PART 2. SCALABLE FRAMEWORKS FOR REAL-TIME BIG DATA ANALYTICS

2. SERVING LAYER: CASE STUDY - CASSANDRA

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

Today's topics

- Cassandra
- Partitioning

FAQs

- Midterm (Nov. 16)
  - 9:30AM-10:45AM CSB425
- PA2 demo: 11/16-11/17
  - www.SignUpGenius.com/go/30E0E4ECAE29A6F0D-pa2

SYN: GossipDigestSynMessage

- Initiator sends a digest of all the nodes it knows about to a peer

  - HeartBeatState
    - Generation stays the same when server is running and grows every time the node is started
    - Each node has one HeartBeatState associated with it

ACK: GossipDigestActMessage

- 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
CS535 Big Data
Fall 2017 Colorado State University
http://www.cs.colostate.edu/~cs535

11/14/2017
Sangmi Lee Pallickara
Week 13 - A

**GossipDigestActMessage**

- For more details on GossipDigestActMessage
  - The EndPointState in 10.0.0.2 will be
    - EndPointState 10.0.0.1
    - HeartBeatState: generation 1259909635, version 324
    - ApplicationState "load information": 5.2, generation 1259909635, version 45
    - ApplicationState "bootstrapping": bxLpassF3XD8Kyks, generation 1259909635, version 56
    - ApplicationState "normal": bxLpassF3XD8Kyks, generation 1259909635, version 87
  - EndPointState 10.0.0.2
    - HeartBeatState: generation 1259911052, version 63
    - ApplicationState "load information": 2.7, generation 1259911052, version 2
    - ApplicationState "bootstrapping": AujDMftpyUvebtnn, generation 1259911052, version 31
    - ApplicationState "normal": AujDMftpyUvebtnn, generation 1259911052, version 62
  - EndPointState 10.0.0.3
    - HeartBeatState: generation 1259812143, version 2142
    - ApplicationState "load information": 16.0, generation 1259812143, version 1803
    - ApplicationState "normal": W2U1XYUC3wMppcY7, generation 1259812143, version 6
  - EndPointState 10.0.0.4
    - HeartBeatState: generation 1259912942, version 0
    - ApplicationState "load information": 6.7, generation 1259912942, version 3
    - ApplicationState "normal": bj05IVc0lvRXw2xH, generation 1259912942, version 7

- # The GossipDigestActMessage from 10.0.0.2 is
  - 10.0.0.1:1259909635:324
  - 10.0.0.3:1259912238:0
  - 10.0.0.4:1259912942:0
  - 10.0.0.2: [ApplicationState "normal": AujDMftpyUvebtnn, generation 1259911052, version 62], [HeartBeatState, generation 1259911052, version 63]

- # GossipDigestSynMessage From 10.0.0.1
  - 10.0.0.1:1259909635:325
  - 10.0.0.2:1259911052:61
  - 10.0.0.3:1259912238:5
  - 10.0.0.4:1259912942:18

**ACK2: GossipDigestAct2Message**

- Initiator receives ACK
- Applies any AppState and sends back an ACK2
- ACK2 has a map ofAppState which the peer does not have

- # The GossipDigestAct2Message from 10.0.0.1 is
  - 10.0.0.1: [ApplicationState "load information": 5.2, generation 1259909635, version 45], [ApplicationState "bootstrapping": bxLpassF3XD8Kyks, generation 1259909635, version 56], [ApplicationState "normal": bxLpassF3XD8Kyks, generation 1259909635, version 87], [HeartBeatState, generation 1259909635, version 324]
  - 10.0.0.3: [ApplicationState "load information": 12.0, generation 1259912238, version 3], [HeartBeatState, generation 1259912238, version 3]
  - 10.0.0.4: [ApplicationState "load information": 6.7, generation 1259912942, version 3], [ApplicationState "normal": bj05IVc0lvRXw2xH, generation 1259912942, version 7], [HeartBeatState: generation 1259912942, version 18]

**AppState Reconciliation**

- Generation
- Heartbeat
- AppState based on comparing version

**Reconciliation example**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>gen:1234</td>
<td>Hb: 994</td>
</tr>
<tr>
<td></td>
<td>Status: normal {4}</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>gen:1234</td>
<td>Hb: 990</td>
</tr>
<tr>
<td></td>
<td>Status: normal {4}</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>gen:2345</td>
<td>Hb: 10</td>
</tr>
<tr>
<td></td>
<td>Status: bootstrap {1}</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>gen:2345</td>
<td>Hb: 17</td>
</tr>
<tr>
<td></td>
<td>Status: normal {2}</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>gen:5555</td>
<td>Hb: 1111</td>
</tr>
<tr>
<td></td>
<td>Status: normal {5}</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>gen:2222</td>
<td>Hb: 4444</td>
</tr>
<tr>
<td></td>
<td>Status: normal {3}</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>gen:3333</td>
<td>Hb: 11</td>
</tr>
<tr>
<td></td>
<td>Status: normal {3}</td>
<td></td>
</tr>
</tbody>
</table>

**Inter-node Communication 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

UP/DOWN affects
- Stop sending writes/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
- It is initiated by the decommission tool
  - You can use it when the node is up and running

Remove node
- STATUS = REMOVING
- Rebalance cluster
  - Notify coordinator
- Delete hint
- STATUS = REMOVED, expiryTime
- Node tool "removenode" can initiate
  - It is used usually when the node is down
  - If removenode does not work, use "assassinate"

Replace node
- Cassandra.replace_address
- "shadow gossip"
  - Allows viewing complete gossip information without modifying or augmenting it
  - Provides time to adjust to a new node with a new UUID
- 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

Failure detection: Φ Accrual Failure Detector
- Φ Accrual Failure Detector
- Φ Accrual Failure Detection does not emit a Boolean value stating a node is up or down
  - Emits a value which represents a suspicion level for each of the monitored nodes
  - Threshold is defined as Φ
- Dynamically adjusts to reflect network and load conditions at the monitored nodes
Failure detection (2/2)

- Given some threshold $\Phi$, and assuming that we decide to suspect a node $A$.
  - e.g. when $\Phi = 1$, then the likelihood that we will make a mistake is about 10%.
  - The likelihood is about 1% with $\Phi = 2$.
  - The likelihood is about 0.1% with $\Phi = 3$.

- Every node maintains a sliding window of inter-arrival times of gossip messages from other nodes in the cluster.

Bootstrapping

- When a node joins the ID ring, the mapping is persisted to the disk locally and in Zookeeper.
  - Then the token information is gossiped around the cluster.

- With bootstrapping, a node joins with a configuration file that contains a list of a few contact points.
  - Seeds of the cluster.

- Seeds can be provided by a configuration service (e.g. Zookeeper).