ASPLOS 2025

Rotterdam, The Netherlands

Squeezing Operator Performance Potential for the Ascend Architecture

Yuhang Zhou[1], Zhibin Wang[1], Guyue Liu[2], Shipeng Li[1], Xi Lin[1], Zibo Wang[1], Yongzhong Wang[3], Fuchun Wei[3], Jingyi Zhang[3], Zhiheng Hu[3], Yanlin Liu[3], Chunsheng Li[3], Ziyang Zhang[3], Yaoyuan Wang[3], Bin Zhou[4], Wanchun Dou[1], Guihai Chen[1], Chen Tian[1]

[1] Nanjing University [2] Peking University [3] Huawei Technologies Co., Ltd. [4] Shandong University

Outline

- Introduction
- System Design
- Case Study
- Evaluation
- Conclusion

Outline

AI Domain-Specific Architecture (DSA)

deepseek

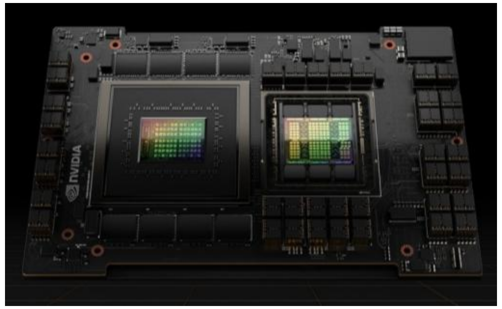
Qwen2.5

OPEN SORA

Deep learning models

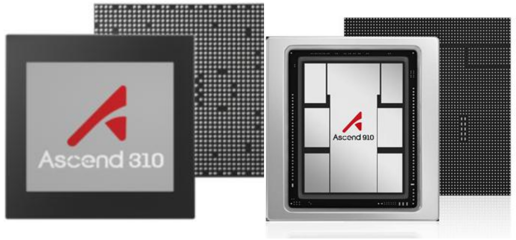
Better arithmetic support

Domain-specific architecture



NVIDIA GPU

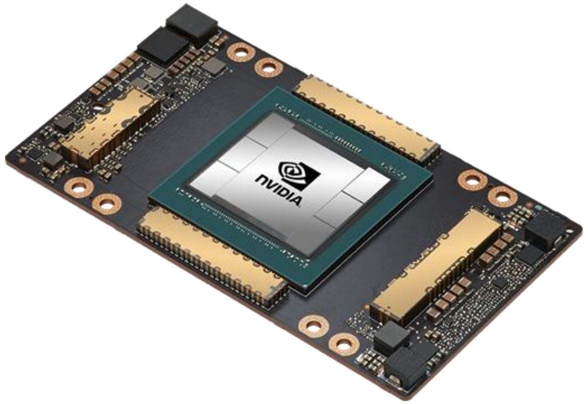
Google TPU



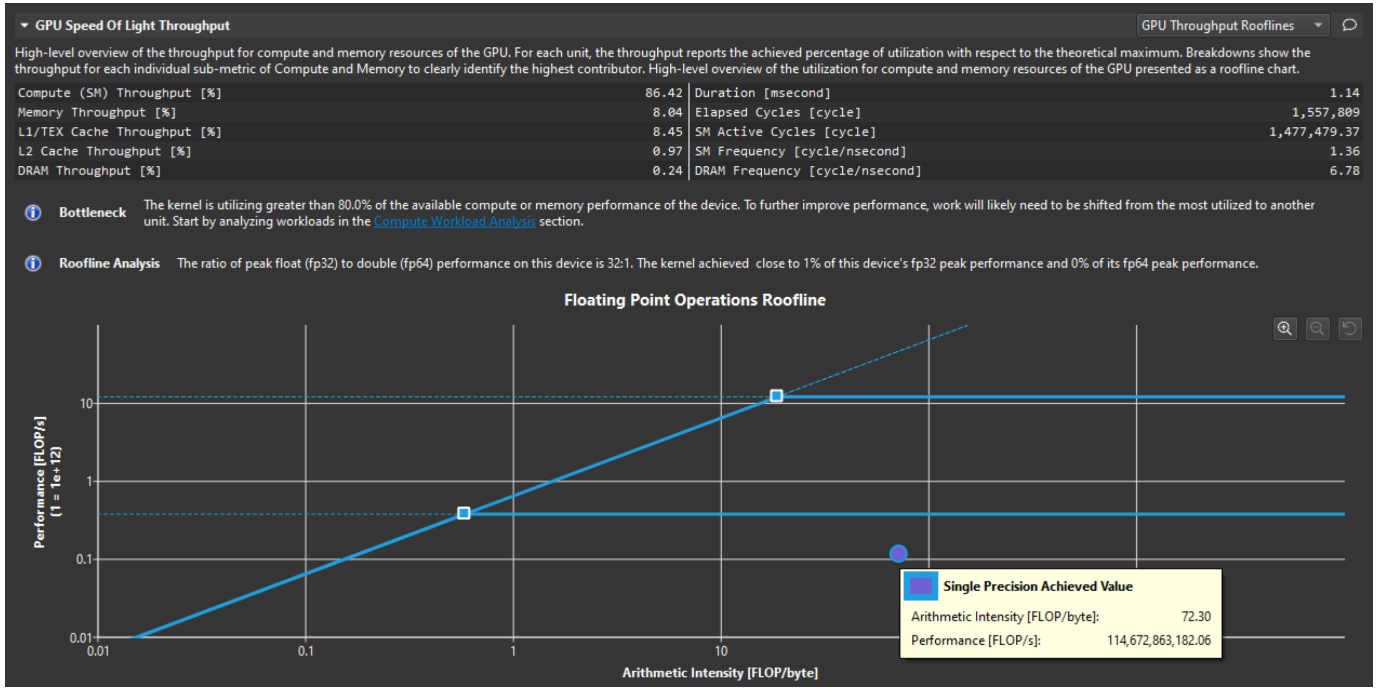
Huawei Ascend NPU

Cambricon MLU

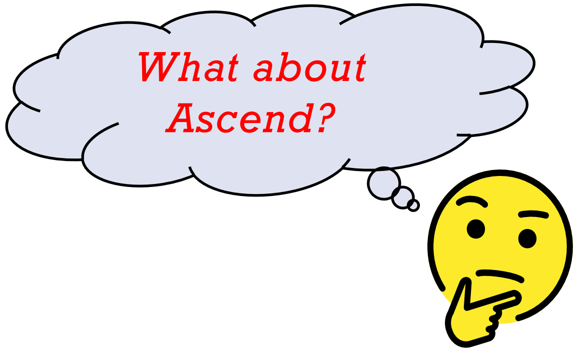
Operator Optimization Needs Profiling



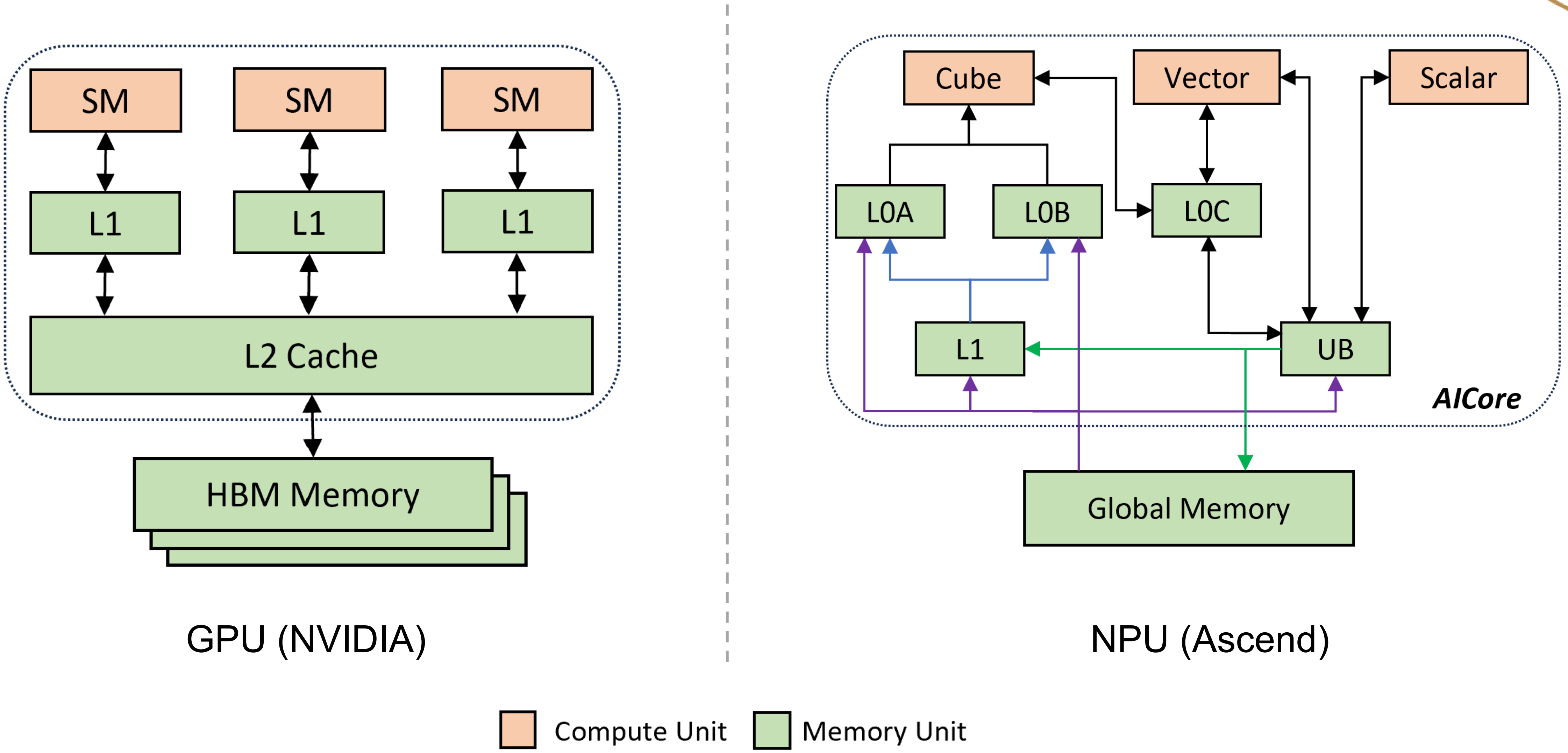
NVIDIA Nsight™ Compute



Ascend

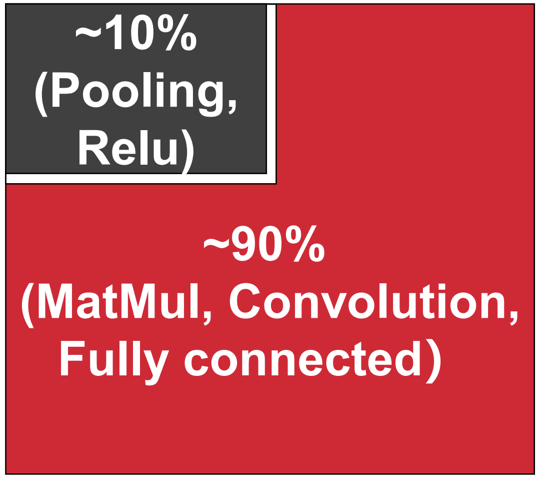


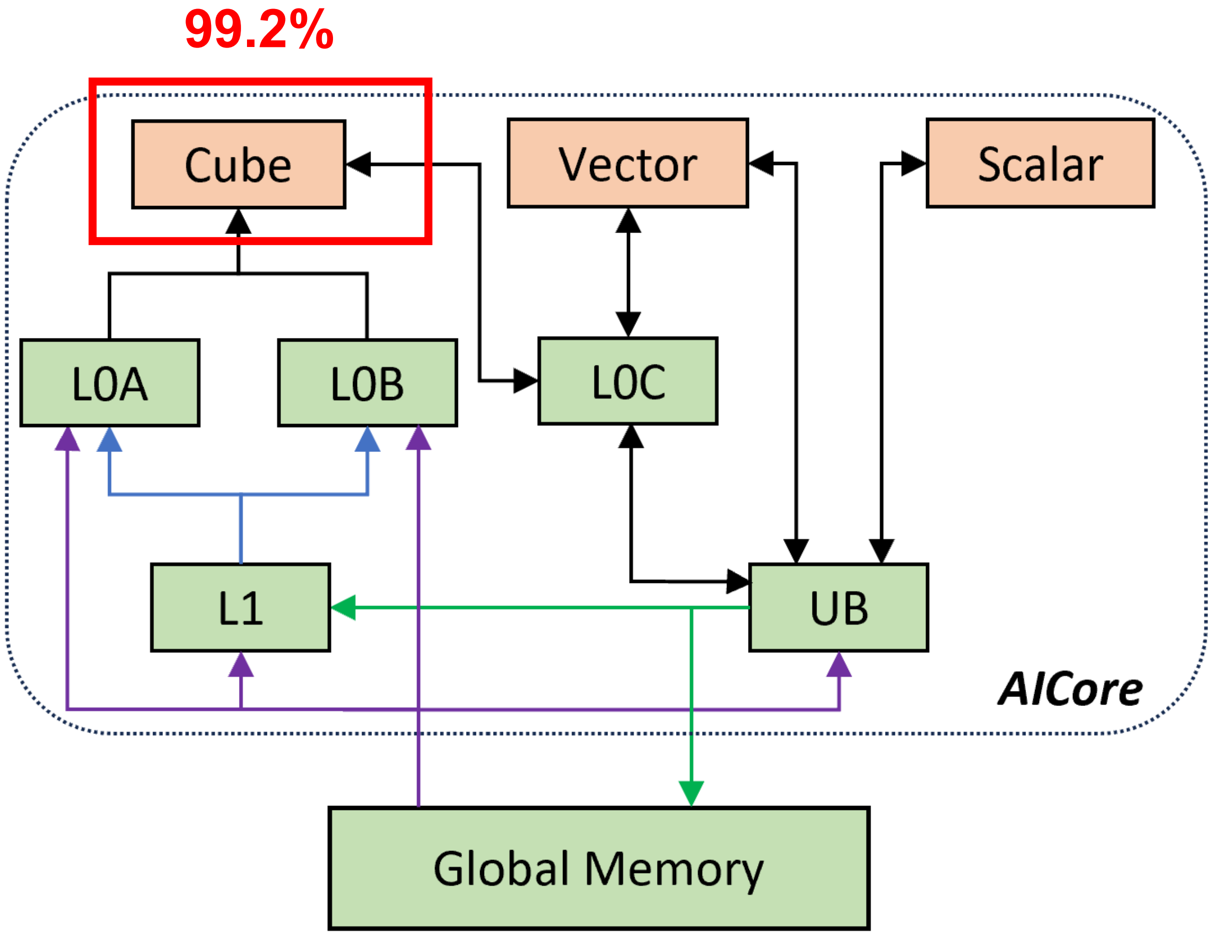
Ascend Architecture



Dedicated Compute Units

Transformer computation





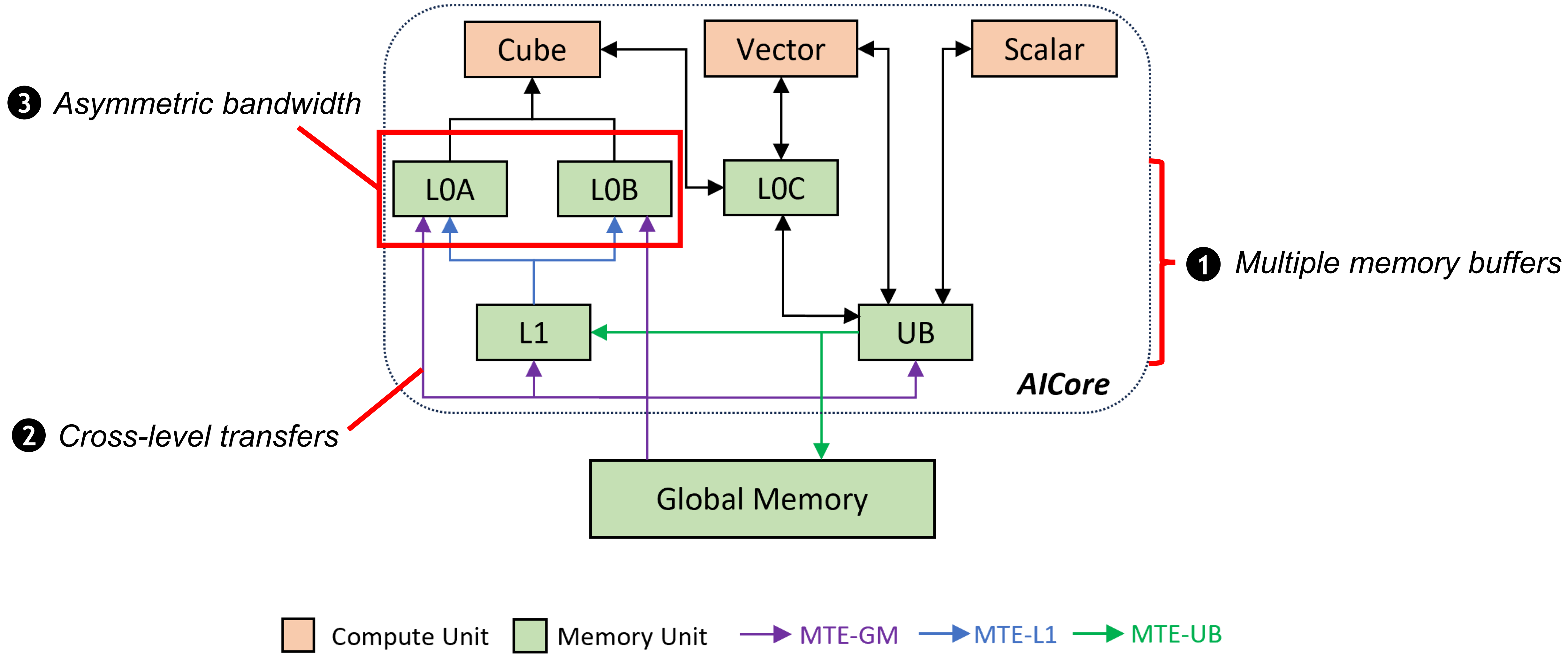
Computing precision

Cube: INT8/FP16

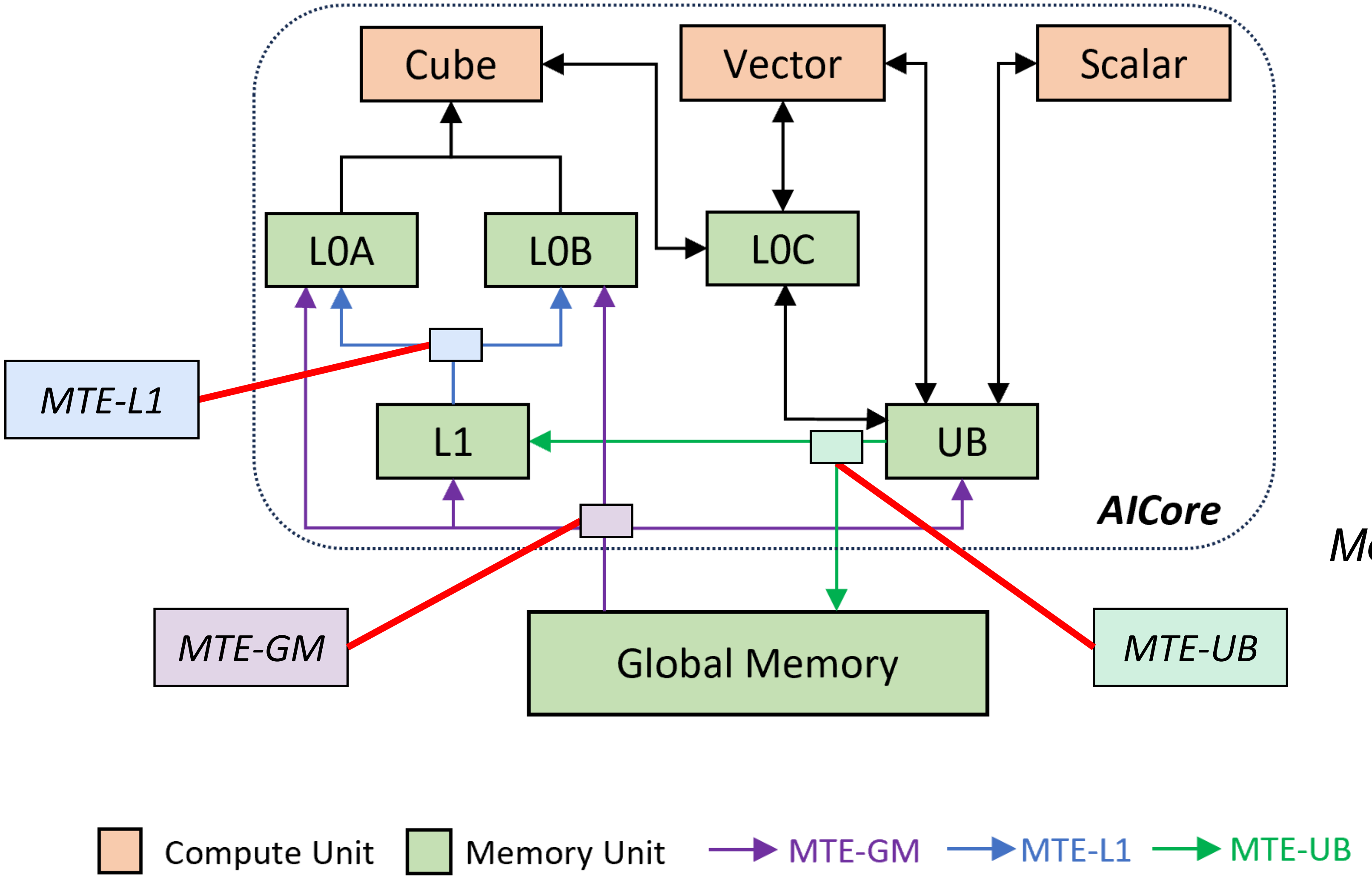
Vector: INT8/FP16/FP32

Scalar: INT32/FP32/…

Customized Memory Architecture

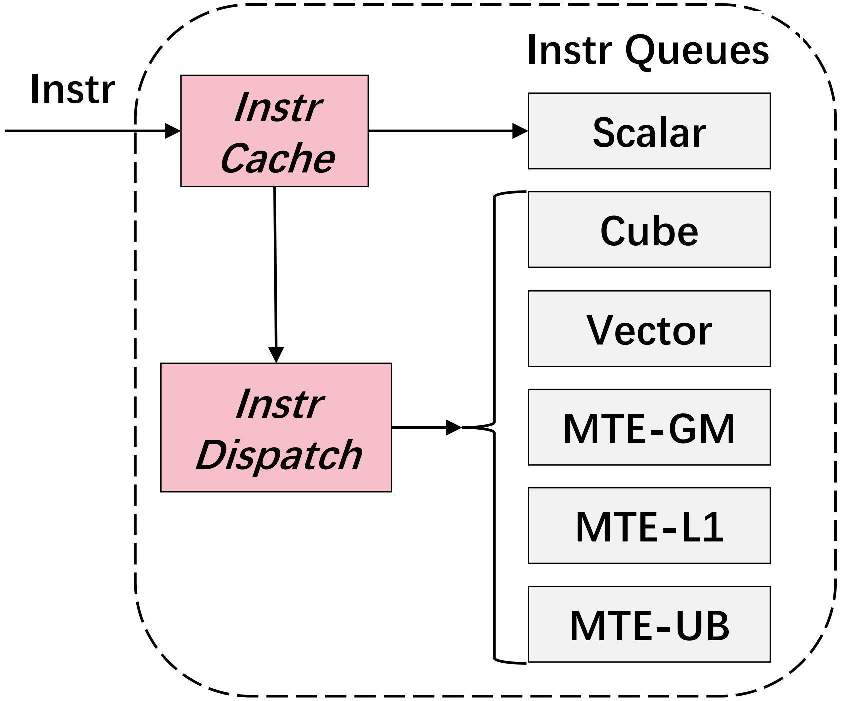


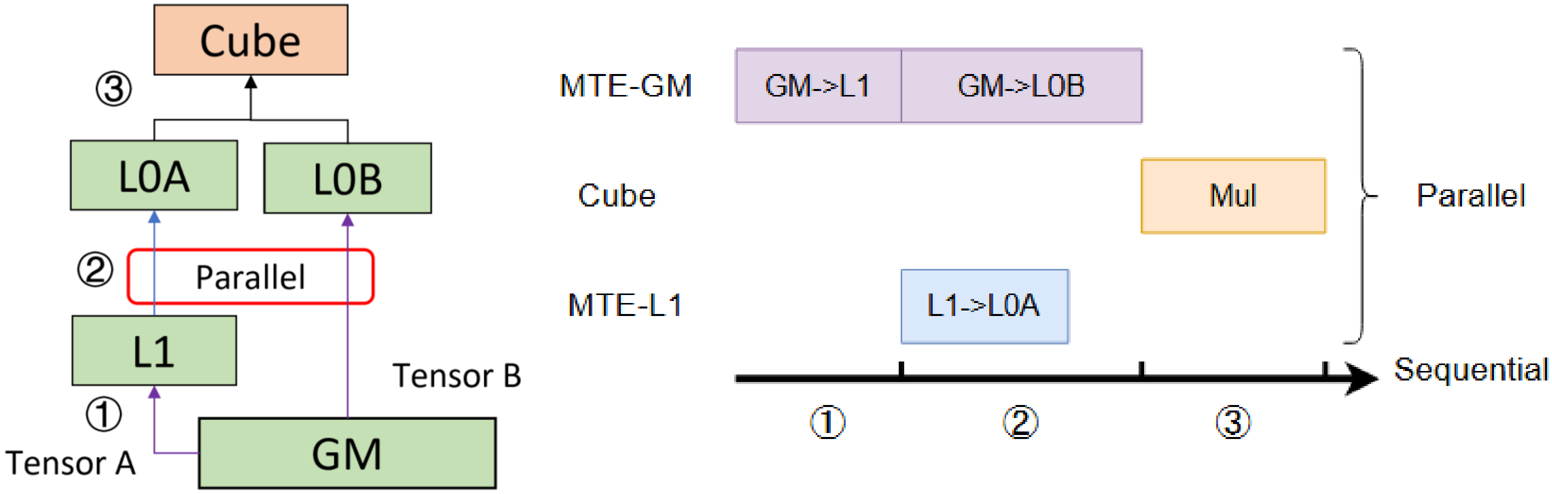
Efficient Transfer Control Units



Memory transfer engine

Instruction Pipeline





Example of matrix multiplication $A \times B$

Inter parallelism, intra serialism.

Summary of Ascend Architecture

Pros

Ascend Architecture

Cons

Dedicated compute units

Accurately identifying operator bottleneck is a challenging, but essential task!

Operational flexibility

puting

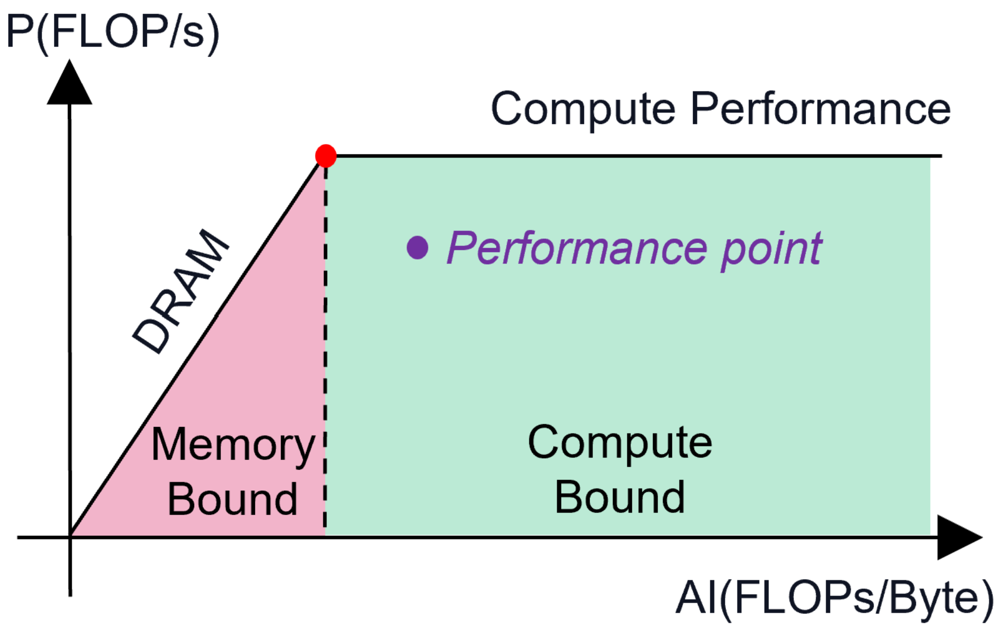
ITE

eline

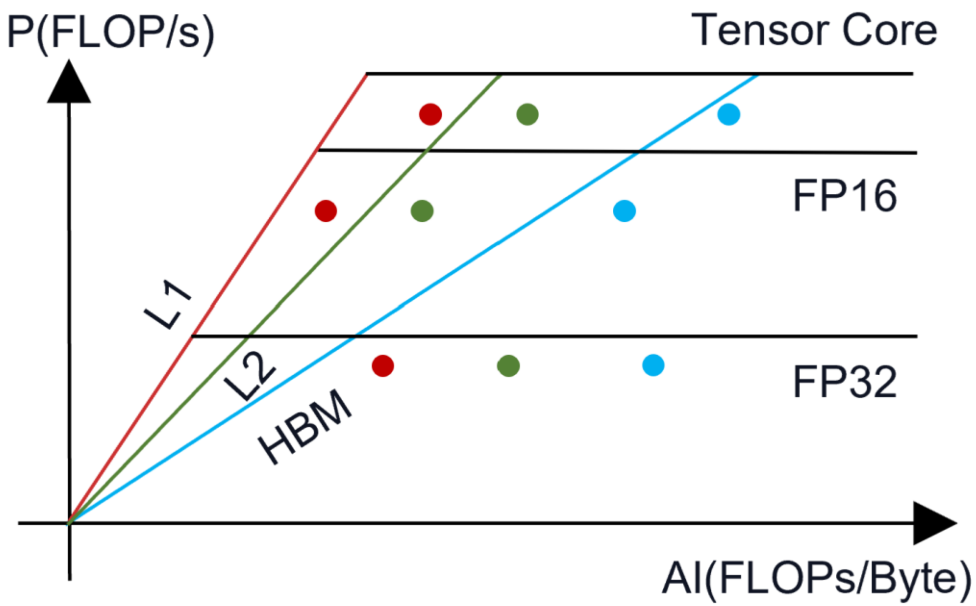
...

Efficient transfer control and instruction pipeline

Existing Operator Performance Analysis

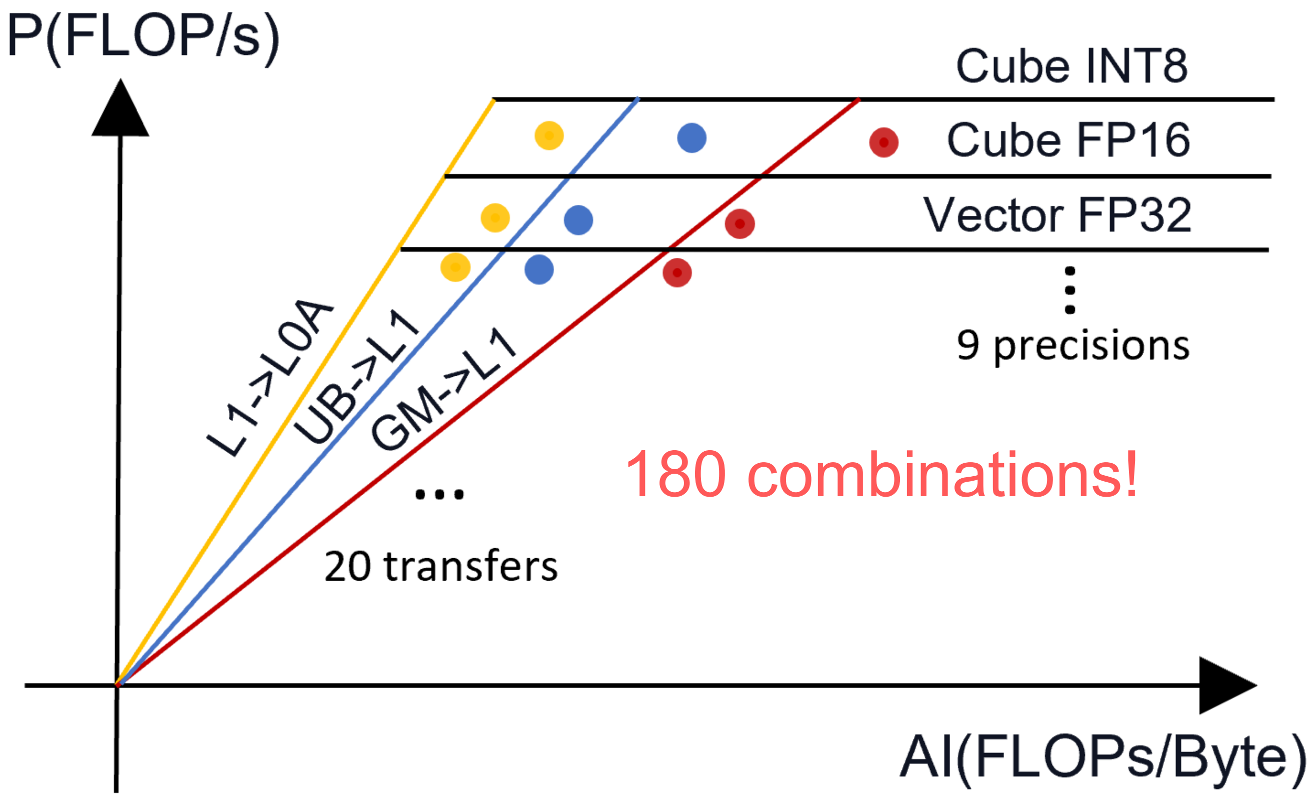


DRAM Roofline



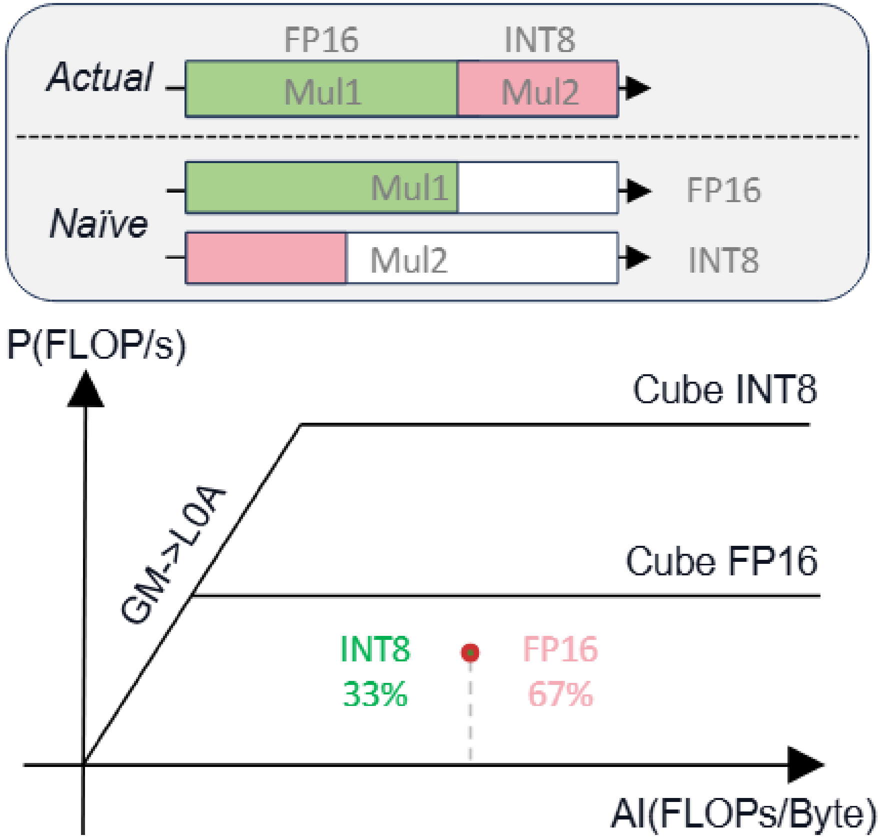
Hierarchical Roofline

Limitations of Performance Analysis



(i) Massive combinations between precisions and transfers

Limitations of Performance Analysis



Underutilization?

(ii) Incorrect analysis by ignoring the sequential execution

Our Goals

Problems	Solutions
Complicated analysis	Simplified and accurate operator performance analysis
Incorrect analysis	Simplified and accurate operator performance analysis
Non-trivial optimization	Effective guidance for operator optimization

Outline

Introduction

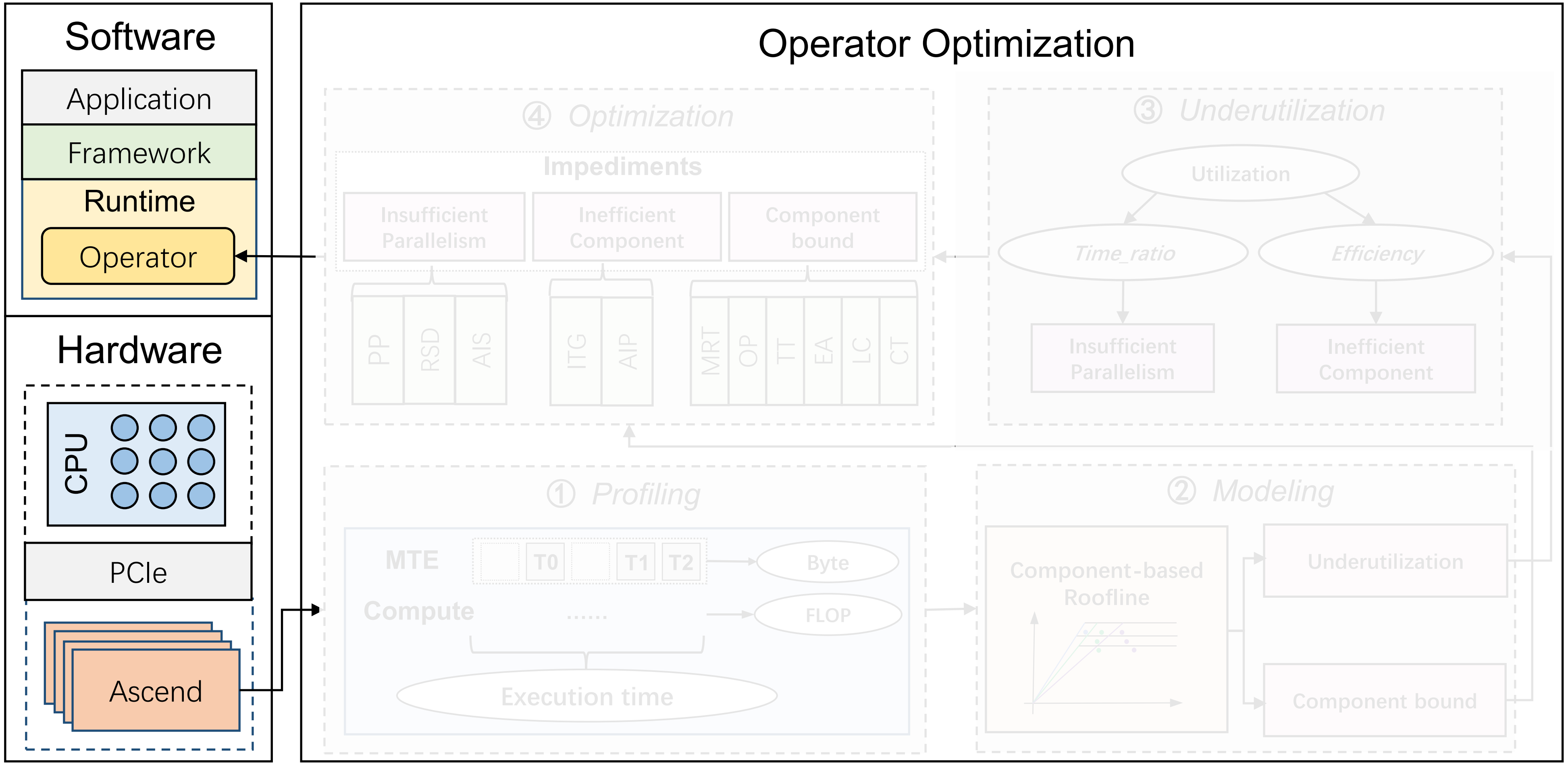
System Design

Case Study

Evaluation

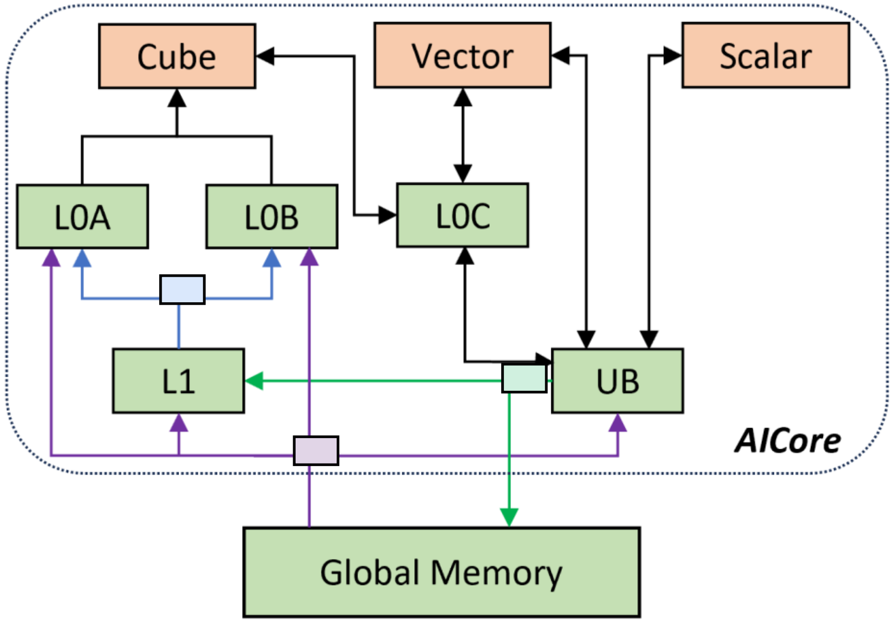
Conclusion

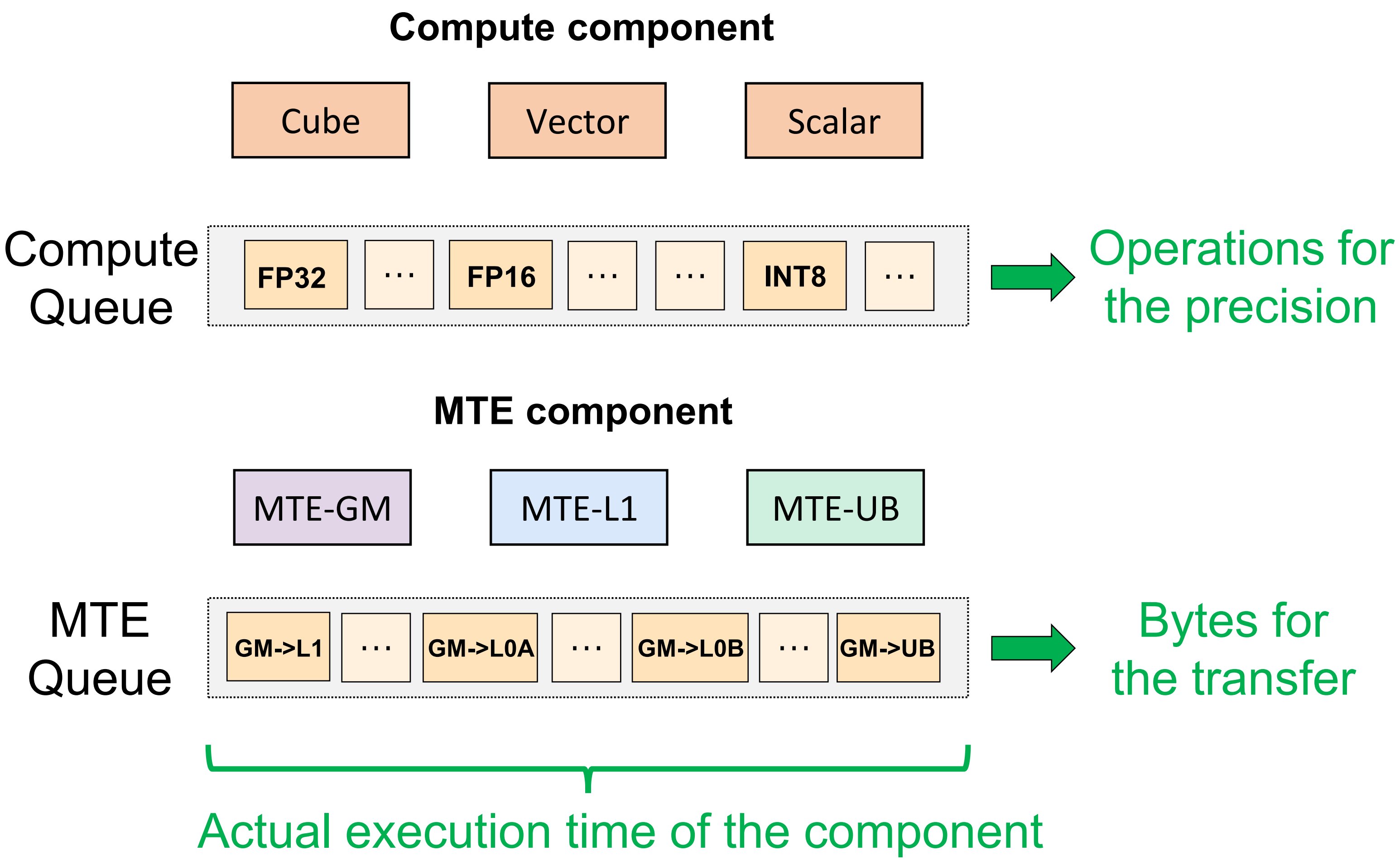
Overview



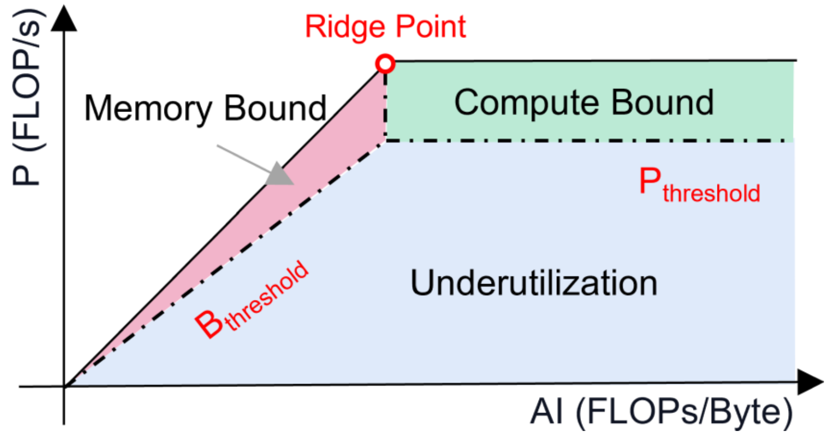
Software
Application
Framework
Runtime
Operator
Hardware
CPU
PCIe
Ascend
Operator Optimization
④ Optimization
Impediments
Insufficient Parallelism
Inefficient Component
Component bound
PP
RSD
AIS
ITG
AIP
MRT
OP
TT
EA
LC
CT
③ Underutilization
Utilization
Time_ratio
Efficiency
Insufficient Parallelism
Inefficient Component
① Profiling
MTE
T0
T1
T2
Byte
Compute
......
FLOP
Execution time
② Modeling
Component-based Roofline
Underutilization
Component bound

Profiling and Component Abstraction





Component-based Roofline Model



Utilization of component can reflect the operator's bottleneck.

Operator-aware ideal performance

$$U_{\text{cube}} = \frac{A_{\text{cube}}}{I_{\text{cube}}} \begin{cases} A_{\text{cube}} = \dfrac{O_{\text{cube}}}{T_{\text{total}}} & \text{Profiling} \\ I_{\text{cube}} & \text{Different precisions?} \end{cases}$$

$$I_{\text{cube}} = \frac{\sum_{\text{prec}} O_{\text{prec}}}{\sum_{\text{prec}} \frac{O_{\text{prec}}}{P_{\text{prec}}}}$$

Harmonic Mean Arithmetic Power

Underutilization Analysis

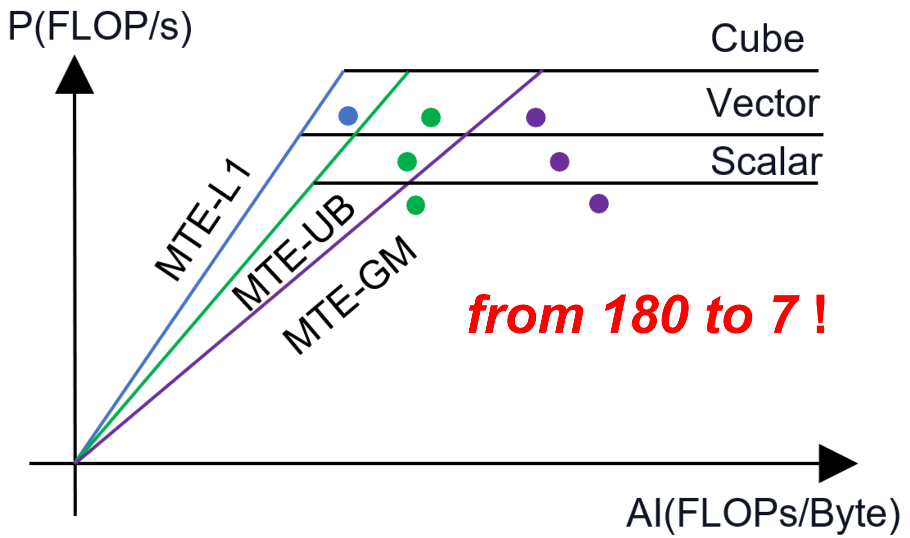
❶ $$U_{cube} = \frac{A_{cube}}{I_{cube}} = \underbrace{\frac{O_{cube}}{T_{cube} \cdot I_{cube}}}_{E_{cube}} \cdot \underbrace{\frac{T_{cube}}{T_{total}}}_{R_{cube}}$$

❷ $$E_{\text{component}} \leq \frac{R_{\text{threshold}}}{U_{\text{threshold}}} \qquad R_{\text{component}} < R_{\text{threshold}}$$

Inefficient Component ***Insufficient Parallelism***

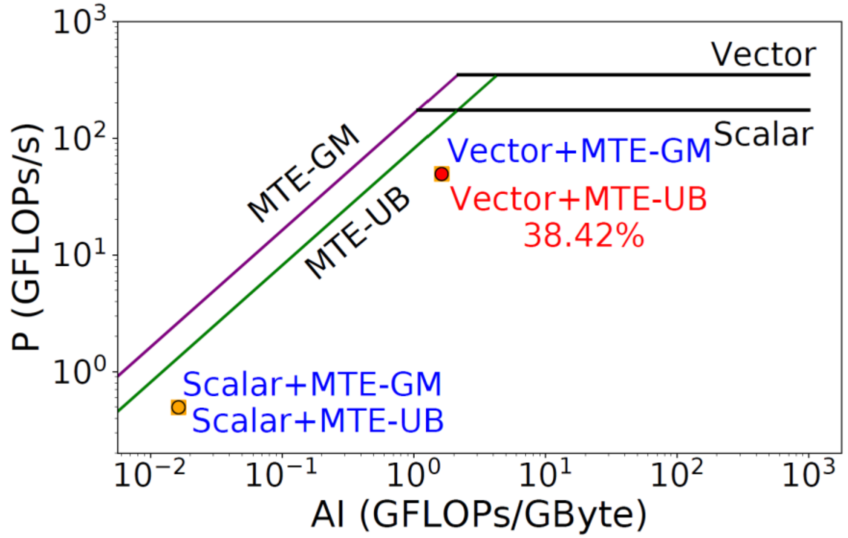
Pruning, Visualization and Analysis

Pruning results



- ✓ Component abstraction
- ✓ Remove irrelevant components
- ✓ Remove impossible combinations

Roofline Analysis of Add_ReLU Operator



$U_{component}$ of Vector+MTE-UB (*38.42%*):

Underutilization

$R_{component}$ of MTE-GM (*58.68%*):

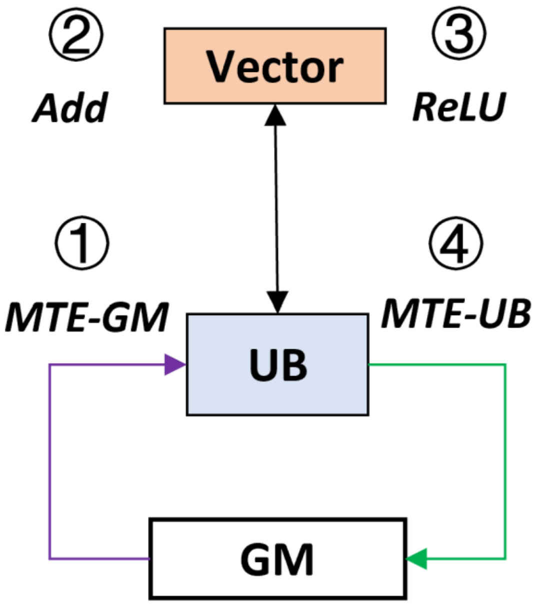
Insufficient parallelism

Outline

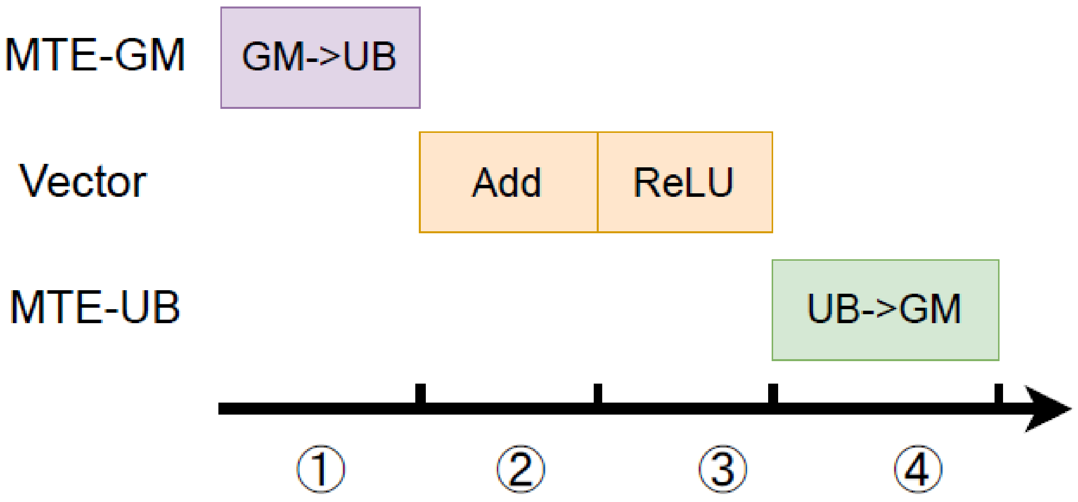
- Introduction
- System Design
- **Case Study**
- Evaluation
- Conclusion

Case Study: Optimization of *Add_ReLU Operator*

$$\mathrm{Add\_ReLU}(x) = \mathrm{ReLU}(x + c)$$



Data flow

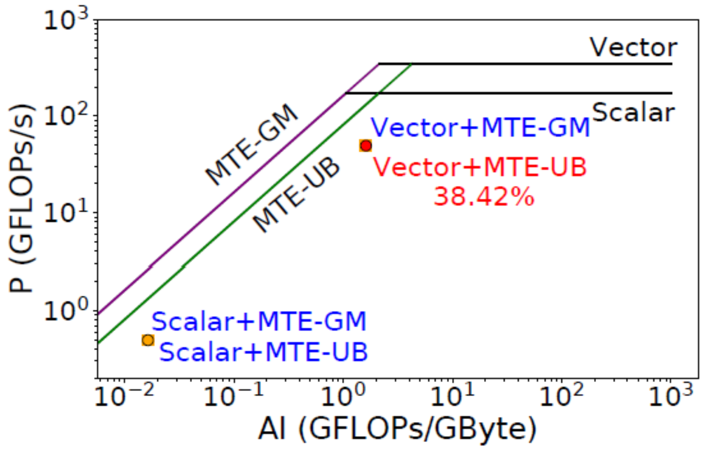


Instruction timeline

Iteration 1: *Reducing spatial dependency*

Original Code

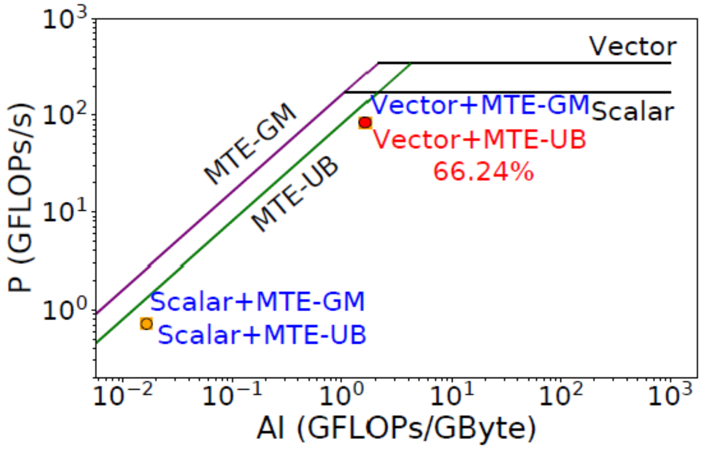
```
① ...
② ub_to_gm(gm_1, ub_1);
③ gm_to_ub(ub_1, gm_2);
④ ...
```



Insufficient parallelism (*38.42%*)

Optimized Code

```
① ...
② ub_to_gm(gm_1, ub_2);
③ gm_to_ub(ub_1, gm_2);
④ ...
```

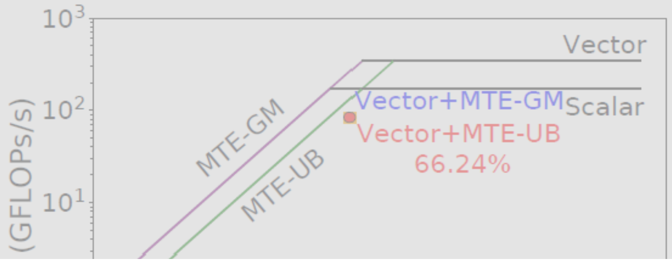


MTE-UB bound (66.24%)

Iteration 2: *Minimizing redundant transfer*

Original Code

```
① for i = 1 to n do
②    gm_to_ub(ub_1, c);
③    ...
```



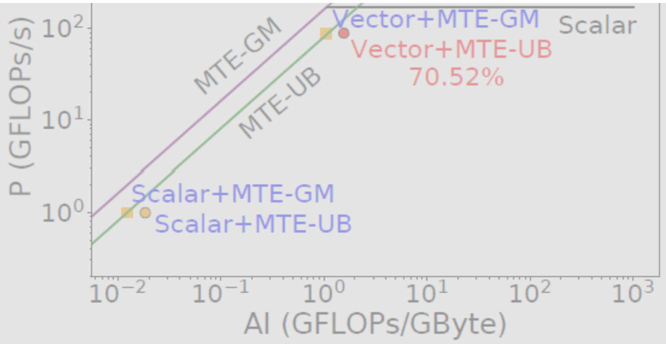
MTE-UB bound
(66.24%)

The single operator time reduced by 1.73×.

The *component_utilization* up by 32.1%.

The total inference latency down by 244.261 μs.

```
① gm_to_ub(ub_1, c);
② for i = 1 to n do
③    ...
④ end for
```



MTE-UB bound
(70.52%)

Optimization Experience

We summarize the common bottleneck causes and optimization strategies.

Bottleneck Cause	**Compute Bound**	**MTE Bound**	**Insufficient Parallelism**	**Inefficient Compute**	**Inefficient MTE**
Strategy					

Operator	Bottleneck Cause and Optimization Strategy					Speedup
	Compute Bound	MTE Bound	Insufficient Parallelism	Inefficient MTE	Inefficient Compute	
Add_ReLU		MRT	RSD			1.72
Depthwise		MRT	AIS,RUS,PP	ITG		1.26
AvgPool					AIP	4.31
Mul			RSD			1.34
Conv2D		MRT	RSD			2.65
FullyConnection				ITG		1.22
MatMul		OF				1.10
GeLU	EA					1.06

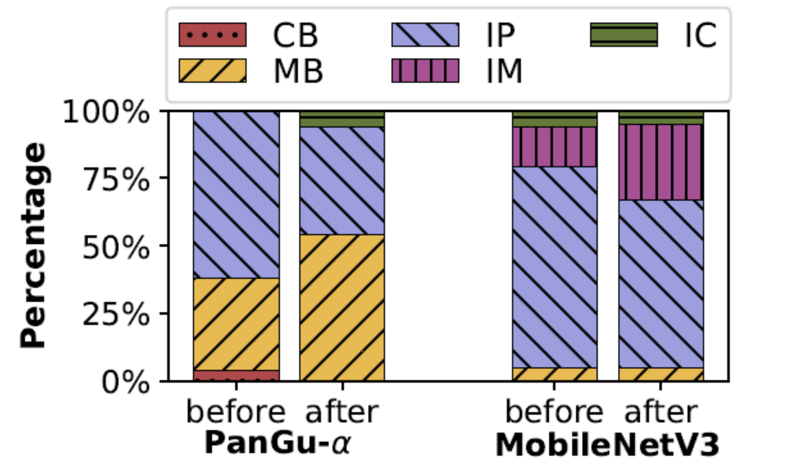
In MobileNetV3 inference, Our operator optimizations perform well with speedups of 1.06-4.31×.

More cases can be found in the paper.

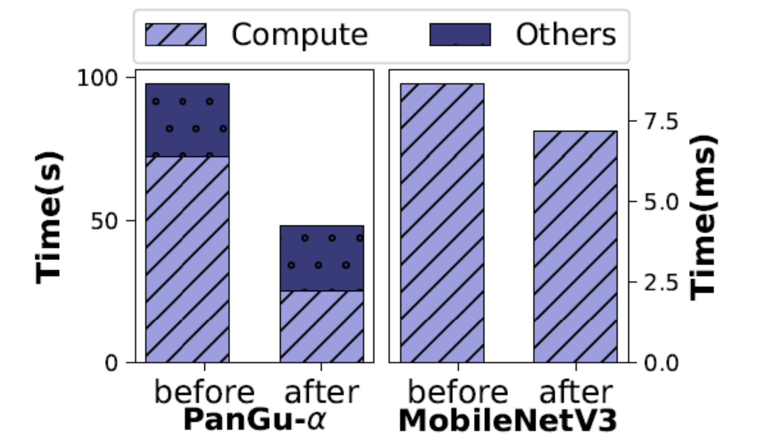
Outline

Evaluation on End-to-End Optimization

Device: Ascend 910 (Training); Ascend 310 (Inference)
Workloads: 100B PanGu-α (Training); MobileNetV3 (Inference)



Bottleneck cause

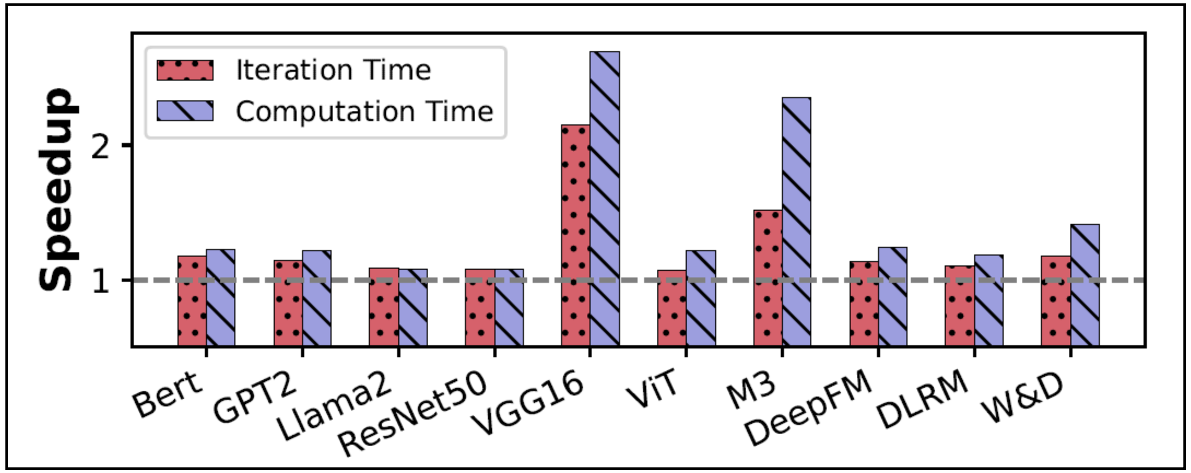


Execution time

Training	The ratio of *insufficient parallelism* reduced by 21.38%.	The *iteration time* speedup is 2.04x.
Inference	The ratio of *insufficient parallelism* reduced by 11.61%.	The *total time* speedup is 1.21x.

Overall Optimization Results

Type	Model	Parameter	Dataset	#NPUs
Vision	MobileNetV3(M3)	5.4M	ImageNet2012	8
	ResNet50	25.6M		
	ViT	86M		
	VGG16	138.4M		
NLP	Bert	110M	WikiText2	8
	GPT2	355M		
Recommendation	DeepFM	16.5M	Criteo	8
	Wide and Deep(W&D)	75.84M		
	DLRM	540M		
LLM	Llama 2	7B	WikiText2	8
	PanGu-α	100B	1.1TB Chinese Dataset	128



Our optimizations cover 11 different models.

Computation time speedups range from 1.08-2.7×.

Iteration time speedups range from 1.07-2.15×.

Outline

Conclusion

1. We propose a component-based roofline model and underutilization analysis to identify the operator bottlenecks on Ascend.

2. Through in-depth operator optimization case studies, we guide users on how to complete optimization.

3. Based on extensive practical optimization experiments, we share our practical insights and valuable experiences.

Future Work

1. The component-based roofline model can extend to other DSAs like TPU.

2. Depth studies of hardware architecture, especially its interaction with the software.

Thanks

Q&A

yuhangzhou@smail.nju.edu.cn