

Fault Tolerant Server Cluster

This project is a multi-threaded fault tolerant server cluster which provides a serverless Java execution service. The cluster implementation includes a public gateway, leader node, and follower nodes. Clients send java source code to the gateway, and the gateway sends work to the leader node, which then distributed the work among worker nodes in a round robin fashion. System out and System err results of compilation and running of Java code are returned from workers to the client via the gateway node. Leader elections are used, as well as queues, all-to-all heartbeats and gossip, to provide resilience in the face of leader or worker death. HTTP, TCP and UDP protocols are employed as appropriate.

The classes used by this project are as follows:

TCPSender

Used by a server to send TCP messages. Outgoing messages are added to a `LinkedBlockingQueue` which is passed to the constructor, along with a `ConcurrentHashMap` of server ID numbers to their `InetSocketAddress`, and a reference to the server this instance of `TCPSender` is being used by, in order to access any relevant state information, as well as information about other servers, from the server using this `TCPSender`.

TCPReceiver

This class listens on a specified port for incoming TCP messages, and adds them to a server's `LinkedBlockingQueue` of incoming messages.

UDPMessageSender

This class is used to send UDP messages, taken from a `LinkedBlockingQueue` of outgoing messages.

UDPMessageReceiver

This class listens on a specified port for incoming UDP messages. Bodies of incoming messages are split on comma separations. This allows the message to be classified based on certain headers, and action is taken depending on the nature of the incoming message, which may be gossip, a heartbeat or leader election related messages.

HeartbeatScanner

This class constantly loops over a map of server ID numbers to the last time a heartbeat was received by the `UDPMessageReceiver`. If a heartbeat has not been received after ten seconds, the server is assumed dead.

HeartbeatSender

This class sends a heartbeat broadcast for a server every two seconds, as well as the full gossip table of the server to one random other server in the cluster.

JavaRunnerImpl

This class compiles and runs java source code received via an `InputStream`. The `System.out` and `System.err` values are temporarily redirected to `OutputStreams` which can then be used to return the outputs. The

output redirection is synchronized across all `JavaRunnerImpl` instances on a single machine by synchronizing on the `JavaRunnerImpl` class object's monitor.

ClientImpl

This is the client to the server cluster. It sends source code to the public gateway, and returns the response.

Driver

This class starts a cluster of servers locally, and keeps them alive through a `while(true)` loop in the main method.

Gateway

This class implements the server acting as a public gateway, and sends information back and forth between the leader node and clients.

ZooKeeperPeerServerImpl

This class implements the leader and worker nodes, using the previously mentioned classes to communicate, send and receive work, and compile and run Java source code sent from the leader node. Servers elect a leader using instances of the **`ZookeeperLeaderElection`** class.