Parallelized Vector Search

15-618 Final Project Milestone Report

Akash Nayar Dhruva Byrapatna

November 2025

1 Work Summary

To date, we have completed a significant portion of the project deliverables listed in the project proposal, and we believe that we are on track with respect to our schedule. We began by developing a serial implementation of our vector search implementation, including a serial k-means initialization and training, a serial build of the inverted file index, and serial search and add functions. We worked on templating our code to be able to sub in and out different versions of computations like distance calculations, and created bindings allowing us to call and test these functions using Python. After successfully implementing our serial code, we created correctness and benchmark harnesses for verification and performance testing. Our correctness tests check the results of our computations against Scikit-Learn's k-means and a brute force kNN implementation, and our benchmarking tests compare vector search version performance to the serial implementation.

After completing this work, we also implemented several parallelized versions of portions of the code to improve performance. We successfully used OpenMP to parallelize query searches in our search function. We also used intrinsics _m256 to create a SIMD-based version of the distance calculation, which is used to determine the similarity between two vectors during multiple stages of the IVF program. We have been able to complete base versions so far of these two, and have confirmed correctness and good performance—more details can be found in Section 4 of this report. We have also worked on a "cache-friendly" version of the distance computation to further improve performance and reduce bandwidth costs. We are currently still working on this—while we have confirmed correctness and have seen some performance improvements, we are not seeing the level of performance increase we expected.

2 Are we on Track?

We are on track to complete our project according to the specifications discussed in the proposal. While we have not yet fully optimized the distance calculation for performance, we have done a first pass at query-based parallelization, which was originally scheduled to be done in the upcoming week, so we therefore believe we are in a good spot. A full schedule of what we will do with our remaining time is shown below.

- Dec 1 to Dec 3:
  - Submit midway report
  - Further optimized "cache-friendly" distance computation
  - Explore additional optimizations to query-based parallelization
- Dec 4 to Dec 6:
  - Conduct additional performance testing with full-scale datasets
  - Work on and complete candidate-based parallelization
  - Port portions of code to ISPC and experiment with ISPC to test for additional performance improvements.
- Dec 7 to Dec 8:
  - Test program performance on PSC Bridges machines
  - Finish data collection and any other outstanding items
  - Submit final report

3 Deliverables

Since there is no poster/demo session this semester, we plan on delivering a comprehensive final report detailing our development process, parallelization techniques, and results. We want to analyze the effects of each optimization we do and understand how they come together to provide the final speedups. Furthermore, we want to inspect how our system holds up at high thread counts on the PSC machines (32-128 cores).

4 Preliminary Results

Our preliminary results are very promising. So far, we've implemented a base implementation, SIMD-ized the distance kernel, and parallelized over queries. We notice that the SIMD-ized distance kernel provides $\sim 5\times$ speedups, with parallelization over queries yielding an additional $\sim 5.5\times$ on 8 cores on the GHC machines (bringing us up to a $\sim 27\times$ speedup). These prelimiary results are on synthetic datasets, for the final report we want to measure performance on

real-world/academic datasets (GIST, OpenAI Large 3, etc.). Furthermore, we started experimenting on our cache friendly distance computation idea, where we keep the same chunk of the query resident in cache for multiple candidates instead of constantly streaming for each candidate. However, we are still developing this implementation and our results aren't yet matching our expectations.

5 Future Concerns

Our main concern is that while our current implementation seems to be scaling decently with the number of threads on the GHC machines, we're not sure how far that'll go on the PSC machine with much higher core counts (32-128). Since IVF technically doesn't guarantee each query gets the same amount of work (since there are no guarantees on the sizes of the clusters), it's possible that workload imbalance starts to become an issue when there are many theads. In fact, it is very easy to construct adversarial cases where workload imbalance can thwart speedups, though typical datasets hopefully won't exhibit this.