Investigating Genome Analysis Pipeline Performance on GATK with Cloud Object Storage

Tatsuhiro Chiba and Takeshi Yoshimura

IBM Research
Agenda

- Background, Motivation and Challenges

- Performance Scalability Analysis
  - GATK with Spark/HDFS
  - GATK with Spark/COS

- Performance and Cost Optimization

- Summary
Genome Analysis and GATK

**Genome Analysis**
- Next Generation Sequencing (NGS) generates tons of genome sequence data
- DNA structure analysis is essential for medical & life scient research
- High scalable system is always required to optimize genome analysis
  1. Accelerating Speed
  2. Reducing Cost

**What is GATK?**
- most widely used genome analysis toolkit written in Java
- contains many tools and utilities, such as data preprocessing/cleanup, sequence data quality control by recalibration, HaplotypeCaller, etc.
- Users can build their own workflow pipeline to perform variant discovery analysis by combining those tools

**Example of Variant: SNP**

Reference: T G A C G A T A G C C

Sample: T G A C G G T A G C C
GATK Best Practices and Spark

- GATK Team defines typical variant discovery workflows as **GATK Best Practices**
- GATK leverages Spark to achieve node-level / core-level scalability
- **ReadsSparkPipeline**, a Spark program in GATK, performs a set of well-defined workflows

**typical pipeline for variant discovery**

Main steps for Germline Single-Sample Data

- **PRE-PROCESSING**
  - Raw Unmapped Reads
  - Map to Reference
  - Raw Mapped Reads
  - Mark Duplicates
  - Recalculate Base Quality Scores

- **VARIANT DISCOVERY**
  - Call Variants Pre-Sample
    - HaplotypeCaller in single sample mode
  - Call Variants Post-Sample
  - Filter/VariantTranches

- **CALLSET REFINEMENT**
  - Raw SNPs + Indels
    - CNNScoreVariants
    - Filter/VariantTranches

- **FUNCTIONALITY ANNOTATED VARIABLES**
  - Analysis-Ready Reads
    - VCF
  - SNPs and Indels

- **Further Analysis**
  - GATK Pipeline
  - Storage (Local File System, HDFS, Object Storage, etc.)

**A Chain of Spark Jobs in ReadsSparkPipeline**

- **GATK Pipeline**
  - BWA
  - Mark Duplicate
  - BQRS
  - Haplotype Caller

- **Focused Steps**
  - RAW data (FASTQ Format)
  - Whole Genome (BAM Format)

- **Variant Result** (VCF Format)

---

https://gatk.broadinstitute.org/hc/en-us/articles/360035535932-Germline-short-variant-discovery-SNPs-Indels-

---

---

---

---

---

---

---

---

---

---
Migrating Genome Analysis Platform from Local to Clouds

What benefits do we want to achieve after migration?
- Cost Efficiency
- Performance Scalability

What is needed to take full advantage of Cloud Capabilities?
- Decoupling compute and storage for Resource Elasticity
  - Storage Elasticity: Cloud Object Storage rather than self-managed local storage
  - Compute Elasticity: Containers rather than self-managed nodes
- Adjusting resource demands dynamically

What problems do we need to solve?
- GATK + Spark reference architecture heavily depends on HDFS (i.e. tightly-coupled)
- Data loading and system setup are not negligible overhead
- Performance characteristics in analysis pipeline are different
Challenges and Summary of Contributions

**Challenges**
- reveals **performance characteristics** in Genome analysis pipeline
- decouples compute and *storage* to exploit cloud elasticity
- adjusts resource capacity dynamically based on the pipeline demands

**Contributions**
- Identifies performance scalability and elasticity issues in Genome analysis pipeline running on GATK with Spark/HDFS
- Provides a new best practice to use **Cloud Object Storage** instead of **HDFS**
- Demonstrates the entire pipeline improvement
  - **Performance**: up to 28% faster
  - **Cost**: up to 67% cost saving

![Diagram showing From Local to Clouds transition]

- **Local**: GATK (Pipeline), VM, Volume
- **Spark + HDFS**: GATK (Spark), Spark, HDFS, VM, Volume
- **Spark + COS**: GATK (Spark), Spark, Object Storage, VM, Volume, IBM COS, AWS S3
- **Spark + COS + Kubernetes**: GATK (Spark), Spark, PVC, Object Storage, Kubernetes
- **WDL/Cromwell + Cloud Backend Executor**: WDL, Cromwell, Google Cloud Life Sciences API, AWS Batch, HPC Scheduler
Agenda

- Background, Motivation and Challenges

- Performance Scalability Analysis
  - GATK with Spark/HDFS
  - GATK with Spark/COS

- Performance and Cost Optimization

- Summary
GATK Performance Scalability with Spark/HDFS
Performance Analysis at Scale: Spark/HDFS (1/4)

- Built Spark/HDFS cluster on IBM Cloud
  - Spark w/ HDFS (20K-IOPS): Attached 10 IOPS/GB profiled 1TB volume ... (up to 20,000 IOPS)
  - Spark w/ HDFS (3K-IOPS): Attached 3 IOPS/GB profiled 1TB volume ... (up to 3,000 IOPS)
  - Utilized both storage volumes independently to understand how disk speed makes an impact to the performance

Software and Spark Configuration

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATK</td>
<td>4.1.7.0 (latest w/ COS support)</td>
</tr>
<tr>
<td>Spark</td>
<td>2.4.5 (latest)</td>
</tr>
<tr>
<td>Hadoop</td>
<td>2.7.7</td>
</tr>
<tr>
<td>JVM</td>
<td>1.8.0_242 (OpenJ9)</td>
</tr>
</tbody>
</table>

Spark Config
- Executors/Node: 4
- Cores/Executor: 8
- Mem/Executor: 35GB(heap), 15GB(off-heap)

Execution Command on Driver Node

```bash
$ gatk ReadsPipelineSpark
-I WGS-94982-NA12878-no-NC_007605.bam
-O WGS-94982-NA12878.vcf
-R human_g1k_v37.fasta
--known-sites dbsnp_138.b37.vcf.gz
--pairHMM AVX_LOGLESS_CACHING_OMP
--max-reads-per-alignment-start 10
--java-options "-XX:MaxDirectMemorySize=8589934592"
--disable-sequence-dictionary-validation true
--spark-runner SPARK --spark-master spark://master:7077
--executor-cores 8 --num-executors 48 --executor-memory 35g ...
```
Performance Analysis at Scale: Spark/HDFS (2/4)

- GATK Pipeline has a good scalability (5.5x scaling against 6x resources)
- Based on Job execution time breakdown, 20K-IOPS can reduce Spark Job 0 and 1 time drastically
- Compared to speed up ratio in each job, most of jobs has a good scalability
- Job 0 and 1 are disk read heavy *(loading Genome Data from HDFS)*
- Job 3, 4, 5 are disk write and network heavy (writing intermediate data and shuffling them between nodes)
- Last Job 7 has many shuffle read and CPU intensive (sorting in-memory data and writing a result into HDFS)
Performance Analysis at Scale: Spark/HDFS (4/4)

- In 3K-IOPS, disk r/w bandwidth is bounded up to 45 MB/sec in total
- Disk read happens in Job 0 and 1 only
- Almost all shuffled data resides in memory as file cache.

**Resource Usage on HDFS (3K-IOPS)**

**CPU Usage (%)**

- Job 0, 1, 2, 3, 4, 5, 6, 7

**Disk R/W (GB/sec)**

- Read
- Write

**Network R/W (GB/sec)**

- Send
- Receive

**Memory Usage (GB)**

- Used
- Buffer
- Cache

**Resource Usage on HDFS (20K-IOPS)**

**CPU Usage (%)**

- Job 0, 1, 2, 3, 4, 5, 6, 7

**Disk R/W (GB/sec)**

- Read
- Write

**Network R/W (GB/sec)**

- Send
- Receive

**Memory Usage (GB)**

- Used
- Buffer
- Cache
What Challenges Still Exist?

Storage Elasticity
- HDFS (20K-IOPS) is quite faster, but spark jobs does not always require high-throughput disk
- Hard to resize HDFS capacity/nodes, and need to keep paying high cost even if not required
- load time (copying data to HDFS) is not negligible
- Cloud Object Storage (COS), such as AWS S3 & IBM Cloud Object Storage, has a capability to overcome the limit of storage scalability → Can we utilize COS instead of HDFS for GATK?

HDFS vs. Cloud Object Storage
- POSIX File System vs REST API Based Storage
- Performance depends on DISK bandwidth vs. Network bandwidth

Architecture Overview
- Spark (Connector) can access COS via Hadoop FileSystem API
- Data load time (COS): only once even if resizing compute resource
- Data load time (HDFS): every time if resizing compute resource
GATK Performance Scalability with Spark/COS
**Experiment Settings: GATK on Spark with COS vs. HDFS**

- Read and write genomics dataset **not from/to HDFS but from/to COS directly**
- Modified GATK to use `cos://bucket/object`, and integrated with Stocator (Spark Connector for COS)
- Compared systems: Spark w/ HDFS (3K-IOPS), w/ HDFS (20K-IOPS), w/ COS (3K-IOPS), and w/ COS (20K-IOPS)

**VPC Architecture**

- **VPC Gen2 (London)**
  - Zone (eu-gb-3)
  - **Total: 12 Nodes, 384 vCores, 3TB RAM**
  - Driver VM
  - Worker VM (mx2-32x256)
  - 1TB Disk
  - Run GATK on Spark/HDFS

**Software and Spark Configuration**

- **Software**
  - GATK: 4.1.7.0 (latest w/ COS support)
  - Spark: 2.4.5 (latest)
  - Hadoop: 2.7.7
  - JVM: 1.8.0_242 (OpenJ9)

- **Spark Config**
  - Executors/Node: 4
  - Cores/Executor: 8
  - Mem/Executor : 35GB(heap), 15GB(off-heap)

**Software Stack Overview**

- **GATK (ReadsPipelineSpark)**
- **Spark**
- **Hadoop**
- **Object Storage Connector** (S3A, Stocator, etc.)
- **Local Disk**

**Genomics dataset (BAM, FASTA, etc.)**

- WGS-G94982-NA12878-no-NC_007605.bam (154GB)
- dbSNP_138.b37.vcf (10GB)
- human_g1k_v37.fasta (3GB)

**Cloud Object Storage**

- **WGS**
- **G94982**
- **NA12878**
- **no-NC**
- **007605.bam** (154GB)
- **dbsnp_138.b37.vcf** (10GB)
- **human_g1k_v37.fasta** (3GB)

**Total: 12 Nodes, 384 vCores, 3TB RAM**
Weak Scaling Performance – COS vs. HDFS

- GATK Pipeline has a good scalability in both cases basically (achieved 5.5x scaling against 6x resources)
- COS (20K-IOPS) case is slightly worse scaling than other three (explain it later)
- As for resource usage on Spark w/ COS, disk bandwidth is consumed only by shuffle write
- Instead, Spark w/ COS can highly utilize network capacity
Why GATK with COS does not scale well in many nodes?

- GATK with COS has an additional cost to write a result (VCF file)
  - Takes around 10 mins for finalizing 1GB output file onto COS, depending on the result size

- Why HDFS does not have the overhead, but COS has?
  - Difference in the supported file system operations between HDFS and COS
  - HDFS supports (logical) concat operation on the file system inside, but COS does not

- GATK explicitly calls concat operation in the finalization phase
  - HDFS can complete concat operation without any copies (just logically move it on HDFS)
  - Object Storage connector cannot support concat operation directly (several copies happen between systems)
Performance and Cost Optimization for GATK with Spark/COS
Protocol Detail and Optimization in Finalization Phase

**HDFS**
- each reducer tasks store the parts of files into HDFS
- Main program calls Concat Hadoop FileSystem API → just move and set a representative name to VCF file

**COS (Original)**
- each reducer tasks store the parts of files into COS
- Main program calls Concat Hadoop FileSystem API → NotSupportedException Exception
- As an exception handling, main driver gathers all pieces locally, merges them, and stores it back to COS

**COS (Opt)**
- each reducer tasks store the parts of files into COS
- Main program calls Concat Hadoop FileSystem API → implemented a dummy concat operation
- delegates concat task to client (VCF reader)
- constantly eliminate data sink time (i.e. 10 mins) → COS (3K-IOPS) is up to 28% faster than HDFS (3K-IOPS)
Price/Performance Comparison: COS vs. HDFS

**Solid Line:**
- Comparing computation pipeline time in COS (Original) with HDFS
- COS (Original) always achieves better cost performance than HDFS
- 3K-IOPS is **15 – 60 % better** cost performance than 20K-IOPS

**Dotted Line:**
- Comparing whole clock time in COS (Opt) with HDFS, which includes data loading time (i.e. 30 mins)
- Achieved **up to 67% cost saving** with COS (Opt) w/ 3K-IOPS, and **up to 61% w/ 20K-IOPS**

**TABLE III**

<table>
<thead>
<tr>
<th>System setup time</th>
<th>create volumes</th>
<th>create instances</th>
<th>load data into HDFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>elapsed time</td>
<td>56 sec</td>
<td>2.5 mins</td>
<td>30 mins</td>
</tr>
</tbody>
</table>

Lower is Better
Conclusion

Summary
- Identified performance scalability and elasticity issues in Genome analysis pipeline running on GATK with Spark/HDFS
- Provided a new best practice to use Cloud Object Storage instead of HDFS
- Demonstrated the entire pipeline improvement
  - Performance: up to 28% faster
  - Cost: up to 67% cost saving

Next Steps
- Demonstrates compute elasticity with container & Kubernetes
- Applies our investigation results and optimization to GATK + Cromwell