Distributed Pub-Sub System with Replication

Ajinkya Prabhu - amprabhu **Athilesh Arputham - aarputha**

Introduction

Our aim in this project is to create a publisher-subscriber system which will follow the naming server and storage server architecture used in the lab 3 distributed file system project.

Design

Some definitions

1. Topic - A location/URI where messages will be posted by publishers and consumed by subscribers.
2. Publisher/Producer - will generate messages and will follow a schema per topic
3. Subscriber/Consumer - will consume messages from a topic for later use

Our Design includes two kinds of servers.

1. Naming Server - Will hold the mappings between topics and Servers. This will then be the bookkeeper and maintain a copy of all the servers assigned to a specific topic and store their IP address and port.
2. Topic Server - will hold the actual data related to the topic such as the timestamp of each message and current producers and consumers

The topic servers will first register with the naming server with the topic they intend to serve and their IP address and port.

The Topic servers will replicate their data across multiple instances so as to maintain fault tolerance in the event a topic server fails and is uncommunicative the client will automatically request the naming server for a new topic server with the same topic and try to recover.

Our initial plan was to have topic servers automatically delete messages, but we found that topic servers can get away without needing to store messages by simply adopting a push model. This has two benefits the onus of starvation is on the publisher and the subscriber can simply idle wait for messages. This greatly simplifies the subsriber design as it is another rpc server with its ip on the topic server. Futhermore a push model can reduce complexity on the topic server architecture by using it as a forwarding server. Replication is also simplified as topic servers only need to know peers who are serving a given topic and only forward published messages to them.

Project Plan and Testing and evaluation

1. Implement the naming server, Server will hold in a key-value store the details mentioned above
2. Implement the Topic server will have a theoretically unlimited buffer of messages and will delete messages in the process as described above.
3. Implement replication of messages between topic servers and forwarding messages to clients on the same topic but different servers
4. Testing with unit tests and regressions tests which check for the following
   - Checkpoint
     - Register on the naming server - 10 - `test_checks_if_topic_server_registers_with_naming_server_on_startup`
     - Naming Server Notifying Topic servers of new peer on topic - 20

   `test_naming_server_communicates_replication_server_to_all_servers`

- Topic server replication - 25

  `test_should_replicate_messages`

- Topic server multiple subscribers - 30

  `test_should_have_multiple_subscribers`

- Topic Server Multiple Pub-Sub and Replication - 35

  `test_should_be_able_handle_multple_publishers_subscribers_and_replication`

- Test Naming server registration RPC - 5

  `test_should_register_topic_servers_with_naming_servers`

- Test naming server start - 5

  `test_start_naming_server`

- Test basic actors pub and sub - 20

  `test_should_setup_all_basic_actors_and_publish_subscribe`

Tooling

Language - We would like to use Rust
Libraries

- tokio for async and threading
- serde for serialization deserialization
- tarpc for RPC communication

Makefile - will include how to build and test our project using the testing framework provided by rust

Evaluation

Testcases points - 165