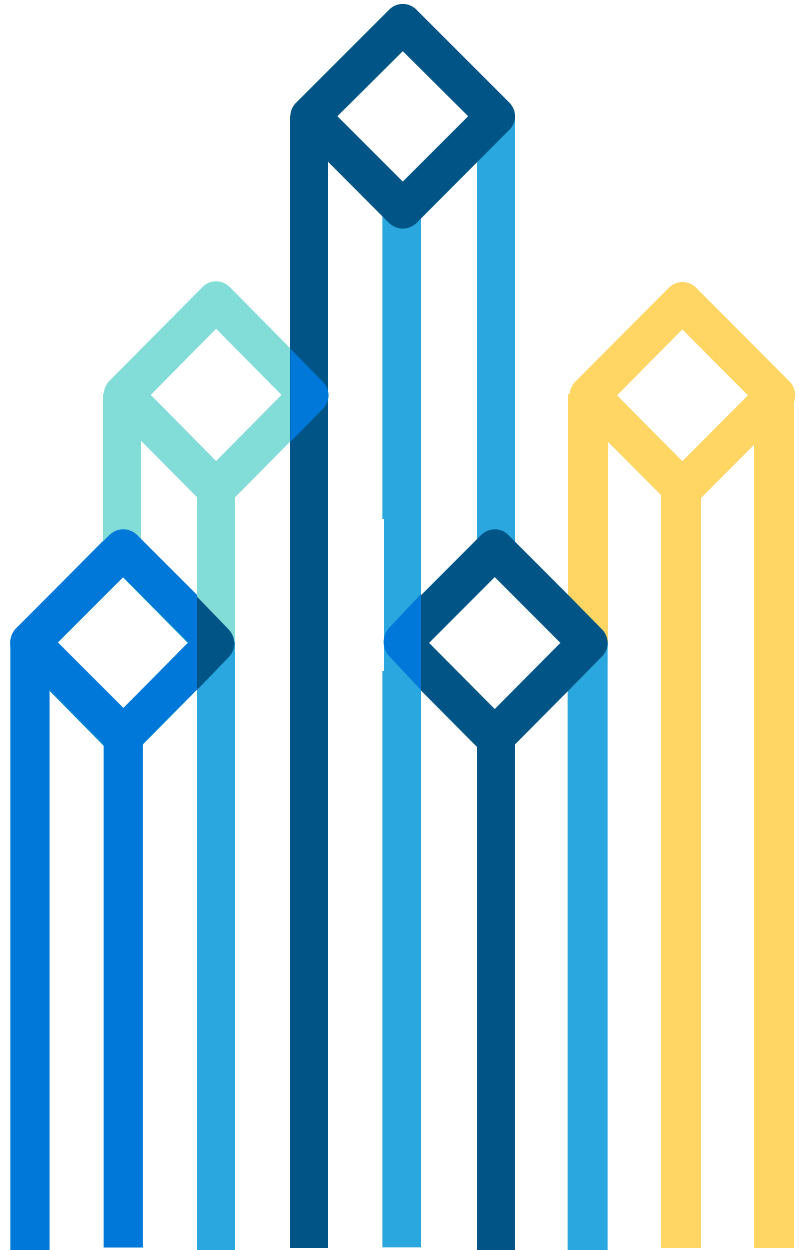
cloudera®



Apache HBase: Overview and Use Cases

Apekshit Sharma

About myself

Apekshit Sharma

- Software Engineer, Cloudera
  - Ex-Software Engineer, Google
- Apache HBase contributor
- Performance improvements, replication, build infra, etc

Contents

- Motivation
- Apache HBase data model
- Overview of Architecture
- Few usage patterns

cloudera

Motivation

Bigtable: A Distributed Storage System for Structured Data

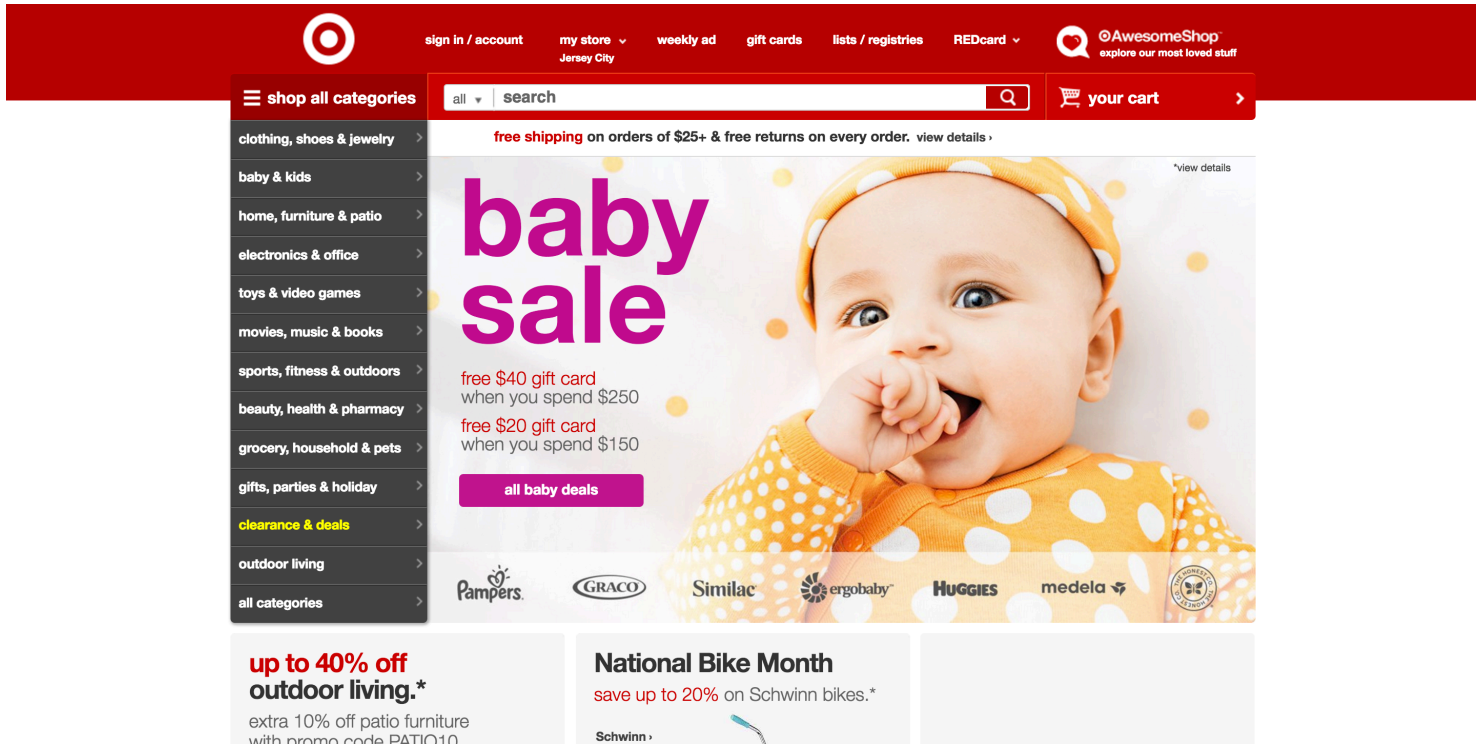
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

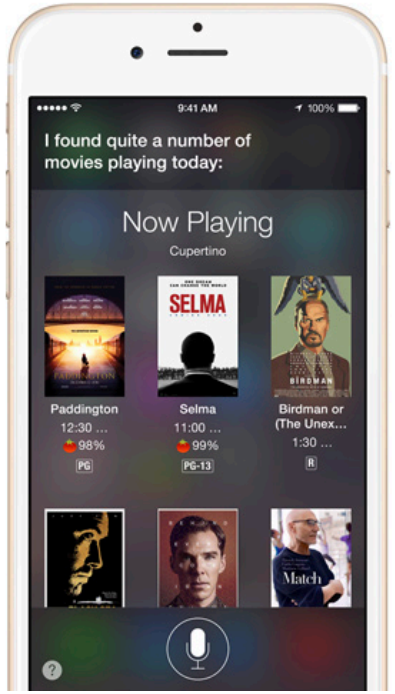
Google, Inc.

Motivation

- What if you're not trying to index the internet?

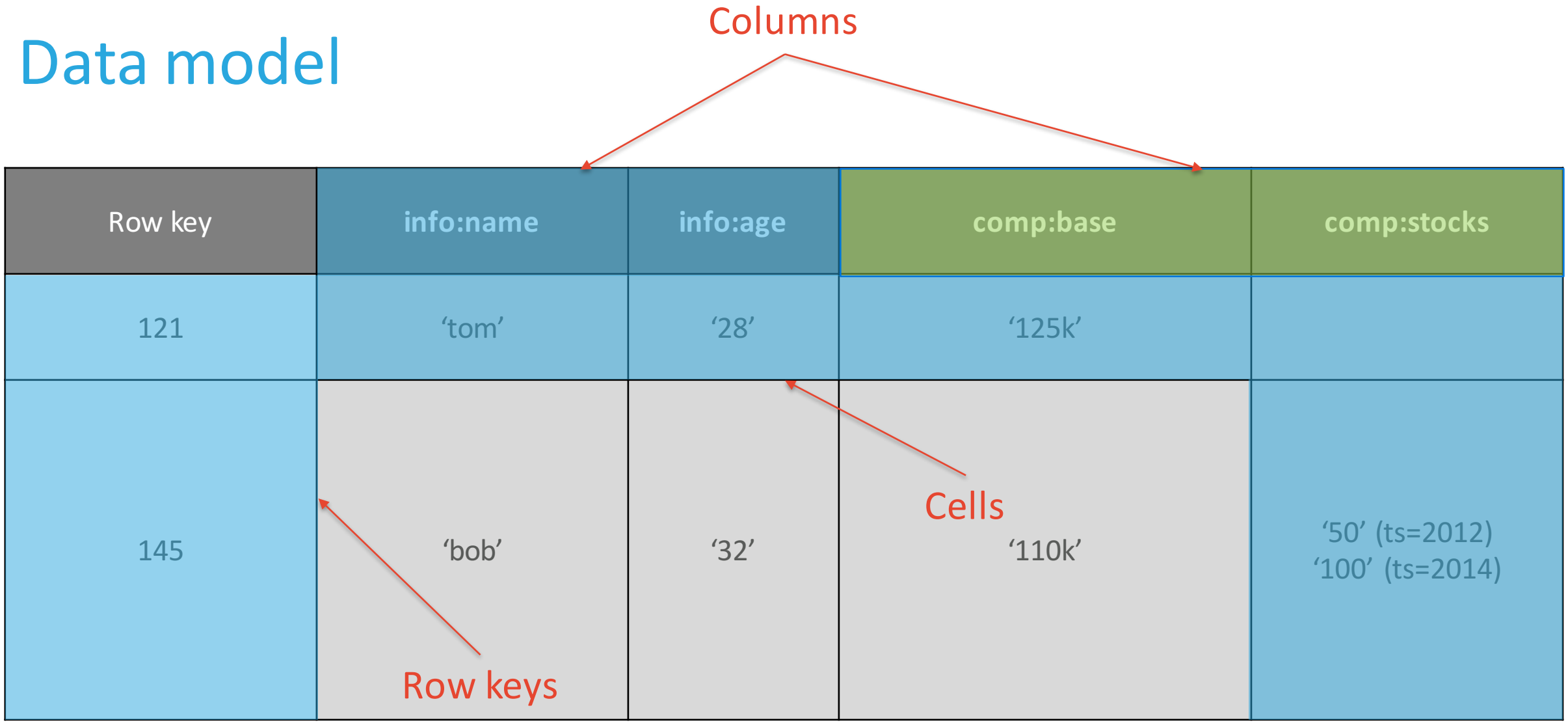


"What movies are playing today?"



- Open Source
- Horizontally Scalable
- Consistent
- Random access, low latency
- Built on top of HDFS

Data model



cloudera

Data model

- Sorted rows : supports billions of rows
- Columns : Supports millions of columns
- Cell : intersection of row and column.
  - Can have multiple values (which are time-stamped)
  - Can be empty. No storage/processing overheads

HBase Architecture

Unique id	Name	price	weight	store1	store2	store3
“1000000”	snickers	$9.99	4 Oz	Yes	Yes	Yes
“3000000”	almonds	$9.99	8 Oz	Yes	No	Yes
“8000000”	coke	$9.99	16 Oz	Yes	Yes	Yes

cloudera

HBase Architecture

Unique id	Name	price	weight	store1	store2	store3
“1000000”	snickers	$9.99	4 Oz	Yes	Yes	Yes
“3000000”	almonds	$9.99	8 Oz	Yes	No	Yes
“8000000”	coke	$9.99	16 Oz	Yes	Yes	Yes
“4000000”	new	$34.63	16 Oz	No	Yes	Yes

HBase Architecture

Unique id	Name	price	weight	store1	store2	store3
“1000000”	snickers	$9.99	4 Oz	Yes	Yes	Yes
“3000000”	almonds	$9.99	8 Oz	Yes	No	Yes
“8000000”	coke	$9.99	16 Oz	Yes	Yes	Yes
“4000000”	foo	$34.63	16 Oz	No	Yes	Yes
“5000000”	bar	$22.54	16 Oz	Yes	Yes	Yes

HBase Architecture

Unique id	Name	price	weight	store1	store2	store3
“1000000”	snickers	$9.99	4 Oz	Yes	Yes	Yes
“3000000”	almonds	$9.99	8 Oz	Yes	No	Yes
“8000000”	coke	$9.99	16 Oz	Yes	Yes	Yes
“4000000”	foo	$34.63	16 Oz	No	Yes	Yes
“5000000”	bar	$22.54	16 Oz	Yes	Yes	Yes
“9000000”	new1	$2.5	16 Oz	Yes	Yes	Yes
“7000000”	new2	$6.4	16 Oz	Yes	Yes	Yes
“2000000”	new3	$6.4	16 Oz	Yes	Yes	Yes

HBase Architecture | Regions

[“”, “5000000”)

Row Key	Name	brand	price	weight	store1	store2	store3
“1000000”	snickers	xxx	$9.99	4 Oz	Yes	Yes	Yes
“2000000”	new3	xxx	$6.4	16 Oz	Yes	Yes	Yes
“3000000”	almonds	xxx	$9.99	8 Oz	Yes	No	Yes
“4000000”	foo	xxx	$34.63	16 Oz	No	Yes	Yes

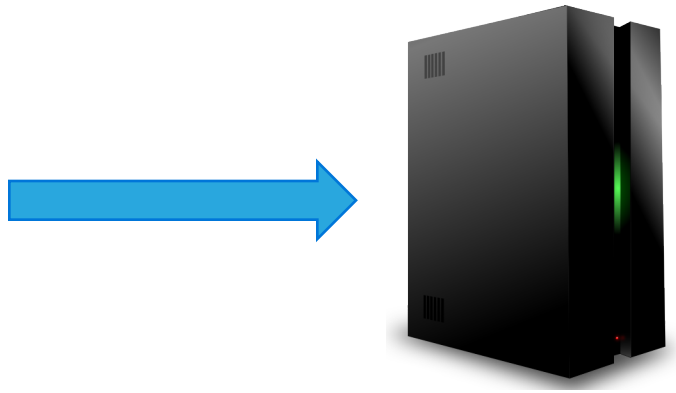
[“5000000”, “”)

Row Key	Name	brand	price	weight	store1	store2	store3
“5000000”	bar	xxx	$22.54	16 Oz	Yes	Yes	Yes
“7000000”	new2	xxx	$6.4	16 Oz	Yes	Yes	Yes
“8000000”	coke	xxx	$9.99	16 Oz	Yes	Yes	Yes
“9000000”	new1	xxx	$2.5	16 Oz	Yes	Yes	Yes

cloudera

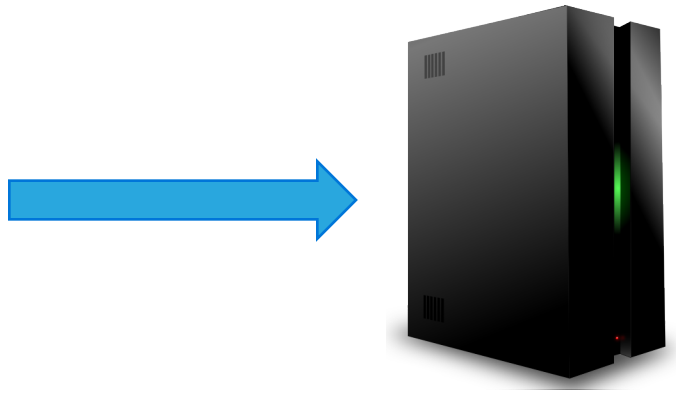
HBase Architecture | RegionServer

Row Key	Name	price	weight	….
“1000000”	snickers	$9.99	4 Oz	….
“2000000”	new3	$6.4	16 Oz	….
“3000000”	almonds	$9.99	8 Oz	….
“4000000”	foo	$34.63	16 Oz	….



Server 12

Row Key	Name	price	weight	….
“5000000”	bar	$22.54	16 Oz	….
“7000000”	new2	$6.4	16 Oz	….
“8000000”	coke	$9.99	16 Oz	….
“9000000”	new1	$2.5	16 Oz	….



Server 7

cloudera

HBase Architecture | Regions

- **Horizontal split** of tables
- Regions are served by RegionServers
- Automatically (based on configurations). Can be done manually too.

HBase Architecture

Row Key	Name	price	weight	store1	store2	store3
“1000000”	snickers	$9.99	4 Oz	Yes	Yes	Yes
“2000000”	new3	$6.4	16 Oz	Yes	Yes	Yes
“3000000”	almonds	$9.99	8 Oz	Yes	No	Yes
“4000000”	foo	$34.63	16 Oz	No	Yes	Yes
“5000000”	bar	$22.54	16 Oz	Yes	Yes	Yes
“7000000”	new2	$6.4	16 Oz	Yes	Yes	Yes
“8000000”	coke	$9.99	16 Oz	Yes	Yes	Yes
“9000000”	new1	$2.5	16 Oz	Yes	Yes	Yes

HBase Architecture | Column family

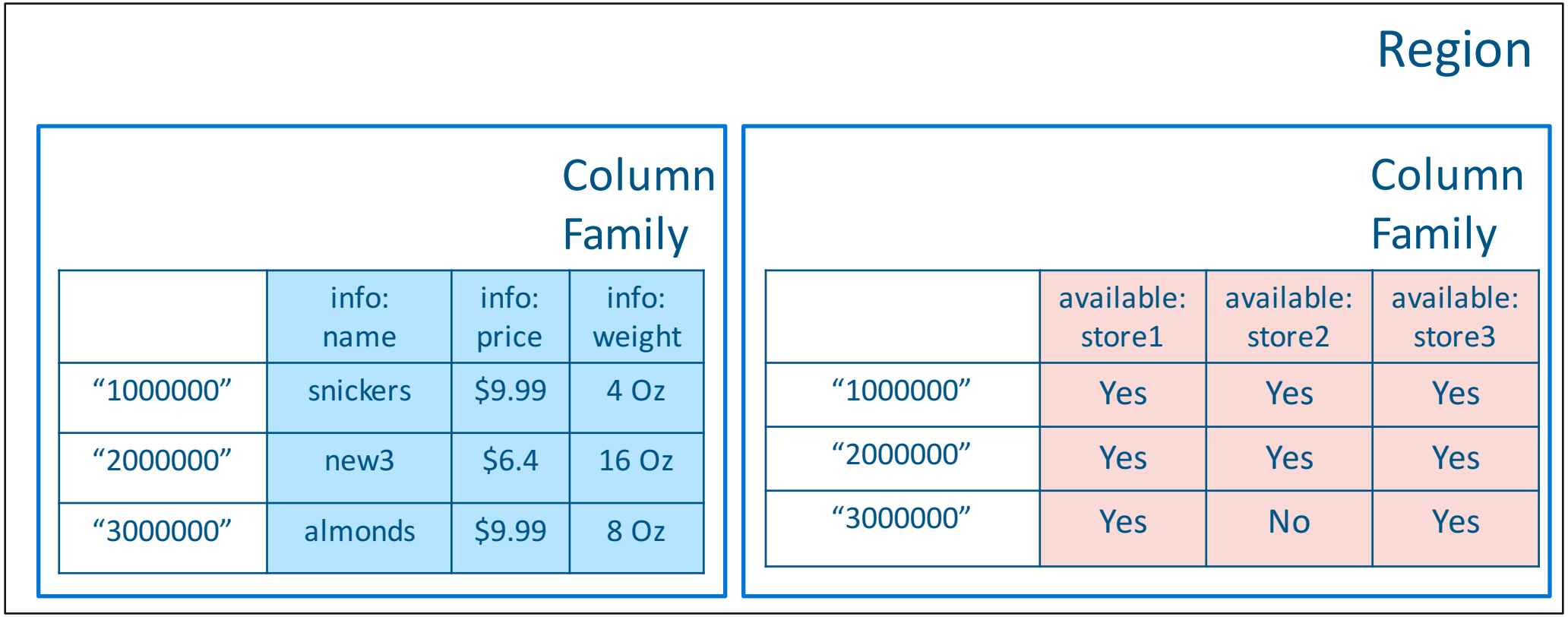
Row Key	info: name	info: price	info: weight	availability: store1	availability: store2	availability: store3
“1000000”	snickers	$9.99	4 Oz	Yes	Yes	Yes
“2000000”	new3	$6.4	16 Oz	Yes	Yes	Yes
“3000000”	almonds	$9.99	8 Oz	Yes	No	Yes
“4000000”	foo	$34.63	16 Oz	No	Yes	Yes
“5000000”	bar	$22.54	16 Oz	Yes	Yes	Yes
“7000000”	new2	$6.4	16 Oz	Yes	Yes	Yes
“8000000”	coke	$9.99	16 Oz	Yes	Yes	Yes
“9000000”	new1	$2.5	16 Oz	Yes	Yes	Yes

HBase Architecture | Column family

- You can also see it as **vertical splits**
- Data stored in separate files
- Tune performance
  - In-memory
  - Compression
  - Version retention policies
  - Cache priority
- Needs to be specified by the user

cloudera

HBase Architecture

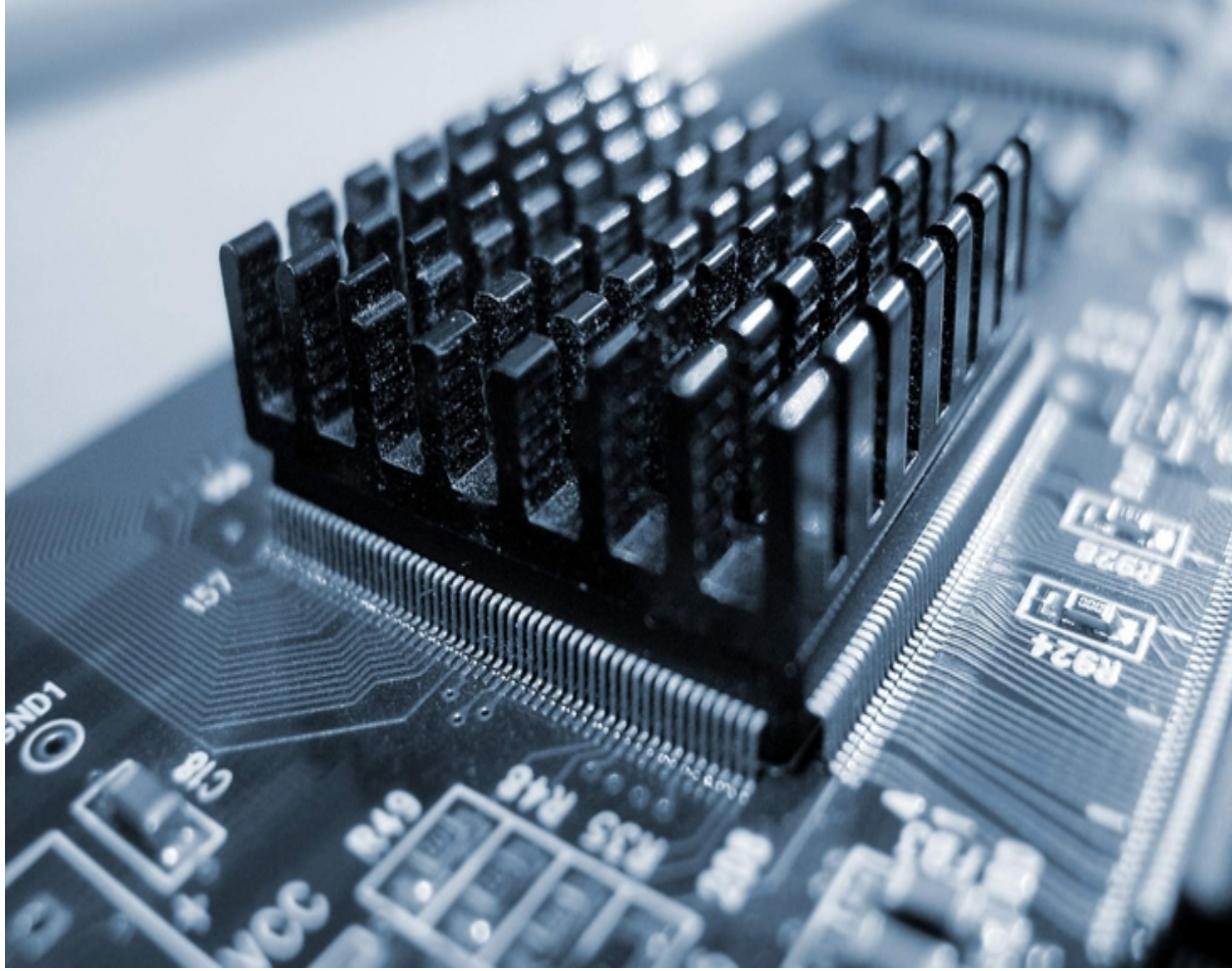


	info: name	info: price	info: weight
“1000000”	snickers	$9.99	4 Oz
“2000000”	new3	$6.4	16 Oz
“3000000”	almonds	$9.99	8 Oz

	available: store1	available: store2	available: store3
“1000000”	Yes	Yes	Yes
“2000000”	Yes	Yes	Yes
“3000000”	Yes	No	Yes

Enough of Architecture

Lets move to use patterns



cloudera

Apache HBase "Nascar" Slide

Apache HBase “Nascar” Slide

Apache HBase "Nascar" Slide

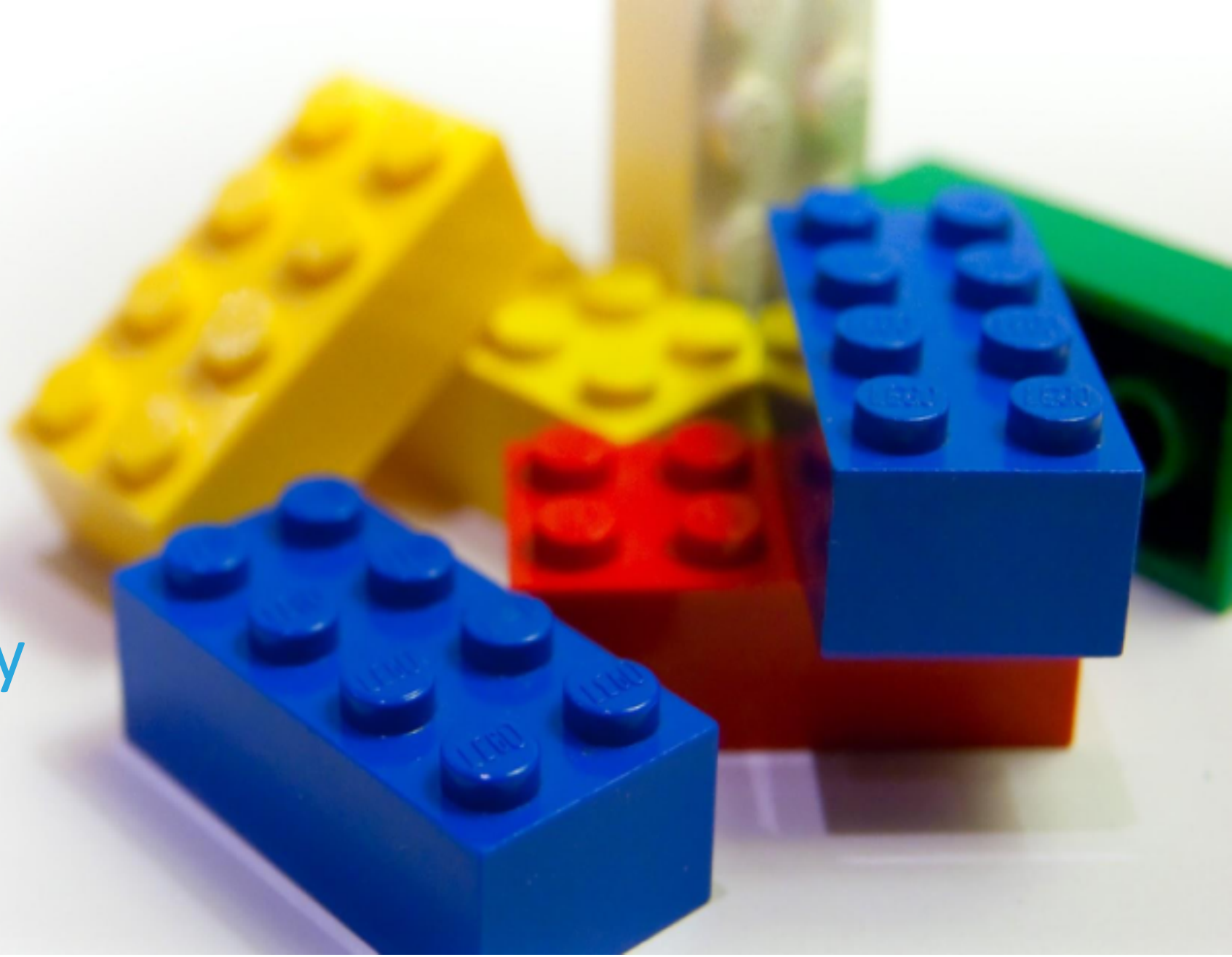
Apache HBase "Nascar" Slide

cloudera®

What have we learned from so many users?

There are some patterns which repeat often.

Just like a Lego block, maybe you can fit one directly in your system!



cloudera

Know your ...

Data

- Entity Data
- Time-centric Event Data

How it goes in and out

- Real-time vs Batch
- Random vs Sequential

cloudera

Know your data ...

There are primarily two kinds of big data workloads. They have different storage requirements.

- Entity centric data
- Time centric event data

cloudera

Entity centric data

Users

Accounts

Location

Clicks and Metrics

- Scales up with # of entities
- Billions of distinct entities

cloudera

Time centric event data

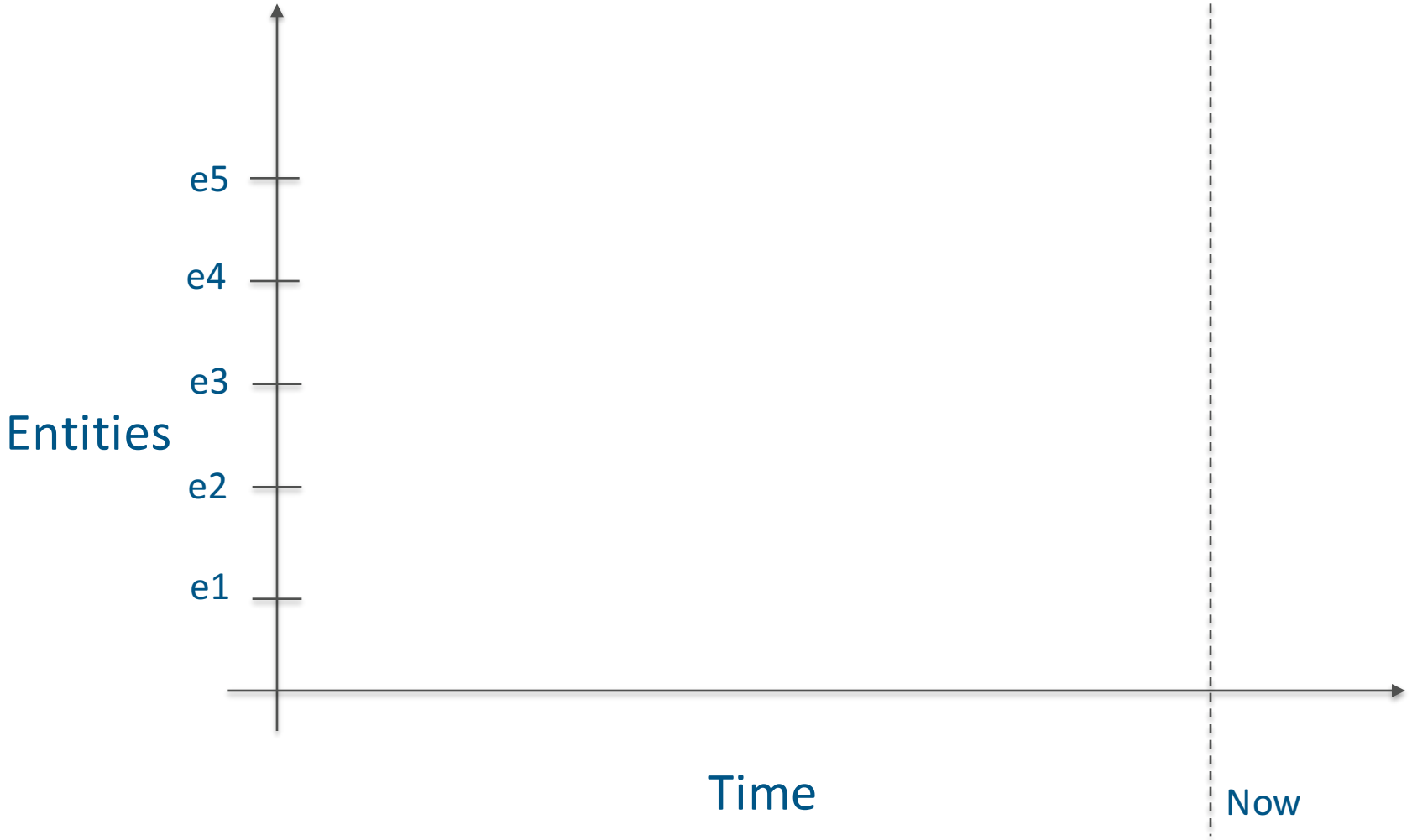
Stock Ticker Data

Monitoring applications

Periodic Sensor Data

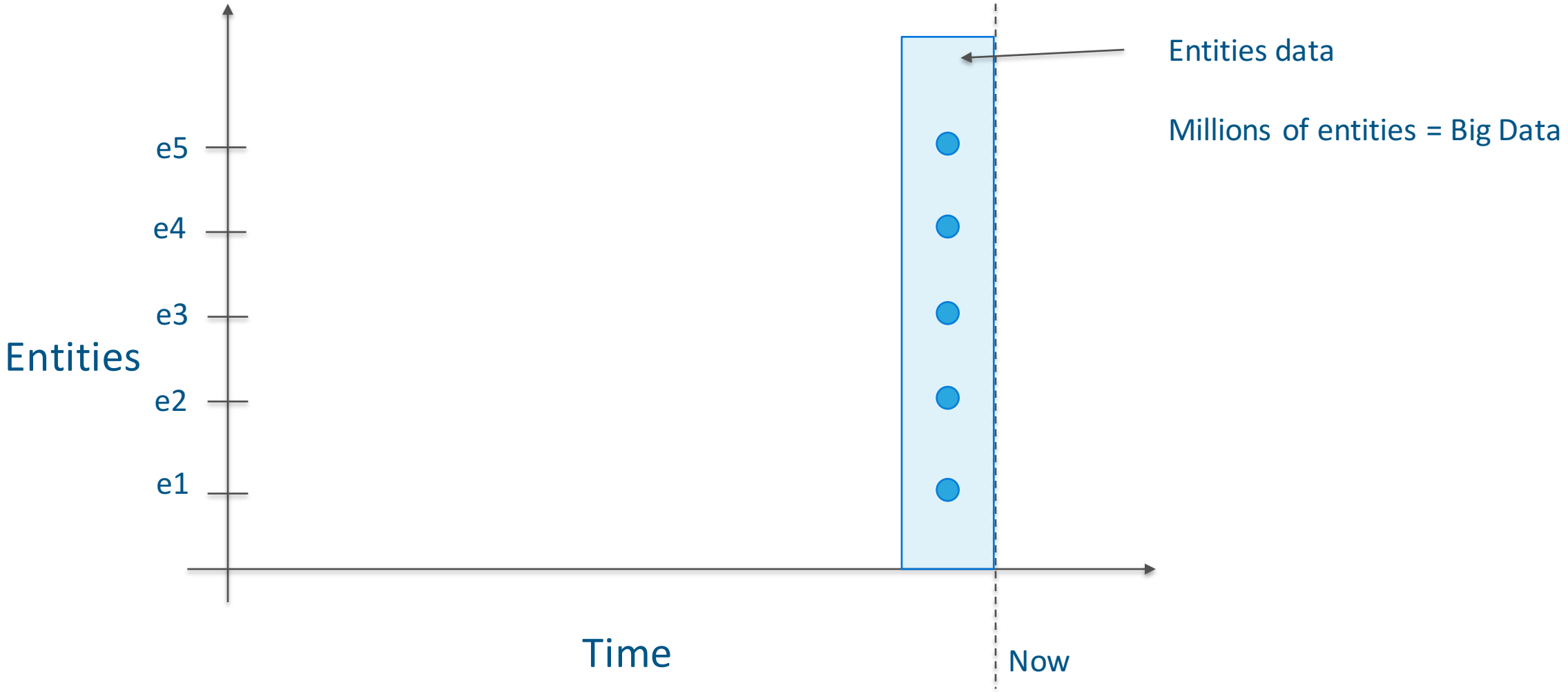
- Time-series data points over a period
- Scales up due to finer grained intervals, retention policies, and the passage of time

cloudera



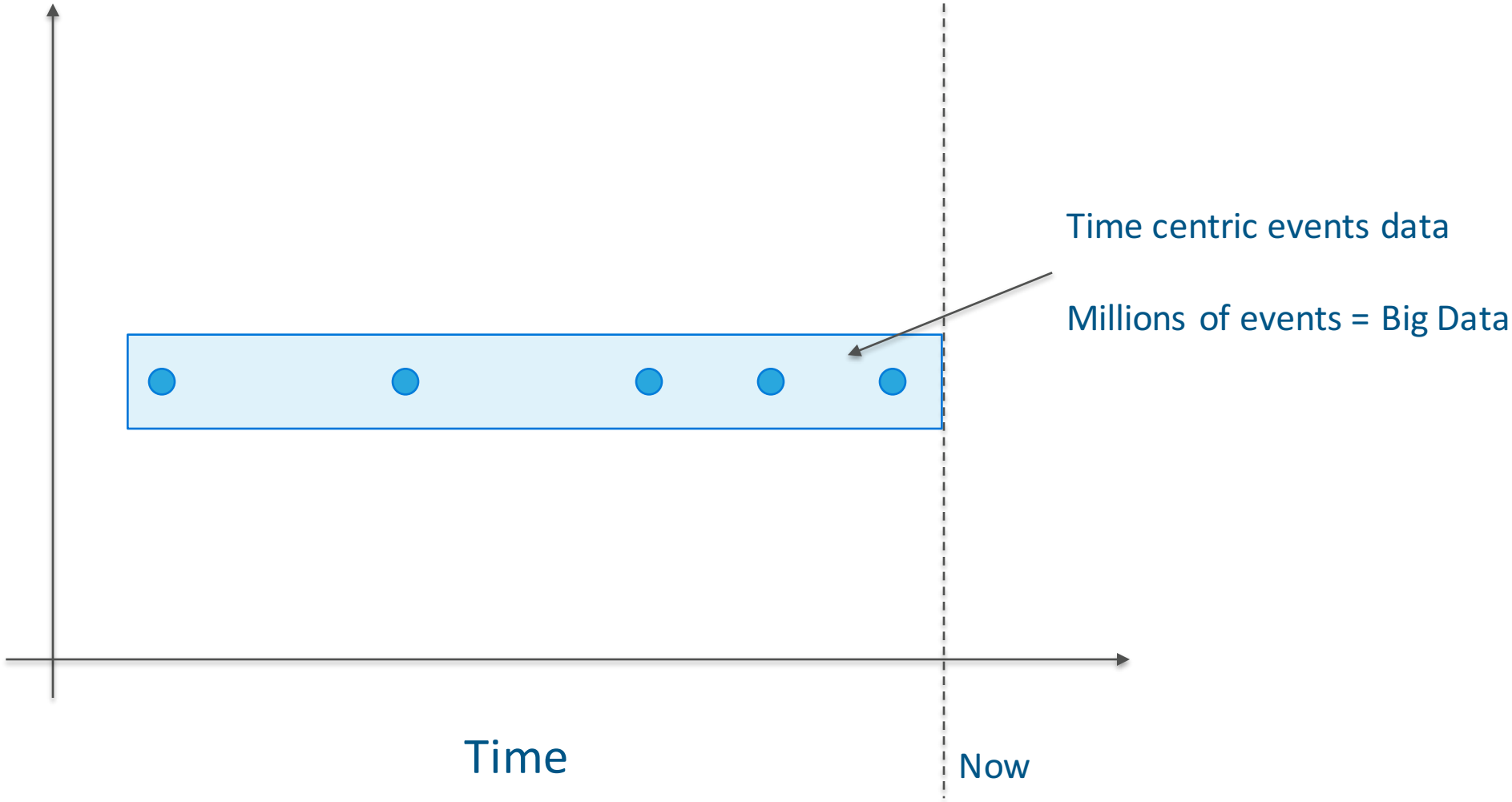
cloudera

Entities data



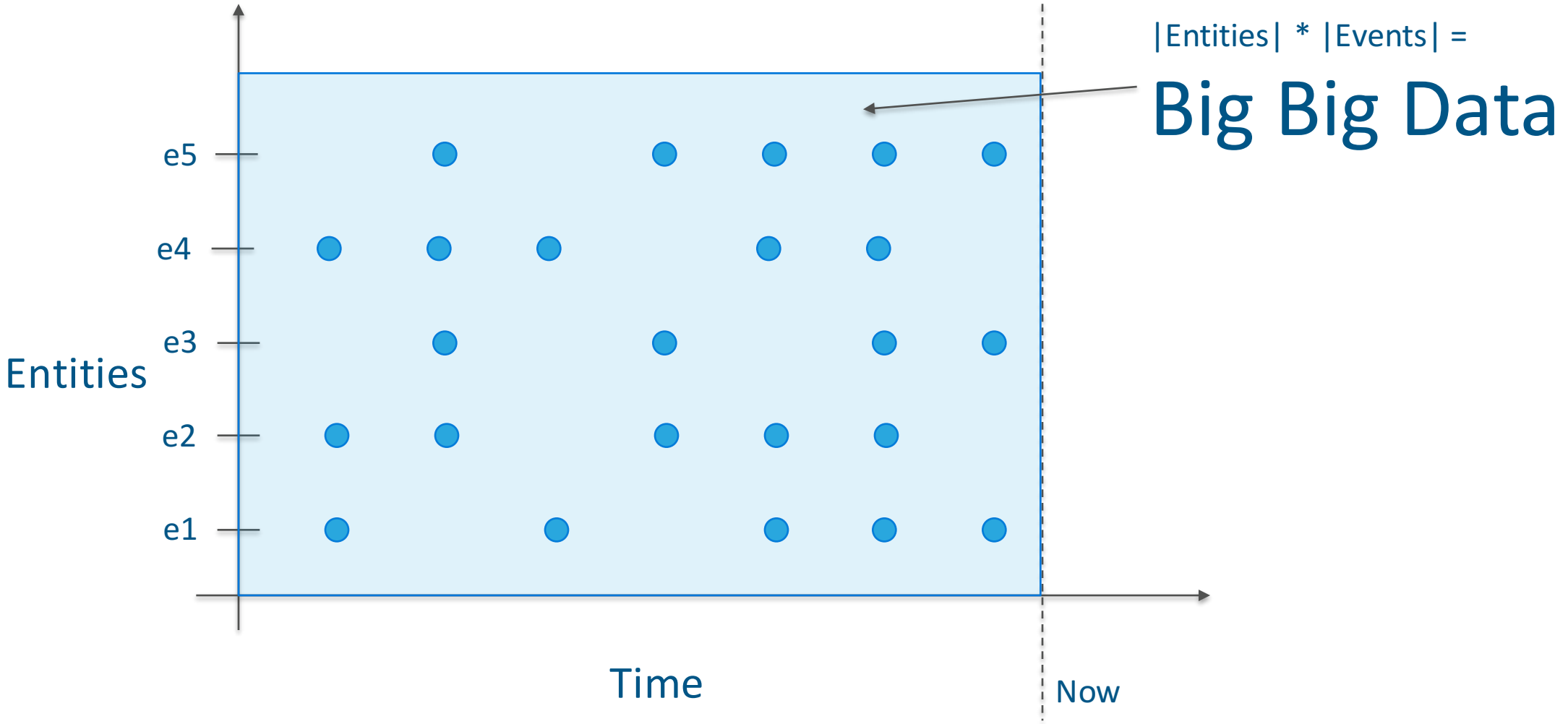
cloudera

Time-centric events data



cloudera

Time-centric events about Entities



cloudera

What questions do you ask?

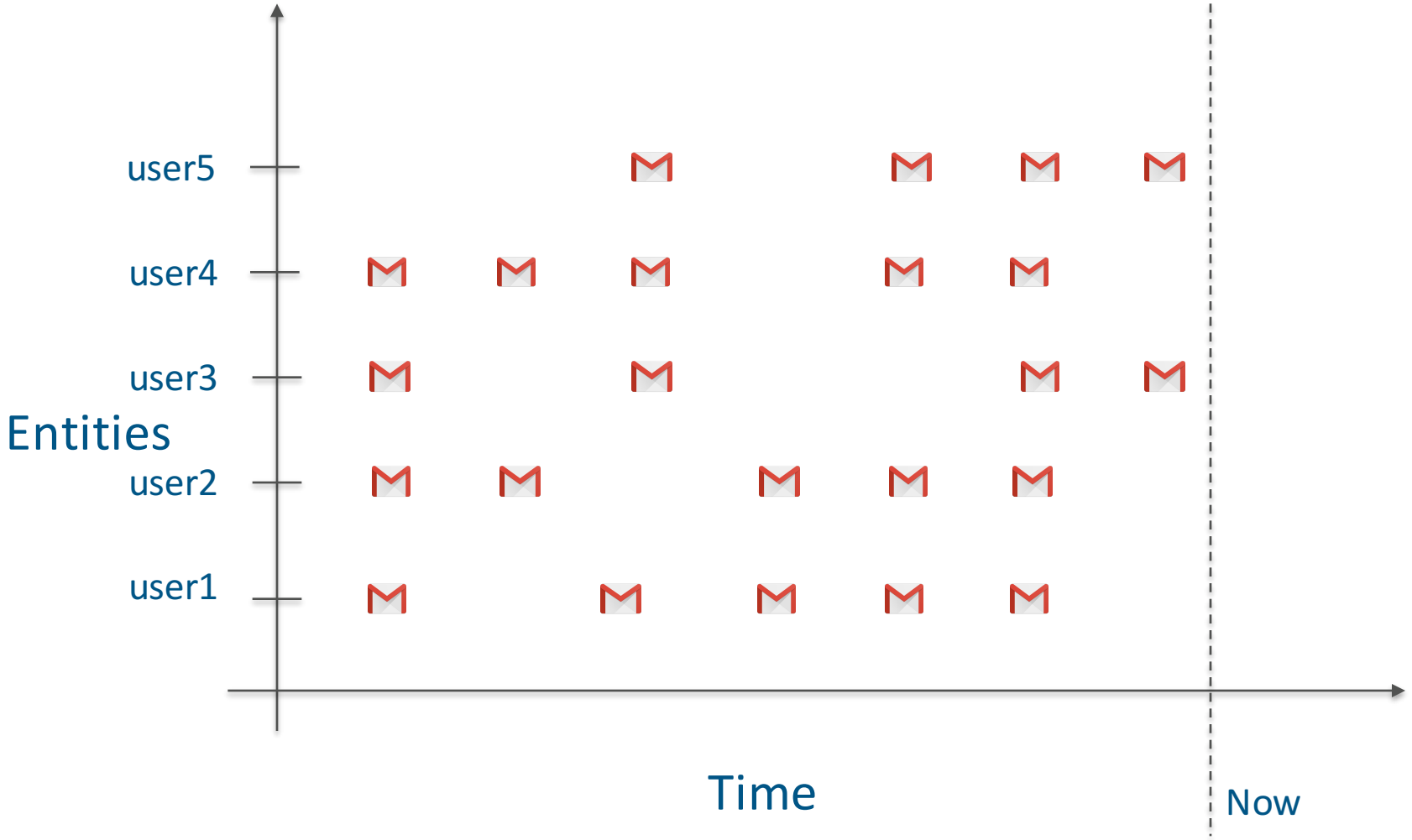
- Do you focus in on entity first?

OR

- Do you focus in on time ranges first?

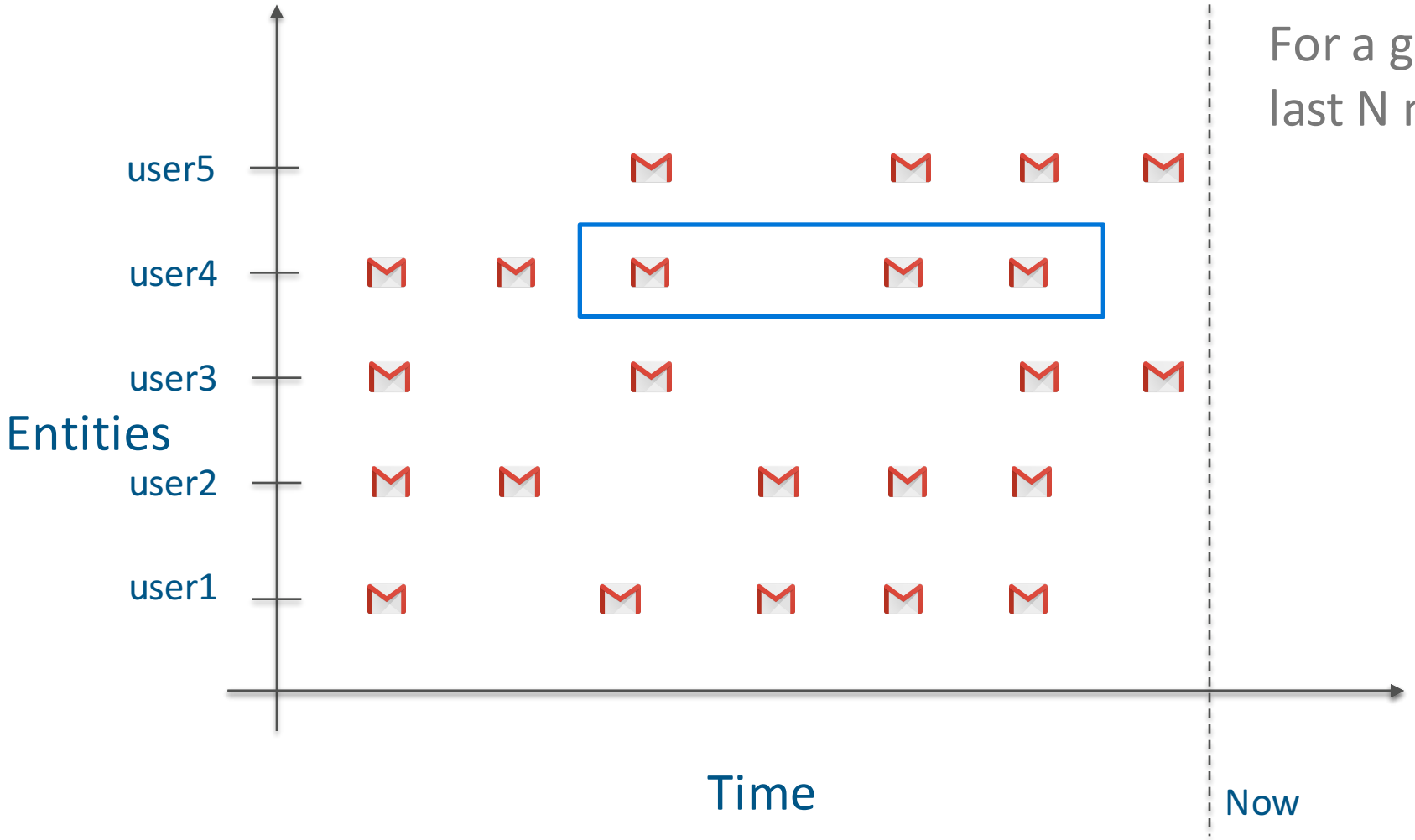
- Your answer will help you determine where and how to store your data.

Let’s say you start your own e-mail service….



cloudera

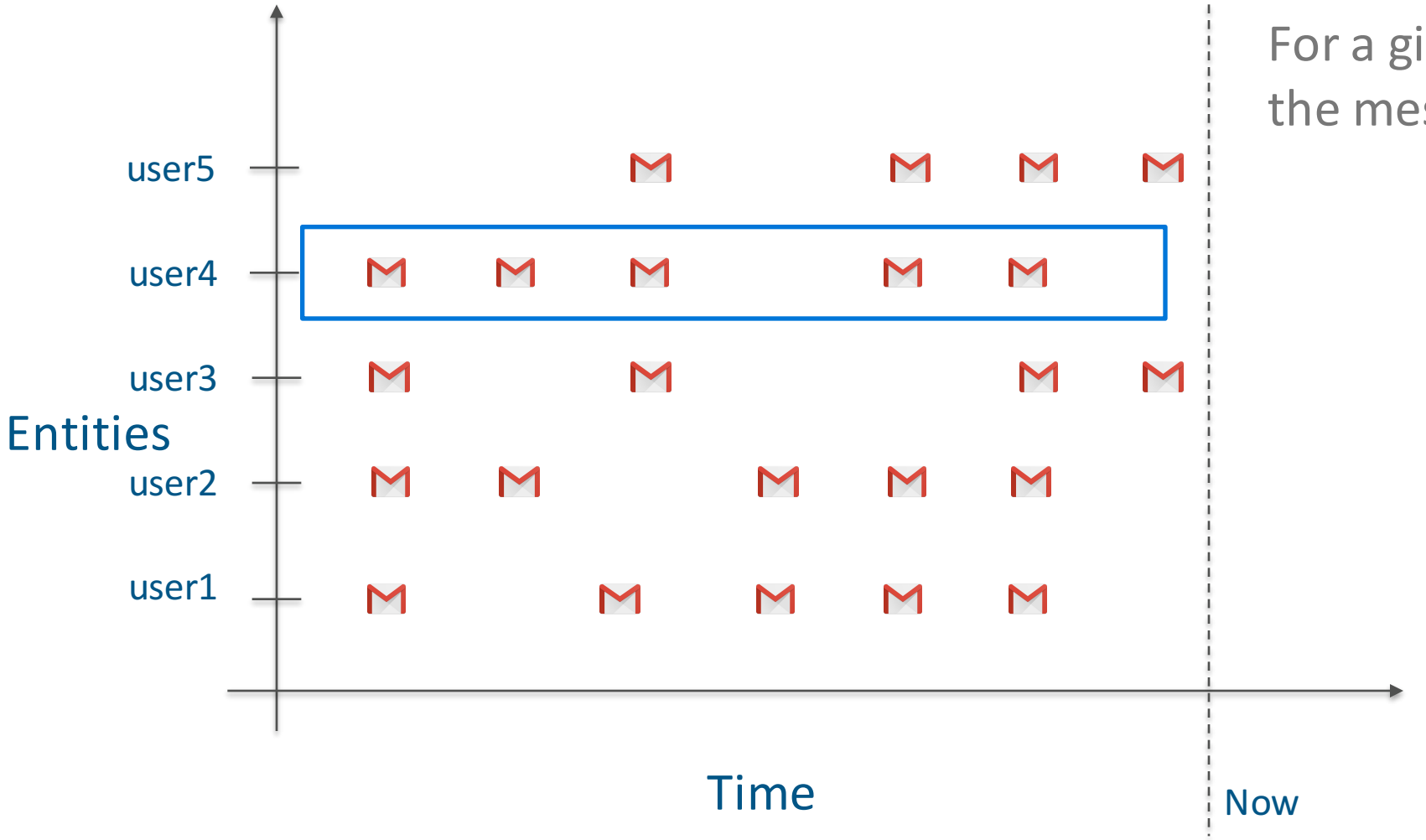
Entity first questions…



For a give user, show last N messages.

cloudera

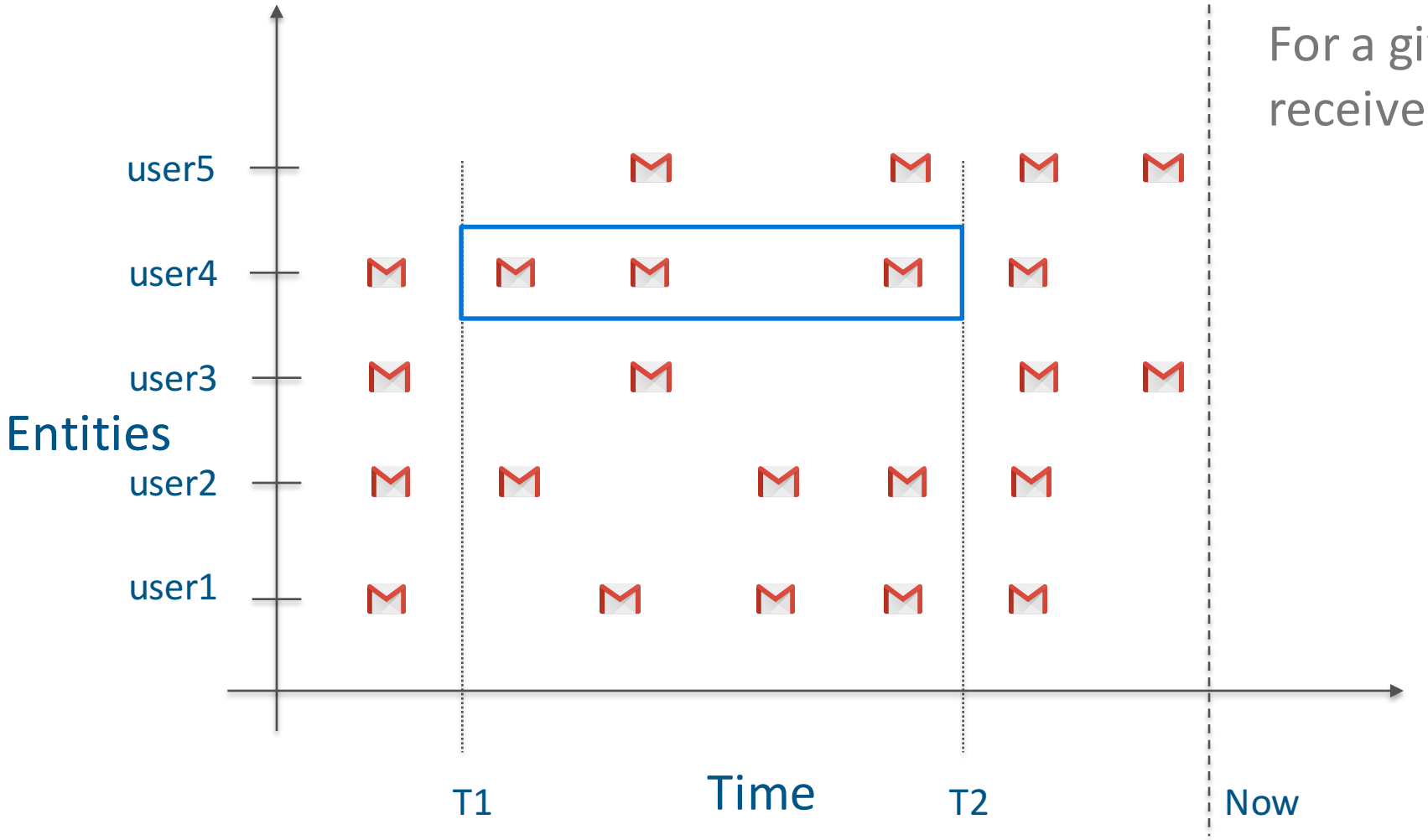
Entity first questions…



For a give user, show all the messages.

cloudera

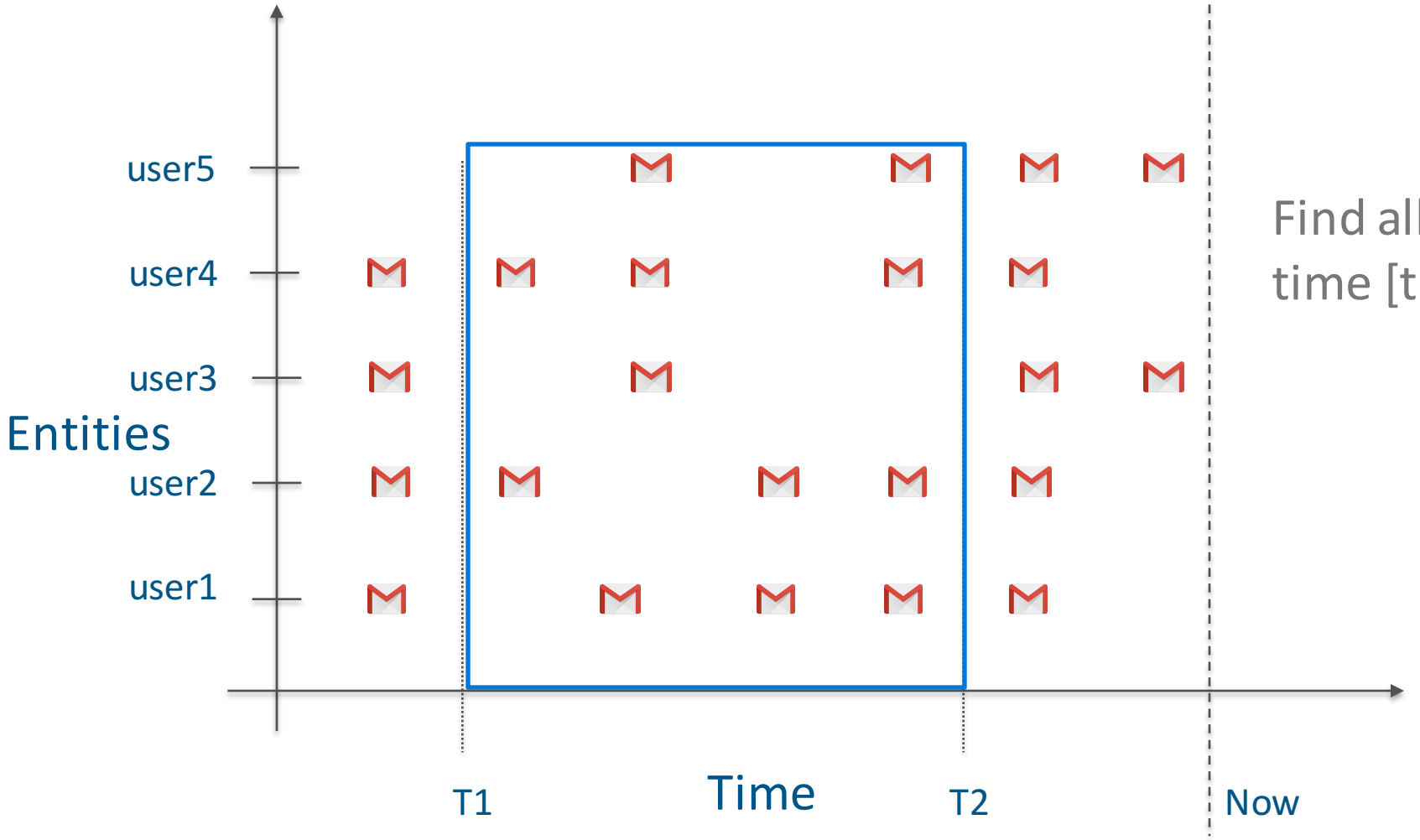
Entity first questions…



For a give user, show all messages received between time [t1, t2].

cloudera

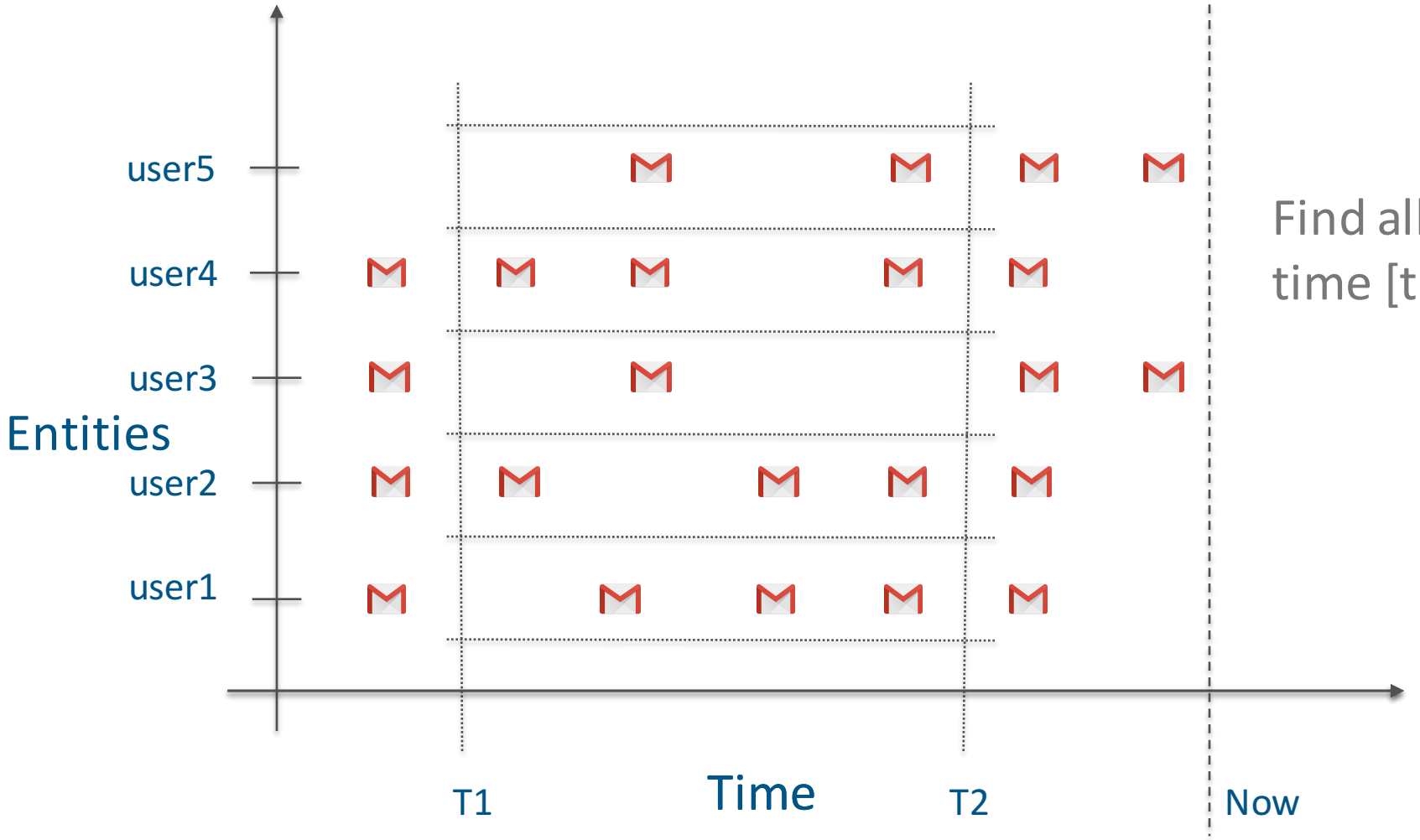
Time centric event first questions…



Find all messages between time [t1, t2].

cloudera

Time centric event first questions…



Find all messages between time [t1, t2] **for all users.**

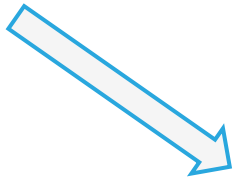
cloudera

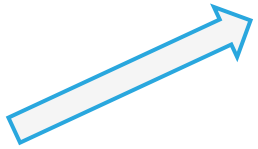
cloudera®

How does the data get in and out of HBase?

Getting data in...

Put, Incr, Append

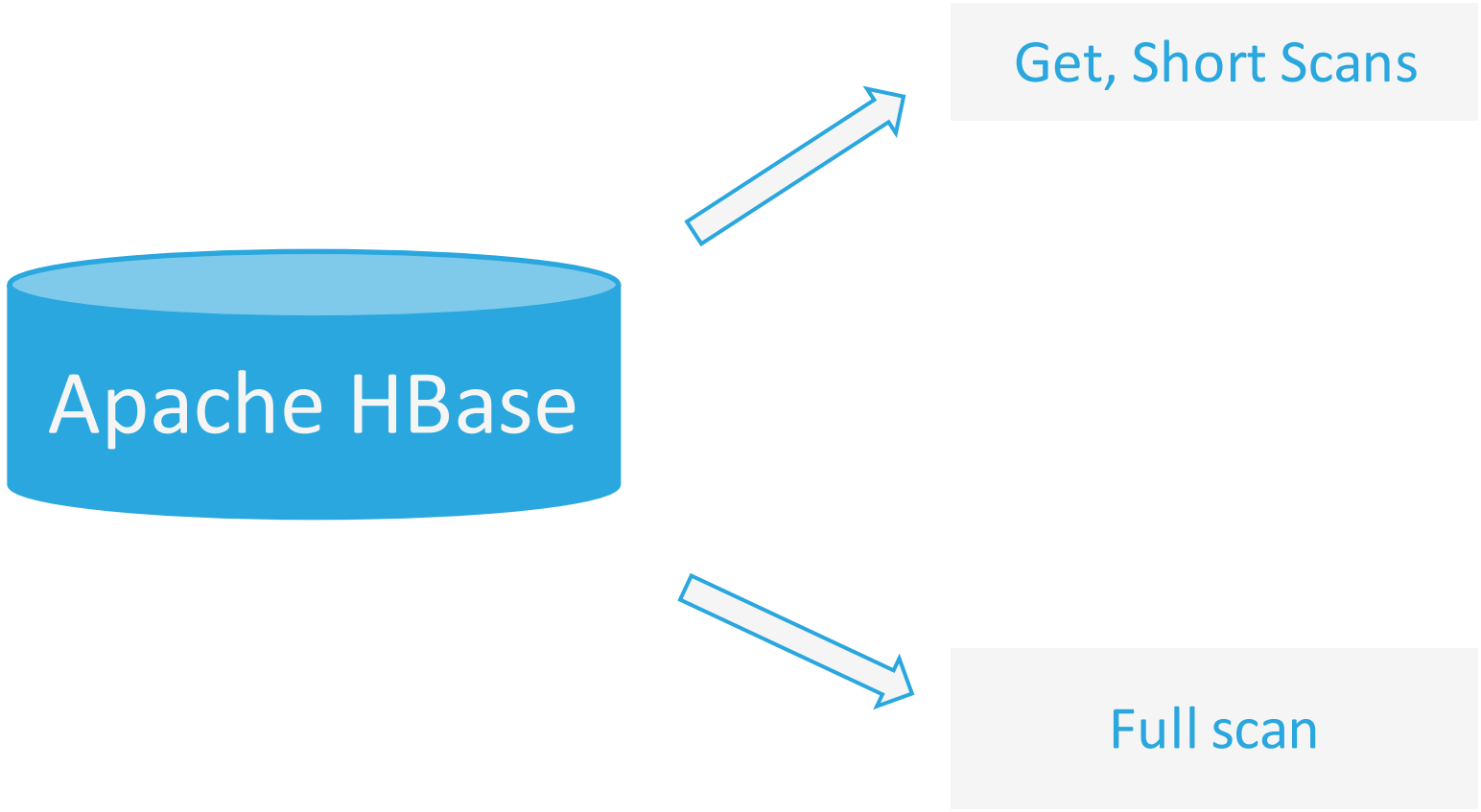




Bulk Import

cloudera

Getting data out...



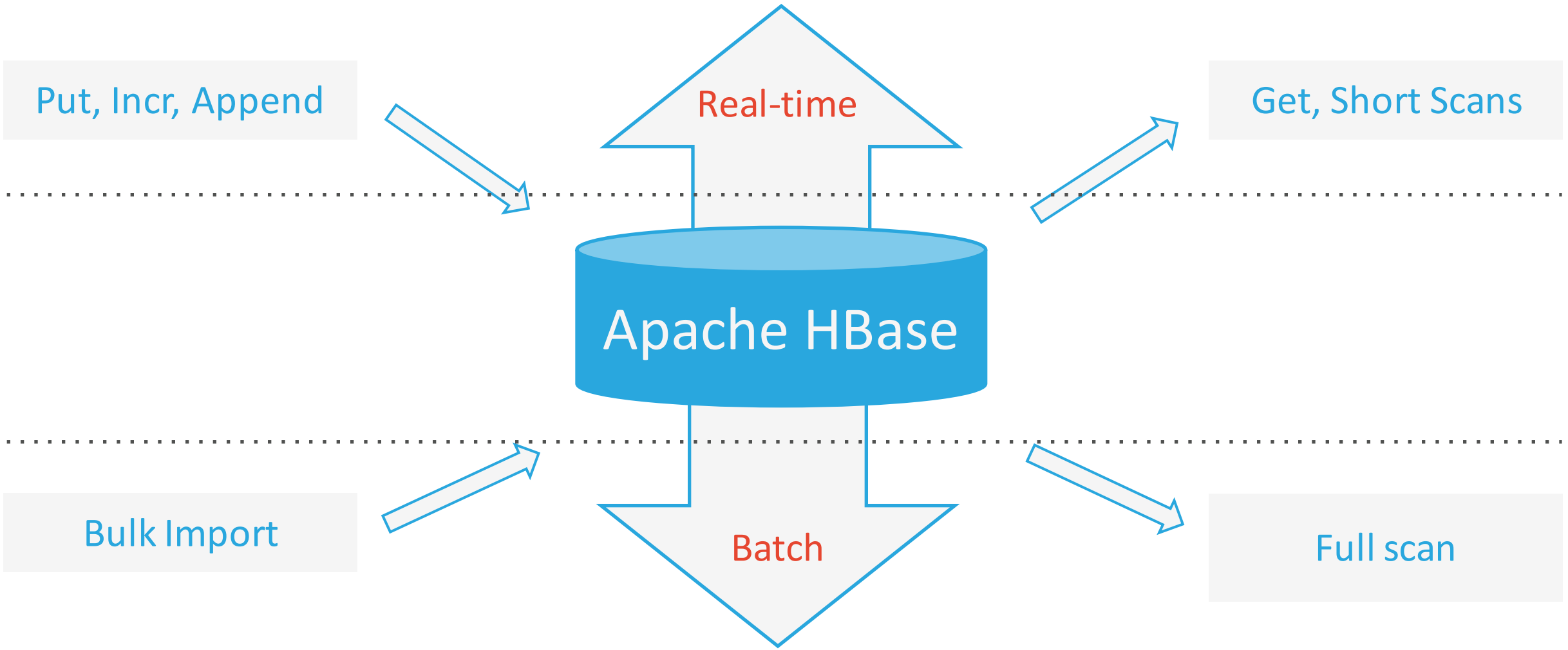
So, what’s the best way?

Depends on your use case

Bottom-line: Disk I/O takes times.

- Limited disk read-write heads in a cluster

- Use the I/O bandwidth of your cluster efficiently



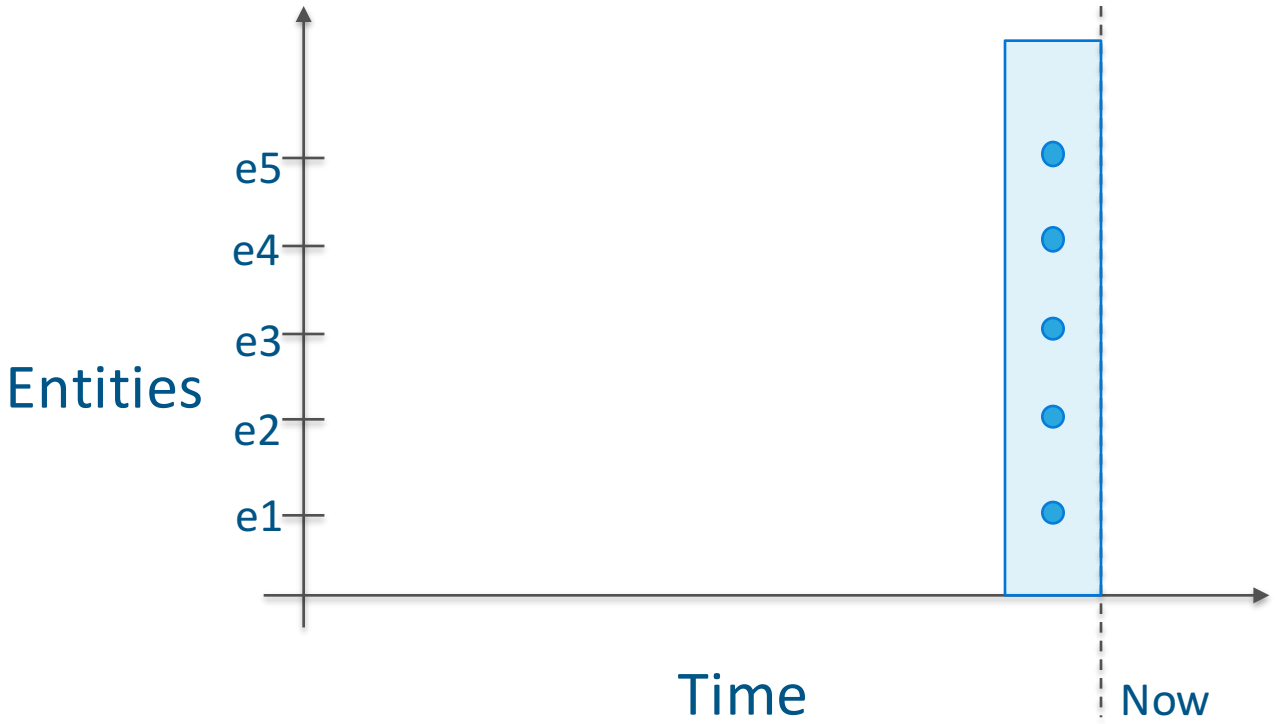
cloudera

Let’s dive into use case ...

cloudera

Simple Entities

- Purely entity data, no relation between entities
  - Often from many different sources
  - Could be a well-done de-normalized RDBMS



cloudera

Simple Entities : Schema

- One row per entity
- Row key => entity ID, or hash of entity ID
- Column => Property / field, possibly timestamp

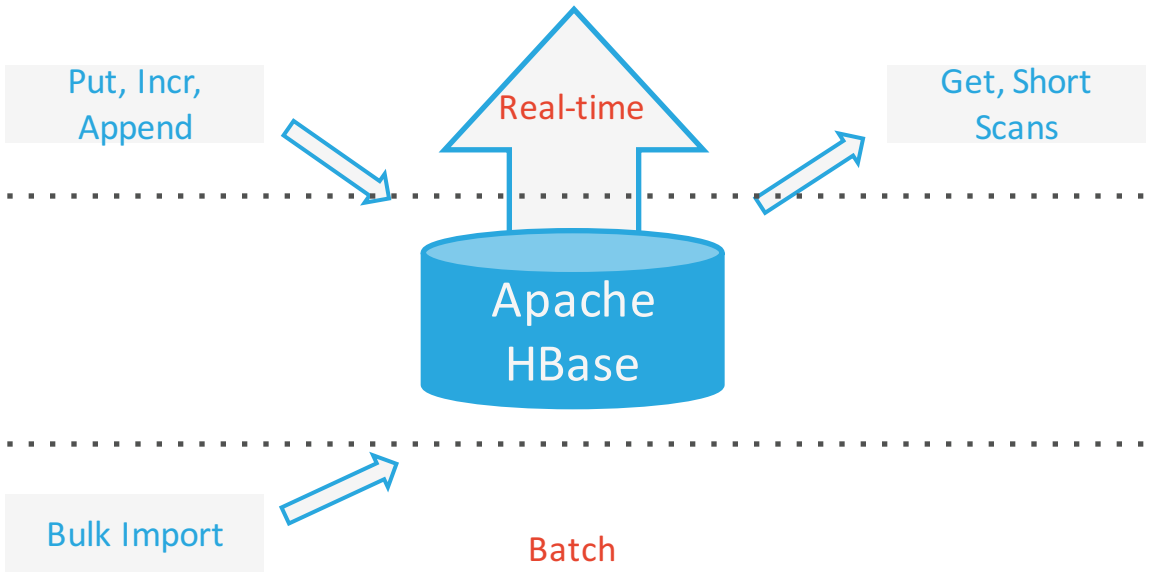
cloudera

Simple Entities : Example

OCLC : Online Computer Library Center

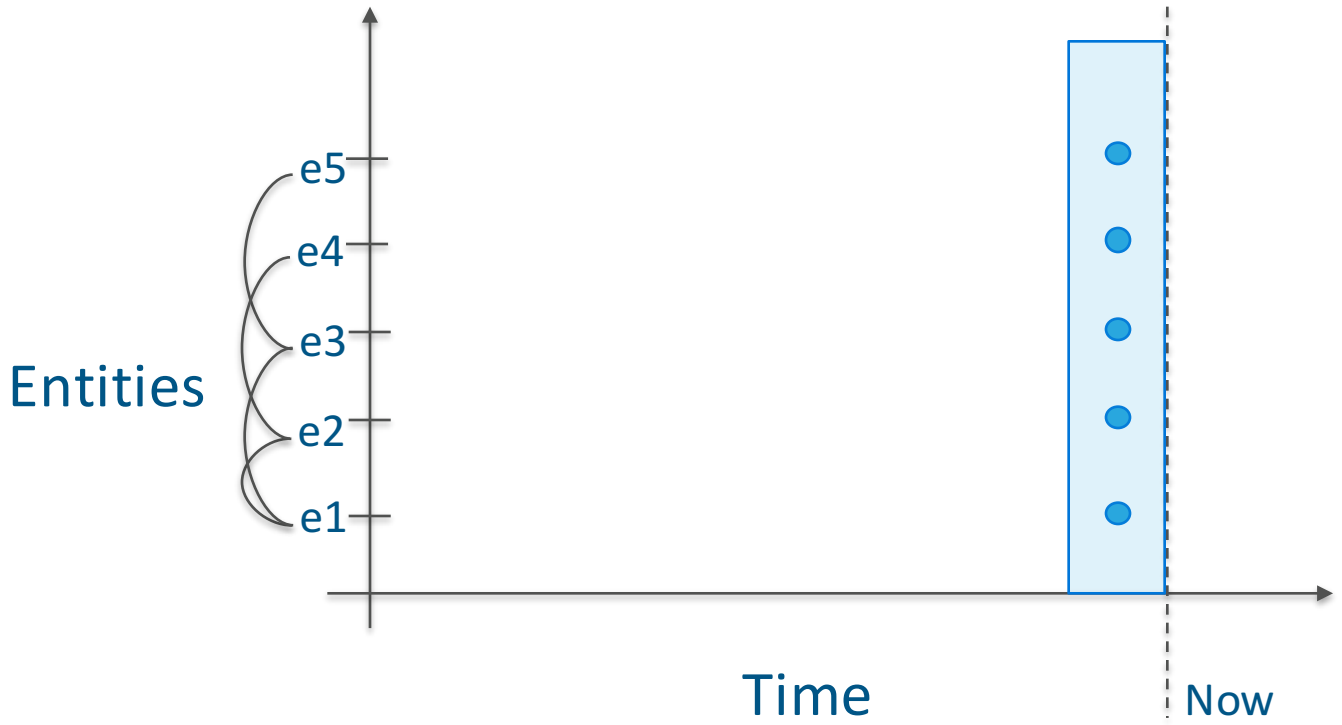
Workloads:

- Lookup books → Real time read
- Add new book one at a time, update information about existing books, issue books → Real-time write
- New library joins the group, import its data → Batch write



Linked Entities (Graph Data)

- Entity are linked to form a graph



cloudera

Linked Entities (Graph Data) : Schema

- One row per Node (Entity)
- Row key => Node ID (Entity ID)
- One Column Family to store edges =>

  “Relationship:OtherNodeID”
- Value => Meta data about relationship
- Second column family to store other information about node

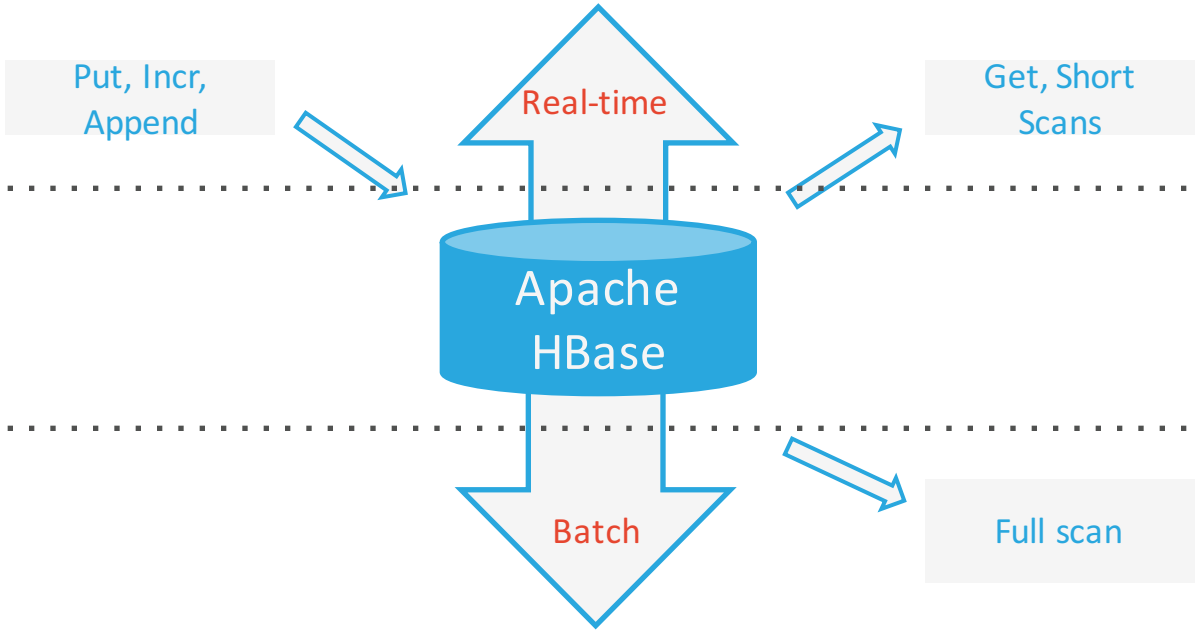
cloudera

Linked Entities (Graph Data) : Example

Social Network (Facebook)

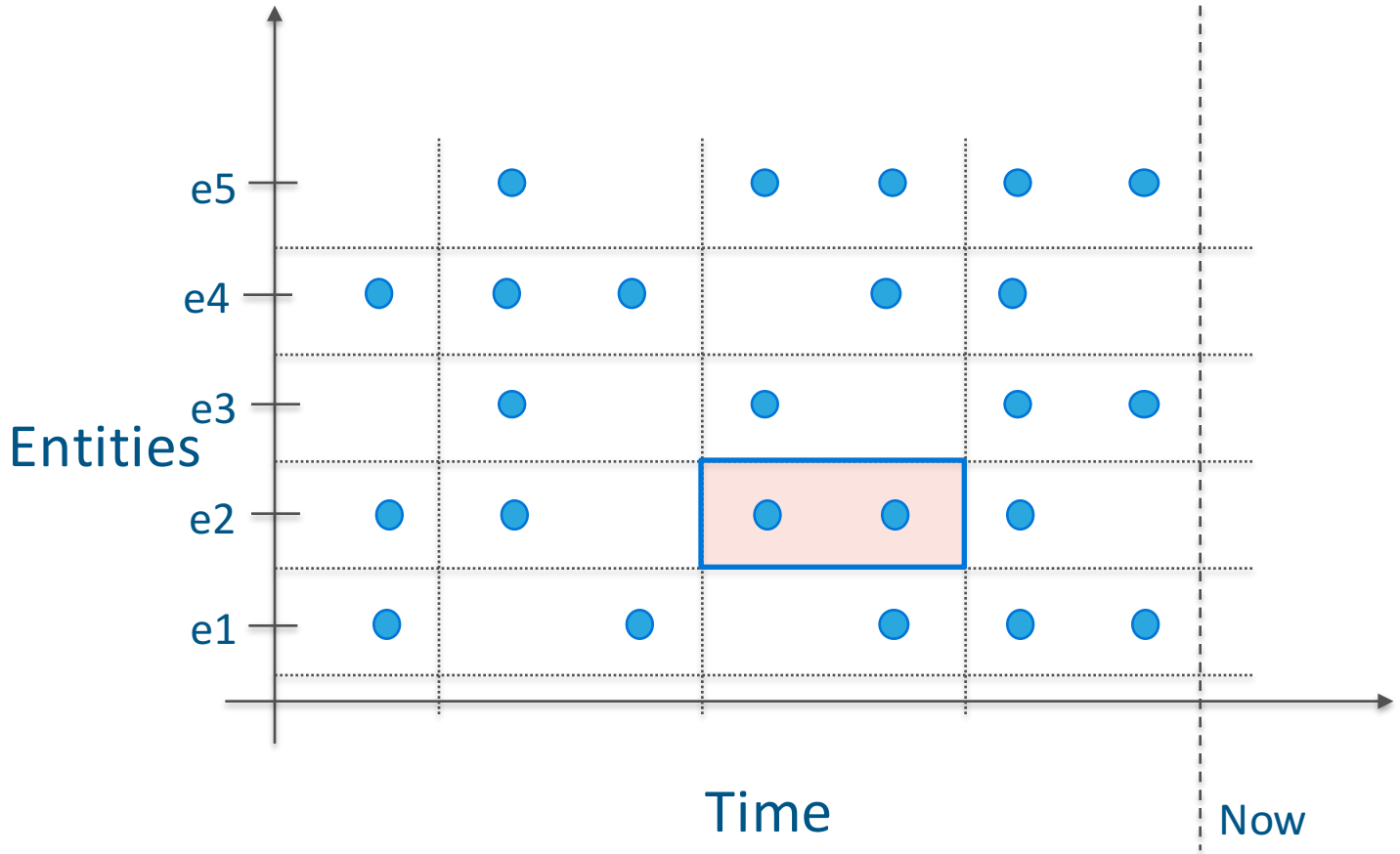
Workloads:

- Get any info about a user → Real time read
- Update any info about a user → Real time write
- Limited graph analysis (based on immediate friends) → Batch read



Time-coupled entities

- Events about entities
- **Focus on entities first**



cloudera

Time-coupled entities : Schema

- Row = Entity's events in a time slice
- Row key = Entity ID + (time / k)
- Column Qualifier = timestamp

cloudera

Time-coupled entities: Example

Messaging service

Primary Workload

- Sending a message, update metadata (read, star, move, delete) →

  Real-time write

- Reading a message, get last N messages → Real-time read

cloudera

HBase is great!

But there are some use cases which are better off without it...

Current HBase weak spots

- HBase architecture can handle a lot
  - Engineering tradeoffs optimize for some use cases
  - HBase can still do things it is not optimal for
  - Other systems are fundamentally more efficient for some workloads

A not so good use case: Large Blob Store

- Saving large objects >50 MB per cell
- Examples
  - Raw video storage in HBase
- Problems:
  - **Write amplification** when re-optimizing data for read (compactions on large unchanging data)
- New: Medium Object (MOB) supported (lots of 100KB-10MB cells)

Another not good use case: Analytic archive

- Store data chronologically, time as primary index
- Schema
  - Row key: timestamp
  - Monotonically increasing row key
  - Column qualifiers: properties with data or counters
- Example
  - Machine logs organized **by timestamp** (causes write hot-spotting)

That’s all folks!

cloudera

cloudera

Questions ?

Sources

- A Survey of HBase Application Archetypes
  - Lars George, Jon Hsieh
  - http://www.slideshare.net/HBaseCon/case-studies-session-7
- OpenTSDB 2.0
  - Benoit Sigoure, Chris Larsen
  - http://www.slideshare.net/HBaseCon/ecosystem-session-6
- Hadoop and HBase: Motivations, Use cases and Trade-offs
  - Jon Hsieh
- Phoenix
  - https://phoenix.apache.org

cloudera