About Me

• Data Science Team at Cloudera

• Apache Hadoop Committer, PMC Member, Apache Member

• Author of “Hadoop: The Definitive Guide”
What is genomics?
Reference chromosome

Location
“... decoding the Book of Life”
What is bioinformatics?
Bioinformatics!

> read1
TTGGACATTTCGGGTCTCAGATT
> read2
AATGTTGTTAGAGATCCGGGATTT
> read3
GGATTCCCCGCCGTTTGAGAGCCT
> read4
AGGTTGTACCCGCAAAAAGCGCAT
Pipelines!

- Alignment
- Dedup
- Recalibrate
- QC/Filter
- Variant Calling
- Variant Annotation
The 100,000 Genomes Project

The project will sequence 100,000 genomes from around 70,000 people. Participants are NHS patients with a rare disease, plus their families, and patients with cancer.

The aim is to create a new genomic medicine service for the NHS - transforming the way people are treated. Patients can benefit from applying what we learn from doing the research for the project.
It’s pipelines all the way down!
How can Hadoop be used in bioinformatics?
Genomics on Hadoop – A Potted History

• 2010 - Hadoop-BAM - MR input/output formats for bio (BAM, VCF, etc)
• 2011 - Seal - MR tools for reads
• 2012 - SeqPig - Pig interface for Hadoop-BAM
• 2013 - ADAM - a genomics analysis platform on Spark, Avro, and Parquet
• 2013 - OpenCGA - a variant store built on HBase
• 2014 - Halvade - a tool to run the GATK best practices pipeline using MR
• 2014 - Guacamole - Spark variant caller for ADAM
• 2015 - GATK4 - a toolkit for running genomics pipelines on Spark
• 2016 - Hail - PLINK-like tool for whole genome association analysis
Spark + Genomics = ADAM

- Hosted at Berkeley and the AMPLab
- Apache 2 License
- Contributors from both research and commercial organizations
- Core spatial primitives, variant calling
- Avro and Parquet for data models and file formats

cloudera
Genome Analysis Toolkit (GATK)

• Developed by the Broad Institute
• Core is MIT license, some proprietary tools on top
• Version 4 has been re-written to use Spark, now competitive with ADAM for speed
• Uses existing bio file formats for input and output, but Spark RDDs for intermediate data
Bioinformatics File Formats

• Hand crafted
• Poorly specified
• Text based
• Unsplittable (in the Hadoop sense)
BAM files

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001 99 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCCTAAGCTAA * SA:Z:ref,29,,-,6H5M,17,0;
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 2064 ref 29 17 6H5M * 0 0 TAGGC * SA:Z:ref,9,+,5S6M,30,1;
r001 147 ref 37 30 9M = 7 39 CAGCGGCAT * NM:i:1
```
Example: splitting BAM files

BAM records ("reads")

Repeatedly advance by 1 byte until next record is found, and confirm reading 3 records.
Example: splitting BAM files (BGZF compression)

BGZF block

header

header

header includes block length (as gzip extension), so easy to index and split
Why not use Hadoop formats?
Dedup
Mark Duplicates

Fragment 1
- 1
- 1
- 2
- 1
- 2

Fragment 2
- 1
- 2

reads

sort

reads

1
1
2
1
2

1
1
2
2

1
1
2

reads

dedup

1
2
/**
 * Main work method. Reads the BAM file once and collects sorted information about
 * the 5' ends of both ends of each read (or just one end in the case of pairs).
 * Then makes a pass through those determining duplicates before re-reading the
 * input file and writing it out with duplication flags set correctly.
 */
protected int doWork() {
    // build some data structures
    buildSortedReadEndLists(useBarcodes);
    generateDuplicateIndexes(useBarcodes);

    final SAMFileWriter out =
        new SAMFileWriterFactory().makeSAMOrBAMWriter(outputHeader, true, OUTPUT);
    final CloseableIterator<SAMRecord> iterator = headerAndIterator.iterator;
    while (iterator.hasNext()) {
        final SAMRecord rec = iterator.next();
        if (!rec.isSecondaryOrSupplementary()) {
            if (recordInFileIndex == nextDuplicateIndex) {
                rec.setDuplicateReadFlag(true);
                // Now try and figure out the next duplicate index
                if (this.duplicateIndexes.hasNext()) {
                    nextDuplicateIndex = this.duplicateIndexes.next();
                } else {
                    // Only happens once we've marked all the duplicates
                    nextDuplicateIndex = -1;
                }
            } else {
                rec.setDuplicateReadFlag(false);
            }
            recordInFileIndex++;
        }
        if (!this.REMOVE_DUPLICATES || !rec.getDuplicateReadFlag()) {
            out.addAlignment(rec);
        }
    }
}
@Option(shortName = "MAX_FILE_HANDLES",
    doc = "Maximum number of file handles to keep open when spilling " +
    "read ends to disk. Set this number a little lower than the " +
    "per-process maximum number of file that may be open. This " +
    "number can be found by executing the 'ulimit -n' command on " +
    "a Unix system.")

public int MAX_FILE_HANDLES_FOR_READ_ENDS_MAP = 8000;
Spark Implementation

JavaPairRDD<String, Iterable<GATKRead>> keyedReads = ...;

JavaPairRDD<String, PairedEnds> keyPairs =
    keyedReads.flatMapToPair(keyedRead -> {
        ... 
    });

JavaPairRDD<String, Iterable<PairedEnds>> keyedPairs =
    keyPairs.groupByKey(numReducers);

JavaRDD<GATKRead> markedDups = markPairedEnds(keyedPairs,
    scoringStrategy, finder, header);
Lessons Learned

1. Figure out how to read and write existing formats efficiently
2. Spark is a great API, but developers need to understand consequences of e.g. the shuffle, serialization cost
3. Work with domain experts, on existing projects, if possible
Future developments
Kudu for Variant Stores

- Kudu fills gap between HDFS and HBase
- Fast scans and updateable
- Add new annotations to genomics data (variants) without rewriting whole dataset
- Key = genome position
- Range partitioning
Hail

• Scalable variant analytics in Spark
• Command-line tools like PLINK
• Parquet-based storage by default, other storage possibilities like Kudu
Links

- ADAM
  - https://github.com/bigdatagenomics/adam
- GATK4
  - https://github.com/broadinstitute/gatk
Acknowledgements

UCBerkeley
Matt Massie
Frank Nothaft
Michael Heuer

MSSM
Jeff Hammerbacher
Ryan Williams

Tamr
Timothy Danford

Cloudera
Uri Laserson
Sandy Ryza
Sean Owen
Thank you
@tom_e_white
tom@cloudera.com