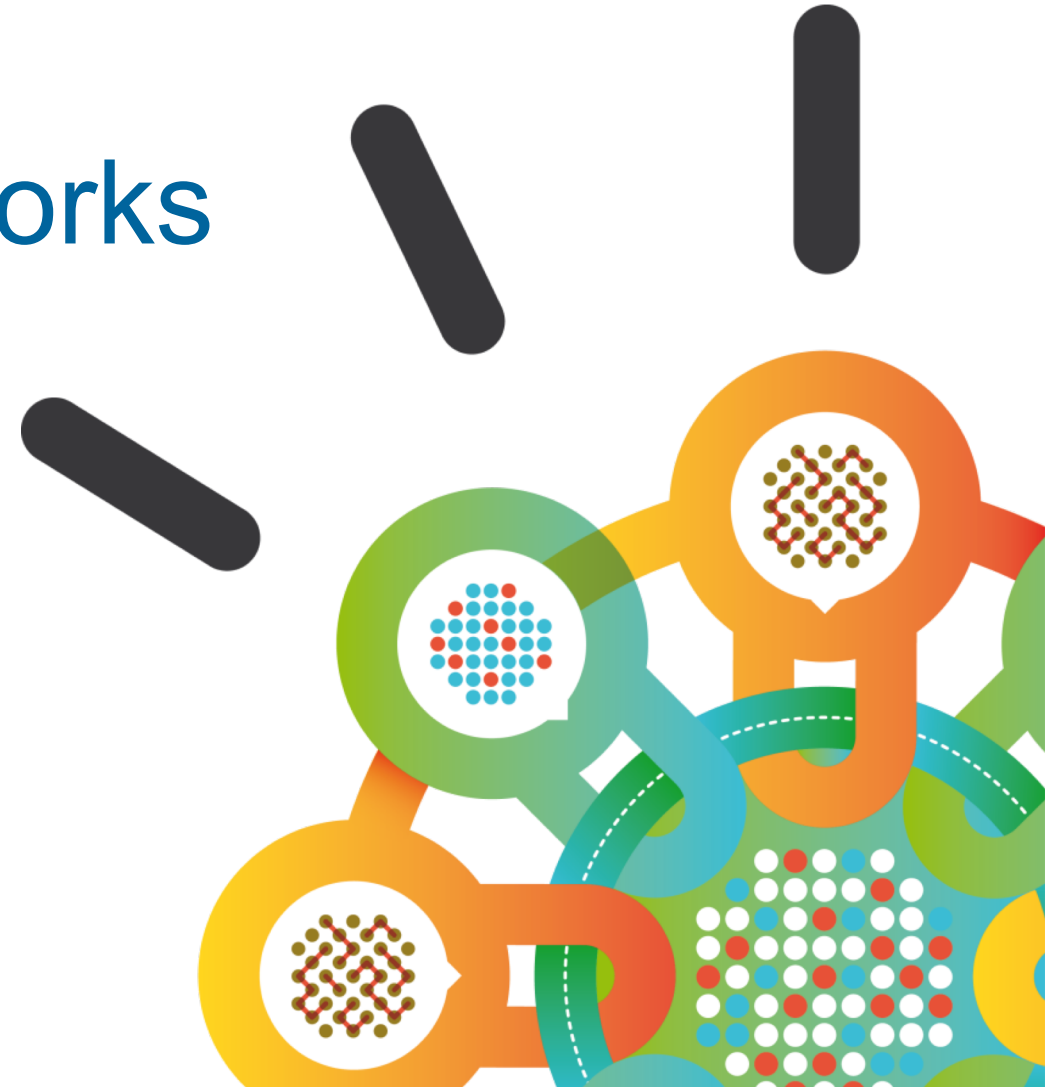


# Object Storage for Deep Learning Frameworks

Or Ozeri, **Effi Ofer**, Ronen Kat  
IBM Research – Haifa

Dec 2018

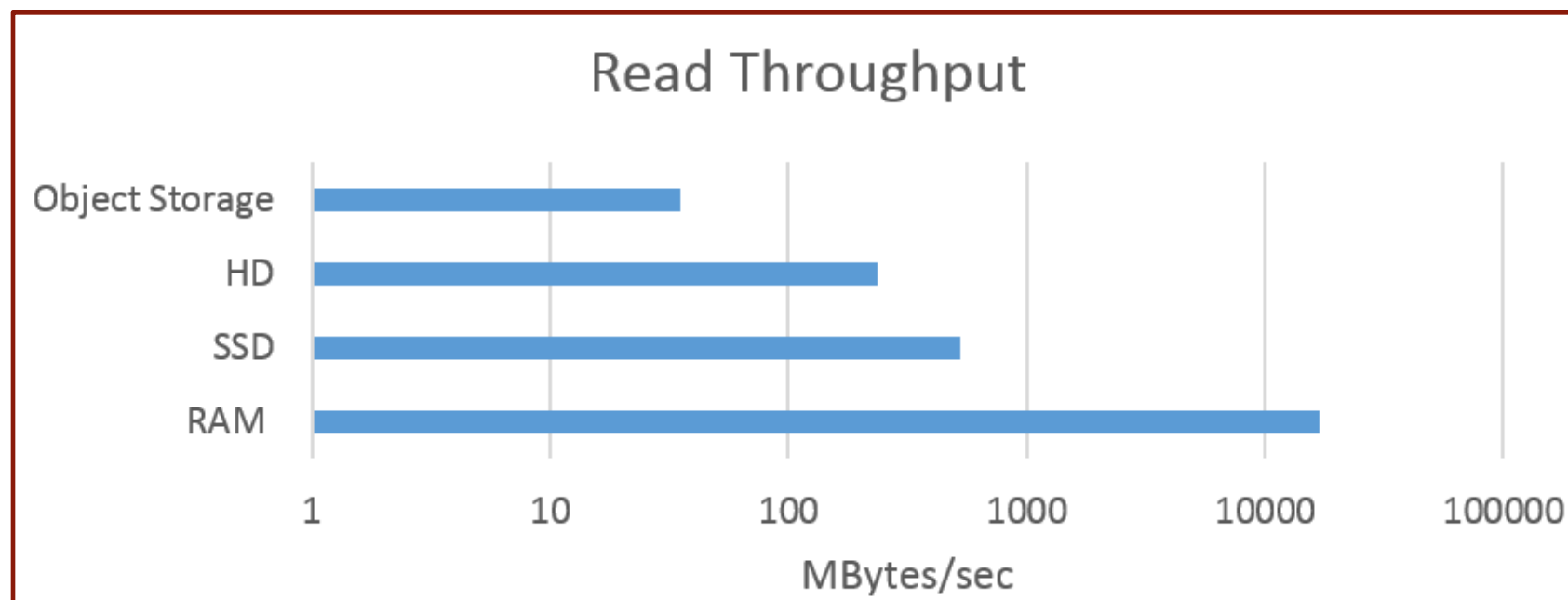


- Machine learning is quickly becoming the disruptive technology of the decade
  - Fueled by:
    - 1) The explosion in big data
    - 2) High-speed accelerators such as GPUs, FPGAs, and Tensor Processing Units
    - 3) Advancement of training algorithms and architectures



- Deep learning systems traditionally:
  - Use a POSIX file system interface
  - Keep the data locally on the same machine as the GPUs
- However, this model does not scale with the number of users, the volume of data, and the variety of workloads

- To handle all this big data, object storage has become a storage model of choice offering
  - disaggregated storage
  - high scalability
  - low cost
  - extremely high durability
- But its performance characteristics have traditionally been a barrier for analytics and machine learning workloads



- In this talk we explore how to run high throughput workloads on data that resides in inexpensive object storage without the need to pre-load or stage the data

# How Much Throughput is Good Enough?

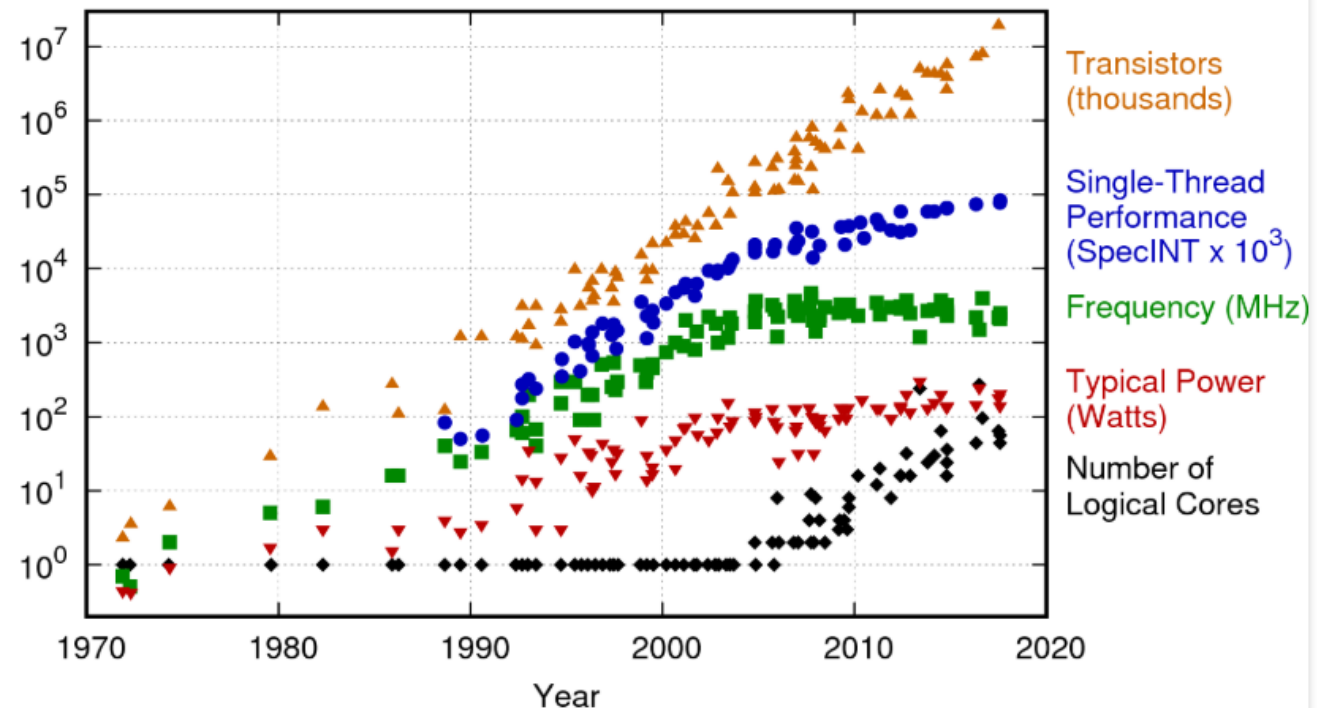
- Most of the heavy work in machine learning is done by the GPUs
- With some GPUs selling for over \$10K, they are often the most expensive component in a deep learning system
- We aim to provide throughput that is sufficient to keep the GPUs busy 100% of the time



- Storage Bandwidth (MB/s) of popular deep learning workloads running with varying number of Nvidia Volta V100 GPUs →

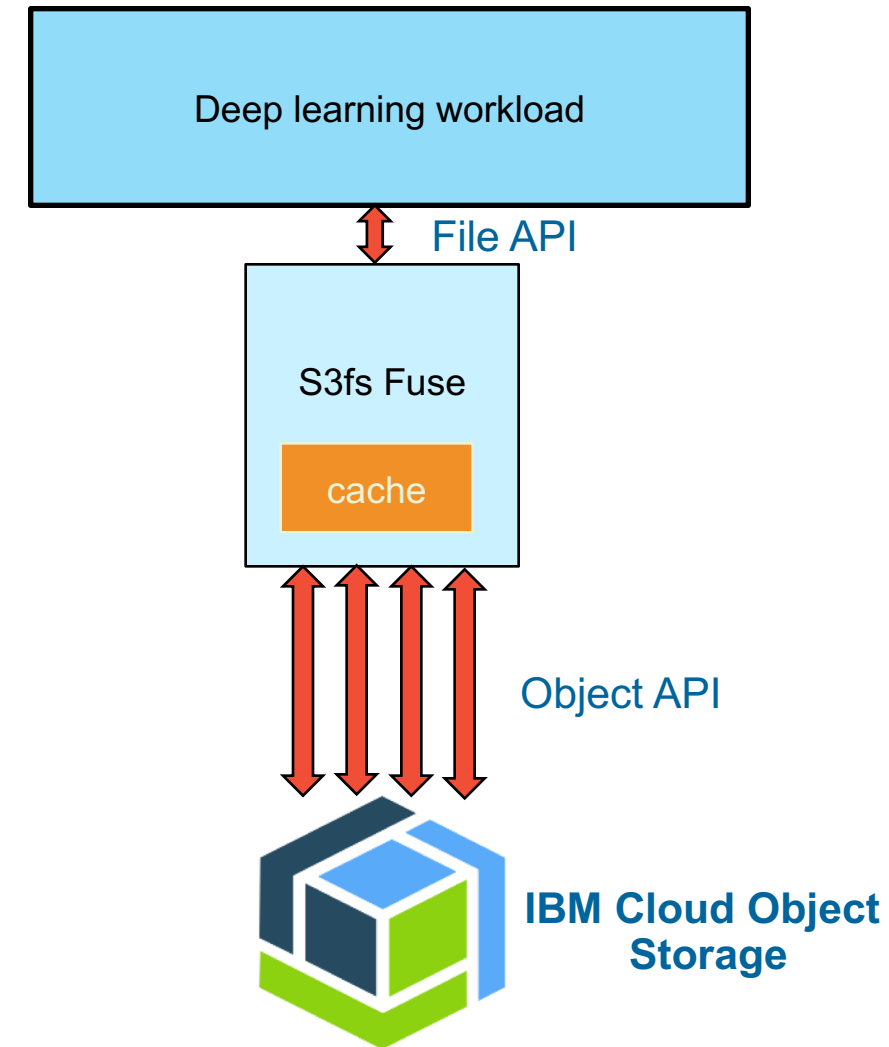
GPUs	Resnet152	Resnet50	VGG11	Alexnet	Speech LSTM
1	28.8	62.9	77.0	246.1	17.8
2	58.3	136.9	122.9	423.9	29.4
4	107.4	224.4	174.4	570.1	64.3
8	180.6	370.6	208.9	526.4	107.0

- Microprocessor trend data →
- CPUs are currently advancing at a rate of about 1.1x performance improvement per year
- GPUs are advancing at a rate of about 1.5x per year or 10x performance improvement every 5 to 6 years
- Can we expect machine learning workload to consume 5000MB/s+ in 2024?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten  
New plot and data collected for 2010-2017 by K. Rupp

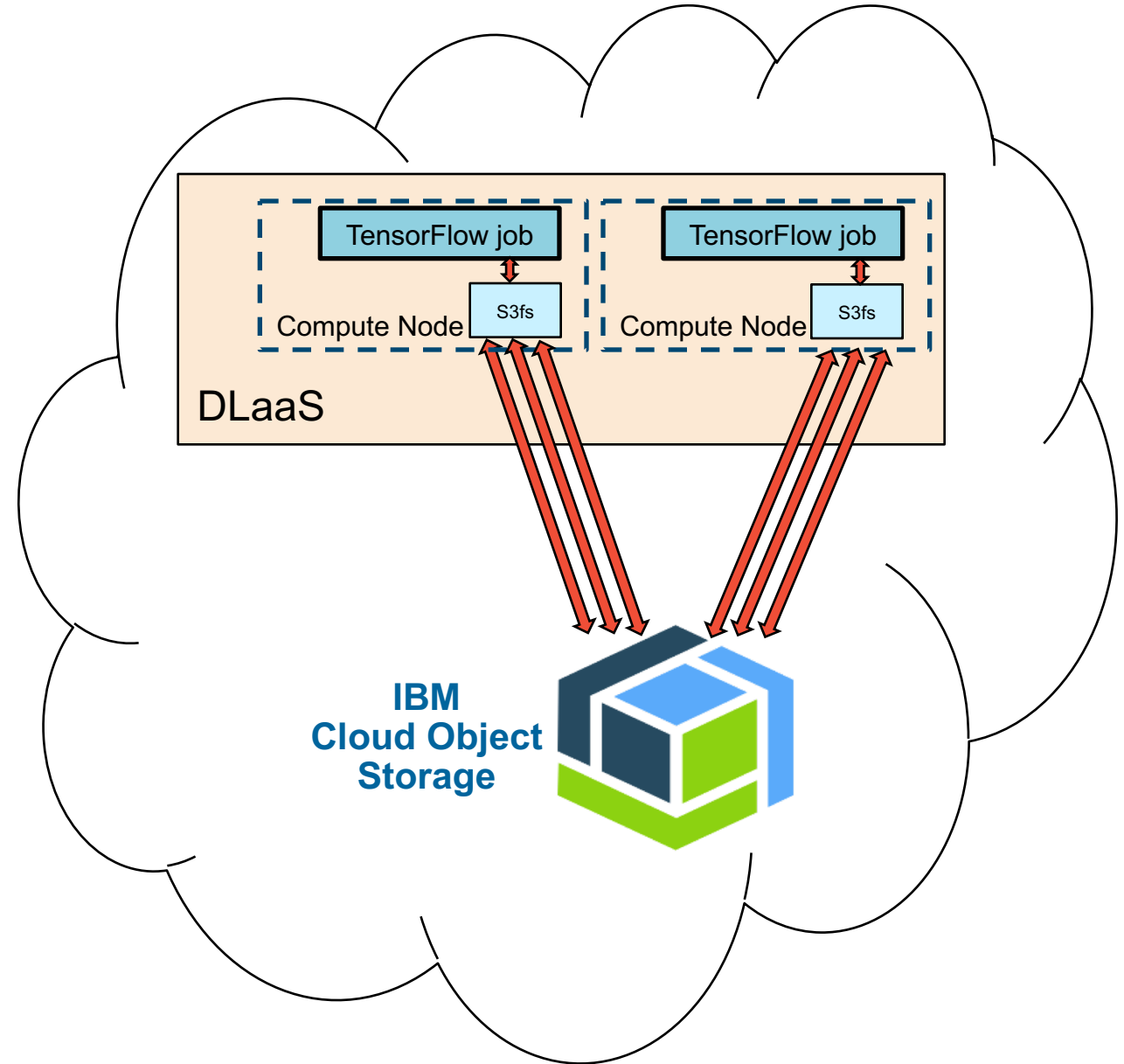
- Keep the data on inexpensive object storage
  - Training data
  - Results (logs + trained model)
- Support existing deep learning workloads without modifications
  - Enable deep learning workload to continue using POSIX interface
- Keep the GPUs busy
  - Supply data faster than object store single connection speeds
  - Support prefetching and optimize traffic
  - Leverage a local in-memory cache



# Supply Data Faster Than Object Store Single Connection Speed

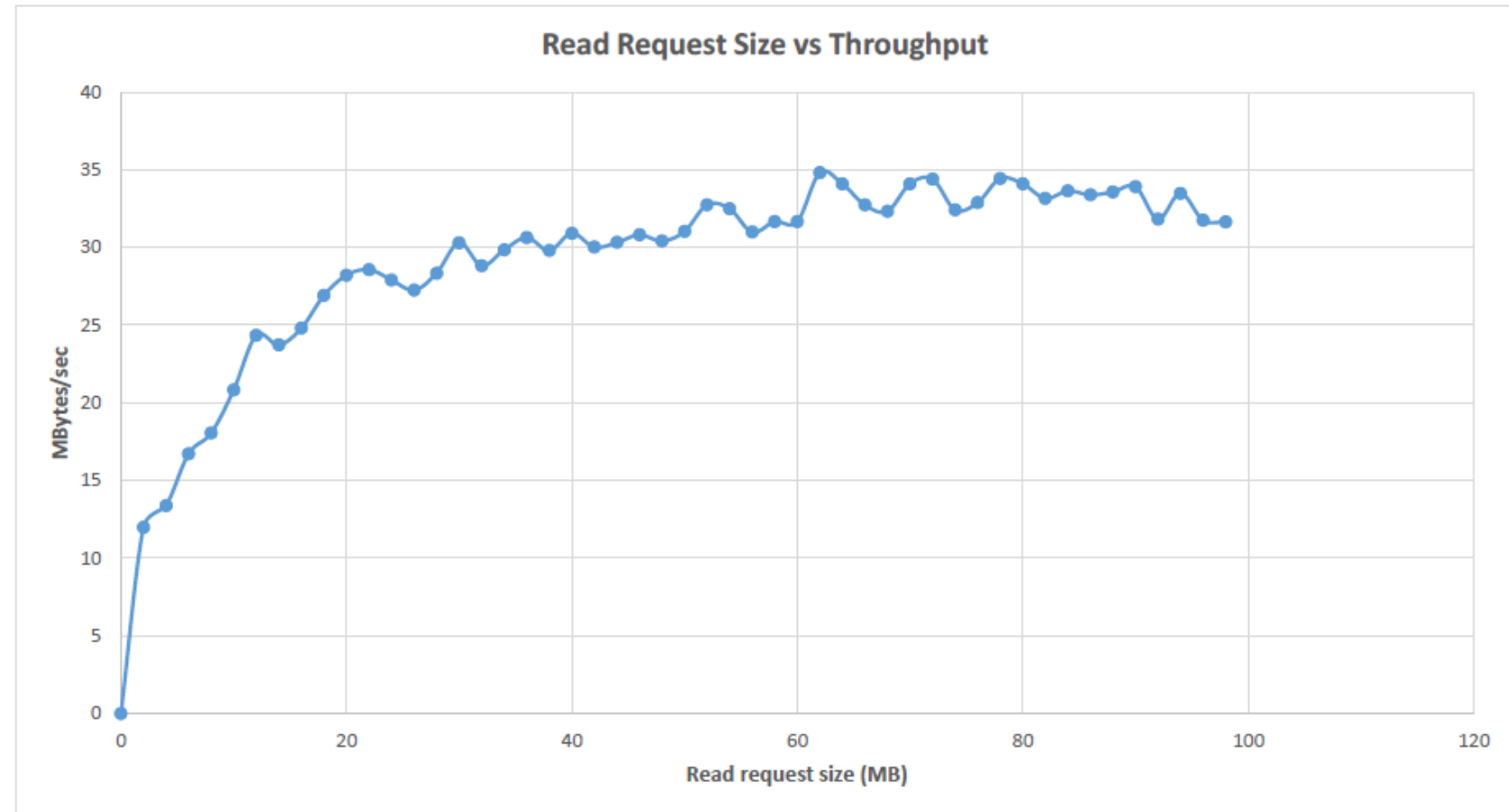


- Single threaded performance is well below the last network link bandwidth
- Driven by the multi user, multi tenant, distributed nature of object storage
- When a client requests multiple requests, an object storage can easily saturate his or her network
- We therefore convert read requests into multiple concurrent requests
- This allows us to sustain local disk like throughputs (550MBytes/sec) for single object read



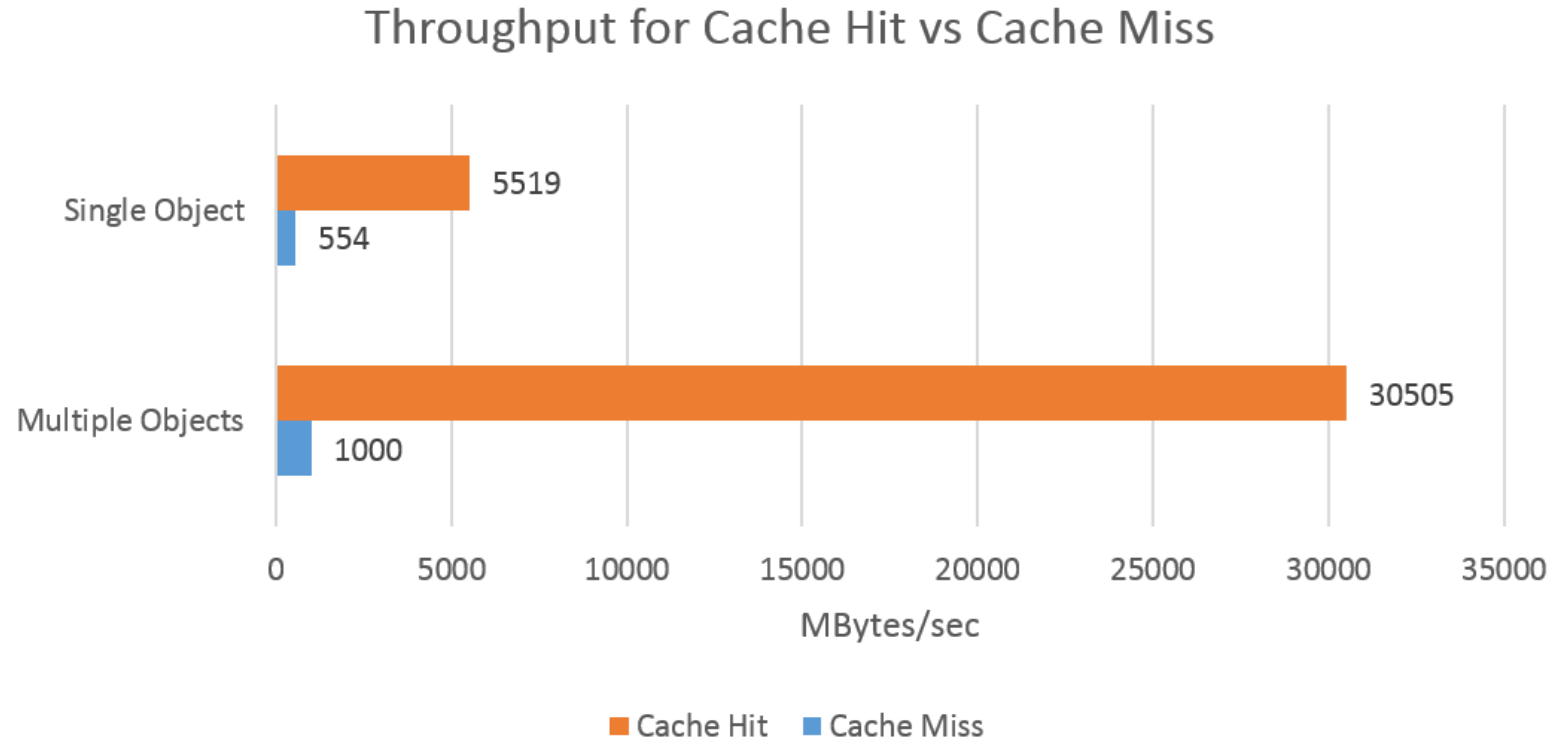


- We read data in chunks rather than read the precise amount that the client requests
- A larger chunk size enables a 'poor man's read ahead prefetching'
- A smaller chunk size enables additional concurrent reads
- We choose chunk size of 52 MB
- Why 52 MB?
  - Optimal chunk size for our environment is around 50MB
  - Our object storage best practices suggest using range read requests that are multiples of 4 MB



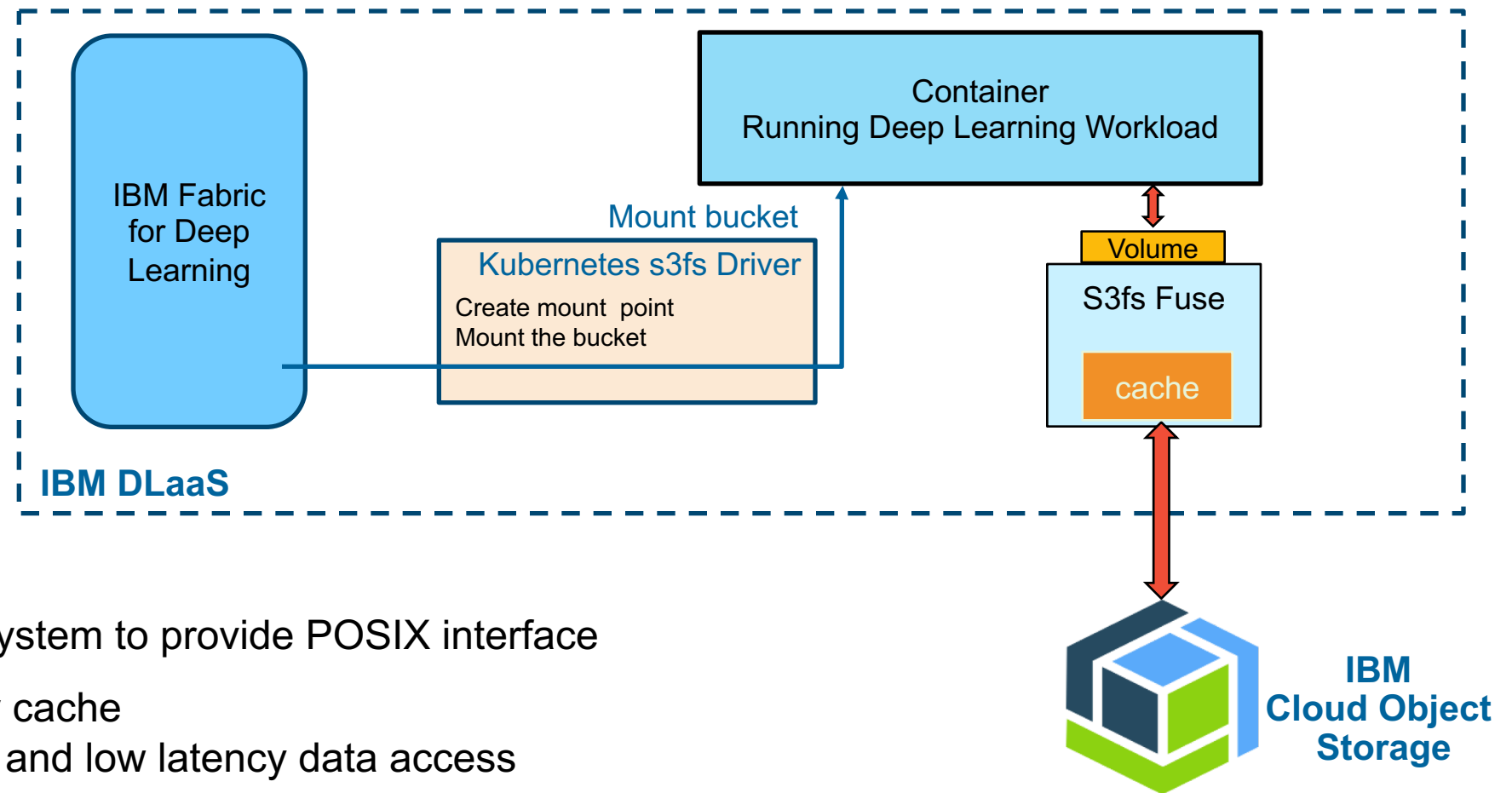


- We have deployed an in-memory cache using the Linux kernel page cache
- Collocated with the compute nodes
- Caching both
  - data chunks
  - meta data
- Limited by
  - client side network bandwidth
  - overheads from the FUSE fs
  - client side copy
- Since deep learning frameworks often run their training in multiple epochs this results in substantial speed improvement as long as the data fits within the cache



- Small objects suffer disproportionately from the overhead of the time-to-first-byte from object storage
- Best practices suggest combining multiple training records into a single data file
  - E.g., TFRecord in Tensorflow or HDF5 in pyTorch
  - Makes it easy to mix, match, and shuffle data sets
- But when data is packaged into larger objects, the read-ahead feature of the cache avoids this overhead

- Deployed in IBM Deep Learning as a Service (DLaaS) on the IBM Cloud
- Machine learning workload runs on Kubernetes containers
- Training data and trained model stored in IBM Cloud Object Storage
- Use s3fs FUSE based file system to provide POSIX interface
- Leverage a local in-memory cache
  - provide high throughput and low latency data access
- Supply data faster than object store single connection speeds



- The core of Deep Learning as a Service
  - A deep learning platform offering TensorFlow, Caffe, PyTorch etc. as a Service on Kubernetes

<https://github.com/ibm/ffd>

- Kubernetes volume plug-in that enables pods to access IBM Cloud Object Storage buckets.

Includes:

- A dynamic provisioner
- A driver for mounting the buckets using s3fs-fuse on a worker node

<https://github.com/IBM/ibmcloud-object-storage-plugin>

- S3fs enhancements have been contributed to the s3fs project repository

<https://github.com/s3fs-fuse/s3fs-fuse>



## Fabric for Deep Learning (FfDL)

