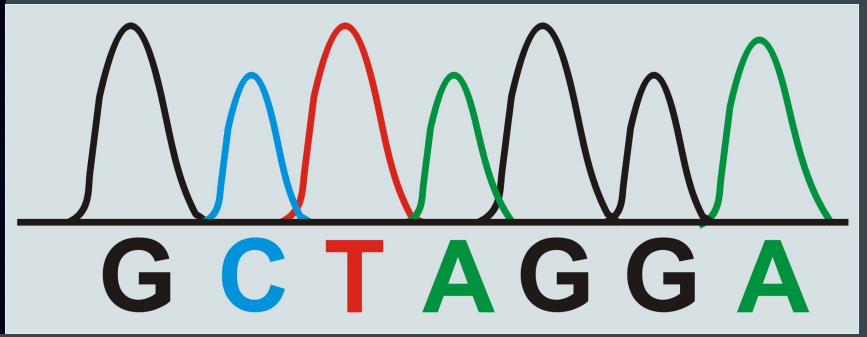
Parallelizing DNA Read Mapping

Sunny Nahar

What is DNA Sequencing?

Finding the base-pairs for the genome.

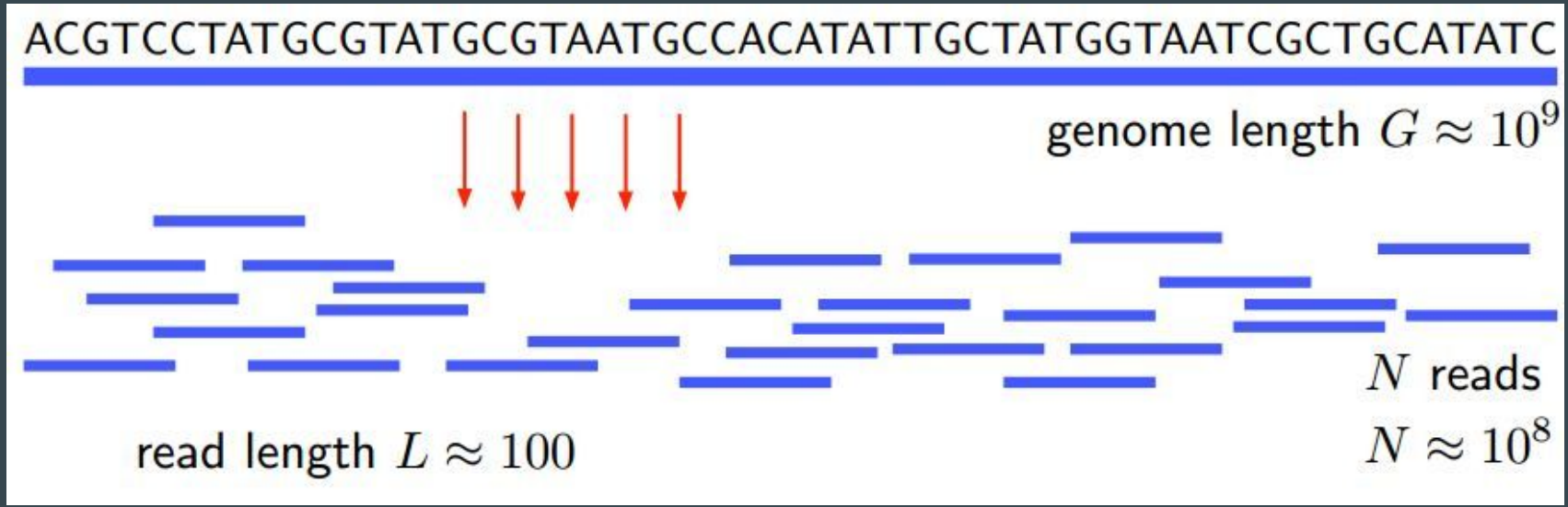


Why is this useful?

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

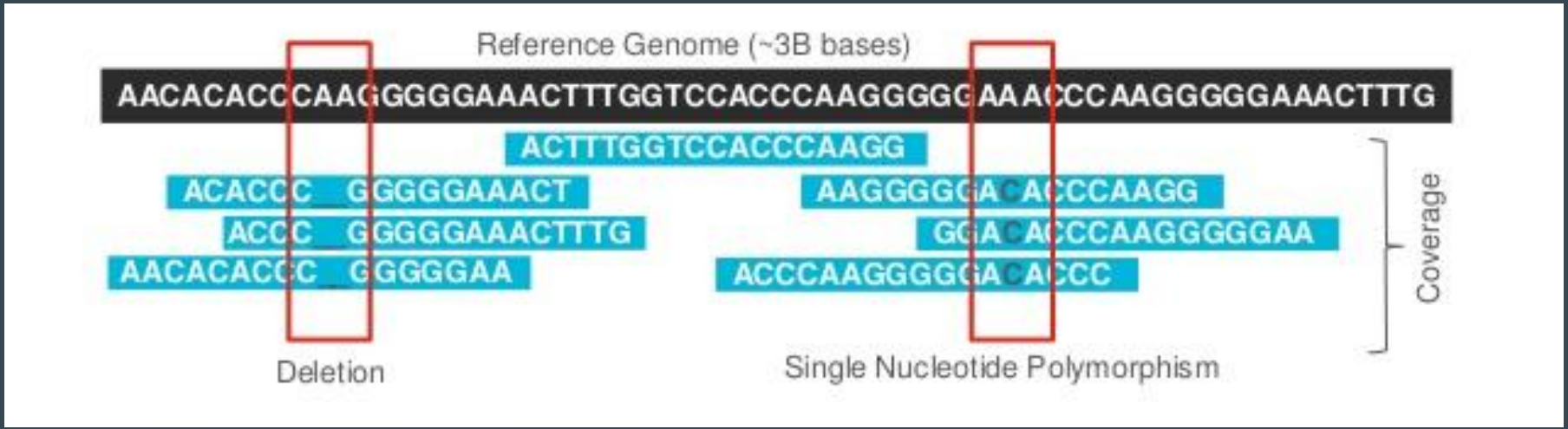
Current Technology (Shotgun)

- Split DNA into small pieces (reads).



Read Mapping

- Have access to a reference genome.
- Align reads to reference.

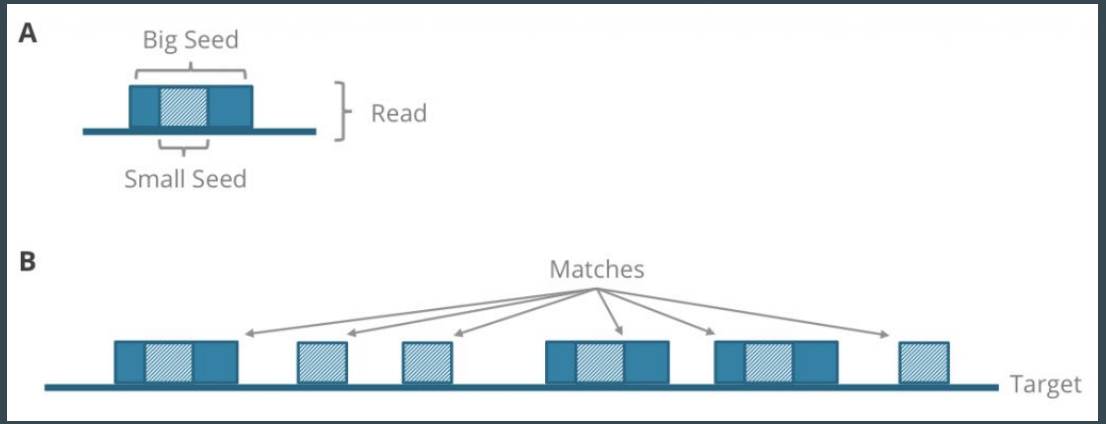


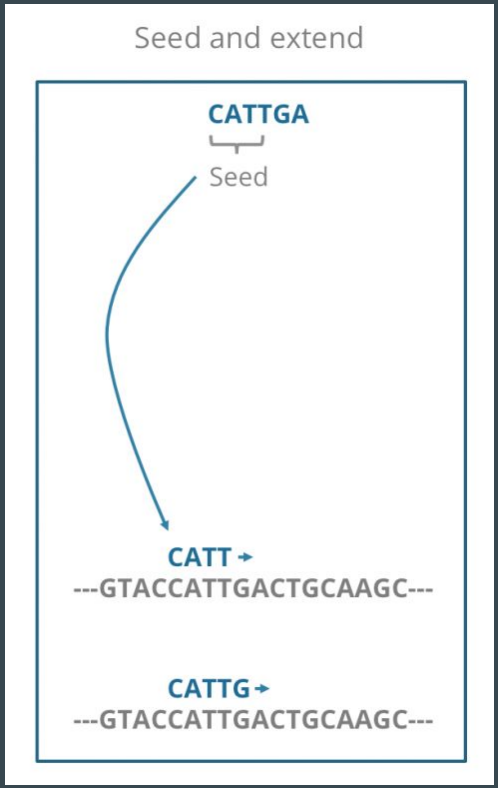
Computationally Challenging

- Billions of reads.
- Fuzzy matching.
  - Handle insertions, deletions, mutations, errors.
- Multiple mapping locations.
- Assemble with high probability.

How do we map a read? (Seed-and-extend Method)

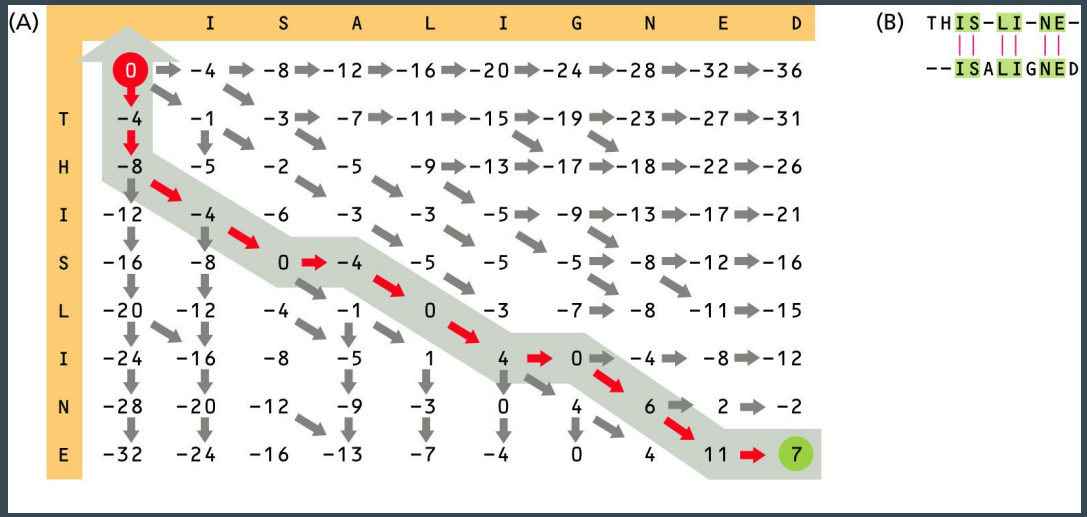
- Match substrings (seeds) exactly to the reference.
  - Possible locations.





How do we map a read? (Seed-and-extend Method)

- Use edit distance to determine quality.
  - Dynamic Programming (score based)
    - Needleman-Wunsch
    - Smith-Waterman
- Choosing less frequent substrings is important.



Research Project

- Improve the speed of the mapper (aligning reads).
- Develop novel algorithms and heuristics.
- Low complexity, memory efficient, cache efficient.

This Presentation

- Focuses on one part of the pipeline: Seed Selection.
  - Given set of reads, output seeds.
  - These are used to find potential mapping locations.
- Discuss parallel optimizations and improvements to runtime.

Parallelizing the Infrastructure

...

DNA Read Processing Pipeline

1. Generating the HashTree representation of genome.
2. Building a frequency predictor.
3. Performing seed selection.
4. Pipe results into next stage (edit distance).

Machine Specs

Test machine:

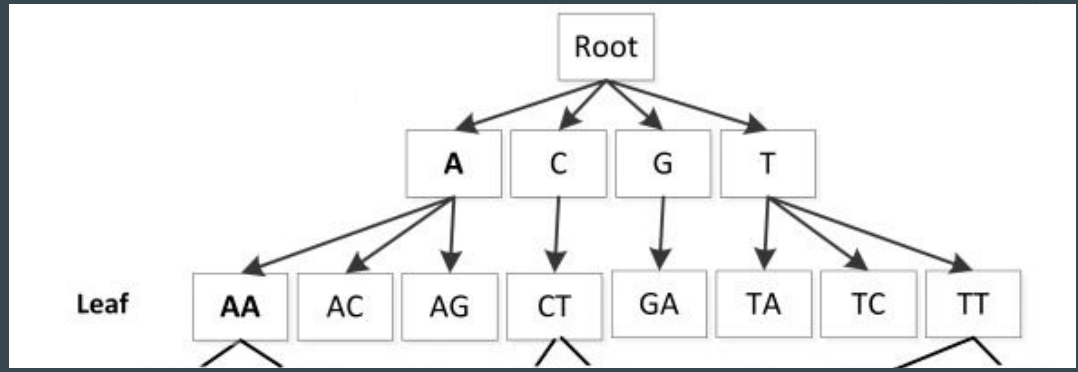
- 4 sockets.
- 10 cores and 256GB RAM (NUMA) per socket.
- 2 hardware threads per core (Intel Hyperthreading).
    - Memory latency.
- In total, 1TB RAM with 80 logical threads.

Generating the HashTree

...

Genome Representation

- Hashtable of Frequency Tries.
  - Each node stores a character and frequency.
  - Hashtable on first 10 letters.
- Bounded (length 30)
- ~80GB on disk.
- String frequency queries.
- L cache misses.



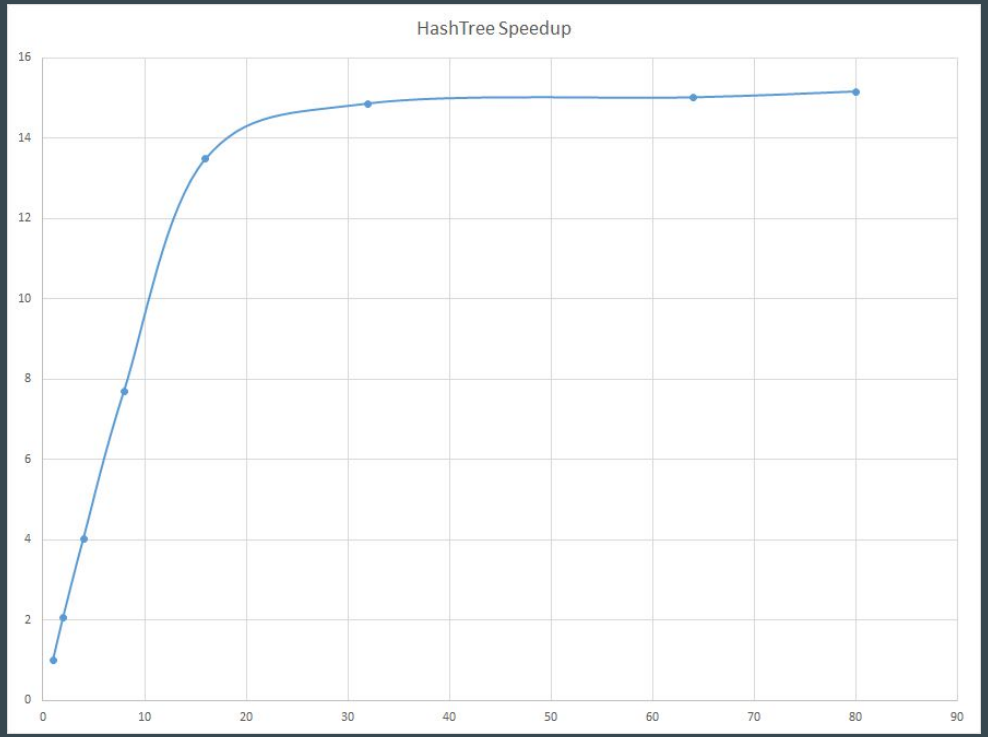
Generation in Parallel

- Reading from disk is sequential.
  - Threads take turn reading.
  - Memory mapped IO removes need for explicit copying.
  - Copied to Kernel page cache as opposed to user memory.
  - Incur TLB misses vs cache misses.
- Each trie can be independently generated.
  - Only issue is memory allocator.
  - Traditional malloc has a lock.
  - Only need alloc (not free).
  - Implement own allocator which is locality aware.
  - Was initial bottleneck. (2hrs to 10 minutes)

Generation in Parallel

- Dynamic work scheduling + greedy allocation.
  - Trie sizes are highly nonuniform.
  - Schedule largest tries first to balance workload.
    - Estimate from filesize.
- Kernel aware access policies.
  - TLB linear access.
  - File linear access.

Speedup Graph



Frequency Predictor

...

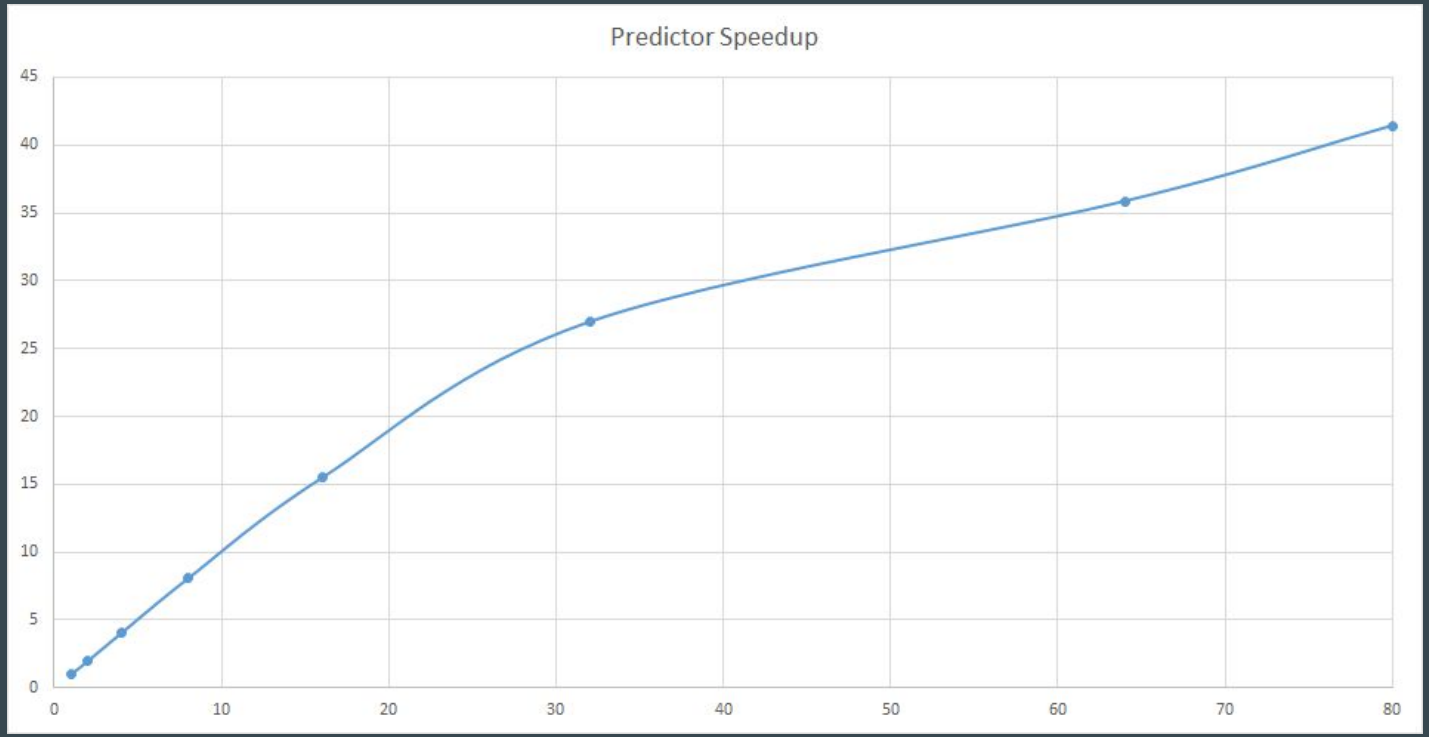
What is the Frequency Predictor?

- Access to HashTree is costly (L cache misses)
  - Instead, give an estimated frequency.
- Reduces to 1 cache miss.
- Store a table:
  - table[base][L][R] -> base (10 letters) extended to left by L letters, right by R letters.
- Example: AGCTGACG ATGCTAGCTA GCTCG
  - Lookup table[ATGCTAGCTA][8][5]

Construction of Predictor

- Requires traversing through the entire HashTree.
- Updating a large table
  - Synchronization at same entries.
  - Accomplished with atomic writes.

Speedup Graph



Seed Selection

...

What is Seed Selection?

- Given input set of reads, output set of seeds for each read.
  - Based on input parameters.
  - GCAGTCAGTCGATCGATCGATCGTACGTACGTACAGCTAGCTA
- Algorithms use mix of accesses to HashTree and predictor to determine seeds.

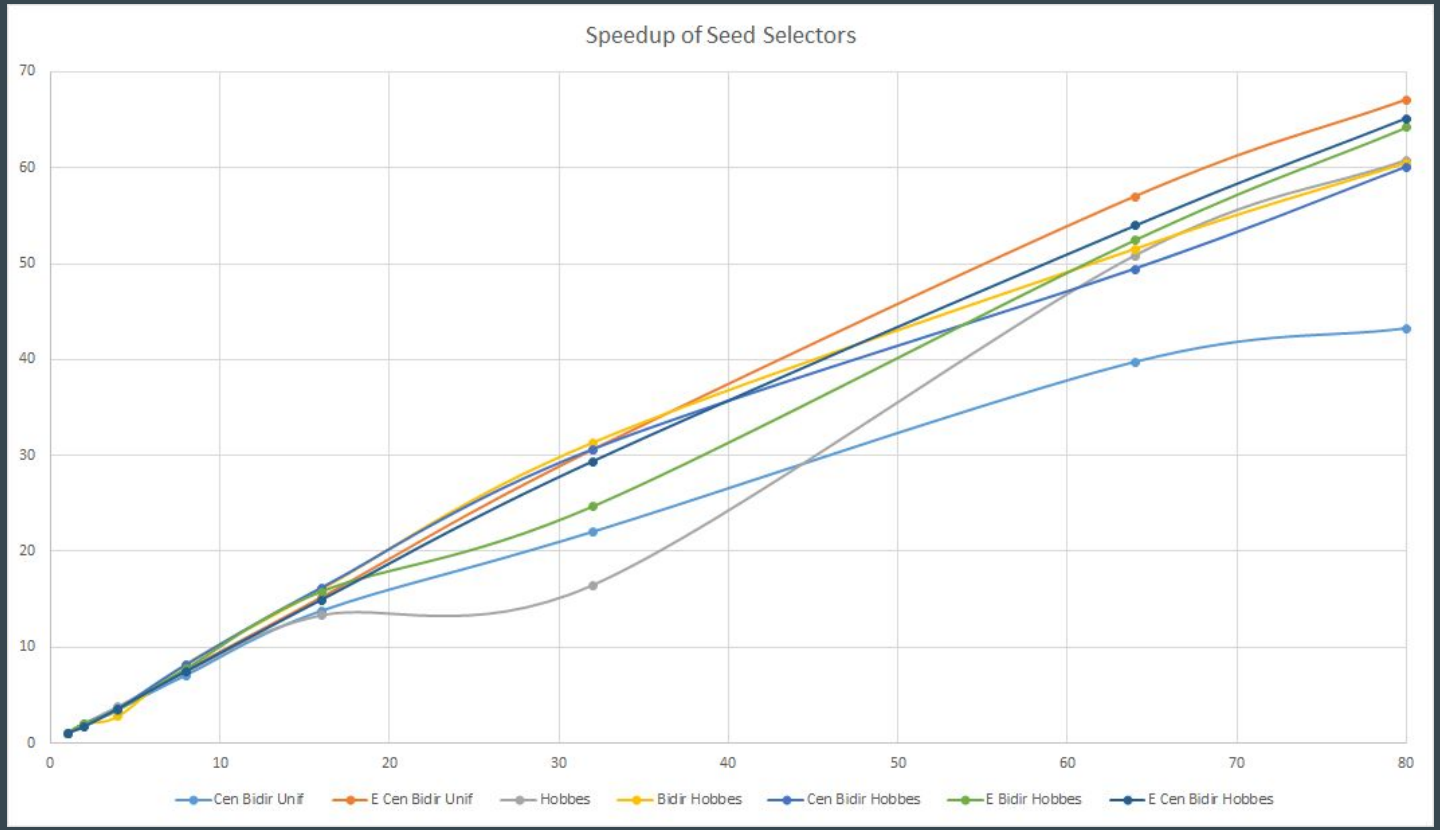
Parallelization

- Selection is parallelized over reads.
  - Per thread data structures, reduced at end.
- Both the HashTree and Predictor are loaded in memory.
  - Generally sparse accesses.
  - Cache write coherence is not an issue, since only read access to memory.
  - Cache reads create coherence traffic (costly on socket architecture).

Parallelization

- Stack memory vs malloc.
- NUMA degrades performance.
  - Threads closest to HashTree, Predictor perform much faster.
  - Observed up to 2x overhead.

Speedup Graph



Questions

...