Team 3: superfast SV graph analysis

Baylor SV Hackathon, October 2019

Project Overview

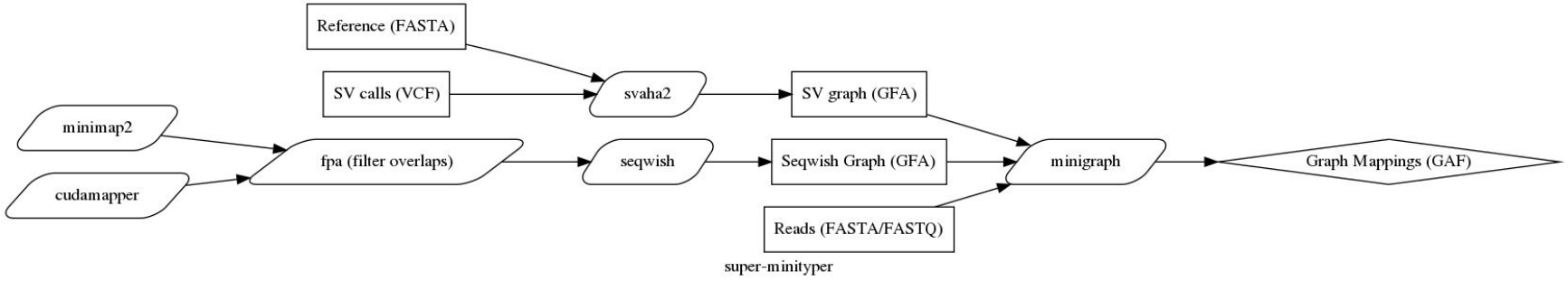
Graph genomes can improve structural variant analysis, but current implementations are too slow to be widely used.

Goal: make simple and fast SV graph analyses available in the cloud

Team:

Fernanda Foertter
Andreas Hehn
Joyjit Daw
Yuanqing feng
Yilei Fu
Qi Wang
Eric Dawson
Eric Venner
Gigon Bae

super-minityper: fast SV graph analyses

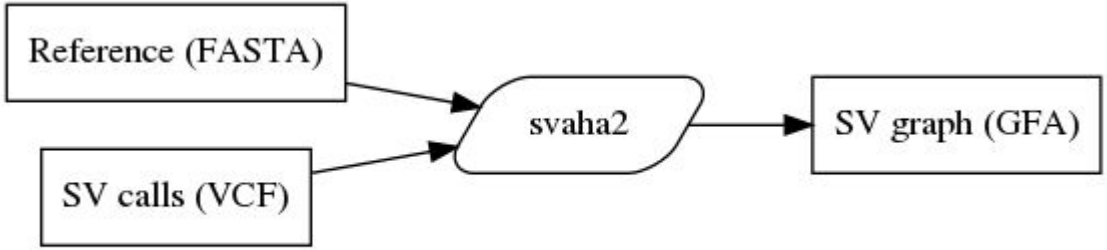


Designed to answer the question “Do my reads contain any of the SVs in the graph?”

- Genotyping of known structural variants / those in unannotated assembly graphs
- Provides a fast, easy-to-use way to answer this question compared to many previous approaches.

Generate a graph from an SV VCF

Utilizes svaha2, a fast (but limited) variation graph constructor.

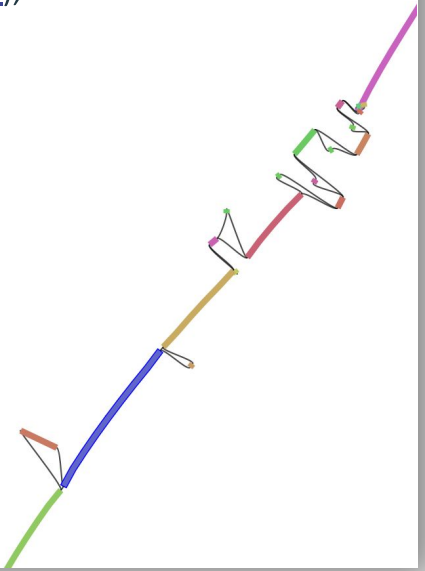


svaha2

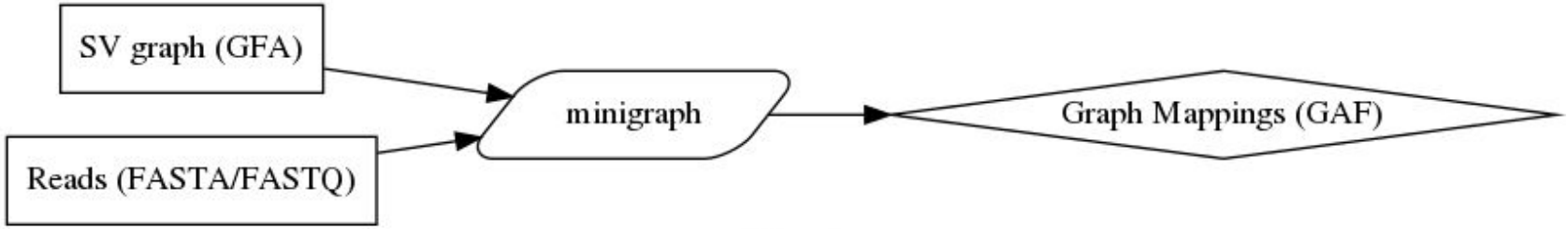
GIAB SV graph

Working from the GIAB v0.6 SV calls (https://www.biorxiv.org/content/10.1101/664623v3), we sorted, deduplicated, and munged the VCF to make one compatible with svaha2.

Chr10 graph, 3633 indels
(9 seconds to build with 64bp nodes)



Align reads with minigraph



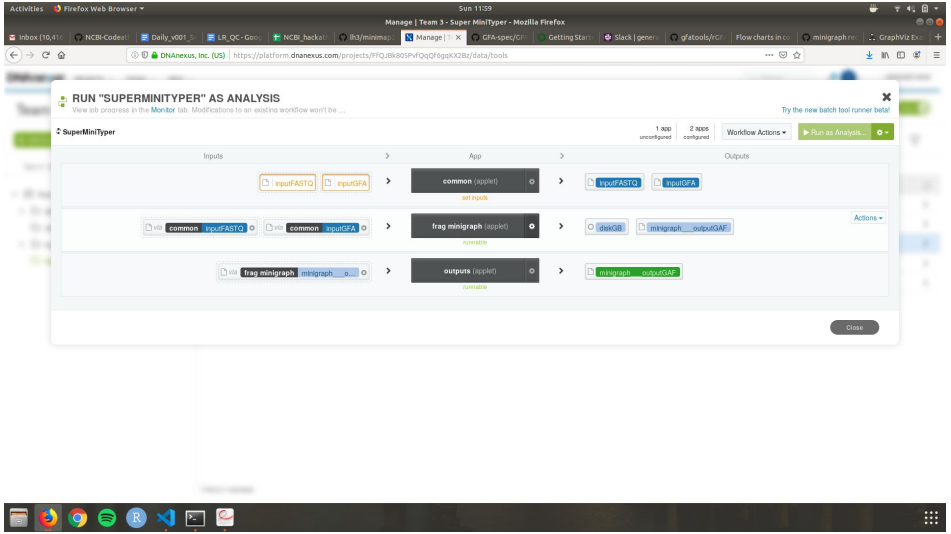
minigraph

https://github.com/lh3/minigraph

Aligning reads to the GIAB chr10 SV graph

We aligned ONT reads from HG002 (chr10 BAM) to the chromosome 10 HG002 graph.

Time to map >1 million reads from chr10:
15 minutes



Exploring the GAF output

Total time from VCF, FASTA and reads to GAF:
<20 minutes.

A simple analysis script is provided for basic GAF stats.

Branch: master super-minityper / analysis / gaf.py Find file Copy path

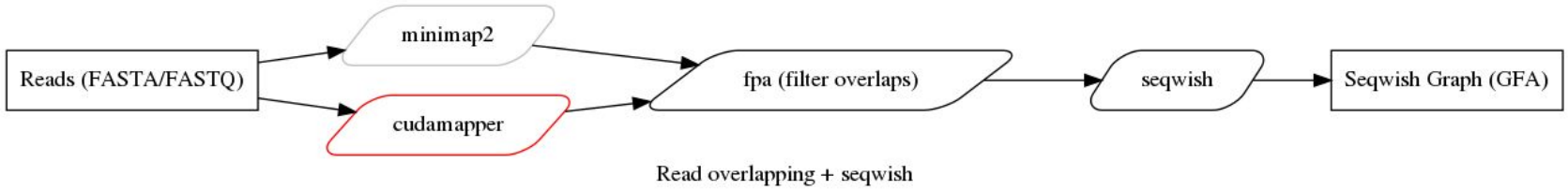
tiljoJwad [analysis] output some basic stats a393ed8 1 hour ago

1 contributor

41 lines (33 sloc) 1.39 KB Raw Blame History

```python
import sys

inputFile = sys.argv[1]

with open(inputFile) as f:
    #1   string  Query sequence name
    #2   int Query sequence length
    #3   int Query start (0-based; closed)
    #4   int Query end (0-based; open)
    #5   char    Strand relative to the path: "+" or "-"
    #6   string  Path matching /([><][^\s><]+(:\d+-\d+)?)+|([^\s><]+)/
    #7   int Path length
    #8   int Start position on the path (0-based)
    #9   int End position on the path (0-based)
    #10  int Number of residue matches
    #11  int Alignment block length
    #12  int Mapping quality (0-255; 255 for missing)

    total_alignments = 0

    align_perc_90 = 0
    align_perc_50 = 0
    align_perc_other = 0
    for line in f:
        row = line.strip().split("\t")
        querySeqName, querySeqLen, queryStart, queryEnd, strand, path, pathLen, startPosPath, endPosPath, numMatches, alignBlockLen

        total_alignments += 1

        # Find measure of overlap
        align_perc = float(alignBlockLen) / float(querySeqLen)
```

Generating long read assembly graphs

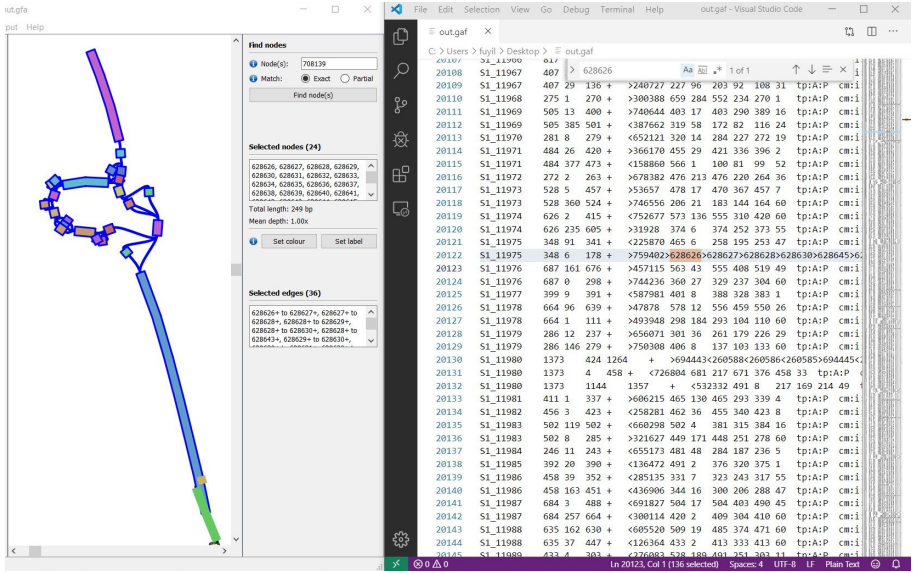


Sometimes, we won't have variant calls and/or a reference genome but may have long reads.
We can generate a graph to query by performing de novo graph assembly.
We can then map reads to this graph.

Graph alignment to metagenomic assembly graph

Out-of-cloud experiment:

1. Generate simulated PacBio reads from two close related genomes
2. Use minimap2+seqwish to generate GFA for the first genome
3. Map the second genome reads to the GFA
4. Found a read in the second genome that can be mapped to the bubble
5. Theoretically proved our pipeline is correct



Remaining issues

- svaha2 doesn’t yet support rGFA or tag nodes with their corresponding variant, so GAF output isn’t particularly useful
- svaha2 can’t currently run on DNAnexus (illegal instructions) - we’ll need help debugging that...
- svaha2 is very particular about the input VCF (variants must be unique, well-tagged, and non-adjacent)
- Cudamapper performance is dominated by shuttling data between GPU and CPU - lots of room for improvement
- Graph genomes are still in their toddler stages!

Future development

- Replace minimap2 with cudamapper (once its performance justifies such a swap)
- Calculate graph assembly statistics during the workflow with GFAKluge and report them to the user
- Add a script for genotyping SVs from the GAF output
  - Requires modifying the output of svaha2 to be valid rGFA with variant annotations
- Prove the pipeline on metagenomic samples and multiple whole-genome human samples
  - This was a goal initially and we think could still be very interesting