Parallelized Vector Search

15-618 Final Project Proposal

Akash Nayar Dhruva Byrapatna

November 2025

1 URL

Our project proposal, reports, and source code are on GitHub.

2 Summary

We're building a parallelized implementation of an IVF vector search index. We hope to analyze the accretive effects of different optimizations (base vs. SIMD vs. multiprocessing vs. others).

3 Background

Vector search, also known as similarity search or approximate nearest neighbor search (ANNS) is a core component of modern AI systems. The retrieval of similar vectors within a database of possibly millions, or even billions, of data points powers prominent technologies such as RAG and recommendation systems. ANNS works in two stages. When it gets a query, it first cleverly filters the dataset into a set of relevant "candidates." This initial step is called "filtering." It then performs naive k-Nearest Neighbors (kNN) among just these filtered candidates. This second step is called "refinement."

IVF (Inverted Vector File) is a popular and conceptually elegant example of an ANNS index. Its filtering method works as follows: During index creation, the dataset is partitioned into n_{list} clusters using k-means clustering. During query time, you begin by finding the n_{probe} nearest cluster centroids to the query. Filtering simply selects all points within those n_{probe} nearest clusters. In very large datasets, n_{list} and n_{probe} can be tuned to balance speed with accuracy.

There are two main possible modes of parallelization. Typical similar search workloads batch incoming queries to amortize various overheads. As such, these algorithms can benefit greatly from parallelizing over queries. Second, each query must be compared against each of its candidates. This distance calculation exposes the opportunity for SIMD-vectorization. A third option, albeit less

common, is to parallelize over candidates. We plan to test out all combinations of these techniques and perform a thorough analysis of our findings.

4 The Challenge

There are several challenges related to this work. First of all, the varied problem steps will require different approaches; for example, efficiently parallelizing the distance calculations will require different techniques than query-based parallelization. We will need to make sure that the different techniques and frameworks we use are compatible with each other.

Moreover, certain portions of the work present memory locality challenges. The SIMD optimized nearest neighbor computation will have important memory locality considerations—as neighbors are checked, computations will have to be arranged to ensure optimal cache usage, to avoid evicting query vector data from the cache before all computations on it are computed. Moreover, for large datasets the memory locality in the k nearest neighbors could pose a challenge. We will have to investigate whether ensuring vectors in the same cluster are stored contiguously (and the memory traffic this would entail) will be worth the tradeoff of faster accesses during kNN search.

Divergent execution will also be another potential challenge. The IVF algorithm has no guarantees on cluster sizes being approximately the same size. As such, parallelizing over queries may not be as trivial as it seems. While the read only nature of search should prevent cache coherence issues across similar queries, we will have to investigate different methods of distributing work to ensure certain processors are not overloaded.

The capabilities of the GHC system are well known to us. We do not expect to be bandwidth bound, as all parts of the process do have a relatively even communication to computation ratio. One thing we can consider a constraint is the limit of 8 processors available to us-this may limit potential speedups in very large datasets/high search frequency applications. Time permitting, we will explore getting around these constraints as part of our stretch goals extending our implementation to the PSC Bridges machines and exploring using ISPC.

5 Resources

We will be using the GHC machines (x86_64 with AVX2 support) as our computing platform for this project. All experiments for both correctness and performance will run on the GHC machines. We will choose to implement our system in C++ from the ground up. We will use reference papers and libraries for understanding the algorithms we intend to implement, such as Faiss (Douze et al. 2025) and Inverted Files (Bruch et al. 2024), but we will not be using starter code for this project—as discussed above, the implementation will be entirely our own. We believe the GHC machines we have access to will be sufficient for this project, but may benefit from additional compute on the PSC Bridges

machines if our stretch goals are reached to test portability and performance

6 Goals and Deliverables

Our goals are separated below into what we plan to achieve (base goals) and what we hope to achieve (stretch goals). Within each section our goals are ordered in the sequence we expect to tackle them in.

Base Goals:

1. Create an initial working serial implementation of an IVF vector search index.
2. Develop a test harness for evaluating correctness and a bench harness for evaluating performance
3. Develop a SIMD-ized nearest neighbor distance computation that will be used for both efficient filtering and efficient refinement
4. Test and optimize SIMD-ized distance computation for speedup
5. Implement query-based parallelization on top of SIMD-ized distance computation
6. Test and optimize query-based parallelization for speedup

Stretch Goals:

1. Implement candidate-based parallelization
2. Test implementation on additional machines
3. Replace C++ SIMD code with ISPC code

We believe these goals are an accurate representation of we can reasonably hope to achieve given the project timeframe and the fact that a core component of our project involves implementing these methods from scratch. We expect to achieve all of our base goals and be able to at least start on our stretch goals by the time our report is due. If we run into obstacles, we may reduce our focus on our final base goal (implementing query-based parallelization), and prioritize getting a performant and correct version of the SIMD-ized distance computation running.

Under our base implementation, we expect to achieve a speedup in the range of 4-6x for the SIMD-ized distance computation implementation over our serial code, and a total speedup of 16-32x for query-based parallelization over the serial code.

With respect to our stretch goals, we expect to see a speedup of 15x if we are able to implement candidate-based parallelization. We also expect to see

speedup further scale with the number of cores if we are able to test on the PSC Bridges machines. With respect to ISPC code, we will have to explore the speedups associated with this further if we are able to get to this point.

7 Platform Choice

As discussed, we will be writing our code on the GHC machines. We believe this is the right choice for our program. As the GHC machine cores are typical of what we might expect for a backend system handling vectorized search, we will essentially developing a program according to plausible real-world considerations. Moreover, we are familiar with the GHC architecture and the instruction sets available to us-given the timeframe of this project, developing on these machines makes more sense than learning a different architecture.

In addition, we intend to program in C++. While ISPC would also be an option for performing SIMD based calculations, C++ provides us with much more flexibility and will allow us to explore more different ideas.

8 Schedule

- Week of Nov 17:
  - Submit project proposal
  - Complete working serial implementation
  - Develop SIMD-ized distance calculation
  - Begin implementation of SIMD-ized distance calculation into program
- Week of Nov 24:
  - Complete SIMD-ized distance calculation implementation
  - Build testing and bench harness
  - Test SIMD-ized distance calculation performance
- Week of Dec 1:
  - Submit midway report
  - Build query-based parallelization
  - Test query-based parallelization
  - If time, work on stretch goals
- Week of Dec 8:
  - Submit final report