Predicting Slow Network Transfers in Scientific Computing

Robin Shao
University of California
Berkeley, CA, USA
robin_shao@berkeley.edu

Jinoh Kim
Texas A&M University
Commerce, TX, USA
jinoh.kim@tamuc.edu

Alex Sim
Lawrence Berkeley National Laboratory
Berkeley, CA, USA
asim@lbl.gov

Kesheng Wu
Lawrence Berkeley National Laboratory
Berkeley, CA, USA
kwu@lbl.gov

ABSTRACT

Data access throughput is one of the key performance metrics in scientific computing, particularly for distributed data-intensive applications. While there has been a body of studies focusing on *elephant* connections that consume a significant fraction of network bandwidth, this study focuses on predicting *slow* connections that create bottlenecks in distributed workflows. In this study, we analyze network traffic logs collected between January 2019 and May 2021 at National Energy Research Scientific Computing Center (NERSC). Based on the observed patterns from this data collection, we define a set of features to be used for identifying low-performing data transfers. Through extensive feature engineering and feature selection, we identify a number of new features to significantly enhance the prediction performance. With these new features, even the relatively simple decision tree model could predict slow connections with a F1 score as high as 0.945.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Network performance analysis**; **Network monitoring**; • **Computing methodologies** → **Feature selection**; **Classification and regression trees**.

KEYWORDS

network transfer, slow connection, prediction, machine learning, scientific computing

ACM Reference Format:
Robin Shao, Jinoh Kim, Alex Sim, and Kesheng Wu. 2022. Predicting Slow Network Transfers in Scientific Computing. In *Proceedings of the Fifth International Workshop on Systems and Network Telemetry and Analytics (SNTA'22), June 30, 2022, Minneapolis, MN, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3526064.3534112

1 INTRODUCTION

Data access throughput is one of the key performance metrics in scientific computing. Data-intensive scientific applications, such as climate modeling [9], bioinformatics [13], and particle physics [1], often exchange a large amount of data across geographically dispersed sites. For instance, a Large Hadron Collider (LHC) experiment produces petabyte-scale data and distributes it to 160 computing facilities around the world [3]. Another example is the Sloan Digital Sky Survey (SDSS), which creates a significant volume of data used by many astronomers from different continents [16]. Most of these users would not have the storage resource to replicate the whole dataset, so they rely on dynamic mechanisms to retrieve the necessary data records [6].

To effectively support such remote data accesses in scientific computing, we need to ensure appropriate networking and storage resources are allocated as needed. For example, if it would be possible to predict traffic patterns at different times of a day, network resources could be optimized to improve the overall quality of network experiences [10]. In the research literature, there is a body of publications on massive data flows, known as "elephant" flows, which consume a significant fraction of network bandwidth over a long period time [2, 5]. Such elephant flows have to be handled appropriately, otherwise, they could easily fill up network buffers causing heavy congestion, considerable queuing delay, and packet losses.

At the same time, it is also important to identify "slow" connections that might be a symptom of a misbehaving network. Such slow connections would also increase the time needed for data transfers and cause unexpected delays in large scientific workflows, because many workflows contain sequences of interdependent tasks [14]. In such applications, any delay in data accesses may lead to further delays in subsequent tasks with compounding effects. Even for a single independent application, the user may want to utilize a reliable prediction to more efficient decide tasks such as reserving the required storage and networking resources along with the necessary compute resource. Despite its importance, much less attention has been given to the understanding and predicting of low-performing communications. *In this study, we explored various properties of slow connections using a set of network traffic logs from a scientific computing facility (NERSC[1]) and developed a prediction mechanism to identify under-performing connections.*

We explore and analyze five-month-long network traffic data recorded by `tstat` [7]. In scientific facilities, data transfer nodes (DTNs) are often defined as dedicated systems for data transfers with the mission to improve network performance [8]. For instance, NERSC maintains DTNs to facilitate data exchange with other sites

ACM ISBN 978-1-4503-9315-7/22/06…$15.00
https://doi.org/10.1145/3526064.3534112

[1]https://www.nersc.gov/

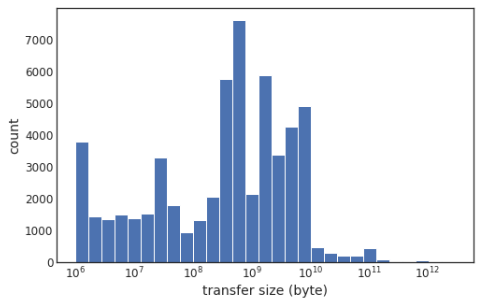


Figure 1: Distribution of transfer size (> 1MB)

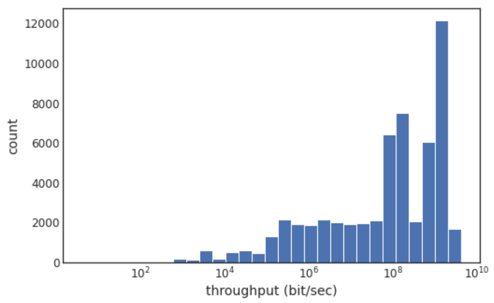


Figure 2: Distribution of transfer throughput (in bps)

over a large-scale network. This study focuses on the network data created by one of the DTNs (dtn01), collected from January 2021 through May. Based on the data exploration, we define a set of features from recently completed data transfers. We will show that the defined feature set is highly effective for predicting slow connections in advance, with up to 0.945 prediction performance (in F1 score) even with conventional tree-based learning algorithms without intensive model optimizations.

The key contributions of this paper can be summarized as follows:

- We present a careful statistical study of a large set of network traffic measurement. Our data exploration reveals several interesting findings impacting data transfer performance, including data size, retransmission rates, and geographical locations (e.g., country codes and network domains) (Section 2);
- We present our predictive model based on features known before the start of a data transfer. These features include information such as file size, source, and destination. To capture the state of the network between source and destination, we extracted information from the most recently completed data transfer between the same source and destination networks (Section 3);
- We optimize the prediction model and show that the extracted information from recently completed transfers is effective in enhancing the prediction results. We are able achieve an F1 score of 0.945 on the test data. Additionally, we also show the relative importance of the individual features (Section 4).

2 DATA DESCRIPTION AND EXPLORATION

2.1 Description of `tstat` Data

Tstat is a log format[2], in which each row corresponds to a different flow and each column is associated with a specific measure. The columns are grouped according to Client-to-Server (C2S) and Server-to-Client (S2C) traffic directions.

The dataset has 116 features among which we focused on the set of features that contains the basic information for all TCP flows. Our dataset provides both C2S and S2C data for the IP addresses, the number of bytes transmitted in payload, number of retransmitted time, first and last payload absolute time, flow duration, and all

[2]http://tstat.tlc.polito.it/measure.shtml

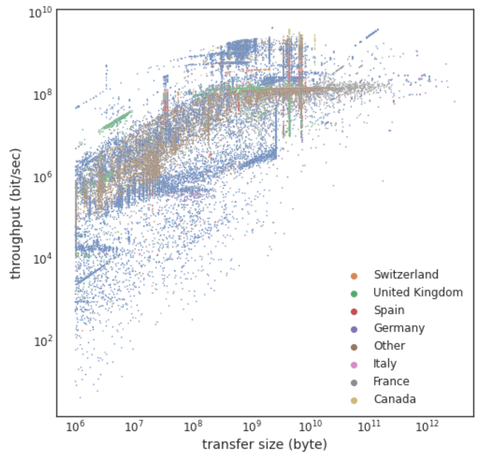


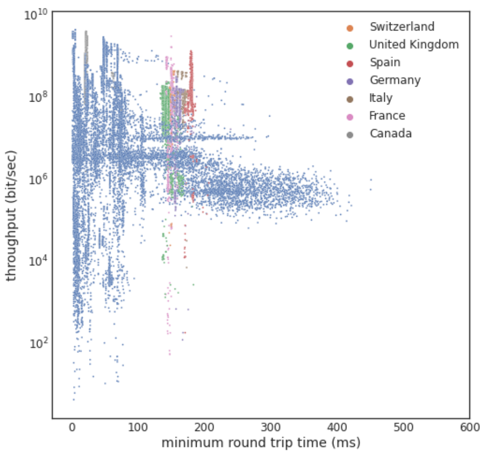
Figure 3: Correlation between transfer size and throughput (colored by country)

the round trip time (RTT) features. These are the basic features we would use in the study. It is important to mention that most of these data won't be available before the transfers take place. We only know the IP address and the number of transmitted bytes in advance. With limited known information, a major challenge of this study is to define new features to predict network performance.

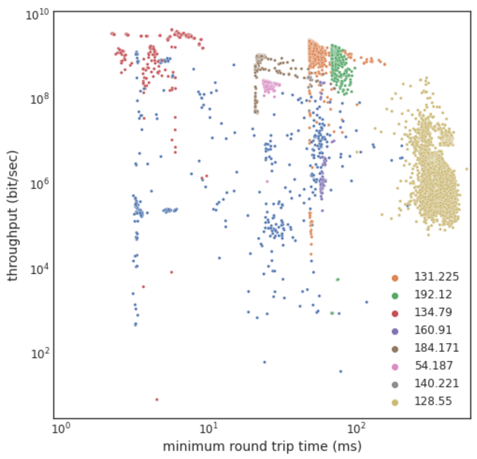
The data we used to build this classification model is from dtn01 in 2021 which is the latest available data when we started the research. This specific dataset alone has over 3 million logs. The size of all the data could not even fit into the over 40 GB online computing memory, so we applied divide-and-conquer to load all the data from compressed files.

2.2 Data Exploration

The source of all the data used in this project is the DTNs at NERSC. The dataset contains all tstat data for 4 DTN nodes from January 2019 to May 2021. We started from data within a single day in 2019 and eventually use all the data from 2021 to build the model. An important and tricky fact of the tstat is that a lot of the features are separated into two different entries, the client-to-server (C2S) and server-to-client (S2C) parts. This is different from the origin and the destination of the transfer. The destinations in our dataset are all NERSC, but the NERSC can be either the client or the server. The



(a) By country



(b) By network address

Figure 4: Impact of minimum RTT to throughput

origins of the transfers are different labs or even personal computers from all over the world.

We have to either analyze the S2C and C2S separately or combine them together and only care about which are the origins and destinations of the connections. Since the number of S2C data is significantly smaller in our observation than that of C2S data, and the distributions of different features are similar, we decide to combine these two different settings. We could distinguish S2C and C2S by the entries available because only one of the fields is available in a single log. In the rest of the paper, we will not mention C2S or S2C which could create a lot of confusion. All the features follow the direction from a distinguished origin to the NERSC, and S2C or C2S is not a factor we consider in our model.

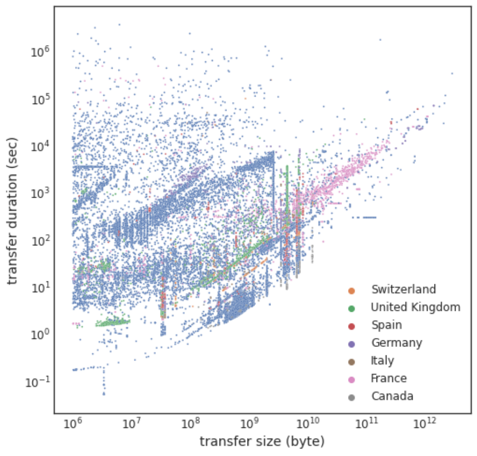
To understand the dataset better, a comprehensive exploratory data analysis was done. The first few steps include deriving new features that are crucial in network transfers but not directly recorded in tstat, changing the units to more intuitive ones, and setting up basic thresholds. We calculated throughput which is one of the most important factors of network performance. We also calculated retransmission rates and decoded the countries or labs from IP addresses. For the units, we changed millisecond to second, and byte per second to bit per second. In the whole study, we only considered transfers for files that are larger than 1×10^6 bytes because smaller files won't be so influential to the overall performance. We also got rid of transfers with recorded minimum round trip time smaller than 1 ms. These are either local transfers within NERSC or not properly recorded. Neither of them are scenarios worth considering under our research target.

The focus of our study is the long-lived and bandwidth-hog connections. This type of connection is always relevant to the transfers of relatively large files. Also, there is a clear gap around $1{\times}10^6$ in the histogram of transfer size in the sampled transfers from both 2019 and 2021 data. The transfer smaller than this threshold may be acknowledgements or code transfers. As a result, we only consider transfers for files that are larger than 1×10^6 bytes. The smaller transfers will not be included in any part of the study. In the transfer size histogram in Figure 1, it shows the distribution of transfer size in a log scale.

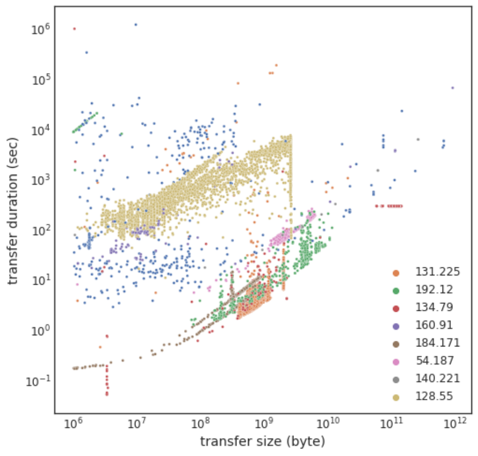
Talking about network performance, throughput is the most direct variable measuring the bandwidth of a transfer. Throughput tells you how much data was transferred from a source at any given time. However, throughput is not directly given in tstat, so we calculated the throughput of a transfer by dividing the payroll size by the duration of the transfer. The transfers with lower throughput could be considered low-performance generally. In Figure 2, we show the distribution of throughput in log scale. The left tail of the distribution can naturally be our definition of low performance as the majority of the transfers are exponentially larger in throughput performance.

Retransmission is another mechanism that is relevant to performance theoretically. Retransmission is the resending of packets that have been either damaged or lost. It could slow down the overall transfer efficiency. The retransmission size is given, but we need the retransmission rate which more directly determines how unstable the transfer is. The retransmission rate's correlation coefficient with throughput is 4 times that of retransmission size. Intuitively, we should not ignore a retransfer in a small transfer as a larger transfer going through the same path may experience similar damage or loss which could lead to more influential retransmission. We considered using retransmission as part of our definition of low-performance in our study. This is discussed in the next section.

Previously, we have discussed the importance of the transfer size in network transfers. The only other accessible feature before the transfer is the IP address. If one-hot encoding is directly applied to IP addresses, it could create so many features and thus may cause significant overfitting. We decided to convert the IP addresses to the countries and the institutions that they are assigned. We used GeoIP2 database to find the respective countries. From the scatter plots shown in Figure 3, we use the most important two variables, and the country code gives us a general idea of how the transfers from each major countries perform. Intriguingly, the majority of the slow transfers are from the US. We explored if there are any



(a) By country



(b) By network address

Figure 5: Correlations between transfer size and duration

specific problematic network flows to NERSC. We found most of them are from random locations which could be a personal laptop at a work-from-home scenario, while the large computing facilities are not the origin of those low-performance transfers.

While exploring the important features, we also confirmed the relationship between RTTs and the distances of the transfers. We calculated the correlations of three different RTT variables with the throughput. The minimum RTT has the highest correlation coefficient of -0.45, and the slope for the fitted linear regression is -5×10^6. As shown in Figure 4, the scatter plot of minimum RTT is colored by country and IP information described in the previous paragraph. It is expected that all the transfers with short minimum RTTs are from the US and Canada which are geographically closer to NERSC. On the other hand, the European sites all have the minimum RTT in a certain range between 130 and 200 milliseconds. The Figure 4 (b) has its x-axis in log scale because the US RTTs are concentrate on the smaller values. The cluster to the far right has a much higher minimum RTT may be due to the VPN usage. In the scatter plot of transfer duration vs transfer size in Figure 5, there is a weak positive trend that meets the expectation. We could understand the vertical lines representing the files of similar or the same sizes, but the horizontal lines representing concentrations of similar or same duration are unexpected. They could be caused by time limitations set by the particular institutions.

3 PREDICTION METHODS

3.1 Slow Connections

To achieve the goal of predicting the low-performance network flows, we determined the threshold for low-performance. We decided to use the throughput of 1 megabit per second as the threshold for the low-performance based on the observed performance at NERSC, real-life data transfer experience, and clues from data visualizations like the ones shown above. Under this threshold for the low performance, 15.5% of the transfers are categorized as low-performance transfers. We decided not to add features that are dependent on throughput to keep the definition clean and adjustable. We have found a negative linear relationship between retransmission rate and throughput even though the retransmission size is not so relevant. A high retransmission rate could be a good indicator of low performance. In fact, there could be a scientifically more rigorous way to define the low-performance, but we would use the relatively intuitive definition using only throughput as it is not the main focus of this study.

3.2 Feature Engineering

As discussed before, we calculated transfer throughputs, retransmission rates, and generated country and institution features based on IP addresses. The other newly created features are features derived from previous transfers. Since we do not have the access to important measurements of a transfer like duration, throughput, and retransmission rate before the transfer, we use these features of the most recent transfer from the same subnet. The transfers from a similar location and host should have similar behaviors. These features are used in model building and testing. In addition, we added the ratio between the current transfer size and the previous transfer size. This could be helpful as it measures the difference between current and previous transfers. The other added feature is the gap between the previous transfer and the current transfer. The larger the gap is, the less relevant the two transfers may be.

The low-performance transfers we defined are those below 1 megabit per second. However, to check the effectiveness of the model and provide broader prediction information, we also created labels for transfers under 100 kilobits per second and 10 megabits per second. Then, we applied one-hot encoding for countries so that they can be fitted into the models we plan to use. We also standardized all the numerical features so that they have equal weight in the model and reduce the bias created by specific features with relatively large absolute sizes. The other features were directly used after basic data cleaning.

As the only two features entirely based on the information from the current transfer, the importance of transfer size and country

Table 1: Features defined for prediction

Feature	Description
`prev_tput`	Latest throughput measured between the same source and destination networks ("a.b.c.0")
`prev_size`	Latest transfer size (in bytes) between the same source and destination networks ("a.b.c.0")
`size_ratio`	Ratio between the latest transfer size (`prev_size`) vs. current transfer size
`prev_durat`	Latest transfer duration (in msec) between the same source and destination networks ("a.b.c.0")
`prev_min_rtt`	Latest minimum RTT between the same source and destination networks ("a.b.c.0")
`prev_rtt`	Latest average RTT between the same source and destination networks ("a.b.c.0")
`prev_max_rtt`	Latest maximum RTT between the same source and destination networks ("a.b.c.0")
`prev_retx_rate`	Latest retransmission rate between the same source and destination networks ("a.b.c.0")
`time_gap`	Time gap from the latest transfer to the current transfer between the same source and destination networks ("a.b.c.0")

derived from IP address (country) are explained in the data exploration section. They would be considered as potential features in our models. Meanwhile, the majority of the features used are derived from the previous transfers. While Table 1 shows the summary of the defined variables in this study, the basic idea and intuition about the selection of the features are as follows.

Since the performance is categorized based on the throughput, the previous throughput (`prev_tput`) is the most direct measure of the performance in the previous transfer and could be indicative of the current transfer's performance. Size could influence the throughput as larger transfers tend to have higher efficiency. We would include both previous transfer size (`prev_size`) and the ratio of the current size to previous size (`size_ratio`) to capture the relationship between transfers. Also, we include the other part of throughput which is the duration. In general, transfers with longer duration tend to have smaller throughputs. Duration is included as previous duration (`prev_durat`). The other important aspect of a transfer is its round trip time (RTT). We include all three features: previous minimum round trip time (`prev_rtt_min`), previous average round trip time (`prev_rtt`), and previous maximum round trip time (`prev_rtt_max`). Minimum RTT is the most accurate reflection of the distance of the transfer, while the average and the maximum are also influenced by the condition of the network. Retransmission rate is also clearly influential to the performance, so we include the previous retransmission rate (`prev_retx_rate`). The transfers with high retransmission rates tend to be less stable and could create a lot of extra work. In addition to size_ratio, we come up with another measurement of the relevance of the current and previous transfer. The time length of the gap between two transfers (`gap`) could reflect if the previous transfer features are particularly relevant.

To confirm the effectiveness of features, we visualize the distribution differences between normal and low-performance transfers for different features. We show the six most representative plots in Figure 6. Size is the most important feature based on our model. It obviously has the most noticeable difference in the distribution. The difference in the gap is the least obvious, and as expected, it is not included in the top feature combinations for the classification model. Even though this is not a direct indicator of the effectiveness of features, it gives us some insights into potential choices.

In the context of the study, we did train-test-split and only used the training data for the model building phase and the testing data for evaluation and testing. We would test the model on more recent data when they are available.

3.3 Prediction Algorithms

After setting up all the features and labels, we build binary classification models to predict the low-performance transfers from different data origins to NERSC. We could use the size, country, previous transfer features (`prev_tput`, `prev_size`, `size_ratio`, `prev_durat`, `prev_rtt_min`, `prev_rtt`, `prev_rtt_max`, `prev_retx_rate`, `gap`) to predict the binary label of whether the transfer is a low-performance one.

We build a few classification models, including decision trees, random forests, SVM, XGBoost, and neural networks. Tree-based models could be particularly effective because the number of features is not too large and the dependencies among features can be expressed. Decision tree is the most simple model and is the basis for the other two tree-based models. We started from the decision tree and found out that it is the best model. More details are in the Section 4. Random forest is the decision tree with bagging. It essentially trains multiple decision trees and lets each of them predict a result. The majority would be considered the prediction result. Since our binary label is not separated equally, the extra bagging may not be effective. We expected that the XGBoost would improve the performance from the decision tree as it is considered one of the most effective supervised learning methods. [4] We also tried the relatively simple decision boundary method, SVM. It is not very suitable in this case, but it could be more time efficient than other tree models. Neural networks can achieve more complicated decision boundaries, but they are harder to train and may create overfitting. We used the basic fully connected neural network, but the more advanced deep learning methods like LSTM and RNN may be effective and would be explored in future work.

4 EVALUATION

4.1 Experimental Setting

The data we used to build this classification model is from dtn01 in 2021. The dataset has 116 features and over 3 million logs. We only used data in 2021 because it is the latest data and thus best represents the behaviors of NERSC transfers right now. We didn't expand the training data span because one year is a very natural chronological cycle. The functionalities, performance, and usages of NERSC change over years, so a model trained on data from years before would not be ideal for predicting current network performance. Also, we don't want to make the cycle even smaller (eg. a month) because we have already achieved respectful performance on yearly basis. We only used dtn01 because it is the first DTN for

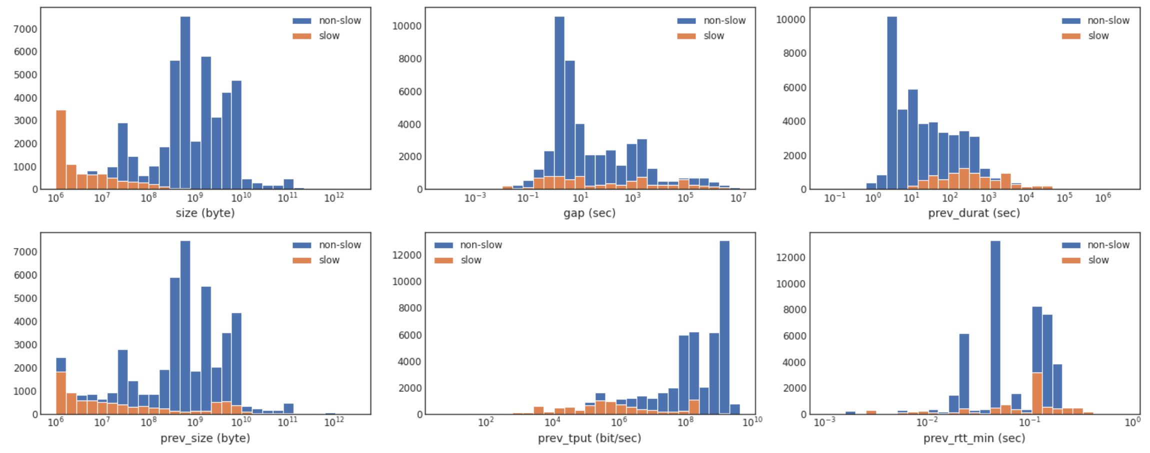


Figure 6: Distribution differences of critical features divided by transfer performance (slow vs. non-slow)

users to remember without specific instructions. It has the most transfers for both low-performance and overall.

As mentioned in the previous section, we did a train-test-split to use the existing data to build the model and test its performance. We use the data from January 2021 to April 2021 as the training set and the May 2021 data as the testing set. When all the 2021 data are available, we'll train our model on a whole year basis.

To measure the prediction performance, we basically refer to the conventional confusion matrix for binary classification, consisting of TP (True Positive), FP (False Positive), FN (False Negative), and TN (True Negative). Intuitively, the fraction of slow connections is small, while the majority of connections would perform normally. Hence, reporting the simple accuracy measure may misguide the audience. We measure the prediction performance using *F1 score*, a harmonic mean of $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. The metric of F1 score is defined as: F1 score $= 2 \times \frac{Precision \times Recall}{Precision+Recall}$. A greater F1 score indicates a better performance in prediction.

4.2 Prediction Performance

The performance of models is evaluated based on the data from the test set. We compare the models numerically with the accuracy, precision, recall, and F1 score. We also visualize the results from the best-performing model to ensure the predictions make intuitive sense.

We start with a baseline model. Intuitively, the previously measured throughput information should have a strong correlation with the current transfer rate. Therefore, to predict the label created based on the throughput of the transfer, the most intuitive model would be the decision tree with the previous transfer's throughput as the only variable without any hyperparameters tuning. Again, the previous transfer refers to the closest previous transfer from the same subnet IP address. In Figure 7, the clear linear relationship is shown between the previous throughput and current throughput, which confirms the effectiveness of `prev_tput`. The baseline model creates a hard linear decision boundary solely based on the previous performance. We observed that the prediction using this dummy model does not go beyond 0.77 (in F1 score). Even though it would be better than a random guess, this prediction performance is far from a perfect model and unacceptable in practice.

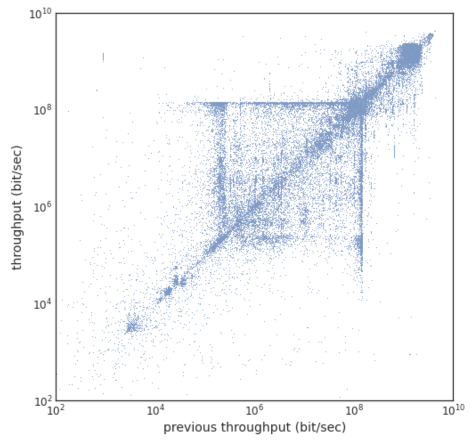


Figure 7: Correlation between previous vs. current transfer throughput: The result shows the clear linear relationship is shown between previous throughput vs current throughput but the baseline model solely relying on the previous throughput information is limited to 0.77 as the prediction performance.

Table 2: Prediction performance with different threshold for slow connections

Threshold	% of low-performance	F1 score
< 100 Kbps	4.9%	0.938
< 1 Mbps	15.5%	0.913
< 10 Mbps	26.7%	0.945

From the baseline model, we improve the decision tree model by selecting the most effective feature combination and applying hyperparameter tuning. With all the features provided in tstat and extra features mentioned in previous sections, we exhaustively tried all the feature combinations. We cover the feature selection in more detail in the next part. We then do hyperparameter tuning using the GridSearchCV. The best parameters are min_samples_split = 3, min_samples_leaf = 2, and max_depth = 10. The F1 score is

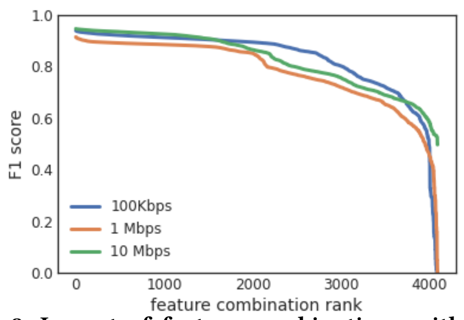


Figure 8: Impact of feature combinations with different threshold ranges for defining slow transfers (100 Kbps, 1 Mbps, and 10 Mbps)

improved to 0.913. We create two other labels using different thresholds to confirm our model is applicable and effective in other settings. As shown in Table 2, the other two thresholds actually create better performance. We are then more confident with our model and feature choices.

Moreover, we confirmed the necessity and effectiveness of our feature engineering and selection. We have proved it to be better than the naive model with only one important feature. The model with only variables provided in tstat which are size and country can only achieve an F1 score of 0.826. Our model boosts it by 0.087, which means our extra features generated from the latest transfer improve the performance substantially. On the other hand, the model with all the features we created can only achieve an F1 score of 0.891. It is a lot worse than our best model after feature selection. Those are features truly influential features.

The xgboost model has similar performance as the decision tree, but it is computationally more costly and takes significantly more time. Try-all feature search is not possible in the gradient boost model, so we only tried the top feature combinations from the decision tree feature selection. We still used GridSearchCV hyperparameter tuning. The F1 score is slightly worse than that from the decision tree at 0.913. This is unexpected as xgboost is considered one of the most effective tree-based model. One explanation is the decision tree has already created one of the best models, and it is hard to improve from it using a gradient method when it has already been close enough to the optimum. From Figure 8, we can see that all the top 2000 combinations create respectful results with any of the three thresholds. This is the same problem that makes the random forest to be even worse. Random forest generates some predictions with high precision but extremely low recall. The resulting F1 score never goes above 0.3.

We also tried some non tree-based methods. SVM is not effective in this case because this classification problem is too complex to be expressed by a single boundary. The features are dependent and have higher order relationships. The F1 score is not close to that of the decision tree models. We also tried fully connected neural network. With moderate efforts on constructing the most fitted model, we did not find a structure that has a clear potential for better performance than the tree based models.

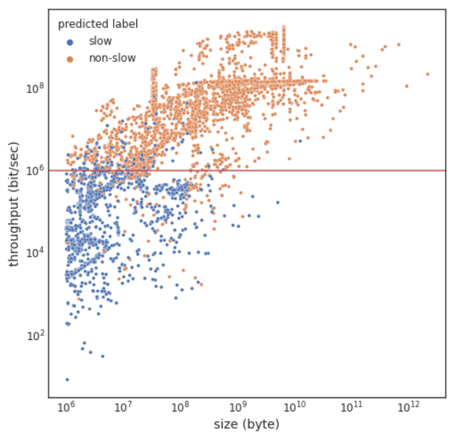


Figure 9: Distribution of prediction result: the blue points under the borderline are correctly predicted, and vice versa. Similarly, the orange points located above the border line are correctly predicted.

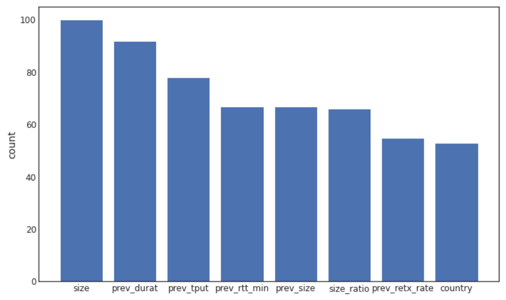


Figure 10: Number of occurrence of the features in Top-100 feature combinations: The transfer size is the essential feature as it appears in all of the 100 best combinations, while the duration and throughput features follow and show their importance.

We found that the model with the best performance and time efficiency is the relatively simple decision tree model with transfer size, previous throughput, previous minimum round trip time, previous transfer size, and size ratio. Figure 9 shows the tput vs size scatter plot of the test set data. All the blue dots above the red line and all the orange dots below the red line are the misclassified ones. There are only a few of them and mostly concentrated in the area close to the red line which represents the boundary we used to determine if a transfer should be labeled as low-performance. This shows our prediction model is valid.

4.3 Impact of Features

Feature selection has been an active research topic over the past decade for identifying higher priority features by eliminating less essential ones [12]. Feature selection is an important aspect of the study not only because it helps improve the performance of the prediction models but also because it brings insights into the factors influencing the performance of network transfers. From the exhaustive feature search, Top-100 feature combinations were selected with the best F1 scores in the decision tree model.

In Figure 10, it presents the number of occurrence of top features in Top-100 combinations. Among them, we find that the transfer size is the essential feature as it appears in all of the 100 best combinations. All the features should be considered important factors influencing the performance of network transfer in our experiment settings.

5 RELATED WORK

Monitoring network traffic is one of the essential tasks in network operations and management. It is also crucial in scientific computing to support ever-increasing data-intensive scientific exploration and computing. In particular, identifying elephant flows has been a critical problem as the flows consume significant amounts of network capacity. A study in [2] introduced an algorithm estimating the traffic volume of individual flows, which is used to detect elephant flows' total byte count. The authors define two hash tables recording a counter representing the volume of the flow with the associated flow ID from the packet trace, which is then used to detect elephant flows showing a pre-defined threshold. In [5], the authors tackled the problem of the classification between elephant (large transfer) flows and mice (small) flows. This previous study takes an unsupervised learning approach and the presented clustering scheme (using GMM/EM) produces two clusters (one for elephant flows and the other for mice flows) from the NetFlow data. While highly important to identify elephant flows, our study focuses on predicting slow connections that significantly impact data-intensive scientific applications.

There have been several studies analyzing tstat data. In [15], the authors presented a classification mechanism to detect the low throughput time intervals. The classification mechanism consists of two phases, the first phase assigning binary classification labels for each time window (either anomalous or not), and the second phase performing actual classification by constructing a supervised learning model using the assigned label information. Another study in [11] performed the evaluation of deep learning models, including Multilayer perceptron (MLP), Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), in order to predict throughput for each time interval. While these studies focused on analyzing tstat data based on time windows, our study focuses on the connection-level prediction.

6 CONCLUSIONS

This study explores `tstat` logs collected on data transfer nodes at NERSC. Our exploration of the network measurement data reveals several interesting findings that could impact to data transfer performance. Based on the findings, we defined a set of features that could be easily computed before start of a file transfer. These features are based on the latest completed transfer between the same source and destination networks, rather than considering the entire historical information that could be highly burdensome to keep track of. In addition, we performed feature selection to uncover better combinations of features and improve prediction performance. With even relatively simple prediction models, we are able to achieve F1 score above 0.91 for three different thresholds from 100 Kbps to 10 Mbps. When the threshold is set to 10 Mbps, our prediction model achieved 0.945 F1 score.

In this study, we chose lightweight predictors (conventional tree-based learning algorithms) with a minimal computational requirement so as to focus on feature engineering. It would be interesting to explore more advanced prediction algorithms such as some deep learning techniques. Another interesting direction for future exploration is to explore additional features including statistics about recently completed transfers, such as, average and standard deviation.

ACKNOWLEDGMENTS

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, and also used resources of the National Energy Research Scientific Computing Center (NERSC).

REFERENCES

[1] A Alekseev, A Kiryanov, A Klimentov, T Korchuganova, V Mitsyn, D Oleynik, A Smirnov, S Smirnov, and A Zarochentsev. 2020. Scientific Data Lake for High Luminosity LHC project and other data-intensive particle and astro-particle physics experiments. In *Journal of Physics: Conference Series*, Vol. 1690. 012166.

[2] Ran Ben Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. 2017. Optimal elephant flow detection. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 1–9.

[3] Thomas Beermann, Olga Chuchuk, Alessandro Di Girolamo, Maria Grigorieva, Alexei Klimentov, Mario Lassnig, Markus Schulz, Andrea Sciaba, and Eugeny Tretyakov. 2021. Methods of Data Popularity Evaluation in the ATLAS Experiment at the LHC. In *EPJ Web of Conferences*, Vol. 251. EDP Sciences, 02013.

[4] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794.

[5] Anshuman Chhabra and Mariam Kiran. 2017. Classifying elephant and mice flows in high-speed scientific networks. *Proc. INDIS* (2017), 1–8.

[6] Bjoern Enders, Debbie Bard, Cory Snavely, Lisa Gerhardt, Jason Lee, Becci Totzke, Katie Antypas, Suren Byna, Ravi Cheema, Shreyas Cholia, et al. 2020. Cross-facility science with the superfacility project at LBNL. In *2020 IEEE/ACM 2nd Workshop on Extreme-scale Experiment-in-the-Loop Computing (XLOOP)*. 1–7.

[7] Alessandro Finamore, Marco Mellia, Michela Meo, Maurizio M Munafo, Politecnico Di Torino, and Dario Rossi. 2011. Experiences of internet traffic monitoring with tstat. *IEEE Network* 25, 3 (2011), 8–14.

[8] Rajkumar Kettimuthu, Zhengchun Liu, Ian Foster, Peter H Beckman, Alex Sim, Kesheng Wu, Wei-keng Liao, Qiao Kang, Ankit Agrawal, and Alok Choudhary. 2018. Towards autonomic science infrastructure: Architecture, limitations, and open issues. In *Proceedings of the 1st International Workshop on Autonomous Infrastructure for Science*. 1–9.

[9] Zhenlong Li, Qunying Huang, Yuqin Jiang, and Fei Hu. 2020. SOVAS: a scalable online visual analytic system for big climate data analysis. *International Journal of Geographical Information Science* 34, 6 (2020), 1188–1209.

[10] Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcón, Marc Solé, Victor Muntés-Mulero, David Meyer, Sharon Barkai, Mike J Hibbett, et al. 2017. Knowledge-defined networking. *ACM SIGCOMM Computer Communication Review* 47, 3 (2017), 2–10.

[11] M Nakashima, A Sim, and J Kim. 2020. Evaluation of Deep Learning Models for Network Performance Prediction for Scientific Facilities. In *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics*. 53–56.

[12] Makiya Nakashima, Alex Sim, Youngsoo Kim, Jonghyun Kim, and Jinoh Kim. 2021. Automated feature selection for anomaly detection in network traffic data. *ACM Transactions on Management Information Systems (TMIS)* 12, 3 (2021), 1–28.

[13] Taylor Reiter, Phillip T Brooks, Luiz Irber, Shannon EK Joslin, Charles M Reid, Camille Scott, C Titus Brown, and N Tessa Pierce-Ward. 2021. Streamlining data-intensive biology with workflow systems. *GigaScience* 10, 1 (2021), giaa140.

[14] Oleg Sukhoroslov. 2021. Toward efficient execution of data-intensive workflows. *The Journal of Supercomputing* 77, 8 (2021), 7989–8012.

[15] Astha Syal, Alina Lazar, Jinoh Kim, Alex Sim, and Kesheng Wu. 2019. Automatic detection of network traffic anomalies and changes. In *Proceedings of the ACM Workshop on Systems and Network Telemetry and Analytics*. 3–10.

[16] Benjamin A Weaver, Michael R Blanton, Jon Brinkmann, Joel R Brownstein, and Fritz Stauffer. 2015. The Sloan digital sky survey data transfer infrastructure. *Publications of the Astronomical Society of the Pacific* 127, 950 (2015), 397.