NWEN 303 Concurrent Programming (2015)
Project One: Service Broking

1.One to one matching

Design

In my design i have 5 classes: a “Board” class, “Client” class, “Providers” class, “StopWatch” class and “OneToOneMacthing” class.

In “OneToOneMatching” class i have A semaphore which is use to limit the the number threads accessing the shared memory.
A “evaluateTime” and “postPeriod” variable to delay the threads.
A main method which go through the number of threads given and each time it go through the loop there will be a new thread created to run the clients and providers class by passing in a random message. After all the threads finish running the thread pool will terminate or the thread pool will automatically terminate after 5 minutes.

In the “Board” class i have two arraylist representing the bulletin board one arraylist use to store the messages posted by the clients and one for the provider. I did this is because a client can only look at the message that posted by the provider and a provider can only look at the message that posted by the clients.

In “StopWatch” class it has a start and stop method which is used to start the timing and stop the timing and there are two method which is use to get the time in milliseconds and in seconds which are getElapsedTime and getElapsedTimeSecs.

In “Clients” class its constructor takes a message and call the check() method, in the check method it use a random variable which will either return 0 or 1. If it returns 1 then call doCheckFirst() method which will check the message first, if found the message it will take some time to evaluate it and remove message if it meet the client's need and post message if message was not found. If the random variable returns 0 then it will call doPostFirst() method which will post a message first then check if the message can find in the provider's messages, it will take some time to evaluate it and if found then remove the message that was posted and the provider's message.

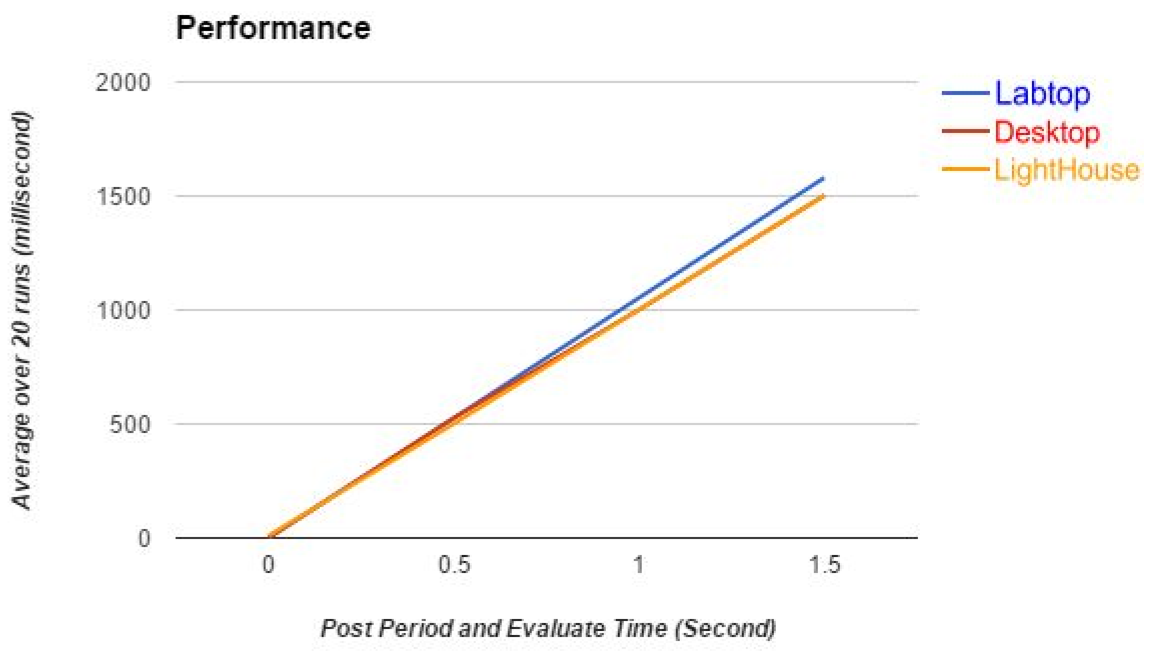
In “Providers” class i did the same thing as “Clients” class.

Performance

Desktop - 6 Cores, Maximum speed: 2.60 GHz, 85 Processes
Labtop - 2 Cores, Maximum speed: 2.40GHz, 69 processes

LightHouse

Cycles run = 20			
	Total time spend (millisecond)		
Post Period and Evaluate Time (Second)	Labtop	Desktop	LightHouse
0	47	31	235
0.5	10544	10539	10041
1	21081	20032	20040
1.5	31599	30039	30042
	Average over 20 runs (millisecond)		
Post Period and Evaluate Time (Second)	Labtop	Desktop	LightHouse
0	2.35	1.55	11.75
0.5	527.2	526.95	502.05
1	1054.05	1001.6	1002
1.5	1579.95	1501.95	1502.1



2.Many to one matching

Design

In my design i have 5 classes: a “Board” class, “Client” class, “Providers” class, “StopWatch” class and “ManyToOneMacthing” class.

In “StopWatch” class it has a start and stop method which is used to start the timing and stop the timing and there are two method which is use to get the time in milliseconds and in seconds which are getElapsedTime and getElapsedTimeSecs.

In the “Board” class i have two arraylist representing the bulletin board one arraylist use to store the messages posted by the clients and one for the provider. the other two arraylist one representing the providers object and one representing the client objects.

In the “Clients” class i have a getMessage message method which just use to get the message of the client, a setMessage method which is use to set the message of the client and i have a toString method which will return the message of the client.

In the “Providers” class i have all the method from the “Clients” class except that i have a list of string to store the client’s sign ups.

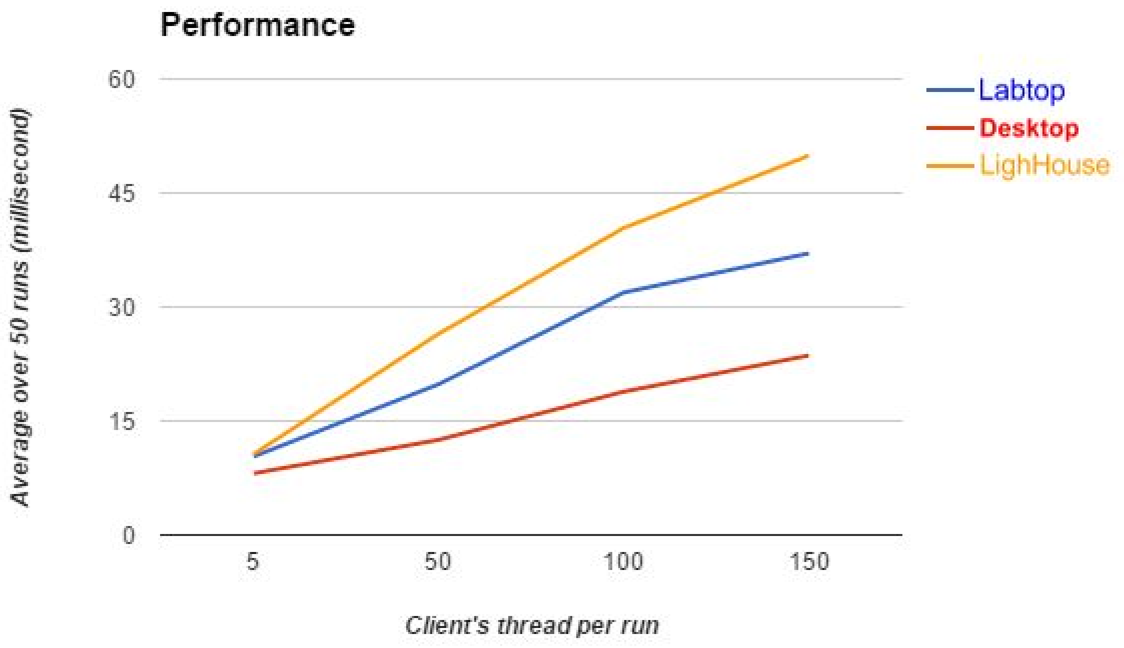
In the “ManyToOneMatching” class i have a semaphore which is use to limit the the number threads accessing the shared memory.
I have a maxSignUps variable to represent the number of sign ups for each provider’s offer.
A choose variable which will return 1 or 0 and in the check method it will decide whether the offer should be time limit or not.
A timeLimit method which represent the time the offer will stay.
A cycles variable to decide how much cycle we should run the program.
A threads variable to represent the number of clients in each cycle.
A ClientSignUp method which will go through each provider's message and find the message that matches the client's request if it matches it will add the message into the providers list if the list is full it will remove the list. if the message is not found it will post a request up to the board. It also has a chance to remove its own request.
A ProvidersOffer method will add a providers offer first then it will check each client's message and see if any message matches the offer if match, it will call providercheck method which will add the client's message into the providers list and remove it from the clients list. It also has chance to remove its own offer.
A timeLimitOffer method which is the same as the ProviderOffer method but it has a time limit to each provider’s offer.
A ProviderCheck method which will check each request from the clientsMessage list and see if any request matches the offer if so it will add it into the providers list.
A main method which go through the number of threads given and each time it go through the loop there will be a new thread created to run the clients and providers class by passing in a random message. After all the threads finish running the thread pool will terminate or the thread pool will automatically terminate after 5 minutes.

Performance

Desktop - 6 Cores, Maximum speed 2.60 GHz, 85 Processes

Labtop - 2 Cores, Maximum speed: 2.40GHz, 69 processes
LightHouse

Labtop (Total time spend)	516	992	1595	1853
Desktop (Total time spend)	405	626	943	1181
LightHouse (Total time spend)	532	1324	2020	2496
maxSignUps = 5				
Offers Time Limit = 0.3 second				
Cycles run = 50				
	5 clients to 1 provider	50 clients to 1 provider	100 clients to 1 provider	150 clients to 1 provider
Client threads per cycle	5	50	100	150
Labtop (Average over 50 runs)	10.32	19.84	31.90	37.06
Desktop (Average over 50 runs)	8.10	12.52	18.86	23.62
LightHouse (Average over 50 runs)	10.64	26.48	40.40	49.92


Performance
Average over 50 runs (millisecond)
Client's thread per run
Labtop
Desktop
LighHouse