SageMaker Debugger Profiling Report

SageMaker Debugger auto generated this report. You can generate similar reports on all supported training jobs. The report provides summary of training job, system resource usage statistics, framework metrics, rules summary, and detailed analysis from each rule. The graphs and tables are interactive.

Legal disclaimer: This report and any recommendations are provided for informational purposes only and are not definitive. You are responsible for making your own independent assessment of the information.

In [4]:

```
# Parameters
processing_job_arn = "arn:aws:sagemaker:us-east-1:057716757052:processing-job/pytorch-training-
2023-04-0-profilerreport-4ff94a91"
```

Training job summary

The following table gives a summary about the training job. The table includes information about when the training job star how much time initialization, training loop and finalization took. Your training job started on 04/03/2023 at 14:46:32 and ran seconds. Your training job started on 04/03/2023 at 14:46:32 and ran for 507 seconds.. No step information was profiled fro job. The time spent on initialization and finalization cannot be computed.

| #		Job Statistics
0	Start time	14:46:32 04/03/2023
1	End time	14:54:59 04/03/2023
2	Job duration	507 seconds

System usage statistics

The 95th percentile of the total GPU utilization on node algo-1 is only 38%. The 95th percentile of the total CPU utilization switching to a smaller instance type.

The following table shows statistics of resource utilization per worker (node), such as the total CPU and GPU utilization, ar and GPU. The table also includes the total I/O wait time and the total amount of data sent or received in bytes. The table sl as p99, p90 and p50 percentiles.

#	node	metric	unit	max	p99	p95	p50
0	algo-1	Network	bytes	34889618.62	1139419.37	0	0
1	algo-1	GPU	percentage	75.5	67.86	38.16	0
2	algo-1	CPU	percentage	35.99	19.61	18.52	7.19
3	algo-1	CPU memory	percentage	12.99	12.98	12.95	6.88
4	algo-1	GPU memory	percentage	1.5	0.75	0.5	0
5	algo-1	I/O	percentage	2.77	2.07	1.53	0
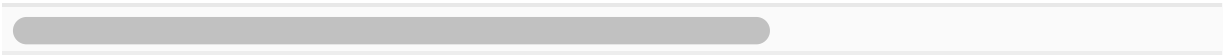
Framework metrics summary

Rules summary

The following table shows a profiling summary of the Debugger built-in rules. The table is sorted by the rules that triggered the most frequently. During your training job, the Dataloader rule was the most frequently triggered. It processed 0 datapoints and was triggered 0 times.

	Description	Recommendation	Number of times rule triggered	Number of datapoints	Rule parameters
Dataloader	Checks how many data loaders are running in parallel and whether the total number is equal the number of available CPU cores. The rule triggers if number is much smaller or larger than the number of available cores. If too small, it might lead to low GPU utilization. If too large, it might impact other compute intensive operations on CPU.	Change the number of data loader processes.	0	0	min_threshold:70 max_threshold:200
IOBottleneck	Checks if the data I/O wait time is high and the GPU utilization is low. It might indicate IO bottlenecks where GPU is waiting for data to arrive from storage. The rule evaluates the I/O and GPU utilization rates and triggers the issue if the time spent on the IO bottlenecks exceeds a threshold percent of the total training time. The default threshold is 50 percent.	Pre-fetch data or choose different file formats, such as binary formats that improve I/O performance.	0	511	threshold:50 io_threshold:50 gpu_threshold:10 patience:1000
GPUMemoryIncrease	Measures the average GPU memory footprint and triggers if there is a large increase.	Choose a larger instance type with more memory if footprint is close to maximum available memory.	0	508	increase:5 patience:1000 window:10
MaxInitializationTime	Checks if the time spent on initialization exceeds a threshold percent of the total training time. The rule waits until the first step of training loop starts. The initialization can take longer if downloading the entire dataset from Amazon S3 in File mode. The default threshold is 20 minutes.	Initialization takes too long. If using File mode, consider switching to Pipe mode in case you are using TensorFlow framework.	0	0	threshold:20
LoadBalancing	Detects workload balancing issues across GPUs. Workload imbalance can occur in training jobs with data parallelism. The gradients are accumulated on a primary GPU, and this GPU might be overused with regard to other GPUs, resulting in reducing the efficiency of data parallelization.	Choose a different distributed training strategy or a different distributed training framework.	0	508	threshold:0.2 patience:1000

	Description	Recommendation	Number of times rule triggered	Number of datapoints	Rule parameters
LowGPUUtilization	Checks if the GPU utilization is low or fluctuating. This can happen due to bottlenecks, blocking calls for synchronizations, or a small batch size.	Check if there are bottlenecks, minimize blocking calls, change distributed training strategy, or increase the batch size.	0	508	threshold_p95:70 threshold_p5:10 window:500 patience:1000
StepOutlier	Detects outliers in step duration. The step duration for forward and backward pass should be roughly the same throughout the training. If there are significant outliers, it may indicate a system stall or bottleneck issues.	Check if there are any bottlenecks (CPU, I/O) correlated to the step outliers.	0	0	threshold:3 mode:None n_outliers:10 stddev:3
BatchSize	Checks if GPUs are underutilized because the batch size is too small. To detect this problem, the rule analyzes the average GPU memory footprint, the CPU and the GPU utilization.	The batch size is too small, and GPUs are underutilized. Consider running on a smaller instance type or increasing the batch size.	0	507	cpu_threshold_p95:70 gpu_threshold_p95:70 gpu_memory_threshold_p95:70 patience:1000 window:500
CPUBottleneck	Checks if the CPU utilization is high and the GPU utilization is low. It might indicate CPU bottlenecks, where the GPUs are waiting for data to arrive from the CPUs. The rule evaluates the CPU and GPU utilization rates, and triggers the issue if the time spent on the CPU bottlenecks exceeds a threshold percent of the total training time. The default threshold is 50 percent.	Consider increasing the number of data loaders or applying data pre-fetching.	0	511	threshold:50 cpu_threshold:90 gpu_threshold:10 patience:1000

Analyzing the training loop

Step duration analysis

The StepOutlier rule measures step durations and checks for outliers. The rule returns True if duration is larger than 3 time also takes the parameter mode, that specifies whether steps from training or validation phase should be checked. In your p as None. Typically the first step is taking significantly more time and to avoid the rule triggering immediately, one can use n outliers to ignore. n_outliers was set to 10. The rule analysed 0 datapoints and triggered 0 times.

GPU utilization analysis

Usage per GPU

The LowGPUUtilization rule checks for a low and fluctuating GPU usage. If the GPU usage is consistently low, it might be
bottlenecks or a small batch size. If usage is heavily fluctuating, it can be due to bottlenecks or blocking calls. The rule com
and 5th percentile of GPU utilization on 500 continuous datapoints and found 0 cases where p95 was above 70% and p5 w
If p95 is high and p5 is low, it might indicate that the GPU usage is highly fluctuating. If both values are very low, it would m
machine is underutilized. During initialization, the GPU usage is likely zero, so the rule skipped the first 1000 data points. T
508 datapoints and triggered 0 times.

Workload balancing

The LoadBalancing rule helps to detect issues in workload balancing between multiple GPUs. It computes a histogram of G
GPU and compares then the similarity between histograms. The rule checked if the distance of histograms is larger than th
initialization utilization is likely zero, so the rule skipped the first 1000 data points.

The following histogram shows the workload per GPU on node algo-1. You can enable/disable the visualization of a worklo
label in the legend.

Workloads on node algo-1
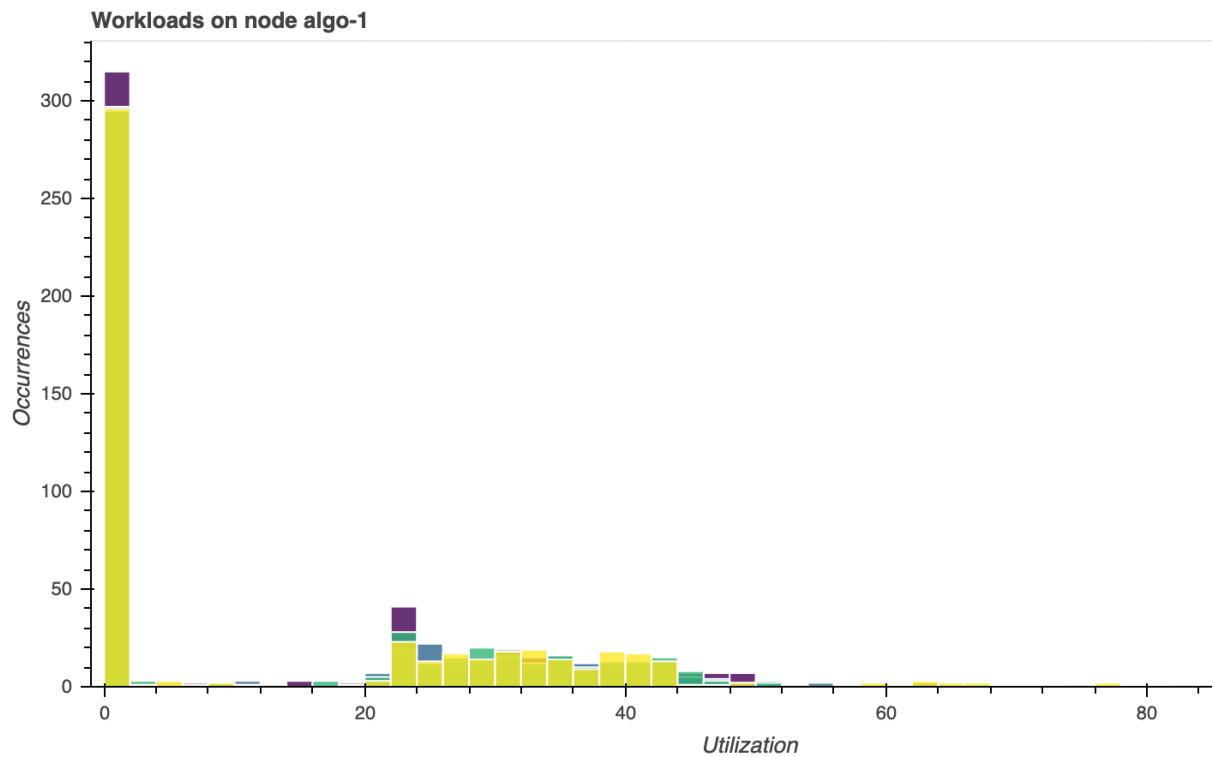
Dataloading analysis

The number of dataloader workers can greatly affect the overall performance of your training job. The rule analyzed the nu
that have been running in parallel on the training instance and compares it against the total number of cores. The rule chec
smaller than 70% or larger than 200% the total number of cores. Having too few dataloader workers can slowdown data pr
underutilization. Having too many dataloader workers may hurt the overall performance if you are running other compute ir
rule analysed 0 datapoints and triggered 0 times.
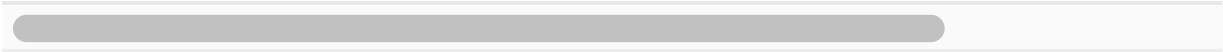
Batch size

The BatchSize rule helps to detect if GPU is underutilized because of the batch size being too small. To detect this the rule
GPU memory footprint, CPU and GPU utilization. The rule checked if the 95th percentile of CPU utilization is below cpu_th
70%, the 95th percentile of GPU utilization is below gpu_threshold_p95 of 70% and the 95th percentile of memory footprin
gpu_memory_threshold_p95 of 70%. In your training job this happened 0 times. The rule skipped the first 1000 datapoints.
computed the percentiles over window size of 500 continuous datapoints. The rule analysed 507 datapoints and triggered (

CPU bottlenecks

The CPUBottleneck rule checked when the CPU utilization was above cpu_threshold of 90% and GPU utilization was belo
initialization utilization is likely to be zero, so the rule skipped the first 1000 datapoints. With this configuration the rule foun
16% of the total time. This is below the threshold of 50% The rule analysed 511 data points and triggered 0 times.
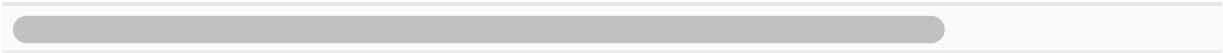
I/O bottlenecks

The IOBottleneck rule checked when I/O wait time was above io_threshold of 50% and GPU utilization was below gpu_thre
utilization is likely to be zero, so the rule skipped the first 1000 datapoints. With this configuration the rule found 21 I/O bott
time. This is below the threshold of 50%. The rule analysed 511 datapoints and triggered 0 times.

GPU memory

The GPUMemoryIncrease rule helps to detect large increase in memory usage on GPUs. The rule checked if the moving a
more than 5.0%. So if the moving average increased for instance from 10% to 16.0%, the rule would have triggered. Durin
so the rule skipped the first 1000 datapoints. The moving average was computed on a window size of 10 continuous datap
violations where the moving average between previous and current time window increased by more than 5.0%. The rule an
triggered 0 times.