# Table of Contents

# Simulator Implementation

## Motivations behind Canssandra simulator

Cassandra is a distributed system. This nature makes it harder for developers and testers to rapidly develop and test applications without proper hardware and network setups. It is particular hard for testing, when we want to know how system behaves under stressed network conditions, hardware changes and failures, and many other scenarios. The motivations of the building a Cassandra simulator is to provide a tool to simplify procedures to achieve above needs.

## Design Requirement:

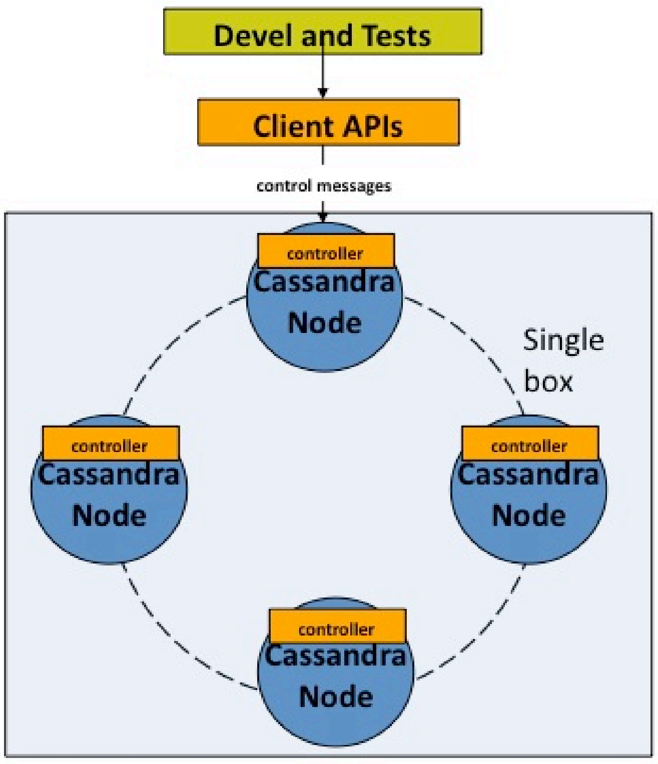
- Simulator is a separate project from Canssandra main project, and does not insert and/or modify code in Cassandra main project.
- Only support Linux OS. Other OS will be for future considerations.
- Being able to simulate multiple node operations within single linux box
- Being able to simulate Hardware configuration changes and failures
- Being able to simulate network conditions
- Being able to carry out read/write operations at the simulated environment
- Being able to provide APIs to access logging information

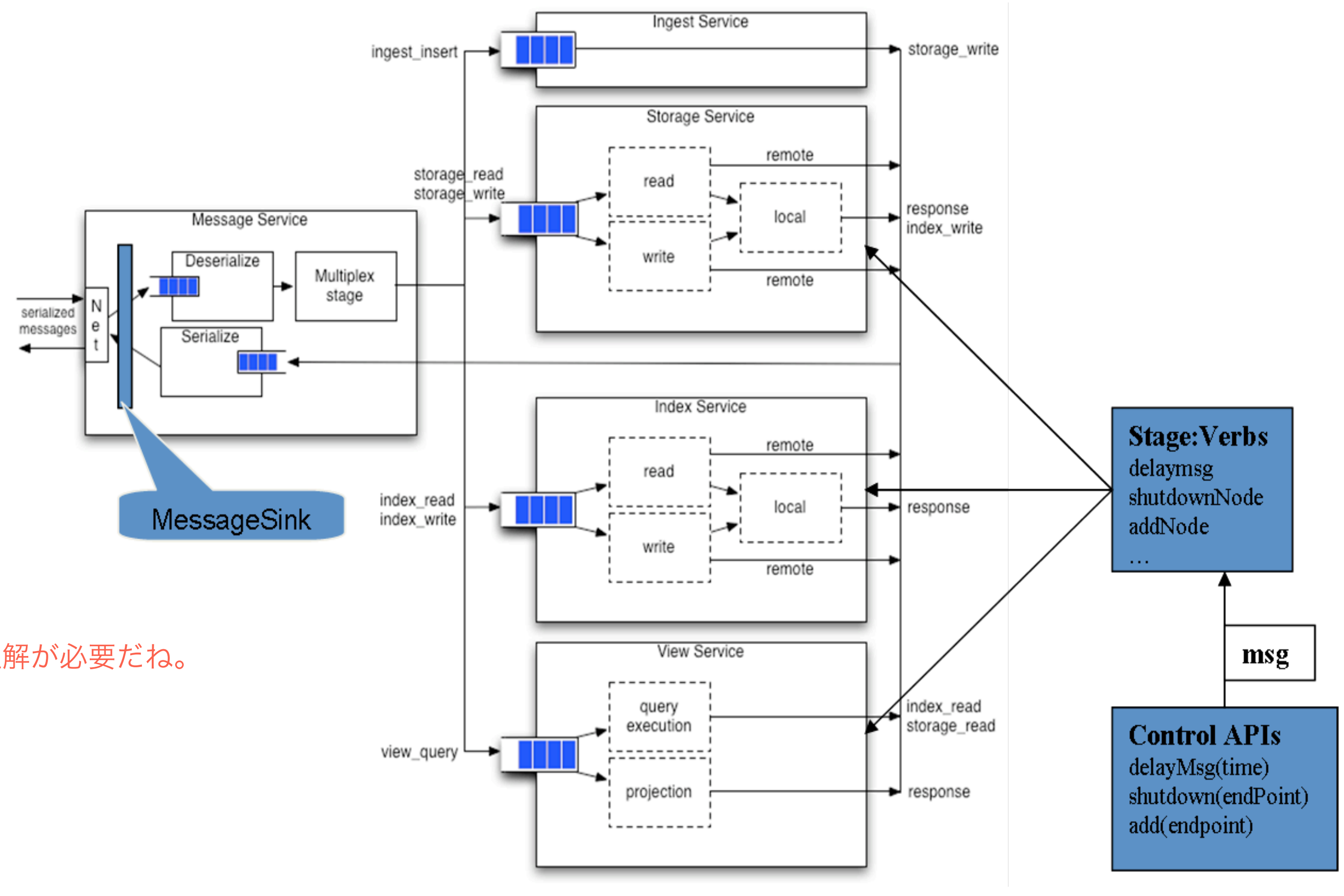
## Design Scope:

| | |
|---|---|
| Hardware simulations | • start simulator by given number of nodes<br>• shutdown one or more nodes<br>• simulate node temporarily out-of-service<br>• add new nodes to the ring |
| Network simulations | • network latency<br>• network hiccups<br>• network failures |
| R/W simulations | • data localities<br>• data consistency |
| statistics and loggin | • log analysis |

## General Architecture:

- Simulator has two main components: controller and client
- Controller runs as an service in every cassandra node so it is part of the ring in the system. It's main function is to receive the control messages and execute control message commands (verbs) and return results to the client.
- The client is not part of the cassandra node thus it is not part of the ring in the system. It has a set of APIs ranging from start/stop nodes, and network simulations. Users call these APIs will actually send control messages to the controller of given endpoint for desired operations.





SEDAの理解が必要だね。

General Architecture:

# Quick Setup Guide:

## Step 1: import source code into a Eclipse project:

- Note: simulator only being tested on RHEL 4 and 5 boxes
- check out simulator source code from repository
- Under project root folder, enter "mvn eclipse:eclispe"
- using Eclipse, do project -> import and browse to simulator root folder. You will get a eclipse java project for simulator

## Step 2: Add dependencies to the project:

- Right click on simulator project, choose "build path" and then "configure build path"
- Add apache-cassandra-incubator-0.4.0-dev.jar (which is built via Cassandra project)
- Add all the librarires from Cassandra/lib folder:

```
antlr-3.1.3.jar,
commons-collections-3.2.1.jar,
google-collect-1.0-rc1.jar,
jline-0.9.94.jar,
clhm-20090629.jar,
commons-javaflow-1.0-SNAPSHOT.jar,
groovy-1.5.6.jar,
junit-4.6.jar,
log4j-1.2.15.jar,
commons-cli-1.1.jar,
commons-lang-2.4.jar,
high-scale-lib.jar,
libthrift.jar
```

## Step 3: Configurations:

- Make following folders:( this is based on yahoo standard)
    - ♦ mkdir **/home/y/conf/simulator**
    - ♦ mkdir **/tmp/cassandra**
- make soft link between resource folder:

```
 "ln -s <projectRoot>/src/main/resources/com/yahoo/content/assembly/simulator/utility \
/home/y/conf/simulator/bin"
```

-
    - ♦ ⚠ Note: If you want to use another directory, other than **/home/y/conf/simulator**, you can modify the source code in **org.apache.cassandra.simulator.common.StaticLib.java**.
- Edit **/etc/sudoers** file: This script will call `sudo ifconfig` for establishing virtual interface, which requires you to enter password. If you're a lazy people like me, you can enter a line like in `/etc/sudoers` with `sudo visudo` command,

  ```
  SYSAD ALL= NOPASSWD:/sbin/ifconfig
  ```
- configure **anode.ini** file:
    - ♦ Locate **anode.ini** file in **src/main/resource** folder
    - ♦ Set the correct value for **CassandraHome** and **SimulatorHome**. Read the comments for how to set the value.

## Step 4: Run the test:

- There are number of unit tests in **src/test/java** folder. All tests are written in TestNG. It is suggest to install TestNG eclipse plug-in.
- Locate Sprint3DemoTest in **org.apache.cassandra.simulator.testcases** package.
- The demo will do the followings:
    1. start a ring with 5 nodes and 3 replica factors
    2. query a ring topology, the node with actually IP will always be named as node 0. The numbering will be clock wise.
    3. write specified data into the node given
    4. query data locations ants replication nodes
    5. read data from a non-data node
    6. entering message delay state where every message will be blocked for 2 sec before sending out. (simulate network hiccups)
    7. query data again
    8. exit message delay state
    9. shutdown simulator and clean up.

## Step 5: Trouble shooting:

- Virtual nodes folder can be found in **/tmp/cassandra/nodeX** where X =0, 1, 2, etc. You will find conf, logs and other working folders here.
- ifconfig command will show all the virtual IPs created
- if nodes are not starting properly, go to **/tmp/cassandra/nodeX/conf** folder, and start a node manually by `sudo ./cassandra -f`. You should be able to debug from there. Most likely the classpath is not set correctly. The virtual node conf files are generated based on **anode.ini** file. So, you need to go back to set the right values.

## Use cases:

### A). Bring up multiple nodes within same host

やはり、1 node / 1 JVM

- One jvm for each node
- Using virtual IPs for each node
- Using fixed TCP (messaging) and UDP(gossiping) ports for each node
- How to configure multiple node with virtual IP on Linux
- Perl script to create multiple nodes
- Java Wrapper of Perl Script

### B). Simulate message communications

- Implement MessageSink interface to simulate following scenarios:
    - ♦ Simulate message with write operation (normal situation)
    - ♦ Simulate message with read operation (normal situation)
    - ♦ Simulate message with write operation (delay for a given time period)
    - ♦ Simulate message with read operation (delay for a given time period)
    - ♦ Simulate message with write operation (drop completely)
    - ♦ Simulate message with read operation (drop completely)
    - ♦ Simulate message with write operation (overload)
    - ♦ Simulate message with read operation (overload)
- The implementation for MessageSink in Simulator

### C). Provide APIs to control the state of the nodes

- To take down a node on a temporary base from the ring
- To take down a node permanently from the ring
- To add a node into the ring
- (take down node with data, or writing data into a disabled nodes, etc. should belongs to test cases, not there)

### D). Provide APIs to access data information from different node

- To read data from any given node in the ring
- To read data from a failed node in the ring
- To write data to any given node in the ring
- To write data to any failed node in the ring

### E). Provide utility to parse log files

- To parse the log files generated from various nodes
- To filter/sort log message
- To save in CSV file
- Usage of Log Parser

Last revised by: weili99 on 18 Nov 2009

ATTACHMENT LIST

%ATTACHMENTLIST{format="$fileName?"}%