I/O Analysis is All You Need: An I/O Analysis for Long-Sequence Attention

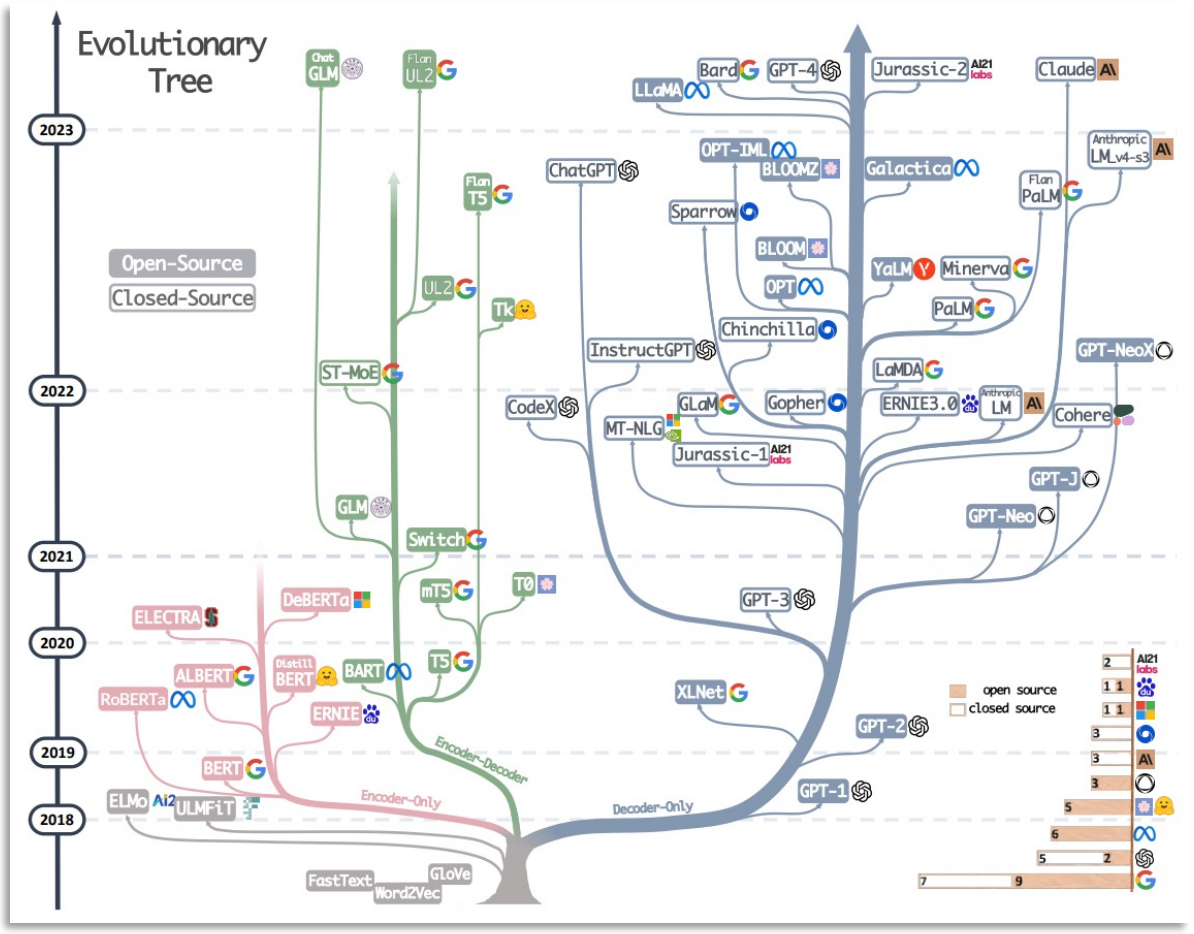
Xiaoyang Lu, Boyu Long
Xiaoming Chen, Yinhe Han, Xian-He Sun

ILLINOIS TECH

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

The Era of Large Language Models



Source: Yang et al., [Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond]

ChatGPT Claude Gemini Grok GitHub Copilot

LLMs increasingly require long-sequence processing

The Era of Large Language Models

ChatGPT

Claude

Gemini

Grok

How do we efficiently scale LLMs to longer sequences?

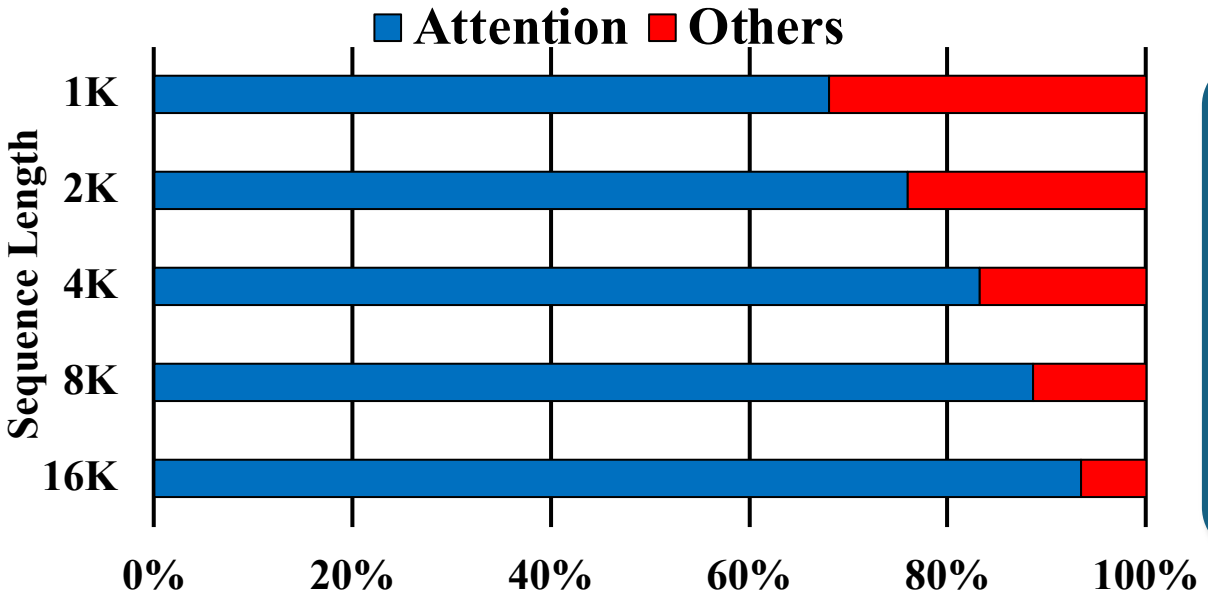
LLMs increasingly require long-sequence processing

Source: Yang et al., Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond

Long-Sequence Attention is the Bottleneck

Attention is the core operation in Transformer-based LLMs

- Inputs: Q, K, V
- $O = \text{Softmax}(QK^T)V$
- **Compute and memory cost scale quadratically with sequence length**



Attention dominates GPT-3 prefilling as sequence length grows.

- **Attention becomes the dominant runtime cost**
- **Naïve execution incurs repeated data movement (I/O) between on-chip memory and off-chip HBM.**

Prior Studies

FlashAttention [NIPS'22]

- Uses block-wise online softmax to avoid materializing the full score matrix
- Reduces HBM traffic by tiling attention into on-chip blocks
- Still relies on **heuristic tiling,** not fully utilized the on-chip memory

FLAT [ASPLOS'23]

- Uses a fused row-granularity dataflow to keep more intermediates on chip
- Avoids additional I/O for softmax, but storing long rows on chip
- With long sequences, row storage can **increase I/O again**

Both prior studies aim to reduce I/O, but neither derives a principled I/O-optimal tiling and scheduling strategy under realistic memory constraints.

I/O Analysis is Needed

I/O analysis provides principled answers to three key questions

- **I/O lower bound:** What is the minimum I/O under a given on-chip memory budget?
- **Optimal tiling:** What tile sizes minimize data movement and maximize on-chip memory utilization?
- **Practical scheduling:** What scheduling strategy realizes this lower bound in practice?

Key Contributions

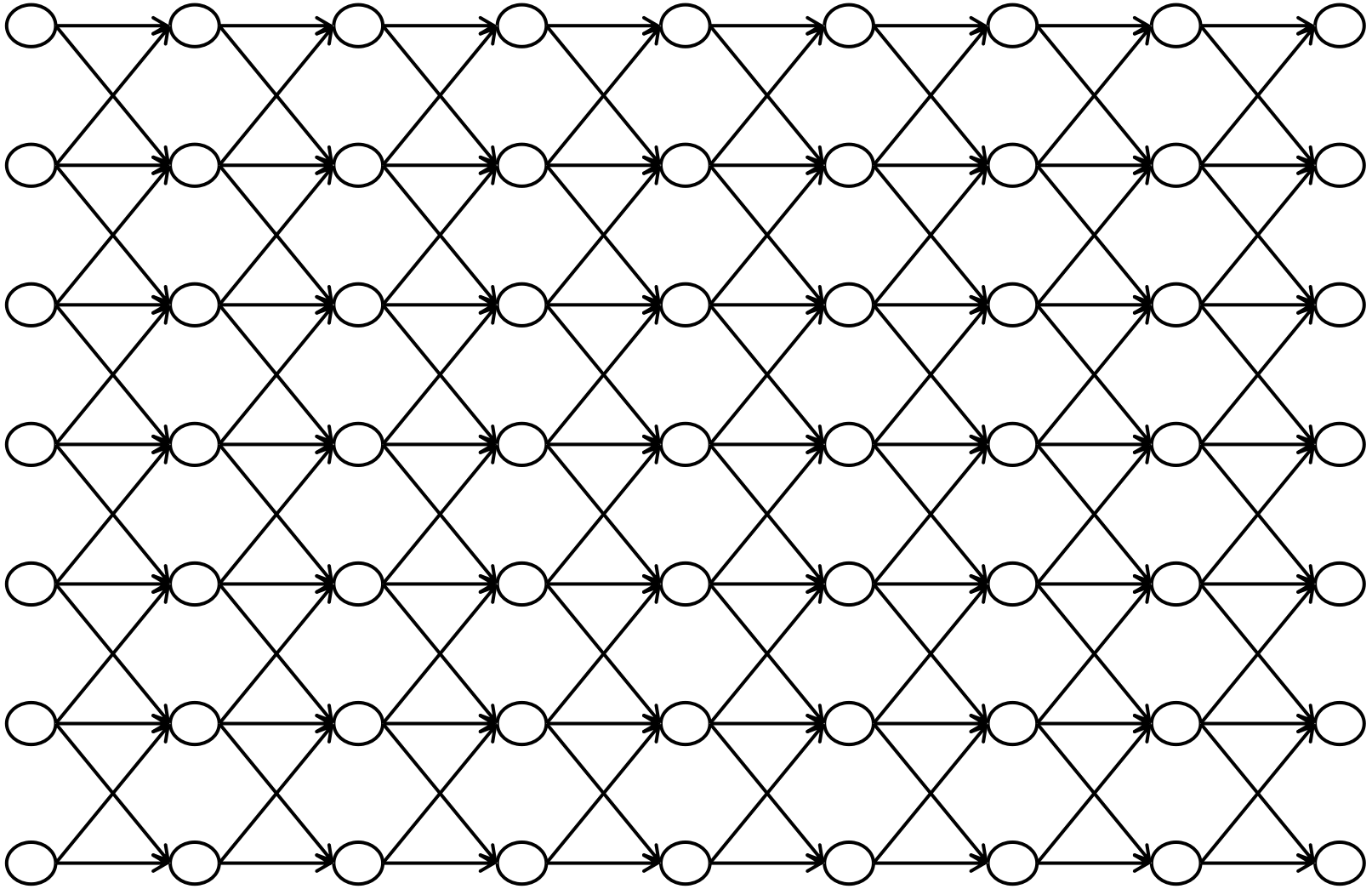
I/O Analysis

- Analyze tall-and-skinny matrix multiplication under realistic on-chip memory constraints

AttenIO Accelerator

- **I/O-Optimal Dataflow:** Derive tiling and scheduling for exact long-sequence attention
- **Three-Level Overlap:** Hide I/O stalls with fine-grained communication-computation overlapping
- **Parallel Softmax:** Exploit parallel patterns for efficient softmax execution

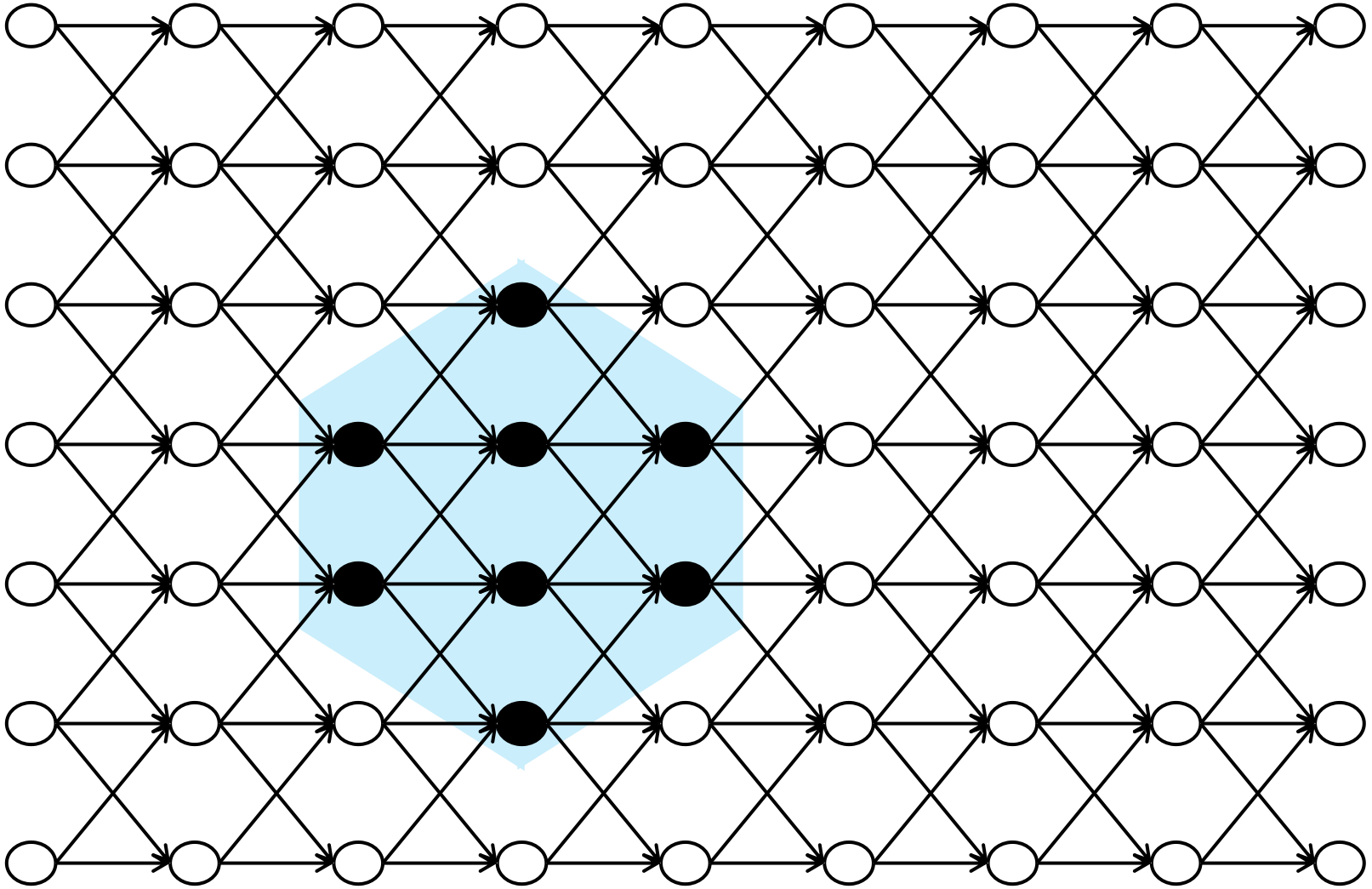
I/O Analysis: Foundation



Red-Blue Pebble Game
[Hong, Kung. 1981]
CDAG abstraction

- Vertex: data entry or intermediate result
- Edge: data dependency

I/O Analysis: Foundation



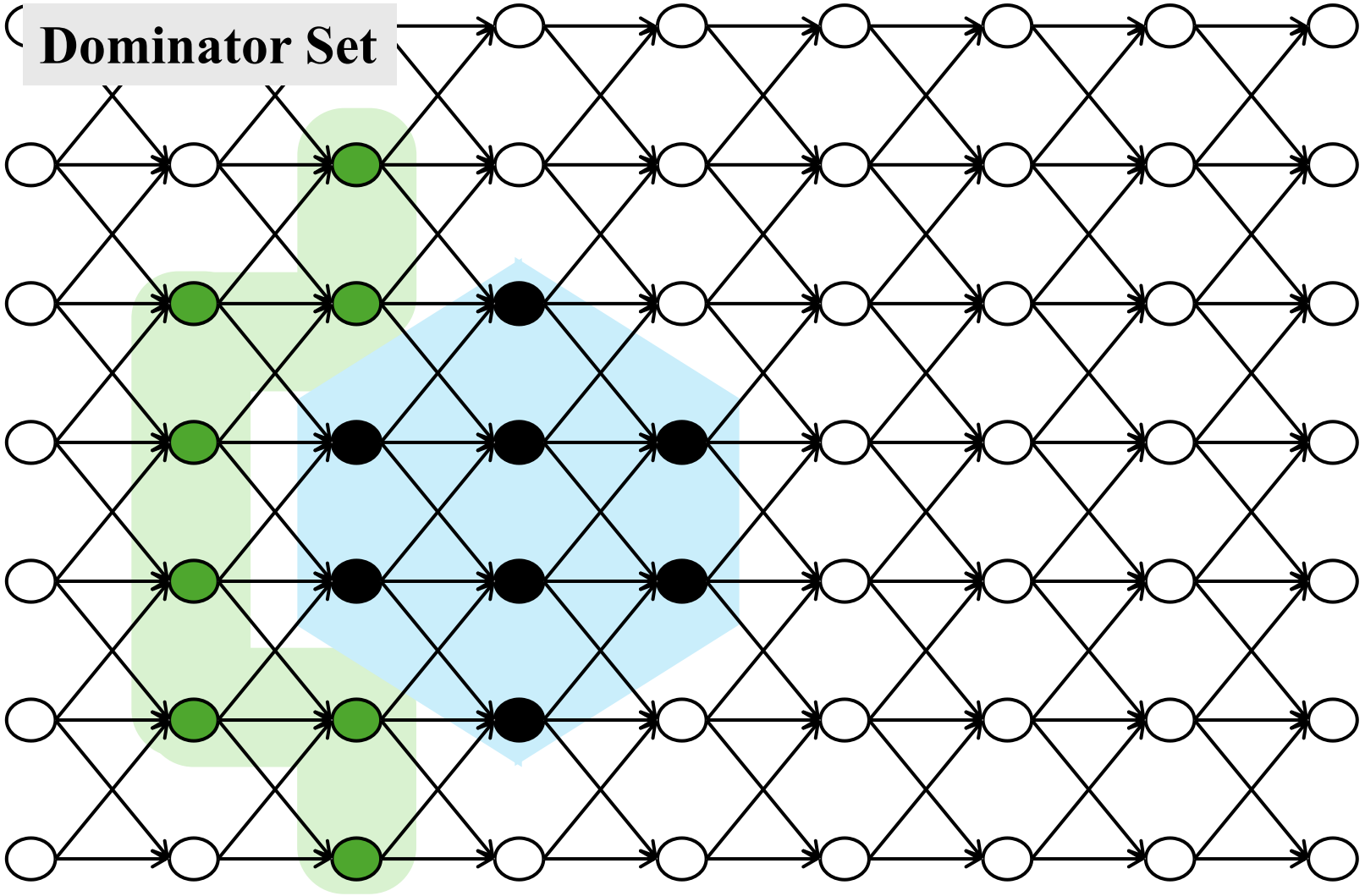
Red-Blue Pebble Game
[Hong, Kung. 1981]
CDAG abstraction

- Vertex: data entry or intermediate result
- Edge: data dependency

Subcomputation

- One local region of CDAG

I/O Analysis: Foundation

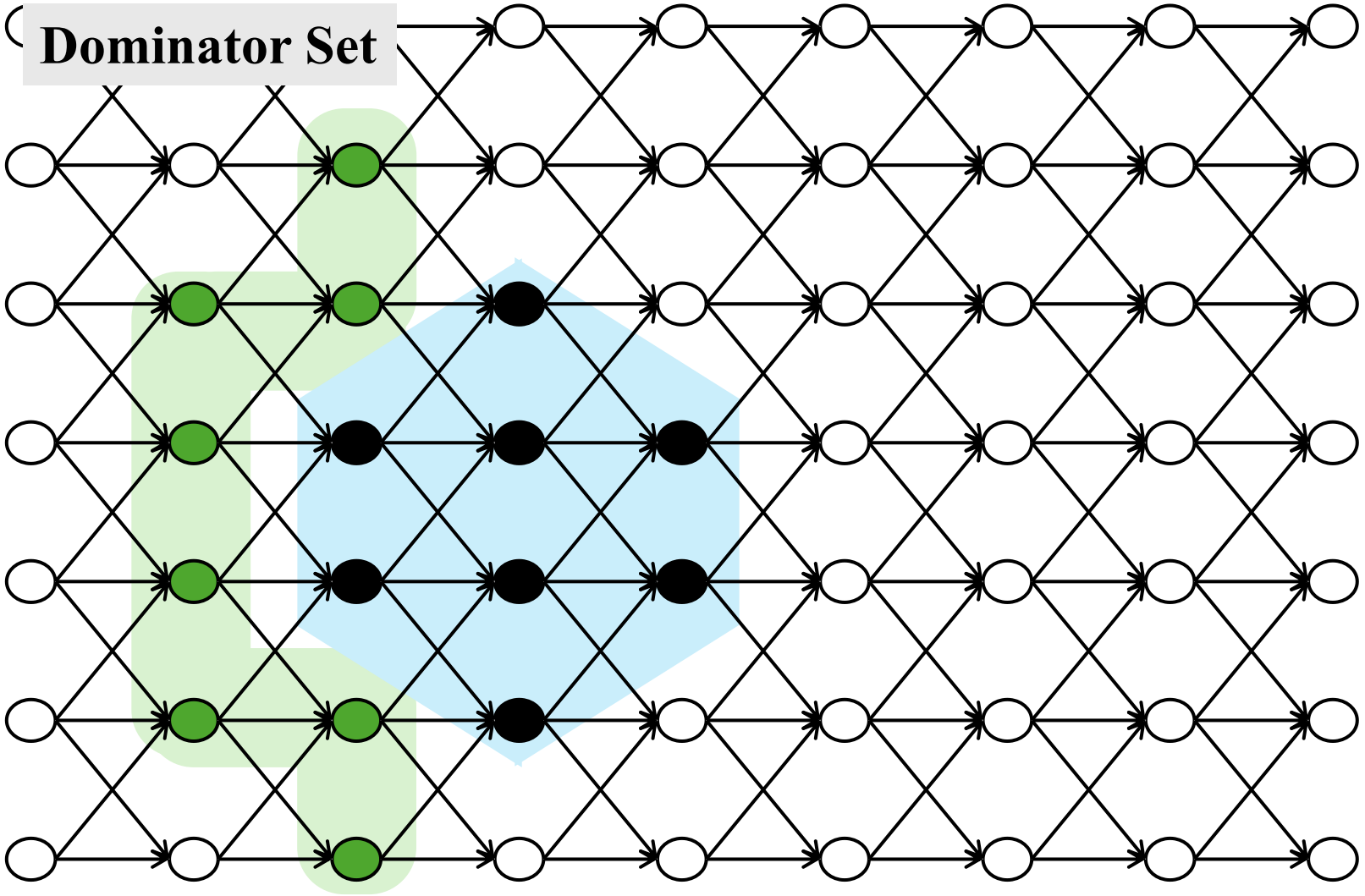


Red-Blue Pebble Game
[Hong, Kung. 1981]
CDAG abstraction
- Vertex: data entry or intermediate result
- Edge: data dependency

Subcomputation
- One local region of CDAG

Dominator set (D_r)
- Minimum inputs needed

I/O Analysis: Foundation



Red-Blue Pebble Game
[Hong, Kung. 1981]

CDAG abstraction

- Vertex: data entry or intermediate result
- Edge: data dependency

Subcomputation

- One local region of CDAG

Dominator set (D_r)

- Minimum inputs needed

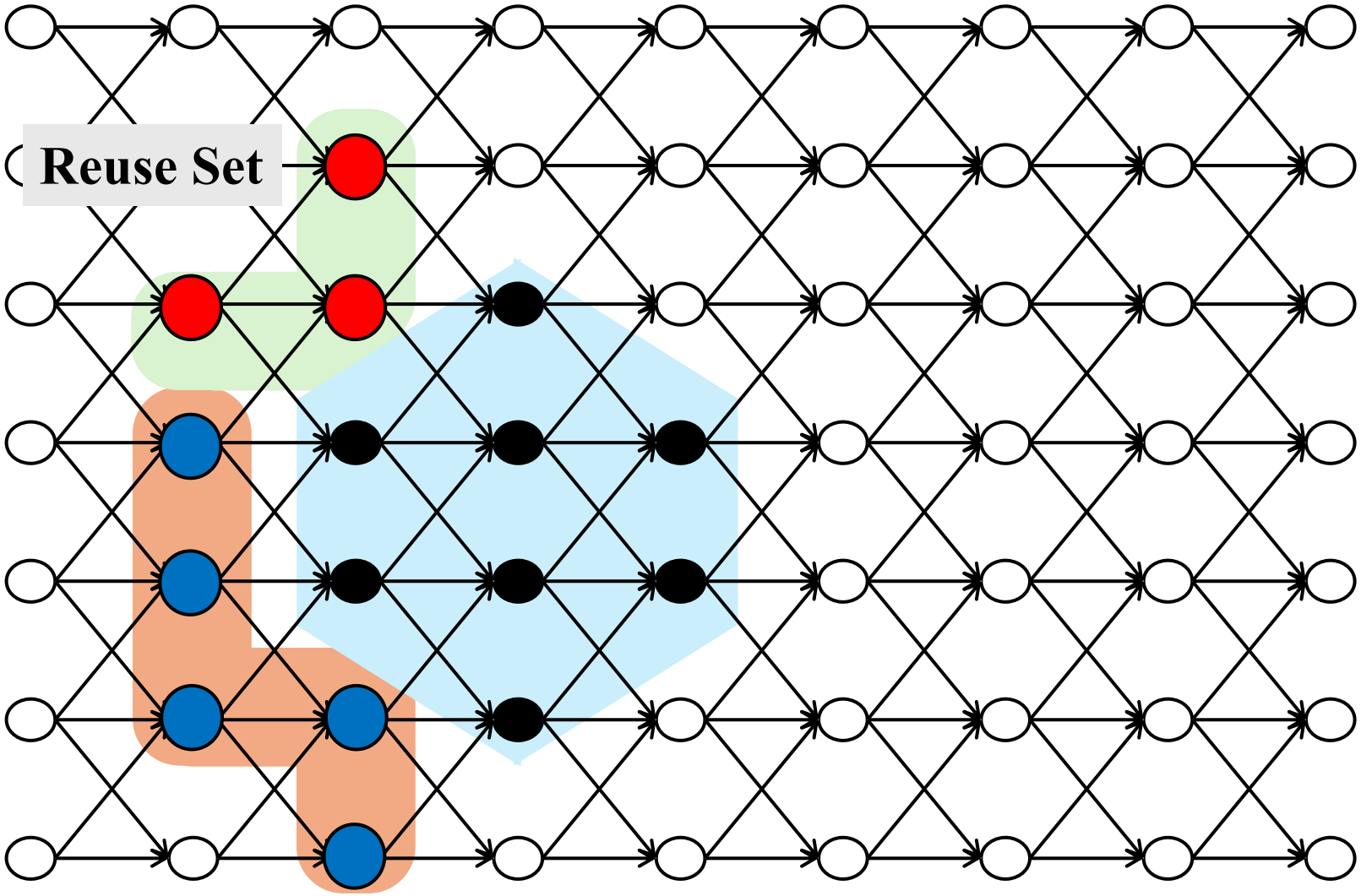
Red Pebble

- Data in fast memory

Blue Pebble

- Data in slow memory

I/O Analysis: Foundation



Red-Blue Pebble Game
[Hong, Kung. 1981]

Red Pebble

- Data in fast memory

Blue Pebble …

- Data in slow memory

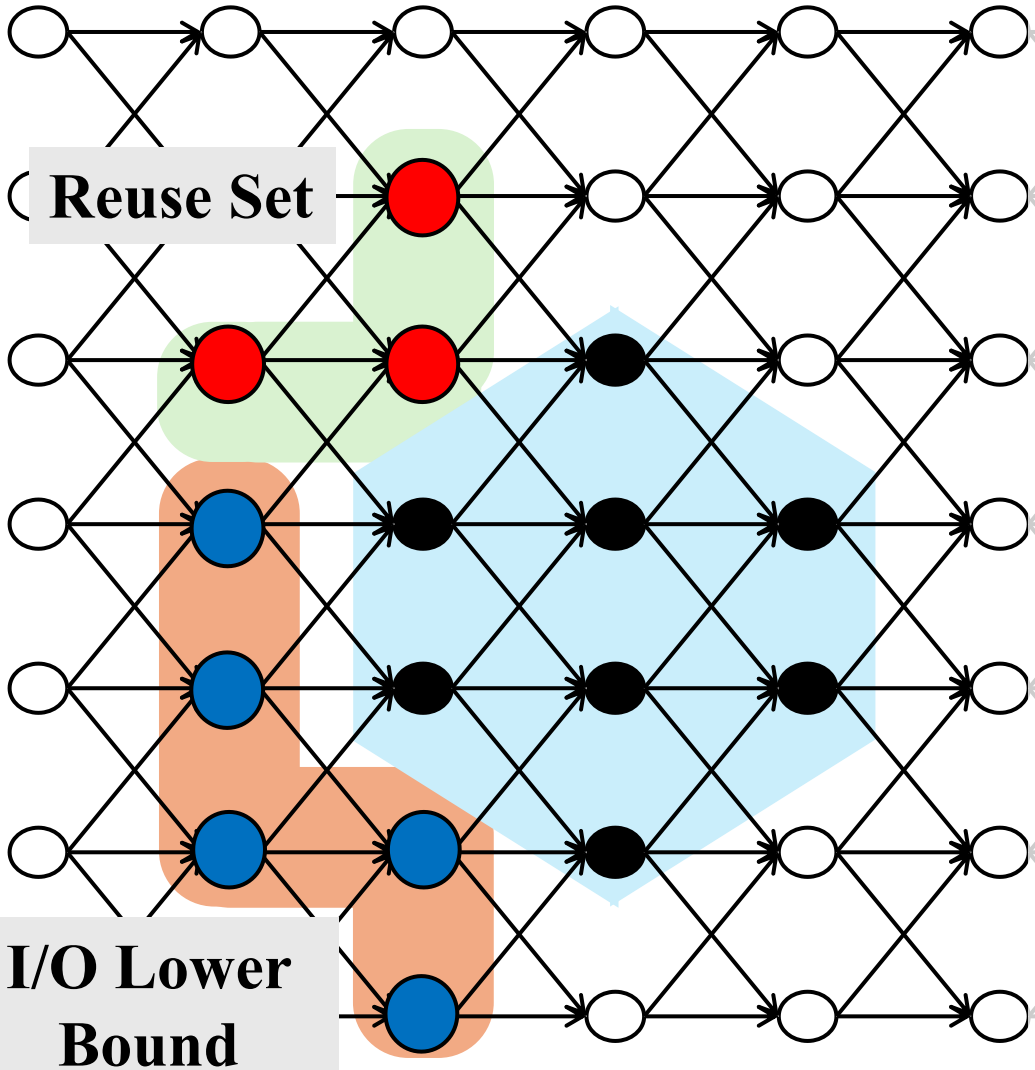
Reuse Set (R_r)

- Data need not be loaded again if they can be reused

Store Set (W_r)

- Data must be stored back to slow memory.

I/O Analysis: Foundation



I/O lower bound

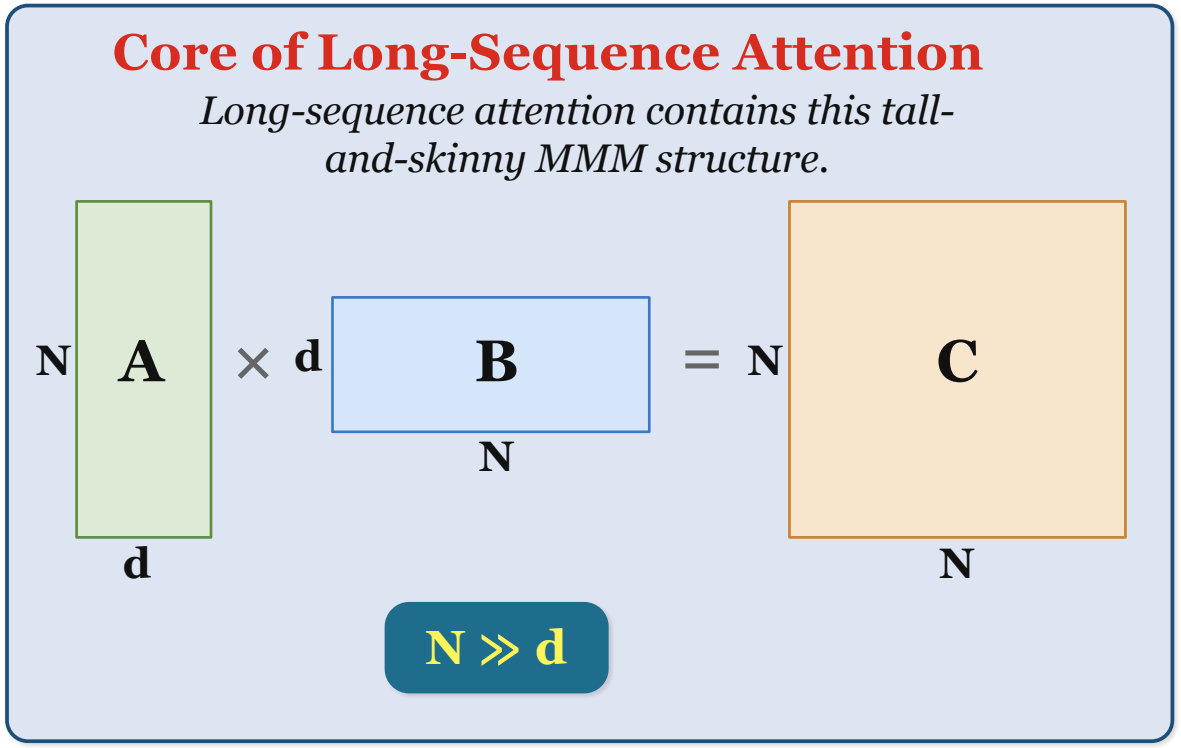
Maximum reuse size

$$T_r \geq |D_r| - |R_r| + |Wr|$$

Dominator set size

Minimum store size

I/O Analysis: Tall-and-Skinny MMM

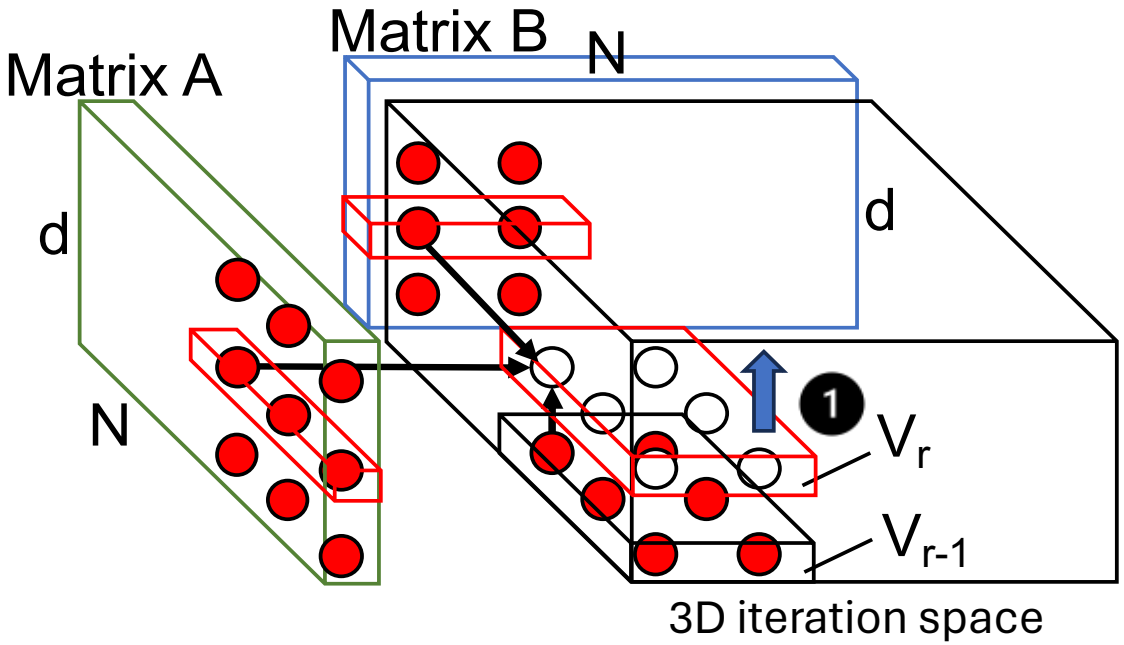


What we analyze

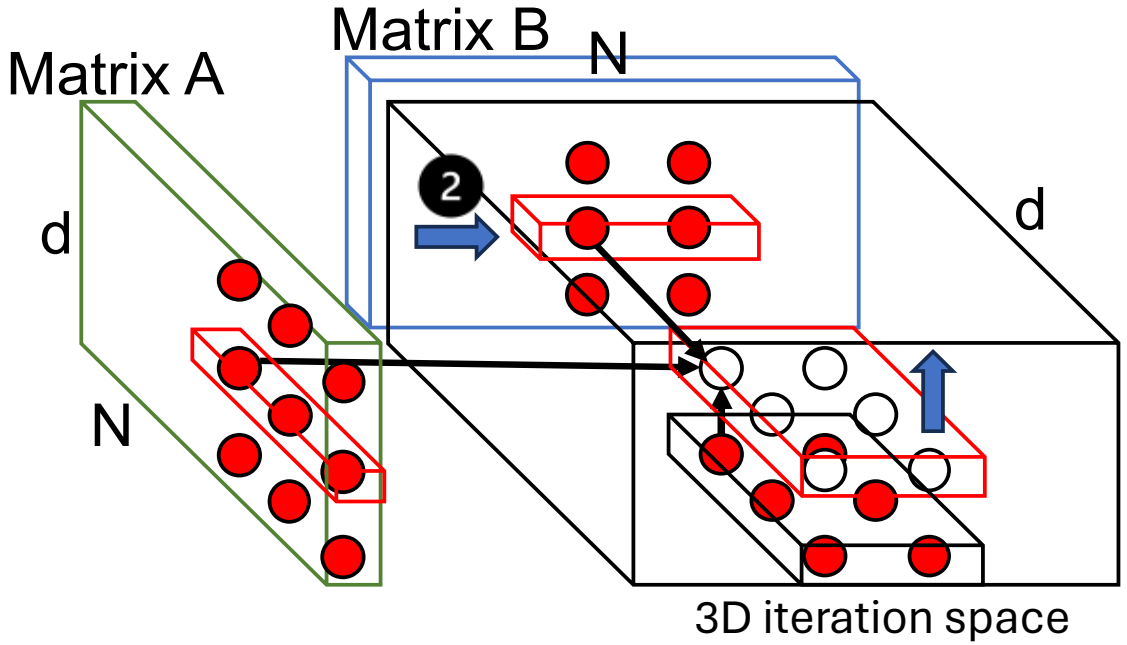
- **Tall-and-Skinny MMM:** C = AB, where N ≫ d, under on-chip memory budget M.
- **Immediate reuse:** keep partial outputs on chip for direct reuse by subsequent subcomputations, avoiding write-back.
- **Future reuse:** keep one input block on chip until it has been fully reused.
- **Objective:** maximize the compute-to-I/O ratio under realistic memory constraints.

I/O Analysis: Tall-and-Skinny MMM

Scheduling and Tiling



Maximize immediate reuse; minimize stores



Keep the A tile on chip for future reuse

Derive tile sizes analytically to maximize the compute-to-I/O ratio
Meaning: the optimal tile shape is derived analytically, not chosen heuristically
Key result: optimal tall-and-skinny MMM I/O scales as $O(N^2 d^2 / M)$

AttenIO: I/O-Optimal Dataflow

Based on the analysis of tall-and-skinny MMM, exact long-sequence attention follows this I/O-optimal schedule:

1. Keep Q_i on chip

Reuse one Q block across the full traversal of K and V

2. Stream K_j blocks

Compute each score block on chip

3. Update online softmax

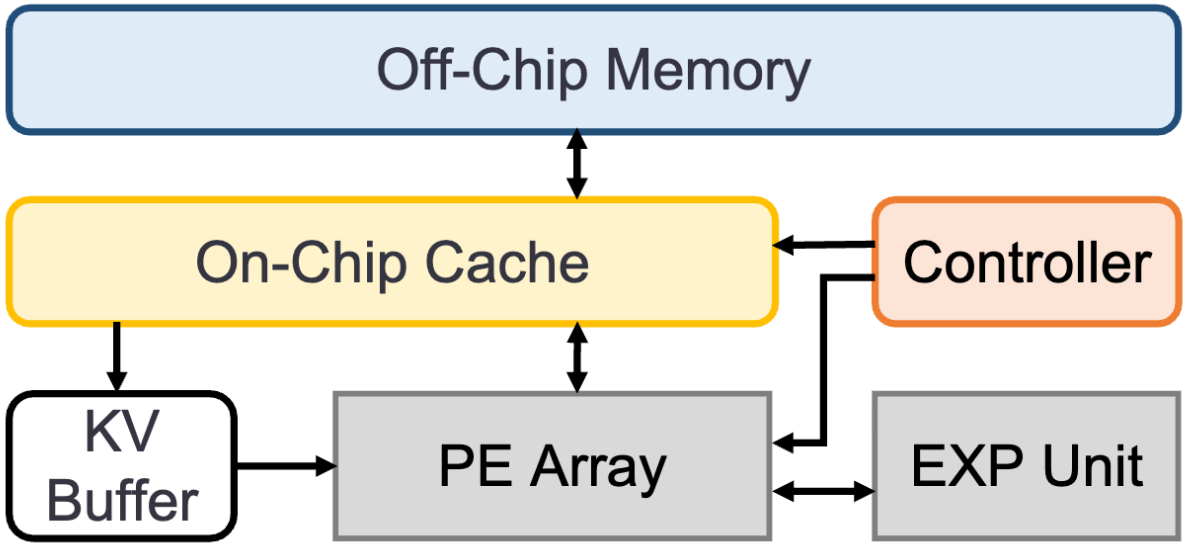
Avoid materializing the full score matrix

4. Stream V_j blocks

Update O_i and keep partial outputs on chip

Tiling: Tile sizes are derived analytically to maximize the compute-to-I/O ratio under on-chip memory constraints.

AttenIO: Architecture

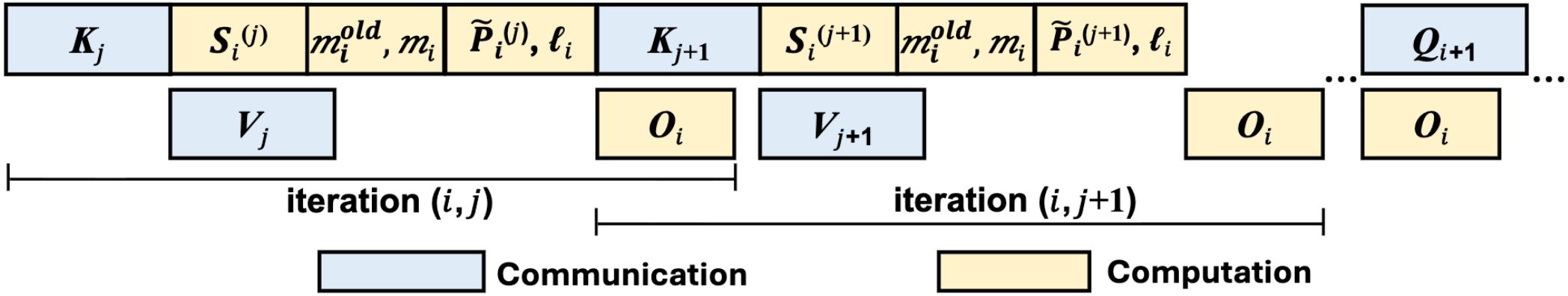


Controller: implements the I/O-optimal dataflow and coordinates data movement and computation

PE Array + EXP Unit: execute matrix operations and softmax-related element-wise operations

KV Buffer + On-Chip Cache: keep reused data on chip and enable communication-computation overlap

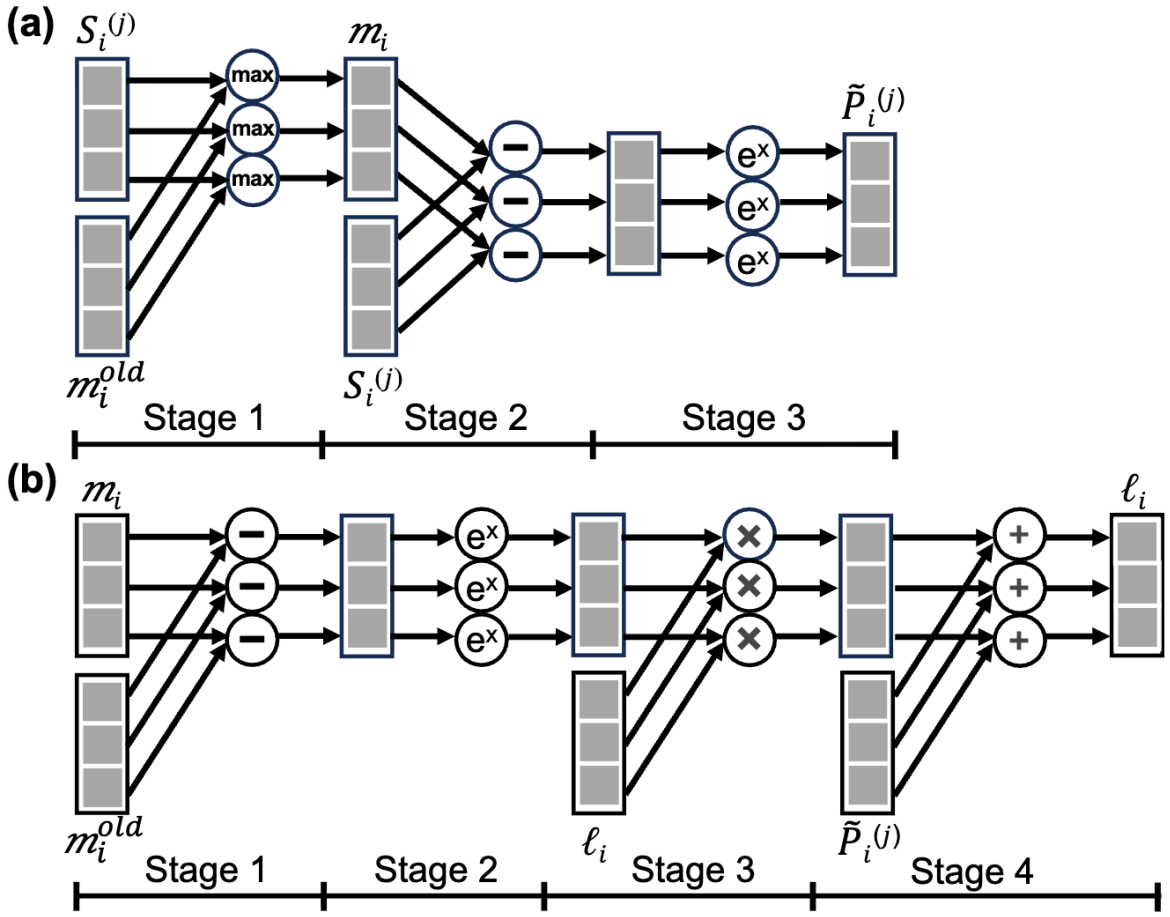
AttenIO: Communication-Computation Overlap



Hide remaining I/O latency through three levels of overlap

Intra-inner-iteration: preload V_j while computing the current score block
Inter-inner-iteration: preload the next K block while updating the current output block
Inter-outer-iteration: preload the next Q block while finalizing the current output

AttenIO: Parallel Softmax with Parallel Patterns



Element-wise softmax:
enabled by I/O-optimal tiling

Pipelined execution:
overlap across stages

Higher utilization:
better use of the PE array and EXP units

Evaluation Methodology

Configuration

Components	Configuration
PE Array	64×32 MACs, 1 GHz
EXP Unit	128 EXP Modules, 1 GHz
KV Buffer	0.25 KB
On-Chip Cache	512 KB
Off-Chip Memory	128 GB/s, 16 64-bit HBM channels, 8GB/s per channel

Baselines

- **Standard:** default exact attention execution flow
- **FLAT [ASPLOS '23]:** fused row-granularity dataflow
- **FlashAttention-2:** block-wise online softmax dataflow

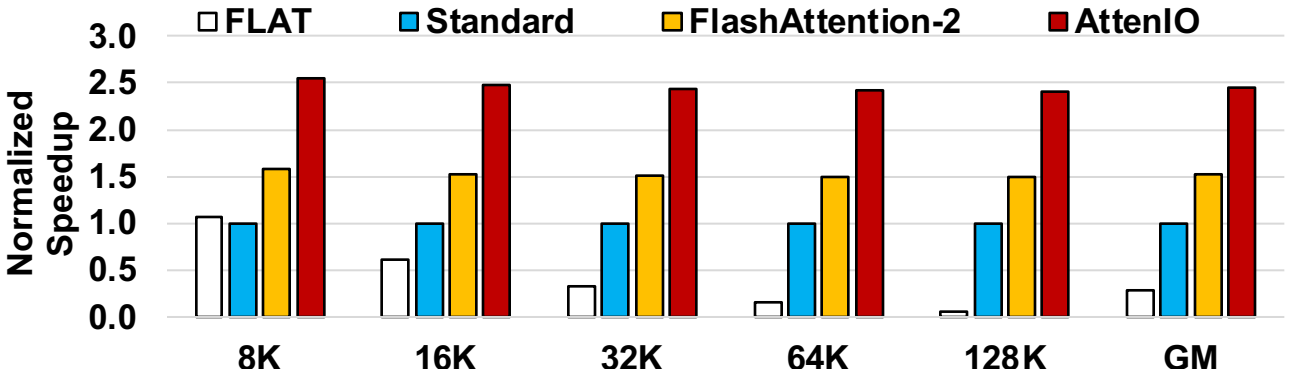
Long-Sequence Attention

- **Sequence lengths:** 8K to 128K
- **Head dimensions:** 64 and 128
- **Precision:** FP16

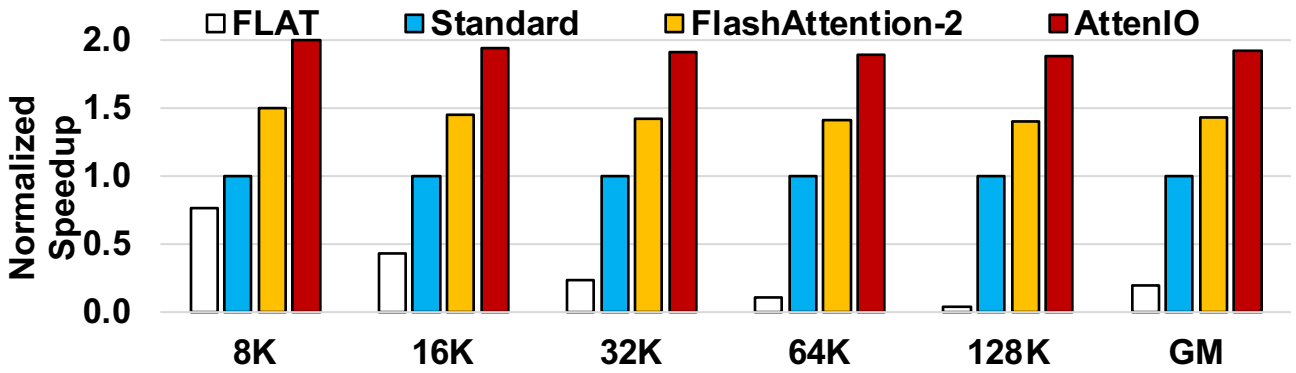
Fairness: all evaluations use the same hardware configuration

Performance

Head dimension 64



Head dimension 128



AttenIO consistently outperforms all baselines across sequence lengths and head dimensions.

Head dimension 64:

8.8× over FLAT

2.5× over Standard

1.6× over FlashAttention-2

Head dimension 128:

9.9× over FLAT

1.9× over Standard

1.3× over FlashAttention-2

More Evaluations in the Paper

Much lower I/O

- substantially less data movement than all baselines
- up to **26.8**× lower data movement than FlashAttention-2

Higher efficiency

- PE utilization reaches up to **90.3%**

Robustness

- consistent gains across cache sizes
- consistent gains with block-wise causal masking

Real impact

- up to **3.0**× over FlashAttention-3 (H100 GPU)

Conclusion

I/O analysis reveals how **to maximize reuse** under realistic **on-chip memory constraints**

I/O analysis provides a principled way to derive **tiling** and **scheduling** for long-sequence attention.

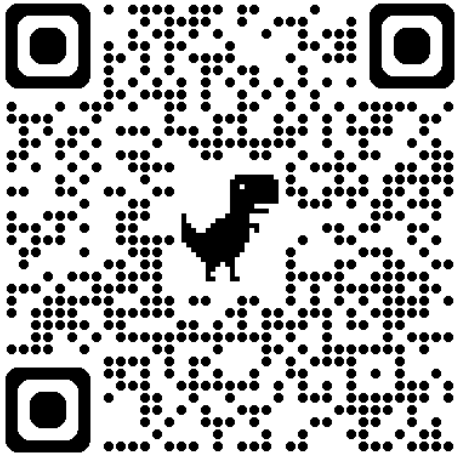
AttenIO translates these insights into an **I/O-optimal dataflow**, **three-level overlap**, and **efficient parallel softmax**.

AttenIO **significantly reduces I/O** and **consistently outperforms** prior exact-attention baselines.

I/O Analysis is All You Need: An I/O Analysis for Long-Sequence Attention

Xiaoyang Lu, Boyu Long, Xiaoming Chen, Yinhe Han, Xian-He Sun

xlu40@illinoistech.edu



ILLINOIS TECH

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

AttenIO: I/O-Optimal Dataflow

Algorithm 1 I/O-Optimal Forward Pass Dataflow for Long-Sequence Attention

Matrices $Q, K, V \in \mathbb{R}^{N\times d}$ in off-chip slow memory, on-chip fast memory of size M.

1: Set block sizes $\mathrm{a} = \left\lfloor \frac{M-d}{2d+4} \right\rfloor$, $\mathrm{b} = 1$.

2: Divide Q into $x = \left\lceil \frac{N}{\mathrm{a}} \right\rceil$ blocks, of size $\mathrm{a} \times d$ each.

3: Divide K, V into $y = \left\lceil \frac{N}{\mathrm{b}} \right\rceil$ blocks, of size $\mathrm{b} \times d$ each.

4: **for** $0 \le i < x$ **do**

5: Initialize $O_i = (0) \in \mathbb{R}^{\mathrm{a}\times d}$ and ℓ_i , $m_i = (0), (-\infty) \in \mathbb{R}^{\mathrm{a}}$ on-chip.

6: Load $Q_i \in \mathbb{R}^{\mathrm{a}\times d}$ from off-chip slow memory to on-chip fast memory.

7: **for** $0 \le j < y$ **do**

8: Load $K_j \in \mathbb{R}^{\mathrm{b}\times d}$ from off-chip slow memory to on-chip fast memory.

9: Compute $S_i^{(j)} = Q_i K^T{}_j \in \mathbb{R}^{\mathrm{a}\times \mathrm{b}}$.

10: Update $m_i^{old} = m_i$ and update $m_i = max\ (m_i^{old}, rowmax(S_i^{(j)}))$.

11: Compute $\tilde{P}_i{}^{(j)} = exp(S_i^{(j)} - m_i) \in \mathbb{R}^{\mathrm{a}\times \mathrm{b}}$, $\ell_i = exp(m_i^{old} - m_i)\ell_i + \mathrm{rowsum}(\tilde{P}_i{}^{(j)}) \in \mathbb{R}^{\mathrm{a}}$.

12: Load $V_j \in \mathbb{R}^{\mathrm{b}\times d}$ from off-chip slow memory to on-chip fast memory.

13: Update $O_i = diag\left(exp\left(m_i^{old} - m_i\right)\right)^{-1} O_i + \tilde{P}_i{}^{(j)} V_j \in \mathbb{R}^{\mathrm{a}\times d}$.

14: **end for**

15: Compute $O_i = diag(\ell_i)^{-1} O_i$

16: Store O_i to the off-chip slow memory.

17: **end for**

I/O Analysis is All You Need: An I/O Analysis for Long-Sequence Attention