A guide to GATK4 best practice pipeline performance and optimization on the IBM OpenPOWER system

Contents

Summary

The evolutional next-generation sequencing (NGS) technology prompts researchers and scientists to seek highly optimized computing solutions to handle an immense volume of NGS sequence data. These solutions include hardware and software tools to analyze and manage data. GATK4 is an open source toolkit frequently used by most genomic research and clinical analyses. The high-performance data and analytics (HPDA) solution, based on IBM® OpenPOWER and IBM Spectrum® computing, dramatically accelerates the analysis workloads. Our benchmark results of 50x the whole genome sequence (WGS) germline pipeline show nearly a 10x speed up through optimization and the analysis from sequence alignment to variant filtering can be completed in nine hours on a 40-core IBM POWER9 system.

Introduction

Genome Analysis Toolkit (GATK),[1] developed by Broad Institute, is an open source genomics analysis package that contains all variant tools for germline and cancer genomic analysis. GATK4 best practice pipelines, published by Broad Institute,[2] are widely adopted by the genomics community. The latest versions of GATK, GATK4, contains Spark and traditional implementations, that is the Walker mode, which improve runtime performance dramatically from previous versions. Since the Spark tools are still in beta testing and perform inconsistently, we mainly focus on the tuning and optimization of GATK in the Walker mode. Without using any tuning and optimization, most GATK4 tools execute with a single thread. And it may take more than 85 hours to complete a pipeline analysis from alignment to produce gene variants calling for a 50x coverage WGS data set. See Figure 2. Fortunately, solutions can be created by splitting the workload into sequence intervals for speeding up runtime concurrently with GATK4 tools, as well as with alternative multithreaded or specific IBM Power Systems tools.[4, 5]

The IBM HPDA solution architecture for healthcare and life sciences is designed to provide solutions for genomics analysis, which requires high-performance data manipulation and high-throughput computing to handle data-centric workloads. See Figure 1. The fundamental components of HPDA include POWER9 processors, an OpenPOWER offering, IBM Spectrum LSF for resource and workload management, IBM Spectrum Scale and IBM Elastic Storage Server (ESS) for data storage and management.

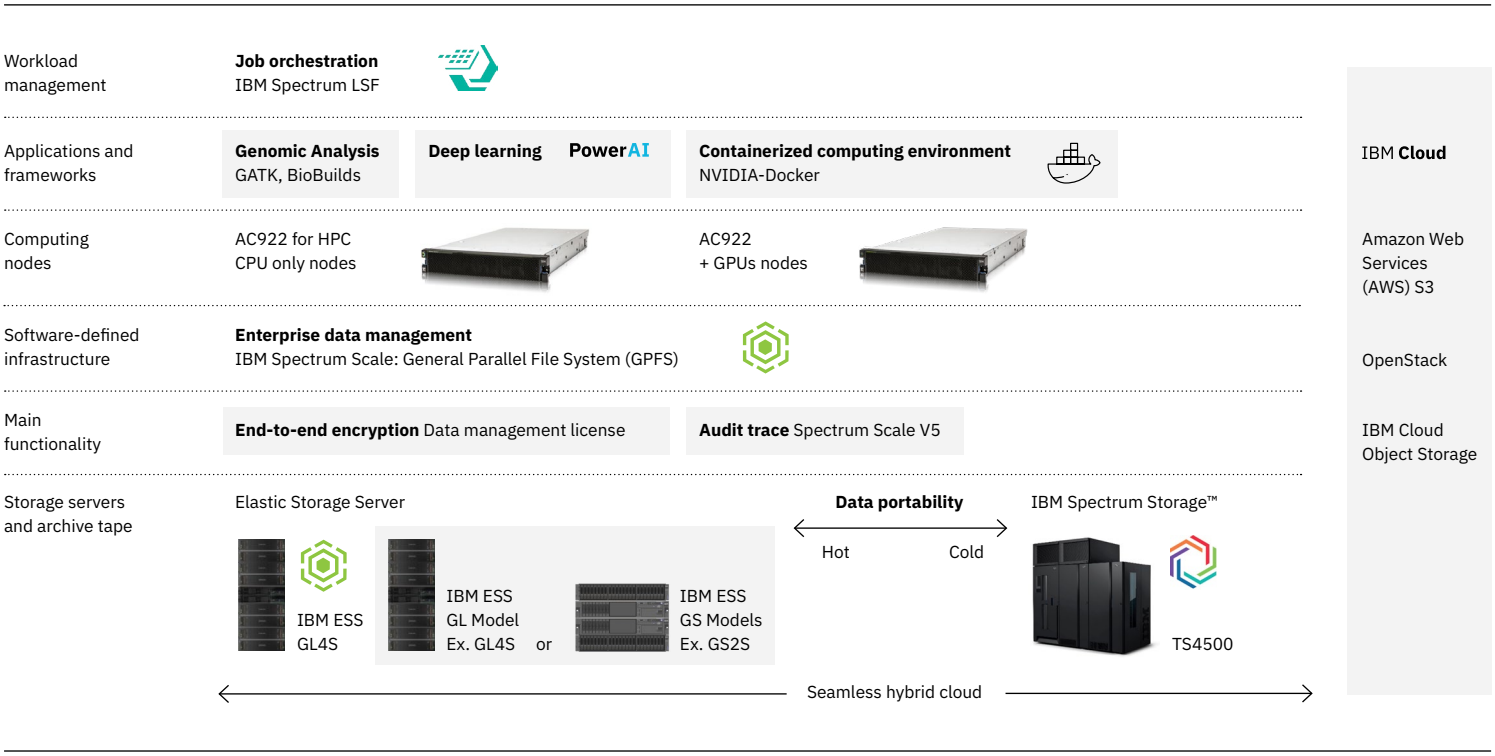


Figure 1: IBM high-performance data analytics solution architecture for healthcare and life sciences, *GATK edition*

The HPDA solution leverages the power of high-performance computing and data storage for next-generation sequencing data analysis by providing:

- *High performance and scalability*: POWER9 systems built for data-centric, memory intensive workloads, including a state-of-the-art input/output (I/O) subsystem. The AC922 and LC922 models, running with the little endian Linux operating system, can be configured with four simultaneous multithreading (SMT)/core, graphics processing units (GPUs) and IBM General Parallel File System (GPFS) to accelerate and scale up application performance.
- *Interoperability and manageability*: IBM Spectrum LSF, IBM Spectrum Discovery and IBM Spectrum Scale help data management, discovery, provenance and data sharing locally or in cloud environments.
- *Usability and complexity avoidance*: With IBM Cloud Private and the IBM Spectrum Storage™ suite, it's designed to provide an easy-to-use orchestration environment and application portability for various users.
- *Improved security and governance*: IBM Power Systems and IBM Spectrum Computing handle the complexities of data security and policies.

This paper uses a best practice germline pipeline as a case study to optimize the GATK4 workload on POWER9 processors. The configurations of the POWER9 system are listed in Table 1. The whole genome sequencing data set NA12878 with 50x coverage was downloaded from the Illumina authorized website.[3-1, 3-2] The reference genome and known variant call format (VCF) reference files are obtained from the Broad Institute resource bundle.[6]

IBM POWER9 system configurations	
Processor	3.6 GHz PowerNV 8335-GTC (P9 AC922)
Socket	2 sockets, 20 cores/socket, 4 SMTs/core
Memory	512 GB DDR4 (RDIMM DDR4 2666 MHz)
Storage	GPFS (ESS GL4 with IBM Spectrum Scale 5.0.0.2)
Operating system	RHEL 7.6 (4.14.0-115.6.1.el7a.ppc64le)
Compilers	Advance toolchains 12.0 (GCC 8.2.1)
GATK version	4.1.0.0 built on POWER9 processors with Native PairHMM
BWA and SAMtools	bwa- 0.7.17-r1188 and SAMtools-1.9 (htslib-1.9) with JEMALLOC
Sam2bam[4]	IBM Power Systems specific tool used for marking duplicates

Table 1: Benchmark system

GATK4 best practice pipelines for the POWER9 system

The GATK4 best practice pipelines are widely used by many research organizations and clinics for variant discovery in normal and cancer genomes. They provide step-by-step recommendations for performing variant discovery analysis in high-throughput sequencing (HTS) data. Currently, there are pipelines available for discoveries of germline or somatic short variants, short variants in RNAseq data, as well as identification of copy number variants (CNVs) in germline and somatic samples. In the following case study, with optimized GATK4 on POWER9 processors, the 50x coverage NA12878 whole WGS genome sequence data was benchmarked for performance tuning and optimization practice. See the installation guide in section C and Figure 2. The WGS was downloaded from the EBI FTP server,[3-1, 3-2] which contains 1,574,530,218 reads in paired-end with 101 bp in length per read. The reference genome Grch38/Hg38, including 1,000 genomes' known VCFs was downloaded from the Broad Institute FTP server.[6] The scripts used for step-by-step processes are described here.

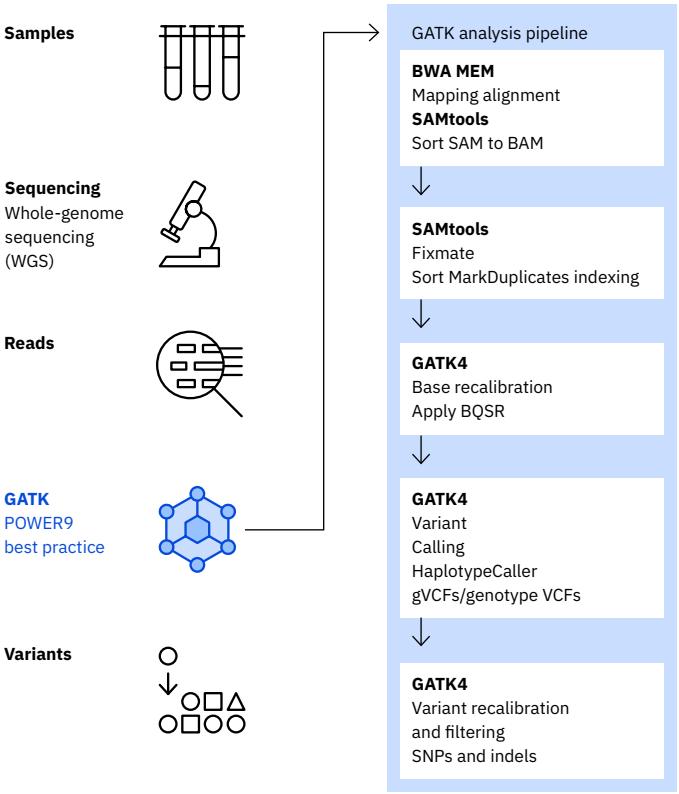


Figure 2: GATK4 germline best practice pipeline on the POWER9 system. This figure is modified from the Broad Institute blog. [2-1]

Sequence alignment and mapping to reference the genome
The paired-end sequence NA12878 inputs, ~48-49 GB gzip files, in FASTQ format are aligned or mapped to reference genome with BWA MEM. A total of 160 threads are used on the 40-core POWER9 system with 4 SMT threads per physical cores. The BWA output is directly piped and sorted to the BAM file with SAMtools.

BWA, SAMtools, GATK tools and the whole pipeline scripts are available for installation through GitHub clone of `https://github.com/ruzhuchen/GATK4-Power.git`. These applications were optimized for POWER9 processors with details in the next section.

```
for i in ${fastqFolder}/*_1.fastq.gz
do
  filename=$(basename $i _1.fastq.gz)
  mapFile=${workPath}/${filename}_bwa.bam
  mapped[${#mapped[*]}]=$mapFile
  bwa mem -t 160 -Ma \
   -R @RG\\tID:${filename}\\tSM:${filename}\\tPL:ILM\\
tLB:${filename} \
   $ref $i ${i%1.fastq.gz}2.fastq.gz \
   |samtools sort - -@ 40 -n -m 4G -T ${i%R1.fastq.gz} -o $mapFile
done
```

Sequence-marking duplicates
There are three ways for marking duplicates in reads after mapping and sorting the sequences. The use of the `gatk (picard) MarkDuplicates` tool is time-consuming where only a single thread is initiated. The latest SAMtools and the specific IBM Power Systems sam2bam tool use multithreads for marking duplicates in reads and significantly accelerate the runtime processes by more than 5 times without the loss of accuracy.

a) Use the `gatk MarkDuplicates` tool

```
for mapFile in ${mapped[*]}
do
gatk --java-options "-Xmx20G -XX:+UseParallelGC \
  -XX:ParallelGCThreads=4" MarkDuplicates \
       -I ${mapFile} -O ${mapFile%.bam}_dedup.bam \
       -M ${mapFile%_bwa.bam}.duplicate_metrics \
       --MAX_RECORDS_IN_RAM 5000000 -MAX_SEQS 5000000 \
       --OPTICAL_DUPLICATE_PIXEL_DISTANCE 2500 \
       --VALIDATION_STRINGENCY SILENT \
       -MAX_FILE_HANDLES 1000
done
```

b) Use SAMtools

```
for mapFile in ${mapped[*]}
do
  samtools fixmate -m -@ 40 ${mapFile} fixmate.bam
  samtools sort -@ 40 -m 6G -o sorted.bam fixmate.bam
  samtools markdup -s -@ 40 sorted.bam ${mapFile%.bam}_dedup.bam
  samtools index -@ 40 ${mapFile%.bam}_dedup.bam
done
```

c) Use sam2bam

```
for mapFile in ${mapped[*]}
do
   export use_storage_mode=yes
   export BAM_PAGEFILE=$workPath/pf
   sam2bam sam2bam -d -p -Fibm_markdup:r \
       -o${mapFile%.bam}_dedup.bam ${mapFile%bam}sam
done
```

Base quality score recalibration (BQSR) and Apply BQSR
To be able to run BQSR or Apply BQSR concurrently, either splitting sequence intervals or the spark mode is available to use with these tools. The sequence intervals are obtained by running `gatk ScatterIntervalsByNs` with reference genome, and the intervals list is divided by number of cores available on the system using `gatk SplitIntervals` tool to create scattered intervals lists.

a) Use split-sequence intervals

```
for mapFile in ${mapped[*]}
do
   for i in `seq -f '%04g' 0 39`
   do
      outfile=${mapFile%.bam}_dedup_recal_data_$i.table
      gatk --java-options "-Xmx4G -XX:+UseParallelGC \
        -XX:ParallelGCThreads=4" BaseRecalibrator \
        -L $i-scattered.interval_list -R $ref \
        -I ${mapFile%.bam}_dedup.bam $knownSiteArg -O $outfile &
   done
   wait
   for i in `seq -f '%04g' 0 39`
   do
      bqfile=${mapFile%.bam}_dedup_recal_data_$i.table
      output=${mapFile%.bam}_dedup_recal_$i.bam
      gatk --java-options "-Xmx4G -Xmx4G -XX:+UseParallelGC \
        -XX:ParallelGCThreads=4" ApplyBQSR -R $ref \
        -I ${mapFile%.bam}_dedup.bam \
         -L $i-scattered.interval_list  -bqsr $bqfile \
         --static-quantized-quals 10 --static-quantized-quals 20 \
         --static-quantized-quals 30 -O $output &
   done
   wait
done
```

b) Use the Spark mode

```
for mapFile in ${mapped[*]}
do
  gatk --java-options "-Xmx8G -XX:+UseParallelGC \
    -XX:ParallelGCThreads=4" BaseRecalibratorSpark \
-R $ref -I ${mapFile%.bam}_dedup.bam $knownSiteArg \
-O ${mapFile%.bam}_dedup_recal_data.table \
-- --spark-runner LOCAL --spark-master local[40] \
--conf spark.local.dir=$workPath
  gatk --java-options "-Xmx8G -XX:+UseParallelGC \
-XX:ParallelGCThreads=4" ApplyBQSRSpark -R $ref \
-I ${mapFile%.bam}_dedup.bam -bqsr $bqfile \
--static-quantized-quals 10 --static-quantized-quals 20 \
--static-quantized-quals 30 -O $output \
-- --spark-runner LOCAL --spark-master local[40] \
--conf spark.local.dir=$workPath
done
```

Variant calling using GATK HaplotypeCaller (HC)

The recalibrated BAM file from the previous step is used to perform variant calling per sample with the `gatk HaplotypeCaller` tool. The output is in GVCF mode, which can be used for joint genotyping with multiple samples. This step runs 40 processes concurrently with each process having 8 OpenMP threads. Native PairHMM library implemented with SIMD extension VSX is enabled with option "`-pairHMM VSX_LOGLESS_CACHING`".

```
# Used Onle native pairhmm
for i in `seq -f '%04g' 0 39`
 do
   infile=${mapFile%.bam}_dedup_recal_$i.bam
   outfile=${mapFile%.bam}_dedup_recal_$i.g.vcf
   gatk --java-options "-Xmx4G -Djava.library.path=$GATK_HOME/libs \
      -XX:+UseParallelGC -XX:ParallelGCThreads=1" HaplotypeCaller \
      -R ${ref} -I $infile -pairHMM VSX_LOGLESS_CACHING \
      -L $REF_HOME/intervals/40c/$i-scattered.interval_list \
      --native-pair-hmm-threads 8 \
      -O $outfile -ERC GVCF -stand-call-conf 10 &
done
wait
```

Consolidate and genotype genomic variant call formats (GVCFs)

Individual GVCF files are either consolidated into one GVCF file with `gatk CombineGVCFs` (or `gatk GatherVcfs`) or directly used for genotyping with `gatk GenotypeGVCFs` and then gathered into one genotype VCF file with `gatk GatherVcfs`.

```
# Combinge gvcf files (optional)
gatk --java-options "-Xmx4G" GatherVcfs -R $ref $gvcfFilesArg \
   -O ${vcfFile%vcf}g.vcf
# genotype gvcf files
for i in `seq -f '%04g' 0 39`
do
  gatk --java-options "-Xmx4G -XX:+UseParallelGC
-XX:ParallelGCThreads=4" \
    GenotypeGVCFs -R $ref -V ${mapFile%.bam}_dedup_recal_$i.g.vcf \
    -L $REF_HOME/intervals/40c/$i-scattered.interval_list \
    -O ${vcfFile%.vcf}_$i.vcf &
done
wait
# merge scattered phenotype vcf files
gatk --java-options "-Xmx4G" GatherVcfs -R $ref $vcfFilesArg -O
$vcfFile
```

Variant-quality score recalibration (VQSR) and filtering

The raw VCFs from previous step are filtered to achieve a high degree of sensitivity and reduce false positives. This process can be achieved by using variant quality score recalibration tools `gatk VariantRecalibrator` and `gatk ApplyVQSR` with single nucleotide polymorphisms (SNPs) and indel modes. This process is the last step of the pipeline and the final VCF file can be used for other genomic analyses.

```
# VARIANT QUALITY SCORE RECALIBRATION - SNPs
gatk --java-options "-Xmx4G -XX:+UseParallelGC
-XX:ParallelGCThreads=40" \
  VariantRecalibrator -V $vcfFile -O recalibrate_SNP.recal \
  -mode SNP --tranches-file recalibrate_SNP.tranches \
  -tranche 100.0 -tranche 99.9 -tranche 99.0 -tranche 90.0 -an
QD -an FS \
  -an MQRankSum -an ReadPosRankSum -an SOR -an MQ --max-
gaussians 6 \

-resource:hapmap,known=false,training=true,truth=true,prior=15.0
$vcfHapmap \
  -resource:omni,known=false,training=true,truth=true,prior=12.0
$vcfOmni \

-resource:1000G,known=false,training=true,truth=false,prior=10.0
$vcfGlk \
  -resource:dbsnp,known=true,training=false,truth=false,prior=7.0
$vcfDbsnp
# Apply recalibration to SNPs
gatk --java-options "-Xmx4G -XX:+UseParallelGC \
  -XX:ParallelGCThreads=20" ApplyVQSR -V $vcfFile \
  -O recalibrated_snps_raw_indels.vcf --recal-file recalibrate_
SNP.recal \
  --tranches-file recalibrate_SNP.tranches \
  -truth-sensitivity-filter-level 99.5 --create-output-variant-
index true \
  -mode SNP
# Run Variant Recalibrator - Indels
gatk --java-options "-Xmx4G -XX:+UseParallelGC
-XX:ParallelGCThreads=40" \
  VariantRecalibrator -V recalibrated_snps_raw_indels.vcf \
  -O $recalibrate_INDEL.recal \
  -mode INDEL --tranches-file recalibrate_INDEL.tranches \
  -tranche 100.0 -tranche 99.9 -tranche 99.0 -tranche 90.0 -an
QD \
  -an FS -an MQRankSum -an ReadPosRankSum -an SOR --max-
gaussians 4 \
  -resource:mills,known=false,training=true,truth=true,prior=12.0
$vcfMills \
  -resource:dbsnp,known=true,training=false,truth=false,prior=2.0
$vcfDbsnp
# Apply recalibration to Indels
gatk --java-options "-Xmx4G -XX:+UseParallelGC
-XX:ParallelGCThreads=20" \
  ApplyVQSR -V recalibrated_snps_raw_indels.vcf \
  -O ${vcfFile%.vcf}.recal.vcf --recal-file recalibrate_INDEL.
recal \
  --tranches-file recalibrate_INDEL.tranches \
  -truth-sensitivity-filter-level 99.0 \
  --create-output-variant-index true -mode INDEL
```

GATK4 performance tuning and optimization on the POWER9 system

As previously described, GATK4 performs poorly without necessary tuning and optimization. Improving the GATK best practice pipeline performance can be achieved by source code optimization, runtime optimization with multithreads or splitting inputs, as well as using alternative tools in the pipeline without losing accuracy.

Installation of GATK4 on Power Systems tools and pipeline scripts

The optimized BWA and SAMtools, sam2bam and installation scripts are cloned from GitHub using `git clone https://github.com/ruzhuchen/GATK4-Power.git`. Then, run "`install_gatk4-power9.x`" inside the `$GATK_HOME` directory to complete the GATK4 installation. These tools are built with the latest compilers with optimization options, mapping Intel SSE to IBM Vector Scalar Extensions (VSXs) and using the memory caching library. To install sam2bam separately, download the build script "build.sh" using "`wget https://raw.githubusercontent.com/OpenPOWER-HCLS/sam-to-bam/master/build.sh`" and then run the script. The "`samtools`" binary is generated and manually coped and renamed as "`sam2bam`" to `$GATK_HOME/bin`. Please note that IBM advance toolchain compilers (V9.0 and V10.0) are required for compiling the code. Additionally, the whole GATK V4.1.0.0 for the IBM Power Systems package is available for download from the link specified in GitHub.[7] Please contact the author at ruzhuchen@us.ibm.com for credential use.

Approaches to accelerate GATK4 pipeline on the POWER9 system

Several approaches have been used to improve GATK4 performance on the POWER9 system. These approaches include splitting sequence intervals, source code modifications, the use of `samtools` or `sam2bam` for marking duplicates in reads, native PairHMM library, runtime optimization through process binding and JVM options.[9] See Table 2.

Tools	Baseline	Source code	Runtime	Specific to IBM Power Systems
BWA + SAMtools	POWER9 system build using GCC 8.2.1	Mapped SSE to VSX, latest htslib 1.9, Jemalloc	Using SMT4, processor affinity	Using SMT4, processor affinity
MarkDuplicates	GATK4 default	SAMtools markup	SAMtools markup	sam2bam
BQSR/Apply BQSR	GATK4 Spark local or split intervals	Split intervals	Split intervals, JVM options	Split intervals, JVM options
Variant calling	GATK4 split intervals	Native PairHMM, split intervals, OpenMP	Native PairHMM, dynamic split intervals, OpenMP	Native PairHMM, dynamic split intervals, OpenMP

Table 2: Germline pipeline tuning and optimization on the POWER9 system

Performance results and discussion

The GATK4 best practice pipeline begins with paired-end WGS alignment with BWA MEM to variant-quality recalibration and filtering. The BWA and SAMtools are multithreaded tools where numbers of 160 and 40 threads are used, respectively, for sequence alignment and sorting. The GATK4 tools are run with splitting data by number of cores on the system and processing each interval concurrently, in which the "`--intervals or -L`" option is available. The benchmark results shown in Table 3 summarize the runtime performance with each optimization approach.

Workload pipeline	Baseline + split intervals	+ Source code optimization	+ Runtime optimization	+ P9-specific optimization
BWA + sort	4.66	4.72	4.69	3.59
MarkDuplicates	8.68	1.97	1.62	1.05
BQSR	0.72	0.55	0.64	0.33
Apply BQSR	0.43	0.43	0.42	0.41
HaplotypeCaller	4.42	4.45	2.53	2.50
CombineGVCFs	0.80	0.76	0.14	0.14
GenotypeVCFs	0.05	0.05	0.05	0.05
VQSR-SNPs	0.45	0.5	0.42	0.53
ApplyVQSR-SNPs	0.03	0.03	0.03	0.03
VQSR - Indels	0.15	0.16	0.15	0.16
ApplyVQSR-Indels	0.02	0.03	0.02	0.03
Total time (Hrs.)	**20.41**	**13.65**	**10.71**	**8.82**

Table 3: GATK4 benchmark of 50x WGS NA12878 germline pipeline on the POWER9 system

When running with splitting data input as sequence intervals, the pipeline runs 4.2x faster than the baseline. With applying source code changes and replacing `gatk MarkDuplicates` with SAMtools, a 50 percent performance improvement was observed. By tuning and optimizing the runtime, further performance improvement is achieved by 30 percent. The optimal performance, where sam2bam is used for duplicate marking, is 9.7x faster than baseline and finishes the whole pipeline workload in 8.8 hours, where more than 85 hours are required for single threaded process. See Figure 3.

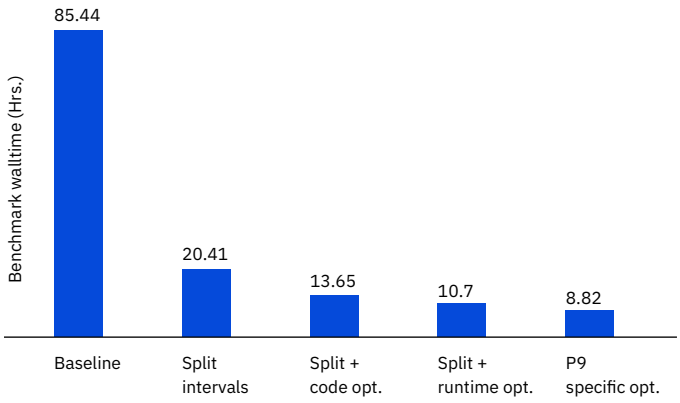


Figure 3: Performance comparison of Power9 tuning and optimization

In summary, with a 40-core OpenPOWER AC922 system, the optimal best practice pipeline is to use optimized IBM Power Systems with BWA, SAMtools, sam2bam, native PairHMM lib and GATK scripts specified in section B to fully use the system resources. Further optimization of the GATK4 pipeline can be achieved by better load balancing of the genomic data.

Running GATK4 on an IBM Cloud Private cluster

Description of IBM Cloud Private

IBM Cloud Private,[8] powered by Kubernetes, Docker and Cloud Foundry, is an application platform for developing and managing on-premises, containerized applications. It's an integrated environment for managing containers that includes a private image repository, a management console and monitoring frameworks. GATK4 on IBM Cloud Private is designed to be an easy-to-use environment to run a genomic pipeline without the need to set up a runtime environment.

GATK4 container for IBM Power Systems

The `Dockerfile` for the GATK4 Power Systems container is available for download from GitHub.[7] The docker image is built and uploaded to the IBM Cloud Private cluster with the following command:

```
# GATK 4 container
% docker login mycluster.icp:8500
% docker build -t gatk:4.1.0.0 .
% docker tag gatk:4.1.0.0 mycluster.icp:8500/admin/gatk:4.1.0.0
% docker push mycluster.icp:8500/admin/gatk:4.1.0.0
```

After loading the GATK4 container to the IBM Cloud Private cluster, use `kubectl` to deploy the GATK Pod with GPFS as persistence storage. The document to create a GPFS persistence volume and claim is described in the IBM Cloud Private manual.

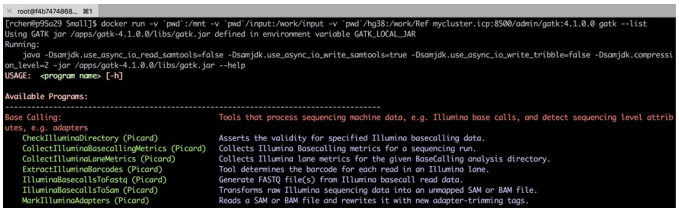
```
# Deploy GATK4 Pod
# configuration file gatk4.yaml
kind: Pod
apiVersion: v1
metadata:
  annotations: kubernetes.io/psp: ibm-privileged-psp
  name: gatk4-power
  namespace: kube-system
spec:
  volumes:
    - name: gpfs-storage
      persistentVolumeClaim:
       claimName: gpfs2mb_pvc
  containers:
    - name: gatk4-power
      image: mycluster.icp:8500/admin/gatk:4.1.0.0
      imagePullPolicy: IfNotPresent
      volumeMounts:
        - mountPath: /var/run/secrets/kubernetes.io/
serviceaccount"
          name: gpfs-storage
# deployment
% kubectl apply -f gatk4.yaml
```

Running GATK4 on the IBM Cloud Private cluster

There are two ways to run GATK4 on the IBM Cloud Private cluster.

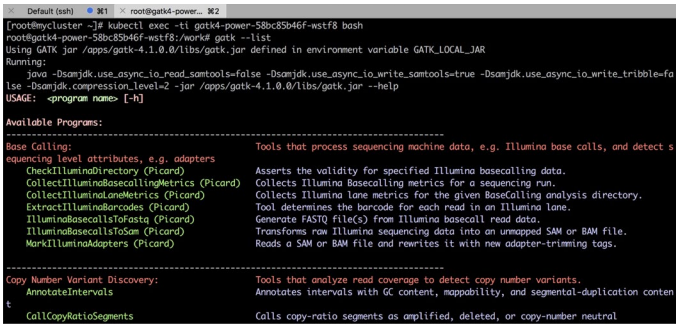
1) Use `kubectl` interactively

```
% kubectl exec -it gatk4-power – bash
```



2) Use docker

```
% Docker run –rm –it -v `pwd`/input:/work/
input -v `pwd`:/work \ mycluster.icp:8500/
admin/gatk:4.1.0.0 bash
```



Acknowledgements

Special thanks to Veera (Chandana) Solasa and Michael K. Meyers in IBM Poughkeepsie Client Center for providing system support and troubleshooting. I would also like to thank my colleagues Farid Parpia, Michael Ackaway, Frank Lee and Joanna Wong for their helpful technical assistance, solutions discussion and project management, and to Hal Porter for his management support.

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
July 2019

1 “Genome Analysis Toolkit.” *Broad Institute*, 2019. https://software.broadinstitute.org/gatk
2 “Introduction to the GATK Best Practices.” *Broad Institute*, January 9, 2018. https://software.broadinstitute.org/gatk/best-practices
3 “Platinum Genomes project.” *Illumina*. https://www.illumina.com/platinumgenomes.html
4 Takeshi Ogasawara, Yinhe Cheng and Tzy-Hwa Kathy Tzeng. “Sam2bam: High-Performance Framework for NGS Data Preprocessing Tools.” *PLOS ONE*, November 18, 2016. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0167100
5 SAMtools. http://samtools.sourceforge.net
6 “Resource Bundle: Reference materials for human analysis.” *Broad Institute*, 2019. https://software.broadinstitute.org/gatk/download/bundle
7 “GATK4-Power.” *GitHub*, 2019. https://github.com/ruzhuchen/GATK4-Power
8 IBM Cloud Private overview. ibm.com/support/knowledgecenter/en/SSBS6K_2.1.0.2/getting_started/introduction.html
9 Jacob R. Heldenbrand, Saurabh Baheti, Matthew A. Bockol et al. “Performance benchmarking of GATK3.8 and GATK4.” *Mayo Clinic and Illinois Strategic Alliance for Technology-Based Healthcare*, June 2018. https://www.biorxiv.org/content/biorxiv/early/2018/06/18/348565.full.pdf
10 Dino Quintero, Luis Bolinches, Marcelo Correia Lima, Katarzyna Pasierb and William dos Santos. “IBM Reference Architecture for Genomics, Power Systems Edition.” *IBM Redbooks®*, April 2016. http://www.redbooks.ibm.com/redbooks/pdfs/sg248279.pdf

56026356USEN-01