

Scalable Processing-Near-Memory for 1M-Token LLM Inference: CXL-Enabled KV-Cache Management Beyond GPU Limits

Minjae Lee

Hanyang University

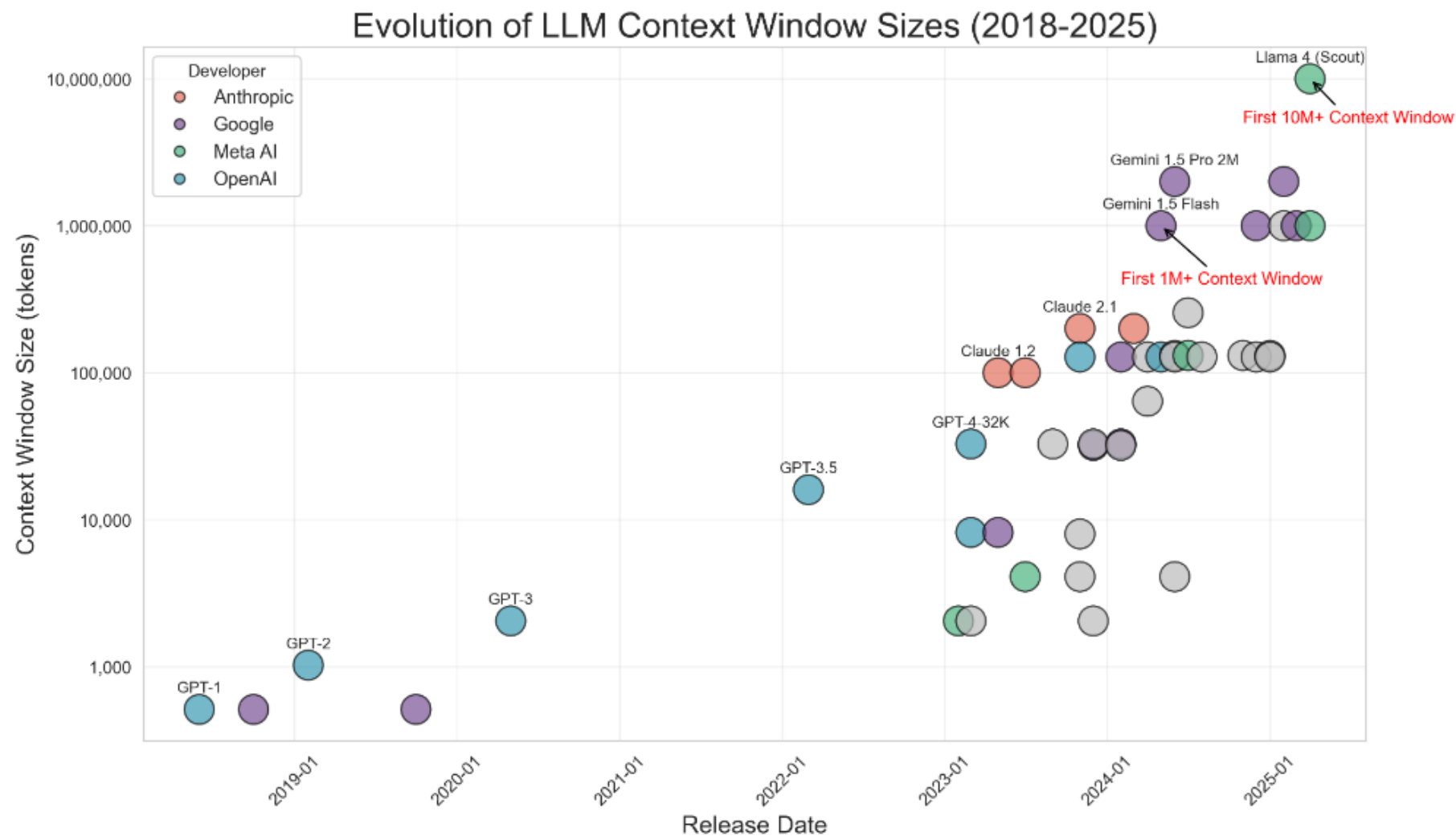
lmj4666@hanyang.ac.kr

2025.10.29

Contents

- Introduction
- KV Cache Management
- GPU+CXL-PNM Heterogeneous System
- Summary

LLM: Context Window Revolution



LLM Applications with Context Window Revolution

Multi-turn Conversation



I'm feeling unmotivated for work and everything.

I'm sorry to hear that! Would you like to share if there is anything particular happening recently?



My paper got rejected from a conference... but I really put a lot of time and efforts in it.



I'm really sorry to hear that your paper got rejected, but it could also be a chance for you to polish your paper and make a bigger deal so that it can shine in a resubmission to the next conference!



Thanks a lot! I feel much better now!



User Satisfaction: 😊

Long Document Analysis

PACT_2025_wikim_mjlee...xl_pnm_kv_ca...1,562 lines

please summarize the key contributions of this technical paper.

Key Contributions Summary

This paper presents a **CXL-enabled Processing-Near-Memory (PNM)** architecture for efficient 1M-token Large Language Model (LLM) inference, addressing critical memory and compute bottlenecks in long-context scenarios.

Main Contributions:

- 1. CXL-PNM Architecture for KV-Cache Management**
 - Offloads token block selection and attention computation directly to PNM accelerators within CXL memory modules
 - Eliminates costly GPU recall operations that plague traditional dynamic selection frameworks
 - Stores full KV-cache in CXL memory, relieving GPU memory pressure and enabling larger batch sizes
- 2. Hybrid Parallelization Strategy**
 - Data Parallelism (DP) for PNM:** Avoids expensive cross-device reductions by having each PNM independently process different batches
 - Tensor Parallelism (TP) for GPU:** Maintains efficient fully-connected (FC) layer computation
- 3. Steady-Token Selection Mechanism (PnG-KV)**
 - Novel GPU-PNM hybrid execution model that balances workload between devices
 - Identifies tokens with persistent relevance and retains them on GPU
 - Maintains large batch sizes ($B \times t$ tokens vs. $B_{\mu} \times T_{\text{budget}}$) while minimizing recall overhead
 - Enables efficient utilization of idle GPU resources during PNM attention processing

Codebase Analysis

CHAT: GITHUB COPILOT

Hi @Mteheran, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

```
using System.Net;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(options =>
{
    options.EnableAnnotations();
    options.SwaggerDoc("v1", new OpenApiInfo
    {
        Version = "1.0.1",
        Title = "API Colombia",
        Description = "Open and free API that contains general information abo",
        TermsOfService = new Uri("https://github.com/Mteheran/api-colombia"),
        Contact = new OpenApiContact
        {
            Name = "Miguel Teheran",
            Url = new Uri("https://mteheran.dev")
        }
    });
});
```

[ERROR] Error: OmniSharp server setting to override the default

Ln 20, Col 70 Spaces: 4 UTF-8 CRLF C# R

Explain code

CHAT: GITHUB COPILOT

Hi @Mteheran, how can I help you?

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve.

```
using System;
namespace api.Models
{
    public class Region
    {
        public int Id { get; set; }
        public string? Name { get; set; }
        public string? Description { get; set; }
        public virtual ICollection<Departamento> Depa
    }
}
```

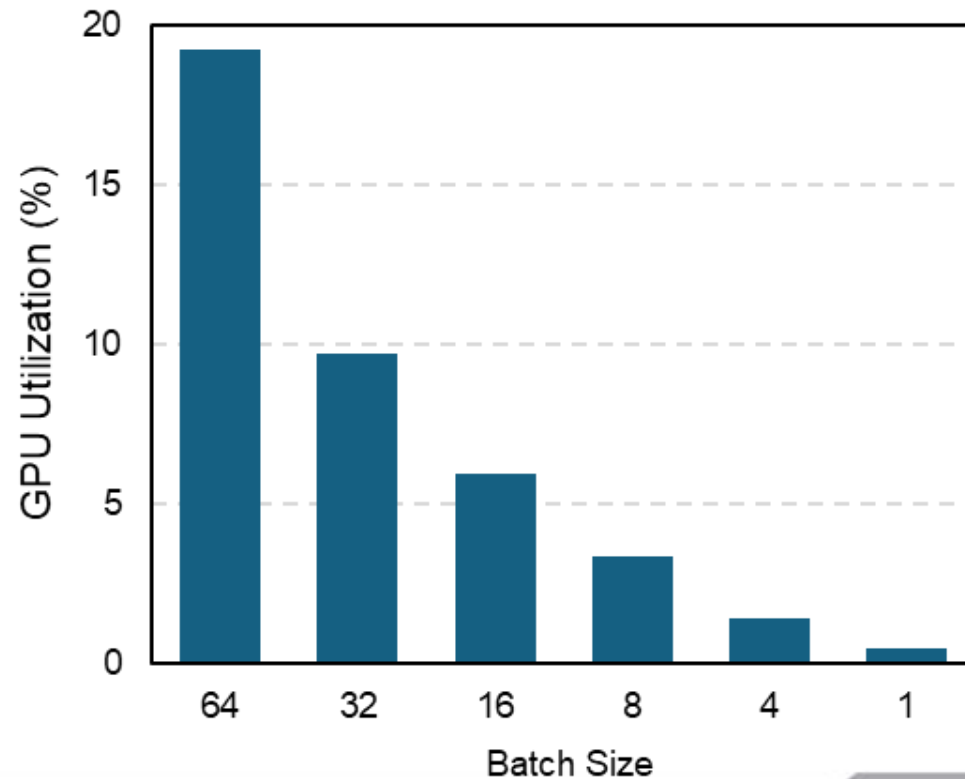
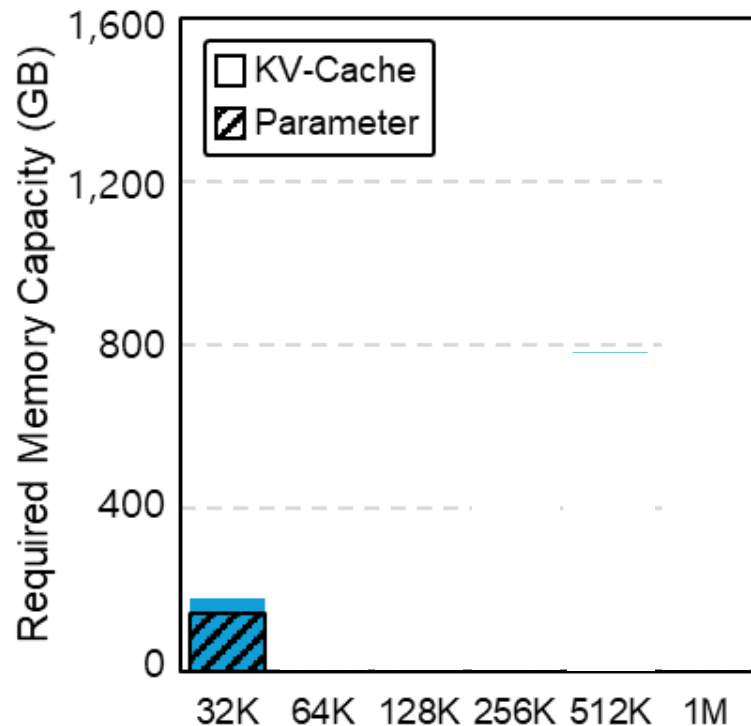
Ln 1, Col 1 Spaces: 4 UTF-8 with BOM CRLF C# R

Generate code

Long Context LLM Inference

- LLM inference's challenge

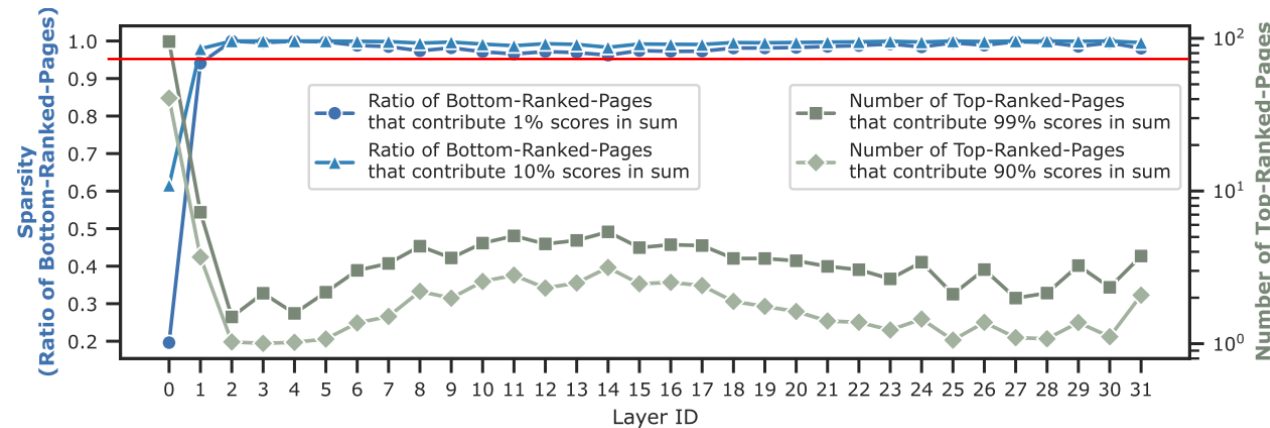
- 1) attention 연산의 복잡도 = $O(N^2)$, N : token length
- 2) **Key/Value cache (KV-cache)를 유지해야 함**
 - Context length가 늘어남에 따라, KV-cache 메모리 사용량이 선형적으로 증가 (model size 고정)
 - 이에 따라 batch size가 작아져 GPU utilization이 떨어지고, GPU capacity를 넘어서기도 함



Sparse Attention for Overcoming Long Context Inference

- Sparse attention using token sparsity

- 대부분 토큰의 $S = \text{softmax}(QK^T)$ (attention score) 값은 전체 값에서 차지하는 비율이 매우 작음
- Token-level sparsity를 살펴보면 상위 1%가 attention score에서 차지하는 비중이 95%를 차지함
- Attention score 합산 99% 비중을 상위 1~2%의 토큰이 차지하고 있음



LongChat-7b-v1.5-32K on LongBench

- KV-cache management

- 1) KV-Cache Compression

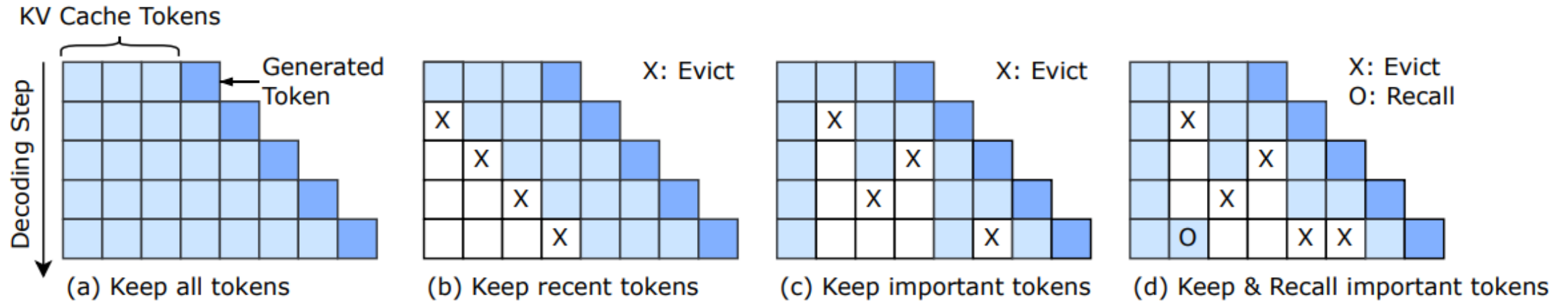
- 사전에 미리 attention score를 예측하여 중요한 KV만 남겨 고정된 KV-cache 사이즈 유지하는 방식 (GPU only)
- 토큰의 KV를 조기에 압축 및 제거하여, 시간흐름에 따른 문맥 변화를 고려하는 데에 한계가 존재

- 2) **KV-Cache Offloading & Dynamic Selection**

- KV를 대용량 외부 메모리를 활용하여 저장한 뒤, 추론 과정에서 쿼리에 따라 attention score가 높은 것을 선택하여 GPU에서 처리하는 방식 (GPU + External memory (CPU))
- 전체 KV를 보존하고 있기 때문에, 시간흐름에 따른 문맥 변화를 고려하기 적합

KV-Cache Offloading & Dynamic Selection

- Comparison with previous KV-cache management works

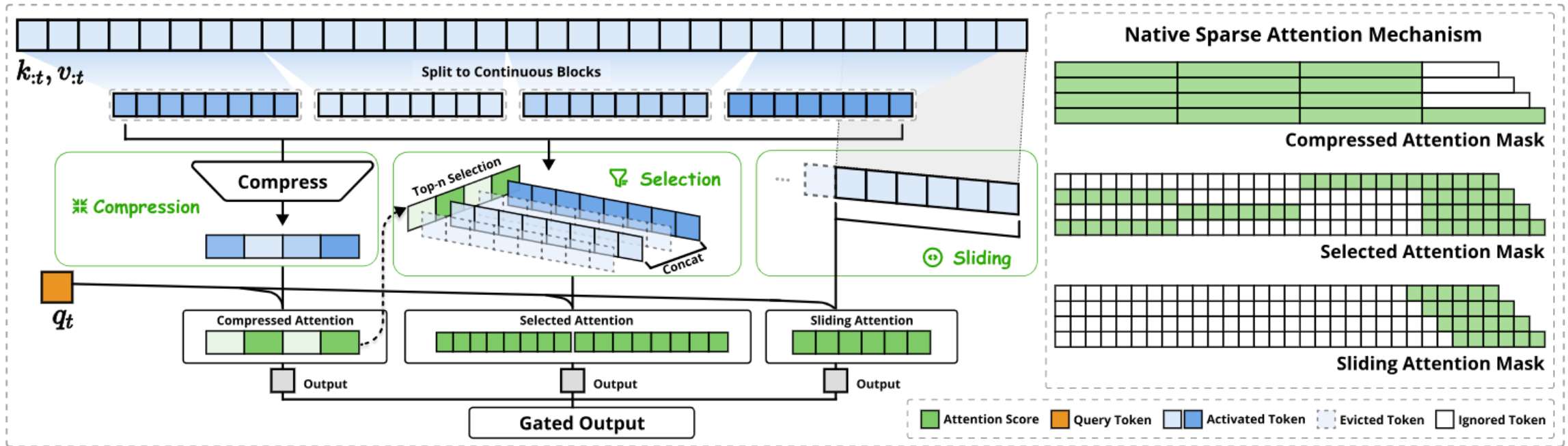


- (a): (Original) Full Attention
- (b), (c): 불필요한 KV는 KV-cache에서 영구적으로 제거
 - Context length와 무관하게 고정된 KV-cache 사이즈 유지
- (d): KV-cache는 대용량의 external memory에 보존
 - Inference 과정에서 매 입력 쿼리에 따라 KV를 선택적으로 사용
 - GPU 메모리 상으로 recall/eviction 됨

GPU 메모리 상으로 고정된 KV만 유지

KV-Cache Offloading & Dynamic Selection Mechanism

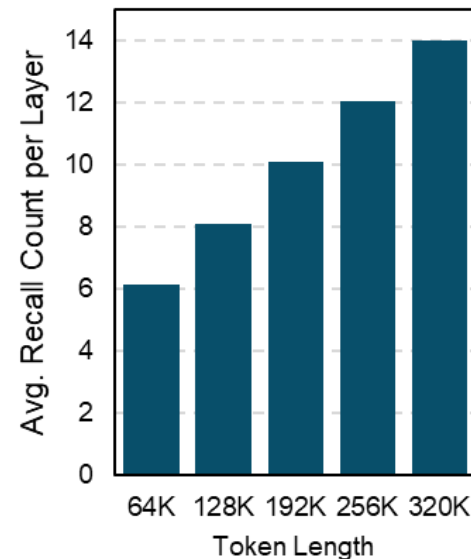
- Native Sparse Attention [arXiv'25] (Deepseek)



- **Compression (summarization)**: 연속된 토큰을 page 단위로 쪼개고 각 page에 대한 대표 값 설정
- **Estimation**: 입력 쿼리에 대한 각 page의 대표 값 사이의 유사도를 평가 (importance 측정)
- **Top-K Selection**: 선택된 page의 KV를 GPU-CPU interface (PCIe)를 통해 전달
- **Sparse Attention**

KV-Cache Offloading & Dynamic Selection Limitation

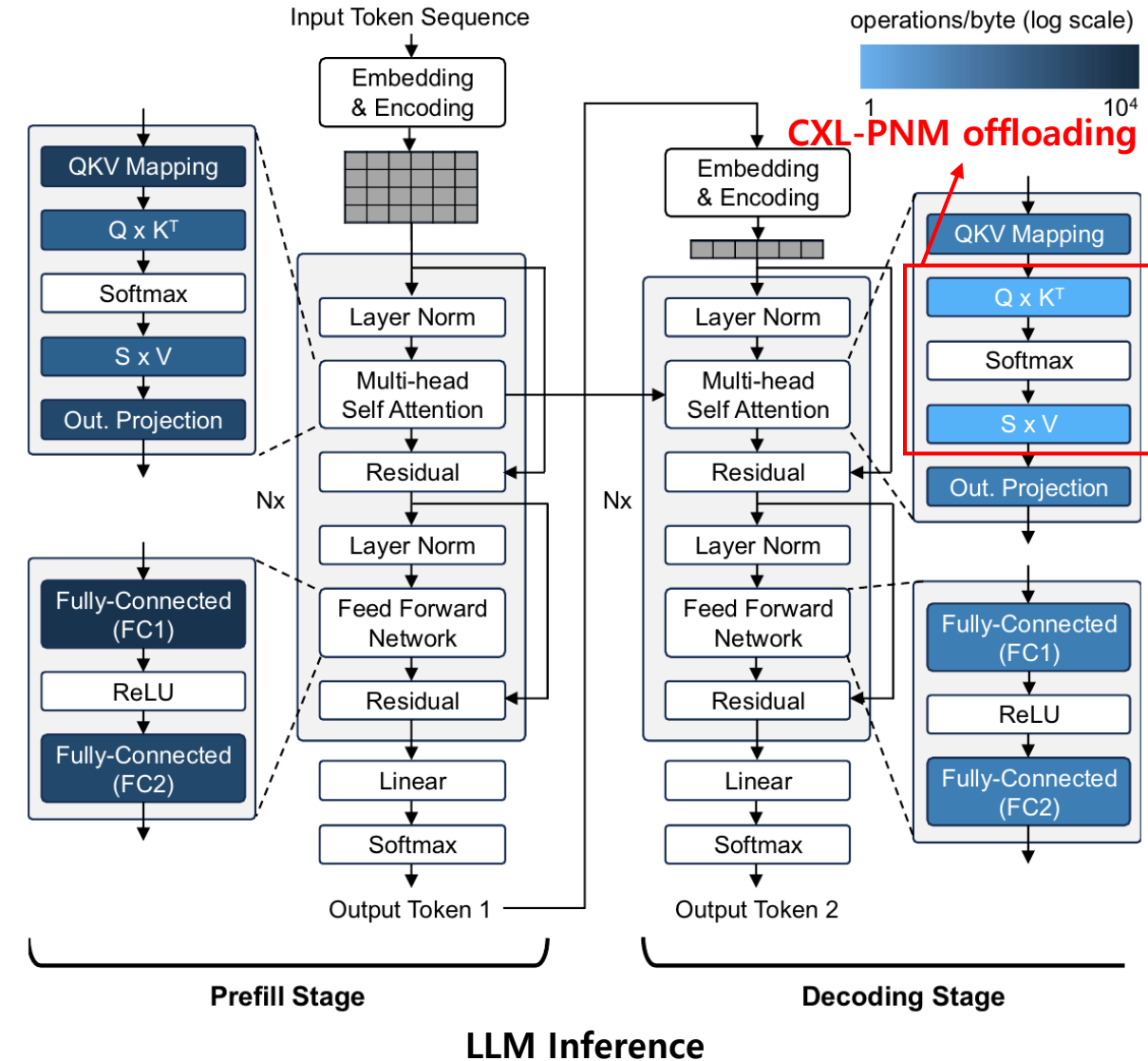
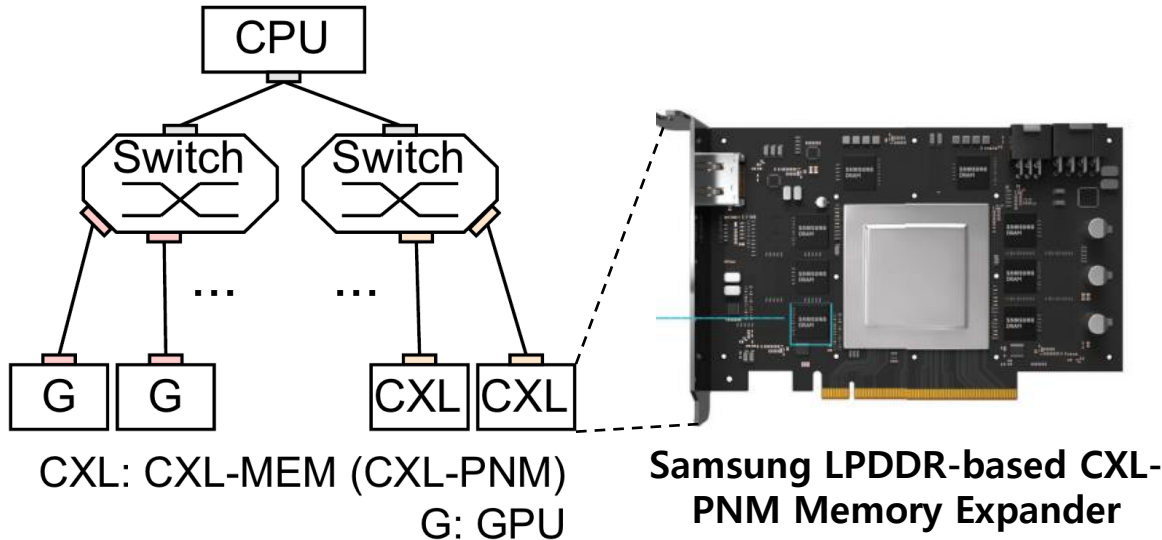
- GPU-CPU bandwidth bottleneck
 - (NVIDIA A100 GPU) HBM2e (internal memory BW) vs. PCIe Gen4 (external memory BW)
 - HBM2e bandwidth: 2 TB/s
 - PCIe Gen4 bandwidth: 64 GB/s
- Recall overhead
 - Context length가 늘어남에 따라 recall도 꾸준히 늘어나, 메모리 전송 병목현상 심화



Heterogeneous System for Long Context LLM Inference

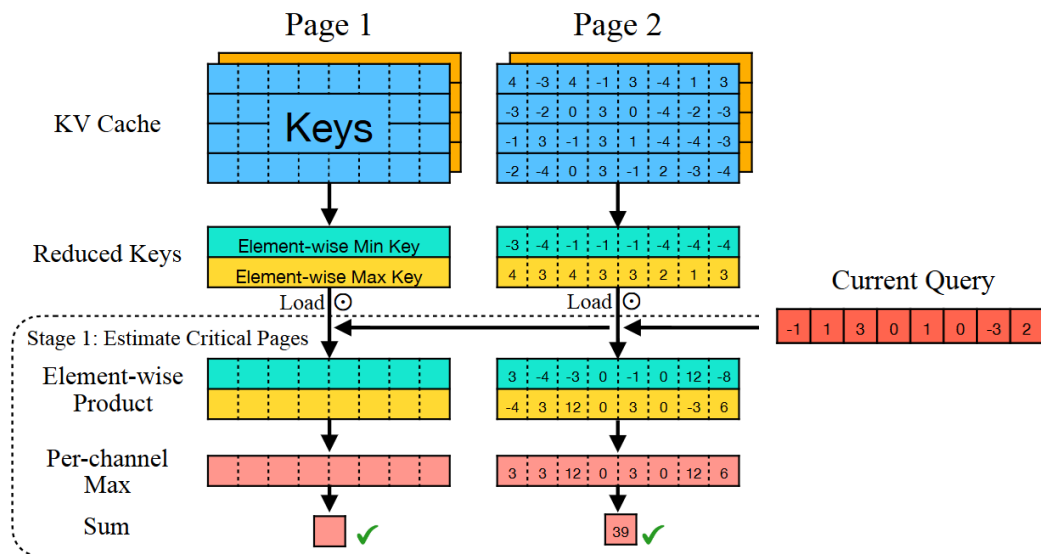
- GPU+CXL-PNM system

- 메모리 용량과 전송 대역폭의 병목을 동시에 해소
- GPU: compute bound 연산
 - Prefill stage
 - Decoding stage's mapping/projection layer 및 FFN
- CXL-PNM: memory bound 연산
 - Decoding stage's QK^T , softmax, SV

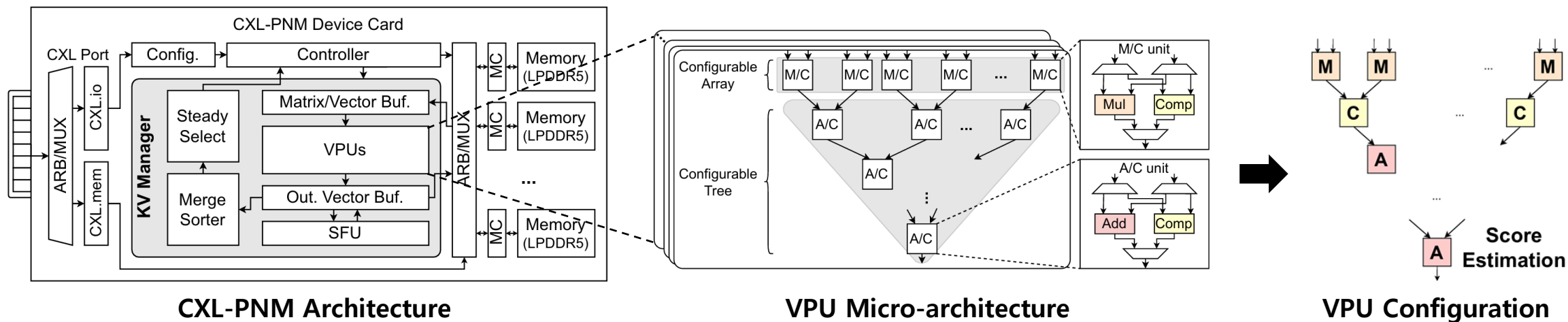


CXL-PNM Platform

- CXL-PNM: offloaded attention 연산을 위한 디바이스
 - **GEMV**: QK^T, SV
 - **Summarization**: $k_n^{max} = \max_i(K[n]), k_n^{min} = \min_i(K[n])$
 - $K \in \mathbb{R}^{N \times 32 \times d_h}$, N : number of pages, d_h : hidden size
 - $K[n] \in \mathbb{R}^{1 \times 32 \times d_h}$, $1 \leq n \leq N$
 - **Estimation**: $R[n] = \sum_{j=0}^{d_h} \max(q \cdot k_n^{max}, q \cdot k_n^{min})$
 - **Ranking**: $R' = \text{Sort}(R)$
 - **Selection**: $\text{Select}(R'[: \text{Top-K}])$



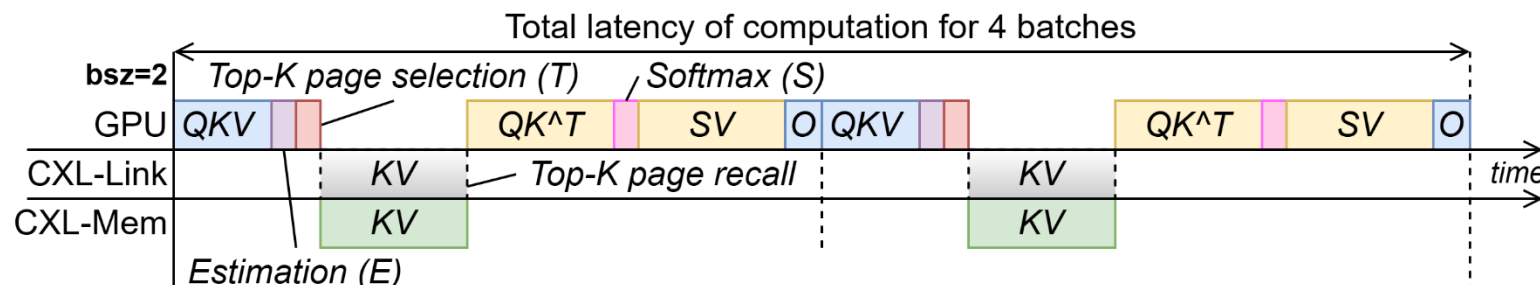
ArkVale[NeurlPS'25]-like KV-cache management



GPU+CXL-Mem (KV-Caching) vs. GPU+CXL-PNM

- GPU+CXL-Memory (like GPU+CPU offloading)
 - 단순 KV-cache 오프로딩 → KV-cache recall로 인한 메모리 전송 병목현상
- GPU+CXL-PNM
 - KV-cache 및 KV-cache management (QK^T , SV 포함) 오프로딩 → Context length와 무관하게 QKV/O 벡터만 전송

GPU-only
KV-Cache Offloading



Low Batch Size
Low Utilization

GPU-PNM
Attention Offloading

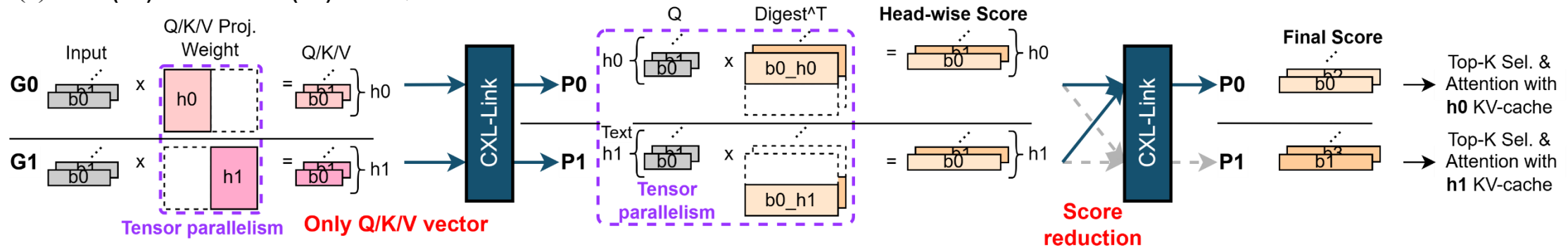
High Batch Size
High Utilization

Parallelism for Multi-GPU/CXL-PNM System

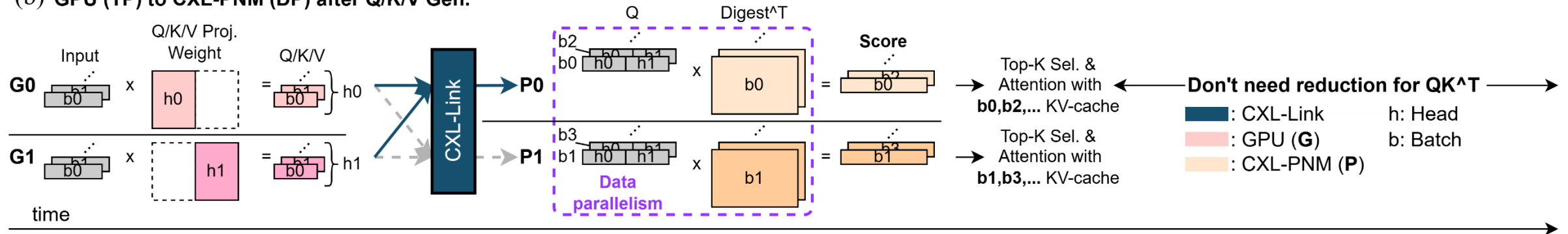
- Parallelism

- GPU: FC 연산을 위해 tensor parallelism (TP)로 병렬화 된 상태를 유지
- PNM: attention 연산에는 data parallelism (DP)를 적용하여 TP에서 필요한 헤드 별 reduction 비용을 피함

(a) GPU (TP) to CXL-PNM (TP) after Q/K/V Gen.



(b) GPU (TP) to CXL-PNM (DP) after Q/K/V Gen.



Hybrid Execution for Attention

- GPU의 유휴 자원을 활용

- Challenge: GPU에서 attention 연산을 처리하기 위해서, recall로 인한 메모리 전송 오버헤드 (GPU-CPU PCIe) 발생
- Solution: 시간이 지나도 지속적으로 관련성이 높은 토큰을 식별하여 GPU 메모리로 로드

Algorithm 1 KV-Cache Replacement ArkVale vs Steady Select

Input: I – page importance scores

Notation: P – on-GPU KV-cache

r – recall KV-cache

e – evict KV-cache

1: **1) Score Ranking:**

2: $S \leftarrow \text{SORT}_{\downarrow}(I)$

3: **2) Page Select:**

4: $\triangleright$ ArkVale: $\triangleright T_{\text{Budget}} \times B_{\mu} \leq |P|$

5: $r \leftarrow S[:K] - P$ $\triangleright$ recall: new Top- K not in P

6: $e \leftarrow (P - S[:K])[-|r|:]$ $\triangleright$ evict: $|r|$ from lowest-score

7: $\triangleright$ Steady-Select: $\triangleright T_{\text{Steady}} \times B \leq |P|$

8: $e \leftarrow P - S[:T_{\text{Budget}}]$ $\triangleright$ evict: not in budget set

9: $r \leftarrow (S[:T_{\text{Budget}}] - P)[:|e|]$ $\triangleright$ recall: $|e|$ top pages not in P

10: **3) Evict & Recall Pages:**

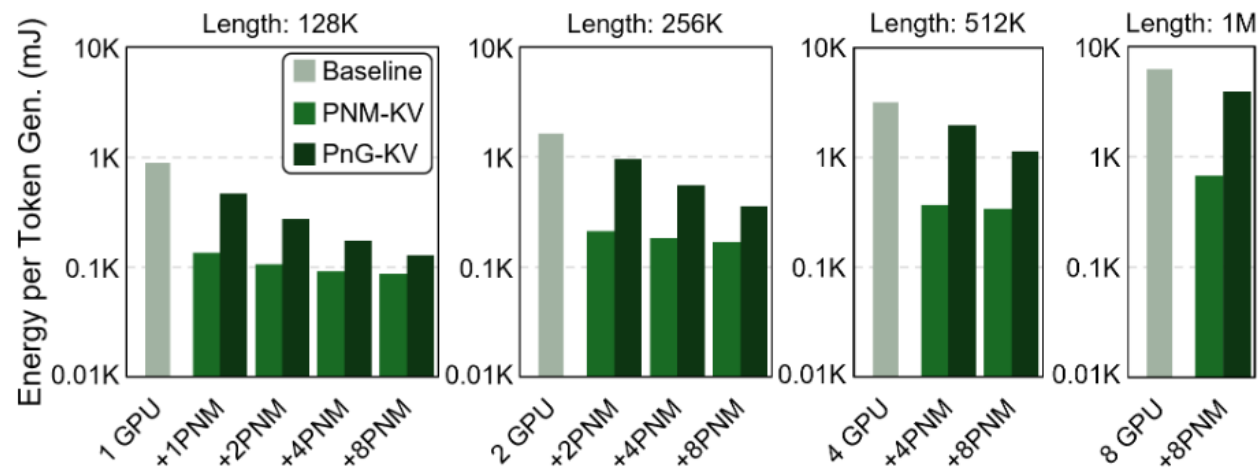
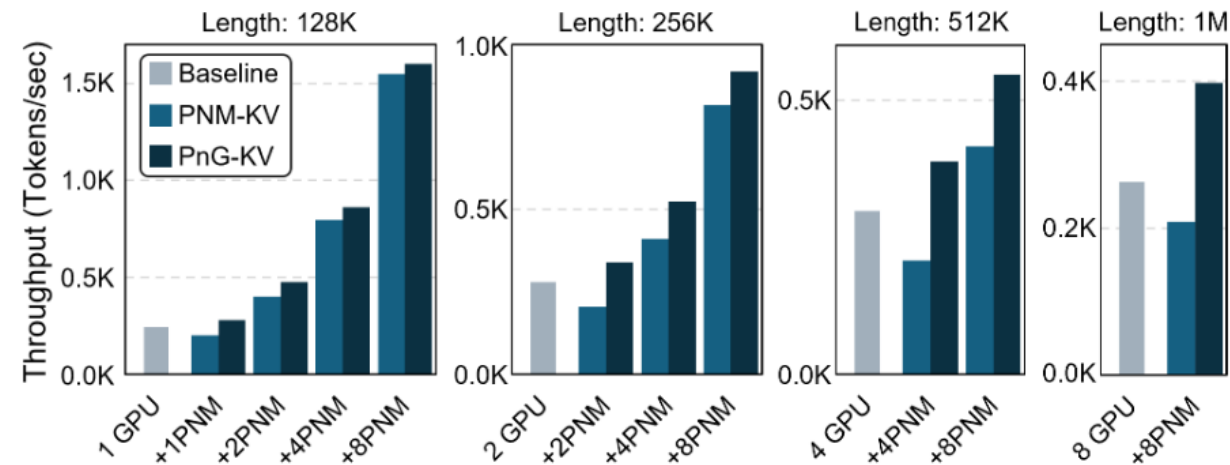
11: $P[e] \leftarrow r$ $\triangleright$ overwrite evicted slots with recalled pages

12: **return** P

GPU+CXL-PNM System Performance

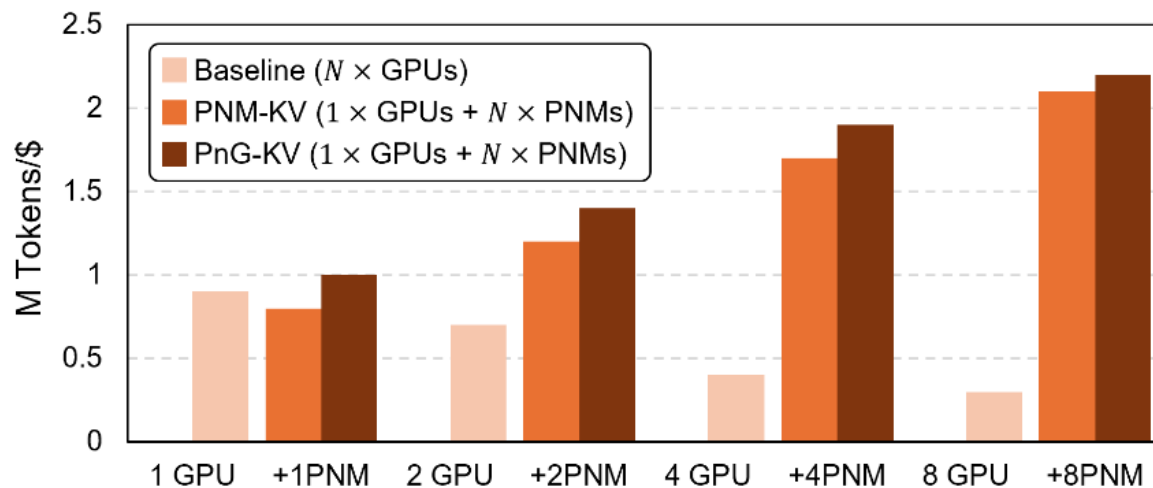
- Evaluation settings

- Baseline: GPU+CXL-Memory Expander
- PNM-KV: GPU+CXL-PNM
- PnG-KV: PNM-KV (+hybrid execution)
- GPU: A100 GPU
 - Compute: (FP16) 312 TFLOPS
 - Memory: (capacity) 80 GB, (bandwidth) 2 TB/s
- CXL-PNM: Samsung LPDDR-based CXL-PNM
 - Compute: (FP16) 8 TFLOPS
 - Memory: (capacity) 512 GB, (bandwidth) 1.1 TB/s
- Model: Llama3.1-8B
- Context length: 128K, 256K, 512K, 1M



Total Cost Efficiency

- GPU scaling vs. PNM scaling



Comparison of total-cost efficiency between GPU scaling and CXL-PNM scaling for Llama3.1-8B, 128K

- GPU scaling
 - GPU 수가 1개에서 8개로 증가할수록 "M Tokens/\$" 값이 급격히 감소
 - 이는 GPU가 고가이며, 긴 컨텍스트 길이에서 메모리 용량 제한으로 인해 처리량 증가가 선형적이지 않아 비용 효율성이 떨어진다는 것을 의미
- PNM scaling
 - 1개의 GPU를 유지하면서 PNM 디바이스를 추가하는 PNM-KV 및 PnG-KV 방식은 PNM 디바이스의 수가 증가할수록 비용 효율성이 증가
 - PNM은 KV-cache management 처리의 비효율성을 개선하여, GPU 뿐만 아니라 전체 시스템의 비용 효율성을 높여 줌

Summary

- LLM의 Long Context 확장은 새로운 가능성을 열었지만, 동시에 메모리 병목현상을 초래함
- KV-cache 관리는 long-context inference의 핵심 병목 지점
- 기존 GPU 중심 구조는 제한된 HBM 용량과 PCIe 대역폭 한계로 scalability 한계 존재
- 앞으로는 더 복잡한 다수의 AI Agent 시스템으로 인해 메모리 병목현상은 더욱 심화 될 것으로 예상되고, 단순 KV-cache 관리를 넘어선 Agent Memory 시스템(memory+PNM)이 필요할 것으로 예상