Global Leader Election in Distributed Architecture

Dharmendra Kushwaha, NEC
Agenda

• Why Leader
• Why Leader Election
• Leader Election Mechanisms
• Requirements
• Challenges
• Solution
Why Leader

• An Organizer for some tasks
• Keeps nodes in synch.
• Responsible for controlling any changes in system.
Why Leader Election

• Leader Election: A process of designating a single process as the organizer, coordinator, initiator or sequencer of some task distributed among several nodes or services.
  – The existence of a centralized controller greatly simplifies process synchronization
  – However, if the central controller breaks down, the service availability can be limited
  – The problem can be alleviated if a new controller (leader) can be chosen.
Leader Election Mechanism

- Bully Algorithm
- Ring Algorithm
- ..
- ..
Requirements
Requirements

Node with Global responsibility

GNS File System

Nodes with Local responsibilities

Distributed service Infrastructure
Recovery and responsibility re-distribution.
Challenges
Time synchronization

- System timing may not be in sync
- Gaps can be more than a minutes.
Split Brain Problem

- May be more than one node can start behaving like leader.
- May be no leader.
- Most of the solutions usage third party components.
GFS accessibility is must
Solution
How encounter those challenges

- Leader must have both network & filesystem connectivity.
- Election coordinator for time sync.
- GNS FileSystem(GFS) to handle split brain.
Leader Prerequisite

Leader Node

Keep updated

Leader Info

GFS

Leader File

Heartbeat

Node 1

Node 2

Node 3

Node 4
Leader connectivity with other nodes in system is less than \( N/2 \), it will initiate an election.

Node-x is not able to send heartbeat to global leader, it will check the last update time of Leader info file. If file is not updating, node will assume that leader is down.
Election configs

- **Lock File:**
  - LOCK_<GL version>_<_election Round>

- **Election File:**
  - ELECT_FILE

Node Id 1<16 byte>: nodeData<16 byte>
Node Id 2<16 byte>: nodeData<16 byte>
...
Node Id n<16 byte>: nodeData<16 byte>
Election configs..

- **Timeouts:**
  - $T_{el}$: Election Timeout
  - $T_{p_{-}el}$: Participant nodes timeout
  - $T_{e\_file\_wait}$: Election file write timeout
  - $T_{notify}$: Notification time out
**Election Algorithm**

**GL_Election:**

1. Get Election version.
2. Check for Lock file
   a. If lock file exist:
      i. Is file’s Leader version is older.
         1. Remove existing lock file.
         2. Create new lock file.
            a. If success, proceed as election manager. Otherwise, proceed as participant.
      ii. Is Leader version is same.
         1. Is file’s election round is older.
            a. Update Lock file.
               i. If success, proceed as election manager. Otherwise, proceed as participant.
         2. Otherwise, proceed as participant.
      iii. Otherwise, proceed as participant.
Election Algorithm:

Proceed as Election Manager:

1. Calculate file size.
2. If election participant file exist on GFS.
   a. Remove existing file.
3. Create a temp file and fill it with null bytes by its size.
4. Move temp to Election participant File
5. Write own Id and data(i.e. connection count) on specific offset in file.
6. Start Timer $T_{el}$, and wait for this timeout.
7. Disable write After time $T_{el}$ (move election participant file to temp elect file)
8. Elect Leader on the basis of data(i.e max connectivity & higher id).
9. Notify newly elected leader to take the ownership, and Start timer $T_{notify}$.
10. If got response from leader within $T_{notify}$ timeout or leader info file is updated.
    a. Leader election done, Clean lock & election participant file.
Otherwise call GL_Election.
Election Algorithm..

**Proceed as Election Participant:**

1. If no Election participant file exist on GFS.
   a. If Temp Election participant file exist.
      i. Election window closes, goes out of election.
   b. Otherwise, Wait for Te_file_wait time.
   c. After Te_file_wait time if Election Participant file is still not created.
      i. Call GL_Election.

2. Write own id & connectivity count in election participant file

3. Start Timer Tp_el and wait.

4. If during timer Tp_el got Leader notification
   a. Stop timer.
   b. If Leader file updated.
      i. Leader election done.
      ii. Update own status and exit.
   c. Otherwise:
      i. Take Leader ownership (i.e. update Leader file)

Respond to election manager and exit.
Election Algorithm: Flow Chart

Start

Identify Election version

Does any lock file exist on GFS

Yes

Remove Lock File & election participant file

No

Is lock file version older

Yes

Create Lock file.

Done

Yes

Proceed as election coordinator

No

Proceed as Participant
Election Algorithm: Flow Chart

1. Proceed as Election coordinator
2. Calculate File Size
3. Create a temp Election participant File
4. Fill this temp file with NULL bytes
5. Move temp to Election participant File
6. Write own Id and connection count on specific offset

7. Start Timer Tel
8. Disable write After time Tel (move election participant file to temp elect file)
9. Elect Leader on the basis of max connectivity & higher id.
10. Notify to leader
11. Start timer Tnotify

12. Got Notification or GL file updated
   - Yes: Leader elected, Clean lock & election participant file
   - No: X
13. Y
14. Stop
Election Algorithm: Flow Chart..

1. Proceed as Election Participant
2. Election participant file exist?
   - Yes: Write own id & connectivity count in election participant file
   - No: Temp election participant file exist?
3. Temp election participant file exist?
   - Yes: Goes Out of Election and wait for Tp_el
   - No: Wait for Te_tie_wait time
4. Is Election Participant file created after Te_file_wait?
   - Yes: Stop timer
   - No: Got GL Notification?
5. Got GL Notification?
   - Yes: GL file updated?
   - No: GL file updated on GFS?
6. GL file updated?
   - Yes: Leader Elected
   - No: Take GL ownership
7. Leader Elected?
   - Yes: Respond notification
   - No: X
Thank You

Q&A?

Dharmendra.Kushwaha@gmail.com