

18-742: Computer Architecture & Systems

Tartan: Microarchitecting a Robotic Processor

Prof. Phillip Gibbons

Spring 2025, Lecture 22

“Agents of Autonomy: A Systematic Study of Robotics on Modern Hardware”

Mohammad Bakhshalipour, Phillip Gibbons 2023

- **Architecture/Robotics gap: Only 3/150 papers in ISCA'22/Micro'22**

“Agents of Autonomy: A Systematic Study of Robotics on Modern Hardware”

Mohammad Bakhshalipour, Phillip Gibbons 2023

- **Architecture/Robotics gap: Only 3/150 papers in ISCA'22/Micro'22**
 - Quality benchmark suite needed to jump-start efforts

“Agents of Autonomy: A Systematic Study of Robotics on Modern Hardware”

Mohammad Bakhshalipour, Phillip Gibbons 2023

- **Architecture/Robotics gap: Only 3/150 papers in ISCA’22/Micro’22**
 - Quality benchmark suite needed to jump-start efforts

Table 1. Feature comparison of related work and RoWild. More ✓ is better.

Paper/Repository	Scope	End-to-End	High-Perf.	Simulator-Friendly	Multi-Platform	Versatile & Modular	Open-Source	System. Study
Lin et al. [143] Yu et al. [208]	Self-Driving Cars	✓	✓✓	Unknown	✓✓	Unknown	✗	✓
MAVBench [74]	Drones	✓	✓	✙	✓	✗	✓	✓
One-off [55, 80, 118]	Single Task	✗	✙	✙	✗	✓	✓	✗
ROS [51]	Broad	✗	✙	✗	✗	✓	✓	✗
Educational [14, 47]	Broad	✗	✗	✙	✗	✗	✓	✗
RTRBench [68]	Broad	✗	✓	✓	✗	✓	✓	✗
<i>RoWild</i>	Broad	✓	✓✓	✓	✓✓	✓	✓	✓✓

✙ Depending on the case (e.g., kernel, simulator), it can be ✗ or ✓. See Appendix §E.

RoWild Workloads

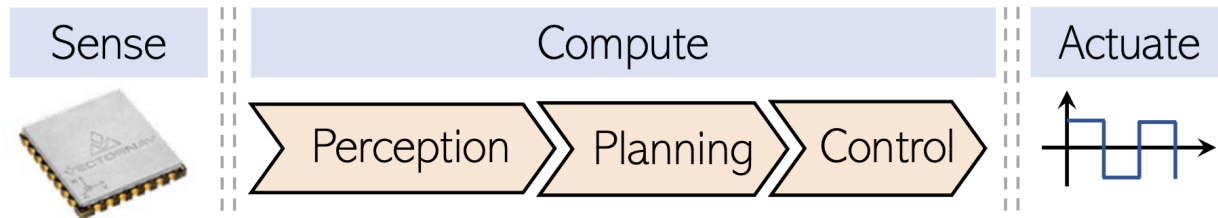
Stage	Task	Algorithm(s) in RoWild
Perception	State Estimation	MCL ^{†‡Y¶ησ} [210], AMCL ^ζ [195]
	Localization and Mapping	EKF ^{Y§τφΛ} [197], Fast ^{‡¶ζ} [158], Graph ^{∂φ} [187], ORB ^{ςϱ} [82]
	Scene Reconstruction	Point-Based Fusion ^{†‡Yκν} [205]
	Grid Generation	Probabilistic Occupancy Map ^{†κ∂} [104]
	Object Detection	MobileNet-SSD ^{‡YϱχΛ} [207]
Planning	(x, y) Search	WA ^{★†¶ζηαΠΛ} [176], GA [★] [215], RA [★] [69], IDA ^{★ς} [129], DQN [157]
	(x, y, z) Search	WA ^{★†¶ζηαΠΛ} [176], GA [★] [215], RA [★] [69], IDA ^{★ς} [129], DQN [157]
	(x, y, θ) Search	WA ^{★†¶ζηαΠΛ} [176], GA [★] [215], RA [★] [69], IDA ^{★ς} [129]
	Timed Search	WA ^{★†¶ζηαΠΛ} [176] + Backward Dijkstra [71]
	2D n -DoF Search	RRT ^{†‡Yκ} [103], RRT ^{★Yωγδ} [105], PRM ^{Y∂ψ} [191], Shortcut [107]
	3D n -DoF Search	RRT ^{†‡Yκ} [103], RRT ^{★Yωγδ} [105], PRM ^{Y∂ψ} [191], Shortcut [107]
	Reactive Planning	Behavior Tree ^{†λ∂φ} [108]
	Task Scheduling	Symbolic Planning ^{§δδ} [72]
Ctrl.	Motion Control	MPC ^{YΠ∂†καψε} [100] PID ^{Y∂ζθφ} [203], PP ^{‡Yκφ} [170]
	Parameter Learning	DMP ^{Y∂ξαθι} [131], CEM ^{§λ} [173], BO ^ϱ [119]

Real-World Robots:

[†]Spot [10]
[‡]DJI Phantom [41]
^YLoCoBot [2]
[§]Atlas [9]
[¶]AMR [48]
^ΠAscTec Pelican [7]
^ζRoomba 980 [49]
^νRoomba i7+ [50]
^ηHusky [21]
[∂]YuMi [65]
^κJackal [28]
^ξLBR [33]
^ϱPepper [53]
^σTurtleBot [60]
^ςTurtleBot3 [61]
^τMiR [34]
^φAG [31]
^ΛPioneer 3-DX [43]
^ψUR10e [57]
^χTALOS [54]
^ψKR60-3 [32]
^ωGrizzly [13]
^YSkydio [52]
^δM-200iA/2300 [18]
^εPerceptIn [207]
^αBoxbot [11]
^θUR5e [58]
^ιFranka Panda [19]
^λPAL TIAGo [59]

Table 2. RoWild’s workloads. Algorithms in **color** are characterized in this paper in the context of end-to-end robotic applications.

Robotic Software Pipeline



End-to-End Applications

Name	Mission	Resembling	Environment
DeliBot	Delivery	Spot	Our Campus
PatrolBot	Patrolling	Pioneer 3-DX	Our Campus
MoveBot	Manipulation	LoCoBot	Synthetic
HomeBot	Cleaning	Roomba i7+	Hypersim
FlyBot	Photography	AscTec Pelican	FR Campus
CarriBot	Transportation	Boxbot	Intel Lab

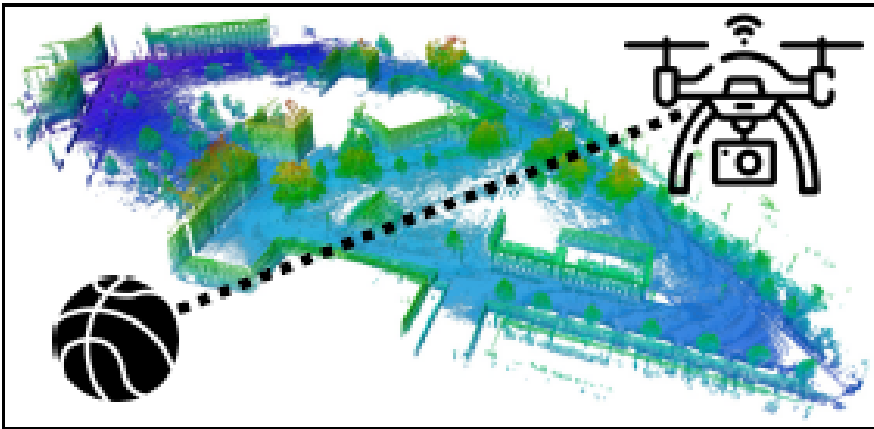
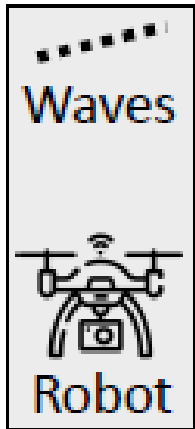
Evaluated Compute Platforms

Table 3. The evaluated compute platforms.
The prices are as of August 9, 2023.

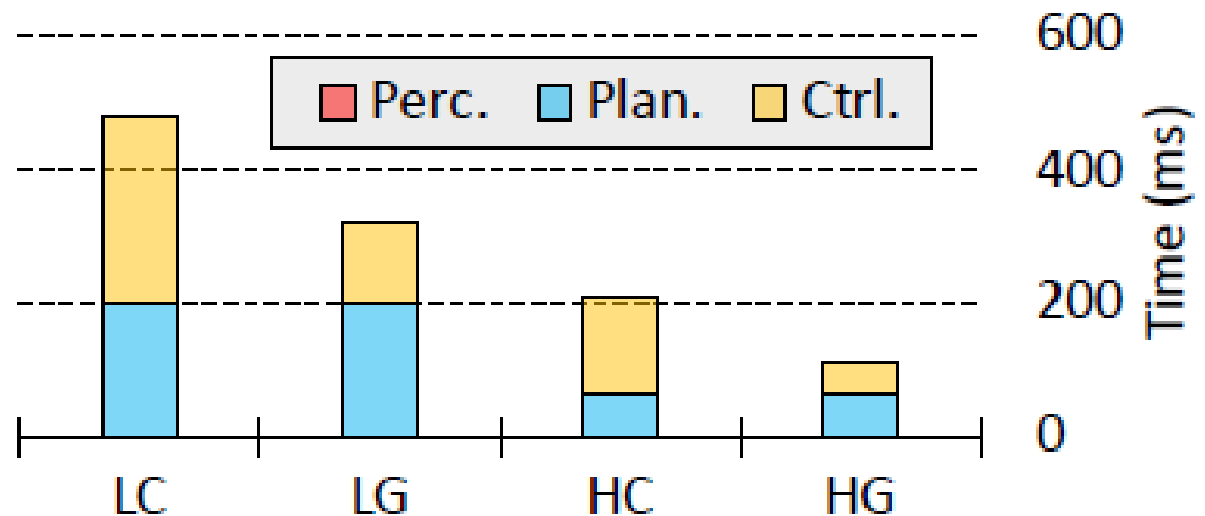
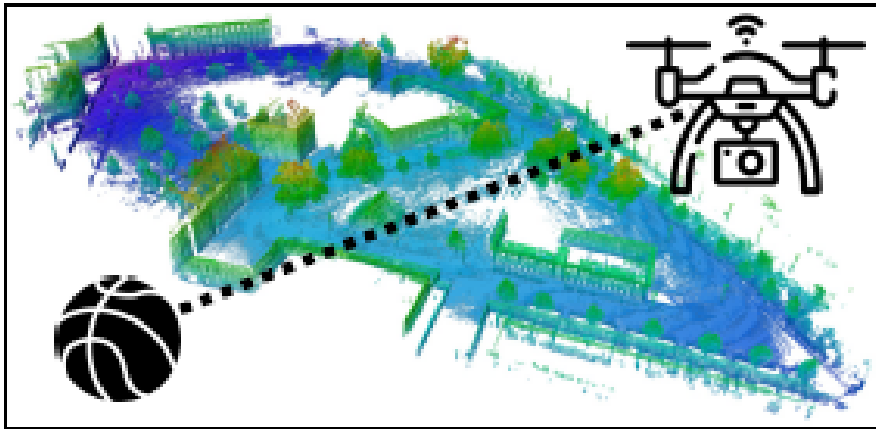
Acronym	Platform	Cores	Freq. (GHz)	TDP (W)	Memory (GB)	Price (US \$)
<i>LC</i>	ARM Cortex A57 CPU	4	1.43	10	4	149
<i>LG</i>	Nvidia Maxwell GPU	128	0.92	10	4	149
<i>HC</i>	Intel Xeon Gold CPU	20 (×2)	2.10	125	384	1493
<i>HG</i>	Nvidia Titan X GPU	3584	1.41	250	12	999

Also: Avnet Ultra96-V2 FPGA (Verilog implementations)

FlyBot: Drone Performing Aerial Photography

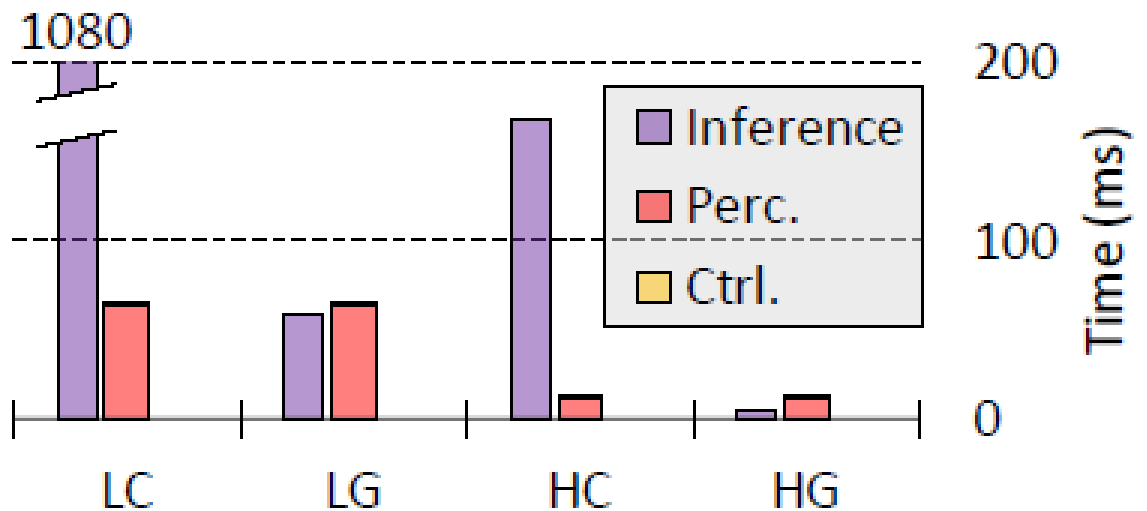
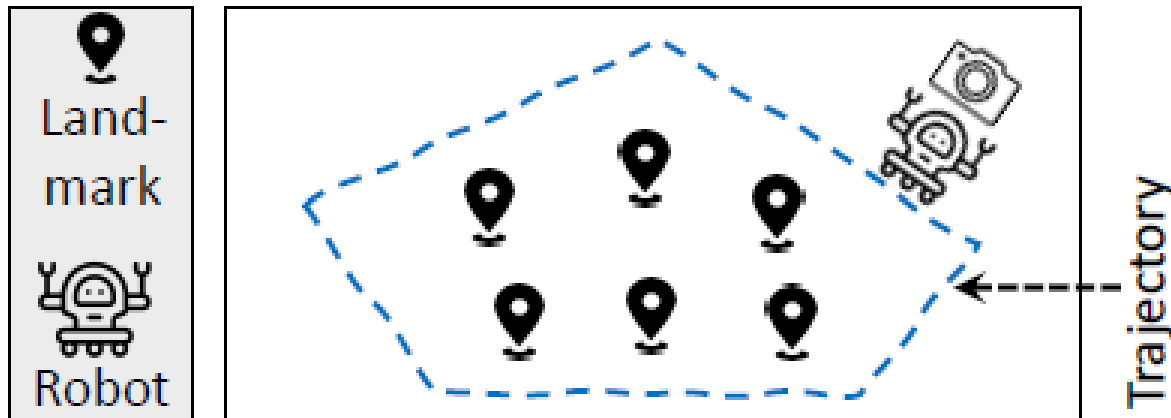


FlyBot: Drone Performing Aerial Photography



Planning & Control dominate. Control benefits from GPU.
Planning requires high single-threaded performance

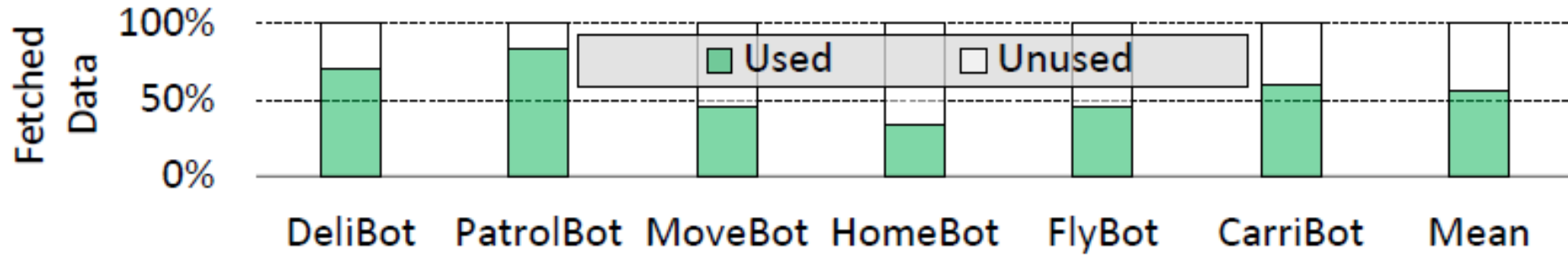
PatrolBot: Wheeled Robot Patrolling



Inference (object classification) is bottleneck for CPU-only platforms.
LG (\$149) outperforms HC (\$1493).

Challenge: Poor Spatial Locality

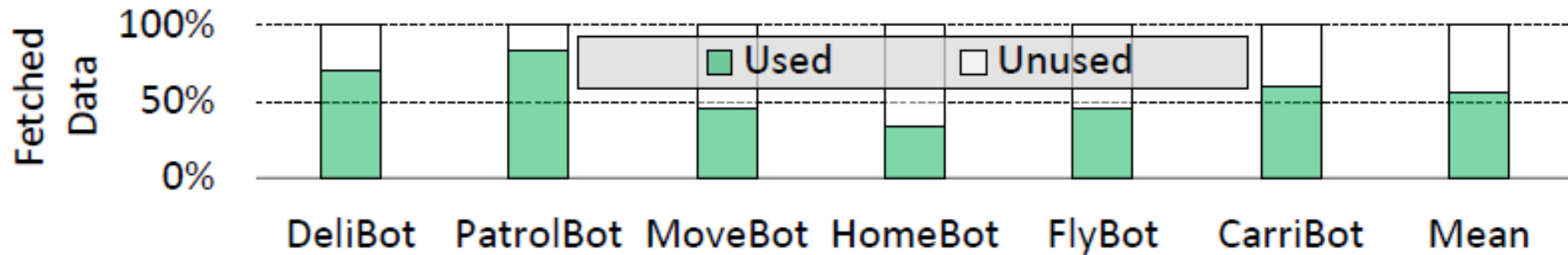
Infinite Sized Cache, No HW prefetching, 64B cache lines



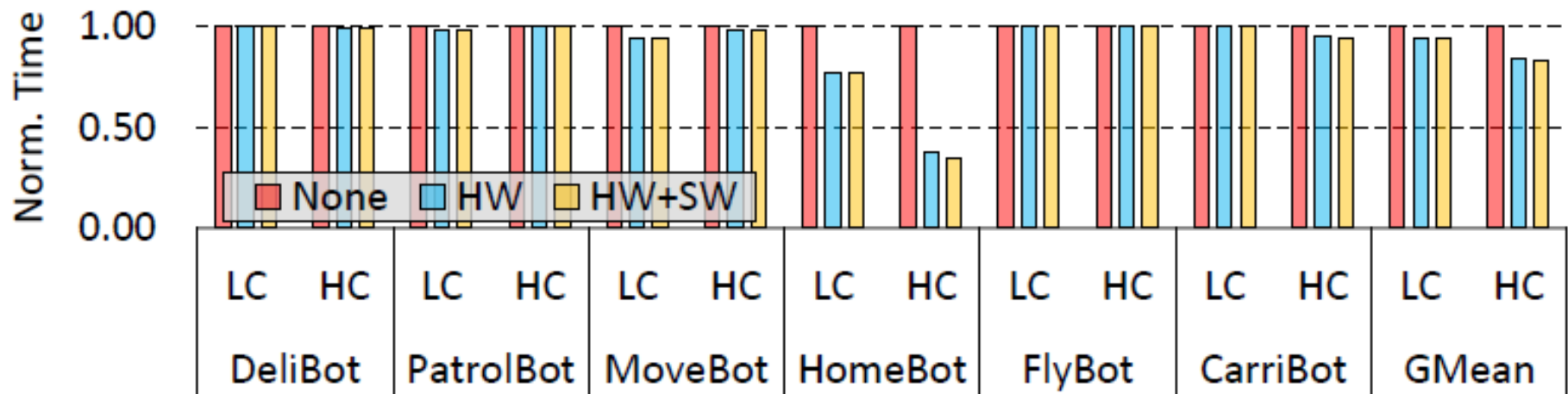
44% of fetched data never used. Existing prefetchers don't help much.

Challenge: Poor Spatial Locality

Infinite Sized Cache, No HW prefetching, 64B cache lines



Challenge: Prefetchers Inadequate



44% of fetched data never used. Existing prefetchers don't help much.

“Tartan: Microarchitecting a Robotic Processor”

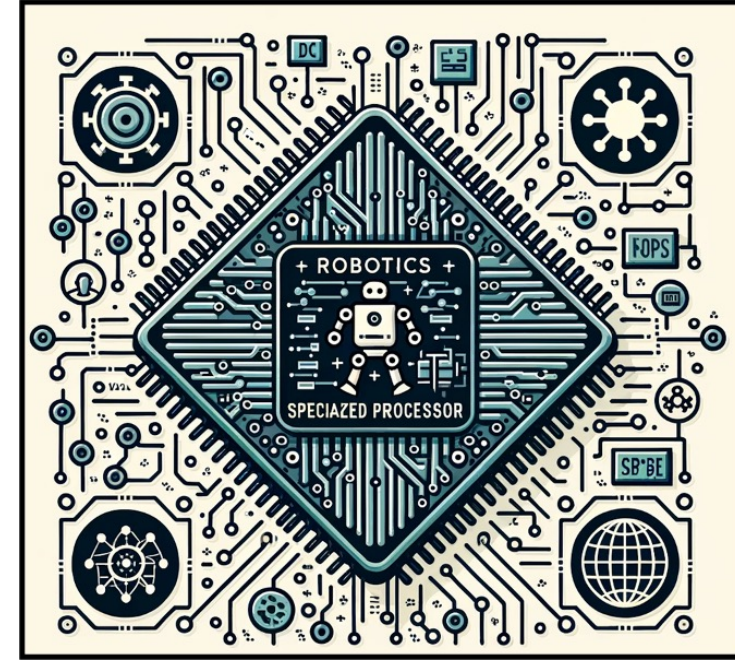
Mohammad Bakhshalipour, Phillip Gibbons 2024



Slides are adapted from ISCA'24 conference talk

Executive Summary

- Architecting a domain-specific processor for robotics
- Extensive profiling of robotics workloads, finding main execution bottlenecks
- Architectural enhancements to address the bottlenecks in robotics workloads
 - Oriented vectorization
 - Approximate acceleration
 - Robot-semantic prefetching
 - Intra-application cache partitioning



Generality-Performance Spectrum

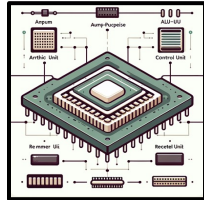


Generality-Performance Spectrum

Intel Core i5

General-Purpose

CPUs



Generality



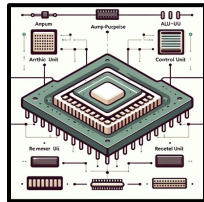
Performance

Generality-Performance Spectrum

Intel Core i5

General-Purpose

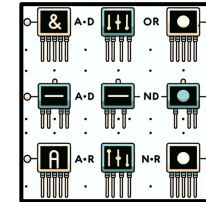
CPUs



RACOD [ISCA'22]

ASIC Hardware

Accelerators



Generality



Performance

Where on Spectrum?

Intel Core i5

Tartan!

RACOD

General-Purpose

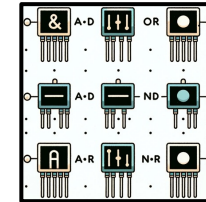
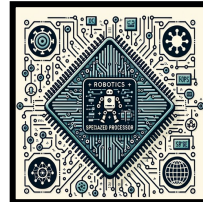
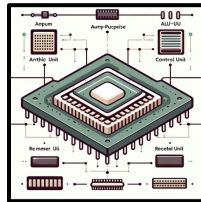
Domain-Specific

ASIC Hardware

CPUs

Processor

Accelerators

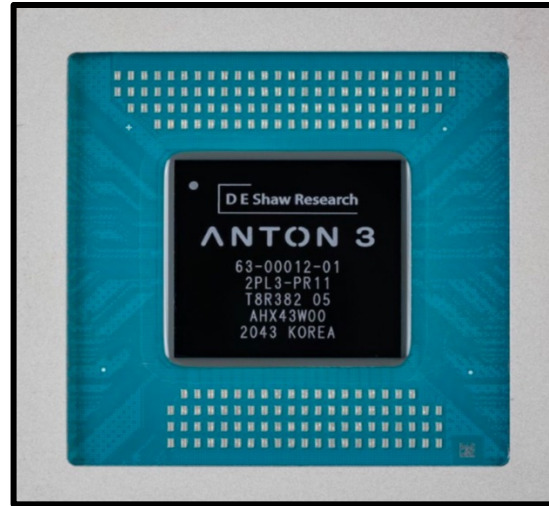


Generality

Performance

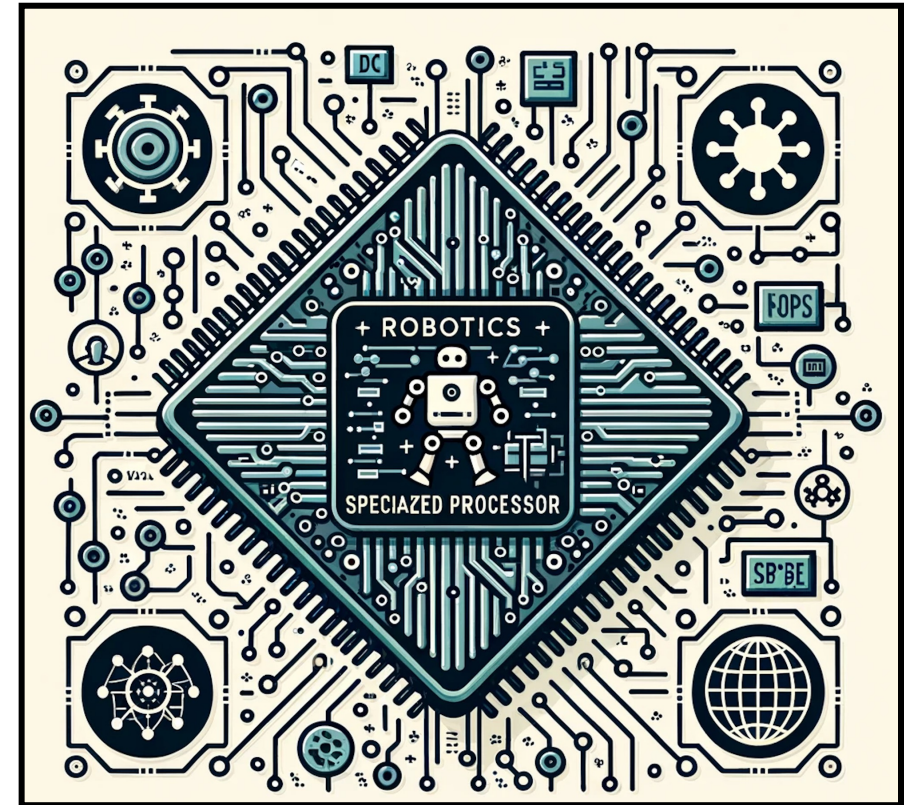
Domain-Specific Processors

- D. E. Shaw Anton
 - Molecular Simulation
- Cisco Silicon One
 - Networking



Tartan: A Robotic Processor

- Microarchitected based on extensive profiling of various robotics workloads
- Includes computational units and memory system extensions that excel robotics workloads

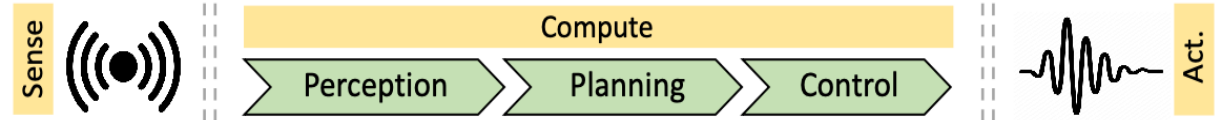


Methodology

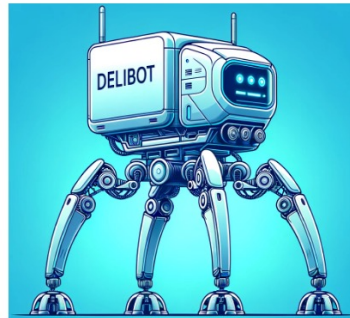
- Simulation framework
 - ZSim [Sanchez+, ISCA'13]
- Baseline processor: Intel Core i7-10610U
 - 4 cores, 8MB L3 cache
 - Integrated in NASA's Valkyrie

Workloads

- RoWild
 - SIGMETRICS'24



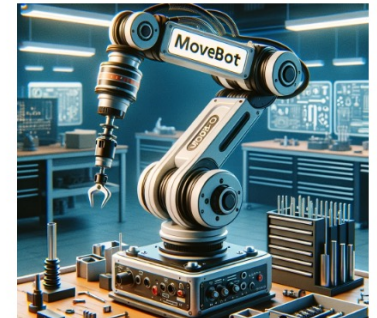
DeliBot



PatrolBot



MoveBot



HomeBot



FlyBot

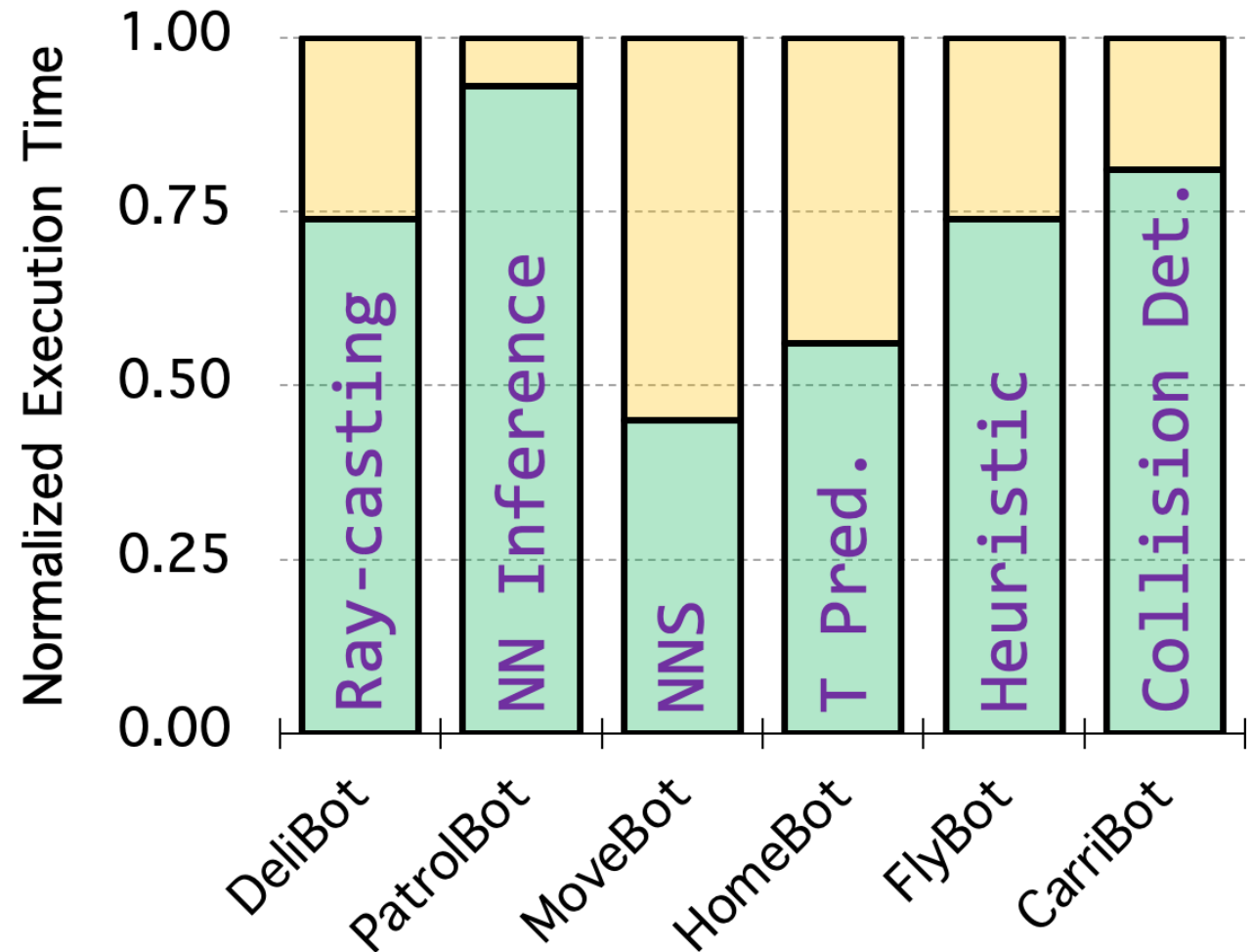


CarriBot



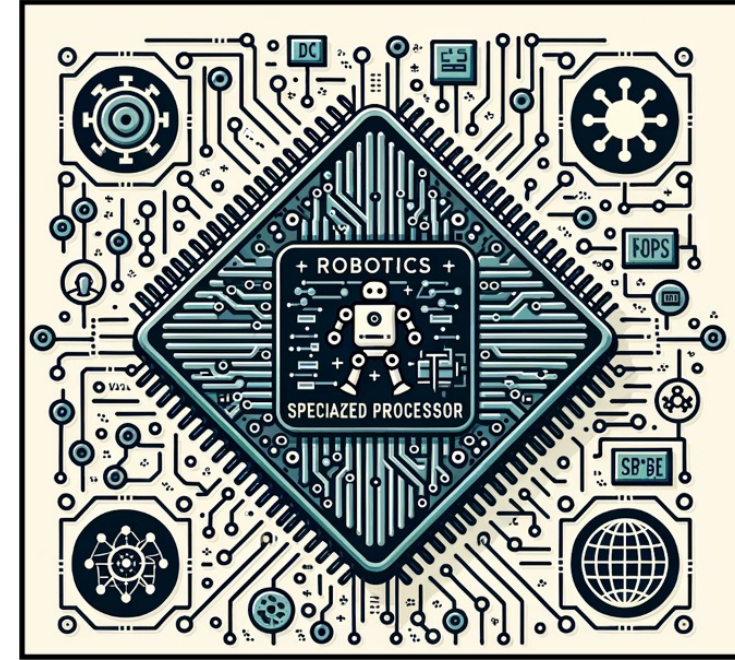
Identifying Performance Bottlenecks

- In pursuit of performance bottlenecks



Executive Summary

- Architecting a domain-specific processor for robotics
- Extensive profiling of robotics workloads, finding main execution bottlenecks
- Architectural enhancements to address the bottlenecks in robotics workloads
 - Oriented vectorization
 - Approximate acceleration
 - Robot-semantic prefetching
 - Intra-application cache partitioning



Discussion: Summary Question #1

What Did the Paper Get Right?

State the 3 most important things the paper says.

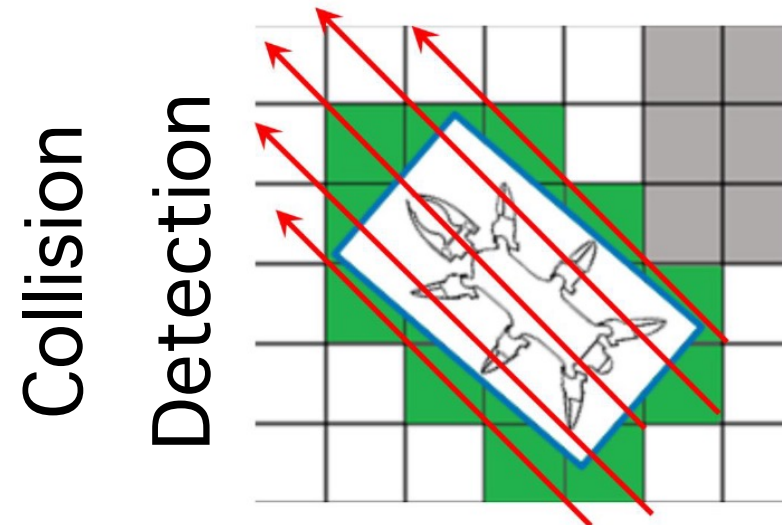
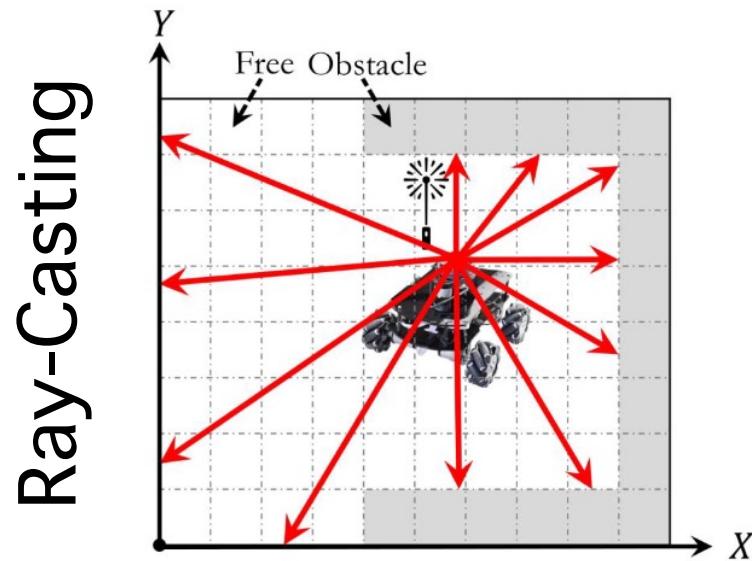
These could be some combination of the motivations, observations, interesting parts of the design, or clever parts of the implementation.

Oriented Vectorization

- Tartan extends CPUs' vectorization unit to handle slanted-line address patterns

Oriented Vectorization

- Tartan extends CPUs' vectorization unit to handle slanted-line address patterns
 - Common in operations like ray-casting and collision detection



Traditional Vectorization: ISA

- Traditional vector instruction:

MOVE %zmm, (%org)

Oriented Vectorization: ISA


- Tartan adds this instruction:

`O_MOVE %zmm, (%org), %orient`

Oriented Vectorization: ISA

- Tartan adds this instruction:

O_MOVE %zmm, (%org), %orient



Base Address

Oriented Vectorization: ISA

- Tartan adds this instruction:

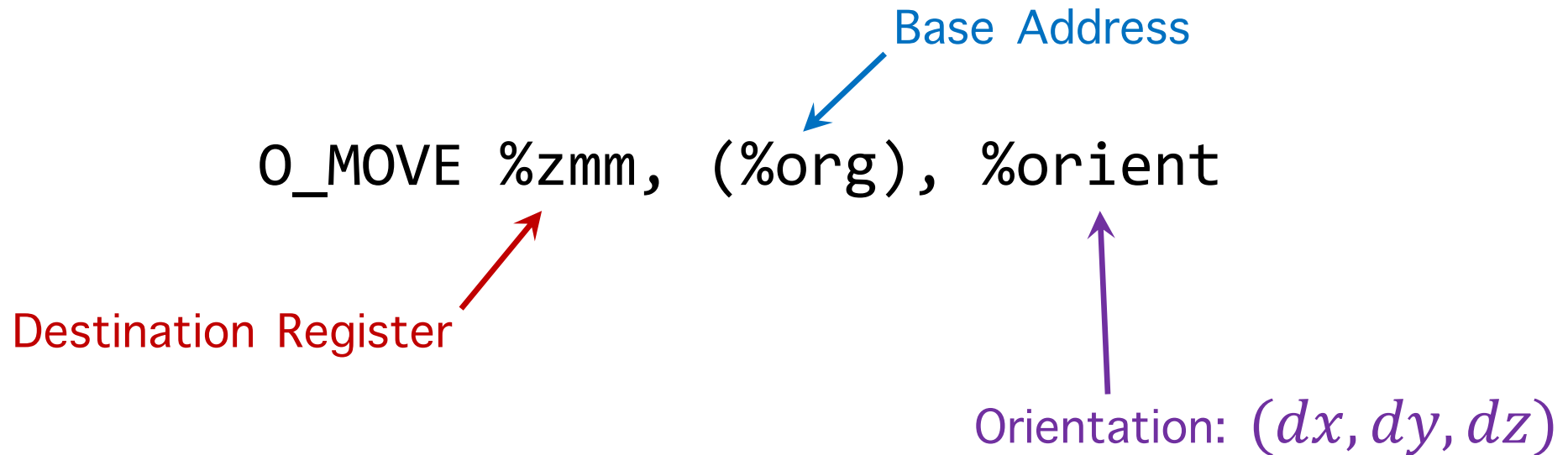
O_MOVE %zmm, (%org), %orient

Base Address

Orientation: (dx, dy, dz)

Oriented Vectorization: ISA

- Tartan adds this instruction:

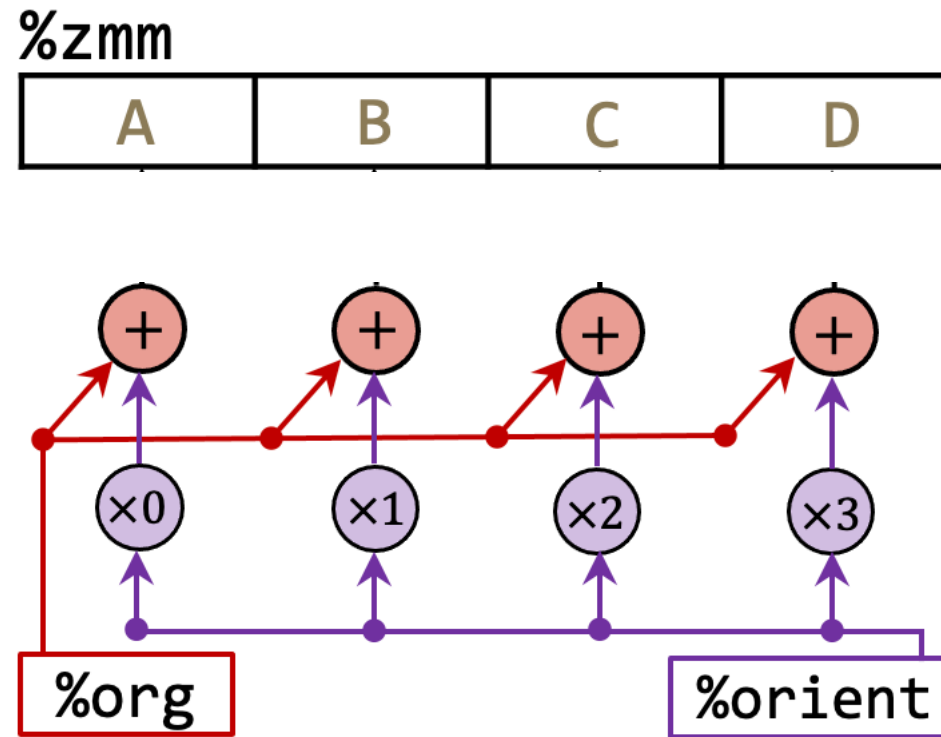


Oriented Vector Address Generation

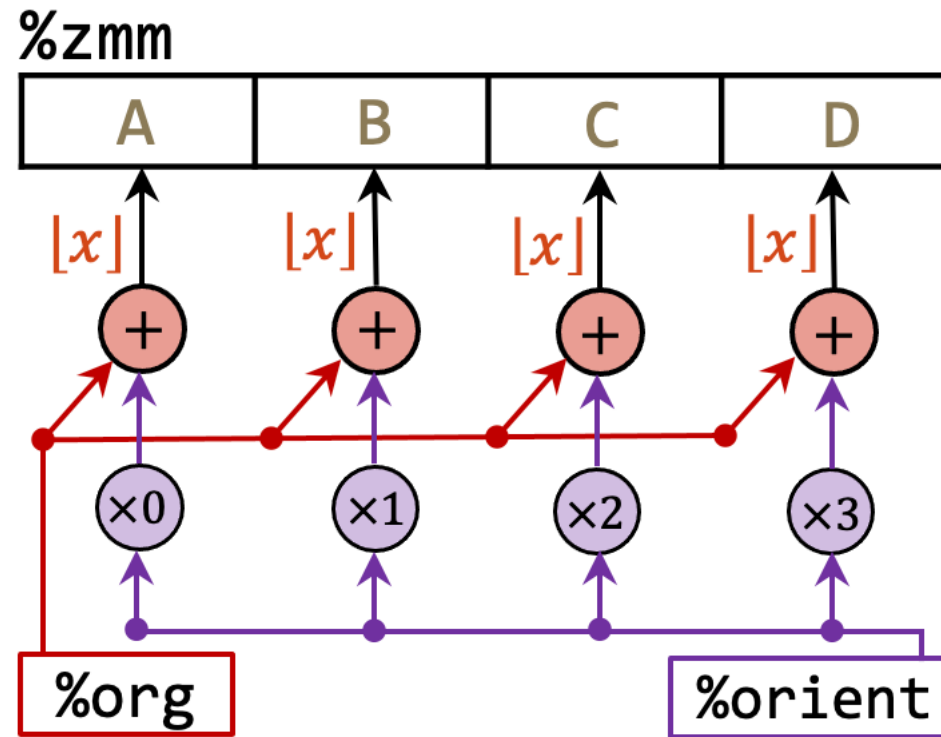
%zmm

A	B	C	D
---	---	---	---

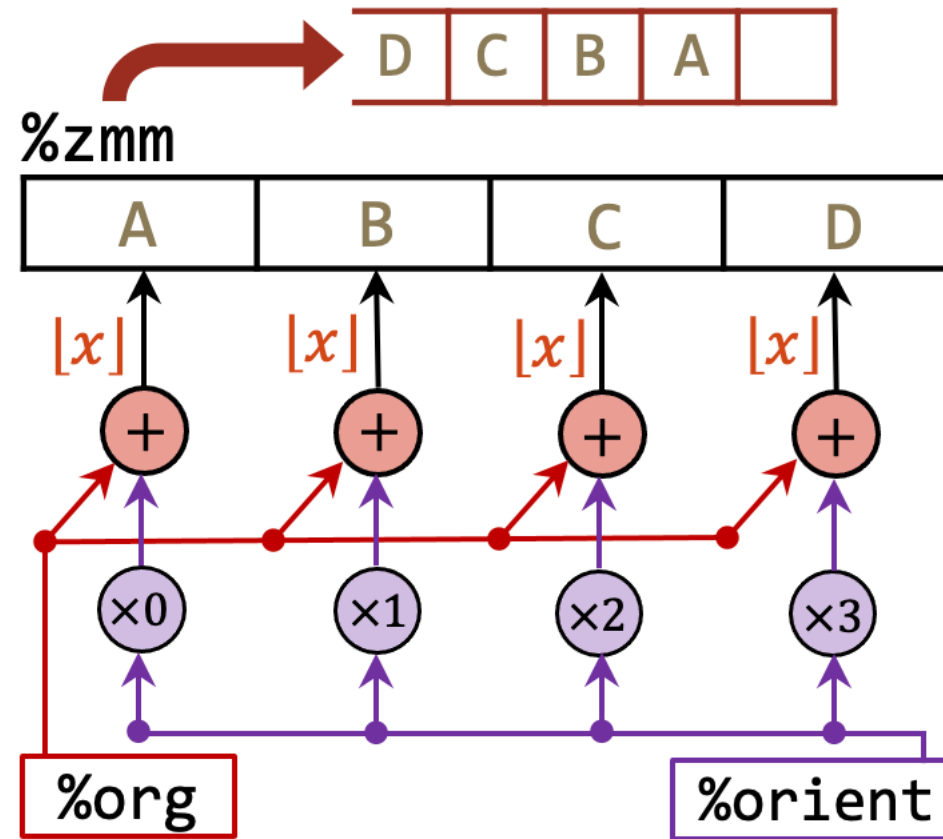
Oriented Vector Address Generation



Oriented Vector Address Generation

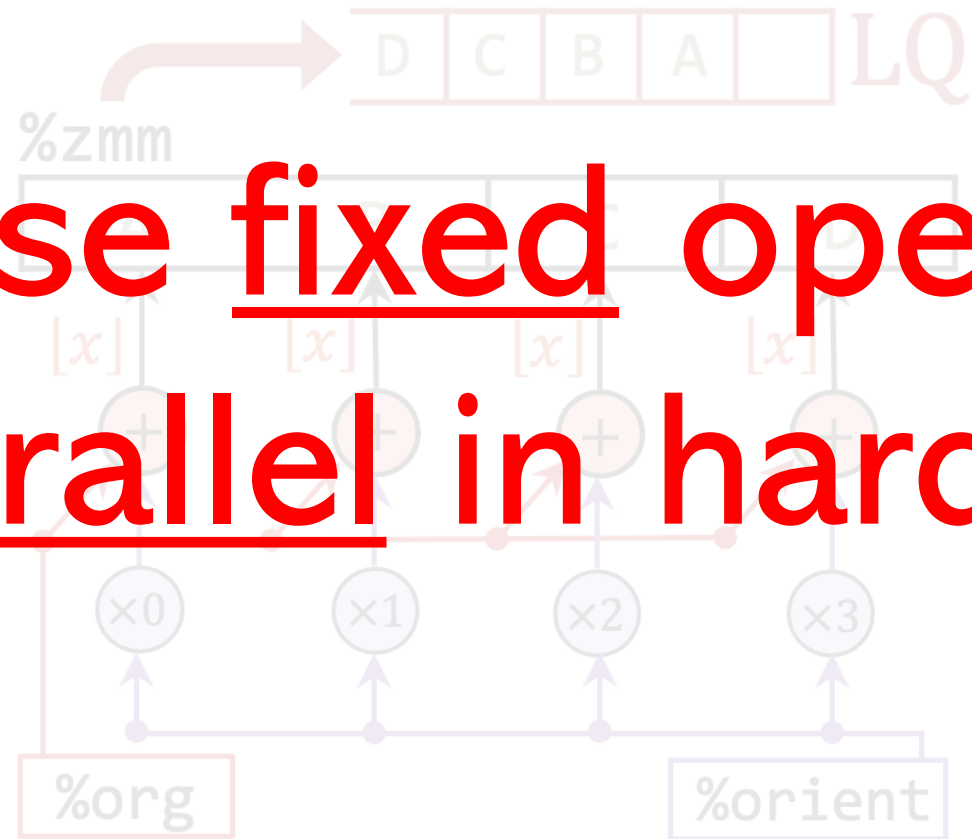


Oriented Vector Address Generation



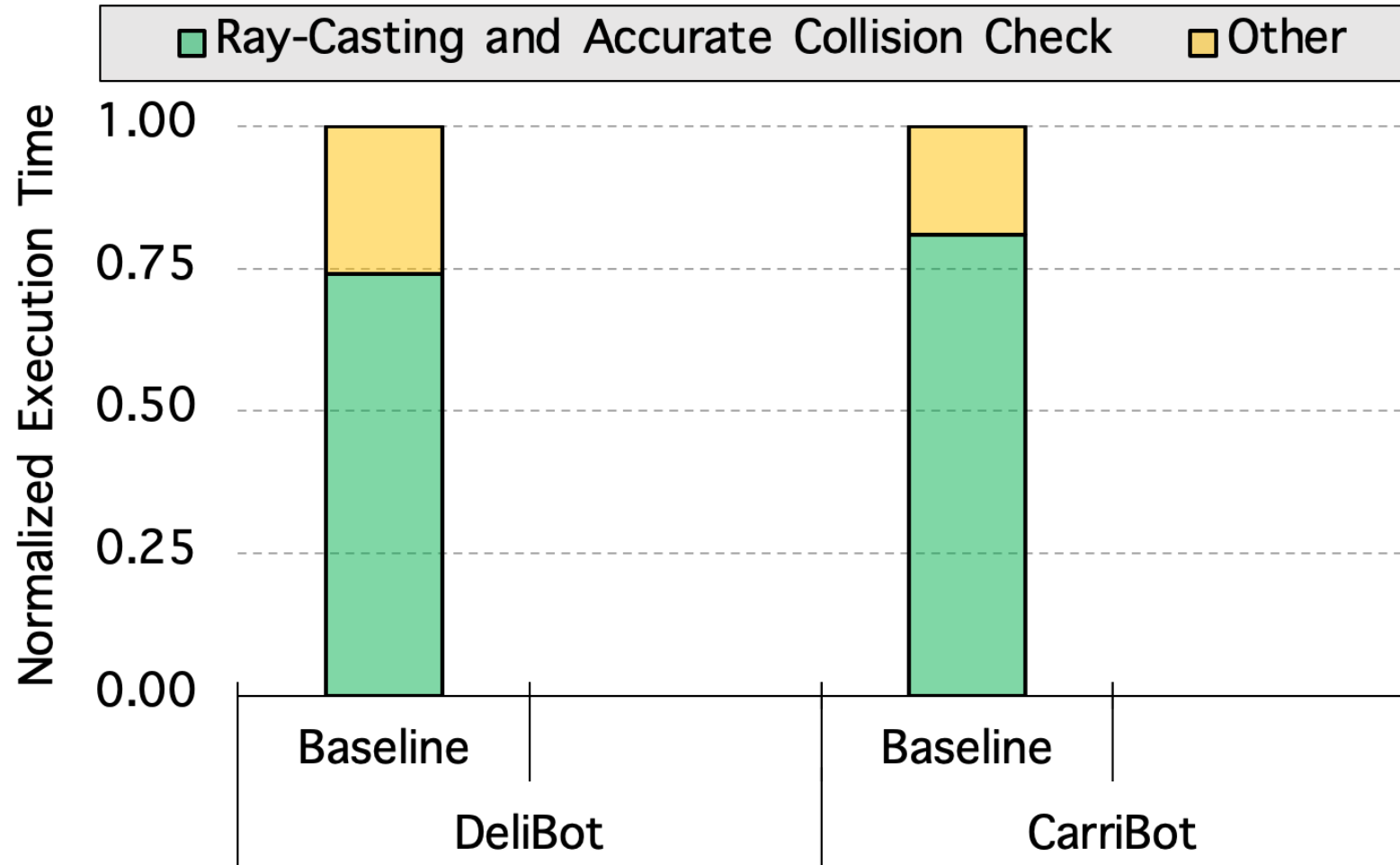
Oriented Vector Address Generation

Do these fixed operations
in parallel in hardware



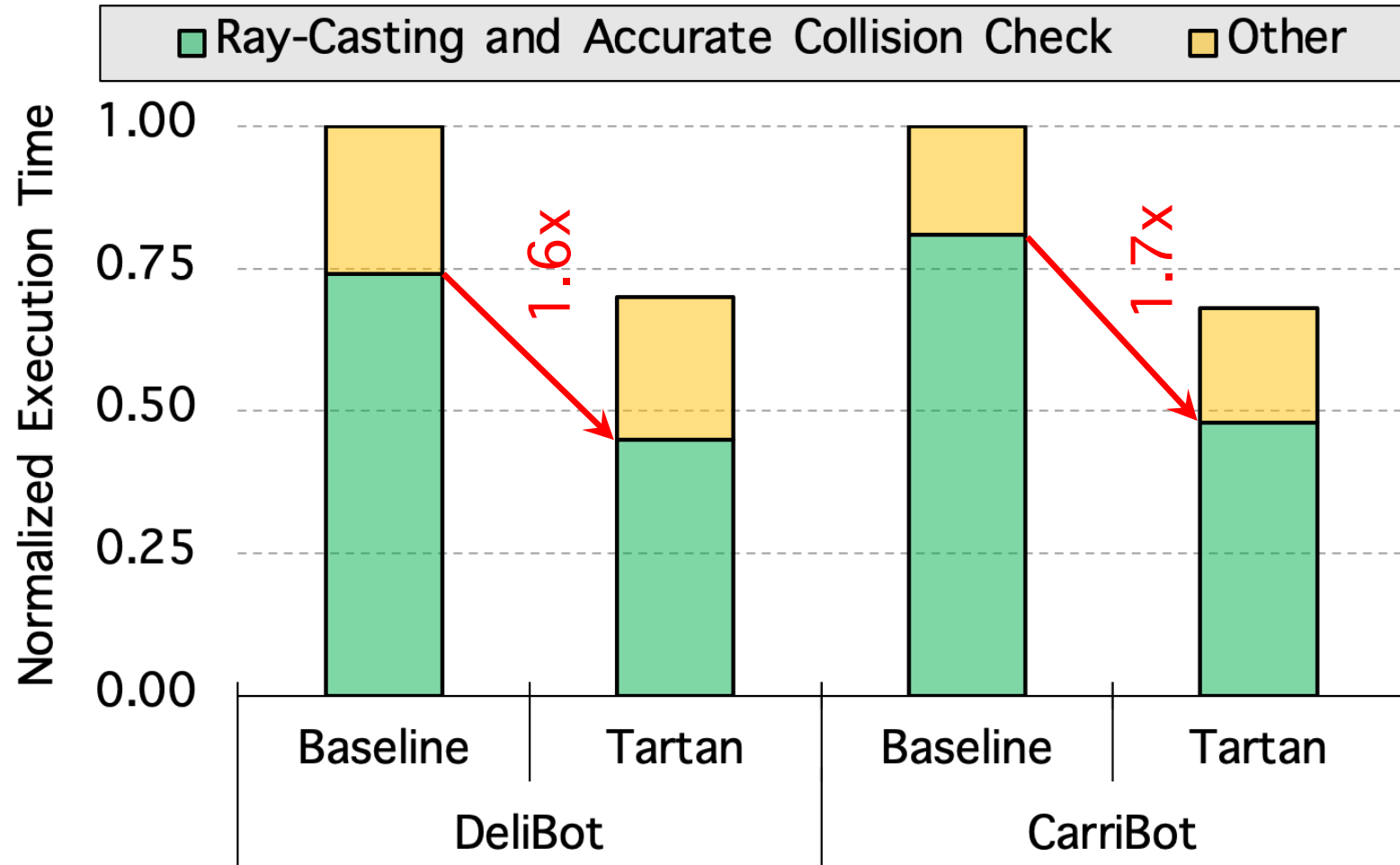
Results: Oriented Vectorization

- DeliBot
 - Ray-Casting
- CarriBot
 - Collision Detection



Results: Oriented Vectorization

- DeliBot
 - Ray-Casting
- CarriBot
 - Collision Detection



Approximate Acceleration

- Many robotic applications allow for approximation
 - Application-level: Cleaning
 - Algorithm-level: Pathfinding

Approximate Acceleration

- Many robotic applications allow for approximation
 - Application-level: Cleaning
 - Algorithm-level: Pathfinding
- Tartan replaces approximable, costly functions by a neural network

Approximate Acceleration

- Many robotic applications allow for approximation
 - Application-level: Cleaning
 - Algorithm-level: Pathfinding
- Tartan replaces approximable, costly functions by a neural network
 - The neural model runs on a hardware accelerator

Approximation Procedure

```
Da"oppinata"~,,eCe) xln ln Cn,,z-che
T (lodnas(- (alsull. )-oOd-,)-)nun)"
| ("oolunaliotoo))-,-) -"z, st-war
| ("anao-"(eis(-nup"-,,)
D) : ("Yill-sbet -"-rN
T (Legunaisan,"-legss@_lhgse))-", lode)
| ("sbee"=(),sexll, l l, (), Lal,))-"-
| ) botian-_- => eieonni(LanaHdmi),)
| (III)
| ("Le" l--)- ( epastill,-pun(( ehs))
P (I6)reszl (esu,"-l-")
D T ((epthzll),lnatiane =,)-"el)),, )))
P l : ("ovelinata],sE--S,canz).N (l- -
| | ((esop"("gsotl,ctj(-aete ", nL ))
| : | niuingaiitionehe(1)))"ca))--Y
Z ( : | l lns,nil,ste) ('+ll ±-(m.-z
T
A "tighan(ngelle,)-+ ±- (nat+-)) ->
D)
T sjadwaltipe"z"-zlll=), l - l(")"))))
| ("unnvan ll, "i >- "notaatian)-'z)'
| ("uothlialasttrindatnation,.com)))))
| (Ka "aol
T mallll(satals="",hasl)tranat"tion"!
± "z" " ± (L,-=")-")
D)
| ((ode,poatonulationl))", -"y N ->e,*)
| : (cavekljng- )-3l,-))"llit,ann) ))
| : (opretiana))", unat),_vdk,om))
| : (ctt--N, lode(teiht,-lno6tlvsion)"z))
```

Approximation Procedure

Sample
Input



```
Da"oppinata"~,,eCe) xln ln Cn,,z-che
T (lodnas(-((alsull.)-oOd-,)-)nun)"
I ("oolunaliotoo))-,-"st-war
I ("anaa-"eis(-nup"-,"
D) : ("Yill-sbet-"rN
T (legunaisan,"-legss_lhgse))-",lode)
I ("sbee"=(),sexll, l l, (),lal,))-"-
I ) botian-_-> eieonni(LanaHdmi);)
I (III)
I ("Le" l--)- ( epastill,-pun(( ehs))
I (I6)reszl (esu,"-l-"
I ((epthzll),lnatiane =,)-"el)),, )))
P l : ("ovelinata,se--S,canz)." (l- -
I ((esop"("gsotl,ctj(-aete ", nL ))
I : | niuingalitionshe(1)))"cd))-Y
Z ( : | lns,nil,ste) ('+ll±-(m.-s
I
A "tighan(ngelle,-)-+±-(nat-"))->
D)
I sjadvaltipie"z"-zlll=, l - l(")"))))
I ("unnvan ll, "i>-notaatian)-'s)'
I ("uothllalasttrindatnation,com)))))
I (Ka "aol
I mallll(satals=","hasl)tranat"tion"!
I "z" " ± (L,-="")
D)
I ((ode,pootonulationl))"-,-N->e,*)
I : (cavekljng-)-3l,-))"llit,ann) ))
I : (opretiana))",unat),_vdk,om))
I : (ctt-X,lode(teiht,-lno6tlvsion)"z))
```



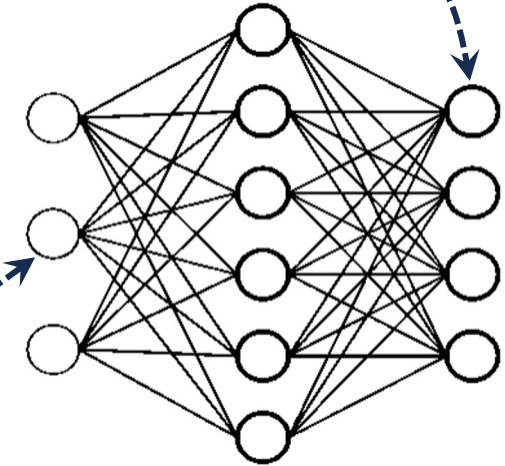
Output

Approximation Procedure

Sample
Input

```
Da"oppinata"~,,eCe) xln ln Cn,,z-che
T (lodnas(-((alsull.)-oOd,-)-)nun)"
I ("oolunaliotoo))-,-"z, st-war
I ("anao-"eis(-nup"-," )
D) : ("Yilll-sbet-"-rN
T (legunaisan,"-legss_lhgse))-,"lode)
I ("sbee"=(),sexll, l l, (),lal,))-,"
I ) botian-_-> eieonni(LanaHdmI);)
I (III)
I ("Le" l--)- ( epastill,-pun{( ehs))
I (I6)reszl (esu,"-l-"
I ((epthzll),lnatiane =,)-"el)),, )))
P I : ("ovelinatal,sE--S,canz).N (l-
I : ((esop("gsetl,ctj(-aete ",nL ))
I : | niuilingalitionshe(1))"cd))-Y
Z ( : | lns,nil,ste) ('+ll=-m.-s
I
A"tighan(ngelle,-)-+;- (nat-")->
D)
I sjadvaltipis"-zlll=),l - l(")"))))
I ("unnvan ll, " > "notaation)-"z)'
I ("uothllalasttrindatnation,com)))
I (Ka "aol
I mallll(satals=","hasl)tranat"tion!
I "z" " ± (L,-="")
D)
I ((ode,pootonulationl))"-,-"z, N->e,*)
I : (cavekljng-)-3l,-))"llit,ann) )
I : (oprehiara))",unat),,vdiw,om) )
I : (ctt-*,lode(teiht,-lno6tlvsion)"z)
```

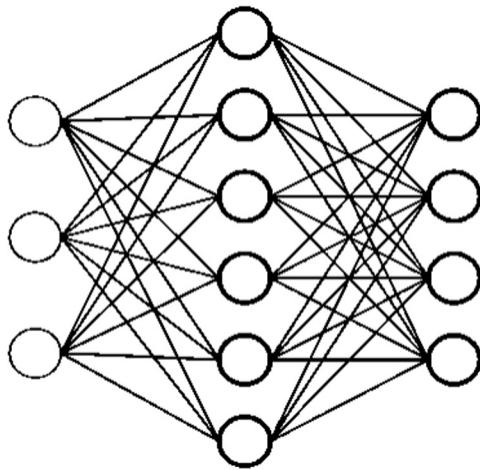
Output



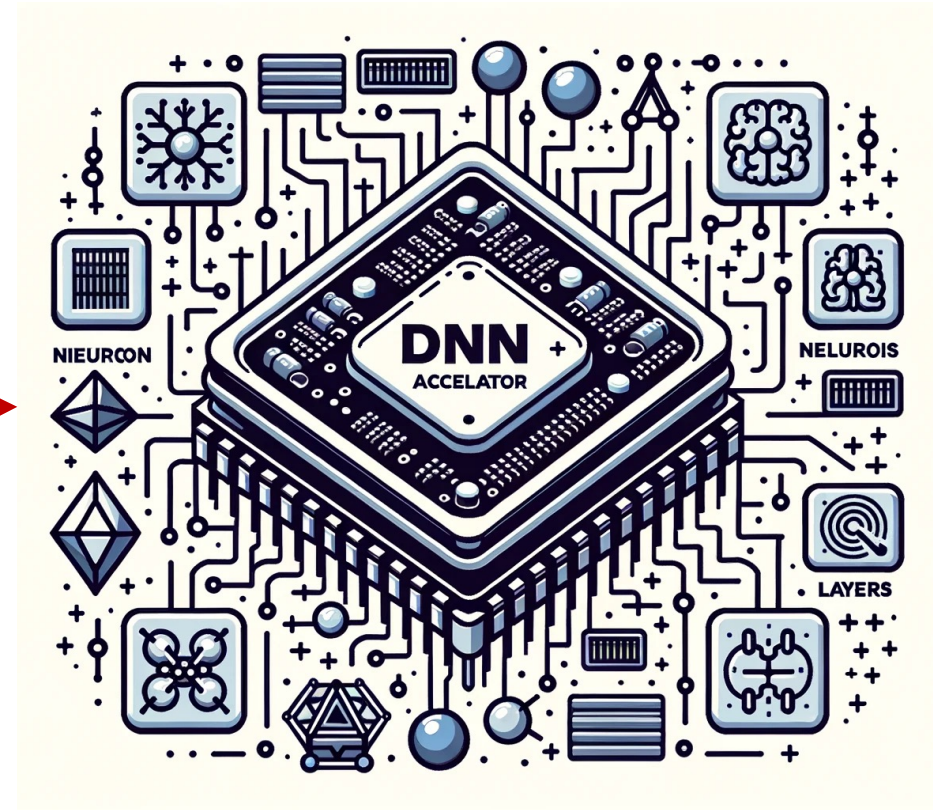
Approximation Procedure



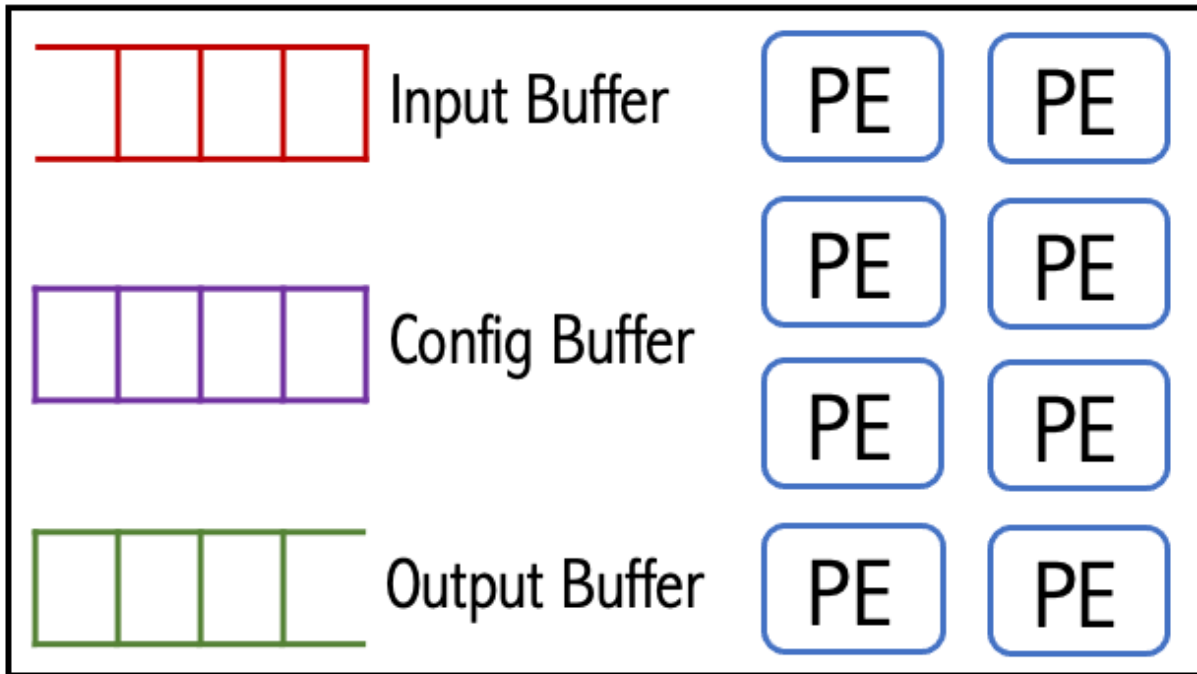
Runtime Operations



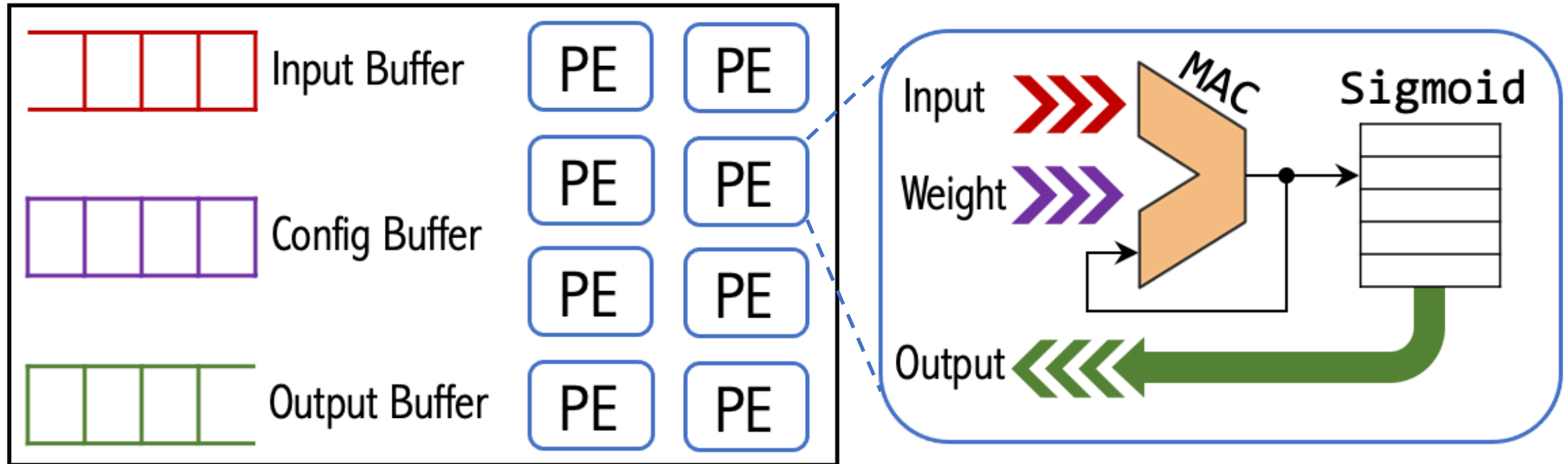
Run



Neural Processing Unit (NPU)

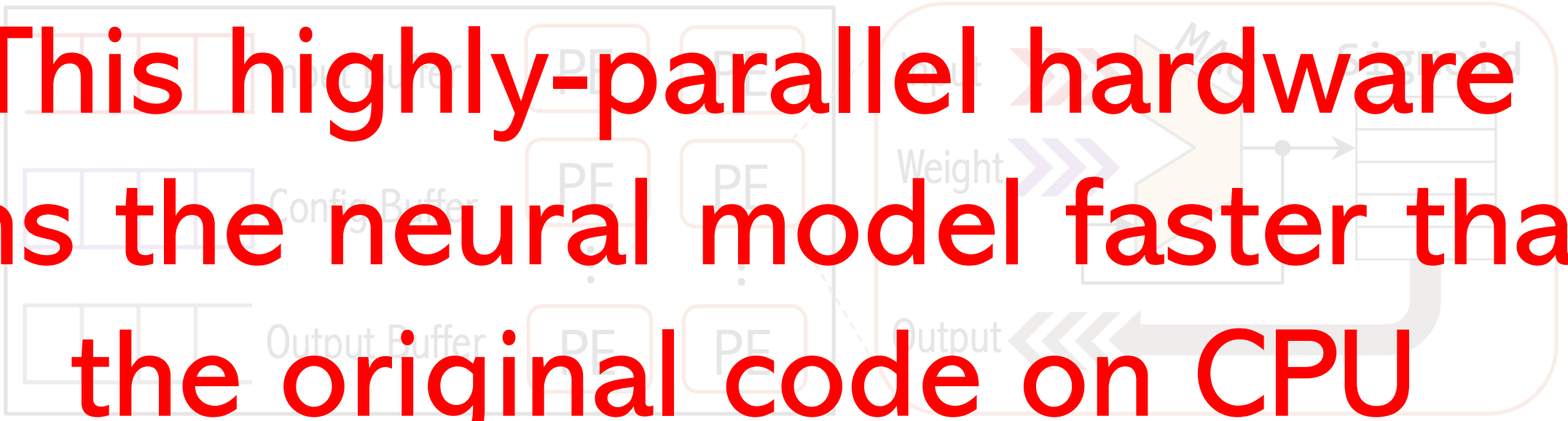


Neural Processing Unit (NPU)



Neural Processing Unit (NPU)

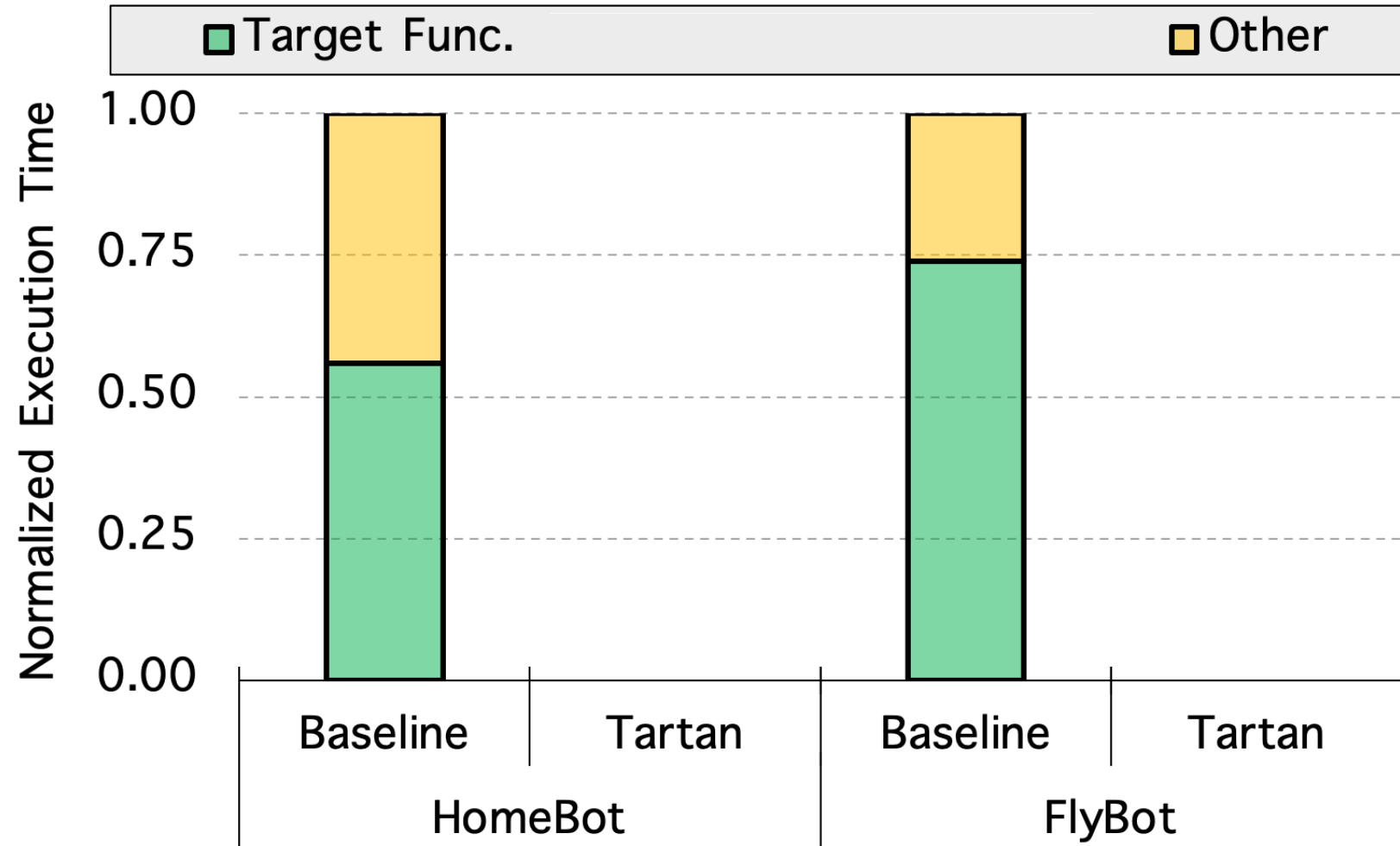
This highly-parallel hardware runs the neural model faster than the original code on CPU

A faint background diagram of an NPU architecture. It shows a grid of Processing Elements (PE) on the left, connected to a central processing block. Labels include 'Config Buffer', 'Output Buffer', 'Weight', and 'Output'. Arrows indicate data flow from the PE grid through the central block to the output.

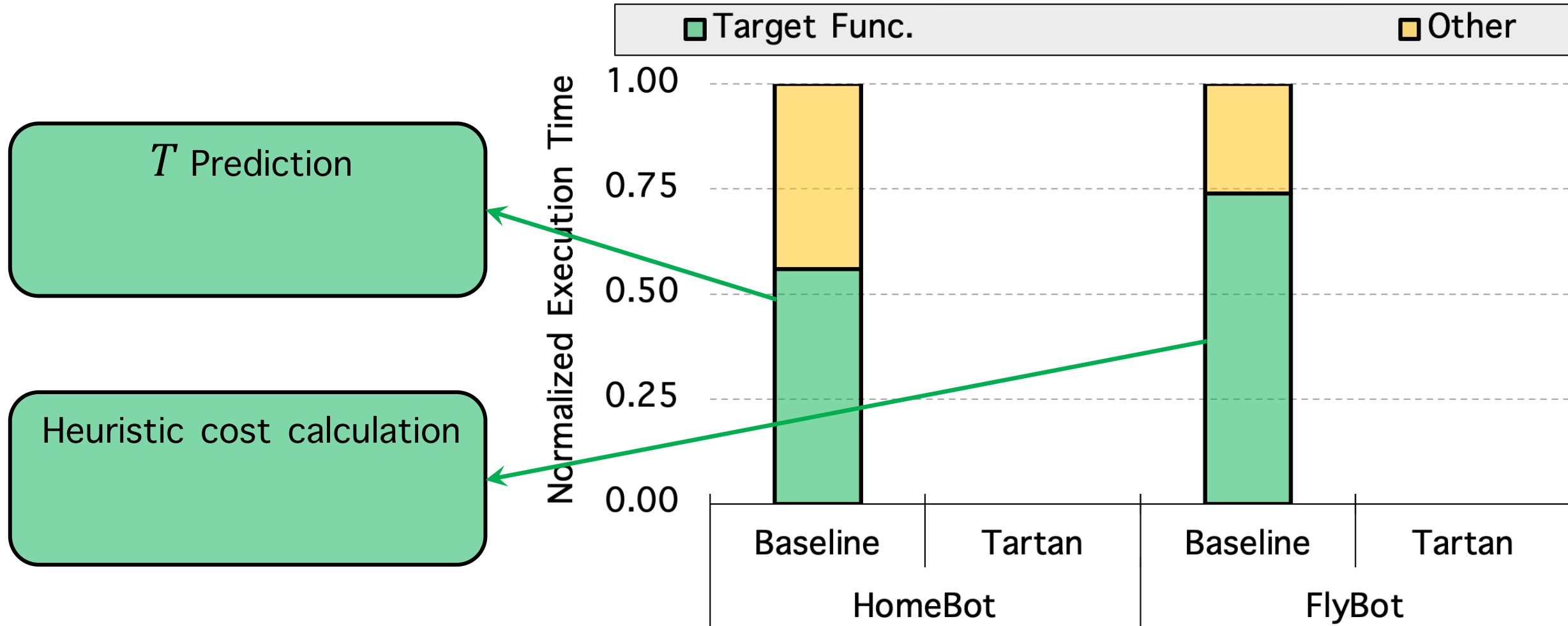
Applications

- Compute intensive functions that can be replaced with “efficient” neural networks
- Efficient refers to
 - Computation
 - Accuracy

Results: Approximate Acceleration



Results: Approximate Acceleration



Results: Approximate Acceleration

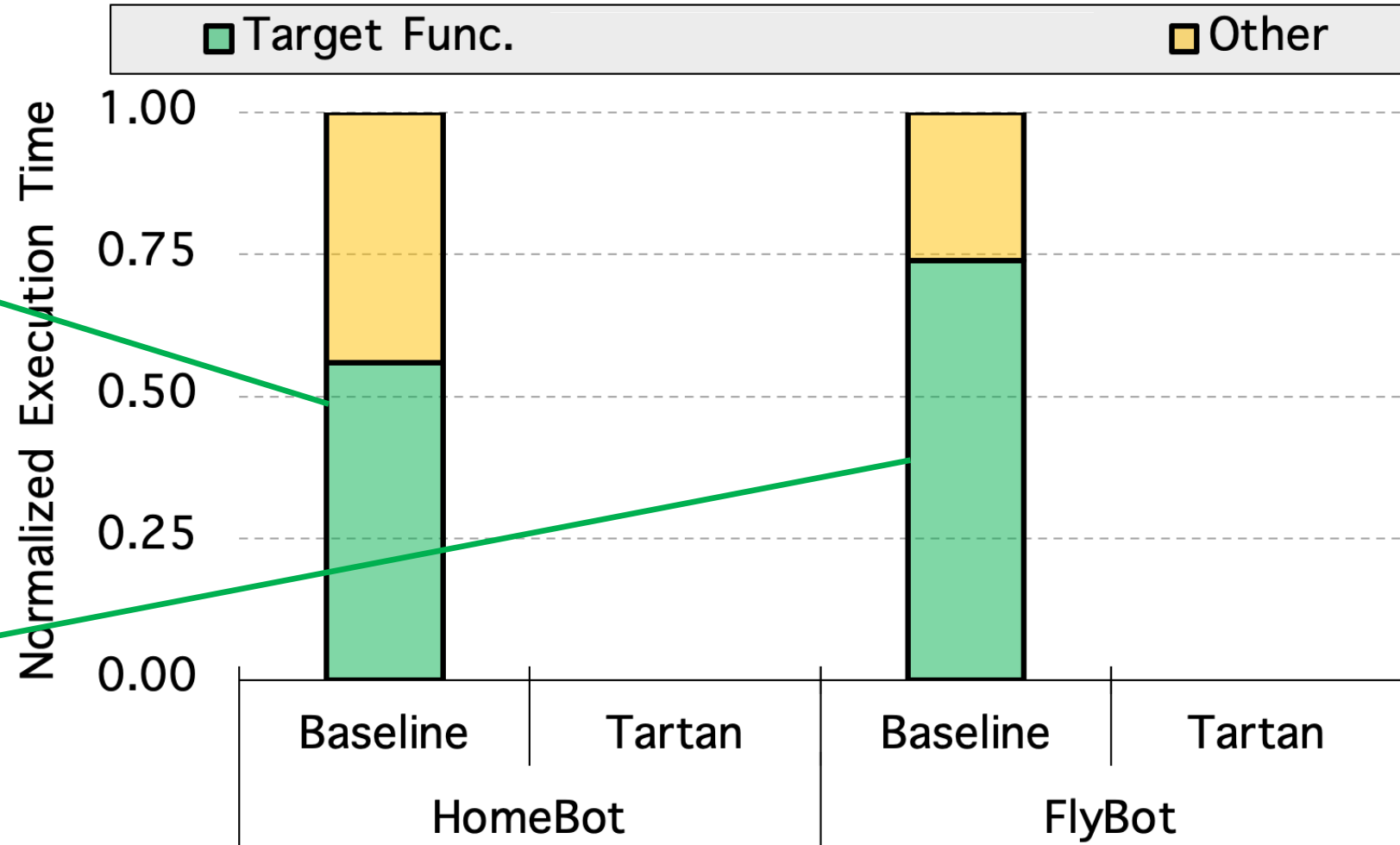
NPU Layers

~~$T_{\text{Prediction}}$~~

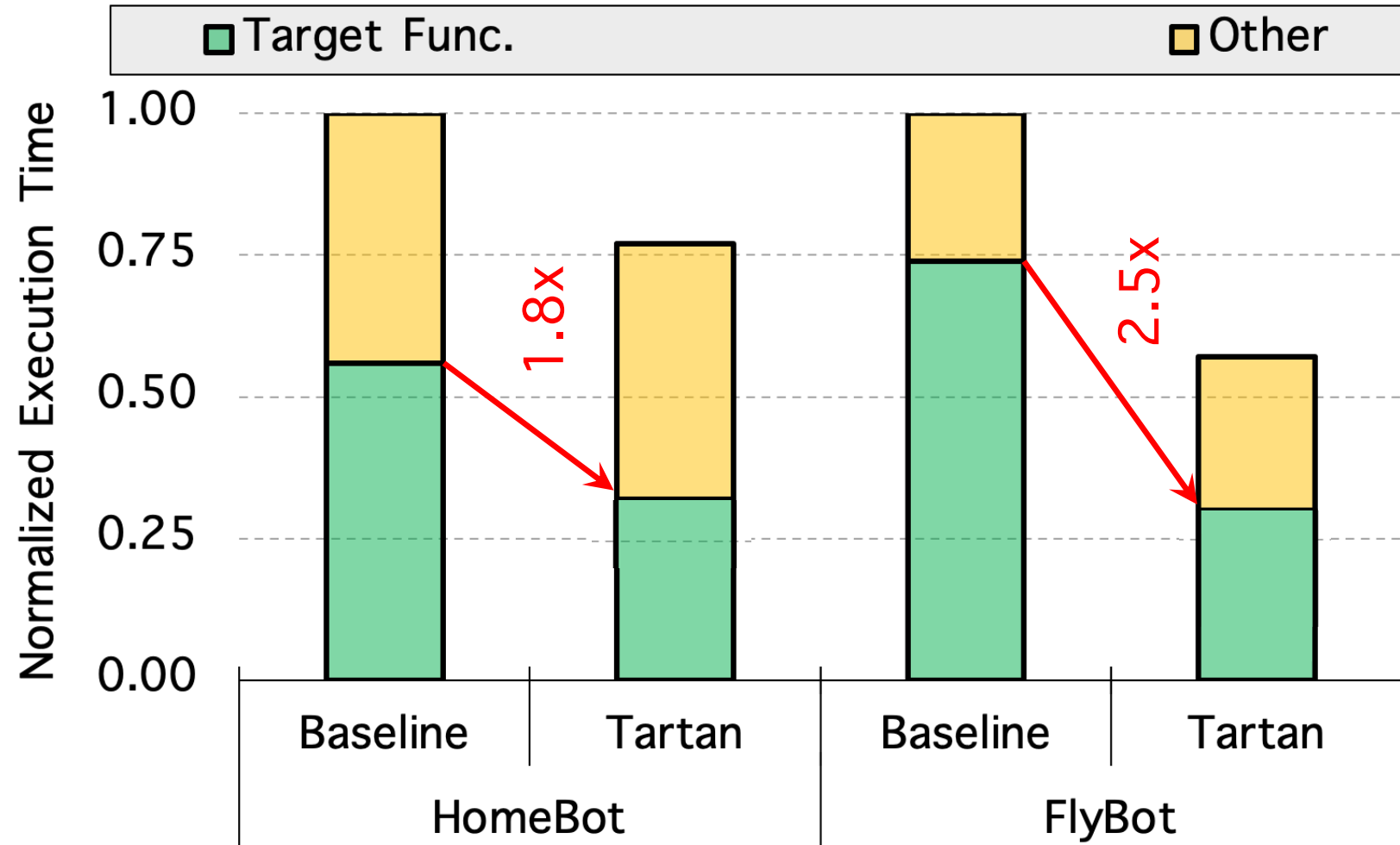
192 → 32 → 32 → 6

~~Heuristic cost calculation~~

6 → 16 → 16 → 1

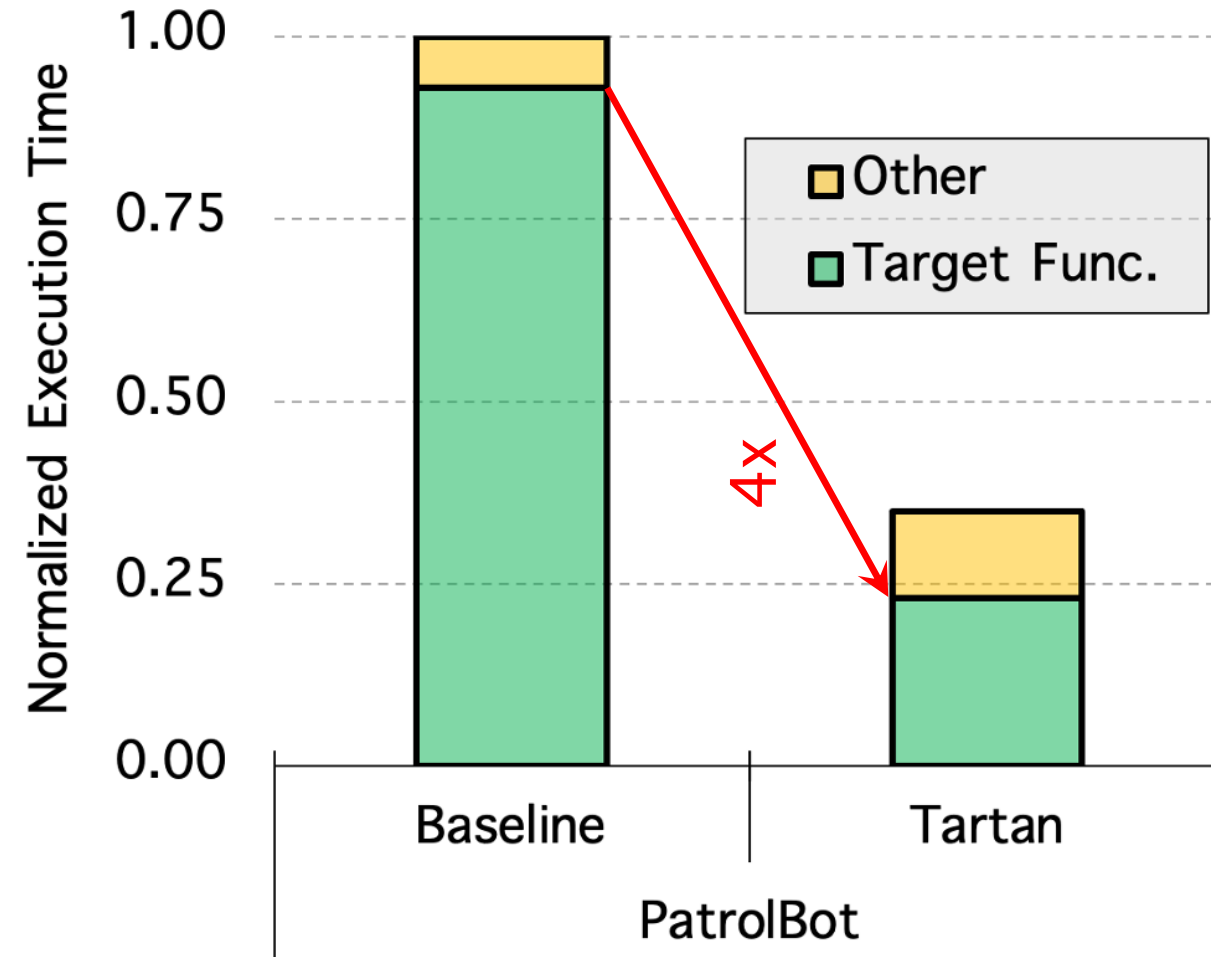


Results: Approximate Acceleration



NPU Is Multimodal

- NPU can accelerate native neural networks



Also in The Paper

- Vectorized nearest-neighbor search
- Semantic-aware hardware prefetching
- Intra-application cache partitioning
- Engineering optimization
 - Cacheline sizing
 - Producer-consumer communications

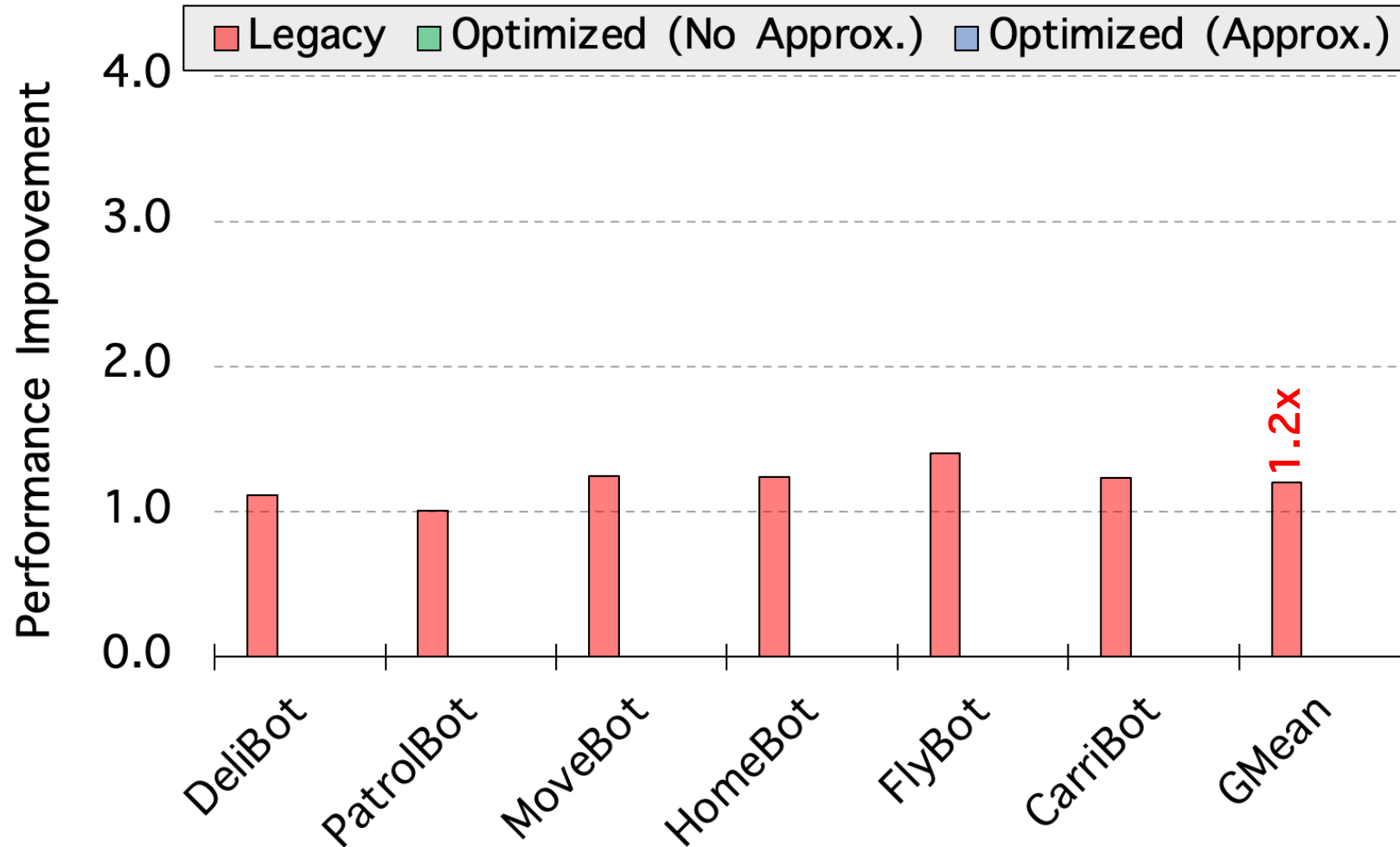
Discussion: Summary Question #2

What Did the Paper Get Wrong?

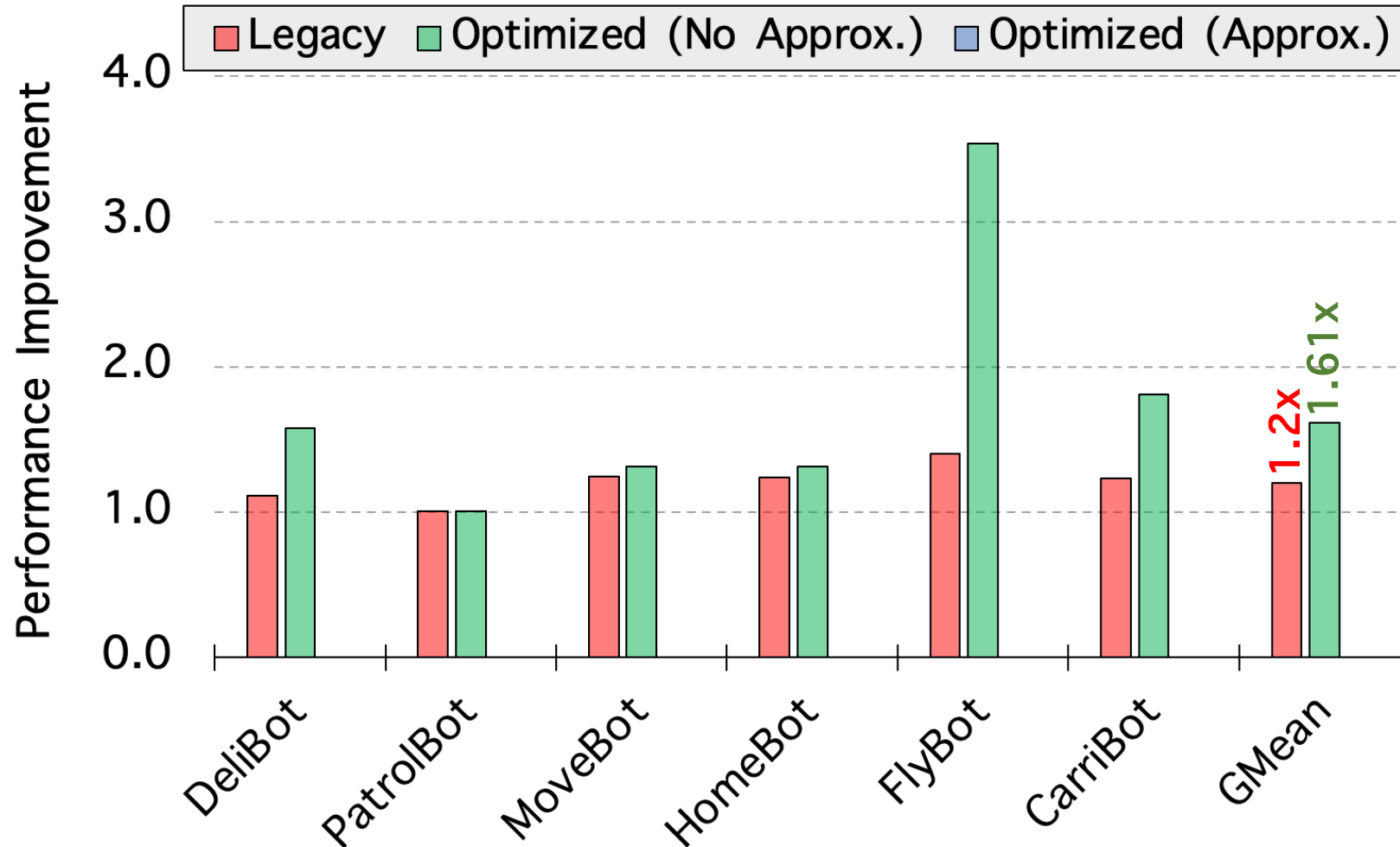
Describe the paper's single most glaring deficiency.

Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.

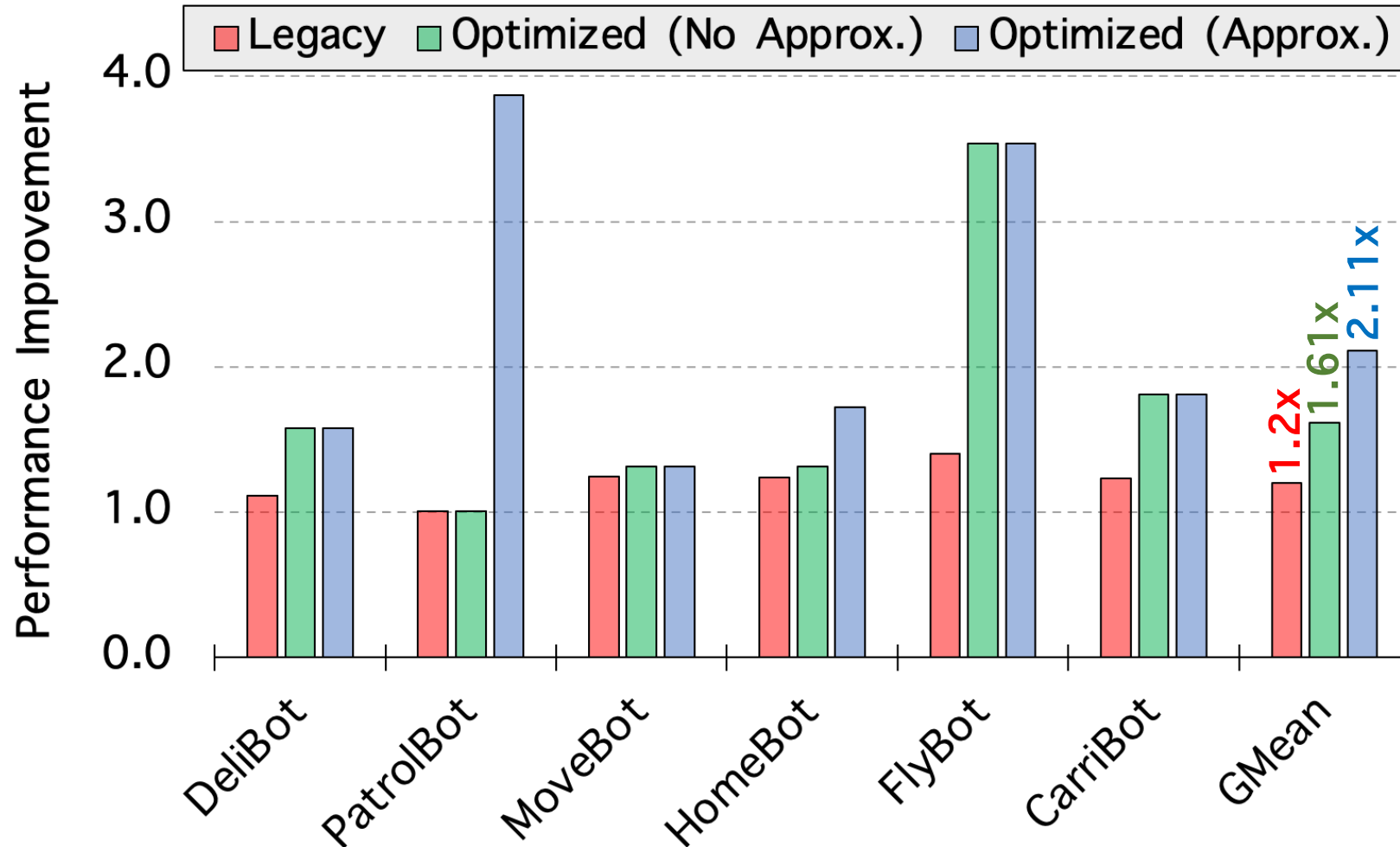
Performance Altogether



Performance Altogether

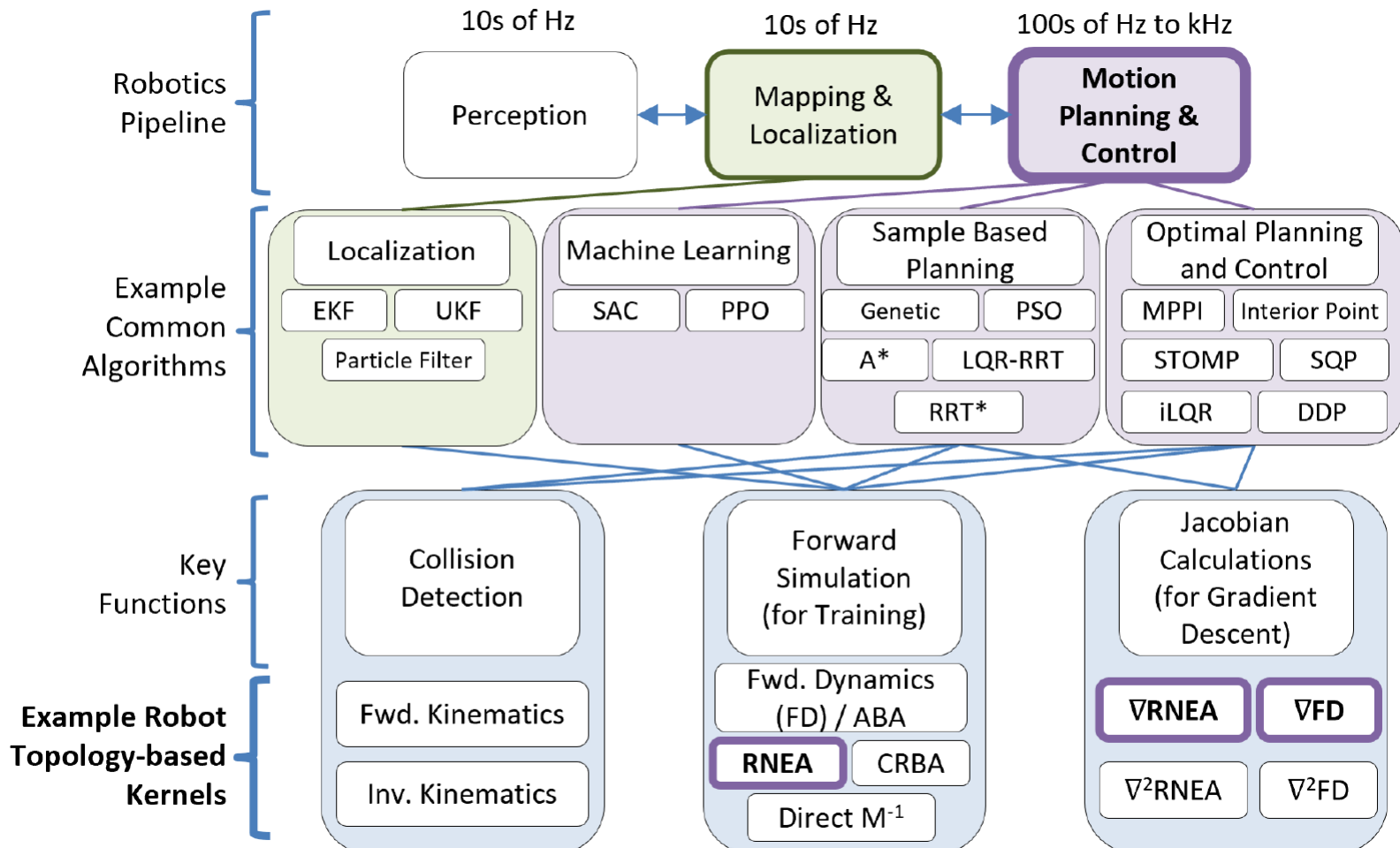


Performance Altogether

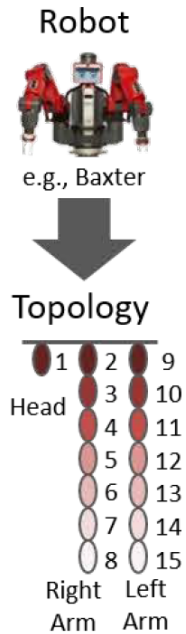


“RoboShape: Using Topology Patterns to Scalably and Flexibly Deploy Accelerators Across Robots”

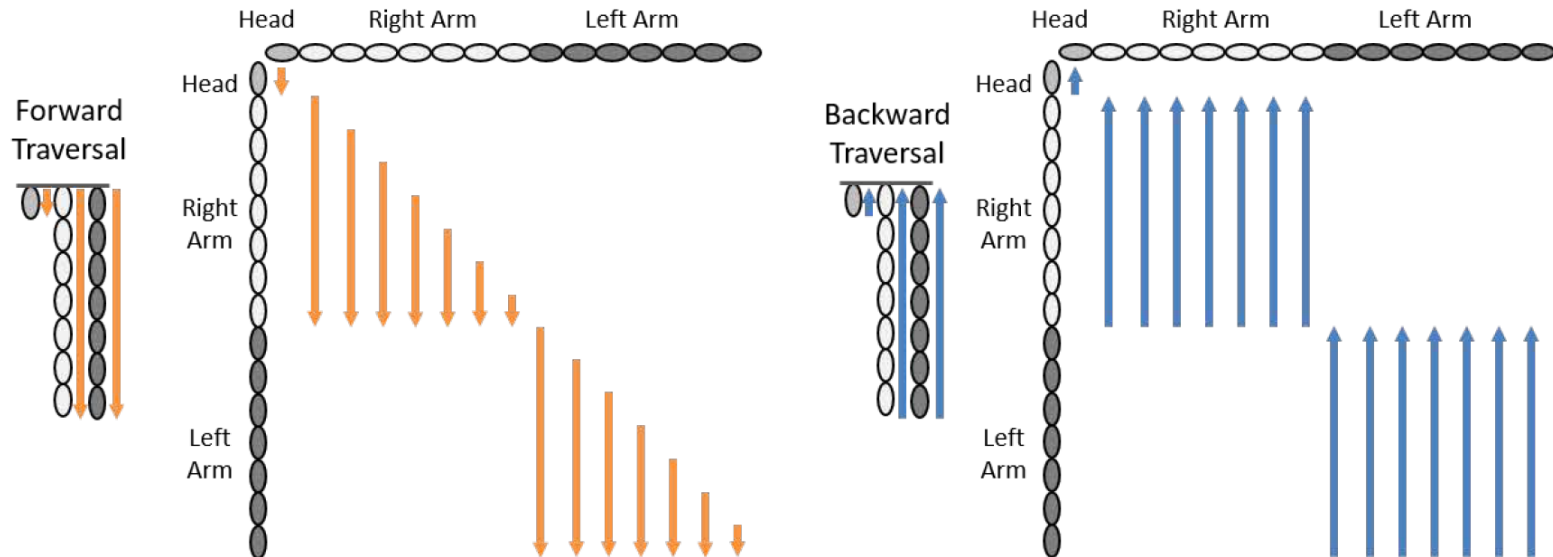
Sabrina Neuman, Radhika Ghosal, Thomas Bourgeat,
Brian Plancher, Vijay Janapa Reddi 2023



Topology Traversal

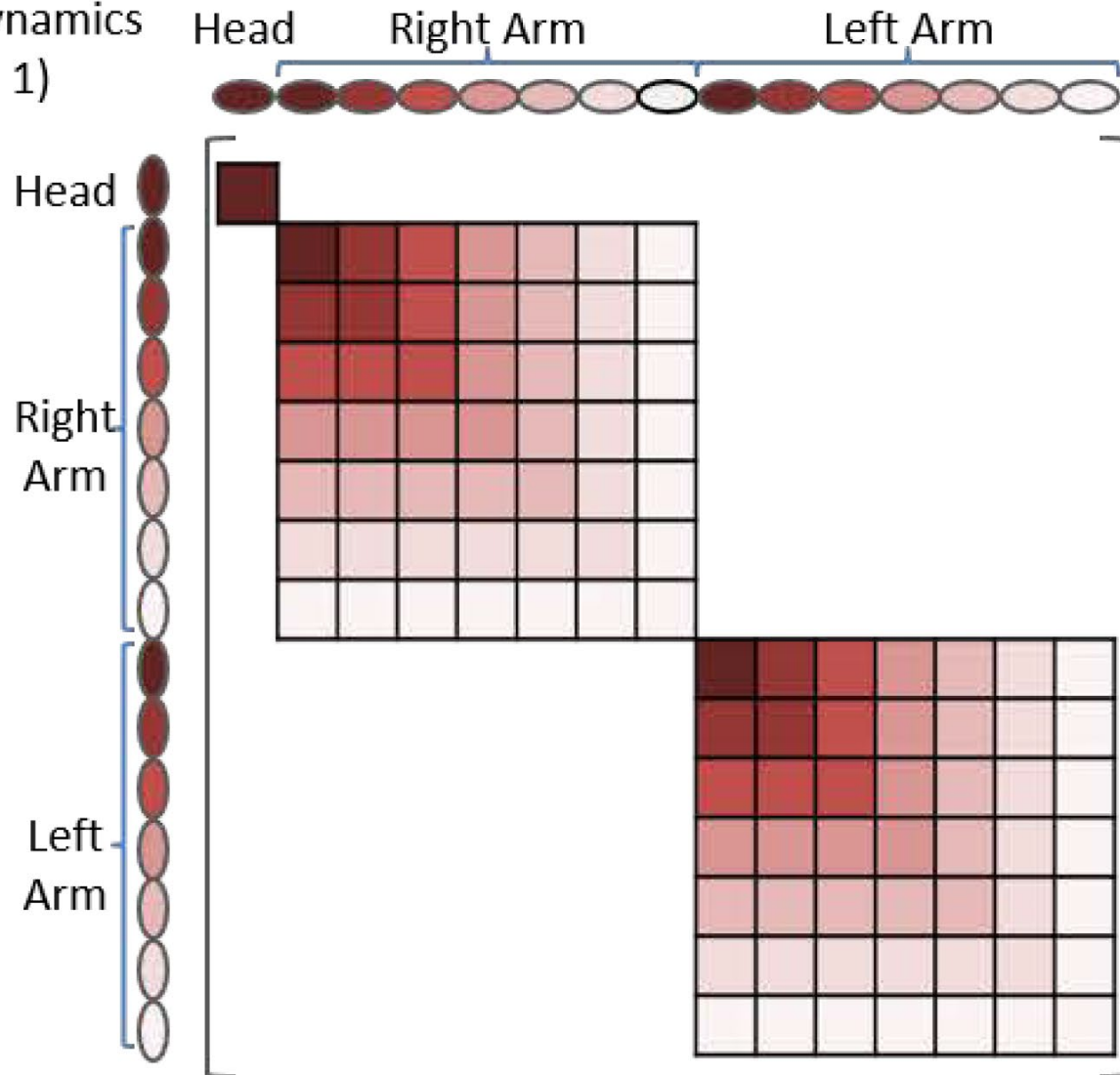


① Topology Traversal Computation Pattern e.g., ∇ RNEA in Dynamics Gradients (Alg. 1)



Topology-based Matrix Computation Pattern

e.g., M^{-1} in Dynamics
Gradients (Alg. 1)



RoboShape Framework

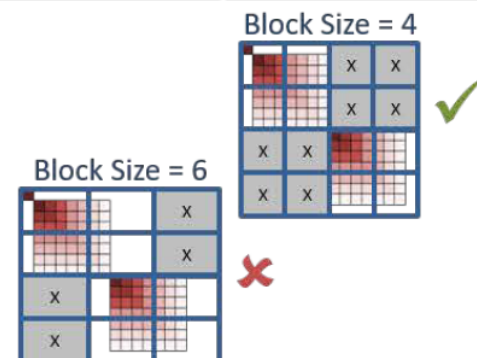
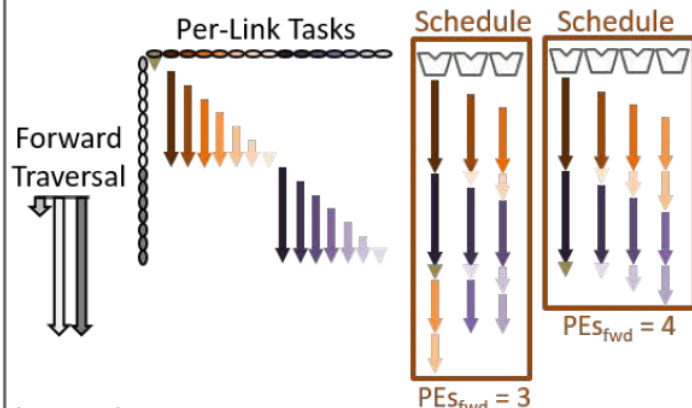
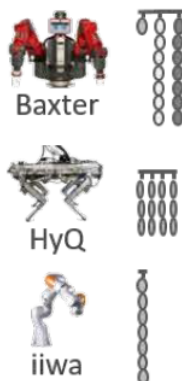
High-Level
Deployment
Information

Robot Topology
Parsing

① Topology Traversal Computation Pattern
→ Task Mapping, Scheduling, & Allocation

② Topology-Based Matrix Computation Pattern
→ Sparse Block Matrix Operations

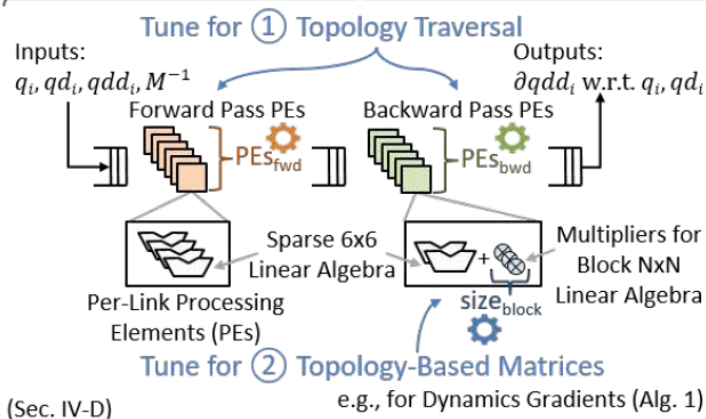
Robot
Computing
Resources
or



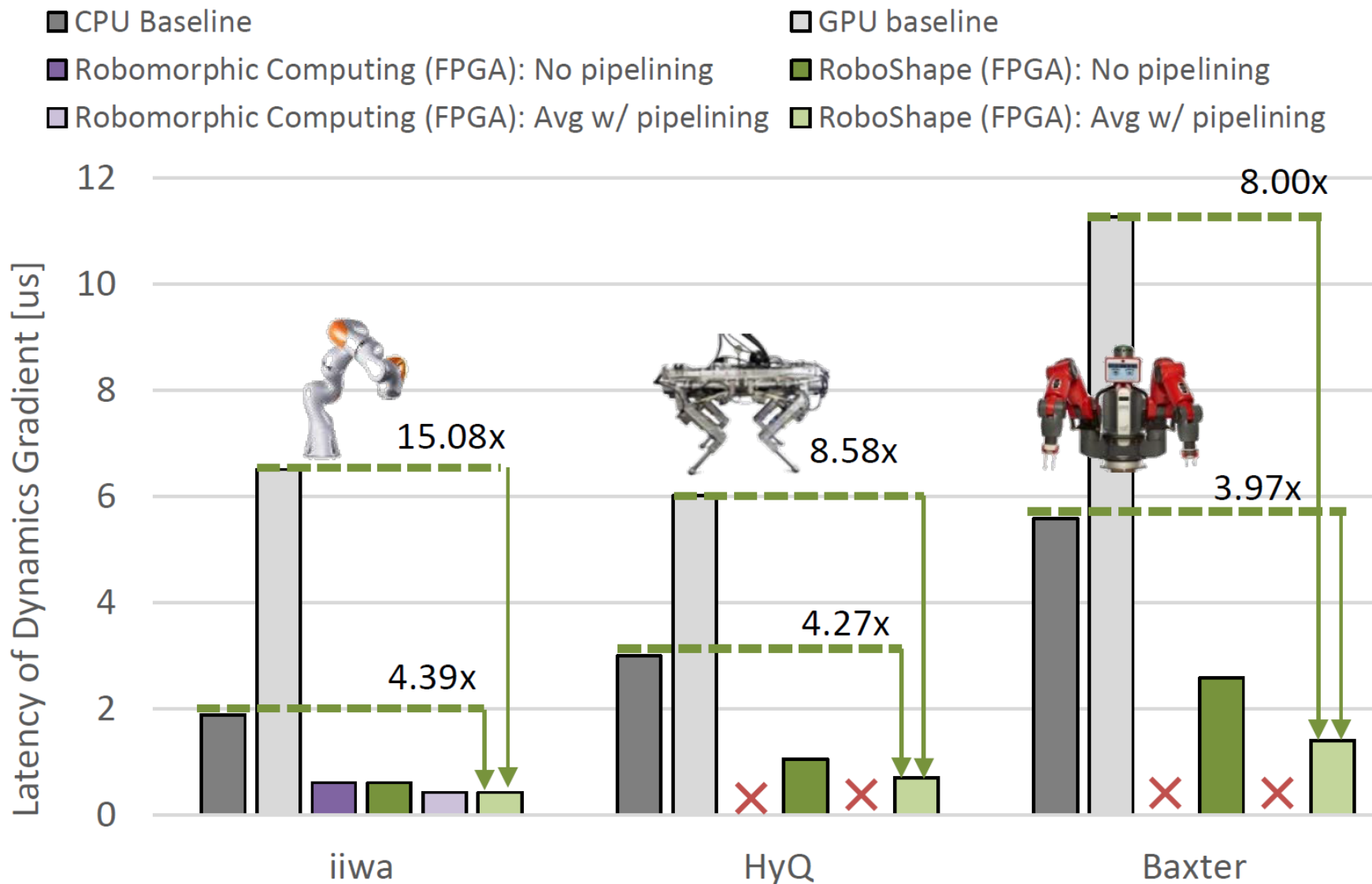
Lowering to
Hardware

Resource
Utilization
Tuning
(Sec. V-C, V-D, V-E)

Accelerators



Latency of Dynamic Gradients



To Read for Wednesday

“In-Datcenter Performance Analysis of a Tensor Processing Unit”

**Norman Jouppi, Cliff Young, Nishant Patil, David Patterson, et al.
2017**

Optional Further Reading:

“Graphite: Optimizing Graph Neural Networks on CPUs Through Cooperative Software-Hardware Techniques”

**Zhangxiaowen Gong, Houxiang Ji, Yao Yao, Christopher Fletcher,
Christopher Hughes, Josep Torrellas 2022**