Oblig 2

Victor Nascimento Bakke

victonba@ifi.uio.no

Exercise 1

Source file: `ex1_call_by_visit.m`

Output file: `ex1_call_by_visit.txt`

To get some distance between the nodes, I opted to use `planetlab1.xeno.cl.cam.ac.uk` and `ple1.cesnet.cz` to house the nodes for this exercise. The machines are more than 500 km apart.

Even with these distances, I could not get meaningful numbers out of the tests. Break-points vary from 1 call up to 4 calls run-by-run, with little consistency. Sometimes, the larger sized lists have lower break-points than the smaller lists, probably due to network instabilities.

Inspecting the actual times required for each type of call, it seems there is considerable overhead in calling the consuming object regardless of parameter object size. Only at 10000 items in the list do we see a noticeable increase in time taken for each type of call.

I tried multiple ways of consuming the parameter object in the remote consuming object, whether it be iterating through the list or just retrieving it, but there does not seem to be any difference in the resulting time it takes to call the method on the consuming object.

The list must be `attached` to the parameter object because otherwise the list would remain at `home` while the parameter object itself was `move`d to the `consumer`, and call-by-visit would (in theory) never have a break-point because it would only add additional overhead to the regular direct invocation method. In the real world, network interference would make it likely that you might encounter some "break-point" of sorts where the un`attached` "call-by-visit" method would be faster.

Exercise 2

Source file: `ex2_time_collector.m`

Output file: `ex2_time_collector.txt`

I used the PlanetLab machines on the following nodes:

- `ple1.cesnet.cz` (Prague, Czech Republic)
- `cse-white.cse.chalmers.se` (Gotenburg, Sweden)
- `kulcha.mimuw.edu.pl` (Warsaw, Poland)
- `planetlab1.xeno.cl.cam.ac.uk` (Cambridge, United Kingdom)

Exercise 3

Source file: `ex3_time_synchronization.m`

Output file: `ex3_time_synchronization.txt`

For the PlanetLab portion of this exercise, I will be reusing the sites I picked for the previous exercise:

- `ple1.cesnet.cz` (Prague, Czech Republic)
- `cse-white.cse.chalmers.se` (Gotenburg, Sweden)
- `kulcha.mimuw.edu.pl` (Warsaw, Poland)
- `planetlab1.xeno.cl.cam.ac.uk` (Cambridge, United Kingdom)

These machines are all more than 500 km apart.

To synchronize the time I do the following:

- For each time agent:
  - Note the current time as `startTime`
  - Request the time of the time agent as `agentTime`
  - Note the current time after receiving time from the agent as `endTime`
  - Calculate the `timeDifference` as `(endTime - startTime) / 2`. This time difference represents the amount of time to send a request to the time agent (network latency). We naively assume the network latency each way is equivalent to half the time of sending and receiving the agent's time.
  - Calculate final time at the agent by doing `agentTime + timeDifference`.
- Then calculate the average clock time on the nodes calculated as `averageTime`. This is the final clock time.

This approach is very simple, but does not take into account anything other than individual node network latency, and even then does not take into account that the nodes are fetched from in sequence, where the earlier in the sequence the agent's time was fetched the more "incorrect" it is from the final calculation. This is visible in the output of the program.

On the other hand, this approach does give an estimated time accuracy down to roughly 25 milliseconds, based on my guesstimate, in the PlanetLab setup.