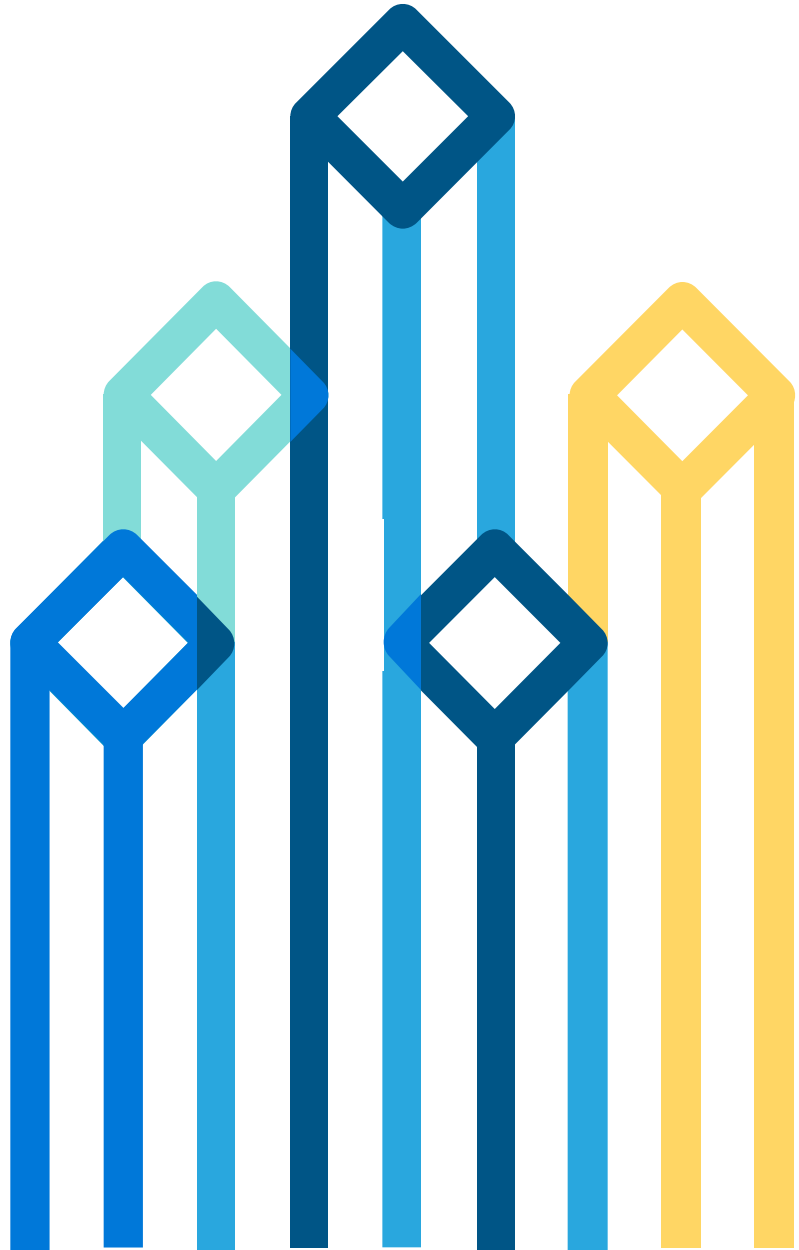
cloudera®



Apache HBase: Overview, Hands-On, and Use Cases

Apekshit Sharma

Dima Spivak

Who we are

Apekshit Sharma

- Distributed Software Engineer, Cloudera
  - Software Engineer, Google
- Apache HBase contributor
- Performance improvements and configuration framework

Dima Spivak (@dimaspivak)

- Distributed Software Engineer, Cloudera
  - Research Assistant (Physics), University of Minnesota
- Apache HBase contributor
- Test frameworks and automation

cloudera

Contents

- Motivation
- Introduction to Apache HBase
  - Data model
- Hands-On: Installation, HBase shell
- Break
- A slightly more in-depth introduction to Apache HBase
  - Apache Hadoop
  - System internals
  - APIs
- Break

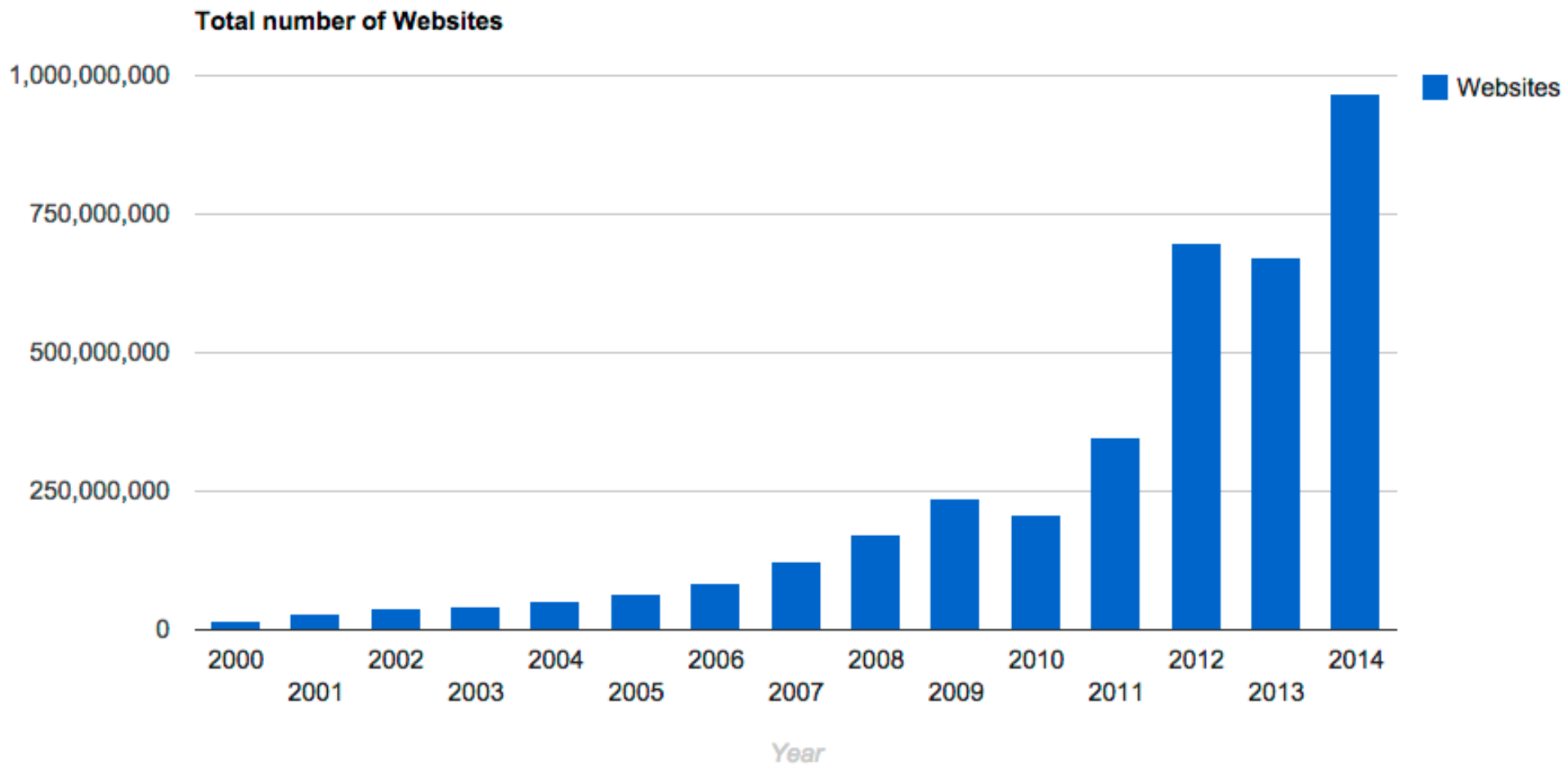
cloudera

Contents

- Industry use cases & patterns
- Augmenting HBase
  - OpenTSDB
  - Apache Phoenix

cloudera

Motivation



http://www.internetlivestats.com/total-number-of-websites/

cloudera

Motivation

"We've known it for a long time: the web is big."

– Jesse Alpert & Nissan Hajaj, Google

http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html

Motivation

- Indexing the internet has challenges:
  - Scale
    - Volume
    - Rate
  - Diversity of content
    - URLs
    - High-resolution images
    - Video
  - Access

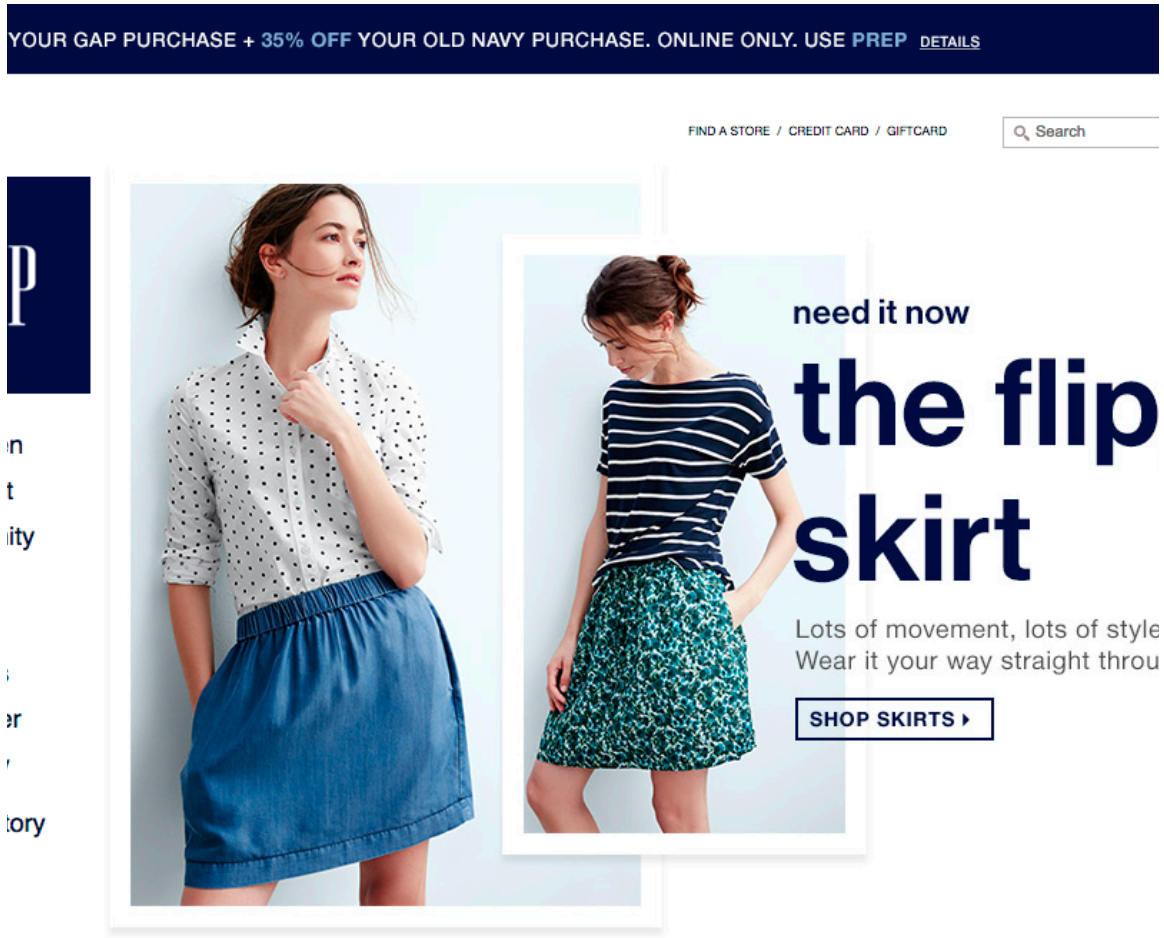
Motivation

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

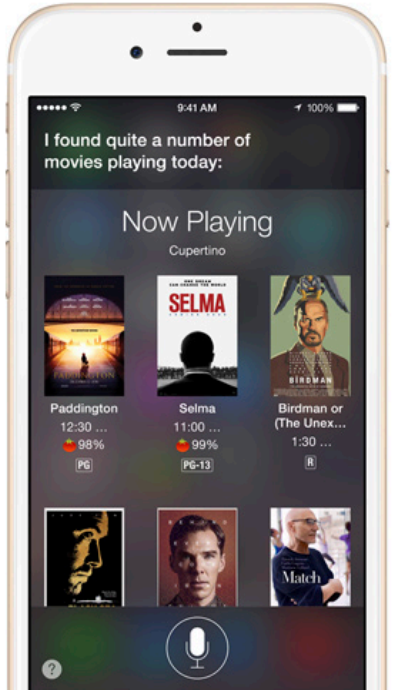
- What if you're not trying to index the internet?



Motivation

- Data for analytical processing
- User-facing real-time platforms

"What movies are playing today?"



cloudera

Introduction to Apache HBase

- "Apache HBase™ is the Hadoop database, a distributed, scalable, big data store."

  http://hbase.apache.org/

Introduction to Apache HBase

- Apache HBase is an open source, horizontally scalable, consistent, random access, low latency data store built on top of Apache Hadoop.

Apache HBase is open source

- Apache 2.0 License
- A community project with committers and contributors from diverse organizations
  - Cloudera, Facebook, Salesforce.com, Huawei, TrendMicro, eBay, HortonWorks, Intel, Twitter, …
- Code license means anyone can modify and use the code.

Apache HBase is horizontally scalable

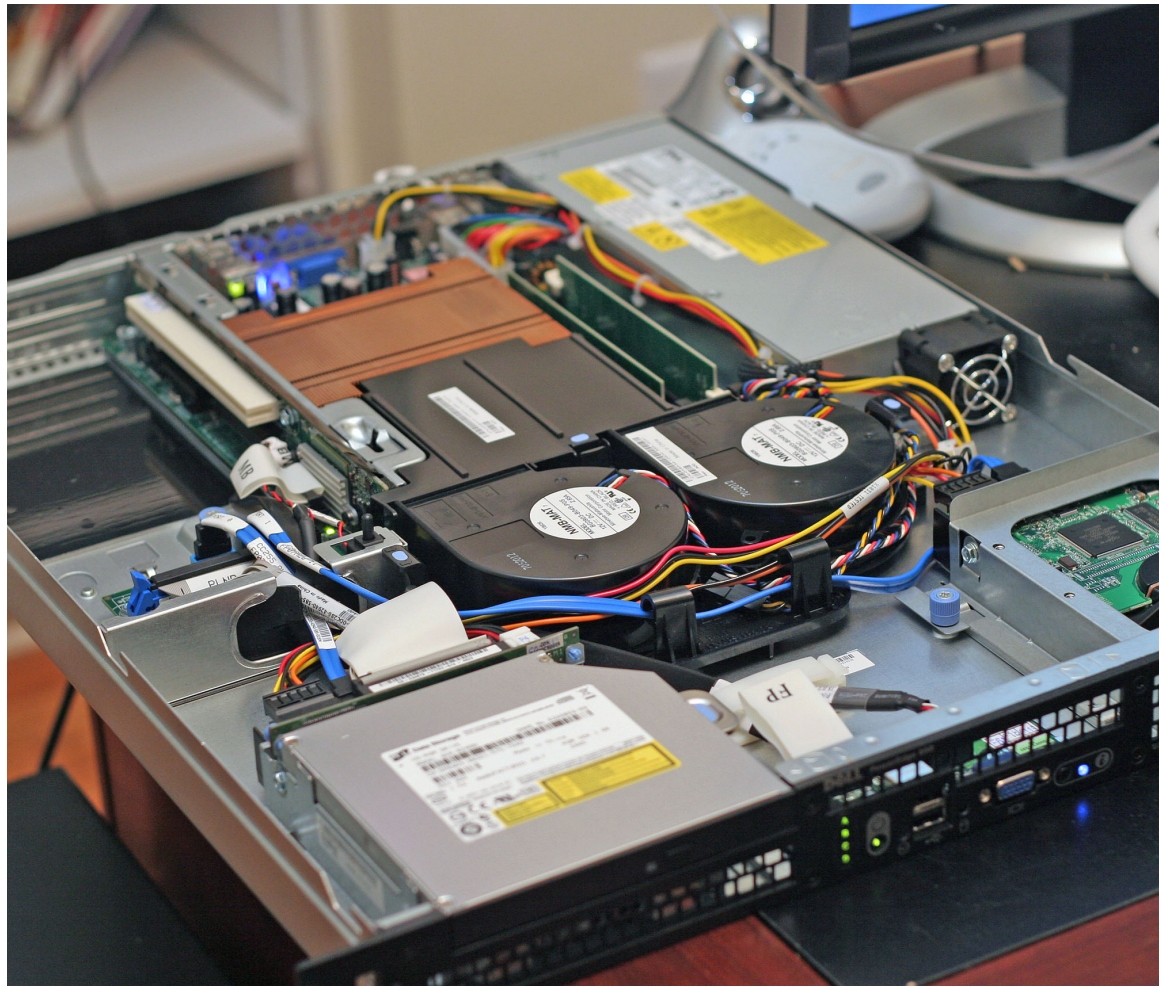
- Adding more servers linearly increases performance and capacity
  - Storage capacity
  - Input/output operations
- Store and access data on commodity servers

- Largest cluster: > 3000 nodes, > 100 PB
- Average cluster: 10-40 nodes, 100-400TB

cloudera

Apache HBase is horizontally scalable

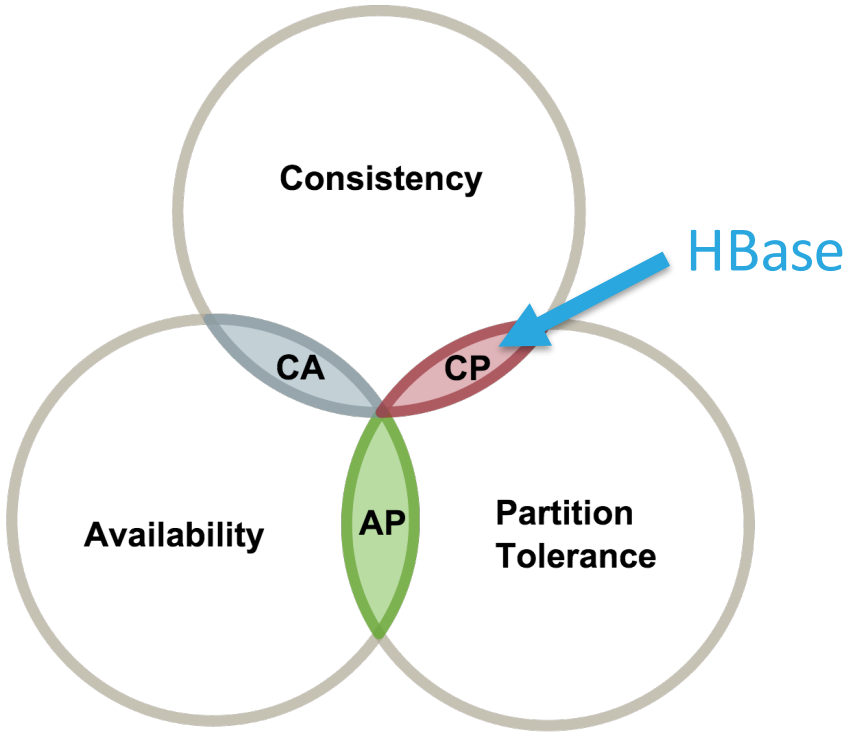
- Commodity servers (crica 2015)
  - 12-24 2-4TB hard disks
  - 2 octa-core CPUs, 2-3 GHz
  - 64 - 512 GBs of RAM
  - 10 Gbps ethernet
  - $5,000 - $10,000 / machine



cloudera

Apache HBase is consistent

- Brewer's theorem
  - Consistency
  - Availability
  - Partition tolerance



cloudera

Data model

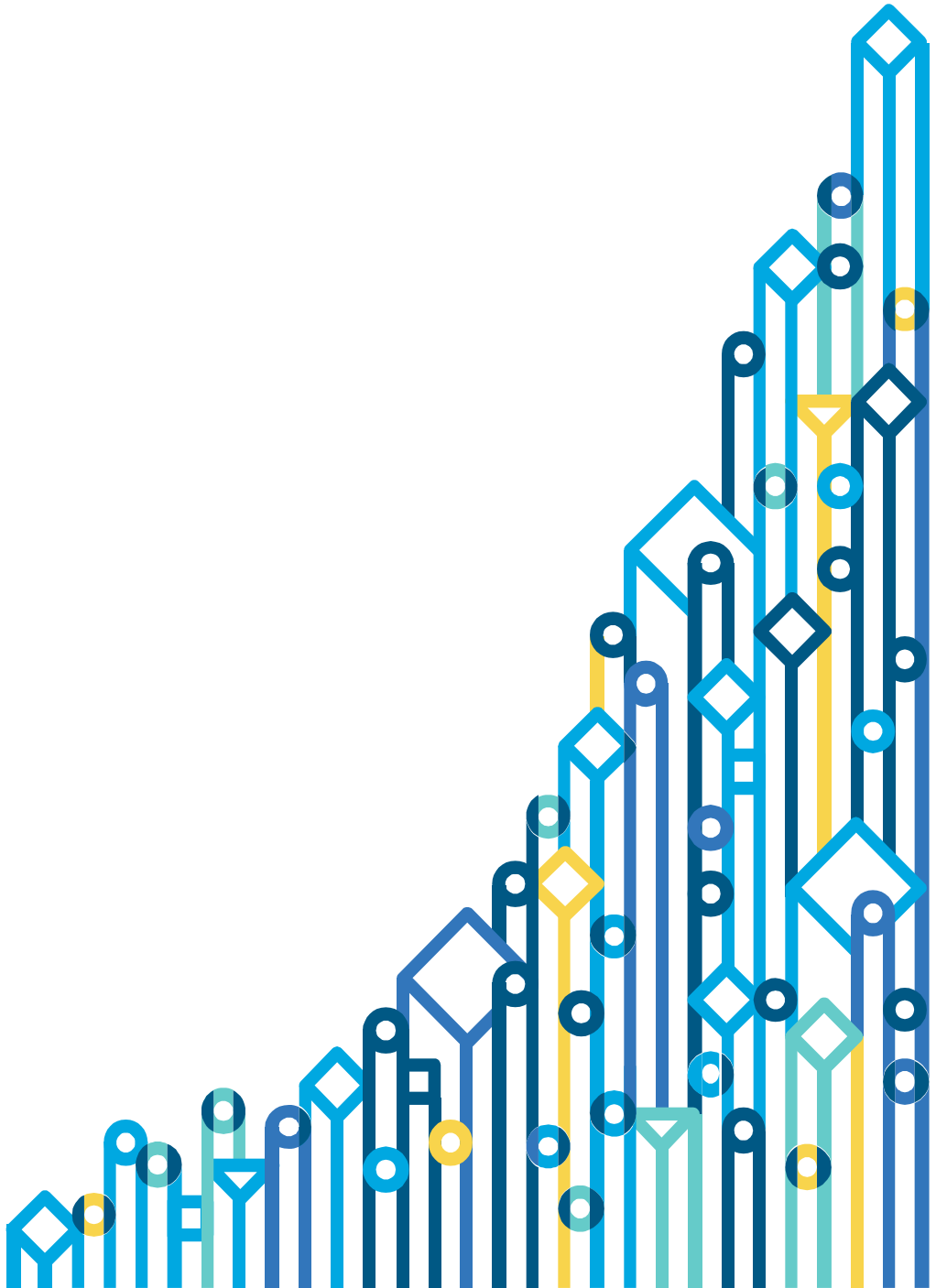
- Data is stored… in a big table
  - Sorted map datastore
- Tables consist of sorted rows, each of which has a primary row key
- Each row has a set of columns
- A column is specified as a column family and column qualifier pair
- A given cell (row, column family:column qualifier) can have different time-stamped values

Data model

Row key	info:height	info:state	roles:hadoop	roles:hbase
cutting	‘9ft’	‘CA’	‘Founder’	
todd	‘5ft7’	‘CA’	‘PMC’ (ts=2011) ‘Committer’ (ts=2010)	‘Committer’

cloudera

cloudera®



Hands On

Apache HBase installation

The HBase shell

http://pastebin.com/nMkZeq5S

Whats up for the next 1 hour?

Understanding basic architecture of

HDFS
(Apache Hadoop)

APACHE HBASE

And, more Hands-On with Apache HBase.

cloudera

Break

Understanding basic architecture of

Hadoop (HDFS)

Apache Hadoop

open source

commodity servers

horizontally scalable

highly fault-tolerant

massive processing power

cloudera

2 Core Components

Apache Hadoop

HDFS
(Hadoop Distributed File System)

MapReduce + YARN

History

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

2003

GFS

- distributed file system
- commodity servers
- horizontally scalable
- highly fault-tolerant
- proprietary

cloudera

HDFS

- distributed file system
- commodity servers
- horizontally scalable
- highly fault-tolerant
- open source

cloudera

HDFS API

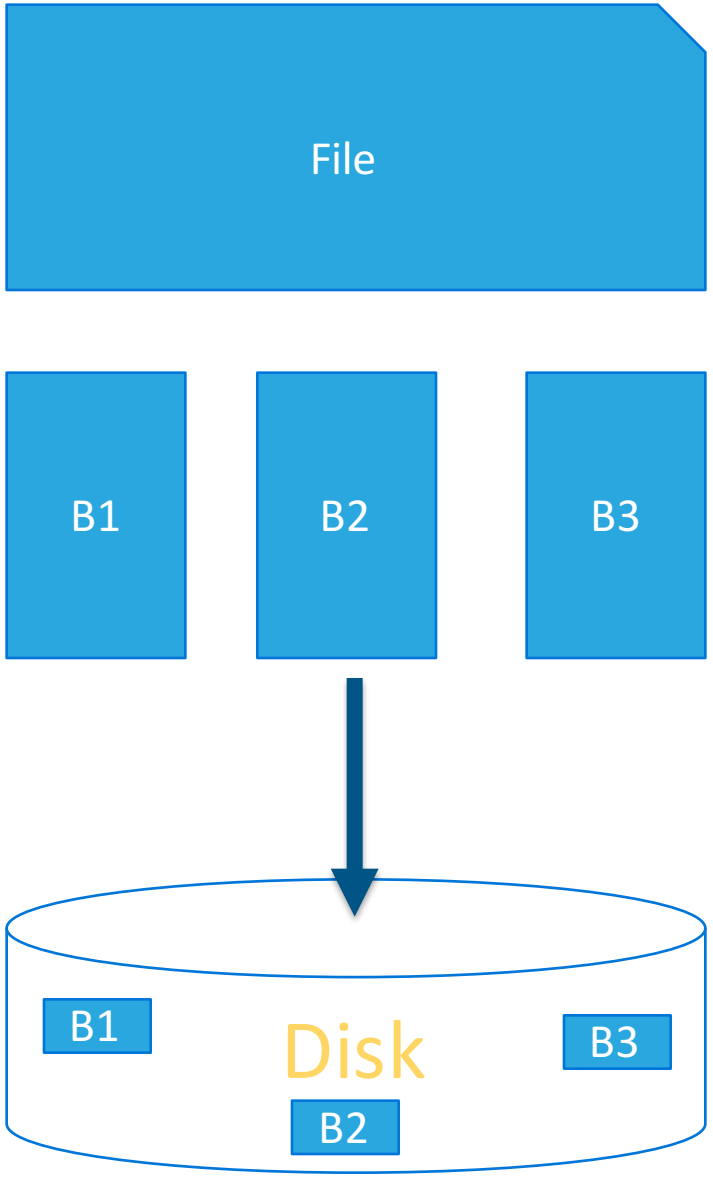
- File
  - Open, Close, Read, Write, Move, etc
- Directories
  - Create, Delete, etc
- Permissions
  - Owners, Groups, rwx permissions

cloudera

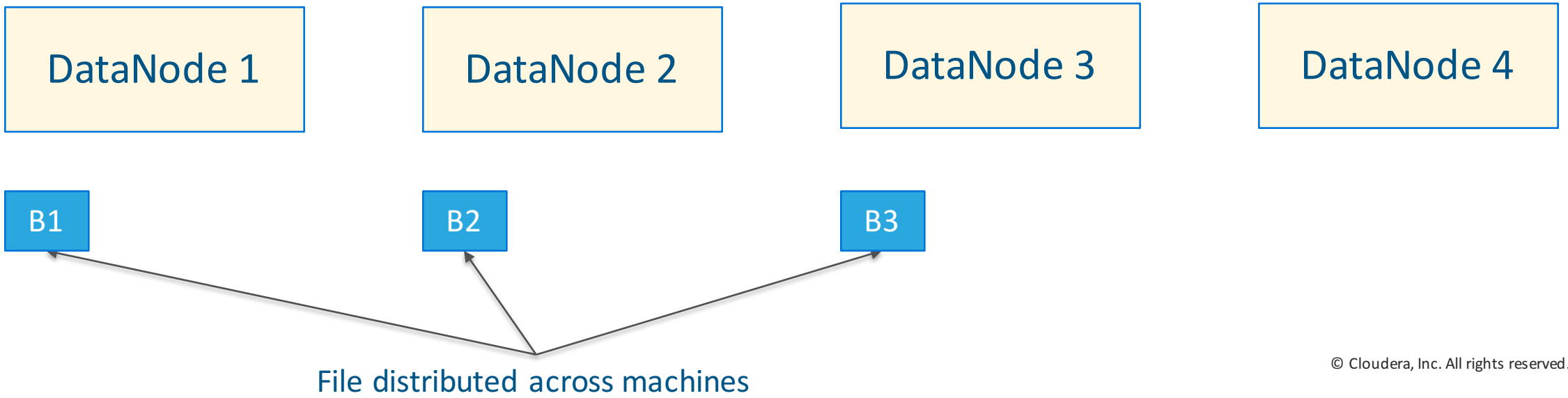
Basic Architecture of HDFS

Local file system

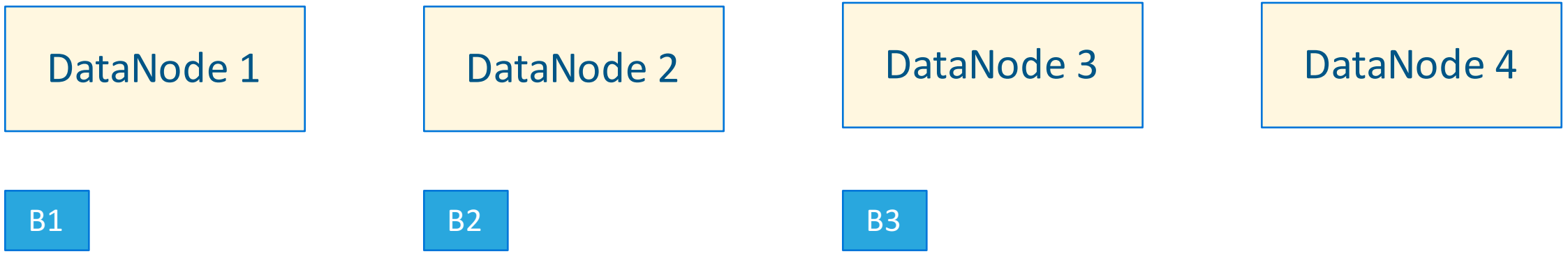
File system will split the file into blocks



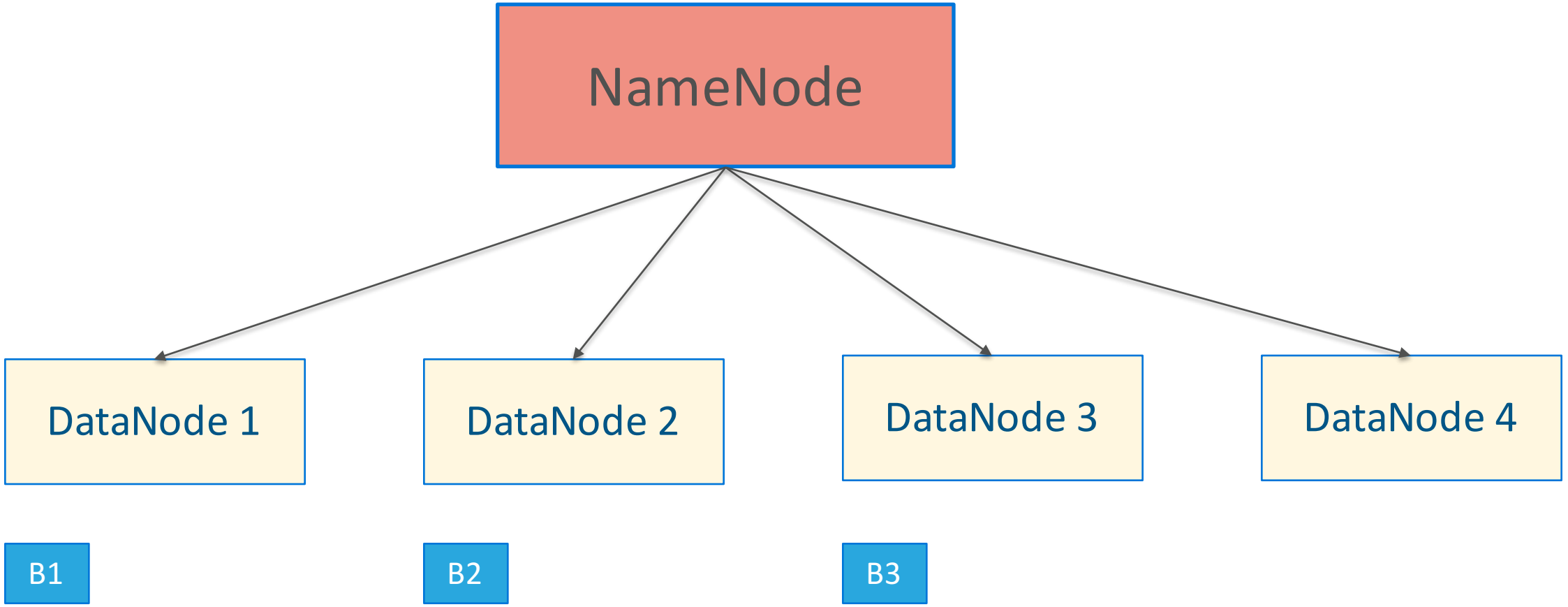
HDFS



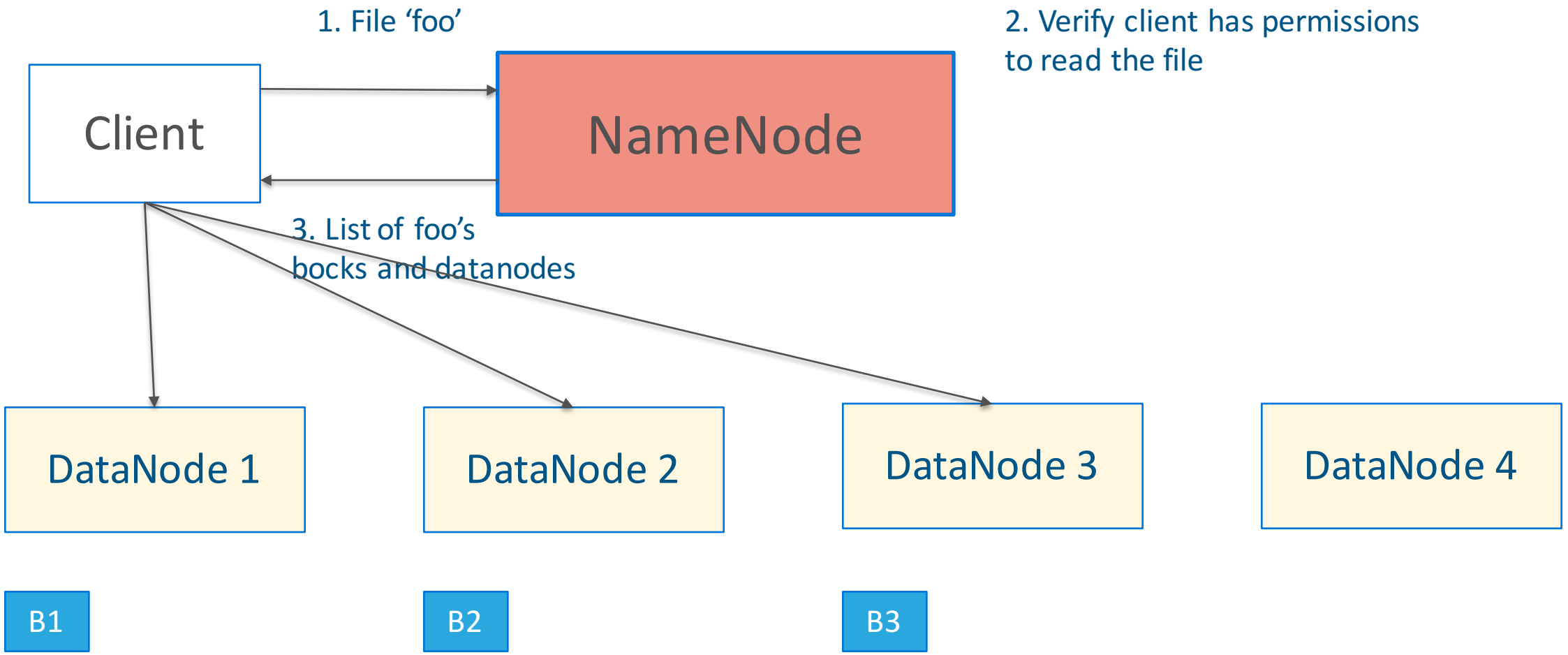
HDFS DataNode



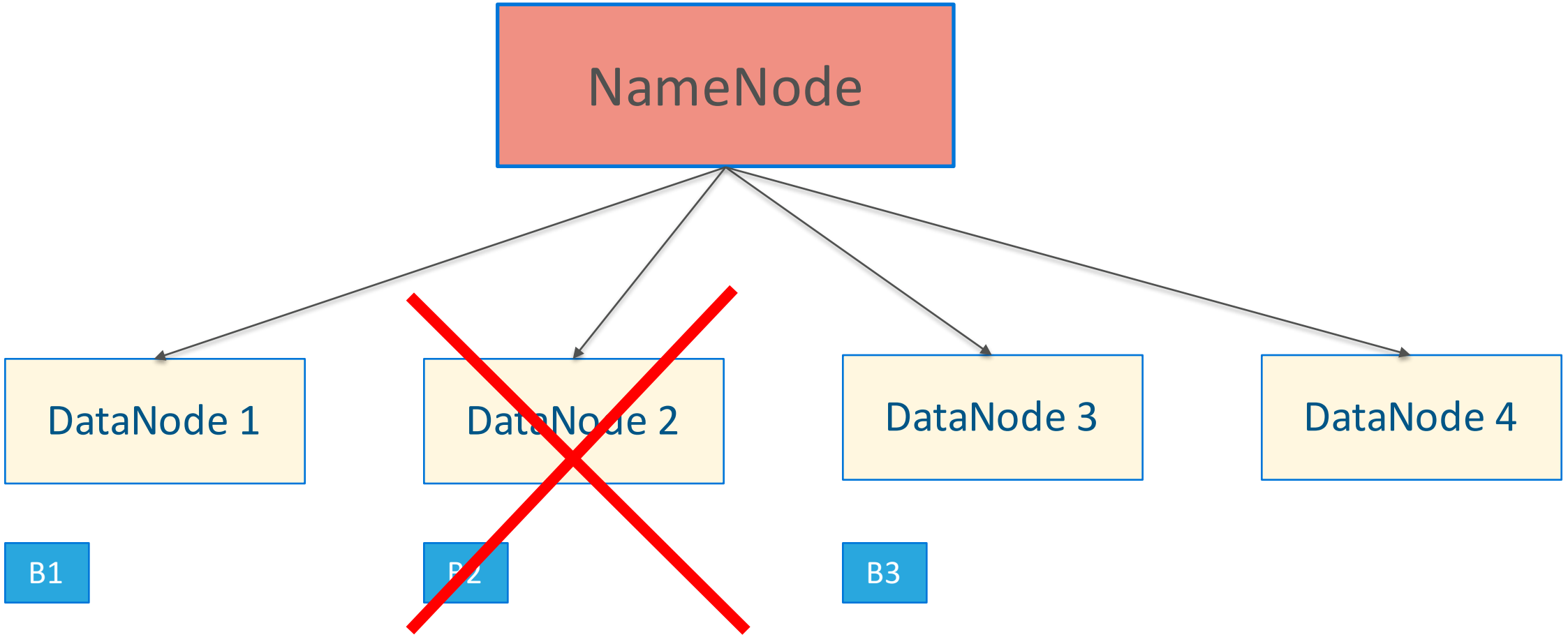
HDFS NameNode



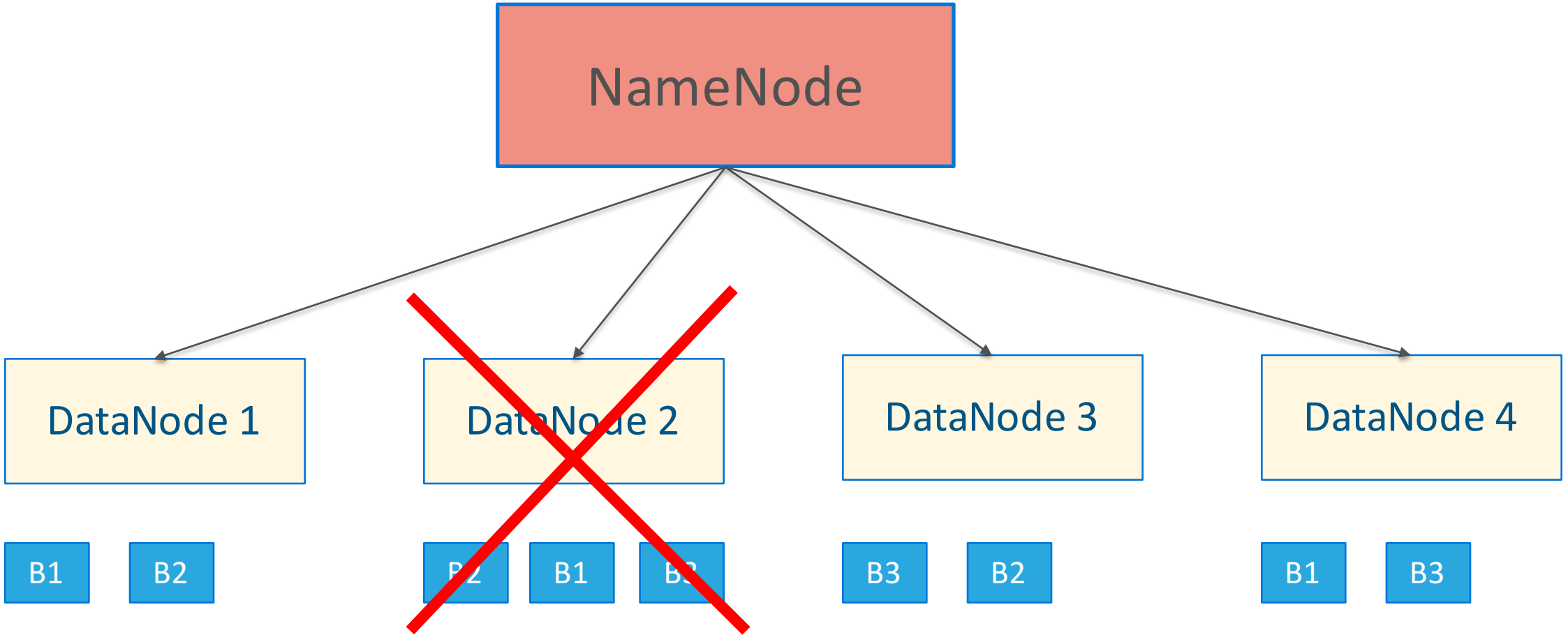
HDFS Reading a file



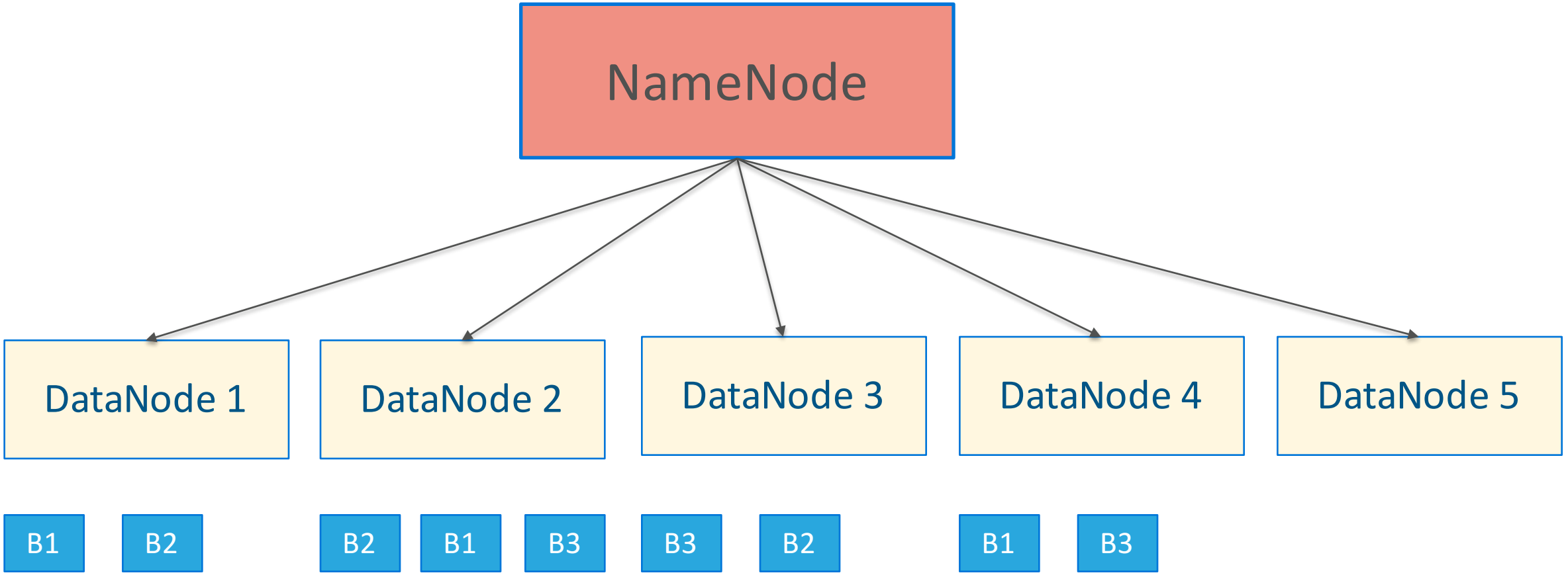
HDFS Fault tolerance



HDFS Redundancy



HDFS Horizontal Scalability



Let’s look at some existing HDFS systems...

- ## Yahoo! HDFS Clusters

  40k+ servers, 100k+ CPUs, 450PB data

- ## Facebook HDFS Cluster

  15TB new data per day

  1200+ machines, 30PB in one cluster

- *Lots* of 5-40 node clusters at companies *without* petabytes of data (web, retail, finance, telecom, research, government)

But…. there are restrictions!

It’s not a magic wand!

Files are **append only**

- Access Model : Write-once-read-many
- Can not change existing contents

cloudera

Not designed for small files

- Block sizes are in MB (default 128MB)
  - Designed for typical GBs / TBs of file sizes
  - Normal files system have 4kb block size!

Summary

HDFS is a great distributed file system!

- Store massive data
- Scalable
- High throughput
- Fault tolerance

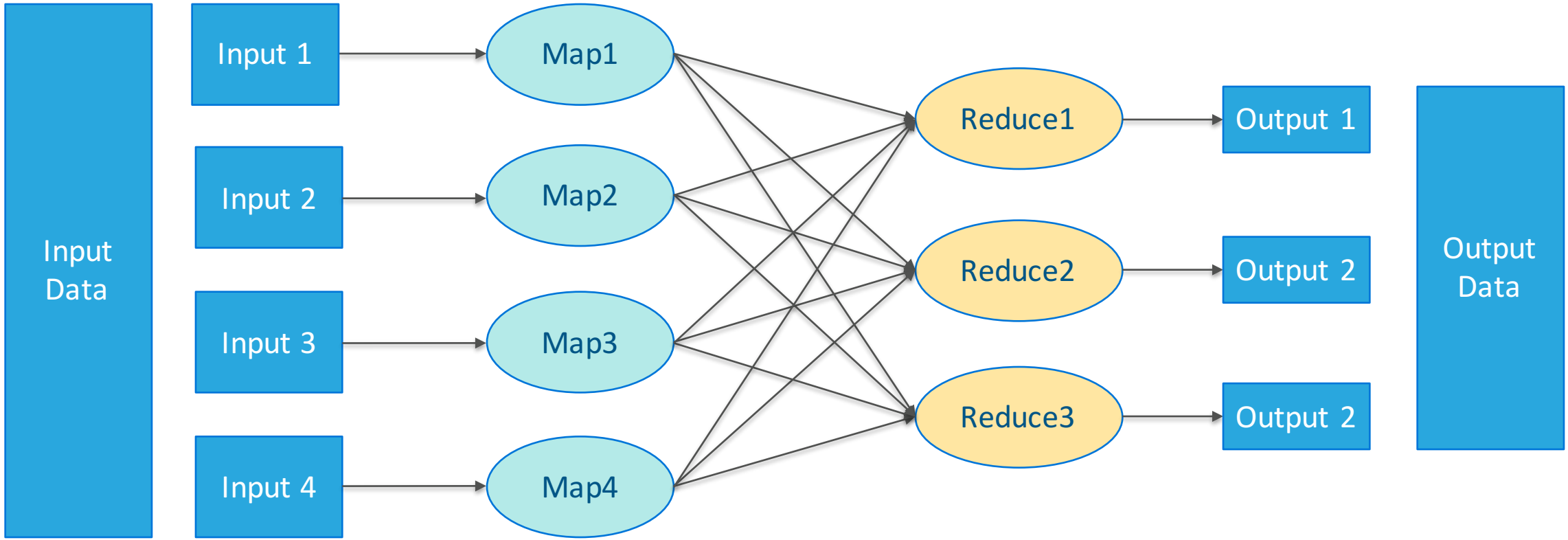
cloudera

MapReduce

- Distributed processing framework
- Commodity machines
- Fault tolerance

cloudera

MapReduce

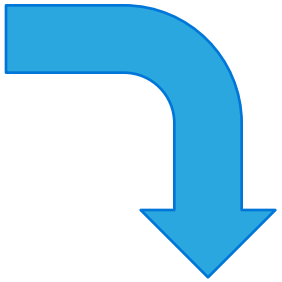


cloudera

cloudera

HBase Architecture

Name	Weight	UPC	Price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98



	info:weight	info:upc	info:price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98

cloudera

HBase Architecture

	info:weight	info:upc	info:price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98

cloudera

HBase Architecture

	info:weight	info:upc	info:price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98
A New Product	4 Oz	xxxxxxxx	$9.99

cloudera

HBase Architecture

	info:weight	info:upc	info:price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98
A New Product	4 Oz	xxxxxxxx	$9.99
Yet Another New Product	8 Oz	xxxxxxxx	$19.99

cloudera

HBase Architecture

	info:weight	info:upc	info:price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98
A New Product	4 Oz	xxxxxxxx	$9.99
Yet Another New Product	8 Oz	xxxxxxxx	$19.99
Four More Products (1)	16 Oz	xxxxxxxx	$9.99
Four More Products (2)	16 Oz	xxxxxxxx	$9.99
Four More Products (3)	16 Oz	xxxxxxxx	$9.99
Four More Products (4)	16 Oz	xxxxxxxx	$9.99

cloudera

HBase Architecture | Regions

- Tables are chopped up into regions (split).
- A region is only served by a single “region server” at a time.
- RegionServer can serve multiple regions.

HBase Architecture | Regions

	info:weight	info:upc	info:price
A New Product	4 Oz	xxxxxxxx	$9.99
Four More Products (1)	16 Oz	xxxxxxxx	$9.99
Four More Products (2)	16 Oz	xxxxxxxx	$9.99
Four More Products (3)	16 Oz	xxxxxxxx	$9.99
Four More Products (4)	16 Oz	xxxxxxxx	$9.99

Served by RegionServer on machine 2

	info:weight	info:upc	info:price
Prego Tomato Sauce	67 Oz	xxxxxxxx	$4.97
Trumoo Lowfat Chocolate Milk	128 Oz	xxxxxxxx	$2.99
Gatorade Lemon-Lime	64 Oz	xxxxxxxx	$3.98
Yet Another New Product	8 Oz	xxxxxxxx	$19.99

Served by RegionServer on machine 3

cloudera

HBase Architecture | Regions

	info:price	info:upc	info:weight
A New Product	$9.99	xxxxxxxx	4 Oz
Four More Products (1)	$9.99	xxxxxxxx	16 Oz
Four More Products (2)	$9.99	xxxxxxxx	16 Oz
Four More Products (3)	$9.99	xxxxxxxx	16 Oz
Four More Products (4)	$9.99	xxxxxxxx	16 Oz

Served by RegionServer on machine 2

	info:price	info:upc	info:weight
Gatorade Lemon-Lime	$3.98	xxxxxxxx	64 Oz
Prego Tomato Sauce	$4.97	xxxxxxxx	67 Oz
Trumoo Lowfat Chocolate Milk	$2.99	xxxxxxxx	128 Oz
Yet Another New Product	$19.99	xxxxxxxx	8 Oz

Served by RegionServer on machine 3

cloudera

HBase Architecture

	info:price	info:upc	info:weight	available:store1	available:store2	available:store3
A New Product	$9.99	xxxxxxxx	4 Oz	Yes	Yes	Yes
Four More Products (1)	$9.99	xxxxxxxx	16 Oz	Yes	Yes	Yes
Four More Products (2)	$9.99	xxxxxxxx	16 Oz	Yes	Yes	No
Four More Products (3)	$9.99	xxxxxxxx	16 Oz	Yes	Yes	No
Four More Products (4)	$9.99	xxxxxxxx	16 Oz	No	No	No

	info:price	info:upc	info:weight	available:store1	available:store2	available:store3
Gatorade Lemon-Lime	$3.98	xxxxxxxx	64 Oz	Yes	Yes	Yes
Prego Tomato Sauce	$4.97	xxxxxxxx	67 Oz	Yes	No	Yes
Trumoo Lowfat Chocolate Milk	$2.99	xxxxxxxx	128 Oz	No	No	Yes
Yet Another New Product	$19.99	xxxxxxxx	8 Oz	Yes	Yes	Yes

cloudera

HBase Architecture

	info:price	info:upc	info:weight	available:store1	available:store2	available:store3
A New Product	$9.99	xxxxxxxx	4 Oz	Yes	Yes	Yes
Four More Products (1)	$9.99	xxxxxxxx	16 Oz	Yes	Yes	Yes
Four More Products (2)	$9.99	xxxxxxxx	16 Oz	Yes	Yes	No
Four More Products (3)	$9.99	xxxxxxxx	16 Oz	Yes	Yes	No
Four More Products (4)	$9.99	xxxxxxxx	16 Oz	No	No	No

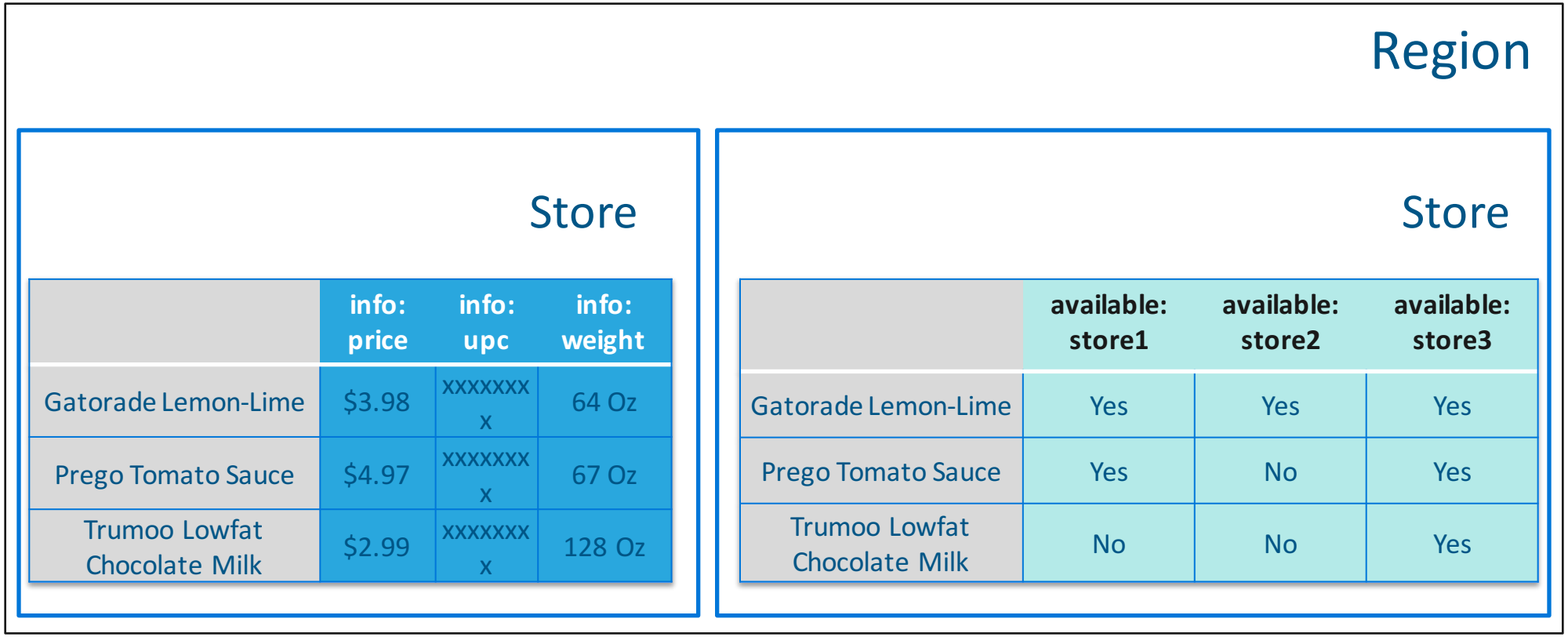
	info:price	info:upc	info:weight	available:store1	available:store2	available:store3
Gatorade Lemon-Lime	$3.98	xxxxxxxx	64 Oz	Yes	Yes	Yes
Prego Tomato Sauce	$4.97	xxxxxxxx	67 Oz	Yes	No	Yes
Trumoo Lowfat Chocolate Milk	$2.99	xxxxxxxx	128 Oz	No	No	Yes
Yet Another New Product	$19.99	xxxxxxxx	8 Oz	Yes	Yes	Yes

cloudera

HBase Architecture | Column family

- A column family is a set of related columns.
  - Group sets of columns that have similar access patterns.
- Tune read performance.
  - Compression
  - Version retention policies
  - Cache priority

HBase Architecture



	info: price	info: upc	info: weight
Gatorade Lemon-Lime	$3.98	xxxxxxx x	64 Oz
Prego Tomato Sauce	$4.97	xxxxxxx x	67 Oz
Trumoo Lowfat Chocolate Milk	$2.99	xxxxxxx x	128 Oz

	available: store1	available: store2	available: store3
Gatorade Lemon-Lime	Yes	Yes	Yes
Prego Tomato Sauce	Yes	No	Yes
Trumoo Lowfat Chocolate Milk	No	No	Yes

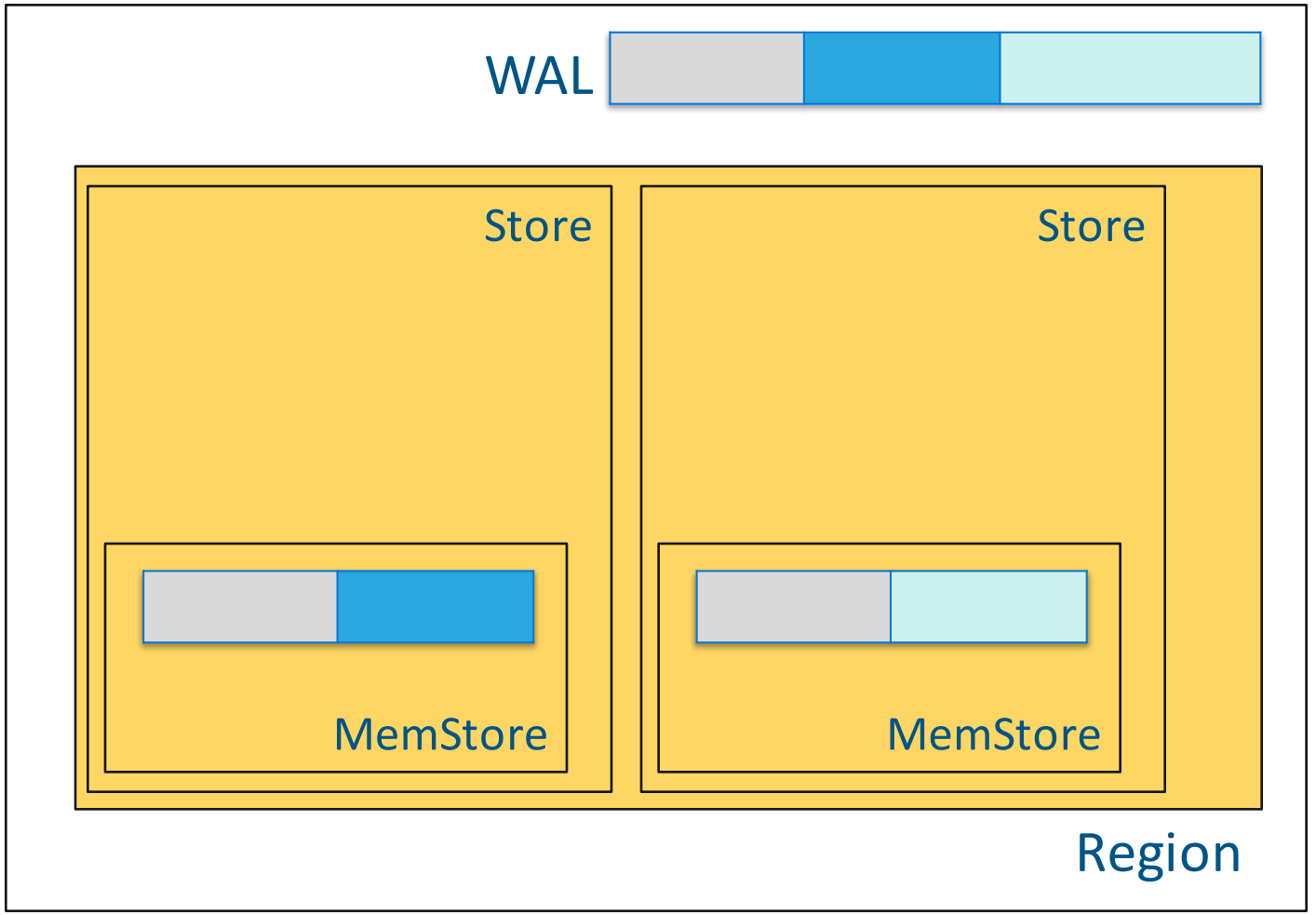
cloudera

HBase Architecture | Write Path

1. Client creates a row to put.
2. Client checks with meta* for which RegionServer hosts this row.
3. Row is written into write-ahead log (WAL).
4. Row is written to MemStore.

HBase Architecture | Write Path

HBase Architecture | Write Path

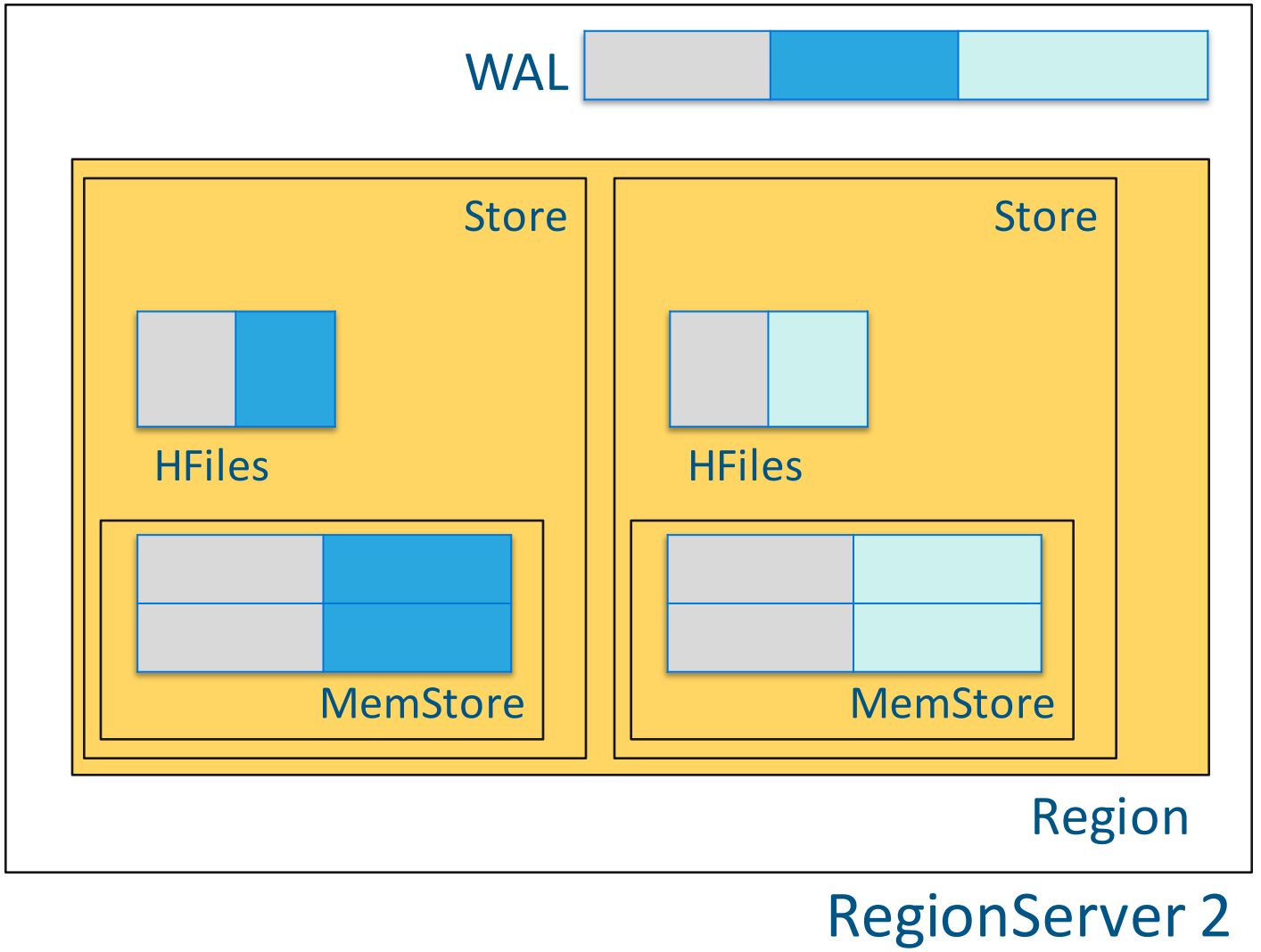


Put

HBase Architecture | Write Path

- When MemStore gets full or a flush is triggered, contents of MemStore are flushed to disk.
  - HFiles are created.

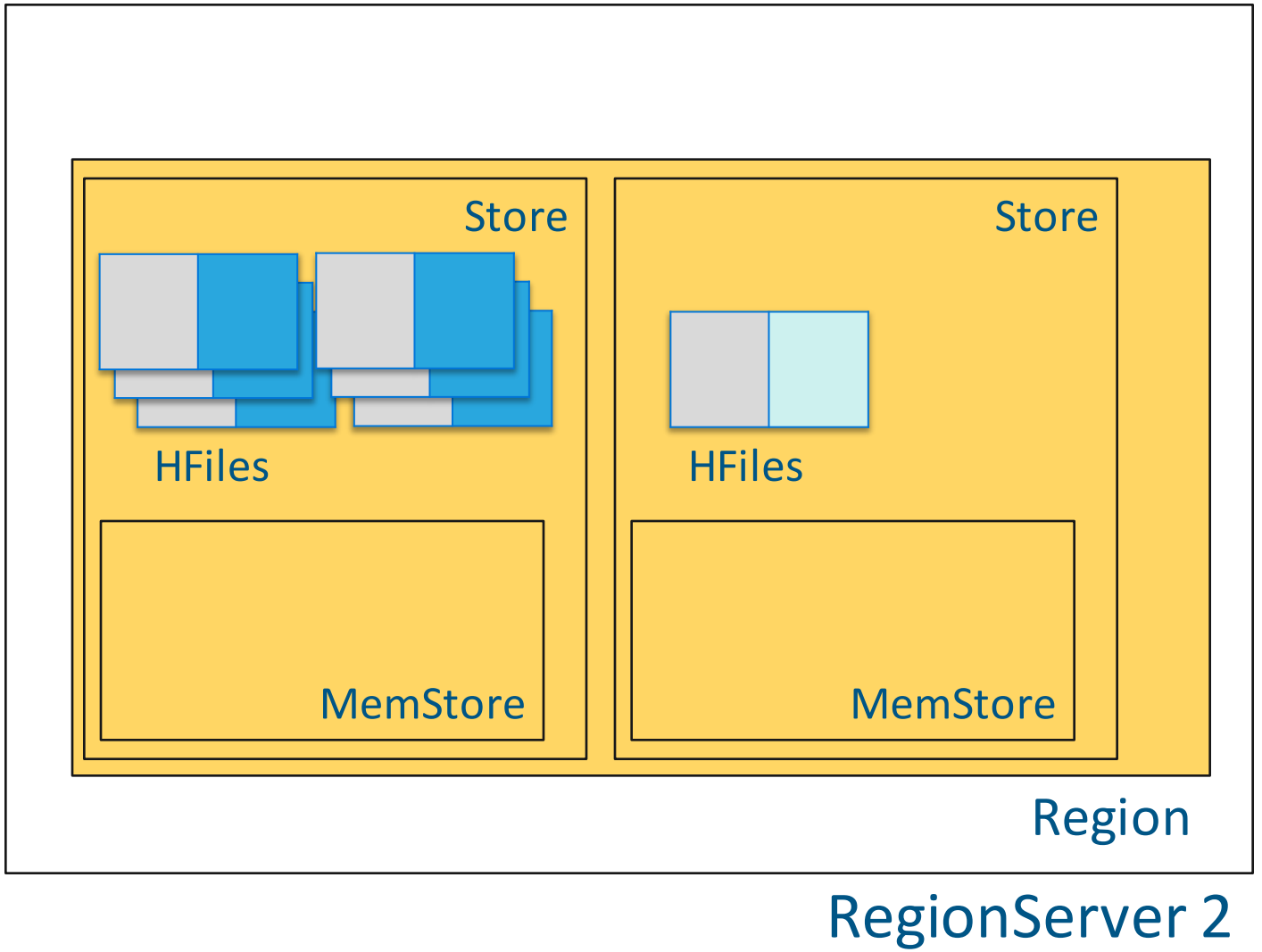
HBase Architecture | Write Path



HBase Architecture | Write Path

- Each subsequent write repeats this process.
  - Write to WAL.
  - Write to MemStore.
  - Flush when MemStore fills or a flush is triggered.
  - Create HFiles.
- Lots of HFiles in a Region mean lots of disk seeks on read.
  - Might be better to combine (compact) HFiles.

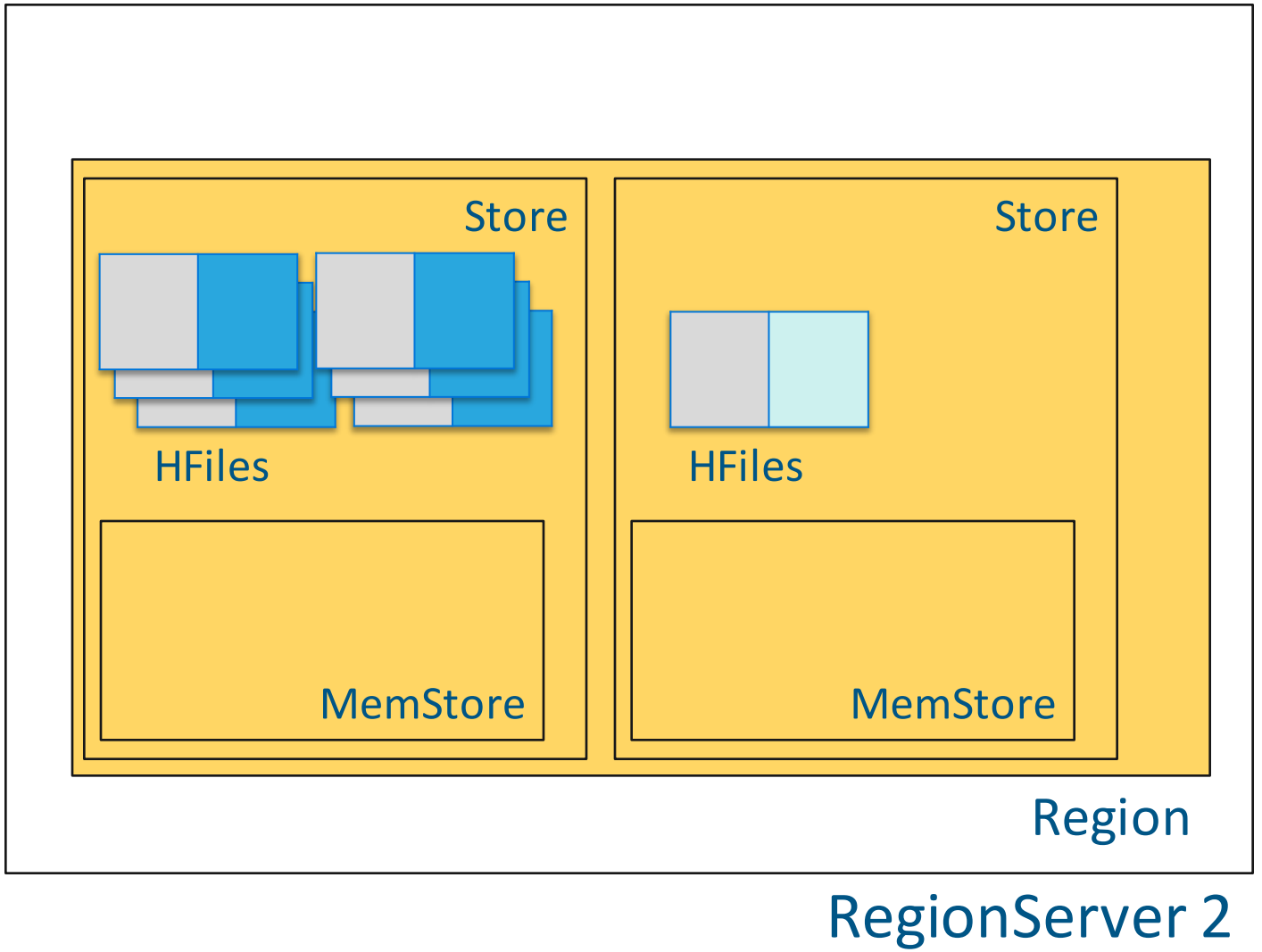
HBase Architecture



HBase Architecture | Compactions

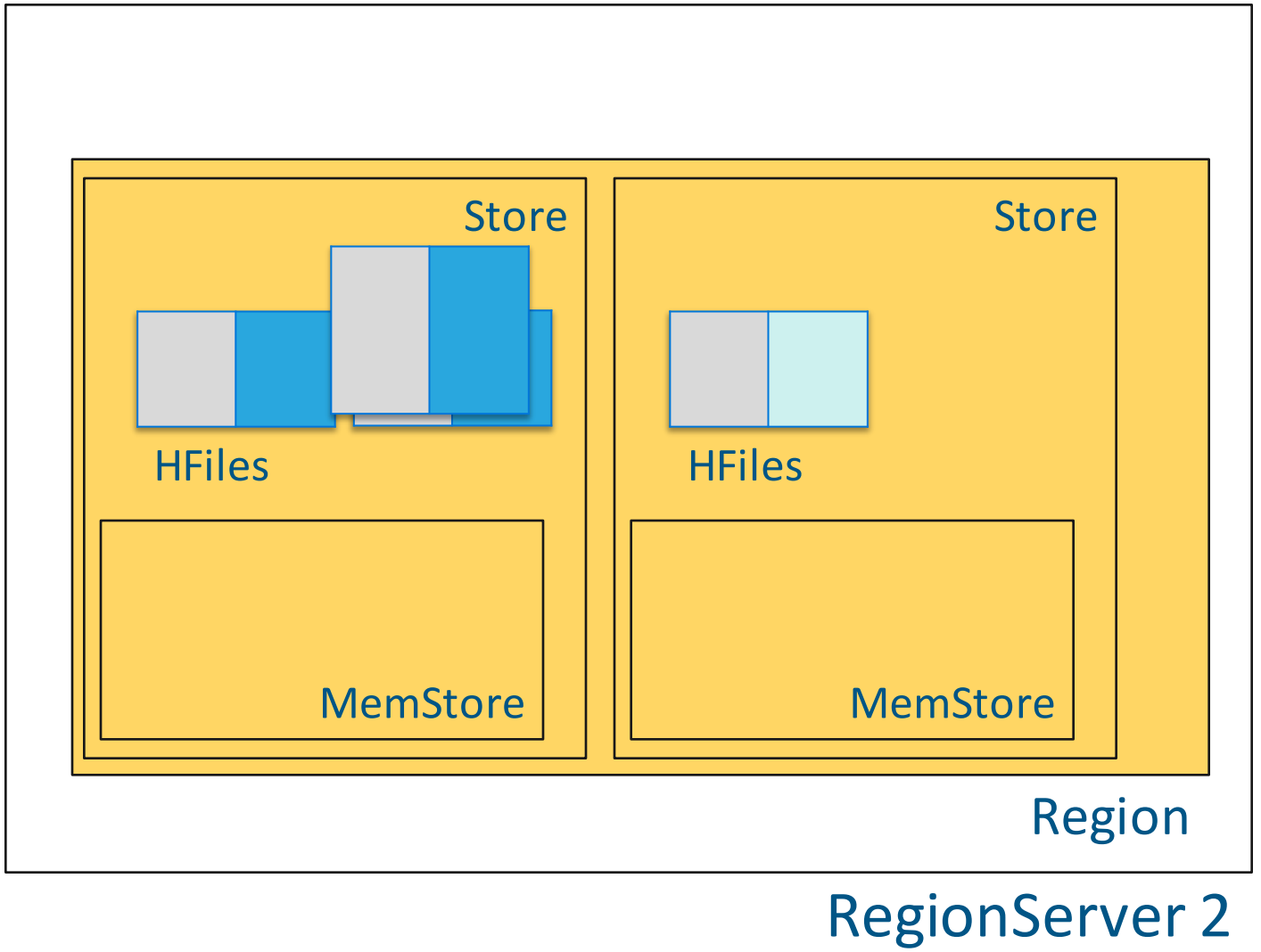
- Minor compactions
  - Merge some HFiles (in a given Store).
- Major compactions
  - Merge all HFiles (in a given Store).
  - Take care of other HBase housekeeping tasks.

HBase Architecture | Compaction



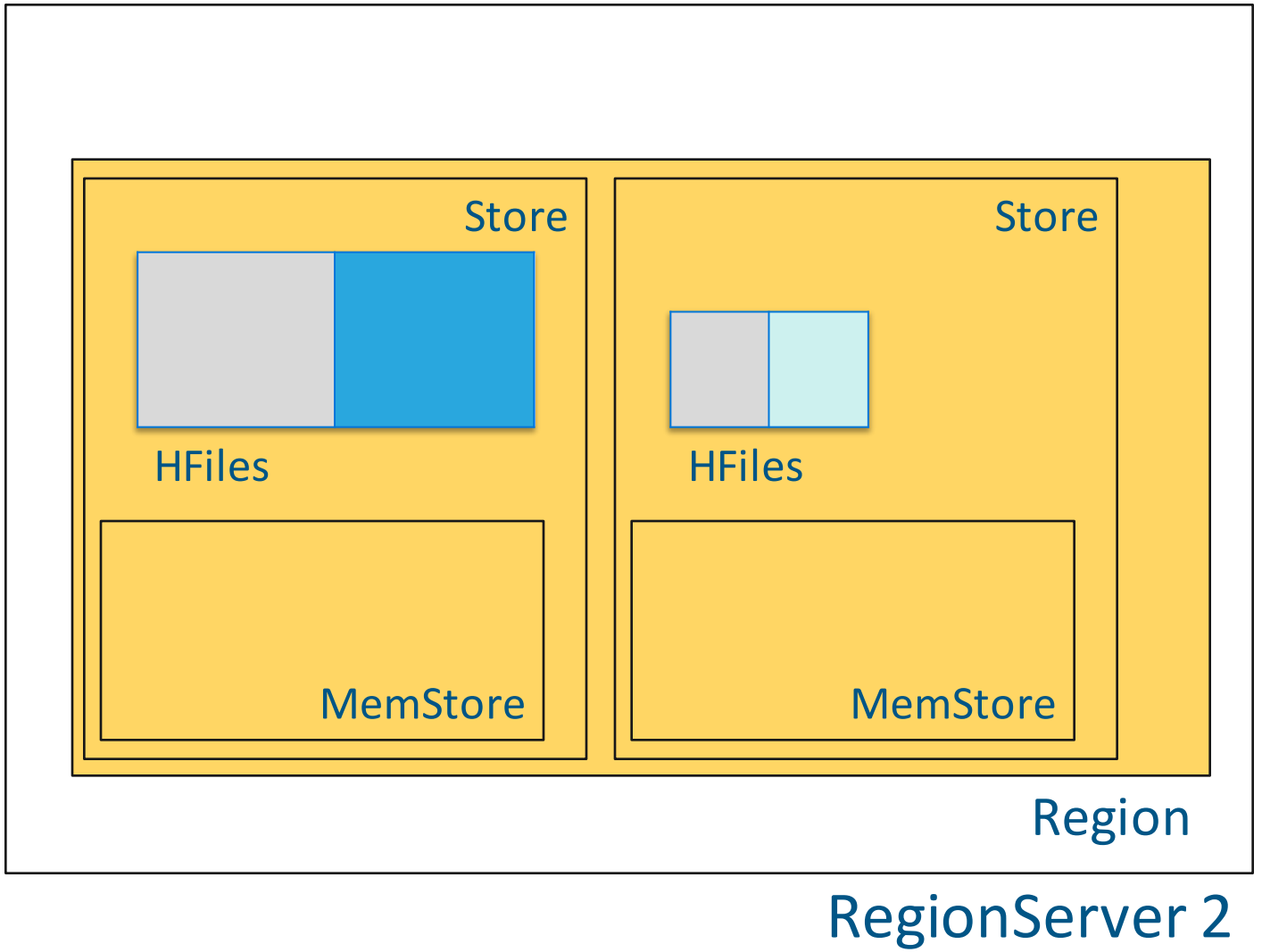
cloudera

HBase Architecture | Minor compaction



cloudera

HBase Architecture | Major compaction



cloudera

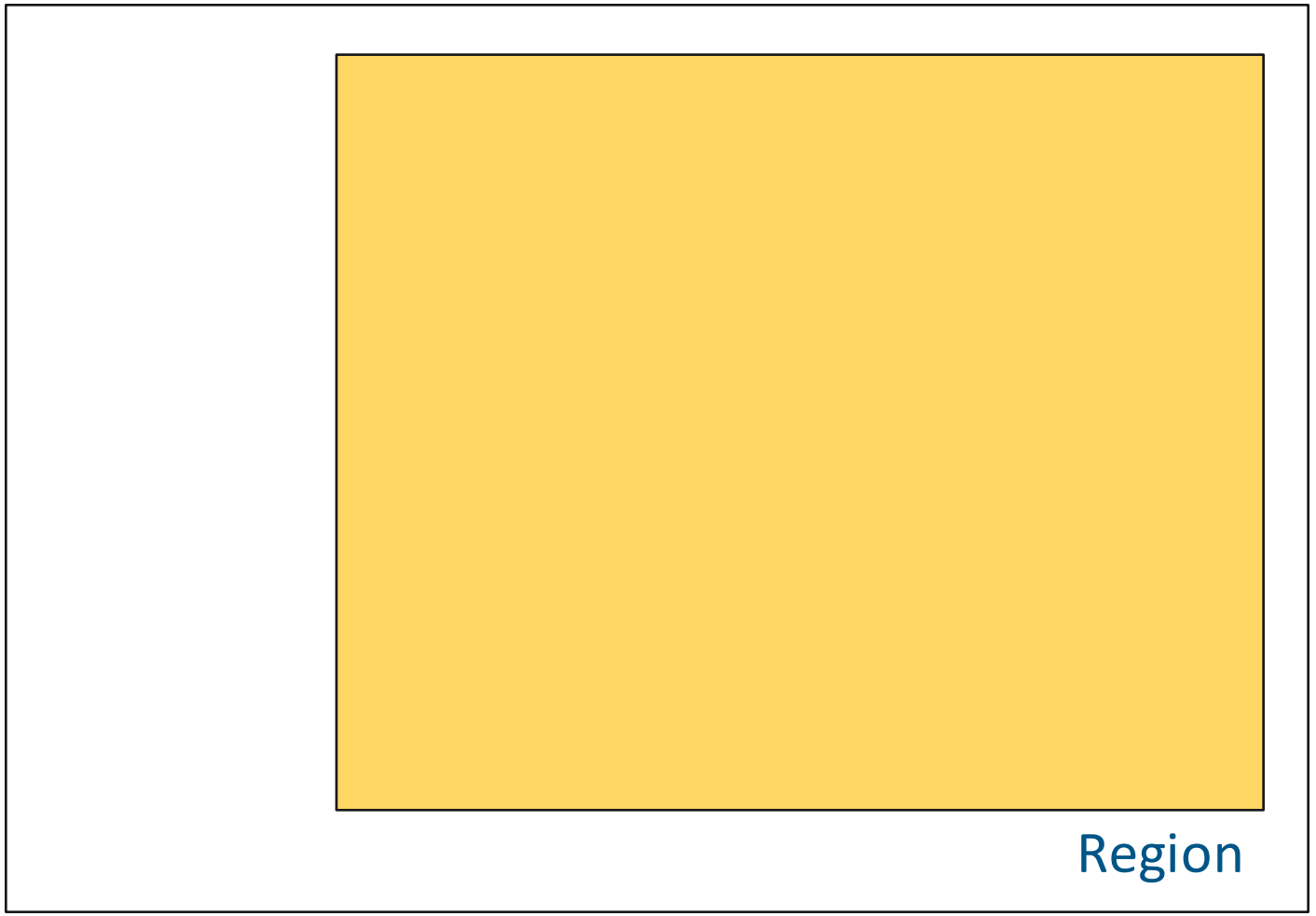
HBase Architecture | Compactions

- Minor compactions
  - Controlled by policy (pluggable).
- Major compactions
  - Automatic (by time) or manually triggered.
  - Tend to be run during off-peak times.

HBase Architecture | Splits

- Eventually, Regions become imbalanced.
  - Some grow to be huge, others remain small.
  - Leads to disparate load across RegionServers.
- In these cases, HBase can split a Region into two.
- Each Region is then available to be moved to a different RegionServer, if necessary.

HBase Architecture | Splits



cloudera

HBase Architecture | Splits

RegionServer 3: Yeah! Pick me!

Master: RegionServer 2 is really busy… Maybe another RegionServer can handle one of its Regions?

cloudera

HBase Architecture | APIs

- Conventional write path can be accessed through multiple APIs:
  - Java API
    - Most full-featured.
  - REST API
    - Easily accessible.
  - Thrift API
    - Support for many languages (e.g. C, C++, Perl, Ruby, Python).

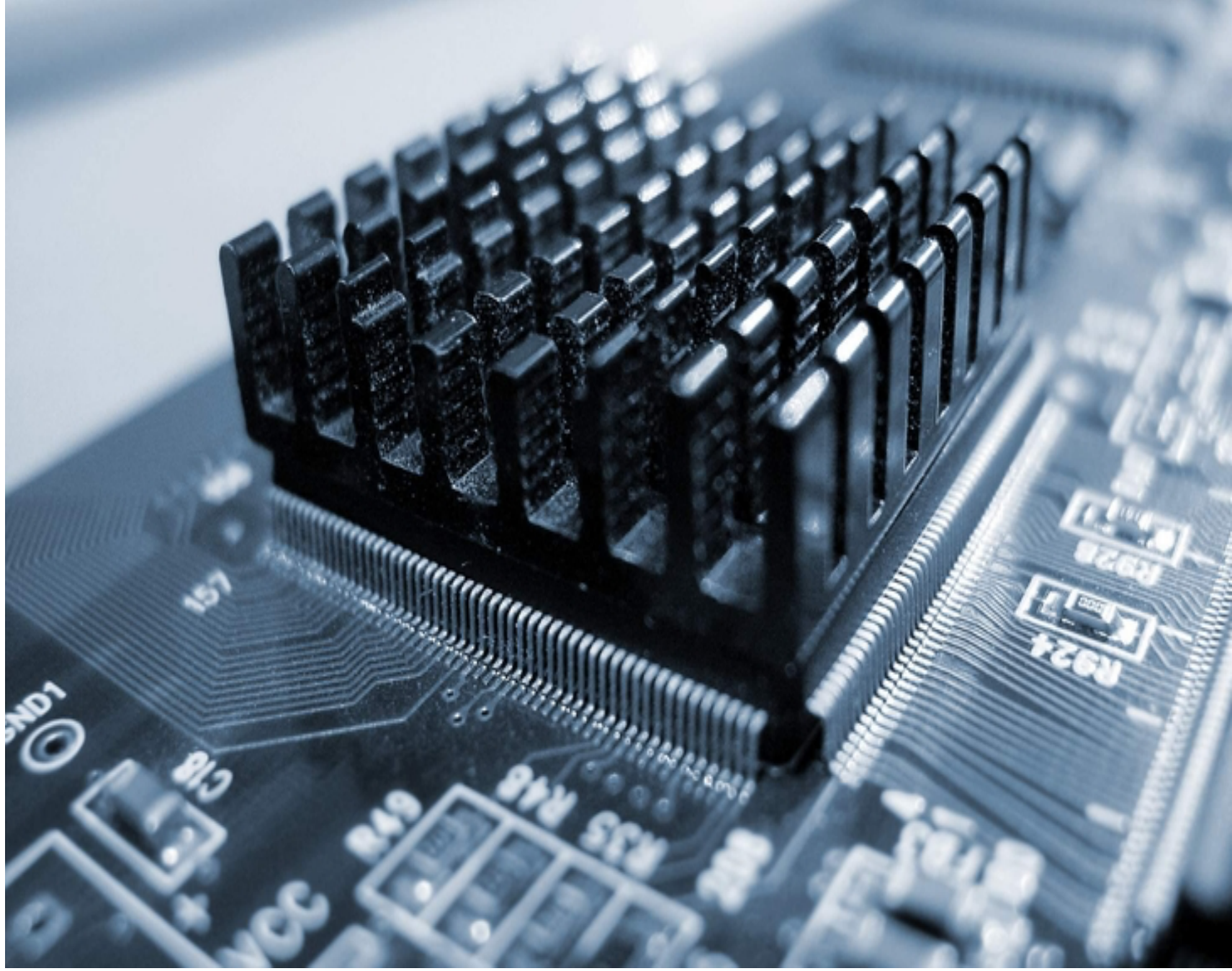
HBase Architecture | APIs

- This write path is durable, but if you're importing a lot of data, it can be problematic…
  - Every put goes into WAL, which means disk seeks. Lots of puts mean lots of disk seeks.
  - Lots of data into MemStores means lots of flushing to disk.
  - Lots of flushing to disk might mean lots of compactions.

HBase Architecture | Bulk Loading

- Bypass conventional write path.
  - Extract data from source.
  - Transform data into HFiles (done with MapReduce job) directly.
  - Tell RegionServers to serve these HFiles.

Enough of Architecture



cloudera

What's up next, Doc?

- Break
- What have we learned from the users
- How can **you** benefit from that information

cloudera

cloudera

Break

Apache HBase "Nascar" Slide

Apache HBase “Nascar” Slide

Apache HBase “Nascar” Slide

Apache HBase "Nascar" Slide

Apache HBase "Nascar" Slide

Apache HBase "Nascar" Slide

Apache HBase "Nascar" Slide

cloudera®

What have we learned from all these 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

Use of data

- Operational
- Analytical

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

Entity centric data

Users

Accounts

Location

Clicks and Metrics

Sensor Data

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

cloudera

Time centric event data

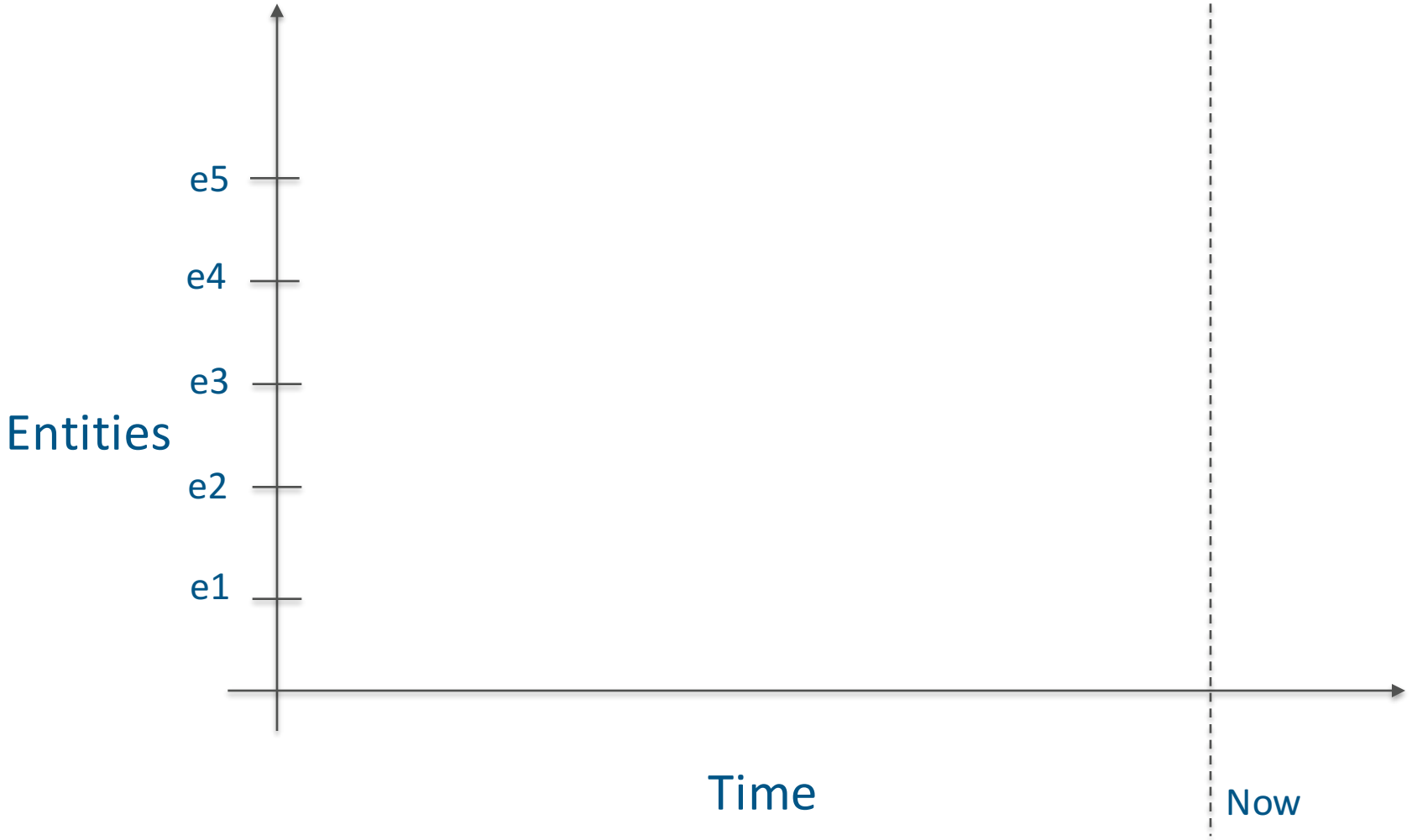
Stock Ticker Data

Monitoring applications

Periodic Sensor Data

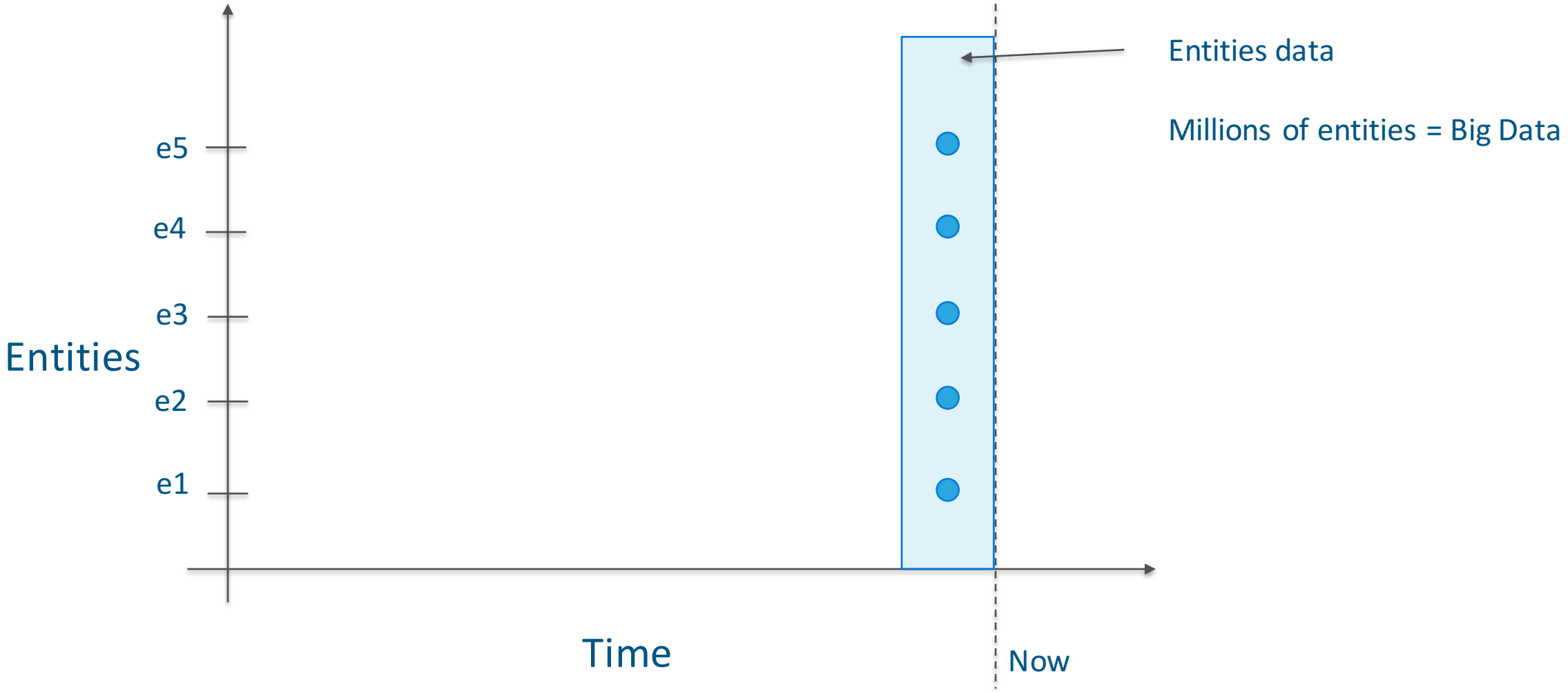
- Time-series data points over a period
- Scales up due to finer grained intervals, retention policies, and the passage of time

cloudera



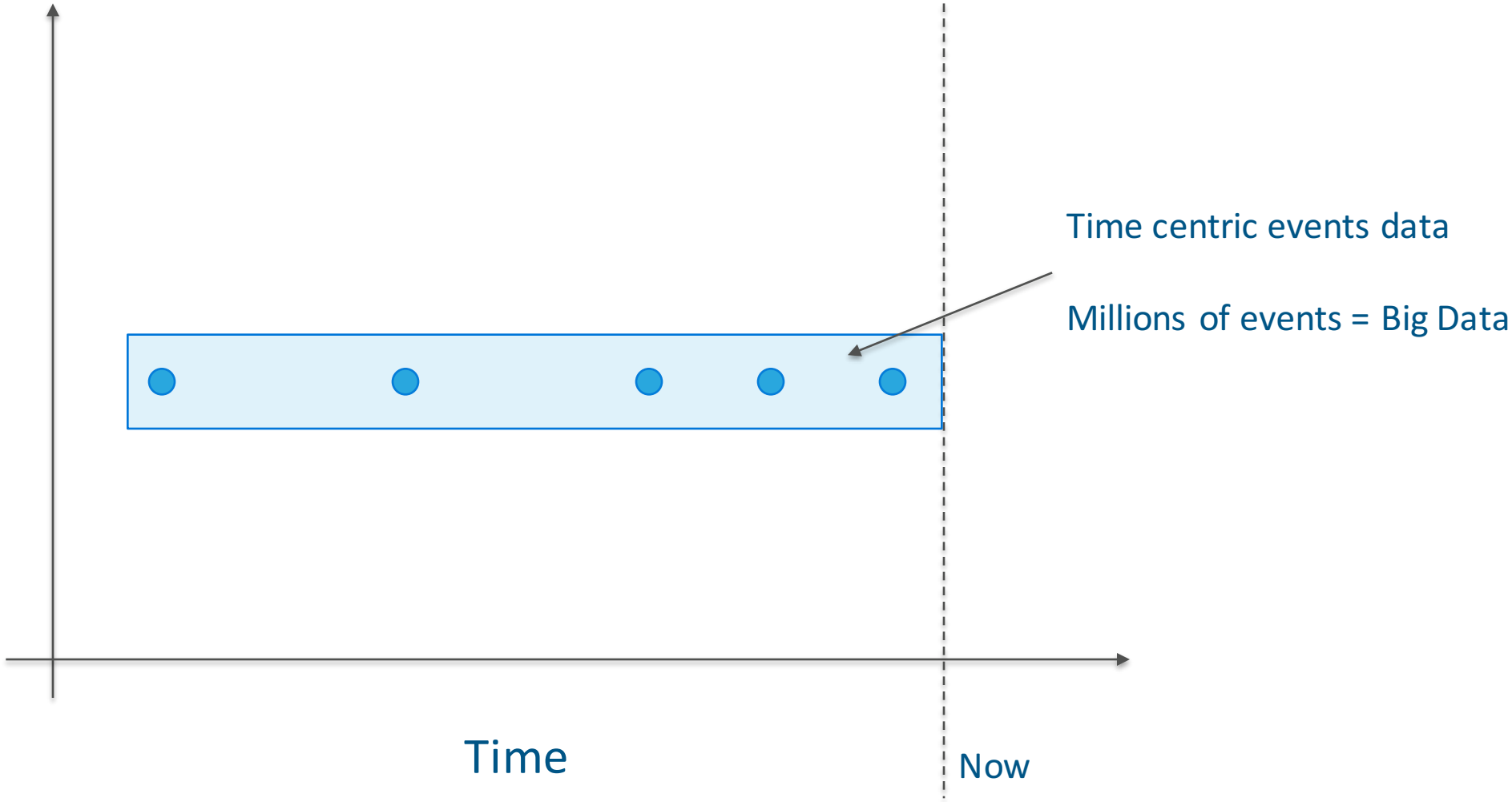
cloudera

Entities data



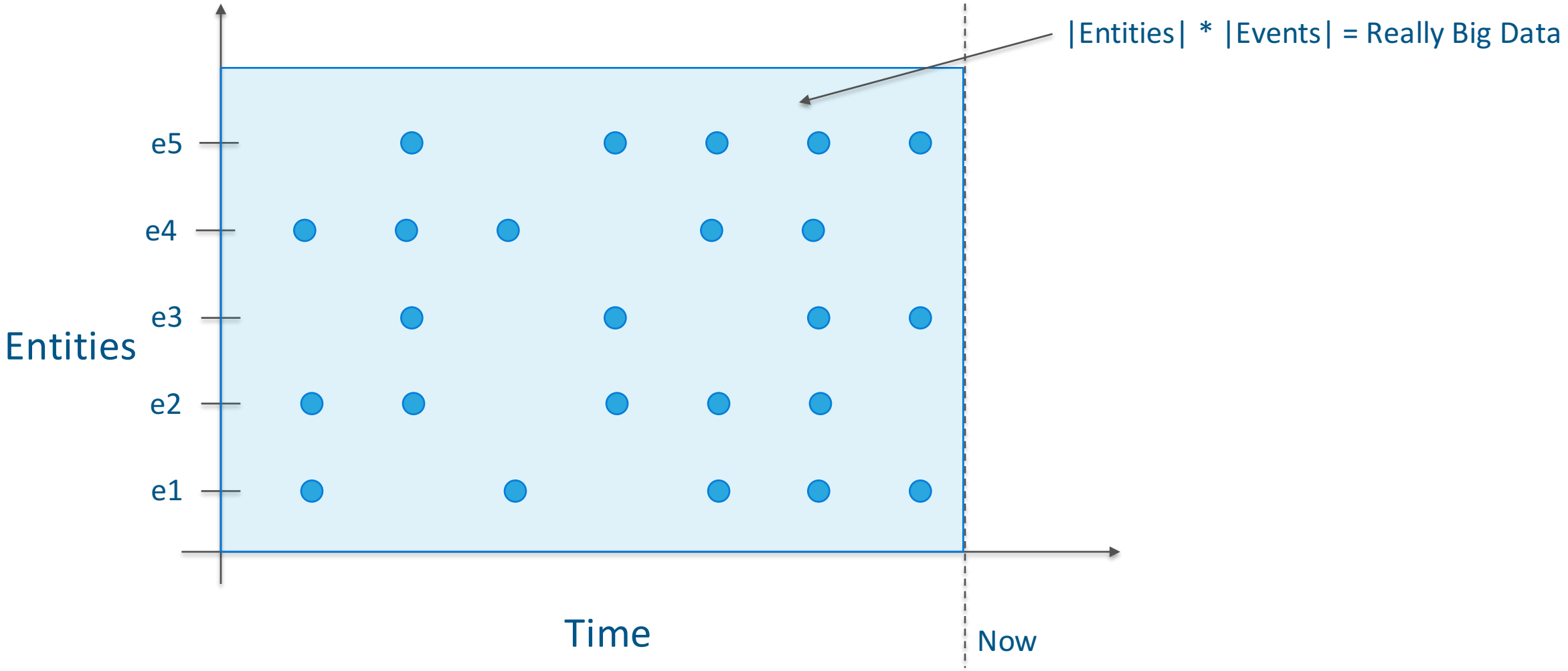
cloudera

Time-centric events data



cloudera

Time-centric events about Entities



cloudera

What questions do you ask?

- Do you focus in on entity first?

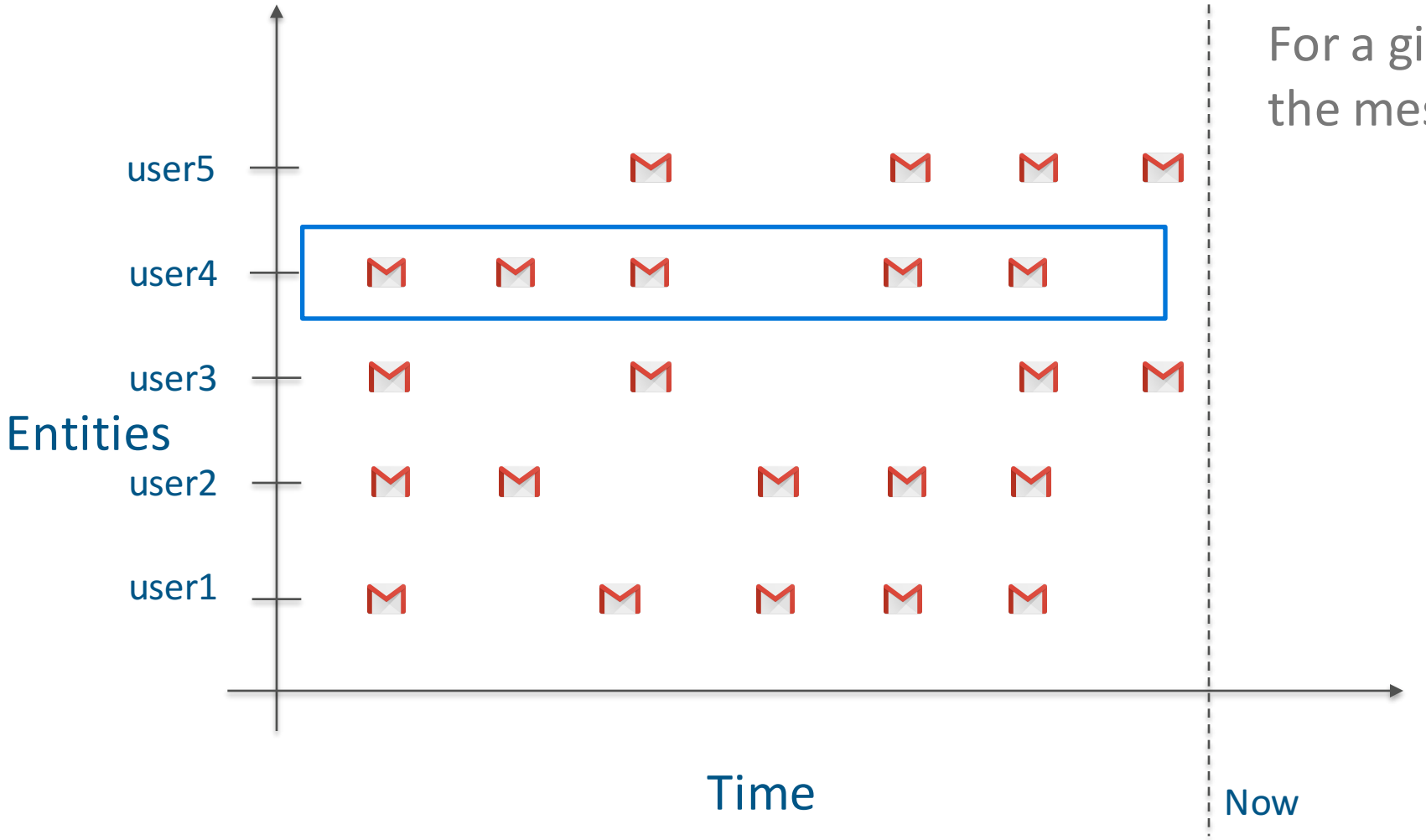
OR

- Do you focus in on time ranges first?

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

cloudera

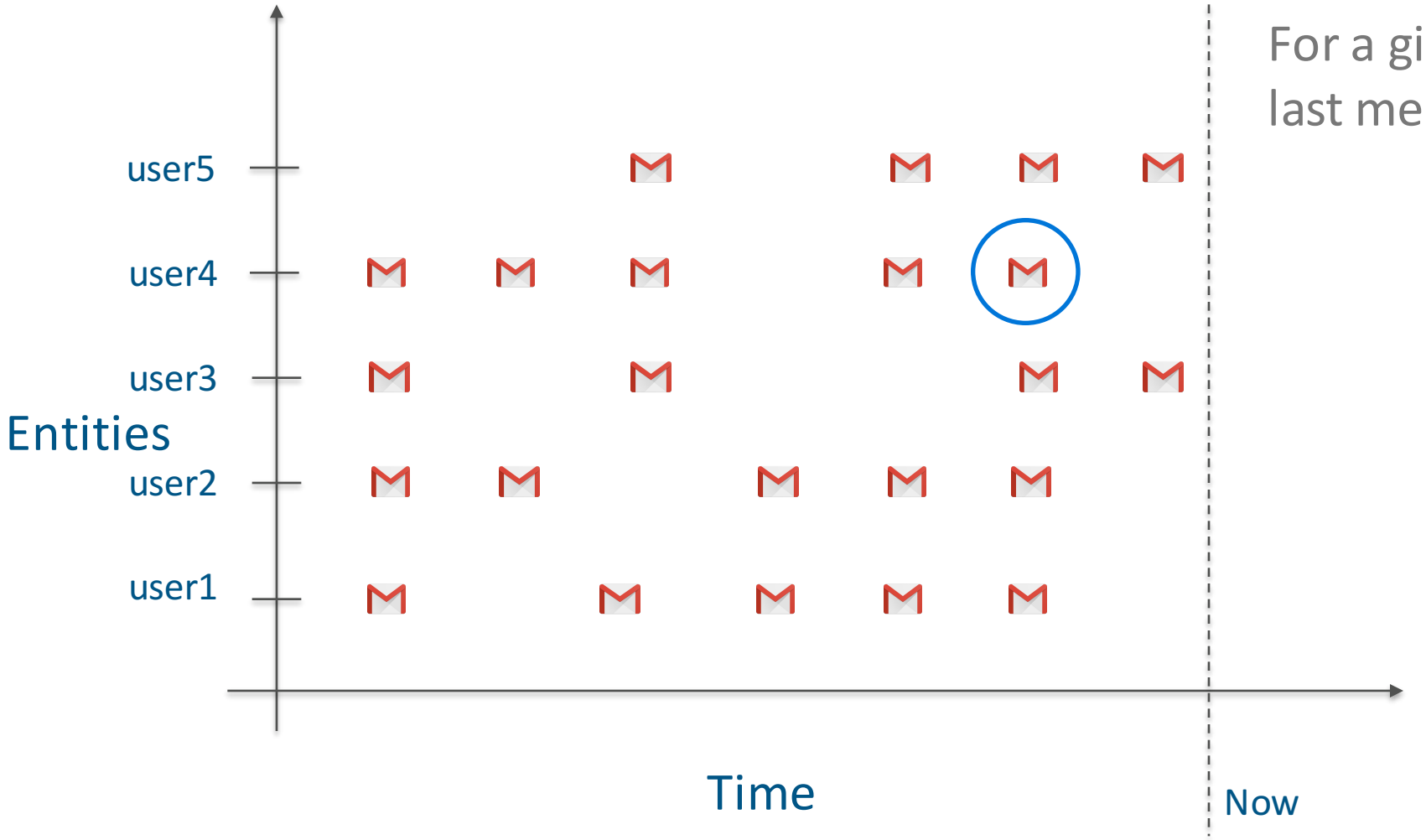
Entity first questions…



For a give user, show all the messages.

cloudera

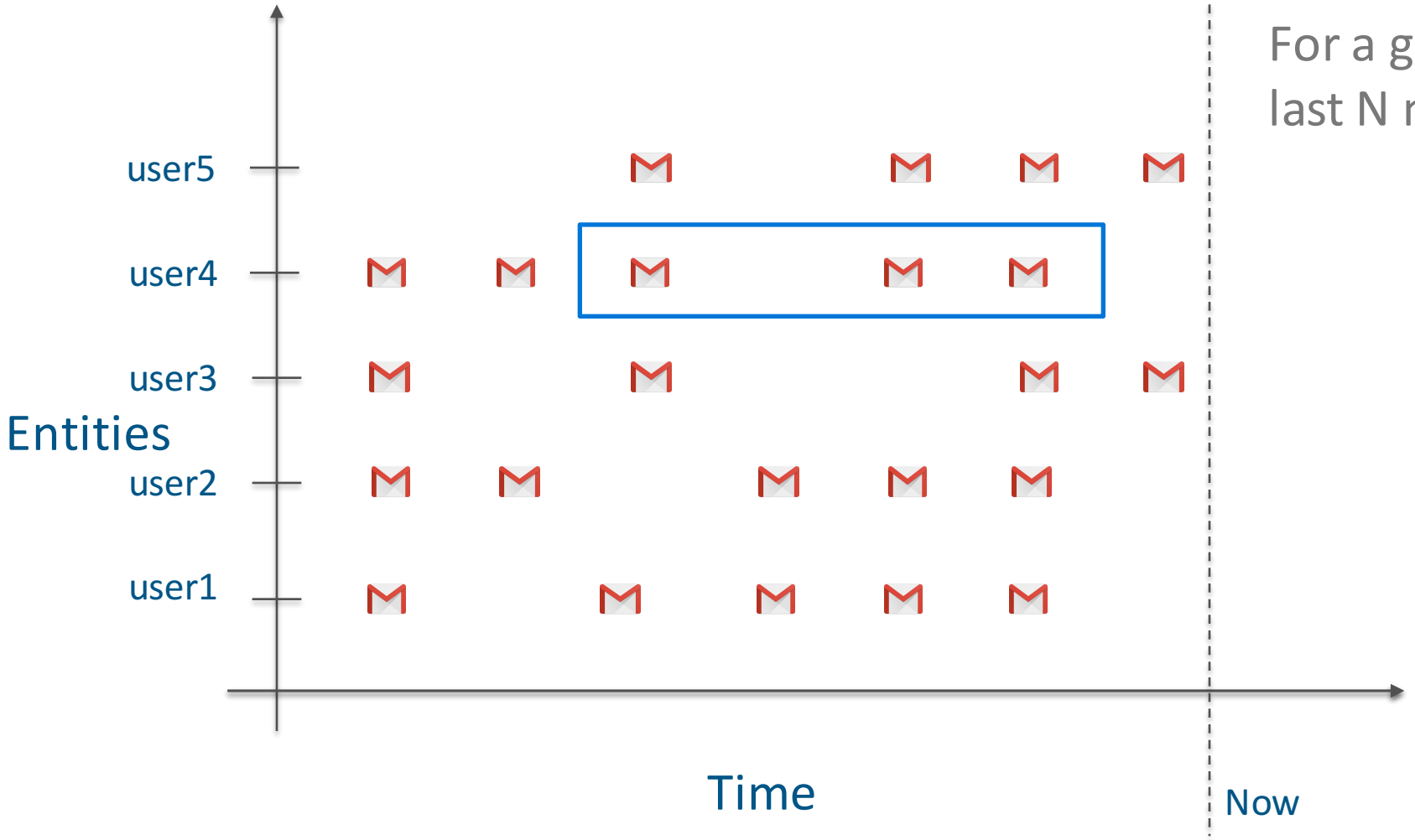
Entity first questions…



For a given user, show the last message.

cloudera

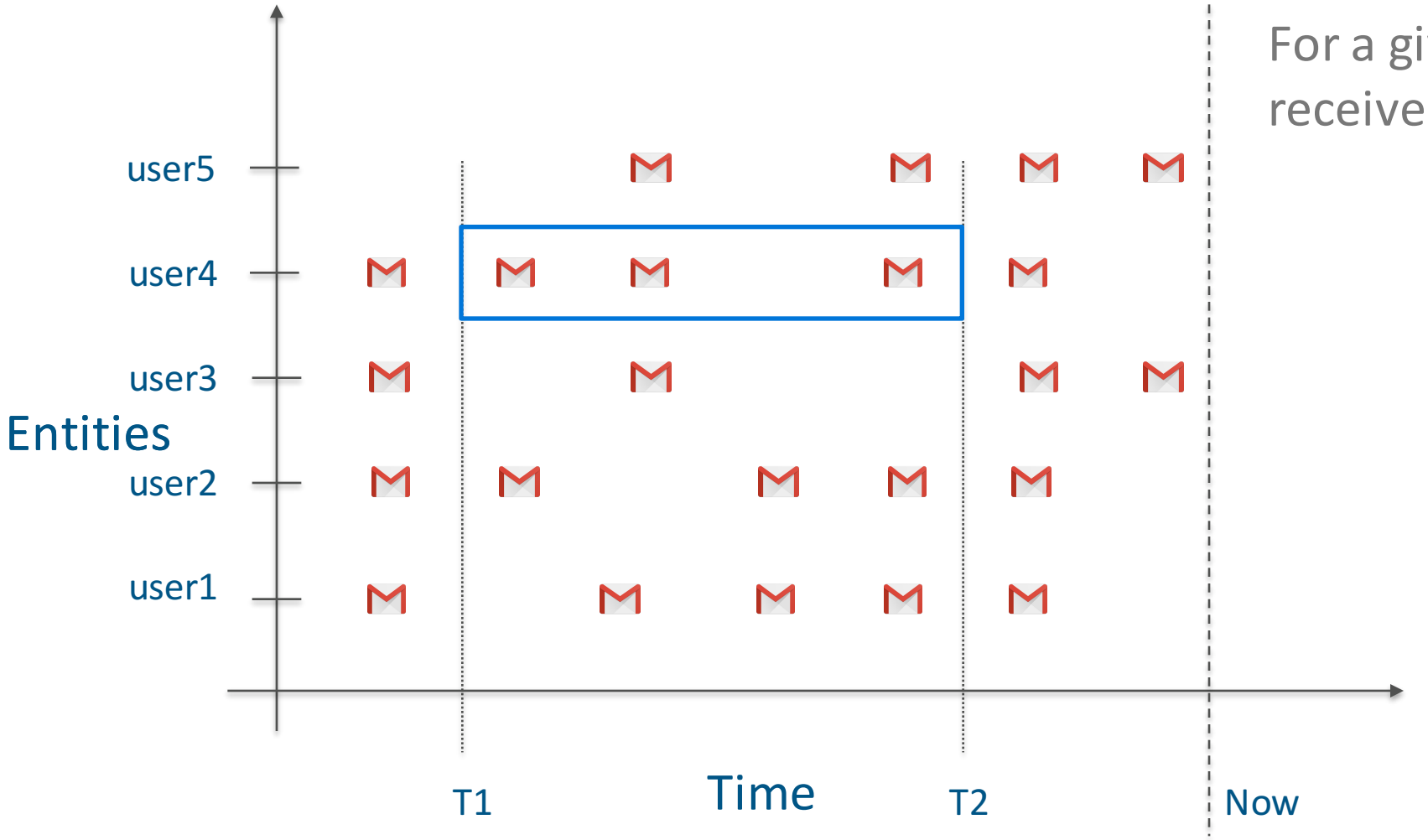
Entity first questions…



For a give user, show last N messages.

cloudera

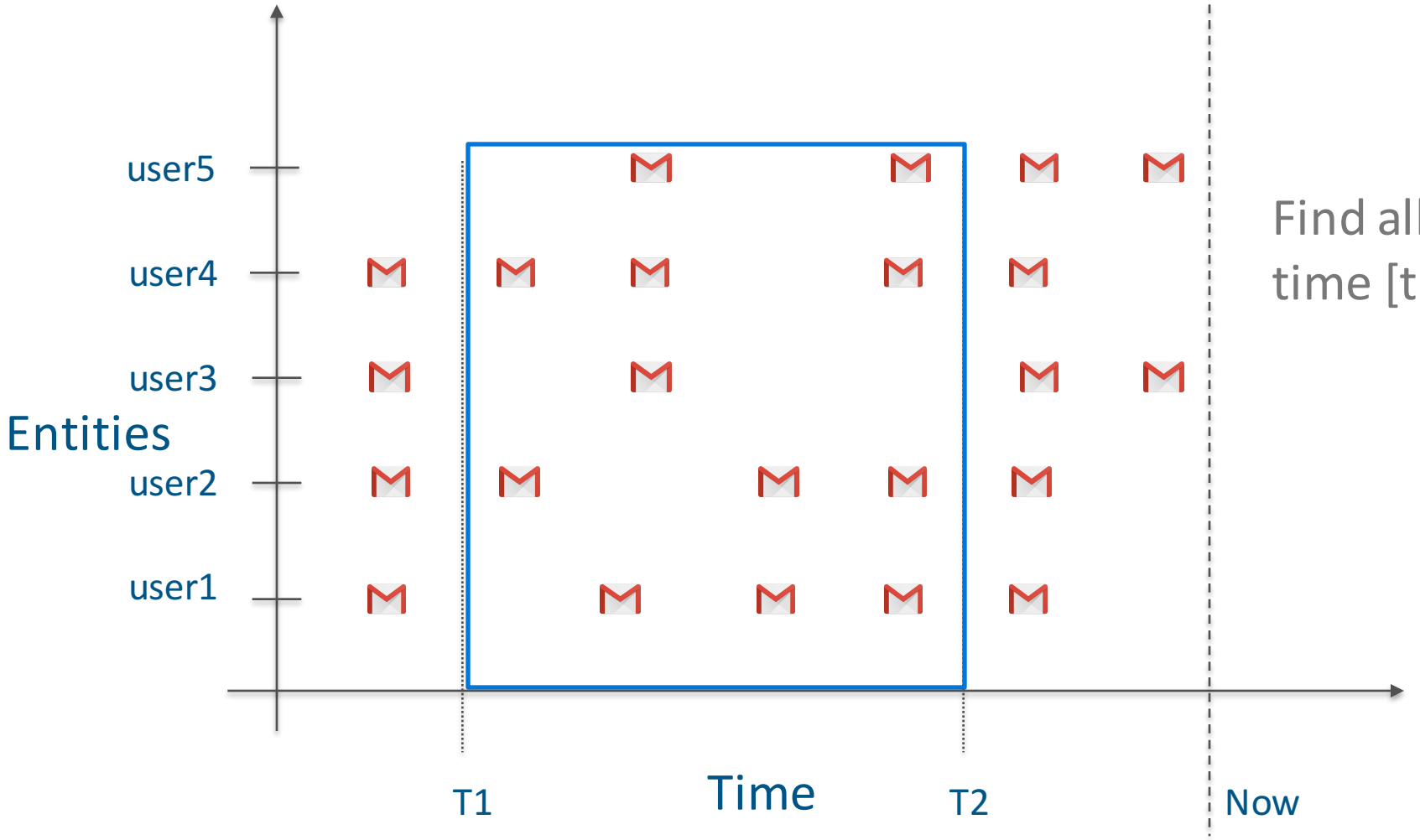
Entity first questions…



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

cloudera

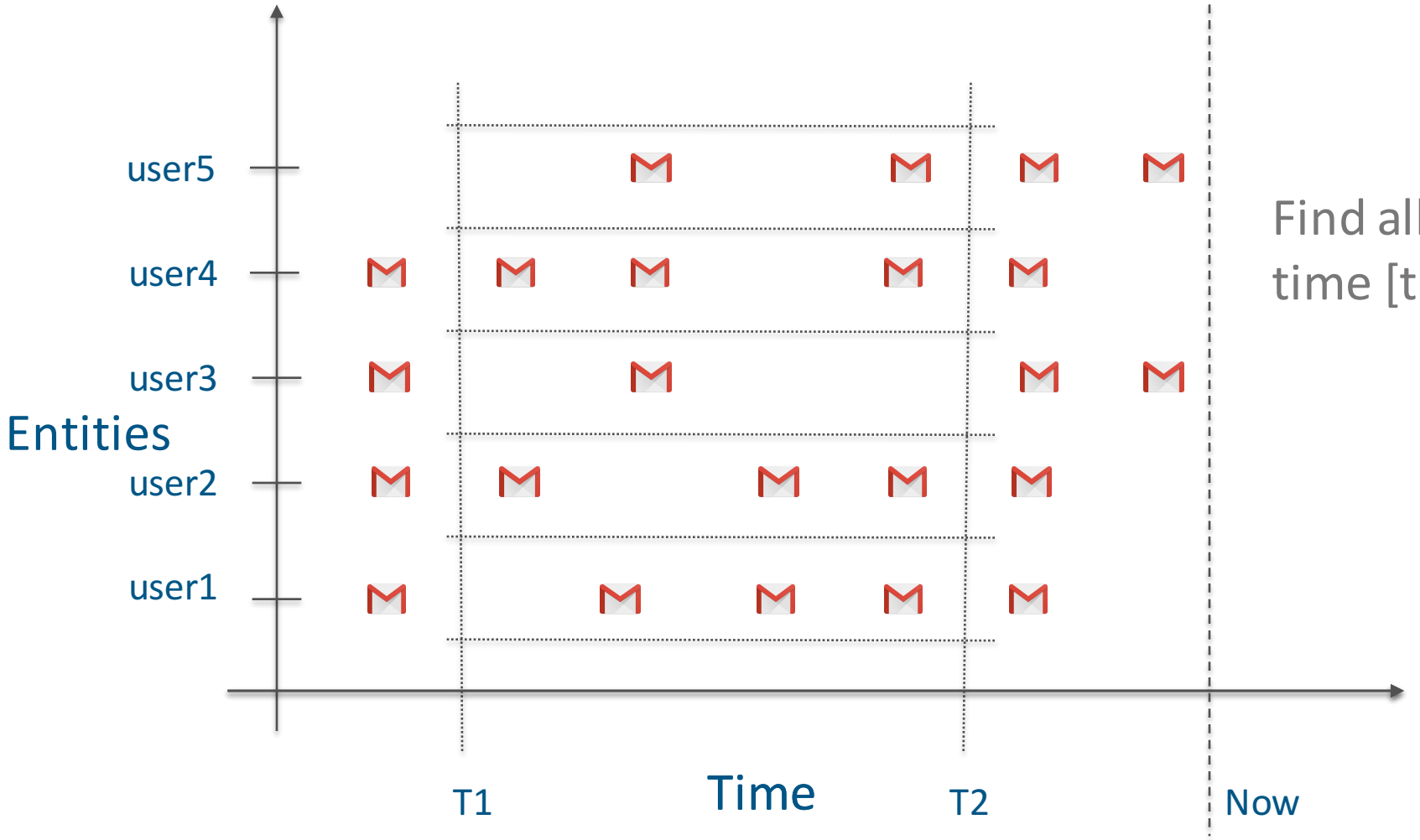
Time centric event first questions…



Find all messages between time [t1, t2].

cloudera

Time centric event first questions…



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

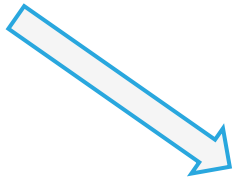
cloudera

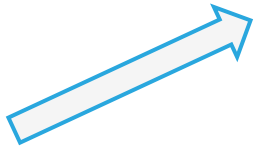
cloudera®

How does the data get in and out of HBase?

Getting data in...

Put, Incr, Append

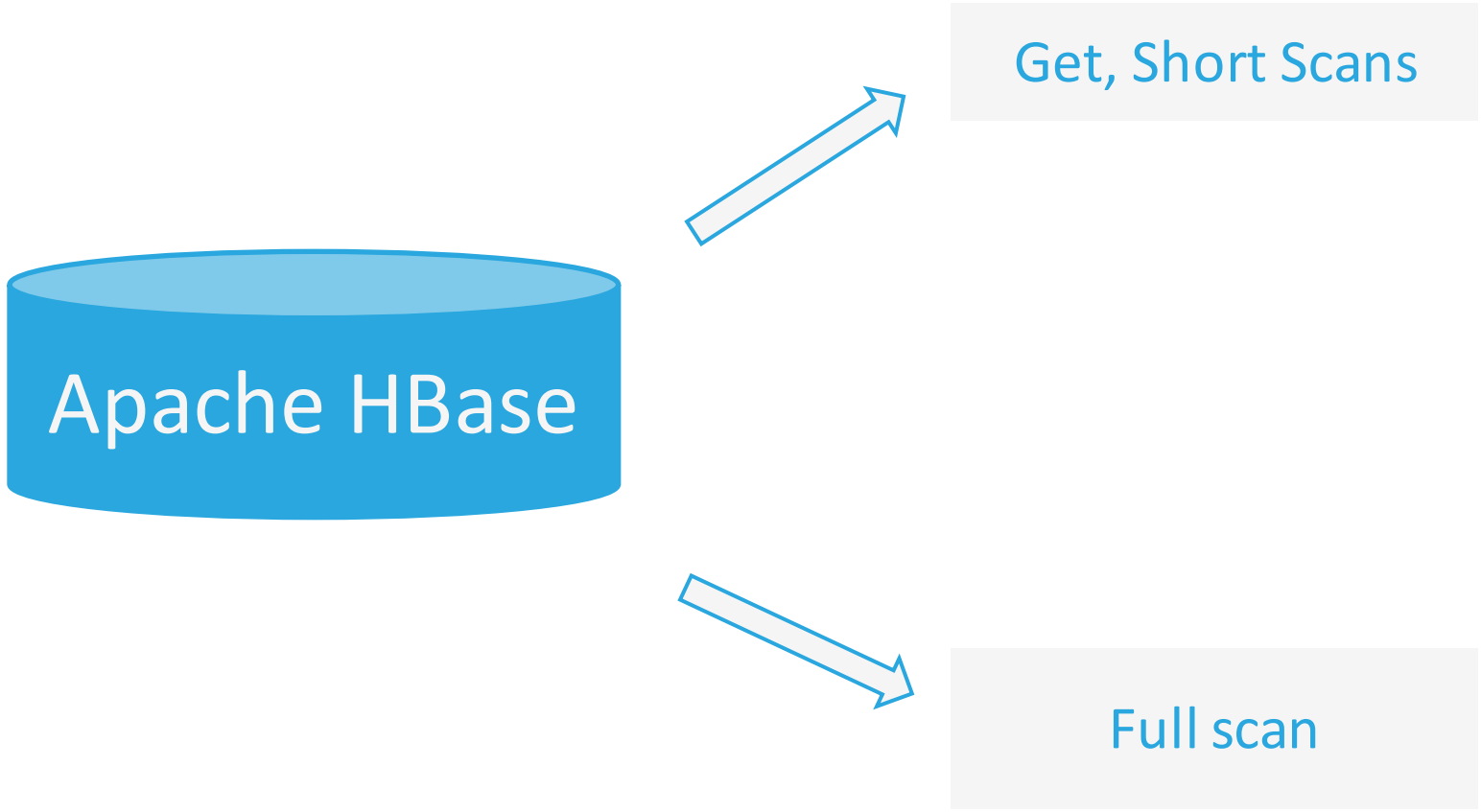




Bulk Import

cloudera

Getting data out...



So, what's the best way?

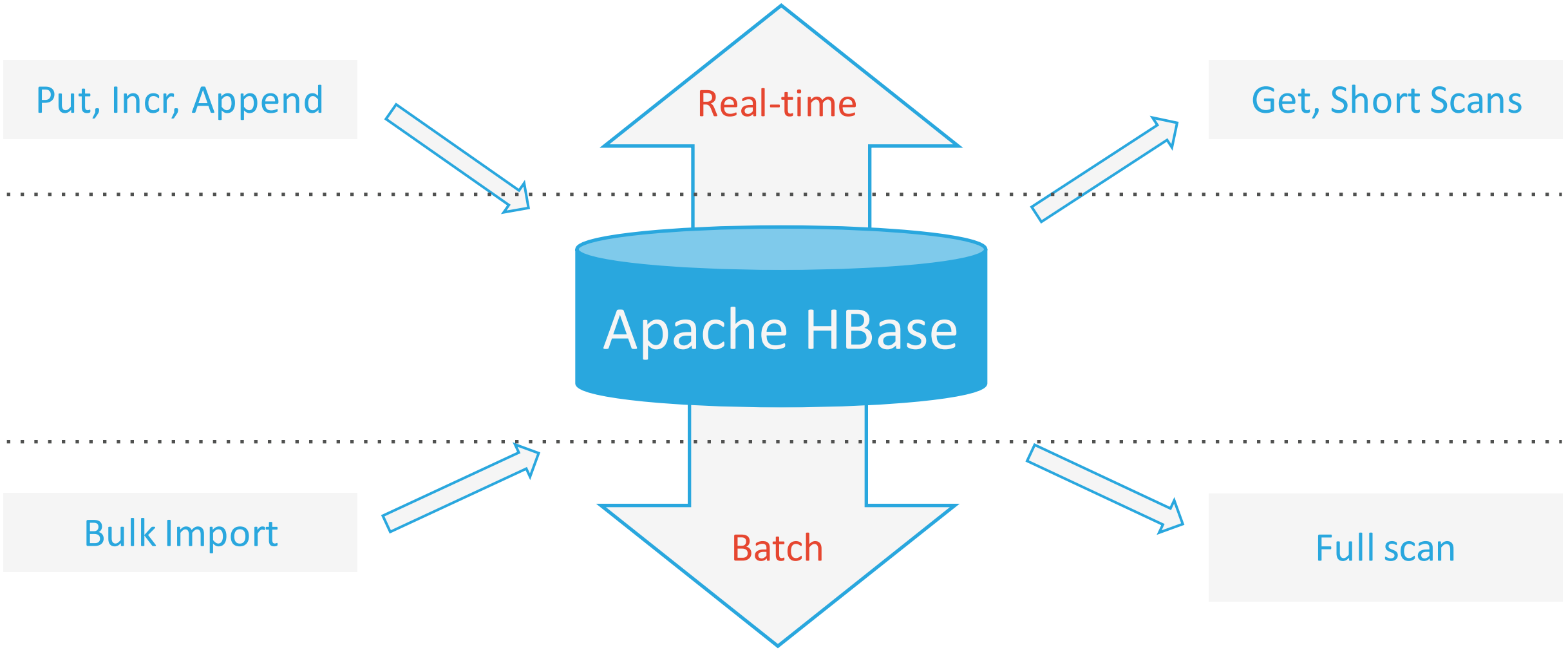
cloudera

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



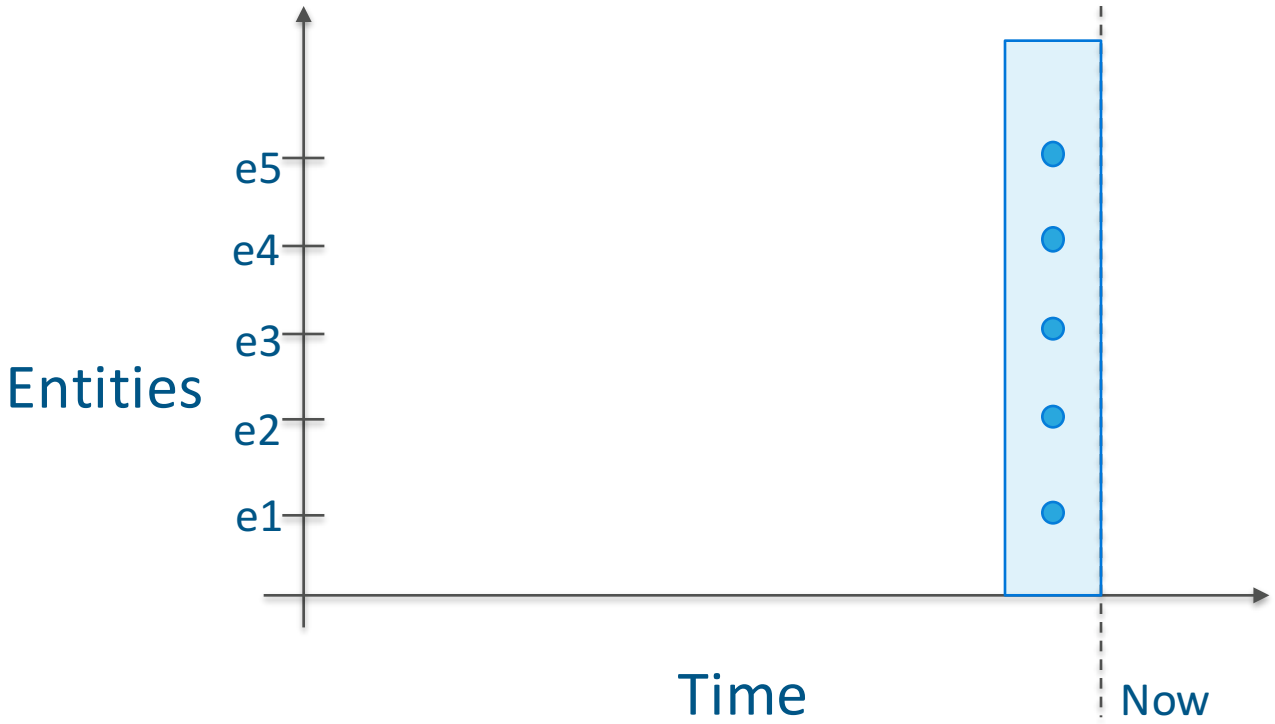
cloudera

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 port



cloudera

Simple Entities : Schema

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

cloudera

Simple Entities : Example

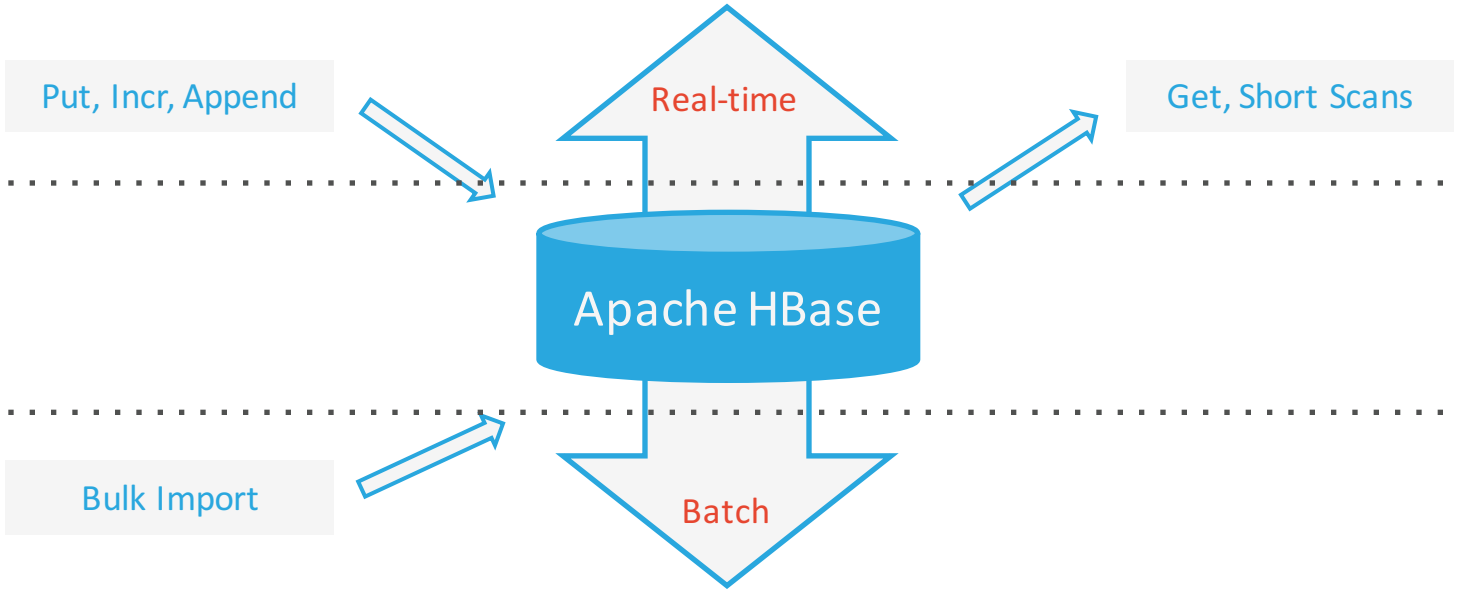
OCLC : Online Computer Library Center 

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

Simple Entities : Access Pattern

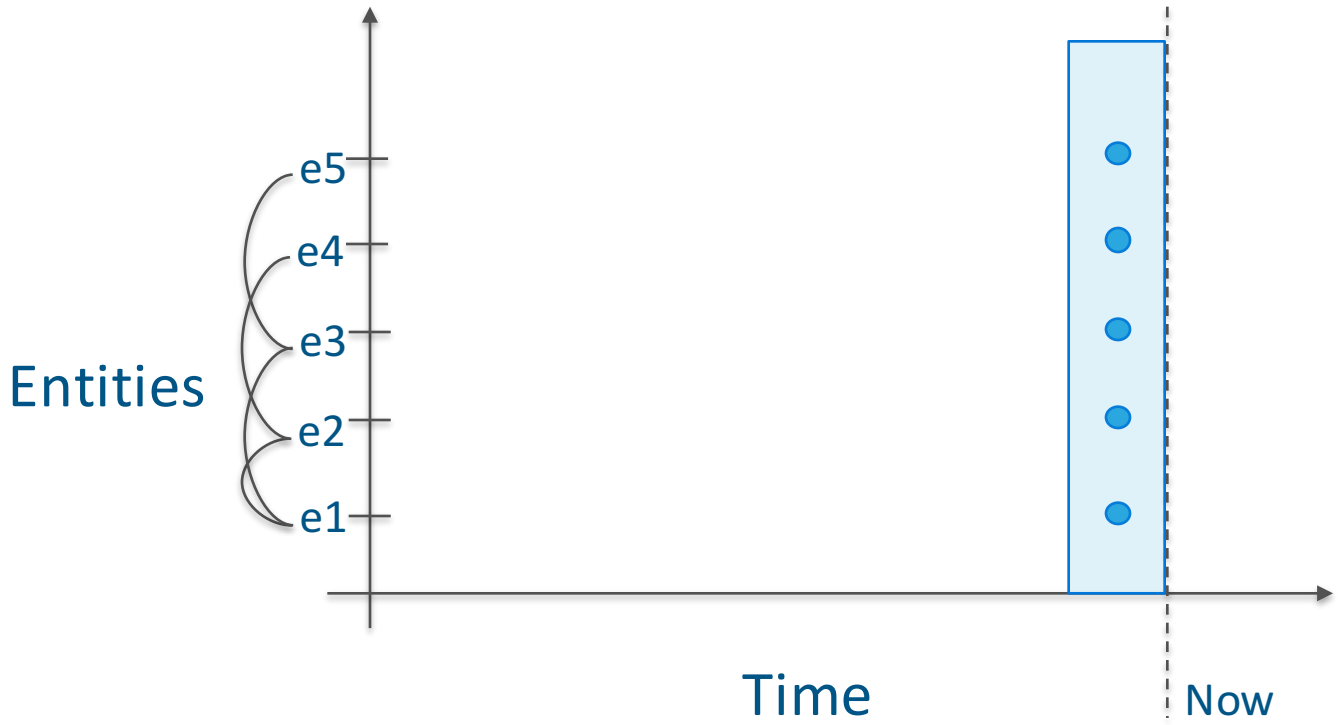
- Access Patterns
  - Writes : Batch / Real-time
  - Reads: Real-time



cloudera

Linked Entities (Graph Data)

- Entity are linked to form a graph



cloudera

Linked Entities (Graph Data) : Schema

- Row per Node (Entity)
- Row key => Node ID (Entity ID)
- Column => “Relationship:OtherNodeID”
- Value => Meta data about relationship

cloudera

Linked Entities (Graph Data) : Example

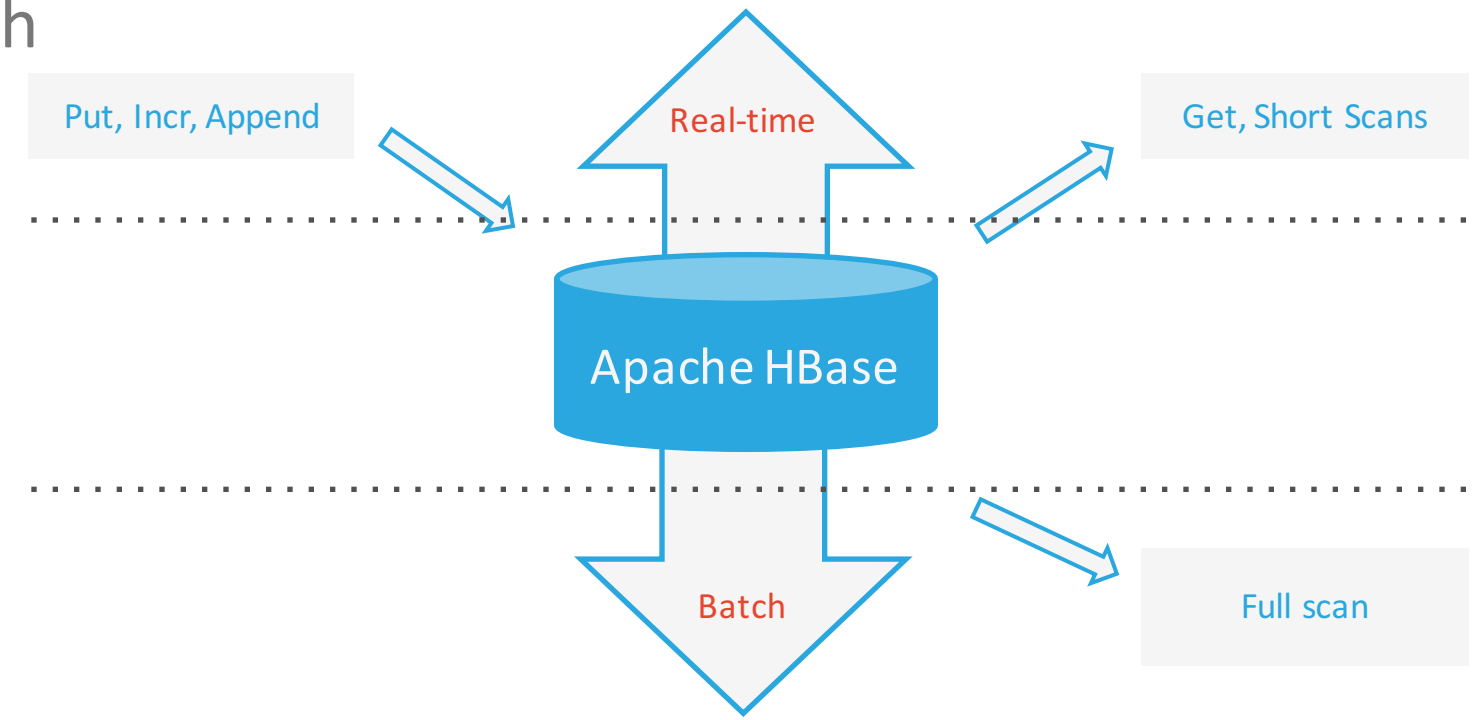
Social Network (Facebook)

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

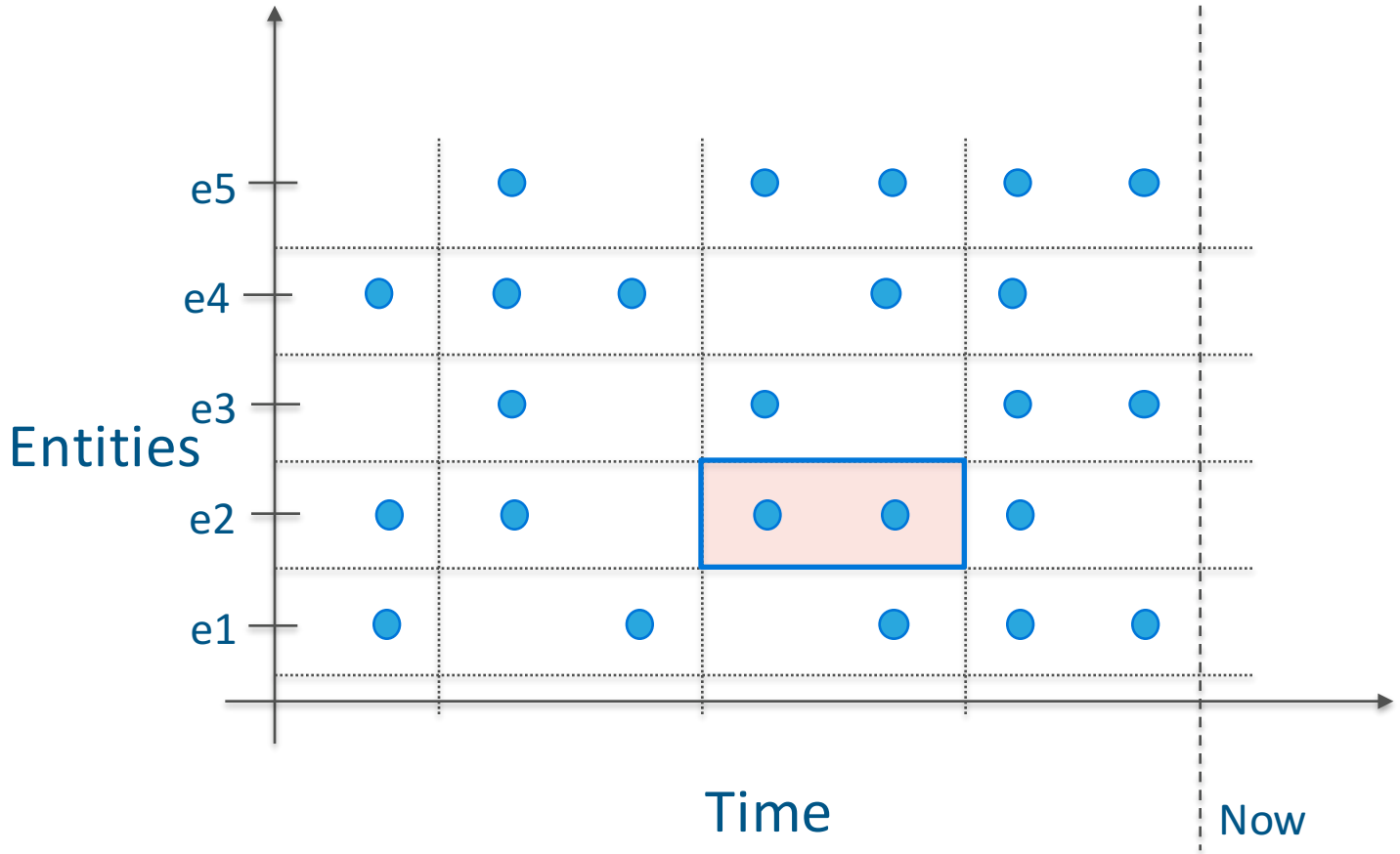
Linked Entities (Graph Data) : Access Pattern

- Access Patterns
  - Reads: Real-time or Batch
  - Writes: Real-time



Time-coupled entities

- Events about entities in time centric
- **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) →

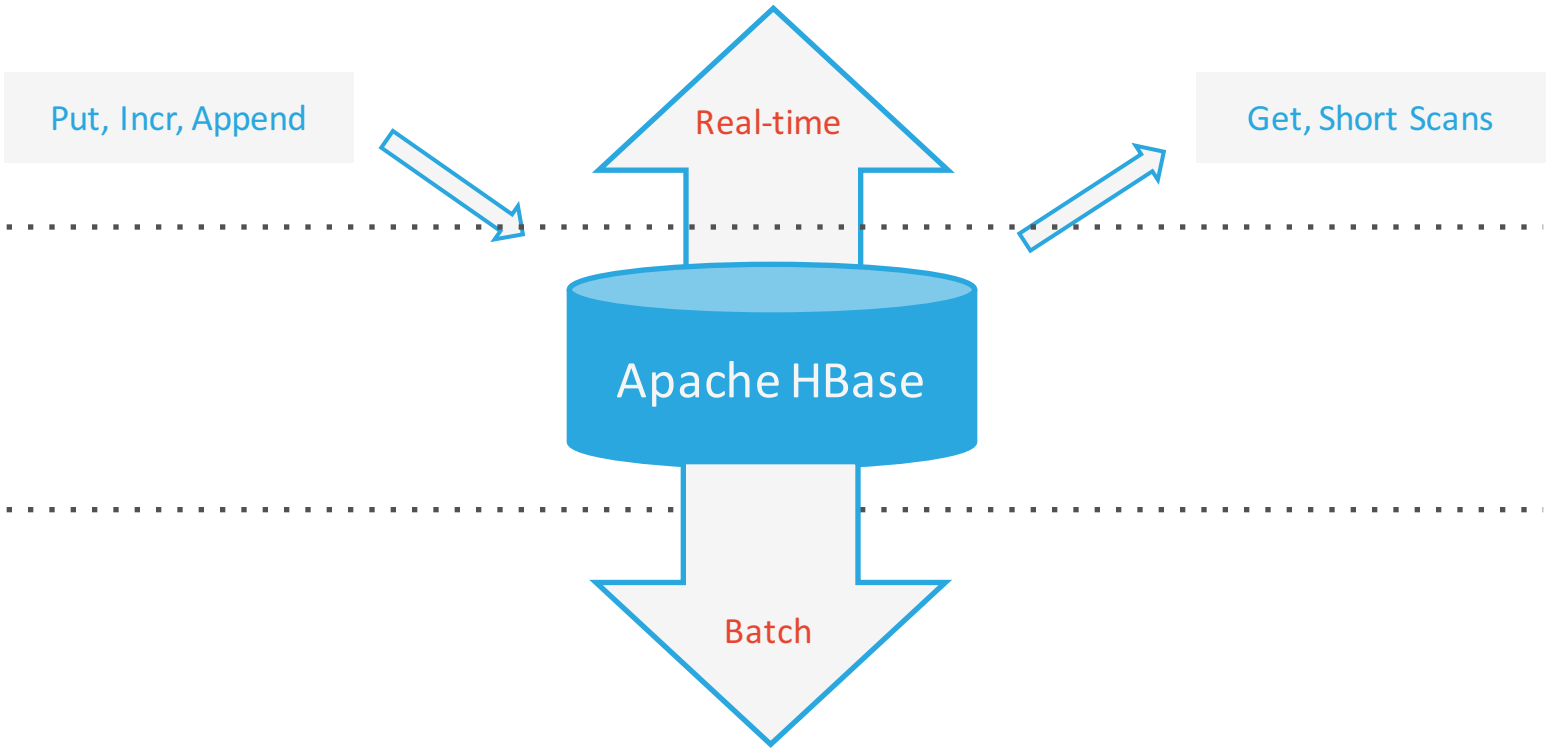
  Real-time write

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

cloudera

Time-coupled entities : Access Pattern

- Access Pattern
  - Writes: Real-time
  - Reads: Real-time



cloudera

HBase is great!

But not for everything ...

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
- Just because it is not good today, doesn't mean it can't be better tomorrow!

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
  - Row key = timestamp
  - Real time writes
  - Column-centric aggregations over all rows
- Schema
  - Row key: timestamp
  - Column qualifiers: properties with data or counters
- Example
  - Machine logs organized **by timestamp** (causes write hot-spotting)

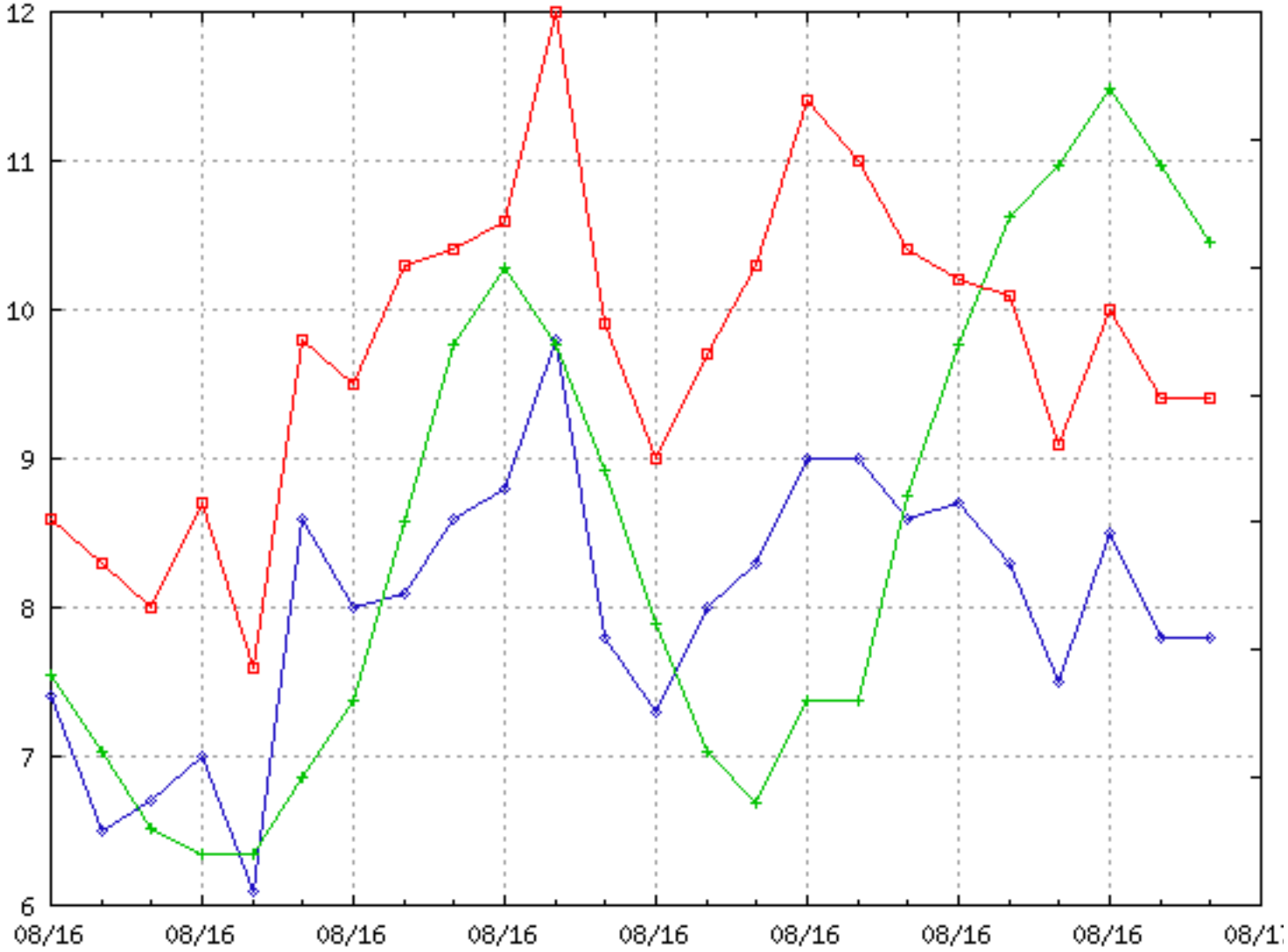
Summary

- HBase is used widely across industry
- Few patterns learnt from these users
- Understanding
  - Data : Entity and time-centric events
  - Questions you ask from your data
  - How does data gets in and out
- When not to use HBase

Scalable time series database

Time-Series

Data points for entities over time



OpenTSDB

- Store trillions of data points
- Millisecond precision
- Keep raw data forever
- Scales to millions of writes per sec
- Generate graphs from GUI

cloudera

OpenTSDB

- Store trillions of data points
- Millisecond precision
- Keep raw data forever
- Scales to millions of writes per sec
- Generate graphs from GUI

cloudera

OpenTSDB : Use Cases

- System Monitoring
  - Servers
  - Network
- Sensor Data
- Stock market data

cloudera

OpenTSDB : Example

OVH

- Large cloud/hosting provider
- Monitor everything: networking, temperature, voltage, application performance, resource utilization, customer-facing metrics, etc.
- 35 servers, 100k writes/s, 25TB raw data

Yahoo!

- Monitoring application performance and statistics
- 15 servers, 280k writes/s

Source: http://www.slideshare.net/HBaseCon/ecosystem-session-6

OpenTSDB : Datapoints

- In OpenTSDB, there are
  - Metric
  - Timestamp
  - Value
  - Tags (key-value pairs) : to identify the entity

OpenTSDB : Datapoints example

- E.g. 10 servers handling requests web requets
- Metric: num_requests_per_second
- Tags: “host=web-server-1”, “host=web-server-2”, and so on

- Example data points
  - num_requests_per_second 1439828251 50 host=web-server-1
  - num_requests_per_second 1439828251 72 host=web-server-2
  - num_requests_per_second 1439828252 30 host=web-server-3
  - …so on

cloudera

OpenTSDB : How it works

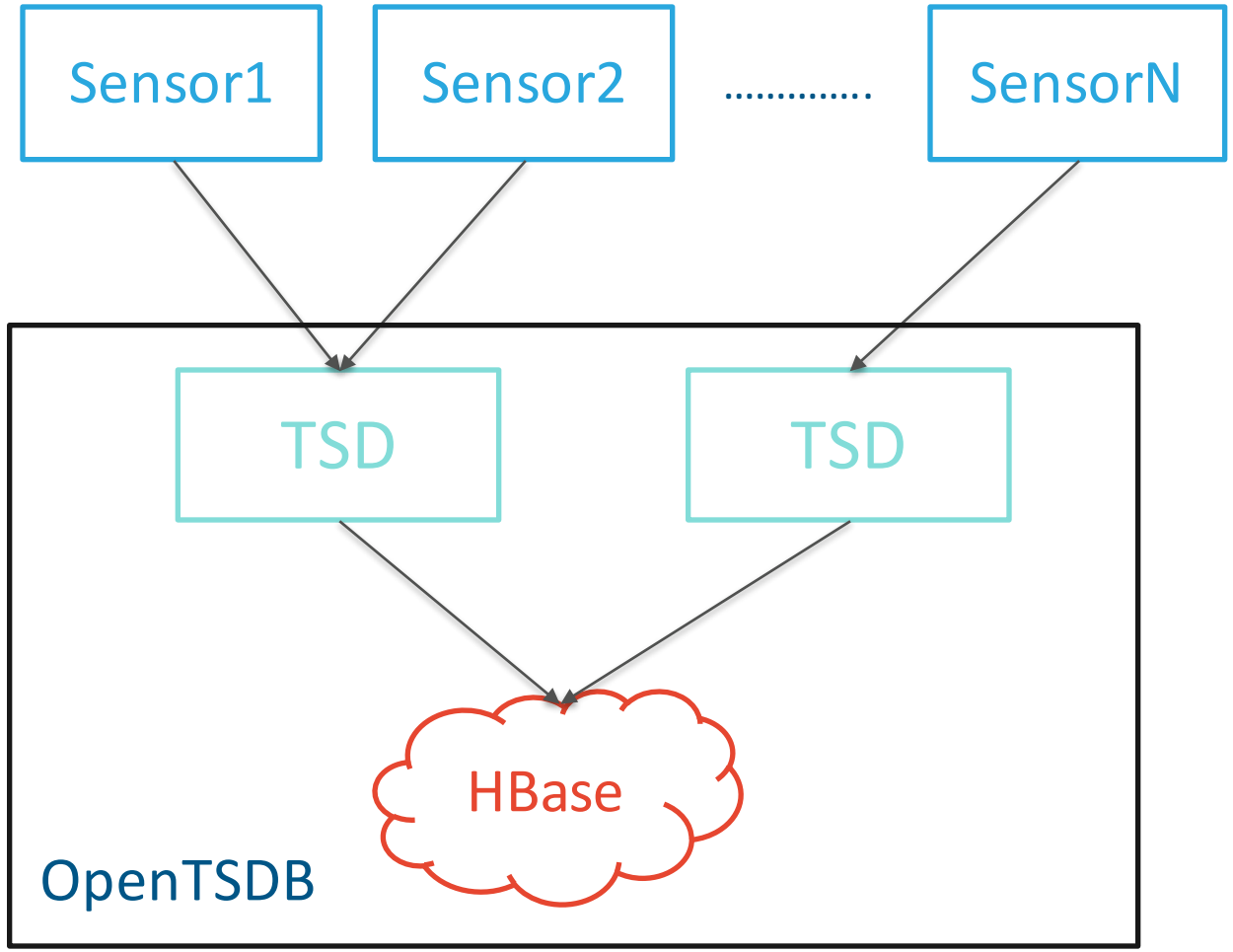


Image source: http://opentsdb.net/overview.html

cloudera

OpenTSDB : Writing data

- Telnet
  - put <metric> <timestamp> <value> <tagk1=tagv1[tagk2=tagv2 ...tagkN=tagvN]>
  - Example: put num_requests_per_second 1439828251 50 host=web-server-1

- HTTP API
  - <host>:<port>/api/put
  - JSON objects containing data points

- Bulk Import
  - Using ‘import’ CLI utility

cloudera

OpenTSDB : Reading data

- OpenTSDB GUI
  - Select metrics and tags to generate graphs

- HTTP API
  - <host>:<port>/api/query

cloudera

OpenTSDB : Storing data – row key

- Row key is a concatenation of metric, timestamp and tags
  - num_requests_per_second1439827200host=web-server-1

- Since data is stored in sorted order, chunking happens in this order
  1. Metric
     - Enables fast scan of all time series for a metric
  2. Time
     - Normalized on 1 hour boundaries
     - All data points for an hour are stored in a single row
  3. Tags

cloudera

OpenTSDB : Storing data – column

- Offset from timestamp in row key
- Example
  - num_requests_per_second 1439828251 50 host=web-server-1
  - num_requests_per_second 1439828251 72 host=web-server-2
  - num_requests_per_second 1439828252 30 host=web-server-3

Row key	Data:1051	Data:1052
num_requests_per_second1439827200host=web-server-1	50	
num_requests_per_second1439827200host=web-server-2	72	
num_requests_per_second1439827200host=web-server-3		30

cloudera

OpenTSDB : GUI

TSD Time Series Database

Graph | Stats | Logs | Version

From 2014/09/28-10:00:00 To (now) Autoreload WxH: 835x400

2014/09/28-22:30:00

transfer_time +

Metric: curl.transfer_time Rate Rate Ctr Right Axis

Tags node ams1 Rate Ctr Max:

x host www.akamai.com Rate Ctr Reset:

Aggregator: avg

Downsample avg 1m

Axes | Key | Style

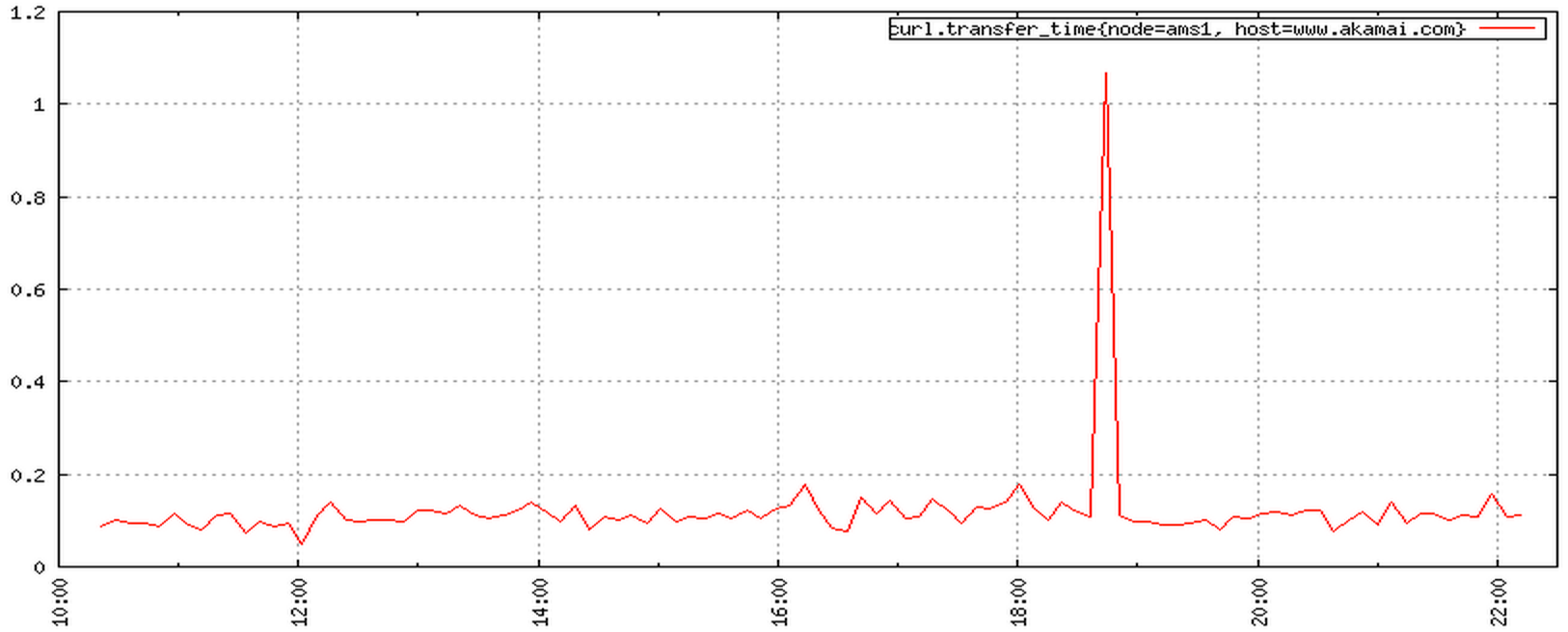
Key location:

Horizontal layout

Box

No key (overrides others)

5656 points retrieved, 655 points plotted in 46ms.



cloudera

- High performance relational database layer over HBase for low-latency applications
- JDBC API

cloudera

Phoenix : Use Case

Scalability of HBase

+

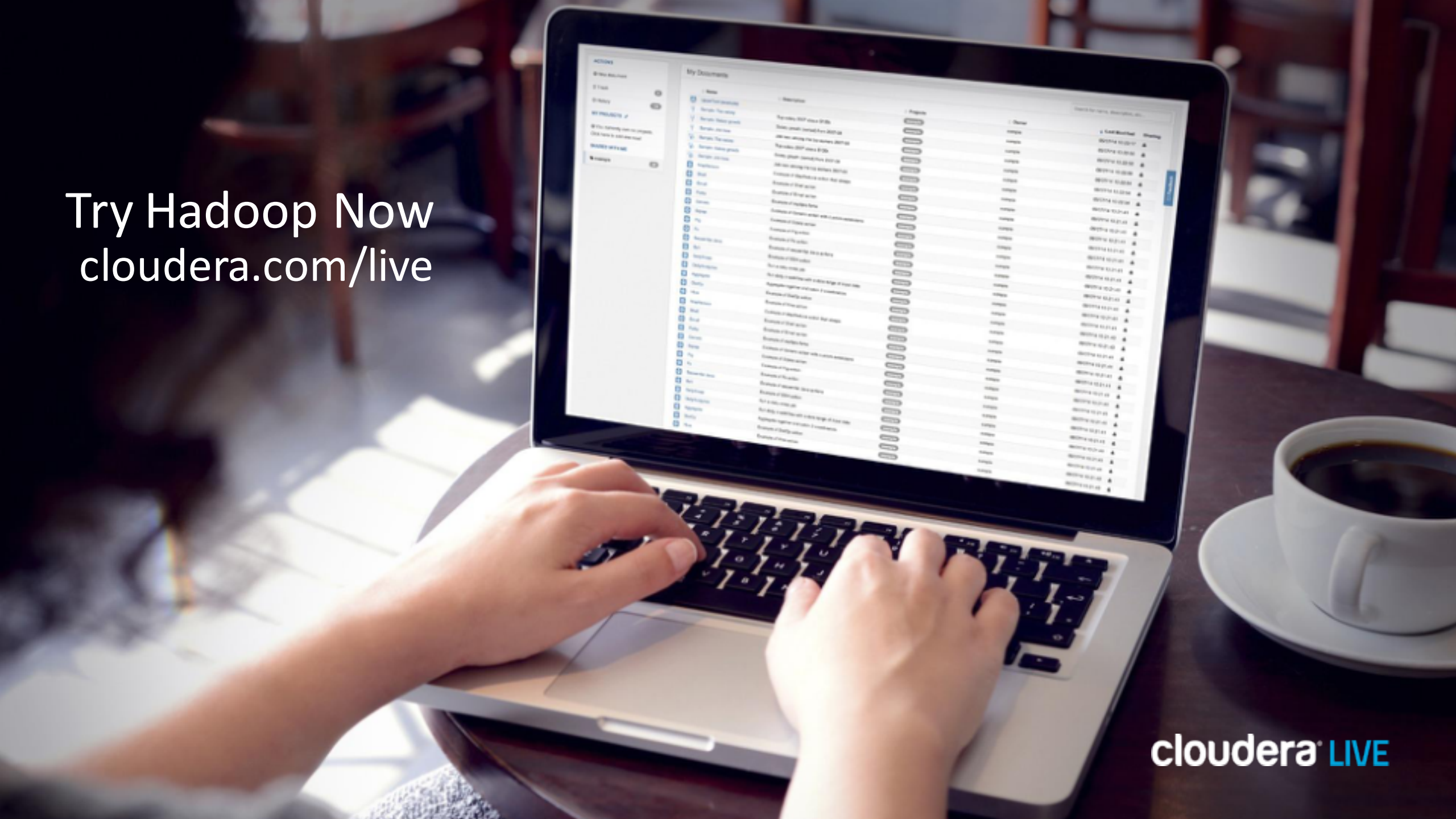
SQL interface access

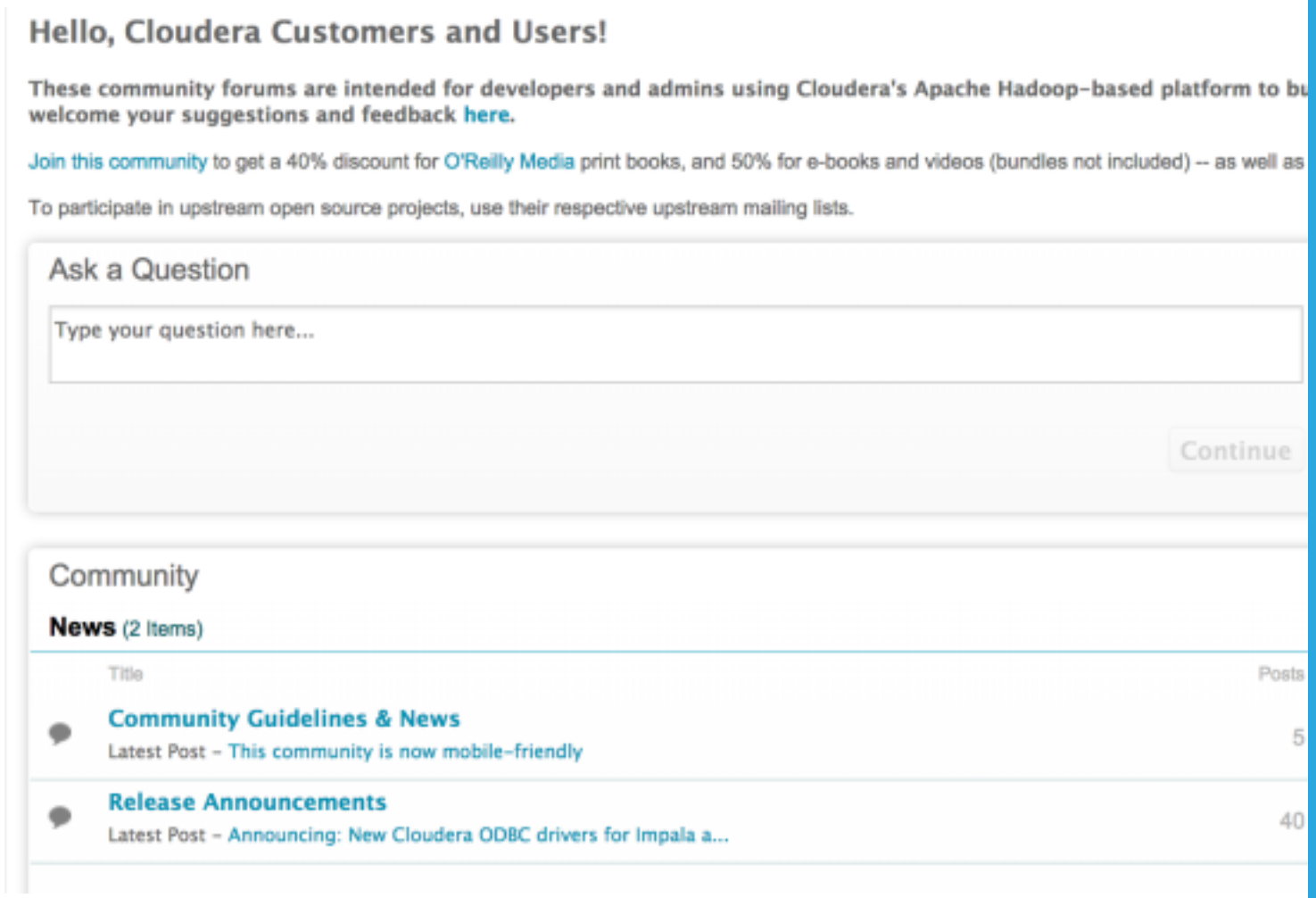
Phoenix

- Provides typed access to data
- Provides secondary indexes
- Compiles SQL queries to native HBase scans
- Executes scans parallely
- Directly uses HBase API, server-side hooks and custom filters
- Brings computation to the data
  - Pushes where clause to server-side filter
  - Executes aggregate queries using server-side hooks

That’s it folks!

cloudera


Try Hadoop Now
cloudera.com/live
cloudera LIVE



Get community help or provide feedback

cloudera.com/community

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

cloudera

Questions ?

