WHITE PAPER

Infrastructure for Deploying GATK Best Practices Pipeline

Abirami Prabhakaran
Design Engineer,
Intel Corporation

Beri Shifaw
Life Science Architect,
Intel Corporation

Mishali Naik
Senior Technology Architect,
Intel Corporation

Paolo Narvaez
Principal Engineer,
Intel Corporation

Geraldine Van der Auwera
Mgr., Outreach & Communications,
Broad Institute of MIT and Harvard

George Powley
Hardware/Software Architect,
Intel Corporation

Serge Osokin
Software Engineer,
Intel Corporation

Ganapati Srinivasa
Sr. Principal Engineer,
Director, Precision Medicine Platform
Intel Corporation

Overview

This document is a guide for the reference platform and benchmarking effort by Intel in collaboration with the Broad Institute. The benchmarking effort involves automating the GATK Best Practices workflow from the Broad Institute as well as providing system-level profiling data for the germline short variant discovery in whole genomes and Exomes. The data gathered in this document shows how best to utilize the latest Intel® Architecture-based platforms.

The GATK Best Practices Workflow is composed of two core pipelines that are to be performed sequentially: 1) pre-processing, which processes the raw reads to analysis-ready mapped reads; and 2) Variant Discovery, which processes the analysis ready reads to variants. The per-sample data pre-processing and variant calling segment of the workflow, from BWA to GATK Haplotype Caller, is implemented as the Single-Sample Calling pipeline and the workflow steps from GenotypeGVCFs to ApplyRecalibration which operates on a cohort of datasets is implemented as the Joint Analysis pipeline.

The flow of both these pipelines are constructed as Perl scripts. In addition to running these tools, system-level data for the individual tools, as well as the overall pipeline, is gathered via Workflow Profiler, an Intel Open Source Project https://01.org/workflow-profiler. Benchmarking results for both the Whole Genome Sequences (WGS) and Exome Sequences (Ex) have been collected and presented in this document for both pipelines.

GATK Best Practices Workflow

Inputs: FASTQ files, b37 bundle include Reference Genome, Hapmap, Omni, dbSNP, 1000G Phase SNPs, and Mills indels.

Outputs: Processed VCF File (as emitted by Single-Sample Calling Haplotype Caller step) and a filtered VCF file (as emitted by Joint Analysis VQSR steps).

Tools and technologies: BWA, Picard Tools, GATK.

Tool versions: Following are the latest versions of the tools we have downloaded for this project, as of August 11, 2015.

TOOL	VERSION	LINK
BWA	0.7.12-r1044	https://github.com/lh3/bwa
PICARD	1.137	https://github.com/broadinstitute/picard
GATK	3.4-46-g8ea4dca	https://github.com/broadgsa/gatk-protected

Table 1. Tool Versions

Table of Contents

Flow of the Single-Sample Calling and Joint Analysis Pipelines

The Single-Sample pipeline assumes we start with the initial FASTQ files and run the stages from BWA-Mem to GATK HaplotypeCaller for the entire sample. Once all samples are processed through the Single-Sample pipeline, the per-sample GVCFs generated by Haplotype Caller are passed to the Joint Analysis pipeline for a cohort study; this ends with a single VCF file of variant calls with genotypes for all samples at all sites, to which filters have been applied to distinguish probable artifacts from true variants.

Figure 1 shows the Broad GATK Best Practices Pipeline (up to HaplotypeCaller) with BWA for mapping to reference and Picard Tools for sorting in the Basecalling + Mapping stages. Figure 2 depicts the implementation of the germline short variant discovery pipeline starting from GenotypeGVCFs and ending with ApplyRecalibration. The processing steps are colored dark blue and inputs/outputs are colored gray.

In the first stage of the pipeline, BWA-Mem performs an alignment on the input FASTQ files and produces a Sequence Alignment/Map (SAM) file, which is then processed through Picard SortSam to produce a sorted BAM (binary version of SAM) file. This sorted BAM file is then passed through Picard Tools MarkDuplicates, which removes any duplicates in the sorted bam file, and creates a merged BAM file that is marked for duplicates. The next few steps: RealignerTargetCreator, IndelRealigner, BaseRecalibrator, PrintReads, and HaplotypeCaller are part of the GenomeAnalysisToolKit (GATK), which is a software package to analyze high-throughput sequencing data. These tools have been configured to meet the GATK Best Practices guidelines. It is important to note the use of the newer HaplotypeCaller as a variant caller in this pipeline as opposed to the older UnifiedGenotyper. HaplotypeCaller has many improvements over the former which is discussed on the GATK web site (https://www.broadinstitute.org/gatk/guide/best-practices.php).

The GenotypeGVCFs tool is then responsible for performing joint genotyping on the per-sample GVCF files (with .g.vcf extension) generated by HaplotypeCaller, and produces a single VCF for the cohort. This cohort VCF can be passed through VariantFiltration for pre-filtering of inbreeding coefficient. Next, variant quality score recalibration (VQSR) is performed by applying the VariantRecalibrator and ApplyRecalibration tools, twice each – once for indel and once for SNPs. VQSR is a filtering technique that uses machine learning algorithms to create a well calibrated score called variant quality score log-odds (VQSLOD), which can be used to filter a callset to a desired level of sensitivity against a given truthset. The final output of the pipeline is one filtered VCF.

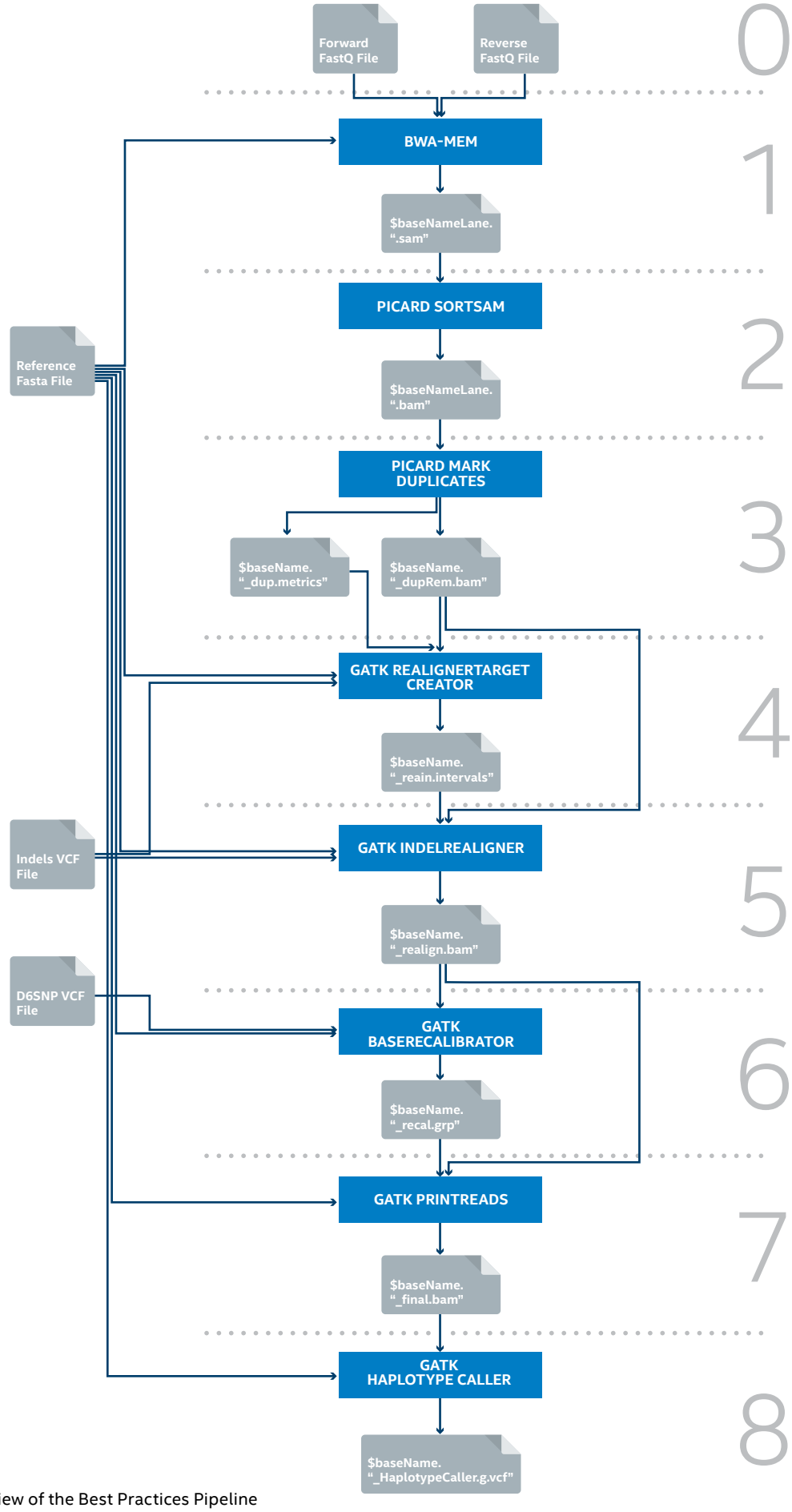


Figure 1. Overview of the Best Practices Pipeline

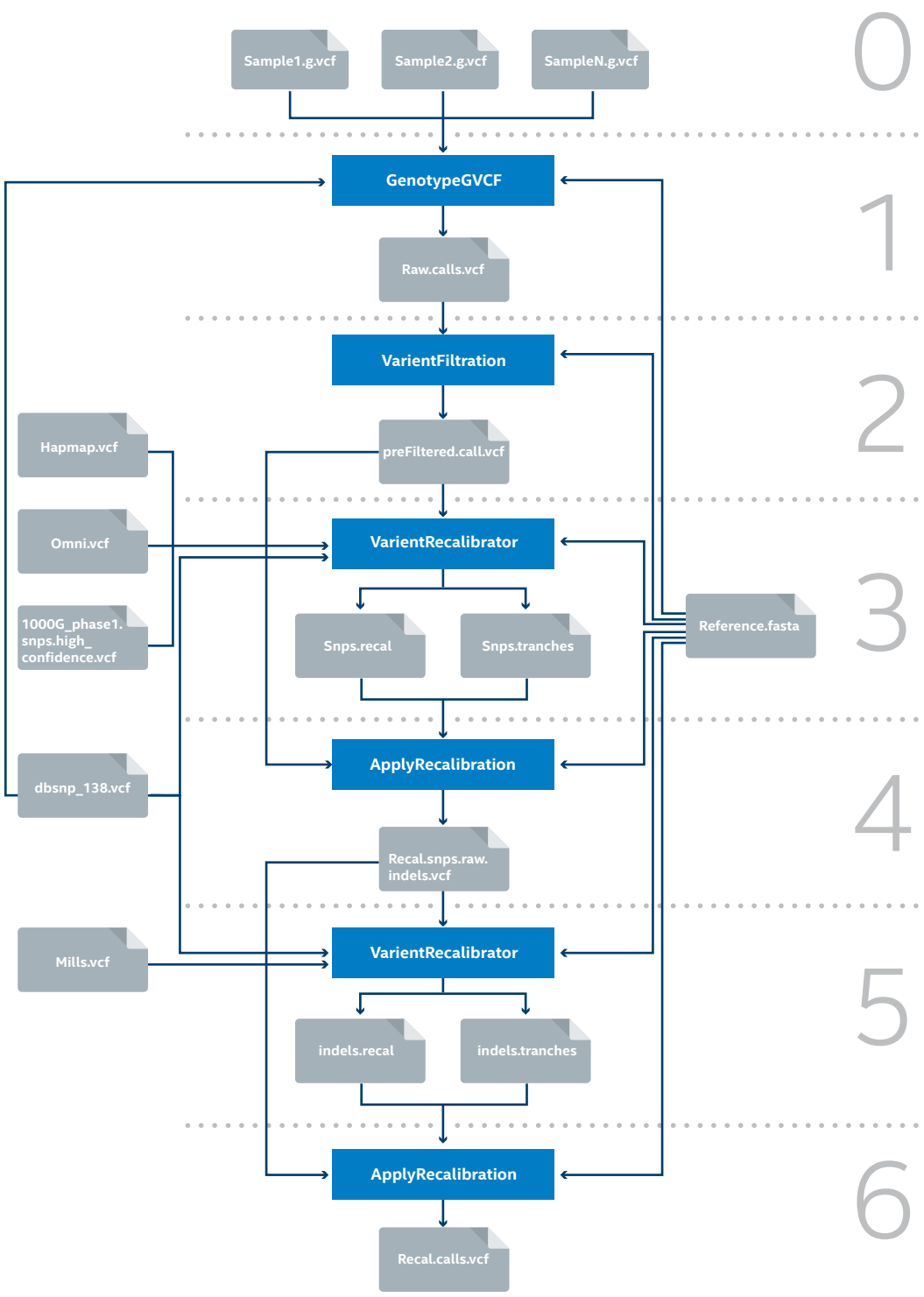


Figure 2. Joint Analysis Pipeline

Arguments for the Pipelines

Details on the arguments for each tool is listed below. For most tools, inputs, and outputs are processed using the "I" and "O" parameters. Where mentioned, "NUMTHREADS" will take in the number of threads provided as input to the Perl script, "GENOMEREF" is the reference fasta file, "DBSNP VCF" is the dbsnp vcf file and "INDELS VCF" is the indels.vcf file, all of which are provided by the user.

BWA-Mem: Map low-divergent sequences against a large reference genome.

-M
-a
-t "NUMTHREADS"
-R "GENOMEREF"
ID:<uniqueid> LB:<libraryname> SM:<samplename> PL:<platformname>

For –R option, give a unique id relating to flowcell and lane numbers, and provide a library name, sample name, and platform name.

SortSam: Sorts the input SAM or BAM and creates an index file.

SORT_ORDER=Coordinate
CREATE_INDEX=TRUE

MarkDuplicates: Examines aligned records in the supplied SAM or BAM file to locate duplicates.

M=$duplicateMetricsFile
CREATE_INDEX=TRUE

RealignerTargetCreator: Emits intervals for the IndelRealigner to target for local realignment.

-nt "NUMTHREADS"
--known "INDELS VCF"
-R "GENOMEREF"
[-L exome_targets.intervals] #if running on exome
[-ip 50] #if running on exome (interval padding)

IndelRealigner: Performs local realignment of reads to correct misalignments that are due to the presence of indels.

-targetIntervals <output from RealignerTargetCreator>
--known "INDELS VCF"
-R "GENOMEREF"

BaseRecalibrator: Identifies systematic errors in base quality scores emitted by the sequencer and computes a recalibration model used to adjust base quality scores accordingly.

--knownSites "DBSNP VCF"
-R "GENOMEREF"
[-L exome_targets.intervals] #if running on exome
[-ip 50] #if running on exome (interval padding)

PrintReads: Produces a recalibrated merged output bam file sorted in coordinate order.

-R "GENOMEREF"
-BQSR <output from BaseRecalibrator>

HaplotypeCaller: Calls germline SNPs and indels via local re-assembly of haplotypes, and emits a genomic VCF containing genotype likelihoods for all possible genotypes at all sites.

-R "GENOMEREF"
-ERC GVCF
--variant_index_type LINEAR
--variant_index_parameter 128000
[-L exome_targets.intervals] #if running on exome
[-ip 50] #if running on exome (interval padding)

GenotypeGVCFs: Merges gVCF's to create a genotyped VCF.

-nt "NUMTHREADS"
-R "GENOMEREF"
-D "DBSNP VCF"
-V <input>

VariantFilteration: Hard filters a VCF based on user defined arguments.

-R "GENOMEREF"
-V <input>
--filterExpression \"InbreedingCoeff < -0.3\"
--filterName \"InbreedingCoeff\"

VariantRecalibrator SNP: Assigns a well-calibrated probability score called VQSLOD to each variant.

-nt "NUMTHREADS"
-R "GENOMEREF"
-recalFile <output>
-tranchesFile <output>
-allPoly
-tranche 100.0 -tranche 99.95 -tranche 99.9 -tranche 99.8 -tranche 99.6 -tranche 99.5 -tranche 99.4 -tranche 99.3 -tranche 99.0 -tranche 98.0 -tranche 97.0 -tranche 90.0
-an QD -an MQRankSum -an ReadPosRankSum -an FS -an MQ -an SOR
-resource:hapmap,known=false,training=true,truth=true,prior=15 <hapmap resource file>
-resource:omni,known=false,training=true,truth=true,prior=12 <omini resource file>
-resource:1000G,known=false,training=true,truth=false,prior=10 <1000G resource file>
-resource:dbsnp138,known=true,training=false,truth=false,prior=7 "DBSNP VCF"
--maxGaussians 6
-mode SNP

ApplyRecalibration SNP: Uses VQSLOD to dynamically filter a VCF.

-nt "NUMTHREDS"
-R "GENOMEREF"
-recalFile <output of VariantRecalSNP>
-tranchesFile <output of VariantRecalSNP>
-ts_filter_level 99.6
-mode SNP

VariantRecalibrator INDEL: Assigns a well-calibrated probability score called VQSLOD to each variant.

-nt "NUMTHREDS"
-R "GENOMEREF"
-recalFile <output>
-tranchesFile <output>
-allPoly
-tranche 100.0 -tranche 99.95 -tranche 99.9 -tranche 99.5 -tranche 99.0 -tranche 97.0 -tranche 96.0 -tranche 95.0 -tranche 94.0 -tranche 93.5 -tranche 93.0 -tranche 92.0 -tranche 91.0 -tranche 90.0
-an FS -an ReadPosRankSum -an MQRankSum -an QD -an SOR
-resource:mills,known=false,training=true,truth=true,prior=12 "INDEL VCF"
-resource:dbsnp138,known=true,training=false,truth=false,prior=2 "DBSNP VCF"
--maxGaussians 6
-mode INDEL

ApplyRecalibration INDEL: Uses VQSLOD to dynamically filter a VCF.

-nt "NUMTHREDS"
-R "GENOMEREF"
-recalFile <output of VariantRecalINDEL>
-tranchesFile <output of VariantRecalINDEL>
-ts_filter_level 95.0
-mode INDEL

Thread and Process Level Parallelism

Thread level and process level parallelism has been enabled where applicable for the above set of tools for the benchmarking. Thread level parallelism can be described as splitting a task into separate independent parts then running these parts using multiple threads as part of the same process. Different threads within the same process can share memory between themselves. While process level parallelism runs multiple similar processes using multiple threads but on different parts of the data, without memory sharing between tasks. Both have the same divide and conquer concept which helps speed up the overall pipeline execution time.

For the Single-Sample Calling pipeline, thread level parallelism has been applied to BWA and GATK RealignerTargetCreator while process level parallelism Scatter-Gather has been applied to all remaining GATK tools—IndelRealigner, BaseRecalibrator, PrintReads, and HaplotypeCaller.

For the Joint Analysis pipeline, VariantRecalibrator is currently unable to conduct process level parallelism and a comparison between both thread and process level parallelism techniques for the rest of the tools showed no significant improvement in time. Thus, all the tools in the Joint Analysis portion uses GATK's integrated –nt argument to apply thread level parallelism. This option is far easier for a common researcher to implement than the process level parallelism which requires scala scripts to be written for each tool.

Scatter-Gather

We have implemented a simple round-robin job scheduler engine "CMPShell" to leverage scatter-gather in GATK using Queue. Scatter-gather is enabled on IndelRealigner, BaseRecalibrator, PrintReads, and HaplotypeCaller. Command-line argument "-jobRunner CMPShell" is passed to the Queue.jar to use CMPShell engine.

Our "CMPShell" scheduling solution works well for this bench-marking exercise, where there is only one pipeline being executed on a single node. It provides us with all the performance and profiling information for a single execution. In more complicated scenarios where there are multiple pipelines being executed at once over multiple nodes, a more sophisticated scheduling solution is required.

Perl Script

The above described flow of tools along with its arguments are packaged in a couple of Perl scripts (**single_sample_calling_*data_collection_gatk_best_practices.pl*** and **single_sample_calling_data_collection_gatk_best_practices_optimized.pl**) for .Single-Sample Calling and a Perl script for the Joint Analysis pipeline (**joint_analysis_*data_collection_gatk_best_practices_optimized.pl***). All these scripts are designed to take the same arguments to keep it consistent for running tests.

For single threaded baseline analysis with no optimizations run the **single_sample_*data_collection_gatk_best_practices.pl*** script with NumThreads as "1."

For both thread level and process level parallelism analysis, use the **single_sample_*data_collection_gatk_best_practices_optimized.pl*** script with NumThreads as proposed in the results section below. The process level parallelism for the GATK components are pre-enabled in this Perl script.

For the joint analysis, run the **joint_analysis_*data_collection_gatk_best_practices_optimized.pl*** script with NumThreads as 1 for baseline and NumThreads as proposed in the results section for optimized run.

All three scripts take the following arguments:

SampleName NumThreads InputDataDirectory TempOutputDirectory profiling

Where:

SampleName: Meaningful name of the sample being tested. (ex: sim1M_pairs).

NumThreads: The number of threads you want the tools to run on the current system. Not all tools take this parameter.

InputDataDirectory: Location of input files to this pipeline (ex: input_datasets).

TempOutputDirectory: Location where you want the output files to reside at. The scripts also produce *_processing.log and *_output.log files in this directory, for logging.

Profiling: ON [1] or OFF [0]. There is no default value and this value must be entered.

If Profiling is ON, you will need to provide the following additional inputs:

- collectstatspath: Path to this script to collect system-level data
- interval: Sampling interval for profiling in seconds [30s default can be used]
- stats: Choose from the following [--sar || --iostat || --collectl]

Examples:

Without profiling:

```
perl single_sample_data_collection_gatk_best_practices.pl
sim1M_pairs 16 input_datasets/ /foo/pipeline_output/ 0
```

With profiling:

```
perl single_sample_data_collection_gatk_best_practices.pl
sim1M_pairs 16 input_datasets/     /foo/pipeline_output/
1 profiling_tools/collect_stats.ksh 30 --sar –iostat --collectl
```

Workflow Profiler

While the above script can be run as a stand-alone with/without profiling, it is recommended to use the workflow profiler open source package provided here to collect profiling statistics when necessary.

Workflow Profiler allows application and system developers to maximize the efficiency of their system resources. This is accomplished through coarse-grained tracking of processor, memory, storage, and network usage across the various stages of a user-defined workflow while still offering a unified view for the overall workflow.

Detailed documentation and User Guide is provided in the above link and the Workflow Profiler Source can be downloaded from here.

A short summary of the usage of the workflow profiler is provided below for your reference:

python3 workflow_profiler.py workflow_script workflow_name sample_name no_of_threads input_directory output_directory [flags]

Positional arguments [Need to be provided in the following order]:

workflow_script location of your workflow script. Example: /foo/data_collection_workflow.pl

workflow_name name of your workflow

sample_name name of the sample

no_of_threads number of threads you want to run on

input_directory directory where the input files are located

output_directory directory where the output data will be stored

Optional arguments:

- pr PROFILING, --profiling PROFILING Do you want to profile the workflow? ON by default
- pp POST_PROCESSING, --post-processing POST_PROCESSING Do you want to run the parser to generate CSVs and plots? ON by default
- int SAMPLING_INTERVAL, --sampling_interval SAMPLING_INTERVAL Sampling interval for profiling in seconds. Default=30
- w SLIDING_WINDOW, --sliding_window SLIDING_WINDOW Sliding window (average) for plots in seconds. Default=100
- p, --plot Plot all data

Statistics: statistics options

-A, --all Parse all statistics

-s, --sar Parse sar information

-i, --iostat Parse iostat information

-c, --collectl Parse collectl information

Examples:

Run a workflow and capture both profiling and post-processing data with default settings (stats collected and plotting enabled)

workflow_profiler.py single_sample_data_collection_gatk_best_practices.pl gatk_best_practices simulated 16 /data/simulated/ /foo/test/-Ap

Run a workflow and capture only sar profiling data with different sampling interval (Post-processing is OFF)

workflow_profiler.py single_sample_data_collection_gatk_best_practices.pl gatk_best_practices simulated 16 /data/simulated/ /foo/test/-pp 0 -int 100 –s

Hardware Specifications

The reference platform we recommend for making use of the Intel optimizations is as follows:

Essentials	Intel® Server Board S2600TPF
CPU Type	Intel® Xeon® processor E5-2699 v3 @ 2.30 GHz
Sockets	2
Core(s) per socket	18
Thread(s) per core	1
Cores(s)	36
RAM	256 GB
Local Drive	Single Seagate Constellation HDD | Model # ST4000NM0033 | 3.6 TB | Max Sustained Transfer Rate 175 MB/s
Hyperthreading	Disabled

Table 2. Reference Hardware

Unless otherwise stated, all runs discussed in the results for the WGS and Ex sequences below were performed 1 pipeline job at a time. All tests were localized to one node, with each node containing 2 CPU sockets with 18 cores each per socket. Hyper threading was disabled thus the Hardware thread per core is 1. The software threads for each run will vary from 1 to 36 threads as discussed in each step of the result. The below key will be specified for those runs where the default configuration was changed.

Key	
# Pipeline	1
# SW Threads/Pipeline	36
# Nodes	1
# Cores/Node	36
# HW Threads/Core	1

Table 3. Key for Results

Single-Sample Calling and Joint Analysis Pipeline applied to Genomes

Datasets

FastQ datasets chosen for the Single-Sample Calling pipeline include the samples mentioned in the Table 4. The pipeline based on Figure 1 was run for the Whole Genome Shotgun (WGS). The datasets used in benchmarking the Joint Analysis pipeline are listed in Table 5. These VCF datasets were obtained by passing the Solexa fastq files through the Single-Sample Calling pipeline. The last tool in the Single-Sample Calling pipeline, HaplotypeCaller, created three files for each WGS sample. The three files are a GVCF file (.g.vcf extension) and two supporting files for the variant file which are an index file and a log file (with extensions .idx and .out, respectively). The reference and resource files used for both pipelines are mentioned in Table 6.

Sequence Type	Sample	File Used	Size (GB)	X Coverage	Read Length
Input WGS dataset	Solexa-272221	Solexa-272221_1.fq	131	30x	2x151
		Solexa-272221_2.fq	131		
	Solexa-272222	Solexa-272222_1.fq	132	30x	2x151
		Solexa-272222_2.fq	132		
	Solexa-272228	Solexa-272228_1.fq	133	30x	2x151
		Solexa-272228_2.fq	133		

Table 4. Dataset Selection for WGS

Sequence Type	Sample	File Used	Size
Input WGS dataset	Solexa-272221	Solexa-272221_1.g.vcf	69G
		Solexa-272221_1.idx	196K
		Solexa-272221_1.out	1.5M
	Solexa-272222	Solexa-272222_1.g.vcf	68G
		Solexa-272222_1.idx	196K
		Solexa-272222_1.out	2.0M
	Solexa-272228	Solexa-272228_1.g.vcf	72G
		Solexa-272228_1.idx	1.5M
		Solexa-272228_1.out	12K

Table 5. WGS Dataset for Joint Analysis Pipeline

File Type	File Used	Size
Reference Genome	human_g1k_v37.fasta	3.0G
dbSNP VCF File	dbsnp_138.b37.vcf	10G
Indels VCF File	1000G_phase1.indels.b37.vcf	0.2G
HapMap VCF File	hapmap_3.3.b37.vcf	0.2G
Omni VCF File	1000G_omni2.5.b37.vcf	0.2G
1000 Genome	1000G_phase1.snps.high_confidence.b37.vcf	6.9G
Mills VCF File	Mills_and_1000G_gold_standard.indels.b37.vcf	.08G
Exome Intervals	Broad.human.exome.b37.interval_list	6.8G

Table 6. Resource Files for Single-Sample and Joint Analysis Pipeline

Profiling Results

Single-Thread vs. Parallelized Run

For the entire WGS sample Solexa-272221, the baseline single-threaded execution times was measured and compared against a partially parallelized pipeline to show the speedup one can achieve by leveraging parallelism at thread- and process-level as described earlier.

The baseline and optimized execution times in Table 7 were captured by running the single_sample_data_collection_gatk_best_practices perl scripts. The NUMTHREADS for the multithreaded run was set to 36 for BWA-Mem and GATK RealignerTargetCreator and scatter-gather enabled for the last four GATK components. The speedup from baseline to optimized is listed in the third column. Overall the execution time from BWA-Mem to HaplotypeCaller stages went from 227 hrs to 36 hrs for this entire sample.

NOTE: Picard SortSam and MarkDuplicates in both cases are running with their default settings.

Tools	Single-Thread Solexa-272221	36 Thread Solexa-272221	Speedup (x)
BWA Mem	92:03:19	3:50:58	23.9
Picard SortSam	7:55:51	6:36:35	1.2
Picard MarkDuplicates	6:34:47	5:45:15	1.1
GATK RealignerTargetCreator	5:58:20	0:18:56	18.9
GATK IndelRealigner	7:24:38	3:52:38	1.9
GATK BaseRecalibrator	19:49:07	1:56:07	10.2
GATK PrintReads	23:54:38	7:28:08	3.2
GATK HaplotypeCaller	63:39:09	6:18:39	10.1
Total Execution Time	**227:19:49**	**36:07:16**	**6.3**

Table 7. Baseline vs. Optimized Execution Time(s) for one WGS sample

Before running the baseline vs. optimized analysis for the Joint Analysis pipeline, the other two WGS were run through the Single-Sample Calling pipeline and the results are available in Table 8. The purpose of this table is to show consistency in the pipeline duration even with datasets from different samples with similar characteristics. All the samples being run show similar times for completing the pipeline. The slight variation in times for each sample may be accounted for by the variation in sample size. The dataset size listed in Table 4 shows Solexa-272221 as having the lowest size and Solexa-272228 having the largest size. This correlates with the runs total time as Solexa-272221 has the shortest duration and Solexa-272228 having the longest duration. Researchers can expect similar times, however variations might also occur due to differences in hardware/software and datasets.

Tools	36T Parallelized Solexa-272221	36T Parallelized Solexa-272222	36T Parallelized Solexa-272228
BWA Mem	3:50:58	3:55:24	3:58:13
Picard SortSam	6:36:35	6:34:48	6:38:06
Picard MarkDuplicates	5:45:15	5:48:14	6:10:11
GATK RealignerTargetCreator	0:18:56	0:16:58	0:45:40
GATK IndelRealigner	3:52:38	3:47:38	4:12:08
GATK BaseRecalibrator	1:56:07	2:03:38	1:57:08
GATK PrintReads	7:28:08	7:30:08	7:13:37
GATK HaplotypeCaller	6:18:39	6:35:09	5:52:08
Total Execution Time	**36:07:16**	**36:31:57**	**36:47:11**

Table 8. Execution Time(s) for three WGS samples using 36 Threads

Table 9 lists the same baseline and optimized comparison for the Joint Analysis pipeline. The table shows GenotypeGVCFs requiring the most time during a single threaded run but drops dramatically as the optimization features are enabled. The optimized features allowed for a 12x speedup in GenotypeGVCFs execution time. Other optimized tools in the pipeline (VariantRecalibrator and ApplyRecalibration) showed a slight speedup but not more than 2.1x. Overall the pipeline showed a ~6.7x speedup in execution time once the optimized features were enabled.

Tools	1 Thread Solexa-Trio	36 Thread Solexa-Trio	Speedup (x)
GenotypeGVCFs	11:23:19	0:56:56	12
VariantFiltration	0:12:14	0:10:06	1.2
VariantRecalibrator_SNP	0:38:45	0:31:44	1.2
ApplyRecalibration_SNP	0:07:07	0:03:22	2.1
VariantRecalibrator_INDEL	0:12:21	0:06:56	1.8
ApplyRecalibration_INDEL	0:03:13	0:03:10	1.0
Total Execution Time	**12:36:59**	**1:52:14**	**6.7**

Table 9. Baseline vs. Optimized Execution Times for the WGS Trio Samples

CPU Utilization

Figure 3 shows the Average CPU utilization chart for the parallelized pipeline for Solexa-272221 generated with Workflow Profiler. The graph displays the percent of all 36 cores being used throughout the course of the pipeline. The different phases or tools of the pipeline are highlighted in different colors, the relative tool is displayed in the legend. The graph shows BWA as being the tool which best utilizes the added CPU resource. The next two phases SortSam and MarkDuplicates do not reach 50% of the available CPU. Two of the tools from GATKs tool set, RealignerTargetCreator and BaseRecalibrator, show high usage in available CPU. However, the other GATK phases show spikes in CPU utilization at the start of each tool (when scatter gather is launching multiple processes) but the majority of the run after the initial spike lowers down to ~5% (when most of the gather work is done by a single process). The CPU utilization for each phase shows some correlation with the speed up time in Table 7, the tools able to best use the available CPU cores have the most speed up time. Other aspects of the resources being used by the pipeline have been measured.

In addition to CPU Utilization, Workflow Profiler provides charts for Average I/O Reads/Writes per sec, I/O Wait times and committed memory map (included in the APPENDIX). The graphs in the appendix section can be further analyzed to determine what aspects of the pipeline other than data processing may hinder speed. I/O Wait is the time it takes for the hard disk to be accessed for reading or writing files. During these times the CPU is waiting for data and is not effectively being used. Understanding all aspects of resources being utilized by the pipeline allows one to profile the individual applications to identify bottlenecks and optimize them further.

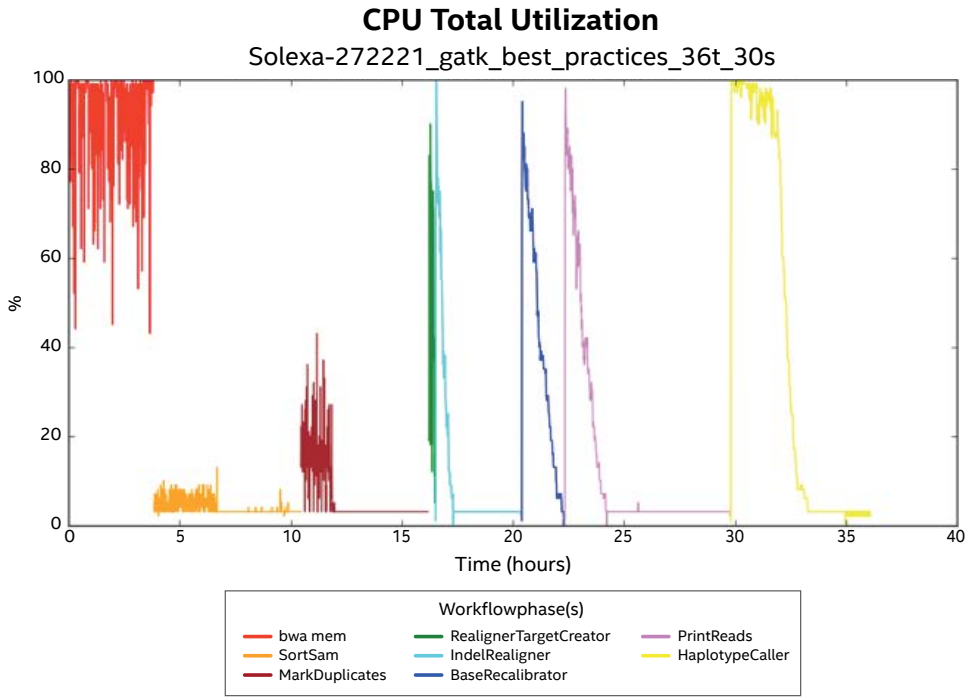


Figure 3. Total CPU Utilization for the Single-Sample Calling Optimized Run

Figure 4 depicts the total CPU utilization over the course of the Joint Analysis pipeline using 36 threads. The first tool in the pipeline, GenotypeGVCFs, uses the most CPU and at times reaches near 100% utilization (efficiently using all available 36 cores). However, this level of CPU utilization is not consistent throughout the run of the tool, on occasion the CPU usage drops down to 30%. The next tool in the pipeline, VariantFiltration, does not leverage GATK's built-in parallelism capabilities, as is evidenced by the low CPU utilization in the figure. The next few steps alternate between VariantRecalibrator and ApplyRecalibration. VariantRecalibrator uses the most cores when first initiated but dramatically reduces its use during the course of the run. ApplyRecalibration shows a similar pattern with high CPU utilization in the beginning followed by a sharp decrease for the short time it is operational. The tools in this pipeline show areas where optimization can be further improved upon.

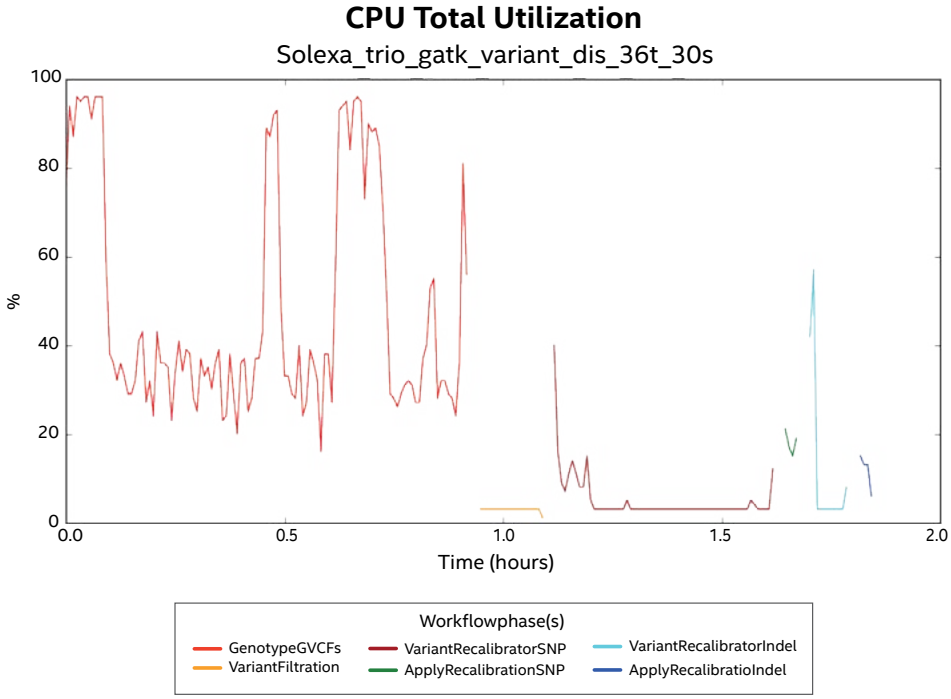
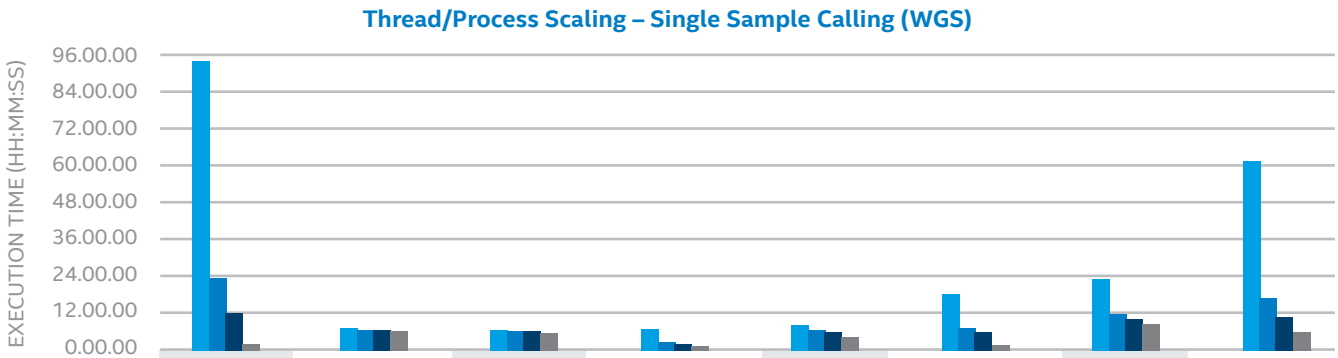


Figure 4. Total CPU Utilization for the Joint Analysis Optimized Run

Thread/Process – Scaling Across Pipeline Components

The chart below shows how the execution time scales with the number of threads/processes across various pipeline components in the pipeline. For this particular dataset, Solexa-272221, all components show a decrease in run time going from 1 to 36 threads. Also, the graph shows the decrease in execution time becomes less significant with higher thread counts. This is due to the nature of parallel processing. Initially having a few threads processing one large datasets decreases the run time. However, dividing the run into smaller parts to be issued on separate threads and finally recombining these pieces into one large file does cost time. This overhead increases as the number of threads being used is increased, therefore at some point more threads begin to hinder the total time of the run. Thus, number of threads/processes cannot be chosen arbitrarily; for instance, thread scaling depends on the size since the amount of work done by each individual thread/process has to outweigh the time spent in combining the work done by each individual thread/process or synchronization. Wisely leveraging the number of samples being analyzed and the available thread count can help researchers complete a set of analysis in a shorter time period.

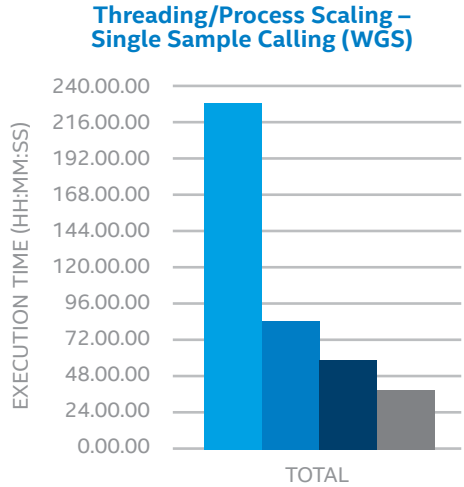


	BWA Mem	Picard SortSam	Picard Markdups	GATK Realigner Target-Creator	GATK Indel-Realigner	GATK Base-Recalibrator	GATK PrintReads	GATK Haplotype-Caller
1T Solexa-272221	92:03:19	7:55:51	6:34:47	5:58:20	7:24:38	19:49:07	23:54:38	63:39:09
4T Solexa-272221	23:55:01	7:07:41	6:38:30	1:09:21	5:17:38	5:55:37	12:44:07	19:37:08
8T Solexa-272221	13:07:04	6:45:20	6:24:17	0:51:56	4:23:08	3:25:37	9:37:38	12:15:03
36T Solexa-272221	3:50:58	6:36:35	5:45:15	0:18:56	3:52:38	1:56:07	7:28:08	6:18:39

Figure 5. Thread/Process Scaling for Solexa-272221

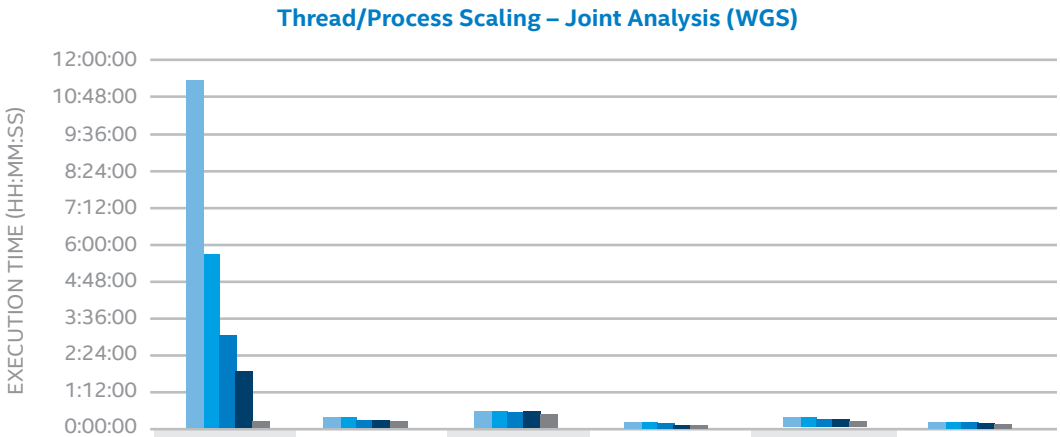
Assuming the only limiting factor is CPU utilization, researchers may use this knowledge to wisely resource their available threads to completing multiple runs with the ideal thread count. For example, instead of running 4 samples consecutively through the pipeline using 36 threads, estimated to take ~144 hours, it would be wiser to run 4 samples in parallel using 8 threads each which would take ~56 hours. This shows using max threads per run may not always be the best solution in terms of reducing execution time, especially when multiple samples require analysis.

Figure 7 depicts thread level tests for the Joint Analysis pipeline, it is clear to see GenotypeGVCFs benefits by the thread count. This is good news considering 90% of the time spent during a single threaded run of the pipeline is in GenotypeGVCFs. The decrease in execution time is initially quite dramatic, as the use of 2 threads instead of 1 cuts the execution time in half. However, additional threads are less valuable as the thread count increase. Figure 7 illustrates the diminishing returns of multithreading in bioinformatics analysis, and supports the idea that maximizing thread count is not an adequate universal solution to efficiently running bioinformatics tools.



1T Solexa-272221	227:19:49
4T Solexa-272221	82:25:03
8T Solexa-272221	55:58:07
36T Solexa-272221	36:07:16

Figure 6. Total Execution time for Single-Sample Calling using WGS



	GenotypeGVCF	VariantFiltration	VariantRecalibrator_SNP	ApplyRecalibration_SNP	VariantRecalibrator_INDEL	ApplyRecalibration_INDEL
1 Thread Solexa-Trio	11:23:19	0:12:14	0:38:45	0:07:07	0:12:21	0:03:13
2 Thread Solexa-Trio	5:34:12	0:11:54	0:33:34	0:04:26	0:08:58	0:03:55
4 Thread Solexa-Trio	2:55:16	0:10:24	0:31:51	0:04:28	0:07:54	0:03:39
8 Thread Solexa-Trio	1:38:16	0:10:31	0:31:18	0:03:28	0:07:12	0:03:39
36 Thread Solexa-Trio	0:56:56	0:10:06	0:31:44	0:03:22	0:06:56	0:03:10

Figure 7. Thread/Process Scaling across the Joint Analysis pipeline

Single-Sample Calling and Joint Analysis Pipeline Applied to Exomes

Datasets

Running the Joint Analysis Pipeline requires a minimum 30 Exomes, in order for the VariantRecalibrator tool to function correctly on this data type. A total of 50 Exomes were used in this benchmarking project. For the Single-Sample Calling pipeline the inputs include three files: two paired-end fastq files and one unpaired fastq for each exome. The unpaired file requires BWA and SortSam to run twice, once for the paired files and secondly for the unpaired file. The two bam files created by the two runs of SortSam is combined in MarkDuplicates and passed through the rest of the pipeline as one input file. Files have been processed through the Single-Sample Calling pipeline, a GVCF file is produced for each exome sample along with supplementary index and log files. Table 10 lists a subset of the samples used for Single-Sample Calling pipeline and Table 11 lists the files used for the Joint Analysis pipeline. The tables do not show the complete 50 Exomes used for benchmarking; the full list is present in the Appendix.

Sequence Type	Sample	File Used	Size	X Coverage	Read Length
Input Exome Dataset	HG02769	HG02769_1.fq	8.6 G	50x	76bp
		HG02769_2.fq	8.6 G		
		HG02769_unpaired.fq	30 M		
	HG02678	HG02678_1.fq	7.5 G	50x	76bp
		HG02678_2.fq	7.5 G		
		HG02678_unpaired.fq	23 M		
	HG02562	HG02562_1.fq	7.7 G	50x	76bp
		HG02562_2.fq	7.7 G		
		HG02562_unpaired.fq	50 M		

Table 10. Subset of Exome Datasets for Single-Sample Calling

Sequence Type	Sample	Size
Input Exome Dataset	HG02769.g.vcf	371M
	HG02678.g.vcf	391M
	HG02562.g.vcf	352M

Table 11. Subset of Exome Datasets for Joint Analysis

Profiling Results

To run the 50 Exomes through Single-Sample Calling pipeline prior to being used in the Joint Analysis pipeline, the samples were run through a grouping experiment to determine the throughput/latency at which the 50 samples can be further processed. Once the optimal grouping was identified, we continued the same experiments (single-thread vs. parallelized, CPU utilization and thread/process scaling) as performed with WGS.

Single vs. Group

For this experiment, we first ran one Exome sample and compared that against a group of four simultaneously running samples on one machine to test what happens when all the machine resources are available to the single run as opposed to the group. Each Exome run within a group was given 8 threads and completed on a single HDD. Also, the individual Exome was run by itself on a machine using 8 threads. The first column in Table 12 shows execution time for HG02769 while running on 8 threads with all machine resources to itself. The second column lists the execution time for HG02769 while sharing the system resources with three other Exomes running simultaneously (HG02561, HG02571, and HG02771). The execution times for grouped runs were slightly higher compared to the single run; BWA in particular contributed the most to the increase by 44 min of runtime (60% of the overall increase in time). With this grouping, running 50 Exomes required a total of 45.5 hours, as per the results in the Appendix. In comparison, this would have taken ~118 hours (2.37 hours HG02769 x 50) if the runs were completed sequentially using max number threads (36) per run.

Tools	Single Exome on HDD	Grouped Exomes on HDD
BWA Mem	0:15:43	0:59:08
Picard SortSam	0:23:37	0:25:59
Picard MarkDuplicates	0:24:43	0:25:24
GATK RealignerTargetCreator	0:04:23	0:04:22
GATK IndelRealigner	0:19:38	0:21:41
GATK BaseRecalibrator	0:14:08	0:24:09
GATK PrintReads	0:40:37	0:49:37
GATK HaplotypeCaller	0:23:07	0:27:38
Total Execution Time	**2:45:56**	**3:57:58**

Key	
#Pipeline	1
# SW Threads/Pipeline	8
# Nodes	1
# Cores/Node	36
# HW Threads/Core	1

Table 12. Single vs Group using HDD

The group and single run were also tested using an SSD, and listed in Table 13. The overall execution time for both tests was shortened, but more importantly the difference in execution time between single and group runs was decreased. This hints at disk I/O being a bottleneck for pipeline runs.

Tools	Single Exome on SSD	Grouped Exomes on SSD
BWA Mem	0:15:12	0:15:30
Picard SortSam	0:22:27	0:25:10
Picard MarkDuplicates	0:23:35	0:24:09
GATK RealignerTargetCreator	0:03:30	0:03:28
GATK IndelRealigner	0:18:07	0:19:38
GATK BaseRecalibrator	0:13:07	0:16:08
GATK PrintReads	0:39:06	0:45:07
GATK HaplotypeCaller	0:17:07	0:18:08
Total Execution Time	**2:32:11**	**2:47:18**

Key	
#Pipeline	1
# SW Threads/Pipeline	8
# Nodes	1
# Cores/Node	36
# HW Threads/Core	1

Table 13. Single vs Group using SSD

This test demonstrates that when running pipelines simultaneously and allocating CPU cores to each run, it would be advantageous to consider using either multiple HDDs (one for each pipeline run) or SSDs. The reason for this is the slowdown in tool execution time due to disk I/O for HDD. The results show that some tools such as BWA, which may need to perform intense I/O, appear to cause slowdowns in the execution time due to multiple pipeline jobs attempting utilize the same disk. By using multiple disks or faster performing disks such as SSDs, there will be a decrease in CPU wait caused by disk I/O.

Single-Thread vs. Parallelized Run

To show the baseline and optimized execution times for the Single-Sample Calling pipeline, we picked the median Exome sample (HG02769) out of the 50 samples. The single thread run for the Single-Sample Calling pipeline was clocked to run ~6.8 hours while the optimized run lasted ~2.4 hours, with the overall speedup calculated to be 2.9x. The three largest speedups produced by the increase in threads affected HaplotypeCaller, BWA, and BaseRecalibrator.

The baseline execution time for the Joint Analysis pipeline is much shorter with a total time of 1.2 hours, and the optimized run for the pipeline shortened the time to ~17 min. The speedup time for the Joint Analysis pipeline is 4x. During the baseline run, most of the tools in the Joint Analysis pipeline were completed in under 10 min, with the exception of GenotypeGVCFs which took ~50 min, but after optimization the execution time of this tool was reduced to ~5 min.

Tools	1 Thread HG02769	36 Thread HG02769	Speedup (x)
BWA Mem	1:31:12	0:12:37	7.2
Picard SortSam	0:23:55	0:24:26	1
Picard MarkDuplicates	0:23:35	0:23:50	1
GATK RealignerTargetCreator	0:12:09	0:10:19	1.2
GATK IndelRealigner	0:22:21	0:18:37	1.2
GATK BaseRecalibrator	0:51:30	0:10:38	4.8
GATK PrintReads	1:31:00	0:32:08	2.8
GATK HaplotypeCaller	1:34:01	0:09:37	9.8
Total Execution Time	**6:49:43**	**2:22:12**	**2.9**

Table 14. Baseline vs Optimized Execution Time(s) for one Exome Sequence

Tools	1 Thread HG02769	36 Thread HG02769	Speedup (x)
GenotypeGVCFs	0:51:36	0:05:24	9.6
VariantFiltration	0:01:46	0:01:46	1
VariantRecalibrator_SNP	0:08:50	0:03:39	2.4
ApplyRecalibration_SNP	0:01:53	0:02:16	0.8
VariantRecalibrator_INDEL	0:05:30	0:02:24	2.3
ApplyRecalibration_INDEL	0:01:52	0:02:13	0.8
Total Execution Time	**1:11:27**	**0:17:42**	**4**

Table 15. Baseline vs Optimized Execution Time(s) for one Exome Sequence

In total, 50 Exomes were passed through the Single-Sample Calling pipeline in groups of four per machine. Table 16 lists one of the 13 groups passed through the pipeline along with its execution time. On average, running Exomes simultaneously through the Single-Sample Calling pipeline using 8 threads takes about 3.7 hours and varies depending on Exome size. BWA and GATK PrintReads take the most time with both taking ~40 min to complete. Table 17 lists the execution time when running 50 Exome samples through the Joint Analysis pipe using 8 threads. The whole process completed in ~20 mins, with GenotypeGVCFs taking the majority of the time with ~8min.

Tools	HG02582	HG02678	HG02562	HG02721
BWA Mem	0:48:11	0:49:09	0:42:55	0:43:52
Picard SortSam	0:21:37	0:21:04	0:24:15	0:21:58
Picard MarkDuplicates	0:20:55	0:20:57	0:22:03	0:21:11
GATK RealignerTargetCreator	0:04:00	0:04:19	0:03:50	0:04:34
GATK IndelRealigner	0:18:48	0:18:31	0:19:39	0:20:39
GATK BaseRecalibrator	0:21:20	0:21:03	0:21:08	0:22:08
GATK PrintReads	0:42:39	0:40:42	0:45:37	0:43:08
GATK HaplotypeCaller	0:24:38	0:25:37	0:22:39	0:26:08
Total Execution Time	**3:22:08**	**3:21:22**	**3:22:06**	**3:23:38**

Key	
#Pipeline	4
# SW Threads/Pipeline	8
# Nodes	1
# Cores/Node	36
# HW Threads/Core	1

Table 16. Grouped Exome on Single-Sample Calling Pipeline using 8 threads

Tools	8 Thread 50 Exomes
GenotypeGVCFs	0:08:38
VariantFiltration	0:01:45
VariantRecalibrator_SNP	0:03:06
ApplyRecalibration_SNP	0:02:13
VariantRecalibrator_INDEL	0:02:16
ApplyRecalibration_INDEL	0:02:14
Total Execution Time	**0:20:12**

Key	
#Pipeline	1
# SW Threads/Pipeline	8
# Nodes	1
# Cores/Node	36
# HW Threads/Core	1

Table 17. Execution Time for 50 Exomes on Joint Analysis Pipeline

CPU Utilization

Similar to the WGS benchmarking, workflow profiler was used to collect system resource usage, one of which is CPU utilization, during the course of the pipeline. Figure 8 and Figure 9 depict the CPU utilization of the Single-Sample Calling and Joint Analysis pipeline. In these line graphs, each color represents a different tool within the pipeline. These graphs display runs with the max available threading (36 threads) provided to the pipeline.

In the Single-Sample Calling pipeline BWA is shown to have an initial 80% spike in CPU usage but this drops down to 40% for the majority of the duration before ending with another spike up to 60%. This differs from its WGS counterpart, where BWA CPU usage is on average above 80%. The next two tools, SortSam and MarkDuplicates, do not have multithreading options but still use 30%-40% of the 36 available cores. This can be accounted for by Java's Garbage Collection method, which automatically uses any available cores. RealignerTargetCreator maintains CPU usage at ~20%, which differs from the WGS test where the CPU usage reached up to 80%. The next few tools use scatter gather for their parallelization method, showing similar results to the previous WGS run. The CPU usage for these tools are initially high then drop down to the single digits, where most of the scatter gather work is done by a single process.

Figure 9 displays the CPU utilization during the Joint Analysis pipeline with the using Exomes. In this graph the intervals for which the stat tools (sar, isostat, collectl) collect system performance has been shortened from every 30 seconds to every 1 second. This helps to record short running tools like VariantFiltration which had a run time of 12 seconds for the exome run. Unlike the WGS run, this figure shows GenotypeGVCFs using a consistent high CPU utilization. The second highest CPU utilization is VariantRecalibrator, reaching a little over 90% usage then dropping steeply. The next tool in the pipeline, ApplyRecalibration, has CPU usage around 60% and again drop steeply after initial startup.

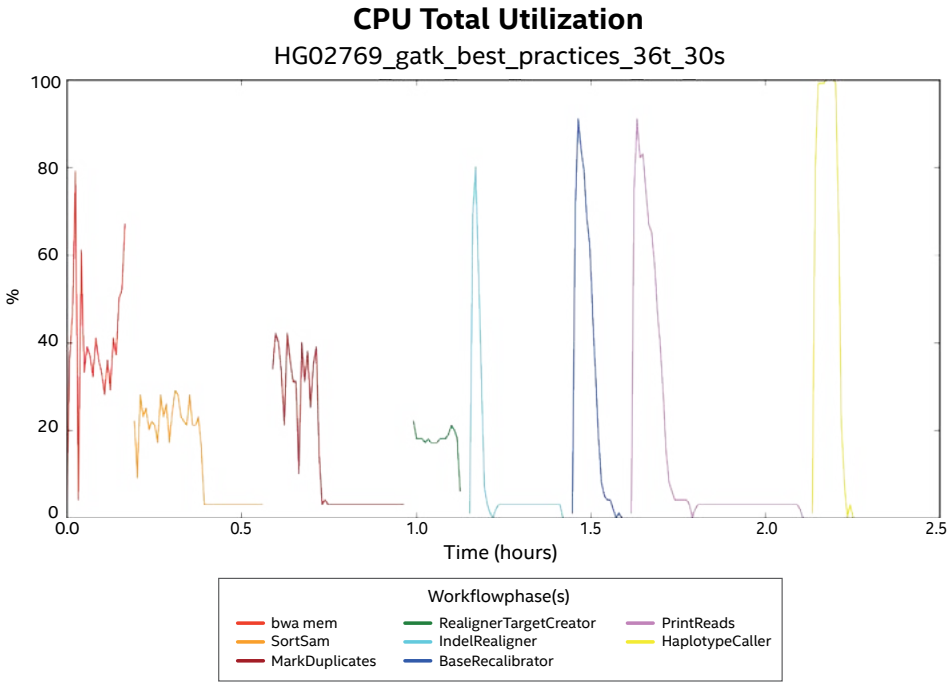


Figure 8. Total CPU Utilization Using HG02769 and 36 threads on Single-Sample Calling pipeline

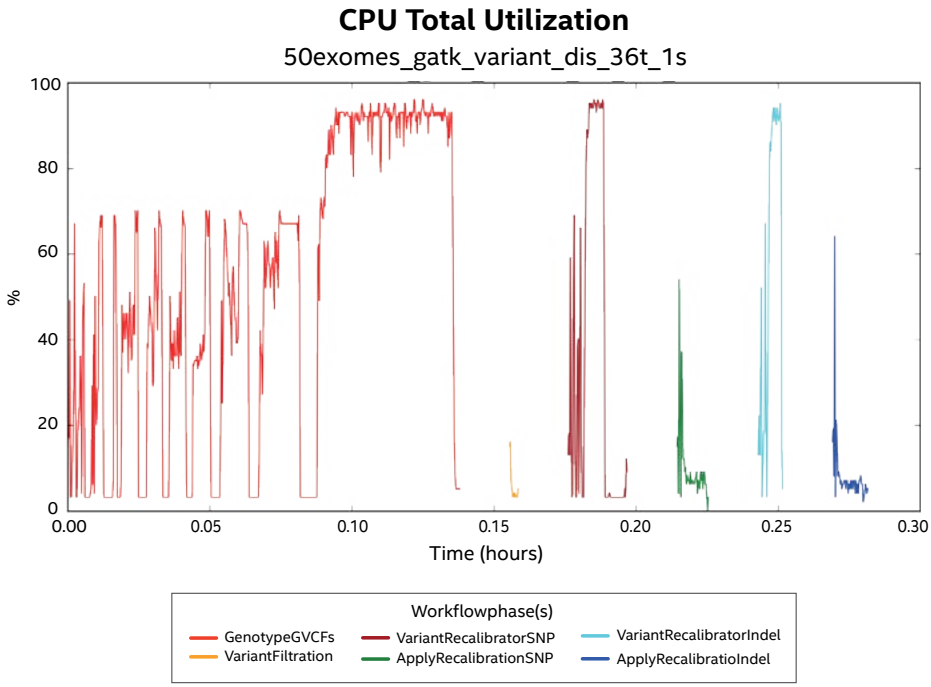
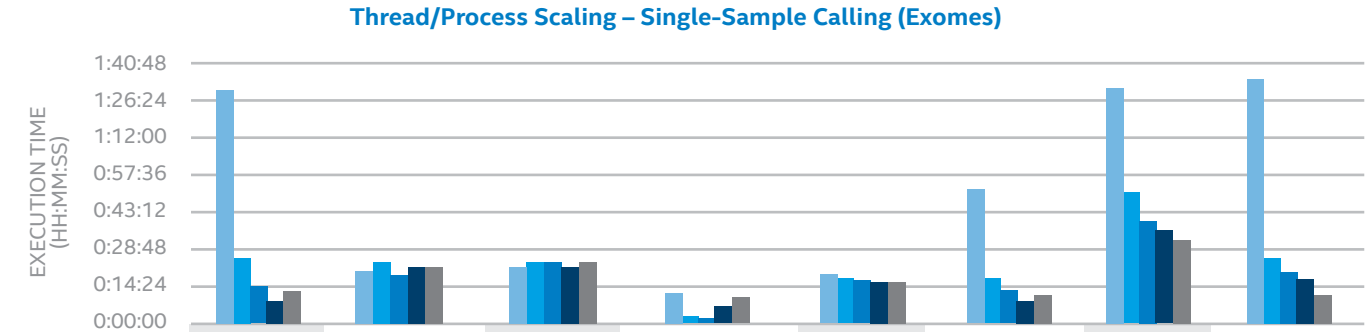


Figure 9. Total CPU Utilization Using 50 Exomes and 36 threads on Joint Analysis pipeline

Thread/Process – Scaling Across Pipeline Components

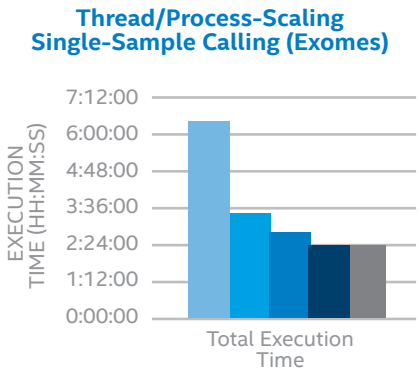
Figure 10 shows a single Exome sample being run through the Single-Sample Calling pipeline as the thread count increases. The preferred Exome sample HG2769 is used in this run. The total execution time for this pipeline using different threads is displayed in Figure 11. These figures illustrate how the value of adding more threads drops dramatically as the number of threads increases. In long running tools such as BWA, HaplotypeCaller, PrintReads, and BaseRecalibrator, execution times are halved by the addition of just 3 threads to the baseline. However, significantly higher thread counts are needed to halve their execution time further.

For GATK RealignerTargetCreator, execution time decreases with the first few threads, but actually increases once the max number of threads is used. RealignerTargetCreator, IndelRealigner, and BaseRecalibrator all showed an increase in execution time when thread count was increased from 16 to 36. This pattern is also present in the Joint Analysis Pipeline as displayed in Figure 12. Both phases of VariantRecalibrator and ApplyRecalibration show an increase in execution time as the maximum number of threads is used. GenotypeGVCFs, showed a continuous drop in execution time with the increase in threads. The overall execution time for both Single-Sample Calling (Figure 11) and Joint Analysis (Figure 13) show a decrease in execution time despite some tools within those pipelines taking longer when given a maximum number of threads.



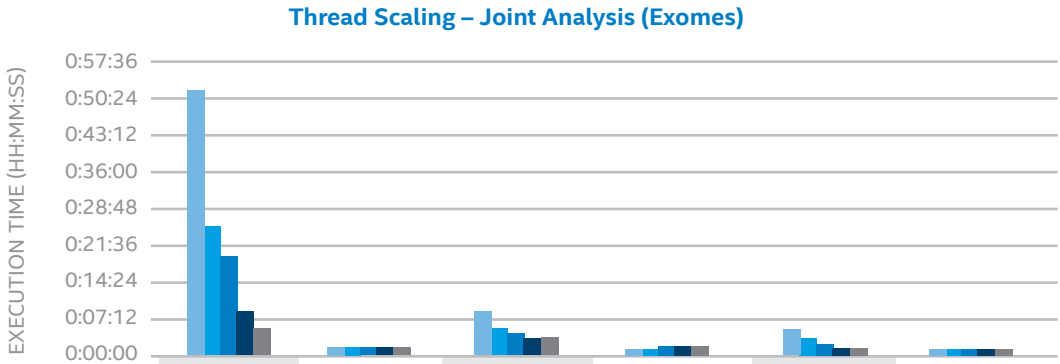
	BWA Mem	Picard SortSam	Picard MarkDuplicates	GATK Realigner TargetCreator	GATK IndelRealigner	GATK BaseRecalibrator	GATK PrintReads	GATK Haplotype Caller
1 Thread	1:31:12	0:23:55	0:23:35	0:12:09	0:22:21	0:51:30	1:31:00	1:34:01
4 Thread	0:25:17	0:24:54	0:23:56	0:04:41	0:20:37	0:18:38	0:48:07	0:27:08
8 Thread	0:15:43	0:23:37	0:24:43	0:04:23	0:19:38	0:14:08	0:40:37	0:23:07
16 Thread	0:09:15	0:23:59	0:22:43	0:06:50	0:18:08	0:10:08	0:33:37	0:17:37
36 Thread	0:12:37	0:24:26	0:23:50	0:10:19	0:18:37	0:10:38	0:32:08	0:09:37

Figure 10. Thread/Process scaling Across Single-Sample Calling pipeline using Exome HG02769



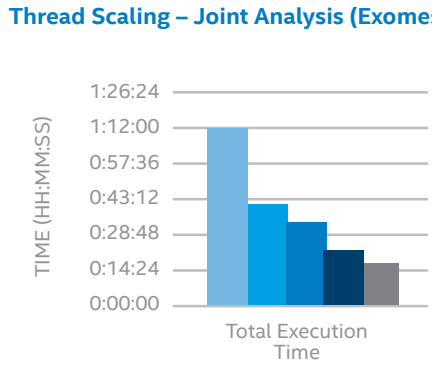
	Total Execution Time
1 Thread	6:49:43
4 Thread	3:13:18
8 Thread	2:45:56
16 Thread	2:22:17
36 Thread	2:22:12

Figure 11. Total Execution Time for Thread/Process scaling on the Single-Sample Calling pipe using Exome HG02769



	GenotypeGVCF	VariantFiltration	VariantRecalibrator_SNP	ApplyRecalibration_SNP	VariantRecalibrator_INDEL	ApplyRecalibration_INDEL
1 Thread	0:51:36	0:01:46	0:08:50	0:01:53	0:05:30	0:01:52
2 Thread	0:26:49	0:01:46	0:05:30	0:02:02	0:03:38	0:02:01
4 Thread	0:19:25	0:01:46	0:04:44	0:01:58	0:02:40	0:02:01
8 Thread	0:08:38	0:01:45	0:03:06	0:02:13	0:02:16	0:02:14
36 Thread	0:05:24	0:01:46	0:03:39	0:02:16	0:02:24	0:02:13

Figure 12. Thread scaling Across Joint Analysis pipe using Exomes



	Total Execution Time
1 Thread	1:11:27
2 Thread	0:41:46
4 Thread	0:32:34
8 Thread	0:20:12
36 Thread	0:17:42

Figure 13. Total Execution Time for the Joint Analysis pipe using Exomes

Conclusion

Advancements in sequencing technology has dramatically lowered the cost of sequencing, leading to the massive production of genomic data requiring analysis. This increase in data has sparked some institutions to help standardize the methods by which the sequencing data is analyzed, one key contributor being the Broad Institute and their Best Practices Guide. This paper aims to further standardization of the sequencing technology by providing a guide for a reference hardware platform and aid in the benchmarking efforts of the GATK Best Practices pipeline. A report on the overall system utilization is demonstrated above using the Intel-based platform recommended in the Hardware Specifications section using the Workflow Profiler in tandem with all scripts to benchmark the workflow.

This paper provides a recommendation of Intel architecture-based hardware for running the GATK Best Practices Pipeline and using this platform to run a set of whole genome sequences through the pipeline while measuring the execution time.

The project was broken into two parts, the first section benchmarked WGS through Single-Sample Calling pipeline and the Joint Analysis pipeline while the second section benchmarked Exomes using the same pipelines. Results from both phases showed execution time for both WGS and Exome sequence decrease as the number of threads increased. The increase in the number of threads for a run has shown to dramatically decrease the process time for BWA-Mem and HaplotypeCaller more so than other parts of the pipeline. One notable difference between the Exome and WGS is the drop in CPU usage for Exomes. It is not certain why Exomes would cause BWA to use less CPU, but it may be related to the small size of the Exome or the contents may be easier to analyze, thus needing less processing resources. Investigation on the CPU utilization and Exome size may help resolve this uncertainty. Using thread and process-level optimizations as we have described, results similar or better than our experiments can be achieved.

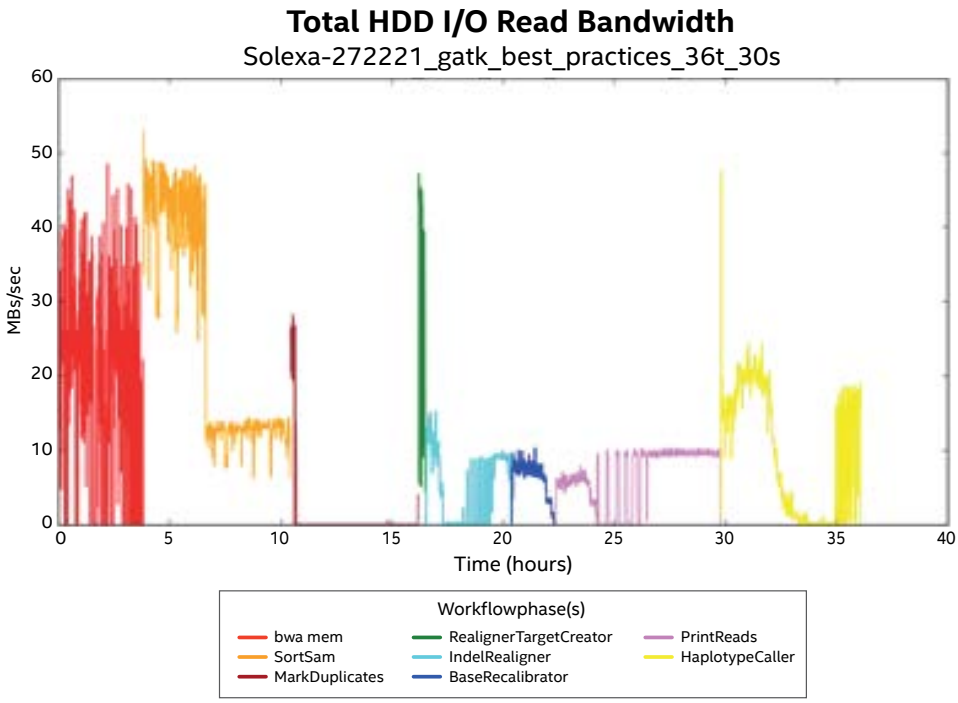
Researchers whose aim is to use this pipeline for multiple datasets may use this paper to scale the number of machines to the amount of datasets which require analysis. For example, an institution whose aim is to analyze 100 WGS a month may need about 5 machines (each with 36 cores) running in parallel to achieve this goal. However, the test above involving grouped vs. single Exomes do show better efficiency when allocating CPU threads and running jobs simultaneously instead of maxing out the threads on one job at a time. Aside from how data is analyzed researchers should also consider the type of hardware performing the analysis.

The hardware recommendations for running GATK Best Practices Pipeline is recorded in this document. It provides a baseline hardware configuration unit to run a basic genomic pipeline. When running multiple pipelines, this unit can be scaled accordingly (e.g., 5 units will run 5 times more genomes/month). More advanced hardware options such as faster storage and I/O may also provide additional benefits to more complex scenarios when many pipelines of different characteristics are being run at the same time over a large cluster. In Phase 2 tests, it was apparent disk I/O can be a bottle neck when running multiple sequence simultaneously. The tests comparing HDD and SDD shows this bottle neck can be avoided by using SSDs. Considering disk I/O can have a significant effect on a tool's performance; it would be interesting to see how well pipeline performance would improve when using Intel® 3D XPoint™ technology. Additionally, servers used in these tests used mid-level grade components, better performance may be achieved by updating the reference hardware. The tests conducted for this projected utilized Intel's version 3 of Haswell CPU processor, a microarchitecture originally released in 2013. This CPU was chosen in consideration for what the average research institution would have available. However, users may see better performance when using Broadwell and Skylake, Intel's latest processors. Boosting the machines hardware is one way of improving performance but optimizing software is an important factor to consider.

Hardware improvements are only part of the solution to faster genome analysis, optimized software are also essential for the progression bioinformatics. The Broad has released a beta version of GATK 4 that utilizes cluster level processing to analyze genomes. Further benchmarking tests on GATK4's ability to effectively utilize cluster sized systems would be beneficial for institutions and businesses who conduct large scale genome analyses, allowing them to better leverage their HP clusters.

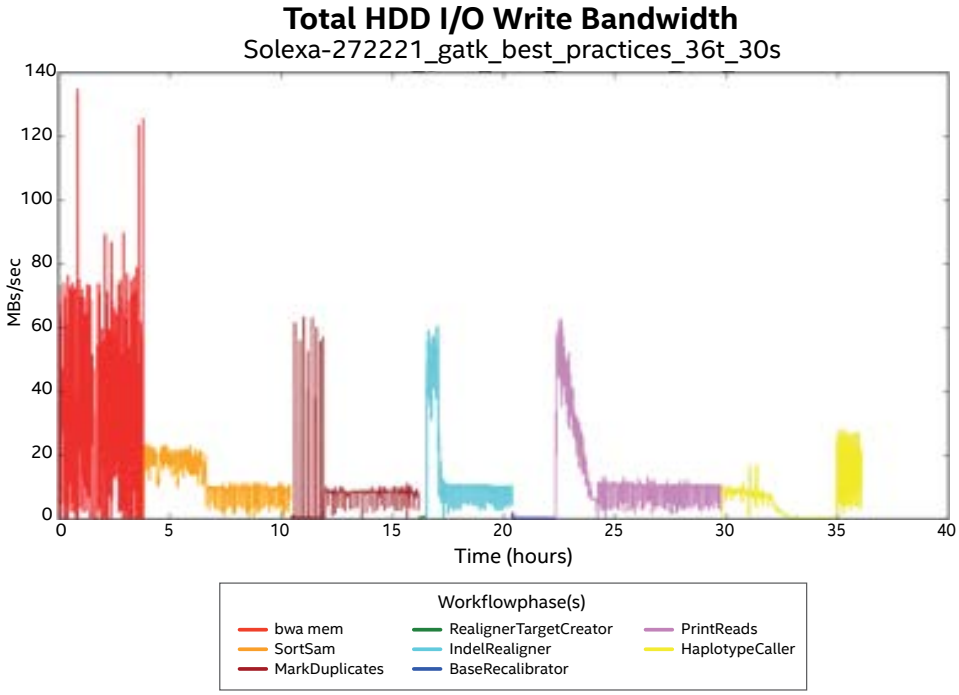
APPENDIX

Appendix A: Average I/O Reads for Solexa-272221.



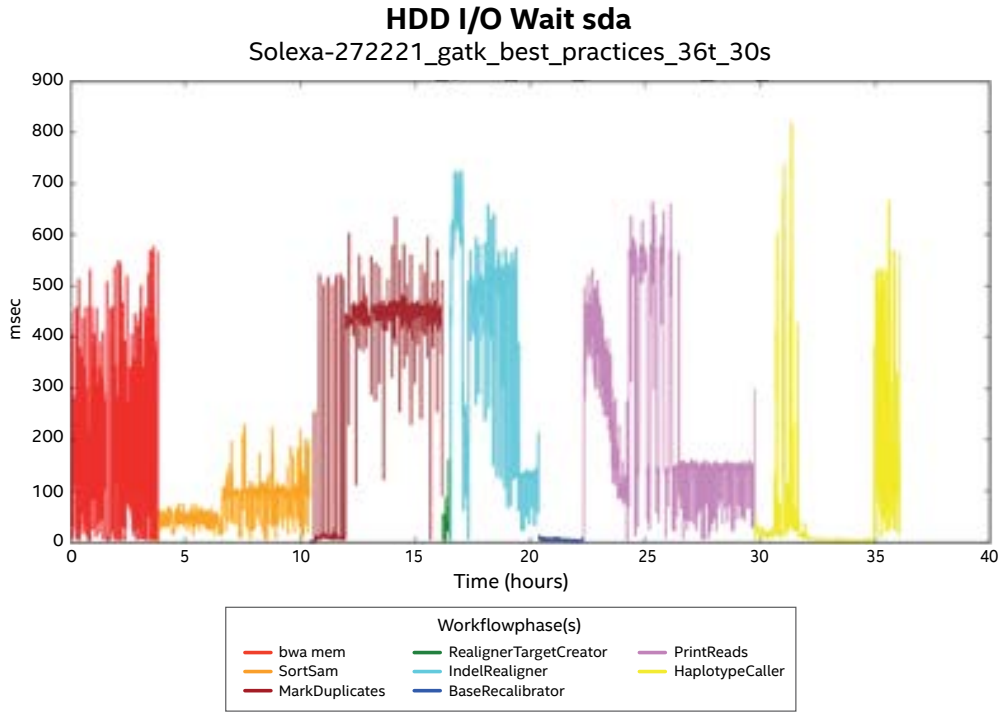
Appendix A. The graph above displays the I/O Read Bandwidth, depicting the speed in megabytes per second at which the system is reading files from storage drives during the execution of the pipeline.

Appendix B: Average I/O Writes for Solexa-272221.



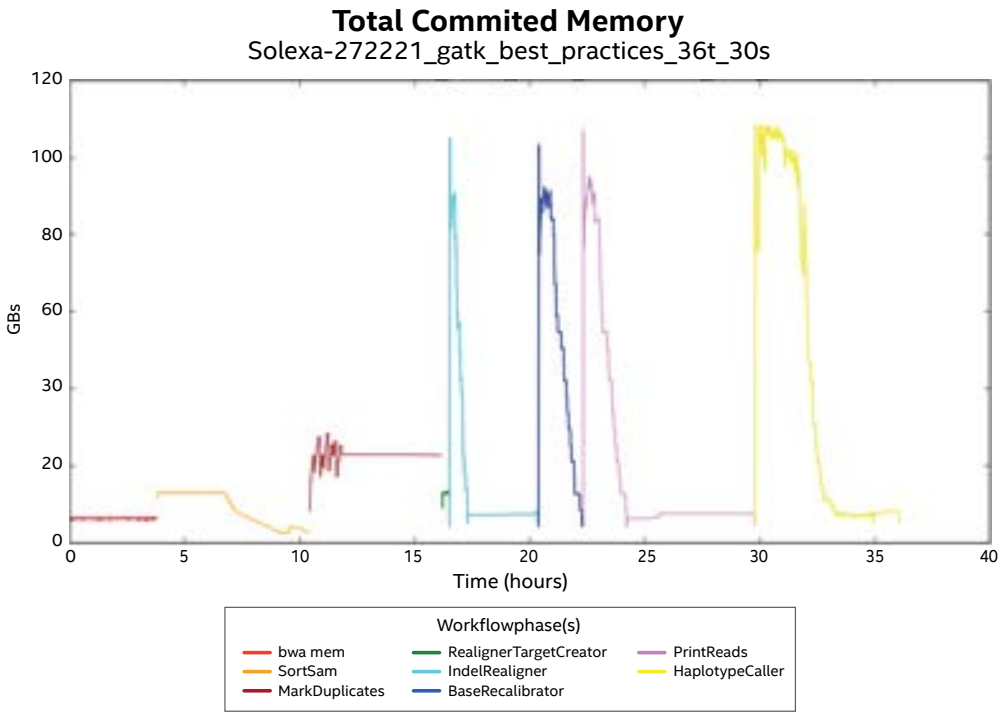
Appendix B. The graph above displays the I/O Write Bandwidth, this represents the speed at which the pipeline is writing files on to the storage drive.

Appendix C: Average I/O Wait Time for Solexa-272221.



Appendix C. The above graph illustrates the wait time in milliseconds while the storage disk is being accessed. A lower I/O Wait is preferred as this indicates the CPU is spending more time manipulating the data instead of waiting on data.

Appendix D: Memory Utilization for Solexa-272221.



Appendix D. The figure above shows the amount committed memory in GB for each phase of the workflow over time. The committed memory is the amount of RAM space a process (or set of processes) reserves for its particular procedure.

Exomes 50 Samples

Sample	File Used	Size	Sample	File Used	Size
HG02582	HG02582_1.fq HG02582_2.fq HG02582_unpaired.fq	7.4G 7.4G 26M	HG03258	HG03258_1.fq HG03258_2.fq HG03258_unpaired.fq	9.7G 9.7G 34M
HG02678	HG02678_1.fq HG02678_2.fq HG02678 _unpaired.fq	7.5G 7.5G 23M	HG02595	HG02595_1.fq HG02595_2.fq HG02595_unpaired.fq	9.8G 9.8G 34M
HG02562	HG02562_1.fq HG02562_2.fq HG02562 _unpaired.fq	7.7G 7.7G 50M	HG02624	HG02624_1.fq HG02624_2.fq HG02624_unpaired.fq	9.8G 9.8G 24M
HG02721	HG02721_1.fq HG02562_2.fq HG02721 _unpaired.fq	7.6G 7.6G 23M	HG02798	HG02798_1.fq HG02798_2.fq HG02798_unpaired.fq	9.9G 9.9G 34M
HG02613	HG02613_1.fq HG02613_2.fq HG02613 _unpaired.fq	7.7G 7.7G 24M	HG02804	HG02804_1.fq HG02804_2.fq HG02804_unpaired.fq	10G 10G 35M
HG02621	HG02621_1.fq HG02621_2.fq HG02621 _unpaired.fq	7.7G 7.7G 24M	HG02810	HG02810_1.fq HG02810_2.fq HG02810_unpaired.fq	10G 10G 34M
HG02635	HG02635_1.fq HG02635_2.fq HG02635 _unpaired.fq	7.7G 7.7G 24M	HG02715	HG02715_1.fq HG02715_2.fq HG02715_unpaired.fq	11G 11G 35M
HG03039	HG03039_1.fq HG03039_2.fq HG03039 _unpaired.fq	7.6G 7.6G 23M	HG02570	HG02570_1.fq HG02570_2.fq HG02570_unpaired.fq	11G 11G 35M
HG02461	HG02461_1.fq HG02461_2.fq HG02461 _unpaired.fq	7.9G 7.9G 24M	HG03040	HG03040_1.fq HG03040_2.fq HG03040_unpaired.fq	11G 11G 36M
HG02620	HG02620_1.fq HG02620_2.fq HG02620 _unpaired.fq	7.9G 7.9G 25M	HG02462	HG02462_1.fq HG02462_2.fq HG02462_unpaired.fq	11G 11G 37M
HG02676	HG02676_1.fq HG02676_2.fq HG02676 _unpaired.fq	7.9G 7.9G 24M	HG03028	HG03028_1.fq HG03028_2.fq HG03028_unpaired.fq	11G 11G 36M
HG02716	HG02716_1.fq HG02716_2.fq HG02716 _unpaired.fq	7.8G 7.8G 27M	HG02561	HG02561_1.fq HG02561_2.fq HG02561_unpaired.fq	8.7G 8.7G 30M
HG02772	HG02772_1.fq HG02772_2.fq HG02772 _unpaired.fq	8.0G 8.0G 28M	HG02571	HG02571_1.fq HG02571_2.fq HG02571_unpaired.fq	8.6G 8.6G 30M
HG02799	HG02799_1.fq HG02799_2.fq HG02799 _unpaired.fq	7.6G 7.6G 23M	HG02771	HG02771_1.fq HG02771_2.fq HG02771_unpaired.fq	8.5G 8.5G 29M
HG02679	HG02679_1.fq HG02679_2.fq HG02679 _unpaired.fq	7.9G 7.9G 24M	HG02756	HG02756_1.fq HG02756_2.fq HG02756_unpaired.fq	8.7G 8.7G 30M
HG02464	HG02464_1.fq HG02464_2.fq HG02464 _unpaired.fq	8.0G 8.0G 30M	HG02757	HG02757_1.fq HG02757_2.fq HG02757_unpaired.fq	8.8G 8.8G 31M
HG03049	HG03049_1.fq HG03049_2.fq HG03049 _unpaired.fq	8.2G 8.2G 25M	HG02629	HG02629_1.fq HG02629_2.fq HG02629_unpaired.fq	8.7G 8.7G 27M
HG02645	HG02645_1.fq HG02645_2.fq HG02645 _unpaired.fq	7.9G 7.9G 24M	HG02634	HG02634_1.fq HG02634_2.fq HG02634_unpaired.fq	8.8G 8.8G 30M
HG02646	HG02646_1.fq HG02646_2.fq HG02646 _unpaired.fq	7.7G 7.7G 44M	HG02623	HG02623_1.fq HG02623_2.fq HG02623_unpaired.fq	8.7G 8.7G 30M

Exomes 50 Samples, continued

Sample	File Used	Size	Sample	File Used	Size
HG02583	HG02583_1.fq HG02583_2.fq HG02583_unpaired.fq	8.4G 8.4G 55M	HG02610	HG02610_1.fq HG02610_2.fq HG02610_unpaired.fq	9.1G 9.1G 32M
HG02585	HG02585_1.fq HG02585_2.fq HG02585 _unpaired.fq	8.3G 8.3G 29M	HG02807	HG02807_1.fq HG02807_2.fq HG02807_unpaired.fq	9.1G 9.1G 32M
HG02768	HG02768_1.fq HG02768_2.fq HG02768_unpaired.fq	8.2G 8.2G 28M	HG02808	HG02808_1.fq HG02808_2.fq HG02808_unpaired.fq	9.3G 9.3G 32M
HG02811	HG02811_1.fq HG02811_2.fq HG02811_unpaired.fq	8.3G 8.3G 29M	HG03046	HG03046_1.fq HG03046_2.fq HG03046_unpaired.fq	9.3G 9.3G 32M
HG02642	HG02642_1.fq HG02642_2.fq HG02642_unpaired.fq	8.4G 8.4G 26M	HG03027	HG03027_1.fq HG03027_2.fq HG03027_unpaired.fq	9.7G 9.7G 34M
HG02588	HG02588_1.fq HG02588_2.fq HG02588_unpaired.fq	8.5G 8.5G 29M	HG02769	HG02769_1.fq HG02769_2.fq HG02769_unpaired.fq	8.6G 8.6G 30M

Table 18. Exome sequence dataset passed through the Single-Sample Calling Pipeline. Note: 50x Coverage and 76bp Read Length

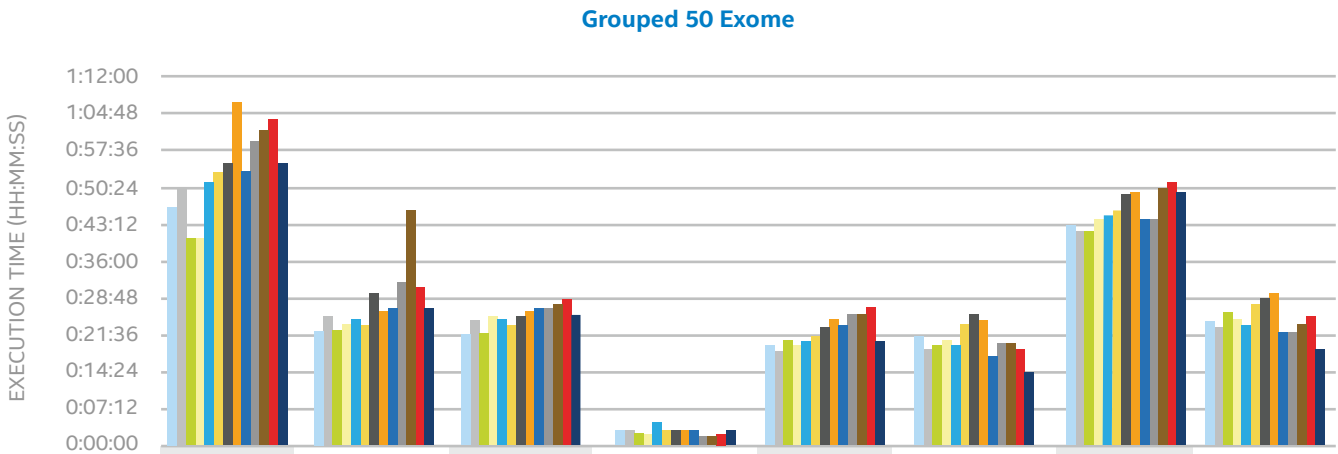
Sample	Size	Sample	Size
HG02461_HaplotypeCaller.g.vcf	364M	HG02678_HaplotypeCaller.g.vcf	391M
HG02462_HaplotypeCaller.g.vcf	296M	HG02679_HaplotypeCaller.g.vcf	388M
HG02464_HaplotypeCaller.g.vcf	371M	HG02715_HaplotypeCaller.g.vcf	319M
HG02561_HaplotypeCaller.g.vcf	381M	HG02716_HaplotypeCaller.g.vcf	439M
HG02562_HaplotypeCaller.g.vcf	352M	HG02721_HaplotypeCaller.g.vcf	389M
HG02570_HaplotypeCaller.g.vcf	316M	HG02756_HaplotypeCaller.g.vcf	396M
HG02571_HaplotypeCaller.g.vcf	410M	HG02757_HaplotypeCaller.g.vcf	374M
HG02582_HaplotypeCaller.g.vcf	424M	HG02768_HaplotypeCaller.g.vcf	399M
HG02583_HaplotypeCaller.g.vcf	329M	HG02769_HaplotypeCaller.g.vcf	371M
HG02585_HaplotypeCaller.g.vcf	406M	HG02771_HaplotypeCaller.g.vcf	412M
HG02588_HaplotypeCaller.g.vcf	387M	HG02772_HaplotypeCaller.g.vcf	425M
HG02595_HaplotypeCaller.g.vcf	346M	HG02798_HaplotypeCaller.g.vcf	336M
HG02610_HaplotypeCaller.g.vcf	332M	HG02799_HaplotypeCaller.g.vcf	438M
HG02613_HaplotypeCaller.g.vcf	384M	HG02804_HaplotypeCaller.g.vcf	314M
HG02620_HaplotypeCaller.g.vcf	369M	HG02807_HaplotypeCaller.g.vcf	368M
HG02621_HaplotypeCaller.g.vcf	378M	HG02808_HaplotypeCaller.g.vcf	352M
HG02623_HaplotypeCaller.g.vcf	404M	HG02810_HaplotypeCaller.g.vcf	334M
HG02624_HaplotypeCaller.g.vcf	325M	HG02811_HaplotypeCaller.g.vcf	402M
HG02629_HaplotypeCaller.g.vcf	334M	HG03027_HaplotypeCaller.g.vcf	330M
HG02634_HaplotypeCaller.g.vcf	381M	HG03028_HaplotypeCaller.g.vcf	354M
HG02635_HaplotypeCaller.g.vcf	370M	HG03039_HaplotypeCaller.g.vcf	407M
HG02642_HaplotypeCaller.g.vcf	340M	HG03040_HaplotypeCaller.g.vcf	306M
HG02645_HaplotypeCaller.g.vcf	410M	HG03046_HaplotypeCaller.g.vcf	349M
HG02646_HaplotypeCaller.g.vcf	458M	HG03049_HaplotypeCaller.g.vcf	369M
HG02676_HaplotypeCaller.g.vcf	370M	HG03258_HaplotypeCaller.g.vcf	336M

Table 19. Exome sequence dataset passed through the Joint Analysis Pipeline

	Average	Max	Min
BWA Mem	0:53:36	1:06:16	0:39:46
Picard SortSam	0:27:18	0:46:15	0:21:04
Picard MarkDuplicates	0:25:05	0:29:52	0:20:55
GATK RealignerTargetCreator	0:04:23	0:06:44	0:03:50
GATK IndelRealigner	0:22:10	0:25:51	0:18:22
GATK BaseRecalibrator	0:20:27	0:24:39	0:13:40
GATK PrintReads	0:46:25	0:53:37	0:31:37
GATK HaplotypeCaller	0:24:57	0:34:07	0:18:38
Total Execution Time	**3:44:21**	**4:21:04**	**3:18:26**

Table 20. Grouped 50 Exome Summary

Exome Single-Sampling Calling Pipeline Grouped Results



	BWA Mem	Picard SortSam	Picard MarkDuplicates	GATK RealignerTargetCreator	GATK IndelRealigner	GATK BaseRecalibrator	GATK PrintReads	GATK Haplotype Caller
Group 1	0:46:02	0:22:14	0:21:17	0:04:11	0:19:24	0:21:25	0:43:02	0:24:46
Group 2	0:50:07	0:24:31	0:23:48	0:04:23	0:19:16	0:18:40	0:42:24	0:23:56
Group 3	0:41:37	0:22:22	0:21:37	0:04:14	0:21:15	0:19:20	0:42:39	0:27:08
Group 4	0:41:30	0:23:17	0:25:17	0:04:11	0:20:26	0:20:27	0:44:49	0:24:38
Group 5	0:51:50	0:23:58	0:24:19	0:05:14	0:20:41	0:19:31	0:45:29	0:23:54
Group 6	0:53:26	0:23:36	0:23:52	0:04:10	0:21:52	0:23:18	0:46:39	0:27:46
Group 7	0:54:39	0:29:16	0:25:11	0:04:18	0:22:48	0:24:31	0:49:30	0:28:46
Group 8	1:06:14	0:25:43	0:25:47	0:04:17	0:23:37	0:23:54	0:49:39	0:29:48
Group 9	0:52:58	0:26:41	0:26:39	0:04:33	0:22:20	0:17:07	0:45:07	0:21:39
Group 10	0:59:09	0:30:52	0:26:47	0:04:13	0:24:46	0:20:18	0:45:00	0:21:40
Group 11	1:01:25	0:45:39	0:27:42	0:04:10	0:24:50	0:20:01	0:50:07	0:23:08
Group 12	1:03:34	0:30:05	0:28:58	0:04:31	0:25:26	0:19:51	0:51:16	0:25:16
Group 13	0:55:10	0:26:16	0:24:27	0:04:45	0:20:53	0:14:33	0:49:09	0:18:53

Figure 14. Single-Sample Calling pipeline using Exome in groups

Printed in USA 1116/DW/HBD/PDF ♻ Please Recycle 333779-002US