# Storage Server

# Many DL Workloads Dominated by I/O

- each GPU easily consumes 1 GByte/s
- 16 GPU machine = 16 GBytes/s in data I/O alone
- hardware
  - 150 MBytes/s rotation storage
  - 3 Gbytes/s SSD
  - 5 Gbytes/s Ethernet/Infiniband

# The YouTube8m Dataset

- 6.5 million videos
- 300 TB of data
- 1 PB when transcoded, normalized
- applications: unsupervised and semi-supervised learning
- assume pre-sharded into 100000 10G shards `chunks-{000000..099999}.tar`

# Datasets Larger than Local Storage

- 7-32 TB of local SSD storage
- need to replicate dadtaset across all nodes for local I/O
- alternatively, partition dataset across all nodes
- 64 GPUs w/DGX-1 = 8 nodes = 56-240 TB of local storage

Limits:

- largest replicated dataset: 32 TB
- largest partitioned dataset: 240 TB
- anything larger $\rightarrow$ distributed storage

# Distributed Storage

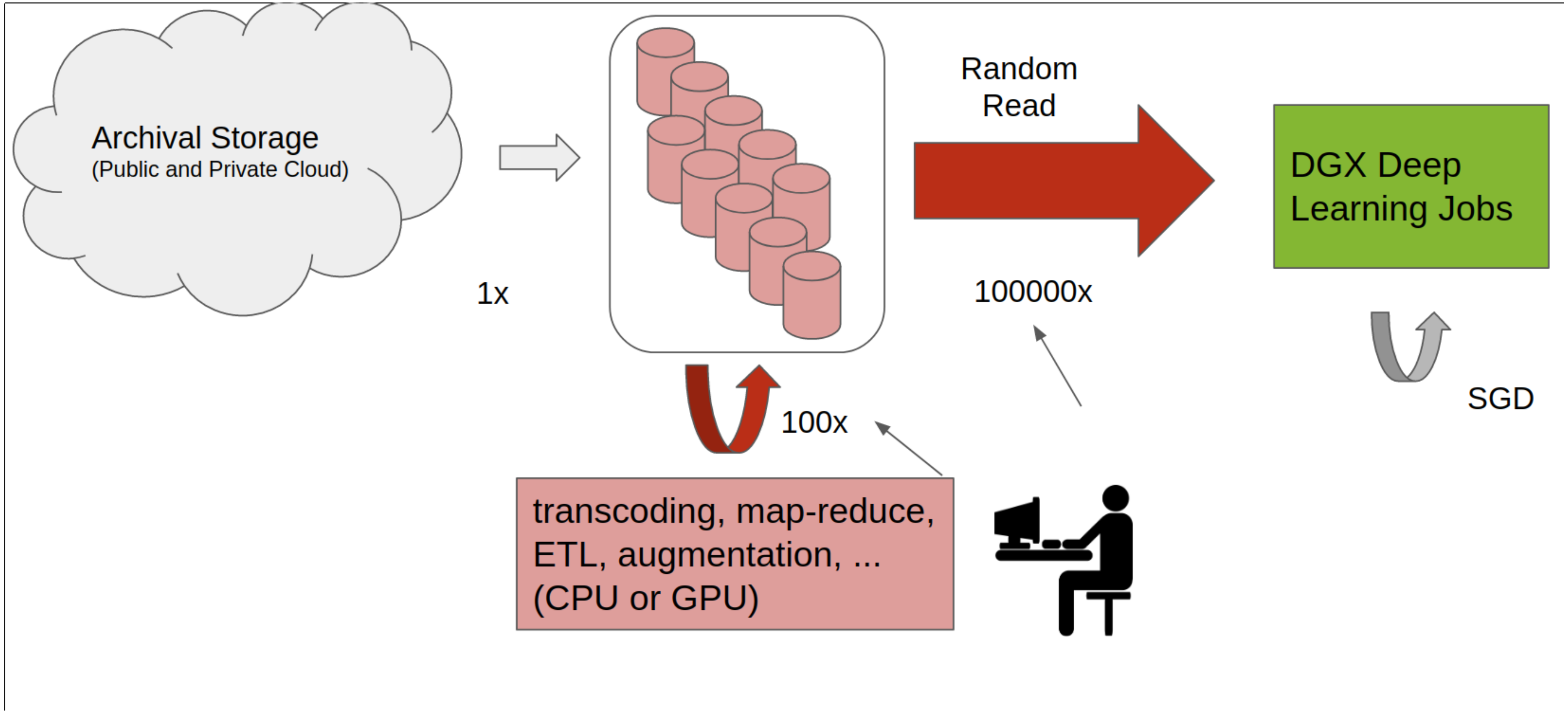
Architecture dependent on application. Classes of technologies:

- full POSIX semantics
  - fully consistent, read, write, seek, partial updates, etc.
  - local file systems
- partial POSIX semantics
  - behaves mostly like POSIX but not fully consistent
  - many network file systems (NFS, etc.)
- object store only
  - get object / put object only
  - object storage servers, web servers, cloud storage (S3 etc.)

# Other Considerations

- loss of drives vs loss of data, recovery
- loss of servers vs loss of service
- access patterns
- effectiveness of caching

# Typical Workflow




Two major operations: (1) ETL (2) training

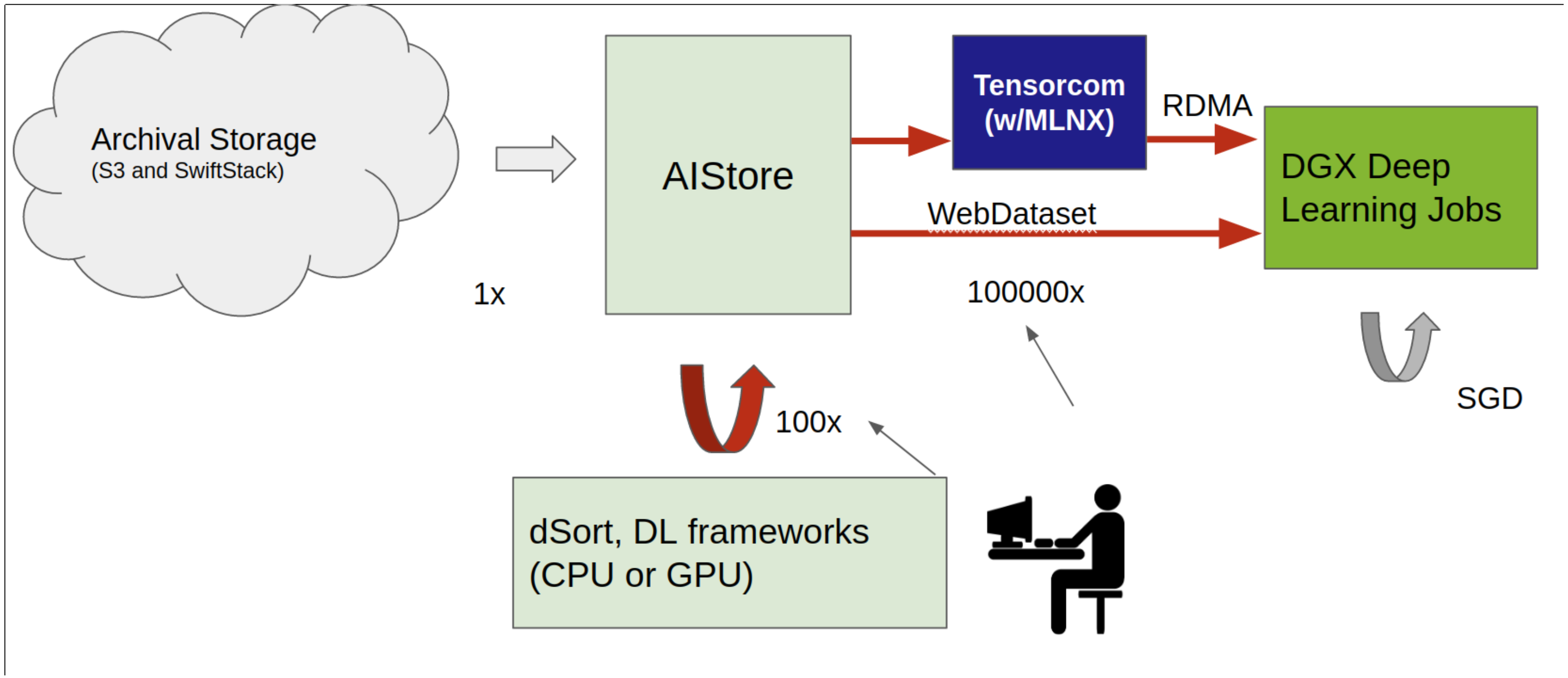
# Deep Learning on File System

- `FileDataSet` assumes POSIX semantics
- each training sample is a file, accessed in random order
- good performance on local SSD
- poor performance on rotational / network file systems
- ETL via file-by-file transformations on file system
- repeated iteration over full dataset makes caching pointless (no hotspots)
- LMDB reduces number of system calls but not random access

# Deep Learning with Sequential Reads

- requires only object store semantics (but works with POSIX)
- thousands of training samples are bundled into shards
- shuffling for training is two stage: shard shuffle and sample shuffle
- good performance on SSD, rotational, network file systems
- ETL via shard-to-shard transformations (only sequential reads)

# Collection of Technologies



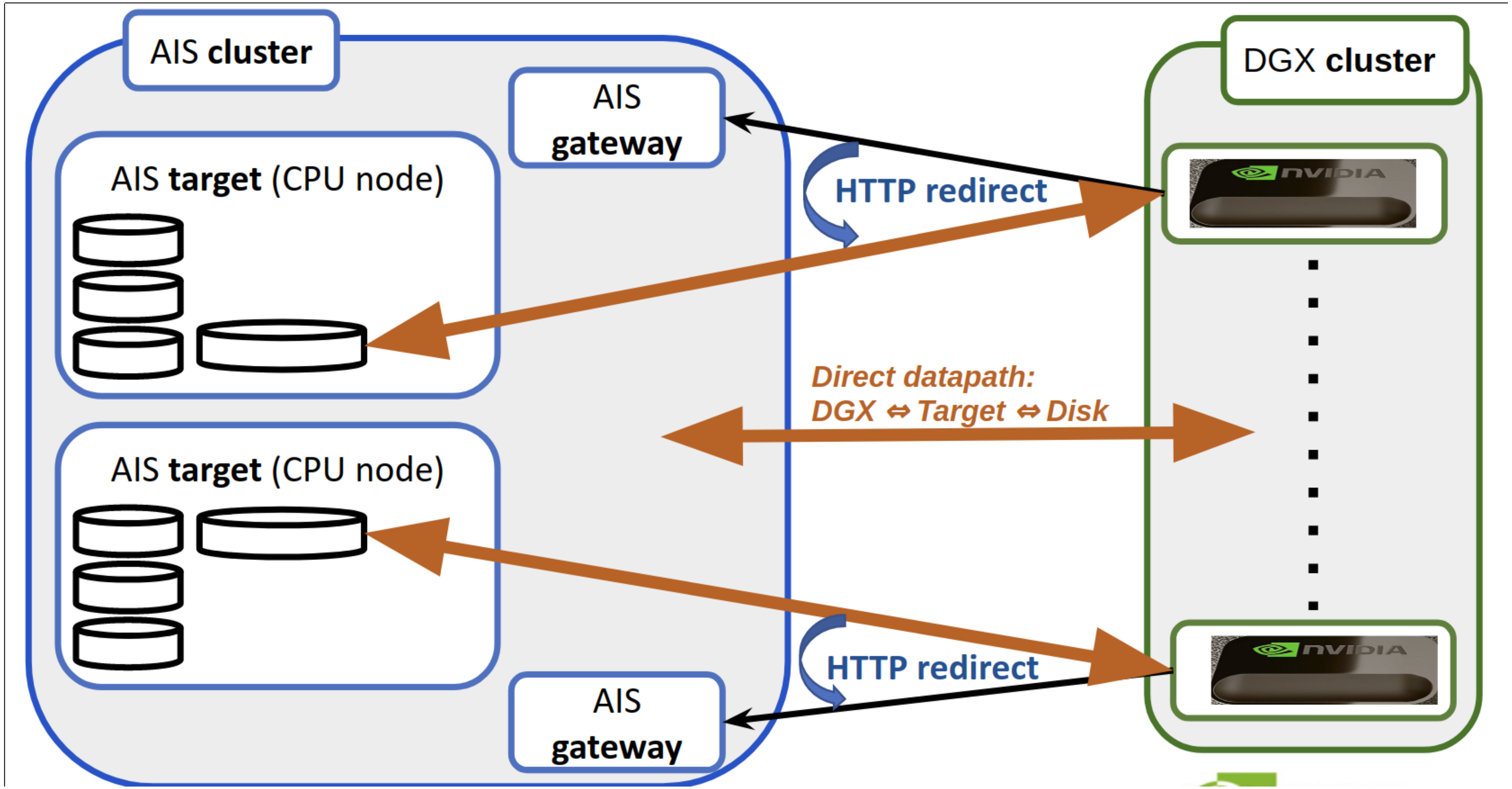

- proven technologies, re-implemented with DL focus and using standard formats
- use for production and/or use as didactic example to understand other systems
- TFRecord/tf.Example in principle used the same way (but little public infrastructure)

# AIStore

- provides an infinitely scalable object store for DL/AI applications
- uses HTTP(s) for all communications
  - avoidance of bottlenecks (using HTTP redirection protocol)
  - location transparency for DL jobs (using HTTP proxy protocol)
- can be deployed in Kubernetes
- provides additional server-side support for DL/AI applications (inference, sort, ...)
- even if you don't use it, illustrates important concepts in high performance storage

(We'll talk about the client libraries separately.)

# Avoiding Bottlenecks in Object Storage



# Handling Drive and Server Failures

Drive and server failures are inevitable with petascale datasets.

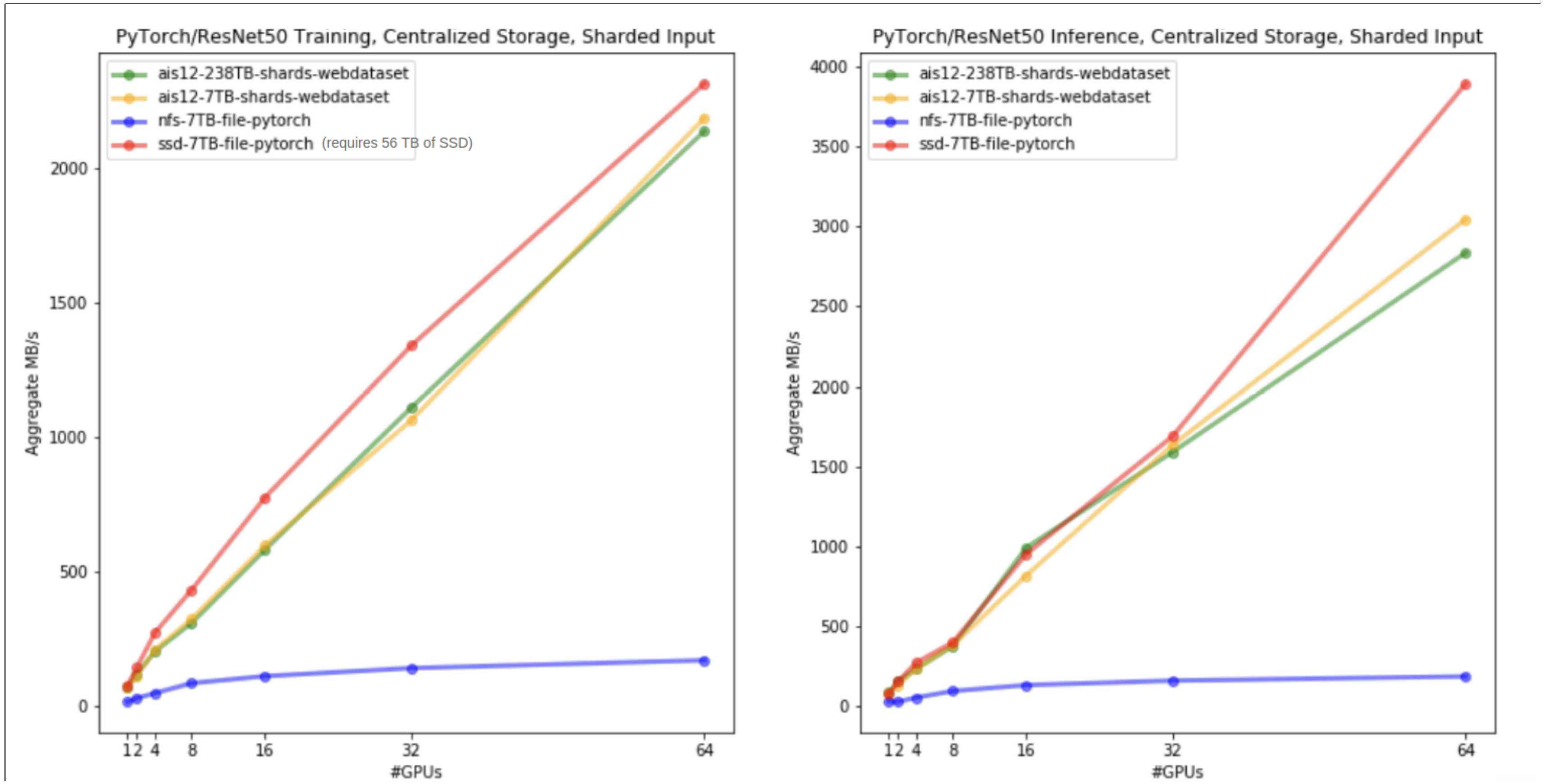
Two modes of recovery from drive failure:

- reconstruct via *erasure coding*
- retrieve from upstream storage (when used as a cache)

Continued operation in the presence of hardware failures:

- use *consistent hashing* to redirect to working server
- server initiates recovery from erasure coded parts or upstream server

# AIStore Performance




- AIStore achieves performance using rotational drives comparable to local SSD on large scale datasets
- AIStore can take maximum advantage of all available drive hardware bandwidth

# How do you use Sharded Distributed Storage?

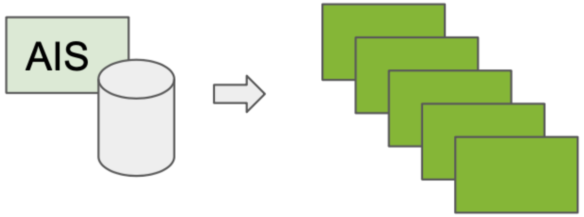
All you need is a URL:

```
samples = WebDataset("http://aistore/yt8m/chunks-{000000..099999}.tar", extensions="mjpeg json)
for clip, json in samples:
    ...
```

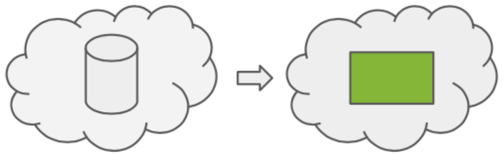
- common protocols ('http:', 'file:', 'gs:', 's3:', etc.)
- brace syntax enumerates the shards (just like in shell)
- works against any web server, cloud storage, local storage
- code is otherwise the same as in multi-node training with local replicated data
- performance is as good as local SSD

# WebDataset and AIStore "Scale Down" Nicely

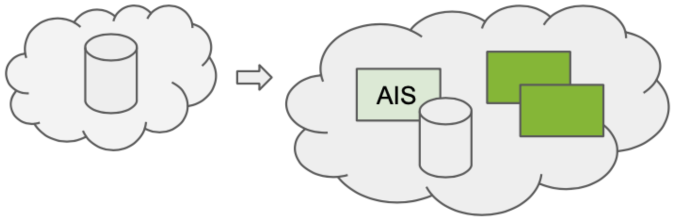
Student / Desktop Use




Small Work Groups / Academic Groups



Simple Cloud Usage




High Performance Cloud Usage

# AIStore and WebDataset

- illustrate important concepts in efficient large scale storage
- fully open source, open standards based solutions
- utilizes full I/O bandwidth of both rotational and SSD drives
- easy migration of existing apps & data
- unlimited scalability (both out and down)