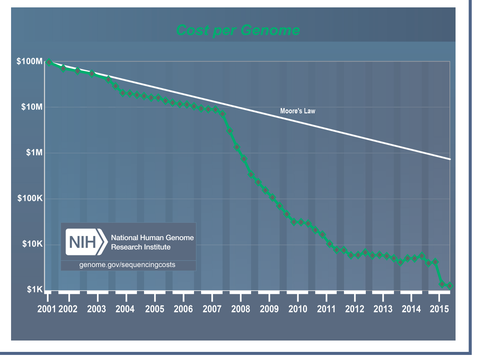
Scalable genomics
From Raw Data to Aligned Reads on Apache Yarn

Luca Pireddu*, Francesco Versaci and Gianluigi Zanetti
CRS4, Polaris, Ed. 1, I-09010 Pula, Italy
*luca.pireddu@crs4.it

Motivation

- Look at the falling cost of sequencing!
- We will be sequencing more and more as it gets cheaper
- Example: population-wide sequencing applications
  - e.g., sequencing for pathology, personalized medicine, etc.
- These applications require very scalable processing solutions



Introduction

We built a scalable sequence alignment pipeline based Hadoop ecosystem
- Flink, HDFS, MapReduce and Yarn

Our solution is:
- distributed and scalable (runs on many computers, efficiently)
- robust (resists hardware failures).

We're on the verge of connecting it to GATK4 to complete a Yarn-based variant calling pipeline

NGS Alignment Pipeline

BCL conversion

Data not written to disk between steps

Demultiplexing

Alignment

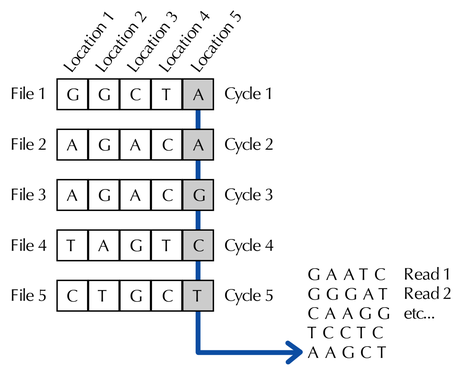
Pipeline is then connected to downstream analysis for variant calling
- Currently SAM output
- Working on CRAM output
- Compatible with GATK4 on Spark

NGS

- NGS produces large amounts of data
  - a single sequencer -> 1 TB/day of raw data
  - these data need to be processed
- Illumina's base calling process produces data in BCL format
- BCL files are arrays of bytes, 1 per tile/cycle
- Each byte encodes base and quality score
- BCL files have associated:
  - .filter files (QC result per read)
  - .locs files (contain metadata)

BCL conversion

- First step in processing: compute reads directly from BCL files
  - akin to a matrix transposition
- Conventionally performed by Illumina's bcl2fastq
- We have a Flink-based distributed implementation written in Scala
- Scalable across many computers, and faster



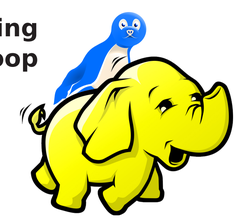
Demultiplexing

- Separate reads by (id tag, lane)
- Sorts the data from various samples sequenced in same run
- Conventionally done by Illumina's bcl2fastq program
- We've implemented a new Flink-based program in Scala to perform this
- Scalable and faster

Alignment

Our pipeline aligns using Seal's seqal aligner
- Integrates BWA-MEM through RAPI plugin and Pydoop
- Runs as a map-only job on Hadoop MapReduce
- Scalable and robust to hardware failures

Seal - NGS processing on Hadoop



Tech Background

- Big Data technologies can boost the scalability of data-driven biology and health workflows by orders of magnitude
- The Hadoop/Yarn platform has already proven its effectiveness
  - successfully used in data-driven industry for much larger processing than any current biological- or health-driven work
- Hadoop/Yarn ecosystem – large number of components, and growing
- Yarn: resource manager, providing a distributed computing platform

MapReduce

- Well known framework for distributed batch computation
- Implements MapReduce paradigm
  - Algorithm formed by a map function and a reduce function
- Runs on Yarn platform

Flink

- New streaming computation framwork
- Very scalable and efficient
- Can do batch computing as a special case
- Has been adopted by Netflix and Alibaba, among others
- Also runs on Yarn platform

Flink tips

While developing this solution on the new Flink platform we learned some valuable lessons.

Try DataStream API even if it doesn't seem like a natural fit
- We found it faster than DataSet API

Avoid fine granularity (e.g., reading bytes)
- much faster to process larger chunks
  - exploit data locality

Keep Flink job units reasonably small
- one minute on one core

Evaluation

We evaluated our pipeline performance and scalability

Equipment

- Amazon EC2, with up to 14 r3.8xlarge nodes
  - 32 virtual cores, 250 GB RAM, 2x320 GB SSD, 10 Gbit Ethernet
- HDFS over cluster nodes
- Yarn over same n nodes
- Flink and MapReduce running on Yarn

Dataset

- Multiplexed run from an Illumina HiSeq 3000 (48 DNA samples)

Baseline

- Pipeline implemented with bcl2fastq2 and bwa-mem
- single-node, multithreaded

Running times

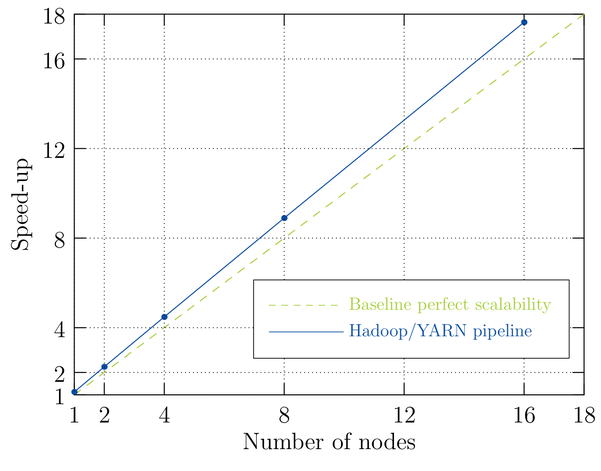
Running times of our Hadoop/Yarn pipeline and the baseline (on a single node).

	Time (minutes)		
Nodes	BCL	Align	Total
baseline	67,3	645	713
1	47,3	571	618
2	24,5	286	311
4	13,5	144	158
8	8,1	72,5	80,7
16	6,1	36,6	42,8

We found that our pipeline is faster than the baseline, even on a single node. Why?
- We use tab-separated PRQ files instead of Fastq
- Baseline BWA has less I/O-CPU overlap
- Seal mmap's the reference data
- Our pipeline uses tens of concurrent read and write streams over both SSDs, per node (through HDFS); baseline has four in bcl2fastq2 and two read and one write streams in the alignment

Scalability

The graph shows the speed-up and absolute scalability of our Yarn-based pipeline w.r.t. the number of computing nodes



Conclusions

- Experiments demonstrate the scalability of our approach
- Scalability will be mandatory to enable large-scale sequencing applications
- Thanks to our careful implementation and to the well-performing Yarn-based frameworks, our pipeline shows advantages even when running on a single node

Read the paper:
http://biorxiv.org/content/early/2016/08/23/071092

Seal is on github: https://github.com/crs4/seal

Future work

- Integrate Flink code in Seal
- Finish CRAM output and connect to GATK4
- Flink-based alignment
- Streaming sequence processing