Distributed OLC Genome Assembly With Ananas

Frank Austin Nothaft
fnothaft@berkeley.edu

Background

For long reads, genome assembly can be framed using the **OLC** paradigm:

1. Find **overlaps** between all reads
2. **Layout** these reads in a graph
3. Derive a **consensus** sequence through the graph

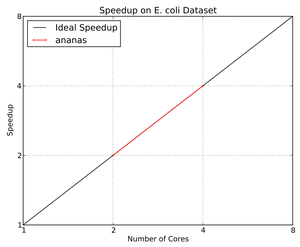
Goals:

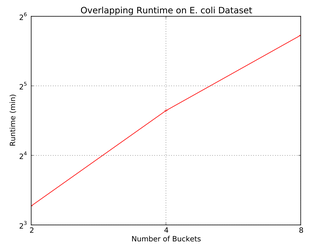
1. Reduce the cost of overlapping, naïvely $\mathcal{O}(n^2)$
2. Use commodity map-reduce to distribute graph processing
3. Enable parallel assembly for long read datasets

Performance

Pipeline Performance:

- Evaluated on 1.5GB E. coli dataset
- Evaluated using 2–4 EC2 `r3.2xlarge` instances
- Achieve linear speedup



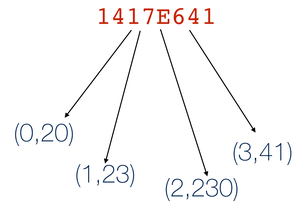


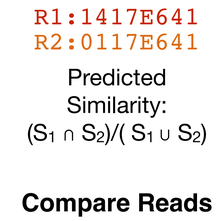
Overlapper Performance:

- Overlapper is ∼10% of runtime
- Runtime degrades with number of buckets used
- 2× increase in number of buckets leads to a 2× increase in data size and shuffle volume
- Naïve cartesian product overlapper does not complete

MinHash-Based Overlapping

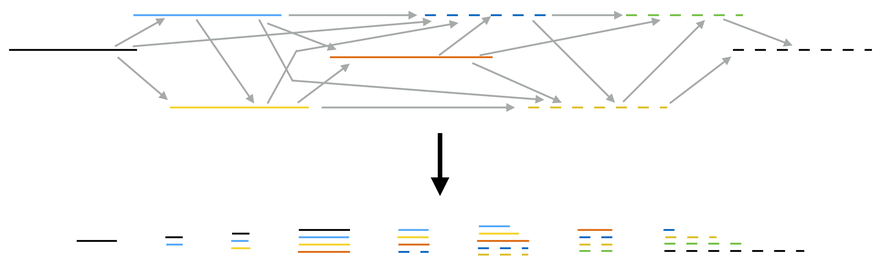
- Overlapping is an all-to-all comparison of reads to see if they contain overlapping sequence
- However, with proper indexing, we can eliminate many of the read vs. read comparisons
- We can effectively achieve this by applying LSH to signatures





Transitive Reduction

- Materialize all edges to vertex and use voting procedure
- Vertices vote to keep the "best" edges that satisfy their reduction conditions:
  - Final graph must keep all node-to-node *paths*
  - Pick *longest* edge (alignment overlap) that satisfied requirement



Transitive Reduction Process

Graph Traversal

- Graph traversal is inexpensive (∼5–10% runtime)
- Label all edges in connected component and then `groupBy` to assemble contig
- Labeling process is complex due to bi-directed nature of edges, and uncoordinated message passing
- Future work: add min cost flow algorithm for estimating copy number of edge/vertex