From System Logs to System Optimization with the Power of Data Science and Machine Learning

Zhengchun Liu

Research Scientist at the University of Chicago

June 10, 2019
Seminar at Argonne National Laboratory

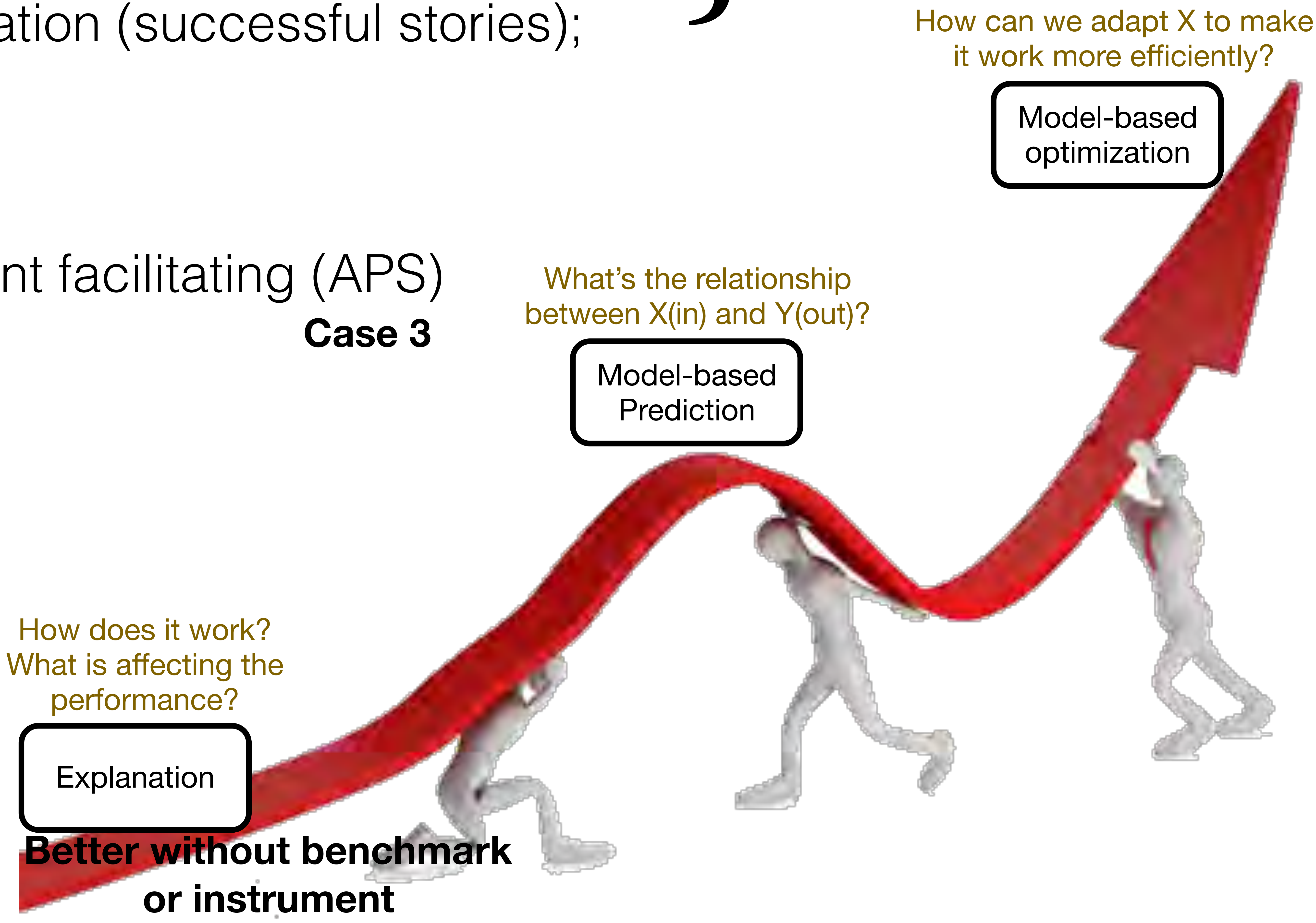
Agenda

- Explain wide area file transfer performance in a quantitive way;
- Characterize file transfer and its infrastructure from logs;
- Transfer information into knowledge for optimization (successful stories);

Case 1

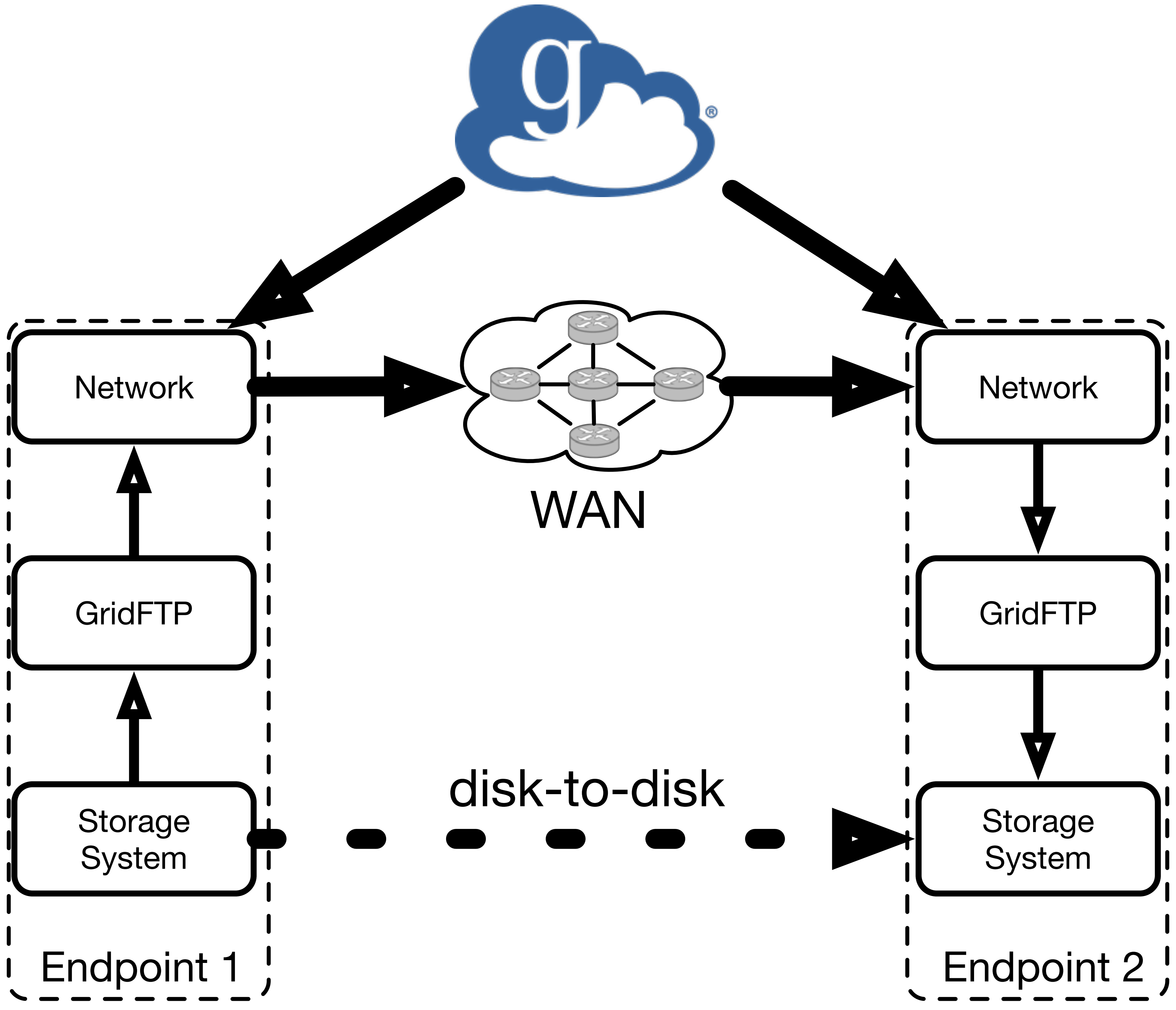
- Computing Facility logs (ALCF); **Case 2**
- Lightsource facility data analysis and Experiment facilitating (APS)

Case 3



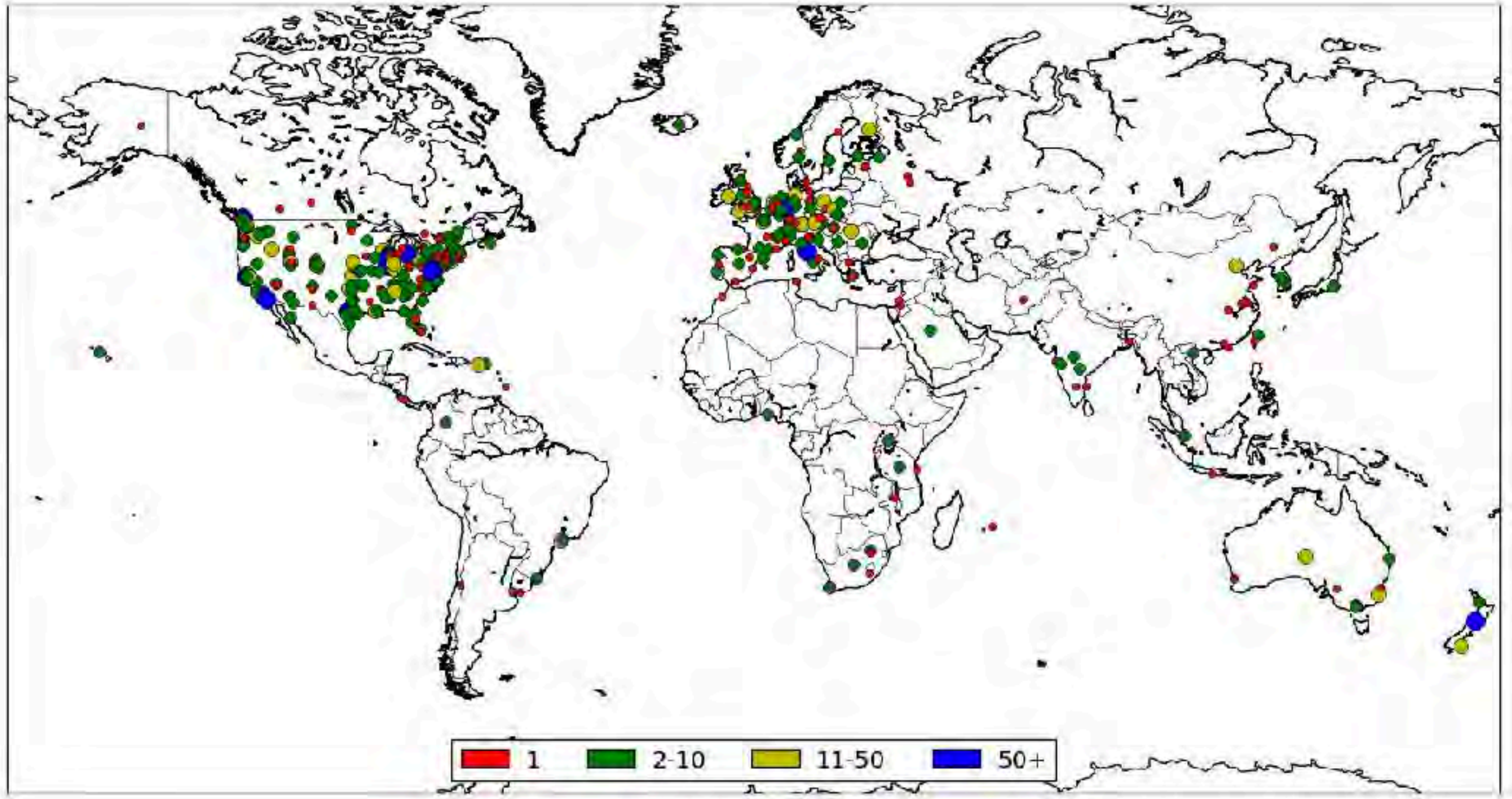
Get a deep understanding of end-to-end file transfer performance
(Explain)

Introduction - globus.org



The Globus transfer service is a cloud-hosted software-as-a-service, to provide convenient, reliable and secure file transfers service between pairs of storage systems

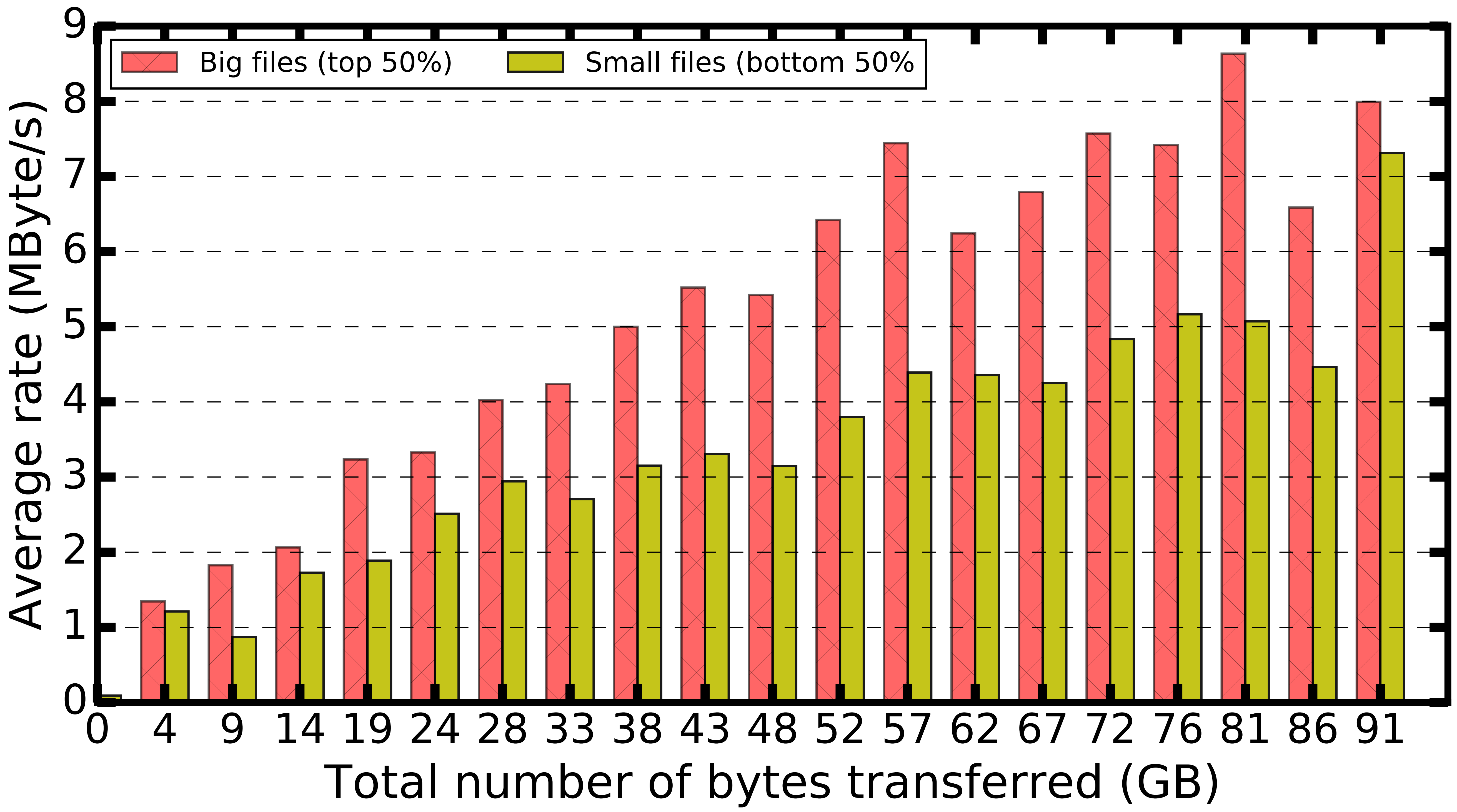
$$R^{max} \leq min(DR^{max}, MM^{max}, DW^{max})$$

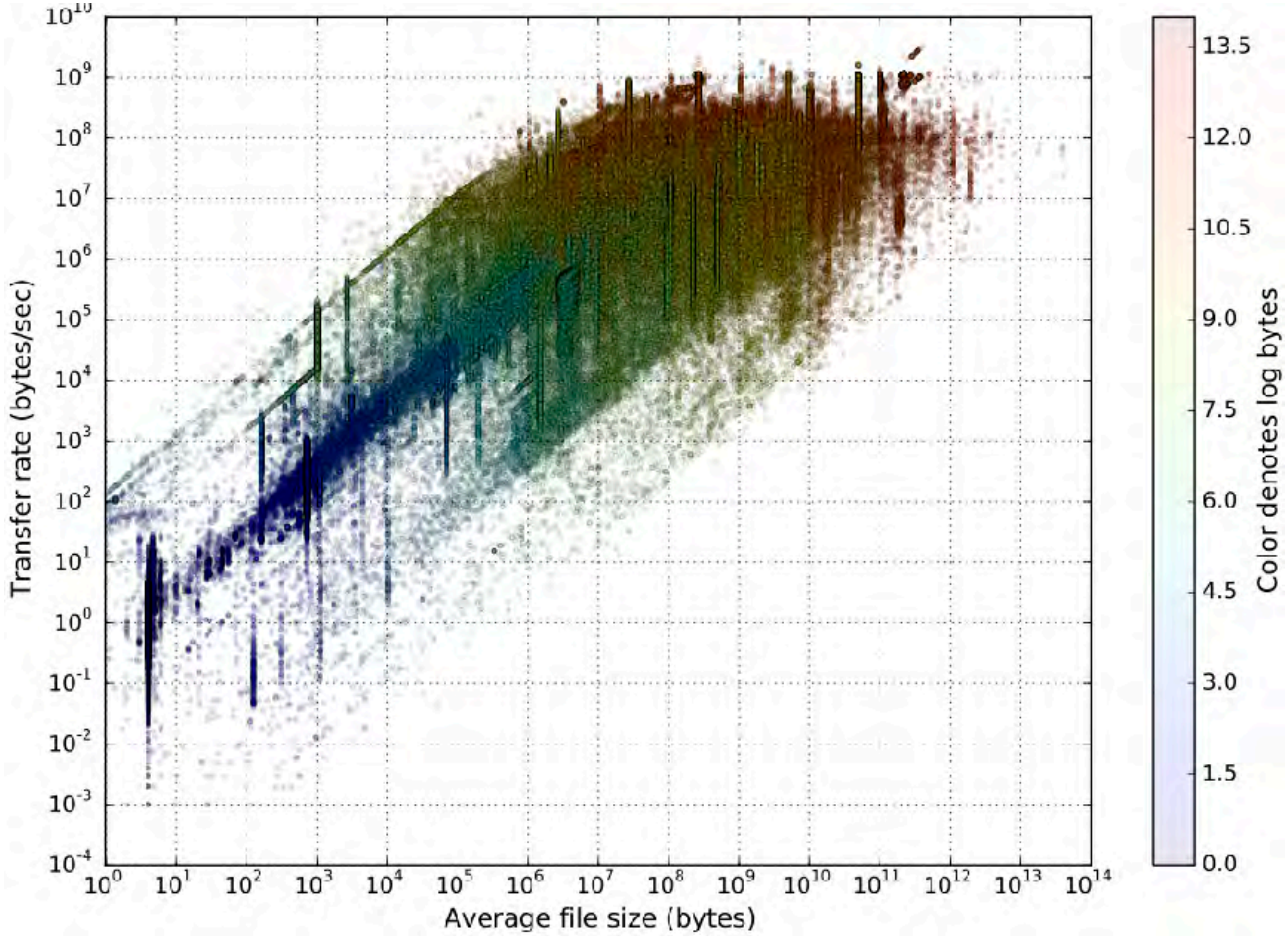


Globus endpoints, grouped by number of deployments in a single location. (Some endpoints geolocate erroneously to the center of countries.)

What affect transfer performance? -1

File characteristics:



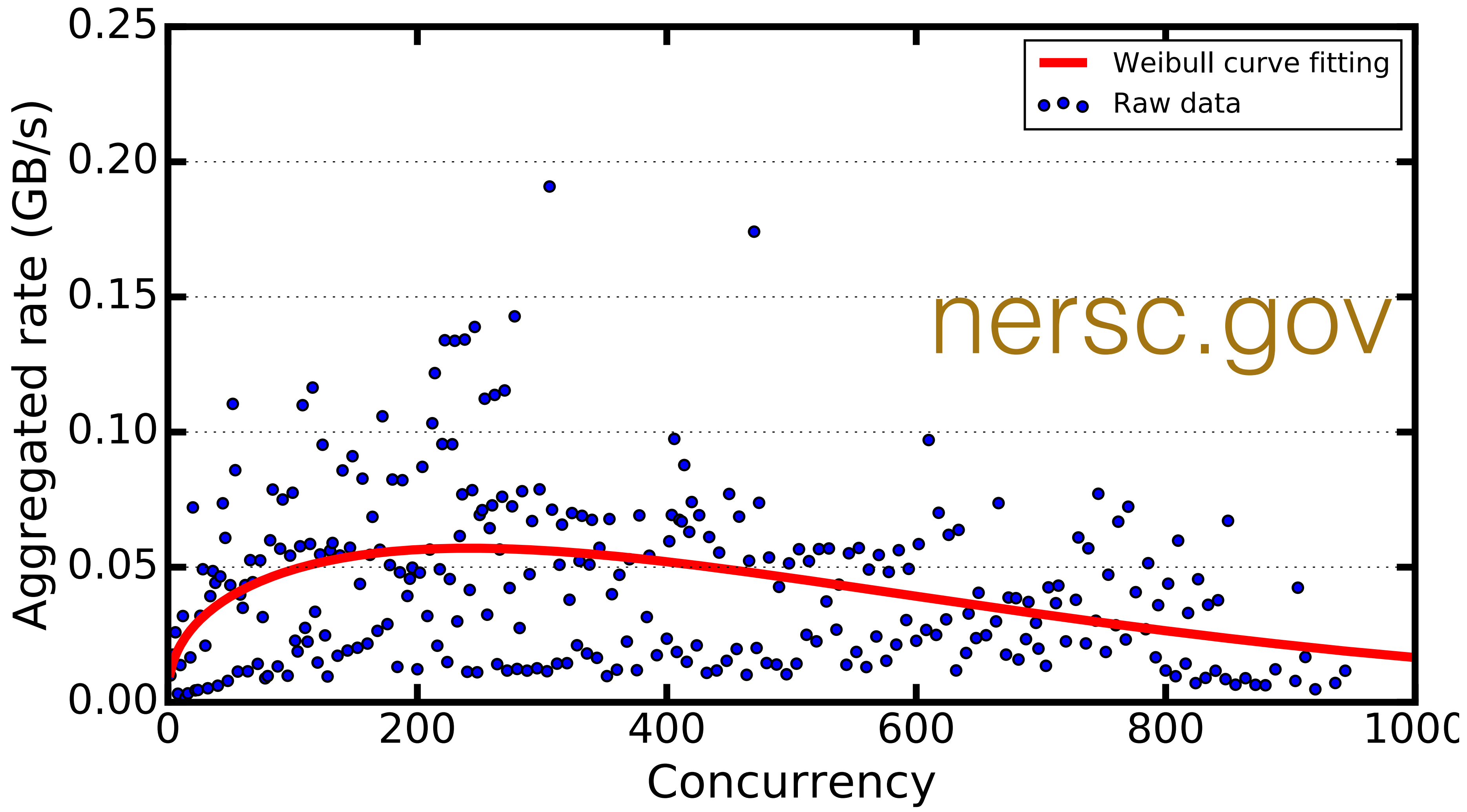


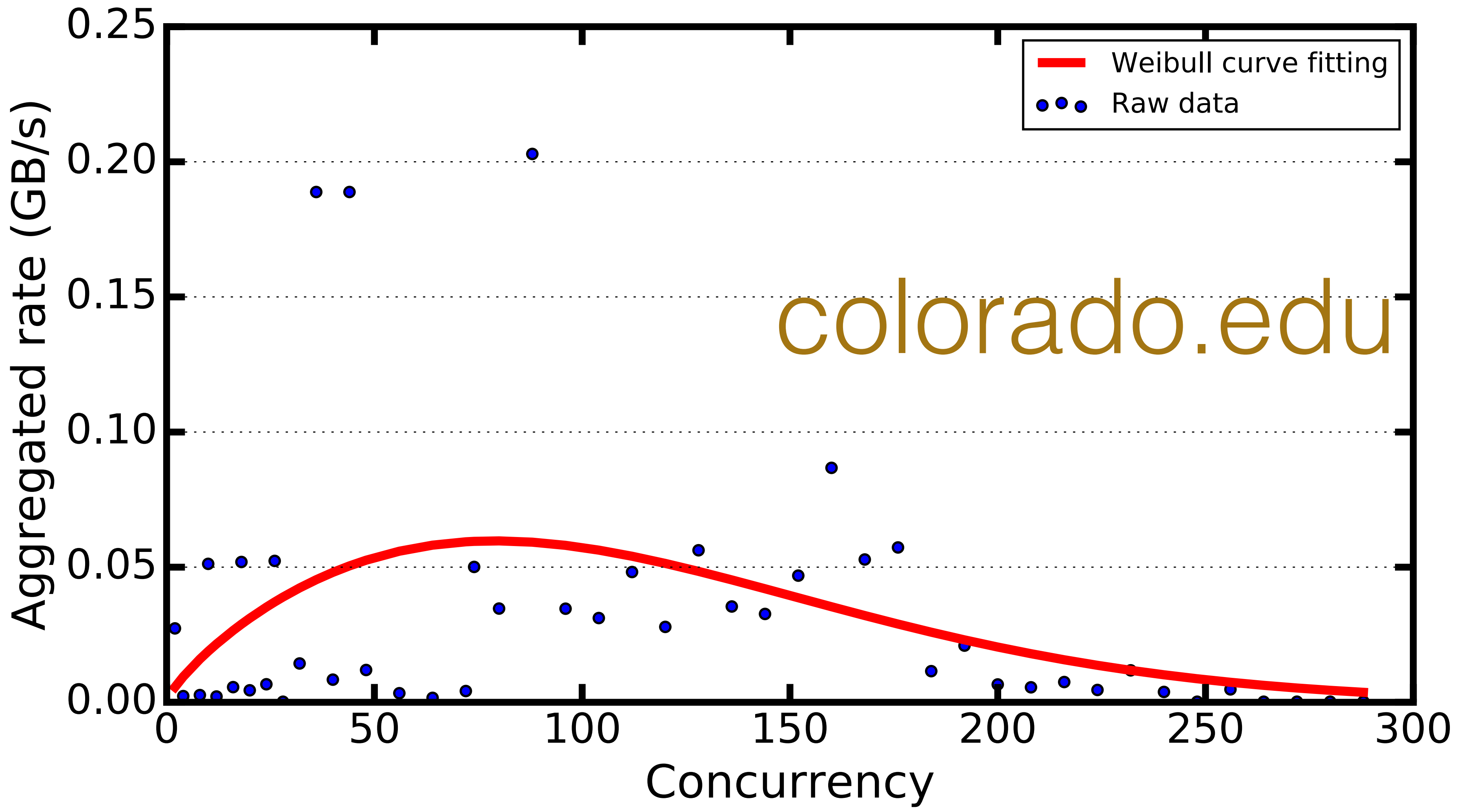
Large transfers with big average file size are more likely to have better performance.
I.E, The startup cost is high.

Liu et al. HPDC'17

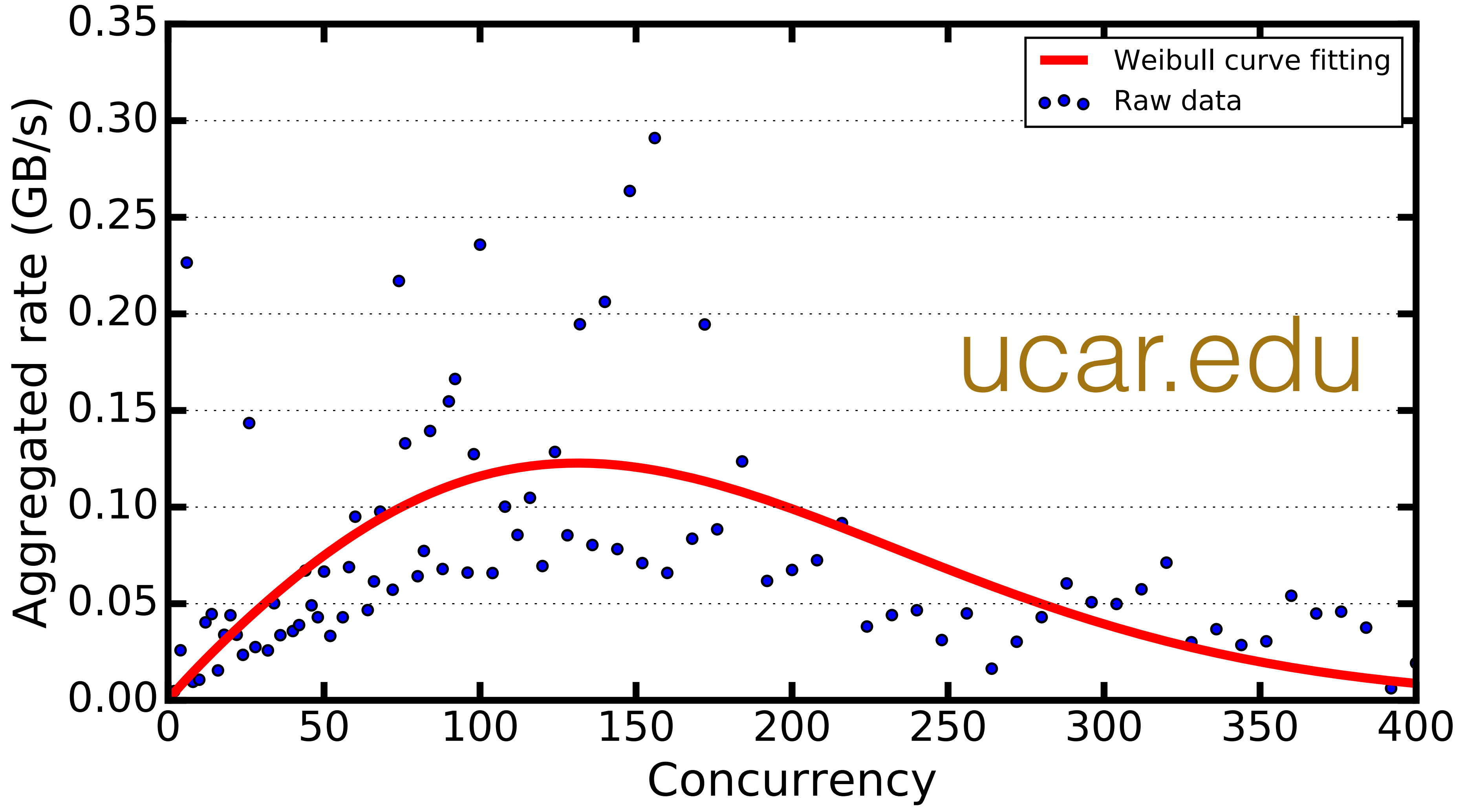
What affect transfer performance? -2

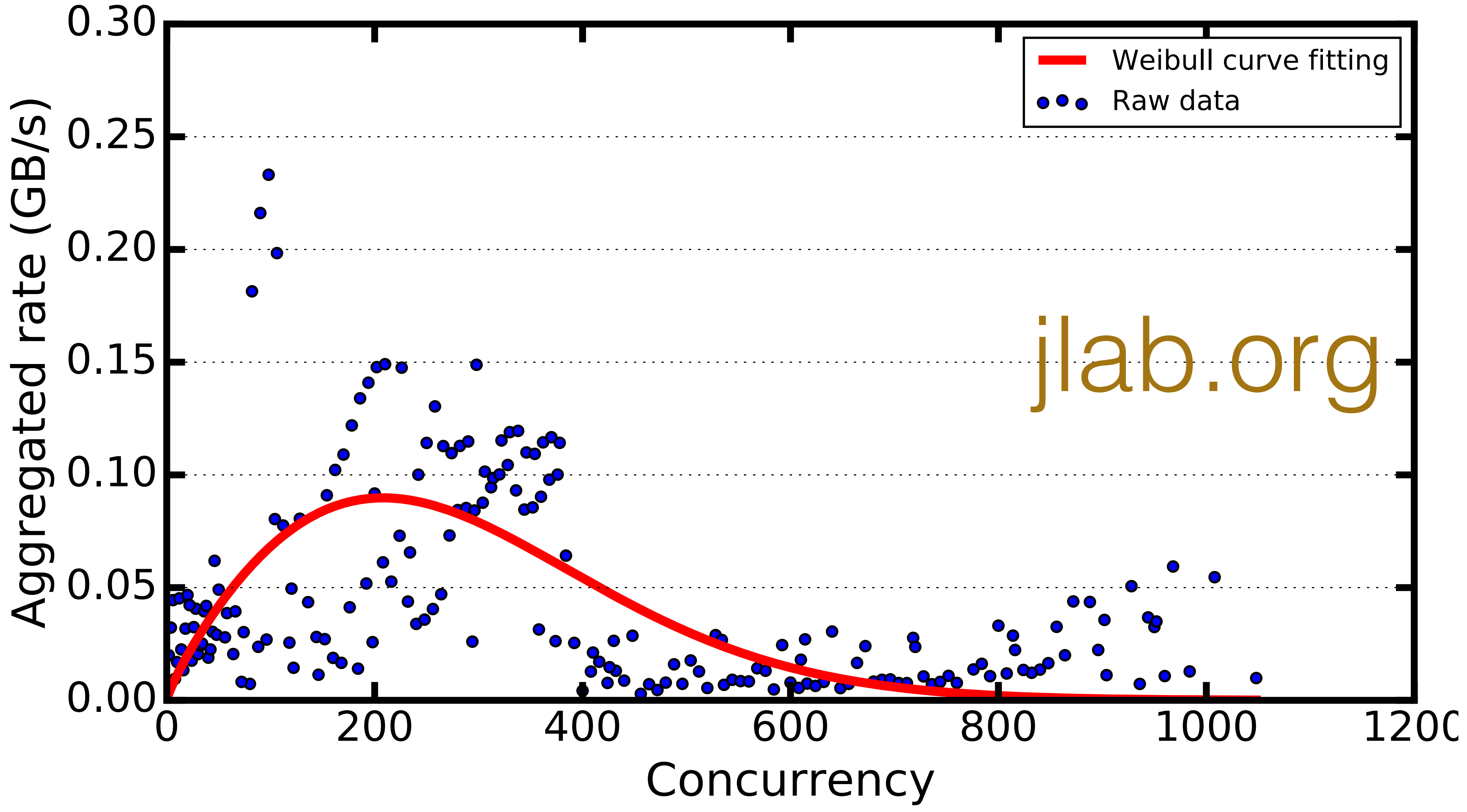
Tunable transfer parameters





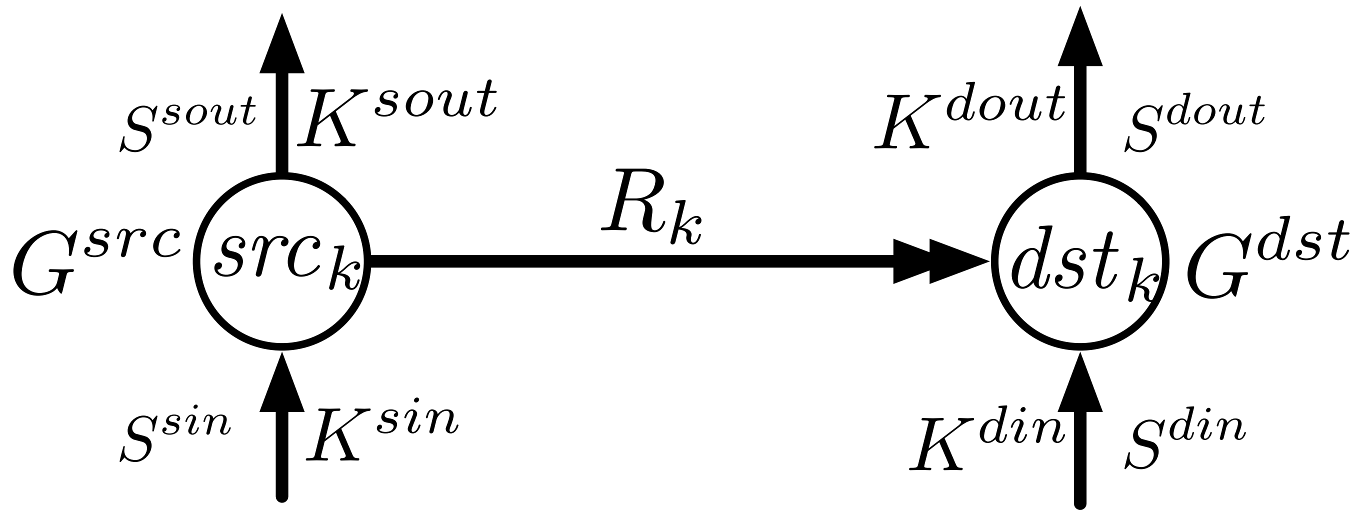
Aggregated concurrency ***versus*** aggregated throughput on the data transfer node.





What affect transfer performance? -3

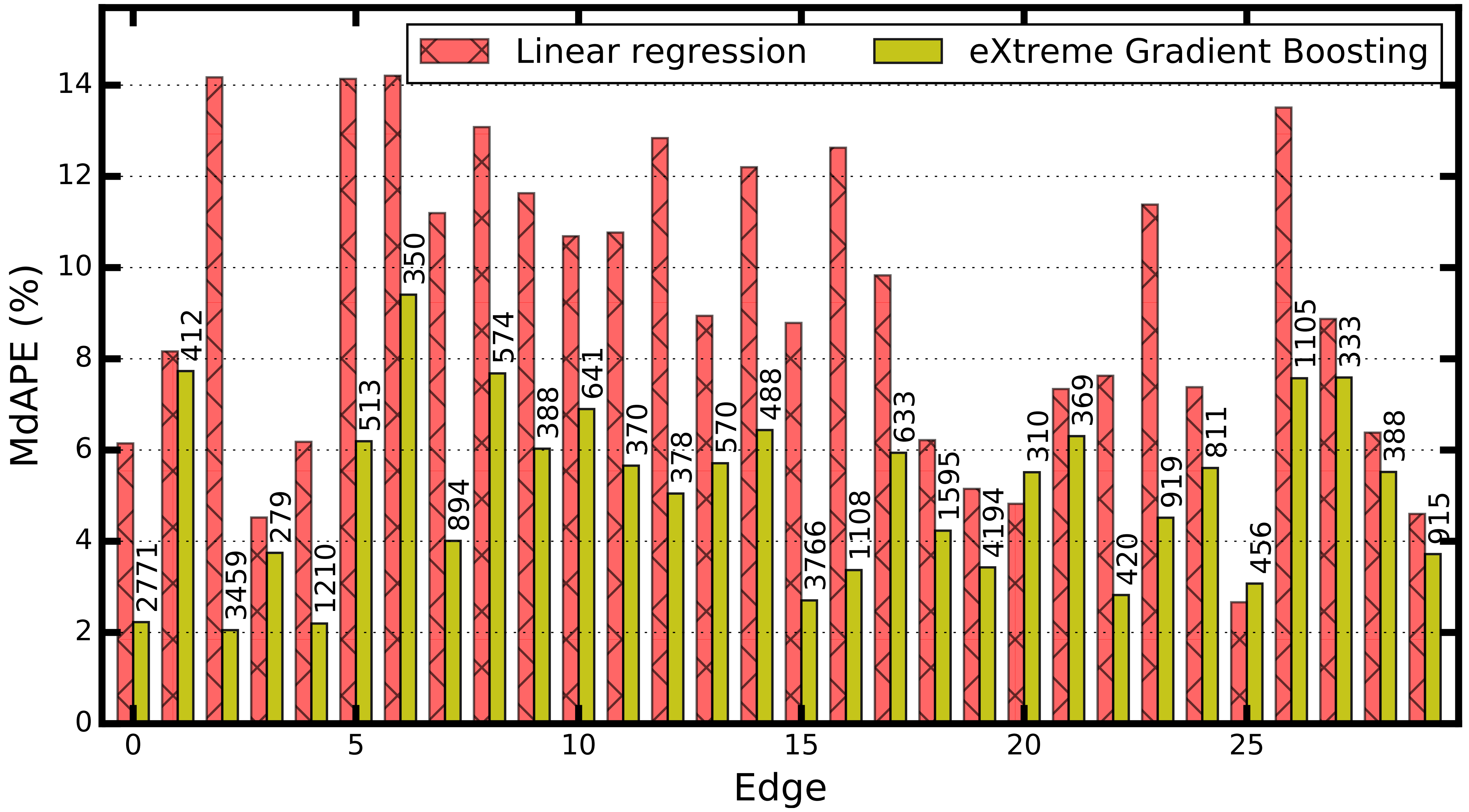
Contention from simultaneous *globus* transfers (I/O, NIC, CPU & RAM):



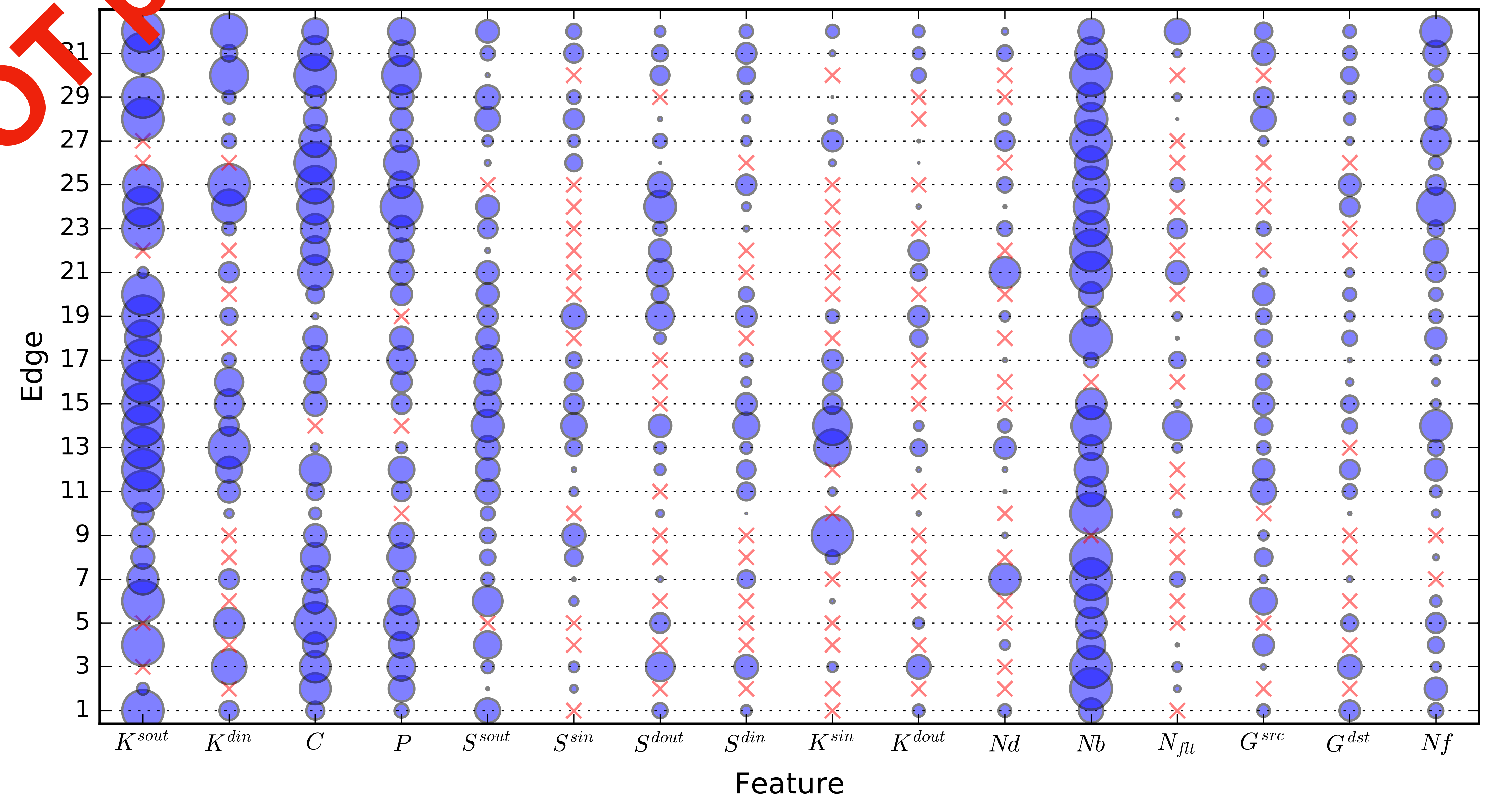
K^{sin}	Contending incoming transfer rate on src_k.
K^{sout}	Contending outgoing transfer rate on src_k.
K^{din}	Contending incoming transfer rate on dst_k.
K^{dout}	Contending outgoing transfer rate on dst_k.
G^{src}	GridFTP instance count on src_k.
G^{dst}	GridFTP instance count on dst_k.

S^{sin}	Number of incoming TCP streams on src_k.
S^{sout}	Number of outgoing TCP streams on src_k.
S^{din}	Number of incoming TCP streams on dst_k.
S^{dout}	Number of outgoing TCP streams on dst_k.

It is NOT perfect



Model based feature interpretability



Model based feature importance

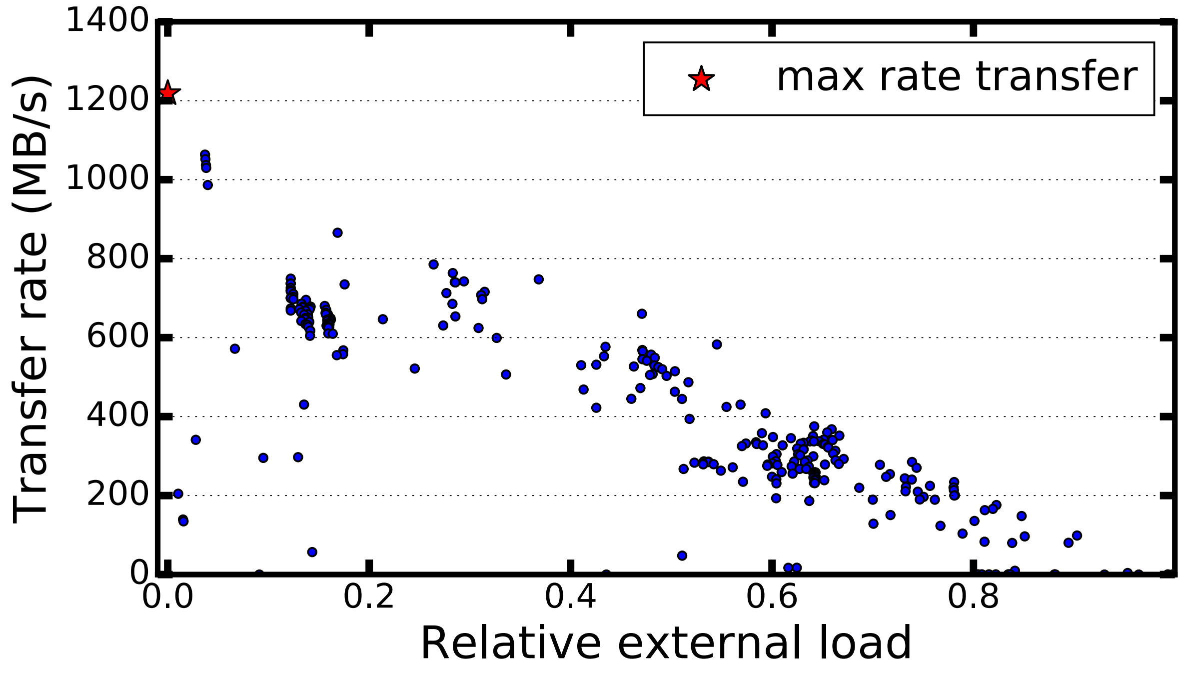
What affect transfer performance? -4

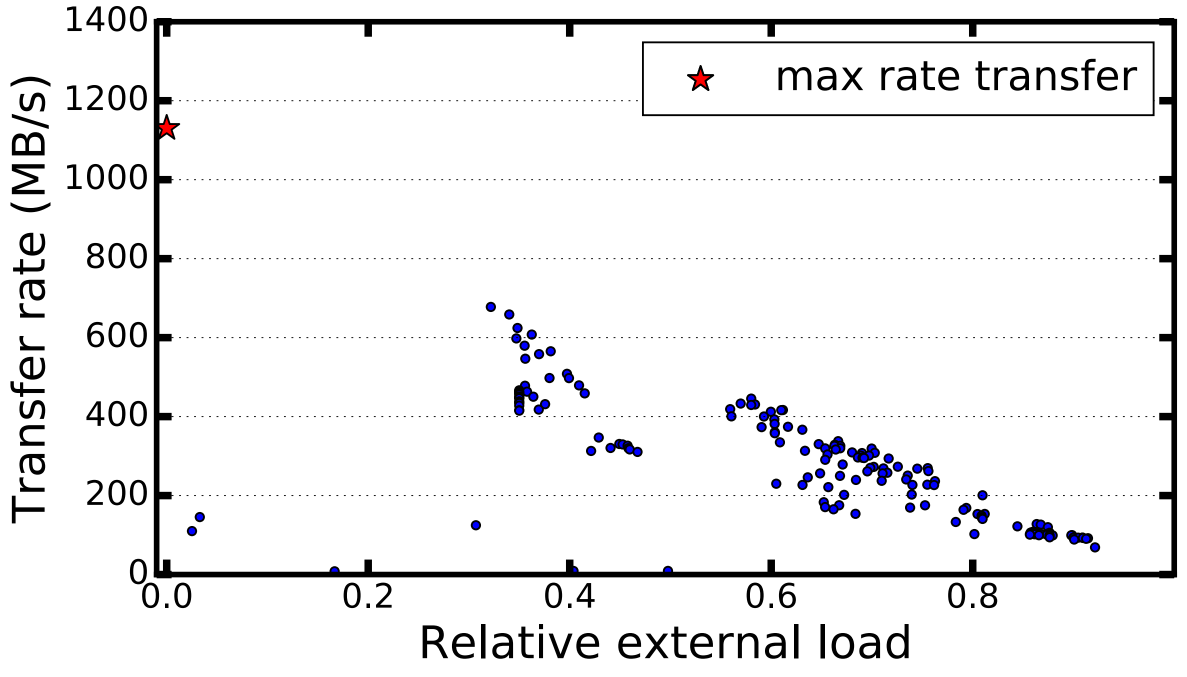
Contention from non-globus programs (shared environment)

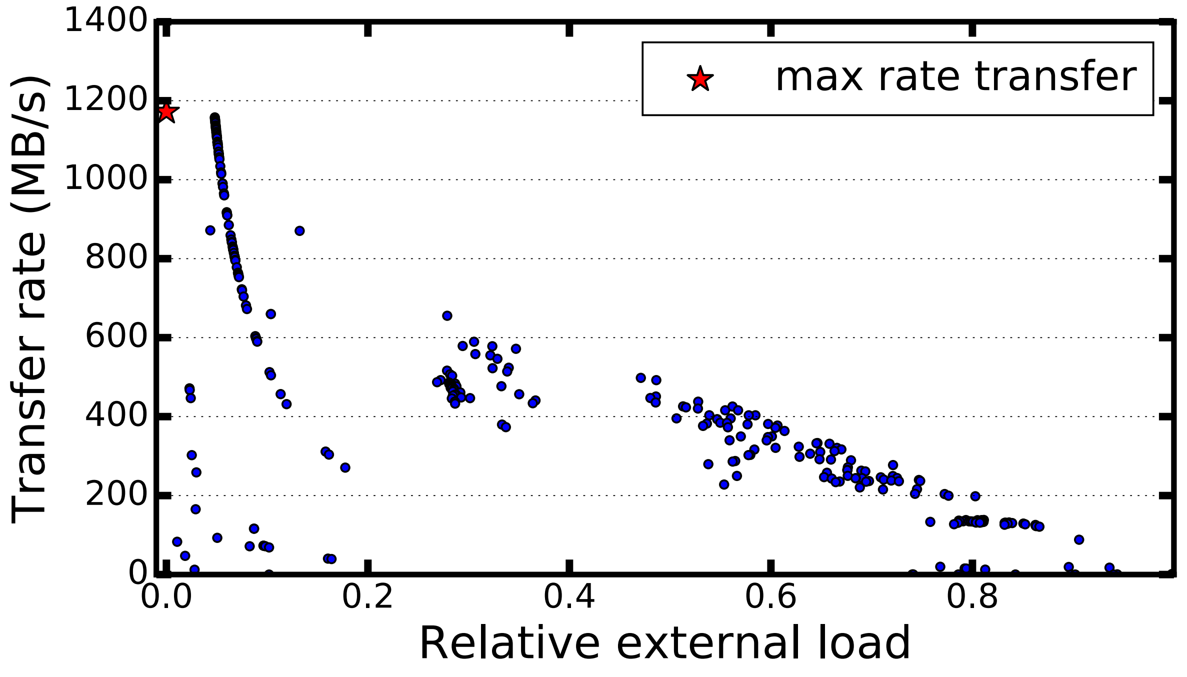
$$ReL = max\left(\frac{K^{sout}}{R_k + K^{sout}}, \frac{K^{din}}{R_k + K^{din}}\right)$$

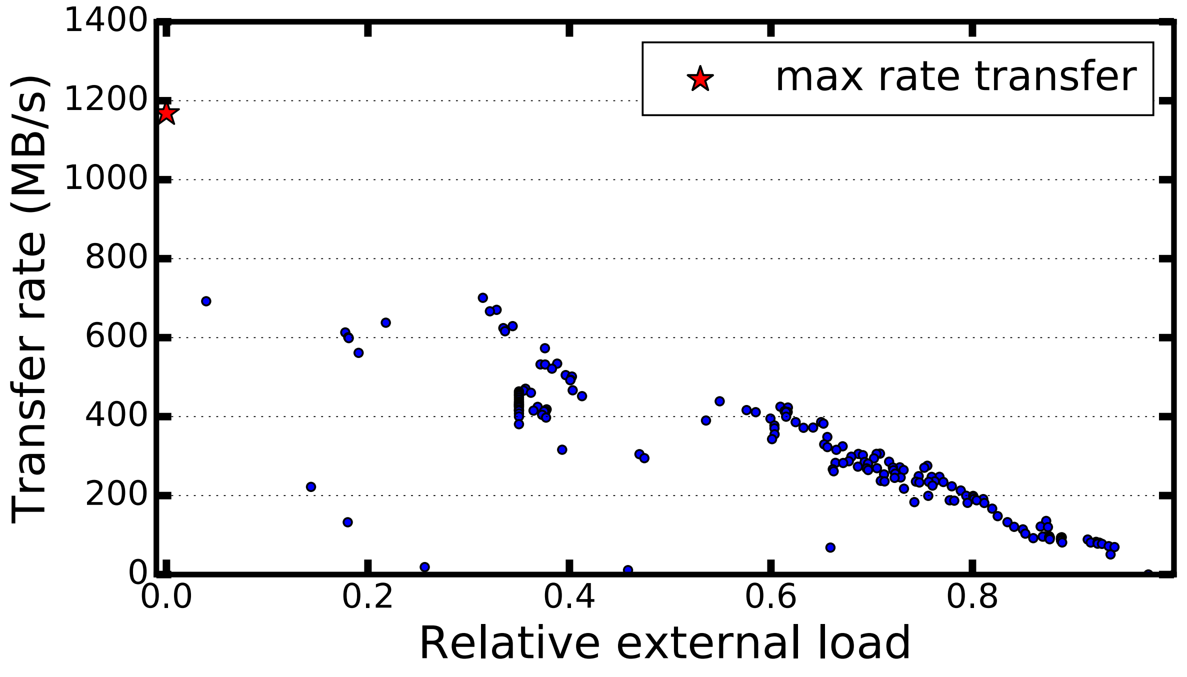
Transfers over ESnet testbed

(less likely to have non-globus load on endpoints)



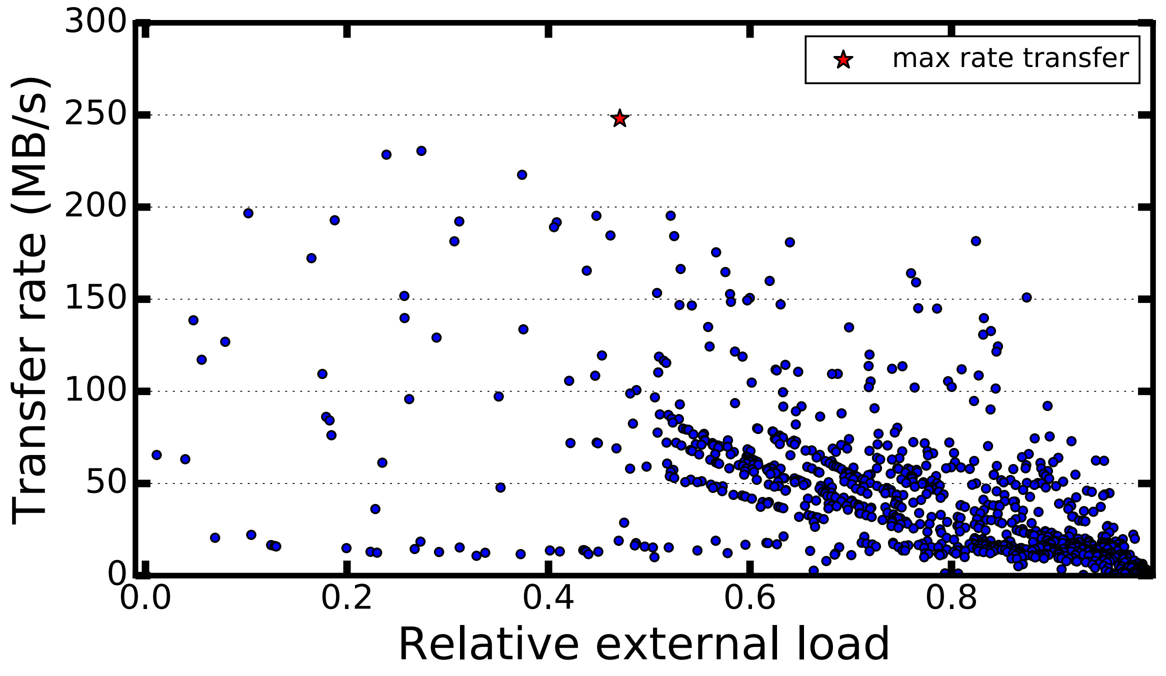


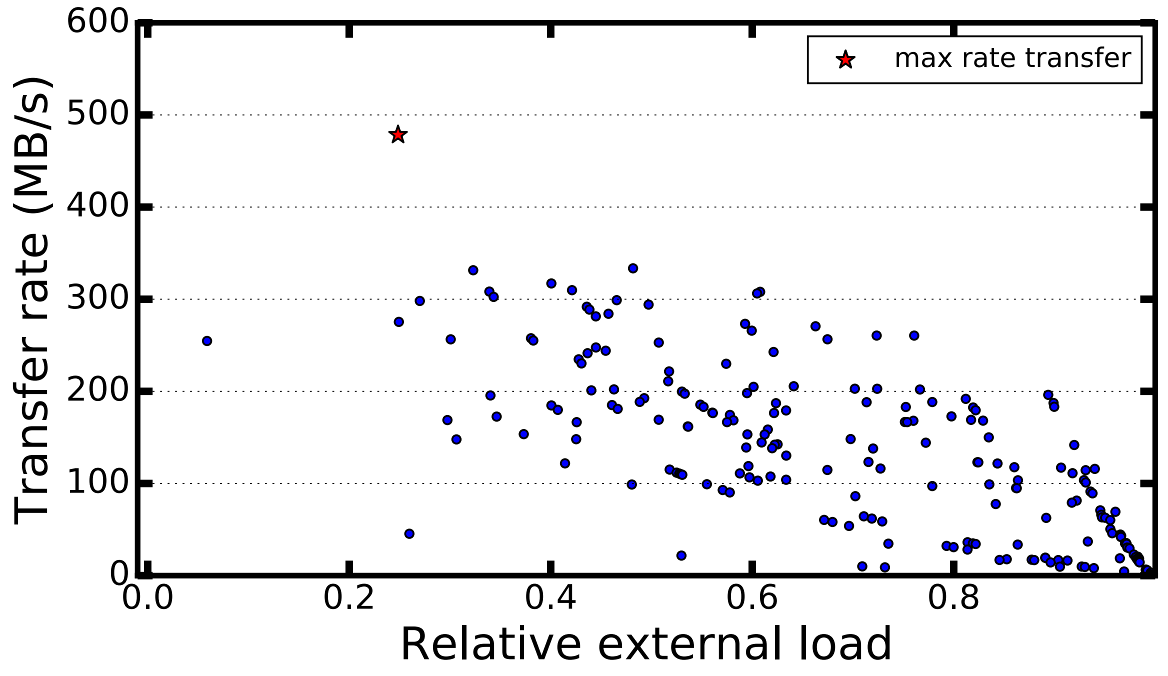


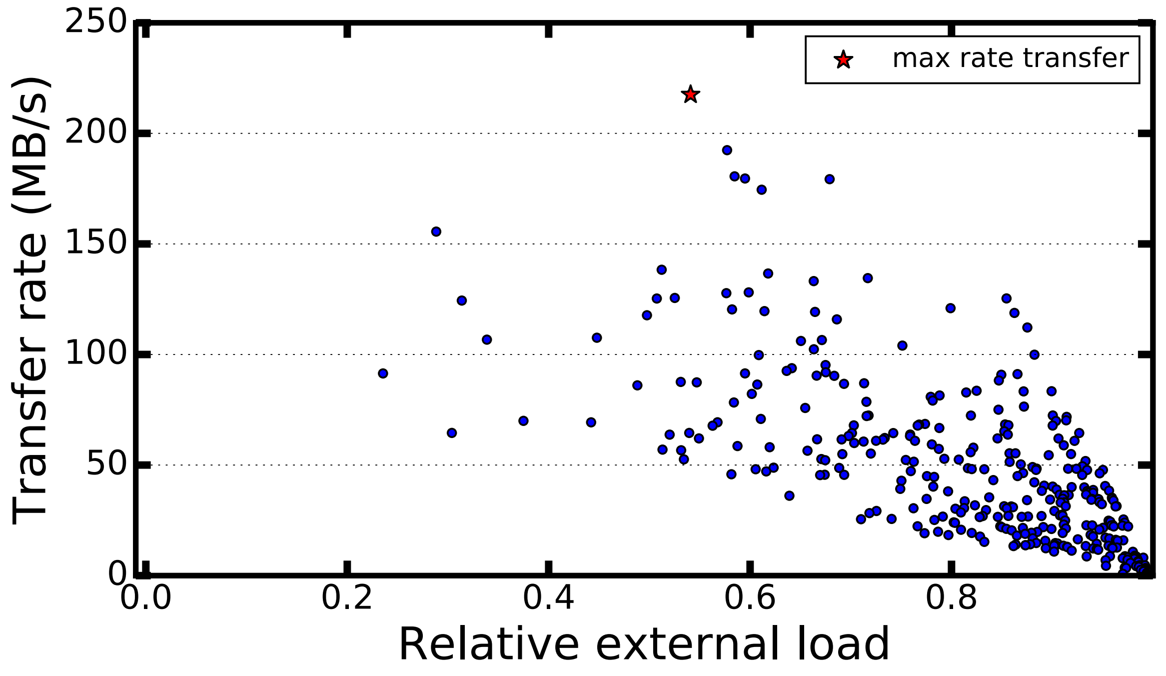


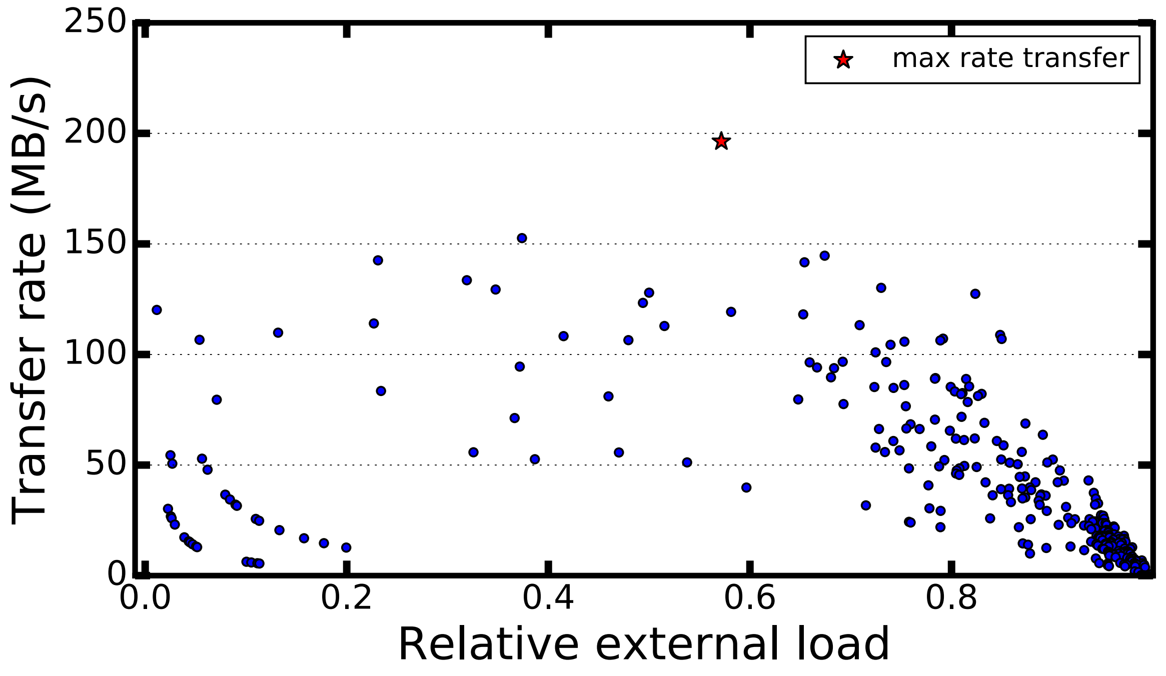
Transfer over production DTN

(more likely to have non-globus load on endpoints)









What affect transfer performance? -4

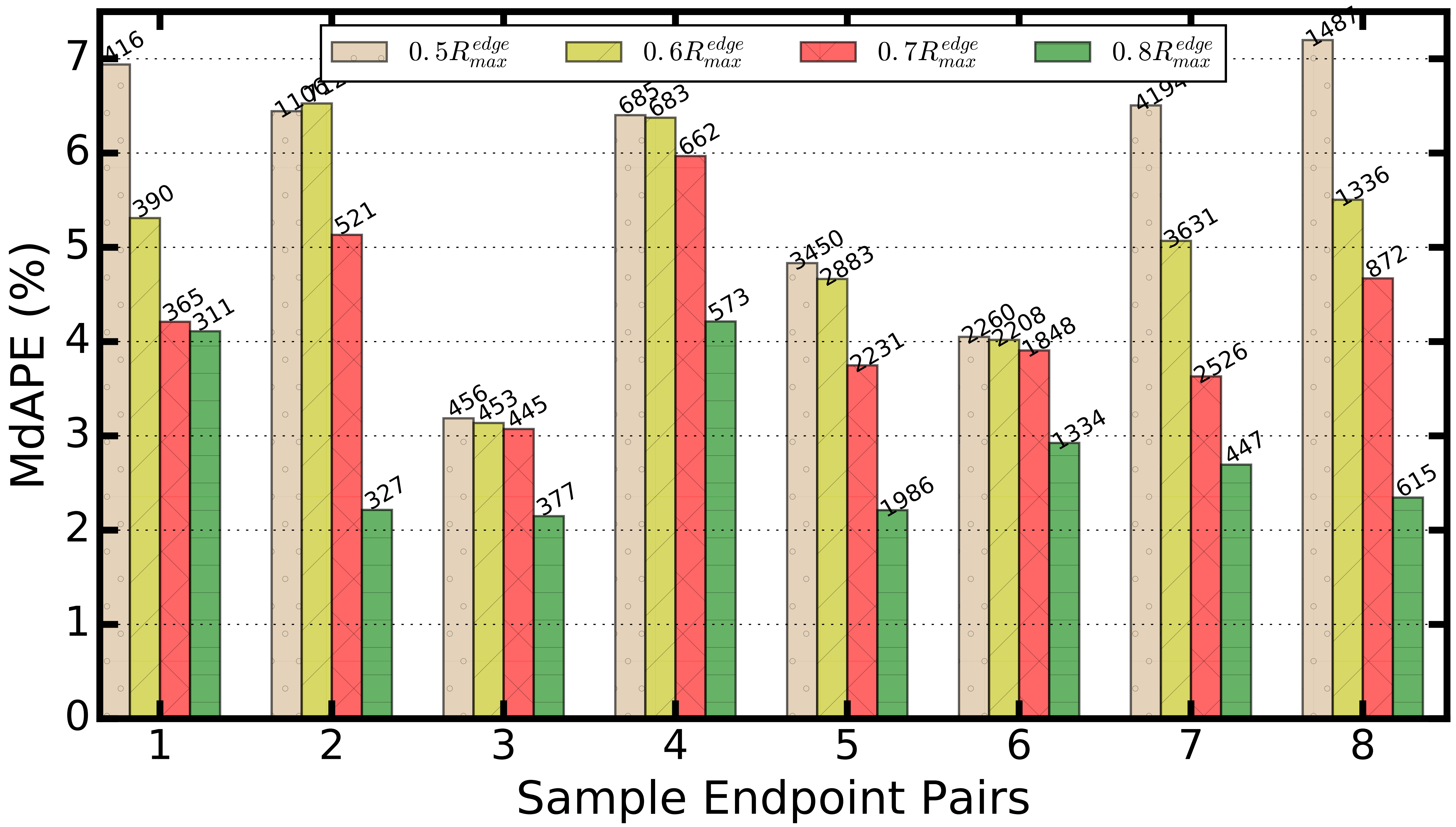
Influence of unknown load:

Select transfers with:

$$\frac{R_k + K^{sout}(k)}{ROmax} \geq \eta \quad and \quad \frac{R_k + K^{din}(k)}{RImax} \geq \eta$$

$$\eta \in \{0.5, 0.6, 0.7, 0.8\}$$

large η means less likely to have unknown load because the max is fixed.



unknown load affects features' interpretability coefficient of determination (R^2).

It is a useful way to filter out noisy logs, extract information from noisy data

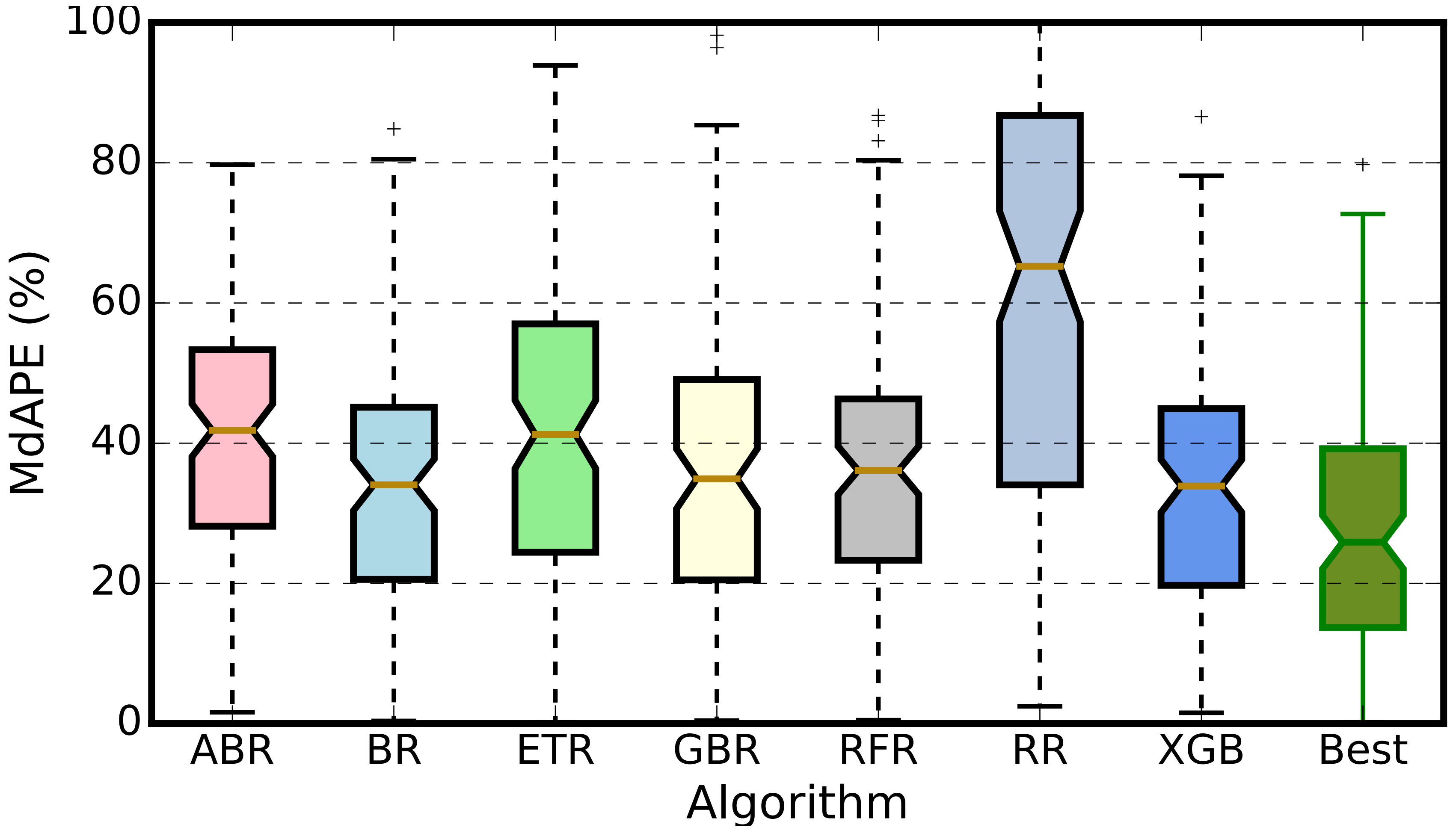
It is time to

Build a Wide-Area File Transfer Performance Predictor

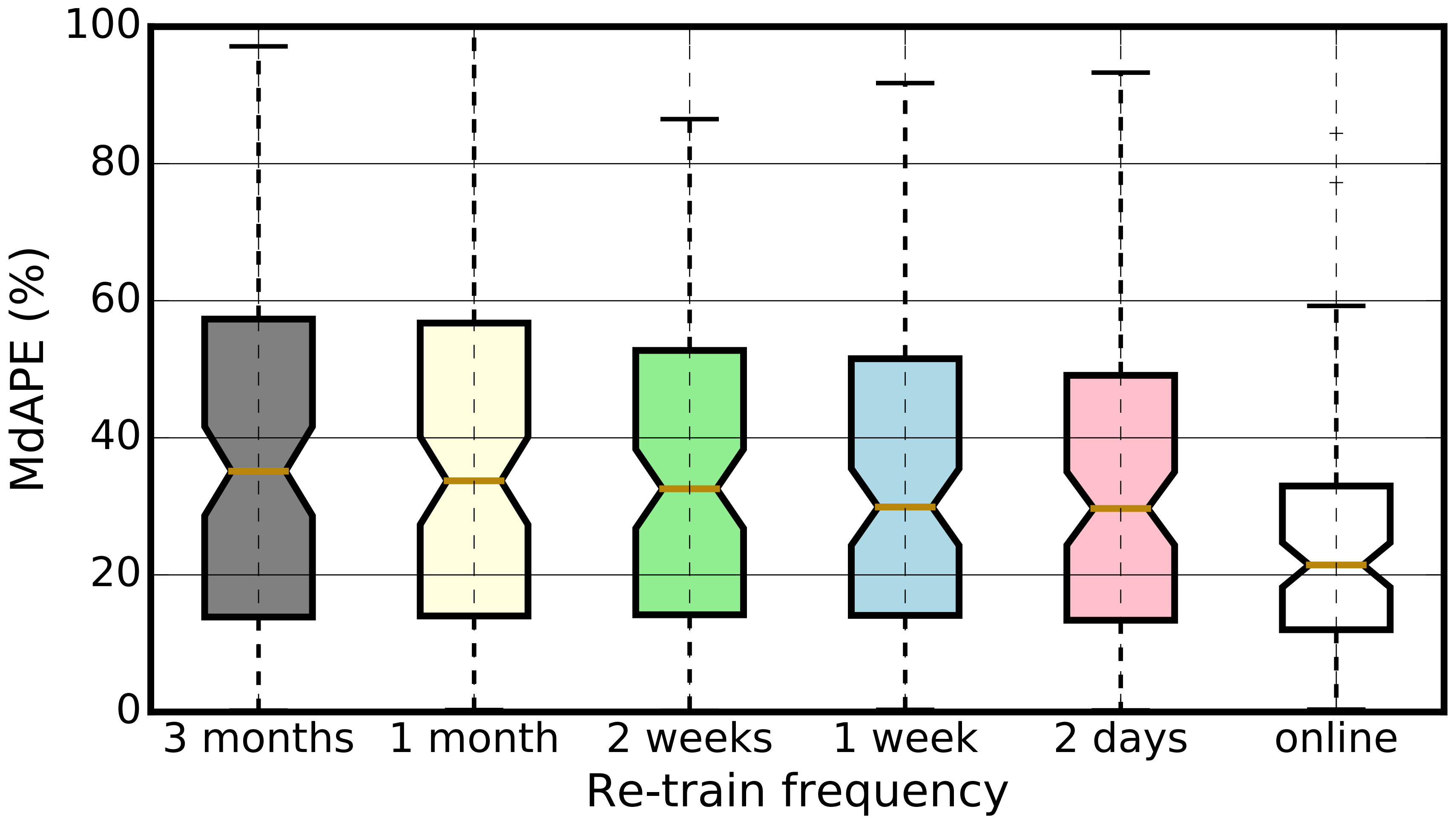
Liu et al. MLN'18

Data transfer: *Prediction*

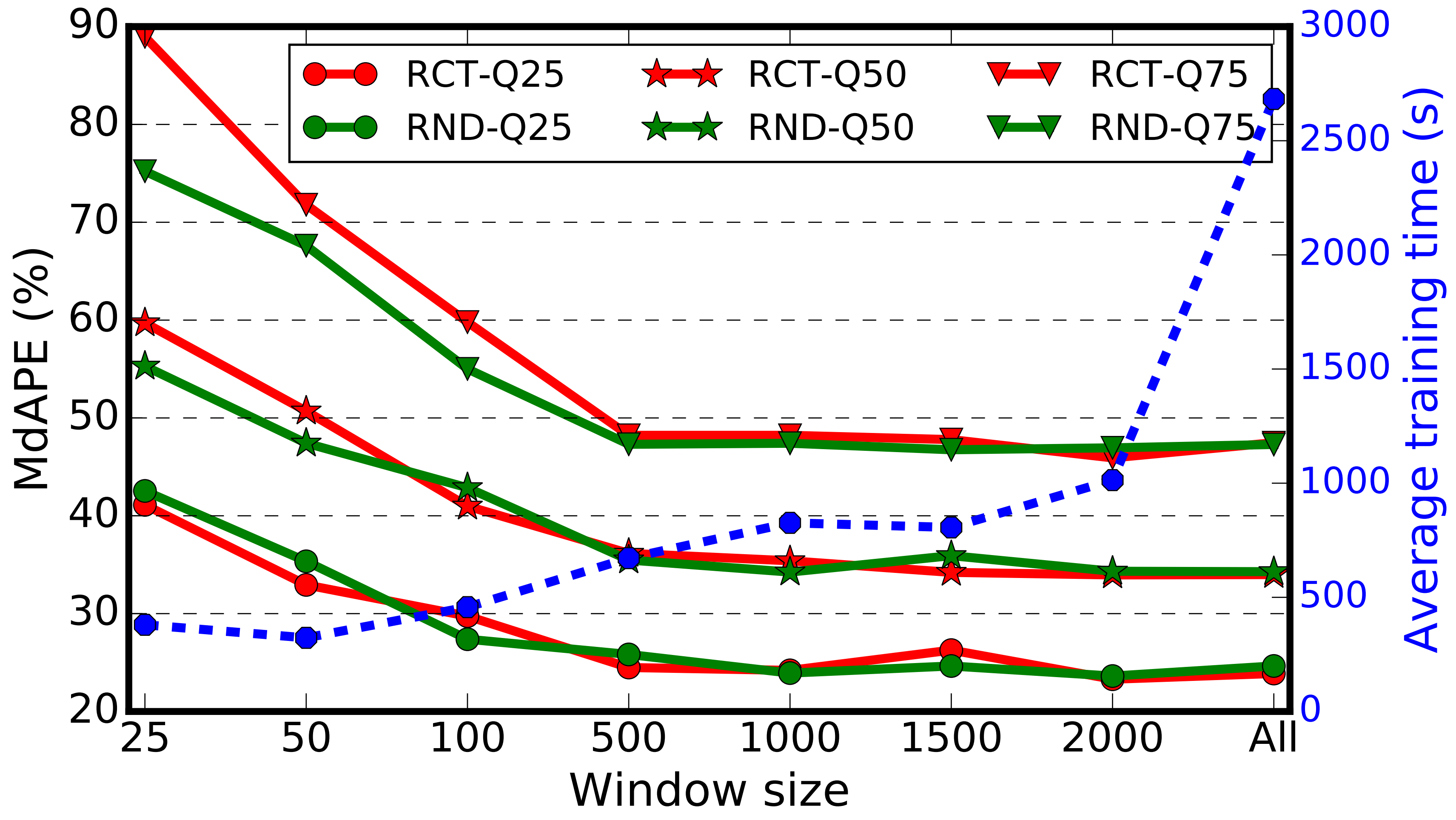
Machine learning based **predictor**



algorithm needs to be edge specific

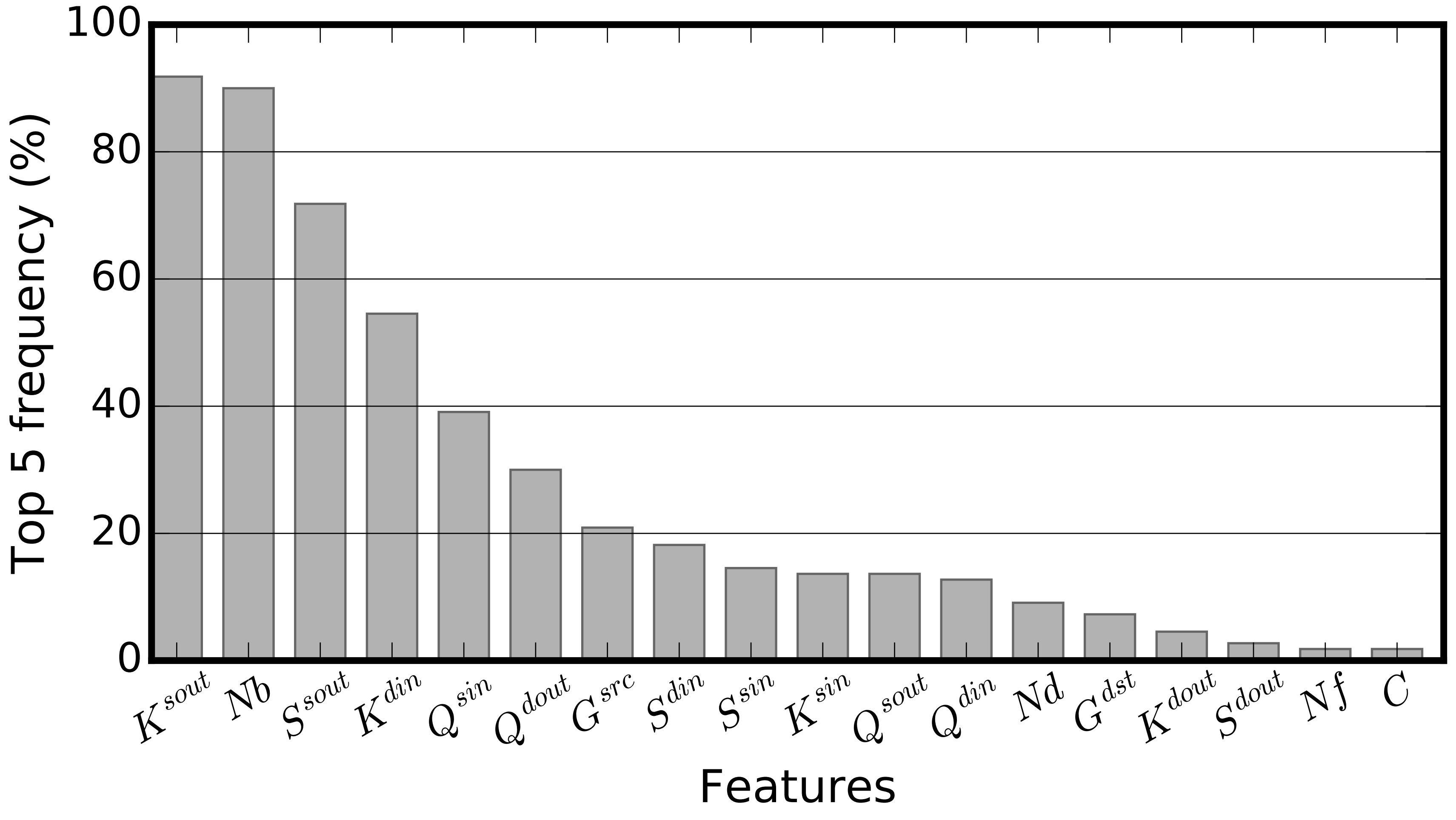


retraining helps



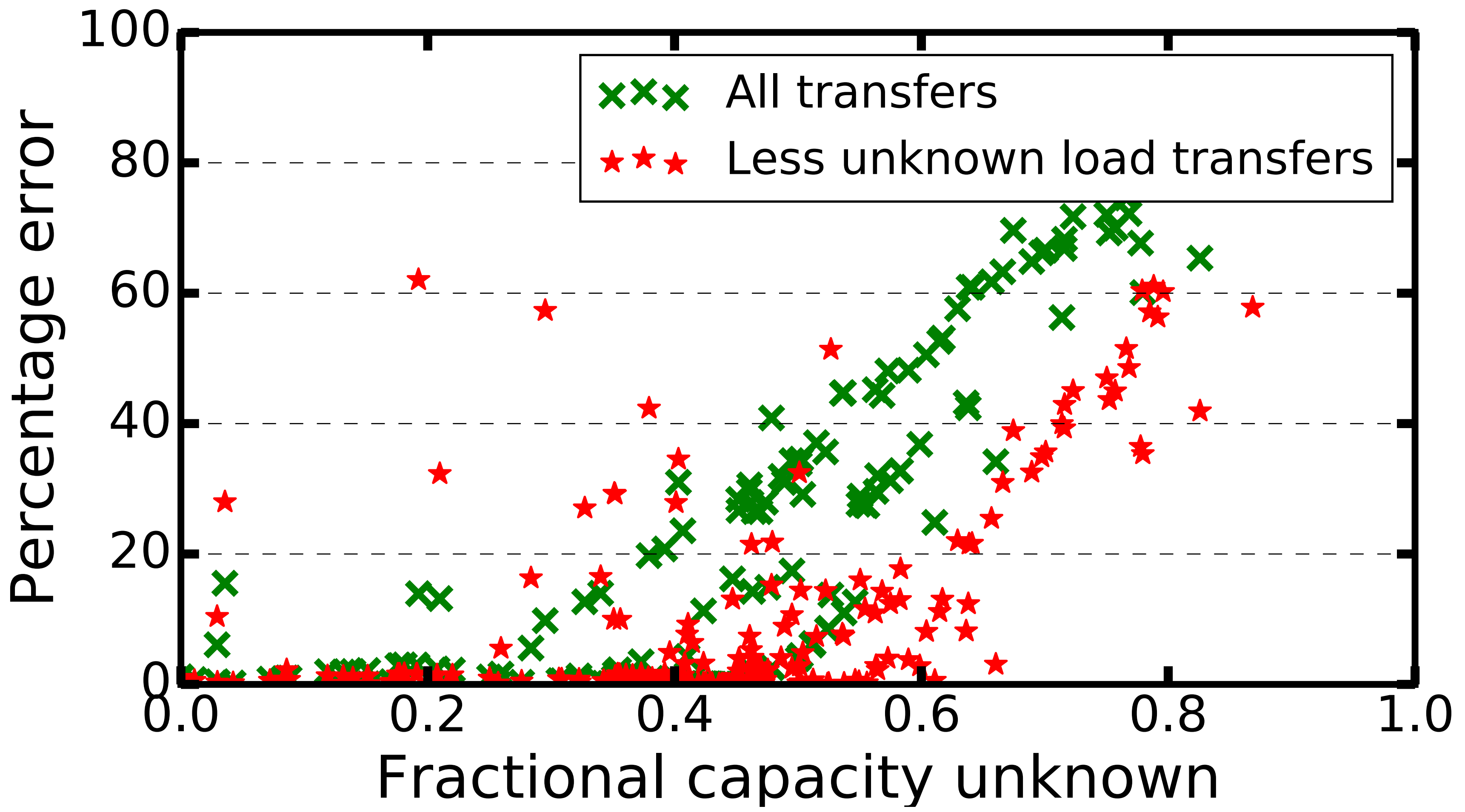
forget old info does not hurt.

Interpret the predictor



Algorithm	Group	Q50(%)	Q75(%)	Q90(%)
Ridge Regression	1	11.24	18.19	22.63
	2	20.04	33.08	64.33
	3	35.37	126.54	223.29
XGBRegressor	1	11.85	22.91	25.20
	2	8.20	18.06	29.36
	3	27.16	51.02	72.49
BaggingRegressor	1	9.54	18.83	25.02
	2	9.46	14.81	32.64
	3	29.85	51.27	133.48

1. fast (top 50%)
2. rest of 1 but heavily (known) loaded.
3. rest of 2, i.e., lots of unknown



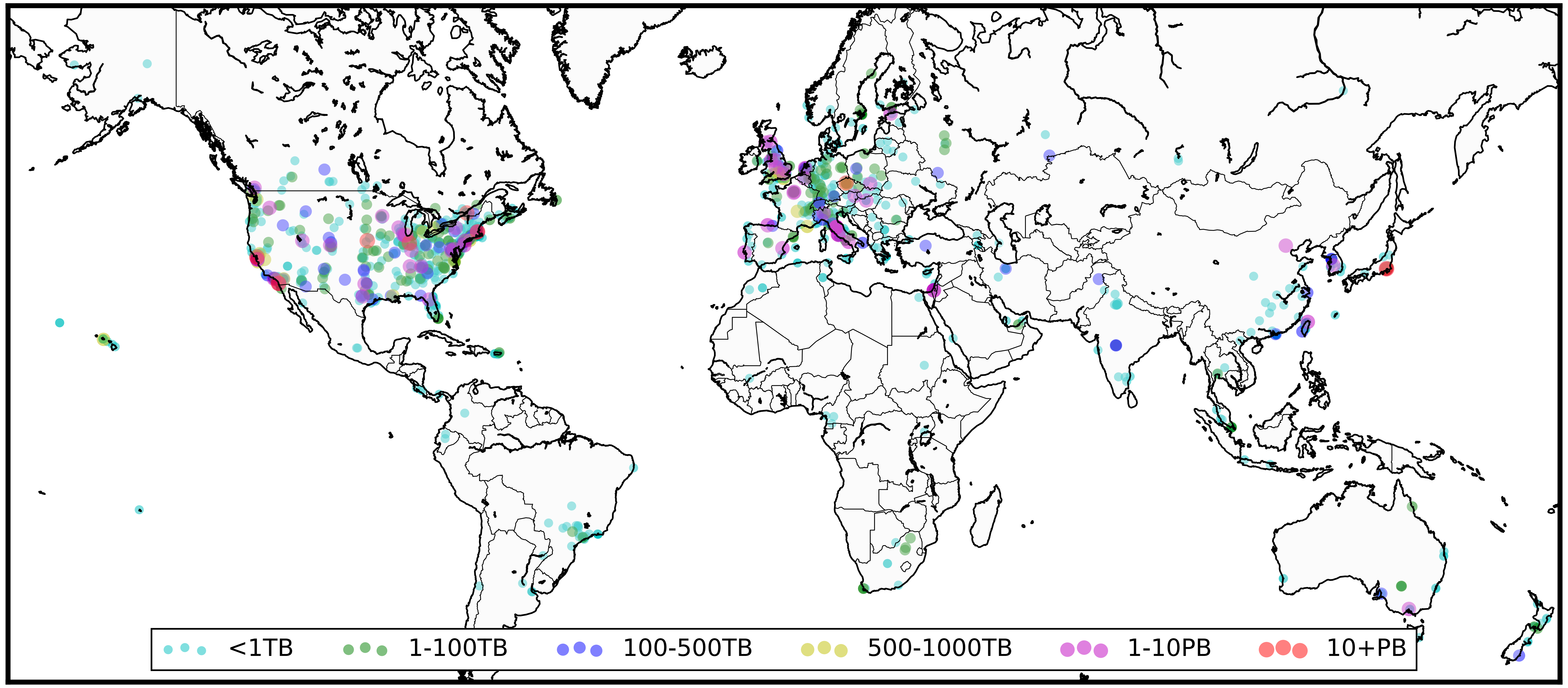
Get a deep understanding of end-to-end file transfer trends and user behavior.

(How does it look like in reality)

Wide Area File Transfer

By using Globus GridFTP, about 20 billion files, totaling 1.8 Exabyte between any two of 63,166 unique endpoints were transferred from 2014 to 2017. On average more than 25,000 files are transferred per minute in 2017.

There are 20.5 billion ***STOR*** logs totaling 1.5 EiB received and 19.4 billion ***RETR*** logs totaling 1.8 EiB transferred.
(not equal? user can disable data collection feature, no perfect data)



Geographical distribution of bytes moved in, per city in 2017

Dataset characteristic
dataset size, # files, average file size, directories, file type and dataset sharing behavior

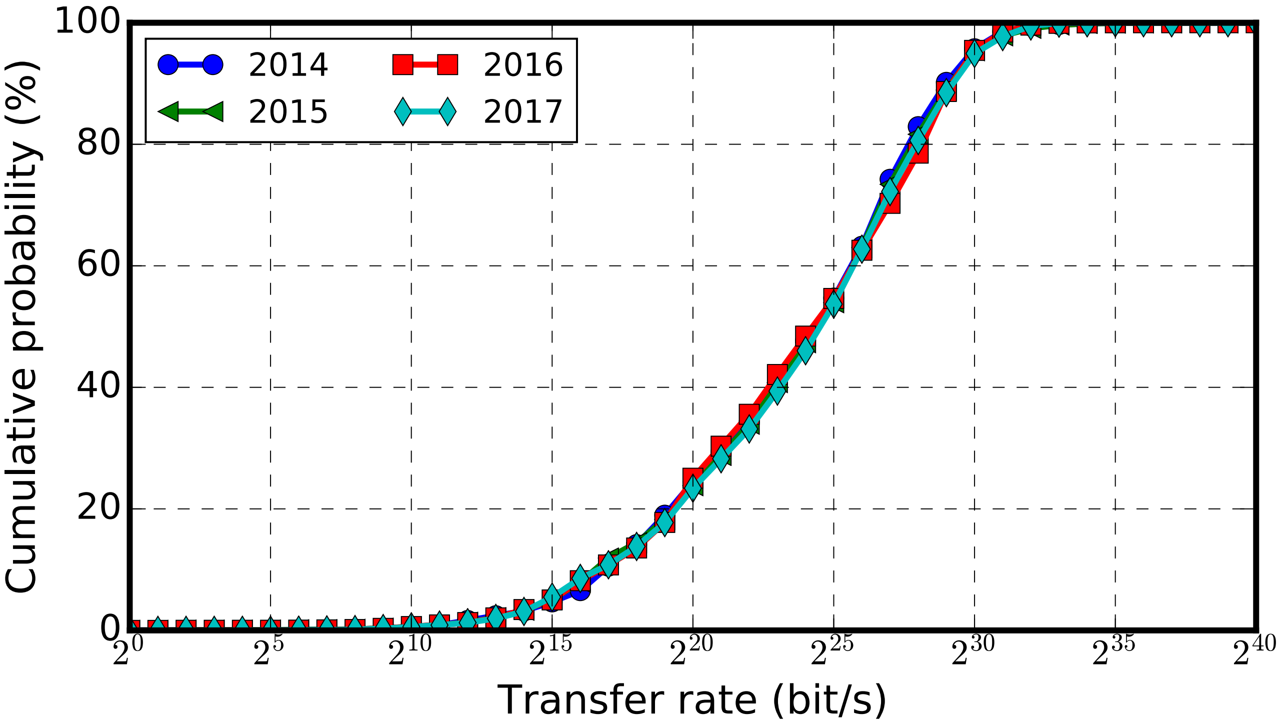
Transfer characteristic
Data integrity checking, encryption, and reliability, transfer direction, performance, duration and transfer parameters

User behavior
transfer frequency, transfer volume, degree of connection to endpoints and pattern of users access endpoint

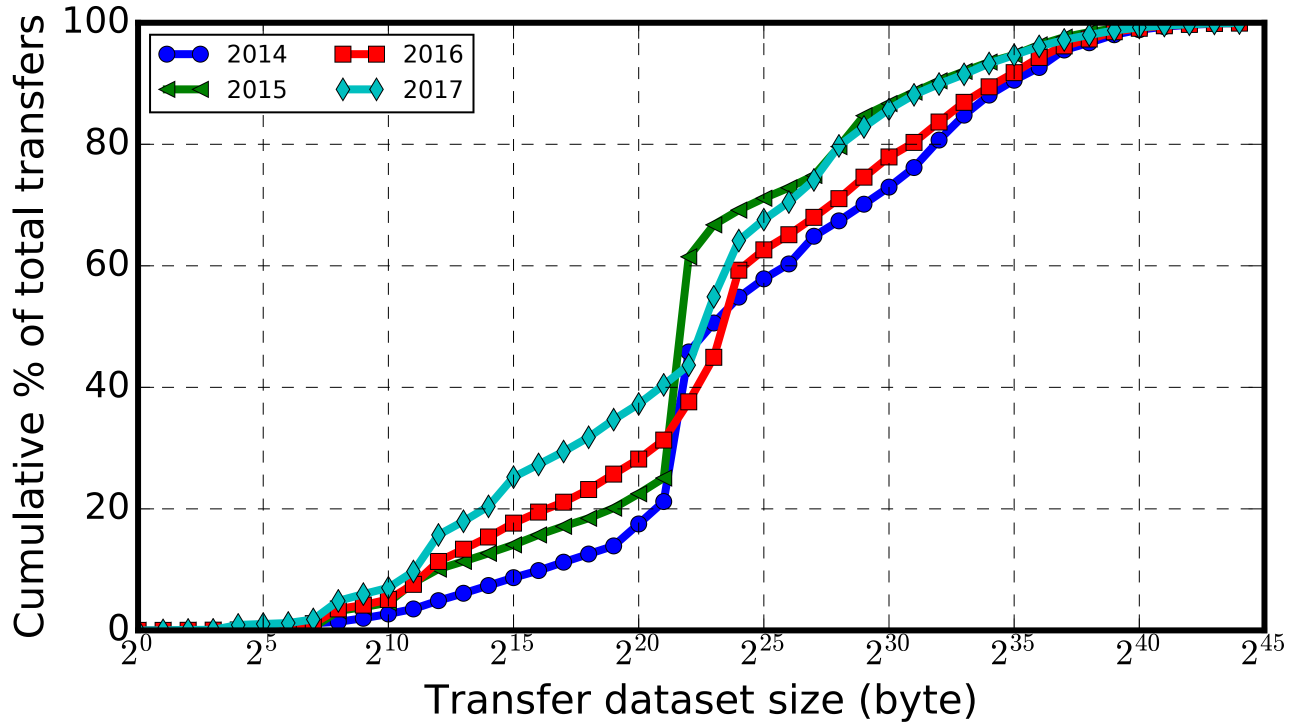
Endpoint
degree of sharing to users, resource utilization (idle time percentage), source-to-destination edge

Data transfer: *Characterization*

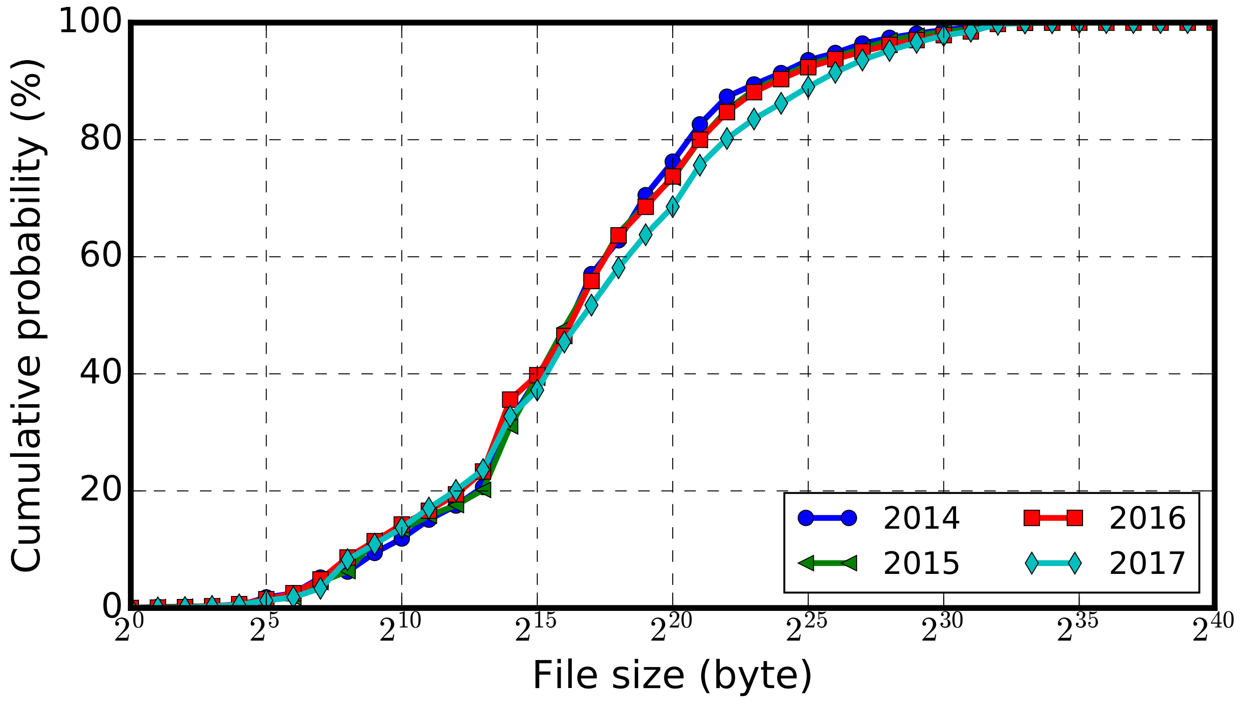
some of the observations from **40 Billion** file transfer records, totaling **3.4 Exabytes** of data transferred.



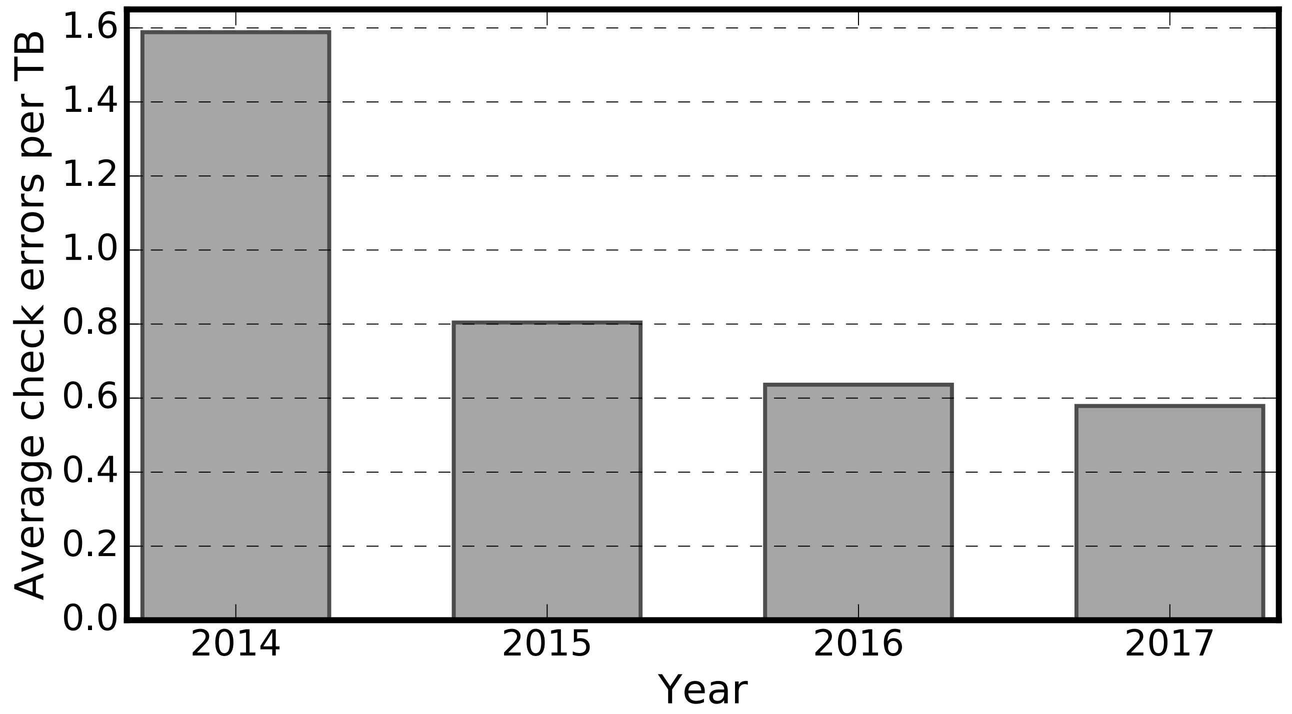
Transfer is slow, median < 50 Mbps



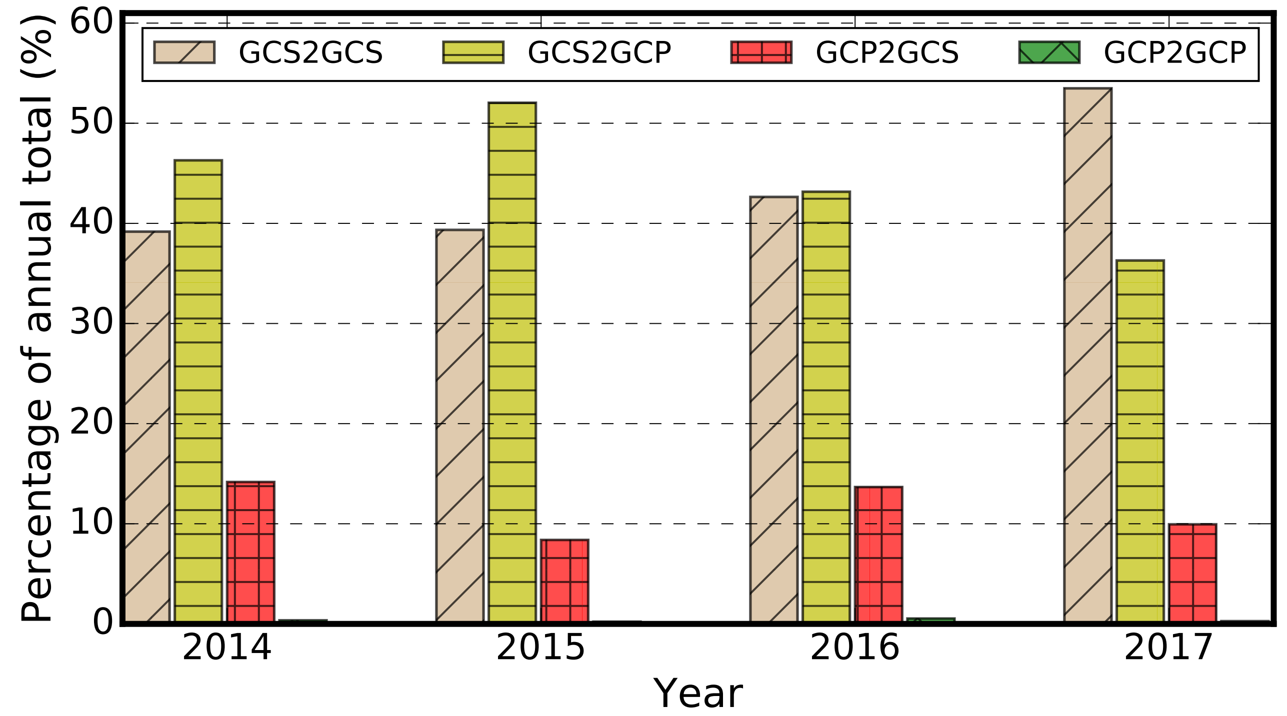
Datasets are small (median is a few MB), has decreased by year.

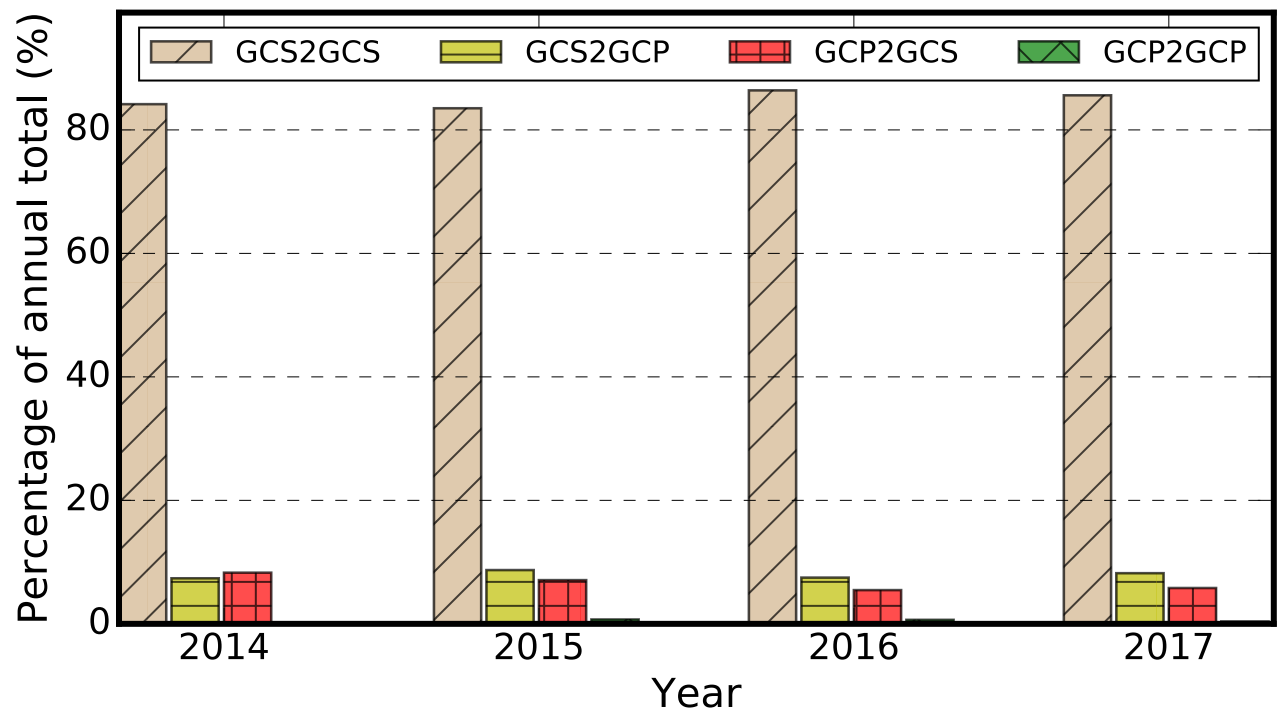


File size is small (the majority is less than 1MB)

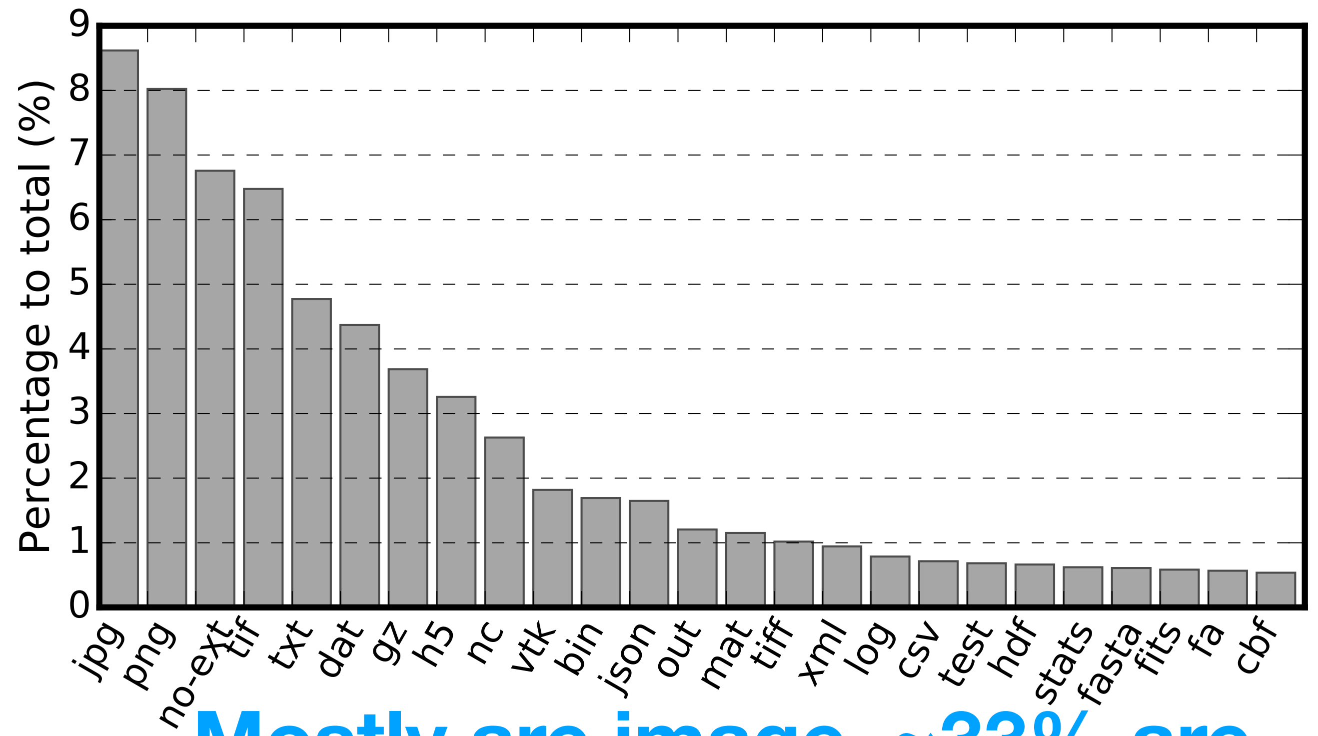


At least one checksum failure occurs per 1.26 TB

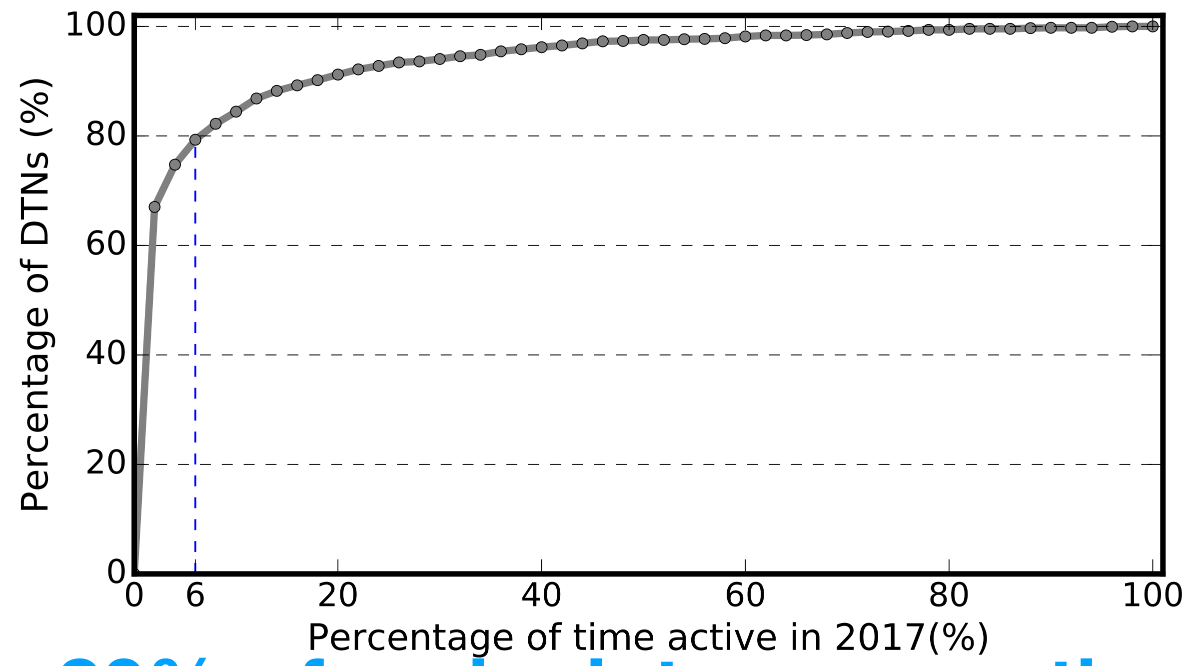




Transfers involve many more downloads (server, e.g., computing facility to personnel) than uploads (personnel to server).



Mostly are image, ~33% are potentially compressible



80% of endpoints were active less than 6% of the time. Some are really busy though (more than 98% active)

Wide Area File Transfer

Liu et al. HPDC'18

Motivational / Counter-Intuitive / Interesting observations

Observation 1. *Most of the datasets moved over the wide area are small. Specifically, the 50th, 75th, and 95th quartiles of dataset size are 6.3 MB, 221.5 MB, and 55.8 GB, respectively. Counterintuitively, the dataset size has decreased year by year from 2014 to 2017.*

No increasing dataset size

Observation 2. *Most of the datasets transferred by the Globus transfer service have only one file. And 17.6% of those datasets (or 11% of the total) have a file size of ≥ 100 MB, motivating the need for striping the single-file transfer over multiple servers.*

lots of single file transfers motivates striping

Observation 3. *The average file size of most datasets transferred is small (on the order of few megabytes). Majority of individual file size is less than 1 MB. These results motivate the need for performance optimizations aimed at small file transfers.*

Files are really small thus challenging

Observation 4. *Image files are the most common file type transferred, followed by raw text files. Scientific formats such as* .h5 *(hierarchical data format) and* .nc *(NetCDF) are in the top 10.*

Lots of image transferred, ~35% are potentially compressible

Observation 5. *Repeated transfers are not common, less than 7.7% of the datasets are transferred more than once. When they do occur, the datasets in question are distributed mostly from one (or a few) endpoints to multiple destinations (i.e., $N_{usr} < N_{dst}$). We also observe multiple users transferring the same data to the same destination.*

Data sharing is not that common

Observation 6. *At least one checksum failure occurs per 1.26 TB. Although integrity checking adds extra load to storage and CPU on the source and destination endpoints, it is worthwhile. The failures are decreasing year by year. Only 1.9% of transfers used encryption.*

Data corruption is common while protection is expensive

Observation 7. *Transfers involve many more downloads (GCS to GCP) than uploads (GCP to GCS).*

Mostly are downloads

Observation 8. *Although some server-to-server transfers achieve high performance (dozens of Gbps), most transfer throughput is low. For example, the median throughput is only tens of Mbps. There is no clear increasing trend in terms of transfer performance over time.*

transfer rate is not that fast

Observation 9. *Most users do not manually tune the transfer parameters (e.g., 94.6% of the transfers use P = 1). Transfer tools should be smart enough to choose the best parameters for each transfer in order to achieve maximum performance.*

Shouldn't rely on users tuning!!

Observation 10. *Of all the bytes transferred, 80% are by just 3% of all users; 10% of the users transferred 95% of the data.*

Observation 11. *The degree distribution of the number of users per endpoint follows a power-law distribution, similar to other real-world social network graphs.*

User (scientist) behaves similarly as human in social network

Observation 12. *DTN utilization is surprisingly low. Since the DTN requirement is high for high-throughput DTNs, some good topics for research would be the use of these computing resource (1) for other purposes; (2) for complex encoding to deal with data corruption and; (3) to compress data to reduce the network bandwidth consumption.*

Most DTNs are not very busy

Observation motivated optimization — C1

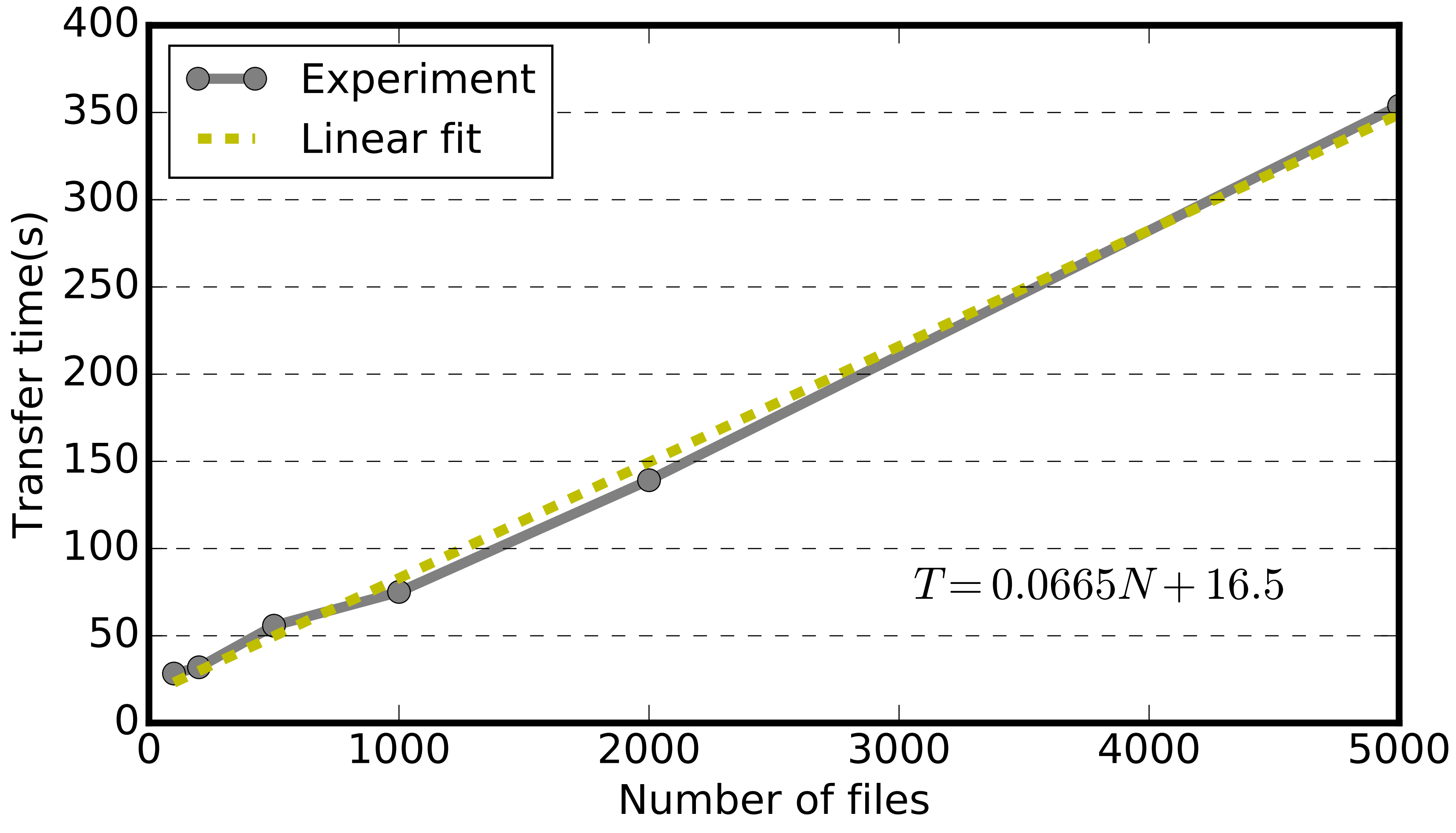
(Real problem motivated)

Target: lots of small files, e.g., median is only a few MiB

Liu et al. CCGriD'19

Insights into transfer performance between scientific facilities

- *Storage read overhead* is introduced by (previous) file close and (next) file open at the source (O_R);
- *Storage write overhead* is introduced by (previous) file close and (next) file open at the destination (O_W);
- *Network overhead* is caused by TCP dynamics due to discontinuity in data flow caused by O_R and/or O_W (O_N);



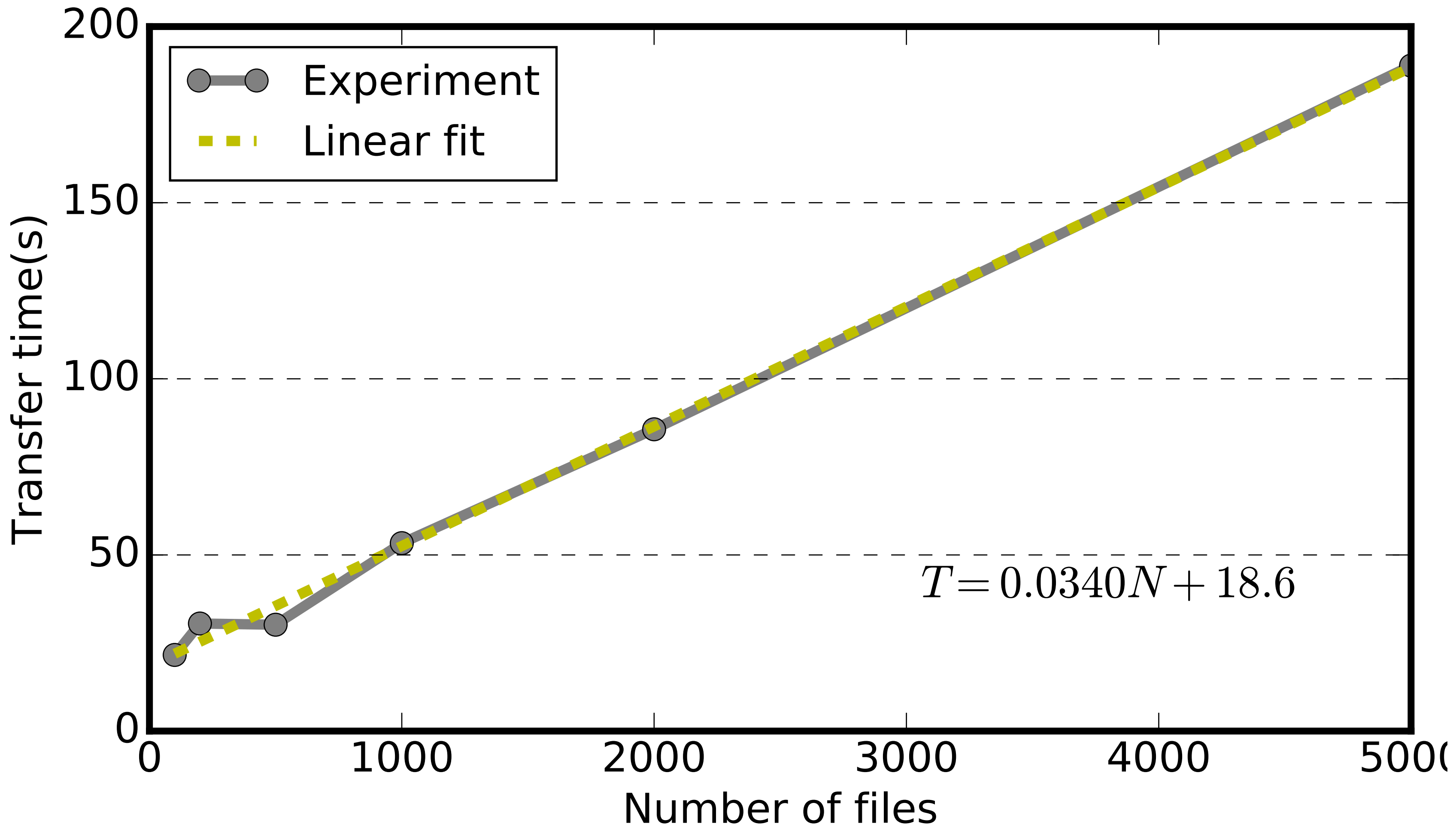
End to end per-file overhead: 66.5 ms

If all subsystems have no buffers at all, the overall per-file overhead would be $O_f = O_R + O_N + O_W$

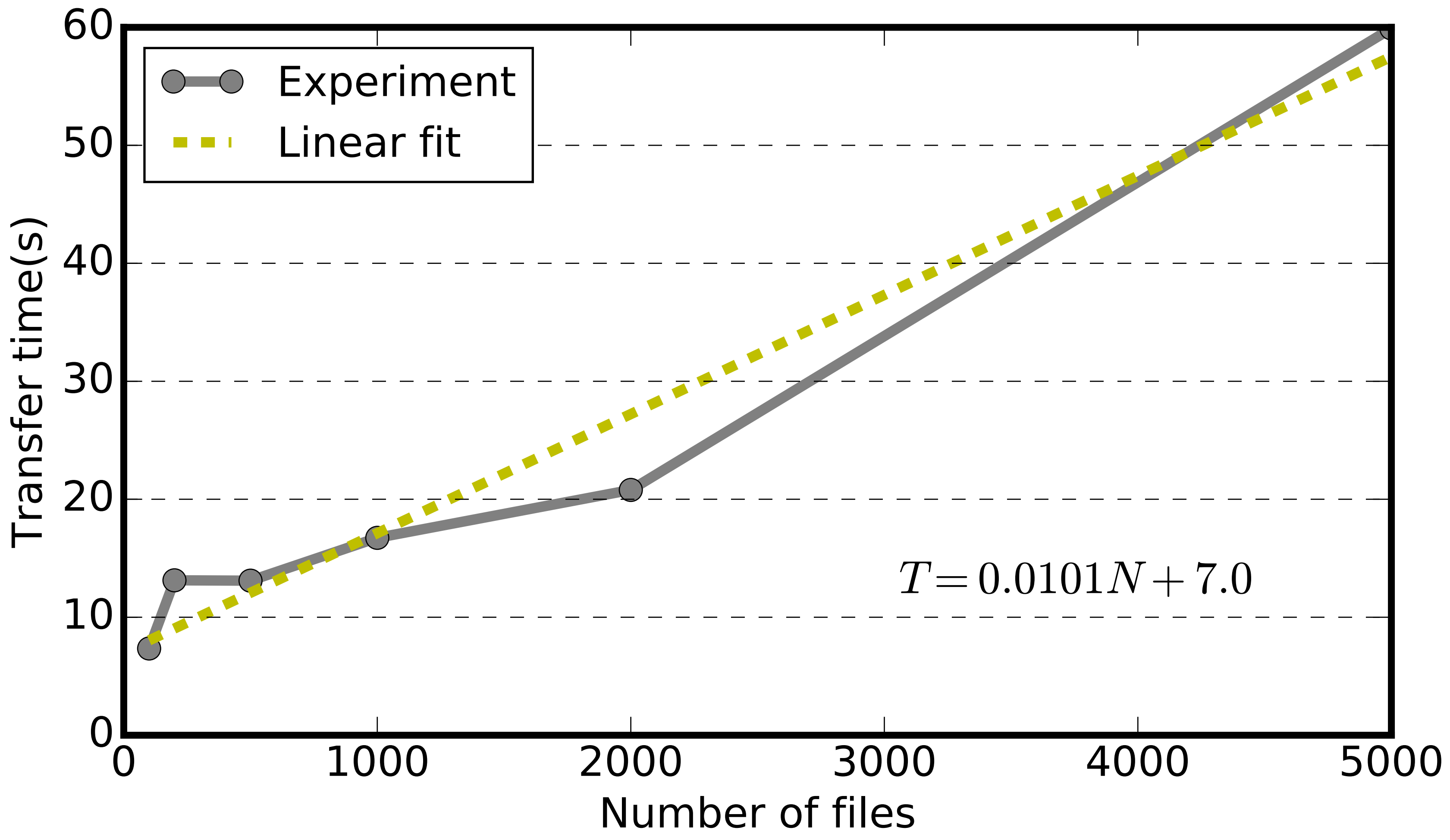
If there are an infinite amount of buffer, the overall per-file overhead would be $O_f = max\left(O_R, O_N, O_W\right)$

With limited buffers, the overall per-file overhead will be between $max\left(O_R, O_N, O_W\right)$ and $O_R + O_N + O_W$

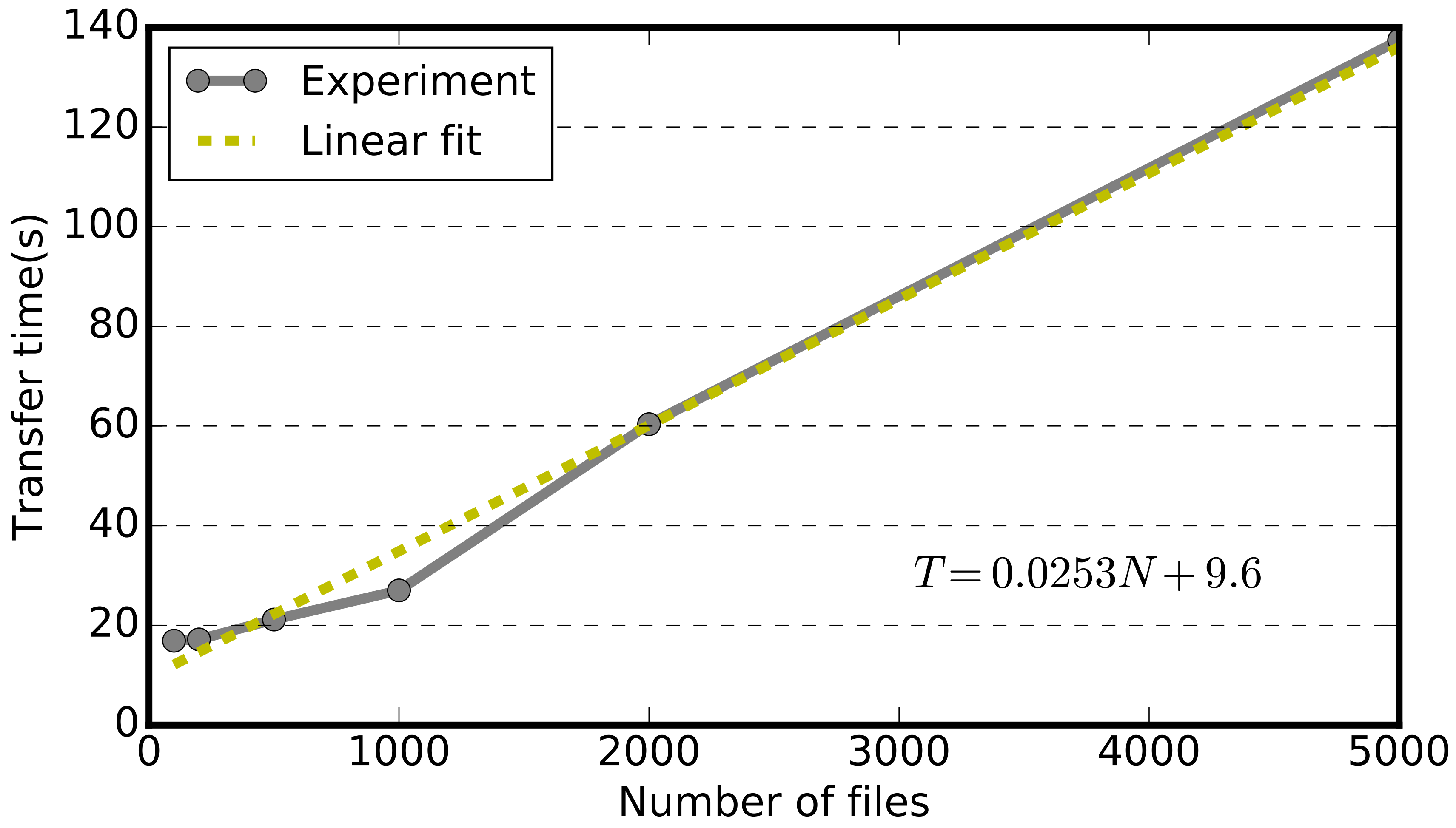
max(43,10,25) < 66.5 < 34+10+25



Read (Files to Memory), 34 ms

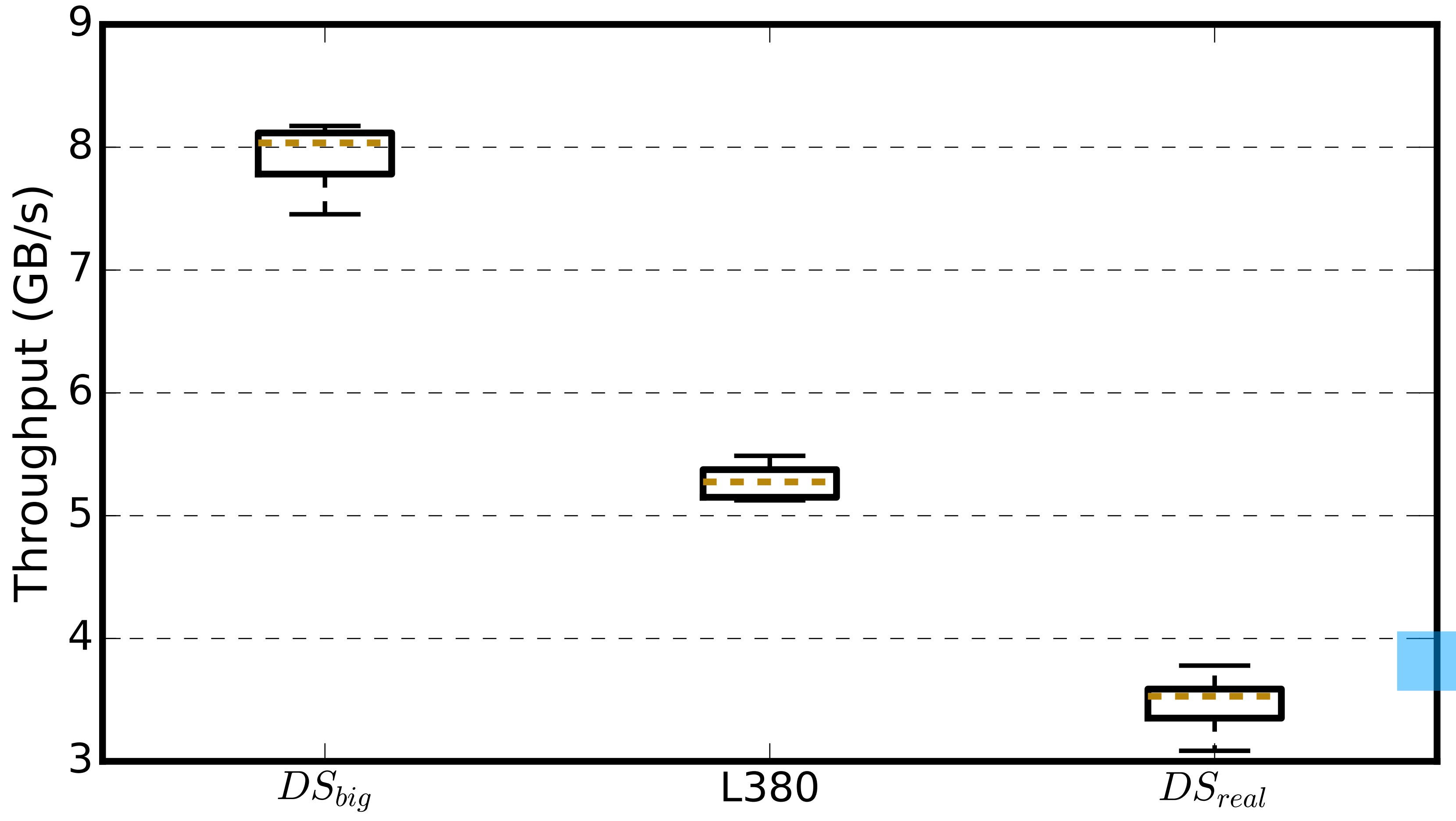


Write (Memory to Files), 10ms

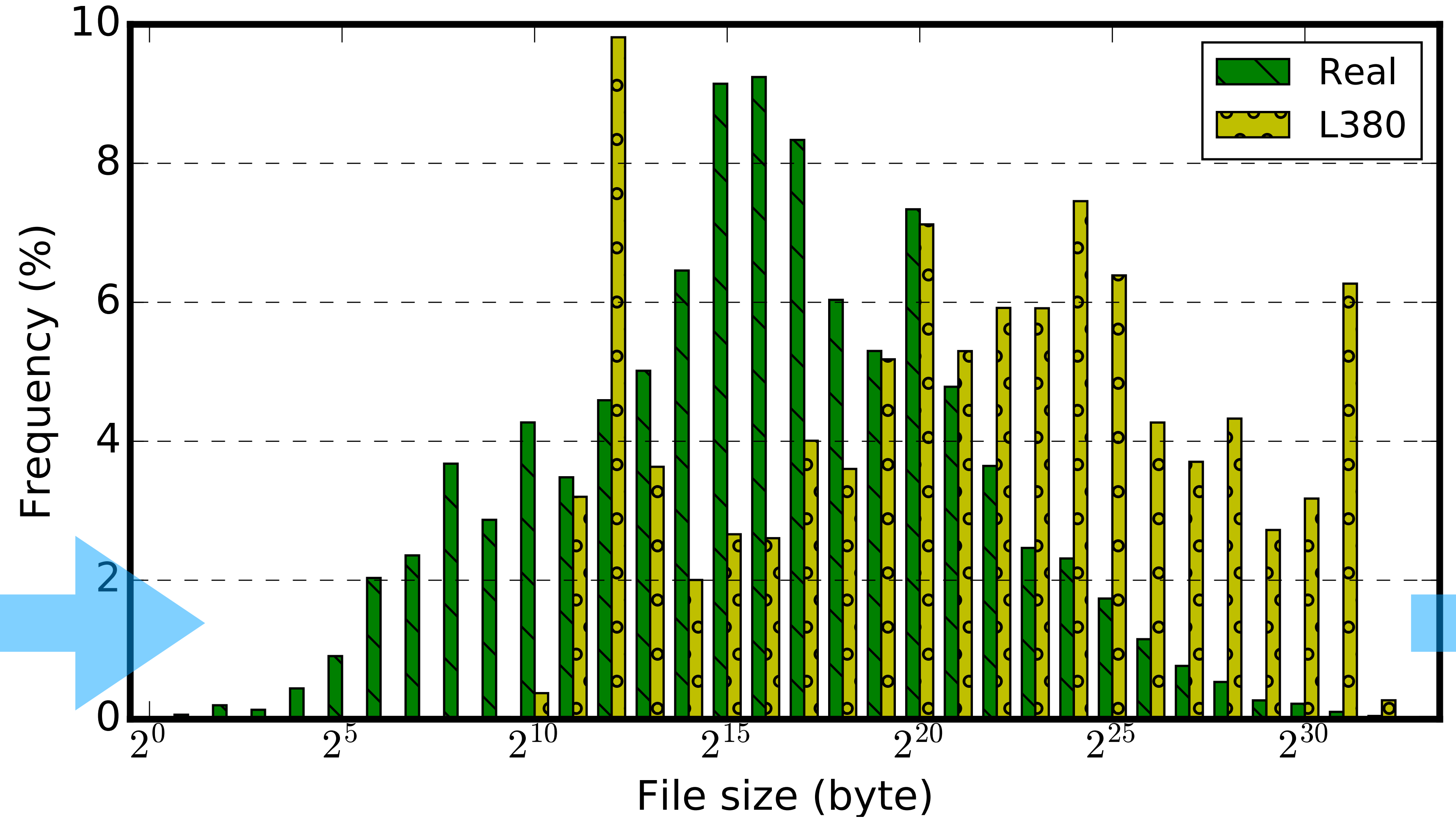


Network (Memory to Memory), 25ms

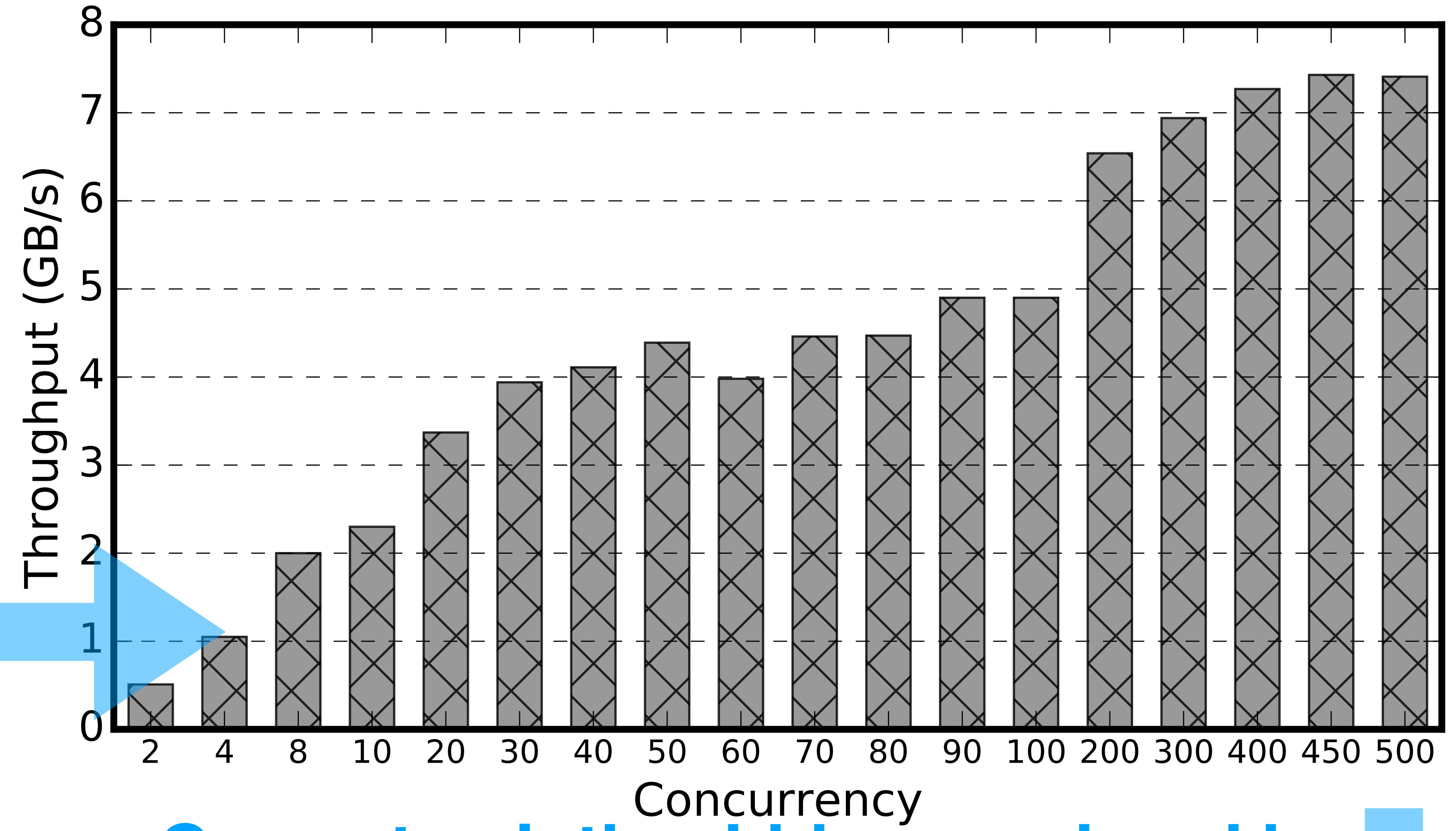
An optimization motivated by the insights got from log analysis



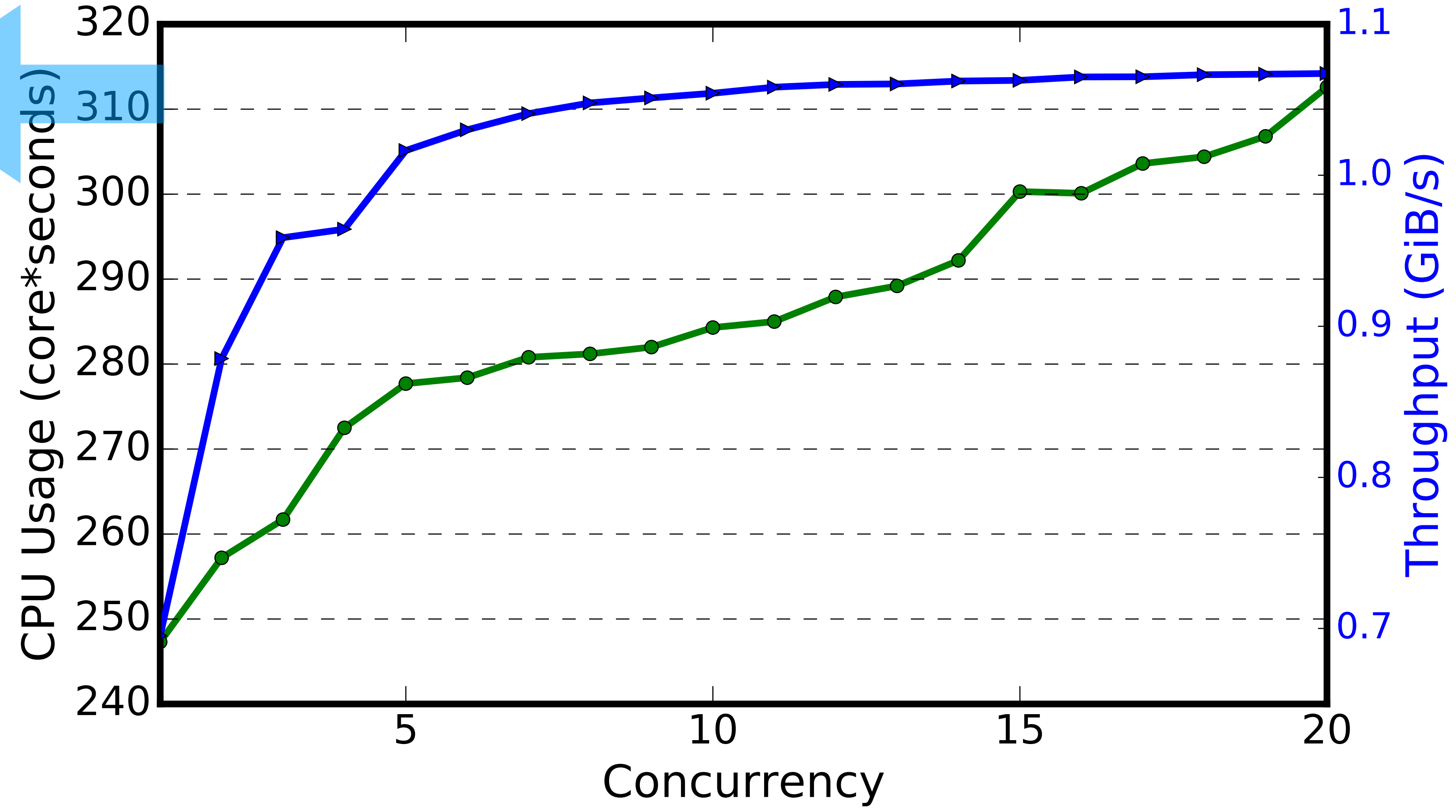
File size influences the performance a lot!



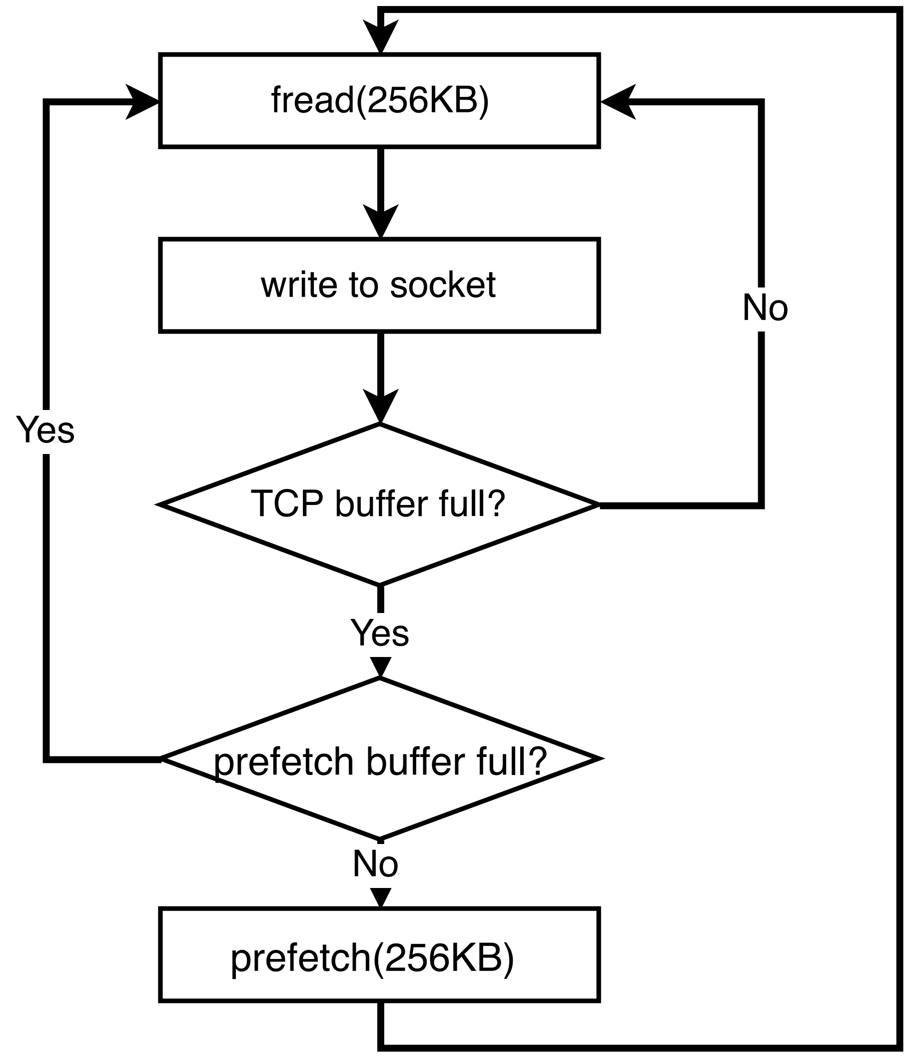
In practice, transferred files are small.



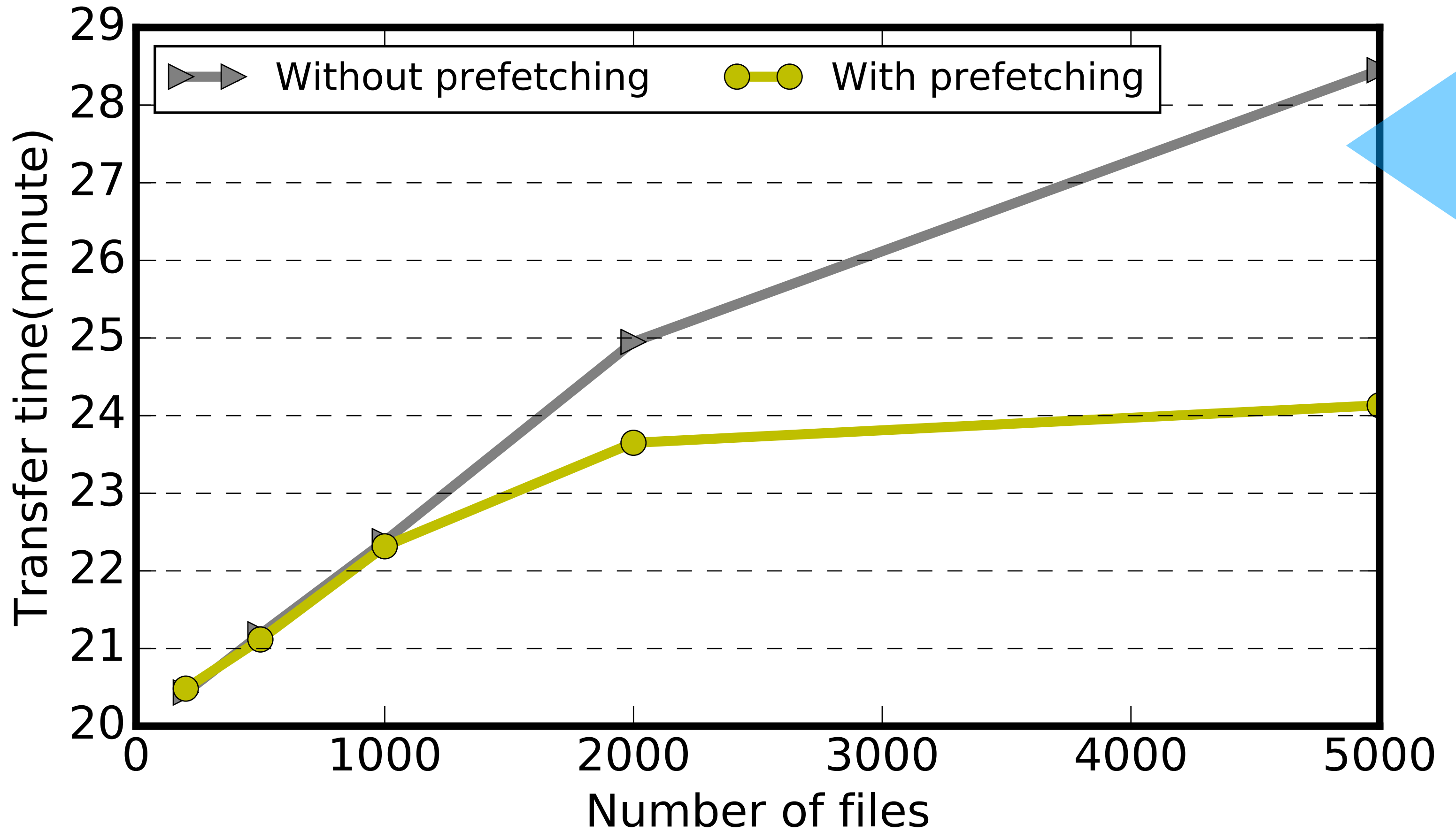
Current solution hides overhead by transferring many files concurrently.



But current solution needs higher end DTNs and consumes more resources.



Pre-fetching

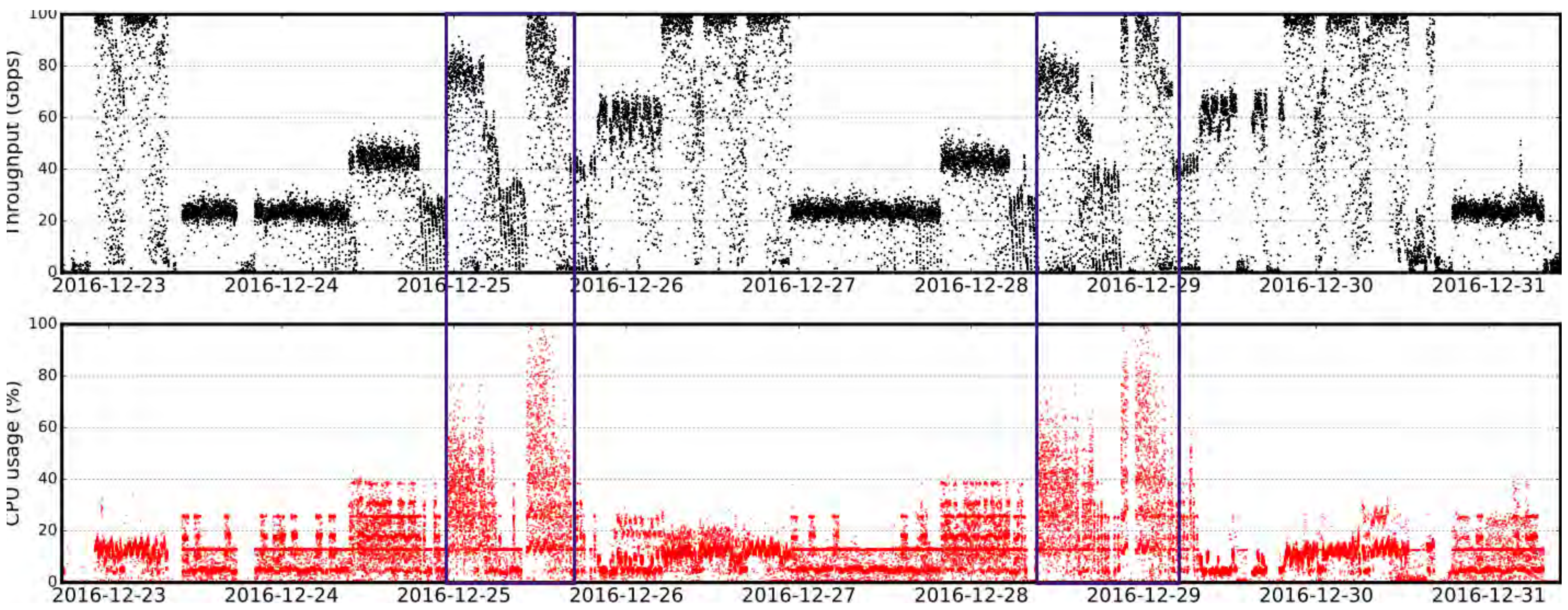


Prefetch on source actively reduces overhead.

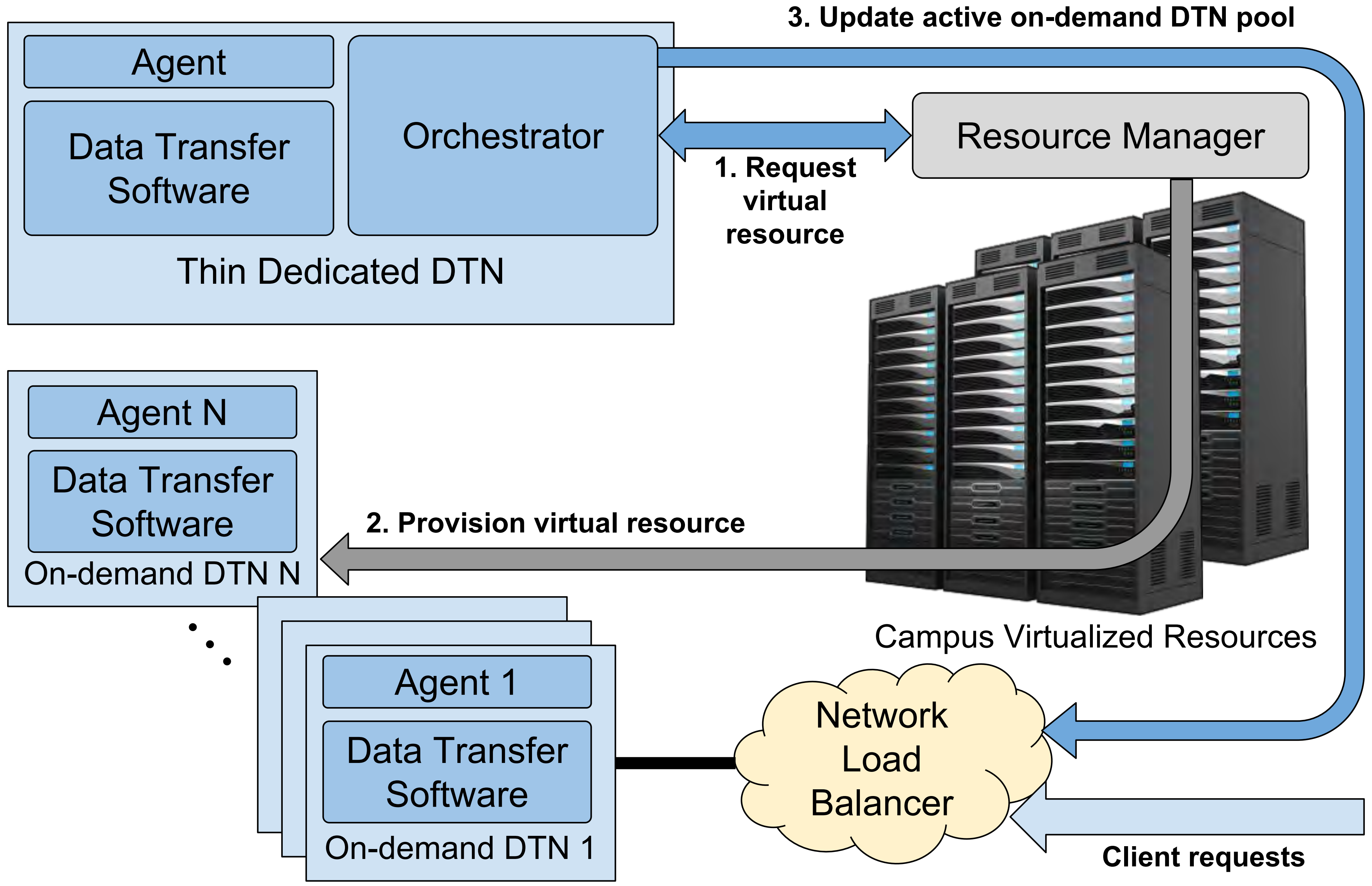
Observation motivated optimization — C2
(Real problem motivated)

Most DTNs (purposely build systems) are not very busy

Embracing Elastic: building an elastic data transfer infrastructure



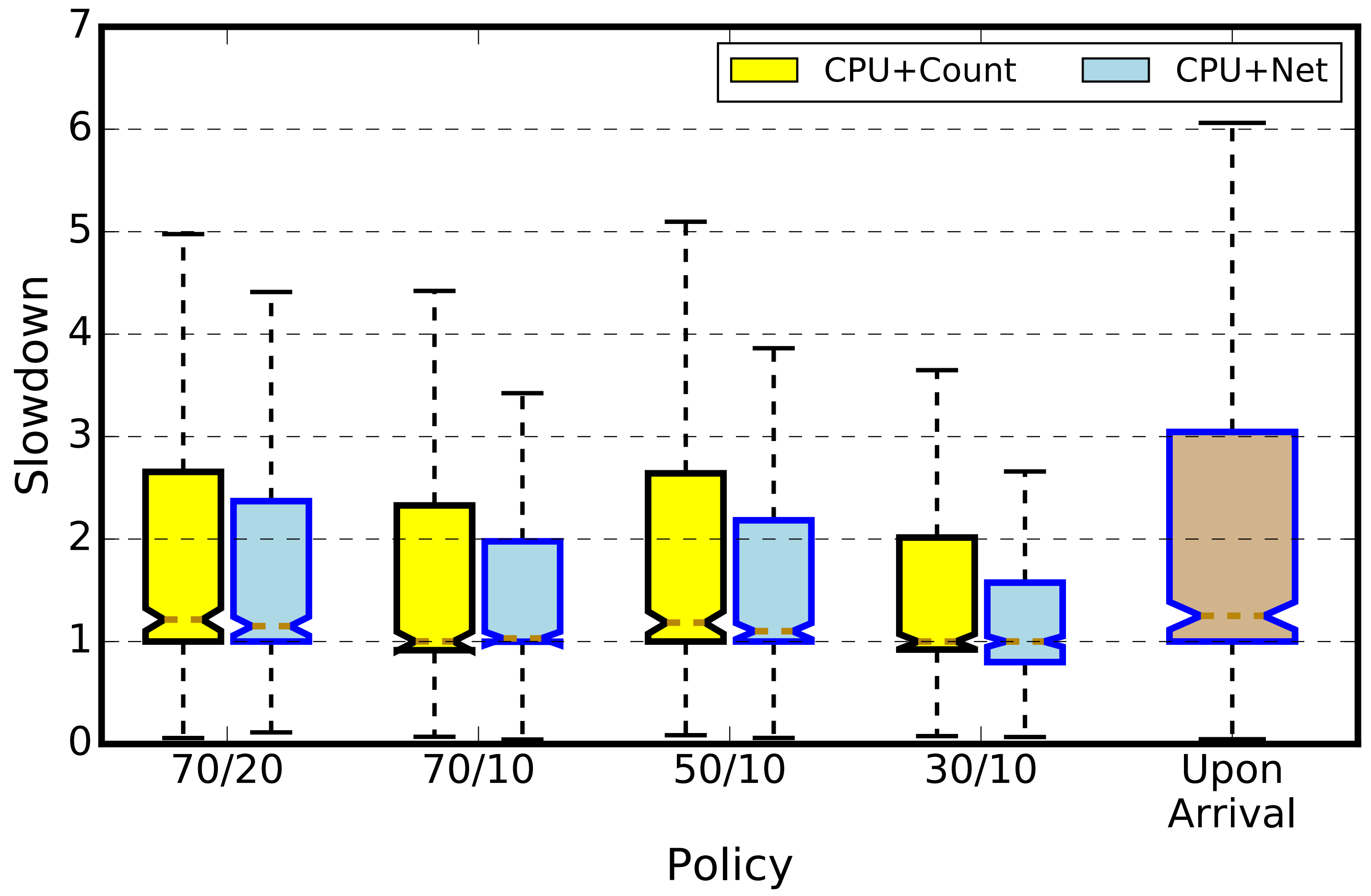
Although most DTNs are not busy, some are very heavily used and sometimes DTN resources are insufficient, i.e., very high fluctuating demand

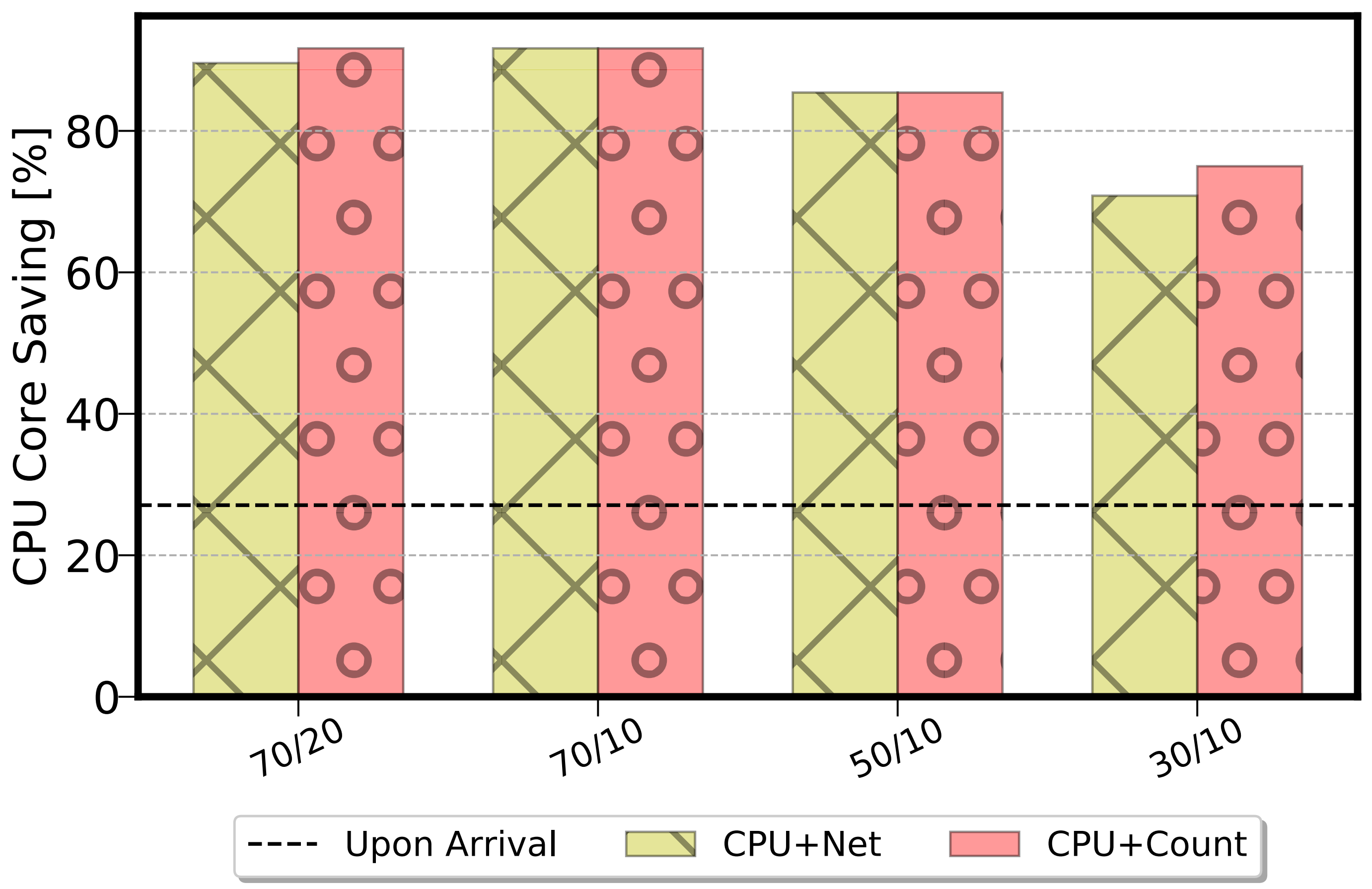


Dynamic Science DMZ architecture

TABLE I: Elastic DTI Design Space

When	Where
Usage threshold	Available core count
Usage threshold	Available network capacity
Upon arrival of new transfer	Available core count
Upon arrival of new transfer	Available network capacity





Preliminary results: Policy evaluation and resource saving compare with static

Collaborating with **Joaquin Chung**

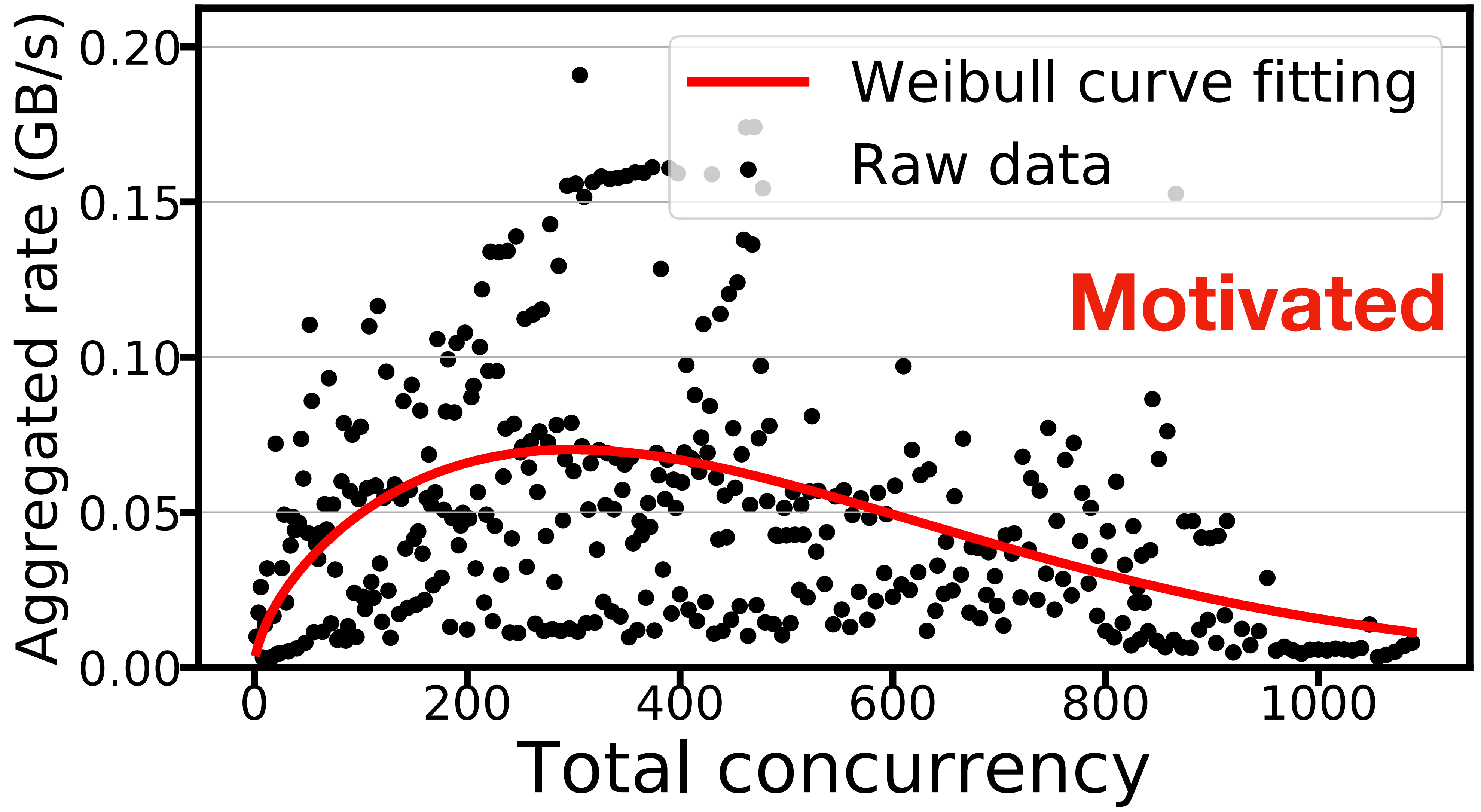
Observation motivated optimization — C3
(Real problem motivated)

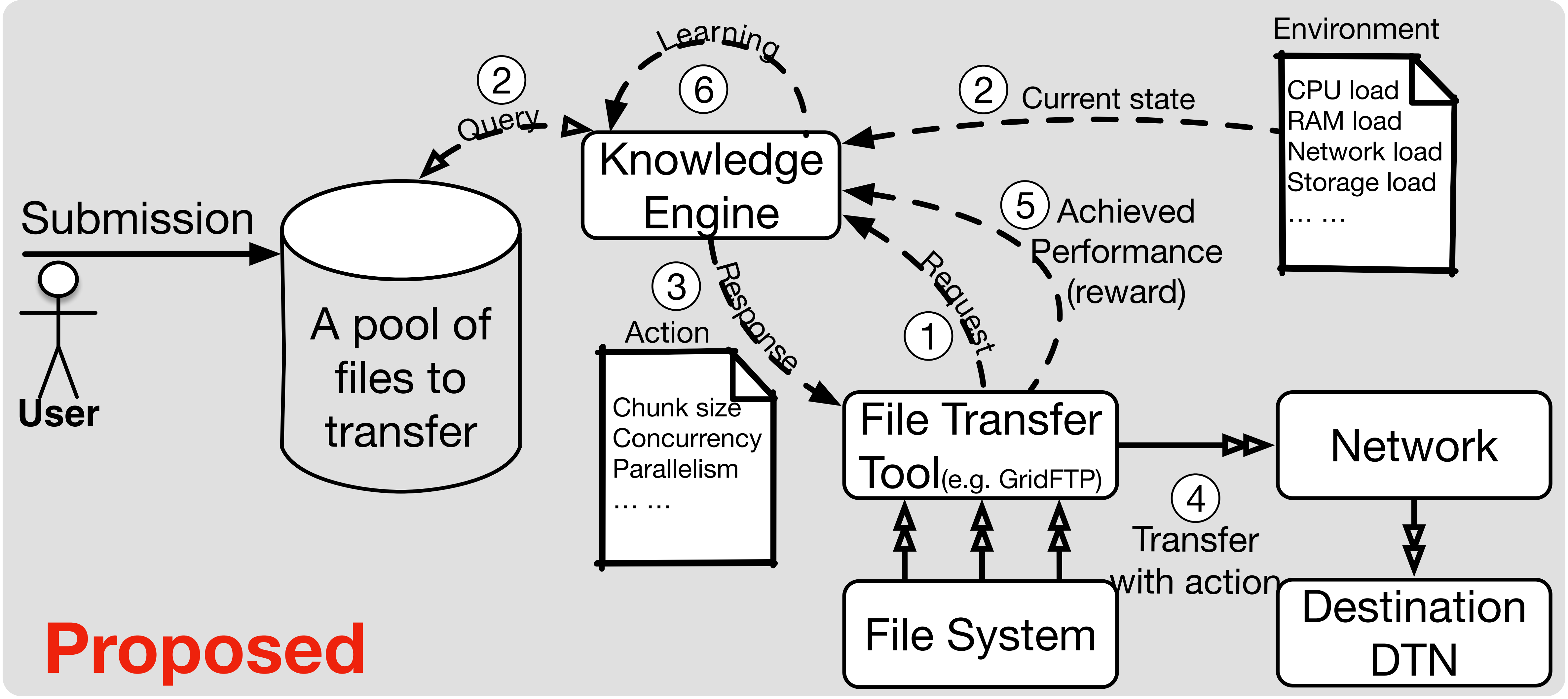
Lots of dynamic things affect file transfer performance, can we

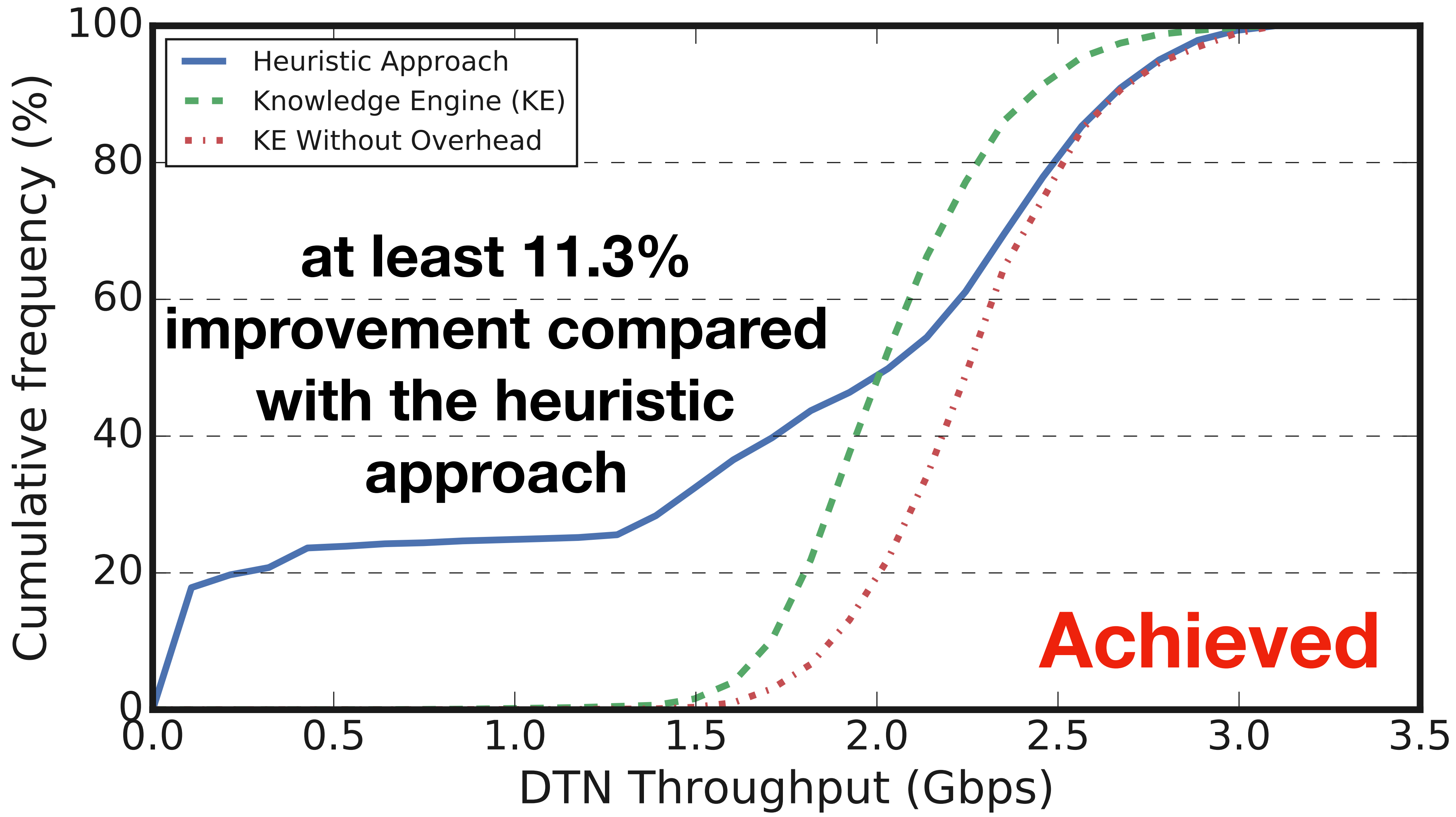
Let AI (Reinforcement Machine Learning) Optimize Data Transfer Node Smartly and Automatically

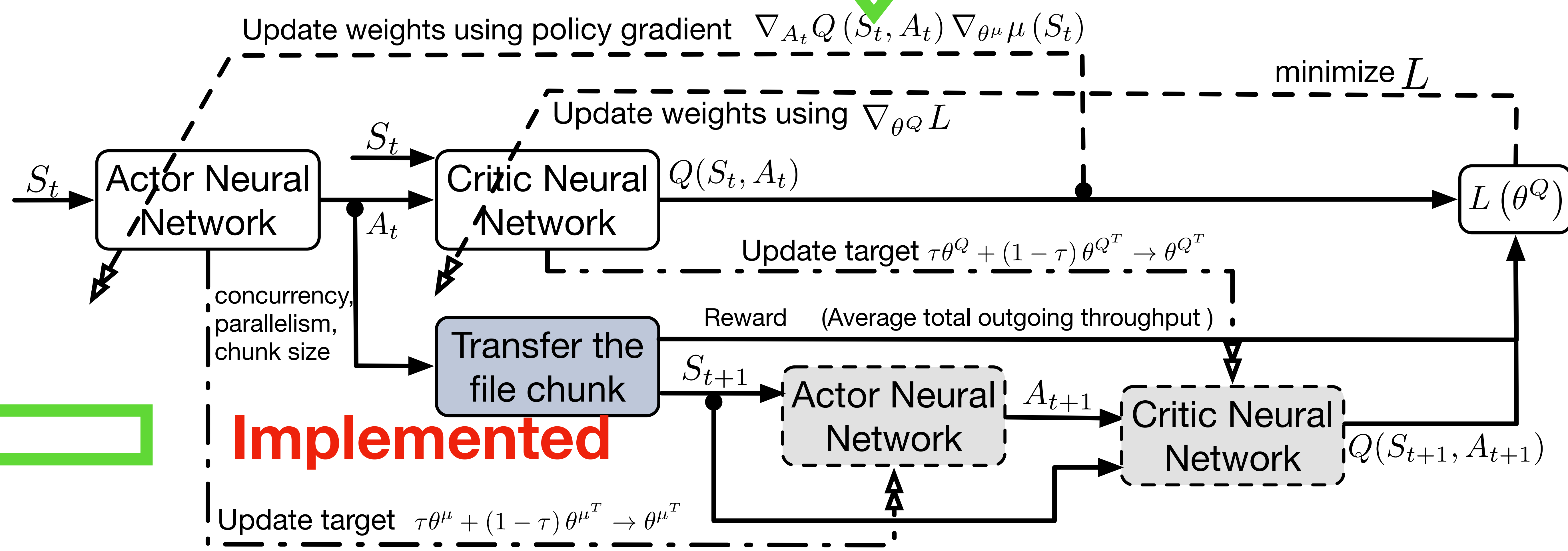
J. FGCS 2018

Data transfer: *Optimization(smartDTN)*









Two more success stories about file transfer

- **A comprehensive study of wide area data movement at a scientific computing facility**
  we characterized the network traffic of a computer facility's DTNs at multiple levels, from user transfer requests down to TCP flows. Load imbalances and opportunities for improvement are identified for planning system upgrades and future investments. *SNTA@ICDCS 2018*

- **Model to achieve sustainable high-throughput, e.g., 1PiB in 6 hours and lessons learnt**
  Lesson learnt and model built to move one PiB in a day for pipeline execution of a cosmology workflow. Then, one PiB in six hours was achieved in 2018.

Several more are proposed to funding agencies seeking support.

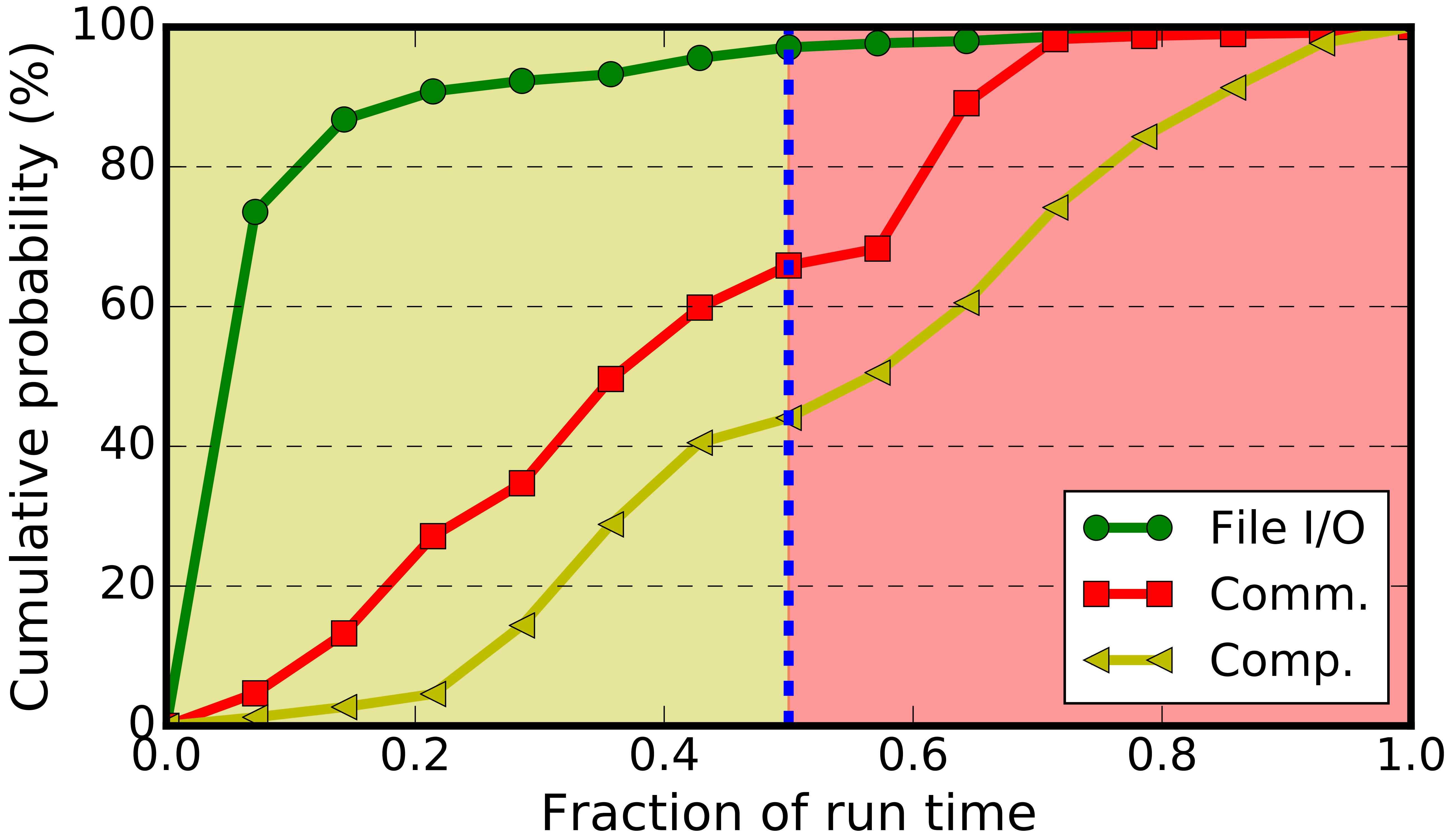
Combine, correlate and analyze logs of a Computing Facility
(ongoing)

- **Darshan:** Instruments the I/O behavior of production applications. It records statistics, such as the number of files opened, time spent performing I/O, and the amount of data accessed by an application as well as the I/O library used.
- **Autoperf**: collects hardware performance counter and MPI information.
- **Scheduler logs**: Job properties, e.g., run time, computing resource requested / used.

Runtime break

Darshan logs capture the time spent on each file using either MPIIO or POSIX IO library.

Autoperf logs record the average time spent on MPI functions for all processes and some hardware performance monitors.



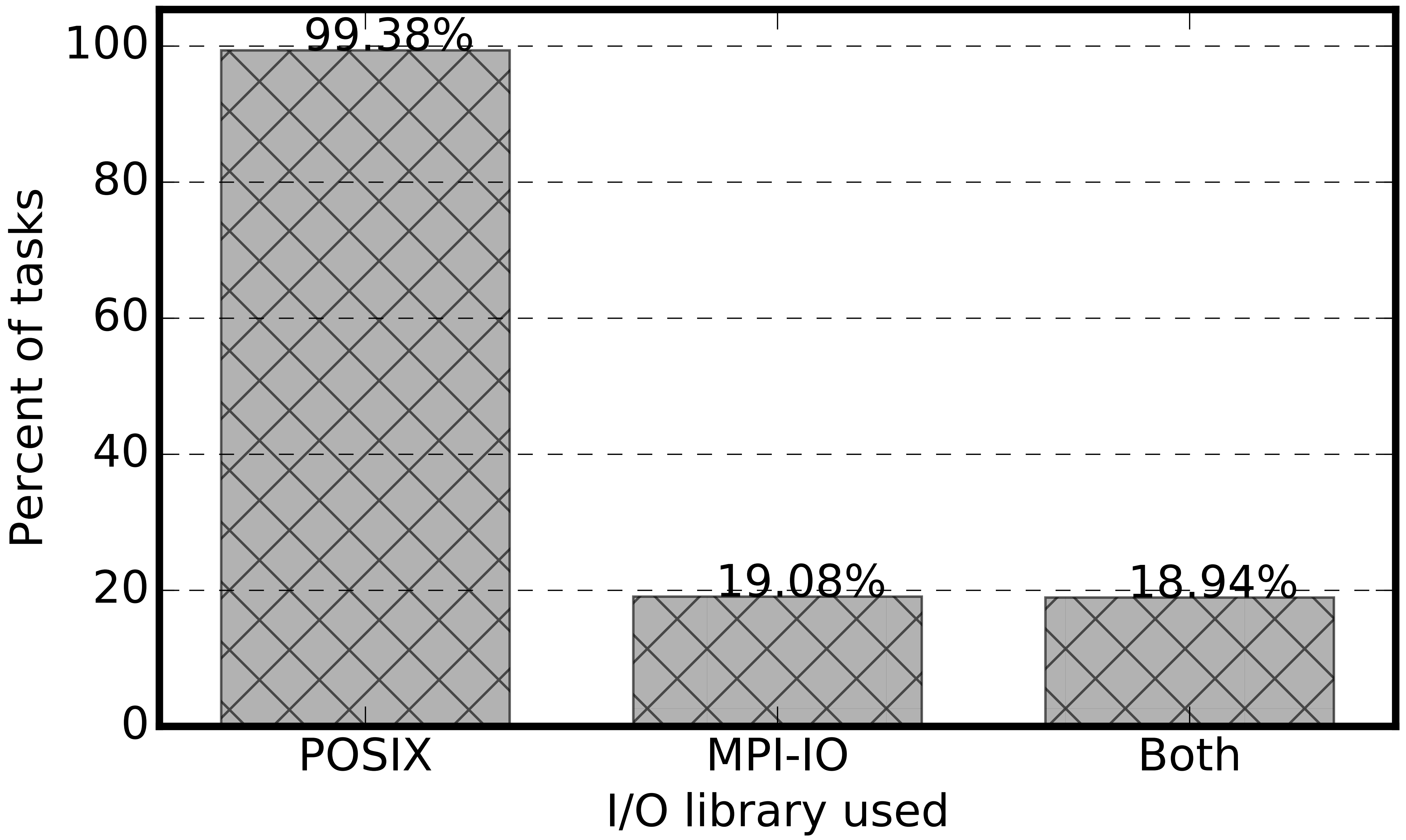
Observations:

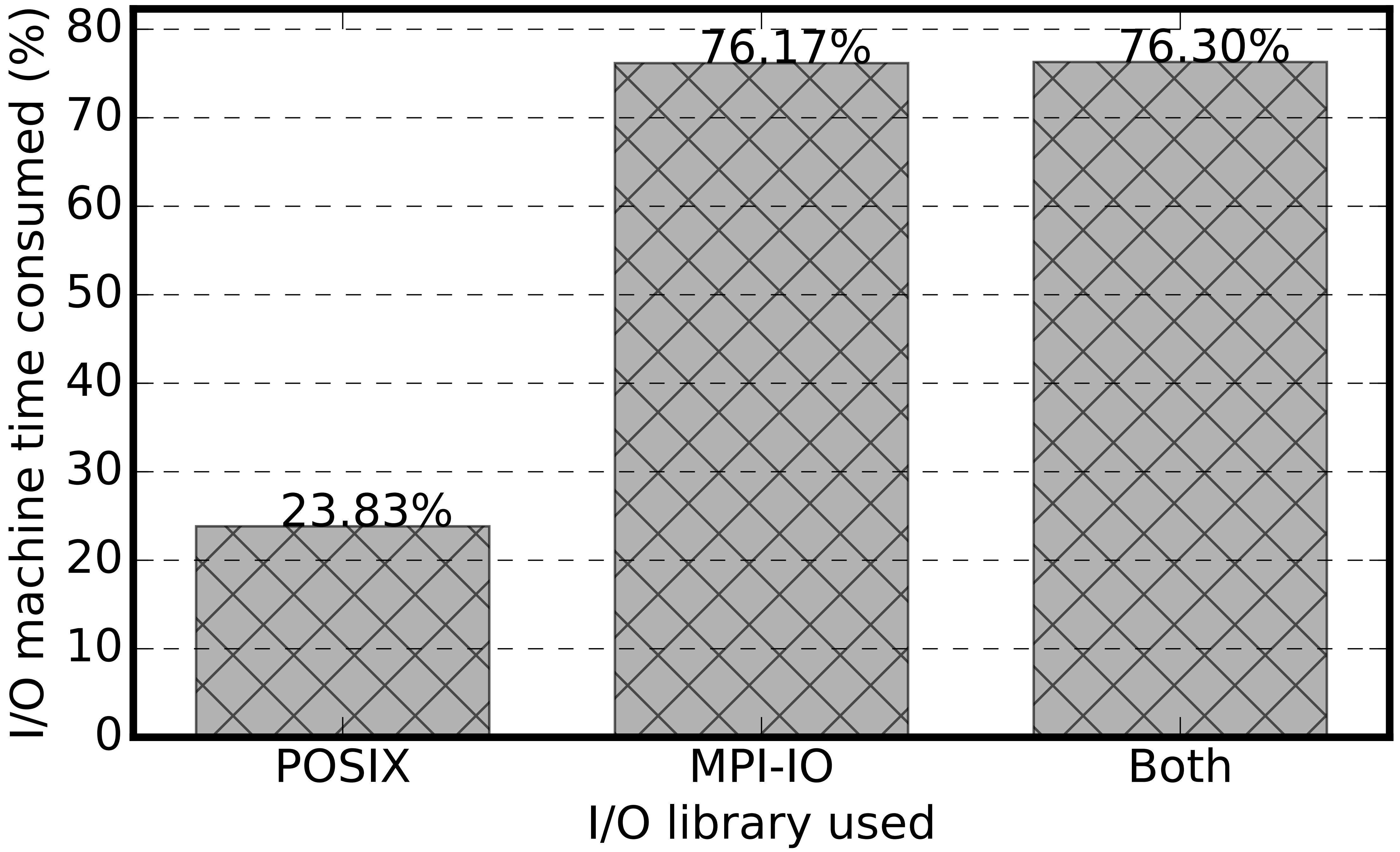
Overall, nearly 30% of the total machine time was spent within MPI. But, about 30% of tasks almost did not spend any time on MPI.

Nearly half of the tasks spent less than 2% of their total machine time on file I/O and nearly 80% of tasks spent almost zero time on file I/O. These findings reveals that computation is still the most intensive operations of HPC applications.

File I/O libraries

MPIIO (and high level libraries like HDF5, netCFD based on it) and POSIX are two libraries used. Darshan records the time spent on each library calls. As for which file I/O library are mostly used in HPC applications, we studied the number of applications that used POSIX or MPIIO, or both.





Observations: In terms of I/O library used frequency, POSIX is much more widely used than MPIIO. Nevertheless, MPIIO consumed significantly more (> 3X) machine time than POSIX did. Although POSIX are used for nearly all applications, it only consumed about 25% of the machine time. This may indicate that developers mostly use MPIIO for large data read and write.

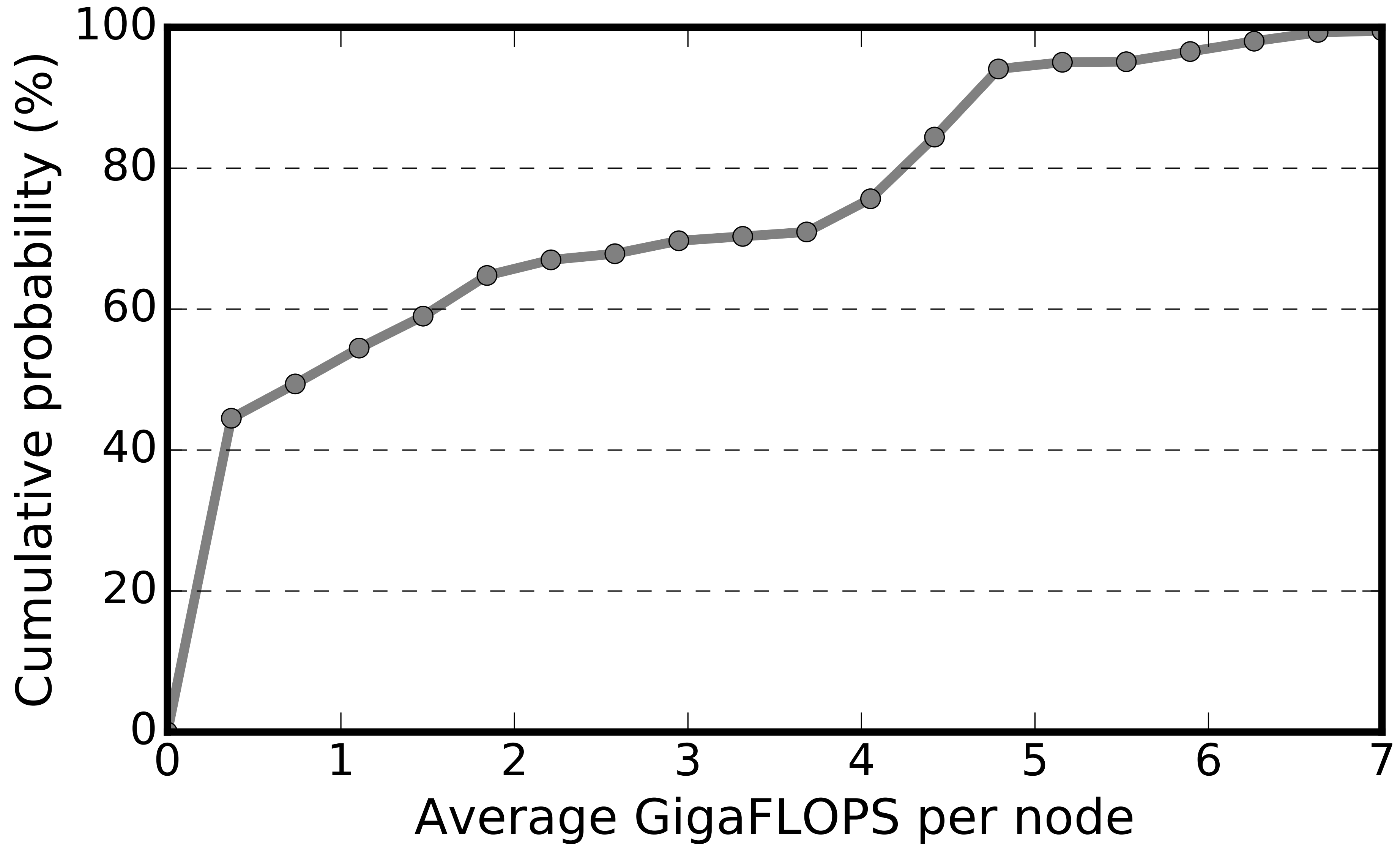
Motivates optimizing storage system for R/W small files using POSIX?

Hardware performance - FLOPS

FLOPS is a commonly used measurement of supercomputer. Counter PEVT_INST_QFPU_ALL gives the total number of floating point operations done per MPI process. Thus,

```
PEVT_INST_QFPU_ALL * numProcessesOnNode / elapsedTime
```

Gives the achieved FLOPS per node.



The peaks floating point performance of PowerPC A2 processor is (1.66 GHz) x (16 cores) x (4 vector lane) x (2 operations per FMA) = 212.48 GFLOPS/node.

Surprising? Recall our observation "Overall, nearly 30% of the total machine time was spent within MPI. But, about 30% of tasks almost did not spend any time on MPI." HPC application does much more than floating point operation; theoretical FLOPS is hard to achieve.

Hardware performance - OPS

`PEVT_INST_XU_ALL + PEVT_INST_QFPU_ALL` gives the total number of operation.

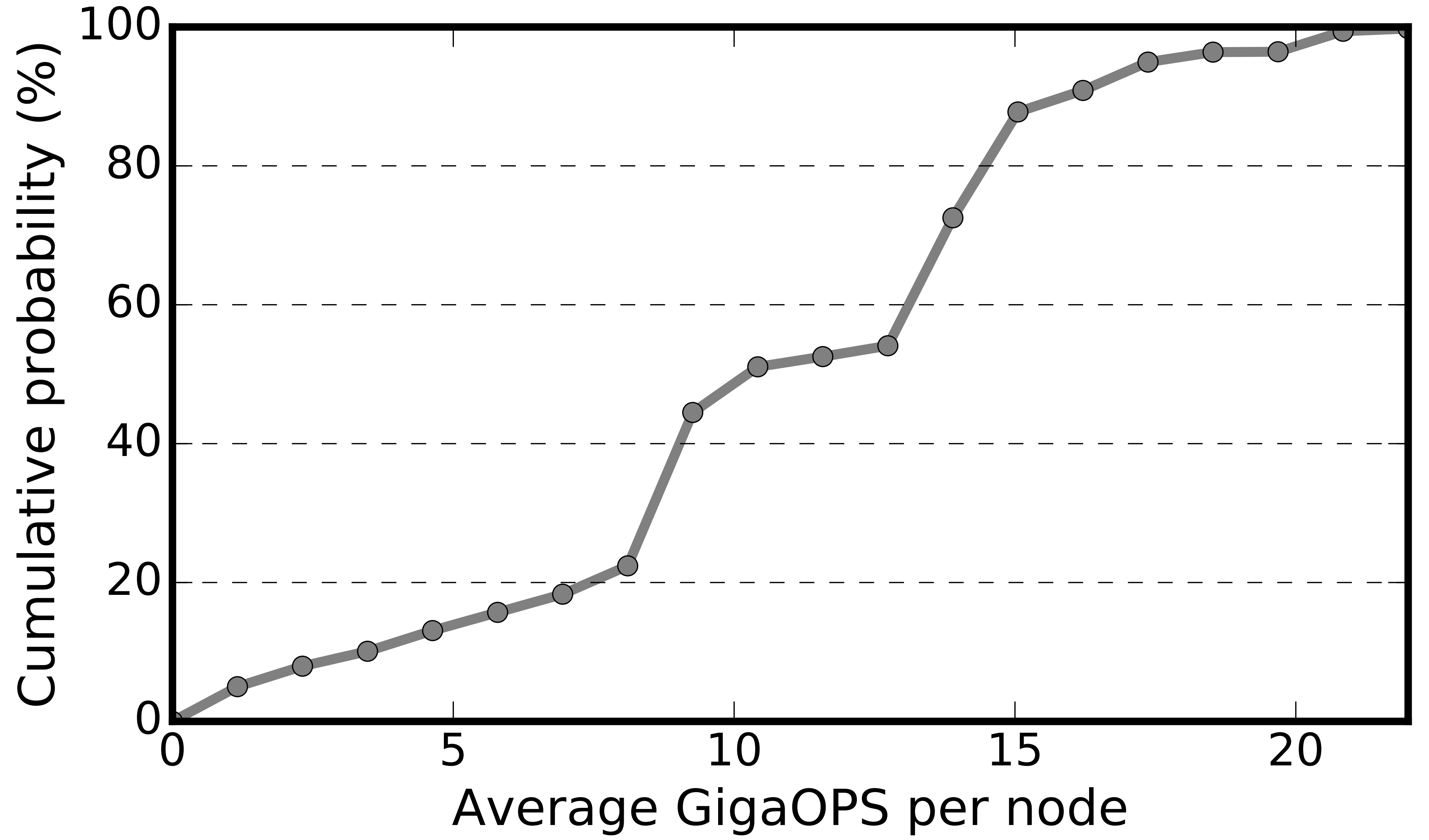
Similarly,
`(PEVT_INST_XU_ALL + PEVT_INST_QFPU_ALL) * numProcessesOnNode / elapsedTime`
Gives the archived operations per second per node.

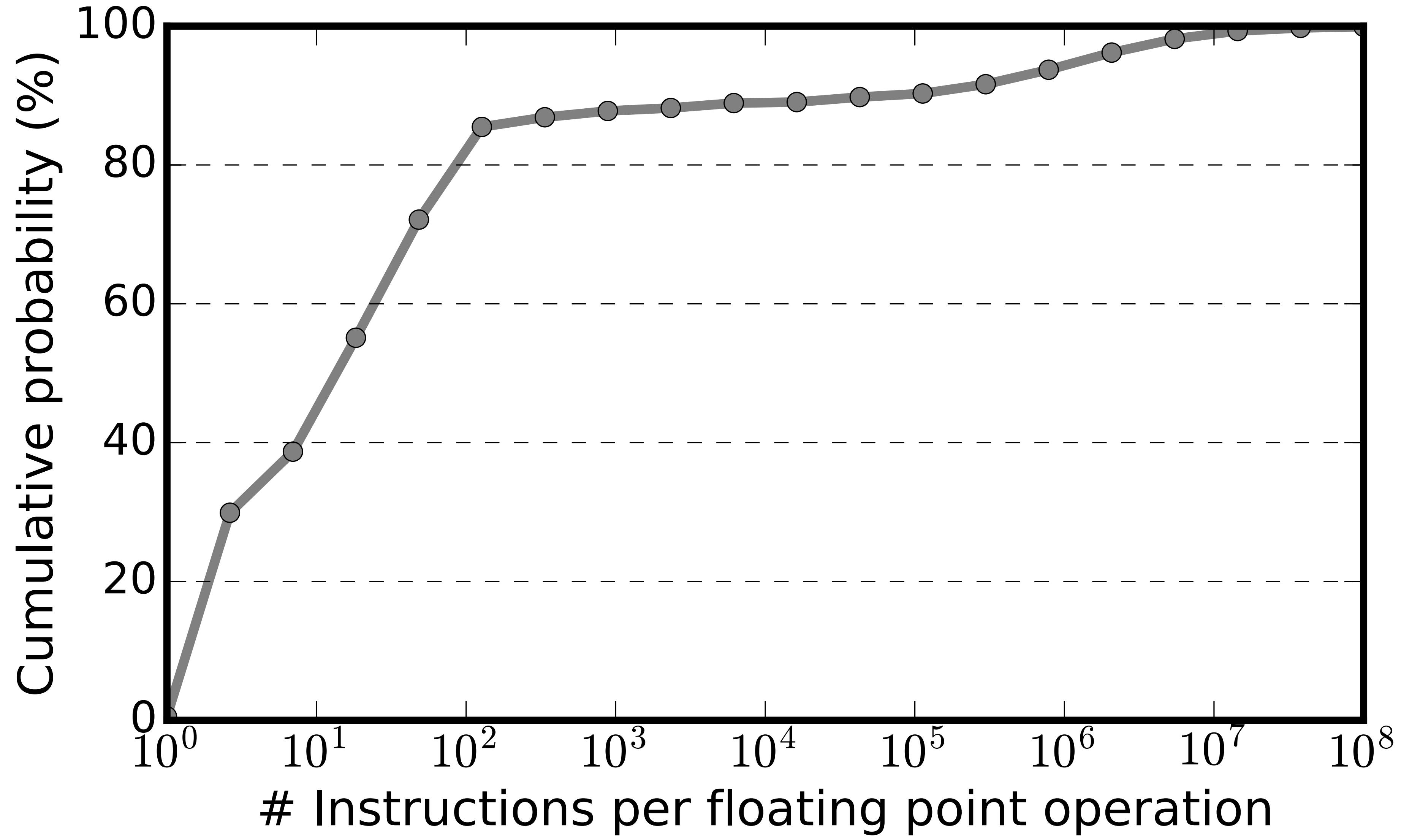
Intuitively, for each float pointing point operation, there are multiple other operations. Here we calculated:

`PEVT_INST_XU_ALL / PEVT_INST_QFPU_ALL`

to get insights of the average number of non-floating operations for each floating point operation.

Observations: Most applications are not floating point operations intensive. Theoretical FLOPS is really hard to achieve by actual applications.



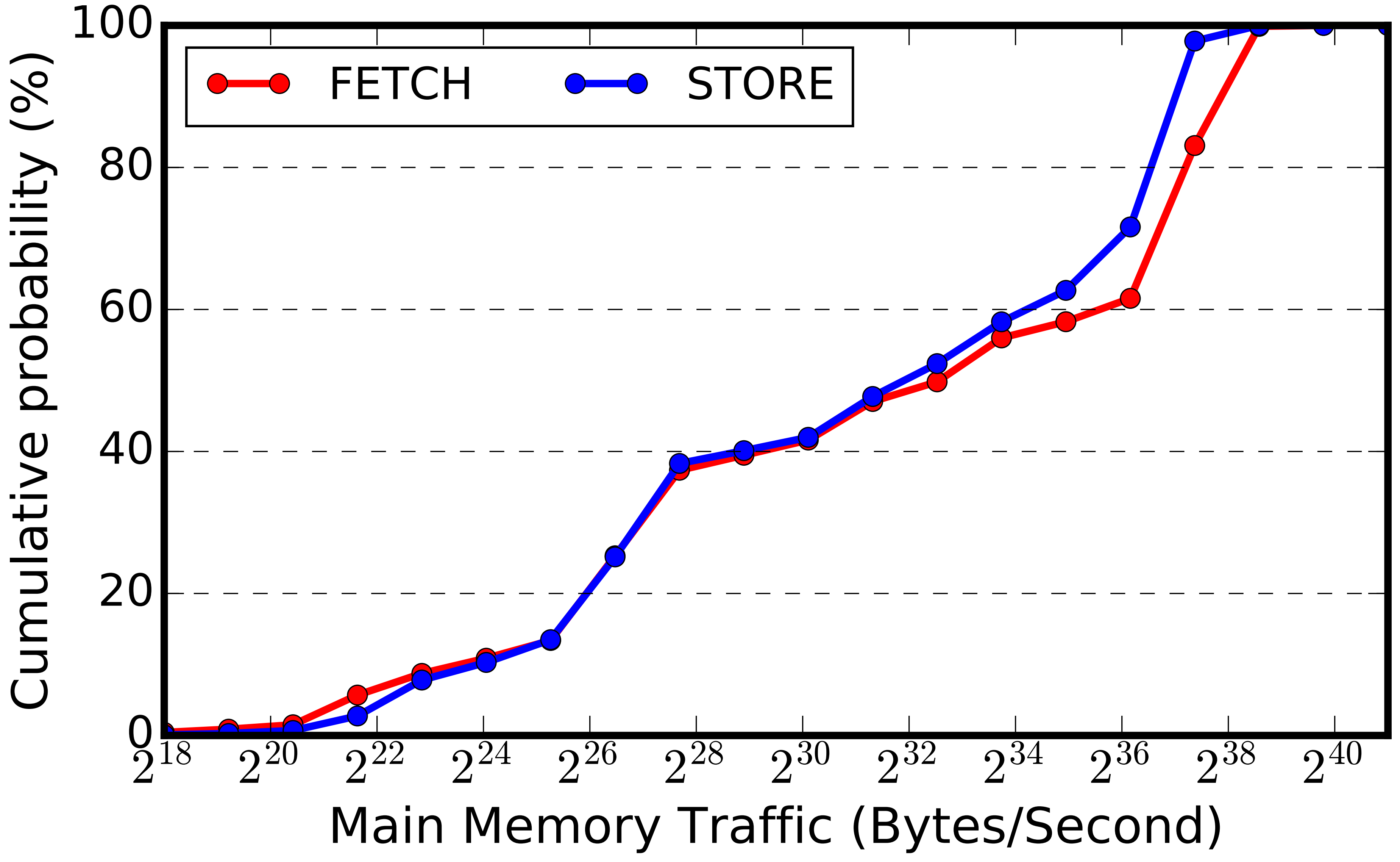


Hardware performance - Memory access

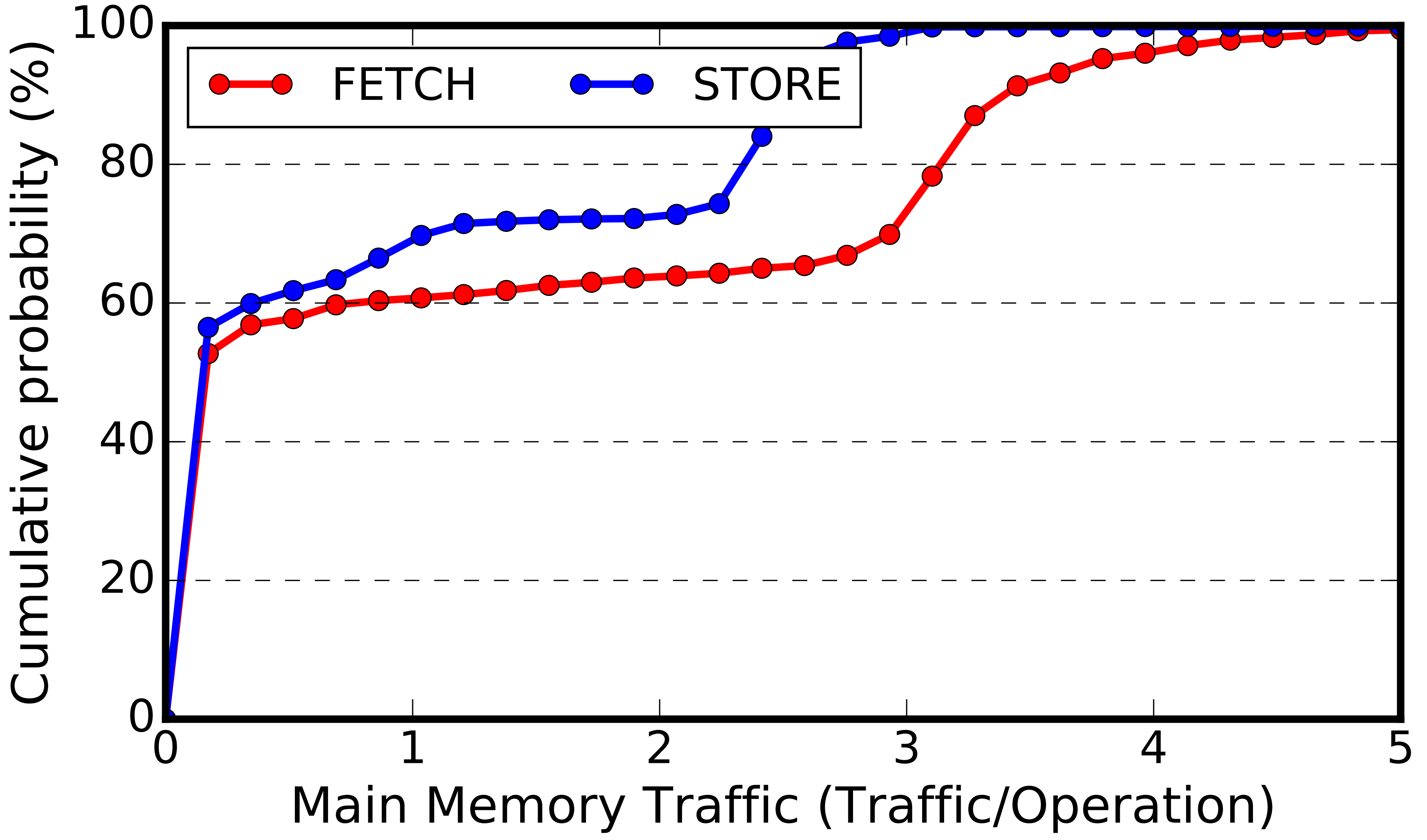
Counter PEVT_L2_FETCH_LINE **and** PEVT_L2_STORE_LINE **give the total number of RAM FETCH and STORE traffic separately.** Each STORE/FETCH transfers 128 bytes. Thus we can calculate the archived main memory throughput.

Each Mira node is equipped with 16GB 1.333GHz DDR3 memory with peak 42.6~GB/s bandwidth.

Thanks to cache, the archived rate can be more than RAM's limit. More than 40% of the jobs achieved more than RAM's limit due to proper use of cache.



In order to investigate if an application is memory bounded or computing bounded, we calculate the average main memory traffic (128 bytes) per operations. [we need memory latency and IPS to conclude]



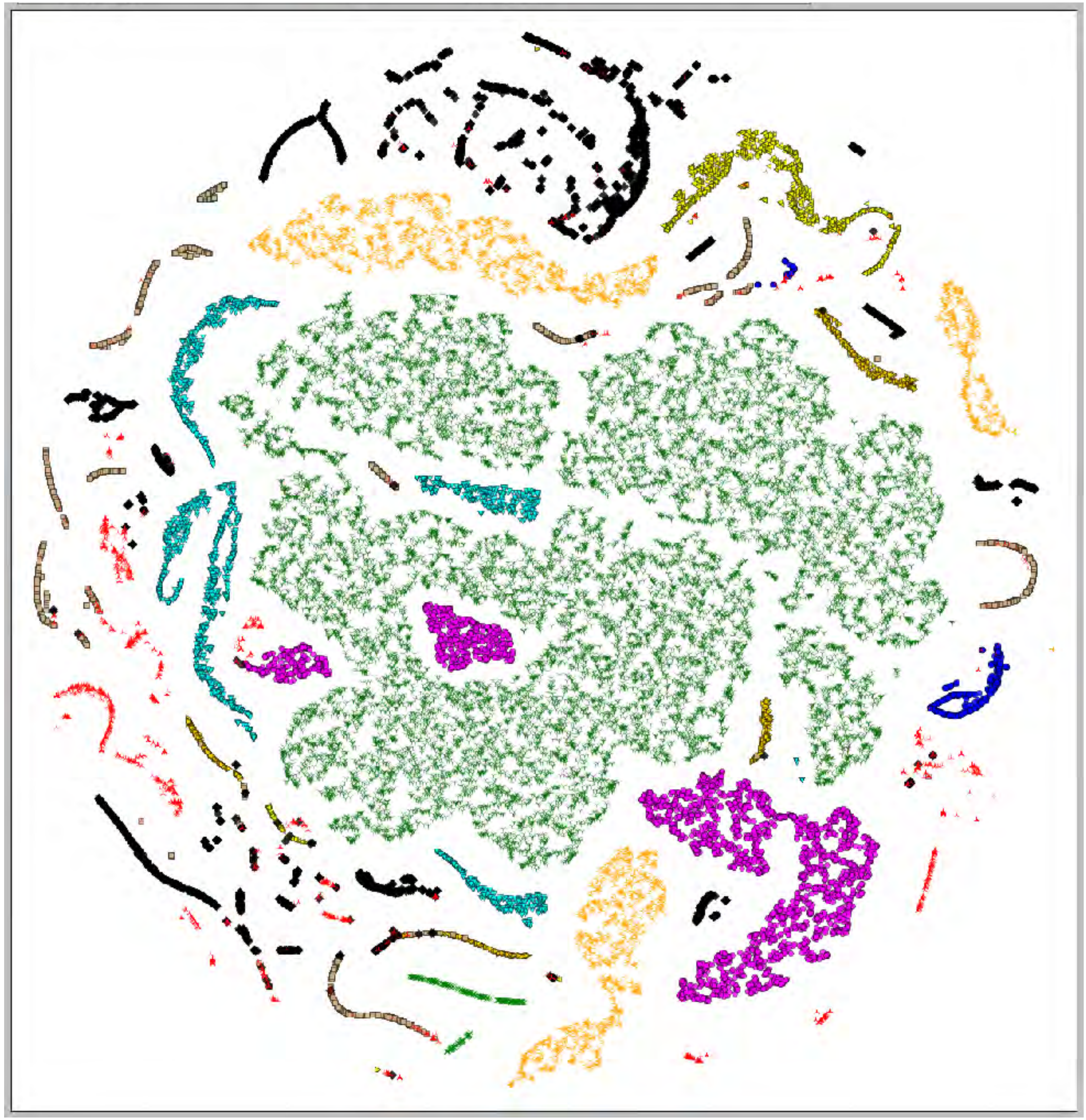
Task grouping

We extract features to represent a task:

- Fraction of communication time
- Fraction of MPI-IO time
- Operations per second and FLOPS
- Process per Node
- RAM fetch/store per cycle
- and more…

And then visualize high dimension features with t-SNE. Each application is marked by a unique color when plotting.

Observations: We can clearly see clusters in the feature representation. We may be able to seek explanations for those clusters with users' project information.



t-SNE visualization of task characterization

Observations: It is clearly possible to build “signature" of each task based on the existing logs, with this "signature", we are able to:

(1) understand if the running application is what proposed to run;
(2) Since "signature" is built on performance counters, we can use it to group applications (taxonomy, e.g., I/O intensive, communication intensive, computing intensive or memory bounded) for optimization and special servicing purpose.
(3) Adding energy consumption dimension could help categorize applications / projects / users for better energy-cap control.
(4) Trained a XGB classifier with 70% tasks for top 20 (covers 94.9% tasks) applications in 2018, testing accuracy is 99.5%. i.e., given 6 “signature” features, classifier can figure out which application it is, proves the interpretability of the features.

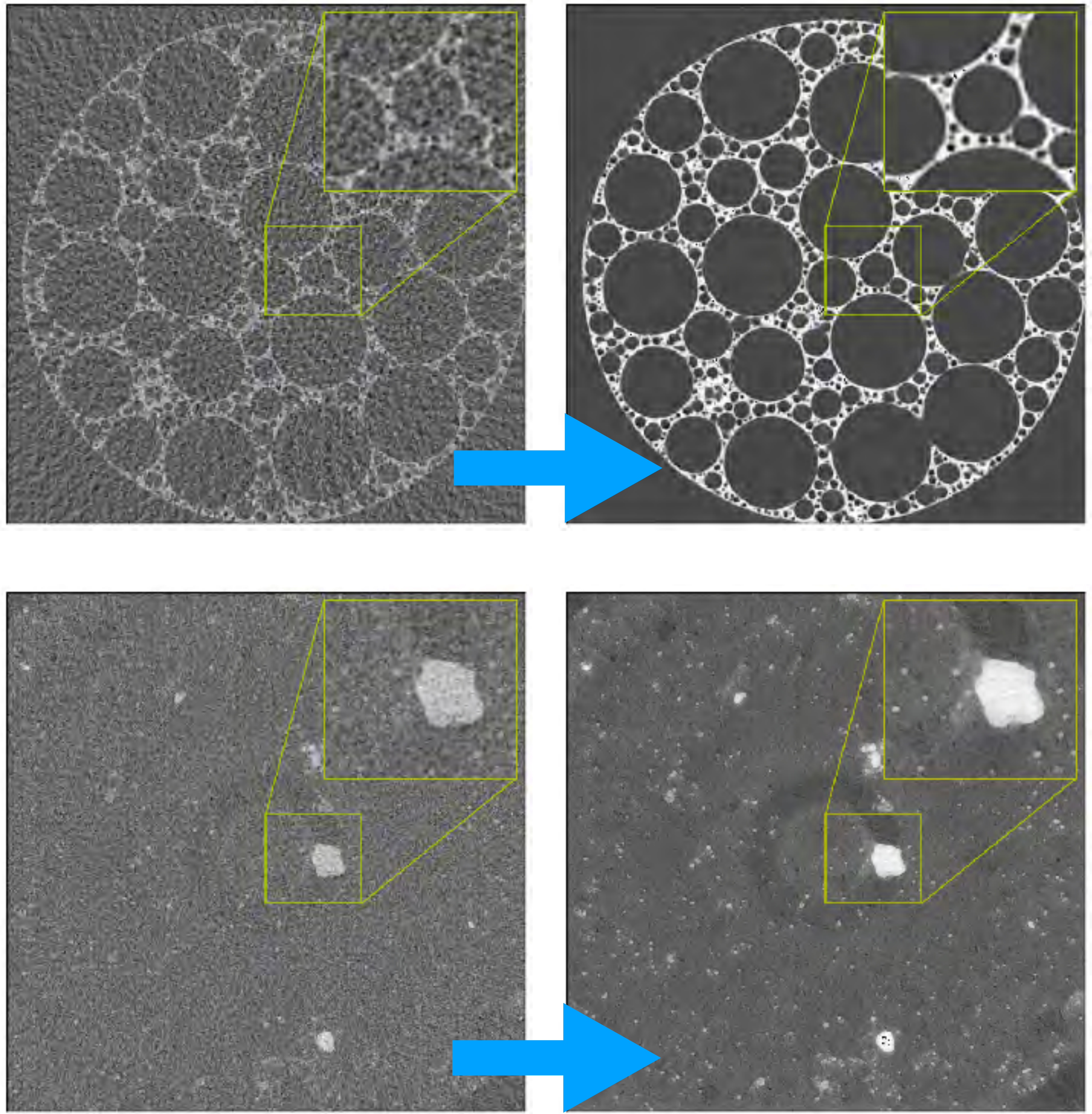
It’s about data science and machine learning, can we help a little big for lightsource facility data?

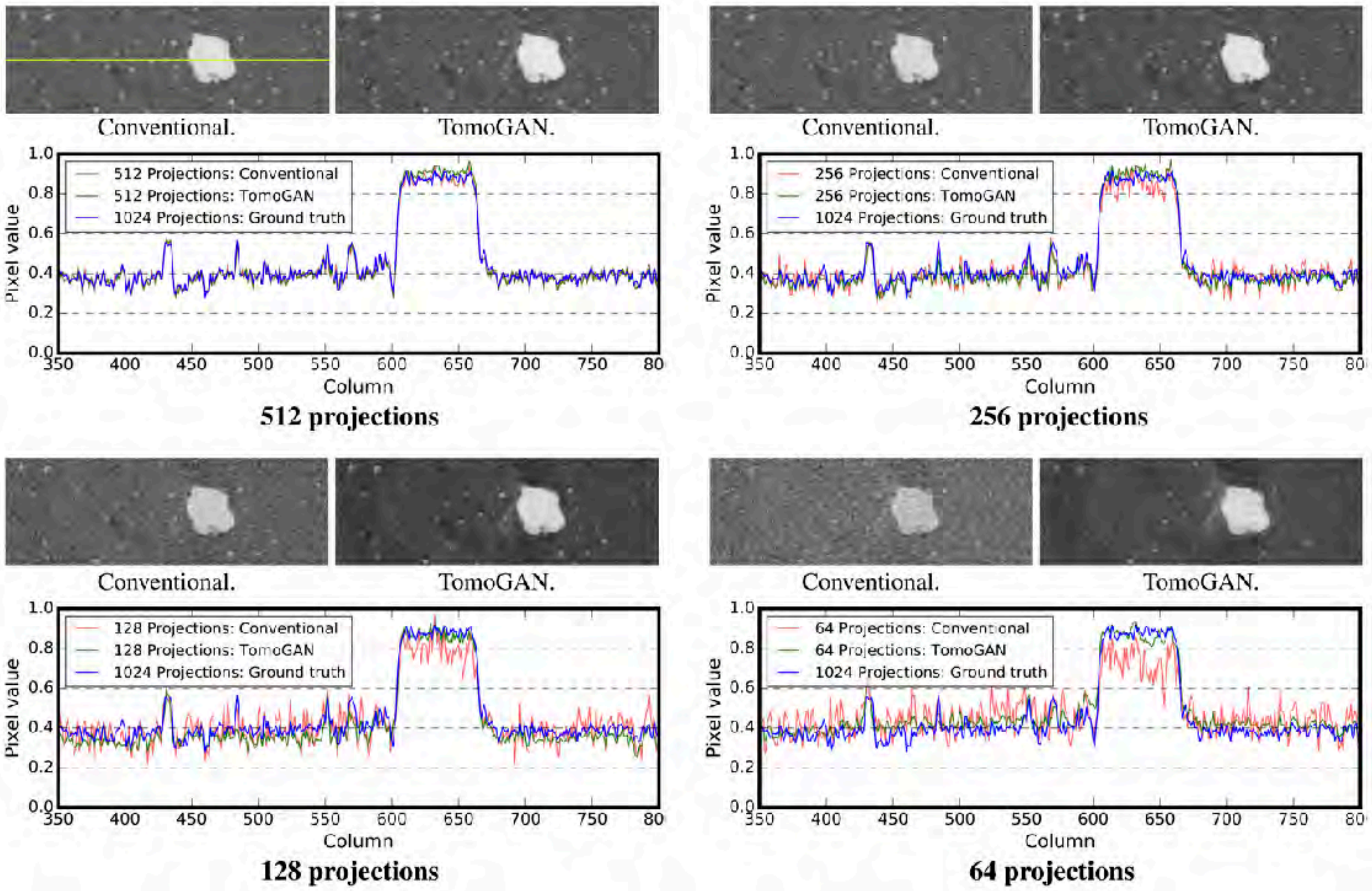
Liu et al. arXiv: 1902.07582
MMLS'19

Intelligence: TomoGAN

Motivation:

(1) lower X-ray dosage for sensitive sample like bio-sample;
(2) faster experiment to capture dynamic features like a fast chemical reaction process.
(3) smaller dataset and less computation for [near] realtime tomography imaging.

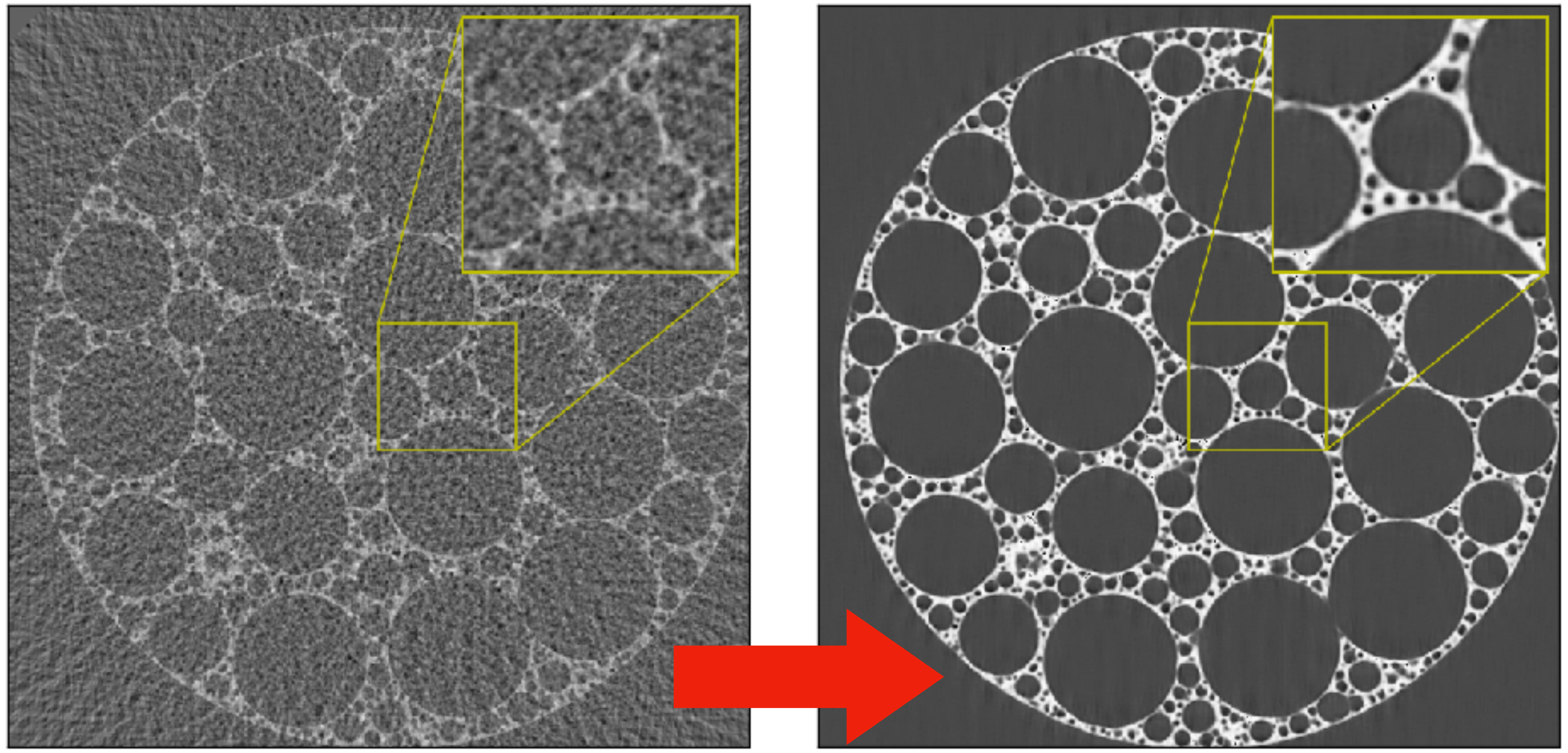




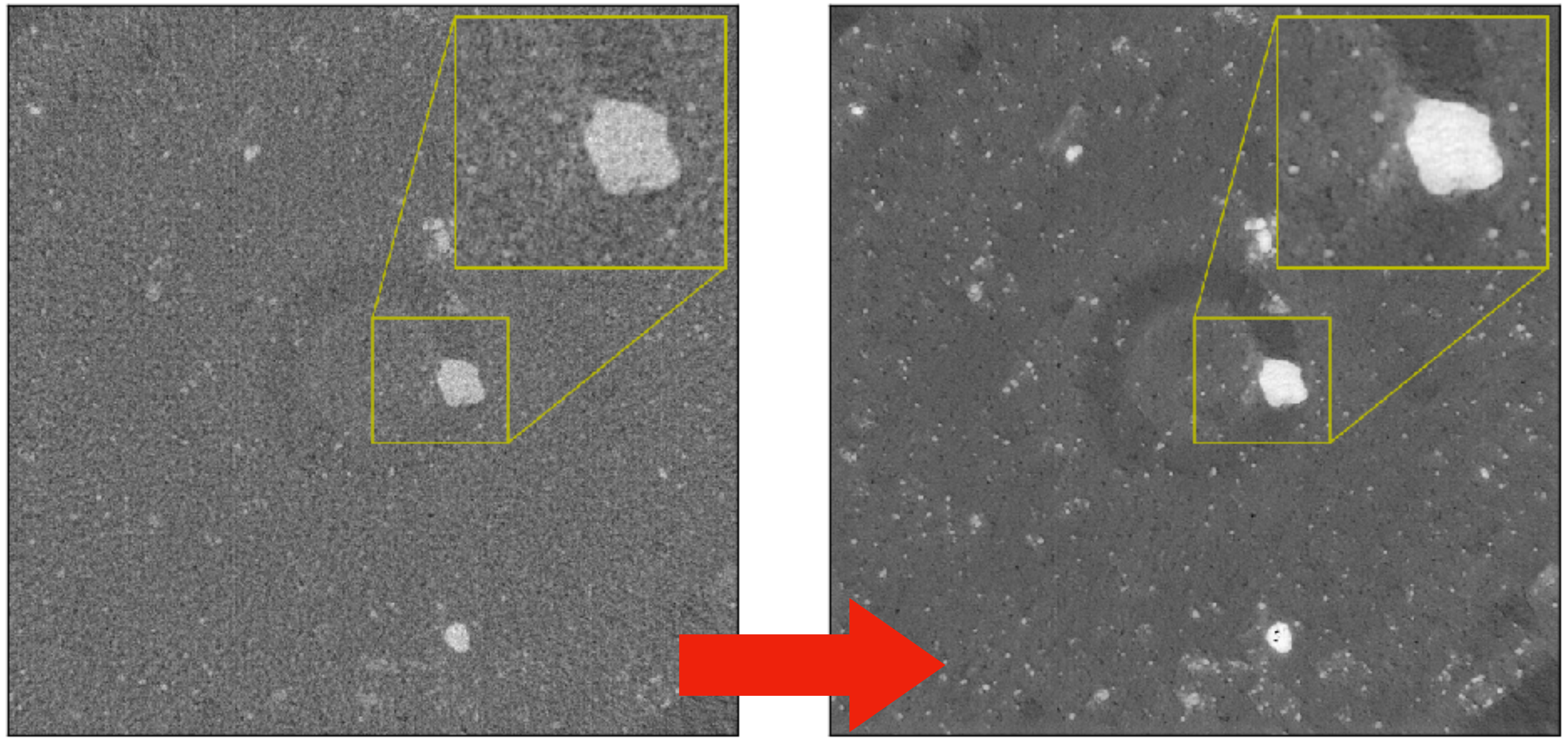
TomoGAN

Liu et al. arXiv: 1902.07582
MMLS'19

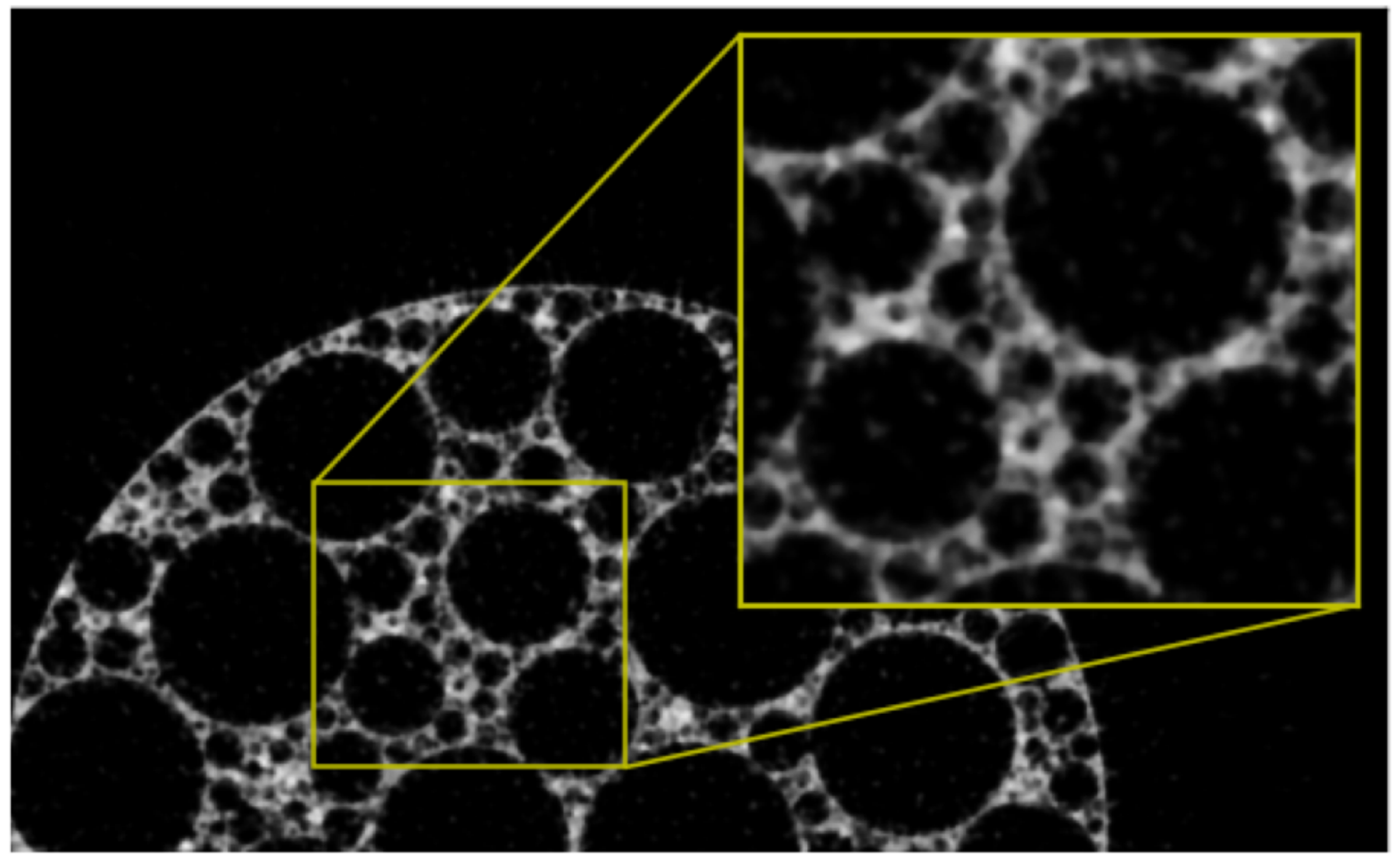
A Conditional Generative Adversarial Network for Low-Dose X-Ray Tomography



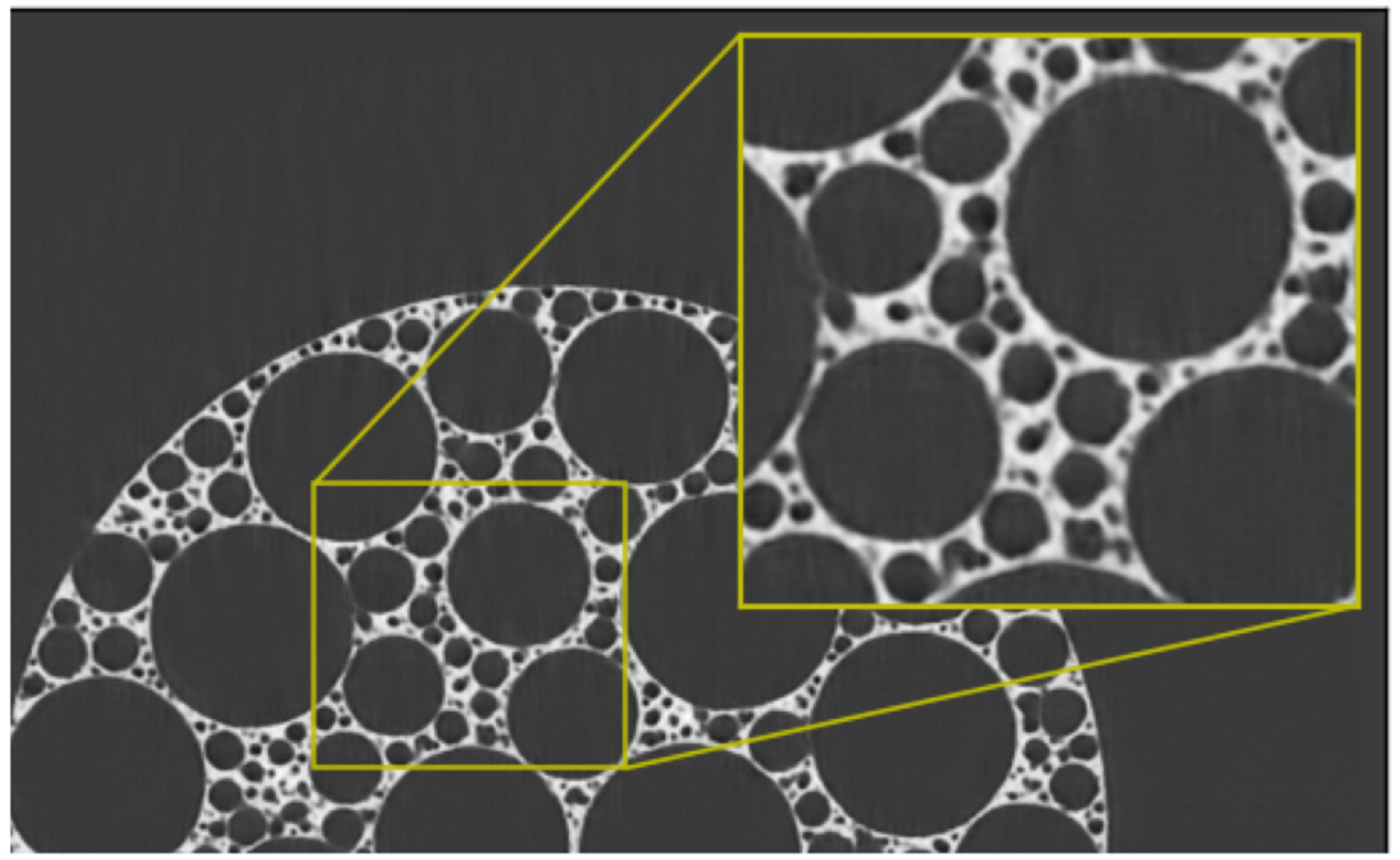
On the left, the results of conventional reconstruction, which are highly noisy. On the right, those same results after denoising with TomoGAN.



Model is trained with one shale sample imaged at APS and tested with **another** shale sample imaged at Swiss Light Source (SLS).



SIRT + total variation (conventional SOTA, 550ms)



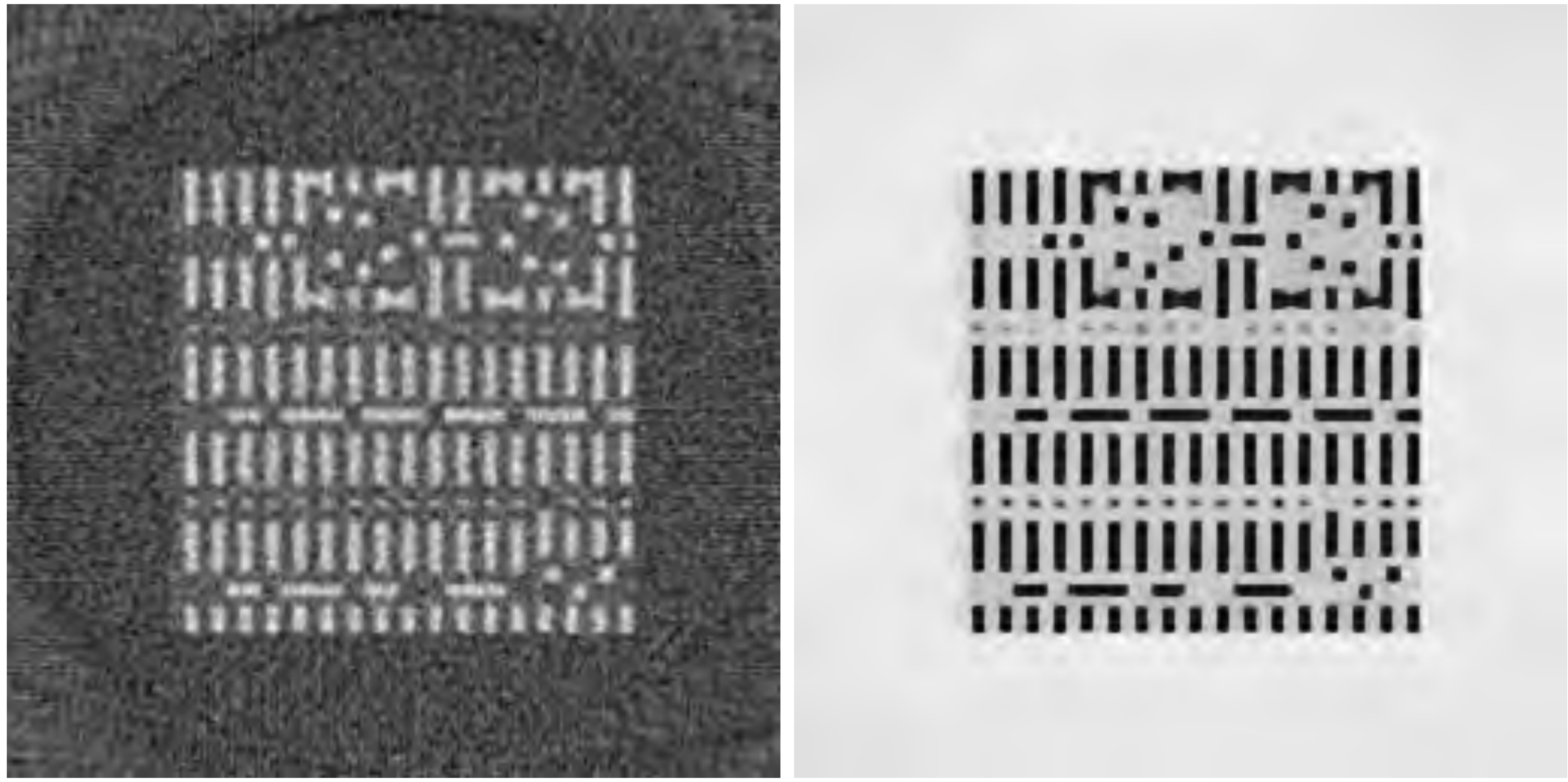
Filtered Back Projection (42ms) + TomoGAN (4ms)

FBP takes 42 ms to reconstruct one image (using TomoPy) and TomoGAN takes 4 ms to enhance the reconstruction, totals 46 ms per image. In contrast, the SIRT based solution (using TomoPy) takes 550 ms (400 iterations), i.e., 12x faster. Times are measured using one Tesla V100 graphic card. Moreover, iterative reconstruction does not provide better image quality than does our method.

TomoGAN - Continue

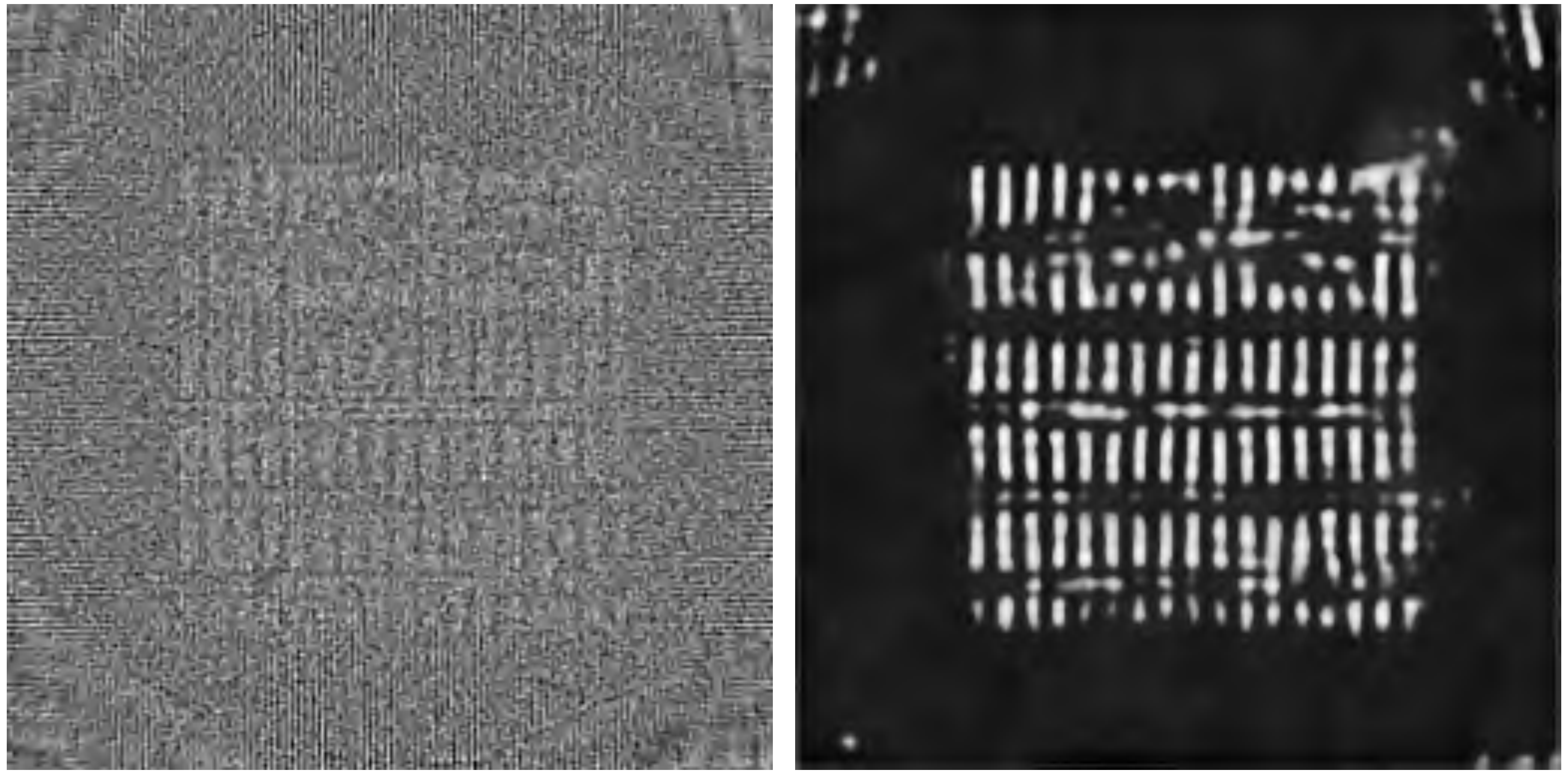
Liu et al. arXiv: 1902.07582
MMLS'19

It has been applied to the joint ptycho-tomography problem for reconstructing the complex refractive index of a 3D object.



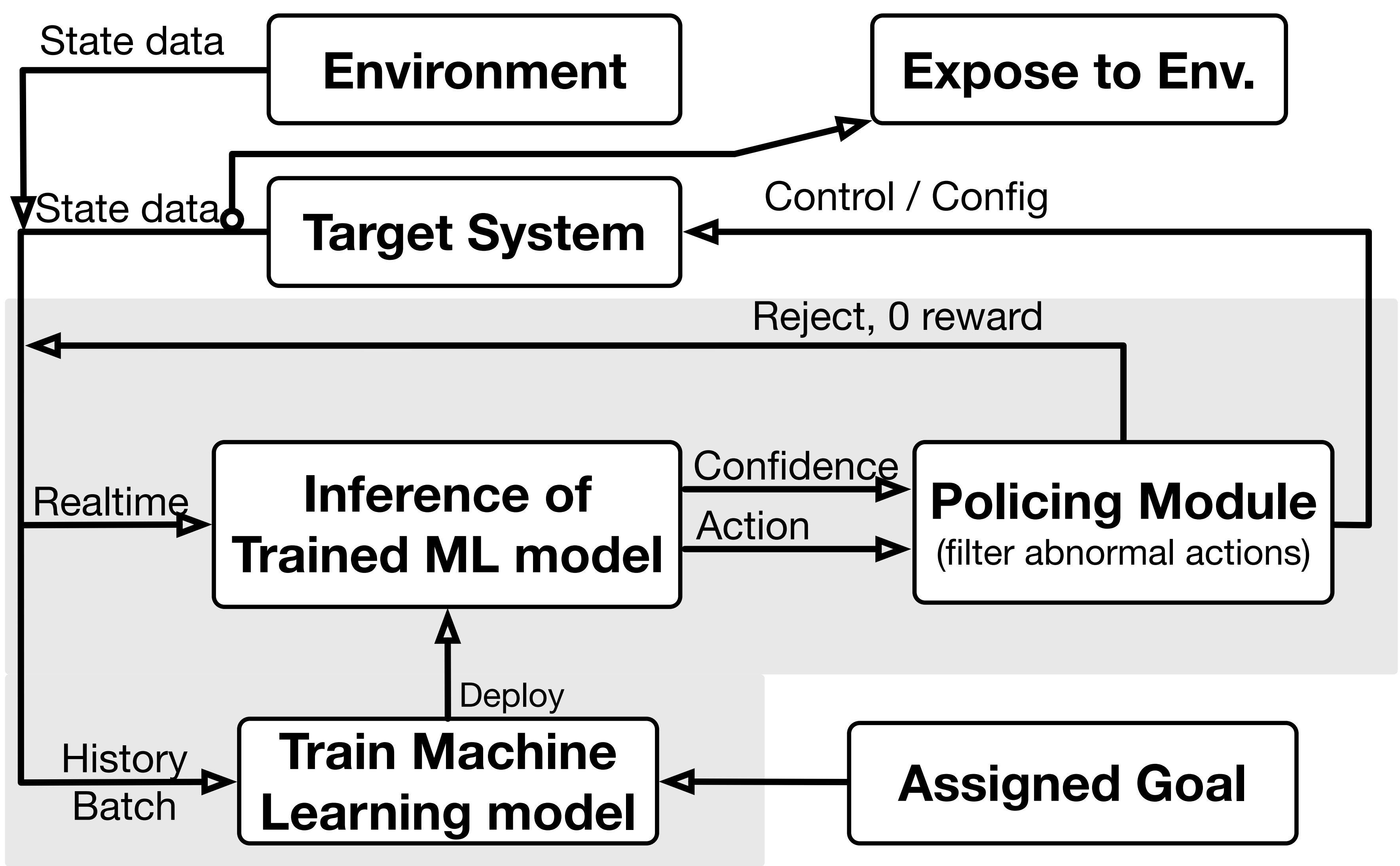
Delta, 0.003

- There is a ptychography process to reconstruct projections needed for tomography. but it is very time consuming to image the sample (month).
- Less datapoint results in noisier ptychography reconstruction and worse tomography images.
- TomoGAN here was used to enhance tomography images with less data points need to collect, i.e., faster experiment.

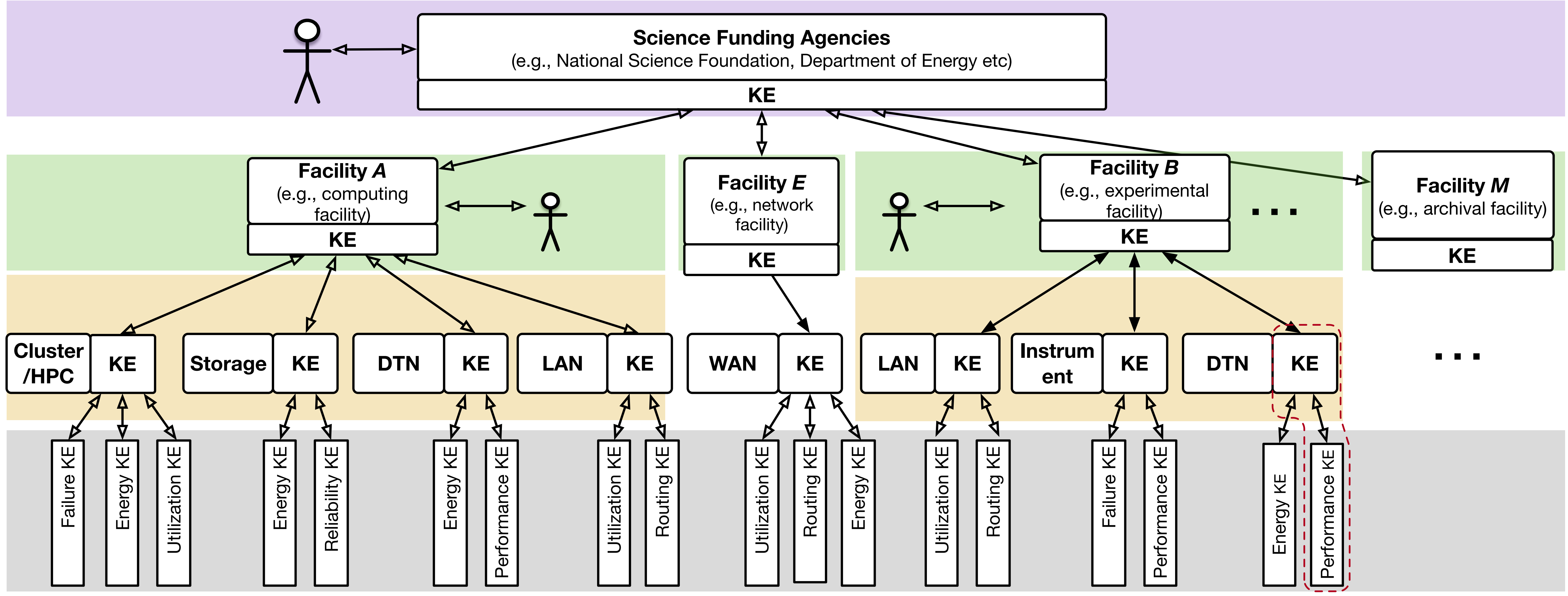


Beta, 0.03

Cyberinfrastructure (for Lightsource facilities)



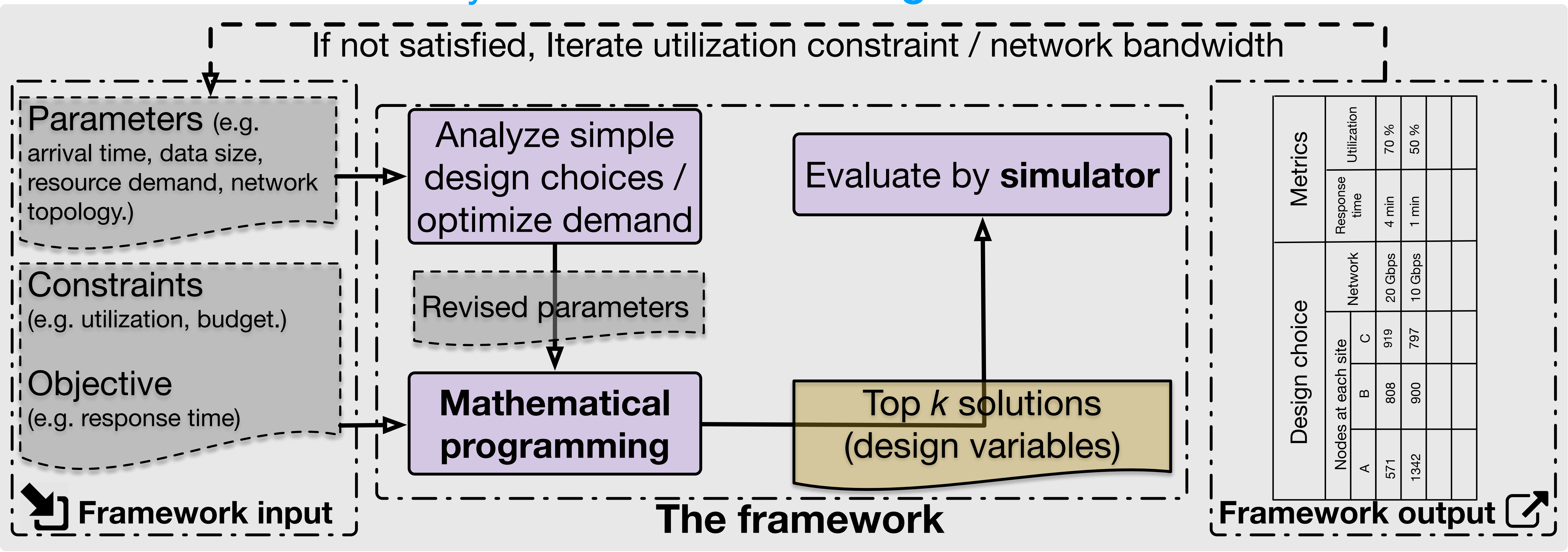
General Architecture of an autonomous system, the shaded area is the **Knowledge Engine**.



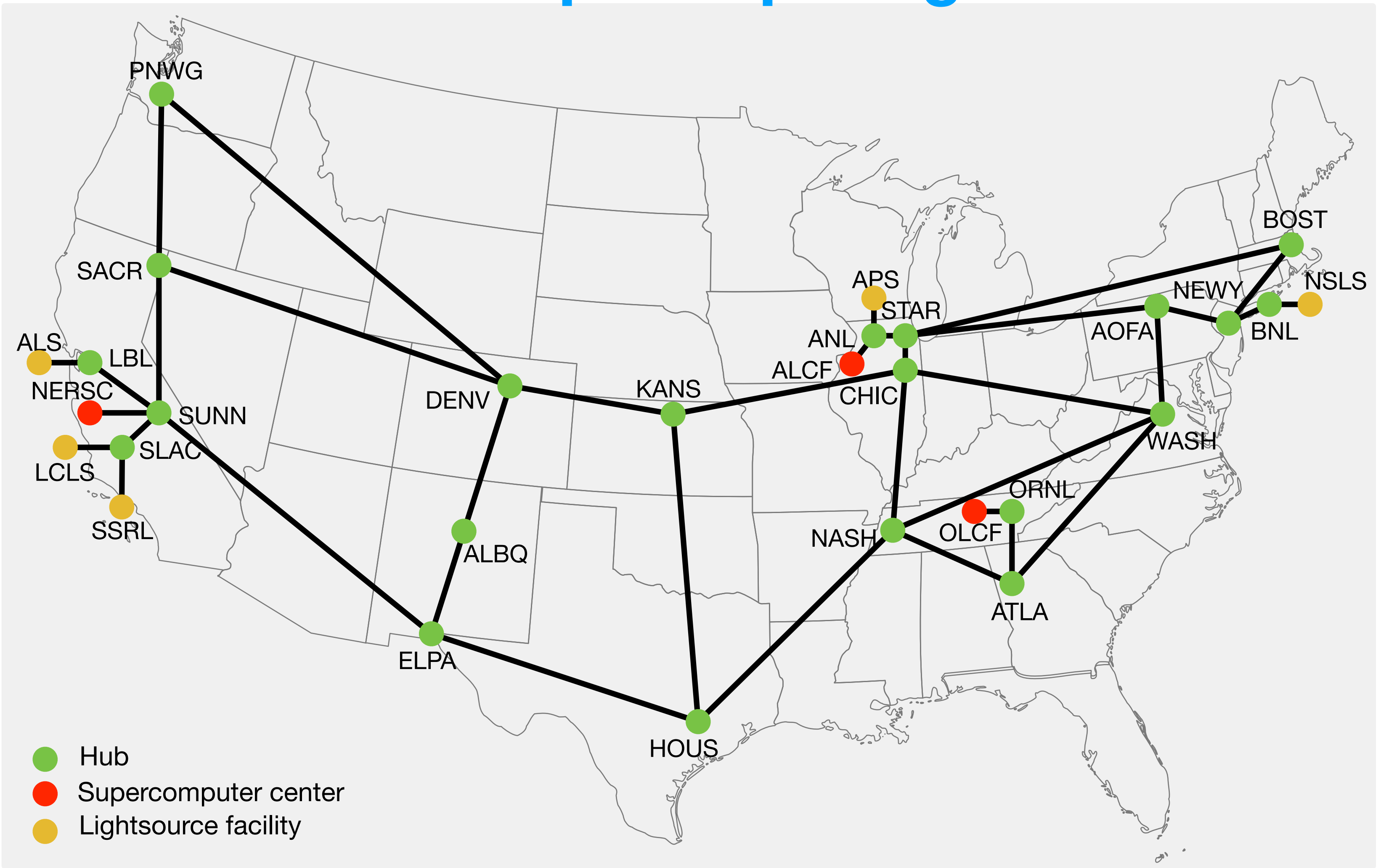
Architecture of autonomous science infrastructure.
Each node is equipped with a knowledge engine

A mathematical programming- and simulation-based framework to **evaluate** cyberinfrastructure **design choices**

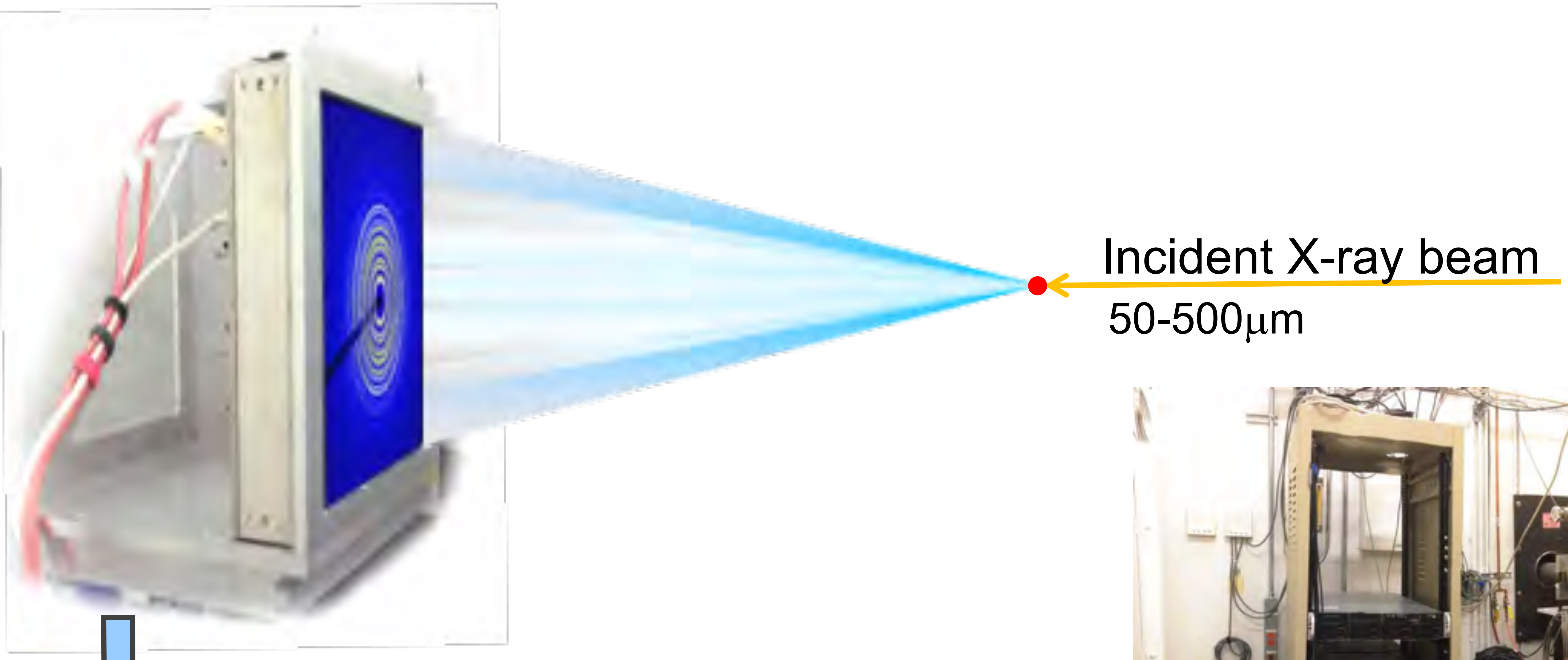
eScience 2017

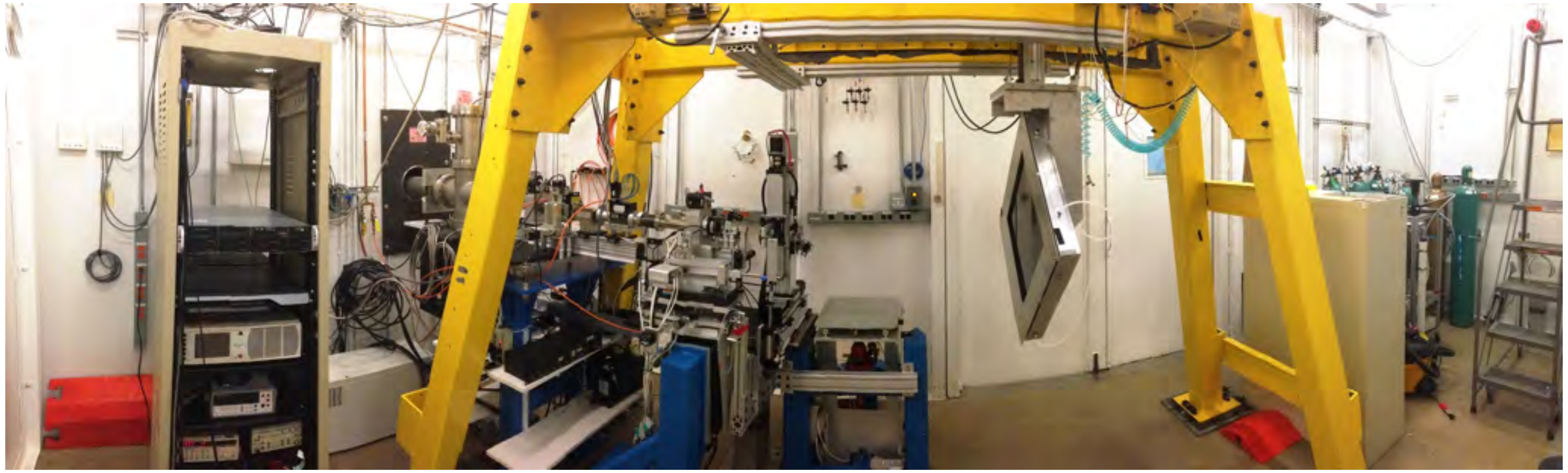


A use case for DOE **light source** facilities and **leadership computing** facilities.

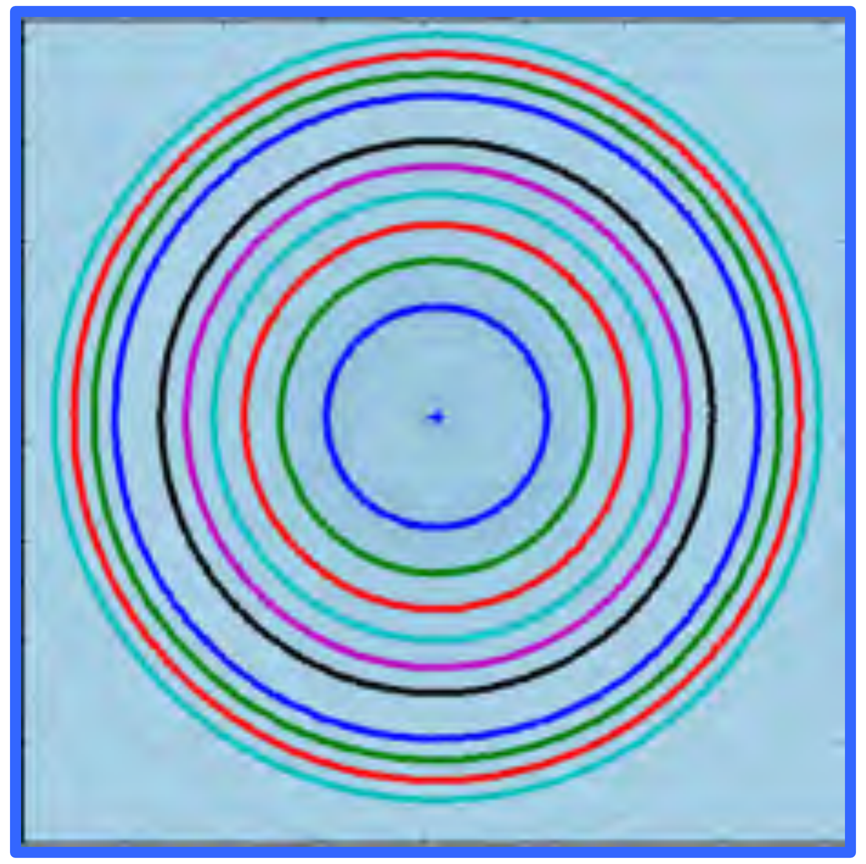


AutoMasking for Rapid Data Acquisition and Reduction (ongoing)

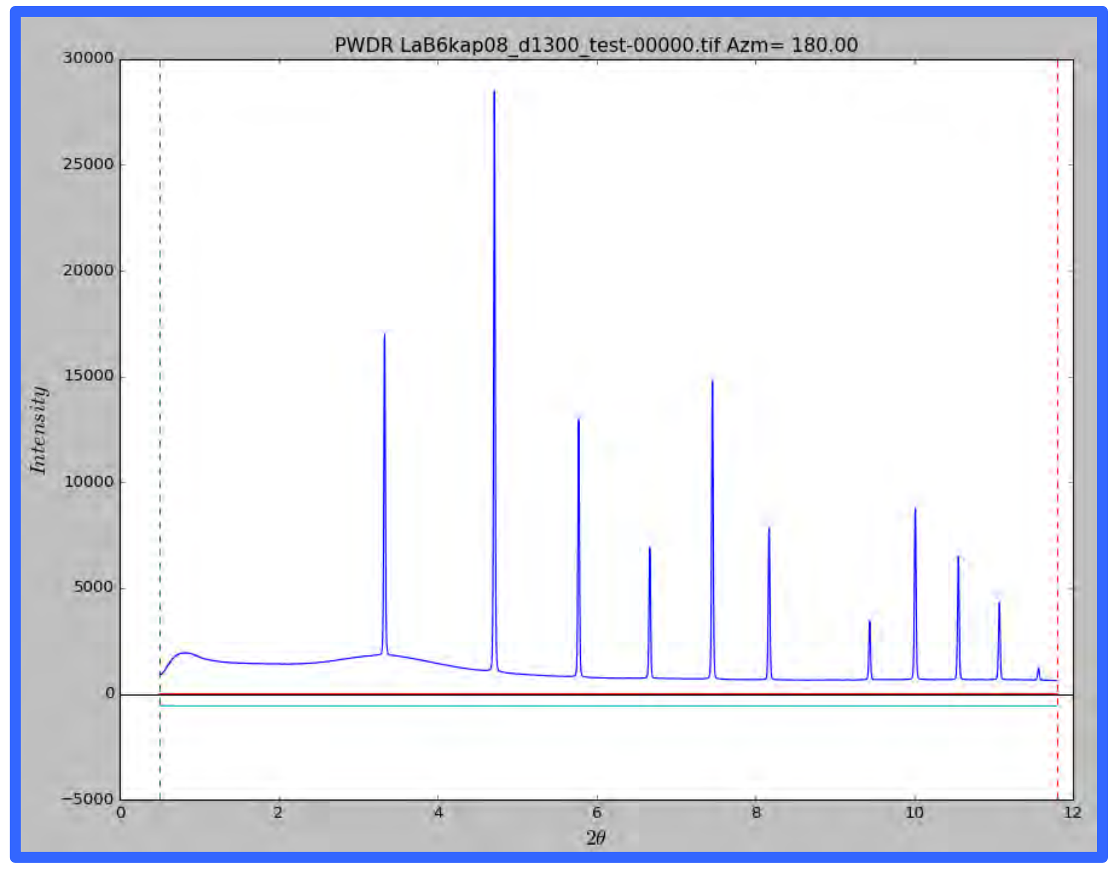




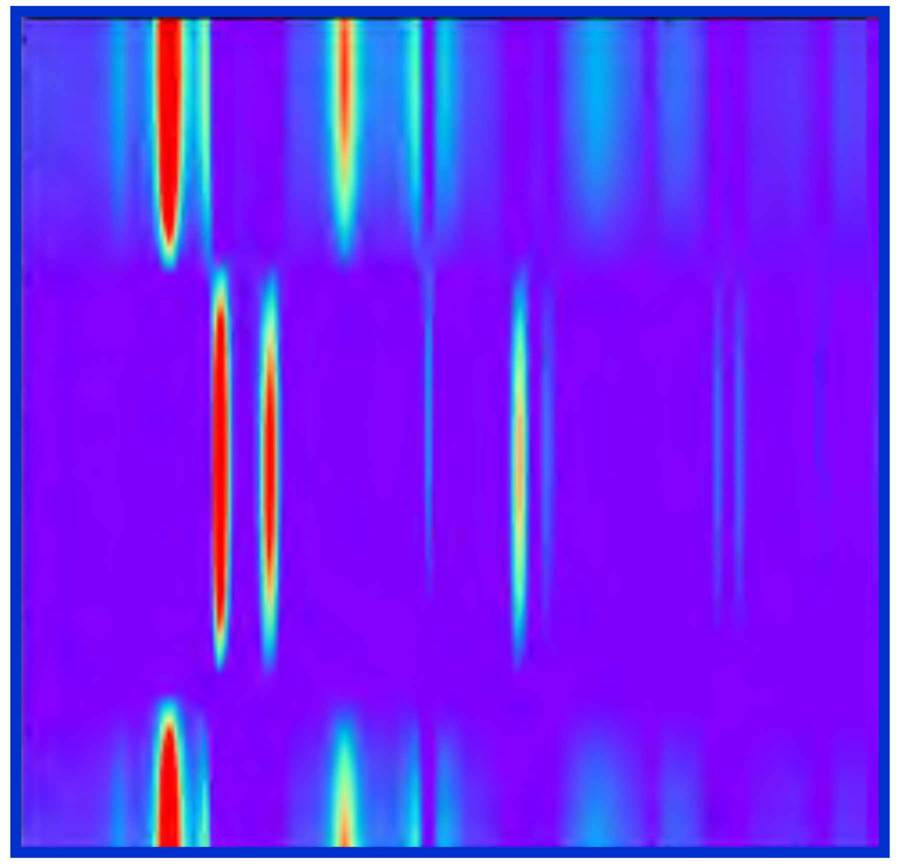
Calibration



Integration



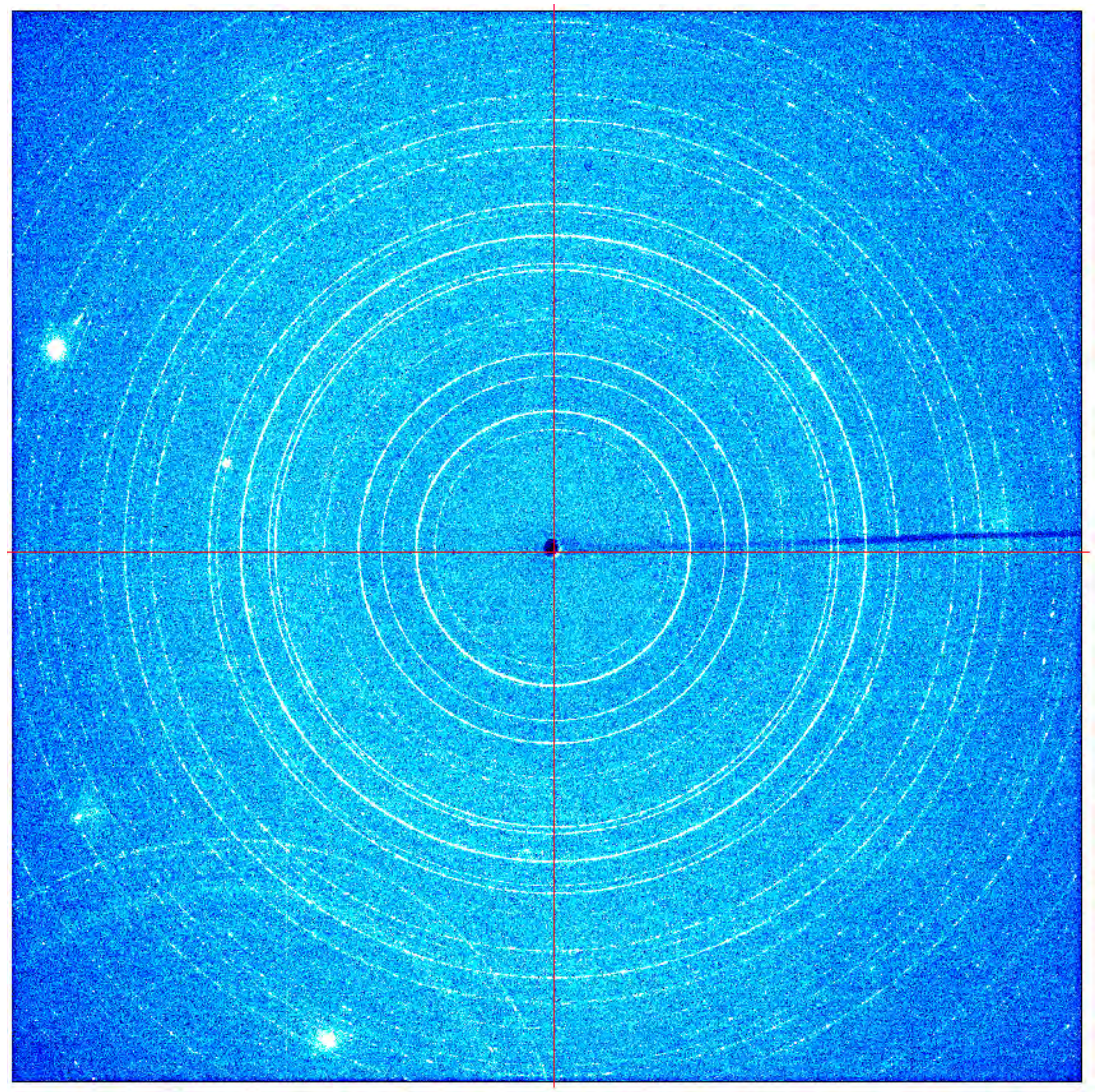
Top-View Plot



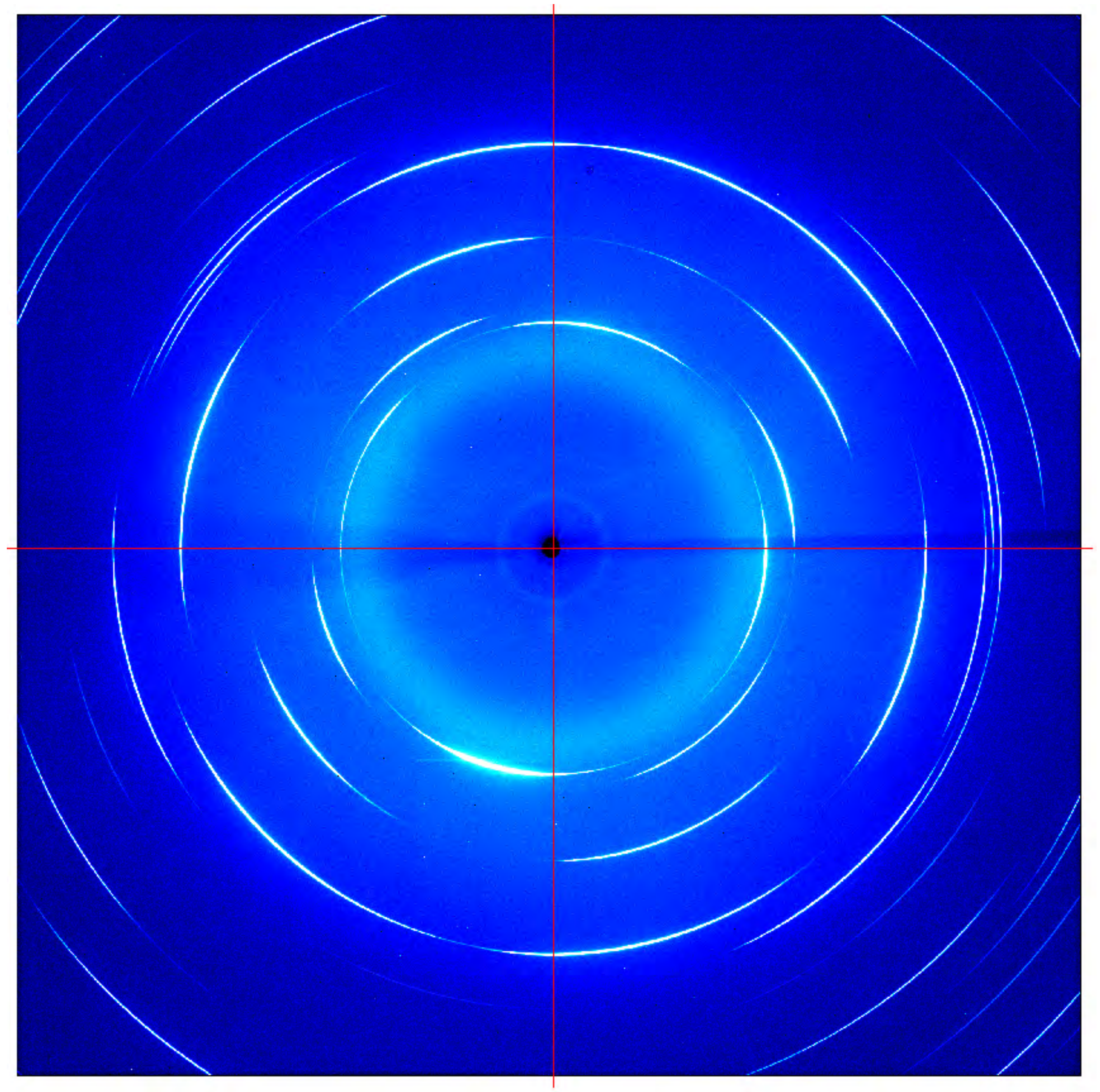
It needs to exclude certain areas on the 2D image from the integration process.

AutoMasking for XRD

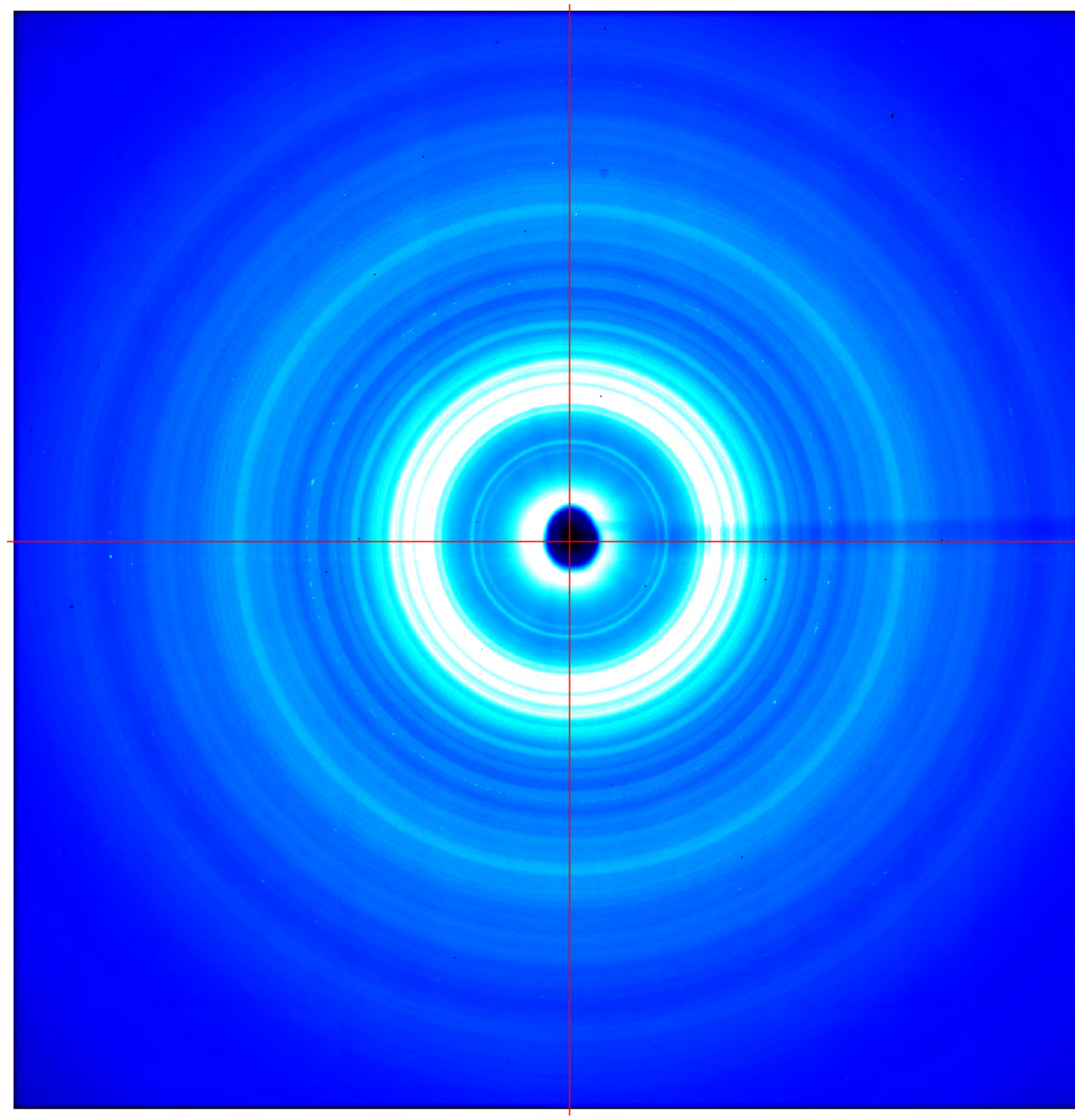
Not deterministic. It can grow, move and diminish across a data series.



Needs to remove



Needs to keep



Tiny but needs to remove

- Now, it is masked manually by experienced beamline scientists.
- Each experiment generates 100x - 1000x images.
- *The tool automatically save experts' manual masks to harvest training dataset, we then train models to learn from expert.*

Selected publications

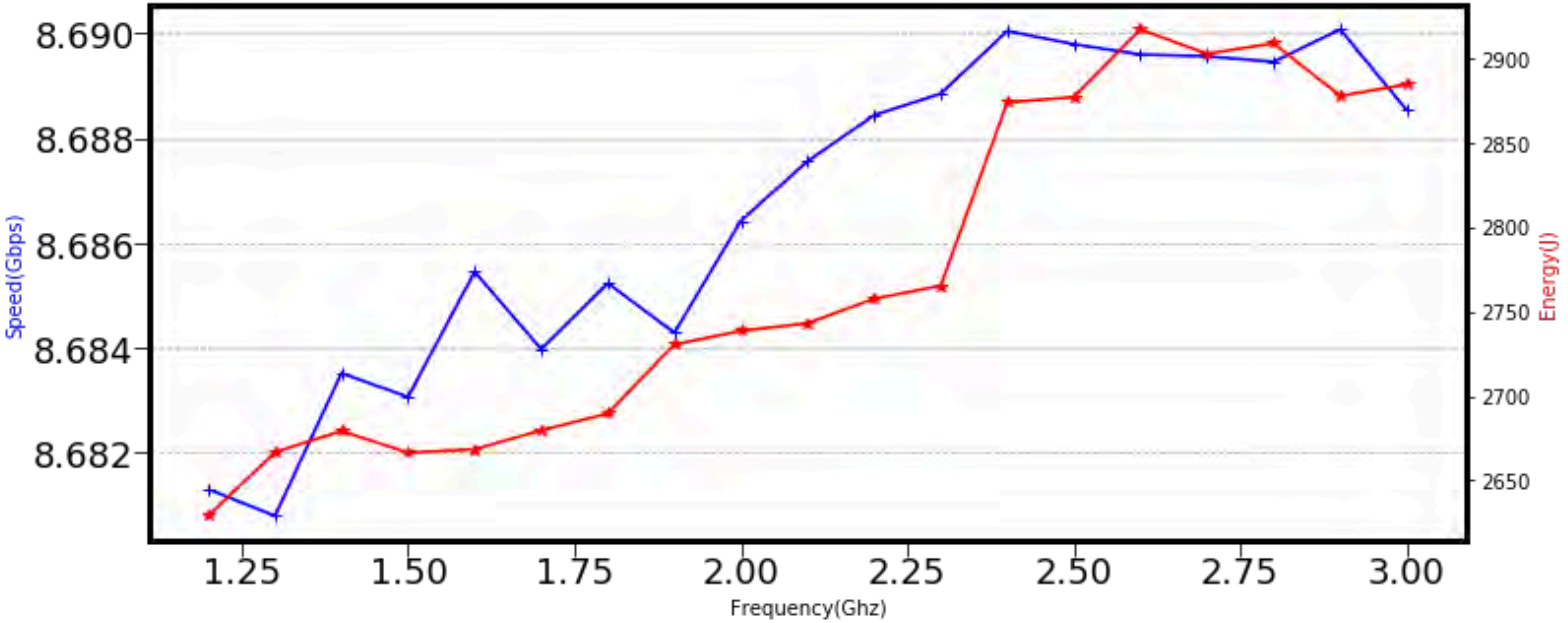
1) **Z. Liu**, T. Bicer, R. Kettimuthu, D. Gursoy, F. Carlo and I. Foster. TomoGAN: Low-Dose X-Ray Tomography with Generative Adversarial Networks. [arXiv:1902.07582].
2) Y. Liu, **Z. Liu**, R. Kettimuthu, N. Rao, Z. Chen and I. Foster. Data transfer between scientific facilities - bottleneck analysis, insights and optimizations. *CCGrid'19*.
3) **Z. Liu**, R. Kettimuthu, P. Balaprakash, N. Rao and I. Foster. Building a Wide-Area Data Transfer Performance Predictor: An Empirical Study. *MLN'18*.
4) R. Kettimuthu, **Z. Liu**, I. Foster, P. Beckman, A. Sim, J. Wu, W. Liao, Q. Kang, A. Agrawal, and A. Choudhary. Toward Autonomic Science Infrastructure: Architecture, Limitations, and Open Issues. *AI-Science@HPDC'18*.
5) **Z. Liu**, R. Kettimuthu, I. Foster and Y. Liu. A comprehensive study of wide area data movement at a scientific computing facility. *SNTA@ICDCS'18*.
6) **Z. Liu**, R. Kettimuthu, I. Foster and N. Rao. Cross-geography Scientific Data Transfer Trends and User Behavior Patterns. *HPDC'18*.
7) **Z. Liu**, R. Kettimuthu, I. Foster, P. Beckman. Towards a Smart Data Transfer Node. *FGCS, 2018(89)*.
8) R. Kettimuthu, **Z. Liu**, D. Wheeler, I. Foster, K. Heitmann, F. Cappello. Transferring a Petabyte in a Day. *FGCS, 2018(88)*.
9) **Z. Liu**, R. Kettimuthu, S. Leyffer, P. Palkar and I. Foster. A mathematical programming and simulation based framework to evaluate cyberinfrastructure design choices. *IEEE eScience'17*.
10) **Z. Liu**, P. Balaprakash, R. Kettimuthu and I. Foster. Explaining Wide Area Data Transfer Performance. *HPDC'17*.

Interests and Plans

Intelligent Infrastructure for Science
(e.g., Machine Learning for System, ML4Science, AI4Science)

Thanks!

Smart [for] Energy



we may save ~10% energy and will lose only 0.1% in performance