Big Data Processing for Genomics

Altti Ilari Maarala[1], **Keijo Heljanko**[1], André Schumacher[1,2], Ridvan Döngelci[1], Luca Pireddu[3], Matti Niemenmaa[1], Aleksi Kallio[4], Eija Korpelainen[4], and Gianluigi Zanetti[3]

[1] Helsinki Institute for Information Technology HIIT and Department of Computer Science, Aalto University
firstname.lastname@aalto.fi
[2] International Computer Science Institute, Berkeley, CA, USA
[3] CRS4 — Center for Advanced Studies, Research and Development, Italy
[4] CSC — IT Center for Science

27.11-2016

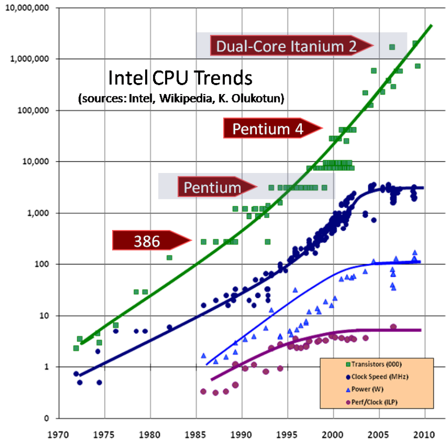
Next Generation Sequencing and Big Data

- The amount of NGS data worldwide is predicted to double every 5 months
  - This growth is much faster than Moore's law (was) for the growth rate of computing (historically transistor counts have doubled every 18-24 months until now)
- 1000 Genomes project has Petabytes of human genomes data sets
- In many GWAS and WGS studies multiple large files (100+ Gigabytes) has to be processed sequentially
- NGS analytics methods has to cope with the data growth rate $\Rightarrow$ Towards distributed computing methods and parallel algorithms to avoid hitting computational limits

Repealing Moore's law

- The number of transistors in a core and the clock speeds of microprocessors are not growing much anymore.
- Smaller transistors have given speed and power consumption advantage (switching on/off states is faster), but now sizes are reaching physical limits (14nm, Intel Broadwell 2014) causing overheating, gate leakage etc. New technologies not yet mature or cost efficient enough. $\Rightarrow$ Increasing concurrency at multiple levels: the number of computing cores in a processor, number of processors in a computer, amount of computers in a cluster $\Rightarrow$
- Programming models need to change to efficiently exploit all the parallelism - scalability to high number of cores/processors is a major focus

No Processor Clock Speed Increases Ahead



- Herb Sutter: The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software. Dr. Dobb's Journal, 30(3), March 2005 (updated graph in August 2009).

Tape is Dead, Disk is Tape, RAM locality is King

- RAM (and SSDs) are radically faster than HDDs: One should use RAM/SSDs whenever possible
- RAM is roughly the same price as HDDs were a decade earlier
  - Workloads that were viable with hard disks a decade ago are now viable in RAM
  - One should only use hard disk based storage for datasets that are not yet economically viable in RAM (or SSD)
  - The Big Data applications (HDD based massive storage) should consist of applications that were not economically feasible a decade ago using HDDs

Hadoop - Linux of Big Data

- Hadoop is Open Source distributed data processing system
  - Based on Google's MapReduce architecture design
  - Cheap commodity hardware for storage
  - Fault tolerant distributed filesystems: HDFS, Tachyon
  - Batch processing systems: Hadoop MapReduce, Apache Hive, and Apache Pig (HDD); Apache Spark (RAM)
  - Parallel SQL implementations for analytics: Apache Hive, Cloudera Impala, Apache Shark, Facebook Presto
  - Fault tolerant distributed database: HBase
  - Distributed machine learning libraries, text indexing & search, (Mahout, Solr etc.)
- Hadoop MapReduce is just one example application on top of the Hadoop Open Source distribution!

Commercial Hadoop Support

- Salability: Hundreds Petabytes of storage deployed on single HDFS installation (Facebook today 300PB+), 4000+ DataNodes (Yahoo!) with 10 000+ hard disks and 30 000+ cores. "HDFS scalability: the limits to growth", K.V.Shvachko.
- Cloudera: Probably the largest Hadoop distributor, partially owned by Intel (740 million USD investment for 18% share). Available from: `http://www.cloudera.com/`
- Hortonworks: Yahoo! spin-off from their large Hadoop development team: `http://www.hortonworks.com/`
- MapR: A rewrite of much of Apache Hadoop in C++, including a new filesystem. API-compatible with Apache Hadoop. `http://www.mapr.com/`

Apache Spark

- General in-memory Big Data processing engine and parallel programming framework.
- Runs on Hadoop and Mesos or standalone, in local cluster or in the cloud.
- Can access diverse data sources eg. HDFS, S3, Cassandra, HBase, Impala, Hive.

Apache Spark

- Based on functional programming with Scala, Java, Python, also R or SparkR.
- Operates with Resilient Distributed Datasets (RDDs) $\Rightarrow$ Fault tolerant parallel data processing in main memory.
- Running iterative algorithms rapidly in main memory instead of hard disks. 10-100x faster than Hadoop MR.
- Caching, fast recovery from failures, easy management and versatile API.

Spark extensions

- Dataframes/Spark SQL - Module for querying structured data with SQL and Dataframe API. Enables filtering, searching, merging with dataframes (tables in DB)
    - Query avg read coverage: *SELECT AVG(depth) FROM pileups WHERE seqID=chr1 AND pos BETWEEN 10000 AND 40000*
- MLlib - Distributed Machine learning Library
- GraphX - Parallel graph processing system
- SparkR - R programming API for Spark
- Spark Streaming - A Streaming Data Processing Framework
- Piping of external standard tools and commands to run in parallel

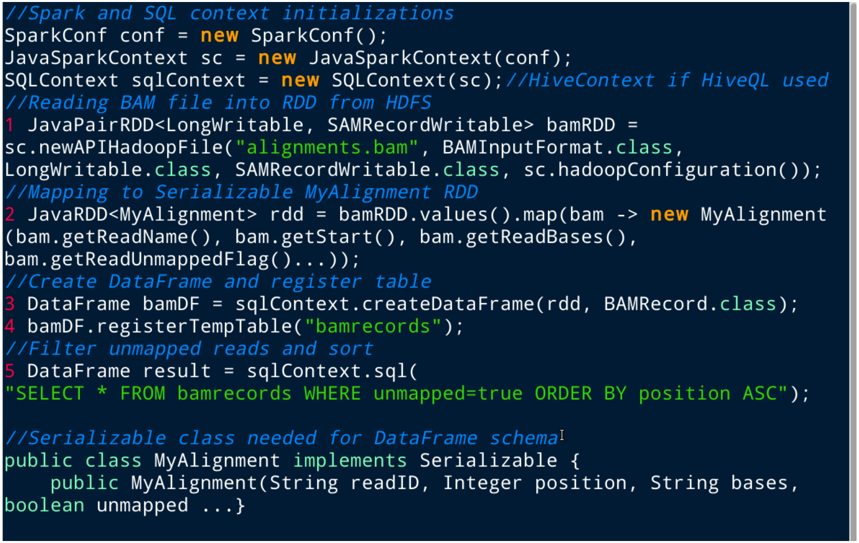
Hadoop-BAM

- A library developed originally in Aalto University for processing NGS data formats in parallel with both Hadoop and Spark
- Includes Hadoop I/O interface and tools for e.g., sorting, merging, filtering read alignments
- Supported fileformats: BAM, SAM, CRAM, FASTQ, FASTA, QSEQ, BCF, and VCF
- Released in Dec 2010, Latest version 7.7.2: `https://github.com/HadoopGenomics/Hadoop-BAM`.
- Used in GATK4(Broad), Adam(UC Berkley), Halvade(Ghent), Seal(CRS4) and SeqPig(Aalto)
- "Hadoop-BAM: Directly Manipulating Next Generation Sequencing Data in the Cloud." Niemenmaa, M., Kallio, A., Schumacher, A., Klemelä, P., Korpelainen, E., and Heljanko, K. Bioinformatics 28(6):876-877, 2012. (`http://dx.doi.org/10.1093/bioinformatics/bts054`).

Hadoop-BAM Integrated

- Hadoop-BAM can be interfaced with SparkSQL and HiveQL allowing SQL queries over genomic data formats
- NGS data can be provided in columnar formats such as RCFile or Parquet for improved compression and query performance
- Using Parquet or RCFile storage also allows BAM files to be queried by other engines such as Impala and Presto.
- ”Analysing sequencing data in Hadoop: The road to interactivity via SQL” Niemenmaa M., Master’s Thesis, Aalto University, 2013.

Hadoop-BAM with SparkSQL example

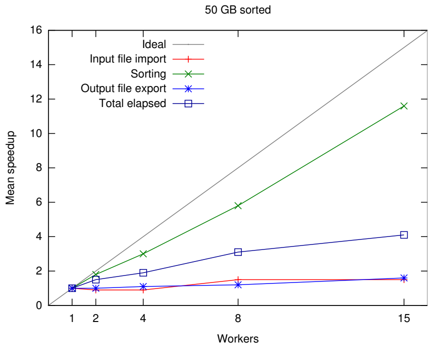


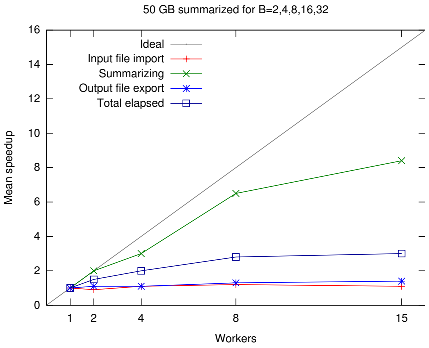
```
//Spark and SQL context initializations
SparkConf conf = new SparkConf();
JavaSparkContext sc = new JavaSparkContext(conf);
SQLContext sqlContext = new SQLContext(sc);//HiveContext if HiveQL used
//Reading BAM file into RDD from HDFS
1 JavaPairRDD<LongWritable, SAMRecordWritable> bamRDD =
sc.newAPIHadoopFile("alignments.bam", BAMInputFormat.class,
LongWritable.class, SAMRecordWritable.class, sc.hadoopConfiguration());
//Mapping to Serializable MyAlignment RDD
2 JavaRDD<MyAlignment> rdd = bamRDD.values().map(bam -> new MyAlignment
(bam.getReadName(), bam.getStart(), bam.getReadBases(),
bam.getReadUnmappedFlag()...));
//Create DataFrame and register table
3 DataFrame bamDF = sqlContext.createDataFrame(rdd, BAMRecord.class);
4 bamDF.registerTempTable("bamrecords");
//Filter unmapped reads and sort
5 DataFrame result = sqlContext.sql(
"SELECT * FROM bamrecords WHERE unmapped=true ORDER BY position ASC");

//Serializable class needed for DataFrame schema
public class MyAlignment implements Serializable {
    public MyAlignment(String readID, Integer position, String bases,
boolean unmapped ...}
```

Aalto University
School of Science

Mean speedup



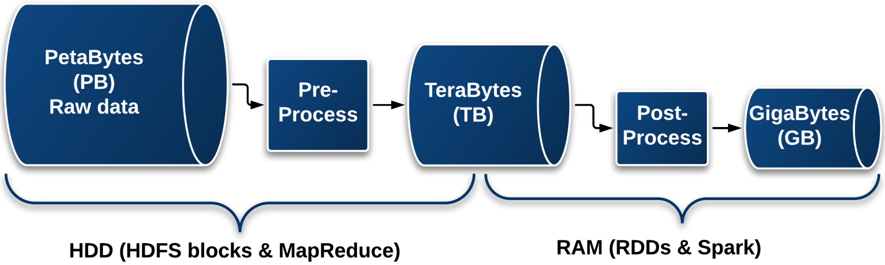


- Note that multiple I/O operations reduce the overall performance seriously. However, total running time for summarizing read coverages from 50GB of BAM file pileups stayed under 1 hour with 8 nodes.

Characteristics of genomics data

- Old sequential algorithms and models (BWT, Bowtie, HMM, assembly algorithm etc.) badly or not at all parallelizable
- Data parallelism only choice for parallel processing without rewriting algorithms
- Genomics data usually parallelizable in chromosomal level and regions by gene locuses + distributing raw reads and alignments
- Deliver code to the data, do not move elephant if not really needed!
- File formats not designed for distributed file systems (especially binary formats BAM, BCF, BED)
- Hadoop-BAM can already handle distributed BAM and BCF files on HDFS in parallel (also in-memory with Spark)
- Typically whole reference index has to be provided for map phase, e.g. aligning chunk of reads $\rightarrow$ New compressed indexing techniques for reference needed, Lempel-Ziv and suffix trees found efficient with genomes.

Typical genomics pipeline



- Processing data in main memory instead of files in hard disks $\Rightarrow$ minimal I/O operations. Map/Reduce data from Petabytes to Gigabytes (million times less in the end!)

General parallel pipelines for genomics

- Broad Institute's GATK4 integrates widely-used tools to be run in parallel on clusters using Apache Spark. Relies on Hadoop-BAM I/O.
  - Current implementation has been focusing mostly on Variant discovery functionalities
  - alignment and variant files can be processed in parallel, includes e.g. sorting, duplicate marking, realignment and variant calling.
- ADAM from UC Berkeley includes basic tools for file transformations, k-mer counting, allele frequencies on Apache Spark. Uses Hadoop-BAM for I/O.
- Halvade uses Broad Institute's best practices pipeline on Hadoop MapReduce. Hadoop-BAM I/O.

Halvade

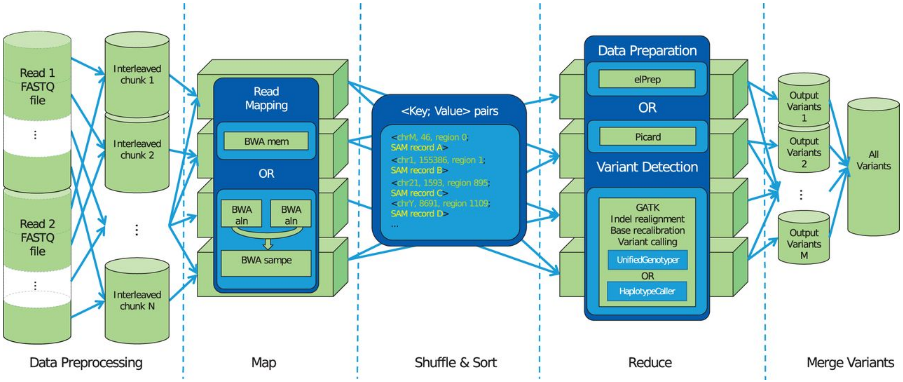
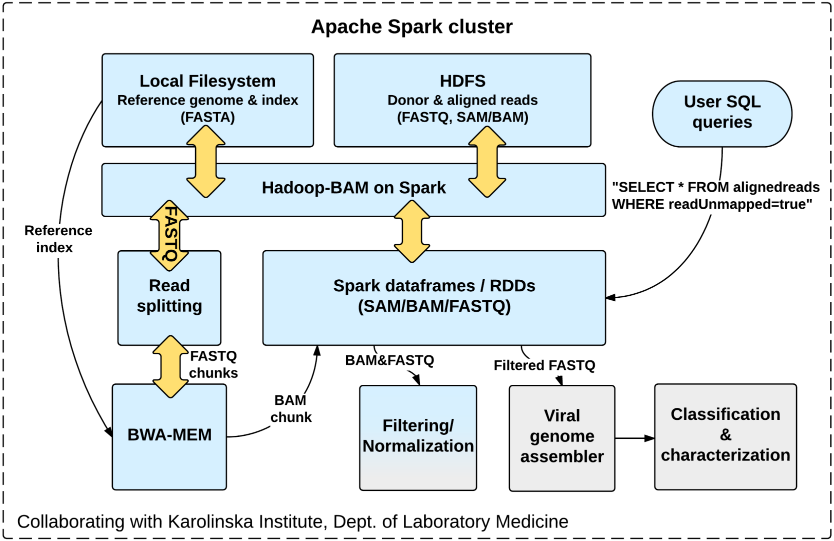


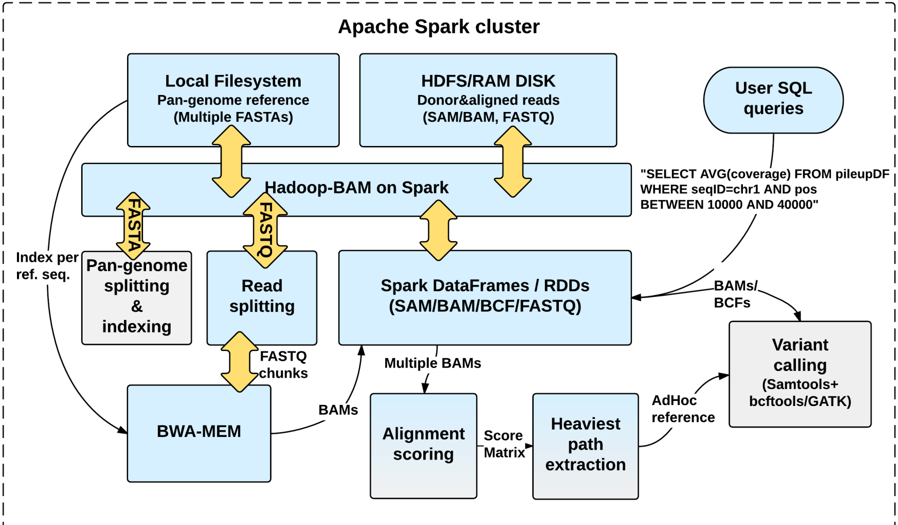
Figure: *Halvade: scalable sequence analysis with MapReduce.* D. Decap, J. Reumers, C. Herzeel, P. Costanza, J. Fostier. Bioinformatics (2015) 31 (15): 2482-2488.

Parallel Pipeline for Metagenomics



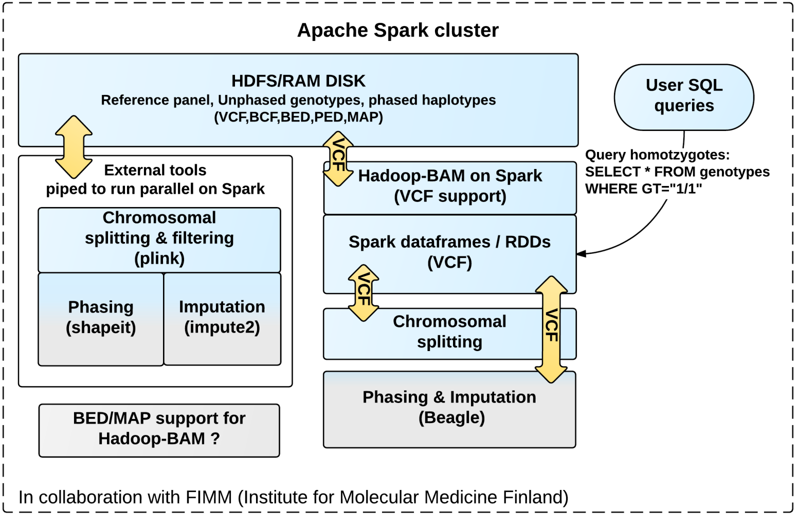
Collaborating with Karolinska Institute, Dept. of Laboratory Medicine

Parallel Pipeline for Pan-Genomics



Collaborating with Helsinki University, Genome-scale algorithmics group (Veli Mäkinen et al.)

Parallel Pipeline for Genotype Imputation


Apache Spark cluster
HDFS/RAM DISK
Reference panel, Unphased genotypes, phased haplotypes
(VCF,BCF,BED,PED,MAP)
User SQL queries
Query homotzygotes:
SELECT * FROM genotypes
WHERE GT="1/1"
External tools
piped to run parallel on Spark
Chromosomal splitting & filtering (plink)
Phasing (shapeit)
Imputation (impute2)
BED/MAP support for Hadoop-BAM ?
VCF
Hadoop-BAM on Spark (VCF support)
Spark dataframes / RDDs (VCF)
VCF
Chromosomal splitting
VCF
Phasing & Imputation (Beagle)
In collaboration with FIMM (Institute for Molecular Medicine Finland)

Future plans

- Implementing parallel variant discovering by following best practice methods for our pipelines
- Extending pipelines with parallel De-novo assembler on Spark
- Classification of viruses on Spark with MLLib
- Spark implementation of parallel Relative LZ indexing for compressed reference index
- Applying standard columnar data formats like Parquet and RCFile into our pipelines for NGS data warehousing
- We are open to working with you on Parallel Next Generation Data Processing on Hadoop and Spark Ecosystems