Tuning GridFTP Pipelining, Concurrency and Parallelism Based on Historical Data

Jangyoung KIM(a), Member

SUMMARY This paper presents a prediction model based on historical data to achieve optimal values of pipelining, concurrency and parallelism (PCP) in GridFTP data transfers in Cloud systems. Setting the correct values for these three parameters is crucial in achieving high throughput in end-to-end data movement. However, predicting and setting the optimal values for these parameters is a challenging task, especially in shared and non-predictive network conditions. Several factors can affect the optimal values for these parameters such as the background network traffic, available bandwidth, Round-Trip Time (RTT), TCP buffer size, and file size. Existing models either fail to provide accurate predictions or come with very high prediction overheads. The author shows that new model based on historical data can achieve high accuracy with low overhead.

key words: big data, throughput optimization, throughput estimation, pipelining, concurrency, parallelism

1. Introduction

In all areas of science and industry, increasingly complex big data are being generated at the scales of petabytes and beyond. Despite the trend of moving the application to the data rather than the data to the application, large data sets still need to be moved around for increased availability, performance, and recovery purposes. Sharing, disseminating, and analyzing these large data sets pose a big challenge despite the deployment of petascale computing systems, and optical networking speeds reaching and exceeding 100 Gbps. The majority of users fail to obtain even a fraction of the theoretical speeds these high-bandwidth networks promise, due to issues such as sub-optimal protocol tuning, inefficient end-to-end routing, disk performance bottlenecks on the sending and/or receiving ends, and server processor limitations.

Different protocol parameters such as TCP pipelining, parallelism and concurrency levels play a significant role in the achievable network throughput. However, setting the optimal numbers for these parameters is a challenging problem, since poorly tuned parameters can cause underutilization of the network or they can overburden the network and degrade the performance due to increased packet loss, end-system overhead, and other factors. In addition, the author’s previous work [1], [2] sought to find a solution for an efficient big data transfer by utilizing the well-known protocol GridFTP with PCP (Pipelining-Concurrency-Parallelism).
is to find optimal Pipelining-Concurrency-Parallelism combinations by using historical database which would be the main difference from previous approaches.

3. Proposed Generic Model

The author describes the proposed generic model and algorithm based on historical data in this section. The goal of this model is to find optimal PCP values by utilizing historical data to achieve high accuracy and low overhead.

Table 1 presents a part of historical PCP database which is the case of FutureGrid [5]. Although not shown in Table 1, all historical data of each testbed (LONI [3], XSEDE [4] and Emulab [6]) is stored in a huge PCP database. The historical data has information such as file size, number of files, bandwidth, RTT, buffer size, PCP, FAST and throughput. Each parameter (file size, number of files, bandwidth, RTT and buffer size) varies based on each testbed (FutureGrid, LONI, XSEDE and Emulab) as stated in Sect. 5. Especially, FAST is a parameter option that could make data transfer fast. However, the author does not use FAST option on FutureGrid as shown in Table 1, since FAST option usually used for large data transfers.

There are three steps for deriving the proposed generic model. A generic model is one that could be applied to any real network environments. First, the author clustered the database based on each file size such as 1MB, 2MB, 3MB, 4MB, 8MB, 16MB, 32MB, 64MB and 128MB of each testbed. Second, the author found PCP values with the highest throughput of each file size in the database. Third, the author applied k-nearest neighbors (k-NN) algorithm to derive a generic model with PCP values. As shown in Table 1, the database is organized with 1MB file size. Thus, the author can select five large throughput values with PCP based on distance. Distance can be considered as the difference between each throughput. For example, five rows (five large throughput) were first selected in Table 1. Then, the author applied k-nearest neighbors (k-NN) algorithm to find a generic PCP row in Table 1. The idea behind k-NN is to select the largest and most frequent value as the generic, within a certain distance. In this case, PCP (8-9-1) would be a generic one. Also, the author applied weight to each different file size or number of files. The author generated real sampling throughput values by running an actual test of certain parameters (e.g. file size, number of files). So, the author used two real sampling throughput values to find the optimal throughput values with PCP, based on different file size and number of files. The first sampling was chosen based on maximum throughput with PCP values from generic model curve and the second sampling was chosen based on maximum throughput with PCP values from another historical information (e.g. different file size, number of files). However, if the first and second samplings were found on the same PCP values coincidentally, the author could use only one sampling to find and predict an optimal throughput with PCP values. For different RTT and bandwidth, the author did not have additional historical information. So, the author used three samplings based on three maximum throughput with PCP values from generic model curve to increase an accuracy. Based on three sampling candidates, the author selected the maximum throughput as the generic one for an optimal throughput with PCP values. Overall, the author used one to three samplings to find and predict optimal throughput and PCP values with very high accuracy on different parameters.

4. Applying Generic Model to Real Network Cases

The author sets a generic model and is going to apply this model to real cases. As described in Sect. 5, the author categorized tests into two types. First, the author can undoubtly apply the generic model in cases based on the same parameters the author sets. Second, the author can apply the generic model in cases based on the different parameters which is more challenging. Since the author has four types of testbeds, the author can apply real cases to one of those four types. For example, if the real user employs 1 Gbps system then Emulab [6] or FutureGrid [5] would be the best match based on RTT. If the RTT is dynamically changed, the author can apply to Emulab. On the contrary, if the user utilizes a 10 Gbps system, it would be applied to LONI [3] or XSEDE [4] based on RTT. For small RTT, the author can apply to LONI otherwise the author can apply to XSEDE.

| FileSize(MB) | NumberofFiles | Bandwidth(Mbps) | RTT(Seconds) | BufferSize(Bytes) | P  | CC  | PP  | FAST | Throughput(Mbps) |
|-------------|---------------|-----------------|-------------|-------------------|----|-----|-----|------|------------------|
| 1           | 1000          | 1000            | 0.06        | 131071            | 1  | 9   | 8   | OFF  | 114.0794092     |
| 1           | 1000          | 1000            | 0.06        | 131071            | 1  | 8   | 7   | OFF  | 113.9152178     |
| 1           | 1000          | 1000            | 0.06        | 131071            | 9  | 9   | 8   | OFF  | 111.567158      |
| 1           | 1000          | 1000            | 0.06        | 131071            | 9  | 1   | 8   | OFF  | 11.137421       |
| 1           | 1000          | 1000            | 0.06        | 131071            | 8  | 7   | 7   | OFF  | 108.7216035     |
Therefore, the author can apply generic model to certain real system cases and find optimal PCP values.

5. Experimental Results

The author conducted experiments on the following testbeds: LONI [3], XSEDE [4], FutureGrid [5] and Emulab [6]. LONI has 10 Gbps bandwidth with 4ms RTT, while XSEDE has 10 Gbps bandwidth with 60ms RTT. Emulab has 1 Gbps bandwidth with 100ms RTT and FutureGrid has 1 Gbps bandwidth with 60ms RTT. The buffer size (Bytes) of LONI, XSEDE, FutureGrid and Emulab is 16777216, 32777216, 31071 and 3107200 respectively. Due to many graphics and limited space, the author can only show FutureGrid cases. Please also note that the author conducted on real testbeds with background network traffic, which does not assume that the network is stationary.

In LONI and XSEDE test cases, the author used file size 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB and 128MB with number of files (e.g. 128, 256, 512) while in Emulab test case, the author used file size 1MB, 2MB, 4MB and 8MB. Since the disk space is quite limited in Emulab, the author could not transfer large files especially. In FutureGrid, the author used file size 1MB, 2MB, 3MB and 4MB with large numbers of files (e.g. 1000, 3000, 10000).

In the experiments, the author conducted two types of test of each testbed. The first type is based on same parameters (e.g. bandwidth, RTT, file size, number of files) and the second type is based on different parameters. So, it is important to observe the accuracy of experimental results based on different parameters which makes generic model strong to apply other network environments. Please note that the author conducted experiments 10 times based on each type. Please note that all these experimental parameters and values are determined based on each different combination of each network environments which would be practical. In addition, the results present based on parameters such as file size, number of files, bandwidth, RTT and PCP because those parameters mainly affect the throughput of data transfers. Hence, this paper focuses more on those parameters than other parameters.

5.1 Based on Same Bandwidth, RTT, File Size and Number of Files

Based on the same parameters, the author applied the same bandwidth, RTT, file size and number of files. The author applied 10 Gbps bandwidth with 4ms RTT in LONI while 10 Gbps bandwidth with 60ms RTT were used in XSEDE. The author applied 1 Gbps bandwidth and 60ms RTT for Emulab. For FutureGrid, the author applied 1 Gbps bandwidth and 60ms RTT. LONI used Eric and Oliver as a source and destination while XSEDE used Lonestar and Steele as a source and destination. Emulab used node6 and node1 as a source and destination while FutureGrid used India and Alamo as a source and destination.

Figure 1 presents FutureGrid test cases and also demonstrates high accuracy (similarity between actual points and generic model’s points) with the same parameters (e.g. 1 Gbps, 60ms). So, the throughput would be higher than other cases. Similarly, overall cases show highly accurate results.

5.2 Based on Different Bandwidth, RTT, File Size and Number of Files

Based on different parameters, the author applied LONI with 10 Gbps bandwidth and 8ms RTT while XSEDE was applied with 10 Gbps bandwidth and 35ms RTT. Emulab was applied with 500 Mbps bandwidth and 50ms RTT while FutureGrid was applied with 1 Gbps bandwidth and 90ms RTT. LONI used Eric and Painter cluster as a source and destination while XSEDE used Lonestar and Open Science Grid as a source and destination. FutureGrid used India.
Fig. 2  Generic PCP Model Curve and Actual data comparison with different parameters on FutureGrid.

and Sierra as a source and destination while Emulab used the same node0 and node1 but with different bandwidth and RTT (e.g. 500 Mbps, 50ms).

Similarly, Fig. 2 shows very promising results that the predicted point matched to the corresponding actual curve. Finally, the author can categorize any real environments into one of the testbeds (Emulab, FutureGrid, LONI and XSEDE) and find optimal throughput with PCP values based on generic model with historical data information.

6. Conclusion

This paper shows very promising results with a concrete prediction model and algorithms. Although it is a challenging task to find optimal PCP (Pipelining-Concurrency-Parallelism) values, the author efficiently found optimal PCP values and verified that the proposed model is strong and accurate and demonstrates promising experimental results. So, the author can apply the prediction model to the real network environments to make big data transfer fast. In future work, the author can add more experiments to extend the historical database, based on different parameters such as RTT and bandwidth. Also, the author can combine the prediction model and the previous model[2] to provide a better solution finding optimal PCP values in the real network environments.

Acknowledgements

This project is partially supported by NSF under award numbers CNS-1131889 (CAREER), OCI-0926701 (STCI-Stork), and CCF-1115805 (CiC-Stork). Also, this work was supported by the GRRC program of Gyeonggi province.[(GRRCSUWON2014-B1), Cooperative Recognition and Response System based on Tension Sensing]. The paper was supported by The University of Suwon in 2014. In addition, this project has used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF under grant number OCI-1053575.

References

[1] E. Yildirim, J. Kim, and T. Kosar, “Optimizing the sample size for a cloud-hosted data scheduling service,” Proc. 2nd International Workshop on Cloud Computing and Scientific Applications (CCSA in conjunction with CCGRID12), 2012.

[2] J. Kim, E. Yildirim, and T. Kosar, “A highly-accurate and low-overhead prediction model for transfer throughput optimization,” Proc. DISCS Workshop, Nov. 2012.

[3] “LONI.” http://www.loni.org/

[4] “XSEDE.” http://www.xsede.org/

[5] “Futuregrid testbed.” http://www.futuregrid.org

[6] “Emulab-network emulation testbed.”

[7] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, and S. Tuecke, “Software as a service for data scientists,” Commun. ACM, vol.55, no.2, pp.81–88, 2012.

[8] R.S. Prasad, M. Jain, and C. Davrolis, “Socket buffer auto-sizing for high-performance data transfers,” J. Grid Computing, vol.1, no.4, pp.361–376, Aug. 2004.

[9] G. Hasegawa, T. Terai, T. Okamoto, and M. M, “Scalable socket buffer tuning for high-performance web servers,” International Conference on Network Protocols (ICNP01), p.281, 2001.

[10] T. Ito, H. Ohsaki, and M. Imase, “On parameter tuning of data transfer protocol gridftp for wide-area networks,” International Journal of Computer Science and Engineering, vol.2, no.4, pp.177–183, Sept. 2008.

[11] K.M. Choi, E. Huh, and H. Choo, “Efficient resource management scheme of tcp buffer tuned parallel stream to optimize system performance,” Proc. Embedded and ubiquitous computing, Nagasaki, Japan, Dec. 2005.

[12] E. Yildirim, M. Balman, and T. Kosar, Data-intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management, ch. Data-aware Distributed Computing, IGI-Global, 2012.