

CC5212-1

PROCESAMIENTO MASIVO DE DATOS

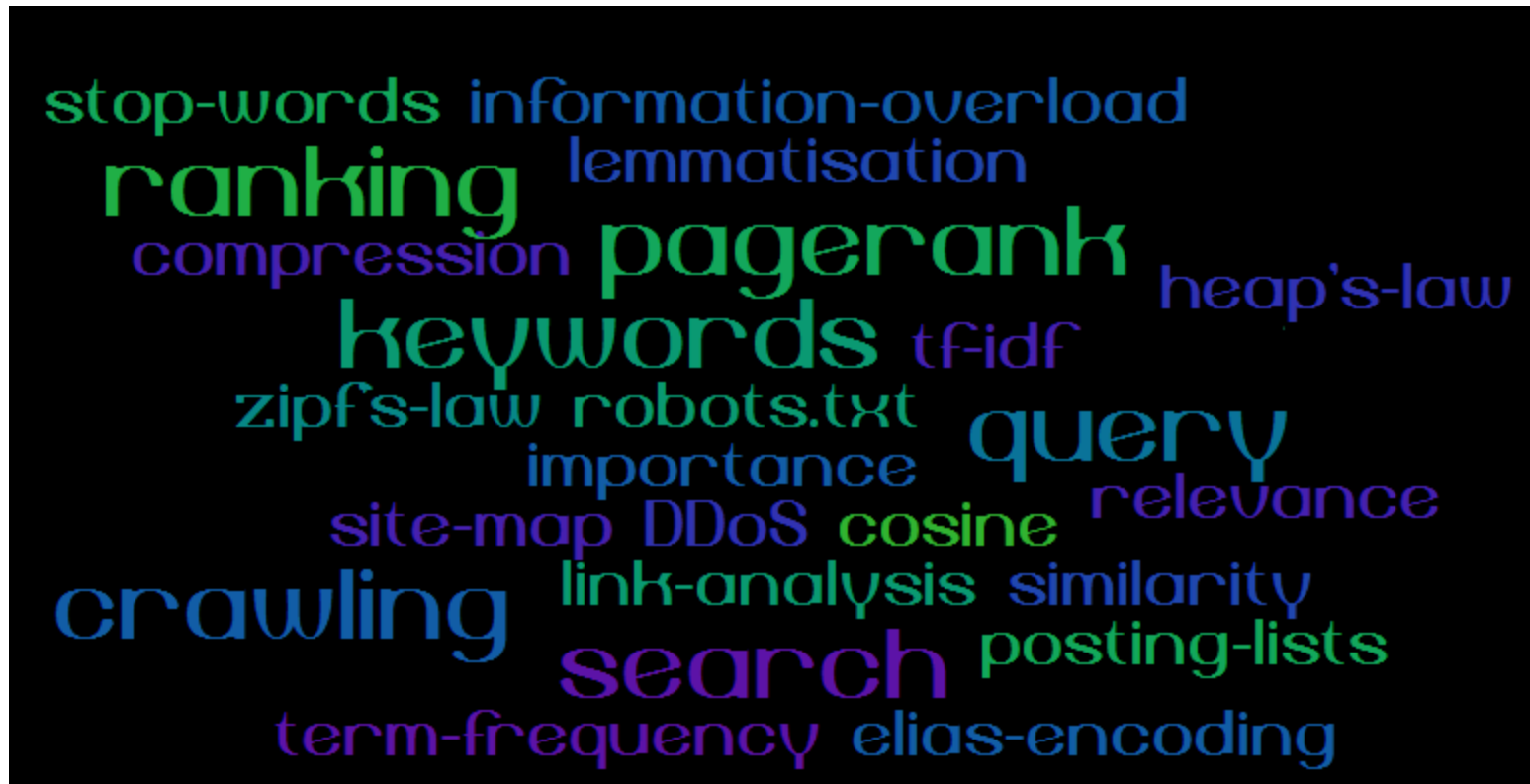
OTOÑO 2016

Lecture 10: NoSQL I

Aidan Hogan

aidhog@gmail.com

Information Retrieval: Storing Unstructured Information



**BIG DATA:
STORING STRUCTURED INFORMATION**

Relational Databases



Relational Databases: One Size Fits All?

“One Size Fits All”: An Idea Whose Time Has Come and Gone



Michael Stonebraker
*Computer Science and Artificial
Intelligence Laboratory, M.I.T., and
StreamBase Systems, Inc.*
stonebraker@csail.mit.edu



Uğur Çetintemel
*Department of Computer Science
Brown University, and
StreamBase Systems, Inc.*
ugur@cs.brown.edu

Abstract

The last 25 years of commercial DBMS development can be summed up in a single phrase: “One size fits all”. This phrase refers to the fact that the traditional DBMS architecture (originally designed and optimized for business data processing) has been used to support many data-centric applications with widely varying characteristics and requirements.

In this paper, we argue that this concept is no longer applicable to the database market, and that the commercial world will fracture into a collection of independent database engines, some of which may be unified by a common front-end parser. We use examples from the stream-processing market and the data-warehouse market to bolster our claims. We also briefly discuss other markets for which the traditional architecture is a poor fit and argue for a critical rethinking of the current factoring of systems services into products.

of multiple code lines causes various practical problems, including:

- *a cost problem*, because maintenance costs increase at least linearly with the number of code lines;
- *a compatibility problem*, because all applications have to run against every code line;
- *a sales problem*, because salespeople get confused about which product to try to sell to a customer; and
- *a marketing problem*, because multiple code lines need to be positioned correctly in the marketplace.

To avoid these problems, all the major DBMS vendors have followed the adage “put all wood behind one arrowhead”. In this paper we argue that this strategy has failed already, and will fail more dramatically off into the future.

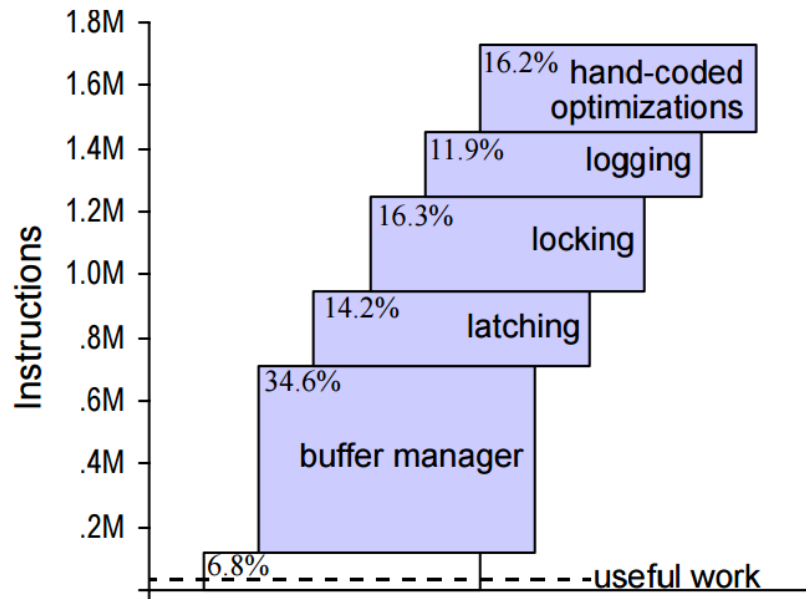
The rest of the paper is structured as follows. In Section 2, we briefly indicate why the single code-line strategy has failed already by citing some of the key characteristics of the data warehouse market. In Section



RDBMS: Performance Overheads

- Structured Query Language (SQL):
 - Declarative Language
 - Lots of Rich Features
 - **Difficult to Optimise!**
- Atomicity, Consistency, Isolation, Durability (ACID):
 - Makes sure your database stays correct
 - Even if there's a lot of traffic!
 - **Transactions incur a lot of overhead**
 - **Multi-phase locks, multi-versioning, write ahead logging**
- **Distribution not straightforward**

Transactional overhead: the cost of ACID



- 640 transactions per second for system with full transactional support (ACID)
- 12,700 transactions per second for system without logs, transactions or lock scheduling

OLTP Through the Looking Glass, and What We Found There

Stavros Harizopoulos
HP Labs
Palo Alto, CA
stavros@hp.com

Daniel J. Abadi
Yale University
New Haven, CT
dna@cs.yale.edu

Samuel Madden Michael Stonebraker
Massachusetts Institute of Technology
Cambridge, MA
{madden, stonebraker}@csail.mit.edu

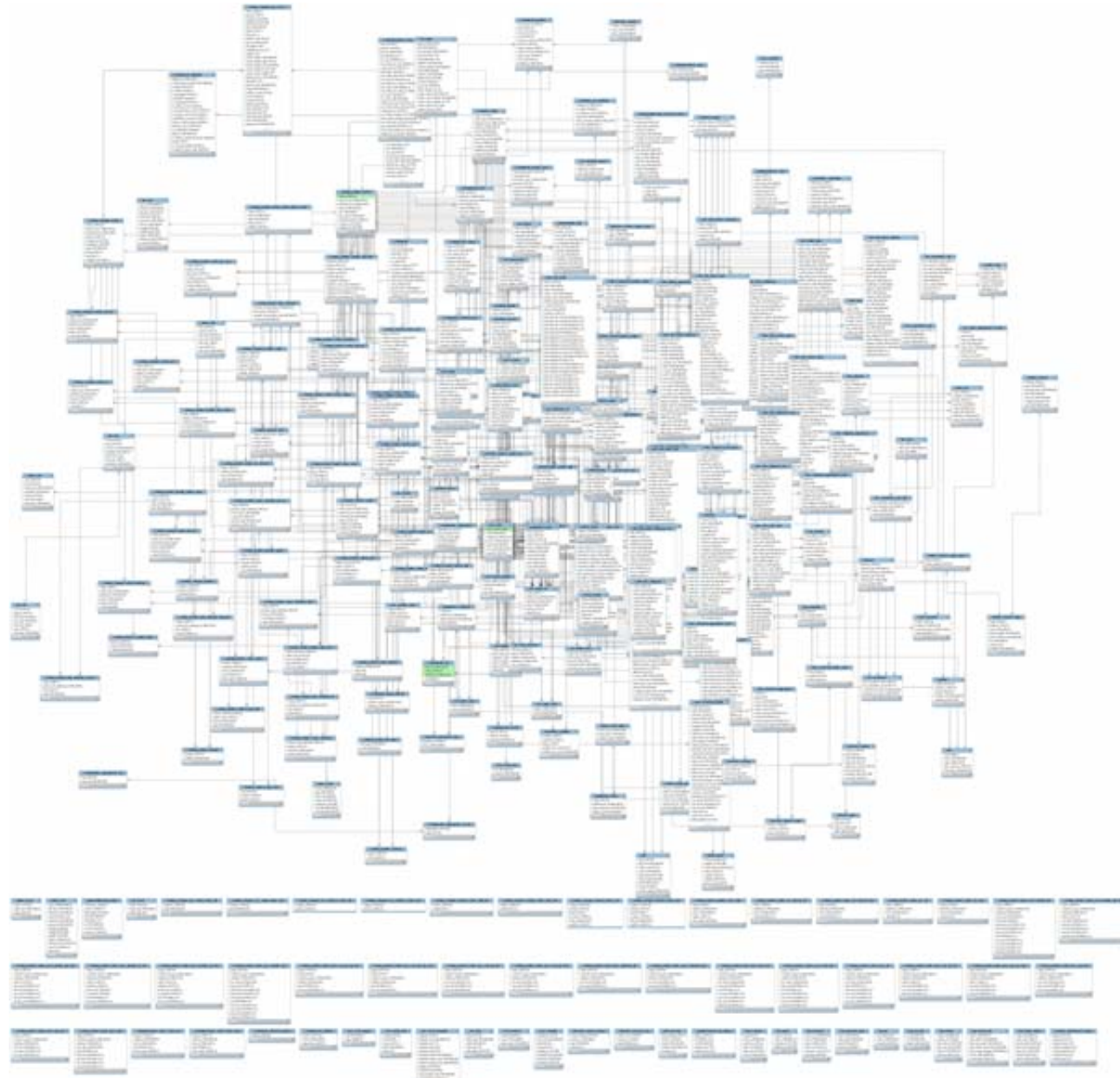
ABSTRACT

Online Transaction Processing (OLTP) databases include a suite of features — disk-resident B-trees and heap files, locking-based concurrency control, support for multi-threading — that were optimized for computer technology of the late 1970's. Advances in modern processors, memories, and networks mean that today's computers are vastly different from those of 30 years ago, such that many OLTP databases will now fit in main memory, and most OLTP transactions can be processed in milliseconds or less. Yet database architecture has changed little

1. INTRODUCTION

Modern general purpose online transaction processing (OLTP) database systems include a standard suite of features: a collection of on-disk data structures for table storage, including heap files and B-trees, support for multiple concurrent queries via locking-based concurrency control, log-based recovery, and an efficient buffer manager. These features were developed to support transaction processing in the 1970's and 1980's, when an OLTP database was many times larger than the main memory, and when the computers that ran these databases cost hundreds of thousands to

RDBMS: Complexity



**ALTERNATIVES TO RELATIONAL
DATABASES FOR QUERYING BIG
STRUCTURED DATA?**

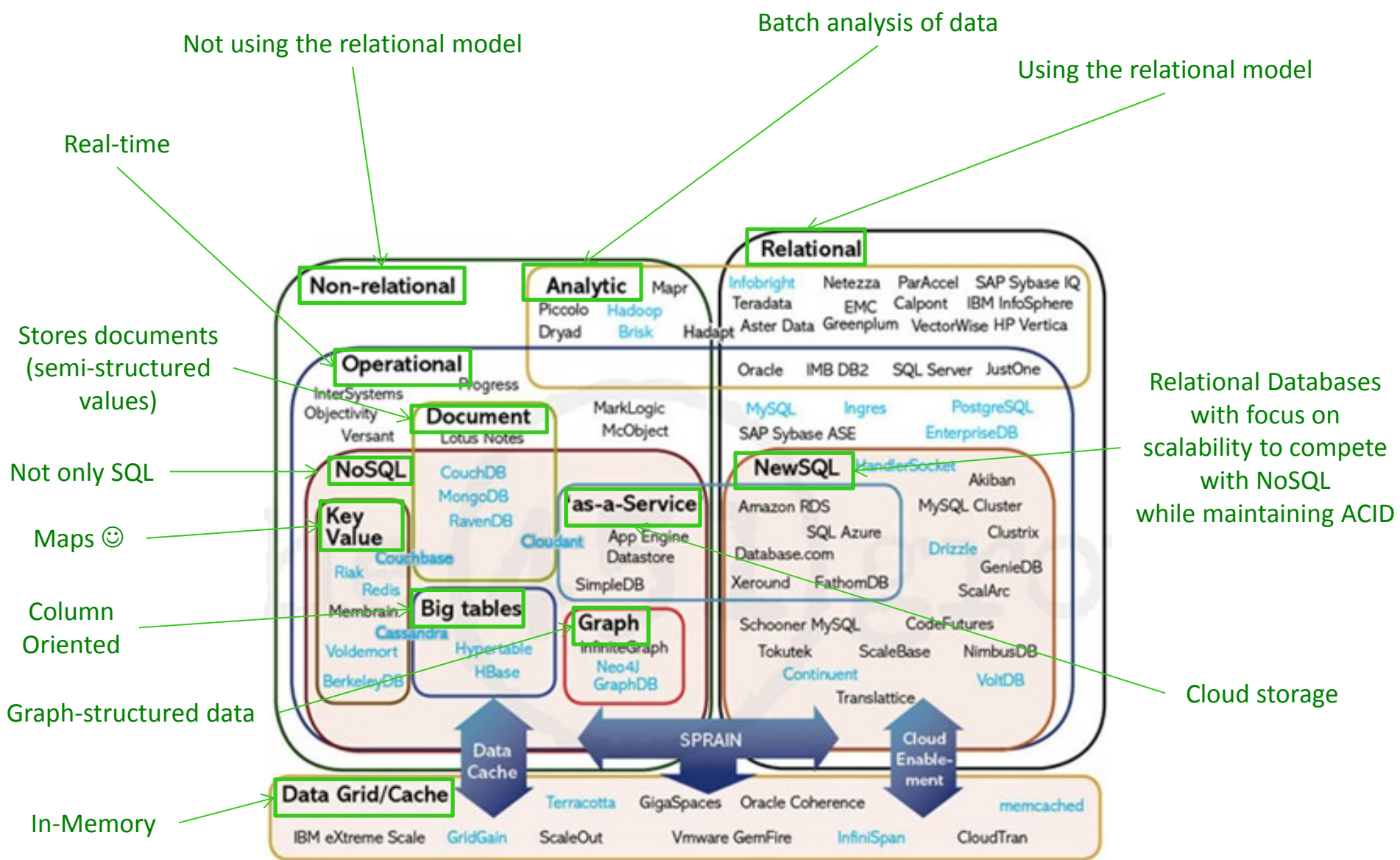
NoSQL

Anybody know anything
about NoSQL?



Not
Only **SQL**

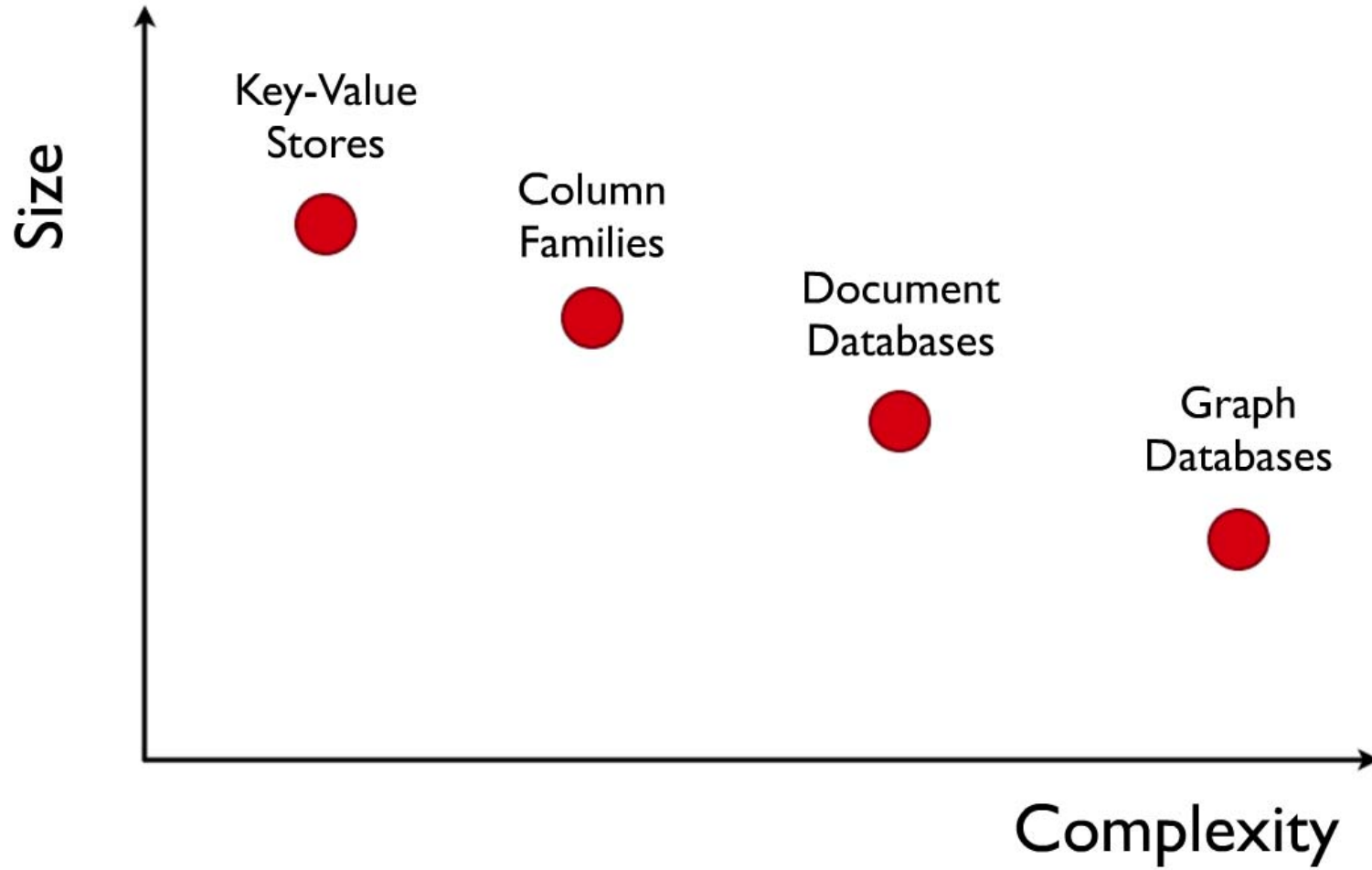
The Database Landscape



Rank			DBMS	Database Model	Score		
May 2016	Apr 2016	May 2015			May 2016	Apr 2016	May 2015
1.	1.	1.	Oracle	Relational DBMS	1462.02	-5.51	+19.93
2.	2.	2.	MySQL	Relational DBMS	1371.83	+1.72	+77.56
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1142.82	+7.77	+11.79
4.	4.	4.	MongoDB	Document store	320.22	+7.78	+42.90
5.	5.	5.	PostgreSQL	Relational DBMS	307.61	+3.89	+34.09
6.	6.	6.	DB2	Relational DBMS	185.96	+1.87	-15.09
7.	8.	8.	Cassandra	Wide column store	134.50	+4.83	+27.95
8.	7.	7.	Microsoft Access	Relational DBMS	131.58	-0.39	-14.00
9.	9.	10.	Redis	Key-value store	108.24	-3.00	+13.51
10.	10.	9.	SQLite	Relational DBMS	107.26	-0.70	+2.10
11.	11.	14.	Elasticsearch	Search engine	86.31	+3.73	+21.48
12.	13.	13.	Teradata	Relational DBMS	73.74	+1.48	+3.62
13.	12.	11.	SAP Adaptive Server	Relational DBMS	71.48	-1.84	-14.01
14.	14.	12.	Solr	Search engine	65.62	-0.40	-17.31
15.	15.	15.	HBase	Wide column store	51.84	+0.35	-9.87
16.	16.	17.	Hive	Relational DBMS	47.51	-1.57	+2.94
17.	17.	16.	FileMaker	Relational DBMS	46.71	+0.60	-6.19
18.	18.	18.	Splunk	Search engine	44.31	+1.96	+3.59
19.	19.	21.	SAP HANA	Relational DBMS	41.37	+1.02	+8.59
20.	21.	25.	MariaDB	Relational DBMS	33.97	+2.38	+10.37
21.	20.	22.	Neo4j	Graph DBMS	32.61	+0.70	+3.97
22.	22.	19.	Informix	Relational DBMS	30.58	-0.94	-6.16
23.	23.	20.	Memcached	Key-value store	27.90	-0.11	-5.21
24.	24.	24.	Couchbase	Document store	24.29	-0.73	-0.95
25.	25.	30.	Amazon DynamoDB	Multi-model	23.60	+0.48	+8.73

<http://db-engines.com/en/ranking>

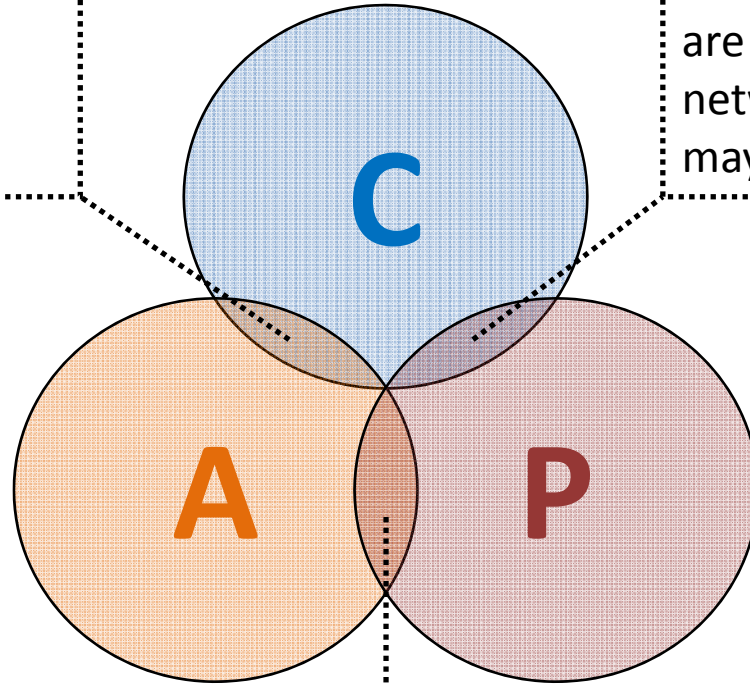
NoSQL



NoSQL: CAP (not ACID)

CA: Guarantees to give a correct response but only while network works fine (*Centralised / Traditional*)

CP: Guarantees responses are correct even if there are network failures, but response may fail (*Weak availability*)



(No intersection)

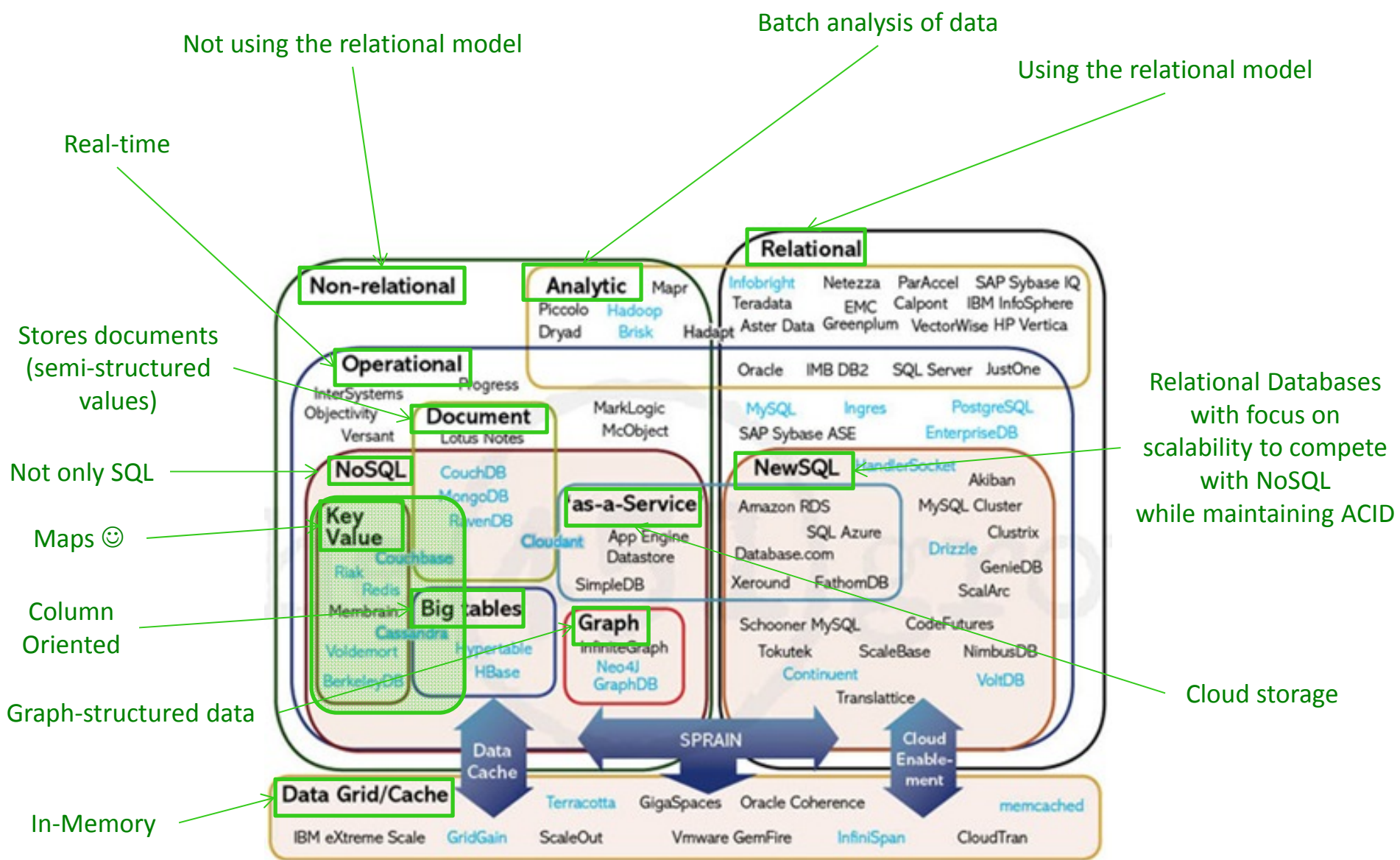
AP: Always provides a “best-effort” response even in presence of network failures (*Eventual consistency*)

NoSQL

- **Distributed!**
 - Sharding: splitting data over servers “horizontally”
 - Replication
- **Lower-level** than RDBMS/SQL
 - Simpler ad hoc APIs
 - But you build the application (programming not querying)
 - Operations simple and cheap
- **Different flavours** (for different scenarios)
 - Different CAP emphasis
 - Different scalability profiles
 - Different query functionality
 - Different data models

NOSQL: KEY-VALUE STORE

The Database Landscape



Key–Value Store Model

It's just a Map / Associate Array 😊

- `put(key, value)`
- `get(key)`
- `delete(key)`

Key	Value
Afghanistan	Kabul
Albania	Tirana
Algeria	Algiers
Andorra la Vella	Andorra la Vella
Angola	Luanda
Antigua and Barbuda	St. John's
...

But You Can Do a Lot With a Map

Key	Value
country:Afghanistan	capital@city:Kabul,continent:Asia,pop:31108077#2011
country:Albania	capital@city:Tirana,continent:Europe,pop:3011405#2013
...	...
city:Kabul	country:Afghanistan,pop:3476000#2013
city:Tirana	country:Albania,pop:3011405#2013
...	...
user:10239	basedIn@city:Tirana,post:{103,10430,201}
...	...

... actually you can model any data in a map (but possibly with a lot of redundancy and inefficient lookups if unsorted).

THE CASE OF AMAZON

The Amazon Scenario

Products Listings: prices, details, stock

The screenshot shows the Amazon search results page for the query "presenter". The header includes the Amazon logo, a search bar with "presenter" entered, and navigation links for "Shop by Department", "Aidan's Amazon.com", "Today's Deals", "Gift Cards", "Sell", and "Help". Below the header, it indicates "1-16 of 19,088 results for 'presenter'".

Show results for

- Office Products >
 - Office Presentation Remotes
 - Office Presentation Pointers
- Computers & Accessories >
 - Tablet Accessories
 - Computer Mice
- Cell Phones & Accessories >
 - Cell Phone Accessories
- Software >
 - Presentations

+ See All 29 Departments

Refine by

- International Shipping**
 - Ship to Ireland
- Eligible for Free Shipping**
 - Free Shipping by Amazon
- Brand**
 - Kensington
 - Logitech
 - Targus
 - Satechi
 - Infiniter
 - August

Point and zoom in presentations with Myo Armband
Shop now ▶

Related Searches: [logitech presenter](#), [mpow presenter](#), [wireless presenter](#).

Logitech Wireless Presenter R400
by Logitech
\$44.29 ~~\$49.99~~ Prime
Get it by **Wednesday, May 27**

Logitech Professional Presenter R800 with Green Laser Pointer
by Logitech
\$57.49 ~~\$79.99~~ Prime
Get it by **Wednesday, May 27**

Kensington Wireless Presenter with Laser Pointer
by Kensington

The Amazon Scenario

Customer info: shopping cart, account, etc.

 **Shopping Cart** Already a customer?
[Sign in](#)

[See more items like those in your Cart](#)

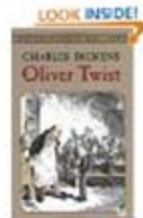
subtotal = \$88.77
Make any changes below? [Update](#)

Shopping Cart Items--To Buy Now		Price:	Qty:
<small>Item added on May 22, 2009</small>	The Principles of Beautiful Web Design - Jason Beard; Paperback Condition: New In Stock	\$26.37 You Save: \$13.58 (34%)	<input type="text" value="1"/>
Save for later	Delete	 Eligible for FREE Super Saver Shipping	
Add gift-wrap/note  (Learn more)			
<small>Item added on May 22, 2009</small>	Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition - Steve Krug; Paperback Condition: New In Stock	\$26.40 You Save: \$13.60 (34%)	<input type="text" value="1"/>
Save for later	Delete	 Eligible for FREE Super Saver Shipping	
Add gift-wrap/note  (Learn more)			

The Amazon Scenario

Recommendations, etc.:

Customers Who Bought This Item Also Bought



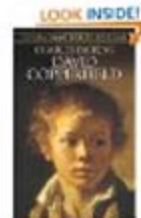
Oliver Twist (Dover Thrift Editions)

> Charles Dickens

★★★★☆ (213)

Paperback

\$3.50



David Copperfield (Dover Thrift Editions)

> Charles Dickens

★★★★☆ (196)

Paperback

\$5.00



JANE EYRE

> Charlotte Bronte

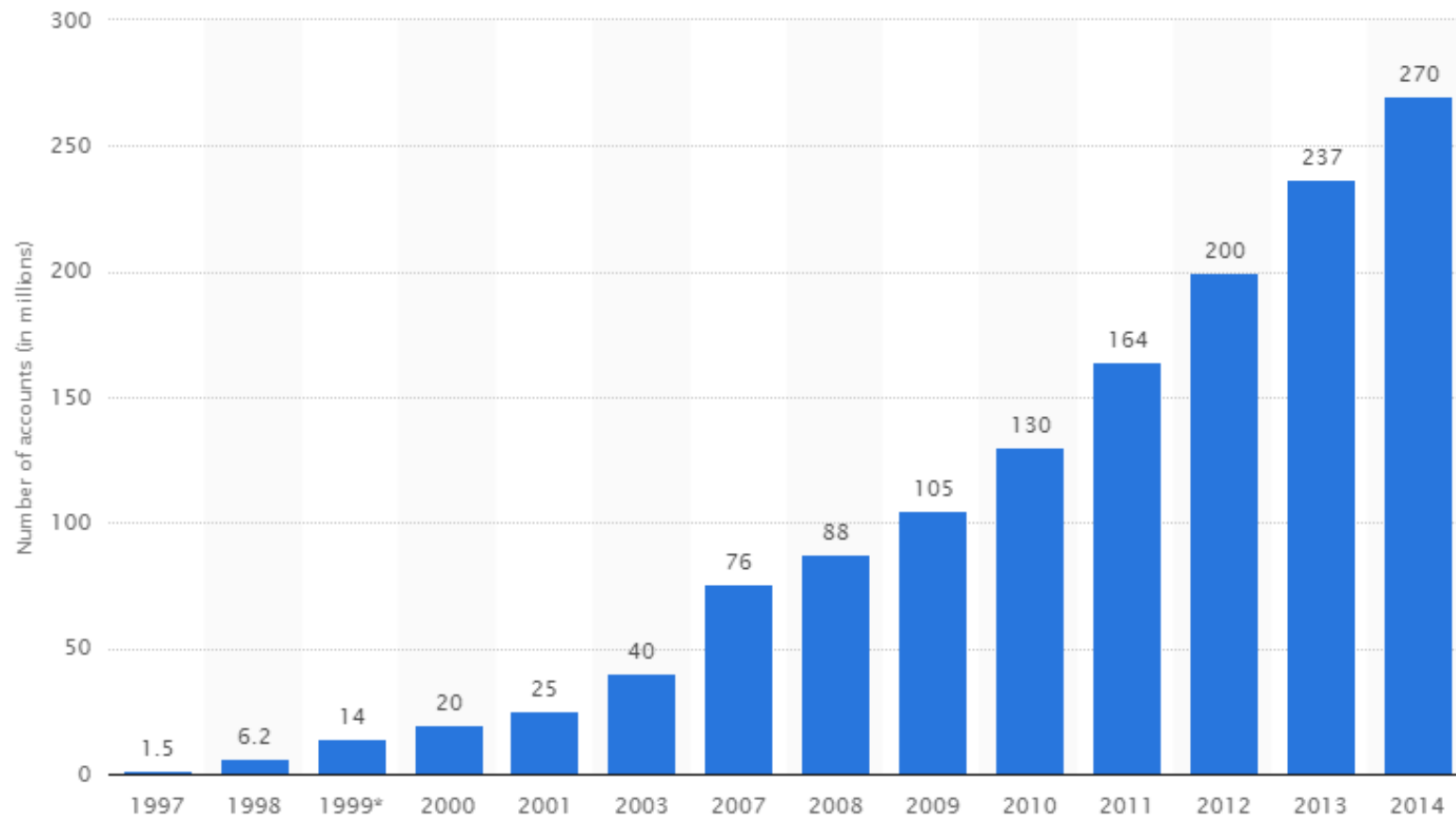
★★★★☆ (1,045)

Paperback

\$2.99

The Amazon Scenario

- Amazon customers:

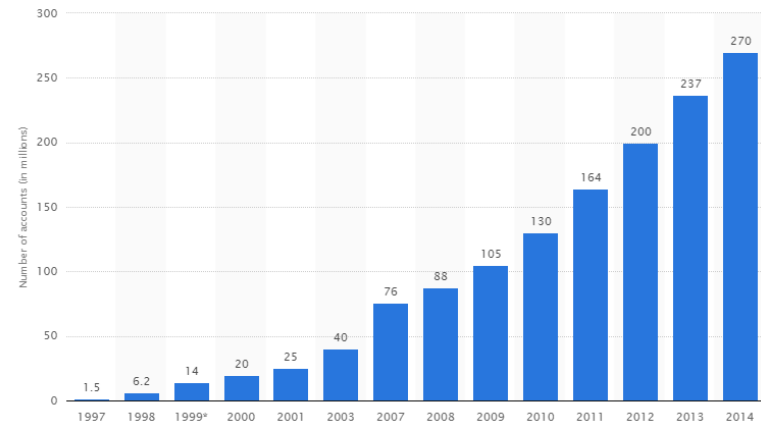


The Amazon Scenario



The Amazon Scenario

Databases struggling ...



But many Amazon services don't need:

- **SQL** (a simple map often enough)

or even:

- **transactions, strong consistency, etc.**

Key–Value Store: Amazon Dynamo(DB)

Dynamo: Amazon’s Highly Available Key-value Store

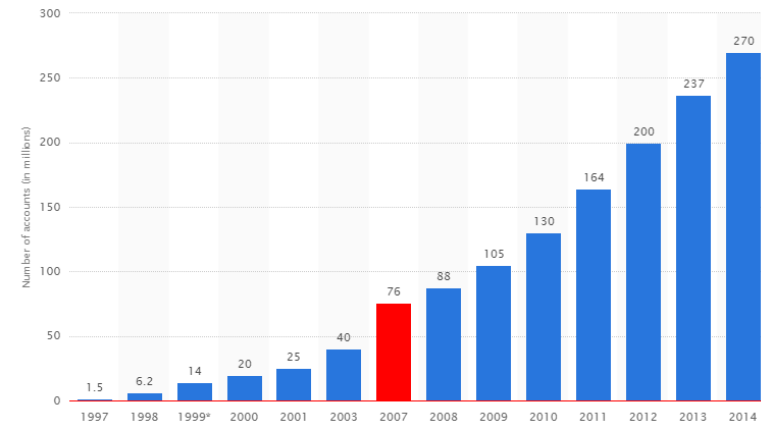
Giuseppe DeCandia, Deniz Hastorun, Madan Jambani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters

One of the lessons our organization has learned from operating Amazon’s platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are changing, or data centers are being



Goals:

Scalability (able to grow)

High availability (reliable)

Performance (fast)

Don't need full SQL, don't need full ACID

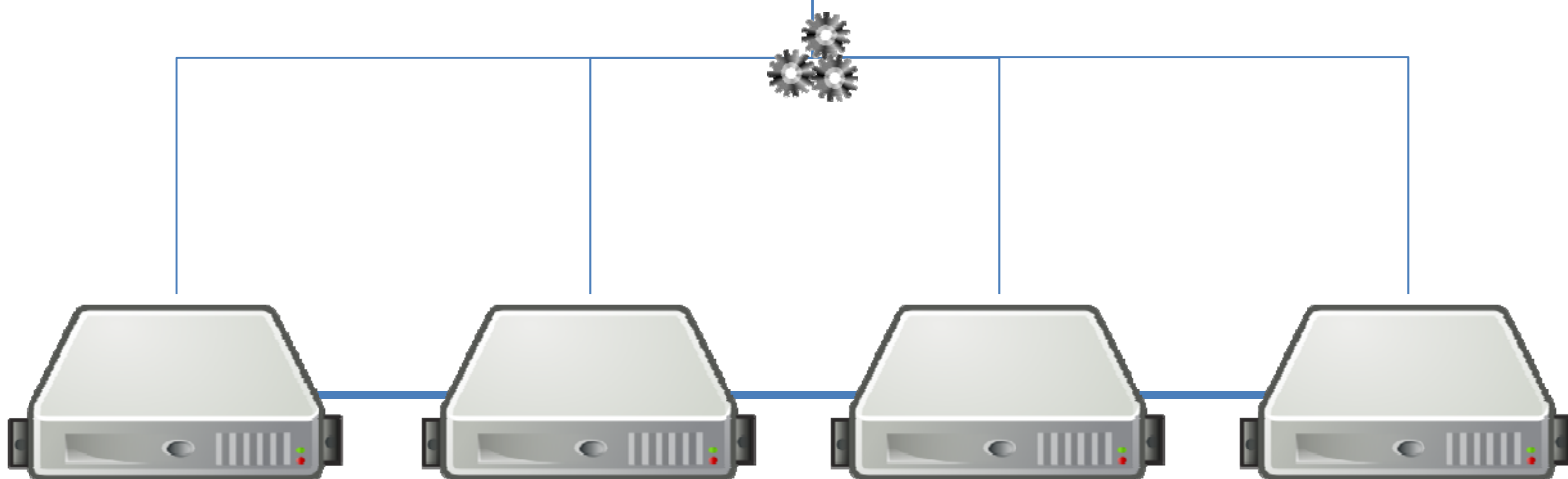
Key–Value Store: Distribution

How might a key–value store be distributed over multiple machines?

Key	Value
country:Afghanistan	capital@city:Kabul,continent:Asia,pop:31108077#2011
country:Albania	capital@city:Tirana,continent:Europe,pop:3011405#2013
...	...
city:Kabul	country:Afghanistan,pop:3476000#2013
city:Tirana	country:Albania,pop:3011405#2013
...	...
user:10239	basedIn@city:Tirana,post:{103,10430,201}
...	...

$$\text{mod}(\text{hash}(key), m)$$

Or a custom partitioner ...



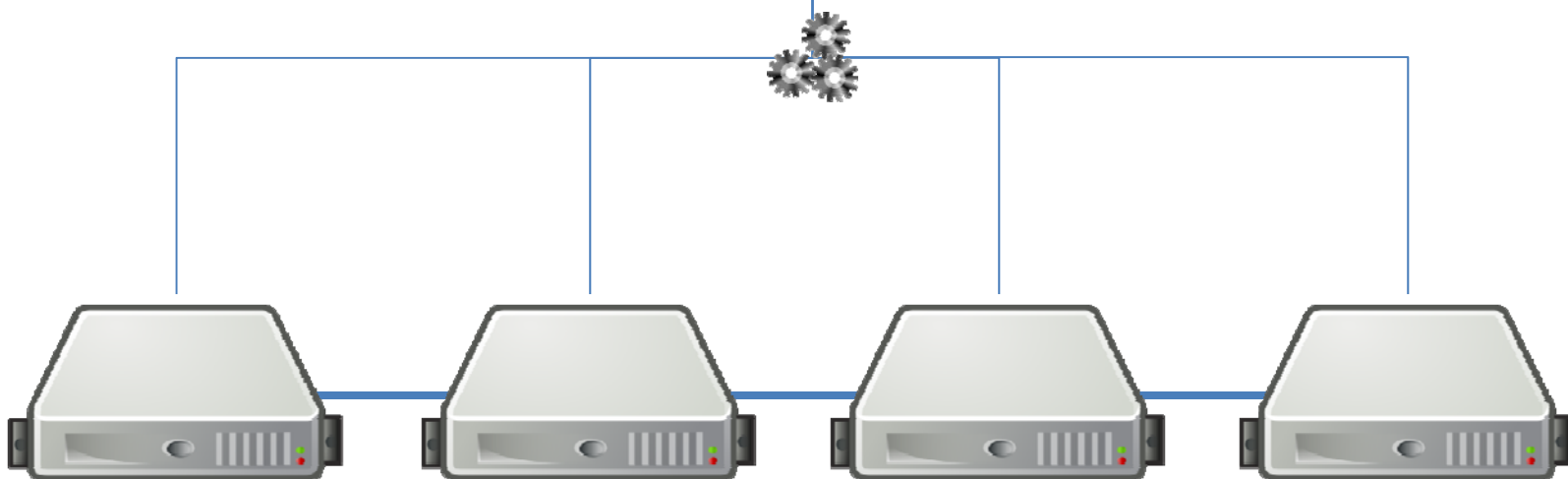
Key–Value Store: Distribution

What happens if a machine joins or leaves half way through?

Key	Value
country:Afghanistan	capital@city:Kabul,continent:Asia,pop:31108077#2011
country:Albania	capital@city:Tirana,continent:Europe,pop:3011405#2013
...	...
city:Kabul	country:Afghanistan,pop:3476000#2013
city:Tirana	country:Albania,pop:3011405#2013
...	...
user:10239	basedIn@city:Tirana,post:{103,10430,201}
...	...

$$\text{mod}(\text{hash}(key), m)$$

Or a custom partitioner ...



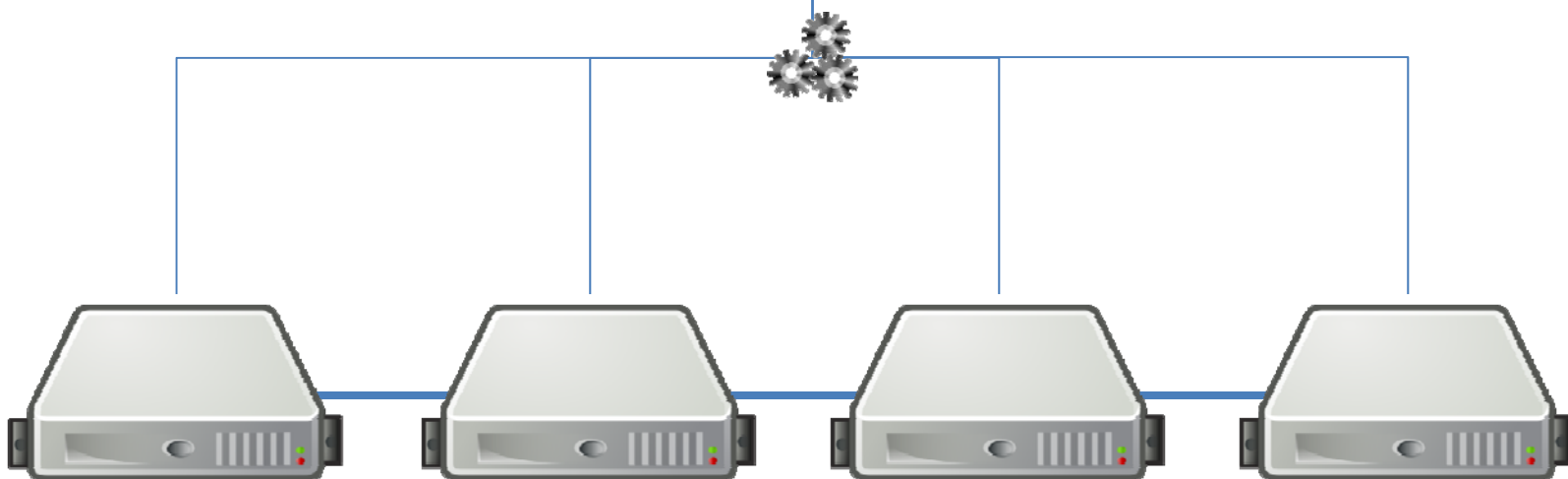
Key–Value Store: Distribution

How can we solve this?

Key	Value
country:Afghanistan	capital@city:Kabul,continent:Asia,pop:31108077#2011
country:Albania	capital@city:Tirana,continent:Europe,pop:3011405#2013
...	...
city:Kabul	country:Afghanistan,pop:3476000#2013
city:Tirana	country:Albania,pop:3011405#2013
...	...
user:10239	basedIn@city:Tirana,post:{103,10430,201}
...	...

$$\text{mod}(\text{hash}(key), m)$$

Or a custom partitioner ...

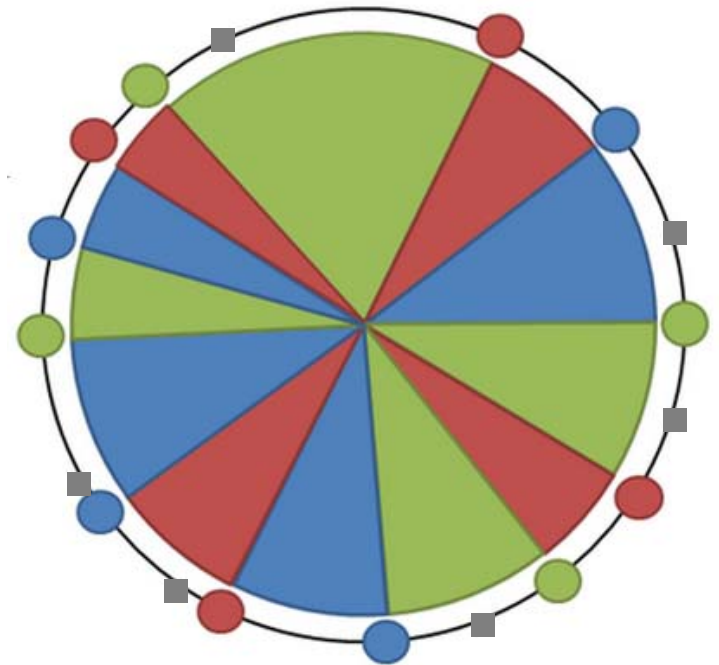


Consistent Hashing

Avoid re-hashing everything

- Hash using a ring
- Each machine picks n pseudo-random points on the ring
- Machine responsible for arc after its point
- If a machine leaves, its range moves to previous machine
- If machine joins, it picks new points
- Objects mapped to ring 😊

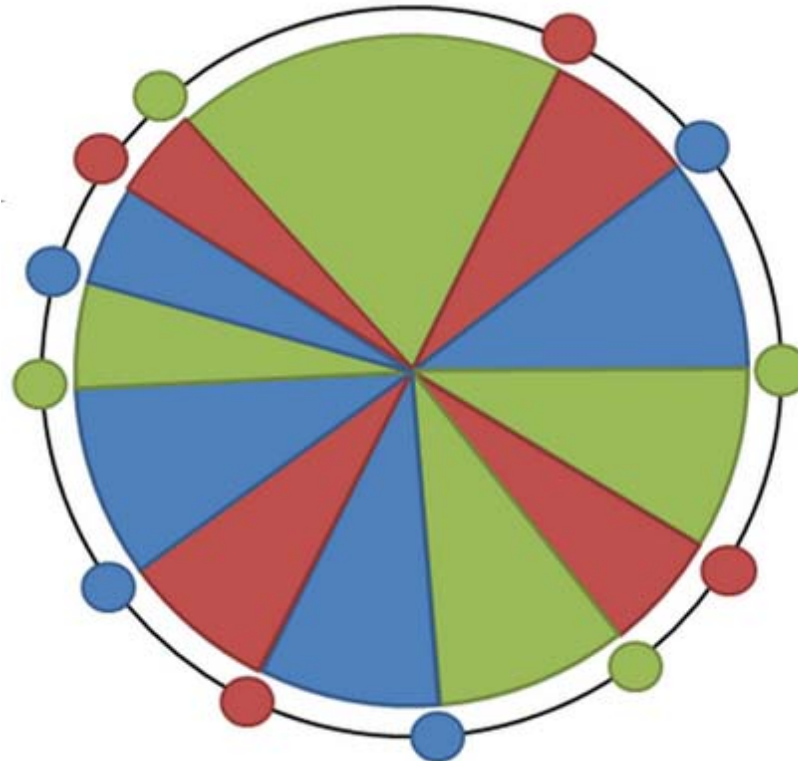
How many keys (on average) need to be moved if a machine joins or leaves?



Amazon Dynamo: Hashing

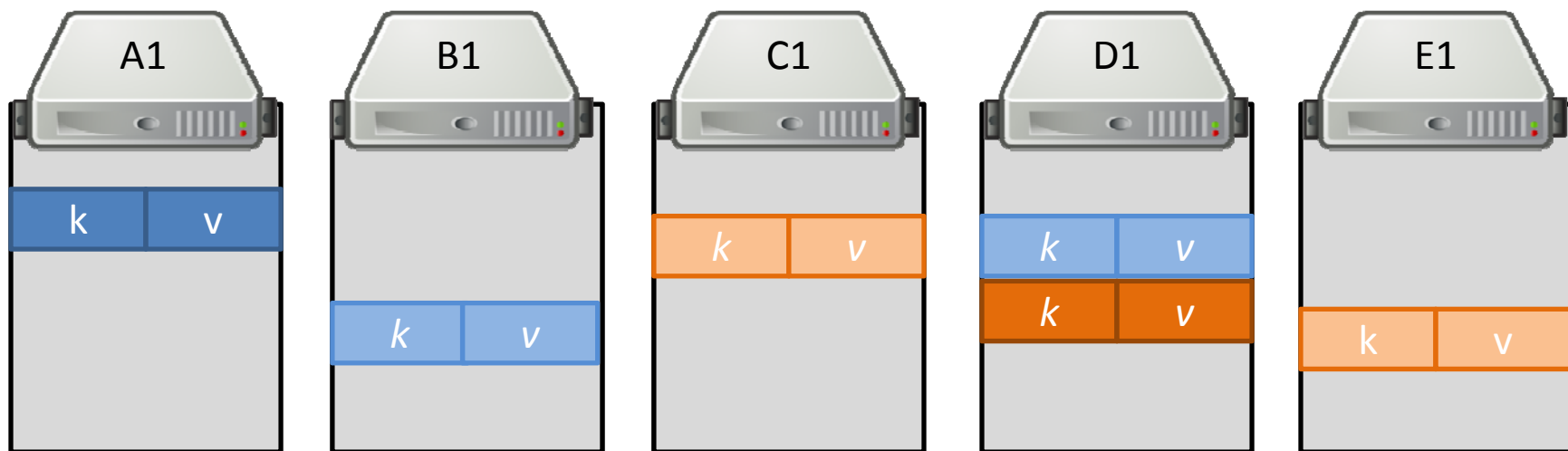


- Consistent Hashing (128-bit MD5)



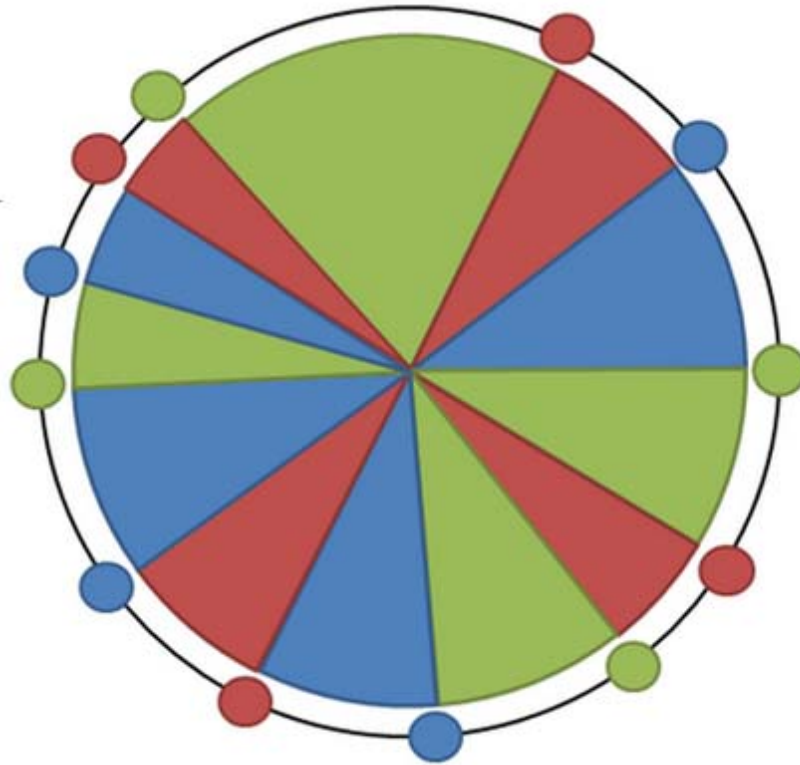
Key–Value Store: Replication

- A set replication factor (here 3)
- Commonly primary / secondary replicas
 - Primary replica elected from secondary replicas in the case of failure of primary



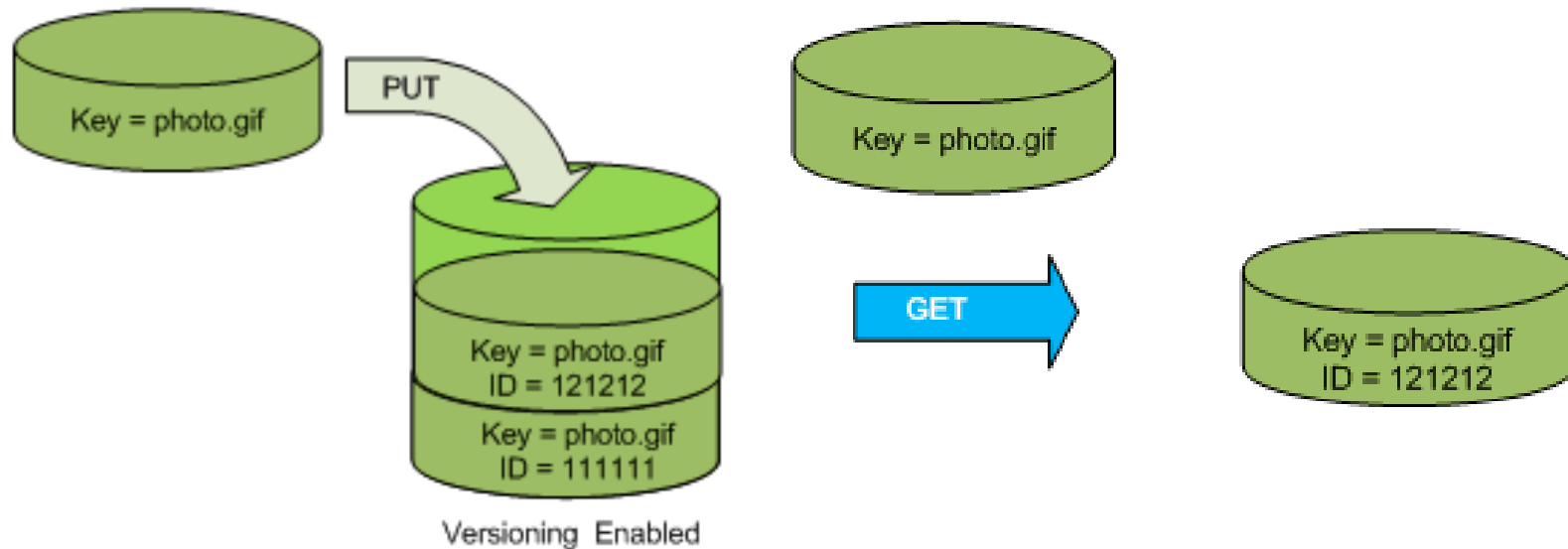
Amazon Dynamo: Replication

- Replication factor of n
 - Easy: pick n next buckets (different machines!)



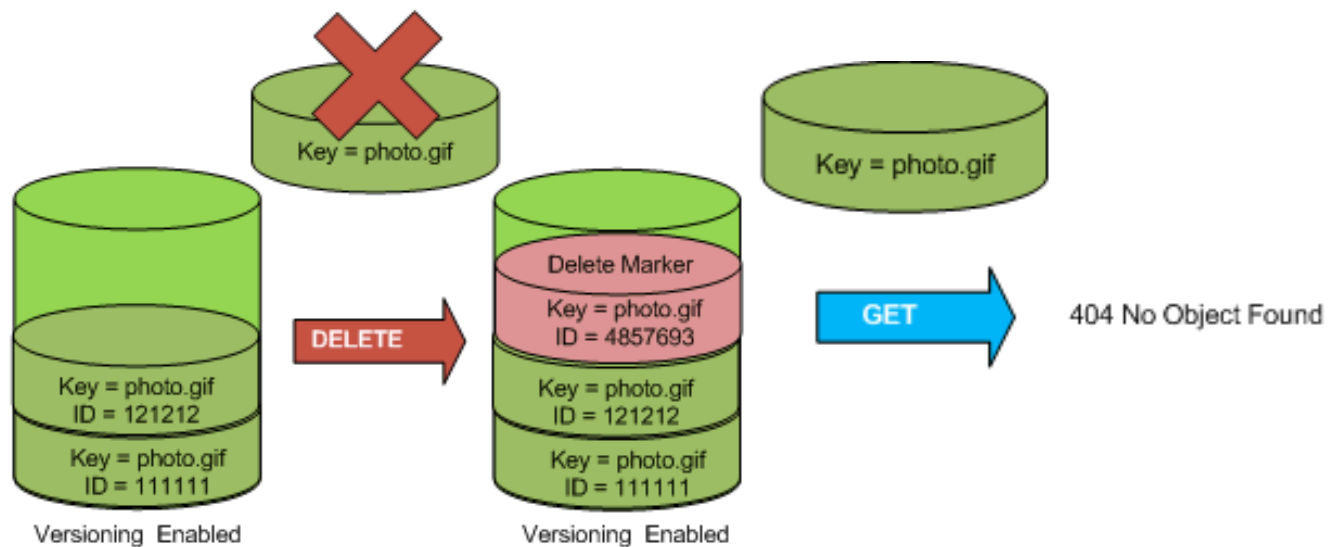
Amazon Dynamo: Object Versioning

- Object Versioning (per bucket)
 - PUT doesn't overwrite: pushes version
 - **GET** returns most recent version



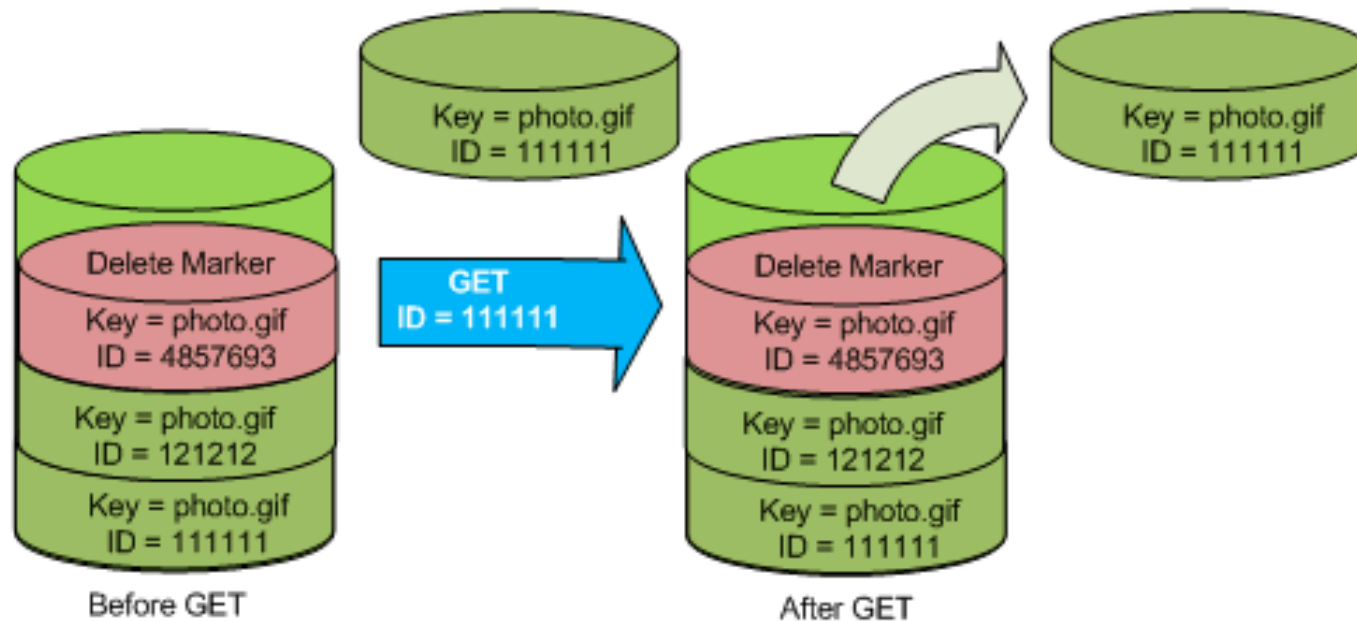
Amazon Dynamo: Object Versioning

- Object Versioning (per bucket)
 - **DELETE** doesn't wipe
 - **GET** will return not found



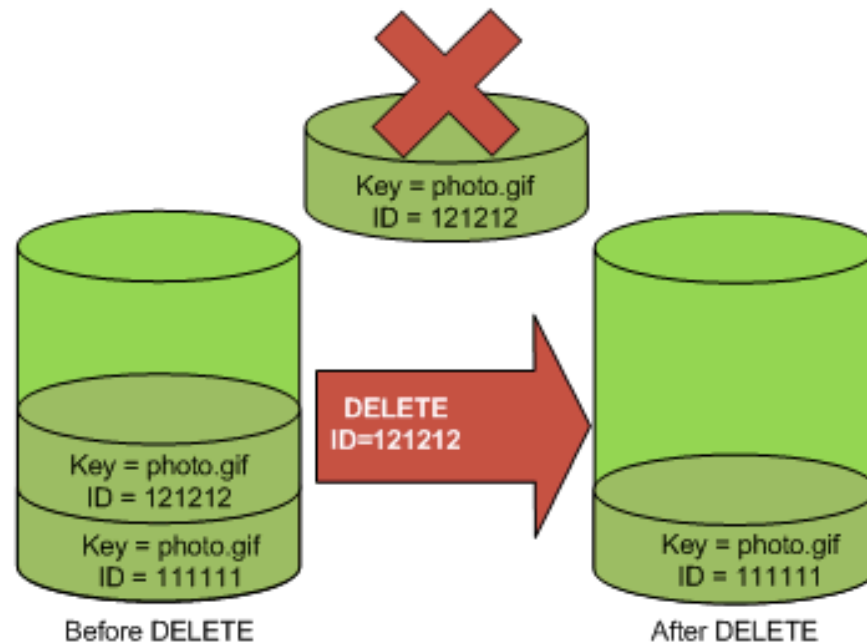
Amazon Dynamo: Object Versioning

- Object Versioning (per bucket)
 - GET by version



Amazon Dynamo: Object Versioning

- Object Versioning (per bucket)
 - **PERMANENT DELETE** by version ... wiped



Amazon Dynamo: Model

- Named table with primary key and a value
- Primary key is hashed / unordered

Countries	
Primary Key	Value
Afghanistan	capital:Kabul,continent:Asia,pop:31108077#2011
Albania	capital:Tirana,continent:Europe,pop:3011405#2013
...	...

Cities	
Primary Key	Value
Kabul	country:Afghanistan,pop:3476000#2013
Tirana	country:Albania,pop:3011405#2013
...	...

Amazon Dynamo: Model

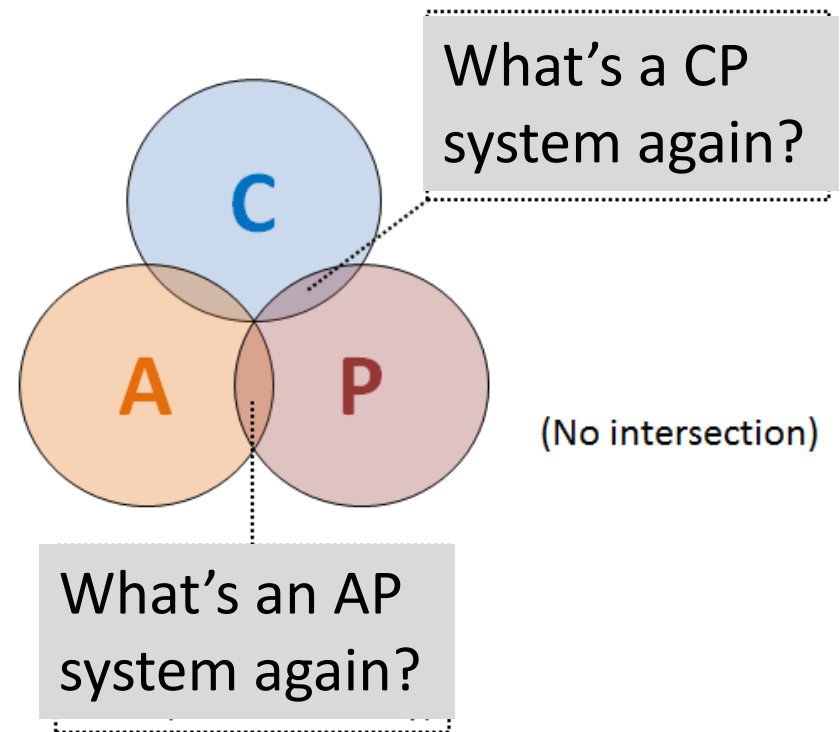
- Dual primary key also available:
 - Hash: unordered
 - Range: ordered

Countries		
Hash Key	Range Key	Value
Vatican City	839	capital:Vatican City,continent:Europe
Nauru	9945	capital:Yaren,continent:Oceania
...		...

Amazon Dynamo: CAP

Two options for each table:

- **AP**: Eventual consistency,
High availability
- **CP**: Strong consistency,
Lower availability







Amazon Dynamo: Consistency

- **Gossiping**
 - Keep alive messages sent between nodes with state
 - Dynamo largely decentralised (no master node)
- **Quorums:**
 - Multiple nodes responsible for a read (R) or write (W)
 - At least R or W nodes acknowledge for success
 - Higher R or W = Higher consistency, lower availability
- **Hinted Handoff**
 - For transient failures
 - A node “covers” for another node while its down

Amazon Dynamo: Consistency

- Two versions of one shopping cart:

Shopping Cart		Price	Quantity
	WD My Passport Ultra 2TB Portable External USB 3.0 Hard Drive with Auto Backup - Red by Western Digital In Stock Eligible for FREE Shipping <input type="checkbox"/> This is a gift Learn more Delete Save for later	\$90.99 You save: \$49.00 (35%)	1
	Logitech Wireless Presenter R400 by Logitech In Stock Eligible for FREE Shipping <input type="checkbox"/> This is a gift Learn more Delete Save for later	\$44.29 You save: \$5.70 (11%)	1
Subtotal (2 items): \$135.28		Total savings: \$54.70	

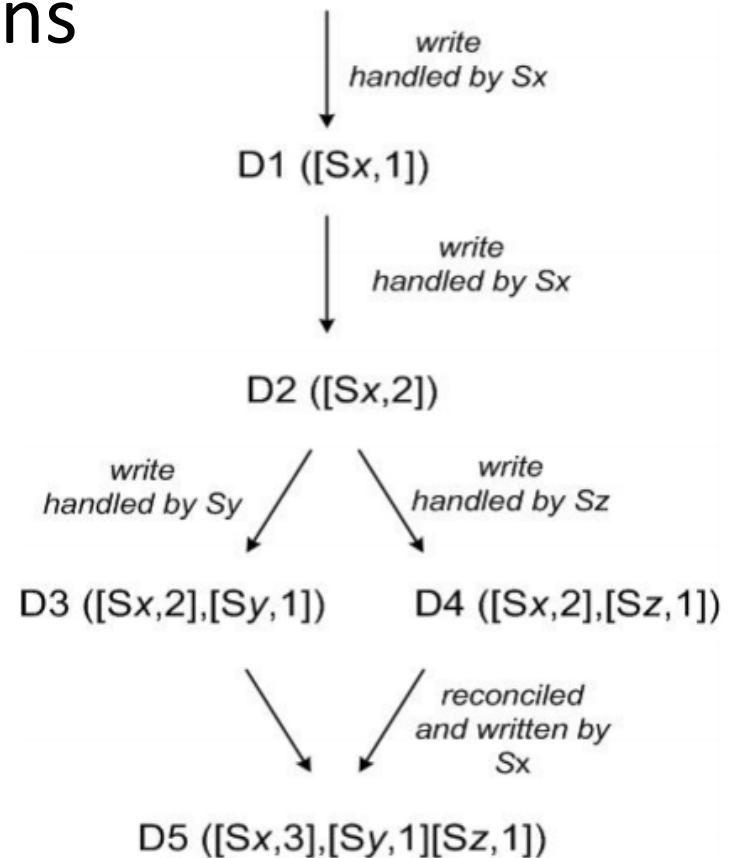
Shopping Cart		Price	Quantity
	AKG Perception P120 Professional Studio Microphone, Silver by AKG Pro Audio Only 2 left in stock. Shipped from: Sam Ash Gift options not available. Learn more Delete Save for later	\$99.00 You save: \$30.00 (23%)	1
	Logitech Wireless Presenter R400 by Logitech In Stock Eligible for FREE Shipping <input type="checkbox"/> This is a gift Learn more Delete Save for later	\$44.29 You save: \$5.70 (11%)	1
Subtotal (2 items): \$143.29		Total savings: \$35.70	

How best to handle multiple conflicting versions of a value (knowing as reconciliation)?

- **Application knows best**
(... and must support multiple versions being returned)

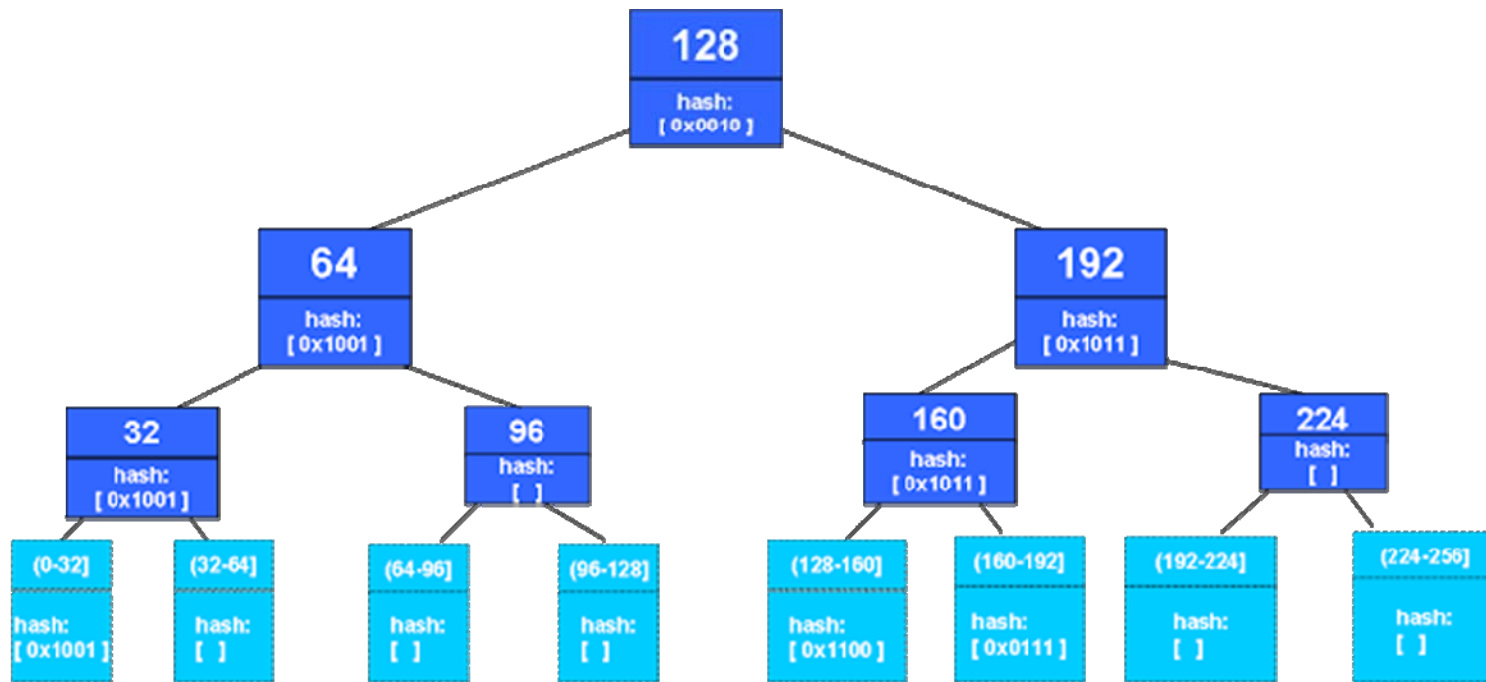
Amazon Dynamo: Vector Clocks

- Vector Clock: A list of pairs indicating a node (i.e., a server) and a time stamp
- Used to track/order versions



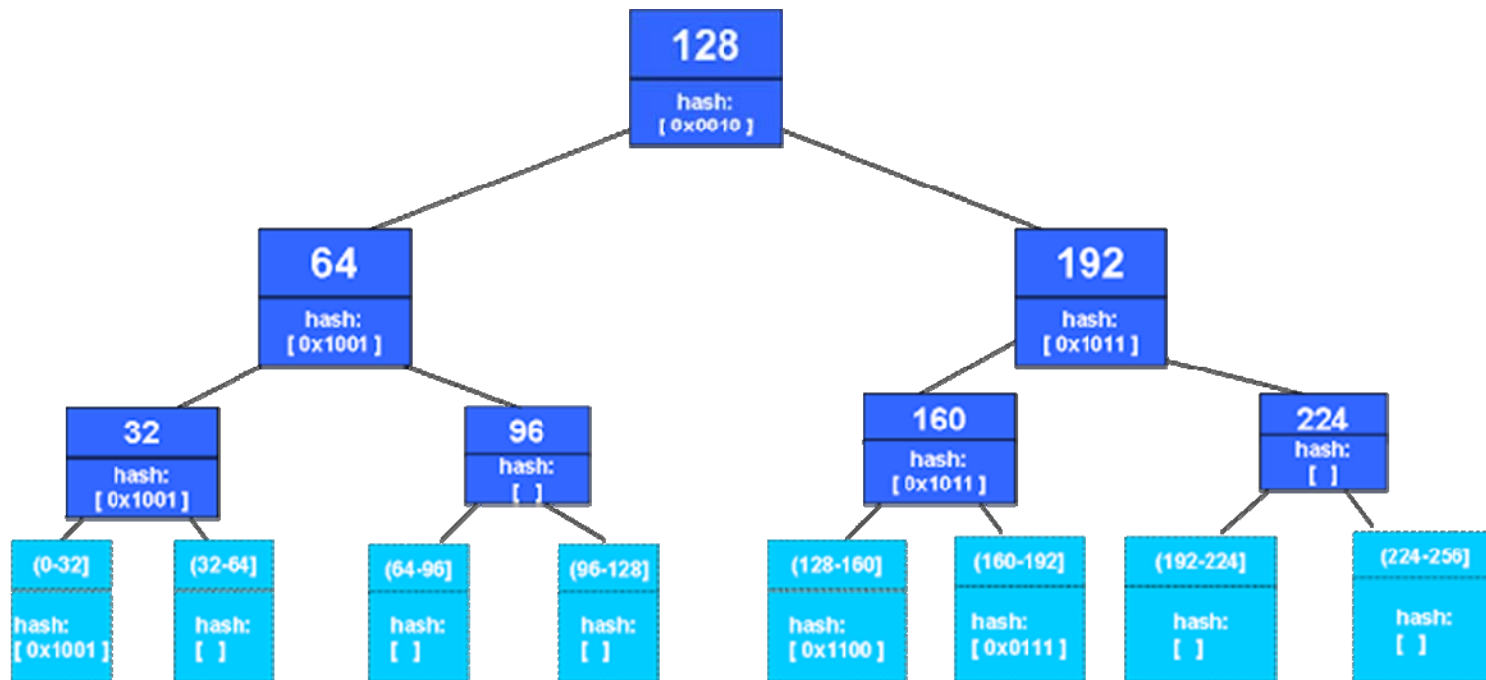
Amazon Dynamo: Eventual Consistency using Merkle Trees

- **Merkle tree** is a hash tree
- Nodes have hashes of their children
- Leaf node hashes from data: keys or ranges



Amazon Dynamo: Eventual Consistency using Merkle Trees

- Easy to verify regions of the Map
- Can compare level-at-a-time



Amazon Dynamo: Budgeting

- Assign throughput per table: allocate resources
- Reads (4 KB resolution):

Expected Item Size	Consistency	Desired Reads Per Second	Provisioned Throughput Required
4 KB	Strongly consistent	50	50
8 KB	Strongly consistent	50	100
4 KB	Eventually consistent	50	25
8 KB	Eventually consistent	50	50

- Writes (1 KB resolution)

Expected Item Size	Desired Writes Per Second	Provisioned Throughput Required
1 KB	50	50
2 KB	50	100

Read More ...



Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati,
Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall
and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters

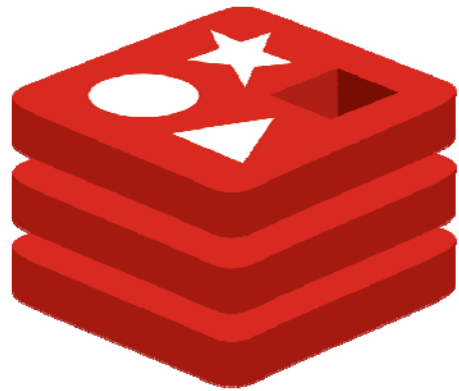
One of the lessons our organization has learned from operating Amazon's platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being

OTHER KEY-VALUE STORES

Other Key–Value Stores



Other Key–Value Stores



redis



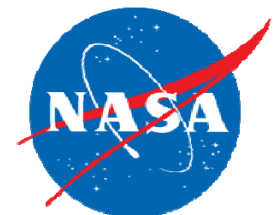
StackExchange



Other Key-Value Stores

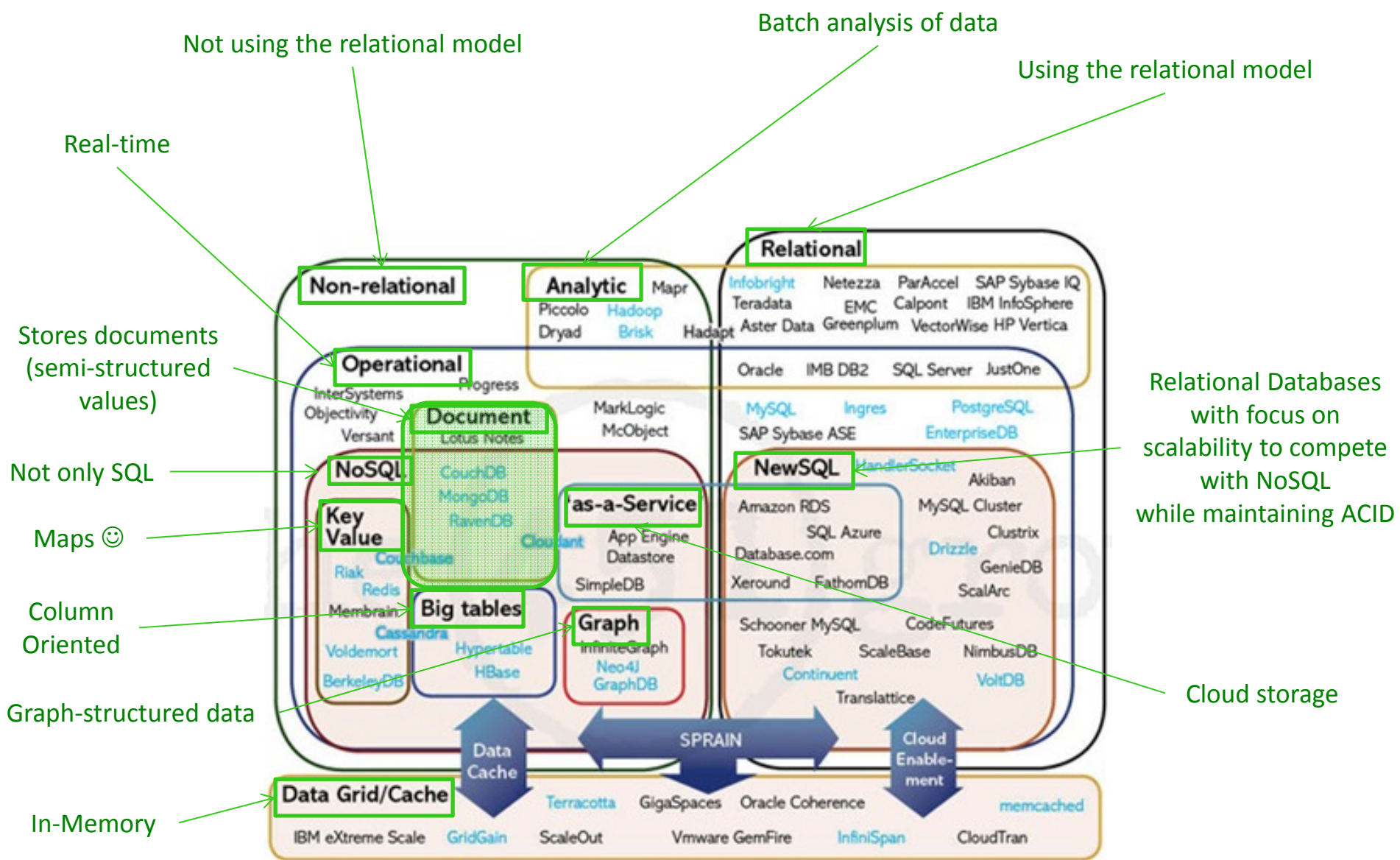


cassandra



NOSQL: DOCUMENT STORE

The Database Landscape



Key–Value Stores: Values are Documents

Key	Value
country:Afghanistan	<pre><country> <capital>city:Kabul</capital> <continent>Asia</continent> <population> <value>31108077</value> <year>2011</year> </population> </country></pre>
...	...

- Document-type depends on store
 - XML, JSON, Blobs, Natural language
- Operators for documents
 - e.g., filtering, inv. indexing, XML/JSON querying, etc.

MongoDB: JSON Based

Key	Value (Document)
6ads786a5a9	<pre>{ "_id" : ObjectId("6ads786a5a9"), "name" : "Afghanistan", "capital" : "Kabul", "continent" : "Asia", "population" : { "value" : 31108077, "year" : 2011 } }</pre>
...	...



- Can invoke Javascript over the JSON objects
- Document fields can be indexed

Document Stores



Couchbase

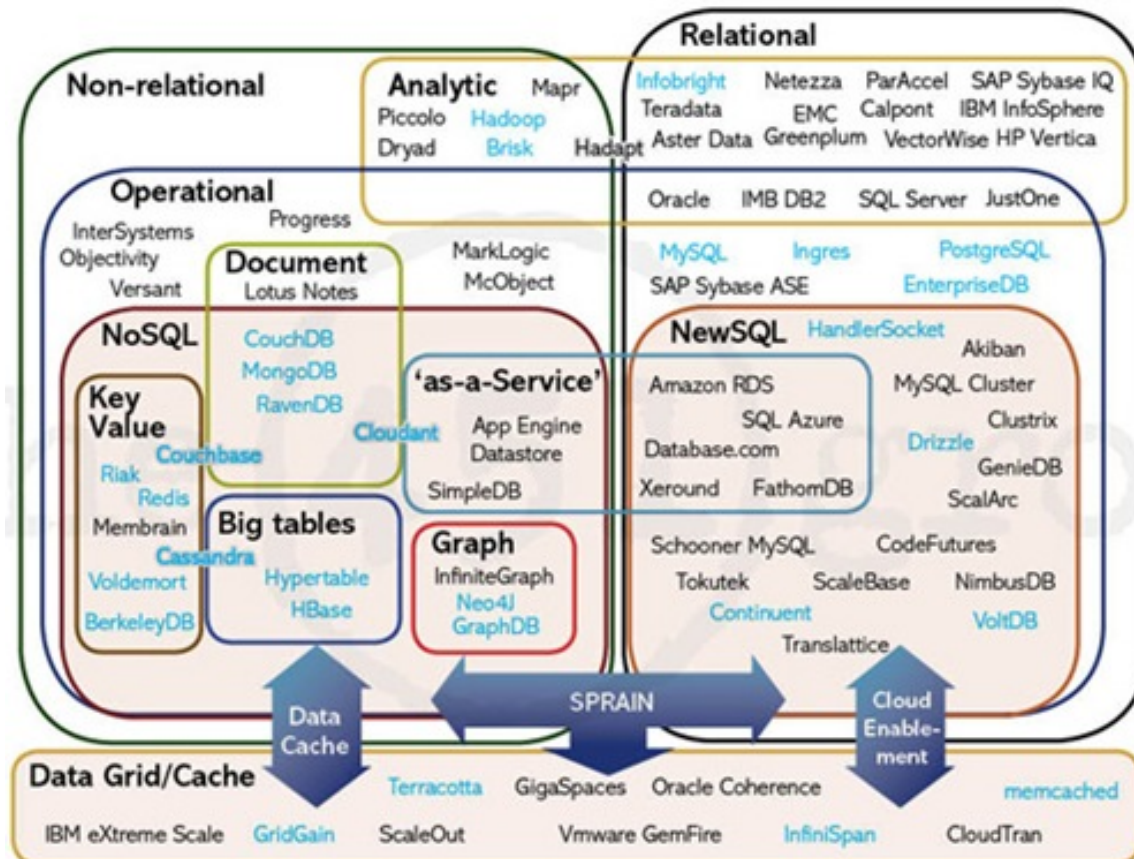


RECAP

Recap

- Relational Databases don't solve everything
 - SQL and ACID add overhead
 - Distribution not so easy
- NoSQL: what if you don't need SQL or ACID?
 - Something simpler
 - Something more scalable
 - Trade efficiency against guarantees

Recap



Recap

- **Key–value stores** inspired by Amazon Dynamo
 - Distributed maps
 - Hash keys and range keys
 - Table names
 - Consistent hashing
 - Replication
 - Object versioning / vector clocks
 - Gossiping / Quorums / Hinted Hand-offs
 - Merkle trees
 - Budgeting
- **Document stores:** documents as values
 - Support for JSON, XML values, field indexing, etc.



Questions

