Lab 2: Causal Consistency

CSE 513 - Distributed Systems
The Pennsylvania State University
Fall 2019

Submitted On: 11/17/2019
Submitted By: Sahithi Rampalli (svr46)
Gulsum Gudukbay (gxg5138)

System Description and Protocol Specifications

In order to demonstrate causal consistency, we simulate three datacenters with a key-value store and a few clients. The datacenters communicate among themselves. This implementation uses a Java API - Java Remote Method Invocation (Java RMI) to communicate among peers and tracker. Java RMI is based on the idea of Remote Procedure Calls (RPCs). It uses the Java Remote Method Protocol (JRMP) and the transport layer protocol - TCP/IP.

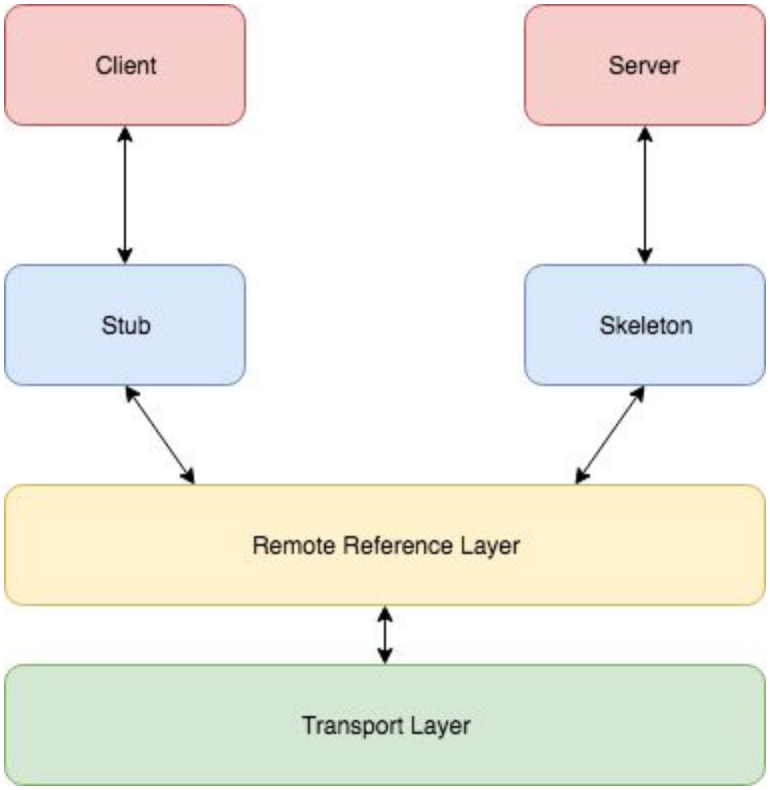

Figure 1 - Client-Server Diagram for Java RMI

As soon as a server is up, it registers itself in a “rmiregistry” with a unique name. By registering itself in the rmiregistry, it makes its remote interfaces available for the rest of the world. Any client/server which would like to communicate with a server can lookup the server in the remote rmiregistry, create an instance of the remote interface (called stub) and invoke the required method. On the server side, the interface is accessed (called skeleton), the method is executed and output is sent back via RMI.

Implementation

Functionalities of the Server/Data Center

- **RegisterReply** - This method, registers a client and creates an empty dependency list for the client.
- **getKey**- This method is invoked when a client tries to read a specific key. This function returns the value of the key if present. It also marks a dependency for this client by adding the latest version of the key to the dependency list.
- **putKeyValue** - This method is invoked by the client when a client tries to write a specific key, value. The writes are replicated to other data centers by passing the current dependency list of the client as a parameter. The dependency list is cleared and updated. The Lamport clock is updated and the key, value is committed.
- **replicatedWrite** - When a write is performed at a datacenter, it requests the other datacenters to replicate. Each datacenter commits the write only if a dependency check is satisfied. A dependency check is satisfied if the versions of the keys in the received dependency list have already arrived at the datacenter. If a check fails, the replicated write is queued and tried again when another replicated write is invoked.

Functionalities of the Client

- **RegisterRequestToServer** - When a client is up, it registers itself in the Server.
- **Read** - Requests the server for the value of the desired key.
- **Write** - Requests the server to put the key value in the key-value store.

Program Structure Description

Java Interfaces

- IServer - Remote Interface for the Server.
- IClient - Remote Interface for a Client.

Java Classes

- Server - Implements IServer remote interface.
- Client - Invokes server methods.
- DepNode - A node structure for dependencies of format:
- NodeStruct - A node structure for clients and servers of format:
- RW - A replicated write structure for the function call replicatedWrite of format:
- Version - A Version structure for storing versions of keys of format:

Implemented Functionalities

- All the client and server methods described above are implemented for the purpose of demonstration of Causal Consistency using Lamport clocks.
- The second example described in the tutorial is simulated for testing.

Simulations

- Server
    - All the servers connect to each other via rmiregistry only after all the servers are up.
    - No client connects before all the servers are connected.
- Client
    - We simulated the second bigger example given in the tutorial.
    - All the clients' messages are stored in a list.
    - Before each message, an interrupt is generated.
    - When the user wants the client to send its next message, they should just press the return key.
    - When the read/write operation is fully performed (read's have a String returned and writes are fully committed), a message will be output.

Sample Outputs

Servers

```
~/Causal-Consistency-Distributed-Systems/classes$ java Server s1 127.0.0.1 2014 20
s3 127.0.0.1 2016
s2 127.0.0.1 2015
num servers: 3
connected to servers: 2
connected to all datacenters
Server ready
Registered client A
Received client A's write request for key 1 with value X_A
```

```
~/Causal-Consistency-Distributed-Systems/classes$ java Server s2 127.0.0.1 2015 20
s3 127.0.0.1 2016
s1 127.0.0.1 2014
num servers: 3
connected to servers: 2
connected to all datacenters
Server ready
Registered client B
Registered client D
inside the first sleep
At datacenter s2 received replicated write for key: 1 and value: X_A and new version:
<1, s1> at time 22:32:43.895
Attached dependency list: []
The Dependency List for B: []
The Dependency List for D: []
Committed key: 1 value: X_A and new version: <1, s1>
At datacenter s2 received replicated write for key: 2 and value: Y_C and new version:
<1, s3> at time 22:32:49.161
Attached dependency list: []
Committed key: 2 value: Y_C and new version: <1, s3>
Received client B's read request for key 1
Received client B's read request for key 2
Received client B's write request for key 3 with value Z_B
Added key: 3 and value: Z_B and the size of keyvstore is: 3
Received client D's write request for key 3 with value Z_D
Added key: 3 and value: Z_D and the size of keyvstore is: 3
```

```
~/Causal-Consistency-Distributed-Systems/classes$ java Server s3 127.0.0.1 2016 20
s1 127.0.0.1 2014
s2 127.0.0.1 2015
num servers: 3
connected to servers: 2
connected to all datacenters
Server ready
Registered client C
inside the second sleep
Received client C's write request for key 2 with value Y_C
Added key: 2 and value: Y_C and the size of keyvstore is: 1
The Dependency List for C: [<2, <1, s3>>]
At datacenter s3 received replicated write for key: 3 and value: Z_B and new version:
<2, s2> at time 22:33:06.373
Attached dependency list: [<1,<1,s1>>,<2,<1,s3>>]
Cannot commit as <1, <1, s1>> is not received
Cannot commit adding key: 3, value: Z_B and new version: <2, s2> to pending list.
At datacenter s3 received replicated write for key: 3 and value: Z_D and new version:
<3, s2> at time 22:33:40.012
Committed key: 3 value: Z_D and new version: <3, s2>
At datacenter s3 received replicated write for key: 1 and value: X_A and new version:
<1, s1> at time 22:34:03.922
Committed key: 1 value: X_A and new version: <1, s1>
At datacenter s3 recalling replicated write for key: 3 and value: Z_B and new version:
 <2, s2>
At datacenter s3 received replicated write for key: 3 and value: Z_B and new version:
<2, s2> at time 22:34:35.910
Attached dependency list: [<1,<1,s1>>,<2,<1,s3>>]
Committed key: 3 value: Z_B and new version: <2, s2>
```

Clients

```
~/Causal-Consistency-Distributed-Systems/classes$ java Client A 127.0.0.1 2020 s1 127.
0.0.1 2014
A ready
A sent register request for server s1
-
Client A successfully wrote to key 1 the value X_A
```

```
~/Causal-Consistency-Distributed-Systems/classes$ java Client B 127.0.0.1 2021 s2 127.
0.0.1 2015
B ready
B sent register request for server s2
-
Client B read key 1 and value is: X_A
-
Client B read key 2 and value is: Y_C
-
Client B successfully wrote to key 3 the value Z_B
```

```
~/Causal-Consistency-Distributed-Systems/classes$ java Client C 127.0.0.1 2022 s3 127.
0.0.1 2016
C ready
C sent register request for server s3
-
Client C successfully wrote to key 2 the value Y_C
```

```
~/Causal-Consistency-Distributed-Systems/classes$ java Client D 127.0.0.1 2024 s2 127.
0.0.1 2015
D ready
D sent register request for server s2
-
Client D successfully wrote to key 3 the value Z_D
```

Compile and Run Instructions

Software Requirements

Java Version 8 or greater is required for this application.

Compile Instructions

cd src && mkdir classes
javac -Xlint -d classes *.java

Run Instructions

cd classes

Server

java Server <server_id> <server_ip> <server_port> <waiting_time_in_seconds_for_all_servers>
java Server s1 127.0.0.1 2014 20
java Server s2 127.0.0.1 2015 20
java Server s3 127.0.0.1 2016 20

Client

java Client <client_id> <client_ip> <client_port> <server_id> <server_ip> <server_port>
java Client A 127.0.0.1 2020 s1 127.0.0.1 2014
java Client B 127.0.0.1 2021 s2 127.0.0.1 2015
java Client C 127.0.0.1 2022 s3 127.0.0.1 2016
java Client D 127.0.0.1 2024 s2 127.0.0.1 2015