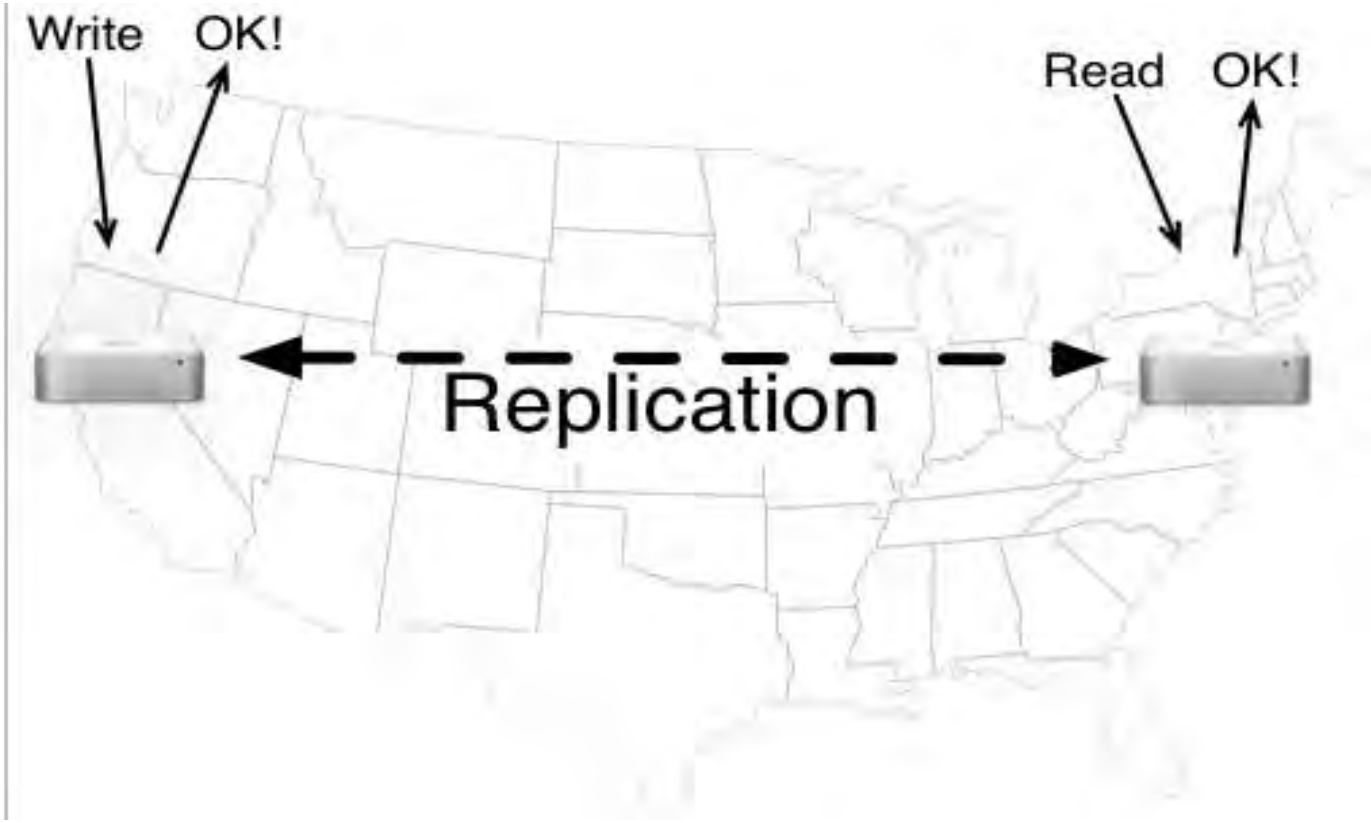
Consistency vs. Availability in a Partitioned System



Choice may depend on context:
Bank Balance (consistency) vs Facebook Likes (availability)

NoSQL

NoSQL Data Stores

- Flexible with consistency vs availability

NoSQL Types

- Key/Value
- Document
- Columnar
- Graph
- RDF (Sparql)

Key-value Stores

Key-value stores are probably the simplest form of database management systems. They can only store pairs of keys and values, as well as retrieve values when a key is known.

These simple systems are normally not adequate for complex applications. On the other hand, it is exactly this simplicity, that makes such systems attractive in certain circumstances. For example resource-efficient key-value stores are often applied in embedded systems or as high performance in-process databases.

key	value
firstName	Bugs
lastName	Bunny
location	Earth

Amazon S3

Amazon S3 is a simple key, value store designed to store as many objects as you want. You store these objects in one or more buckets. An object consists of the following:

- Key
- Version ID
- Value
- Metadata
- Subresources
- Access Control Information

A bucket is a container for objects stored in Amazon S3. Every object is contained in a bucket. For example, if the object named photos/puppy.jpg is stored in the johnsmith bucket, then it is addressable using the URL http://johnsmith.s3.amazonaws.com/photos/puppy.jpg

What is the value?

Document Stores

Document Stores

Document stores, also called document-oriented database systems, are characterized by their schema-free organization of data.

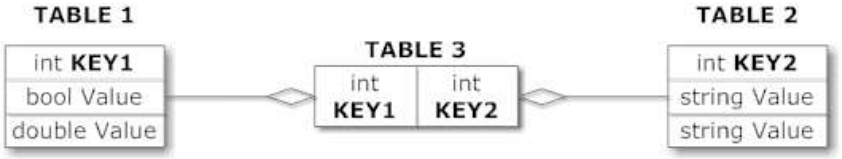
That means:

- Records do not need to have a uniform structure, i.e. different records may have different columns.
- The types of the values of individual columns can be different for each record.
- Columns can have more than one value (arrays).
- Records can have a nested structure.

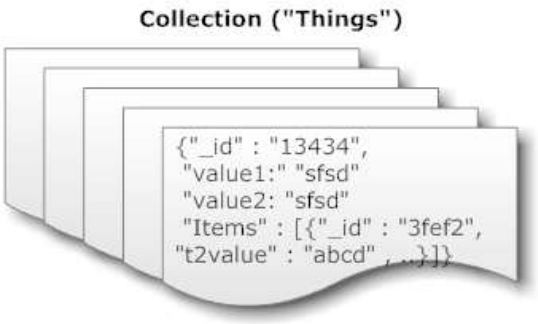
Document stores often use internal notations, which can be processed directly in applications, mostly JSON. JSON documents of course can also be stored as pure text in key-value stores or relational database systems. That would, however, require client-side processing of the structures, which has the disadvantage that the features offered by document stores (such as secondary indexes) are not available.

Document Stores

Relational Model



Document Model




```
{
  person: {
    first_name: "Peter",
    last_name: "Peterson",
    addresses: [
      {street: "123 Peter St"},
      {street: "504 Not Peter St"}
    ],
  }
}
```


TMIP

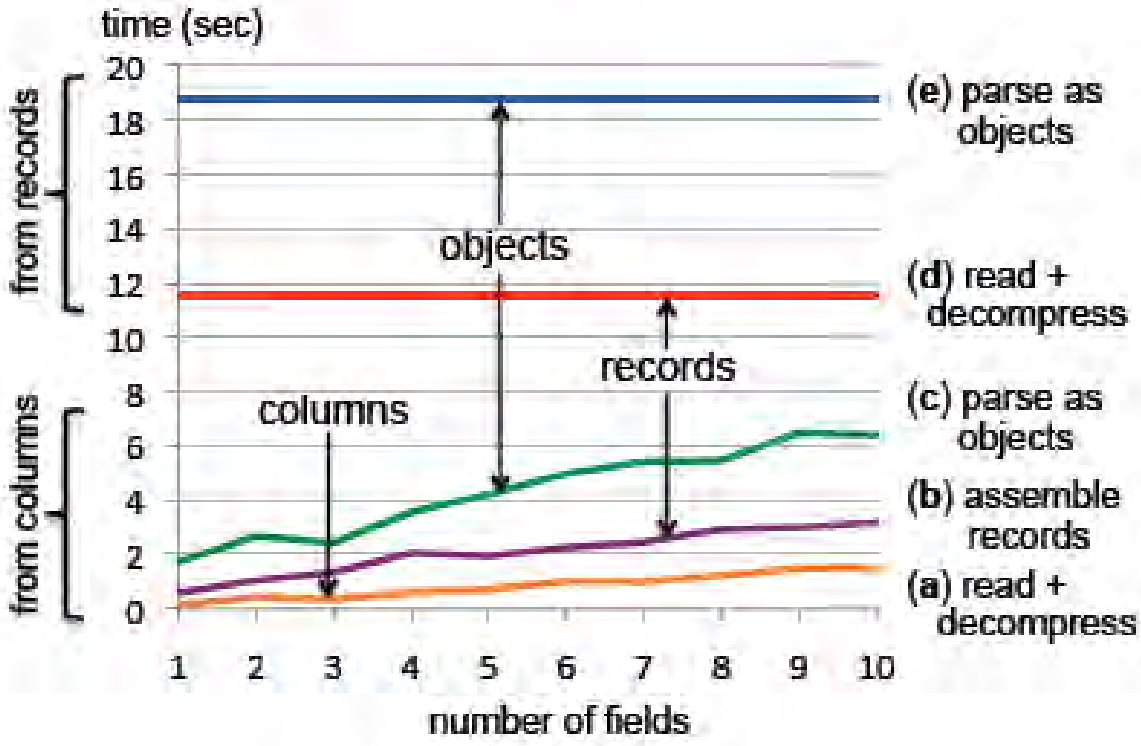
Databases and IO Modes/Models

Speed is Key

Leveraging ar
execution eng
Apache Drill is
optimization,
throughout nc

Liberat

Perform interact
nested and sche
many different s
Cassandra and
included as a sp

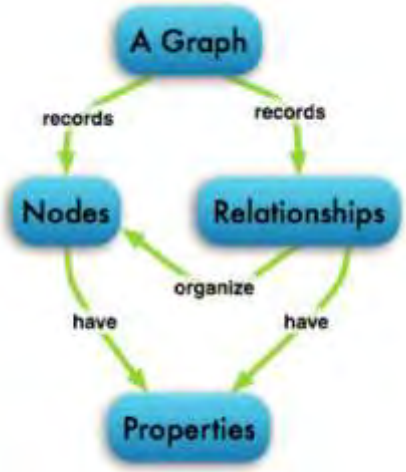


Graph

Graph DBMS

Graph DBMS, also called graph-oriented DBMS or graph database, represent data in graph structures as nodes and edges, which are relationships between nodes. They allow easy processing of data in that form, and simple calculation of specific properties of the graph, such as the number of steps needed to get from one node to another node.

Graph DBMSs usually don't provide indexes on all nodes, direct access to nodes based on attribute values is not possible in these cases.



Big Data Infrastructure

MODELS: STORAGE

Hadoop

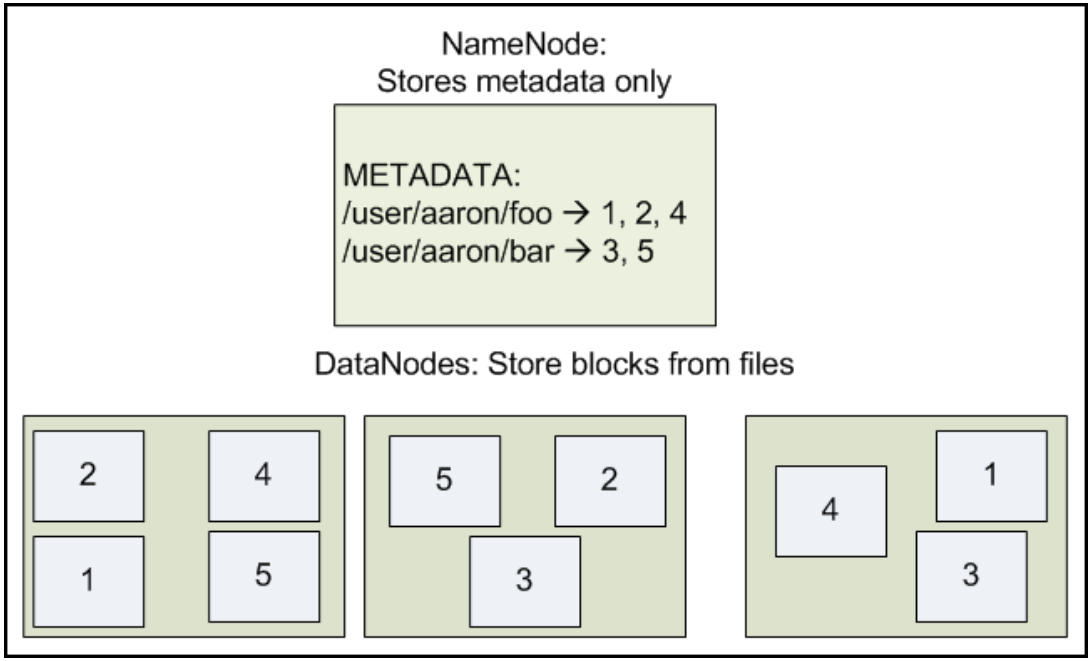
Efficient, automatic distribution of data and work across machines

Moving computation to the data

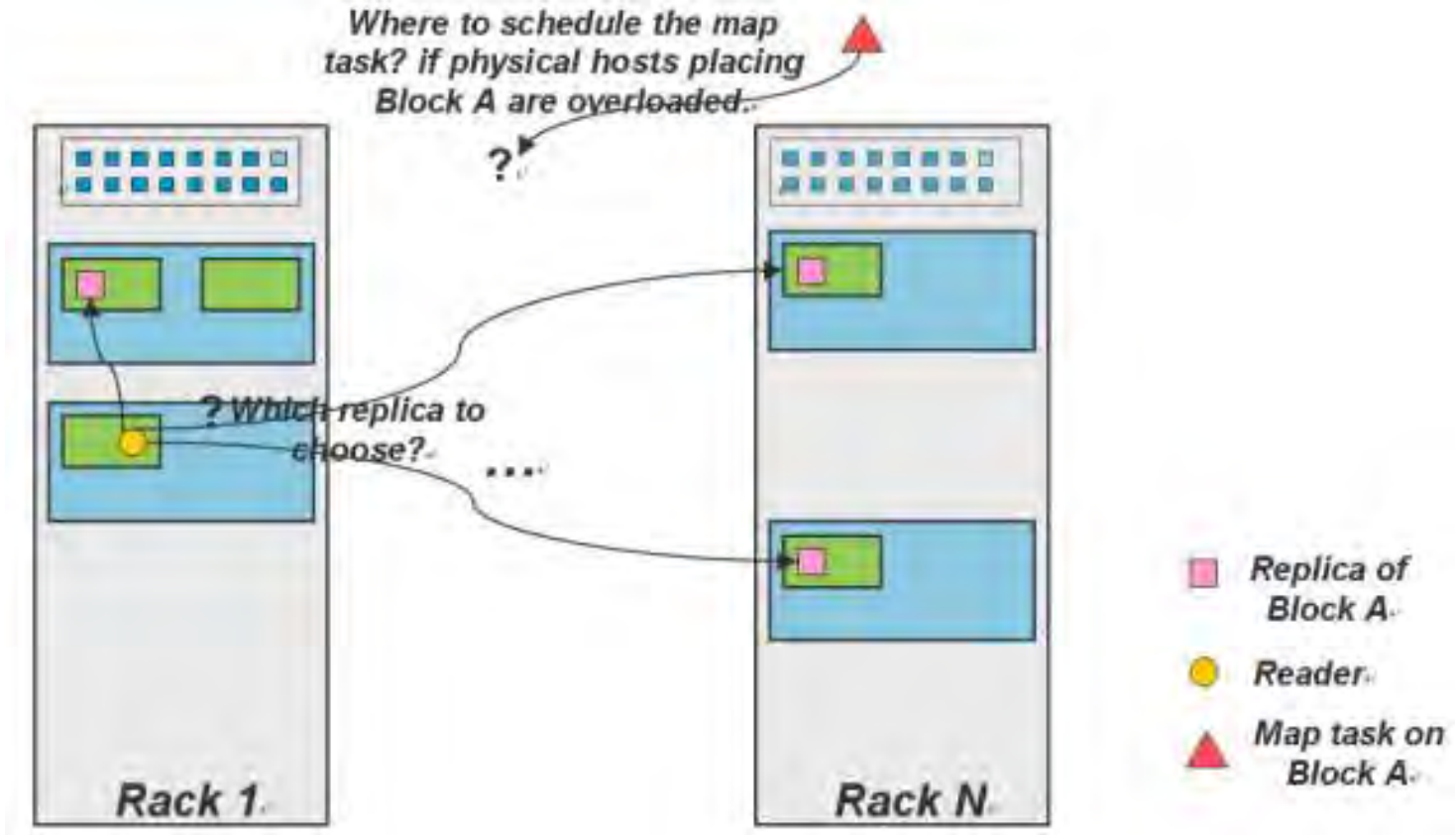
- Data locality
- Network bandwidth is a precious resource in the data center
- Try to run computation task on node where data resides
  - Data local
  - Rack local
  - Off rack

HDFS

Applications that use HDFS are assumed to perform long sequential streaming reads from files. HDFS is optimized to provide streaming read performance; this comes at the expense of random seek times to arbitrary positions in files.



Data Locality



https://github.com/vmware-bdc/hadoop-common-topology/wiki

HDFS Is Good For…

Storing Large Files

- Terabytes and greater
- Millions of files (rather than billions)

Streaming Data

- Write once, read many times
- Optimized for streaming rather than random reads

"Cheap" Commodity Hardware

- Ideal for less reliable computers; no need for super-computers

HDFS Is Not So Good For…

Low-latency Reads

- High-throughput rather than low latency for small chunks of data

Large Amount of Small Files

- Better for millions of large files instead of billions of small files

Multiple Writers

- Single writer per file
- Writes only at the end of file

Webinar Roadmap

Big Data definitions
Infrastructure
- Metal
- Management
- Models

(part 1)
(part 2)

Algorithms
- Data Processing
- Machine Learning

Session 2

Implications
- Privacy
- Smart Cities
- Technology Tradeoffs

Big Data Algorithms

BACKGROUND

What is an Algorithm?

Algorithm

- A set of steps or rules to accomplish a task
- Example:
  - Add 2 scoops of raisins to a box of bran

Heuristic

- A set of guidelines to accomplish a task
- Example:
  - Add some raisins to bran; taste it; add more raisins if needed

Model

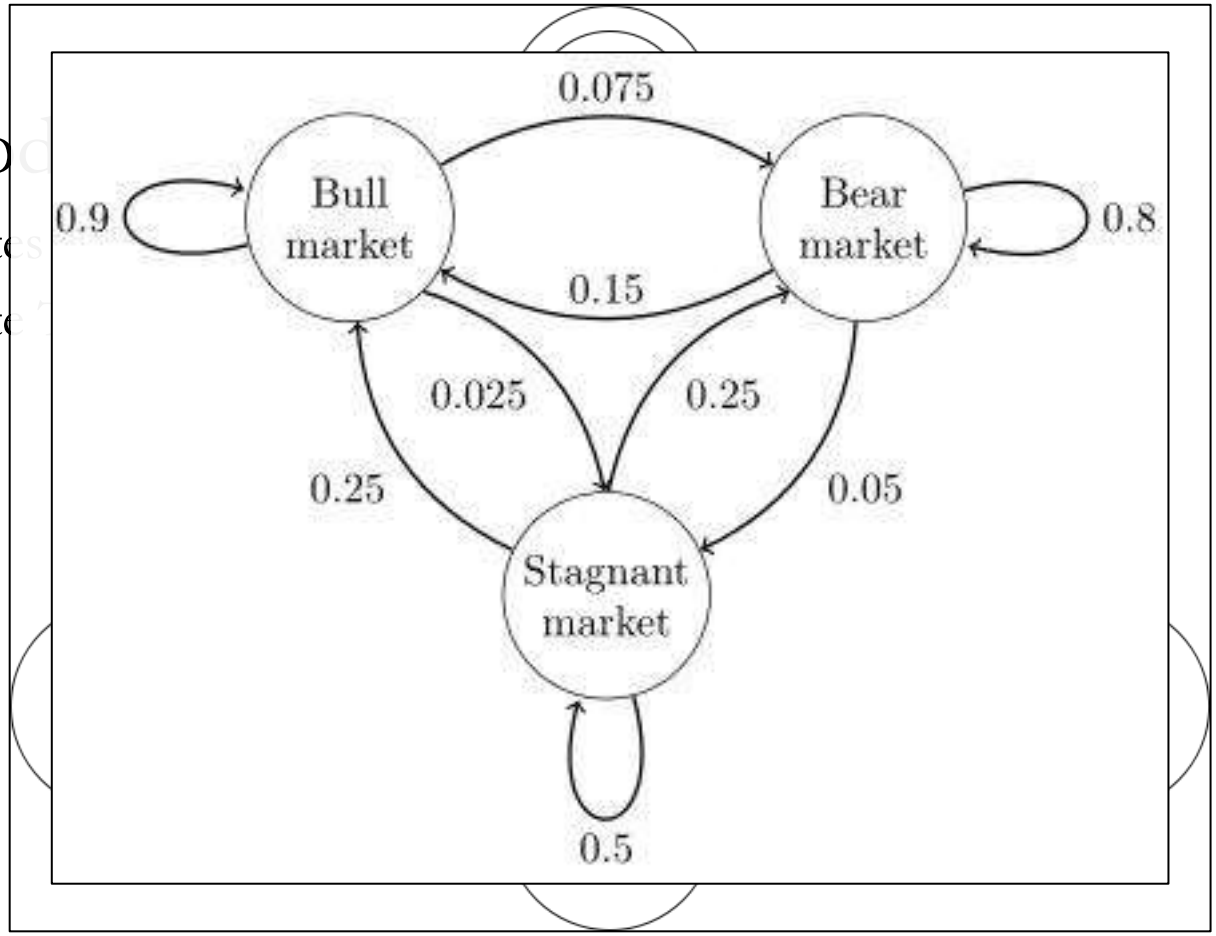
- An abstract description of a system
- Example:
  - Raisin Bran consists of 2 parts raisins to 5 parts bran

Example: Markov Model

Markov Mo

- Set of State
- Set of State

Algorithm?
Heuristic?
Model?



TMIP

What is an Algorithm?

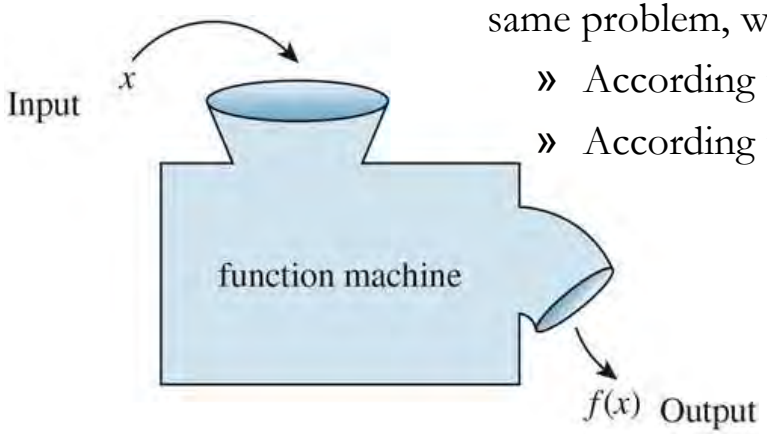
Greatest Common Denominator (GCD)

72	30	Replace 72 by the difference (72 - 30)
42	30	Replace 42 by the difference (42 - 30)
12	30	Replace 30 by the difference (30 - 12)
12	18	Replace 18 by the difference (18 - 12)
12	6	Replace 12 by the difference (12 - 6)
6	6	**BINGO!!!**

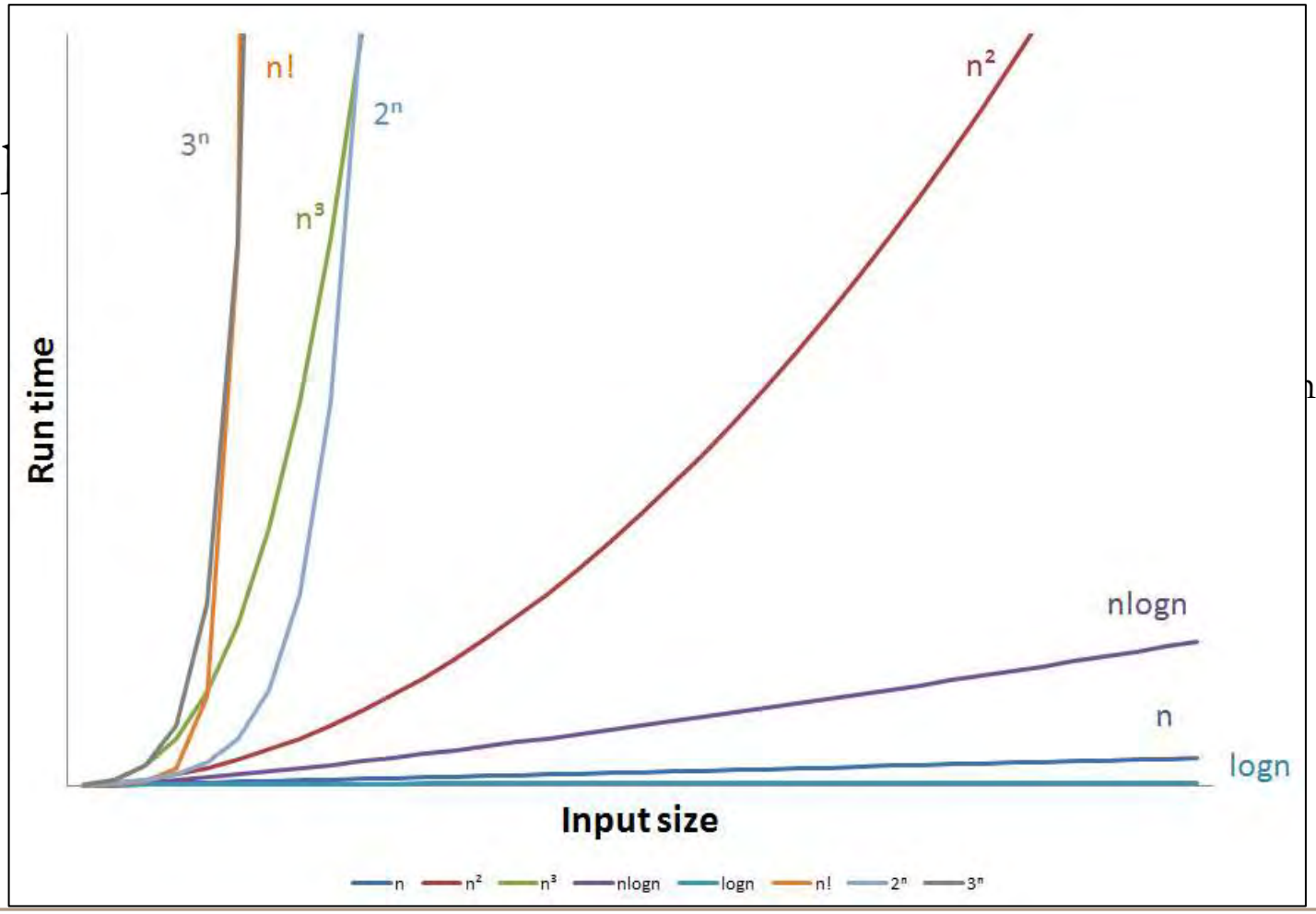
What is an Algorithm?

A Reasonable Definition of Algorithm

- Donald Knuth: "Father of Algorithm Analysis"
  - An algorithm satisfies *finiteness, definiteness, input, output, effectiveness*
  - Goodness of Algorithms
    - Given two algorithms that solve the same problem, which one is *best*?
      - » According to memory?
      - » According to speed?



Algorithmic Complexity



Algorithmic Complexity

Example

- Comp
  - V
- Comp
  - V
- Linea
  - Sc
- Binar
  - Sc

n (list size)	Computer A run-time (in nanoseconds)	Computer B run-time (in nanoseconds)
15	7	100,000
65	32	150,000
250	125	200,000
1,000	500	250,000
...	...	...
1,000,000	500,000	500,000
4,000,000	2,000,000	550,000
16,000,000	8,000,000	600,000
...	...	...
$63{,}072 \times 10^{12}$	$31{,}536 \times 10^{12}$ ns, or 1 year	1,375,000 ns, or 1.375 milliseconds

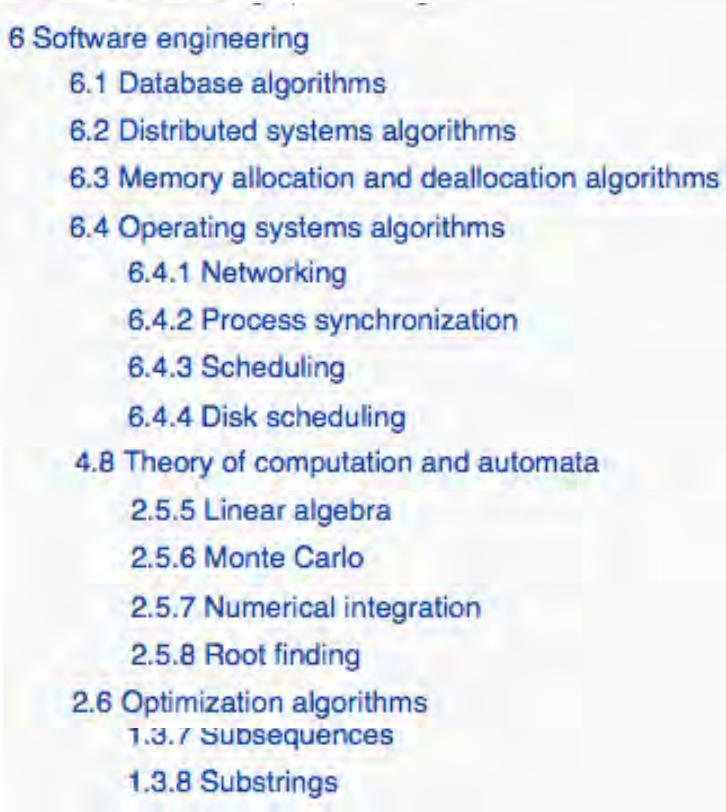
Choosing Algorithms To Use

The List of Algorithms Is Endless

- As many algorithms as recipes
- There is no canonical organization

Common Data Science Algorithms

- Schutt & O'Neil
  - Data Processing algorithms
  - Machine Learning algorithms
  - Optimization algorithms



http://en.wikipedia.org/wiki/List_of_algorithms

TMIP

Webinar Roadmap

Big Data definitions
Infrastructure
- Metal
- Management
- Models

(part 1)

(part 2)

Algorithms

Data Processing

Machine Learning

Session 2

Implications
- Privacy
- Smart Cities
- Technology Tradeoffs

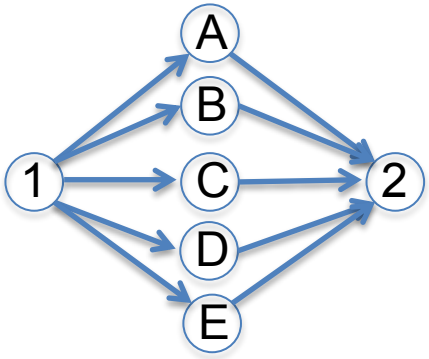
Data Processing Algorithms

The MapReduce, Pregel, etc. Algorithms

- Think of them as Distributed MoCs

Model of Computation (MoC)

- A characterization of computation & communication
  - MapReduce – A particular distributed MoC



Each node executes code

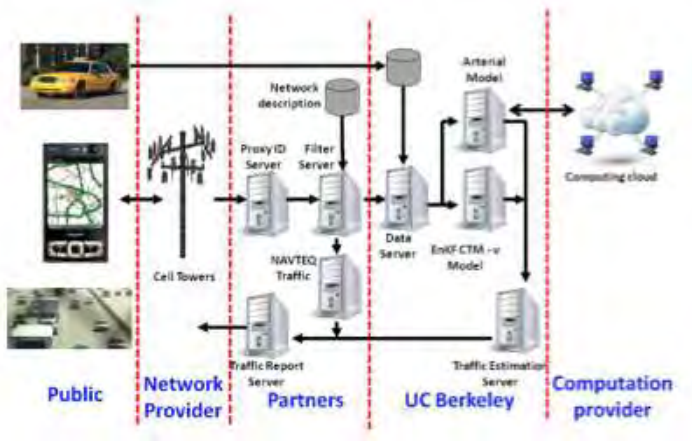
A set of nodes execute an algorithm

Machine Learning Algorithm Preview: Traffic Estimation

Scaling the Mobile Millennium System in the Cloud

Timothy Hunter, Teodor Moldovan, Matei Zaharia, Samy Merzgui, Justin Ma,
Michael J. Franklin, Pieter Abbeel, Alexandre M. Bayen
University of California, Berkeley

"Mobile Millennium uses machine learning to infer traffic conditions for large metropolitan areas from crowdsourced data…evaluates probabilistic distribution of travel times over the road links"



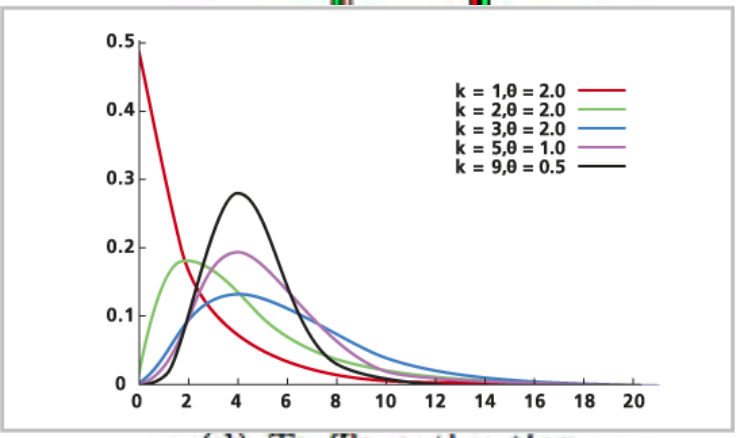
San Francisco Scale: 25,000 road links - 500,000 data points (taxis)

Timelier predictions (fast), larger road networks (scalable), more accurate models (granular)

TMIP

Estimation Process

Goal: infer how congested the links are in an arterial road network, given periodic GPS readings from vehicles moving through the network.



(d) Traffic estimation algorithm

Model: graph (V, E) where V are road intersections and E are streets

Algorithm Inputs: Graph + Observed Trajectories

Problem: Travel times for all links in trajectory, not individual links

Solution: Iterative Expectation Maximization (EM)

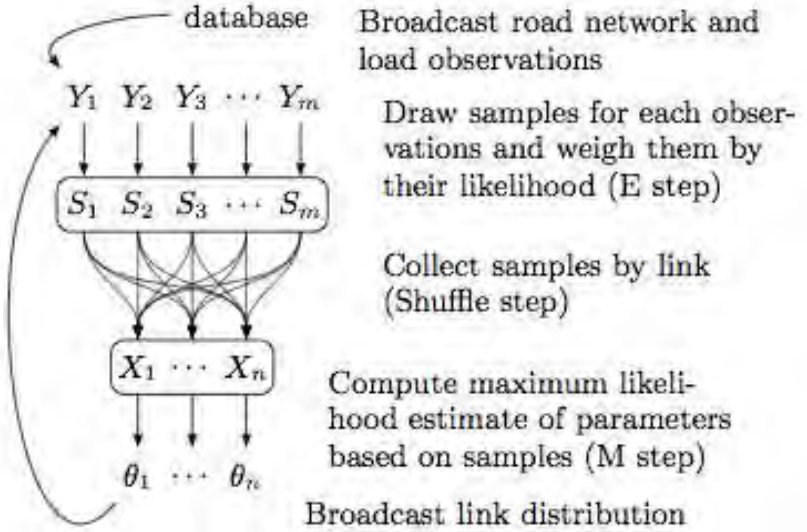
Step 1: Randomly partition observed time over all trajectory links

Step 2 (Expectation): Weigh partition by likelihood according to current estimate of travel time distributions

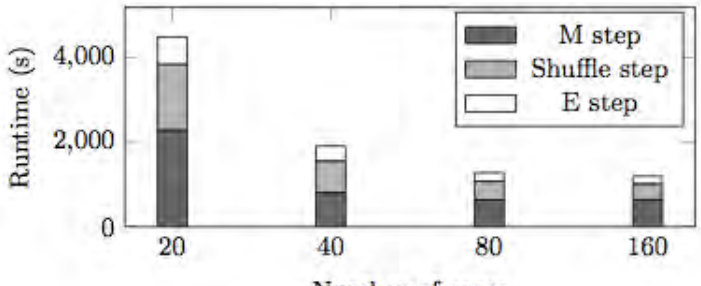
Step 3 (Maximization): Update link travel time distribution parameters to maximize likelihood of weighted samples

Step 4: Repeat until algorithm converges on travel time distribution parameters that fit the data well

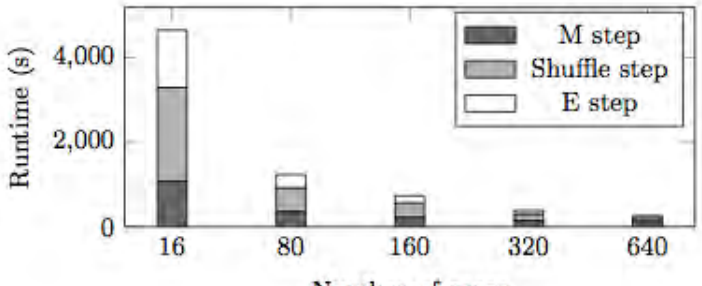
Parallelizing the EM Algorithm in Spark




```
// Load observations into memory as a cached RDD
observations = spark.textFile("hdfs://...")
                    .map(parseObservation).cache()

params = // Initialize parameter values

while (!converged) {
  // E-step: generate (linkId, sampledVals) pairs
  samples = observations.map(
    ob => generateSamples(ob, params))

  // Shuffle and M-step: group samples for each link,
  // update params, and return them to the master
  params = samples.groupByKey().map(
    (linkId, vals) => updateParam(linkId, vals)
  ).collect()
}
```



(a) Amazon EC2



(b) NERSC Cluster

TMIP

Summary

Big Data infrastructure

- Supports extremely large-scale storage
- Often emphasizes system availability over consistency

Big Data algorithms

- Must scale to accommodate large input sets

DISCLAIMER

The views and opinions expressed during this webinar are those of the presenters and do not represent the official policy or position of FHWA and do not constitute an endorsement, recommendation or specification by FHWA. The webinar is based solely on the professional opinions and experience of the presenters and is made available for information and experience sharing purposes only.