CCGRID 2019, LARNACA, CYPRUS. 15TH, MAY, 2019

Argonne NATIONAL LABORATORY

DATA TRANSFER BETWEEN SCIENTIFIC FACILITIES -- BOTTLENECK ANALYSIS, INSIGHTS, AND OPTIMIZATIONS

NAGESWARA S.V. RAO

YUANLAI LIU, ZHENGCHUN LIU, RAJKUMAR KETTIMUTHU, NAGESWARA S.V. RAO, ZIZHONG CHEN, IAN FOSTER

INTRODUCTION

- Massive amount of data is being generated by scientific facilities
- Data needs to be transferred to different locations for analysis
  - HACC generates 20PB data per day, and move data to other sites for analysis
- DOE's ESnet provides connectivity to many science facilities in USA
  - Bandwidth is 100 Gbps or more
- Many tools have been developed for file transfers, including GirdFTP
  - GridFTP is widely used for large science transfers
  - GridFTP is an extension of the standard FTP protocol
  - GridFTP provides high performance, better security, and improved reliability
  - GridFTP uses different number of server processes (named concurrency), depending on the number and sizes of files in a transfer request
  - Globus is a software-as-a-service cloud tool that transfer file on nodes running GridFTP server
  - Globus is a software-as-a-service cloud tool that transfer file on nodes running GridFTP server

INTRODUCTION

- We characterized approximately 40 billion files totaling 3.3 Exabytes transferred by real users using GridFTP and 4.8 million dataset transferred by using Globus transfer service
  - 90% of the total bytes transferred with more than one file
  - 63% of the total bytes transferred with more than 1000 files
  - 42% of the total bytes transferred with more than 10000 files

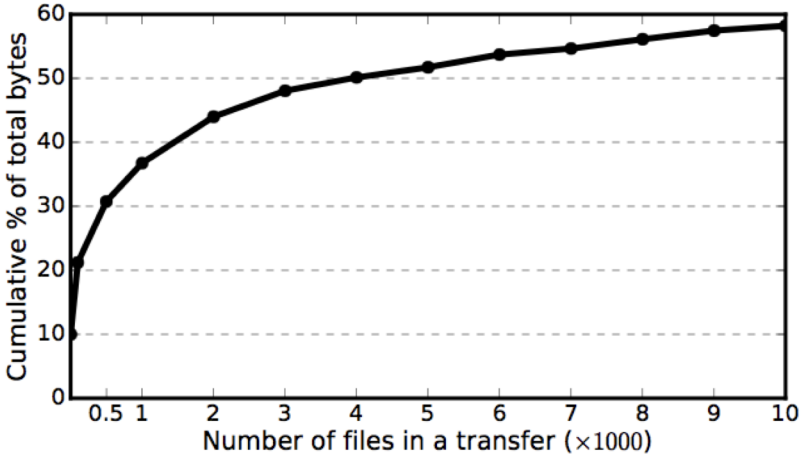


Fig. 1: Cumulative distribution of total bytes transferred using Globus by the number of files in a transfer, from 2014 to 2017.

BACKGROUND

- Petascale DTN project, formed in 2016:
  - Comprising of staff at Energy Science Network (ESnet) and four supercomputing facilities:
  - Project goal: to achieve a wide area file transfer rates of about **15 Gbps**
  - Benchmark dataset: A real world cosmology data set (L380)
  - Benchmark tool: **Globus** transfer service
- Current rate is great but still not perfect, so we are interested in understanding the **current bottleneck**

(a) March 2016

Destination \ Source	ALCF	NCSA	NERSC	OLCF
ALCF	–	8.2	7.3	11.1
NCSA	13.4	–	7.6	13.3
NERSC	10.0	6.8	–	6.7
OLCF	10.5	6.9	6.0	–

(b) June 2017

Destination \ Source	ALCF	NCSA	NERSC	OLCF
ALCF	–	21.2	27.2	23.0
NCSA	19.4	–	20.2	15.1
NERSC	22.9	11.8	–	19.7
OLCF	25.7	15.2	20.6	–

(c) November 2017

Destination \ Source	ALCF	NCSA	NERSC	OLCF
ALCF	–	56.7	42.2	47.5
NCSA	50.0	–	33.7	43.4
NERSC	35.0	22.6	–	33.1
OLCF	46.8	34.7	39.0	–

Table 1: Data transfer rates (Gbps) among four major supercomputing facilities as various optimizations were applied over time

BOTTLENECK ANALYSIS

- Testbed
  - Two of the four sites involved in the Petascale DTN project, ALCF and NERSC
  - ALCF has a 7P GPFS and NERSC has a 28P Lustre filesystem
  - 100Gbps wide area connection between ALCF and ESNet
  - 80Gbps connection between NERSC and ESNet
  - Round trip time between ALCF and NERSC is about 45ms
  - ALCF has 12 Data Transfer Nodes (DTN), each has one Intel Xeon E5-2667 v4 @3.20GHz CPU, 64GB of RAM and one 10Gbps NIC
  - NERSC has 10 DTNs, each DTN has two Intel Xeon E5-2680 v2 @2.80GHz CPU, 128GB of RAM and one 20Gbps NIC

BOTTLENECK ANALYSIS

- Dataset
  - For our analysis we generated a dataset whose file size distribution is similar to that of all production GridFTP transfers, consists **59,589** files totaling **1TB**, noted as DS_{real} , the dataset size can be varied by simply adjusting the number of files sampled
  - We created a dataset that is of the same size as DS_{real} but had just enough number of files(128) to utilize all the concurrent processes(64) used for data transfer using Globus. We refer to this dataset as DS_{big}.
  - Fig. 3 result indicates that the file size characteristics and/or number of files have significant influence on transfer performance

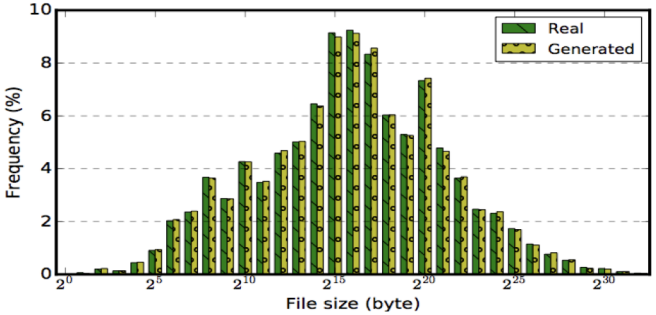


Fig. 2: Distribution of dataset file size, generated versus real.

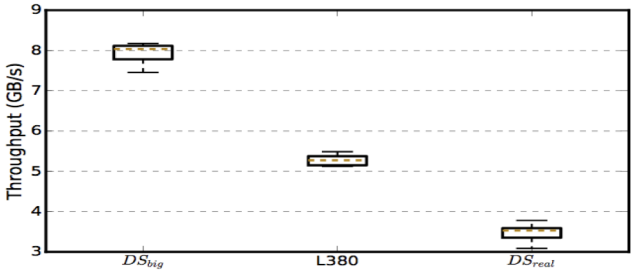
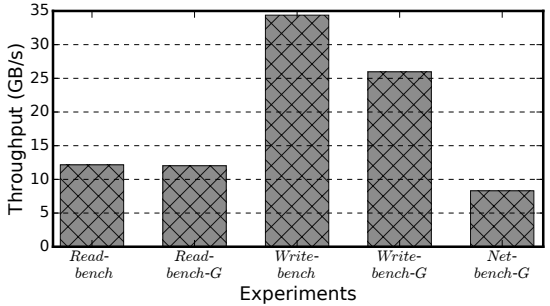


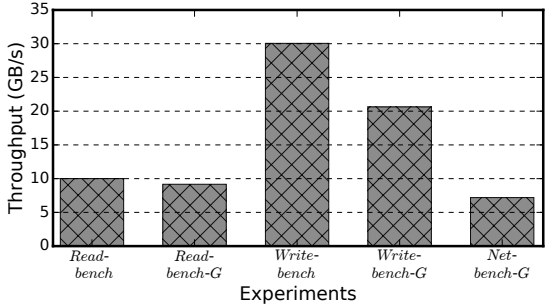
Fig. 3: Comparison of transfer performance for the DS_{big}, L380, and DS_{real} datasets between ALCF and NERSC.

BOTTLENECK ANALYSIS

- Benchmark storage read performance at the source and write performance at the destination with and without using the transfer tool
- Benchmark network by transferring *N* equally sized *dev/zero* at NERSC to */dev/null* at ALCF
- Bottleneck is in fact the **network** and not the source or destination storage for both the DS_{big} and DS_{real} datasets
- There is a noticeable drop in performance for DS_{real} compared to DS_{big} for each case benchmarked
- Indicated that there is a **per-file** overhead in storage read, storage write and the network



(a) Testing using DS_{big}



(b) Testing using DS_{real}

Fig. 4: Storage and network benchmark for file transferring.

FURTHER INSIGHTS

- Break down the overhead for each subsystem to identify directions for optimization
  - Storage read overhead – overhead introduced by (previous) file close and (next) file open at the source (O_R);
  - Storage write overhead – overhead introduced by (previous) file close and (next) file open at the destination (O_W);
  - Network overhead – overhead caused by TCP dynamics due to discontinuity in data flow caused by O_R and/or O_W (O_N);
- $max(O_R, O_N, O_W) <= O_{overrall} <= O_R + O_N + O_W$
- Assume that each file introduces a fixed overhead of t_0, the network throughput is R. Thus, the time T to transfer N files total B bytes will be:

$$T = N * t_0 + B/R \qquad (1)$$

FURTHER INSIGHTS

- To verify Equation (1), we performed a series of experiments.
- We kept the total dataset size same for all experiments but varied the number of files in each experiment. Result:

  $T = 0.0665N + 16.5$

- It implies that the per-file overhead is 66.5ms, and this overhead is the cause for the performance drop.

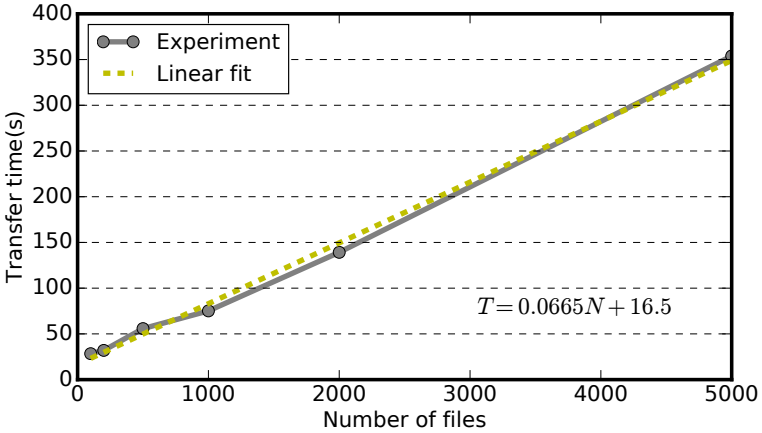
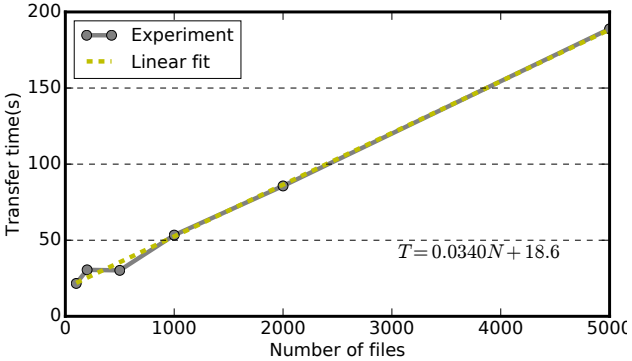


Fig. 5: Transfer time as a function of the number of files for transfer of files between NERSC and ALCF. Transfer size is 5GB.

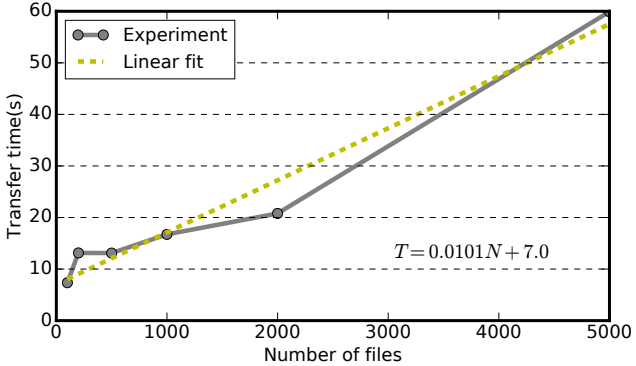
FURTHER INSIGHTS

- O_R = 34.0 ms
- O_W = 10.1 ms
- O_N = 25.3 ms

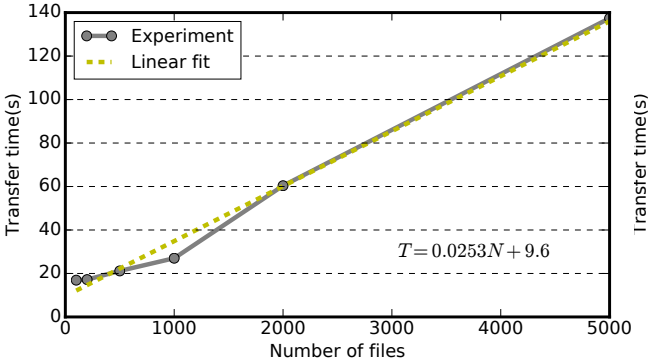
- max(O_R, O_N, O_W) = 34 ms
- $O_R + O_N + O_W$ = 69.4ms
- $O_{overrall}$ = 65.5 ms



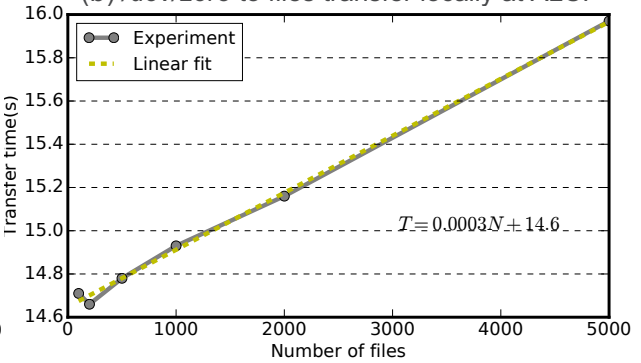
(a) files to */dev/null* transfer locally at NERSC



(b) */dev/zero* to files transfer locally at ALCF



(c) */dev/zero* to */dev/null* transfer over WAN between NERSC and ALCF



(d) */dev/zero* to */dev/null* transfer locally at NERSC

CONCURRENT TRANSFERS

- Concurrent transfers will help improve the performance of transfers with many files
- Beyond a certain value, increasing concurrency can harm performance, determining the “just right” concurrency is hard because of the dynamic environment
- Study how concurrent transfers of multiple files can help reduce the average per-file overhead for each subsystem
- Perform transfer experiments using the representative dataset DS_{real} from NERSC to ALCF

CONCURRENT TRANSFERS

Storage read

- Transfer DS_{real} from the parallel file system at NERSC to */dev/null* locally with varying number of concurrent file transferring

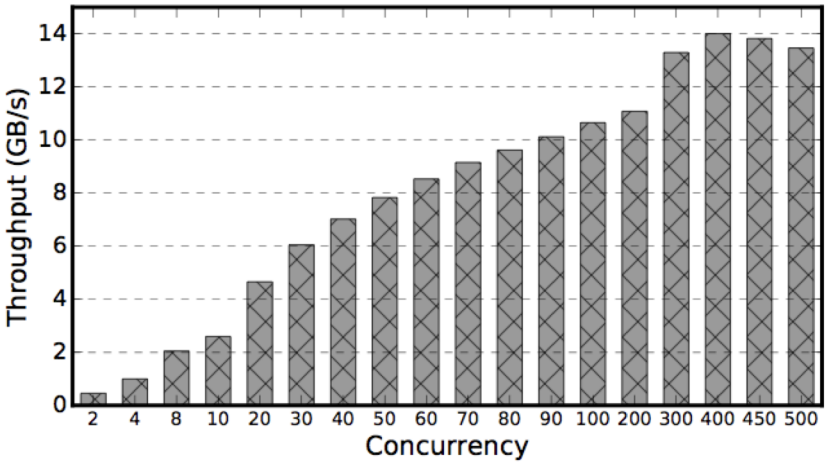


Fig. 6: Lustre read performance test using globus-url-copy

CONCURRENT TRANSFERS

Network

- Transfer from */dev/zero* at NERSC to */dev/null at* ALCF with varying concurrency
- The perf-file overhead is possible to be suppressed with enough concurrency

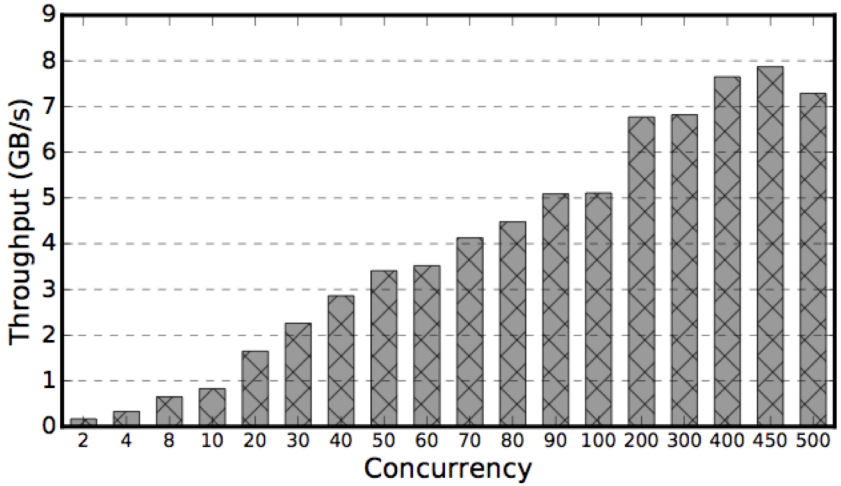


Fig. 7: Transfer files on Lustre at NERSC to */dev/null* at ALCF DTNs.

CONCURRENT TRANSFERS

Storage write

- Transfer data from /dev/zero to the parallel file system locally at ALCF
- Write 59,589 equally sized files totaling 1TB with different concurrency.

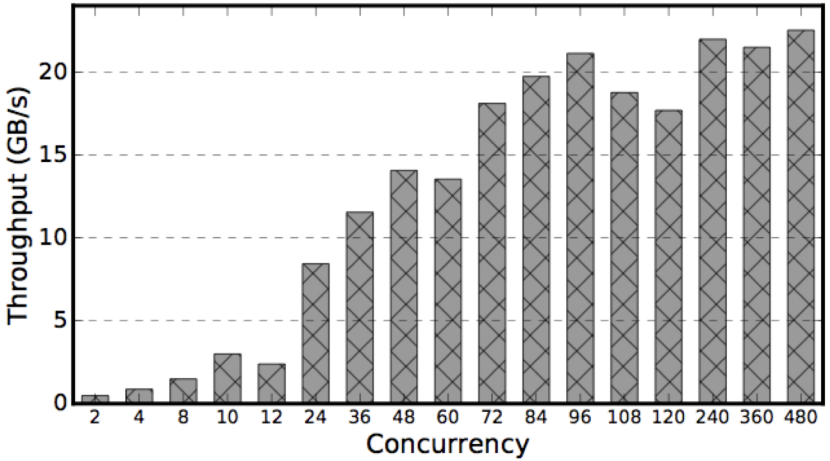


Fig. 8: Transfer from */dev/zero* at ALCF DTNs to files on GPFS at ALCF

CONCURRENT TRANSFERS

- End-to-end file transfer
- Transfer DS_{real} from the parallel file system at NERSC to the parallel file system at ALCF
- Figure 9 is almost identical to Figure 7, because **network** is the bottleneck in both cases

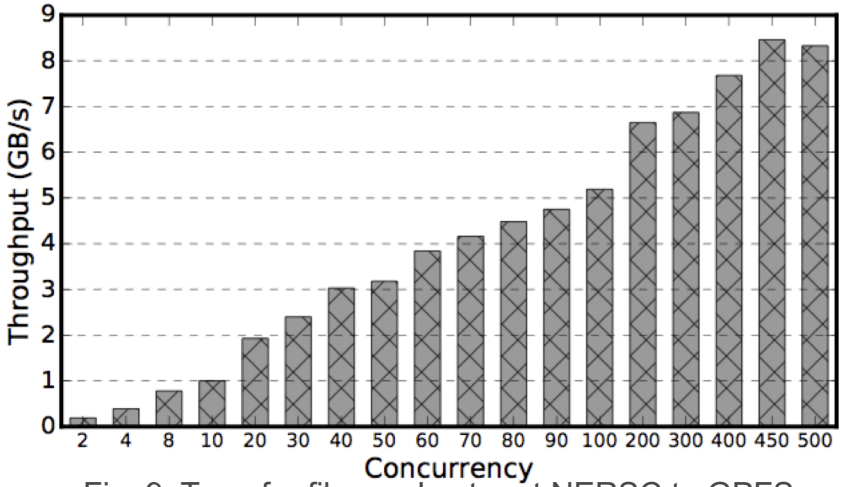


Fig. 9: Transfer files on Lustre at NERSC to GPFS at ALCF.

PREFETCHING – MOTIVATION

- Fig. 10 shows the total CPU utilization (in core*seconds) to transfer a given dataset with different concurrency.
- Although high levels of concurrency achieves better performance, it consumes more CPU as well and thus can negatively impact other transfers.
- Another approach to reduce the per-file overhead is **prefetching**.

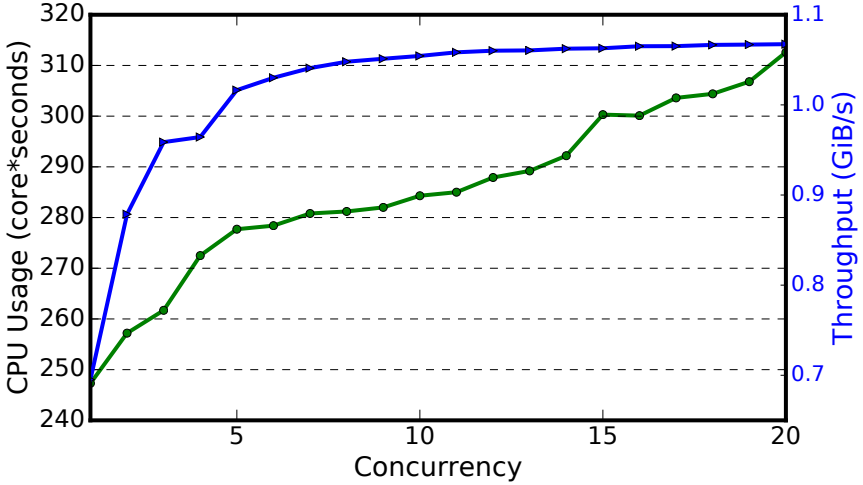


Fig. 10: CPU utilization vs. transfer concurrency.

PREFETCHING – ALGORITHM

- Prefetch one or more blocks of the *Nextfile*, during the transfer of a file.
- So we can start transferring the *Nextfile* immediately upon completion of the ongoing file transfer, avoiding the overhead mentioned above.
- we do the prefetching only when the ongoing transfer has filled the TCP send buffer.

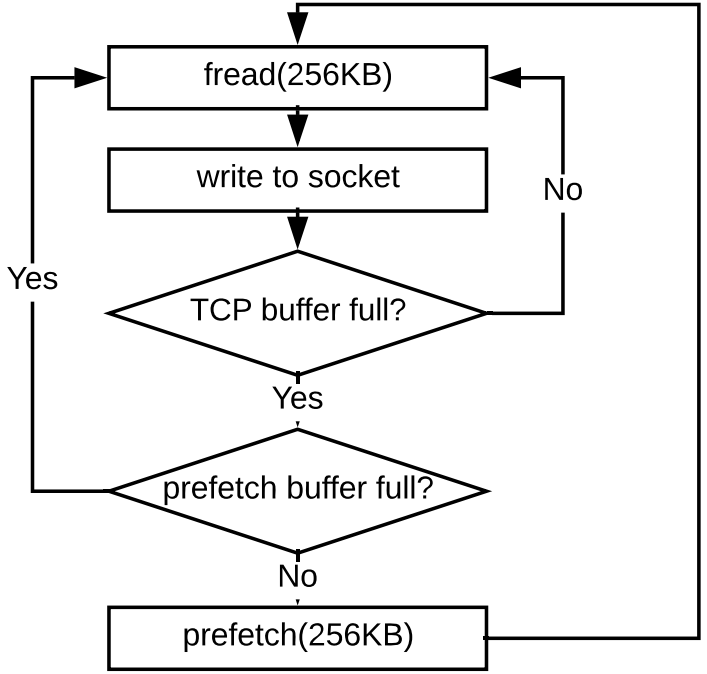


Fig. 11: Flow diagram of the prefetching approach

PREFETCHING – RESULTS

- Fig.12 shows the effectiveness of prefetching using multiple 80GB transfers, each one with different number of files.
- The transfer time increases much slowly with increasing number of files when prefetching is enabled.
- Thus, prefetching can help reduce the per-file overhead significantly.

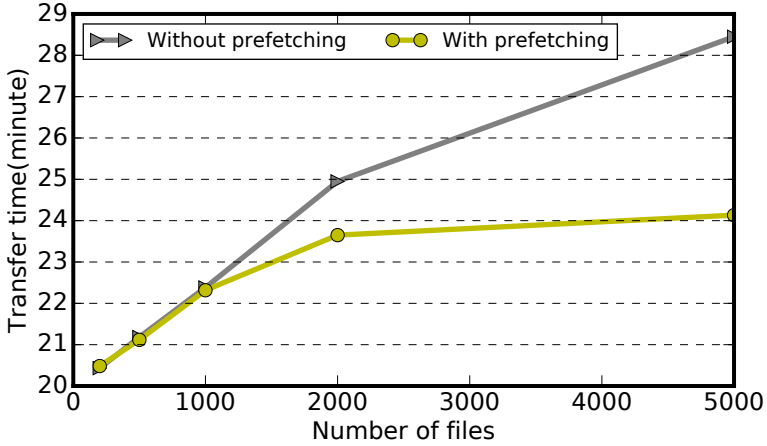


Fig. 12: Transfer time as a function of number of files for 80GB transfers from NERSC to ALCF.

PREFETCHING

- Fig. 13 shows the throughput for 2TB dataset (containing 50,000 files) transfers with and without prefetching for different concurrency values.
- It is clear that prefetching helps achieve a higher throughput with **less** concurrency.

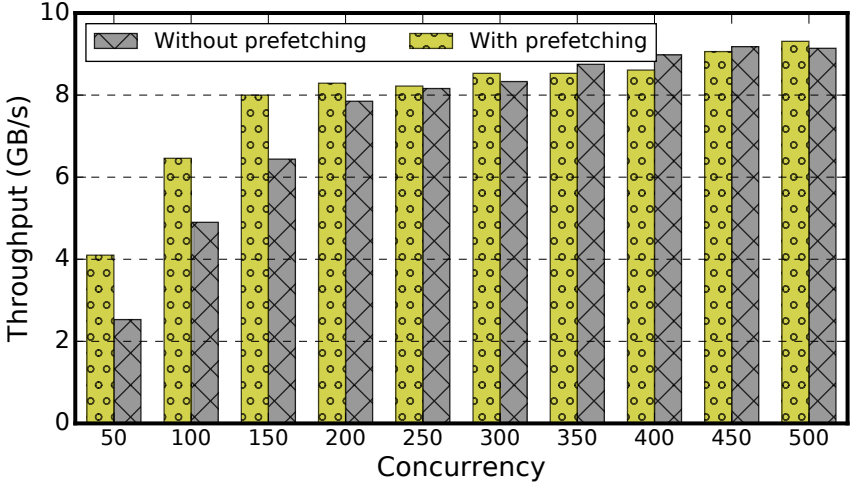


Fig. 13: Transfer files on Lustre at NERSC to GPFS at ALCF.

THANKS