Assignment 03 - Profiling and Numba

DSE 512

Spring 2021

For this assignment, you will extend the code we created in class, located at `/lustre/haven/proj/UTK0150/jhinkl13/kmeans`. **NOTE: for this assignment you may do this assignment either on ISAAC or on your personal computer.** If you choose not to do the work on ISAAC, please remember to copy your work back onto ISAAC or track it on github in a public repository so that we are able to see your git repository. The submission you return to us should be a brief report formatted in HTML, DOCX, or PDF.

For the report that you submit, you do not need to overly format it; you can simply list your responses to each of the problems below. Please do the following using number of clusters `-k 4`, using the **TCGA dataset**:

1. (25 points) Starting from the kmeans repository developed in class, which you extended in the last two assignments, refactor the `kmeans.py` to contain the following subfunctions: `compute_distances()`, `expectation_step()`, `maximization_step()`, called at the appropriate places inside the `kmeans()` function. Ensure that the program still runs. Do this for `kmeans_vectorized.py` as well. In your report, indicate you've completed problem 1 and provide the path to your code on ISAAC (or github if you choose to use it).
2. (25 points) Profile your newly-refactored `kmeans.py` and report the time spent in each of the three new functions both in seconds and as a percentage of the total runtime. Do the same for `kmeans_vectorized.py`. Recall that `kmeans_vectorized.py` attempted to speed up the `compute_distances()` portion. Use Amdahl's Law to compute the theoretical maximum speedup possible by optimizing and parallelizing the `compute_distances()`. What percentage of that speedup have we actually obtained by vectorization with numpy?
3. (25 points) Visualize an icicle plot of the profiling output for `kmeans.py`. You may use Snake-Vis or Viztracer along with `cprofiler`, as Todd demonstrated in Lecture 15. Do the same for `kmeans_vectorized.py`. You may include these as screenshots in your report; please rescale the figures to ensure that we can see the main function names in these plots.
4. (25 points) Using the profiling output from Problem 2, determine the maximum speedup you could obtain by optimizing the `expectation_step()` and `maximization_step()` (note that you may need to refactor further to measure the runtime of the main kmeans loop). In a new file, `kmeans_numba.py`, use the `@numba.jit` decorator (install and import numba in your code), re-profiling your code, and compare your runtime to the ideal speedup given by applying Amdahl's Law. Report the new profiled runtimes for these three functions and the total runtime using Numba in your report.