Dynamic Adaptation for High-Performance Data Transfers[1]

Jan 17, 2011

Mehmet Balman

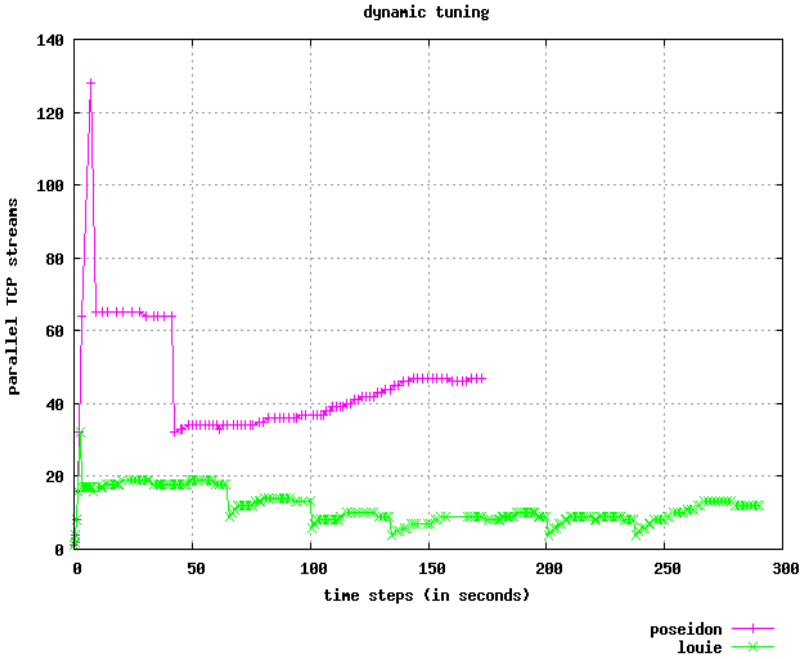
Computational Research Division
Lawrence Berkeley National Laboratory
1 Cyclotron Road MS 50B 3238 Berkeley, CA 94720
mbalman@lbl.gov

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 [1,2,3]. 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 [4,5,6]. 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 [7,8,9,10]. Using multiple data transfer streams is a common technique applied in application layer to increase the network bandwidth utilization [2,5,10]. 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.

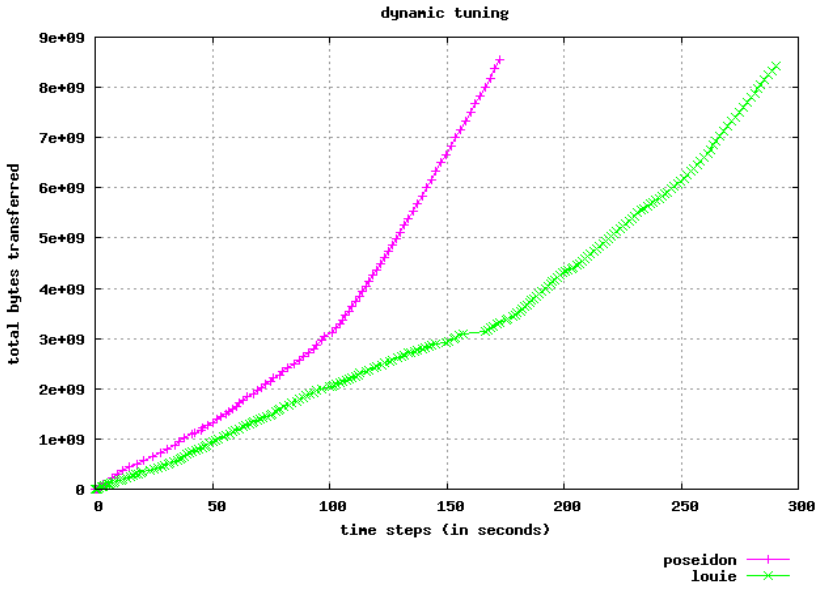
To achieve high throughput, the number of multiple connections needs to be adjusted according to the capacity of the underlying environment. There are several studies on parameter estimation in order to predict the network behavior and to find a good estimation for the level of parallelism [6,11,12,13,14]. 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 [12,14,15]. However, 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 [16]. Further, there are several studies in adaptive parameter tuning [9,11]. 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 [6,14,15], 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. That enables us to adapt varying environmental conditions to come up with a high-quality tuning for best system and network utilization.

Gradually improving the level of concurrency 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 1 shows an illustration of dynamic parameter tuning in which system detects a change in the environment and adjust the level of concurrency for high-performance data transfer.



(a) number of concurrent streams over time



(b) total bytes transferred over time

Figure 1: Adaptive Tuning Algorithm: setting the concurrency dynamically for transfers from poseidon and louie to queenbee machines on LONI network (loni.org / measurements from 2009)

Instead of making measurements with external profilers to set the level of concurrency, transfer parameters are calculated using instant throughput values from currently running data transfer operations. Thus, there is no additional data packets transferred for measurements and there is no additional load in the system for complex parameter estimations. By observing the achieved application throughput, we gradually adjust the number of multiple streams. The best throughput for the current concurrency level is recorded. The actual throughput value 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 directly 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, by considering the fact that there might be other data transfer operations using the same network resources and the achievable throughput can change dynamically.

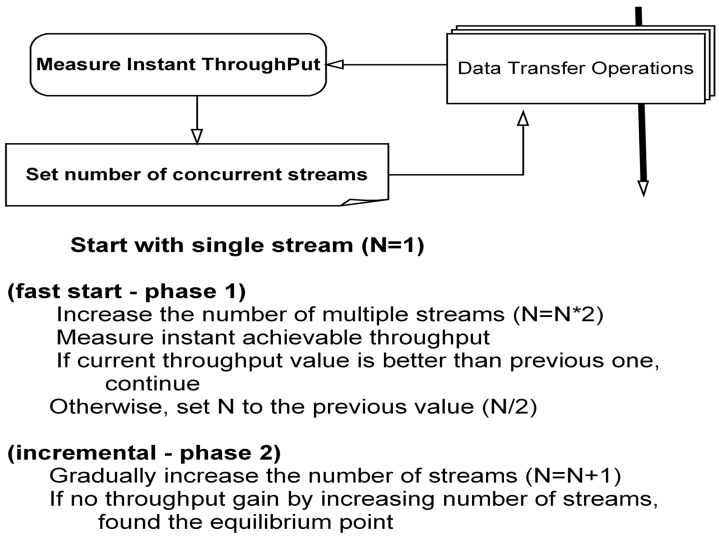


Figure 2: Algorithm searching for the optimal concurrency level

We first start with a single stream and measure the instant achievable throughput. The number of concurrent streams is increased gradually as long as there is any performance gain in terms of overall throughput. Although this incremental approach is practical especially for large-scale data transfers that take long time to complete, a good starting point is highly desirable. Inspired from the TCP congestion window mechanism, we benefit from exponentially increasing the concurrency level in the beginning of the tuning process. Figure 2 gives a glimpse of the algorithm used to set the concurrency level dynamically. 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 adjust the concurrency level by measuring instant throughput in order to come up with an optimal number of streams.

References:

[1] 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

[2] 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

[3] 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, December, 2003

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

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

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

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

[8] 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.

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

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

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

[12] 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.

[13] 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

[14] E. Yildirim, M. Balman, and T. Kosar, "Dynamically Tuning Level of Parallelism in Wide Area Data Transfers", In Proceedings of DADC'08 (in conjunction with HPDC'08), Boston, MA, June 2008D.

[15] Yin, E. Yildirim, and T. Kosar, "A Data Throughput Prediction and Optimization Service for Widely Distributed Many-Task Computing", In Proceedings of MTAGS'09 (in conjunction with SC'09), 2009

[16] M. Balman and T. Kosar, "Dynamic Adaptation of Parallelism Level in Data Transfer Scheduling", In Proceedings of Second International Workshop on Adaptive Systems in Heterogeneous Environments (in conjunction with CISIS2009), 2009

Acknowledgements: This work was supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.