ADAPTIVE TRANSFER ADJUSTMENT IN EFFICIENT BULK DATA TRANSFER MANAGEMENT FOR CLIMATE DATASET

Alex Sim[1], Mehmet Balman[1], Dean Williams[2], Arie Shoshani[1], Vijaya Natarajan[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[2]Lawrence Livermore National Laboratory, Livermore, CA, USA
{asim,mbalman,ashoshani,vnatarajan}@lbl.gov, williams13@llnl.gov

ABSTRACT

Many scientific applications and experiments, such as high energy and nuclear physics, astrophysics, climate observation and modeling, combustion, nano-scale material sciences, and computational biology, generate extreme volumes of data with a large number of files. These data sources are distributed among national and international data repositories, and are shared by large numbers of geographically distributed scientists. A large portion of the data is frequently accessed, and a large volume of data is moved from one place to another for analysis and storage. A challenging issue in such efforts is the limited network capacity for moving large datasets. A tool that addresses this challenge is the Bulk Data Mover (BDM), a data transfer management tool used in the Earth System Grid (ESG) community. It has been managing massive dataset transfers efficiently in the environment where the network bandwidth is limited. Adaptive transfer adjustment was studied to enhance the BDM to handle significant end-to-end performance changes in the dynamic network environments as well as to control the data transfers for the desired transfer performance. We describe the results from our hands-on data transfer management experience in the climate research community. We study a practical transfer estimation model and state our initial results from the adaptive transfer adjustment methodology.

KEY WORDS

Bulk data movement, Climate datasets, Earth System Grid, Adaptive transfer adjustment, Transfer estimation model

1. Introduction

Data intensive applications and experiments such as astrophysics, climate modeling, combustion, high energy and nuclear physics, nano-scale materials science and computational biology, is expected to generate exabytes of data over the next 5-10 years, which must be transferred, visualized, and analyzed by geographically distributed teams of researchers. The large amount of data must be continuously moved from the data source repositories to scientists and to analysis, visualization, and storage facilities. The Earth System Grid (ESG) [1] is one of the communities that faces the difficult challenge of managing the distribution of massive datasets to thousands of scientists around the world. An important new collection of climate datasets, referred to as the “replica centralized archive (RCA)”, is expected to comprise 1.2 petabytes (PB) during the Intergovernmental Panel on Climate Change (IPCC) Fifth Assessment Report (AR5) in 2011. The amount of data collected and produced is expanding at a staggering rate, and projected to exceed hundreds of exabytes by 2020 [2]. It takes 100 Gbps end-to-end bandwidth to move one petabyte in a day, and an additional 10,000 times of performance increase is needed for 100 exabytes in 2020. The ESG and others have recognized that the new centralized data and future datasets can only be efficiently served to researchers around the world by replicating it to sites closer to them [3]. To move data replicas efficiently, the ESG has developed a data transfer management tool called the Bulk Data Mover (BDM) [4] [10]. The BDM is responsible for the successful replication of large datasets, and achieves high performance using a variety of techniques. The performance of the BDM is controlled by application-level parameters such as the number of concurrency transfers and the number of parallel streams. A higher preset on these transfer parameters may overload the storage and network capacity, and could result in overall performance decrease. Adaptive transfer adjustment is essential to handle the dynamics of the shared network environments as well as to optimize the BDM data transfers. The adaptive transfer management in BDM contributes to utilizing the fully available network and storage bandwidth as well as to control the end-to-end data transfers for the desired transfer performance.

2. Background

2.1 Earth System Grid

As the climate community makes its first steps towards building a “science gateway” - a data access and analysis system open to everyone - the “Earth System Grid” (ESG) is central to the current and future infrastructure that enables the large federated enterprise system for the dissemination and management of extreme scale climate resources. ESG provides climate resources such as data, information, models, analysis and visualization tools, and other computational capabilities for data management and diagnosis. The ESG project’s goals are (1) to make data more useful to climate researchers by developing Grid technology that enhances data usability; (2) to meet specific needs which national and international climate projects have for distributed datasets, data access, and data movement; (3) to provide a universal and secure web-based data access portal for broad-based multi-model data collections; and (4) to provide a wide-range of Grid-enabled climate data analysis tools and diagnostic methods to climate communities [5] [11]. Thus, ESG is working to integrate distributed data and computers, high-bandwidth wide-area networks, and remote computing using climate data analysis tools in a highly collaborative problem-solving environment.

Since production began in 2004, the ESG has hosted and distributed significant and often very large data collections for many well-known efforts in climate science. The ESG production system currently has over 20,000 registered users. ESG manages approximately 270 TB of model data, comprising the contents of archives at five sites around the U.S. ESG users have downloaded more than 1PB of data.

2.2 Bulk Data Mover

Climate datasets are characterized by large volume of files with extreme variance in file sizes. BDM as a high-level data transfer management component handles the issue of large variance in file sizes and a big portion of small files by managing the file transfers with optimized transfer queue and concurrency management algorithms. The BDM achieves high performance using a variety of techniques, including multi-threaded concurrent transfer connections, data channel caching, load balancing over multiple transfer servers, and storage I/O pre-fetching. Logging information from the BDM is collected and analyzed to study the effectiveness of the transfer management algorithms.

The BDM can accept a request composed of multiple files or an entire directory. The request also contains the target site and directory where the replicated files will reside. If a directory is provided at the source, then the BDM will replicate the structure of the source directory at the target site. The BDM is capable of transferring multiple files concurrently as well as using parallel TCP streams. The optimal level of concurrency or parallel streams depends on the bandwidth capacity of the storage systems at both ends of the transfer as well as achievable bandwidth on the wide-area network. Setting up the optimal level of concurrency is an important issue, especially in climate datasets, because of the many small files. Concurrency that is too high becomes ineffective (high overheads and increased congestion), and concurrency that is too low will not take advantage of available bandwidth. A similar phenomenon was observed when setting up the level of parallel streams.

3. Efficient Transfer Management

3.1 Concurrent transfers and data streaming

When the datasets consist of a mixture of large and small files such as the climate datasets, it is not simple to maximize the transfer performance with a prefixed number for concurrency and parallel streams. The typical file size distribution in climate dataset in Intergovernmental Panel on Climate Change (IPCC) Coupled Model Intercomparison Project, phase 3 (CMIP-3) indicates that most of the data files have less than 200MB of file size, and among those smaller files, file sizes less than 20MB have the biggest portion. Using parallel streams, in general, improves the performance of datasets with large files, and the pipelining technique in GridFTP transfer protocol [9,21] improves the performance of datasets with lots of small files within the transfer connection. However, when the file size is less than a certain threshold based on the available network bandwidth, parallel streams can decrease the performance of the file transfer.

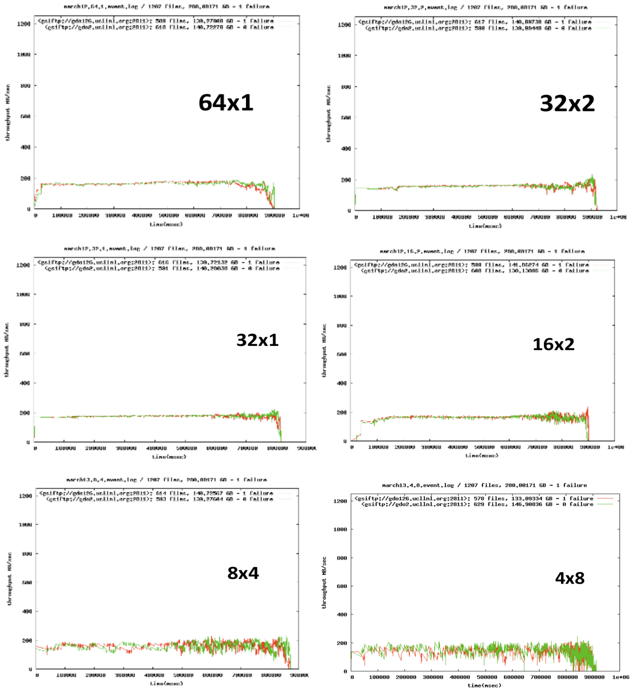


Figure 1: Climate data replication from LLNL to NERSC over shared network. GridFTP transfers of climate dataset from two sources at LLNL to one destination at NERSC show throughput history over time in seconds on different transfer properties.

Figure 1 shows a typical climate dataset transfer over a shared network. It shows transfer throughput performance from two data sources at Lawrence Livermore National Laboratory (LLNL) to one destination at National Energy Research Scientific Computing Center (NERSC) over time in seconds with different level of concurrency and number of parallel streams.

BDM creates concurrent transfer connections, and have files streaming through the connections with a certain number of parallel streams. In Figure 1, BDM managed throughput performance in the climate datasets almost the same in transfers with different parameters, but transfers with less parallel streams show more consistency in file transfer rates throughout the request. For example, the transfers with 4 concurrency and 8 parallel streams per data source (the plot with 4x8) have the same number of total streams 64 (4 concurrency x 8 parallel streams x 2 data sources) as the transfers with 32 concurrency and 1 parallel stream (the plot with 32x1), but it shows more consistent transfer rates with 1 parallel stream. It indicates that the parallel streams do not have much effect in the transfer performance for this type of datasets.

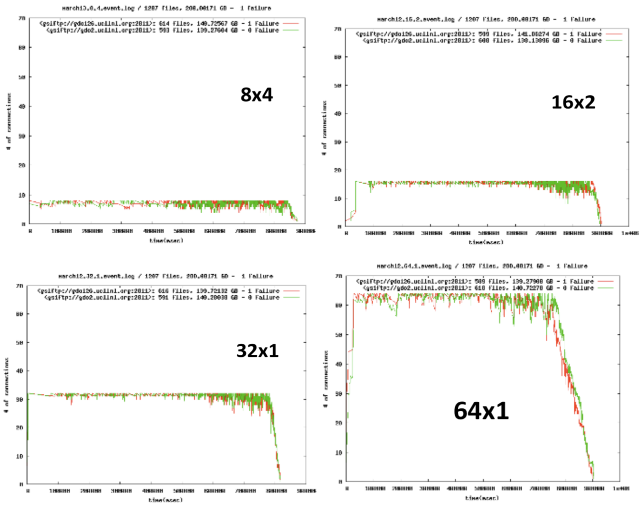


Figure 2: Climate data replication from LLNL to NERSC over shared network showing concurrent GridFTP transfers and load balancing over multiple data sources

Multi-threaded concurrent connections and file streaming which open and maintain *N* different transfer connections and having *N* different files streaming through at the same time, has shown to improve the performance of datasets specially with the mixture of large and small files. An important concern is to keep the network pipe always full. Each connection should maintain the file streaming without any gaps between file transfers. Figure 2 shows the number of concurrent transfers over time in seconds with different parameters. They are from the same transfer runs from Figure 1. It shows that BDM maintains the level of concurrency throughout the transfer run without gaps between file transfers. This feature, high density of data flows in concurrent transfers, is achieved by maintaining transfer queue and storage I/O pre-fetching.

3.2 Balanced transfer connections

When multiple transfer sources are available, transfer connections can be balanced, and the overall throughput performance to the destination can be increased. The balancing algorithm can be as simple as round robin over multiple transfer sources or based on the available bandwidth for each transfer source. BDM manages concurrent connections in mixture of round robin and total file sizes in the transfer queue per connection. BDM transfer queue management module assigns files to transfer queue for each concurrent transfer connection, and when it detects the total sizes of the files waiting in the queue is more than the certain threshold, the connection does not get any more files assigned until file transfers are completed in that particular connection. In that way, each transfer connection maintains similar byte sizes in its transfer queue to other transfer connections, but not the similar number of files unless the files are all in similar sizes. Figure 1 and Figure 2 show transfers from two data sources (one shown in green and another shown in red), and number of total concurrent transfers and cumulative throughput are very similar for two data sources.

3.3 Transfer Queue Management

Transfer queue management and concurrent connection management contribute to more transfer throughput, including both network and storage. When the dataset has a large variance in the file sizes, continuous data flow from the storage into the network can be achieved by pre-fetching data from storage on to the transfer queue of each concurrent transfer connection. This overlapping of storage I/O with the network I/O helps improve the transfer performance.

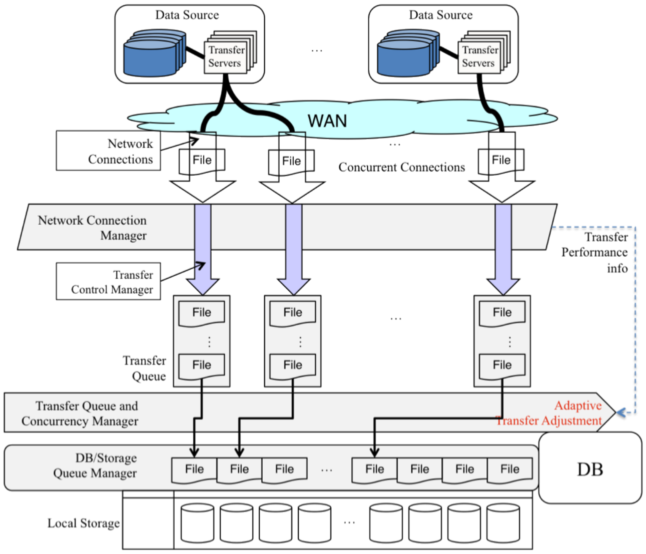


Figure 3: Transfer and concurrency management in BDM, showing adaptive transfer adjustment

As in Figure 3, BDM manages a DB queue from the concurrent transfer connections, and also manages the transfer queues for concurrent file transfers. Each transfer queue checks the configurable threshold for the queued total files size and gets more files to transfer from the DB queue when the queued total files size goes below the configured threshold. Default threshold is set to 200MB based on the file size distribution as discussed in section 3.1.

Storage I/O pre-fetching includes inode creation for writing files at the destination. In many file system cases, many inode creations at the same time cause a significant overhead in file system performance, and this overhead affects the transfer performance. By creating inodes at the destination paths when files are being on the transfer queue, BDM achieves faster storage I/O during the transfers.

Figure 4 shows another climate data transfers from LLNL to NERSC for 4.8TB of a climate dataset from two source servers to one destination. Transfer throughput was consistent most of the time throughout the request, as expected. In the middle of the dataset transfers, low performance was detected, as shown in the middle of the plot, but the number of concurrency was still at 64 all together. This caused each concurrent connection performance to be much lower, and may have caused packet loss too. The dynamic transfer adjustment can help this case in minimizing overhead of slow data transfers during the low performance period, and the BDM can reduce the number of concurrent transfers to maximize the per-connection throughput which could maximize the resource usability during those time.

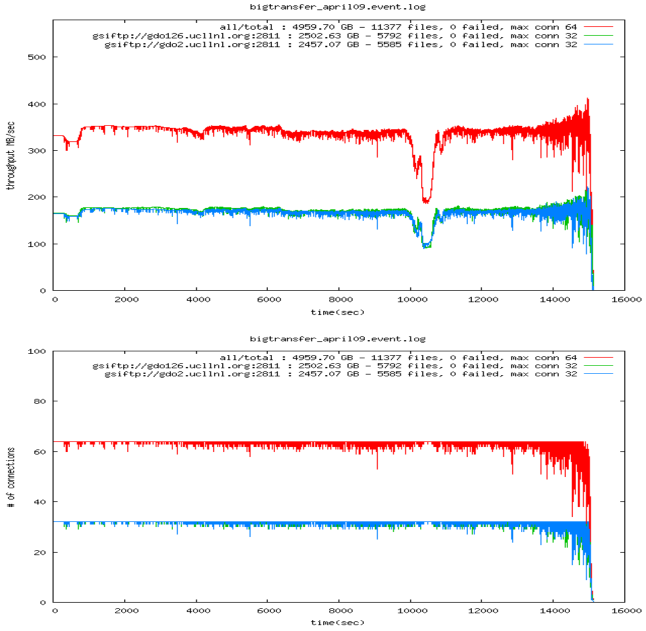


Figure 4: Climate data replication from LLNL to NERSC over shared network. Transfers from 11208 files in 4.8TB of climate dataset from two sources at LLNL to one destination at NERSC with 32 concurrency and 1 parallel stream for each data source show throughput history over time in seconds on the top and the number of concurrency over time in seconds on the bottom.

4. Adaptive Transfer Adjustment

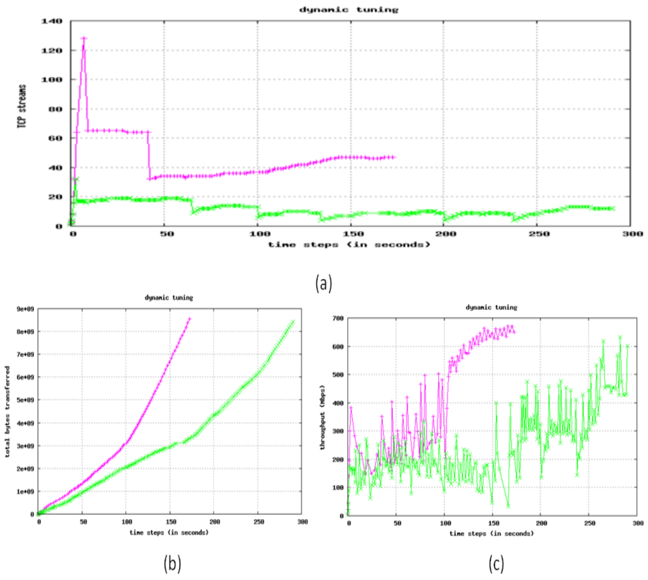
Characteristics of the communication infrastructure determine which action should be taken when tuning data transfer operations in order to obtain high transfer rates. Local area networks and wide area networks have different characteristics, so they demonstrate diverse features in terms of congestion, failure rate, and latency. In most cases, congestion is not a concern in dedicated high bandwidth networks. However, the latency wall in data transfers over high bandwidth connections is still an issue [12,13,14]. Enough data should be obtained from the applications and storage layers for high throughput performance. Data transfer optimization has been deeply studied in the literature [15,16,17]. However, many of the solutions require kernel level changes that are not preferred by most domain scientists. In this study, we concentrate on application level auto-tuning methodologies that are applied in user-space for better transfer performance [18,19,20,21]. Using multiple data transfer streams is a common technique applied in application layer to increase the network bandwidth utilization [13,17,22]. Instead of a single connection at a time, multiple streams are opened for a single data transfer service. Larger bandwidth in a network is gained with less packet loss rate; concurrent data transfer operations that are initiated at the same time better utilize the network and system resources.

4.1 Application-level dynamic tuning

To achieve high throughput, the number of multiple connections needs to be adjusted according to the capacity of the underlying environment. There are several studies on parameter estimation in order to predict the network behavior and to find a good estimation for the level of parallelism [17,22,23,24,25]. However, those techniques usually depend on performance results of sample transfers with different parameters. The systems probe and measurements with external profilers are needed. Complex models are used to calculate the optimum number of multiple streams with the help of sample measurements in order to make a prediction [23,25,26]. Further, network conditions may change over time in the shared environments, and the estimated value might not reflect the most recent state of the system. The achievable end-to-end throughput and the system load in communicating parties might change during the period of a data transfers, especially when large volume of data needs to be transmitted.

Dynamically setting the number of optimal parallel streams has been introduced in [27]. Further, there are several studies in adaptive parameter tuning [20,22]. We have designed a similar approach in which the number of concurrent connections is set dynamically in a large-scale data transfer. The proposed methodology operates without depending on any historical measurements and does not use external profiles for measurement. Instead of using predictive sampling as proposed in [17,25,26], we make use of the instant throughput information gathered from the actual data transfer operations that are currently active. The number of multiple streams is set dynamically in an adaptive manner by gradually increasing the number of concurrent connections up to an optimal point. The adaptive approach does not require complex models for parameter optimization. This also enables us to adapt varying environmental conditions to come up with a high-quality tuning for best system and network utilization.

Gradually improving concurrency level brings a near optimal value without the burden of complex optimization steps to find the major bottleneck in a data transfer. In this adaptive algorithm, a change in the performance is detected and the number of concurrent connections is adjusted accordingly. Figure 5 shows the results from an adaptive transfer performance with the number of concurrent TCP streams. We have conducted our experiments in a 1-Gbps network where synthetic data transfer operations were started in order to simulate a communication channel with the shared bandwidth. The adaptive tuning by adjusting the concurrency level dynamically results in better throughput performance. Figure 5.a shows the number of streams over time in seconds. Figure 5.b shows the total volume of data transferred over time, and Figure 5.c shows the instant throughput measured while data transfer operation is active. The changes in the performance as in Figure 5.c were detected live, and the number of concurrent streams was adjusted over time as shown in Figure 5.a.



(a)

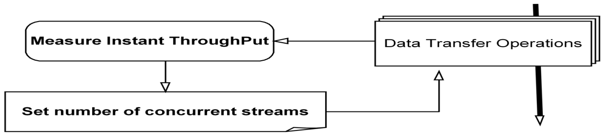
(b)

(c)

Figure 5: Adaptive transfer adjustment: (a) TCP streams, (b) total bytes transmitted, (c) instant throughput over the time in seconds.

Instead of making measurements with external profilers to set the level of concurrency, transfer parameters are calculated using information from current data transfer operations. Thus, the network would not have extra packets and extra load is not put onto the system due to extraneous calculations for exact parameter settings. The number of multiple streams is set by observing the achieved application throughput for each transfer operation, and parameters are gradually adjusted according to the current performance merit. The transfer time of each operation is measured and the total throughput is calculated. The best throughput for the current concurrency level is recorded. The actual throughput value of the data transfers is calculated, and the number of multiple streams is increased if the throughput value is larger than the best throughput seen so far. In this dynamic approach, we try to reach to a near optimum value gradually, instead of finding the best parameter achieving the highest throughput at once. We underline the fact that the focus is on application level tuning such that we do not deal with low-level network and server optimization. We adjust the number of multiple streams according to the dynamic environmental conditions, and also taking into the consideration of the fact that there might be other data transfer operation using the same network resources.

We first start with a single stream of a transfer and measure the instant achievable throughput. The number of concurrent transfers running at the same time is increased gradually as long as there is any performance gain in terms of overall throughput. Although this incremental approach is practical especially for a large-scale data transfer that takes time to complete, a good starting point is highly desirable in terms of the number of multiple streams. Inspired from the TCP congestion window mechanism, we benefit from exponentially increasing the concurrency level in the beginning of the tuning process. Figure 6 gives a glimpse of the algorithm used to set the optimum concurrency level. We analyze the search pattern in two phases. In the first phase, we exponentially increase the number of multiple streams to quickly find a good starting point. In the second phase, we gradually set the concurrency level by measuring instant throughput between every parameter update in order to come up with the optimal number of multiple streams in a dynamic manner.



```
Start with single stream (N=1)

(fast start - phase 1)
    Increase the number of multiple streams (N=N*2)
    Measure instant achievable throughput
    If current throughput value is better than previous one,
        continue
    Otherwise, set N to the previous value (N/2)

(incremental - phase 2)
    Gradually increase the number of streams (N=N+1)
    If no throughput gain by increasing number of streams,
        found the equilibrium point
```

Figure 6: Algorithm searching for the optimal concurrency level

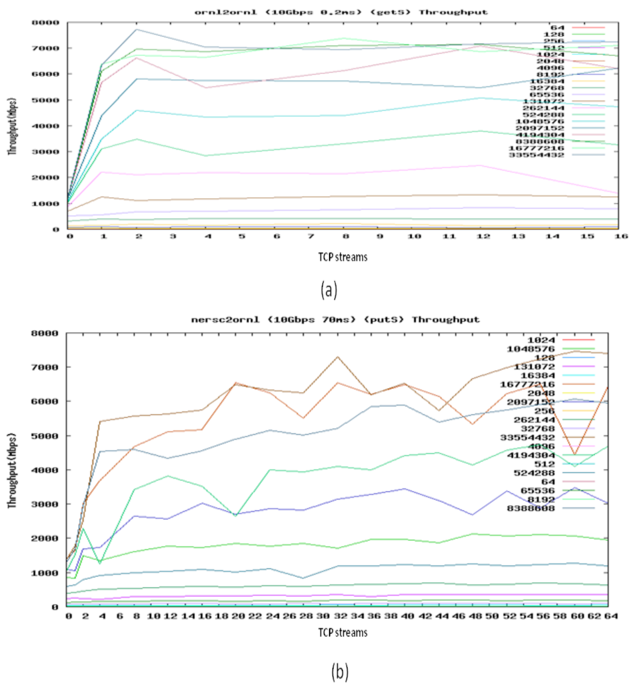
The interval between the adjustment points is another important issue. We measure the instant throughput performance, but it may not be appropriate to make adjustment on the number of concurrent streams after every measurement point. Considering the possibility of minor fluctuations in the network throughput performance, we set a threshold value based on the transferred data size before observing any changes in the achievable throughput performance and deciding the needs of adjustments on the number of concurrent streams. This property has also shown in Figure 5.a where the number of concurrent streams is adjusted based on the major changes in the achievable throughput. Figure 5.c shows the corresponding changes in the instant throughput during the entire transfer. If a significant drop change in the throughput performance has been detected, the number of concurrent streams is decreased by half (*N/2*), and searching for the optimal number of concurrent streams gets started as described in Figure 6.

4.2 A Simple Throughput Prediction Model

We have performed several experiments with various file sizes by changing the number of concurrent TCP streams. Figure 7 shows the overall throughput performance over the number of concurrent TCP streams under different round trip time (RTT) values when different sizes of files are transferred. The first observation is that, the latency directly affects the behavior of the throughput performance curve. Figure 7.a shows throughput performance on a 10-

Gbps network with round-trip time 0.5ms. As seen in Figure 7.b, more TCP streams are needed to fill the network bandwidth when latency is higher.

Our second observation is that we can use power-law to come up with a simple prediction schema. We see that the relationship between the number of multiple streams and the throughput gain can be approximated by a simple power-law model. Figure 8 illustrates log-log graphs for total throughput versus the number of multiple streams. We can classify the behavior into two parts. In the first part, where we reach 80% of the achievable throughput, the power law approximation models the behavior of the multiple streams versus throughput. Based on this information, we present a power-law approximation to predict the number of multiple streams.



(a)

(b)

Figure 7: Total throughput over the number of streams; (a) rtt=0.5 ms, (b) rtt=70ms

Power-law demonstrates the mathematical relationship between two quantities where one attribute varies as a power of another attribute. Many functions, especially man-made phenomena, follow power law [28,29]. In our case, the achievable throughput varies as a power of the number of streams where the scaling exponent is related to the round-trip time. It seems to represent the tradeoff between the gain and the cost of adding TCP streams in a data transfer operation over a network.

A simple model was also developed to estimate the starting point based on round-trip time (*RTT*) between the source and destination hosts. The goal is to set the initial number of multiple streams that would be calculated in the fast-start phase of the algorithm given in Figure 6, and it will be used as the base point in the second phase of the algorithm, where we gradually adjust for optimum tuning. Note that we try to obtain a good starting point that will be used later for fine-tuning.

The power law approximation is modeled as

$$T = (n / c)^{(RTT / k)} \qquad (1)$$

where *T* is achievable throughput in percentage, *n* is the number of multiple streams ($n > 0$), *RTT* is the round trip time, and *c* and k are constant factors. Unlike other models [23,24,25] trying to find an approximation model for the multiple streams and throughput relationship, this model only focuses on the initial behavior of the transfer performance.

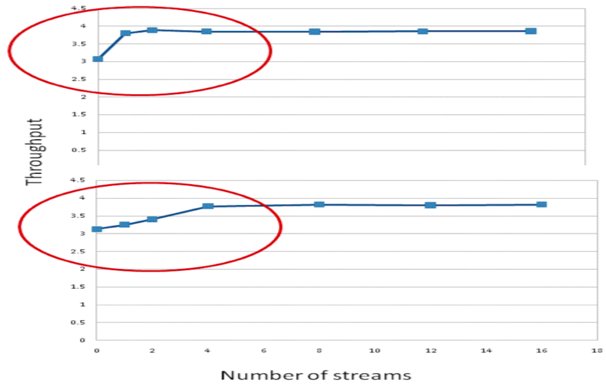


Figure 8: Total throughput over the number of streams (log-log scale).

As in Figure 9, test runs show achievable throughput over the number of concurrent transfers in different *RTT* values. When *RTT* is low, the achievable throughput starts high with the low number of streams and quickly approaches to the optimal throughput. When *RTT* is high, more number of streams is needed for higher achievable throughput. Our goal is to come up with a proper starting point for the number of concurrent streams. The simple estimation model must capture the relationship between the latency and the throughput performance. The initial estimation value will be used in dynamic parameter tuning for the optimum number of streams.

Since our simple model estimates the achievable throughput in percentage, (*n* / *c*) should be less than 1. Further, the exponent (*RTT* / *k*) should be less than 1, in order to capture the relationship between the achievable throughput in percentage and cost of adding additional transfer streams into the transfer operation. In our test, shown in Figure 9, where we have conducted experiments over high-bandwidth networks with high and low latency, we set *c* as 100 as the maximum number of concurrent streams, and *k* as 300 the maximum *RTT*. The constant values in the given formula can be adjusted to obtain more accurate model. However, accurate starting point is not required in our case, and the model can estimate the number of streams to give 80% of achievable throughput performance as starting point, similar to 80-20 rule in Pareto distributions [29].

$$0.8 = (n / c)^{(RTT / k)} \qquad (2)$$

$$n = (e^{(k * \ln 0.8 / RTT)}) \cdot c \qquad (3)$$

According to the equation (3), the initial estimated value for number of streams *n* is; 10 if *RTT* is 30ms (Figure 9.d), 38 if *RTT* is 70ms (Figure 9.e), 61 if *RTT* is 140ms (Figure 9.f), and 0.1 (which is rounded to 1) if *RTT* is 10ms (Figure 9.c).

A

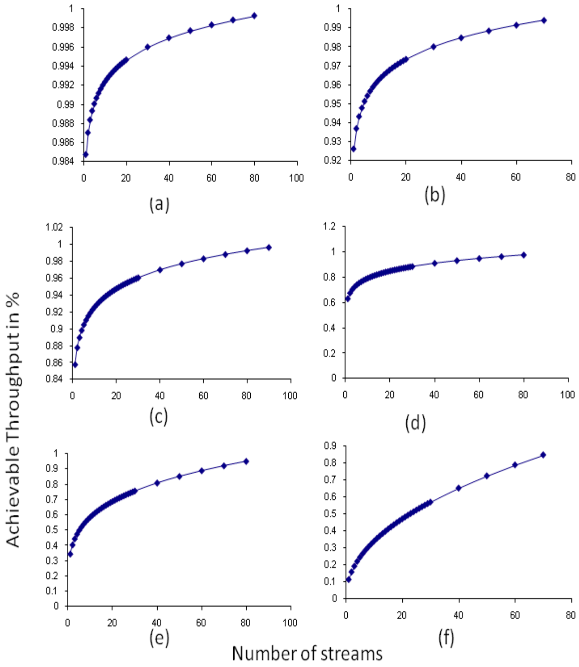


Figure 9: Achievable throughput in percentage over the number of streams with low/medium/high *RTT*;

(a) *RTT*=1ms, (b) *RTT*=5ms, (c) *RTT*=10ms, (d) *RTT*=30ms, (e) *RTT*=70ms, (f) *RTT*=140ms

practical throughput prediction model for approximating the initial behavior of the transfer performance is important. The adaptive transfer adjustment model gives a base for optimal transfer throughput management in dynamic environments. In addition to that, we need a mathematical model that will be used to obtain a good starting point for adaptive fine-tuning. This model should be simple and easily applicable inside BDM for initial concurrency estimation. The test runs in real shared environment show that the dynamic transfer management in BDM with the initial transfer estimation would provide an effective way in obtaining optimal transfer performance.

5. Testbed

The Green Data Oasis (GDO) [6] at LLNL has over 600 TB of spinning disk and serves 35 TB of IPCC CMIP-3 multi-model data. Two GridFTP server nodes with Solaris 10 running ZFS on AMD-64 hardware were used with access to the 10-Gbps ESnet network. Two NERSC Data Transfer Nodes [7] were used to transfer data located on NERSC storage units based on GPFS. A 10-Gbps SDN through OSCARS [8] could be reserved through ESnet between NERSC and LLNL. In this test setup, randomly selected a few climate datasets from IPCC CMIP-3 were replicated for test runs under different transfer conditions. Dataset sizes range from 40 GB to 10 TB.

6. Discussion and Future Work

The ESG has the difficult challenges of managing the distribution of massive datasets and accessing and analyzing them. The IPCC CMIP-3 holds over 35 TB of data at the LLNL site. The IPCC Coupled Model Intercomparison Project, phase 5 (CMIP-5) is projected to be 10 PB. Bulk Data Mover (BDM) is to provide the efficient data delivery required for the scalability that the ESG needs for data access in the highly collaborative decentralized environment, with efficient and adaptive transfer management.

We have studied dynamic transfer adjustment to enhance end-to-end data transfer performance. In our adaptive approach, we set the level of concurrency on the fly. The number of multiple streams is set by observing the achieved application throughput for each transfer operation, and parameters are gradually adjusted according to the current performance merit. We have studied several versions of this dynamic adaption approach and enhanced the algorithm after intensive testing and analysis. We have been working on applying and implementing our dynamic adaptation algorithm in BDM.

We also observed that we can use power-law to come up with a simple prediction schema. We see that the relationship between the number of multiple streams and the throughput gain can be approximated by a simple power-law model. We presented a power-law approximation to predict the number of multiple streams. The achievable throughput varies as a power of the number of streams where the scaling exponent is related to the round-trip time. It represents the tradeoff between the gain and the cost of adding TCP streams in a data transfer operation over a network. This power law model is used along with the dynamic adaptation algorithm. The goal is to set the initial number of multiple streams that would be calculated in the first phase. This will be used as the base point in the second phase of the algorithm, where we gradually adjust for optimum tuning. Note that we try to obtain a good starting point that will be used later for fine-tuning (dynamic adaptation). Unlike other models in the literature (trying to find an approximation model for the multiple streams and throughput relationship), this model only focuses on the initial behavior of the transfer performance. When RTT is low, the achievable throughput starts high with the low number of streams and quickly approaches to the optimal throughput. When RTT is high, more number of streams is needed for higher achievable throughput. We are in the process of applying this model inside BDM. Besides, we plan to enhance our mathematical approach by conducting several other experiments in different environments.

Acknowledgements

This work was funded by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contracts DE-AC02-05CH11231. We would like to thank Jeff Long at Lawrence Livermore National laboratory, and Jason Hick at National Energy Research Scientific Computing Center for their support on our experiments.

References

[1] Earth System Grid (ESG), http://www.earthsystemgrid.org

[2] E. Dart and B. Tierney (editors), "BES Science Network Requirements", Report of the Basic Energy Sciences Network Requirements Workshop sponsored by Basic Energy Sciences Program Office, DOE Office of Science and the Energy Sciences Network 2007.

[3] D. N. Williams, D. E. Middleton, M. Anitsecu, V. Balaji, W. Bethel, S. Cotter, W. G. Strand, K. Schuchardt, and A. Shoshani, "Extreme Scale Data Management, Analysis, Visualization, and Productivity in Climate Change Science," Report for the Extreme Scale Computing Workshop sponsored by DOE Joint Office of Biological and Environmental Research (BER) and the Office of Advanced Scientific Computing Research (ASCR), http://esg-pcmdi.llnl.gov/publications_and_documents/Extreme_Scale_Data_Mgmt_Panel%20Report.pdf 2008.

[4] A. Sim, D. Gunter, V. Natarajan, A. Shoshani, D. Williams, J. Long, J. Hick, J. Lee, E. Dart, "Efficient Bulk Data Replication for the Earth System Grid", International Symp. on Grid Computing (ISGC), 2010.

[5] Williams et al., “The Earth System Grid: Enabling Access to Multimodel Climate Simulation Data”, in the Bulletin of the American Meteorological Society, February 2009.

[6] Green Data Oasis (GDO), https://computing.llnl.gov/resources/gdo/

[7] NERSC Data Transfer Node (DTN), http://www.nersc.gov/nusers/systems/datatran/

[8] OSCARS, http://www.es.net/OSCARS/

[9] Allcock, W., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I., and Foster, I., "The Globus Striped GridFTP Framework and Server" In Proc. of the 2005 ACM/IEEE Conference on Supercomputing.

[10] Bulk Data Mover (BDM), http://sdm.lbl.gov/bdm/

[11] D. Bernholdt, S. Bharathi, D. Brown, K. Chancio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, D. Williams, "The Earth System Grid: Supporting the Next Generation of Climate Modeling Research," Proceedings of the IEEE, vol. 93, pp. 485- 495, March 2005.

[12] Wu, Y., Kumar, S., and Park, S., "Measurement and performance issues of transport protocols over 10Gbps high-speed optical networks", *Computer Network* 54, 3 (Feb. 2010), 475-488

[13] M. Balman and T. Kosar, "Data Scheduling for Large Scale Distributed Applications", In Proceedings of the 9th International Conference on Enterprise Information Systems Doctoral Symposium (DCEIS 2007), 2007

[14] H. Bullot, R. Les Cottrell and R. Hughes-Jones, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks", Journal of Grid Computing, Springer, Volume 1, Number 4, Dec. 2003

[15] FastTCP. An alternative congestion control algorithm in tcp. http://netlab.caltech.edu/FAST.

[16] sTCP. Scalable TCP. http://www.deneholme.net/tom/scalable/, 2006.

[17] T. Dunigan, M. Mathis, and B. Tierney, "A tcp tuning daemon”, In Proceedings of SuperComputing: High-Performance Networking and Computing, 2002.

[18] M. Gardner, S. Thulasidasan, and W. Feng, "User-space auto tuning for tcp flow control in computational grids", Computer Communications, 27:1364-1374, 2004.

[19] S. Soudan, B. Chen, and P. Vicat-Blanc Primet, "Flow scheduling and endpoint rate control in grid networks", Future Gener. Comput. Syst., 25(8):904–911, 2009.

[20] W. Feng, M. Fisk, M. Gardner, and E. Weigle, "Dynamic right sizing:An automated, lightweight, and scalable technique for enhancing grid performance", In Proc. of the 7th IFIP/IEEE International Workshop on Protocols for High Speed Networks, 2002.

[21] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser and I. Foster, "GridFTP Pipelining", Proceedings of the 2007 TeraGrid Conference, June, 2007

[22] T. Ito, H. Ohsaki, and M. Imase, "On parameter tuning of data transfer protocol gridftp in wide-area grid computing", In Proc. of Second International Workshop on Networks for Grid Applications, GridNets, 2005.

[23] Hacker, T. J., Noble, B. D., and Athey, B. D., "Adaptive data block scheduling for parallel TCP streams", In Proceedings of the High Performance Distributed Computing, 2005.

[24] Mirza, M., Sommers, J., Barford, P., and Zhu, X., "A machine learning approach to TCP throughput prediction", SIGMETRICS Perform. Eval. Rev. 35, pg 97-108, 2007

[25] E. Yildirim, M. Balman, and T. Kosar, "Dynamically Tuning Level of Parallelism in Wide Area Data Transfers", In Proceedings of DADC'08, June 2008

[26] D. Yin, E. Yildirim, and T. Kosar, "A Data Throughput Prediction and Optimization Service for Widely Distributed Many-Task Computing", In Proceedings of MTAGS, 2009

[27] M. Balman and T. Kosar, "Dynamic Adaptation of Parallelism Level in Data Transfer Scheduling", In Proc. of Second International Workshop on Adaptive Systems in Heterogeneous Environments, 2009

[28] Faloutsos, M., Faloutsos, P., and Faloutsos, C., "On power-law relationships of the Internet topology", In Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication SIGCOMM 1999.

[29] M. Newman, "Power laws, Pareto distributions and Zipf's law", Contemporary Physics, Volume 46, Issue 5, pages 323 – 351, September 2005.