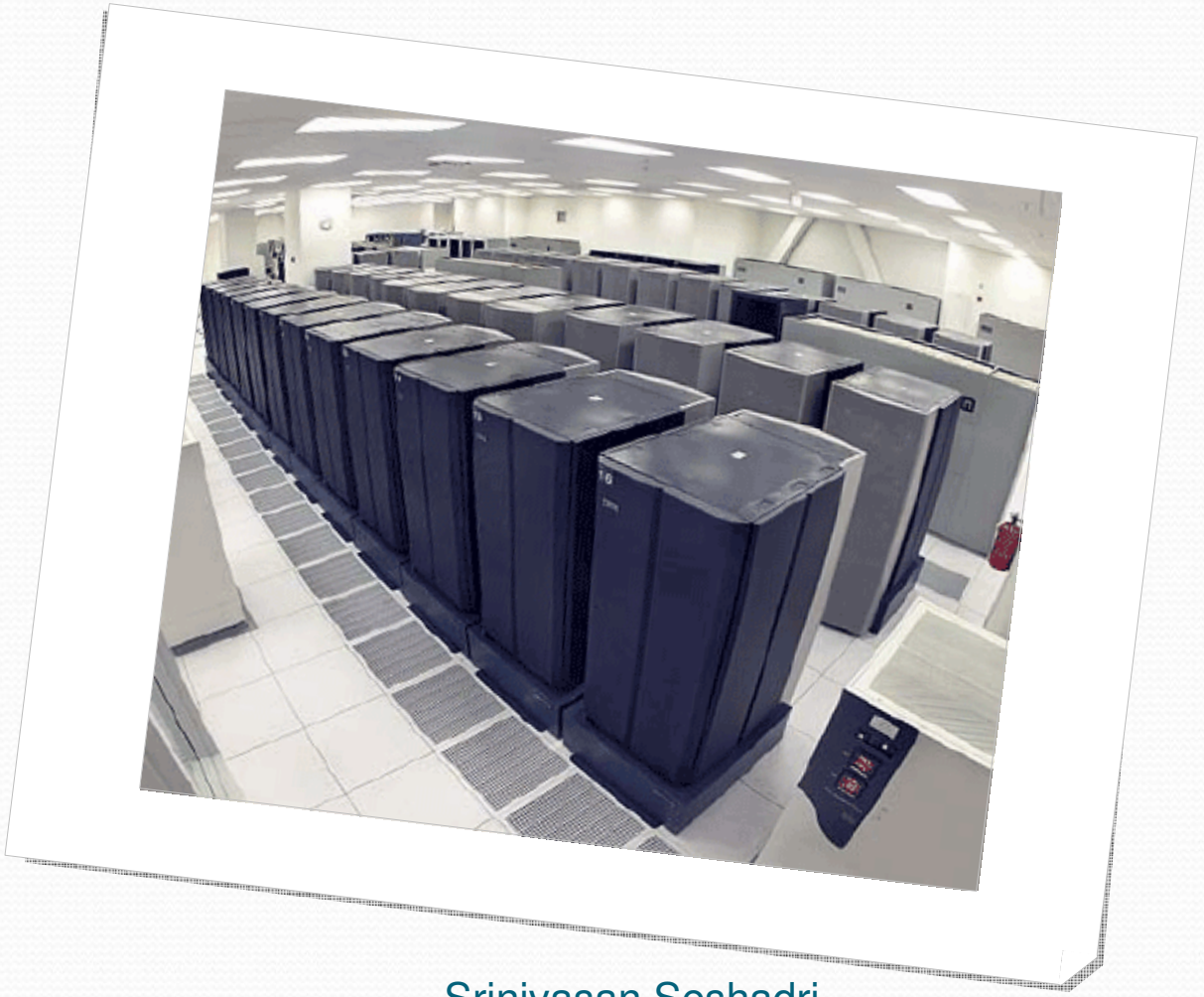
“Trends in Data Management”



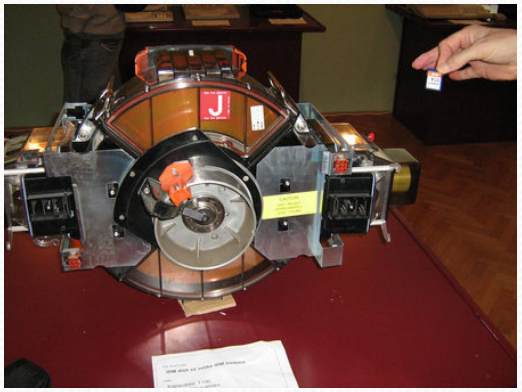
Srinivasan Seshadri
Boltell

Conventional Big Data Management Axioms

- Disk Accesses are very very expensive
- ACID is non negotiable
- You need a fixed schema
- Data is always in First Normal Form (at least)
  - No nested relations or more structure to a column
- An entire row is typically stored together
- SQL (a declarative language) is the only right access method to a data store
- A terabyte was Big Data in 1990's (TeraSort) – Scale is huge now – need thousands of machines for data management tasks

Hard Disk Evolution

1956: 5MB



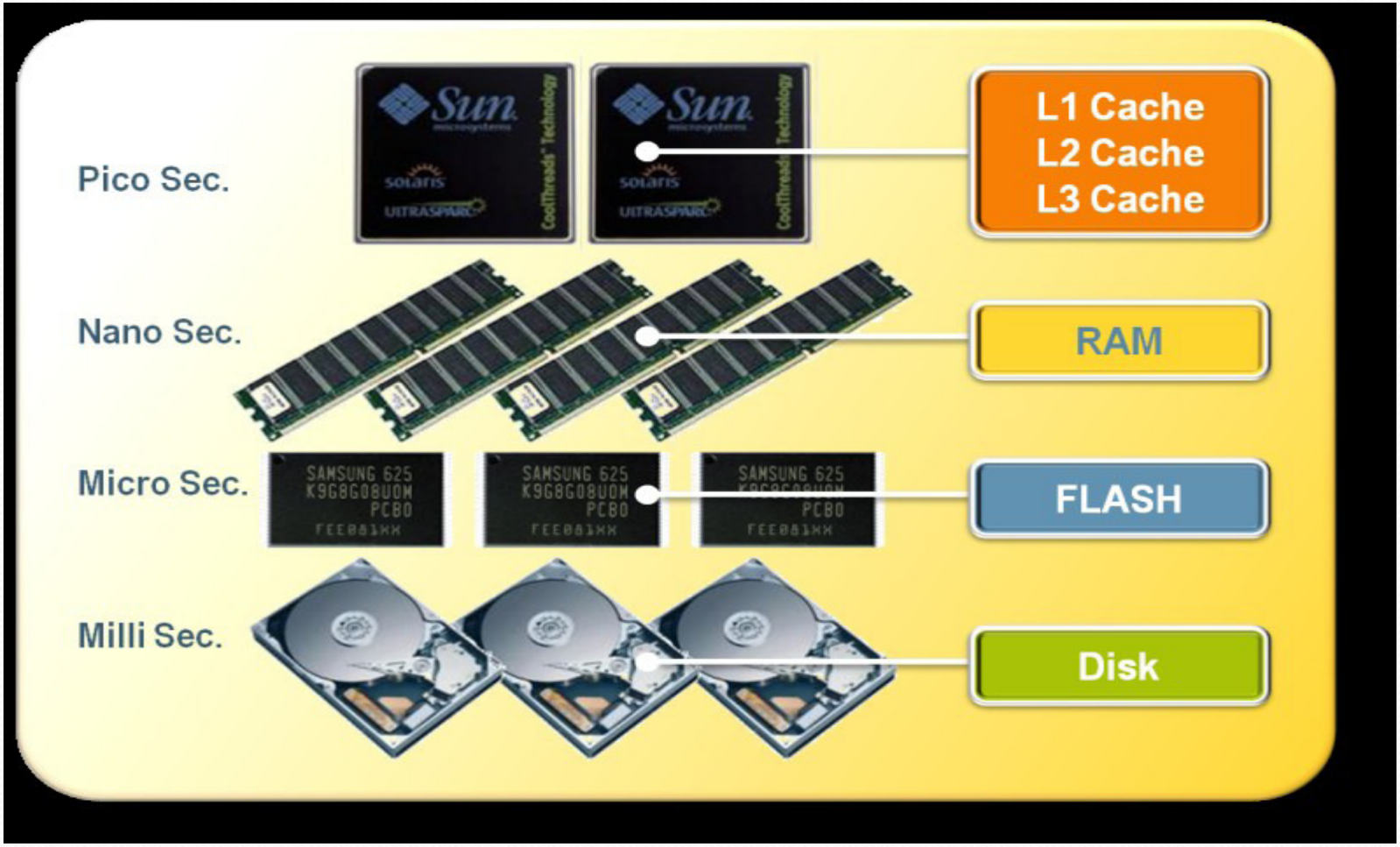
1980: 1GB

2008: 1TB

Enterprise SSD vs. HDD

- Great random read IOPS (> 100x)
  - 20 to 80K 8K reads/second (vs. 300)
- Very good random write IOPS (> 50x)
  - 5 to 30K 8K writes/second (vs. 300)
- Very good latency (< 1/50)
  - 100 to 300 us (vs. 7000)
- Good sequential bandwidth (> 3x)
  - 100 to 500 MB/s reads
- More expensive per GB
  - $2/GB: vs ($0.2/GB)
- Much cheaper per IOPS (50x)
- Consumes lot less power

The New Storage Hierarchy



ACID does not scale..

- A Simplistic, Practical interpretation of CAP Theorem. You can get only one of the following in the presence of arbitrary failures
  - Consistency: All replicas have the same value (or there is a total order on the writes)
  - Availability: You shall Always get a Non Null response

Conventional Data Management Systems choose Consistency.

Newer Data Management Systems choose Eventual Consistency.

Fixed Schemas are too rigid

- Schema Evolution is expensive in relation to the “fail fast” paradigm of startups!

- Several Approaches:
  - Column Families in BigTable
  - Dynamic Schema in PIG Latin

First Normal Form is too rigid..

- Real world models are not relational in general..
- Flashback to the OO days!

Columnar Stores

- Relaxing first normal form makes this even more appealing
- Can yield better compression and access speeds
- Not new – vertical partitioning was always possible even in conventional databases

SQL is no longer Intergalactic Dataspeak

- First Order Relational Calculus is not Turing Complete
- SQL was not designed for schema less or non first normal form data stores
- MapReduce (essentially group by) takes care of gory parallelization details (load balancing, failure handling, etc.) and is more amenable to programmers

Notion of Scale has changed dramatically

- Thousands of Commodity machines are needed to handle petabytes of data.
- These impose different architectural considerations and required distributed systems at that scale to be built ground up

NOSQL Data Stores

- Often, file management system, not DBMS
- Large volumes of data
- Non-relational and no support for joins
- Improved performance for large data sets
- Distributed databases and queries
- Fault tolerance so that an application will continue even if some connection is lost
- Horizontally scalable using component nodes
- Schema-free (so they have a more flexible structure)
- Eventually consistent (ACID-free)
- Easy to replicate so as to improve availability
- Access via API support or other non-SQL interfaces