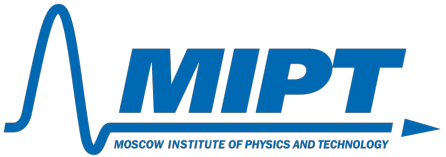


Vectorization Of Narrow Matrix Multiplication for Ascend AI Inference Acceleration

The 16th IEEE International Conference on Cloud Computing Technology and Science

Shurygin Anton, PhD Student
Frolov Aleksandr

Introduction

- Optimizing matrix multiplication (MatMul) is crucial for the efficiency of neural networks
- Huawei Ascend NPU series is one of the leading platform
- Cube Unit (AIC) mainly performs matrix-related operations
- Vector Unit (AIV) is responsible for vector operations

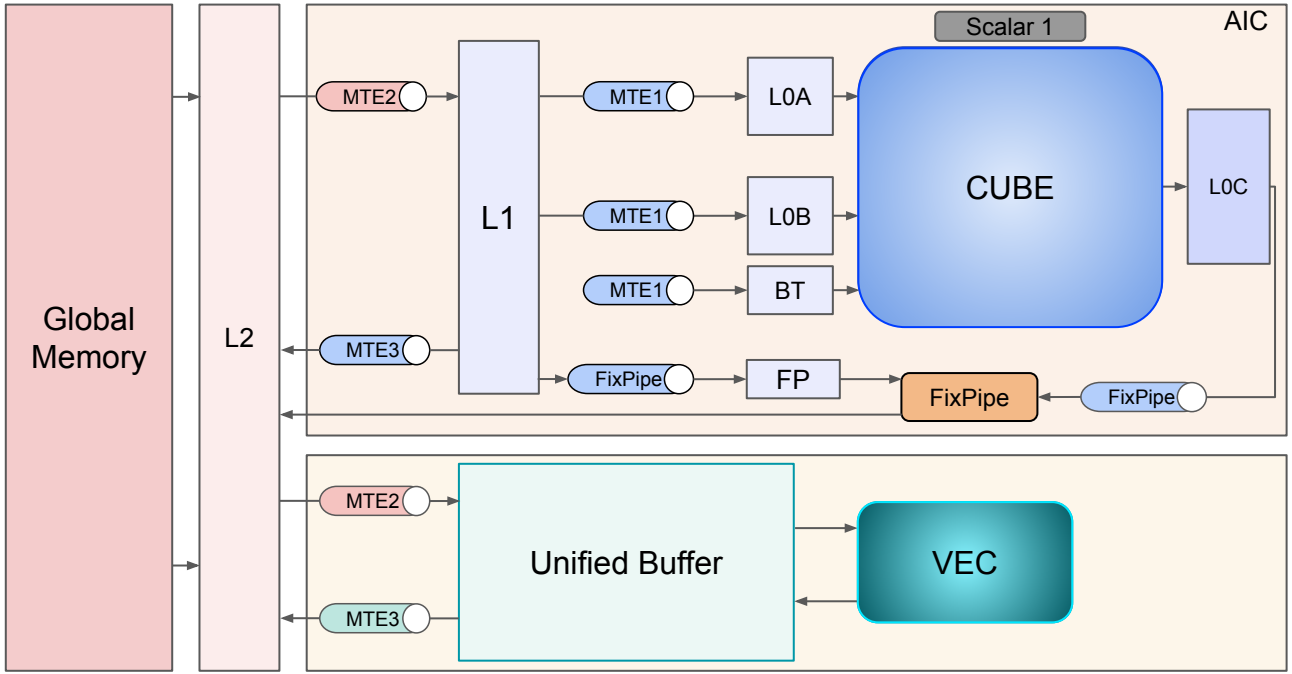


Fig. 1: AI core architecture (Ascend 910B)

Background

- Let math operator runs on Ascend AI with P physical cores ⇨ available 2P Vector Units
- We noticed, that the Vector Unit (AIV) remains idle for most of the operator runtime

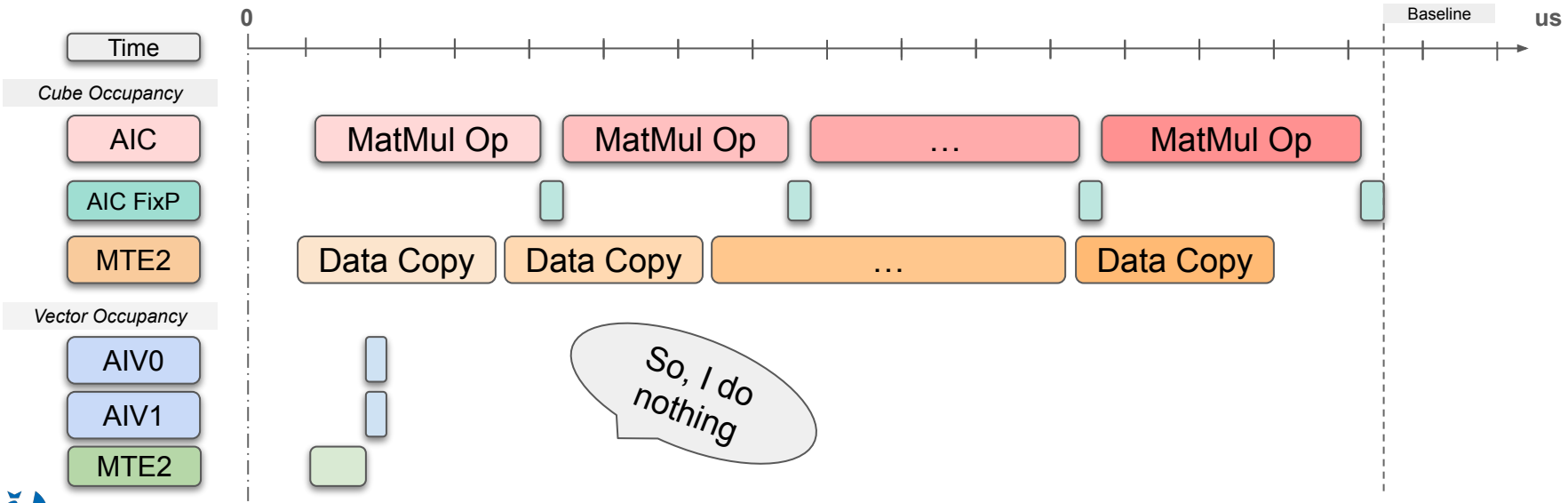
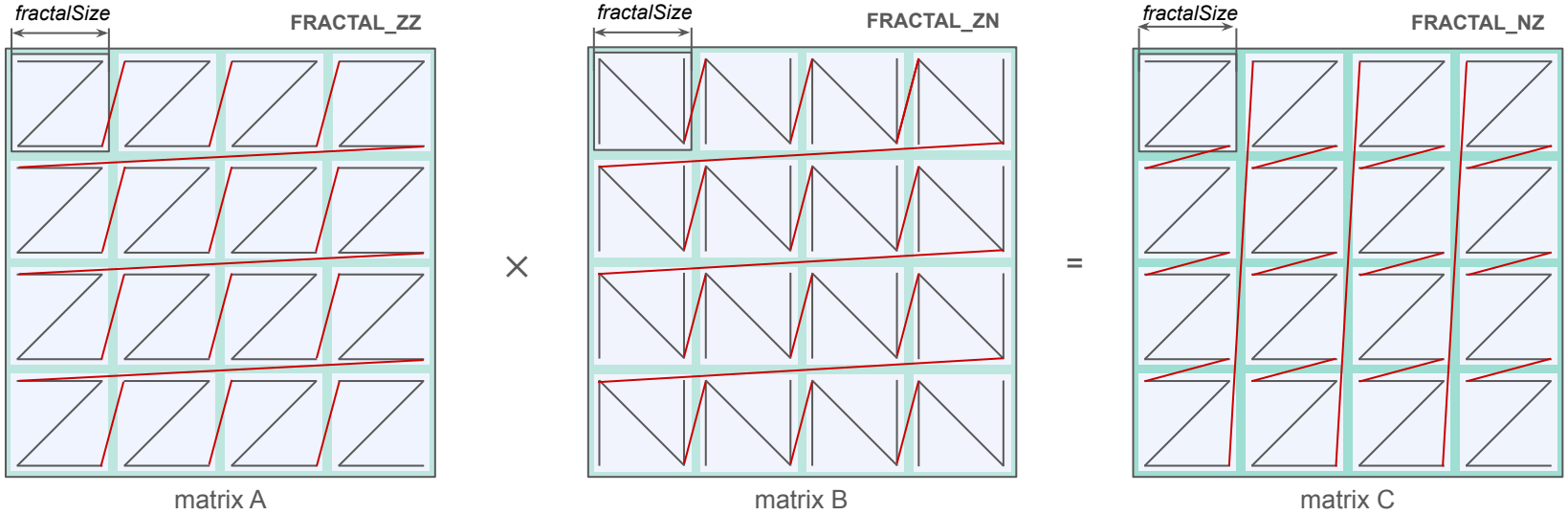


Fig. 2: Scheme of operator runtime profiling on the specific AI core

Problem Statement

- It is often necessary to split the matrix into tiles:
  - Limited capacity on the chip cache
  - Optimal distributing computations across AI cores



Problem Statement

- It is often necessary to split the matrix into tiles:
  - Limited capacity on the chip cache
  - Optimal distributing computations across AI cores
- In case matrix-vector multiplication **(narrow MatMul)**, the computational efficiency of the Cube Unit drops by 16 times

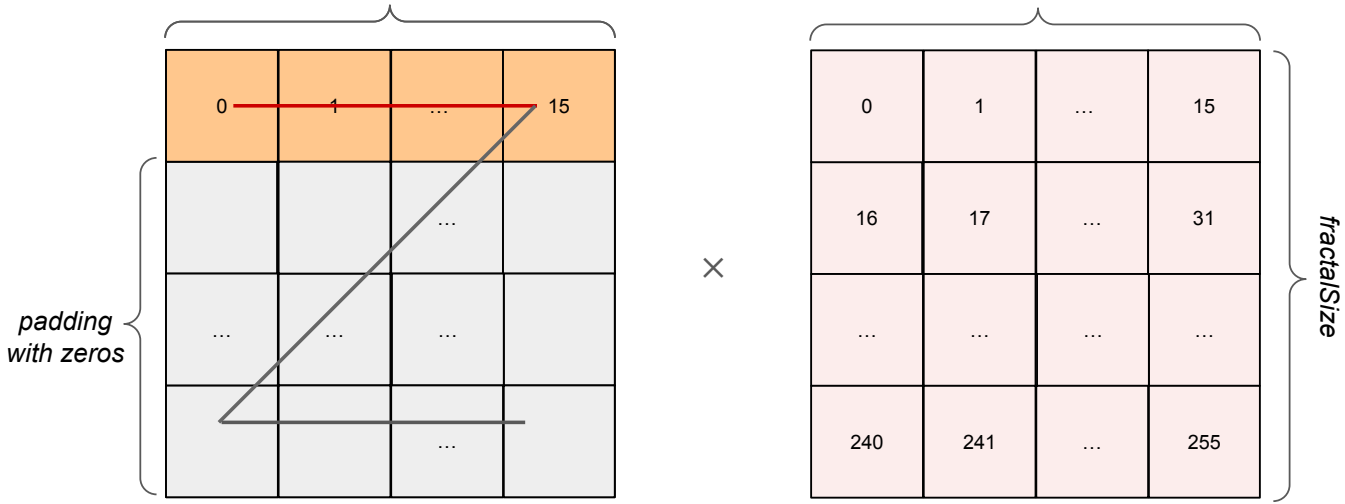


Fig. 3: Scheme of Cube Unit data processing in case of excessive padding

Proposed Optimization

- In fact, a single AIV is approximately 64 times slower than a single AIC
- Offloading MatMul A × B from AIC to AIV is worth it if following conditions:
  - **Data Independence**: subsequent operator computations on AIC must not depend on the results of Vector MatMul
  - **Narrow Dimension**: one of the matrices, either A or B, is a vector.
  - **Vector Unit is idling**: $V_{idling} \approx 2P$

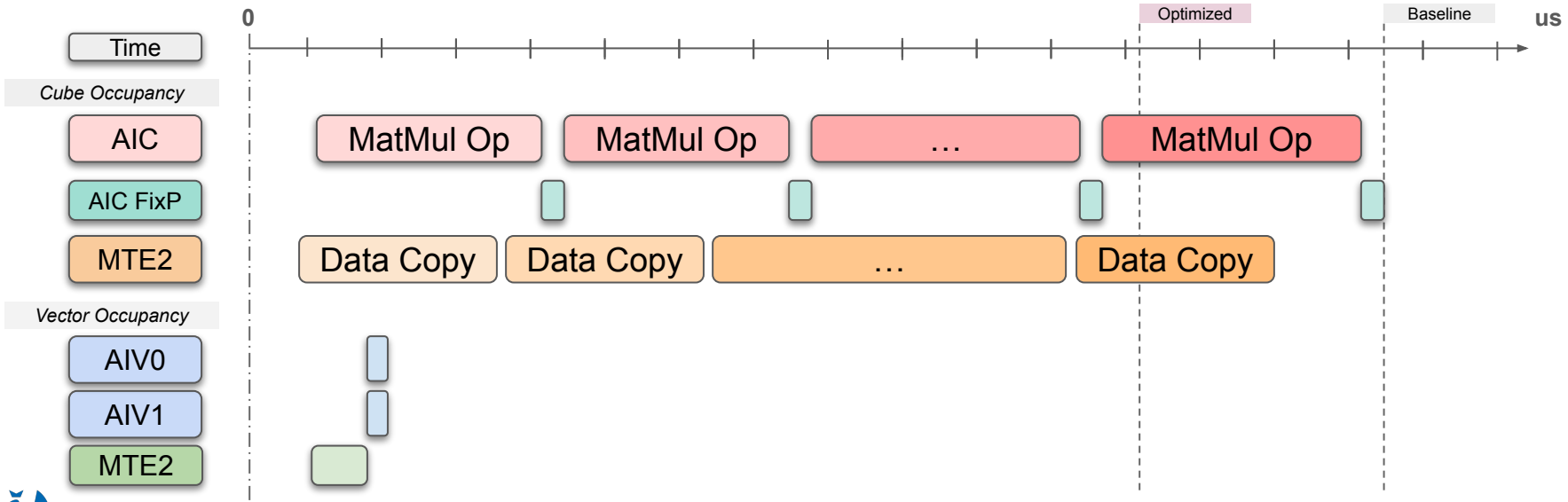


Fig. 4: Operator runtime on the specific AI core after MatMul offloading on AIV

Proposed Optimization

- In fact, a single AIV is approximately 64 times slower than a single AIC
- Offloading MatMul A × B from AIC to AIV is worth it if following conditions:
  - **Data Independence**: subsequent operator computations on AIC must not depend on the results of Vector MatMul
  - **Narrow Dimension**: one of the matrices, either A or B, is a vector.
  - **Vector Unit is idling**: $V_{idling} \approx 2P$

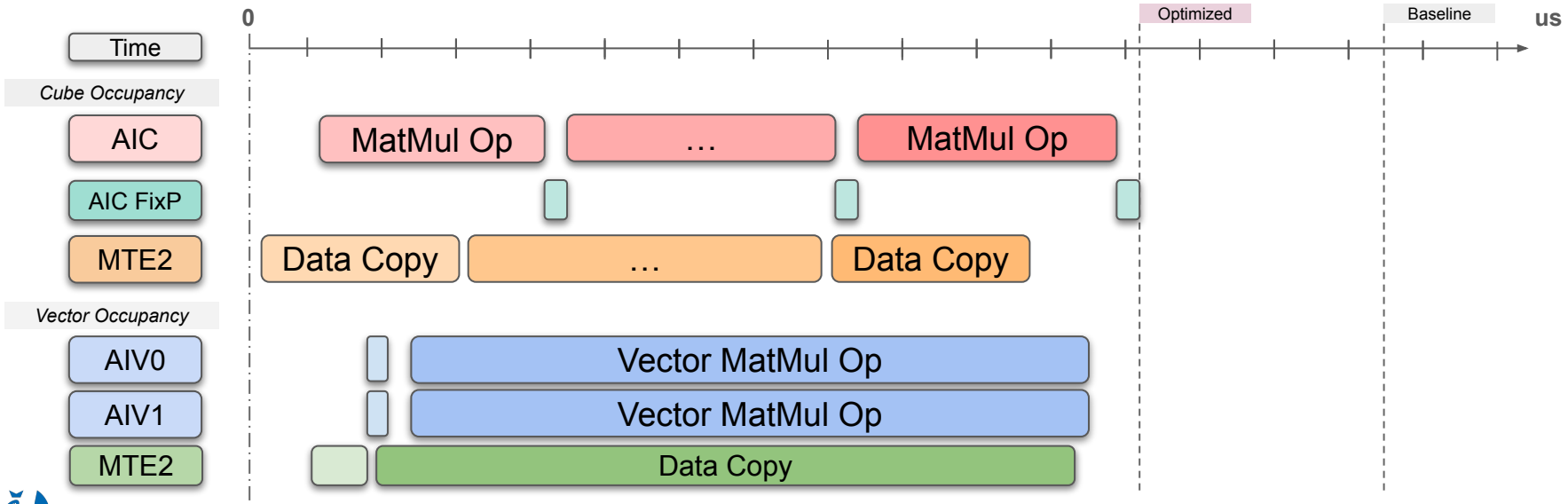


Fig. 4: Operator runtime on the specific AI core after MatMul offloading on AIV

Generalized Algorithm

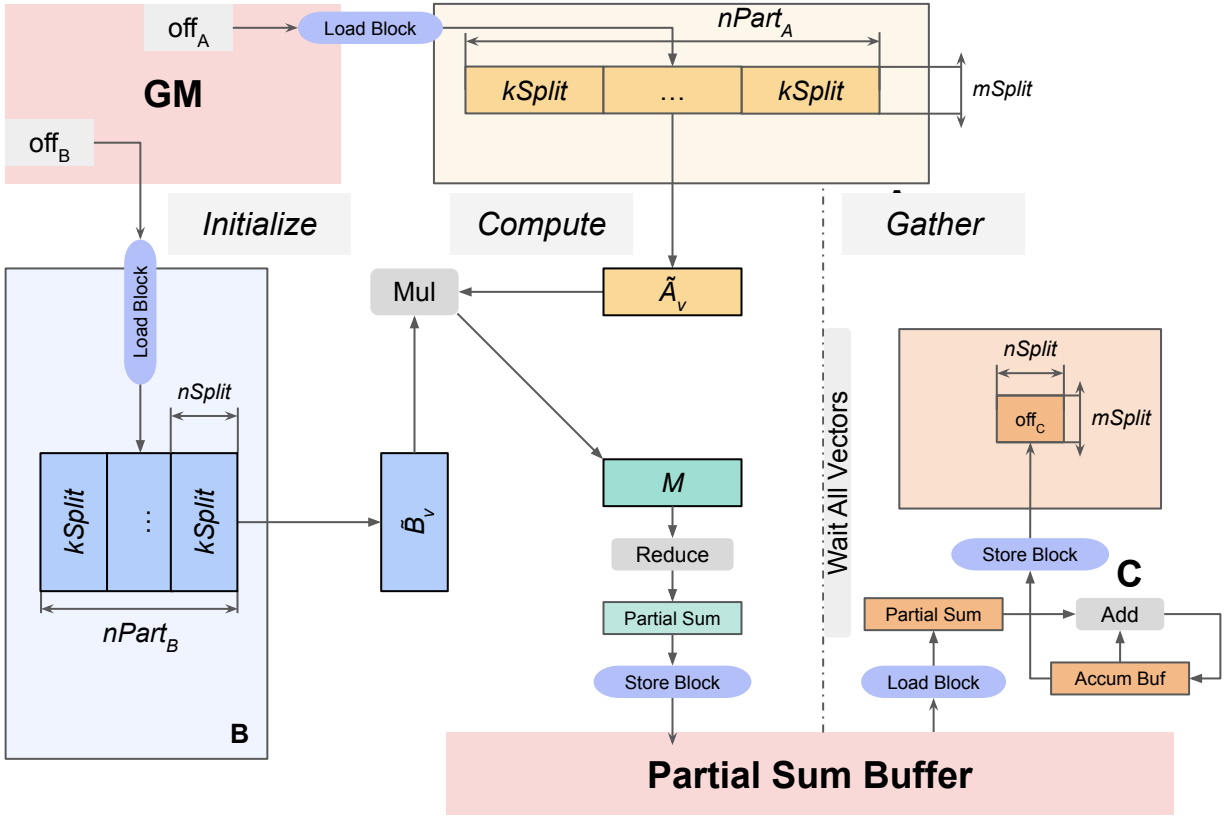


Fig. 5: Generalized MatMul algorithm scheme on AIV

MLA Operator Optimization

- This optimization was specialized to tune performance MLA DeepSeek-V3 operator
- We analyzed baseline MLA operator runtime on cycle-accurate simulator
- We found out two main candidates to offload on AIV
  - **Data independence** condition is met
  - **AIV idling** is also satisfied
  - Target case – single token, i.e. *batch size = 1*

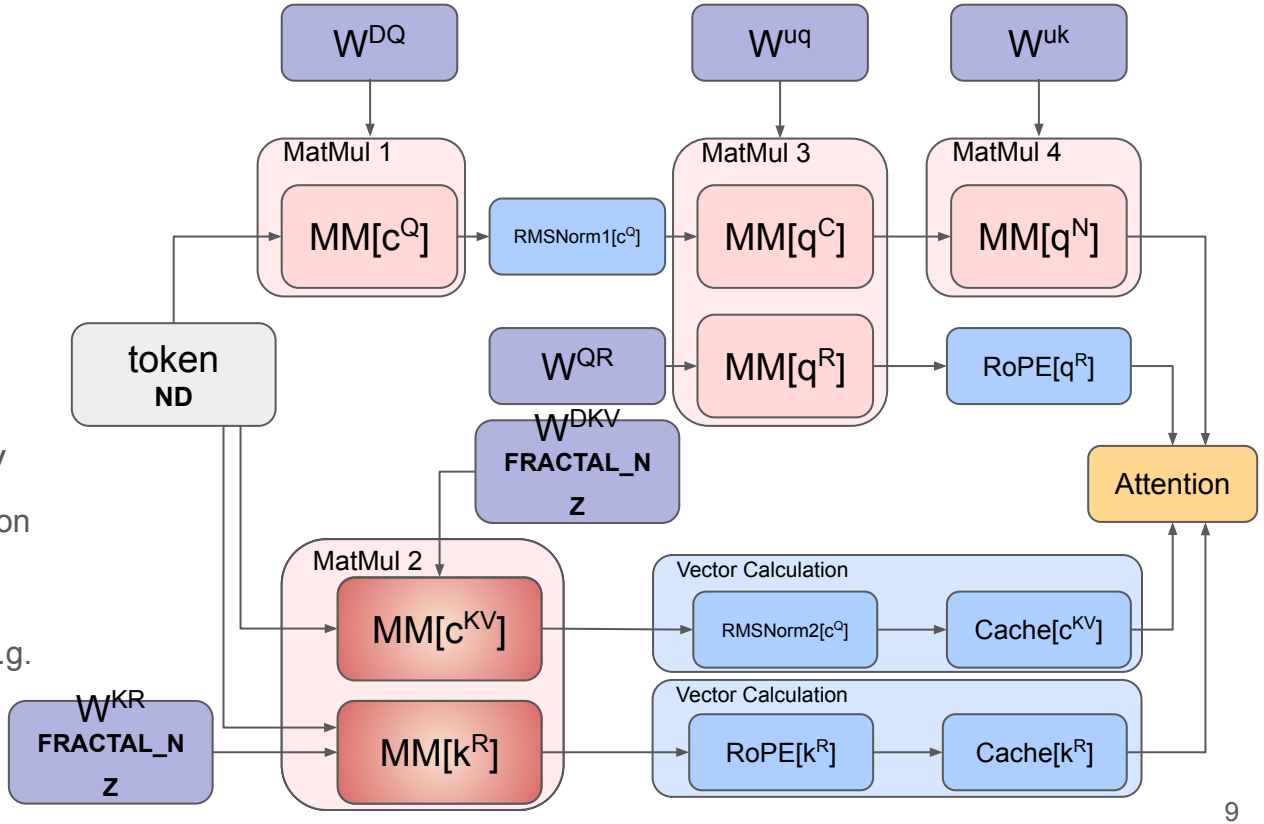


Fig. 7: Optimized MLA operator architecture (single token case).

Fig. 6: Baseline MLA operator architecture (single token case).

MLA Operator Optimization

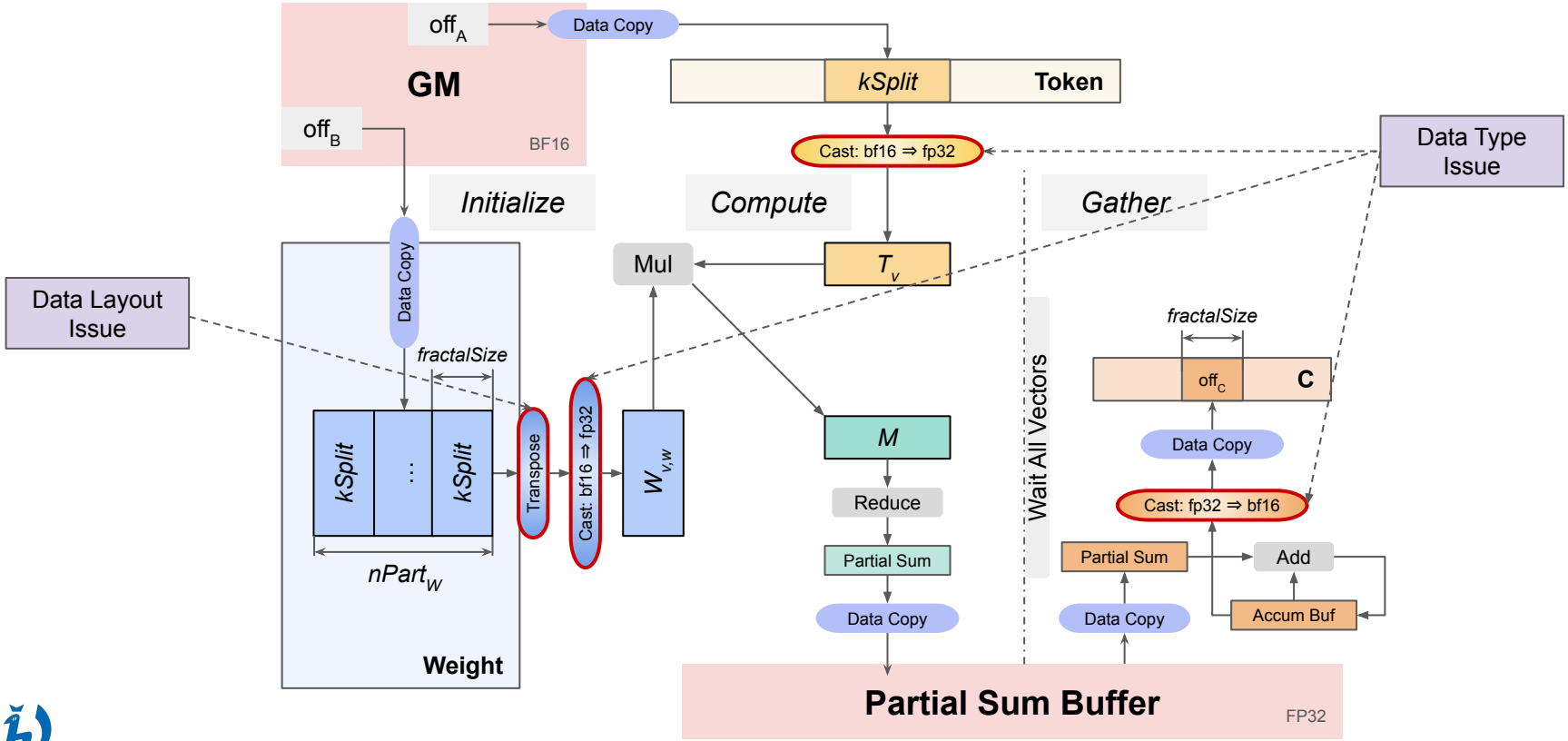


Fig. 8: Specialized MatMul Algorithm in DeepSeek-V3 MLA operator

Integration Issues

- It was necessary to maintain the same as when calculating MatMul 2 operation on Cube Unit:
  - Data layout in Global Memory
  - Calculation accuracy

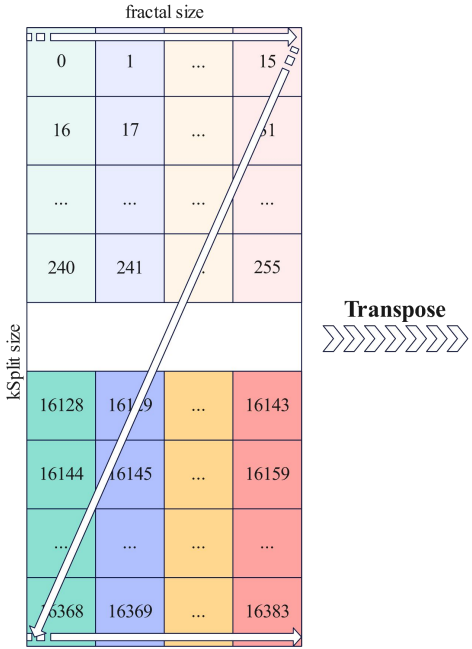


Fig. 9: Fractal Column Transposition Scheme

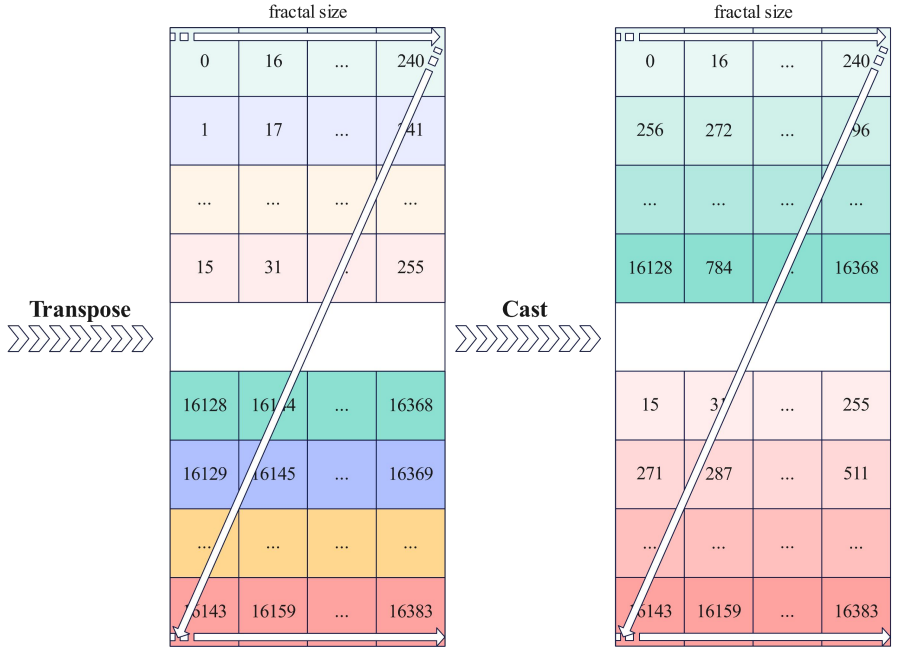


Fig. 10: Fractal Column Cast Scheme

Evaluation

- We tested the developed optimization both on a simulator and on real hardware.
- In both cases we see a performance gain:
  - CA Simulation – 17 %
  - Ascend 910B – 20%

Timing simulation (24 AI Cores)			
	Baseline,ms	Optimized,ms	Speed up,%
Total	49.574	40.782	17
AIC	49.573	40.779	17
AIV	49.532	40.105	19

Table 1: Comparison of execution time for the baseline and optimized MLA Operator on AscendAI (cycle-accurate simulation)

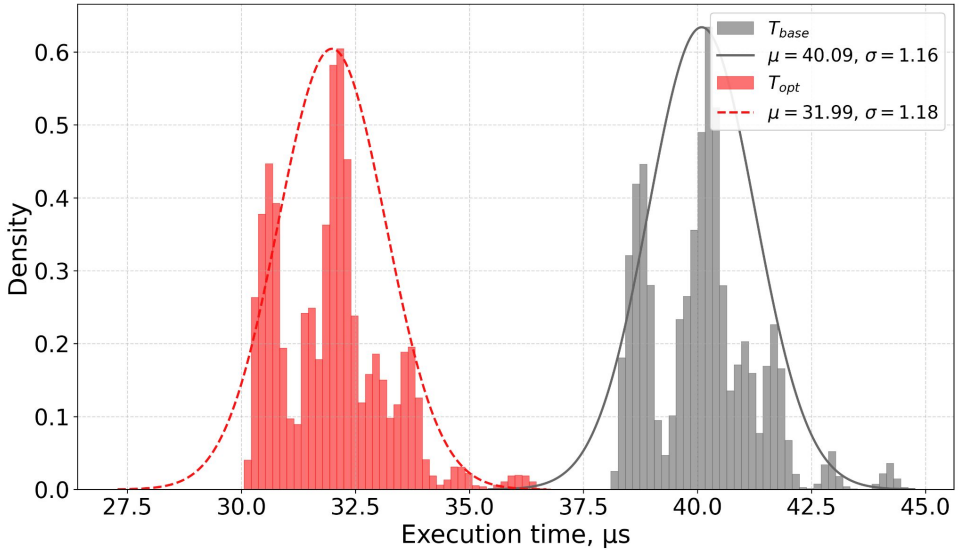


Fig. 11: MLA operator execution time distribution on Ascend910B (24 AI Cores)

Conclusion

- Our contribution:
  - We designed a generalized MatMul algorithm for AIV
  - Successful application of optimization to accelerate the MLA DeepSeek-V3 operator inference
- There are a few trade-offs:
  - Additional pressure to the MTE bus
  - The significant Unified Buffer memory usage per Vector Unit
  - May arise some data/type integration issues

Thank you for your attention

Any questions?

shurygin.aa@phystech.edu

aksdr.frolov@phystech.edu