Improving throughput by dynamically adapting concurrency of data transfer

Prasanna Balaprakash[1,2], Vitali Morozov[2], Rajkumar Kettimuthu[1]
[1]Mathematics and Computer Science Division & [2]Argonne Leadership Computing Facility
Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL 60439

Abstract

Improving the throughput of data transfer over high-speed long-distance networks has become increasingly difficult and complex. Numerous factors such as varying congestion scenarios, external factors which are hard to characterize analytically, dynamics of underlying transfer protocol, contribute to this difficulty. In this study, we consider optimizing memory to memory transfer via TCP, where the data is transferred from a source memory to a destination memory using TCP. Inspired by the simplicity and the effectiveness of additive increase and multiplicity decrease scheme of TCP variants, we propose a tuning algorithm that can dynamically adapt the number of parallel TCP streams to improve the aggregate throughput of data transfers.

Throughput optimization

Given a source ***src***, destination ***dst***, and size **Y** of the data that need to be transferred, the problem of optimizing the performance of the file transfer can be formulated as follows:

$$\underset{x \in \mathcal{D}}{\operatorname{argmax}} \int_{T_{start}}^{T^f_{end}} f_t(x, s_t, \delta_t, \theta_t^{src}, \theta_t^{dst})dt,$$

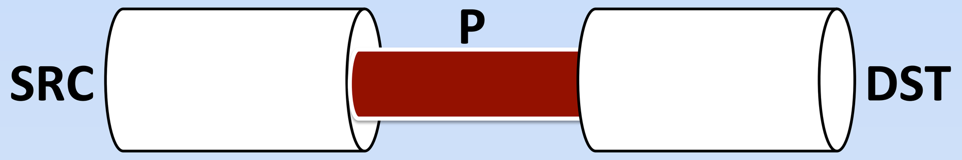
THE OBJECTIVE

- $\boldsymbol{x}$ is a vector of m controllable tuning parameters
- S_t is the size of the data transferred from t''-t' = dt
- $\boldsymbol{\delta_t}$ is a hyperparameter capturing net- work condition at time t
- θ_t^{src} and θ_t^{dst} are the parameters describing the source and the destination loads at time $\boldsymbol{t}$
- $\boldsymbol{T_{start}}$ and $\boldsymbol{T_{end}}$ denote the file transfer starting and ending time, respectively
- The non-negative function $\boldsymbol{f_t(x,\ldots)}$ is the transfer performance metric, typically throughput, at time t, for the parameter configuration $\boldsymbol{x}$

DYNAMIC TUNING

- The network congestion and source-destination bottlenecks can vary with time because of other transfers/workloads can share the same resources
- Maximum achievable throughput will change over time
- To take into account this dynamism, solution to the problem consists in maximizing throughput for every $\boldsymbol{t}$ between $\boldsymbol{T_{start}}$ and $\boldsymbol{T_{end}}$

Congestion link model



- Wide area network consists of sources and destinations connected together via links and routers
- All of the sources operates TCP-like congestion control algorithm
- The links and queues along a network path form a 'pipe' that contain packets in flight
- TCP congestion control is achieved by dynamically adapting the window size according to an additive-increase multiplicative-decrease scheme
  - Source probes the network for spare capacity for additional bandwidth and back-off the number of packets transmitted when congestion is detected

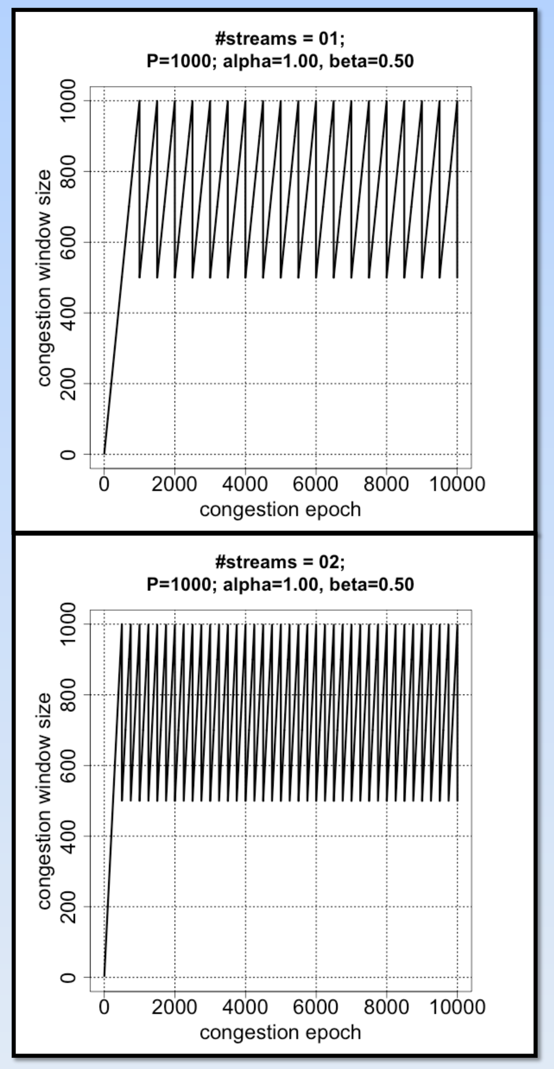
Goal

Develop an online tuning framework for optimizing file transfers throughputs

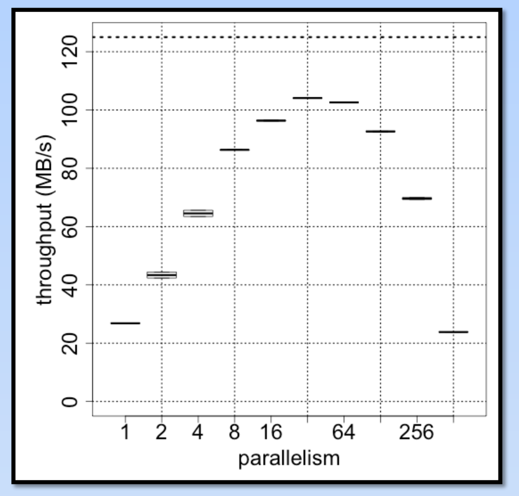
Illustrate the effectiveness on DOE facilities by accelerating by accelerating data analysis workflows

Factors affecting throughput

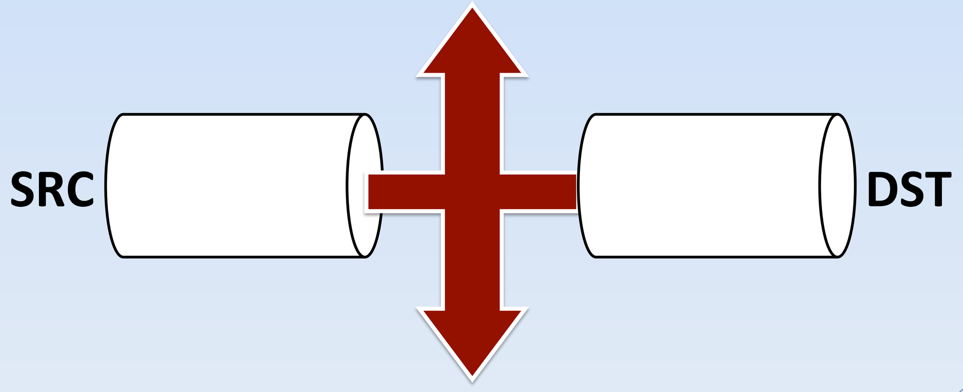
TCP dynamics



Host cannot keep up with the network bandwidth delay product



Congestion in the bottleneck link due to external traffic



Tuning algorithm

- At each control epoch c, the following control logic determines $\boldsymbol{ns_c}$:

$$ns_c = \begin{cases} ns_{c-1} + 1, & \text{if } ns_{c-1} = ns_{c-2} \text{ and } |\Delta_c| > \epsilon \\ ns_{c-1} + 1, & \text{if } ns_{c-1} \neq ns_{c-2} \text{ and } \delta_c > \epsilon \\ ns_{c-1} - 1, & \text{if } ns_{c-1} \neq ns_{c-2} \text{ and } \delta_c < -\epsilon \\ ns_{c-1}, & \text{otherwise} \end{cases}$$

where,

$$\Delta_c = \frac{f_{c-1}(ns_{c-1}, \cdots) - f_{c-2}(ns_{c-2}, \cdots)}{f_{c-2}(ns_{c-2}, \cdots)}$$

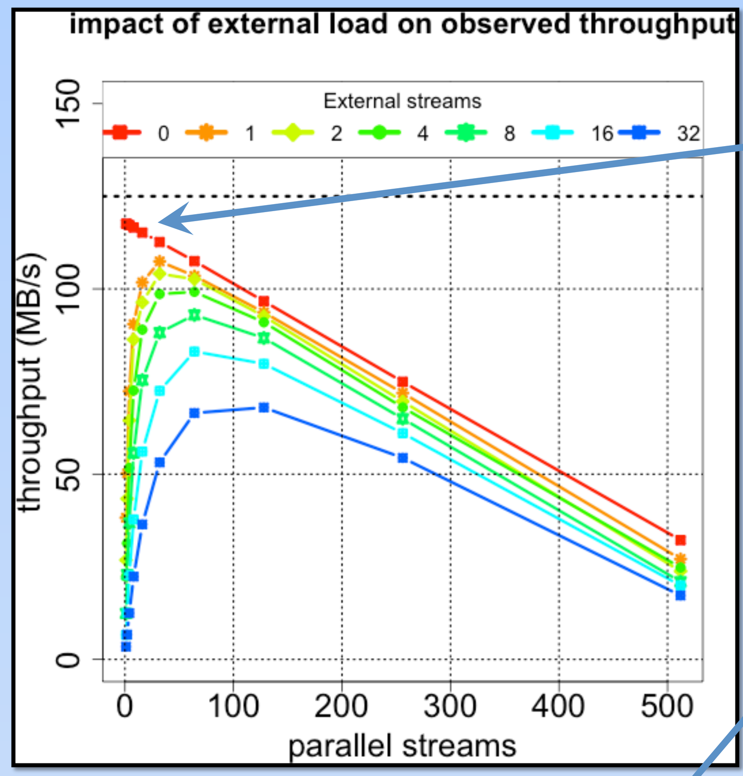
and

$$\delta_c = \frac{\Delta_c}{ns_{c-1} - ns_{c-2}}$$

- When link or TCP becomes bottleneck, ***increase*** the number of streams
- When host becomes bottleneck, ***decrease*** the number of streams
- No significant change in throughput, ***do not change*** the number of streams

Experimental results

- Source: Argonne; Destination: Univ. of Chicago.
- The peak throughput 1 Gb/s (125 MB/s).
- Globus-url-copy from Globus toolkit for memory to memory transfer
- Artificially introduce congestion in the 1 Gb/s link by starting a transfer in the background from source

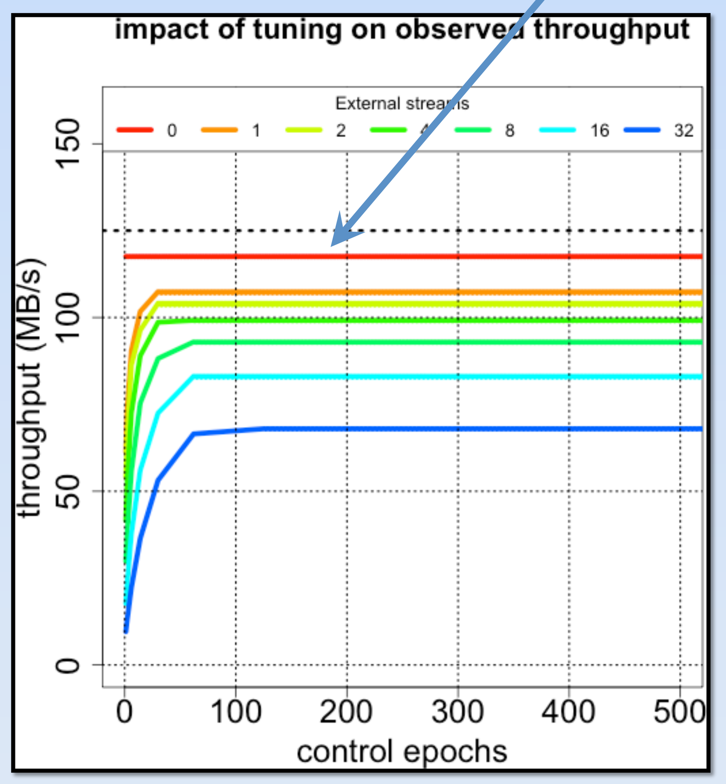


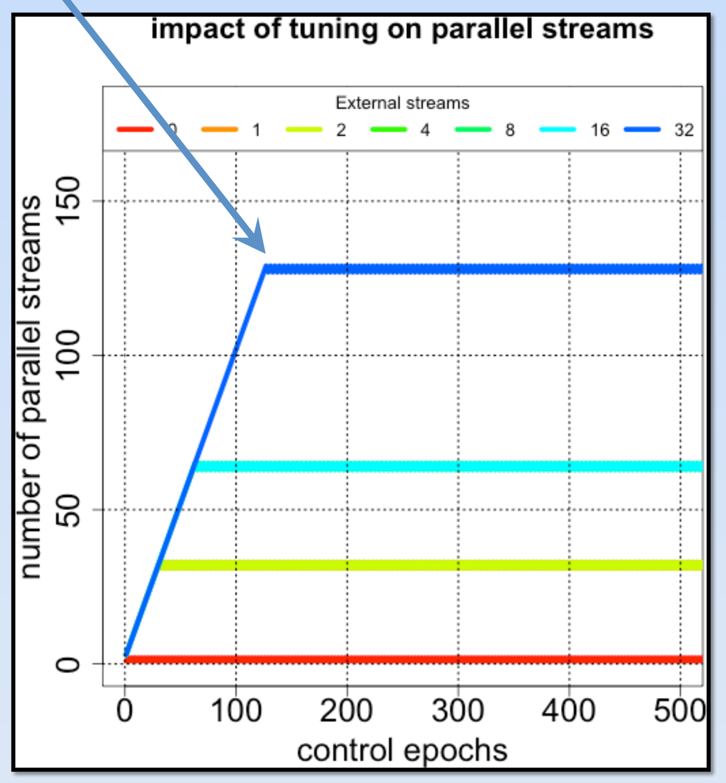
Exploratory analysis

- When there is no congestion, single stream can saturate outbound link; increasing the streams decreases the throughput.
- When there is congestion (external streams), single stream's throughput becomes poor and multiple streams increases the throughput.
- The required number of streams to reach maximum achievable throughput increases with an increase in congestion.
- Increasing the number of streams beyond certain point, which is determined by congestion, results in performance degradation

Tuning results

- With the tuning algorithm, all transfers reach the maximum achievable throughput
- The point at which the host becomes bottleneck increases with congestion
- The number of control epochs required increases with congestion (by default, the controller starts with 1 streams, it needs K control epochs to reach the best value)





Our studies show that results show significant throughput improvement potential under various congestion conditions

Future investigations

- Algorithmic improvements - Rapid adaptation of parameters similar to H-TCP
- Wide area transfer studies - Dedicated such as ESNET and non-dedicated networks
- Including other tuning parameters
- Disk to disk transfer and other potential parameters
- Implementation in production tools - Congestion and bottleneck analysis

[1] W. Bao, V. W. Wong, and V. Leung. A model for steady state throughput of TCP CUBIC. In Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pages 1–6. IEEE, 2010.
[2] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In Network and parallel computing, pages 2–13. Springer, 2005.
[3] T. J. Hacker, B. D. Athey, and B. Noble. The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network. In Vehicle Navigation and Information Systems Conference, 1993., Proceedings of the IEEE-IEE, pages 10–pp. IEEE, 1993.
[4] R. Kettimuthu, G. Vardoyan, G. Agrawal, and P. Sadayappan. Modeling and optimizing large-scale wide-area data transfers. In Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on, pages 196–205. IEEE, 2014.
[5] D. Leith and R. Shorten. H-TCP: TCP for high-speed and long-distance networks. In Proceedings of PFLDnet, volume 2004, 2004.

Argonne NATIONAL LABORATORY

Support for this work was provided through RAMSES project funded by the US Department of Energy's Office of Advanced Scientific Computing Research (ASCR), under the leadership of Rich Carlson