CS535 Big Data

4/29/2019 Week 15-A Sangmi Lee Pallickara

PART B. GEAR WORKSHOP IV
SCALABLE DATA STORAGE,
RETRIEVAL, AND ANALYTICS

Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs535

FAQs

- Quiz 7: 5/1
  - Workshop IV

- Term project
  - Final Report
  - Software (software demonstration)
  - Presentation

- Three criteria for TP grading
  - Relevance
  - Completeness
  - Challenge

Today’s topics

- Cassandra
  - Replication, virtual nodes, and gossip protocol

This material is built based on


http://www.cs.colostate.edu/~cs535

Spring 2019 Colorado State University, page 1
Avalanche Analysis for hash functions

- Indicates how well the hash function mixes the bits of the key to produce the bits of the hash
- Whether a small change in input causes a significant change in the output
- Whether or not it achieves “avalanche”
  - \( P(\text{Output bit } i \text{ changes | Input bit } j \text{ changes}) = 0.5 \)
  for all \( i \)
- If we keep all of the input bits the same, and flip exactly 1 bit
  - Each of our hash function’s output bits changes with probability ½

- The hash is “biased”
  - If the probability of an input bit affecting an output bit is greater than or less than 50%
  - Large amounts of bias indicate that keys differing only in the biased bits may tend to produce more hash collisions than expected.

2. RandomPartitioner

- RandomPartitioner was the default partitioner prior to Cassandra 2.1
- Uses MD5
- 0 to 2^127 - 1

3. ByteOrderPartitioner

- This partitioner orders lexically by key bytes
- The ordered partitioner allows ordered scans by primary key
  - If your application has user names as the partition key, you can scan rows for users whose names fall between Jake and Joe

- Disadvantage of this partitioner
  - Difficult load balancing
  - Sequential writes can cause hot spots
  - Uneven load balancing for multiple tables

Replication

- Provides high availability and durability
- For a replication factor (replication degree) of \( N \)
  - The coordinator replicates these keys at \( N-1 \) nodes
  - Client can specify the replication scheme
  - Rack-aware/Rack-unaware/Datacenter-aware

- There is no master or primary replica
- Two replication strategies are available
  - SimpleStrategy
  - Use for a single data center only
  - NetworkTopologyStrategy
  - Multi-data center setup

GEAR Workshop IV | Scalable Data Storage, Retrieval, and Analytics
Apache Cassandra
Data Replication

http://www.cs.colostate.edu/~cs535
1. What is m (number of bits to represent identifiers)?
2. What are the ranges that each node is responsible for?

Nodes are at 2, 6, 10, 13, 15

1. Assume that this Cassandra provides the replication factor of 3 and the replicas are placed in the clockwise direction. What will be the ranges of ids that each node is responsible for?

Nodes are at 2, 6, 10, 13, 15

1. Now, there is a new node (color-coded as black) joining this cluster and the id will be 4. Design your data movement for this join process.

Nodes are at 2, 6, 10, 13, 15

1. Assume that this Cassandra provides the replication factor of 3 and the replicas are placed in the clockwise direction. What will be the ranges of ids that each node is responsible for?

Nodes are at 2, 6, 10, 13, 15

1. Now, there is a new node joining this cluster at 4. Design your data movement for this join process.

Nodes are at 2, 6, 10, 13, 15
1. **SimpleStrategy**

- Used only for a **single data center**
- Places the first replica on a node determined by the partitioner
- Places additional replicas on the next nodes clockwise in the ring without considering topology
- Does not consider rack or data center location

2. **NetworkTopologyStrategy** (1/3)

- For the data cluster deployed across multiple data centers
- This strategy specifies how many replicas you want in each data center
- Places replicas in the same data center by **walking the ring clockwise until it reaches the first node in another rack**
- Attempts to place replicas on distinct racks
- Nodes in the same rack (or similar physical grouping) often fail at the same time due to power, cooling, or network issues.

3. **NetworkTopologyStrategy** (2/3)

- When deciding how many replicas to configure in each data center, you should consider:
  - being able to satisfy reads locally, without incurring cross data-center latency
  - failure scenario
- The two most common ways to configure multiple data center clusters
  - **Two replicas in each data center**
    - This configuration tolerates the failure of a single node per replication group and still allows local reads at a consistency level of **ONE**.
  - **Three replicas in each data center**
    - This configuration tolerates either the failure of one node per replication group at a strong consistency level of **LOCAL_QUORUM** or multiple node failures per data center using consistency level **ONE**.

4. **NetworkTopologyStrategy** (3/3)

- **Asymmetrical replication groupings**
  - For example, you can maintain 4 replicas
    - Three replicas in one data center to serve real-time application requests
    - A single replica elsewhere for running analytics.

---

**What are Vnodes?**

- With consistent hashing, a node owns exactly one contiguous range in the ring-space
- Vnodes change from one token or range per node, to many per node
- Within a cluster these can be randomly selected and be non-contiguous, giving us many smaller ranges that belong to each node

---

[GEAR Workshop IV Scalable Data Storage, Retrieval, and Analytics Apache Cassandra Virtual Node]

http://www.cs.colostate.edu/~cs535
Virtual nodes are Randomly selected and Non-contiguous

Advantages of Vnodes

- Example
  - 30 nodes and replication factor of 3
  - A node dies completely, and we need to bring up a replacement
  - A replica for 3 different ranges to reconstitute
  - 1 set of the first natural replica
  - 2 sets of replica for replication factor of 3
  - Since our RF is 3 and we lost a node, we logically only have 2 replicas left, which for 3 ranges means there are up to 6 nodes we can stream from
  - With the setup of RF3, data will be streamed from 3 other nodes total

If vnodes are spread throughout the entire cluster
  Data transfers will be distributed on more machines

Restoring a new disk with vnodes

- Process of restoring a disk
  - Validating all the data and generating a Merkle tree
  - This might take an hour
  - Streaming when the actual data that is needed is sent
  - This phase takes a few minutes

- Advantage of using Vnodes
  - Since the ranges are smaller, data will be sent to the damaged node in a more incremental fashion
  - Instead of waiting until the end of a large validation phase
  - The validation phase will be parallelized across more machines, causing it to complete faster

The use of heterogeneous machines with vnodes

- Newer nodes might be able to bear more load immediately
  - You just assign a proportional number of vnodes to the machines with more capacity
  - e.g. If you started your older machines with 64 vnodes per node and the new machines are twice as powerful, give them 128 vnodes each and the cluster remains balanced even during transition

GEAR Workshop IV | Scalable Data Storage, Retrieval, and Analytics
Apache Cassandra
Gossip (Internode communications)
Use of Gossip in Cassandra

- Peer-to-peer communication protocol
  - Periodically exchange state information about nodes themselves and about other nodes they know about.
  - Every node talks to up to three other nodes in the cluster.
  - A gossip message has a version associated with it.
    - During a gossip exchange, older information is overwritten with the most current state for a particular node.

What is gossip?

- Broadcast protocol for disseminating data
- Decentralized, peer-to-peer networks
- ‘epidemic’
- Fault tolerant

- Epidemic broadcast protocol provides a resilient and efficient mechanism for data dissemination.
- Cassandra uses gossip for peer discovery and metadata propagation.

Why Gossip for Cassandra?

- Reliably disseminate node metadata to peers
  - Cluster membership
  - Heartbeat
  - Node status
  - Each node maintains a view of all peers.

What gossip is NOT for in Cassandra?

- Streaming
- Repair
- Reads/write
- Compaction
- Hint
- CQL query parsing/execution

Data structure

- HeartBeatState
- ApplicationState
- EndpointState
  - Wrapper of a heartbeat state and a set of application state.
## HeartBeatState
- **Generation**
  - Timestamp that generated when the process was launched
- **Heartbeat**
  - Periodically update monotonically increasing value

## Application state
- (enum_name, value, version)
  - Contained as a map in `EndpointState` per peer

## ApplicationState enum
- **DC/RACK**
  - Where you are
- **SCHEMA**
- **LOAD**
  - Updated every 60 seconds
- **SEVERITY**
  - I/O load
- **STATUS**

## Status (AppState)
- **Bootstrap**
  - For new nodes
- **Hibernate**
- **Normal**
- **Leaving/Left**
- **Removing/Removed**

## Gossip messaging
- Every second, each node starts a new round
- Peer selection (1-3 peers)
  - Live peer
  - Seed (maybe)
  - Unreachable peer (maybe)

## Gossip Exchange
- **SYN/ACK/ACK2**
  - Similar to TCP 3-way handshake
  - Add anti-entropy to gossiping

http://www.cs.colostate.edu/~cs535
**SYN: GossipDigestSynMessage**  
- Initiator sends a digest of all the nodes it knows about to a peer  
- (ipAddr, generation, heartbeat)

**ACK: GossipDigestAckMessage**  
- Peer receives GossipDigestSynMessage  
- Sort gossip digest list according to the difference in max version number between sender’s digest and own information in descending order  
- Handle those digests first that differ mostly in version number  
- Produces a diff and sends back an ACK  
- Diff contains  
  - Map of APPStates (for any node) that the peer has which the initiator does not  
  - Digest of nodes (and their corresponding metadata) which a peer needs from an initiator

**ACK2: GossipDigestAct2Message**  
- Initiator receives ACK  
- Applies anyAppState and sends back an ACK2  
- ACK2 has a map of AppStates which the peer does not have

**AppState Reconciliation**  
- Generation  
- Heartbeat  
-AppState based on comparing version

---

http://www.cs.colostate.edu/~cs535
Reconciliation example

<table>
<thead>
<tr>
<th></th>
<th>gen:1234</th>
<th>gen:1234</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>hb: 994</td>
<td>status: normal {4}</td>
</tr>
<tr>
<td>B</td>
<td>gen:2345</td>
<td>hb: 17</td>
</tr>
<tr>
<td>C</td>
<td>gen:5555</td>
<td>hb: 1111</td>
</tr>
<tr>
<td>D</td>
<td>gen:2222</td>
<td>hb: 4444</td>
</tr>
</tbody>
</table>

Messaging summary

- Each node starts a gossip round every second
- 1-3 peers per round
- 3 messages passed
- Constant amount of network traffic

Practical implications

- Who is in the cluster?
- How are peers judged UP or DOWN?
- When does a node stop sending a peer traffic?
- When is one peer preferred over another?
- When does a node leave the cluster?

Cluster membership

- Gossip with a seed upon startup
- Learn about all peers
- Gossip
- Lather, rinse, repeat

UP/DOWN?

- Local to each node
- Not shared via gossip
- Determined via heartbeat

Failure Detection

- Glorified heartbeat listener
- Records timestamp when heartbeat update is received for each peer
- Keeps backlog of timestamp intervals between updates
- Periodically checks all peers to make sure that we’ve heard from them recently

http://www.cs.colostate.edu/~cs535
UP/DOWN affects
- Stop sending writes (hints)
- Sending reads
- Gossip
  - It is down
  - This node is treated as an unavailable node
- Repair/stream sessions are terminated

What if a peer is really slow?
- Peer is NOT marked down
  - We will try to avoid it

Dynamic “Snitch”
- Determine when to avoid a slow node
- Scoring peers based on response times
  - Scores recalculated every 100ms (default)
  - Scores reset every 10m (default)

How do nodes leave?
- STATUS = LEAVING
  - Stream data
  - Stream hints
  - STATUS = LEFT, expiryTime

Decomission
- STATUS = LEAVING
- Stream data
- Stream hints
- STATUS = LEFT, expiryTime

Remove node
- STATUS = REMOVING
- Rebalance cluster
  - Notify coordinator
- Delete hint
- STATUS = REMOVED, expiryTime
Replace node

- Cassandra.replace_address
- "shadow gossip"
- Take tokens/hostID (hints)
- Check that previous owner hasn’t gossiped
- Stream data

"Assassinate!"

- Managing hanging non-functional nodes
- unsafeAssassinateEndpoint(ipAddr)
  - Use with caution
  - Forces change to peer

Local persistence: Read Operation

- First queries the in-memory data structure
- Disk lookup
  - Look-up a key
  - To narrow down the lookup process
    - a bloom filter is stored in each data file and memory