Homework 4 5 report

Michael Peterson

My project consists of the following classes:

ServerMain: The starting point of the server, this is where the every client connection is initially established.

JobQueue: This class stores the jobs in and is responsible for adding and removing jobs from the queue. If the queue is full, the addJob() method returns false to the ServerMain class.

ServerLogger: This class is responsible for producing a log of everything that the server does, it precedes every entry with a time stamp. The output is sent to standard out, and optionally to a file specified in the constructor.

SharedQueue: The generic, thread safe queue object that actually stores the jobs. The dequeue() method will cuase the calling thread to wait if there is nothing in the queue, also the enqueue() method will notify all of the threads when a new element has been added.

TestClient: This is the tester primary tester for the project, it is responsible for producing 100 clients to connect to the server in a rapid succession, each of which will send the command "Client Thread # says hi", sleep for one second, then close the connection.

ThreadManager: This class is responsible for periodically checking the threadpool and jobqueue if there is a need to increase, or decrease the size of the pool.

ThreadPool: This class maintains references to all of the server threads that are responsible for handling requests from clients. Separate lists are maintained for keeping track of threads that are running (currently handling a job), and waiting (waiting for a new job from the job queue). These separate lists are used in determining if it is appropriate to change the size of the thread pool.

WorkerThread: The actuall class that handles communication with the client as well as interpreting the commands sent by the client. This class will call ThreadPool.stopPool() when the command "kill" is received.

Testing

The test client class was used to test the dynamic properties of the threadpool. The following log entries show Examples of increasing and decreasing the threadpool based on change in demand:

```
2/19/2016 9:02:56 PM : upper job count threashold reached, increasing threadpool size
2/19/2016 9:02:56 PM : thread pool increasing pool size from 5 to 10
2/19/2016 9:02:56 PM : there are now 50 jobs in the queue
2/19/2016 9:02:56 PM : Worker Thread 5 fetched job (Socket[addr=/127.0.0.1,port=50032,localport=9898]) from
2/19/2016 9:02:56 PM : there are now 49 jobs in the queue
2/19/2016 9:02:56 PM : Worker Thread 7 fetched job (Socket[addr=/127.0.0.1,port=50033,localport=9898]) from
2/19/2016 9:02:56 PM : there are now 48 jobs in the queue
2/19/2016 9:02:56 PM : Worker Thread 6 fetched job (Socket[addr=/127.0.0.1,port=50034,localport=9898]) from
2/19/2016 9:02:56 PM : Worker Thread 7 received command "client thread 6 says hi" from client(Socket[addr=/
2/19/2016 9:02:56 PM : there are now 47 jobs in the queue
2/19/2016 9:02:56 PM : Worker Thread 5 received command "client thread 5 says hi" from client(Socket[addr=/
2/19/2016 9:02:56 PM : there are now 46 jobs in the queue
2/19/2016 9:02:56 PM : Worker Thread 8 fetched job (Socket[addr=/127.0.0.1,port=50036,localport=9898]) from
2/19/2016 9:02:56 PM : Worker Thread 8 received command "client thread 9 says hi" from client(Socket[addr=/
2/19/2016 9:02:56 PM : Worker Thread 8 sending response "Error processing command: client thread 9 says hi"
2/19/2016 9:02:56 PM : Worker Thread 9 fetched job (Socket[addr=/127.0.0.1,port=50035,localport=9898]) from
2/19/2016 9:02:56 PM : Worker Thread 6 received command "client thread 7 says hi" from client(Socket[addr=/
2/19/2016 9:02:56 PM : Worker Thread 6 sending response "Error processing command: client thread 7 says hi"
2/19/2016 9:02:56 PM : Worker Thread 7 sending response "Error processing command: client thread 6 says hi"
2/19/2016 9:02:56 PM : Worker Thread 9 received command "client thread 8 says hi" from client(Socket[addr=/
2/19/2016 9:02:56 PM : Worker Thread 5 sending response "Error processing command: client thread 5 says hi"
2/19/2016 9:02:56 PM : Worker Thread 9 sending response "Error processing command: client thread 8 says hi"
2/19/2016 9:02:57 PM : upper job count threashold reached, increasing threadpool size
2/19/2016 9:02:57 PM : thread pool increasing pool size from 10 to 20
2/19/2016 9:02:57 PM : there are now 45 jobs in the queue
2/19/2016 9:02:57 PM : there are now 44 jobs in the queue
2/19/2016 9:02:57 PM : Worker Thread 10 fetched job (Socket[addr=/127.0.0.1,port=50037,localport=9898]) fro
2/19/2016 9:02:57 PM : there are now 43 jobs in the queue
2/19/2016 9:02:57 PM : Worker Thread 11 fetched job (Socket[addr=/127.0.0.1,port=50038,localport=9898]) fro
2/19/2016 9:02:57 PM : there are now 42 jobs in the queue
2/19/2016 9:02:57 PM : Worker Thread 13 fetched job (Socket[addr=/127.0.0.1,port=50039,localport=9898]) fro
2/19/2016 9:02:57 PM : there are now 41 jobs in the queue
2/19/2016 9:02:57 PM : Worker Thread 12 fetched job (Socket[addr=/127.0.0.1,port=50040,localport=9898]) fro
2/19/2016 9:02:57 PM : there are now 40 jobs in the queue
2/19/2016 9:02:57 PM : lower job count threashold reached, decreasing threadpool size
2/19/2016 9:02:57 PM : thread pool decreasing pool size from 20 to 10
2/19/2016 9:02:57 PM : Worker Thread 4 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 5 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 5 terminating...
2/19/2016 9:02:57 PM : Worker Thread 0 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 0 terminating...
2/19/2016 9:02:57 PM : Worker Thread 16 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 16 terminating...
2/19/2016 9:02:57 PM : Worker Thread 7 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 7 terminating...
2/19/2016 9:02:57 PM : Worker Thread 18 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 18 terminating...
2/19/2016 9:02:57 PM : Worker Thread 13 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 13 terminating...
2/19/2016 9:02:57 PM : Worker Thread 19 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 19 terminating...
2/19/2016 9:02:57 PM : Worker Thread 12 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 12 terminating...
2/19/2016 9:02:57 PM : Worker Thread 11 received interrupt
2/19/2016 9:02:57 PM : Worker Thread 11 terminating...
2/19/2016 9:02:57 PM : Worker Thread 4 terminating...
2/19/2016 9:02:58 PM : lower job count threashold reached, decreasing threadpool size
2/19/2016 9:02:58 PM : thread pool decreasing pool size from 10 to 5
2/19/2016 9:02:58 PM : Worker Thread 15 received interrupt
2/19/2016 9:02:58 PM : Worker Thread 15 terminating...
2/19/2016 9:02:58 PM : Worker Thread 10 received interrupt
2/19/2016 9:02:58 PM : Worker Thread 10 terminating...
2/19/2016 9:02:58 PM : Worker Thread 8 received interrupt
2/19/2016 9:02:58 PM : Worker Thread 8 terminating...
2/19/2016 9:02:58 PM : Worker Thread 9 received interrupt
2/19/2016 9:02:58 PM : Worker Thread 9 terminating...
2/19/2016 9:02:58 PM : Worker Thread 3 received interrupt
2/19/2016 9:02:58 PM : Worker Thread 3 terminating...
```

CapatalizeClient was also used for testing the add, subtract, multiply and divide commands. Notice how the invalid commands “div,5,0” and “no op” are handled gracefully by the server thread.

```
2/19/2016 9:52:29 PM : Worker Thread 4 received command "add,3,4" from client(Socket[addr=/127.0.0.1,port=50283,localport=9898])
2/19/2016 9:52:29 PM : Worker Thread 4 sending response "7" to client Socket[addr=/127.0.0.1,port=50283,localport=9898]
2/19/2016 9:52:32 PM : Worker Thread 4 received command "sub,4,6" from client(Socket[addr=/127.0.0.1,port=50283,localport=9898])
2/19/2016 9:52:32 PM : Worker Thread 4 sending response "-2" to client Socket[addr=/127.0.0.1,port=50283,localport=9898]
2/19/2016 9:52:36 PM : Worker Thread 4 received command "mul,2,4" from client(Socket[addr=/127.0.0.1,port=50283,localport=9898])
2/19/2016 9:52:36 PM : Worker Thread 4 sending response "8" to client Socket[addr=/127.0.0.1,port=50283,localport=9898]
2/19/2016 9:52:47 PM : Worker Thread 4 received command "div,5,0" from client(Socket[addr=/127.0.0.1,port=50283,localport=9898])
2/19/2016 9:52:47 PM : Worker Thread 4 sending response "Error processing command: div,5,0 cannot divide by zero" to client Sock
2/19/2016 9:52:50 PM : Worker Thread 4 received command "no op" from client(Socket[addr=/127.0.0.1,port=50283,localport=9898])
2/19/2016 9:52:50 PM : Worker Thread 4 sending response "Error processing command: no op" to client Socket[addr=/127.0.0.1,port=
```

Here is the log output for the kill command being used.

```
2/19/2016 9:03:42 PM : "kill" command received from client, calling ThreadPool.stopPool()
2/19/2016 9:03:42 PM : Worker Thread 1 closing connection with client Socket[addr=/127.0.0.1,port=50131,localport=9898
2/19/2016 9:03:42 PM : Worker Thread 1 received interrupt
2/19/2016 9:03:42 PM : Worker Thread 6 received interrupt
2/19/2016 9:03:42 PM : Worker Thread 6 terminating...
2/19/2016 9:03:42 PM : Worker Thread 14 received interrupt
2/19/2016 9:03:42 PM : Worker Thread 14 terminating...
2/19/2016 9:03:42 PM : Worker Thread 2 received interrupt
2/19/2016 9:03:42 PM : Worker Thread 2 terminating...
2/19/2016 9:03:42 PM : Worker Thread 17 received interrupt
2/19/2016 9:03:42 PM : Worker Thread 17 terminating...
2/19/2016 9:03:42 PM : exception "Socket closed"received in server main thread, closing server slocket...
2/19/2016 9:03:42 PM : Thread Manager received an interrupt, exiting...
2/19/2016 9:03:42 PM : terminating server...
```

When the job queue is full and the addJob() method is called, a value of false is returned to the caller (the server main thread). The main thread then closes the connection to the client because it is not possible to handle the request.

```
/19/2016 9:02:56 PM : connected to client: /127.0.0.1:50088
/19/2016 9:02:56 PM : adding job to jobQueue: Socket[addr=/127.0.0.1,port=50088,localport=9898]
/19/2016 9:02:56 PM :  Unable to add job from client Socket[addr=/127.0.0.1,port=50088,localport=9898] to job queue (the job queue is full), closing client connection.
```

Problems

One problem that I ran into was that when the lower threshold of the jobqueue was reached and the thread manager tried to half the number of threads, it didn’t take into account the threads that were still processing jobs, as a result it tried to kill threads that were not finished yet. I fixed this problem by keeping track of which threads were working, and which were waiting. The waiting threads are the only ones that are available to interrupt during the decrease operation. I created methods in the threadpool (moveThreadToWaitingList() and moveThreadToRunningList()) class that allows a worker thread to announce that it is about to switch state from running to working (or visa versa), the threadpool is then responsible for moving the reference of that thread from one list to the other.

```
_threadPool.moveThreadToWaitingList(this);
Socket thisSocket = _jobQueue.fetchJob();
_serverLogger.appendToLog(this.getName() + " fetched job (" + thisSocket + ") from
job queue, waiting for command...");
_threadPool.moveThreadToRunningList(this);
```

WorkerThread.java lines 36-39

The thread manager now takes several things into account before it tries to increase or decrease the thread pool:

```
int currentJobCount = _jobQueue.length();
int currentThreadCount = _threadPool.numberThreads();
int noWaitingThreads = _threadPool.getWaitingCount();

if (    // we have reached the lower threshold
        currentJobCount   <  _lowThreshold
        // if we half the number of current threads, we will have at least the min left
        && ((currentThreadCount / 2) >= _minThreadCount)
        // make sure there are enough waiting threads to get rid of
        && (noWaitingThreads) >= (currentThreadCount - (currentThreadCount / 2) )) {

    _serverLogger.appendToLog("lower job count threshold reached, decreasing threadpool size");
    _threadPool.decreaseThreadsInPool();
}
else if (currentJobCount >= _highThreshold && (currentThreadCount * 2) <= (_maxThreadCount -
currentThreadCount)) {
    _serverLogger.appendToLog("upper job count threshold reached, increasing threadpool size");
    _threadPool.increaseThreadsInPool();
}
```

ThreadManager.java lines 36-53