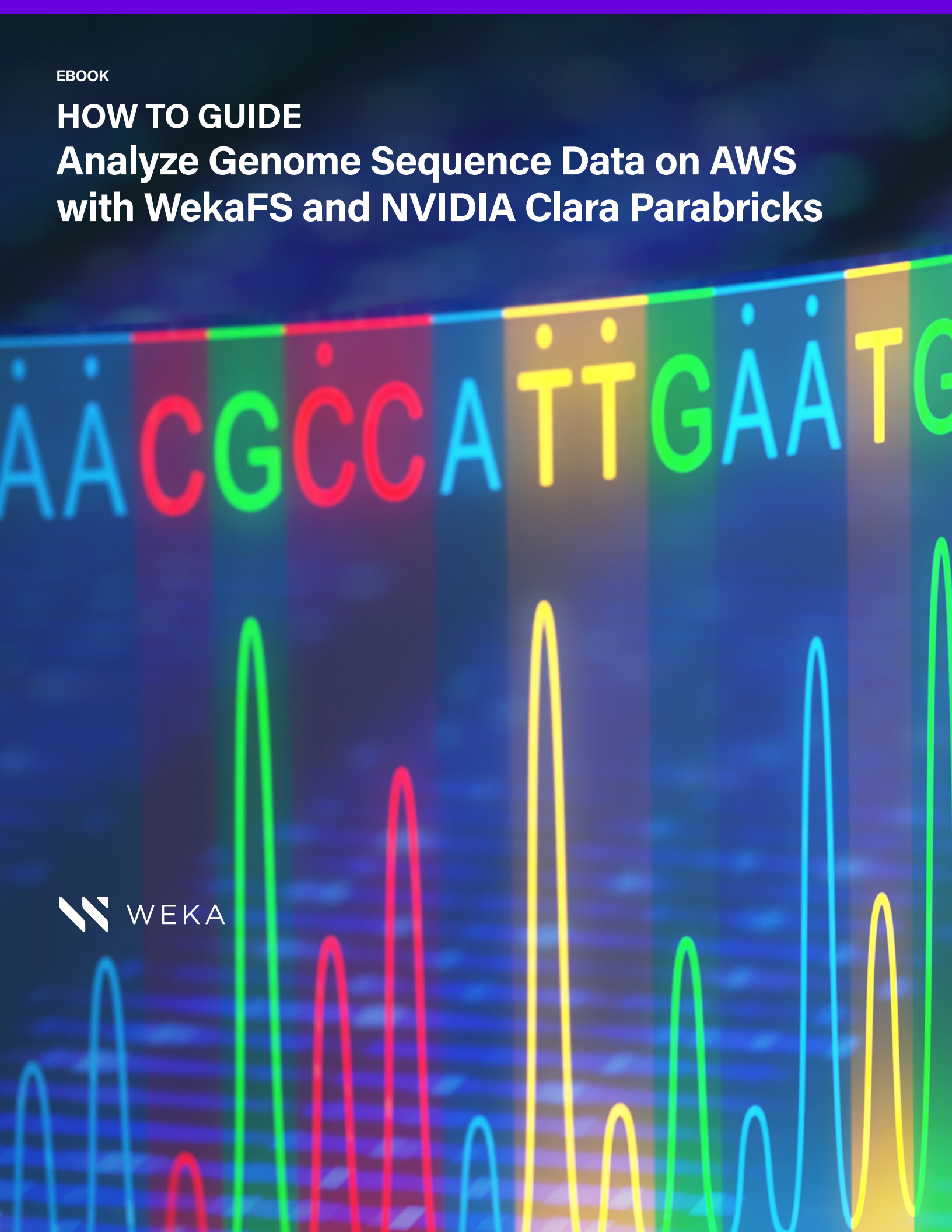
EBOOK

HOW TO GUIDE
Analyze Genome Sequence Data on AWS with WekaFS and NVIDIA Clara Parabricks



WEKA

TABLE OF CONTENTS

INTRODUCTION

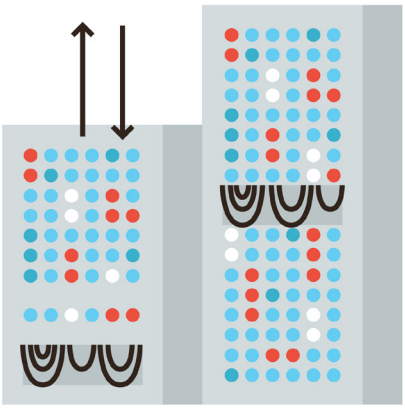
Whole-genome sequencing has become an essential and foundational part of genomic research, enabling researchers to identify genetic signatures associated with diseases, differentiate sequencing errors from biological signals, and better characterize various organisms' genomes. With pandemics like COVID-19 that threaten human life, characterizing and understanding genomes is now more crucial than ever. Commercially available, next-generation sequencing platforms allow researchers to decode an entire human genome in less than a day. This acceleration in research capabilities can help understand the susceptibility with infection by SARS-COV-2; can be used as the basis for vaccine creation; and can be used for therapy selection for an individual based upon their unique genetic signatures, along with many other use-cases.

Traditionally, sequencing a whole human genome takes multiple days, coupled to a heavy compute power using CPU resources. The Genome Analysis Toolkit (GATK), developed by the Broad Institute, is a multi-purpose software suite for analyzing DNA and RNA based sequence data with the primary goal to identify genetic variants, and it is generally considered the industry standard toolset. The GATK suite contains several tools, including the Base Quality Score Recalibration (BQSR), the Burrows Wheeler Aligner Maximal Effect Match (BWA MEM) for aligning and calibrating genomes, and the HaplotypeCaller, which efficiently identifies variants in sequences.

NVIDIA Clara Parabrick Pipelines replicate the functionality of GATK while harnessing GPUs' power to provide the fastest method to analyze genomes. The software, leveraging NVIDIA's CUDA libraries to accelerate key algorithms, dramatically reduces the time required to analyze a sequence compared to the CPU-only GATK tools and includes Google's state-of-the-art DeepVariant algorithm. Combined with the power of Amazon Web Services (AWS) and NVIDIA's storage partner WekaIO™ (Weka), the high performance sequence analysis of NVIDIA Clara Parabricks integrates with the convenience of the cloud to offer a robust, fast, and simple solution to enable faster genomic research over existing state-of-the-art CPU-based solutions, offering over 33x improvements in performance with over 99.9% concordance of results.

This eBook is a how-to guide targeted for genomic researchers, data scientists, and the IT teams supporting them, the latter responsible for creating architectures to run sequence analysis jobs. This eBook covers rationale for various components, creation of the Weka filesystem in AWS, setting up clients to run the germline variant calling pipeline using Parabricks, various optimizations for both the storage and GPUs to provide the best performance, plus an analysis of all results gathered during the tests. Sufficient details will be provided in order to replicate these results on your own.

CHAPTER 1:
The Weka File System



Weka offers WekaFS™, the world's fastest file system, a parallel file system built on NVMe, boasting high throughput and low latencies in an easy-to-manage package. Not only does Weka provide on-premises solutions through OEM storage vendors, but they also support cloud deployments and hybrid environments, integrating seamlessly with AWS. Weka was a natural choice for this workflow as they target many industries, including artificial intelligence (AI), life sciences, and financial services (FinServ). Given today's modern workflow requirements for large shared storage, low latencies, and the ability to work in the cloud, Weka is well positioned to support those trying to analyze sequence data rapidly. Above all else, Weka is simple to set up and use.

Hybrid Model and Cloud Bursting

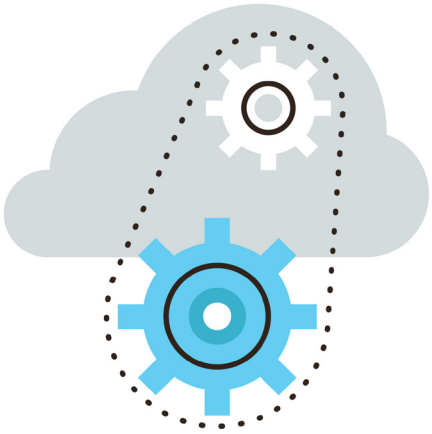
Suppose this is your introduction to Parabricks Pipelines. In that case, you may wish to begin by experimenting with the pipelines and pulling data on a local server until you are comfortable with the process. This process will allow you to verify all tools' functionality and validate your dataset before scaling up to more extensive tests. In our test lab, we used an on-premises Weka cluster to store all data and easily archive all data in an S3 bucket in AWS, allowing us to pull the data later from any cluster with access to AWS servers.

This process strategy has several advantages, with one being the ability to experiment on a development machine to understand how the tools work before running on more expensive or time-sensitive production servers, whether on-premises or in the cloud. Weka makes it easy to transition data to the cloud and tier data to larger, slower storage with a single global namespace. Just a few clicks in Weka's web interface is all it takes to set up this connection.

CPU Baseline

To get a good feel for how quickly Parabricks can analyze genomes while leveraging NVIDIA GPUs, we decided to run a quick test of the latest CPU-optimized pipelines available to the community. Sadly, these tests were anything but "quick." It took us the better part of a week to finally complete the tests successfully. While the wall-clock time for this analysis could certainly be optimized with better pipelining of the stages, the best-case compute time still took a staggering 33 hours to analyze a single genome, which is 33 times longer than it takes Parabricks with four V100 GPUs to analyze the same genome with equivalent accuracy. Meanwhile, there was a considerable amount of risk with a pipeline requiring that much time to complete. In the unfortunate scenario that one of the stages fails or access to a system goes away, I could have lost several days' worth of progress.

CHAPTER 2:
AWS Instance Creation



Creating the WekaFS File System

Before starting the germline variant analysis pipeline, several EC2 instances plus an S3 object store need to be provisioned to house both the file system and the GPU clients. You will be configuring a Weka file system to store and read your dataset, given that its high-performance characteristics are more than capable of handling this genomics work. Luckily, the setup process is simple, thanks to the pre-configured scripts provided at start.weka.io. To configure a file system on AWS, open start.weka.io in a browser and perform the following steps:

1. In the window on the left side of the page, set the total capacity to 10TB.
2. Click the tiering option, so it is set to use both SSD and S3.
3. Select the appropriate region for the work, if applicable. Otherwise, the default value will suffice.
4. In the main window, click the "Deploy to AWS" button corresponding to the "i3en.2xlarge" instance type; this will create a file system comprised of 7 instances with a total SSD capacity of 13.5TB and a maximum bandwidth of 5GB/sec.
5. Ensure that "Add clients to the cluster" is checked in the menu that pops up, as you want Weka's CloudFormation script to configure EC2 instances for us automatically.
6. For Client Type, select the p3.8xlarge instance type; this is a GPU-optimized instance with 4x 16GB V100 GPUs. This instance type has proven to be more stable than its bigger brother, p3.16xlarge, and is the preferred instance for this work.
7. Set the number of clients to 4, which will bring the total GPU count to 16 for the cluster, though this number can be tweaked depending on your local needs.
8. Select a Custom AMI for the AMI type, and enter "ami-08a5e4d75a3914234" for the AMI ID; this is the "Deep Learning AMI (Ubuntu 18.04) Version 24.1" image which is optimized for GPU workloads. While newer versions of the Deep Learning AMIs exist, this version uses kernel version 4.15.0-1065-aws, which is supported by Weka version 3.7.2, while newer kernel versions are not yet supported.
9. After the above steps are complete, click "Deploy to AWS" to begin the deployment process. At this point, the window should look similar to Figure 1.

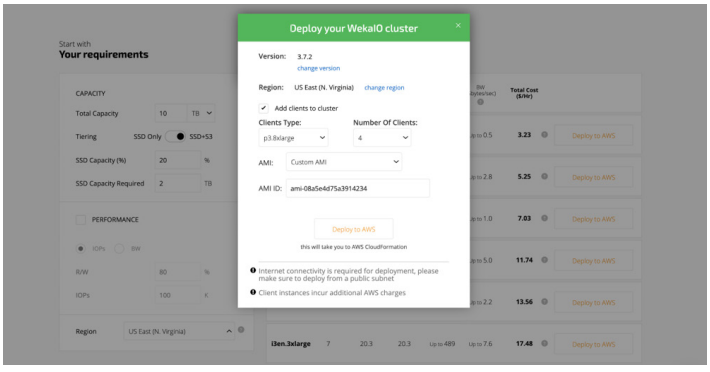


Figure 1: configuring a cluster to deploy on start.weka.io

After following the above steps, a new page will load to configure the AWS CloudFormation stack. This template will automatically build all of our EC2 instances and mount the filesystem on your clients, minimizing the time required to get up and running. The following steps will finish the setup of Weka on AWS. Note that at this point, you will need to be logged into whichever account you plan on using for AWS.

1. Enter a name for the cluster in the "Stack name" field, such as "weka-parabricks-germline".
2. In the VPC and Subnet fields, select the desired values based on your environment.
3. Select "Internet Facing" for the Load Balancer Type to make it easy to connect to the filesystem GUI externally.
4. Select an appropriate key to connect to the instances based on your needs.
5. Leave the remaining values at their default settings.
6. Check the box to acknowledge that AWS CloudFormation might create IAM resources.
7. Click the "Create stack" button at the bottom of the page; this will initiate the CloudFormation creation process and begin building a Weka file system with four connected EC2 clients on AWS.
8. After several minutes, the CloudFormation script will show as completed with a "CREATE_COMPLETE" status in green, as seen in Figure 2.

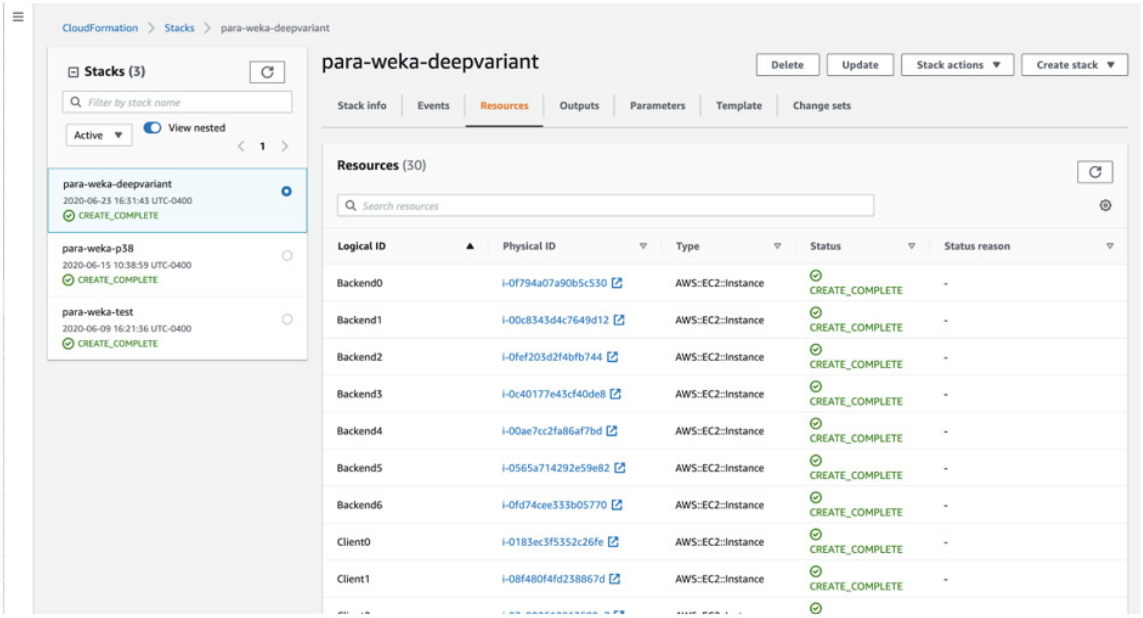
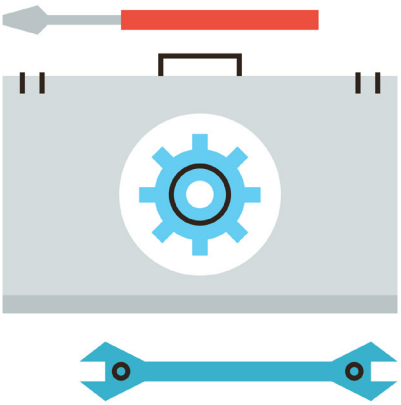


Figure 2: successful completion of the CloudFormation script on AWS

CHAPTER 3:
Configuring Weka



After the CloudFormation script has finished, click the "Outputs" tab on the CloudFormation page and open the first link in the list, as shown in Figure 3. This link opens the Weka GUI, which can be used to configure the file system, attach an object store, and view performance statistics. The default credentials for the Weka file system are "admin"/"admin". For security purposes, it is highly encouraged to change the password immediately. After logging in, you will be greeted with a system-level view of your cluster, including several performance metrics.

From here, open the "Object Stores" page using the menu on the left side of the screen to add the S3 object store, which was created with the start.weka.io scripts. Click the "+" sign next to Object Stores, then fill out the window with the necessary information for the S3 object store that you already created, and click the "Validate" button. If the validation succeeds, click "Configure" to configure the S3 bucket to be used with Weka.

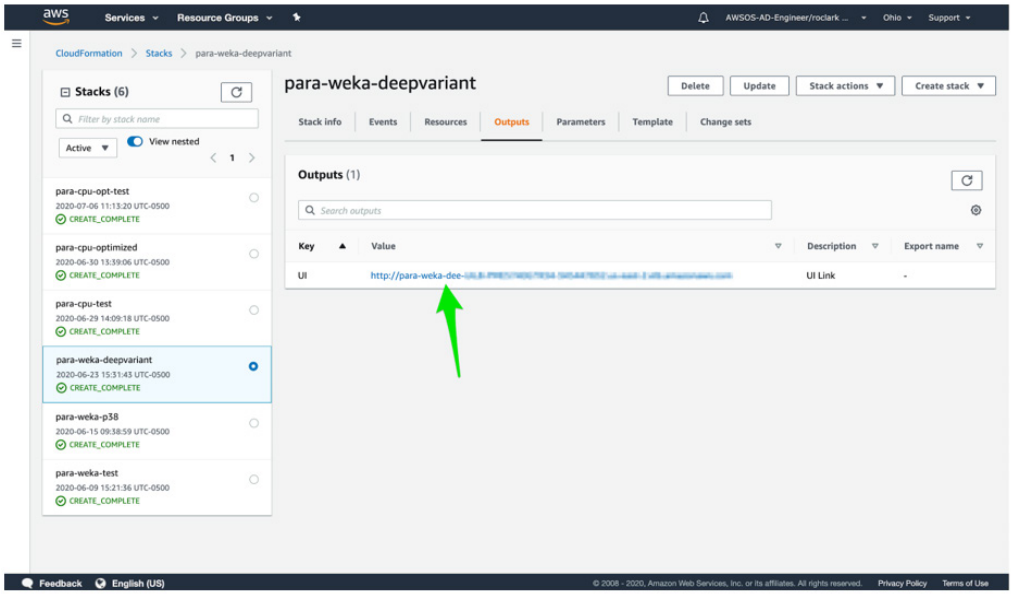


Figure 3: clicking the link in the Outputs tab opens up the WekaFS homepage

After the S3 object store has been configured, it needs to be attached to the default file system to be used as a backup for future work. To do so, navigate to the "Filesystems" page using the main menu. In the new page, click the "default" filesystem to display several new options. Click "Attach Object Store" and select the object store that was just added from the drop-down list and click "Attach."

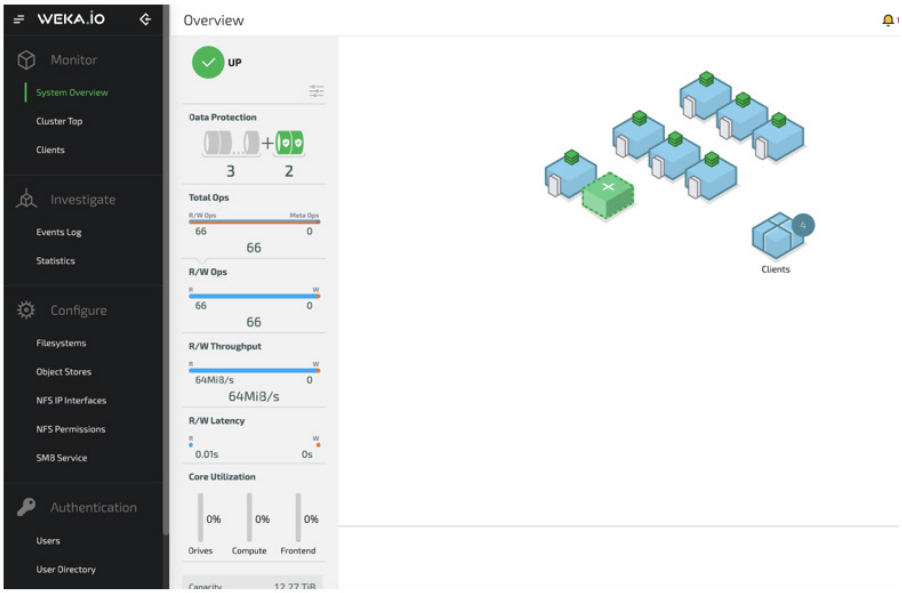


Figure 4: the default WekaFS homepage for the created cluster

Rehydrating File Systems

If a new cluster is created, an existing S3 object store can be used to pre-fill a file system with a copy of the dataset already stored in S3; this makes it easy to deploy new clusters as needed and ensure no data is lost during the creation and deletion processes.

To create a new file system using existing contents on an object store, check there is still free capacity to be used for other file systems, and click "New Filesystem" in the Filesystems page. Enter a name for the new file system, and check "Tiering" while selecting the object store to be used. Configure the capacities as desired, select "From Uploaded Snapshot," and enter the object store locator for the S3 bucket. After clicking "Create," a new file system will be created, pre-filled with the dataset from earlier runs. As our S3 object store is currently empty, however, we won't be able to do this on our first pass of the testing, but as long as the S3 bucket is intact, we can use it to accelerate all future deployments that rely on a common dataset.

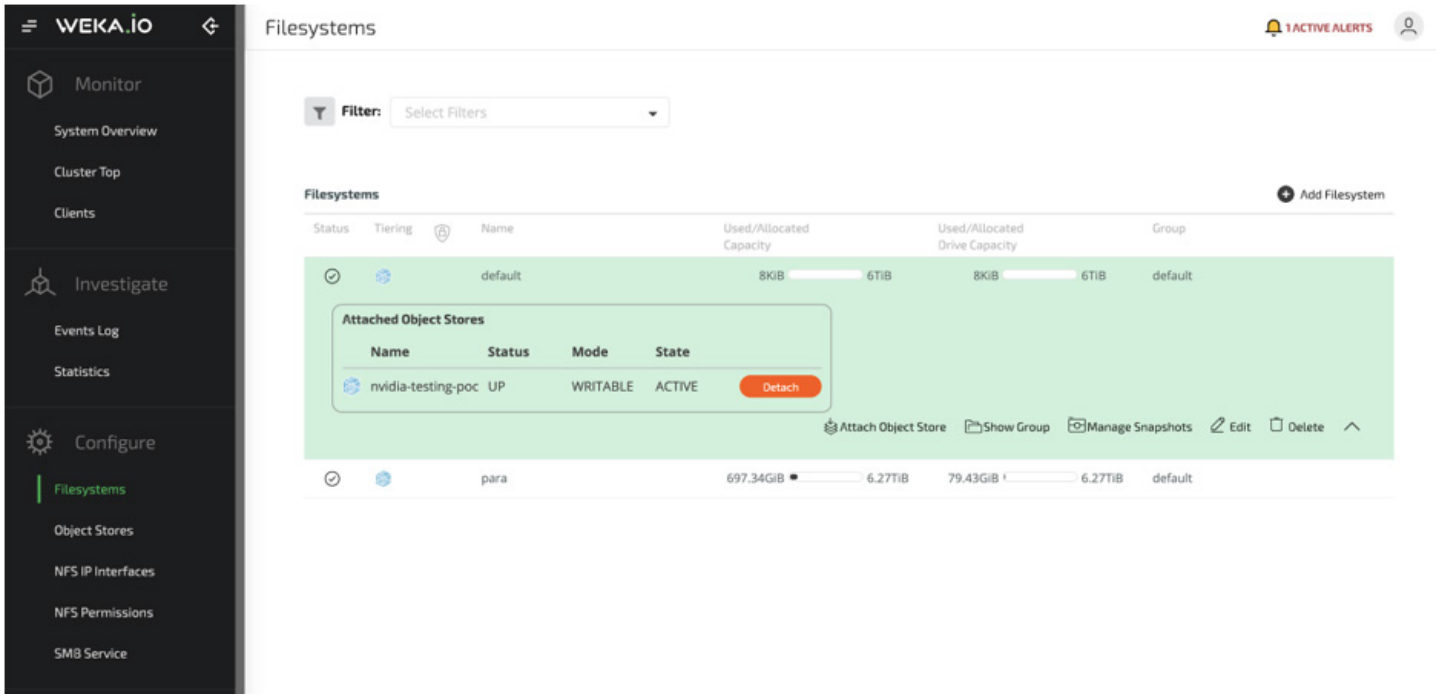


Figure 5: The default WekaFS homepage for the created cluster

CHAPTER 4:
Optimizing Performance

Before installing Parabricks and launching tests, several tweaks can be made to optimize the pipeline's performance and finish analyzing genomes more rapidly. These optimizations include tuning the GPUs for maximum performance and pre-fetching data from the object store to ensure data is stored in flash.

Tuning the GPUs

Amazon provides documentation on how to optimize GPUs for EC2 instances for maximum performance. For our selected p3.8xlarge instances, we will want to run the following steps:

1. Enable the GPU persistence daemon, which ensures the GPUs are always initialized, even when no clients are connected to the GPUs.

   ```
   sudo nvidia-persistenced
   ```

2. Set the maximum frequency for the memory and graphics clocks on the GPUs to improve throughput.

   ```
   sudo nvidia-smi -ac 877,1530
   ```

Pre-Fetching Data from Object Storage Tier

When the dataset has been replicated in an S3 bucket in AWS which offers an easy method to hydrate file systems and provides a layer of protection for the data by ensuring it is always backed up. While the S3 bucket is a great way to rehydrate storage and backup data, it is recommended to pre-fetch data from the object store so all data is loaded in flash for optimal throughput. Weka has a built-in tool that, combined with some standard Linux commands, will fetch all files from the object store and load them into the flash tier, which provides the fastest read and write speeds. To pre-fetch the data, run the following command **(replace <directory path> with the location of the Weka mount point):**

```
find -L <directory path> -type f | xargs -r -n512 -P64 weka fs tier fetch -v
```

As before, this will only matter for replicated runs when pre-loading a file system from an S3 bucket.

Newer versions of Weka (version 3.8.0 and above) include a function, `weka fs tier release`, which prompts the file system to update the object store with the latest data in flash on demand.

CHAPTER 5:
Using Parabricks

Now that a filesystem has been created and the clients have been optimized for maximum performance, it is time to install Parabricks and run the germline pipeline. Note that while results have been collected for both Parabricks version 2.5.3 and 3.0.0 for comparison's sake, only the steps to install version 3.0.0 are listed below. Additionally, a license will need to be requested from the Parabricks developer portal.

Installing Parabricks on GPU nodes

The latest version of the Parabricks software can be installed with a few simple commands. First, download the software on all p3.8xlarge instances using curl.

```
curl --output /tmp/parabricks.tar.gz --silent
"https://s3.amazonaws.com/parabricks.licenses/v3002_end_aug/parabricks.tar
.gz?AWSAccessKeyId=AKIAJGDUNN2G2ZAH3Q3A&Signature=9DN1bO2404uGeKwqqm8%2BI7
SV6yo%3D&Expires=1598030499"
```

Next, extract the package and run the installer.

```
cd /tmp/
tar -xvf parabricks.tar.gz
cd ./parabricks
sudo ./installer.py
```

The installer will display the EULA and give several prompts. Answer "yes" for all prompts. Once complete, Parabricks is installed and pipelines can finally be run. This can be verified by checking the version of pbrun installed:

```
$ pbrun version
pbrun: v3.0.0.2
```

Running the Germline Pipeline

The first pipeline to run is the commonly-used germline variant analysis pipeline which analyzes sequence data from a human genome or exome. To run the test, navigate to the Weka mount point on the filesystem and run the following:

```
cd /mnt/weka
pbrun germline --ref Ref/Homo_sapiens_assembly38.fasta --in-fq NA12878xCO_combined_R1.fastq.gz NA12878xCO_combined_R2.fastq.gz --knownSites Ref/Homo_sapiens_assembly38.known_indels.vcf --out-bam output.bam --out-variants output.vcf --out-recal-file report.txt
```

This command will take a little over an hour to complete as it goes through the various phases of the pipeline. Once complete, the output will display blocks of text indicating the total time to complete each phase, similar to the following:

```
-------------------------------------------------------------------------------
----

||        Program:                      GPU-BWA mem, Sorting Phase-I
||

||        Version:                                           v3.0.0
||

||        Start Time:                       Tue Jun 16 18:08:55 2020
||

||        End Time:                         Tue Jun 16 18:59:43 2020
||

||        Total Time:                          50 minutes 48 seconds
||

-------------------------------------------------------------------------------
----
```

Running the DeepVariant Pipeline

The next pipeline is the DeepVariant algorithm from Google which is another commonly-used tool for sequencing genomes, though it tends to be seen as too slow to run while still being practical. Luckily, with Parabricks running on NVIDIA GPUs, we don't have to wait very long as the algorithm will finish in just under 40 minutes on an aligned genome. To run DeepVariant, execute the following command from the file system:

```
pbrun deepvariant --ref Ref/Homo_sapiens_assembly38.fasta --in-bam output.bam --out-variants output.vcf
```

CHAPTER 6:
Results

Time to Completion

These results showcase the average time to complete the main phases of the germline and DeepVariant pipelines. The total time for the germline pipeline is the sum of the fq2bam alignment and HaplotypeCaller phases, while the DeepVariant pipeline equals the sum of fq2bam and DeepVariant. The fq2bam results in particular indicate how much of a performance improvement version 3.0.0 of the Parabricks software provides over version 2.5.3.

TEST PHASE	PARABRICKS VERSION	AVERAGE COMPLETION TIME
FQ2BAM	2.5.3	81 minutes, 29 seconds
	3.0.0	50 minutes, 51 seconds
HaplotypeCaller	2.5.3	16 minutes
	3.0.0	15 minutes, 36 seconds
DeepVariant	2.5.3	38 minutes, 18 seconds
	3.0.0	38 minutes, 31 seconds

File System Performance

These results display typical storage requirements for the pipelines as seen by the WekaIO file system. Given Weka's ability to handle large IO requirements, it is able to keep up with the Parabricks tests without breaking a sweat. The Weka file system still has plenty of headroom to handle additional clients sequencing genomes at the same time.

GERMLINE PHASE	PARABRICKS VERSION	MAX READ BW (GB/s)	MAX WRITE BW (GB/s)	MAX READ IOPS	MAX WRITE IOPS
FQ2BAM	**2.5.3**	**1.003**	**0.41**	**6,870 IO/s**	**4,236 IO/s**
	3.0.0	**1.525**	**0.231**	**21,327 IO/s**	**20,530 IO/s**
HaplotypeCaller	**2.5.3**	**1.984**	**0.567**	**16,496 IO/s**	**6,718.2 IO/s**
	3.0.0	**1.003**	**1.051**	**15,072 IO/s**	**85,213 IO/s**

Performance Compared to CPUs

While breaking down the completion time of the various phases of the pipelines running on both CPUs and NVIDIA GPUs, the dramatic performance improvements are readily apparent, as seen in the following table. Note that while Parabricks uses the fq2bam aligner, GATK uses BWA MEM for identical results.

TEST PHASE	CPU/GPU	AVERAGE COMPLETION TIME
FQ2BAM/BWA MEM	CPU	879 minutes, 58 seconds
	GPU	50 minutes, 51 seconds
HaplotypeCaller	CPU	1081 minutes, 58 seconds
	GPU	15 minutes, 36 seconds
DeepVariant	CPU	389 minutes, 19 seconds
	GPU	38 minutes, 31 seconds

Accuracy

While the performance numbers from Parabricks are certainly beyond impressive, it's arguably more impressive than the software is able to achieve these numbers with equivalent accuracy. The Parabricks implementation has greater than 99.9% identity compared to the CPU-optimized pipeline that I ran against the same dataset.

CONCLUSION

NVIDIA's Parabricks combined with the power of AWS and Weka provides numerous advantages over existing CPU-optimized solutions, saving time, money, and resources, all without sacrificing the accuracy of the results. Weka made it simple to expand from my local test environment to the cloud, all while keeping my data intact. With a few additional clicks in Weka's easy-to-use interface, I was able to rehydrate a new filesystem with data from my S3 bucket - a process that took only a few minutes for a few hundred gigabytes of data. This added an extra layer of security and simplicity to managing my dataset and filesystem in case I lost access to any components. Given the comparable performance between on-premises and cloud-based Weka filesystems, researchers can opt for the solution that best suits their needs.

For anyone looking to accelerate their genomics pipelines, NVIDIA Parabricks absolutely needs to be a part of their toolset as it is able to provide the same accuracy and analysis capabilities as the CPU-based GATK toolset, while dramatically reducing the time by as much as 33x and simplifying the steps required to analyze a complete genome.

910 E Hamilton Avenue, Suite 430, Campbell, CA 95008 T: 408.335.0085 E: info@weka.io www.weka.io

W03EB202010