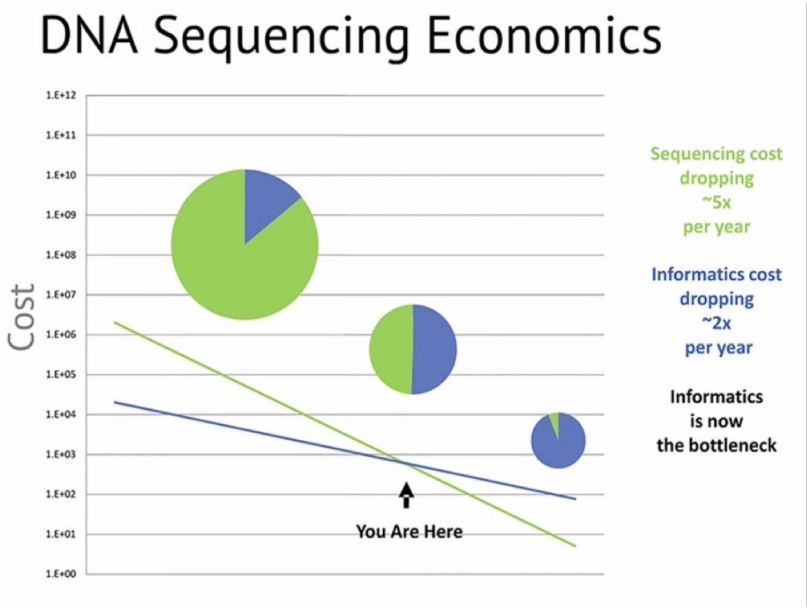
Hummingbird

Introduction

- Advances in medical computing result in more and more applications being ported to cloud
- Medical applications use large amounts of data, which means large computation time, which means large costs
- Lack of performance prediction frameworks for medical pipelines
- Solution?
- Hummingbird: Efficient Performance Prediction for Genomics Applications

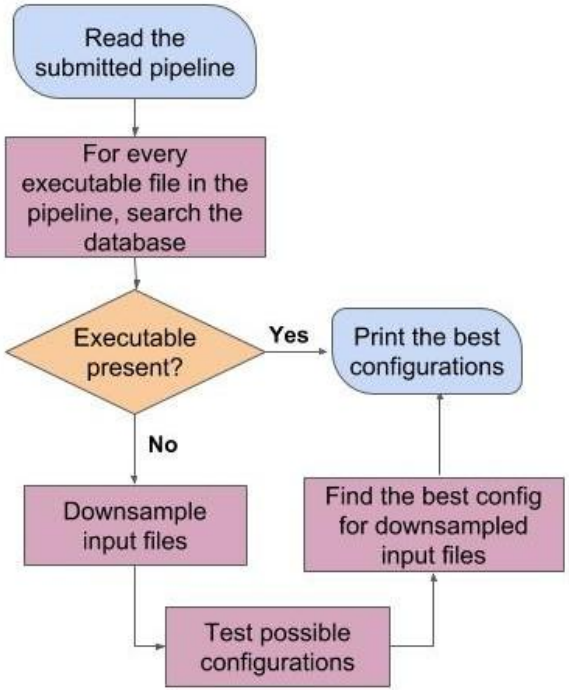


1) Today, the cost of sequencing per genome via HiSeqX is around $1,000 [1]

2) It should be possible to sequence entire genomes at a cost of less than $100 by using NovaSeq technology [2]

Method

- Read user input config file
- Downsample input files
- Execute pipeline for downsampled files on different instance types
- Display cheapest, fastest, and fastest & cheapest configurations



Downsampling

- One of Hummingbird’s main contributions is downsampling the input files
- Downsampling means taking the entire input file and taking the top “n” number of lines and saving that to a new file
- The entire pipeline is now executed on the entire input file
- Once we get results for executing the downsampled input on the entire pipeline we can make assumptions about execution of whole input files on the entire pipeline

Output after Downsampling

- After execution of downsampled input there will be three tables. One each for standard, high-cpu and high-mem
-

Machine	Ideal Speedup	Real Speedup	Execution Time	Normalized Speedup	Cost(based on exec time)
T_2 (base)	1	$1 = T_2/T_2$		1	
T_4	2	$S_4 = T_2/T_4$		S_4	
T_8	4	$S_8 = T_2/T_8$		$S_8/2$	
T_{16}	8	$S_{16} = T_2/T_{16}$		$S_{16}/4$	
T_{32}	16	$S_{32} = T_2/T_{32}$		$S_{32}/8$	

Here T2 indicates a machine with 2 VCPUs

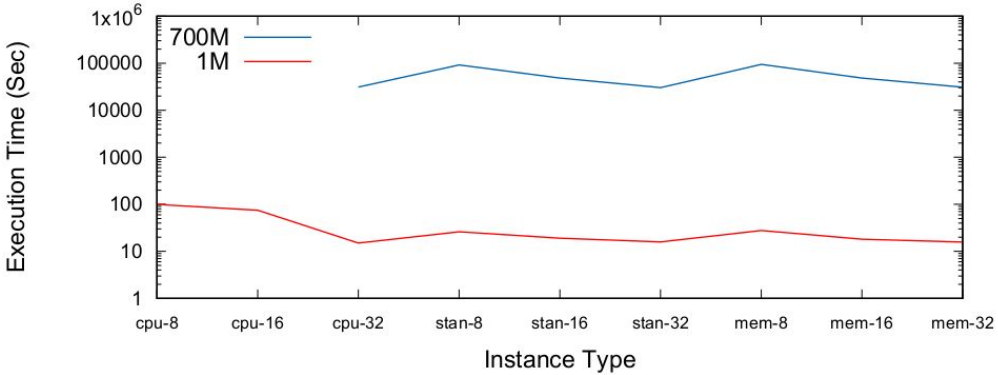
Output after Downsampling (Contd.)

- Once the three tables are populated the user's choice has to be taken into account. If he/she wants a fastest and cheapest configuration, the following steps apply
    - The row with the highest normalized score will be pulled from each table (there are three tables in total)
    - Among the three, the configuration with the least cost will be chosen as the fastest and cheapest configuration
- If the user wants a cheapest or fastest configuration:
    - The row with the cheapest cost or least execution time will be pulled from each table
    - Among those three configurations, the configuration with the least cost or least execution time will be chosen, leading to the cheapest or the fastest configuration

Experimental Setup

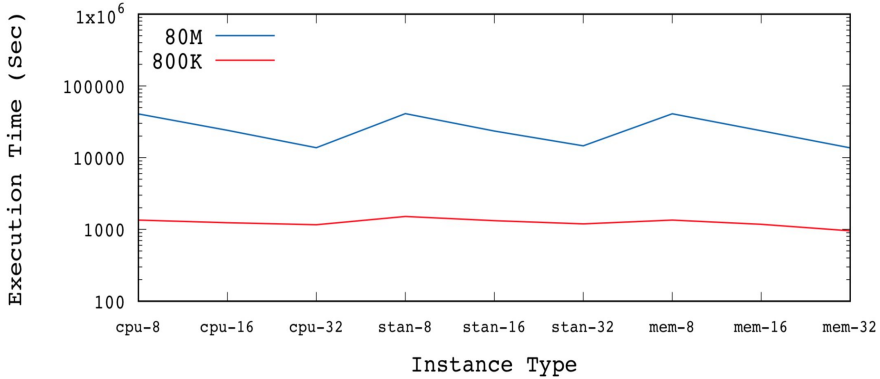
- Hummingbird executes on two pipelines
  - GATK - Illumina Platinum Genomes
    - GATK has multiple different stages
    - Our aim is to provide a recommendation for each stage
  - MuTect2 - Texas Cancer Research Biobank
- Uses *dsub* as job scheduler for Google cloud
- Google cloud has three categories of instances
  - Highcpu
  - Standard
  - Highmem
- Experiments conducted on 8, 16 and 32 VCPU variant of three categories

Result



- On the left is downsampling for BWA
- Trends remain the same in spite of downsampling

- On the right is downsampling for MuTect
- Just like BWA, trends remain the same in spite of downsampling



Result(contd.)

MuTect2

	Hummingbird		Whole Input	
	Config	Exec Time(sec)	Config	Exec Time(sec)
Cheapest	highcpu-8	1348.591	highcpu-8	40754.48
Fastest	highmem-32	957.078	highmem-32	13674.71
Fastest and Cheapest	highcpu-16	1164.396	highcpu-16	24087.63

BQSR Stage 1

	Hummingbird		Whole Input	
	Config	Exec Time(sec)	Config	Exec Time(sec)
Cheapest	standard-8	776.193	standard-8	29003.23
Fastest	highcpu-32	608.741	highmem-32	15944.26
Fastest and Cheapest	standard-16	751.42	standard-16	20113.56

- Hummingbird predicts the correct configuration with high accuracy
- Takes minimal time for prediction
- Almost 2 orders of magnitude less time than the whole input

Result

	Fastest(H)	highmem-32
Exec Time(hour)	66.9	75.5
Cost($)	80	143

- **Fastest**(H) & highmem-32
  - Reduction in cost: **43%**
  - Reduction in execution time: 11%

	Cheapest(H)	standard-16
Exec Time(hour)	92.5	77.9
Cost($)	35	59

- **Cheapest**(H) & standard-16
  - Reduction in cost: **40%**
  - Reduction in execution time: -18%

	Fastest & Cheapest (H)	highmem-32
Exec Time(hour)	77.5	75.5
Cost($)	38	143

- **Fastest & Cheapest**(H) & highmem-32
  - Reduction in cost: **72%**
  - Reduction in execution time: -2%

Work ongoing at NC State

- Profiling of applications
  - Lot of parameters that can be tuned
  - Vary those parameters and record the changes
  - Will help in suggesting to the user an optimum set of parameters for executing jobs

Future Work

- Ability to predict execution time
- Robust error handling
- Cloud agnostic
- Generic framework

OAI Pipeline

Introduction

- Collection of MR Images of knee
- Taken over a time period
- Aim is to understand the progression of Osteoarthritis over time
- Another aim is to interpret the images to develop a cure for OA

Method

- Get patient ID from image names
- Store patient ID and associated image names in a database
- Distribute images of each patient to a node
- Perform segmentation to extract cartilage

Tools

- File system to hold images and intermediate data - HDFS
- Resource Manager for scheduling jobs to different nodes - YARN
- Database to hold patient data - SQLite

Future Work

- Autotuning
  - Various parameters in a medical framework that can be autotuned
- Heterogeneous Scheduling

Hummingbird & OAI Pipeline

- Aim is to integrate Hummingbird and the OAI pipeline
- Once generic version of Hummingbird is developed, it can be used to predict the optimum configuration for executing OAI pipeline in the cloud
- Downsampling can be applied to the OAI pipeline as well
    - Considering one image per patient, and then executing the entire pipeline on one image for each patient, and a limited number of patients could potentially give the optimum configuration