Parallelizing Seed Selection in DNA Read Mapping

Sunny Nahar

Carnegie Mellon University

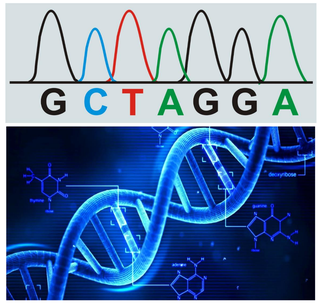
SAFARI

What is DNA Sequencing?

Genomes are composed of **base pairs**:

- *Adenine, Guanine, Cytosine, Thymine*.

The goal is to find the **sequence** of base pairs which compose the genome.

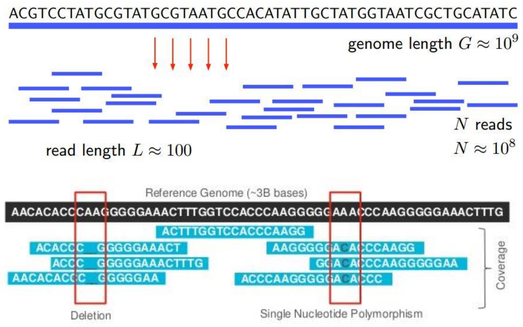


Why is this useful?

- Tracing evolution.
- Correlating genes with diseases.
- Forensics and identification.

How is Sequencing Done?

- DNA split into small pieces called **reads**.
- Using a **reference genome**, reads are mapped to potential locations.
- Must account for errors in reads: **insertions**, **deletions**, and **substitutions**.



This problem is very **computationally challenging**:

- Billions of reads.
- Fuzzy string matching.
- Multiple mapping locations per read.

Research Question

Question: Can we exploit multi-core machines to improve speed?

- Offset high cost of computation: read mapping is slow.
- Efficiently use resources (caches, cores, etc).
- Achieve good speedup and minimize overhead of concurrency.
- Build a low complexity, cache-efficient, memory efficient system.
- DNA Infrastructure is inherently parallel.

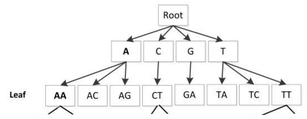
Machine Specifications:

- 4 sockets.
- 10 cores and 256GB RAM (NUMA) per socket.
- Intel Hyperthreading.
- Total 80 logical threads, 1TB RAM.

Generating the HashTree

Description:

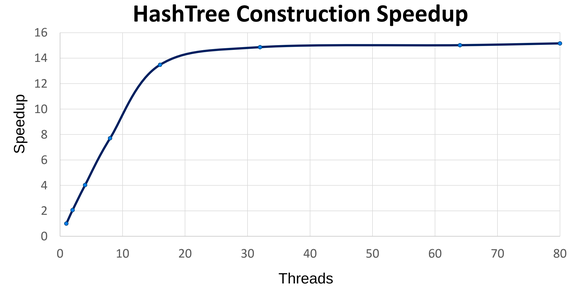
- Hashtable of frequency **tries** of the reference genome which stores seed frequencies.
- Queries are **O(L)** cache misses.
- Need to load from disk to construct and perform computation: 80GB.



Parallelization:

- Each thread reads portion of file.
- Using memory mapped IO removes copy to main memory (kernel page cache).
- Each trie can be independently generated. Construction time hides disk latencies.
- Dynamic work scheduling.

HashTree Construction Speedup



Bidirectional Frequency Predictor Construction

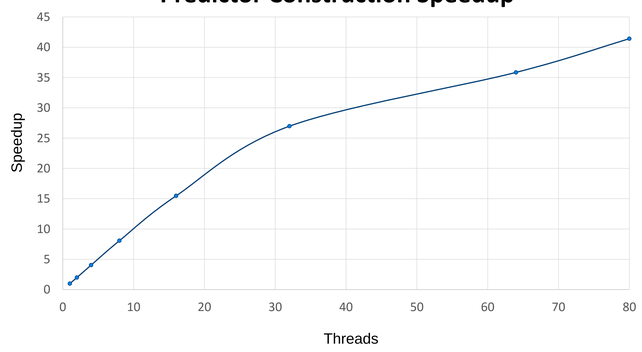
Tool for speeding up seed selection. A data structure to **estimate** frequency of seeds:

- Used to **minimize** reference trie queries.
- Predicts frequency given base seed, left and right extension.
- Gives **O(1)** cache miss complexity.

Parallelization:

- Requires single DFS traversal of HashTree.
- Threads take set of tries from the hashtable.
- Only synchronization is atomic writes.

Predictor Construction Speedup



Parallel Seed Selection

Seed Selection:

- Given a read, output set of seeds.
- Used in next stage (edit distance filtering).
- Low frequency seeds are important.

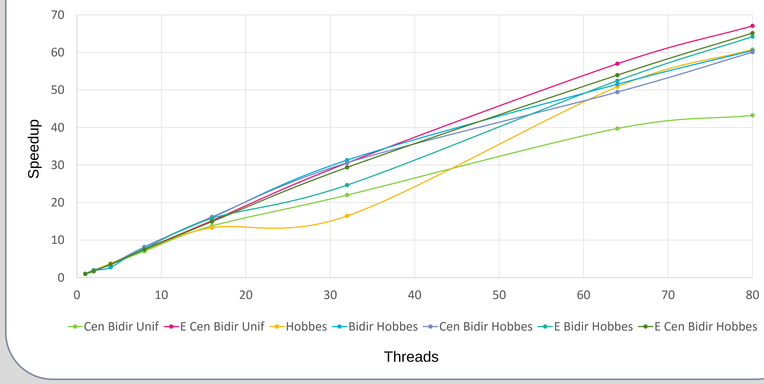
There are multiple seed selection algorithms and heuristics:

- Mix of accesses to HashTree and predictor.
- Varying levels of complexity.

Parallelization:

- Parallelize selection over read set.
- HashTree and predictor are static: read coherence overhead.
- NUMA effects on non-local threads (up to 2x overhead of read).

Speedup of Seed Selectors



Conclusion and Future Work

We were able to efficiently parallelize various components of the seed selection pipeline.

- **Exhibit large speedups:**
  - 15x for HashTree construction.
  - 41x for Frequency Predictor construction.
  - 60-70x for Seed Selection algorithms.
- **Still room for further optimizations.**
  - Processor affinity.
  - Masking NUMA with work distribution.

Acknowledgements

I would like to thank Prof. Onur Mutlu and Hongyi Xin for their help and advice on the project and for enabling this research.