



 HCMC AI MEETUP #1 - MENLO RESEARCH

TRAINING ON NVIDIA GH200 OPTIMIZING FOR ARM ARCHITECTURE.

STUBBORN STRAWBERRIES

Stubborn Strawberries



Bao Huynh Thai
Student@UIT



Phat Nguyen Thuan
Student@UIT



Thai Hoang Minh
Student@UIT



Long Le Bao
Student@UIT



Quoc-Bao Nguyen
Staff@Zalo

We are a **research team** from **UIT-VNUHCM** and **Zalo**.
Our research areas are: **Text to 3D, GPU Workload**.

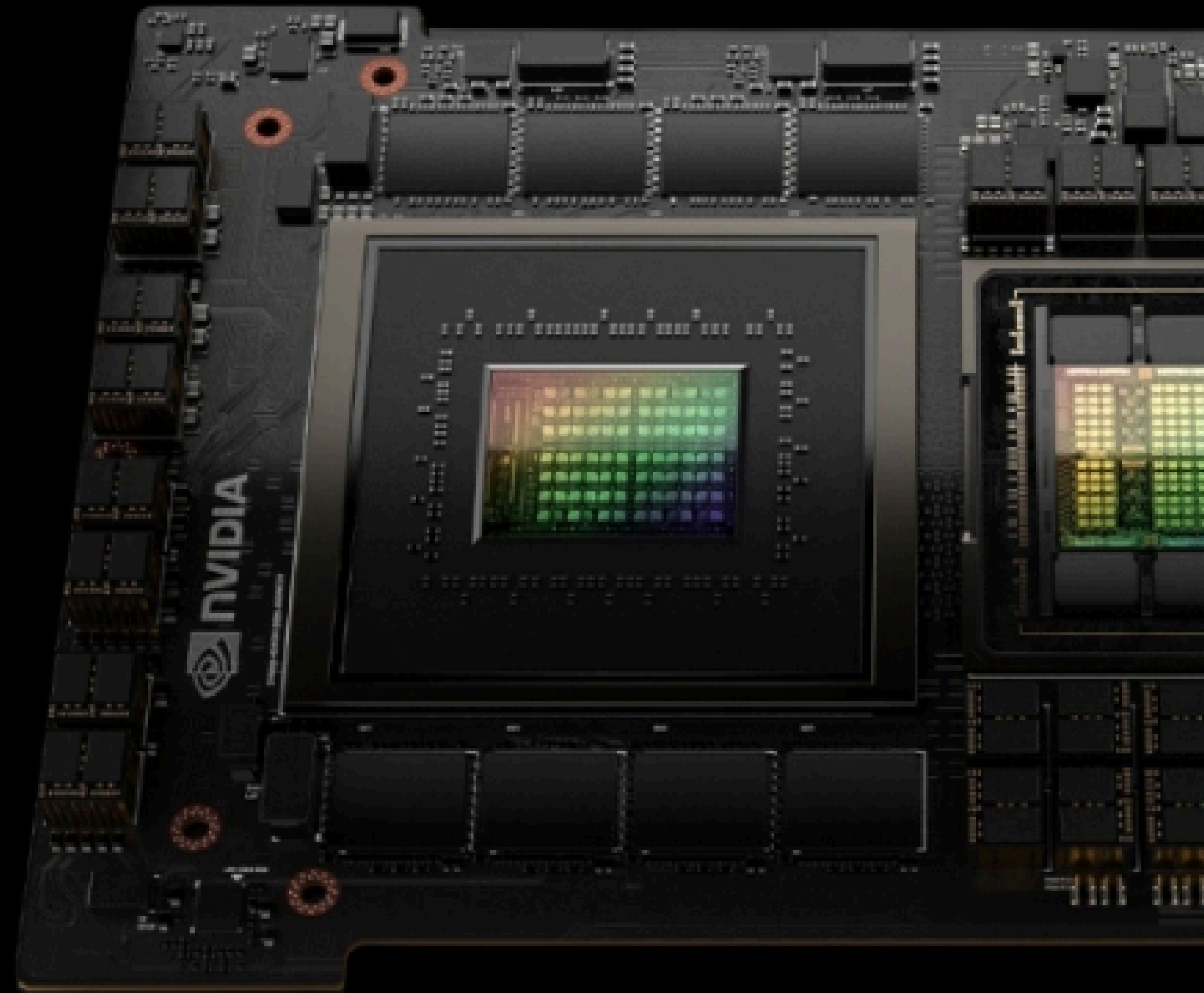
Table of Contents

- 1 NVIDIA GH200
- 2 LGM
- 3 Problem & solution
- 4 GH200 vs A100 - Compare
- 5 Conclusion

01 About NVIDIA GH200

NVIDIA GH200

- **GH200** = Grace CPU + Hopper GPU
- **NVLink-C2C** for a unified CPU+GPU memory model.
- **900 GB/s** bandwidth — **7× faster** than PCIe Gen5.
- **HBM3/HBM3e** for high memory bandwidth and capacity.
- **Optimized** for AI, HPC, and generative AI workloads.
- Supports **NVIDIA AI Enterprise, HPC SDK, and Omniverse™**.





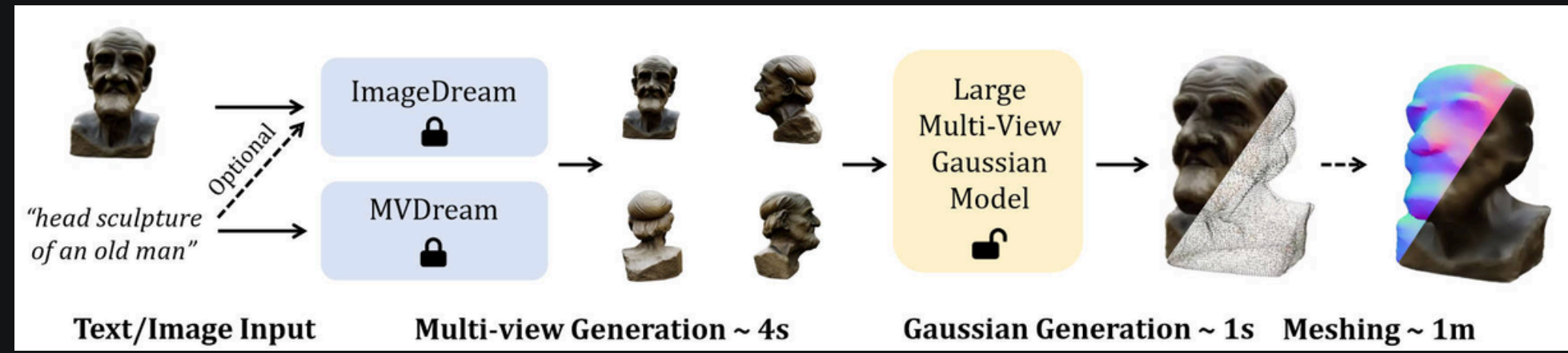
02

LGM

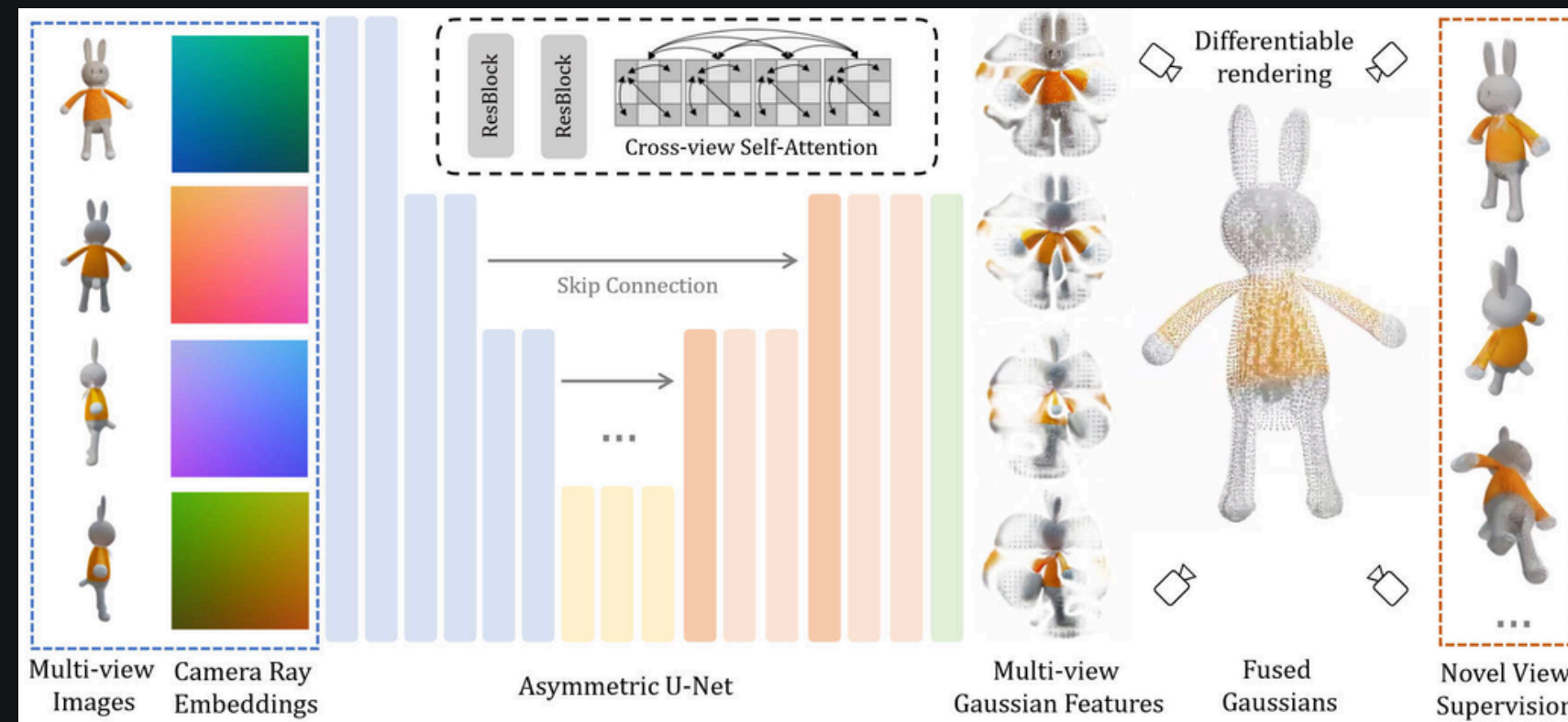
Large Multi-View Gaussian Model for
High-Resolution 3D Content Creation.

LGM

- **LGM** generates **high-resolution 3D models** from text or images in **~5s**.
- Uses **multi-view Gaussian features** and **asymmetric U-Net** for rendering.
- **Fixes blurry output and high compute** in feed-forward models.
- Efficient multi-view fusion enables **faster, sharper** 3D generation.



Pipeline



Architecture

Scan here





03

Problem

Problems when train and infer LGM

Problems for AArch64-based machines

- **xFormers** is **crucial** for training LGM and modern AI models.
- Building xFormers built on **Flash-Attn 2.x.x from source** is often **infeasible**:
 - Uses **precompiled CUDA binaries** for x86_64 architectures.
 - Installing xFormers via pip often **resulted in long compilation times**.

Problems for AArch64-based machines



lw on Nov 12, 2024

Contributor ...

This is unlikely to happen anytime soon, as we don't have the time and resources to debug and fix issues on ARM64, nor to duplicate all the build and test jobs in our CI.

We might accept fixes that resolve compatibility with ARM64 if they do not introduce too much complexity, but this would remain best-effort and community-driven.



6



drikster80 on Jul 12, 2024

Contributor ...

Had a similar problem on the GH200 (aarch64 Grace CPU).

Similar to [@haileyschoelkopf](#), I updated the Dockerfile and requirements to work with v0.5.1. Here is the forked version: <https://github.com/drikster80/vllm/tree/gh200-docker>

Main issues that needed to be overcome:

- Use Nvidia's Pytorch container due to PyTorch not supporting ARM64. (specifically nvcr.io/nvidia/pytorch:24.04-py3 to ensure PyTorch 2.3 build and latest optimizations (e.g. Lightning-Thunder). [Release Notes for 24.04-py3](#)
- xformers hangs on pip install. Not sure why (maybe just taking forever to compile?)
- Triton needs to be installed from source
- vllm-flash-attn needs to be built from source
- Comment out "torch", "xformers", and "vllm-flash-attn" in requirements files (handling that in the Dockerfile directly).

For future updating, you can see the changes here: [drikster80@359fd4f](#)



4



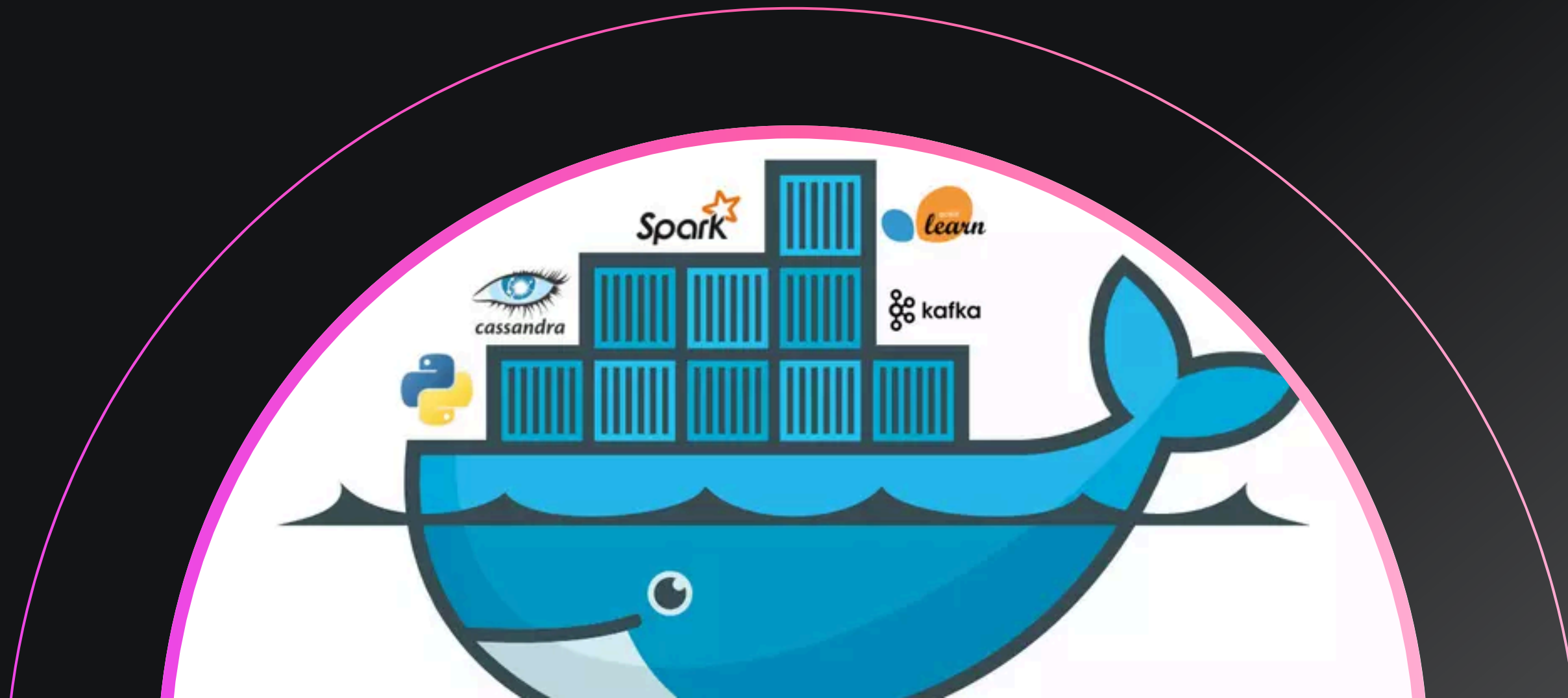
Attempts to fix these issues

- Use docker
- Cross-compilation
- Build from source


Use Docker

QEMU emulation --> install x64_x86 version into Docker


*Note: Emulation with QEMU is **slower** than native,
only: **testing or building**.





Use Docker

 **gh200-llm** Public

 Watch **3**

 **main** ▾


 **2 Branches**  **0 Tags**

 Go to file

 t






Add file ▾

 **Code** ▾

 **arvindsun** Update to vllm 0.7.3

d77503b · 3 months ago

 **56 Commits**

 scripts	Latest docker with new pytorch; vllm and xformers	9 months ago
 training	New script + readme updates	4 months ago
 Dockerfile	Update to vllm 0.7.3	3 months ago
 LICENSE	Initial commit	2 years ago
 README.md	Update README.md	3 months ago


```
e wgmma.fence without CUTE_ARCH_MMA_SM90A_ENABLEDroot@c88530b76a4b:/workspace/root/LGM/LGM#
```


Use Docker

[illegible]

We are building on a ARM architecture (GH200) **without a compatible GPU** → the compiler doesn't know what to set **CUTE_ARCH_MMA** to. We were trying to set **TORCH_CUDA_ARCH_LIST = 9.0** (for ARM) but it didn't work

Cross-compilation

- xFormers was **compiled** for A100 with **TORCH_CUDA_ARCH_LIST=9.0**.
- Used **QEMU emulation** and **multi-arch Docker** to run **x86_64** in the container.
- Enables porting to A100 via **containerized x86 environment**.
- Downside: Requires a **virtualized runtime**, reducing **performance** and **not fully utilizing GH200's speed**.


```
e wgmma.fence without CUTE_ARCH_MMA_SM90A_ENABLEDroot@c88530b76a4b:/workspace/root/LGM/LGM#
```


First attempt to build from source

- **Building from source** from xFormers repo.
- **Uses Flash-Attention 2.x.x** and **Torch 2.6.0**
- **Issue Encountered:**

```
ERROR: Failed building wheel for xformers
Running setup.py clean for xformers
Running command python setup.py clean
/root/yes/envs/test/lib/python3.12/site-packages/setuptools/dist.py:759: SetuptoolsDeprecationWarning:
License classifiers are deprecated.
!!

*****
Please consider removing the following classifiers in favor of a SPDX license expression:

License :: OSI Approved :: BSD License

See https://packaging.python.org/en/latest/guides/writing-pyproject-toml/#license for details.
*****

!!
  self._finalize_license_expression()
running clean
'build/lib.linux-aarch64-cpython-312' does not exist -- can't clean it
'build/bdist.linux-aarch64' does not exist -- can't clean it
'build/scripts-3.12' does not exist -- can't clean it
Failed to build xformers
ERROR: Failed to build installable wheels for some pyproject.toml based projects (xformers)
```

First attempt to build from source

Reason for build fails:

- Limited support for “**sm_90a**” - a variant of the Hopper architecture.
- Flash-Attn 2.x.x and torch 2.6.0 **only supports** up to “**sm_90**” (NVIDIA H100 (Hopper GPU)).

```
ValueError: invalid literal for int() with base 10: '90a'  
error: subprocess-exited-with-error
```


Second attempt to build from source

- **Latest update from xFormers adds support for local attention on the Flash3 backend (H100).**

`v0.0.30` - build for PyTorch 2.7.0 Latest

Pre-built binary wheels are available for PyTorch 2.7.0. Following PyTorch, we build wheels for CUDA 11.8, 12.6, and 12.8 only (we no longer build for CUDA 12.4).

xFormers now requires PyTorch ≥ 2.7

Added

- [fMHA] Added support for local attention on the Flash3 backend (H100)
- [fMHA] Added a new paged gappy attention bias

Improved

- [fMHA] The FlashAttention3 backend now ships with more head dimensions to support MLA, and with a FLOPs formula in order to be compatible with PyTorch's partitioner-base automatic activation checkpointing
- The fused operators for sequence parallelism were migrated to PyTorch's SymmetricMemory
- The profiler prepends the traces' filenames with the rank of the process when doing distributed training

Removed

- Removed documentation for legacy unmaintained components

Second attempt to build from source

- **Flash Attention 3** is specifically built for **Hopper GPUs**.
- It's **1.5-2.0x faster** than FlashAttention-2, i.e., **75% utilization** of H100 theoretical **max FLOPS**

Tri Dao

AboutBlogPublicationsRepositoriesctrl kQ

FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision

AUTHORS	AFFILIATIONS	PUBLISHED
Jay Shah	Colfax	July 11, 2024
Ganesh Bikshandi	Colfax	
Ying Zhang	Meta	
Vijay Thakkar	NVIDIA, Georgia Tech	
Pradeep Ramani	NVIDIA	
Tri Dao	Princeton, Together AI	

Second attempt to build from source

- **Torch2.7.0** now supports “sm_90a”!

```
(test) root@gh200-new:~/test/xformers# python3
Python 3.12.10 | packaged by conda-forge | (main, Apr 10 2025, 22:10:27) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.cuda.get_arch_list())
['sm_50', 'sm_80', 'sm_86', 'sm_89', 'sm_90', 'sm_90a']
>>> |
```


GH200 vs A100

Speed

Power Draw

Time

Cost

We **trained and inferred LGM** at the same settings on **A100 and GH200** to compare the following metrics

Speed

Compare **SM Clock Speed** (Streaming Multiprocessor Clock):

- SM Clock: **main speed** of CUDA cores.
- **Handles tensor ops** like GEMM, convolution, attention.
- Key for **FP16, BF16, and Tensor Core workloads**.
- Comparing SM Clock: reveals **true training compute speed**.

Speed

GPU

gpu-a100 ▾

<

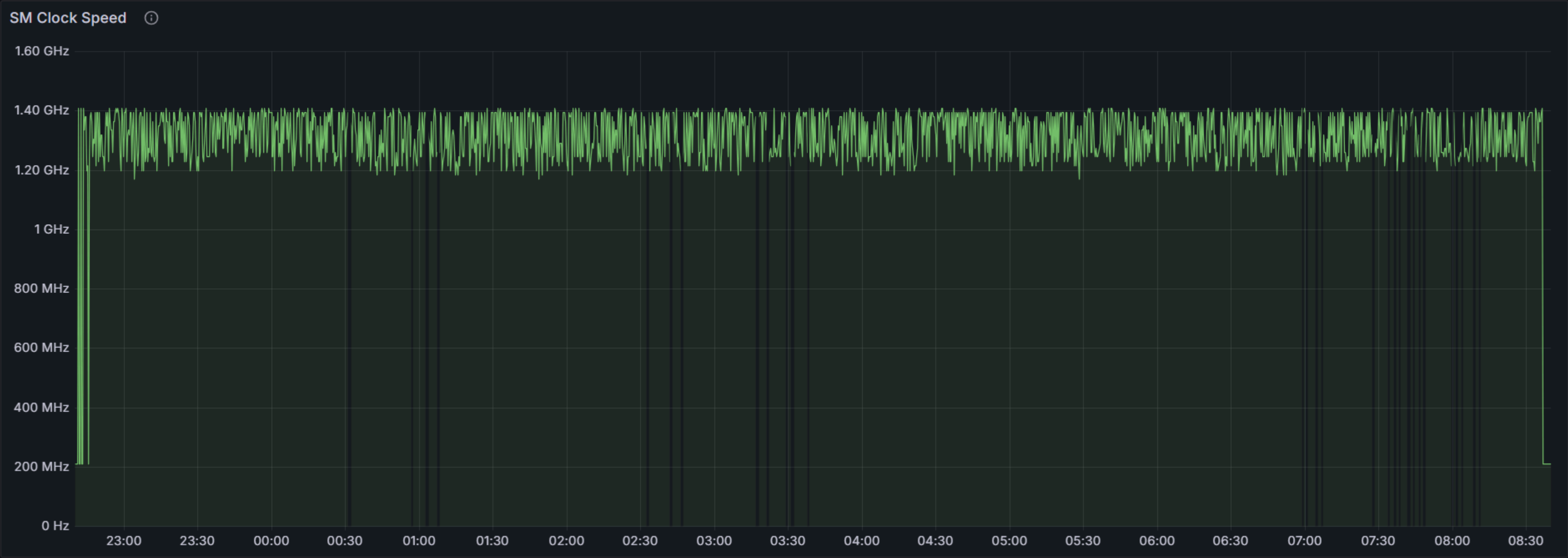
🕒 2025-05-20 22:40:00 to 2025-05-21 08:40:00 ▾

>

🔍

🔄 Refresh

10s ▾



Speed

GPU

gh200-new

<

🕒

2025-05-20 08:50:00 to 2025-05-20 13:20:00

>

🔍

🔄

Refresh

10s



Power

GPU

gpu-a100 ▾

<

🕒 2025-05-20 22:40:00 to 2025-05-21 08:40:00 ▾

>

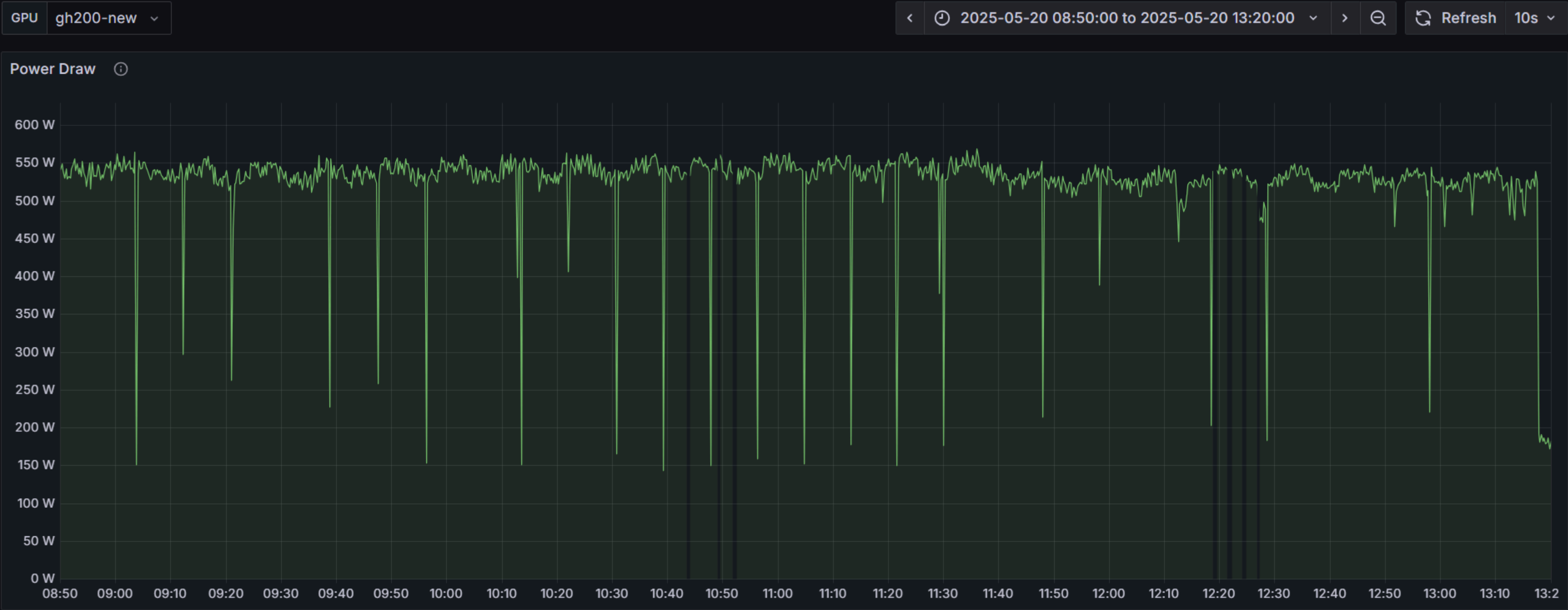
🔍

🔄 Refresh

10s ▾



Power



Time

A100: 22:40:00(20/5) --> 08:40:00 (21/5)

GH200: 08:50:00(20/5) --> 13:20:00 (20/5)

Cost *rent cost

	GH200	A100
Price for rent in 1h (\$/h)	1.5\$	1.25 - 1.5\$ (Depends on the provider)
Training time (h)	4,5h	10h
Total price (\$)	6.75\$	12.5 - 15\$

Why use ARM CPUs instead of Intel or AMD?

- Faster **CPU <> GPU** transfer: **NVLink-C2C @ 900 GB/s**, much faster than **PCIe** → **reduces bottlenecks** when offloading during **training/inference**.
- Less **offloading needed**: **GH200** has slightly **more GPU VRAM** → less data needs to go through CPU, **boosting speed**.
- **AI-tuned CPU**: **Grace CPU** is **optimized for ML workloads**, better at **feeding data and coordinating tasks** than general-purpose Intel/AMD CPUs.

Is GH200 suitable for training, inference, or both? Why?

- GH200 is great for both training and inference, but best for large-scale training.
- Combines Grace ARM CPU + H100 GPU via NVLink-C2C (900 GB/s) and unified memory (up to 480 GB).
- **Training:**
 - Fast CPU-GPU memory sharing, no bottlenecks.
 - H100 excels with FP8/TF32 Tensor Cores.
- **Inference:**
 - Unified memory handles large batches well.
 - But may be overkill for small-scale inference.

Why is the default kernel page size 64KB (on ARM), not 4KB?

- **Higher efficiency:** 64KB pages mean fewer OS-managed pages → better TLB hit rate, lower latency.
- **Ideal for AI/HPC:** Large tensors benefit from less fragmentation and overhead.
- **ARM optimized:** Many ARM kernels use `CONFIG_ARM64_64K_PAGES=y`.
- **Trade-off:** May waste memory for small allocations, but negligible in AI workloads.

Contact us



Bao Huynh Thai
Student@UIT



Phat Nguyen Thuan
Student@UIT



Thai Hoang Minh
Student@UIT



Long Le Bao
Student@UIT



Quoc-Bao Nguyen
Staff@Zalo

{23520105, 23521146, 23521414, 23520877}@gm.uit.edu.vn

baonq5@vng.com.vn



Thank You