Advanced Database Technologies Course

Vijay Gadepally

Chansup Byun, Lauren Edwards,
Jeremy Kepner, Julie Mullen, Scott Sawyer

August 2014

Course Overview

- Day 1 – Advanced Database Technologies
  - Introduction to Big Data/Cloud Computing [30 min]
  - Introduction to Database Technologies [45 min]
    - Different technologies
    - Interacting with different technologies
    - What do sponsors use?
  - Database technology - Apache Accumulo [45 min]
    - Overview
    - Architecture/Components
    - How it works
    - Schema/Schema design
- Day 2 – Using Accumulo at Lincoln Laboratory
  - D4M technology [75 min]
    - About
    - Associative Arrays
    - Hands-on Example
  - Using Python Thrift Bindings to access Apache Accumulo [40 min]
  - Conclude and summarize [5]

Day 1

- Roughly broken into 3 parts:
  - Part 1: Introduction to Big Data and Cloud Computing
  - Part 2: Database Technologies
  - Part 3: Introduction to Apache Accumulo
- Day 2 hands-on exercises will reinforce the concepts taught

Course Goal:
To give you the knowledge needed to effectively use Apache Accumulo and knowledge about other DB technologies

Part 1: Introduction

Part 1: Agenda

- Introduction
- Systems Engineering
- Databases and Accumulo
- Cloud Computing
- MATLAB/Octave Fundamentals
- D4M

Common Big Data Challenge

Users

Operators

Analysts

Commanders

Rapidly increasing
- Data volume
- Data velocity
- Data variety

Data

Gap

Users

2000 2005 2010 2015 & Beyond

Data

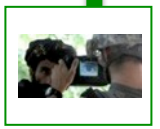
OSINT

Weather

HUMINT

C2

Ground

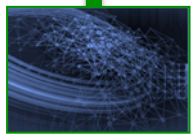


Maritime

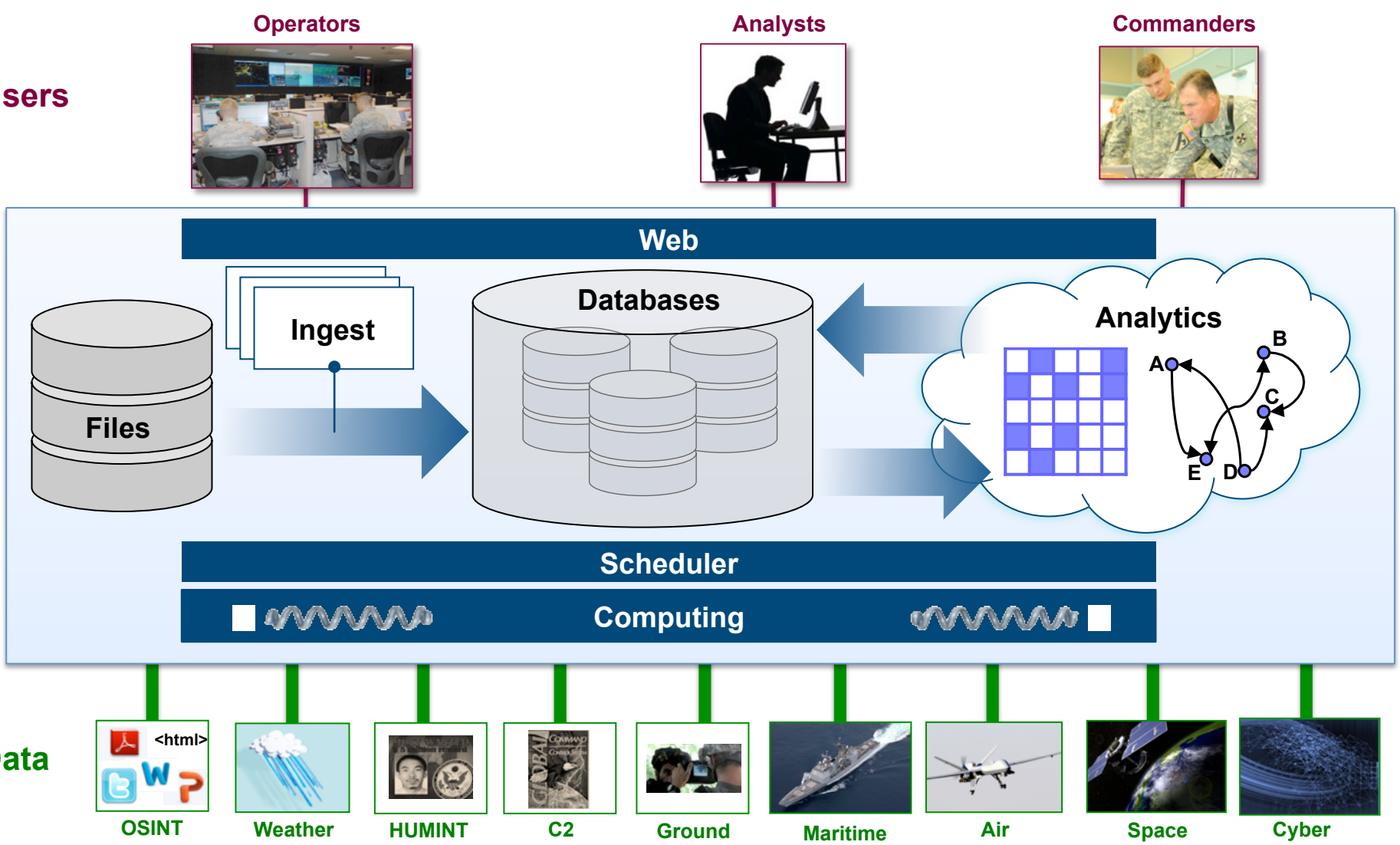
Air

Space

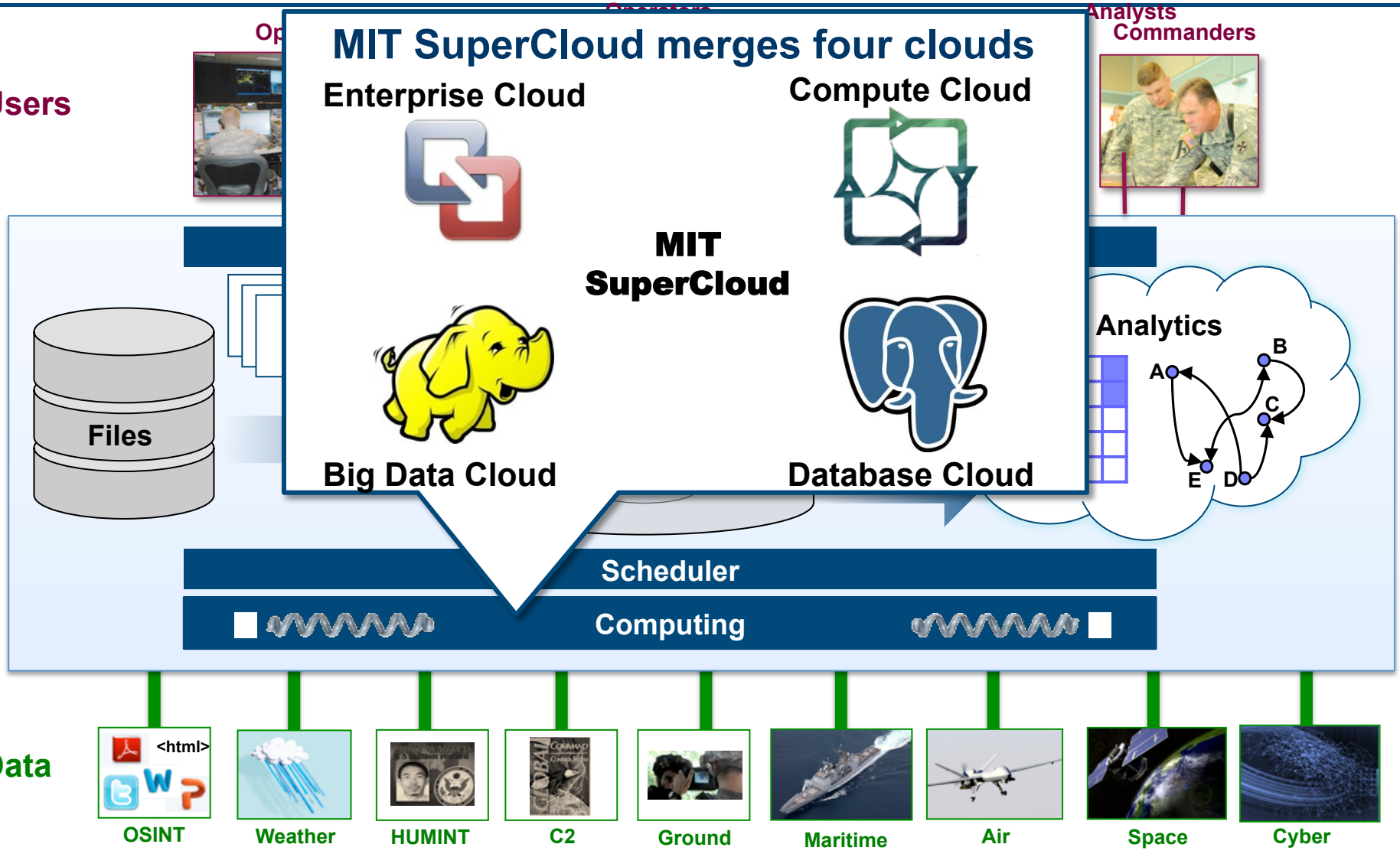
Cyber



Common Big Data Architecture



Common Big Data Architecture
- Data Volume: Cloud Computing -



LLSuperCloud: Sharing HPC Systems for Diverse Rapid Prototyping, Reuther et al, IEEE HPEC 2013

Common Big Data Architecture
- Data Velocity: Accumulo Database -

Users

Operators

Analysts

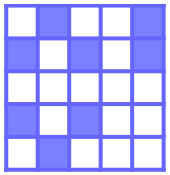
Commanders

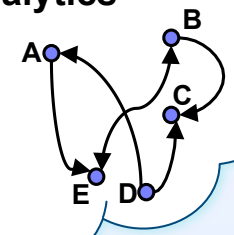
Web

Ingest

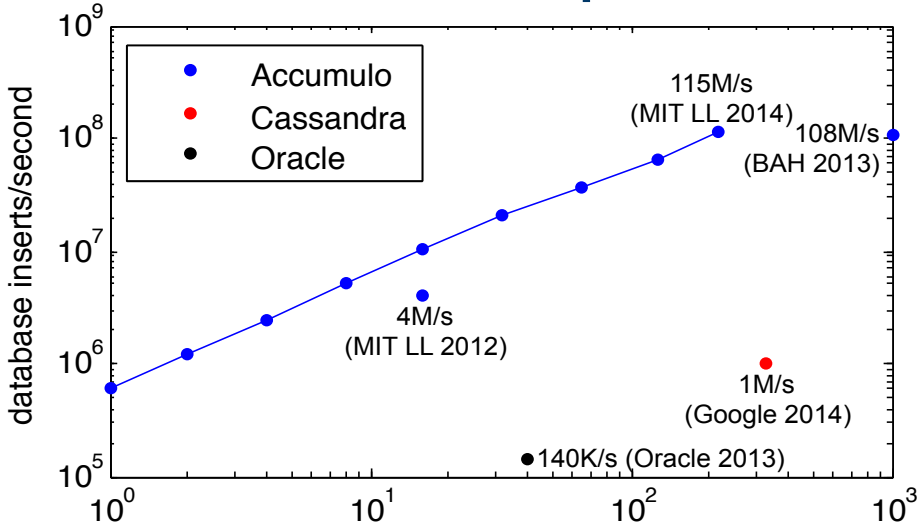
Databases

Analytics





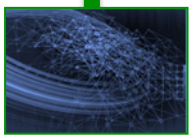
Lincoln benchmarking validated Accumulo performance



Maritime

Air

Space



Cyber

Common Big Data Architecture
- Data Variety: D4M -

Users

Operators

Analysts

Commanders

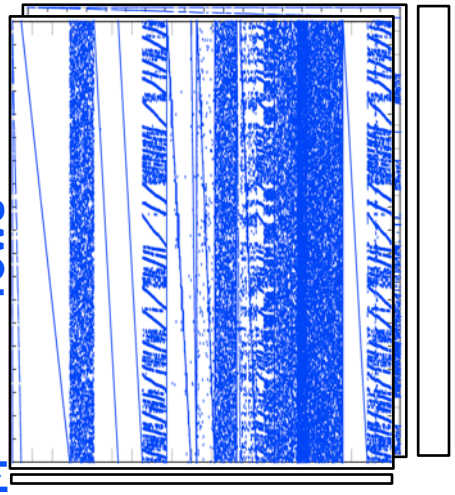
D4M ability to handle diverse data through schema and DB support

columns

raw

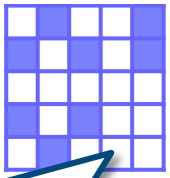
rows

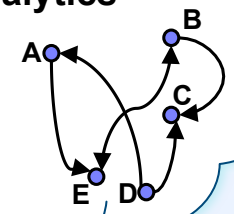
Σ



Files

Analytics





intel reports, DNA, medical data, health records, publication citations, web logs, social media, … all handled by a common 4 table schema

Data

OSINT

Weather

HUMINT

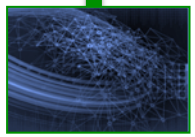
C2

Ground

Maritime

Air

Space



Cyber

D4M 2.0 Schema: A General Purpose High Performance Schema for the Accumulo Database, Kepner et al, IEEE HPEC 2013

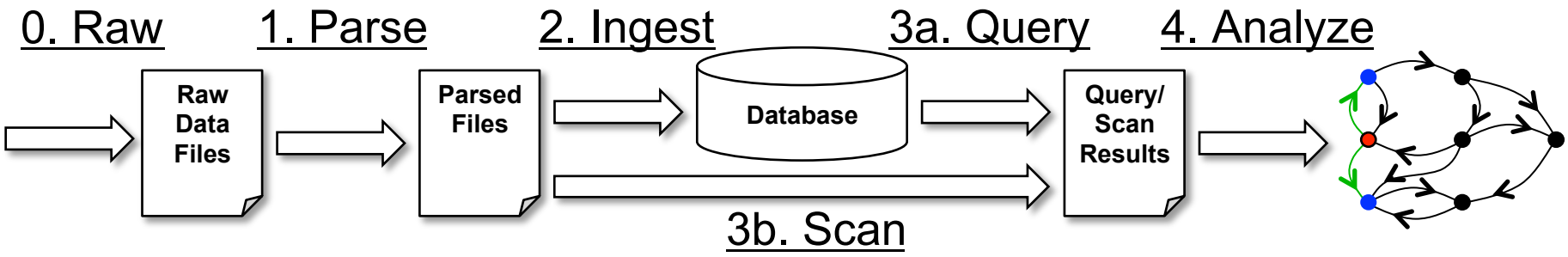
Questions

- What are the 3 V's of Big Data?
- Do you have any application(s) in mind in which you can map the 3 V's to your problem?

What is Systems Engineering?

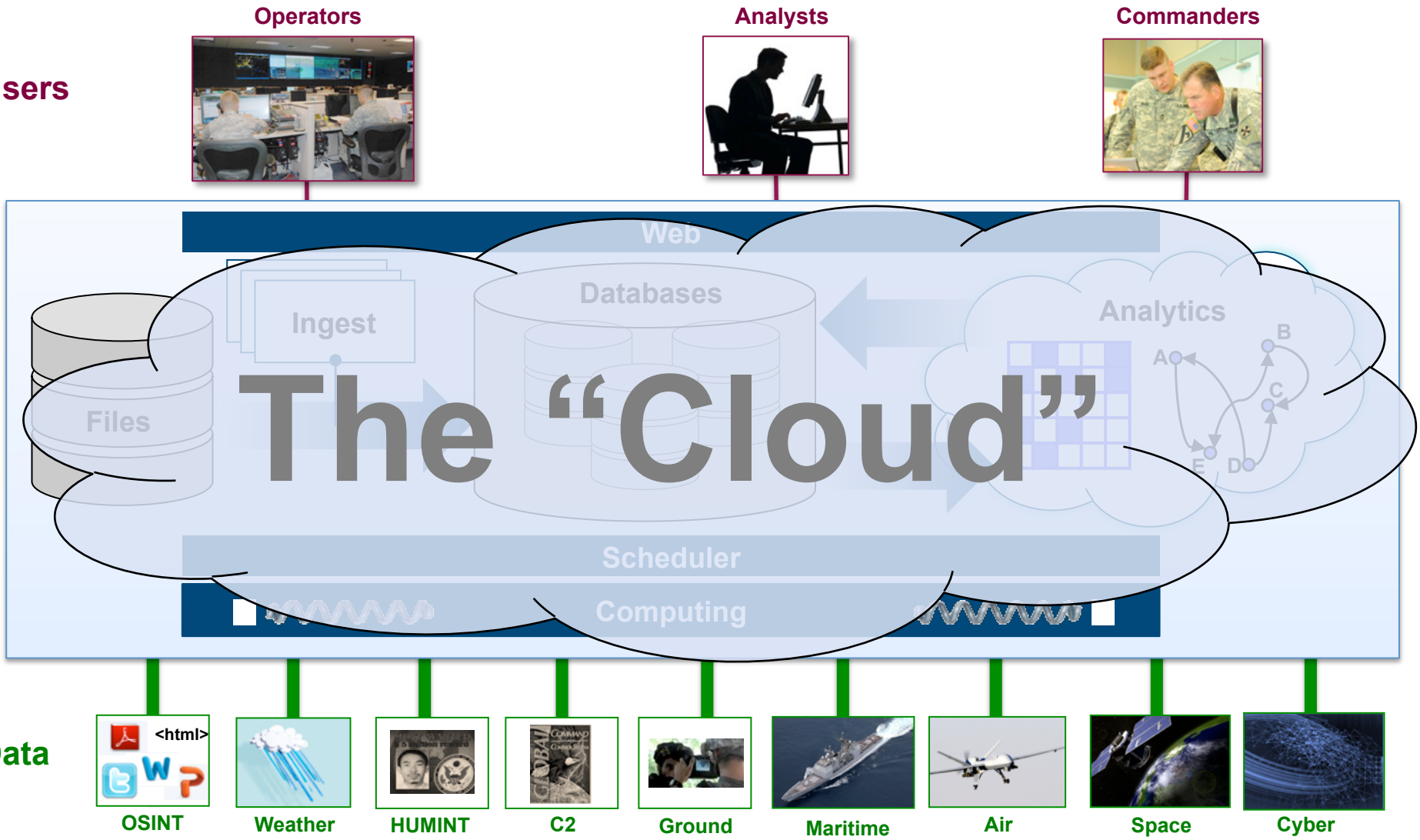
- The design and management of complex engineering projects
- Major concerns:
  - Reliability of system
  - Coordination of system components
  - System response
  - System integration
  - Information provenance

Systems Engineering Applied to Big Data

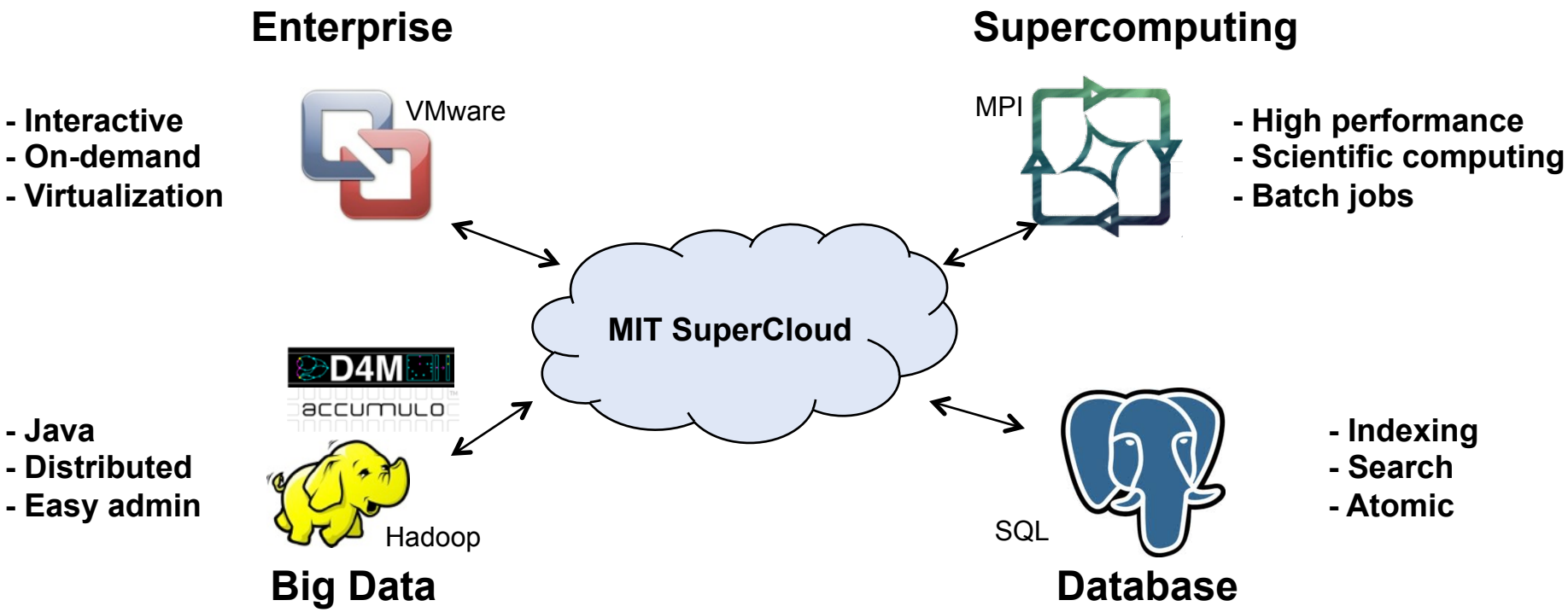


- **Raw: ingests raw data from source**
- **Parse: converts raw data into files suitable for ingest and scanning**
- **Ingest: inserts data into database**
- **Query: extracts selected data from database**
- **Scan: traverses a large number of parsed files**
- **Analyze: converts query results or scan results into meaningful relationships**

The cloud in the context of CBDA



Cloud Ecosystems



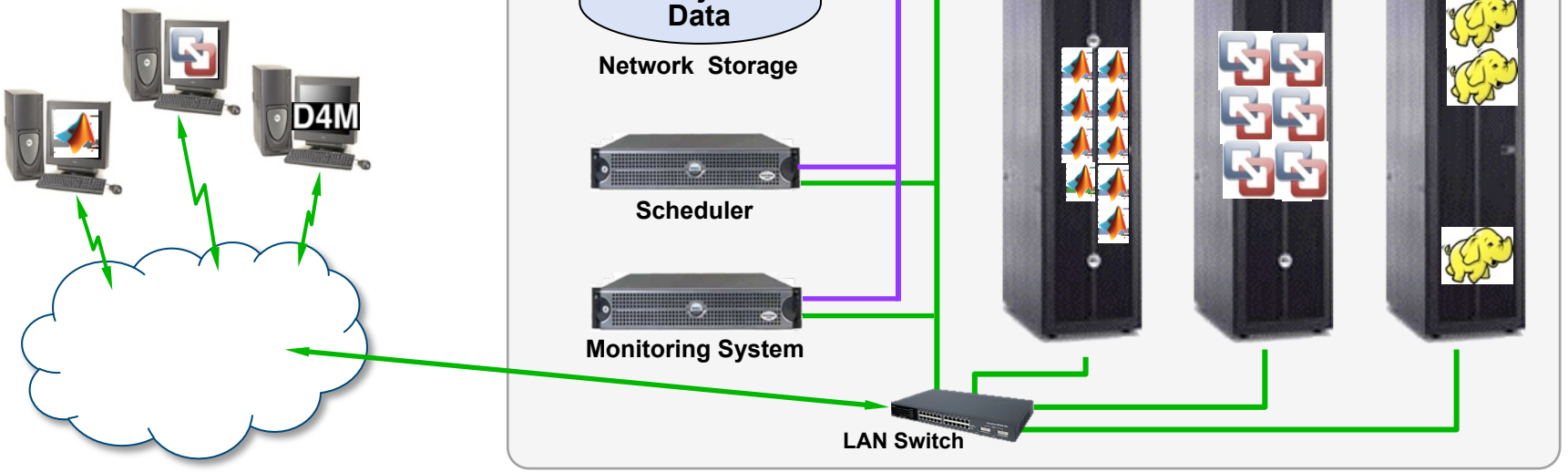
- **Allows different architectures to be dynamically combined and tested**

MIT SuperCloud

Interactive Compute Job

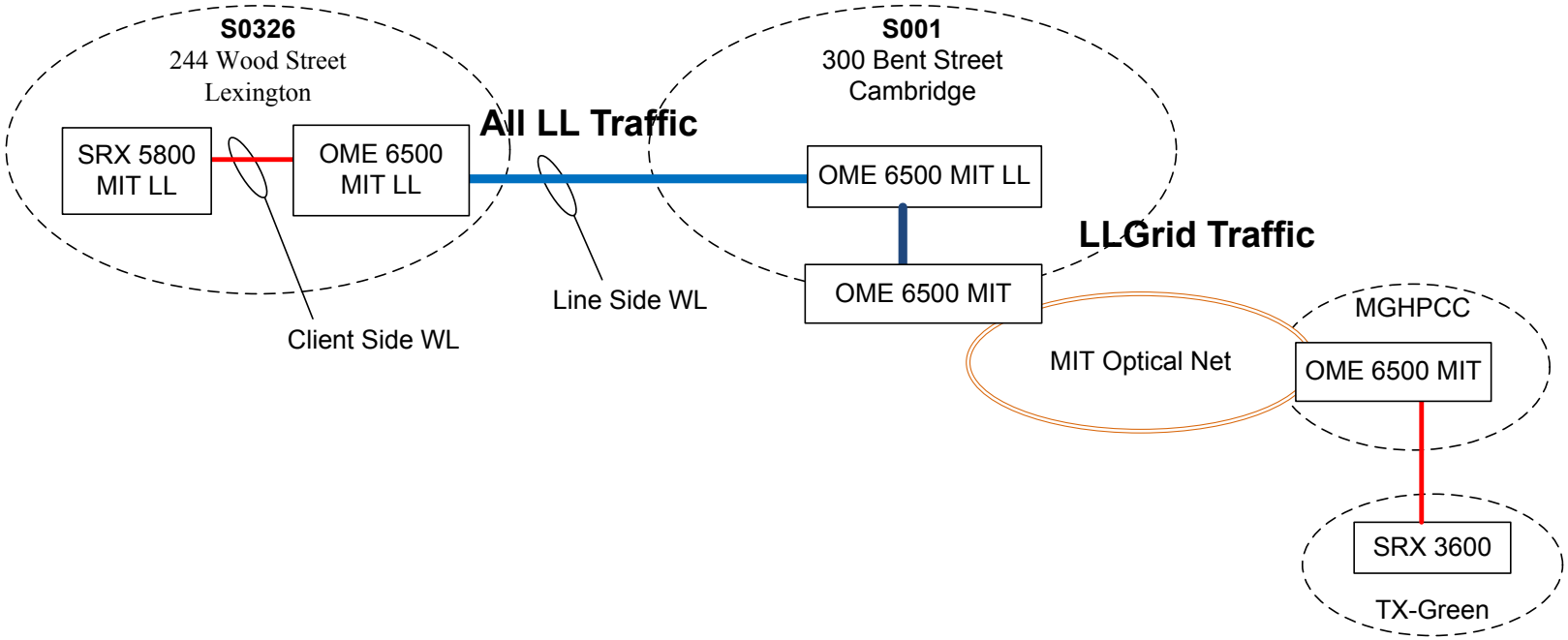
Interactive VM Job

Interactive Database Job



TX-2500/Green
Lexington/Holyoke, MA

Connection to Holyoke, MA (for the curious)



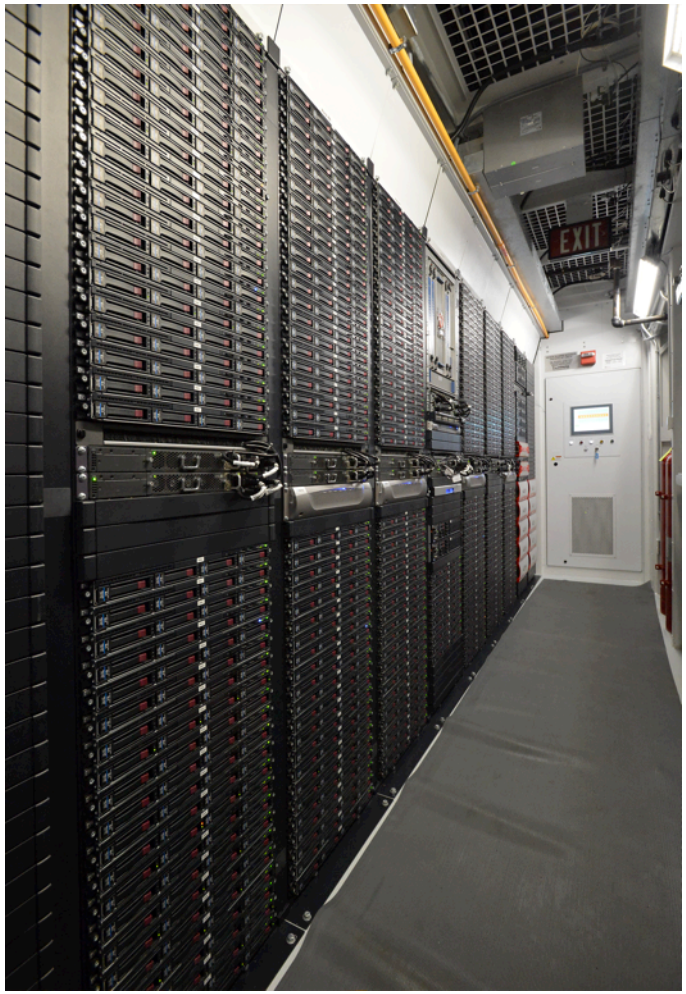
- **2x10G Muxponder (optical transponder) Module on OME 6500**
  - **Switches located at LL, Bent Street and MGPHCC**
- **Cross Connect to MIT Optical Network at 300 Bent St.**
- **10GB Fiber from SRX3600 to Voltaire 8500**

Cloud Computing @ MIT

- The cloud computing infrastructure at Lincoln Laboratory is based on the MIT Supercloud infrastructure which allows the different cloud eco systems to co exist
- Users can make use of different cloud environments using the same hardware and software platform
- One of the centerpieces of MIT SuperCloud is the big data capabilities that make use of various DB technologies

HPC and Big Data @ Lincoln Laboratory LLGrid

- LLGrid provides HPC resources to Lincoln Laboratory researchers
- Each system runs the MIT SuperCloud stack
  - TX-E1 for collaborative work outside of Lincoln
  - TX-2500 and TX-Green for internal Lincoln Laboratory work
- Lincoln Laboratory developed D4M technology provides mechanism to make use of big data resources



D4M

- Library that allows unstructured data to be represented as triples in Associative Arrays and can be manipulated using standard linear algebraic operations.
  - Currently implemented in MATLAB
  - Complex database operations and queries are composable as algebraic operations on associative arrays
  - D4M API makes it easy to develop analytics (just a few lines of code)
- Website: http://d4m.mit.edu
- A collection of tutorials can be found by emailing grid-help@ll.mit.edu

Associative Arrays

- **Extends associative arrays to 2D and mixed data types**

```
A('#a1','#b2') = 'same_tweet'
```

- **Key innovation: 2D is 1-to-1 with triple store**

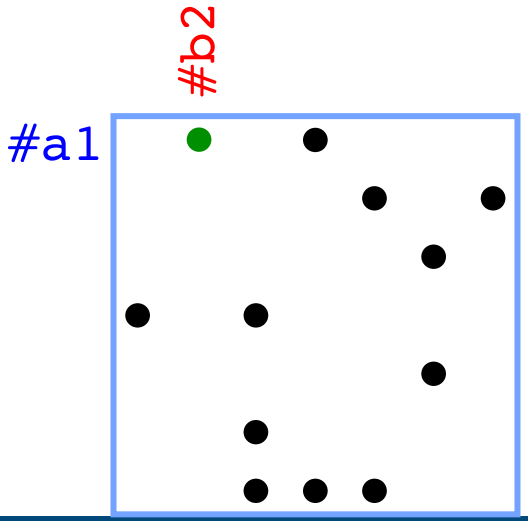
```
('#a1','#b2','same_tweet')
```

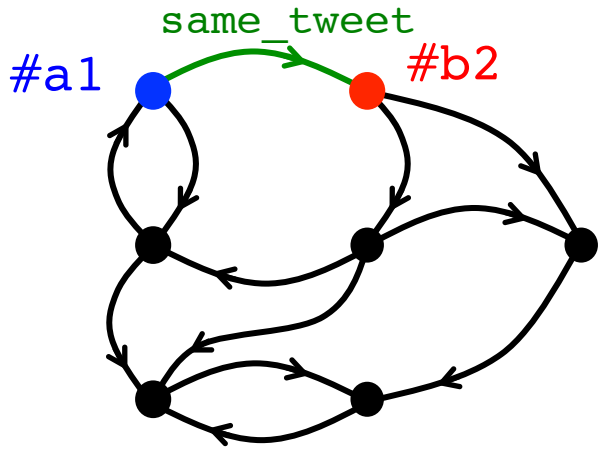
- **Enables composable mathematical operations**

```
A + B      A - B      A & B      A|B      A*B
```

- **Enables composable query operations via array indexing**

```
A('#al b2',:)    A('#al,',:)    A('#a* ',:)
A('#al: b2',:)   A(1:2,:)         A == #b2
```





Recap

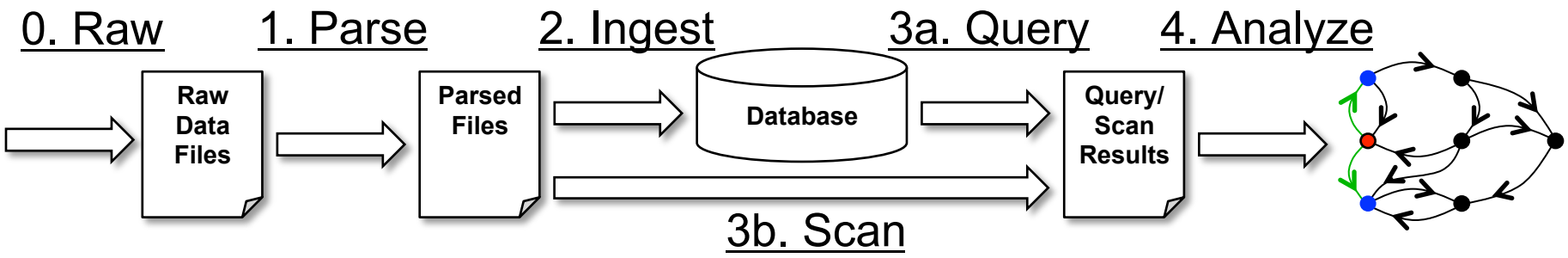
LLGrid...

If there's somethin' strange in the size of your data
Who ya gonna call?

If it's somethin' weird and it won't fit in memory
Who ya gonna call?

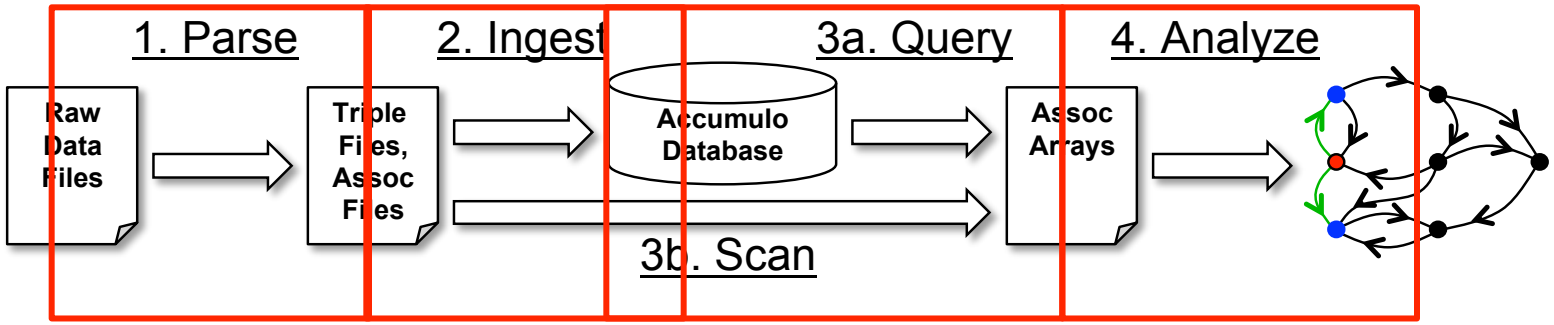
Designing a Big Data system

- **We have a collection of tools we can use towards developing a Big Data System**
  - **Understanding of systems engineering for big data systems**
  - **Understanding of cloud computing concepts**
  - **Database Fundamentals**
- **Let us look at using these tools to develop a big data system using raw Twitter data**



Common System Design Case

- *Starting*: Given a raw set of data that contains geo information
- *Goal*: Perform analytics and visualize results
- Use the D4M pipeline:
  1. Parse the raw data
  2. Insert/Ingest triples into a database
  3. Query database (a) or Scan system (b) for data (depends on data volume and request size)
  4. Perform analytics
  5. Visualize analytics



#Twitter

- Twitter is a micro blog that allows 140 character or less posts:
  - >500 million active registered users
  - >50 million tweets per day
  - >100 million active twitter users/month
  - ~10,000 tweets/sec (record is 143,199 Tweets/sec)
- Lots of ongoing research on what to make of data:
  - http://www.danah.org/researchBibs/twitter.php
- A prototype dataset with approximately 2 million tweets will be used for system development

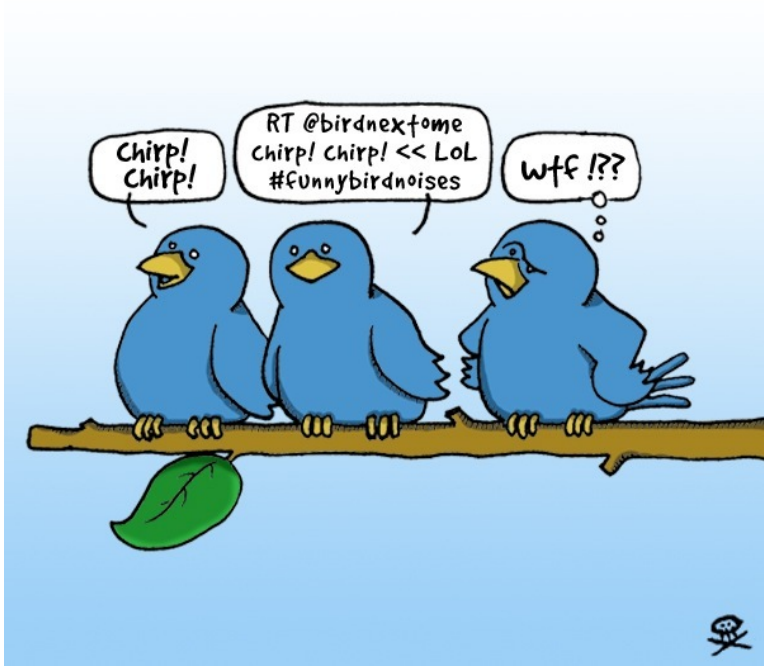
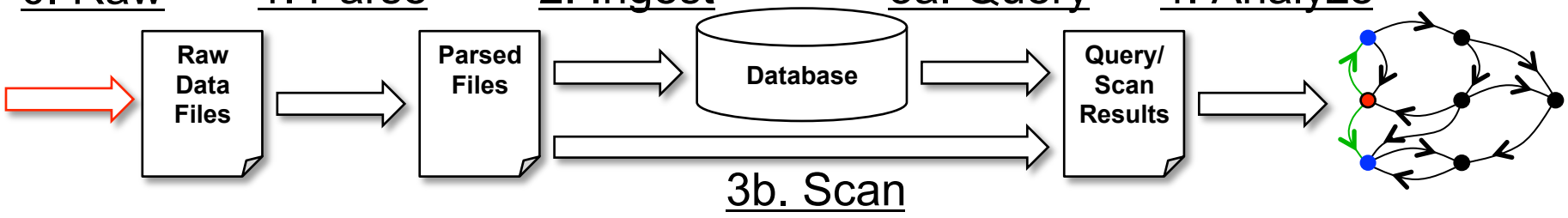


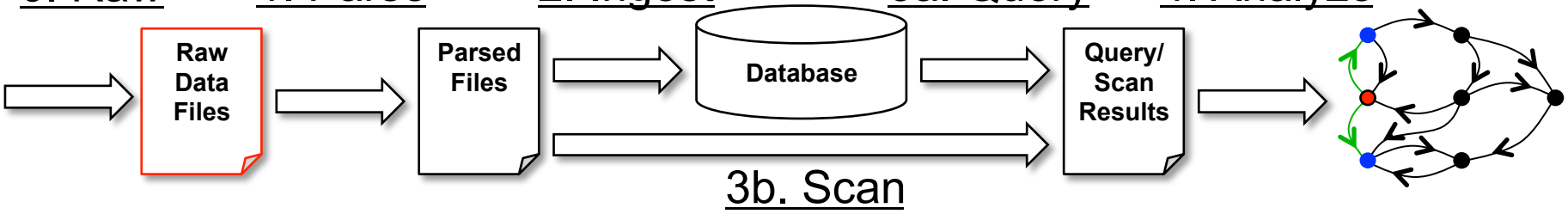
Image source: http://twittertoolsbook.com/essential-acronyms/

Creating Raw Data Files



- **Information you receive may come through live streaming or files from other sources in JSON or XML format**
  - **Convert such representation to either CSV or TSV formatted files**
- **Decisions:**
  - **Size of individual raw data files (optimal size between 10 and 100 MB)**
  - **Format (TSV, CSV, etc.)**
- **Factors:**
  - **File system block size**
  - **Memory or RAM of processing system**
  - **Rate of incoming data**
  - **Amount of parallelism you can afford**
  - **Contents of raw data**

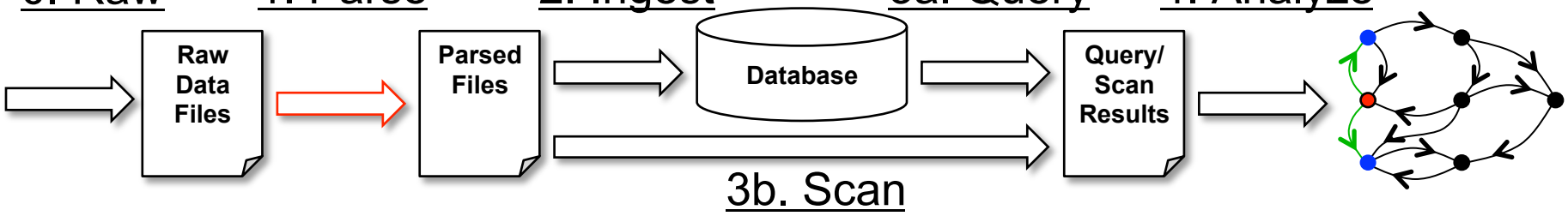
Raw Data
-Twitter Corpus-



TweetID	Time	Lat	Long	User	Source	Text
334458513407488001	2013-05-15 00:01:38	41.50442327	-71.3080676	alesiatasso	Twitter for iPhone	@LM_Rudd okay, I'm...
334458512308576256	2013-05-15 00:01:37	42.56289604	-84.82935883	skally_pal	Twitter for iPhone	@ItsMackk I'll see you…
334458513755602944	2013-05-15 00:01:38	41.39560229	-81.73950863	AmbahDee	Twitter for iPhone	@NataliaProkop come…
334458515798245377	2013-05-15 00:01:38	37.19058402	-93.31629432	RockSteady	Twitter for iPhone	@ktp35 I didn't think to...
334458515383005184	2013-05-15 00:01:38	35.6337457	139.6607298	karate_h	Twitter for Android	"""@IngatanSekolah: …
334458520244219904	2013-05-15 00:01:39	42.2041867	-87.8126571	yobebeau	Endomondo	Was out cycling 31.64 …
…	…	…		…		…

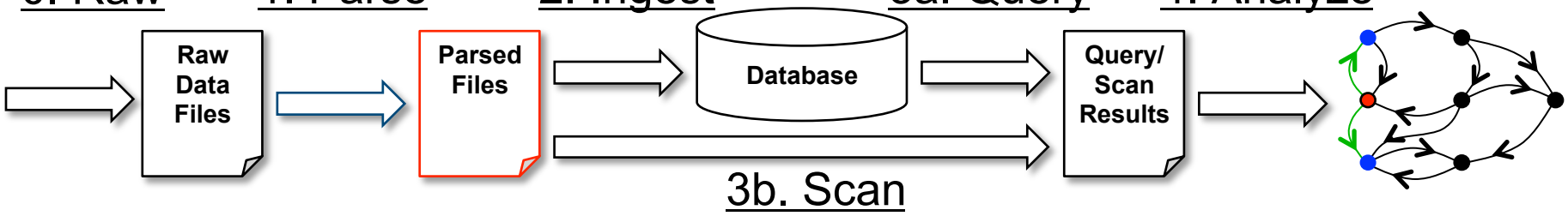
- **Mixture of structured (TweetID, User, Lat/Long, Time) and unstructured (Text)**
- **Tweet ID is unique number associated with each individual tweet**
- **Use the D4M schema to represent data**

Parsing Raw Data



- **Convert raw data to files suitable for ingesting to database and scanning**
- **Decisions:**
  - **Parsed data contents (i.e., what do you need from raw data?)**
  - **Data representation (i.e., what is the schema?)**
  - **Parser design (C, C++, Java, MATLAB, etc.)**
- **Factors:**
  - **Information that will be used later in the system (for analytics)**
  - **Use D4M schema for general purpose**
  - **Amount of parallelism desired in parsing**
  - **How much data needs to be parsed**
  - **Real time parsing vs. One time parsing**

Parsed Data
-Twitter Corpus-

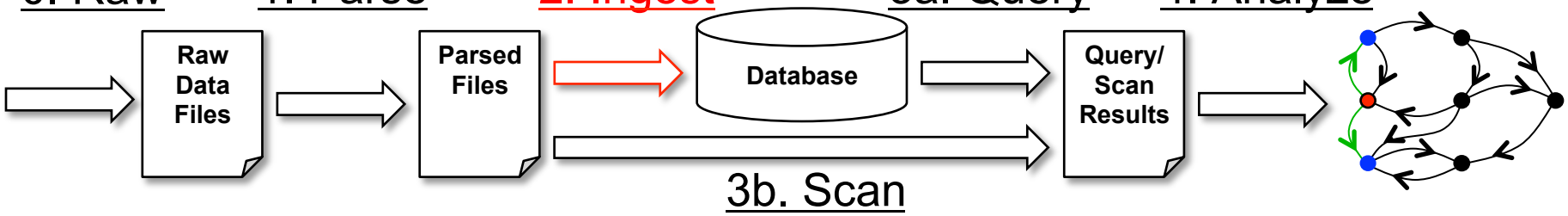


- **Parsed data is stored in Associative Arrays**
- **Each raw data file creates a .mat file (MATLAB data file) with 3-tuple representation of associated raw data file**
- **For 2 million tweet set – approximately 2000 parsed files**

```
3369993125989048 32
2013-05-22 00:17:51
1033273868 Coffeshootweb
682c5a667856ef42
üşenipte takip etmeyenin
hamına #SeriTakipleşmeVar
```

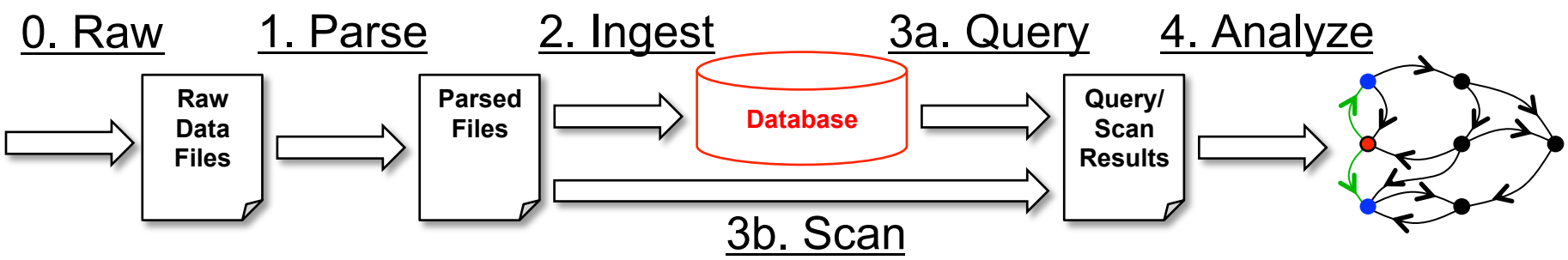
```
>>A('23840989521399633,',:)
(23840989521399633,place|682c5a667856ef42)     1
(23840989521399633,time|2013-05-22 00:17:51)     1
(23840989521399633,userID|1033273868)     1
(23840989521399633,user|Coffeshoot)     1
(23840989521399633,word|#SeriTakipleşmeVar)     1
(23840989521399633,word|etmeyenin)     1
(23840989521399633,word|hamına)     1
(23840989521399633,word|takip)     1
(23840989521399633,word|üşenipte)     1
```

Ingesting Parsed Data



- **Put parsed data into suitable database**
- **Decisions:**
  - **Size of database (how many tablets, nodes, etc.)**
  - **Number of ingestors**
  - **Any preprocessing tricks to improve ingestion rate, load balancing, etc.**
- **Factors:**
  - **Desired Ingest Rate (rule of thumb: 100k entries/sec/ingestor)**
  - **Amount of data in the database**
  - **Amount of parallelism desired in ingesting**
  - **Structure of parsed data, sharding, etc.**

Database
-Twitter Corpus-



Refresh Status

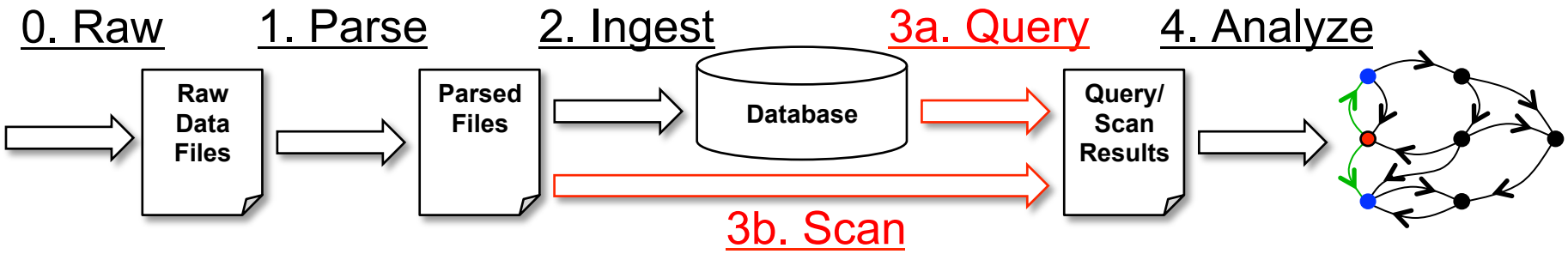
Folder Name	Type	Status	Actions
classdb01	Accumulo v1.4.1	started	View Info Stop
classdb02	Accumulo v1.4.1	started	View Info Stop
classdb03	Accumulo v1.4.1	started	View Info Stop
classdb04	Accumulo v1.4.1	started	View Info Stop
classdb04c1	Accumulo v1.4.1	started	View Info Stop
classdb05	Accumulo v1.4.1	stopped	View Info Start Checkpoint

TABLE LIST

Show Legend

TABLE NAME	STATE	# TABLETS	# OFFLINE TABLETS	ENTRIES	ENTRIES IN MEMORY	INGEST	QUERY	HOLD TIME	RUNNING SCANS	MINOR COMPACTIONS	MAJOR COMPACTIONS
!METADATA	ONLINE	3	0	1.05K	16	0	3	–	0 (0)	0 (0)	0 (0)
geo_tweets01_Tedge	ONLINE	40	0	2.67B	0	0	0	–	0 (0)	0 (0)	0 (0)
geo_tweets01_TedgeDeg	ONLINE	11	0	442.53M	0	0	0	–	0 (0)	0 (0)	0 (0)
geo_tweets01_TedgeT	ONLINE	49	0	2.68B	0	0	0	–	0 (0)	0 (0)	0 (0)
geo_tweets01_TedgeTxt	ONLINE	10	0	166.45M	0	0	0	–	0 (0)	0 (0)	0 (0)
trace	ONLINE	1	0	8.33M	1.48K	0	0	–	0 (0)	0 (0)	0 (0)

Querying/Scanning for Results



- **Extract data from file system or database**
- **Decisions:**
  - **Whether to scan parsed files or query database**
  - **Whether to use an iterator**
  - **Whether to perform parallel query/scan**
- **Factors:**
  - **Amount of data to be extracted (estimated by querying the Degree table)**

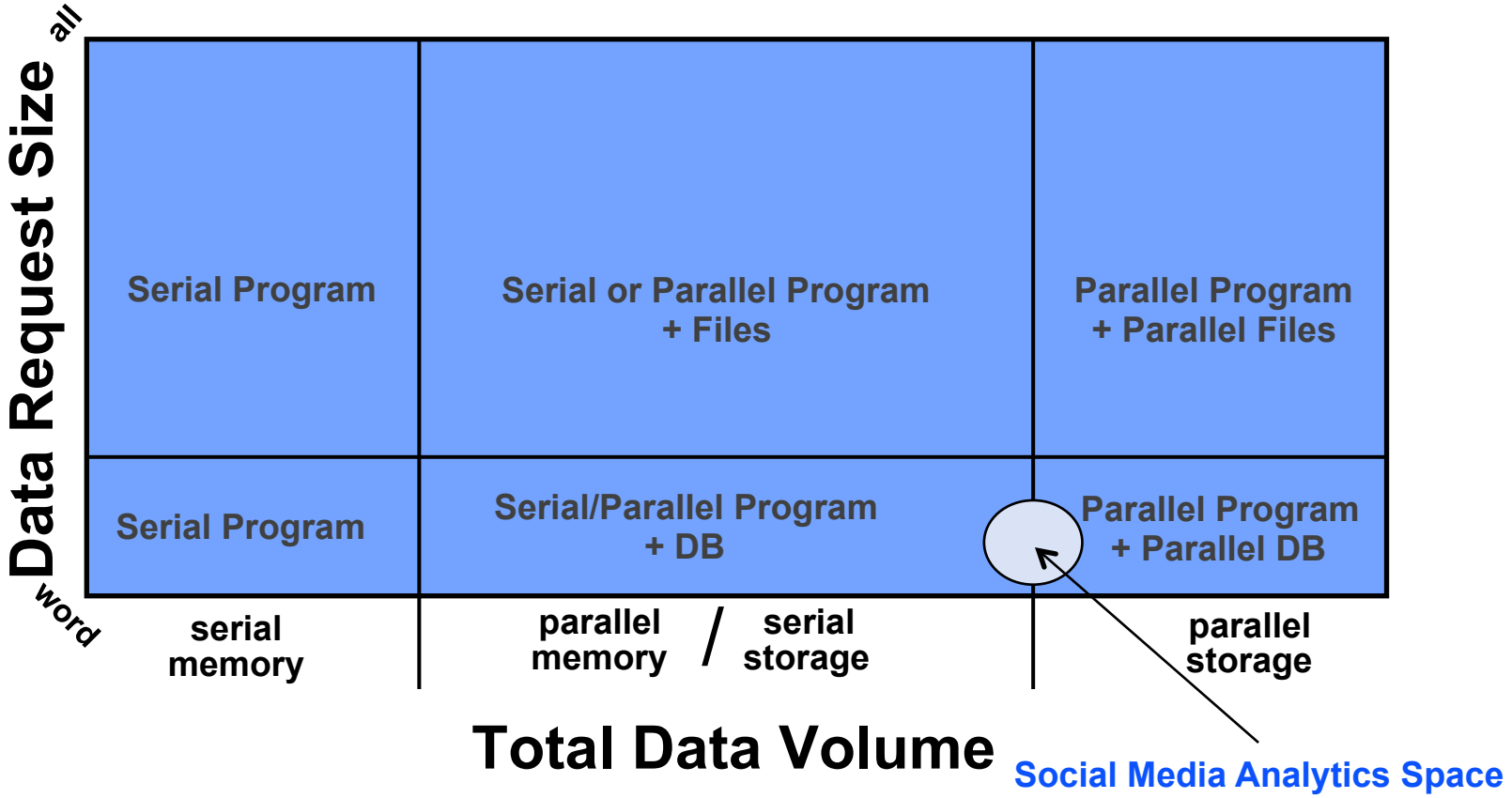
Table Entries

```
>> TedgeDeg('word|#randomtext,',:)
(word|#randomtext,Degree)     13
>> Tedge(:, 'word|#randomtext,')
(016427245783658443,word|#randomtext)     1
(027686813370870833,word|#randomtext)     1
(042094214119692453,word|#randomtext)     1
(236396225248481923,word|#randomtext)     1
(272015749079243553,word|#randomtext)     1
(278780367455870543,word|#randomtext)     1
(605778503156232753,word|#randomtext)     1
(610418338544481923,word|#randomtext)     1
(677764954478444043,word|#randomtext)     1
(718814086377900543,word|#randomtext)     1
(796187875762080453,word|#randomtext)     1
(846116979232252933,word|#randomtext)     1
(904543173254510133,word|#randomtext)     1
>> TedgeTxt(Row(Tedge(:, 'word|#randomtext,')),:)
(016427245783658443,text)     #randomtext bless him :) #cute haha x http://t.co/sCtN9RP18y
(027686813370870833,text)     Apparently some random person thinks I'm fat... #randomtext #fat http://t.co/k5MAmmbNsg
(042094214119692453,text)     Random texts from my dad  #randomtext #textfromdad #toofunny http://t.co/gvFN8QsNYn
(236396225248481923,text)     #randomtext @ UCR HQ http://t.co/ag4WG775qQ
(272015749079243553,text)     Best #randomtext ever! #freshprince http://t.co/qWGkfWlqbc
(278780367455870543,text)     Ahaha I guess you're having fun already @AustinGash #randomtext #whathappensinvegas
(605778503156232753,text)     Am I missing something? Lol #randomtext
(610418338544481923,text)     #randomtext @ UCR HQ http://t.co/jaVJDfJiPH
(677764954478444043,text)     Someone should take my phone when I drink! Please don't hate #randomtext #notcool #dontremember  #notgood
(718814086377900543,text)     "I'm trying to go hard, and I know you are on that level" why yes, yes I am :) #randomtext #edc
(796187875762080453,text)     When the past tries to hold you back, keep moving forward. #randomtext
(846116979232252933,text)     #randomtext that makes no sense @ Labrador Park MRT Station (CC27) http://t.co/p5VZ0HLlJH
(904543173254510133,text)     #randomtext @ UCR HQ http://t.co/gI9m44KQ0j
>>
```

Associative Arrays

Querying data using D4M API

To Scan or not to Scan

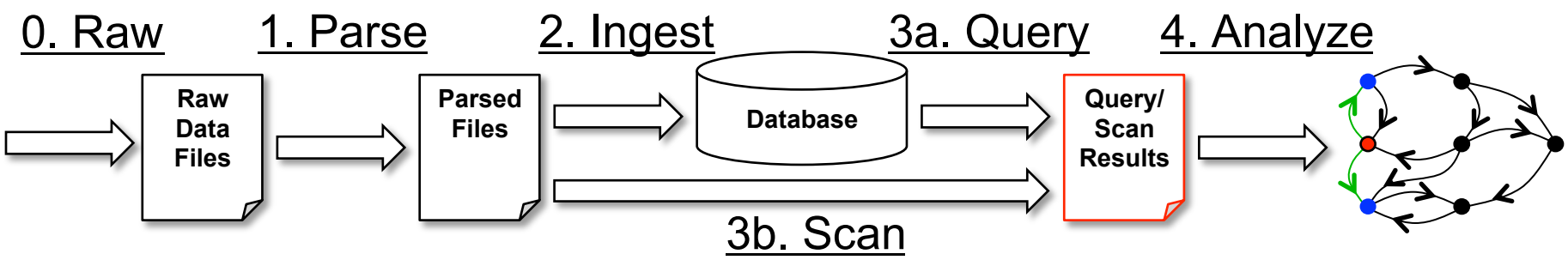


- Data volume and data request size determine best approach
- Always want to start with the simplest and move to the most complex
- In social media analysis, often, big data volume and relatively small request sizes

Query and Scan Tips

- Use degree table to estimate number of results prior to running scan/query
- Query Planning: When performing a 'join' or 'filter' operation, your first query should be for the one with lower entries
- Use an iterator if querying for a large number of elements
- Often, you need a combination of scanning and querying in getting your results – nothing wrong with this!
- General rule of thumb: Scan file system when the total data volume you need is greater than 5% of the total data size.
  - For example, if you have 2 million tweets in the database, you probably should scan the file system for any queries that may return more than 100,000 results.

Query/Scan Results



- **Results may be in the form of Associative Arrays**
- **You can perform algebraic operations on associative arrays.**
- **For example:**

```
>> Aedge = Tedge('00000069295661733,',:)
(00000069295661733,latlon|-
+010107..5103447857706000000)     1
(00000069295661733,lat|-000.500000)     1
(00000005661733,lon|+117.1347576000)     1
…
```

Aedge

+

```
>> Atxt = TedgeTxt('00000069295661733,',:)
(00000069295661733,text)
"@Super7_Updates: Ready Pertnyaan ke 2?"
```

Atxt

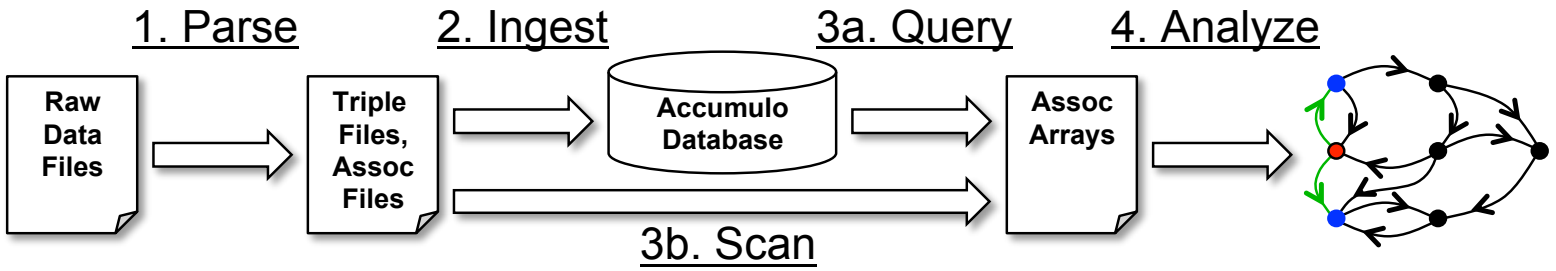
=

```
>> Afull = Aedge+Atxt
(00000069295661733,latlon|-
+010107..51034477857706000000)     1
(00000069295661733,lat|-000.5047870000)     1
(00000069295661733,lon|+117.1347576000)     1
(00000069295661733,place|8f6d8fbc7060e679)     1
(00000069295661733,text)     "@Super7_Updates: Ready
Pertnyaan ke 2?"
(00000069295661733,time|2013-05-22 11:22:34)     1
(00000069295661733,userID|1189242306)     1
(00000069295661733,user|foreverbagasSMD)     1
(00000069295661733,word|"@Super7_Updates:)     1
(00000069295661733,word|2?")     1
(00000069295661733,word|Pertnyaan)     1
(00000069295661733,word|Ready)     1
(00000069295661733,word|ke)     1
```

Afull

Visualization Use Case

- *Starting*: Given a raw set of data that contains geo information
- *Goal*: Perform analytics and visualize results
- Use the D4M pipeline:
  1. Parse the raw data
  2. Insert/Ingest triples into a database
  3. Scan file system or Query database for data
  4. Perform analytics
  5. Analyze and Display analytics



A note about Visualization Options

- Often, geo location data needs to be visualized

- Many other tools are available

Visualization Example
-D4M syntax-

- **>> A = Tedge(Row(Tedge(:,'word|#Egypt,')),:);**
- **>> Assoc2KML(A(:, latlon|++003301..0243,:,latlon|++003301..0254,')**

Break

- What are the 5 steps of the pipeline?
- Name 2 design decisions you need to make when setting up your own big data pipeline

Part 2: Database Technologies

Database Fundamentals

- **Database:** Collection of data and supporting data structures
- **Database Management Systems:** Software that provides interface between user and database

- **Common User-DBMS interactions:**
  - **Defining new data, new schema, etc.**
  - **Updating data**
  - **Retrieving (Querying) data**
  - **DB administration, security, permissions, etc.**

A Brief History of Open-Source Big Data

DATABASES

NoSQL

MySQL

PostgreSQL

MySQL Cluster

HBASE

BigTable

amazon web services

accumulo

Dremel

Cassandra

NewSQL?

1995 2004 2006 2008 2010 2012

PARALLEL PROCESSING

MapReduce

Pregel

D4M

GraphLab

GRID ENGINE

Hadoop

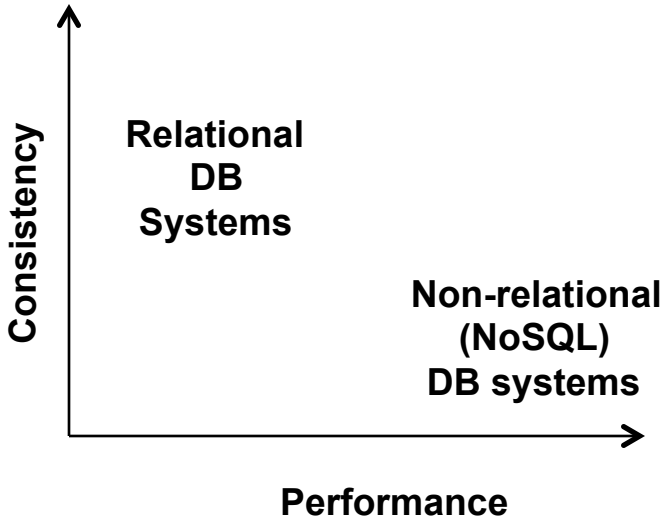
Graph Algorithms in the Language of Linear Algebra

Giraph

Slide Source: S. Sawyer, B. D. O'Gwynn, A. Tran, T. Yu. Understanding Query Performance in Accumulo. HPEC 2013.

Choosing the right DB

- **CAP Theorem:**
  - **Impossible for a distributed system to simultaneously provide guarantees on consistency (all nodes see same data all the time), availability (guarantee that every request receives a response) and partition tolerance (system can operate even after system failures)**
- **Atomicity, Consistency, Isolation, Durability (ACID) vs. Basically Available Soft-state services with Eventual-consistency (BASE)**



Relational Databases

- **What it is:**
  - Database that stores information about data and how it is related.
  - Table based databases, and tables contain n rows when you have n data entries
  - Predefined schema/organization of data
  - Vertically scalable (Depends on hardware power. Scales with better hardware)
  - Use SQL as query interface
  - Typically provide full consistency (only one version of stored data in the whole cluster)
- **Use Cases:**
  - Strong need to have consistent results (for example dealing with $$)
  - Willing to trade performance for accuracy
  - Need for ACID guarantees
- **Examples**:
  - mySQL, postgreSQL, Oracle

Relational Databases (2)

- **Who uses it:**
  - Banks
  - DoD
  - Financial systems
  - Older databases
- **When to use it:**
  - Performance not a primary issue or metric
  - Applications already in relational database
  - Problem sizes are not gigantic (>billions of records)
- **How to use it:**
  - Many options
  - SQL command line

Non Relational/Distributed Databases (noSQL)

- **What it is:**
  - Database based on documents, key-value pairs, graphs, or wide-column stores
  - No standard schema definitions necessary to adhere to. Dynamic schema
  - Horizontal scalability (Usually run on COTS, scales with more systems)
  - Typically provide “eventual consistency” there may be different valid versions of the same data in the cluster with different values.
- **Use Cases:**
  - OK with BASE (Basically available, soft state, eventual consistency) guarantees
- **Examples:**
  - Accumulo, Cassandra, MongoDB, Google Big Table

Non Relational/Distributed Databases (2)

- **Who uses it:**
  - Academic Community
  - Lincoln Laboratory
  - IC/DoD
- **When to use it:**
  - When you have large unstructured datasets
  - Strong need for high performance
  - Running on COTS hardware or large clusters
- **How to use it:**
  - Most have Python/JAVA bindings
  - Lincoln Laboratory D4M
  - Command Line

Quick Reference
RDBMS vs. NoSQL

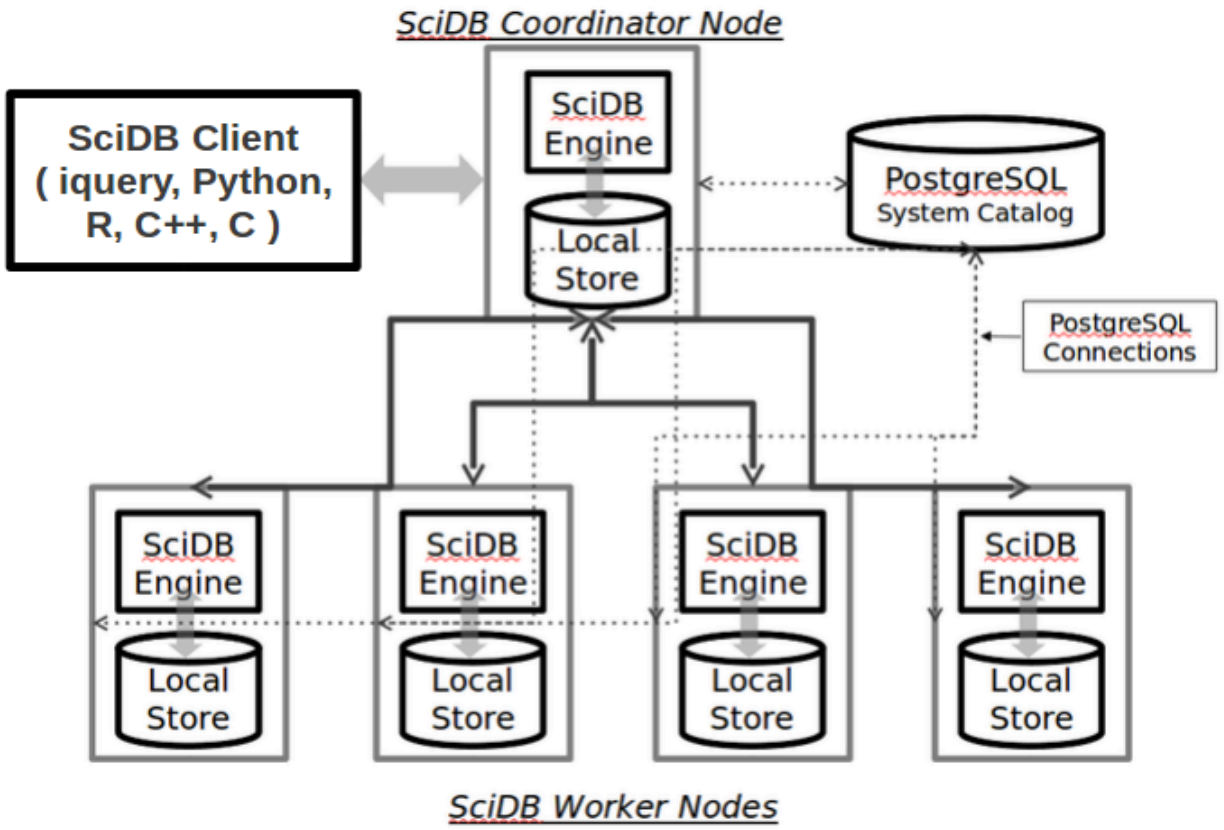
Relational Databases	NoSQL
MySQL, PostgreSQL, Oracle	HBase, Cassandra, Accumulo
Typed columns with relational keys	Schema-less
Single-node or sharded	Distributed, scalable
ACID transactions	Eventually consistent
SQL, indexing, joins, and query planning	Low-level API (scans and filtering)

Slide Source: S. Sawyer, B. D. O'Gwynn, A. Tran, T. Yu. Understanding Query Performance in Accumulo. HPEC 2013.

NewSQL

- **What it is:**
  - Modern database systems that emulate performance of NoSQL along with ACID guarantees of RDBMSs
  - Usually looks like a scaled up version of a relational database
  - Often uses an array data model in which individual worker databases are responsible for a portion of a distributed array
- **When to use it:**
  - Large multidimensional datasets such as imagery or time-series data
  - Data that doesn't fit in traditional databases
- **Examples**:
  - TileDB, SciDB, ...
- **Who uses it:**
  - Academic Community

NewSQL Example: SciDB



Question 1

- Your sponsor gives you a very large unstructured dataset with minimal documentation (surprised?)
- They ask you to recommend a DB technology that can provide high performance insert and extraction of the data and can fit in with their infrastructure.
- Raw data is in the form of TSV files and represent some network related logs
- What are the questions you would ask them, and what would be your first thoughts regarding a relevant DB technology?

Question 2

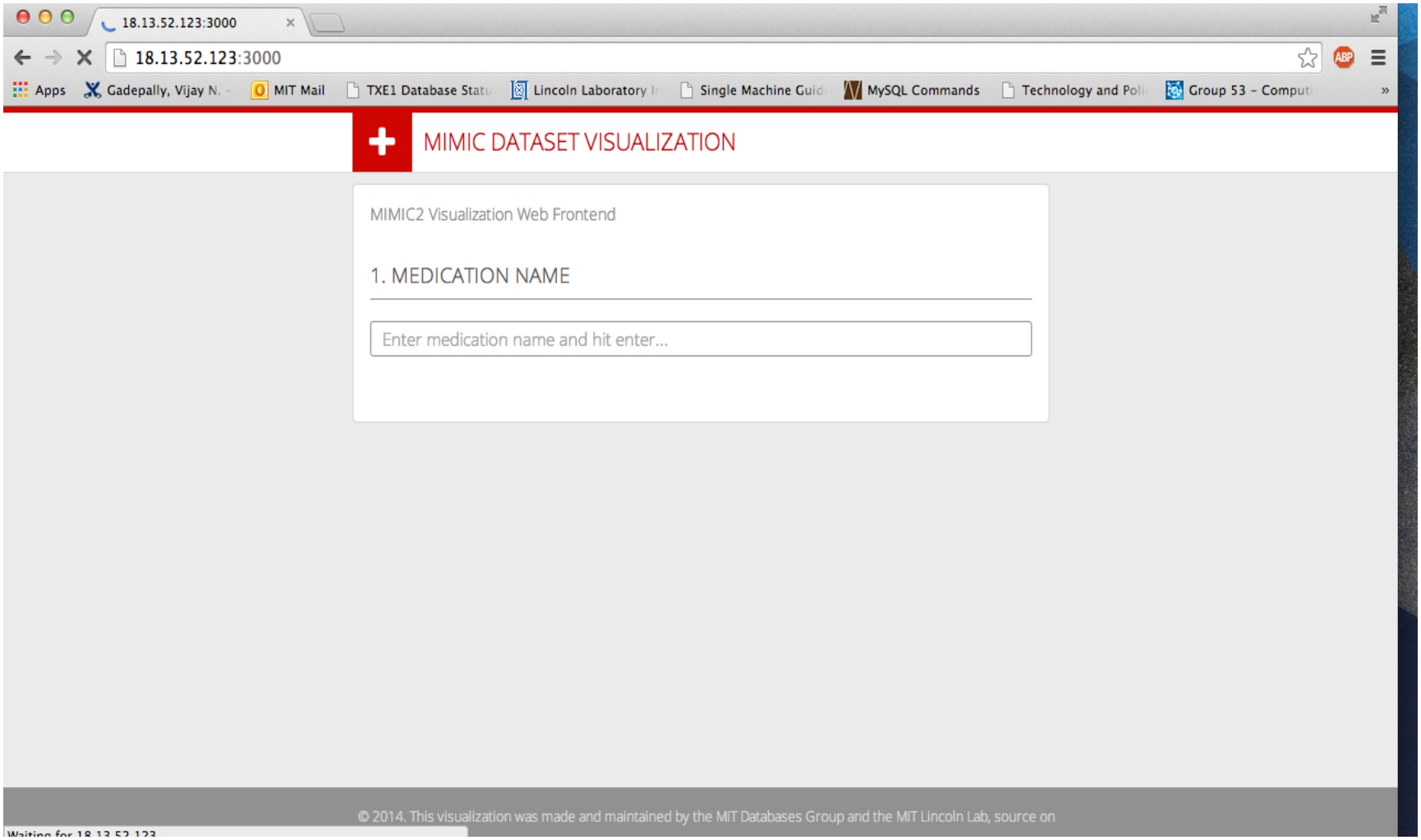
- You are given access to a large dataset of multi-parameter recordings of ICU patients that contains a large volume of structured patient information (such as Name, Gender, ...) and unstructured data collected from a variety of sensors that patients are hooked to.
- What are your first thoughts?
- Any thoughts on DB technology to use?

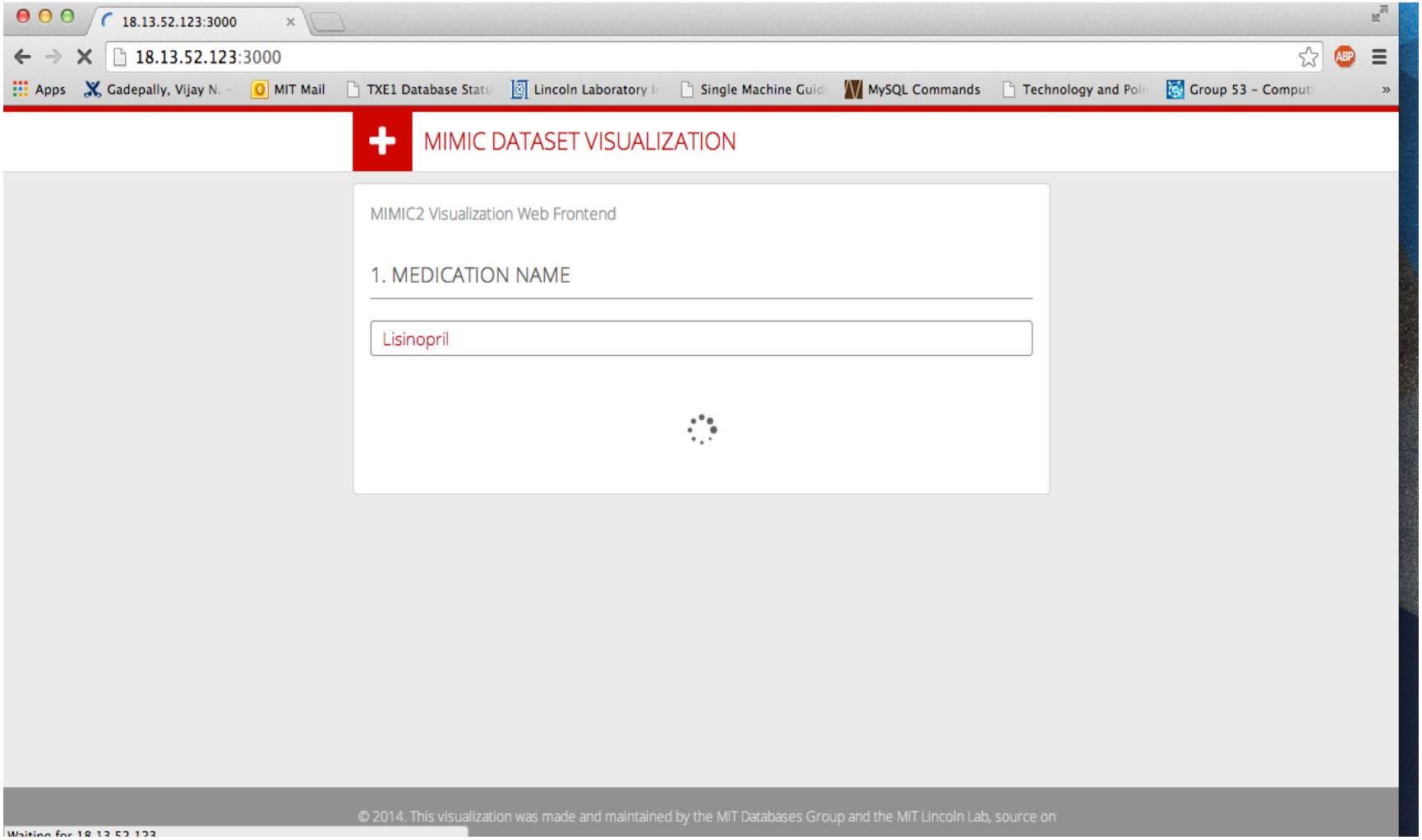
Question 2

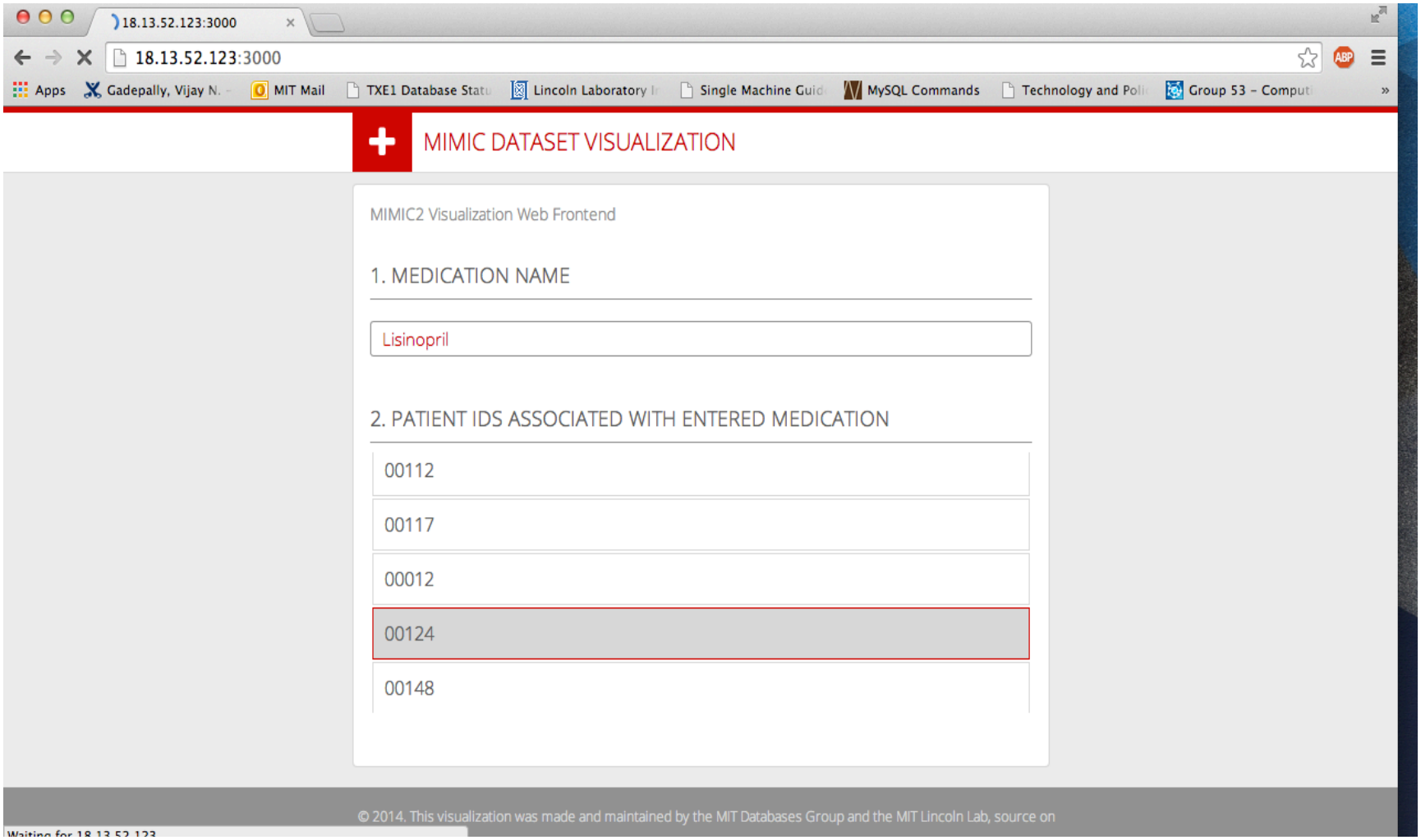
- This is something we are working on and such a dataset is called MIMIC2
  - Mulitiparameter Intelligent Monitoring in Intensive Care
  - http://physionet.org/mimic2/
- Patient records stored using mySQL
- Mapping between patients and waveform records in Accumulo
- Waveform records stored in SciDB

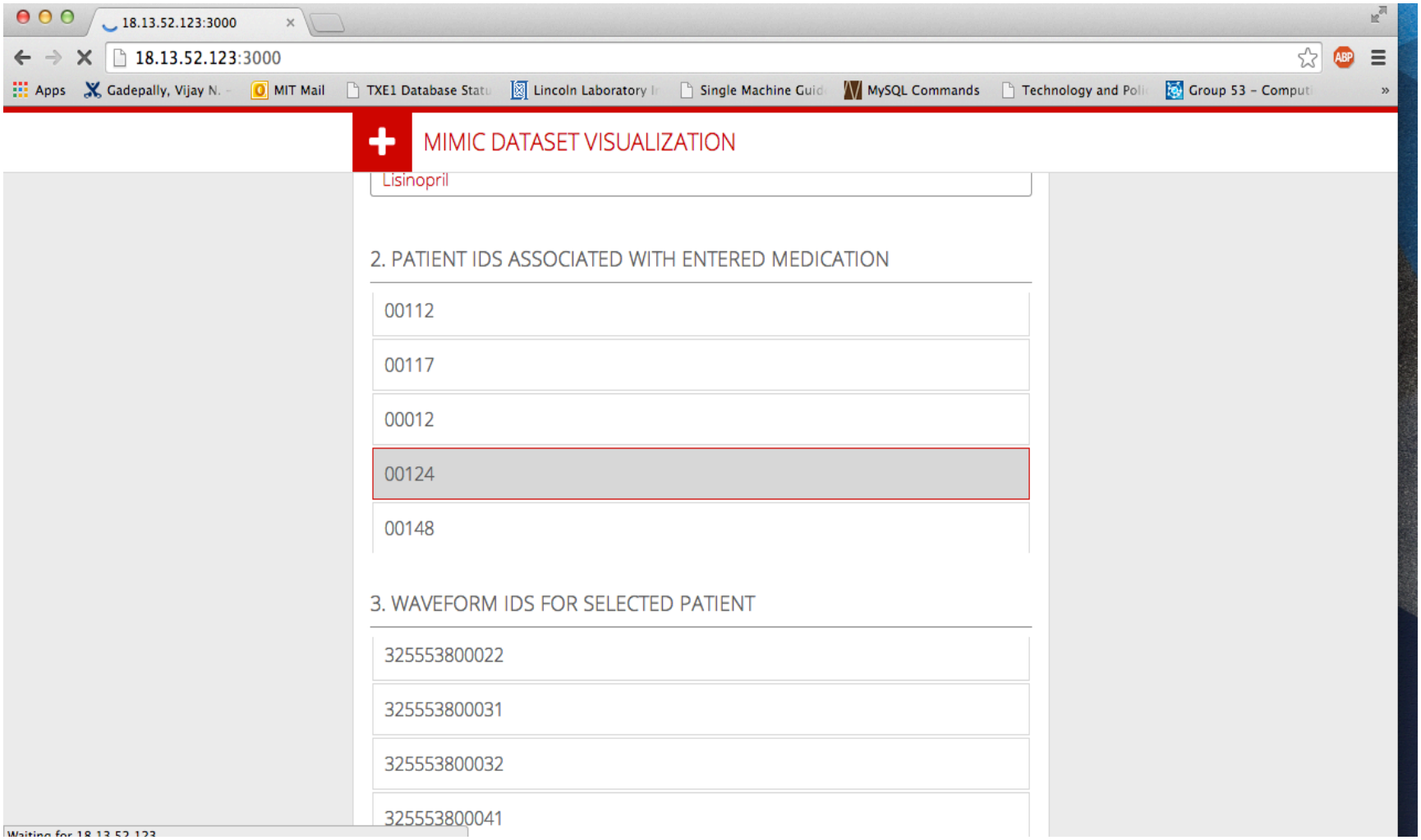
Solution = SQL + NoSQL + NewSQL

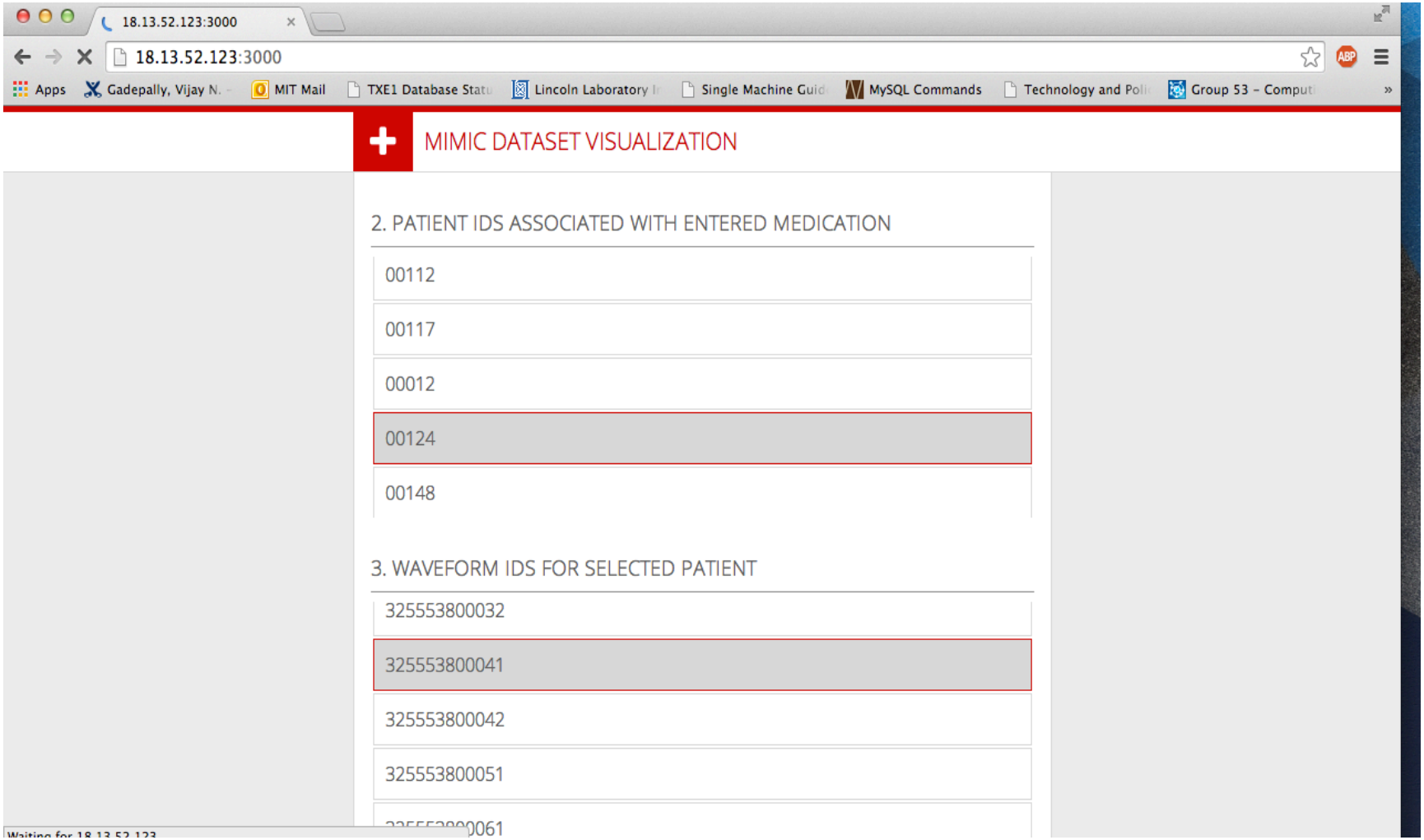
Demo

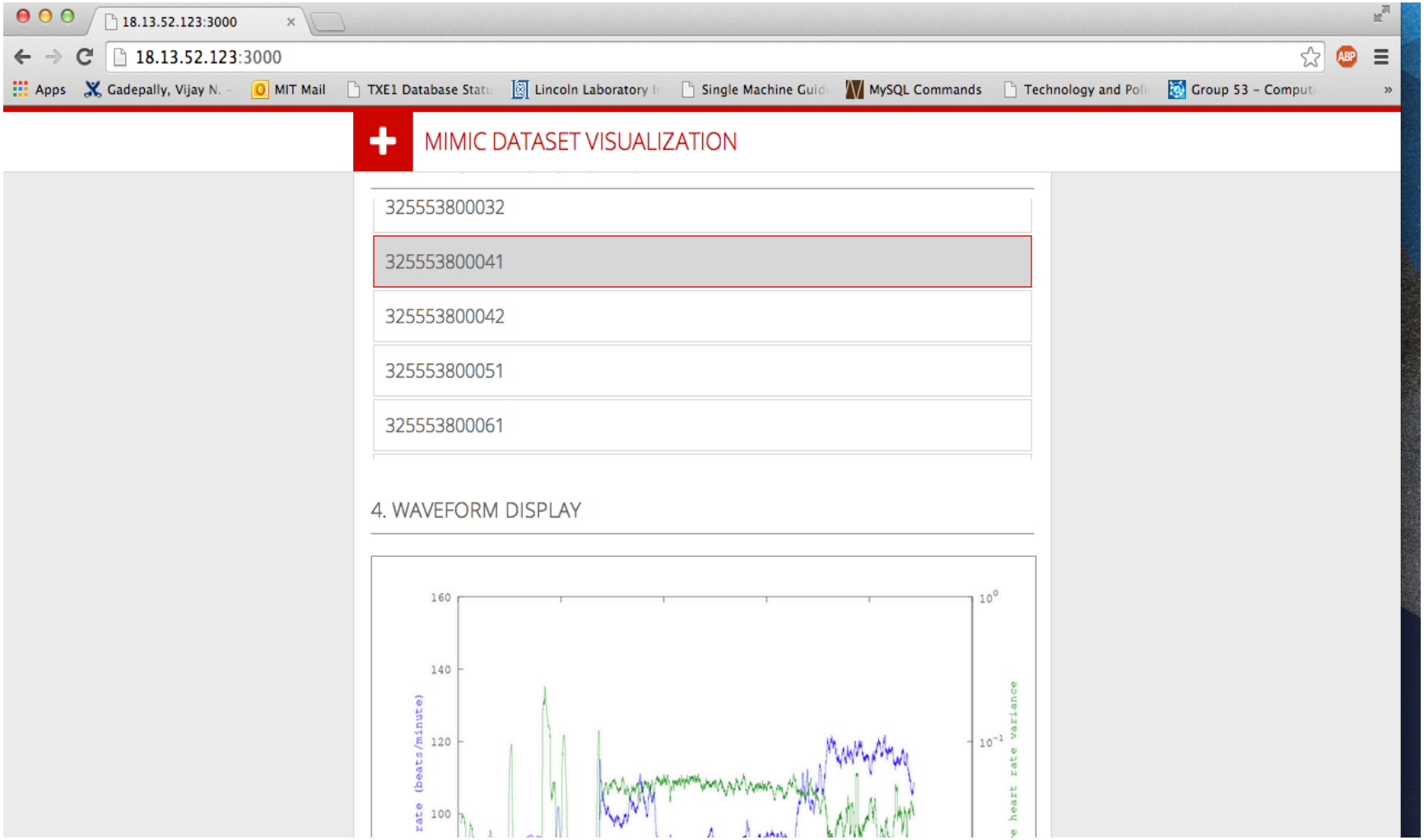

18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory I
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 – Comput
MIMIC DATASET VISUALIZATION
MIMIC2 Visualization Web Frontend
1. MEDICATION NAME
Enter medication name and hit enter...
© 2014. This visualization was made and maintained by the MIT Databases Group and the MIT Lincoln Lab, source on

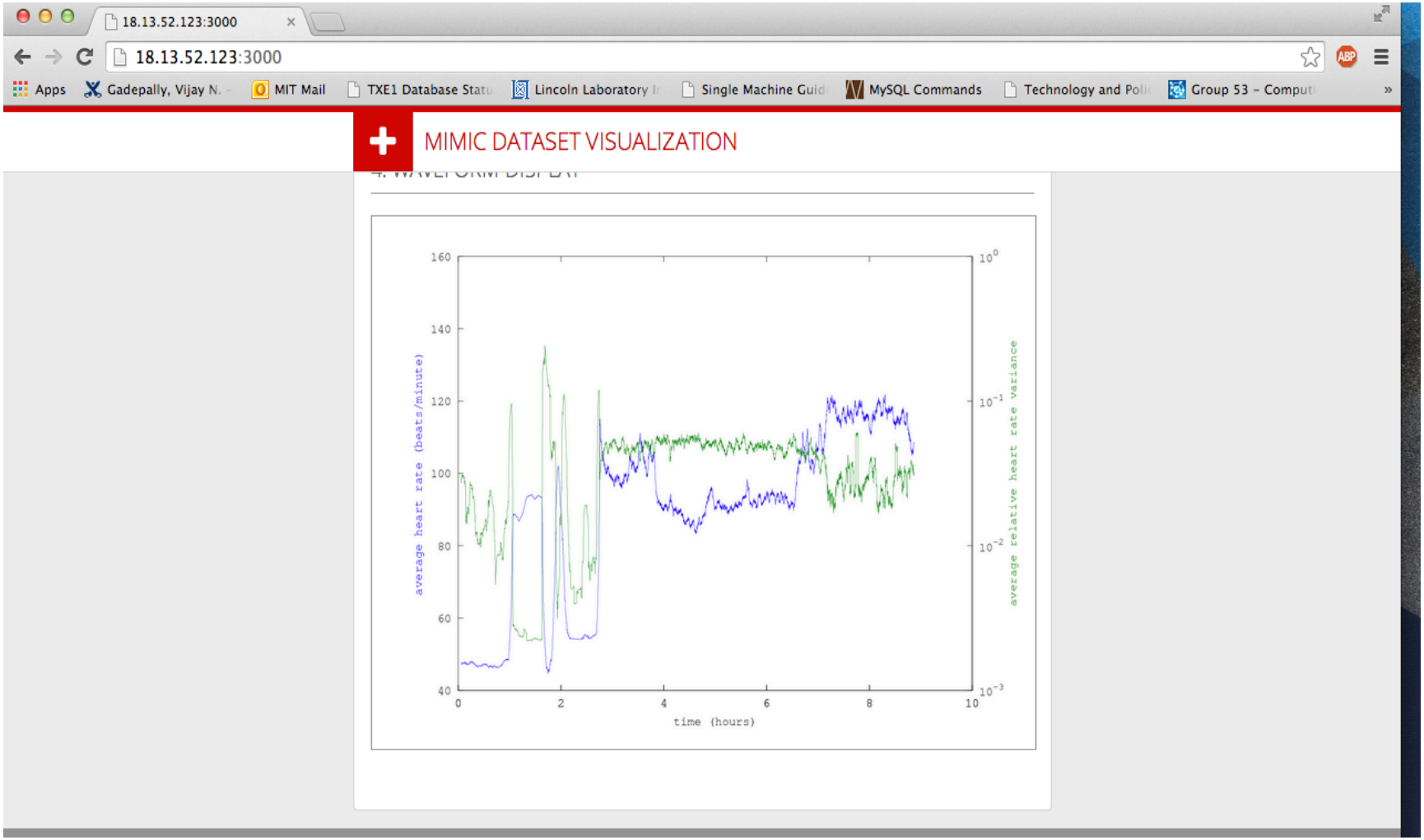

18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 – Comput
MIMIC DATASET VISUALIZATION
MIMIC2 Visualization Web Frontend
1. MEDICATION NAME
Lisinopril
© 2014. This visualization was made and maintained by the MIT Databases Group and the MIT Lincoln Lab, source on


18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 – Comput
MIMIC DATASET VISUALIZATION
MIMIC2 Visualization Web Frontend
1. MEDICATION NAME
Lisinopril
2. PATIENT IDS ASSOCIATED WITH ENTERED MEDICATION
00112
00117
00012
00124
00148
© 2014. This visualization was made and maintained by the MIT Databases Group and the MIT Lincoln Lab, source on


18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 – Comput
MIMIC DATASET VISUALIZATION
Lisinopril
2. PATIENT IDS ASSOCIATED WITH ENTERED MEDICATION
00112
00117
00012
00124
00148
3. WAVEFORM IDS FOR SELECTED PATIENT
325553800022
325553800031
325553800032
325553800041


18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 – Comput
MIMIC DATASET VISUALIZATION
2. PATIENT IDS ASSOCIATED WITH ENTERED MEDICATION
00112
00117
00012
00124
00148
3. WAVEFORM IDS FOR SELECTED PATIENT
325553800032
325553800041
325553800042
325553800051


18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 – Comput
MIMIC DATASET VISUALIZATION
325553800032
325553800041
325553800042
325553800051
325553800061
4. WAVEFORM DISPLAY
160
140
120
100
rate (beats/minute)
10^0
10^{-1}
heart rate variance


18.13.52.123:3000
18.13.52.123:3000
Apps
Gadepally, Vijay N.
MIT Mail
TXE1 Database Statu
Lincoln Laboratory
Single Machine Guid
MySQL Commands
Technology and Poli
Group 53 - Comput
MIMIC DATASET VISUALIZATION
average heart rate (beats/minute)
average relative heart rate variance
time (hours)
160
140
120
100
80
60
40
0
2
4
6
8
10
10^{0}
10^{-1}
10^{-2}
10^{-3}

Government Big Data Systems Example Instances

Module	DISA Cloud RFP	DISA CSAAC	Red Disk DCGS-A ICITE	Ghost Machine	DIA Orion	MIT LL CySA	MIT LL SuperCloud
Data Source Repository	High Speed SAN	HDFS (data) NFS (code)	HDFS (data) Lustre (code) Lustre (data)	HDFS (data) NFS (code)	HDFS	Lustre (data) Lustre (code)	Lustre (data) Lustre (code)
Ingest & Enrichment	Compute Host / VM	Storm, Niagara Java / Hadoop MapReduce	Storm, Niagara Java / Hadoop MapReduce	Java / Hadoop MapReduce	Java / Hadoop MapReduce	Any Language / Any Model	Any Language / Any Model
Advanced Analytics	Compute Host / VM	Java / Hadoop MapReduce	Java / Hadoop MapReduce	Java / Hadoop MapReduce	Java / Hadoop MapReduce	Any Language / Any Model	Any Language / Any Model
Scheduler Resource Manager	Compute Host / VM	cron Hadoop	cron Hadoop	cron Hadoop		cron GridEngine	cron GridEngine
Elastic Computing	Compute Host / VM	Hadoop	Hadoop	Hadoop	Hadoop	LLGrid	LLGrid
High-Performance Database	Oracle, MySQL	Accumulo (2x), Postgres	Accumulo (2x)	Accumulo (2x)	Cloudbase, Elastic Search, …	Accumulo (2x)	Any Database (Nx)
External Service Layer	Apache, TomCat, WebSphere, Oracle, IIS	JBOSS	REST, VM	REST	REST/SOAP	REST (+ syslogs, files, libs)	REST, VM

- **Specific Common Big Data Architecture components are chosen based on mission requirements**

5 minute break

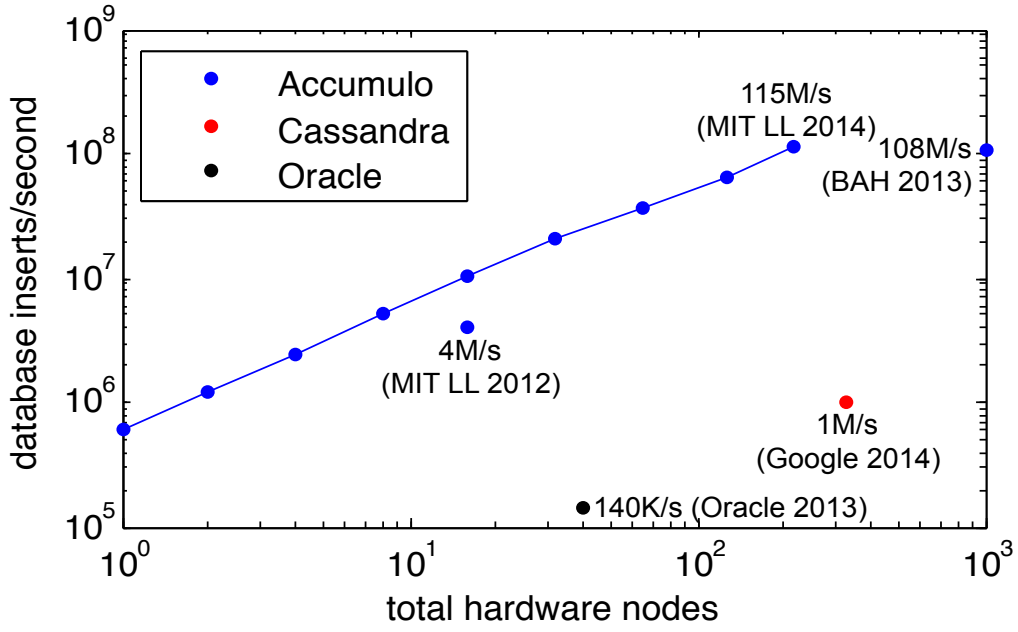
Part 3: Apache Accumulo

Apache Accumulo Outline

- About Accumulo
- Accumulo Design
  - Data Model
  - Components
- Accumulo Architecture
  - Relation to other Apache projects
  - Features
  - Using Accumulo

Apache Accumulo

- **Highest performance open source database**
- **Contributed to Apache project by the NSA in 2011**
- **Used extensively for government applications**
- **Requires a schema for storing and organizing data to obtain full benefits**



Accumulo Design Drivers

1 Cell-Level Security

- Express common security requirements in the infrastructure, not just in the application
- Data-centric approach encourages secure sharing

2 Scalability

- Near linear performance improvements at thousands of nodes
- Durable and reliable under increased failures that come with scale

3 Diverse, Interactive Analytics

- Sorted key/value core performs well in a diverse set of domains
- Information retrieval, statistics, graph analysis, geo indexing, and more

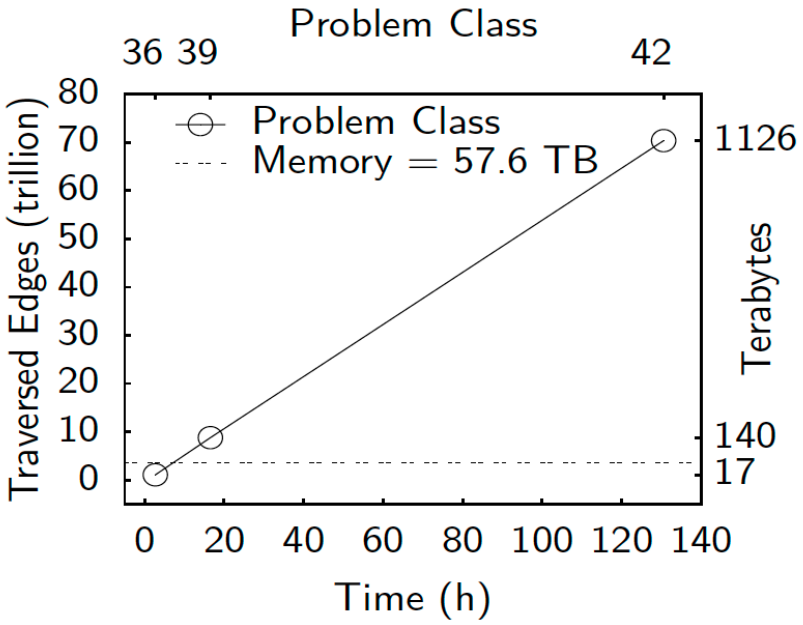
4 Flexible, Adaptive Schema

- Start with universal structures and indexing
- Refine the schema over time

Source: Sqrrl Data Inc

Scalability

Volume
(1200-nodes, petabyte-scale)



Velocity
(100 million inserts per second)

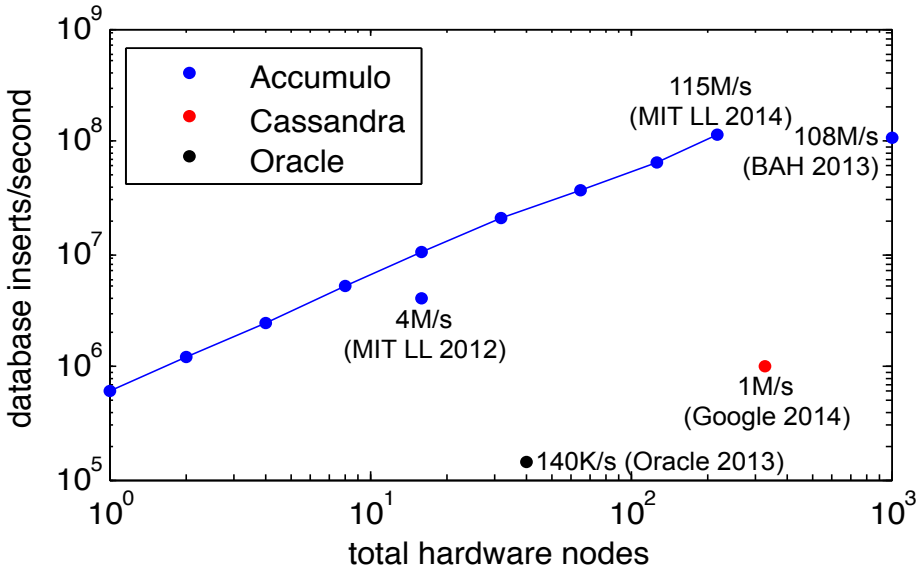


Fig 1 source: P. Burkhardt and C. Waring, "An NSA Big Graph experiment," National Security Agency, Tech. Rep. NSA-RD-2013-056002v1, 2013.

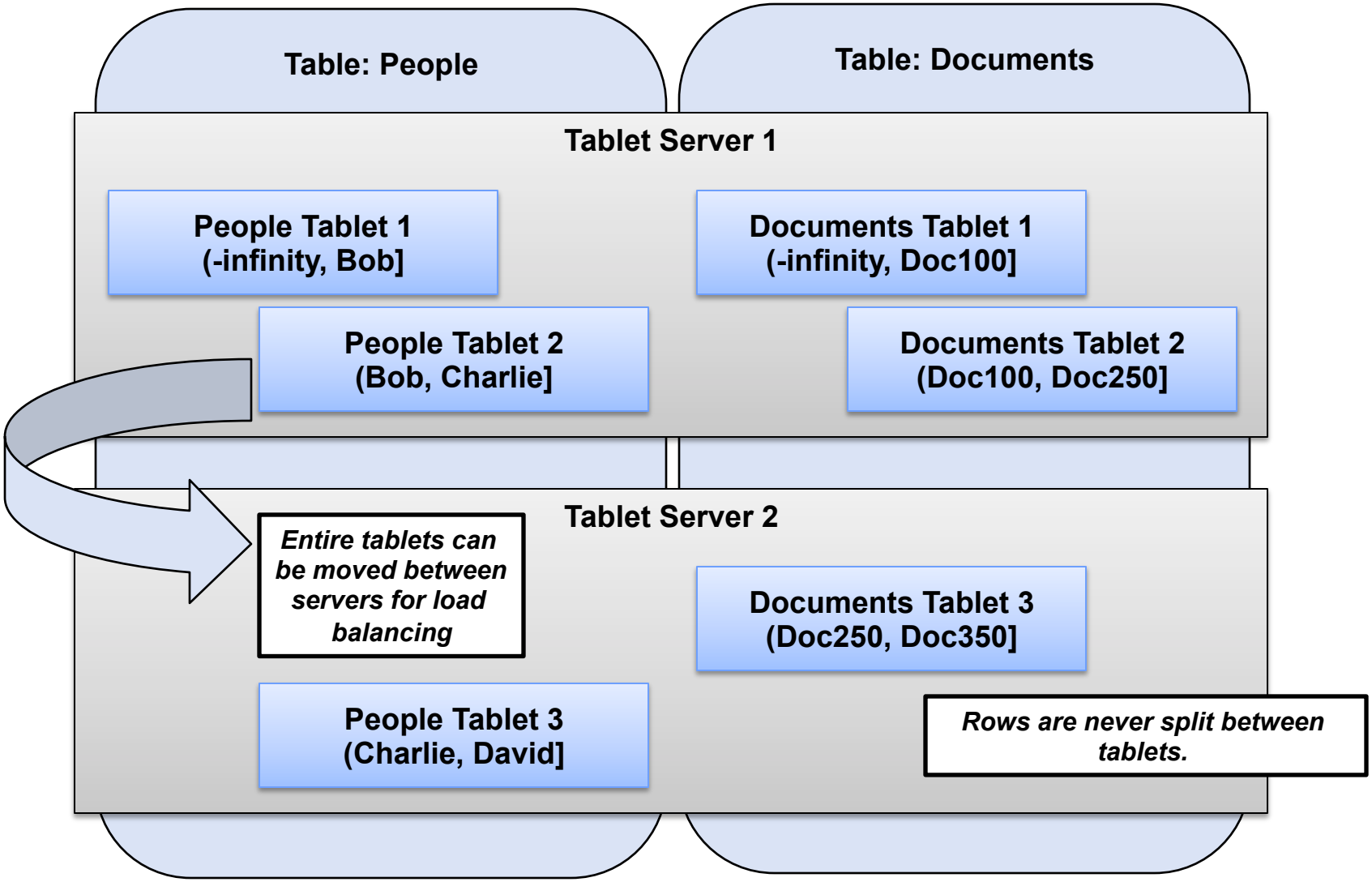
Accumulo Components

- **Tablet Servers:**
  - Responsible for managing some subset of all tables or partitions.
  - Write data, persist writes to write-ahead log, sort new key-value pairs into memory, flush sorted key-value pairs to new files in HDFS, respond to reads
  - Also recover tablets from servers that failed
- **Loggers**
  - Write Tablet Server updates and write them to disk storage
- **Garbage Collectors**
  - Periodically delete files no longer needed
- **Master**
  - Load balancing, detection and response to tablet server failures, ensure that each tablet has one tablet server, handles table creation, alteration, deletion

Accumulo Definitions

- **Table:** A map of key-value pairs with a global sort order among keys
- **Tablet:** A row range within a Table
- **Tablet Server:** The mechanism that hosts Tablets, and provides the following functionality:
  - Hosting remote procedure calls (RPCs) such as read, write, etc.
  - Managing resources (RAM, CPU, File I/O, etc.)
  - Scheduling background tasks (compactions, caching, etc.)
  - Handling key-value pairs (usually through iterator framework)

Tables, Tablets and Tablet Servers



Accumulo Data Model

- Accumulo is based on Google's BigTable design and is a *tabular key-value store*
- Each "cell" or "entry" in Accumulo is a key (a tuple) mapped to a value:
  - (row, column) → value
  - Each row and column has a string label
- Row store – constant time lookup of rows
- This flexible data model can be interpreted in many ways:
  - Semantic triples (subject, predicate, object)
  - Documents (maps of arbitrary key-value pairs)
  - Large sparse tables
  - Incidence matrices for graphs

Visualizing a BigTable

- **Entries are persisted as tuples and create a logical table**
- **Example: store a graph counting words in a document**

Sparse Tuples:

(doc1, apple) → 1
(doc1, banana) → 5
(doc1, carrot) → 2
(doc2, banana) → 1
(doc2, daikon) → 2
(doc3, carrot) → 4
(doc3, eggplant) → 1

Logical Table:

Row ID	apple	banana	carrot	daikon	eggplant
doc1	1	5	2		
doc2		1		2	
doc3			4		1

D4M: Unifying Four Abstractions

- **Extends associative arrays to 2D and mixed data types**

```
A('alice ','bob ') = 'cited '
```

or

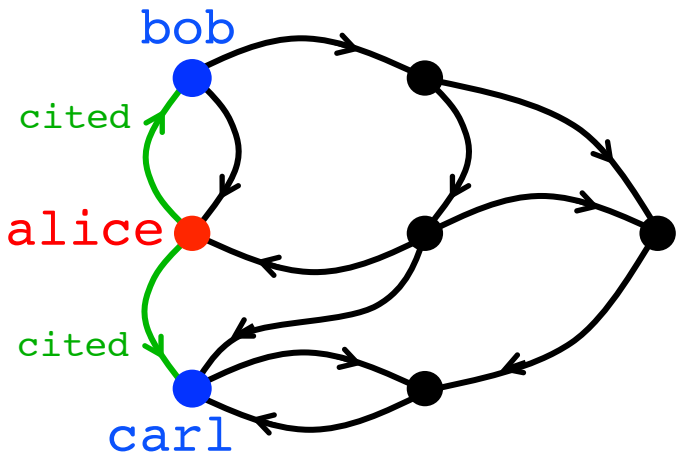
```
A('alice ','bob ') = 47.0
```

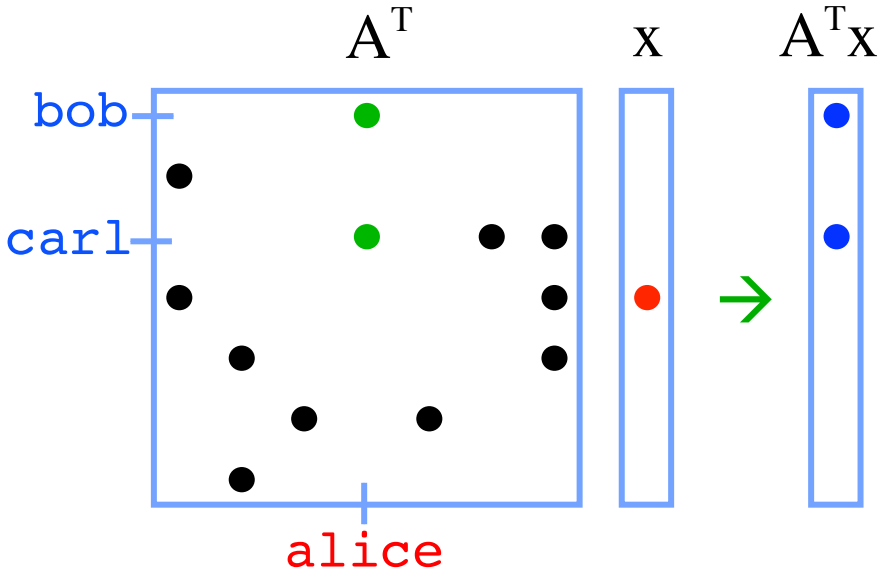
- **Key innovation: 2D is 1-to-1 with triple store**

```
('alice ','bob ','cited ')
```

or

```
('alice ','bob ',47.0)
```





Accumulo Data Model

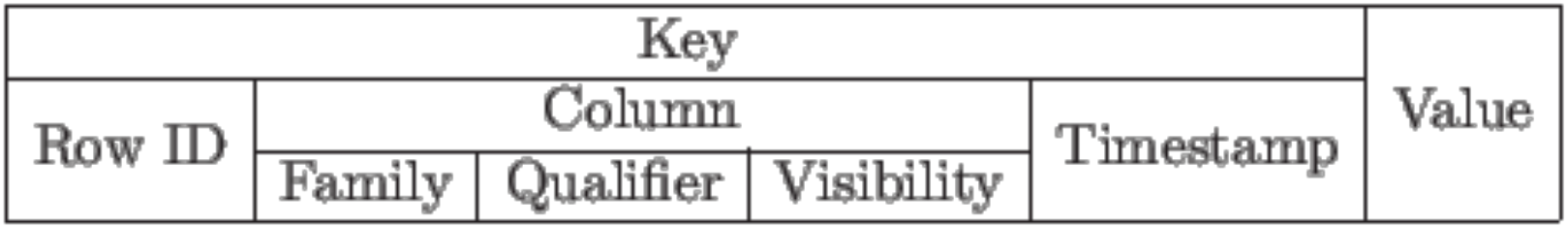
- **Accumulo is based on Google's BigTable* design and is a *tabular key-value store***
- **Each "cell" or "entry" in Accumulo is a key (a tuple) mapped to a value:**
  - **(row, column) → value**
  - **Each row and column has a string label**
- **This flexible data model can be interpreted in many ways:**
  - **Semantic triples (subject, predicate, object)**
  - **Documents (maps of arbitrary key-value pairs)**
  - **Large sparse tables**
  - **Incidence matrices for graphs**

*Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." ACM Transactions on Computer Systems (TOCS) 26.2 (2008): 4.

Accumulo Data Model

- **Referred to as key-value store database:**

(key) ➔ value



- **Keys sorted lexicographically**
- **Entries are organized into tables. (we'll see this in a bit)**
- **Portion of table persisted as a tablet, stored on tablet servers.**

Image Source: Accumulo manual: http://accumulo.apache.org/1.4/user_manual/Accumulo_Design.html#Data_Model

Accumulo Key Structure

Analyzing the 5-tuple

Row: Controls Atomicity
Column Family: Controls Locality
Column Qualifier: Controls Uniqueness
Visibility Label: Controls Access
Timestamp: Controls Versioning

Row	Col. Fam.	Col. Qual.	Visibility	Timestamp	Value
JohnDoeID	Notes	PCP	PCP_JD	20120912	Patient suffers from an acute …
JohnDoeID	Test Results	Cholesterol	JD|PCP_JD	20120912	183
JohnDoeID	Test Results	Mental Health	JD|PSYCH_JD	20120801	Pass
JohnDoeID	Test Results	X-Ray	JD|PHYS_JD	20120513	1010110110100…

Source: Sqrrl Data

A Simpler View of Accumulo Data

- Consider the data to be a 3-tuple:
  - (rowkey, columnkey, value)
- Use the rowkey to control atomicicy
- Use the columnkey to control locality and uniqueness
- Use the D4M schema to take an convert an arbitrary dataset into a triples representation
- This representation works for most cases

Rowkey: Row	Columnkey: Column Fam.|Column Qual		Value
JohnDoeID	Notes|PCP		Patient suffers from an acute …
JohnDoeID	Test Results|Cholesterol		183
JohnDoeID	Test Results|Mental Health		Pass
JohnDoeID	Test Results|X-Ray		1010110110100…

Accumulo Partitioning and Distribution

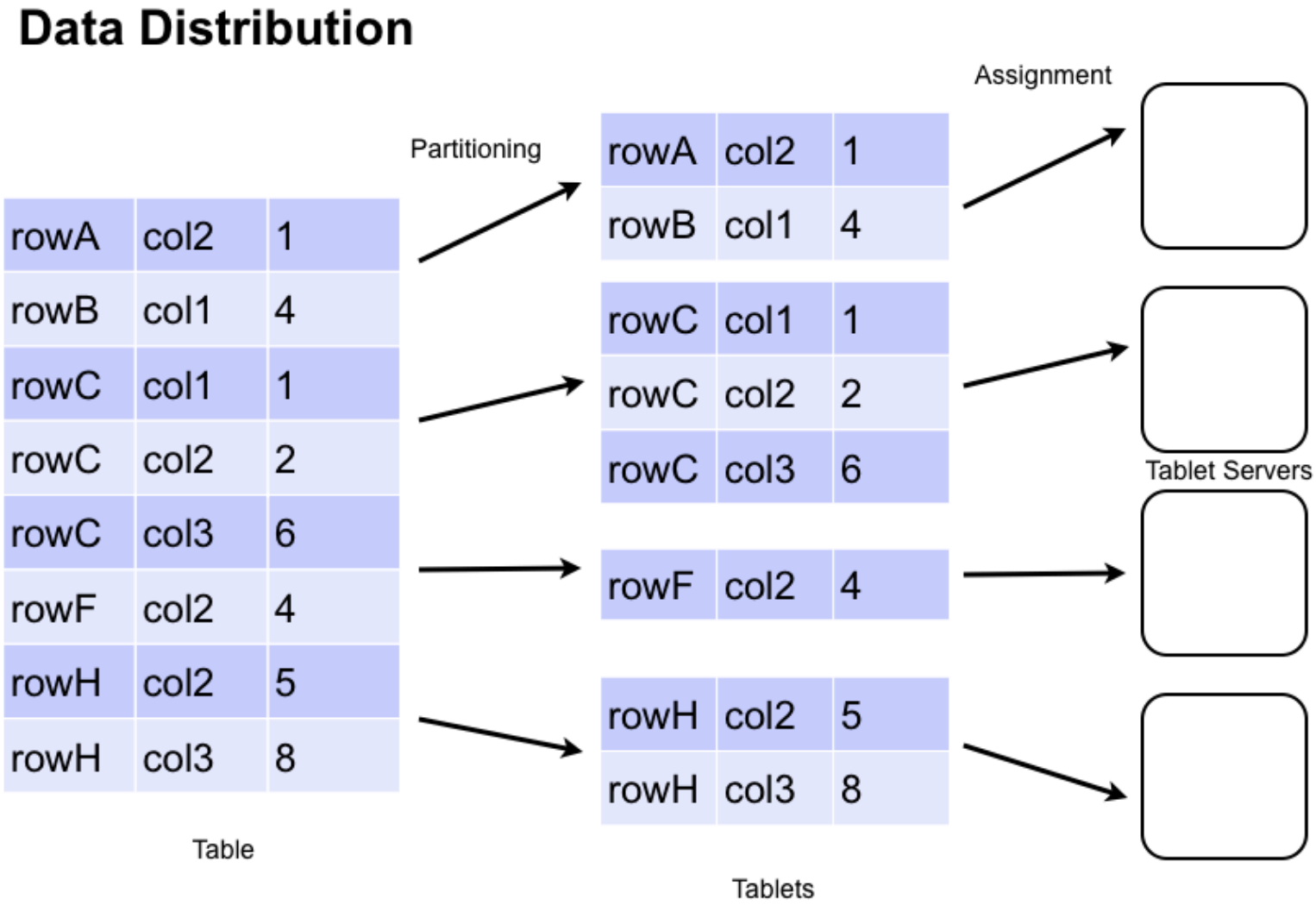
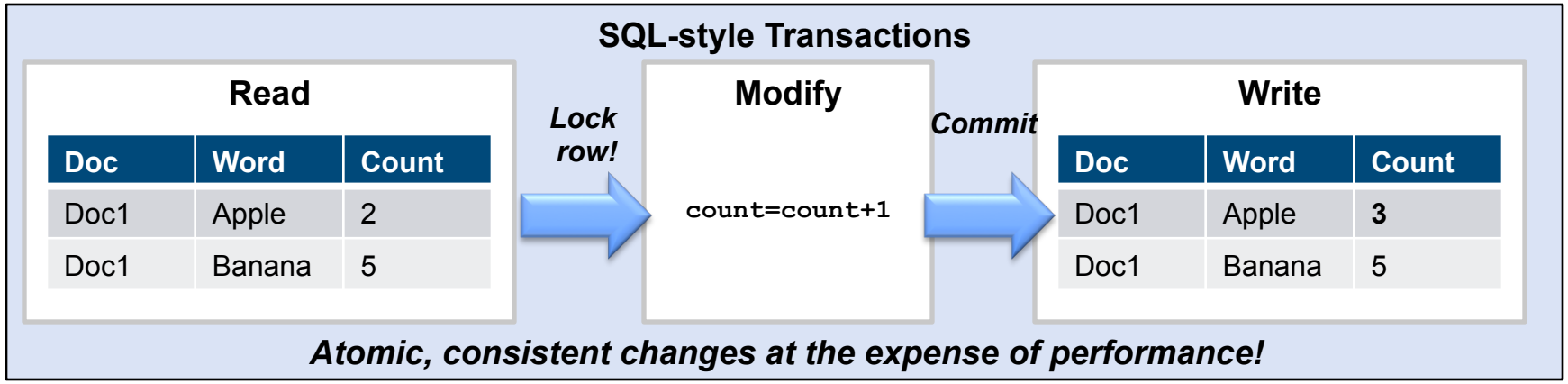
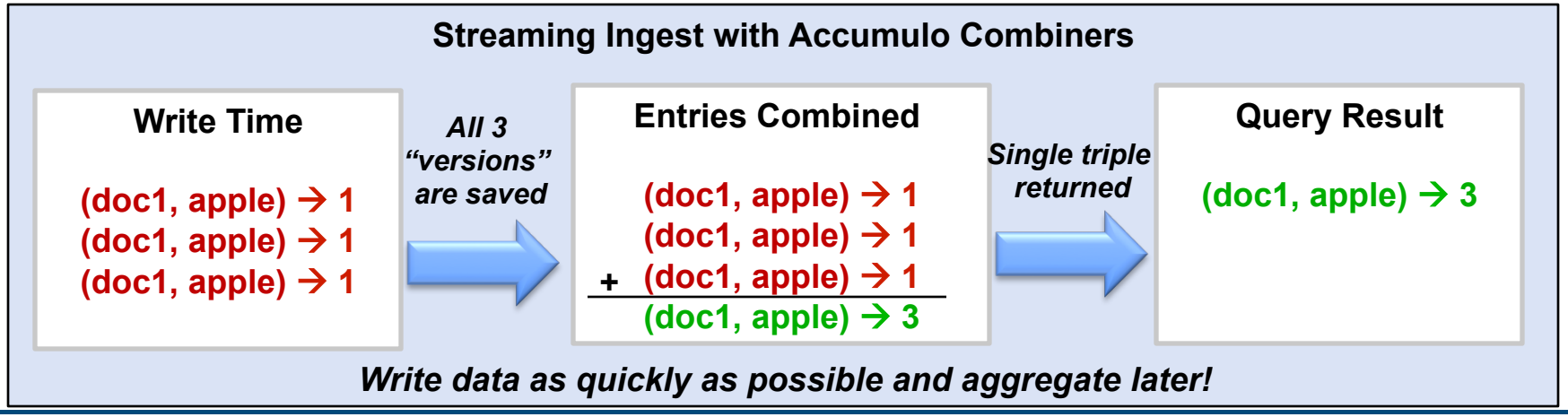


Image Source: http://accumulo.apache.org/user_manual_1.3-incubating/Accumulo_Design.html

Writing Data to Accumulo

- Accumulo is specialized for high-speed insert/ingest of streaming data
- No support for RDBMS-style "modify" transactions

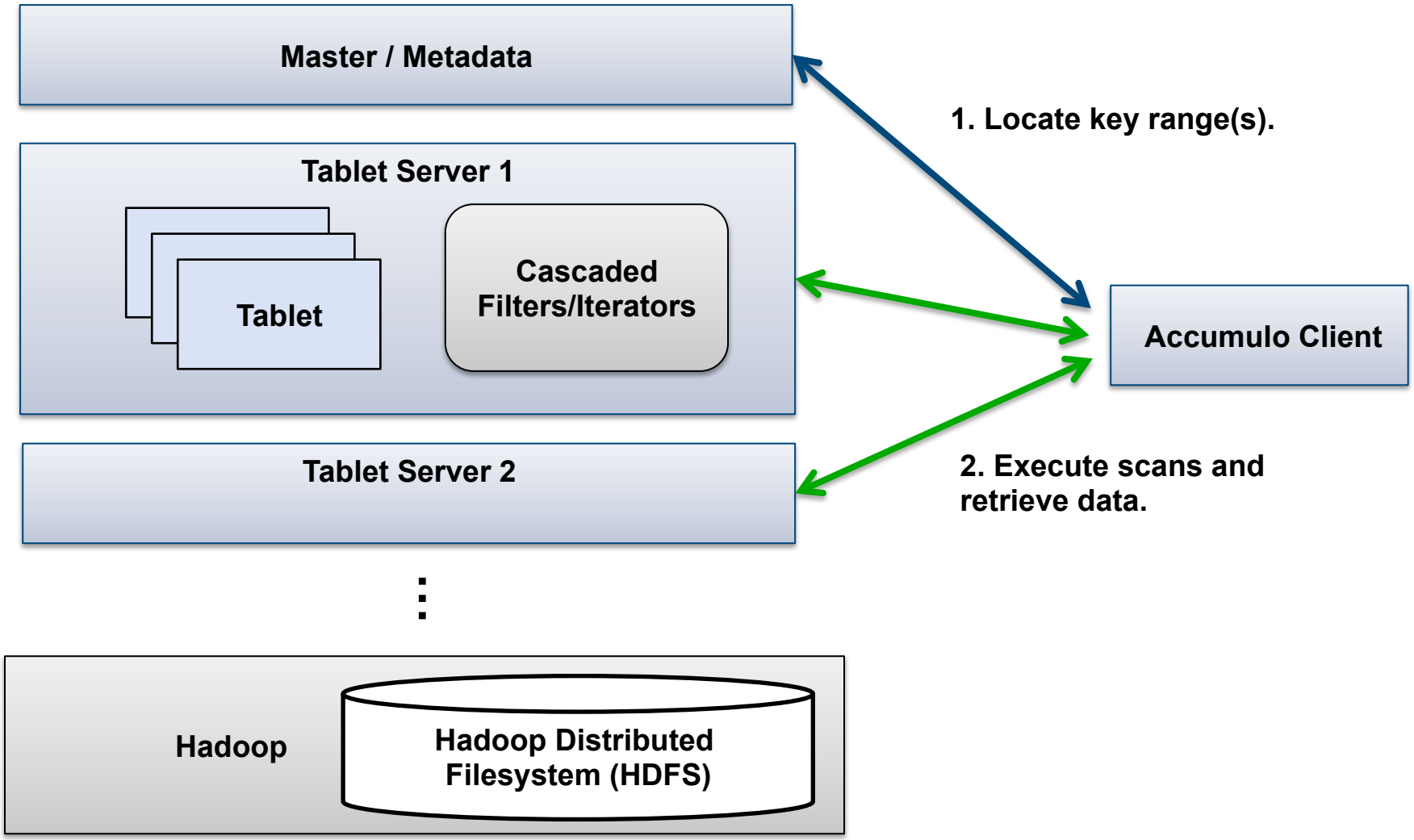




Accumulo Architecture

- Written in Java and sits on top of Hadoop File System (HDFS)
  - Consists of a Master Node, Tablet Servers, Iterators, Tablets, etc.
- Uses Apache Zookeeper for coordination
- Connect to Accumulo via Apache Thrift proxy or JDBC driver

Accumulo Architecture



Slide Source: S. Sawyer, B. D. O'Gwynn, A. Tran, T. Yu.
Understanding Query Performance in Accumulo. HPEC 2013.

Accumulo Writes

- Writing data to Accumulo consists of the following steps:
  - Data written to a Write-Ahead Log (WAL) that helps preserve Atomicity in case of failure
  - Data is inserted into a sorted data structure called MemTable
  - MemTable is periodically flushed (via a minor compaction) HDFS file called the Indexed Sequential Access Method (ISAM) file
  - New MemTable is created and record of compaction added to WAL
  - When number of files becomes large, a set of ISAM files are combined into one file (major compaction)

Compactions

- At ingest time, entries are initially stored in a memory cache
- Data written to small files as necessary (minor compaction)
- Small files aggregated into larger files (major compaction)

Accumulo Reads

- Reading data from Accumulo consists of the following steps:
  - Request arrives at Tablet Server
  - Tablet server performs binary search in MemTable and associated ISAM file(s) to find relevant values
  - Several possible key-value pairs are returned to client in order from the MemTable and ISAM files (a merge sort is performed as they are returned)

Failure Handling

Automatic Failure Handling

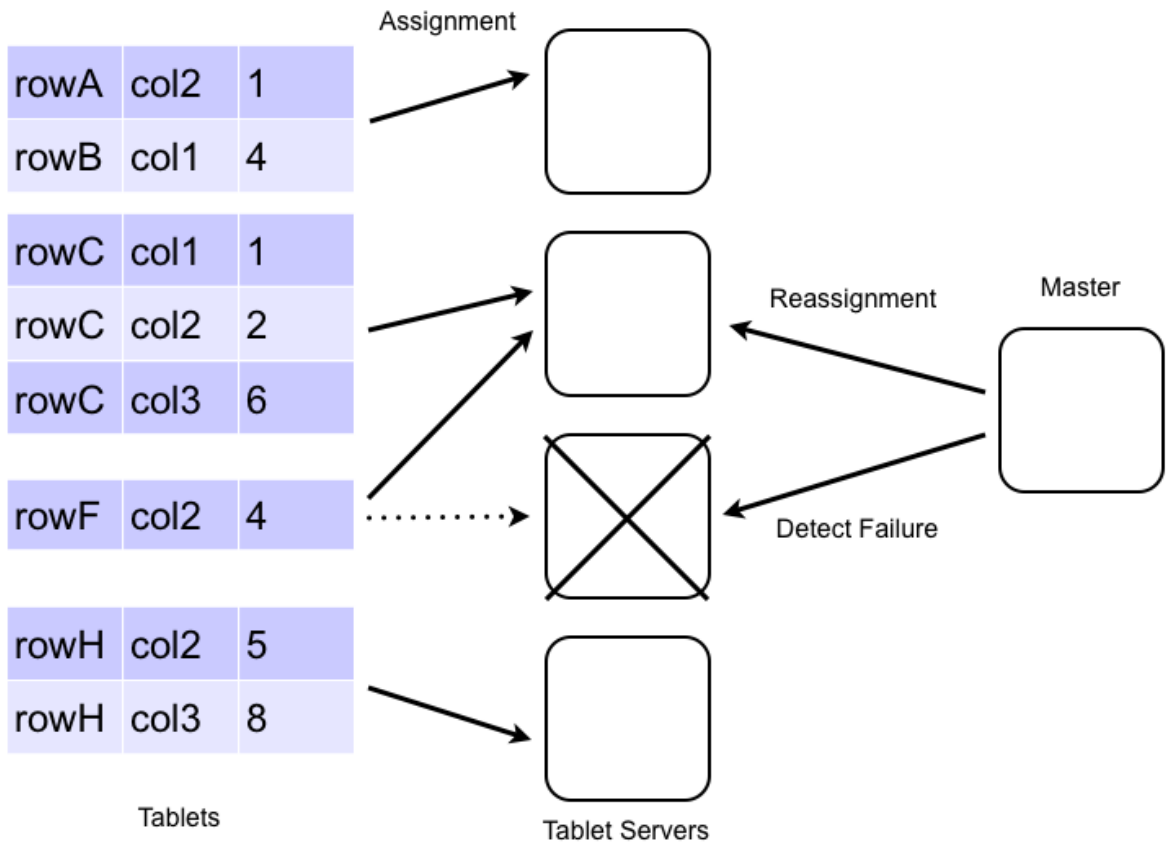


Image source: http://accumulo.apache.org/user_manual_1.3-incubating/Accumulo_Design.html

Accumulo and other Apache Projects

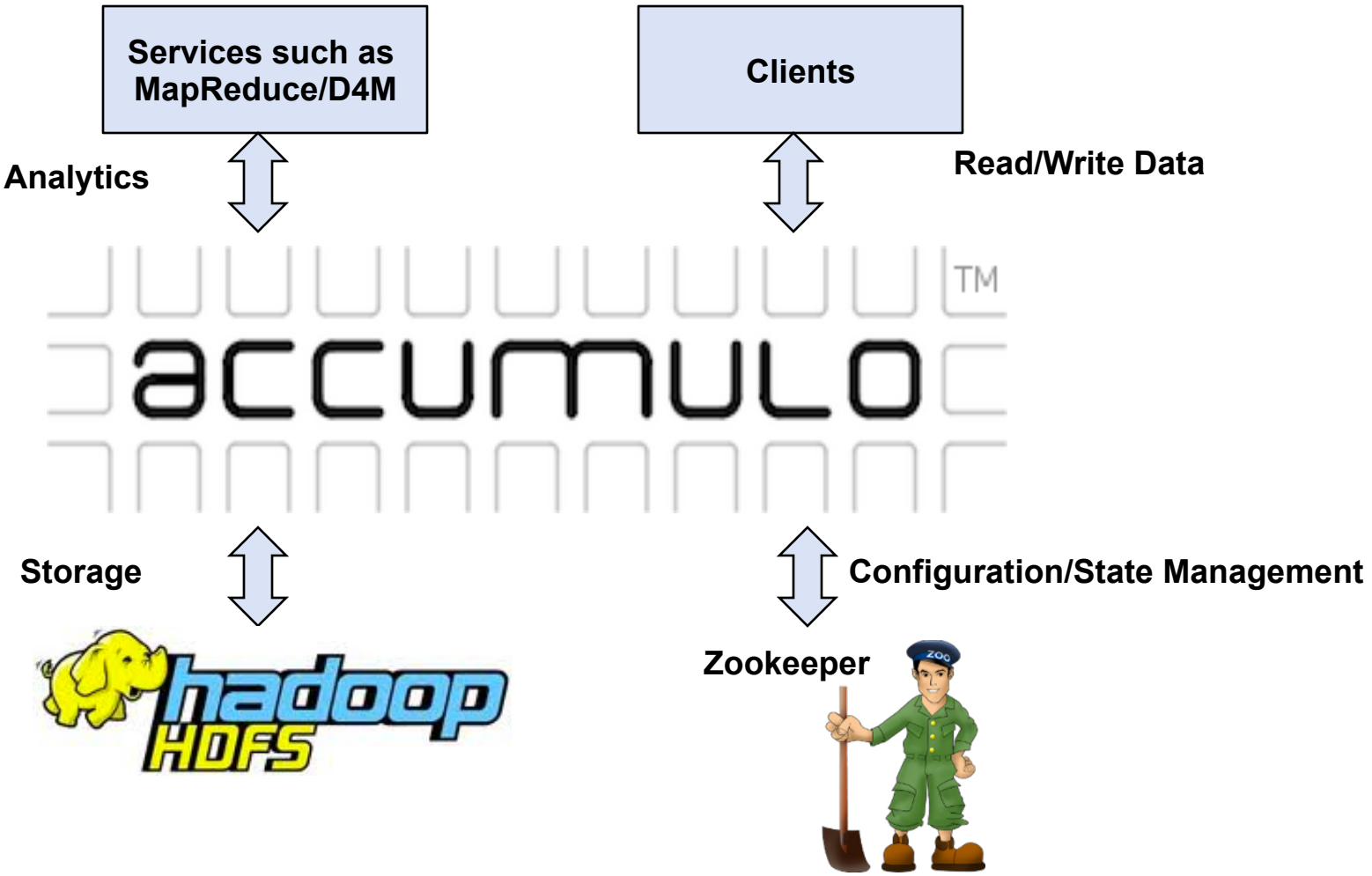


Figure adapted from: http://www.slideshare.net/jaredwinick/introduction-to-apache-accumulo

Accumulo Architecture

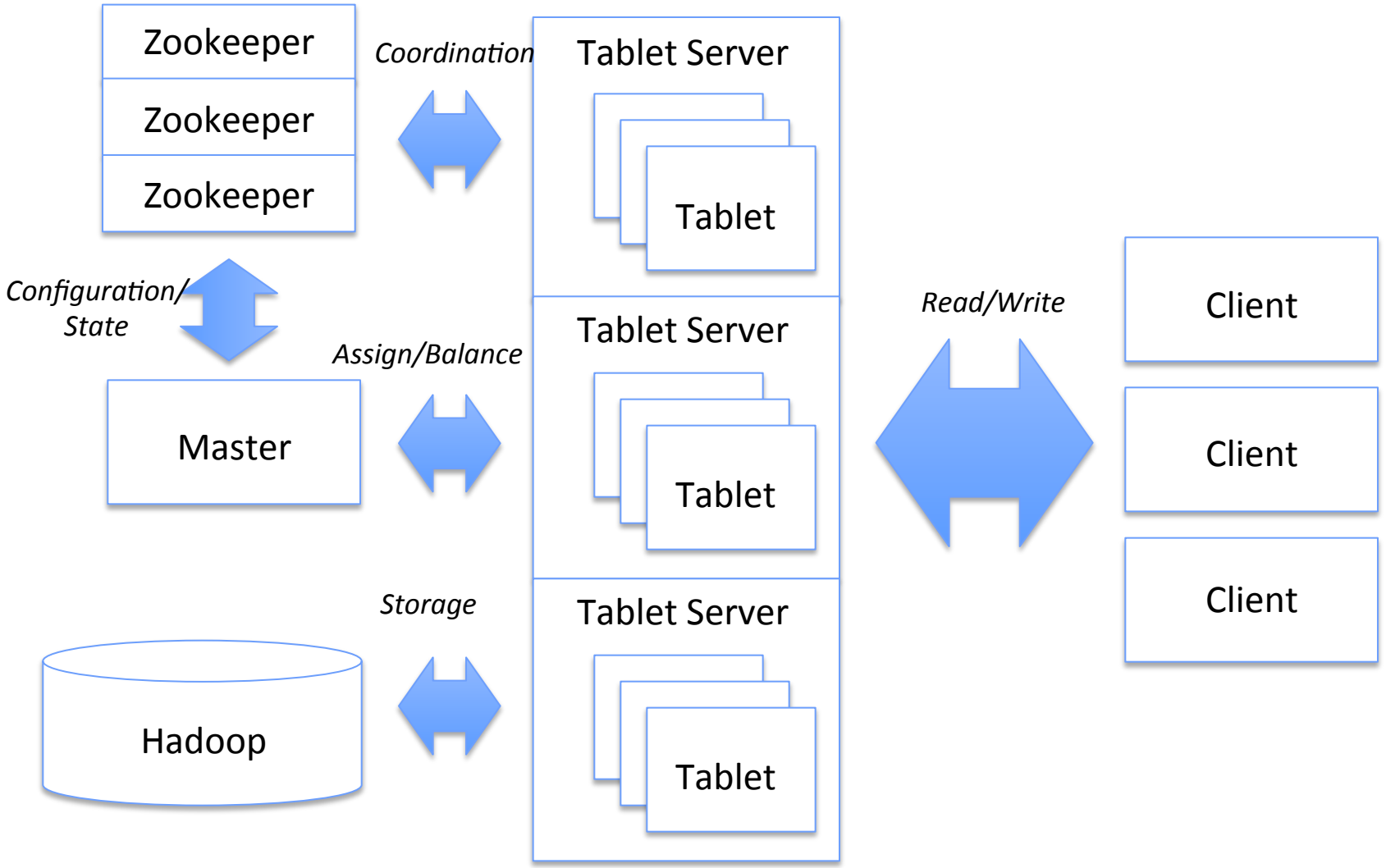
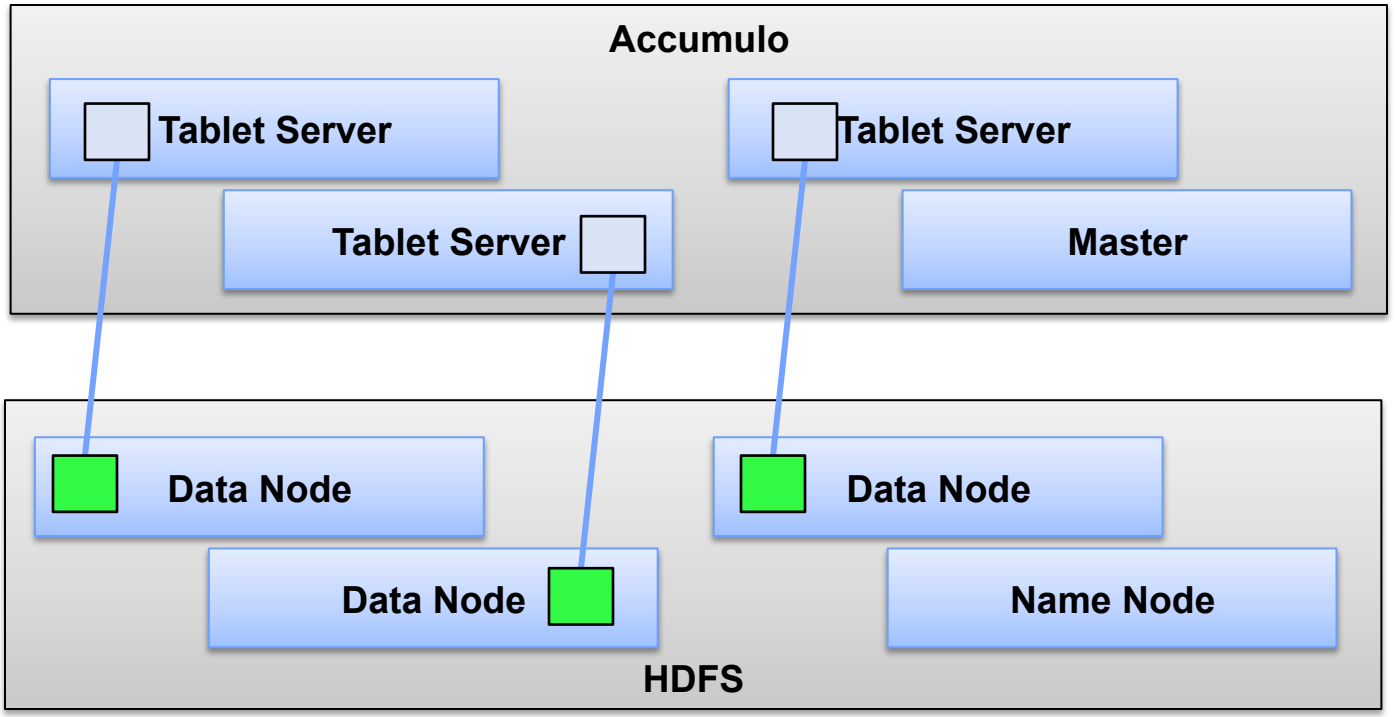


Image Source: Sqrrl Inc

Accumulo - HDFS

- Tablets are mapped to HDFS Data Node.
- Failures handled by moving tablet from one Tablet Server to another and Master reassigns the Data Node to new Tablet Server



Accumulo - Zookeeper

- Zookeeper is a coordination service for distributed applications and exposes an interface for naming, configuration management, locking and synchronization.
- Accumulo uses zookeeper for coordination between Tablet servers and configuration services
  - Keeps track of where Tablet Server are located
  - Coordinating data movement between Tablet servers
  - Locking to ensure only one master is active
  - Ensuring fault tolerance

Retrieving Data in Accumulo

- Data is retrieved in Accumulo by specifying key ranges
- Entries are lexicographically sorted by key ensuring fast lookup
- Examples:
  - (-infinity to +infinity)
  - (Doc1 to Doc5)
  - [Doc1 to Doc5)
  - Doc2
- Accumulo API exposes a Scanner (returns ordered entries from a single range) and a BatchScanner (returns entries from a set of ranges without ordering guarantee)

Questions

- What do we mean when we say that tuples in Accumulo are sparse?
- How are transactions (i.e., read-modify-write cycles) avoided in Accumulo?
- How does a user specify records for retrieval in Accumulo?

Accumulo Features

- Visibility Labels
- Iterators
- Automatic table splitting
- Support for Apache Thrift proxy

Visibility Labels

- Allows users to specify cell level labels to control access
  - Cell level means at the individual key-value pair level
- Can use boolean expressions to chain different access labels
- Stored in visibility part of key
- Not the same as user ACLs
- User given authorization sets and this is checked against labels of individual cells

Document Labels

Doc_1 : (Federation)
Doc_2 : (Klingon|Vulcan)
Doc_3 : (Federation&Human&Vulcan)
Doc_4 : (Federation&(Human|Vulcan))

User Authorization Sets

CptKirk : {Federation,Human}
MrSpock : {Federation,Human,Vulcan}

Image Source: Sqrrl Data, Inc.

Iterators

- Iterators are a real time processing framework that essentially allow you to take in a bunch of data, do some operation, and throw out the results

- For example:
  - Aggregating
  - File Reading
  - Deleting
  - Cell Level Security

- Can chain multiple iterators together

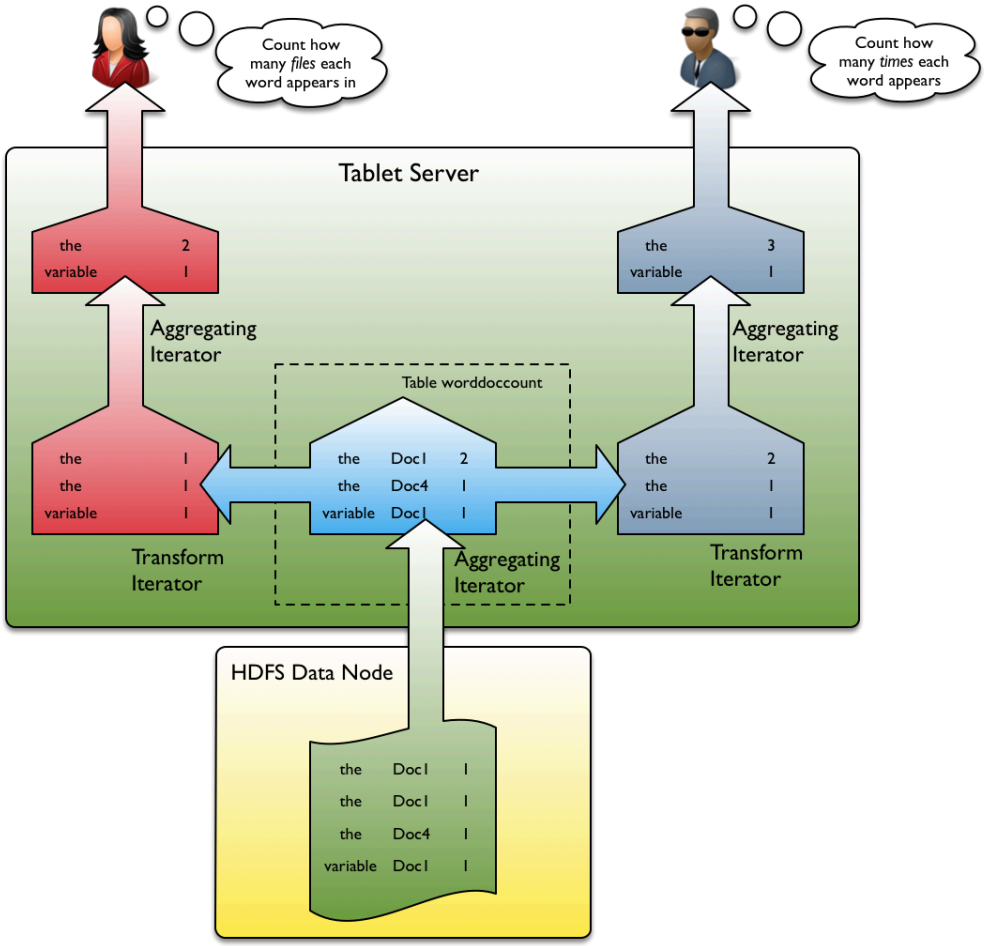
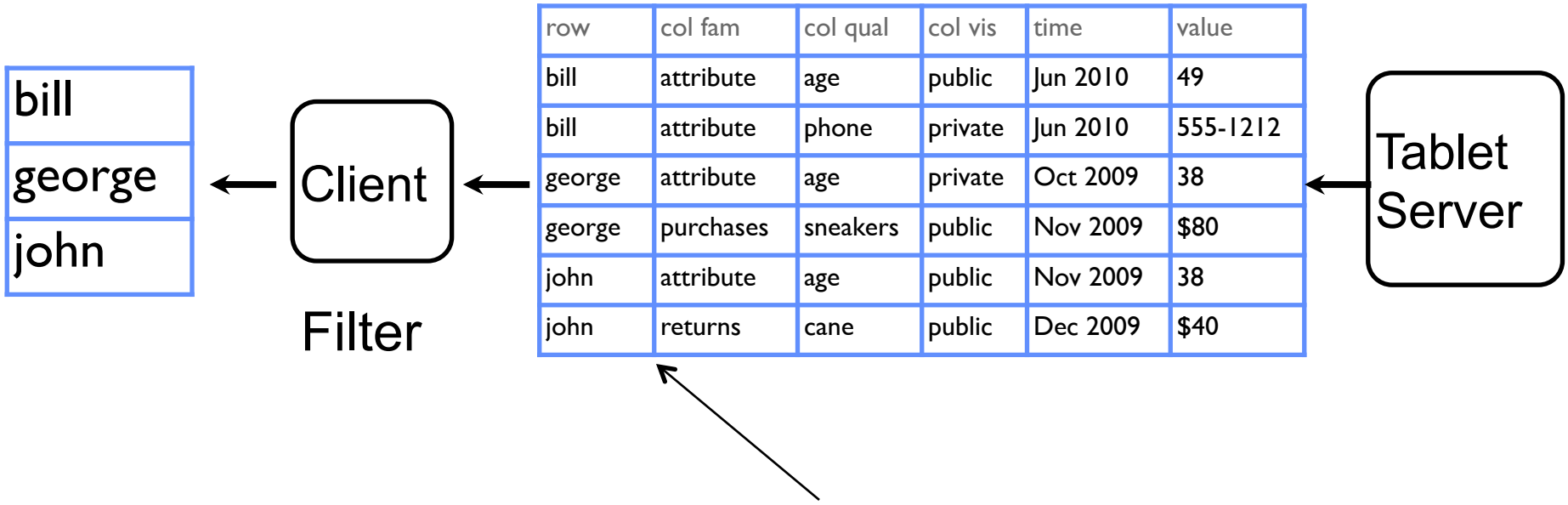


Image Source: Sqrrl Data, Inc.

An example without iterators

- **Tons of data is passed around, client has to do the filtering**



Can anyone think of an issue with this schema?

Image Source: Sqrrl Data, Inc.

An example with iterators

- **Data is filtered before leaving the Tablet Server**

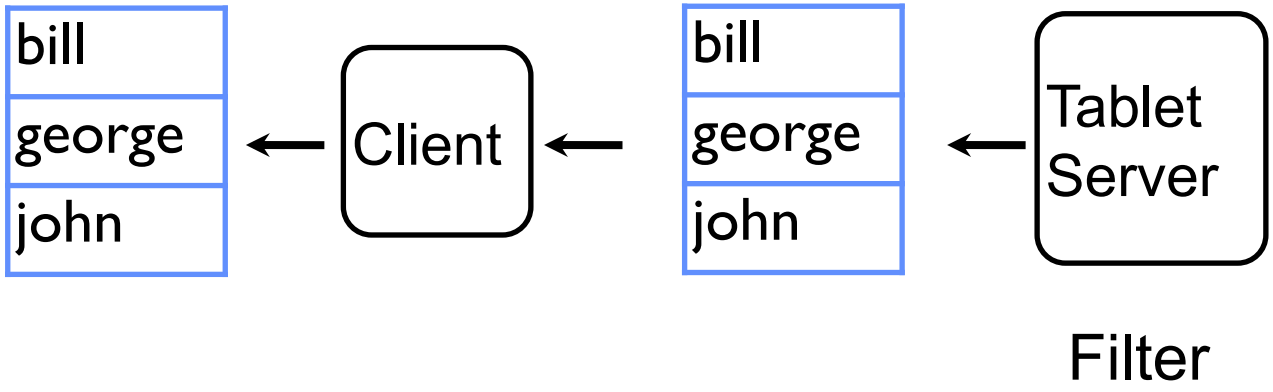
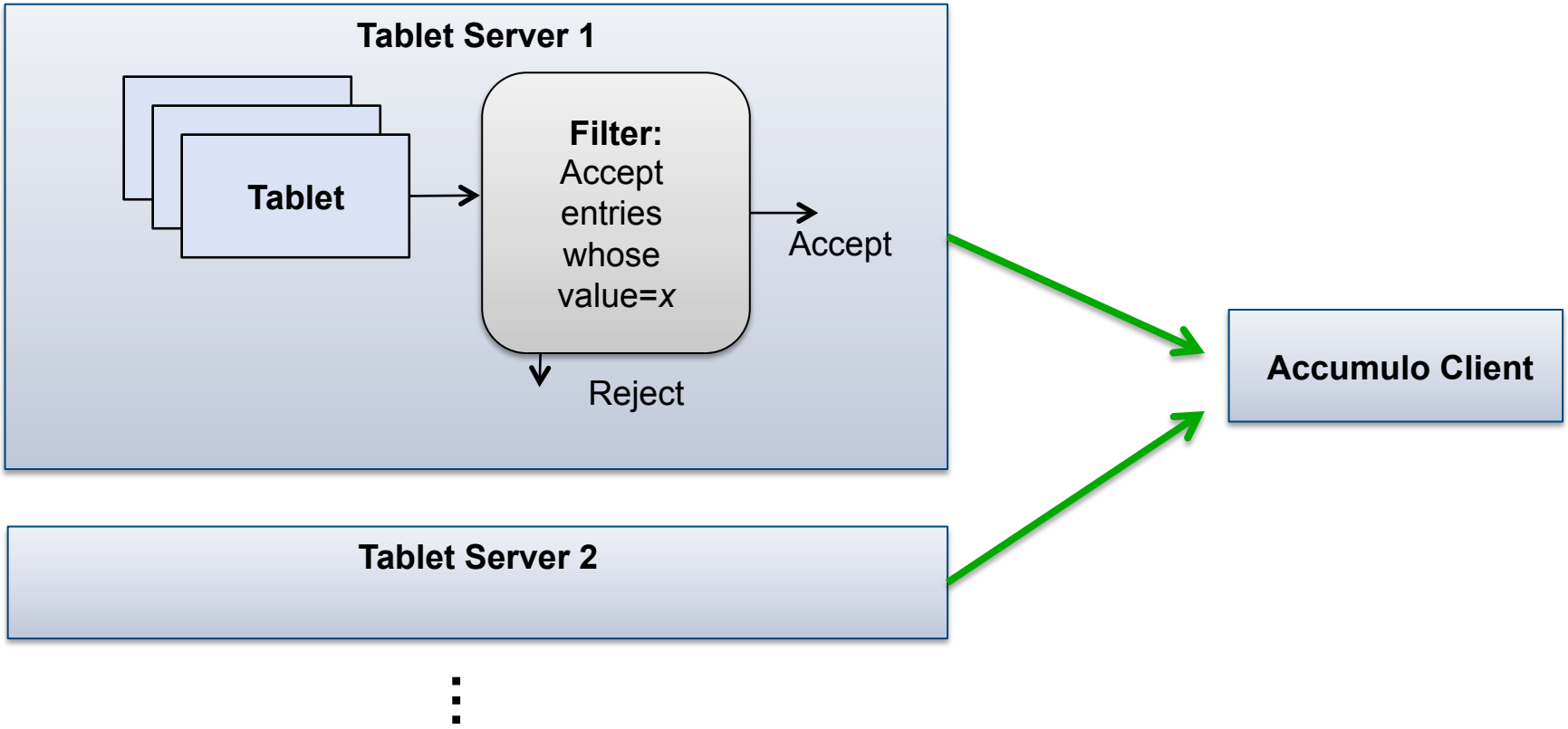


Image Source: Sqrrl Data, Inc.

Iterator Framework

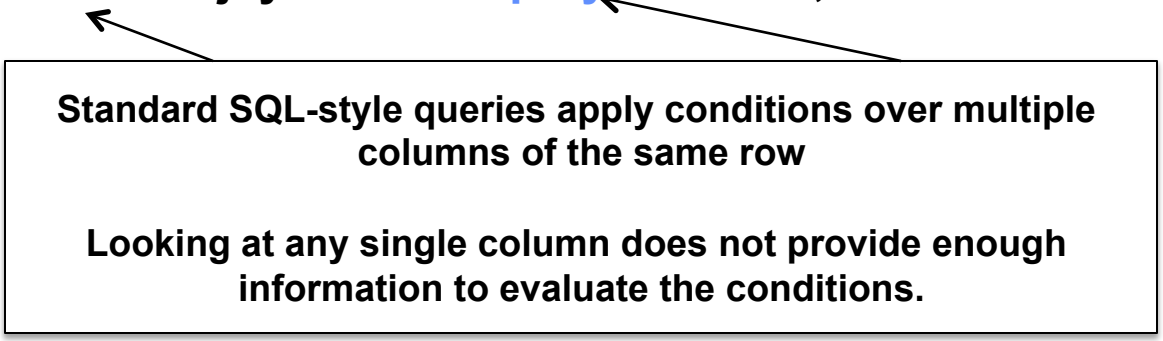
- "Iterator" refers to the software design pattern (frequently used in Java) of looping over a collection of items
- All server-side processing implemented as iterator
- Iterators can be cascaded together (example: pipe-and-filter pattern)
- Internal Accumulo functionality implemented in iterator framework
- Limitations: iterators can only access entries within a single row
  - No guarantee that neighboring rows are located on the same server

Iterator Use Case: Filtering Entries



Whole Row Iterator

- **Many queries require analyzing an entire row to decide whether conditions are satisfied:**
  - **SELECT * FROM myTable**
    WHERE name="Vijay" AND employer="MIT";



- **Accumulo's WholeRowIterator loads entire row into memory and evaluates user-specified filter function**

Table Configuration

- Accumulo automatically handles tablet splits in order to balance and distribute data across all tablet servers
- Often, it is useful to pre-split tables so that you do not need to wait for the table to become large enough that Accumulo performs the split.
  - Improved performance as you are not limited to the speed of a single node until split occurs
- You pre-split by selecting row keys to split on.
- For example:
  - addsplits -t newTable g n t
    creates 4 tablets with the following row ranges:
  - (-inf,g) (g,n) (n,t), (t,inf)

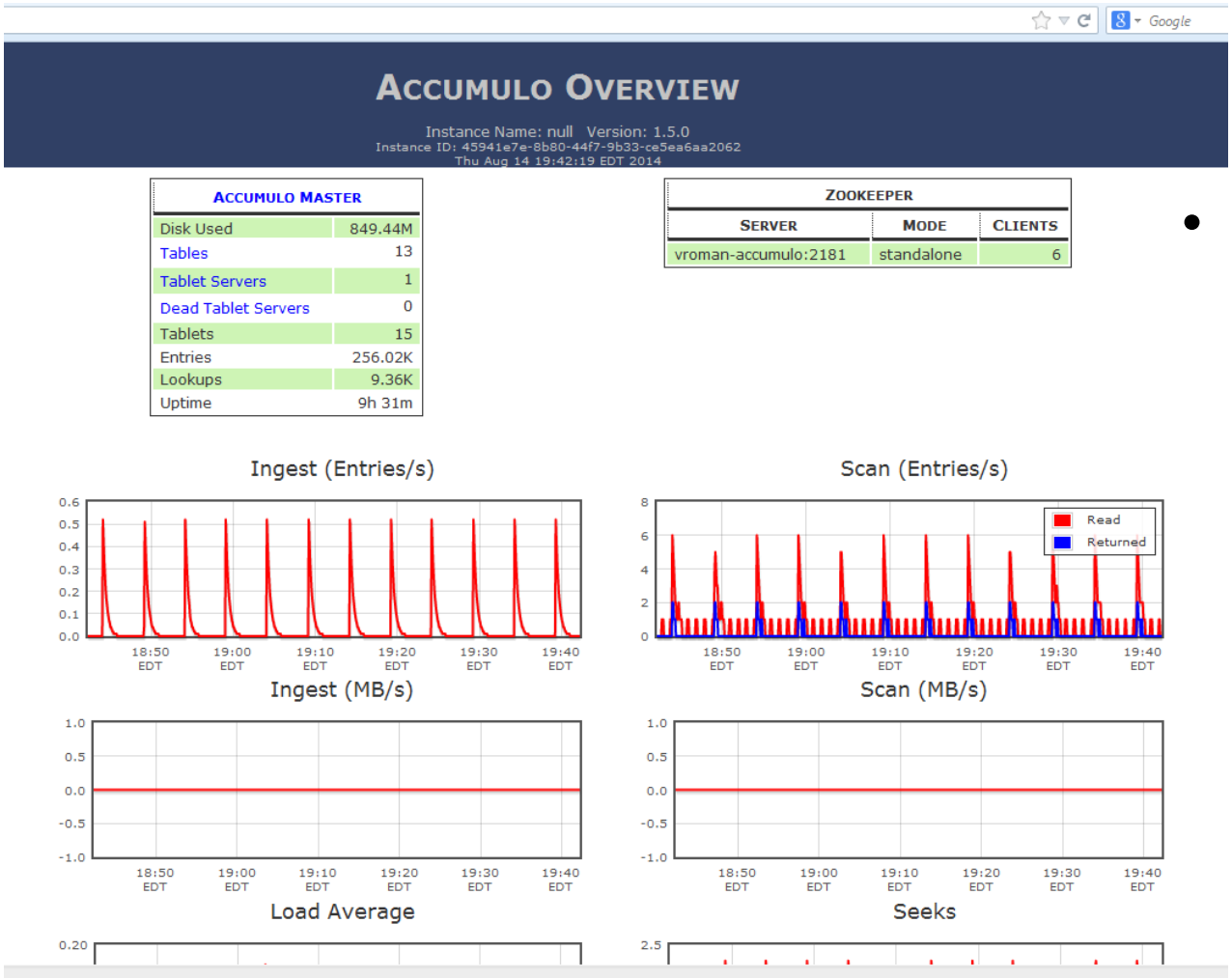
Questions

- How are entries distributed across a cluster?
- What are some use cases for server-side iterators?
- What are advantages of a distributed database compared to a single-node database?
- What is an advantage of pre-splitting tablets?

Using Apache Accumulo

- Online Monitoring Service
- Tuning/Scaling
- Schema
- Example
- Connecting to Accumulo

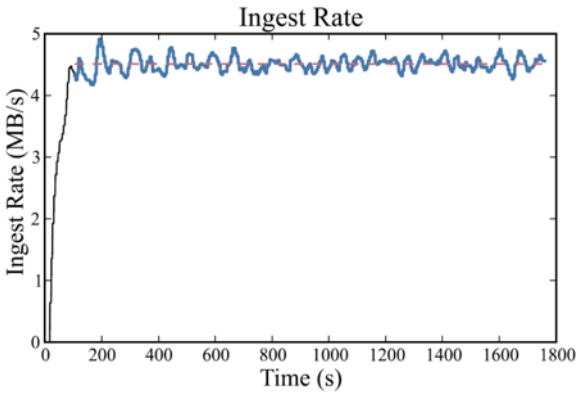
Accumulo Monitor



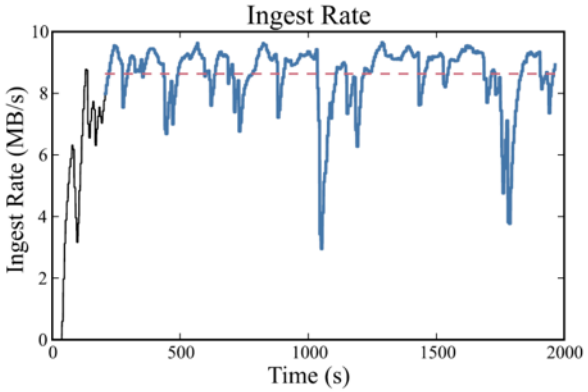
- Information:
  - Process Status
  - Tablet Assignments
  - Ingest & Query Activity
  - Background Activity
  - Cluster Space

Ingest Performance
-putting data in-

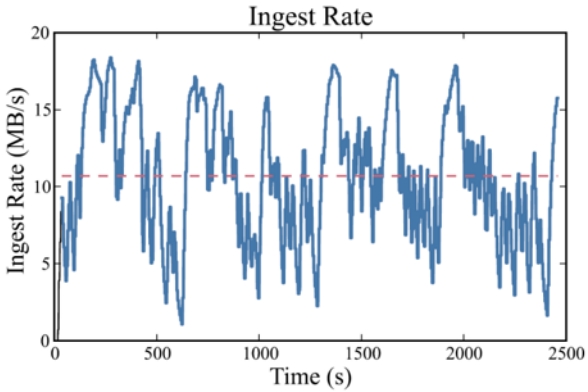
- **Proven to scale past 100M inserts per second**
- **Parallel clients required to achieve high insert rates**
  - **Ideal number of clients depends on number of tablet servers and your particular data and hardware**
- **Too many ingestor processes can saturate Accumulo and degrade performance and stability**



Healthy ingest

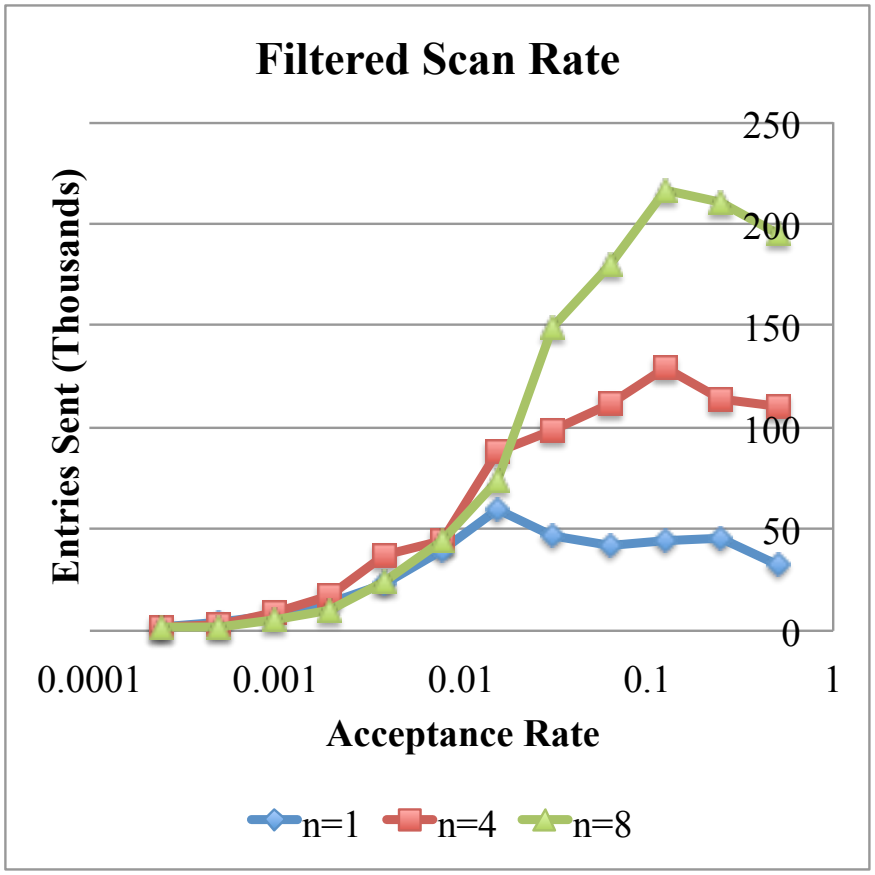


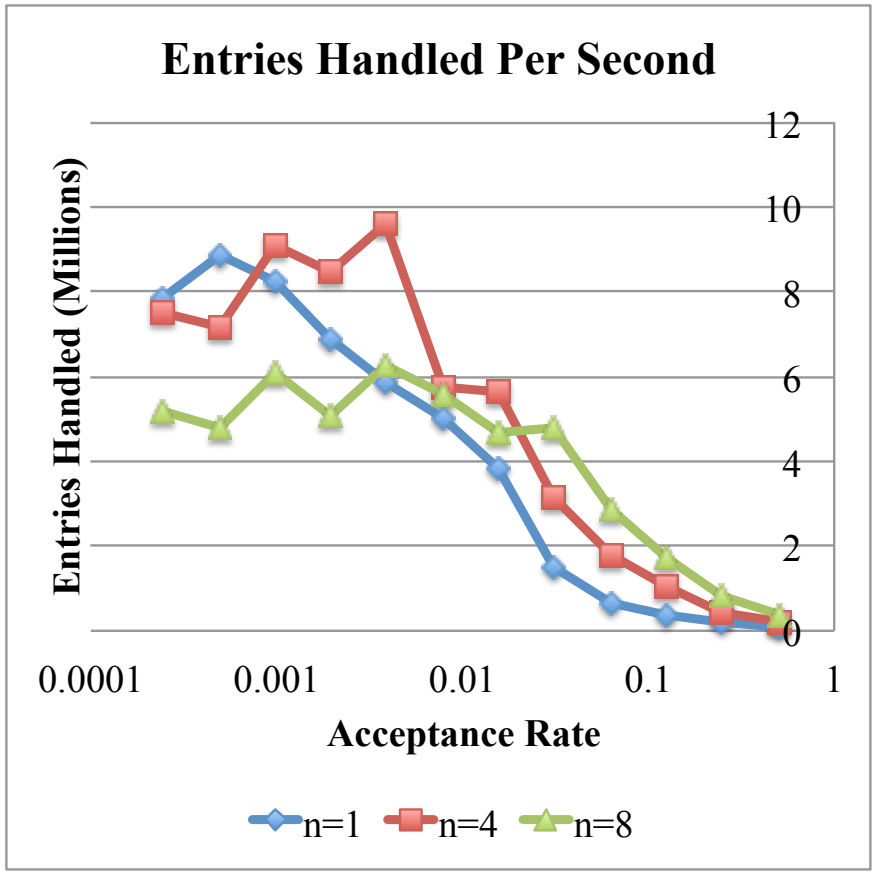
Some backpressure



Saturation

Scan Performance
-pulling data out-





High Acceptance Rates: Bound by client ability to accept data
Low Acceptance Rates: Bound by server filesystem

Accumulo Tuning

- Factors that affect performance and stability:
  - Number of ingestors
  - How data is distributed (random vs. serial, natural load balancing)
  - Pre-splitting of tables
  - Number of tables
  - Hardware resources (number of nodes/RAM/Disk Space)
- More advanced factors:
  - Number of replication factors
  - Bulk Ingest
  - Write ahead logging
  - Number of Zookeeper instances
  - Forced synchronization between Zookeeper Znodes
  - Modifying number of simultaneous compactions

Schema

- A well designed schema can help ensure that data is distributed well and that future analytics can access the data they need efficiently
- A structure described in a language supported by the database management system
- Needs to make use of Accumulo row-store properties
- Schema should NOT be the same as a SQL schema
  - We've seen people try this...doesn't work out too well

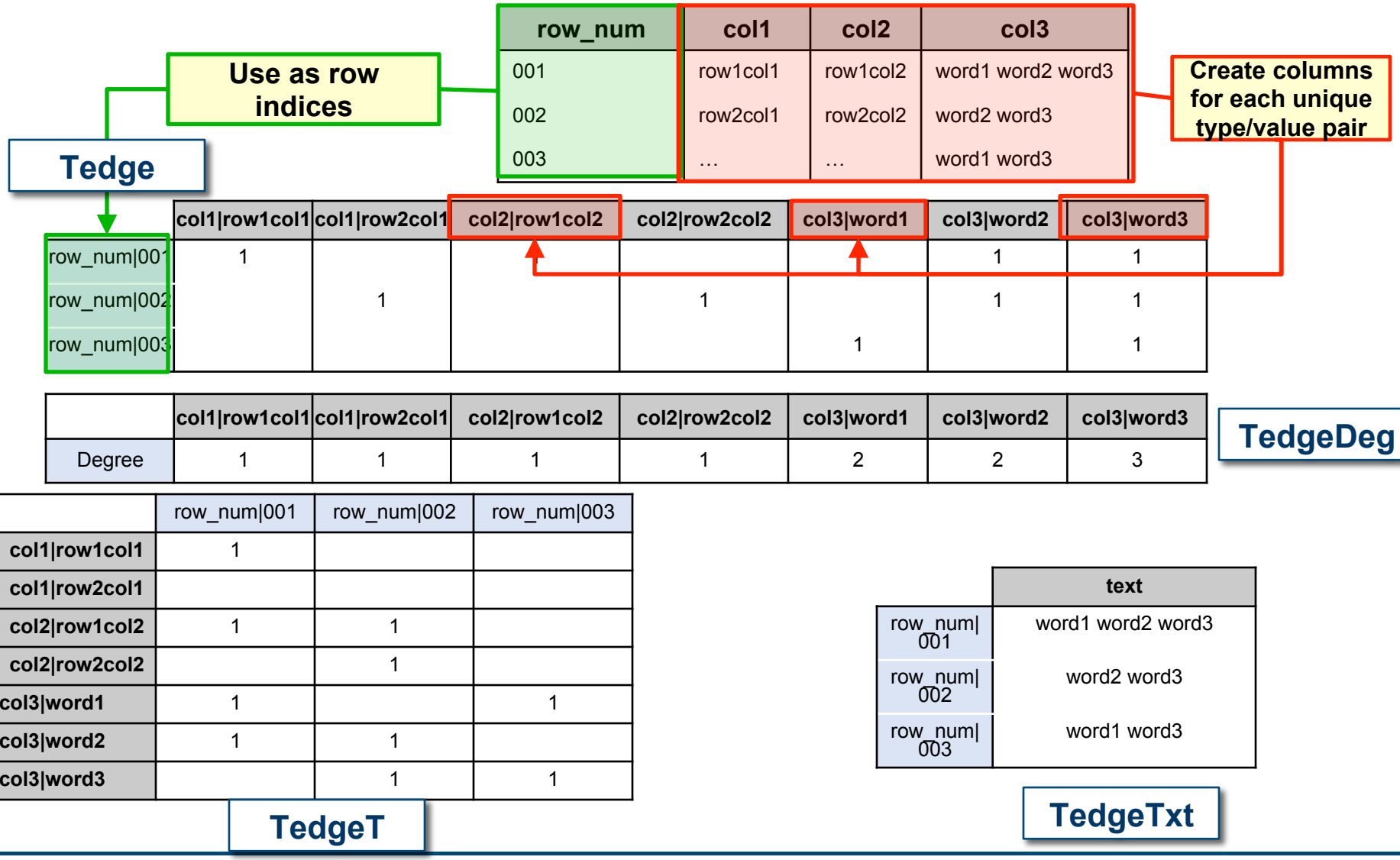
Data Schema

- Accumulo supports triples
  - How can we represent heterogeneous data types in a common data schema?
  - Use the D4M schema
- D4M schema converts structured or unstructured raw data to the 3-tuple representation supported by Accumulo:

  (row, column) ➔ value

  - row is a unique identifier (often some variation of a time stamp)
  - column is a unique representation of the data
  - value is typically just '1'
- Usually use a 4 table representation
  - The Edge Table, the Transpose Table, Degree Table, Raw Table

Exploded Table



row_num	col1	col2	col3
001	row1col1	row1col2	word1 word2 word3
002	row2col1	row2col2	word2 word3
003	…	…	word1 word3

	col1|row1col1	col1|row2col1	col2|row1col2	col2|row2col2	col3|word1	col3|word2	col3|word3
row_num|001	1					1	1
row_num|002		1		1		1	1
row_num|003					1		1

	col1|row1col1	col1|row2col1	col2|row1col2	col2|row2col2	col3|word1	col3|word2	col3|word3
Degree	1	1	1	1	2	2	3

	row_num|001	row_num|002	row_num|003
col1|row1col1	1		
col1|row2col1			
col2|row1col2	1	1	
col2|row2col2		1	
col3|word1	1		1
col3|word2	1	1	
col3|word3		1	1

	text
row_num|001	word1 word2 word3
row_num|002	word2 word3
row_num|003	word1 word3

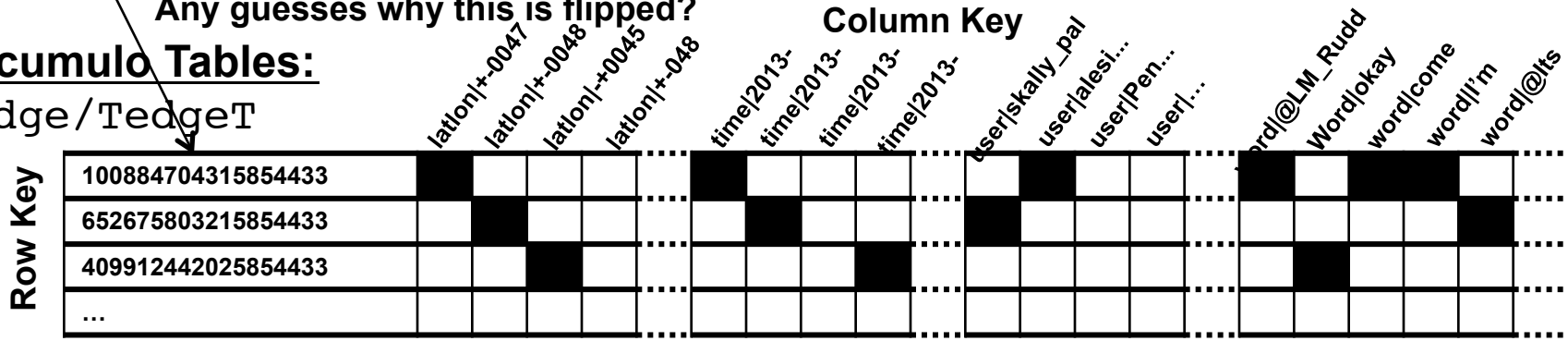
Schema Example

TweetID	Time	Lat	Long	User	Source	Text
334458513407488001	2013-05-15 00:01:38	41.50442327	-71.3080676	alesiatasso	Twitter for iPhone	@LM_Rudd okay, I'm...
334458512308576256	2013-05-15 00:01:37	42.56289604	-84.82935883	skally_pal	Twitter for iPhone	@ItsMackk I'll see you…
334458513755602944	2013-05-15 00:01:38	41.39560229	-81.73950863	AmbahDee	Twitter for iPhone	@NataliaProkop come…
334458520244219904	2013-05-15 00:01:39	42.2041867	-87.8126571	yobebeau	Endomondo	Was out cycling 31.64 …
…	…	…		…		…

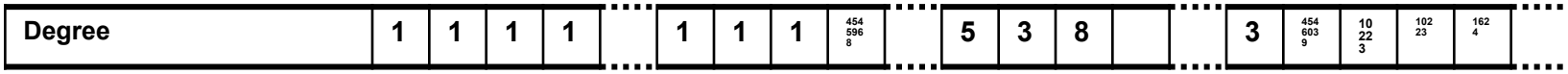
Any guesses why this is flipped?

Accumulo Tables:

Tedge/TedgeT



TedgeDegt



TedgeTxt

Row Key	text
100884704315854433	@LM_Rudd okay, I'm gonna get in comfy clothes then drive to your house then we can adventure! lol
652675803215854433	@ItsMackk I'll see you tomorrow while we're moving
409912442025854433	Was out cycling 31.64 miles with #Endomondo. See it here: http://t.co/9CpxSipkIs
…	…

Question

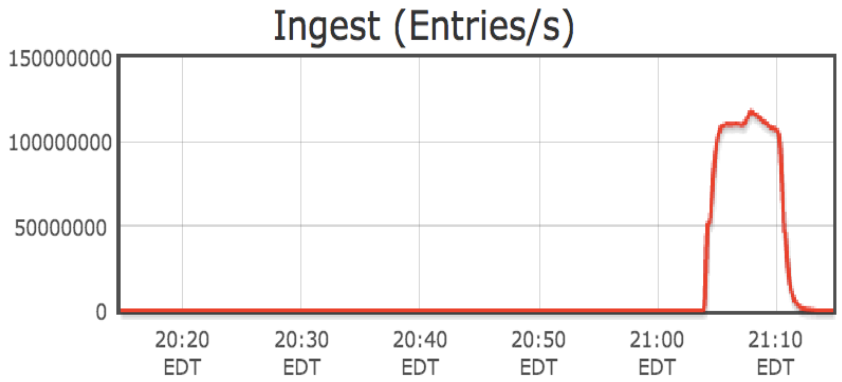
- **Design a schema for the following data sample raw data:**

log_id	src_ip	srv_ip
001	128.0.0.1	208.29.69.138
002	192.168.1.2	157.166.255.18
003	128.0.0.1	74.125.224.72 208.29.69.138

	src_ip|128.0.0.1	src_ip|192.168.1.2	srv_ip|157.166.255.18	srv_ip|208.29.69.138	srv_ip|74.125.224.72
log_id|100	1			1	
log_id|200		1	1		
log_id|300	1			1	1

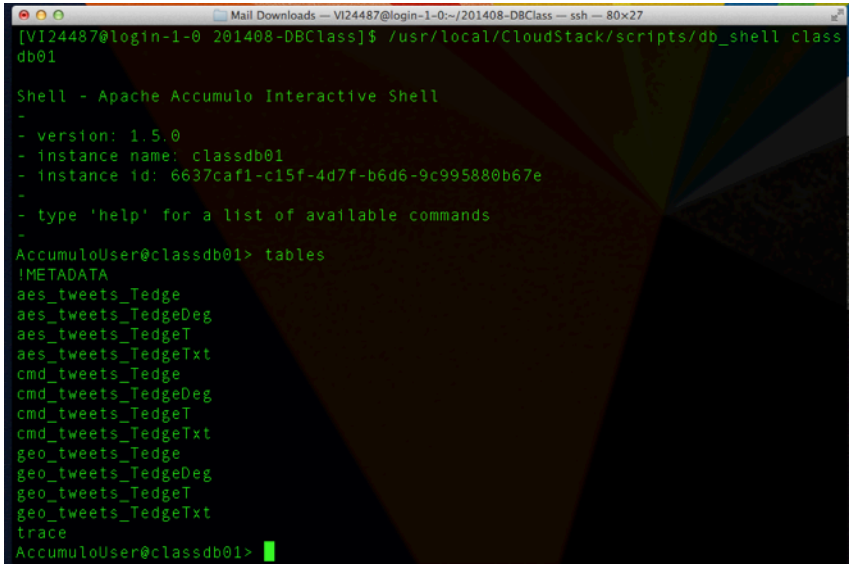
Lincoln Record Accumulo Performance

- Achieving 155,000,000 million inserts/second into Accumulo
- Data generated from Graph 500 data generator
  - Largest size inserted: 5.4B x 131K and 43B entries
- Optimizations to achieve record:
  - Scale ingestors with problem size
  - Pre-split tables
  - HDFS replication turned off
  - Reduce number of Zookeeper followers and move to observer status
  - Turn off write-ahead logging
  - Increase number of simultaneous minor compactions



Connecting to Accumulo

- **Options:**
  - **Via Accumulo shell**
  - **Using JDBC Driver (D4M)**
  - **Using Thrift Proxy (Python Examples tomorrow)**
- **Allow creation of tables, inserting data, table configuration and server administration**



We will spend a lot of time on this during the second part of the class

Accumulo Summary
-what you should remember-

- Accumulo is a distributed key value store database.
- Key features include:
  - High performance/scalability
  - Runs on COTS hardware
  - Schema flexibility
  - Cell level security
  - Iterators
- Used extensively by the US DoD and Intelligence Community
- Many ways to connect to Accumulo

Part 1 Summary

- Big Data produces many challenges which are addressed by a computing cloud
- Many different options in this cloud and very specific to intended application
- Databases form an important part of most cloud infrastructures
- Database choices are vast
- Apache Accumulo is a popular NoSQL database
- Tomorrow's class:
  - Bring a laptop with SSH capabilities
  - Install Google Earth if you would like to view demo results

References

Backup