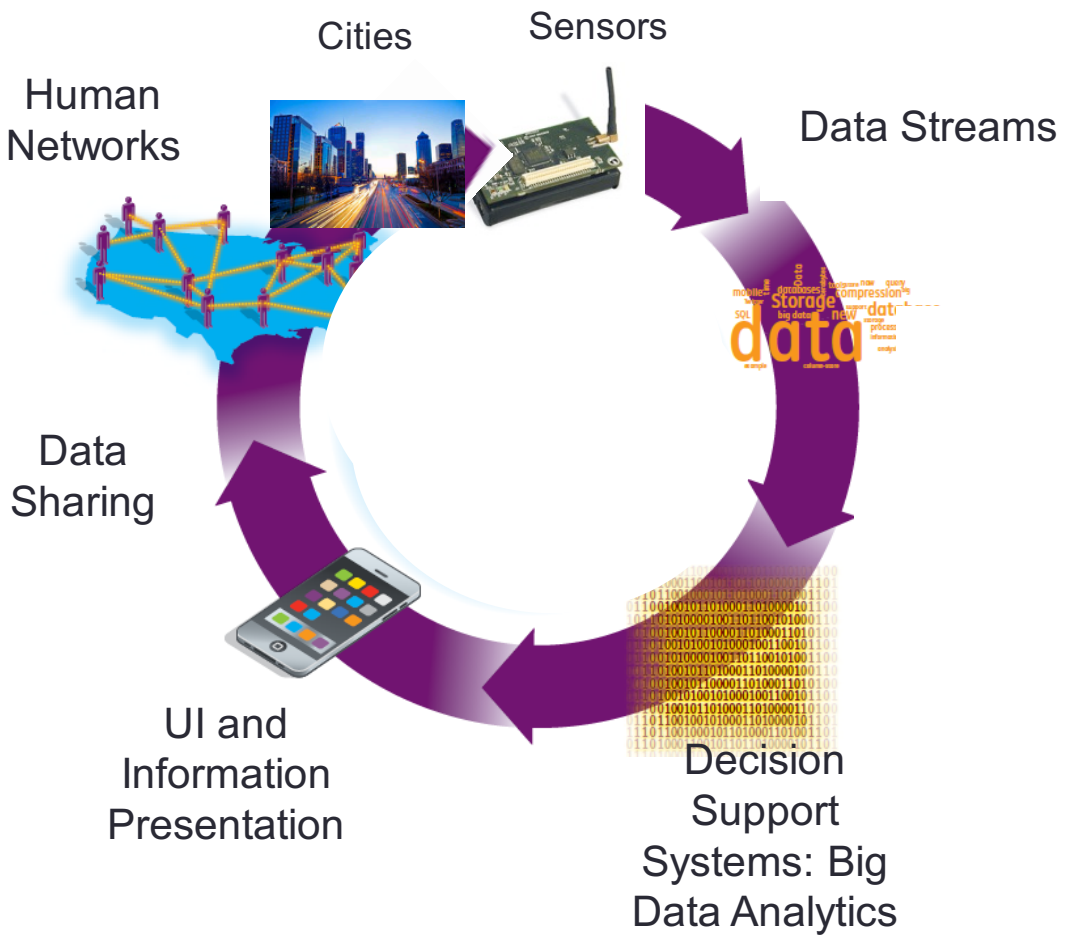
Big Data

Abhishek Dubey

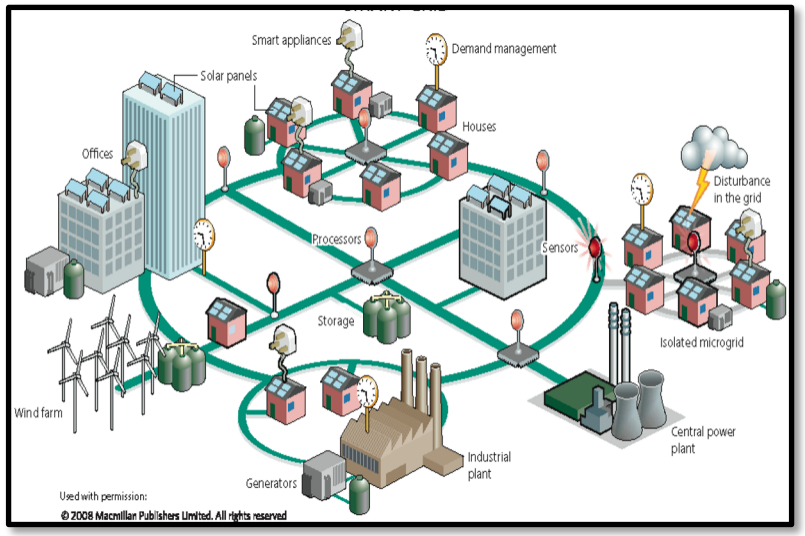
Tel (615) 343-7472 Fax (615) 343-7440
1025 16th Avenue South|Nashville, TN 37212
www.isis.vanderbilt.edu

VANDERBILT UNIVERSITY

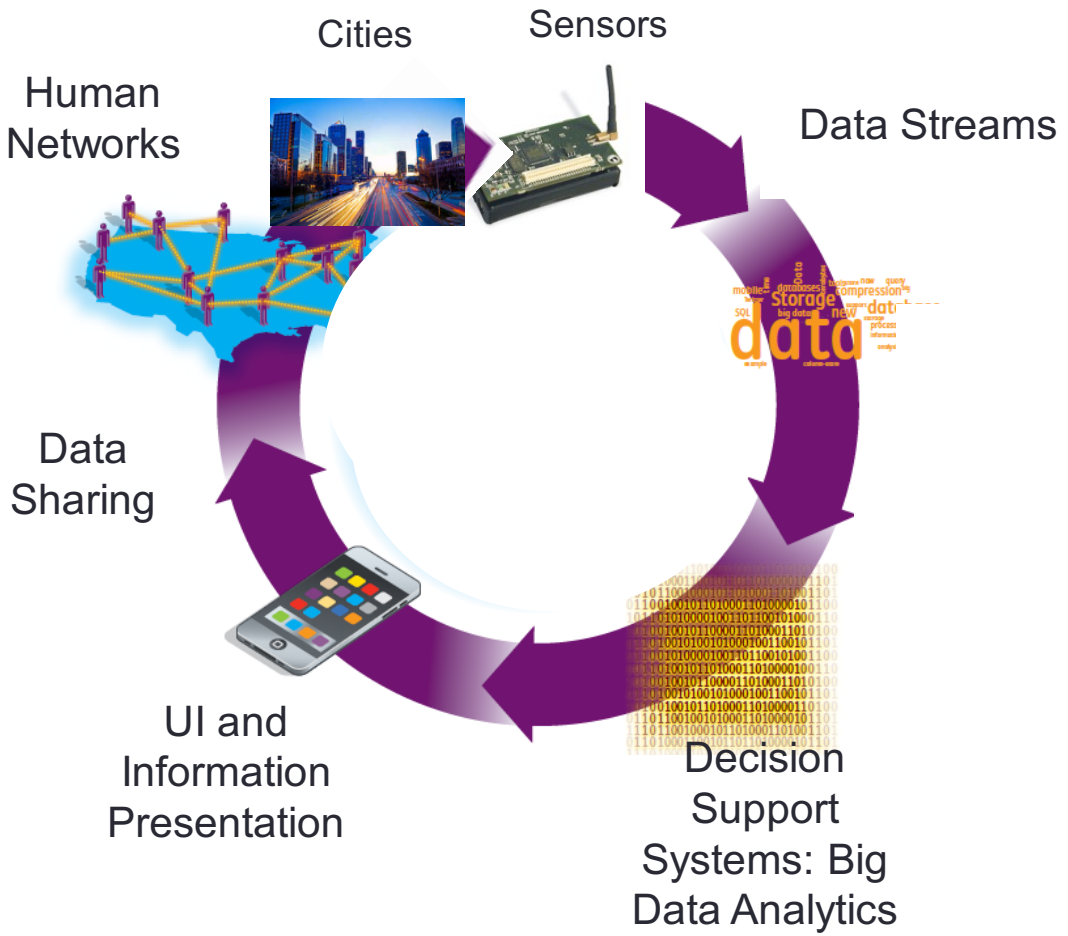
Big Data + AI


Cities
Sensors
Human Networks
Data Streams
Data Sharing
UI and Information Presentation
Decision Support Systems: Big Data Analytics

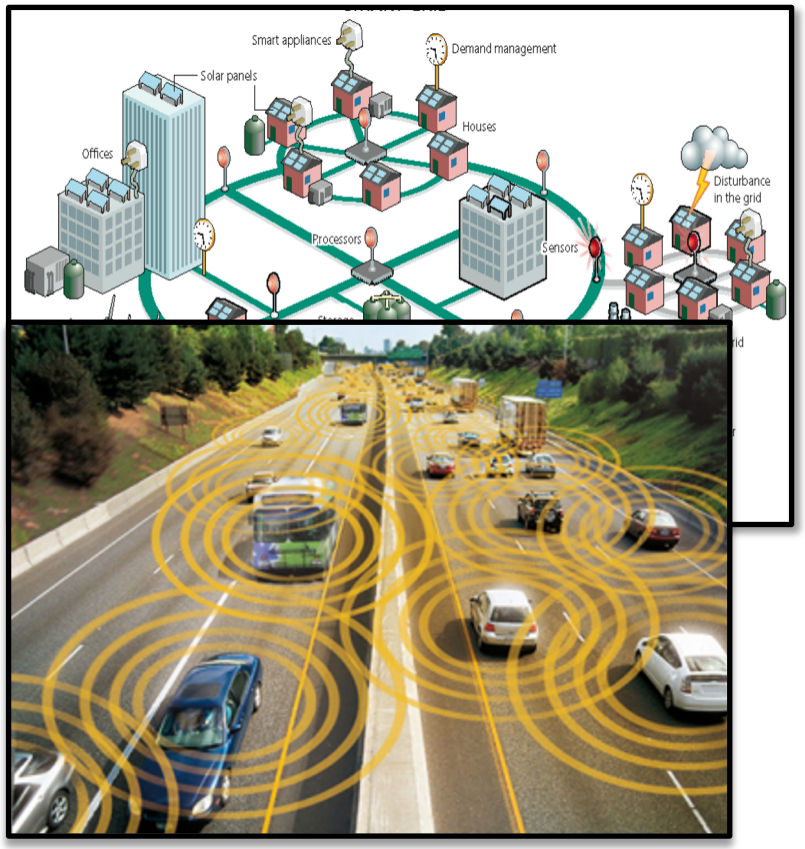
Big Data + AI



Smart Grid

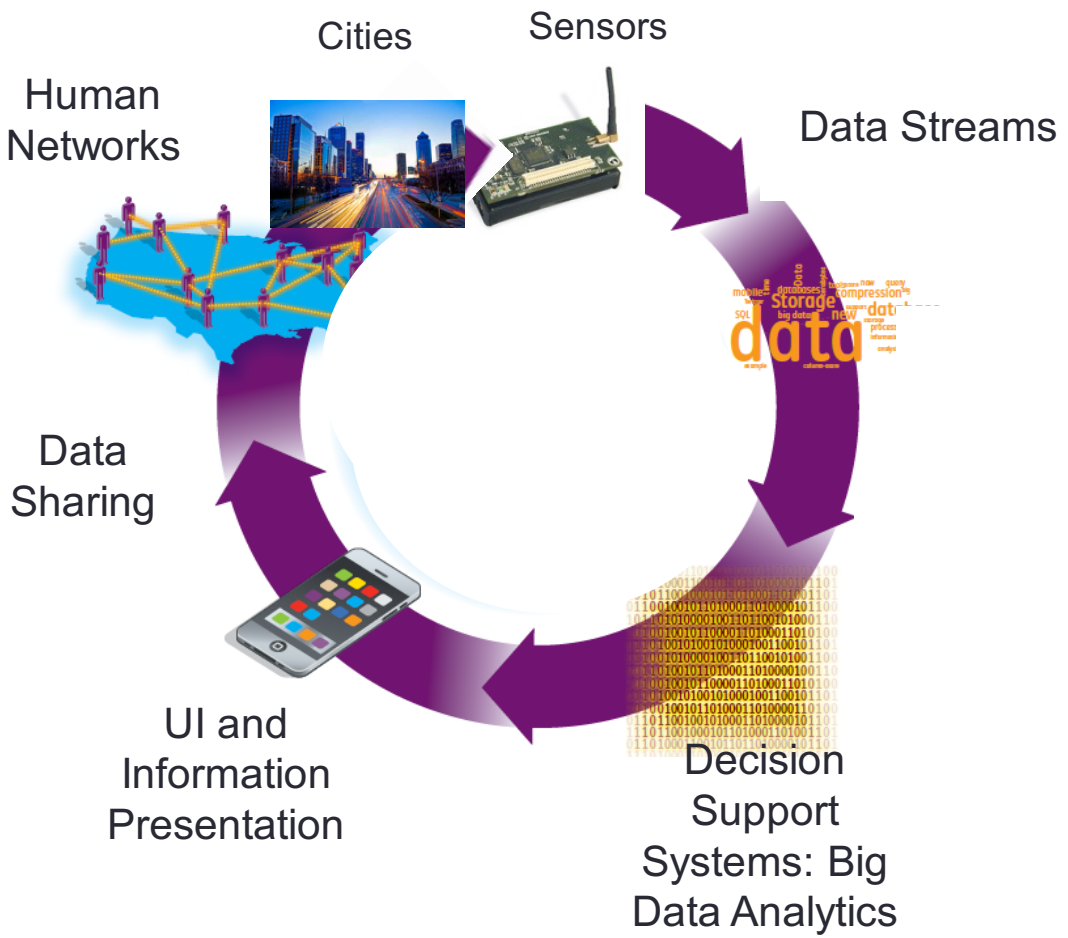


Big Data + AI

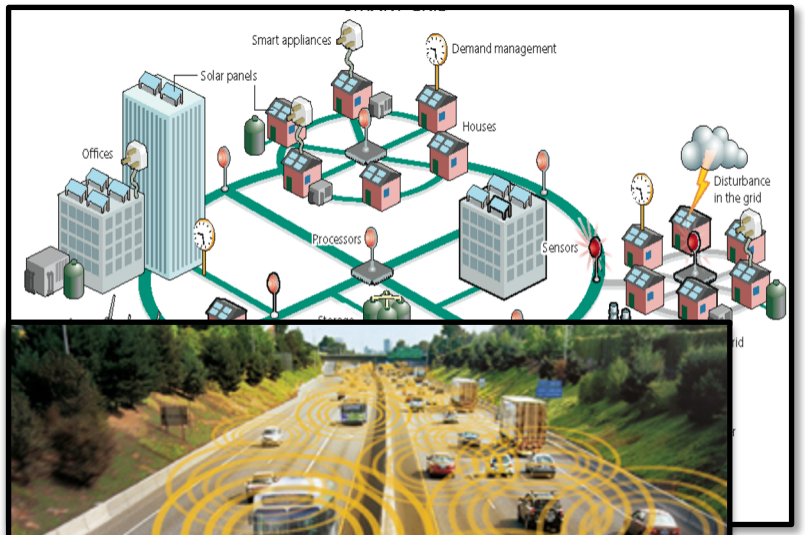


Smart Grid

Smart Transportation

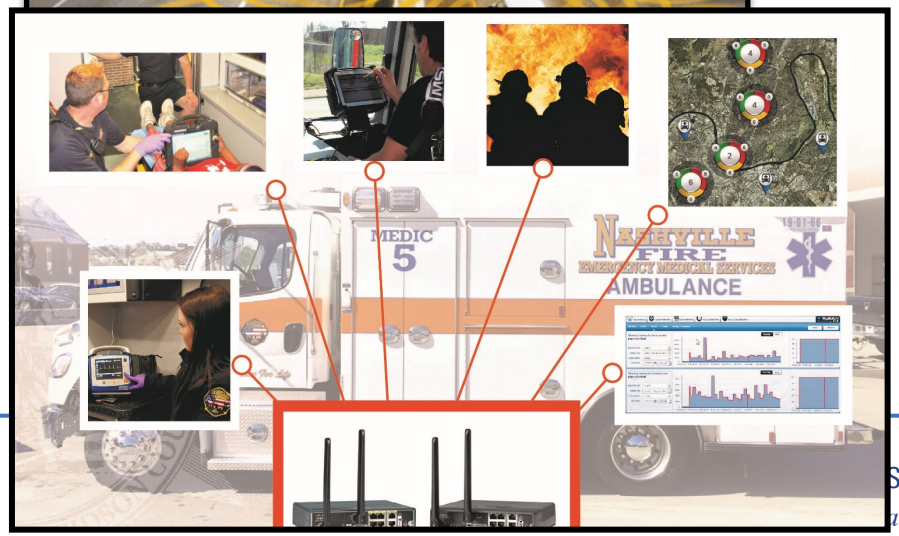


Big Data + AI

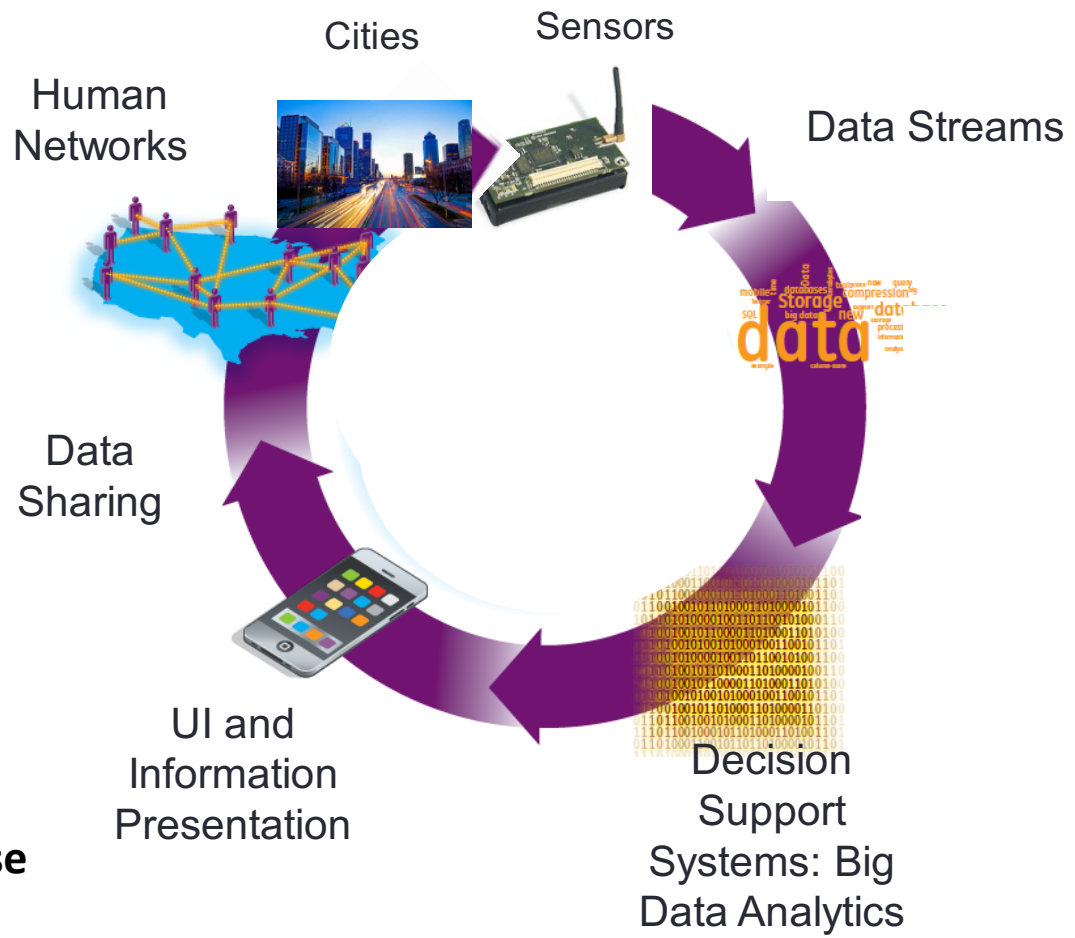


Smart Grid

Smart Transportation



Smart Emergency Response



Big Data Era

Google: "Every 2 Days We Create As Much Info As We Did Up To 2003"

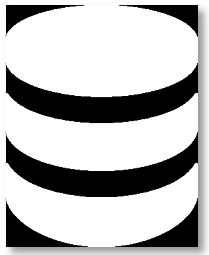
Facebook: "500+ TB of New Data Each Day"

Twitter: "500 million tweets per day."

Source: TechCrunch, August 2010, http://techcrunch.com/2010/08/04/schmidt-data/

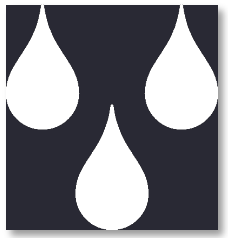
Source: Gigaom.com, August 2012, http://bit.ly/13wyPj7

Source: Twitter Blog, August 2013, http://bit.ly/14ySYrr



Volume

Velocity



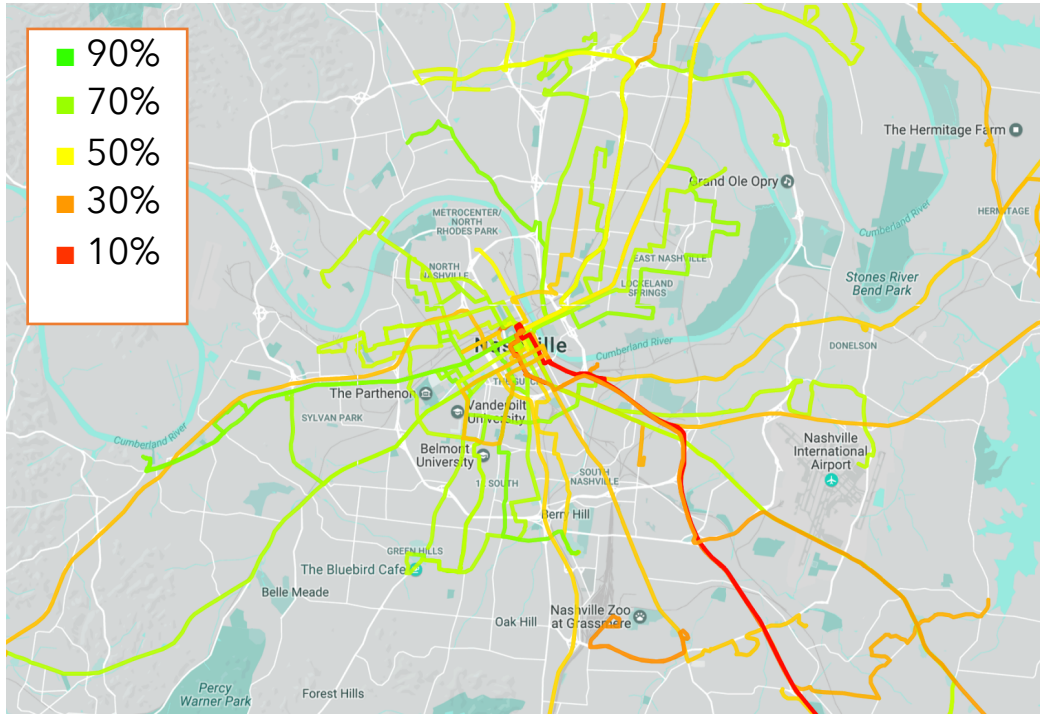
Variety

Data has value

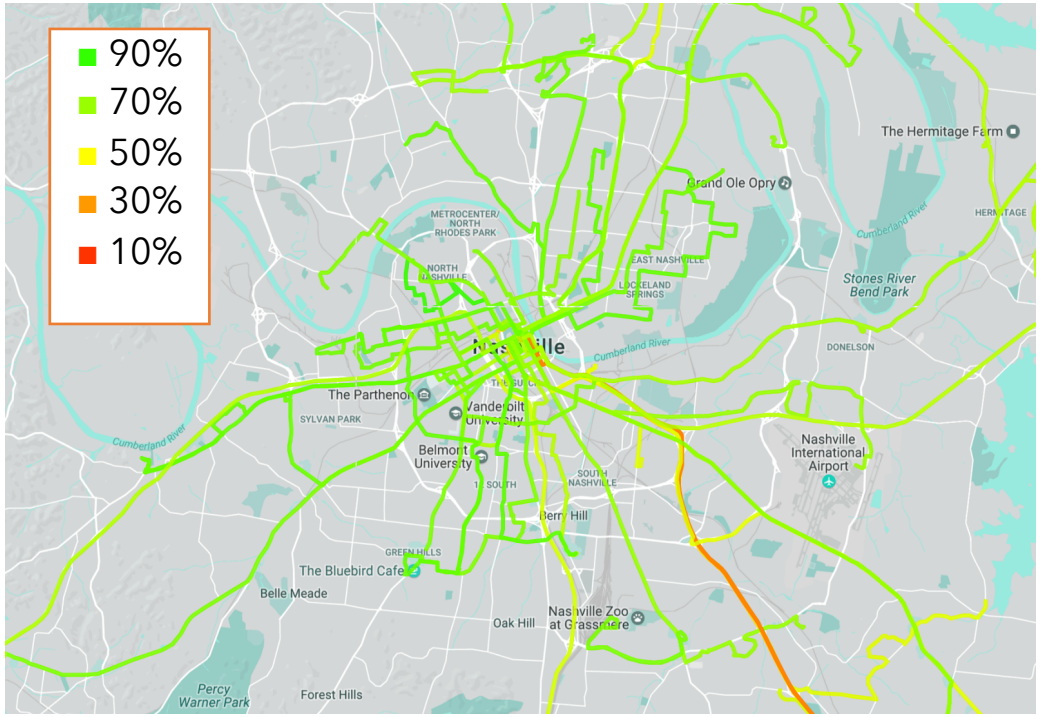
http://newsbytes.ph/2015/10/15/gartner-customer-data-has-monetary-value-but-many-firms-ignore-it/

Decision Making

Public Transit Improvement

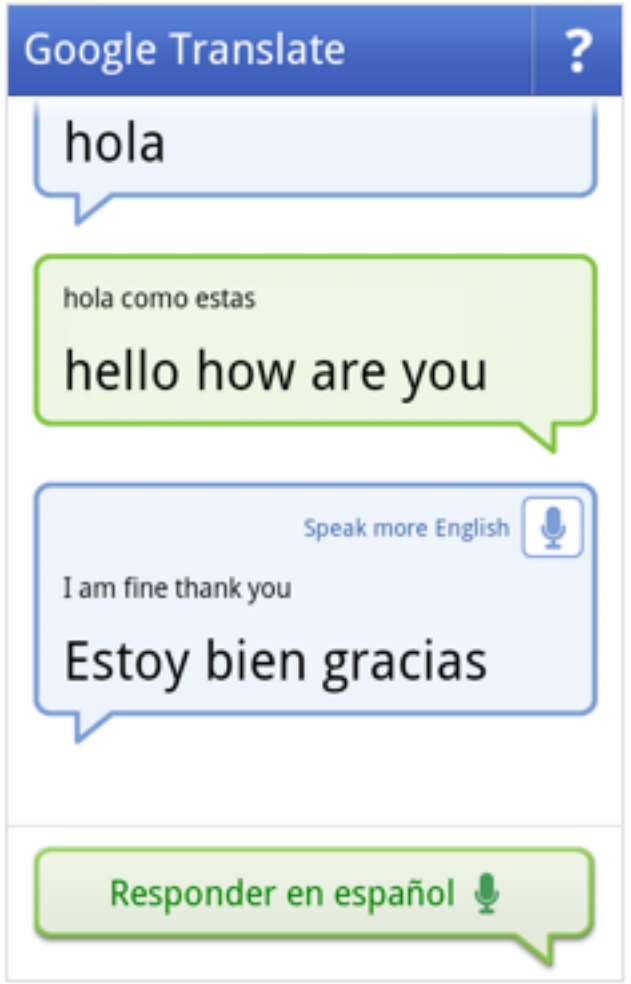


Original



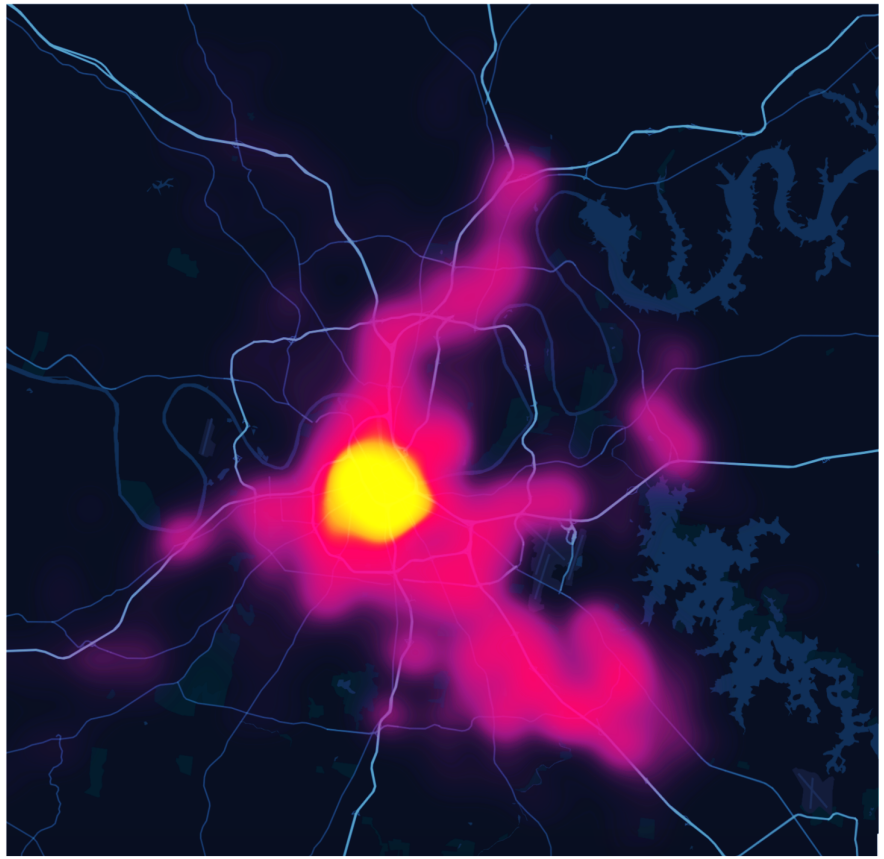
Optimized (in Simulation)

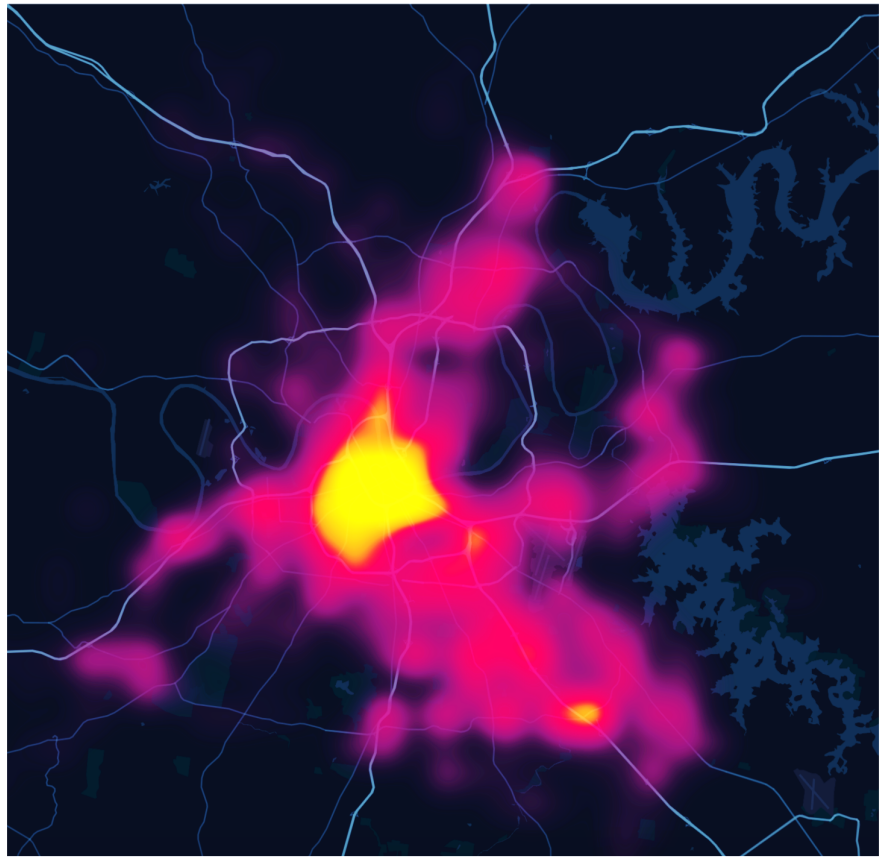
Language Translation



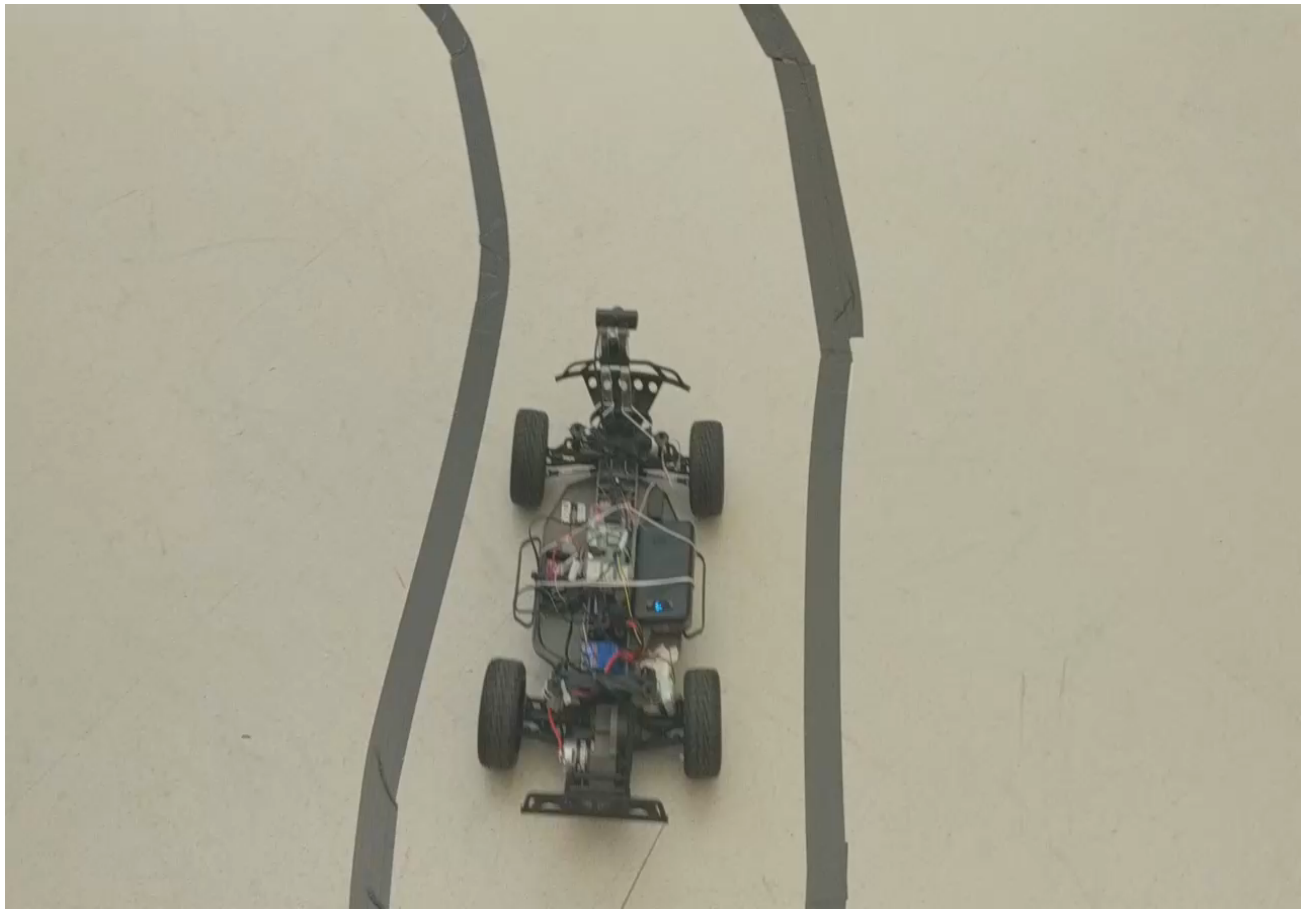
Public Safety

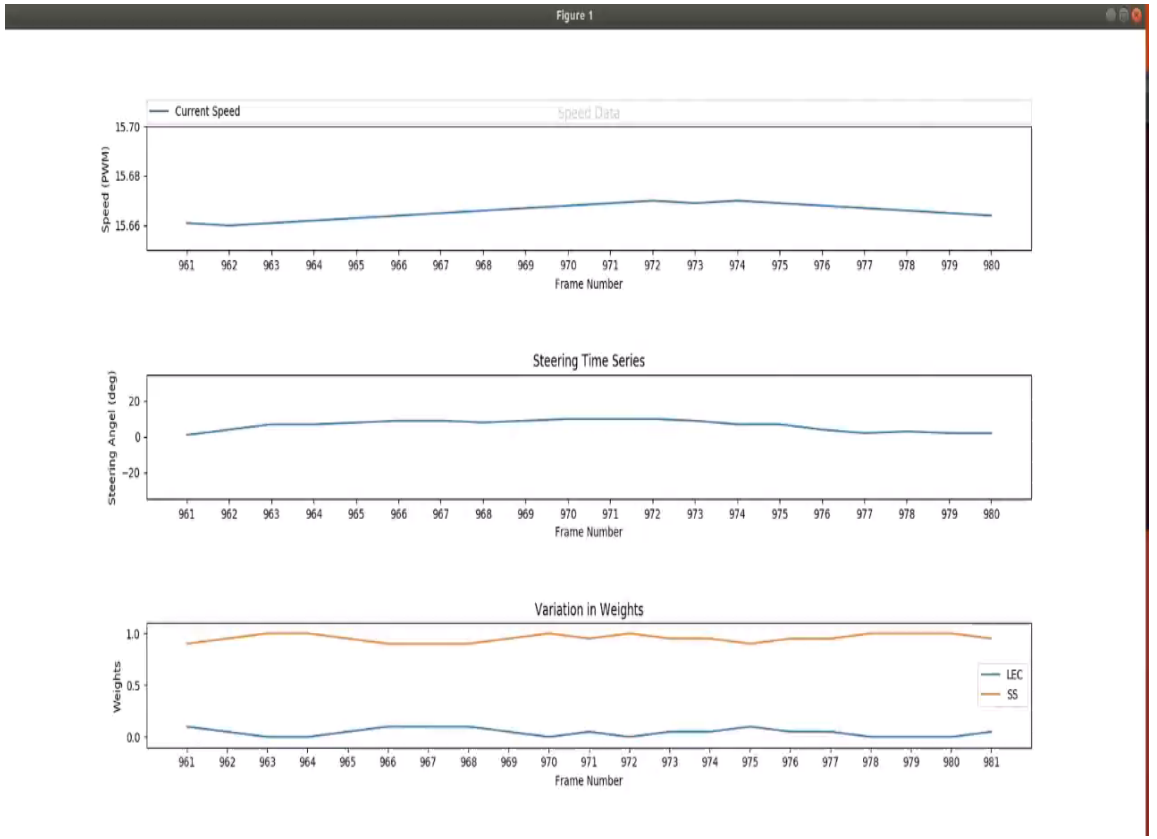
- Comparison of (1) incidents predicted by model (left), and (2) real incident distribution (right) over January 2019





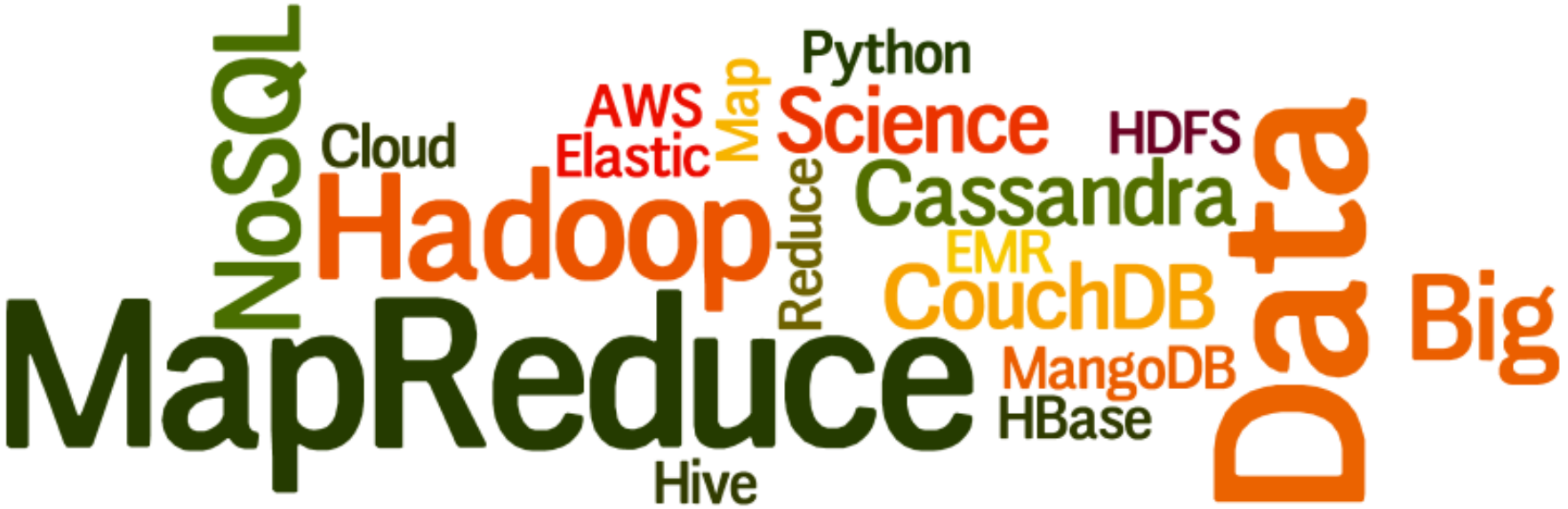
Self-Driving Cars





SO WHAT IS BIG DATA

The many words of Big Data



Glorified Plumbing

Processing
Storing data
Supporting ML

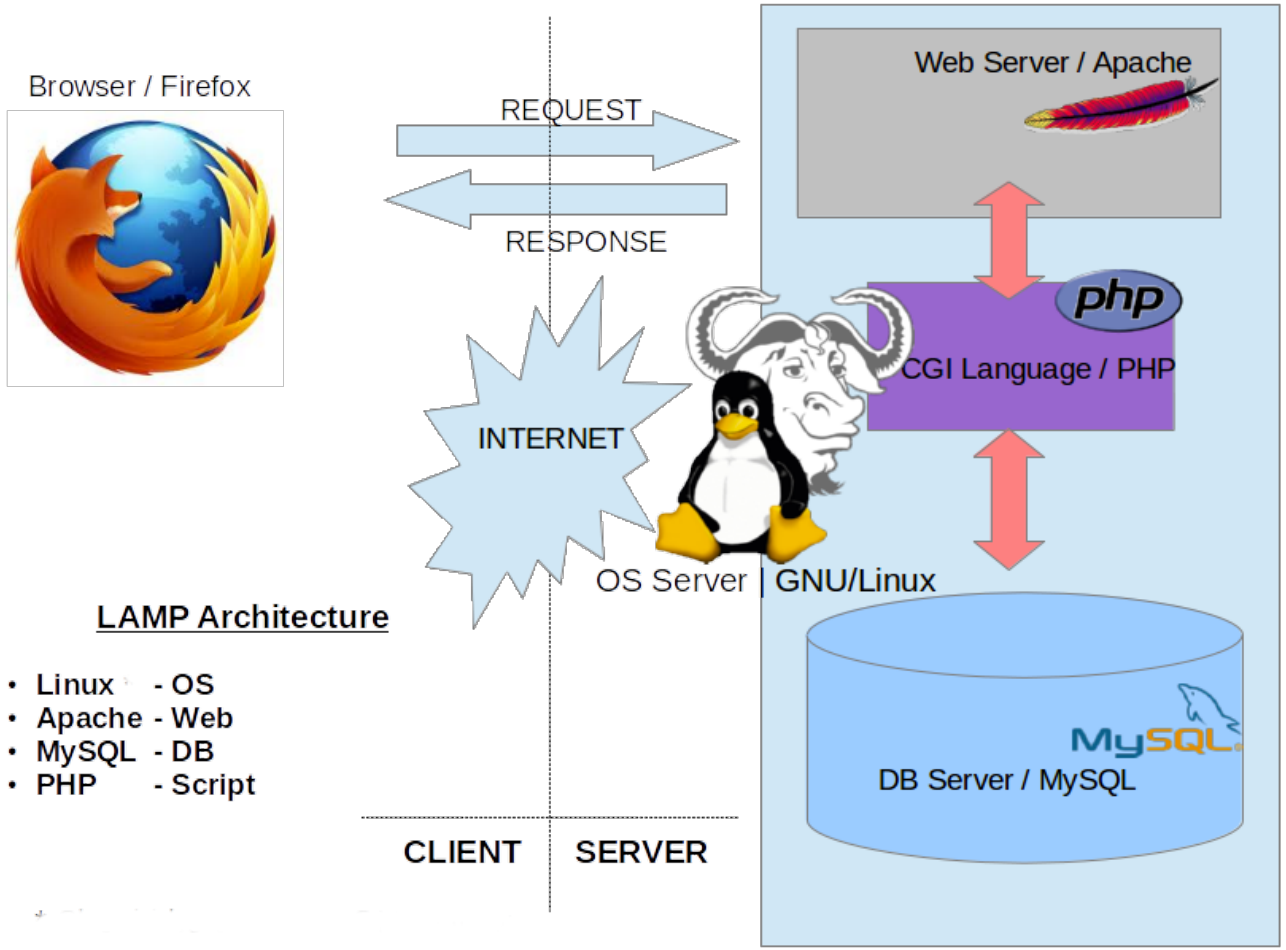
Example: requirements of a Smart City Data Store

- Must handle high Velocity/Volume data.
- Efficient mechanisms for querying and processing geospatial data.
- Real time processing is a requirement.
- Resilient.
- Data has high Variety – MDS, HERE, custom data models.
- Extendable – as new applications are added, system should be able to add new data models without taking system off-line.

Traditional Approach: Client-Server Model

1. User triggers communication with server through web browser – makes a **request.**
2. Web server includes multiple processes awaiting requests. When request is received – routes the request to resource/application logic to process request.
3. Application logic communicates with data model to query, access or update data as needed.

Traditional Approach



Ref: https://en.wikipedia.org/wiki/LAMP_(software_bundle)

VOLUME – THE BIG BOTTLENECK

System Demands

- Financial applications - User x needs to access very specific data related to that person.
- Twitter – request stream from person y, need to find specific data (tweets) tagged for that person.

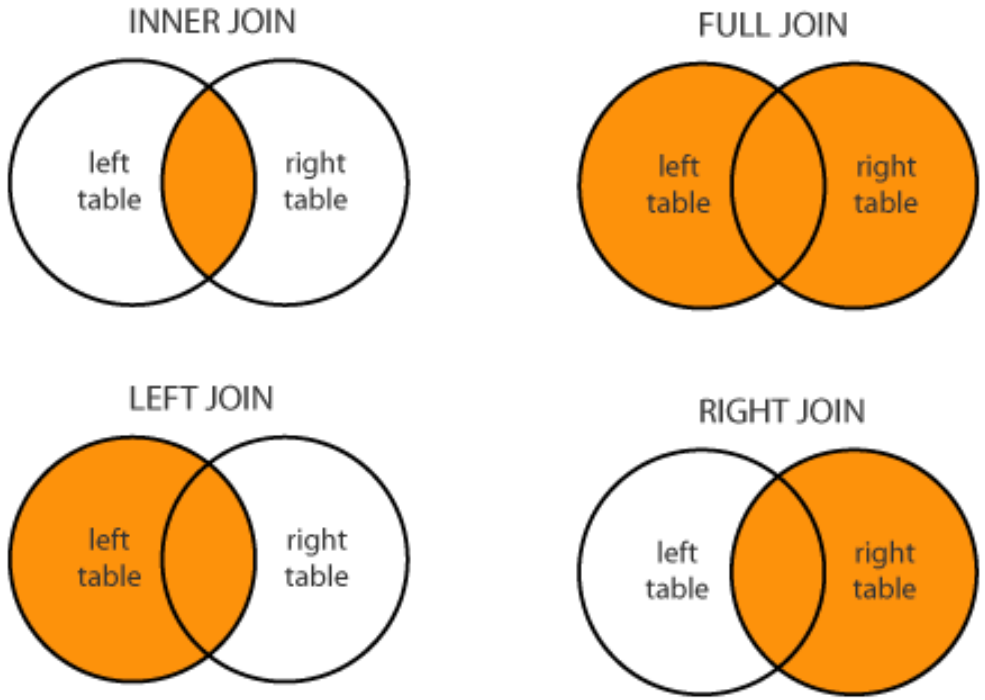
Key Question: How do we efficiently find record 4 in a data store of millions of records?

SQL Databases

- SQL databases arose to efficiently store structured data.
- While structured? Structured data provides a predictable format that can be tailored to specific applications.
- Properly structured data can fit seamlessly into application logic.

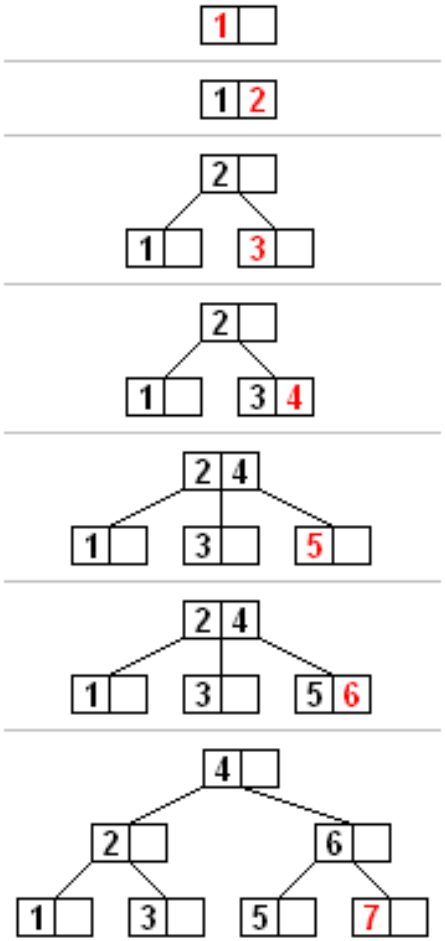
SQL Data Model

- Rooted in relational logic.
- Support complex data retrieval – joins.
- Find individual records, combine result and return to application.



Indexing

- To help retrieve data efficiently, fields can be indexed.
- B-tree is most common.
- Efficient for both finding individual elements (traditional web applications) and aggregation queries (batch processing discussed later …)

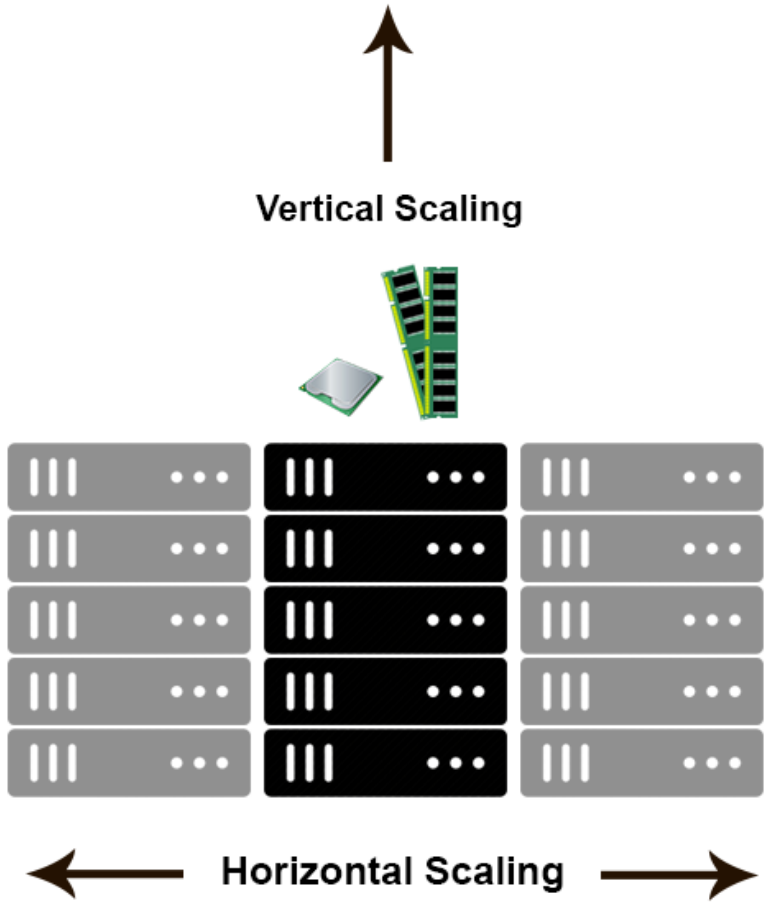


B-tree insertion

Volume – Storage Demand

- Terabytes (Petabytes) of new data.
- Not possible to store all data in one node - vertical scaling not a viable solution.

Requirement: data store must scale horizontally. Data must be stored across multiple nodes.



Ref: https://medium.com/@abhinavkorpal/scaling-horizontally-and-vertically-for-databases-a2aef778610c

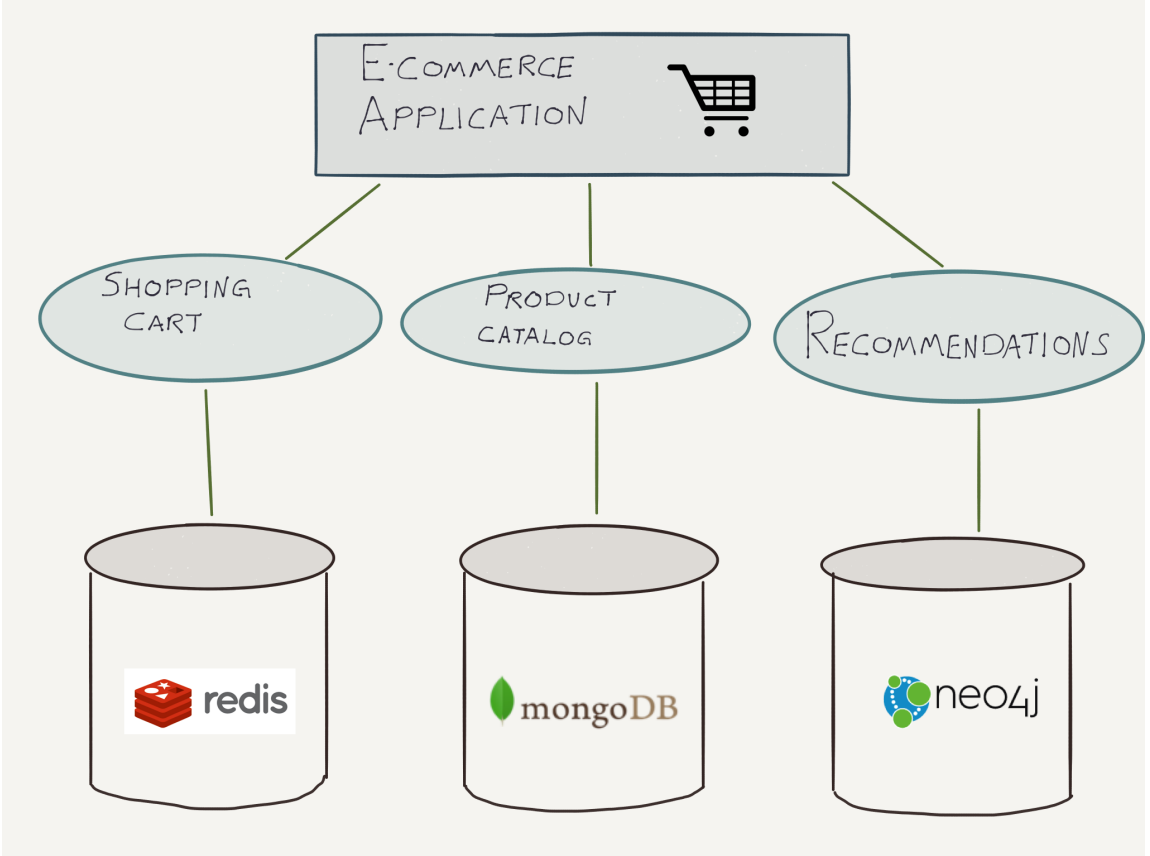
Handling Volume - NoSQL

- SQL built for vertical scaling.
- Limitations led to revisiting how we structure data.
- NoSQL: a general term for database systems not based on SQL
- Common models:
  - Key-value -> fast lookup, great for in memory caches and storage (DynamoDB, Memcached, Redis).
  - Document models -> stores data as “records”, typically JSON. Great for unstructured data (MongoDB, CouchDB). Easy sharding across nodes.
  - Graph models -> stores data as a mathematical graph structure, great for performing traversal queries and working with highly inter-connected data.
- Improved **scalability**, but **NOT** ACID
- Coding complexity passed to developer

Ref: https://neo4j.com/blog/neo4j-doc-manager-polyglot-persistence-mongodb/

Example

Multiple data models can be used in the same application where each model is selected as optimal solution to a specific objective.



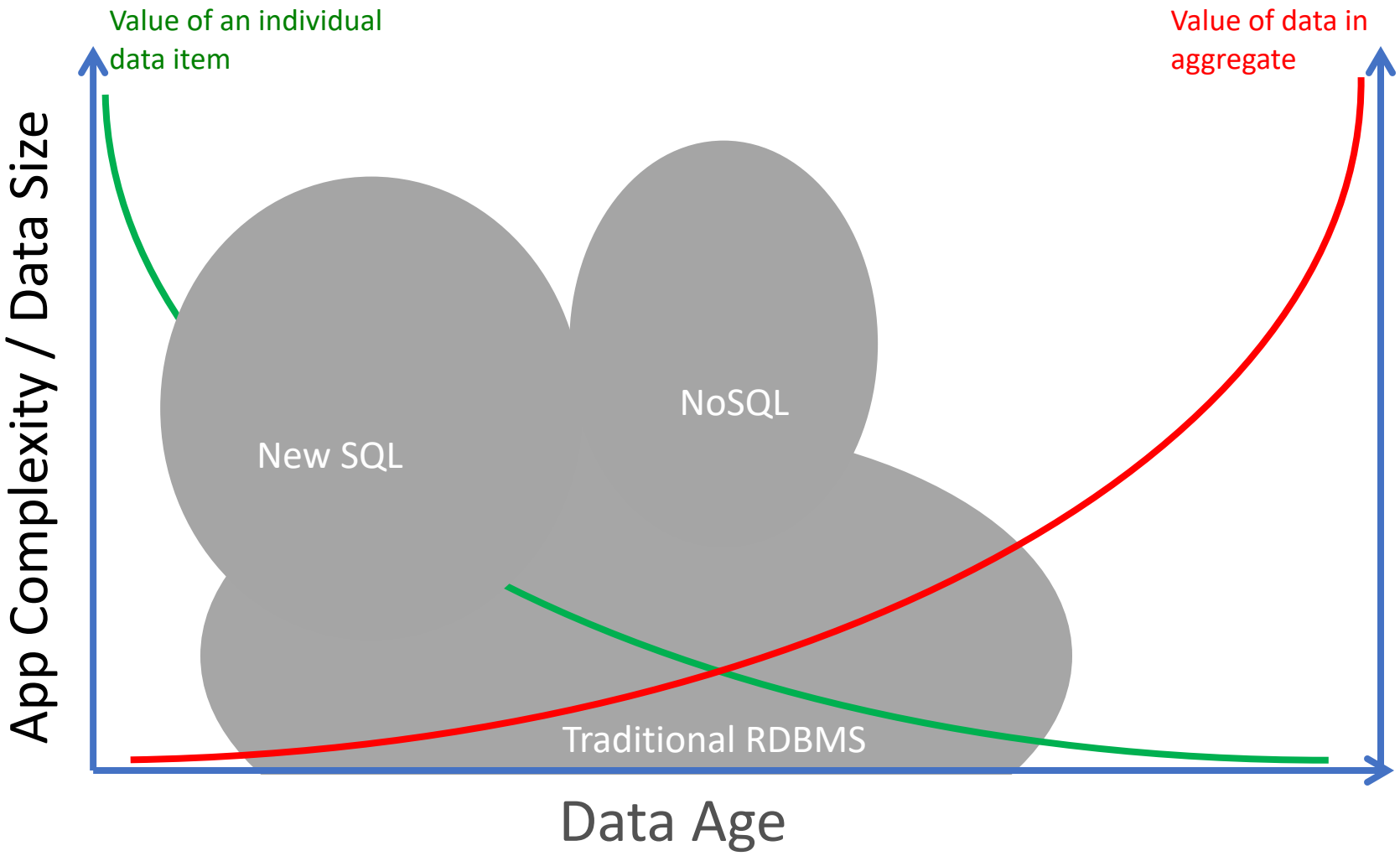
Ref: https://neo4j.com/blog/neo4j-doc-manager-polyglot-persistence-mongodb/

NewSQL DBS

- Scalability with ACID (*for some types of queries)*
- Typically used for HUGE databases spanning data centers
- Slower than simpler NoSQL / RDBMs
- May require atomic clocks…see Google's Spanner.

Google Spanner

The Big Data Storage Landscape



Other axes to consider: Response Time

PROCESSING – THE SECOND BOTTLENECK

Big Data Problem

- Identifying trending topics on Twitter

- Objective:
  - Find most frequently occurring hash tag today

Relational Databases (RDBMs)

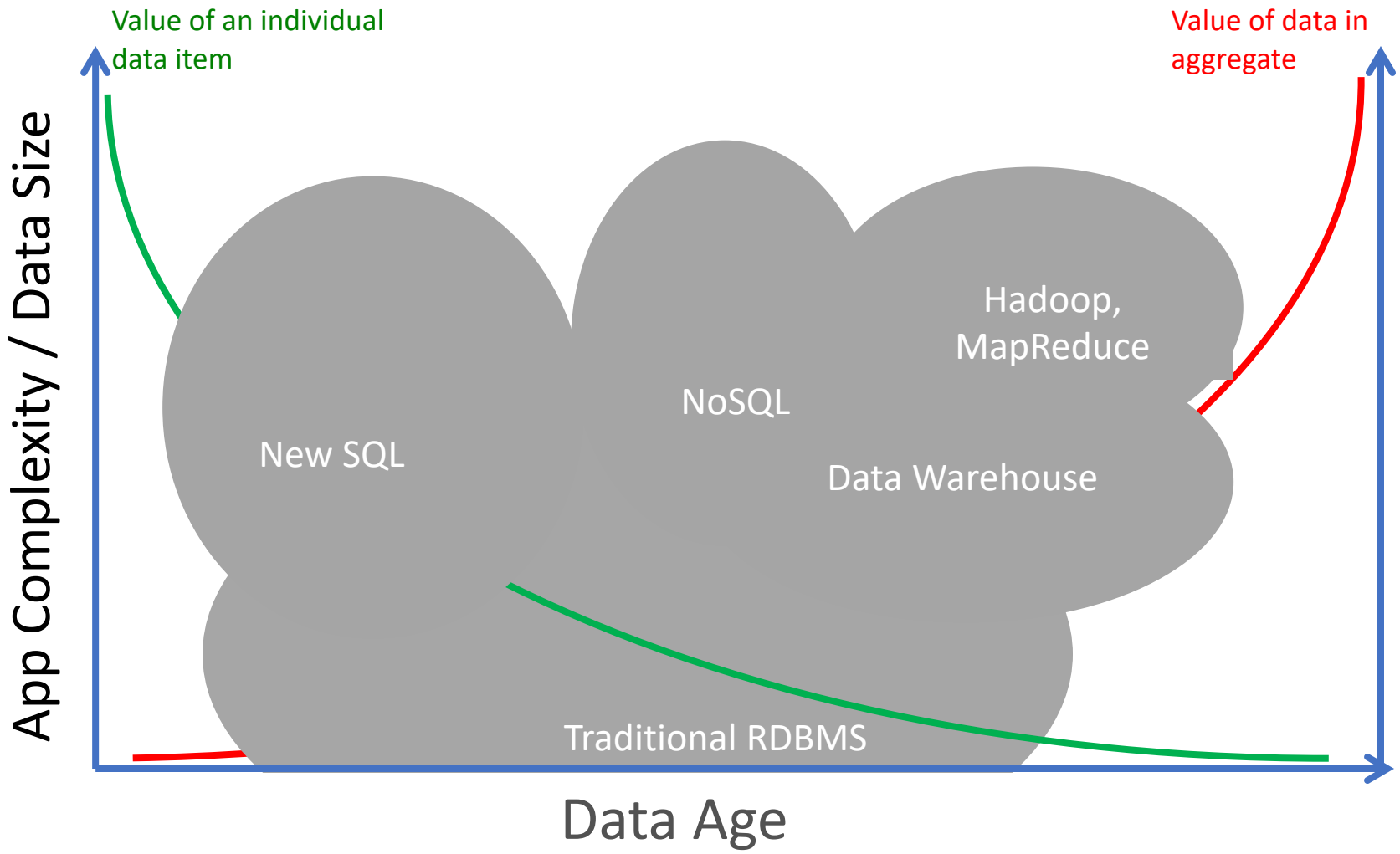
- **SQL**: ‘Easy’ to express complex queries
- Designed to scale **horizontally** (partition data by location)
- **Schema** makes it easier to understand data structure

```
SELECT get_first_hash_tag(tweet), count(*)
FROM tweets
WHERE tweet_dt = TODAY()
GROUP BY get_first_hash_tag(tweet)
```

Big Data Processing

- Services (online systems) – traditional model. Service waits for request or instruction from client. Main requirement is find and handle small subsets of data quickly.
- Batch processing (offline systems) – Takes a large amount of data and runs a long-running intensive process. Processes are scheduled periodically.
- Stream processing (near-real-time systems) – Like batch processing, it consumes inputs and produces outputs. However operates on data and events as they come in.

The Big Data Storage Landscape



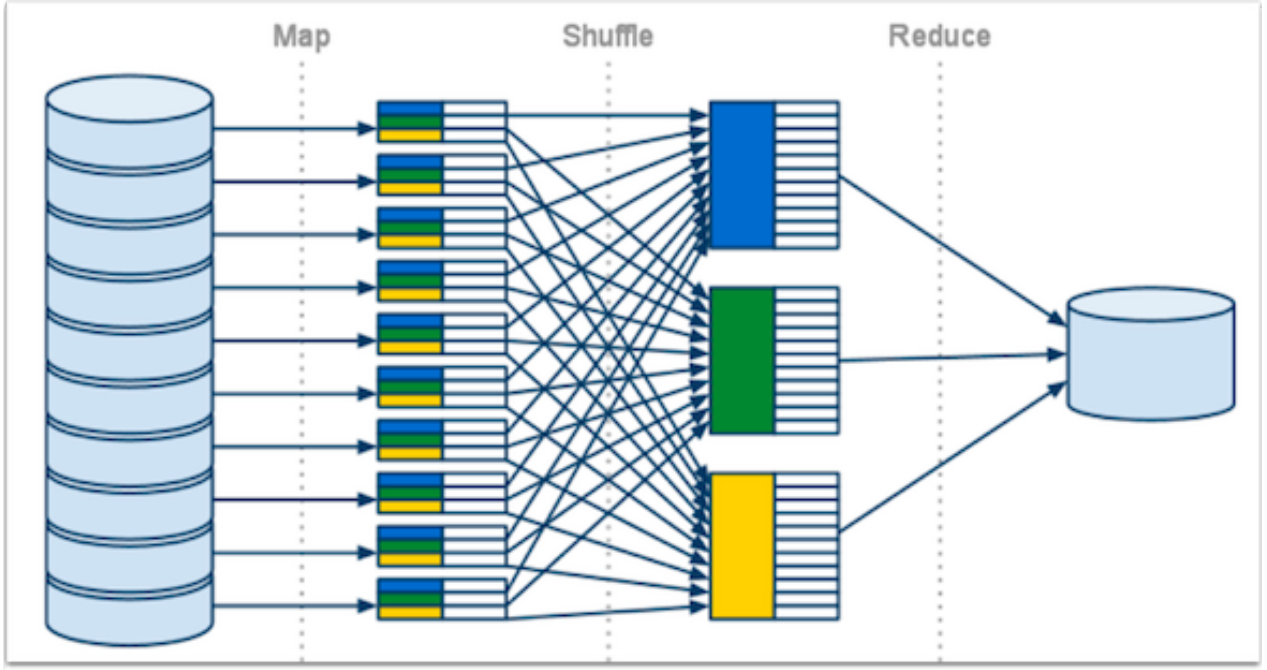
Other axes to consider: Response Time

Batch Processing

- Jobs are periodically scheduled to run on large batches of data.
- Outputs aggregated analysis.
- Common for data analysts, Machine learning …
- Examples – bank statements for all users at end of the month, aggregate monitoring statistics for IoT applications.

Batch Processing - MapReduce

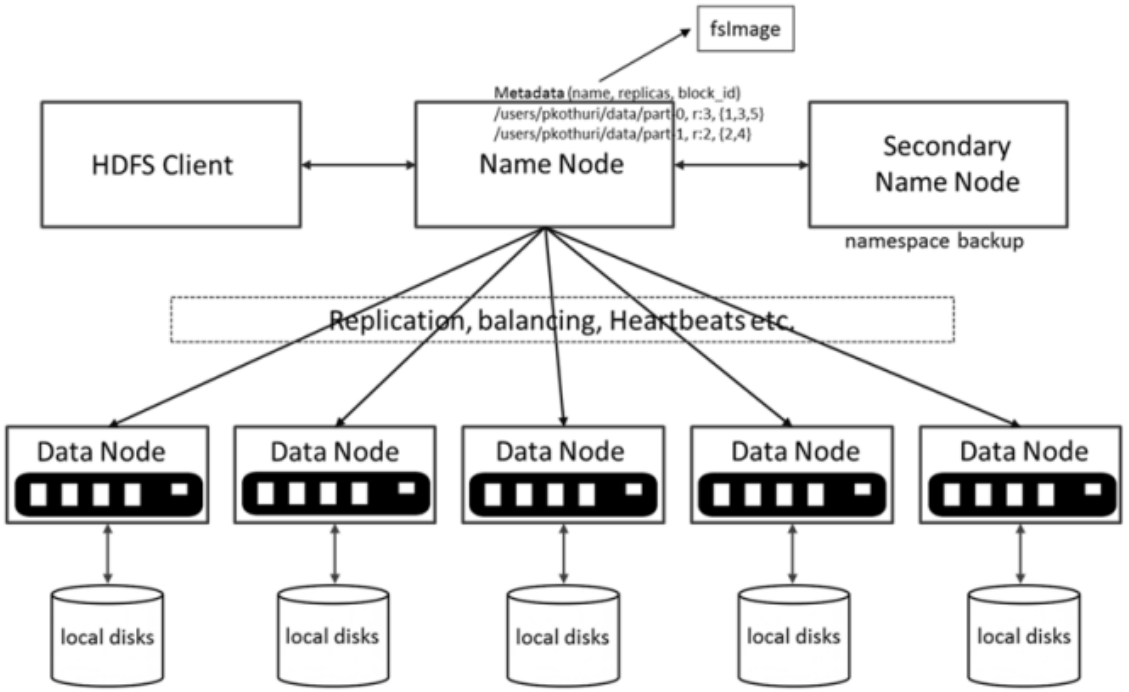
- Originally created by Google for parallel processing large batch jobs.
- Data is split, mapping functions are applied to each subset and finally the processed outputs are combined and returned.
- Computation is done in parallel



1) Map Step: <date_time, hash_tag>, 2) Reduce Step: Find most frequent

Distributed File Systems (HDFS)

- MapReduce uses HDFS to store data as blocks of flat files. MapReduce jobs and processes are mapped over the nodes, file system closely resembles the MapReduce algorithm itself



HDFS

Stream Processing

- Near real-time processing.
- Apply lightweight functions to transform data as it arrives.
- Spark is a common processing framework for real-real time streaming applications.