Forecasting Network Throughput of Remote Data Access in Computing Grids

Volodimir Begy[1,2], Martin Barisits[1], Mario Lassnig[1], and Erich Schikuta[2]

[1] CERN, Geneva, Switzerland
volodimir.begy@cern.ch
[2] University of Vienna, Vienna, Austria

Abstract. Computing grids are key enablers of computational science. Researchers from many fields (High Energy Physics, Bioinformatics, Climatology, etc.) employ grids for execution of distributed computational jobs. These computing workloads are typically data-intensive. The current state of the art approach for data access in grids is data placement: a job is scheduled to run at a specific data center, and its execution commences only once the complete input data has been transferred there. An alternative approach is remote data access: a job may stream the input data directly from arbitrary storage elements. Remote data access brings two innovative benefits: (1) the jobs can be executed asynchronously with respect to the data transfer; (2) when combined with data placement on the policy level, it can aid in the optimization of the network load, since these two data access methodologies partially exhibit nonoverlapping bottlenecks. However, in order to employ this technique systematically, the properties of its network throughput need to be studied carefully. This paper presents experimentally identified parameters of remote data access throughput, statistically tested formalization of these parameters and a derived throughput forecasting model. The model is applicable to large computing workloads, robust with respect to arbitrary dynamic changes in the grid infrastructure and exhibits a long-term prediction horizon. Its purpose is to assist various stakeholders of the grid in decision-making related to data access patterns. This work is based on measurements taken on the Worldwide LHC Computing Grid at CERN.

Keywords: Grid computing · Remote data access · Network modeling · Applied time series forecasting · Applied machine learning.

1 Motivation

Computing grids [5] are collections of geographically sparse data centers. Employing a wide range of heterogenous hardware, these participating facilities work together to reach a common goal in a coordinated manner. Grids store vast amounts of scientific data, and numerous users run computational jobs to analyze these data in a highly distributed and parallel fashion. For example, within the World-Wide LHC Computing Grid (WLCG) more than 150 computing sites are employed by the ATLAS experiment at CERN. WLCG stores more than 450 petabytes of ATLAS data, which is used for distributed analysis by more than 5000 users.

In order to achieve a modular architecture, the grid resources are typically divided into three major classes: storage elements, worker nodes and network. The function of storage elements is to replicate and persist the data for the long term using various technologies (hard disks, solid state drives or tapes). On the other hand, the worker nodes perform CPU intensive tasks. WAN and LAN networks facilitate the data transfer between storage elements and worker nodes. Such strict division of labour has led to the fact that the current best practice for data access in the grid is data placement [3]. Data placement is performed by dedicated distributed data management systems. Consider a job scheduled to run at the data center DC_1, while the required input replica is located at the data center DC_2. The job may commence its execution only after the completion of the following workflow: (1) the input replica is transferred from the relevant storage element at DC_2 to a storage element at DC_1; (2) the input replica is staged-in from the relevant storage element at DC_1 to the worker node. This simplistic approach has two major disadvantages: (1) the jobs are staying idle while waiting for the input data; (2) due to the limited infrastructure resources the distributed data management system handling the data placement may queue the transfers up to several days. An alternative approach is remote data access, which allows to directly stream the input replica from a local or remote storage element.

To demonstrate the potential of remote data access to optimize the resource utilization when combined on policy level with traditional data placement, consider two following scenarios.

In the first example, a computational workload is submitted to a worker node at the data center DC_1. The jobs require the input replica R, which is persisted in the remote storage element at the data center DC_2. Currently, the storage element in the local data center DC_1 has a high load because of transfers to various other data centers. Assume that the bottleneck is only present at this local storage element. This is recognized by the distributed data management system, and it queues the transfer of the input replica R. Employing remote data access, the bottleneck can be immediately bypassed.

In another example, the input replica is located at the local storage element. However, the worker node does not have enough disk space available for stage-in. The distributed data management system queues the file transfer, until the required disk space becomes available. In contrast, using remote data access to the local storage element, the input data can be streamed in fractions without persisting the complete input file.

Given the magnitude of real-life computing grids, it is obvious that many more potential scenarios will benefit from remote data access in the context of resource utilization. This has motivated us to study the properties of its network throughput. Furthermore, a forecasting model of the network throughput is needed for coordination of scientific workloads.

2 Related Work

A substantial body of related work in the literature focuses on prediction of network throughput [18,11,7,17,1,13,2]. However, many authors present only short-term forecasts. In addition, numerous proposed models operate on low-level metrics, which cannot be obtained globally in a computing grid. Finally, the results from the older contributions are outdated with respect to the contemporary network requirements in computing grids. Below we present some notable contributions in more detail.

The authors in [4] describe a method for derivation of short-term throughput predictions based on regression, whose coefficients are continuously updated based on a running time window. The effect of concurrent traffic is not explicitly investigated in this work. The experiments are performed with data probes of up to 16 megabytes. In contrast, our model delivers long-term predictions, taking into account the concurrent traffic of parallel threads and processes. Our work is based on experiments with file transfers totaling up to tens of gigabytes.

Kim et al. [9] have presented an analytical model for prediction of TCP throughput with varying amounts of parallel streams. The performance of the model is not demonstrated for extended forecasting horizons. The model operates on low-level parameters, such as maximum segment size, round-trip time and packet loss ratio. In a grid computing setting such low-level metrics are typically not available globally, since they are not monitored by the data management systems. On the other hand, our model requires only realistically obtainable metrics on file accesses.

Mirza et al. [12] have applied support vector regression for TCP throughput prediction, considering following features as inputs: estimated available bandwidth, queueing delay and packet loss. During the experimental evaluation of the model in the wide area setting, the authors make certain simplifying assumptions (selection of nodes with no or minimal concurrent third-party workloads and concrete operating system), which are impractical for computing grids. Only transfers of 2MB files are considered.

A novel approach for prediction of throughput in data centers is proposed in [14]. First, various components of a data center (topology, configuration, etc.) are modeled in a highly abstract fashion. Then, the initial meta-model is transformed into a Queueing Petri Net. In order to derive predictions, the net is executed by a simulator. The approach is evaluated in a minimal isolated LAN environment.

Recent work in [10] demonstrates feature engineering for throughput prediction based on a wide spectrum of Globus logs. File transfers, which are expected to encounter high third-party concurrent loads during the execution are filtered out from the analyzed dataset. Moreover, the authors assume that nodes within a given site may be treated as equivalent. Our detailed investigation reveals that this is not the case.

3 Parameters of Remote Data Access Throughput

The first phase of this study is the experimental identification of the crucial parameters, which impact remote data access throughput. In our experiment the throughput is indirectly represented by transfer time for a given filesize. Note that these quantities can be easily converted to one another. The hardware of the communicating hosts is a black box. The properties of the hosts are represented exclusively by the observed network throughput. A single computational job is equivalent to a process. Within a process, a new file access is executed by an individual thread. The null hypothesis for the experiment states that for a computational job streaming a given remote file, the total transfer time T cannot be linearly regressed onto the following independent variables: (1) S, the size of the original file; (2) $ConTh$, the cumulative concurrent traffic originating from other threads in the same process during the streaming of the original file; (3) $ConPr$, the cumulative concurrent traffic originating from other processes in the given computing workload during the streaming of the original file. The alternative hypothesis states that such a regression represents the true relationship between the mentioned independent and dependent variables.

The independent variable $ConPr$ stands for traffic, which is created as part of the investigated computing workloads. Other traffic originating from latent workloads is present on the investigated links. In practice it is not possible to obtain complete metrics about a grid. Nevertheless, our minimal approach (following the Occam's razor principle) of monitoring a single additive workload allows us to construct an accurate forecasting model for regression coefficients. The dynamic aspects of the grid are implicitly learned by the model.

To sample the data, 12 computational jobs are launched on a single worker node at the CERN data center (Switzerland) in the period of 28.04.2018 00:00 - 30.04.2018 14:30. In 15-minute steps the processes initiate remote accesses to the storage element GRIF-LPNHE_SCRATCHDISK at the GRIF-LPNHE data center (France). At each step, various numbers of concurrent processes (up to 12) launch various amounts of concurrent threads (up to 4). The threads stream files of different sizes (300MB - 3GB). This allows us to sample a wide range of data for the independent variables. Each launched file access is treated as an observation in the final dataset. After such sampling and transformations we derive 1122 observations, each having values for the variables *T, S, ConTh* and *ConPr*. The protocols used for remote data access are WebDAV/HTTP. Note that such an intrusive sampling design was necessary, because currently these metrics are not logged in WLCG. Once proper remote data access monitoring is implemented grid-wide, the observations can be mined from logs unintrusively. Since WLCG employs commodity hardware, the presented findings are universally generalizable.

A multiple linear regression is fit to the sampled data using ordinary least squares, resulting into the following equation:

$$T = 0.023568 * S + 0.043705 * ConTh + 0.001538 * ConPr \tag{1}$$

The fit has an F-statistic of 4.514e+04 on 3 degrees of freedom, 1119 residual degrees of freedom and a p-value of <2.2e-16. Thus, the null hypothesis is rejected.

Note that the coefficient of *ConTh* is about 28.4 times greater than the coefficient of *ConPr*. This means that concurrent traffic among threads within one process is penalized to a much higher extent in terms of throughput than concurrent traffic among different processes.

To check for interaction among the variables *ConTh* and *ConPr*, two reduced multiple linear regression models are fit on accordingly adjusted datasets using ordinary least squares. The resulting models are demonstrated below:

$$T = 0.02565 * S + 0.04293 * ConTh \tag{2}$$

which is statistically significant, having an F-statistic of 3.119e+04 on 2 degrees of freedom and 340 residual degrees of freedom and a p-value of <2.2e-16, and

$$T = 0.023195 * S + 0.001575 * ConPr \tag{3}$$

which is also statistically significant with an F-statistic of 2.035e+04 on 2 degrees of freedom and 832 residual degrees of freedom and a p-value of <2.2e-16. These fits are depicted in Fig. 1 and 2.

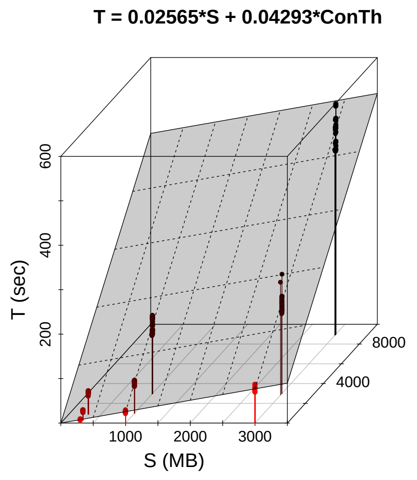

Fig. 1. Regression fit T ∼ 0+S+ConTh

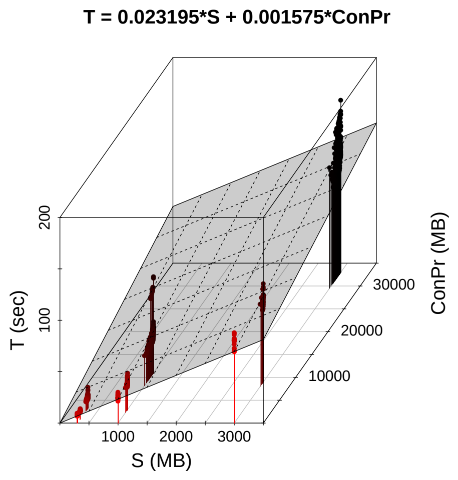

Fig. 2. Regression fit T ∼ 0+S+ConPr

The coefficient of *ConTh* in the reduced model exhibits a log change of only -0.018 in relation to the full model. The coefficient of *ConPr* has a log change of 0.024. This is a hint that in the true relationship, there is no interaction between the variables *ConTh* and *ConPr*.

4 Forecasting Model

The idea behind our novel throughput forecasting model is the following. We have demonstrated that transfer time T can be regressed onto the variables *S, ConTh* and *ConPr*. To facilitate modeling of transfer time T at various points in time, multivariate time series of regression coefficients are formed. Computing grids are highly complex and dynamic systems. Thus, it is practically impossible to capture all the independent variables, which impact the response *T*. The uncaptured variables remain latent. The 3 presented independent variables *S, ConTh* and *ConPr* explain a significant portion of the variability in the response, but not all of it. The effect of the latent variables manifests itself in the variability of the coefficient values throughout time, as can be seen further in their time series plots. Using time series analysis and forecasting techniques, the effect of the latent variables is implicitly captured by learning the patterns in the time series data. The derived model allows to accurately calculate the transfer time (and thus throughput) given arbitrary amounts for *S, ConTh* and *ConPr* at any past, present or forecasted time step. Such calculations can be iteratively applied to every file access in a complex distributed workload.

Consider the time series plot in Fig. 3, which shows transfer times for 2 threads, transferring 2 gigabytes and 300 megabytes respectively, within 1 computing job in the time window 10.03.2018 22:00 - 12.04.2018 21:30. The observations are sampled once every 30 minutes. The computing job runs in the worker node pool ANALY_AUSTRALIA and streams the data from a local storage element AUSTRALIA-ATLAS_SCRATCHDISK. Once per day the job gets resubmitted to a new worker node in the pool, which is selected by the workload management system. This is done to emulate the real-life workflow during job submission. Throughout the rest of the paper a sampling day is equivalent to 48 sampling points. It is evident from the daily shifts along the y-axis that various worker nodes exhibit either very different, or very similar throughputs. We assume that these large shifts are caused by major differences in the hardware of various nodes. Given a single worker node - storage element pair, the captured independent variables explain the vast majority of the variability in transfer time *T*. Thus, the regression model needs to be constructed for each worker node - storage element pair separately. In what follows, we normalize the sampled data with respect to the daily shifts in T to emulate a time series for a single worker node - storage element pair.

Let *a, b* and *c* denote the coefficients of the regression

$$T \sim \ 0 + a * S + b * ConTh + c * ConPr \tag{4}$$

The time series plot in Fig. 3 demonstrate that given a single worker node - storage element pair, the coefficient values slightly vary over time due to latent factors. The sampled time series can be transformed into a multivariate time series of coefficients a and b. The reduced regression is written as

$$T \sim \ 0 + a * S + b * ConTh \tag{5}$$

Let T_1 and T_2 be the sampled transfer times for the 2GB and 300MB files respectively. Since the transfer of the 300MB file always terminates first in our data, the coefficients a and b can be estimated for each time step using the following system of equations:

$$\begin{cases} T_1 = 2000 * a + 300 * b \\ T_2 = 300 * a + (2000 * (T_2/T_1)) * b \end{cases} \tag{6}$$

after substitution we get:

$$a = -((17 * T_1 * T_2)/(100 * (9 * T_1 - 400 * T_2))) \tag{7}$$

$$b = (T_1 - 2000 * a)/300 \tag{8}$$

Note that we did not sample data, which would allow us to reconstruct the time series for the coefficient c. This is due to the demonstrated fact that a vast amount of concurrent traffic among processes is required, in order to reach a noticeable effect. Such sampling would be highly intrusive during a complete month. The complete multivariate coefficient time series can be unintrusively mined from logs, once proper remote data access monitoring is implemented.

Fig. 4 demonstrates the transformation of the sampled time series of transfer times into time series of normalized regression coefficients. The variate for the coefficient c is simulated based on the previous demonstration that in case of remote data access from CERN to GRIF-LPNHE data centers, its value on average was 28.4 times smaller, than that of the coefficient b.

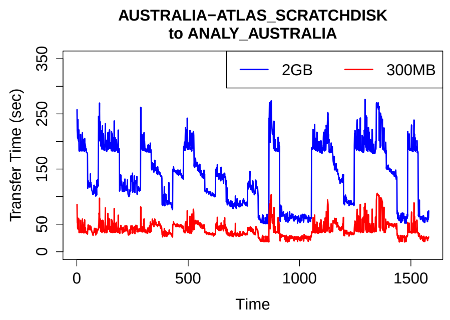

Fig. 3. Australia: transfer times

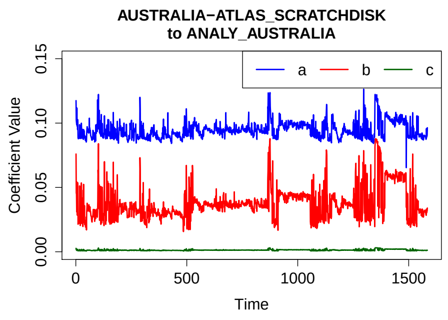

Fig. 4. Australia: normalized coefficients

We have performed an analogous sampling and analysis for the storage element - worker node pool pair BNL-OSG2_SCRATCHDISK - ANALY_BNL_LONG (USA). The transfer time and coefficient time series are shown in Fig. 5 and 6. In infrequent instances in Fig. 6 the values of the coefficients are negative, because the time series is normalized with respect to the first sampling day.

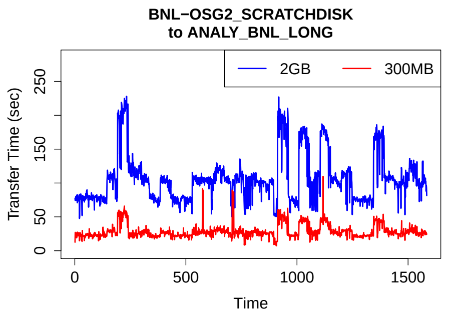

Fig. 5. BNL: transfer times

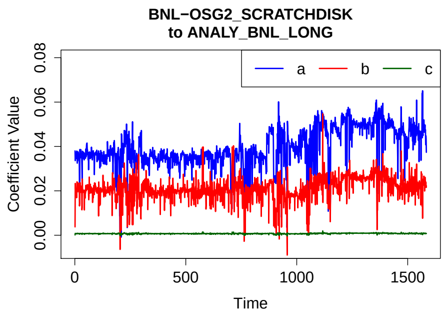

Fig. 6. BNL: normalized coefficients

4.1 Input Features

The forecasting model individually predicts the 3 variates of the multivariate coefficient time series. When forecasting a single variate, following features are considered as inputs to the forecasting model: (1) lagged values of the variate itself; (2) lagged values of 2 other variates; (3) slightly lagged values of the time series representing the overall traffic at the data center of interest.

Consider Fig. 7, which depicts the cross-correlation functions between the normalized coefficients a and b sampled at data centers Australia-ATLAS and BNL-ATLAS.

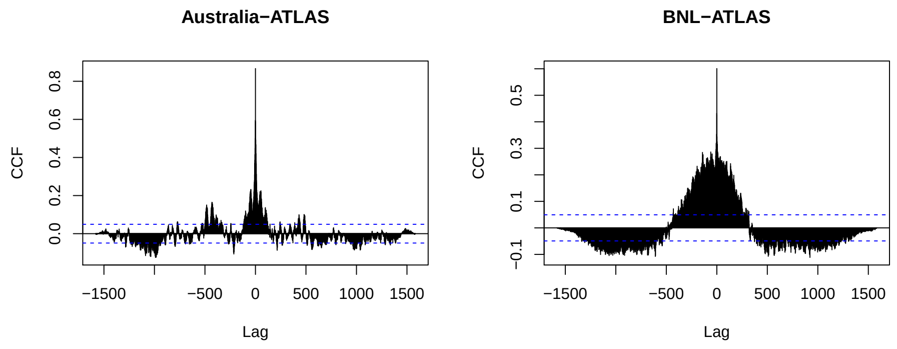

Fig. 7. CCF between normalized coefficients a and b

It is evident from the CCF plots that the values are highly symmetric around the origin. Thus, we do not use other variates as input features to predict a given variate, since they do not posses any additional information for forecasting.

Without a posteriori knowledge, a plausible hypothesis is that the time series of overall network load at the data center of interest leads the amplitude of the coefficient time series with a positive cross-correlation at small negative lags (i.e. if the amount of the overall traffic increases, the time to transfer an additional unit will increase shortly after). We have mined 4 different traffic time series from the logs recorded by the WLCG's distributed data management system Rucio [6] for the investigated period at the data center Australia-ATLAS: (1) intra-data center traffic in bytes; (2) inter-data center traffic in bytes; (3) sum of intra- and inter-data center traffic in bytes; (4) amount of intra-data center file accesses. It can be seen from the cross-correlation plots in Fig. 8 that none of the 4 mentioned traffic time series is significantly leading the normalized coefficient a at small lags with positive cross-correlation. Due to the symmetric CCF between the coefficients a and b, this result is also present in the analysis of CCF between the mentioned traffic categories and the normalized coefficient b. The same analysis delivers insignificant results at the BNL-ATLAS data center. Thus, the overall traffic time series is not used as an input feature for the prediction of the coefficients.

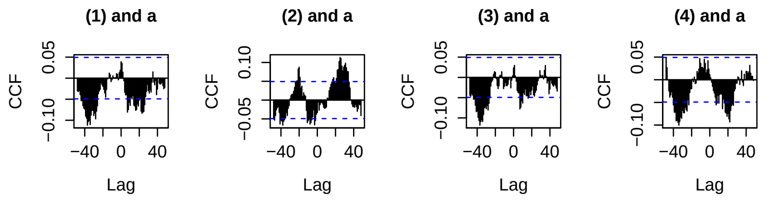

Fig. 8. Australia: CCF between traffic time series and normalized coefficient a

Finally, only the lagged values of the variate of interest will be used as inputs to predict the future values.

We have not sampled authentic data for derivation of coefficient c due to the high associated cost. We assume that in the demonstrated scenarios the CCF between c and other variates exhibits an analogous behavior.

4.2 Benchmark of Forecasting Models

The time series of the normalized coefficient a at the Australia-ATLAS (Fig. 4) and BNL-ATLAS (Fig. 6) data centers are used for demonstrational purposes in this paper. Due to the shown symmetric CCF between coefficients a and b it is known that the coefficient b exhibits an analogous behavior. We assume, that so does the coefficient c.

The aforementioned time series are split into 16 datasets respectively, each spanning a period of 2 sampling days. 80% of each dataset are used for the construction of the model, the remaining 20% are used for its validation. Thus, we are forecasting the series for the next 9.5 hours. Following methods are benchmarked in the context of coefficient time series forecasting: (1) Auto-Regressive

Integrated Moving Average (ARIMA) [15]; (2) Long Short-Term Memory Artificial Neural Networks (LSTM ANN) [8].

ARMA is a statistical state of the art approach for derivation of a mathematical model describing the time series of interest in terms of its past values (AR part) and forecast errors (MA part). ARMA models work with stationary time series. ARIMA, the integrated version of ARMA, allows to work with time series, which become stationary after a certain degree of differencing. The model parameters are identified based on the analysis of the (partial) auto-correlation function and stationarity tests. Such models are computationally cheap to construct. They remain one of the dominant forecasting methods due to their simplicity and effectiveness.

Consider Fig. 9, which depicts the auto-correlation functions of the normalized coefficients a and b at both the data centers.

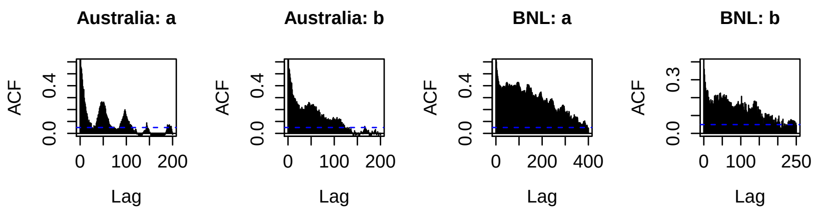

Fig. 9. ACF of normalized coefficients a and b

It is evident from the figure that all the time series have a seasonal component with a period of 48 points, which is equivalent to a day. To fit ARIMA models we use the *auto.arima()* function from R, which will estimate the model parameters based on a number of statistical tests (KPSS, OCSB and others). When passing a time series object to the function, we indicate the discovered seasonality.

Long Short-Term Memory neural networks are a kind of recurrent neural networks. Employing a gating architecture and a recurrent state, LSTM nets are optimized to partially overcome the vanishing gradient problem and learn from arbitrarily extended lags. They have been used for time series forecasting by researchers in different domains, e.g. for traffic speed prediction, detection of faults in industrial data, etc. It has been also demonstrated at the recent time series forecasting competition CIF 2016 [16] that LSTM has delivered outstanding results, outperforming common statistical and soft computing methods. In contrast to ARIMA, LSTM models can learn more families of functions.

We use the PyTorch library to construct LSTM models. At each timestep the input is 1- and the output is 19-dimensional. Prior to the training the data is projected onto the [-1; 1] interval with the goal to achieve stronger gradients. The models are trained using the mean squared error (MSE) loss, since no further normalization is required. We use the Adam algorithm for the optimization of the model parameters. The first 3 datasets from each of the data centers (6 in total) are used to perform a grid search and determine the optimal values for hyperparameters based on the validation set approach. The number of considered hidden layers varied from 1 to 3, the number of training epochs from 1 to 2000 and the learning rate from 1e-4 to 1e-2. A dropout layer with probability 0.5 is introduced between the hidden layers for regularization. Based on the results of the grid search, the final model has one hidden layer, and is trained for 1827 epochs (which corresponds to early stopping regularization) with a learning rate of 1e-2. Fig. 10 shows the validation loss for different combinations of epochs and hidden layers, given the optimal learning rate.

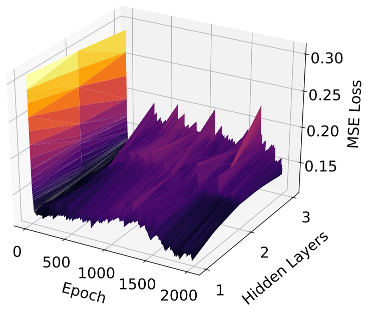

Fig. 10. Grid search validation loss

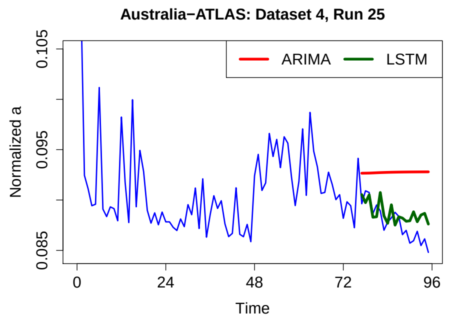

Fig. 11. LSTM outperforming ARIMA

The remaining 13 datasets from each of the data centers (26 in total) are used to compare the performance of ARIMA and LSTM. Given the input time series, the *auto.arima()* function returns a deterministically computed ARIMA model. In contrast, LSTM instances will differ from run to run due to the stochastic aspects of their training. To facilitate an extensive comparison of the performance of both methods, we train 50 LSTM instances on each dataset. The scale of the LSTM forecasts is reversed from [-1; 1] to the original one. Normalized root mean squared error (NRMSE) is used to compare loss among all datasets and forecasting methods. For each of the datasets the corresponding LSTM training runs are considered, and the log changes between ARIMA and LSTM NRMSE (holding ARIMA NRMSE as reference) are computed, resulting into a distribution. Our findings show that given a dataset, one approach will systematically outperform the other. Consider Fig. 11 and 14, which show concrete examples of LSTM outperforming ARIMA(3,0,2) with non-zero mean and ARIMA(0,1,1)(0,0,1)[48] with drift outperforming LSTM.

We have performed the augmented Dickey-Fuller test implying stationarity under the alternative hypothesis on each dataset (on its fraction used for model construction). We have then tested for an association between the ADF test statistic and the type of the forecasting method, which dominates in terms of performance given the dataset. Unfortunately, this analysis did not deliver sig-

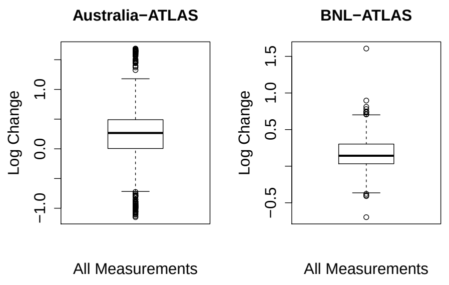

Fig. 12. Log NRMSE changes

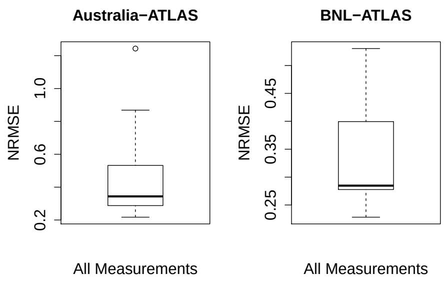

Fig. 13. Performance of ARIMA

nificant results. Thus, the method, which is more beneficial on average needs to be selected. Fig. 12 shows boxplots of all measured log NRMSE changes for the two separate data centers. The median log NRMSE changes are 0.27 and 0.14 for Australia-ATLAS and BNL-ATLAS data centers respectively. According to these results, ARIMA models exhibit better performance. We attribute this to the fact that low-capacity models are able to robustly fit high-variance data and adopt ARIMA for our throughput forecasting model. The boxplots of ARIMA NRMSE across all datasets are shown in Fig. 13.

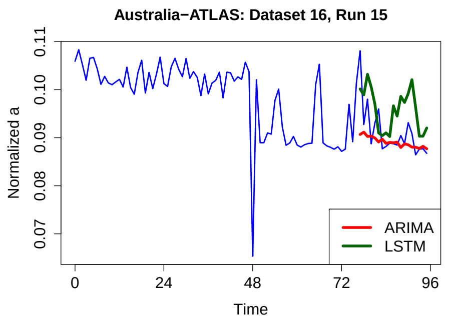

Fig. 14. ARIMA outperforming LSTM

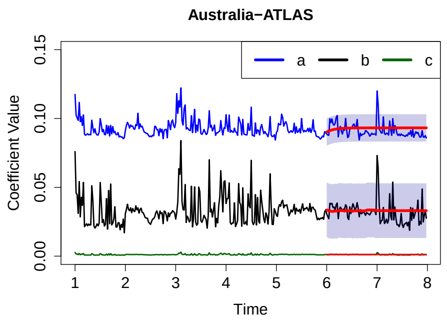

Fig. 15. Long-term forecasts

The forecasting model delivers short- and long-term predictions. Fig. 15 demonstrates multivariate 2 days forecasts. The mean forecasts are depicted in red and their 95% confidence intervals in purple. Note that the confidence intervals miss only few outliers. Fig. 14 demonstrates that by increasing the time resolution, the forecasts become more accurate in the short-term.

5 Conclusions and Future Work

In this paper we have demonstrated that the network throughput of remote data access in computing grids can be framed as a multiple linear regression. The regression needs to be fit for each worker node - storage element pair. The estimates of the regression coefficients can be mined from logs in form of time series. This requires realistically obtainable logs on file accesses from only 2 concurrent processes, one of them having 2 concurrent threads. Given a proper implementation of remote data access monitoring, these time series can be mined unintrusively. We have identified useful input features for the forecasting model. The ARIMA model learns to explain the residual variance in the response induced by the latent factors, e.g. changes in the grid infrastructure. A benchmark of LSTM ANN has partially shown good results, but did not significantly outperform ARIMA. The model can be constructed with a predefined resolution and deliver short- and long-term forecasts with the state of the art accuracy.

These findings allow to robustly model the remote data access throughput for any virtual links in the grid. The model was constructed and validated using performance metrics from randomly selected data centers in the World-Wide LHC Computing Grid.

Currently we are focusing on the following aspects for the future work:

– Formalization and forecasting of network throughput of other data access techniques in the grid (data placement and stage-in).
– Construction of a grid simulator with a heavy emphasis on various data access techniques.
– Evolutionary optimization of data access patterns within bags of jobs with the objective to minimize the joint data transfer time. This constitutes a constrained optimization problem. The solution fitness evaluation employs the aforementioned simulator.

Acknowledgment

This work was performed on behalf of the ATLAS Collaboration.

References

1. Akioka, S., Muraoka, Y.: Extended forecast of cpu and network load on computational grid. In: IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004. pp. 765–772 (April 2004). https://doi.org/10.1109/CCGrid.2004.1336711
2. Arslan, E., Guner, K., Kosar, T.: Harp: Predictive transfer optimization based on historical analysis and real-time probing. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 25:1–25:12. SC '16, IEEE Press, Piscataway, NJ, USA (2016), http://dl.acm.org/citation.cfm?id=3014904.3014938

3. Chervenak, A., Deelman, E., Livny, M., Su, M.H., Schuler, R., Bharathi, S., Mehta, G., Vahi, K.: Data placement for scientific applications in distributed environments. In: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing. pp. 267–274. GRID '07, IEEE Computer Society, Washington, DC, USA (2007). https://doi.org/10.1109/GRID.2007.4354142
4. Faerman, M., Su, A., Wolski, R., Berman, F.: Adaptive performance prediction for distributed data-intensive applications. In: SC '99: Proceedings of the 1999 ACM/IEEE Conference on Supercomputing. pp. 36–36 (Nov 1999). https://doi.org/10.1109/SC.1999.10048
5. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. The Elsevier Series in Grid Computing, Elsevier Science (2004)
6. Garonne, V., Vigne, R., Stewart, G., Barisits, M., Beermann, T., Lassnig, M., Serfon, C., Goossens, L., Nairz, A.: Rucio – the next generation of large scale distributed system for atlas data management. Journal of Physics: Conference Series **513**(4), 042021 (2014), http://stacks.iop.org/1742-6596/513/i=4/a=042021
7. Hacker, T.J., Athey, B.D., Noble, B.: The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network. In: Proceedings 16th International Parallel and Distributed Processing Symposium. pp. 10 pp– (April 2002). https://doi.org/10.1109/IPDPS.2002.1015527
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
9. Kim, J., Yildirim, E., Kosar, T.: A highly-accurate and low-overhead prediction model for transfer throughput optimization. Cluster Computing **18**(1), 41–59 (Mar 2015). https://doi.org/10.1007/s10586-013-0305-4
10. Liu, Z., Balaprakash, P., Kettimuthu, R., Foster, I.: Explaining wide area data transfer performance. In: Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing. pp. 167–178. HPDC '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3078597.3078605
11. Lu, D., Qiao, Y., Dinda, P.A., Bustamante, F.E.: Characterizing and predicting tcp throughput on the wide area network. In: 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05). pp. 414–424 (June 2005). https://doi.org/10.1109/ICDCS.2005.17
12. Mirza, M., Sommers, J., Barford, P., Zhu, X.: A machine learning approach to tcp throughput prediction. IEEE/ACM Trans. Netw. **18**(4), 1026–1039 (Aug 2010). https://doi.org/10.1109/TNET.2009.2037812
13. Nine, M.S.Q.Z., Guner, K., Kosar, T.: Hysteresis-based optimization of data transfer throughput. In: Proceedings of the Fifth International Workshop on Network-Aware Data Management. pp. 5:1–5:9. NDM '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2832099.2832104
14. Rygielski, P., Kounev, S.: Data center network throughput analysis using queueing petri nets. In: 2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW). vol. 00, pp. 100–105 (June 2014). https://doi.org/10.1109/ICDCSW.2014.11
15. Shumway, R., Stoffer, D.: Time Series Analysis and Its Applications: With R Examples. Springer Texts in Statistics, Springer New York (2010)
16. Štěpnička, M., Burda, M.: On the results and observations of the time series forecasting competition cif 2016. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–6 (July 2017). https://doi.org/10.1109/FUZZ-IEEE.2017.8015455

17. Vazhkudai, S., Schopf, J.M.: Predicting sporadic grid data transfers. In: Proceedings 11th IEEE International Symposium on High Performance Distributed Computing. pp. 188–196 (July 2002). https://doi.org/10.1109/HPDC.2002.1029918
18. Wolski, R.: Experiences with predicting resource performance on-line in computational grid settings. SIGMETRICS Perform. Eval. Rev. **30**(4), 41–49 (Mar 2003). https://doi.org/10.1145/773056.773064