**Maintaining Causal Consistency in Replicated Datacenters**

Rahul Titus George – [rtg64@psu.edu](mailto:rtg64@psu.edu); Yashasvi Asthana – [yxa150@psu.edu](mailto:yxa150@psu.edu)

**Files:**

* Datacenter.c – Datacenter implementation
* DBHandler.c – Handles all the read/write request to the Database
* DependencyUtilities – Implementation of all the functions related to dependencies
* Client.c – Client implementation
* Test1.c – Datacenter implementation with hardcoded delays to test the first test case

**Data Structure format for maintaining dependency (DependencyUtilities.h):**

* struct Dependency: holds key, timestamp and datacenter ID
* struct Operation: holds key, new data, and datacenter ID
* struct DependencyList: holds list of dependencies, number of dependencies and the operation to execute if the dependency check satisfies
* Pending queue is a list of DependencyList

**Datacenter Functions implemented:**

* initializeSockets() – initializes the datacenter server sockets and the client socket array to handle multiple client connections
* initMulticastSocket() – initializes the multicast socket which is used to broadcast replicated writes to other datacenters
* readFromDataStore() – reads and returns data corresponding to a given key from the database
* writeToDataStore() – writes data corresponding to a given key to the database
* sendReplicatedWrite() – sends replicated write request to a particular datacenter
* messageHandler() – handles read, write and replicated write requests, and also initiates replicated writes to all the other datacenters
* listening() – ‘select’ based implementation of server that listens to all the read, write or replicated write requests from multiple connections simultaneously

**Dependency Functions implemented:**

* appendClientDependencyList() – creates or appends dependency list of a particular client
* clearDependencyList() – clears dependency list of a particular client
* checkDependency() – checks if the dependencies of a replicated write request satisfy
* appendPendingQueue() – appends the pending queue with a dependency list
* removerFromPendingQueue() – removes a dependency list from the pending queue

**Client Functions implemented:**

* sendWriteRequest() – sends write request to the connected datacenter
* sendReadequestRequest() – sends read request to the connected datacenter
* initializeSocket() – creates the socket and also establishes a connection to the datacenter

**Working:**

* Datacenters listen on their fixed ports waiting for connections and messages.
* A client connects to one of the datacenters (let’s say DC1) to read/write.
* If it’s a write request, DC1 updates its local time, commits the operation, creates an empty dependency list of the client and sends replicated write requests with the current dependency list of that client to all the other datacenters.
* After that DC1 updates the dependency list of that client with the recent write.
* Other datacenters on receiving the replicated write perform a dependency check, if it passes that operation is committed to the database (local clock is updated), and the datacenter issues a dependency check on all the operations in the pending queue, otherwise that operation, with its dependency list, is added to the pending queue to be serviced later.
* If it’s a read request, DC1 updates the dependency list of that client and send a reply message with the data of the requested key.

//ADD SCREENSHOTS HERE

**What is working?**

**Compiling and executing:**

* *make* to compile all the files
* *./Datacenter <PORT>* to start a datacenter with listening port set to ‘PORT’ (for eg. “./Datacenter 1”)
* *./Client <PORT>* to run a client and connect to the datacenter listening on ‘PORT’ (for eg. “./Client 1”)

Create 3 datacenters with PORTS = 1, 2 and 3 to see the working.

Test cases are implemented in Test$.c files. To test, run “./Test$ 1” and create other 2 datacenters using “./Datacenter <PORT>”