CASSANDRA: A DECENTRALIZED STRUCTURED STORAGE SYSTEM

Avinash Lakshman, Prashant Malik

Facebook

Presented by: Besat Kassaie

Agenda

- Background
- Data Model
- Architecture
- Implementation
- Facebook Inbox Search
- Conclusion
- Discussion

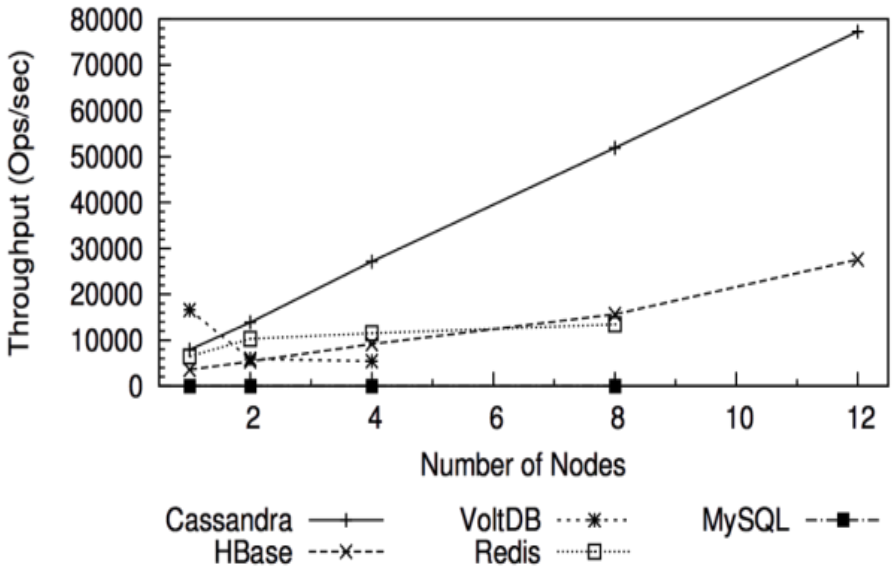
History

- Cassandra was created by Facebook to fullfill the storage needs of the Facebook Inbox Search
- Facebook open-sourced Cassandra in 2008 to Apache
- The latest version released by Apache is 2.1.0

Motivations

High Scalability:

Read/write throughput increases linearly when number of nodes increases

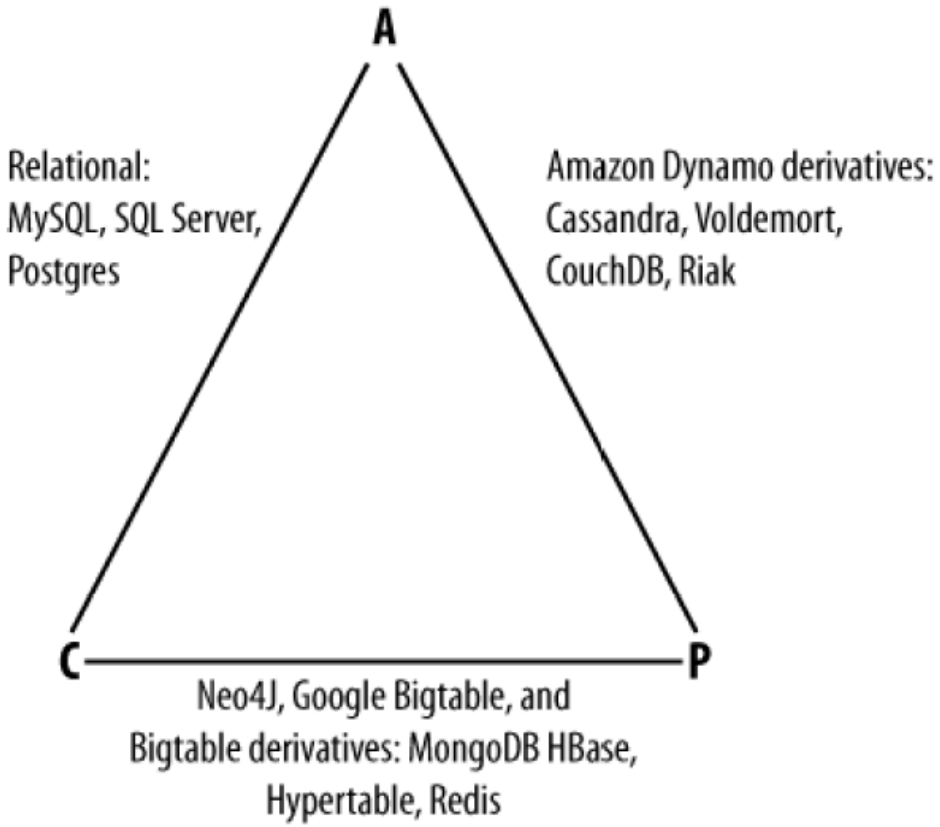


High Availability: Cassandra treats failures as the norm rather than exception

High write throughput: By efficient disk access policy and flexible consistency level

Cassandra & CAP

- CAP theorem : In any given system, we can strongly support only two out of consistency, availability and partition tolerance.



Related Systems

	Cassandra	Bigtable	Dynamo
Data Model	Column-Oriented	Column-Oriented	Key-Value
CAP Theorem	AP	CP	AP
Distributed Architecture	Decentralized P2P	Master-Slave	Decentralized P2P

Related Systems

Google Bigtable

- Column Families
- Memtables
- SSTables

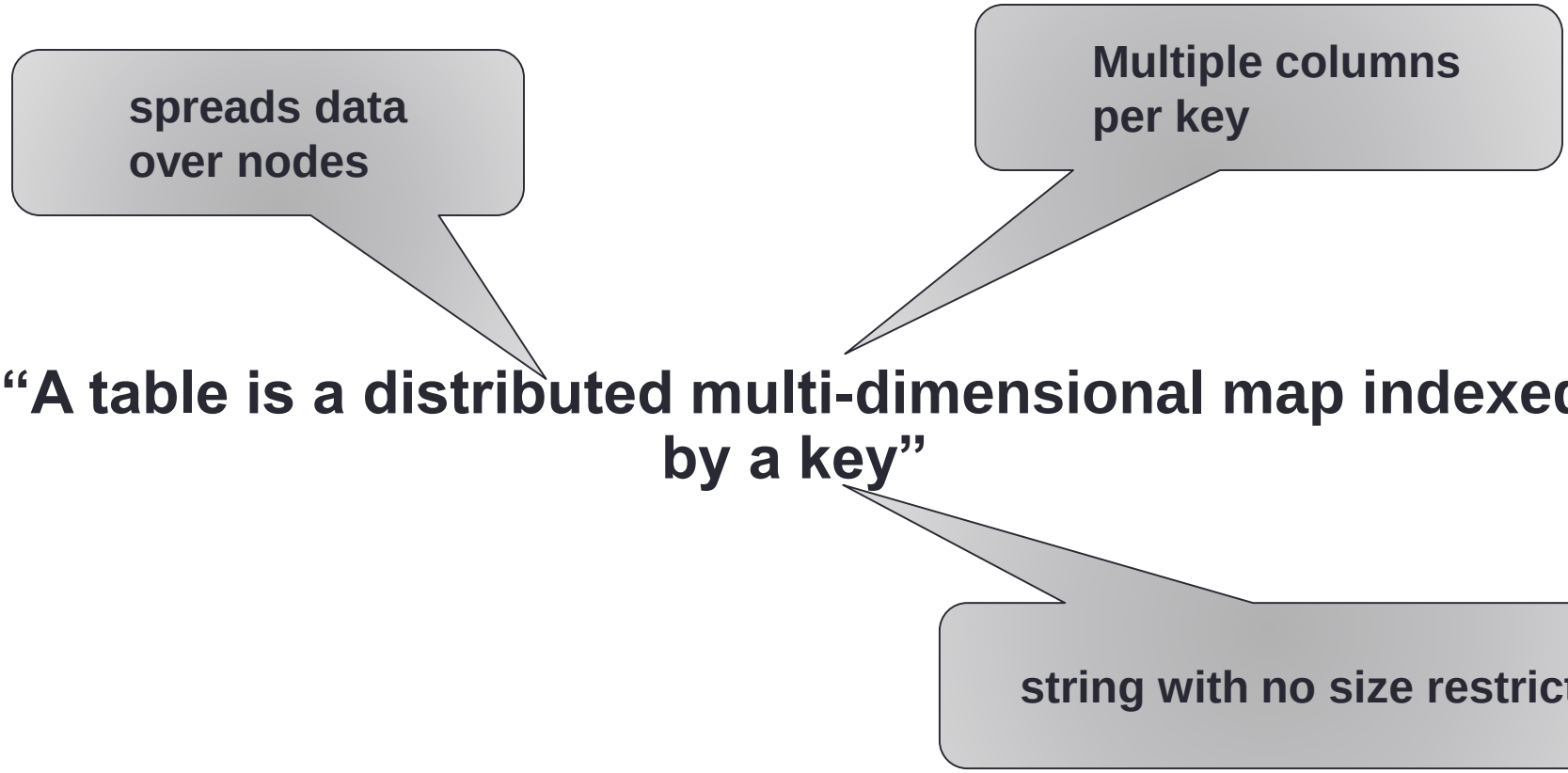
Amazon Dynamo

- Consistent hashing
- Partitioning
- Replication

Agenda

- Background
- Data Model
- Architecture
- Implementation
- Facebook Inbox Search
- Conclusion
- Discussion

Data Model

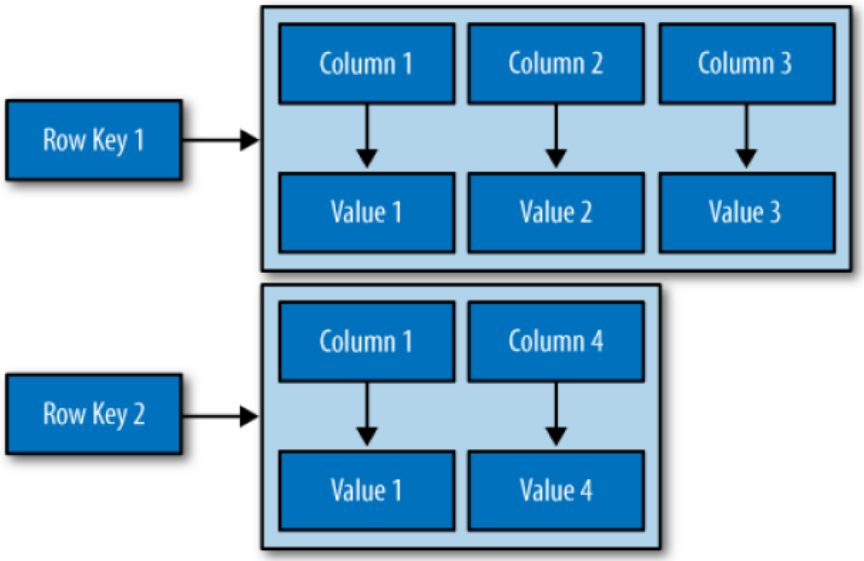


- Operations are **atomic** on each row per replica.

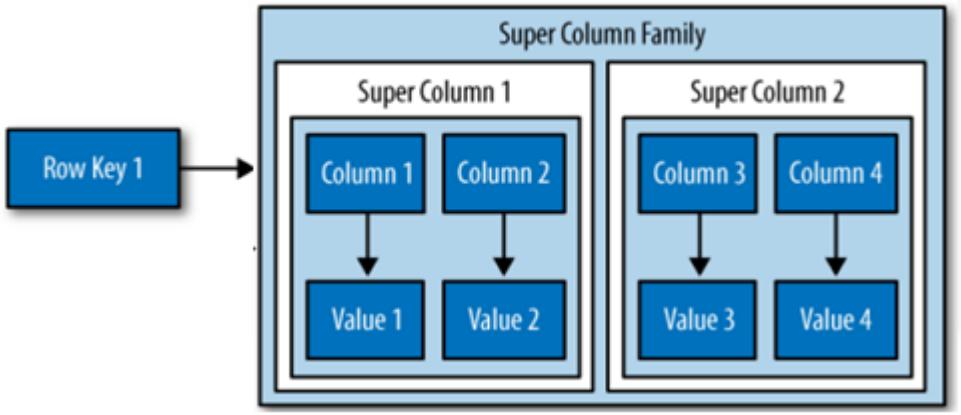
Data Model

- Columns are grouped into Column Families(CF):
  - CFs have to be defined in advance → structured storage system
  - The number of CFs is not limited per table
- Types of Column Families:
  - Simple
  - Super (nested Column Families)
- Column
  - Has (Name, Value, Timestamp) and Can be ordered by timestamps or name
- Row
  - Can have a different number of columns

Data Model



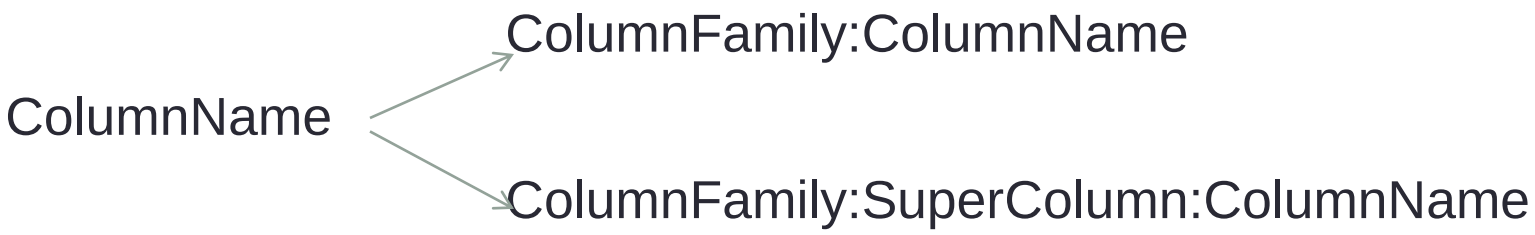
Simple Column Family



Super Column Family

API

- Insert(table,key,rowMutation)
- Get(table,key,ColumnName)
- Delete(tabel,key, ColumnName)



Agenda

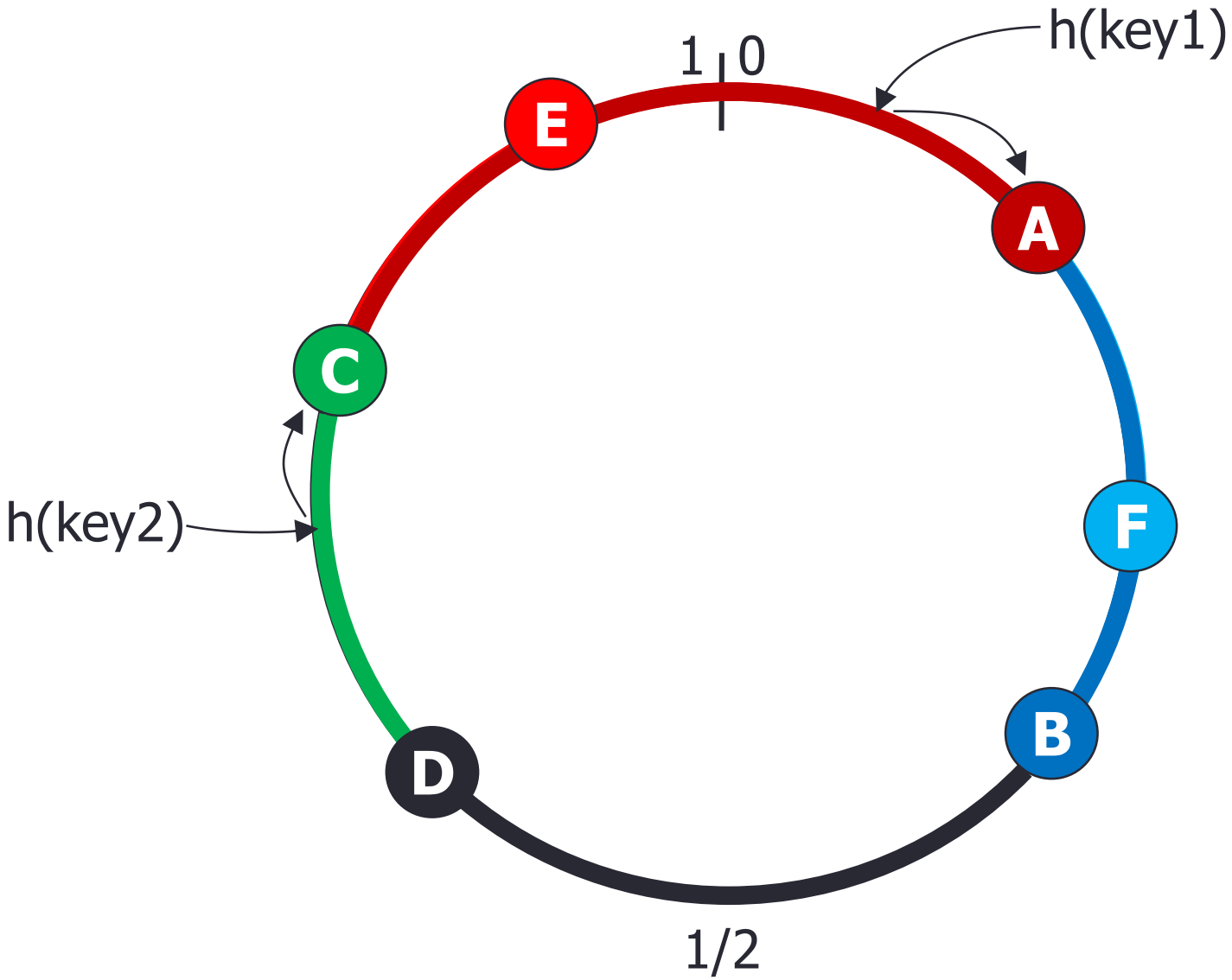
- Background
- Data Model
- Architecture
- Implementation
- Facebook Inbox Search
- Conclusion
- Discussion

Architecture

- **Partitioning**
  - How data is partitioned across nodes to achieve high scalability
- **Replication**
  - How data is duplicated across nodes to achieve high availability and durability
- **Cluster Membership**
  - How nodes are added/deleted to the cluster
- **Bootstrapping**
  - How nodes start for the first time

Partitioning

- **Partitions data through consistent hashing**
  - Order preserving hash function
- **Nodes are structured in a ring**
  - Each node receives a value representing its position
  - Hashing rounds off after certain value to support ring structure
- **Hashed value of data key determines data position in the ring**
  - Walking the ring clockwise first node will be the coordinator node


1 | 0
h(key1)
E
A
C
F
h(key2)
B
D
1/2

Partitioning

Consistent Hashing:

Advantage	**Drawbacks**
• Nodes departure/arrival only affects the immediate neighbors	• Non-uniform load distribution • Unaware of the node performance heterogeneity

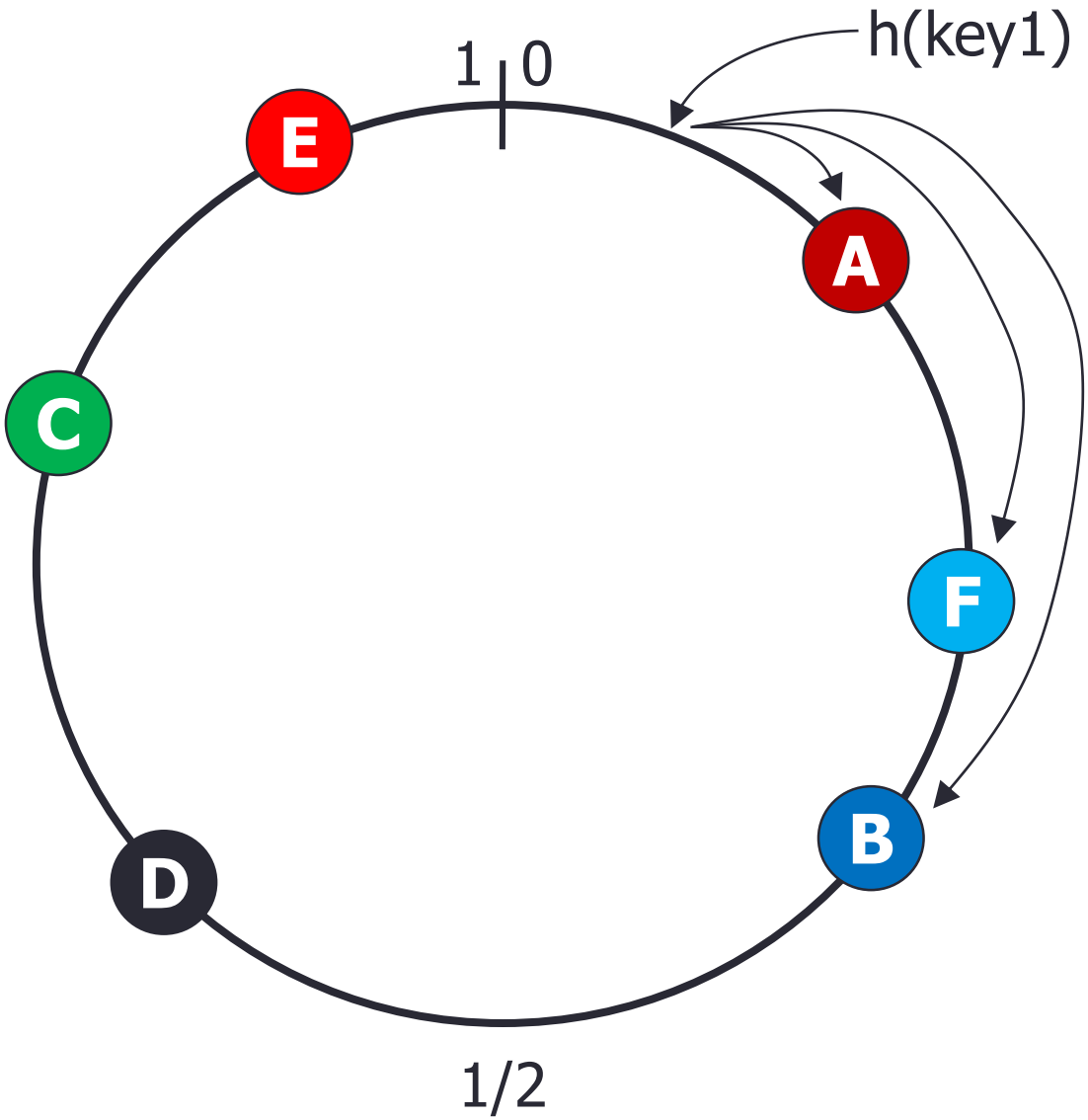
Solutions
1- Assigning nodes to multiple positions in the ring (Virtual Nodes) 2- Analyze nodes' load information and change the nodes' location **Cassandra uses the second approach**

Replication

- **Data is replicated at N (replication factor) hosts**
- **Cassandra Replication Policies:**

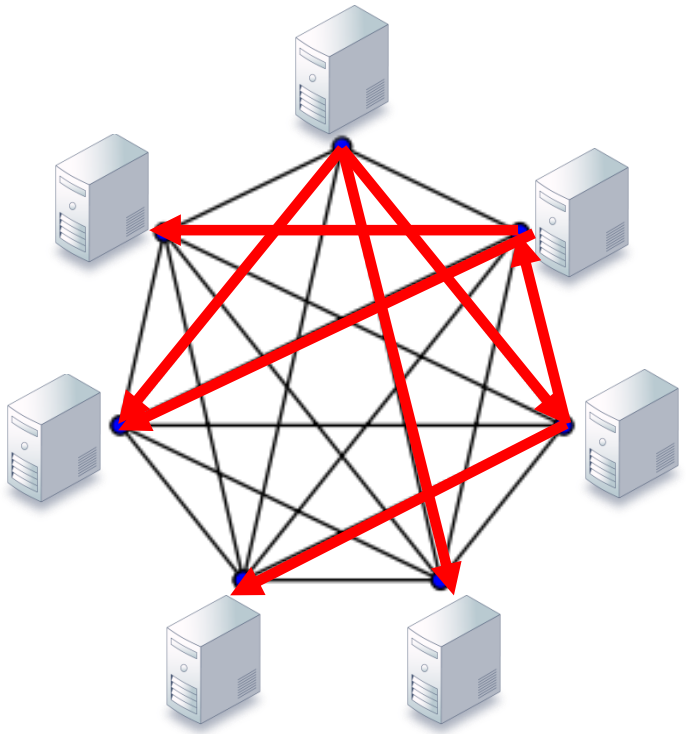
Rack Unaware	replicate data at N-1 successive nodes after its coordinator
Rack Aware	‘Zookeeper’ chooses a leader which tells nodes the range they are replicas for
Datacenter Aware	similar to Rack Aware but leader is chosen at Datacenter level instead of Rack level

Rack Unaware Replication

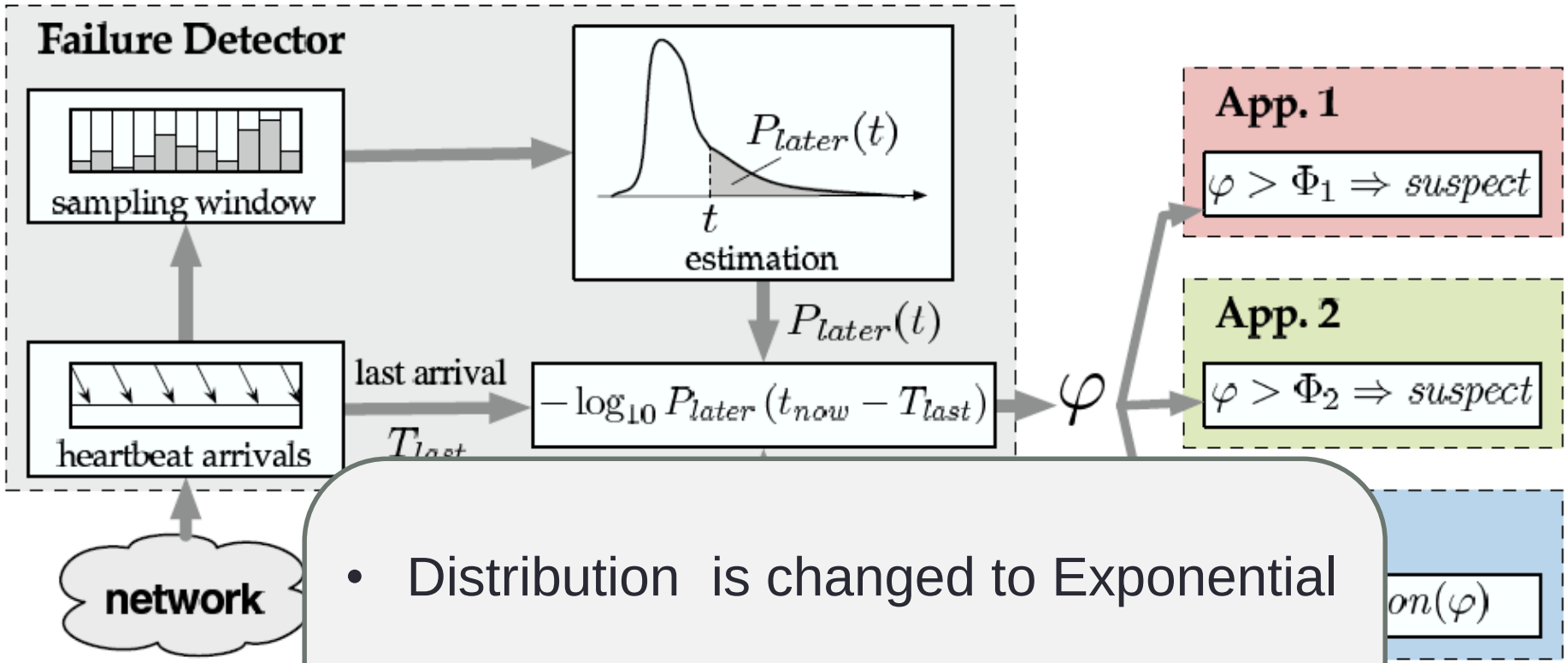


Membership

- Cluster membership:
  based on an anti-entropy Gossip based mechanism (called Scuttlebutt)
- Gossip:
  - Network Communication protocols based on real life rumor spreading
- Anti Entropy Gossip:
  - Repairs replicated data by comparing and reconciling differences



Failure Detection



- Distribution is changed to Exponential
- Cassandra is the first implementation of Accrual Failure Detection in a gossip based configuration

Bootstrapping and Scaling

- Bootstrapping
  - A node starts for the First time
  - Receives a random token for its position in the ring
  - Persists mapping locally and in Zookeeper
  - Gossips the token information to others
- Scaling:
  - Joining node receives a token to help an overloaded node
  - Overloaded node copy a related range of data to the new node

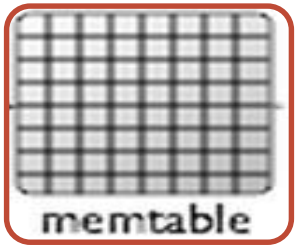
Agenda

- Background
- Data Model
- Architecture
- Implementation
- Facebook Inbox Search
- Conclusion
- Discussion

Persistence Components

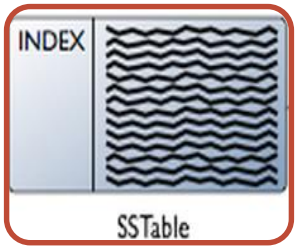
Commit Log File

- Is an append only file
- Has a dedicated local disk



MemTable

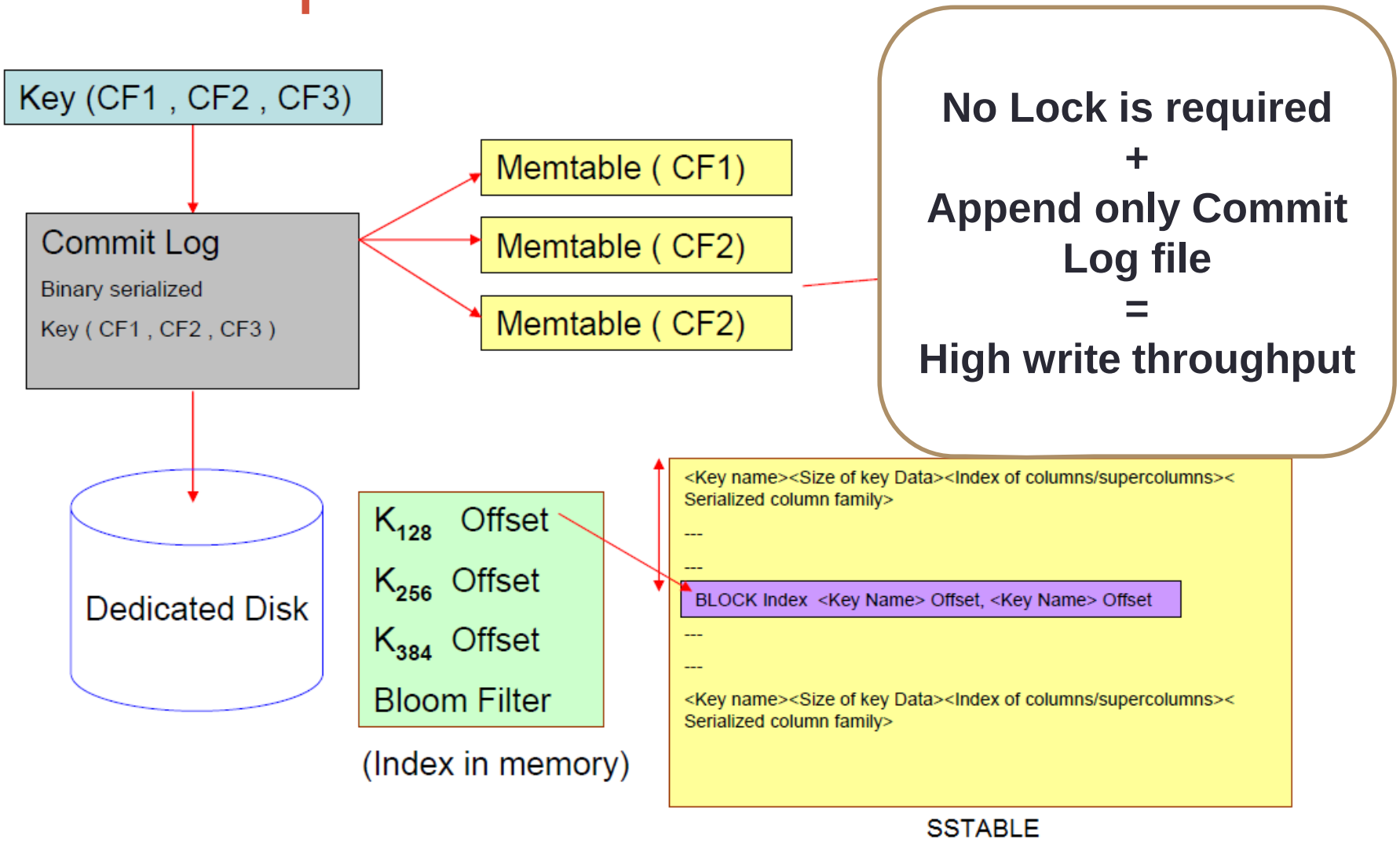
- In-memory data structure
- One memtable for each column family



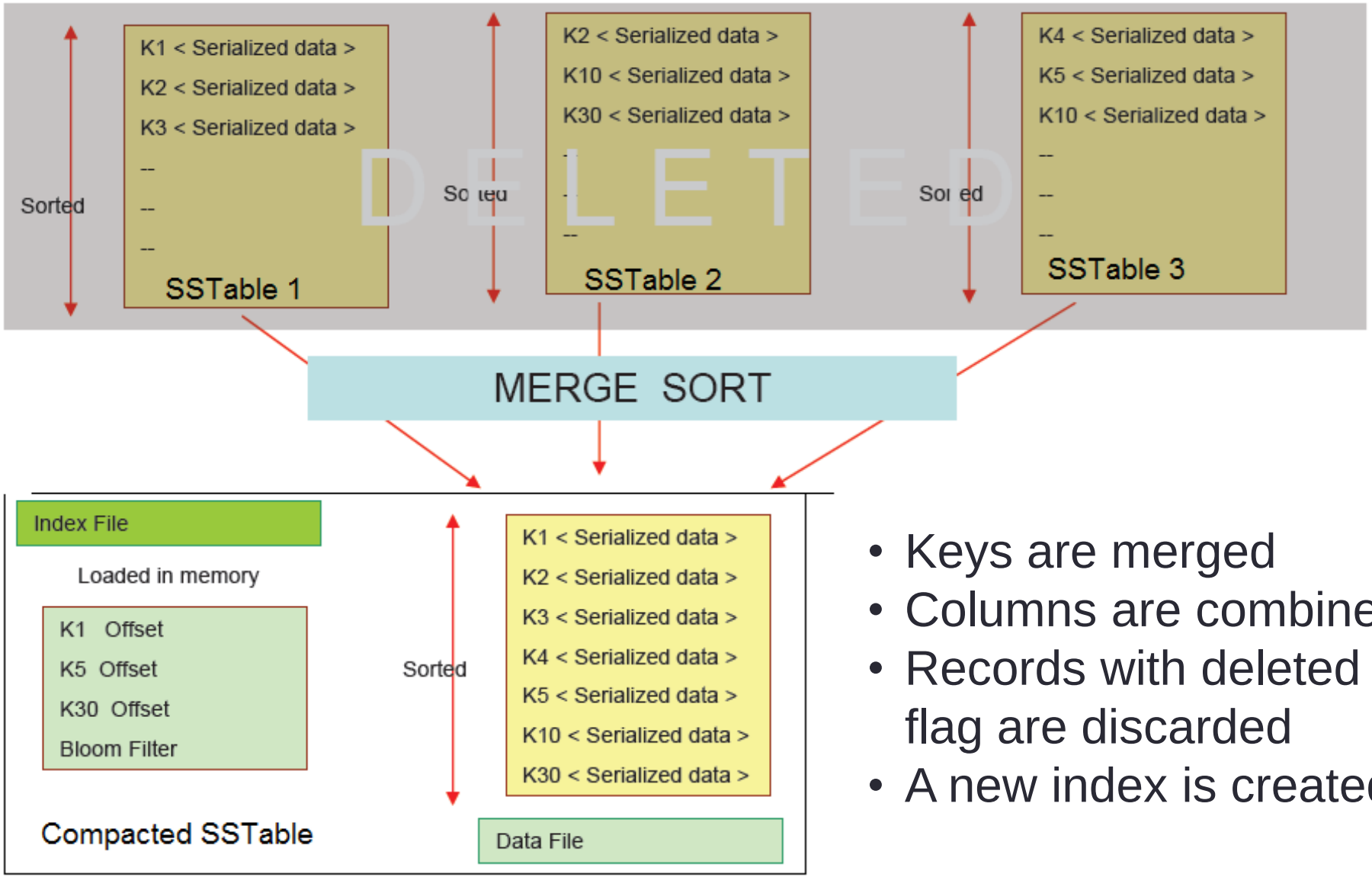
SSTable(Sorted Strings Table)

- On-disk data structure
- Unchangeable once written

Write Implementation

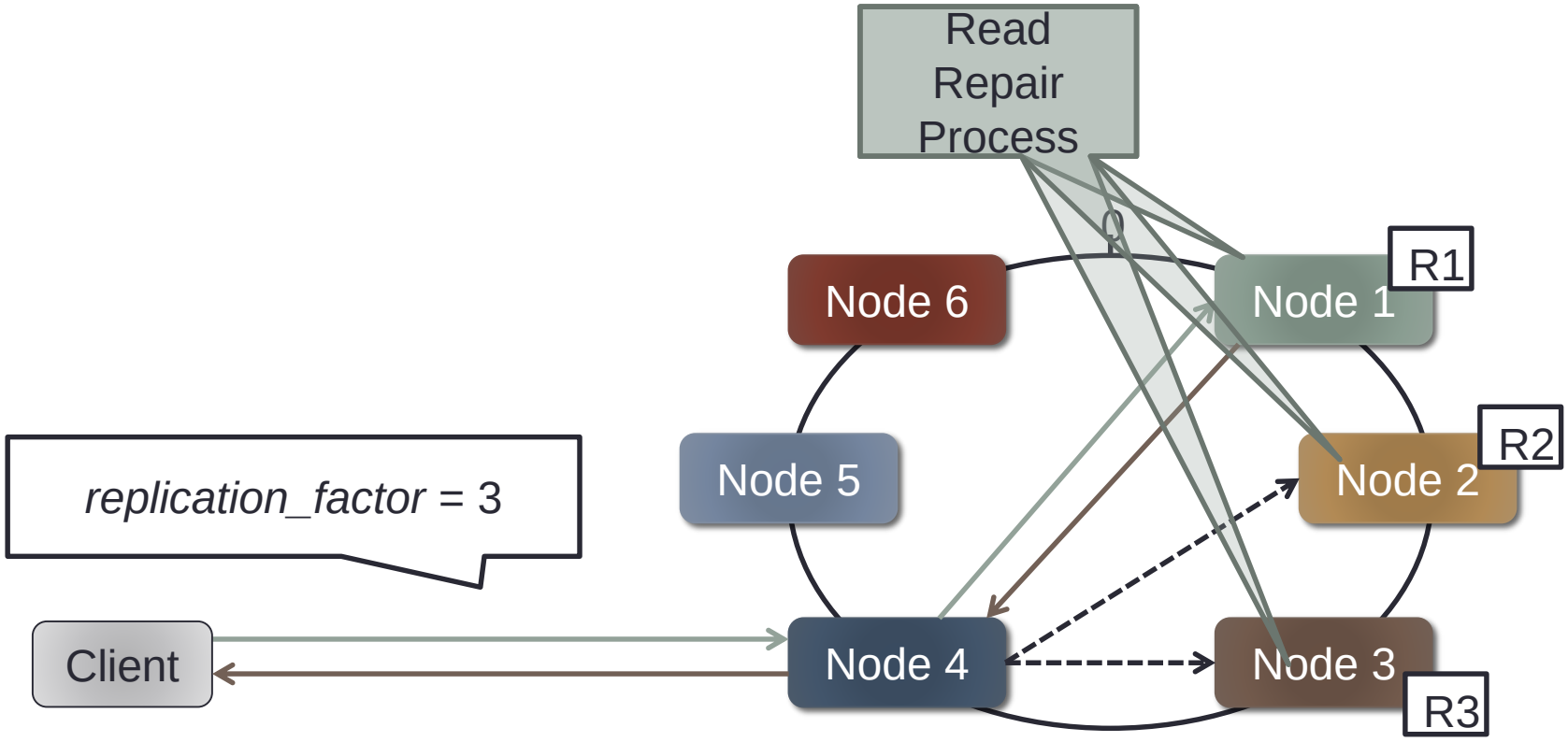


Compaction Process



- Keys are merged
- Columns are combined
- Records with deleted flag are discarded
- A new index is created

Read Repair



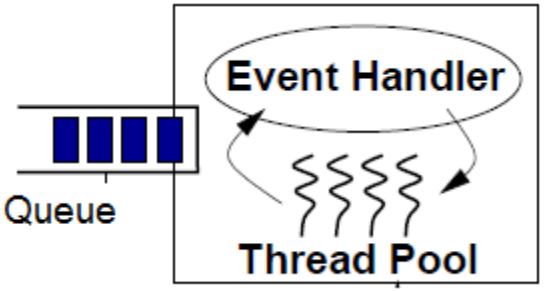
CONSISTENCY LEVEL = ONE

Read Path

- Search the in-memory data structures
- Disk lookup is required when data is not in memory
- Each SSTable has a Bloom filter and index file
- Bloom filter is consulted to reduce the number of files for search
- Index is used to access the right chunk of disk

Staged Event-Driven Architecture (SEDA)

- SEDA is a concurrency model consisting of some stages.
- Stage is a basic unit of work
  - a queue,
  - an event handler
  - a thread pool



- Operations transit from one stage to the next.
- Each stage can be handled by a different thread pool-> High performance

Commit Log Maintenance

- The commit log is rolled out after its size reaches a threshold
- Each commit log contains a header to show whether each updated memtable persisted
- The header will be checked to make sure that all data is persisted before purging the commit log

Implementation modules

- Cassandra Modules in each node:
  - Partitioning
  - Cluster membership and failure detection
  - Storage engine
- Modules are implemented in Java
- The architecture is based on SEDA

Agenda

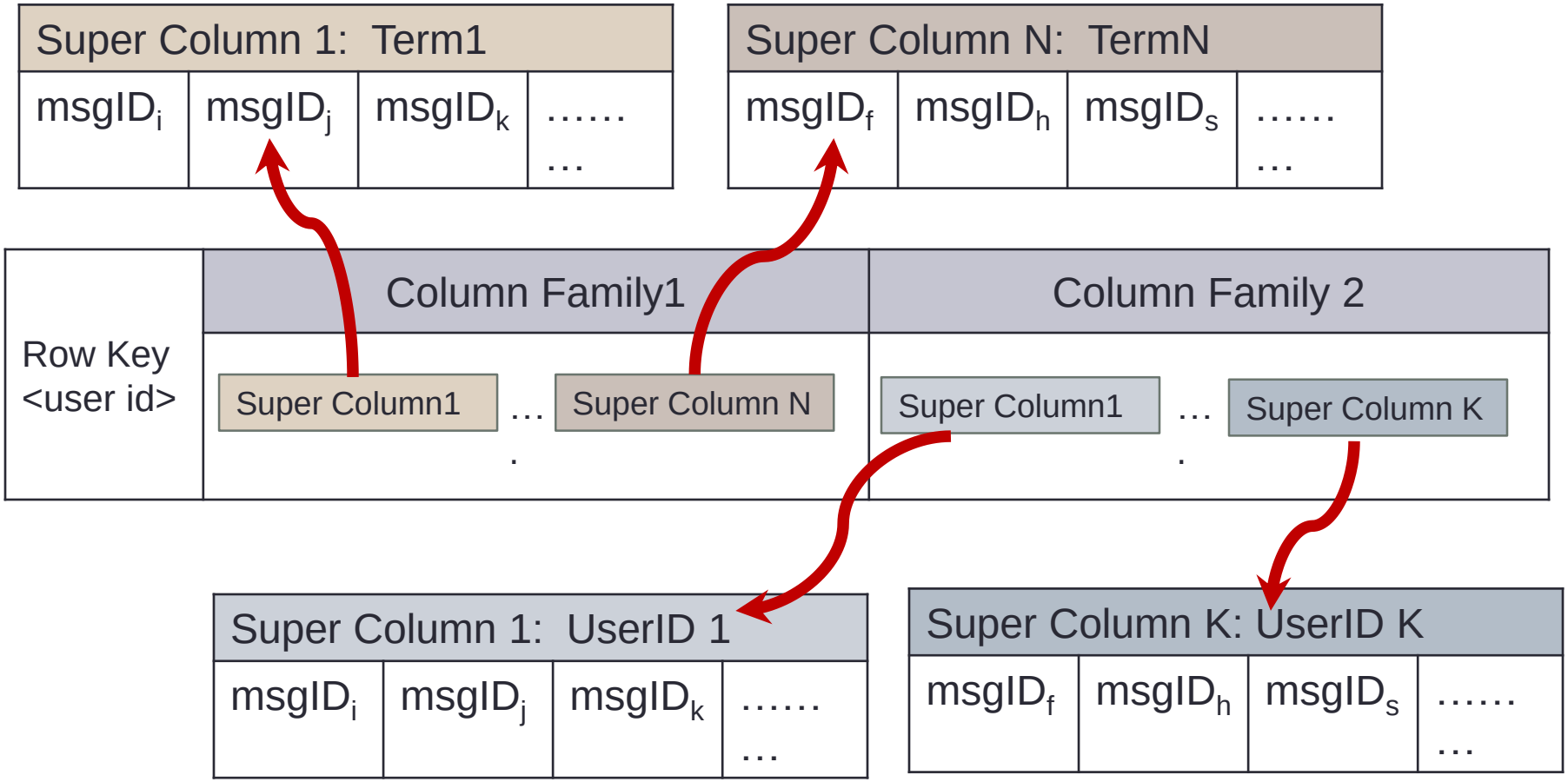
- Background
- Data Model
- Architecture
- Implementation
- Facebook Inbox Search
- Conclusion
- Discussion

Facebook Inbox Search

- Two types of search:
  - Term search
    - All messages containing a specific term
  - Interaction search
    - All messages between two specific users

Currently term search does not work in Facebook !

Inbox Search Schema



Agenda

- Background
- Data Model
- Architecture
- Implementation
- Facebook Inbox Search
- Conclusion
- Discussion

First release vs 2.0

Feature	Change
Data Model	• No super column • Terminology is changed
API	• CQL offered
Partitioning	• Virtual nodes added
Replication	• All replicas are not equal for reads • No Zookeeper
Persistence	• Automatic memtable sizes • Automatic flush policy

Point to be considered …..

- **Cassandra power :**
  - High write throughput
  - No single points of failure
  - Linear scalability
- **Cassandra weakness** :
  - No Join
  - Atomic only per row
  - Thinking in Reverse for data modeling

References

[1] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *SIGOPS Oper. Syst. Rev.*, 44(2): 35-40, 2010.

[2] T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, “Solving Big Data Challenges for Enterprise Application Performance Management,” *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 1724–1735, Aug. 2012.

[3] E. Hewitt, *Cassandra: The Definitive Guide*, 1 edition. Sebastopol, CA; Köln u.a.: O’Reilly Media, 2010.

[4]http://www.cse.buffalo.edu/~okennedy/courses/cse704fa2012/6.1-Cassandra.ppt

[5] N. Hayashibara, X. Defago, R. Yared, and T. Katayama, “The " PHI Accrual Failure Detector,” in *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*, Washington, DC, USA, 2004, pp. 66–78.

[6] http://www.odbms.org/wp-content/uploads/2013/11/cassandra.pdf

[7]http://vanets.vuse.vanderbilt.edu/dokuwiki/lib/exe/fetch.php?media=teaching:cassandra_presentation_final.pptx

[8] http://www.eecs.harvard.edu/~mdw/talks/seda-sosp01-talk.pdf

[9]http://www.datastax.com/documentation/articles/cassandra/cassandrathenandnow.html

Thank you !

Q&A

Discussion

- Security
  - Many NoSQL databases like Cassandra, do not have security features similar to what we see in GRANT/REVOKE operations in relational databases.
  - The products themselves say that they are only designed to be accessed from “trusted environments”.
  - Isn’t this a restriction?
- Calculus based
  - The relational model has a strong Relational Algebra base.
  - There is not such a foundation for NoSQL databases.
  - Can NoSQL databases be a lasting solution despite this fact?