Towards Compute Flexibility for Genome Analysis in the Hybrid Cloud

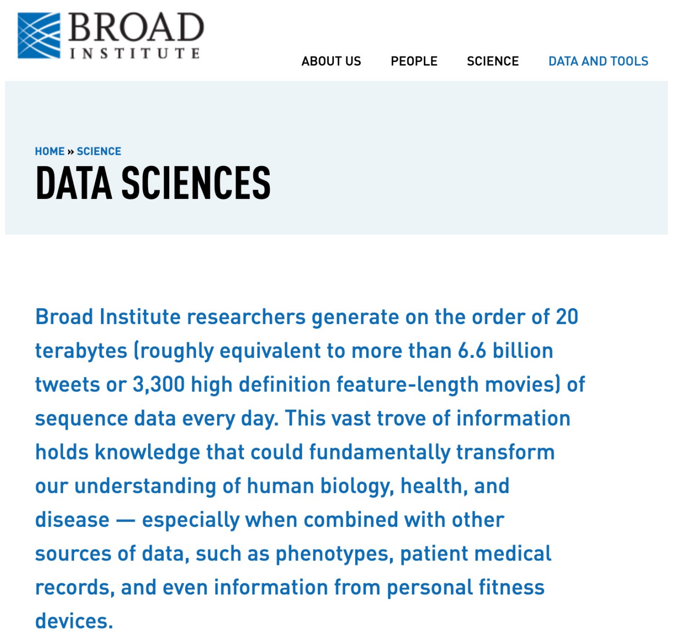
Takeshi Yoshimura, Tatsuhiro Chiba

IBM Research – Tokyo

tyos@jp.ibm.com

Genome analysis

- A key element for medical and life sciences
  - e.g., variant discoveries
- Problem: big data
  - Next Generation Sequencing generates tons of genome sequence data every day

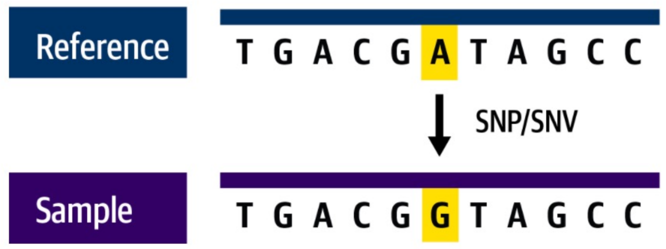


DATA SCIENCES

Broad Institute researchers generate on the order of 20 terabytes (roughly equivalent to more than 6.6 billion tweets or 3,300 high definition feature-length movies) of sequence data every day. This vast trove of information holds knowledge that could fundamentally transform our understanding of human biology, health, and disease — especially when combined with other sources of data, such as phenotypes, patient medical records, and even information from personal fitness devices.

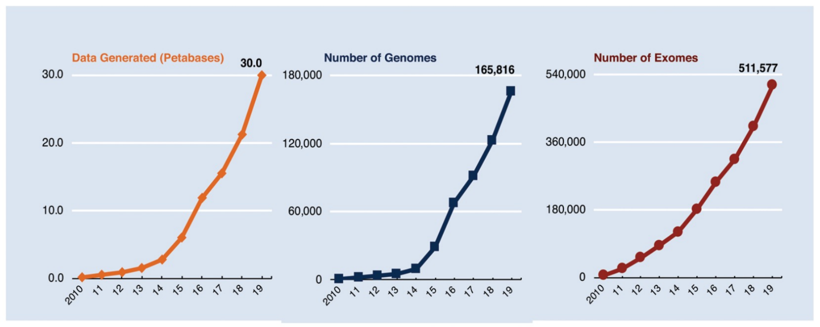
https://www.broadinstitute.org/data-sciences

Example of Variant: SNP



Ref: Fig 2-6 of Genomics in the Cloud, O'Reilly Media

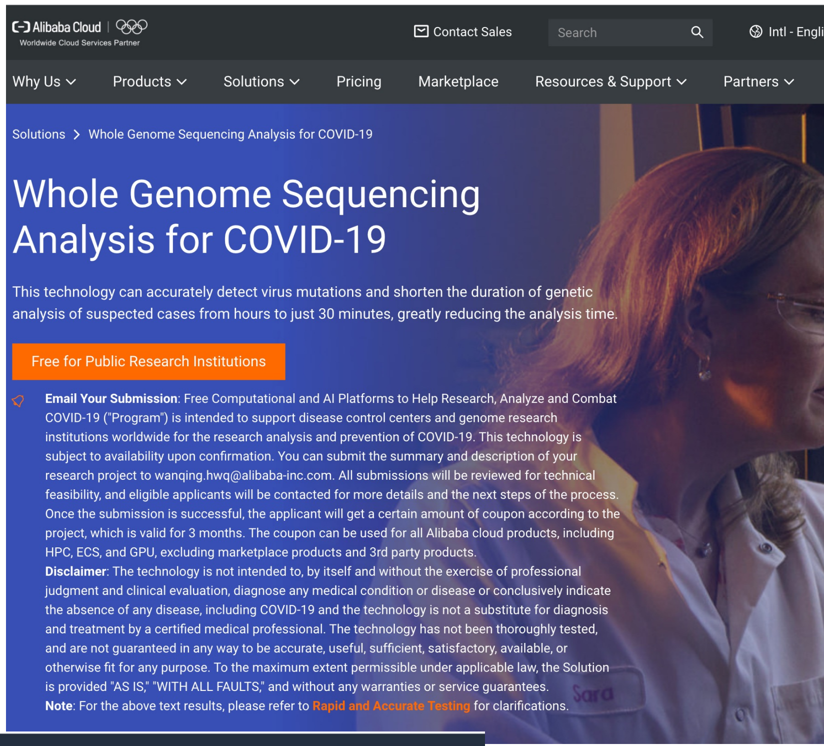
Growth in Data Production at the Broad Institute



Ref: Fig 1-1 of Genomics in the Cloud, O'Reilly Media

Genomics in the cloud

- Google, AWS, etc. offer platforms for genome analysis with ***GATK, WDL, and Cromwell***
  - Huge datastore with cloud object storage (COS)
  - Cost-efficient batch processing with cluster autoscaling

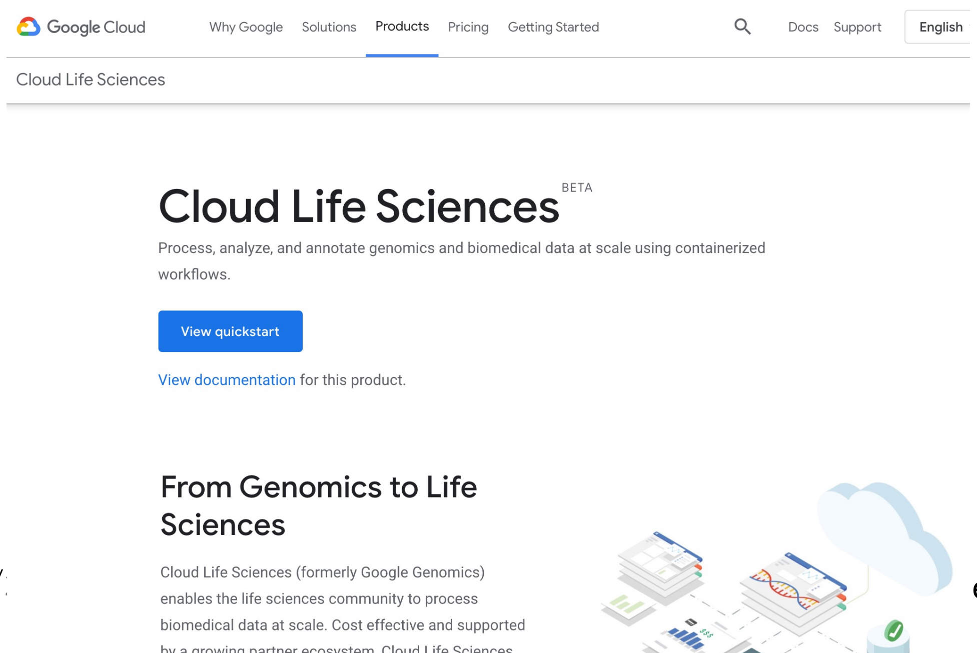


Whole Genome Sequencing Analysis for COVID-19

This technology can accurately detect virus mutations and shorten the duration of genetic analysis of suspected cases from hours to just 30 minutes, greatly reducing the analysis time.

Free for Public Research Institutions

Email Your Submission: Free Computational and AI Platforms to Help Research, Analyze and Combat COVID-19 ("Program") is intended to support disease control centers and genome research institutions worldwide for the research analysis and prevention of COVID-19. This technology is subject to availability upon confirmation. You can submit the summary and description of your research project to wanqing.hwq@alibaba-inc.com. All submissions will be reviewed for technical feasibility, and eligible applicants will be contacted for more details and the next steps of the process. Once the submission is successful, the applicant will get a certain amount of coupon according to the project, which is valid for 3 months. The coupon can be used for all Alibaba cloud products, including HPC, ECS, and GPU, excluding marketplace products and 3rd party products.
Disclaimer: The technology is not intended to, by itself and without the exercise of professional judgment and clinical evaluation, diagnose any medical condition or disease or conclusively indicate the absence of any disease, including COVID-19 and the technology is not a substitute for diagnosis and treatment by a certified medical professional. The technology has not been thoroughly tested, and are not guaranteed in any way to be accurate, useful, sufficient, satisfactory, available, or otherwise fit for any purpose. To the maximum extent permissible under applicable law, the Solution is provided "AS IS," "WITH ALL FAULTS," and without any warranties or service guarantees.
Note: For the above text results, please refer to Rapid and Accurate Testing for clarifications.



Cloud Life Sciences BETA

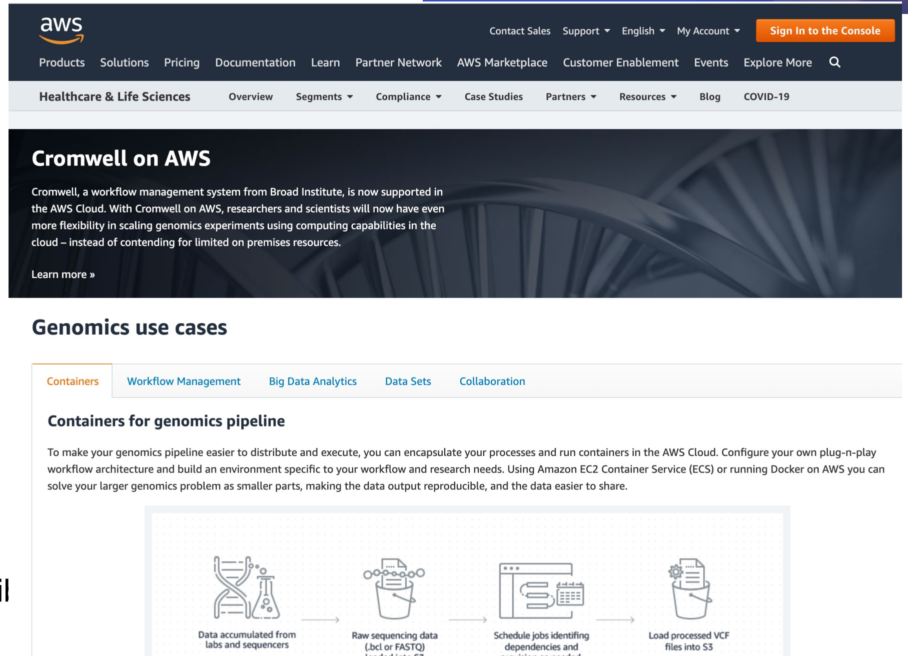
Process, analyze, and annotate genomics and biomedical data at scale using containerized workflows.

View quickstart

View documentation for this product.

From Genomics to Life Sciences

Cloud Life Sciences (formerly Google Genomics) enables the life sciences community to process biomedical data at scale. Cost effective and supported by a growing partner ecosystem. Cloud Life Sciences



Cromwell on AWS

Cromwell, a workflow management system from Broad Institute, is now supported in the AWS Cloud. With Cromwell on AWS, researchers and scientists will now have even more flexibility in scaling genomics experiments using computing capabilities in the cloud – instead of contending for limited on premises resources.

Learn more »

Genomics use cases

Containers for genomics pipeline

To make your genomics pipeline easier to distribute and execute, you can encapsulate your processes and run containers in the AWS Cloud. Configure your own plug-n-play workflow architecture and build an environment specific to your workflow and research needs. Using Amazon EC2 Container Service (ECS) or running Docker on AWS you can solve your larger genomics problem as smaller parts, making the data output reproducible, and the data easier to share.

Data accumulated from labs and sequencers
Raw sequencing data (.bcl or FASTQ)
Schedule jobs identifing dependencies and
Load processed VCF files into S3

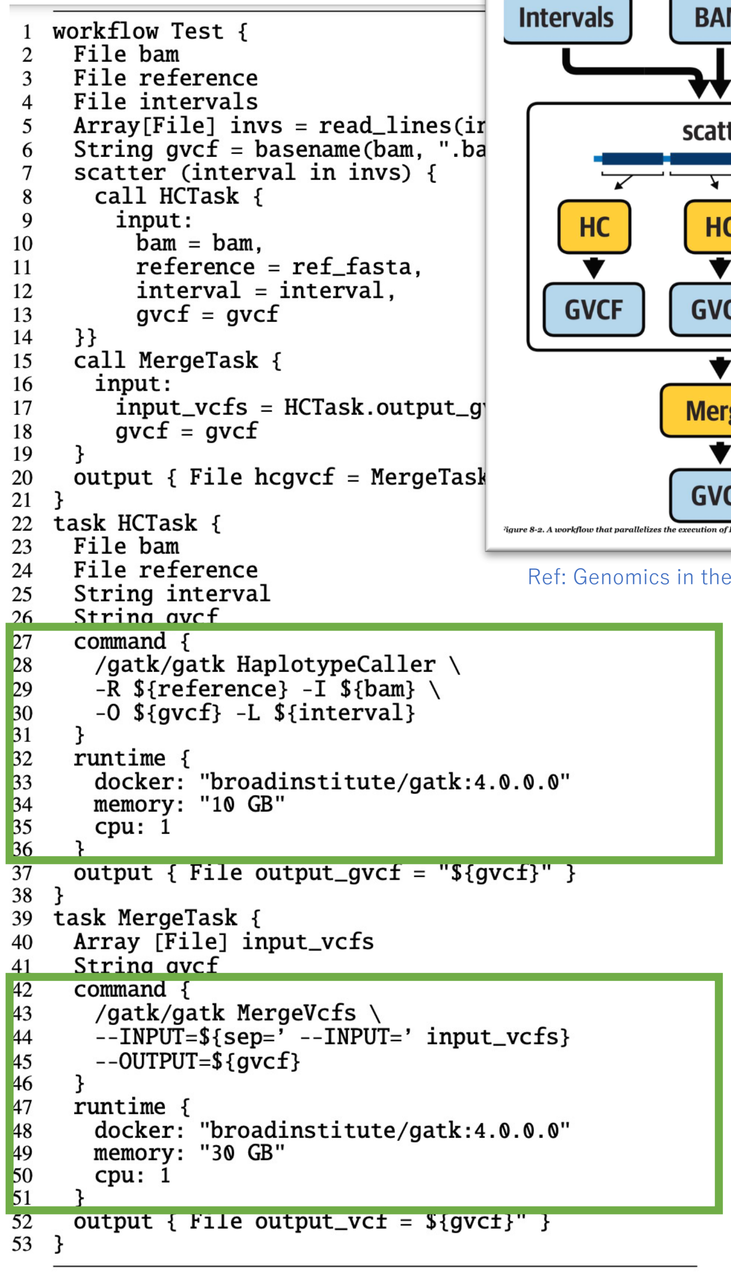
GATK – genomics in the cloud (1/3)

- CLI command collection for genome analysis
  - e.g., data preprocessing, variant discovery
  - Support scatter-gather parallelization
  - Enable genome pipelines with various subcommands

```
$ gatk HaplotypeCaller ¥
  -R Homo_sapiens_assembly38.fasta ¥
  -I NA12878_24RG_small.hg38.bam ¥
  -O part-0/NA12878_24RG_small.g.vcf.gz ¥
  -L hg38_wgs_scattered_calling_intervals.txt ¥
...
$ gatk MergeVcfs ¥
  --INPUT=part-0/NA12878_24RG_small.g.vcf.gz  ¥
  --INPUT=part-1/NA12878_24RG_small.g.vcf.gz ¥
  --OUTPUT=NA12878_24RG_small.g.vcf.gz
```

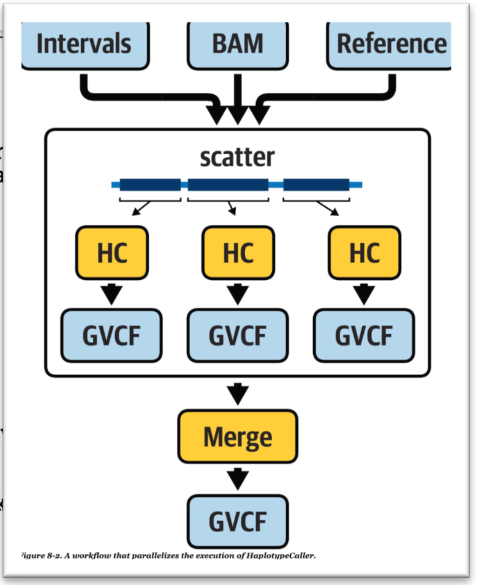
WDL – genomics in the cloud (2/3)

- Workflow Description Language
  - Define complex, reproducible pipelines
  - Define inputs/outputs for each task, e.g., GATK commands
  - Support scatter-gather parallelization
  - Assume containers with Docker images as execution runtimes
- Genome pipelines use GATK and common Linux utils (e.g., python)
  - https://github.com/gatk-workflows



```
workflow Test {
  File bam
  File reference
  File intervals
  Array[File] invs = read_lines(ir
  String gvcf = basename(bam, ".ba
  scatter (interval in invs) {
    call HCTask {
      input:
        bam = bam,
        reference = ref_fasta,
        interval = interval,
        gvcf = gvcf
  }}
  call MergeTask {
    input:
      input_vcfs = HCTask.output_g
      gvcf = gvcf
  }
  output { File hcgvcf = MergeTask
}
task HCTask {
  File bam
  File reference
  String interval
  String gvcf
  command {
    /gatk/gatk HaplotypeCaller \
    -R ${reference} -I ${bam} \
    -O ${gvcf} -L ${interval}
  }
  runtime {
    docker: "broadinstitute/gatk:4.0.0.0"
    memory: "10 GB"
    cpu: 1
  }
  output { File output_gvcf = "${gvcf}" }
}
task MergeTask {
  Array [File] input_vcfs
  String gvcf
  command {
    /gatk/gatk MergeVcfs \
    --INPUT=${sep=' --INPUT=' input_vcfs}
    --OUTPUT=${gvcf}
  }
  runtime {
    docker: "broadinstitute/gatk:4.0.0.0"
    memory: "30 GB"
    cpu: 1
  }
  output { File output_vcf = "${gvcf}" }
}
```

Fig. 1 Example WDL.

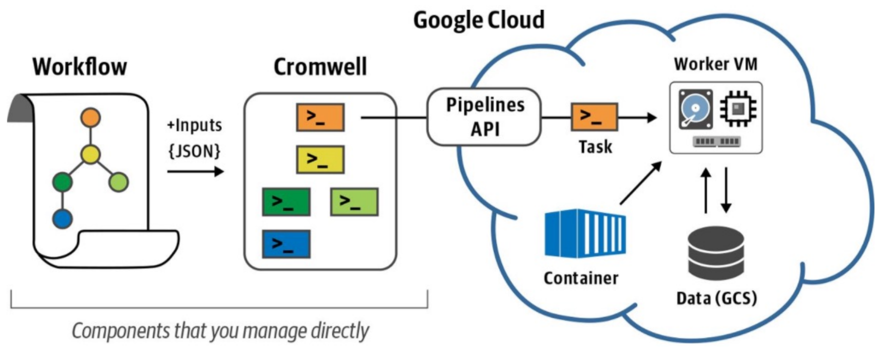


igure 8-2. A workflow that parallelizes the execution of HaplotypeCaller.

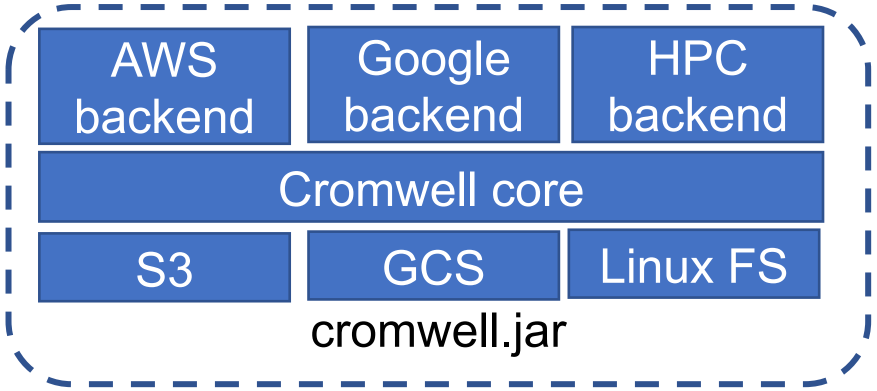
Ref: Genomics in the Cloud, O'Reilly Media

Cromwell – genomics in the cloud (3/3)

- A workflow engine to execute WDL files in the cloud
  - Translate each task in WDL files into a container job
    - A job runs main commands (e.g., GATK) but also copies input/output file from/to task container FS and COS
  - Call target cloud APIs to start containers
  - Support different clouds with various backends



Example execution flows with Google Cloud



* Pipelines API is now called as Cloud Life Sciences APIs (https://cloud.google.com/life-sciences/docs/apis)

Problem: Vendor lock-in

- Cromwell needs new backends for each new infrastructure
- New infrastructures may not have rich cloud features
  - User-friendly cluster management, cluster autoscaling for cost-saving
- Huge data may be already on existing COS
  - Data migration is not practical
  - On-premise/multi-/hybrid- cloud clusters may need to access multi-COS

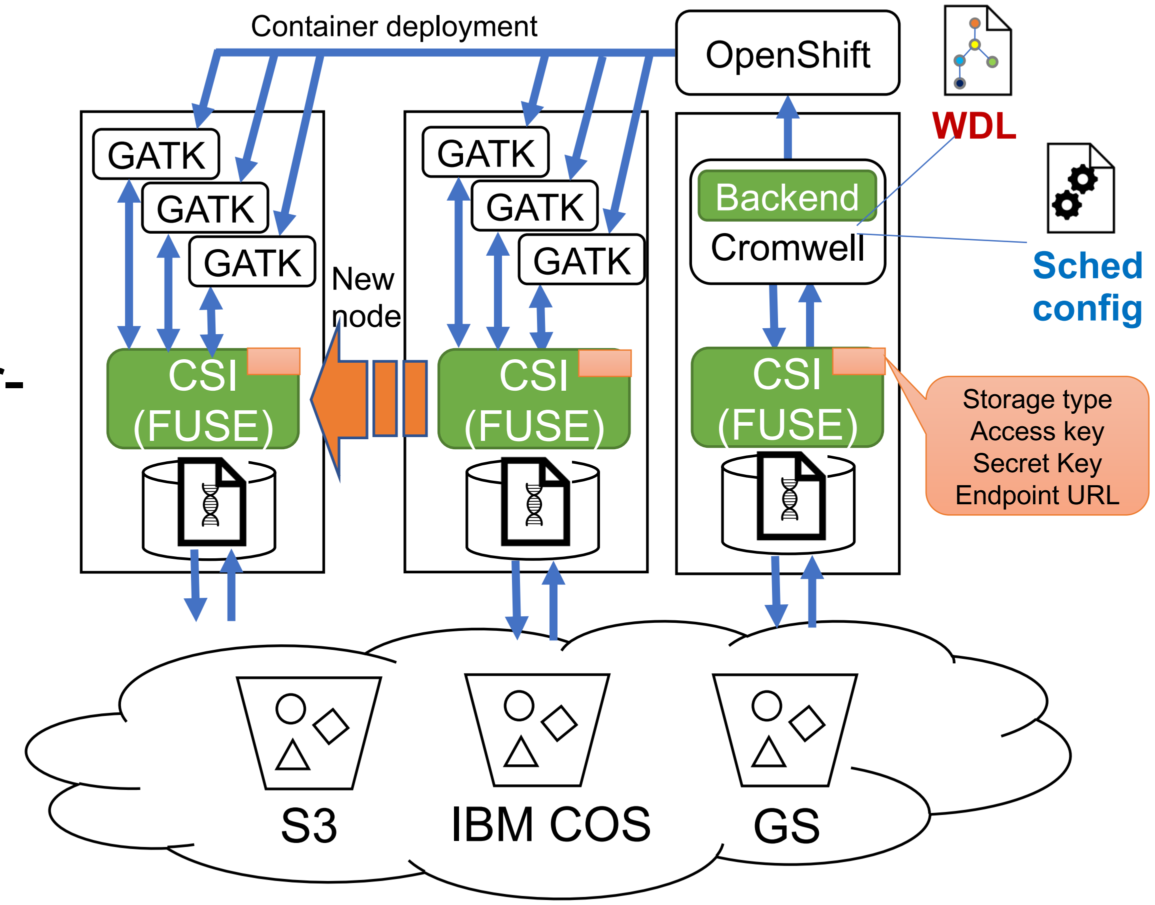
Goals

- Build a new Kubernetes (K8s) backend for Cromwell
  - Provide rich cluster management on any clouds
  - Utilize OpenShift for our experiments
- Leverage K8s customizability for optimization
  - Leverage CSI* to enable and optimize multi-COS accesses via a Linux FS
  - Leverage the ClusterAutoscaling add-on for cost-saving
- Show existing workflows with a multi-cloud environment
  - Connecting to on-premise is future work

* CSI: container storage interface to let containers attach custom volumes through a container orchestration system (e.g., mount a FS on a container local path with a persistent volume claim in Kubernetes)

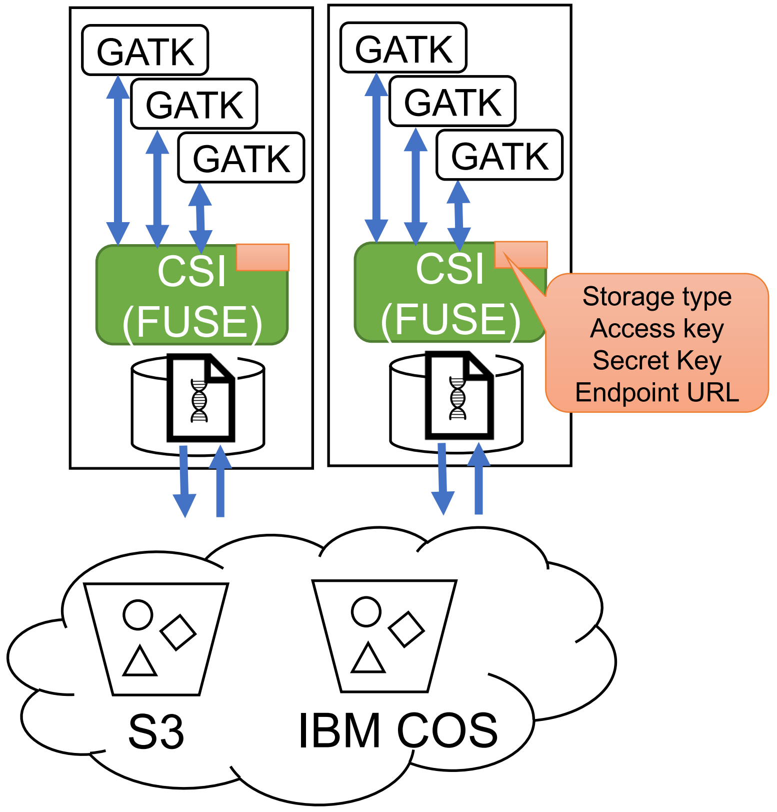
Architecture overview

- CSI driver for multi-COS
  - Enable multi-COS as a Linux FS
  - Deduplicate redundant I/O
- WDL translation into K8s jobs
  - Reduce file copies between container-local FS and COS
- Job scheduling config
  - Reduce compute costs with cluster autoscaling

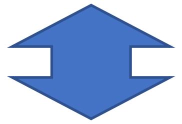


CSI driver for multi-COS

- Modify Goofys (FUSE for S3) to mount COS buckets as FS dirs under a single, root dir
  - Load a K8s secret for COS credentials
  - Redirect file I/O into existing storage backends according to accessed FS paths
- Mount a single FUSE at a node with --bind
  - Let containers share Linux page cache to deduplicate redundant I/O
  - Use *write-through* mode to simplify cache consistency at cluster scaling



```
s3://s3-gatk-test-data/path/to/A
cos://ibm-cos-output/path/to/B
```



```
/csi-root/s3-gatk-test-data/path/to/A
/csi-root/ibm-cos-output/path/to/B
```

WDL translation

- Reuse WDL translations in other backends
- Optimize file copies: direct reads and indirect writes from/to CSI paths
  - Most of workflows read files sequentially but some need random writes

```
gatk HaplotypeCaller ¥
  -R gs://s3-gatk-test-data/Homo_sapiens_assembly38.fasta ¥
  -I gs://s3-gatk-test-data/NA12878_24RG_small.hg38.bam ¥
  -O s3://ibm-cos-output/part-0/NA12878_24RG_small.g.vcf.gz ¥
  -L gs://s3-gatk-test-data/hg38_wgs_scattered_calling_intervals.txt ¥
...
```

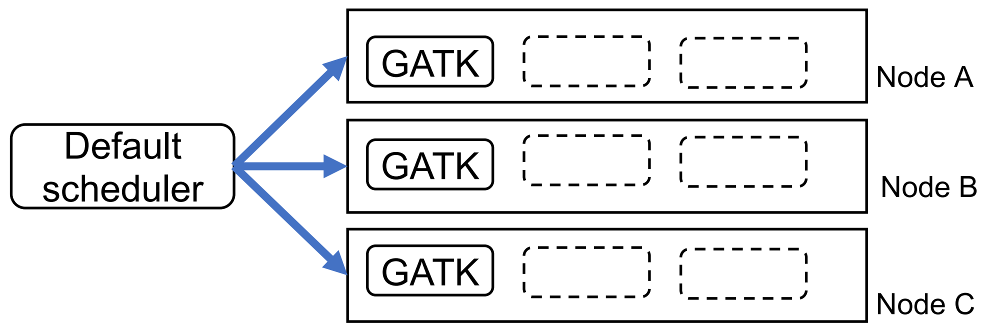
```
gatk HaplotypeCaller ¥
  -R /csi_root/s3-gatk-test-data/Homo_sapiens_assembly38.fasta ¥
  -I /csi_root/s3-gatk-test-data/NA12878_24RG_small.hg38.bam ¥
  -O /tmp/ibm-cos-output/part-0/NA12878_24RG_small.g.vcf.gz ¥
  -L /csi_root/s3-gatk-test-data/hg38_wgs_scattered_calling_intervals.txt ¥
...
cp /tmp/ibm-cos-output/part-0/NA12878_24RG_small.g.vcf.gz ¥
   /csi_root/ibm-cos-output/part-0/NA12878_24RG_small.g.vcf.gz
```

Job scheduling config

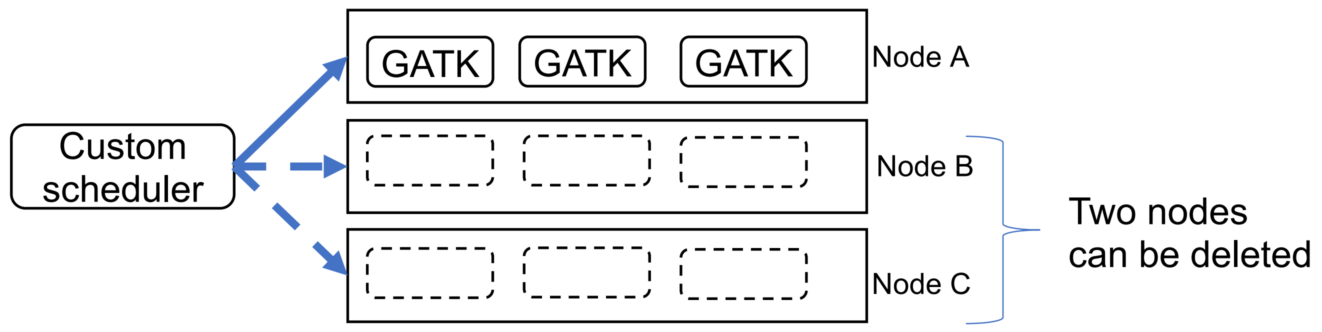
- Customize job scheduling for cluster autoscaling
  - Custom scheduler: prioritizing most-frequently used nodes
  - Support node selector and taint tolerations

```
backend {
  default = "k8s"
  providers { k8s {
    actor-factory = "..."
    config {
      auth = "k8sauth"
      filesystems {
        s3 { auth = "s3_auth" },
        gcs {auth = "no_auth" }
      }
      namespace = "cromwell"
      k8sServiceAccountName = "cromwell-sa"
      pullImageSecrets = ["regcred"]
      s3PvcName = "cos-pvc"
      root = "/cromwell_root/cos-bucket/cromwell"
      schedulerName = "my-scheduler"
      tolerations = "app=cromwell:NoSchedule"
      nodeSelector = "nodeType:cromwell"
    }
  }}
}
```

cromwell.conf



Default: least-frequently used



Custom: most-frequently used

Comparison with other backends

- File copy optimization
  - Many backends need file copies before/after jobs
  - HPC backends depend on hard links but FUSEs for S3 do no support them
- Job scheduling for autoscaling
  - Public clouds and LSF have autoscaling
  - TESK does not allow job scheduling suitable to autoscaling
- Multi-COS supports
  - No existing backends support multi-COS

Infrastructure	Copy opt.	Autoscaling	Multi-COS
AWS, Google	No	Yes	No
TESK	No	No	No
LSF	Yes	Yes	No
Our backend	Yes	Yes	Yes

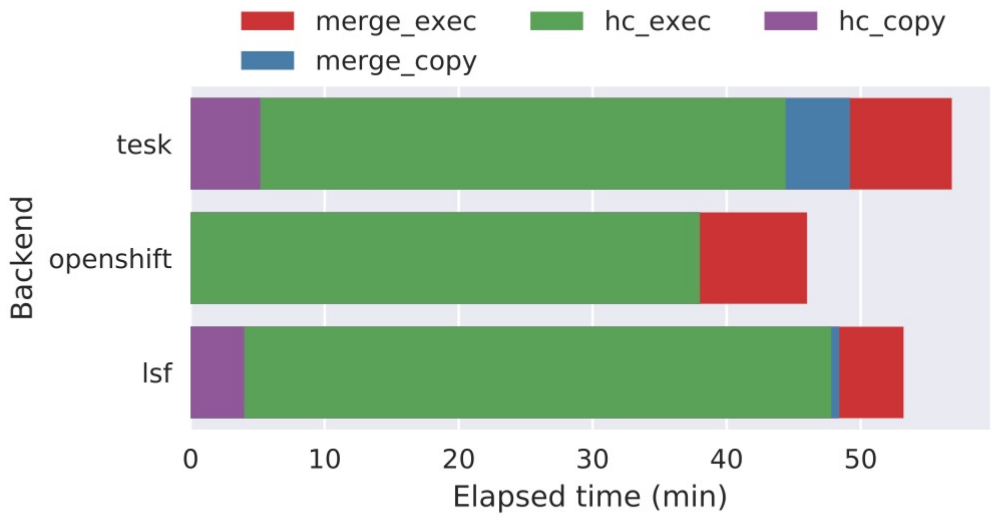
Experiments

- Run Cromwell on managed OpenShift 4.6 on IBM Cloud
- Goal:
  - Show performance improvements by file copy reduction
  - Show cost efficiency with cluster autoscaling and custom job scheduling
  - Run existing workflows for Google Cloud on IBM Cloud
- Experiments:
  - Performance comparison with LSF and TESK backends using an example scatter-gather workflow
  - Performance comparison with and without cluster autoscaling using an existing best-practice workflow for Google Cloud

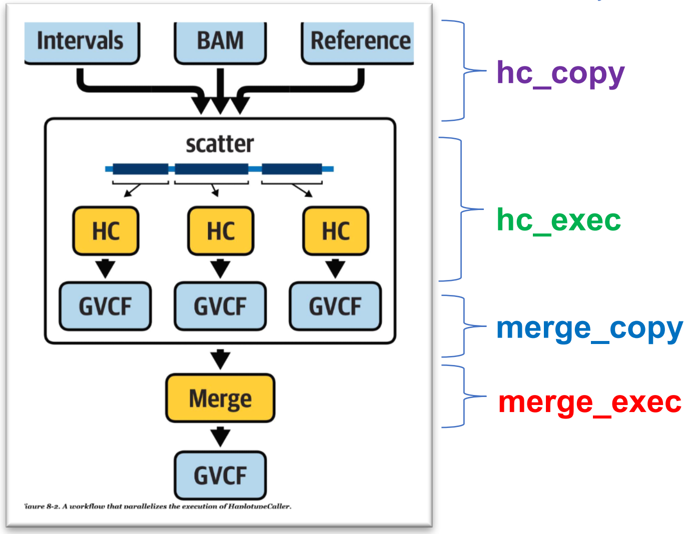
Experiment #1: Copy reduction

- OpenShift backend reduced the total elapsed time by 14% and 20% compared to LSF and TESK
  - Breakdown showed file copies were the major reason of speedups
  - Execution time was also slightly improved
    - FUSE bind-mount could deduplicate COS accesses with page cache

IBM Cloud (jp-tok), 10 nodes of bx2-8x32 (8 vCPUS, 32 GB RAM, 16 Gbps network, 100-GB, 3kIOPS block storage)
TESK creates 10GB, 100 IOPS block storage for each task

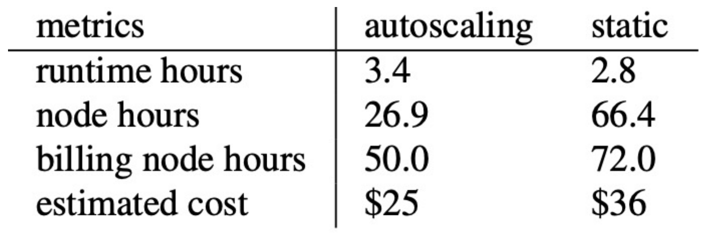


Ref: Genomics in the Cloud, O'Reilly Media



Experiment #2: Cluster autoscaling

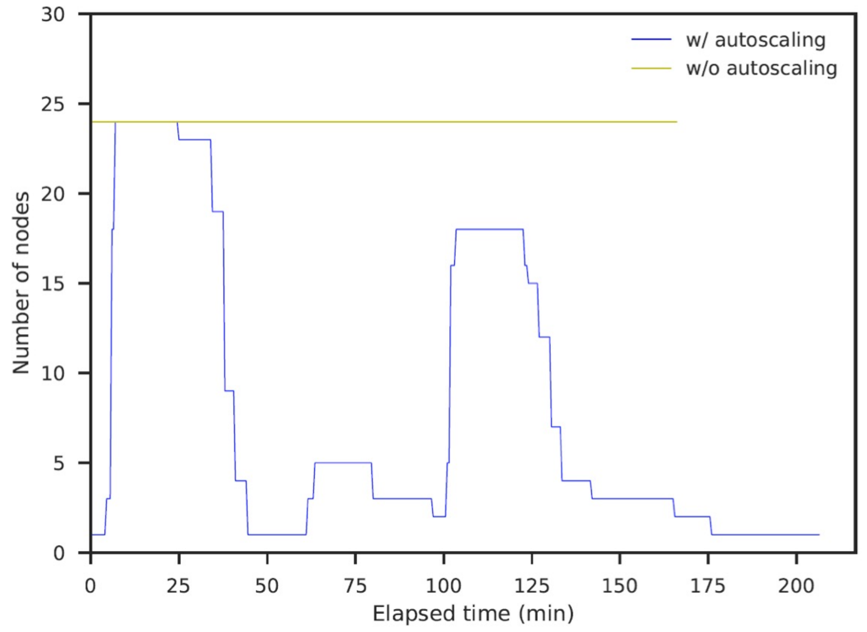
- Cluster autoscaling improved total compute costs by 31%
  - Custom job scheduling successfully co-locate as many pods as possible
- Cluster resource utilization was still <50%

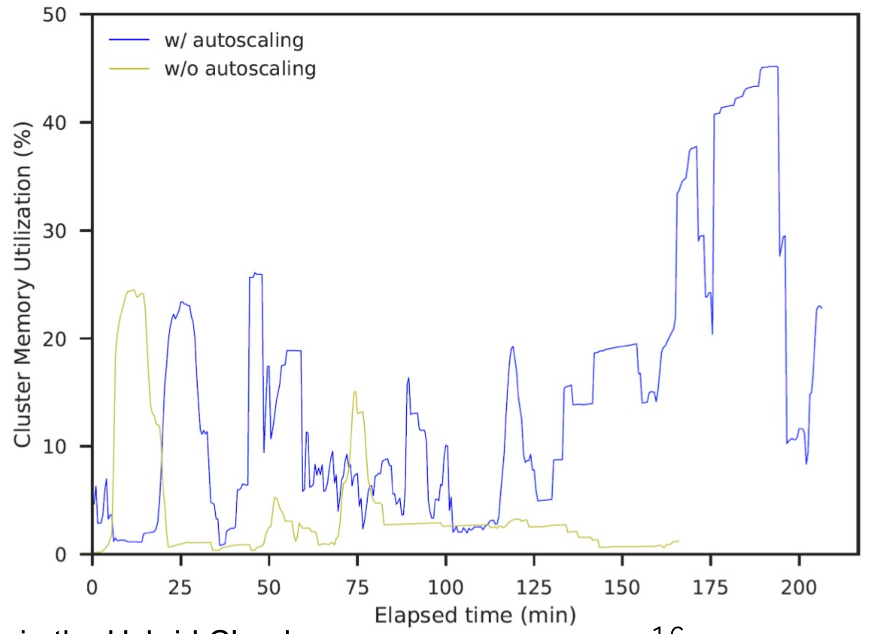


metrics	autoscaling	static
runtime hours	3.4	2.8
node hours	26.9	66.4
billing node hours	50.0	72.0
estimated cost	\$25	\$36

Fig. 6 Cost and performance.

IBM Cloud (jp-tok), 1 - 24 nodes of bx2-32x128 (32 vCPUS, 128 GB RAM, 16 Gbps network, 100-GB, 3kIOPS block storage)





Summary

- Genome analysis is a key element for medical and life sciences
- GATK, WDL, and Cromwell enable genomics in the cloud, but have a problem of vendor lock-in
- This work leveraged K8s to run a genome workflow on multi-COS environments
  - File copy reduction speeded up a scatter-gather workflow by 14%
  - Cluster autoscaling reduced compute costs by 31%
  - However, there is still room for improving resource utilization

Paid internship info (our team is in Hybrid cloud)

- https://www.ibm.com/jp-ja/employment/#jobs?%23jobs=&job-search=trl