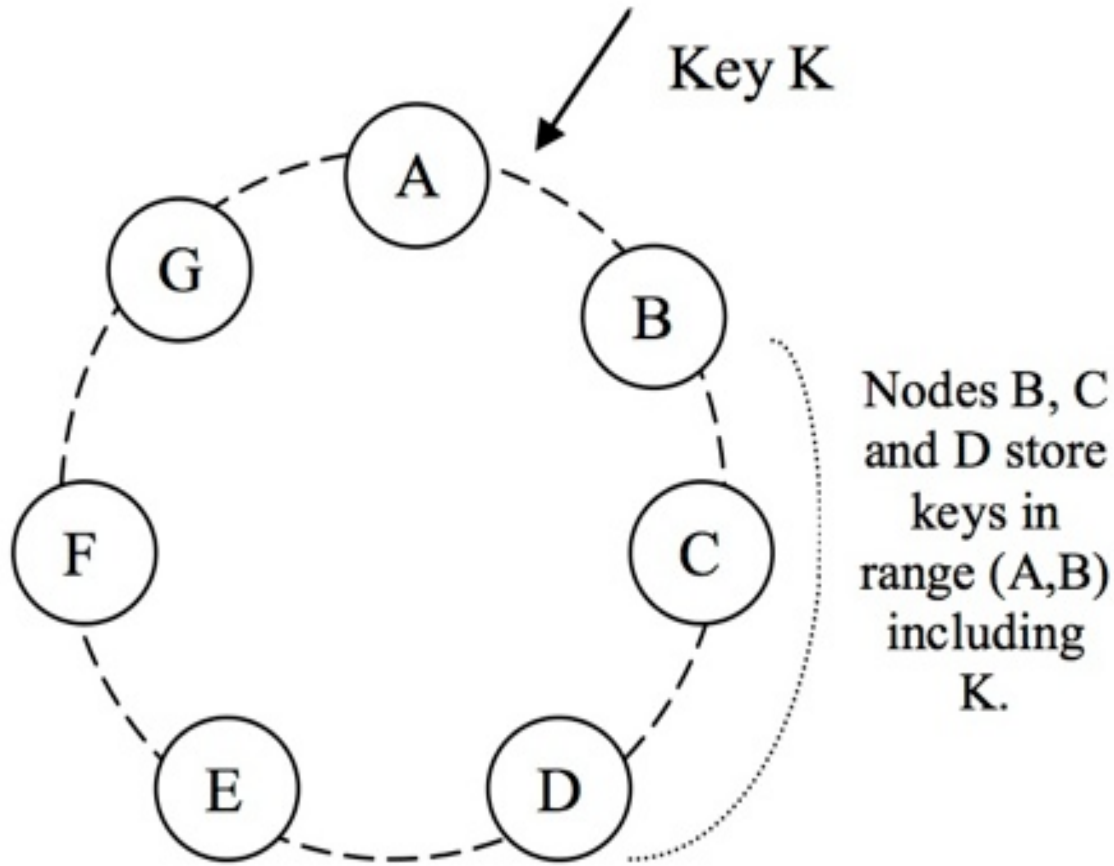
source: http://robhirschfeld.com/category/development/cap-theorem/

Amazon Dynamo - Distributed Hash Tables

Simple API

- **Sharded by hash of the Key**
  - Data = get (Key)
  - put (Key, Data)
  - delete (Key)

Dynamo Concept



source: James Hughes, CERN computing seminar

N*SQL

- Originally "no SQL" but more recently "not only SQL"
  - Databases which depart from relational model in several different ways
    - departing from something does not yet define where you are going…
- Physical data model
  - hierarchical / object / document databases
  - key-value stores
  - column stores
- Scalability / Availability
  - scale-out instead of scale-up
  - replication and fault-tolerance on node level instead of media level
- Consistency
  - eventual consistency instead of pessimistic/strict consistency

Physical Structure : Row vs Column vs Document

- Traditional RDBMS - transactional write load (eg OLTP)
  - one table row is changed / accessed together
- Analysis based on subset of attributes (eg data mining)
  - columns are stored / compressed together
  - performance advantage as less data is retrieved / transferred
- Access based on full, complex objects (document)
  - avoids complex joins
  - document store gives schema flexibility (no upfront schema)
    - application is responsible to handle unexpected data!

Key-Value Stores

- Scale-out - VOLDEMORT (LinkedIn)
  - Concept: Distributed, persistent, fault-tolerant hash table
  - Transparent data partitioning allows for cluster expansion without rebalancing all data
  - In-memory caching and durable storage system
    - no additional caching tier required
    - typically 10-20 kOperations/s

- Embedded / data structure server - REDIS
  - list, sets, hash-maps with atomic operations
    - optional asynchronous persistency
  - all data in-memory (no IO on key-lookup)
  - support publish/subscribe and large number of programming languages
  - often used for statistic data, histories

Mixed Store

- key-value / document store
  - view defined by java script
  - focus on clustering
    - scale-out with node count
    - new node can be added online
  - consistent hashing (to partition the data)
    - each node picks up part/shard of total data
    - three replicas by default with low rebalancing load after node addition
  - all nodes in the cluster are the same
    - all can be asked for any key
  - implemented in Erlang

Document Databases

- Scale-out example - CouchDB
  - read/writes via disk
  - JSON documents
- All access via http/REST (get/put/post/delete)
  - caching via reverse proxy (eg varnish)
  - specialised on web applications
- Application defined views
  - "schema" definition at query time
  - supporting server side calculation
  - map-reduce (in java script)
- Replication - "offline by default"
  - synchronise/replicate with data on other instances
  - explicit conflict handling (via doc revision/identity)

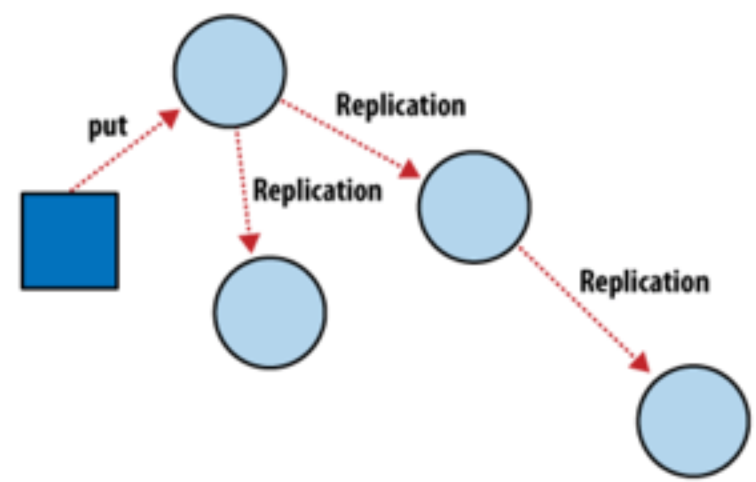
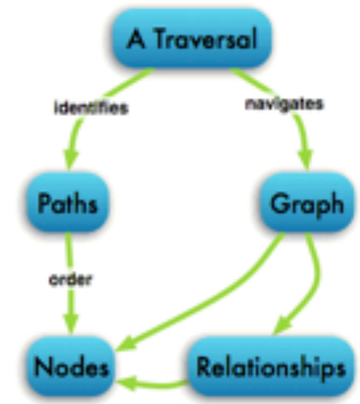


Figure 4. Incremental replication between CouchDB nodes

Graph Databases

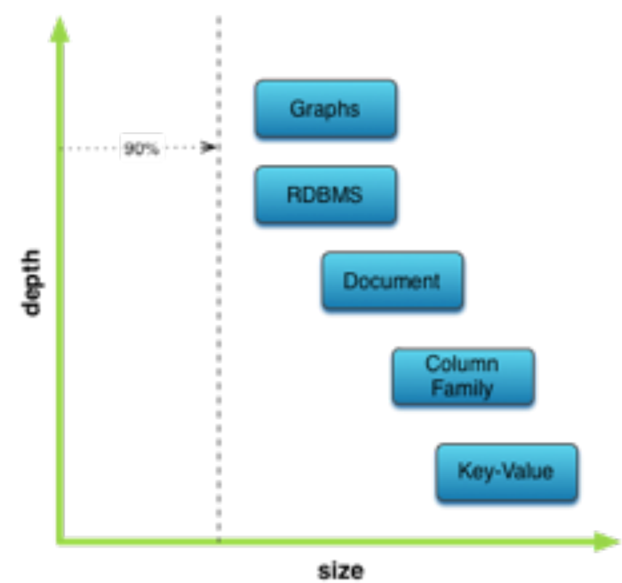
Neo4j
the graph database

- Data model
  - nodes, edges and attributes are first class objects
  - support for navigational and proximity queries
  - avoids data duplication / joins of relational network/graph representations



- fully transactional (ACID)

- Typical applications
  - social networks
  - geo-spacial

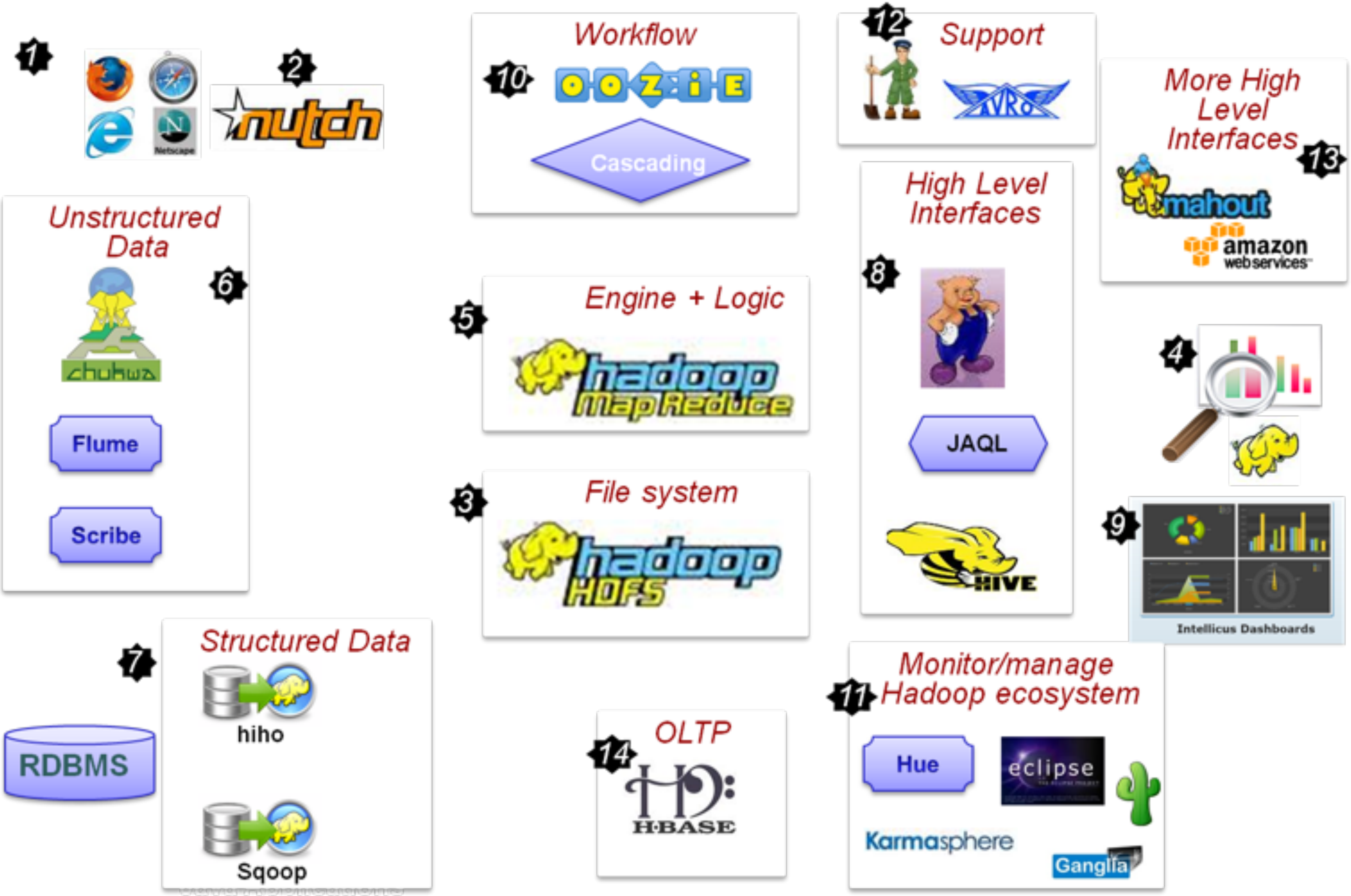


Hadoop

- Hadoop is more not just clustered storage
  - integrated data & processing infrastructure with generalised resource scheduling
- Parallel "local" access
  - Map Reduce
  - PIG/Latin
- Consistency constrained (scalable) database
  - HBase
  - Spark, Impala

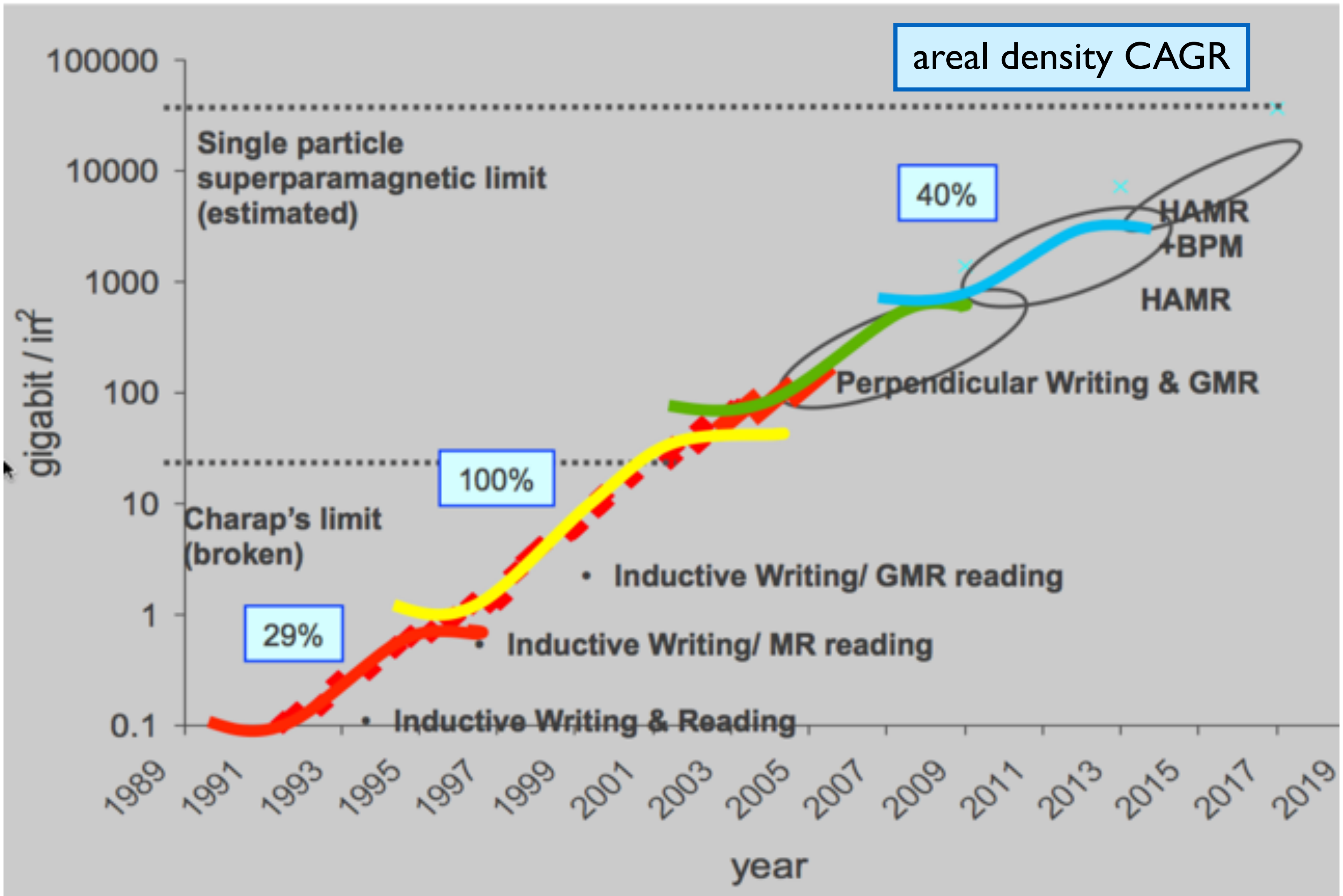
- Significant interest for analytics from CERN IT and LHC experiments
  - for analysis of infrastructure metrics & logs
    - often unstructured and without upfront data model
  - but also successfully to replace previous Oracle based operational log (eg what happened to file xyz in the last two weeks?)

Hadoop Ecosystem Map (always outdated)



source: http://indoos.wordpress.com/2010/08/16/hadoop-ecosystem-world-map/

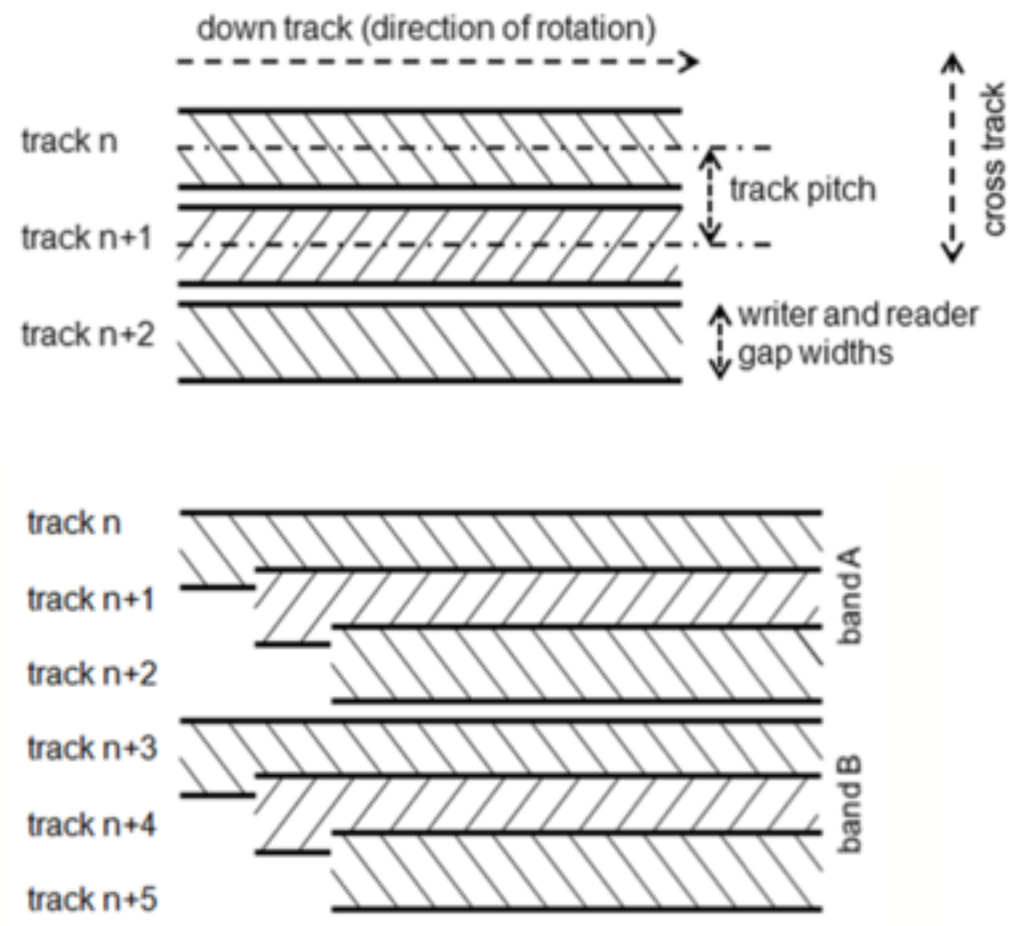
Does Kryder's law still hold? What's next for disk storage?



source: HDD Opportunities & Challenges, Now to 2020, Dave Anderson, Seagate

Shingled Recording

- Shingled Media
  - wide write head
  - narrow read head
- Result
  - continued density increase
  - but, write amplification within a band



Impact of Shingled Recording

- Gap between Read and Write performance increases
  - need to check eg if meta data mixing with data is still feasible
- Market / Application Impact
  - Will there be several types of disks?
    - emulation of a traditional disk
    - explicit band management by application
    - constraint semantics (object disk)
- Open questions:
  - which types will reach a market share & price that makes them attractive for science applications ?
  - how can the constrained semantics be mapped to science workflows?
  - CERN openlab R&D area

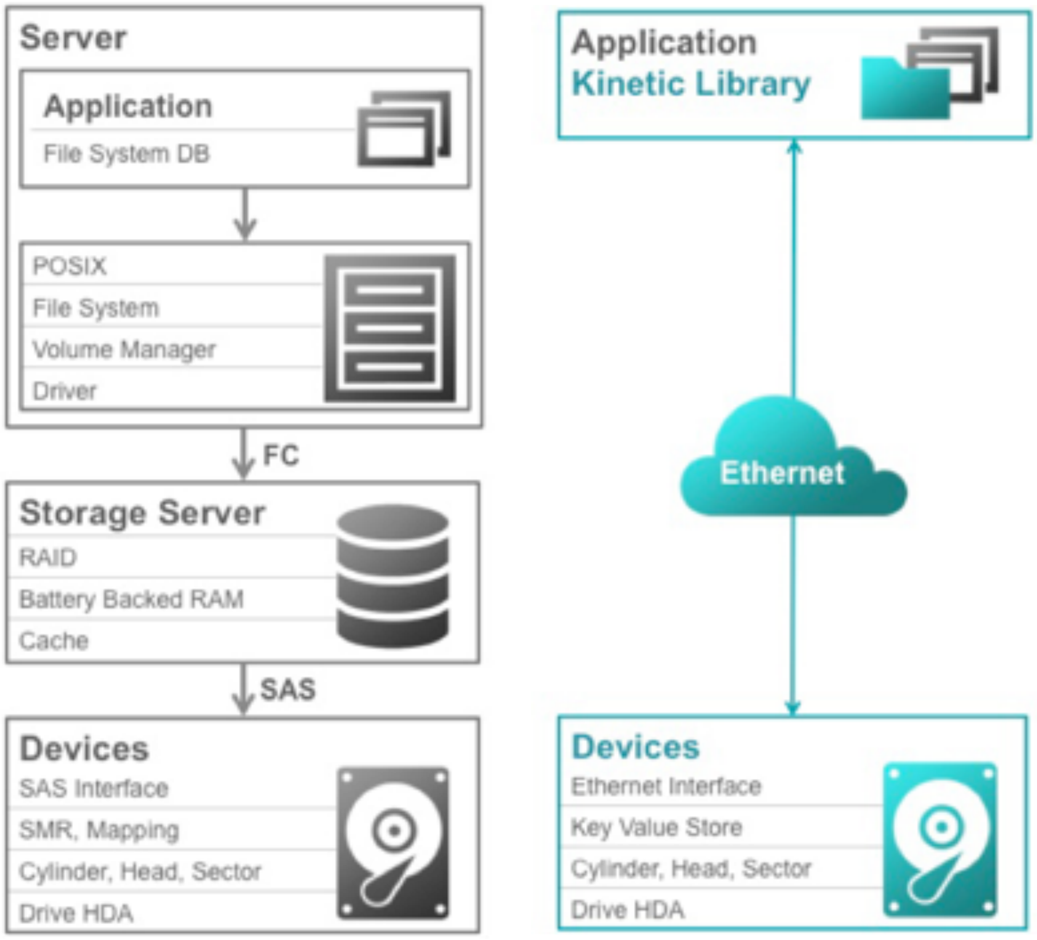
Object Disk

Seagate

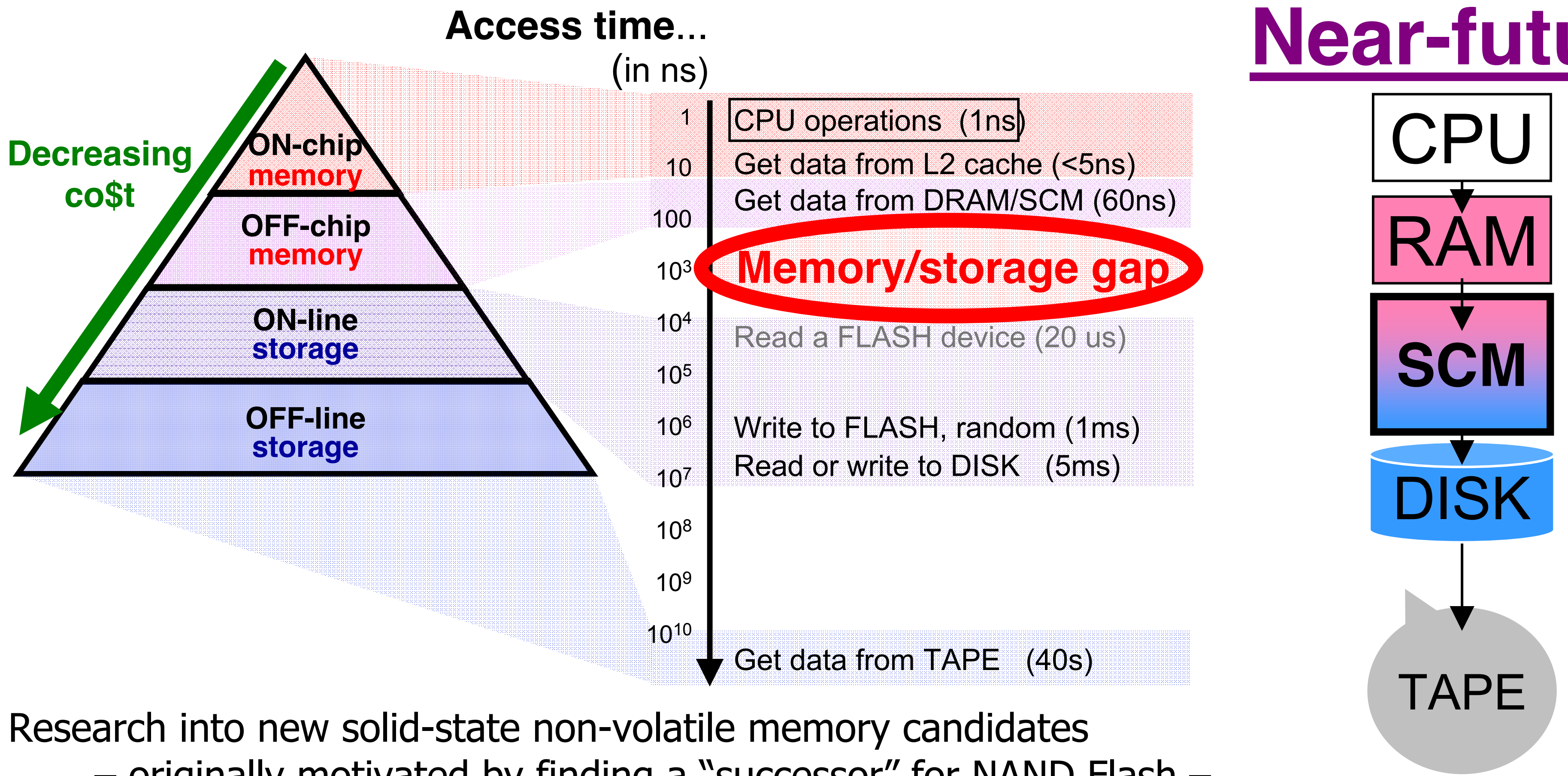
- Each disk talks object storage protocol over TCP
  - replication/failover with other disks in a networked disk cluster
  - open access library for app development

– Why now?
  - shingled disks match constrained object semantic: eg no updates

– Early stage with several open questions
  - port price for disk network vs price gain by reduced server/power cost?
  - standardisation of protocol/semantics to allow app development at low risk of vendor binding?



Problem (& opportunity): The access-time gap between memory & storage



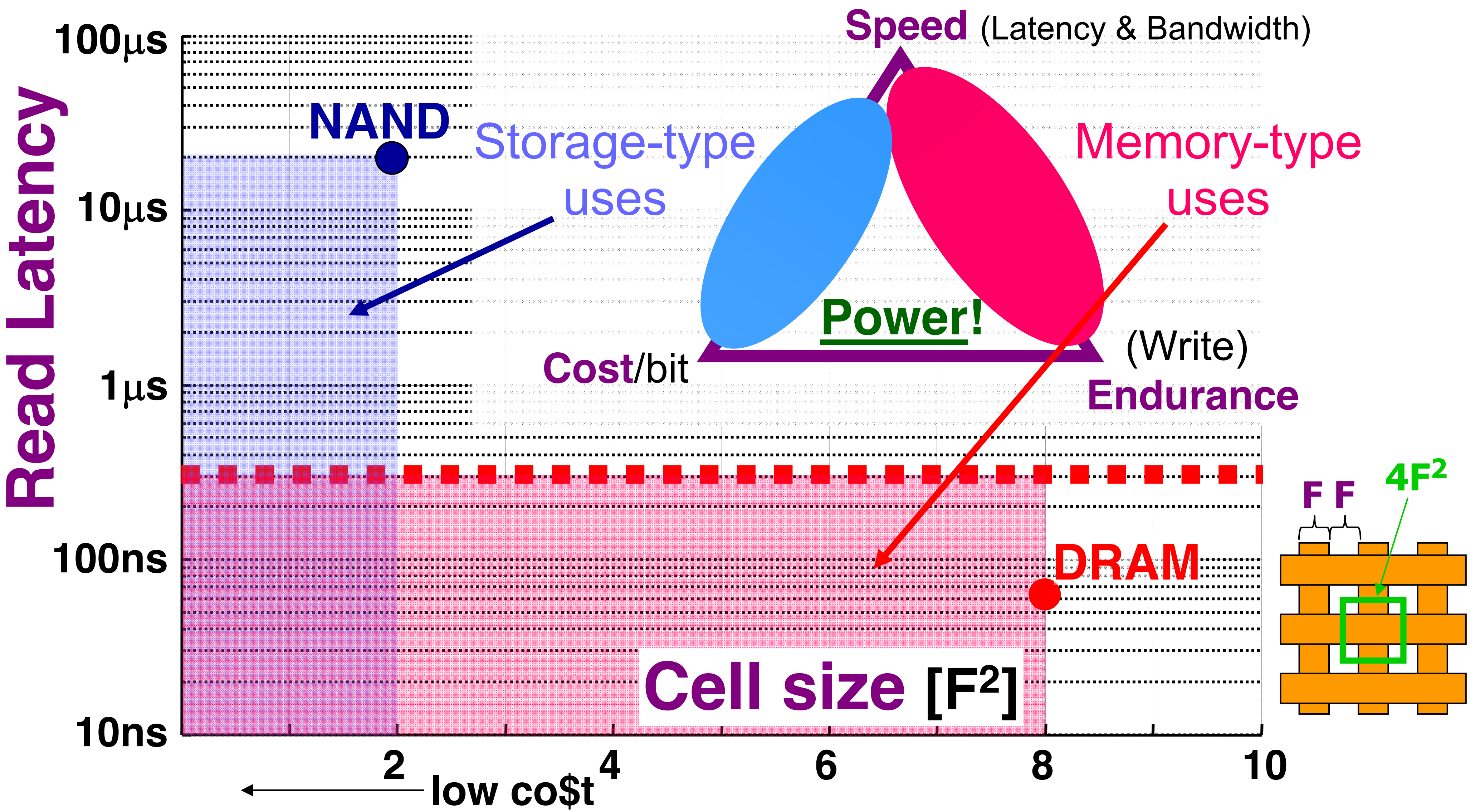
Research into new solid-state non-volatile memory candidates
– originally motivated by finding a "successor" for NAND Flash –
has opened up several interesting ways to change the memory/storage hierarchy...

1) **Embedded** Non-Volatile Memory – low-density, fast ON-chip NVM
2) **Embedded** Storage – low density, slower ON-chip storage

3) M-type **Storage Class Memory** – **high-density**, fast OFF- (or ON*)-chip NVM
4) S-type **Storage Class Memory** – **high-density**, very-near-ON-line storage

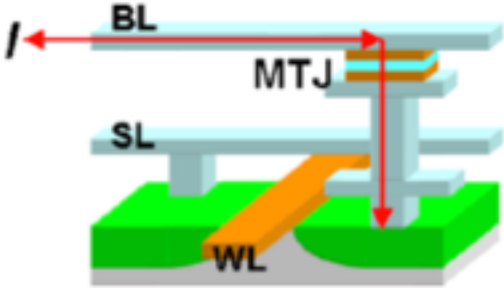
* ON-chip using 3-D packaging

Storage-type vs. memory-type Storage Class Memory



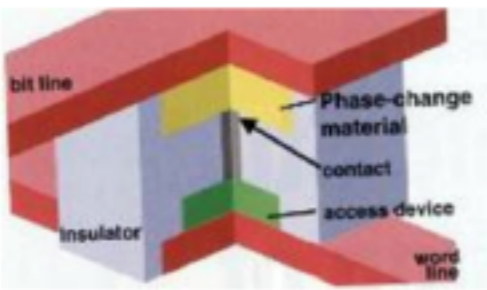
The cost basis of semiconductor processing is well understood – the paths to higher density are
1) shrinking the minimum lithographic pitch **F**, and 2) storing **more bits PER 4F²**

STT-MRAM



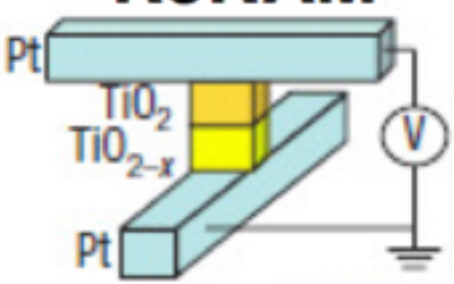
- High speed operation and non-volatility
- Main contender for DRAM replacement
- Eliminating DRAM refresh is a latency, bandwidth & power opportunity for STT-MRAM
- Complicated MTJ stacking structure, Yield challenge
- High temperature process & Low resistance ratio
- Margin Challenges, Soft errors
- 1x nm scaling and cost competitiveness??

PCM



- Most mature amongst emerging memory candidates – low density PCM in production for NOR replacement
- Drift challenges with high density PCM, Stuck Faults – reliability challenge
- Active Power, write current & latency – power/thermal challenges, too slow to work as main memory
- Scaling vs Thermal disturbance ??

ReRAM



- Very simple materials and structure
- Fast access, moderate endurance and low power
- Various and unclear switching mechanisms
- Large cell-to-cell variability
- EUV needed vs 3D NAND
- Stacking required for high density – manufacturing & yield challenges??

Summary

- The large data base area has seen a major differentiation from RDBMS to many, simpler, more scalable and more specialised systems
- Driven by
  - large available real memory
  - more flexible scale-out as demand grows
  - trading consistency for scalability
  - more natural match to application data model
- Upcoming technology changes in rotational and direct access memory will further blur the traditional split between database and other storage systems
- Application designers have a much larger toolbox available, but also need to be well aware of their specific requirements and acceptable trade-offs in order to exploit their advantages
- For larger organisations the need to consolidate centrally provided services will still be an important factor in choosing a technology for longer term projects