Michael Christensen
October 17, 2013
CS 360

Lab 3 – Download Accelerator

Methodology:

To run the experiments, I created the `downloadAccelerator.py` module, which defined both the `DownloadAccelerator` and `DownThread` classes. The `DownloadAccelerator` requests via the specified url's header the length of the whole file to download and then spawns some specified number of threads, giving them the responsibility to download different, equal-sized chunks of the file. When the threads end, the contents they've downloaded are concatenated into one output file. Using the given `experiments.py`, I ran the downloader multiple times by downloading a small, medium, and large sized file with 1, 2, 3, 5, and 10 threads, 10 times for each number of threads so I could determine the median amount of time that using a certain amount of threads takes. I did this while both on-campus and off-campus.

Graphs:

On-campus:

Fig. 1: download-time-small:

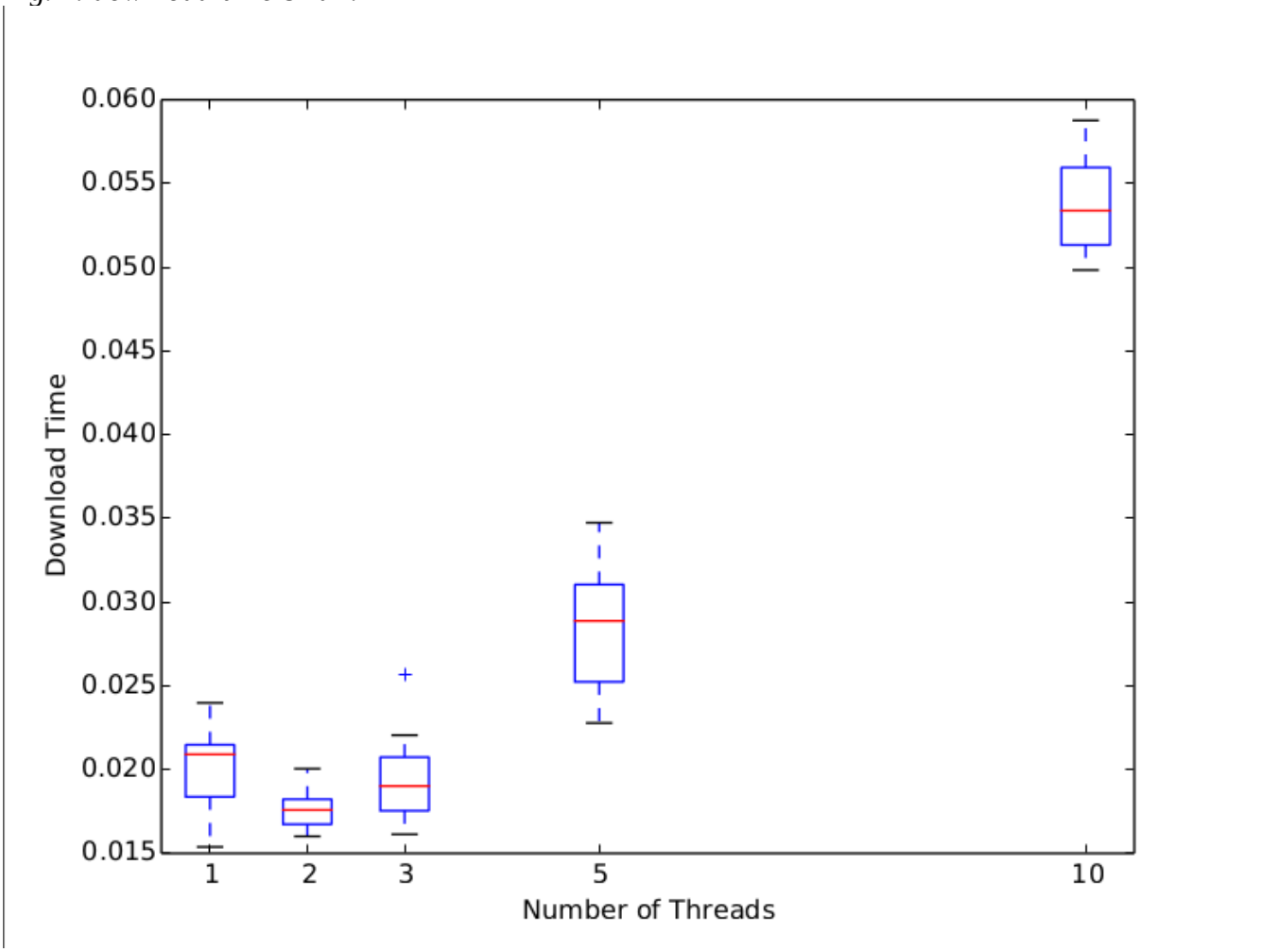


Fig. 2: download-time-medium:

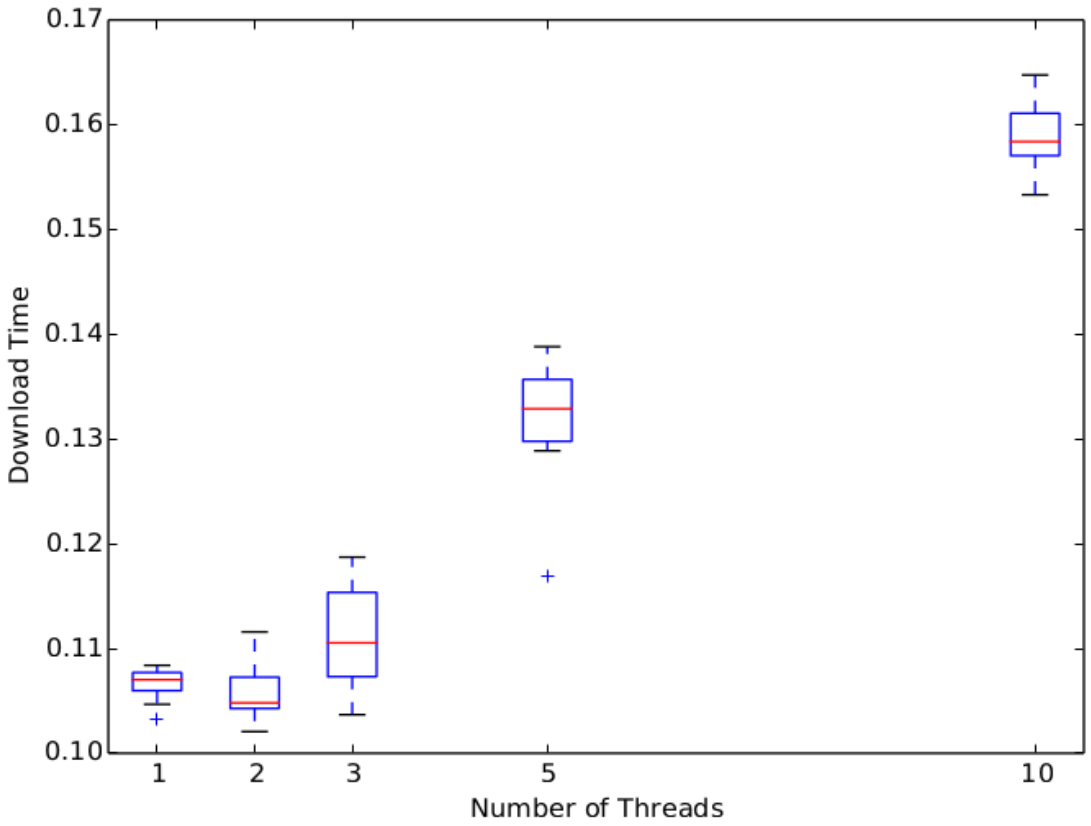
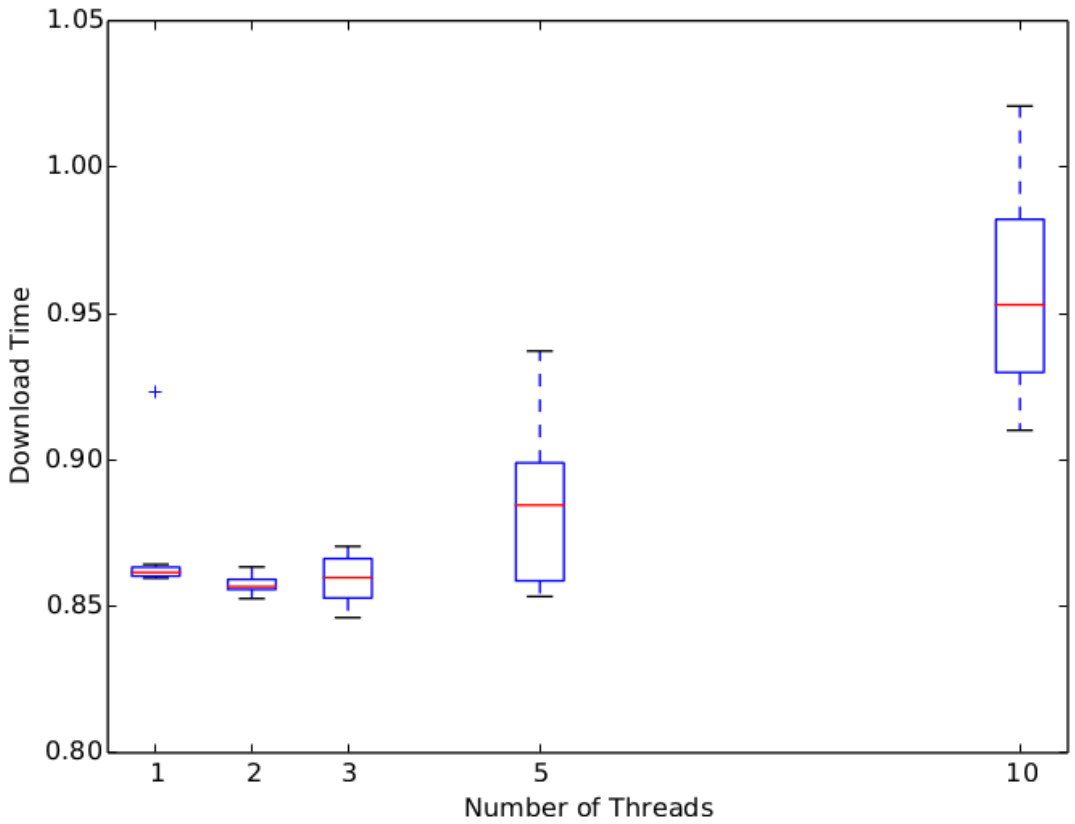


Fig. 3: download-time-large:



Off-campus:
Fig. 4: download-time-small:

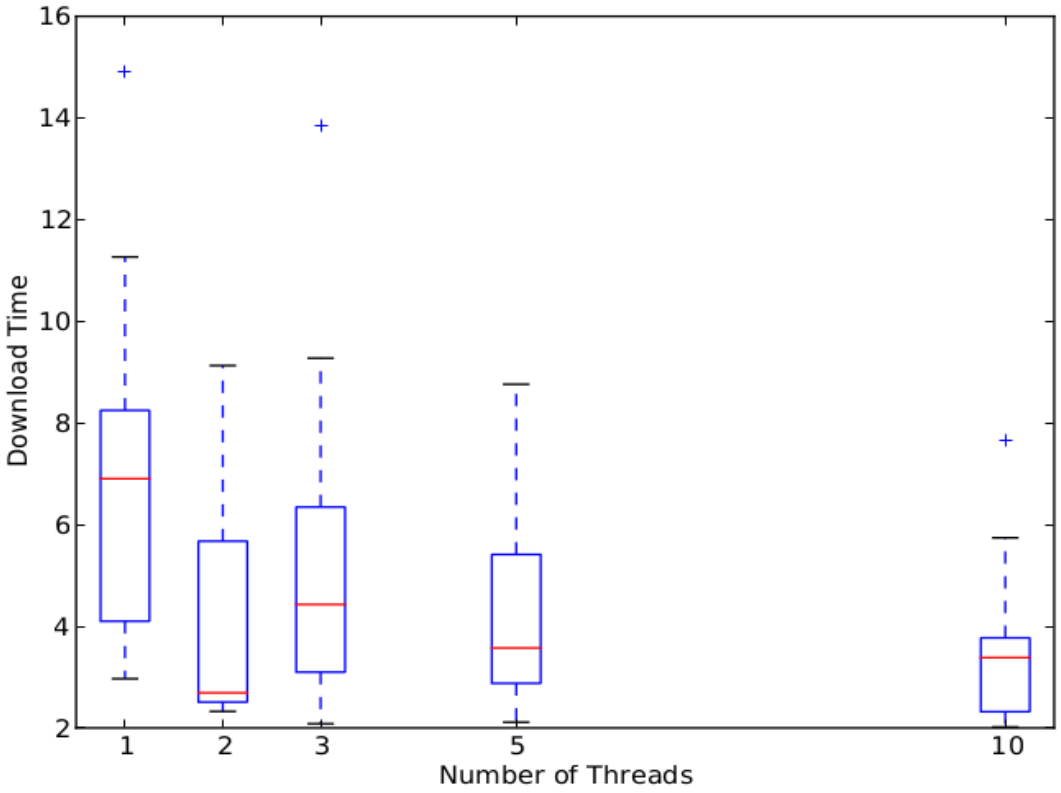


Fig. 5: download-time-medium:

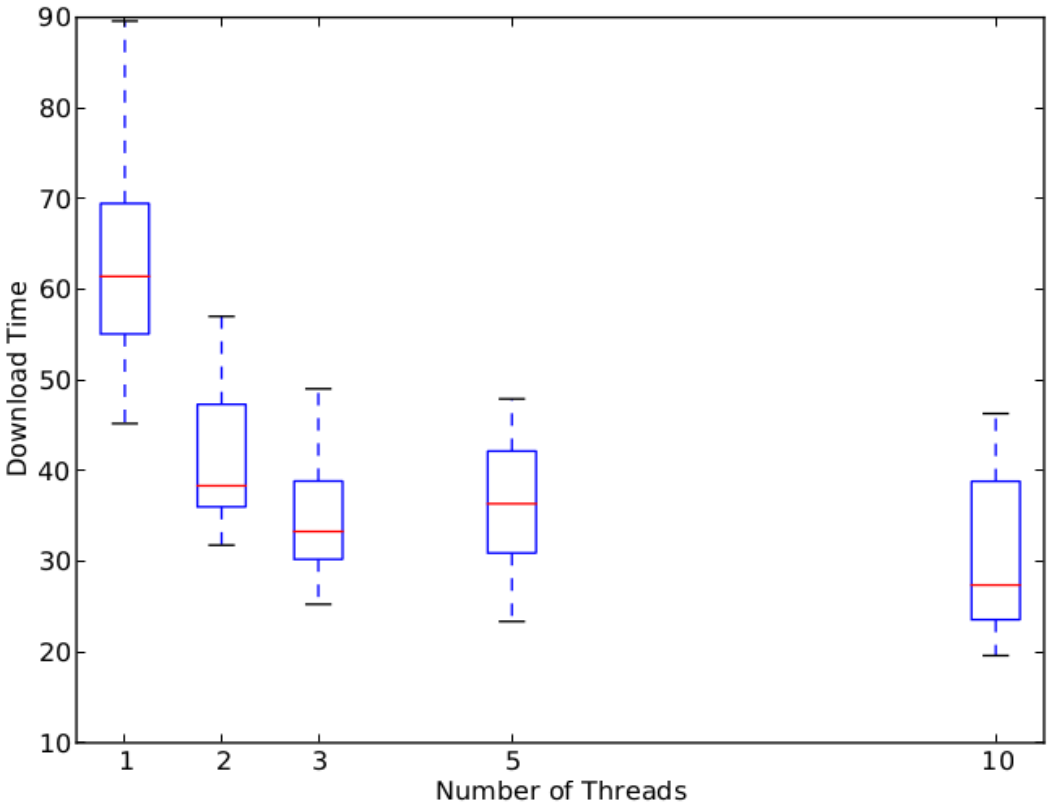


Fig. 6: download-time-large: Took too long to download off-campus, and since the lab specifications said it wasn't necessary to continue trying this if that was the case, it is not included here.

Discussion:

The graphs shown in Figures 1 through 5 above show surprising results. For off-campus downloading, the increase in number of threads mostly correlated with the decrease in download time (the one exception is for 2 threads in Figure 4; however, for this figure, you can see that the box comprised of all four quartiles for 10 threads is lower than the box for 2 threads, in addition to the one outlier for 10 threads lying at around 7.8 seconds, which is probably skewing the results. I feel that given another 10 times running, the median and average time for 10 threads in Figure 4 would be lower than the median and average for 2, supporting the hypothesis that more threads translates to lower download times). However, for on-campus downloading (Figures 1-3), it becomes obvious that adding more threads *doesn't* necessarily mean lower download times. In all three cases (small, medium, and large-sized files, Figures 1-3 respectively), 2 threads is clearly the optimal number of threads and increasing the number of threads used after that point only makes the performance worse. In Figure 3, the results for 1, 2, and 3 threads are extremely close to one another (~0.05 seconds), leading one to conclude that perhaps adding more threads becomes more important the larger the file, but this difference is only really apparent and needed when the network speed is slower. That is, since downloading a file is more of an IO-bound process rather than a CPU-bound process, while on campus an increase in the number of threads makes no strong difference since the network speed is so fast (the difference between CPU and network speeds is small). However, off-campus, where the difference between CPU speed and network speed is great (due to the slowness of off-campus Internet), extra threads are helping to alleviate the burden because they can distribute the work and spend the time necessary on this IO-bound activity.