

Performance Engineering Platform

Fixstars AI Booster

Case Studies



Try it for Free!

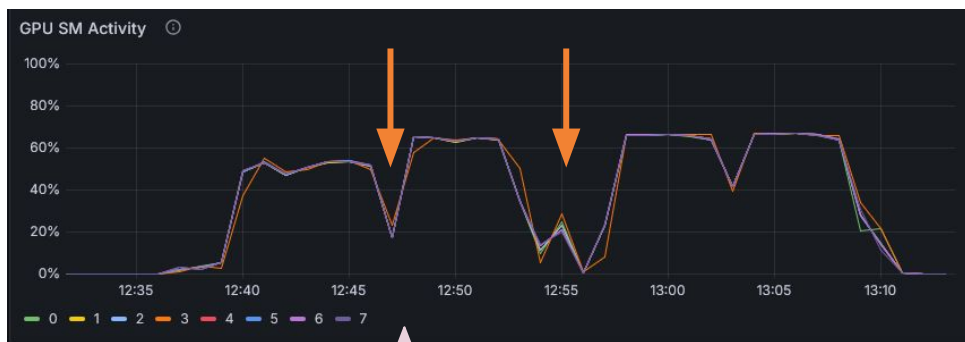
<https://www.fixstars.com/en/ai/booster>

Case 1: Continuous Visualization of AI Training Workloads for Anomaly Detection

By regularly monitoring changes in GPU utilization during AI job execution, users can quickly notice performance degradation.
→you can't improve what you don't measure

Issues with Similar GPU Visualization Tools:

Tool	Issues
MLFlow	<ul style="list-style-type: none">• Captures data per iteration/epoch (low monitoring frequency)
Nsight Systems	<ul style="list-style-type: none">• Requires user to have sudo privileges• Data collection must be performed manually
PyTorch Profiler	<ul style="list-style-type: none">• High overhead• Analysis of results required



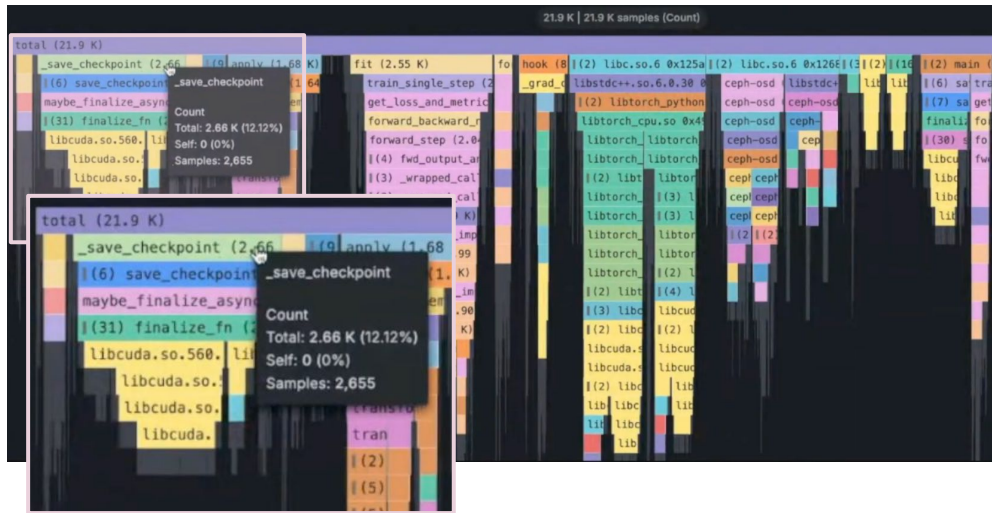
- Significant drops in SM Activity between epochs cannot be ignored
- Marked decrease in SM Activity during validation phases

Case 2: Identifying Software Bottlenecks using Flame Graph

- The dashboard's "Flame Graph" visualization feature breaks down processes at the library and function level.
- Identify bottlenecks throughout the job and subsequently obtain detailed profiles to facilitate improvements.

What is a Flame Graph?

A flame graph visualizes the time spent by software processes, intuitively illustrating which functions or operations consume the most time. The vertical axis shows the call stack hierarchy, while the horizontal axis indicates execution time proportionally—longer bars represent longer durations. It is highly effective for identifying performance bottlenecks.



Checkpoint operations (saving intermediate AI training states to resume from that point if training halts) account for **12% of the measured duration**.
→ Depending on your system, consider reducing checkpoint frequency or omitting them entirely.

AI Booster Practical Tutorial: Improving Processing Speed for Llama4 Scout's Continued Pre-training

Step 1: Check Dashboard to Identify Current Issues

GPU Utilization appears consistently high at first glance.

99.1%

GPU core utilization efficiency is low (~20%).

21.3%

Periodic GPU stalls observed.

Interconnect bandwidth utilization for Send/Recv averages about 6GB/s (theoretical bandwidth: 25GB/s).



AI Booster Practical Tutorial:

Improving Processing Speed for Llama4 Scout's Continued Pre-training



Step 2. Inspect the Flame Graph to Identify Bottlenecks

```
'total' (23578)
'python3.11' (20461)
'pt_autograd_1' (1330)
(6) '[unknown] in libc.so.6' (1246)
(2) 'BackwardCFunction.apply [0x5d] [5d]' (1245)
(5) 'CheckpointFunction.backward [0x1ac] [1ac]' (563)
(2) 'Module._call_impl.<locals>.inner [0x14c] [14c]' (562)
(5) 'Llama4TextDecoderLayer.forward [0x66] [66]' (552)
(3) 'Module._call_impl.<locals>.inner [0xfb] [fb]' (550)
(10) 'DeepSpeedZeRoOffload._register_deepspeed_module.<locals>._pre_forward_module_hook [0x15] [15]' (549)
(6) 'PartitionedParameterCoordinator.fetch_sub_module [0x36a] [36a]' (545)
'[unknown] in'
'[unknown] in'
```

Blocked within [DeepSpeed's fetch_sub_module](#) method

Comment for
the method

This method does the following (in order):

1. kick off fetch for parameters in immediately required sub module
2. kick off fetch for next few parameters we will need later (prefetch)
3. block on parameters in immediately required sub module

Possibly waiting for distributed
parameters across nodes?

AI Booster Practical Tutorial: Improving Processing Speed for Llama4 Scout's Continued Pre-training



Step 2. Identifying Another Bottleneck via Flame Graph (Part 2)

```
'total' (23578)
'python3.11' (20461)
'pt_autograd_0' (1912)
'[unknown] in libc.so.6' (1831)
(5) '[unknown] in libstdc++.so.6.0.30' (1232)
(2) 'BackwardCFunction.apply [0x5d] [5d]' (1230)
(8) 'CheckpointFunction.backward [0x2e7] [2e7]' (692)
(2) '[unknown] in libtorch_python.so' (685)
(2) 'instrument_w_nvtx.<locals>.wrapped_fn [0x43] [43]' (673)
(5) 'DeepSpeedZeroOptimizer Stage3.create_reduce_and_remove_grad_hooks.<locals>.wrapper_pre_hook.<locals>.forward_pre_hook.<locals>.reduce_leaf_module_grads'
(2) 'context_decorator.<locals>.decorate_context [0x13] [13]' (637)
(6) 'DeepSpeedZeroOptimizer Stage3._reduce_and_partition_ipg_grads [0x155] [155]' (626)
'[unknown] in'
'[unknown] in'
```

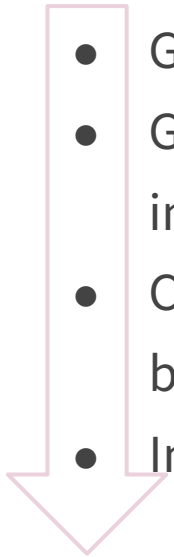
Blocked within DeepSpeed ZeRo3's
[reduce and partition ipg_grads](#) method

Comment for
the method

```
while self.param_reduce_events and self.param_reduce_events[0].query():
    self.param_reduce_events.popleft()
if len(self.param_reduce_events) > self.max_param_reduce_events:
    self.param_reduce_events.popleft().synchronize()
```

Possibly waiting for gradient
aggregation completion events?

Step 3. Analyzing the Gathered Information

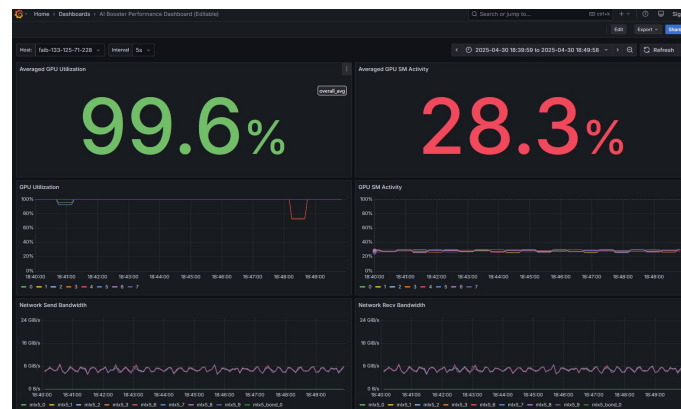
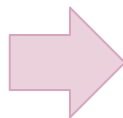
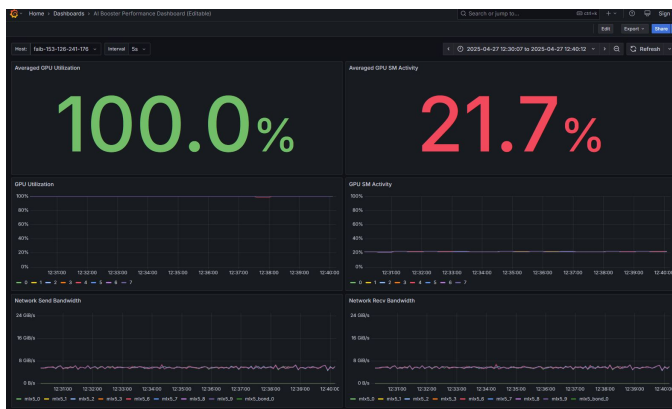
- 
- GPU not operating at full power.
 - GPU stalls likely due to checkpoint writing, though minimally impactful overall.
 - Comprehensive analysis of flame graphs and source code indicates blockages due to data aggregation from distributed nodes.
 - Interconnect bandwidth appears to have additional capacity.

Potential inefficiencies identified within DeepSpeed ZeRO?

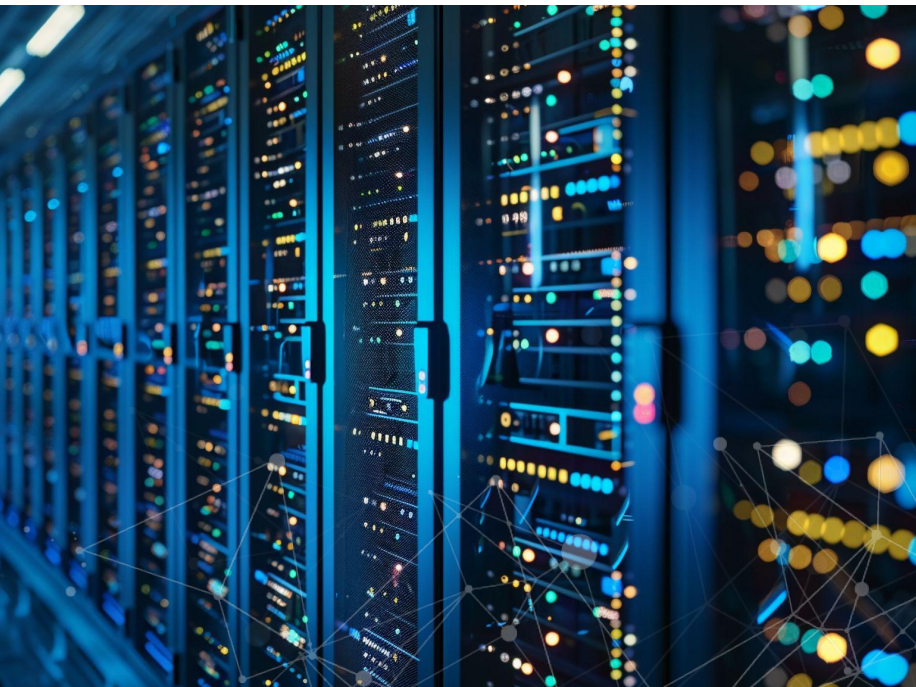
AI Booster Practical Tutorial: Improving Processing Speed for Llama4 Scout's Continued Pre-training

Step 4. Optimizing the Process

- Based on analysis, efforts were made to optimize DeepSpeed ZeRO.
- Use parameters derived from 40 trials performed by the DeepSpeed Hyperparameter Tuner



- Reduced processing time by 20% (from 50[s/itr] to 40[s/itr]).
- Improved communication bandwidth (from 6.0 GiB/s to 7.5 GiB/s).
- Enhanced GPU processor efficiency (from 22% to 28%).



Contact Us

For more details about Fixstars AI Booster, please contact us

Fixstars Corporation

Email support-ai@fixstars.com

Web <https://www.fixstars.com/en/ai/booster>