DB Group @ unimo

Big Data: Principles an Technologies

Sonia Bergamaschi
dipartimento di Ingegneria "Enzo Ferrari"

Slides partially taken from:
Gautam Shroff's Web Intelligence and Big Data course on Corusera
NoSQL Principles and Systems:
"Tecnologia delle Basi di Dati" course
for Computer Engineering Master students

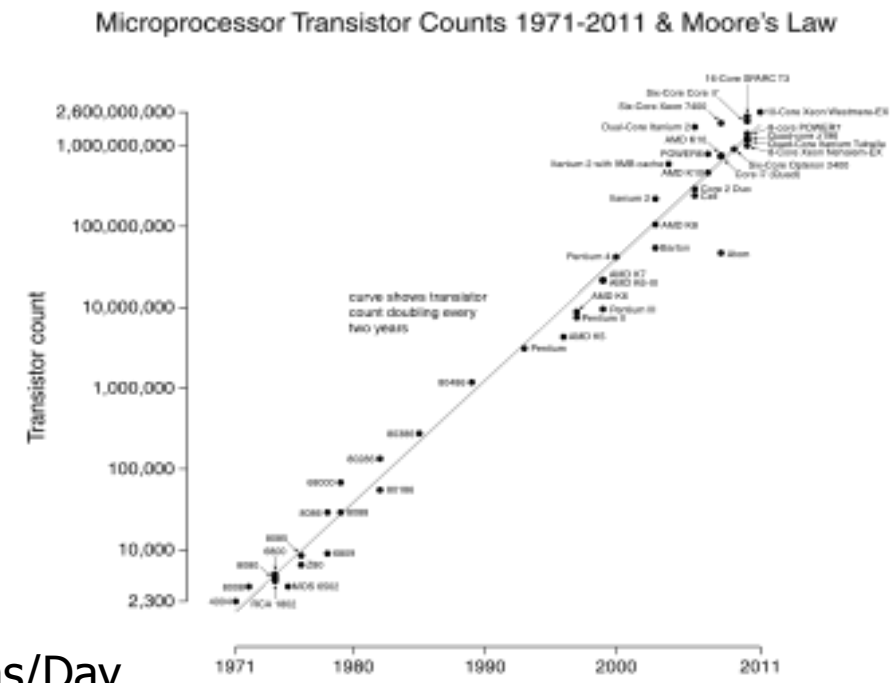
Big Data: Definitions

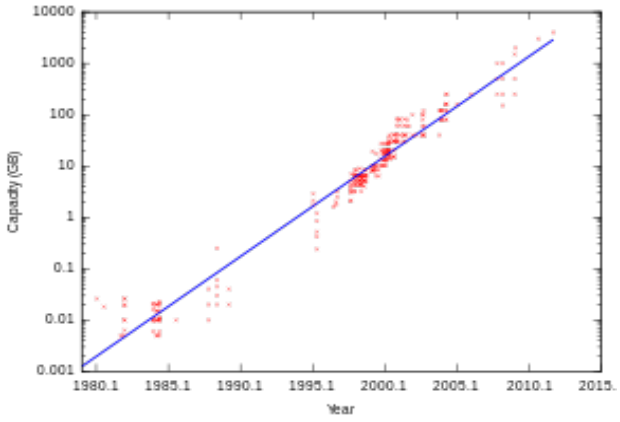
- From Wikipedia: **big data** is a collection of data sets so large and complex that it becomes **difficult to process using on-hand database management tools**. The challenges include capture, curation, storage, search, sharing, analysis, and visualization. The trend to larger data sets is due to the **additional information derivable from analysis of a single large set** of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found.

- From a more technological point of view, **Big Data** refers to the tools, processes and procedures allowing an organization to create, manipulate, and manage very large data sets and storage facilities.

Big Data?

- Lots and lots of web pages
- A billion Facebook users
- Hundreds of million tweets per day
- Billions of Google queries per day
- Millions of servers, petabytes (10^{15} byte) of data

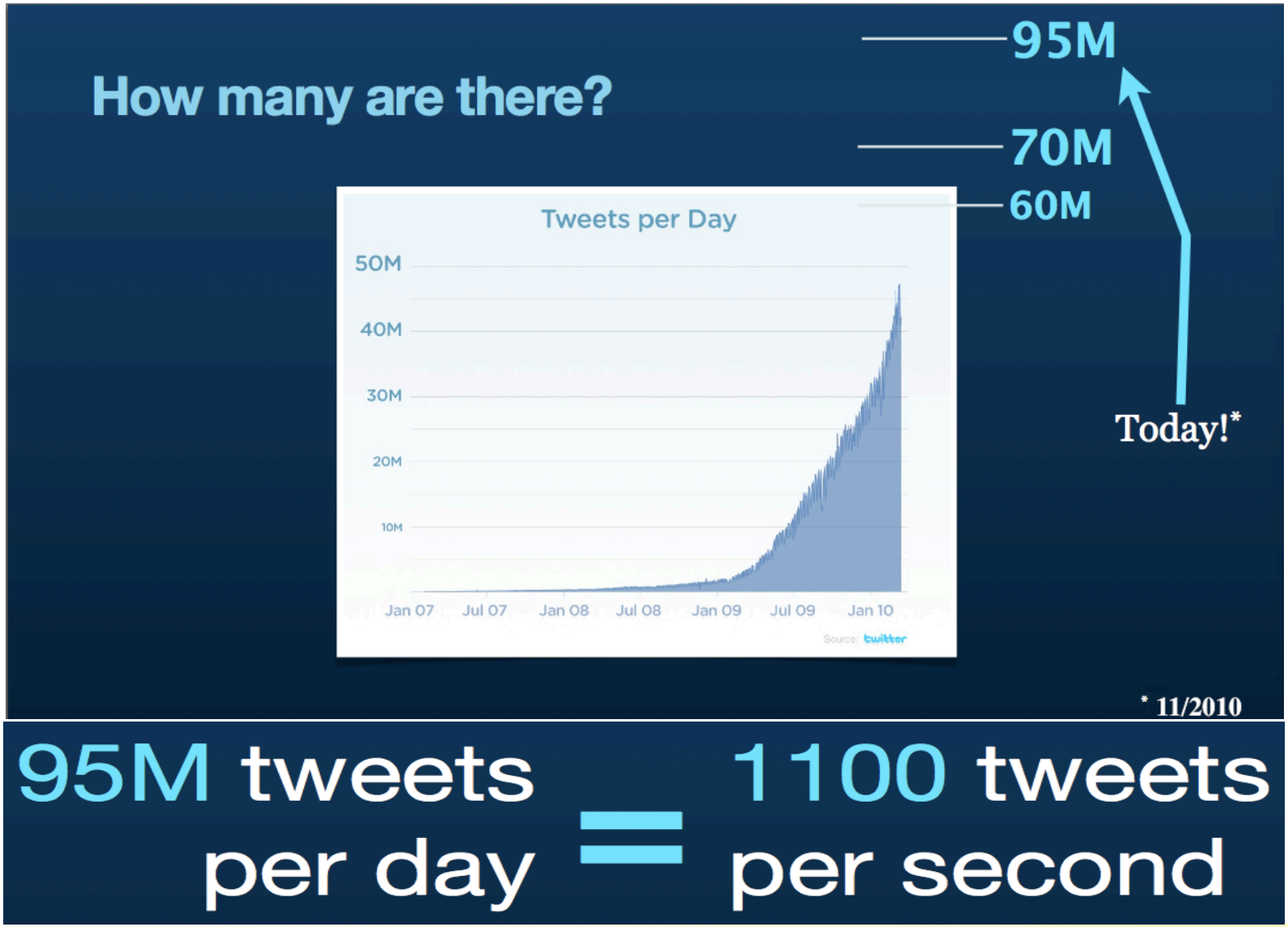




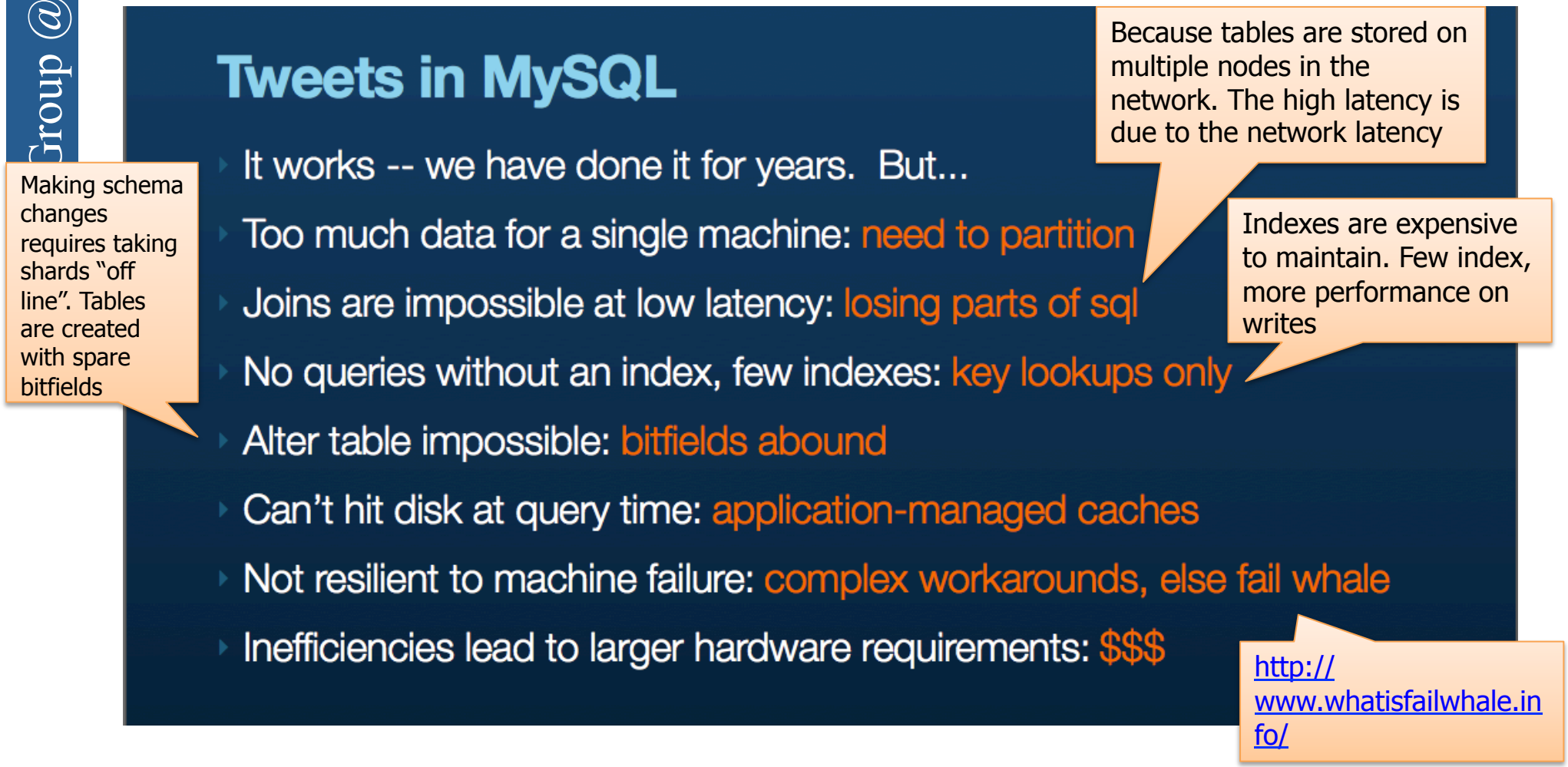
- In contrast, typical large enterprise:
  - 5000-50000 servers
  - Terabytes of data, millions of Transactions/Day

- 2 Laws:
  - Moore's Law: the number of transistors on integrated circuits doubles approximately every two years
  - Kryder's Law: the density of information doubles roughly every 13 months

Example: Web 2.0 (Twitter)



Real-life example: Twitter



(Kevin Weil "Emerging Trends in Data Storage and Processing")[9]

Web Intelligence and Big Data

- So, we have **lots of application oriented data**
- **BUT WHAT can we DO with all this data?**

The quest for knowledge used to begin with grand theories.

Now it begins with massive amounts of data. **Welcome to the Petabyte Age**!

From Business Analysis to Business Analytics

'Any fool can know … the point is to understand.'

-Albert Einstein

and … the goal of understanding is to **predict**

- AI techniques at web-scale for '**predictive intelligence**'
  - Online advertising – predicting intent and interest
  - Gauging consumer sentiment and predicting behavior
  - Intelligent question answering such as in Watson
  - Categorizing and recognizing places, faces, people, …
  - Personalized genomic medicine of the future
  - Building more intelligent public services
  - Securing ourselves better…

Web Intelligence and Big Data

- Traditional 'business intelligence' using databases:

- 'Predict the future using AI and Big Data'

Search

Machine Learning

Information Extraction

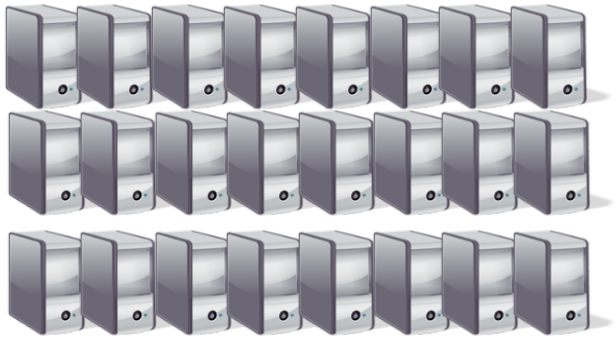
Reasoning

Data Mining

NoSQL

Big Data Technology

Load



parallel programming using map-reduce

Look

Finding 'stuff':

on the web, on one's computer,
in the room, hidden in data
… from one's memories

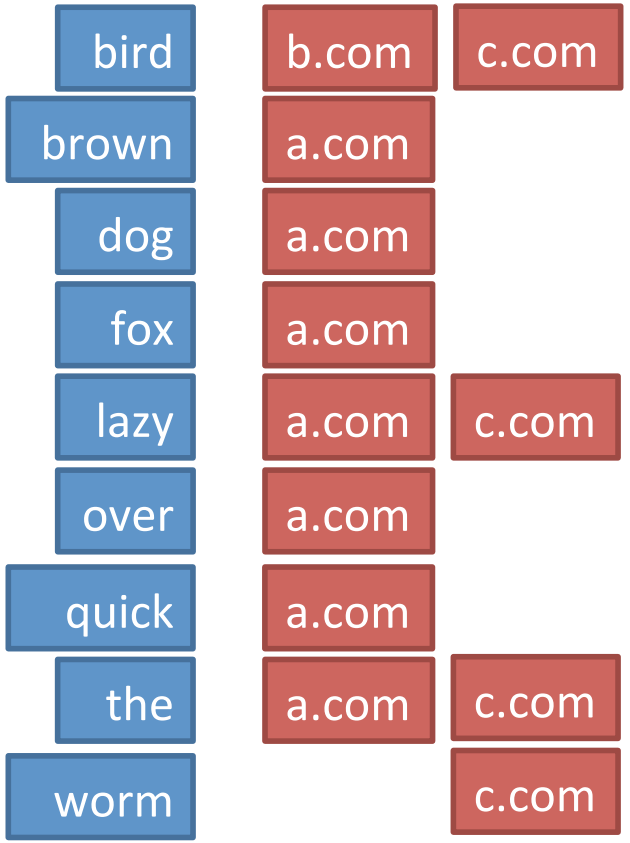
Inverted Index

a.com "the quick brown fox jumps over the lazy dog"

b.com "a bird in hand is worth two in a bush"

c.com "the lazy bird misses the worm"

bird



Now that we know what an Index is…

- How many *indexed** web pages are there?

2-5 billion

30-40 billion

200-300 billion

trillions

* *Indexed* web-pages are those that are returned by search results, unlike, for example, the pages that required a 'login'. The latter are often referred to as the 'deep web'

Now that we know what an Index is…

- How many *indexed** web pages are there?

2-5 billion

✔ 30-40 billion

200-300 billion

trillions

- Search for a common word, such as 'a' or 'the' on Google and see how many results are returned

* *Indexed* web-pages are those that are returned by search results, unlike, for example, the pages that required a 'login'. The latter are often referred to as the 'deep web' (estimated 4.500TB in 2001)

How to Arrange the Results of Search?

- What if the result is very large?
  - E.g., search for 'a' in Google
  - Also, how to assemble results of a multiple keywords query
  - Search for 'Clinton plays India cards':

    "Clinton to visit India but Islamabad was not on the cards…"

    OR "Clinton Cards acquired, will save hundreds of jobs in India…"

- Similarity (from search index) vs importance
  - name the first word that come to mind…

    Starting with 'A'? Starting with 'G'?

    Are some words more important that others?
  - Top 10 documents matching 'Clinton plays India cards'

    Importance = PageRank + … but is there anything deeper?

Searching Structured Data

consider a *LYRICS* database:

Artist (A)

ArtistID	Name
a1	D12
a2	Eminem
a3	Code Red
a4	jojo

Artist-Album (AB)

AB-ID	ArtistID	AlbumID
ab1	a1	b1
ab2	a2	b1
ab3	a4	b3

Album (B)

AlbumID	Title
b1	D12 World
b2	Off the Wall
b3	jojo

Album-Song (BS)

BS-ID	AlbumID	SongID
bs1	b1	s1
bs2	b2	s2
bs3	b3	s3

Song (S)

SongID	Title	Lyrics
s1	How Come	How come we don't … by... more..
s2	The Kids	And everyone should get along ...
s3	Leave(Get out)	I've been waiting all day…

SQL to get albums with "*World*" in the title:

```
Select * from Album B
Where Contains (B.title, 'World' 1) > 0
Order by score(1) desc
```

Searching Structured data (2)

- How many SQL queries will it take to retrieve the names of each artist and the lyrics of every song in an album that has '*World*' in its title?

Artist (A)

ArtistID	Name
a1	D12
a2	Eminem
a3	Code Red
a4	jojo

Artist-Album (AB)

AB-ID	ArtistID	AlbumID
ab1	a1	b1
ab2	a2	b1
ab3	a4	b3

Album (B)

AlbumID	Title
b1	D12 World
b2	Off the Wall
b3	jojo

Album-Song (BS)

BS-ID	AlbumID	SongID
bs1	b1	s1
bs2	b2	s2
bs3	b3	s3

Song (S)

SongID	Title	Lyrics
s1	How Come	How come we don't … by... more..
s2	The Kids	And everyone should get along ...
s3	Leave(Get out)	I've been waiting all day…

Query 1: *'World' from Album* *
Query 2: *"lyrics how come by D12"*
Query 3: *"album by D12 and Eminem"*

"Effective keyword search in relational databases", Liu et. SIGMOD06

Searching Structured Data (3)

- Compare writing SQLs with issuing a 'search' query "*off the world*"
  - Schema needs to be understood
  - Partial matches are missed: e.g., "*World*", "*off the wall*"
  - Many queries, or a complex join are needed
- But there is more:
  - Suppose there were multiple databases, each with a different schema, and with partial or duplicate data?
  - Most important – some unstructured data in documents, other structured in databases: how to search both *together*
- Searching structured data <u>well</u> remains a research problem

"Keyword search over relational databases: a metadata approach". Bergamaschi et. SIGMOD11

'Looking' vs Searching

- Seeing: recognizing objects and activities
- Browsing a bookshelf, flipping pages of a book
- Looking at a data: time-series, histogram, charts

- "*visualizing a scene*"
- "*getting a feel for a document, collection or data*"

- ❑ clustering and classification
- ❑ topic discovery, summarization
- ❑ correlation and `interestingness'

Load

Big Data Technology

Why Big-Data Technology?

- Challenges of Traditional Data Warehouse Technology:
  - (*or why did Google, Yahoo! and Facebook need to invent a new stack?*)
  1. Fault-tolerance at scale
  2. Variety of data types
  3. Manage data volumes without archiving
  4. Parallelism was an add-on
  - ➢ *Could not scale*
  - ➢ *Not suited for compute-intensive deep analytics*
  - ➢ *Price-performance challenge*
- Main innovations: **Map-Reduce on Distributed File-Systems and No SQL Databases / DBMS**
- Main message: *different approaches to data-processing*
- Caveats:
  - Many databases innovations remain unique to the traditional stack
    - *Variety of indexing, complex query optimization, storage optimization*
  - All of these are being re-discovered and re-invented for big-data

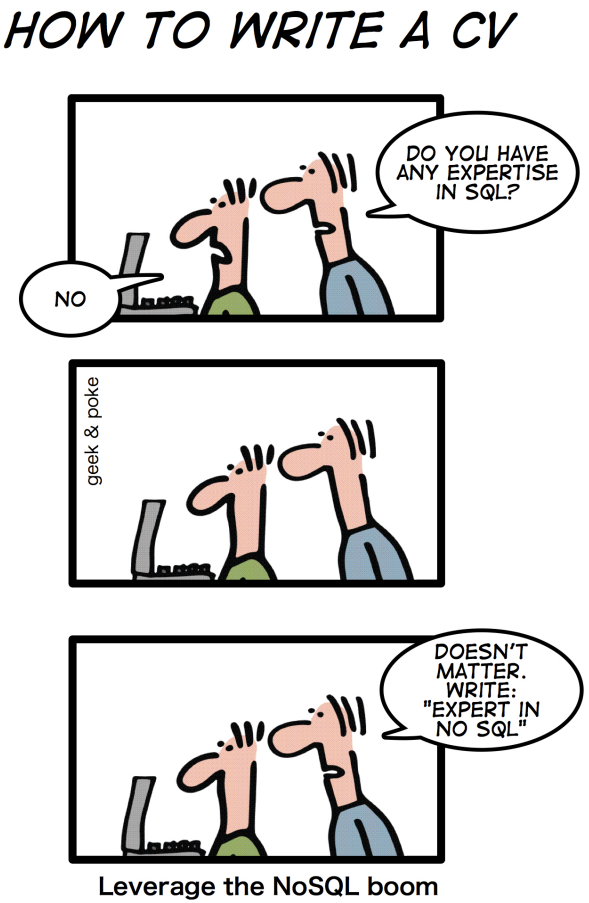
An emerging "movement" around non-relational software for Big Data

- NOSQL stands for "Not Only SQL" (but is not entirely agreed upon), where SQL doesn't really mean the ery language, but instead it denotes the traditional ational DBMS.

Who does use Memcached?
Wikipedia, LiveJournal, Flickr, Twitter, Youtube, Digg, Facebook, eBay, …

ogle **Bigtable**, **Memcached**, and Amazon's **namo** are the "proof of concept" that inspired any of the new data stores and management:

  - Memcached demonstrated that in-memory indexes can be highly scalable, distributing and replicating objects over multiple nodes
  - Dynamo pioneered the idea of *eventual consistency* as a way to achieve higher availability and scalability [5]
  - BigTable demonstrated that persistent record storage could be scaled to thousands of nodes [4]



What is new with NOSQL

New systems with:

- High scalability for **simple operations** (**SO**) on multiple nodes
  - By "simple operations", we refer to key lookups, reads and writes of one record or a small number of records. With the advent of the Web 2.0 sites where millions of users may both read and write data, scalability for simple database operations has become more important
  - This is in contrast to complex queries or joins, read-mostly access, or other application loads
  - SO are highly parallelizable
- **Data partitioning** (**sharding**) and **replication** on multiple nodes
  - Relaxed consistency therefore higher performance and availability
- **Flexibility** on the data structure

But with some sacrifice:

- Interface far more easy then SQL.
  - No declarative query language -> more low level programming
- Transaction management less rigorous
  - Relaxed consistency
- Queries that span multiple shards are very inefficient or impossible

Relational DBMS

The relational model of data was proposed in 1970 by Ted Codd [1] as the desired solution for the DBMS problems of the day, namely business data processing, having the following requirements:

- Persistency, sharing, reliability
- Data with a simple structure, with symbolic/numeric type
- Short-lived concurrent transaction (OLTP)
- Complex query

The dominant commercial vendors (Oracle, IBM, Microsoft) as well as the major open source systems (MySQL, PostgreSQL) all look about the same, and we can term these systems **general-purpose DBMS**. They share the following features:

- Disk-oriented storage
- Tables stored row-by-row on disk (hence, a row store)
- B-trees as the indexing mechanism
- Dynamic locking as the concurrency control mechanism
- A write-ahead log (WAL) for crash recovery
- SQL as the access language
- A "row-oriented" query optimizer and executor

(Stonebraker & Cattel, CACM 2011)[2]

"One-size-fits-all"

According to Stonebraker, the relational systems are proposed as an universal solution.

But Swiss army knives are not the best for everything!!

In fact, in the relational world, this is not true for quite long time:

- A bounce of application families, with very different requirements

Candidates for a Separate Engine

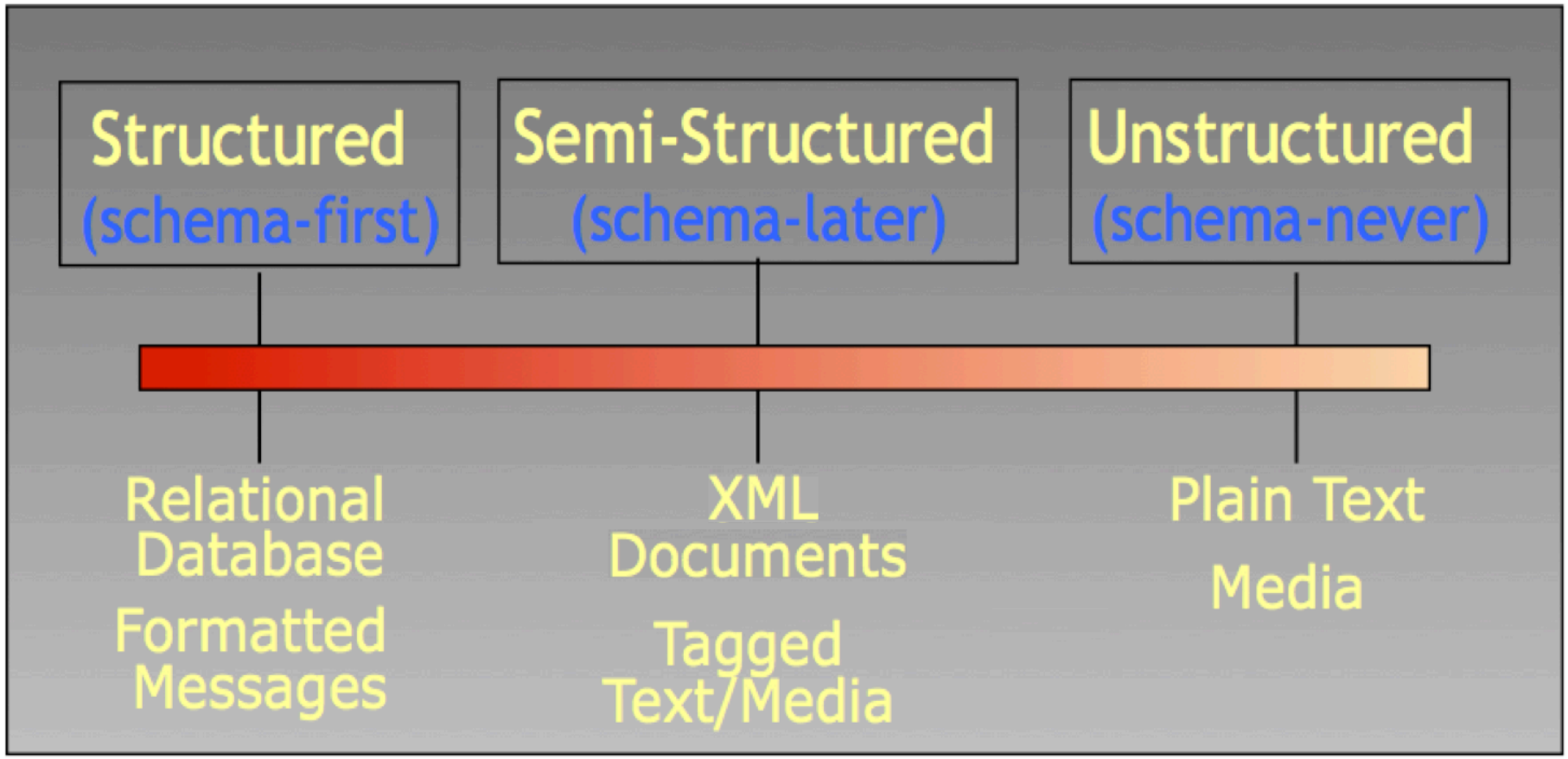
- OLTP
- Warehouses
- Stream processing
- Sensor networks (TinyDB, etc.)
- Text retrieval (Google, etc.)
- Scientific data bases (lineage, arrays, etc.)
- XML (argued by some)

(Stonebraker & Çetintemel, ICDE 2005)[3]

Data Structure Flexibility: Why

- Data change over time:
  - As the business model evolves, concepts and data model often struggle to evolve and keep pace with changes
  - The result is often a data structure that is filled with patches and adapted data
  - Some applications require each object in a collection to have different attributes
- Relational data model may not be optimal for not-relational data:
  - If you only require a lookup of objects based on a single key, then a key-value store is more adequate (and probably easier to understand than a relational DBMS)

The Structure Spectrum



Scaling Up

PC

Server

Cluster

Data center

Network of data centers

Google data center location (inferred):

The Cloud

- The power plant analogy:

- Cloud computing is a **model** for enabling convenient, **on-demand** network access to a **shared pool** of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction
- Characteristic:
  - On-demand self service
  - Measured service (pay as you go)
  - Rapid elasticity (resources can be rapidly and elastically provisioned to quickly scale up and rapidly released to quickly scale down)
  - Resource pooling

The SPI model

- The cloud provides three types of commonly distinguished service:
  - **Software as a service** (SaaS)
    - The cloud provides an entire application (word processor, calendar, **data storage**, etc)
    - Customer pays cloud provider
    - Example: Google Apps, Saleforce.com
  - **Platform as a service** (PaaS)
    - The cloud provides just the middleware/infrastucture
    - Customer pays SaaS provider for the service, SaaS provider pays the cloud for the infrastructure
    - Example: Windows Azure, Google App Engine
  - **Infrastructure as a service** (IaaS)
    - Virtual machine, blade server, hard disk
    - Customer pays SaaS provider for the service, SaaS provider pays the cloud for the resources
    - Example: Amazon Web Services (AWS), Rackspace Cloud, GoGrid
- Who can became a customer of the cloud? Three types of cloud
  - **Public cloud**: commercial service, open to (almost anyone)
    - Amazon Web Services , Windows Azure, Google App Engine
  - **Community cloud**: shared by several similar organizations
    - Google's "Gov Cloud"
  - **Private cloud**: shared within a single organization
    - Internal datacenter of a large company

10 Obstacles and Opportunities for Cloud Computing [14]

1. Availability
2. Data lock-in
   - How do I move my data from one cloud to another?
3. Data confidentiality
   - How do I make sure that the cloud doesn't leak my confidential data?
4. Data transfer bottlenecks
   - How do I copy large amounts of data from/to the cloud?
5. Performance unpredictability
   - Remember, you are sharing resources with other customers
6. Scalable storage
   - The cloud model (short-term usage, infinite capacity on demand) does not fit well with persistent storage
7. Bugs in large distributed systems
8. Scaling quickly
9. Reputation fate sharing
   - One customer's bad behavior can affect the reputation of others using the same cloud
10. Software licensing

Some issues with replicas and partitions

Brewer's CAP Theorem [6][13]:

An asynchronous system can have only two out of three of the following properties:

- **Consistency**: in a system containing replicas, these must be all up-to-date before they can be accessed (access are under concurrency control with locks or multi-versions)
- **Availability**: for a distributed system to be continuously available, every request received by a node in the system must result in a response. An available system must have replicas because failures on the server side may occur
- **Partition-tolerance**: if replicas are on different nodes, the system must continue to operate even in the presence of a network partition (i.e. if I have a database running on 80 nodes across 2 racks and the connection between racks is lost, my database is now partitioned. If the system is partition-tolerant, then the database will still be able to perform read and write operations while partitioned)

CAP Theorem

- "Demonstration": if we have partition-tolerance and in each partition we can read without interruption (availability), then we can not have consistency, since updates can not be propagated
- Observations by Stonebraker and Cattel [2]:
  - high availability is very hard to obtain given all the kinds of failure that share-nothing DBMS can face:
    - Application failures (where the application corrupts the database)
    - DBMS failures where the bug can be recreated (deterministic bugs or so called Bohr bugs)
    - DBMS failures where the bug cannot be recreated (non-deterministic bugs, or so called Heisenbugs)
    - Hardware failures of all kinds
    - Lost network packets
    - Denial of service attacks
    - Network partitions
  - Consistency in some systems is fundamental

Consistency

3 types of consistency:

- **Strong Consistency:** updates are correctly visible from all the readings
- **Weak Consistency:** it is not granted that updates are visible from successive readings
- **Eventual Consistency:** updates are guaranteed to be propagated to all nodes eventually, therefore data fetched are not guaranteed to be up-to-date

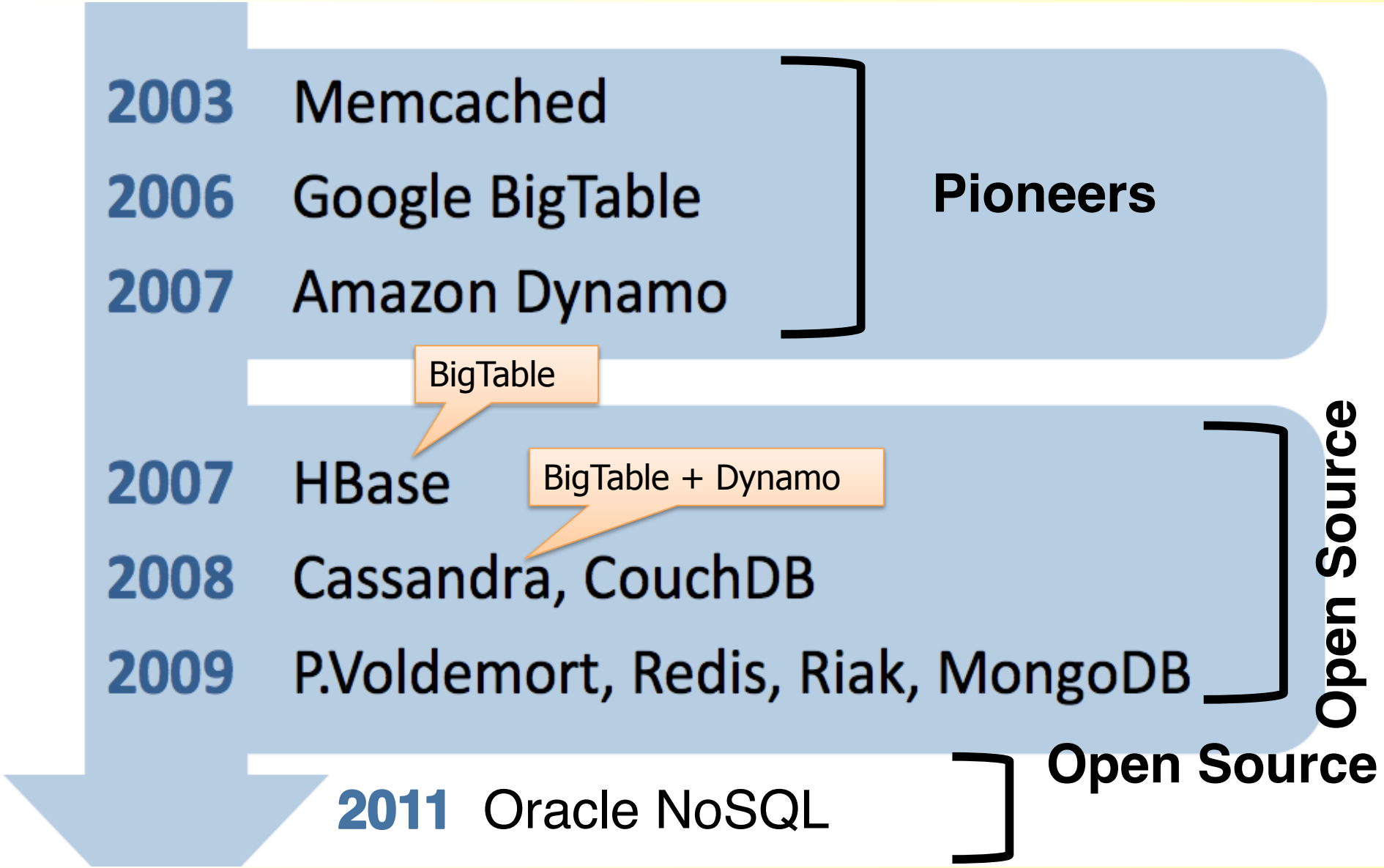
In a system containing replicas, updates and data fetches can be implemented using the **quorum** mechanism (if you are interested, see [10] [11]). Suppose:

- **N** replicas
- **W** number of replicas that have to be updated (and confirmed) in order to consider the writing process complete
- **R** number of (equal) copies that have to be looked up in order to reply to a data fetch request
- If the systems is not able to update **W** nodes or the system is not able to read **R** nodes, the operation is considered failed (the availability property is lost)

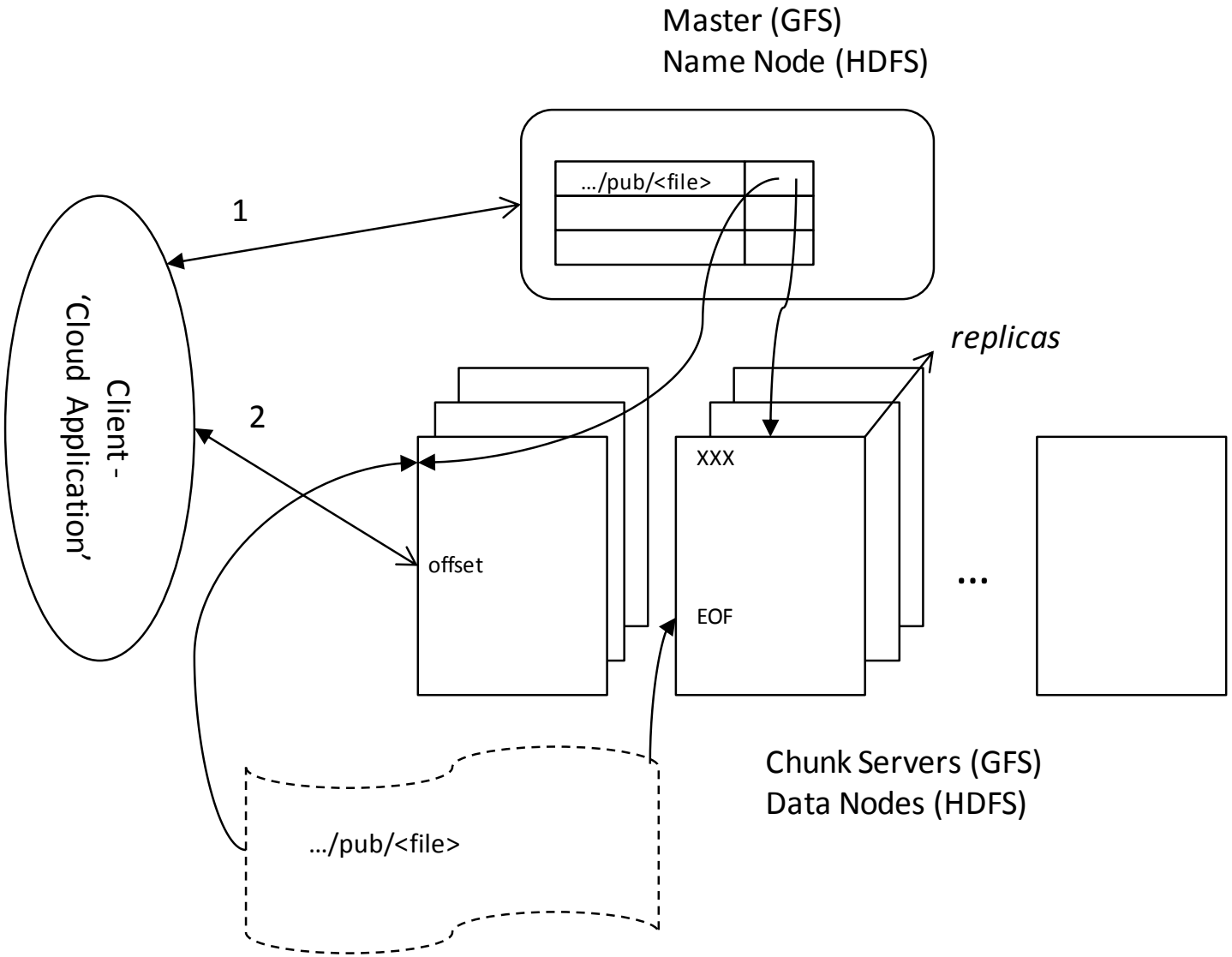
Strong Consistency

- The Strong Consistency requires:
  - **W + R > N**
  - In this way it is assured that readings contains at least one updated data
  - The set of fetched elements and the set of updated replicas have elements in common. If the fetched data is coherent, we can assume we have read the current value
- Weak Consistency:
  - **W + R ≤ N**
  - The set of fetched elements may not have elements in common with the set of updated replicas
  - Practically it is used only with R = 1 in order to achieve high scalability and availability even if partitioning is present

Historical View



Distributed File Systems (GFS, HDFS)



Map-Reduce

- **MapReduce** is a **programming model** and an **associated infrastructure** for **large scale computing** on large data sets using **commodity hardware**
- Developed by Google and first presented in:
  - Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Operating Systems Design \& Implementation - Volume 6* (OSDI'04), Vol. 6. USENIX Association, Berkeley, CA, USA, 10-10.

- Who use Map-Reduce?

  (Actually Hadoop http://wiki.apache.org/hadoop/FrontPage

  implements a Map-Reduce open source version)
  - Amazon CloudSearch, Accela Communication, Adobe, AOL, adyard, Able Grape, Adknowledge, Aguja, Alibaba, AOL, ARA.COM.TR, Atbrox, BabaCar, Basenfasten, Benipal Technologies, BrainPad, Brilig, Brockmann Consult GmbH, Caree.rs, CDU now!, Charleston, Cloudspace, Contestweb, Cooliris, Cornell University Web Lab, CRS4, crowdmedia, Datagraph, Detikcom, devdaily.com, DropFire, eBay, Enet, Explore.To Yellow Pages, Facebook…
  - More at http://wiki.apache.org/hadoop/PoweredBy

Google MapReduce Framework (Hadoop)

- **MapReduce** is a **programming model** and an **associated infrastructure** for **large scale computing** on large data sets using **commodity hardware**
- Challenges:
  - How to distribute computation
  - How to deal with machine fails
- Idea:
  - Bring computation close to the data
  - Store files multiple times for reliability
- Basically users specify a **map** function that processes a key/value pair to generate a set of intermediate key/value pairs, and a **reduce** function that merges all intermediate values associated with the same intermediate key
- The run-time system takes care of the details of **partitioning** the input data, **scheduling** the program's execution across a set of machines, **handling machine failures**, and **managing** the required inter-machine communication. [18]

MapReduce Example: Word Counting

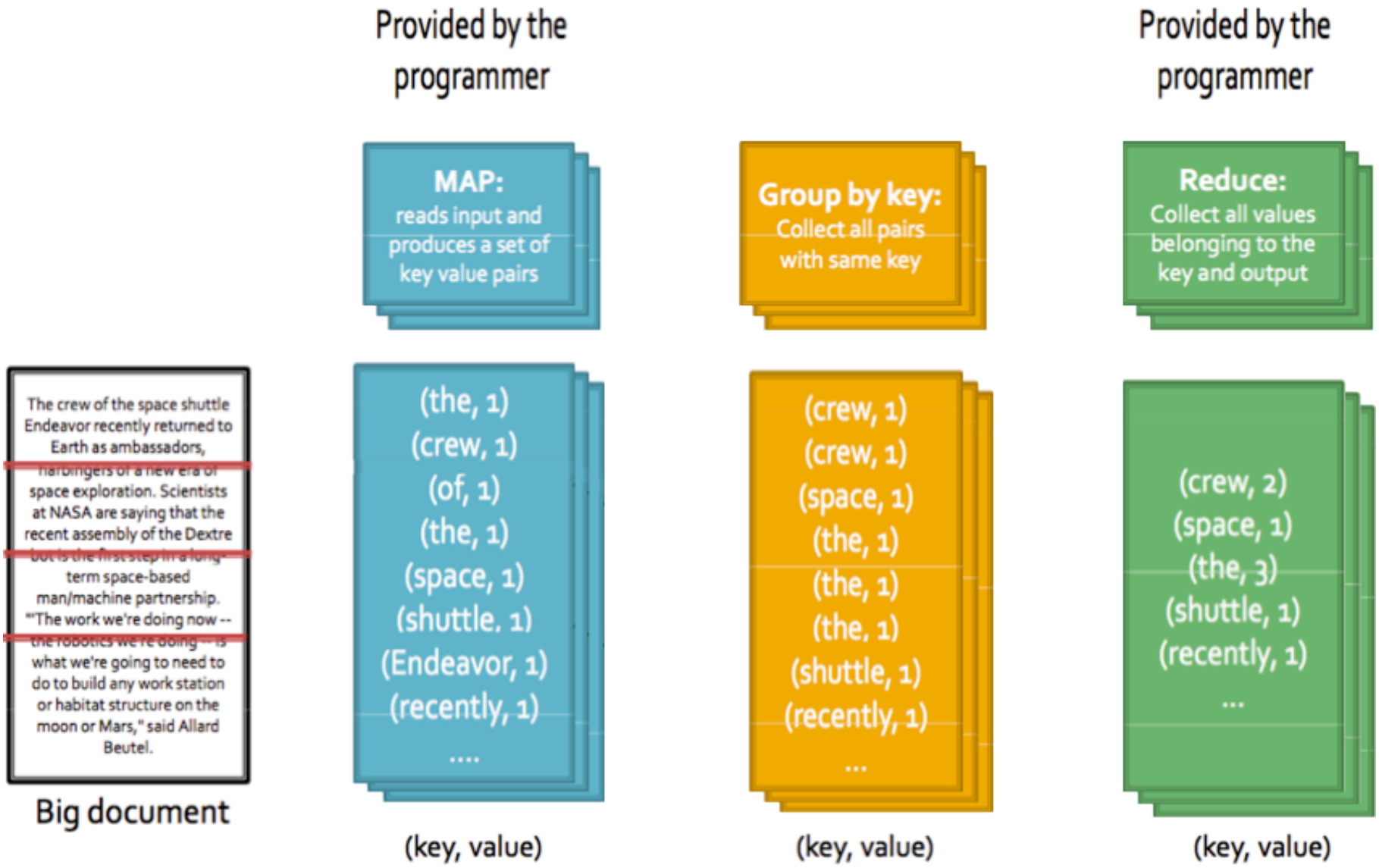
- Consider the problem of counting the number of occurrences of each word in a large collection of documents:

```
map(String key, String value):
   // key: document name
   // value: document contents
   for each word w in value: EmitIntermediate(w, "1");

reduce(String key, Iterator values):
   // key: a word
   // values: a list of counts
   int result = 0;
   for each v in values: result += ParseInt(v);
   Emit(AsString(result));
```

- The map function emits each word plus an associated count of occurrences (just '1' in this simple example). The reduce function sums together all counts emitted for a particular word.

MapReduce Example: Word Counting - 2



MapReduce Data Flow

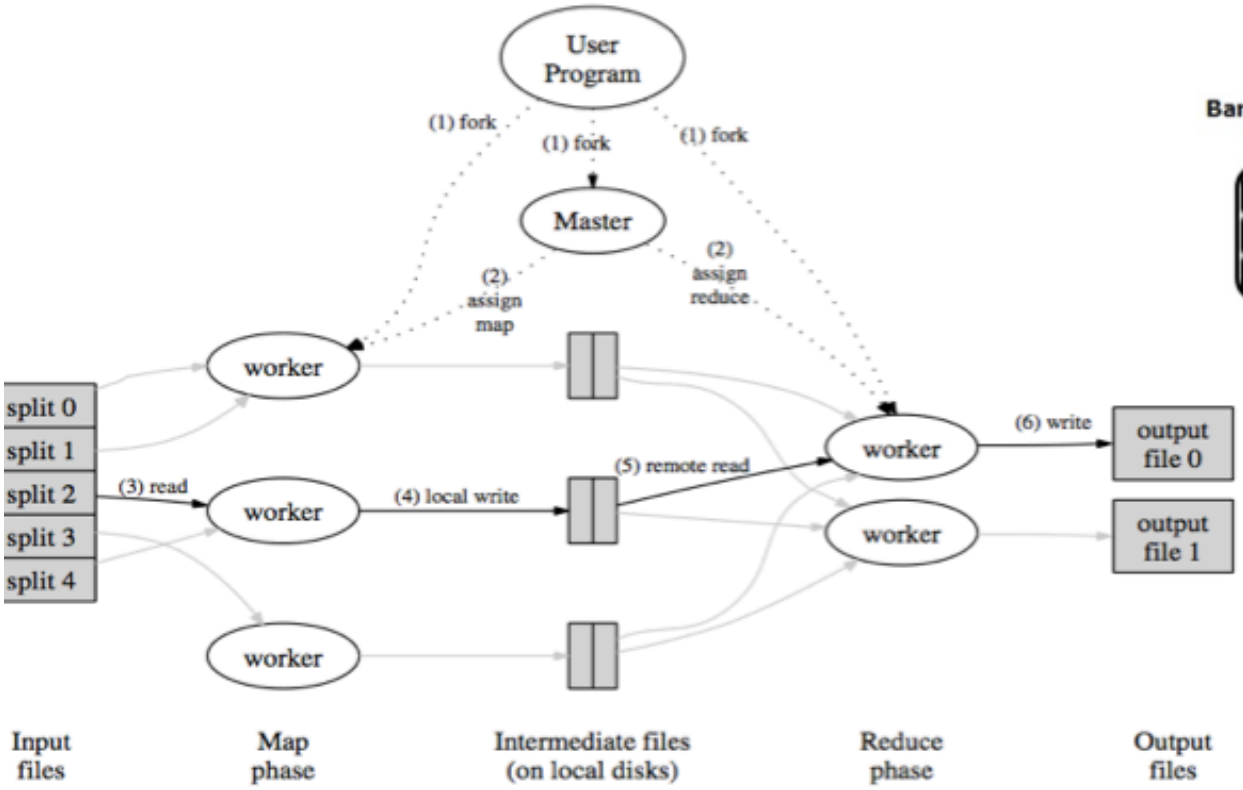
- When the user program calls the MapReduce function, the following sequence of actions occurs:
  1. The MapReduce library in the user program first splits the input files into M pieces. It then starts up many copies of the program on a cluster of machines
  2. One of the copies of the program is special – **the master**. The rest are **workers** that are assigned work by the master. There are **M** map tasks and **R** reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task
  3. A worker who is assigned a map task reads the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-defined Map function. The intermediate key/value pairs produced by the Map function are buffered in memory
  4. Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers
  5. When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. The sorting is needed because typically many different keys map to the same reduce task.

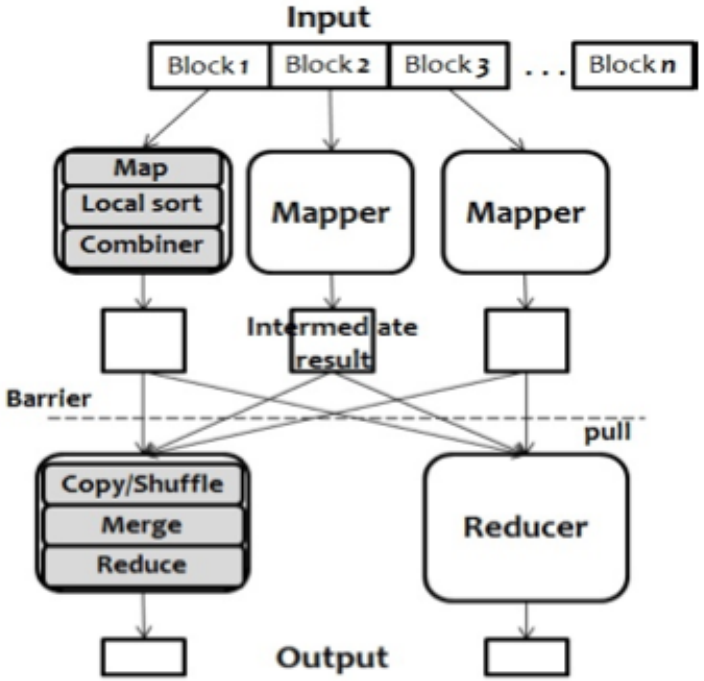
MapReduce Data Flow -2

6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is appended to a final output file for this reduce partition
7. When all map tasks and reduce tasks have been completed, the master wakes up the user program

- After successful completion, the output of the MapReduce execution is available in the **R** output files.
- Typically, users do not need to combine these **R** output files into one file, they often pass these files as input to another MapReduce call, or use them from another distributed application that is able to deal with input that is partitioned into multiple files.

Inside Map-Reduce

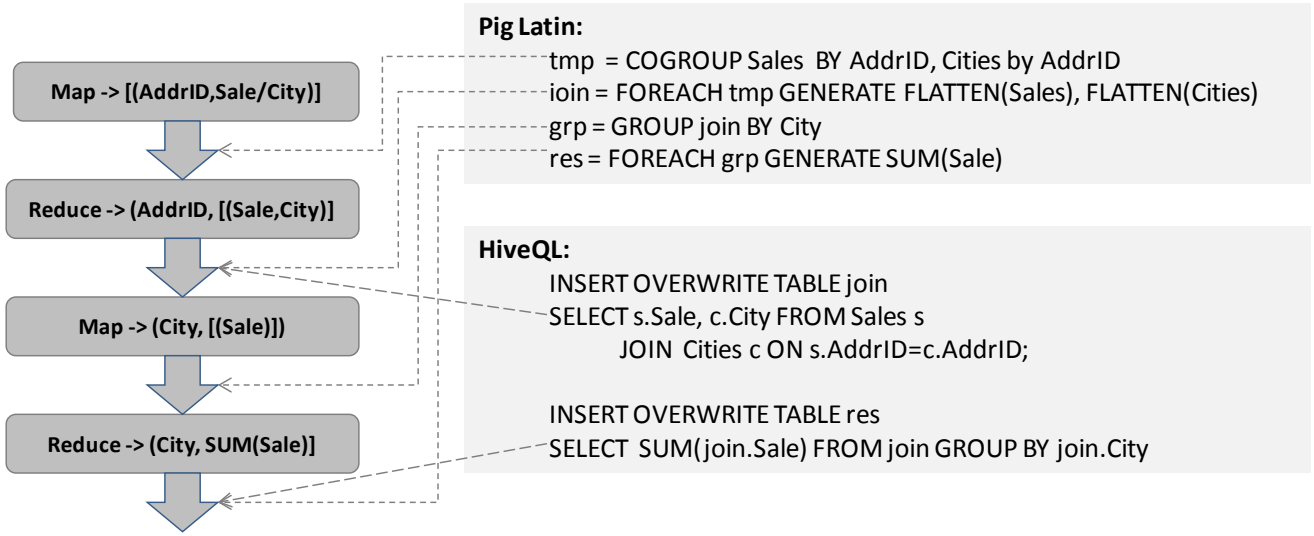




High-level languages and MapReduce

In the MapReduce framework relational schemas and declarative query are missed:

- **Apache HiveQL** (Facebook): provides schemas and SQL-like query language
- **Apache Pig Latin** (Yahoo!): more imperative than Hive but with relational operators
- Both compile to workflow of MapReduce (actually Hadoop) jobs
- Both are especially used in a data warehousing context



SQL: SELECT SUM(Sale), City from Sales, Cities WHERE Sales.AddrID=Cities.AddrID GROUP BY City

Listen and Learn

Recognize a shopper from a browser…
...gauge opinion and sentiment
...understant what people are saying

Measuring information: what is news?

The New York Times

Europe

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION

AFRICA | AMERICAS | ASIA PACIFIC | EUROPE | MIDDLE EAST

At British Inquiry, Murdoch Apologizes Over Scandal

By ALAN COWELL
Published: April 26, 2012

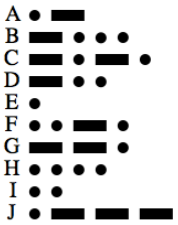
LONDON — After a day of testimony at a British judicial inquiry over his ties, friendships and disputes with British politicians, Rupert Murdoch returned to the witness stand on Thursday, saying he apologized for failing to take measures to avert the hacking scandal that has convulsed his media outpost here.

FACEBOOK
TWITTER
GOOGLE+
E-MAIL
SHARE

Claude Shannon (1948): *information* is related to surprise

a message informing us of an event that has probability *p* conveys

$- \log_2 p$ bits of *information* $- \log .5 = 1$



a, in, the, ..
information
miscellaneous

"It from bit" John Wheeler, 1990

when we pick up a newspaper, we are looking for maximum information, so more `surprising' events make for better news!

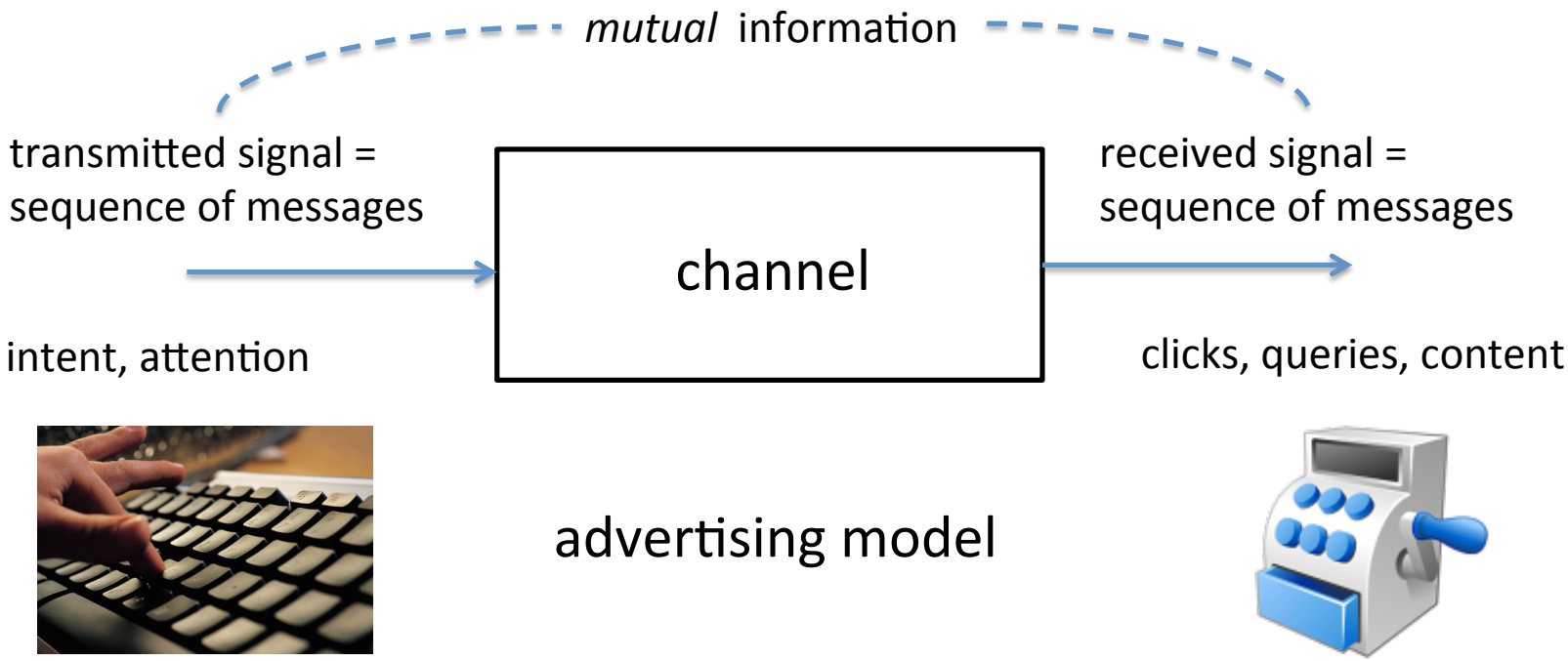
in passing, you glance at some ads, and the paper makes money!

Information and Online Advertising

when to place and ad, and *where* to place an ad?

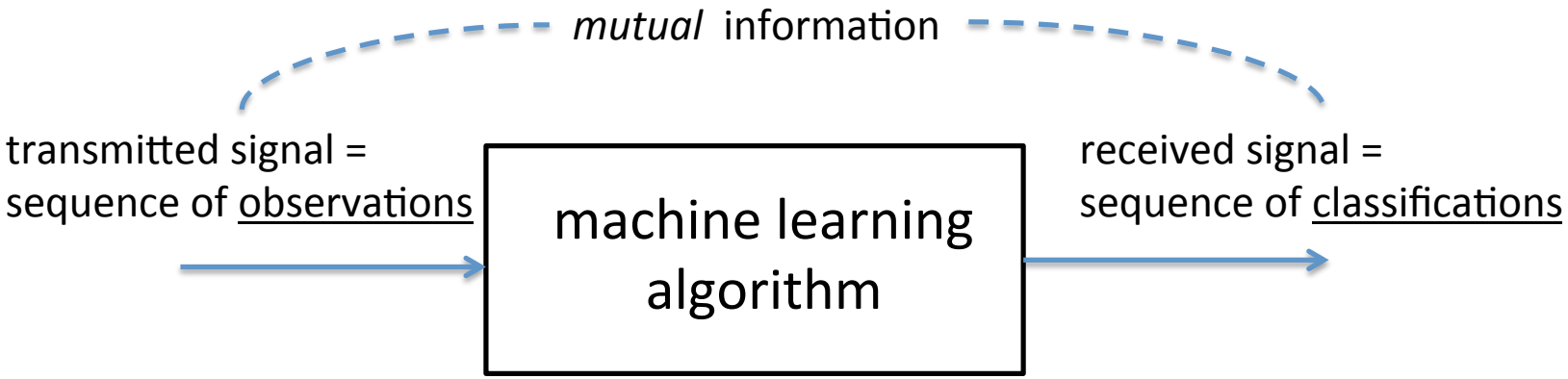
what if the interesting news is on the sports page?

communication along a noisy channel (Shannon):



advertising model

Learning and Information Theory



Shannon defined *capacity* for communications channels:

> *"maximum mutual information between sender and receiver per second"*

what about machine learning?

> "… complexity of Bayesian learning using information theory and the VC dimension", Haussler, Kearns and Schapire, J. Machine Learning, 1994

`right' Bayesian classifier will eventually learn any concept

… how fast? … it depends on the concept itself – 'VC' dimension"

Connect and Predict

Beyond learning, reasoning…
…logic and reasoning under uncertantiy...
...and predict

Reasoning

"book me an American flight to NY ASAP"

"this New Yorker who fought at the battle of Gettysburg was once considered the inventor of baseball"

Alexander Cartwright or Abner Doubleday – *Watson got it right*

"who is the Dhoni of USA?"

– *analogical reasoning* - X is to USA what Cricket is to India (?)

+ *abductive reasoning* – there *is no* US baseball team … so ?

find *best possible answer^*

+ *reasoning under uncertainty* … who is the "most" popular ?

Semantic Web:

- web of linked *data*, *inference* rules and engines, query
  - pre-requisite: extracting *facts* from text, as well as *rules*

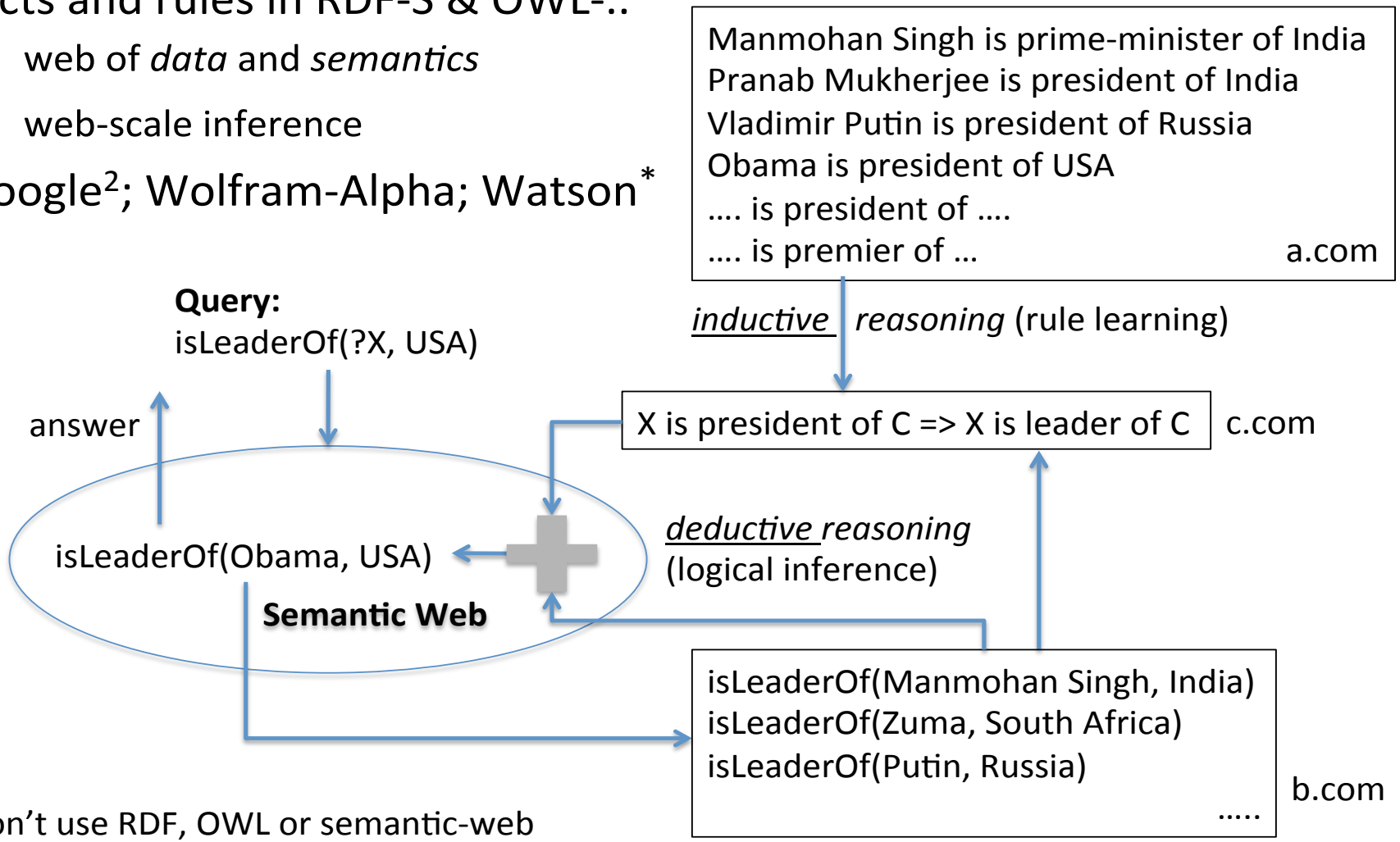
Semantic Web: the Vision

facts and rules in RDF-S & OWL-..

web of *data* and *semantics*

web-scale inference

Google[2]; Wolfram-Alpha; Watson*



*don't use RDF, OWL or semantic-web *technology* though they have similar intent, spirit …

Predict – Decide - Control

robo-soccer

predict where the ball *will* be; decide best path; navigate there

predict how *other* players will move

self-driving cars

predict the path of a pedestrian; decide path to avoid; steer car

predict traffic; decide *all* optimal routes to destination

energy-grid

predict energy demand; decide & control distribution

predict supply by 'green-ness'; adjust prices *optimally*

supply-chain

predict demand for products; decide best production plan; execute it

detect potential risk & evaluate impact; re-plan production; execute it

marketing

predict demand; decide promotion strategy by region; execute it

References

[1] Codd, E. F., "A Relational Model of Data for Large Shared Databanks," CACM, June 1970.

[2] Stonebraker, Cattel, "Ten Rules for Scalable Performance in "Simple Operation" Datastores", CACM 2011.

[3] Michael Stonebraker, Ugur Çetintemel: "One Size Fits All: An Idea Whose Time Has Come and Gone", ICDE 2005: pp. 2-11.

[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber: "Bigtable: A Distributed Storage System for Structured Data", OSDI 2006.

[5] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, "Dynamo: Amazon's Highly Available Key-value Store", SOSP 2007.

[6] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, and partition-tolerant web services", ACM SIGACT News 33, 2, pp 51-59, 2002.

[7] Dan Pritchett, "BASE: An Acid Alternative", ACM Queue 2008.

[8] Rick Cattel, "Scalable SQL and NoSQL Data Stores", ACM SIGMOD Record 2011.

[9] Kevin Weil, "Emerging Trends in Data Storage and Processing", Stanford CS 145, 2010, http://infolab.stanford.edu/~widom/old145/weil.pdf.

[10] Leslie Lamport, "Paxos Made Simple", ACM SIGACT News 32, 4, pp 51-58, 2001.

[11] Leslie Lamport "The Part-Time Parliament" ACM Transaction on Computer Systems 2000.

References

[12] David J. DeWitt, Samuel Madden, Michael Stonebraker, "How to Build a High-Performance Data Warehouse", http://db.csail.mit.edu/madden/high_perf.pdf

[13] Eric A. Brewer, "Towards robust distributed systems". (Invited Talk) Principles of Distributed Computing, Portland, Oregon, 2000.

[14] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei ZahariaA, "View of Cloud Computing", CACM April 2010

[15] Stadford University CS 345a, "MapReduce", 2010. http://www.stanford.edu/class/cs345a/slides/02-mapreduce.pdf

[16] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System", Symposium of Operating Systems Principles, SOSP 2003

[17] University of Pennsylvania CIS 399 "What is the cloud?" http://www.cis.upenn.edu/~mkse212/slides/02-TheCloud.pptx

[18] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Operating Systems Design and Implementation, OSDI 2004

[19] University of Berkley CS 186 "NoSQL and Course Wrap Up" http://dl.dropbox.com/u/8950924/17Sp11-NoSQLWrapUp.pptx

[20] Prof. Paolo Atzeni "NoSQL: Concetti generali" http://www.dia.uniroma3.it/~torlone/bd2/noSQL-1.pdf

References

[21] Christof Strauch, "NoSQL Databases" lecture, 2011
http://www.christof-strauch.de/nosqldbs.pdf

[22] Merkle, Ralph C. "A Digital Signature Based on a Conventional Encryption Function", A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, CRYPTO 87

[23] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, "Dremel: Interactive Analysis of Web-Scale Datasets", VLDB 2010

[24] Marc Shapiro, Nuno Preguiça, Carlos Baquero, Marek Zawirski: "Conflict-Free Replicated Data Type", Stabilization, Safety, and Security of Distributed Systems (SSSD2011)

[25] Neil Conway "Bloom and CALM", 2012
http://basho.com/blog/technical/2012/07/16/Neil-Conway-Basho-Chats-CALM-Bloom/

[26] Sonia Bergamaschi, Matteo Interlandi, "NoSQL Principles and Systems, UNIMORE, 2011.