facebook

# Storage Infrastructure Behind Facebook Messages

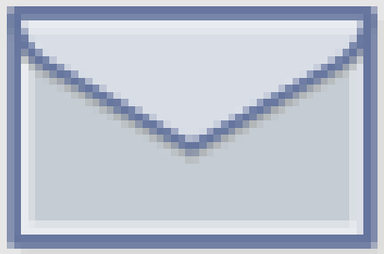
HBase/HDFS/ZK/MapReduce

*Kannan Muthukkaruppan*
*Software Engineer, Facebook*

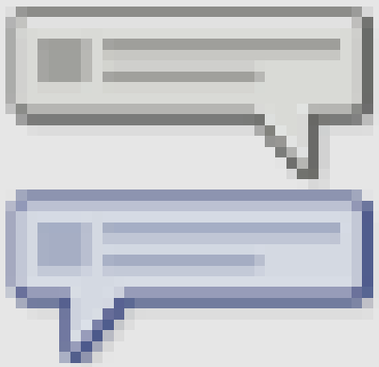
Big Data Experiences & Scars, HPTS 2011
Oct 24th, 2011

# The New Facebook Messages

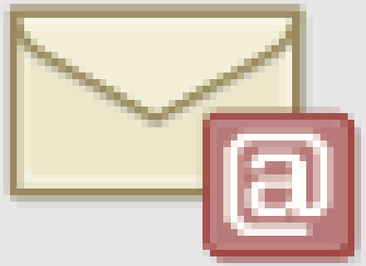
## Messages



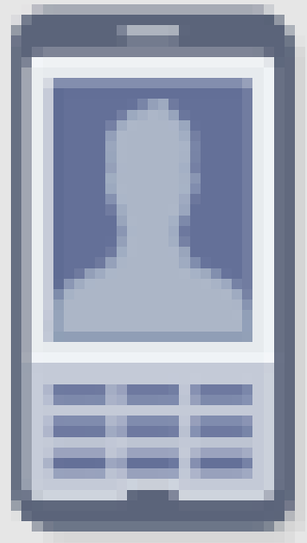
## Chats



## Emails



## SMS



# Why we chose HBase

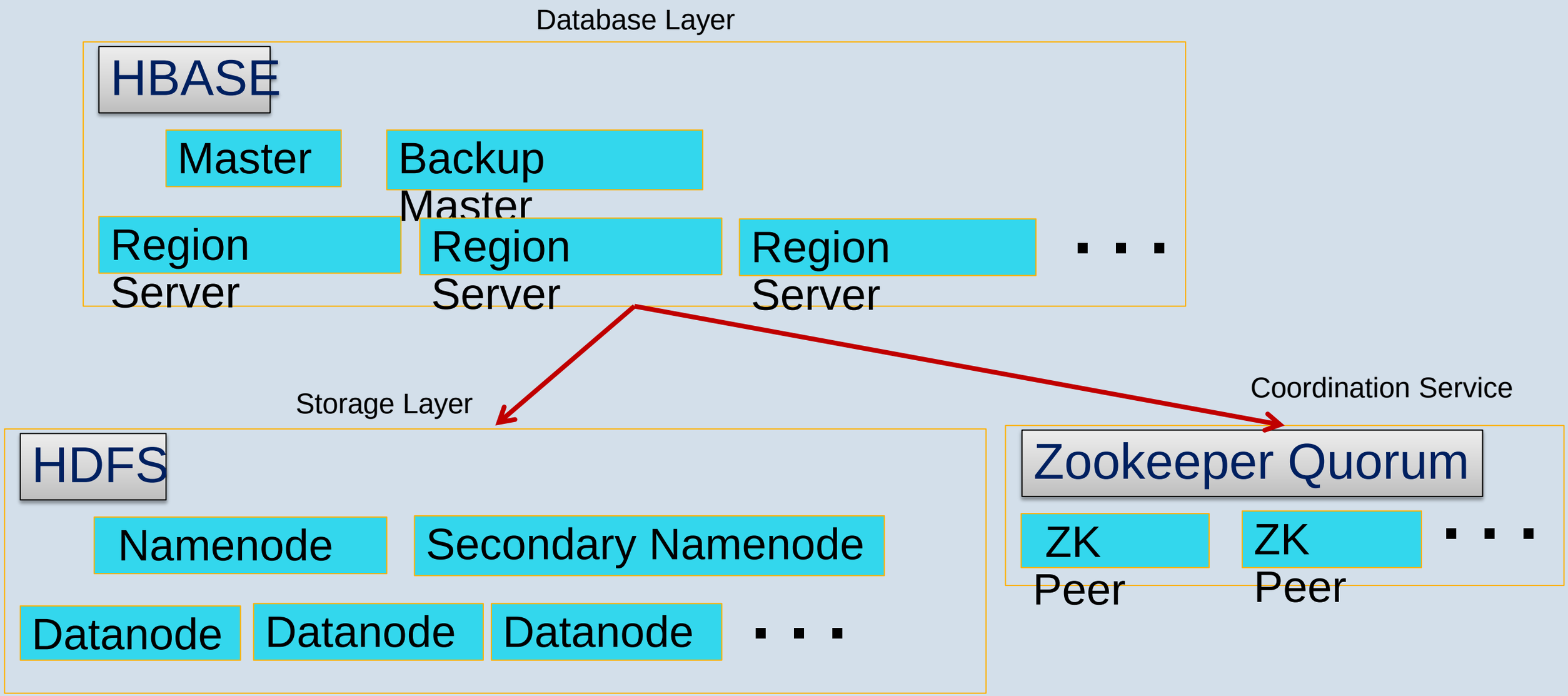
- High write throughput
- Good random read performance
- Horizontal scalability
- Automatic Failover
- Strong consistency
- *Benefits of HDFS*
  - *Fault tolerant, scalable, checksums, MapReduce*
  - *internal dev & ops expertise*

# What do we store in HBase

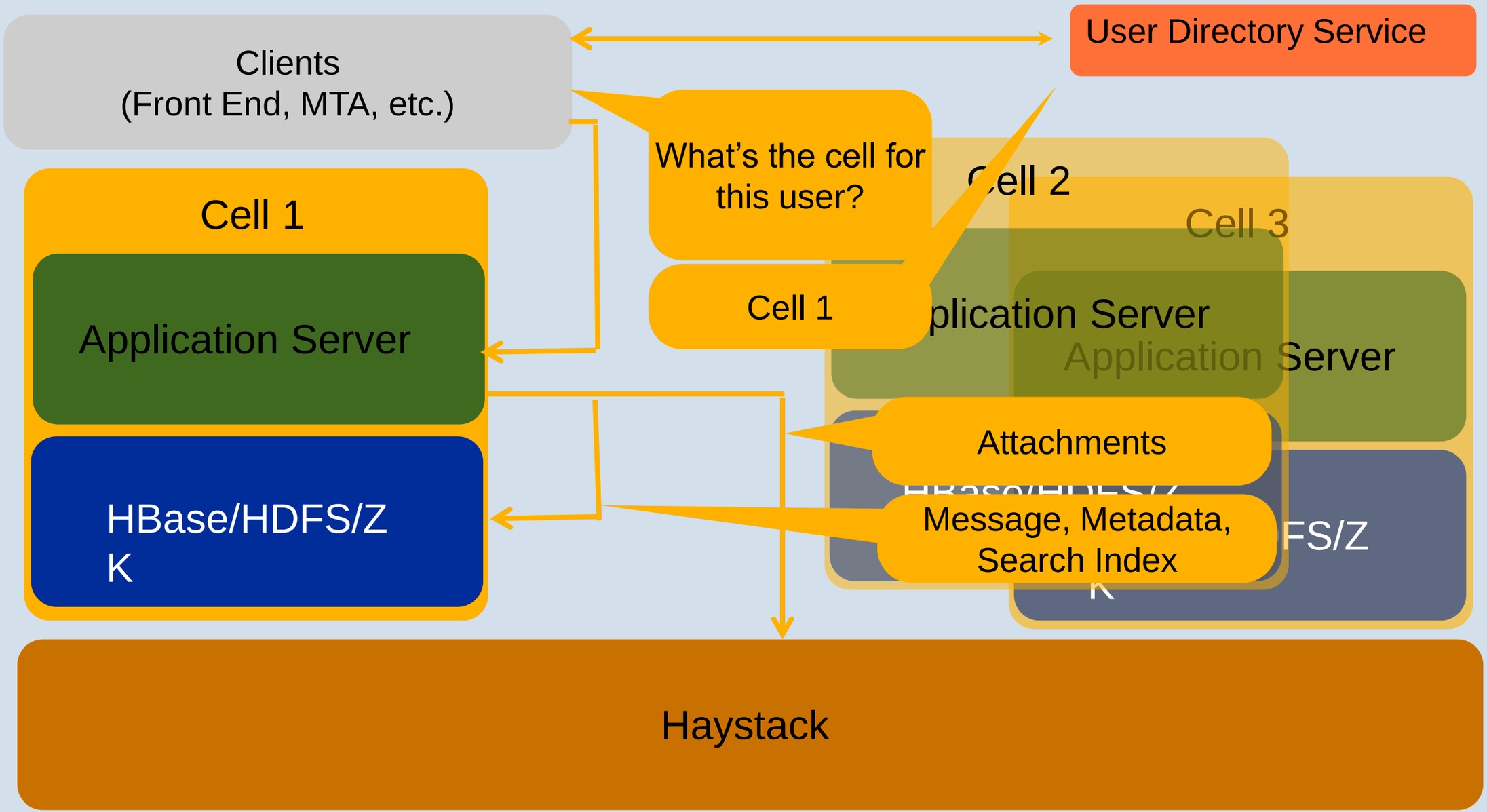
- HBase
  - Small messages
  - Message metadata (thread/message indices)
  - Search index

- Haystack (our photo store)
  - Attachments
  - Large messages

# HBase-HDFS System Overview

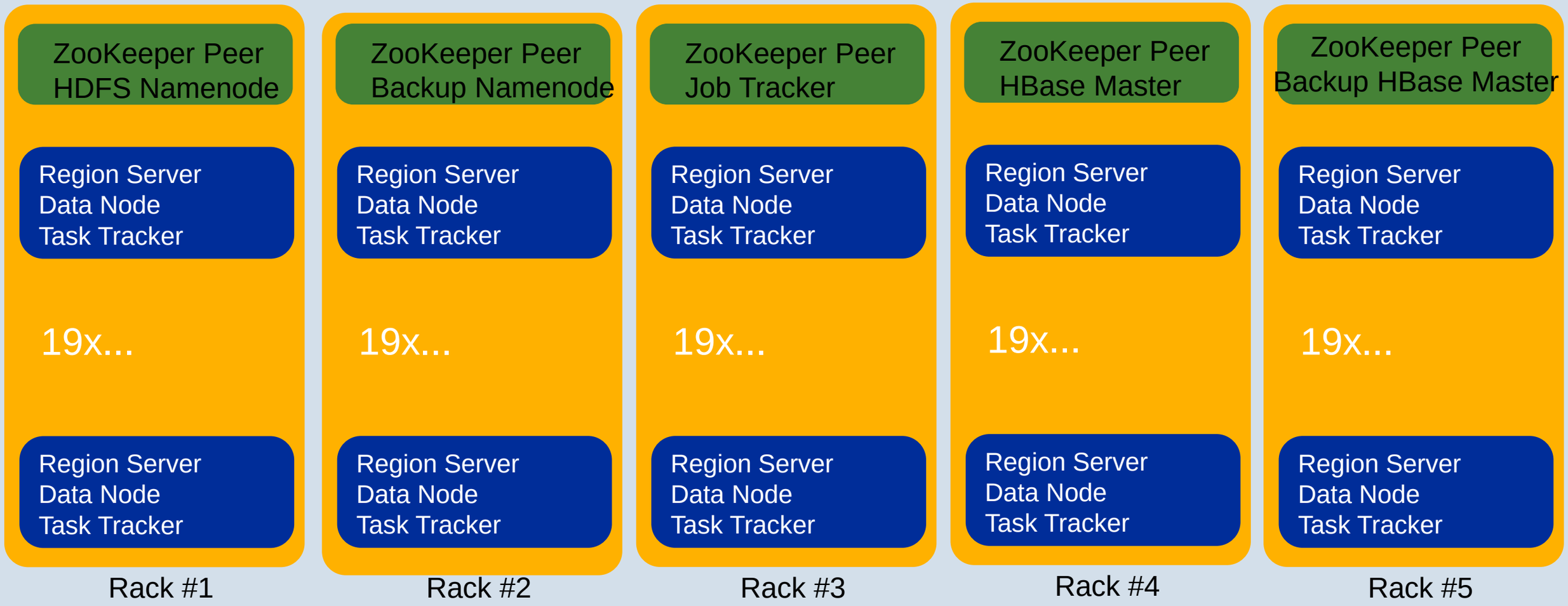


# Facebook Messages Architecture



# Typical Cluster Layout

- Multiple clusters/cells for messaging
  - 20 servers/rack; 5 or more racks per cluster
- Controllers (master/Zookeeper) spread across racks



# Facebook Messages: Quick Stats

- 6B+ messages/day

- Traffic to HBase
  - 75+ Billion  R+W ops/day
  - At peak: 1.5M ops/sec
  - ~ 55% Read vs. 45% Write ops
  - Avg write op inserts ~16 records across multiple column families.

# Facebook Messages: Quick Stats (contd.)

- 2PB+ of online data in HBase (6PB+ with replication; excludes backups)
  - message data, metadata, search index
- All data LZO compressed
- Growing at 250TB/month

# Facebook Messages: Quick Stats (contd.)

Timeline:

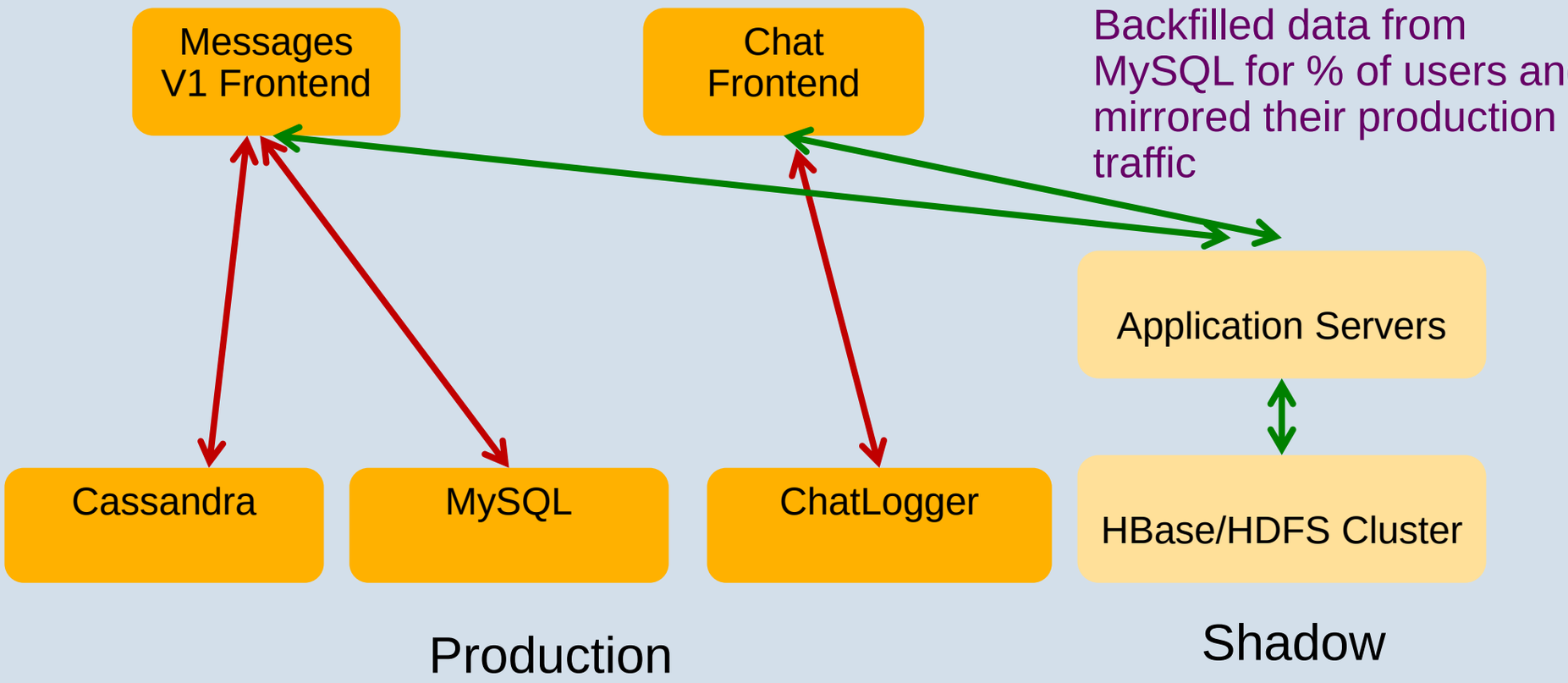
- Started in Dec 2009
- Roll out started in Nov 2010
- Fully rolled out by July 2011 (migrated 1B+ accounts from legacy messages!)

While in production:

- Schema changes: not once, but twice!
- Implemented & rolled out HFile V2 and numerous other optimizations in an upward compatible manner!

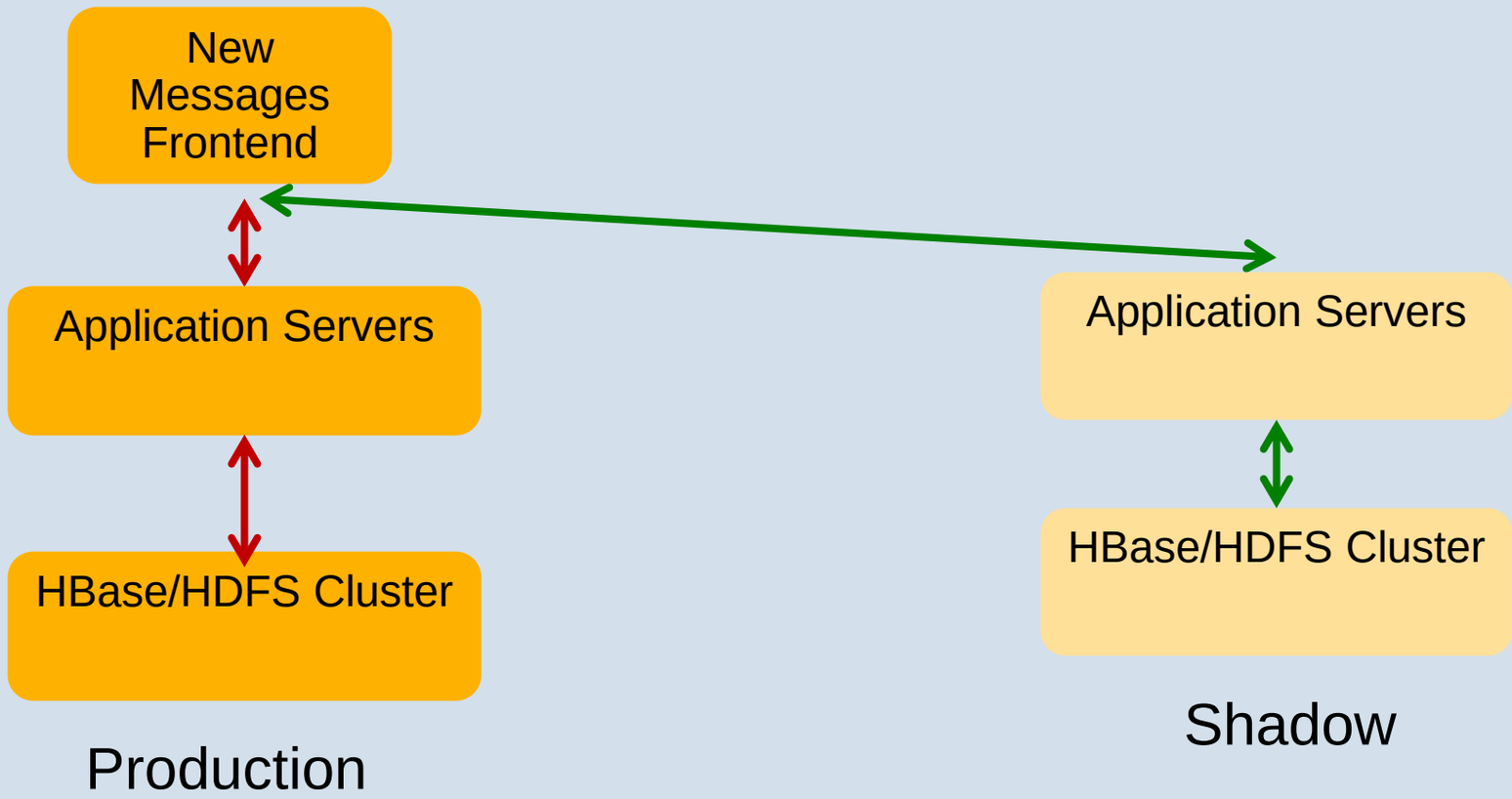
# Shadow Testing: Before Rollout

- Both product and infrastructure were changing.
- Shadows the old messages product + chat while the new one was under development



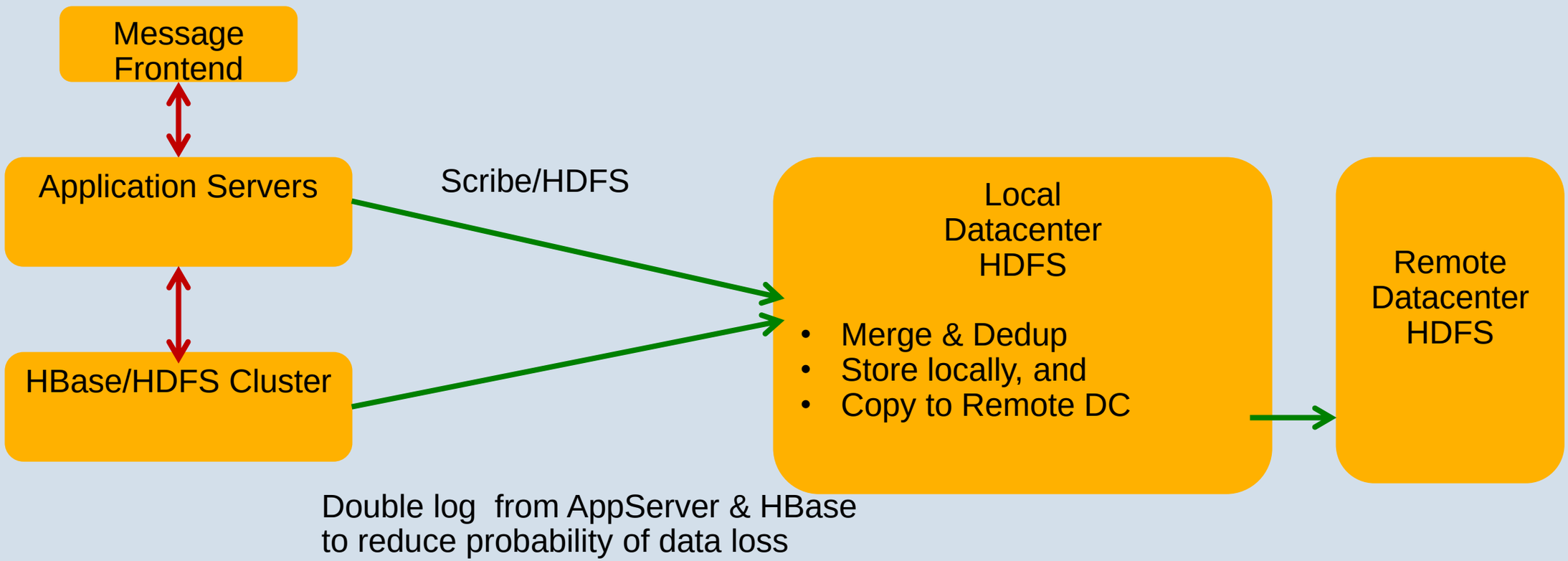
# Shadow Testing: After Rollout

- *Shadows new version of the Messages product.*
- *All backend changes go through shadow cluster before prod push*



# Backup/Recovery (V1)

- During early phase, concerned about potential bugs in HBase.
- Off-line backups: written to HDFS via Scribe
- Recovery tools; testing of recovery tools



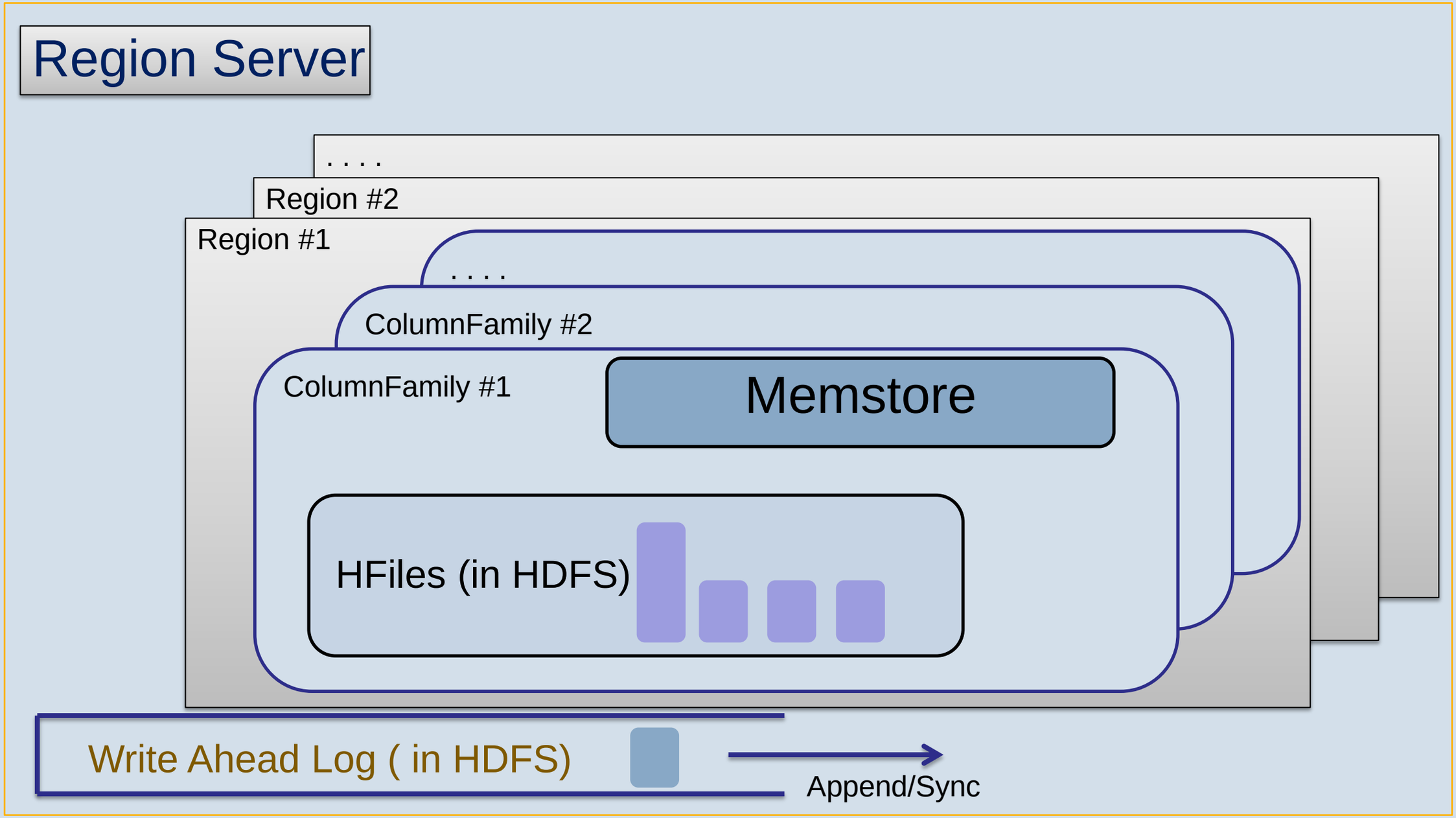
# Backups (V2)

- Now, does periodic HFile level backups.
- Working on:
  - Moving to HFile + Commit Log based backups to be able to recover to finer grained points in time
    - Avoid need to log data to Scribe.
  - Zero copy (hard link based) fast backups

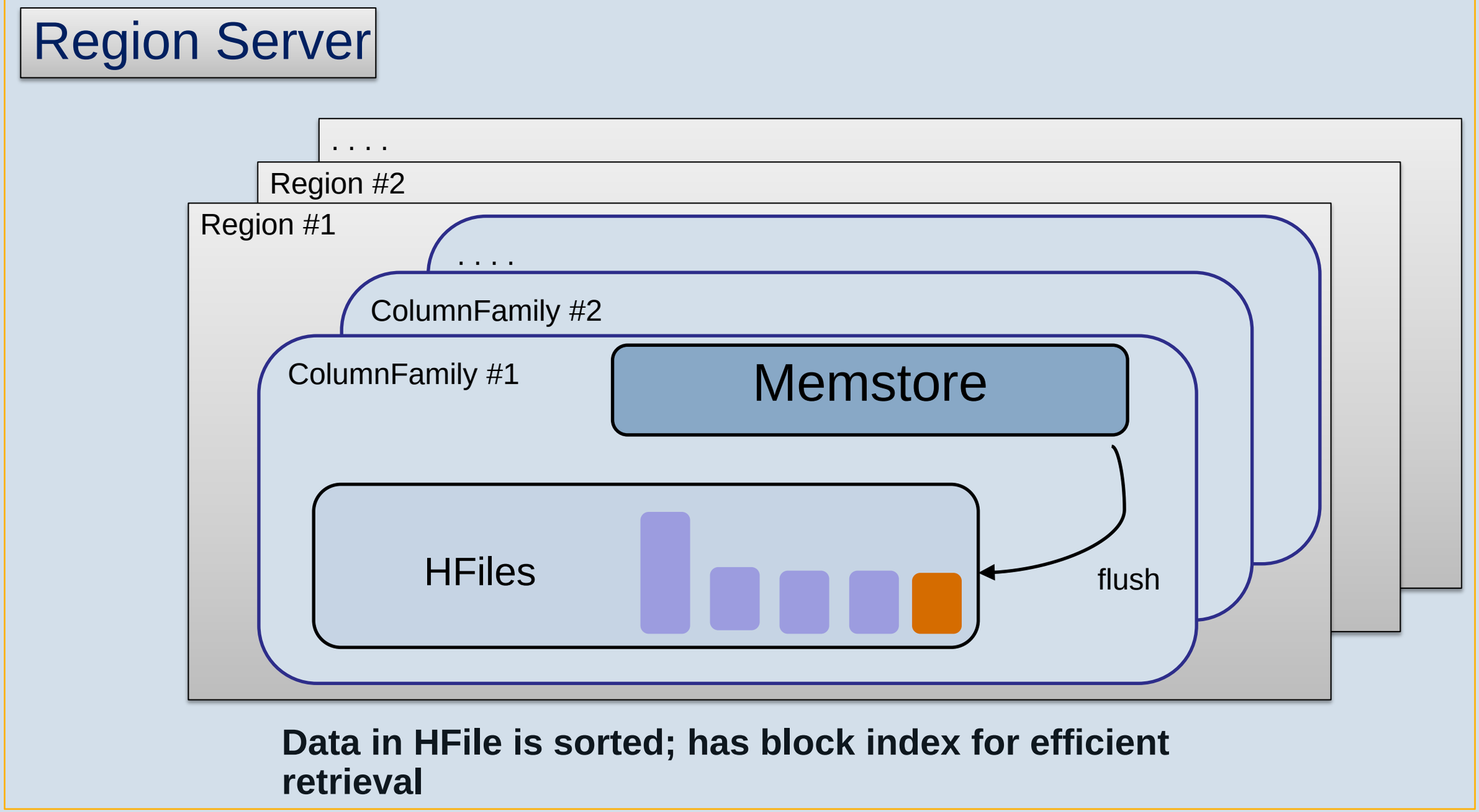
# Messages Schema & Evolution

- “Actions” (data) Column Family the source of truth
  - Log of all user actions (addMessage, markAsRead, etc.)
- Metadata (thread index, message index, search index) etc. in other column families
- Metadata portion of schema underwent 3 changes:
  - Coarse grained snapshots (early development; rollout up to 1M users)
  - Hybrid (up to full rollout – 1B+ accounts; 800M+ active)
  - Fine-grained metadata (after rollout)
- MapReduce jobs against production clusters!
  - Ran in throttled way
  - Heavy use of HBase bulk import features

# Write Path Overview

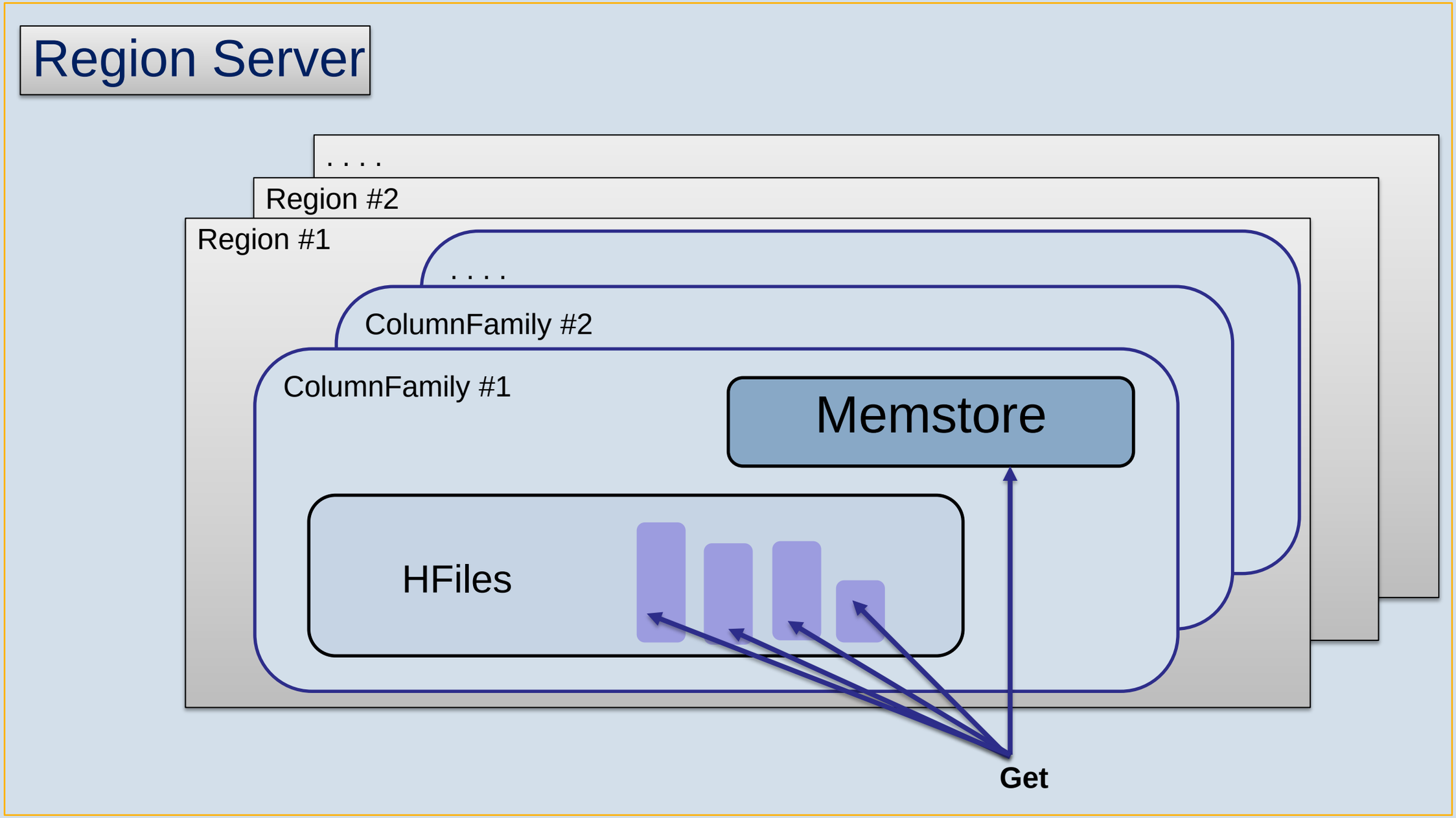


# Flushes: Memstore -> HFile

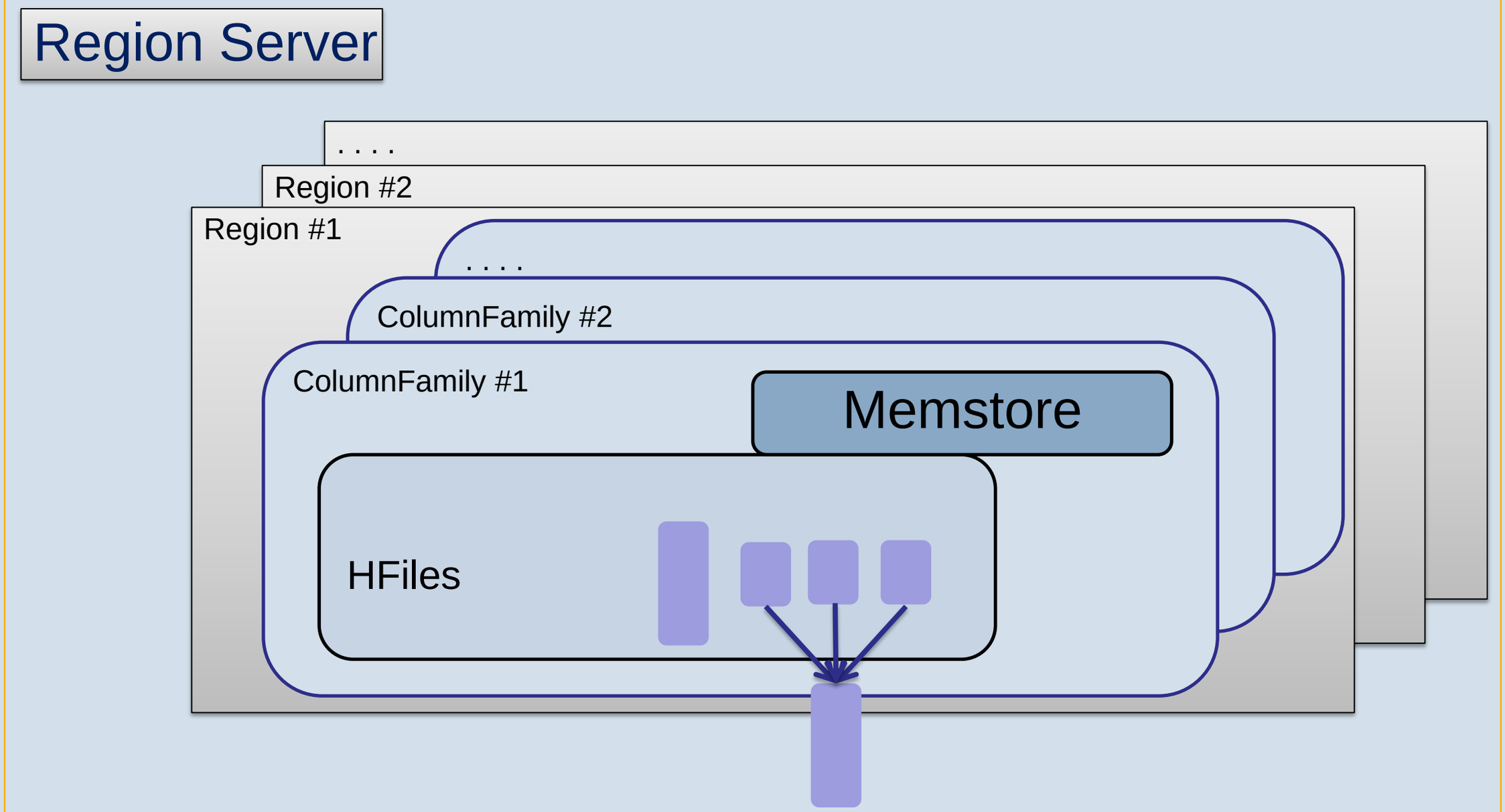



**Data in HFile is sorted; has block index for efficient retrieval**

# Read Path Overview



# Compactions



# Reliability: Early work

- HDFS sync support for durability of transactions
- Multi-CF transaction atomicity
- Several bug fixes in log recovery
- New block placement policy in HDFS
  - To reduce probability of data loss

# Availability: Early Work

- Common reasons for unavailability:
  - S/W upgrades
    - *Solution: rolling upgrades*
  - Schema Changes
    - Applications needs new Column Families
    - Need to change settings for a CF
    - *Solution: online “alter table”*
  - Load balancing or cluster restarts took forever
    - Upon investigation: stuck waiting for compactions to finish
    - *Solution: Interruptible Compactions!*

# Performance: Early Work

- Read optimizations:
  - Seek optimizations for rows with large number of cells
  - Bloom Filters
    - minimize HFile lookups
  - Timerange hints on HFiles (great for temporal data)
  - Multigets
  - Improved handling of compressed HFiles

# Performance: Compactions

- Critical for read performance
- Old Algorithm:

#1. Start from newest file (file 0); include next file if:

- size[i] < size[i-1] * C  (good!)

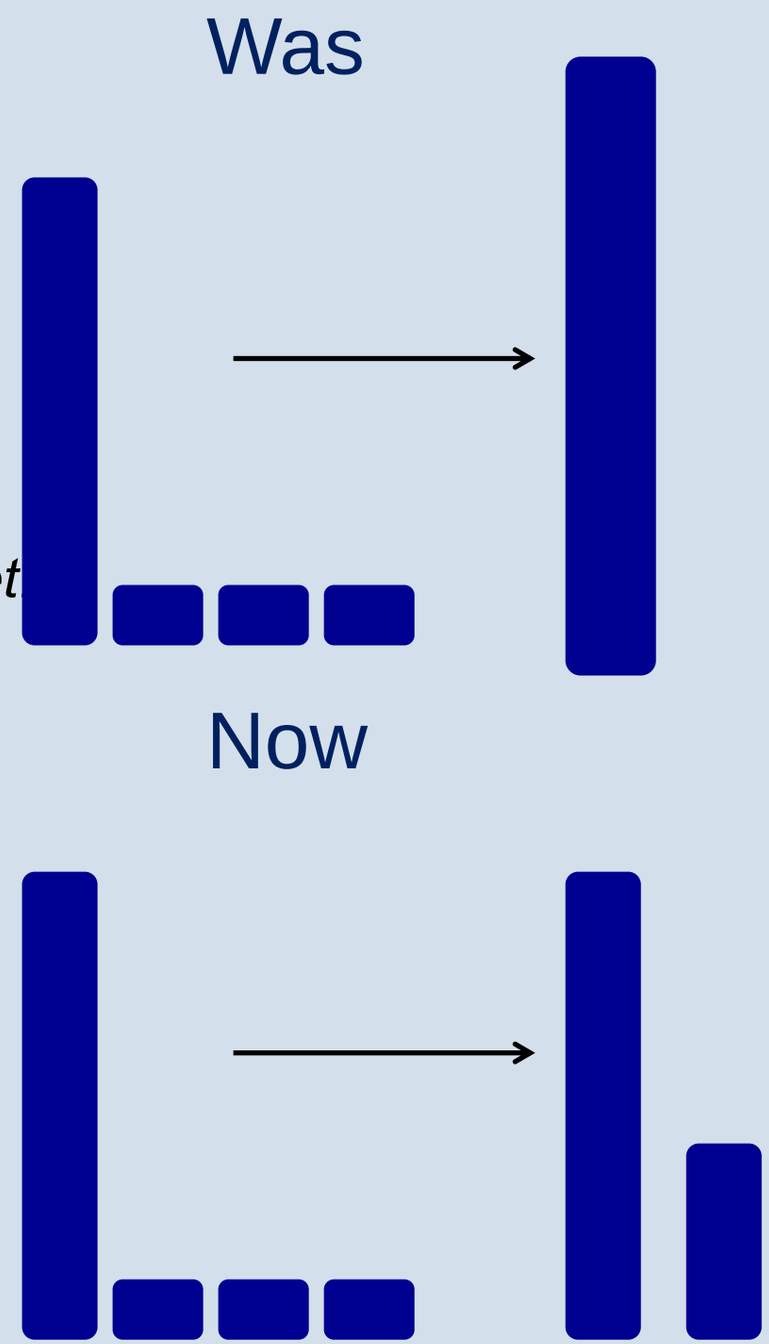
*#2. Always compact at least 4 files, even if rule #1 isn't met*

Solution:

#1. Compact at least 4 files, *but only if eligible files found*.

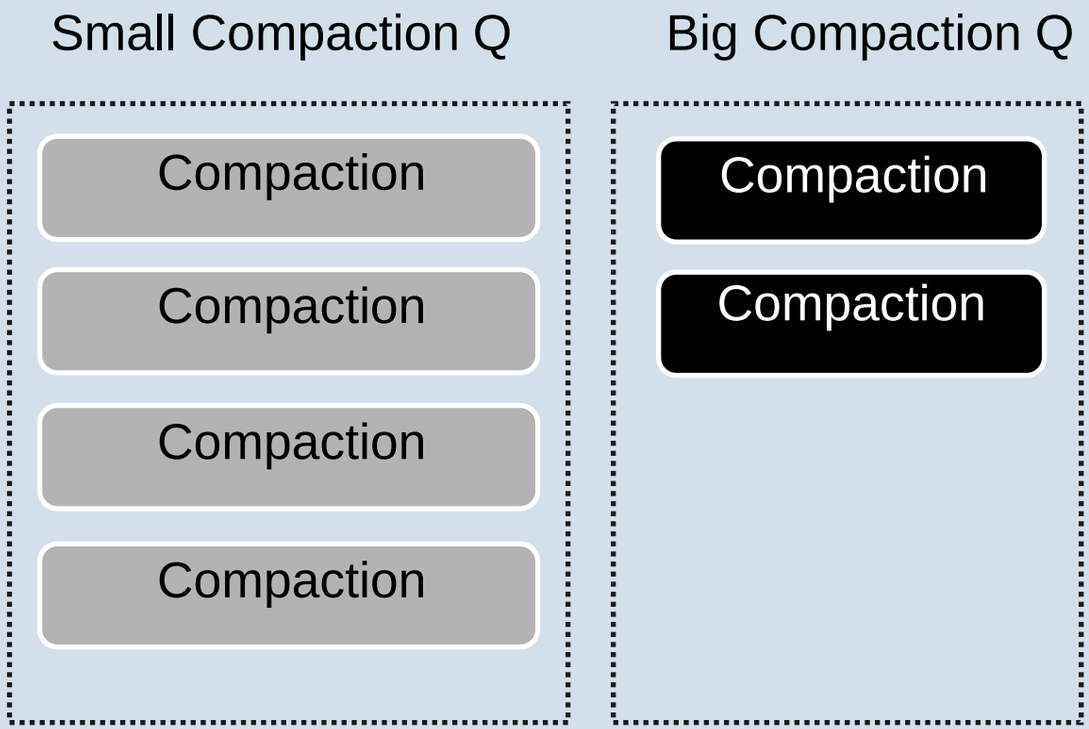
#2. Also, new file selection based on summation of sizes.

size[i] < (size[0] + size[1] + …size[i-1]) * C



# Performance: Compactions

- More problems!
  - Read performance dips during peak
  - Major compaction storms
  - Large compactions bottleneck
- Enhancements/fixes:
  - Staggered major compactions
  - Multi-thread compactions; separate queues for small & big compactions
  - Aggressive off-peak compactions



# Metrics, metrics, metrics…

- Initially, only had coarse level overall metrics (get/put latency/ops; block cache counters).
- Slow query logging
- Added per Column Family stats for:
  - ops counts, latency
  - block cache usage & hit ratio
  - memstore usage
  - on-disk file sizes
  - file counts
  - bytes returned, bytes flushed, compaction statistics
  - stats by block type (data block vs. index blocks vs. bloom blocks, etc.)
  - bloom filter stats

# Metrics (contd.)

- HBase Master Statistics:
  - Number of region servers alive
  - Number of regions
  - Load balancing statistics
  - ..
- All stats stored in Facebook's Operational Data Store (ODS).
- Lots of ODS dashboards for debugging issues
  - Side note: ODS planning to use HBase for storage pretty soon!

# Need to keep up as data grows on you!

- Rapidly iterated on several new features while in production:
  - Block indexes upto 6GB per server! Cluster starts taking longer and longer. Block cache hit ratio on the decline.
    - Solution: HFile V2
      - Multi-level block index, Sharded Bloom Filters
  - Network pegged after restarts
    - Solution: Locality on full & rolling restart
  - High disk utilization during peak
    - Solution: Several “seek” optimizations to reduce disk IOPS
      - Lazy Seeks (use time hints to avoid seeking into older HFiles)
      - Special bloom filter for deletes to avoid additional seek
      - Utilize off-peak IOPS to do more aggressive compactions during

# Scares & Scars!

- Not without our share of scares and incidents:
  - s/w bugs. (e.g., deadlocks, incompatible LZO used for bulk imported data, etc.)
    - found a edge case bug in log recovery as recently as last week!
  - performance spikes every 6 hours (even off-peak)!
    - cleanup of HDFS's Recycle bin was sub-optimal! Needed code and config fix.
  - transient rack switch failures
  - Zookeeper leader election took than 10 minutes when one member of the quorum died. Fixed in more recent version of ZK.
  - HDFS Namenode – SPOF
  - flapping servers (repeated failures)

# Scares & Scars! (contd.)

- Sometimes, tried things which hadn’t been tested in dark launch!
  - Added a rack of servers to help with performance issue
    - Pegged top of the rack network bandwidth!
      - Had to add the servers at much slower pace. Very manual ☹.
      - Intelligent load balancing needed to make this more automated.
- A high % of issues caught in shadow/stress testing
- Lots of alerting mechanisms in place to detect failures cases
  - Automate recovery for a lots of common ones
  - Treat alerts on shadow cluster as hi-pri too!
- Sharding service across multiple HBase cells also paid off

# Future Work

- Reliability, Availability, Scalability!
- Lot of new use cases on top of HBase in the works.

  - HDFS  Namenode HA
  - Recovering gracefully from transient issues
  - Fast hot-backups
  - Delta-encoding in block cache
  - Replication
  - Performance (HBase and HDFS)
  - HBase as a service Multi-tenancy
  - Features- coprocessors, secondary indices

# Acknowledgements

Work of lot of people spanning HBase, HDFS, Migrations, Backup/Recovery pipeline, and Ops!

Karthik Ranganathan
Nicolas Spiegelberg
Aravind Menon
Dhruba Borthakur
Hairong Kuang
Jonathan Gray
Rodrigo Schmidt
Amitanand Aiyer
Guoqiang (Jerry) Chen
Liyin Tang
Madhu Vaidya
Mikhail Bautin
Pritam Damania
Prakash Khemani
Doug Porter
Alex Laslavic
Kannan Muthukkaruppan
+ Apache HBase
+ Interns
+ Messages FrontEnd Team
+ Messages AppServer Team
+ many more folks…

# Thanks! Questions?

facebook.com/engineering