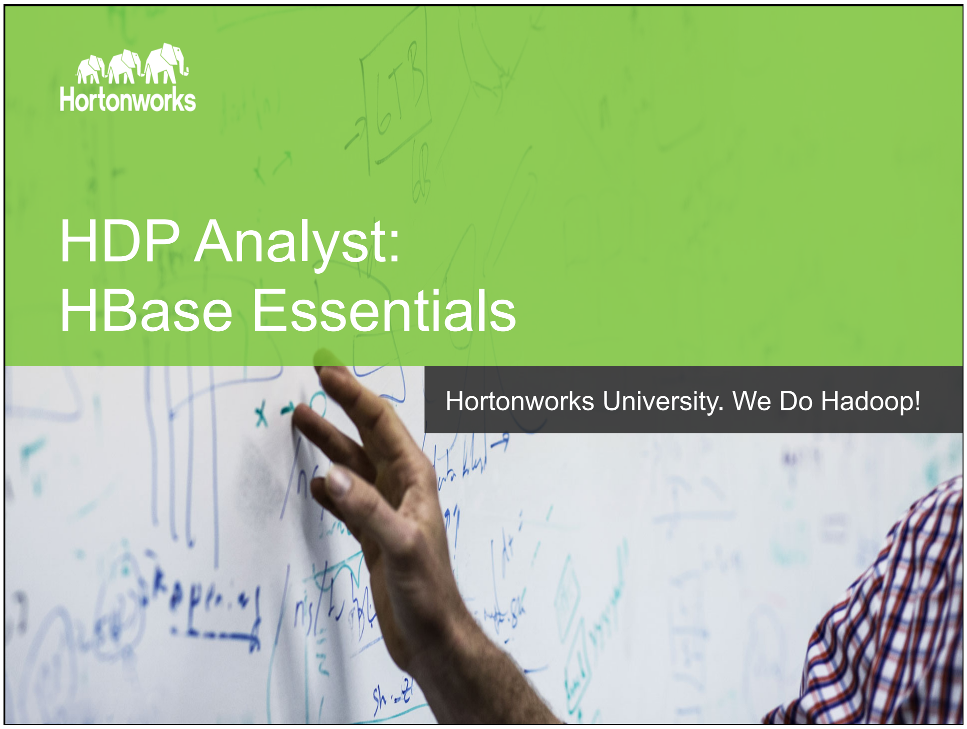
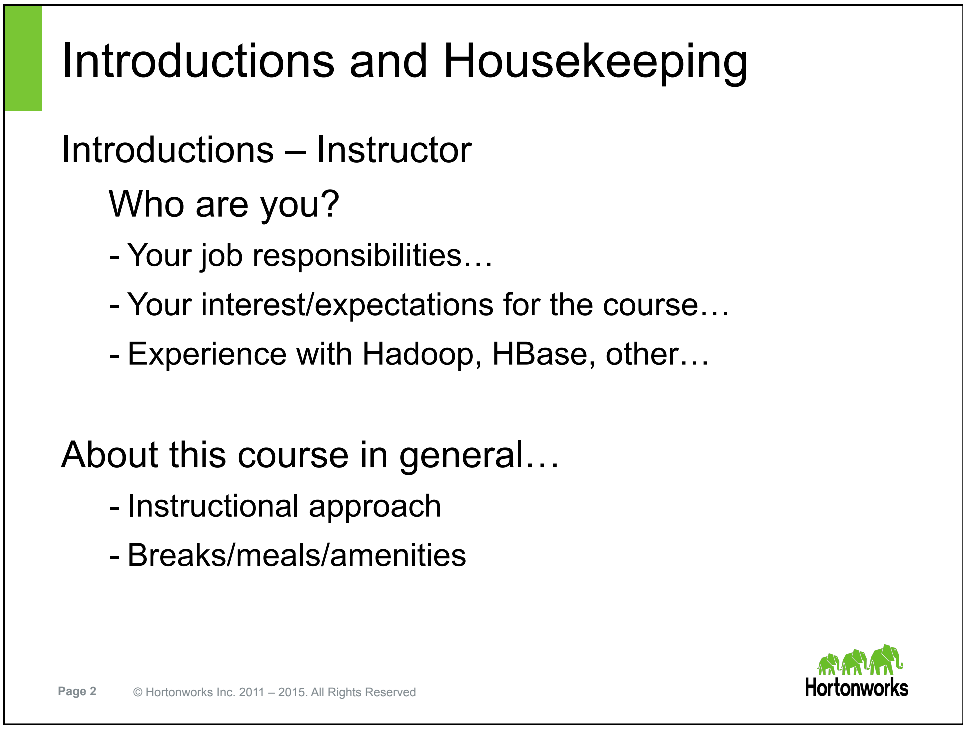


Hortonworks

HDP Analyst: HBase Essentials

Hortonworks University. We Do Hadoop!



Introductions and Housekeeping

Introductions – Instructor

Who are you?

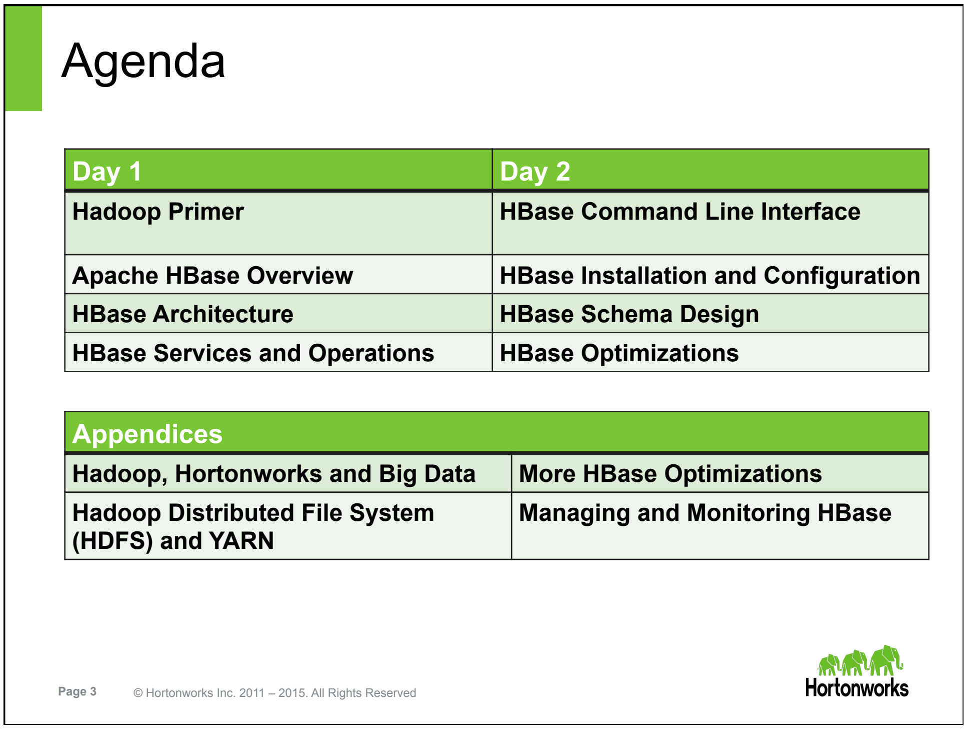
- Your job responsibilities…
- Your interest/expectations for the course…
- Experience with Hadoop, HBase, other…

About this course in general…

- Instructional approach
- Breaks/meals/amenities

Page 2

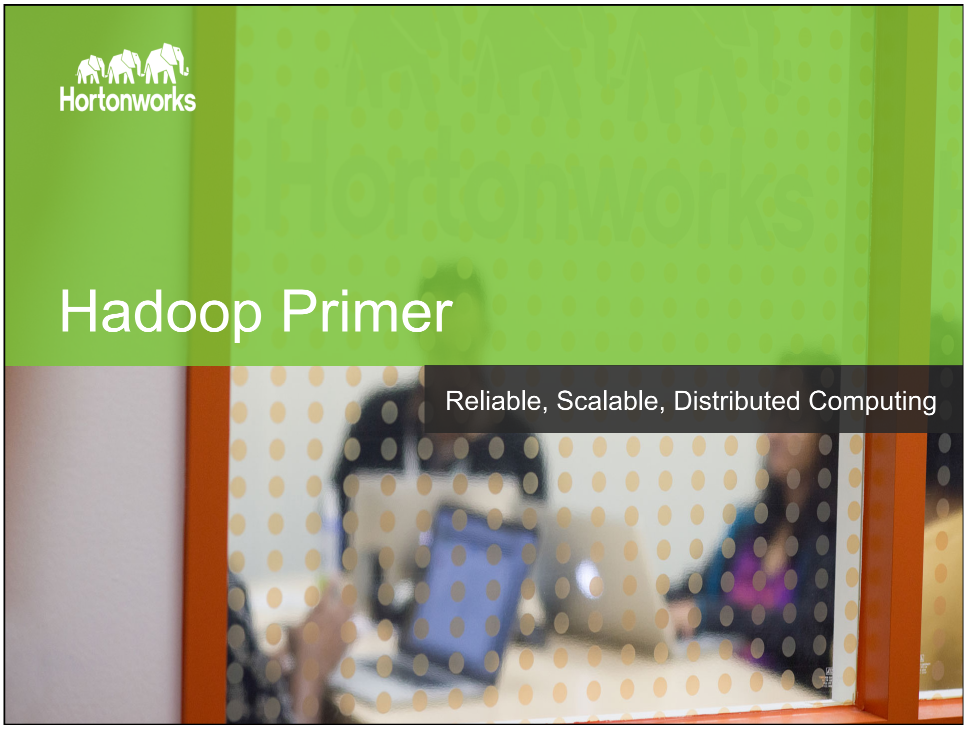
Hortonworks



Agenda

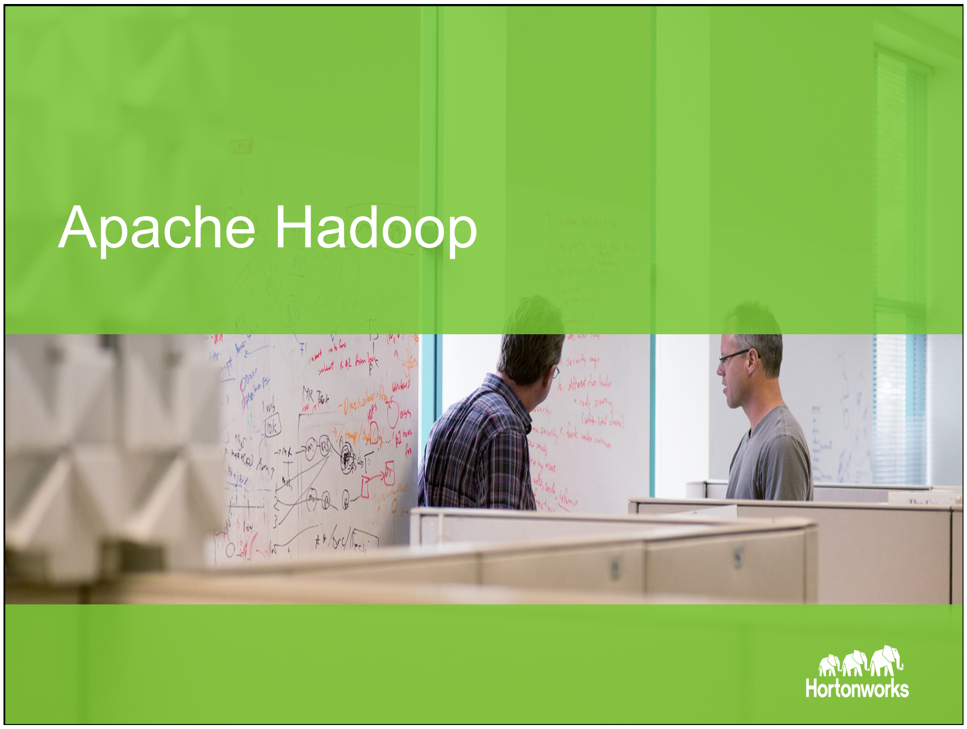
Day 1	Day 2
Hadoop Primer	HBase Command Line Interface
Apache HBase Overview	HBase Installation and Configuration
HBase Architecture	HBase Schema Design
HBase Services and Operations	HBase Optimizations

Appendices	
Hadoop, Hortonworks and Big Data	More HBase Optimizations
Hadoop Distributed File System (HDFS) and YARN	Managing and Monitoring HBase

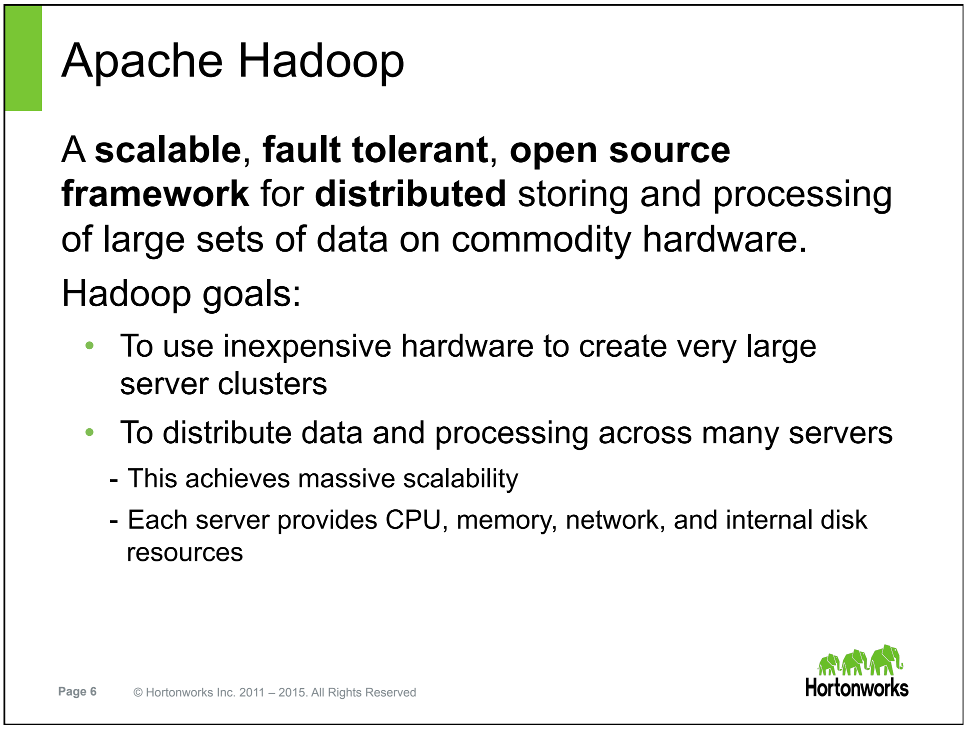


Hadoop Primer

Reliable, Scalable, Distributed Computing



Apache Hadoop



Apache Hadoop

A **scalable**, **fault tolerant**, **open source framework** for **distributed** storing and processing of large sets of data on commodity hardware.

Hadoop goals:

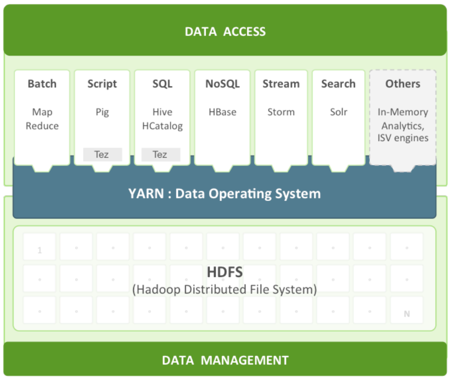
- To use inexpensive hardware to create very large server clusters
- To distribute data and processing across many servers
  - This achieves massive scalability
  - Each server provides CPU, memory, network, and internal disk resources

Page 6 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hadoop and Hortonworks Data Platform (HDP)

Hadoop is a framework that includes many components

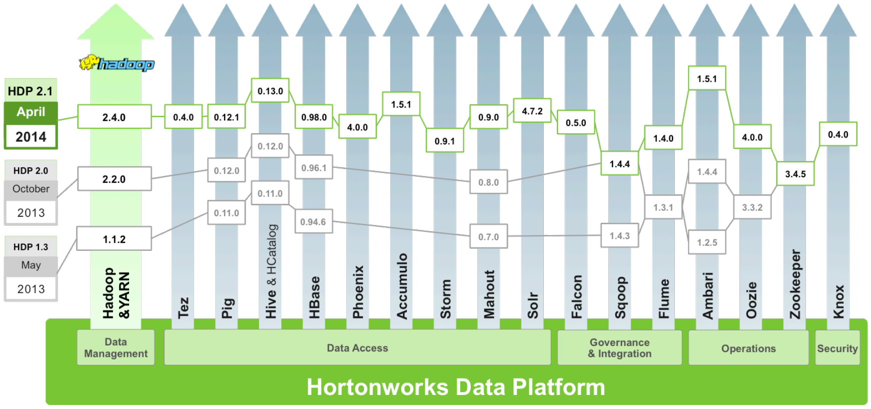
- A collection of other frameworks and tools managed as *projects* by the Apache Software Foundation
- A few projects are illustrated here



Page 7 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

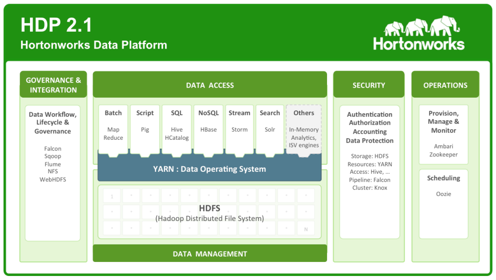
HDP and Apache Hadoop Project Versions



Page 8 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

HDP: Comprehensive Data Management Platform



Includes projects for:

- Core Hadoop (HDFS and YARN)
- Data access and processing
- Data governance and integration
- Security
- Managing Hadoop operations

Page 9 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

How Hadoop Stores Files

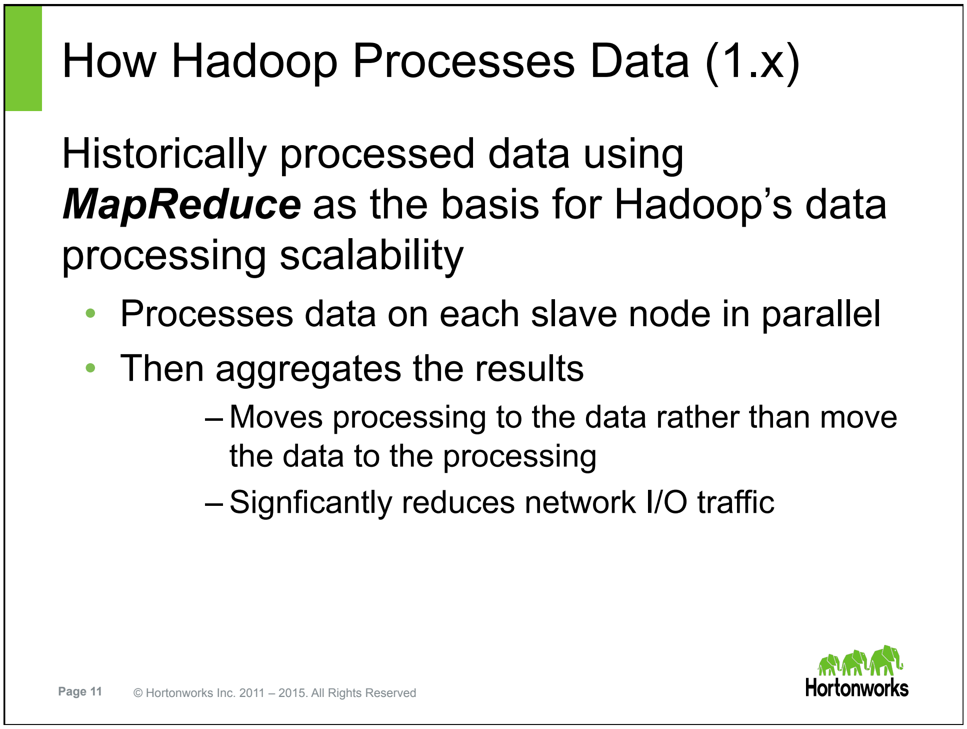
Uses the ***Hadoop distributed file system*** (HDFS) as

basis for Hadoop's storage scalability and availability

- Splits large data files into smaller chunks called blocks
- Spreads those blocks across different slave nodes
- Tracks data block location
- Automatically replicates data for high availability

Page 10 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

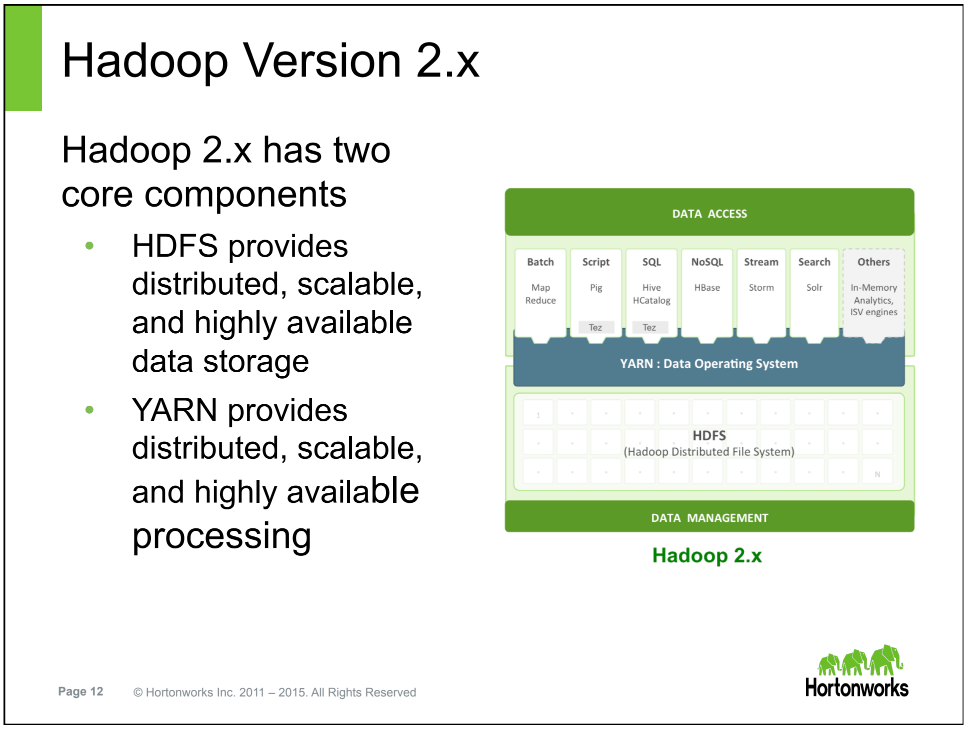


How Hadoop Processes Data (1.x)

Historically processed data using ***MapReduce*** as the basis for Hadoop's data processing scalability

- Processes data on each slave node in parallel
- Then aggregates the results
  - Moves processing to the data rather than move the data to the processing
  - Signficantly reduces network I/O traffic

Hortonworks



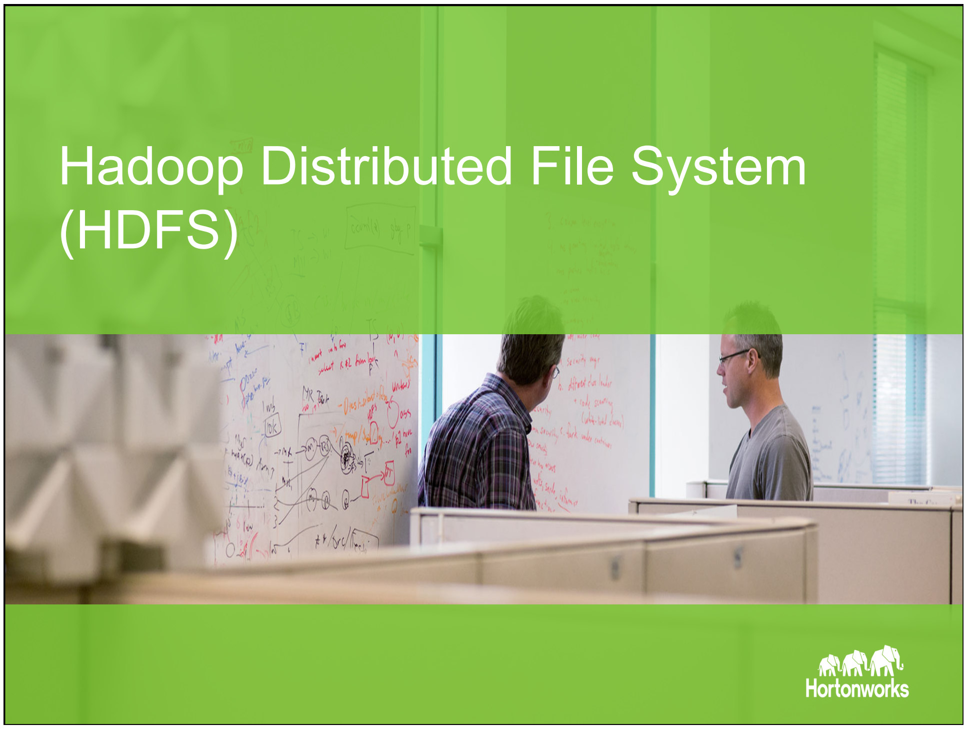
Hadoop Version 2.x

Hadoop 2.x has two core components

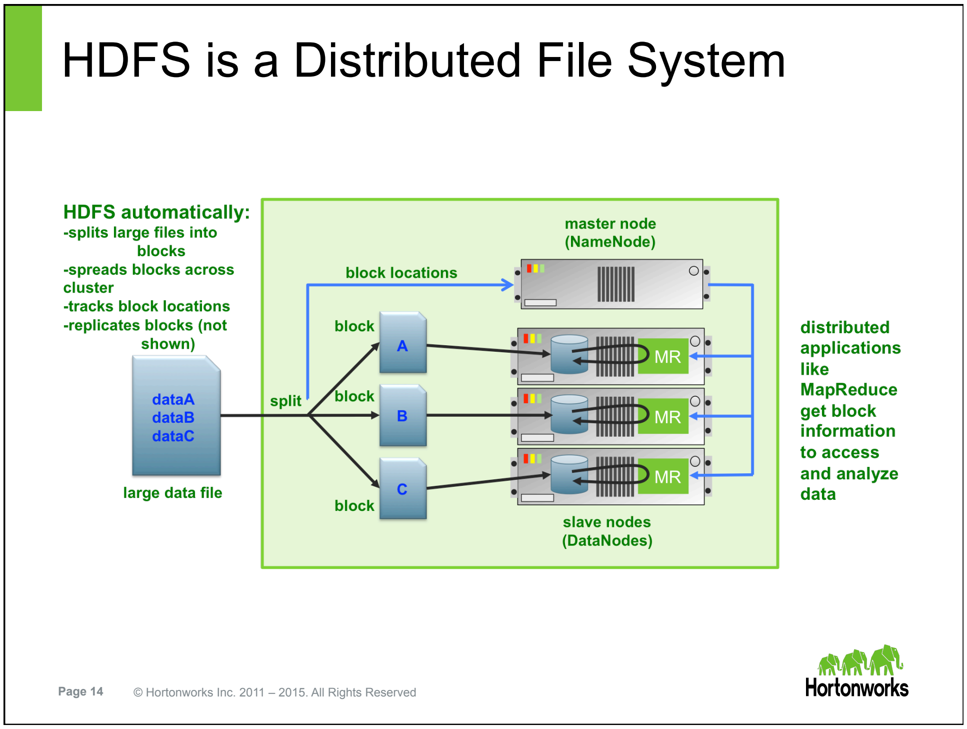
- HDFS provides distributed, scalable, and highly available data storage
- YARN provides distributed, scalable, and highly available processing

Hadoop 2.x

Hortonworks



Hadoop Distributed File System (HDFS)

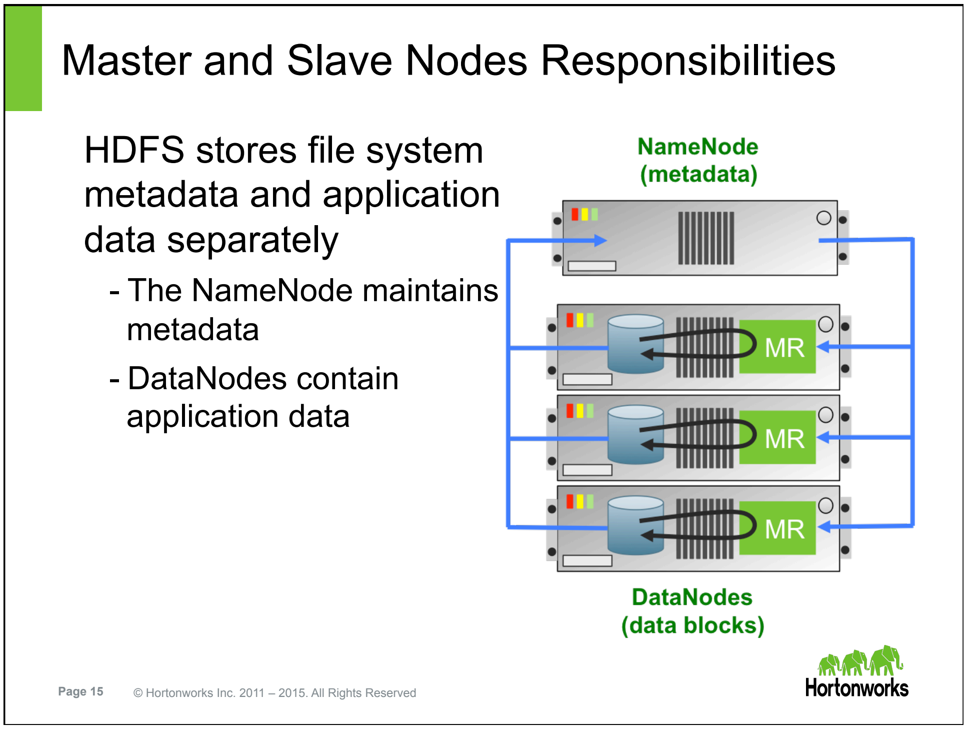


HDFS is a Distributed File System

HDFS automatically:
-splits large files into blocks
-spreads blocks across cluster
-tracks block locations
-replicates blocks (not shown)

distributed applications like MapReduce get block information to access and analyze data

Page 14



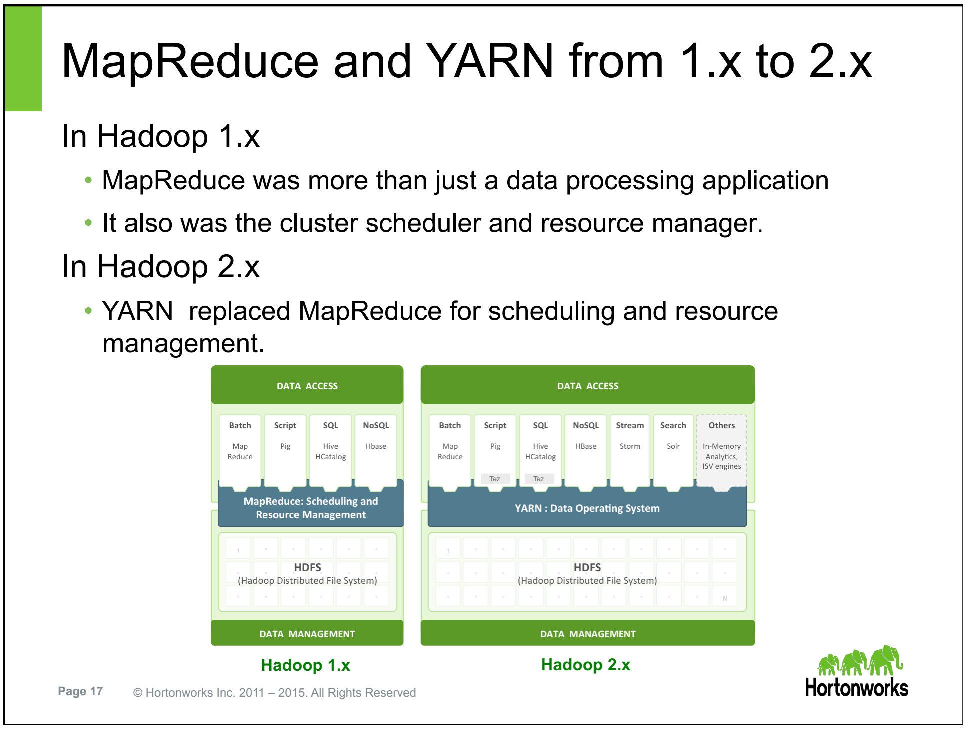
Master and Slave Nodes Responsibilities

HDFS stores file system metadata and application data separately

- The NameNode maintains metadata
- DataNodes contain application data

Page 15

YARN



MapReduce and YARN from 1.x to 2.x

In Hadoop 1.x

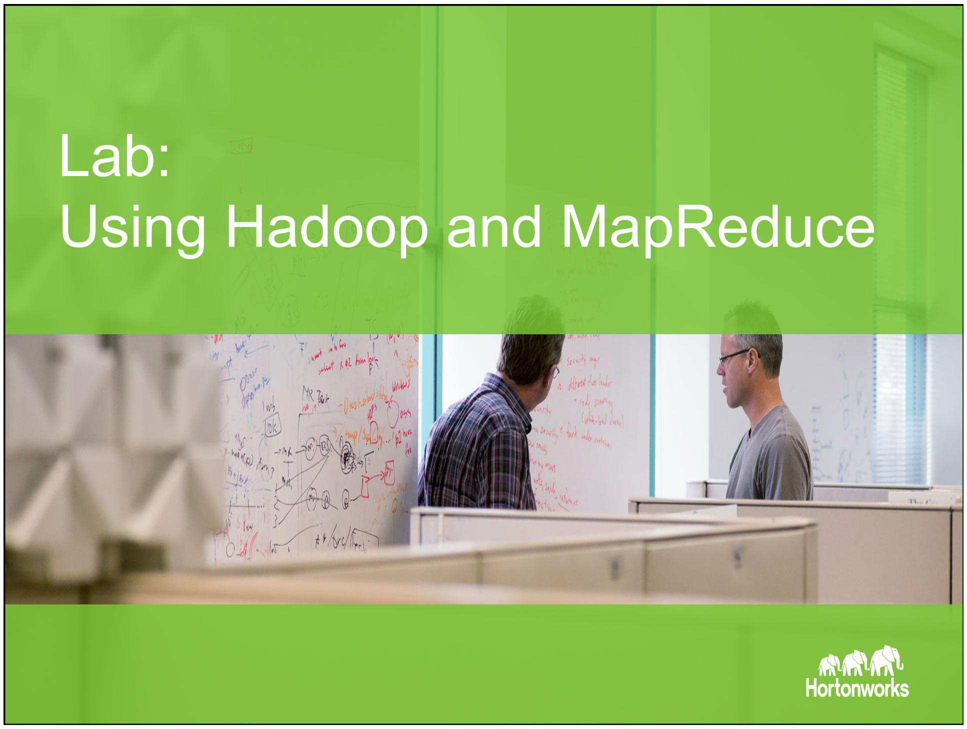
- MapReduce was more than just a data processing application
- It also was the cluster scheduler and resource manager.

In Hadoop 2.x

- YARN replaced MapReduce for scheduling and resource management.

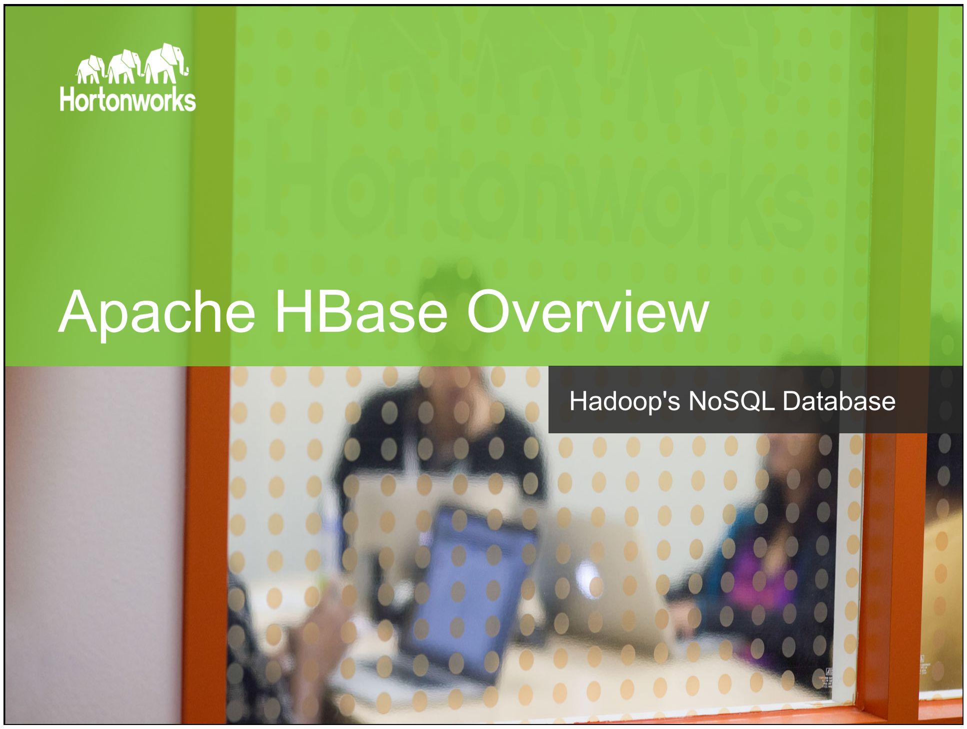
Page 17

Hortonworks



Lab:
Using Hadoop and MapReduce

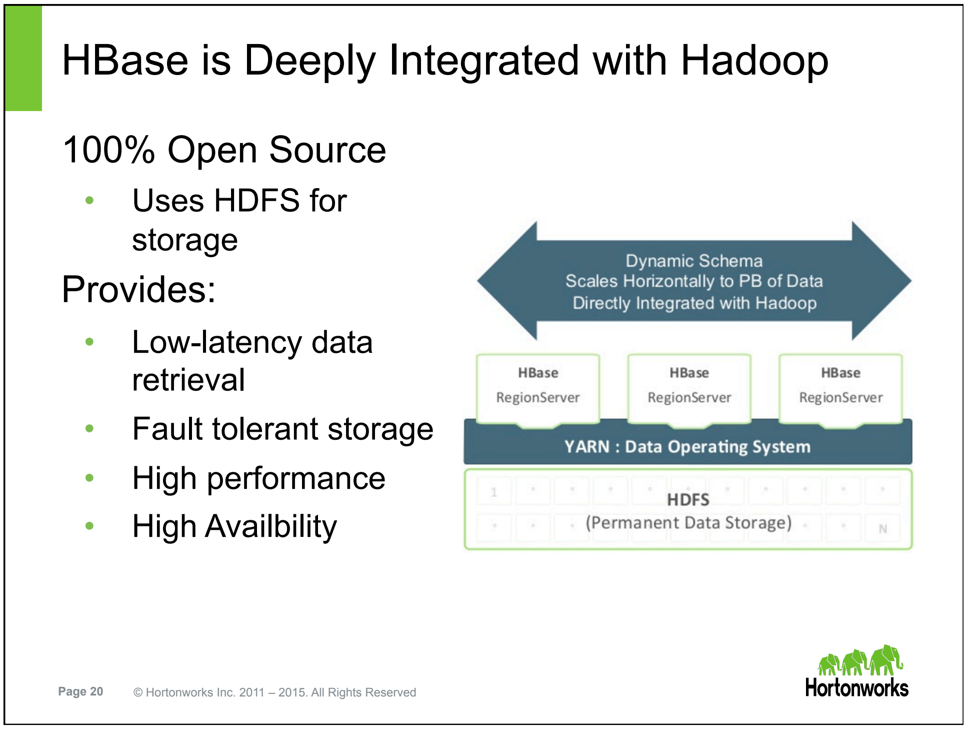
Hortonworks



Hortonworks

Apache HBase Overview

Hadoop's NoSQL Database



HBase is Deeply Integrated with Hadoop

100% Open Source

- Uses HDFS for storage

Provides:

- Low-latency data retrieval
- Fault tolerant storage
- High performance
- High Availbility

Page 20

Hortonworks

Apache HBase

A non-relational (NoSQL) database

- Created for hosting very large tables with billions of rows and millions of columns

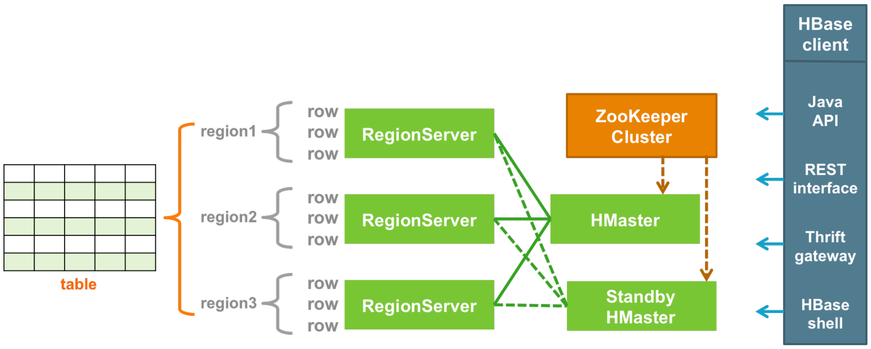
HBase:

- Provides random, real-time data access
- Allows table inserts, updates, and deletes
- Runs on top of the Hadoop distributed file system
  - HBase data is automatically replicated by HDFS for higher availability.

Page 21

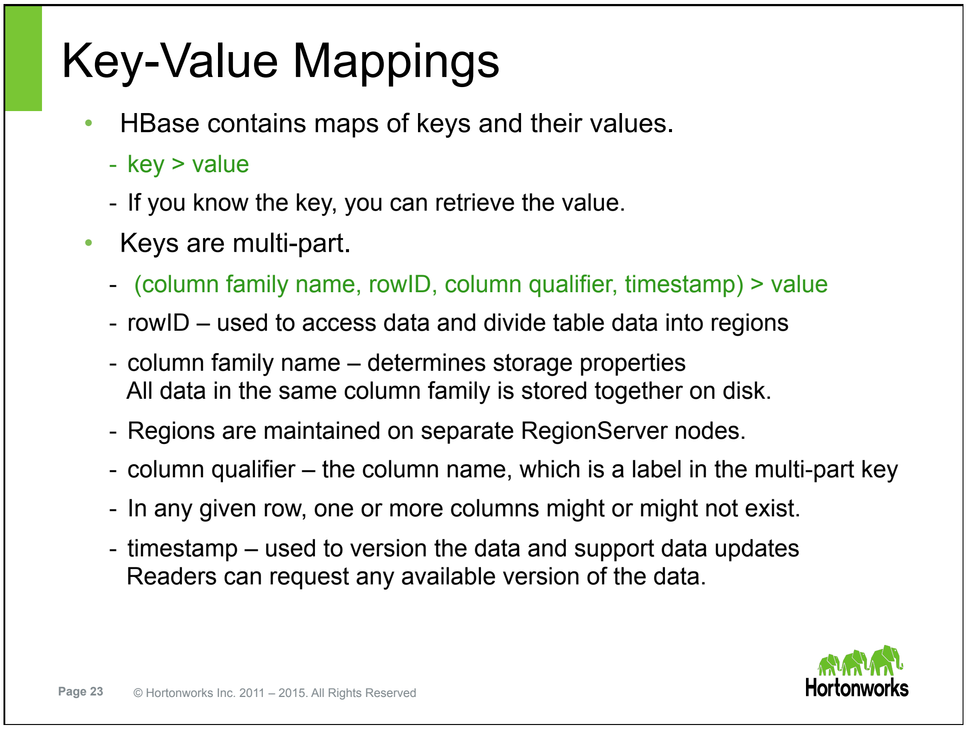
Hortonworks

HBase Architecture



Page 22

Hortonworks

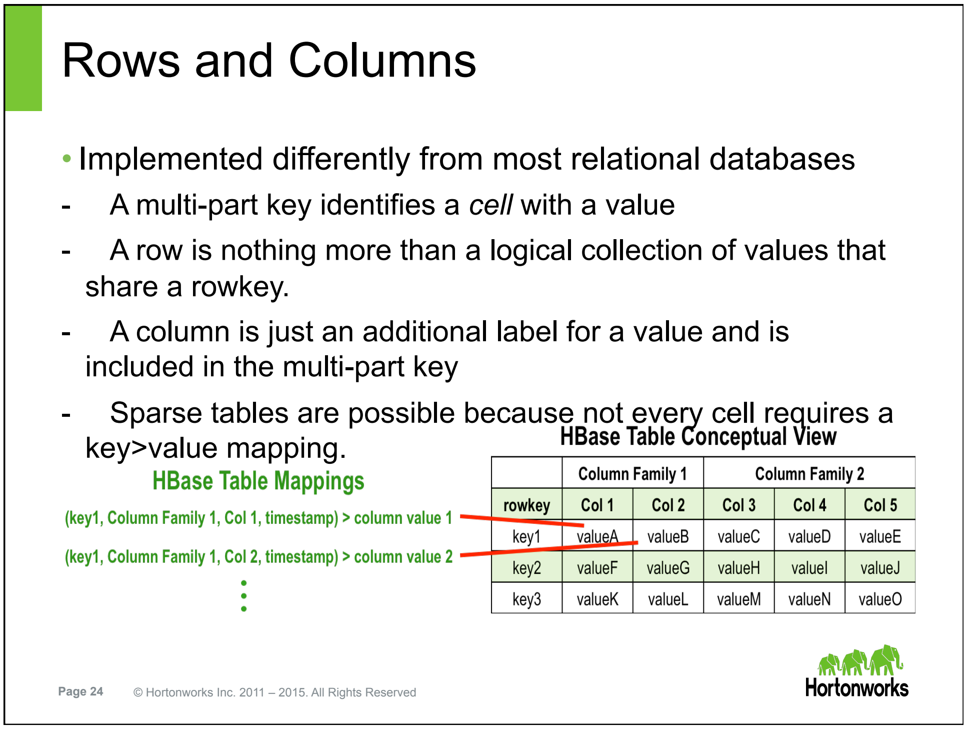


Key-Value Mappings

- HBase contains maps of keys and their values.
  - key > value
  - If you know the key, you can retrieve the value.
- Keys are multi-part.
  - (column family name, rowID, column qualifier, timestamp) > value
  - rowID – used to access data and divide table data into regions
  - column family name – determines storage properties
    All data in the same column family is stored together on disk.
  - Regions are maintained on separate RegionServer nodes.
  - column qualifier – the column name, which is a label in the multi-part key
  - In any given row, one or more columns might or might not exist.
  - timestamp – used to version the data and support data updates
    Readers can request any available version of the data.

Page 23 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Rows and Columns

- Implemented differently from most relational databases

- A multi-part key identifies a *cell* with a value
- A row is nothing more than a logical collection of values that share a rowkey.
- A column is just an additional label for a value and is included in the multi-part key
- Sparse tables are possible because not every cell requires a key>value mapping.

HBase Table Mappings

(key1, Column Family 1, Col 1, timestamp) > column value 1

(key1, Column Family 1, Col 2, timestamp) > column value 2

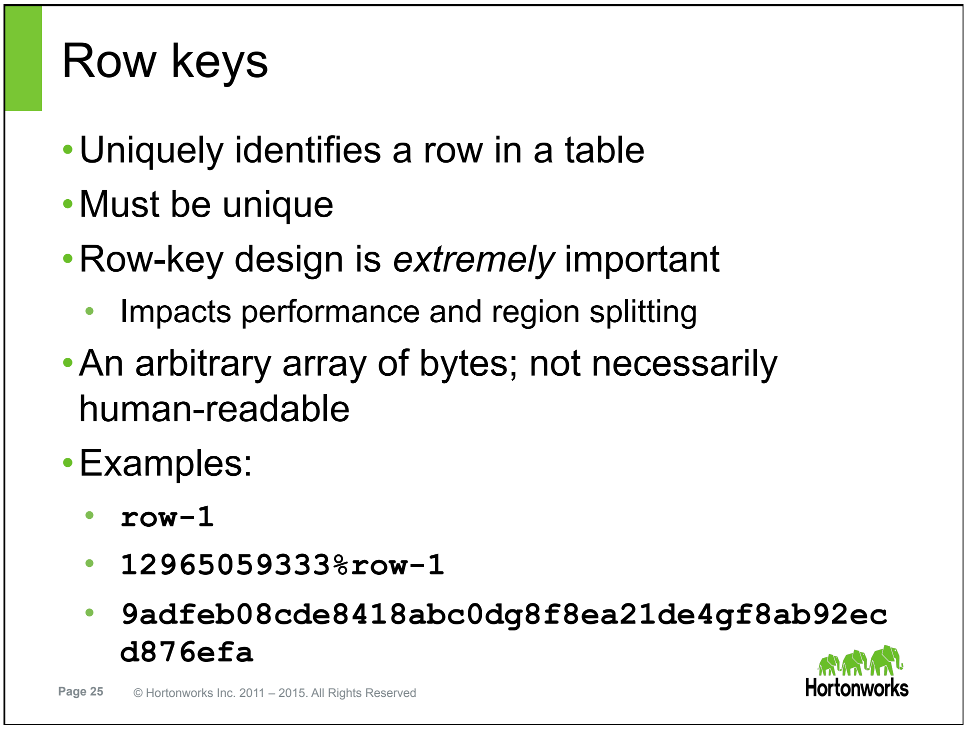
⋮

HBase Table Conceptual View

	Column Family 1		Column Family 2		
rowkey	Col 1	Col 2	Col 3	Col 4	Col 5
key1	valueA	valueB	valueC	valueD	valueE
key2	valueF	valueG	valueH	valueI	valueJ
key3	valueK	valueL	valueM	valueN	valueO

Page 24 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

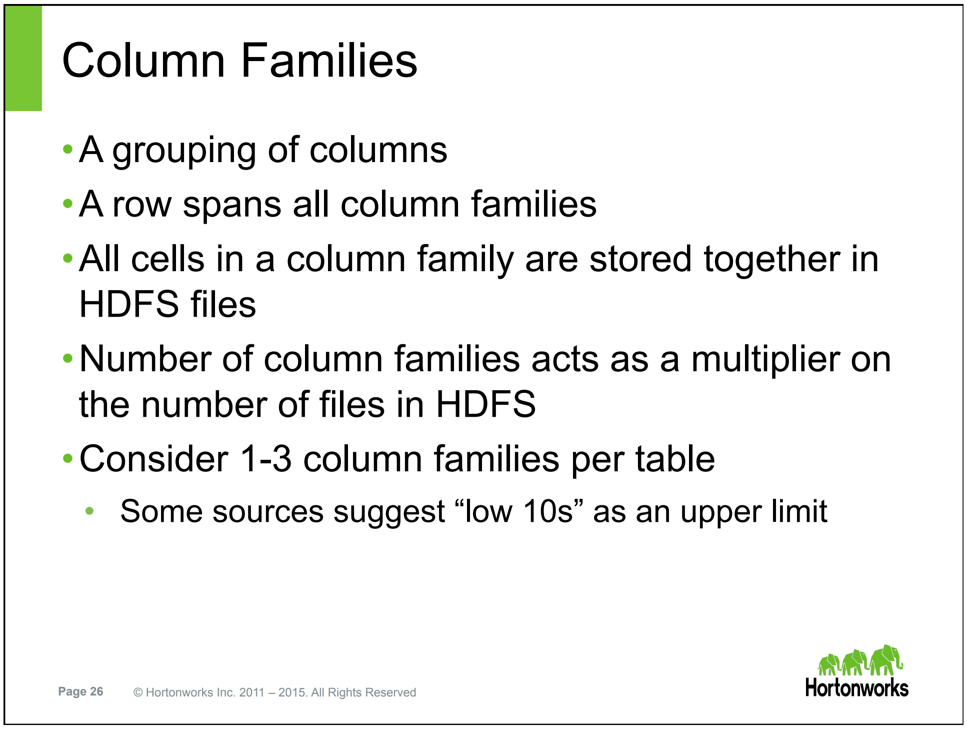
Hortonworks



Row keys

- Uniquely identifies a row in a table
- Must be unique
- Row-key design is *extremely* important
  - Impacts performance and region splitting
- An arbitrary array of bytes; not necessarily human-readable
- Examples:
  - `row-1`
  - `12965059333%row-1`
  - `9adfeb08cde8418abc0dg8f8ea21de4gf8ab92ec d876efa`

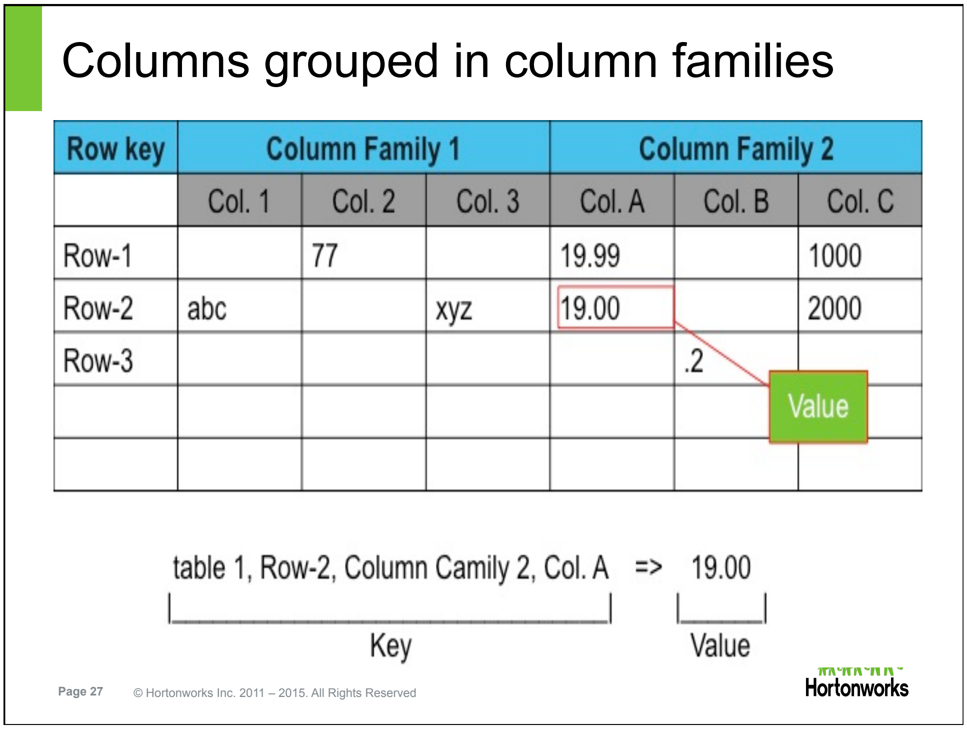
Page 25



Column Families

- A grouping of columns
- A row spans all column families
- All cells in a column family are stored together in HDFS files
- Number of column families acts as a multiplier on the number of files in HDFS
- Consider 1-3 column families per table
  - Some sources suggest “low 10s” as an upper limit

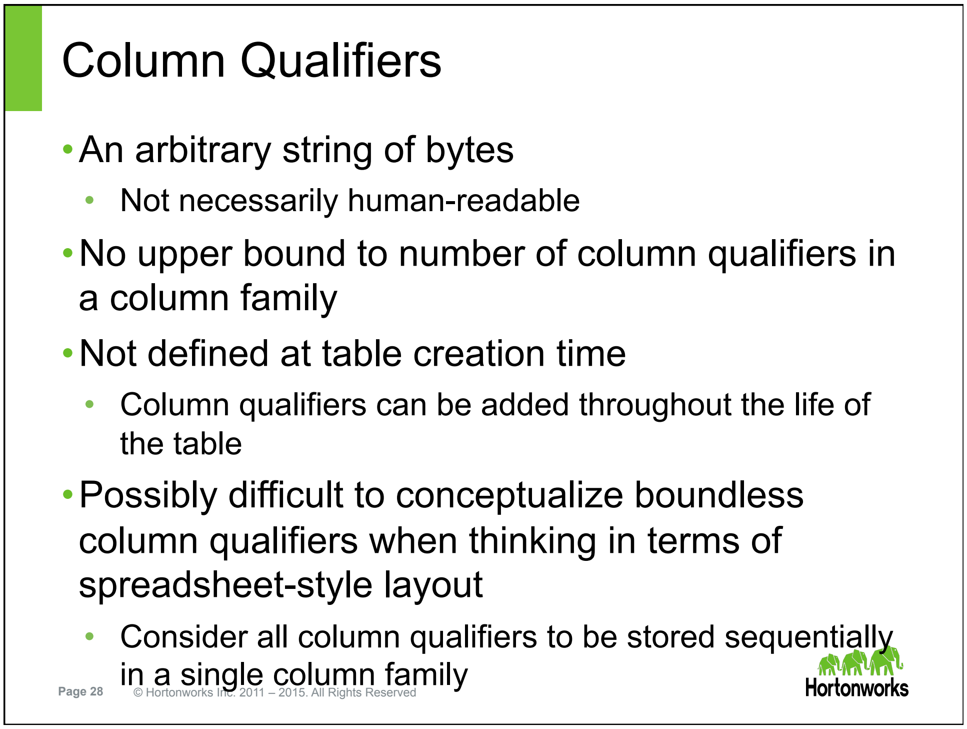
Page 26



Columns grouped in column families

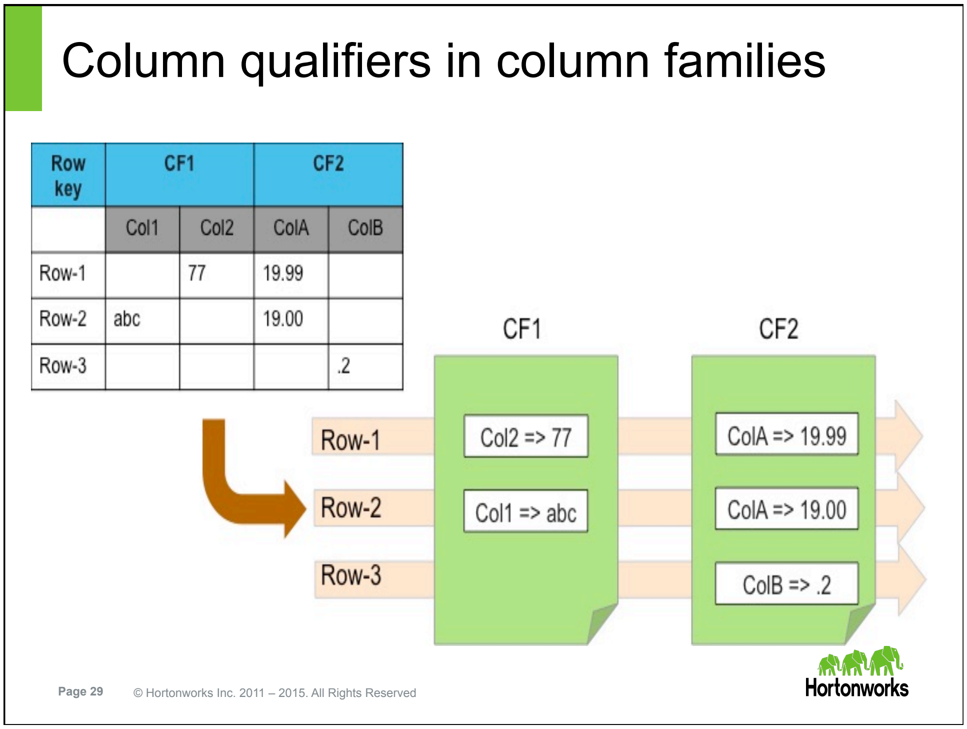
Row key	Column Family 1			Column Family 2		
	Col. 1	Col. 2	Col. 3	Col. A	Col. B	Col. C
Row-1		77		19.99		1000
Row-2	abc		xyz	19.00		2000
Row-3					.2	

table 1, Row-2, Column Camily 2, Col. A => 19.00



Column Qualifiers

- An arbitrary string of bytes
  - Not necessarily human-readable
- No upper bound to number of column qualifiers in a column family
- Not defined at table creation time
  - Column qualifiers can be added throughout the life of the table
- Possibly difficult to conceptualize boundless column qualifiers when thinking in terms of spreadsheet-style layout
  - Consider all column qualifiers to be stored sequentially in a single column family

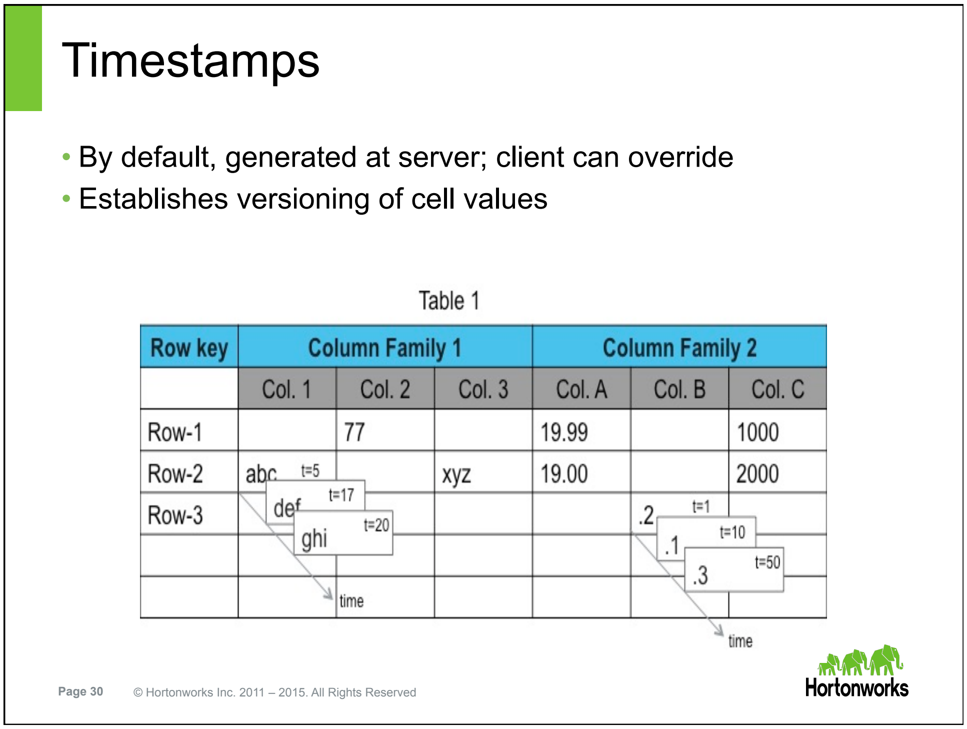


Column qualifiers in column families

Row key	CF1		CF2	
	Col1	Col2	ColA	ColB
Row-1		77	19.99	
Row-2	abc		19.00	
Row-3				.2

Page 29

Hortonworks



Timestamps

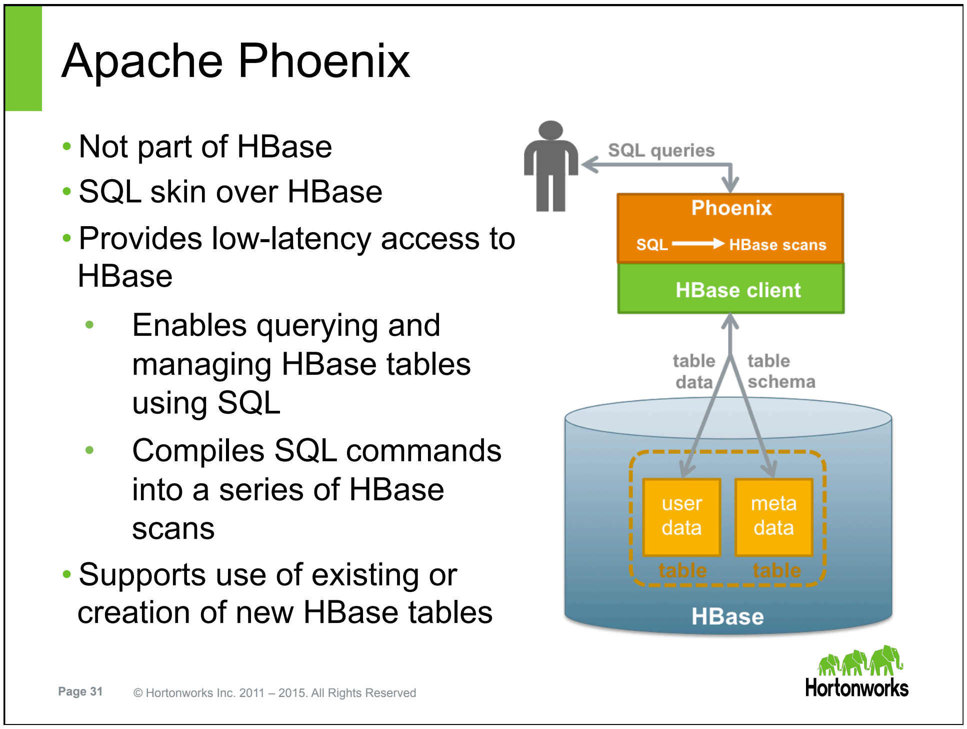
- By default, generated at server; client can override
- Establishes versioning of cell values

Table 1

Row key	Column Family 1			Column Family 2		
	Col. 1	Col. 2	Col. 3	Col. A	Col. B	Col. C
Row-1		77		19.99		1000
Row-2	abc t=5 def t=17 ghi t=20		xyz	19.00		2000
Row-3					.2 t=1 .1 t=10 .3 t=50	

Page 30

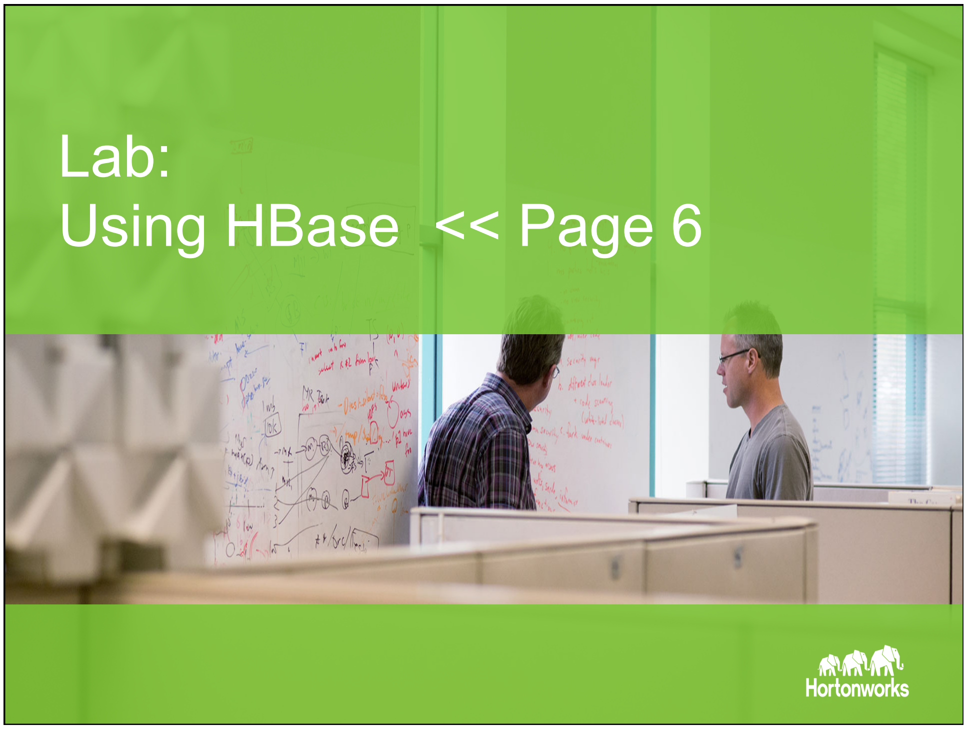
Hortonworks



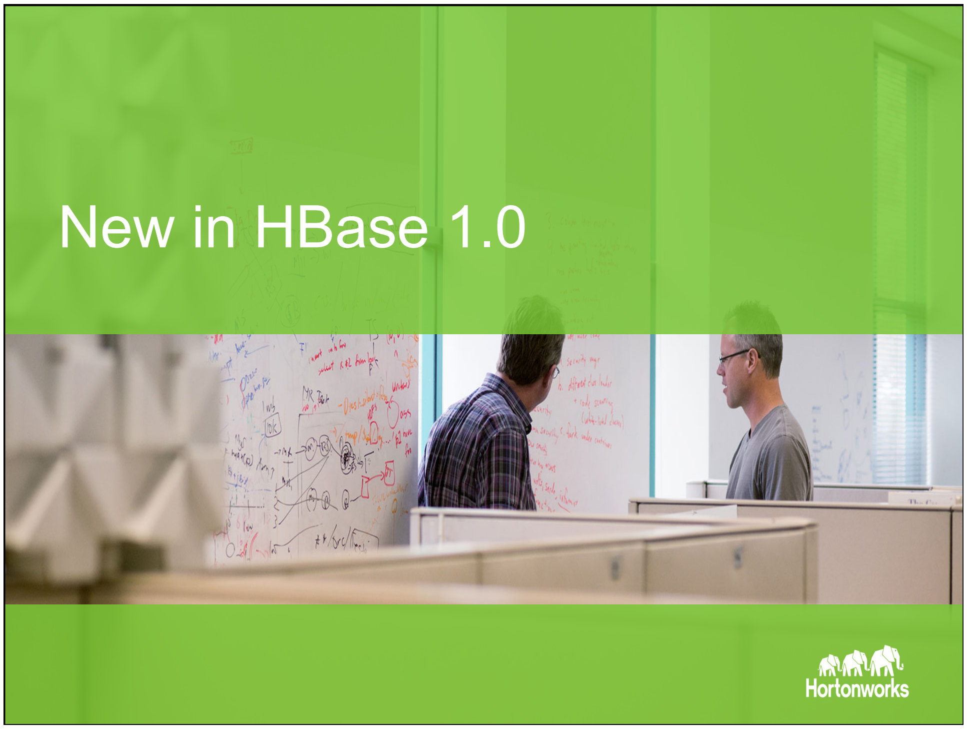
Apache Phoenix

- Not part of HBase
- SQL skin over HBase
- Provides low-latency access to HBase
  - Enables querying and managing HBase tables using SQL
  - Compiles SQL commands into a series of HBase scans
- Supports use of existing or creation of new HBase tables

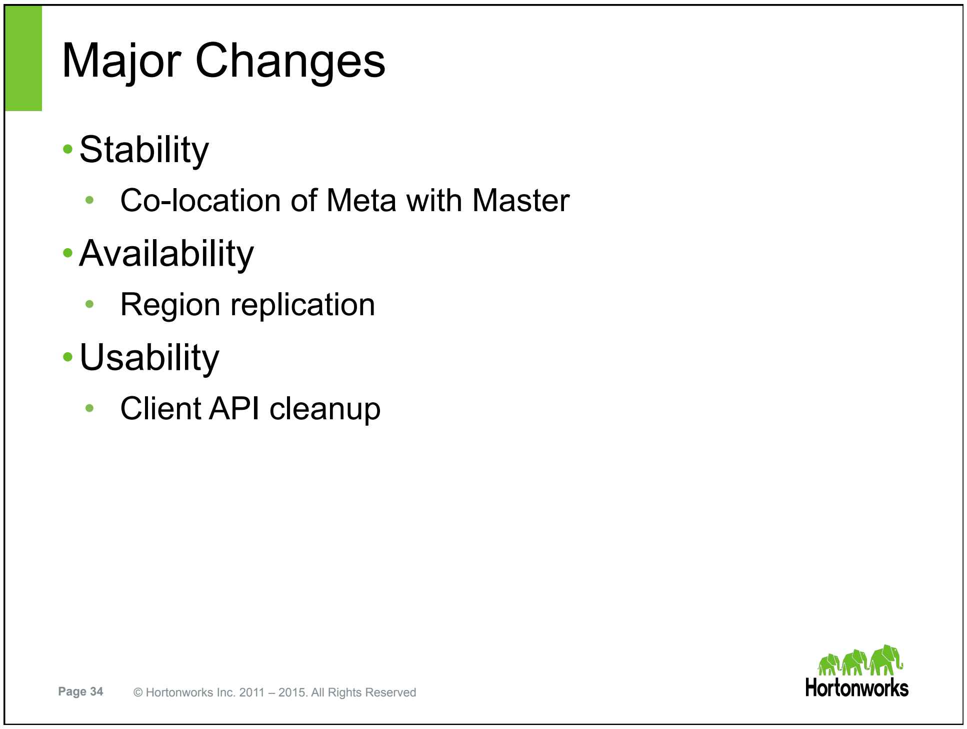
Page 31



Lab: Using HBase << Page 6



New in HBase 1.0

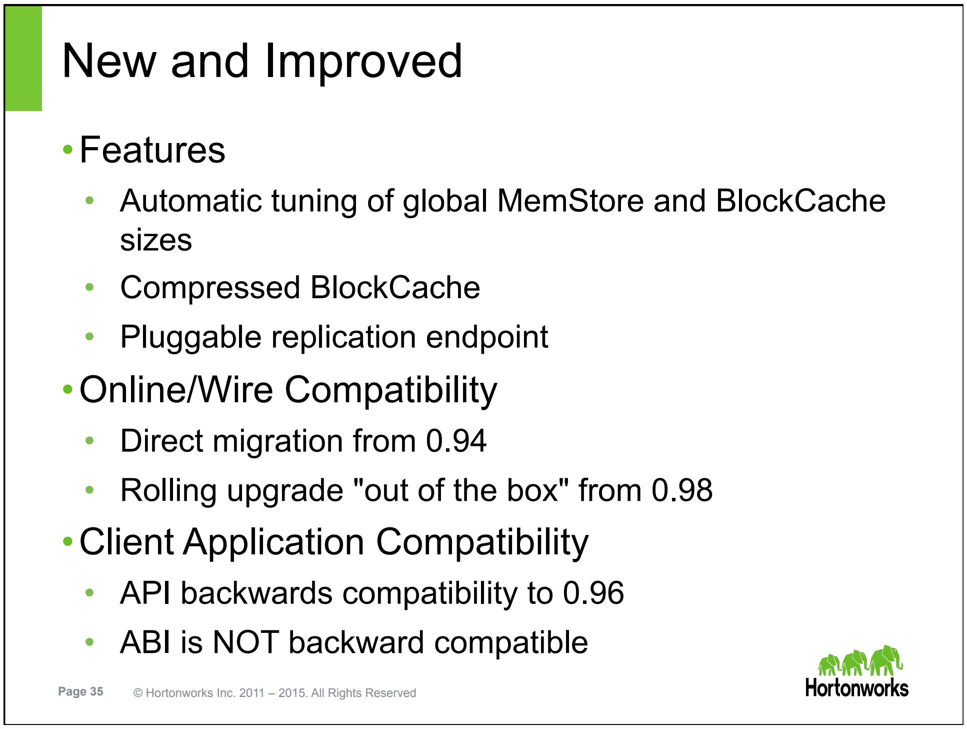


Major Changes

- Stability
  - Co-location of Meta with Master
- Availability
  - Region replication
- Usability
  - Client API cleanup

Page 34

Hortonworks



New and Improved

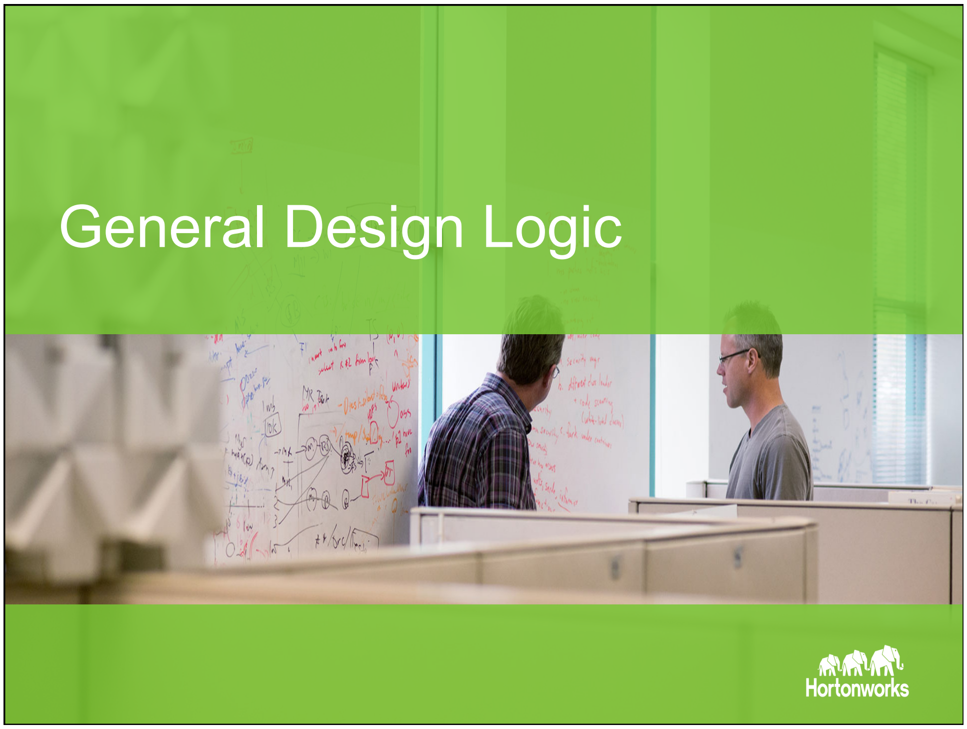
- Features
  - Automatic tuning of global MemStore and BlockCache sizes
  - Compressed BlockCache
  - Pluggable replication endpoint
- Online/Wire Compatibility
  - Direct migration from 0.94
  - Rolling upgrade "out of the box" from 0.98
- Client Application Compatibility
  - API backwards compatibility to 0.96
  - ABI is NOT backward compatible

Page 35

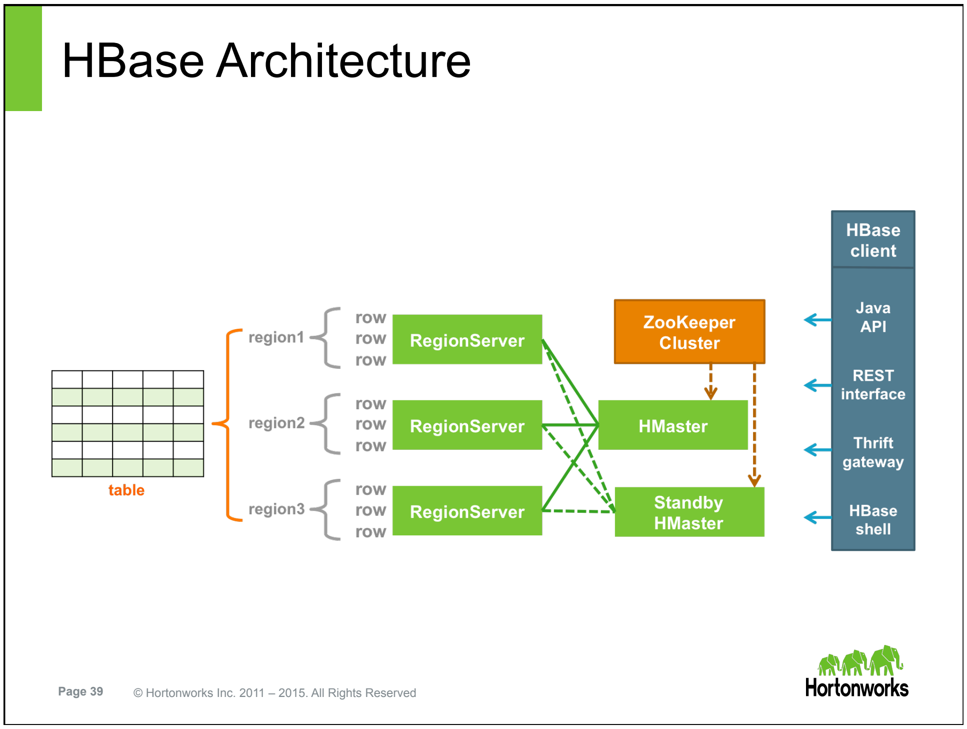
Lab: Importing from MySQL << Pg 12

HBase Architecture

Under the Hood



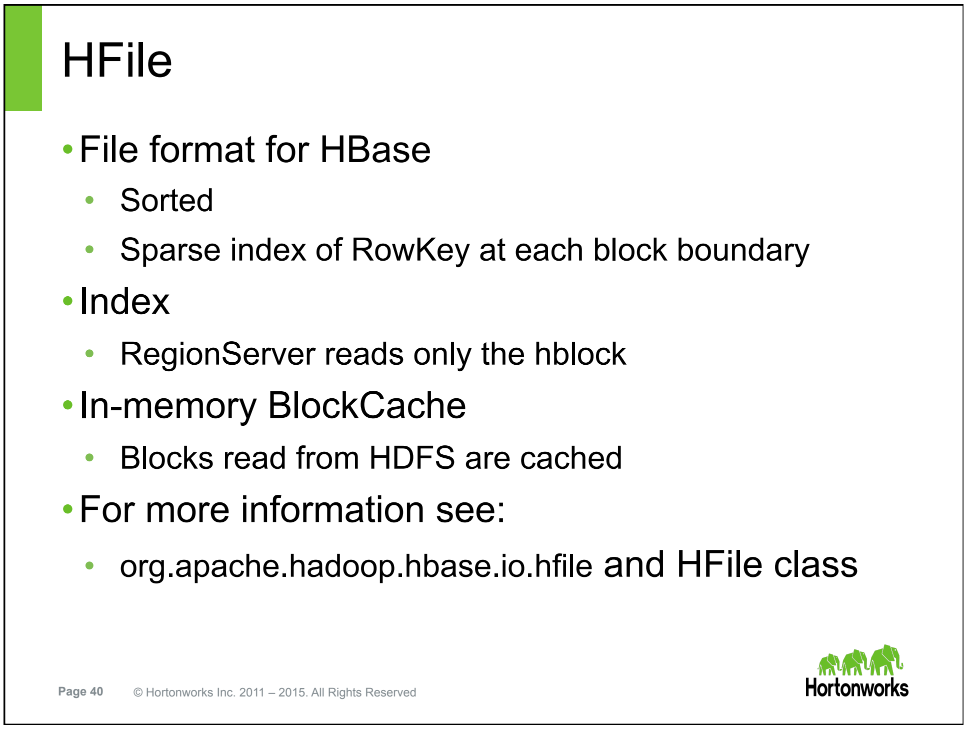
General Design Logic



HBase Architecture

Page 39

Hortonworks



HFile

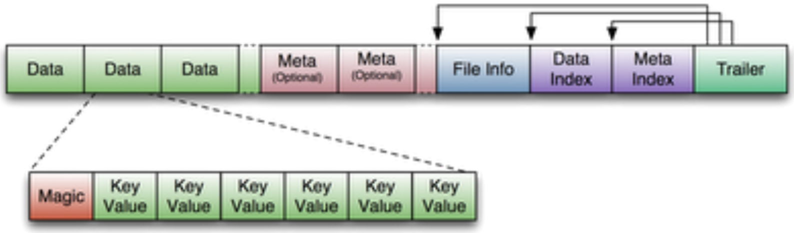
- File format for HBase
  - Sorted
  - Sparse index of RowKey at each block boundary
- Index
  - RegionServer reads only the hblock
- In-memory BlockCache
  - Blocks read from HDFS are cached
- For more information see:
  - org.apache.hadoop.hbase.io.hfile and HFile class

Page 40

Hortonworks

Block Index

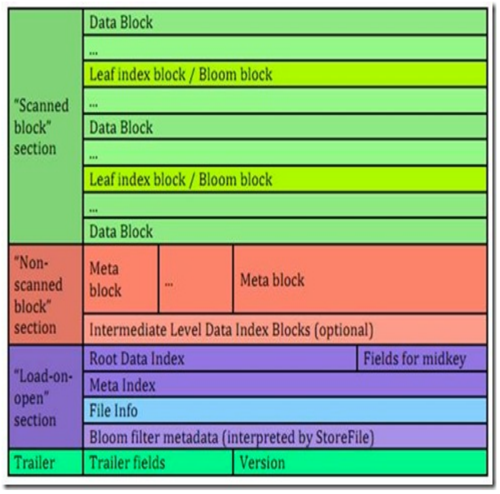
- Contains:
  - Offset (long)
  - Uncompressed size (int)
  - Key (a serialized byte array written using Bytes.writeByteArray)
    - Key length as a variable-length integer (Vint)
    - Key bytes



Data | Data | Data | Meta (Optional) | Meta (Optional) | File Info | Data Index | Meta Index | Trailer

Magic | Key Value | Key Value | Key Value | Key Value | Key Value | Key Value

Another Look at the HFile



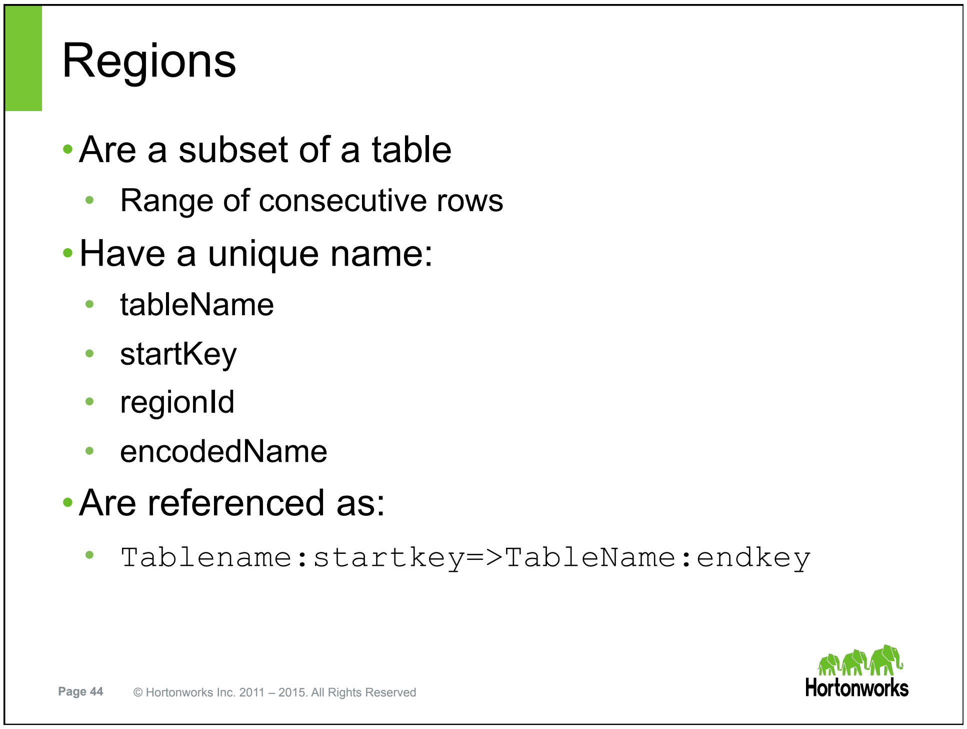
"Scanned block" section:
Data Block
...
Leaf index block / Bloom block
...
Data Block
...
Leaf index block / Bloom block
...
Data Block

"Non-scanned block" section:
Meta block | ... | Meta block
Intermediate Level Data Index Blocks (optional)

"Load-on-open" section:
Root Data Index | Fields for midkey
Meta Index
File Info
Bloom filter metadata (interpreted by StoreFile)

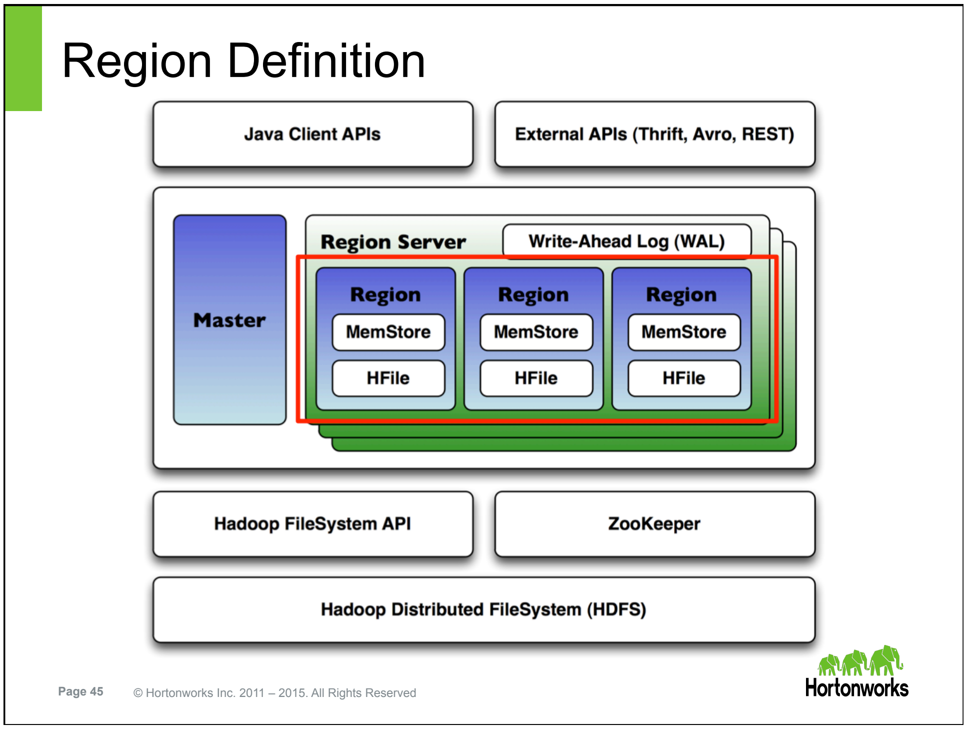
Trailer:
Trailer fields | Version

HBase Regions



Regions

- Are a subset of a table
  - Range of consecutive rows
- Have a unique name:
  - tableName
  - startKey
  - regionId
  - encodedName
- Are referenced as:
  - `Tablename:startkey=>TableName:endkey`



Region Definition

Java Client APIs

External APIs (Thrift, Avro, REST)

Master

Region Server

Write-Ahead Log (WAL)

Region

MemStore

HFile

Region

MemStore

HFile

Region

MemStore

HFile

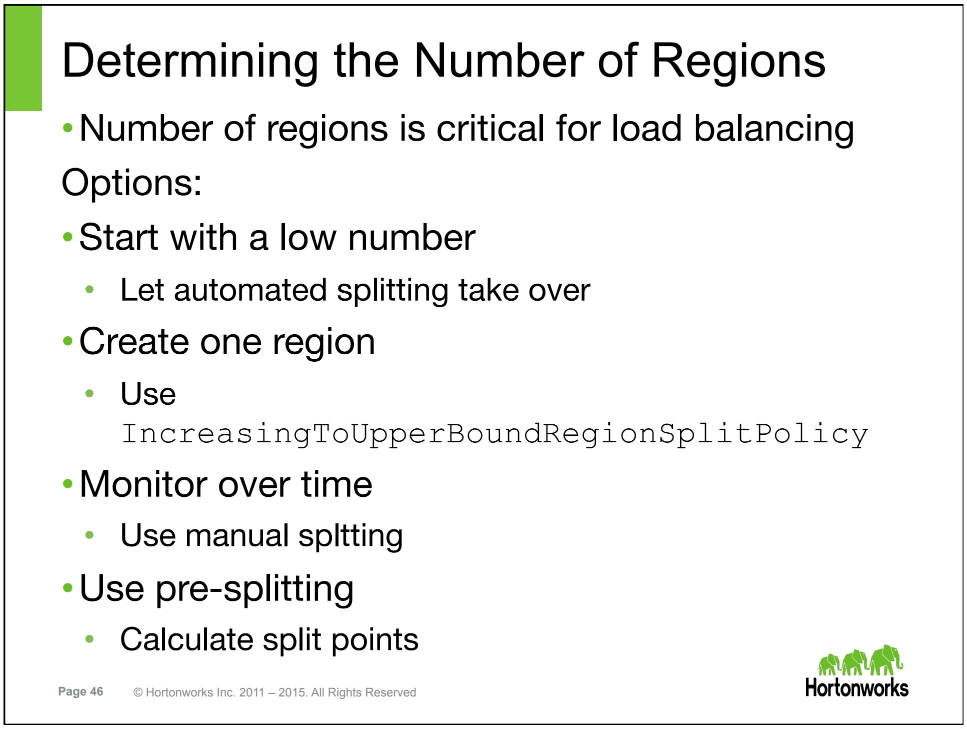
Hadoop FileSystem API

ZooKeeper

Hadoop Distributed FileSystem (HDFS)

Page 45

Hortonworks



Determining the Number of Regions

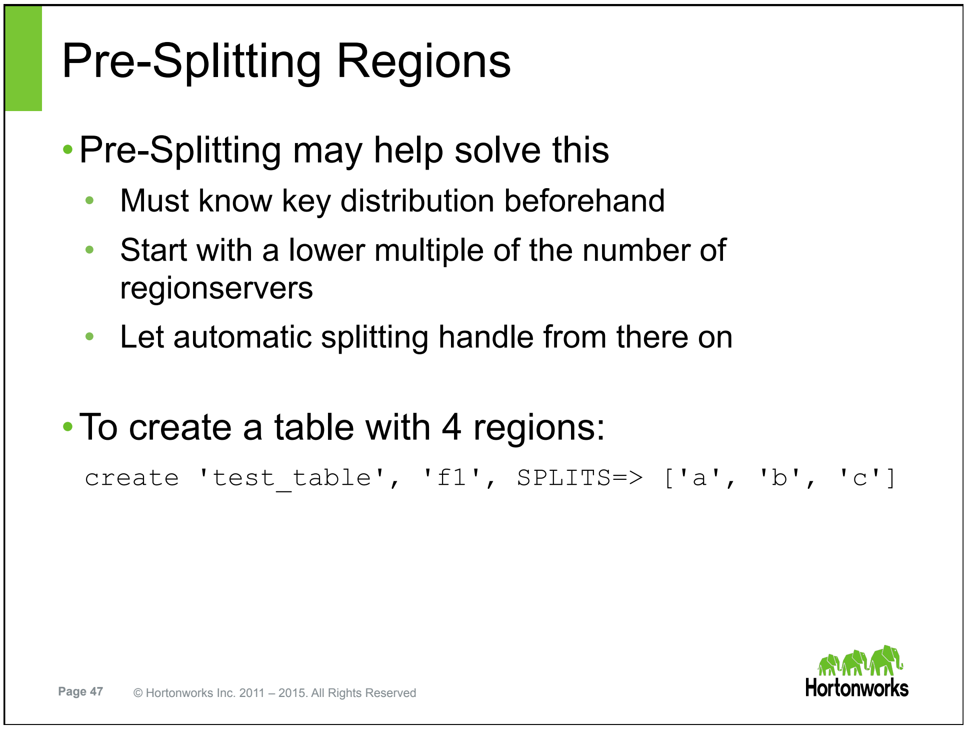
- Number of regions is critical for load balancing

Options:

- Start with a low number
  - Let automated splitting take over
- Create one region
  - Use `IncreasingToUpperBoundRegionSplitPolicy`
- Monitor over time
  - Use manual spltting
- Use pre-splitting
  - Calculate split points

Page 46

Hortonworks



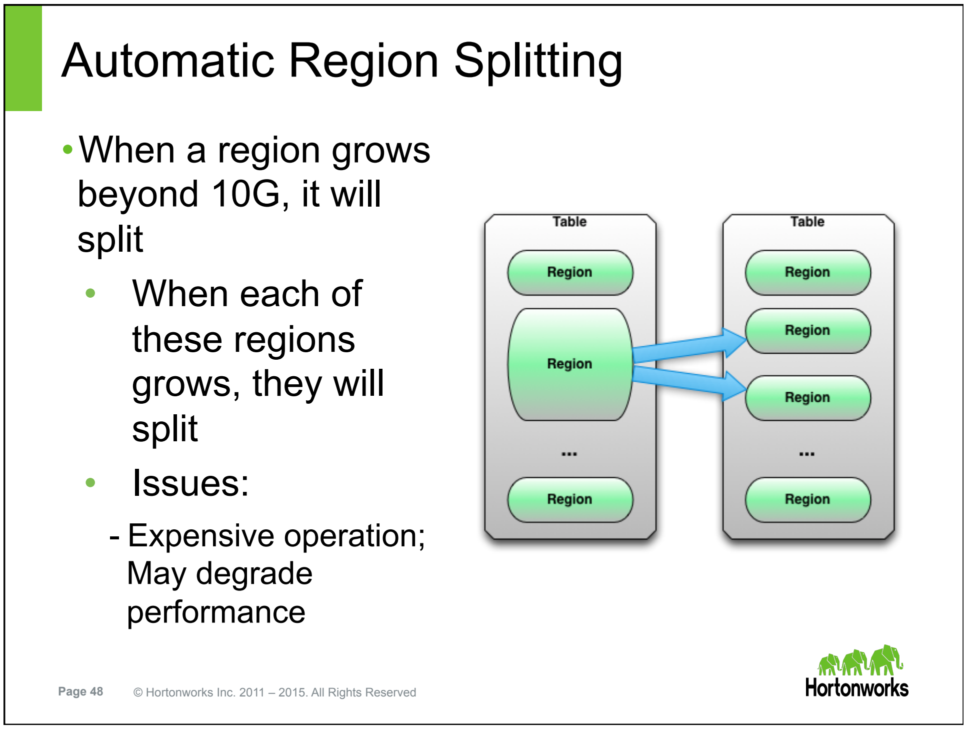
Pre-Splitting Regions

- Pre-Splitting may help solve this
  - Must know key distribution beforehand
  - Start with a lower multiple of the number of regionservers
  - Let automatic splitting handle from there on

- To create a table with 4 regions:

```
create 'test_table', 'f1', SPLITS=> ['a', 'b', 'c']
```

Page 47



Automatic Region Splitting

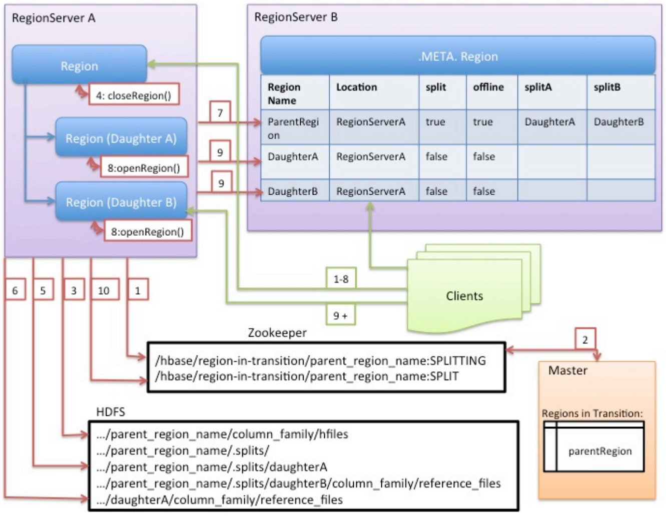
- When a region grows beyond 10G, it will split
  - When each of these regions grows, they will split
  - Issues:
    - Expensive operation; May degrade performance

Page 48

Forced Splits

- Can force a split from the client side
- Examples
  - Split all regions of a table
  - Split a region
- When used
  - If monitoring reveals uneven load distribution
    - Consider manually splitting those regions
  - If initial splits turn out to be suboptimal and automated splits are disabled

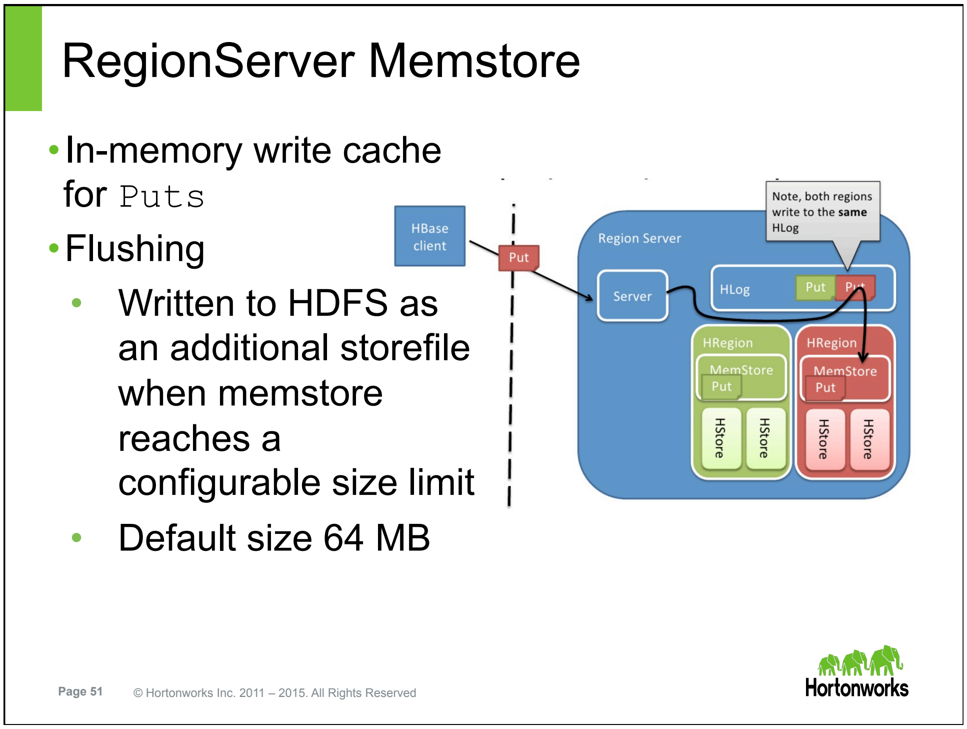
Implementing Region Splits



Region Name	Location	split	offline	splitA	splitB
ParentRegion	RegionServerA	true	true	DaughterA	DaughterB
DaughterA	RegionServerA	false	false		
DaughterB	RegionServerA	false	false		

/hbase/region-in-transition/parent_region_name:SPLITTING
/hbase/region-in-transition/parent_region_name:SPLIT

.../parent_region_name/column_family/hfiles
.../parent_region_name/.splits/
.../parent_region_name/.splits/daughterA
.../parent_region_name/.splits/daughterB/column_family/reference_files
.../daughterA/column_family/reference_files



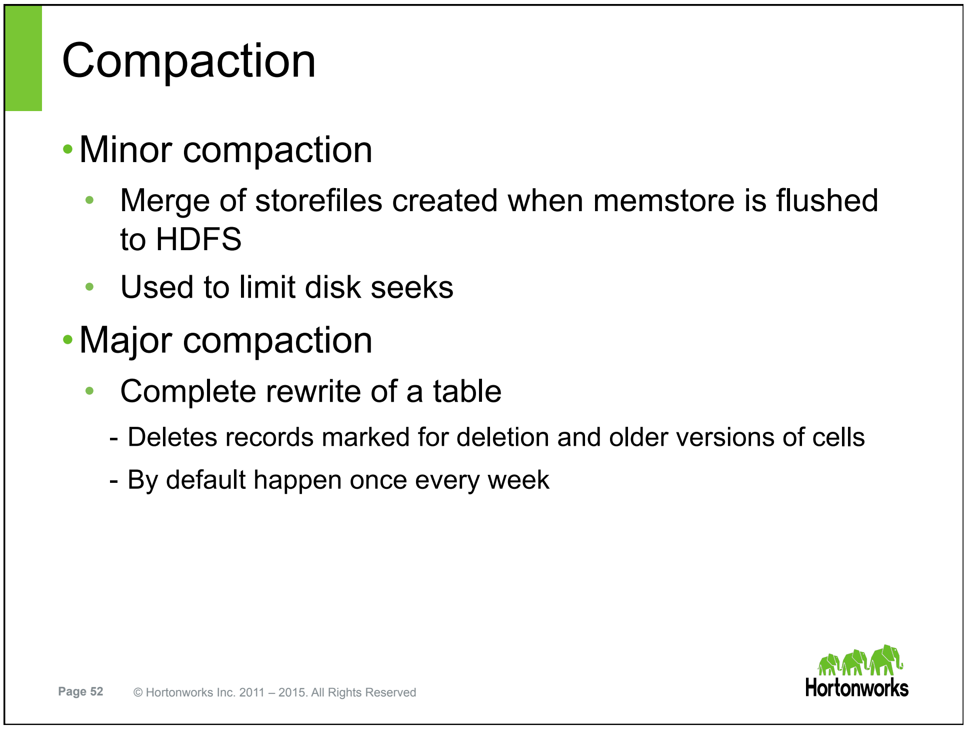
RegionServer Memstore

- In-memory write cache for `Puts`
- Flushing
  - Written to HDFS as an additional storefile when memstore reaches a configurable size limit
  - Default size 64 MB

Note, both regions write to the **same** HLog

Page 51 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

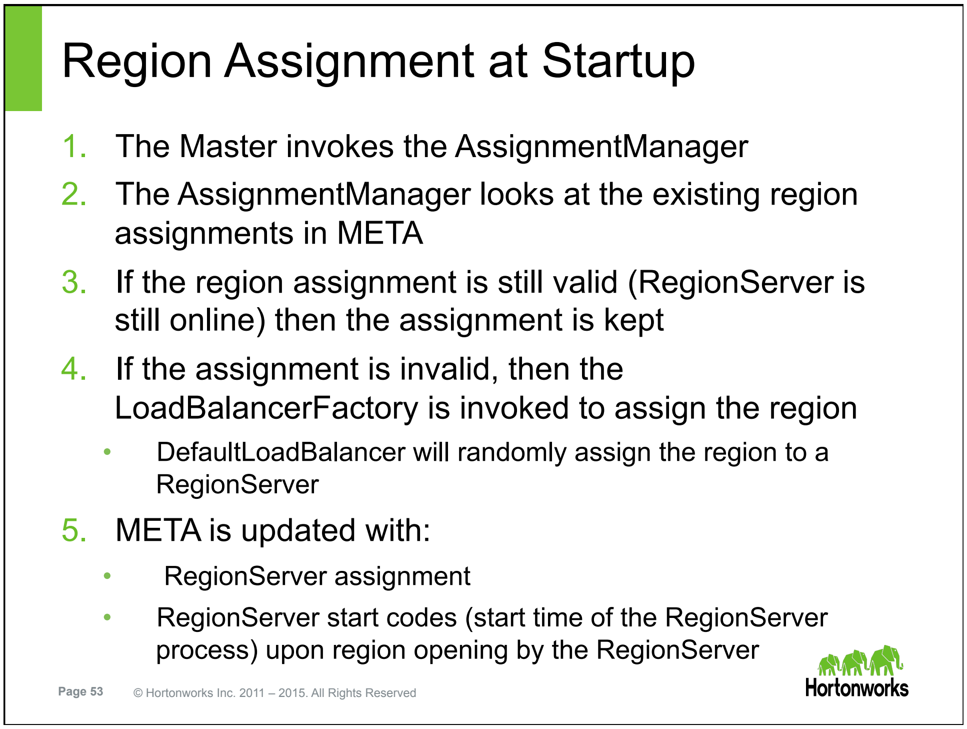


Compaction

- Minor compaction
  - Merge of storefiles created when memstore is flushed to HDFS
  - Used to limit disk seeks
- Major compaction
  - Complete rewrite of a table
    - Deletes records marked for deletion and older versions of cells
    - By default happen once every week

Page 52 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

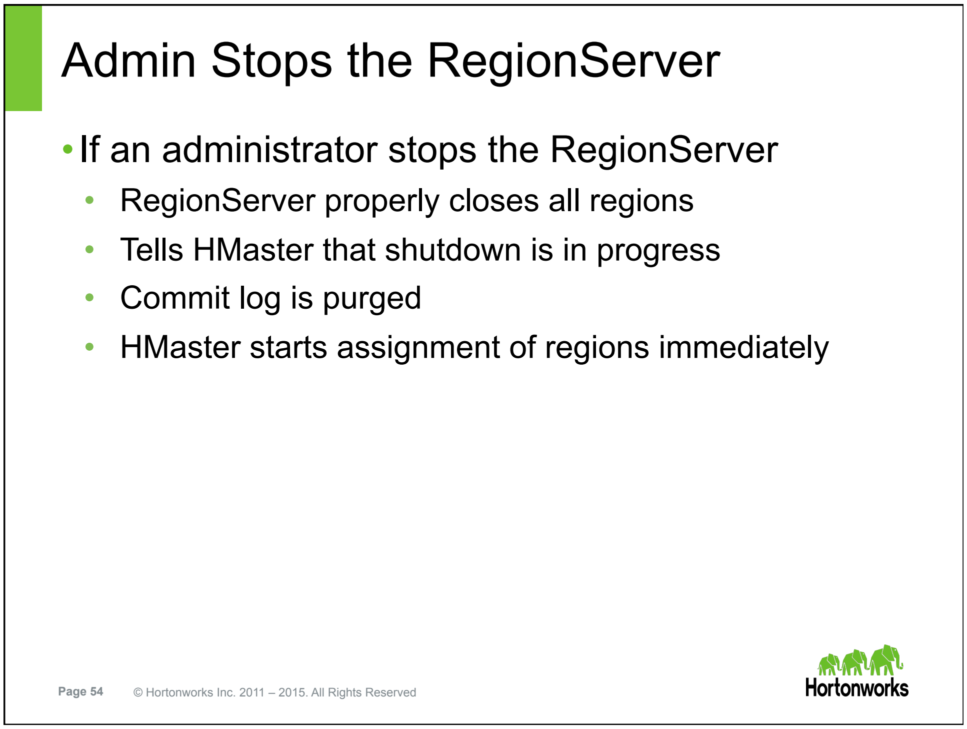
Hortonworks



Region Assignment at Startup

1. The Master invokes the AssignmentManager
2. The AssignmentManager looks at the existing region assignments in META
3. If the region assignment is still valid (RegionServer is still online) then the assignment is kept
4. If the assignment is invalid, then the LoadBalancerFactory is invoked to assign the region
   - DefaultLoadBalancer will randomly assign the region to a RegionServer
5. META is updated with:
   - RegionServer assignment
   - RegionServer start codes (start time of the RegionServer process) upon region opening by the RegionServer

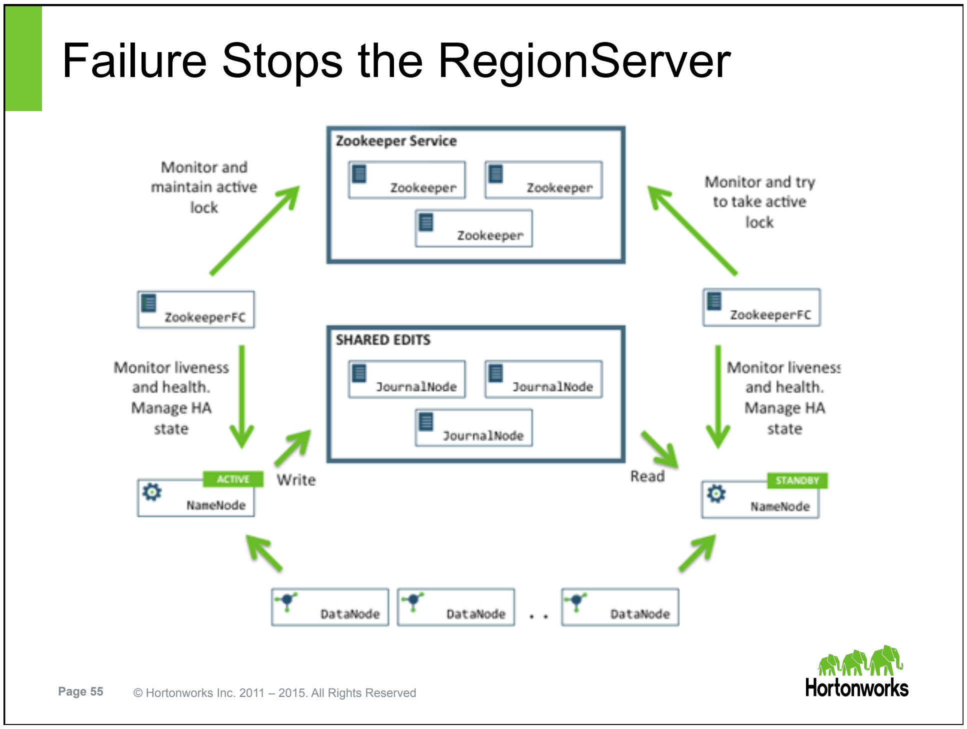
Page 53 Hortonworks



Admin Stops the RegionServer

- If an administrator stops the RegionServer
  - RegionServer properly closes all regions
  - Tells HMaster that shutdown is in progress
  - Commit log is purged
  - HMaster starts assignment of regions immediately

Page 54 Hortonworks



Failure Stops the RegionServer

Zookeeper Service

Zookeeper

Zookeeper

Zookeeper

Monitor and maintain active lock

Monitor and try to take active lock

ZookeeperFC

ZookeeperFC

SHARED EDITS

JournalNode

JournalNode

JournalNode

Monitor liveness and health. Manage HA state

Monitor liveness and health. Manage HA state

ACTIVE

NameNode

Write

Read

STANDBY

NameNode

DataNode

DataNode

. .

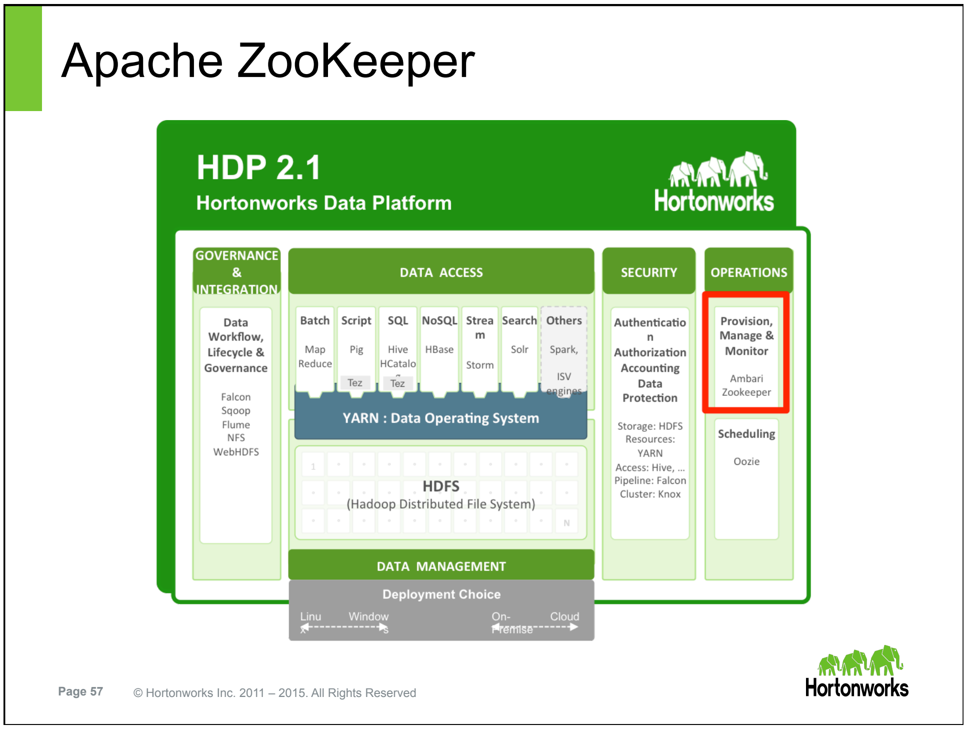
DataNode

Page 55

Hortonworks

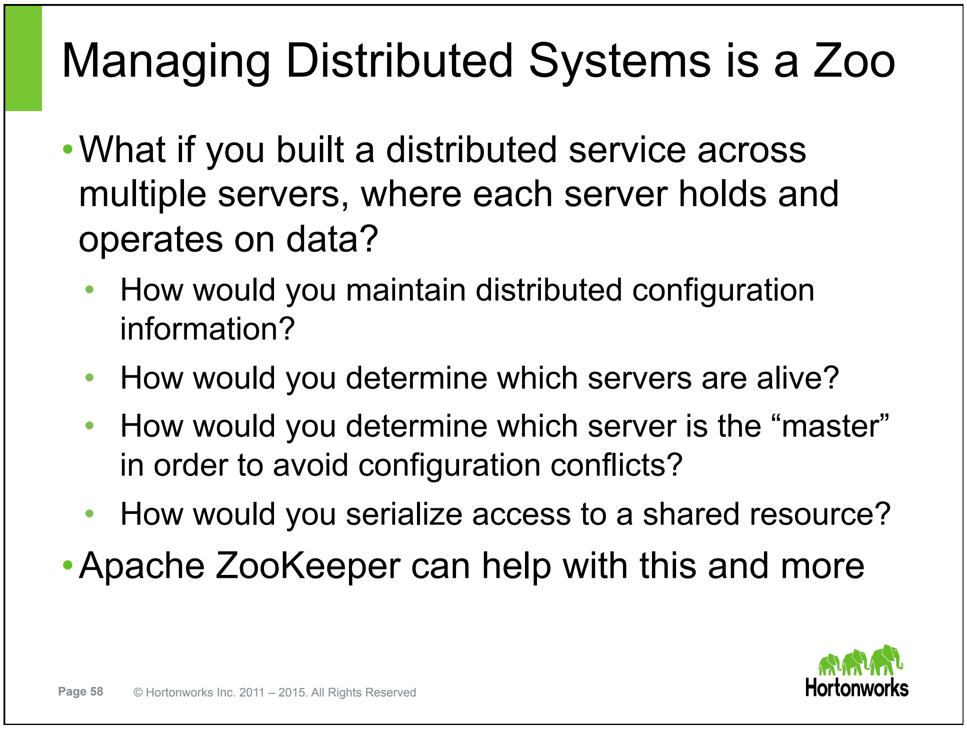
Zookeeper

Hortonworks



Apache ZooKeeper

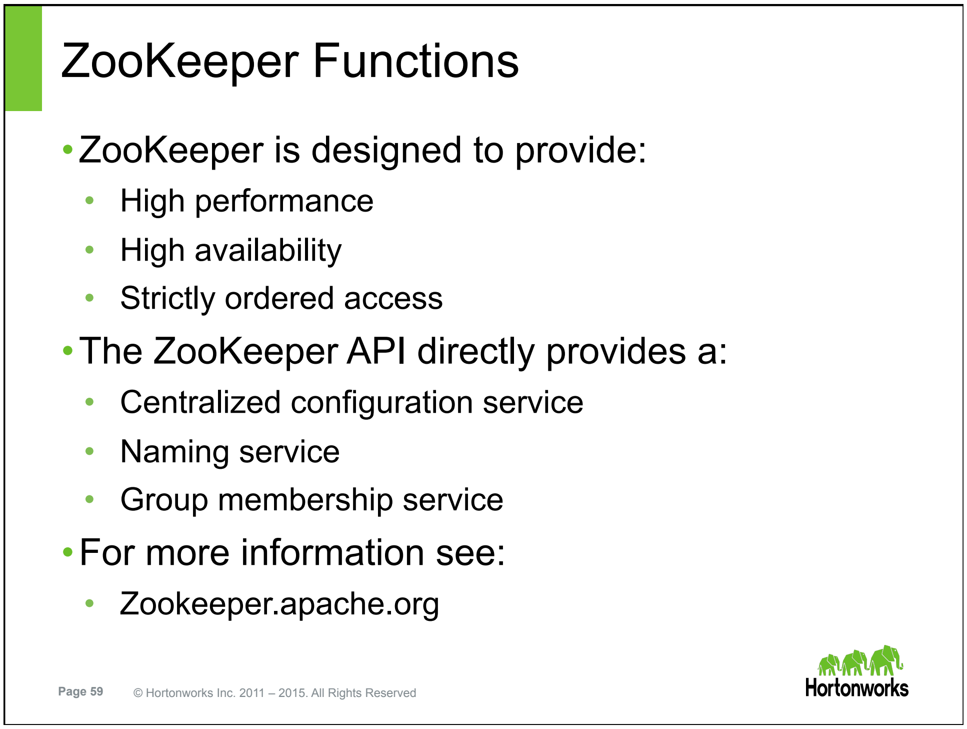
Page 57



Managing Distributed Systems is a Zoo

- What if you built a distributed service across multiple servers, where each server holds and operates on data?
  - How would you maintain distributed configuration information?
  - How would you determine which servers are alive?
  - How would you determine which server is the "master" in order to avoid configuration conflicts?
  - How would you serialize access to a shared resource?
- Apache ZooKeeper can help with this and more

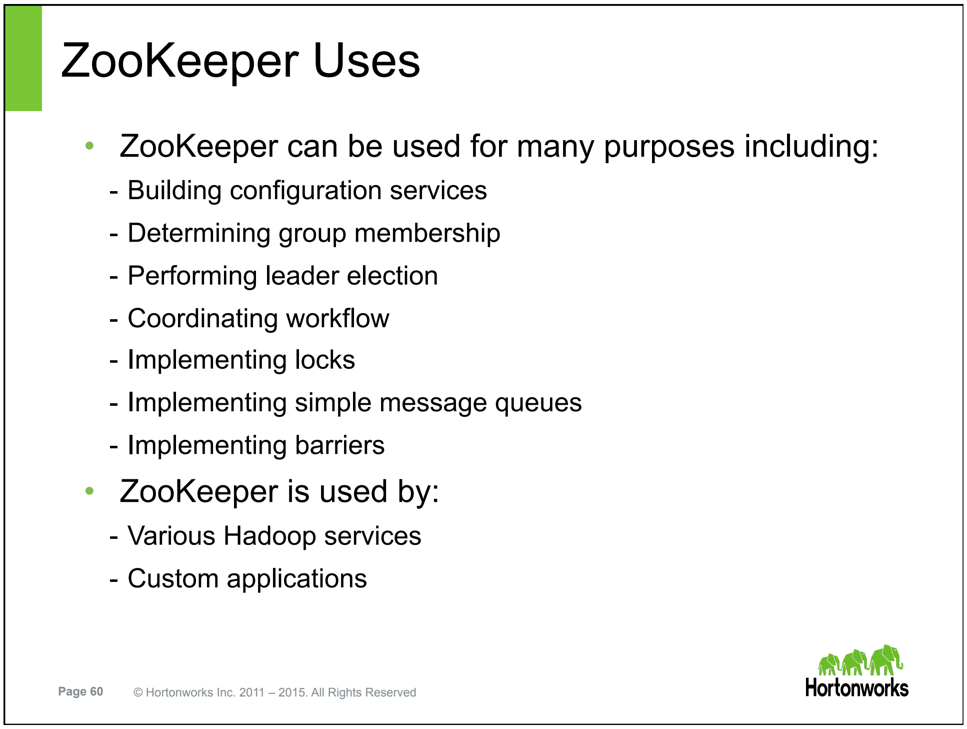
Page 58



ZooKeeper Functions

- ZooKeeper is designed to provide:
  - High performance
  - High availability
  - Strictly ordered access
- The ZooKeeper API directly provides a:
  - Centralized configuration service
  - Naming service
  - Group membership service
- For more information see:
  - Zookeeper.apache.org

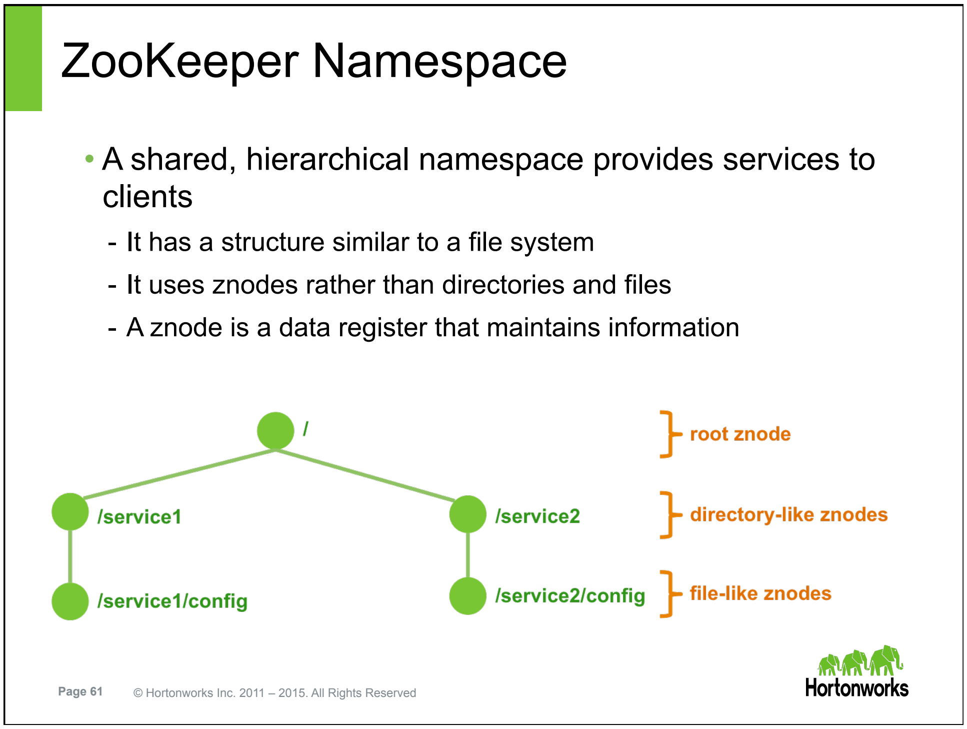
Page 59



ZooKeeper Uses

- ZooKeeper can be used for many purposes including:
  - Building configuration services
  - Determining group membership
  - Performing leader election
  - Coordinating workflow
  - Implementing locks
  - Implementing simple message queues
  - Implementing barriers
- ZooKeeper is used by:
  - Various Hadoop services
  - Custom applications

Page 60



ZooKeeper Namespace

- A shared, hierarchical namespace provides services to clients
  - It has a structure similar to a file system
  - It uses znodes rather than directories and files
  - A znode is a data register that maintains information

/

/service1

/service2

/service1/config

/service2/config

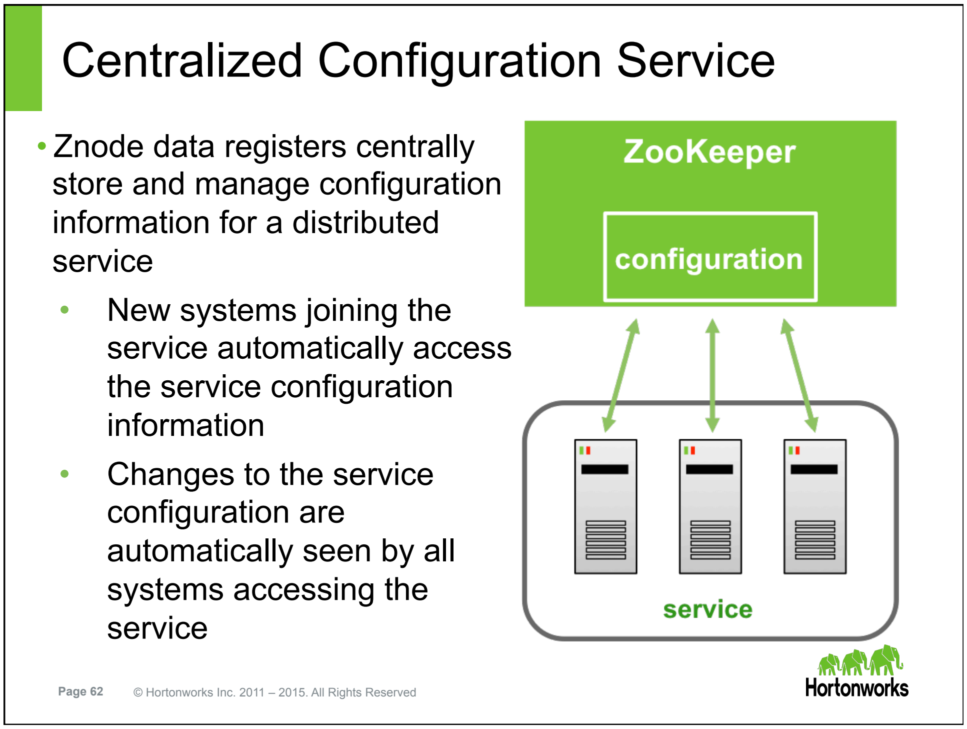
root znode

directory-like znodes

file-like znodes

Page 61 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Centralized Configuration Service

- Znode data registers centrally store and manage configuration information for a distributed service
  - New systems joining the service automatically access the service configuration information
  - Changes to the service configuration are automatically seen by all systems accessing the service

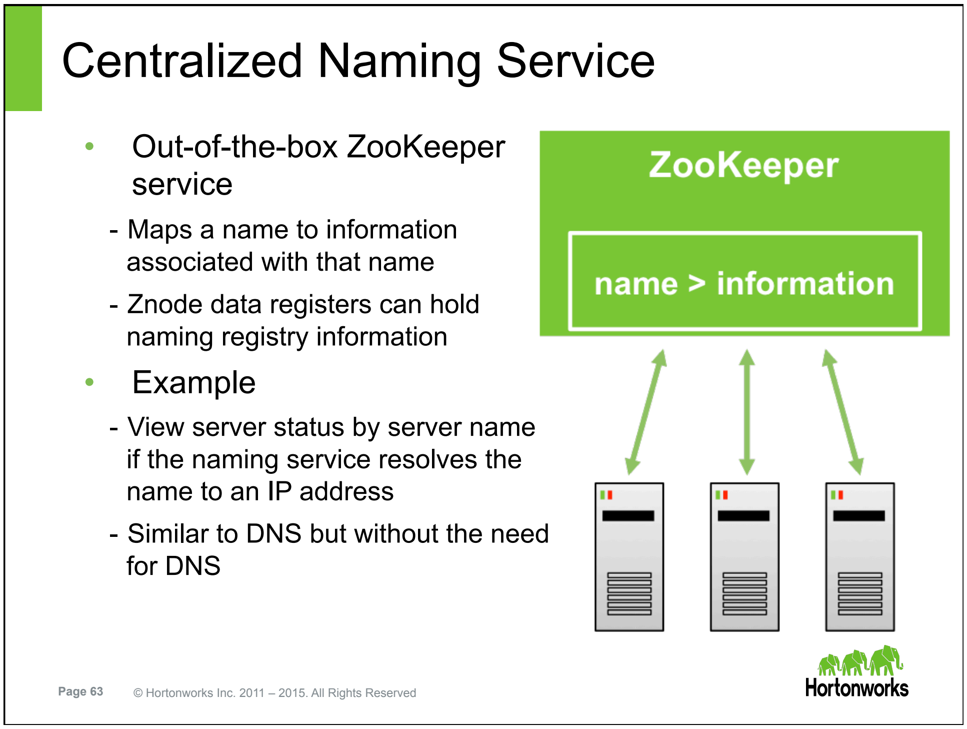
ZooKeeper

configuration

service

Page 62 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

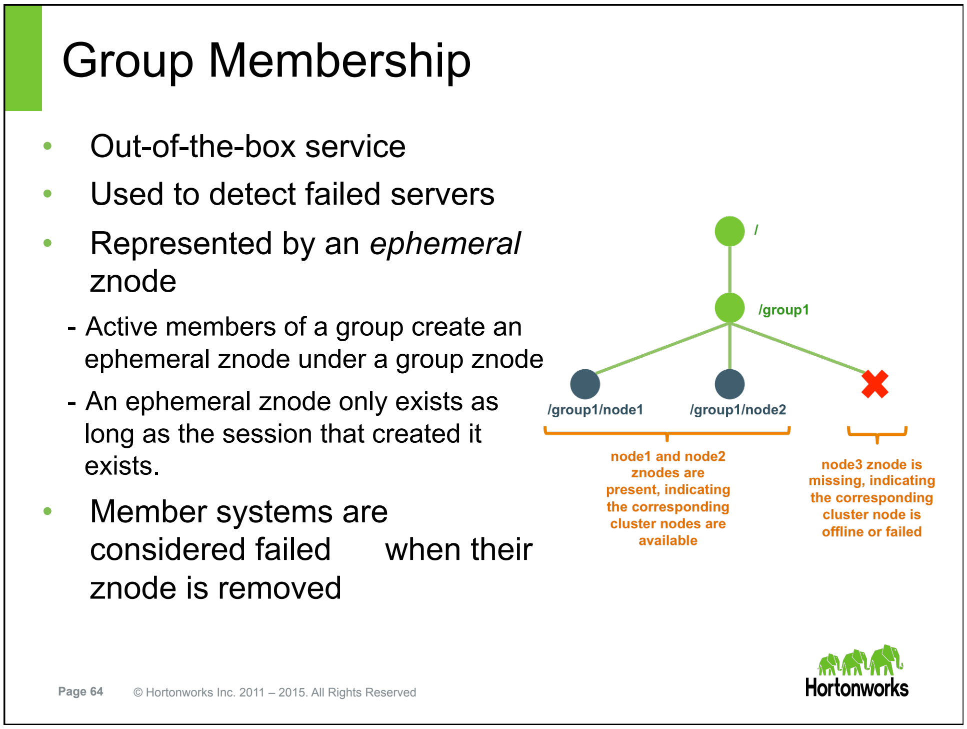
Hortonworks



Centralized Naming Service

- Out-of-the-box ZooKeeper service
  - Maps a name to information associated with that name
  - Znode data registers can hold naming registry information
- Example
  - View server status by server name if the naming service resolves the name to an IP address
  - Similar to DNS but without the need for DNS

Page 63

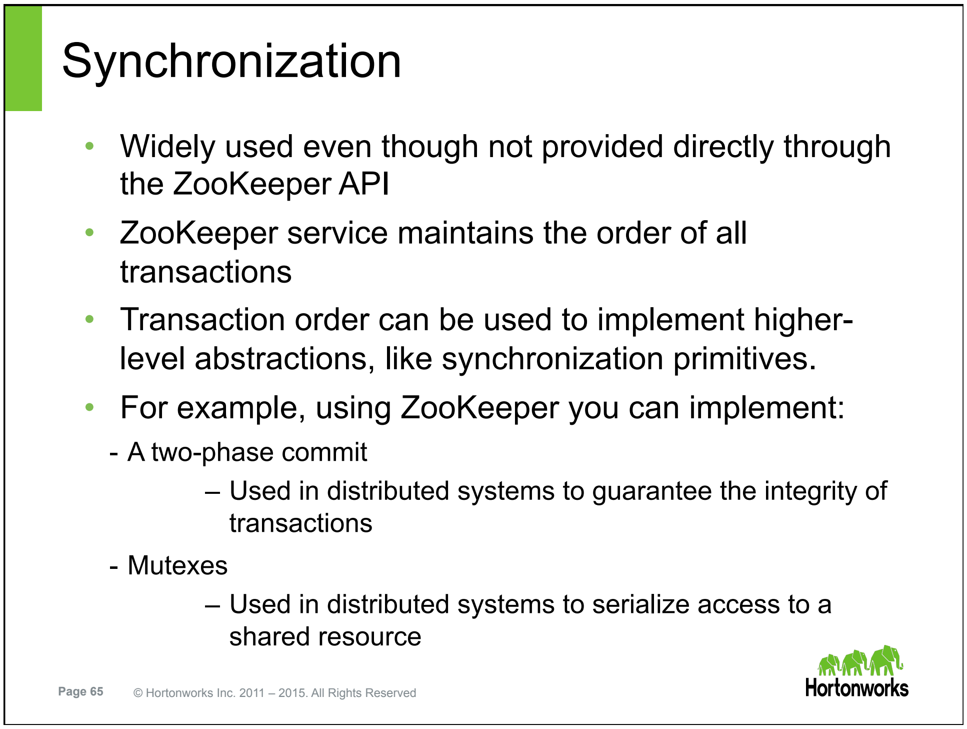


Group Membership

- Out-of-the-box service
- Used to detect failed servers
- Represented by an *ephemeral* znode
  - Active members of a group create an ephemeral znode under a group znode
  - An ephemeral znode only exists as long as the session that created it exists.
- Member systems are considered failed when their znode is removed

/
/group1
/group1/node1
/group1/node2
node1 and node2 znodes are present, indicating the corresponding cluster nodes are available
node3 znode is missing, indicating the corresponding cluster node is offline or failed

Page 64

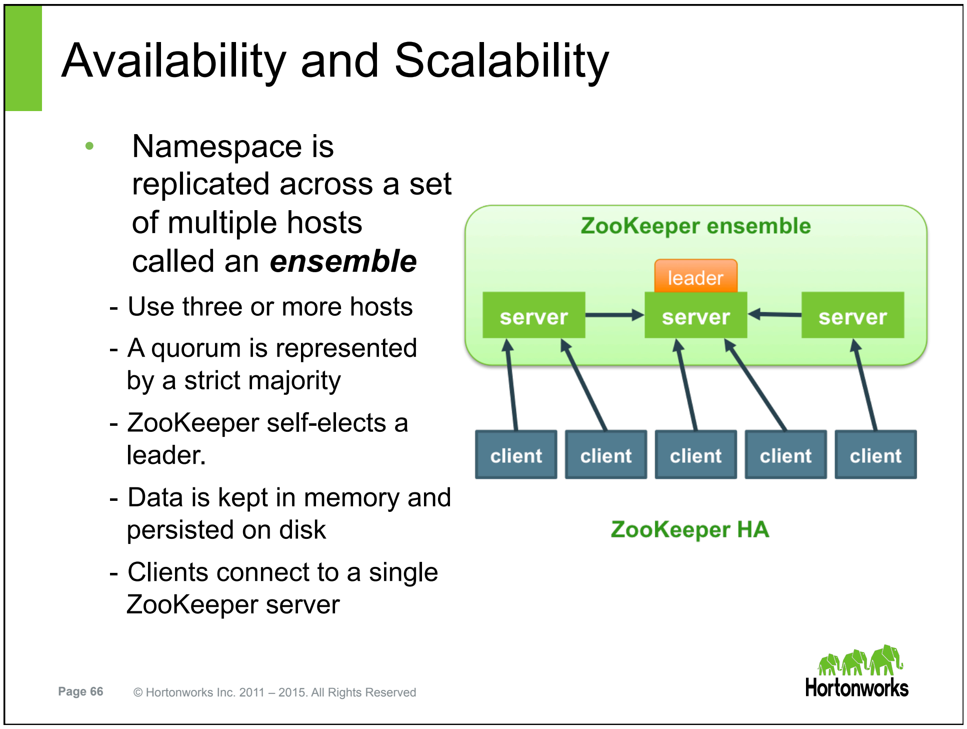


Synchronization

- Widely used even though not provided directly through the ZooKeeper API
- ZooKeeper service maintains the order of all transactions
- Transaction order can be used to implement higher-level abstractions, like synchronization primitives.
- For example, using ZooKeeper you can implement:
  - A two-phase commit
    - Used in distributed systems to guarantee the integrity of transactions
  - Mutexes
    - Used in distributed systems to serialize access to a shared resource

Page 65

Hortonworks



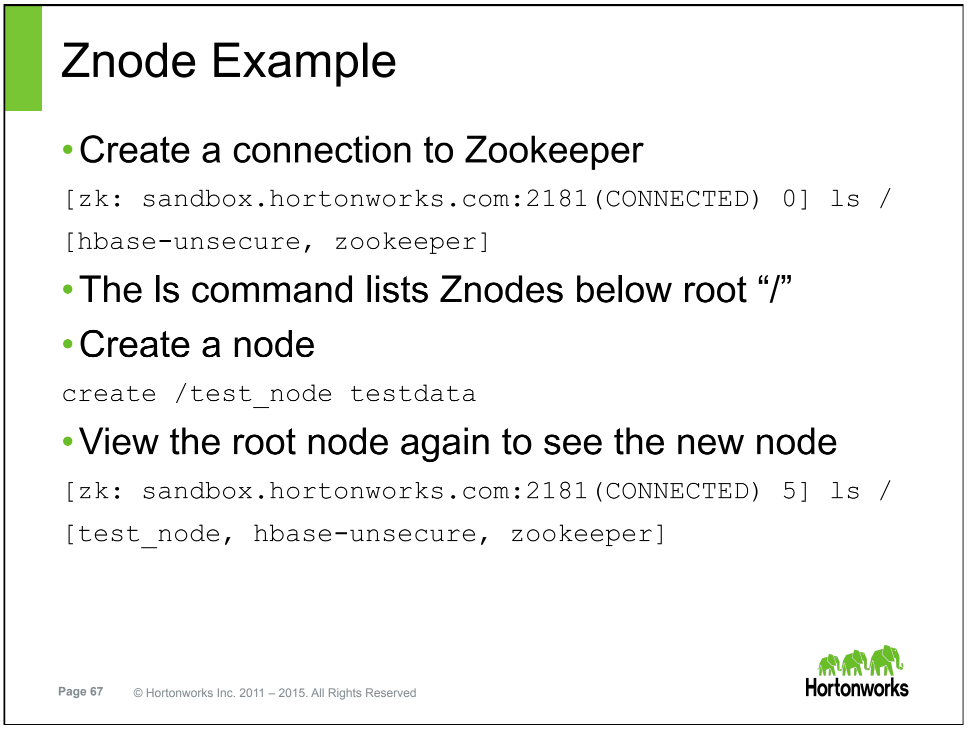
Availability and Scalability

- Namespace is replicated across a set of multiple hosts called an ***ensemble***
  - Use three or more hosts
  - A quorum is represented by a strict majority
  - ZooKeeper self-elects a leader.
  - Data is kept in memory and persisted on disk
  - Clients connect to a single ZooKeeper server

ZooKeeper HA

Page 66

Hortonworks



Znode Example

- Create a connection to Zookeeper

```
[zk: sandbox.hortonworks.com:2181(CONNECTED) 0] ls /
[hbase-unsecure, zookeeper]
```

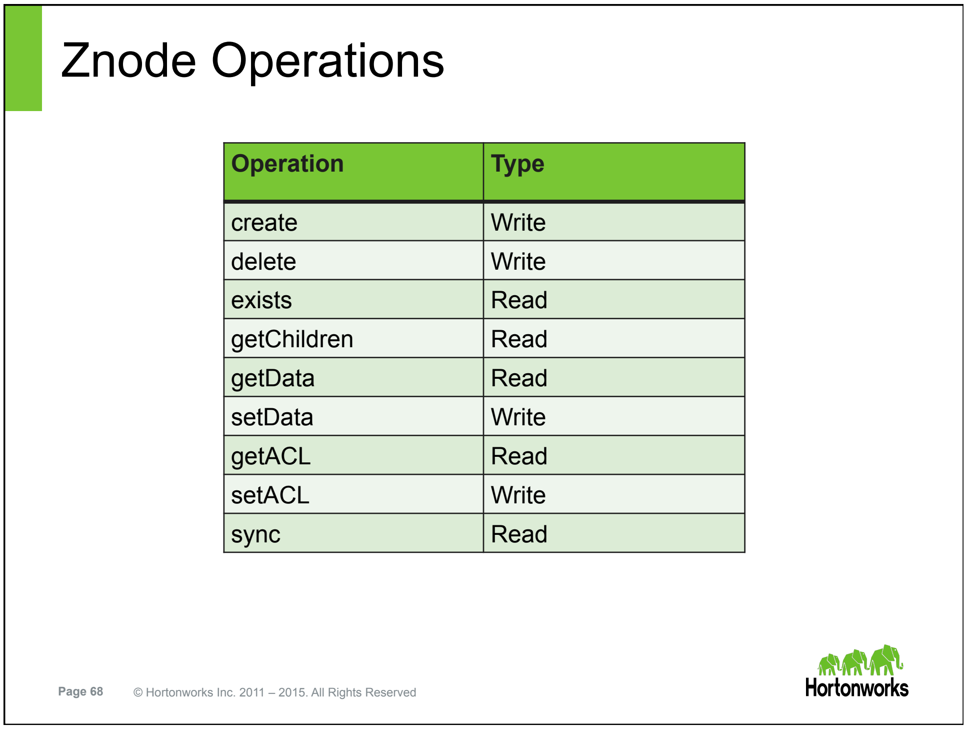
- The ls command lists Znodes below root "/"
- Create a node

```
create /test_node testdata
```

- View the root node again to see the new node

```
[zk: sandbox.hortonworks.com:2181(CONNECTED) 5] ls /
[test_node, hbase-unsecure, zookeeper]
```

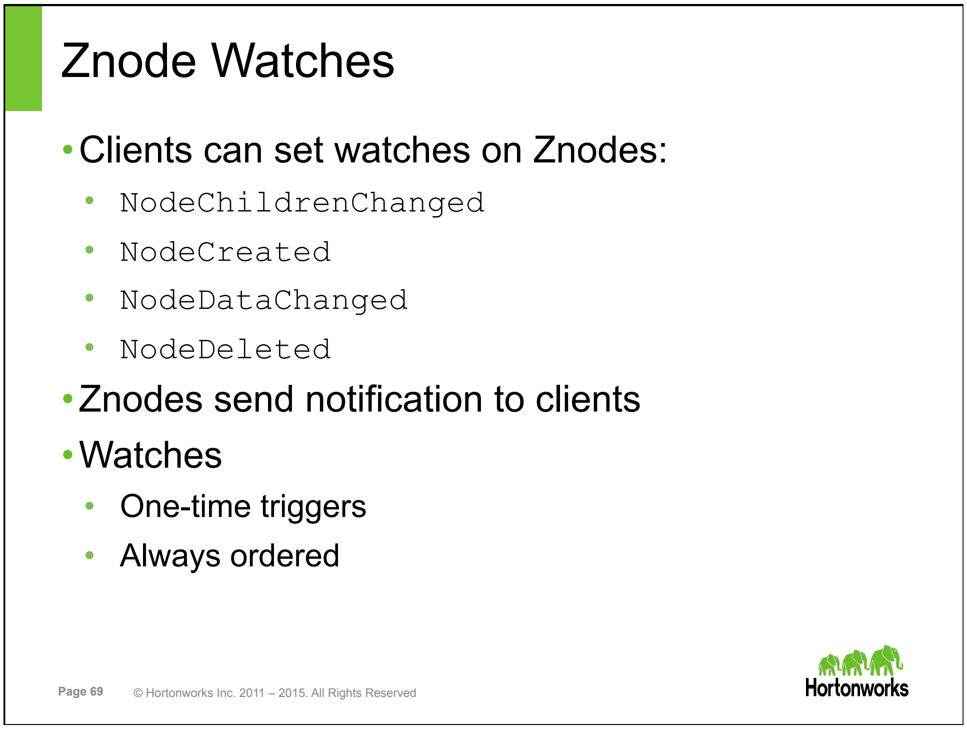
Page 67



Znode Operations

Operation	Type
create	Write
delete	Write
exists	Read
getChildren	Read
getData	Read
setData	Write
getACL	Read
setACL	Write
sync	Read

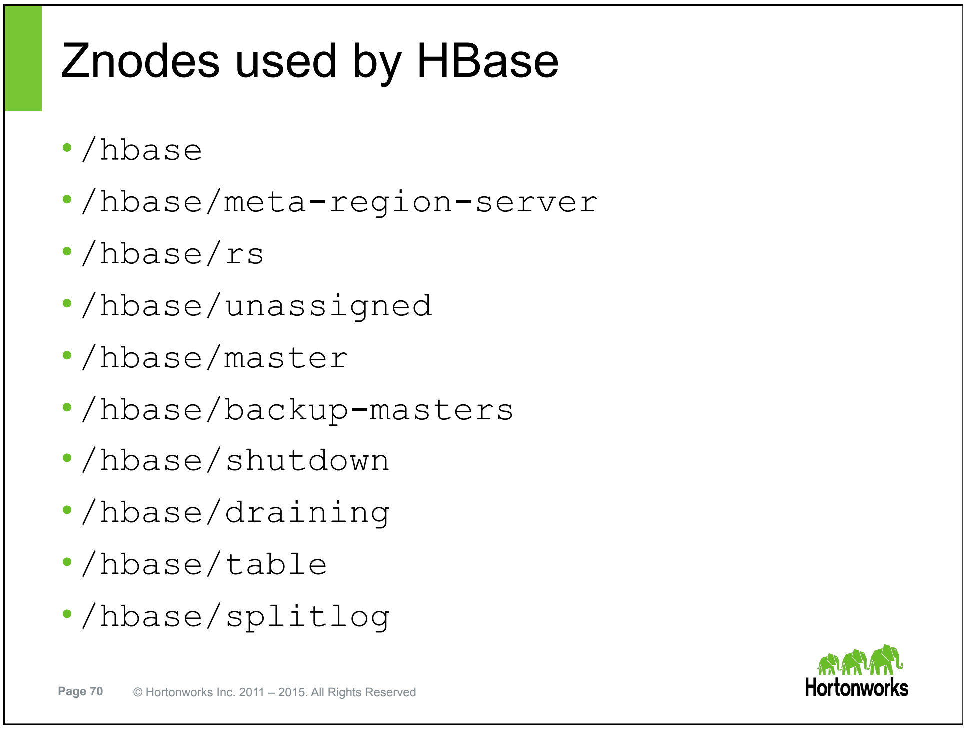
Page 68



Znode Watches

- Clients can set watches on Znodes:
  - `NodeChildrenChanged`
  - `NodeCreated`
  - `NodeDataChanged`
  - `NodeDeleted`
- Znodes send notification to clients
- Watches
  - One-time triggers
  - Always ordered

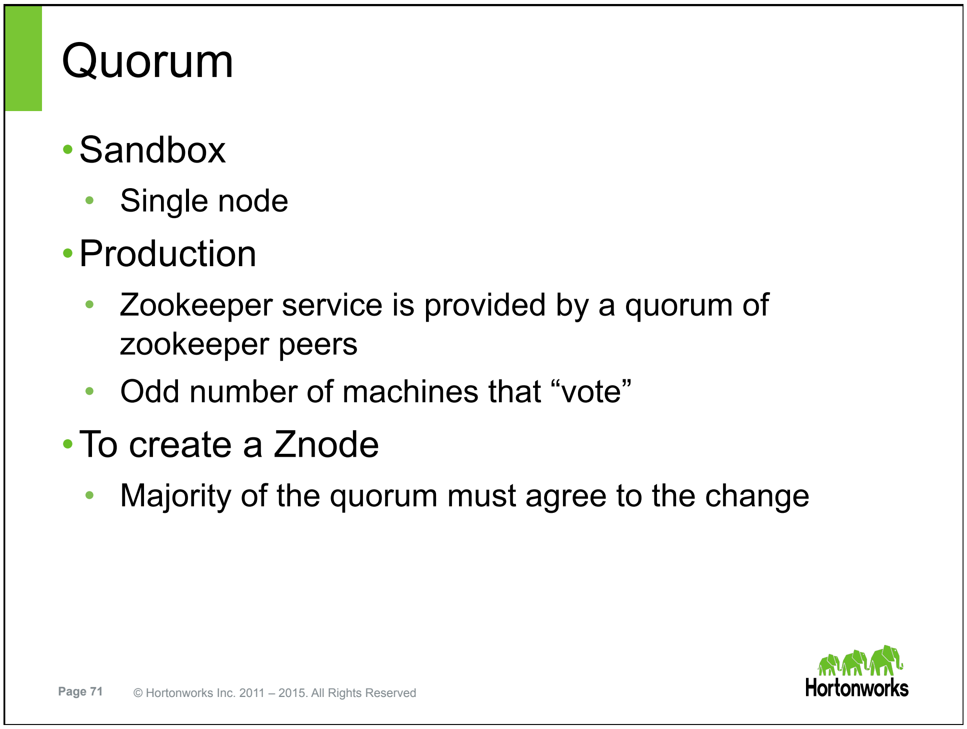
Page 69



Znodes used by HBase

- `/hbase`
- `/hbase/meta-region-server`
- `/hbase/rs`
- `/hbase/unassigned`
- `/hbase/master`
- `/hbase/backup-masters`
- `/hbase/shutdown`
- `/hbase/draining`
- `/hbase/table`
- `/hbase/splitlog`

Page 70

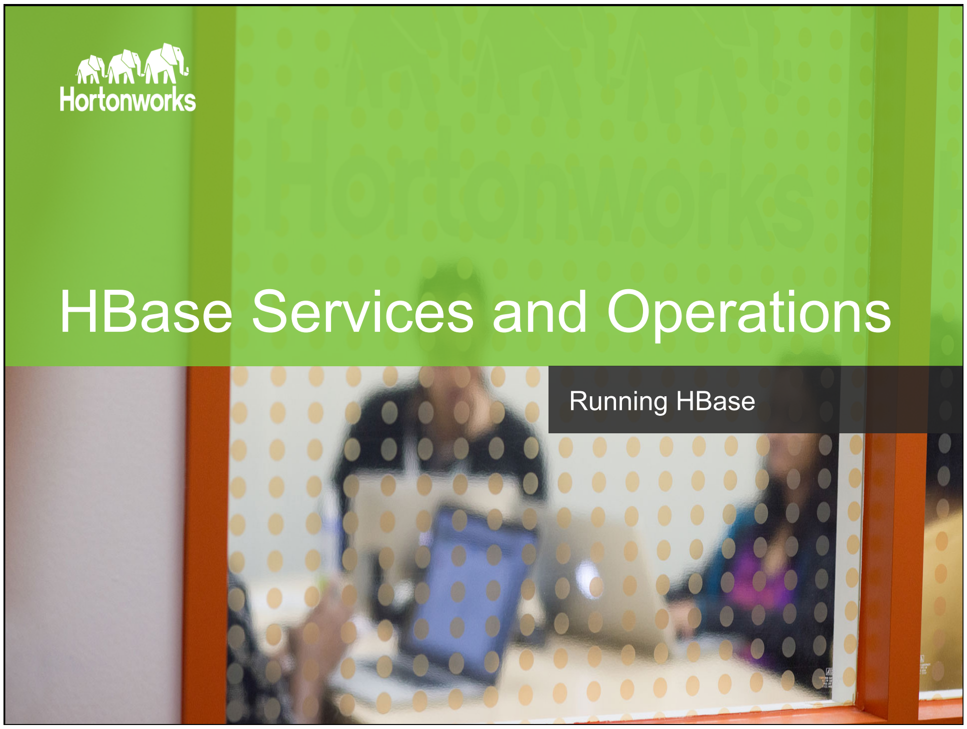


Quorum

- Sandbox
  - Single node
- Production
  - Zookeeper service is provided by a quorum of zookeeper peers
  - Odd number of machines that “vote”
- To create a Znode
  - Majority of the quorum must agree to the change

Page 71 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Lab: Zookeeper



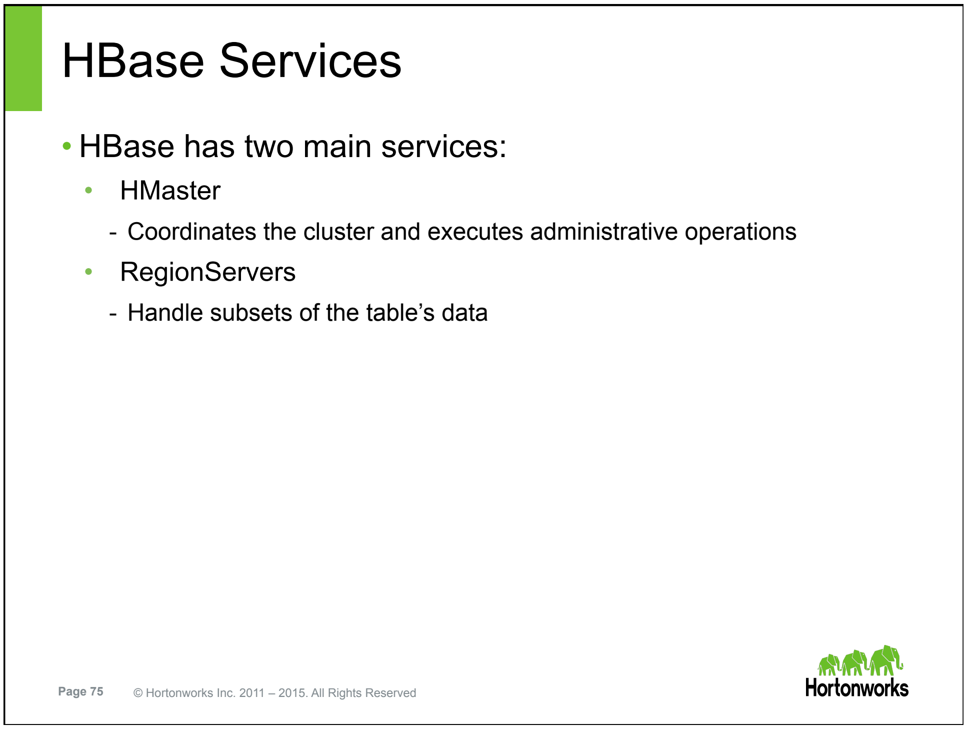
Hortonworks

HBase Services and Operations

Running HBase

HBase Services & HA

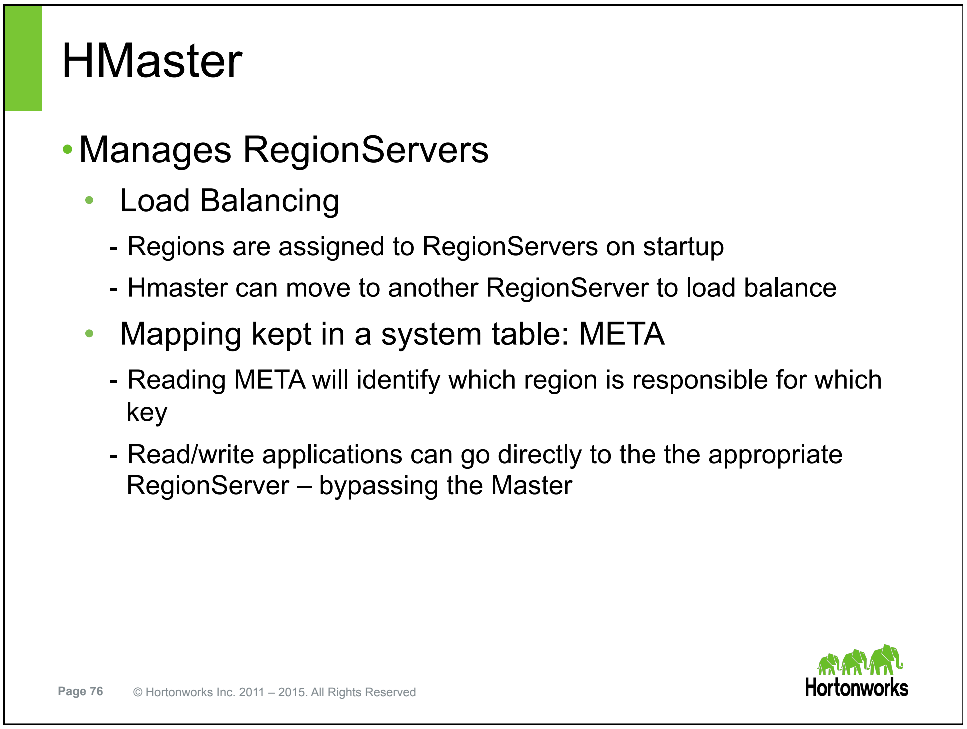
Hortonworks



HBase Services

- HBase has two main services:
  - HMaster
    - Coordinates the cluster and executes administrative operations
  - RegionServers
    - Handle subsets of the table's data

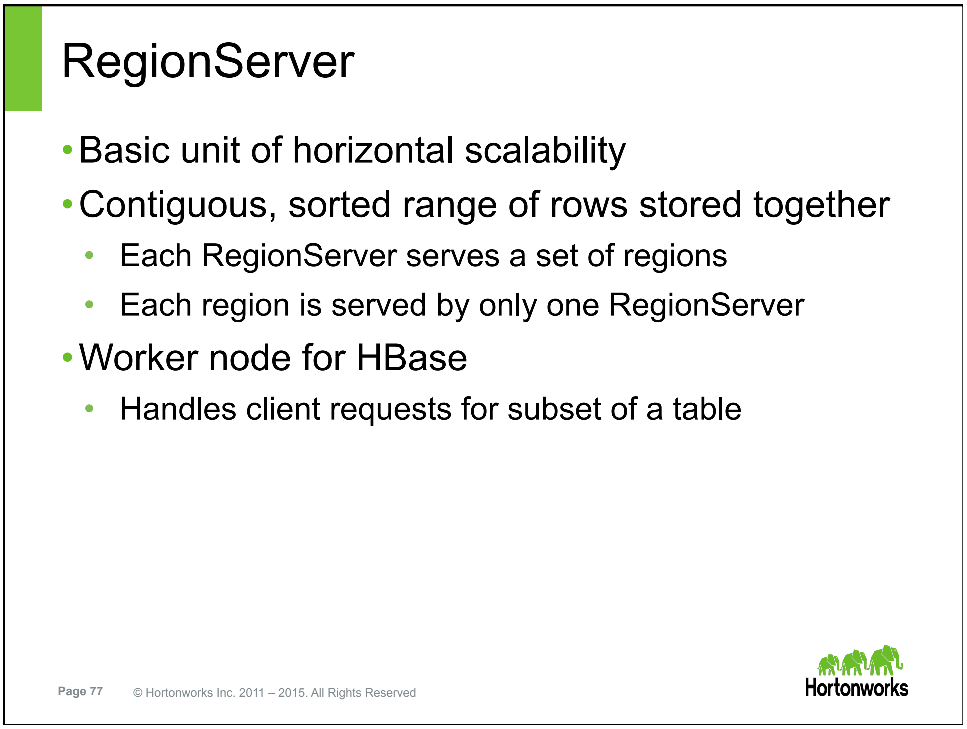
Page 75



HMaster

- Manages RegionServers
  - Load Balancing
    - Regions are assigned to RegionServers on startup
    - Hmaster can move to another RegionServer to load balance
  - Mapping kept in a system table: META
    - Reading META will identify which region is responsible for which key
    - Read/write applications can go directly to the the appropriate RegionServer – bypassing the Master

Page 76

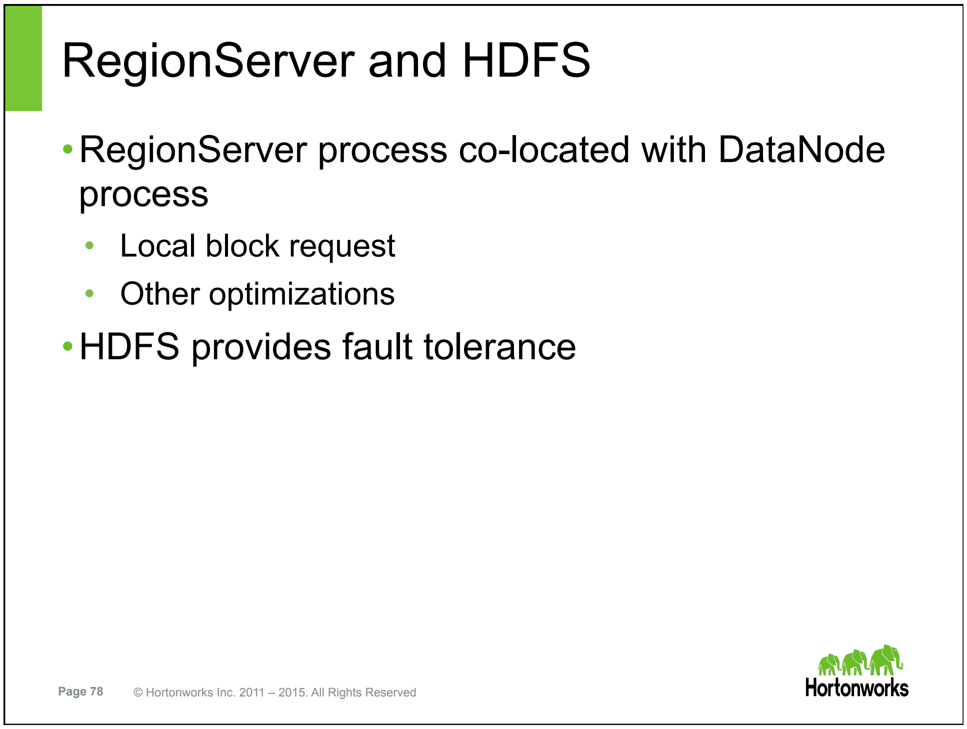


RegionServer

- Basic unit of horizontal scalability
- Contiguous, sorted range of rows stored together
  - Each RegionServer serves a set of regions
  - Each region is served by only one RegionServer
- Worker node for HBase
  - Handles client requests for subset of a table

Page 77

Hortonworks

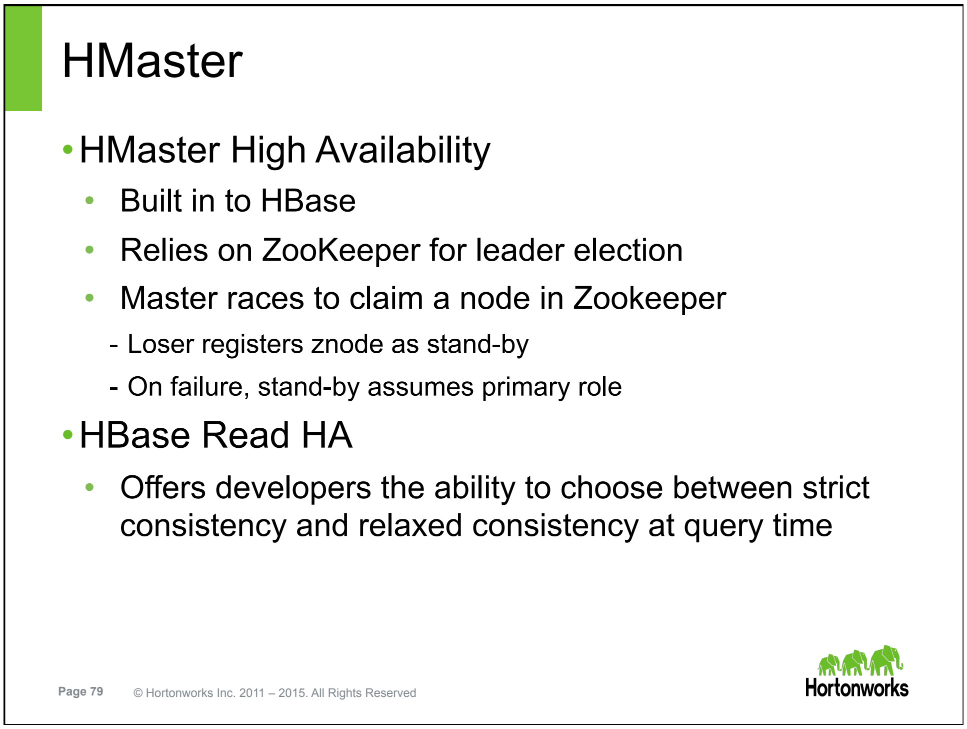


RegionServer and HDFS

- RegionServer process co-located with DataNode process
  - Local block request
  - Other optimizations
- HDFS provides fault tolerance

Page 78

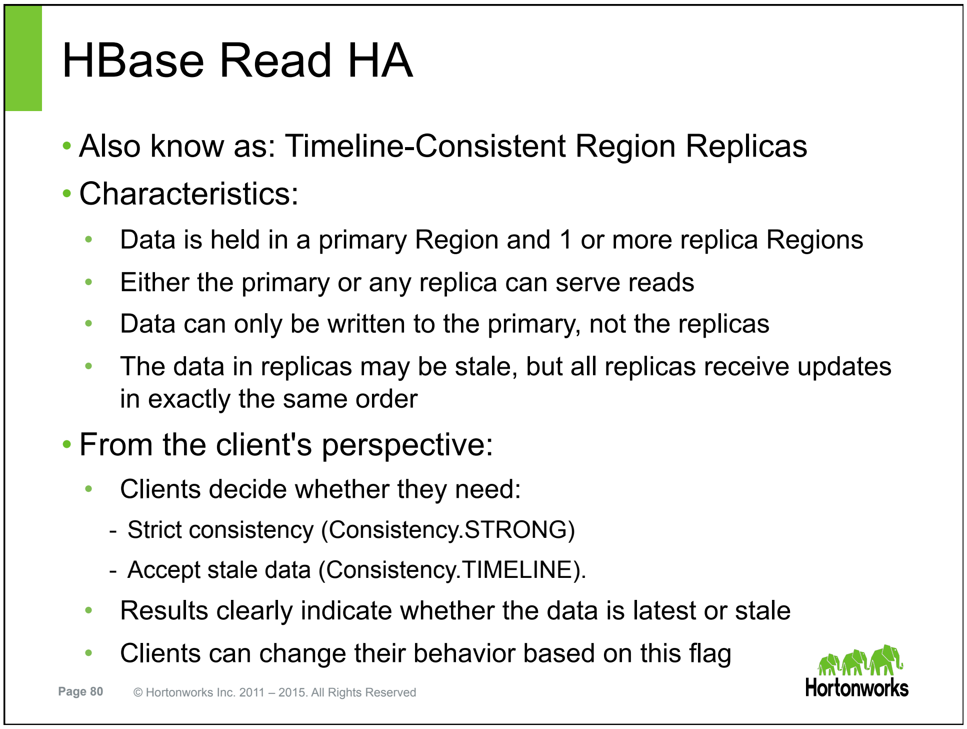
Hortonworks



HMaster

- HMaster High Availability
  - Built in to HBase
  - Relies on ZooKeeper for leader election
  - Master races to claim a node in Zookeeper
    - Loser registers znode as stand-by
    - On failure, stand-by assumes primary role
- HBase Read HA
  - Offers developers the ability to choose between strict consistency and relaxed consistency at query time

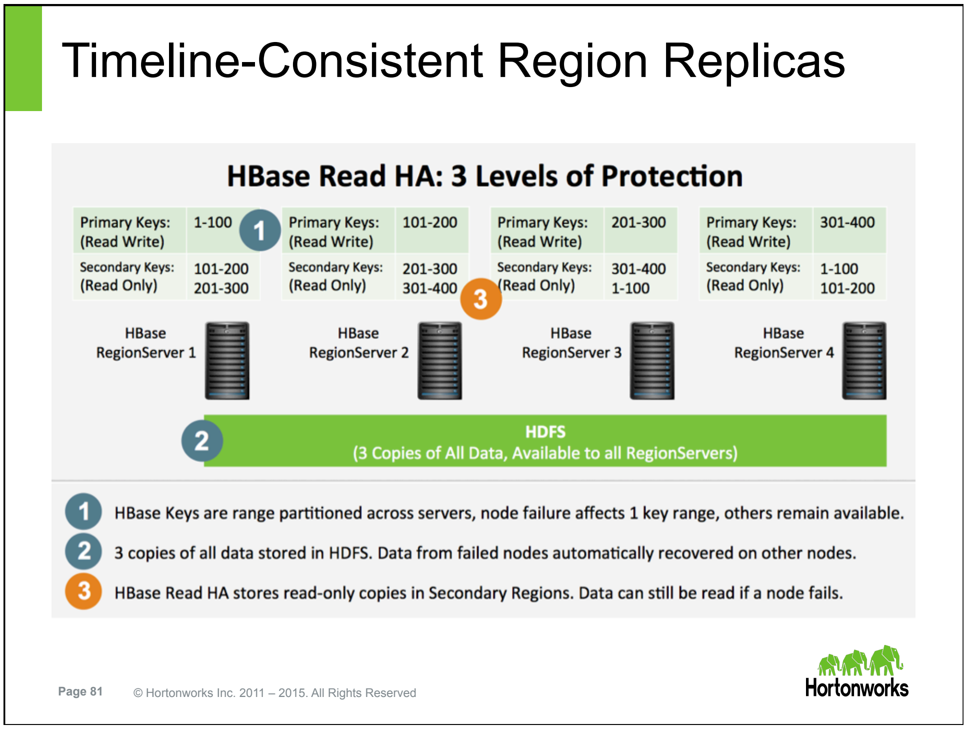
Page 79 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



HBase Read HA

- Also know as: Timeline-Consistent Region Replicas
- Characteristics:
  - Data is held in a primary Region and 1 or more replica Regions
  - Either the primary or any replica can serve reads
  - Data can only be written to the primary, not the replicas
  - The data in replicas may be stale, but all replicas receive updates in exactly the same order
- From the client's perspective:
  - Clients decide whether they need:
    - Strict consistency (Consistency.STRONG)
    - Accept stale data (Consistency.TIMELINE).
  - Results clearly indicate whether the data is latest or stale
  - Clients can change their behavior based on this flag

Page 80 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



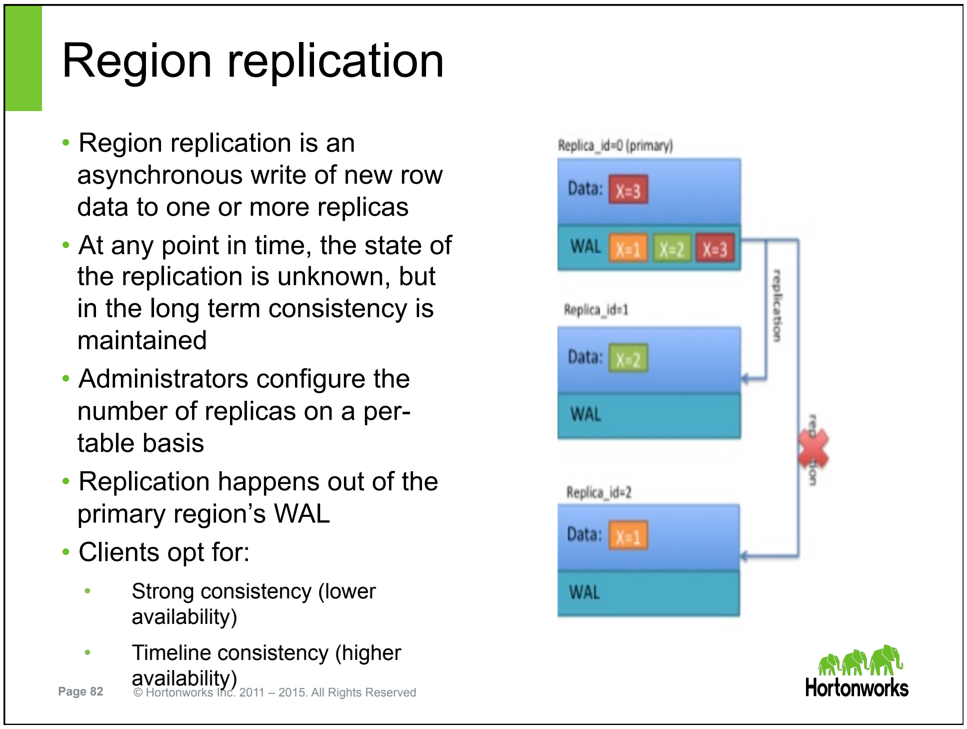
Timeline-Consistent Region Replicas

HBase Read HA: 3 Levels of Protection

HBase RegionServer 1		HBase RegionServer 2		HBase RegionServer 3		HBase RegionServer 4	
Primary Keys: (Read Write)	1-100	Primary Keys: (Read Write)	101-200	Primary Keys: (Read Write)	201-300	Primary Keys: (Read Write)	301-400
Secondary Keys: (Read Only)	101-200 201-300	Secondary Keys: (Read Only)	201-300 301-400	Secondary Keys: (Read Only)	301-400 1-100	Secondary Keys: (Read Only)	1-100 101-200

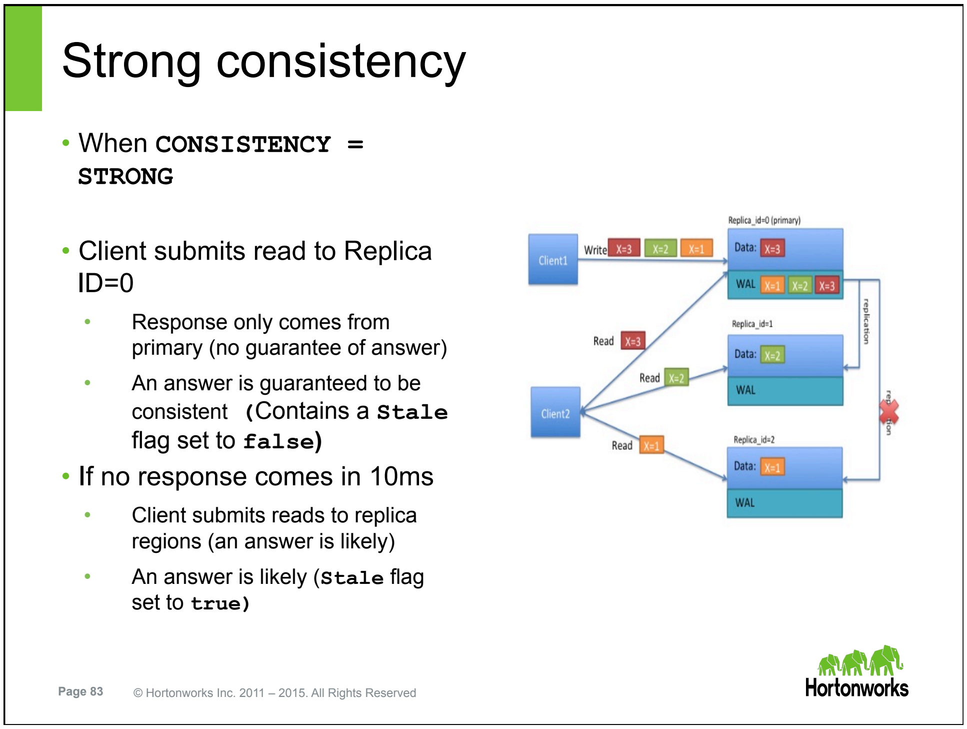
HDFS
(3 Copies of All Data, Available to all RegionServers)

1. HBase Keys are range partitioned across servers, node failure affects 1 key range, others remain available.
2. 3 copies of all data stored in HDFS. Data from failed nodes automatically recovered on other nodes.
3. HBase Read HA stores read-only copies in Secondary Regions. Data can still be read if a node fails.



Region replication

- Region replication is an asynchronous write of new row data to one or more replicas
- At any point in time, the state of the replication is unknown, but in the long term consistency is maintained
- Administrators configure the number of replicas on a per-table basis
- Replication happens out of the primary region's WAL
- Clients opt for:
  - Strong consistency (lower availability)
  - Timeline consistency (higher availability)

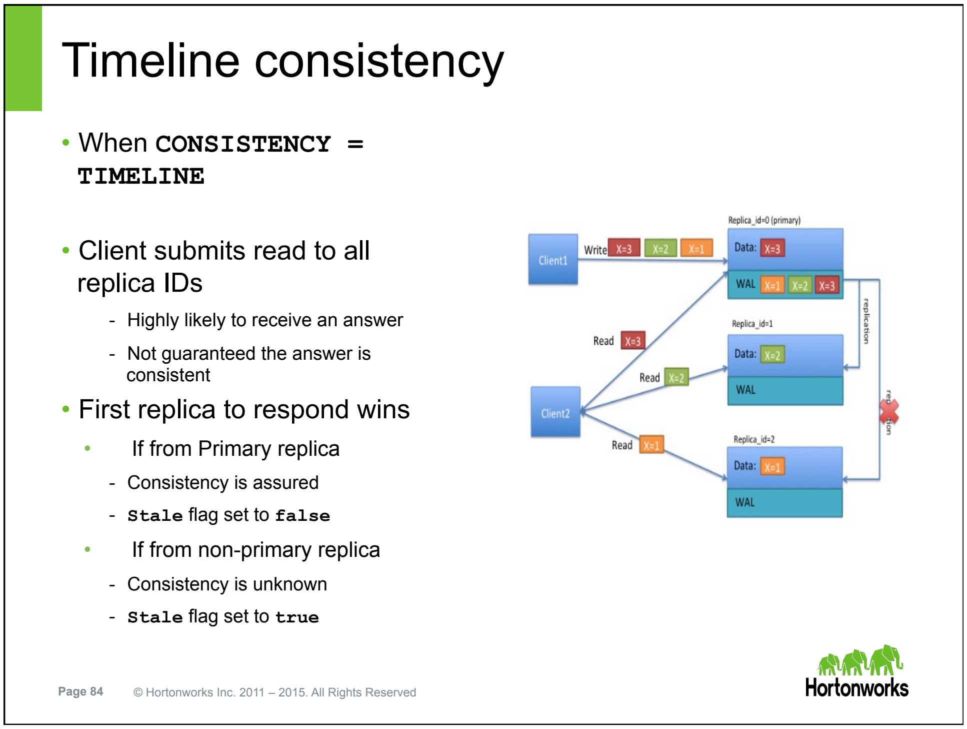


Strong consistency

- When **`CONSISTENCY = STRONG`**
- Client submits read to Replica ID=0
  - Response only comes from primary (no guarantee of answer)
  - An answer is guaranteed to be consistent (Contains a **`Stale`** flag set to **`false`**)
- If no response comes in 10ms
  - Client submits reads to replica regions (an answer is likely)
  - An answer is likely (**`Stale`** flag set to **`true`**)

Client1
Write X=3 X=2 X=1
Replica_id=0 (primary)
Data: X=3
WAL X=1 X=2 X=3
replication
Replica_id=1
Data: X=2
WAL
Read X=3
Read X=2
Client2
Read X=1
replication
Replica_id=2
Data: X=1
WAL

Page 83

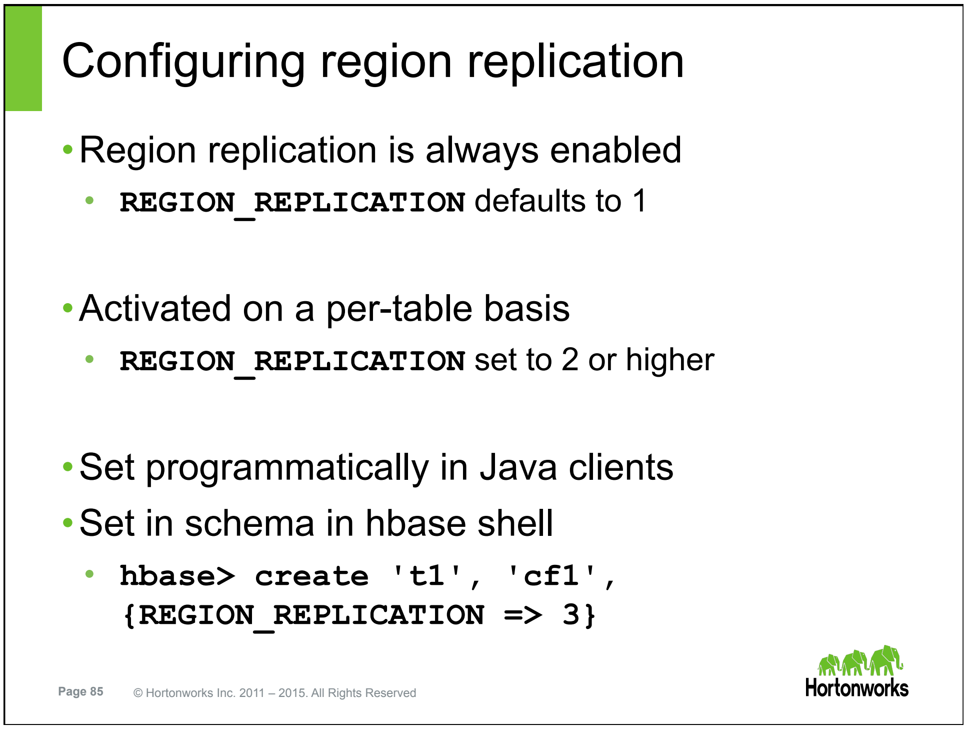


Timeline consistency

- When **`CONSISTENCY = TIMELINE`**
- Client submits read to all replica IDs
  - Highly likely to receive an answer
  - Not guaranteed the answer is consistent
- First replica to respond wins
  - If from Primary replica
    - Consistency is assured
    - **`Stale`** flag set to **`false`**
  - If from non-primary replica
    - Consistency is unknown
    - **`Stale`** flag set to **`true`**

Client1
Write X=3 X=2 X=1
Replica_id=0 (primary)
Data: X=3
WAL X=1 X=2 X=3
replication
Replica_id=1
Data: X=2
WAL
Read X=3
Read X=2
Client2
Read X=1
replication
Replica_id=2
Data: X=1
WAL

Page 84



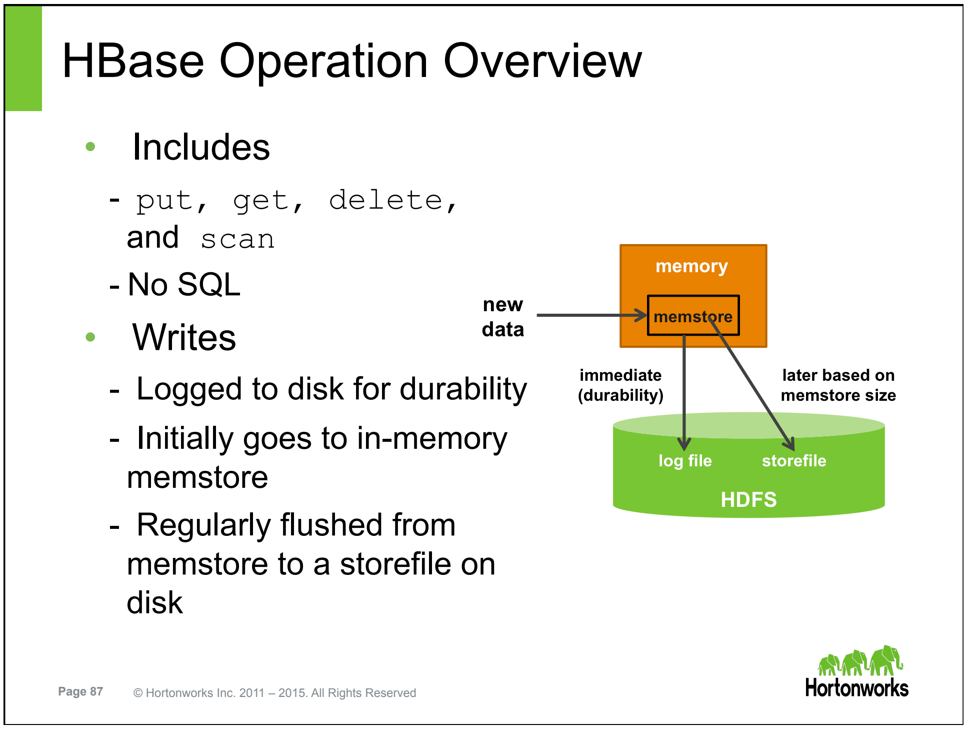
Configuring region replication

- Region replication is always enabled
  - **`REGION_REPLICATION`** defaults to 1
- Activated on a per-table basis
  - **`REGION_REPLICATION`** set to 2 or higher
- Set programmatically in Java clients
- Set in schema in hbase shell

  -
    ```
    hbase> create 't1', 'cf1',
    {REGION_REPLICATION => 3}
    ```

Page 85

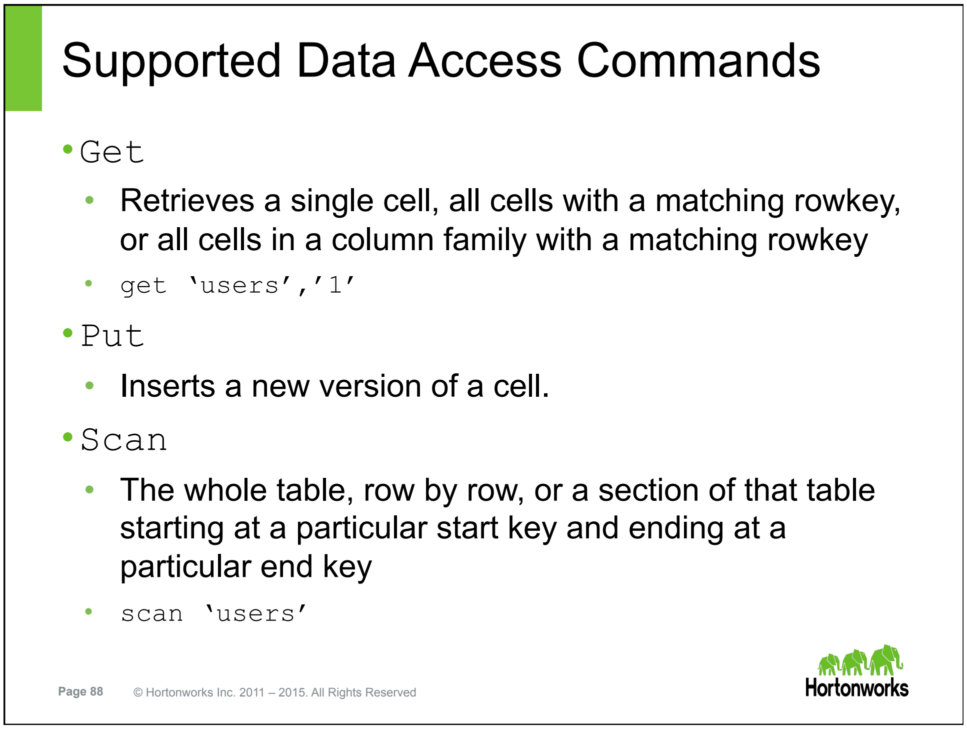
HBase Data Operations



HBase Operation Overview

- Includes
  - `put, get, delete,` and `scan`
  - No SQL
- Writes
  - Logged to disk for durability
  - Initially goes to in-memory memstore
  - Regularly flushed from memstore to a storefile on disk

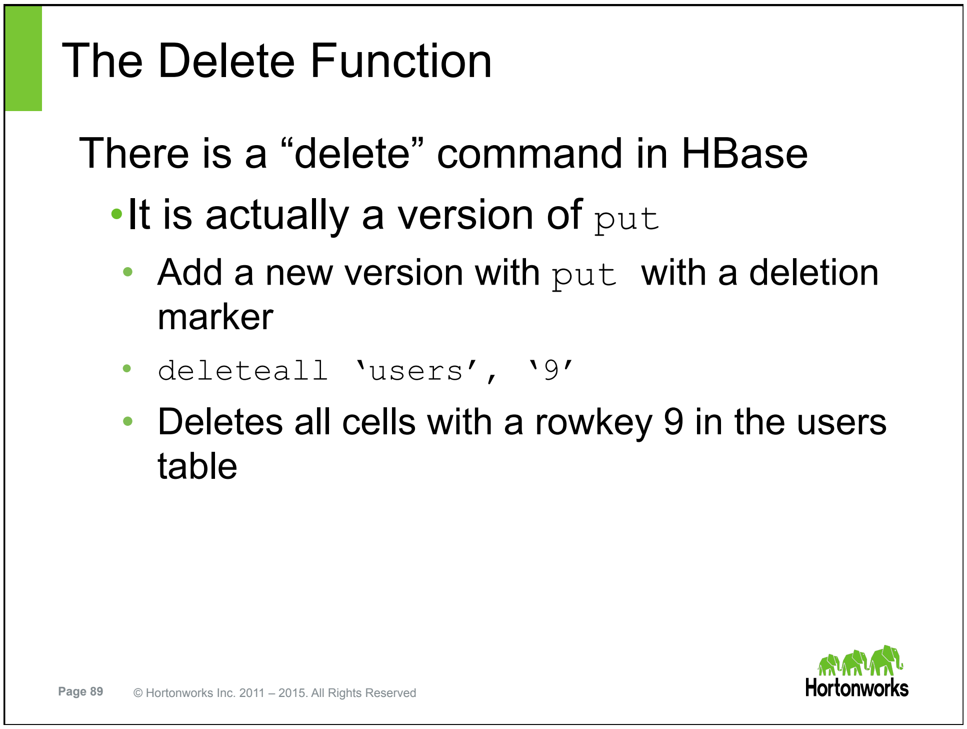
Page 87



Supported Data Access Commands

- `Get`
  - Retrieves a single cell, all cells with a matching rowkey, or all cells in a column family with a matching rowkey
  - `get ‘users’,’1’`
- `Put`
  - Inserts a new version of a cell.
- `Scan`
  - The whole table, row by row, or a section of that table starting at a particular start key and ending at a particular end key
  - `scan ‘users’`

Page 88

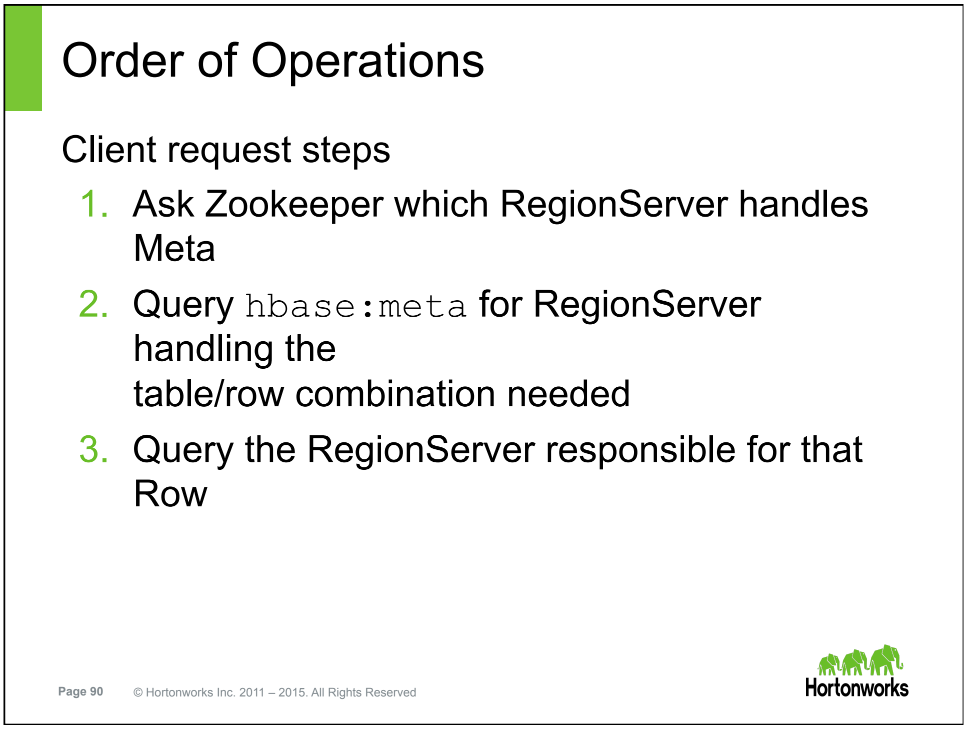


The Delete Function

There is a “delete” command in HBase

- It is actually a version of `put`
  - Add a new version with `put` with a deletion marker
  - `deleteall 'users', '9'`
  - Deletes all cells with a rowkey 9 in the users table

Page 89

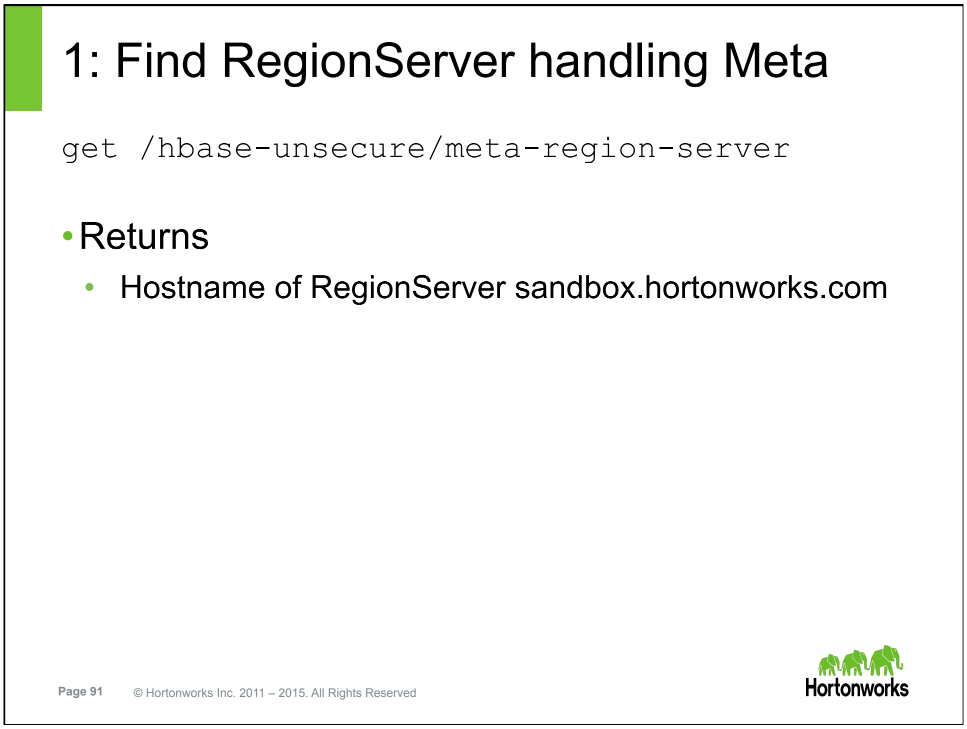


Order of Operations

Client request steps

1. Ask Zookeeper which RegionServer handles Meta
2. Query `hbase:meta` for RegionServer handling the table/row combination needed
3. Query the RegionServer responsible for that Row

Page 90

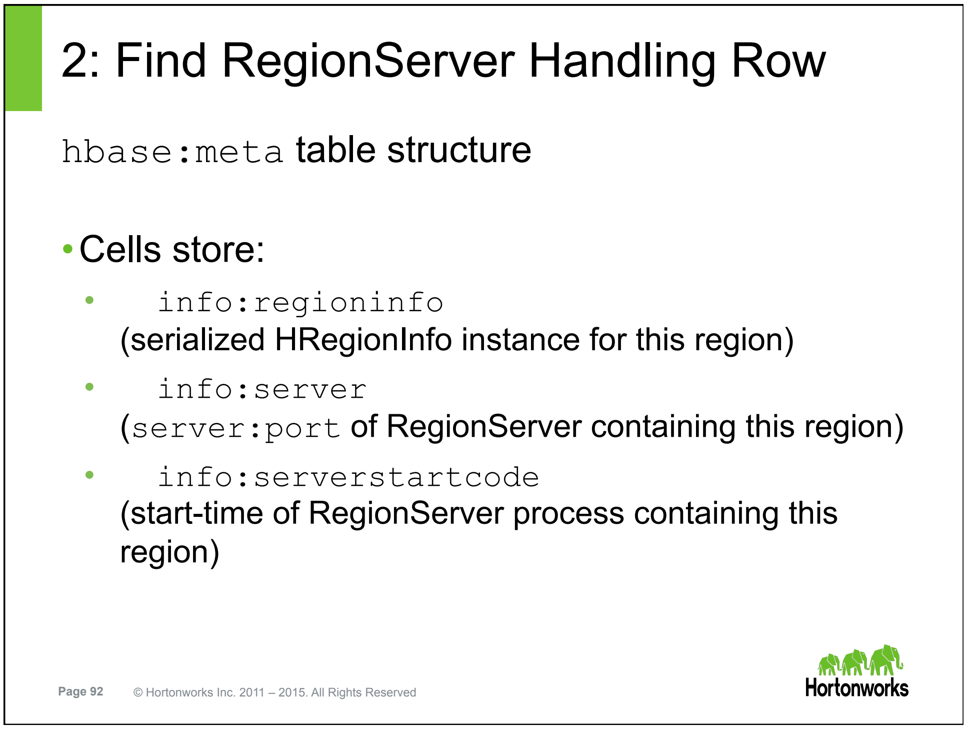


1: Find RegionServer handling Meta

```
get /hbase-unsecure/meta-region-server
```

- Returns
  - Hostname of RegionServer sandbox.hortonworks.com

Page 91

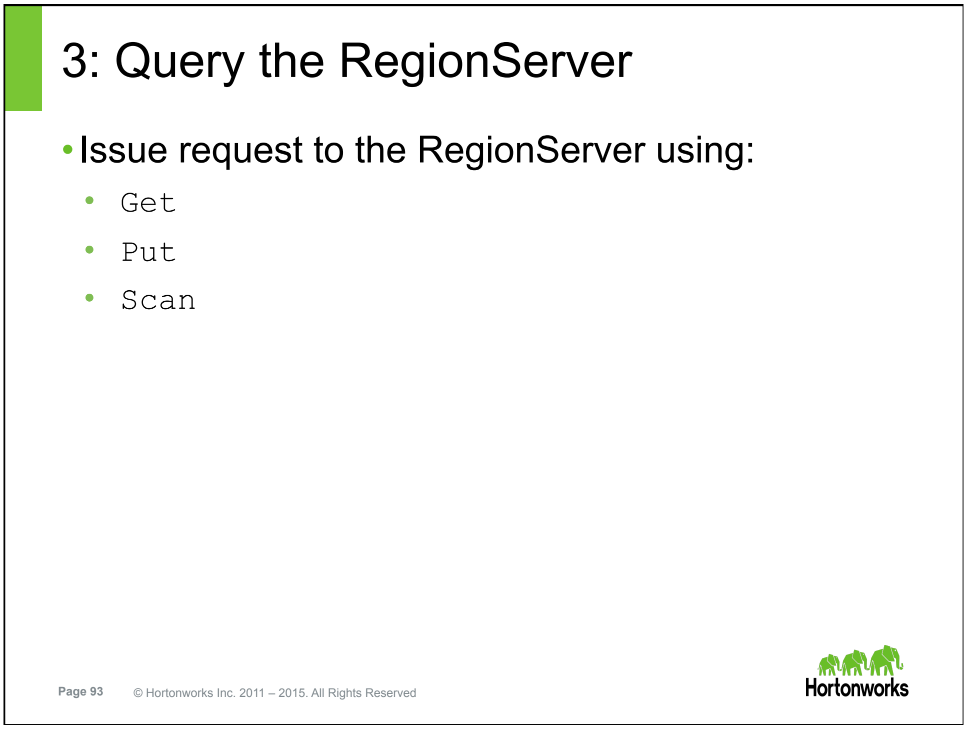


2: Find RegionServer Handling Row

`hbase:meta` table structure

- Cells store:
  - `info:regioninfo`
    (serialized HRegionInfo instance for this region)
  - `info:server`
    (`server:port` of RegionServer containing this region)
  - `info:serverstartcode`
    (start-time of RegionServer process containing this region)

Page 92

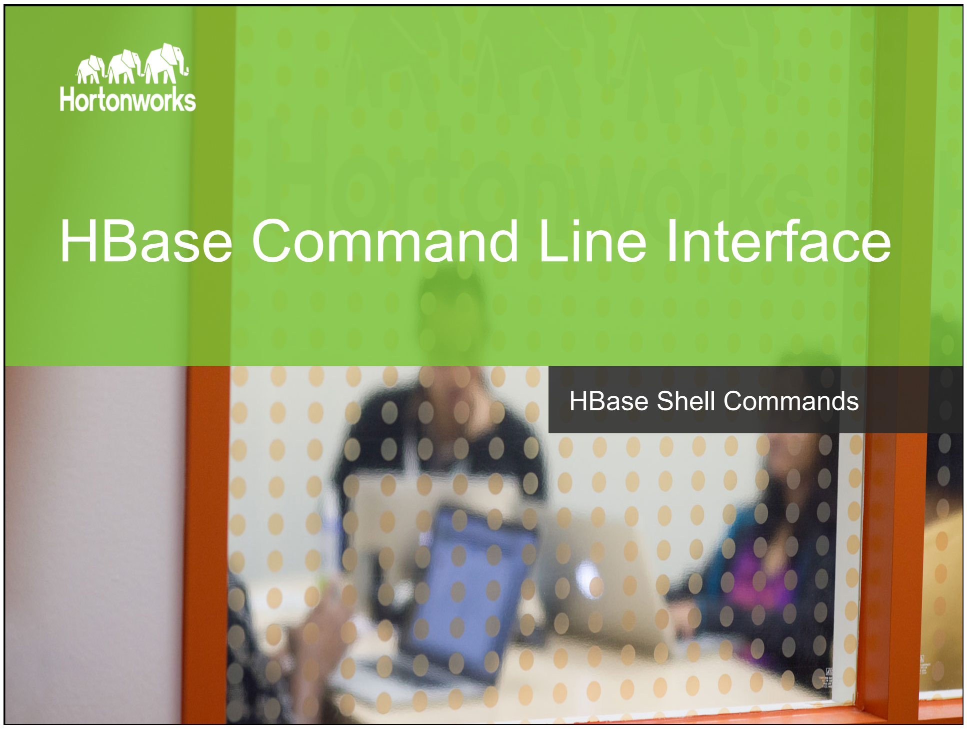


3: Query the RegionServer

- Issue request to the RegionServer using:
  - `Get`
  - `Put`
  - `Scan`

Page 93

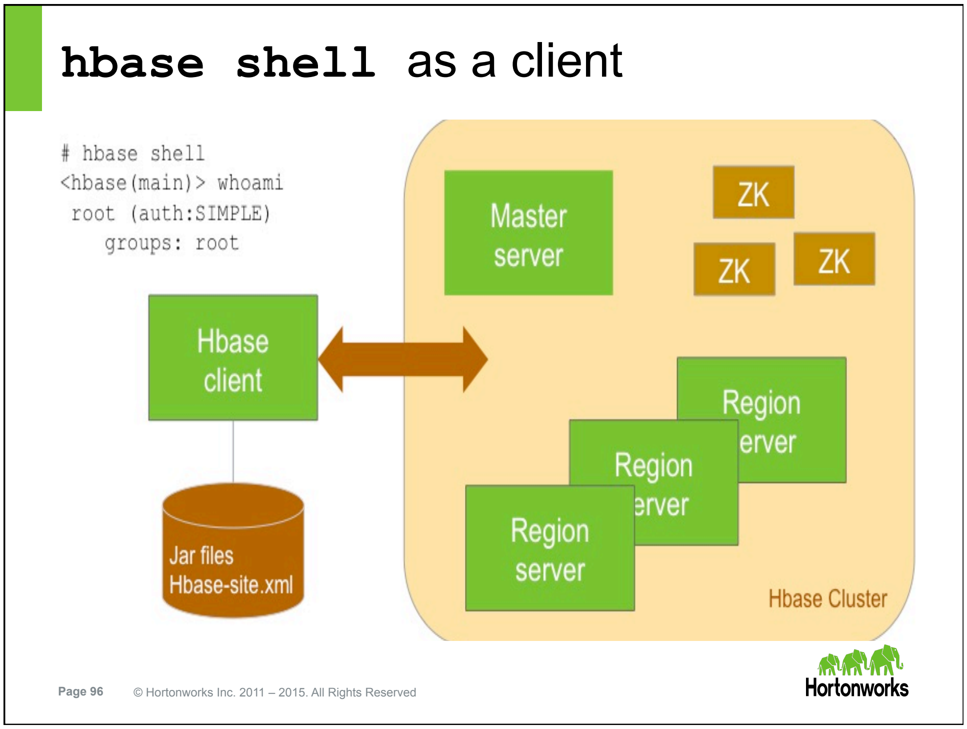
Lab:
Examining Configuration Files



Hortonworks

HBase Command Line Interface

HBase Shell Commands

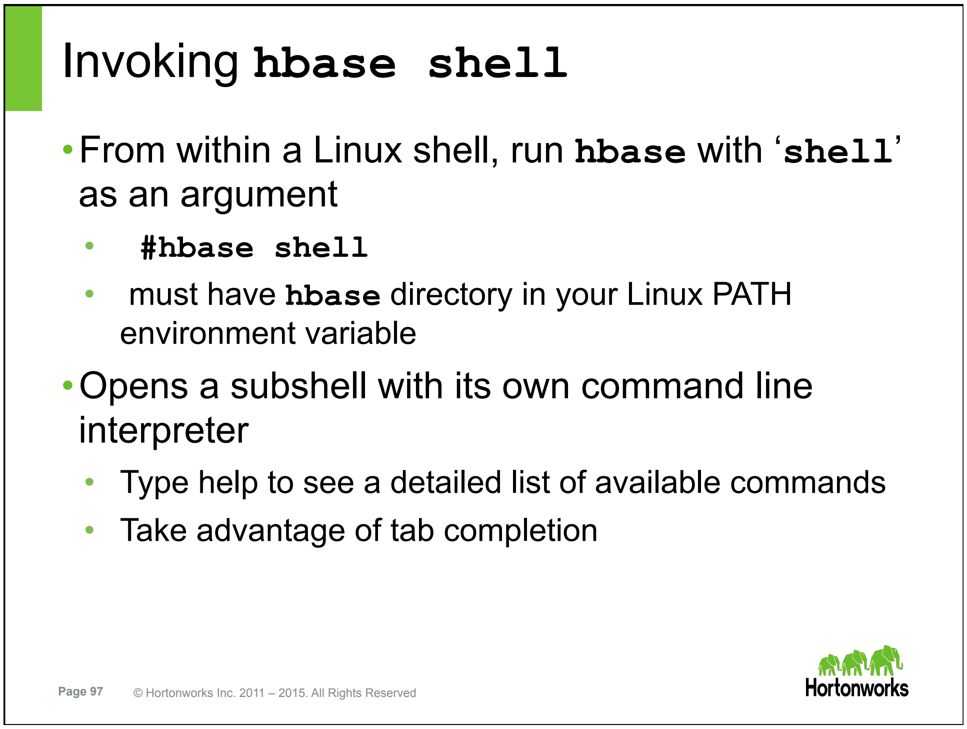


`hbase shell` as a client

```
# hbase shell
<hbase(main)> whoami
 root (auth:SIMPLE)
    groups: root
```

Page 96 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



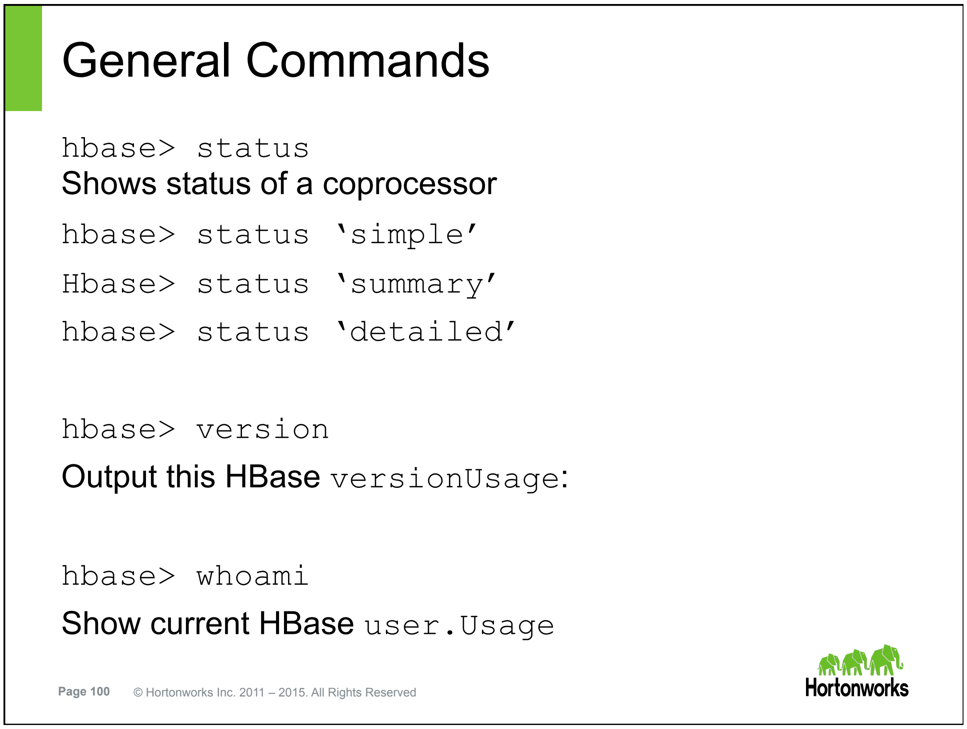
Invoking `hbase shell`

- From within a Linux shell, run **`hbase`** with '**`shell`**' as an argument
  - **`#hbase shell`**
  - must have **`hbase`** directory in your Linux PATH environment variable
- Opens a subshell with its own command line interpreter
  - Type help to see a detailed list of available commands
  - Take advantage of tab completion

Page 97 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Shell Command Categories

General Commands



General Commands

```
hbase> status
```

Shows status of a coprocessor

```
hbase> status ‘simple’
Hbase> status ‘summary’
hbase> status ‘detailed’
```

```
hbase> version
```

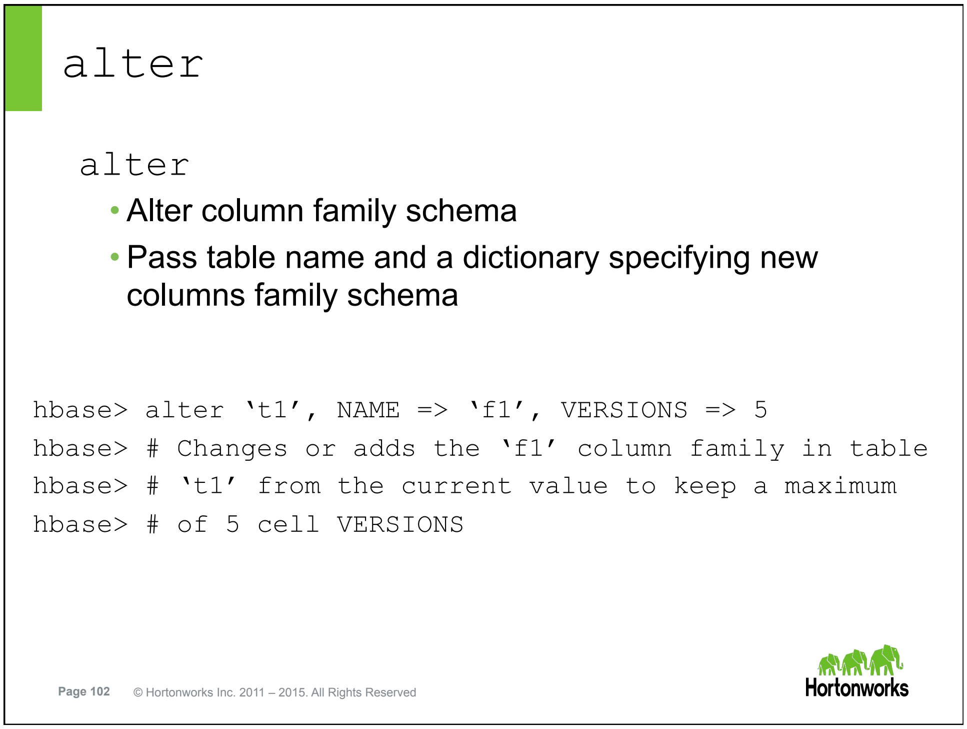
Output this HBase `versionUsage`:

```
hbase> whoami
```

Show current HBase `user.Usage`

Hortonworks

Table Management Commands



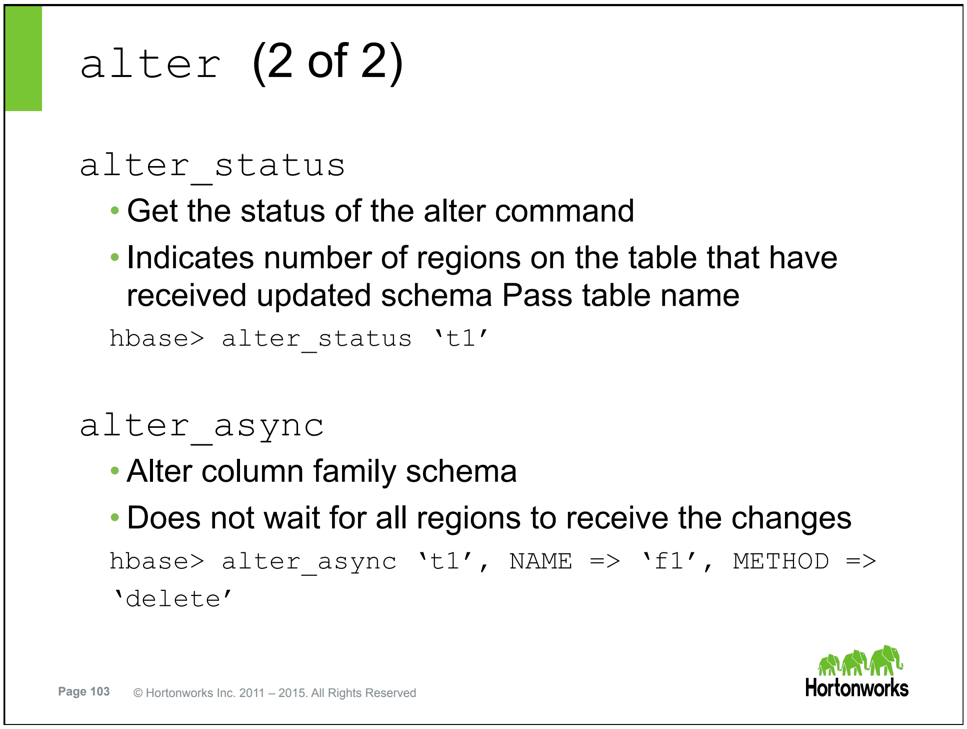
alter

alter

- Alter column family schema
- Pass table name and a dictionary specifying new columns family schema

```
hbase> alter ‘t1’, NAME => ‘f1’, VERSIONS => 5
hbase> # Changes or adds the ‘f1’ column family in table
hbase> # ‘t1’ from the current value to keep a maximum
hbase> # of 5 cell VERSIONS
```

Page 102



alter (2 of 2)

alter_status

- Get the status of the alter command
- Indicates number of regions on the table that have received updated schema Pass table name

```
hbase> alter_status 't1'
```

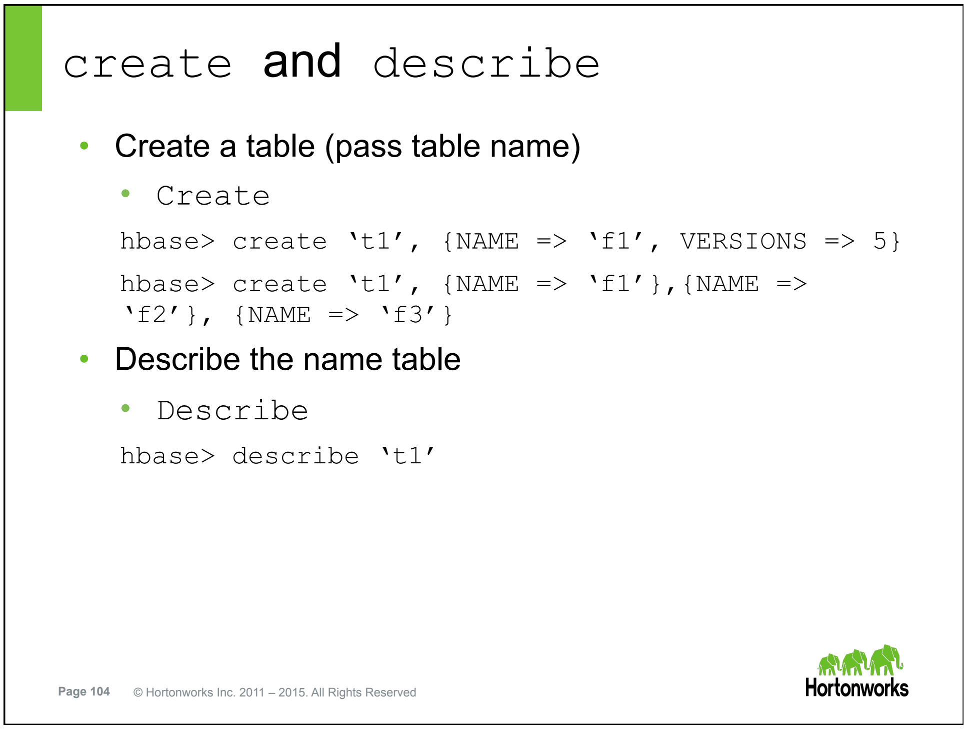
alter_async

- Alter column family schema
- Does not wait for all regions to receive the changes

```
hbase> alter_async 't1', NAME => 'f1', METHOD =>
'delete'
```

Page 103

Hortonworks



create and describe

- Create a table (pass table name)
  - Create

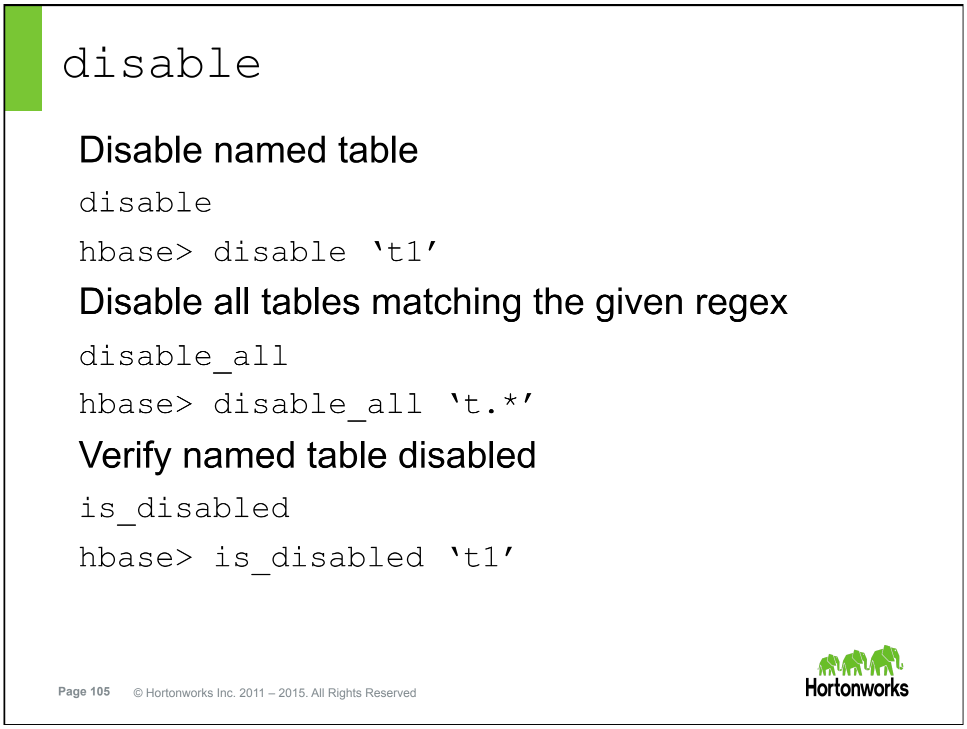
```
hbase> create 't1', {NAME => 'f1', VERSIONS => 5}
hbase> create 't1', {NAME => 'f1'},{NAME =>
'f2'}, {NAME => 'f3'}
```

- Describe the name table
  - Describe

```
hbase> describe 't1'
```

Page 104

Hortonworks



disable

Disable named table

```
disable
hbase> disable ‘t1’
```

Disable all tables matching the given regex

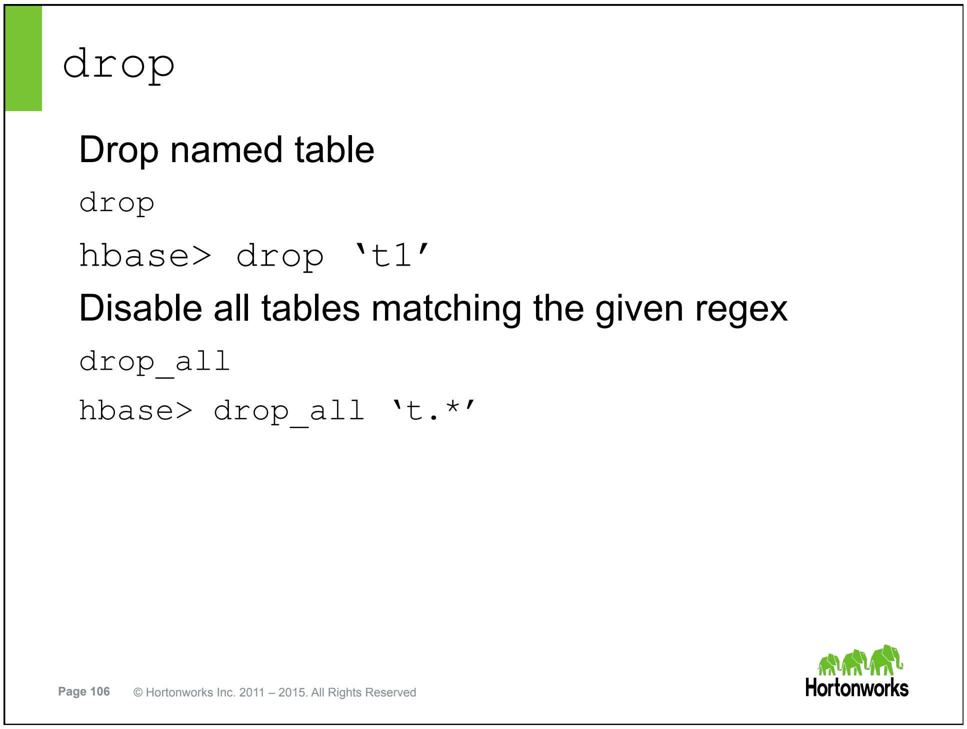
```
disable_all
hbase> disable_all ‘t.*’
```

Verify named table disabled

```
is_disabled
hbase> is_disabled ‘t1’
```

Page 105 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



drop

Drop named table

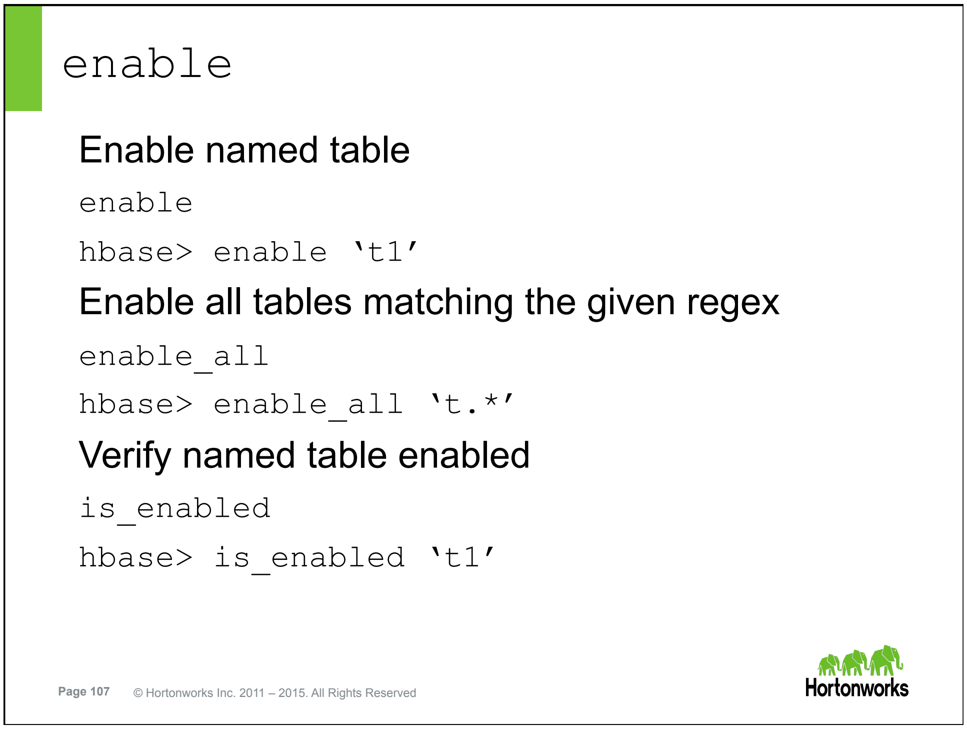
```
drop
hbase> drop ‘t1’
```

Disable all tables matching the given regex

```
drop_all
hbase> drop_all ‘t.*’
```

Page 106 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



enable

Enable named table

```
enable
hbase> enable ‘t1’
```

Enable all tables matching the given regex

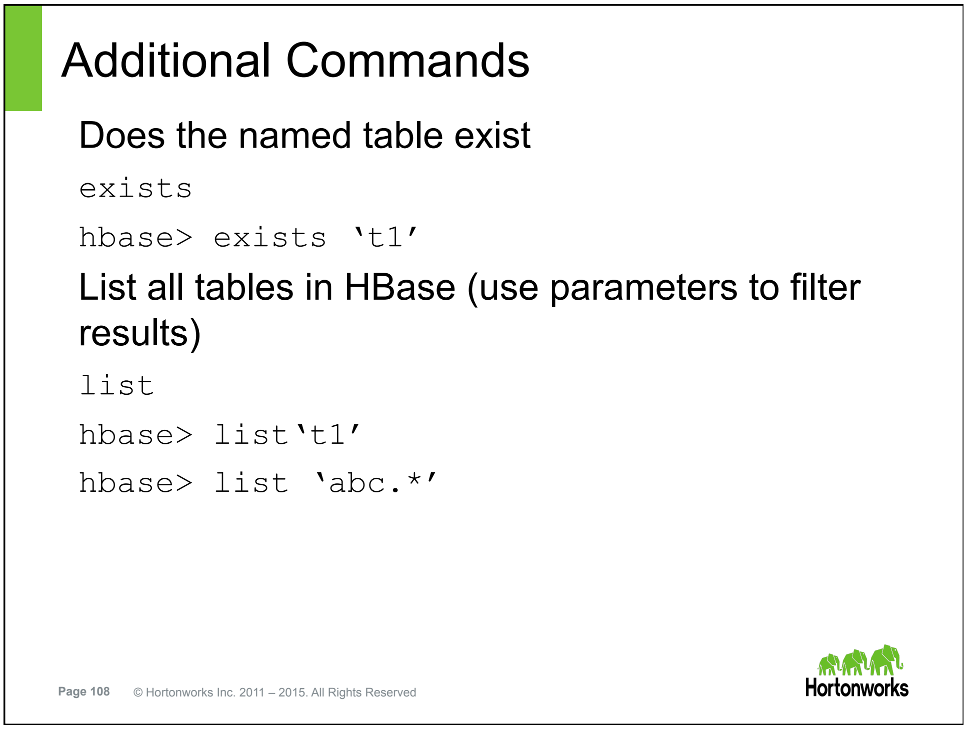
```
enable_all
hbase> enable_all ‘t.*’
```

Verify named table enabled

```
is_enabled
hbase> is_enabled ‘t1’
```

Page 107

Hortonworks



Additional Commands

Does the named table exist

```
exists
hbase> exists ‘t1’
```

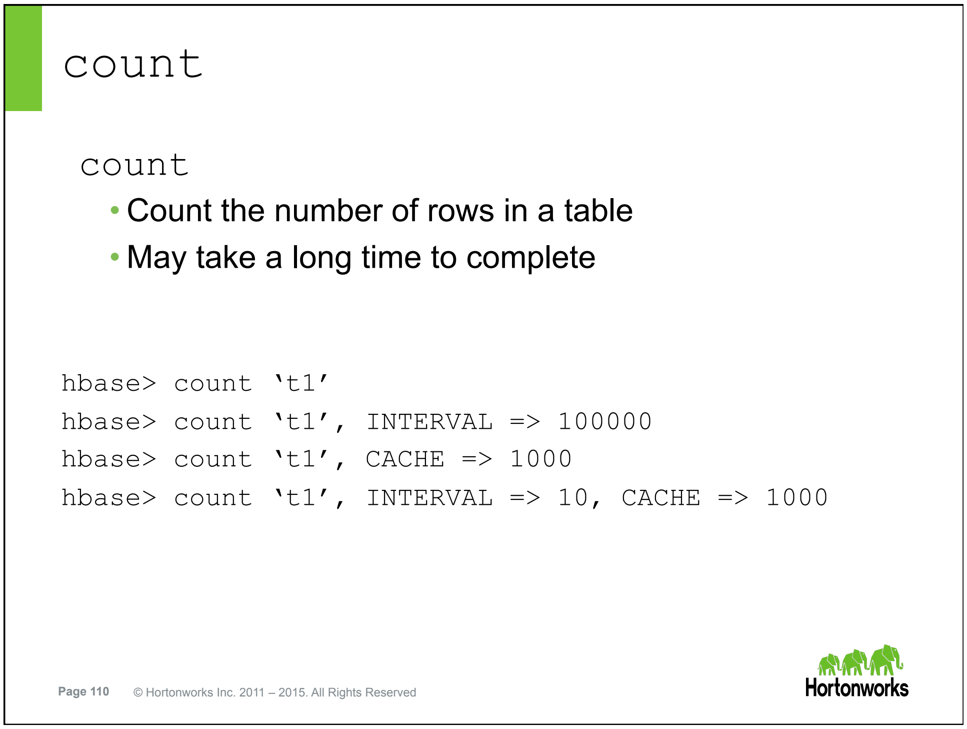
List all tables in HBase (use parameters to filter results)

```
list
hbase> list‘t1’
hbase> list ‘abc.*’
```

Page 108

Hortonworks

Data Manipulation Commands



count

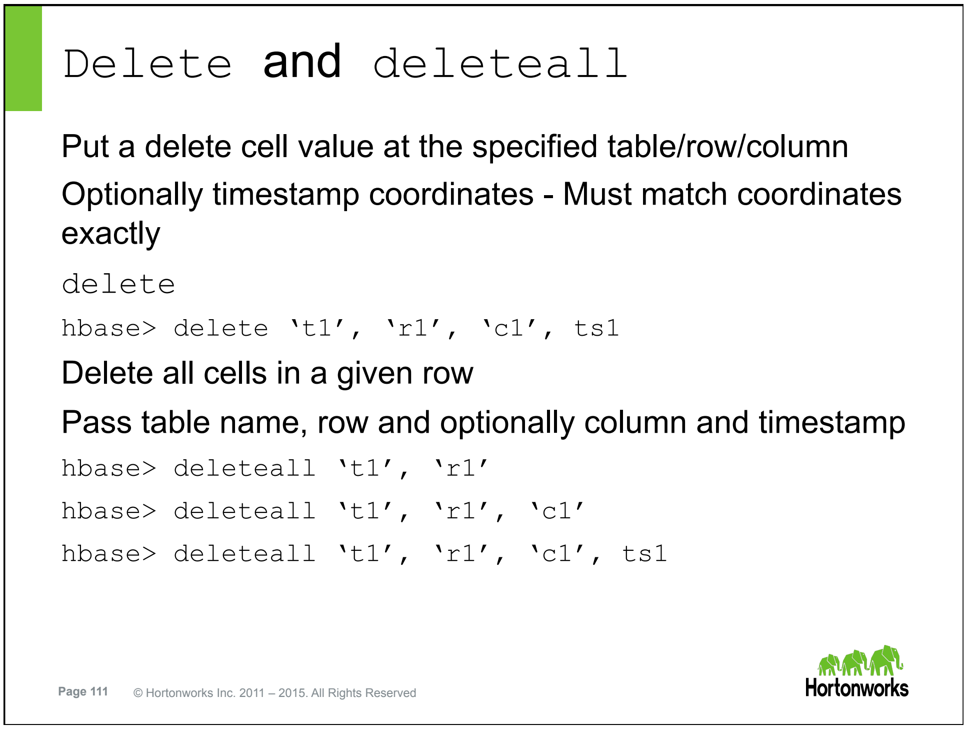
count

- Count the number of rows in a table
- May take a long time to complete

```
hbase> count ‘t1’
hbase> count ‘t1’, INTERVAL => 100000
hbase> count ‘t1’, CACHE => 1000
hbase> count ‘t1’, INTERVAL => 10, CACHE => 1000
```

Page 110 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Delete and deleteall

Put a delete cell value at the specified table/row/column

Optionally timestamp coordinates - Must match coordinates exactly

delete

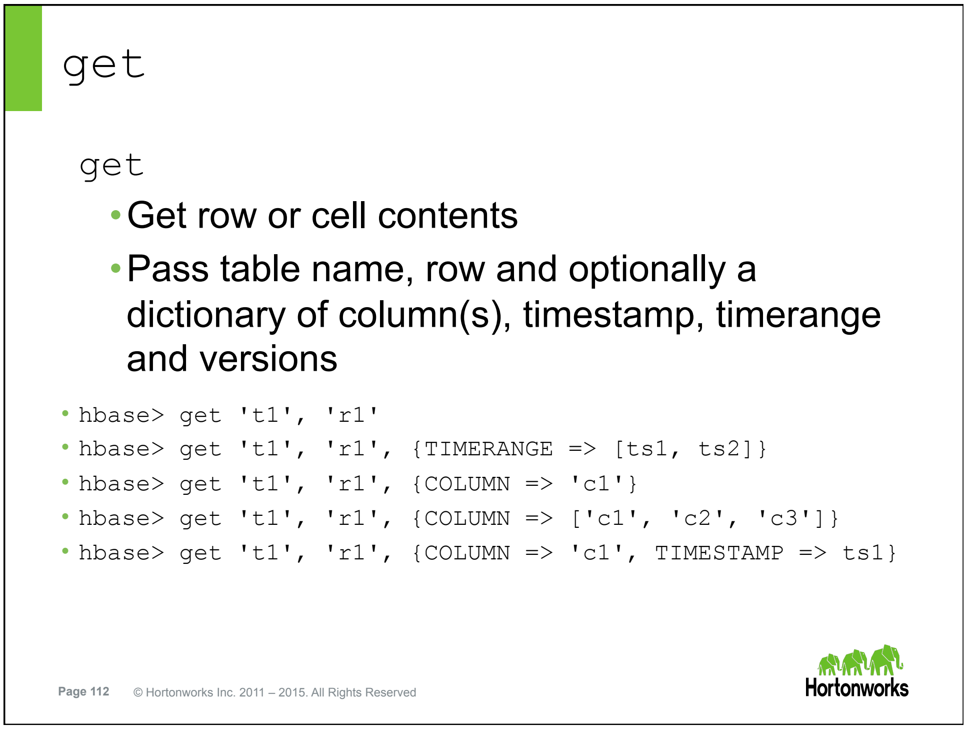
```
hbase> delete ‘t1’, ‘r1’, ‘c1’, ts1
```

Delete all cells in a given row

Pass table name, row and optionally column and timestamp

```
hbase> deleteall ‘t1’, ‘r1’
hbase> deleteall ‘t1’, ‘r1’, ‘c1’
hbase> deleteall ‘t1’, ‘r1’, ‘c1’, ts1
```

Page 111



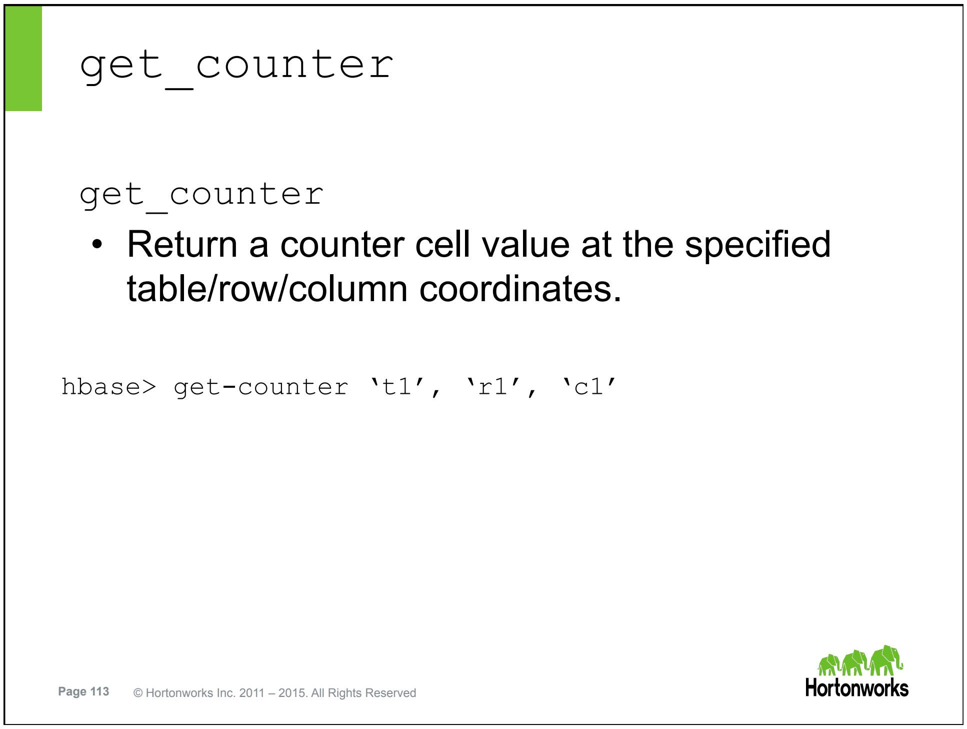
get

get

- Get row or cell contents
- Pass table name, row and optionally a dictionary of column(s), timestamp, timerange and versions

```
• hbase> get 't1', 'r1'
• hbase> get 't1', 'r1', {TIMERANGE => [ts1, ts2]}
• hbase> get 't1', 'r1', {COLUMN => 'c1'}
• hbase> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}
• hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
```

Page 112

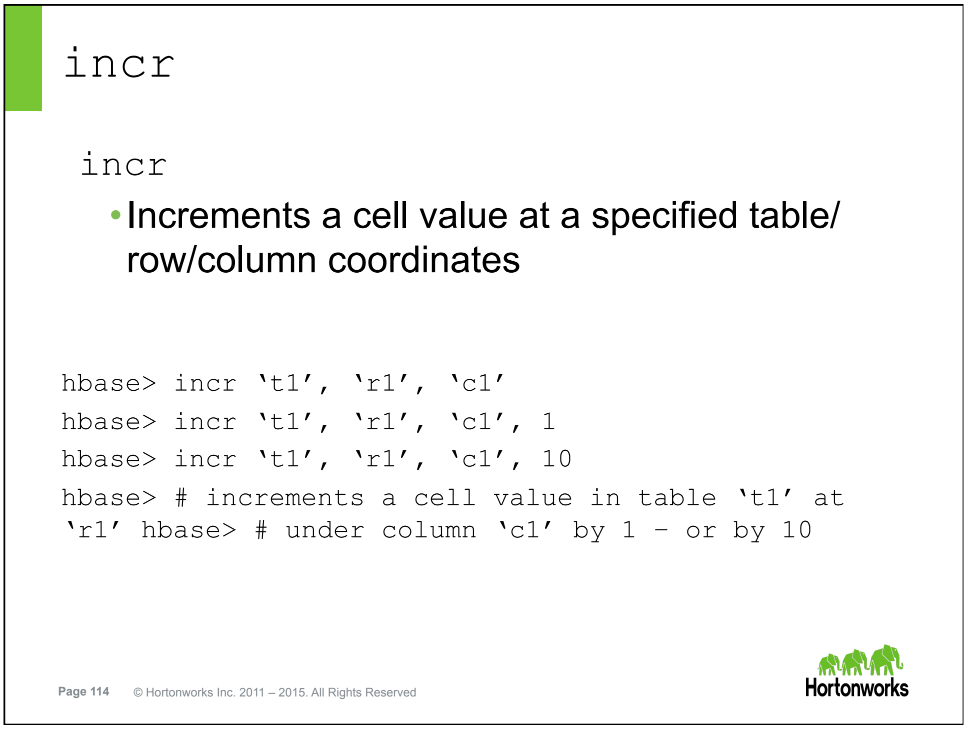


get_counter

get_counter

- Return a counter cell value at the specified table/row/column coordinates.

```
hbase> get-counter ‘t1’, ‘r1’, ‘c1’
```

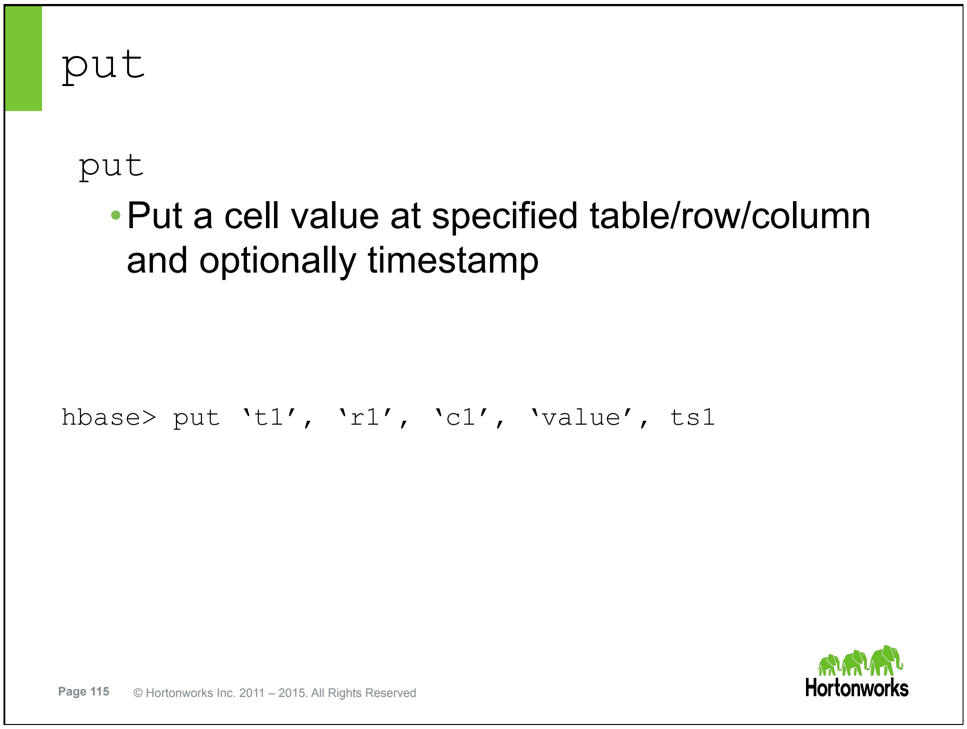


incr

incr

- Increments a cell value at a specified table/ row/column coordinates

```
hbase> incr ‘t1’, ‘r1’, ‘c1’
hbase> incr ‘t1’, ‘r1’, ‘c1’, 1
hbase> incr ‘t1’, ‘r1’, ‘c1’, 10
hbase> # increments a cell value in table ‘t1’ at
‘r1’ hbase> # under column ‘c1’ by 1 – or by 10
```



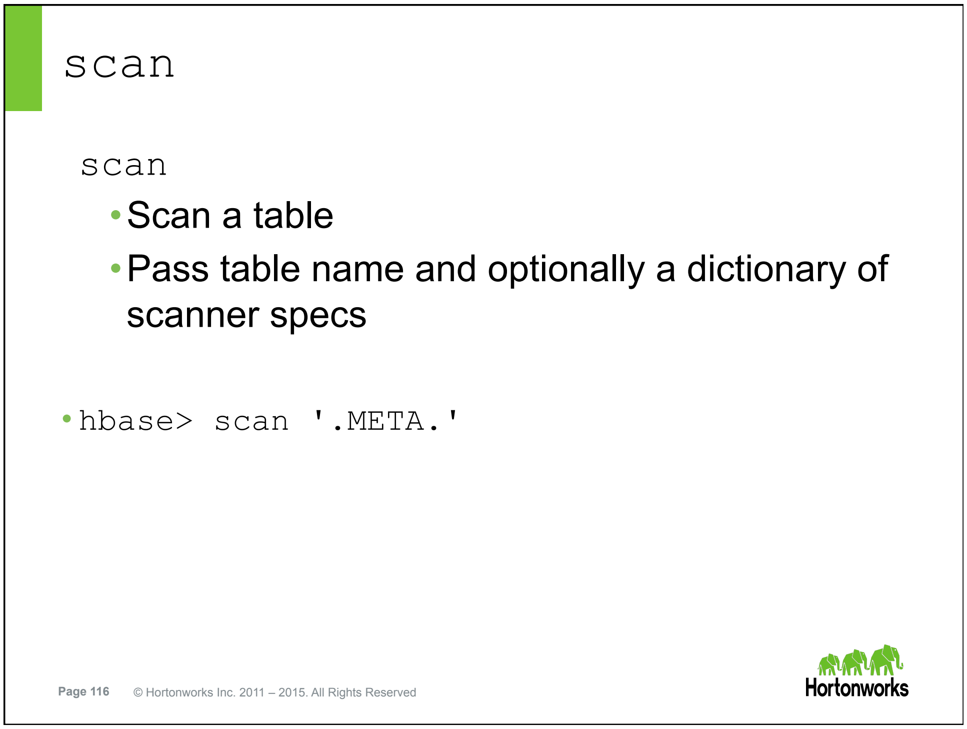
put

put

- Put a cell value at specified table/row/column and optionally timestamp

```
hbase> put ‘t1’, ‘r1’, ‘c1’, ‘value’, ts1
```

Page 115



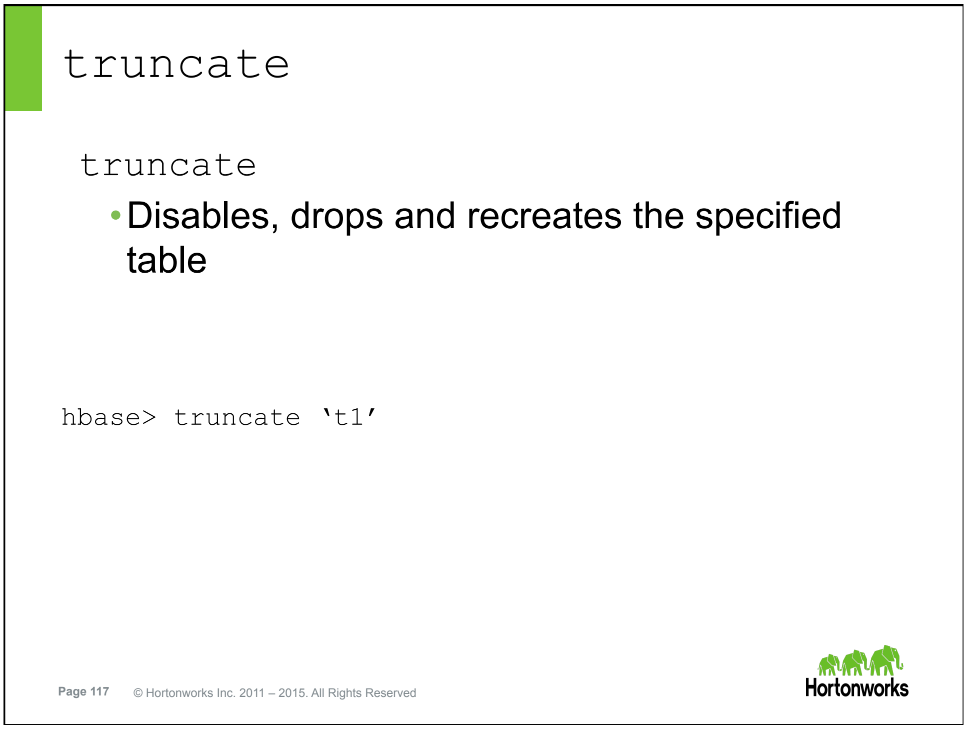
scan

scan

- Scan a table
- Pass table name and optionally a dictionary of scanner specs

- `hbase> scan '.META.'`

Page 116



truncate

truncate

- Disables, drops and recreates the specified table

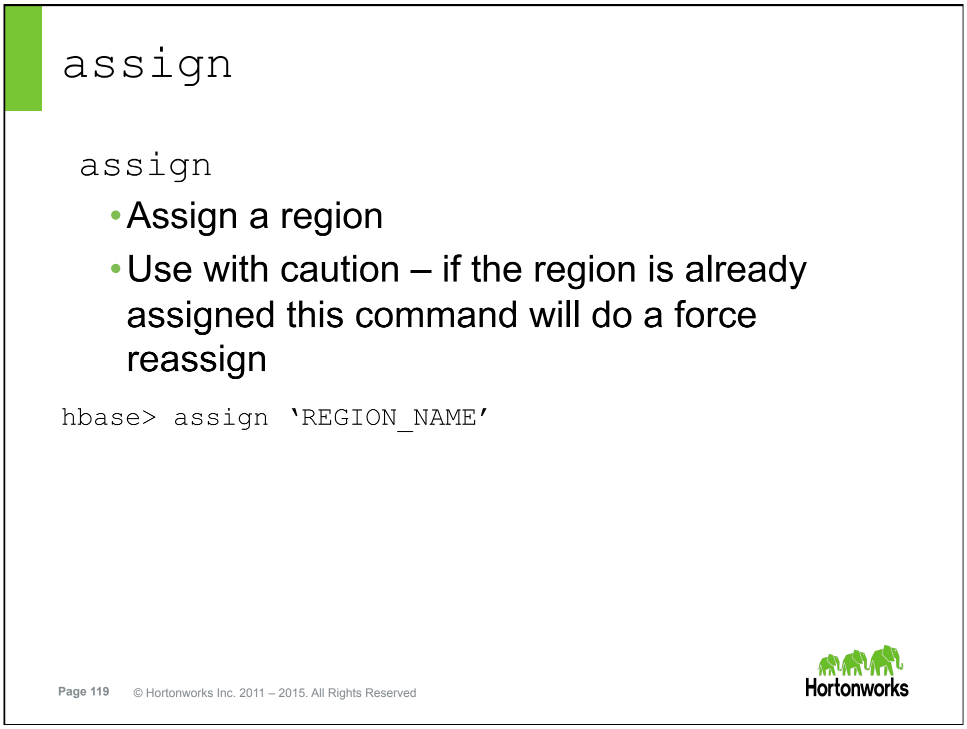
```
hbase> truncate ‘t1’
```

Page 117

Hortonworks

Surgery Tools

Hortonworks



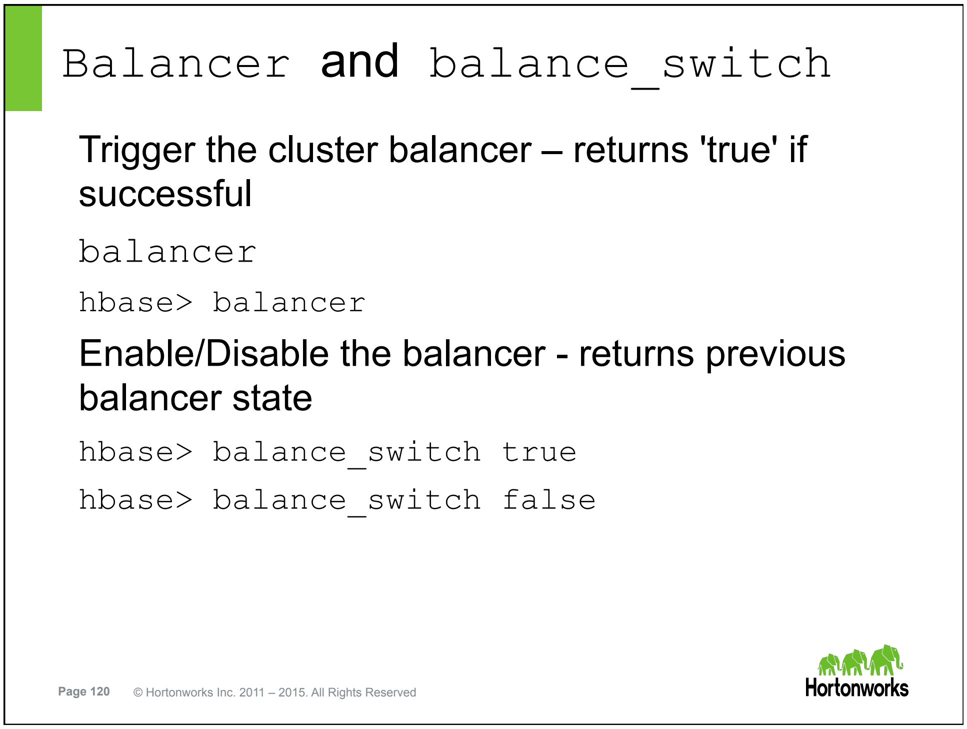
assign

assign

- Assign a region
- Use with caution – if the region is already assigned this command will do a force reassign

```
hbase> assign ‘REGION_NAME’
```

Page 119 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Balancer and balance_switch

Trigger the cluster balancer – returns 'true' if successful

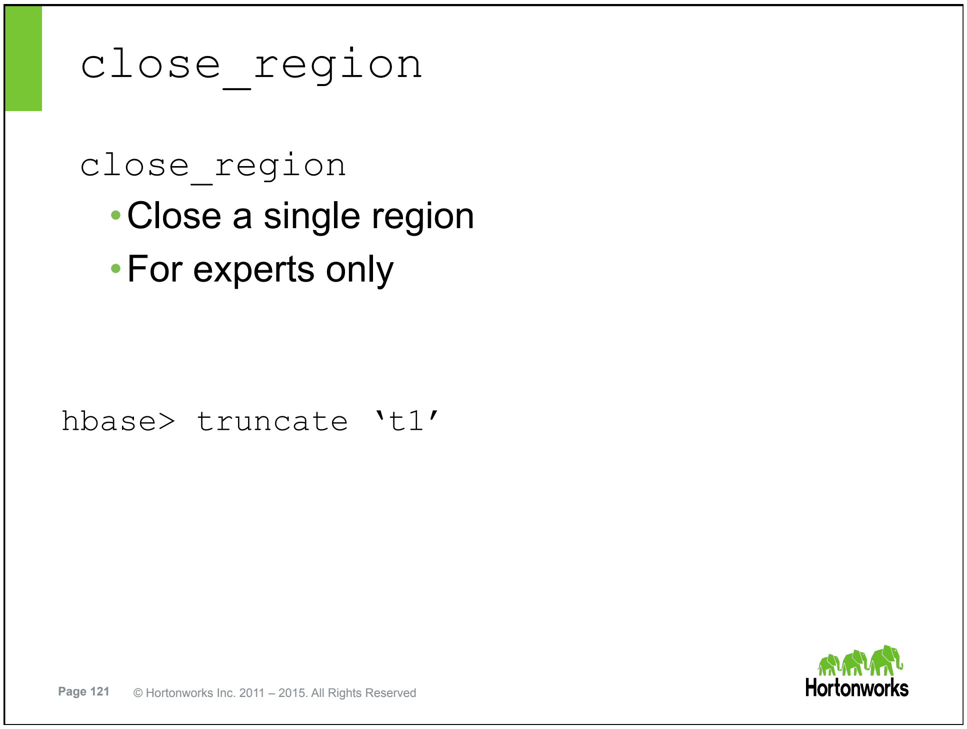
balancer

```
hbase> balancer
```

Enable/Disable the balancer - returns previous balancer state

```
hbase> balance_switch true
hbase> balance_switch false
```

Page 120 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

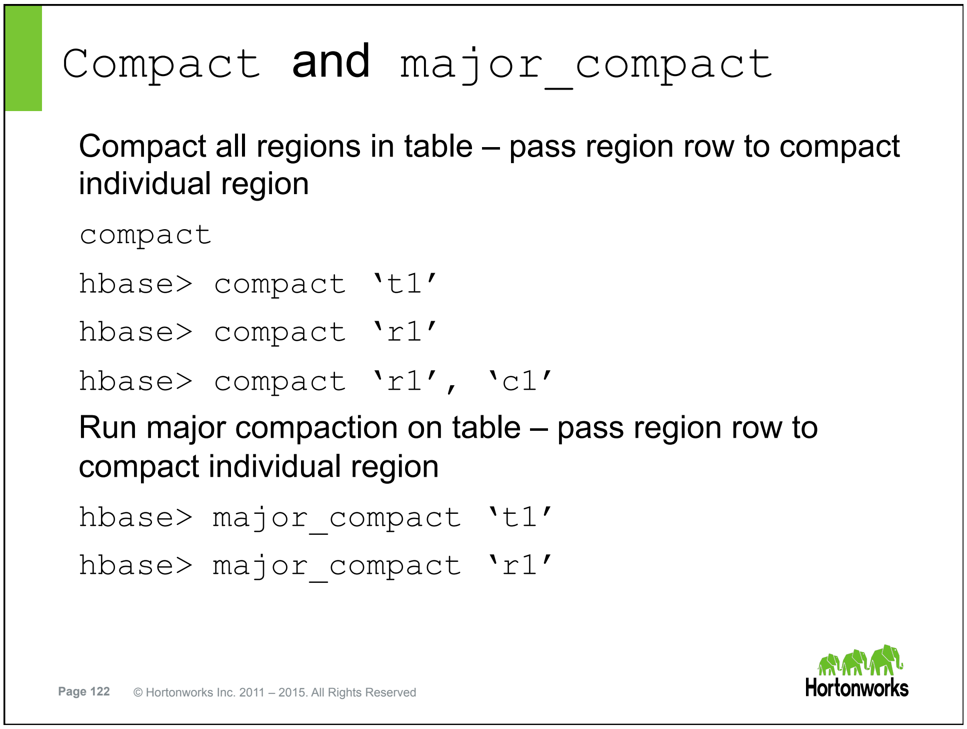


close_region

`close_region`

- Close a single region
- For experts only

```
hbase> truncate ‘t1’
```



Compact and major_compact

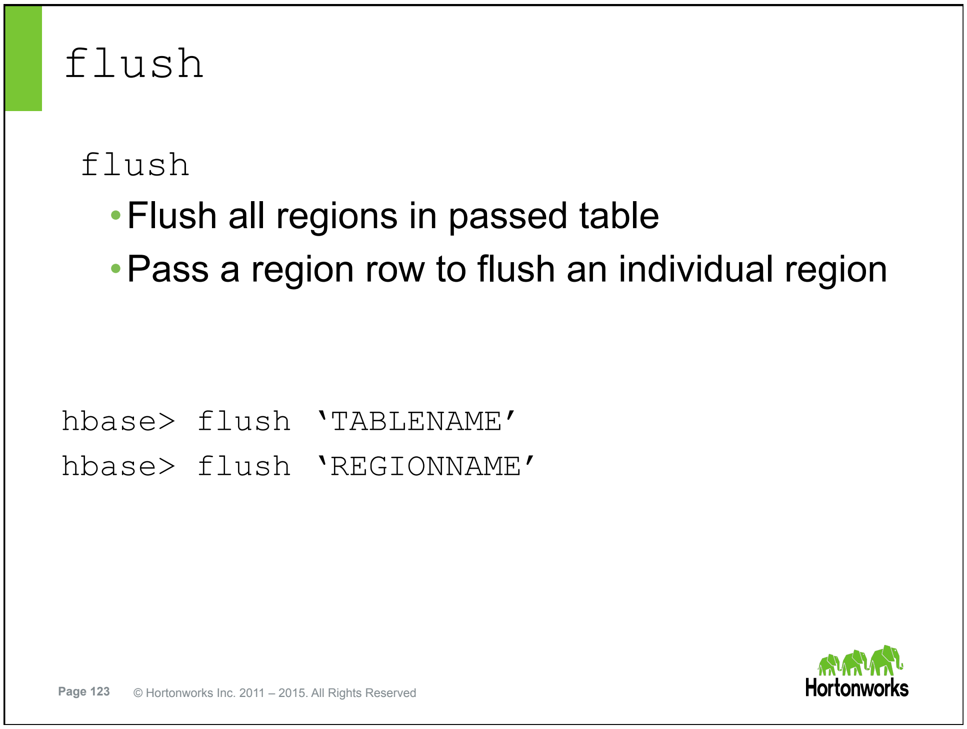
Compact all regions in table – pass region row to compact individual region

`compact`

```
hbase> compact ‘t1’
hbase> compact ‘r1’
hbase> compact ‘r1’, ‘c1’
```

Run major compaction on table – pass region row to compact individual region

```
hbase> major_compact ‘t1’
hbase> major_compact ‘r1’
```



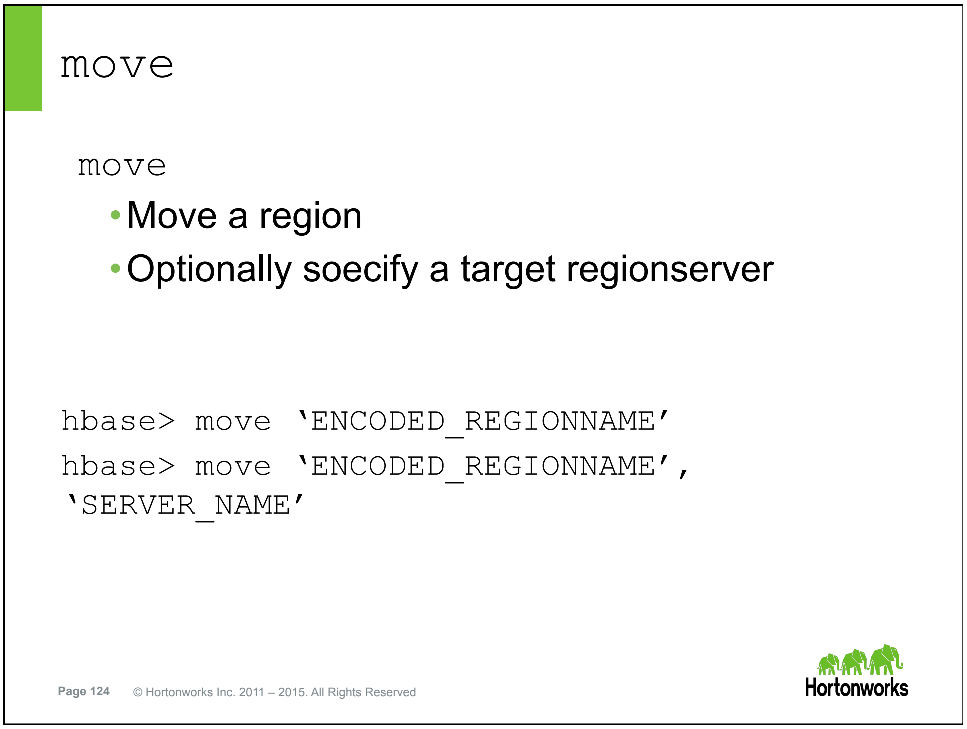
flush

flush

- Flush all regions in passed table
- Pass a region row to flush an individual region

```
hbase> flush ‘TABLENAME’
hbase> flush ‘REGIONNAME’
```

Page 123 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



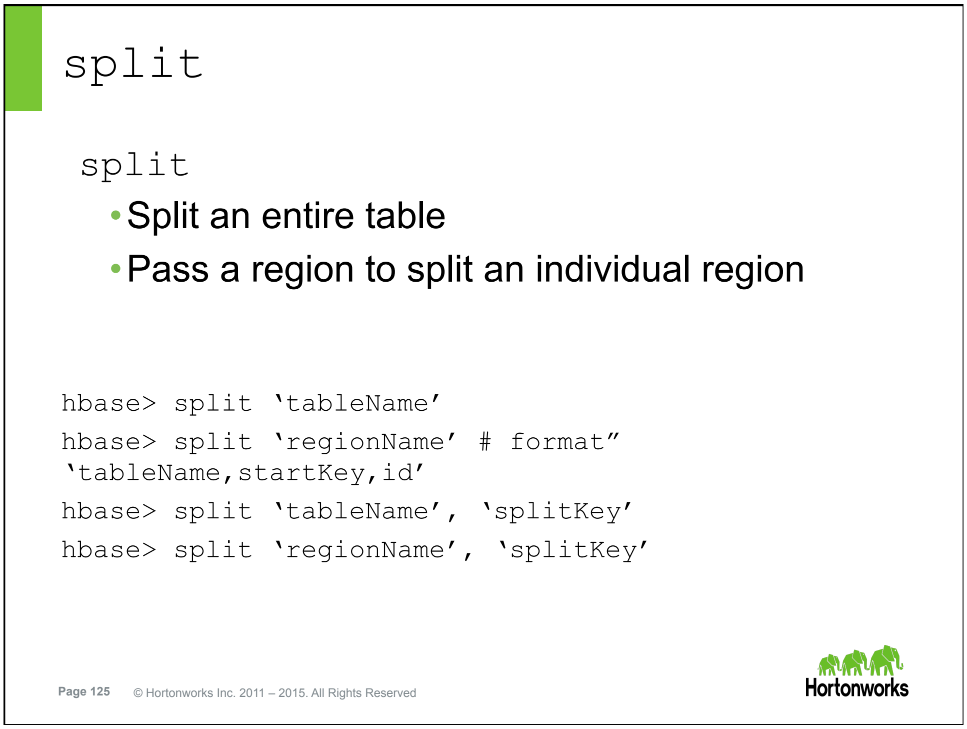
move

move

- Move a region
- Optionally soecify a target regionserver

```
hbase> move ‘ENCODED_REGIONNAME’
hbase> move ‘ENCODED_REGIONNAME’,
‘SERVER_NAME’
```

Page 124 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



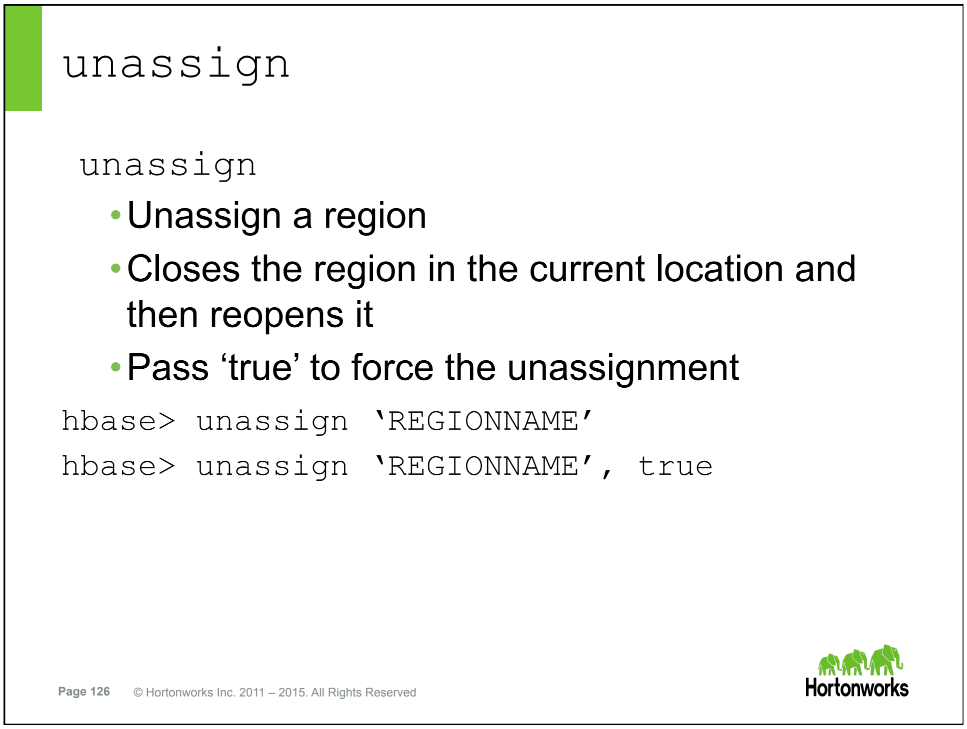
split

split

- Split an entire table
- Pass a region to split an individual region

```
hbase> split ‘tableName’
hbase> split ‘regionName’ # format”
‘tableName,startKey,id’
hbase> split ‘tableName’, ‘splitKey’
hbase> split ‘regionName’, ‘splitKey’
```

Page 125 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



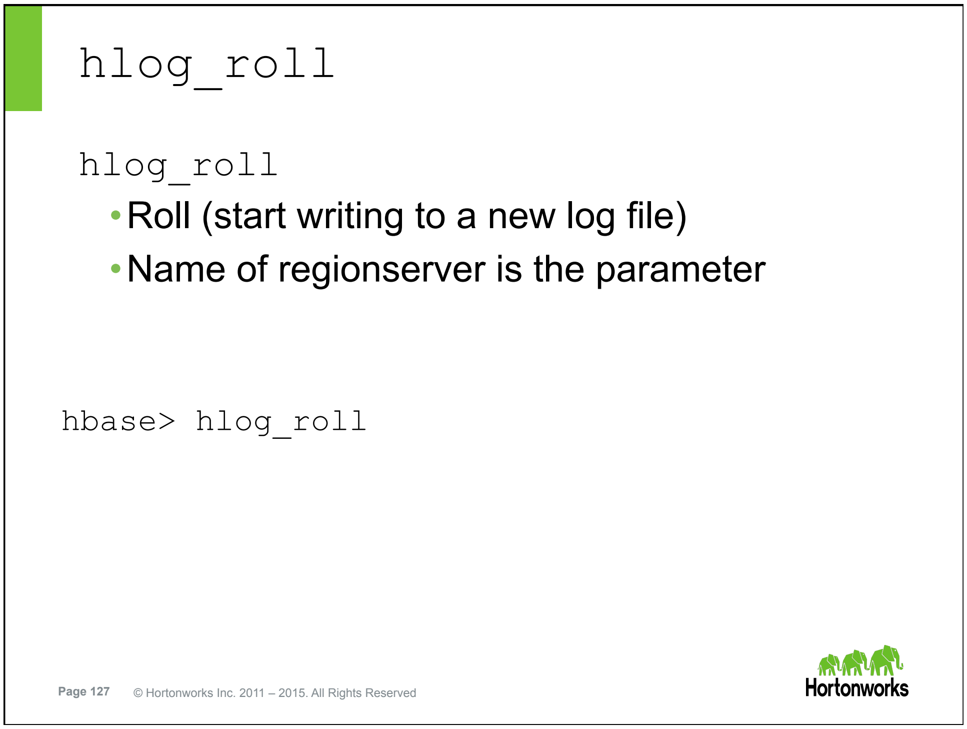
unassign

unassign

- Unassign a region
- Closes the region in the current location and then reopens it
- Pass ‘true’ to force the unassignment

```
hbase> unassign ‘REGIONNAME’
hbase> unassign ‘REGIONNAME’, true
```

Page 126 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



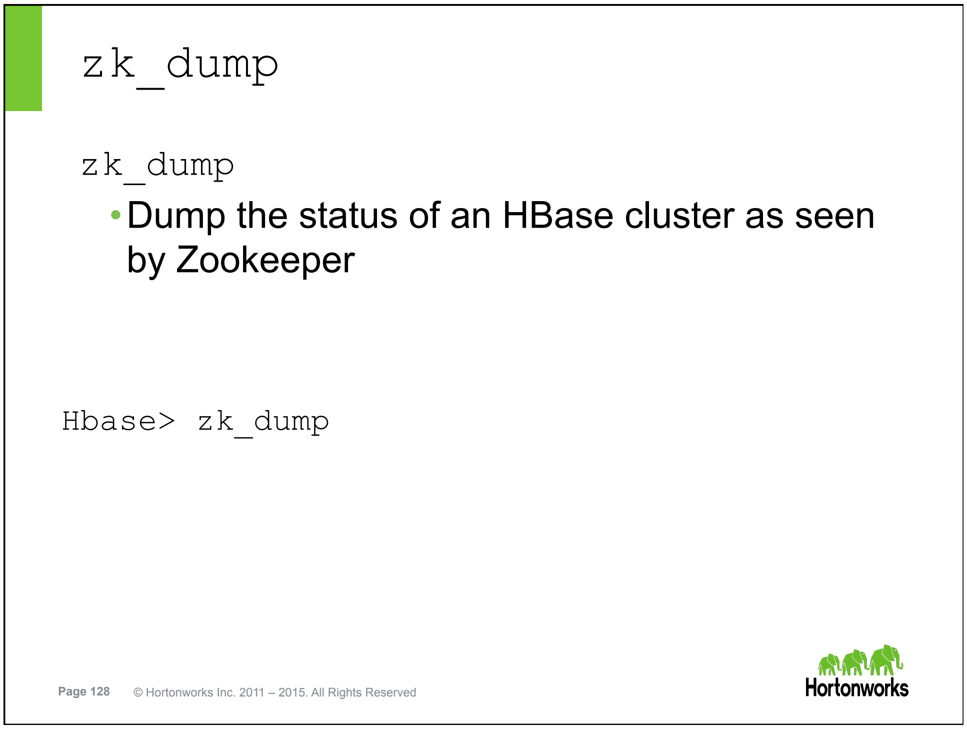
hlog_roll

`hlog_roll`

- Roll (start writing to a new log file)
- Name of regionserver is the parameter

```
hbase> hlog_roll
```

Page 127 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



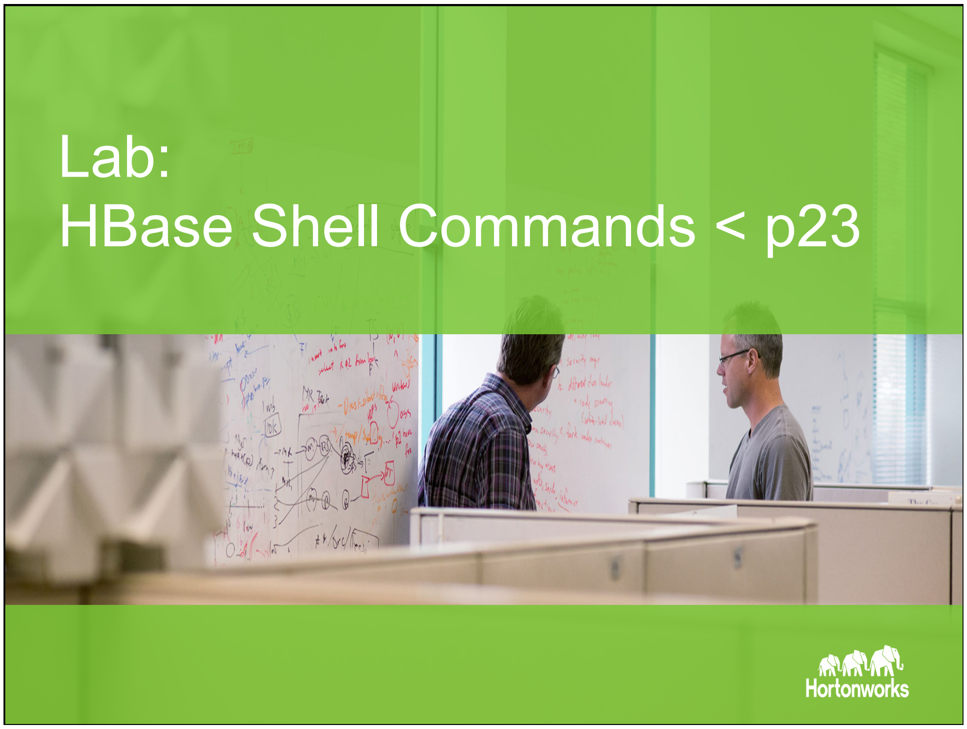
zk_dump

`zk_dump`

- Dump the status of an HBase cluster as seen by Zookeeper

```
Hbase> zk_dump
```

Page 128 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

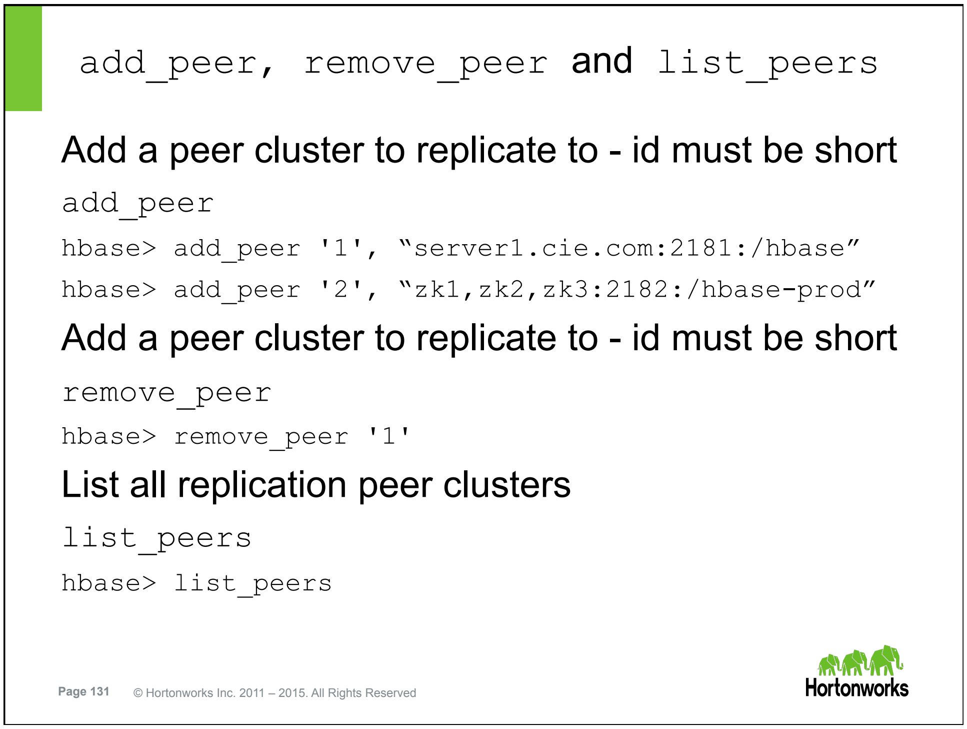


Lab:
HBase Shell Commands < p23

Hortonworks

Cluster Replication Tools

Hortonworks



add_peer, remove_peer and list_peers

Add a peer cluster to replicate to - id must be short

add_peer

```
hbase> add_peer '1', “server1.cie.com:2181:/hbase”
hbase> add_peer '2', “zk1,zk2,zk3:2182:/hbase-prod”
```

Add a peer cluster to replicate to - id must be short

remove_peer

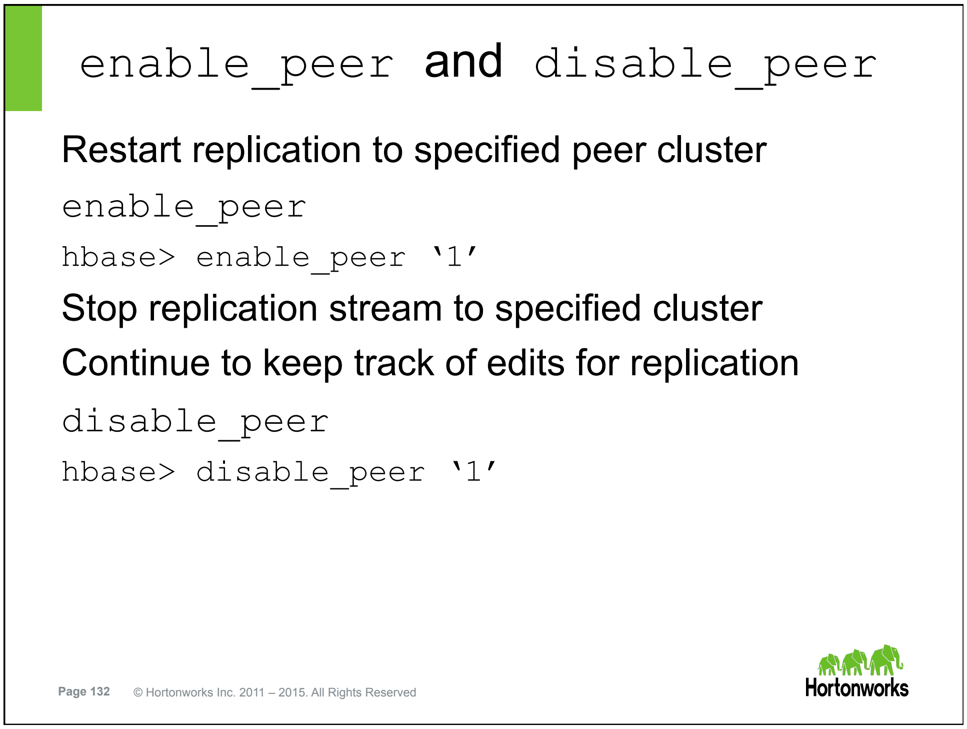
```
hbase> remove_peer '1'
```

List all replication peer clusters

list_peers

```
hbase> list_peers
```

Page 131



enable_peer and disable_peer

Restart replication to specified peer cluster

enable_peer

```
hbase> enable_peer ‘1’
```

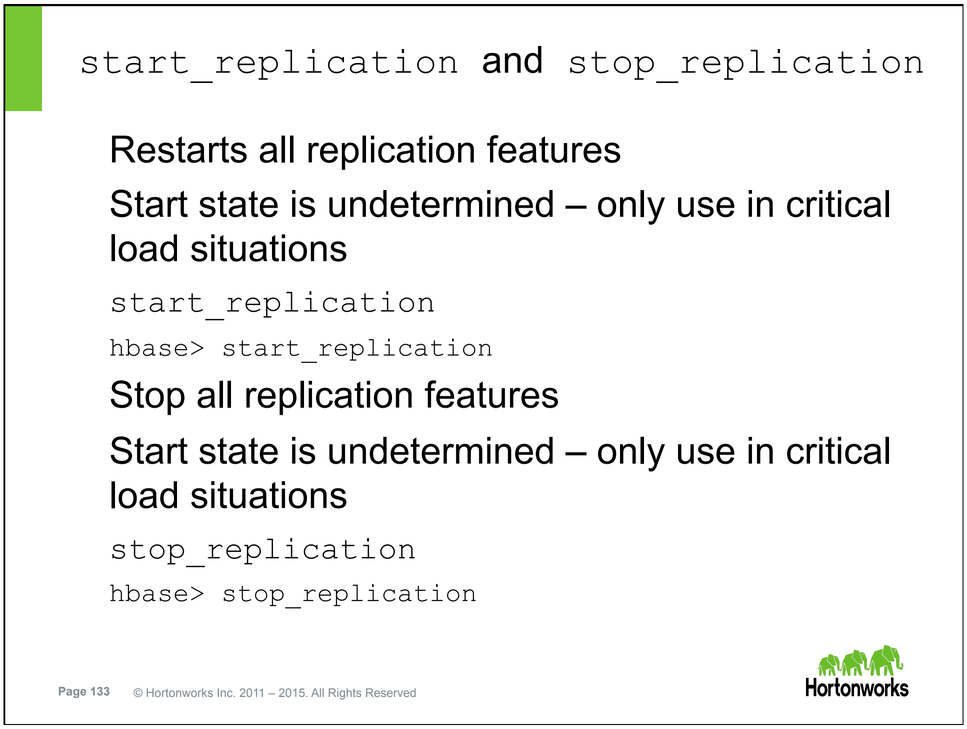
Stop replication stream to specified cluster

Continue to keep track of edits for replication

disable_peer

```
hbase> disable_peer ‘1’
```

Page 132



`start_replication` and `stop_replication`

Restarts all replication features

Start state is undetermined – only use in critical load situations

`start_replication`

```
hbase> start_replication
```

Stop all replication features

Start state is undetermined – only use in critical load situations

`stop_replication`

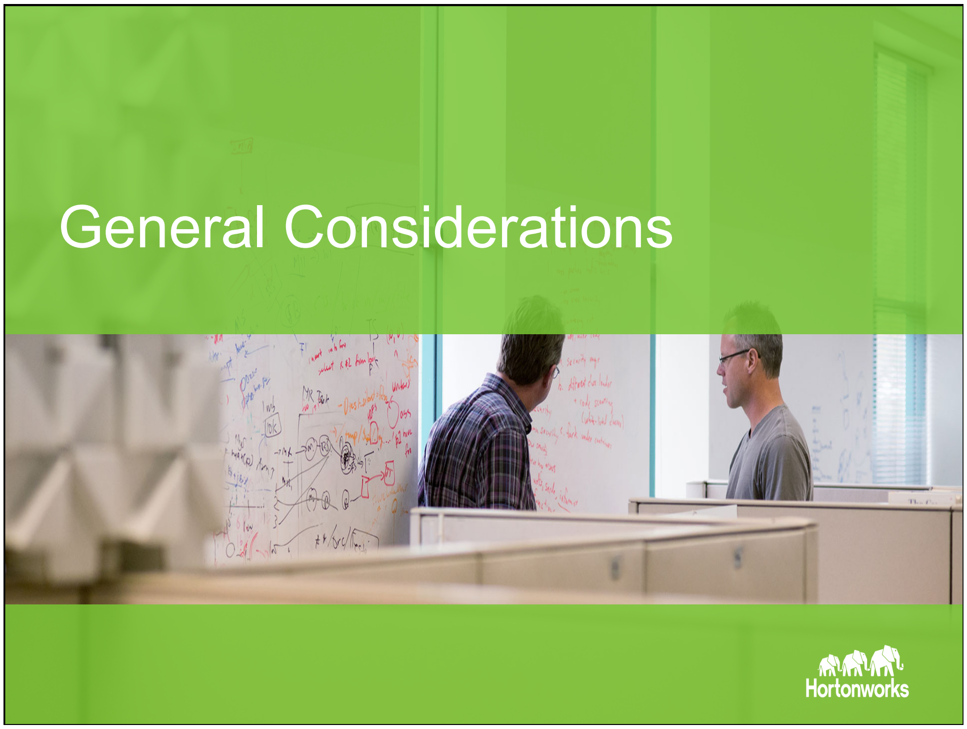
```
hbase> stop_replication
```

Page 133 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

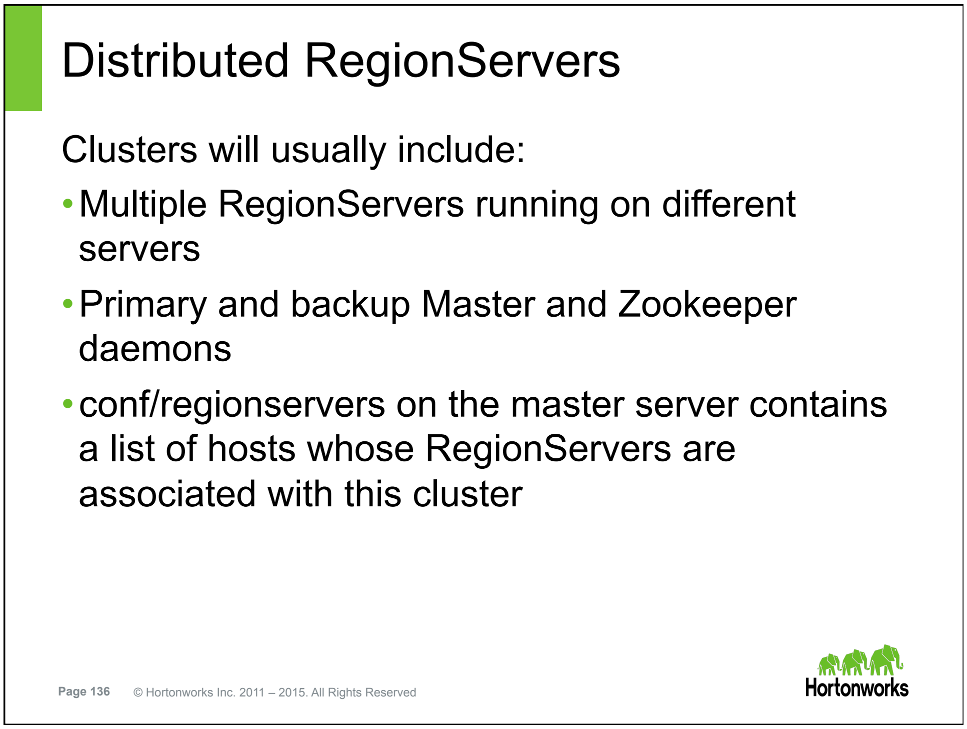
Hortonworks

Hortonworks

HBase
Installation and Configuration



General Considerations

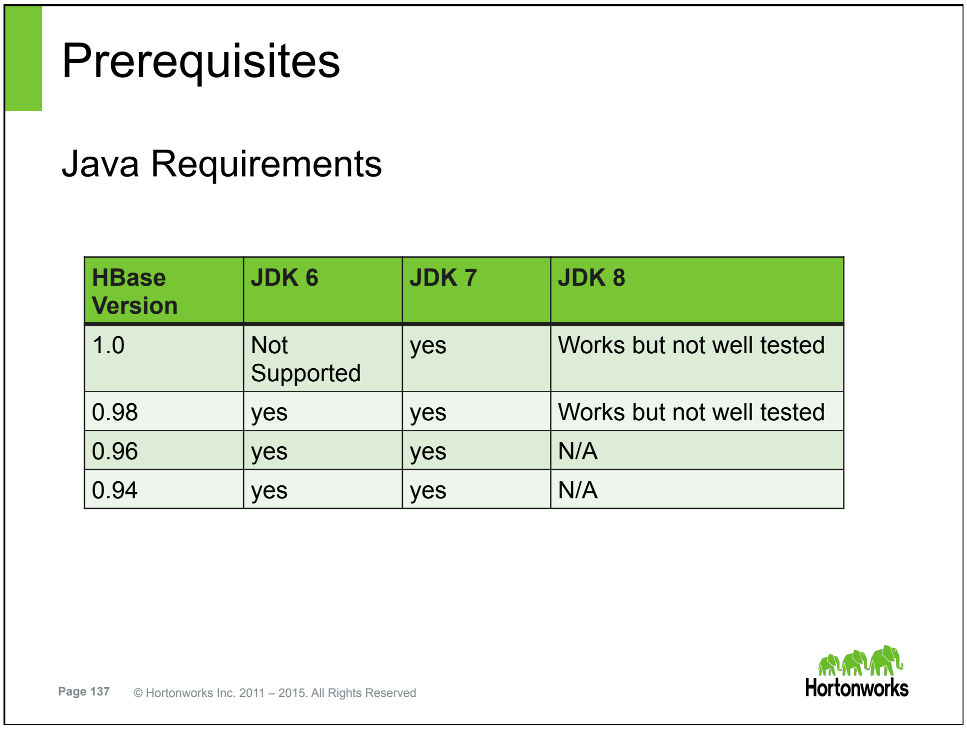


Distributed RegionServers

Clusters will usually include:

- Multiple RegionServers running on different servers
- Primary and backup Master and Zookeeper daemons
- conf/regionservers on the master server contains a list of hosts whose RegionServers are associated with this cluster

Page 136

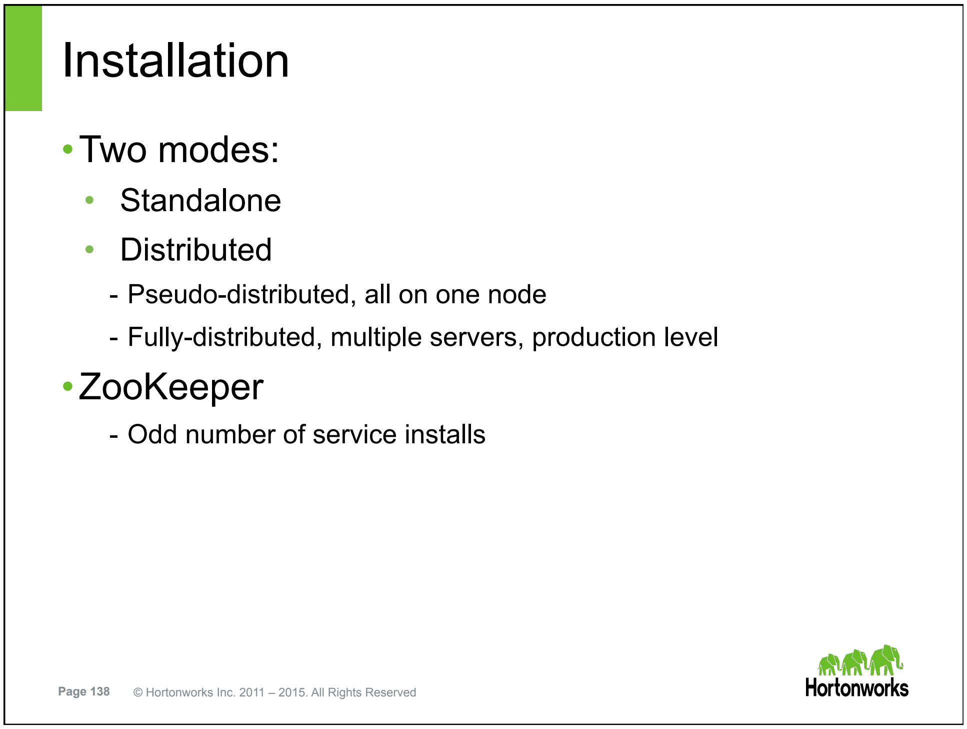


Prerequisites

Java Requirements

HBase Version	JDK 6	JDK 7	JDK 8
1.0	Not Supported	yes	Works but not well tested
0.98	yes	yes	Works but not well tested
0.96	yes	yes	N/A
0.94	yes	yes	N/A

Page 137

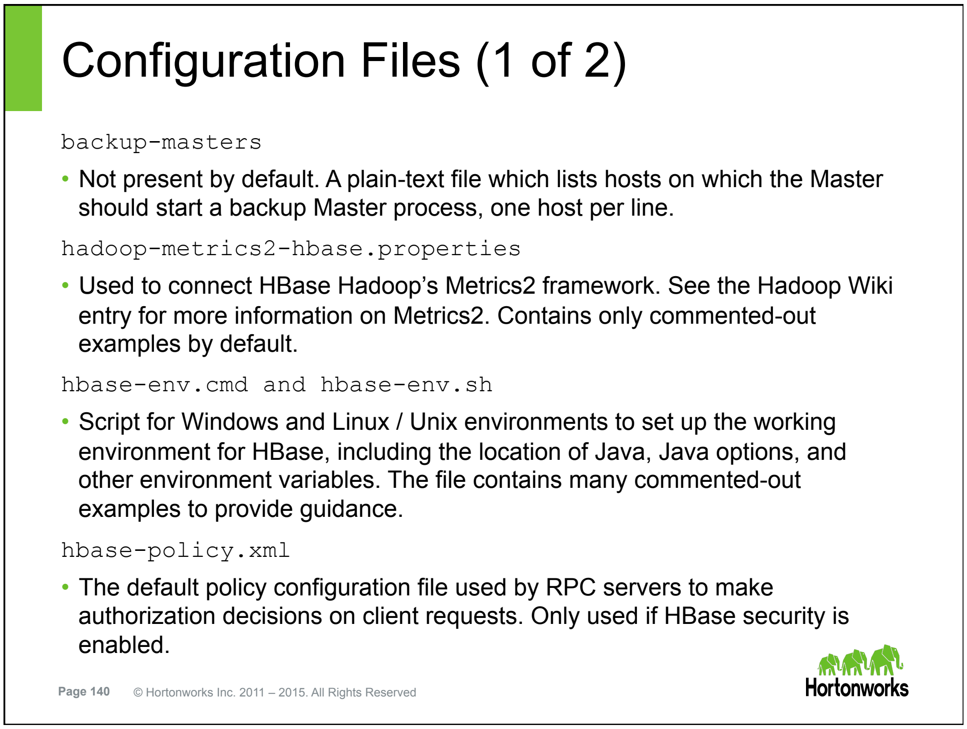


Installation

- Two modes:
  - Standalone
  - Distributed
    - Pseudo-distributed, all on one node
    - Fully-distributed, multiple servers, production level
- ZooKeeper
  - Odd number of service installs

Page 138

HBase Configuration



Configuration Files (1 of 2)

`backup-masters`

- Not present by default. A plain-text file which lists hosts on which the Master should start a backup Master process, one host per line.

`hadoop-metrics2-hbase.properties`

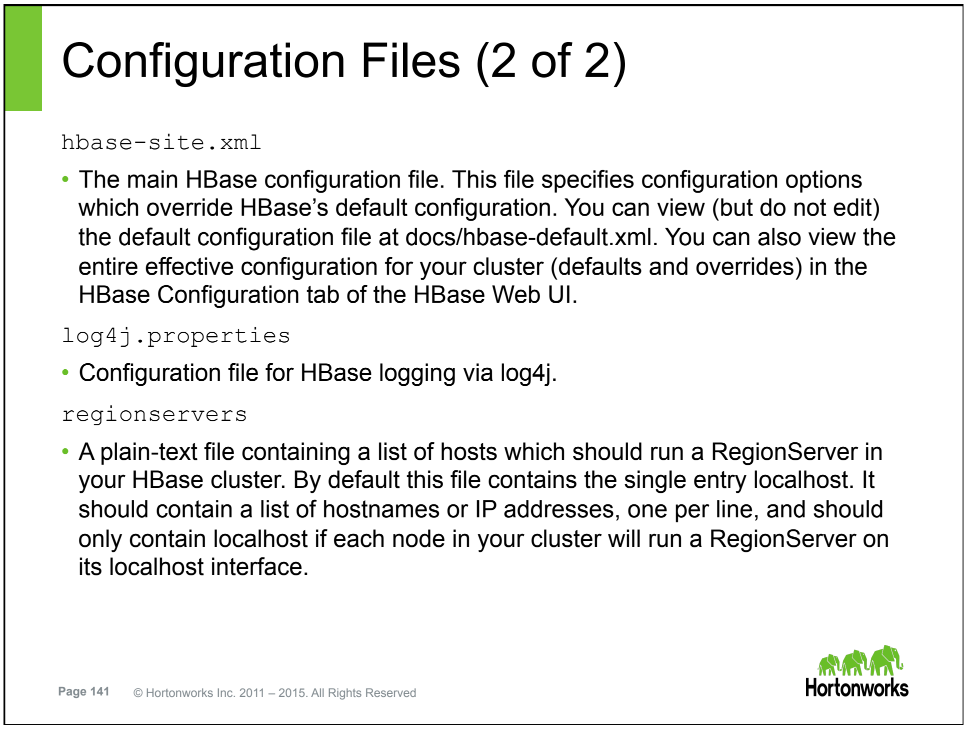
- Used to connect HBase Hadoop's Metrics2 framework. See the Hadoop Wiki entry for more information on Metrics2. Contains only commented-out examples by default.

`hbase-env.cmd and hbase-env.sh`

- Script for Windows and Linux / Unix environments to set up the working environment for HBase, including the location of Java, Java options, and other environment variables. The file contains many commented-out examples to provide guidance.

`hbase-policy.xml`

- The default policy configuration file used by RPC servers to make authorization decisions on client requests. Only used if HBase security is enabled.



Configuration Files (2 of 2)

`hbase-site.xml`

- The main HBase configuration file. This file specifies configuration options which override HBase’s default configuration. You can view (but do not edit) the default configuration file at docs/hbase-default.xml. You can also view the entire effective configuration for your cluster (defaults and overrides) in the HBase Configuration tab of the HBase Web UI.

`log4j.properties`

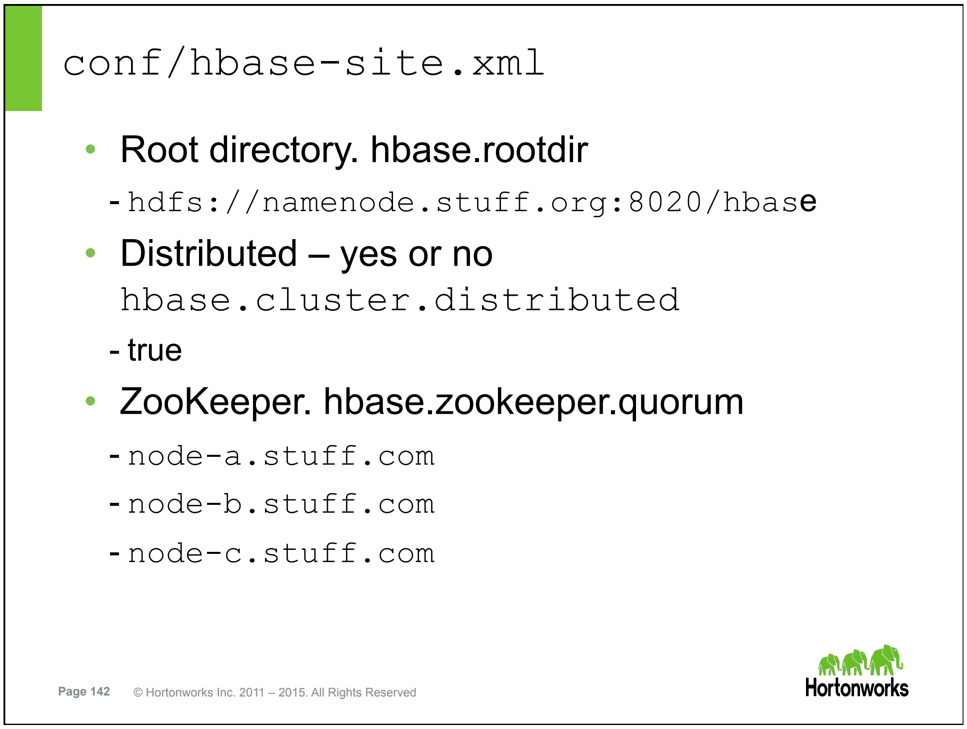
- Configuration file for HBase logging via log4j.

`regionservers`

- A plain-text file containing a list of hosts which should run a RegionServer in your HBase cluster. By default this file contains the single entry localhost. It should contain a list of hostnames or IP addresses, one per line, and should only contain localhost if each node in your cluster will run a RegionServer on its localhost interface.

Page 141

Hortonworks

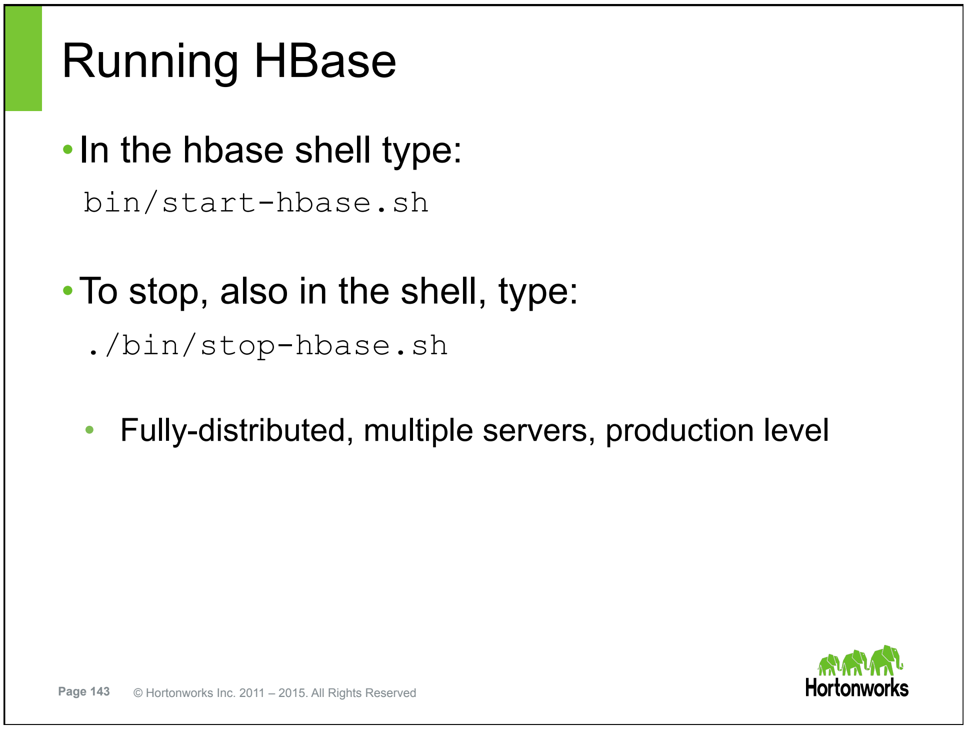


`conf/hbase-site.xml`

- Root directory. hbase.rootdir
  - `hdfs://namenode.stuff.org:8020/hbase`
- Distributed – yes or no
  `hbase.cluster.distributed`
  - true
- ZooKeeper. hbase.zookeeper.quorum
  - `node-a.stuff.com`
  - `node-b.stuff.com`
  - `node-c.stuff.com`

Page 142

Hortonworks



Running HBase

- In the hbase shell type:

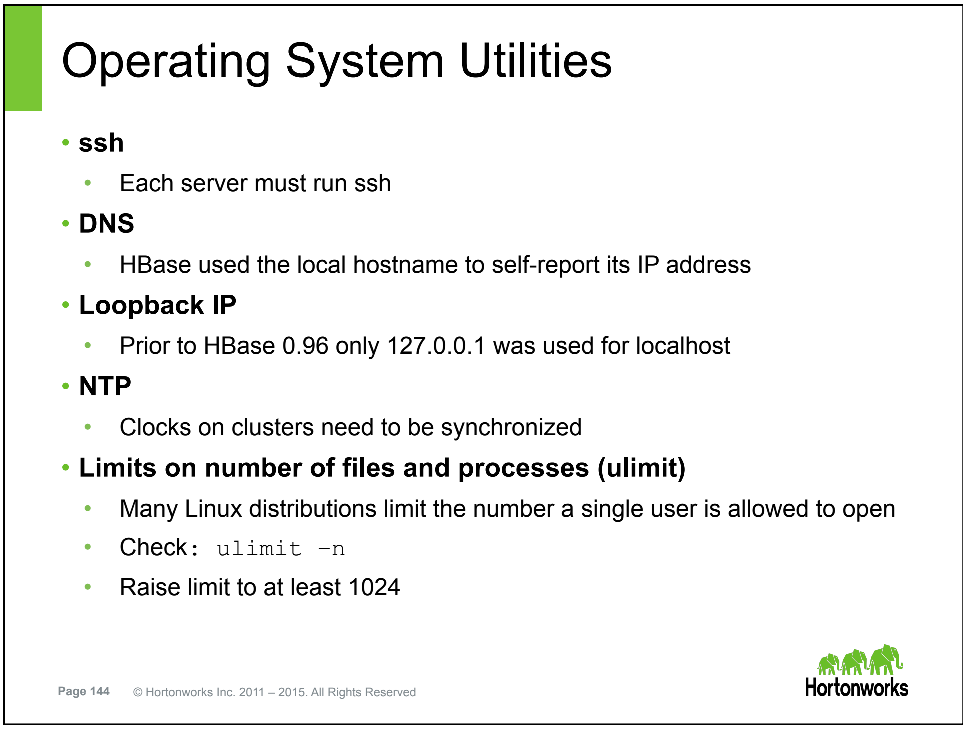
```
bin/start-hbase.sh
```

- To stop, also in the shell, type:

```
./bin/stop-hbase.sh
```

  - Fully-distributed, multiple servers, production level

Page 143

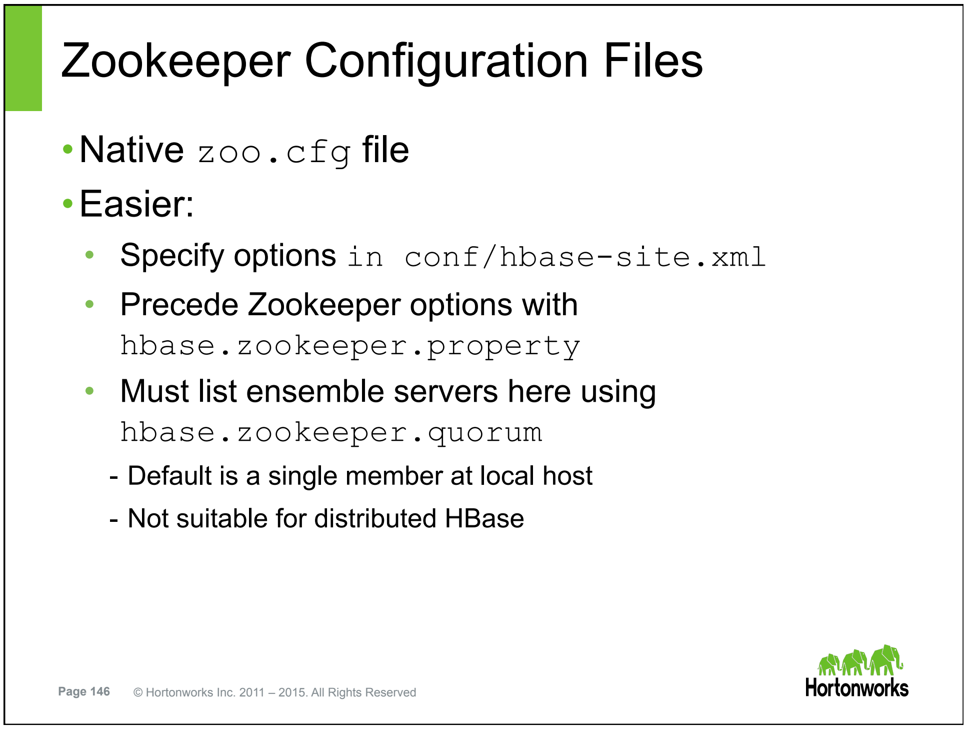


Operating System Utilities

- **ssh**
  - Each server must run ssh
- **DNS**
  - HBase used the local hostname to self-report its IP address
- **Loopback IP**
  - Prior to HBase 0.96 only 127.0.0.1 was used for localhost
- **NTP**
  - Clocks on clusters need to be synchronized
- **Limits on number of files and processes (ulimit)**
  - Many Linux distributions limit the number a single user is allowed to open
  - Check: `ulimit -n`
  - Raise limit to at least 1024

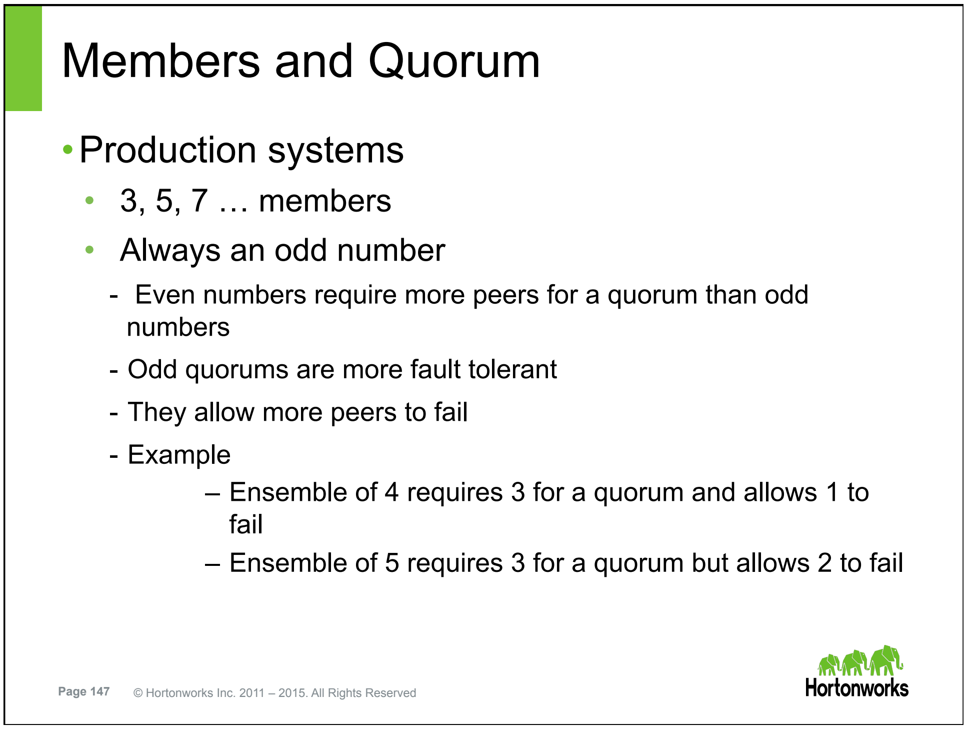
Page 144

Configuring Zookeeper



Zookeeper Configuration Files

- Native `zoo.cfg` file
- Easier:
  - Specify options `in conf/hbase-site.xml`
  - Precede Zookeeper options with `hbase.zookeeper.property`
  - Must list ensemble servers here using `hbase.zookeeper.quorum`
    - Default is a single member at local host
    - Not suitable for distributed HBase

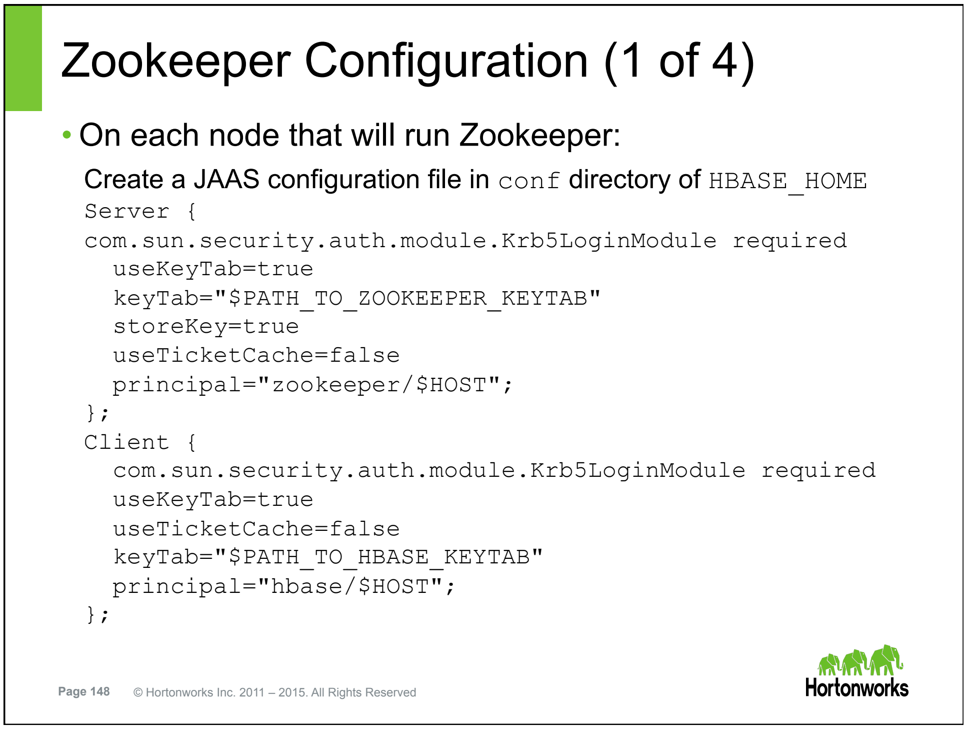


Members and Quorum

- Production systems
  - 3, 5, 7 … members
  - Always an odd number
    - Even numbers require more peers for a quorum than odd numbers
    - Odd quorums are more fault tolerant
    - They allow more peers to fail
    - Example
      - Ensemble of 4 requires 3 for a quorum and allows 1 to fail
      - Ensemble of 5 requires 3 for a quorum but allows 2 to fail

Page 147

Hortonworks



Zookeeper Configuration (1 of 4)

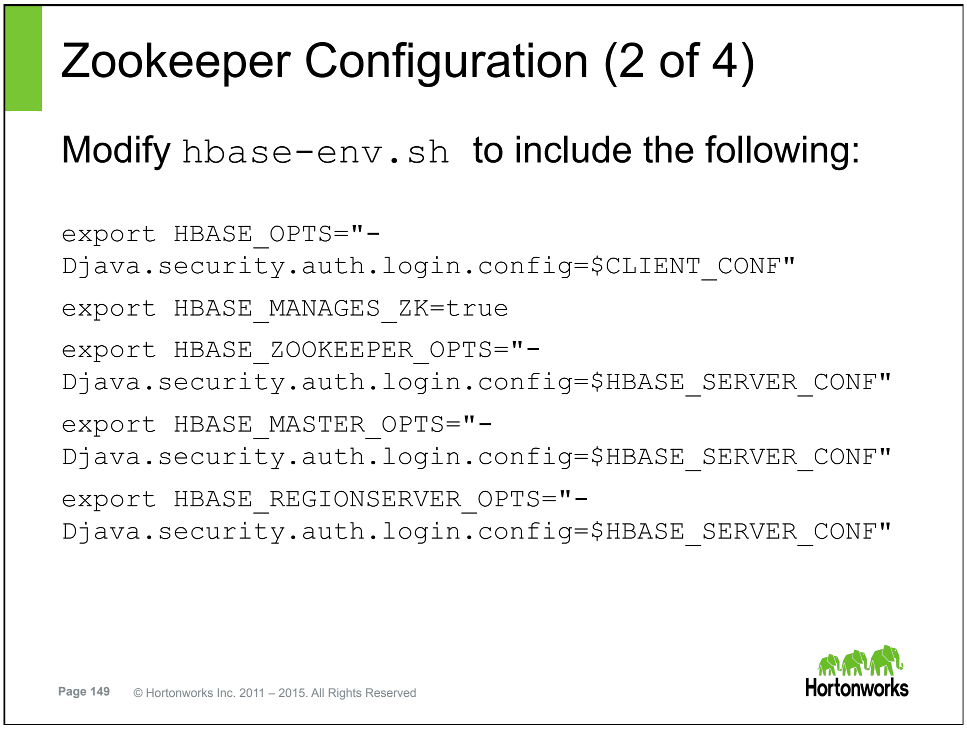
- On each node that will run Zookeeper:

Create a JAAS configuration file in `conf` directory of `HBASE_HOME`

```
Server {
com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  keyTab="$PATH_TO_ZOOKEEPER_KEYTAB"
  storeKey=true
  useTicketCache=false
  principal="zookeeper/$HOST";
};
Client {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  useTicketCache=false
  keyTab="$PATH_TO_HBASE_KEYTAB"
  principal="hbase/$HOST";
};
```

Page 148

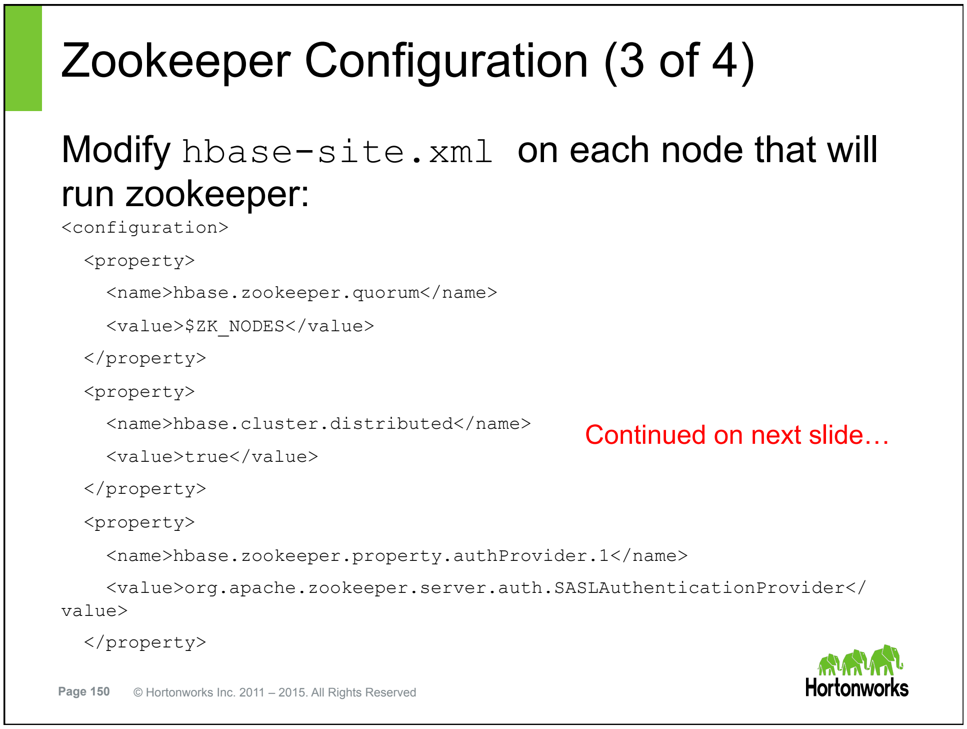
Hortonworks



Zookeeper Configuration (2 of 4)

Modify `hbase-env.sh` to include the following:

```
export HBASE_OPTS="-
Djava.security.auth.login.config=$CLIENT_CONF"
export HBASE_MANAGES_ZK=true
export HBASE_ZOOKEEPER_OPTS="-
Djava.security.auth.login.config=$HBASE_SERVER_CONF"
export HBASE_MASTER_OPTS="-
Djava.security.auth.login.config=$HBASE_SERVER_CONF"
export HBASE_REGIONSERVER_OPTS="-
Djava.security.auth.login.config=$HBASE_SERVER_CONF"
```

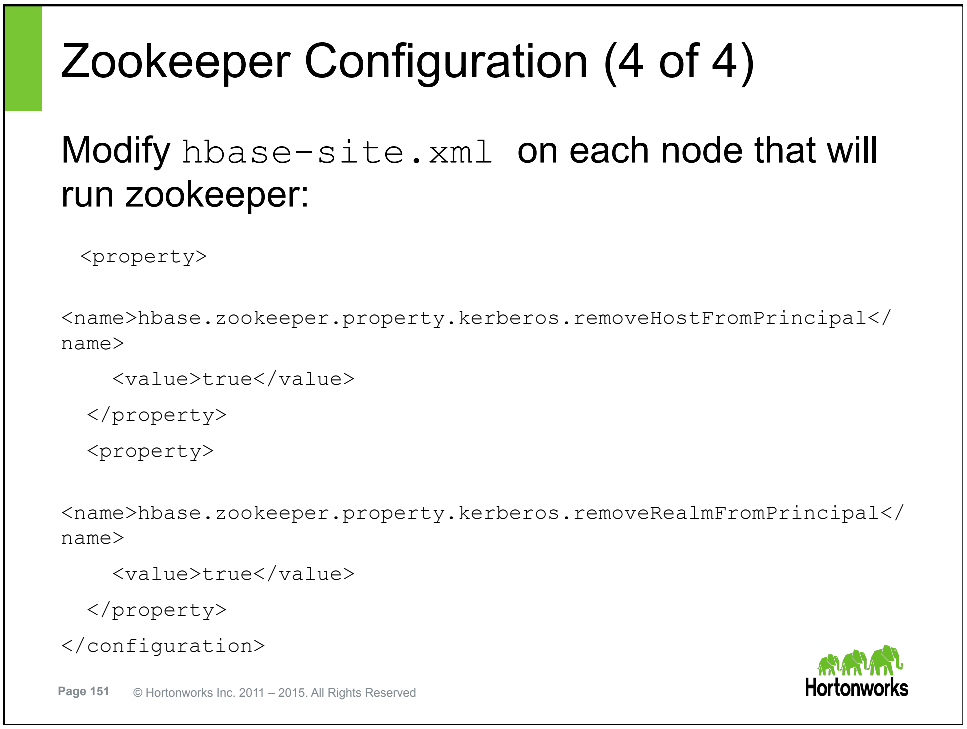


Zookeeper Configuration (3 of 4)

Modify `hbase-site.xml` on each node that will run zookeeper:

```
<configuration>
  <property>
    <name>hbase.zookeeper.quorum</name>
    <value>$ZK_NODES</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.authProvider.1</name>
    <value>org.apache.zookeeper.server.auth.SASLAuthenticationProvider</
value>
  </property>
```

Continued on next slide…

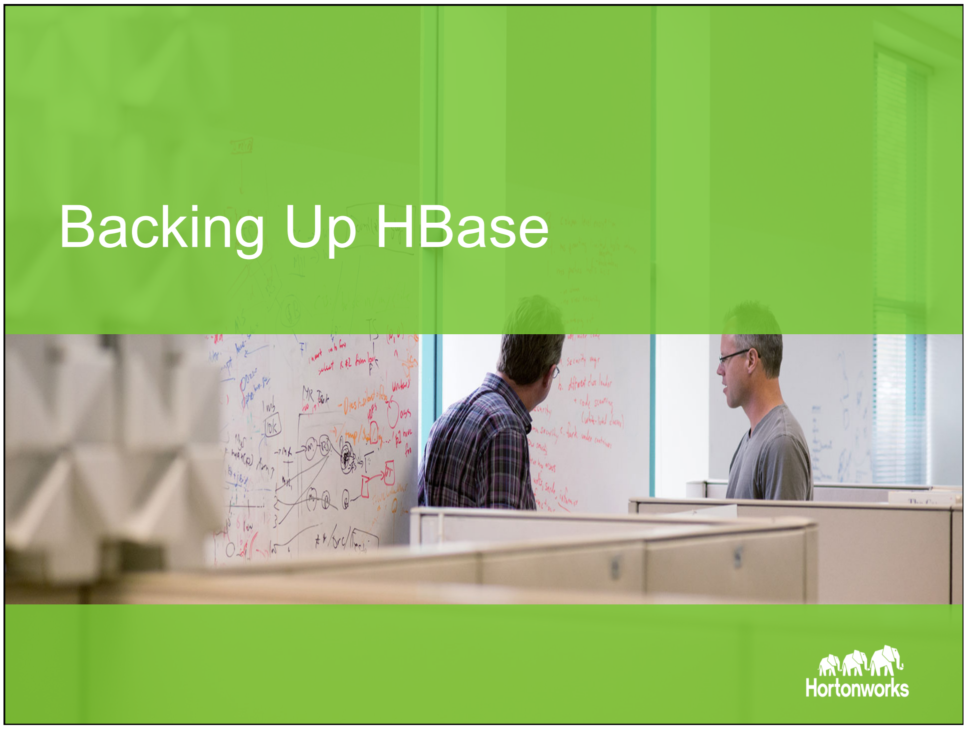


Zookeeper Configuration (4 of 4)

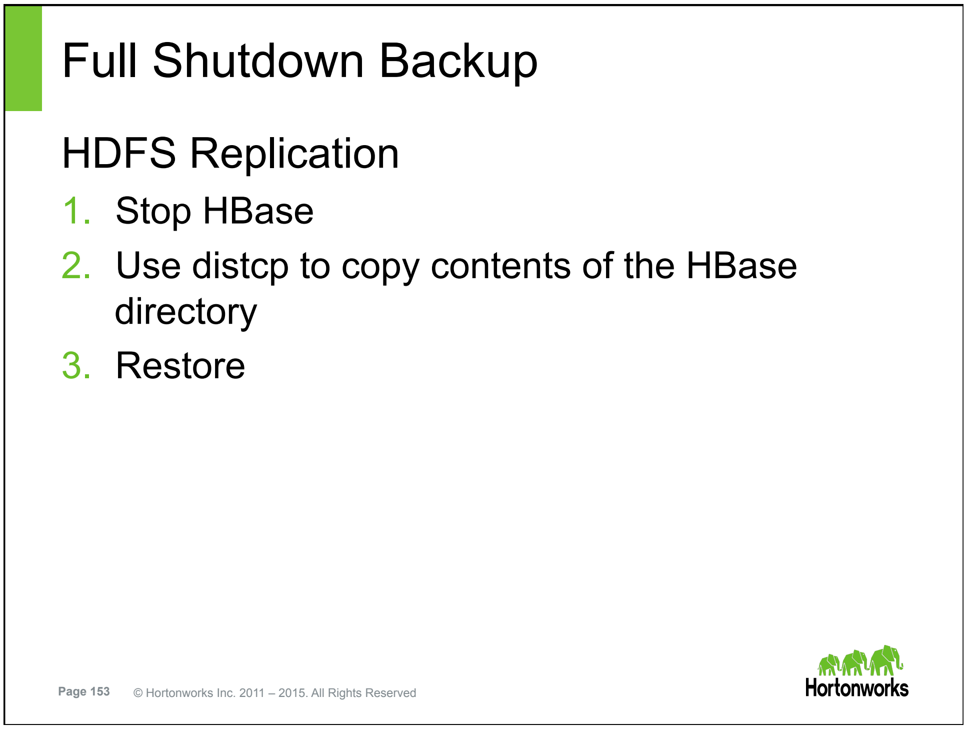
Modify `hbase-site.xml` on each node that will run zookeeper:

```
  <property>

<name>hbase.zookeeper.property.kerberos.removeHostFromPrincipal</
name>
    <value>true</value>
  </property>
  <property>

<name>hbase.zookeeper.property.kerberos.removeRealmFromPrincipal</
name>
    <value>true</value>
  </property>
</configuration>
```

Page 151 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Backing Up HBase



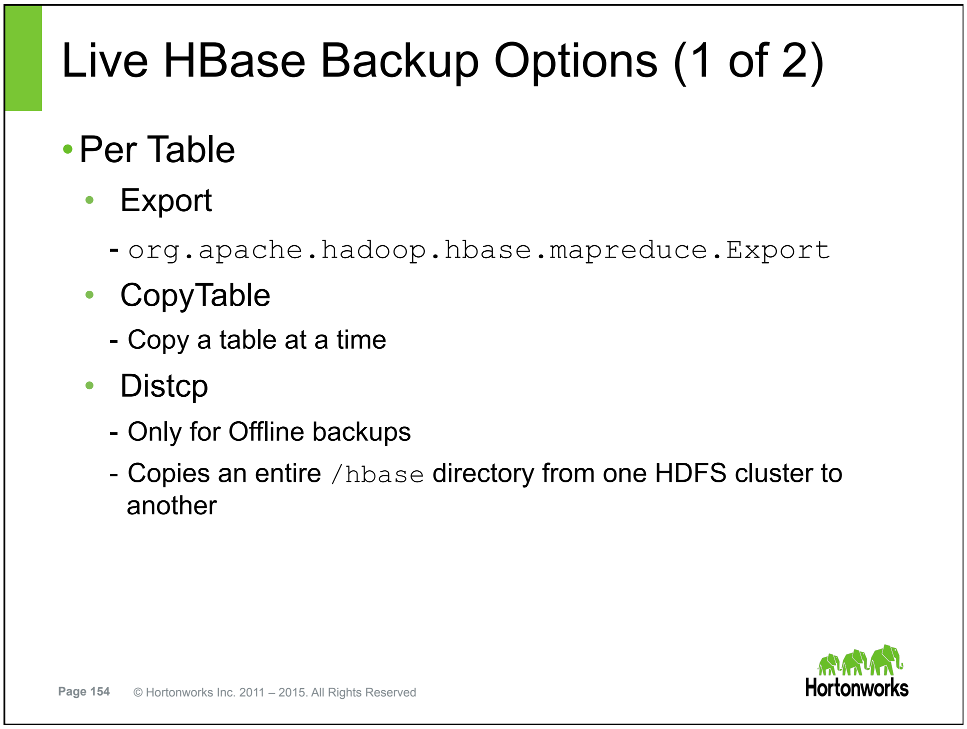
Full Shutdown Backup

HDFS Replication

1. Stop HBase
2. Use distcp to copy contents of the HBase directory
3. Restore

Page 153

Hortonworks

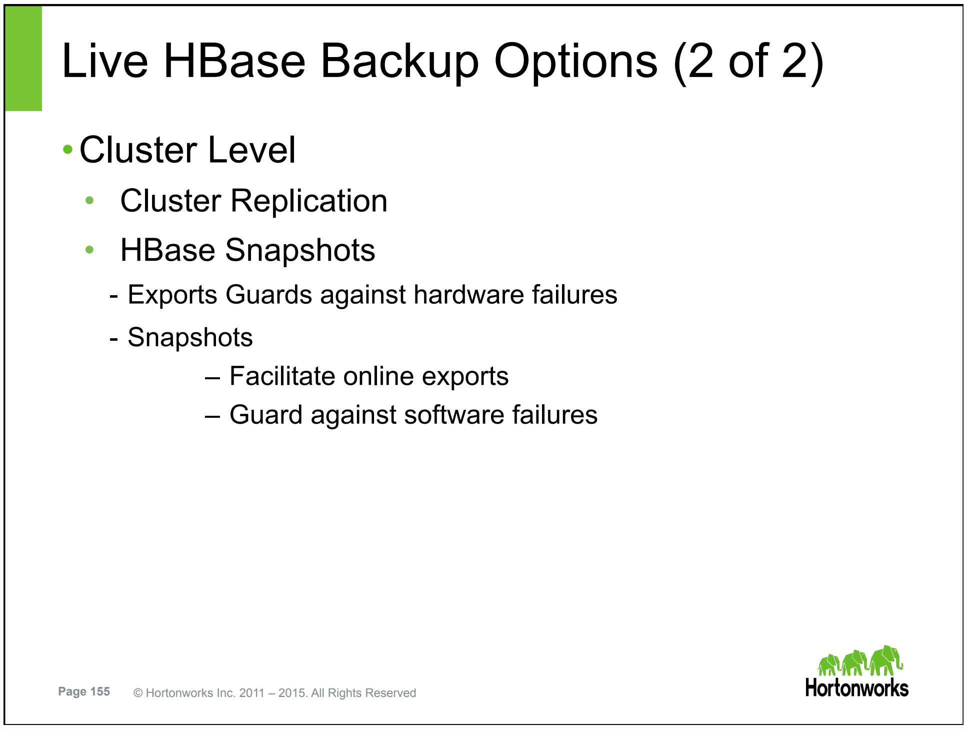


Live HBase Backup Options (1 of 2)

- Per Table
  - Export
    - `org.apache.hadoop.hbase.mapreduce.Export`
  - CopyTable
    - Copy a table at a time
  - Distcp
    - Only for Offline backups
    - Copies an entire `/hbase` directory from one HDFS cluster to another

Page 154

Hortonworks

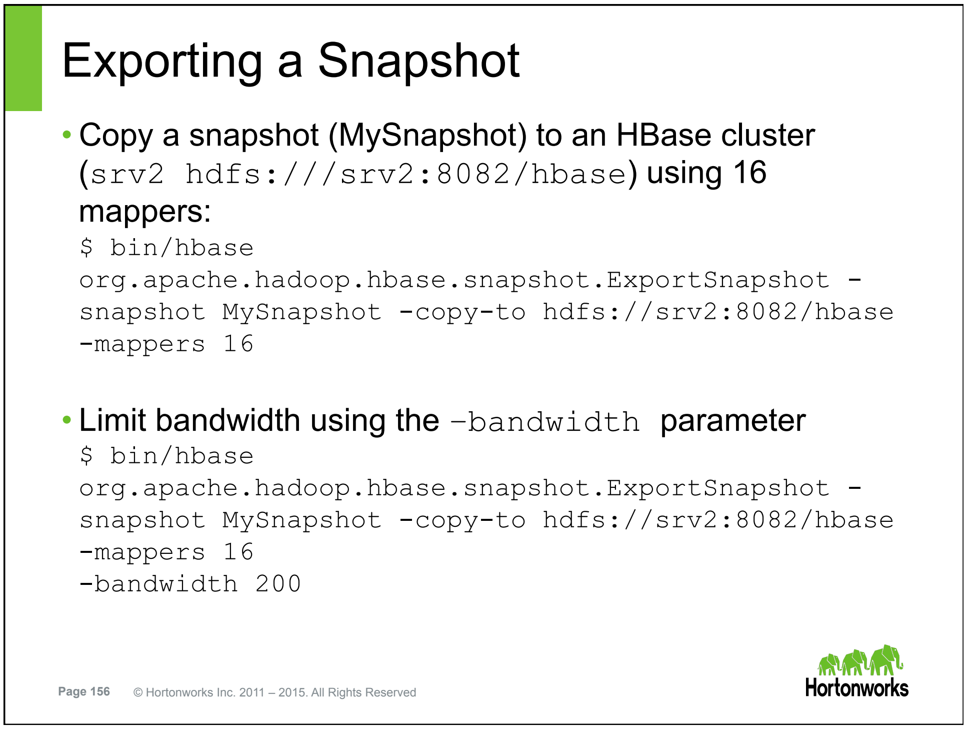


Live HBase Backup Options (2 of 2)

- Cluster Level
  - Cluster Replication
  - HBase Snapshots
    - Exports Guards against hardware failures
    - Snapshots
      - Facilitate online exports
      - Guard against software failures

Page 155

Hortonworks



Exporting a Snapshot

- Copy a snapshot (MySnapshot) to an HBase cluster (`srv2 hdfs:///srv2:8082/hbase`) using 16 mappers:

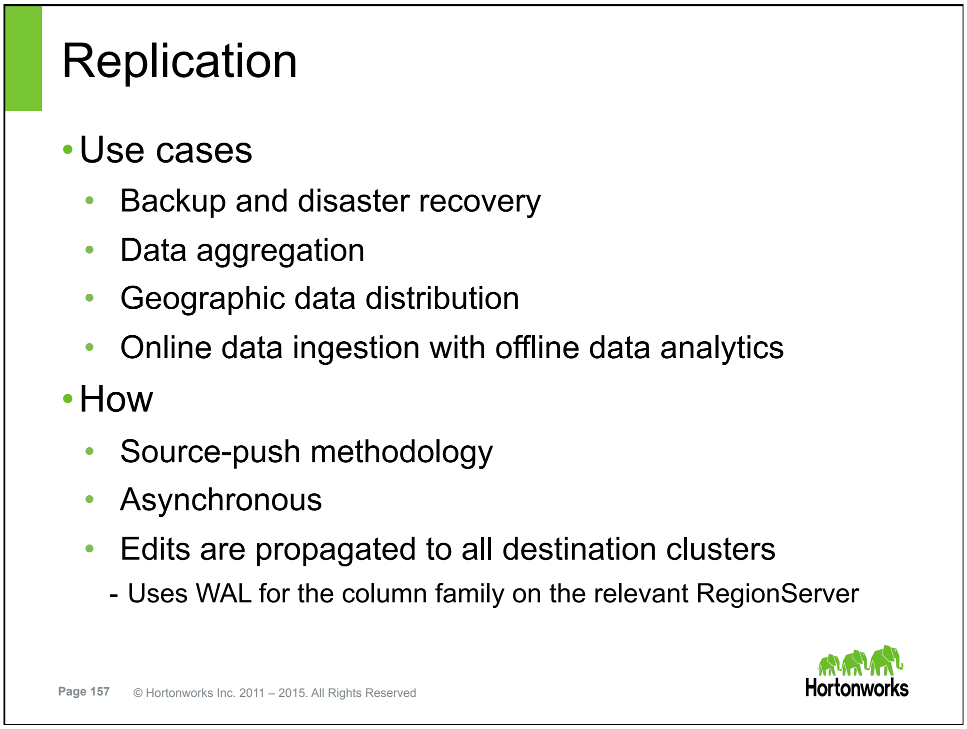
```
$ bin/hbase
org.apache.hadoop.hbase.snapshot.ExportSnapshot -
snapshot MySnapshot -copy-to hdfs://srv2:8082/hbase
-mappers 16
```

- Limit bandwidth using the `–bandwidth` parameter

```
$ bin/hbase
org.apache.hadoop.hbase.snapshot.ExportSnapshot -
snapshot MySnapshot -copy-to hdfs://srv2:8082/hbase
-mappers 16
-bandwidth 200
```

Page 156

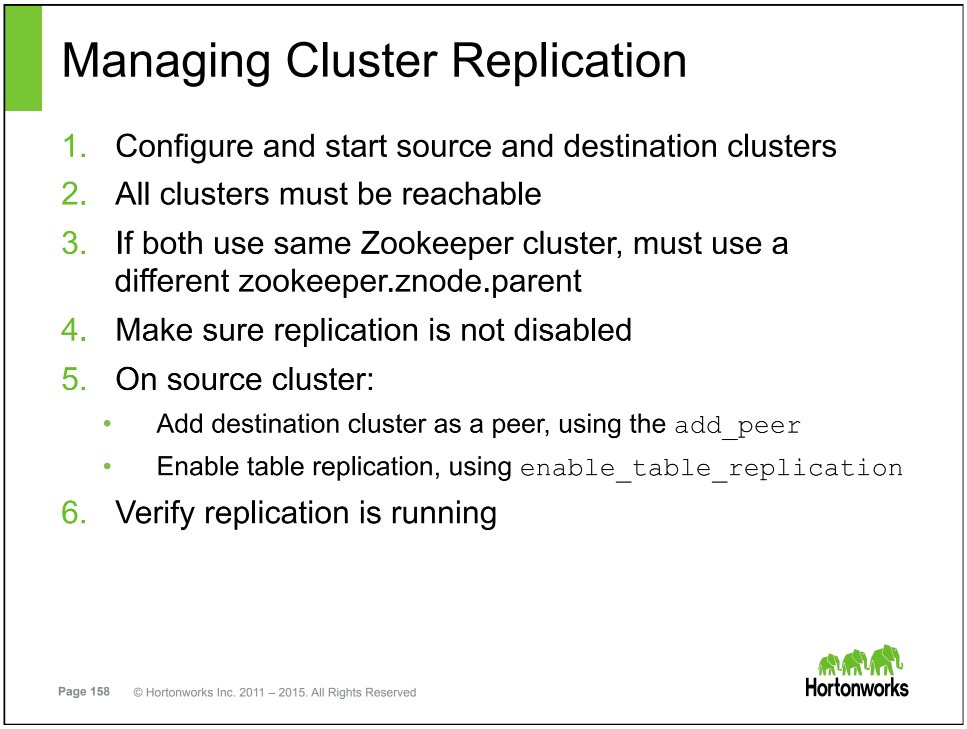
Hortonworks



Replication

- Use cases
  - Backup and disaster recovery
  - Data aggregation
  - Geographic data distribution
  - Online data ingestion with offline data analytics
- How
  - Source-push methodology
  - Asynchronous
  - Edits are propagated to all destination clusters
    - Uses WAL for the column family on the relevant RegionServer

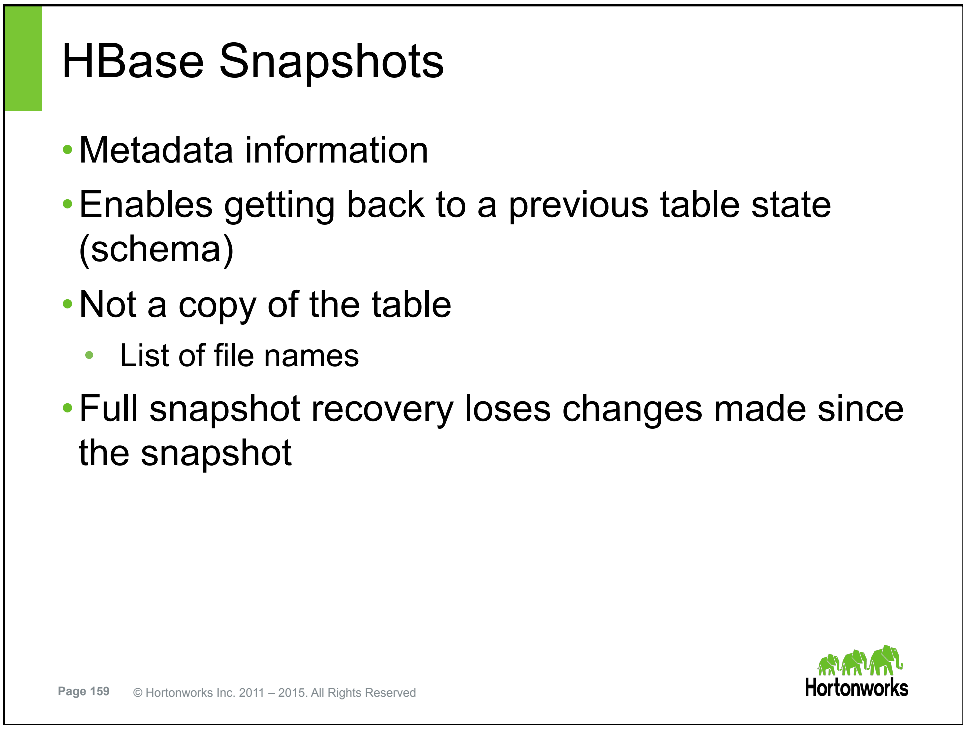
Page 157



Managing Cluster Replication

1. Configure and start source and destination clusters
2. All clusters must be reachable
3. If both use same Zookeeper cluster, must use a different zookeeper.znode.parent
4. Make sure replication is not disabled
5. On source cluster:
   - Add destination cluster as a peer, using the `add_peer`
   - Enable table replication, using `enable_table_replication`
6. Verify replication is running

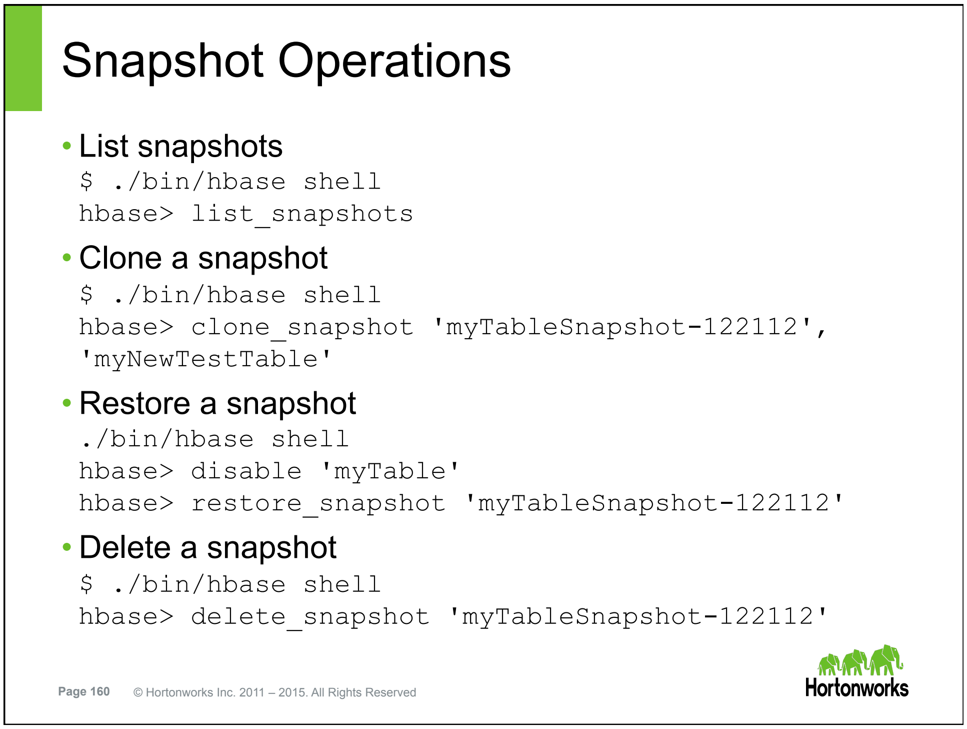
Page 158



HBase Snapshots

- Metadata information
- Enables getting back to a previous table state (schema)
- Not a copy of the table
  - List of file names
- Full snapshot recovery loses changes made since the snapshot

Page 159



Snapshot Operations

- List snapshots

```
$ ./bin/hbase shell
hbase> list_snapshots
```

- Clone a snapshot

```
$ ./bin/hbase shell
hbase> clone_snapshot 'myTableSnapshot-122112',
'myNewTestTable'
```

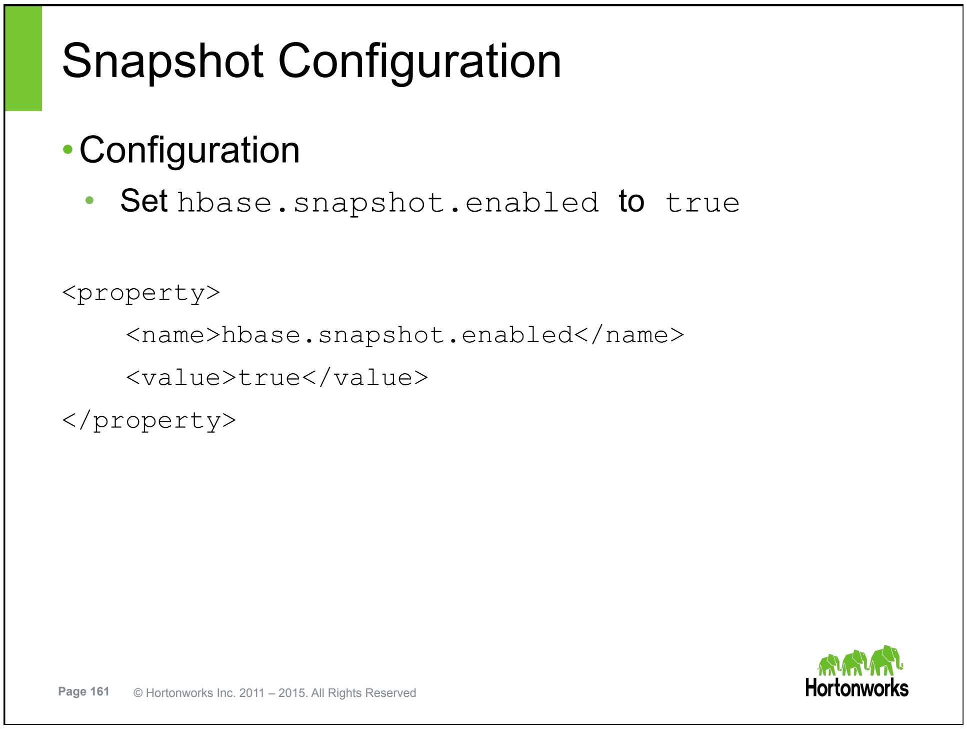
- Restore a snapshot

```
./bin/hbase shell
hbase> disable 'myTable'
hbase> restore_snapshot 'myTableSnapshot-122112'
```

- Delete a snapshot

```
$ ./bin/hbase shell
hbase> delete_snapshot 'myTableSnapshot-122112'
```

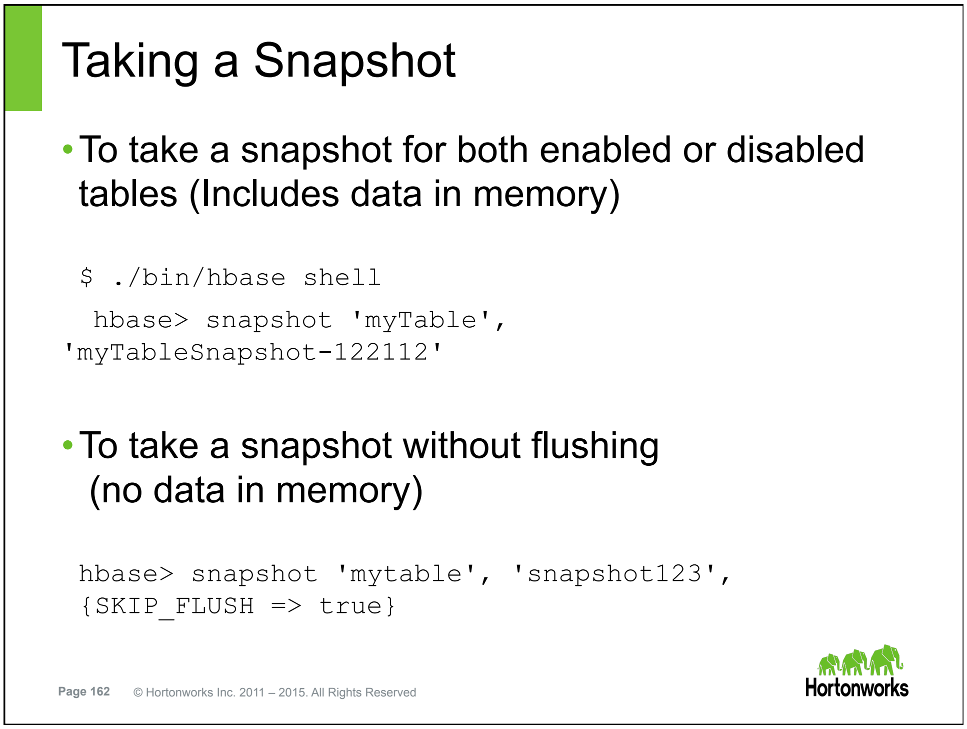
Page 160



Snapshot Configuration

- Configuration
  - Set `hbase.snapshot.enabled` to `true`

```
<property>
    <name>hbase.snapshot.enabled</name>
    <value>true</value>
</property>
```



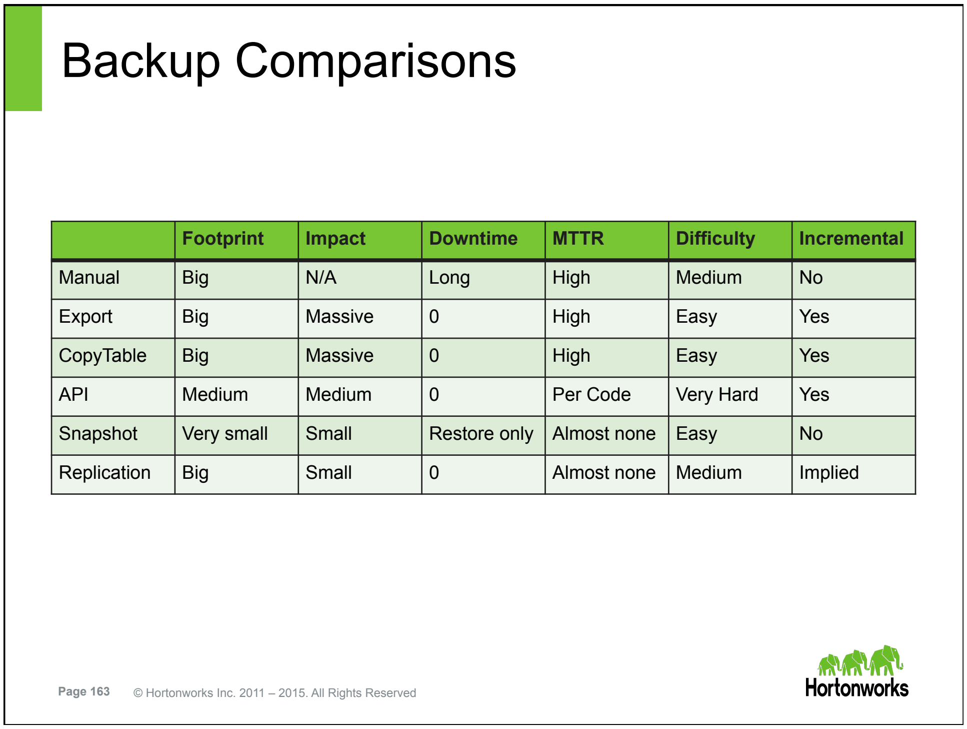
Taking a Snapshot

- To take a snapshot for both enabled or disabled tables (Includes data in memory)

```
 $ ./bin/hbase shell
  hbase> snapshot 'myTable',
'myTableSnapshot-122112'
```

- To take a snapshot without flushing (no data in memory)

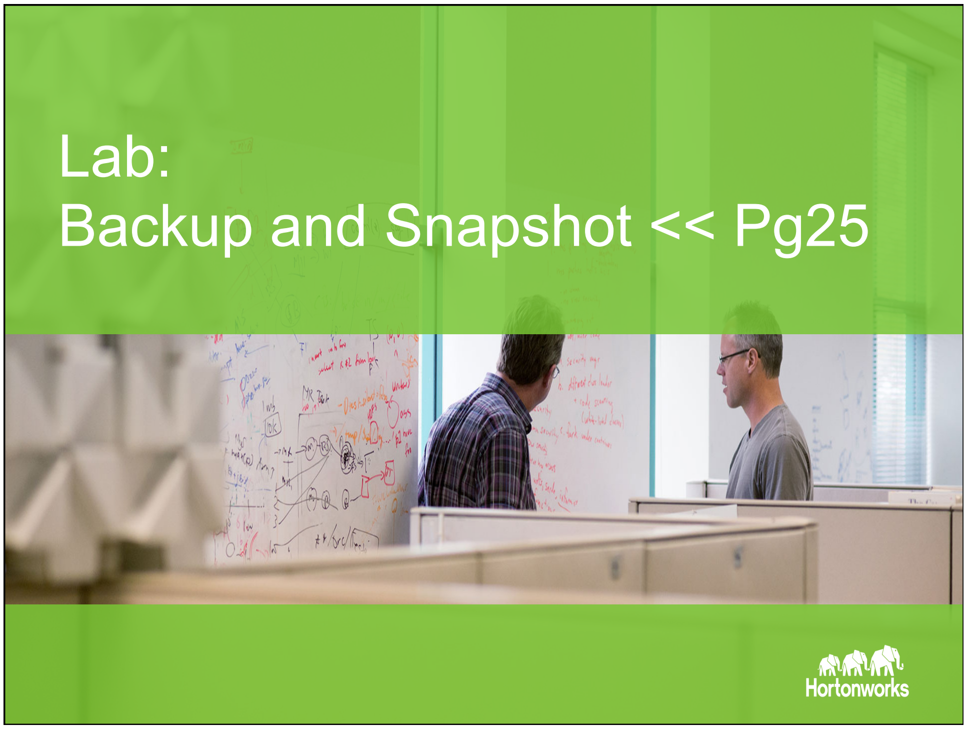
```
 hbase> snapshot 'mytable', 'snapshot123',
 {SKIP_FLUSH => true}
```



Backup Comparisons

	Footprint	Impact	Downtime	MTTR	Difficulty	Incremental
Manual	Big	N/A	Long	High	Medium	No
Export	Big	Massive	0	High	Easy	Yes
CopyTable	Big	Massive	0	High	Easy	Yes
API	Medium	Medium	0	Per Code	Very Hard	Yes
Snapshot	Very small	Small	Restore only	Almost none	Easy	No
Replication	Big	Small	0	Almost none	Medium	Implied

Page 163



Lab:
Backup and Snapshot << Pg25

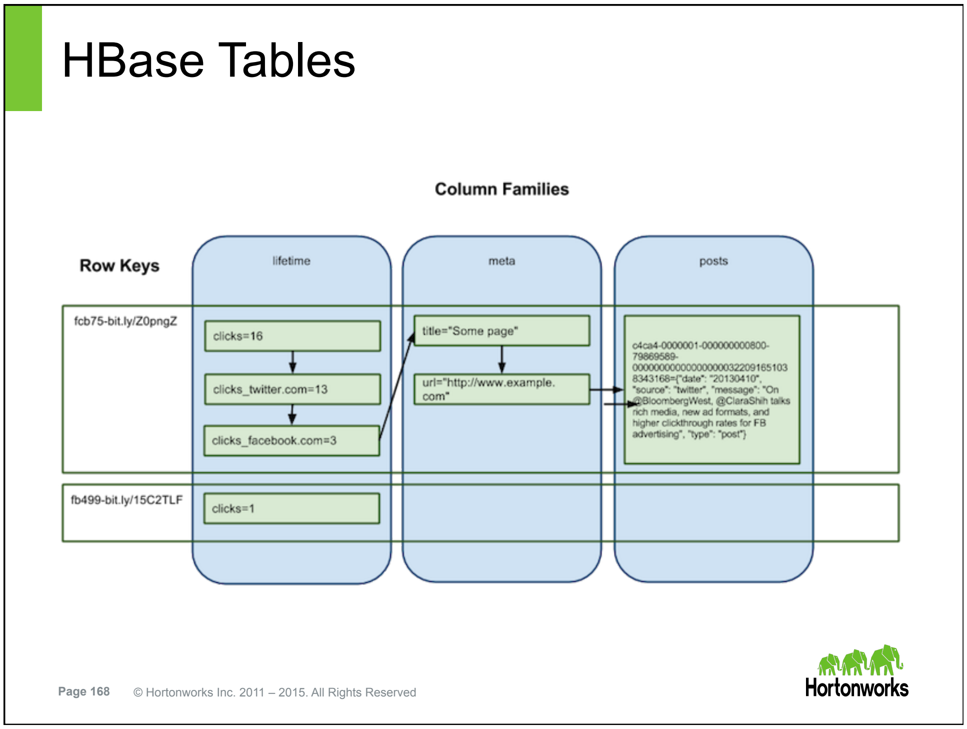
Lab: Export with Pig; Import with `ImportTSV`

Hortonworks

Hortonworks

HBase Schema Design

HBase Data Model



HBase Tables

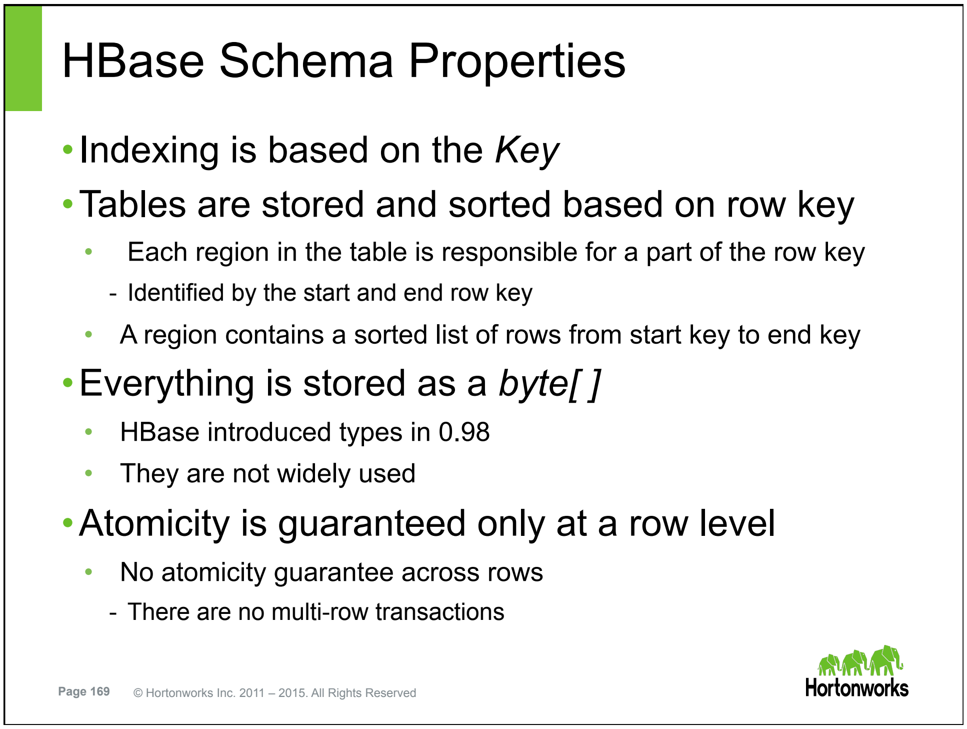
Column Families

Row Keys	lifetime	meta	posts

fcb75-bit.ly/Z0pngZ	clicks=16
	clicks_twitter.com=13
	clicks_facebook.com=3
		title="Some page"
		url="http://www.example.com"
			c4ca4-0000001-000000000800-79869589-0000000000000000000032209165103 8343168={"date": "20130410", "source": "twitter", "message": "On @BloombergWest, @ClaraShih talks rich media, new ad formats, and higher clickthrough rates for FB advertising", "type": "post"}

fb499-bit.ly/15C2TLF	clicks=1		

Page 168 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

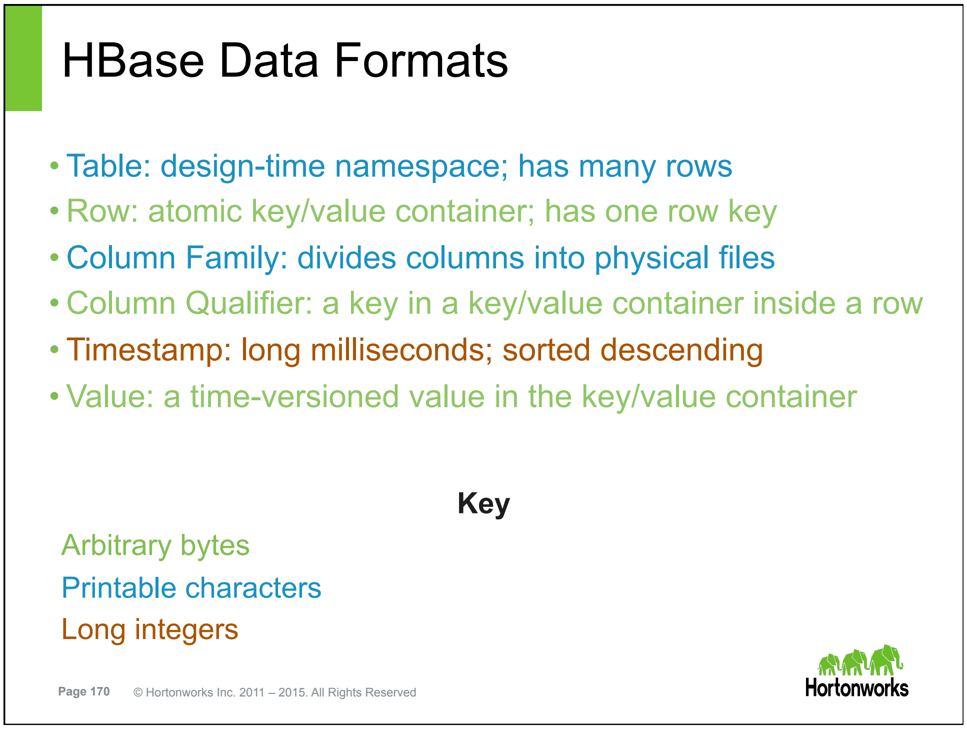


HBase Schema Properties

- Indexing is based on the *Key*
- Tables are stored and sorted based on row key
  - Each region in the table is responsible for a part of the row key
    - Identified by the start and end row key
  - A region contains a sorted list of rows from start key to end key
- Everything is stored as a *byte[ ]*
  - HBase introduced types in 0.98
  - They are not widely used
- Atomicity is guaranteed only at a row level
  - No atomicity guarantee across rows
    - There are no multi-row transactions

Page 169

Hortonworks



HBase Data Formats

- Table: design-time namespace; has many rows
- Row: atomic key/value container; has one row key
- Column Family: divides columns into physical files
- Column Qualifier: a key in a key/value container inside a row
- Timestamp: long milliseconds; sorted descending
- Value: a time-versioned value in the key/value container

Key

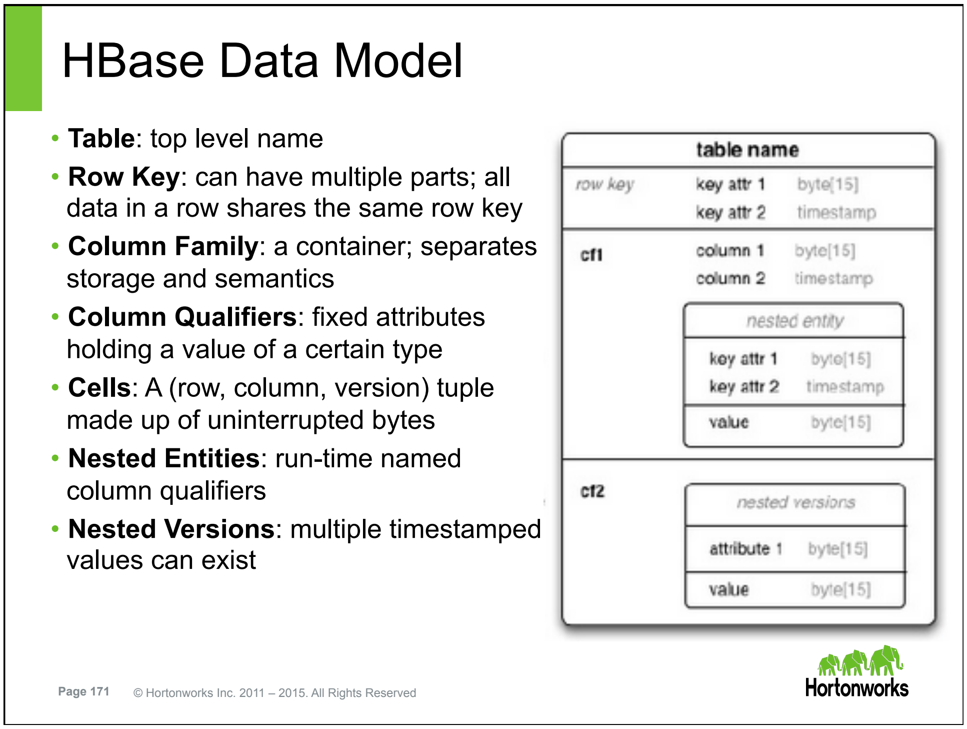
Arbitrary bytes

Printable characters

Long integers

Page 170

Hortonworks



HBase Data Model

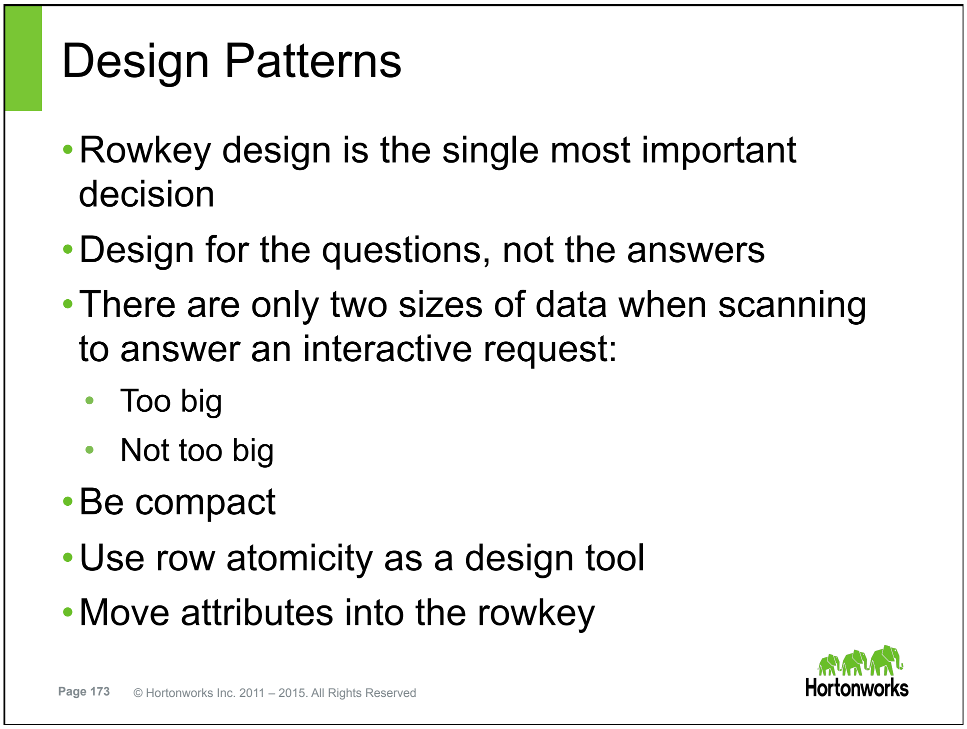
- **Table**: top level name
- **Row Key**: can have multiple parts; all data in a row shares the same row key
- **Column Family**: a container; separates storage and semantics
- **Column Qualifiers**: fixed attributes holding a value of a certain type
- **Cells**: A (row, column, version) tuple made up of uninterrupted bytes
- **Nested Entities**: run-time named column qualifiers
- **Nested Versions**: multiple timestamped values can exist

Page 171

Hortonworks

Designing Rowkeys

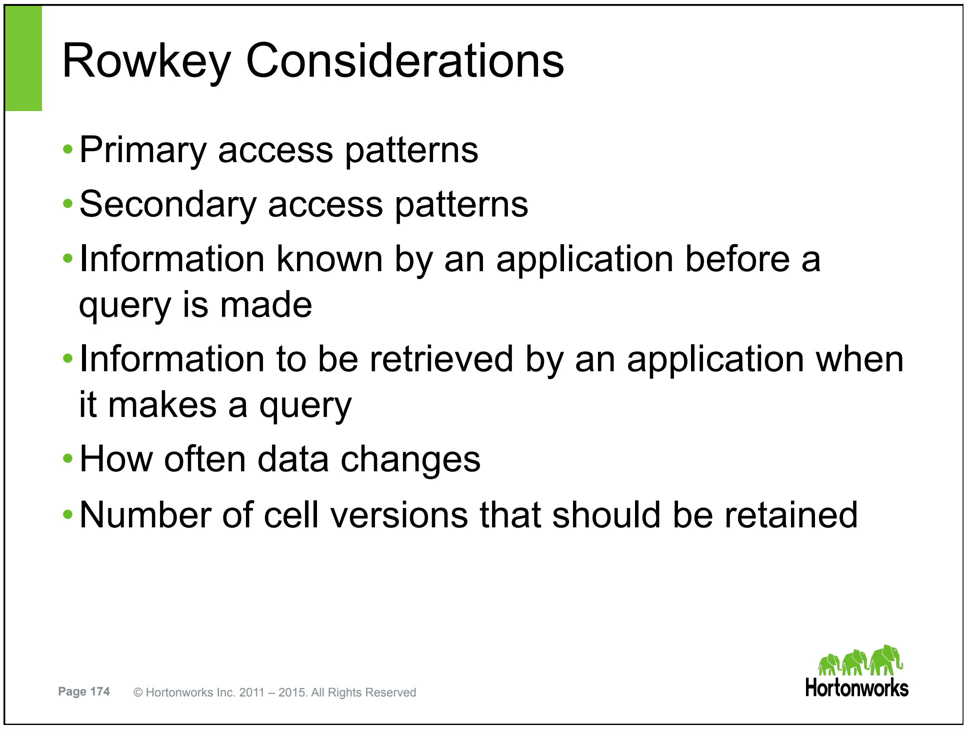
Hortonworks



Design Patterns

- Rowkey design is the single most important decision
- Design for the questions, not the answers
- There are only two sizes of data when scanning to answer an interactive request:
  - Too big
  - Not too big
- Be compact
- Use row atomicity as a design tool
- Move attributes into the rowkey

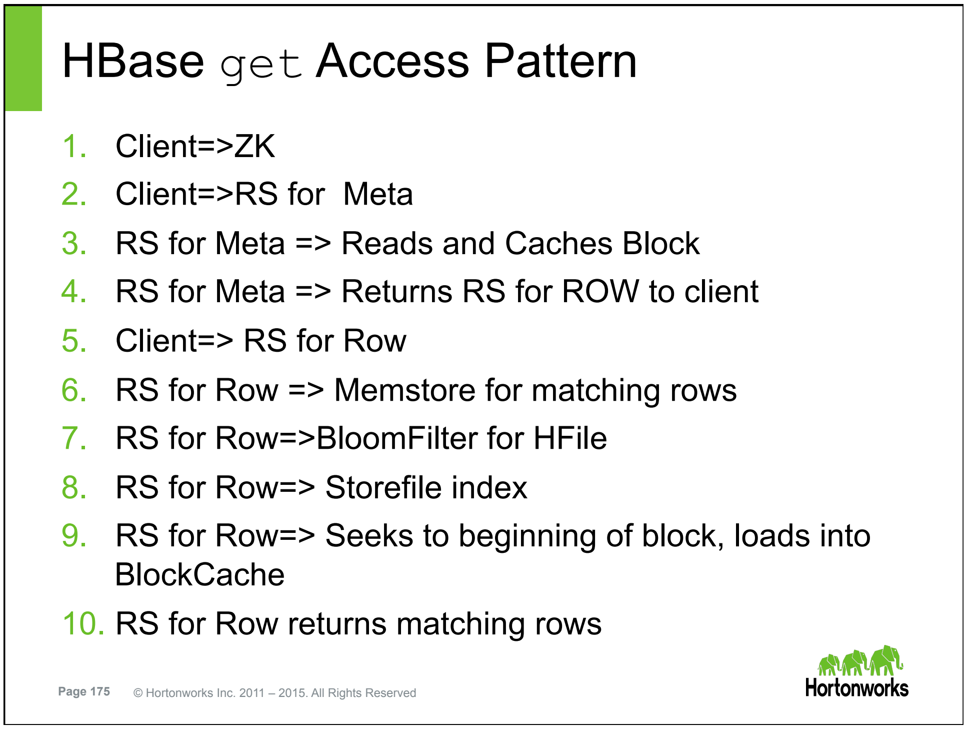
Page 173



Rowkey Considerations

- Primary access patterns
- Secondary access patterns
- Information known by an application before a query is made
- Information to be retrieved by an application when it makes a query
- How often data changes
- Number of cell versions that should be retained

Page 174

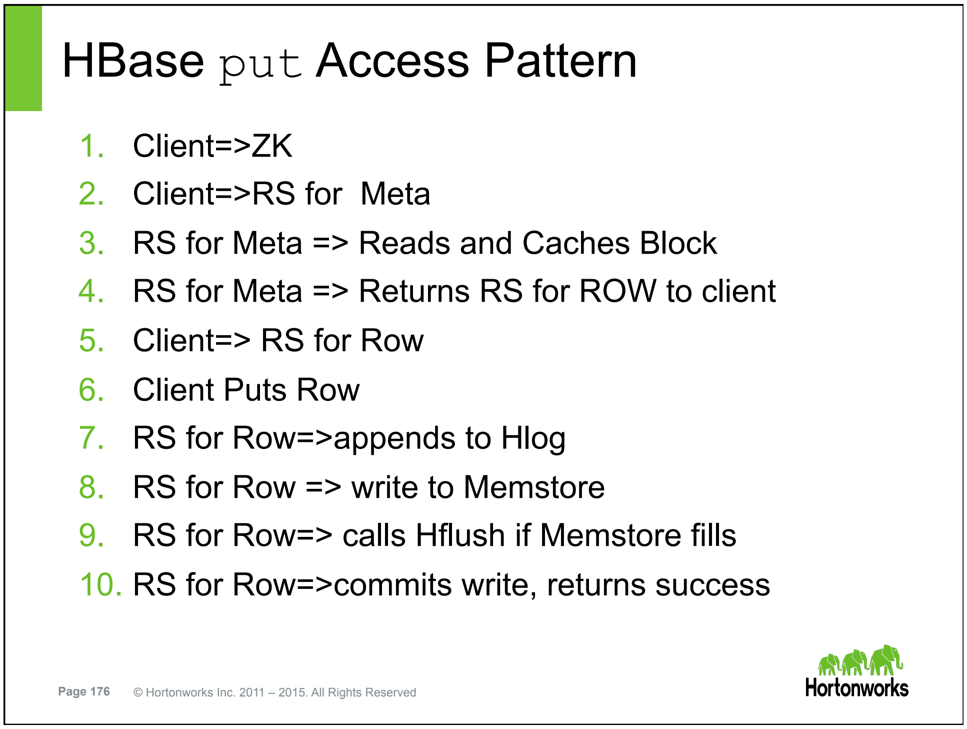


HBase `get` Access Pattern

1. Client=>ZK
2. Client=>RS for Meta
3. RS for Meta => Reads and Caches Block
4. RS for Meta => Returns RS for ROW to client
5. Client=> RS for Row
6. RS for Row => Memstore for matching rows
7. RS for Row=>BloomFilter for HFile
8. RS for Row=> Storefile index
9. RS for Row=> Seeks to beginning of block, loads into BlockCache
10. RS for Row returns matching rows

Page 175 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

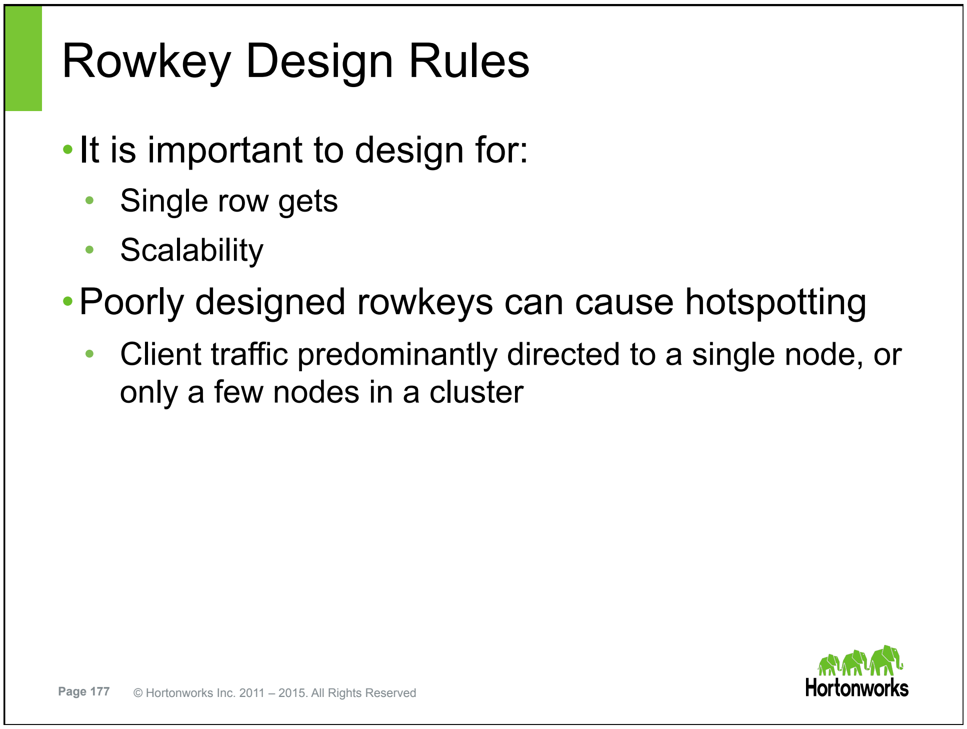


HBase `put` Access Pattern

1. Client=>ZK
2. Client=>RS for Meta
3. RS for Meta => Reads and Caches Block
4. RS for Meta => Returns RS for ROW to client
5. Client=> RS for Row
6. Client Puts Row
7. RS for Row=>appends to Hlog
8. RS for Row => write to Memstore
9. RS for Row=> calls Hflush if Memstore fills
10. RS for Row=>commits write, returns success

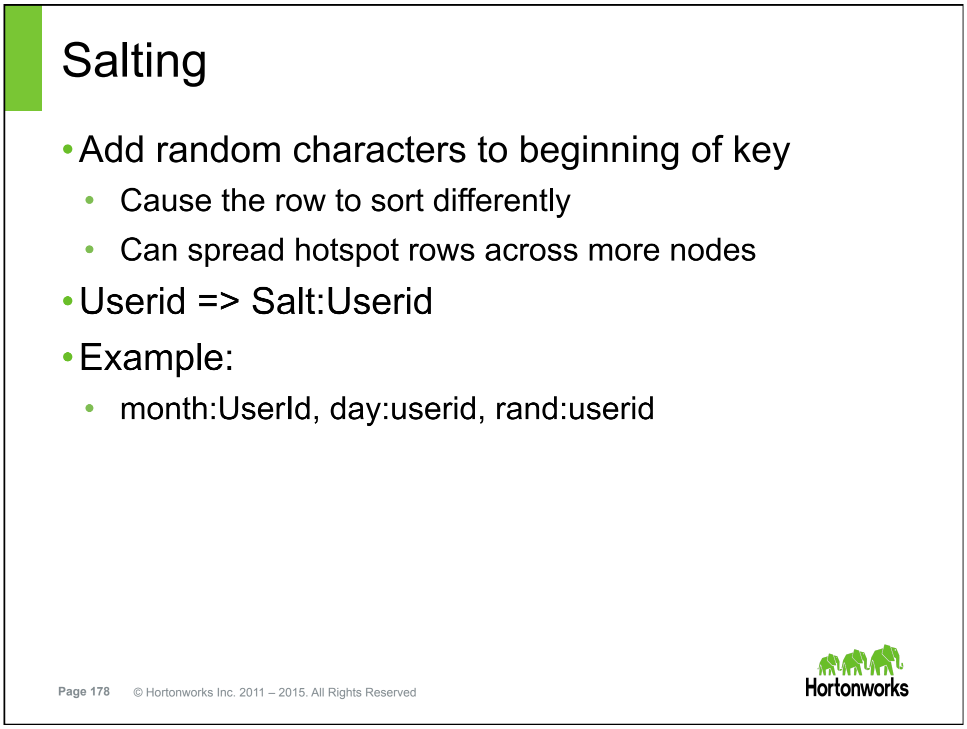
Page 176 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



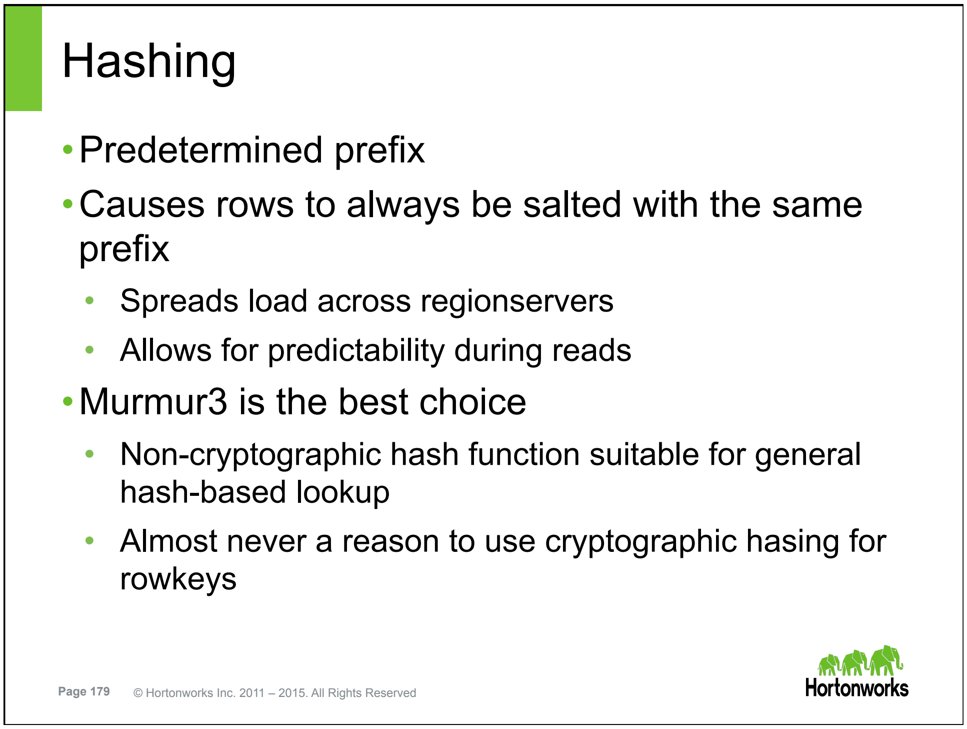
Rowkey Design Rules

- It is important to design for:
  - Single row gets
  - Scalability
- Poorly designed rowkeys can cause hotspotting
  - Client traffic predominantly directed to a single node, or only a few nodes in a cluster



Salting

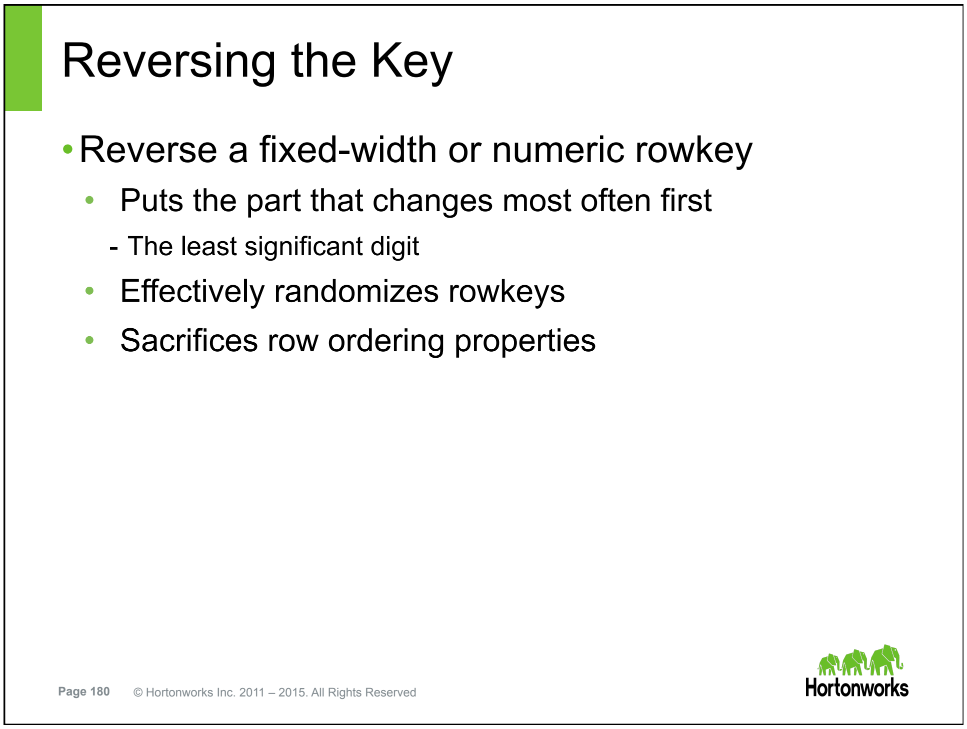
- Add random characters to beginning of key
  - Cause the row to sort differently
  - Can spread hotspot rows across more nodes
- Userid => Salt:Userid
- Example:
  - month:UserId, day:userid, rand:userid



Hashing

- Predetermined prefix
- Causes rows to always be salted with the same prefix
  - Spreads load across regionservers
  - Allows for predictability during reads
- Murmur3 is the best choice
  - Non-cryptographic hash function suitable for general hash-based lookup
  - Almost never a reason to use cryptographic hasing for rowkeys

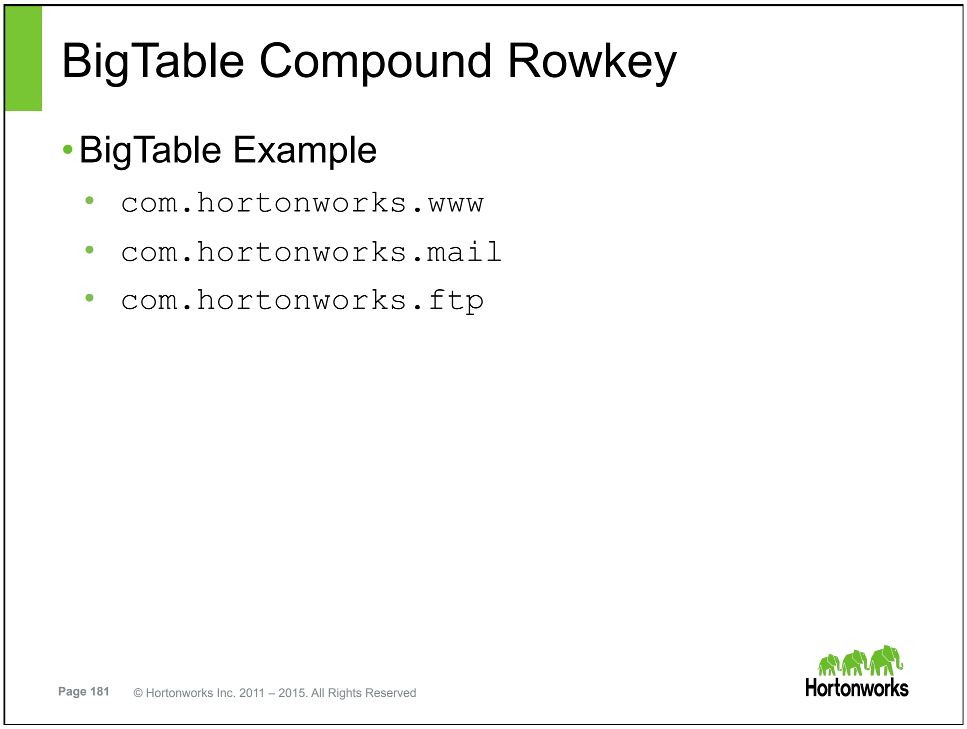
Page 179 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Reversing the Key

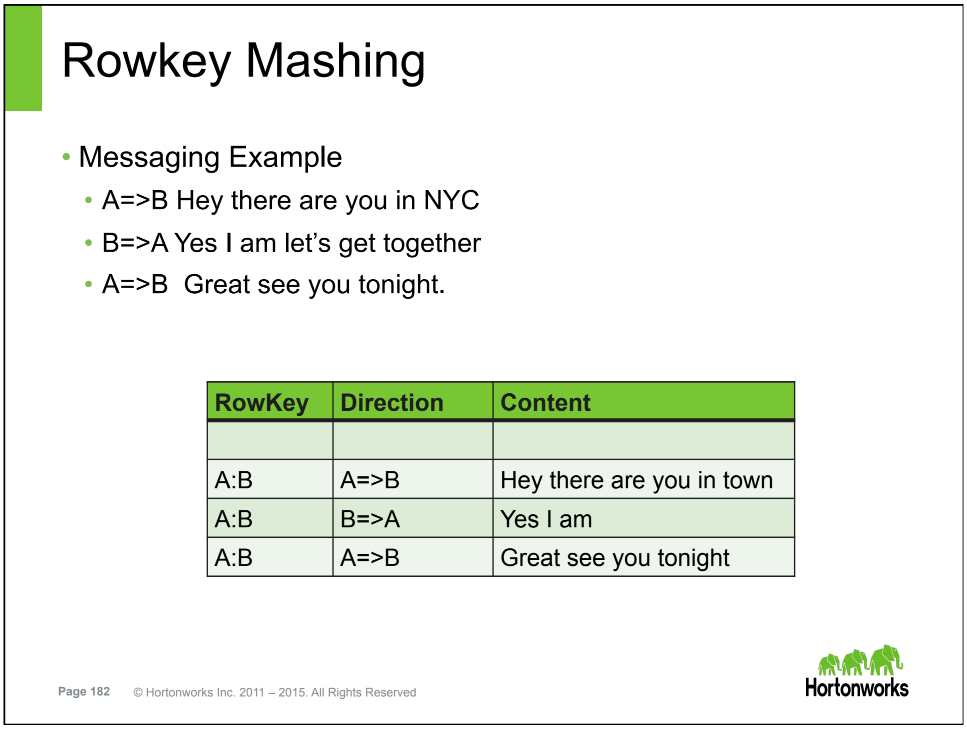
- Reverse a fixed-width or numeric rowkey
  - Puts the part that changes most often first
    - The least significant digit
  - Effectively randomizes rowkeys
  - Sacrifices row ordering properties

Page 180 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



BigTable Compound Rowkey

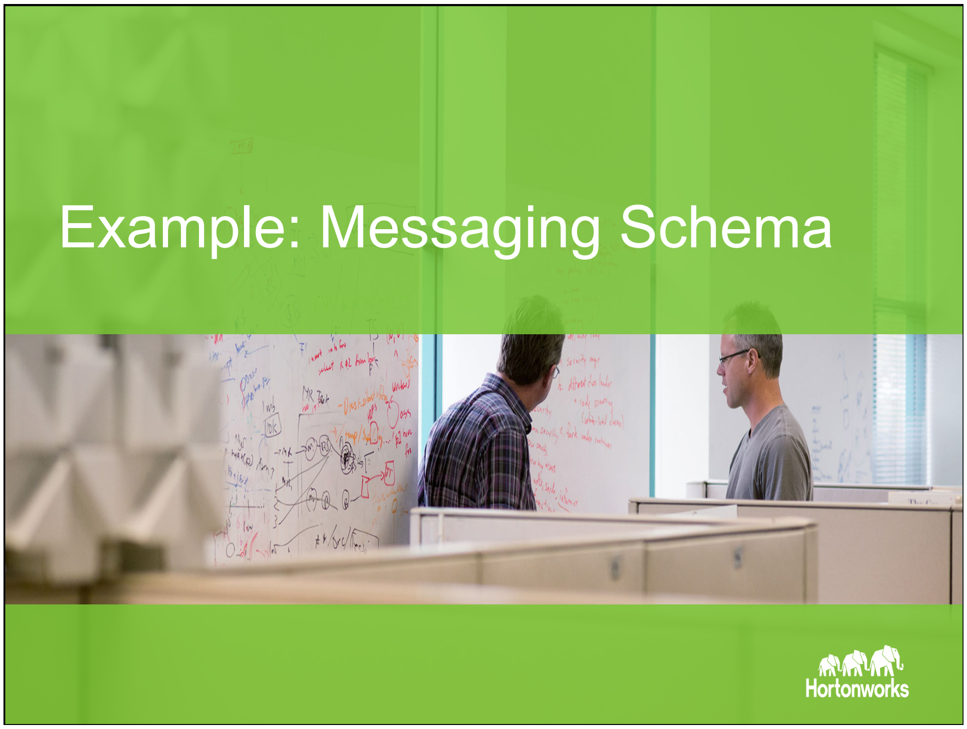
- BigTable Example
  - `com.hortonworks.www`
  - `com.hortonworks.mail`
  - `com.hortonworks.ftp`



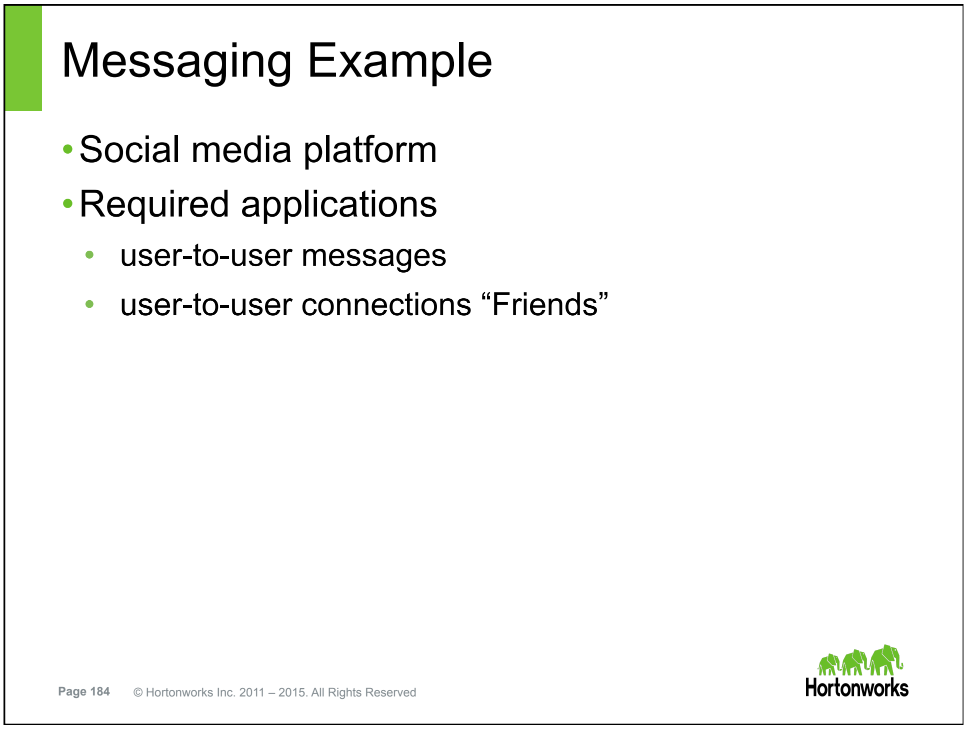
Rowkey Mashing

- Messaging Example
  - A=>B Hey there are you in NYC
  - B=>A Yes I am let’s get together
  - A=>B Great see you tonight.

RowKey	Direction	Content
A:B	A=>B	Hey there are you in town
A:B	B=>A	Yes I am
A:B	A=>B	Great see you tonight



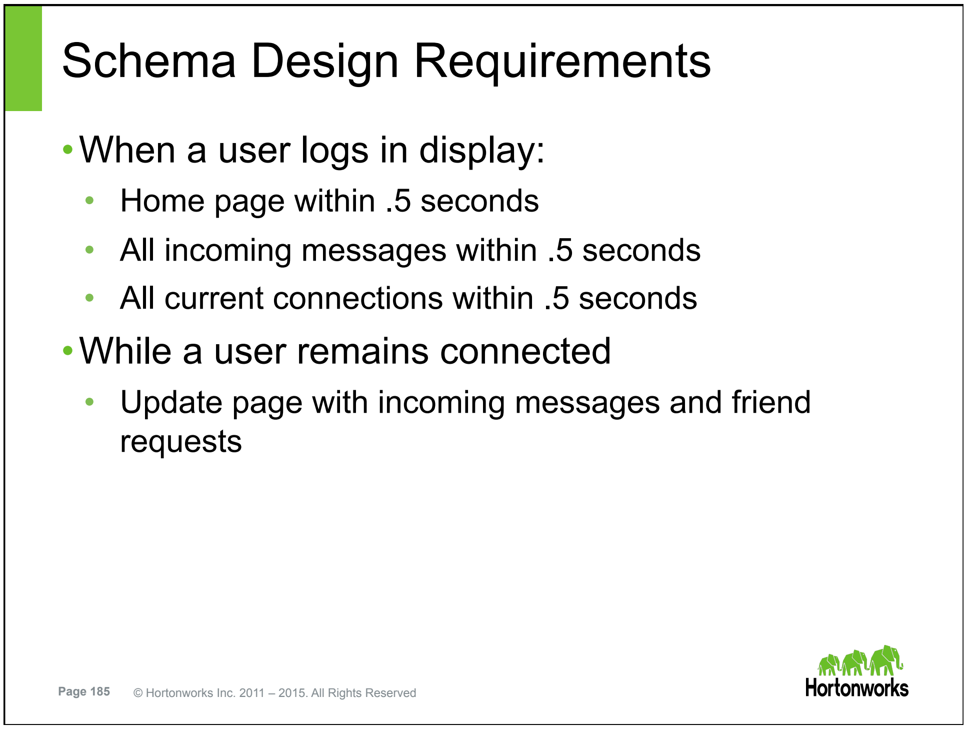
Example: Messaging Schema



Messaging Example

- Social media platform
- Required applications
  - user-to-user messages
  - user-to-user connections “Friends”

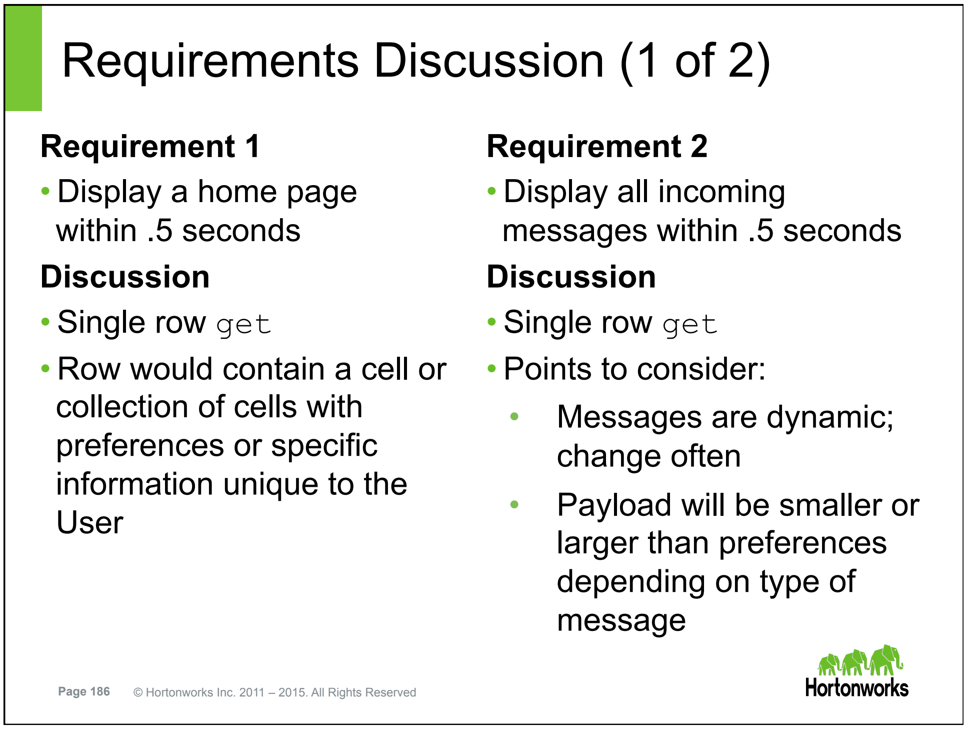
Page 184



Schema Design Requirements

- When a user logs in display:
  - Home page within .5 seconds
  - All incoming messages within .5 seconds
  - All current connections within .5 seconds
- While a user remains connected
  - Update page with incoming messages and friend requests

Page 185



Requirements Discussion (1 of 2)

Requirement 1

- Display a home page within .5 seconds

Discussion

- Single row `get`
- Row would contain a cell or collection of cells with preferences or specific information unique to the User

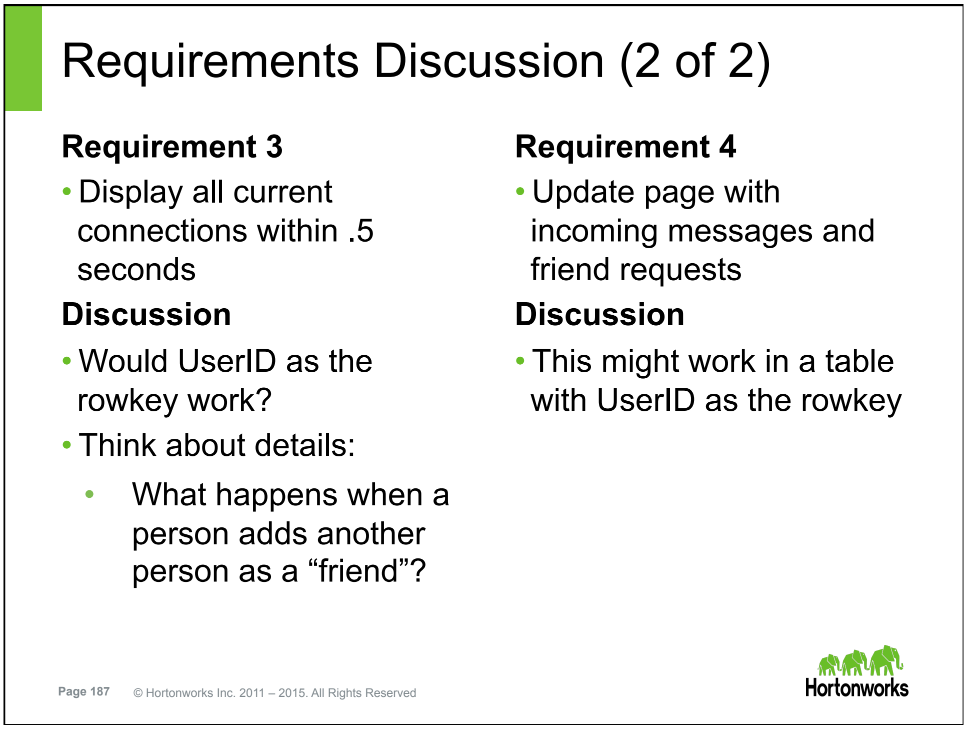
Requirement 2

- Display all incoming messages within .5 seconds

Discussion

- Single row `get`
- Points to consider:
  - Messages are dynamic; change often
  - Payload will be smaller or larger than preferences depending on type of message

Page 186



Requirements Discussion (2 of 2)

Requirement 3

- Display all current connections within .5 seconds

Discussion

- Would UserID as the rowkey work?
- Think about details:
  - What happens when a person adds another person as a “friend”?

Requirement 4

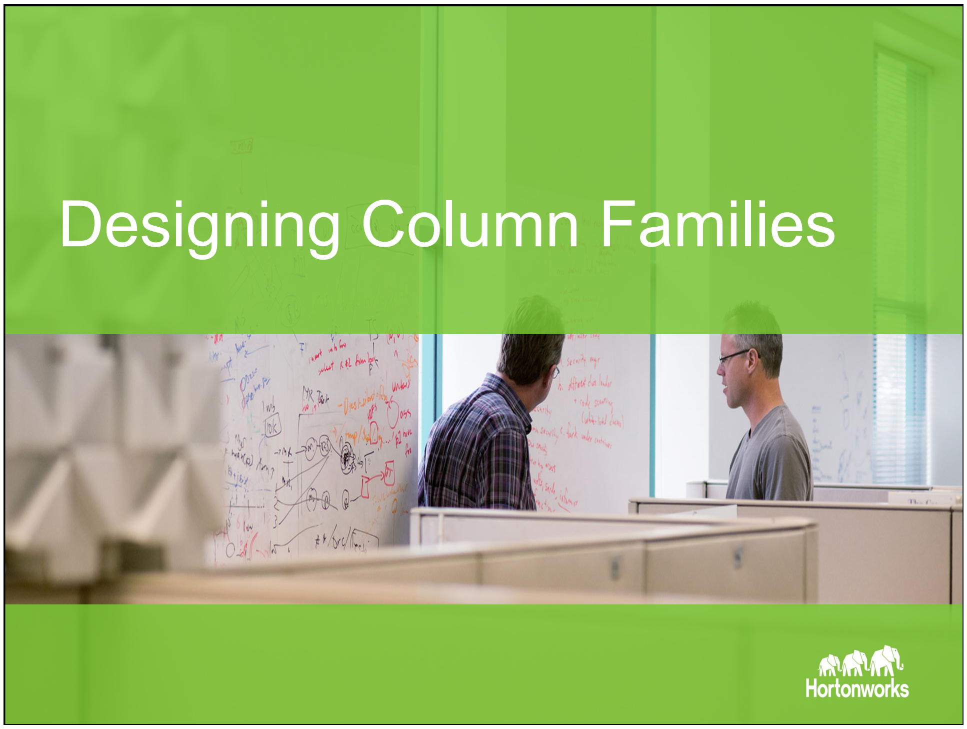
- Update page with incoming messages and friend requests

Discussion

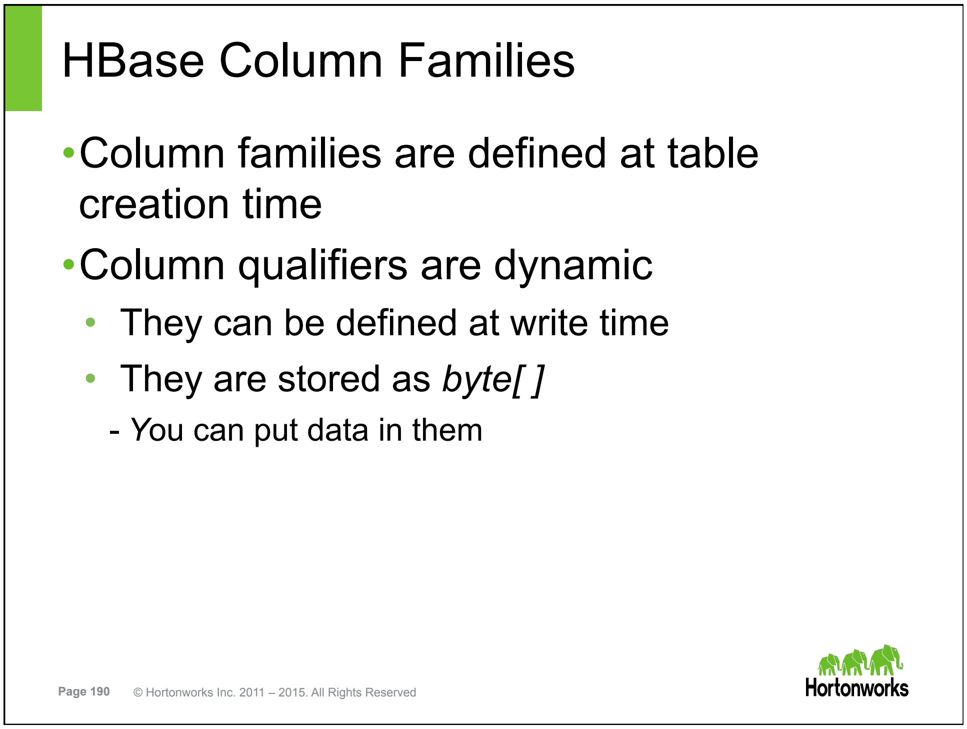
- This might work in a table with UserID as the rowkey

Page 187 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Discussion: Rowkey Design Implications



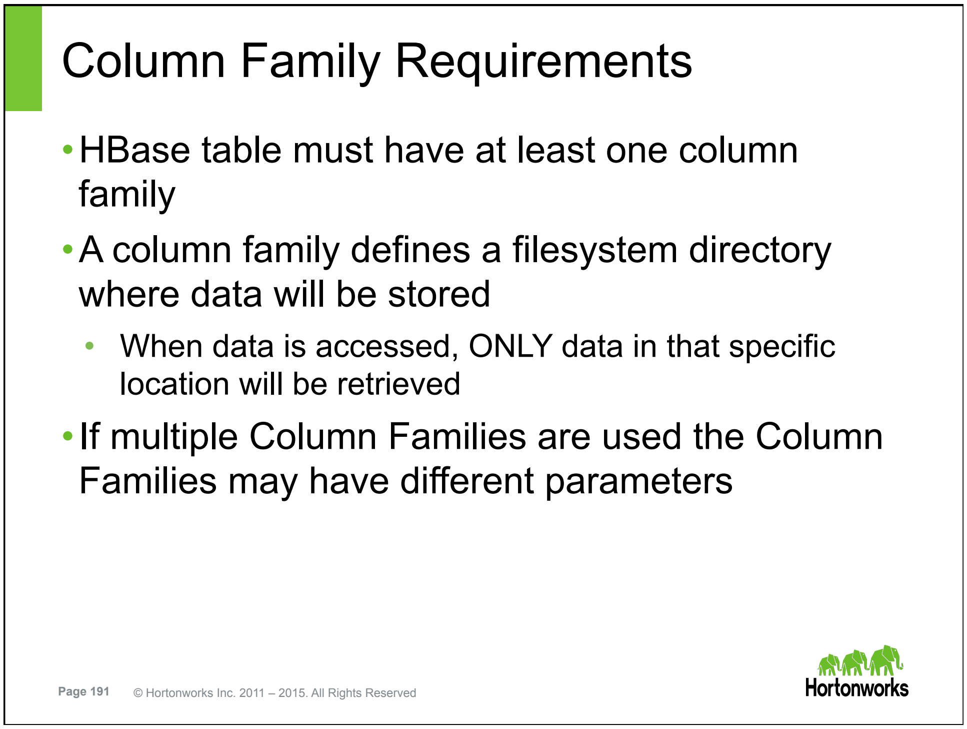
Designing Column Families



HBase Column Families

- Column families are defined at table creation time
- Column qualifiers are dynamic
  - They can be defined at write time
  - They are stored as *byte[ ]*
    - *You* can put data in them

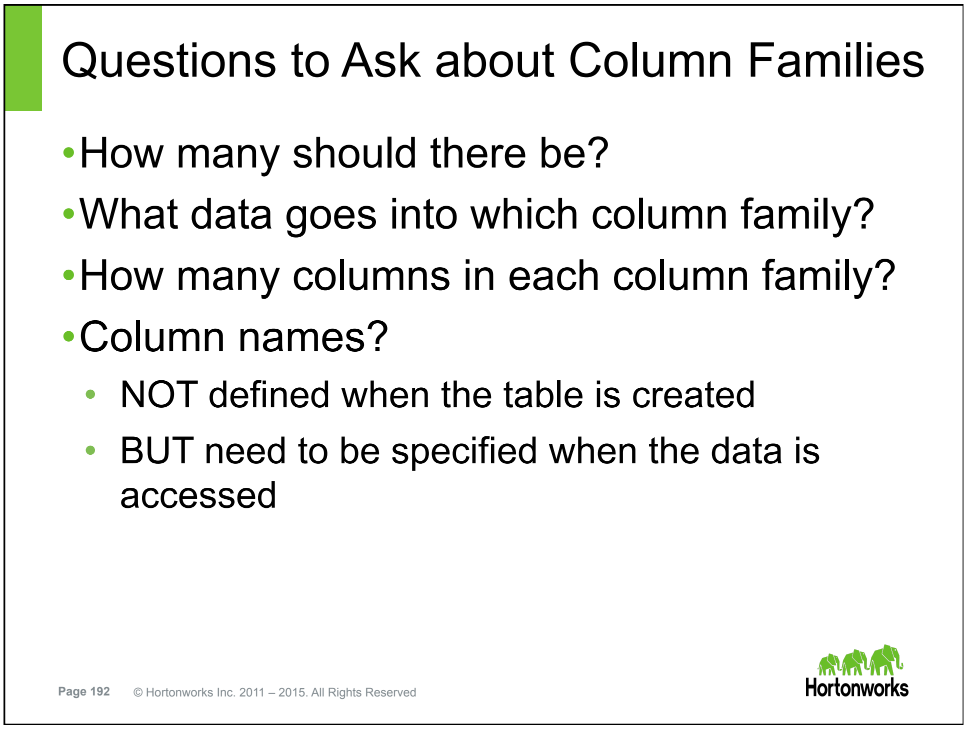
Page 190



Column Family Requirements

- HBase table must have at least one column family
- A column family defines a filesystem directory where data will be stored
  - When data is accessed, ONLY data in that specific location will be retrieved
- If multiple Column Families are used the Column Families may have different parameters

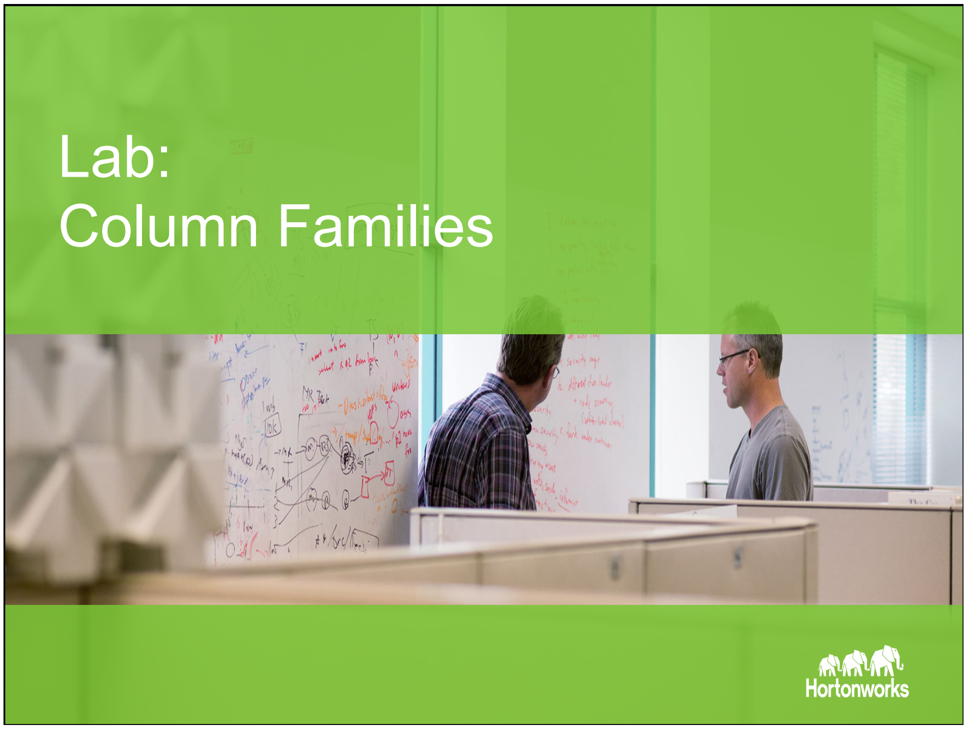
Page 191 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Questions to Ask about Column Families

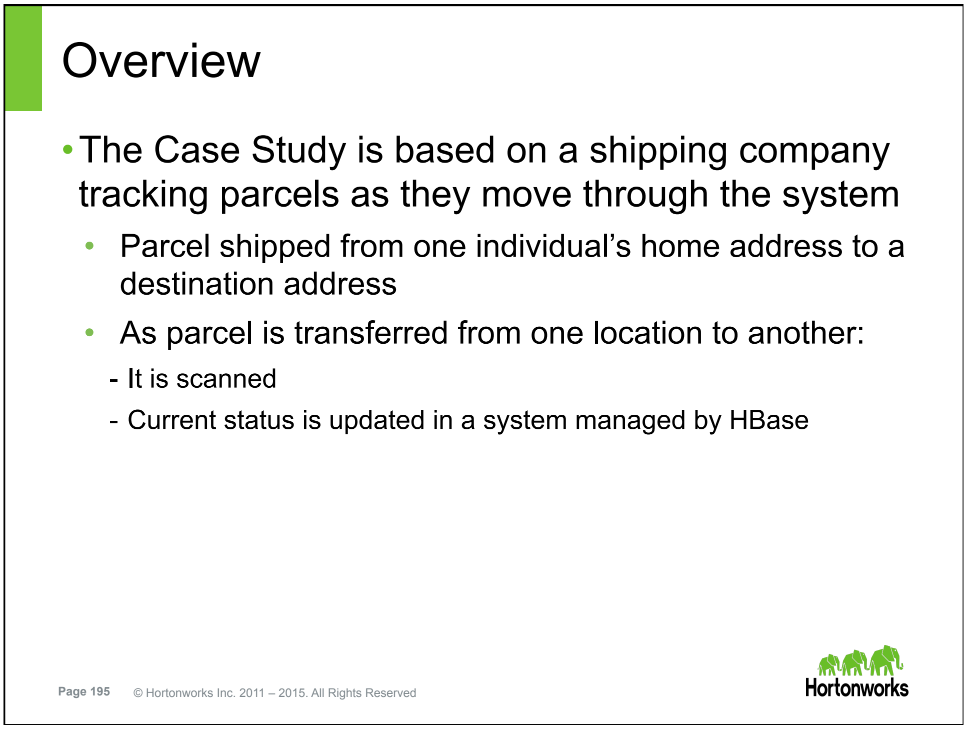
- How many should there be?
- What data goes into which column family?
- How many columns in each column family?
- Column names?
  - NOT defined when the table is created
  - BUT need to be specified when the data is accessed

Page 192 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Lab:
Column Families

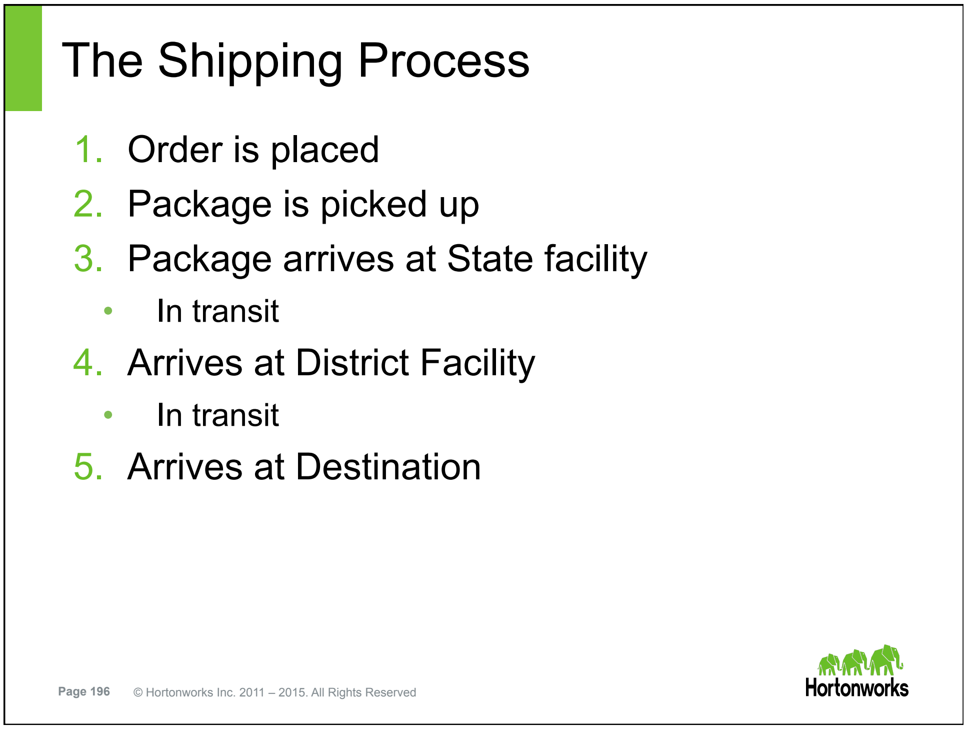
Case Study: Trucking Company



Overview

- The Case Study is based on a shipping company tracking parcels as they move through the system
  - Parcel shipped from one individual's home address to a destination address
  - As parcel is transferred from one location to another:
    - It is scanned
    - Current status is updated in a system managed by HBase

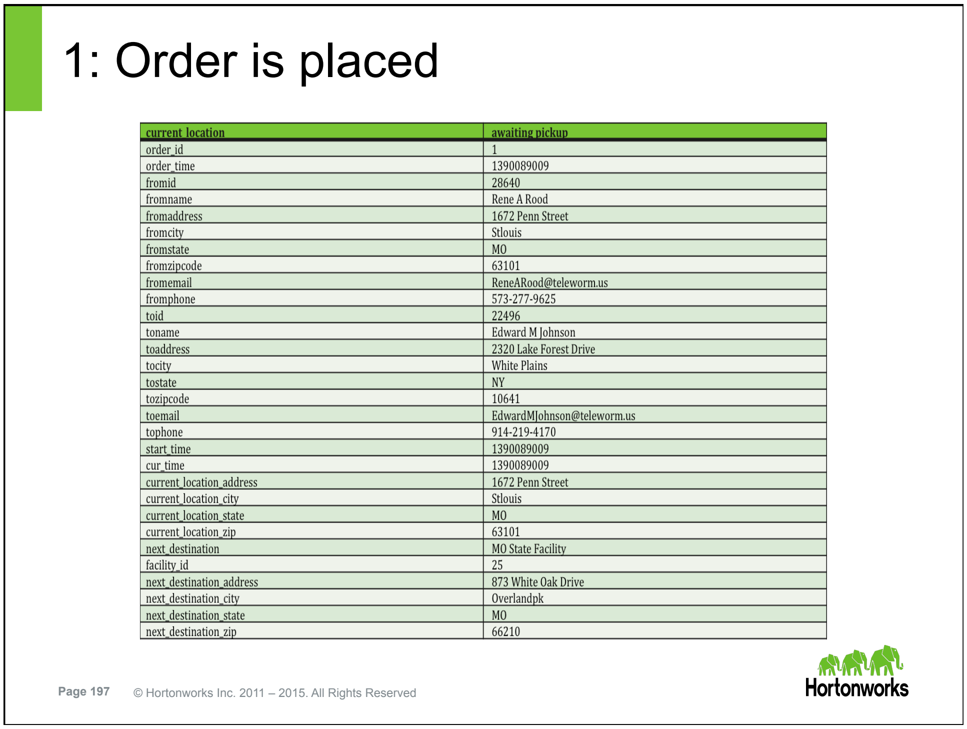
Page 195



The Shipping Process

1. Order is placed
2. Package is picked up
3. Package arrives at State facility
   - In transit
4. Arrives at District Facility
   - In transit
5. Arrives at Destination

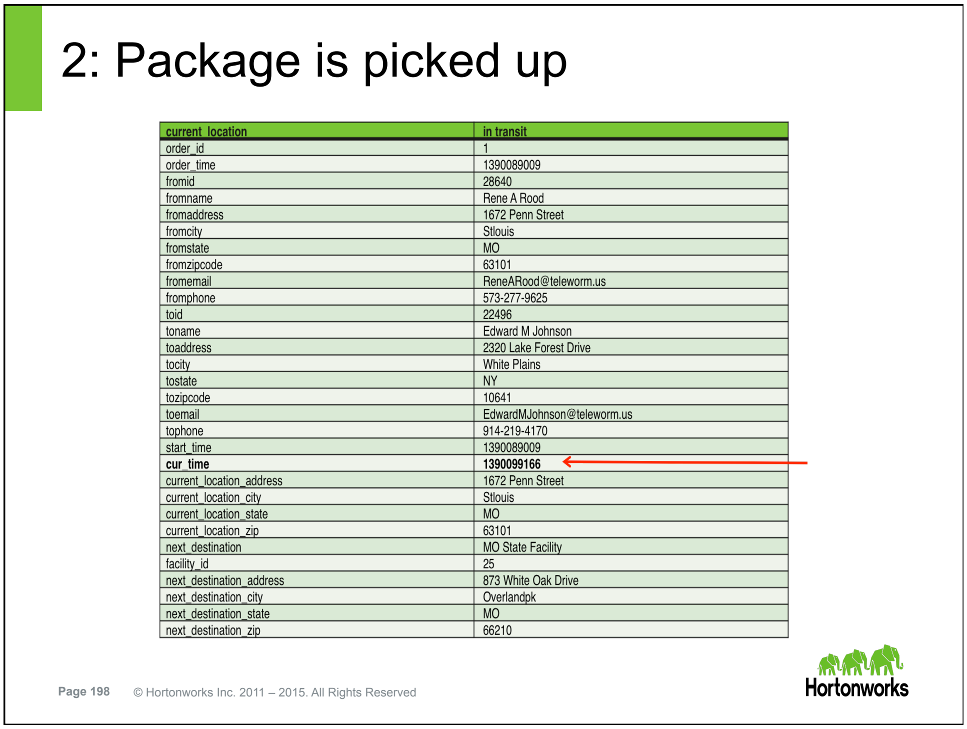
Page 196



1: Order is placed

current_location	awaiting pickup
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	1390089009
current_location_address	1672 Penn Street
current_location_city	Stlouis
current_location_state	MO
current_location_zip	63101
next_destination	MO State Facility
facility_id	25
next_destination_address	873 White Oak Drive
next_destination_city	Overlandpk
next_destination_state	MO
next_destination_zip	66210

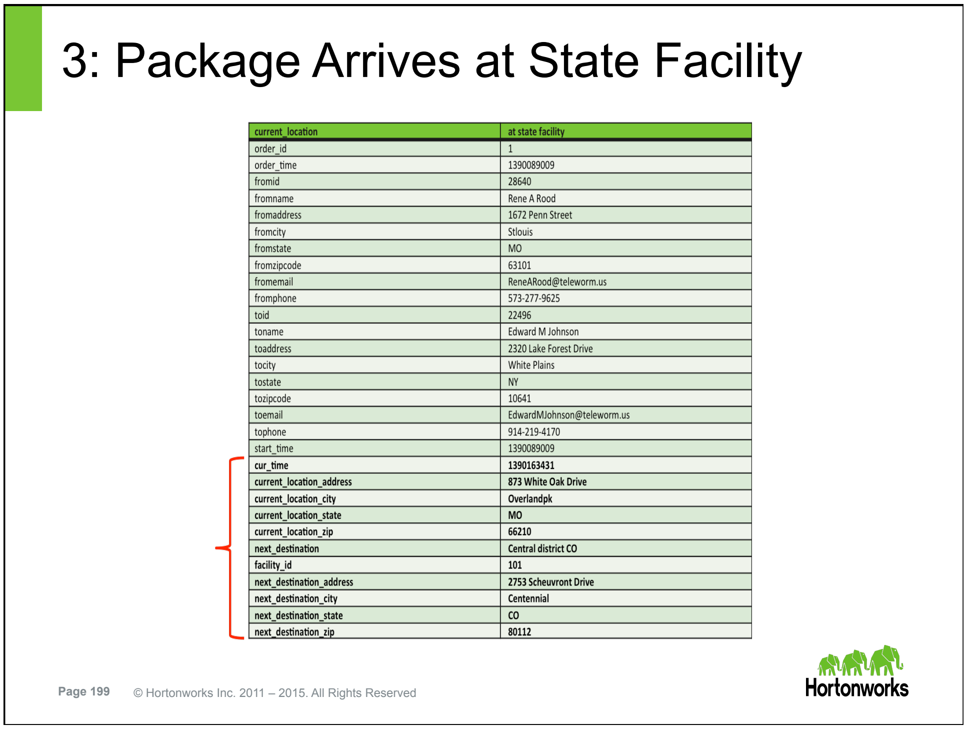
Page 197 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



2: Package is picked up

current_location	in transit
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	**1390099166**
current_location_address	1672 Penn Street
current_location_city	Stlouis
current_location_state	MO
current_location_zip	63101
next_destination	MO State Facility
facility_id	25
next_destination_address	873 White Oak Drive
next_destination_city	Overlandpk
next_destination_state	MO
next_destination_zip	66210

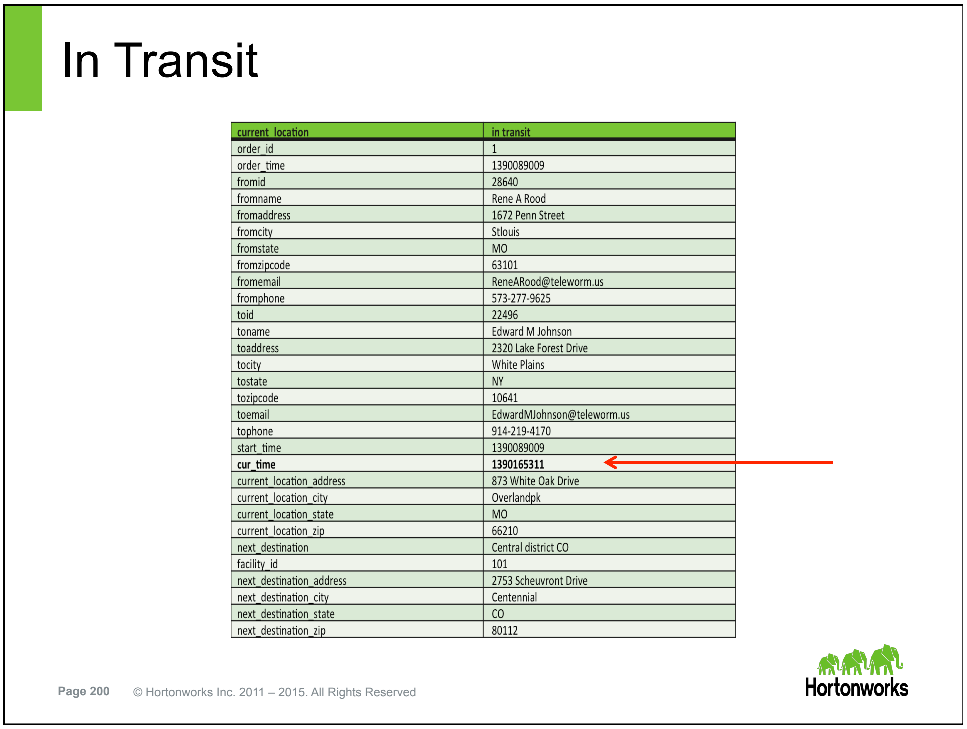
Page 198 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



3: Package Arrives at State Facility

current_location	at state facility
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	**1390163431**
current_location_address	**873 White Oak Drive**
current_location_city	**Overlandpk**
current_location_state	**MO**
current_location_zip	**66210**
next_destination	**Central district CO**
facility_id	**101**
next_destination_address	**2753 Scheuvront Drive**
next_destination_city	**Centennial**
next_destination_state	**CO**
next_destination_zip	**80112**

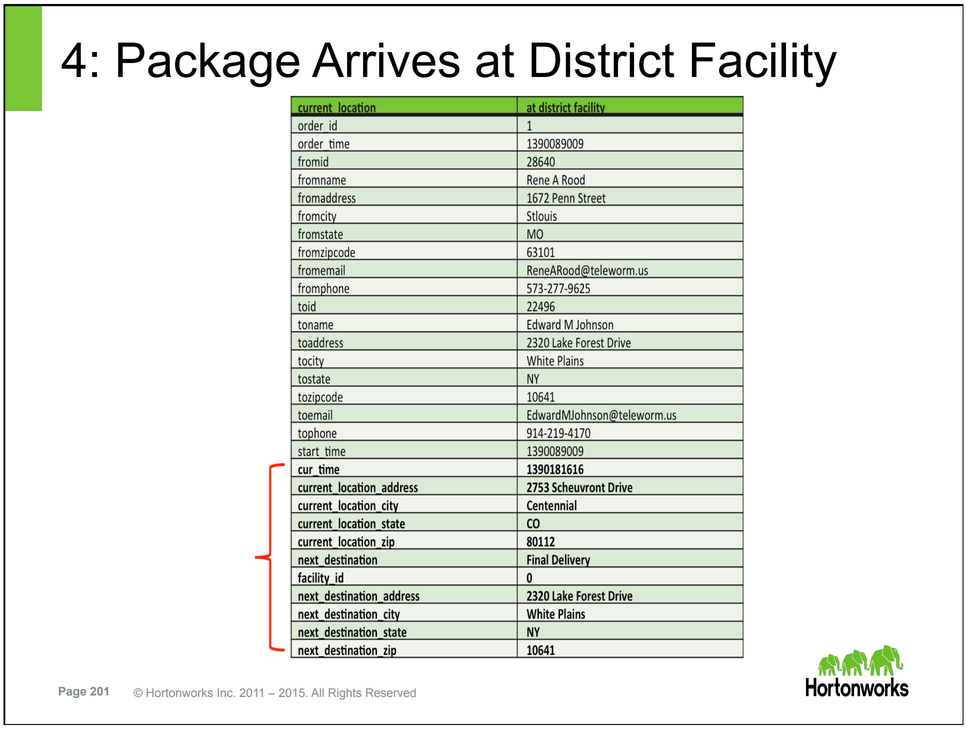
Page 199 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



In Transit

current_location	in transit
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	**1390165311**
current_location_address	873 White Oak Drive
current_location_city	Overlandpk
current_location_state	MO
current_location_zip	66210
next_destination	Central district CO
facility_id	101
next_destination_address	2753 Scheuvront Drive
next_destination_city	Centennial
next_destination_state	CO
next_destination_zip	80112

Page 200 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

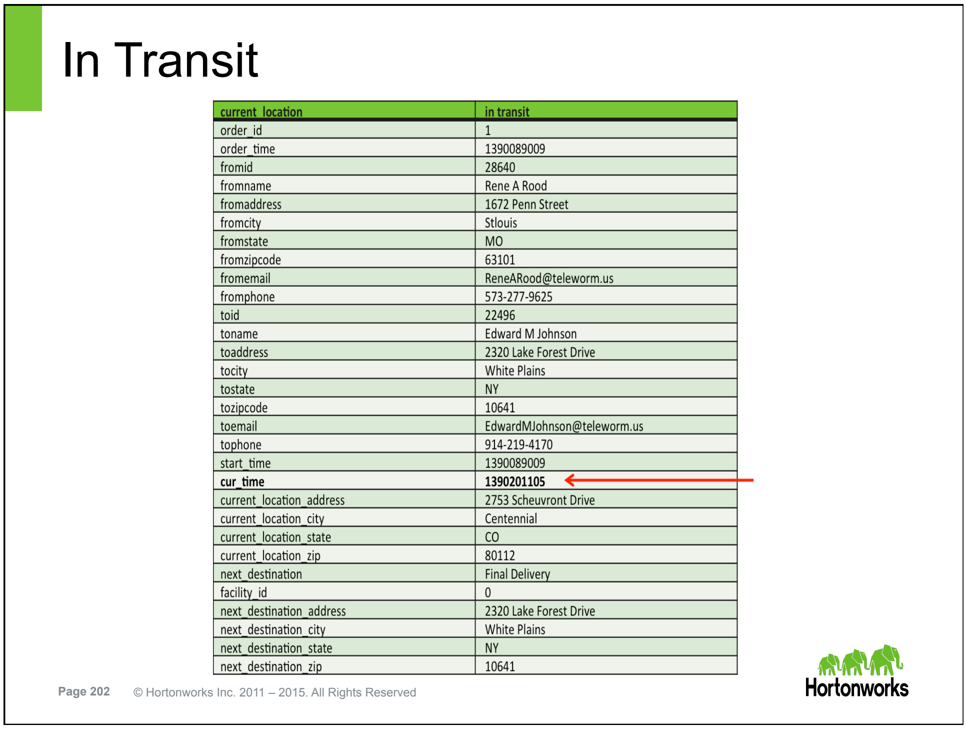


4: Package Arrives at District Facility

current_location	at district facility
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	**1390181616**
current_location_address	**2753 Scheuvront Drive**
current_location_city	**Centennial**
current_location_state	**CO**
current_location_zip	**80112**
next_destination	**Final Delivery**
facility_id	**0**
next_destination_address	**2320 Lake Forest Drive**
next_destination_city	**White Plains**
next_destination_state	**NY**
next_destination_zip	**10641**

Page 201 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

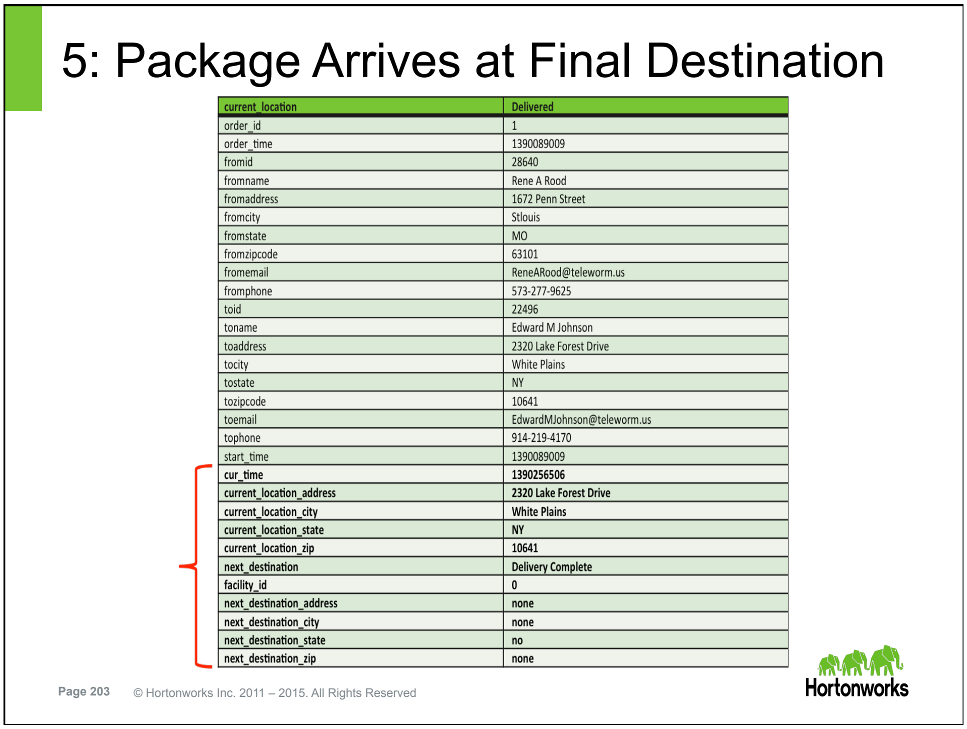


In Transit

current_location	in transit
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	**1390201105**
current_location_address	2753 Scheuvront Drive
current_location_city	Centennial
current_location_state	CO
current_location_zip	80112
next_destination	Final Delivery
facility_id	0
next_destination_address	2320 Lake Forest Drive
next_destination_city	White Plains
next_destination_state	NY
next_destination_zip	10641

Page 202 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

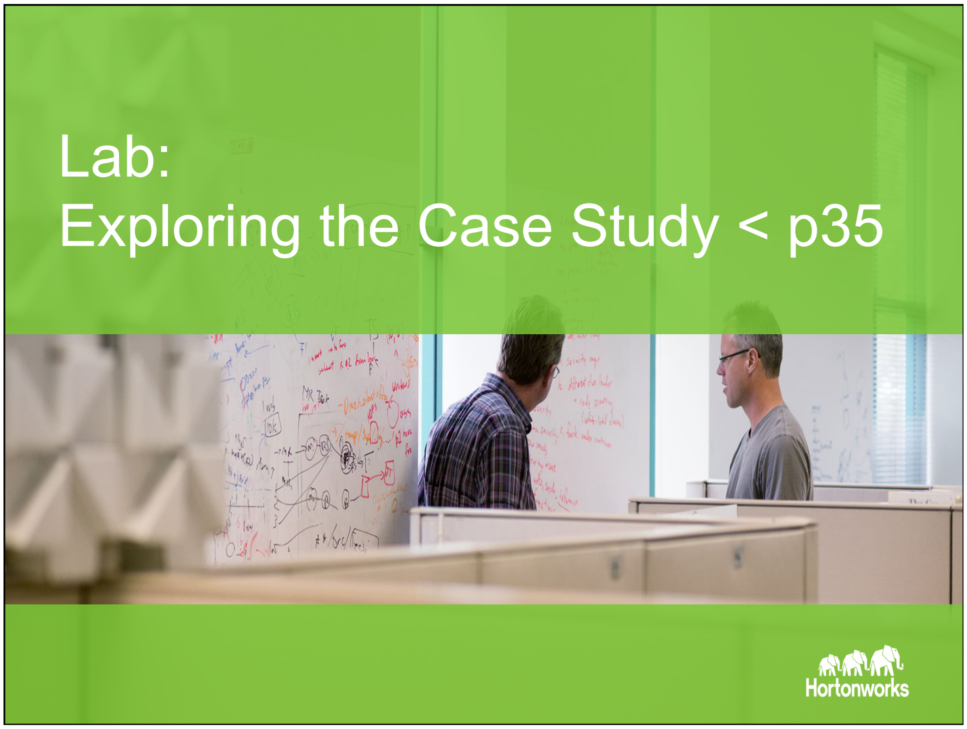


5: Package Arrives at Final Destination

current_location	Delivered
order_id	1
order_time	1390089009
fromid	28640
fromname	Rene A Rood
fromaddress	1672 Penn Street
fromcity	Stlouis
fromstate	MO
fromzipcode	63101
fromemail	ReneARood@teleworm.us
fromphone	573-277-9625
toid	22496
toname	Edward M Johnson
toaddress	2320 Lake Forest Drive
tocity	White Plains
tostate	NY
tozipcode	10641
toemail	EdwardMJohnson@teleworm.us
tophone	914-219-4170
start_time	1390089009
cur_time	**1390256506**
current_location_address	**2320 Lake Forest Drive**
current_location_city	**White Plains**
current_location_state	**NY**
current_location_zip	**10641**
next_destination	**Delivery Complete**
facility_id	**0**
next_destination_address	**none**
next_destination_city	**none**
next_destination_state	**no**
next_destination_zip	**none**

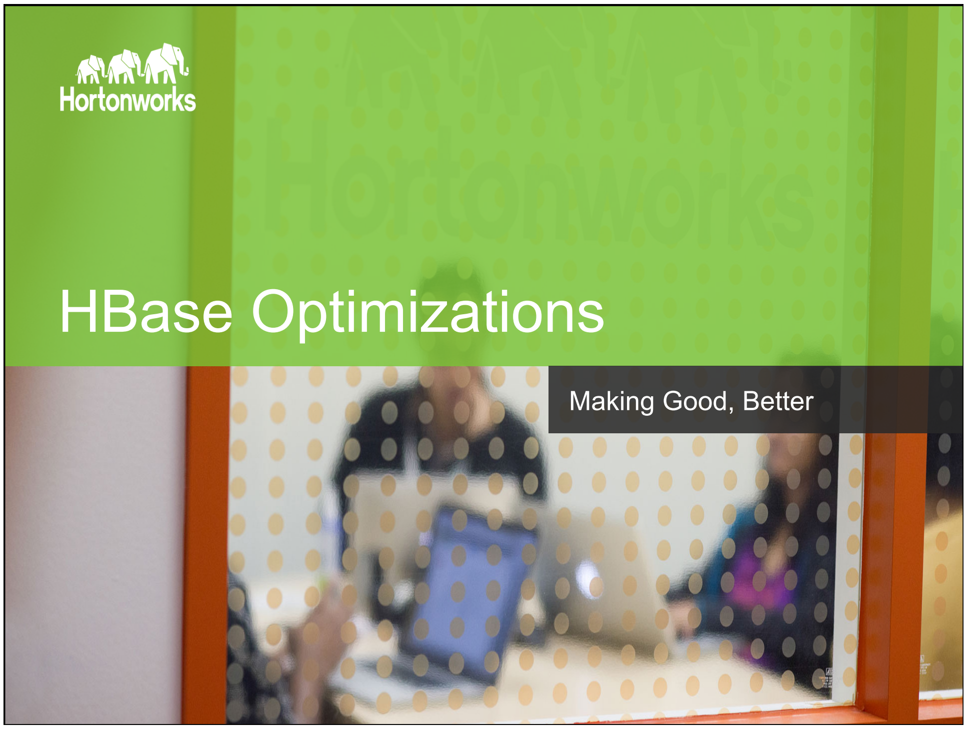
Page 203 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Lab:
Exploring the Case Study < p35

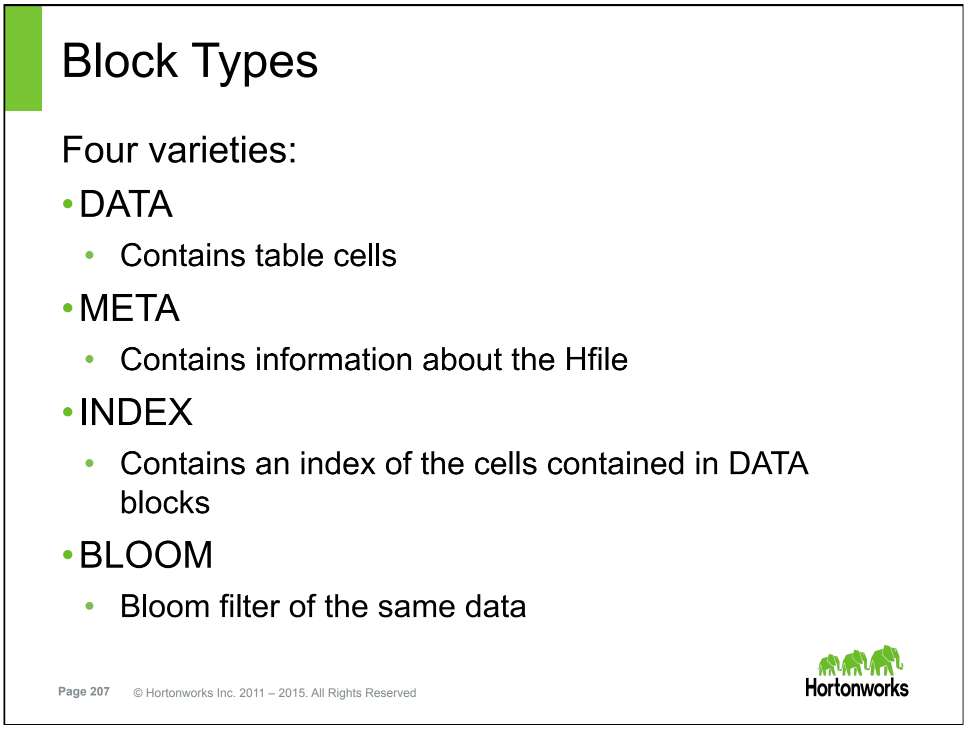
Hortonworks



HBase Optimizations

Making Good, Better

Setting Block Size

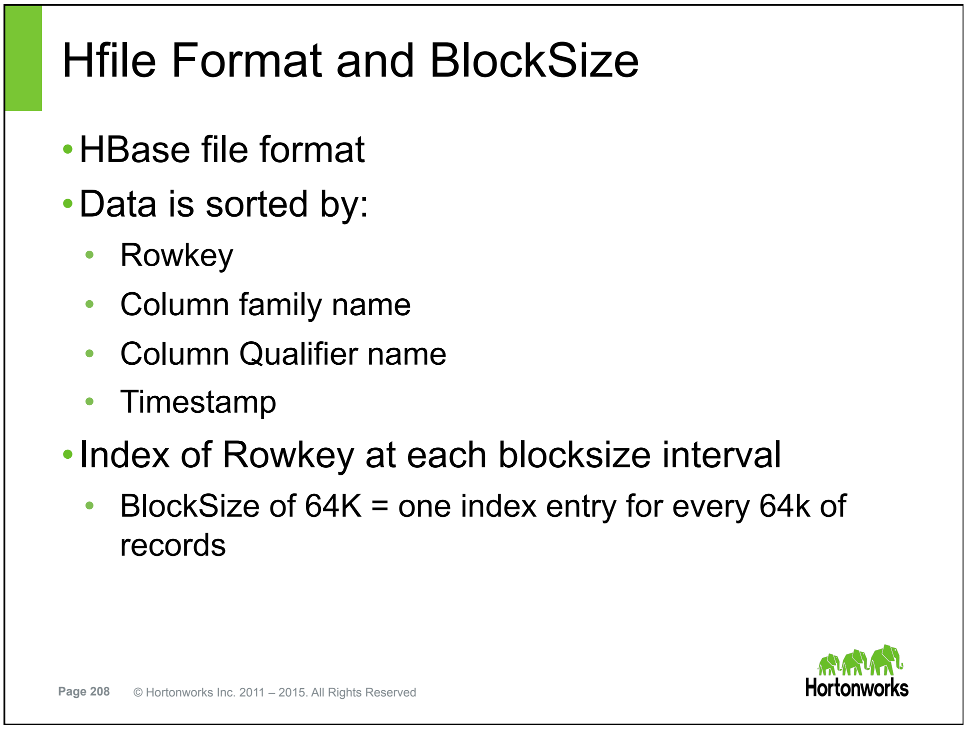


Block Types

Four varieties:

- DATA
  - Contains table cells
- META
  - Contains information about the Hfile
- INDEX
  - Contains an index of the cells contained in DATA blocks
- BLOOM
  - Bloom filter of the same data

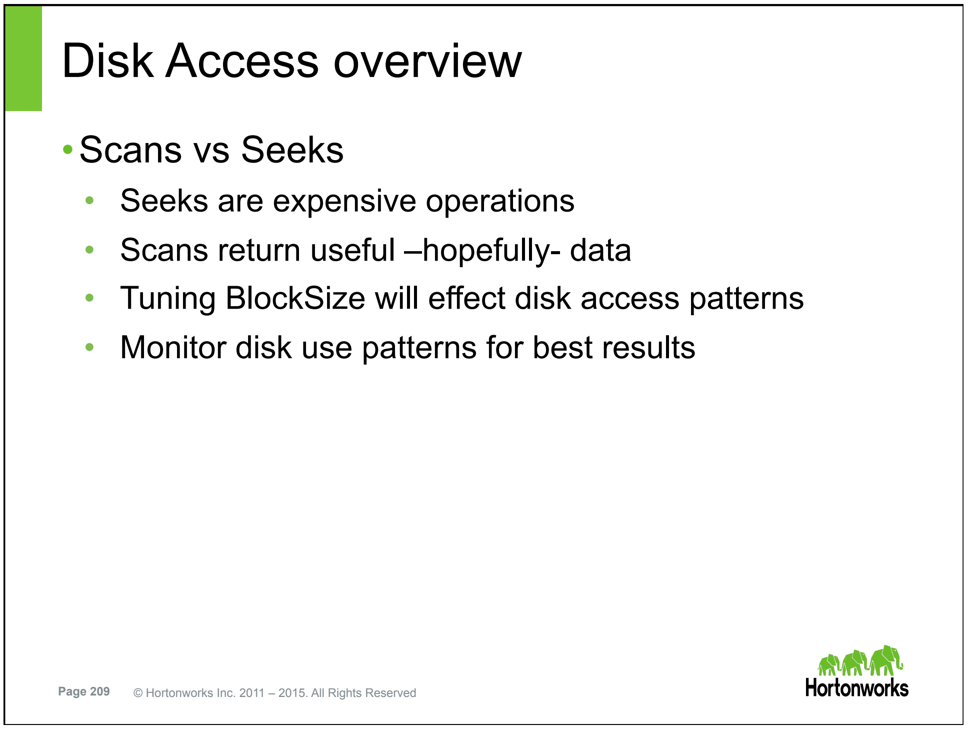
Page 207 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Hfile Format and BlockSize

- HBase file format
- Data is sorted by:
  - Rowkey
  - Column family name
  - Column Qualifier name
  - Timestamp
- Index of Rowkey at each blocksize interval
  - BlockSize of 64K = one index entry for every 64k of records

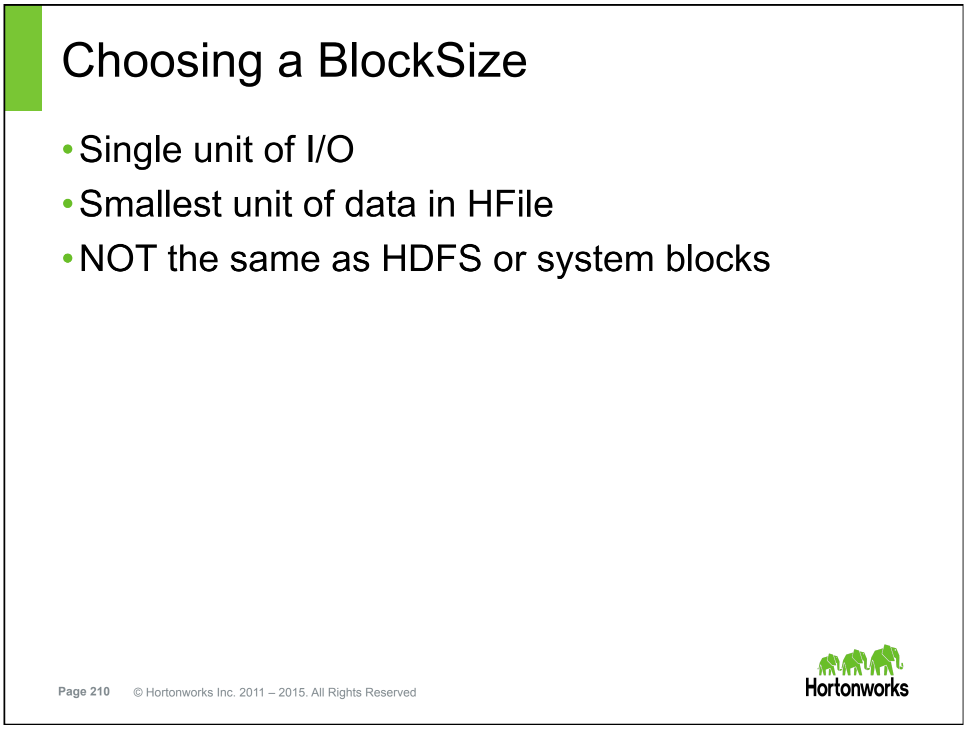
Page 208 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Disk Access overview

- Scans vs Seeks
  - Seeks are expensive operations
  - Scans return useful –hopefully- data
  - Tuning BlockSize will effect disk access patterns
  - Monitor disk use patterns for best results

Page 209

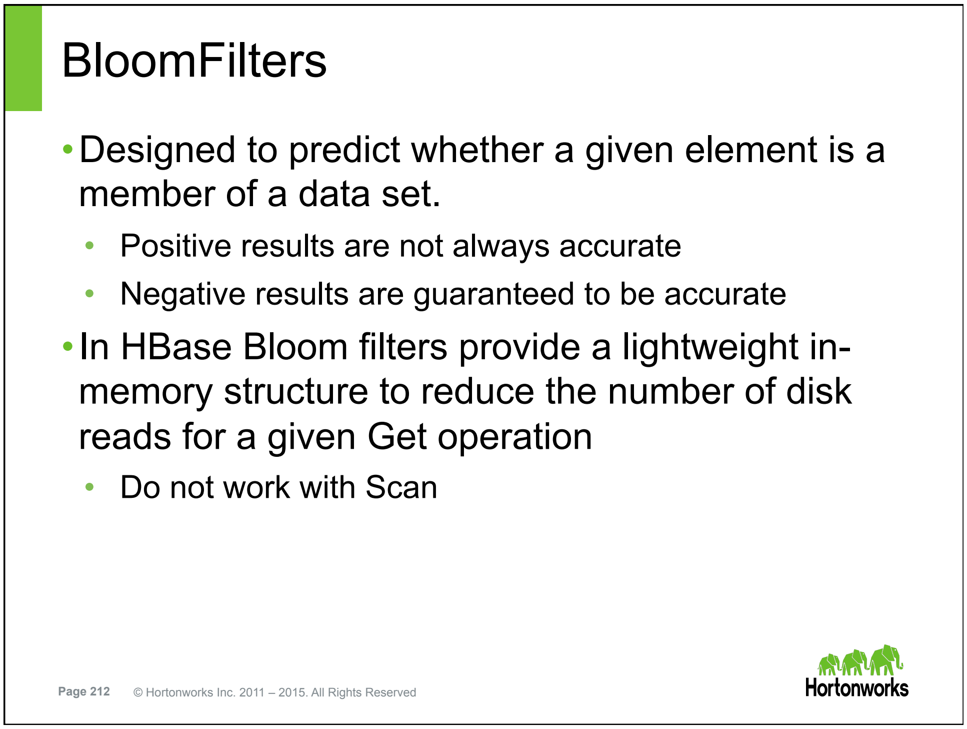


Choosing a BlockSize

- Single unit of I/O
- Smallest unit of data in HFile
- NOT the same as HDFS or system blocks

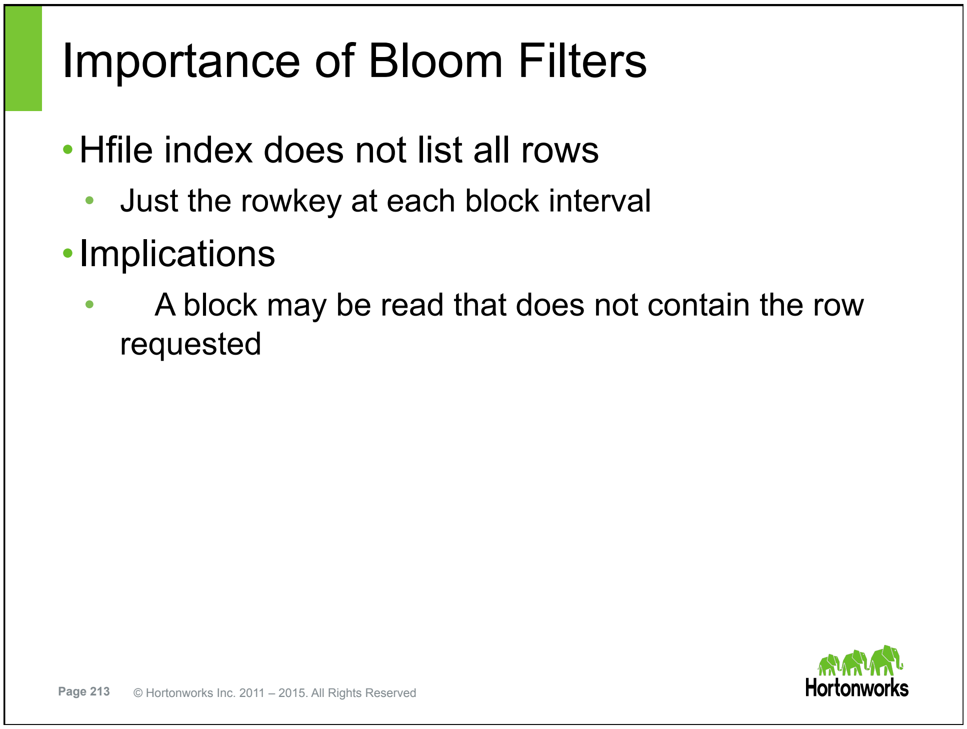
Page 210

Using Bloom Filters



BloomFilters

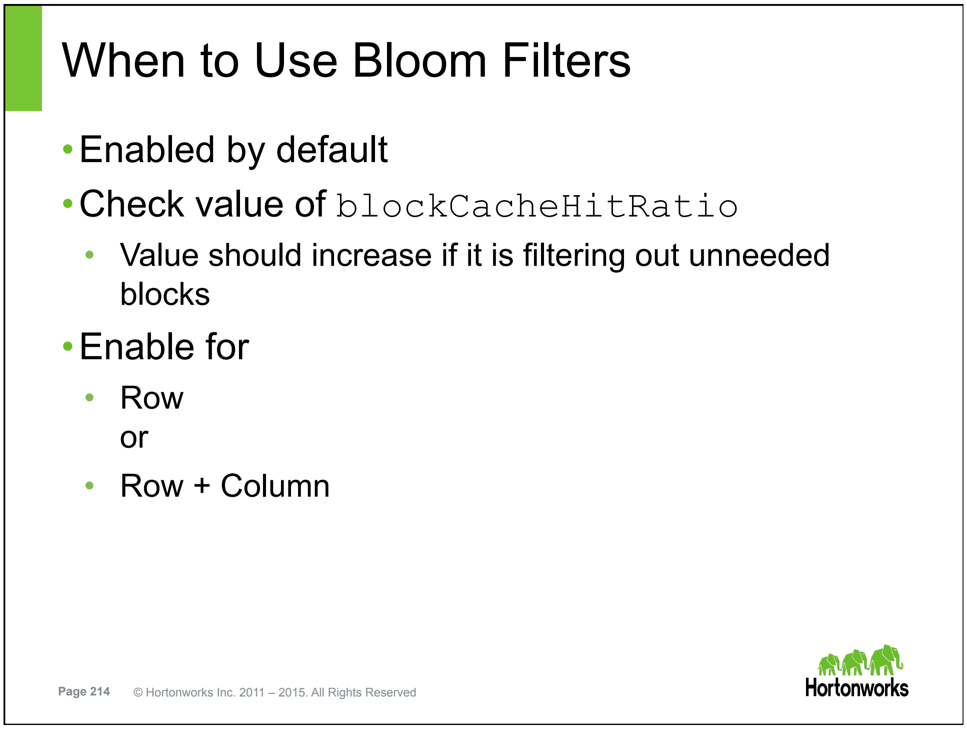
- Designed to predict whether a given element is a member of a data set.
  - Positive results are not always accurate
  - Negative results are guaranteed to be accurate
- In HBase Bloom filters provide a lightweight in-memory structure to reduce the number of disk reads for a given Get operation
  - Do not work with Scan



Importance of Bloom Filters

- Hfile index does not list all rows
  - Just the rowkey at each block interval
- Implications
  - A block may be read that does not contain the row requested

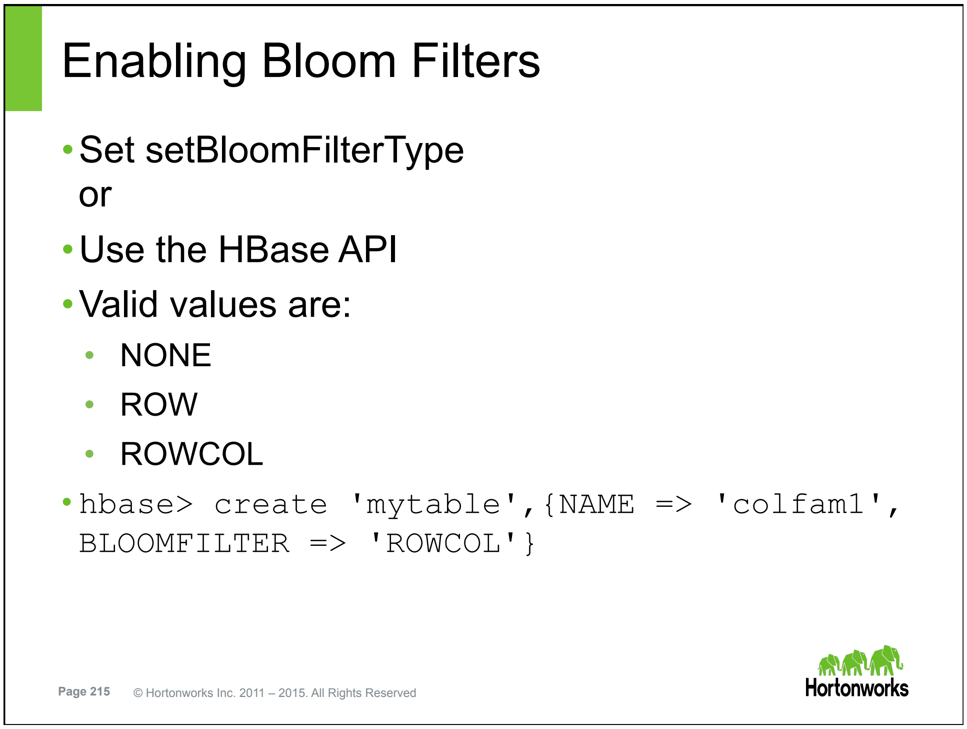
Page 213



When to Use Bloom Filters

- Enabled by default
- Check value of `blockCacheHitRatio`
  - Value should increase if it is filtering out unneeded blocks
- Enable for
  - Row
    or
  - Row + Column

Page 214



Enabling Bloom Filters

- Set setBloomFilterType
  or
- Use the HBase API
- Valid values are:
  - NONE
  - ROW
  - ROWCOL

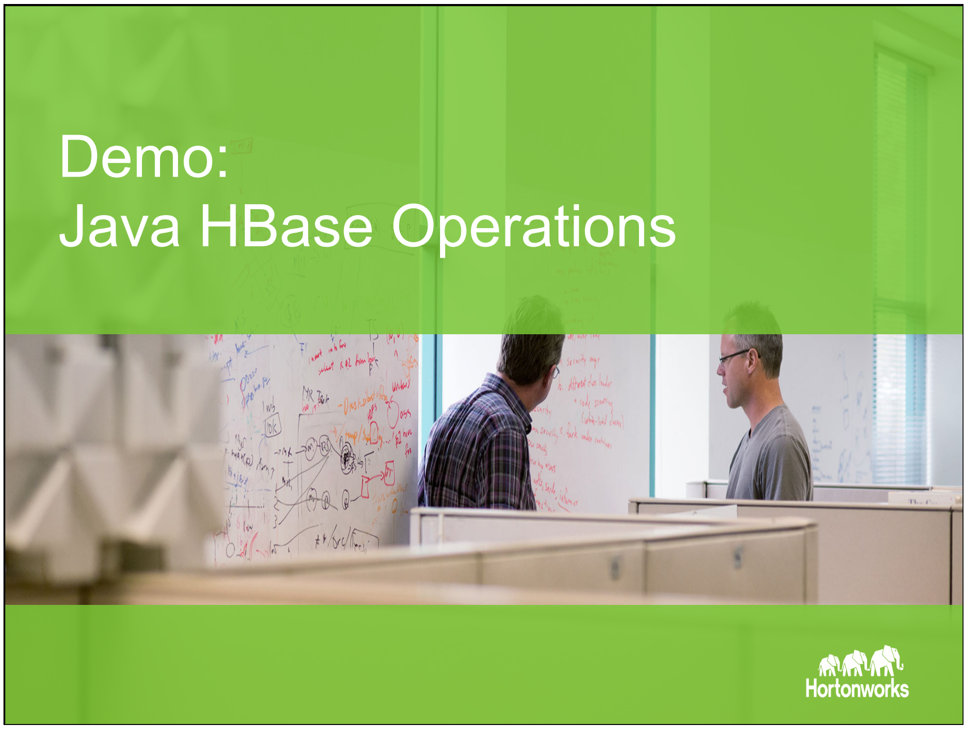
-
  ```
  hbase> create 'mytable',{NAME => 'colfam1',
  BLOOMFILTER => 'ROWCOL'}
  ```

Page 215

Hortonworks

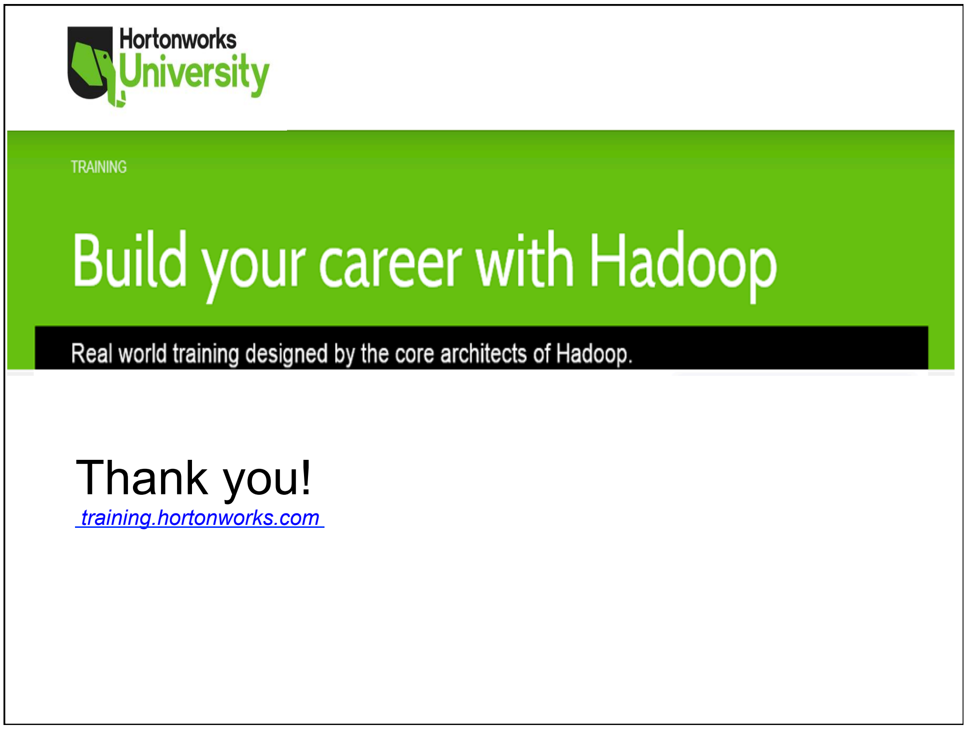
Lab:
Block Size and Bloom Filters

Hortonworks



Demo:
Java HBase Operations

Hortonworks



Hortonworks University

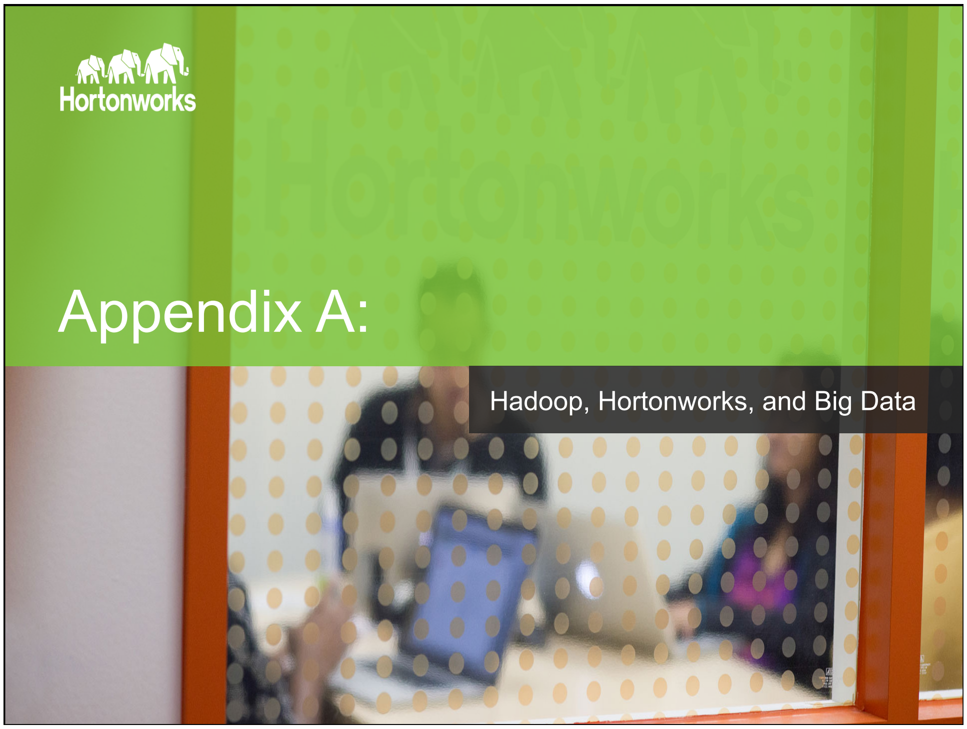
TRAINING

Build your career with Hadoop

Real world training designed by the core architects of Hadoop.

Thank you!

training.hortonworks.com



Hortonworks

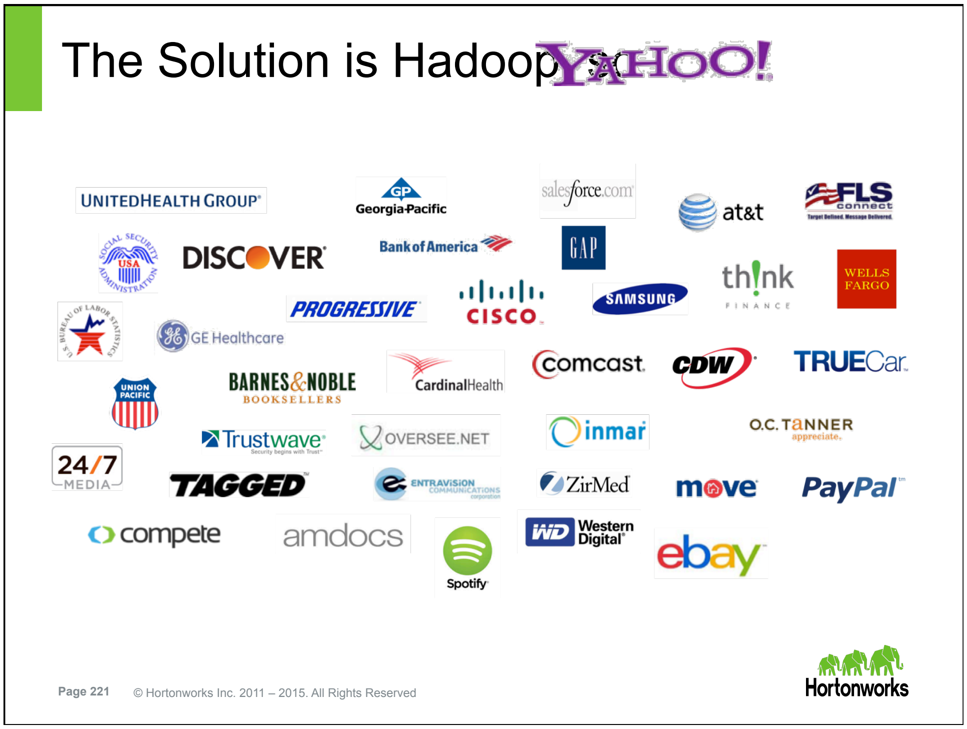
Appendix A:

Hadoop, Hortonworks, and Big Data

Consider These Scenarios

- Researchers at a hospital collect billions of data points from patient sensors. How do they analyze this data in order to improve diagnostics and patient care?
- A large online travel company collects millions of log entries. How do they store and analyze these log entries in order to improve the hotel rankings they provide to their customers?
- A nationwide trucking company collects billions of data points from sensors on their trucks. How do they analyze this data in order save fuel costs?

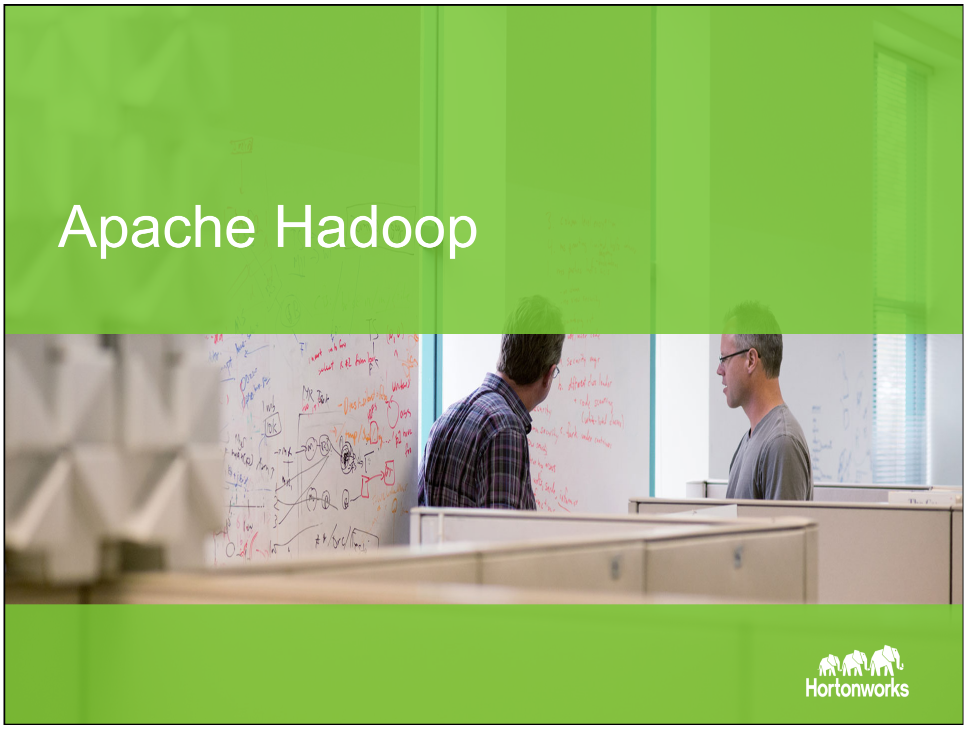
Page 220 Hortonworks



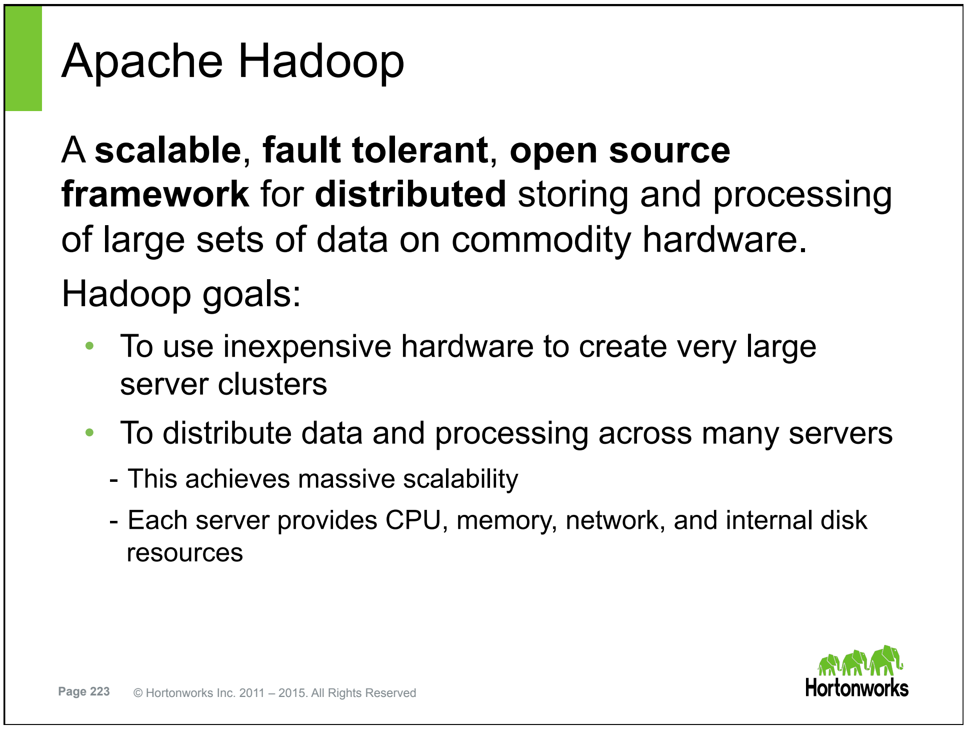
The Solution is Hadoop

YAHOO!
UNITEDHEALTH GROUP
Georgia-Pacific
salesforce.com
at&t
FLS connect
Social Security Administration USA
DISCOVER
Bank of America
GAP
think FINANCE
WELLS FARGO
U.S. Bureau of Labor Statistics
GE Healthcare
PROGRESSIVE
CISCO
SAMSUNG
UNION PACIFIC
BARNES&NOBLE BOOKSELLERS
CardinalHealth
comcast
CDW
TRUECar
Trustwave
OVERSEE.NET
inmar
O.C. TANNER appreciate.
24/7 MEDIA
TAGGED
ENTRAVISION COMMUNICATIONS corporation
ZirMed
move
PayPal
compete
amdocs
Spotify
WD Western Digital
ebay
Hortonworks

Page 221 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Apache Hadoop



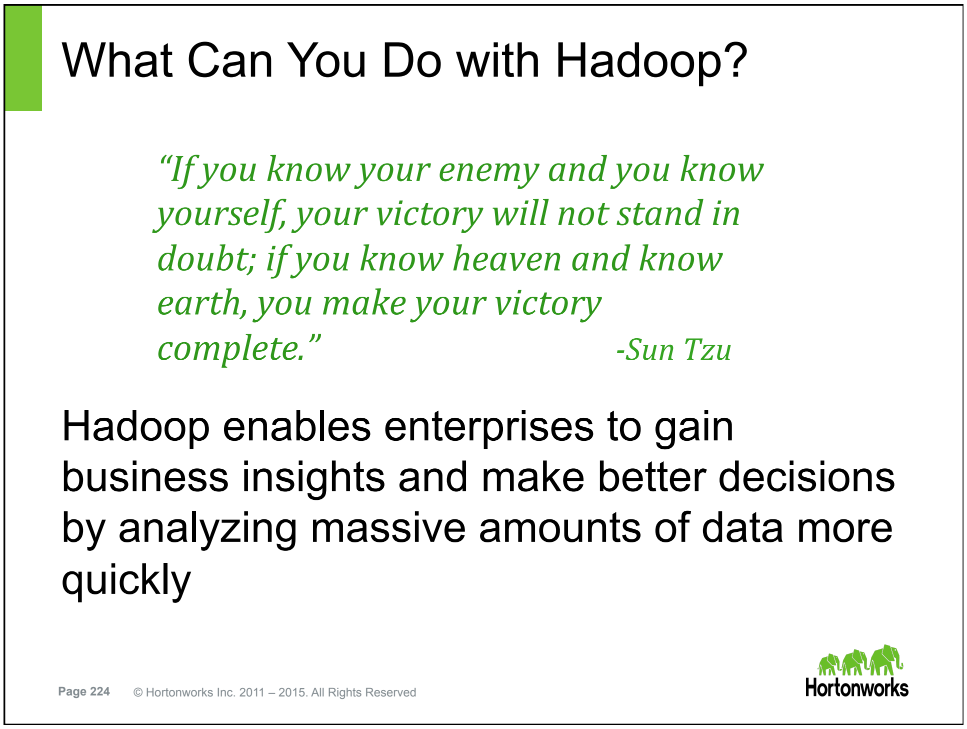
Apache Hadoop

A **scalable**, **fault tolerant**, **open source framework** for **distributed** storing and processing of large sets of data on commodity hardware.

Hadoop goals:

- To use inexpensive hardware to create very large server clusters
- To distribute data and processing across many servers
  - This achieves massive scalability
  - Each server provides CPU, memory, network, and internal disk resources

Page 223

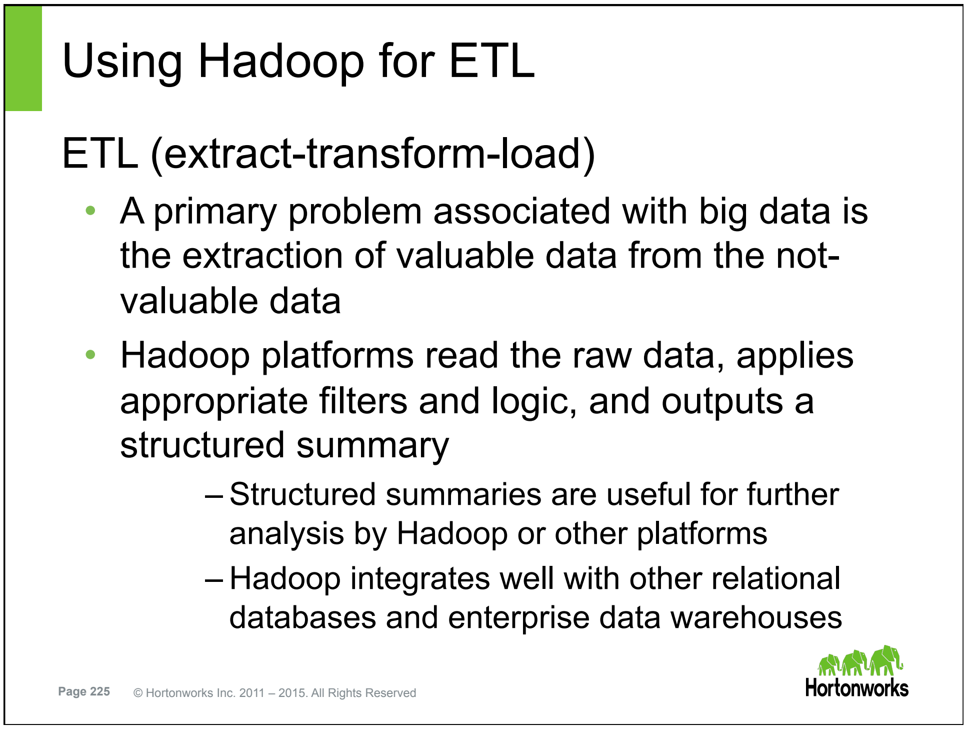


What Can You Do with Hadoop?

> *"If you know your enemy and you know yourself, your victory will not stand in doubt; if you know heaven and know earth, you make your victory complete."* *-Sun Tzu*

Hadoop enables enterprises to gain business insights and make better decisions by analyzing massive amounts of data more quickly

Page 224

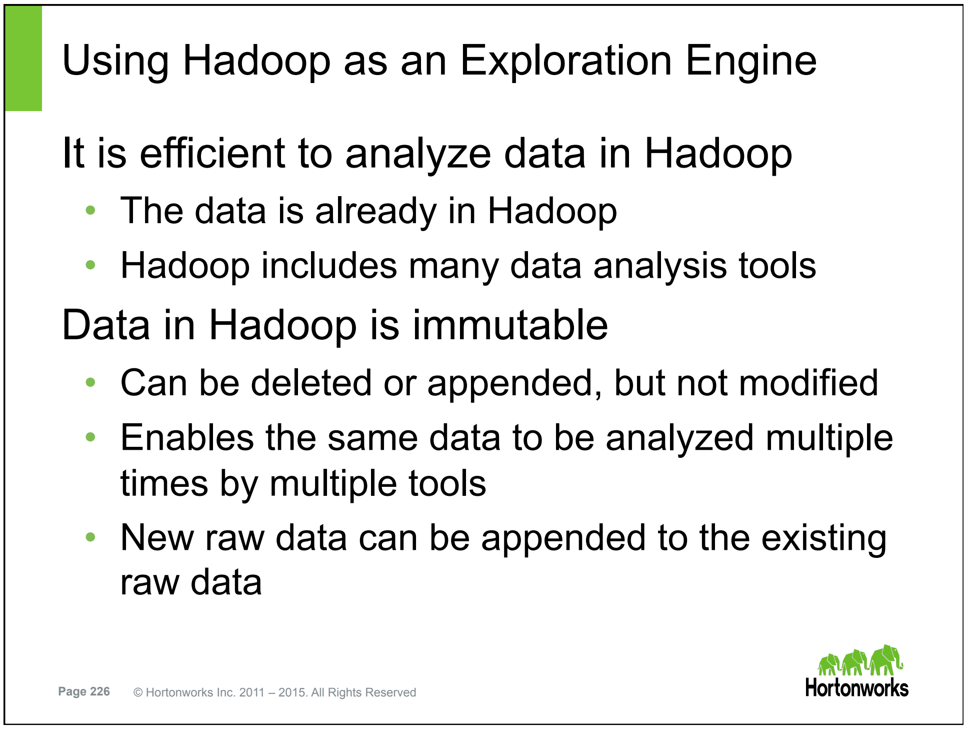


Using Hadoop for ETL

ETL (extract-transform-load)

- A primary problem associated with big data is the extraction of valuable data from the not-valuable data
- Hadoop platforms read the raw data, applies appropriate filters and logic, and outputs a structured summary
  - Structured summaries are useful for further analysis by Hadoop or other platforms
  - Hadoop integrates well with other relational databases and enterprise data warehouses

Page 225



Using Hadoop as an Exploration Engine

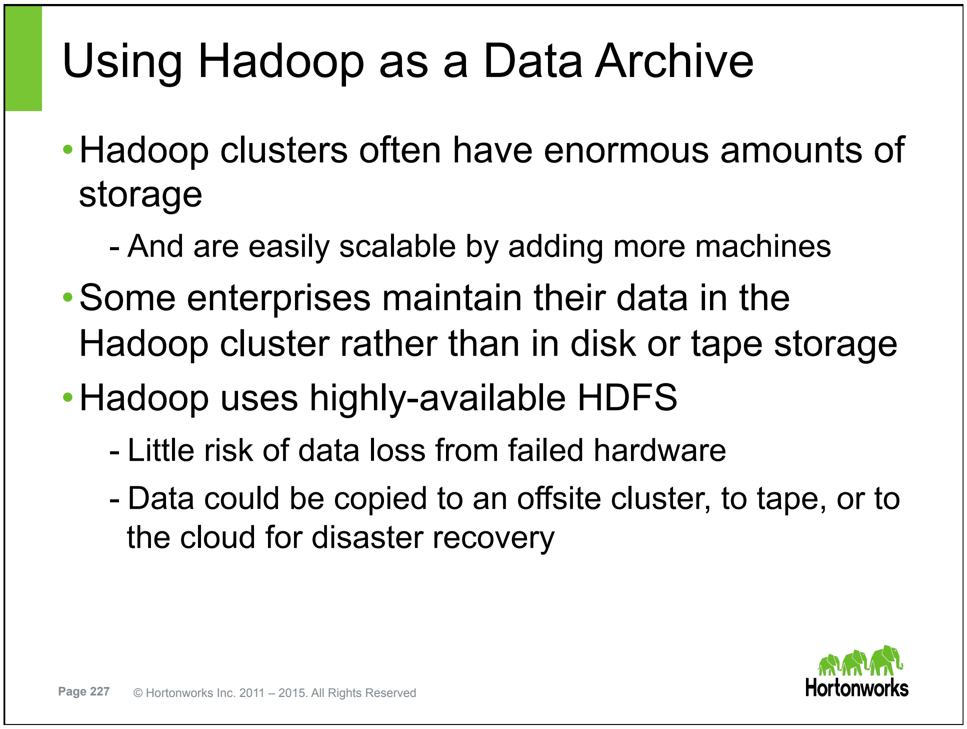
It is efficient to analyze data in Hadoop

- The data is already in Hadoop
- Hadoop includes many data analysis tools

Data in Hadoop is immutable

- Can be deleted or appended, but not modified
- Enables the same data to be analyzed multiple times by multiple tools
- New raw data can be appended to the existing raw data

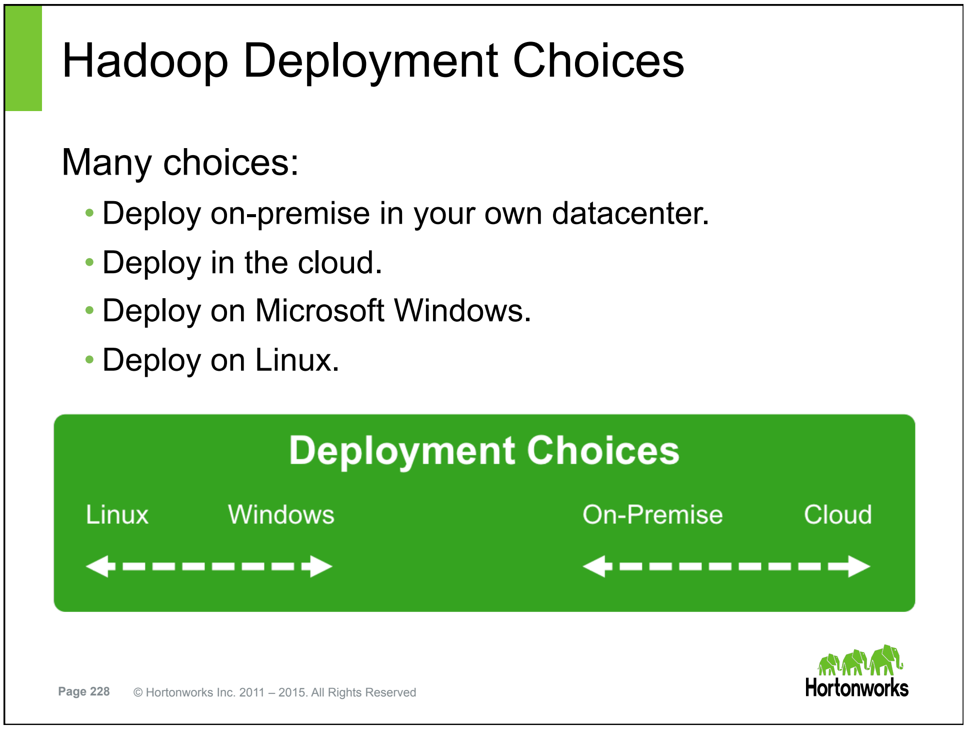
Page 226



Using Hadoop as a Data Archive

- Hadoop clusters often have enormous amounts of storage
  - And are easily scalable by adding more machines
- Some enterprises maintain their data in the Hadoop cluster rather than in disk or tape storage
- Hadoop uses highly-available HDFS
  - Little risk of data loss from failed hardware
  - Data could be copied to an offsite cluster, to tape, or to the cloud for disaster recovery

Page 227 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



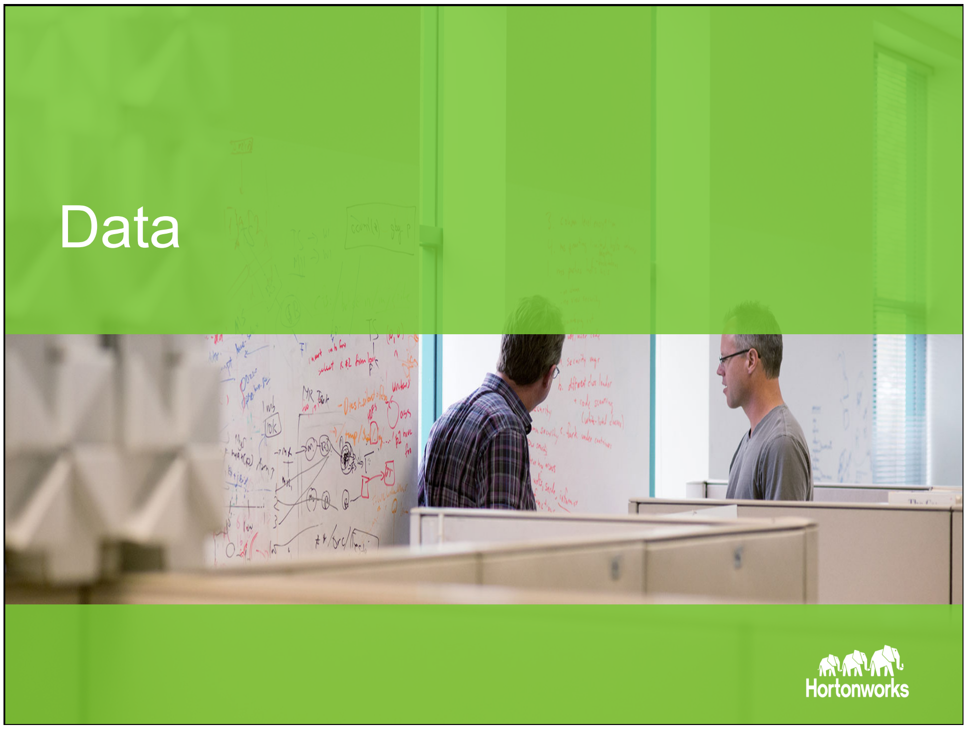
Hadoop Deployment Choices

Many choices:

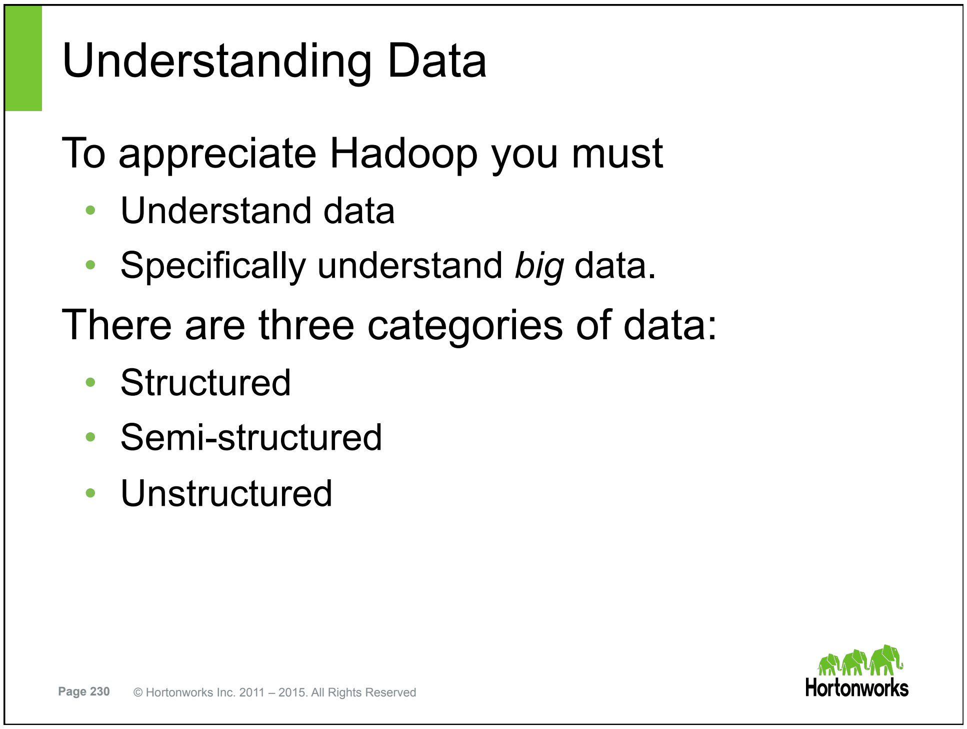
- Deploy on-premise in your own datacenter.
- Deploy in the cloud.
- Deploy on Microsoft Windows.
- Deploy on Linux.

Deployment Choices
Linux
Windows
On-Premise
Cloud

Page 228 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Data



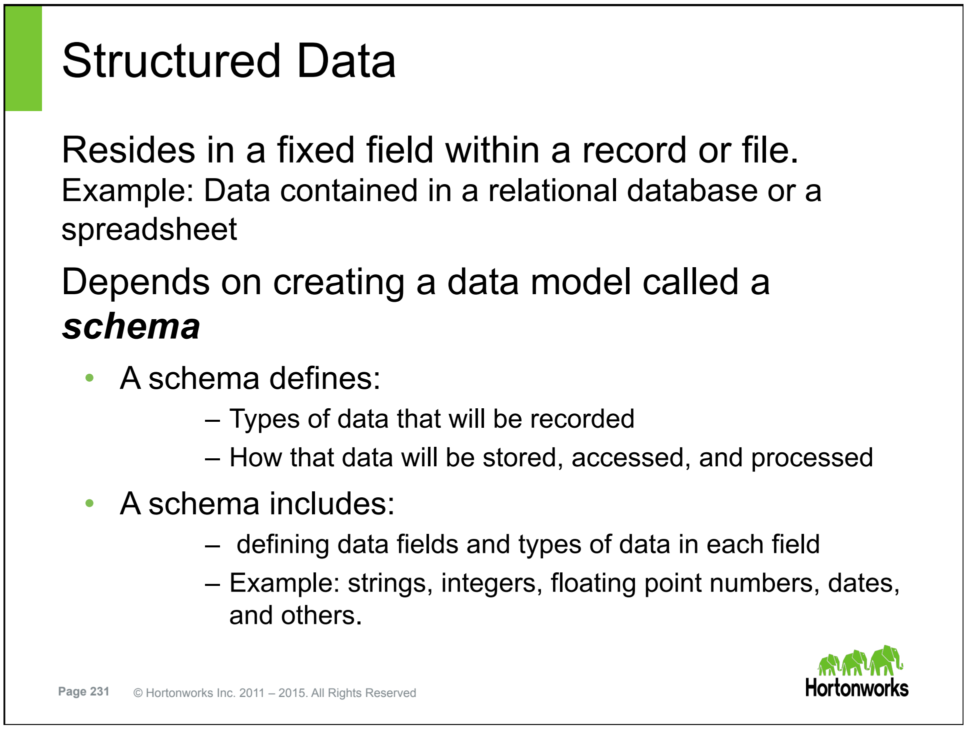
Understanding Data

To appreciate Hadoop you must

- Understand data
- Specifically understand *big* data.

There are three categories of data:

- Structured
- Semi-structured
- Unstructured



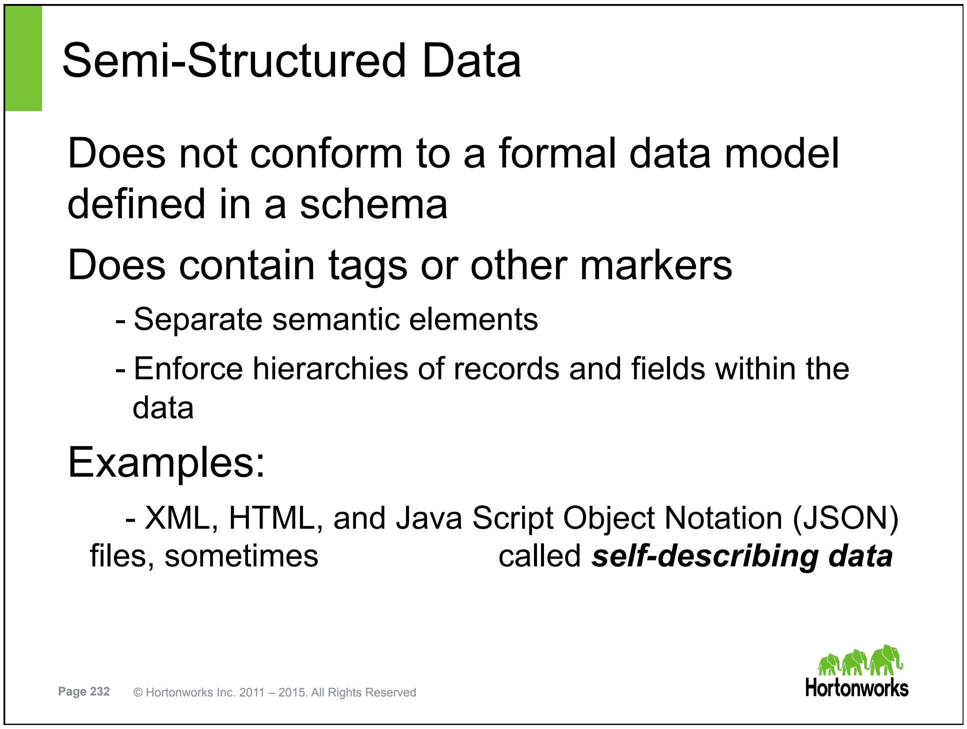
Structured Data

Resides in a fixed field within a record or file.

Example: Data contained in a relational database or a spreadsheet

Depends on creating a data model called a ***schema***

- A schema defines:
  - – Types of data that will be recorded
  - – How that data will be stored, accessed, and processed
- A schema includes:
  - – defining data fields and types of data in each field
  - – Example: strings, integers, floating point numbers, dates, and others.



Semi-Structured Data

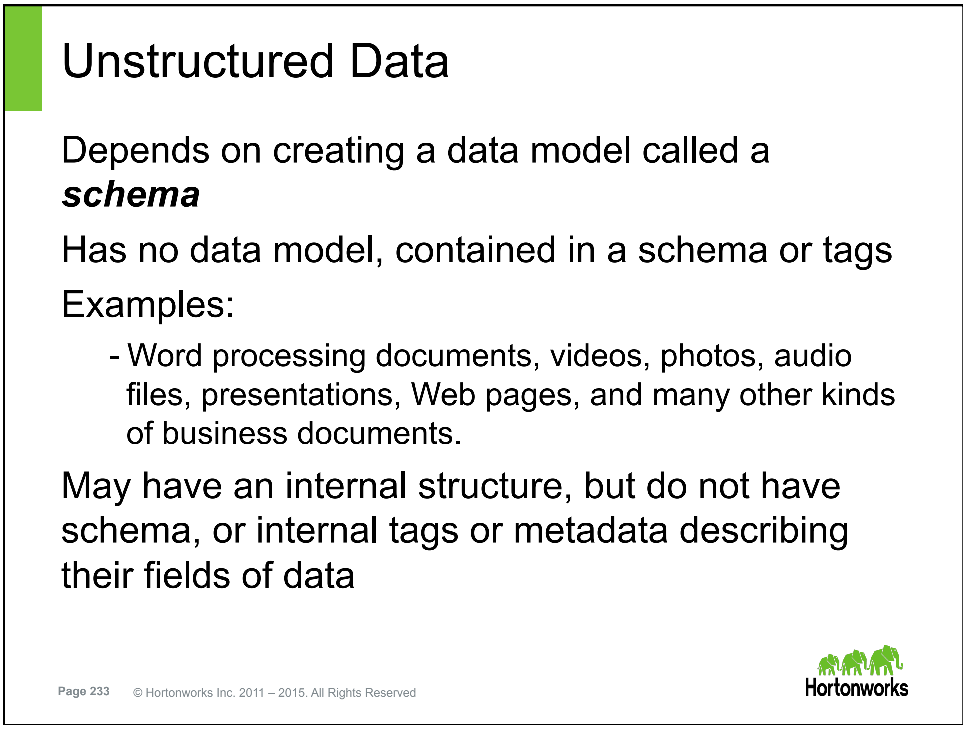
Does not conform to a formal data model defined in a schema

Does contain tags or other markers

- Separate semantic elements
- Enforce hierarchies of records and fields within the data

Examples:

- XML, HTML, and Java Script Object Notation (JSON) files, sometimes called ***self-describing data***



Unstructured Data

Depends on creating a data model called a ***schema***

Has no data model, contained in a schema or tags

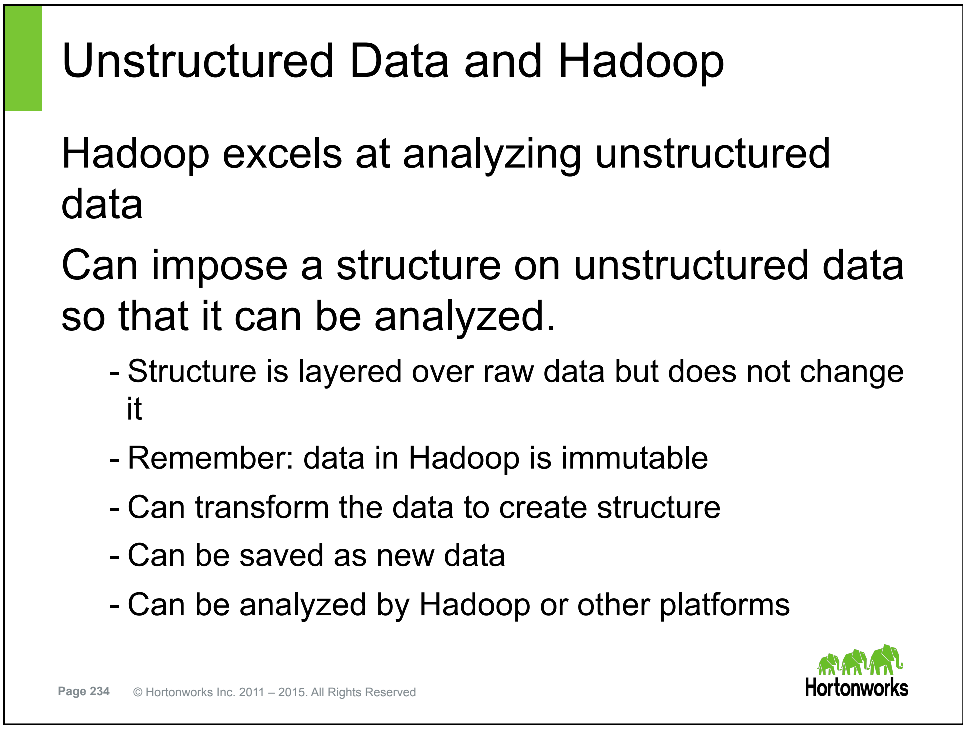
Examples:

- Word processing documents, videos, photos, audio files, presentations, Web pages, and many other kinds of business documents.

May have an internal structure, but do not have schema, or internal tags or metadata describing their fields of data

Page 233

Hortonworks



Unstructured Data and Hadoop

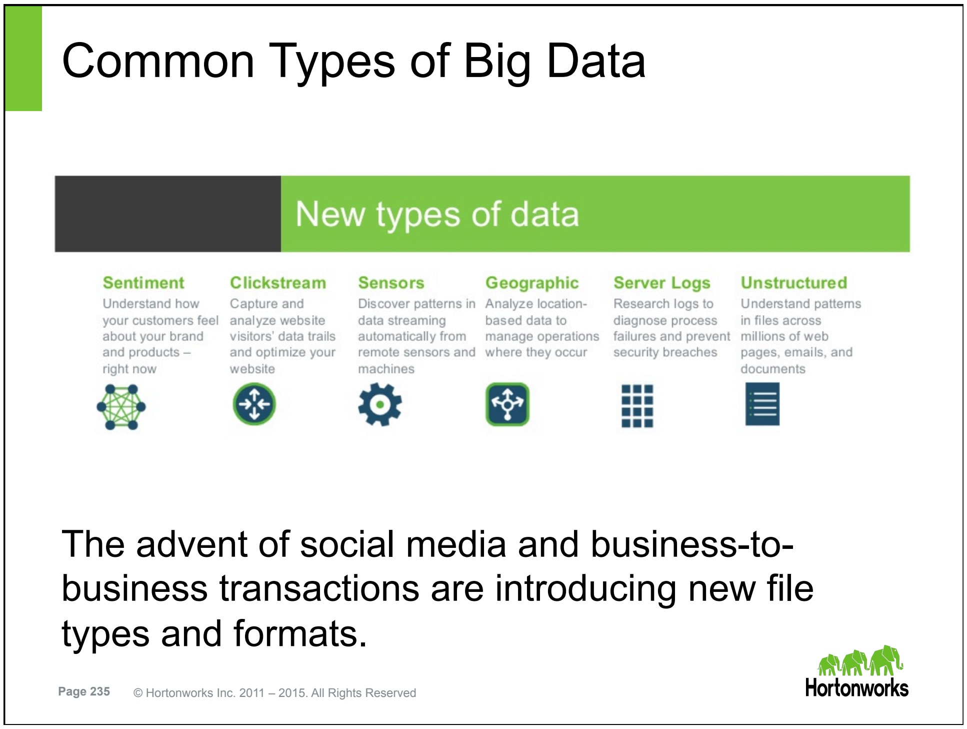
Hadoop excels at analyzing unstructured data

Can impose a structure on unstructured data so that it can be analyzed.

- Structure is layered over raw data but does not change it
- Remember: data in Hadoop is immutable
- Can transform the data to create structure
- Can be saved as new data
- Can be analyzed by Hadoop or other platforms

Page 234

Hortonworks



Common Types of Big Data

New types of data

Sentiment
Understand how your customers feel about your brand and products – right now

Clickstream
Capture and analyze website visitors' data trails and optimize your website

Sensors
Discover patterns in data streaming automatically from remote sensors and machines

Geographic
Analyze location-based data to manage operations where they occur

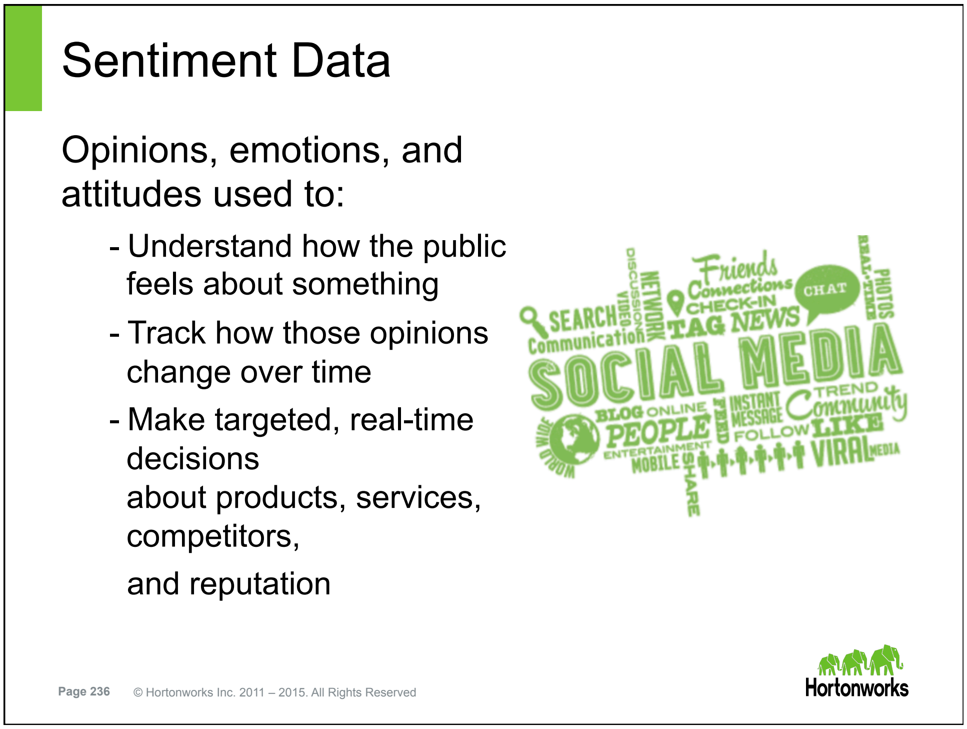
Server Logs
Research logs to diagnose process failures and prevent security breaches

Unstructured
Understand patterns in files across millions of web pages, emails, and documents

The advent of social media and business-to-business transactions are introducing new file types and formats.

Page 235 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Sentiment Data

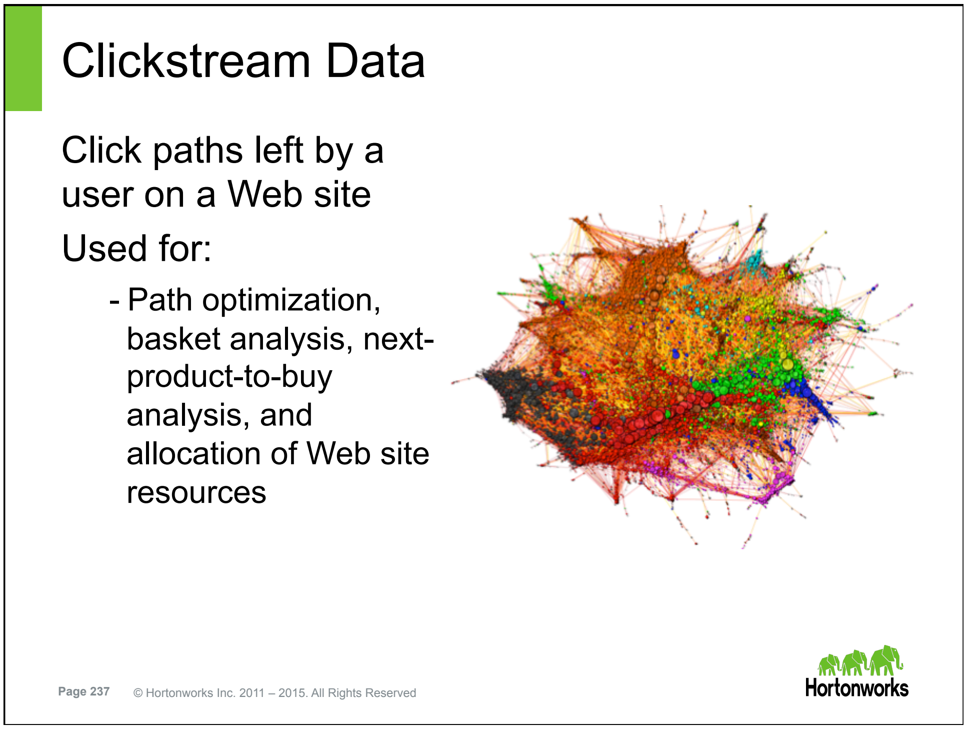
Opinions, emotions, and attitudes used to:

- Understand how the public feels about something
- Track how those opinions change over time
- Make targeted, real-time decisions
  about products, services, competitors,
  and reputation

SOCIAL MEDIA
SEARCH
Communication
VIDEO
DISCUSSION
NETWORK
Friends
Connections
CHECK-IN
TAG
NEWS
CHAT
REAL-TIME
PHOTOS
BLOG
ONLINE
PEOPLE
ENTERTAINMENT
MOBILE
SHARE
WORLD WIDE
INSTANT MESSAGE
FOLLOW
TREND
Community
LIKE
VIRAL
MEDIA

Page 236 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



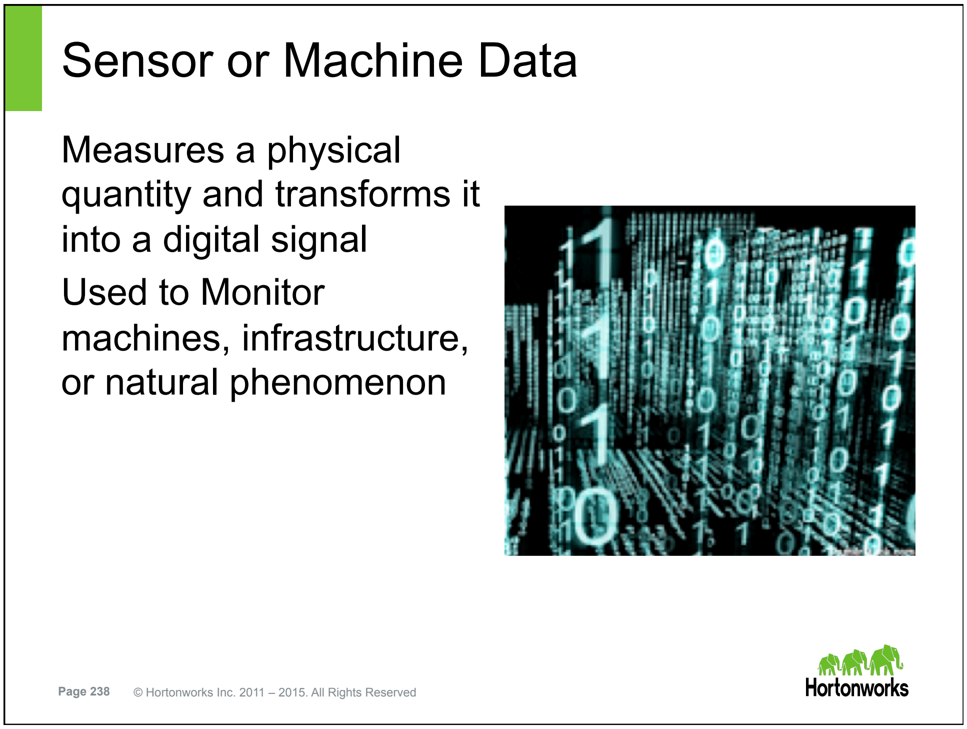
Clickstream Data

Click paths left by a user on a Web site

Used for:

- Path optimization, basket analysis, next-product-to-buy analysis, and allocation of Web site resources

Page 237



Sensor or Machine Data

Measures a physical quantity and transforms it into a digital signal

Used to Monitor machines, infrastructure, or natural phenomenon

Page 238

Geolocation Data

Location of an object or individual at a moment in time

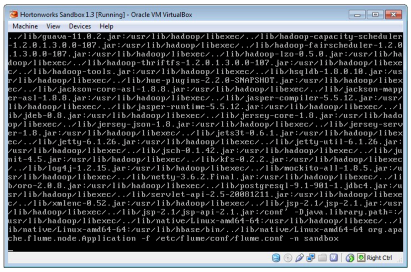
May take the form of coordinates or an actual street address

Used to enable location-based marketing, and real-time customer support

Server Log Data

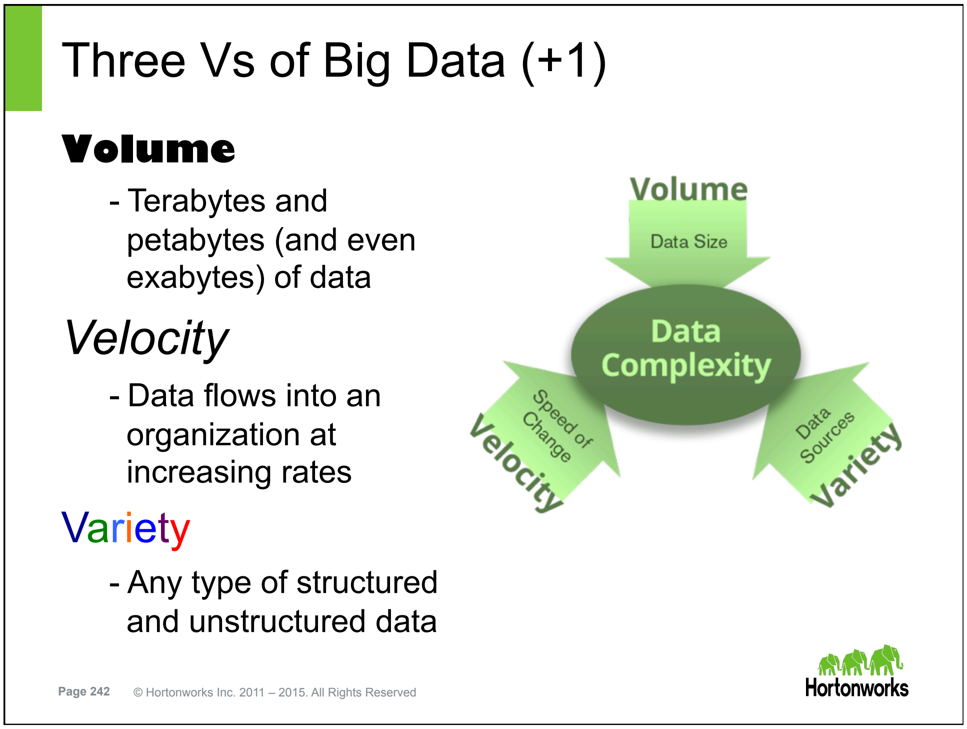
Captures network and server operations data

Used for managing network operations, especially for security and regulatory compliance



```
../lib/guava-11.0.2.jar:/usr/lib/hadoop/libexec/../lib/hadoop-capacity-scheduler
-1.2.0.1.3.0.0-107.jar:/usr/lib/hadoop/libexec/../lib/hadoop-fairscheduler-1.2.0
.1.3.0.0-107.jar:/usr/lib/hadoop/libexec/../lib/hadoop-lzo-0.5.0.jar:/usr/lib/ha
doop/libexec/../lib/hadoop-thriftfs-1.2.0.1.3.0.0-107.jar:/usr/lib/hadoop/libexe
c/../lib/hadoop-tools.jar:/usr/lib/hadoop/libexec/../lib/hsqldb-1.8.0.10.jar:/us
r/lib/hadoop/libexec/../lib/hue-plugins-2.2.0-SNAPSHOT.jar:/usr/lib/hadoop/libex
ec/../lib/jackson-core-asl-1.8.8.jar:/usr/lib/hadoop/libexec/../lib/jackson-mapp
er-asl-1.8.8.jar:/usr/lib/hadoop/libexec/../lib/jasper-compiler-5.5.12.jar:/usr/
lib/hadoop/libexec/../lib/jasper-runtime-5.5.12.jar:/usr/lib/hadoop/libexec/../l
ib/jdeb-0.8.jar:/usr/lib/hadoop/libexec/../lib/jersey-core-1.8.jar:/usr/lib/hado
op/libexec/../lib/jersey-json-1.8.jar:/usr/lib/hadoop/libexec/../lib/jersey-serv
er-1.8.jar:/usr/lib/hadoop/libexec/../lib/jets3t-0.6.1.jar:/usr/lib/hadoop/libex
ec/../lib/jetty-6.1.26.jar:/usr/lib/hadoop/libexec/../lib/jetty-util-6.1.26.jar:
/usr/lib/hadoop/libexec/../lib/jsch-0.1.42.jar:/usr/lib/hadoop/libexec/../lib/ju
nit-4.5.jar:/usr/lib/hadoop/libexec/../lib/kfs-0.2.2.jar:/usr/lib/hadoop/libexec
/../lib/log4j-1.2.15.jar:/usr/lib/hadoop/libexec/../lib/mockito-all-1.8.5.jar:/u
sr/lib/hadoop/libexec/../lib/netty-3.6.2.Final.jar:/usr/lib/hadoop/libexec/../li
b/oro-2.0.8.jar:/usr/lib/hadoop/libexec/../lib/postgresql-9.1-901-1.jdbc4.jar:/u
sr/lib/hadoop/libexec/../lib/servlet-api-2.5-20081211.jar:/usr/lib/hadoop/libexe
c/../lib/xmlenc-0.52.jar:/usr/lib/hadoop/libexec/../lib/jsp-2.1/jsp-2.1.jar:/usr
/lib/hadoop/libexec/../lib/jsp-2.1/jsp-api-2.1.jar:/conf' -Djava.library.path=:/
usr/lib/hadoop/libexec/../lib/native/Linux-amd64-64:/usr/lib/hadoop/libexec/../l
ib/native/Linux-amd64-64:/usr/lib/hbase/bin/../lib/native/Linux-amd64-64 org.apa
che.flume.node.Application -f /etc/flume/conf/flume.conf -n sandbox
_
```

Big Data



Three Vs of Big Data (+1)

Volume

- Terabytes and petabytes (and even exabytes) of data

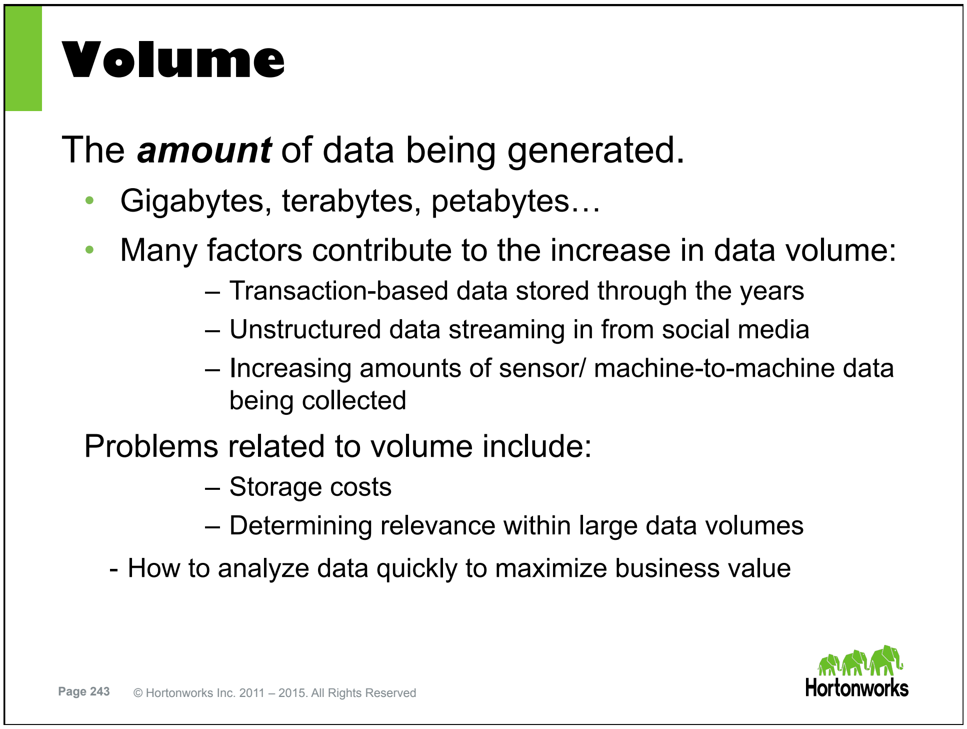
Velocity

- Data flows into an organization at increasing rates

Variety

- Any type of structured and unstructured data

Page 242 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Volume

The ***amount*** of data being generated.

- Gigabytes, terabytes, petabytes…
- Many factors contribute to the increase in data volume:
  - Transaction-based data stored through the years
  - Unstructured data streaming in from social media
  - Increasing amounts of sensor/ machine-to-machine data being collected

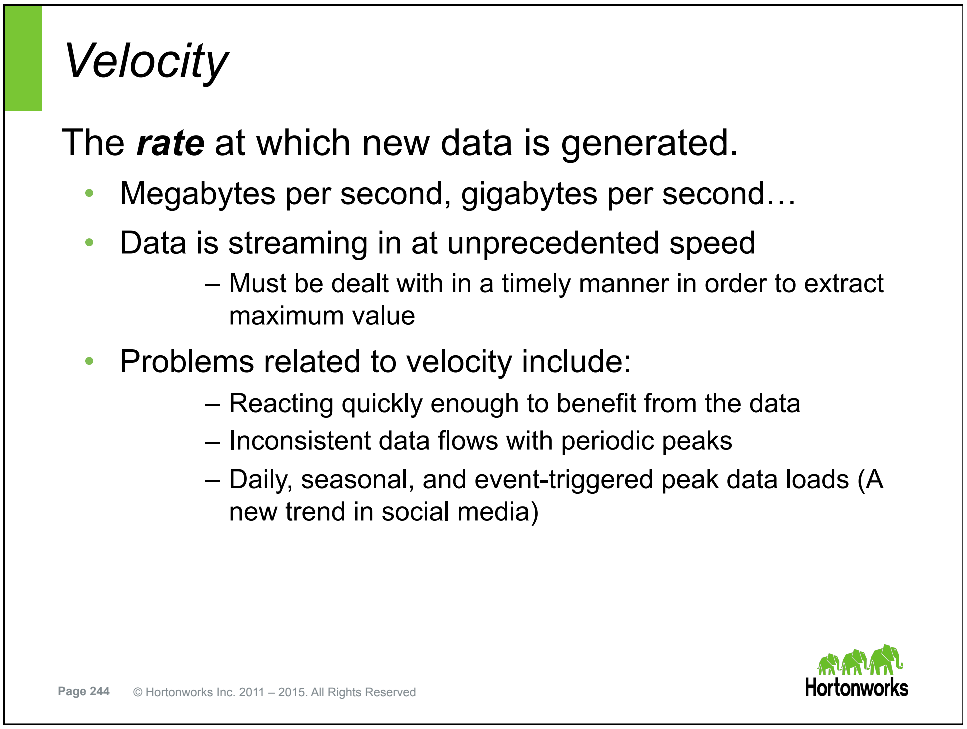
Problems related to volume include:

- Storage costs
- Determining relevance within large data volumes

- How to analyze data quickly to maximize business value

Page 243 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



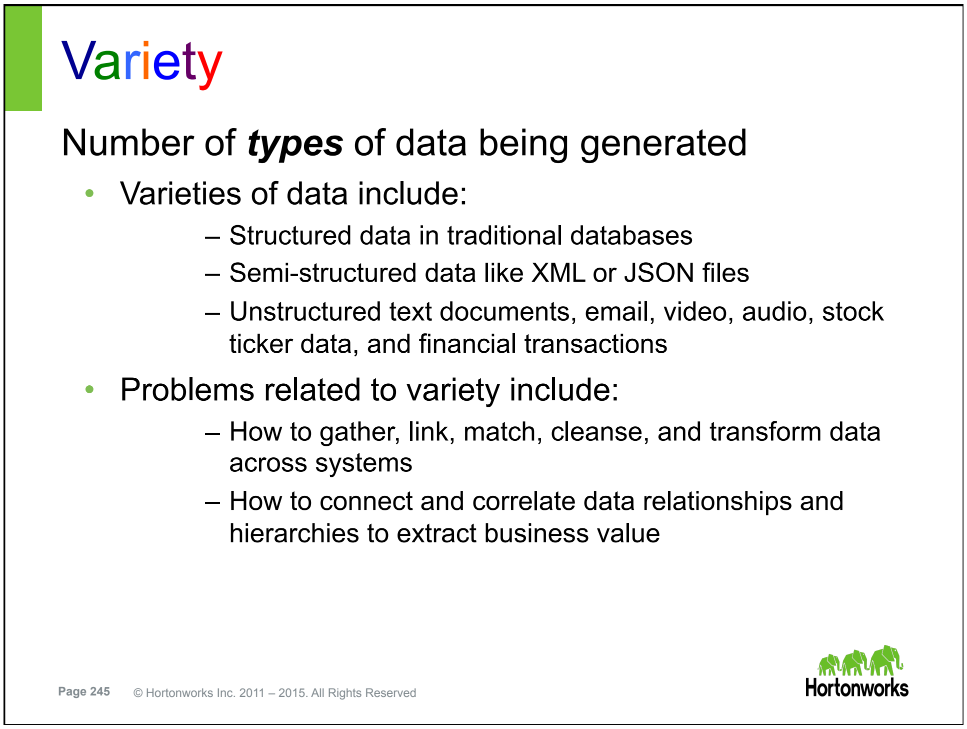
Velocity

The ***rate*** at which new data is generated.

- Megabytes per second, gigabytes per second…
- Data is streaming in at unprecedented speed
  - Must be dealt with in a timely manner in order to extract maximum value
- Problems related to velocity include:
  - Reacting quickly enough to benefit from the data
  - Inconsistent data flows with periodic peaks
  - Daily, seasonal, and event-triggered peak data loads (A new trend in social media)

Page 244 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

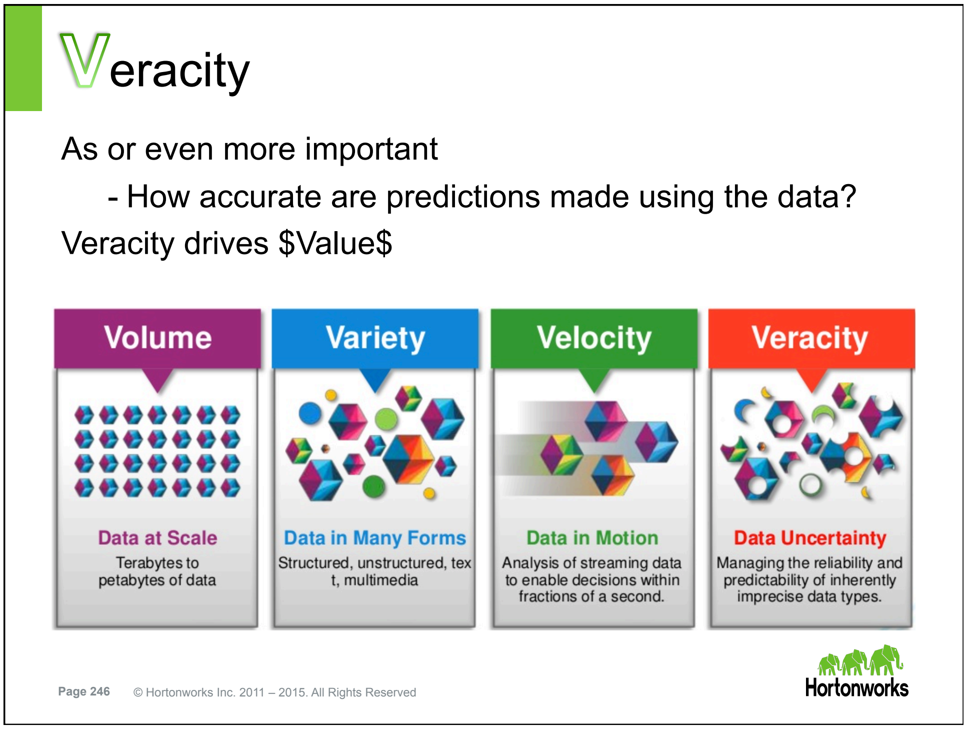
Hortonworks



Variety

Number of ***types*** of data being generated

- Varieties of data include:
  – Structured data in traditional databases
  – Semi-structured data like XML or JSON files
  – Unstructured text documents, email, video, audio, stock ticker data, and financial transactions
- Problems related to variety include:
  – How to gather, link, match, cleanse, and transform data across systems
  – How to connect and correlate data relationships and hierarchies to extract business value



Veracity

As or even more important

- How accurate are predictions made using the data?

Veracity drives $Value$

Volume

Data at Scale

Terabytes to petabytes of data

Variety

Data in Many Forms

Structured, unstructured, tex t, multimedia

Velocity

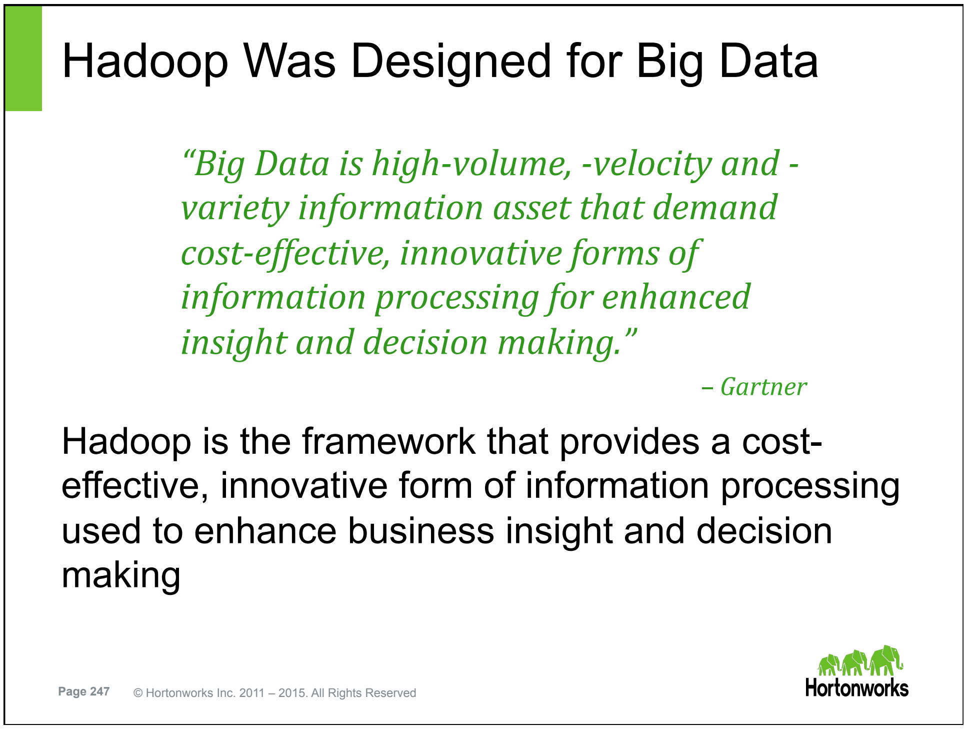
Data in Motion

Analysis of streaming data to enable decisions within fractions of a second.

Veracity

Data Uncertainty

Managing the reliability and predictability of inherently imprecise data types.



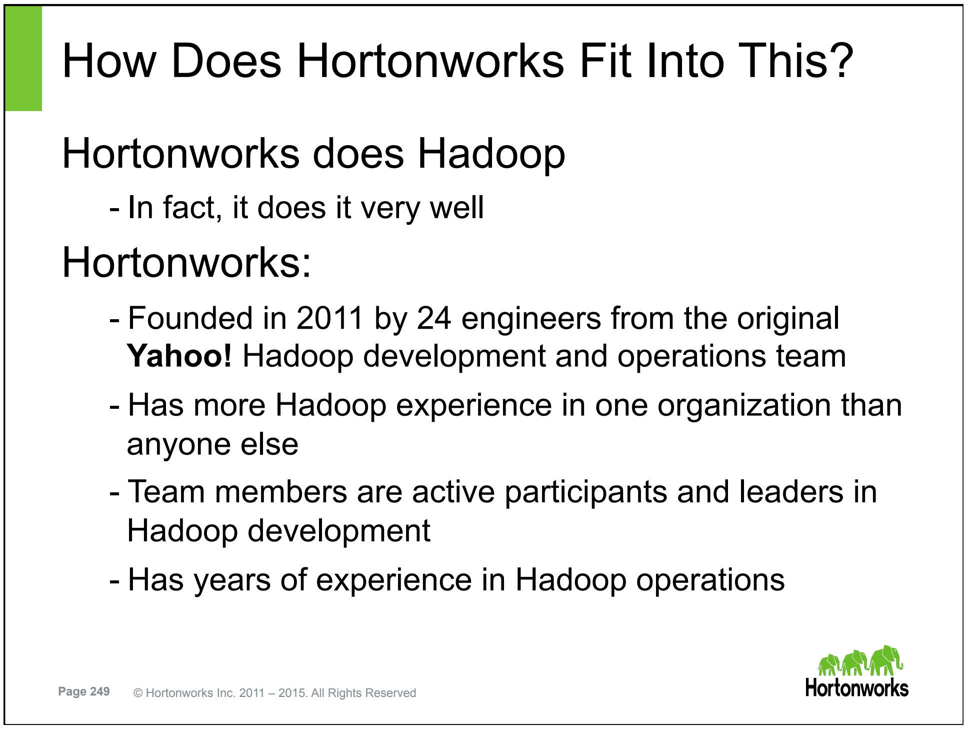
Hadoop Was Designed for Big Data

> *“Big Data is high-volume, -velocity and -variety information asset that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.”*
>
> *– Gartner*

Hadoop is the framework that provides a cost-effective, innovative form of information processing used to enhance business insight and decision making

Page 247 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



How Does Hortonworks Fit Into This?

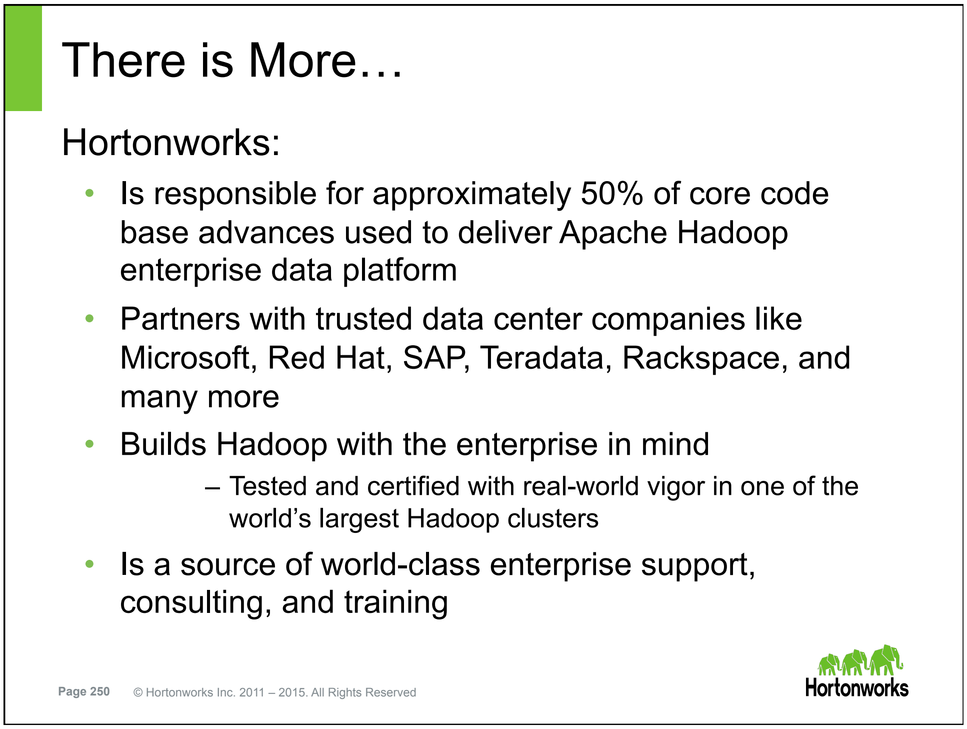
Hortonworks does Hadoop

- In fact, it does it very well

Hortonworks:

- Founded in 2011 by 24 engineers from the original **Yahoo!** Hadoop development and operations team
- Has more Hadoop experience in one organization than anyone else
- Team members are active participants and leaders in Hadoop development
- Has years of experience in Hadoop operations

Page 249

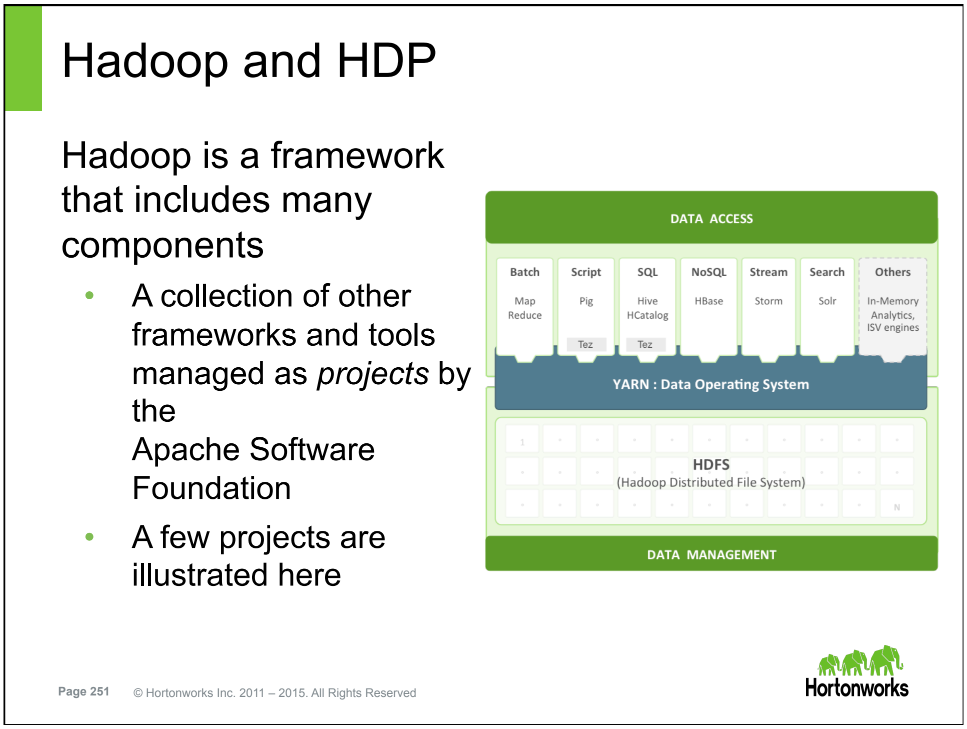


There is More…

Hortonworks:

- Is responsible for approximately 50% of core code base advances used to deliver Apache Hadoop enterprise data platform
- Partners with trusted data center companies like Microsoft, Red Hat, SAP, Teradata, Rackspace, and many more
- Builds Hadoop with the enterprise in mind
  - Tested and certified with real-world vigor in one of the world's largest Hadoop clusters
- Is a source of world-class enterprise support, consulting, and training

Page 250



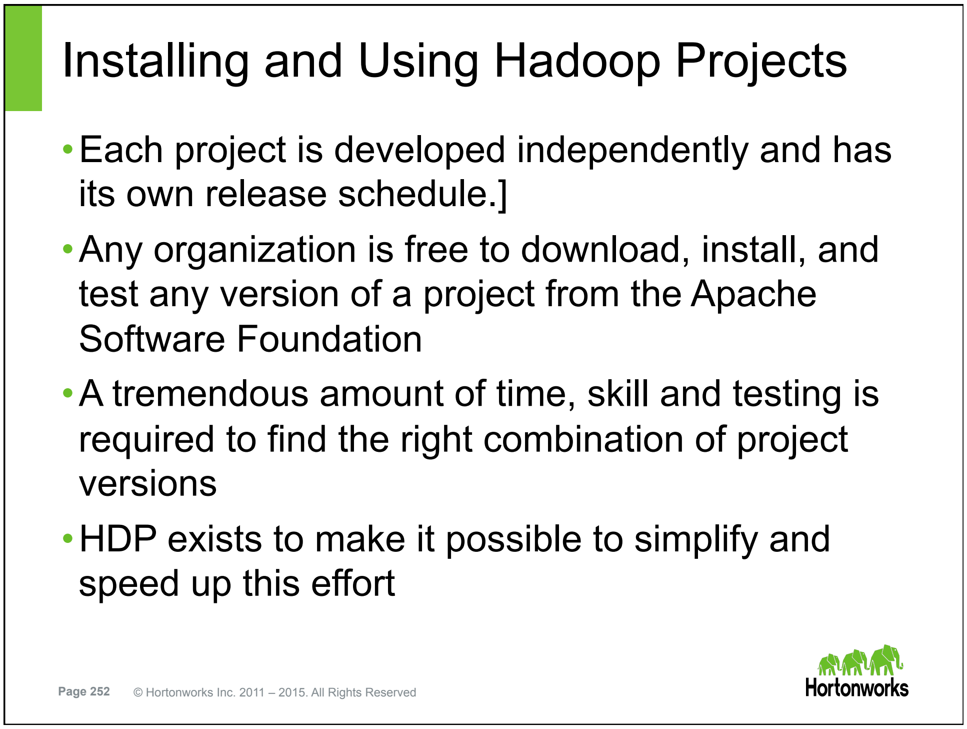
Hadoop and HDP

Hadoop is a framework that includes many components

- A collection of other frameworks and tools managed as *projects* by the Apache Software Foundation
- A few projects are illustrated here

Page 251 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Installing and Using Hadoop Projects

- Each project is developed independently and has its own release schedule.]
- Any organization is free to download, install, and test any version of a project from the Apache Software Foundation
- A tremendous amount of time, skill and testing is required to find the right combination of project versions
- HDP exists to make it possible to simplify and speed up this effort

Page 252 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

Getting Back to HDP – What is it?

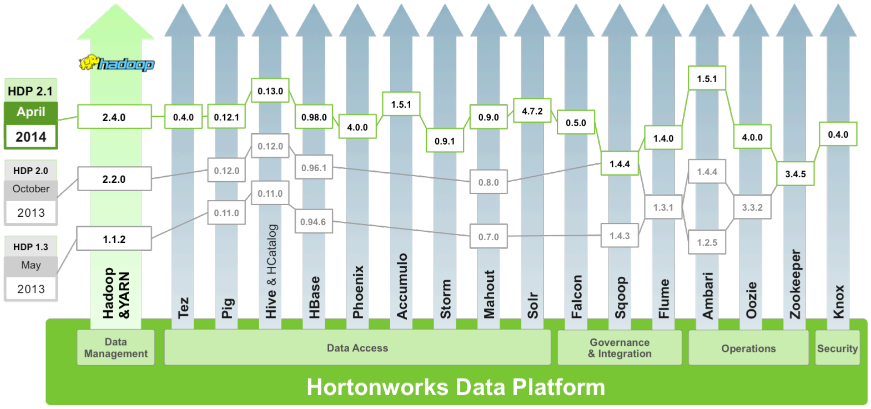
It is an enterprise version of Hadoop distributed by Hortonworks

- Includes a single installation utility to install most of the Apache Hadoop software
- Goes through rigorous system, function, and regression testing
  - – Ensures that any versions of any projects included in the distribution work seamlessly together in a secure and reliable manner

Page 253

Hortonworks

HDP and Hadoop Project Versions



Page 254

Hortonworks

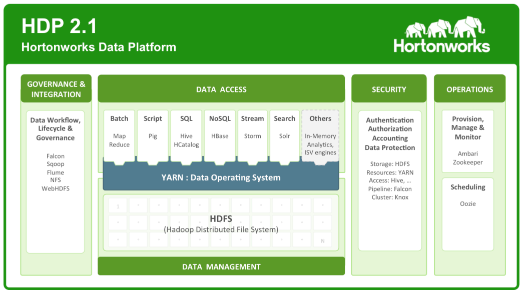
HDP is 100 Percent Pure Hadoop

Contains no proprietary software add-ons or extensions

- 100 percent Apache Hadoop software with no possibility of vendor lock-in
- Includes latest enhancements and fixes developed by a worldwide set of developers
- Other software often runs along side of Hadoop but might not work properly with proprietary extensions and features.

Page 255

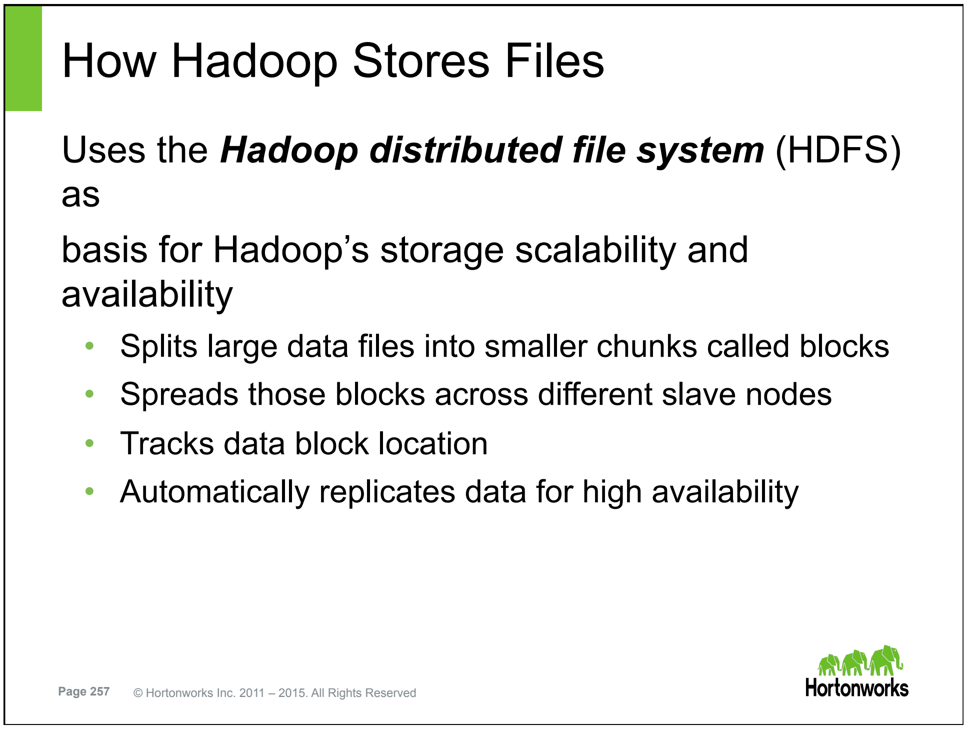
HDP Delivers a Comprehensive Data Management Platform



Includes projects for:

- Core Hadoop (HDFS and YARN)
- Data access and processing
- Data governance and integration
- Security
- Managing Hadoop operations

Page 256



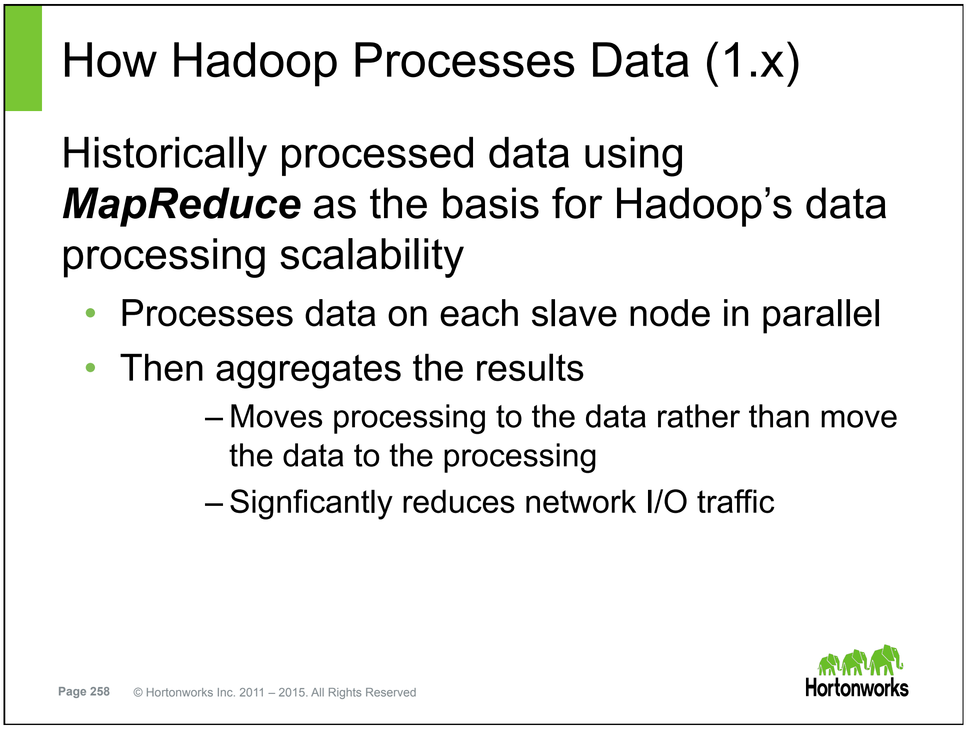
How Hadoop Stores Files

Uses the ***Hadoop distributed file system*** (HDFS) as

basis for Hadoop's storage scalability and availability

- Splits large data files into smaller chunks called blocks
- Spreads those blocks across different slave nodes
- Tracks data block location
- Automatically replicates data for high availability

Page 257

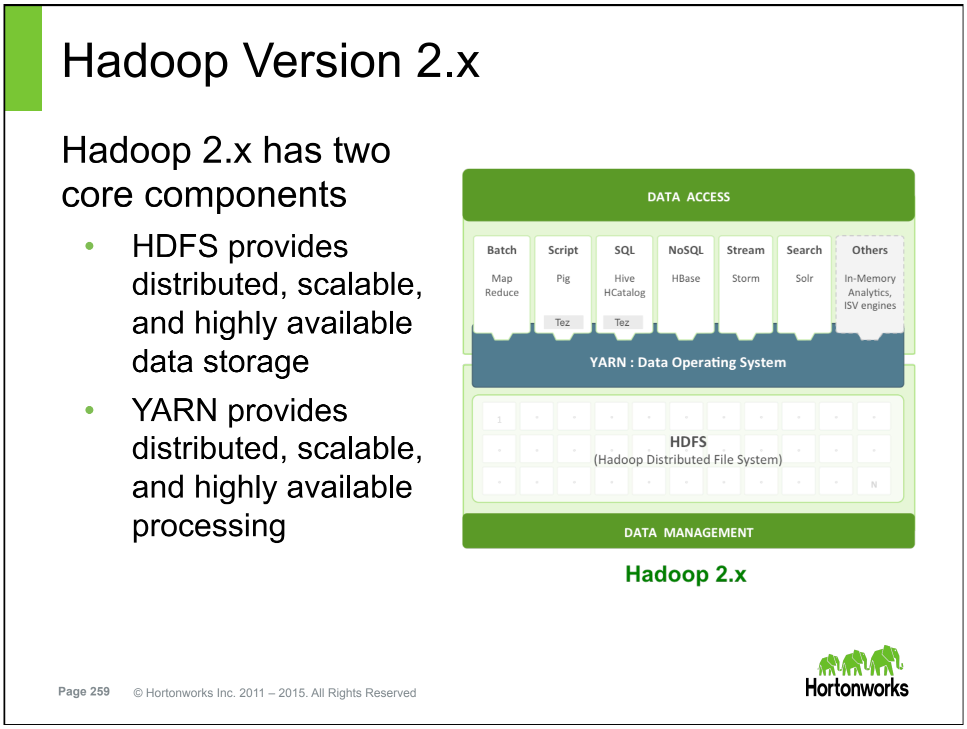


How Hadoop Processes Data (1.x)

Historically processed data using ***MapReduce*** as the basis for Hadoop's data processing scalability

- Processes data on each slave node in parallel
- Then aggregates the results
  - Moves processing to the data rather than move the data to the processing
  - Signficantly reduces network I/O traffic

Page 258



Hadoop Version 2.x

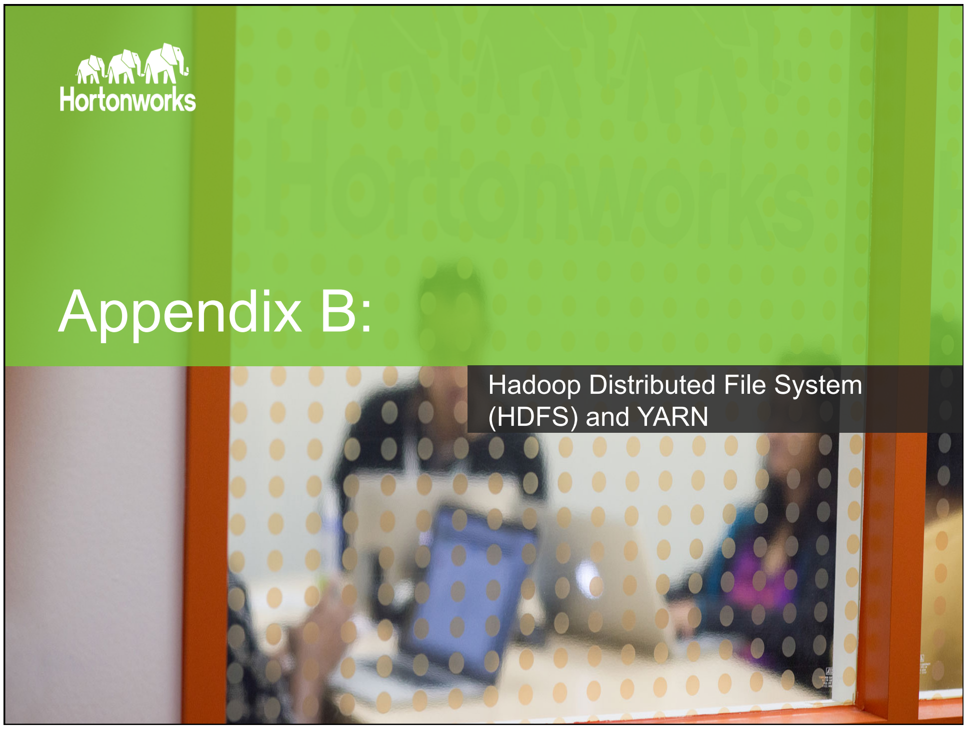
Hadoop 2.x has two core components

- HDFS provides distributed, scalable, and highly available data storage
- YARN provides distributed, scalable, and highly available processing

Hadoop 2.x

Page 259

Hortonworks

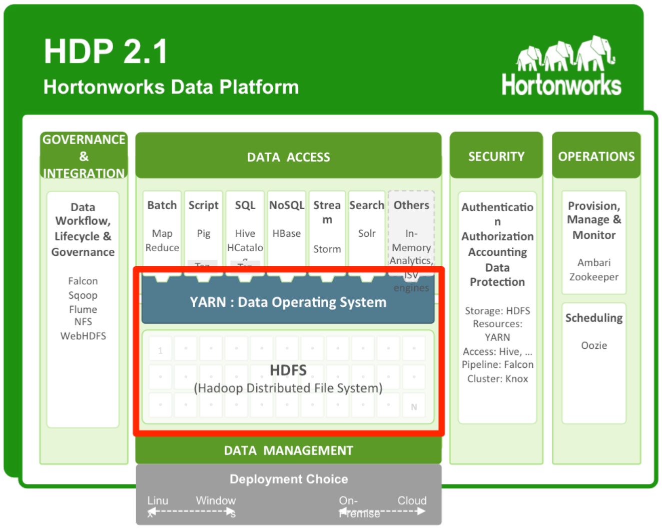


Hortonworks

Appendix B:

Hadoop Distributed File System (HDFS) and YARN

HDFS and YARN: Part of the Core of Hadoop



HDP 2.1

Hortonworks Data Platform

GOVERNANCE & INTEGRATION

Data Workflow, Lifecycle & Governance

Falcon
Sqoop
Flume
NFS
WebHDFS

DATA ACCESS

Batch	Script	SQL	NoSQL	Strea m	Search	Others
Map Reduce	Pig	Hive HCatalo	HBase	Storm	Solr	In-Memory Analytics, ISV engines

YARN : Data Operating System

HDFS
(Hadoop Distributed File System)

DATA MANAGEMENT

SECURITY

Authenticatio n
Authorization
Accounting
Data Protection

Storage: HDFS
Resources: YARN
Access: Hive, …
Pipeline: Falcon
Cluster: Knox

OPERATIONS

Provision, Manage & Monitor

Ambari
Zookeeper

Scheduling

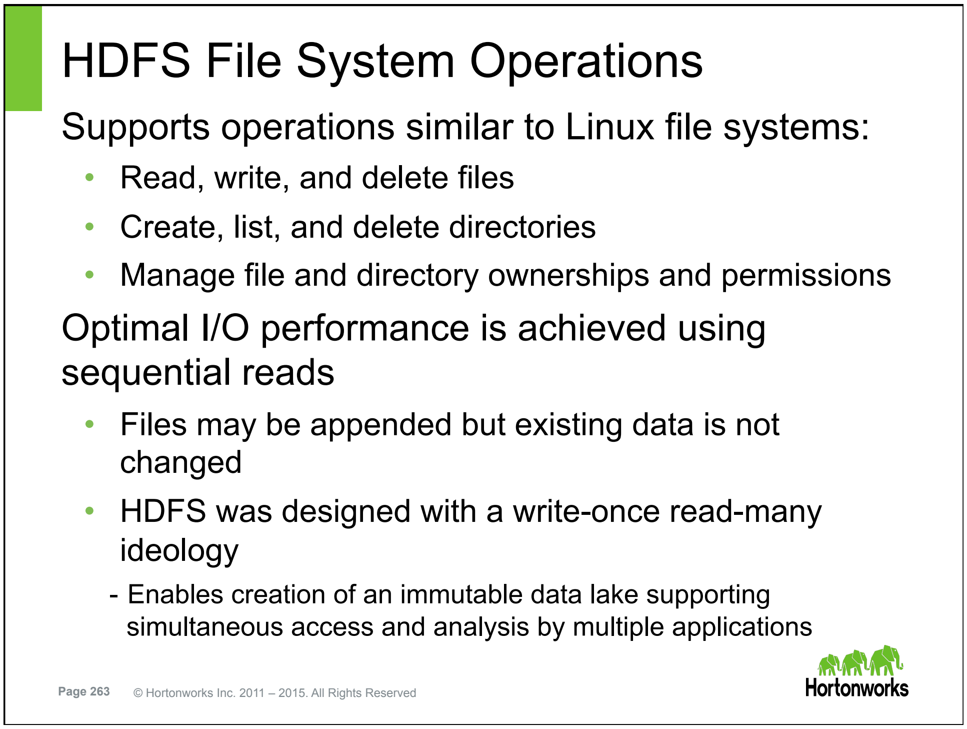
Oozie

Deployment Choice

Linu x ← → Window s On-Premise ← → Cloud

Page 261

HDFS



HDFS File System Operations

Supports operations similar to Linux file systems:

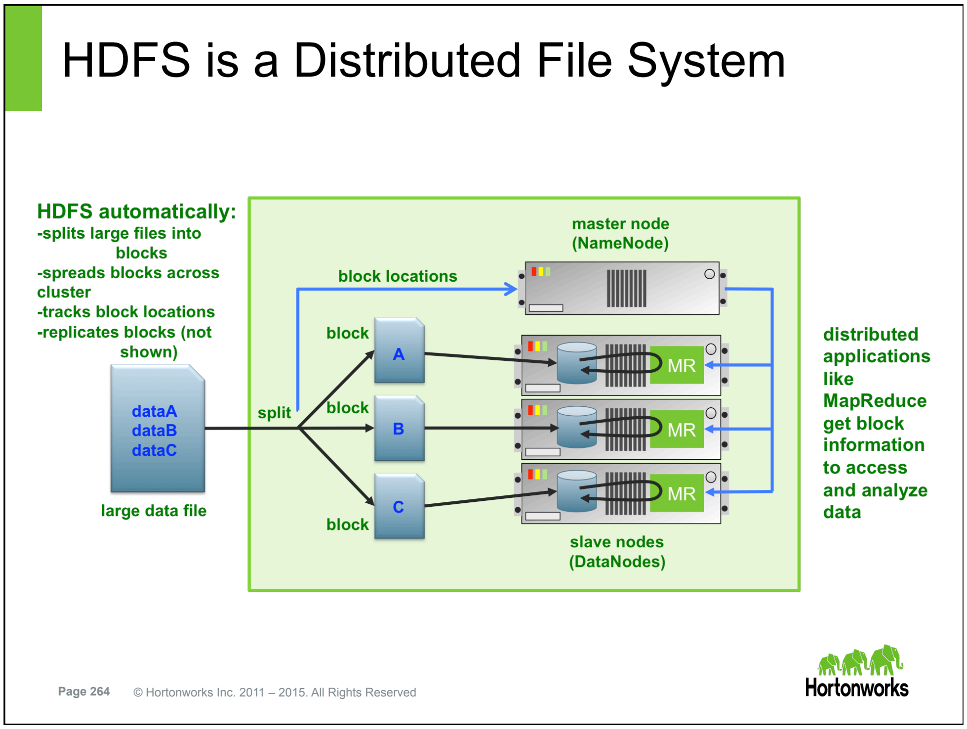
- Read, write, and delete files
- Create, list, and delete directories
- Manage file and directory ownerships and permissions

Optimal I/O performance is achieved using sequential reads

- Files may be appended but existing data is not changed
- HDFS was designed with a write-once read-many ideology
  - Enables creation of an immutable data lake supporting simultaneous access and analysis by multiple applications

Page 263

Hortonworks



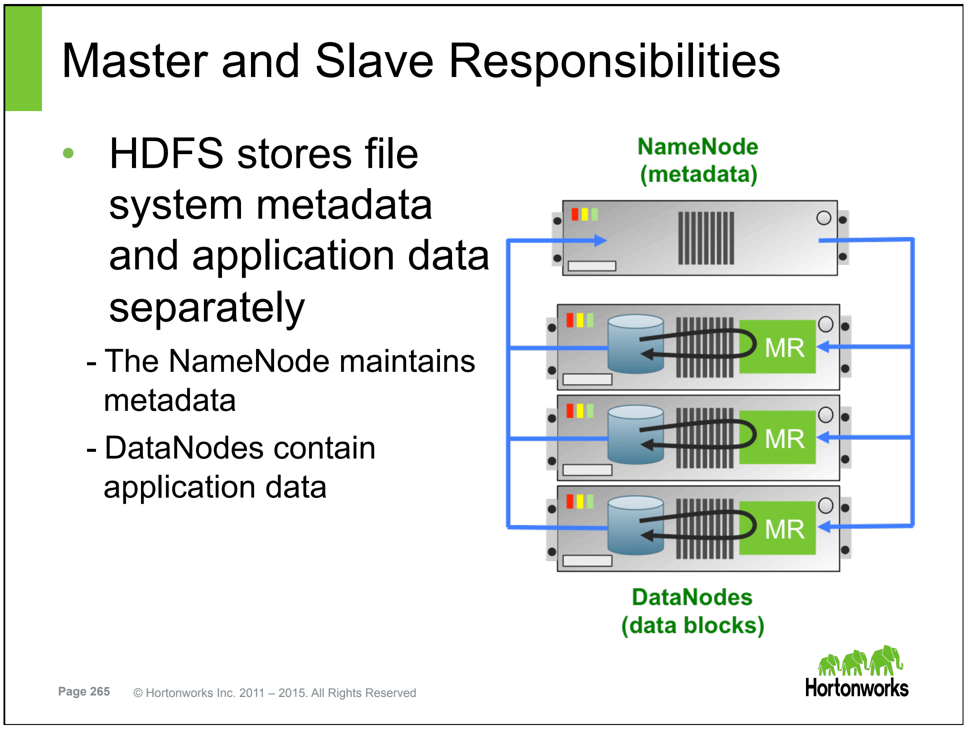
HDFS is a Distributed File System

HDFS automatically:
-splits large files into blocks
-spreads blocks across cluster
-tracks block locations
-replicates blocks (not shown)

distributed applications like MapReduce get block information to access and analyze data

Page 264

Hortonworks



Master and Slave Responsibilities

- HDFS stores file system metadata and application data separately
  - The NameNode maintains metadata
  - DataNodes contain application data

NameNode (metadata)

MR

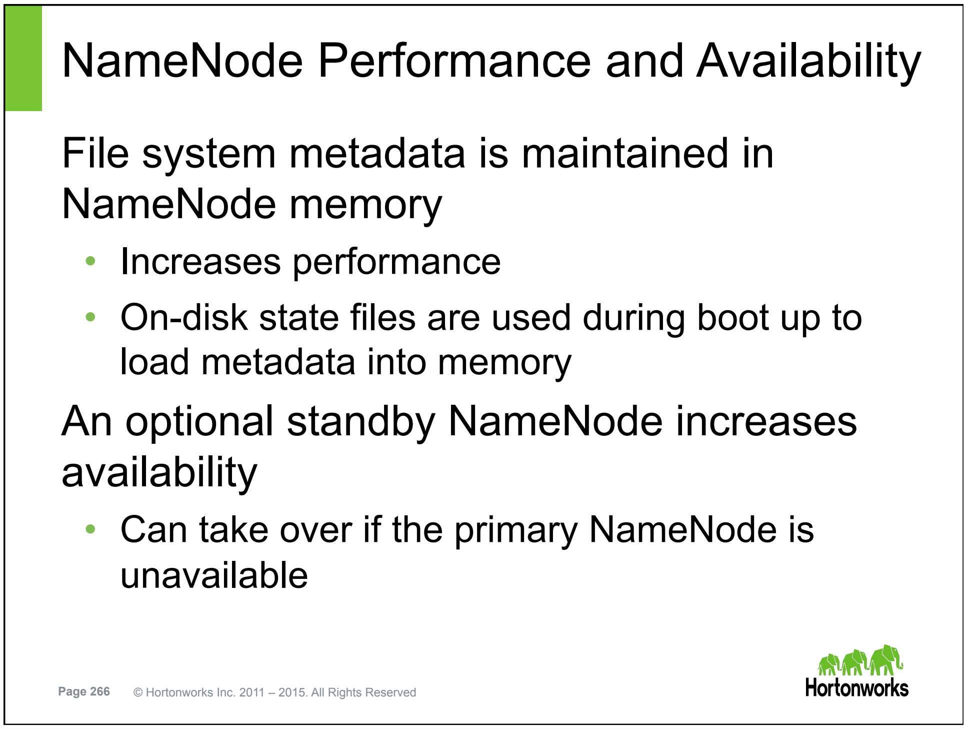
MR

MR

DataNodes (data blocks)

Page 265

Hortonworks



NameNode Performance and Availability

File system metadata is maintained in NameNode memory

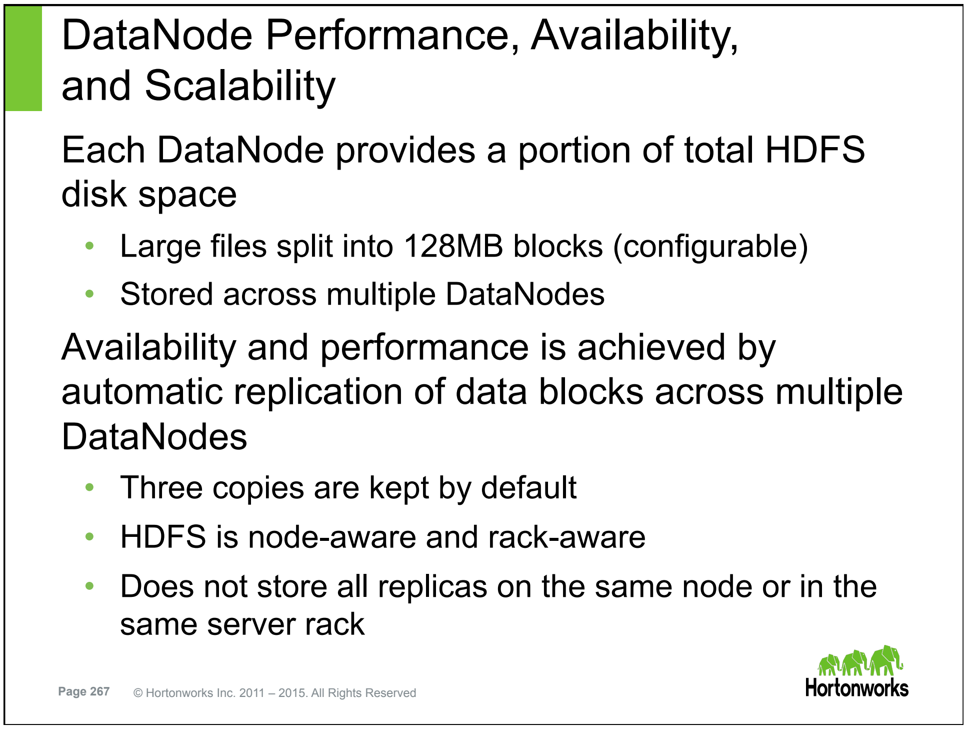
- Increases performance
- On-disk state files are used during boot up to load metadata into memory

An optional standby NameNode increases availability

- Can take over if the primary NameNode is unavailable

Page 266

Hortonworks



DataNode Performance, Availability, and Scalability

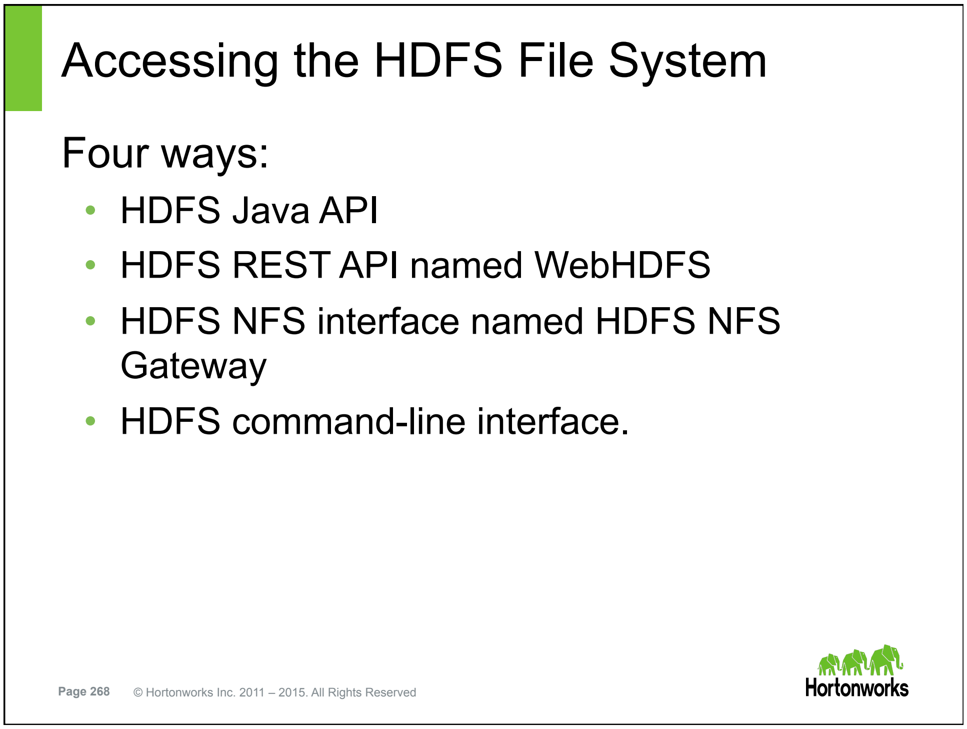
Each DataNode provides a portion of total HDFS disk space

- Large files split into 128MB blocks (configurable)
- Stored across multiple DataNodes

Availability and performance is achieved by automatic replication of data blocks across multiple DataNodes

- Three copies are kept by default
- HDFS is node-aware and rack-aware
- Does not store all replicas on the same node or in the same server rack

Page 267

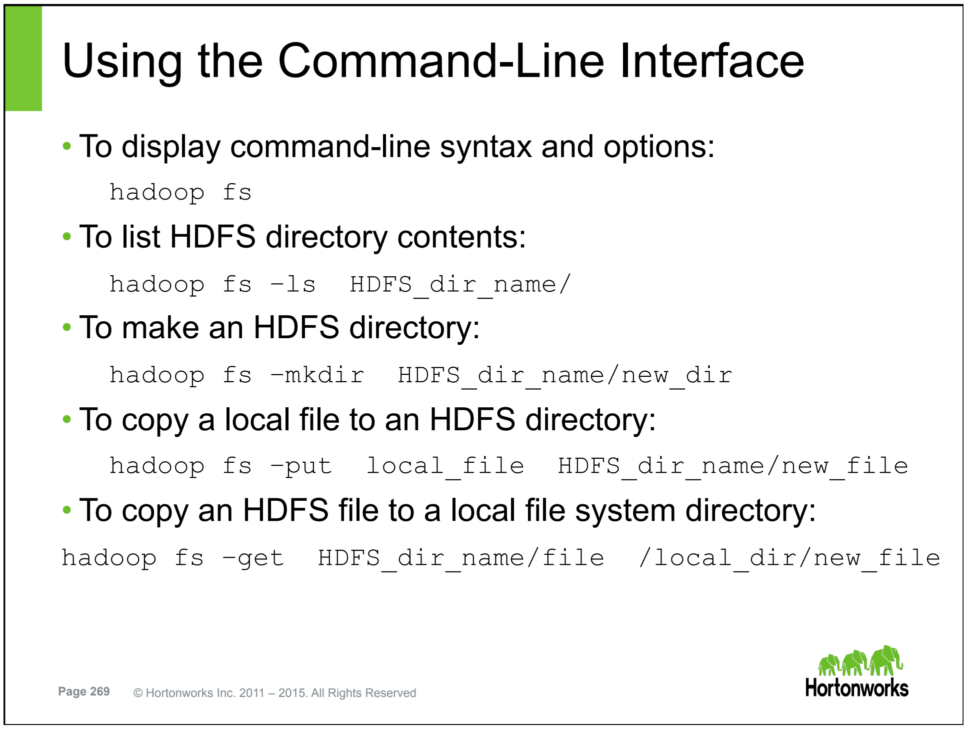


Accessing the HDFS File System

Four ways:

- HDFS Java API
- HDFS REST API named WebHDFS
- HDFS NFS interface named HDFS NFS Gateway
- HDFS command-line interface.

Page 268



Using the Command-Line Interface

- To display command-line syntax and options:

```
hadoop fs
```

- To list HDFS directory contents:

```
hadoop fs -ls  HDFS_dir_name/
```

- To make an HDFS directory:

```
hadoop fs -mkdir  HDFS_dir_name/new_dir
```

- To copy a local file to an HDFS directory:

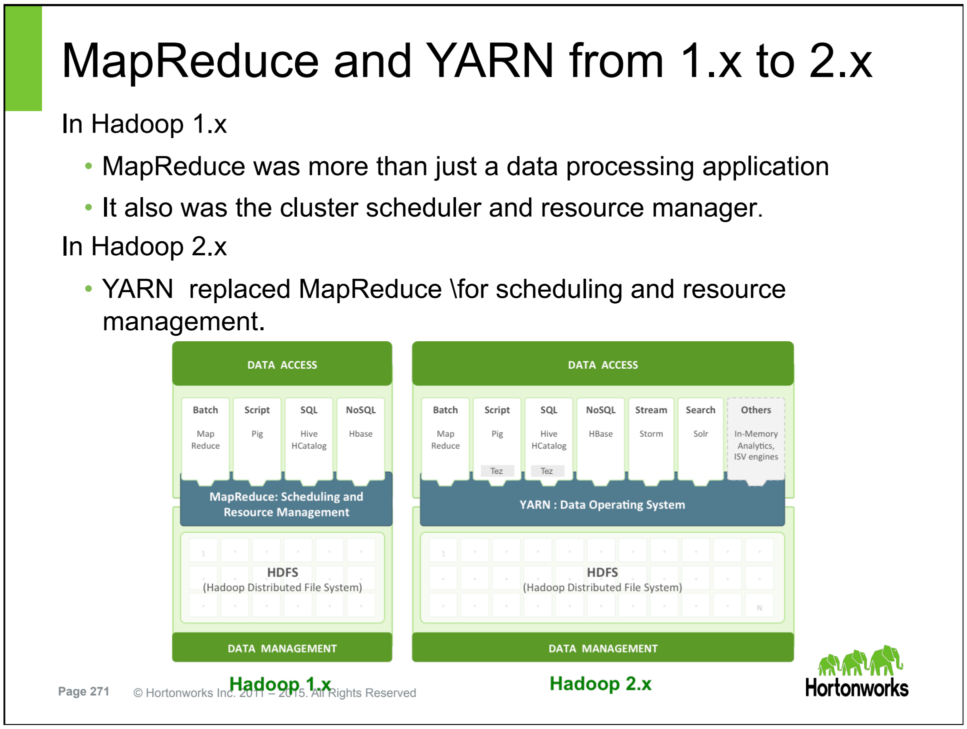
```
hadoop fs -put  local_file  HDFS_dir_name/new_file
```

- To copy an HDFS file to a local file system directory:

```
hadoop fs -get  HDFS_dir_name/file  /local_dir/new_file
```

Page 269

YARN



MapReduce and YARN from 1.x to 2.x

In Hadoop 1.x

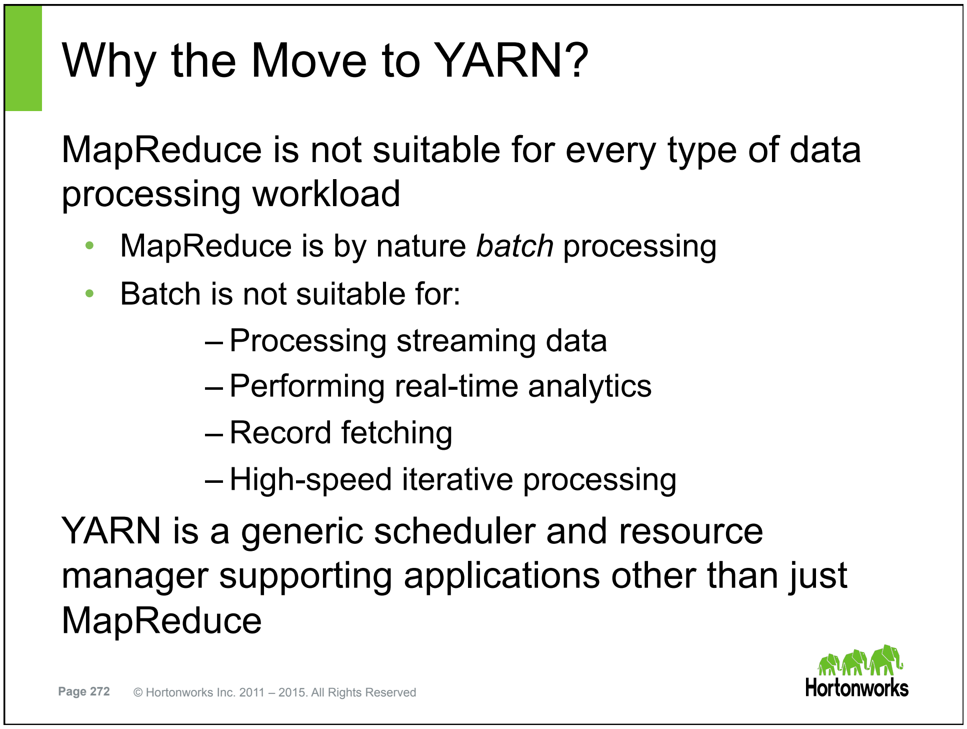
- MapReduce was more than just a data processing application
- It also was the cluster scheduler and resource manager.

In Hadoop 2.x

- YARN replaced MapReduce \for scheduling and resource management.

Page 271 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Why the Move to YARN?

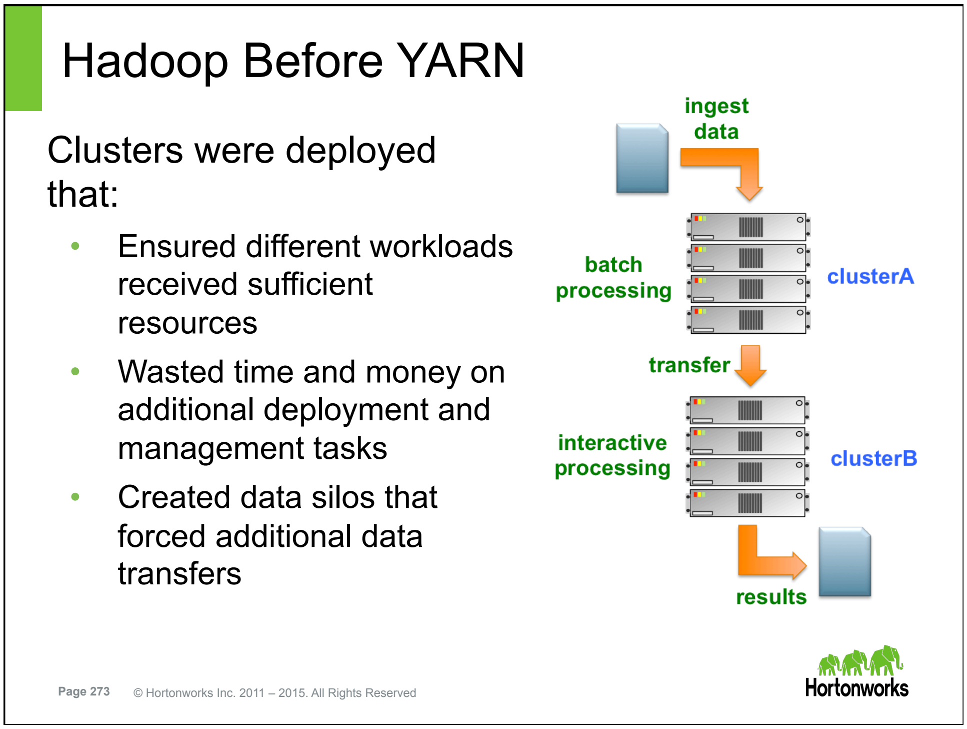
MapReduce is not suitable for every type of data processing workload

- MapReduce is by nature *batch* processing
- Batch is not suitable for:
  - Processing streaming data
  - Performing real-time analytics
  - Record fetching
  - High-speed iterative processing

YARN is a generic scheduler and resource manager supporting applications other than just MapReduce

Page 272 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



Hadoop Before YARN

Clusters were deployed that:

- Ensured different workloads received sufficient resources
- Wasted time and money on additional deployment and management tasks
- Created data silos that forced additional data transfers

ingest data

batch processing

clusterA

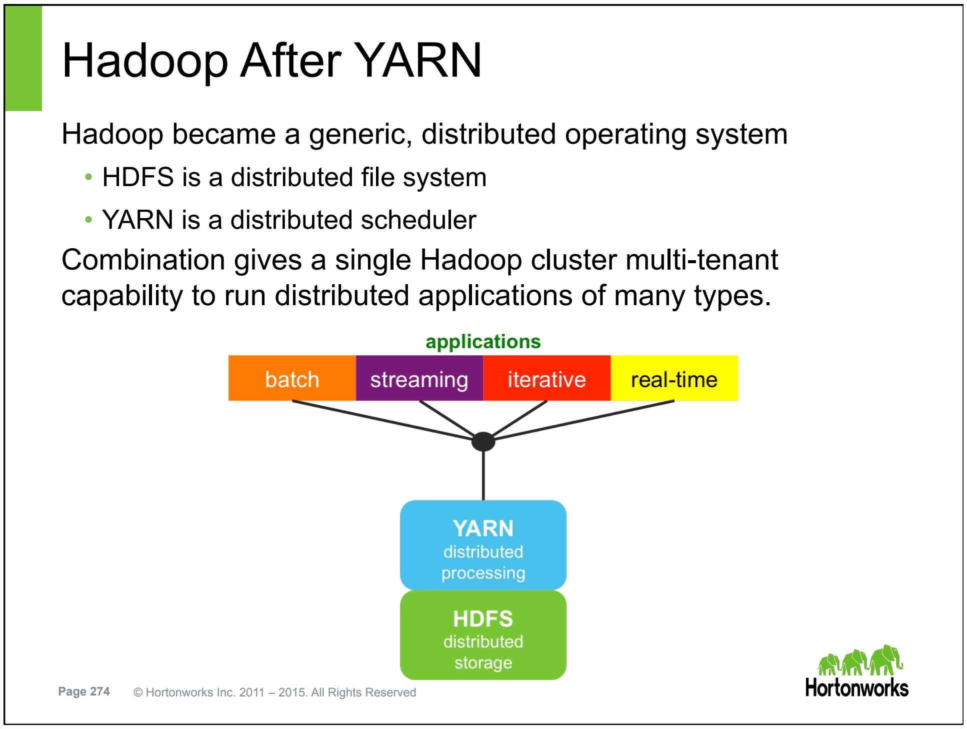
transfer

interactive processing

clusterB

results

Page 273 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Hadoop After YARN

Hadoop became a generic, distributed operating system

- HDFS is a distributed file system
- YARN is a distributed scheduler

Combination gives a single Hadoop cluster multi-tenant capability to run distributed applications of many types.

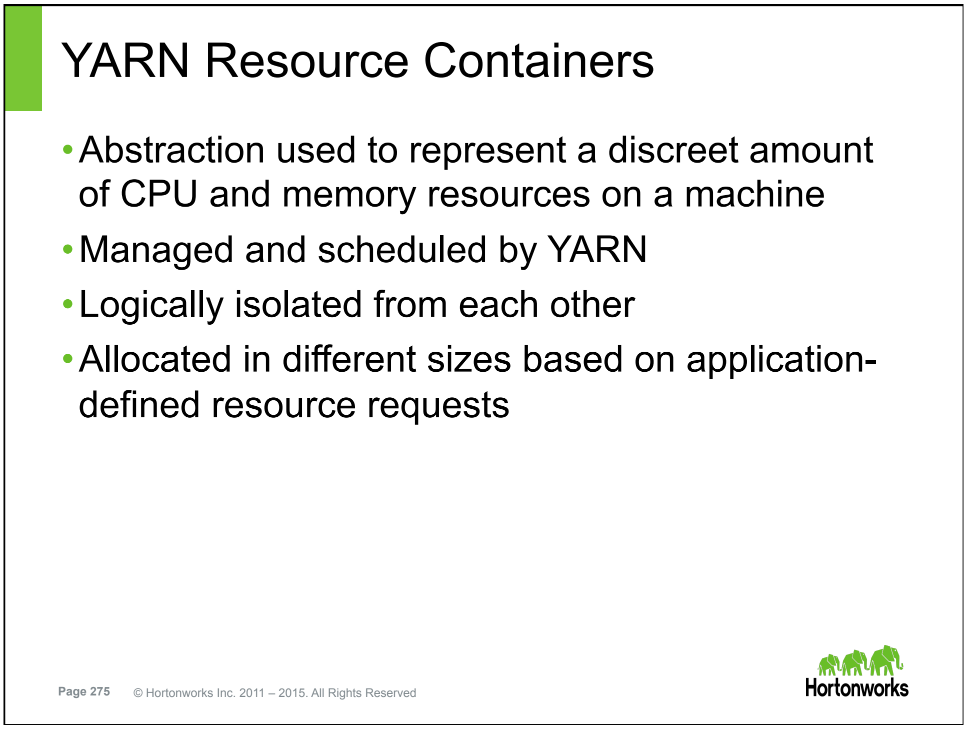
applications

batch | streaming | iterative | real-time

YARN distributed processing

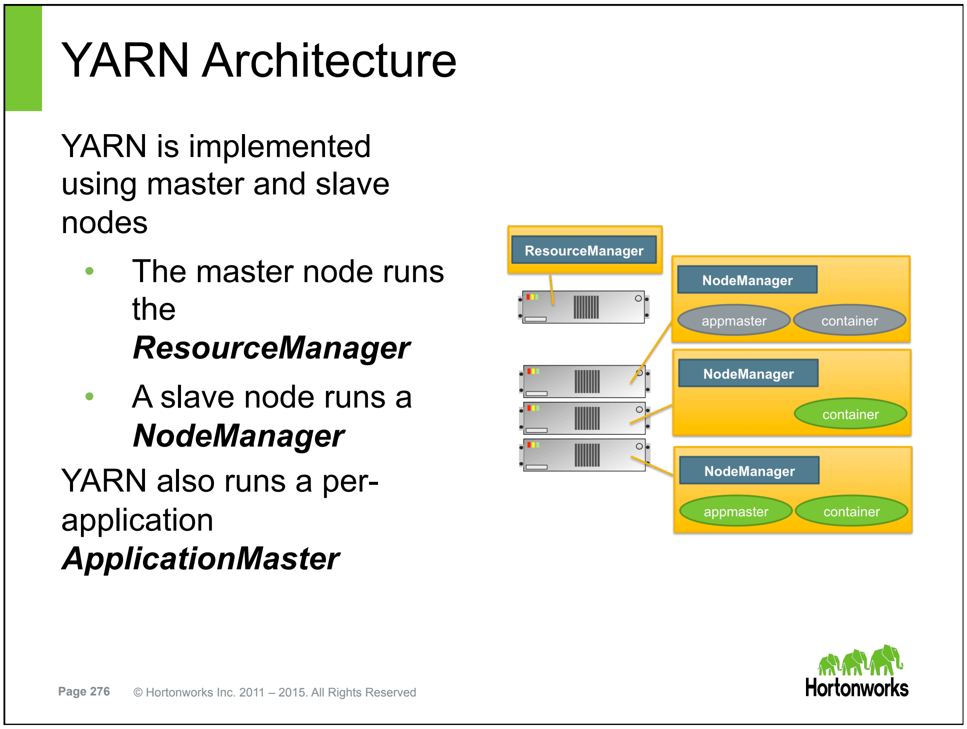
HDFS distributed storage

Page 274 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



YARN Resource Containers

- Abstraction used to represent a discreet amount of CPU and memory resources on a machine
- Managed and scheduled by YARN
- Logically isolated from each other
- Allocated in different sizes based on application-defined resource requests



YARN Architecture

YARN is implemented using master and slave nodes

- The master node runs the ***ResourceManager***
- A slave node runs a ***NodeManager***

YARN also runs a per-application ***ApplicationMaster***

ResourceManager

NodeManager

appmaster

container

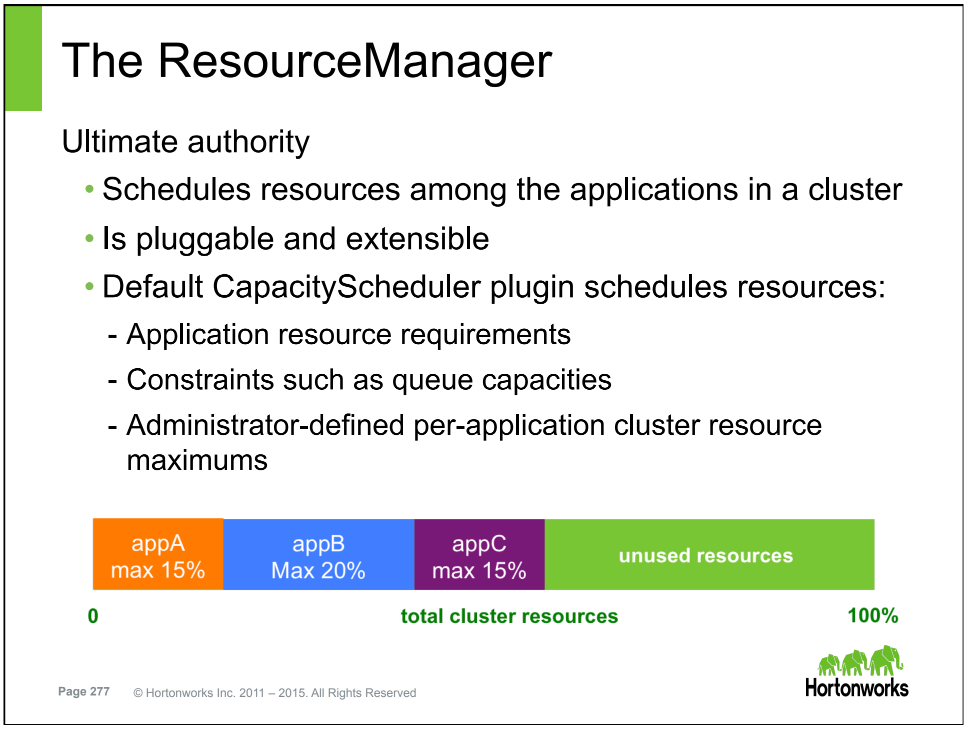
NodeManager

container

NodeManager

appmaster

container



The ResourceManager

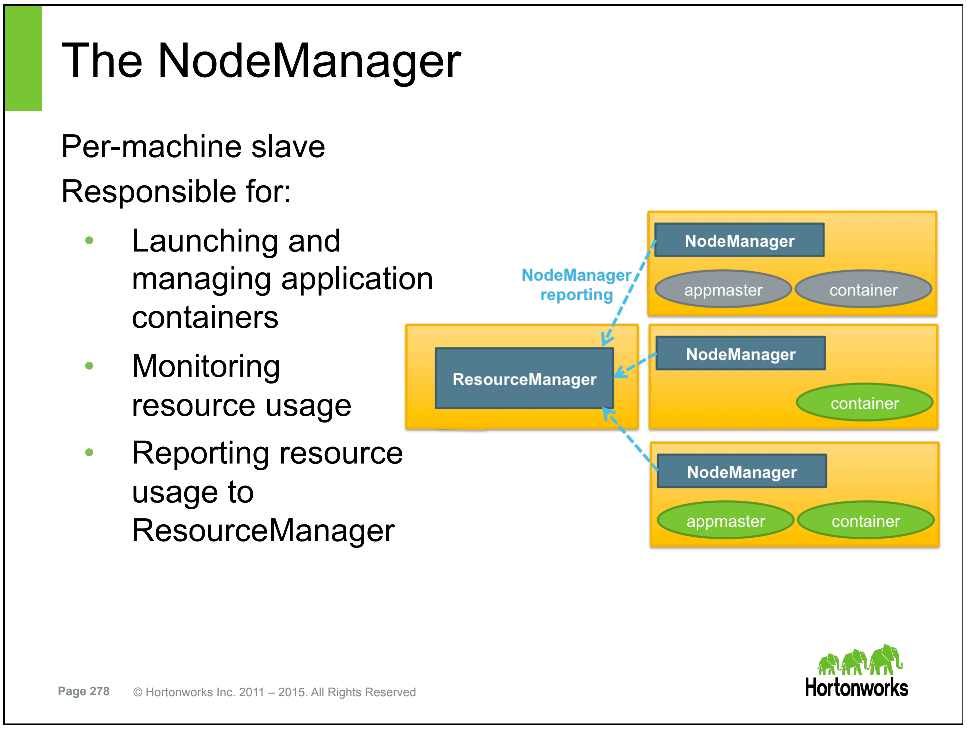
Ultimate authority

- Schedules resources among the applications in a cluster
- Is pluggable and extensible
- Default CapacityScheduler plugin schedules resources:
  - Application resource requirements
  - Constraints such as queue capacities
  - Administrator-defined per-application cluster resource maximums

appA
max 15%
appB
Max 20%
appC
max 15%
unused resources
0
total cluster resources
100%

Page 277 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks



The NodeManager

Per-machine slave

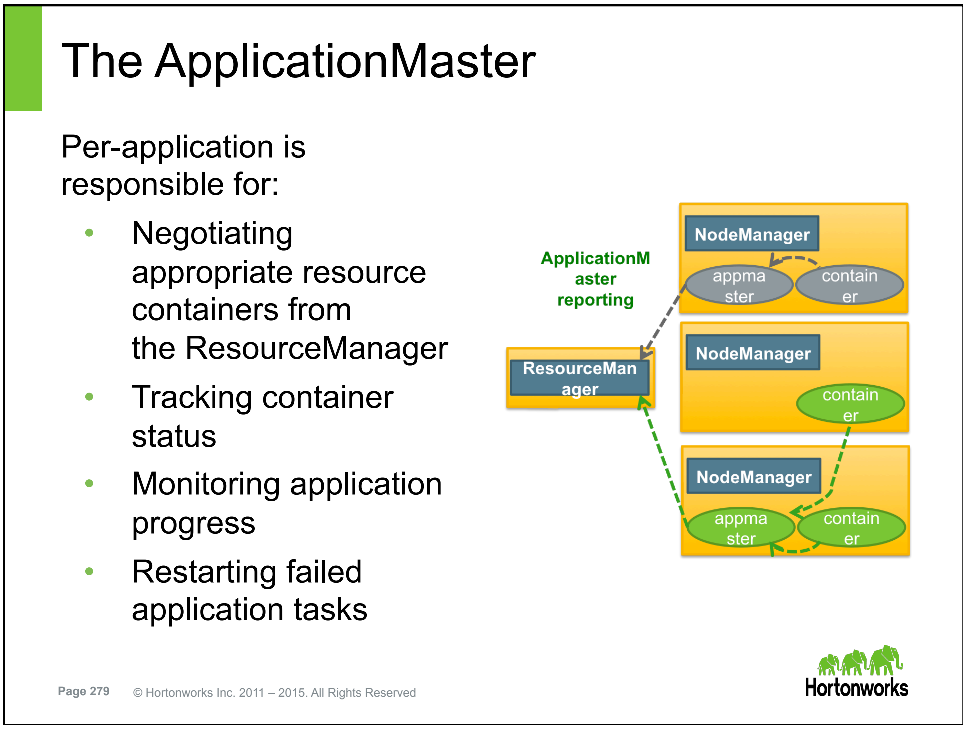
Responsible for:

- Launching and managing application containers
- Monitoring resource usage
- Reporting resource usage to ResourceManager

NodeManager
appmaster
container
NodeManager reporting
ResourceManager
NodeManager
container
NodeManager
appmaster
container

Page 278 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

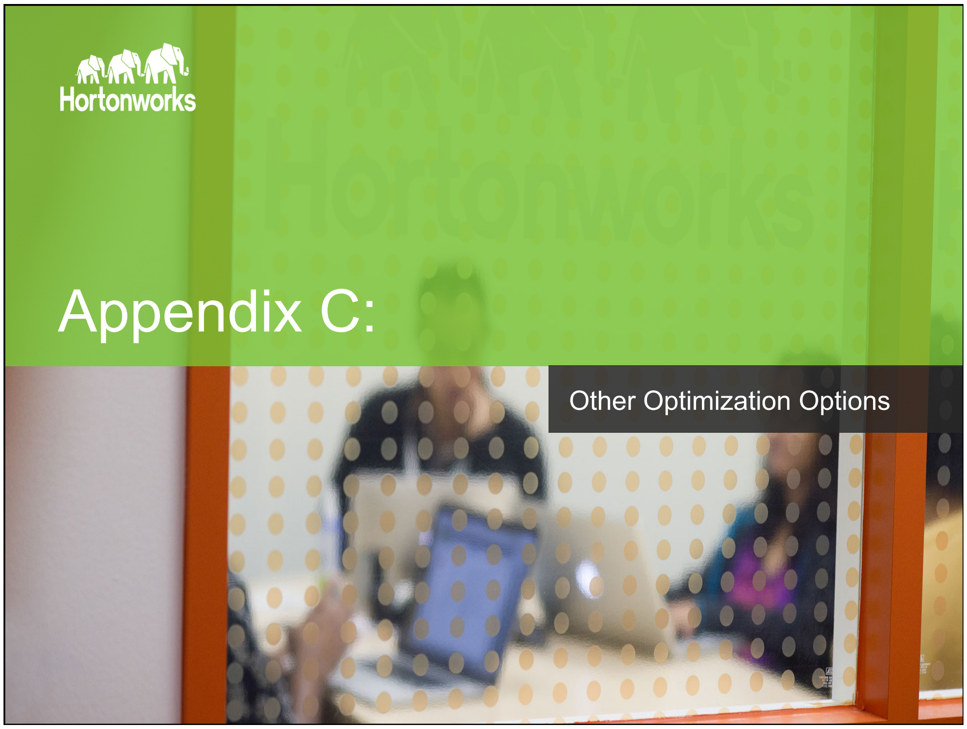


The ApplicationMaster

Per-application is responsible for:

- Negotiating appropriate resource containers from the ResourceManager
- Tracking container status
- Monitoring application progress
- Restarting failed application tasks

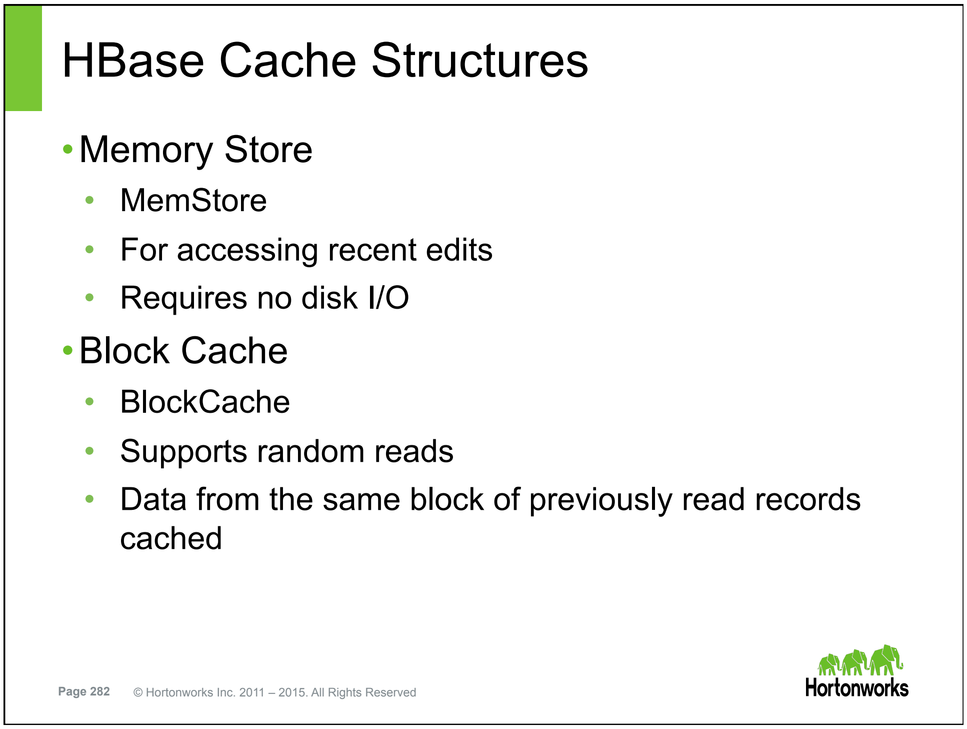
Page 279 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Appendix C:

Other Optimization Options

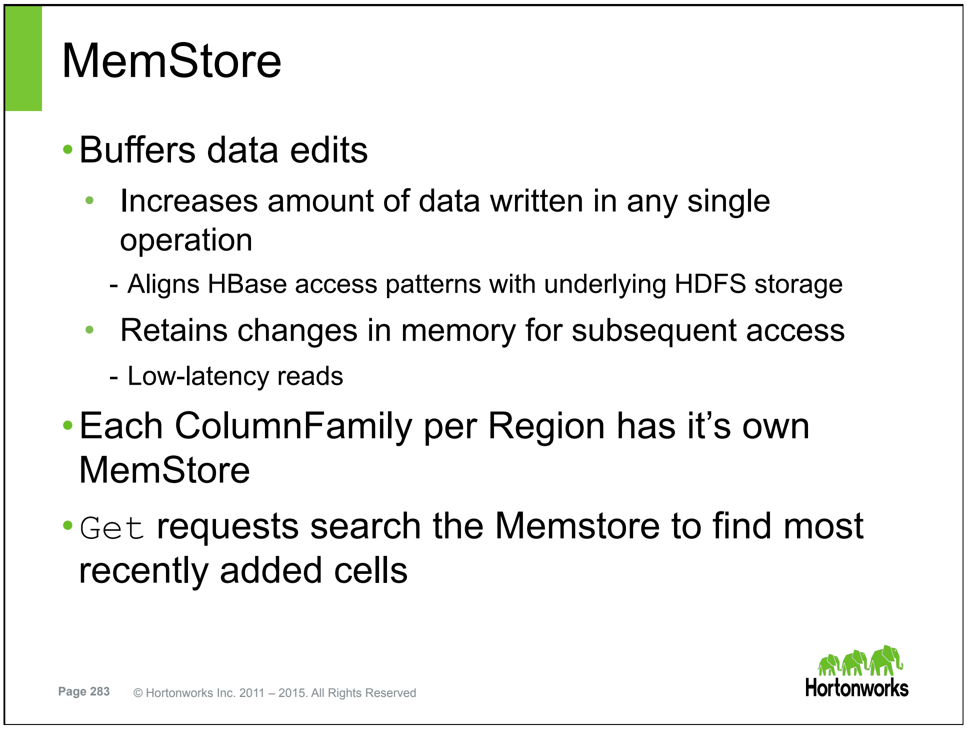
Adjusting Cache



HBase Cache Structures

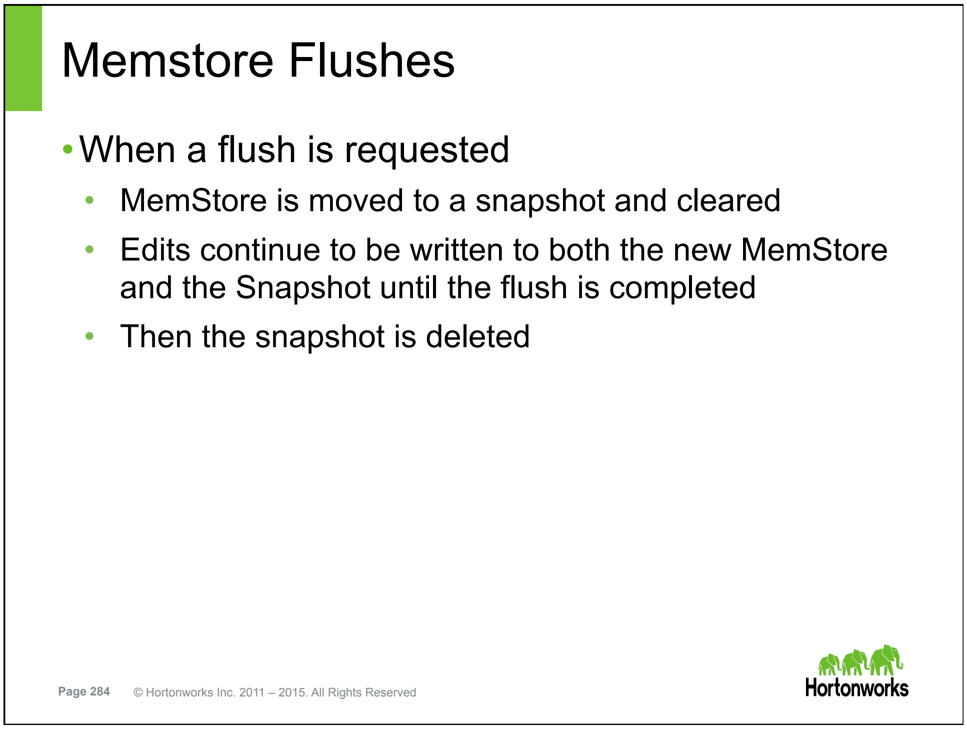
- Memory Store
  - MemStore
  - For accessing recent edits
  - Requires no disk I/O
- Block Cache
  - BlockCache
  - Supports random reads
  - Data from the same block of previously read records cached

Page 282



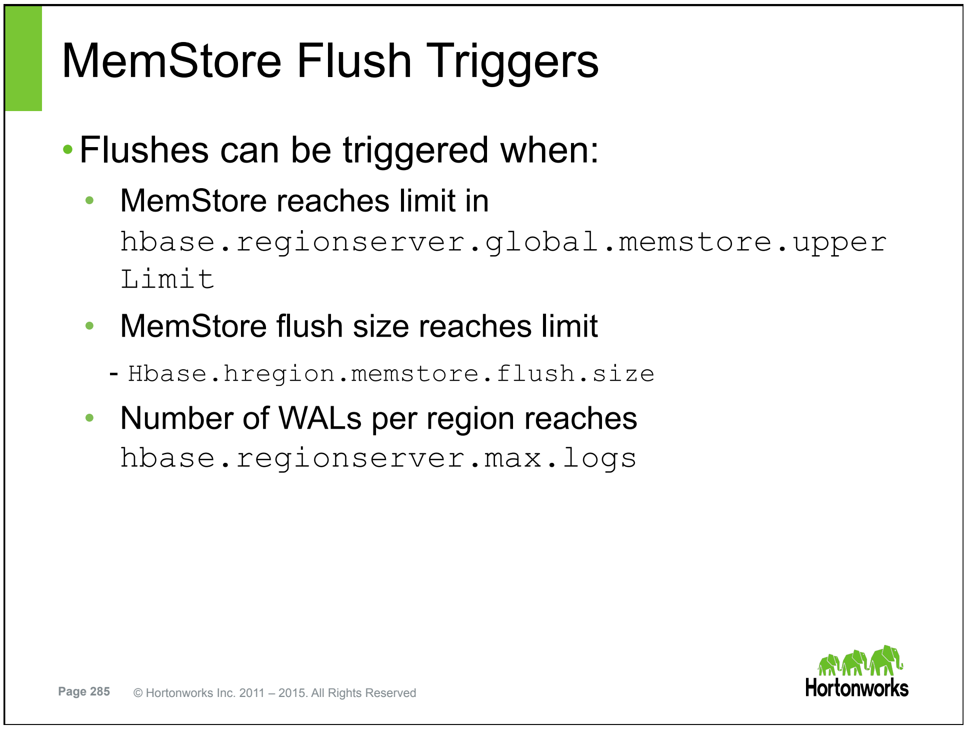
MemStore

- Buffers data edits
  - Increases amount of data written in any single operation
    - Aligns HBase access patterns with underlying HDFS storage
  - Retains changes in memory for subsequent access
    - Low-latency reads
- Each ColumnFamily per Region has it's own MemStore
- `Get` requests search the Memstore to find most recently added cells



Memstore Flushes

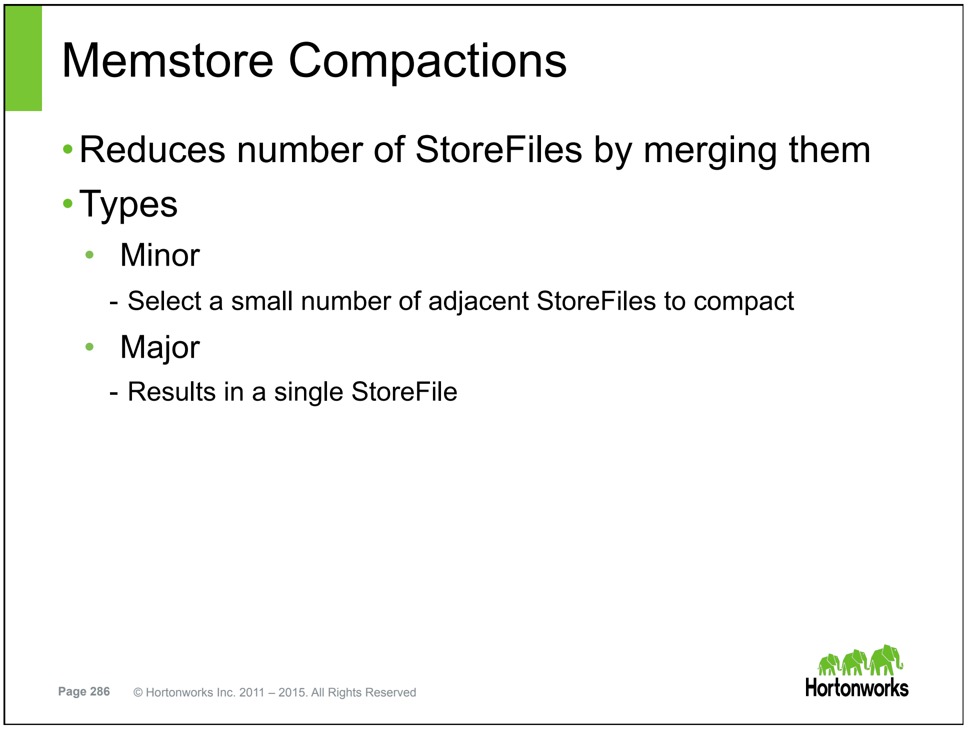
- When a flush is requested
  - MemStore is moved to a snapshot and cleared
  - Edits continue to be written to both the new MemStore and the Snapshot until the flush is completed
  - Then the snapshot is deleted



MemStore Flush Triggers

- Flushes can be triggered when:
  - MemStore reaches limit in `hbase.regionserver.global.memstore.upperLimit`
  - MemStore flush size reaches limit
    - `Hbase.hregion.memstore.flush.size`
  - Number of WALs per region reaches `hbase.regionserver.max.logs`

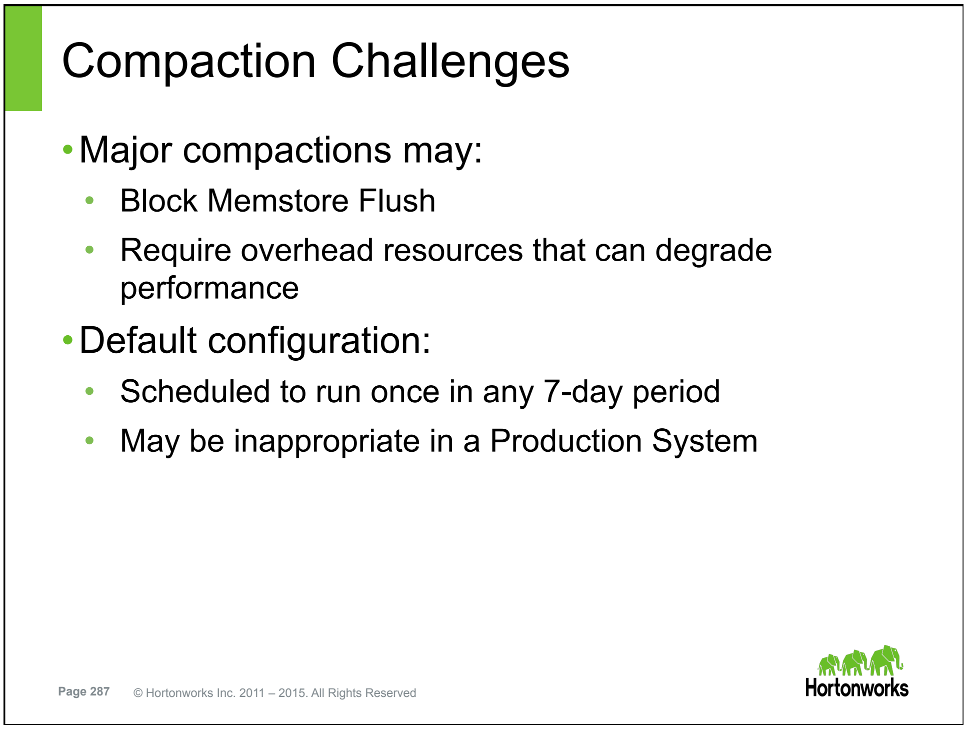
Page 285



Memstore Compactions

- Reduces number of StoreFiles by merging them
- Types
  - Minor
    - Select a small number of adjacent StoreFiles to compact
  - Major
    - Results in a single StoreFile

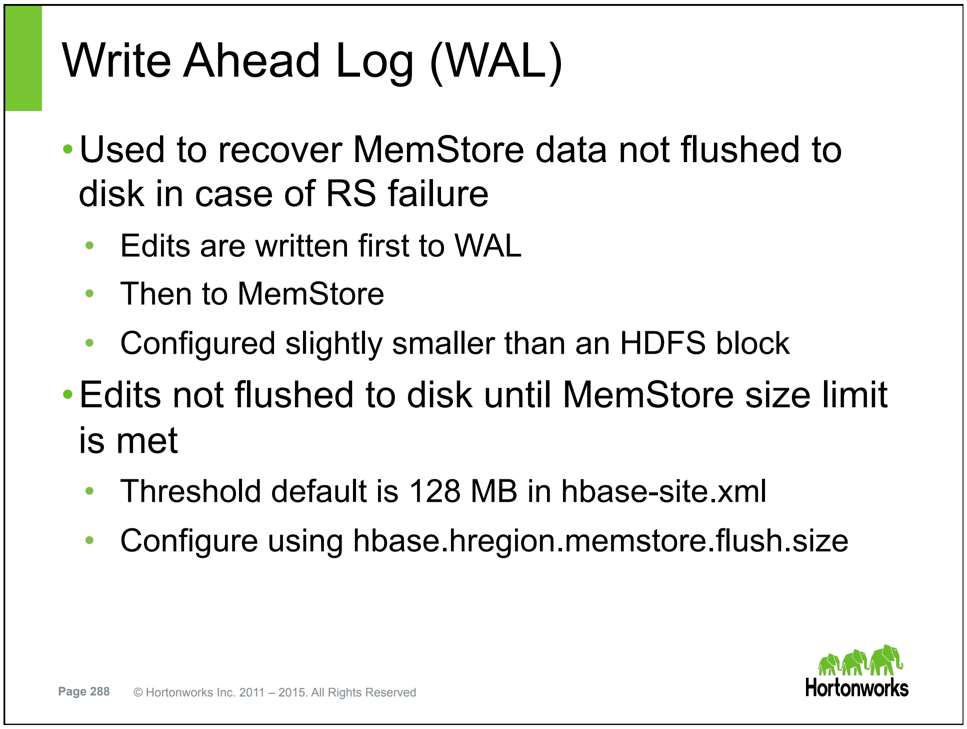
Page 286



Compaction Challenges

- Major compactions may:
  - Block Memstore Flush
  - Require overhead resources that can degrade performance
- Default configuration:
  - Scheduled to run once in any 7-day period
  - May be inappropriate in a Production System

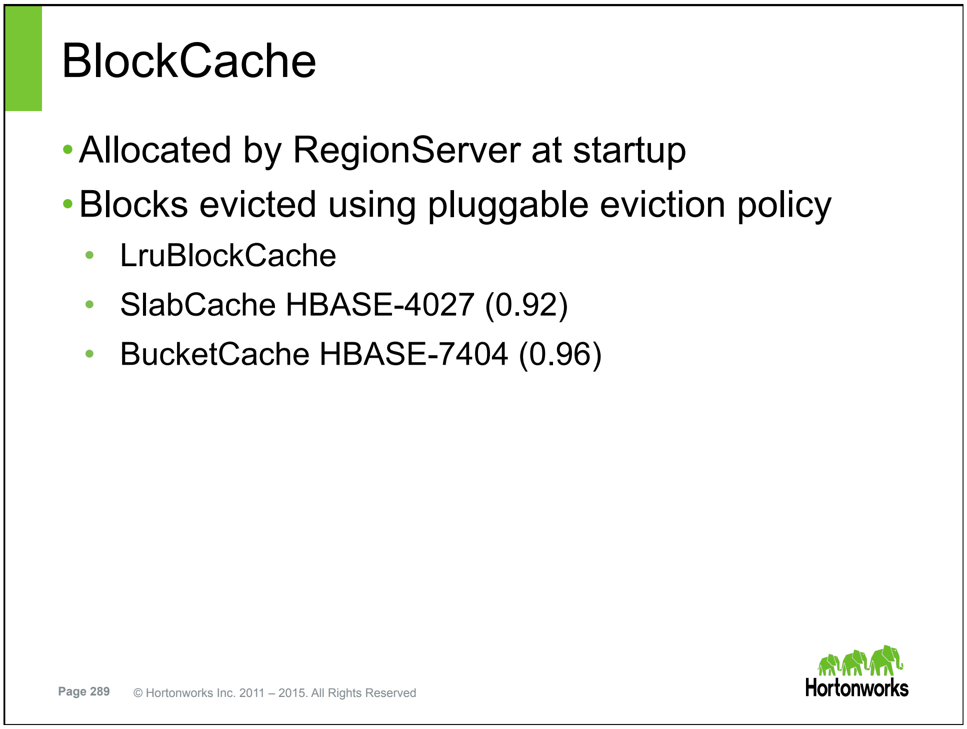
Page 287 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Write Ahead Log (WAL)

- Used to recover MemStore data not flushed to disk in case of RS failure
  - Edits are written first to WAL
  - Then to MemStore
  - Configured slightly smaller than an HDFS block
- Edits not flushed to disk until MemStore size limit is met
  - Threshold default is 128 MB in hbase-site.xml
  - Configure using hbase.hregion.memstore.flush.size

Page 288 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

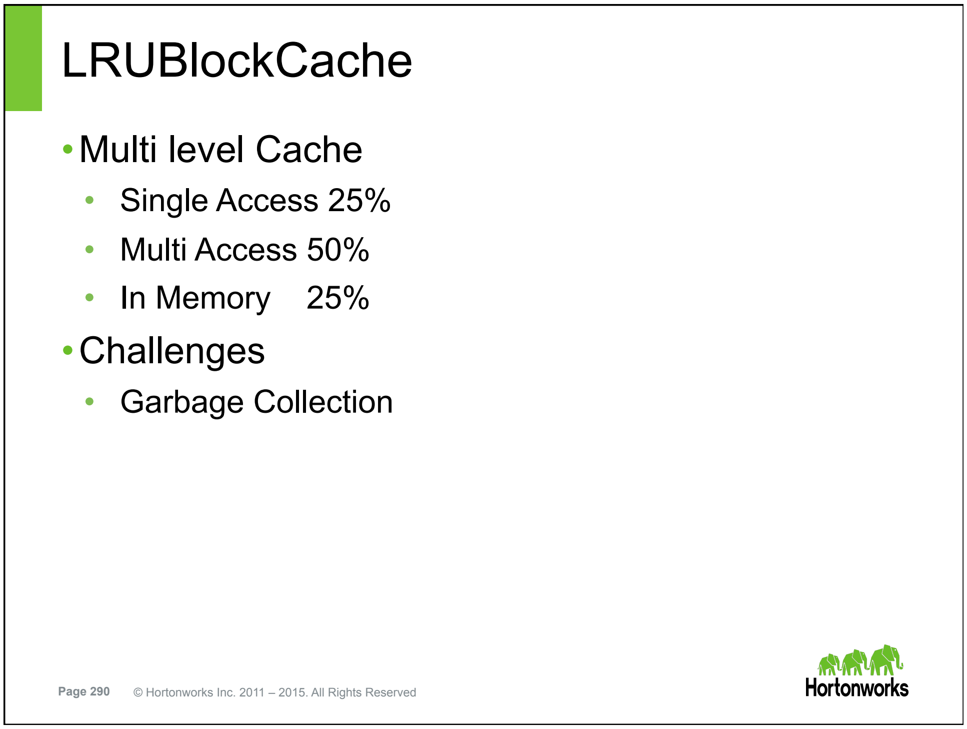


BlockCache

- Allocated by RegionServer at startup
- Blocks evicted using pluggable eviction policy
  - LruBlockCache
  - SlabCache HBASE-4027 (0.92)
  - BucketCache HBASE-7404 (0.96)

Page 289

Hortonworks

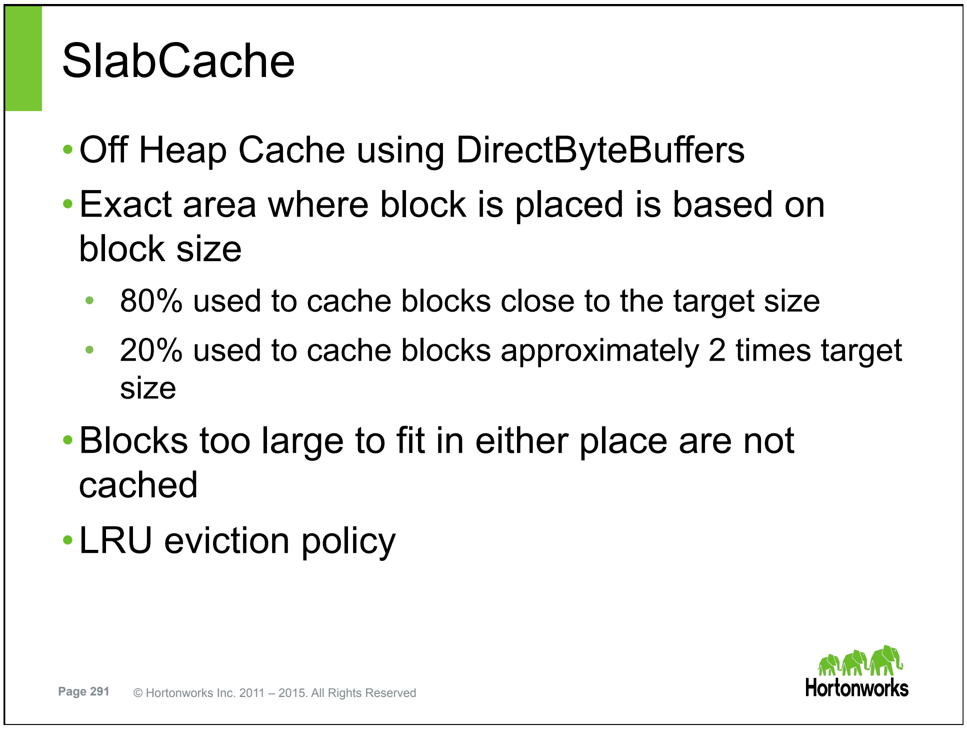


LRUBlockCache

- Multi level Cache
  - Single Access 25%
  - Multi Access 50%
  - In Memory 25%
- Challenges
  - Garbage Collection

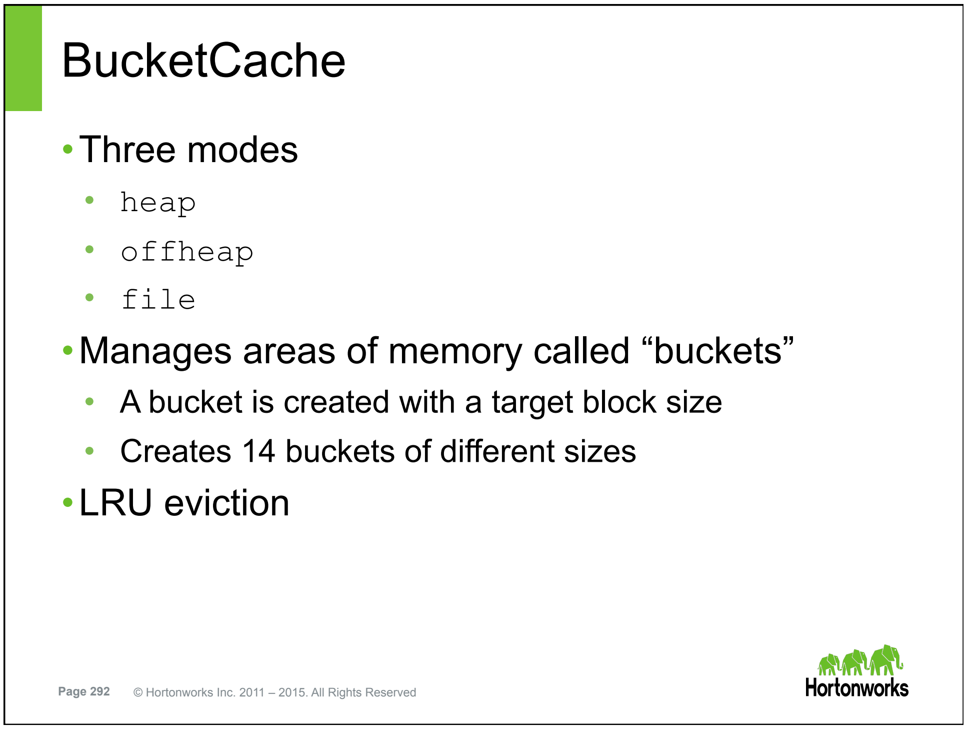
Page 290

Hortonworks



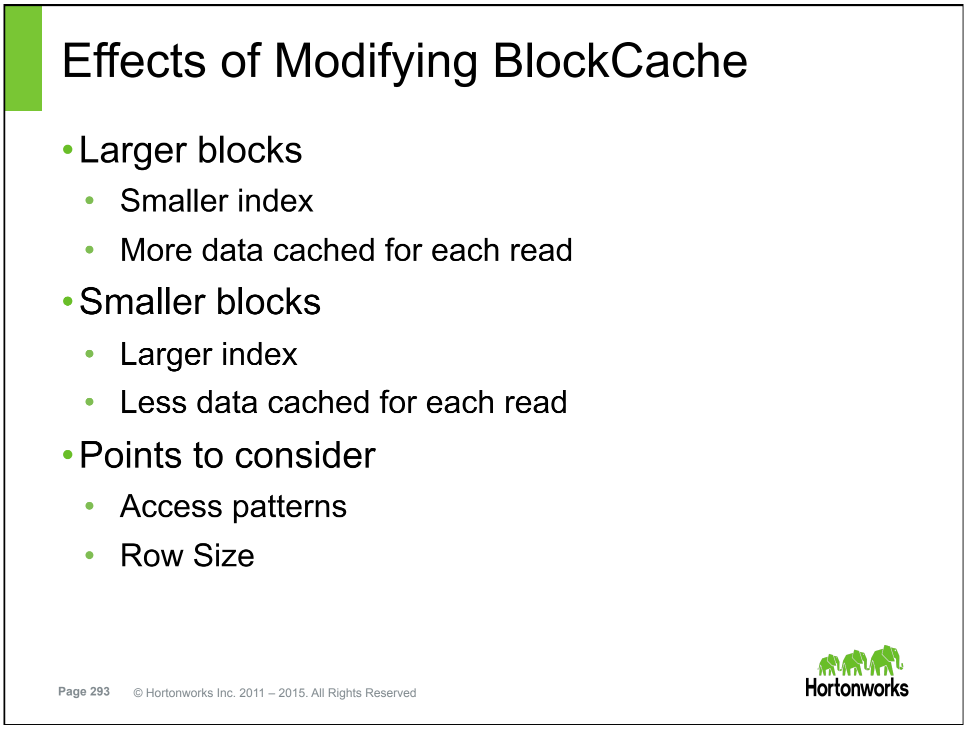
SlabCache

- Off Heap Cache using DirectByteBuffers
- Exact area where block is placed is based on block size
  - 80% used to cache blocks close to the target size
  - 20% used to cache blocks approximately 2 times target size
- Blocks too large to fit in either place are not cached
- LRU eviction policy



BucketCache

- Three modes
  - `heap`
  - `offheap`
  - `file`
- Manages areas of memory called “buckets”
  - A bucket is created with a target block size
  - Creates 14 buckets of different sizes
- LRU eviction

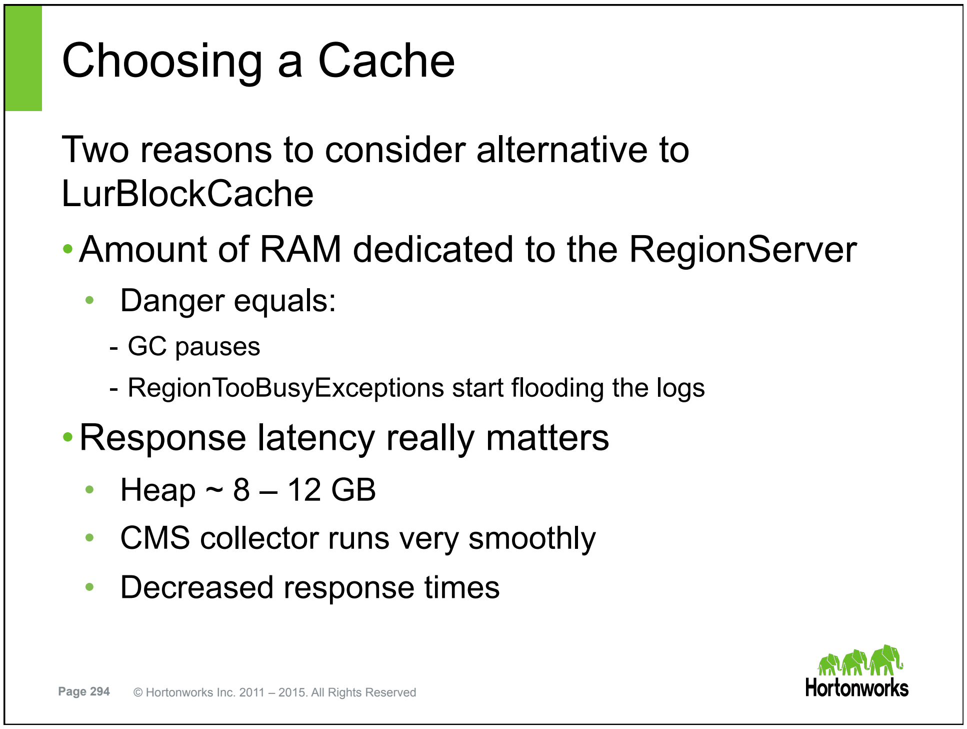


Effects of Modifying BlockCache

- Larger blocks
  - Smaller index
  - More data cached for each read
- Smaller blocks
  - Larger index
  - Less data cached for each read
- Points to consider
  - Access patterns
  - Row Size

Page 293

Hortonworks



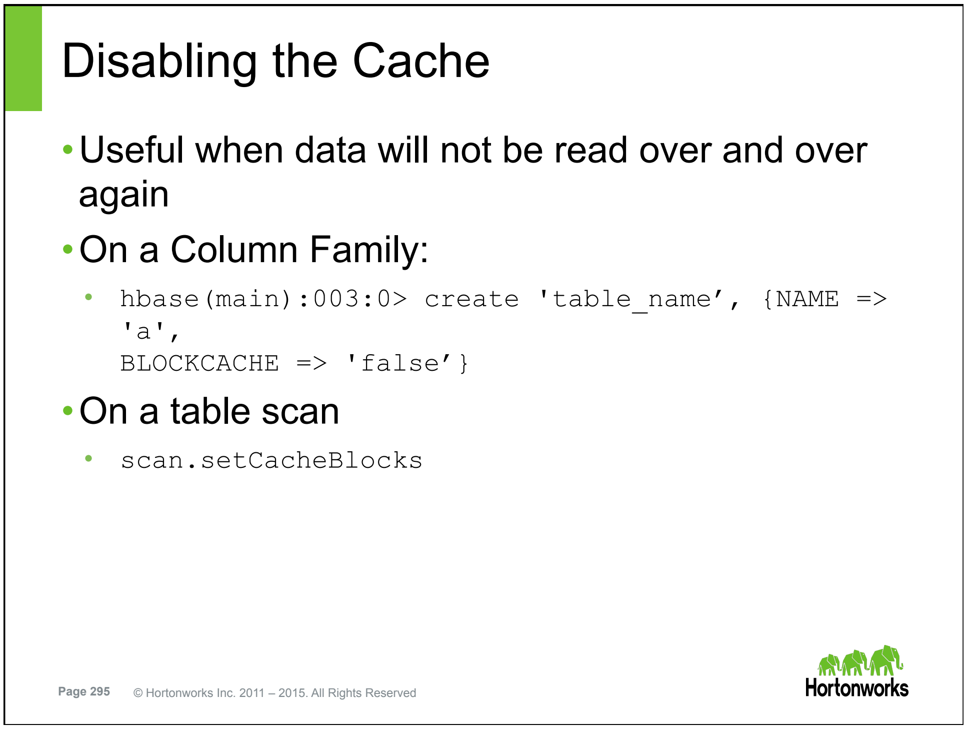
Choosing a Cache

Two reasons to consider alternative to LurBlockCache

- Amount of RAM dedicated to the RegionServer
  - Danger equals:
    - GC pauses
    - RegionTooBusyExceptions start flooding the logs
- Response latency really matters
  - Heap ~ 8 – 12 GB
  - CMS collector runs very smoothly
  - Decreased response times

Page 294

Hortonworks

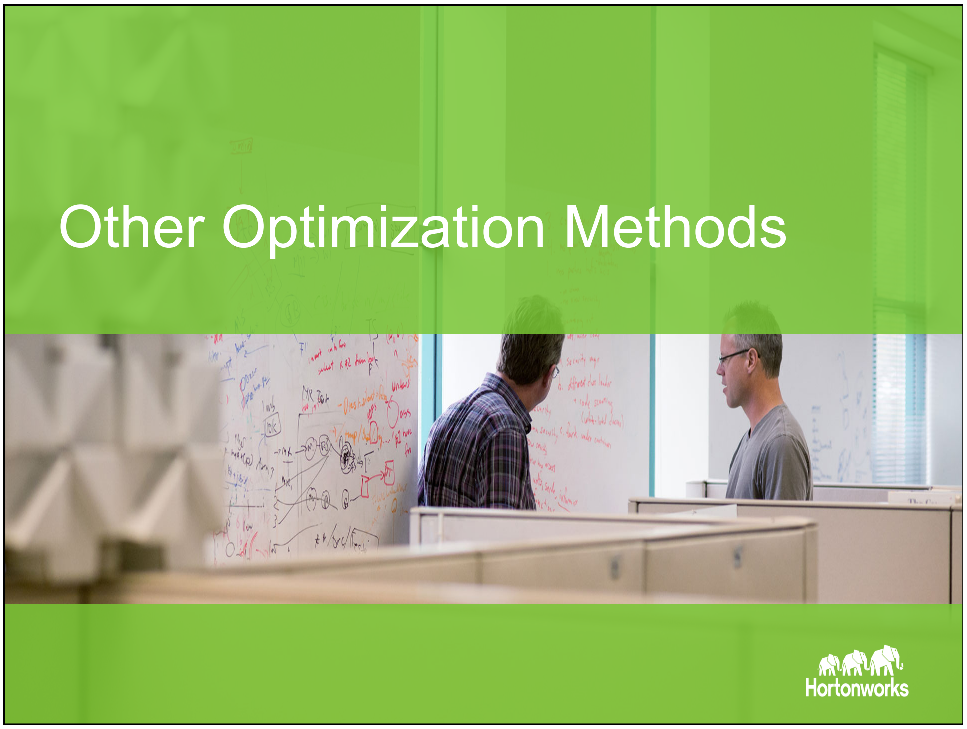


Disabling the Cache

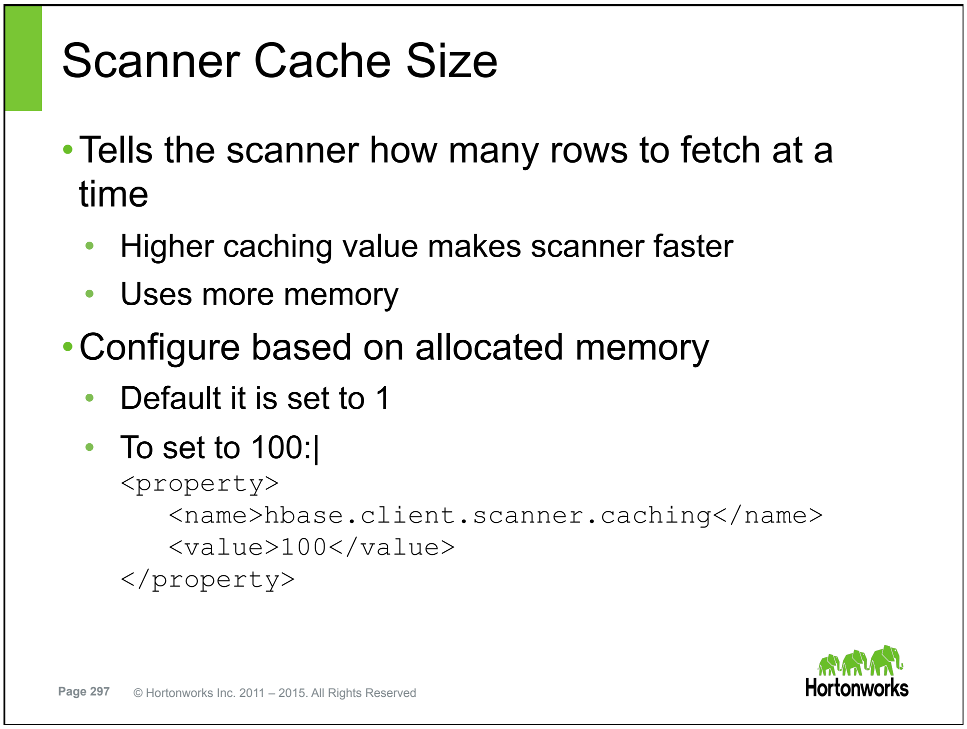
- Useful when data will not be read over and over again
- On a Column Family:

  -
    ```
    hbase(main):003:0> create 'table_name’, {NAME =>
    'a',
    BLOCKCACHE => 'false’}
    ```
- On a table scan
  - `scan.setCacheBlocks`

Page 295 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



Other Optimization Methods



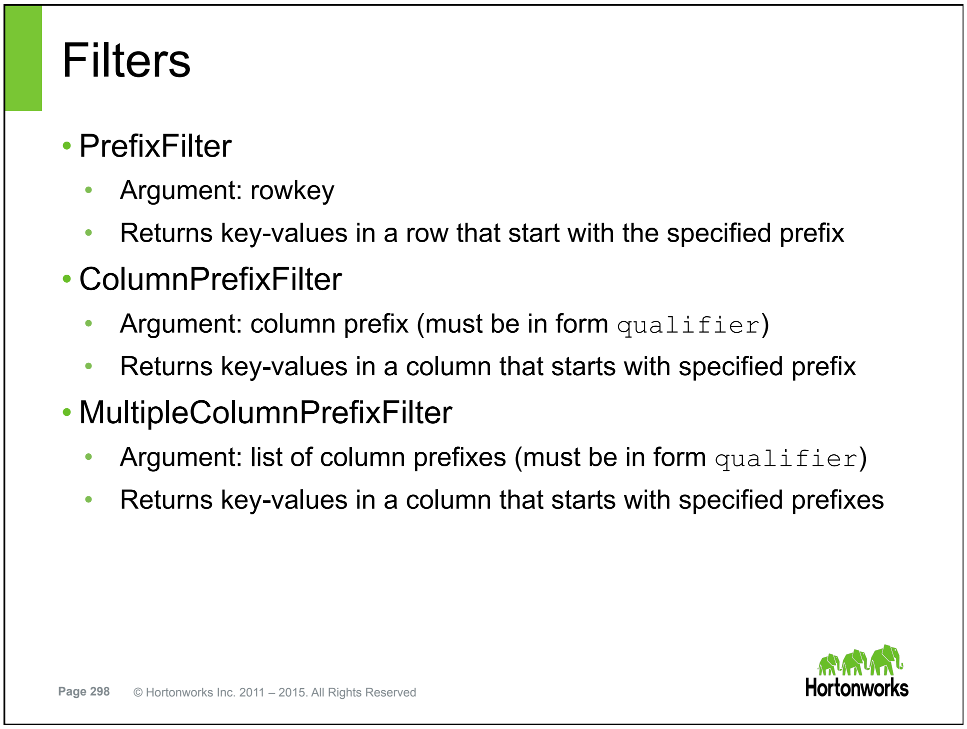
Scanner Cache Size

- Tells the scanner how many rows to fetch at a time
  - Higher caching value makes scanner faster
  - Uses more memory
- Configure based on allocated memory
  - Default it is set to 1
  - To set to 100:|

```
<property>
   <name>hbase.client.scanner.caching</name>
   <value>100</value>
</property>
```

Page 297 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

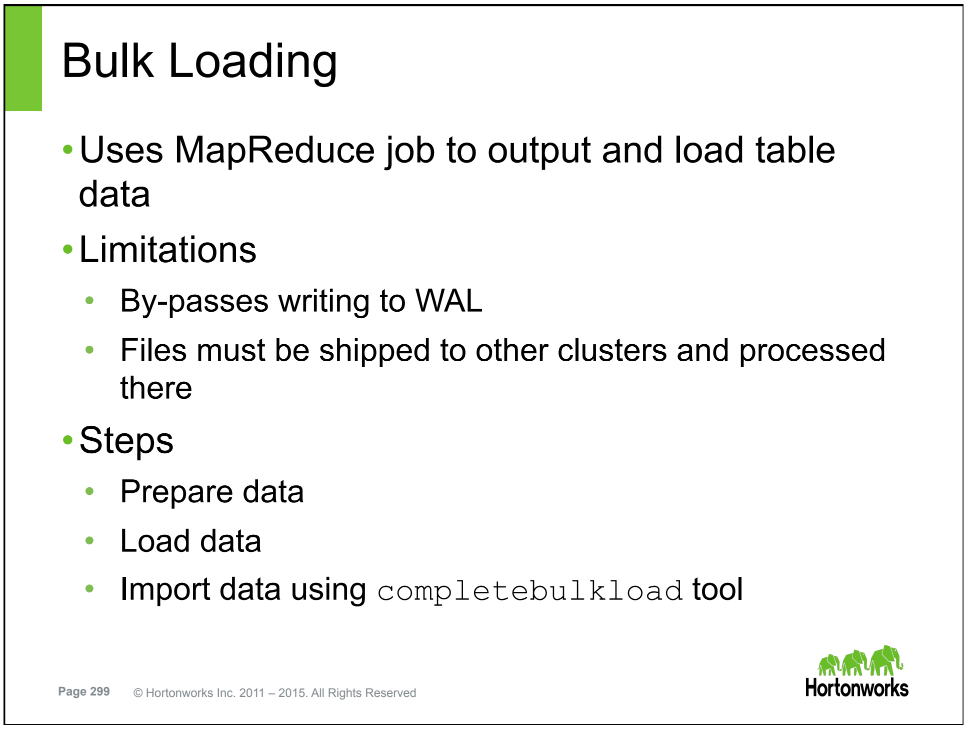


Filters

- PrefixFilter
  - Argument: rowkey
  - Returns key-values in a row that start with the specified prefix
- ColumnPrefixFilter
  - Argument: column prefix (must be in form `qualifier`)
  - Returns key-values in a column that starts with specified prefix
- MultipleColumnPrefixFilter
  - Argument: list of column prefixes (must be in form `qualifier`)
  - Returns key-values in a column that starts with specified prefixes

Page 298 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

Hortonworks

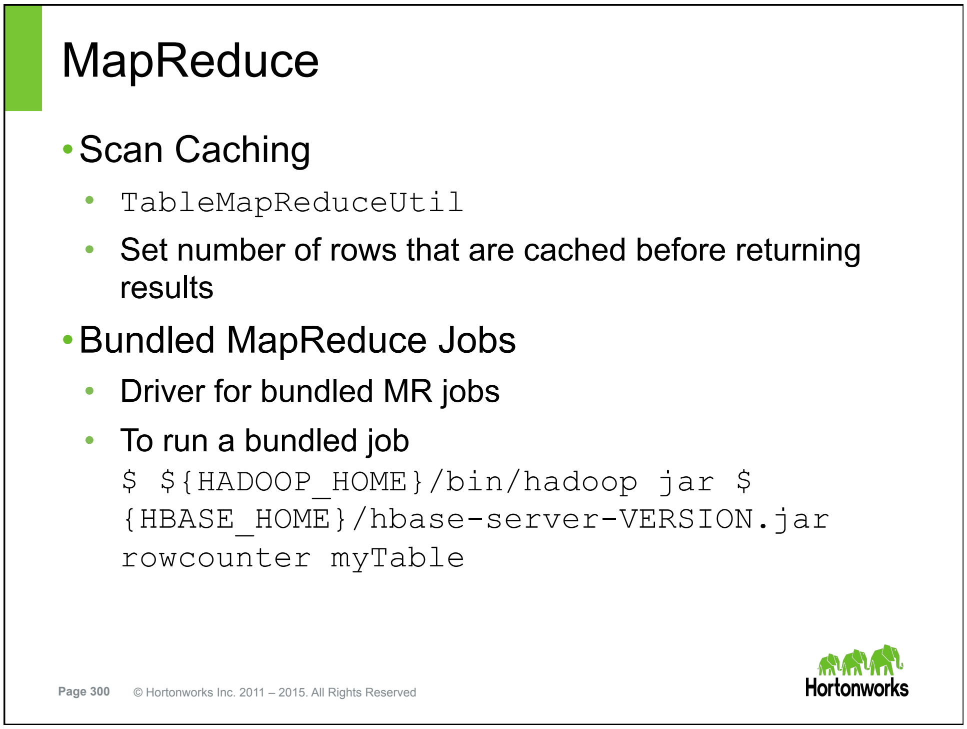


Bulk Loading

- Uses MapReduce job to output and load table data
- Limitations
  - By-passes writing to WAL
  - Files must be shipped to other clusters and processed there
- Steps
  - Prepare data
  - Load data
  - Import data using `completebulkload` tool

Page 299

Hortonworks



MapReduce

- Scan Caching
  - `TableMapReduceUtil`
  - Set number of rows that are cached before returning results
- Bundled MapReduce Jobs
  - Driver for bundled MR jobs
  - To run a bundled job

```
$ ${HADOOP_HOME}/bin/hadoop jar $
{HBASE_HOME}/hbase-server-VERSION.jar
rowcounter myTable
```

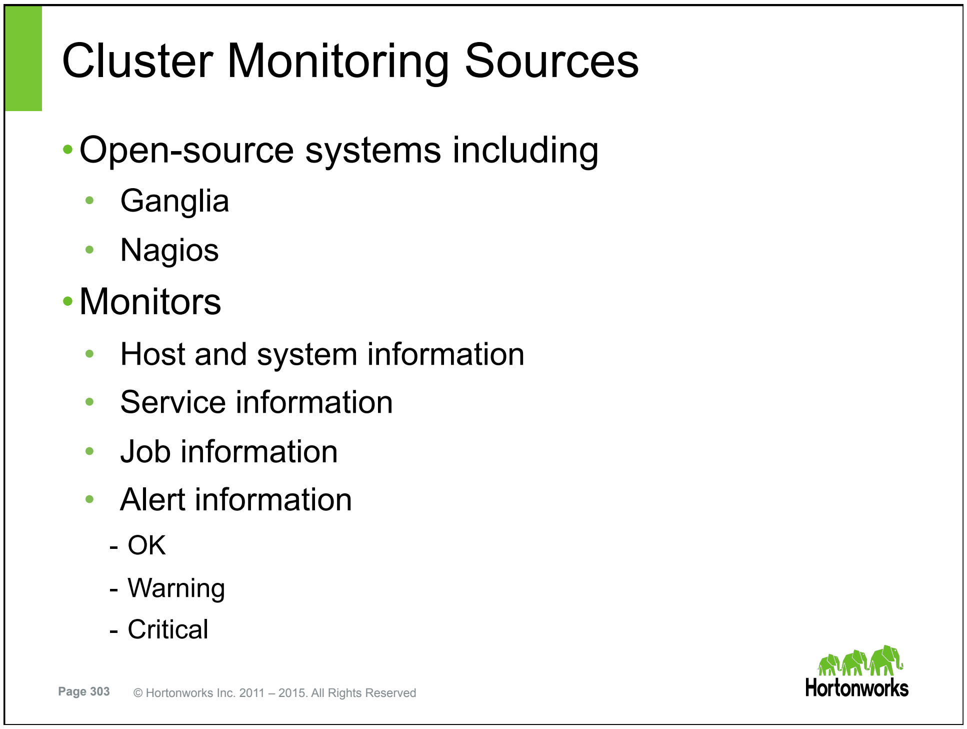
Page 300

Hortonworks

Appendix D

Monitoring and Managing HBase

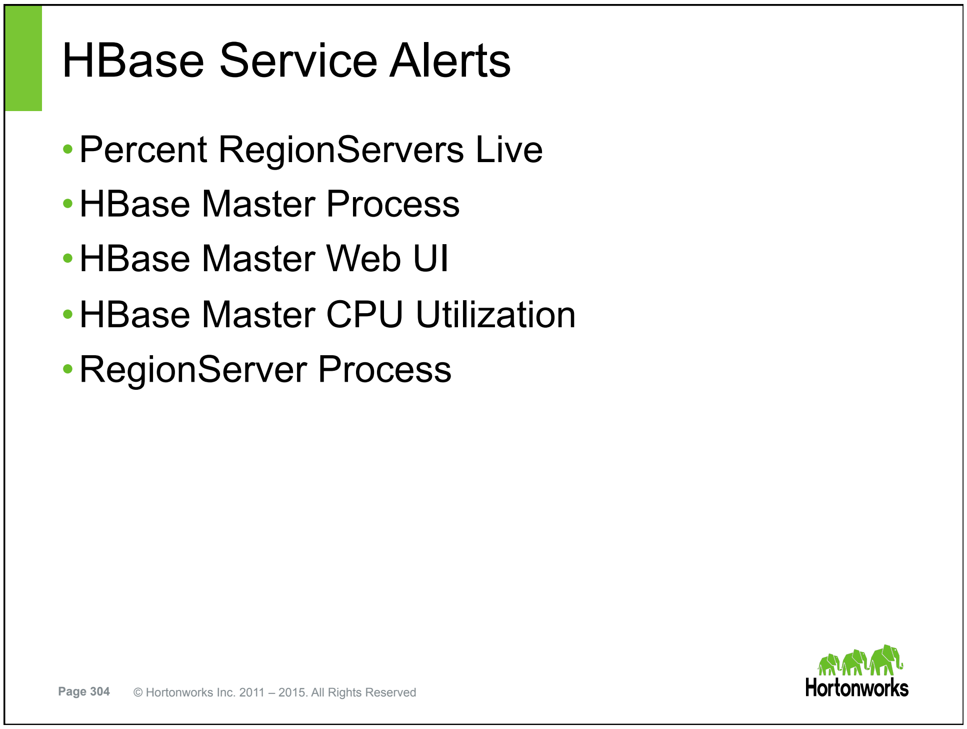
Monitoring HBase



Cluster Monitoring Sources

- Open-source systems including
  - Ganglia
  - Nagios
- Monitors
  - Host and system information
  - Service information
  - Job information
  - Alert information
    - OK
    - Warning
    - Critical

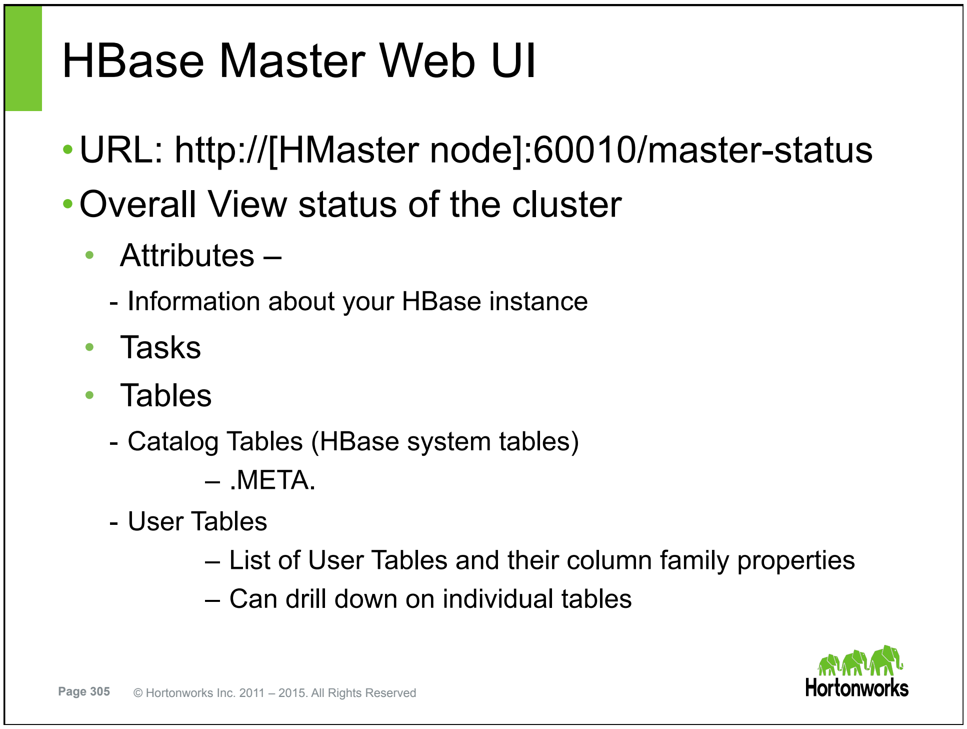
Page 303



HBase Service Alerts

- Percent RegionServers Live
- HBase Master Process
- HBase Master Web UI
- HBase Master CPU Utilization
- RegionServer Process

Page 304

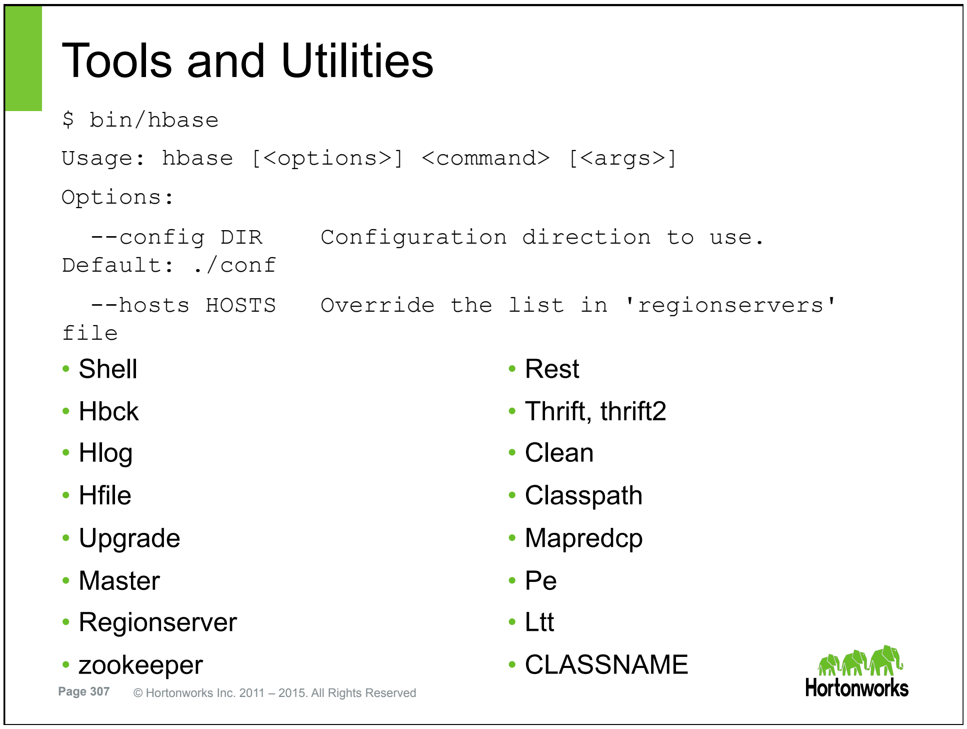


HBase Master Web UI

- URL: http://[HMaster node]:60010/master-status
- Overall View status of the cluster
  - Attributes –
    - Information about your HBase instance
  - Tasks
  - Tables
    - Catalog Tables (HBase system tables)
      - .META.
    - User Tables
      - List of User Tables and their column family properties
      - Can drill down on individual tables

Page 305

Managing Data



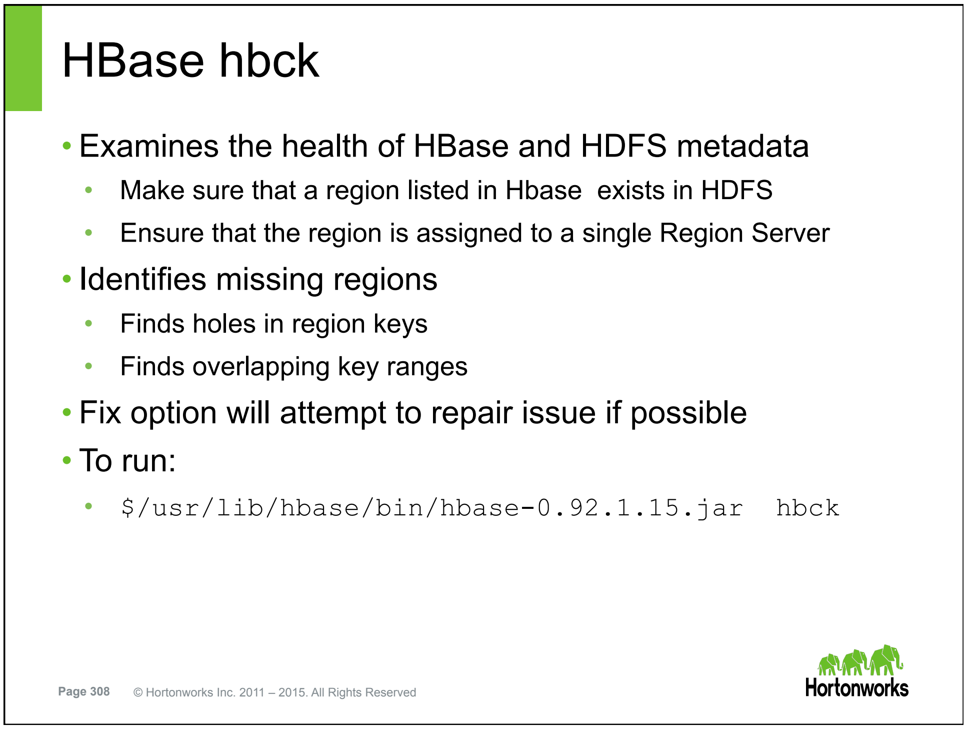
Tools and Utilities

```
$ bin/hbase
Usage: hbase [<options>] <command> [<args>]
Options:
  --config DIR    Configuration direction to use.
Default: ./conf
  --hosts HOSTS   Override the list in 'regionservers'
file
```

- Shell
- Hbck
- Hlog
- Hfile
- Upgrade
- Master
- Regionserver
- zookeeper
- Rest
- Thrift, thrift2
- Clean
- Classpath
- Mapredcp
- Pe
- Ltt
- CLASSNAME

Page 307

Hortonworks

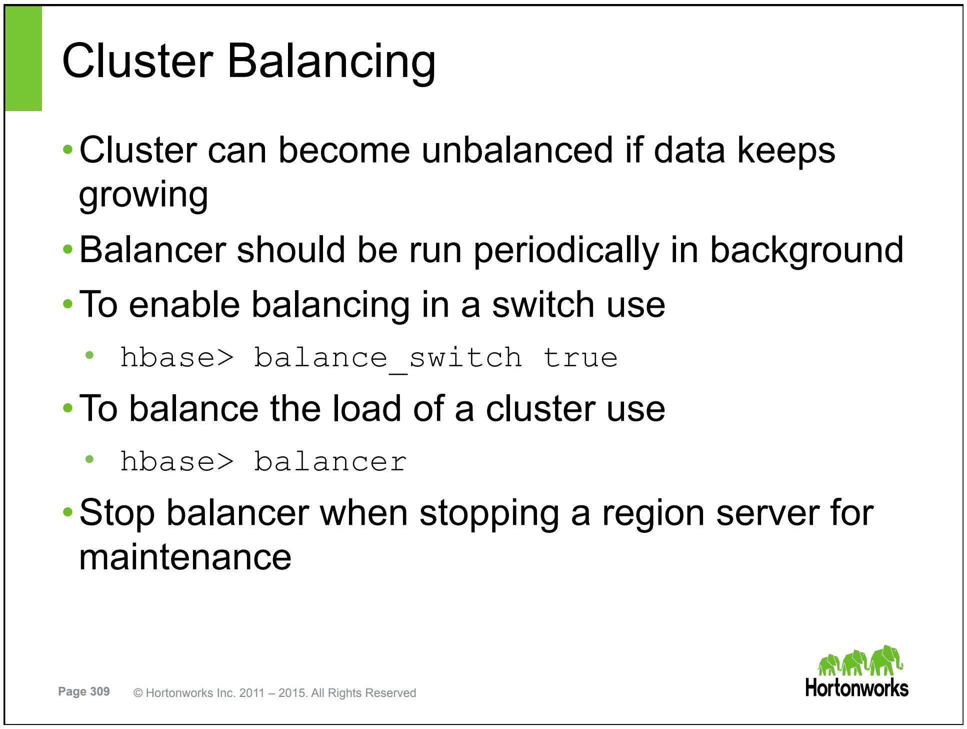


HBase hbck

- Examines the health of HBase and HDFS metadata
  - Make sure that a region listed in Hbase exists in HDFS
  - Ensure that the region is assigned to a single Region Server
- Identifies missing regions
  - Finds holes in region keys
  - Finds overlapping key ranges
- Fix option will attempt to repair issue if possible
- To run:
  - `$/usr/lib/hbase/bin/hbase-0.92.1.15.jar  hbck`

Page 308

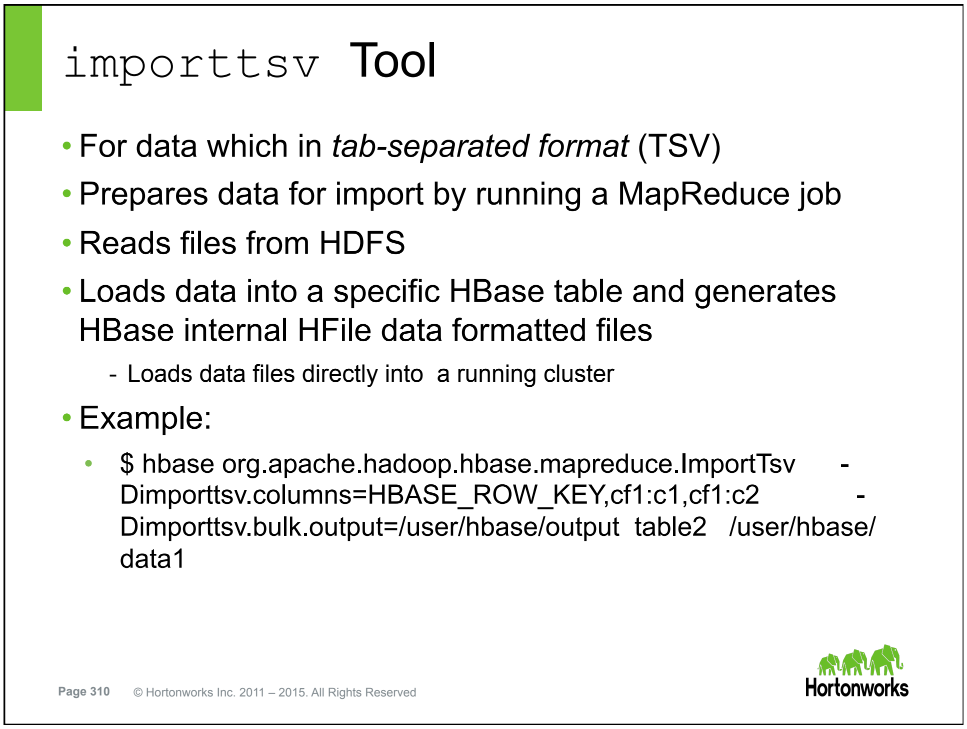
Hortonworks



Cluster Balancing

- Cluster can become unbalanced if data keeps growing
- Balancer should be run periodically in background
- To enable balancing in a switch use
  - `hbase> balance_switch true`
- To balance the load of a cluster use
  - `hbase> balancer`
- Stop balancer when stopping a region server for maintenance

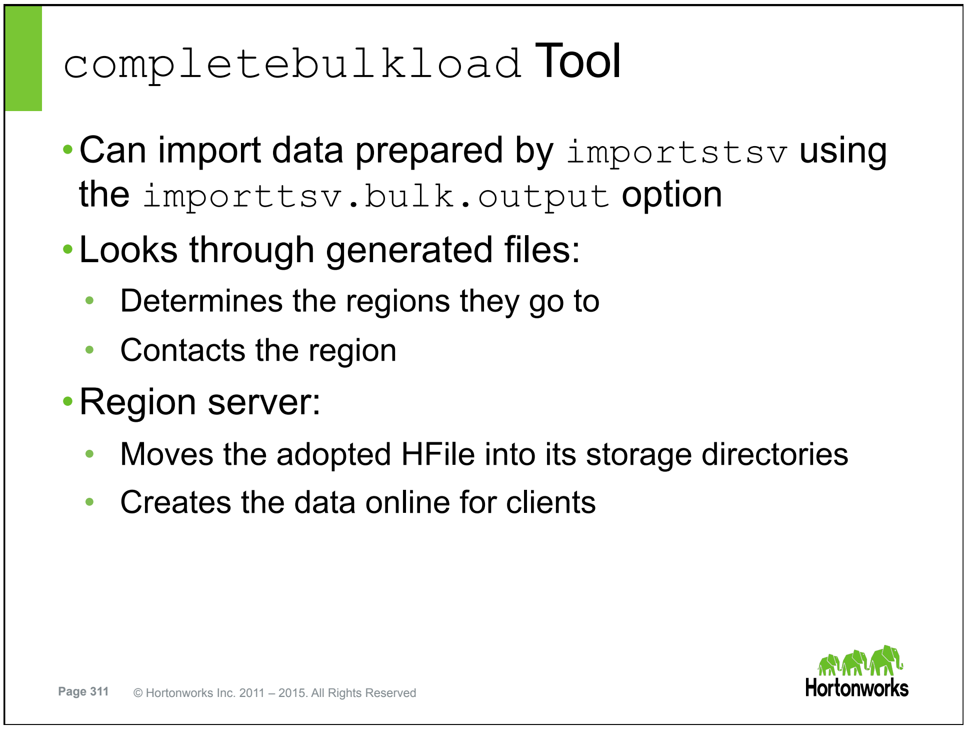
Page 309 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



`importtsv` Tool

- For data which in *tab-separated format* (TSV)
- Prepares data for import by running a MapReduce job
- Reads files from HDFS
- Loads data into a specific HBase table and generates HBase internal HFile data formatted files
  - Loads data files directly into a running cluster
- Example:
  - $ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.columns=HBASE_ROW_KEY,cf1:c1,cf1:c2 -Dimporttsv.bulk.output=/user/hbase/output table2 /user/hbase/data1

Page 310 © Hortonworks Inc. 2011 – 2015. All Rights Reserved

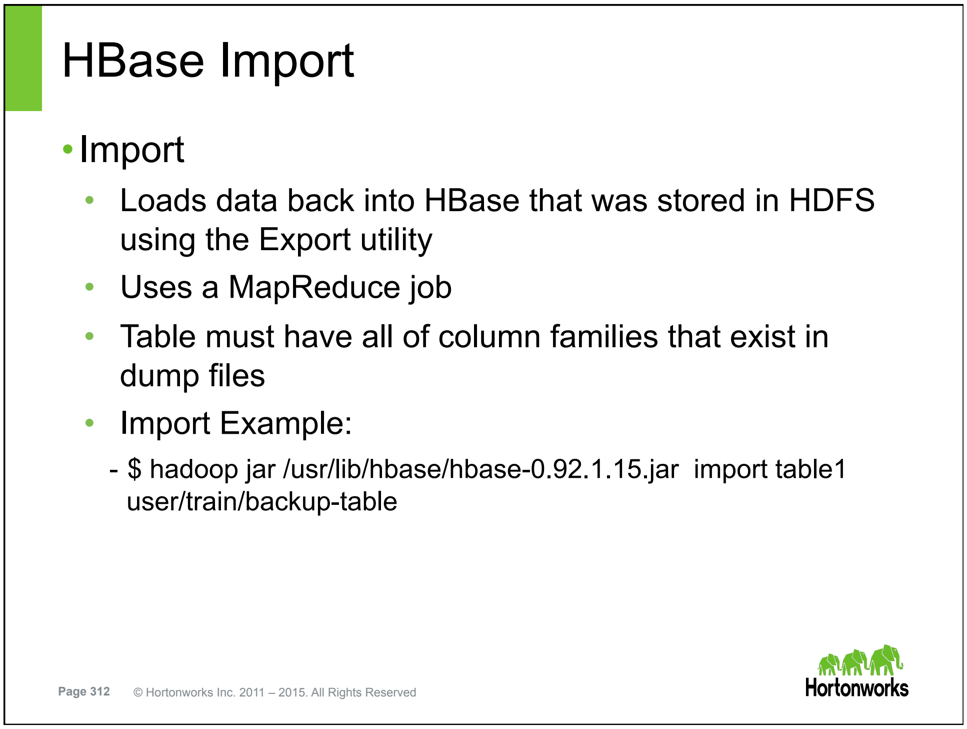


`completebulkload` Tool

- Can import data prepared by `importstsv` using the `importtsv.bulk.output` option
- Looks through generated files:
  - Determines the regions they go to
  - Contacts the region
- Region server:
  - Moves the adopted HFile into its storage directories
  - Creates the data online for clients

Page 311

Hortonworks

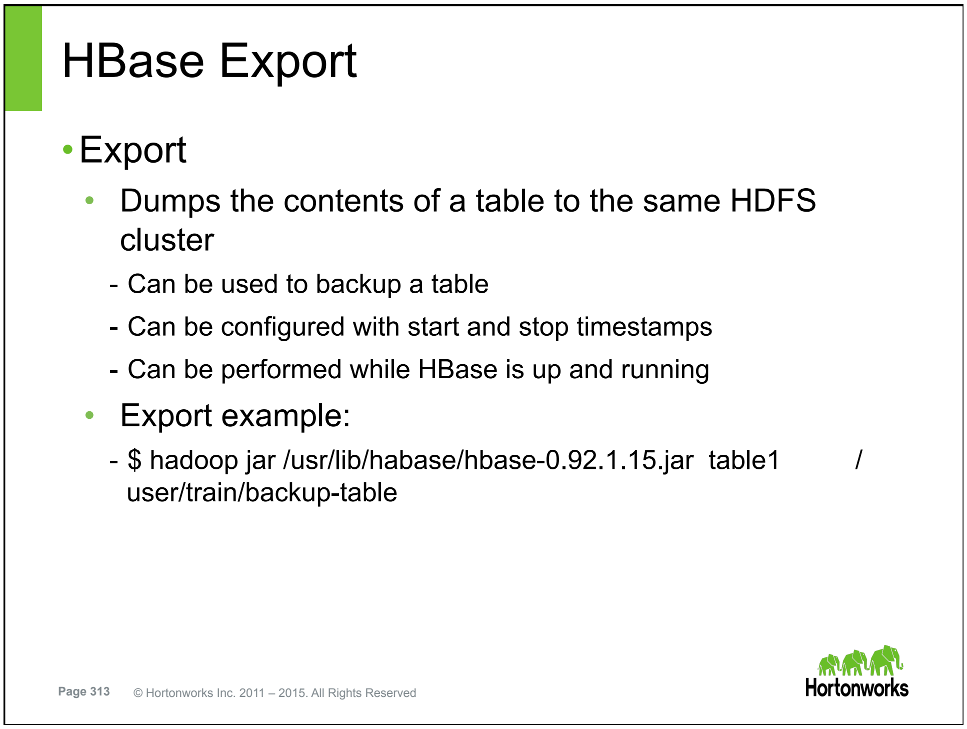


HBase Import

- Import
  - Loads data back into HBase that was stored in HDFS using the Export utility
  - Uses a MapReduce job
  - Table must have all of column families that exist in dump files
  - Import Example:
    - $ hadoop jar /usr/lib/hbase/hbase-0.92.1.15.jar import table1 user/train/backup-table

Page 312

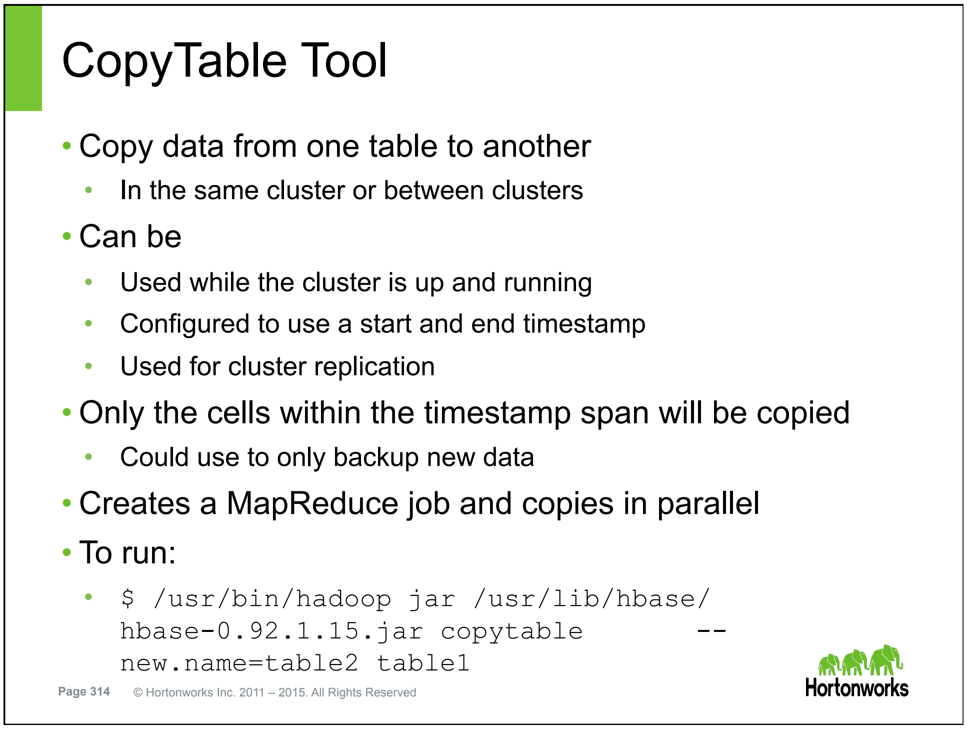
Hortonworks



HBase Export

- Export
  - Dumps the contents of a table to the same HDFS cluster
    - Can be used to backup a table
    - Can be configured with start and stop timestamps
    - Can be performed while HBase is up and running
  - Export example:
    - $ hadoop jar /usr/lib/habase/hbase-0.92.1.15.jar table1 /user/train/backup-table

Page 313 © Hortonworks Inc. 2011 – 2015. All Rights Reserved



CopyTable Tool

- Copy data from one table to another
  - In the same cluster or between clusters
- Can be
  - Used while the cluster is up and running
  - Configured to use a start and end timestamp
  - Used for cluster replication
- Only the cells within the timestamp span will be copied
  - Could use to only backup new data
- Creates a MapReduce job and copies in parallel
- To run:

```
$ /usr/bin/hadoop jar /usr/lib/hbase/
hbase-0.92.1.15.jar copytable       --
new.name=table2 table1
```

Page 314 © Hortonworks Inc. 2011 – 2015. All Rights Reserved