Tackling Big Data

Michael Cooper & Peter Mell
NIST Information Technology Laboratory
Computer Security Division

IT Laboratory Big Data Working Group

- What exactly is Big Data?
- What are the issues associated with it?
- What role should NIST play with regard to Big Data?

- What is the relationship between Big Data and IT Security?

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

What is Big Data?

- You know it when you see it .....
- NIST
  - Astronomical Image data from ALMA ~1Tb / day
  - Border Gateway Protocol (BGP) Data ~ 10 Tb
- Government
  - Census
  - NIH/ NCI
- Industry
  - Amazon
  - Google

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

What are the issues associated with Big Data?

- Taxonomies, ontologies, schemas, workflow
- Perspectives – backgrounds, use cases

- Bits – raw data formats and storage methods
- Cycles – algorithms and analysis
- Screws – infrastructure to support Big Data

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

IT Security and Big Data

- Big data sources become rich targets
- Composition of data in one large source as well as across sources
- Security data becoming the source for big data repositories
  - Log/event aggregation and correlation
  - IDS/IPS databases

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

NIST ITL Big Data Planned Activities

- ITL/SSD Big Data Workshop – 13 – 14 June
- NIST Internal workshop this summer
- Government/ industry / academia conference this fall

National Institute of Standards and Technology
U.S. Department of Commerce

An Overview of Big Data Technology and Security Implications

Peter Mell
Senior Computer Scientist
NIST Information Technology Laboratory
http://twitter.com/petermmell

Disclaimer

The ideas herein represent the author’s notional views on big data technology and do not necessarily represent the official opinion of NIST.

Any mention of commercial and not-for-profit entities, products, and technology is for informational purposes only; it does not imply recommendation or endorsement by NIST or usability for any specific purpose.

Presentation Outline

- Section 1: Introduction and Definitions
- Section 2: Big Data Taxonomies
- Section 3: Security Implications and Areas of Research
- Section 4: MapReduce and Hadoop
- Section 5: Notable Implementations
- Appendix A: Seminal Research Results
- Appendix B: Overview of Big Data Framework Types

Section 1: Introduction and Definitions

Big Data – the Data Deluge

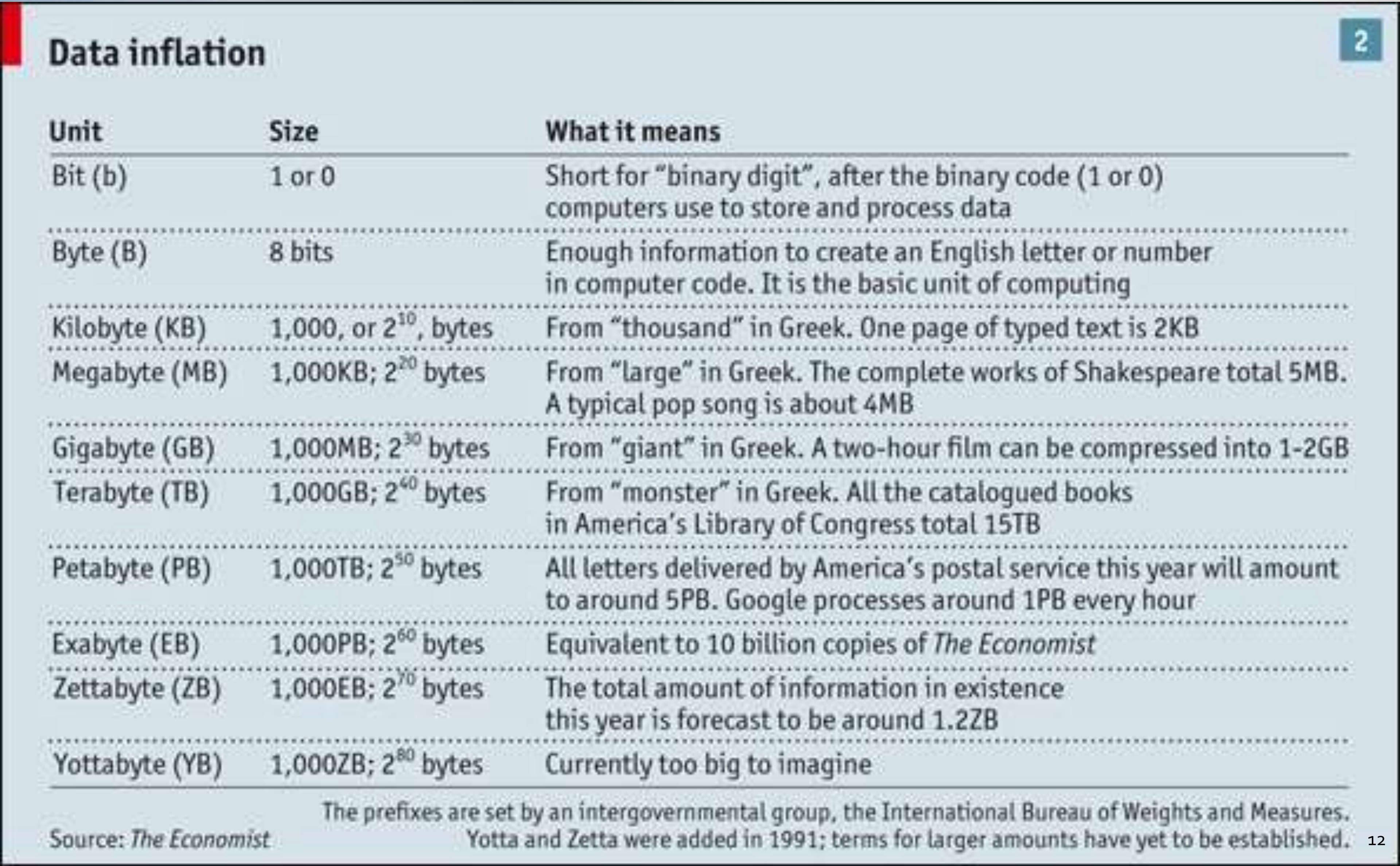
- The world is creating ever more data
  - (and it's a mainstream problem)

- Mankind created data
  - 150 exabytes in 2005
    - (exabyte is a billion gigabytes)
  - 1200 exabytes in 2010
  - 35000 exabytes in 2020 (expected by IBM)

- Examples:
  - U.S. drone aircraft sent back 24 years worth of video footage in 2009
  - Large Hadron Collider generates 40 terabytes/second
  - Bin Laden's death: 5106 tweets/second
  - Around 30 billion RFID tags produced/year
  - Oil drilling platforms have 20k to 40k sensors
  - Our world has 1 billion transistors/human

Credit: The data deluge, Economist; Understanding Big Data, Eaton et al.

A Quick Primer on Data Sizes



Data inflation 2

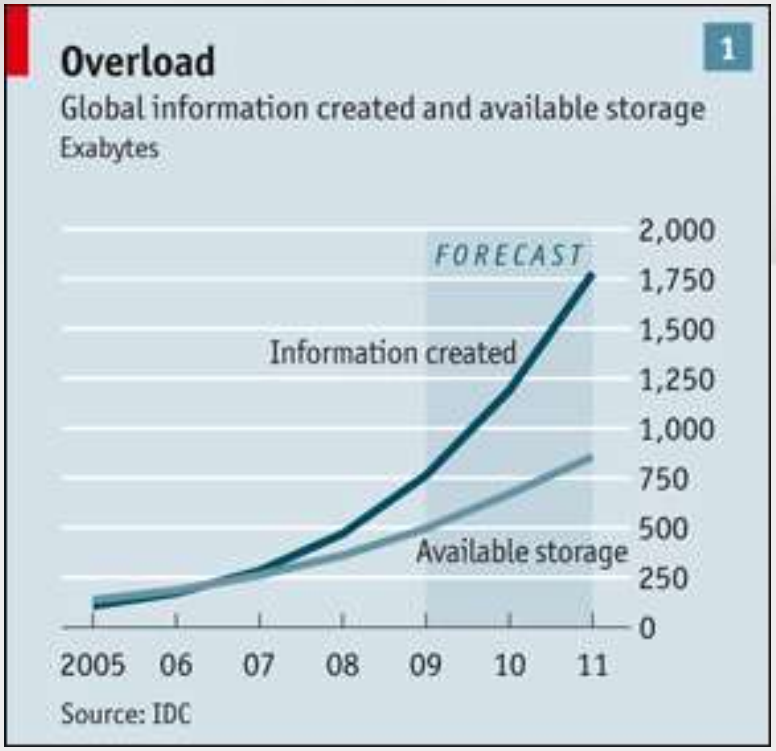
Unit	Size	What it means
Bit (b)	1 or 0	Short for "binary digit", after the binary code (1 or 0) computers use to store and process data
Byte (B)	8 bits	Enough information to create an English letter or number in computer code. It is the basic unit of computing
Kilobyte (KB)	1,000, or 2^{10}, bytes	From "thousand" in Greek. One page of typed text is 2KB
Megabyte (MB)	1,000KB; 2^{20} bytes	From "large" in Greek. The complete works of Shakespeare total 5MB. A typical pop song is about 4MB
Gigabyte (GB)	1,000MB; 2^{30} bytes	From "giant" in Greek. A two-hour film can be compressed into 1-2GB
Terabyte (TB)	1,000GB; 2^{40} bytes	From "monster" in Greek. All the catalogued books in America's Library of Congress total 15TB
Petabyte (PB)	1,000TB; 2^{50} bytes	All letters delivered by America's postal service this year will amount to around 5PB. Google processes around 1PB every hour
Exabyte (EB)	1,000PB; 2^{60} bytes	Equivalent to 10 billion copies of *The Economist*
Zettabyte (ZB)	1,000EB; 2^{70} bytes	The total amount of information in existence this year is forecast to be around 1.2ZB
Yottabyte (YB)	1,000ZB; 2^{80} bytes	Currently too big to imagine

Source: *The Economist*

The prefixes are set by an intergovernmental group, the International Bureau of Weights and Measures. Yotta and Zetta were added in 1991; terms for larger amounts have yet to be established.

Predictions of the "Industrial Revolution of Data" – Tim O'Reilly

- Data is the new "raw material of business" – Economist
- Challenges to achieving the revolution
  - It is not possible to store all the data we produce
  - 95% of created information was unstructured in 2010
- Key observation
  - Relational database management systems (RDBMS) will be challenged to scale up or out to meet the demand



Credit: Data data everywhere, Economist; Extracting Value from Chaos, Gantz et al.

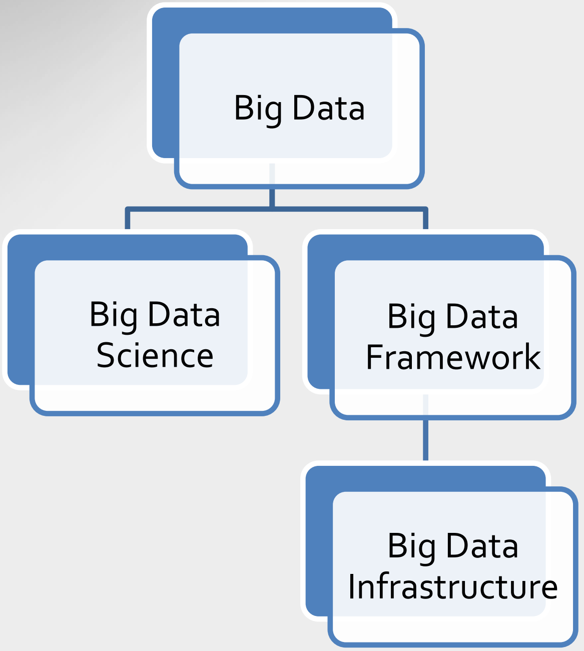
Industry Views on Big Data

- O’Reilly Radar definition:
  - Big data is when the size of the data itself becomes part of the problem
- EMC/IDC definition of big data:
  - *Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis.*
- IBM says that ”three characteristics define big data:”
  - Volume (Terabytes -> Zettabytes)
  - Variety (Structured -> Semi-structured -> Unstructured)
  - Velocity (Batch -> Streaming Data)
- Microsoft researchers use the same tuple

Credit: Big Data Now, Current Perspectives from O’Reilly Radar (O’Reilly definition); Extracting Value from Chaos, Gantz et al. (IDC definition); Understanding Big Data, Eaton et al. (IBM definition) ; The World According to LINQ, Meijer (Microsoft research)

Notional Definition for Big Data

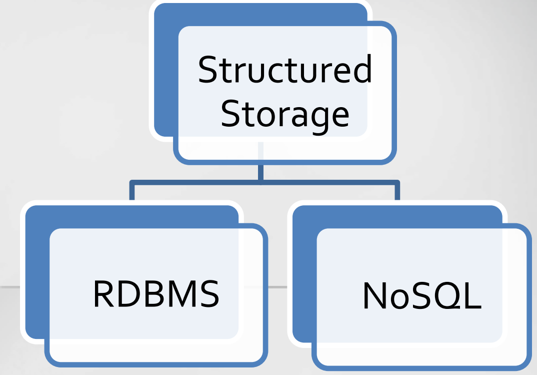
- Big Data
  - Big data is where the data volume, acquisition velocity, or data representation limits the ability to perform effective analysis using traditional relational approaches or requires the use of significant horizontal scaling for efficient processing.



More Notional Definitions

- Big Data Science
  - Big data science is the study of techniques covering the acquisition, conditioning, and evaluation of big data. These techniques are a synthesis of both information technology and mathematical approaches.
- Big Data Frameworks
  - Big data frameworks are software libraries along with their associated algorithms that enable distributed processing and analysis of big data problems across clusters of compute units (e.g., servers, CPUs, or GPUs).
- Big Data Infrastructure
  - Big data infrastructure is an instantiation of one or more big data frameworks that includes management interfaces, actual servers (physical or virtual), storage facilities, networking, and possibly back-up systems. Big data infrastructure can be instantiated to solve specific big data problems or to serve as a general purpose analysis and processing engine.

Big Data Frameworks are often associated with the term NoSQL



- NoSQL Origins
  - First used in 1998 to mean "No to SQL"
  - Reused in 2009 when it came to mean "Not Only SQL"
  - Groups non-relational approaches under a single term
- The power of SQL is not needed in all problems
  - Specialized solutions may be faster or more scalable
  - NoSQL generally has less querying power than SQL
- Common reasons to use NoSQL
  - Ability to handle semi-structured and unstructured data
  - Horizontal scalability
- NoSQL may complement RDBMS (but sometimes replaces)
  - RDBMS may hold smaller amounts of high-value structured data
  - NoSQL may hold vast amounts of less valued and less structured data

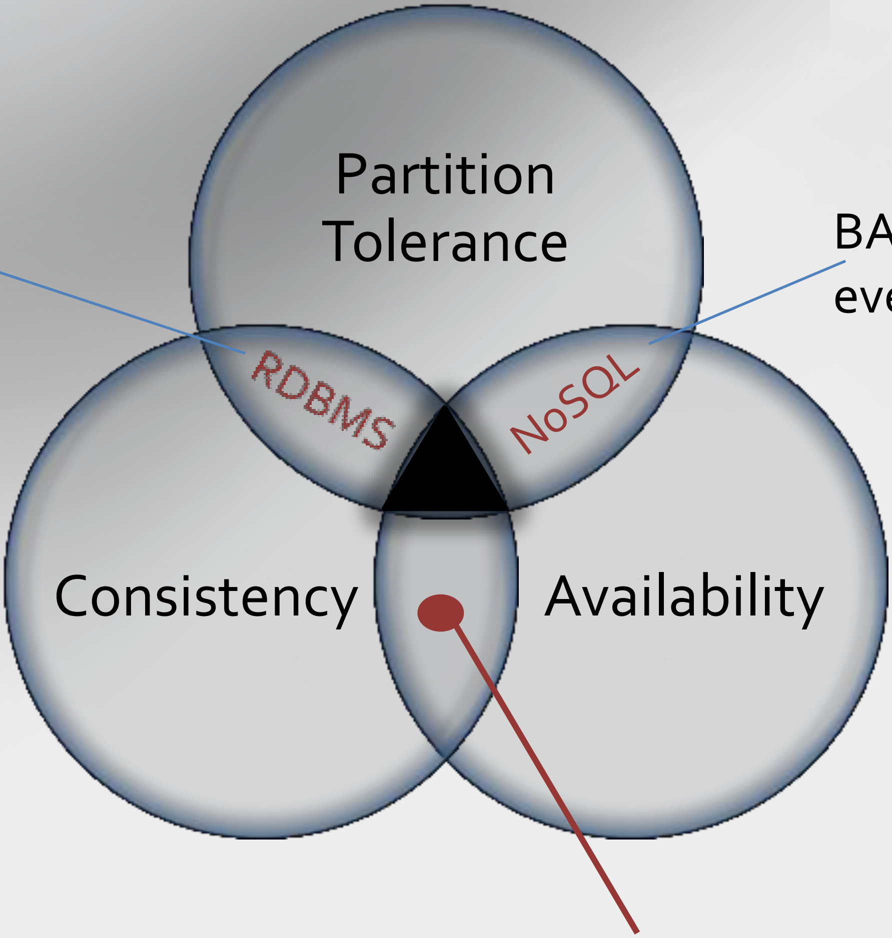
Credit: NoSQL Databases, Strauch; Understanding Big Data, Eaton et al.

Common Tradeoffs Between Relational and NoSQL Approaches

- Relational implementations provide ACID guarantees
  - **Atomicity**: transaction treated an all or nothing operation
  - **Consistency**: database values correct before and after
  - **Isolation**: events within transaction hidden from others
  - **Durability**: results will survive subsequent malfunction
- NoSQL often provides BASE
  - **Basically available**: Allowance for parts of a system to fail
  - **Soft state**: An object may have multiple simultaneous values
  - **Eventually consistent**: Consistency achieved over time
- CAP Theorem
  - It is impossible to have **consistency**, **availability**, and **partition tolerance** in a distributed system
  - the actual theorem is more complicated (see CAP slide in appendix A)

Credit: Principles of Transaction-Oriented Database Recovery, Haerder and Reuter, 1983; Base: An ACID Alternative, Pritchett, 2008;
Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services, Gilbert and Lynch

CAP Theorem with ACID and BASE Visualized



ACID with eventual availability

BASE with eventual consistency

Small data sets can be both consistent and available

Section 2: Big Data Taxonomies

Big Data Characteristics and Derivation of a Notional Taxonomy

Volume	Velocity	Variety (semi-structured or unstructured)	Requires Horizontal Scalability	Relational Limitation	Big Data
No	No	No	No	No	No
No	No	Yes	No	Yes	Yes, Type 1
No	Yes	No	Yes	Maybe	Yes, Type 2
No	Yes	Yes	Yes	Yes	Yes, Type 3
Yes	No	No	Yes	Maybe	Yes, Type 2
Yes	No	Yes	Yes	Yes	Yes, Type3
Yes	Yes	No	Yes	Maybe	Yes, Type 2
Yes	Yes	Yes	Yes	Yes	Yes, Type 3

Types of Big Data:
Type 1: This is where a non-relational data representation required for effective analysis.
Type 2: This is where horizontal scalability is required for efficient processing.
Type 3: This is where a non-relational data representation processed with a horizontally scalable solution is required for both effective analysis and efficient processing.
In other words, the data representation is not conducive to a relational algebraic analysis.

NoSQL Taxonomies

- Remember that big data frameworks and NoSQL are related but not necessarily the same
  - some big data problems may be solved relationally

- Scofield: Key/value, column, document, graph
- Cattel: Key/value, extensible record (e.g., column), document
- Strauch: Key/value, column, document (mentions graph separately)
- Others exist with very different categories

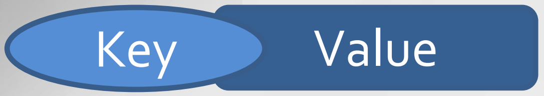
- Consensus taxonomy for NoSQL:
  - Key/value, column, document, graph
- Notional big data framework taxonomy:
  - Key/value, column, document, graph, sharded RDBMSs

Credit: NoSQL Databases, Strauch; NoSQL Death to Relational Databases(?), Scofield; Scalable SQL and NoSQL Data Stores, Cattel

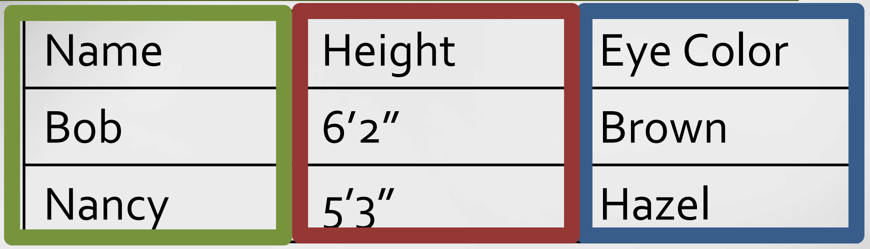
Notional Big Data Framework Taxonomy

Conceptual Structures:

Key Value Stores
- Schema-less system

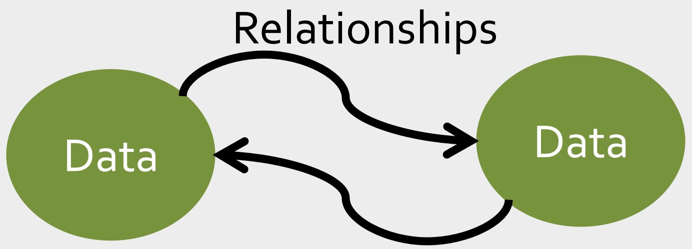


Column-oriented databases
- Storage by column, not row

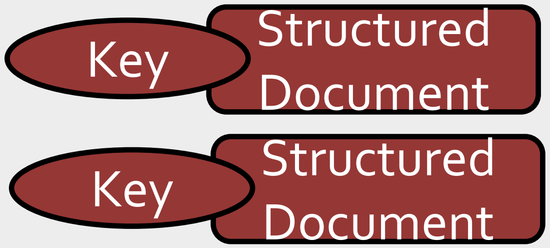


Name	Height	Eye Color
Bob	6’2”	Brown
Nancy	5’3”	Hazel

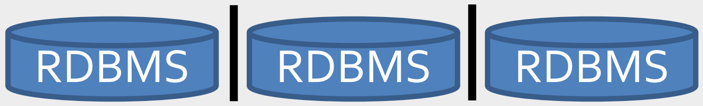
Graph Databases
- Uses nodes and edges to represent data
- Often used for Semantic Web



Document Oriented Database
- Stores documents that are semi-structured
- Includes XML databases



Sharded RDBMS



Comparison of NoSQL and Relational Approaches

	Performance	Horizontal Scalability	Flexibility in Data Variety	Complexity of Operation	Functionality
Key-Value stores	high	high	high	none	variable (none)
Column stores	high	high	moderate	low	minimal
Document stores	high	variable (high)	high	low	variable (low)
Graph databases	variable	variable	high	high	graph theory
Relational databases	variable	variable	low	moderate	relational algebra

Matches columns on the big data taxonomy

Credit: NoSQL Death to Relational Databases(?), Scofield (column headings modified from original data for clarity)

Notional Suitability of Big Data Frameworks for types of Big Data Problems

	Horizontal Scalability	Flexibility in Data Variety	Appropriate Big Data Types
Key-Value stores	high	high	1, 2, 3
Column stores	high	moderate	1 (partially), 2, 3 (partially)
Document stores	variable (high)	high	1, 2 (likely), 3 (likely)
Graph databases	variable	high	1, 2 (maybe), 3 (maybe)
Sharded Database	variable (high)	low	2 (likely)

Section 3: Security Implications and Areas of Research

Hypothesis: Big Data approaches will open up new avenues of IT security metrology

- "Revolutions in science have often been preceded by revolutions in measurement," - Sinan Aral, New York University

- Arthur Coviello, Chairman RSA
  - "Security must adopt a big data view… The age of big data has arrived in security management."
  - We must collect data throughout the enterprise, not just logs
  - We must provide context and perform real time analysis

- There is precious little information on how to do this

Credit: Data, data everywhere, Economist; http://searchcloudsecurity.techtarget.com/news/2240111123/Coviello-talks-about-building-a-trusted-cloud-resilient-security

Big Data is Moving into IT Security Products

- Several years ago, some security companies had an epiphany:
  - Traditional relational implementations were not always keeping up with data demands
- A changed industry:
  - Some were able to stick with traditional relational approaches
  - Some partitioned their data and used multiple relational silos
  - Some quietly switched over to NoSQL approaches
  - Some adopted a hybrid approach, putting high value data in a relational store and lower value data in NoSQL stores

Credit: This is based on my discussions with IT security companies in 12/2011 at the Government Technology Research Alliance Security Council 2011

Security Features are Slowly Moving into Big Data Implementations

- Many big data systems were not designed with security in mind – Tim Mather, KPMG
- There are far more security controls for relational systems than for NoSQL systems
  - **SQL security**: secure configuration management, multifactor authentication, data classification, data encryption, consolidated auditing/reporting, database firewalls, vulnerability assessment scanners
  - **NoSQL security**: cell-level access labels, kerberos-based authentication, access control lists for tables/column families

Credit: Securing Big Data, Cloud Security Alliance Congress 2011, Tim Mather KPMG

Public Government Big Data Security Research Exists

- Accumulo
  - Accumulo is a distributed key/value store that provides expressive, cell-level access labels.
  - Allows fine grained access control in a NoSQL implementation
  - Based on Google BigTable
  - 200,000 lines of Java code
  - Submitted by the NSA to the Apache Foundation

Credit: http://wiki.apache.org/incubator/AccumuloProposal, http://www.informationweek.com/news/government/enterprise-apps/231600835

What further research needs to be conducted on big data security and privacy?

- Enhancing IT security metrology
- Enabling secure implementations
- Privacy concerns on use of big data technology

Research Area 1: The Computer Science of Big Data

1. What is a definition of big data?
   - What computer science properties are we trying to instantiate?
2. What types of big data frameworks exist?
   - Can we identify a taxonomy that relates them hierarchically?
3. What are the strengths, weaknesses, and appropriateness of big data frameworks for specific classes of problems?
   - What are the mathematical foundations for big data frameworks?
4. How can we measure the consistency provided by a big data solution?
5. Can we define standard for querying big data solutions?

With an understanding of the capabilities available and their suitability for types of problems, we can then apply this knowledge to computer security.

Research Area 2: Furthering IT Security Metrology through Big Data Technology

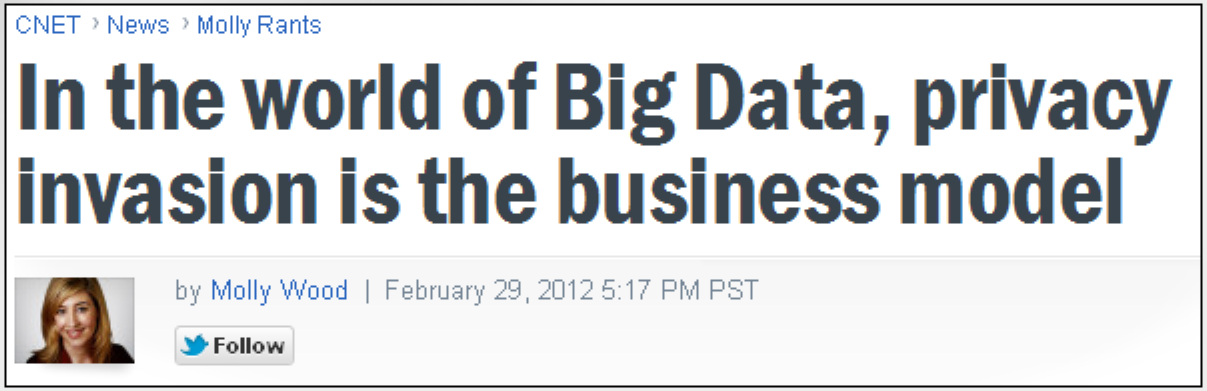
1. Determine how IT security metrology is limited by traditional data representations (i.e., highly structured relational storage)
2. Investigate how big data frameworks can benefit IT security measurement
   - What new metrics could be available?
3. Identify specific security problems that can benefit from big data approaches
   - Conduct experiments to test solving identified problems
4. Explore the use of big data frameworks within existing security products
   - What new capabilities are available?
   - How has this changed processing capacity?

Research Area 3: The Security of Big Data Infrastructure

1. Evaluate the security capabilities of big data infrastructure
   - Do the available tools provide needed security features?
   - What security models can be used when implementing big data infrastructure?
2. Identify techniques to enhance security in big data frameworks (e.g., data tagging approaches, sHadoop)
   - Conduct experiments on enhanced security framework implementations

Research Area 4: The Privacy of Big Data Implementations

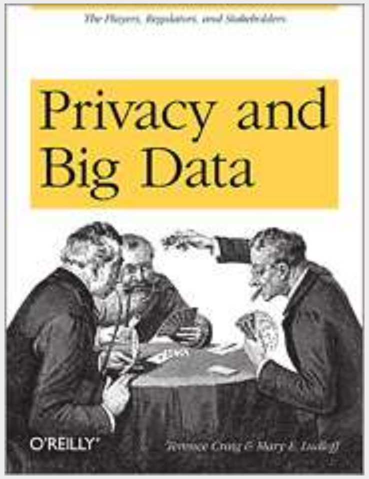
- Big data technology enables massive data aggregation beyond what has been previously possible
- Inferencing concerns with non-sensitive data
- Legal foundations for privacy in data aggregation
- Application of NIST Special Publication 800-53 privacy controls


CNET › News › Molly Rants

In the world of Big Data, privacy invasion is the business model

by Molly Wood | February 29, 2012 5:17 PM PST

Follow


The Players, Regulators, and Stakeholders
Privacy and Big Data
O'REILLY
Terence Craig & Mary E. Ludloff

Needed Research Deliverables

- Area 1 (not security specific)
  - Publication on harnessing big data technology
    - Definitions, taxonomies, and appropriateness for classes of problems
- Area 2 (security specific)
  - Publication on furthering IT security metrology through big data technology
  - Research papers on solving specific security problems using big data approaches
- Area 3 (security specific)
  - Publication on approaches for the secure use of big data platforms
- Area 4 (privacy specific)
  - Not yet identified

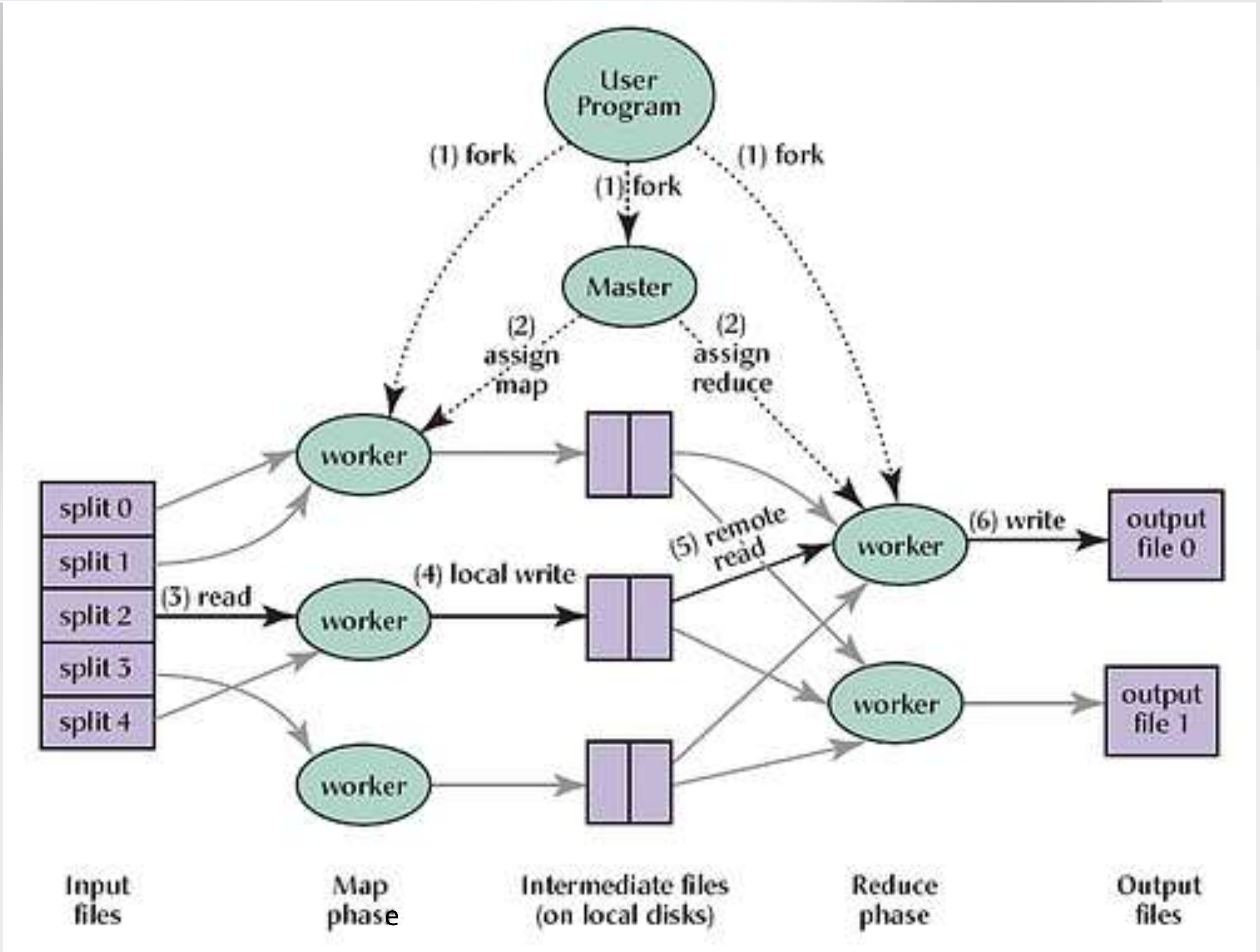
Section 4: MapReduce and Hadoop

MapReduce – Dean, et al.

- Seminal paper published by Google in 2004
  - Simple concurrent programming model and associated implementation
- Model handles the parallel processing and message passing details
  - Simplified coding model compared to general purpose parallel languages (e.g., MPI)
- Three functions: Map -> Parallel sort -> Reduce
  - Map: Processes a set of key/value pairs to produce an intermediate set of key/value pairs
  - Parallel sort: a distributed sort on intermediate results feeds the reduce nodes
  - Reduce: for each resultant key, it processes each key/value pair and produces the result set of values for each key
- Approachable programming model
  - Handles concurrency complexities for the user
  - Limited functionality
  - Appears to provide a sweet spot for solving a vast number of important problems with an easy to use programming model

Credit: MapReduce: Simplified Data Processing on Large Clusters, Dean et al.

MapReduce Diagram from Google's 2004 Seminal Paper



Storage, MapReduce, and Query (SMAQ) Stacks

Query

- Efficient way of defining computation
- Platform for user friendly analytical systems

Map Reduce

- Distributes computation over many servers
- Batch processing model

Storage

- Distributed and non-relational

Credit: 2011 O'Reilly Radar, Edd Dumbill

Hadoop

- Widely used MapReduce framework

- "The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model" – hadoop.apache.org

- Open source project with an ecosystem of products
- Core Hadoop:
  - Hadoop MapReduce implementation
  - Hadoop Distributed File System (HDFS)
- Non-core: Many related projects

Credit: http://hadoop.apache.org

Hadoop SMAQ Stack (select components)

Query

- Pig (simply query language)
- Hive (SQL like queries)
- Cascading (workflows)
- Mahout (machine learning)
- Zookeeper (coordination service)
- Hama (scientific computation)

Map Reduce

- Hadoop Map Reduce implementation

Storage

- HBase (column oriented database)
- Hadoop Distributed File System (HDFS, core Hadoop file system)

Alternate MapReduce Frameworks

- BashReduce
- Disco Project
- Spark
- GraphLab Carnegie-Mellon
- Storm
- HPCC (LexisNexis)

Section 5: Notable Implementations (both frameworks and infrastructure)

Google File System – Ghemawat, et al.

- Design requirements: "performance, scalability, reliability, and availability"
- Design assumptions:
  - Huge files
  - Expected component failures
  - File mutation is primarily by appending
  - Relaxed consistency <- think of the CAP theorem here
- Master has all its data in memory (consistent and available!!)
- All reads and writes occur directly between client and chunkservers
- For writes, control flow is decoupled from pipelined data flow

Credit: The Google File System, Ghemawat et al.

Google File System Architecture

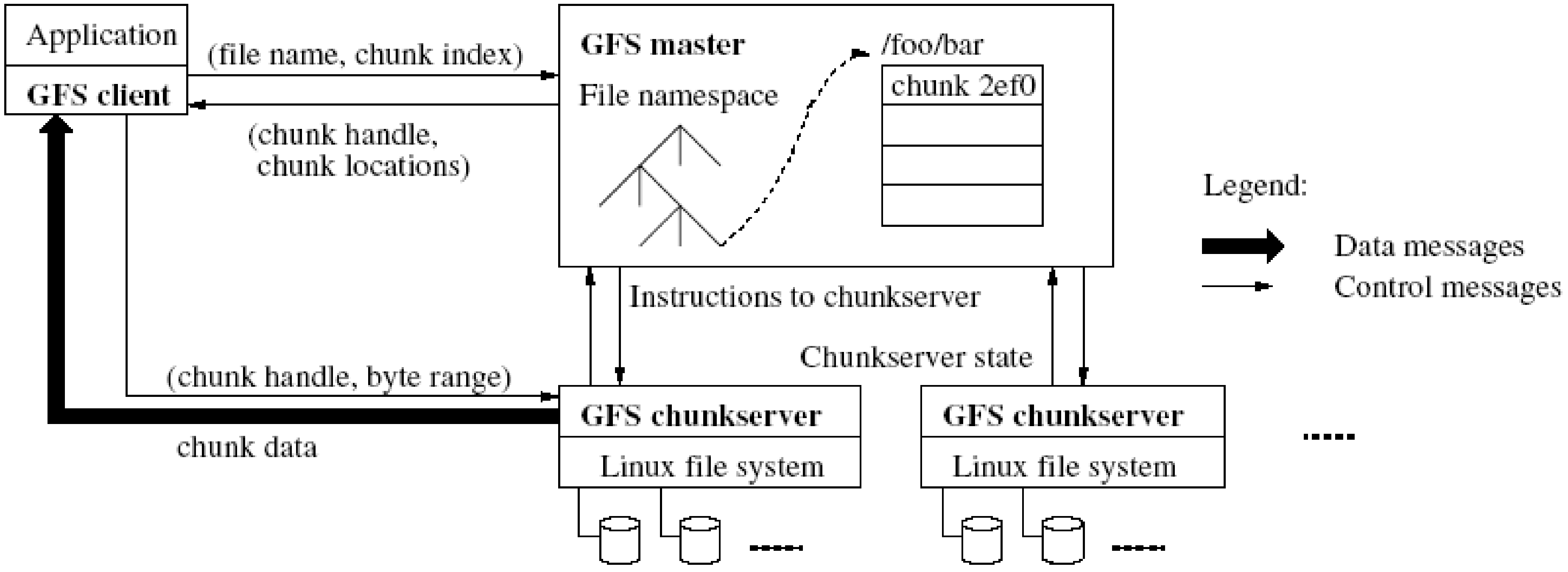


Figure 1: GFS Architecture

Google File System Write Control and Data Flow

- Master assigned a primary replica
- Client pipelines data through replicas
- Client contacts primary to instantiate the write

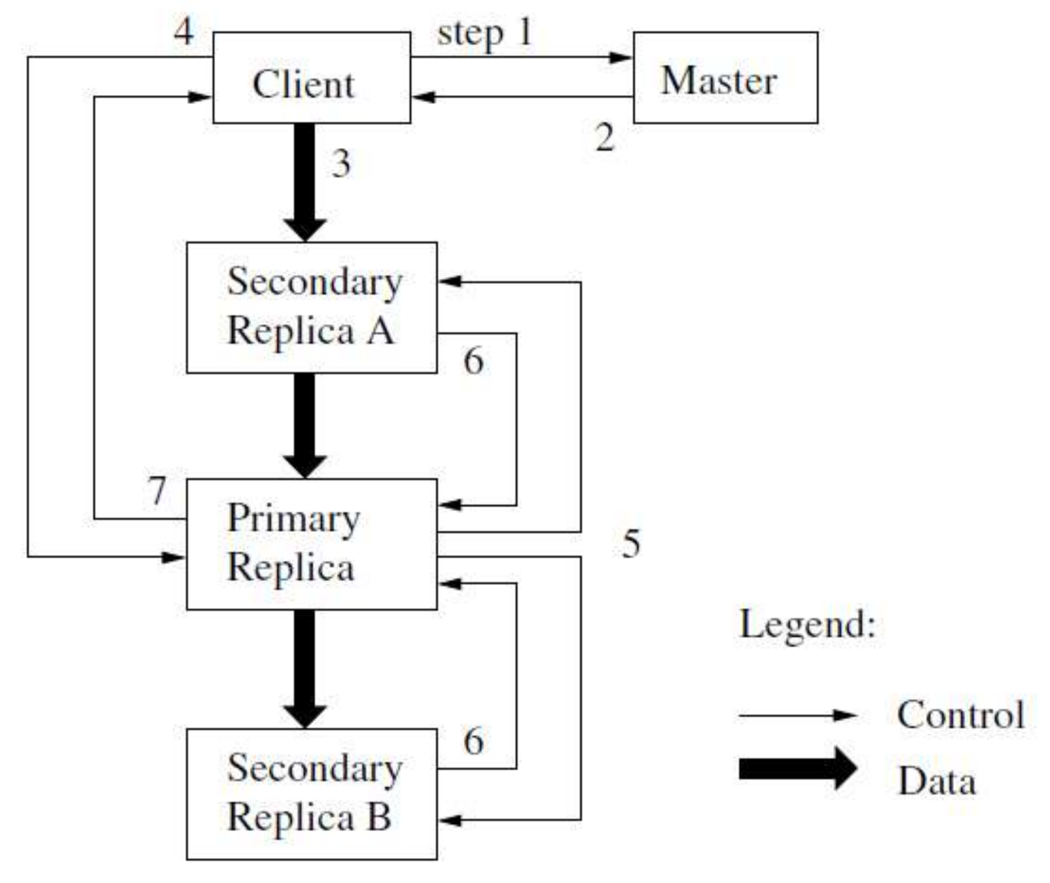


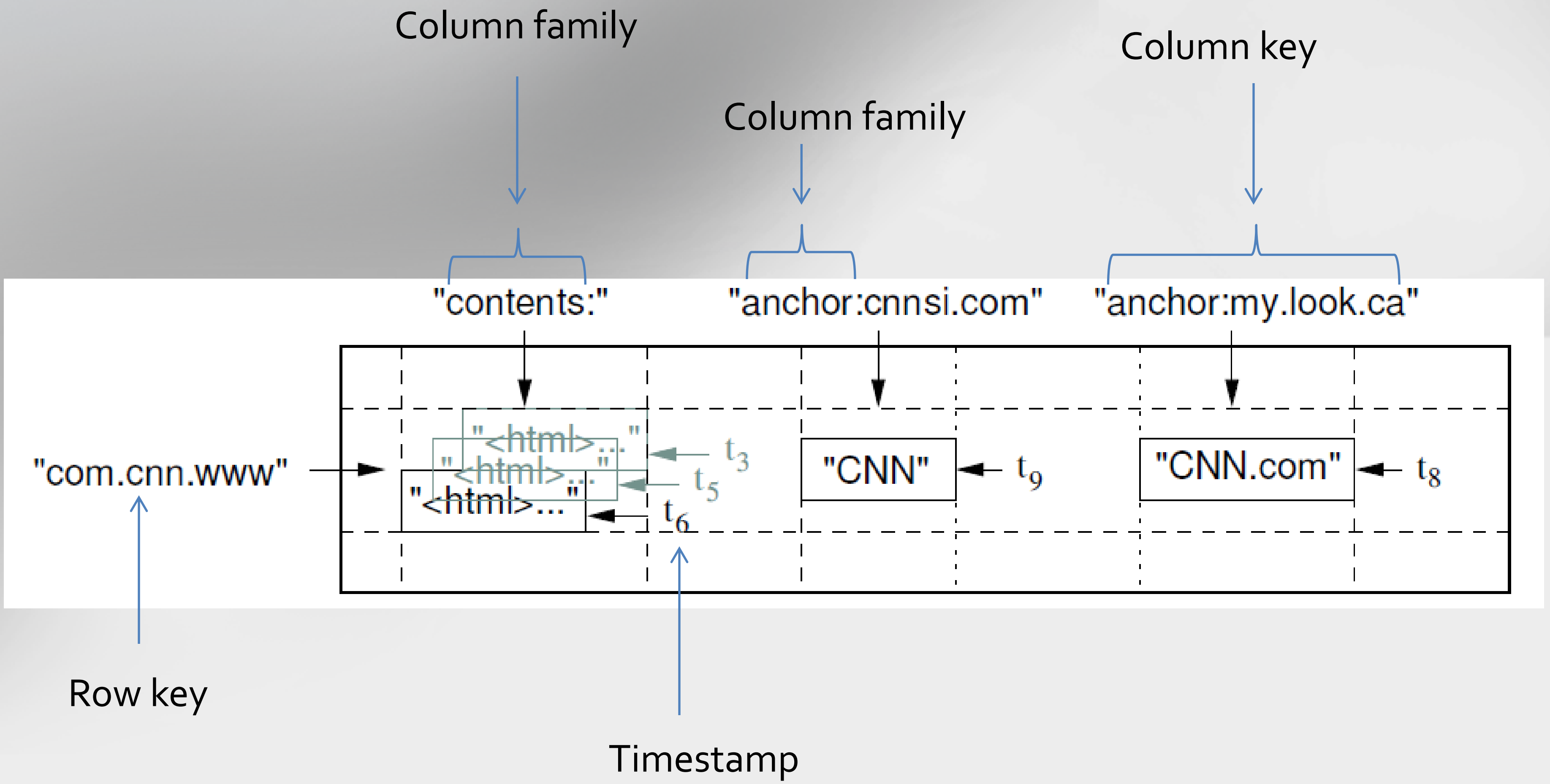
Figure 2: Write Control and Data Flow

Big Table – Chang, Dean, Ghemawat, et al.

- "Big table is a sparse, distributed, persistent multi-dimensional sorted map"
  - It is a non-relational key-value store / column-oriented database
- Row keys- table data is stored in row order
  - Model supports atomic row manipulation
- Tablets- subsets of tables (a range of the rows)
  - Unit of distribution/load balancing
- Column families group column keys
  - Access control done on column families
- Each cell can contain multiple time stamped versions
- Uses Google File System (GFS) for file storage
- Distributed lock service- Chubby
- Implementation- Lightly loaded master plus tablet servers
- Internal Google tool

Credit: Bigtable: A Distributed Storage System for Structured Data, Chang et. al.

Big Table Storage Paradigm



Hbase

- Open source Apache project
  - Modeled after the Big Table research paper
  - Implemented on top of the Hadoop Distributed File System

- "Use HBase when you need random, realtime read/write access to your Big Data… hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware. HBase is an open-source, distributed, versioned, column-oriented store modeled after Google's Bigtable"

Credit: http://hbase.apache.org

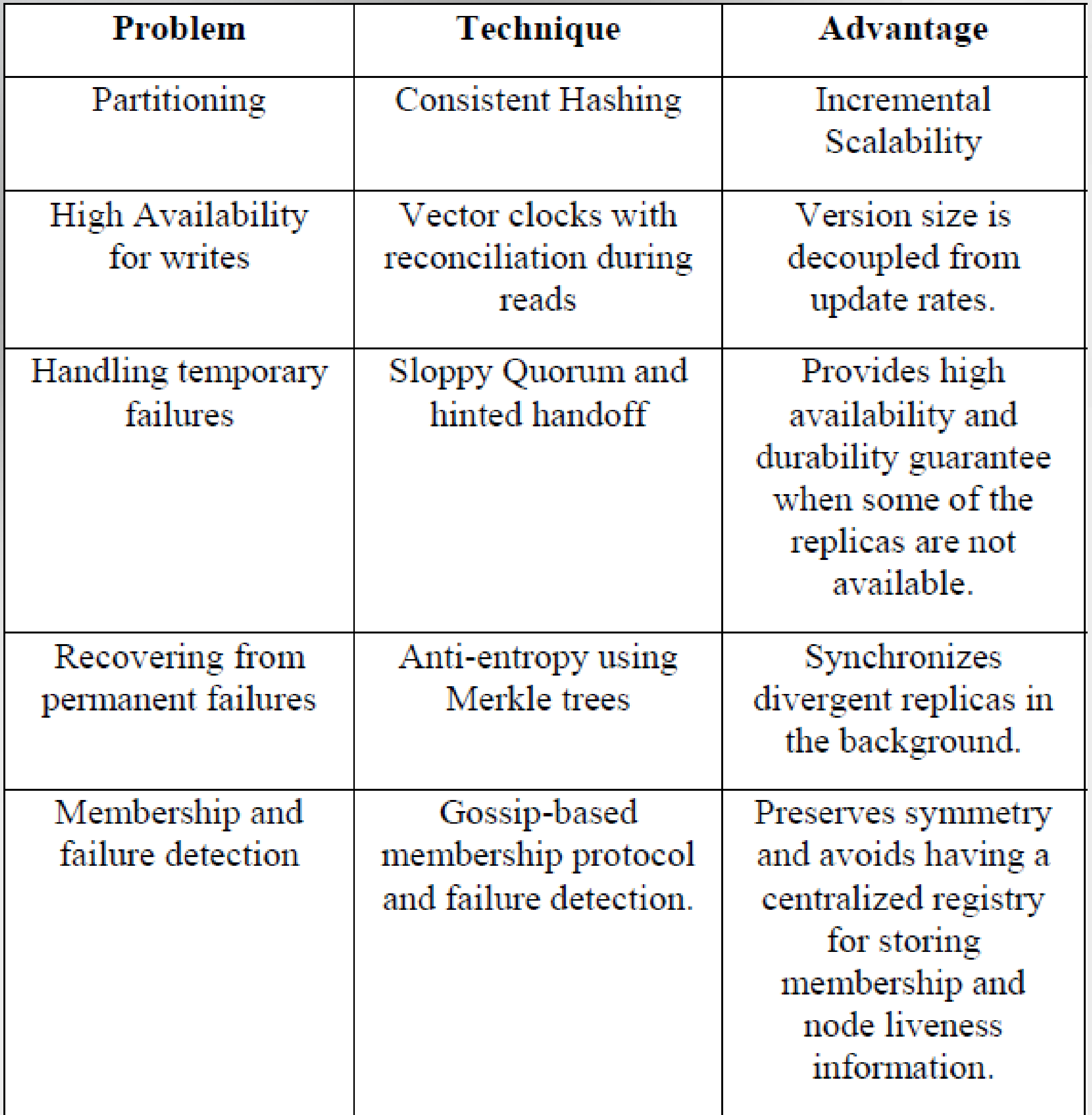
Dynamo – DeCandia, Hastorun,… , Vogels

Werner Vogels, Amazon CTO

- 2007 paper describing Amazon's implementation that had been in production use for one year
- Dynamo is a key-value eventually consistent database designed for scalability and high availability
  - Key lookup only (no relational schema or hierarchical namespace)
  - No central point of failure (i.e., nodes are symmetric)
  - Handles nodes of differing capability (i.e., heterogeneity)
  - Offers incremental scalability (i.e., you can add one node at a time)
  - Writes rarely rejected, reads may return multiple versions
- Strong focus on SLA performance guarantees (for 99.9% of operations)

Credit: Dynamo: Amazon's Highly Available Key-value Store, DeCandia et al.

Dynamo Techniques and Advantages



Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

Dynamo Configurability and Production Implementation

- "the primary advantage of Dynamo is that it provides the necessary knobs… to tune their instances" – Dynamo paper
- Users controls three variables:
  - N, number of hosts on which to replicate data
  - R, minimum number of hosts that must participate in a read
    - low value could increase inconsistency
  - W, minimum number of hosts that must participate in a write
    - low value could decrease durability

- Amazon Dynamo infrastructure
  - Dial-in the desired requests per second (elasticity)
  - Data stored in solid state drives for low latency access
  - Automatic replication across availability zones
  - Zero administration burden or even features
  - Developers choose between eventually consistent and strongly consistent reads

Apache Cassandra

- Described as a “big table model running on an Amazon dynamo-like infrastructure”
- Distributed database management system
  - No single point of failure
  - Designed for use with commodity servers
  - Key-value with column indexing system
    - “Rows” are keys that are randomly partitioned among servers
    - Keys maps to multiple values
    - Values from multiple keys may be grouped into “column families”
    - Each key’s values are stored together (like a row oriented RDBMS and yet the data has a column orientation)
  - Cassandra Query Language (SQL like querying)
- Initially developed by Facebook and then open sourced

Credit: http://cassandra.apache.org

Questions and Comments

Peter Mell
NIST
Senior Computer Scientist
301-975-5572
peter.mell@nist.gov
http://twitter.com/petermmell

Appendix A: Seminal Research Results

CAP Theorem – Brewer, 2000

- CAP: Consistency, Availability, Partition-tolerance
- It is impossible to have all three CAP properties in an asynchronous distributed read/write system
  - Asynchronous nature is key (i.e., no clocks)
  - Any two properties can be achieved
- Delayed-t consistency possible for partially synchronous systems (i.e., independent timers)
  - All three properties can be guaranteed if the requests are separated by a defined period of lossless messaging.
- 'most real world systems are forced to settle with returning "most of the data most of the time."'

- Take away: Distributed read/write systems are limited by intersystem messaging and may have to relax their CAP requirements

Credit: Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services, Gilbert and Lynch

BASE vs. ACID – Pritchett 2008

- CAP: Consistency, Availability, Partition Tolerance
- For distributed systems, we must have partitioning
- ACID (atomicity, consistency, isolation, durability)
  - Focuses on consistency, not availability
  - Properties are transparently provided by the database
- BASE (basically available, soft state, eventually consistent)
  - Focuses on availability, not consistency
  - Requires independent analysis of each application to achieve these properties (i.e., this is harder than ACID)
  - Promotes availability by avoiding two phase commits over the network
  - Application code may provide idempotence to avoid 2 phase commits

Credit: BASE: An Acid Alternative, Pritchett

Origin of SQL – Codd 1970

- Invented by Edgar F. Codd, IBM Fellow
- “A Relational Model of Data for Large Shared Data Banks” published in 1970
- Relational algebra provides a mathematical basis for database query languages (i.e. SQL)
- Based on first order logic

- “[His] basic idea was that relationships between data items should be based on the item's values, and not on separately specified linking or nesting. This notion greatly simplified the specification of queries and allowed unprecedented flexibility to exploit existing data sets in new ways“ - Don Chamberlin, SQL co-inventor

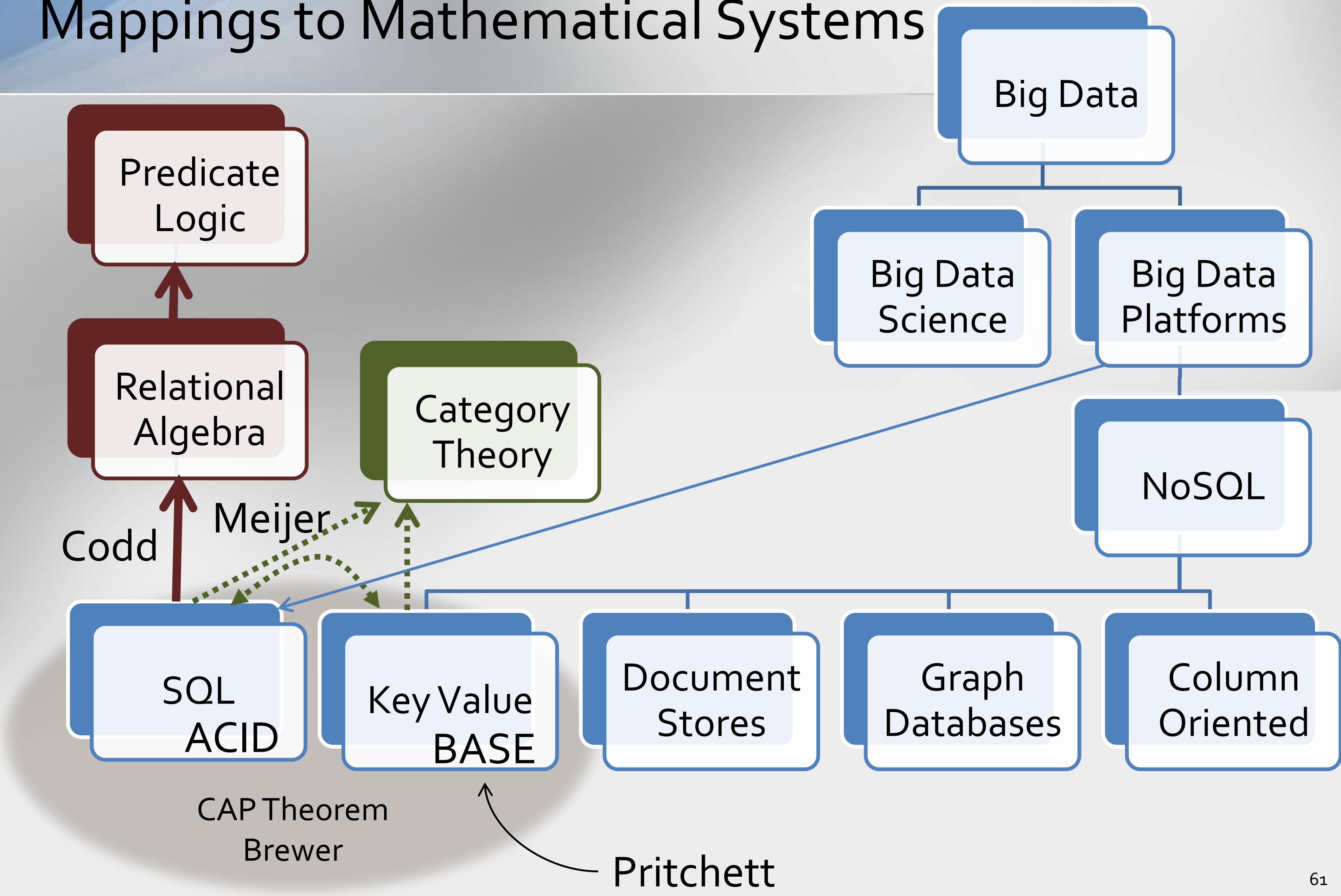
Credit: A Relational Model of Data for Large Shared Data Banks, Codd

The Notion of CoSQL – Meijer, et al. 2011

- SQL and key-value NoSQL (called coSQL) are mathematical duals based on category theory
- SQL and CoSQL can transmute into each other
- Duality between synchronous ACID and asynchronous BASE
  - Transmute data to take advantage of either ACID or BASE
- "monads and monad comprehensions provide a common query mechanism for both SQL and coSQL"
- Microsoft's Languge-Integrated Query (LINQ) implements this query abstraction
- There could be a single language, analogous to SQL, used for all coSQL databases!!

Credit: A co-Relational Model of Data for Large Shared Data Banks, Meijer

Notional Big Data Taxonomy w/ Mappings to Mathematical Systems



Appendix B: Overview of Big Data Framework Types

Overview – Key Value Stores

- Key value stores are distributed associative arrays (collections of key/value pairs)
  - Implementing maps or dictionaries
- Enables storage of schema-less data
- Simple operations:
  - add(key,value), set(key,value), get(key), delete(key)
- Values may be complex and unstructured objects
- Indexed on keys for searching
  - No joins, no SQL, no real queries
- Often a speed advantage in storage and retrieval

Credit: Key-Value stores: a practical overview, Seeger

Example Uses- Key Value Stores

- Key value is one of the most prominent NoSQL approaches
- Used for a wide variety of big data problems
- Most popular type is MapReduce
  - First used for indexing the Internet (Google)
- Example:
  - Visa used Hadoop to process 36 terabytes of data
  - Traditional approach: one month
  - Key value approach: 13 minutes

Overview – Column Oriented Databases

- Table data is stored by column NOT row
  - Efficient for calculating metrics on a particular set of columns
  - Efficient for updating all values in a single column
  - Inefficient for row operations effecting multiple columns
  - Inefficient for writing new rows
- Key concern is gaining efficiency in hard disk accesses for particular types of jobs
- Since column data is of the same type, some compression advantages can be achieved over row-oriented databases
- This concept is not new. It has been used since the 1970s

Credit: C-Store: A Column-oriented DBMS, Stonebraker

Example – Column Oriented Databases

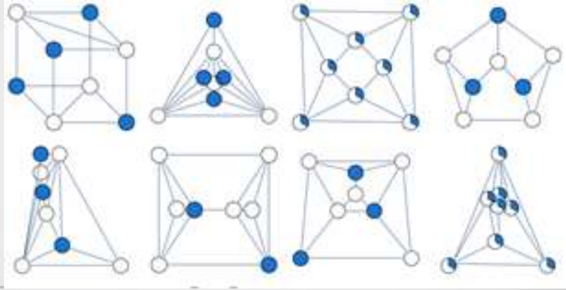
2010 Car	0-30 mph (sec)	Turning circle	Horsepower
Nissan Altima 2.5S	3.3	40 ft.	175
Mini Cooper Clubman	3.8	37 ft.	118
Smart	5.1	30 ft.	71

Column oriented storage:
Nissan Altima 2.5S, Mini Cooper Clubman, Smart
3.3, 3.8, 5.1
40 ft., 37 ft., 30 ft.
175, 118, 72

Row oriented storage:
Nissan Altima 2.5S, 3.3, 40 ft., 175
Mini Cooper Clubman, 3.8, 37 ft., 118
Smart, 5.1, 30 ft., 71

Credit: Consumer Reports New Car Preview 2011

Overview - Graph Databases



- Implement the “Associative Model of Data”
  - By definition: index free adjacency
  - Not record based and no global index
  - Nodes and edges
- Graphs, Directed Graphs, Multi-graphs, Hyper-graphs
- Compared to RDBMS, graph database:
  - Are slower for applying operations to large datasets
  - Have no join operations
  - Excellent for application of graph algorithms
  - Can be faster for associative data sets
  - Map well to object oriented structures
  - Can evolve to changing data types (there is no rigid schema)
- Useful for semantic web implementations (RDF)

Credit: http://www.w3schools.com/web/web_semantic.asp

Example Use- Graph Databases for Semantic Web

- The semantic web gives meaning to data by defining relationships and properties
  - Entities and relationships are described using RDF
    - Subject -> Predicate -> Object
  - Ontologies are defined using RDFS/OWL
    - Enables inferences
    - Full use of OWL can produce NP-complete computations
- RDF can be "naturally" represented using a labeled, directed, multi-graph
- SPARQL enables RDF querying
- Relational databases can store RDF triples easily but efficient SPARQL->SQL querying is difficult
- Native graph databases may provide enhanced performance

Overview - Document Oriented Database

- Focused on efficient management of semi-structured data
  - Focus on self-describing structures (e.g., YAML, JSON, PDF, MS Office, or XML)
- Documents are like records in a RDBMS
- Each document
  - may use a different schema
  - may populate different fields (there are no ‘empty’ fields)
- Document may be accessed through
  - Unique keys
  - Metadata/tagging
  - Collections of documents / Directory structures
  - Query languages (varying by implementation)

Credit: NoSQL and Document Oriented Databases, Morin

Example Use - Document Oriented Databases for XML

- Native XML vs. XML-enabled
- Rationale: If XML is used for communication, why not store the data natively in XML?
- Definition:
  - Model based on XML document structure, not the data
  - Each "record" is an XML document (analogous to a row in an RDBMS)
  - There are no requirements on the storage technique (e.g., relational vs. non-relational)
- Many provide "collections" of documents that may be arranged hierarchically like a file system
- Querying Languages: Xpath, XQuery (extends Xpath)
- Transformations for output may use XSLT

- Hybrid databases have been developed that support querying with SQL and XQuery

Credit: XML Database Products, Bourret; XML and Databases, Bourret