In [ ]:

```
unset GZIP
```

Extract Transform Load, Map Reduce

ETL, MapReduce

- often data needs to be preprocessed prior to training
- common frameworks:
    - `TFRecord` + Google Map Reduce
    - Hadoop, Spark (Sequence files, Parquet, Avro)
- Issues
    - not K8s native
    - use serialized data structures rather than files

ETL/MapReduce based on Files

Basic approach:

- files + sequential storage = `tar` archives
- same format as `WebDataset`
- process each shard with `tarproc`

Big datasets:

- split a single dataset into multiple archives to enable more parallelism
- store in object stores (Swift, AIStore), cloud storage buckets (S3, Azure, Google)
- process shards in parallel with K8s Jobs

ImageNet Bucket

We have a bucket consisting of ImageNet shards that we want to perform format conversions on.

In [2]:

```
gsutil ls gs://lpr-imagenet | sed 10q
```

```
gs://lpr-imagenet/imagenet_train-0000.tgz
gs://lpr-imagenet/imagenet_train-0001.tgz
gs://lpr-imagenet/imagenet_train-0002.tgz
gs://lpr-imagenet/imagenet_train-0003.tgz
gs://lpr-imagenet/imagenet_train-0004.tgz
gs://lpr-imagenet/imagenet_train-0005.tgz
gs://lpr-imagenet/imagenet_train-0006.tgz
gs://lpr-imagenet/imagenet_train-0007.tgz
gs://lpr-imagenet/imagenet_train-0008.tgz
gs://lpr-imagenet/imagenet_train-0009.tgz
close failed in file object destructor:
sys.excepthook is missing
lost sys.stderr
```

In [3]:

```
gsutil ls gs://lpr-imagenet | sed 's/[0-9]/0/g' | sort -u
```

```
gs://lpr-imagenet/imagenet_train-0000.tgz
gs://lpr-imagenet/imagenet_val-0000.tgz
```

Shard Contents

- groups of `.cls`, `.jpg`, and `.json` files
- all files comprising a training sample have the same basename
- everything past the first dot: `.input.png`, `.output.png`

In [4]:

```
gsutil cat gs://lpr-imagenet/imagenet_train-0000.tgz | tar -ztvf - | sed 10q
```

```
-rw-rw-rw- bigdata/bigdata   3 2019-06-08 12:12 n03788365_17158.cls
-rw-rw-rw- bigdata/bigdata 75884 2019-06-08 12:12 n03788365_17158.jpg
-rw-rw-rw- bigdata/bigdata   382 2019-06-08 12:12 n03788365_17158.json
-rw-rw-rw- bigdata/bigdata     3 2019-06-08 12:12 n03000247_49831.cls
-rw-rw-rw- bigdata/bigdata 57068 2019-06-08 12:12 n03000247_49831.jpg
-rw-rw-rw- bigdata/bigdata   104 2019-06-08 12:12 n03000247_49831.json
-rw-rw-rw- bigdata/bigdata     3 2019-06-08 12:12 n03000247_22907.cls
-rw-rw-rw- bigdata/bigdata 97447 2019-06-08 12:12 n03000247_22907.jpg
-rw-rw-rw- bigdata/bigdata   450 2019-06-08 12:12 n03000247_22907.json
-rw-rw-rw- bigdata/bigdata     3 2019-06-08 12:12 n04597913_10741.cls
tar: write error
```

The `tarproc` Command Processes Tar Archives

- extract each group of files into a temporary directory
- run the command
- (optionally) package the result into a new tar file
- delete the temporary directory

In [6]:

```
gsutil cat gs://lpr-imagenet/imagenet_train-0000.tgz | tarproc -c ls --count 3
```

```
__key__  __source__  sample.cls  sample.jpg  sample.json
__key__  __source__  sample.cls  sample.jpg  sample.json
__key__  __source__  sample.cls  sample.jpg  sample.json
```

The `__key__` Identifies Samples

- extracted files always use `sample` as their basename
- the `__key__` file contains the basename of the file inside the archive

In [7]:

```
gsutil cat gs://lpr-imagenet/imagenet_train-0000.tgz | tarproc -c 'cat __key__; echo' --count 3
```

```
n03788365_17158
n03000247_49831
n03000247_22907
```

Example: Large Scale Image Format Conversion

- convert each `.jpg` image to `.ppm` (using ImageMagick)
- store result in a new `.tar` file

In [8]:

```
gsutil cat gs://lpr-imagenet/imagenet_train-0000.tgz |
tarproc -c '
    convert sample.jpg sample.ppm
    rm sample.jpg
' --count 3 -o out.tar
```

In [9]:

```
tar tvf out.tar
```

```
-r--r--r-- bigdata/bigdata   3 2019-12-09 15:24 n03788365_17158.cls
-r--r--r-- bigdata/bigdata 382 2019-12-09 15:24 n03788365_17158.json
-r--r--r-- bigdata/bigdata 478515 2019-12-09 15:24 n03788365_17158.ppm
-r--r--r-- bigdata/bigdata      3 2019-12-09 15:24 n03000247_49831.cls
-r--r--r-- bigdata/bigdata    104 2019-12-09 15:24 n03000247_49831.json
-r--r--r-- bigdata/bigdata 499515 2019-12-09 15:24 n03000247_49831.ppm
-r--r--r-- bigdata/bigdata      3 2019-12-09 15:24 n03000247_22907.cls
-r--r--r-- bigdata/bigdata    450 2019-12-09 15:24 n03000247_22907.json
-r--r--r-- bigdata/bigdata 562515 2019-12-09 15:24 n03000247_22907.ppm
```

`tarproc` supports Multicore Processing

In [10]:

```
gsutil -m cp gs://lpr-imagenet/imagenet_train-0000.tgz imagenet_train-0000.tgz
```

```
Copying gs://lpr-imagenet/imagenet_train-0000.tgz...
/ [1/1 files][946.3 MiB/946.3 MiB] 100% Done  82.4 MiB/s ETA 00:00:00
Operation completed over 1 objects/946.3 MiB.
```

In [11]:

```
time /bin/bash -c '
cat imagenet_train-0000.tgz | tarproc -c "convert sample.jpg sample.ppm; rm sample.jpg" -o out.tar
'
```

```
real	3m19.963s
user	2m9.339s
sys	1m9.149s
```

To enable multicore processing, just use the `-p` flag.

In [12]:

```
time /bin/bash -c '
cat imagenet_train-0000.tgz | tarproc -p 8 -c "convert sample.jpg sample.ppm; rm sample.jpg" -o out.tar
'
```

```
real	0m52.049s
user	2m4.979s
sys	1m13.184s
```

Processing Using `WebDataset`

In [13]:

```
cat > mapper.py <<'EOF'
import sys, argparse
from webdataset.dataset import WebDataset
from webdataset.writer import TarWriter

parser = argparse.ArgumentParser("convert jpg to png in imagenet-style databases")
parser.add_argument("--count", type=int, default=9999999999)
parser.add_argument("input"); parser.add_argument("output")
args = parser.parse_args()

sink = TarWriter(args.output)
for i, (key, image, cls) in enumerate(WebDataset(args.input, extensions="__key__ jpg cls")):
    if i%1000==0: print(i, key, file=sys.stderr)
    if i>=args.count: break
    sink.write(dict(__key__=key, ppm=image[::2,::2,:], cls=cls ))
sink.close()
EOF
```

Running the Job

You have the full power of PyTorch available, including GPU processing.

In [14]:

```
python3 mapper.py --count 2000 imagenet_train-0000.tgz out1.tar
```

```
0 n03788365_17158
1000 n02965783_5956
2000 n10565667_409
```

In [15]:

```
cat > kubetpl.yaml <<'EOF'
image: gcr.io/research-191823/bigdata19
memory: 4G
cpu: 1
app: bigdata19
port:
  - 7880
EOF
```

In [16]:

```
kubectl delete job/job0000 || true
kubetpl job -n job0000 -c '
curl -s http://storage.googleapis.com/lpr-imagenet/imagenet_train-0000.tgz |
tarproc --count 3 -c "ls -l"
' | kubectl apply -f -
```

```
Error from server (NotFound): jobs.batch "job0000" not found
job.batch/job0000 created
```

In [17]:

```
sleep 15
```

In [18]:

```
kubectl logs job/job0000
```

```
total 92
-rw-r--r-- 1 root root    15 Dec  9 23:29 __key__
-rw-r--r-- 1 root root     1 Dec  9 23:29 __source__
-rw-r--r-- 1 root root     3 Dec  9 23:29 sample.cls
-rw-r--r-- 1 root root 75884 Dec  9 23:29 sample.jpg
-rw-r--r-- 1 root root   382 Dec  9 23:29 sample.json
total 72
-rw-r--r-- 1 root root    15 Dec  9 23:29 __key__
-rw-r--r-- 1 root root     1 Dec  9 23:29 __source__
-rw-r--r-- 1 root root     3 Dec  9 23:29 sample.cls
-rw-r--r-- 1 root root 57068 Dec  9 23:29 sample.jpg
-rw-r--r-- 1 root root   104 Dec  9 23:29 sample.json
total 112
-rw-r--r-- 1 root root    15 Dec  9 23:29 __key__
-rw-r--r-- 1 root root     1 Dec  9 23:29 __source__
-rw-r--r-- 1 root root     3 Dec  9 23:29 sample.cls
-rw-r--r-- 1 root root 97447 Dec  9 23:29 sample.jpg
-rw-r--r-- 1 root root   450 Dec  9 23:29 sample.json
```

In [19]:

```
kubectl delete job/job0000 || true
kubetpl job -n job0000 -c '
curl -s http://storage.googleapis.com/lpr-imagenet/imagenet_train-0000.tgz |
tarproc --count 3 -c "gm convert sample.jpg sample.ppm; rm sample.jpg" -o - |
dd of=/dev/null # put a copy-to-destination here
' | kubectl apply -f -
```

```
job.batch "job0000" deleted
job.batch/job0000 created
```

In [20]:

```
sleep 15
```

In [21]:

```
kubectl logs job/job0000
```

```
3040+0 records in
3040+0 records out
1556480 bytes (1.6 MB, 1.5 MiB) copied, 0.679939 s, 2.3 MB/s
```

In [22]:

```
kubectl delete jobs --all
kubectl delete pods --all
```

```
job.batch "job0000" deleted
No resources found
```

Processing All Shards in Parallel on K8s

In [23]:

```
for i in {0000..0147}; do
kubetpl job -n job$i -c "
curl -s http://storage.googleapis.com/lpr-imagenet/imagenet_train-$i.tgz |
tarproc -c 'gm convert sample.jpg sample.ppm; rm sample.jpg' -o - |
dd of=/dev/null # put a copy-to-destination here
" | kubectl apply -f - >> _log
done
tail _log
```

```
job.batch/job0138 created
job.batch/job0139 created
job.batch/job0140 created
job.batch/job0141 created
job.batch/job0142 created
job.batch/job0143 created
job.batch/job0144 created
job.batch/job0145 created
job.batch/job0146 created
job.batch/job0147 created
```

In [24]:

```
sleep 60
```

In [25]:

```
kubectl get pods | sed 1d | awk '{print $3}' | sort | uniq -c
```

```
    148 Running
```

In [26]:

```
kubectl delete jobs --all >> _log
kubectl delete pods --all >> _log
```

In [27]:

```
tail _log
```

```
pod "job0137-wdshf" deleted
pod "job0138-b6qmr" deleted
pod "job0139-fzhjj" deleted
pod "job0140-k6xhz" deleted
pod "job0141-wvkcz" deleted
pod "job0142-5cw7k" deleted
pod "job0143-4vd5k" deleted
pod "job0144-wnrcs" deleted
pod "job0145-lwmwd" deleted
pod "job0146-xqb45" deleted
```

ETL, Map Reduce

- DL and AI require large scale data preprocessing
- we use sequential file formats and sharding to make this efficient
- several platforms to choose from: Hadoop, Spark
- DL/AI data is usually file based, so examples use POSIX archives
- parallelization using K8s Jobs