Automatic Monitoring of Large-Scale Computing Infrastructure

Bockjoo Kim* and *Dimitri* Bourilkov

Department of Physics, University of Florida, Gainesville, FL 32611, U.S.A.

Abstract. Modern distributed computing systems produce large amounts of monitoring data. For these systems to operate smoothly, under-performing or failing components must be identified quickly, and preferably automatically, enabling the system managers to react accordingly. In this contribution, we analyze jobs and transfer data collected in the running of the LHC computing infrastructure. The monitoring data is harvested from the Elasticsearch database and converted to formats suitable for further processing. Based on various machine and deep learning techniques, we develop automatic tools for continuous monitoring of the health of the underlying systems. Our initial implementation is based on publicly available deep learning tools, PyTorch or TensorFlow packages, running on state-of-the-art GPU systems.

1 Introduction

LHC [1] experiments produce a huge amount of data daily whether they are coming from the Proton-Proton collisions at CERN or the Monte Carlo (MC) simulations. CMS [2], one of the LHC experiments, distributes data produced at the LHC to various sites called Tier1, Tier2, or Tier3 sites. The collision data produced at CERN and the samples produced by the Monte Carlo simulation at CMS Tier2 sites need to be replicated and transferred to various sites. To move data and the MC samples around various sites, CMS sites provide data read and write service using XRootD [3]. In addition to the transfers of the data and the samples between XRootD servers, the stored data and the samples in each site are read by the MC production jobs and the physics analysis jobs.

Particularly, XRootD servers in the data centers are the backbone of the sample transfer system as well as the data access by the MC production jobs. The physics analysis jobs are producing its own monitoring data constantly. The monitoring data for the data and the sample transfers are accumulated through one of the data and the sample transfer systems called Rucio [4]. On the other hand, the monitoring data for the data and the sample access by various jobs are needed to be monitored through the specially designed monitoring system called the XRootD Monitoring Collector [5][6].

The XRootD Monitoring Collector is ingesting monitoring data from the XRootD server, aggregating it into one monitoring record per file read, and sending a resulting JSON-formatted record into an AMQP-based message bus. The XRootD Monitoring Shoveler [7]

* Corresponding author: bockjoo@phys.ufl.edu

is designed to accept the XRootD monitoring packets at each site and shovel them to the OSG [8] message bus in the XRootD Monitoring Collector using a UDP stream. The XRootD Monitoring Collector and the XRootD Monitoring Shoveler are developed by the OSG XRootD monitoring team. The monitoring data used in this article are further sent to the Elasticsearch database at CERN and this article mainly use the monitoring data produced at sites with the XRootD Monitoring Shoveler.

For the XRootD system to operate smoothly, underperforming or failing components must be identified quickly, and preferably automatically, enabling the system managers to react accordingly. To accomplish this, we can use the XRootD monitoring data to be processed by a deep machine learning technique for the automatic monitoring of the health of the XRootD system at sites.

Machine learning techniques are ubiquitous. The XRootD monitoring data that are collected have a few typical features that fit best for certain machine learning techniques. Since the XRootD monitoring data are recorded in sequence, one can consider the analysis of the monitoring data with the time series technique. However, the monitoring data is not always fed into the monitoring system in order of the time as the time to read XRootD data is different depending on the size of the files read through the XRootD. As a result, one can consider a feature-based deep learning technique.

The article will describe the sample collection in Section 2. In Section 3, we describe how we selected features used in the study. In Section 4, we describe how we used the deep learning technique to check the health of the XRootD site automatically. In Section 5, we present the results of the study. Lastly, we present plans for future studies related to this study.

2 Sample Collection

The MC production and the physics analysis jobs are running all over the CMS Tier2 sites. These jobs are accessing the data at the CMS Tier2 sites either through read activities or write activities. These activities create traces of bytes in and out of the XRootD servers.

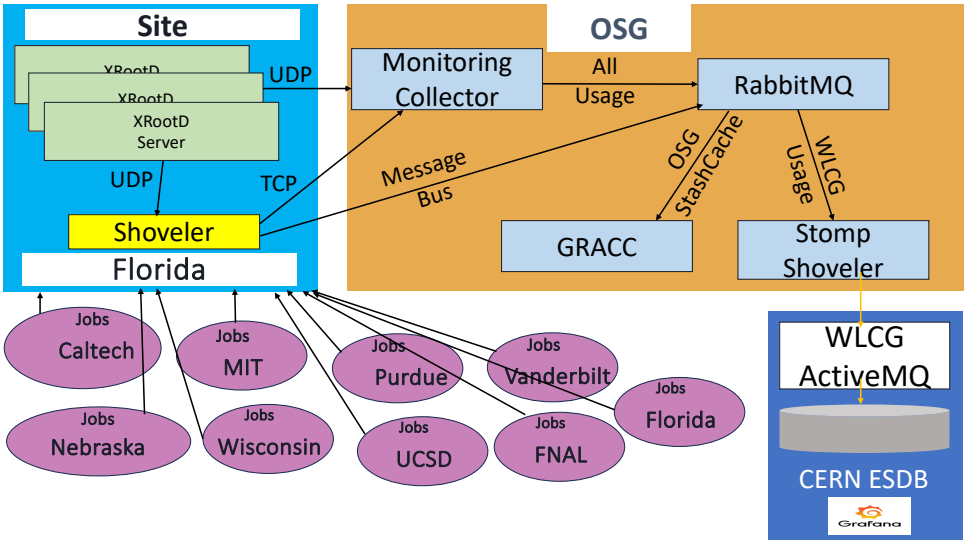

Figure 1 Sample Collection Process

The XRootD Monitoring Shoveler at a site can keep track of these activities. The monitoring data that went through the XRootD Monitoring Shoveler are sent to the XRootD

Monitoring Collector at OSG using the UDP stream instead of TCP to prevent loss of packets. The collected XRootD monitoring data are eventually stored in an Elasticsearch database at CERN. Since there are CMS sites that are not in the U.S., the final destination of the monitoring data is the Elasticsearch database at CERN. Typically, the Elasticsearch database can be accessed through the Grafana monitoring system. The whole sample collection process is depicted in .

For the analysis of the monitoring data, the XRootD monitoring data stored in the Grafana monitoring system is accessed using a python script. The python script extracts the XRootD monitoring data in the JSON format.

At the moment, there are only US sites with the XRootD Monitoring Shoveler. As a result, there is only limited access to the monitoring data. We decided to study the monitoring for a single site where the data read activities are monitored. Consequently, we have implemented the sample collection script to collect the monitoring data for all sites. An example output in the JSON format downloaded from the Grafana server is shown in Figure .


```
{
  "site_name": "ESNET-LBNL59-CMS-XCACHE",
  "client_host": "transfer-2.ultralight.org",
  "server_host": "lbnl59-cms-xcache01.es.net",
  "server_ip": "198.124.238.226",
  "file_lfn": "/store/temp/user/jbalcas.cachetest/2023-8-13-13-cache-test",
  "file_size": 16777216,
  "read_single_bytes": 16777216,
  "ipv6": "True",
  "end_time": 1691933902000,
  "operation": "read",
  "user_protocol": "xroot",
  "write_bytes": 0,
  "read_bytes_at_close": 16777216,
  "read_min": 0,
  "read_sigma": 0,
  "read_single_max": 0,
  "read_single_operations": 2,
  "read_vector_average": 0,
  "read_vector_count_max": 0,
  "read_vector_count_sigma": 0,
  "read_vector_min": 0,
  "read_vector_sigma": 0,
  "write_average": 0,
  "write_max": 0,
  "write_operations": 0,
  "metadata": "{}"
}
```

```
  "fallback": "True", "user_dn":       "",
  "client_domain": "ultralight.org",
  "server_domain": "es.net",
  "unique_id": "1c1f3dae-1f34-4bed-a720-ba273f70cf5b",
  "read_bytes": 16777216,
  "read_vector_bytes": 0,
  "start_time": 1691933902000,
  "operation_time": 0,
  "server_site": "ESNET-LBNL59-CMS-XCACHE",
  "vo": "cms",
  "read_average": 8388608.0,
  "read_max": 0,
  "read_operations": 2,
  "read_single_average": 8388608.0,
  "read_single_min": 2147483647,
  "read_single_sigma": 0,
  "read_vector_count_average": 0,
  "read_vector_count_min": 0,
  "read_vector_max": 0,
  "read_vector_operations": 0,
  "throughput": 0,
  "write_bytes_at_close": 0,
  "write_min": 0,
  "write_sigma": 0,
```


Figure 2 An example of sample collection in the JSON format

Most machine learning utilities use samples in the CSV format. Once the monitoring sample is downloaded from the Grafana in the JSON format, the JSON format is converted to a CSV format, and the raw CSV is saved as a zip file. A single zip file corresponds to 6 hours of monitoring samples.

3 Feature and Target Output Selection

Once we have collected the monitoring samples for the analysis of the health of the XRootD system, we need to decide what is the most relevant information in the monitoring samples that can be fed into one of the machine learning techniques. There are a total of 51 variables that can be used for a machine learning technique as one can see from Figure . Some of the attributes may not be appropriate as features in machine learning as they have values that do not discriminate characteristics of the monitoring sample. Those with the value 0 do not contain any information that can discriminate against the health of the XRootD system. Although one can use some machine learning techniques to convert string values into numerical values, some of those variables with string values are not useful in discriminating the characteristics of the monitoring sample. On the other hand, the file size, the read rate, and the variables related with the network are features of interest for an obvious reason as the monitoring data are related with the file access operation.

We chose a machine learning technique that helps to select the appropriate features. Among the many machine learning techniques and considering the monitoring samples at hand, we select a technique for the file read activity that ranks the input features and produces the target output. For this type of machine learning model, we chose these features as the input for a deep learning model: ‘read_bytes’, ‘file_size’, ‘read_bytes per file_size’, ‘ipv6’,

and ‘operation_time’. ‘read_bytes’ is the number of bytes read by the XRootD server. ‘file_size’ is the size of the file that is read. ‘read_bytes per file_size’ is the number of bytes read divided by the size of the file read and is a feature derived from ‘read_bytes’ and ‘file_size’. ‘ipv6’ is a logical variable that shows whether the read activity is through the IPv6 or not. ‘operation_time’ is the time it took to read a file. Also, the most important characteristic in the read activity has to be chosen to predict the health status of the XRootD system. The variable ‘throughput’ is also measured. In file transfers or file read activities, the throughput is the most important variable that can represent the health of the XRootD system and is thus chosen as the target variable to predict the health status of the XRootD system.

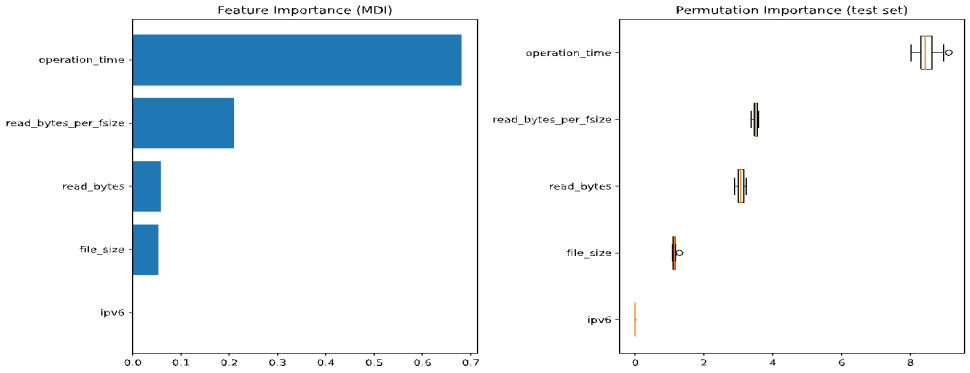

Figure 2 Feature Importance and Permutation Importance

Once the features are chosen, it would be useful to understand the importance of features. In Scikit-learn [9], there is a machine learning implementation to measure the importance of the features. The ‘Gradient Boosting’ in the Scikit-learn is used to measure the importance of the features. The feature importance and the permutation importance are shown in Figure 2 . We can see the operation time has the highest importance as we can expect this from intuition.

4 Monitoring of the XRootD System

There are various deep learning techniques that can be used for the XRootD monitoring samples. We have tried to follow similar examples to save time in finding the best technique. One example that detects anomalies in a time series can be found in Keras [10]. Another example is an application of deep learning algorithms to alarm for grid jobs [11]. A directly related technique is prediction of file transfer durations using Keras [12]. One of these techniques may be used for the XRootD monitoring samples, but the best technique needs to be chosen to monitor any deviation of the XRootD system behavior. Once a technique is chosen, next is getting the best model, applying the model to the XRootD monitoring samples, and defining the metric that needs to be monitored.

4.1 The Machine Learning Technique

We could try the technique used in the ref. [12] which utilize data transfer samples similar to what we are using in this study and have initially considered the possibility of using the Long Short-Term Memory (LSTM) networks, which is what is used in the ref. [12], a type of recurrent neural network capable of learning order dependence in sequence prediction as we are dealing with a sample of time series. However, the XRootD monitoring samples are

recorded not necessarily in order of time because of the file size difference. Instead, we have decided to use a simple deep learning algorithm based on the Keras sequential model as is shown in Figure 4.

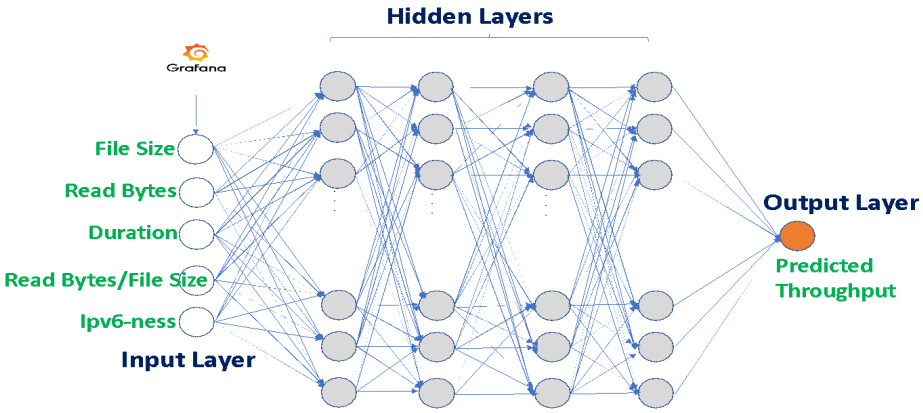

Figure 4 A neural network model that is used for the monitoring of the XRootD system

4.2 Implementation of the Machine Learning Model

The training and validation samples are chosen from initial samples collected for two weeks. The samples have 40 million events with 54 variables from which one can choose the model features.

The saved monitoring samples must be reduced as they contain information that can not be used directly to build the machine learning model. For this study, we have focused on one site (Florida) and limited the read activities to allow only from nine U.S. sites. Here, nine US sites are one Tier1 site at FNAL and eight Tier2 sites at Caltech, Florida, MIT, Nebraska, Purdue, UCSD, Vanderbilt, and Wisconsin. For the input layer for the machine learning model, five features are selected as is shown in Figure 4. The features, file size and duration, are required to be greater than zero as the zero value of these features do not provide any useful information. In addition, the throughput value is also required to be greater than zero. After reducing samples by requiring these conditions, we have chosen 80% of the sample for training and 20% for validation. One of the features, ipv6-ness, is not numerical and we have used the SciKit-Learn encoder to encode the feature. We have used the Sequential model in Tensorflow's Keras module to build the machine learning model.

4.3 Optimizing and Saving the Machine Learning Model

To optimize the performance of the model selected, we tuned the model hyperparameters such as number of hidden layers, application of regularization, and addition of dropout layers to ensure the model does not overfit and will generalize well on the testing data.

Each combination of the number of hidden layers, the regularization function, and dropout layers is tried by checking the learning curve. Eventually, we have decided to use 12 hidden layers with 512 units each, the *ReLU* activation, the *Adam* optimizer, and a configuration of an early stopping. Since the model is decided, the model is built and saved to a file for the test samples that need to be monitored for the health of the XRootD system.

4.4 Monitoring XRootD Health Metric

The health of the XRootD needs to be monitored automatically and we need to define the metric that can tell the health of the XRootD system. We could use the method provided by the deep learning packages, e.g., the binary classifier. However, for the metric we wanted to monitor for the health, we decided to use the root mean squared error (RMSE) of the throughput. The metric is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(OTP_i - PTP_i)^2}$$

where n is the number of samples in a given interval, OTP_i is the i-th observed throughput, PTP_i is the i-th throughput predicted by the machine learning model.

To determine whether the XRootD system is healthy or not, we monitor the *RMSE* variation for each combination of the site and 10 clients that read files on the XRootD server of the site.

5 Discussion

Using the technique described in Section 4, we have monitored the health of the XRootD system at one site (Florida) that we focused on using the average throughput OTP_i and its *RMSE.* The average OTP_i and its *RMSE* during the one-week period as a function of the clients that read files on the XRootD server site is shown in Table 1.

Table 1 Typical average throughput and its RMSE of one week period for an XRootD server site. Each column represents the site where the clients were reading files from the XRootD server site.

Client Site	Caltech	Florida	MIT	Nebraska	Purdue	UCSD	Vanderbilt	Wisconsin	FNAL
OTP (MB/s)	0.9	1.42	0.68	0.95	0.13	1.24	0.43t	0.26	0.26
RMSE (MB/s)	1.57	1.89	1.35	2.53	0.71	3.67	2.36	1.69	1.12

The average OTP_i is typically very small and varies substantially among client sites. In the typical data transfer rate between sites, the throughput is much higher and at least an order of magnitude higher than this read throughput. *RMSE* is also comparable to the OTP_i. We do not have the information of the variation of these due to limited time but during our limited monitoring period, we noted *RMSE* is stable and this means *RMSE* is a good metric of monitoring the health of the XRootD system.

As was described in Section 4.4, we monitored the RMSE variation. We chose the variation of the *RMSE* to be within 1.5 of the trained *RMSE* value for a site to be healthy. This choice of the condition for a healthy site was arbitrary after observing only for a few weeks of data. However, this can be tuned further.

Obviously, we will need more samples to have more stable choice of the health condition. In addition, we could try to explore other possible metrics.

6 Outlook

We have presented the monitoring for a XRootD server site we focused on. We have seen the metric we chose behaves stably and we can expand this technique for other XRootD Shoveler sites. For the client site, we could have included other sites as well. We could also add other client sites to regularly monitor the metric.

Currently, there is a very limited number of features, but we also plan to include more features as they become available in predicting the throughput and monitoring the metric. In addition to the throughput for the read activity, once the XRootD Shoveler can handle the write activity, we could also add the write throughput.

It would be very informative to monitor the variation of OTP_i as a function of the monitoring period as it can tell us the health of the XRootD system. In addition to the *RMSE,* we can consider a study of the health metric that can be used.

Acknowledgement

We appreciate the development of the Shoveler monitoring by the Open Science Grid team. With the reliable Shoveler, this study was possible to use the collected samples.

References

[1] Lyndon Evans and Philip Bryant, LHC Machine, JINST 3 S08001(2008).

[2] The CMS Collaboration, et. al., The CMS experiment at the CERN LHC, JINST 3 S08004(2008).

[3] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky, Xrootd - A highly scalable architecture for data access, WSEAS Transactions on Computers (2005).

[4] M. Barisits, et.al., Rucio: Scientifc Data Management, Computing and Software for Big Science (2019) 3:11.

[5] D. Weitzel, et. al., XRootD Monitoring Collector, https://zenodo.org/record/4670589

[6] B. Garrido, A. Forti, D. Weitzel, J. Andreeva, and S. McKee, New XRootD Monitoring implementation, To Be Submitted to the CHEP2023 Proceedings.

[7] D. Weitzel, et. al., XRootD Monitoring Shoveler, https://zenodo.org/record/8269867

[8] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick, The open science grid, J. Phys. Conf. Ser., 78, 012057(2007). https://doi.org/10.1088/1742-6596/78/1/012057

[9] F. Pedregosa, et. al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830(2011).

[10] F. Chollet, et. al., Keras, Available at https://github.com/fchollet/keras(2015) and Timeseries Anomaly Detection Example, Available at https://keras.io/examples/timeseries/timeseries_anomaly_detection/

[11] E. Torres, Applying Deep Learning Algorithms to Alarm/Anomaly Detection for Grid Jobs, FNAL SIST Program (2017)

[12] V. Sharma, Keras predicting random file transfer durations, https://github.com/vyomshm/Keras-predicting-random-file-transfer-durations (2017)