Big Data Overview

Agenda

- What is “Big Data” all about?
- Fundamental Database Concepts
- The “Problem”: Relational Databases and Big Data
- The “Solution”: Scaling Techniques, NoSQL Products
- Hadoop

Current Trends in Computing

- Massive Amounts of data being generated & collected
  - Terabytes (TB 10^{12}) PetaBytes (PB 10^{15}), ExaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billions of people all over the world
- Lots of Different Kinds of data
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of the Marketplace
  - Competition is fierce
  - No tolerance for downtime, slow performance
  - Real-time data

Current Trends in Computing

- Ma[illegible]ed & collected
  - [illegible]kaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billions of people all over the world
- Lots of Different Kinds of data
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of the Marketplace
  - Competition is fierce
  - No tolerance for downtime, slow performance

Volume

Current Trends in Computing

- Ma[Volume]ed & collected
  - [Volume]kaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billion[Variety]
- Lots of D[Variety]
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of the Marketplace
  - Competition is fierce
  - No tolerance for downtime, slow performance

Current Trends in Computing

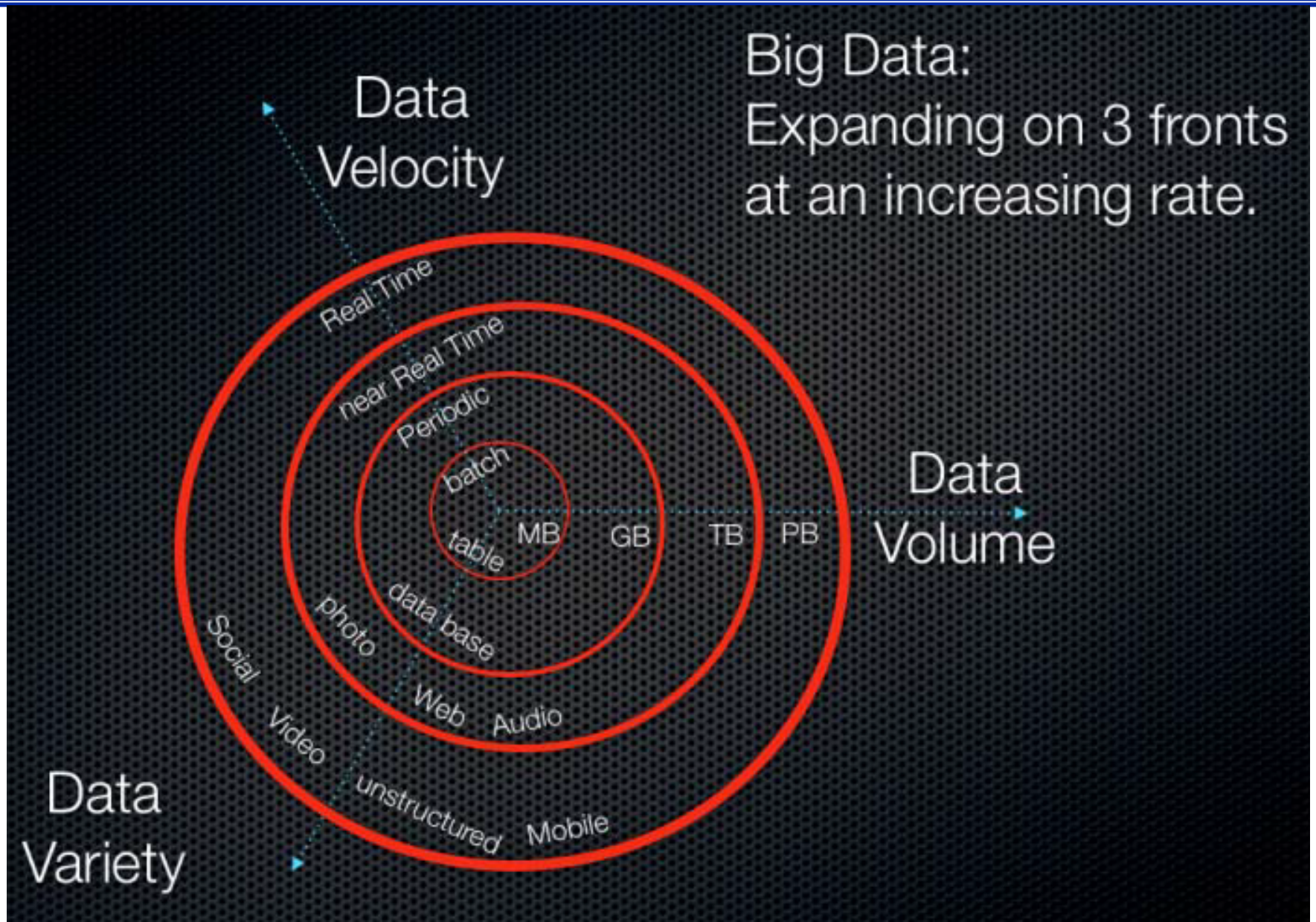
- Ma… ed & collected
  - …kaBytes (EB 10^{18})
- Explosion of Concurrent Users
  - Billion…
- Lots of D…
  - Unstructured Text, Log Streams, Click Data, Mobile Devices
- Demands of …
  - Competitior…
  - No tolerance for downtime, slow performance

Volume

Variety

Velocity

Big Data Overview



A Few Amazing Facts (1)

- Annual global IP traffic passed the zettabyte threshold by the end of 2016, and will reach 2.3 ZB per year by 2020.
- Smartphone traffic will exceed PC traffic by 2020.
- Traffic from wireless and mobile devices will account for two-thirds of total IP traffic by 2020.
- The number of devices connected to IP networks will be more than three times the global population by 2020.

Cisco The Zettabyte Era—Trends and Analysis http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

What is driving this data explosion?

- Smart Phones
- Apps
- The Internet of Things
- Digital Commerce
- Online Entertainment
- Cloud Computing
- Social Media

"Social media facilitate the most intimate of discussions among users, who seem to forget that they're communicating on a public forum."

Data Crush, Christopher Surdak

Smart Phones

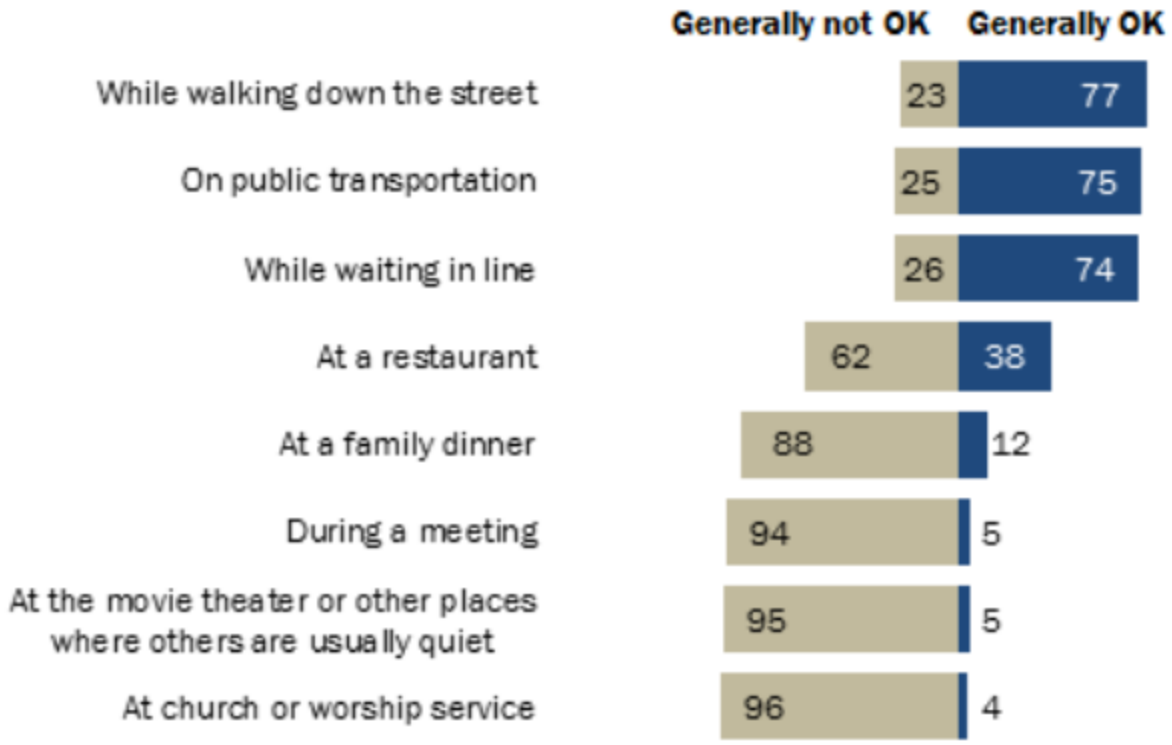
- Smartphones -- Only here for a little over a decade: the first iPhone came out in January 2007
- In 2010, 4.5 B people owned a mobile phone
- By 2017, there were 6.8 B mobile phone users (90% of humanity)
  - Of which 2.5 B are “smartphones”
- The mobile phone has become the individual’s dominant point of interaction with society

g Data Overview

"On the Internet, nobody knows you're a dog."

People Have Varying Views About When It Is OK Or Not OK To Use Their Cellphones

% of adults who believe it is OK or not to use a cellphone in these situations



Source: Pew Research Center American Trends Panel survey, May 30-June 30, 2014.
N=3,217 adults.

PEW RESEARCH CENTER

Apps

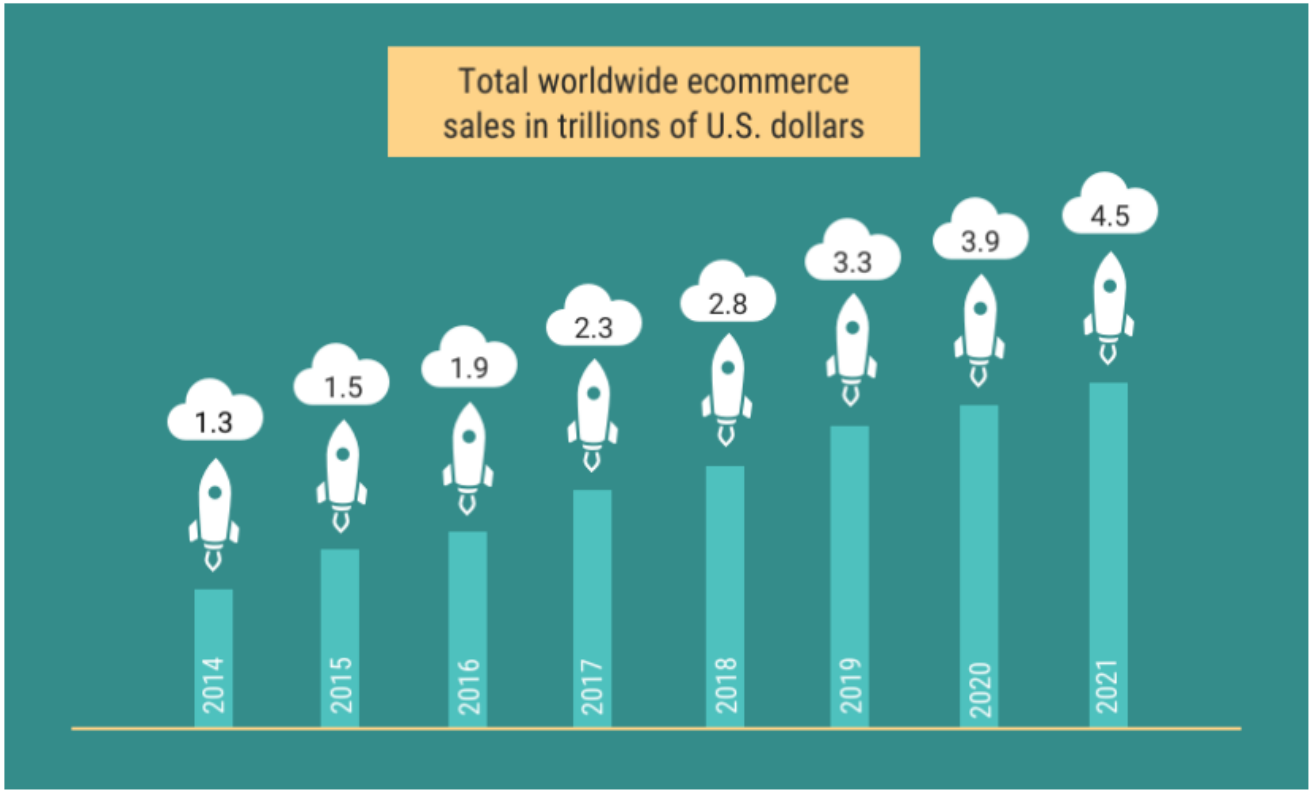
- 2012 – Apple reached 25 B annual downloads of apps
- 2013 – Apple reached 40 B annual downloads of apps
- Android apps – 50 B downloads in 2015
- Android apps – 90 B downloads in 2016
- By December 2017 – combined downloads = 197 B
- Google Play provides over a million downloadable apps & games

IOT – The internet of things

- Internet connected devices that are collecting and transmitting data
  - TVs with wifi connectivity
  - Cars with OnStar
  - Appliances (refrigerators, thermostat, doorbell, etc.)
  - Medical devices
  - “Wearables”
  - FitBit
  - Apple Watch
  - Robots
  - Assistants (Alexa, Google Home, Siri)

eCommerce

1. Global Retail Ecommerce Sales Will Reach $4.5 Trillion by 2021



eCommerce

Data from Statista forecasts a 246.15% increase in worldwide ecommerce sales, from $1.3 trillion in 2014 to $4.5 trillion in 2021.

Nearly threefold lift in online revenue.

Online Entertainment

- Online video has a 90 percent penetration rate among internet users on any device.
- In the United States, current data indicates more than 180 million digital video viewers with a penetration rate of approximately 75 percent of the online population.
- The majority of mobile video revenues are generated via subscriptions, followed by advertising.
- Current US mobile video subscription revenues have already surpassed 1 billion US dollars annually.

Cloud Computing

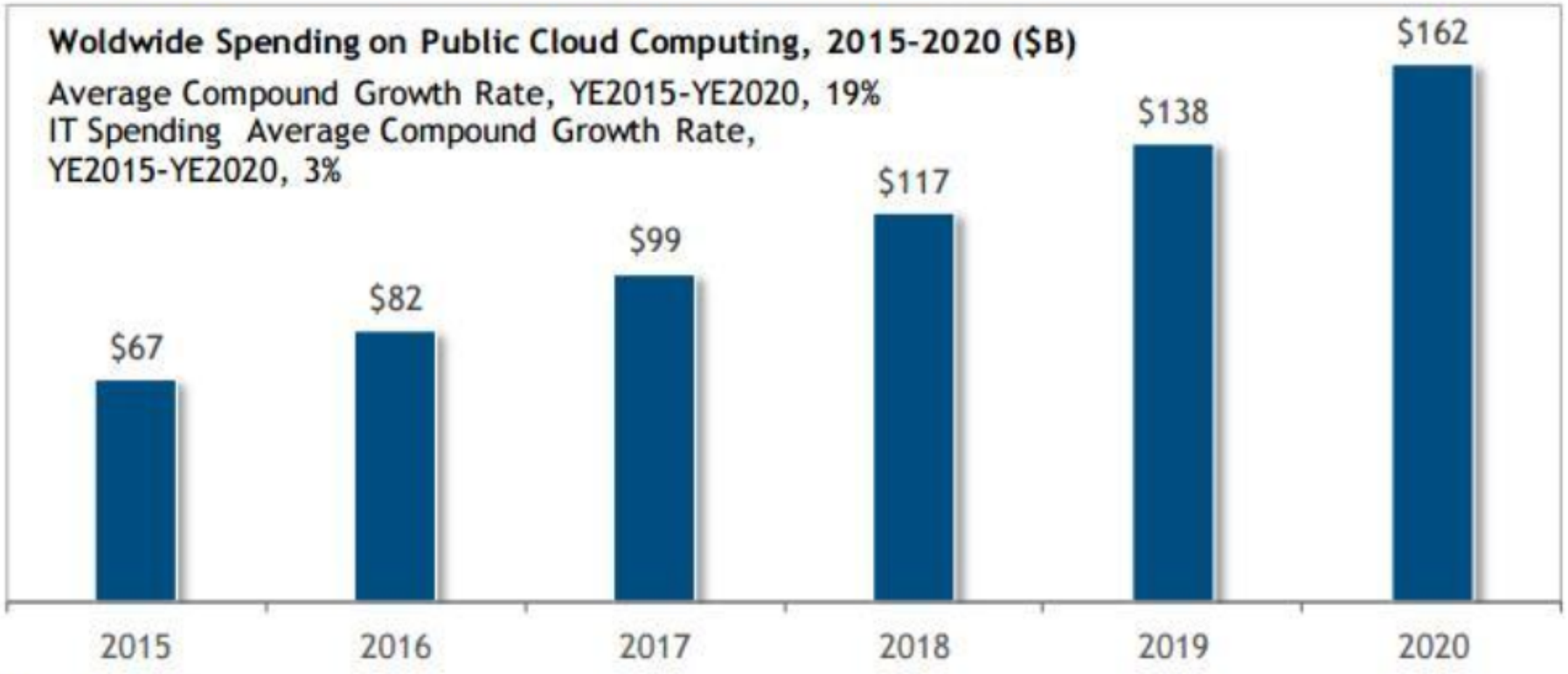
- Cloud computing is projected to increase from $67B in 2015 to $162B in 2020 attaining a compound annual growth rate (CAGR) of 19%.
- Gartner reports the worldwide public cloud services market grew 18% in 2017 to $246.8B, up from $209.2B in 2016.
- 74% of Tech Chief Financial Officers (CFOs) say cloud computing will have the most measurable impact on their business

Cloud Computing

Cloud computing spending is growing at 4.5 times the rate of IT spending since 2009 and is expected to grow at better than 6 times the rate of IT spending from 2015 through 2020.

According to IDC, worldwide spending on public cloud computing will increase from $67B in 2015 to $162B in 2020 attaining a 19% CAGR.

The Rapid Growth of Cloud Computing, 2015-2020



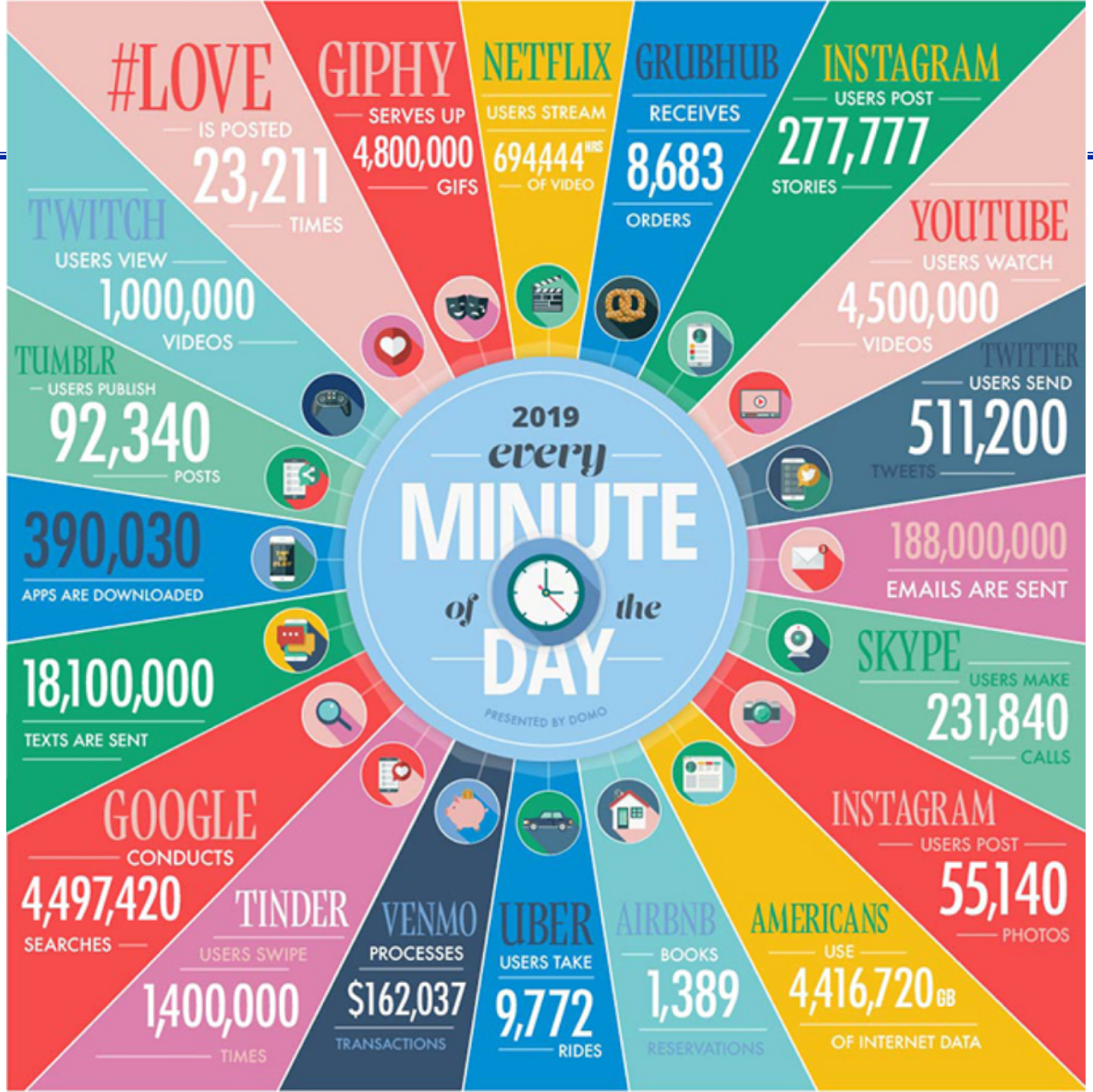
Source: IDC, 2016

Big Data Overview

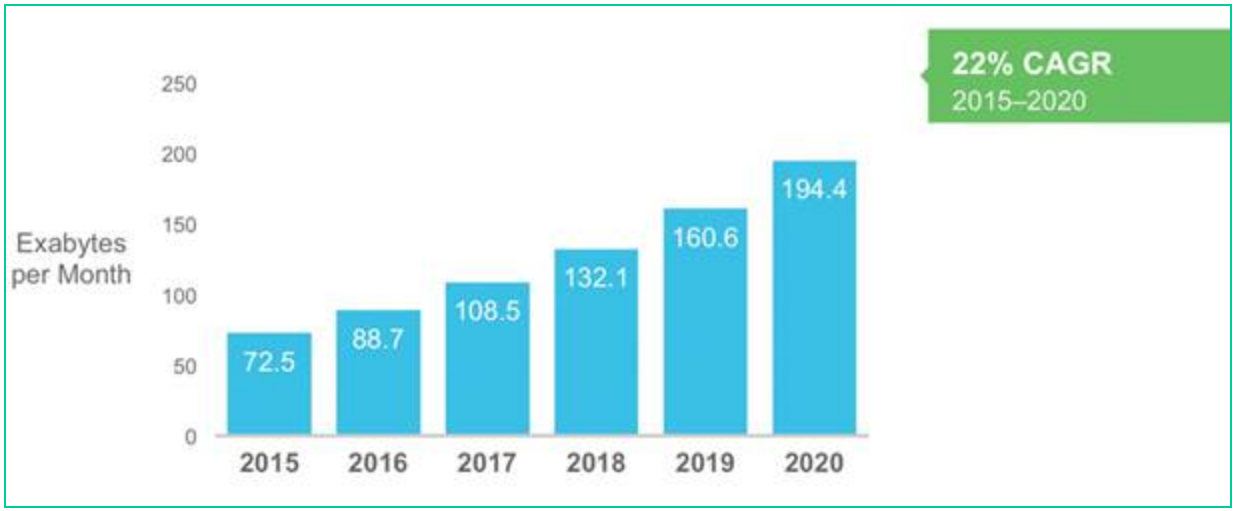
- 1,209,600 new data producing social media users each day.
- 656 million tweets per day!
- More than 4 million hours of content uploaded to Youtube every day, with users watching 5.97 billion hours of Youtube videos each day.
- 67,305,600 Instagram posts uploaded each day
- There are over 2 billion monthly active facebook users, compared to 1.44 billion at the start of 2015 and 1.65 at the start of 2016.
- Facebook has 1.32 billion daily active users on average as of June 2017
- 4.3 BILLION Facebook messages posted daily!
- 5.75 BILLION Facebook likes every day.
- 22 billion texts sent every day.
- 5.2 BILLION daily Google Searches in 2017.

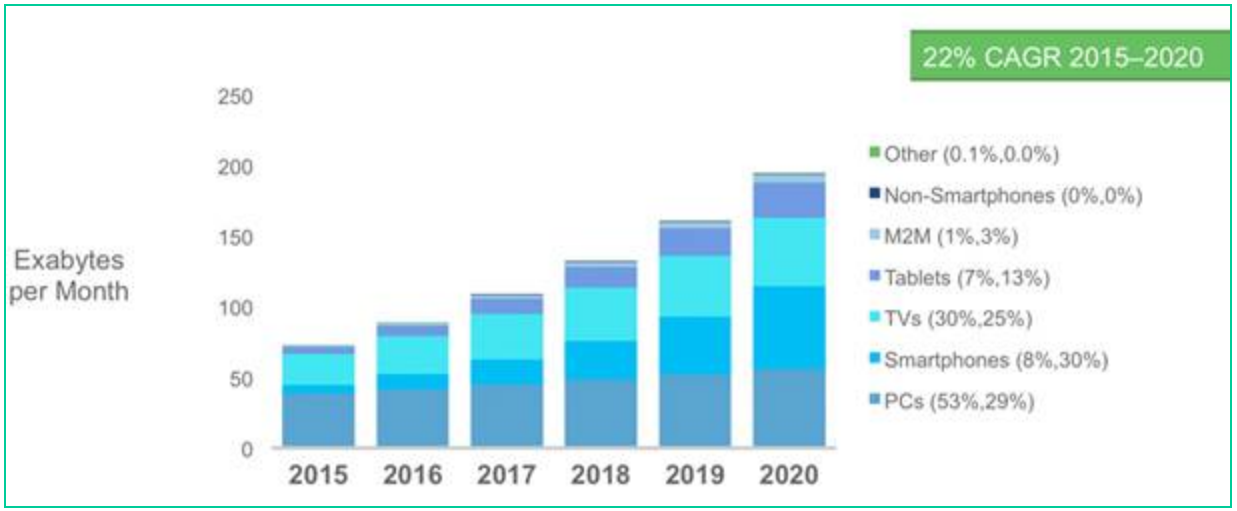
Big Data Overview

- The Mobile app Millennials used the most in 2015 – Facebook (21% of users)
- The mobile app Millennials used the most in 2017 – Amazon (35% of users)
- The time spent per user with digital media on mobile in US daily in 2017 – 2.3 hours
- The age group that spends the most time on apps monthly in US in 2015 – 18-24 (90.6 hrs on smartphone apps, 34.7 hrs. on tablet apps)
- The age group that spends the most time on apps monthly in US in 2017- 18-24 (93.5 hrs / month on smartphone apps, 27.6 hrs. / month on tablet apps)
- The major age group of users that operate a smartphone with two hands – 55+ (39% of users)


#LOVE
IS POSTED
23,211
TIMES
GIPHY
SERVES UP
4,800,000
GIFS
NETFLIX
USERS STREAM
694444 HRS
OF VIDEO
GRUBHUB
RECEIVES
8,683
ORDERS
INSTAGRAM
USERS POST
277,777
STORIES
TWITCH
USERS VIEW
1,000,000
VIDEOS
YOUTUBE
USERS WATCH
4,500,000
VIDEOS
TUMBLR
USERS PUBLISH
92,340
POSTS
TWITTER
USERS SEND
511,200
TWEETS
2019
every
MINUTE
of the
DAY
PRESENTED BY DOMO
390,030
APPS ARE DOWNLOADED
188,000,000
EMAILS ARE SENT
18,100,000
TEXTS ARE SENT
SKYPE
USERS MAKE
231,840
CALLS
GOOGLE
CONDUCTS
4,497,420
SEARCHES
INSTAGRAM
USERS POST
55,140
PHOTOS
TINDER
USERS SWIPE
1,400,000
TIMES
VENMO
PROCESSES
$162,037
TRANSACTIONS
UBER
USERS TAKE
9,772
RIDES
AIRBNB
BOOKS
1,389
RESERVATIONS
AMERICANS
USE
4,416,720 GB
OF INTERNET DATA

Big Data Overview





Cisco The Zettabyte Era—Trends and Analysis http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html

Challenges

- How can we collect, process and store so much data?
- How can we wade through so much data in a timely manner?
- How can we analyze so much data and gain meaningful insights?
- Can my existing systems/architectures handle this?
  - Why not?
- Can my existing staff (skills & tools) make the transition?

The Relational Problem

- The Relational Database: 40+ year-old technology
  - Tables, Rows, Columns, Keys, Joins, Commits
  - ACID Transaction Compliance
  - RDBMS Software Choices
- Traditional Database Server Architecture
  - Memory, CPU, Storage
  - The cost of scaling “UP”

Notes

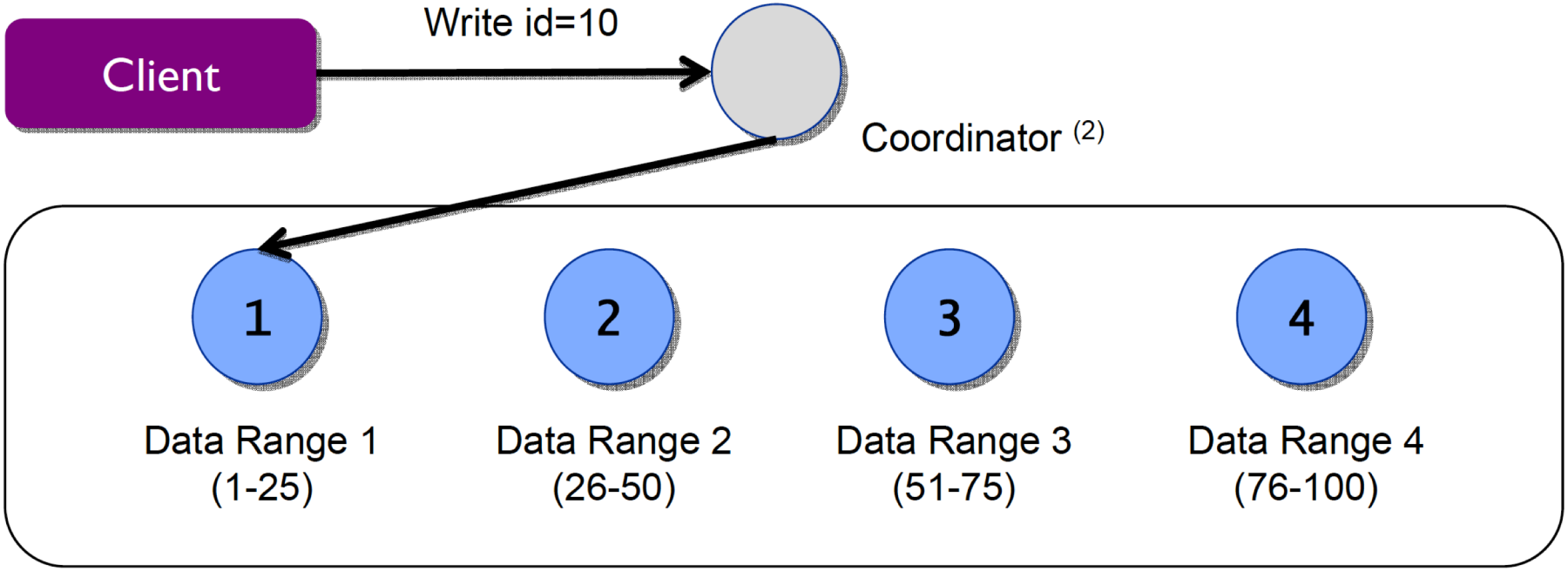
- **Be sure to cover:**
  - Scaling – Up versus Out
  - Clustering, replication, parallelization
  - Sharding
  - Clustering without sharding for HA – no SPOF
  - Master-Master, Master-Slave
  - Watcher
  - Problems with geo distributed clusters

Notes

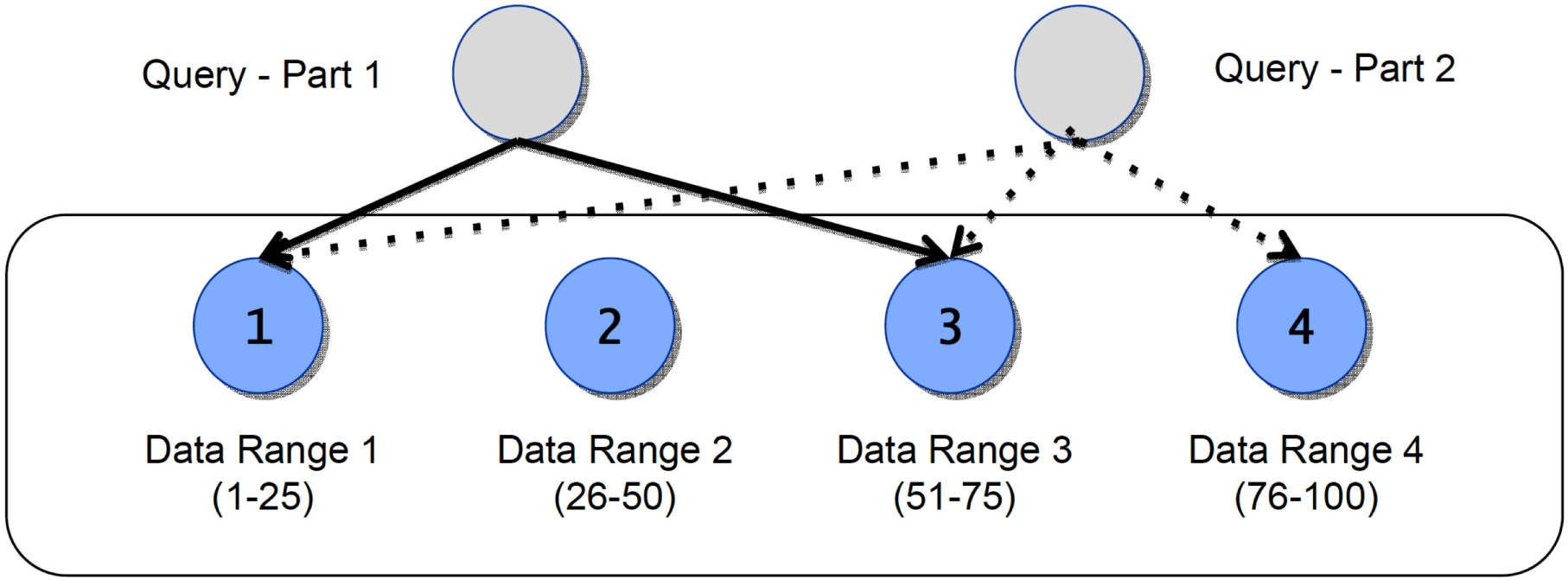
Stopped here Monday nov 18

- **Scale Out**: Use large clusters of commodity hardware
  - As opposed to **scale up** (with heavy duty servers)
  - Often done in the cloud for ease of startup and growth
- **Distribute**: Spread data and processing across nodes
  - **Distribute the data**, increasing data capacity
  - **Move processing to the data**, increasing processing capacity
- **Replicate data**: For availability and processing capacity
- **Relax Restrictions**: In various ways
  - **Relax** transactional and consistency guarantees
  - **Denormalize** data - sometimes radically
- **Specialize**: Design for your data and needs
  - Use data models and tools that work well for them

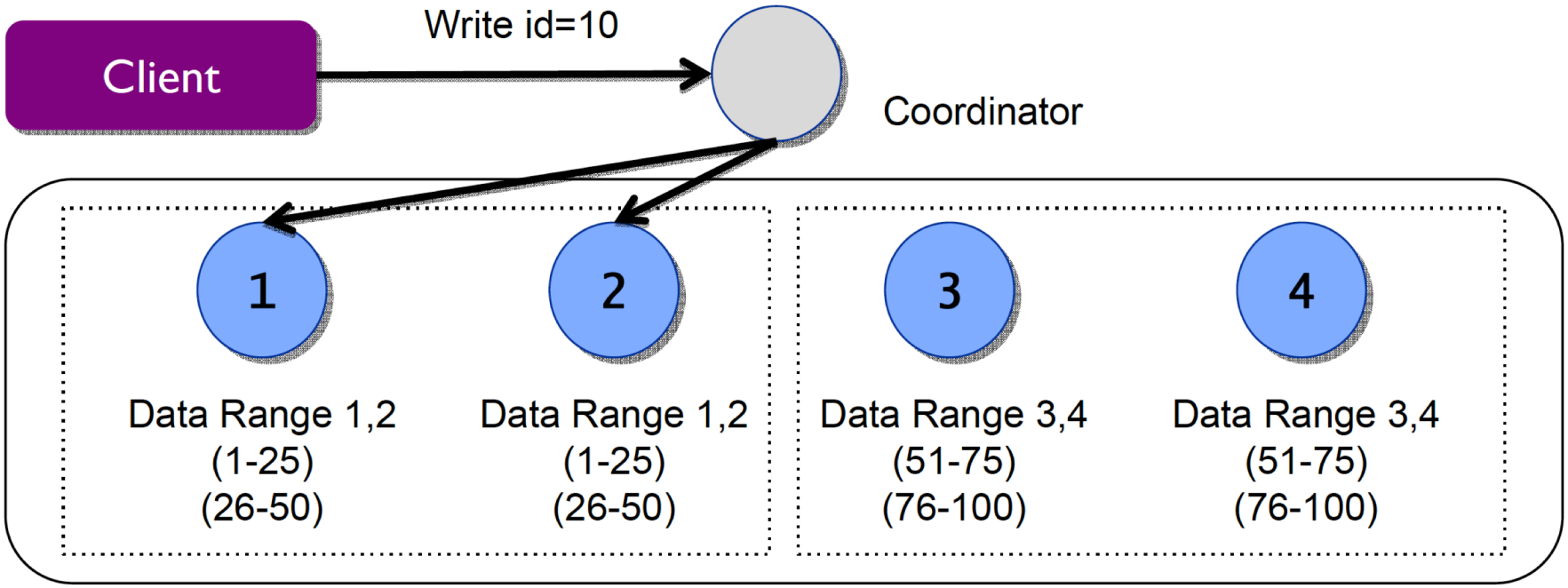
- **Sharding** partitions large data sets into smaller shards
  - Each shard hold parts of the data, so data is distributed
- Requires a partitioning scheme to shard data across nodes (1)
  - As well as a way to determine where sharded data goes to
- Below, 4 nodes each hold ¼ of a data range (1-100)
  - An incoming write with id=10 goes to node 1



- Consider a join that has to first read id=10 and id=70
  - And based on data in the read, reads data with id = 12, 55, and 80
    - Perhaps from another table
- You end up with many multiple reads across the cluster
  - Lots of network traffic and work to merge all the data into a result
  - Inherent in the relational model, and affects scalability [1]



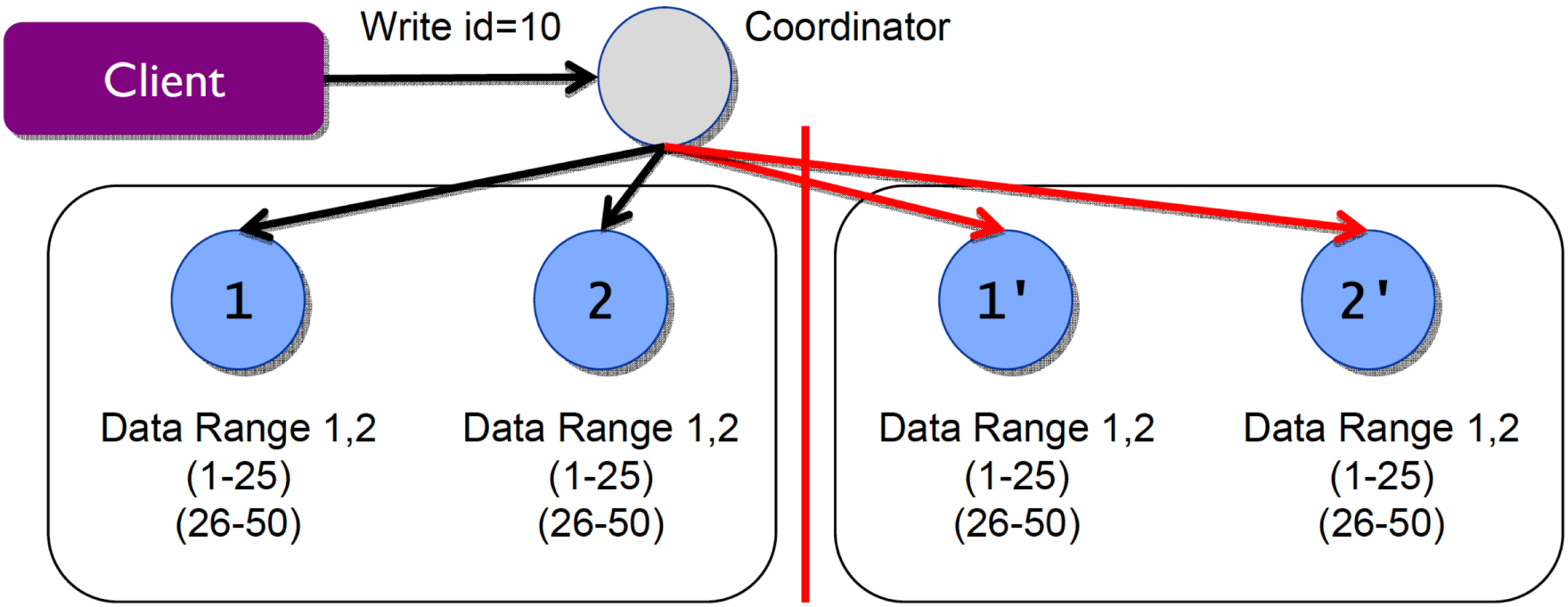
- Replication **duplicates data** to multiple nodes (1)
  - Increases scalability - operations can go to all replicas
  - Increases fault tolerance - operations continue if any replica is up
  - Two general types: **master/slave** and **peer-to-peer**
- Below, a write with id=10 goes to both nodes 1 and 2
  - The data is replicated to both nodes (2)



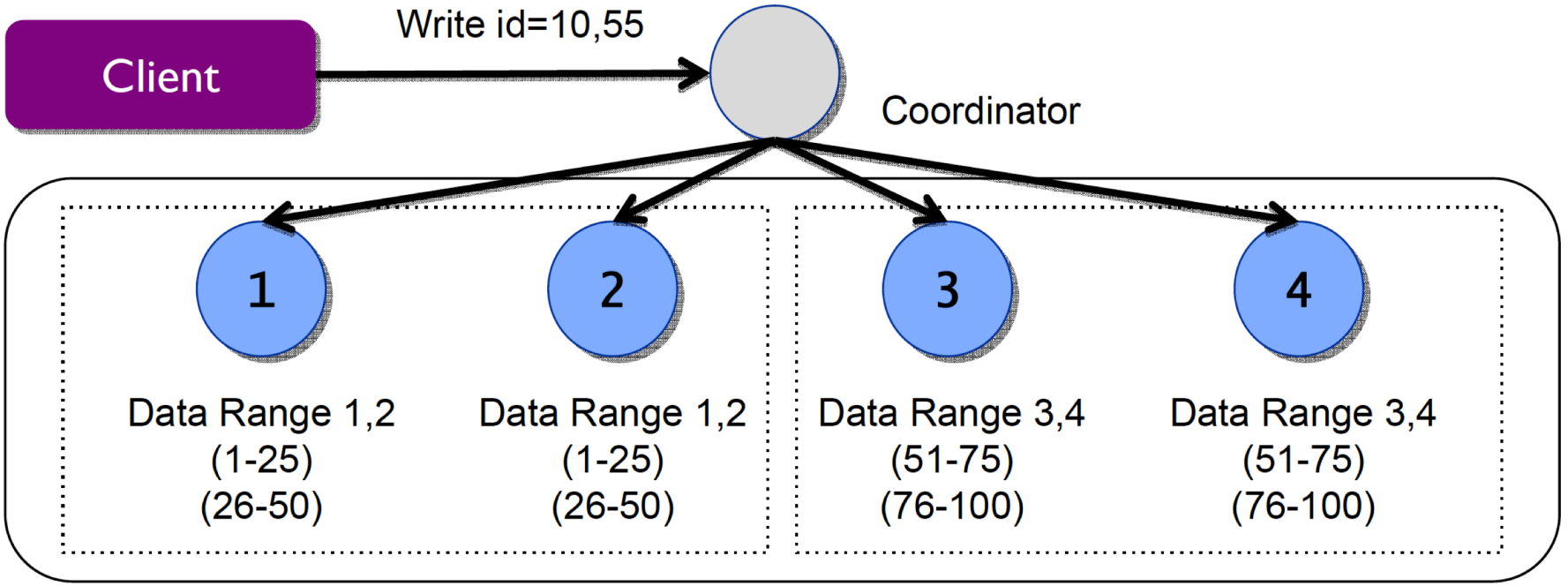
- **Master/Slave replication**: One node (master) is primary holder of data (1)
  - Updates go to master, and are then propagated to slaves
  - Reads can go to any node (master or slave)
  - Provides good scalability/availability for reads by adding slaves (2)
    - Since reads can go to any node with the data
- **Advantages**:
  - Reads scale well, and can easily be made highly available
  - Can give guarantees on update isolation (3)
- **Issues**:
  - Updates are limited by master's capabilities, limiting throughput
  - If master fails, failover to slave can occur, but may take time or manual intervention, limiting availability

- **Peer-to-peer replication**: All nodes are equal
  - Reads and writes can go to any node
  - Updates on any node are propagated to all replicas
  - Provides good scalability/availability for reads and writes
    - Both can go to any node with the data
- **Advantages**:
  - Reads and writes scale well and are highly available
  - Failure of a node doesn't necessarily prevent any reads/writes
- **Issues**:
  - These solutions are fairly complex
  - Hard to give any guarantees on updates (1)

- Replication **raises data consistency issues** among replicas
  - e.g. A read occurs before propagation of a write to all replicas (1)
- Consider two datacenters - with data replicated across them
  - What if the two datacenters can't communicate (a **partition**) (2)
  - You now face choices in how you continue to operate (or not)



- Consider a write that has to write to rows with ids=10 and 55
  - You'll write to all 4 nodes (because of replication and sharding)
  - For ACID guarantees, you'll have to use a complex TX protocol (e.g. 2 phase commit) to insure data integrity
  - The overhead of this protocol will affect scalability, and may add other complexities



Alternatives for handling Big Data

- Scale “OUT”, not “UP”
- Use cheap, commodity hardware, local disk
- Use SSDs
- Seamless Fault Tolerance -- No Single Point of Failure
- Distribute the Data: Replication & Sharding
- Distribute the Processing: Parallelization
- Redundancy: Replication
  - Master-Master
  - Master-Slave
- Relaxed ACID compliance restrictions

NOSQL Data Models

- **NoSQL** (“Not Only SQL”)
  - Uses clusters: Distribute the Data and the Processing
  - Replication provides redundancy, high availability, parallel processing
  - Horizontally scalable
  - Does not store data in tables with rigid row/column structure
    - Uses Aggregate model
  - Restricts join capabilities
  - Relaxes ACID transaction compliance
  - May use *non-SQL* query language

NOSQL Data Models

Relational

- Schema defines rigid structure
  - Tables, Rows, Columns
- Foreign Key relationships
  - Which support joins
- Uses SQL
- Maintains ACID compliance
- Normalized: store a value only once
- Clustering is a challenge
- Main DBMS players are quite expensive

NoSQL

- Stores related values in aggregates
- Flexible structure:
  - Ranges from none to some
- No joins
- Relaxed ACID compliance
- De-normalized
- Uses alternate query language
- Easily supports clustering
- Almost all players are open source

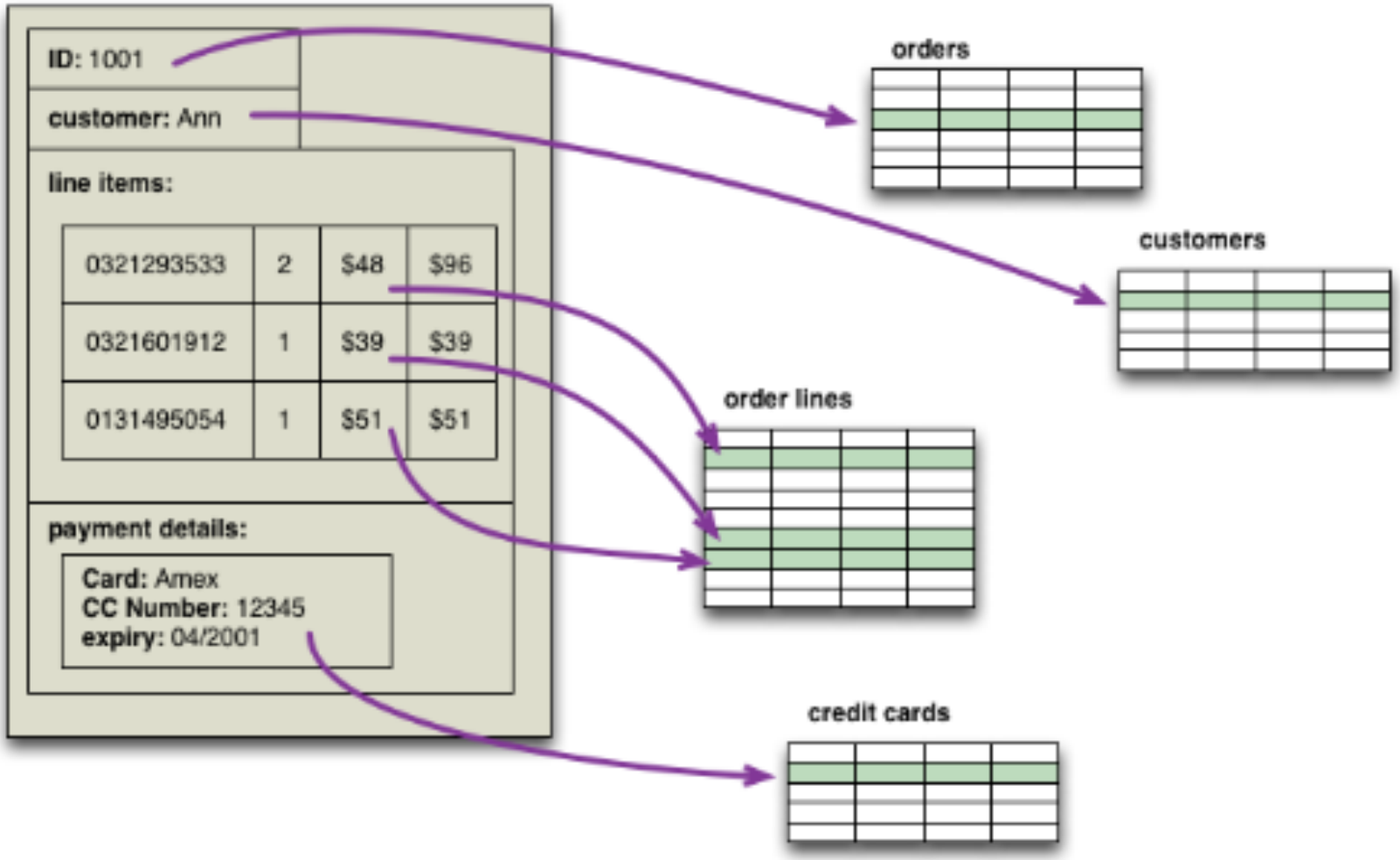
NOSQL Data Models

- **NoSQL "aggregate" Data Model**
  - RDBMS requires Tables, Rows, Columns as data stores
    - Columns have "domains"
    - Third normal form – no multi-values, "store it once"
  - NoSQL sytems use AGGREGATES as data stores
    - Data values are grouped together as users need them
    - Think of an unnormalized document, or an "object"
    - Contains related values that are retrieved and manipulated together

NOSQL Data Models

- **Why aggregates?**
  - It is difficult to spread a relational model across nodes in a cluster
    - Replication and sharding
  - Each query should minimize the number of nodes being accessed
  - Data values that are accessed together should live on the same node

NOSQL Data Models



NOSQL Data Models

- **NoSQL Data Schema/Models**
  - Document Store (using XML or JSON format)
  - Graph (using node/edge structures with properties)
  - Key-Value pairs
  - Wide Columnar store (rows with dynamic columns holding key-value pairs)

NOSQL Data Models

- Document
  - MongoDB
- Graph
  - Neo4j
- Key-Value
  - Amazon Dynamo
- Wide Column
  - Google BigTable, Apache Cassandra

NOSQL Data Models

- **Document Database**
  - Organized around a “document” containing text
  - Can handle very large data volumes
  - Provides Speed and Scalability
  - Document format is easily understood by humans
  - No “schema”, but JSON/XML provides internal structure within a document
  - Documents are indexed and stored within “collections”
  - Supports full text search
- **Popular Implementations** (open source)
  - MongoDB
  - CouchDB

JSON

```
{
   _id: ObjectId(7df78ad8902c)
   title: 'MongoDB Overview',
   description: 'MongoDB is no sql database',
   by: 'tutorials point',
   url: 'http://www.tutorialspoint.com',
   tags: ['mongodb', 'database', 'NoSQL'],
   likes: 100,
   comments: [
      {
         user:'user1',
         message: 'My first comment',
         dateCreated: new Date(2011,1,20,2,15),
         like: 0
      },
      {
         user:'user2',
         message: 'My second comments',
         dateCreated: new Date(2011,1,25,7,45),
         like: 5
      }
   ]
}
```

XML

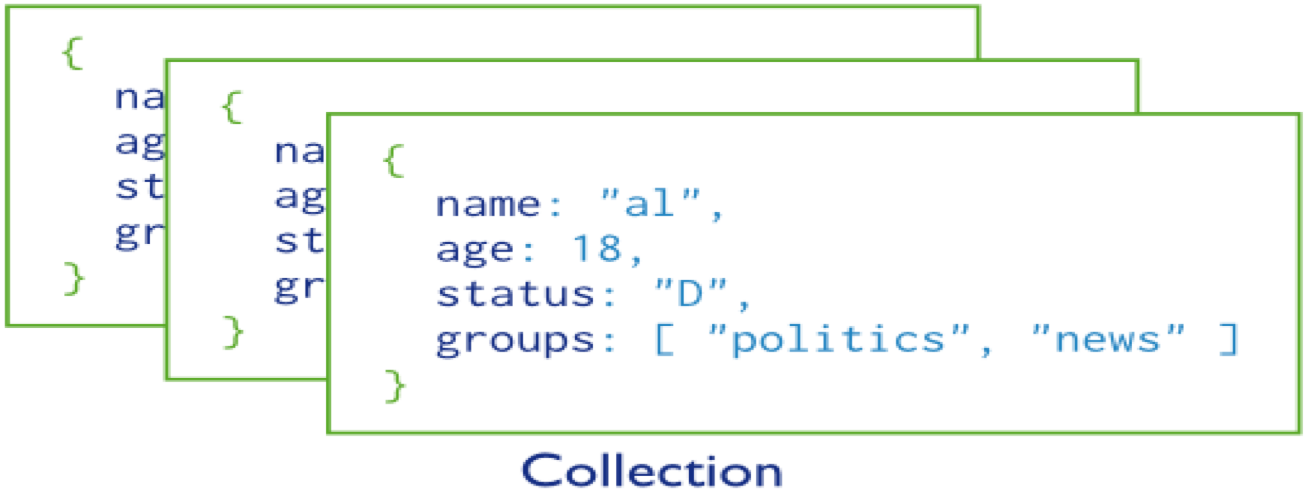
```
<contact>
        <firstname>Bob</firstname>
        <lastname>Smith</lastname>
        <phone type="Cell">(123)555-0178</phone>
        <phone type="Work">(890)555-0133</phone>
        <address>
                <type>Home</type>
                <street>123 Black St.</street>
                <city>Big Rock</city>
                <state>AR</state>
                <zip>23225</zip>
                <country>USA</country>
        </address>
</contact>
```

NOSQL Data Models

- **Document Database**
  - Keeps related information together (not normalized into tables)
  - Access to a document is fast (index/key/URL)
  - ACID compliance is maintained only within a document
  - Cannot easily “join” across documents
  - Documents are kept in “collections”

NOSQL Data Models

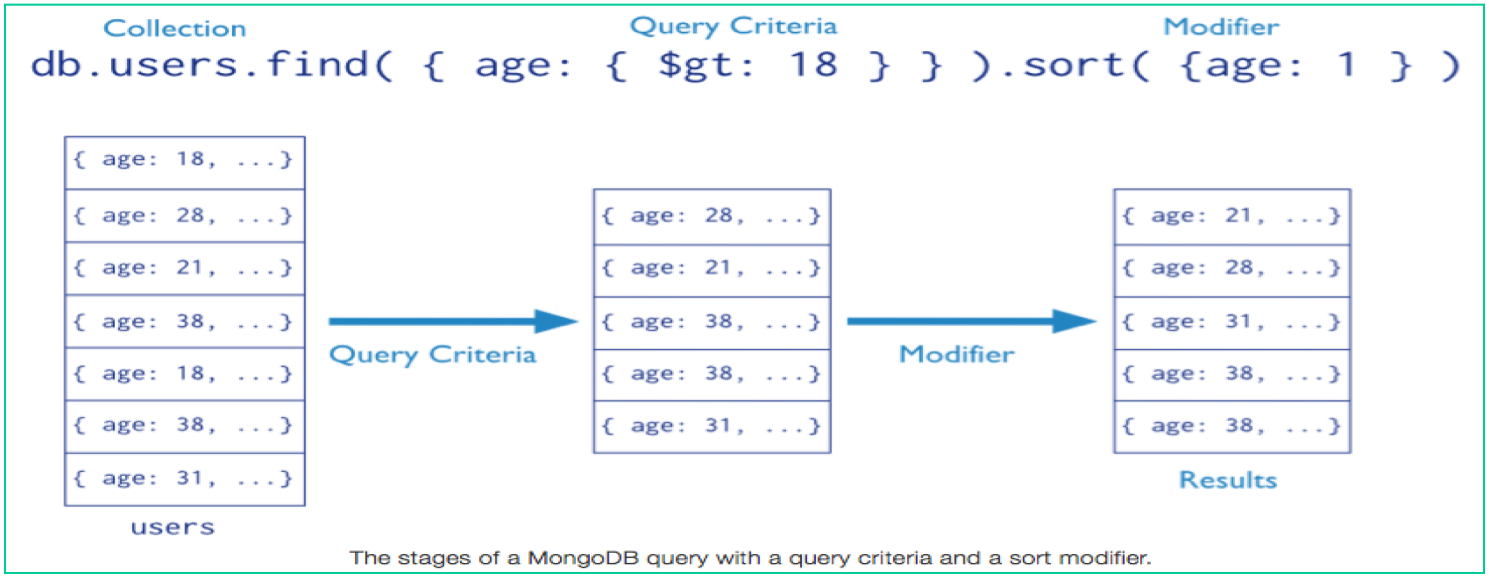
A MongoDB document.



A collection of MongoDB documents.

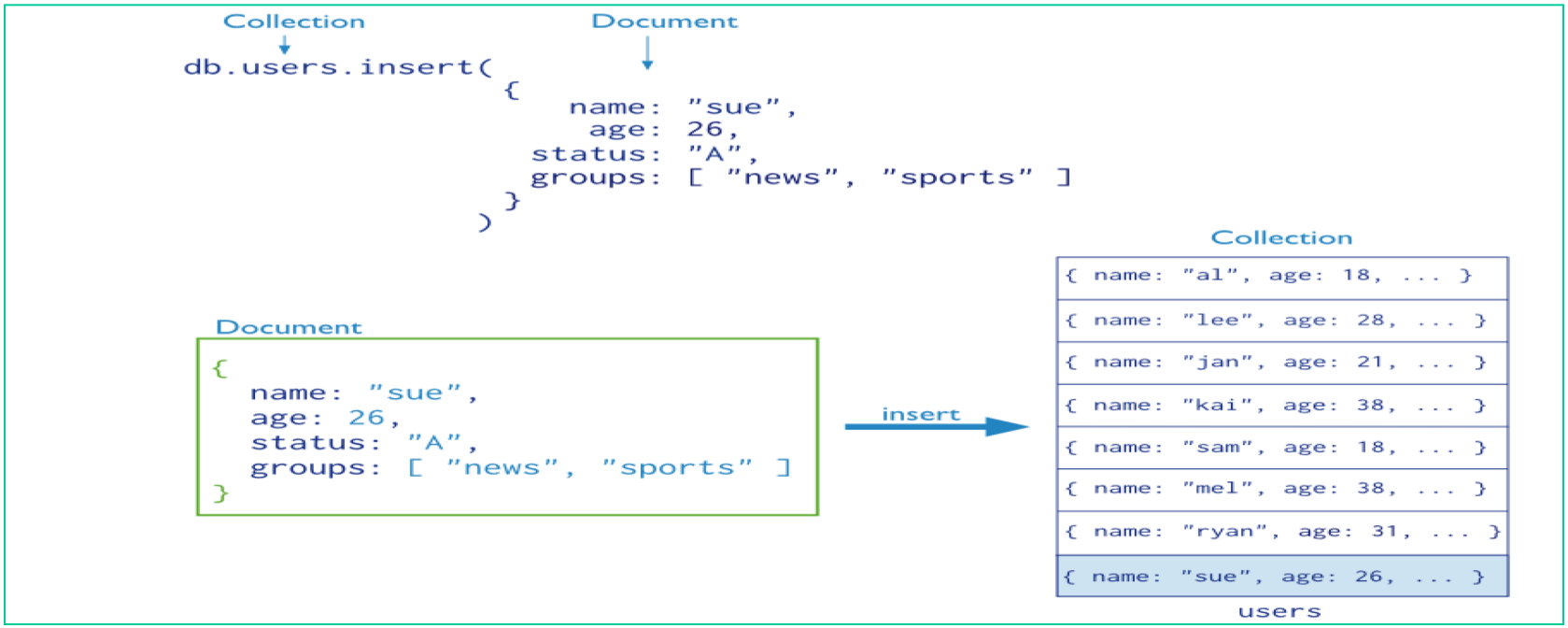
In MongoDB a query targets a specific collection of documents. Queries specify criteria, or conditions, that identify the documents that MongoDB returns to the clients.

A query may include a *projection* that specifies the fields from the matching documents to return. You can optionally modify queries to impose limits, skips, and sort orders.



The stages of a MongoDB query with a query criteria and a sort modifier.

Data modification refers to operations that create, update, or delete data. In MongoDB, these operations modify the data of a single *collection*. For the update and delete operations, you can specify the criteria to select the documents to update or remove.



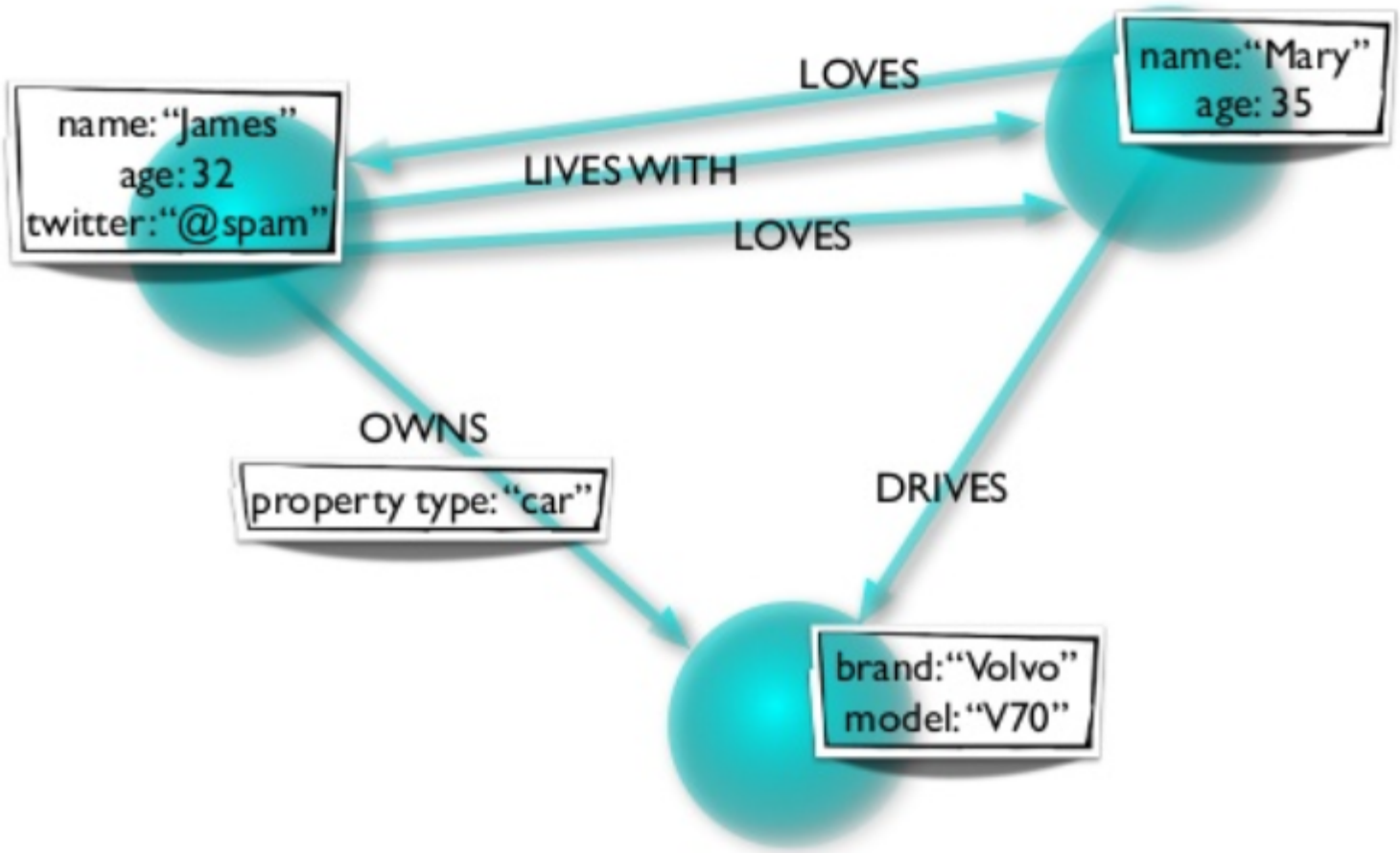
NOSQL Data Models

- **Start here on Friday Nov 22**

NOSQL Data Models

- **Graph Database**
  - Uses a graph structure consisting of
    - Nodes – Represents an entity (like a person)
    - Edges – Represents a relationship between entities
    - Properties – Attributes associated with Nodes and Edges
  - Supports navigation along edges from a starting point node
  - Designed for applications tracking the inter-connections among entities
    - Who is friends with whom? (In a social network application)
    - Who is following me, who am I following?
  - Uses a pattern matching query language to navigate nodes & edges
- **Popular Implementations** (open source)
  - Neo4j

NOSQL Data Models



NOSQL Data Models

- **Key-Value Pairs Database**
  - "Schemaless"
  - Maps a key to an opaque value, like an associative array
  - Simple operations (put, get, remove, modify)
  - Keys are unique in a collection
  - Most useful when access is by the primary key
  - May be a building block for other data models (such as key-value pairs within a wide-columnar store like Cassandra)
- **Popular Implementations**
  - Amazon Dynamo (available via AWS in the cloud)
  - Riak, Redis (open source)

Product Catalog Example

```
{
    Id: 206,
    Title: "20-Bicycle 206",
    Description: "206 description",
    BicycleType: "Hybrid",
    Brand: "Brand-Company C",
    Price: 500,
    Color: ["Red", "Black"],
    ProductCategory: "Bike",
    InStock: true,
    QuantityOnHand: null,
    RelatedItems: [
        341,
        472,
        649
    ],
    Pictures: {
        FrontView: "http://example.com/products/206_front.jpg",
        RearView: "http://example.com/products/206_rear.jpg",
        SideView: "http://example.com/products/206_left_side.jpg"
    },
    ProductReviews: {
        FiveStar: [
                "Excellent! Can't recommend it highly enough!  Buy it!",
                "Do yourself a favor and buy this."
        ],
        OneStar: [
                "Terrible product!  Do not buy this."
        ]
    }
}
```

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.CaseStudy.html

Key-Value Pairs Database Example (Amazon DynamoDB)

- The key value (Id) is 206.
- Most of the attributes have simple data types, such as String, Number, Boolean and Null.
- One attribute (Color) is a String Set.
- The following attributes are document data types:
  - A List of Related Items. Each element is an Id for a related product.
  - A Map of Pictures. Each element is a short description of a picture, along with a URL for the corresponding image file.
  - A Map of ProductReviews. Each element represents a rating and a list of reviews corresponding to that rating. Initially, this map will be populated with five-star and one-star reviews.

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.CaseStudy.html

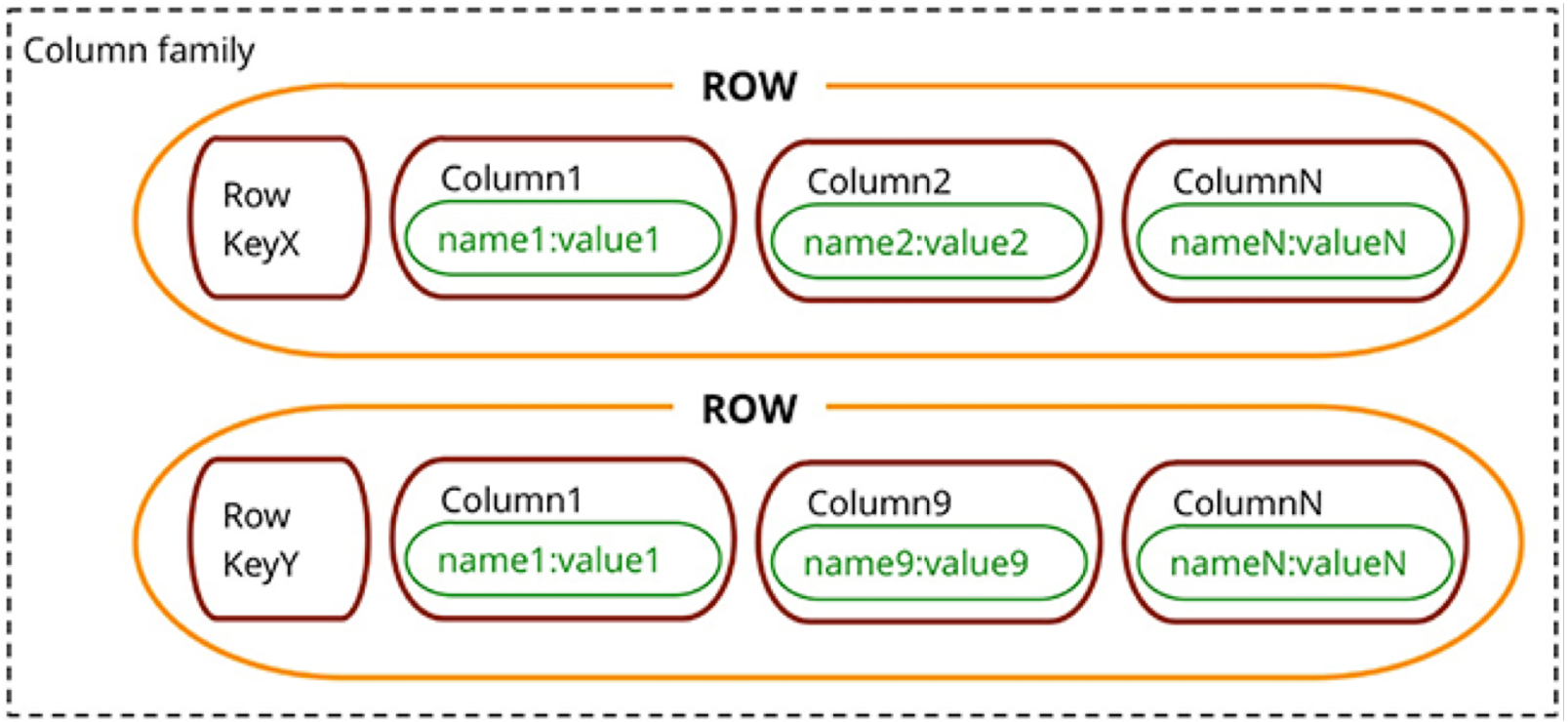
NOSQL Data Models

- **Wide-Column (Column Family) Store Database**
  - A TWO-LEVEL aggregate
  - Data is stored within “collections” of dynamic related columns
  - Similar to key/value with the value having columnar structure
- **Based on Google’s “Big Table”**
- **Popular Implementations** (open source)
  - Cassandra
  - HBase

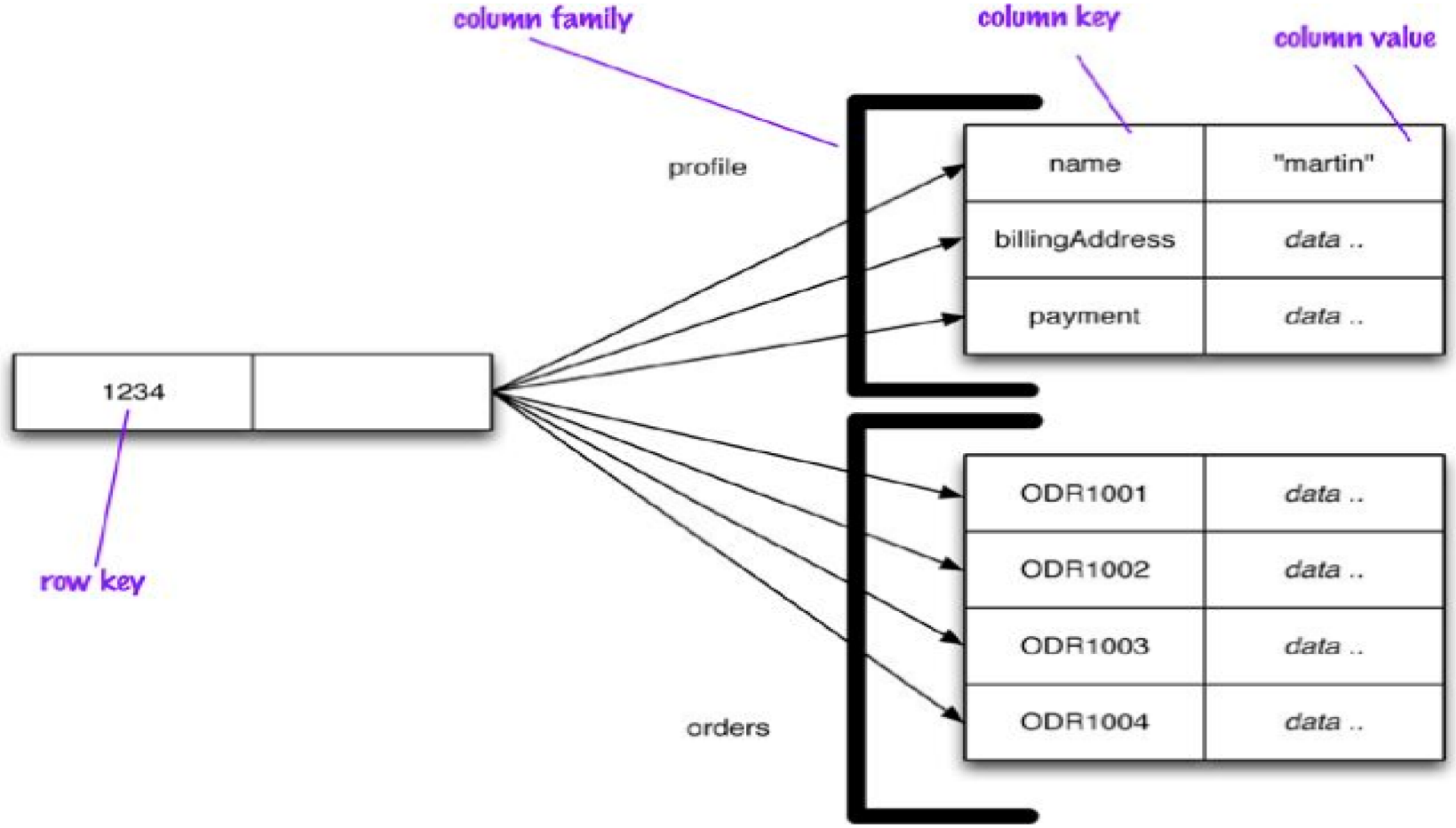
NOSQL Data Models

- **The first key is the row-key**
  - The entire row is an aggregate of related data
  - The row consists of many key-value pairs (“columns”)
- **The second key is the column family**
  - Each column family consists of sparse key-value pairs
    - “Sparse” means the column value isn’t stored if it isn’t needed

NOSQL Data Models



NOSQL Data Models



- **Wide-Column Store Example**
- Consider an application that needs to store stock trades and retrieve them by stock symbol
- Some data that might be stored includes the following:
  - symb: The stock symbol
  - id: A unique trade id
  - date: Date of the trade
  - price: Trade price
  - quant: Number of shares traded
  - notes: General comment field.

NOSQL Data Models

- Below, we illustrate trade data in a wide column store
  - The stock symbol (symb) is the row key
  - Generally the complete row is stored on a single node
  - The trades are stored as columns in the row
  - If there were many trades, the row could get very wide
  - Note also, that Trade 2 has no notes column
  - Contrast this to a normalized relational model with a trade table which would generally store trades in separate rows

symb	id	date	price	quant	notes	id	…	quant	id	…	notes
GM	1c3f	2014-09-15	32.96	100	blah	2a5f		400	2e99		bar

Benefits of a Wide Column Store

- Lookup of rows by row key can be **very fast**
- In a distributed cluster system, the complete row is stored on one node
- May be stored redundantly across the cluster
- Can be retrieved with one access
- Good fit for **scale-out** systems
- Can scale to large capacity and high availability
- The columns are "**sparse**"
  - That is, columns with no values are absent, saving space
  - As shown in Trade 2 having no Notes column
- Flexible access to column data in a row
  - You can flexibly query on them
  - e.g. - get all trades, get the last 10 trades, or 1st ten trades, etc.

Issues with a Wide Column Store

- The data model is **complex**, and **not very intuitive**
- Generally, you create your data model **based on your queries**
  - Possibly taking into account how data is distributed
- For example, querying for trades by stock symbol is very fast for the trade table shown earlier -- it's just one query to one node
- Consider finding all trades (for all stocks) on a particular date
  - The previous data model is not very good for this
  - You'd need to query every node, get the trades from that node, then combine them all for the result
- In practice, that may not scale
- You could, however, store data optimized for retrieval by date
- This would require another table / column layout

Issues with a Wide Column Store

- Client code **may be extensive**
  - For example, to keep track of trades by both stock symbol and trade date, you may need to maintain two sets of data
  - Which needs to be done via application code on a client
- Different from relational model where you maintain one set of data, and just write queries as needed
- However, wide column stores tend to have much more scale out capability -- Which is why we use them in the first place

Stop here

Linear Scalability as in Cassandra

- Capacity may be easily added simply by adding new nodes
- For example,
  - Suppose 2 nodes can handle 100,000 transactions per second,
  - 4 nodes will support 200,000 transactions/sec, and
  - 8 nodes will tackle 400,000 transactions/sec: