DATA ANALYSIS PLATFORM FOR STREAM AND BATCH DATA PROCESSING ON HYBRID COMPUTING RESOURCES

S. Belov[1 2], I. Kadochnikov[1,2,a], V. Korenkov[1,2], A. Reshetnikov[1,2] R. Semenov[1,2], P. Zrelov[1,2]

[1] *Joint Institute for Nuclear Research, 6 Joliot-Curie st., Dubna, 141980, Russia*

[2] *Plekhanov Russian University of Economics, 36 Stremyanny lane, Moscow, 117997, Russia*

E-mail: [a] kadivas@jinr.ru

The modern Big Data ecosystem provides tools to build a flexible platform for processing data streams and batch datasets. Supporting both the functioning of modern giant particle physics experiments and the services necessary for the work of many individual physics researchers results in generating and transferring large amounts of semi-structured data. Thus, it is promising to apply cutting-edge technologies to study these data flows and make the services' provisioning more effective. In this work, we describe the structure and implementation of our data analysis platform, built on the Apache Spark cluster. With the official support for GPU computing now available in Spark version 3, we propose a change in the architecture to utilize these more performant resources while keeping the platform's functionality provided by using mainstream Big Data software. Furthermore, the necessity for GPU support entails a change in the computing resource management infrastructure from Apache Mesos to Kubernetes. Finally, to demonstrate the features and operation of the system, we use the task of network packet analysis for security monitoring and anomaly detection in both batch and stream modes.

Keywords: big data, GPU computing, stream processing, containers, machine learning

Sergey Belov, Ivan Kadochnikov, Vladimir Korenkov, Andrey Reshetnikov, Roman Semenov, Petr Zrelov

1. Introduction

High-energy physics experiments, such as those being conducted at the Large Hadron Collider (LHC) at CERN and will be conducted at the Nuclotron-based Ion Collider fAcility (NICA) at JINR, produce actual experimental data at the scale of terabytes per second [1]–[3]. This data is usually processed and analyzed using specialized libraries on dedicated computing platforms [4]. In addition, modern large experiments and institutions generate many streams of ancillary data that plays a critical role in supporting their operations. This information has an immediate technical purpose, but it can also be collected for a more thorough cross-referential analysis.

Projects in the Big Data ecosystem provide robust and scalable software to build a platform for collecting and processing such datasets. A prototype of such a platform was proposed and implemented in [5]. This work aims to build on the given progress by implementing support for GPU computing resources. The speedup that GPU processing ensures for different processing and analysis operations can be then measured for more effective scale-out and task scheduling in the future. We expect GPUs to be especially effective for accelerating the training of machine learning models built with deep artificial neural networks.

2. Platform architecture

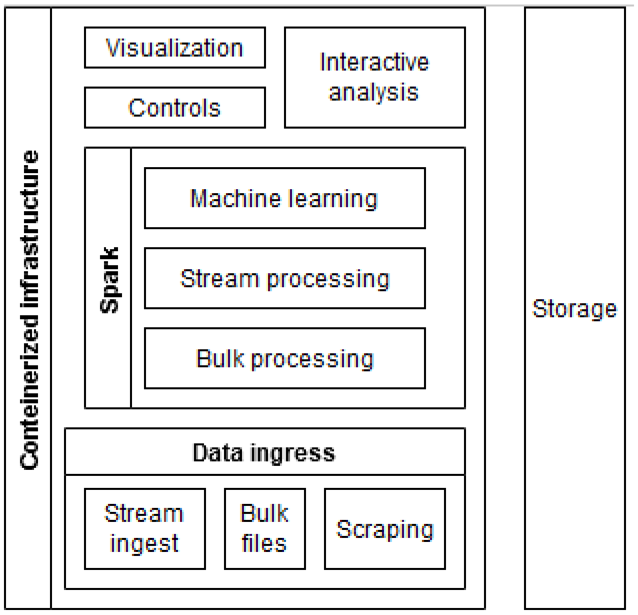

Figure 9. General platform structure and functionality

2.1 Big Data and Apache Spark

The core processing and analysis framework of the platform is Apache Spark, which facilitates batch and stream processing, contains machine learning libraries, and can interface with many data management and storage tools in the Big Data ecosystem.

Distributed storage is provided within the platform by the MooseFS file system. This does not give the performance benefits of data locality afforded by HDFS, which stores data directly on compute nodes. However, data locality is reported to be less essential for modern Big Data platforms than it was at the inception of the Hadoop ecosystem [6].

2.2 Resource management in the Spark cluster

The Spark cluster can be run standalone, or use a resource manager: YARN, Mesos or Kubernetes. Mesos was used as the resource management tool of the prototype framework in [5], however, Spark does not yet support GPU resource management with Mesos. Kubernetes was selected as the resource manager for the future, as it allowed consolidating the management of the computing resources and the containerization of platform services.

Running Spark in the standalone cluster mode is straightforward, but it is less flexible and less desirable in a production environment than the other modes. YARN is a resource manager specialized for the Big Data ecosystem; it would be preferable if we had an established Hadoop-based platform to add Spark onto.

2.3 GPU resource support in Spark

To use NVIDIA GPUs as resources in Spark jobs running on the Kubernetes cluster, the underlying containers need to support NVIDIA hardware. The libraries and tools provided by NVIDIA for this support are multi-layered [7]:

- libnvidia-container provides an API and CLI to set up containers with NVIDIA GPU support
- nvidia-container-toolkit provides a runC prestart hook to apply these compatibility tweaks on container startup
- nvidia-container-runtime wraps runC, adding this prestart hook to any container config started through this wrapper
- nvidia-docker2 installs the runtime into the local Docker configuration, allowing to start GPU-enabled containers more easily

The actual need for these tools and the compatibility between Kubernetes and the NVIDIA driver version is not very well-documented. Kubernetes suggests using k8s-device-plugin, which purportedly requires a specific NVIDIA driver version (384.81) [8], [9]. NVIDIA themselves provide more up-to-date and complete documentation on installing a Kubernetes cluster with GPU support [10].

We used the NVIDIA DeepOps approach suggested by that article. It provides an easy way to deploy and configure most of the Kubernetes components needed to run a production cluster by building on top Kubespray for Kubernetes deployment with Ansible [11]. The specific up-to-date procedure allowed us to quickly deploy and test the cluster for our platform, but it can create support and configuration issues in the future when we might want to deviate from the suggested cluster architecture.

An important aspect of managing GPU resources with Kubernetes for Spark is resource discovery. That is, finding and annotating the GPUs present on the Kubernetes node to direct specific Spark jobs to utilize such resources. DeepOps configured the containerized service for resource discovery by default.

3. Platform testing

3.1 Distributed Tensorflow machine learning

To test GPU resource support on the platform, a sample distributed machine-learning job was run on the platform. Spark-tensorflow-distributor [12] provides a method for distributing Tensorflow workflows across the Spark cluster, basically utilizing Spark as a resource and workload manager. It also provides a sample script to demonstrate the training of a small convolutional neural network for the classic problem of handwritten digit classification on the standard MNIST dataset. As Tensorflow can run with or without a GPU, the same test script was used for testing throughout this project to ensure that the GPU virtualization of the NVIDIA T4 GPU worked with the standalone Spark node, that the distributor library worked correctly on the Kubernetes cluster, and that GPU resources were available to Spark through the Kubernetes node.

3.1 Network packet analysis

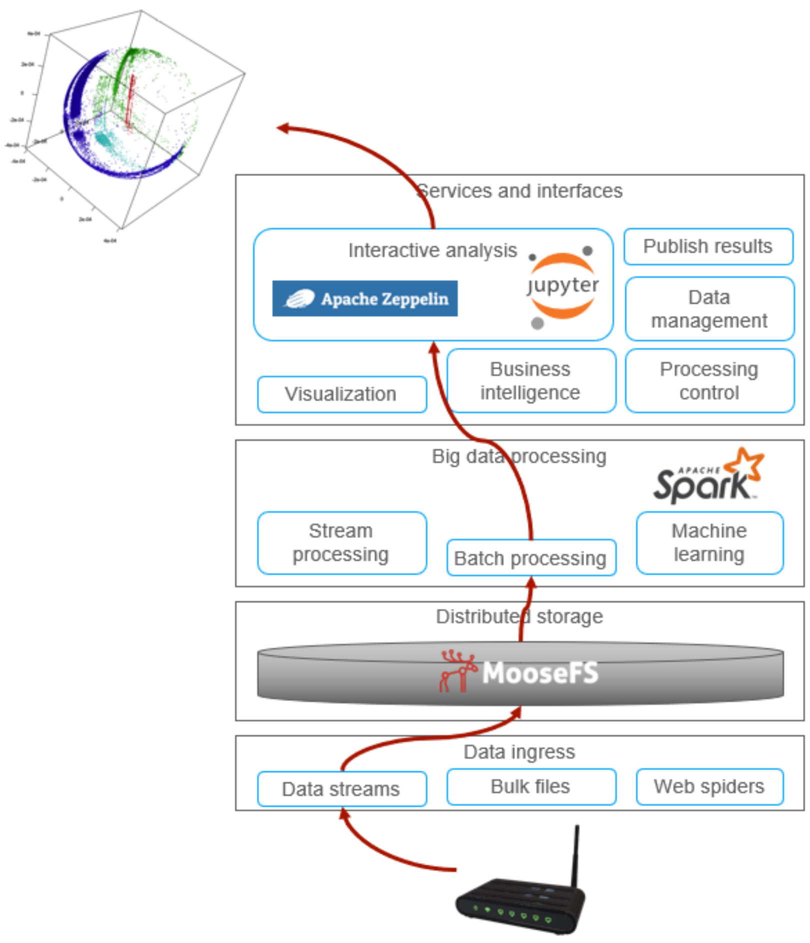

Figure 10. Data flow through the platform in the network packet analysis problem.

To demonstrate a more practical application of the platform at a scale closer to Big Data, a prototype pipeline to collect and process network packets was implemented on the new framework, in an approach similar to the way the same problem was solved on the prototype framework [5].

Network packets from one local laboratory subnetwork were duplicated and sent to one physical machine that did not take part in the main Kubernetes cluster. Raw packet headers were extracted with tshark[13] running in a Docker container and dumped into temporary 10Mb files continuously. They were parsed by running 7 tshark instances with GNU parallel[14] in another container, with the communication of files to be parsed managed by incrontab and a named FIFO pipe. The parsed JSON files were immediately compressed into ~14Mb zstd archives and stored on the distributed storage. This collection step ran for continuous capture for 7 days, resulting in a parsed dataset of 700Gb ready for analysis.

The analysis was carried out in Spark, with the processing steps submitted from the Zeppelin notebook-style web interface running entirely within the same Kubernetes resource cluster that runs the resulting Spark jobs. Thanks to building our own Docker images hosted on a private Gitlab repository, mounting the distributed storage into the Zeppelin and worker nodes, version compatibility, and adding support for reading zstd-compressed files were minor problems.

The analysis consisted of using the Numeric Aggregate and Mode (NAGM) method to aggregate and extract network node features from the network packet TCP and IP headers. This method, specifically for Darknet packet analysis, is described in detail in [15]. We apply it to normal network packets to test the performance of the framework and engineer network node features with an existing established method for further analysis. The same approach was used to test the prototype Big Data framework in [5].

4. Conclusion

Most of the platform changes from the 2020 prototype to the current state were motivated by the inclusion of GPU resources, which necessitated the change of the resource manager, the update of Apache Spark, the use of Ansible for initial deployment. A list of changes and the motivation for them are presented in Table 3.

Table 3. Framework components modified from the prototype to today

	2020 prototype	**2021 framework**	**Motivation**
Resources	CPU nodes	CPU nodes + Nvidia T4	GPU
Analysis core	Spark 2.4	Spark 3.1	GPU
Container repository	DockerHub	Gitlab.com	Performance
Resource management	Mesos	Kubernetes	GPU
Configuration management	None with plans for Puppet	Ansible with plans for Puppet	GPU
Supporting services	Docker swarm	Kubernetes	The Kubernetes cluster is already set up for Spark
Coordination	Zookeeper	etcd	Deepops default
Authentication	None	FreeIPA in progress	Auth is important for the platform

5. Acknowledgement

The study was carried out at the expense of the Russian Science Foundation grant (project No. 19-71-30008).

References

[1] G. Bauer *et al.*, "The data-acquisition system of the CMS experiment at the LHC," *J. Phys. Conf. Ser.*, vol. 331, no. 2, p. 022021, Dec. 2011, doi: 10.1088/1742-6596/331/2/022021.

[2] J. G. Panduro Vazquez, "The ATLAS Data Acquisition System in LHC Run 2," *J. Phys. Conf. Ser.*, vol. 898, p. 032017, Oct. 2017, doi: 10.1088/1742-6596/898/3/032017.

[3] V. D. Kekelidze, "NICA project at JINR: status and prospects," *J. Instrum.*, vol. 12, no. 06, pp. C06012–C06012, Jun. 2017, doi: 10.1088/1748-0221/12/06/C06012.

[4] M. Lamanna, "The LHC computing grid project at CERN," *Nucl. Instrum. Methods Phys. Res. Sect. Accel. Spectrometers Detect. Assoc. Equip.*, vol. 534, no. 1, pp. 1–6, Nov. 2004, doi: 10.1016/j.nima.2004.07.049.

[5] S. Belov, I. Kadochnikov, V. Korenkov, R. Semenov, and P. Zrelov, "Batch and Stream Big Data Processing Platform: Case of Network Traffic Analysis," in *Proceedings of the Big data analysis*

tasks on the supercomputer GOVORUN Workshop, Dubna, Russia, Sep. 2020, vol. 2772, pp. 52–57. Accessed: Sep. 30, 2021. [Online]. Available: http://ceur-ws.org/Vol-2772/#52-57-paper-8

[6] "What about locality?," *Red Hat Storage*, Jul. 11, 2018. https://redhatstorage.redhat.com/2018/07/11/what-about-locality/ (accessed Mar. 19, 2019).

[7] "What's the difference between the lastest nvidia-docker and nvidia container runtime? · Issue #1268 · NVIDIA/nvidia-docker," *GitHub*. https://github.com/NVIDIA/nvidia-docker/issues/1268 (accessed Sep. 16, 2021).

[8] "Schedule GPUs," *Kubernetes*. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/ (accessed Sep. 16, 2021).

[9] *NVIDIA device plugin for Kubernetes*. NVIDIA Corporation, 2021. Accessed: Sep. 16, 2021. [Online]. Available: https://github.com/NVIDIA/k8s-device-plugin

[10] "Install Kubernetes — NVIDIA Cloud Native Technologies documentation." https://docs.nvidia.com/datacenter/cloud-native/kubernetes/install-k8s.html (accessed Sep. 30, 2021).

[11] "deepops/docs at master · NVIDIA/deepops," *GitHub*. https://github.com/NVIDIA/deepops (accessed Sep. 16, 2021).

[12] "ecosystem/spark/spark-tensorflow-distributor at master · tensorflow/ecosystem," *GitHub*. https://github.com/tensorflow/ecosystem (accessed Sep. 30, 2021).

[13] "Wireshark · Go Deep." https://www.wireshark.org/ (accessed Sep. 30, 2021).

[14] O. Tange, *Gnu Parallel 2018*. Zenodo, 2018. doi: 10.5281/ZENODO.1146014.

[15] R. Niranjana, V. A. Kumar, and S. Sheen, "Darknet Traffic Analysis and Classification Using Numerical AGM and Mean Shift Clustering Algorithm," *SN Comput. Sci.*, vol. 1, no. 1, p. 16, Aug. 2019, doi: 10.1007/s42979-019-0016-x.