Projects with Available Mentors:

- [Vigya - pref. 1] Multi-Vector Support for HNSW Search in Lucene –
    - This idea has been discussed before in https://github.com/apache/lucene/issues/12313 and I have a working prototype in a Draft pull request – https://github.com/apache/lucene/pull/14173
    - However, there's still a lot of work pending in getting this PR from "Draft" to ready. There are missing pieces like 'merging' for multi-vectors, adding backwards compatibility support, adding functional and unit tests, and adding appropriate benchmarks.
    - Additionally, there are tangential important workstreams like benchmarking recall and result relevance on some open industry datasets.
    - This change is deep in Lucene internals. It has the benefit of existing discussions and a guiding prototype, and the potential to fork off several sub-projects.
- [Vigya - pref. 2/ Mike M/ Rahul G] A multi-tenant concurrent merge scheduler – https://github.com/apache/lucene/issues/13883
- [Vigya - pref. 2 / Mike M] **Benchmarking**: so crucial (it is our compass), yet often 2nd class citizen. There are many open issues in luceneutil.
- [Vigya - can help] Trawl through open issues on GitHub (same issues also searchable here in our dogfood Lucene issue/PR search app)
- [Mike M] **Machine learned merge policy**
- [Mike M / Vigya/ Rahul G / Luca C] **Intra-segment query concurrency**
- **[Mike M / Mike S] Indexed queries**

Possible summer internship Lucene open source projects:

- **SIMD on ARM**: Lucene devs have worked hard to take advantage of SIMD instructions on x86-64 (AMD, Intel), but less so on ARM (e.g. AWS Graviton CPUs)
- **Machine learned merge policy**: A bandwidth-cap'd merge policy that optimizes reclaiming of deletes. Can it be machine-learned, with dual metrics (minimal write amplification, delete %)? This project would use some fun tooling, e.g. recently added segment tracing, and pull from InfoStream for the app to fine-tune to the apps behavior.
- **Intra-segment query concurrency**: Lucene uses multiple threads ("thread per segment") to execute one query. It's a great feature, big latency reduction for apps that have lowish query traffic relative to the cluster resources. But if the index is optimized to a single segment, you lose all such concurrency (horrible leaky abstraction)! There has been good initial progress here, but this is still not usable for queries that have high per-segment Scorer initialization costs (like range queries). Let's make this feature so usable that it becomes the default execution mode!
- **KNN search**: maybe wrap more external vector engines as Lucene Codec, and evaluate performance using luceneutil. Maybe improved quantization? Long tail of important fixes here, e.g. not replicating full precision vectors.

- **OS/hardware mechanical sympathy**: tightly integrate Linux mincore system call to identify which fields of which parts of the Lucene index are hot (cached in OS's buffer cache).
- **Tighter profiling**: Low level kernel profiling (Berkely packet filter, something that detects kernel level lock waits).
- Async IO/io_uring? Or better IO prefetch? Cold indices on fast SSDs could benefit greatly … Lucene is poorly optimized for saturating all SSD concurrency. (Same idea could help w/ super high latency but concurrent store like S3).
- **Use LLMs to improve our docs, find possible bugs/refactorings, whatnot**: LLM coding tools have rapidly come a long ways – can we use them (one time, and then maybe integrated to our build for continuous future times) to improve something in our sources?
- [Vigya - pref. 2] **Benchmarking**: so crucial (it is our compass), yet often 2nd class citizen. There are many open issues in luceneutil.
- [Vigya - can help] Trawl through open issues on GitHub (same issues also searchable here in our dogfood Lucene issue/PR search app)
- Lucene test framework improvements – Lucene has a strong randomized testing infra (see past talks from Dawid Weiss), maybe there are open issues / improvements for it?
- An LLM based agent / tool to make onboarding and contributing to Lucene easy. It would be nice to have an AI agent that can explain deep internals of Lucene, point to the right classes and probable solutions for a given github issue, and help debug with test failures. Some of this might be off the shelf with "vibe coding" tools, but building an agent with deeper understanding would need deeper Lucene context. Maybe parsing, tokenizing, embedding and indexing Lucene codebase into a Lucene based RAG set up.
- Facets/aggregations?
    - Promoting the new aggregation engine out of sandbox: https://github.com/apache/lucene/issues/14619
- **Automation / Community tools**
    - A project dashboard similar to Apache Beam's. It helps us see how many PRs we're opening/reviewing/closing, so we can make sure we're giving enough attention to new PRs, especially from first-time contributors. It also has metrics on the automated tests that run and it keeps track of performance against benchmarks. Maybe this is an extension to luceneutil, which already has some of those things?
    - Backport bot to automate testing and publishing a backport PR for changes merged to main. On its own, this might not be enough work for a team of 5.
- Perform **"optimistic" prorated concurrent hit collection** when executing search queries, avoiding the need for any cross-thread communication during an initial collection phase, followed by a second round after checking in as needed.
- [Vigya - pref. 2] A multi-tenant concurrent merge scheduler – https://github.com/apache/lucene/issues/13883
- [Vigya - pref. 1] Multi-Vector Support for HNSW Search in Lucene –

    - This idea has been discussed before in https://github.com/apache/lucene/issues/12313 and I have a working prototype in a Draft pull request – https://github.com/apache/lucene/pull/14173
    - However, there's still a lot of work pending in getting this PR from "Draft" to ready. There are missing pieces like 'merging' for multi-vectors, adding backwards compatibility support, adding functional and unit tests, and adding appropriate benchmarks.
    - Additionally, there are tangential important workstreams like benchmarking recall and result relevance on some open industry datasets.
    - This change is deep in Lucene internals. It has the benefit of existing discussions and a guiding prototype, and the potential to fork off several sub-projects.
- Tensor Support in Lucene –
    - Going beyond high dimensional vectors, similarity search on tensors finds application in Life Science problems.
      "For example in structural biology, AI models like AlphaFold use 3D tensors to represent spatial relationships between amino acids. These tensors help the model learn how proteins fold, twist, and interact—critical for understanding disease mechanisms and designing therapeutics. This breakthrough wouldn't be possible without the ability to represent complex spatial data in tensor form and train neural networks to reason over it."
      –
      https://blog.vespa.ai/beyond-vectors-how-tensors-are-transforming-ai-in-life-sciences/
    - Would be cool to explore adding this capability to Lucene.