Hardware-Assisted Virtualization of Neural Processing Units for Cloud Platforms

Yuqi Xue, Yiqi Liu, Lifeng Nai, Jian Huang
University of Illinois Urbana-Champaign, Google

PRESENTER:

SUSHIL RAJ REGMI

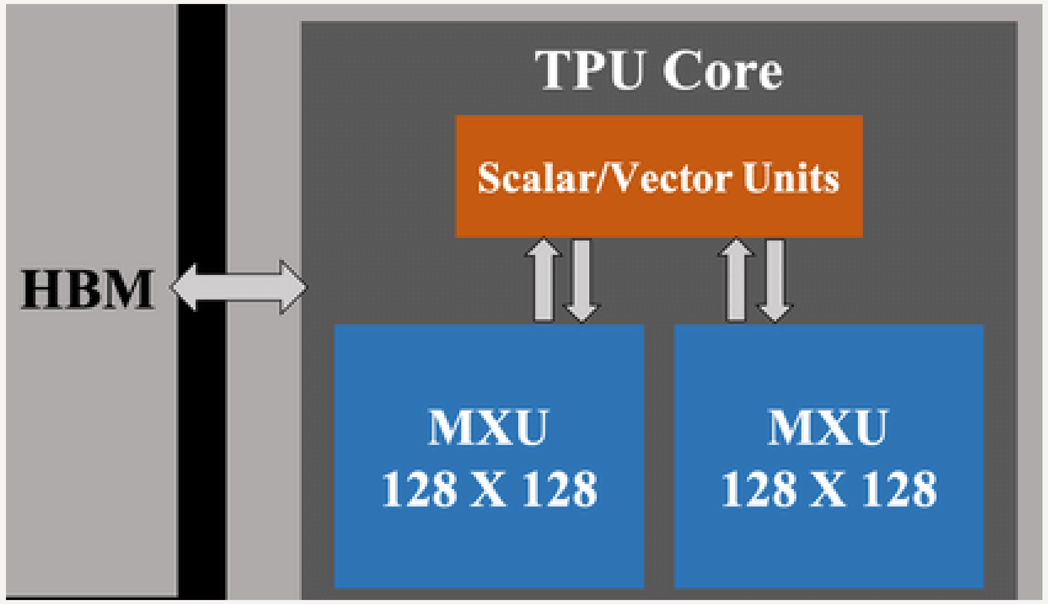
UNIVERSITY OF NORTH TEXAS

Neural Processing Units (NPUs)

- NPUs are hardware accelerators designed to accelerate neural network computations (matrix multiplication and convolution)
  - Matrix engines (MEs)

    matrix based computation, matrix multiplication, convolution
  - Vector engines (VEs)

    element-wise operations, activation functions



Virtualizing NPUs in cloud

- **Efficient Resource Utilization**

  Prevents underutilization of physical NPUs by sharing resources across multiple workloads

  Dynamically allocates idle compute units to active workloads

- **Cost efficiency**

  Enables a "pay-as-you-go" model

- **Scalability and Flexibility**

  Allows dynamic scaling of NPU resources based on real-time workload demands

- **Supports Cloud-Native AI workloads**

  Facilitates multi-tenancy and seamless execution of diverse AI workloads

- **Diverse demands on MEs and VEs**

  Accommodate diverse work-load demands

Challenges in virtualizing NPUs

- **Lack of system abstraction support for NPUs**

  Existing cloud platforms expose homogenous NPU cores to the user

  Need of a flexible system abstraction that allows users to specify the ME/VE resource and dynamic resource scheduling

- **Lack of architectural support for NPU virtualization**

  Existing NPU sharing approach sacrifice isolation or suffer from high preemption overload

  Need architectural support to achieve both improved performance isolation and NPU utilization.

- **Lack of ISA support for virtualized NPUs**

  Number of compute units need to be explicitly specified at the compilation time which limits the NPU utilization at runtime

  Need to design NPU ISA to facilitate dynamic resource scheduling

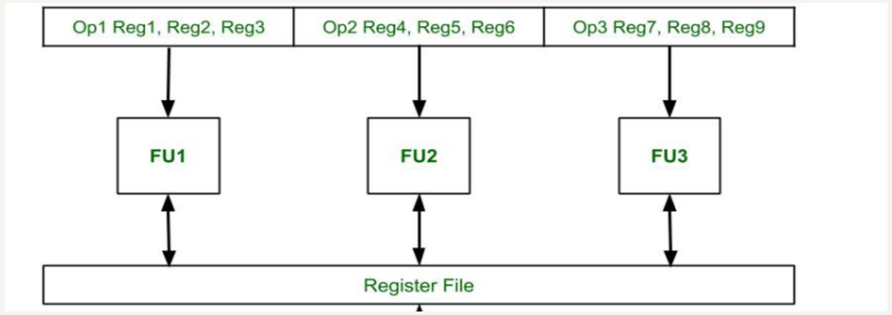
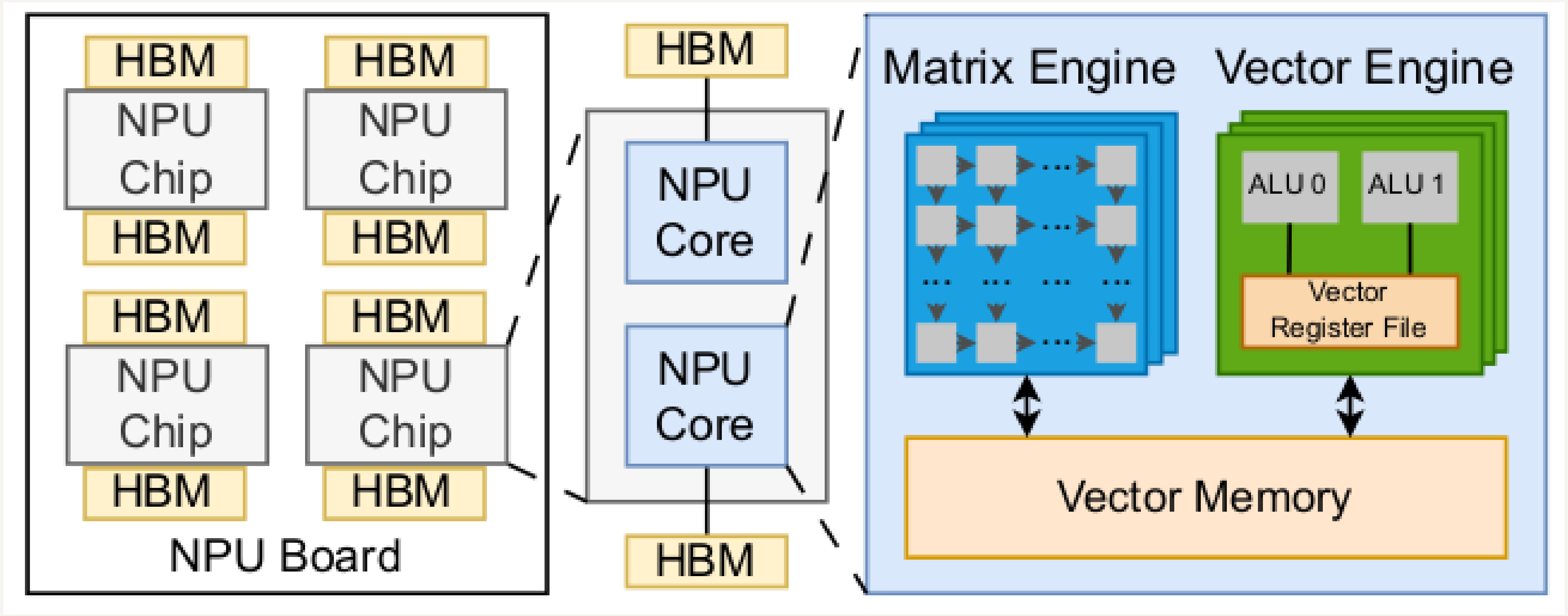


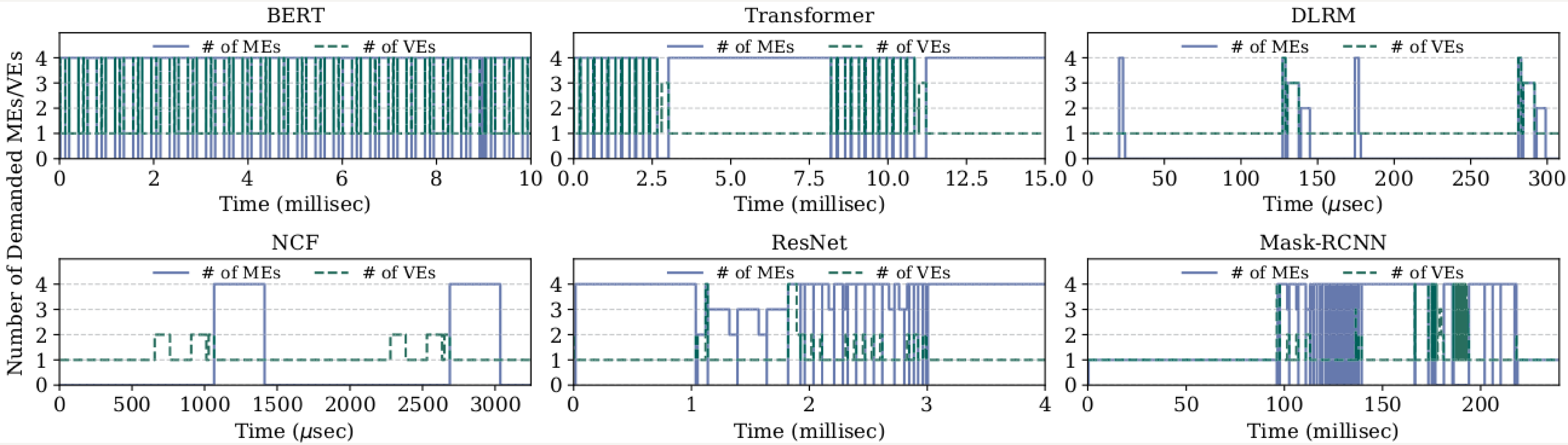
Fig: VLIW style ISA

NPU System Architecture



Characterization of ML inference services

Diverse demands on MEs/VEs

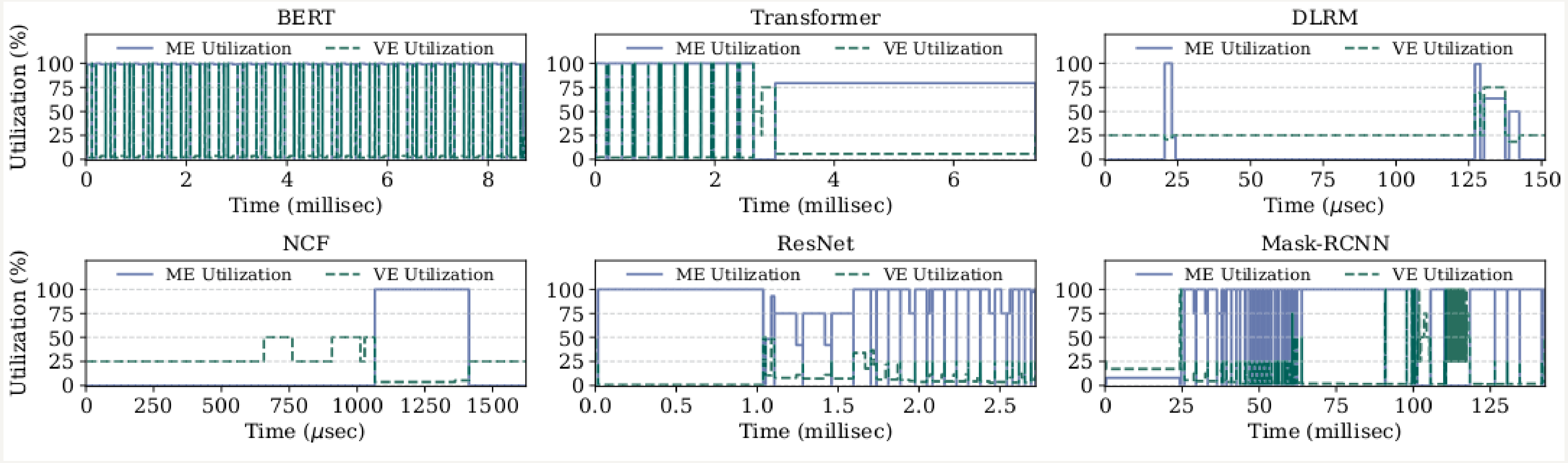


Imbalanced demands are determined by ML model architecture

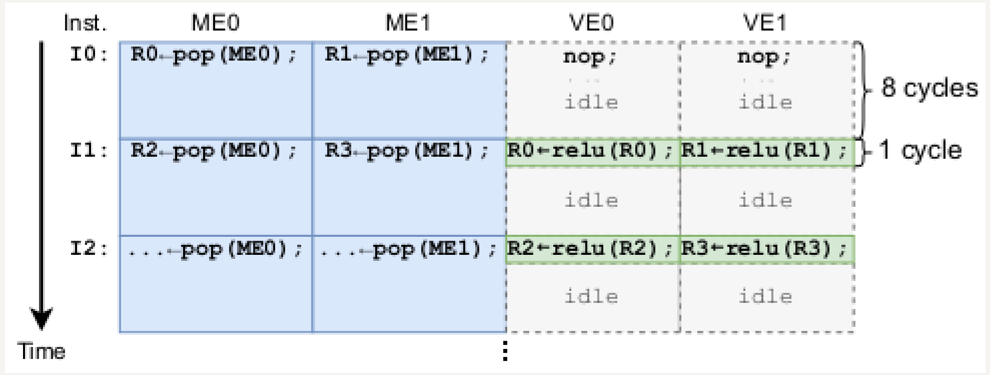
ResNet is dominated by convolutions while DLRM contains many vector operators

Characterization of ML inference services

Low NPU resource utilization



VE underutilization in ME-intensive operator



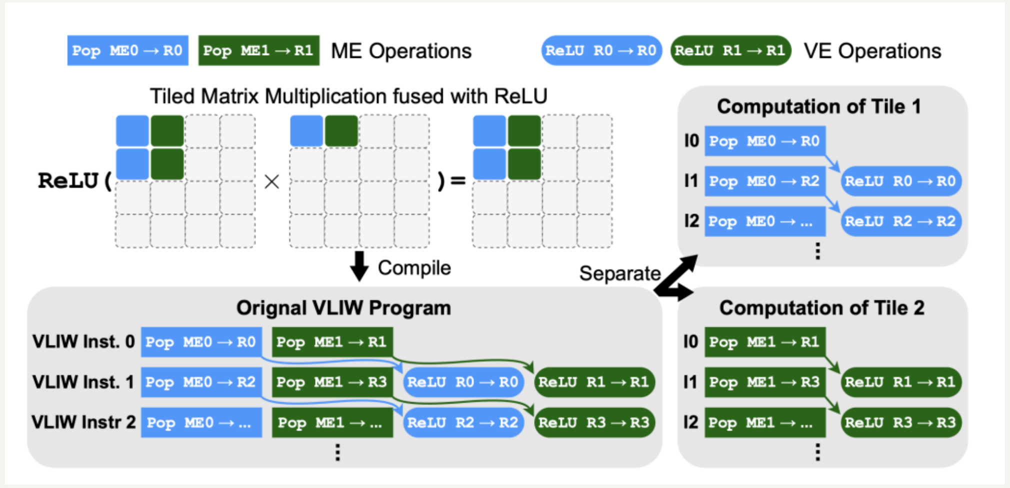
Opportunities in NPU virtualization

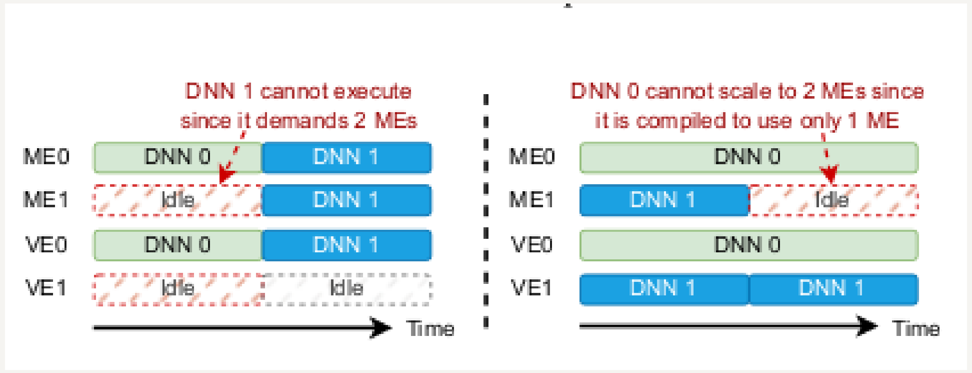
New abstraction required for fine-grained virtualization

- Need to provide flexibility to users to specify the number of MEs and VEs
- Need to enable dynamic resource scheduling as there is diverse resource demands of different operators over time.

ISA limitations for enabling virtualized NPU scheduling

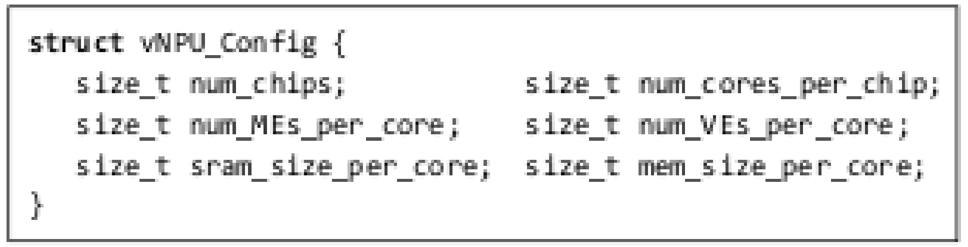
- Statically scheduled ISA cannot fully exploit hardware resource at runtime.
- Unnecessarily couple the control flows of all MEs in tensor operator




DNN 1 cannot execute since it demands 2 MEs
ME0
ME1
VE0
VE1
DNN 0
DNN 1
Idle
Time
DNN 0 cannot scale to 2 MEs since it is compiled to use only 1 ME
ME0
ME1
VE0
VE1
DNN 0
DNN 1
Idle
Time

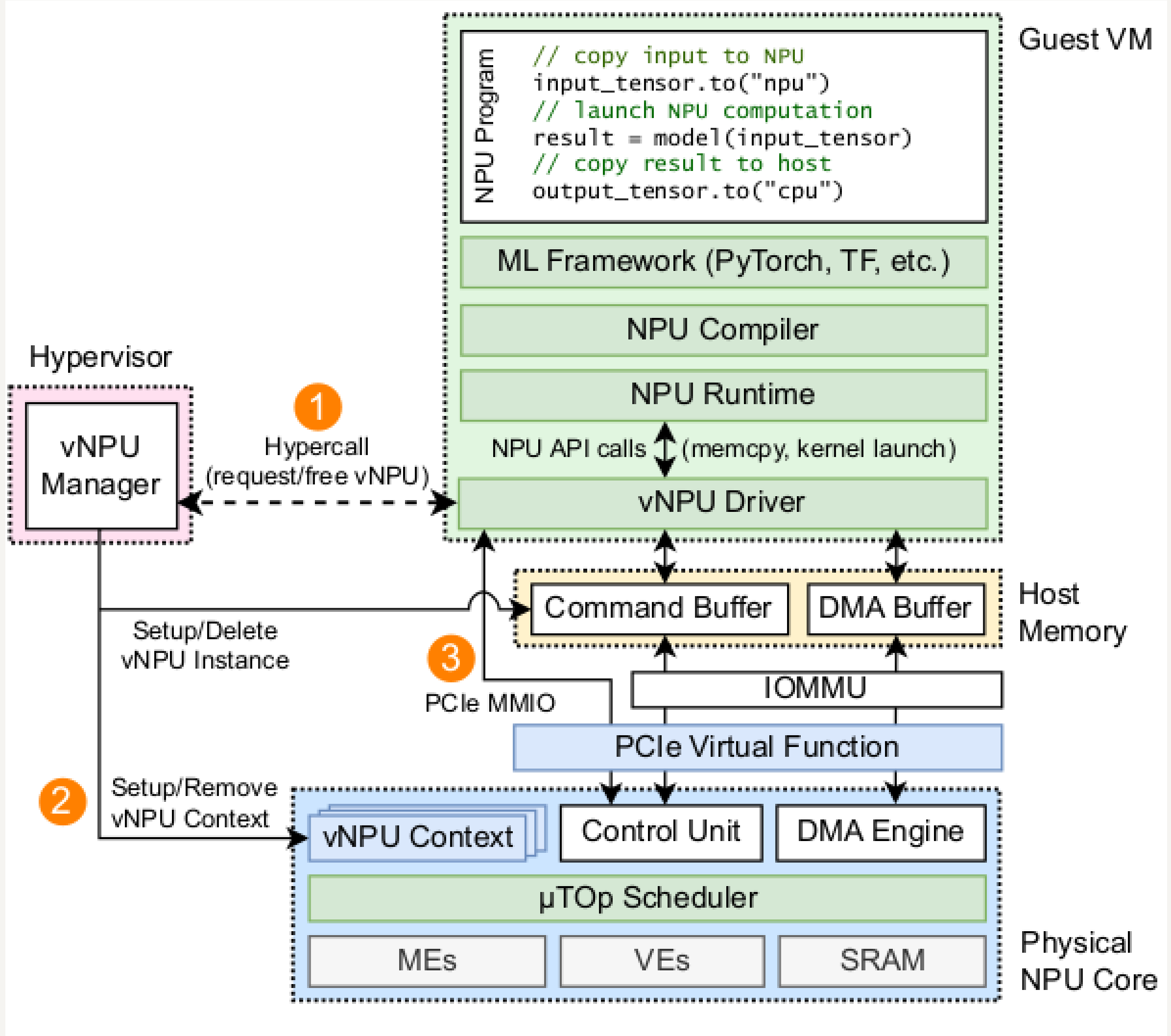
Implementation: Neu10

vNPU: abstraction for NPU virtualization



```
struct vNPU_Config {
   size_t num_chips;          size_t num_cores_per_chip;
   size_t num_MEs_per_core;   size_t num_VEs_per_core;
   size_t sram_size_per_core; size_t mem_size_per_core;
}
```

Fig. 10: vNPU configuration.



```
// copy input to NPU
input_tensor.to("npu")
// launch NPU computation
result = model(input_tensor)
// copy result to host
output_tensor.to("cpu")
```

Fig. 11: System architecture of Neu10.

Neu10: vNPU Allocation and Deallocation

Neu10 allows users to specify the total number of execution units(EUs)

ME/VE allocation for workload

m = ME active runtime/NPU total runtime, v = VE active runtime/NPU total runtime

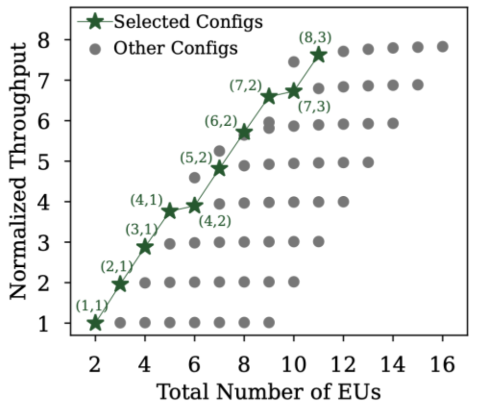
When m < 0.5, allocate $\sqrt{m/(1-m)}$ time more MEs than VEs

When v < 0.5, allocate $\sqrt{(1-v)/v}$ time more MEs than VEs

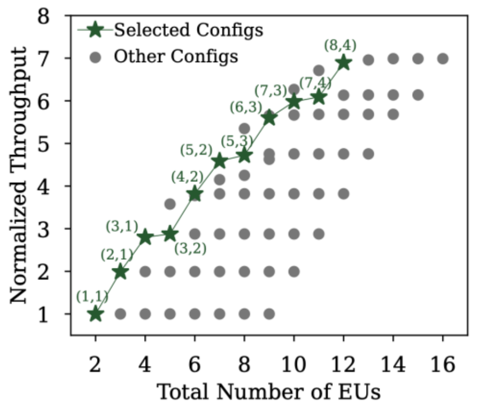
When m > 0.5 and v > 0.5, allocate the same number of VEs and MEs

NeuIO: vNPU Allocation and Deallocation

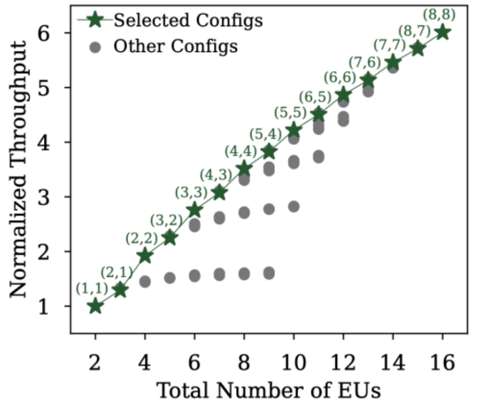
Cost-effective analysis



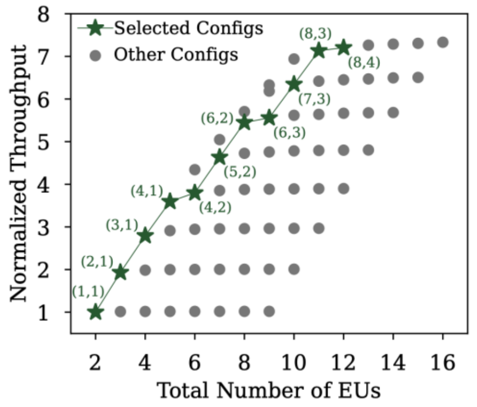
(a) BERT (batch size 32).



(b) ResNet (batch size 32).



(c) EfficientNet (batch size 32).



(d) ShapeMask (batch size 8).

Neu10 allocation algorithm selects configuration with better performance than others for same number of EUs

NeuIO: vNPU mapping

vNPUs with many EUs and small memory collocated with vNPUs with few EUs and large memory

vNPU mapping scheme:

- Hardware isolated (spatial isolated) mapping
  - vNPU is mapped to dedicated EUs and SRAM
  - Collocates a set of vNPUs as long as the total resource requirement does not exceed the pNPU.
- Software isolated (temporal sharing) mapping
  - Multiple vNPUs temporally share the same EUs
  - Aims to load-balance the pNPUs while allowing oversubscription

NeuIO: ISA Extension for NPU virtualization

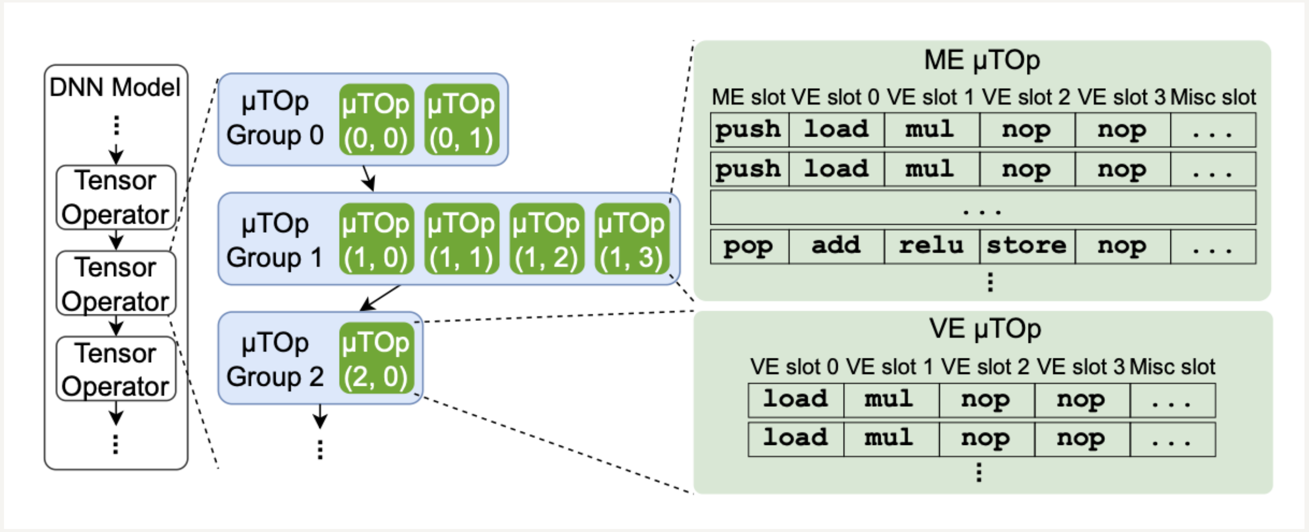
NeuISA: Enables dynamic ME/VE scheduling by decomposing tensor operators into runtime-scheduled "sub-tasks"

µTOps in NeuISA: Decouples ME/VE control flows into independent instruction sequences (µTOps)

µTOps Types:

For pNPU with nx MEs and ny VEs

- ME µTOp: contains instruction with one ME slot and ny VE slots
- VE µTOp: contains instruction with no ME slot and ny VE slots



Support fused operator (ReLU x MatMul) by organizing µTOps into a sequence of µTOps groups.

NeuI0: Architectural support for NeuISA

Hardware scheduler for NeuISA

Dynamic µTOp Selection: µTOp scheduler selects the µTOps to be executed next

Operation Scheduler: selects which operation from the instruction queues to be executed at every cycle.

Low-Overhead Preemption: Saves ME/VE states to SRAM for fast context switching when reclaiming a harvested ME.

Parallel Queues: Dedicated ME/VE queues (nx + ny) enable concurrent execution.

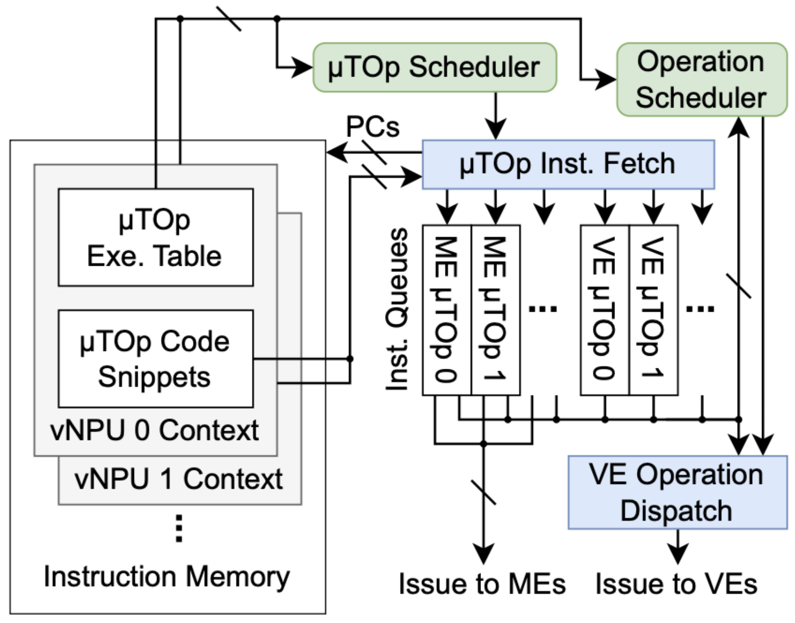
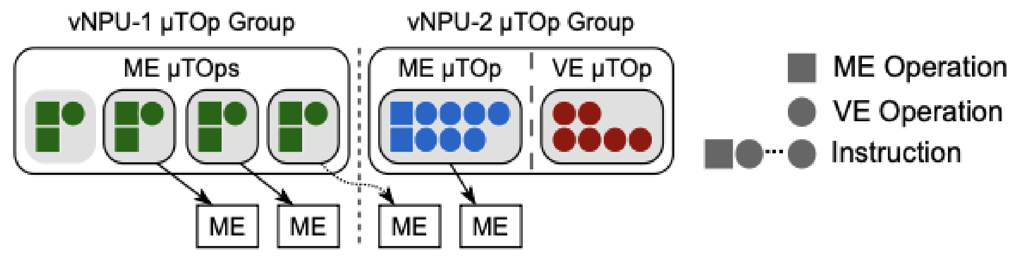
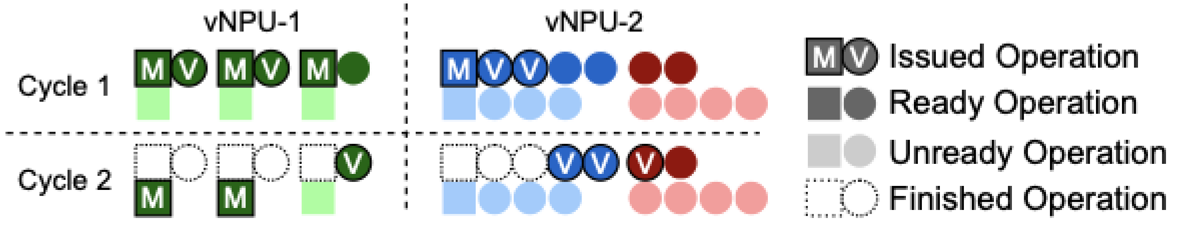


Fig. 17: NPU core pipeline frontend for NeuISA.





Neu10: Implementation

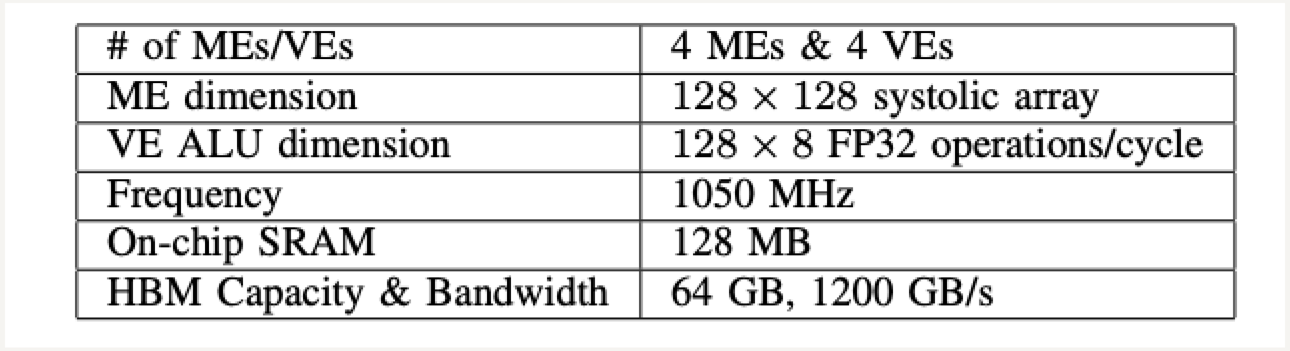
Implemented Neu10 with production level event-driven NPU simulator.

Obtained operator execution traces (ME/VE time, HBM time, tensor shapes, tile size) on real Google Cloud TPUs

Modify the frontend of simulator to implement the scheduling and harvesting policy

Prototyped the hardware scheduler in Verilog

Hardware area overhead of Neu10 is only 0.04% on a TPUv4 chip



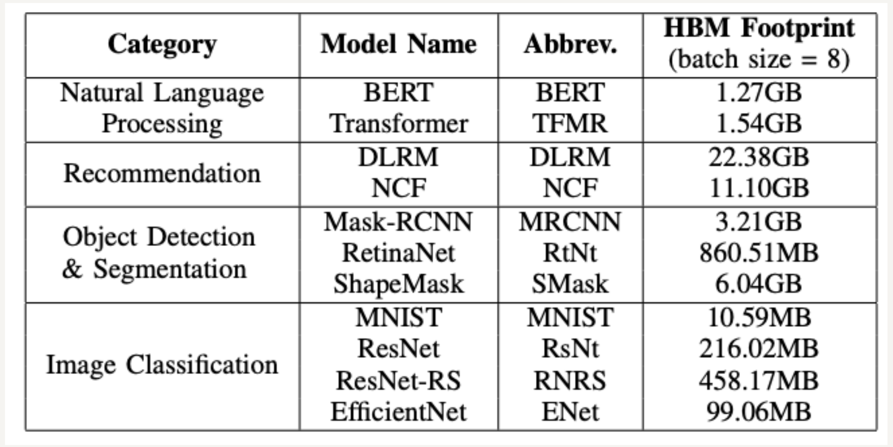
# of MEs/VEs	4 MEs & 4 VEs
ME dimension	128×128 systolic array
VE ALU dimension	128×8 FP32 operations/cycle
Frequency	1050 MHz
On-chip SRAM	128 MB
HBM Capacity & Bandwidth	64 GB, 1200 GB/s

Evaluation

Setup:

Workload combination

- **Low ME/VE contention:** (DLRM+SMask, DLRM+RtNT, NCF+RsNt)
- **Medium ME/VE contention:** (Enet+SMask, BERT+ENet,ENet+MRCN)
- **High ME/VE contention:** (ENet+TFMR, MNIST+RtNt, RNRS+RtNt)



Category	Model Name	Abbrev.	HBM Footprint (batch size = 8)
Natural Language Processing	BERT	BERT	1.27GB
	Transformer	TFMR	1.54GB
Recommendation	DLRM	DLRM	22.38GB
	NCF	NCF	11.10GB
Object Detection & Segmentation	Mask-RCNN	MRCNN	3.21GB
	RetinaNet	RtNt	860.51MB
	ShapeMask	SMask	6.04GB
Image Classification	MNIST	MNIST	10.59MB
	ResNet	RsNt	216.02MB
	ResNet-RS	RNRS	458.17MB
	EfficientNet	ENet	99.06MB

Each workload runs on a vNPU with 2 MEs and 2 VEs

Maps 2 vNPUs to physical NPU core with 4 MEs and 4 VEs

Benchmark for comparison:

PMT: temporal-sharing of the entire NPU core among multiple vNPUs.

V10: temporal-sharing of all MEs and VEs among the vNPUs with priority based preempting policy

Neu10-NoHarvest (Neu10-NH): spatial-isolated vNPUs with dedicated MEs/VEs without dynamic scheduling.

Evaluation

Tail Latency:

Neu10 improves the 95% tail latency over V10 by upto 1.56x on average

Average Latency:

Neu10 improves the average latency by 1.33x over PMT and 1.12x over V10

Throughput:

- V10 and Neu10 improve the throughput over PMT (by 1.58× and 1.62× on average), by overlapping the execution of ME intensive operators andVE-intensive operators.
- When the ME/VE contention is high, Neu10 improves the throughput of DNN workloads over V10 by up to 1.41×.

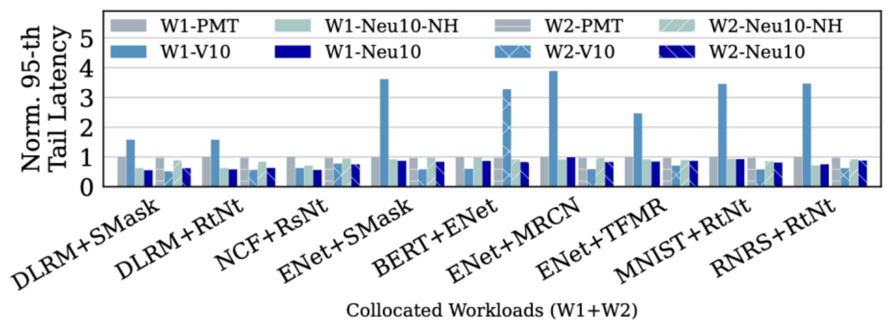


Fig. 19: 95% Percentile latency of Neu10 (normalized to PMT).

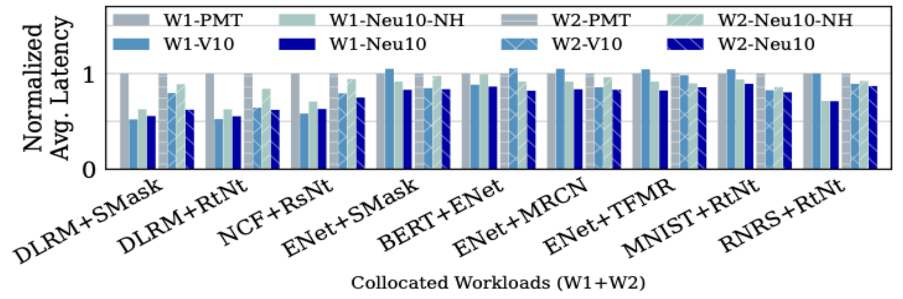


Fig. 20: Average request latency of Neu10 (normalized to PMT).

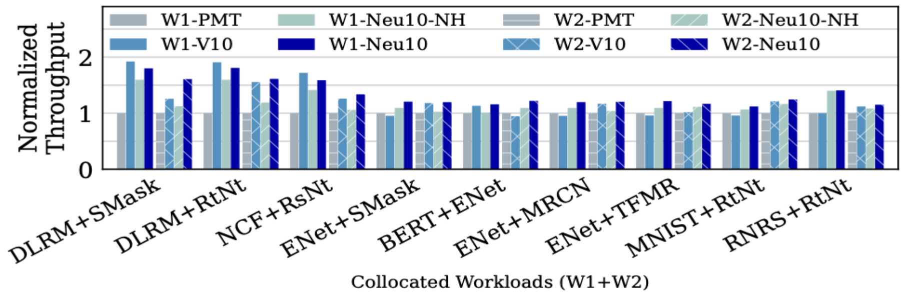


Fig. 21: Throughput of Neu10 (normalized to PMT).

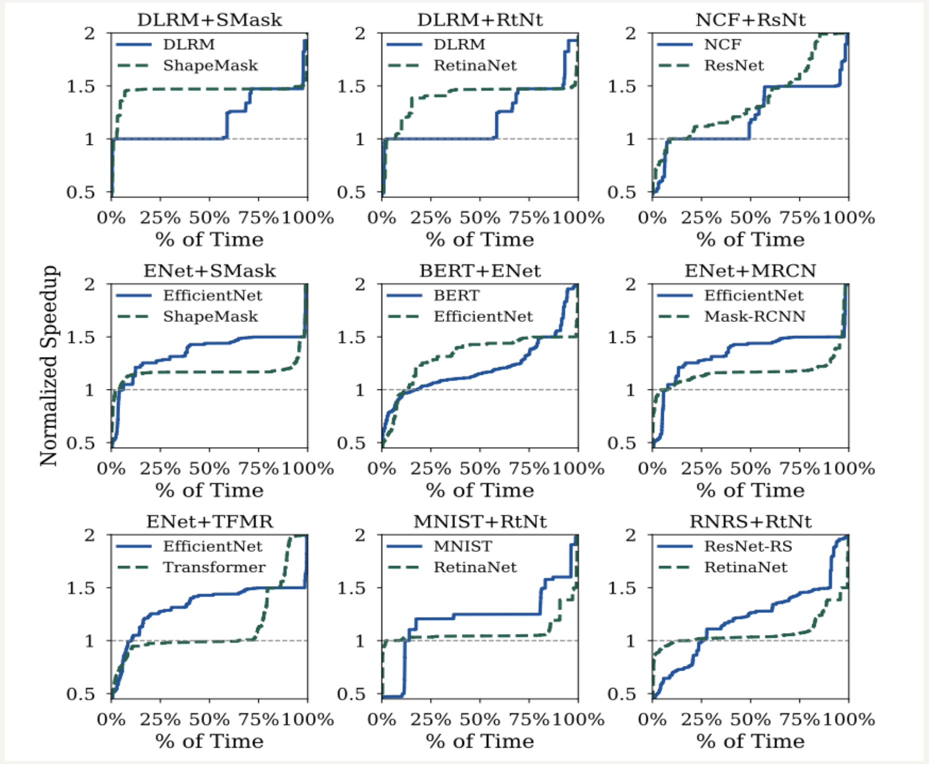
Evaluation

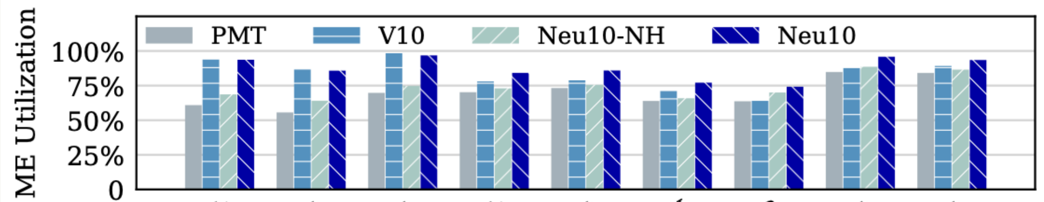
Resource Utilization Improvement:

1.26x ME and 1.2x VE utilization improvement over PMT on average

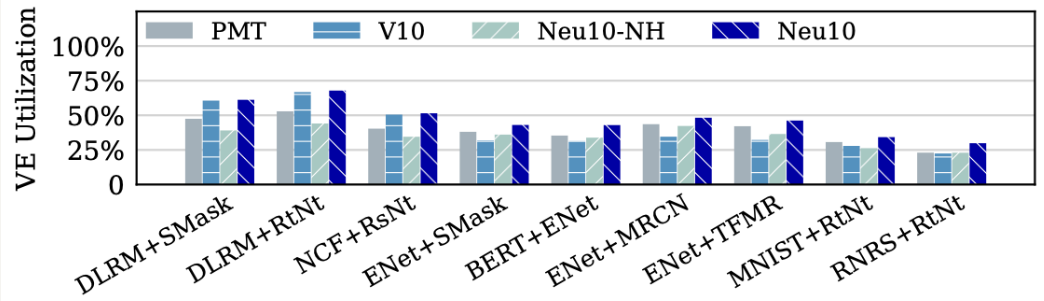
Benefit of ME/VE harvesting

Speed up of each operator in Neu10 over Neu10-NH





(a) Total ME utilization of the NPU core.



(b) Total VE utilization of the NPU core.

Evaluation

Impact of varying MEs and VEs

Neu10 benefits since there is more flexibility for dynamic ME/VE scheduling

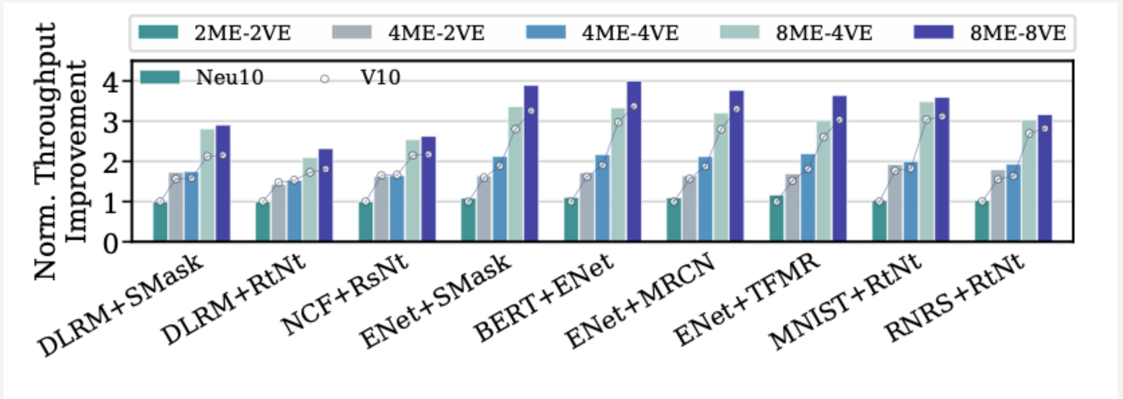


Fig. 25: Throughput improvement of Neu10 with varying numbers of MEs and VEs over V10 with 2 MEs and 2 VEs.

Impact of varying Memory Bandwidth

Comparable throughput

For memory intensive workloads, Neu10 brings more benefit with more bandwidth

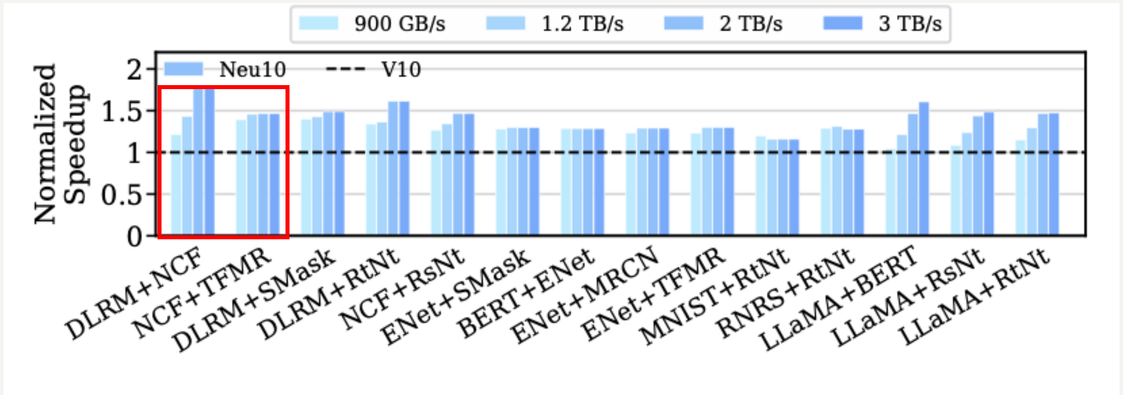


Fig. 26: Throughput improvement of Neu10 with varying HBM bandwidth (normalized to V10).

Conclusion

- Highlighted challenges in NPU virtualization, including lack of fine-grained abstraction, rigid scheduling, and architectural limitations.
- Proposed Neu10, a framework enabling fine-grained abstraction, dynamic resource allocation, and hardware-assisted virtualization.
- Demonstrated improved utilization, reduced tail latency, and better performance isolation for multi-tenant ML workloads.
- Paved the way for scalable, efficient, and flexible NPU resource management in cloud platforms to meet growing AI demands.