Design Principles for Scalable Genomic Analysis Systems

Frank Austin Nothaft

Abstract

Improvements in the throughput and cost of genomic sequencing techniques have made it practical to incorporate sequencing assays into research and clinical practice. However, current analysis toolkits cannot achieve the latency required for clinical work, nor the scale required for next-generation population genomics studies. By using the computational frameworks that have been developed for large-scale data analytics, we can design processing pipelines that achieve our latency/throughput goals by scaling across several hundred computers.

1 Introduction

1. Introduction
    (a) Application
        i. End-to-end pipeline for genomic analysis:
            - Input: raw reads from sequencer output
            - Align reads to reference genome
            - Process reads to de-skew data
            - Call variants
        ii. Variant call outputs are important in clinical use, and for population-level analysis
            - Variant calls describe distribution of alleles in population, as well as genotypes of individuals
            - Majority of called variants are single nucleotide polymorphisms (SNPs)
            - We are also interested in calling short insertions/deletions (INDELs) and copy number variations (CNVs, the duplication of a gene)
        iii. End-to-end latency is >50 hrs for current systems, this is highly limiting for clinical analysis
            - Sub-24hr latency is required for personalized medicine
            - Pipeline latency also limits throughput of sequencing assays on populations
    (b) Aims
        i. Develop a pipeline that can perform this processing end-to-end in one hour
        ii. Use distributed computing frameworks to make a system that can easily be scaled to hundreds of computers
        iii. Develop software under non-viral open source license to enable acceptance and use
2. Related Work
    (a) Computational Frameworks
        - Spark
        - Parquet

        - Avro

    (b) Variant Calling Frameworks
        - Genome Analysis Toolkit (GATK)
        - SAMTools
        - Short Oligonucleotide Analysis Package (SOAP)
        - FreeBayes

2 Architecture

1. Scalable analysis pipeline in Spark on top of ADAM—see fig. 1
2. Alignment
    (a) Alignment performed using SNAP using genome index
    (b) Alignment is distributed across n nodes inside of SNAP framework
    (c) Use interleaved FASTA input format to simplify division/streaming
    (d) To keep data from touching disk, open streams to/from SNAP
3. Preprocessing
    (a) Preprocessing serves to eliminate latent biases in reads
    (b) Preprocessing done using ADAM, described in §3
4. Variant Calling
    (a) Have implemented both pileup based and assembly calling methods
    (b) Support genomic region dependent processing methods—brief discussion of CAGE and SiRen
    (c) See §4

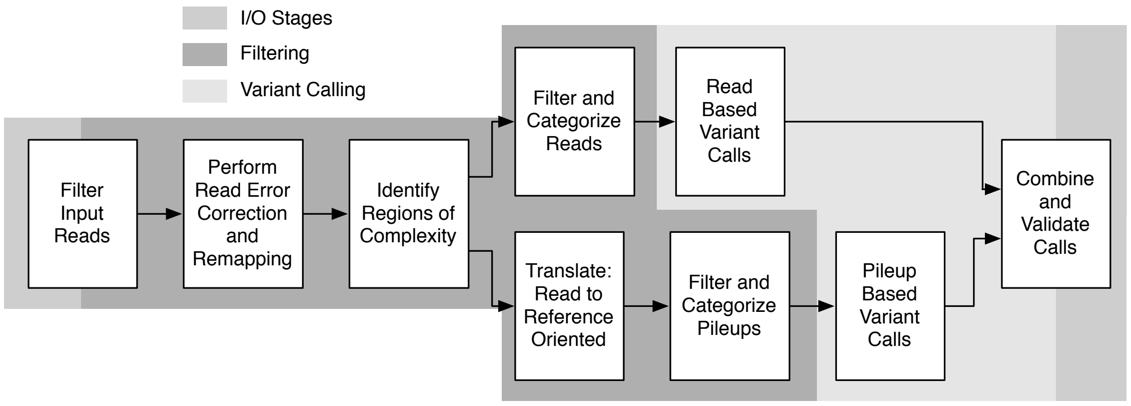
I/O Stages
Filtering
Variant Calling
Filter Input Reads
Perform Read Error Correction and Remapping
Identify Regions of Complexity
Filter and Categorize Reads
Read Based Variant Calls
Translate: Read to Reference Oriented
Filter and Categorize Pileups
Pileup Based Variant Calls
Combine and Validate Calls

Figure 1: System architecture

3 Read Preprocessing Algorithms

1. Duplicate Marking/Removal
    (a) Goal is to remove PCR/optical duplicate reads

   (b) Mostly an issue for WES/target assays
   (c) Describe heuristics for identifying duplicate reads—TBA
2. Indel Realignment
   (a) Short read alignment achieves correct global alignment but can have incorrect local alignment
   (b) Local realignment is a heuristic algorithm that cleans local alignments IFF an improvement in weighted Hamming distance can be achieved
   (c) Discuss different consensus sequence determination methods and impact on accuracy
   (d) Why is indel realignment needed when local assembly is available? Somatic variant calling.
3. Base Quality Score Recalibration
   (a) TBA

In this section (and the section below), go through discussion of algorithms and performance characteristics for above algorithms. Characterize:

- Across datasets: WGS (with PCR/PCR-free for duplicate marking), WES, targeted, high/low coverage, somatic/germline
- Across configuration parameters
- Across dataset sizes

4 Variant Calling Algorithms

1. Single Locus Calling
   - Qualitative comparison of current method vs. Unified Genotyper/FreeBayes/mpileup
   - Discussion of internal EM algorithm for maximization
   - Discussion of impact of joint variant calling as coverage changes—compare to CAGE++
2. Local Assembly
3. Copy Number Variant calling—TBA

5 Discussion

1. Comprehensive discussion of concordance/discordance between pipelines
   - SNAP/`avocado`
   - SNAP/GATK
   - BWA-MEM/GATK
   - BWA-MEM/FreeBayes
   - SOAP suite
2. Future Work
   - Somatic variant calling
   - Sufficient statistics/variant calling vs. database