# Predicting ALICE Grid throughput using recurrent neural networks

**Mircea Popa**[1], **Costin Grigoras**[2] **and Sofia Vallecorsa**[2]

[1]PUB, Splaiul Independenței 313, Bucharest 060042, Romania

[2] CERN, 1, Esplanade des particules, Geneva, Switzerland

E-mail: `Sofia.Vallecorsa@cern.ch`

**Abstract.**
The Worldwide LHC Computing Grid (WLCG) is the infrastructure enabling the storage and processing of the large amount of data generated by the LHC experiments, and in particular the ALICE experiment. With the foreseen increase in the computing requirements of the future High Luminosity LHC experiments, a data placement strategy which increases the efficiency of the WLCG computing infrastructure becomes extremely relevant for the scientific success of the LHC scientific programme. Currently, the data placement at the ALICE Grid computing sites is determined by heuristic algorithms. Optimisation of the data storage could yield substantial benefits in terms of efficiency and time-to-result. This has however proven to be arduous due to the complexity of the problem. In this work we propose a modelisation of the behaviour of the system via principal component analysis, time series analysis and deep learning, starting from the detailed data collected by the MonALISA monitoring system. We show that it is possible to analyse and model the throughput of the ALICE Grid to a level that has not been possible before, comparing the performance of different deep learning architectures based on recurrent neural networks. Analyzing about six weeks of activity, the Grid I/O throughput trend is successfully predicted with a mean relative error of 4%, while the prediction of the throughput itself performs at 5%.

## 1. Introduction

The LHC Computing Grid is the backbone for data analysis, transfer and storage for the LHC experiments at CERN, with computing farms spread all around the Globe [1]. Data processing jobs read and write data files located at different locations following the current data placement management, performed by the MonALISA framework (Monitoring Agents using a Large Integrated Services Architecture)[2]. Its role is twofold. On one hand, it gathers statistical information about the Grid (e.g runs read/write availability tests, gathers data on throughput per various storage elements etc.), while, on the other hand, it directs jobs on where they should read/write a file. Services located at every Grid computing centre send data on Storage Elements (SE) availability, network topology and throughput, computing performance etc.., to a set of central aggregator nodes. In MonALISA, two data structures contain heuristic information on the state of the Grid: a "distance" matrix, linking a computing farm to a SE, whose 6700 elements are produced by network monitoring and a "demotion" matrix, describing the health, availability and occupancy of each SE [4, 3]. For both READ and WRITE operations, the steps are similar. In the case of a READ operation, a job sends a query to the central nodes (CNs) requesting directions on where to read a certain file. The CNs use a catalogue to determine at which SEs the file copies are located. Next, they assign to each SE a heuristic score which is the sum of values obtained from the two previously mentioned matrices. SEs are then sorted from the lowest to the highest score and returned to the job as a sorted list. The job tries to read from the top-score SE. If the attempt fails, then it tries the next SE in the list and so on until the last SE is reached. The process is identical for a WRITE operation with the modification that the returned list of sorted SE contains all available SEs, since a job can write a file to any of them. The ever increasing amount of data generated by the LHC demands the development of a novel approach to manage data placement. The current system, MonALISA, uses heuristics which will, likely, not adapt fast enough to match the rate at which the data files will be requested over the grid, once the High Luminosity LHC run will start. Reducing the frequency at which bottlenecks on the network communication lines of the different computing farms happen would, in fact, improve the overall performance of the Grid.

Preliminary to any kind of optimisation, however, careful modelling of the network behaviour is required. This work is aimed at modelling and forecasting, with a high level of accuracy, the data transfer throughput across the ALICE Grid sites with respects to the real-time network status. In order to capture the infrastructure status, the workload and data traffic, and the measured throughput, we rely entirely on MonALISA. In particular, since 90% of the Grid traffic is represented by read operations, we focus, for simplicity, on the throughput generated by read operations at the CERN site [5]. Therefore, in the context of this study, the throughput is defined by how much data is read from a SE and it is collected by services running locally on the storage element. This work analyses the performance of different techniques aimed at modelling and predicting the throughput: time series analyses approaches, highlighting the effect of different periodic components, and principal components analysis are compared to different deep learning architectures taking inspiration from Seq2Seq [6] models.

## 2. Pre-processing ALICE Grid and MonAlisa data

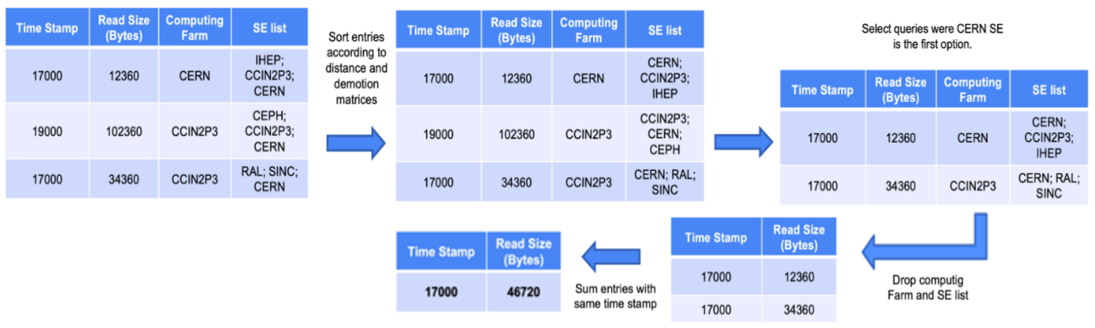
There are three sources of data: the heuristic matrices (distance and demotion), read queries (interpreted as a proxy to grid activity) and throughput. All of them are stored in CSV files. The distance matrix contains network related information for each possible pair made of a computing farm and a storage element for which a float value is assigned, hence it contains around 6700 elements. In other words, it describes how quick data may travel on the Grid. Internally, it is updated roughly once every 7 hours. On the other hand, the demotion matrix (contains around 80 elements) describes the availability of storage elements towards read or write operations. It is generated as a result of several tests.

Read queries, arriving at the central nodes at CERN are stored (˜300 million per week) by MonALISA and contain information such as the query creation time, the file size and the storage elements which hold a copy of the file. Finally, the throughput can be processed into a list of pairs of time tag and bytes-per-second. MonALISA sums up the read bytes every 2 minutes and averages the value. In this paper, we focus on the throughput from CERN.

Query data is highly non-homogeneous and this raises the question of finding an optimal way to aggregate the individual queries. Here we use binning, defining equally long consecutive temporal intervals (bins) and summing up queries whose time tag belongs to a certain bin. Though fairly simple, this method has the disadvantage of generating a wide range of hyper-parameters to tune: bin number, spacing of bins in time, read size splitting per bin (e.g. split read sizes per storage element, per computing farm etc.) etc.

Since the distance and demotion matrices change with different frequencies, we calculate the sum of the two in a special way, the "total matrix", each time one of the 2 changes. Given that a line in the distance matrix represents the quality of the connection between one SE and all of the available computing farms (the line is an array of numerical values equal in size to the count of computing farms), we add to each value in the line the storage element's availability score from the demotion matrix. This process is repeated for each SE resulting in a new 6700 elements matrix. The last bit of information needed as input to our problem is knowing the ordered list of storage elements containing a copy of the file desired by a job. For each query, this list can be deduced by identifying the total matrix which was active during the time the query was generated (working under the assumption that there is no significant delay between the creation of the query and the arrival at the central nodes): this information is registered every second.

Each of the aforementioned quantities (queries, distance matrices, demotion matrices and throughput), have associated time tags which are crucial to pair or to order them for inference/training on a recurrent neural network. The data selection procedure is explained in Fig. 1.



| Time Stamp | Read Size (Bytes) | Computing Farm | SE list |
|---|---|---|---|
| 17000 | 12360 | CERN | IHEP; CCIN2P3; CERN |
| 19000 | 102360 | CCIN2P3 | CEPH; CCIN2P3; CERN |
| 17000 | 34360 | CCIN2P3 | RAL; SINC; CERN |

Sort entries according to distance and demotion matrices

| Time Stamp | Read Size (Bytes) | Computing Farm | SE list |
|---|---|---|---|
| 17000 | 12360 | CERN | CERN; CCIN2P3; IHEP |
| 19000 | 102360 | CCIN2P3 | CCIN2P3; CERN; CEPH |
| 17000 | 34360 | CCIN2P3 | CERN; RAL; SINC |

Select queries were CERN SE is the first option.

| Time Stamp | Read Size (Bytes) | Computing Farm | SE list |
|---|---|---|---|
| 17000 | 12360 | CERN | CERN; CCIN2P3; IHEP |
| 17000 | 34360 | CCIN2P3 | CERN; RAL; SINC |

| Time Stamp | Read Size (Bytes) |
|---|---|
| 17000 | 12360 |
| 17000 | 34360 |

| Time Stamp | Read Size (Bytes) |
|---|---|
| **17000** | **46720** |

Figure 1: The data pre-processing step, illustrated for an example set of queries.

One quintessential fact that needs to be clarified at this point is that one does not know where a job reads a file from for sure. The central nodes only offer suggestions to the jobs and in no way are they constraining the read operation. This makes predicting the throughput a challenging endeavour, but as shown in the following sections, not an impossible task.

## 3. Preliminary studies

The measured throughput contains large fluctuations (see Figure 2). In order to distinguish between fluctuations due to the job submission pattern and the overall system performance, we have applied additive time series decomposition to the data. We have chosen a frequency of the seasonal component that provides a noise component with a gaussian distribution centered around 0. Its value is the time interval in which 4260 consecutive throughput values are gathered by MonALISA. Since a value is reported every 2 minutes, the interval rounds up to roughly 5.91 days (a little more than a working week). Seasonal and noise components of the raw throughput are discarded at this stage, since we aim at optimising the throughput trend.

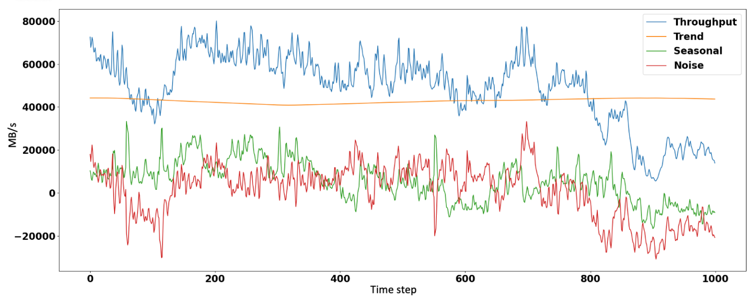


Figure 2: Measured throughput and the noise, seasonal and trend components.

In order to reduce the dimensionality of the input (the heuristic matrices contain, together, thousands of elements), we used and compared two alternative approaches: a principal components analysis (PCA) applied to the input matrixes followed by a recurrent network (RNN), we call this approach Principle Component Network (PCN), and an encoder-decoder network (EDN) architecture coupled to a RNN to output the corresponding throughput trend. For both approaches multiple hyperparameters have been optimised, ranging from the time sequence window to the regularization approach (a 10% dropout strategy) for the EDN architecture. The optimal dimension of the downsized matrix space, expressed by the number of principal components for the PCA is 11 (yielding up to 95% of the total variance) and the latent variables in the case of the EDN is 10. Fig. 3 shows the results on the validation set for the PCN (left) and EDN (right) respectively. In both cases the total accuracy on the throughput trend prediction is around 4%. The accuracy is calculated using an independent validation set extracted from the MonALISA monitoring systems. Predicting the original throughput value yields much worse performance with an accuracy of about 20-30%. The RNN is small in size (from hundreds to thousands of parameters) and is comprised of time distributed dense layers and 2 bidirectional LSTM layers. The EDN (hundreds of thousands of parameters) has one input branch (the encoder), comprised of a single time distributed dense layer, and 2 output branches: the trend predictor and the decoder. The predictor has a similar architecture to the RNN, while the decoder mirrors the encoder. Both use 40 sequential time steps.

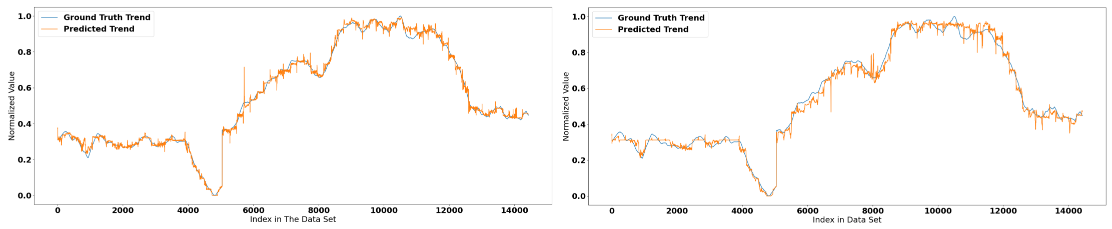
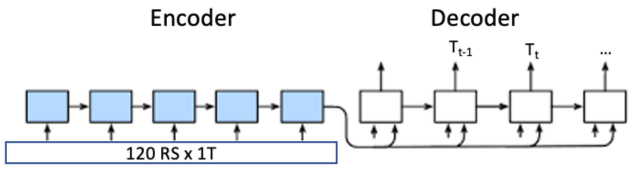


Figure 3: Overlay between the RNN prediction and ground truth on validation data set. The PCA pre-processing step is applied in the left pane. The Encoder_Decoder approach (EDN) on the right.

## 4. The Seq2seq Models

Both the PCA and encoder-decoder architectures encode the matrices information as input to the throughput trend prediction. Our next test explores, instead, the possibility to forecast throughput (not the throughput trend) without using the matrices information and focusing and relying only on the past queries read sizes (RS) and measured throughput values. In order to do so, we designed an architecture inspired by Seq2Seq models. In this case, the input is the total read size calculated over all queries received during 2 minutes, binned in one second intervals (for a total of 120 bins). As explained above, a job may have multiple options on where to read a file from. Based on the total matrix, we filter out those queries which do not have CERN as the first option. Then, we sum the read size from the queries which were generated in a second, then in the next second and so on. In the end, a list of time tag and read size is produced. The 120 read size array spans 120 seconds, or 2 minutes. This interval matches the rate at which the throughput is logged.

The Seq2Seq architecture contains 2 components. The first is an encoder, primarily made by LSTM cells. Per cell, the input is 121 values: the 120 consecutive read sizes and 1 throughput value. The second component is a decoder network which does the prediction. The encoder communicates with the decoder by sending the output and the internal state of the its last LSTM to the first LSTM cell of the decoder. In addition the decoder gets, as input, an array



The architecture of the Seq2Seq model. The LSTM cell input is chosen as the 120 RS values corresponding to 120 seconds in a 2 minutes interval. The corresponding throughput value is measured at the end of the same interval.

Table 1: Sequence-to-sequence architecture details.

| Seq2seq Layers | Output |
|---|---|
| Layers | Dimension |
| Encoder Input | (40,121) |
| Time Distributed | (40,25) |
| BachNormalization | (40,25) |
| Time Distributed | (40,15) |
| BachNormalization | (40,15) |
| LSTM | (40+10,5) |
| Decoder Input | (10,1) |
| Time Distributed | (10,1) |

of consecutive throughput values that follow in time the throughput values from the encoder. However, the first throughput value is the same as the last throughput value from the encoder (they have the same time stamp). The decoder predicts the same sequence shifted forwards in time by one time step (2 minutes). More details can be found in the figure above and in table 1 (in this particular example, the encoder is 40 cells long and the decoder is 10 cells long). The underlying principle is that the encoder extracts information from the previously seen data and sends it to the decoder in order to formulate predictions. By providing, as input, both throughput and read sizes, the Seq2Seq architecture is able to differentiate between how many bytes of data have been transferred before the registering of a throughput value and how many bytes end up affecting future throughput values. This is probably one of the reasons the networks performs so well even when working on the original throughput.

## 5. Results

In order to optimise the prediction performance we tune two main hyper-parameters: the input and the output time steps sequence lengths. They take the following values: 10, 20, 30 and 40 which are equivalent to 20, 40, 60 and 80 minutes due to the fact that the throughput is logged every 2 minutes. We test the performance on the train and validation dataset (TVDS) and a second independent control dataset (CDS) logged a few months later. The TVDS contains 24683 ordered data points, was previously used for the PCN and EDN as well, was split 80% for training and 20% for validation and used in this manner for all 3 networks. The CDS contains 9182 ordered data points and it was used for the Seq2Seq model only as a second validation data set.

We define T as the last time step in the encoder part of the network. As expected, the best prediction is obtained for the T+1 time step (meaning one time step into the future or 2 minutes ahead since throughput values are logged every 2 minutes): 6.35% error on the TVDS and 5.54% error on the CDS. One interesting observation with regards to how the LSTM works is that the mean absolute percentage error (MAPE) between 2 consecutive time steps is around 8.28%. For predicting only one time step into the future, a LSTM modifies the input only slightly which still results in an improvement. However, if one takes into account prediction further into the future, the difference between the current time step and the predicted one is significant. The prediction uncertainty gradually deteriorates while forecasting throughput values over larger time intervals: the throughput value 20 minutes (10 time steps) in the future is predicted with 31.39% error on the TVDS and 33.33% error on the CDS. It is also interesting to notice that, for short term forecasting, the performance of the network on the control data set, CDS, stays very similar to the one measured on TVDS. This result suggests a good generalisation capability of the network over a short time range.

Fig. 4 shows the T+1 prediction over the TVDS dataset together with a zoom in on an area with values in the throughput range [0, 0.5]. One can observe that the difference prediction for one step into the future and the ground truth, that is one step behind, is minimal. This is to be expected as the difference in MAPE between the 2 hovers around 2% in favor of the prediction. At under 34%, the T+10 prediction keeps up remarkably well for predicting 20 minutes into the future. Because most of the throughput values lie under the 0.5 mark, the models have a small bias to under-predict, which is even more accentuated the further the model tries to predict into the future. We can see this as well in Fig. 5: the residuals are calculated as the difference between the ground truth and the best T+1 prediction. The values have a mean of approximately 0.008. Figure 5 shows how the choice of the input and output sequence length affects the quality of the forecast: the best results are achieved using a small input/output size about 5% on the next-step prediction (T + 1). The performance degrades when prediction extends over longer periods of time (about 15% at T + 3) and when increasing input size (probably due to the increasing amount of noise).

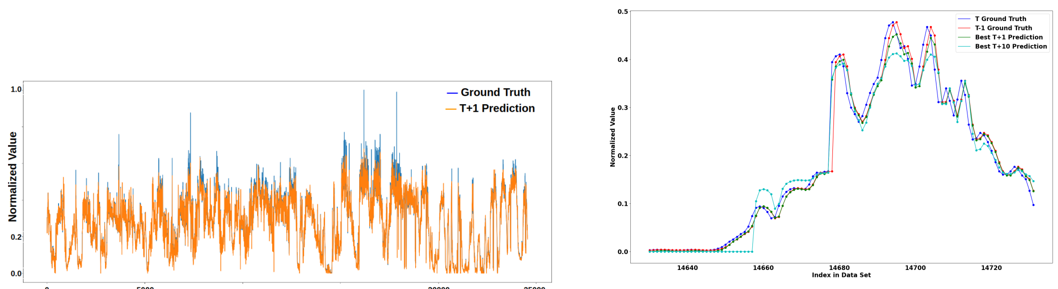


Figure 4: Zoom in for low (left) and high (right) values

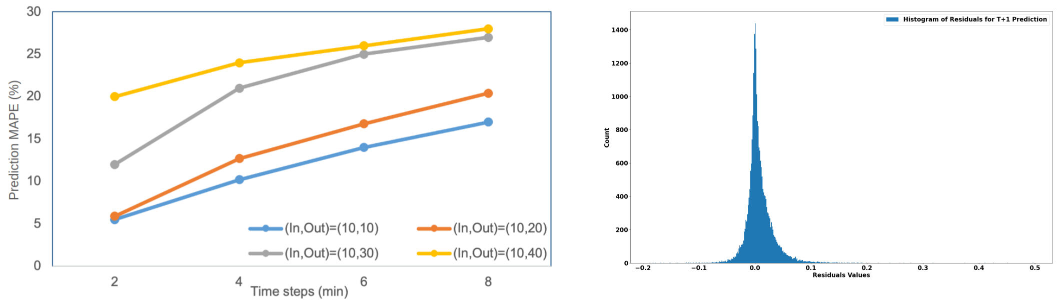


Figure 5: (left) Prediction accuracy for different input and output sequences, measured on the control dataset. (right) Histogram of TVDS Residuals for T+1 Prediction

## 6. Conclusions and Plans

We have shown preliminary studies aimed at predicting the behaviour of a complex system, such as the WLCG, using Deep Learning methods. In particular we have focused on real data, collected by MonALISA, for sites belonging to the ALICE experiment. We have compared different approaches to correlate the Grid network status (parametrized by the distance and demotion matrices) and the job read queries to the read throughput evolution, achieving good forecasting capabilities. The availability of a sizeable training data set, together with the complex procedures needed to extract relevant information from the raw data logged by different systems, have been one of the main limitations of this study. Increasing the training data set will likely yield better predictions.

## References

[1] Shiers, J. (2007). The worldwide LHC computing grid. Computer physics communications, 177(1-2), 219-223.
[2] Legrand, I. et al. (2009). MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems. Computer Physics Communications, 180(12), 2472-2498.
[3] Monalisa. http://monalisa.cern.ch/monalisa.html. Accessed january 2022.
[4] Grigoras C., et al. Eur. Phys. J. Plus (2011) 126: 9
[5] Welcome to ALIMonitor [online] Available at http://alimonitor.cern.ch. Accessed january 2022
[6] Sutskever, I. et al. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.