In5570 oblig 2 thomaalm

Exercise 1 - break-even for call-by-visit

Files: break-even.m,break-even.txt

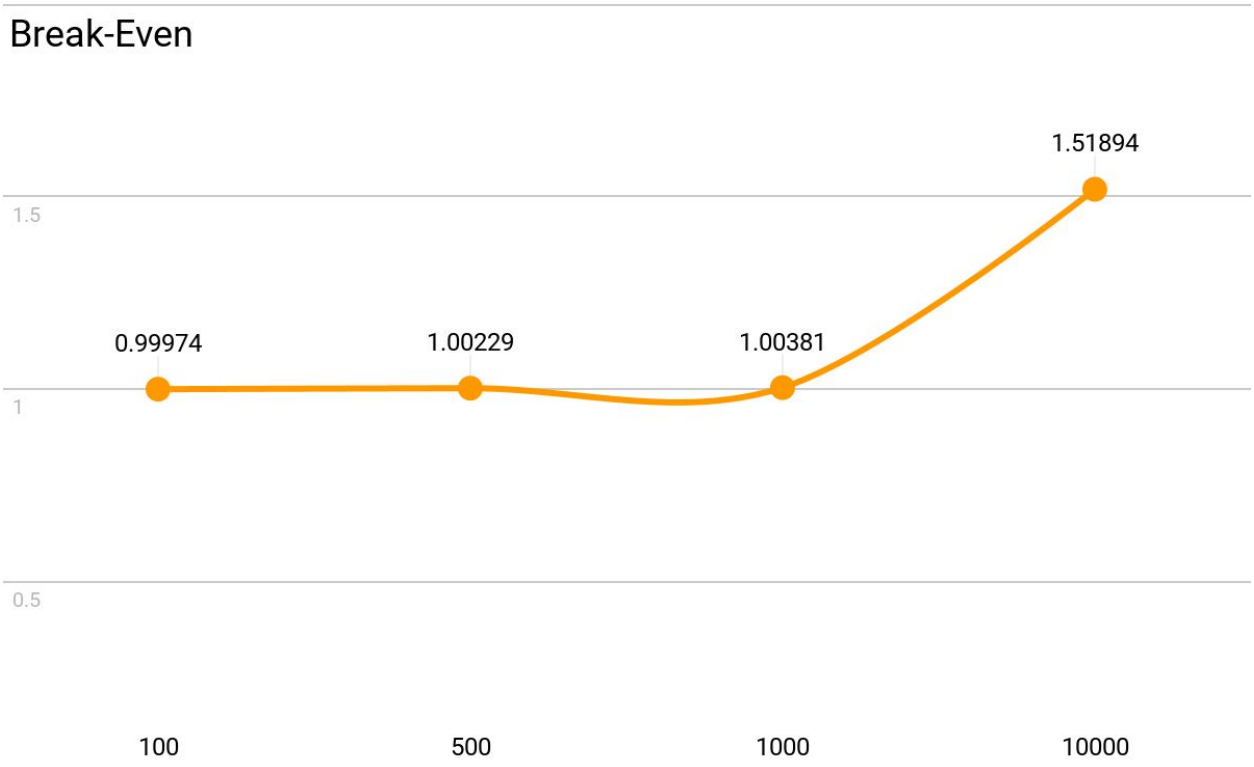
This program works by having two nodes connected. A remote object and an origin object.
The remote object contains an operation the origin object is supposed to “call” on, with a parameter that varies in size. The program emulates two different ways of calling the function.

Call-by-visit: the parameter is moved when needed
Call-by-move: the parameter is moved with its whole register, and is moved back when the operation is finished.

The big difference between these in terms of performance is when the parameter is used/called a billion times, it will call back to the origin node several times depending on the attached instance variables. But in call-by-move, with an attached variable, it will move its whole registry to the other object, and will benefit of not having to call back to origin. This however is very costly, and is only worth to do on large objects.

Below is a graph illustrating the break-even points for the array sizes [50,100,500,1000,10000]. As you can see, it only starts to change at 10k, i.e. at pretty large sizes.

The results in multiple iterations were very much the same, so I do not see the need to provide min/max/median/avg results here.



Exercise 2 - time-collector

Files: time-collector.m, time-collector.txt

This program works like kilroy, but it also prints the local time at the node it is visiting. With this we can see how long it takes for the program to visit each node, one by one. My results varies in about 100ms, which makes sense since the servers I chose were not that far apart. (Italy,Poland,Czech,Netherlands)

Exercise 3 - time-synchronization

Files: time-sync.m, time-sync.txt

I understood that other people moved individual objects to the different nodes, and then called on them to get the times. I did it by looping throgh the nodes and getting their local time instead. But I got similar results, so I think this is alright.

For the synchronization algorithm I used Cristian's. Which gets the round-trip time (RTT) and then estimates the synced time for a node like this: Time+RTT/2.

So basically it estimates the time it takes to send its current time, and then adds that time to the localtime of the node. Very simple.

The algorithm has an error of failure up to RTT, since the time it takes to request the localtime could take 0.9*RTT and to send the result back could essentially take very little time.

The algorithm should work when the time difference is less than the round-trip time. In my results we see that this is the case.