ELK Stack – Improving the Computing Clusters at DFCTI Through Log Analysis

Robert Poenaru  
DFCTI  
NIPNE-HHMagurele, Romania  
robert.poenaru@protonmail.chDragos Ciobanu-Zabet  
DFCTI  
NIPNE-HHMagurele, Romania  
zdragos@nipne.ro

*Abstract*—The full stack logging service provided by ElasticTM has become a powerful tool within the high-performance computing community due to its ease of use, lightweight impact on the machines, response time, and scalability. In the current work, we attempt to deploy such a stack on a server within our department, which will be used for ingesting, parsing, and finally analyzing logs coming from multiple clusters. By analyzing the overall performance of each machine that is under continuous monitoring, we can provide immediate support in case of any issues that might occur, and more importantly, we can improve the computing power of our clusters through optimizations in terms of system management, networking, and other specific features.

Keywords— Elasticsearch, Kibana, Logstash, pipelines, logs, metrics, clusters, compute notes, Kubernetes.

# Introduction

Extending the computational resources in terms of raw compute power, storage space, memory bandwidth, and network capabilities is a top priority for any research institute nowadays. Recently, the Department of Computational Physics and Information Technology at the National Institute of Nuclear Physics and Engineering – Horia Hulubei (situated in Magurele, Romania) is planning a big upgrade in terms of the computing equipment that will be used for system administration, to improve the network infrastructure within the institute center, and also to provide the computational resources required for the research teams and their projects.

With the increase in size of the computing resources comes a challenge in having a proper system administration and making sure that every component is working properly, well within its performance margins. Having a large number of CPUs does not necessarily scale up the performance of a compute cluster without proper optimizations; accessing large amount of data from a storage unit could also result in potential issues if the configuration is not done properly. Problems like these could appear during the early development stages of the infrastructure itself, or even after its final deployment. In order to minimize this kind of problems, it is crucial that a constant real-time analysis and monitorization of the system metrics should be performed by the system administration team. Thus, having a great log management tool, that is capable of ingesting logs from a multitude of sources, convert them to in a human-readable format, analyze them in real-time, and also store them for later use (if necessary) is essential within our department. Fortunately, there is such a toolset available to us: Elasticsearch [1]. The team at Elastic developed a very useful instrument which became very popular over the years, mainly due to the fact that it provides the entire stack for monitoring a computing resource: log shipping, log ingesting, log parsing, log storing and finally analysis – ELK stack [2].

We attempt to build and configure a full Elasticsearch logging service (ELK stack) that will be used within the department for analyzing logs from a multitude of compute clusters. In this way, the team will be able to check the status of the machines that run simulations (or, in other words, compute jobs) and see if there are any issues that require immediate fix. Not only that, but by having an insight on the resource performances with time, one could pinpoint certain patterns preventing issues or even deploy optimizations throughout the system. Analyzing the overall performance for the compute nodes which run large simulations could also be useful in helping the research teams that develop the actual simulations: through allocating the available resources in such a way that great scalability is achieved with the increase in numerical algorithm’s complexity. All these aspects act as a motivation for adopting a service which will be able to provide us a tool capable of improving the computing architecture of the department (DFCTI) and later on for the entire research institute, that is NIPNE-HH.

In the present work, an ELK stack is developed as a testing phase for what will eventually be a proper log analysis service inside DFCTI (and further, to the entire institute). The stack is deployed on a virtual machine from one of the available servers inside the department. In order to see how the log ingesting performs in terms of dropped connections, network lag (which induces a delay between each logging event), or even issues related to the local VM, we sent metrics from different sources, each requiring individual parsing. One of the sources was a Kubernetes (k8s for short) cluster [3] managed by the OpenStack service [4]. This has a major importance, as the cluster will be part of an incoming project for extending the cloud infrastructure that is available at DFCTI [5]. Within the testing phase of the stack, we noticed an overall great performance of the ingesting pipeline and more importantly, the data parsing. The data parsing that is made through Logstash was properly configured for all the sources (e.g. system metrics from a personal computer, logs from a SLURM workload manager [6] and the k8s compute cluster), and we were able to extract the necessary information from parsed logs, which is planning for fixing and improving the infrastructure. Elasticsearch has been proven to be a great tool for storing the incoming logs, which can be accessed at a later time through the available REST API. Visualization of logs was also successful by using the Kibana UI. The stored logs (in Elasticsearch indices) and the Kibana interface were deployed on the web for remote access via browser.

In this section we gave the motivation behind constructing an ELK stack. In section II, a detailed workflow of the resources which will be monitored is given. Furthermore, in the third section we aim at discussing each service that was used for the project (with a few arguments on the choices we made). **Finish**

# Compute Resources & Log Sources