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-1:440191285067:processing-job/pytorch-training-2021-12-0-profilerreport-5ff792c2"
```

Training job summary

The following table gives a summary about the training job. The table includes information about whe
how much time initialization, training loop and finalization took. Your training job started on 12/03/202
seconds. Your training job started on 12/03/2021 at 13:23:47 and ran for 192 seconds.. No step infor
job. The time spent on initialization and finalization cannot be computed.

#		Job Statistics
0	Start time	13:23:47 12/03/2021
1	End time	13:26:59 12/03/2021
2	Job duration	192 seconds

System usage statistics

The 95th percentile of the total GPU utilization on node algo-1 is only 24%. However, the 95th percer
because of CPU bottlenecks.

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

#	node	metric	unit	max	p99	p95
0	algo-1	Network	bytes	3807.13	140.02	0
1	algo-1	GPU	percentage	24	24	24
2	algo-1	CPU	percentage	97	94.27	89.96
3	algo-1	CPU memory	percentage	23.85	22.73	22.65
4	algo-1	GPU memory	percentage	21	21	20
5	algo-1	I/O	percentage	78.04	62.74	52.34

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	R
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	mi ma
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	385	
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	

	Description	Recommendation	Number of times rule triggered	Number of datapoints	R
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	385	
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	
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	385	thr tl
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	392	cp gp

	Description	Recommendation	Number of times rule triggered	Number of datapoints	R
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	392	i gp
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	384	cpu_thr gpu_thr gpu_memory_th

Analyzing the training loop

Step duration analysis

The StepOutlier rule measures step durations and checks for outliers. The rule returns True if duratic also takes the parameter mode, that specifies whether steps from training or validation phase should as None. Typically the first step is taking significantly more time and to avoid the rule triggering imme outliers to ignore. n_outliers was set to 10. The rule analysed 0 datapoints and triggered 0 times.

GPU utilization analysis

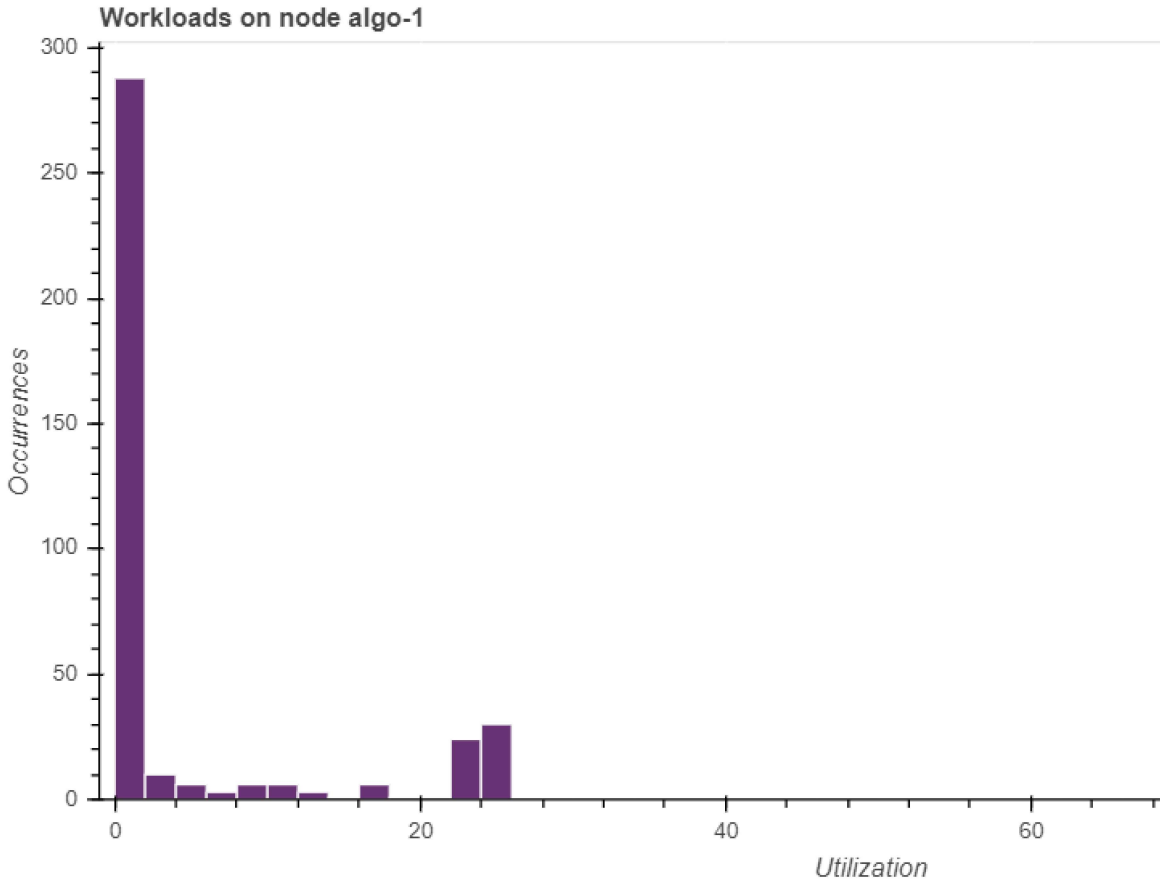
Usage per GPU

The LowGPUUtilization rule checks for a low and fluctuating GPU usage. If the GPU usage is consis
bottlenecks or a small batch size. If usage is heavily fluctuating, it can be due to bottlenecks or block
and 5th percentile of GPU utilization on 500 continuous datapoints and found 0 cases where p95 wa
If p95 is high and p5 is low, it might indicate that the GPU usage is highly fluctuating. If both values a
machine is underutilized. During initialization, the GPU usage is likely zero, so the rule skipped the fi
385 datapoints and triggered 0 times.

Workload balancing

The LoadBalancing rule helps to detect issues in workload balancing between multiple GPUs. It com
GPU and compares then the similarity between histograms. The rule checked if the distance of histo
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 vi
label in the legend. Your training job only used one GPU so there is no workload balancing issue.



Dataloading analysis

The number of dataloader workers can greatly affect the overall performance of your training job. The that have been running in parallel on the training instance and compares it against the total number o smaller than 70% or larger than 200% the total number of cores. Having too few dataloader workers underutilization. Having too many dataloader workers may hurt the overall performance if you are rur 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. GPU memory footprint, CPU and GPU utilization. The rule checked if the 95th percentile of CPU utili 70%, the 95th percentile of GPU utilization is below gpu_threshold_p95 of 70% and the 95th percent gpu_memory_threshold_p95 of 70%. In your training job this happened 0 times. The rule skipped the computed the percentiles over window size of 500 continuous datapoints. The rule analysed 384 dat

CPU bottlenecks

The CPUBottleneck rule checked when the CPU utilization was above cpu_threshold of 90% and GP initialization utilization is likely to be zero, so the rule skipped the first 1000 datapoints. With this conf 37% of the total time. This is below the threshold of 50% The rule analysed 392 data points and trigg

I/O bottlenecks

The IOBottleneck rule checked when I/O wait time was above io_threshold of 50% and GPU utilizatio utilization is likely to be zero, so the rule skipped the first 1000 datapoints. With this configuration the time. This is below the threshold of 50%. The rule analysed 392 datapoints and triggered 0 times.

GPU memory

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