NetApp®

Ripples Case Study

Conducted by: Nökkvi Dan Elliðason, Software Engineer Intern | nokkvi@netapp.com

Shaking things up with DAS on Wave Kubernetes

Innovating the landscape of real time seismic measuring with cloud solutions and distributed acoustic sensing in fiber optic cables

Our understanding of activity under the earth surface is largely limited by our ability to collect and process data. Up to now, seismic measurements have mainly been carried out with broadband seismometers, which can report the seismic activity with accuracy if they are conveniently located at any given time. Proper information of the movements of the geological plates is therefore entirely limited to the unrealistic idea of covering the earth's surface with these broadband seismometers, which is an extremely expensive task in terms of time and capital. These limitations can lead to natural disasters occurring with short notice without any knowledge of mankind. Considering these limitations of modern seismic sensors there has been a strong demand for innovation and progressive ways to increase humanity's knowledge and awareness of seismic activity.

In recent years, it has proved realistic to use Distributed Acoustic Sensing (DAS) measurements to gather information on seismic activity through fiber optic cables. With the ideology surrounding DAS technology, traditional fiber optic cables can be used to compile extensive datasets that are used to assess seismic activity through error calculations.

Data collection of this nature creates a continuous stream of data that belongs to the subject of big data. The complexity of gathering, storing, and utilizing these kinds of big data has so far been one of the main challenges associated with this type of measurement.

Conducted by: Nökkvi Dan Elliðason, Software Engineer Intern | nokkvi@netapp.com

DAS and its technology

The DAS sensor, traditionally called an "interrogator", is connected to one end of the fiber optic cable where it fires several hundred laser pulses per second through the fiber optic cable. When the laser pulse is launched into an optical fiber, a fraction of the light is elastically scattered (Rayleigh backscattered). These same DAS sensors can detect and collect the backscattered photons from the internal impurities in the core of the fiber. For any section of the fiber, the phase difference of photons scattered at both ends of the section is linearly related to the length of this section. When the section of the sensing fiber is unmoved, the length and consequently the phase difference remains unchanged. Thus, any external distress on the fiber will change that difference. This strain rate can therefore be mapped along the fiber by examining the changes in the phase of the elastically backscattered photons between successive measurements.

Data collection of this nature creates a continuous stream of data that belongs to the subject of big data. The complexity of gathering, storing, and utilizing these kinds of big data has so far been one of the main challenges associated with this type of measurement.

NetApp

NetApp is a leader in data warehousing and offers world-renowned cloud solutions in collaboration with many of the world's largest service providers, such as AWS, Azure and Google. NetApp is one of the most progressive companies in the world in terms of processing, storage, and utilization of large databases with data storage in the cloud. With advances in algorithms and the mastery of big data, it has now become possible to collect and utilize data streams for analytics and knowledge. These data stream acts as a continuously updating real-time database that is difficult to construct for provision of information. The Ripples case study aims to combine historical static data of seismic sensing to with a streaming application to provide enriched live data information.

NetApp’s expertise in Kubernetes makes this project feasible. Kubernetes automates deployment, scaling, and management of containerized applications such as the Ripples project. Ocean adds on top of that an optimally utilized and cost-efficient ways of automating these cloud infrastructures for the containers. The Wave environment automates the infrastructure provisioning and scaling for big data cloud environments, namely spark applications. Thus by running applications like these on Wave you eliminate manual infrastructure, save up to 90% of infrastructure costs and enjoy rich observability into resource utilization.

Conducted by: Nökkvi Dan Elliðason, Software Engineer Intern | nokkvi@netapp.com

The Ripples case study

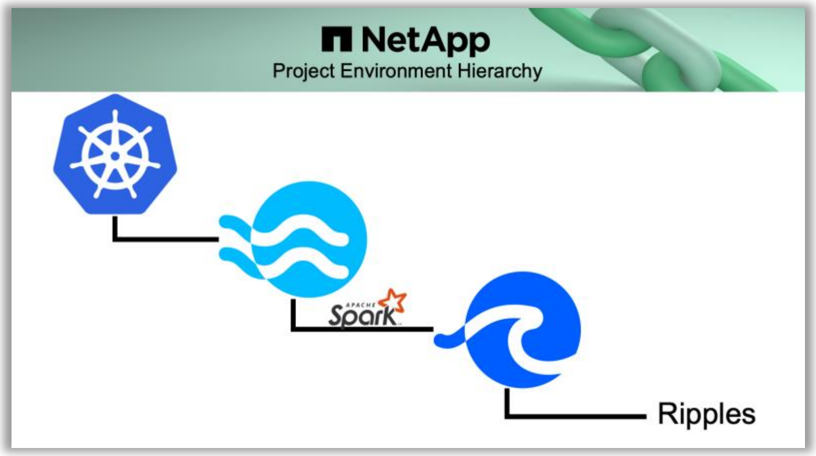
The idea behind the implementation of a project of this type is the usage of the Apache Spark programming environment, where NetApp is well-versed. The project in its core, is creating a Spark Streaming App (or App Suite) which can run on many systems and is ideal for NetApp's in solutions. The benefits of a project like this can lead to a much more efficient spatial information of seismic activities and thus optimization of knowledge. There are as well countless benefits from transferring information and data from standard computing modules databases to the Spark stream. Furthermore, NetApp software's, Ocean, and Wave, create an efficient and cost saving cloud environment to manage these extensive data sets in an almost futuristic way.

NetApp looks at the Ripples case study as a service to the community in which it thrives, as well as enhancing the image of the company. With NetApp's dominant market position in big data and cloud solutions, Iceland's unique location and the Icelandic Meteorological Office's expertise, a project like this will have meaningful benefits.

Preface of deployment

The Ripples case study uses many programming languages and environments. The core of the case study is to run on Kubernetes with the infrastructural management of Wave. To be able to run that process, the reader needs to have the right environment setup locally. The reader needs a Wave cluster, Apache Spark, Java, Docker, Command line tools (kubectl & spotctl) and AWS account connected to Spot. The entire list of preface requirements can be found here below and instructions on how to install these programs and environments can be found by clicking here https://docs.spot.io/wave/getting-started/.

The hierarchy of the environment in which the project is conducted can be seen here:



Conducted by: Nökkvi Dan Elliðason, Software Engineer Intern | nokkvi@netapp.com

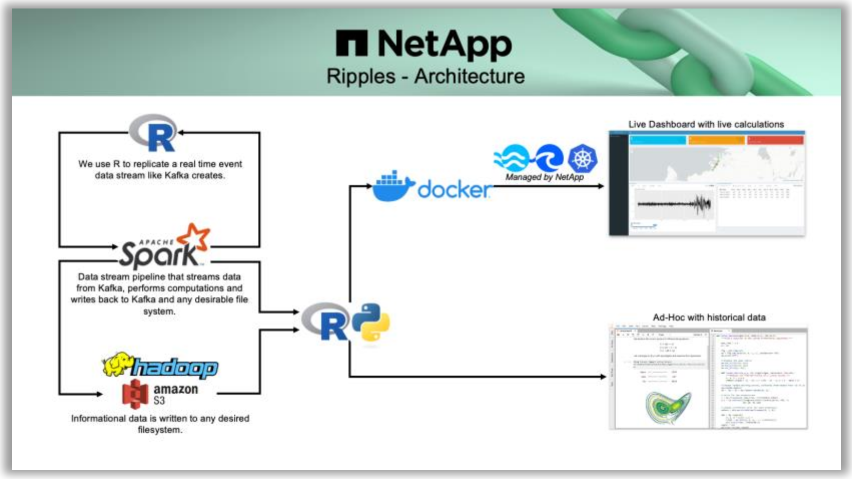
When you have finished downloading and setting up the Wave environment the one level higher environment is also needed to run both locally and in the cloud. To make sure that you can run both locally and on Kubernetes you should download each program and/or file listed here below:

- **Java openJDK:** 1.8.0_265
- **R:** >4.0.3
- **Apache Spark:** 3.1.2
- **Jar files:** hadoop-aws-3.2.0.jar, aws-java-sdk-bundle-1.11.874.jar, jets3t-0.9.4.jar

Note: you need to move the jar files needed to the SPARK_HOME/jars directory.

Deploying the application

Along with this section it can be helpful to watch a video where the creator of the application, Nökkvi Dan Elliðason, goes through the process of deploying the application. This video can be found here . Getting a good sense of the architecture will accommodate this part of the article and will as well make the deploying process more pleasant. The architecture of the Ripples project which can be seen here below:


NetApp
Ripples - Architecture
We use R to replicate a real time event data stream like Kafka creates.
Spark
Data stream pipeline that streams data from Kafka, performs computations and writes back to Kafka and any desirable file system.
hadoop
amazon S3
Informational data is written to any desired filesystem.
docker
Managed by NetApp
Live Dashboard with live calculations
Ad-Hoc with historical data

Cloning the repository:
Now you should navigate too https://github.com/NetApp/ripples and clone the repository. Too check if you have everything required to run the application you can run the process in `docker_wave_r` directory which can be found in the repository.

Dashboard configurations:
Open the `dashboard_s3.R` with RStudio or other IDE of choice (or the terminal if you are feeling fancy). There you want to configure the file with you AWS credentials and bucket of choice. This opens a connection to your AWS account, namely the S3 bucket we want to read and write too. Now you should be ready to move on, in the deploying process.

Conducted by: Nökkvi Dan Elliðason, Software Engineer Intern | nokkvi@netapp.com

Docker
Docker takes away repetitive, mundane configuration tasks and makes the for fast, easy portable application development. This makes our process of deploying the Ripples project reproducible. The reader should create an account on Docker hub and create a repository there before going further.

Now you need to configure the Docker file in the GitHub repository which you have locally to match your credentials and preferences. The `Dockerfile` in the repository is a text document that contains all the needer commands to call on the command line to assemble an image. Using *docker build* then creates an automated build that executes several command-line instructions in succession.

- Change the Docker file tag with your credentials, for mine it would be:
    - *username/repository:TAG ---> nokkvidan/ripplesrepo:1*
- Build the docker image with your credentials, for me it would be:
    - *docker build . -t nokkvidan/ripplesrepo:1*
- Push the image to docker hub, for me it would be:
    - *docker push nokkvidan/ripplesrepo:1*

Spark-Submit
Now change the `spark-submit file with *vim spark-submit*. The spark-submit script in the directory is used to launch applications on a cluster. It can use all of Spark's supported cluster managers through a uniform interface, we use Kubernetes.

- Here you need to change the *--conf spark.kubernetes.container.image* with your credentials, for me it would be:
    - *--conf spark.kubernetes.container.image=nokkvidan/ripplesrepo:1*

Start stream
This section creates a DAS data stream which is written to the AWS S3 bucket of your choice. We only have one sample location of DAS data and thus we create 4 scaled down amplitudes of the sample thread. Thus, we can show the dashboard with 5 locations of DAS data along 1 fiber optic cable running through Iceland. The stream will run with 180 measurements every 10 seconds. This is approximately the same as real time which gives around 18 measurements each second. For quite a while nothing interesting happens so if reader wants some action configure as the comments in the scripts say.

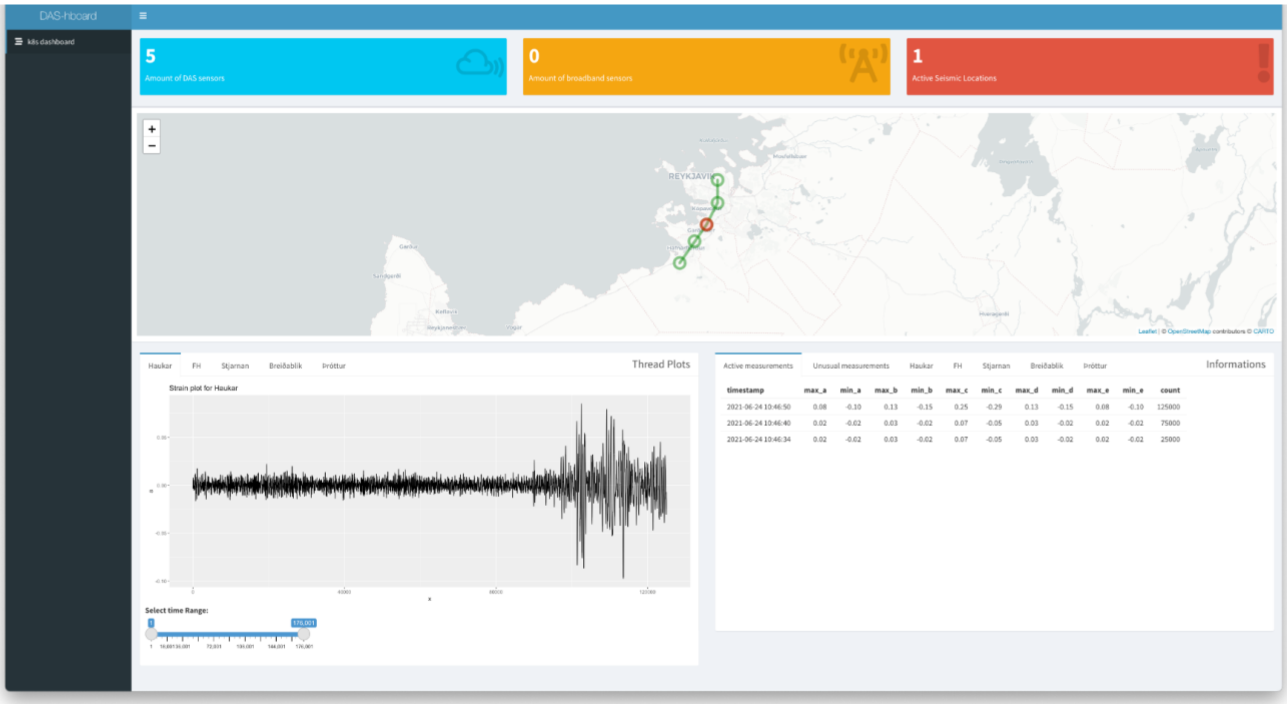
Now open the `stream_sampler_s3.R` and add your AWS access keys and regions. You can now run the stream locally which writes to S3. Running the stream can be done with both command line and RStudio (recommended). To run it with RStudio you open the `stream_sampler_s3.R` and simply run the while loop script. You may need to set the `DAS_K8s` folder as working directory.

Ripples Case Study

Conducted by: Nökkvi Dan Elliðason, Software Engineer Intern | nokkvi@netapp.com

Run dashboard
Now to the exportation of the application to your Kubernetes-Wave cluster. This exportation of the dashboard gives you a live dashboard of the DAS data being read in real-time. The R script that creates this is a single script which creates the UI of the dashboard and backend of SparkR at the same time. In more detail the Rscript creates an R environment to reach Spark, read from S3 into Spark, execute ETL commands and creates the live shiny dashboard.

- Go into this directory through the terminal
- Run the spark-submit shell: `sh spark-submit`
- Check if the pod is being created or is still running: `kubectl get pods --all-namespaces`
- View the logs of the spark-job: `kubectl -n spark-jobs logs <name of pod>`
    - Here you should see where it's listening "Listening on http://0.0.0.0:5050"
- port-forward the output: `kubectl -n spark-jobs port-forward <name of pod> 8090:5050`
    - Now you should be able to open the dashboard at http://localhost:8090



timestamp	max_a	min_a	max_b	min_b	max_c	min_c	max_d	min_d	max_e	min_e	count
2021-06-24 10:46:50	0.08	-0.10	0.13	-0.15	0.25	-0.29	0.13	-0.15	0.08	-0.10	125000
2021-06-24 10:46:40	0.02	-0.02	0.03	-0.02	0.07	-0.05	0.03	-0.02	0.02	-0.02	75000
2021-06-24 10:46:34	0.02	-0.02	0.03	-0.02	0.07	-0.05	0.03	-0.02	0.02	-0.02	25000