



DUALign: A Hardware-Assisted Memory Mapping Strategy for PNM-Accelerated LLM Serving

Euijun Kim[1], Joshua Gabriel Dela Rosa[1], Bogil Kim[1], Youngin Kim[1], Seungyong Lee[2], Joonseop Sim[2], Youngpyo Joo[2], and William J. Song[1]

[1]School of Electrical & Electronic Engineering, Yonsei University
[2]Memory System Research, SK hynix

Content

- **Background:** LLM serving frameworks and PNM technology
- **Motivation:** Performance loss from PNM integration into GPUs
- **DUALign:** Memory mapping strategy for PNM-accelerated LLM serving
- **Evaluation:** Experimental setup and evaluation results
- **Conclusion**

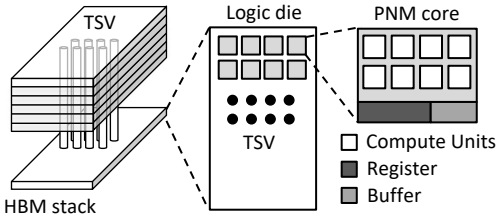
Background: Efficient LLM inference

- LLM inference involves memory-intensive operations that require **large memory capacity** and **memory bandwidth.**
- **Two complementary directions** have emerged to address these bottlenecks.

vLLM

vattention Public

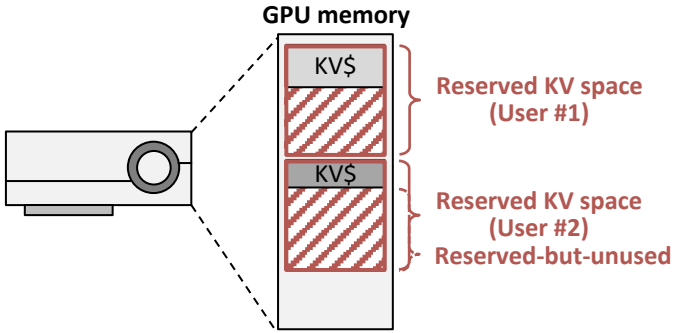
LLM serving frameworks



Processing-near-memory (PNM) technology

Background: LLM serving frameworks

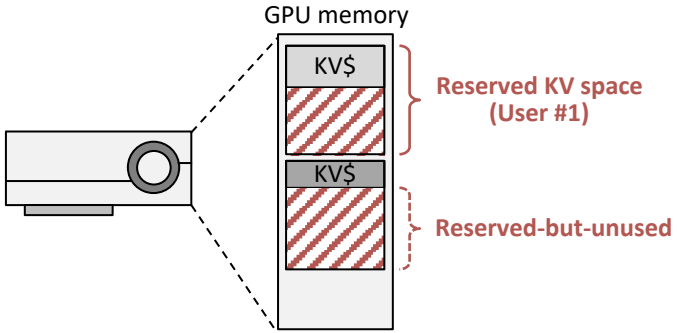
- **Unpredictable KV cache growth** causes memory inefficiency in conventional LLM serving in GPUs.
- Allocate KV cache space on-demand as they grow, minimizing pre-reserved capacity.



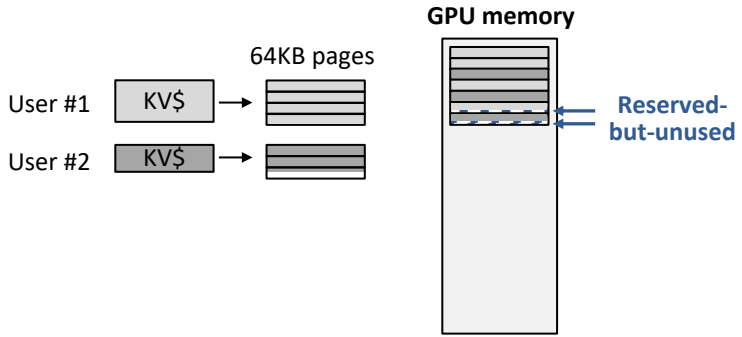
Conventional approach

Background: LLM serving frameworks

- Unpredictable KV cache growth causes memory inefficiency in conventional LLM serving in GPUs.
- Allocate KV cache space **on-demand** as they grow, minimizing pre-reserved capacity.



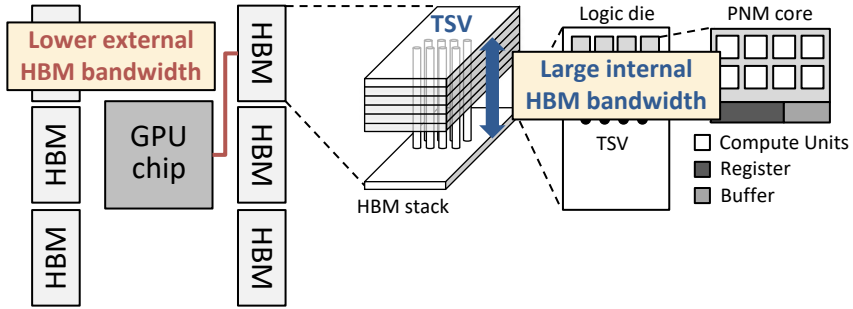
Conventional approach



LLM serving framework (vAttention)

Background: PNM technology

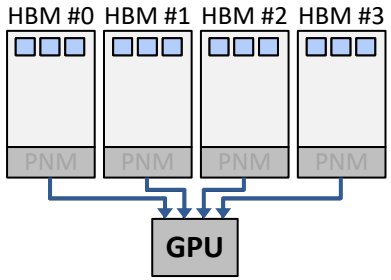
- PNM units are integrated into the HBM logic die to exploit **high internal bandwidth** enabled by advances in through-silicon-via (TSV) technology.
- **Memory-intensive attention** operations during generation are offloaded to **PNM,** while compute-intensive operations remain on the GPU.



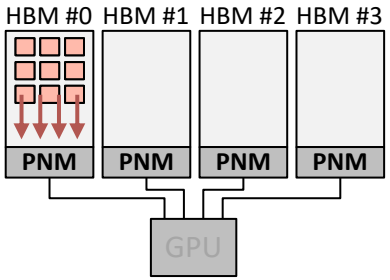
PNM-integrated GPU

Motivation: Performance loss from PNM integration into GPUs

- GPU and PNM have **conflicting memory mapping requirements.**
- **GPU** data should be **distributed across HBM stacks** for aggregate bandwidth.
  PNM data should be **gathered within a local HBM stack** to exploit internal bandwidth.



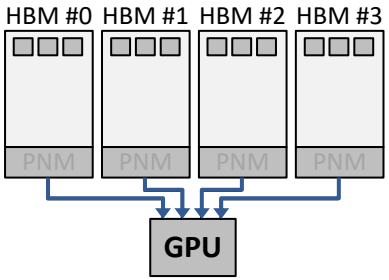
Memory mapping for GPU



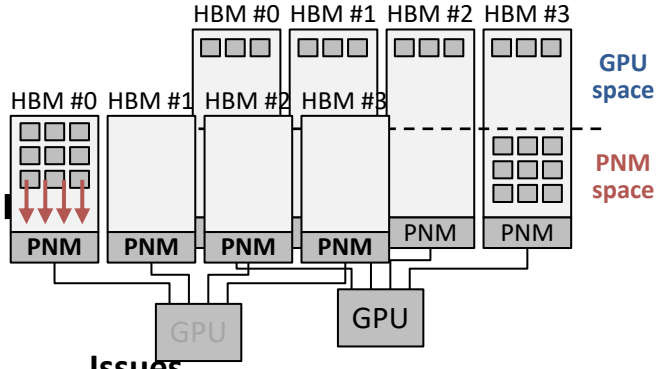
Memory mapping for PNM

Motivation: Performance loss from PNM integration into GPUs

- GPU and PNM have **conflicting memory mapping requirements.**
- **GPU** data should be **distributed across HBM stacks** for aggregate bandwidth.
  PNM data should be **gathered within a local HBM stack** to exploit internal bandwidth.



Memory mapping for GPU

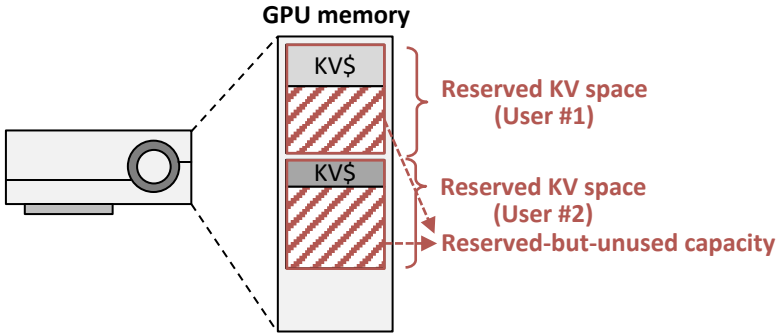


Issues

1. Separated GPU&PNM memory space.
2. Physical-addressing PNM for stack-local data placement.

Motivation: Performance loss from PNM integration into GPUs

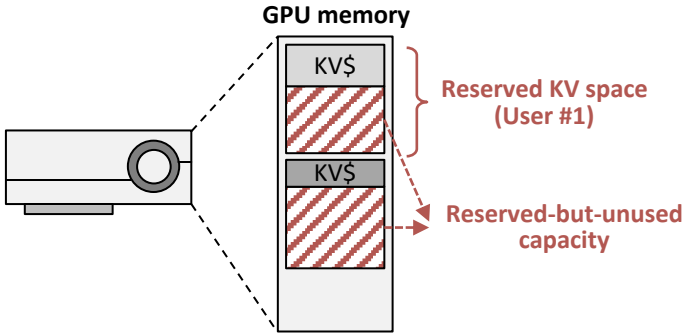
- Physical-addressing PNM: **Incompatible with LLM serving frameworks** that rely on virtual memory.
- Separated memory space: PNMs cannot use free GPU memory.



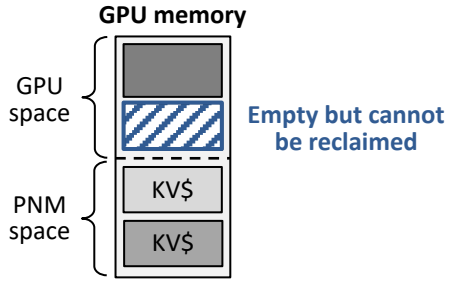
Disabled LLM serving frameworks

Motivation: Performance loss from PNM integration into GPUs

- Physical-addressing PNM: Incompatible with LLM serving frameworks that rely on virtual memory.
- Separated memory space: **PNMs cannot use free GPU memory.**



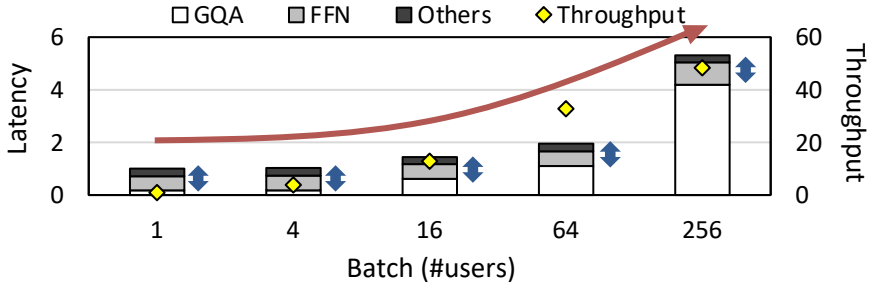
Disabled LLM serving frameworks



PNMs cannot use free GPU space

Motivation: Performance loss from PNM integration into GPUs

- Physical-addressing PNM and separated memory **reduce effective memory capacity.**
- Reduced effective memory capacity **lowers LLM inference throughput.**



FFN latency increases only slightly due to batching.

Memory virtualization should be supported in PNMs.

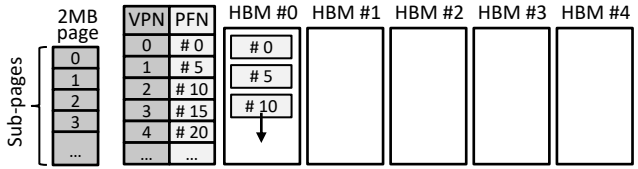
- LLM serving frameworks can be used in PNMs, minimizing memory capacity loss.
- Free GPU memory space can be reclaimed by PNMs through APIs (e.g., malloc and free).

DUALign: Memory mapping strategy

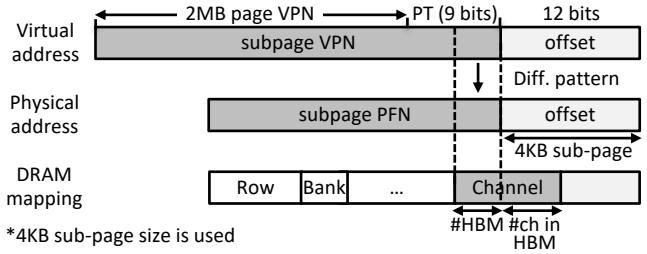
- Enables **HBM-stack-aware data placement** while supporting **virtual memory.**
- **Subpages** are used as a fine-grained data placement unit.



(a) Subpage **PFN assignment for GPU page**



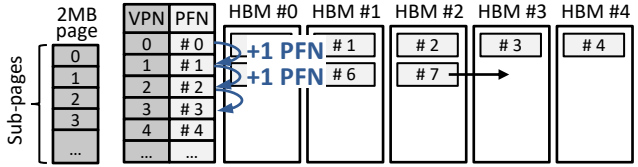
(b) Subpage **PFN assignment for PNM page**



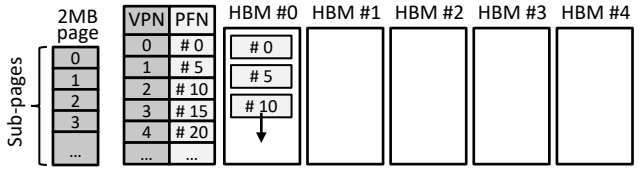
Stack-selection bits are aligned with subpage PFN LSB bits.

DUALign: Memory mapping strategy

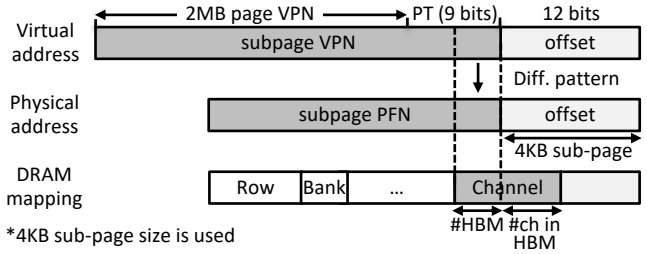
- Enables **HBM-stack-aware data placement** while supporting **virtual memory.**
- **Subpages** are used as a fine-grained data placement unit.



(a) Subpage **PFN assignment for GPU page**



(b) Subpage PFN assignment for PNM page



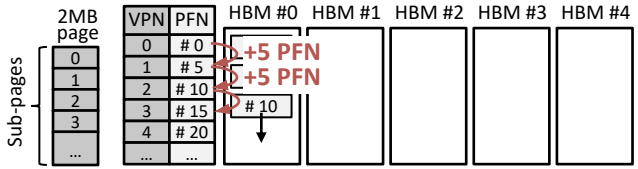
Stack-selection bits are aligned with subpage PFN LSB bits.

DUALign: Memory mapping strategy

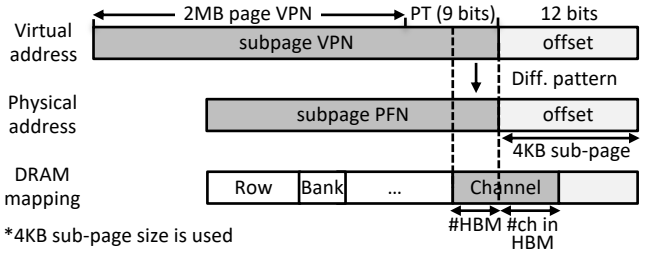
- Enables **HBM-stack-aware data placement** while supporting **virtual memory.**
- **Subpages** are used as a fine-grained data placement unit.



(a) Subpage PFN assignment for GPU page



(b) Subpage **PFN assignment for PNM page**



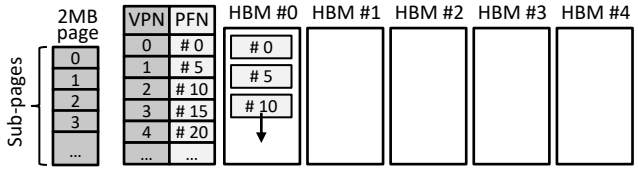
Stack-selection bits are aligned with subpage PFN LSB bits.

DUALign: Memory mapping strategy

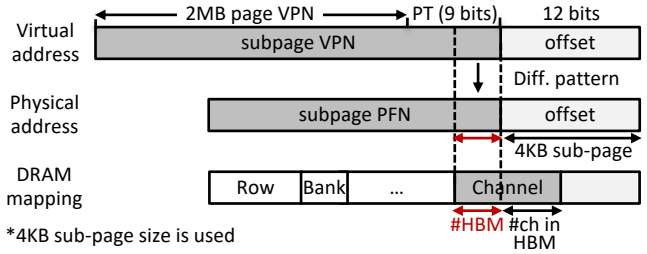
- Enables **HBM-stack-aware data placement** while supporting **virtual memory.**
- **Subpages** are used as a fine-grained data placement unit.



(a) Subpage PFN assignment for GPU page



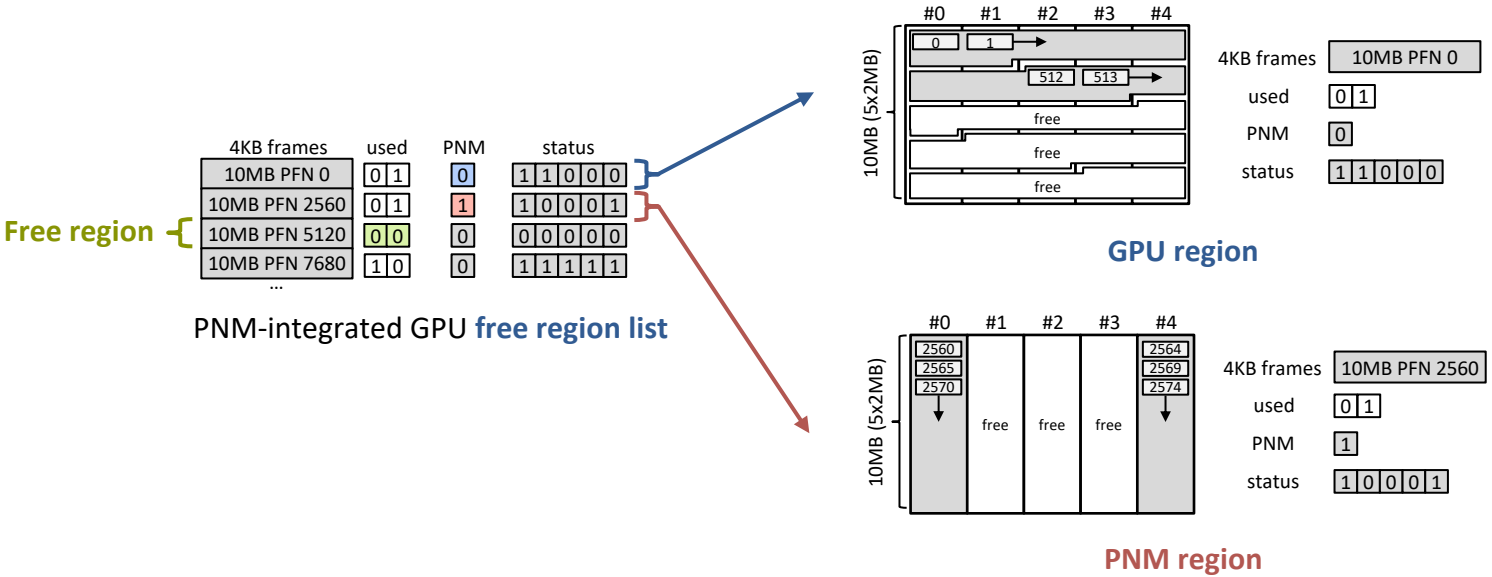
(b) Subpage PFN assignment for PNM page



Stack-selection bits are aligned with **subpage PFN LSB bits.**

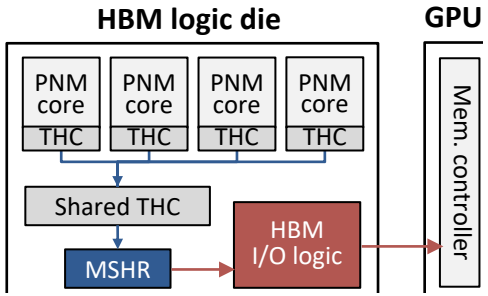
DUALign: Memory management

- The GPU free-page list is extended to **support stack-aware subpage placement.**



DUALign: Page table walk

- Translation cache misses are forwarded to MSHR, and the HBM I/O logic signals GPU on allocation of new MSHR entry.



THC misses are forwarded to MSHRs

↓

HBM logic die signals the GPU

↓

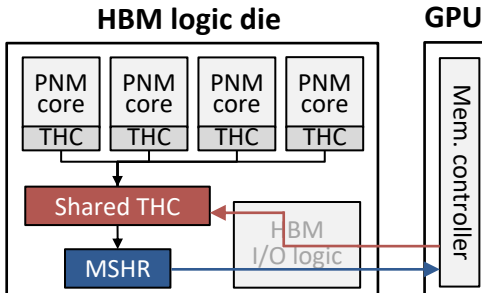
Memory controller reads missed VPN from the MSHR

↓

GPU performs page-table walk and updates THC

DUALign: Page table walk

- GPU reads MSHR entry, executes page table walk, and returns translation results to the shared THC.



THC misses are forwarded to MSHRs

↓

HBM logic die signals the GPU

↓

Memory controller reads missed VPN from the MSHR

↓

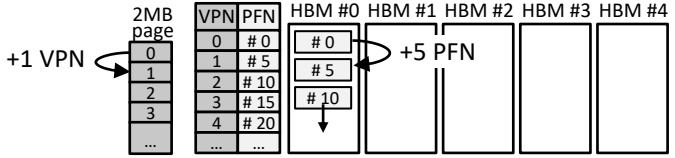
GPU performs page-table walk and updates THC

DUALign: Larger translation cache coverage

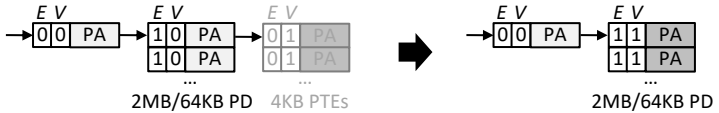
- DUALign **merges GPU subpage PTEs** because subpages within the same GPU page have contiguous VPNs and PFNs.
- **PNM address calculators** derive subpage PFNs from a THC entry for the same PNM page.



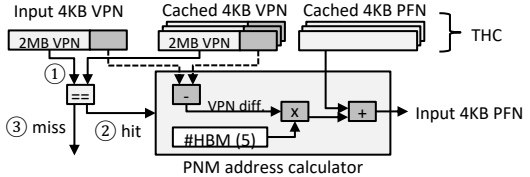
Subpage PFN assignment for GPU page



Subpage PFN assignment for PNM page



Merging GPU PTEs



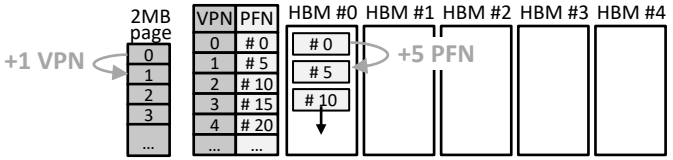
PNM address calculators in THC

DUALign: Larger translation cache coverage

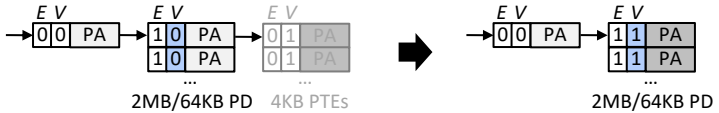
- DUALign **merges GPU subpage PTEs** because subpages within the same GPU page have contiguous VPNs and PFNs.
- **PNM address calculators** derive subpage PFNs from a THC entry for the same PNM page.



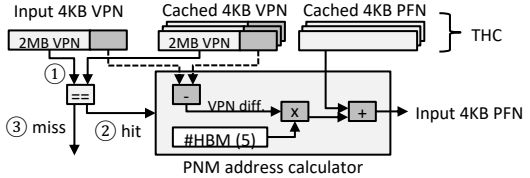
Subpage **PFN assignment for GPU page**



Subpage **PFN assignment for PNM page**



Merging GPU PTEs



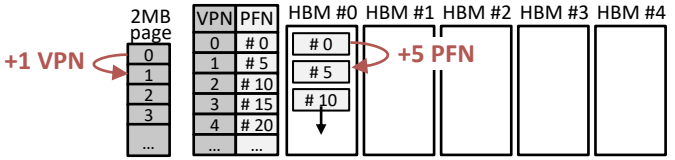
PNM address calculators in THC

DUALign: Larger translation cache coverage

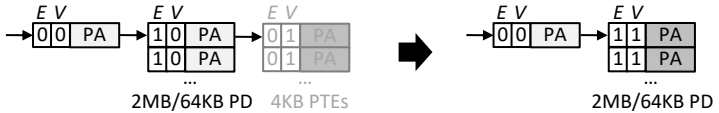
- DUALign **merges GPU subpage PTEs** as subpages from the same GPU page have contiguous VPNs and PFNs.
- **PNM address calculators** derive subpage PFNs from a THC entry for the same PNM page.



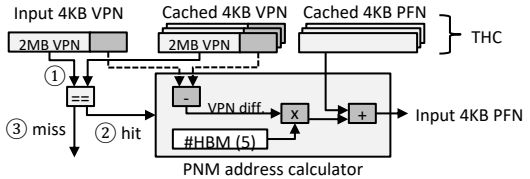
Subpage **PFN assignment for GPU page**



Subpage **PFN assignment for PNM page**



Merging GPU PTEs



PNM address calculators in THC

Evaluation: Experimental setup

- Simulation: Accel-sim, four GPUs
- LLM: Mixtral 8x7B, FP16, attention head partitioning, expert parallelism

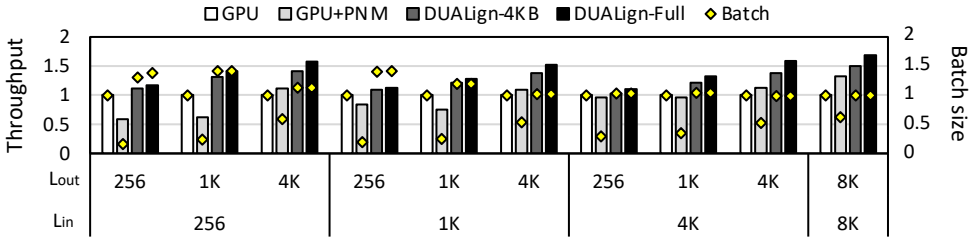
A100 GPU	
Processor	108 SMs, **312 TFLOPs (FP16)** 256KB registers/SM, 192KB L1$/SM 40MB L2$, **80GB HBM2E memory***
Address translator	L1 TLB per 2 SMs, 16 entries L2 TLB per 16 SMs, 2K entries L3 TLB per GPU, 4K entries 32 page-table walkers

***410 GB/s IO bandwidth and 4x internal bandwidth**

PNM	
Processor	16 cores/stack, **8.2 TFLOPs (FP16)** 80KB registers/core, 16KB buffer/core
Address translator	per-core THC, 16 entries shared THC, 512 entries 16 MSHRs per stack

Evaluation: Results

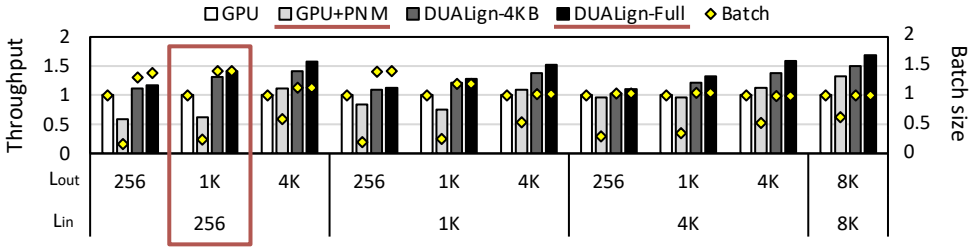
- **Throughput (tokens/s)** under varying input and output token lengths.
- All schemes use the **same memory budget** while maintaining **10 tokens/s rate per user.**



- DUALign **outperforms GPU+PNM** by up to 2.29x with **larger effective memory capacity** provided by LLM serving frameworks and separated memory space.
- DUALign **outperforms baseline GPU** by up to 1.59x by **offloading memory-intensive attention to PNM,** while preserving memory capacity and user concurrency.

Evaluation: Results

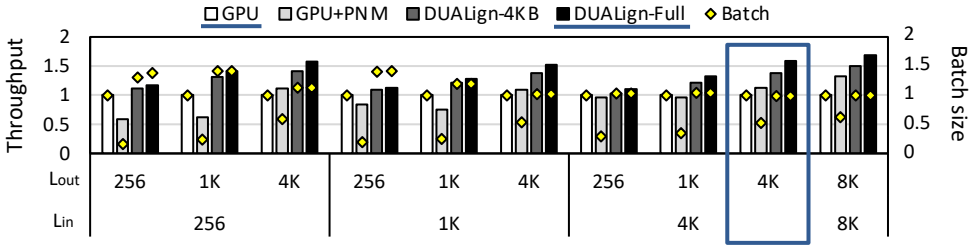
- **Throughput (tokens/s)** under varying input and output token lengths.
- All schemes use the **same memory budget** while maintaining **10 tokens/s rate per user.**



- DUALign **outperforms GPU+PNM** by up to 2.29x with **larger effective memory capacity** provided by LLM serving frameworks and shared GPU-PNM memory space.
- DUALign **outperforms the baseline GPU** by up to 1.59x by **offloading memory-intensive attention to PNM** while preserving memory capacity and user concurrency.

Evaluation: Results

- **Throughput (tokens/s)** under varying input and output token lengths.
- All schemes use the **same memory budget** while maintaining **10 tokens/s rate per user.**



- DUALign **outperforms GPU+PNM** by up to 2.29x with **larger effective memory capacity** provided by LLM serving frameworks and shared GPU-PNM memory space.

- DUALign **outperforms the baseline GPU** by up to 1.59x by **offloading memory-intensive attention to PNM** while preserving memory capacity and user concurrency.

Conclusion

- **PNM units** exploit high internal HBM bandwidth to accelerate memory-intensive attention operation during LLM inference.
- However, **naively integrating** PNM units into existing systems (e.g., GPU) provide limited performance gains due to **hardware and system constraints.**
- DUALign provides lightweight support to **maximize PNM performance while minimizing changes to existing systems.**

Q&A