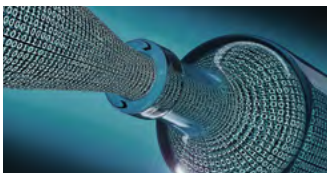
AWS IMPROVES WRITE PERFORMANCE BY 4X WITH MISTRAL I/O PROFILING

An AWS customer wanted to scale their machine learning (ML) workload to hundreds of thousands of machine instances. Their goal was to download large images, including people and cars, from S3 to EBS storage to process for training a self-driving car. Optimizing and scaling storage usage was key, but a bottleneck was created when writing the images to disc. The AWS team profiled the application using Altair Mistral™ to see how the workflow could be improved, with results that were well worth the effort.

The Workflow

Cloud storage users commonly encounter I/O problems as they switch between types of storage and upload/download from various storage tiers. I/O profiling using tools such as Altair Breeze™ and Mistral is becoming even more vital as big data gets bigger and more and more companies turn to platforms such as AWS for compute power. There are many factors to consider for cloud workloads, including the bursty/non-bursty nature of I/O and the choices that must made between EBS volume types — gp2 does well for 80% of cases, but using sc1 and st1 is cheaper. In this case, the customer had opted for the latter. The application under scrutiny was an ML workload that processes images for training a self-driving car. It is a highly parallel workload that the customer wanted to scale to hundreds of thousands of machine instances on AWS.



Understanding I/O bottlenecks is critical for performance

Training data are the building blocks of machine learning

The flow is as follows:

- Job submitted to batch
- Batch submit the job to the ECS task, one task per core on each machine instance
- Each process pulls the training image from S3 object storage onto an EBS volume, which takes several minutes
- Process the image once it has been downloaded, which takes 45 minutes
- Write output file to S3
- End task

This process was taking far longer than desired, which is where Mistral came into action.

File Caching

The element of the workflow that the team analyzed was the download stage. This was optimized using Mistral to tune the file cache settings. File caching is an important performance improvement, but write caching can be tricky. Data can be lost before it is written out asynchronously so caching critical data is undesirable, but a lot of I/O will overwhelm the cache if the settings are too low, forcing the system to block each process while it writes data out to disk, skipping the cache.

vm.dirty_background_ratio is the percentage of system memory that can be filled with memory pages that still need to be written to disk (dirty pages) before the background processes start to write it to disk.

vm.dirty_ratio is the absolute maximum percentage of system memory that can be filled with dirty pages. When this limit is reached the system blocks and writes all the dirty pages to disk.

Both limits can be increased to boost performance if it doesn't matter if you lose data in a system outage. In this case, a failed job could simply be restarted with the input data still safely stored in S3, so reliability is not an issue. Any application where the data can fit in memory could benefit from increasing these values.

Mistral helped the team optimize data download performance

Download time was reduced by 4X, from 12 minutes to less than three

Optimizing File Caching Using Mistral

Mistral was used to measure the mean I/O latency for writes to EBS while pulling data down from S3 in the first stage of each training job. The following contract was set up in Mistral to measure the mean I/O latency for each write operation, the number of write operations, and the total amount of data written:

```
# This line tells Mistral to log data every 1000 seconds.
# Since this exceeds the run time of the application, the
# data reported will be a total for the whole run.
2,monitortimeframe,1000s

# This rule logs the mean write latency.
mean_lat_write,/,write,all,mean-latency,1us

# This rule logs the number of write calls.
num_write,/,write,all,count,1

# This rule logs the amount of data written.
total_write,/,write,all,bandwidth,1B
```

Using the Profile Results

As a result of the data produced by Mistral, the write cache settings were adjusted to minimize the total amount of time spent writing the data out. The optimal cache settings were:

- vm.dirty_ratio=30
- vm.dirty_background_ratio=20
- inode64, noatime, nodiratime, attr2, nobarrier, logbufs=8, logbsize=256k, osyncisdsync, nobootwait, noauto

This reduced download time from over 12 minutes to less than 3 minutes for a 40-image download.

Understanding Burst Credits

Making the right storage choices is key when balancing performance and cost. AWS offers general-purpose SSDs through their gp2 and io2 offerings that are great for almost all workloads but can be expensive for large amounts of data. HDD alternatives in the form of st1 and sc1 offer similar streaming throughput for a much lower price but cannot compete on iops. Burst credits need to be considered to ensure storage can cope with peaks in demand. In the case of this customer's application, the files easily fit onto the smallest volume size, 500GB, which has a burst throughput of 125MiBps. The disks start with 1TiB of burst credits and accumulate credits at a rate of 20MiBps.

After 45 minutes of processing the data images there are 20x45x60=54GiBps credits accumulated, which gives 7 minutes of peak 125MiBps throughput for the next image download. This means that the burst credits won't ever run out and the same volume can be reused for the next task. In this case, the streaming nature of the I/O meant that st1, the throughput-optimized HDD, was a good option, but it's important to profile I/O with a tool such as Mistral to understand the bursty or random nature of the I/O patterns and choose the right storage solution accordingly.