Modeling Data Transfers: Change Point and Anomaly Detection

Cecilia Dao*, Xinyu Liu†, Jiming Jiang‡, Alex Sim§, Craig E. Tull§, and Kesheng Wu§
*Yale University, cecilia.dao@yale.edu
†University of California at Berkeley, xinyu_liu@berkeley.edu
‡University of California at Davis, jimjiang@ucdavis.edu
§Lawrence Berkeley National Laboratory, {asim, cetull, kwu}@lbl.gov

***Abstract*—To help the operations and resource planning of a large experimental facility, we model the time needed for transferring the data files produced by the facility to a computer center, with the goals of predicting expected file transfer time and identifying unusually slow transfers that might require attention from human operators. The file transfer time can be thought of having two parts: a base time depending on the hardware and software involved, and a congestion part due to uncontrollable interferences from other operations on the shared resources including network links, disk storage systems, and CPU involved in the transfers. Since many parameters important to the base time are not available to us, we employ a change point detection algorithm to separate the data records into time periods (called segments) with relatively stable behavior. Within each segment, we apply a non-parametric model to describe the congestion time. When the observed file transfer time is significantly longer than typical expected time, say, above 4 interquartile ranges beyond the expected values, we declare the particular file transfer to be *unusually slow*. When many of these unusually slow file transfers are observed, it is worthwhile to notify the human operator to investigate the abnormal behavior of the system.**

I. Introduction

A. Motivation

The Department of Energy's (DOE) Office of Science operates scientific user facilities serving 10,000's of researchers each year, providing them access to fast detectors, high brightness accelerators, automated experimental techniques, and capabilities for *in situ* and *in operando* experiments. At Berkeley Lab, the Advanced Light Source (ALS) is a synchrotron x-ray light source facility that provides users from around the world access to the brightest beams of soft x-rays, together with hard x-rays and infrared, for scientific research and technology development in a wide range of disciplines [1] [2]. The ALS allows scientists to probe matter with unprecedented spatial and time resolution, but improvements in accelerator and detector technology have led to an increasing data and computing challenge for the facility and scientist users. For decades, ALS users have relied upon a grab-and-go data management model, which has been outpaced by the beyond-Moore's Law growth of data from the instruments. At the same time, the number of users have grown and requirements for real-time analysis to provide feedback for experiments have increased the computational challenge facing ALS researchers. Staff at the ALS, ESnet, and Berkeley Lab's Computing Research Division (CRD) have collaborated to design, develop, and deploy a real-time data management, data processing, and data visualization system called SPOT Suite that gives users of select beamlines fast feedback while conducting experiments at the ALS. One aspect of the SPOT Suite is automated, real-time transfer of data over ESnet (Energy Sciences Network) to the NERSC (National Energy Research Scientific Computing) facility for real-time processing. This vital link in the end-to-end data flow from experimental instrument to first results occurs over shared networks and involves multiple computing resources controlled by different administrative domains. The consequence is that real-world effects of resource contention, hardware and/or software failures and changes, etc. can have detrimental effects on the flow of data. This paper describes the building of a real-time system that can detect data transfer delays and alert administrators to help expedite the recovery of scientific data flow.

B. Data

The dataset we use contains 39,761 file transfer observations from a single ALS beamline to NERSC over a long time period (4.5 years), as the computer and networking hardware and software stacks at both ends changed over this time period. Each observation has the following variables: Bundle (transfer file name), Size (bits), Start Time (of file transfer), and End Time (of file transfer). The data spans from 03-06-13 to 08-23-17, and Table 1 shows some descriptive statistics.

STATS	size (bits)	transfer time (s)
mean	4.178e+10	54.979
std	5.909e+10	172.0275
min	1.062e+04	0.3840
max	1304.142128	15695.385

TABLE I: Descriptive statistics of the data

We created the variables Transfer Time (seconds) and Transfer Rate (bits/seconds) for analysis purpose. For simplicity, we removed overlapping file transfers (about 2% of the data) and failed or incomplete file transfers. The resulting records of file transfers are assumed to be independent from each other. During the exploration phase of this work, we have explored the options of detecting daily and weekly patterns, but were not able to detect obvious patterns. In this work, we therefore adopted this independence assumption.

We began by investigating transfer rate changes over time. Figure 1 shows that the transfer rate has significant drop or rise for the past four years corresponding to significant changes in the network connections and machines at both ends of the file transfers. However, as mentioned before, our data records do not contain information about these changes. It is critical for us to detect these significant changes before building any performance models.

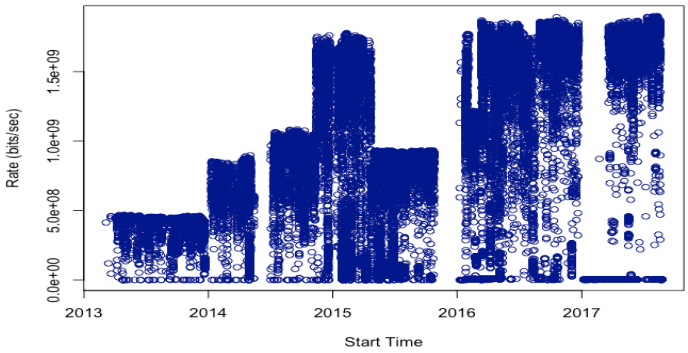

Fig. 1: Transfer Rate over Start Time

Generally we expect larger files to take more time to transfer. Next, we plot the relationship between Size and Transfer Time in Figure 2. We notice that the values of both the Size and Time span many orders of magnitudes. Based on past experiences, we believe that working with *log* of these values would help reduce the skewness of the data distribution and improve the effectiveness of common prediction algorithms.

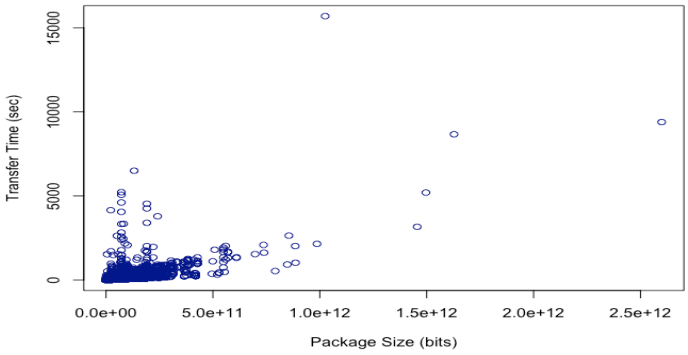

Fig. 2: Transfer Time according to Size

C. Related Work

Related past work has used machine learning or regression methods to predict file transfer time. [3] used machine learning techniques to improve `scp` estimated time of arrive prediction, claiming that support vector regression and extended linear regression performed the best. [4] also provided evidence for support vector regression having greater prediction accuracy compared to the exponentially weighed moving average history-based predictor, using file size and bandwidth as covariates. [5] used regression methods to predict the performance of `GridFTP` large transfer files, and noted that sliding-window variants capture trends in throughput better than statistical summaries such as the mean and median. They also observed that higher accuracy occurs when information on current system/network trends were included. File size has been well known to greatly affect file transfer time. Interestingly, [6] found that almost no correlation existed between file size and transfer time when the file size was small. But for large file sizes, file size and transfer time increasingly correlated well with one another as file size increases; however, using only file size to predict transfer time did not produce highly accurate results. We used these findings to motivate our work, especially in the change detection in Section 2 and anomaly detection in Section 3. None of the aforementioned work took measures to ensure that the predicted rate did not exceed the maximum bandwidth (i.e., the maximum transfer rate possible). To make the prediction more realistic, we took that restriction into account in our prediction process after estimating the maximum bandwidth in the change detection algorithm.

D. Contribution

Though our work was driven by the file management needs of one specific scientific user facility, we believed the performance data records from many different type of applications will have similar trends - the performance would have a guaranteed upper bound (or minimum time) with an unknown part that could be modeled as a random process. In modeling the file transfer time, we had a base time that was determined by the network connection, the storage systems and the CPU of the machines involved in the transfers, plus a time due to congestion from other applications using one or multiple components involved in a file transfer. Our work demonstrates that decomposing the performance model this way is an effective approach.

In our work, we proposed a change point detection algorithm based on the maximum file transfer performance for a given time window. This approach matched well with the characteristics of the data and could potentially be useful in different applications.

We explored the characteristics of the distribution of contention in the file transfer time and eventually decided that the non-parametric kernel method was effective in capturing the distribution of the observed congestion process. Tests shown that this approach led to an effective way of classifying unusually slow file transfers. When multiple unusually slow file transfers occurred during a narrow time window, this situation might be due to a symptom of a severe system issue that requires attention from a system administrator. In which case, we need to alert the administrators to investigate such issues.

This work was based on a set of observations from an active scientific facility. Thus the work has real impact on the operations of a scientific facility with thousands of users. In addition, our performance model could also help with the

resource planning for the expansion of SPOTS suite to more beamlines in the future.

The remaining of this paper is organized as follows. Section 2 describes the change point detection algorithm and results. Section 3 describes the transfer time prediction algorithm and results. Combining the two methods in an online fashion will enable us to identify files that take unusual length of time.

II. Change Point Detection

A. Algorithm

The change point detection algorithm intends to detect changes in the network connections, the storage systems or the computer systems involved in the data transfer operations. The changes in these system components are reflected in the changes in the base time, which can be thought of as the best-case performance. In this work, we proposed to use the maximum bandwidth to detect these changes.

As shown in Fig 1, the transfer rate drastically rises and drops, as the system periodically upgrades or changes. The time period between two change points has relatively stable maximum performance and is considered as one segment for performance modeling.

We assumed each segment follows the same model. With this assumption, we can build a performance model for each segment of the data, and then use the performance model to make prediction and detect anomalies for each segment separately.

Figure 3 is the transfer rates plot with colors based on the file sizes. Note that the transfer rate only reaches the maximum bandwidth given large enough file sizes. Therefore, for all the following analyses in change point detection, we only used the data records where the file sizes are large enough, say, more than 1GB.

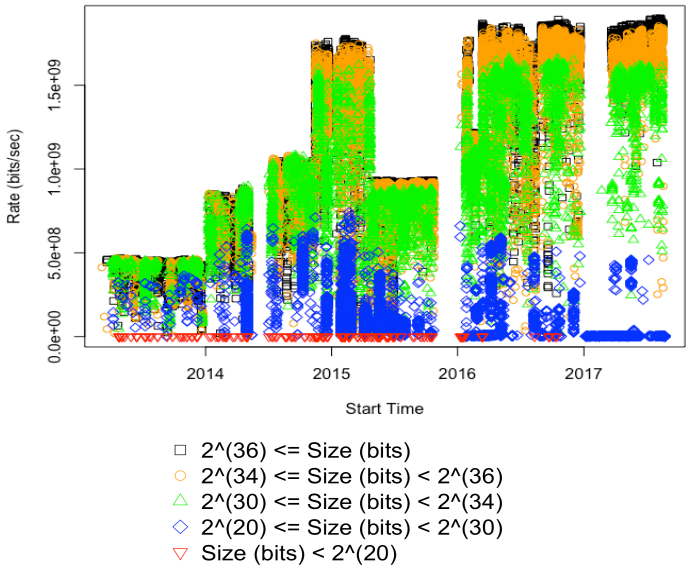

Fig. 3: Transfer Rate over Start Time,color coded based on Size

We proposed to compute the moving maximum, with window size h, to determine change points. We will declare a change point under one of these conditions:

- The absolute value of the next moving maximum value and the previous moving maximum value is greater than or equal to a.
- The time difference between two consecutive observations is at least $break\_day$ days.

We also have the option to set the width of each segment to have at least ch_width observations, which can tune the detection sensitivity. Based on prior knowledge of our ALS data and Figure 3, we set the parameters as $h = 100$, $s = 2.5e+08$ (bits), $a = 8e+09$ (bits), $break\_day = 21$ (days, 3 weeks based on the shut-down breaks), and $ch\_width = 700$.

Algorithm 1 Change Detection

for *each file* i **do**
  $s\_i \leftarrow$ *transfer size for file* i
  $r\_i \leftarrow$ *transfer rate for file* i
  if $s\_i \leq s$ **then**
    Skip i
  end if
  if $i \leq h$ **then**
    Preparing for detection
  else
    $M\_i \leftarrow max(r\_i, r\_\{i-1\}, ...r\_\{i-h\})$
    $M\_i - 1 \leftarrow max(r\_\{i-1\}, ..., r\_\{i-h-1\})$
    if *file* i *has more than* $break\_day$ *days apart from file* $i-1$ **then**
      declare i as a change point
    end if
    if abs$(M_i - M_\{i-1\}) \geq a$ and $i \geq ch\_width$ **then**
      declare i as a change point
    end if
  end if
end for

B. Results

We applied this algorithm to the entire dataset and extracted seven change points.

In Figure 4, segments are separated by colors. The observations with the same segment have the same color, and the color changes when a new change occurs with a vertical line crossing each change point.

We were able to look up the causes of the changes in the system logs for the seven changes. Here are the reasons we were able to locate:

- 2014-01-09 23:47:11 - DTN OS upgrade
- 2014-07-03 01:12:12 - Resume ALS operation after network upgrade
- 2014-11-14 18:46:34 - Science DMZ network configuration upgrade
- 2015-04-29 20:34:42 - NFS-mounted file server shared with 2nd beamline DTN

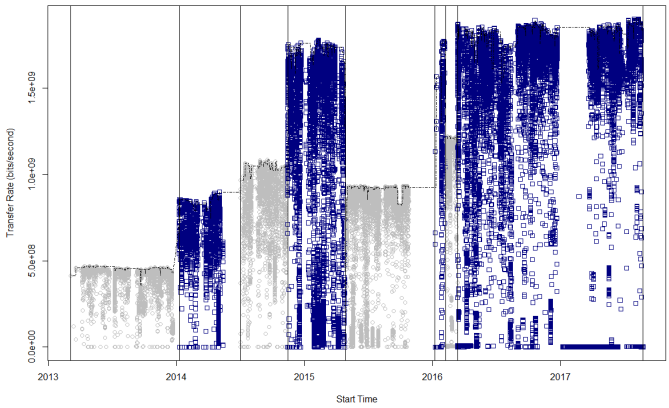

Fig. 4: Transfer Rate over Start Time. The moving maximum line runs across the top, and vertical lines at the change points. Each segment contains observations with the same color.

- 2016-01-08 20:00:00 - 2nd beamline DTN disconnected from NFS file server
- 2016-02-08 07:02:14 - Unknown perturbation
- 2016-03-12 16:15:13 - Restore to baseline performance

We tried a plethora of change-point detection methods, including the Pruned Exact linear Time [7] and Breakout Detection [8]. These were either overly sensitive to change detection or provided unnatural breaks. Our proposed change detection is simple and intuitive based on exploratory plots. More importantly, we were able to determine the causes of these change points, which further confirmed the effectiveness of our moving maximum based change point detection approach.

III. Prediction

A. Algorithm

Earlier, we discussed that the observed data transfer time can be broken into two components: a base time plus a congestion part. Based on this understanding, we designed an online prediction process consisting of two major steps: determine the minimum transfer time (i.e., base time) and then model the anticipated transfer time being the base time plus a random process caused by congestion. This process will guarantee realistic prediction where the predicted transfer time will not lead to transfer rate that surpasses the maximum bandwidth imposed by the system components involved. Due to space limitation, we only described the prediction on the two most recent segments: Segments 7 and 8.

Figure 5 shows the transfer time against file size for those segments. For ease of prediction, we removed the skewness of the data by applying $\log_2$ transformation to the transfer time and file size, as seen in Figure 6 . Figures 5 and 6 illustrate the differences in distribution and minimum baseline between the two segments.

To capture the minimum baseline curve of the transfer time (with respect to the file size) in each segment, we used B-spline minimum quantile regression with 3 degrees of freedom [9] [10], which corresponds to the maximum bandwidth in the segment. Let the j^{th} minimum base of the transfer time in the i^{th} segment be $m_{ij}(\text{Size}_j)$. Define $\alpha_{ij} > 0$ as the difference between $\log_2(\text{Transfer Time}_j)$ and $\log_2(m_{ij}(\text{Size}_j))$ for the j^{th} observation in the i^{th} segment.

We then used nonparametric kernel regression [11] to estimate $\log_2(\alpha_{ij})$ (to ensure that $\alpha_{ij} > 0$) and then re-transformed it to get the estimate of α_{ij}. Covariates included to predict $\log_2(\alpha_{ij})$ are $\log_2(\text{Size}_j)$ and Epoch Time_j. We included epoch time since we assumed recent events would be more similar to one another; perhaps a clog in the system occurred at certain times, which could affect file transfers in the same neighborhood of time. Then, the j^{th} predicted transfer time in the i^{th} segment is $\hat{y}_{ij} = \hat{m}_{ij}(\text{Size}_j) + \hat{\alpha}_{ij}(\log_2(\text{Size}_j), \text{Epoch Time}_j)$. The predicted rate would then be $\hat{r}_{ij} = \dfrac{\text{Size}_j}{\hat{y}_{ij}}$, which aimed to have similar shape of the actual rate. To expedite the prediction procedure, we considered using the most recent k file transfers to train the data, as the difference in results is empirically minimal for Segment 7. For the last two segments, we selected $k = 300$ since the maximum number of consecutive file sizes is 281.

Algorithm 2 Prediction

$y\{ij\} \leftarrow$ transfer time for file j in Segment i
$r\{ij\} \leftarrow$ transfer rate for file j in Segment i
Apply BSpline minimum quantile regression to estimate $m\{ij\}$
$\log_2(\alpha_{ij}) \leftarrow \log_2(y\{ij\}) - m\{ij\}$.
In non-parametric kernel regression, use covariates $\log_2$ of file size and Epoch Time to predict $\log_2(\alpha_{ij})$
$\hat{y}_{ij} = \hat{m}_i(\text{Size}_j) + \hat{\alpha}_{ij}(\log_2(\text{Size}_j), \text{Epoch Time}_j)$
$\hat{r}_{ij} = \dfrac{\text{Size}_j}{\hat{y}_{ij}}$

B. Result

Figure 7 shows the final prediction for expected transfer rate, where the shape confirms well to the data and leans towards the maximum transfer rate, which is expected under minimal disruptions of the system. Note that the prediction also confirms well to expected low transfer rates when the file sizes are small. We have found that parametric and zero-inflated models [12], such as generalized linear model under the Tweedie distribution [13], to estimate $\log_2(\alpha_{ij})$ did not provide the natural shape of the actual data due to unmet restrictive parametric assumptions.

C. Anomaly Detection

We aimed to identify alerts to inform the administrators about anomalies or failures in the system so they can be resolved. In contrast to large change detection in Section 2, these are usually small and short-term changes. Note that the expected transfer rates are skewed towards the maximum bandwidth, as expected under normal amounts of congestion.

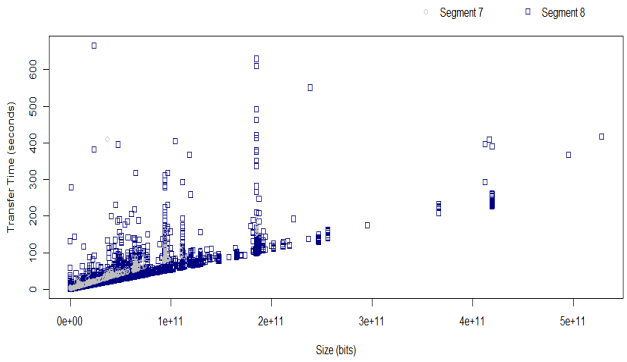

Fig. 5: Segments 7 and 8: Transfer Time according to Size

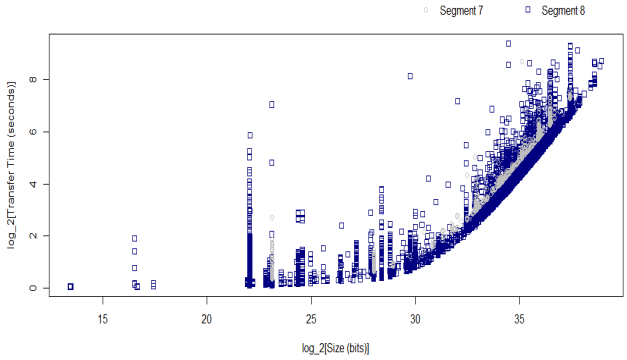

Fig. 6: Segments 7 and 8: $\log_2$(Transfer Time) according to $\log_2$(Size)

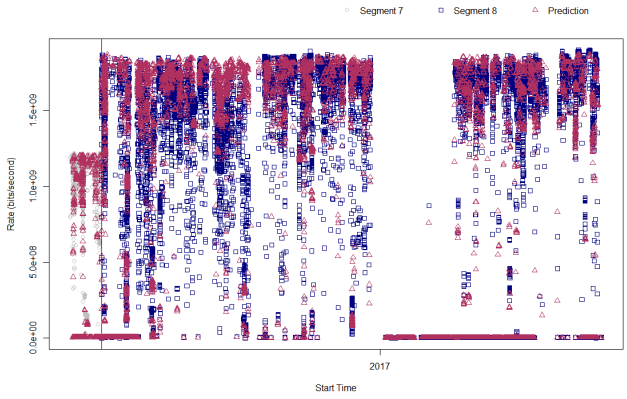

Fig. 7: Segments 7 and 8: Prediction results for Transfer Rate according to Start Time

Therefore, we considered an outlier detection method that adjusts the boxplot to include a robust measure of skewness when defining the whiskers [14]. This skewed-adjusted boxplot can perform automatic outlier detection without any parametric assumptions. We expect large transfer files to reach maximum bandwidth, so we focus on file sizes that are at least 2^{36} bits (this parameter can be changed based on empirical evidence in the data) as represented in black in Figure 3. If the prediction error of a substantial amount of files with at least 2^{36} bits lies beyond the fence boundaries of $4 \times$ (Interquartile Range) within a consecutive or short amount of time, then an alert is signaled to the administrators to improve the maintenance of the system. The fence boundaries were chosen as such because as a rule of thumb, observations beyond the fence when $b \geq 3$ for $b \times$ (Interquartile Range) are considered extreme outliers, and we aimed to find distinctly extreme anomalies.

Figure 8 shows the prediction errors along with identification of files that have sizes at least 2^{36} bits in black and files with lower sizes in light blue. The fence line is in red. We see that an alert occurred on September 18, 2016 continuously for 3 days with the observations circled in red.

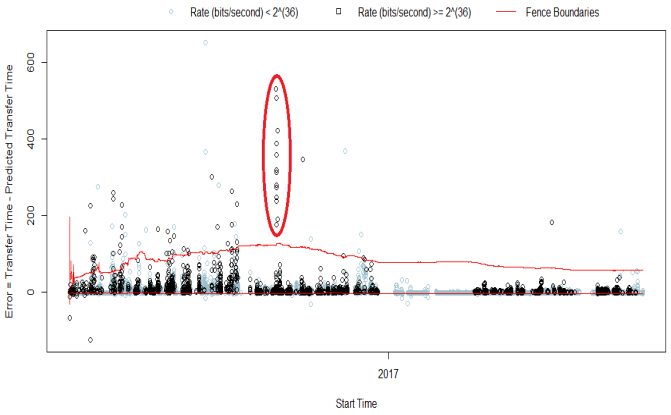

Fig. 8: Segment 8: Anomaly Detection applied on Prediction Errors according to Start Time

Figure 9 shows the detailed of those dates in Figure 3. In the red circle, a cluster of the transfer rates for those days are significantly lower than expected based on their large file sizes. Further exploration in the outage log shows the following occurred: "2016-09-18 for a few hours: System in degraded mode. Engineers are performing emergency maintenance to apply security updates. There may be network interruptions during this time."

Therefore, the alerts based on anomaly detection can fix issues early and prevent prolonged outages.

IV. Summary

Like in many applications, the file transfer performance we seek to model in this work has a guaranteed no-to-exceed performance limit. Since many of the system components are shared among many users, the actual performance is typically lower than the theoretical maximum. To capture these characteristics, we propose a hybrid performance model that

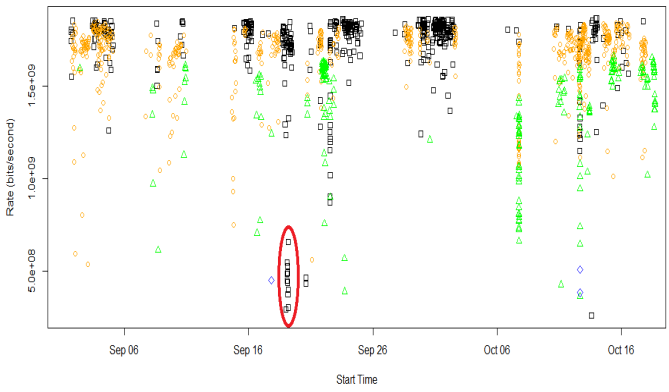

Fig. 9: Zoomed in Figure 3

captures these two elements: a base time plus a congestion time.

To simplify the modeling process, we first employ a change point detection algorithm to separate the data records into periods of relatively uniform performance. Within each segment of these time periods, we apply a prediction model for predicting performance and identifying anomalies. For this change point detection, we propose an algorithm based on the moving maximum of data transfer rates. Our approach is validated to be effective because we successfully found explanations for those change points being identified. By combining both models into one online algorithm, we can predict transfer rates per file based upon recent end-to-end system performance. This permits us to identify large and small changes which may correspond to macroscopic configuration changes or to performance anomalies caused by components along the data flow. The novelty in our work lies in respecting the maximum bandwidth baseline in both change and prediction.

We are in the process of transferring the research code described earlier to be used in the production system. Additionally, we plan to update the algorithms to accommodate higher accuracy for file transfer performance with small sizes, and consider transfer overlaps.

ACKNOWLEDGMENT

The authors gratefully acknowledge data support from Simon Patton. This work was supported by the Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center.

REFERENCES

[1] J. Blair, R. S. Canon, J. Deslippe, A. Essiari, A. Hexemer, A. A. MacDowell, D. Y. Parkinson, S. J. Patton, L. Ramakrishnan, N. Tamura *et al.*, "High performance data management and analysis for tomography," in *Developments in X-Ray Tomography IX*, vol. 9212. International Society for Optics and Photonics, 2014, p. 92121G.

[2] J. Deslippe, A. Essiari, S. J. Patton, T. Samak, C. E. Tull, A. Hexemer, D. Kumar, D. Parkinson, and P. Stewart, "Workflow management for real-time analysis of lightsource experiments," in *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science*. IEEE Press, 2014, pp. 31–40.

[3] D. Del Testa, M. Danieletto, and M. Zorzi, "Applying machine learning techniques to a real cognitive network: File transfer etas prediction," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[4] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1. ACM, 2007, pp. 97–108.

[5] M. Sharma and J. W. Byers, "How well does file size predict wide-area transfer time?" in *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, vol. 3. IEEE, 2002, pp. 2160–2164.

[6] S. Vazhkudai and J. M. Schopf, "Using regression techniques to predict large data transfers," *The International Journal of High Performance Computing Applications*, vol. 17, no. 3, pp. 249–268, 2003.

[7] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[8] N. A. James, A. Kejariwal, and D. S. Matteson, "Leveraging cloud data to mitigate user experience from breaking bad," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3499–3508.

[9] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[10] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of economic perspectives*, vol. 15, no. 4, pp. 143–156, 2001.

[11] J. Racine and Q. Li, "Nonparametric estimation of regression functions with both categorical and continuous data," *Journal of Econometrics*, vol. 119, no. 1, pp. 99–130, 2004.

[12] A. F. Zuur, E. N. Ieno, N. J. Walker, A. A. Saveliev, and G. M. Smith, "Zero-truncated and zero-inflated models for count data," in *Mixed effects models and extensions in ecology with R*. Springer, 2009, pp. 261–293.

[13] G. K. Smyth and B. Jørgensen, "Fitting tweedie's compound poisson model to insurance claims data: dispersion modelling," *ASTIN Bulletin: The Journal of the IAA*, vol. 32, no. 1, pp. 143–157, 2002.

[14] M. Hubert and E. Vandervieren, "An adjusted boxplot for skewed distributions," *Computational statistics & data analysis*, vol. 52, no. 12, pp. 5186–5201, 2008.