Couchbase
Learning Services

Agustín F. Calderón M.

Couchbase
Learning Services

CB020 Fundamentals of NoSQL Data Management

CB020 Fundamentals of NoSQL Data Management

Oct 23 , 2016

Couchbase

Learning Services

What is making data go “big”?

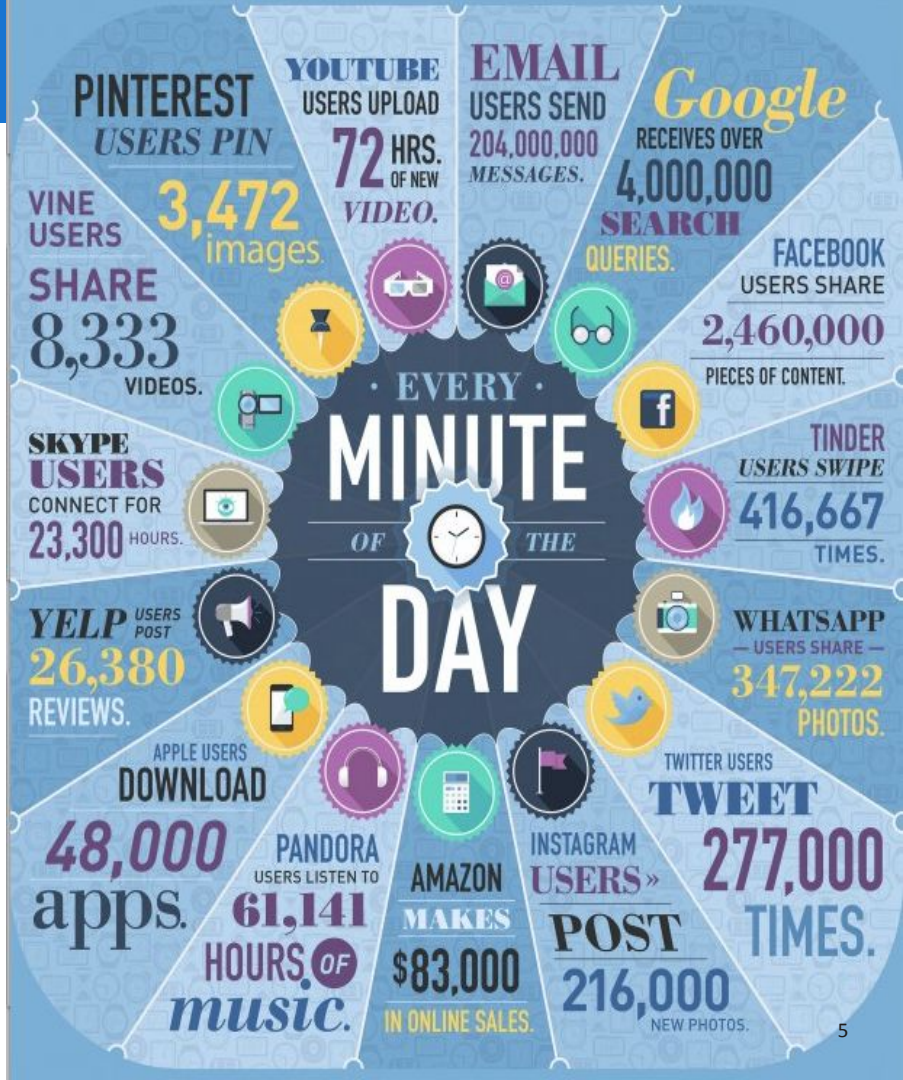
The logical consequence of modern technology.

Magnitude order

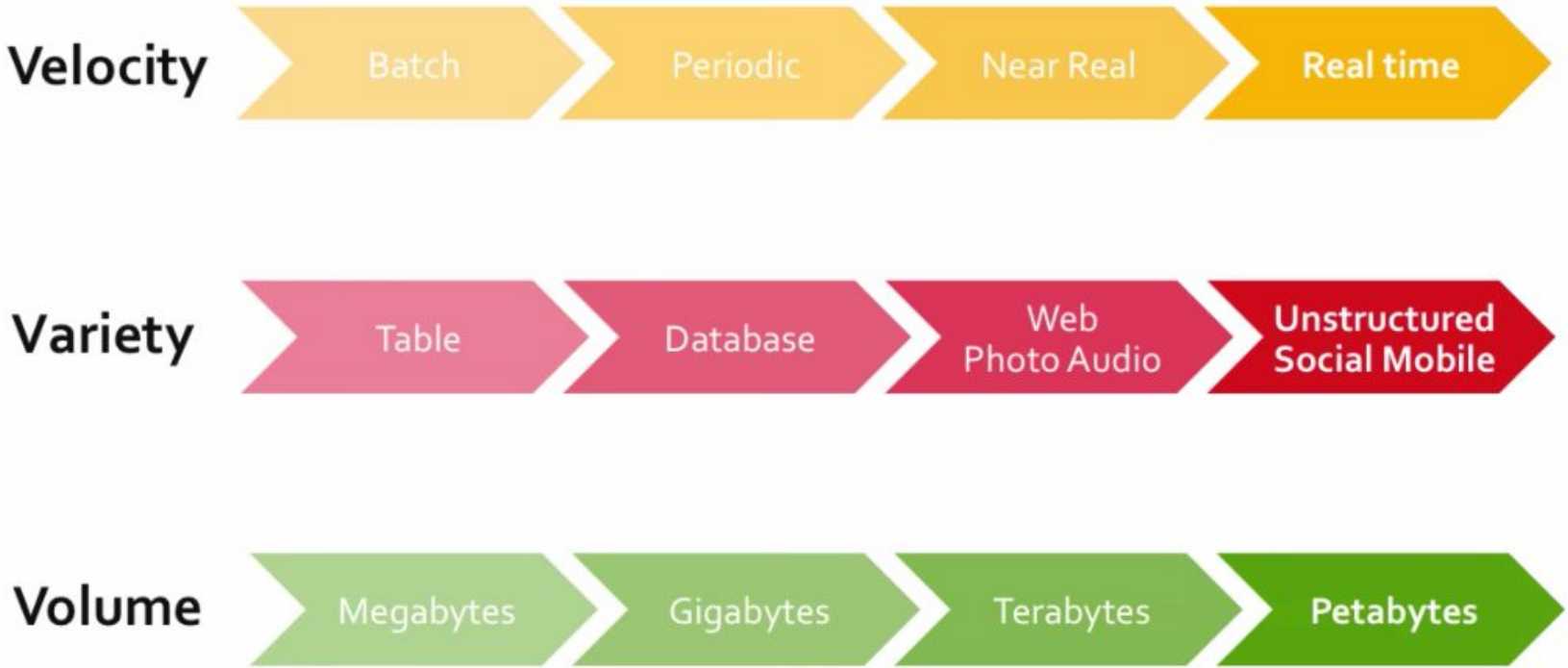
Internet: 2.400 Millions of users.

The global internet population grew 14.3% from 2011-2013.

1.75 Billion smartphone users



What trends are driving us toward big data?



Why are social and mobile content so significant?

U.S. adults spend
7.5 hours per day online

Every single minute …

$272,000 in purchasing
$391,680,000 per day
207 new mobile activations
298,080 per day

Now, imagine these numbers worldwide …

1.75
Billion Smartphone Users

35
Billion Hrs./Mo. Spent Online

3
Billion Global Online Population

How have cloud-based apps changed the game?

Client-Server
apps on LAN/WAN

On premise apps with thousands of users
High-end centralized servers
Gigabytes of well-known, structured data

Tablet-Mobile-Browser
apps in the cloud

Cloud based apps with millions of users
Robust clusters of commodity servers
Terabytes+ of ever-changing, unstructured data

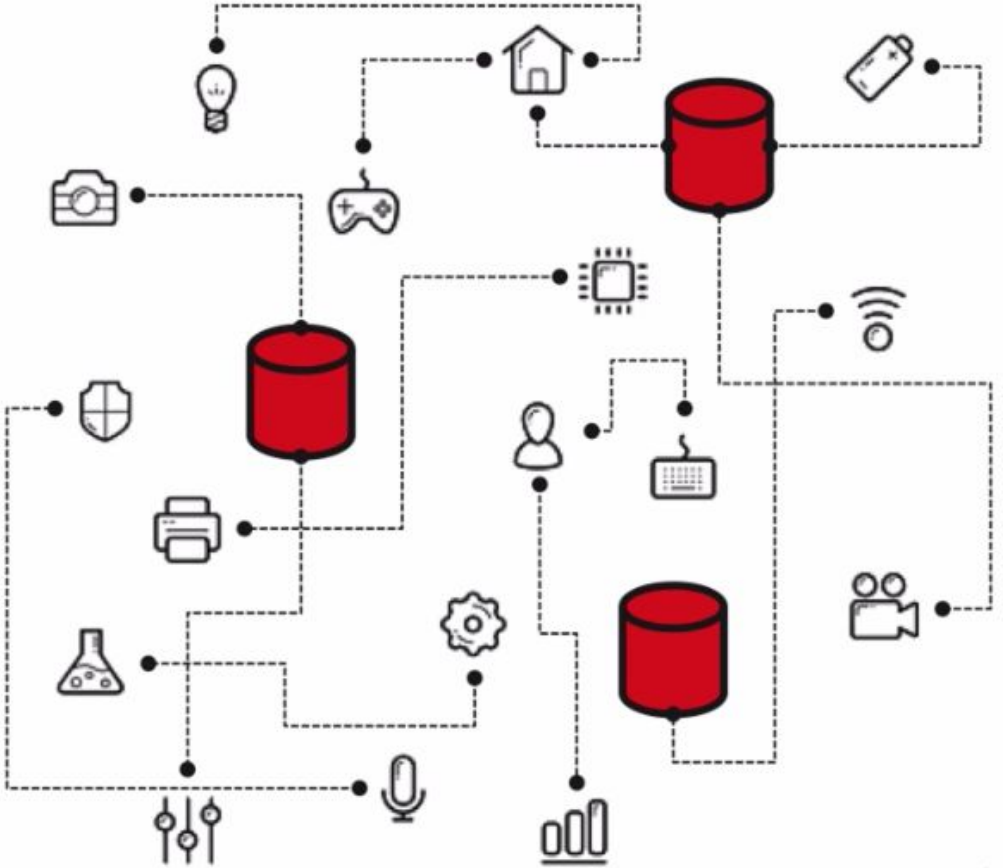
What is the “internet of things”?

Currently 14 billion things connected to the internet

By 2020? 32 billion …

Many tracking unstructured data over time

20% of all data generated from embedded sensors



Why does personalization drive big data flows?

Over 200 million people shop online in the U.S. alone

Triple this worldwide

Each transaction
- ✓ closes the deal by *integrating multiple systems*
- ✓ adds to each customer's *ever-growing history*

Various data structures …
Different platforms …

What is “unstructured” data?

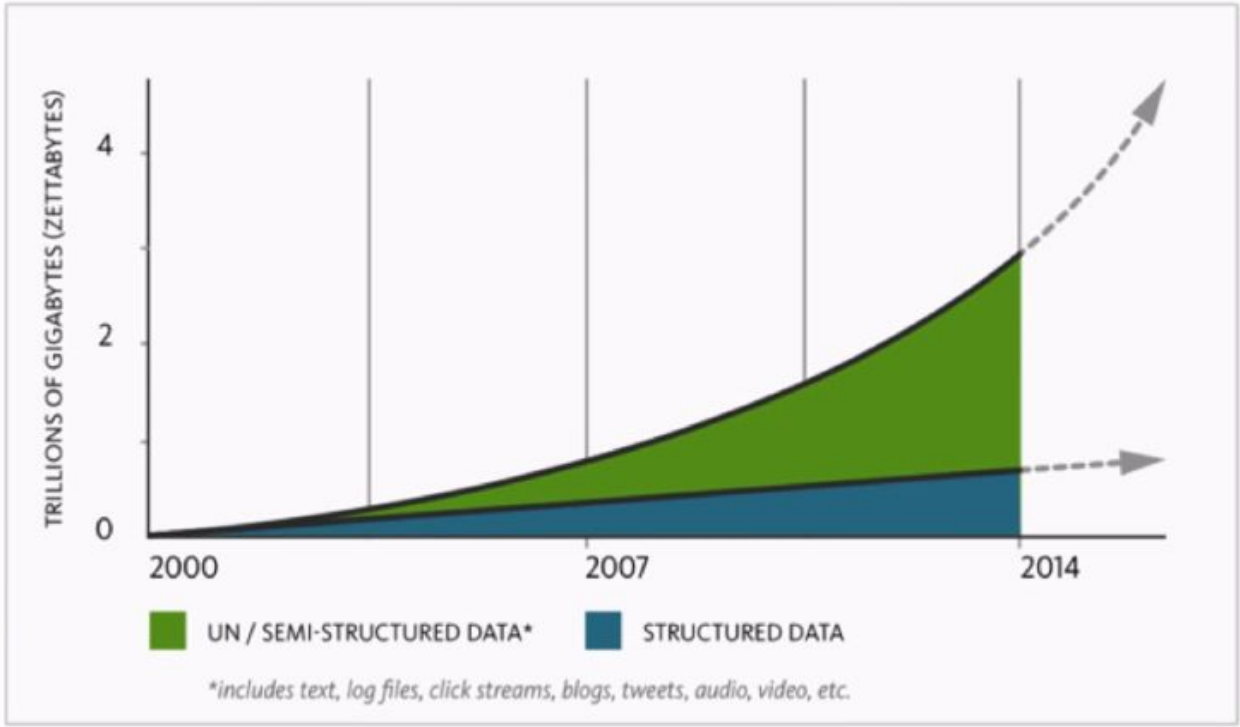
Data where the
number, type, and length
of fields in each record
may vary
and
are not precisely known
until stored

- ✓ Sensor data
- ✓ Customer profiles
- ✓ Blog posts
- ✓ Product catalogs
- ✓ Online content
- ✓ File uploads
- ✓ Social media
- ✓ System logs
- ✓ Personalized news
- ✓ Cloud API data feeds
- ✓ Much, much more …

What data structures are driving all this volume?

User activity
Customer activity
Machine activity
Transactions
Social comments
Media uploads
Etc …

The volume of
unstructured
data is *exploding* …

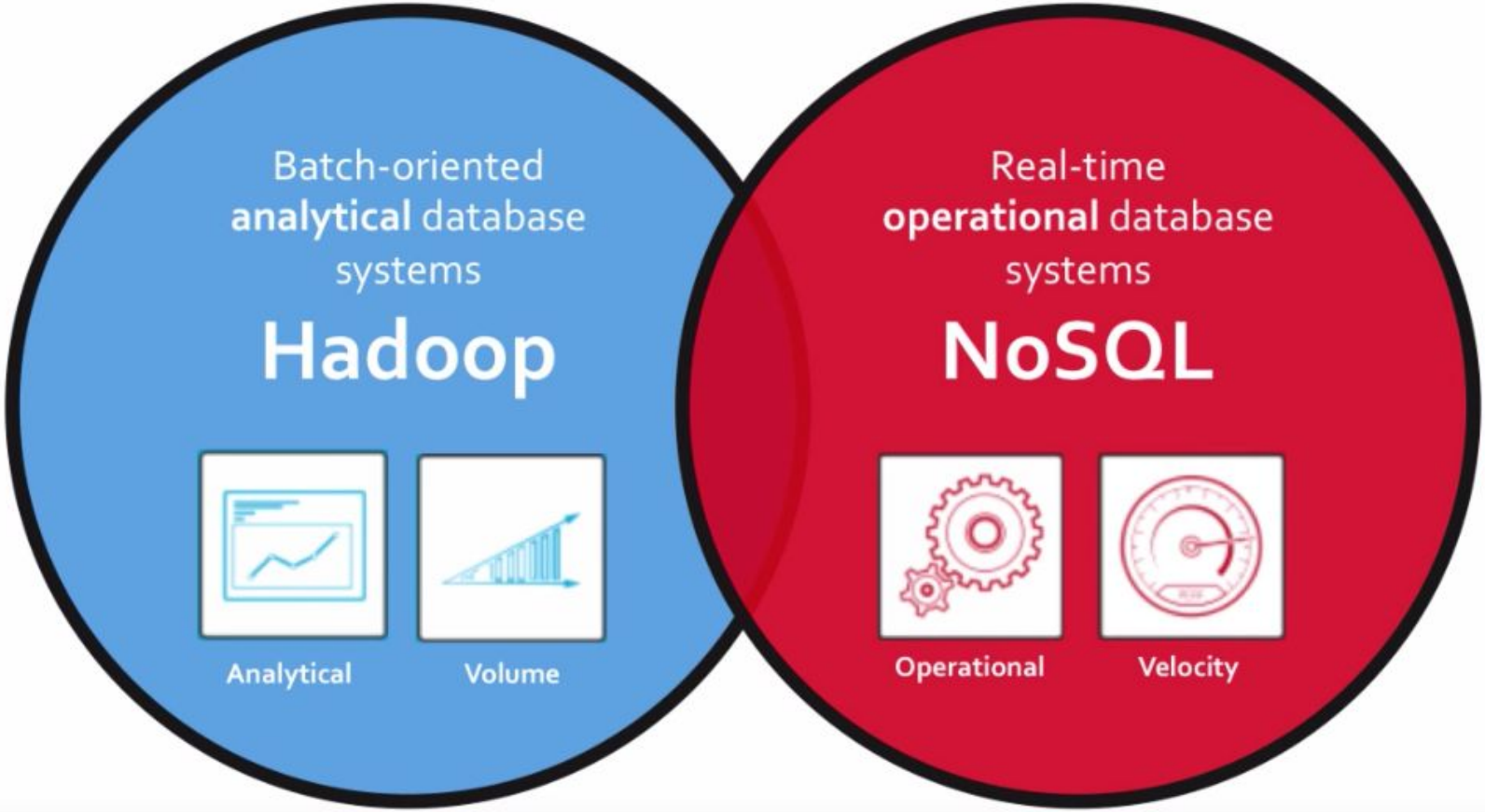


Enterprise data collection volume continues growing 40% to 60% each year
- GigaOM Research, 2014

How do you put all this data to work?

Analytical Use	Operational Use
Batched workloads	Real time intelligence
Vast data aggregations	Focus on data flows and processes
Retrospective analyses	Extremely fast (in-memory) reads
Focus on data pools	Extremely fast (log append) writes
Improve future outcomes	Improve the current outcome

How are analytical and operational uses different?



What we have?

Big Data is the result of increased velocity, variety, and volume

Average adults are spending 7+ hours per day online

Cloud based solutions invite 24 x 7 global scale use

The "internet of things" is opening surging data flows

Most new data flows are unstructured

Analytical databases improve future outcomes

Operational databases improve the current outcome

How does Big Data challenge traditional RDBMS technology?

Did the internet break SQL?

How are traditional RDBMS struggling?

Slow performance

at cloud scale

Millions of concurrent users

Inflexible

for cloud data

Continuously evolving
data models and documents

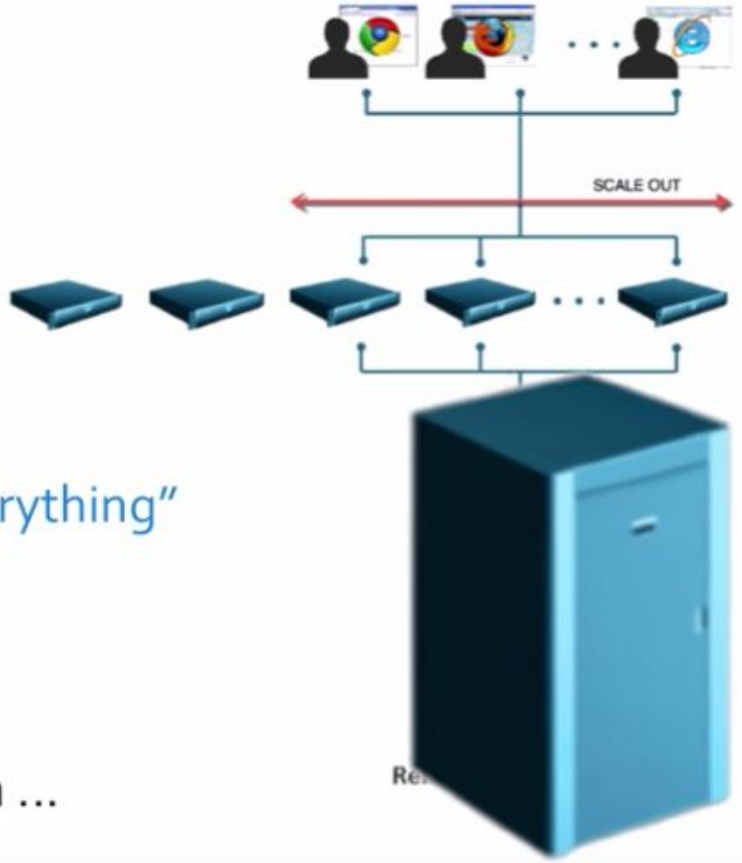
Why does it matter how systems scale?

Application servers "share nothing"
to scale *horizontally/out*

Need more power?
Add another commodity server
And another …

RDBMS servers generally "share everything"
to scale *vertically/up*

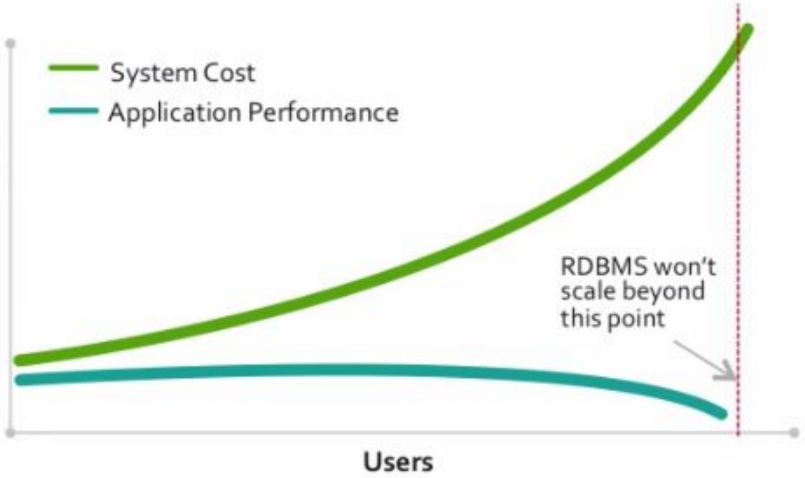
Need more power?
Replace your server. Again, and again …



Why does it matter how systems scale?

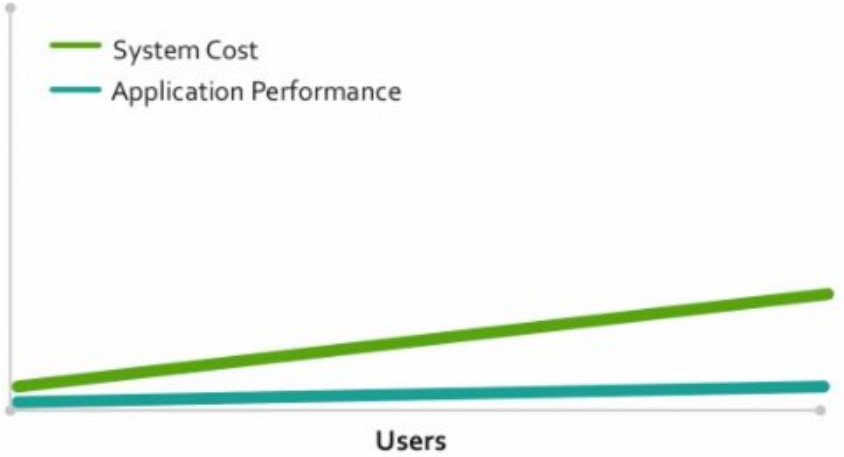
RDBMS Scales Up

Replace your server with something bigger



Cloud applications Scale Out

Just add another box to the rack



But … Enterprise data collection volume continues growing 40% to 60% each year.

- GigaOM Research,

How do the architectures fundamentally differ?

RDBMS approach: **Disk First**

- Virtually all reads/writes must seek on disk …

NoSQL approach: **Memory First**

- All reads/writes can be served immediately from memory …

What's wrong with how RDBMS organize data?

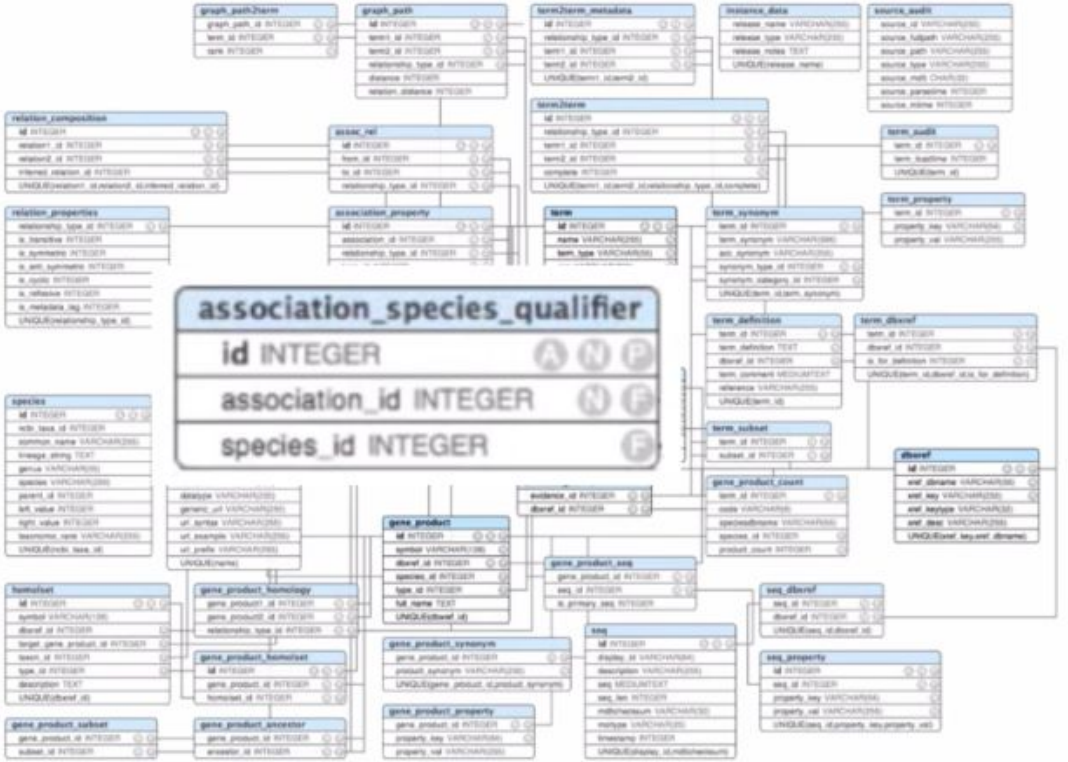
RDBMS technology uses fixed, pre-defined *tables*

- ✓ Invented by E. F. Codd, 1970
- ✓ Commercialized by Oracle, 1979

Each new data type needs planning and integration

- ✓ Normalization
- ✓ New relations
- ✓ Query drafting and optimization

But, applications focus on *objects* … ever-evolving things like *people*, *places*, and *products*



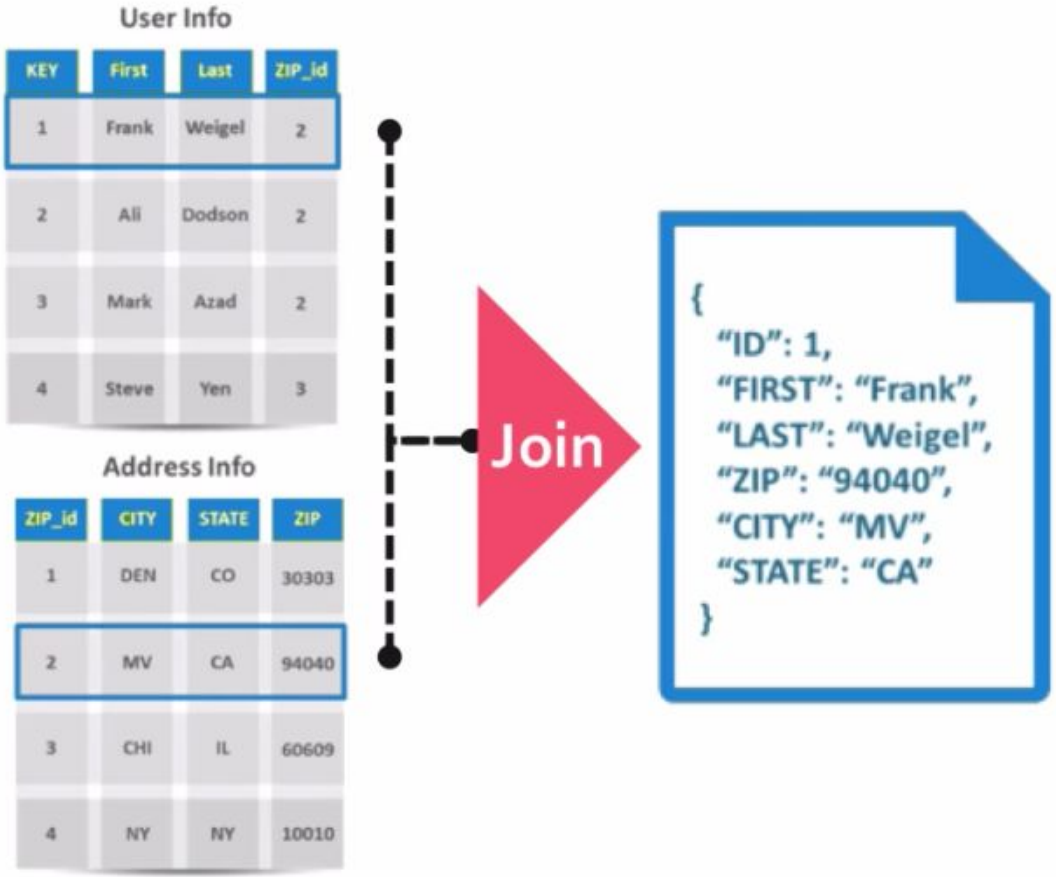
Is there something wrong about rows?

Expensive *disk seeks* and *table joins* required just to assemble logically related data

Complex *object-relational mapping (ORM)* frameworks have evolved in response

But, the *impedance mismatch* between rows and objects leads to complex application code

Expensive to maintain

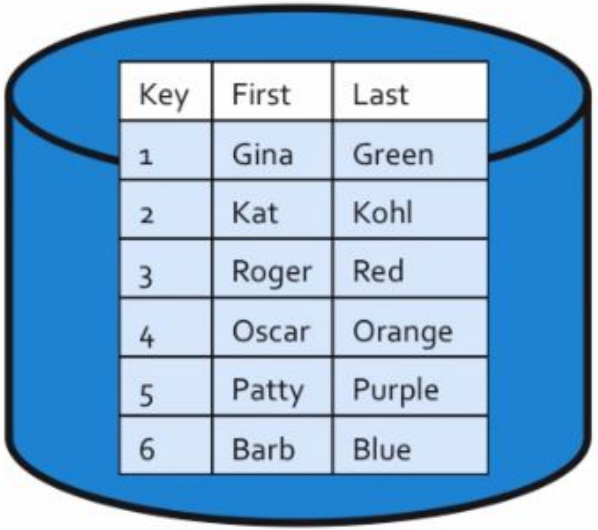


How have RDBMS tried to solve these problems?

Need performance?

Try adding a cache layer

memcached



Key	First	Last
1	Gina	Green
2	Kat	Kohl
3	Roger	Red
4	Oscar	Orange
5	Patty	Purple
6	Barb	Blue

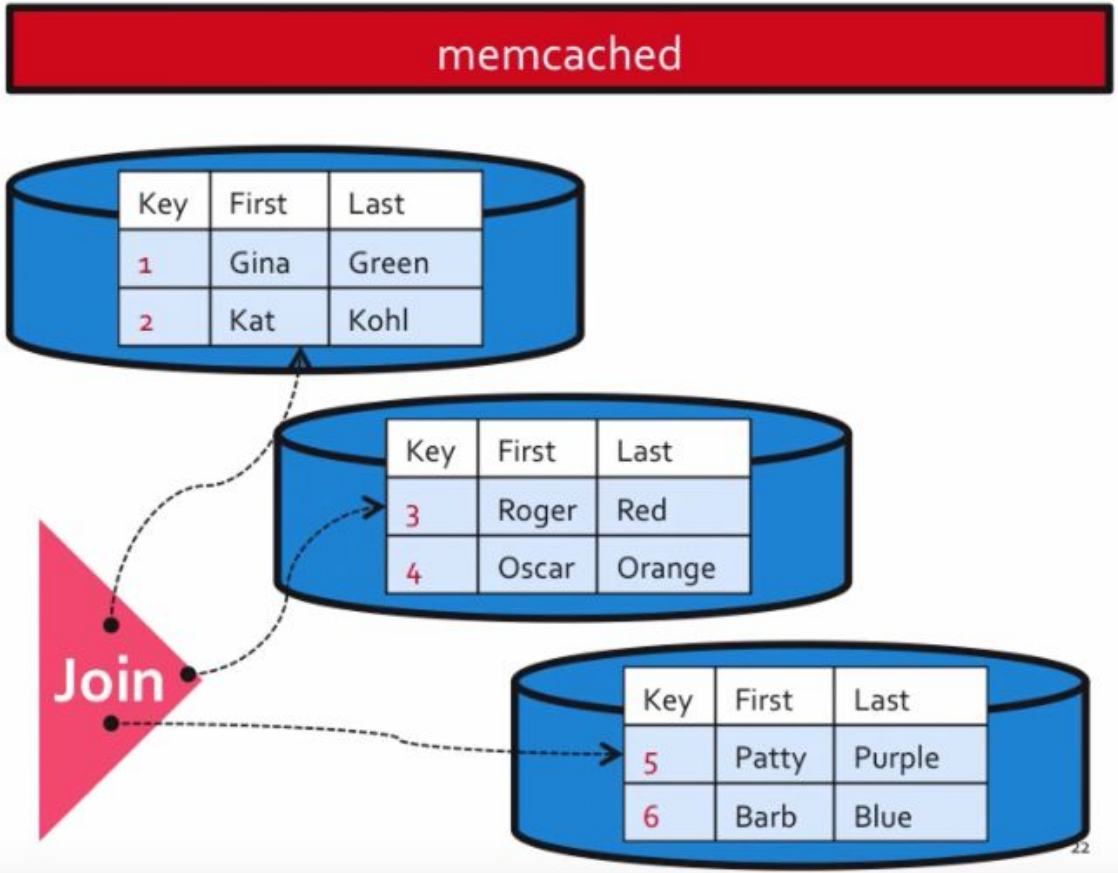
How have RDBMS tried to solve these problems?

Need performance?

Try adding a cache layer

Need scalability?

Try manual table sharding across a cluster



How have RDBMS tried to solve these problems?

Need performance?

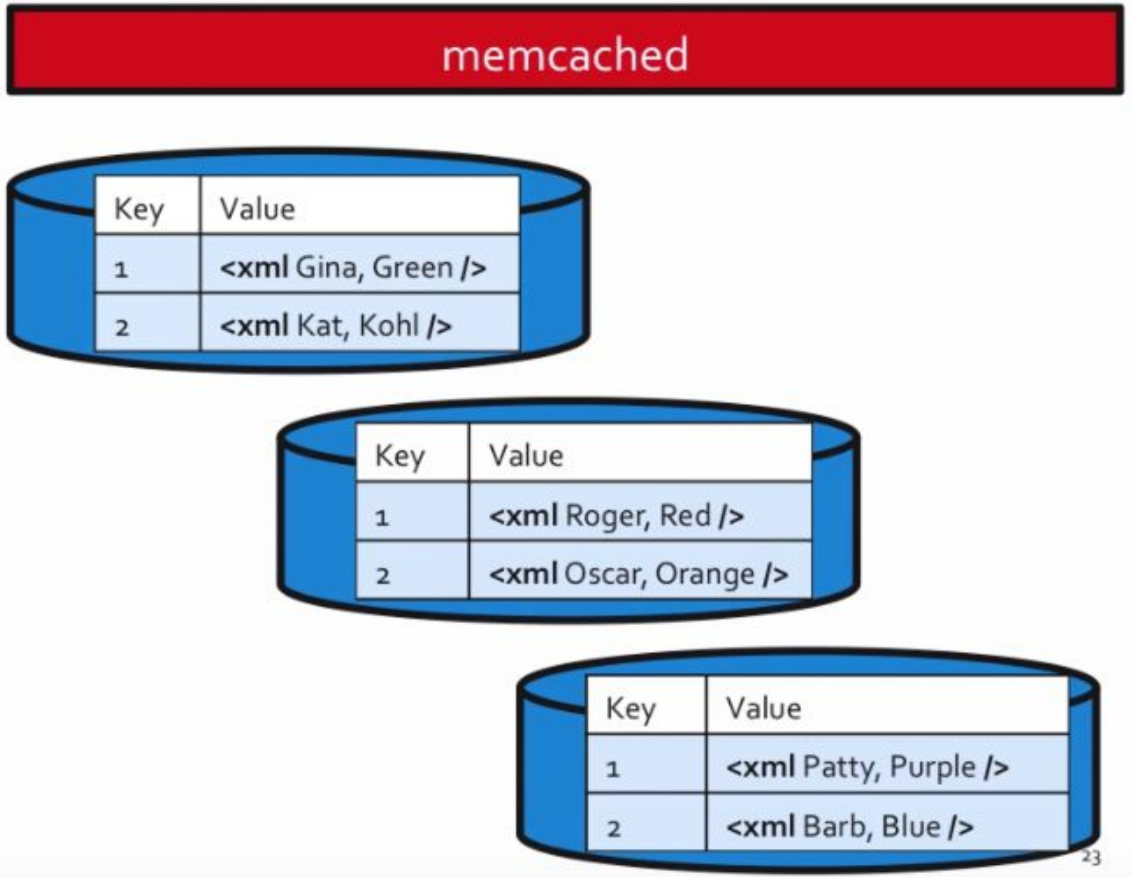
Try adding a cache layer

Need scalability?

Try manual table sharding across a cluster

Need flexibility?

Try storing documents and binaries in a common table



How has NOSQL technology evolved to respond?

Need performance?

Caching built-in and optimized

Memory-first architecture

Need scalability?

Clustering built-in and optimized

Replication and failover fully managed

Need flexibility?

Key-Value/Document storage built-in and optimized

SQL for Documents, Global Secondary Indexing, and MapReduce Views

What we have?

RDBMS systems scale up to bigger hardware, not out

Scale-out architecture has been proven in the application tier

✓ NoSQL takes it to the *persistence* tier

RDBMS architecture is disk-first, for durability

✓ NoSQL is memory-first, for *performance*

RDBMS schema focus on strict tables and rows, to save disk space

✓ NoSQL schema are flexible, to make use and coding flexible

RDBMS try to keep up with caching, table-shards, and BLOBs

✓ NoSQL builds these in, optimized for scalability and speed

What does “NoSQL” mean?

It’s a brave new world out there.

What characterizes NoSQL databases?

Cluster Friendly

Cloud Focused

Flexible

Fast

What are the four broad NOSQL approaches?

Key-Value

key and a simple value

user::1234	**Frank**

Document

key and a structured value (document)

user::1234	{ **name**: 'Frank', **age**: 37, **kids**: ['Sue', 'Ann', 'Bob'] }

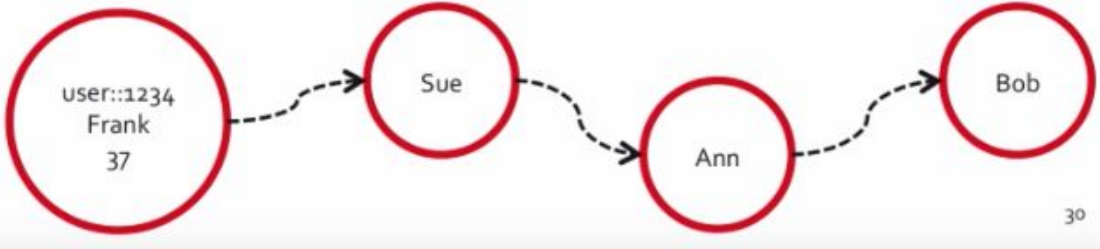
Column Family

key and sets of typed tuples/columns

user::1234	**name**: text	Frank
	age: number	37
	kid: text	Sue
		Ann
		Bob

Graph

linked lists of keyed objects



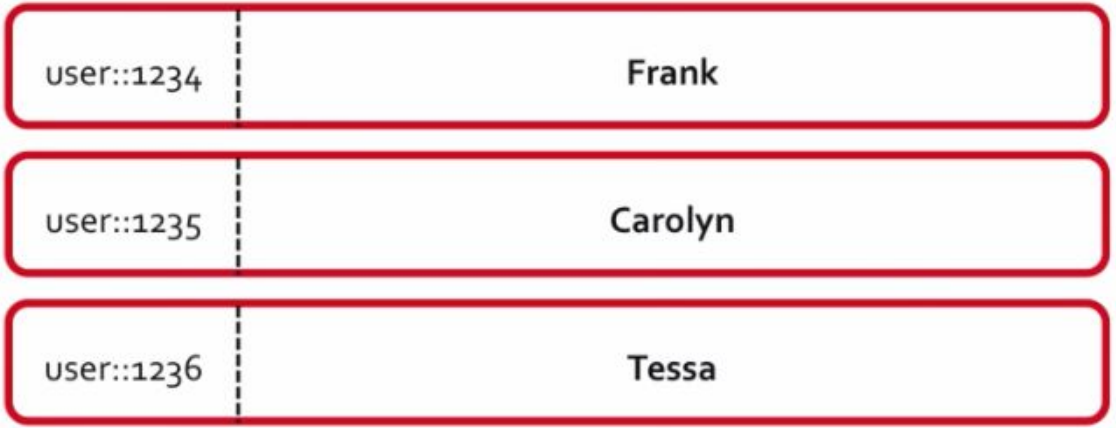
What are the four general NoSQL data models?

Key-Value

key and a simple value

- ✓ Speed
- ✓ Scale
- ✓ Simplicity

Find simple values by key extremely fast



What are the four general NoSQL data models?

Document

key and a structured value (document)

- ✓ Speed
- ✓ Scale
- ✓ Flexibility

Read/write ever-changing data about people, places, and things, at cloud-scale

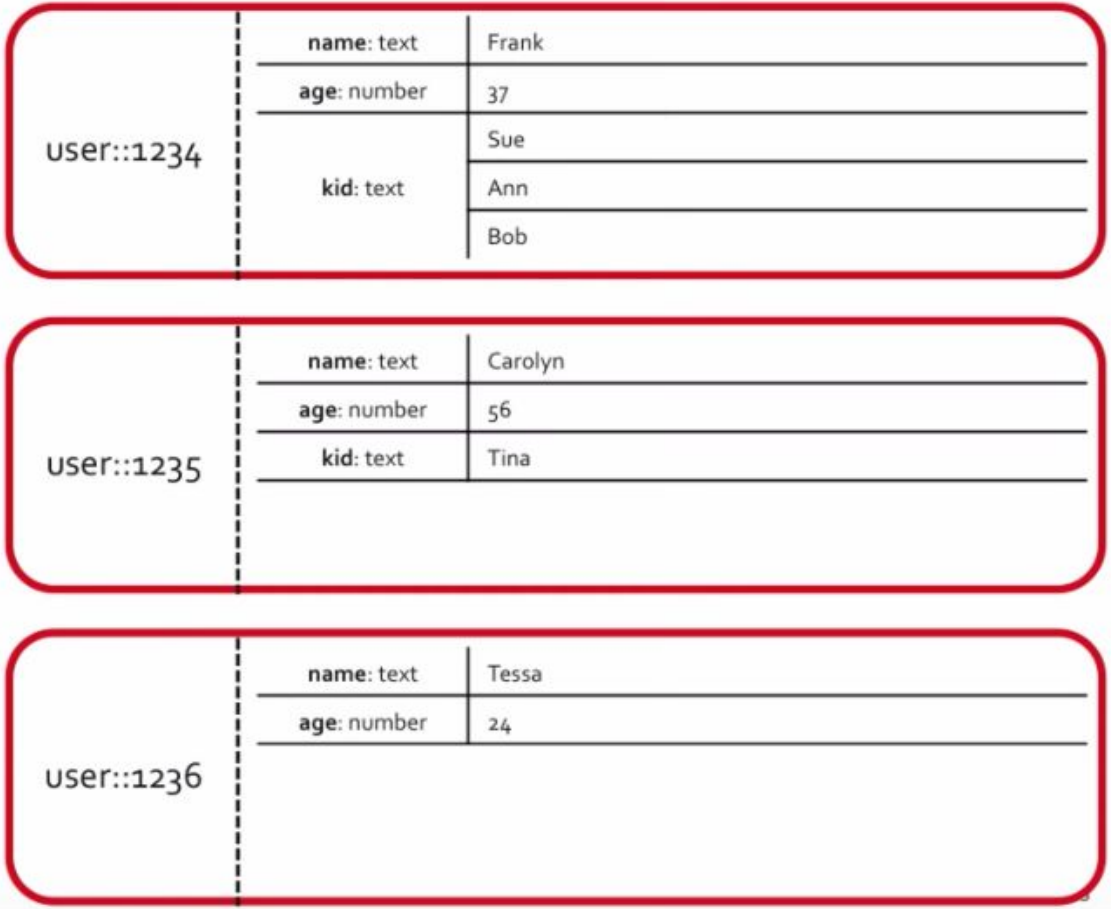
What are the four general NoSQL data models?

Column Family

key and nested sets of typed tuples

- ✓ Scale
- ✓ Ingestion

Write vast volumes of data, with eventually consistent read access



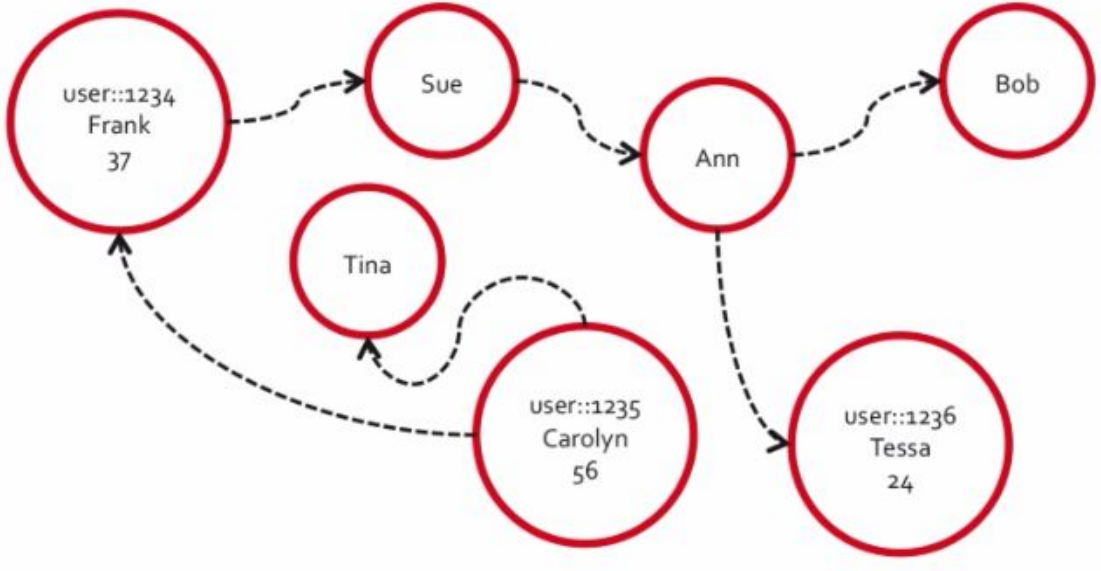
What are the four general NoSQL data models?

Graph

linked lists of keyed objects

- ✓ Relationships

Monitor complex, dynamically networked connections



How can you evaluate all this variety?

CAP Theorem: analyzes limitations inherent in distributed systems

Posited by Eric Brewer

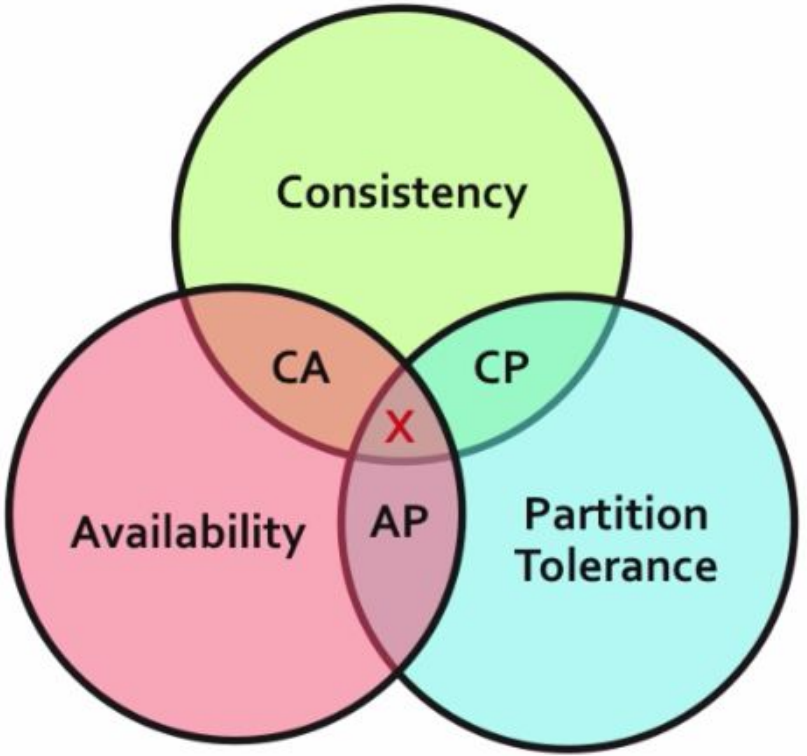
Proved by Nancy Lynch and Seth Gilbert

Consistency – can all nodes see identical data, at all times?

Availability – can all nodes be read from and written to, at all times?

Partition Tolerance – will nodes function normally, even when the cluster breaks?

Pick any <u>two</u> ...

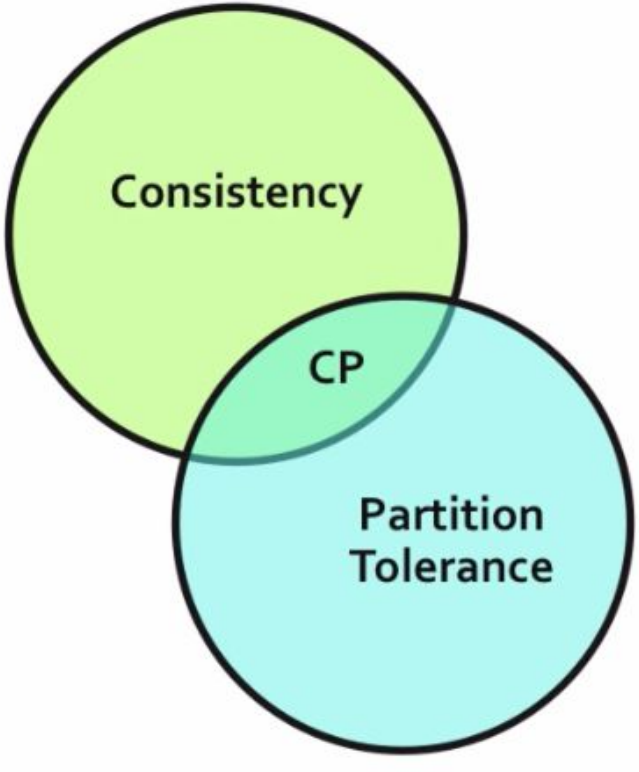


What are the CAP theorem aspects?

CP: Consistency and Partition Tolerance

Immediately consistent data across a horizontally scaled cluster, even with network problems

Couchbase, MongoDB



What are the CAP theorem aspects?

CP: Consistency and Partition Tolerance

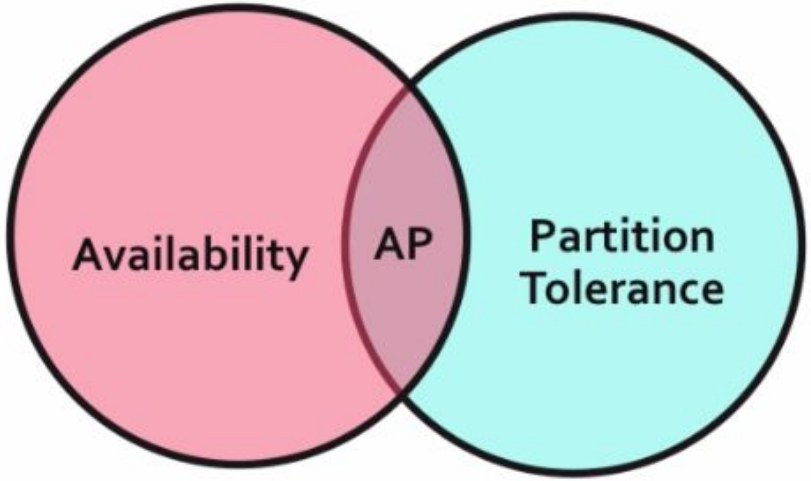
Immediately consistent data across a horizontally scaled cluster, even with network problems

Couchbase, MongoDB

AP: Availability and Partition Tolerance

Always services requests, across multiple data centers, even with network problems, data eventually consistent

Apache HBase or Cassandra, Couchbase (XDCR)



What are the CAP theorem aspects?

CP: Consistency and Partition Tolerance

Immediately consistent data across a horizontally scaled cluster, even with network problems

Couchbase, MongoDB

AP: Availability and Partition Tolerance

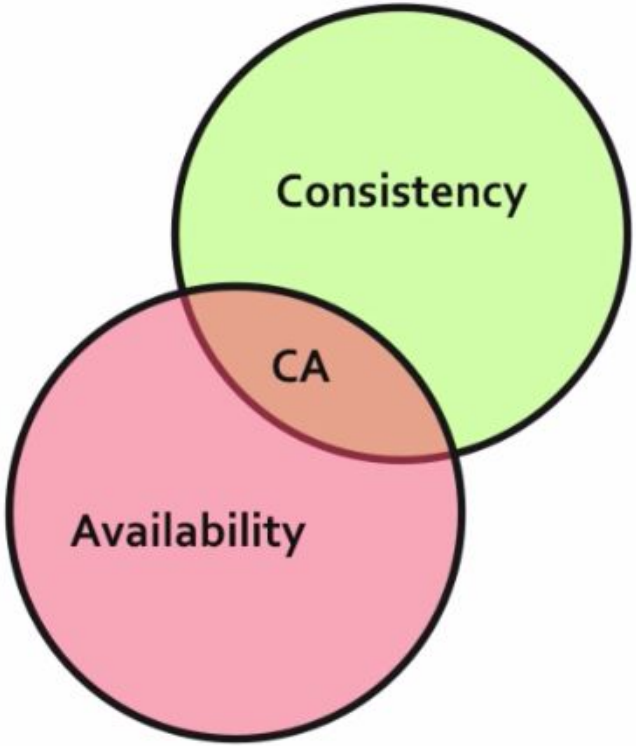
Always services requests, across multiple data centers, even with network problems, data eventually consistent

Apache HBase or Cassandra, Couchbase (XDCR)

CA: Consistency and Availability

Always services requests with immediately consistent data, in a vertically scaled system

MySQL, Oracle, Microsoft SQL Server



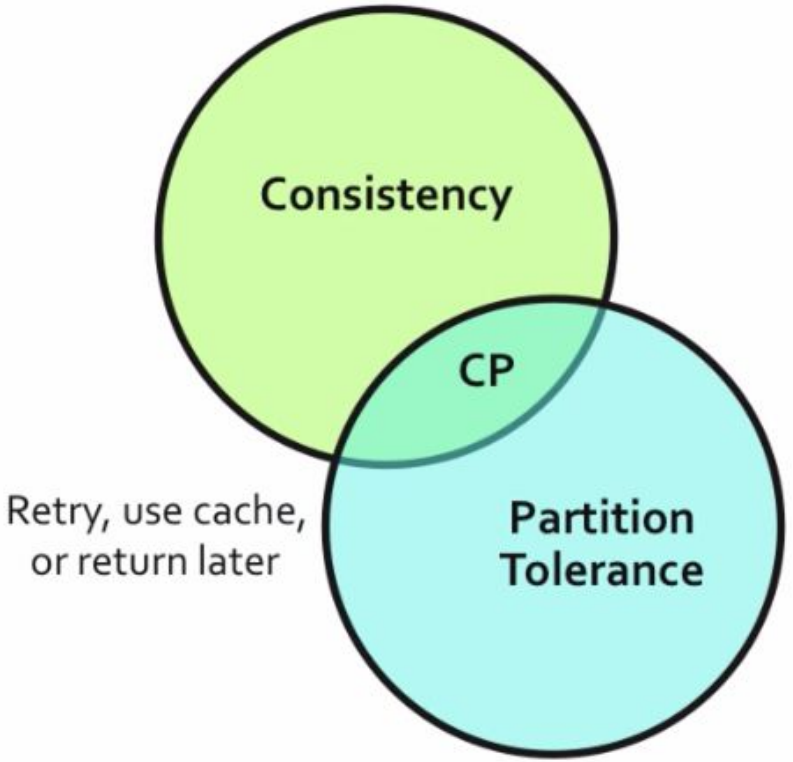
What are the trade offs?

De-emphasize Availability?

Low risk of outage in distributed systems

Retry, use cache, or come back later

Couchbase, MongoDB



What are the trade offs?

De-emphasize Availability?

Low risk of outage in distributed systems

Retry, use cache, or come back later

Couchbase, MongoDB

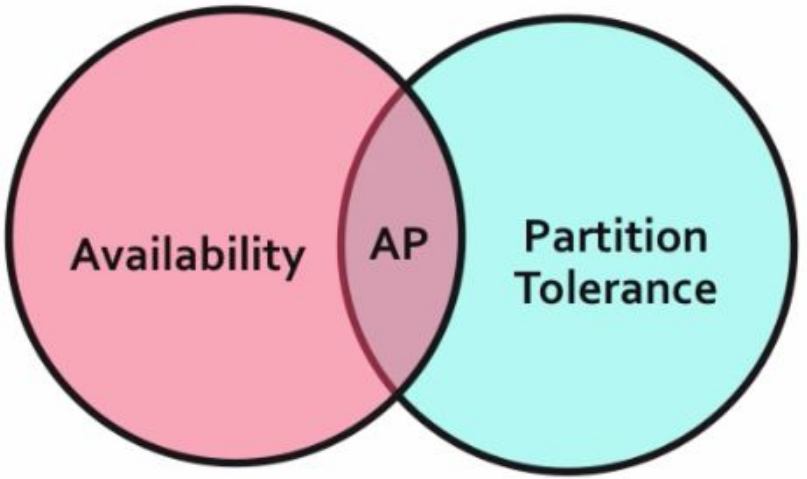
De-emphasize Consistency?

High risk of inconsistency in big data systems

Stale data, or slow reads to wait for multi-node quorum

Apache HBase or Cassandra, Couchbase (XDCR)

Accept stale data
or slow every response



What are the trade offs?

De-emphasize Availability?

Low risk of outage in distributed systems

Retry, use cache, or come back later

Couchbase, MongoDB

De-emphasize Consistency?

High risk of inconsistency in big data systems

Stale data, or slow reads to wait for multi-node quorum

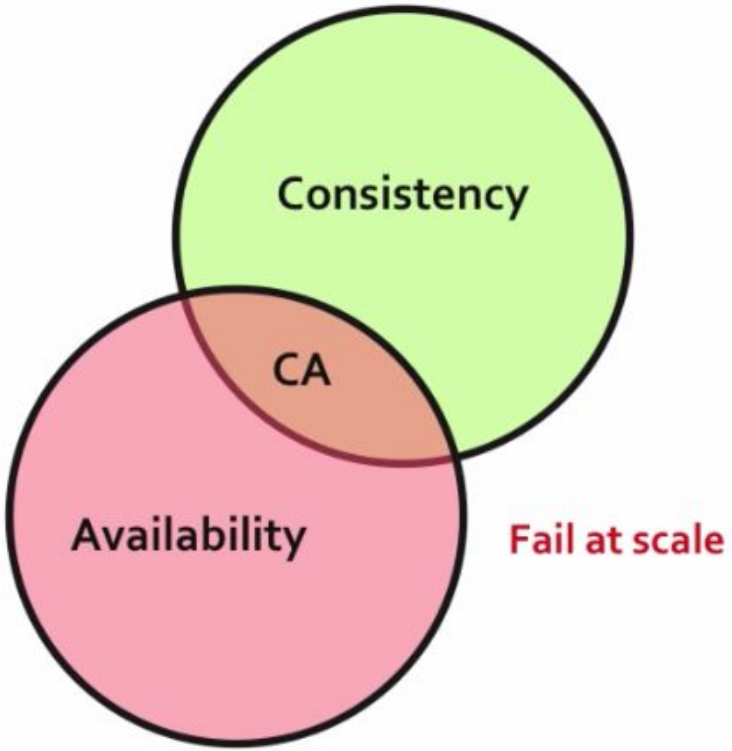
Apache HBase or Cassandra, Couchbase (XDCR)

De-emphasize Partition Tolerance?

High risk of scaling limits unless distributed

Potential total system failure at cloud scale

MySQL, Oracle, Microsoft SQL Server



What we have?

RDBMS are challenged by cloud-scale, clustered architectures

Applications focus on people, places, and things … not rows

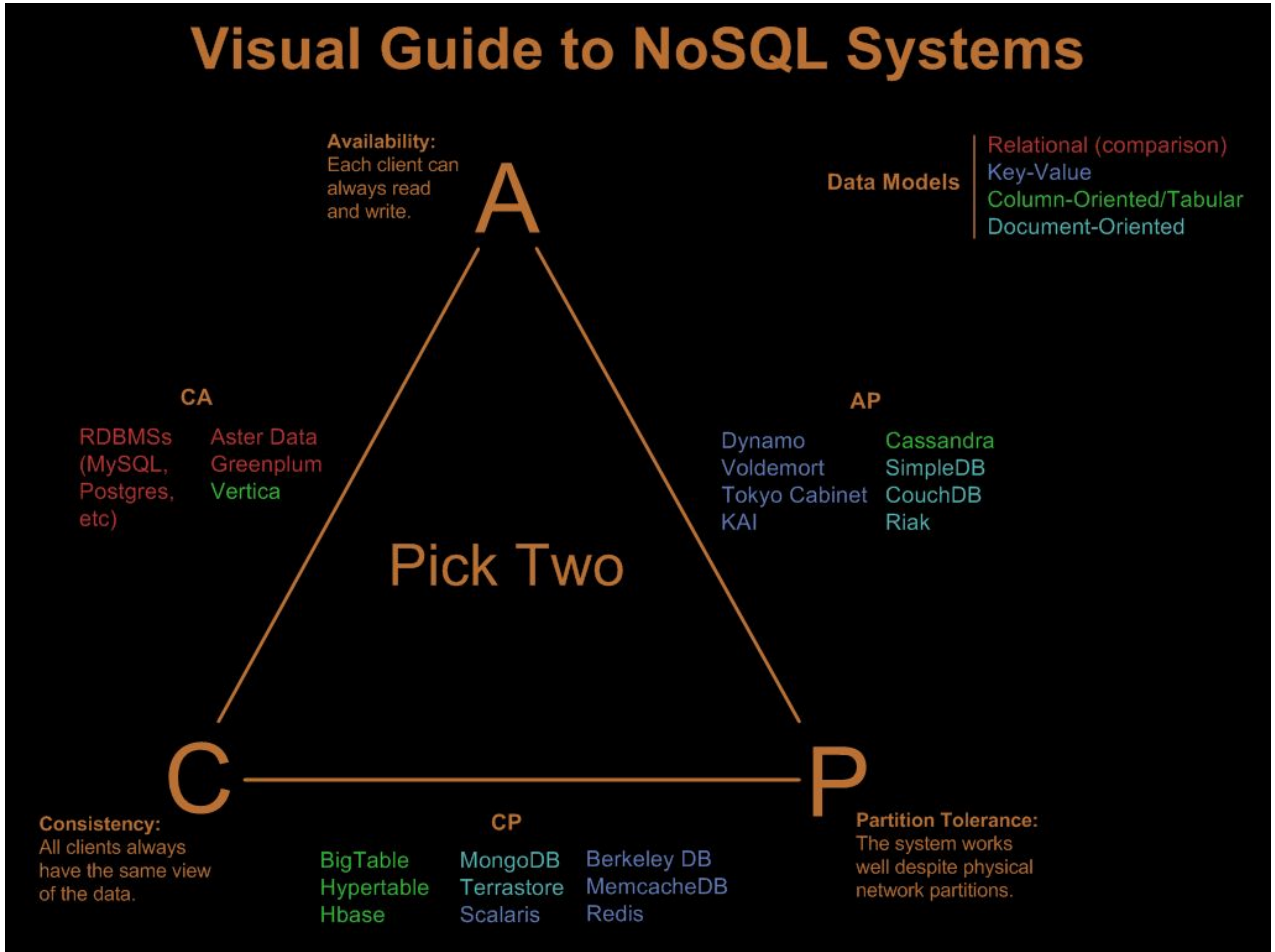
NoSQL databases take four general approaches:

- ✓ Key-Value
- ✓ Document
- ✓ Column Family
- ✓ Graph

Distributed systems are Consistent, Available, Partition tolerant

- ✓ Pick any two …

Magnitude order



How do NoSQL and Relational approaches compare?

Do you need a case of apple? Or, one big orange.

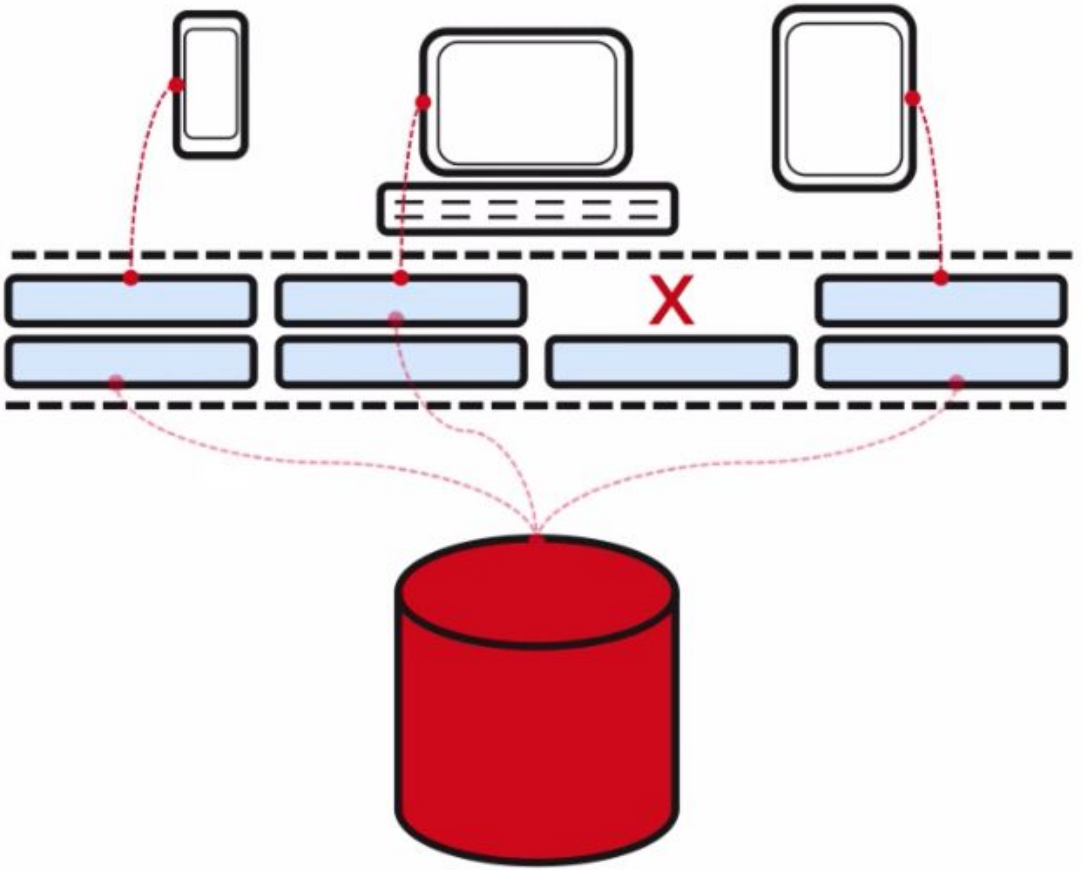
Monolithic vs. distributed systems

Application tier has evolved towards clusters

- ✓ scalable commodity hardware
- ✓ load distribution
- ✓ automatic failover

Persistence tier has remained monolithic

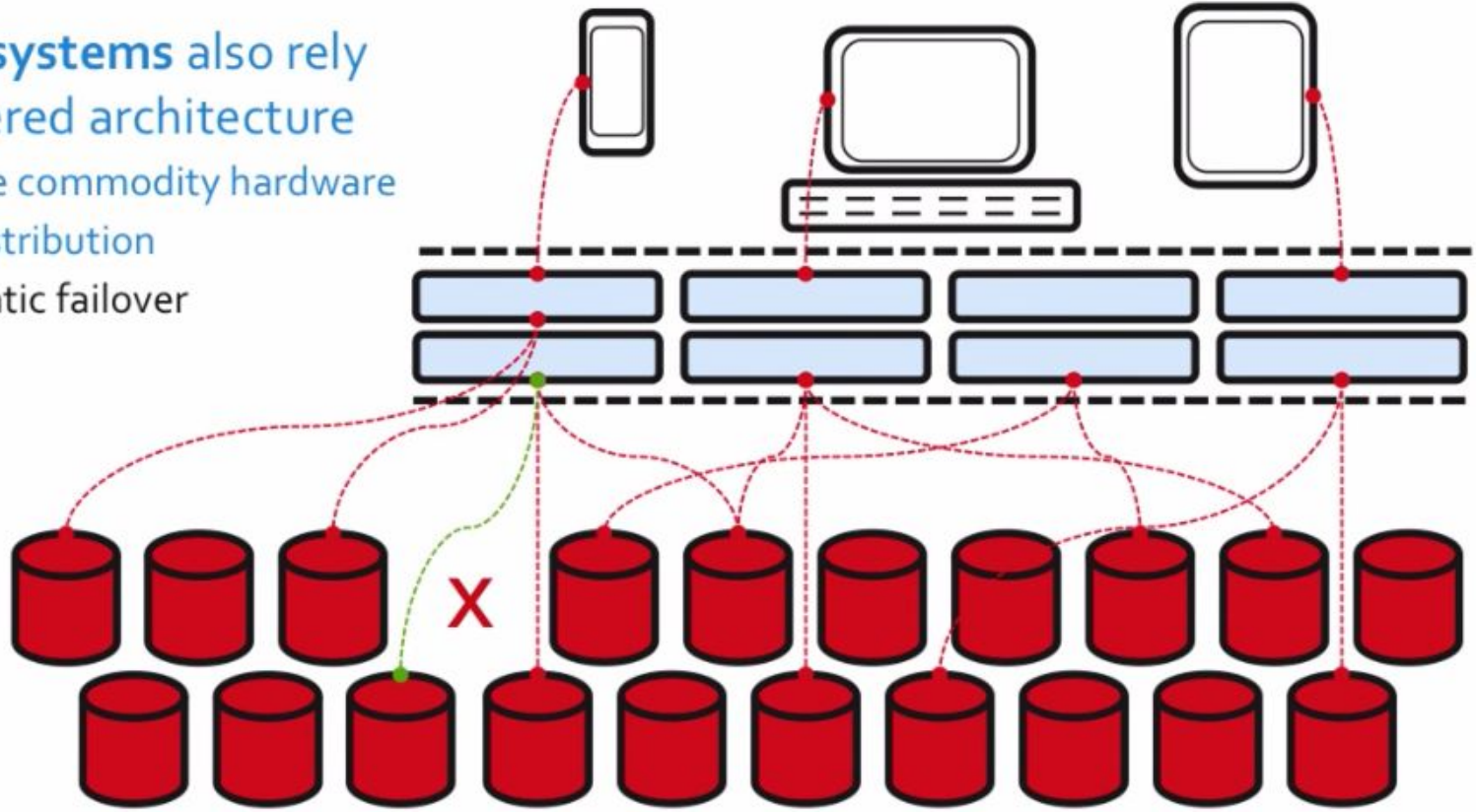
- ✓ premium hardware
- ✓ scale and load limitation
- ✓ single point of failure



Monolithic vs. distributed systems

NoSQL systems also rely on clustered architecture

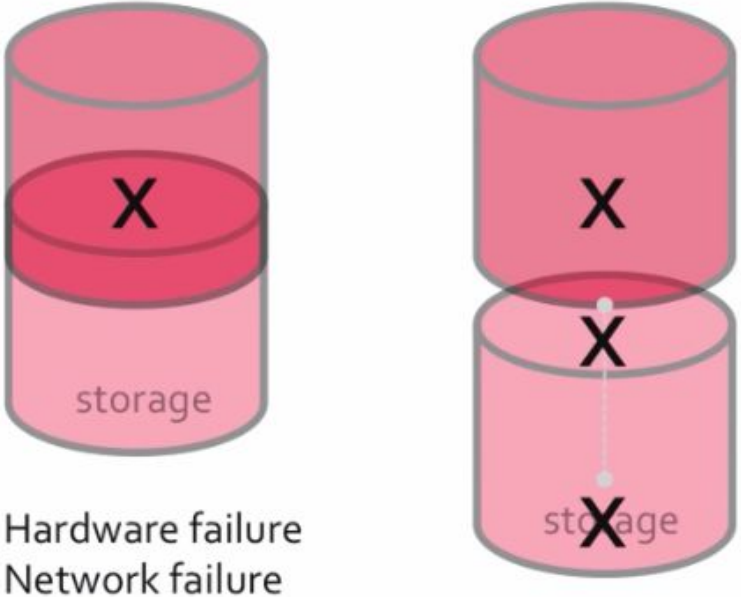
- ✓ scalable commodity hardware
- ✓ load distribution
- ✓ automatic failover



Shared disk vs. shared memory

Monolithic RDBMS architecture

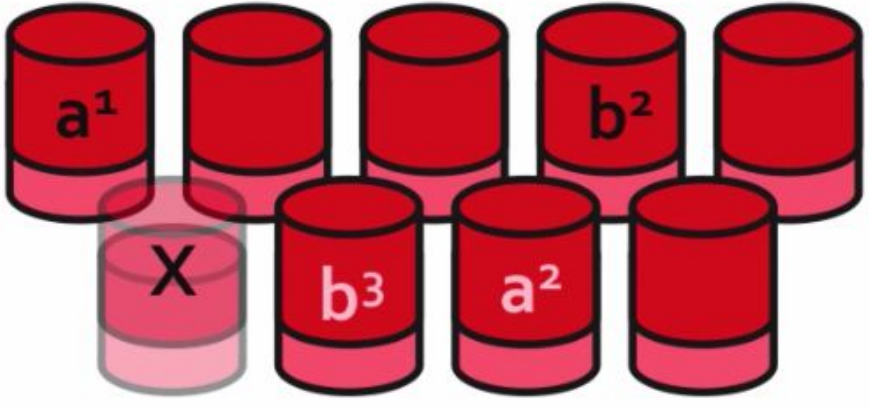
Single points of failure



Hardware failure
Network failure
Software upgrades

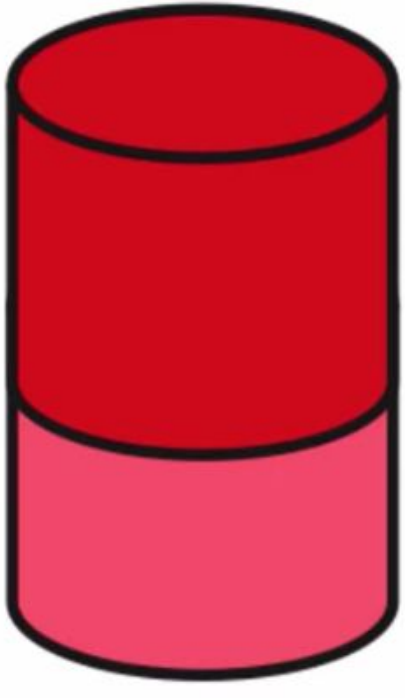
Clustered NoSQL architecture

No single points of failure

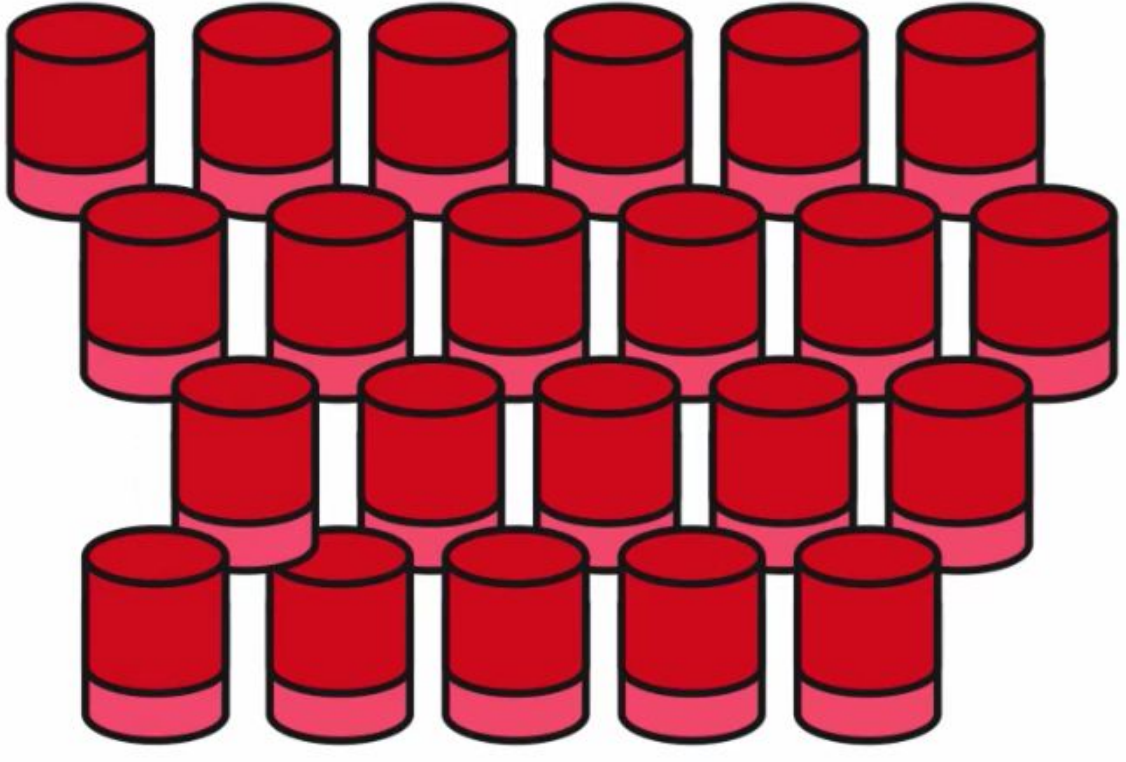


Scaling up vs. scaling out

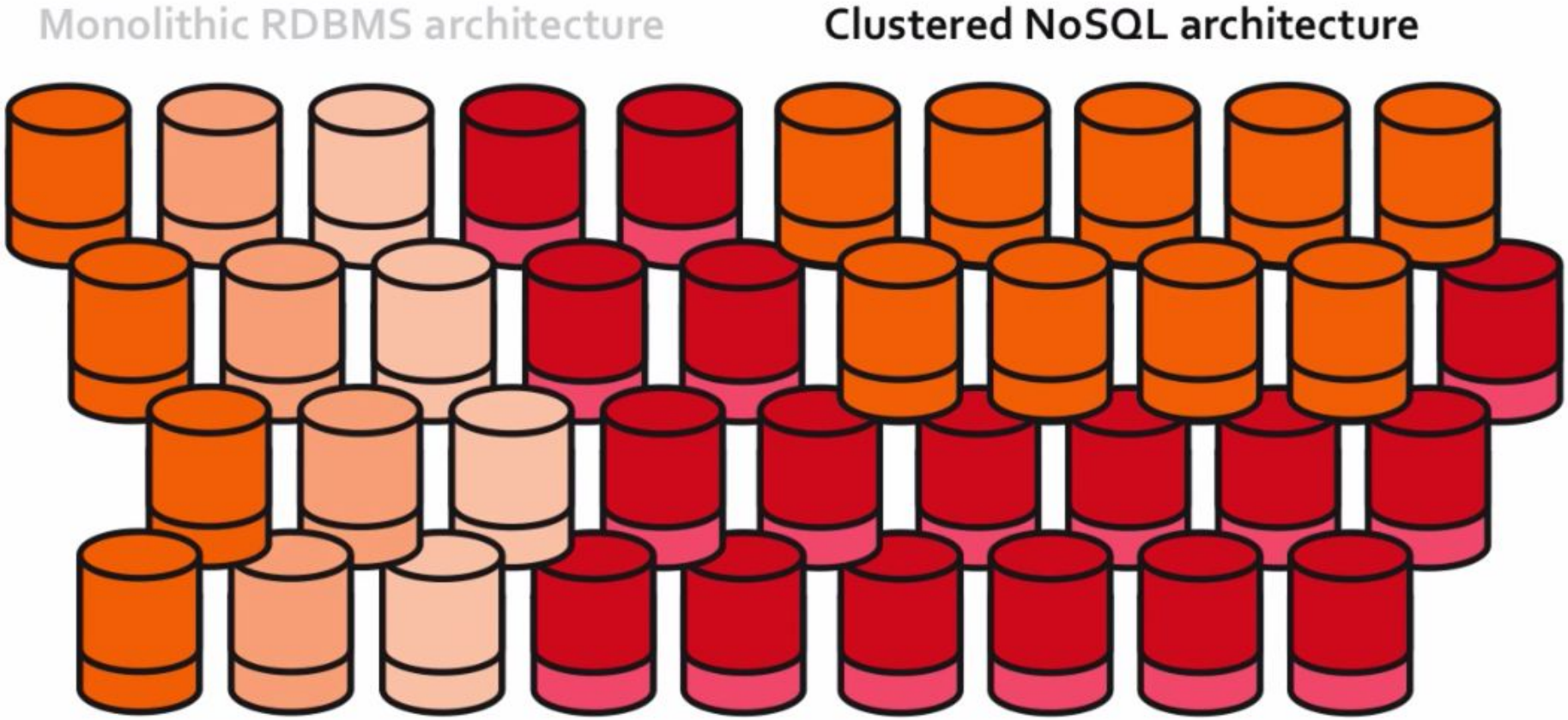
Monolithic RDBMS architecture



Clustered NoSQL architecture



Scaling up vs. scaling out



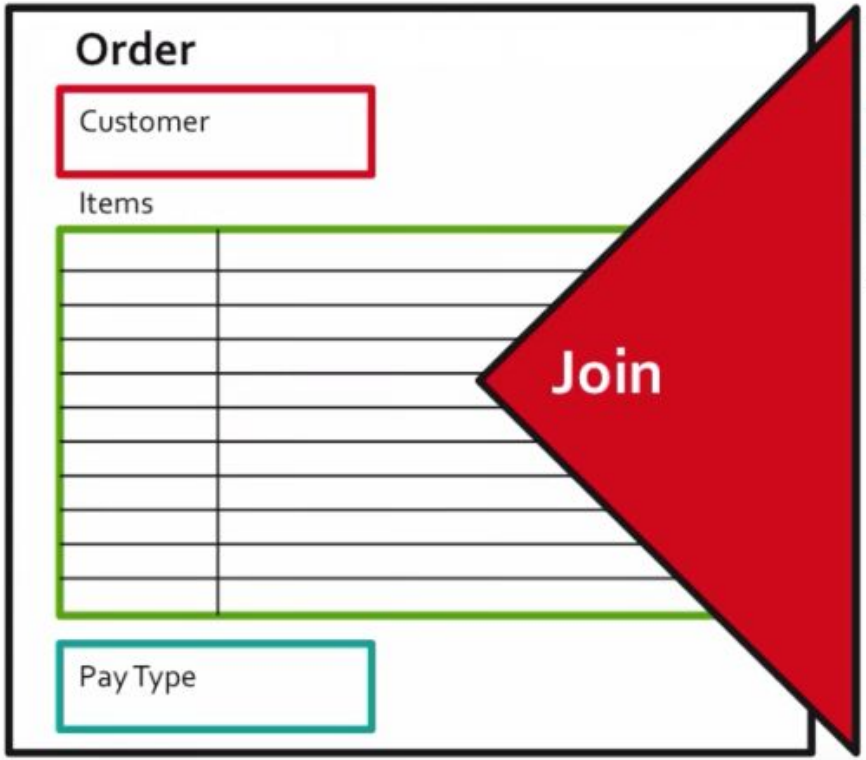
Static vs. flexible data models

Modern applications often aggregate ("mash up") data from different systems

- ✓ RDBMS use rigid, pre-defined schema for data storage
  - ✓ SQL available for access
- ✓ NoSQL store data flexibly, variable by each document
  - ✓ Couchbase N1QL ("Nickel") – SQL superset for JSON documents

Structured vs. unstructured sata

RDBMS isolate related data in highly structured, two-dimensional tables
RAM-intensive joins aggregate flat data into useful information



Made sense … when storage was expensive

CustomerID	Name	AddressID	Email
ABC123	BipCo	abc123	us@bip
XYZ234	Acme	xyz456	al@acme

ProductID	Name	Description
774477	Widget	Fit for all uses.
332299	Gizmo	Perfect for that.
115588	Gadget	Just what I need.

PayTypeID	Name
001	PayPal
002	Visa

Structured vs. unstructured sata

Contextualized data – *information* – is easily represented JSON (or XML, or …)

```
{
 order:123,
 customer: {
   name: "Acme",
   address: "9 W. 2nd",
   email: "al@acme"
 },
 items: [{
   qnty: 2,
   name: "Widget",
   desc: "Fit for all uses."
  },{
   qnty:3,
   name: "Gadget",
   desc: "Just what I need."
  }],
 paytype: {
   name: "PayPal"
  }
}
```

JSON is easily stored and retrieved

Key	Value
order::**123**	{order:123, customer:{name:"Acme", address:"9 W 2nd", email:"al@acme"}, items: [{qnty:2, name:"Widget", desc:"Fit for all uses."}, {qnty:3, name:"Gadget", desc:"Just what I need."}], paytype: {name:"PayPal"}}
order::**124**	{order:124, customer: {name:"BipCo", address:"23 5th", email:"us@bip"}, items: [{qnty:4, name:"Widget", desc:"Fit for all uses."}, {qnty:2, name:"Gadget", desc:"Just what I need."}], paytype: {name:"Visa"}}

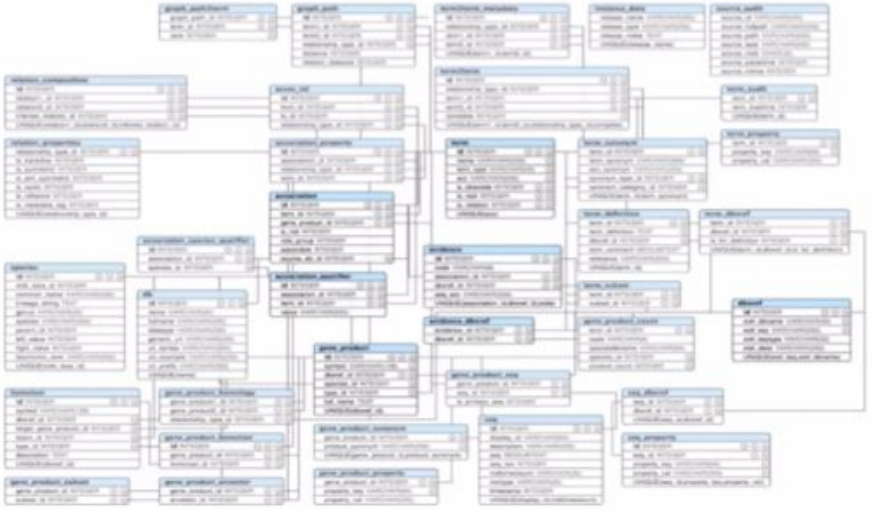
… and, easily mapped to objects, using any of dozens of open source libraries

http://www.json.org

Normalization vs. de-normalization

Normalization - format storage to reduce total data footprint

Classic relational table design



Normalization vs. de-normalization

Normalization - format storage to reduce total data footprint
Classic relational table design

De-normalization - format storage for fast read speed and easy use in code
Document and Key-Value NoSQL systems are optimized for this approach

Key	Value
order::**123**	{order:123, customer:{name:"Acme", address:"9 W 2nd", email:"al@acme"}, items: [{qnty:2, name:"Widget", desc:"Fit for all uses."}, {qnty:3, name:"Gadget", desc:"Just what I need."}], paytype: {name:"PayPal"}}
order::**124**	{order:124, customer: {name:"BipCo", address:"23 5th", email:"us@bip"}, items: [{qnty:4, name:"Widget", desc:"Fit for all uses."}, {qnty:2, name:"Gadget", desc:"Just what I need."}], paytype: {name:"Visa"}}

Key benefits of normalized data

Tables are compact
Ad hoc queries are simplified

Key benefits of de-normalized data

No joins needed
Coding is simplified

What are key strengths of each approach?

Traditional RDBMS	NoSQL
Ad hoc queries and reports	Simplified code
Firmly defined schema	Flexible data models
ACID transactions	Cloud scale consistency
Vendor supported proprietary code	Vendor and community supported open source code
Premium hardware lowers risk of *individual* failure	Clustered commodity hardware removes risk of *system-wide* failure

What are key strengths of each approach?

No ad hoc queries and reports?

Most NoSQL systems fail to provide support for ad-hoc queries

Data manipulation ends up baked into client-side code using system-specific APIs

Couchbase 4.0

N1QL – SQL for JSON documents

- ✓ SELECT, INSERT, UPDATE, DELETE
- ✓ JOIN, WHERE
- ✓ CREATE INDEX, DROP INDEX
- ✓ MIN, MAX, COUNT
- ✓ GROUP BY, HAVING
- ✓ UNION, INTERSECT, EXCEPT

All for JSON documents
ODBC/JDBC drivers available

What we have?

Application tier architecture has evolved to rely on clusters

- ✓ Persistence tier architecture often remains monolithic
- ✓ NoSQL brings cluster architecture to the persistence tier

RDBMS joins are expensive, like the ORM code managing them

- ✓ De-normalized data simplifies development and persistence

Shared disk architectures create single points of failure

- ✓ Shared nothing architectures enable no single point of failure

Scale-out clusters enable hardware diversity and cost control

Traditional RDBMS remains useful for its core use cases

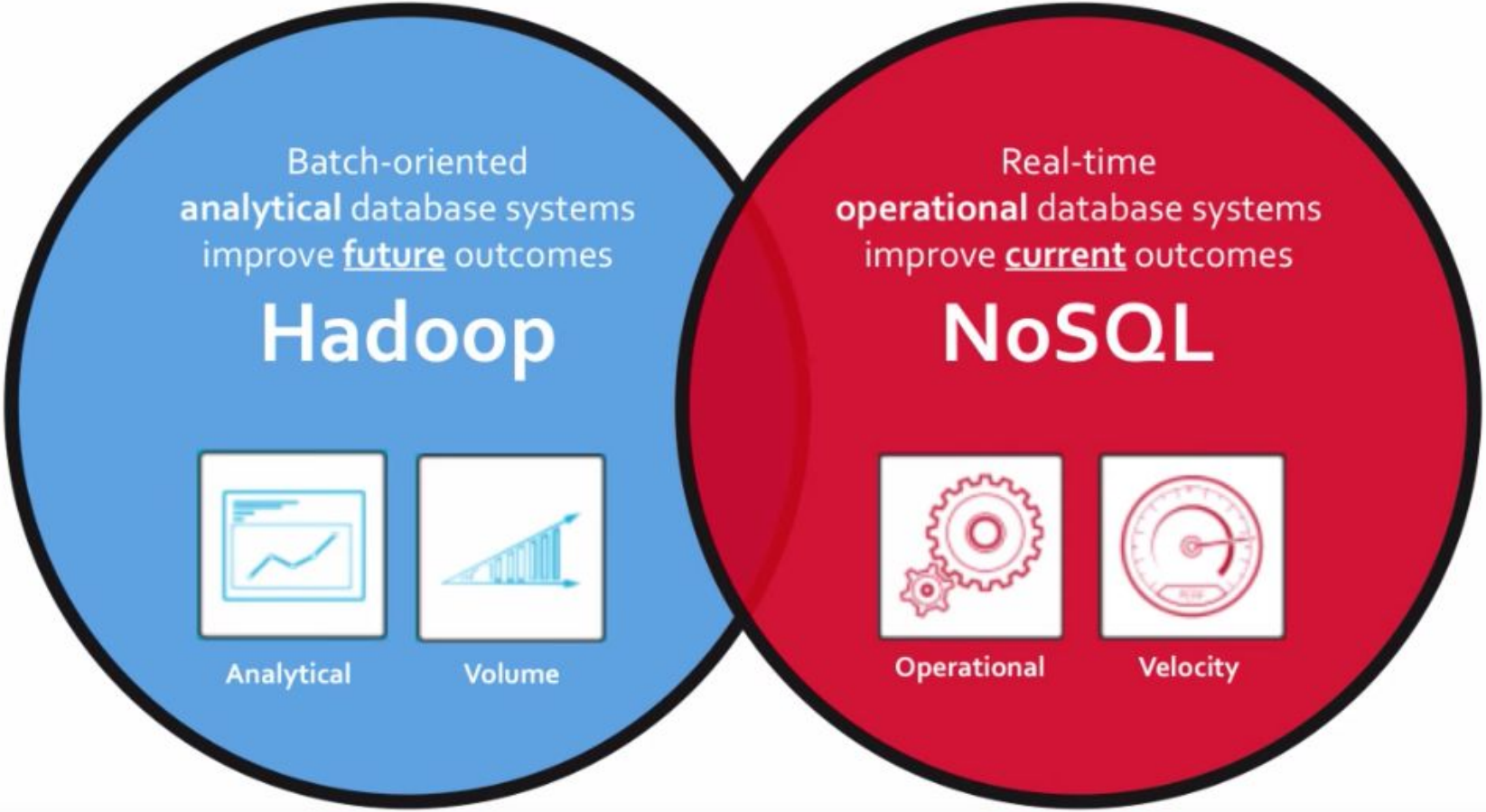
- ✓ NoSQL delivers a compelling alternative for many others

Couchbase 4.0 supports JSON document queries with N1QL

What use cases benefit from Operational NoSQL?

Need speed? Flexibility? Availability? Scale?

How are analytical and operational uses different?



How do enterprises use Operational NoSQL?

Profile Management

Personalization

360 Degree Customer View

Internet of Things

Mobile Applications

Content Management

Catalog

Real Time Big Data

Digital Communication

Fraud Detection

Profile Management

Objective

Drive customer satisfaction and revenue by delighting customers with highly responsive experiences, based on up-to-date profile data

Business requirements

Serve profile data rapidly at the moment of interaction

Accommodate fast changing data types and attributes

Support growing customer base – millions to billions

Technical requirements

Low latency

Data model flexibility

Fast, easy, affordable scalability

The Solution

- ✓ **Integrated cache** – provides fast performance, highly responsive interaction
- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Push-button scalability** on commodity hardware – fast, easy and inexpensive

Personalization

Business requirements

Manage large volumes of fast changing data and new data types, at high speed

Support many concurrent users

Zero downtime

Technical requirements

Data model flexibility

High throughput and low latency

Scalability

24x365 availability

Objective

Enhance your customers' experience and drive engagement by delivering the right offer, content, or recommendation at the "moment of truth"

The Solution

- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Integrated cache** – provides fast performance, high throughput
- ✓ **Push-button scalability** on commodity hardware – fast, easy and inexpensive
- ✓ **Automatic replication** and **built-in bidirectional XDCR** – ensures high availability

Customer 360 Degree View

Business requirements

Aggregate multiple data types from disparate sources

Interact with analytics tools to capture insights

Rapidly access data for highly responsive experiences

Technical requirements

Data model flexibility

Integration with analytics tools like Hadoop

Low latency

Objective

Increase cross-sell and up-sell opportunities by having a single view of your customer

The Solution

- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Real time Hadoop integration via in-memory streaming** – easily export/import data for customer analytics
- ✓ **Integrated cache** – enables fast performance

Internet of Things

Business requirements

Manage massive datasets (e.g. billions of data points)
Interact with diverse, occasionally-connected devices
Capture new and evolving data types at high speed

Technical requirements

Scale to millions of devices, billions of data points
High throughput
Synchronize data between device and cloud
Data model flexibility

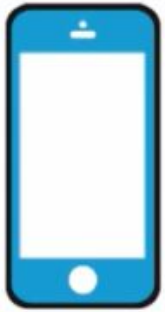
Objective

Deliver new products and services, create new revenue opportunities, and drive agility by connecting with and harnessing data from millions of devices

The Solution

- ✓ **Push-button scalability** – easily scales to support massive data volumes
- ✓ **Integrated cache** – enables high throughput
- ✓ **Embedded JSON database with automated sync –** supports connected and unconnected devices
- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly

Mobile Applications



Business requirements

Read and write data online and offline

Fast time to market - rapidly develop/enhance apps

Support multiple device types and platforms

Technical requirements

Store and access data on the device

Securely sync data between device and cloud

Scalability

Objective

Delight users with great mobile experiences by delivering applications that always work and are always fast, with or without a network connection

The Solution

- ✓ **Couchbase Lite** embedded NoSQL database – ensures data is always accessible, with or without a network connection
- ✓ **Sync Gateway** secure bidirectional data synchronization – simplifies development, minimizes coding required
- ✓ **Scalability** – fast, easy and inexpensive to scale to any size

Content Management

Business requirements

Store and manage multiple data types and attributes – structured, semi- and un-structured

Enable fast access to data

Support large volumes of data and metadata

Technical requirements

Data model flexibility

Low latency

Scalability

Objective

Engage, educate, and inform customers, employees, and other users by providing fast, easy access to disparate types of content

The Solution

- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Integrated cache** – provides fast performance, high throughput
- ✓ **Push-button scalability** on commodity hardware – fast, easy and inexpensive

Catalog

Business requirements

Store large volume of different data types/attributes: SKUs, part numbers, descriptions, metadata, etc.

Manage numerous, rapid updates

Deliver fast response for great customer experiences

Technical requirements

Flexible data model

High read/write throughput

Low latency

Scalability to support large data volume

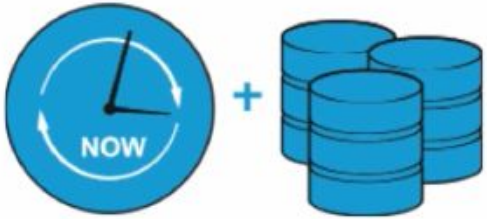
Objective

Reduce inventory, increase cross-sell, and facilitate regulatory compliance, via an easy to maintain and update repository of product and other data

The Solution

- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Integrated cache** – enables high throughput
- ✓ **Push-button scalability** – easily scales to support massive data volumes

Real Time Big Data



Business requirements

Manage massive data volumes at high speed

Store and manage numerous and changing data types

Export/import data to/from analytics platforms

Technical requirements

Scalability and throughput

Data model flexibility

Integrate with Hadoop

Objective

Drive revenue, customer satisfaction, and operational efficiency by leveraging insights from big data analytics in real time

The Solution

- ✓ **Push-button scalability** – fast, easy and inexpensive to scale to any size
- ✓ **Integrated cache** – enables fast performance and high throughput
- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Real time Hadoop integration via in-memory streaming** – easily export data and import analytics results

Digital Communication

Business requirements

Support massive datasets, possibly billions of messages

Real time speed and performance

Zero downtime

Technical requirements

High scalability

Low latency and high throughput

24X365 availability

Objective

Enable customers, employees, partners, and/or other users to interact in real-time, online conversations

The Solution

- ✓ **Integrated cache** – enables low latency performance and high throughput
- ✓ **Push-button scalability** – easily scales to support massive data volumes
- ✓ **Automatic replication** and **built-in bidirectional XDCR** – ensures high availability

Fraud Detection

Business requirements

Update frequently changing data and data types - customer data ,account data, detection rules

Provide real time responsiveness

Process very high volume of interactions

Technical requirements

Data model flexibility

Low latency

High throughput

Objective

Increase profitability, reduce risk, and comply with regulations in real-time by analyzing user, transaction, and contextual data

The Solution

- ✓ **Flexible JSON data model** – easily adapts new data types and attributes on the fly
- ✓ **Integrated cache** – provides real time responsiveness and high throughput processing
- ✓ **Push-button scalability** – fast and easy to meet growth requirements

What we have?

Analytical workloads improve future outcomes

Operational workloads improve current outcomes

Numerous use cases benefit from Operational NoSQL

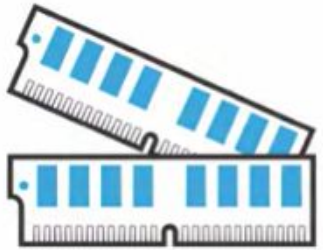
- ✓Profile Management
- ✓Personalization
- ✓360° Customer View
- ✓Internet of Things
- ✓Mobile Application
- ✓Content Management
- ✓Catalog
- ✓Real Time Big Data
- ✓Digital Communication
- ✓Fraud Detection

How does Couchbase meet the needs of modern applications?

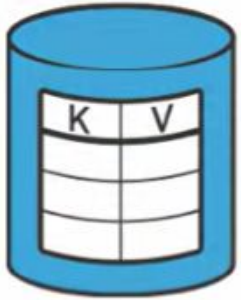
Speed. Flexibility. Availability. Scale.

Couchbase provides complete operational data management

Multi-purpose capabilities support a broad range of apps and use cases



High availability cache



Key-value store

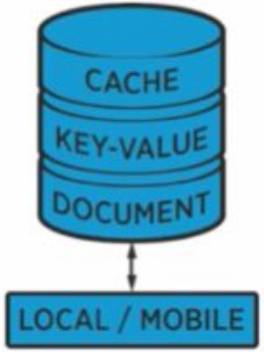
Document database

Embedded database

Sync management

Enterprises often start with cache
then broaden usage to other apps and use cases

Why do enterprises choose Couchbase?



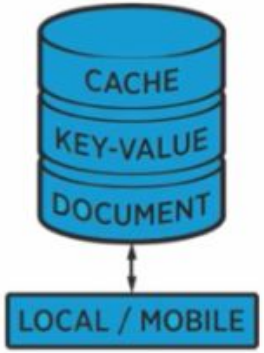
Multi-Purpose

Performance and Scalability leader

Always-On availability

Simple, Flexible Administration

Multi-purpose database support many uses



Tunable Memory

- Immediately consistent data across the cluster
- Tune memory based on requirements

Flexible, index-able SQL query-able JSON

- Represent, store, and index data flexibly using JSON
- N1QL (SQL for JSON), and MapReduce Views

Couchbase Lite on mobile devices

- Lightweight mobile database for always available apps
- Sync Gateway seamlessly syncs data with Couchbase Server

Couchbase leads in performance and scalability

Auto Sharding

- Cluster-wide data distribution is automatic, yet configurable
- Scale out happens naturally as administrators add new nodes

RAM to RAM Replication

- Market's only memory-to-memory database replication across both clusters and geos
- Provides both disaster recovery and data locality across geos

Single Node Type with Tunable Services

- Hugely simplifies cluster management
- Easy to scale by adding nodes and adjusting service resources

Couchbase delivers always-on availability

High Availability

- In-memory replication with manual or automatic fail over
- Rack-zone awareness to minimize data unavailability

Disaster Recovery

- Memory-to-memory cross cluster replication across data centers or geos
- Choose active-active or active-passive XDCR topologies

Backup & Restore

- Full or incremental backup with online restore
- Delta node catch-ups for faster recovery after failures

Simplified administration for exceptional ease of use

Online upgrades and operations

- Upgrade hardware or software without taking database offline
- Index, compact, rebalance, backup, and restore, all while database stays online

Built-in enterprise class admin console

- Perform all administrative tasks with the click of a button
- Monitor system status visually at cluster, database, or server level, with deep-grain metrics

Restful APIs

- All admin operations available via console, command line, or REST API
- Easily integrate third party monitoring tools using REST

How do we deliver all this?

Open Source Code

Source code for every single Couchbase feature is freely available on Github

https://github.com/couchbase

Community Edition

We compile and provide packaged binaries with the core features for the community to use

http://www.couchbase.com/download

Enterprise Edition

We rigorously test and fully support
licensed packaged binaries for your critical production needs

Enterprise Edition Only: Rack-Zone Awareness, Enhanced Security, Enhanced XDCR, Consulting, more …

http://www.couchbase.com/download

Why do customers choose Couchbase?

Best In Class Technology

- Performance and scalability
- Multi-purpose capabilities
- Market-defining innovation

Flexible Product Support

- Enterprise-grade support for licensed products
- Multiple SLA options (Silver, Gold, Platinum)
- Global, 24X7

World Class Service

- Center of Excellence drives customer success
- Comprehensive training
- Expert consulting

What we have?

Couchbase provides a complete Data Management Solution

- ✓ High availability cache
- ✓ Key-value store
- ✓ Document database
- ✓ Mobile embedded database
- ✓ Sync management

A simple, multi-purpose, always-on performance leader

Available as raw open source or Community Edition binary

- ✓ And rigorously tested, fully supported Enterprise Edition

Couchbase

Learning Services