The 27th IEEE International Symposium on High-Performance Computer Architecture

Layerweaver: Maximizing Resource Utilization of Neural Processing Units via Layer-Wise Scheduling

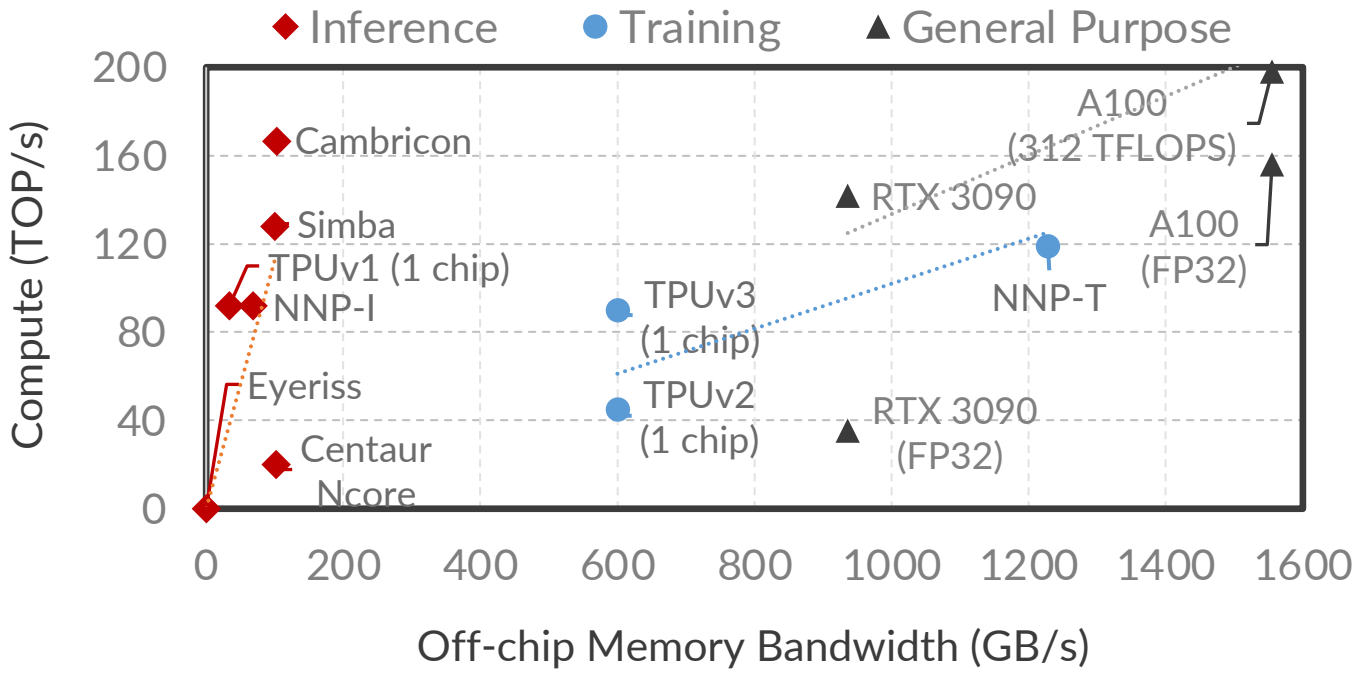
Young H. Oh[†] , Seonghak Kim*, Yunho Jin*, Sam Son*, Jonghyun Bae*
Jongsung Lee*, Yeonhong Park*, Dong Uk Kim*, Tae Jun Ham*, Jae W. Lee*

*Seoul National University

[†]Sungkyunkwan University

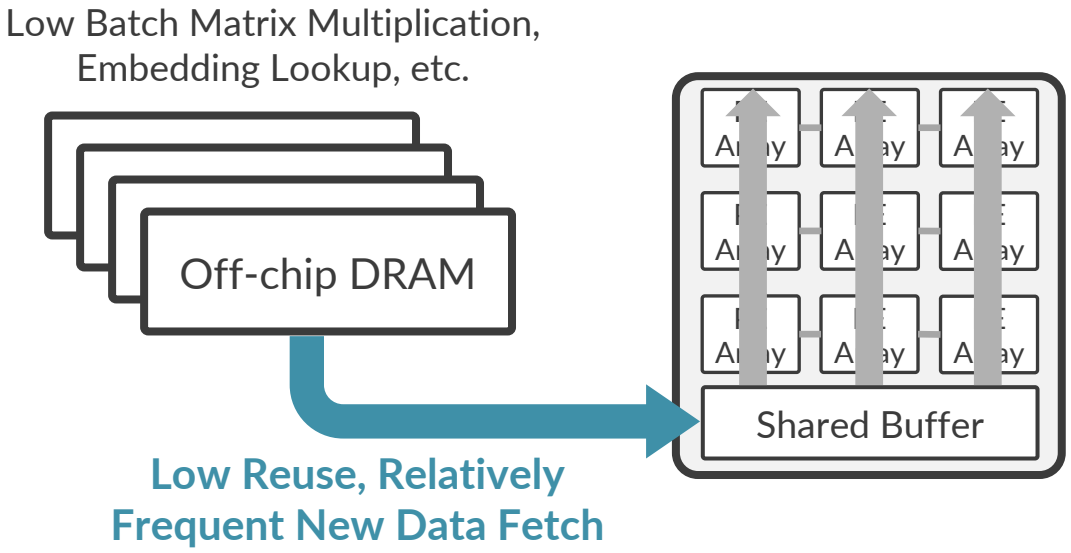
Neural Processing Units for Clouds

- Hardware resources in Neural Processing Units (NPUs)
  - Computing units (TOP/s), On-chip buffers (MB), Off-chip memory bandwidth (GB/s)
- **Compute-to-Memory Bandwidth Ratio** varies widely among NPUs
  - According to target applications (Training or Inference), service scenarios, area budget, etc.

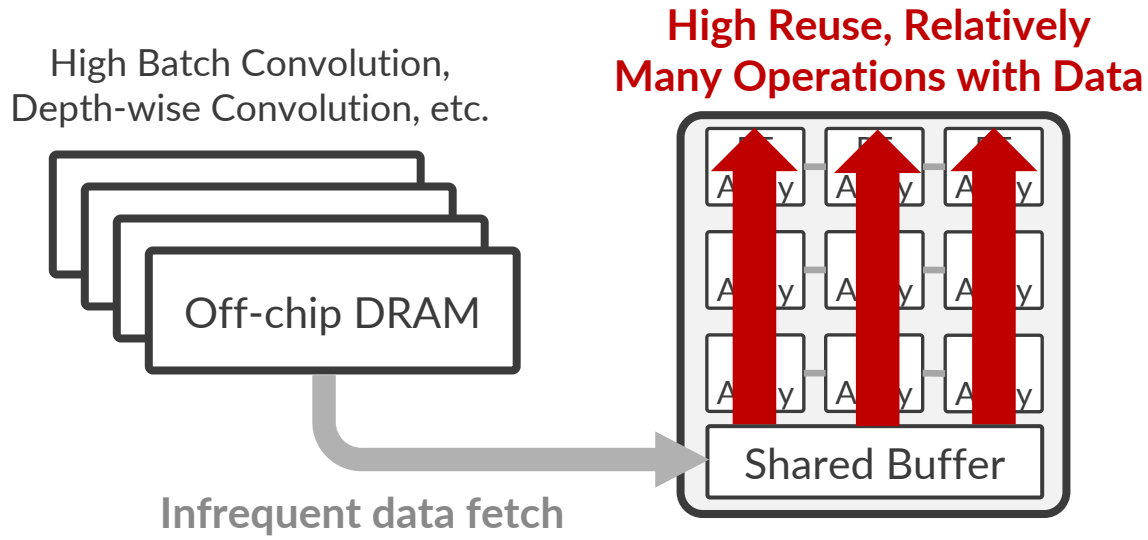


Characteristics of DNN Models

- **Arithmetic Intensity (Ops/Byte):**
  - How many compute operations are performed per byte loaded from off-chip DRAM
- The metric is determined by:
  - Type of Layer: Convolution >> Depth-wise Convolution > Matrix Multiply > Embedding Lookup
  - Batch Size: Large batch > Small batch



Low Arithmetic Intensity



High Arithmetic Intensity

NPU Resource Underutilization Problem

- NPU Capabilities

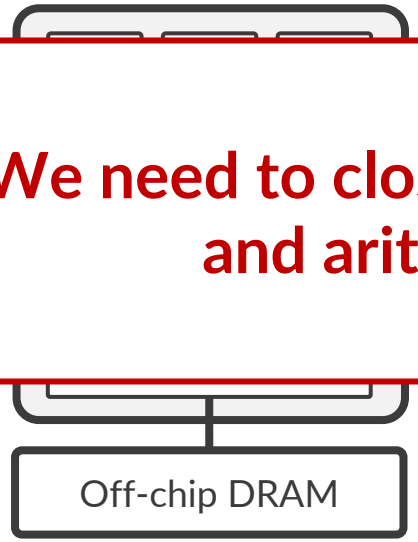
$$\text{Compute to Memory Bandwidth Ratio} = \frac{\textit{Throughput (Ops/sec)}}{\textit{MemoryBandwidth (Bytes/sec)}}$$

≈

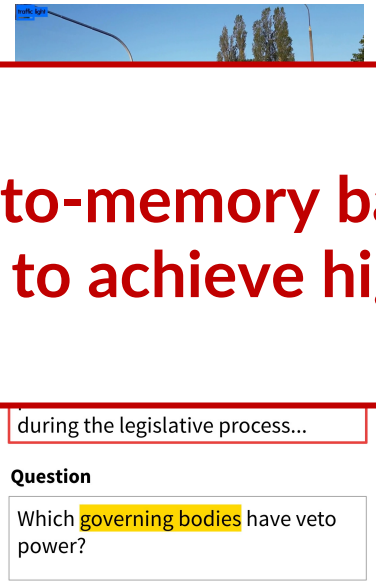
- DNN Model Characteristics

$$\text{Arithmetic Intensity} = \frac{\textit{\# of total operations (Ops)}}{\textit{Total size of data (Bytes)}}$$

Should be balanced for high resource utilization



Memory Bandwidth (GB/s)

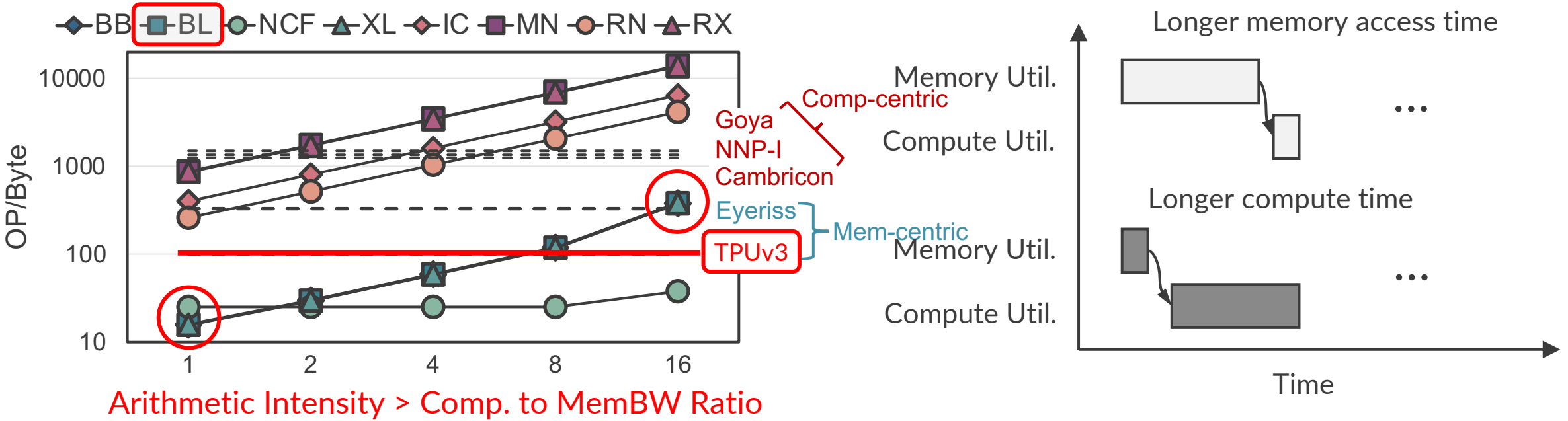


Low Arithmetic Intensity

We need to close this gap between compute-to-memory bandwidth ratio (NPU) and arithmetic intensity (DNN model) to achieve high utilization.

NPU Resource Underutilization Problem

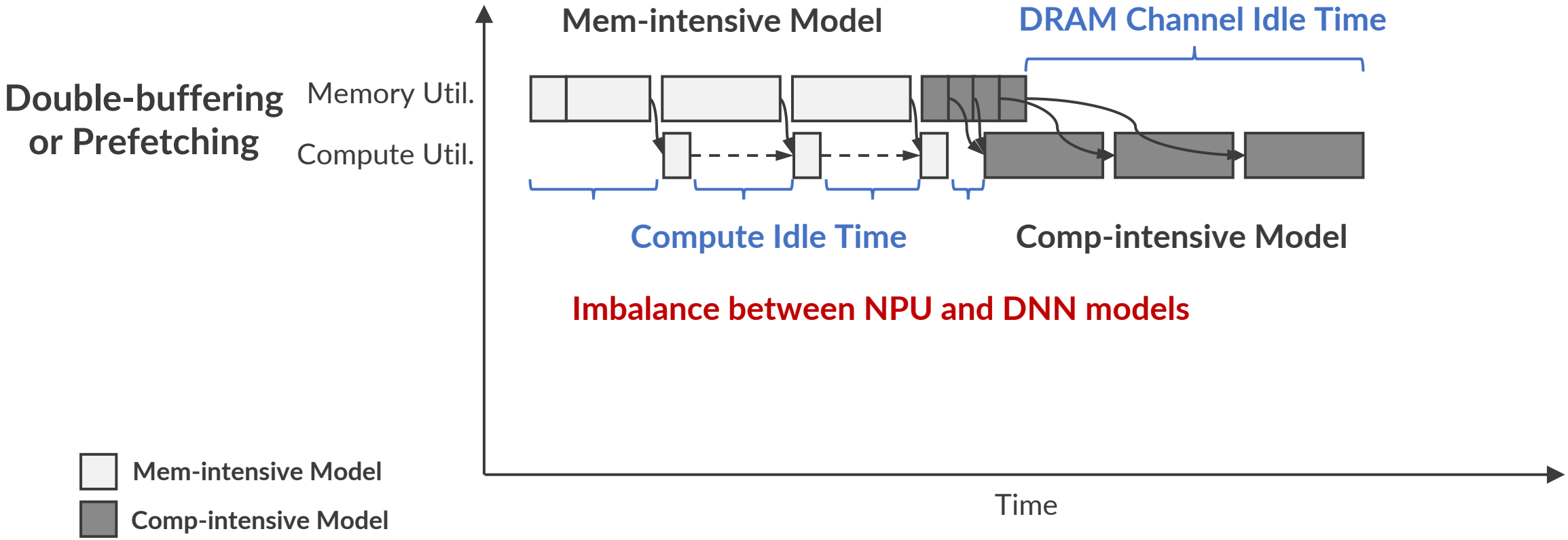
- **No one-size-fits-all NPUs** for various DNN models and batch sizes
  - Compute-centric accelerators are well-suited for Vision Tasks
  - Memory-centric accelerators prefers NLP & Recommendation Tasks
- Significant mismatch between *Arithmetic Intensity* and *Compute-to-Memory BW ratio*



* Each acronym corresponds to BERT-base (BB), BERT-large (BL), NCF recommendation (NCF), XLNet (XL), InceptionV3 (IC), MobileNetV2 (MN), ResNet50 (RN), ResNeXt50 (RX)

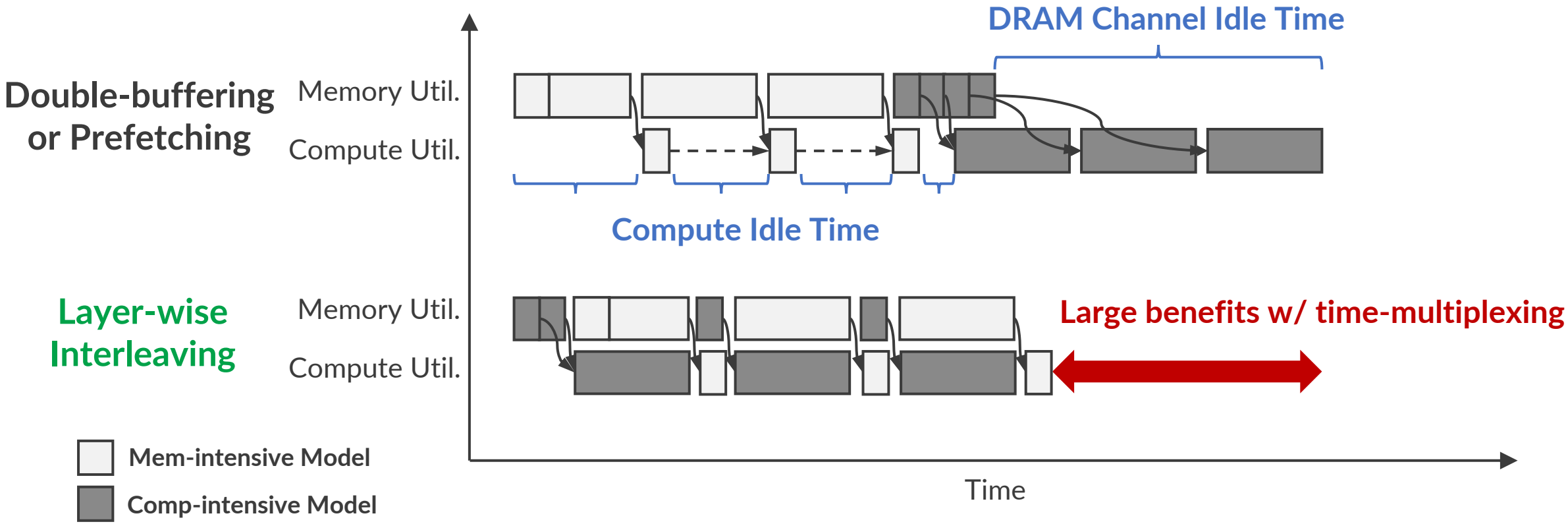
NPU Resource Underutilization Problem

- This mismatch leads to **Resource Underutilization**
  - **Conventional way**: Double-Buffering / Decoupled Access-and-Execution (Prefetching)



NPU Resource Underutilization Problem

- This mismatch leads to **Resource Underutilization**
  - **Our Proposal:** Layer-wise interleaving for balanced resource utilization
  - **Challenge:** Finding a proper prefetch timing is non-trivial due to limited on-chip buffers



Layerweaver: Maximizing NPU Resource Utilization

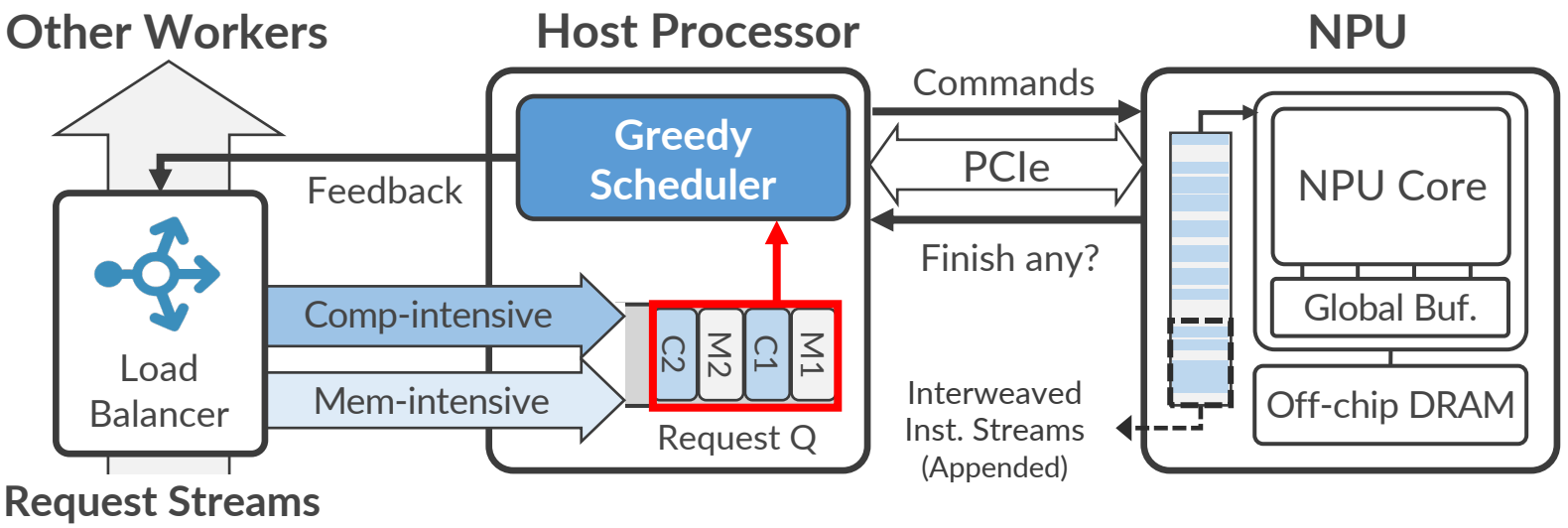
Layerweaver efficiently interweaves layer-wise execution of multiple DNNs to maximize resource utilization and throughput of NPU by utilizing a lightweight scheduler

- **Proposes a lightweight scheduling algorithm** that efficiently interleaves multiple DNNs
- **Carefully considers** on-chip memory size, output forwarding, and starvation
- **Does not require special hardware support to be readily applicable** to existing NPUs
- **Achieves high throughput and nearly-full resource utilization** on 16 pairs of DNN models, various inference scenarios, and NPUs

Presentation Outline

- Resource Underutilization Problem of DNN Accelerators
- Layerweaver: Maximizing Temporal Resource Utilization
- Building Efficient DNN Serving System w/ Layerweaver
  - Overview of Greedy Scheduler w/ Service Requests
  - Calculation of Compute and Memory Idle Time
  - Layer Selection Process
- Evaluation
- Conclusions

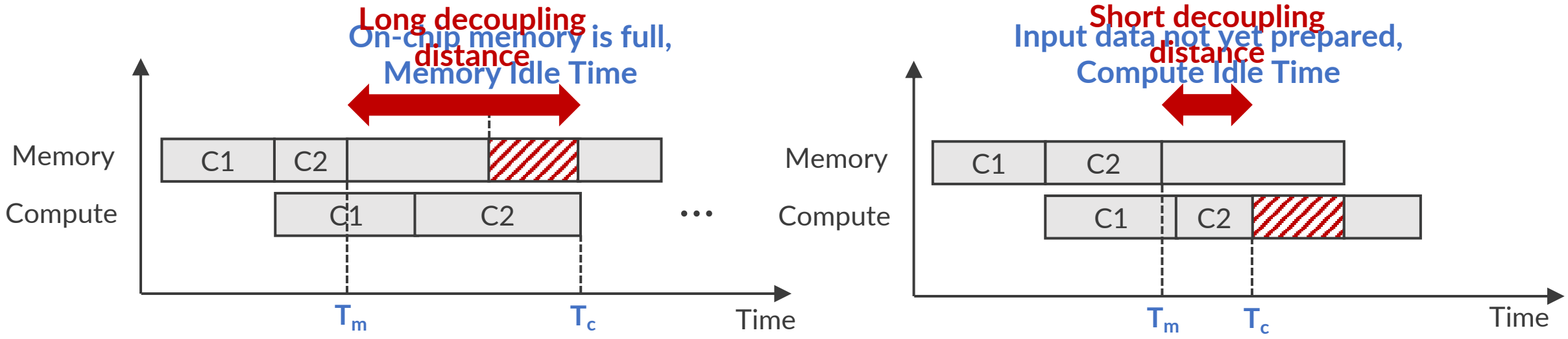
Overview of Layerweaver Serving System



- Assuming a **frontend load balancer** that distributes a mix of both **memory-** and **compute-intensive** requests at pre-determined rates to worker nodes
- Host processor triggers the greedy, layer-wise scheduler when a schedulable batch is formed
- **Greedy Scheduler** accepts requests and adds them to the (scheduling) candidate group
  - Then the scheduler selects the next layer to schedule from this candidate group leading to the least idle time in compute/memory resources

Light-weight Scheduling Algorithm of Layerweaver

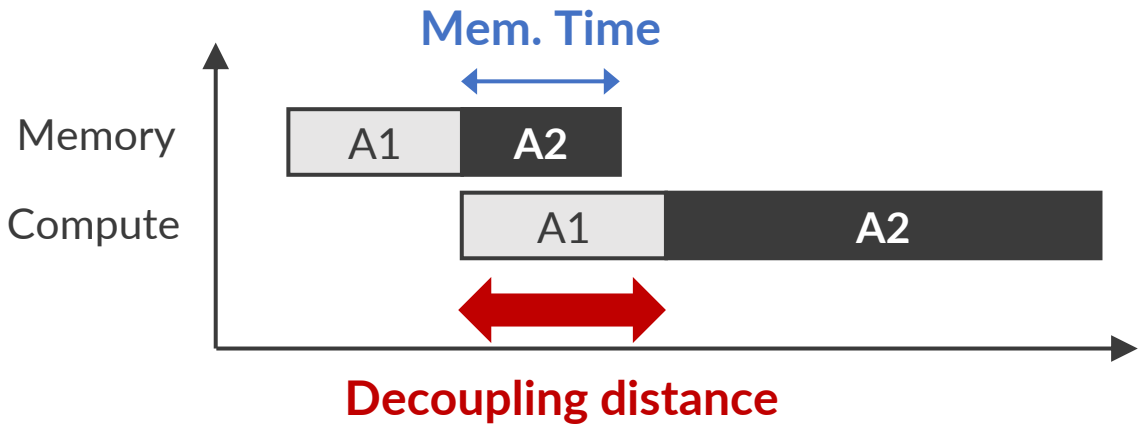
- **Observation:** To minimize idle time, keep a proper **decoupling distance**
  - *Def. Decoupling distance: Time difference btw. the last fetch completion (T_m) and computation finish (T_c)*
  - **Too long decoupling distance:** Potential *Memory Idle Time*
  - **Too short decoupling distance:** Potential *Compute Idle Time*



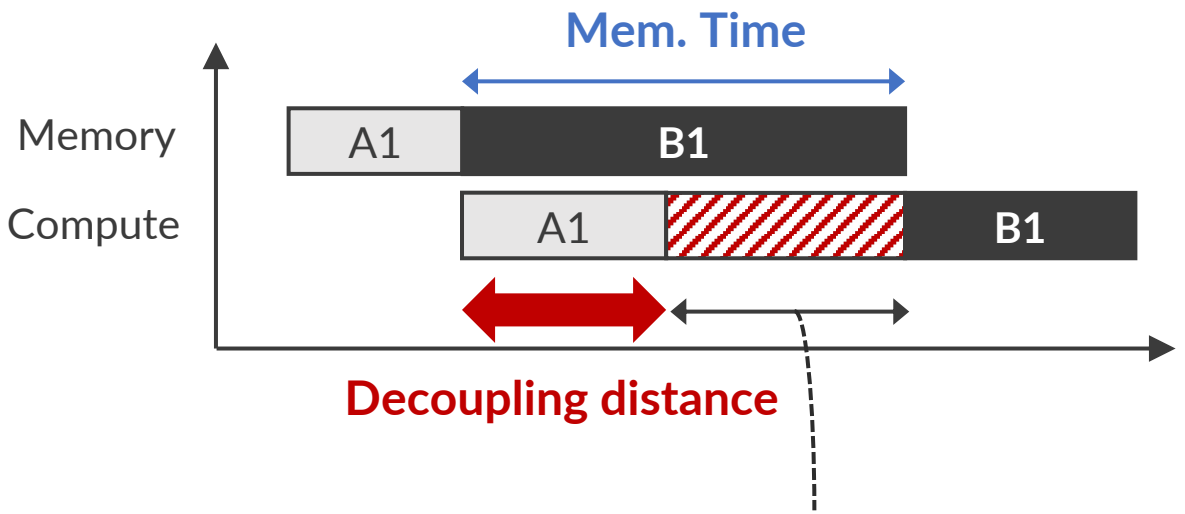
* Dependence graph of input & output activation is omitted

Light-weight Scheduling Algorithm of Layerweaver

- **Calculating Compute Idle Time**
  - When **too short decoupling distance** exists, there could be **Compute Idle Time**
  - Before starting computation of B1, the input data of B1 should be prepared



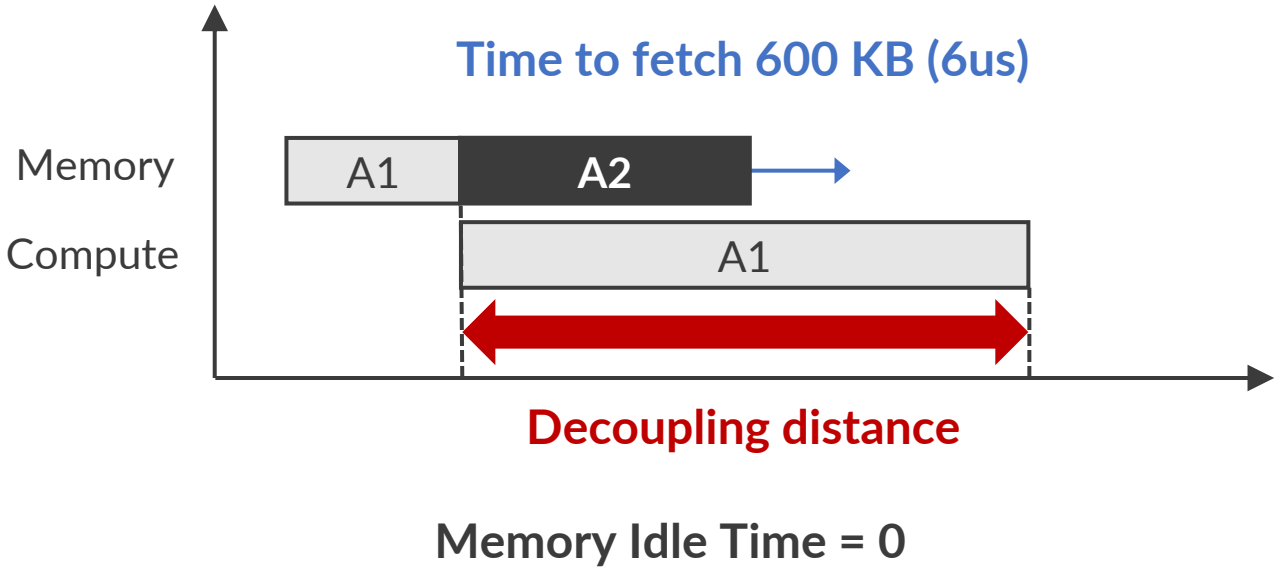
Compute Idle Time = 0



Compute Idle Time = Mem. Time – Decoupling distance

Light-weight Scheduling Algorithm of Layerweaver

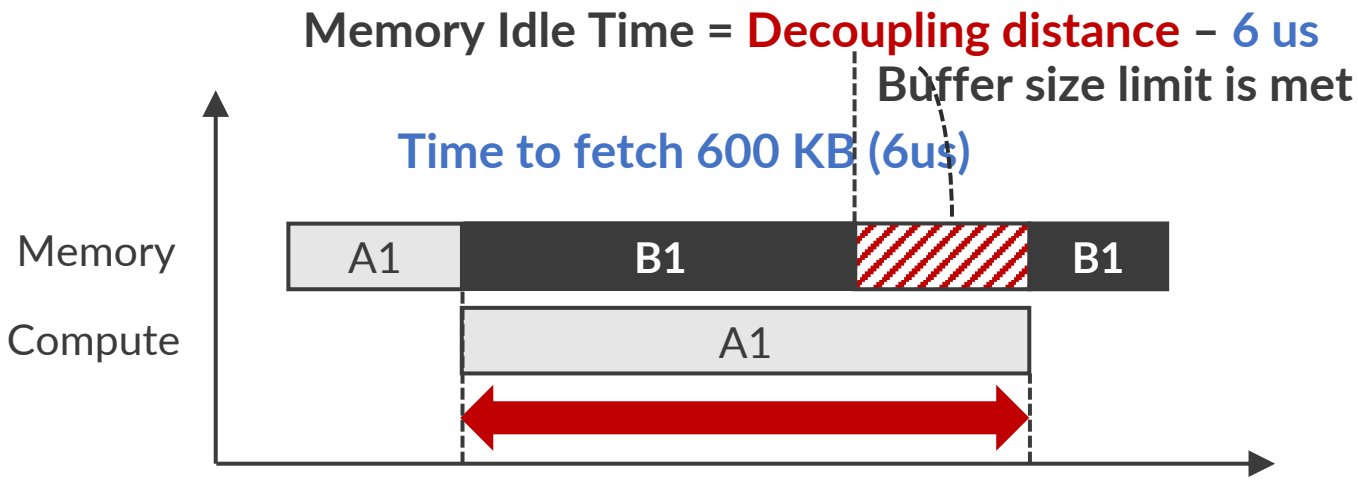
- **Calculating Memory Idle Time**
  - When **too large decoupling distance** exists, there could be **Memory Idle Time**
  - Free on-chip buffer size for now = 600 KB = 600 KB / 100 GB/s (Mem. BW) = 6 us



Memory Idle Time = 0

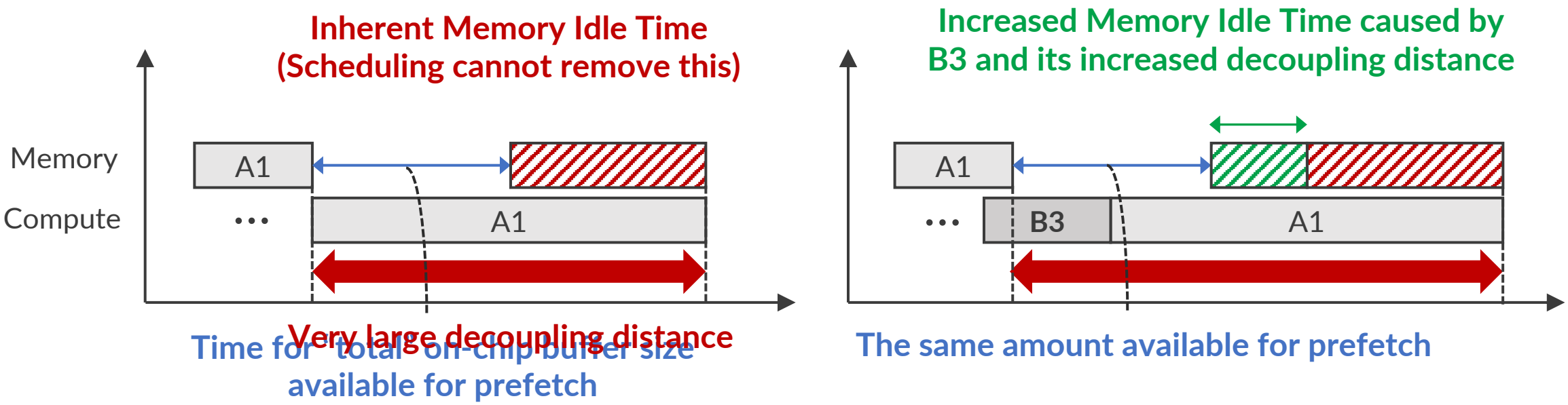
Light-weight Scheduling Algorithm of Layerweaver

- **Calculating Memory Idle Time**
  - When **too large decoupling distance** exists, there could be **Memory Idle Time**
  - Free on-chip buffer size for now = 600 KB = 600 KB / 100 GB/s (Mem. BW) = 6 us



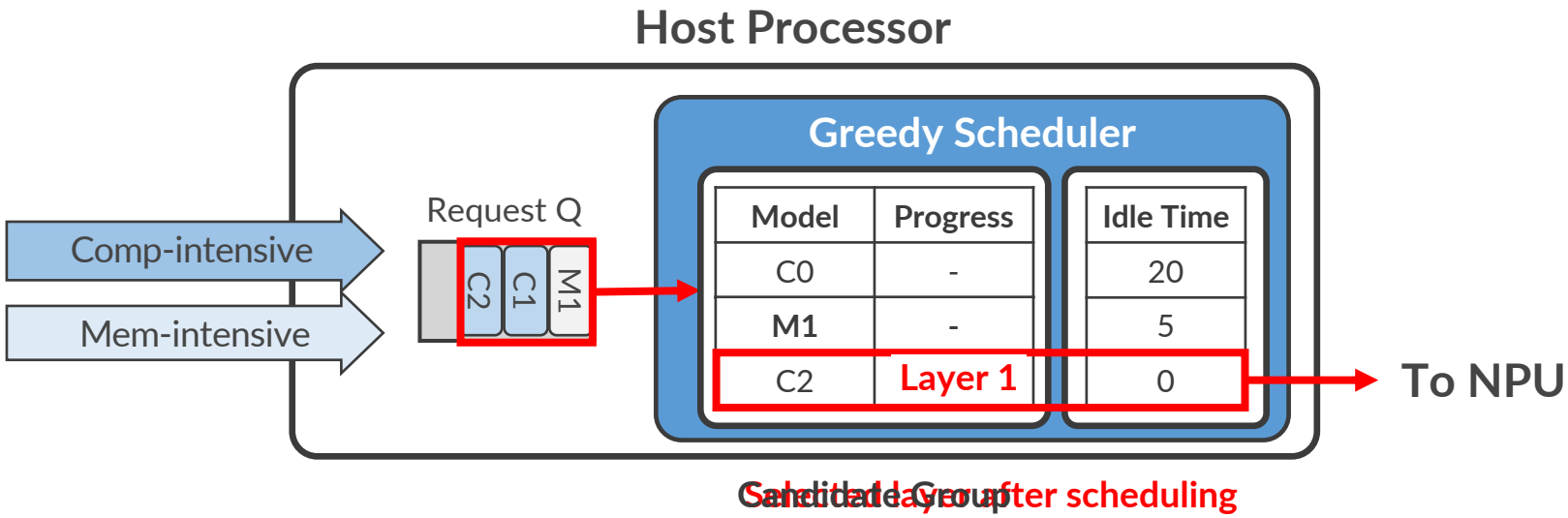
Light-weight Scheduling Algorithm of Layerweaver

- **Calculating Memory Idle Time (Cont'd)**
  - Some models have **Inherent Memory Idle Time**, which cannot be removed by scheduling
  - To prevent **additional memory idle time**, keep minimum **decoupling distance**



Light-weight Scheduling Algorithm of Layerweaver

- **Layer Selection**
  - Detailed formal descriptions about every idle time case are available in our paper
  - After exactly calculates the idle time of **"each layers in progress"** from K candidate requests ~ **O(k)**
  - Select the layer that incurs *minimum memory and compute idle time*, and repeat for N layers ~ **O(kN)**

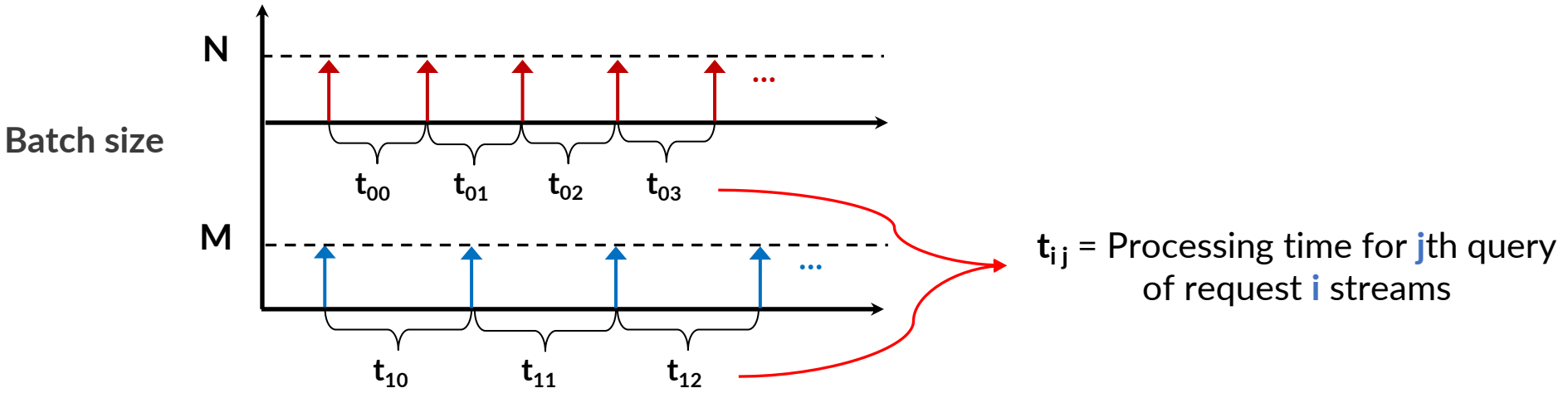


Presentation Outline

- Resource Underutilization Problem of DNN Accelerators
- Layerweaver: Maximizing Temporal Resource Utilization
- Building Efficient DNN Serving System w/ Layerweaver
  - Overview of Greedy Scheduler w/ Service Requests
  - Calculation of Compute and Memory Idle Time
  - Layer Selection Process
- Evaluation
- Conclusions

Evaluation Methodology & Inference Scenario

- **Evaluation Scenario:** Single- / Multi-batch Streams (Extended Single-stream in MLPerf [1])
- **Accelerators**: *Memory-centric* (Google TPUv3 style) & *Compute-centric* (Intel NNP-I style) NPUs
- **Baseline Strategy**: Memory-only / Compute-only / Fair / AI-MT [2]
- **Metrics**: **Normalized System Throughput (STP)**, **Time Utilization (%)**

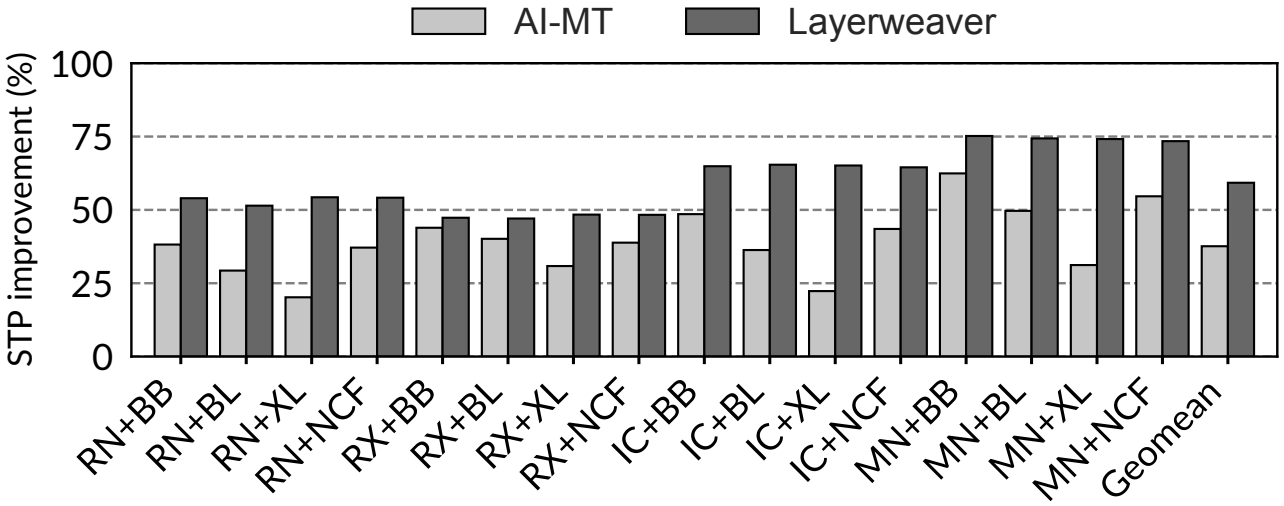


Input Timeline for Multi-Batch Streams (# of Streams = 2)

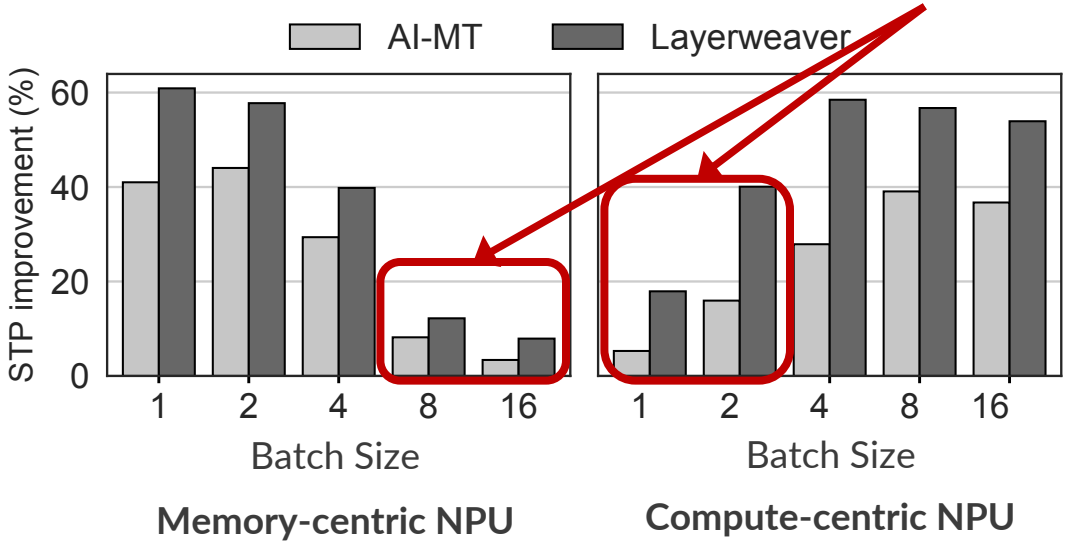
[1] Reddi et al., "MLPerf Inference Benchmark", ISCA '20
[2] Baek et al., "A Multi-Neural Network Acceleration Architecture", ISCA '20

Performance Evaluation (System Throughput)

- **Scenario:** Single-Batch Streams (# of Streams = 2) on TPUv3-style NPU
- **Layerweaver:** 60.1% throughput increase over the baseline (Fair)
- **Against AI-MT:** Layerweaver shows 21.6% higher geomean STP than AI-MT
  - **AI-MT**: State-of-the-art time-multiplexed DNN multi-tasking technique
  - Layerweaver estimates the resource idle time more precisely than AI-MT based on a simple heuristic
- For **various Multi-Batch Streams** configurations, Layerweaver outperforms all the other schemes **Smaller benefits w/ biased config.**



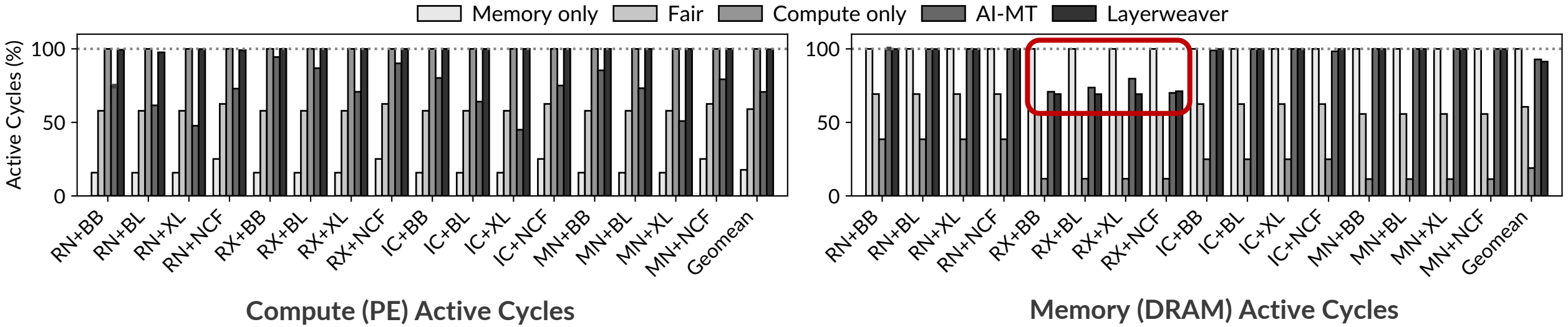
Single-Batch Streams (# of Streams = 2)



Multi-Batch Streams (# of Streams = 2, Geomean)

Overall NPU Resource Utilization

- **Evaluation Scenario**: Single-Batch Streams (# of Streams = 2) w/ TPUv3-style NPU
- Layerweaver almost fully (Geomean 99.7% / 91.3% for Compute / Memory) utilizes active cycles
- ResNeXt causes Inherent Memory Idle Time and about 29-31% drop in memory active cycles (Red box)
  - As we discussed this drop is not attributed to scheduling decisions
- **Memory-** or **Compute-only** suffers severely from either compute or memory bandwidth underutilization
- **AI-MT** shows sub-optimal results due to imprecise estimation of resource idle time



Scheduling Overhead

- Layerweaver scheduler has **O(kN)** complexity
  - **k**: # of candidate models, **N**: total # of layers to schedule
- Carefully modeled scheduling overhead on a host processor
  - Intel i7-7700K CPU @ 4.20GHz
  - **Greedy Scheduler Throughput** is **15 layers / us**
- Compared to the single batch-1 layer latency which ranges from 4.7us to 221us
  - The scheduling overhead is negligible
  - Also, scheduling happens off-critical path most of the time using host CPU
- Thus, **Layerweaver** can flexibly support dynamically-changing request patterns at runtime

Layerweaver: Maximizing NPU Resource Utilization

Layerweaver efficiently interweaves layer-wise execution of multiple DNNs to maximize resource utilization and throughput of NPU by utilizing a lightweight scheduler

- **Proposes a lightweight scheduling algorithm** that efficiently interleaves multiple DNNs
- **Carefully considers** on-chip memory size, output forwarding, and starvation
- **Does not require special hardware support to be readily applicable** to existing NPUs
- **Achieves high throughput and nearly-full resource utilization** on 16 pairs of DNN models, various inference scenarios, and NPUs

The 27th IEEE International Symposium on High-Performance Computer Architecture

Layerweaver: Maximizing Resource Utilization of Neural Processing Units via Layer-Wise Scheduling

Young H. Oh† , Seonghak Kim*, Yunho Jin*, Sam Son*, Jonghyun Bae*
Jongsung Lee*, Yeonhong Park*, Dong Uk Kim*, Tae Jun Ham*, Jae W. Lee*

*Seoul National University

†Sungkyunkwan University

Author Contact Info: younghwan@skku.edu