Accelerating big data applications with multi-Gigabit-per-second file transfers

Charles Shiflett
Aspera, an IBM company

Aspera, an IBM company, provides a set of collaboration and file transfer technologies that leverage the Aspera FASP technology which is a distance neutral transfer technology.

This paper will describe Aspera's FASP transfer technology, both next generation FASP which is designed for 100gbit/s transfers as well as our standard FASP technology. This paper will aslo describe how the FASP protocol integrates into high performance / high scalability environments.

A Brief Introduction to Aspera FASP

Aspera FASP is a file transfer protocol designed to give users a high performance solution to sending data across the world limited only by the performance of your internet connection. Aspera FASP is a full stack transfer solution, and as such it provides congestion control, encryption, reliability, file checksums, and direct-to-cloud functionality.

Aspera FASP is a transfer solution designed around real networks meant to just work. It's a point-to-point solution which works on all platforms and within most use cases; Desktop, Command Line, Mobile, Web, REST, and as a library for most programming languages. The FASP protocol is typically layered on top of UDP with reliability, congestion control, and data transmission algorithms that are tailored to real networks, long haul fiber, satellite, wireless, firewalls, and VPNs.

As compared to TCP/IP, which is the standard IP congestion control protocol and used in most data and file transfer scenarios, TCP/IP doesn't effectively utilize available bandwidth in a number of scenario's.

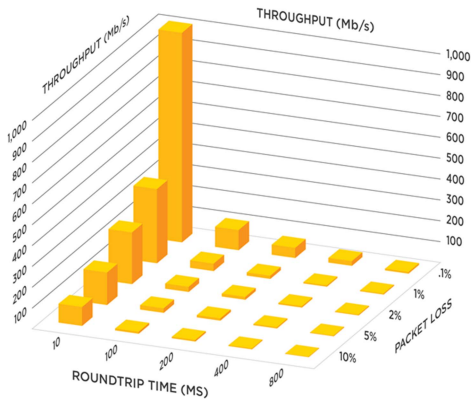

Figure 1: TCP/IP network throughput

As an example of why you wouldn't want to use TCP/IP for everything, *Figure 1* shows typical TCP/IP performance as related to round trip time and/or packet loss on a 1 gig link. In contrast, *Figure 2* shows Aspera FASP performance across the same link. It is exactly due to the relative increase in performance that Aspera FASP is widely used in a number of industries to speed content delivery around the world. For use cases see *asperasoft.com*.

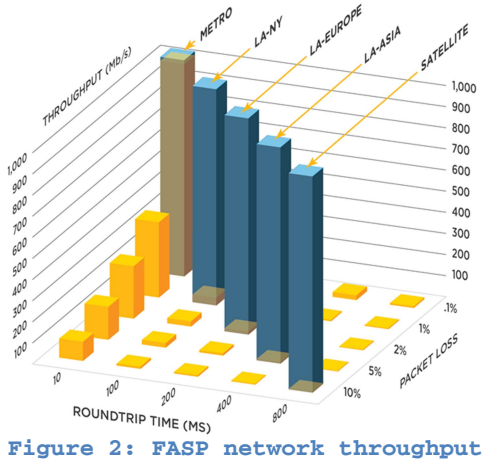

Figure 2: FASP network throughput

Aspera's goal has constantly been to remove bottle necks associated with high throughput file transfers. In 2013 Aspera showed that next-generation FASP was able to utilize 4x10gbit interfaces for a total transfer throughput of about 40 gbit/s in *Big Data Technologies for Ultra-High-Speed Data Transfer in Life Sciences* [http://asperasoft.com/fileadmin/media/Asperasoft.com/Resources/White_Papers/Big_Data_Life_Sciences_AsperaWP.pdf]. In 2014 Aspera showed that FASP was able to operate at near to 80 gbit/s with 2 x 40gbit network cards [https://communities.intel.com/community/itpeernetwork/healthcare/blog/2014/11/12/sc14-accelerating-life-sciences-at-80-gbits]. Our next whitepaper scheduled to be released contemporaneously with this paper will be showing Aspera FASP utilizing a 100 gbit network link using 3 x 40gbit network cards to transfer scientific data sets between HPC centers across the united states.

As speeds increase into the hundred gigabit range it becomes increasingly difficult to expect the storage and operating systems to keep up with network performance when using traditional I/O interfaces. This paper will describe some of the new and emerging technologies which Aspera utilized in scaling into and past 100gbit/s as well as how the FASP framework plugs into those solutions to provide a transfer framework that scales both into the use case of meta data intensive transfers (millions/billions of small files), as well as high throughput transfers with petabyte sized data sets.

A Design for 100 Gigabit Ethernet

Traditional POSIX I/O interfaces, which form the backbone of all I/O on every major platform suffer from designs that assume memory is fast and I/O is slow. As such they are designed to arbitrate between software that might be using hardware inefficiently and hardware that needs to be coddled to achieve maximum performance.

In practice this means massive caching between both storage and networking devices where data sits staged, waiting to be transformed and consumed by it's respective device. This waiting data obviously must be written to memory, and then re-read when a kernel device wakes up and begins to handle the associated queues. This typically means a complete flush of cachelines (The data in a CPU cache associated with an I/O operation), along with increased pressure put on the memory controller as it must write then re-fetch data.

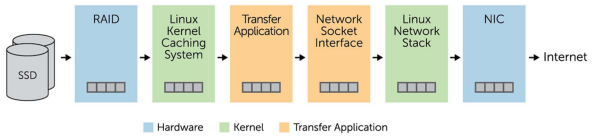

Figure 3: Memory Copies in POSIX Model

Figure 3 shows roughly what this looks like for a traditional transfer application where each of the grey bars represents a copy from one I/O queue/domain to another I/O queue/domain. The result is increased consumption of memory bandwidth and increased latencies when waiting on memory which in turn causes other processes waiting on memory to slow down as individual cores on the same physical piece of silicon compete for the same resources. This is the principal reason that transfers are slow when using traditional I/O stacks and it is a problem that compounds the more you are trying to do on the transfer node.

Exactly how slow this is depends on exactly how things are configured and what else is going on in the system. As an example, using the reference system Aspera used to show 100gbit/s transfers (XFS filesystem, 4x DC P3608, Intel® Xeon® E5-2699 v3), results in about 6 GBytes/s read with 2 or more I/O

threads, where as in direct mode (which bypasses cache), or when using Intel® SPDK (A userland storage framework, as opposed to a kernel framework) it is trivial to show results well past 12 GBytes/s on read.

Using a more traditional or featureful filesystem like EXT4 or ZFS yields reduced performance and in these cases one would expect performance closer to 2.5 GBytes/s (20 gbit/s). As one puts more pressure on the Memory Controller, the performance on the file system drops and throughput decreases. As an example using our classic FASP transfer framework in multi-threaded mode, we would see results on the same system of a little over 1GB/s (9-12 gbit/s) and this is largely due to the additional overhead caused by using the in kernel networking stack (Berkeley Sockets).

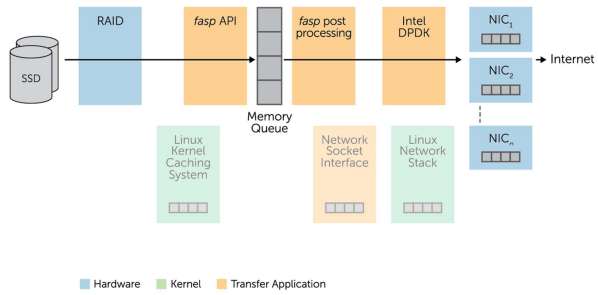

Figure 4: Zero Copy with FASP NX

In Figure 4, an Ideal I/O stack is shown. While this model doesn't eliminate the need to use memory, it fully eliminates the need to perform memory copies and with that significantly reduces the pressure put on the Memory controllers which allows the application to scale in proportion to available memory and I/O bandwidth.

We still retain the important POSIX principles such that we give good performance when doing a transfer. For instance, we read from disk into memory, and then transmit from memory which minimizes latencies when it comes time to transmit a block or in the event that we need to re-transmit a packet due to packet loss or corruption. This is almost the same as reading from disk to cache, and then from cache to application, application to NIC, only we have removed the process domains (kernel->user and vice/versa) as well as prevented double caching of data. Of course a simpler design is not to have an application Cache/queue and just send all of our data to the file system cache as soon as possible, and that is exactly what we do with standard ASCP. The only downside to that approach is that we lose the ability to control fine-grained I/O and memory performance which is important past about 10 gbit/s.

Of course it isn't enough to just eliminate memory copies and claim victory. Memory copies are just part of the problem, and as we consider that portion of the problem to be solved other issues begin to become evident such as how to handle processor intensive tasks (like compression or encryption), per-core limits to how much memory can be utilized (each core is limited to how quickly it can consume memory based on the number of hardware memory prefetchers), and how quickly we can write to I/O and NIC hardware queues.

Aspera FASP Next-generation attempts to address each of these issues through things like lockless multi-threaded architectures which avoid contention for any single object, in better utilizing Hardware offload such that CPU cores can be pinned to specific RX or TX queues on the hardware, and in how threading and encryption is handled such that we are able to fully utilize the hardware available on modern processor architectures.

One of the most interesting things about next-generation FASP is that the architecture had to change to accommodate these features. For instance, it turns out that if you want to design your own highly optimized network stack (which Aspera did for FASP Next Generation), you really can't also coexist with

the Kernel, and you almost certainly don't want to plumb in those changes into a new framework. To provide a high performance path forward, Aspera utilized Intel® DPDK to provide a userland transfer solution which eliminates the need to depend on the Kernel to provide network services. In this model, clients are able to connect to a FASP service via shared memory and initiate transfers between systems.

Intel® DPDK provides other benefits to enable applications to maximize throughput such as pinning a thread to a particular core (which eliminates losing L1/L2 caches), providing a framework to enable NUMA aware applications, and providing a burst oriented framework which enables efficient packet processing.

Optimizing for faster Storage I/O

Storage has undergone a radical transformation as file systems transition from single large systems to clustered and cloud based object stores. In these configurations, metadata is typically separated from the file payload and specific access conditions are typically required to fully utilize the throughput offered by clustered parallel file system and or cloud based object storage.

In both FASP and next-generation FASP, a filesystem abstraction layer is provided which provides flexibility in both how metadata is written and in how file data is written to disk. The two are handled independently and each stage of the I/O chain can be customized to provide the integration you need to fully maximize the potential of your storage solution.

For instance, if your local storage solution requires that you read/write with specific block sizes and you have slow meta-data retrieval you can pass Aspera FASP exactly those hints and it will eliminate all un-necessary metadata queries and perform disk I/O with fixed block sizes.

In the same vein, if you are transferring to or from cloud that same file system abstraction is available to translate from traditional file systems to object based file systems where data is stored through REST API calls.

The same holds true if you are attempting to transfer millions of very small files. In this use case, you can have multiple readers which do nothing but read metadata information along with the file contents and once they have packed enough data into a block they then send the entire block across the internet where the files are written as quickly as possible.

Even in use cases where we are doing transfers on behalf of a system agent where it is expected that we transfer files and then transform that data and interact with system services (as in Aspera's Avid Integration: http://asperasoft.com/partners/joint-partner-solutions/fasp-plug-in-for-avid), Aspera provides a flexible enough framework that the integration point is just going to be mapping API's from one system to another.

In most cases, the only optimization needed for good performance is to use the storage system as designed. However in some cases it isn't enough to just go fast, you need to maximize your storage utilization and the POSIX overhead is creating a barrier for efficient utilization. In these cases we are able to take advantage of Direct I/O when possible to eliminate a copy to cache in use cases where it is not expected that the data being cached will be used again. Aspera can utilize Direct I/O in a number of use cases such as, when we are using RDMA or Intel® Omni-Path, When the file system supports it (i.e. XFS), or when we can take advantage of Intel® QuickData or similar userland transfer technologies.

Aspera provides solutions which optimize both network I/O and storage I/O in a way that is vendor neutral and high performance. Aspera's goal is to provide solutions which solve real problems in both sending data over the network and in efficiently reading and writing that data to disk. In addition Aspera provides the safeguards you need to be sure that the data you created is securely sent across the Internet using industry standards as well as safeguarding the integrity of your data with end-to-end checksums.

A Flexible High Performance FASP Solution

The goal of Aspera FASP is to provide a transfer solution which efficiently utilizes your unused bandwidth to move content as quickly as possible. In the end, be it sending 8k uncompressed video @ 60 FPS (~26 Gbps) in near real time, or replicating your data center complete with disk images which need to be compressed before being sent, Aspera has a solution for you.

All of our solutions are built around the same core FASP framework, they are built to be fast, and they are meant to be a solution to your problems. Since 2004 we have been approaching the problem of how can we send faster, how can we provide more flexibility, and how can we better meet the needs of prospective customers. In addition to our core FASP protocol we have a number of tools to try to make our software easier to use. These include things like web apps for Collaboration, Management, Sharing, and Distribution of content. Tools for synchronizing massive amounts of data, along with automation and policy tools.

Aspera optimizes the FASP protocol across the entire system domain, starting from Storage and ending with people consuming data. At almost every level of the transfer chain, you can tweak things to provide the policy you need within your organization. This includes things like different transfer priorities for different users, virtual links to shape traffic how you want, API's and file notification hooks.

At Aspera our goal is to provide the transfer solution you want and we are more than happy to work with you until things are exactly what you are looking for. Our solutions currently scale to hundreds of gigabits/s per system when utilizing our in development Next-Generation FASP, and they scale out to utilize as many systems as you have when performing parallel transfers.

Cloud, Hybrid, or On Premise Aspera FASP provides the performance and throughput to minimize the time you spend waiting on content and maximize your ability to consume and provide content.