Application name: Chatchat (chatting system)

Group Name

Grazers (from year 2014)

Student Name and Id

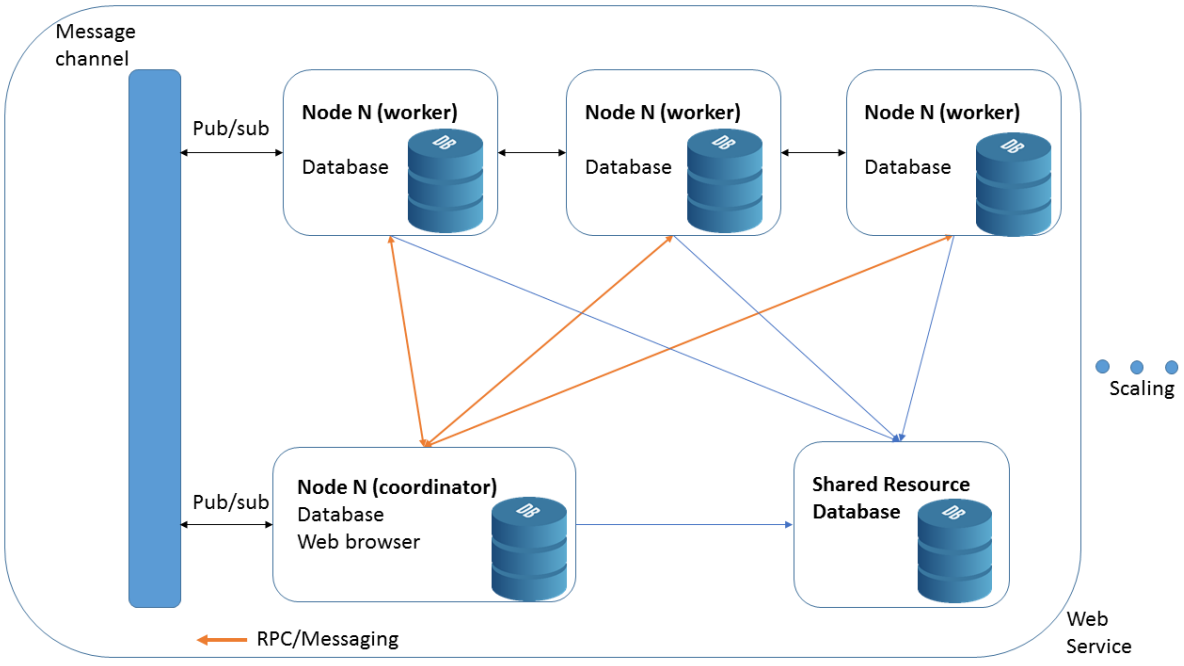
2292191 Haejong Dong

2418111 Alexander Voroshilov

It is a chatting system in which a number of worker nodes and one master node (coordinator) maintain part of the whole system. Master node comprises front-end web interface as well as number of registered worker nodes. Multiple master nodes may serve different front-end web pages concurrently however all data will be synchronized through the shared resource in addition to local replicas.

Basic feature of the application is as follows: 1. Write text on chatting board, 2. Get the list of active users across the system.

System Architecture



Key features(*extra points)

*Fault tolerance and recovery

A number of master nodes (also alias coordinators) can run concurrently. By doing so, system is able to handle in case of one of the master nodes' failure. Once failure occurs, involved worker nodes will automatically migrate to another master node that is active. In a similar sense when new worker node joins to the system, it iteratively searches for only active master nodes. When worker nodes recognize master node stop responding, it start registering to another master node, and behave same as if it was a new worker node.

Also the end-users who interacting with front-end (master node/coordinator) are redirected to another webpage while keeping all data in the shared database when current webpage turn unavailable.

On the other side, if worker node stops responding, master node will try over pre-defined times and then eventually assign the task to another worker node in case of the worker node found failure. Failed nodes are collected by each master node and cleaned in every pre-defined time. All worker node in the list of failed node will be omitted in every worker selection process.

Shared resource with user's session id allows for users to restore all the data from either of shared resource database or worker node database.

*Security

The system leverages AES and RSA (symmetric and asymmetric key) to keep the traveling data secure.

Worker node sends its public key to the master node at the stage of handshake, and the master node handshakes back to the node with data of common-RSA key(for fanout message encryption/decryption) and general server info(ex, server's public key, id) encrypted by public key of the worker node.

Also master server refreshes common-RSA keys for every pre-defined time of interval and broadcast new keys to all worker nodes for updating.

*Communication

List of communication methods the system is leveraging are as follows:

publish/subscribe, RPC, fanout, direct messaging(downstream/upstream).

Master and worker nodes all have the same capacity of communication within the nodes.

When a new node joins or registered node leaves/fails master node notifies to all worker nodes with its info by publish/subscribe (fanout, asynchronous) communication method.

Worker nodes send/receive direct message to/from the master node as needed (direct messaging, asynchronous). Assignment of workload from master node to worker node takes place through RPC method (direct communication, synchronous).

*Distributed synchronization

System selected centralized synchronization algorithm for accessing shared resource that is coordinator's permission based.

When there is a request from the end-user, master node caches it in the queue and later assign the task to a worker node as soon as it takes turn from the queue otherwise master node (coordinator) assigns the task instantaneously with permission to the shared resource to a node to process.

*Naming

Worker nodes are introduced about information including other node's ids and RSA keys when it joins to a master node.

*Consistency and replication

All data from user will be replicated/updated onto shared resource database with session id in which then users can fully restore data from in case of node failure or webpage crash.

Any crashes from either worker or master node is handled gracefully by automated redirect and searching for available nodes which will benefit both entities of end-user and worker node.

Essential data is replicated locally in the node database therefore any requests can be handled gracefully even without permission to access shared resources (weakly-consistent).

System screenshots