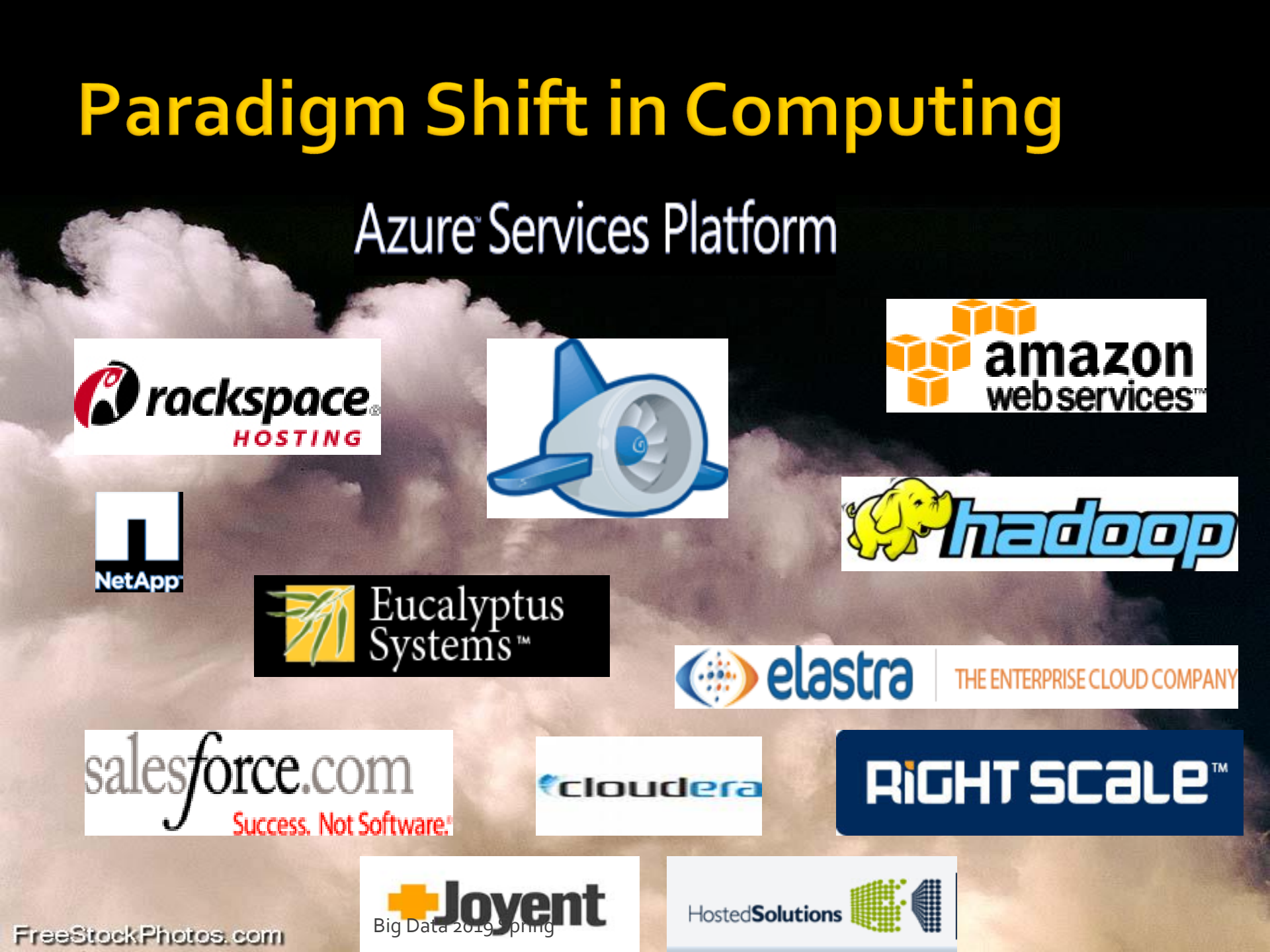
Big Data and Cloud Computing: Current State and Future Opportunities

Why? WEB is replacing the Desktop

Paradigm Shift in Computing



What is Cloud Computing?

- Delivering applications and services over the Internet:
  - Software as a service
- Extended to:
  - Infrastructure as a service: Amazon EC2
  - Platform as a service: Google AppEngine, Microsoft Azure
- Utility Computing: pay-as-you-go computing
  - Illusion of infinite resources
  - No up-front cost
  - Fine-grained billing (e.g. hourly)

Cloud Computing: History

> If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry.

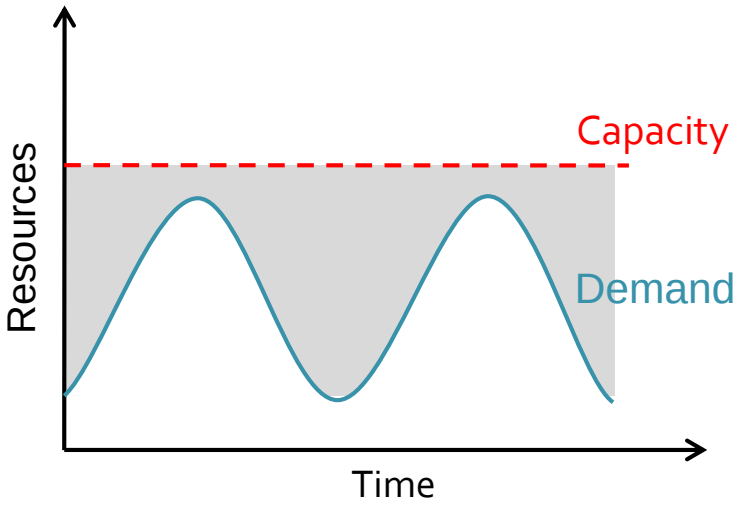
—John McCarthy, speaking at the MIT Centennial in 1961[2]

Cloud Computing: Why Now?

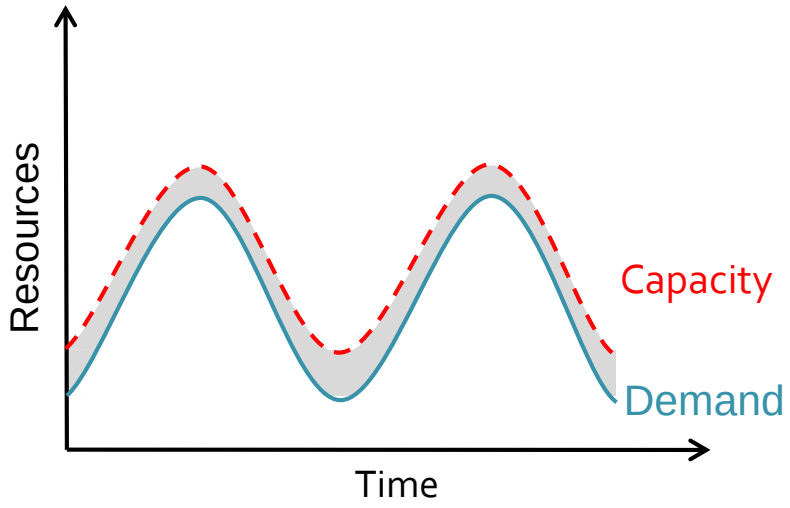
- Experience with very large datacenters
  - Unprecedented economies of scale
  - Transfer of risk
- Technology factors
  - Pervasive broadband Internet
  - Maturity in Virtualization Technology
- Business factors
  - Minimal capital expenditure
  - Pay-as-you-go billing model

Economics of Cloud Users

- Pay by use instead of provisioning for peak



Static data center



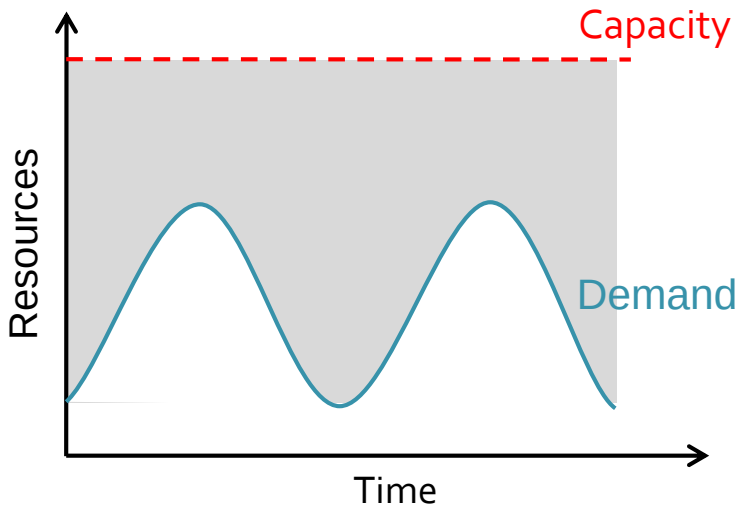
Data center in the cloud

Unused resources

Slide Credits: Berkeley RAD Lab

Economics of Cloud Users

- Risk of over-provisioning: underutilization



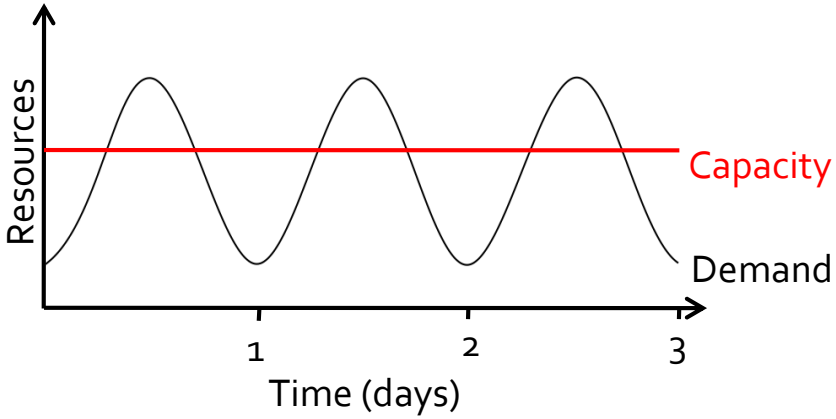
Static data center

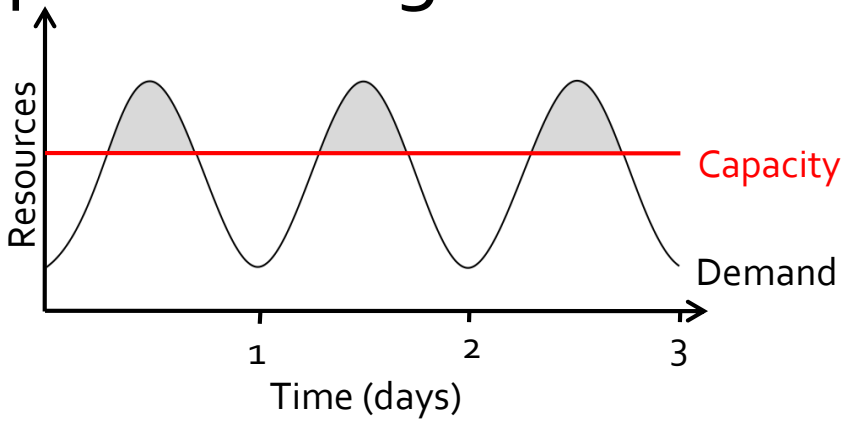
Unused resources

Slide Credits: Berkeley RAD Lab

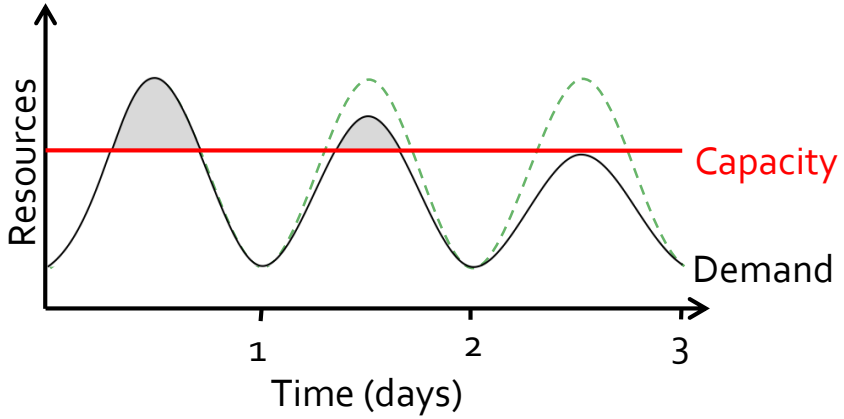
Economics of Cloud Users

- Heavy penalty for under-provisioning





Lost revenue



Lost users

Slide Credits: Berkeley RAD Lab

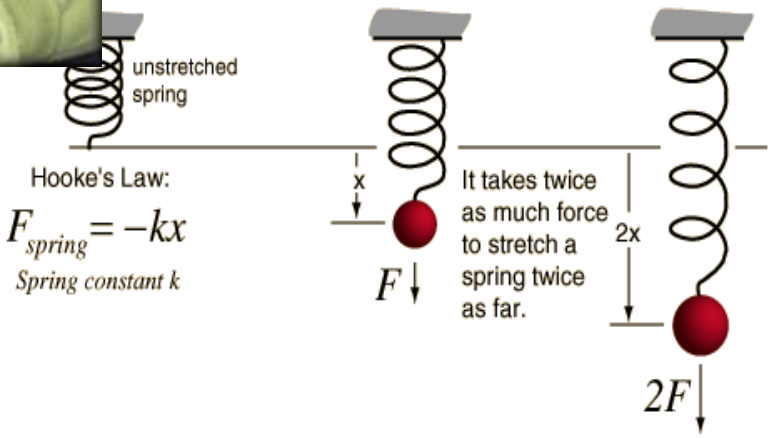
The Big Picture

- Unlike the earlier attempts:
  - Distributed Computing
  - Distributed Databases
  - Grid Computing
- Cloud Computing is likely to persist:
  - Organic growth: Google, Yahoo, Microsoft, and Amazon
  - Poised to be an integral aspect of National Infrastructure in US and other countries

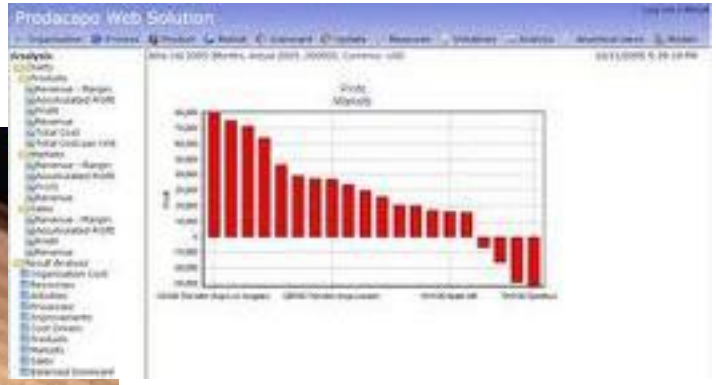
Cloud Reality

- Facebook Generation of Application Developers
- Animoto.com:
  - Started with 50 servers on Amazon EC2
  - Growth of 25,000 users/hour
  - Needed to scale to 3,500 servers in 2 days (RightScale@SantaBarbara)
- Many similar stories:
  - RightScale
  - Joyent
  - …

Cloud Challenges: Elasticity



Cloud Challenges: Differential Pricing Models



Outline

- **Data in the Cloud**
  - Platforms for Data Analysis
  - Platforms for Update intensive workloads
- Data Platforms for Large Applications
- Multitenant Data Platforms
- Open Research Challenges

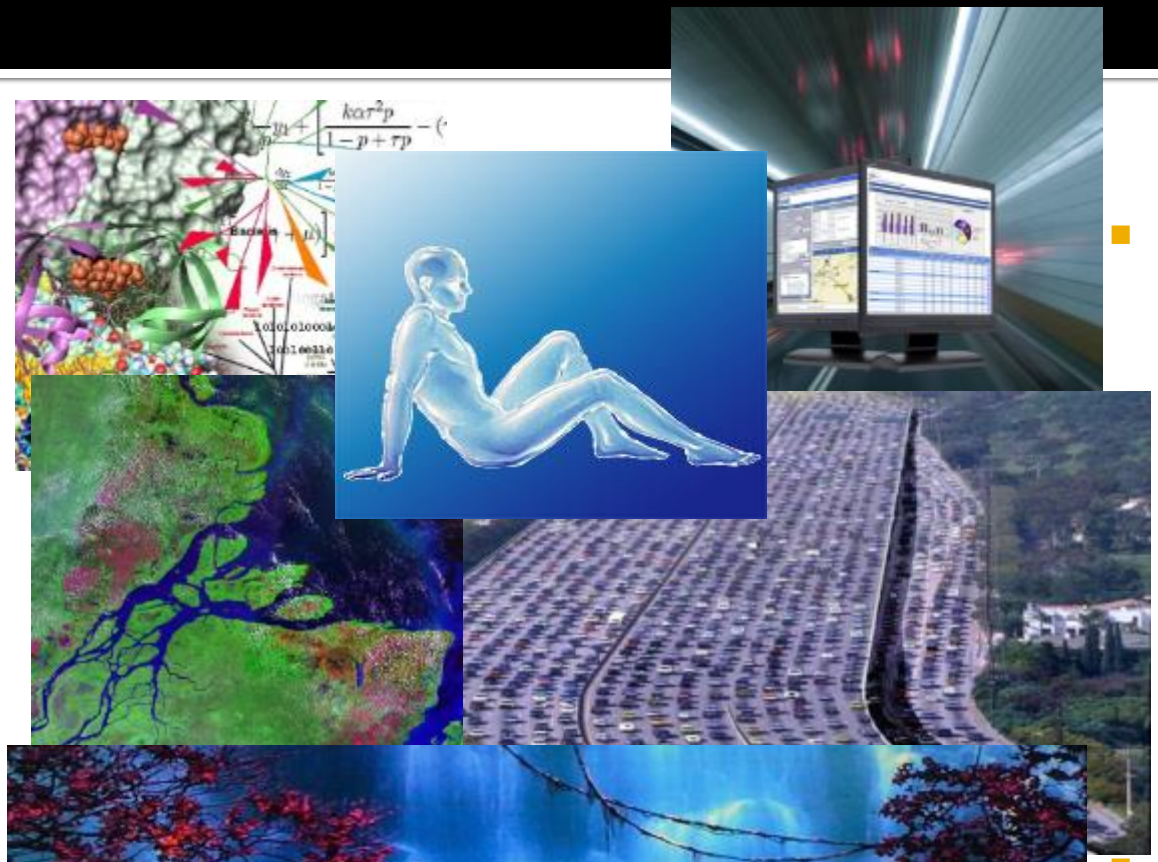
Our Data-driven World

- Science
  - Data bases from astronomy, genomics, environmental data, transportation data, …
- Humanities and Social Sciences
  - Scanned books, historical documents, social interactions data, …
- Business & Commerce
  - Corporate sales, stock market transactions, census, airline traffic, …
- Entertainment
  - Internet images, Hollywood movies, MP3 files, …
- Medicine
  - MRI & CT scans, patient records, …

Data-rich World

- Data capture and collection:
  - Highly instrumented environment
  - Sensors and Smart Devices
  - Network
- Data storage:
  - Seagate 1 TB Barracuda @ $72.95 from Amazon.com (73¢/GB)

What can we do with this wealth?



- What can we do?
  - Scientific breakthroughs
  - Business process efficiencies
  - Realistic special effects
  - Improve quality-of-life: healthcare, transportation, environmental disasters, daily life, …
- Could We Do More?
  - YES: but need major advances in our capability to analyze this data

Cloud Computing Modalities

"Can we outsource our IT software and hardware infrastructure?"

- Hosted Applications and services
- Pay-as-you-go model
- Scalability, fault-tolerance, elasticity, and self-manageability

"We have terabytes of click-stream data – what can we do with it?"

- Very large data repositories
- Complex analysis
- Distributed and parallel data processing

Outline

- **Data in the Cloud**
  - **Platforms for Data Analysis**
  - Platforms for Update intensive workloads
- Data Platforms for Large Applications
- Multitenant Data Platforms
- Open Research Challenges

Data Warehousing, Data Analytics & Decision Support Systems

- Used to manage and control business
- Transactional Data: historical or point-in-time
- Optimized for inquiry rather than update
- Use of the system is loosely defined and can be ad-hoc
- Used by managers and analysts to understand the business and make judgments

Data Analytics in the Web Context

- Data capture at the user interaction level:
  - in contrast to the client transaction level in the Enterprise context
- As a consequence the amount of data increases significantly
- Greater need to analyze such data to understand user behaviors

Data Analytics in the Cloud

- Scalability to large data volumes:
  - Scan 100 TB on 1 node @ 50 MB/sec = 23 days
  - Scan on 1000-node cluster = 33 minutes

➔ Divide-And-Conquer (i.e., data partitioning)

- Cost-efficiency:
  - Commodity nodes (cheap, but unreliable)
  - Commodity network
  - Automatic fault-tolerance (fewer administrators)
  - Easy to use (fewer programmers)

Platforms for Large-scale Data Analysis

- **Parallel DBMS technologies**
  - Proposed in the late eighties
  - Matured over the last two decades
  - Multi-billion dollar industry: Proprietary DBMS Engines intended as Data Warehousing solutions for very large enterprises
- **Map Reduce**
  - pioneered by Google
  - popularized by Yahoo! (Hadoop)

Parallel DBMS technologies

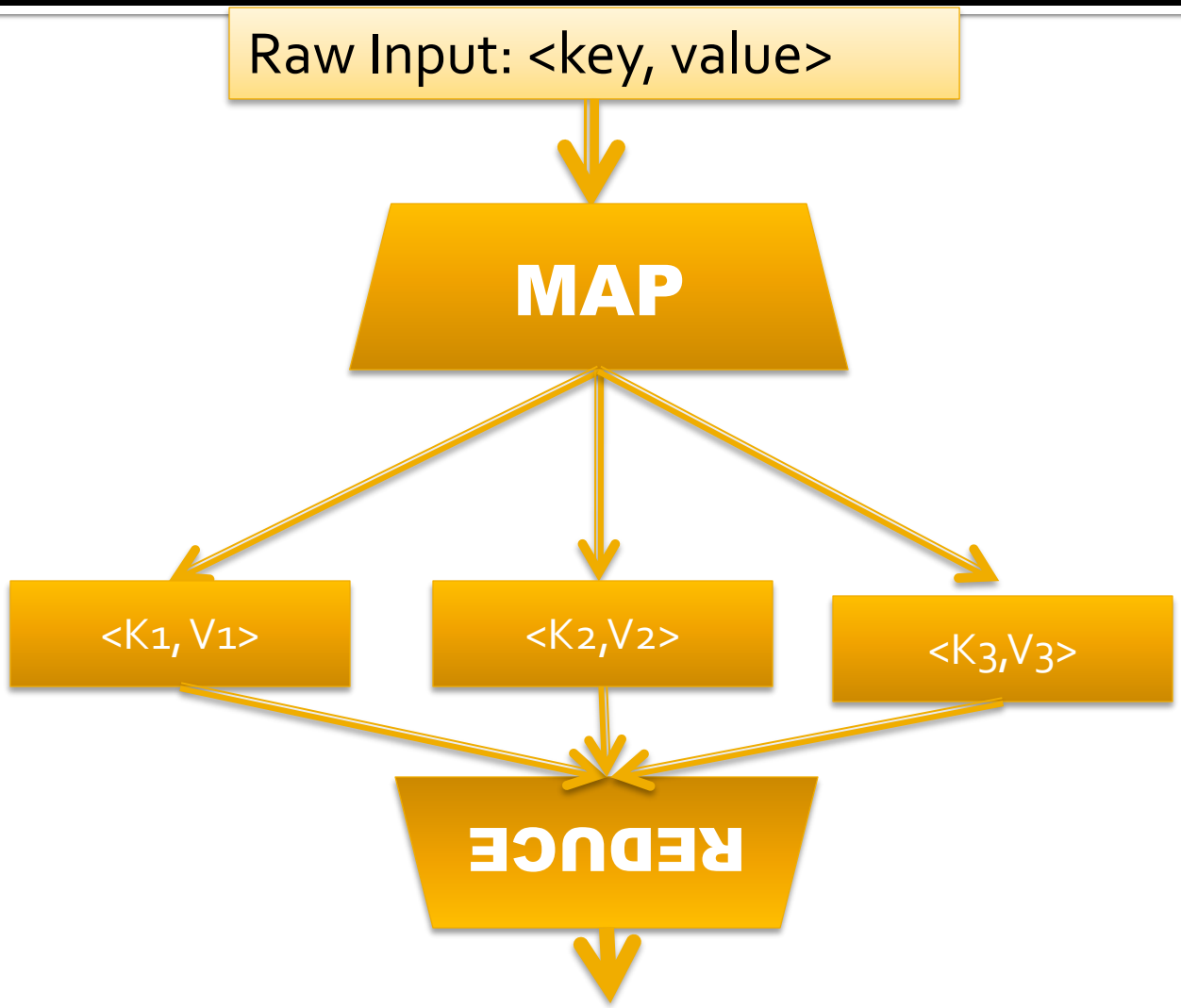
- Popularly used for more than two decades
  - Research Projects: Gamma, Grace, …
  - Commercial: Multi-billion dollar industry but access to only a privileged few
- Relational Data Model
- Indexing
- Familiar SQL interface
- Advanced query optimization
- *Well understood and well studied*

MapReduce

[Dean et al., OSDI 2004, CACM Jan 2008, CACM Jan 2010]

- Overview:
  - Data-parallel programming model
  - An associated parallel and distributed implementation for commodity clusters
- Pioneered by Google
  - Processes 20 PB of data per day
- Popularized by open-source Hadoop project
  - Used by Yahoo!, Facebook, Amazon, and the list is growing …

Programming Framework



MapReduce Advantages

- Automatic Parallelization:
  - Depending on the size of RAW INPUT DATA ➔ instantiate multiple MAP tasks
  - Similarly, depending upon the number of intermediate <key, value> partitions ➔ instantiate multiple REDUCE tasks
- Run-time:
  - Data partitioning
  - Task scheduling
  - Handling machine failures
  - Managing inter-machine communication
- Completely transparent to the programmer/analyst/user

MapReduce Experience

- Runs on large commodity clusters:
  - 1000s to 10,000s of machines
- Processes many terabytes of data
- Easy to use since run-time complexity hidden from the users
- 1000s of MR jobs/day at Google (circa 2004)
- 100s of MR programs implemented (circa 2004)

The Need

- Special-purpose programs to process large amounts of data: crawled documents, Web Query Logs, etc.
- At Google and others (Yahoo!, Facebook):
  - Inverted index
  - Graph structure of the WEB documents
  - Summaries of #pages/host, set of frequent queries, etc.
  - Ad Optimization
  - Spam filtering

MapReduce Contributions

Simple & Powerful Programming Paradigm For Large-scale Data Analysis

Run-time System For Large-scale Parallelism & Distribution

Takeaway

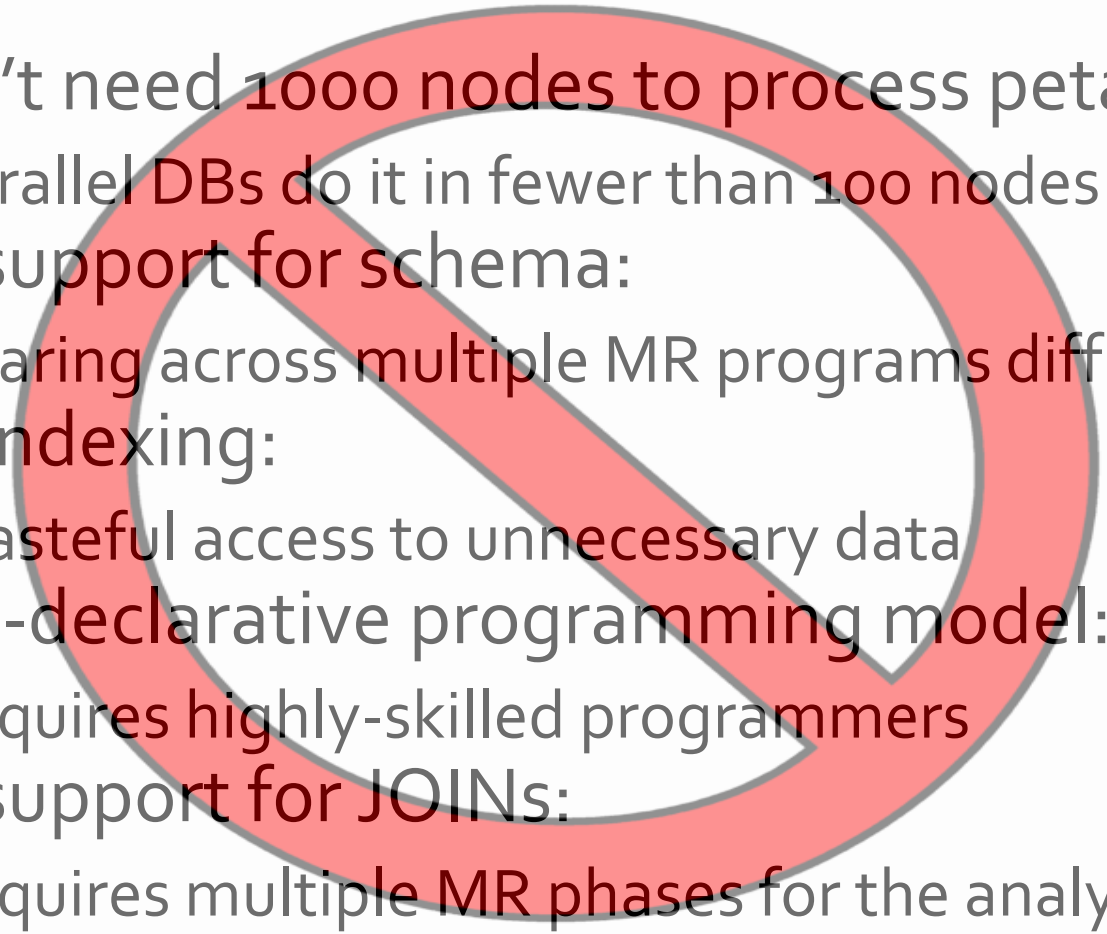
- MapReduce's data-parallel programming model hides complexity of distribution and fault tolerance
- Key philosophy:
  - *Make it scale*, so you can throw hardware at problems
  - *Make it cheap*, saving hardware, programmer and administration costs (but requiring fault tolerance)
- Hive and Pig further simplify programming
- MapReduce is not suitable for all problems, but when it works, it may save you a lot of time

Map Reduce vs Parallel DBMS

[Pavlo et al., SIGMOD 2009, Stonebraker et al., CACM 2010, …]

	Parallel DBMS	MapReduce
Schema Support	✓	Not out of the box
Indexing	✓	Not out of the box
Programming Model	Declarative (SQL)	Imperative (C/C++, Java, …) **Extensions through Pig and Hive**
Optimizations (Compression, Query Optimization)	✓	Not out of the box
Flexibility	Not out of the box	✓
Fault Tolerance	Coarse grained techniques	✓

MapReduce: A step backwards?



- Don't need 1000 nodes to process petabytes:
  - Parallel DBs do it in fewer than 100 nodes
- No support for schema:
  - Sharing across multiple MR programs difficult
- No indexing:
  - Wasteful access to unnecessary data
- Non-declarative programming model:
  - Requires highly-skilled programmers
- No support for JOINs:
  - Requires multiple MR phases for the analysis

MapReduce VS Parallel DB: Our 2¢

- Web application data is inherently distributed on a large number of sites:
  - Funneling data to DB nodes is a failed strategy
- Distributed and parallel programs difficult to develop:
  - Failures and dynamics in the cloud
- Indexing:
  - Sequential Disk access 10 times faster than random access.
  - Not clear if indexing is the right strategy.
- Complex queries:
  - DB community needs to JOIN hands with MR

Leveraging DBMS Technology for Scalable Analytics

Scalable Analytics Innovations

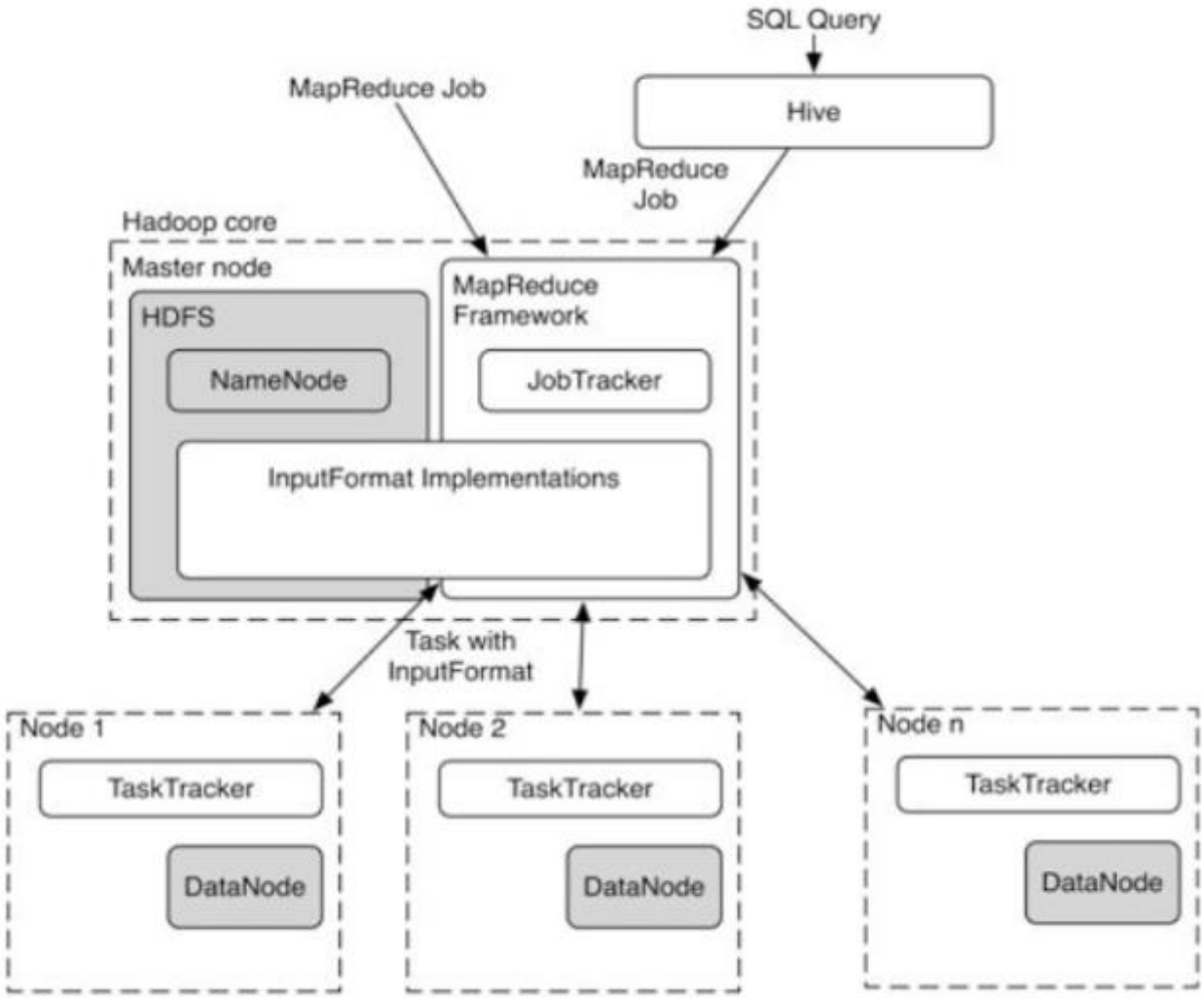
- Asynchronous Views over Cloud Data
  - Yahoo! Research (SIGMOD'2009)
- DataPath: Data-centric Analytic Engine
  - U Florida (Dobra) & Rice (Jermaine) (SIGMOD'2010)
- MapReduce innovations:
  - MapReduce Online (UCBerkeley)
  - HadoopDB (Yale)
  - Multi-way Join in MapReduce (Afrati&Ullman: EDBT'2010)
  - Hadoop++ (Dittrich et al.: VLDB'2010) and others

Hadoop DB – A Hybrid Approach

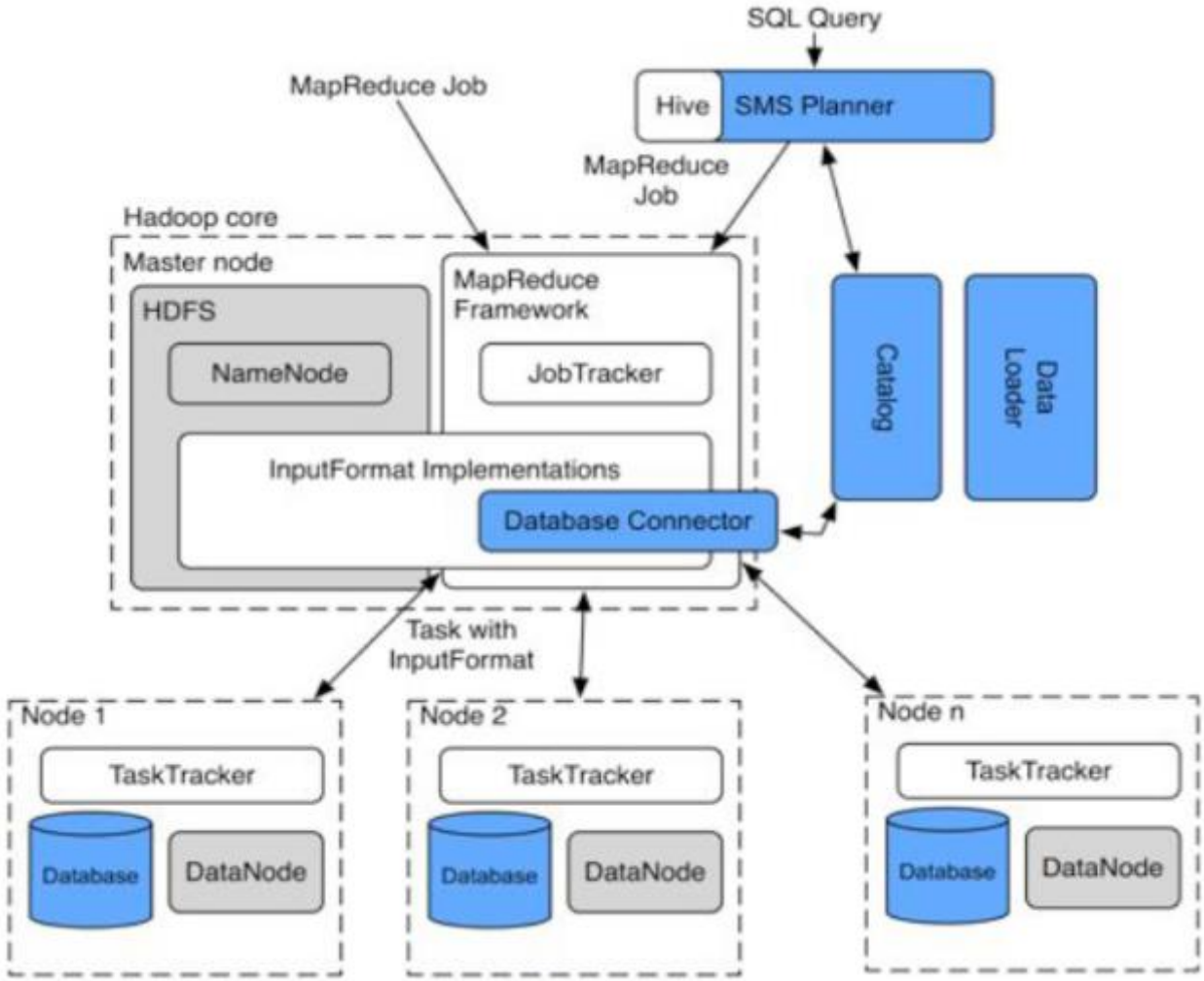
[Abouzeid et al., VLDB 2009]

- An architectural hybrid of MapReduce and DBMS technologies
- Use Fault-tolerance and Scale of MapReduce framework like Hadoop
- Leverage advanced data processing techniques of an RDBMS
- Expose a declarative interface to the user
- ***Goal: Leverage from the best of both worlds***

Architecture of HadoopDB



Architecture of HadoopDB



Two-way Joins and MapReduce

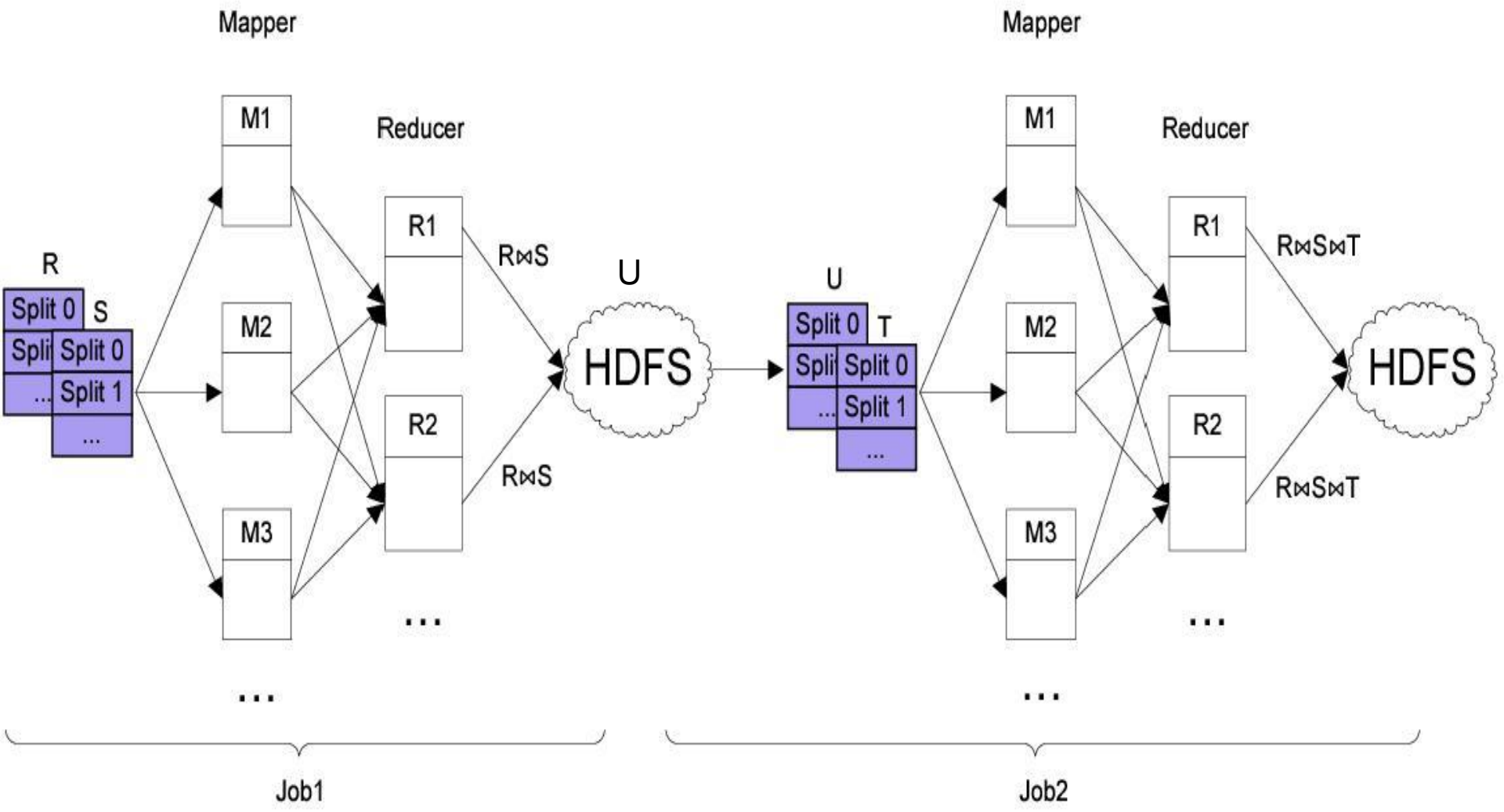
- MapReduce works with a single data source (table):
  - <key K, value V>
- How to use the MR framework to compute:
  - R(A, **B**) ⋈ S(**B**, C)
- Simple extension (proposed independently by multiple researchers):
  - <a, **b**> from R is mapped as: <**b**, [R, a]>
  - <**b**, c> from S is mapped as: <**b**, [S, c]>
- During the reduce phase:
  - Join the key-value pairs with the same key but different relations

Beyond Two-way Joins?

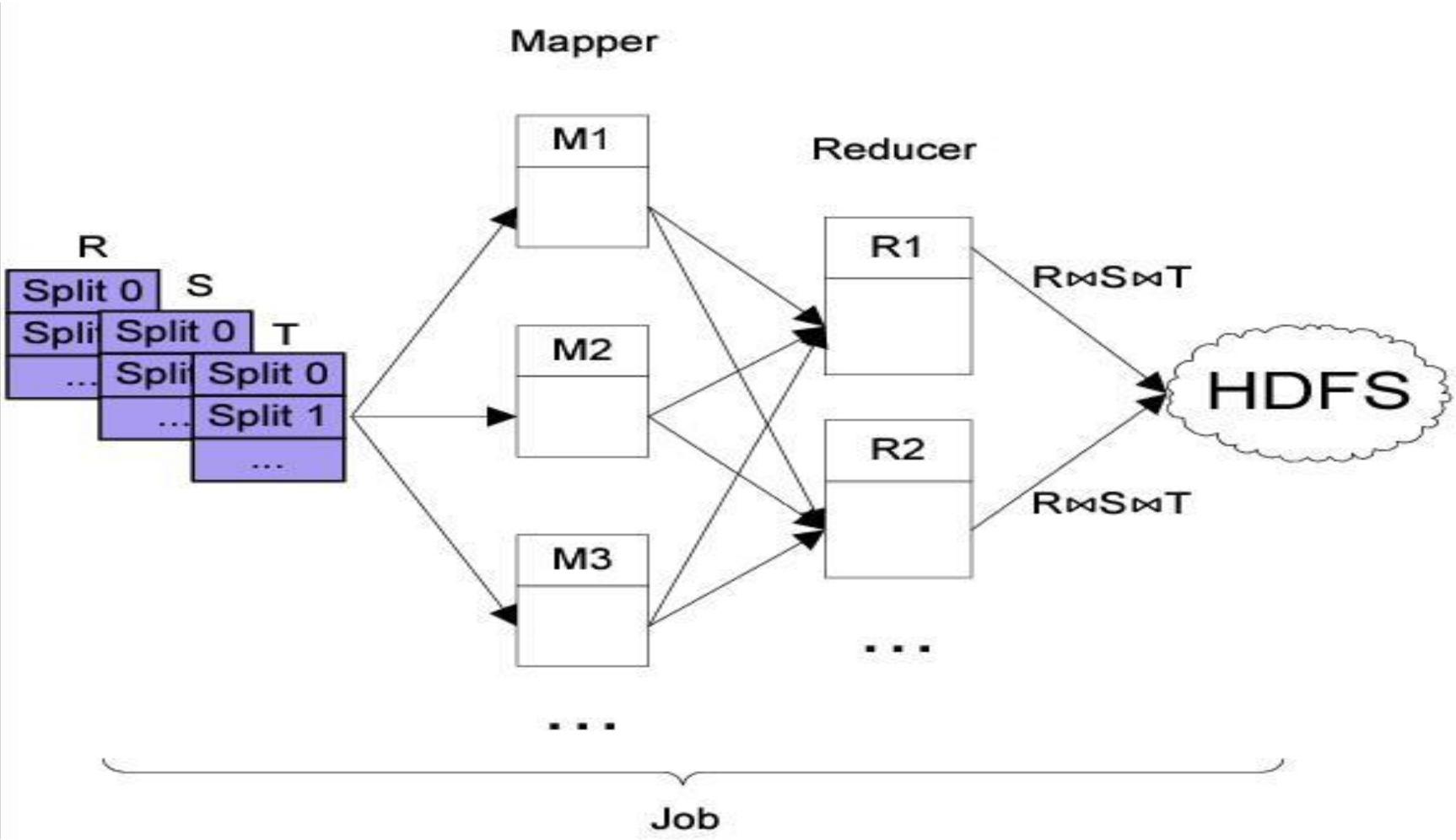
- How to generalize to:
  - R(A, B) ⋈ S(B, C) ⋈ T(C,D)

  in MapReduce?

Beyond Two-way Joins

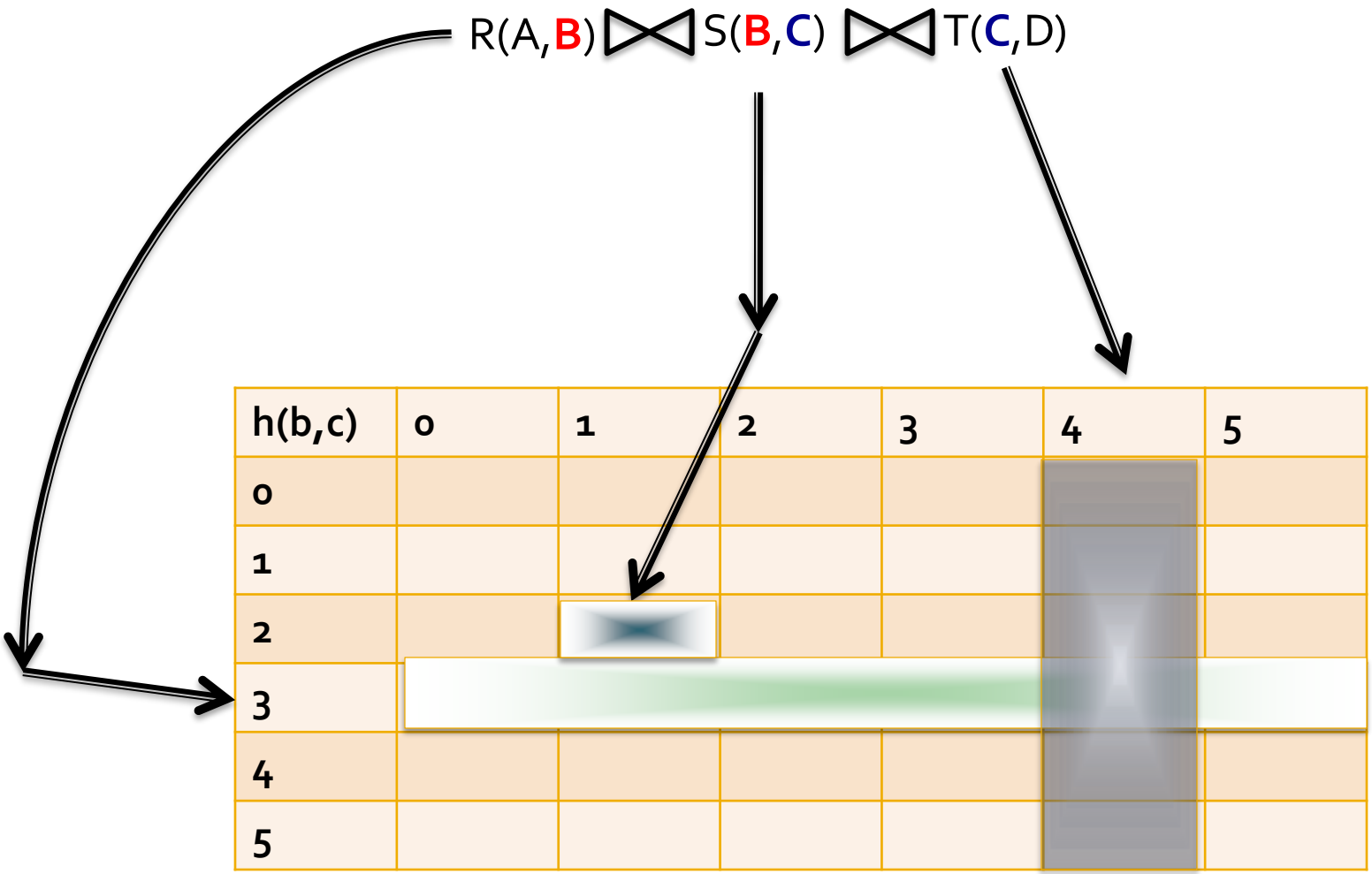


Beyond Two-way Joins?



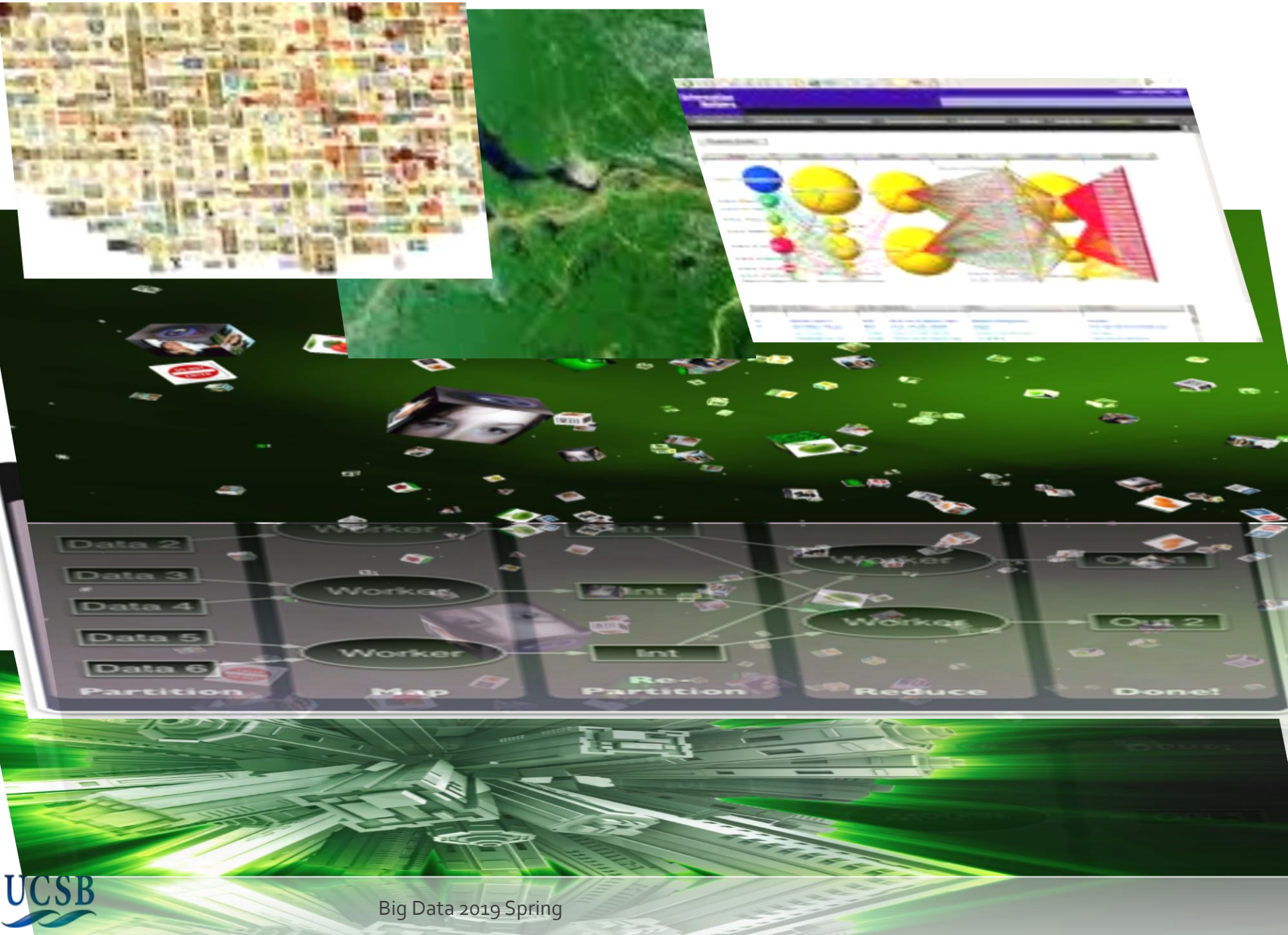
Multi-way Join

[Afrati & Ullman: EDBT'2010]



OLAP over MapReduce

- Multi-way Join over a single MR phase:
  - One-to-many **shuffle**
- Communication cost:
  - 3-way Join: O(n) communication
  - 4-way Join: $O(n^2)$
  - …
  - M-way Join: $O(n^{M-2})$
- Clearly, not feasible for OLAP:
  - Large number of Dimension Tables
  - Many OLAP queries involve Join of FACT table with multiple Dimension tables.


Data 2
Data 3
Data 4
Data 5
Data 6
Worker
Worker
Worker
Int
Int
Worker
Worker
Out 1
Out 2
Partition
Map
Re-
Partition
Reduce
Done!

Scalable OLAP

[SIGMOD'2011: NUS & UCSB Collaboration]

- Observations:
  - STAR Schema (or its variant)
  - DIMENSION tables typically are not very large (in most cases)
  - FACT table, on the other hand, have a large number of rows.
- Design approach:
  - Use MapReduce as a distributed & parallel computing substrate
  - Fact tables partitioned across multiple nodes
  - Partitioning strategy should be based on the dimension that are most often used as selection constraint in DW queries
  - Dimension tables are replicated
  - MAP tasks perform the STAR joins
  - REDUCE tasks then perform the aggregation

Data Analytics in the Cloud

New Challenges and Opportunities

Looking Forward

New Applications

- Complex data processing – Graphs and beyond
- Multidimensional Data Analytics: Location-based data
- Physical and Virtual Worlds: Social Networks and Social Media data & analysis

Conjectures

- New breed of Analysts:
  - Information-savvy users
  - Most users will become nimble analysts
  - Most transactional decisions will be preceded by a detailed analysis
- Convergence of OLAP and OLTP:
  - Both from the application point-of-view and from the infrastructure point-of-view

Outline

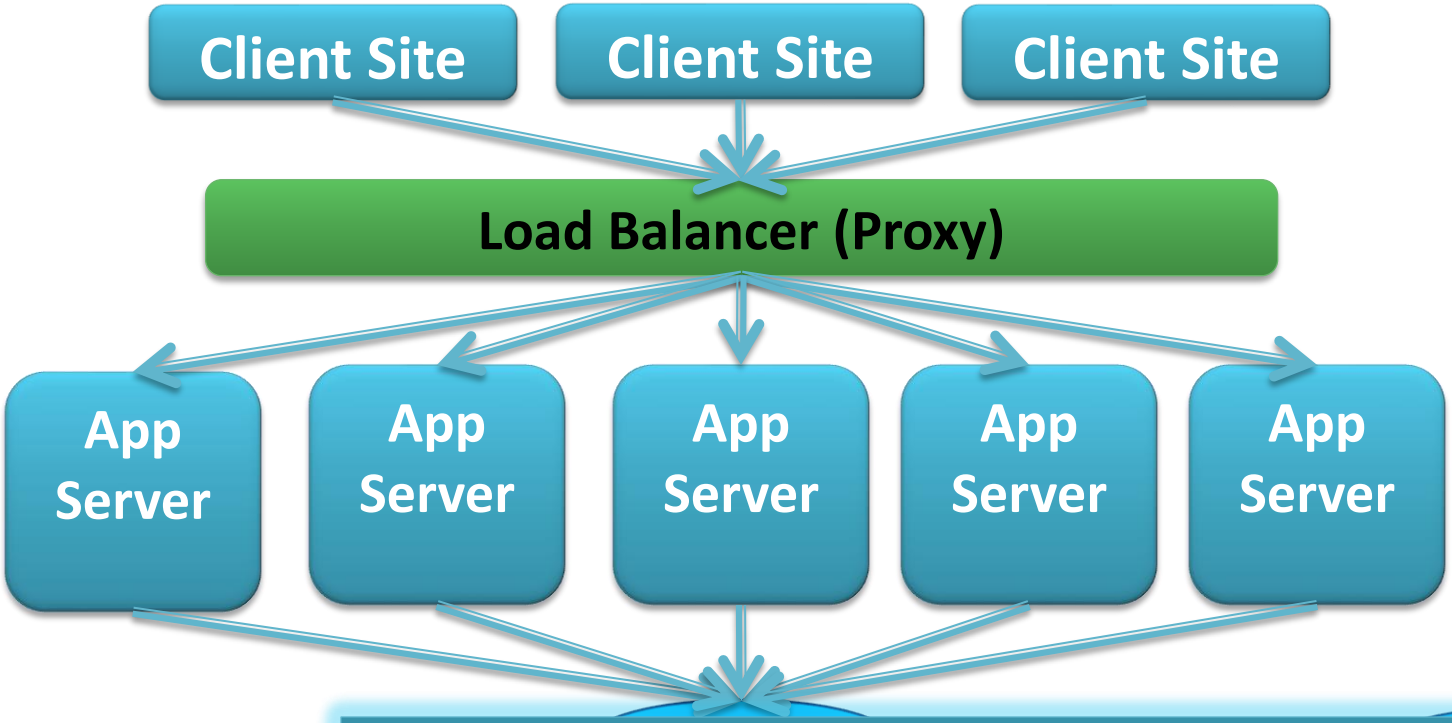
- **Data in the Cloud**
  - Platforms for Data Analysis
  - **Platforms for Update intensive workloads**
- Data Platforms for Large Applications
- Multitenant Data Platforms
- Open Research Challenges

Platforms for Update intensive workloads

- Most enterprise solutions are based on RDBMS technology.
- Significant Operational Challenges:
  - Provisioning for Peak Demand
  - Resource under-utilization
  - Capacity planning: too many variables
  - Storage management: a massive challenge
  - System upgrades: extremely time-consuming
  - Complex mine-field of software and hardware licensing

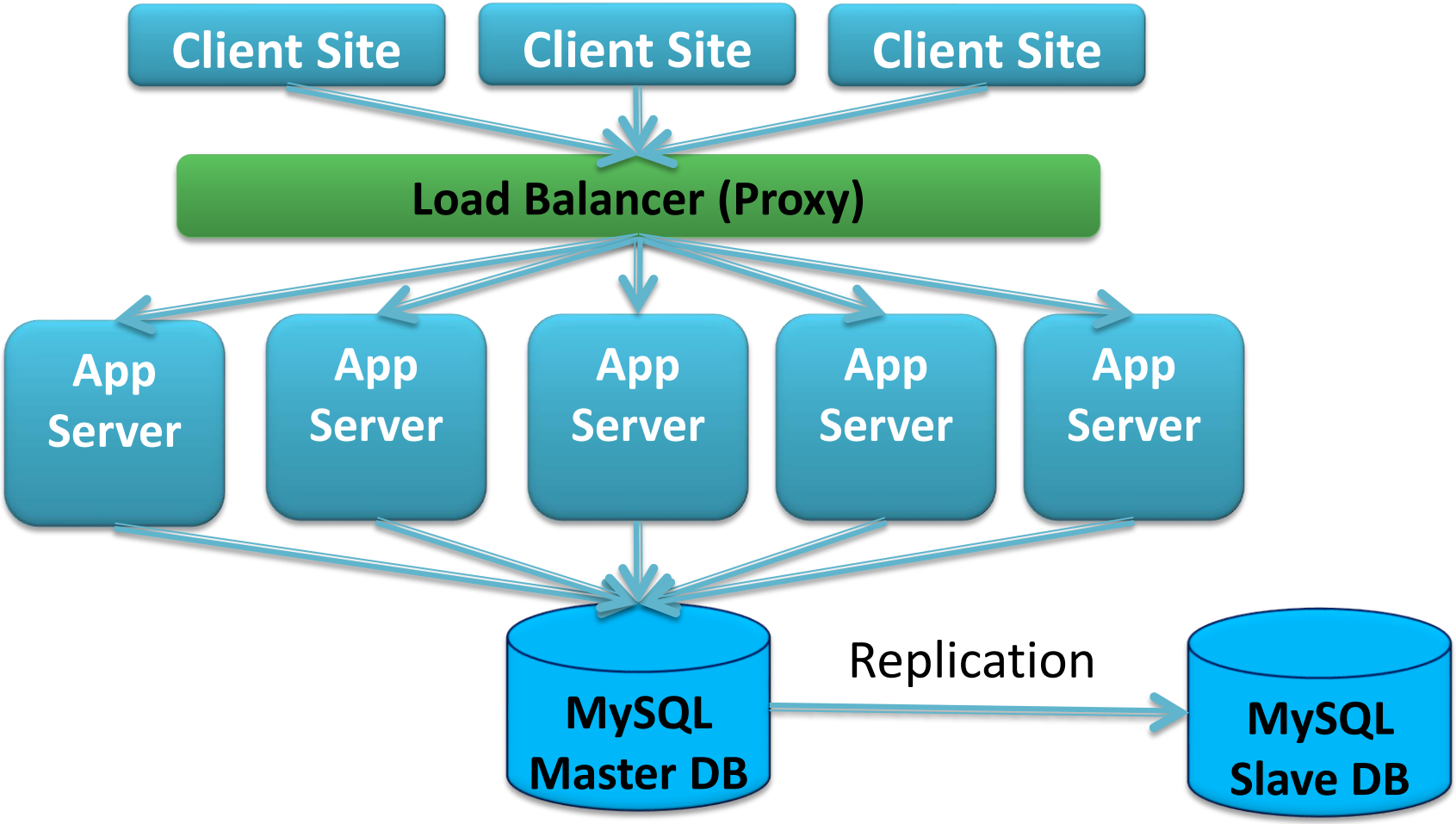
➔ Unproductive use of people-resources from a company's perspective

Scaling in the Cloud

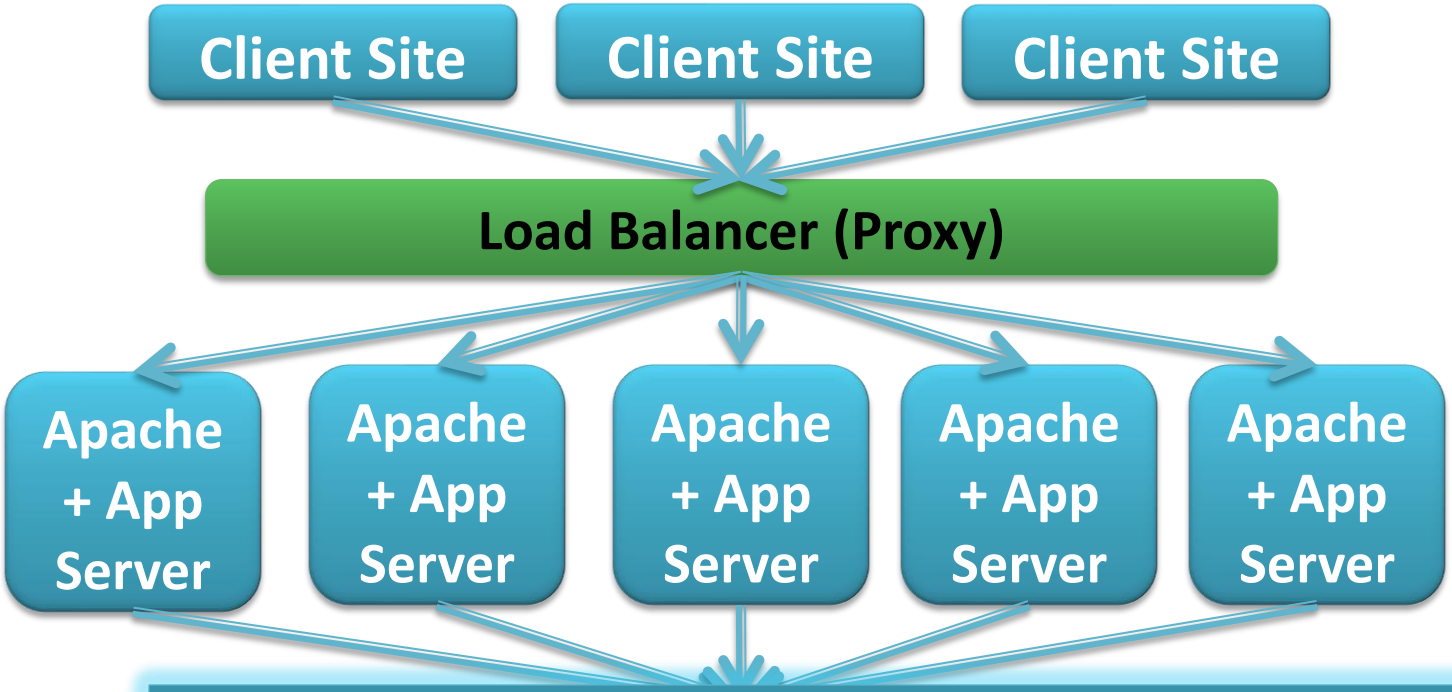


Database becomes the Scalability Bottleneck
Cannot leverage elasticity

Scaling in the Cloud



Scaling in the Cloud



Scalable and Elastic, but limited consistency and operational flexibility

{name: "mongo", type:"DB"}

BLOG Wisdom

- "If you want vast, on-demand scalability, you need a non-relational database." Since scalability requirements:
  - Can change very quickly and,
  - Can grow very rapidly.
- Difficult to manage with a single in-house RDBMS server.
- RDBMS scale well:
  - When limited to a single node, but
  - Overwhelming complexity to scale on multiple server nodes.

The "NoSQL" movement

- Initially used for: "*Open-Source relational database that did not expose SQL interface*"
- Popularly used for: "*non-relational, distributed data stores that often did not attempt to provide ACID guarantees*"
- Gained widespread popularity through a number of open source projects
  - HBase, Cassandra, Voldemort, MongDB, …
- Scale-out, elasticity, flexible data model, high availability

NoSQL has no relation with SQL

- Micheal Stonebraker [CACM Blog]

- Term heavily used (and abused)
- Scalability and performance bottleneck not inherent to SQL
  - Scale-out, auto-partitioning, self-manageability can be achieved with SQL
- Different implementations of SQL engine for different application needs
- SQL provides flexibility, portability

No-SQL → Not Only SQL

- Recently renamed
- Encompass a broad category of "structured" storage solutions
  - RDBMS is a subset
  - Key Value stores
  - Document stores
  - Graph database
- The debate on appropriate characterization continues

Cloud Data Management Desiderata
SQL or NoSQL

- Scalability
- Elasticity
- Fault tolerance
- Self Manageability
- **Sacrifice consistency?**

Perspectives (James Hamilton)

I love **eventual consistency** but there are some applications that are much easier to implement with strong consistency. Many like eventual consistency because it allows us to scale-out nearly without bound ***but it does come with a cost in programming model complexity***.

February 24, 2010

Quest for Scalable, Fault-tolerant, and Consistent Data Management in the Cloud that provides

ELASTICITY

Outline

- Data in the Cloud
- **Data Platforms for Large Applications**
  - **Key value Stores**
  - Transactional support in the cloud
- Multitenant Data Platforms
- Concluding Remarks

Key Value Stores

- Key-Valued data model
  - Key is the unique identifier
  - Key is the granularity for consistent access
  - Value can be structured or unstructured
- Gained widespread popularity
  - In house: **Bigtable** (Google), **PNUTS** (Yahoo!), **Dynamo** (Amazon)
  - Open source: **HBase, Hypertable, Cassandra, Voldemort**
- Popular choice for the modern breed of web-applications

Important Design Goals

- **Scale out: designed for scale**
  - Commodity hardware
  - Low latency updates
  - Sustain high update/insert throughput
- **Elasticity – scale up and down with load**
- **High availability – downtime implies lost revenue**
  - Replication (with multi-mastering)
  - Geographic replication
  - Automated failure recovery

Lower Priorities

- **No Complex querying functionality**
  - No support for SQL
  - CRUD operations through database specific API
- **No support for joins**
  - Materialize simple join results in the relevant row
  - Give up normalization of data?
- **No support for transactions**
  - Most data stores support single row transactions
  - Tunable consistency and availability (e.g., Dynamo)

➔ **Achieve high scalability**

Interplay with CAP

- **C**onsistency, **A**vailability, and Network **P**artitions
  - Only have two of the three together
- Large scale operations – be prepared for network partitions
- Role of CAP – During a network partition, choose between Consistency and Availability
  - RDBMS choose *consistency*
  - Key Value stores choose *availability* [low replica consistency]

Why sacrifice Consistency?

- It is a simple solution
  - nobody understands what sacrificing P means
  - sacrificing A is unacceptable in the Web
  - possible to push the problem to app developer
- C not needed in many applications
  - Banks do not implement ACID (classic example wrong)
  - Airline reservation only transacts reads (Huh?)
  - MySQL et al. ship by default in lower isolation level
- Data is noisy and inconsistent anyway
  - making it, say, 1% worse does not matter

[Vogels, VLDB 2007]

C and A: In a Network Partition

- Dynamo – quorum based replication
  - Multi-mastering keys – Eventual Consistency
  - Tunable read and write quorums
  - Larger quorums – higher consistency, lower availability
  - Vector clocks to allow application supported reconciliation
- PNUTS – log based replication
  - Similar to log replay – reliable log multicast
  - Per record mastering – timeline consistency
  - Major outage might result in losing the tail of the log

Too many choices – Which system should I use?

Benchmarking Serving Systems

[Cooper et al., SOCC 2010]

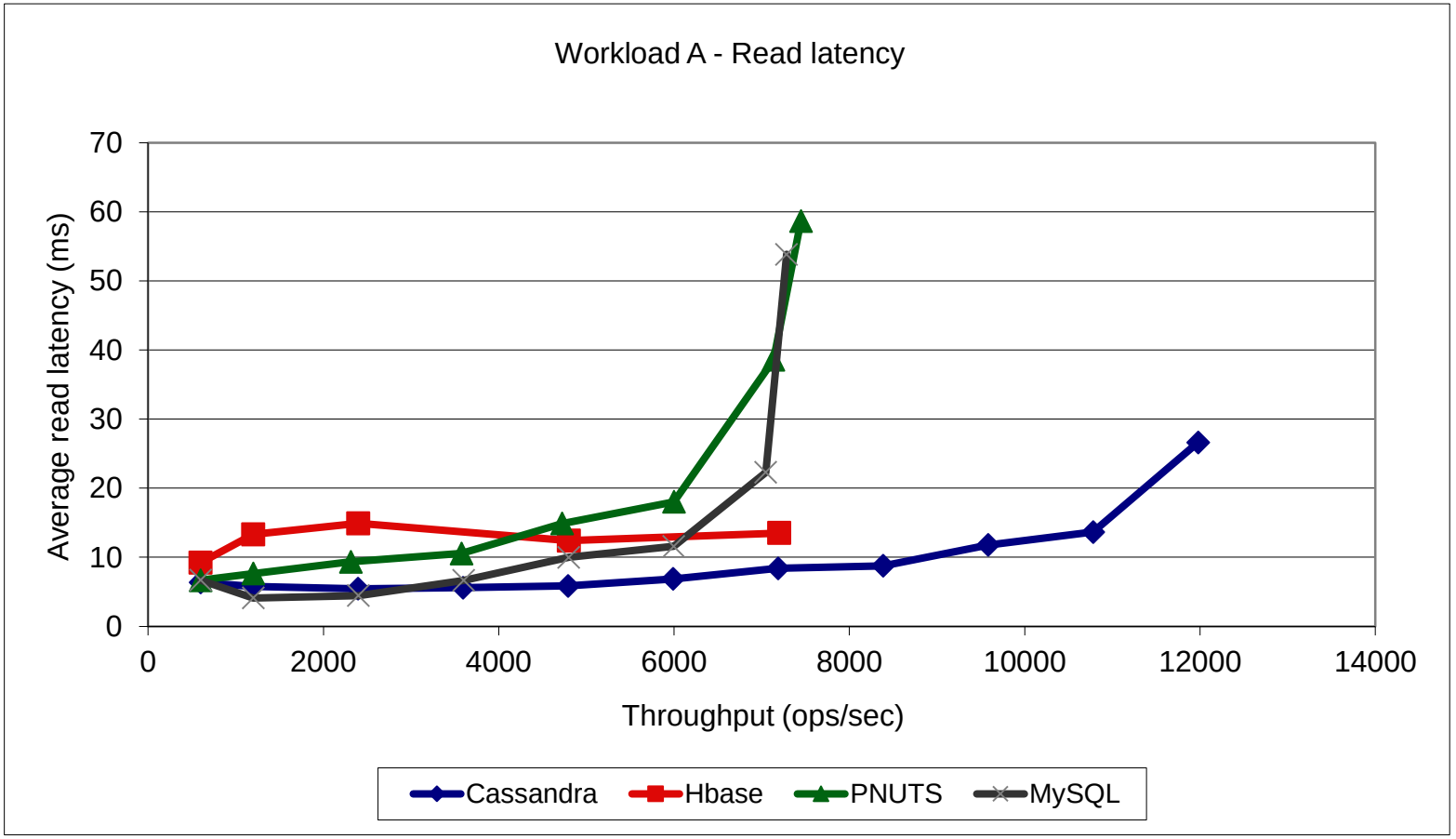
- A standard benchmarking tool for evaluating Key Value stores
- Evaluate different systems on common workloads
- Focus on performance and scale out

Benchmark tiers

- Tier 1 – Performance
  - Latency versus throughput as throughput increases
- Tier 2 – Scalability
  - Latency as database, system size increases
  - "Scale-out"
  - Latency as we elastically add servers
  - "Elastic speedup"

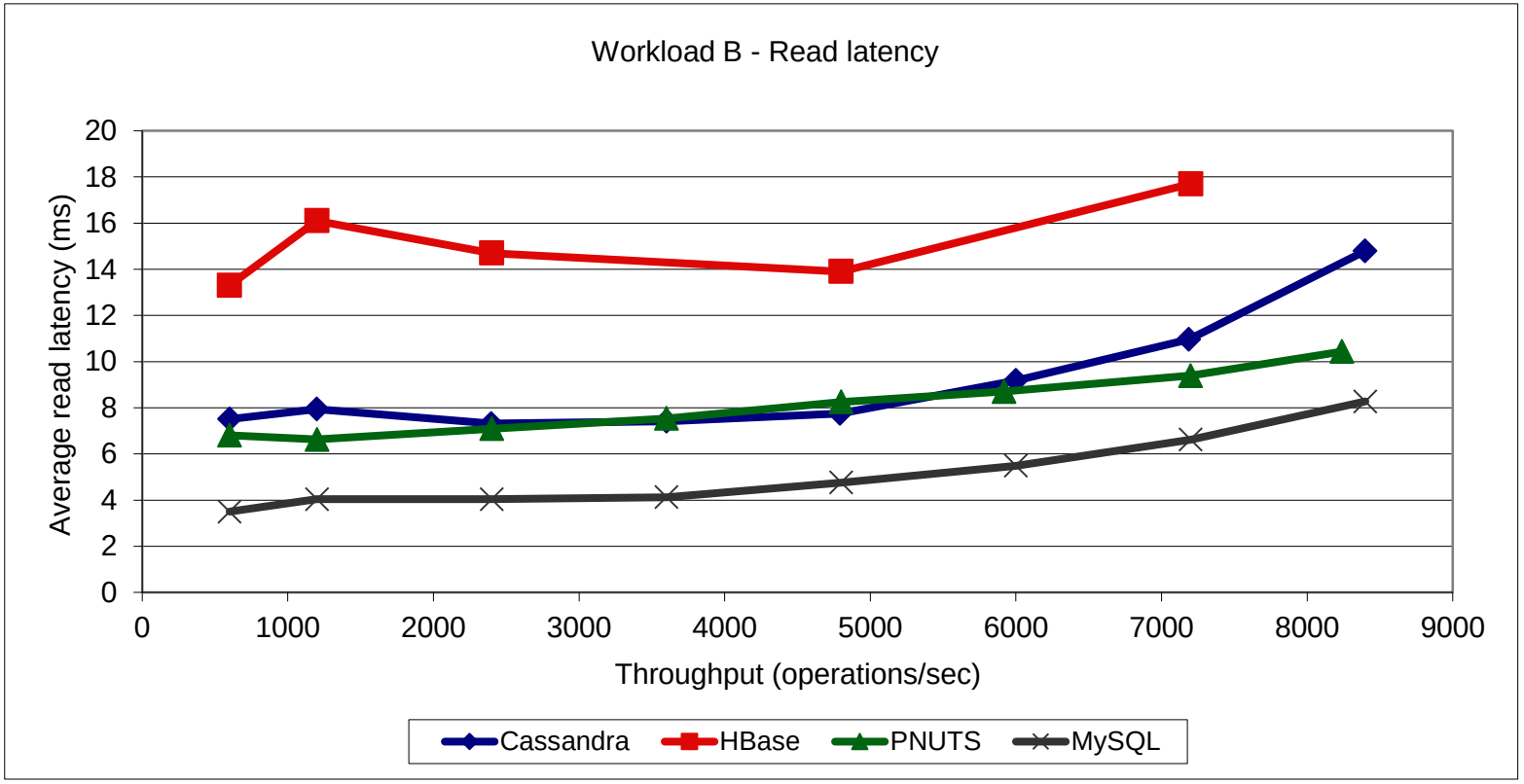
Workload A – Update heavy

- 50/50 Read/update



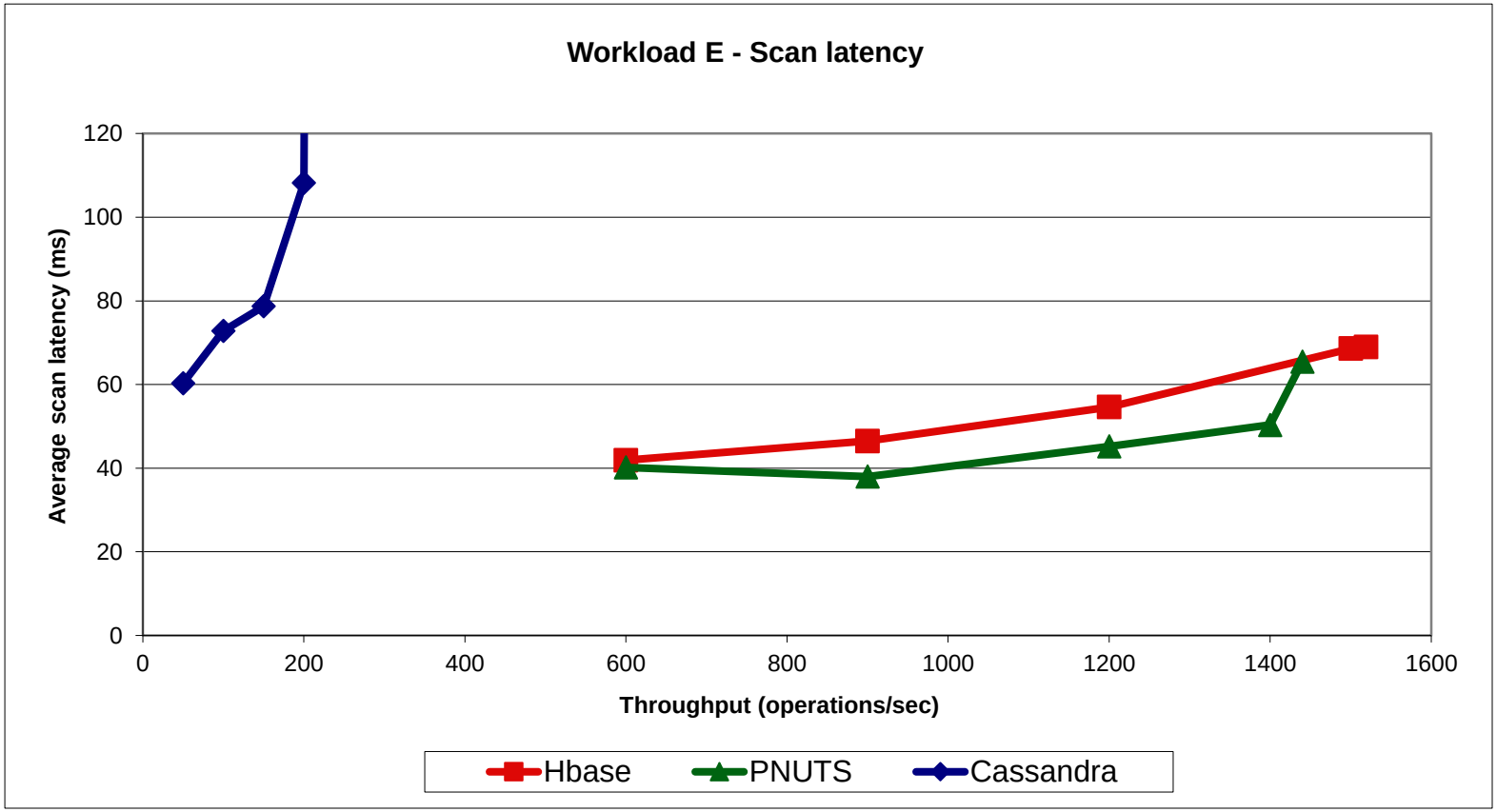
Workload B – Read heavy

- 95/5 Read/update



Workload E – short scans

- Scans of 1-100 records of size 1KB



Summary

- Different databases suitable for different workloads
- Evolving systems – landscape changing dramatically
- Active development community around open source systems