DERCA: DetERministic Cycle-level Accelerator on Reconfigurable Platforms in DNN-Enabled Real-Time Safety-Critical Systems

The 46th IEEE Real-Time Systems Symposium

Shixin Ji*, Zhuoping Yang*, Xingzhen Chen*, Wei Zhang*, Jinming Zhuang*, Alex K. Jones§, Zheng Dong†, Peipei Zhou*

Brown University*; Wayne State University† ;
Syracuse University§

shixin_ji@brown.edu
peipei_zhou@brown.edu
https://peipeizhou-eecs.github.io/

GitHub Repo:
https://github.com/arc-research-lab/DERCA

BROWN

WAYNE STATE UNIVERSITY

Syracuse University

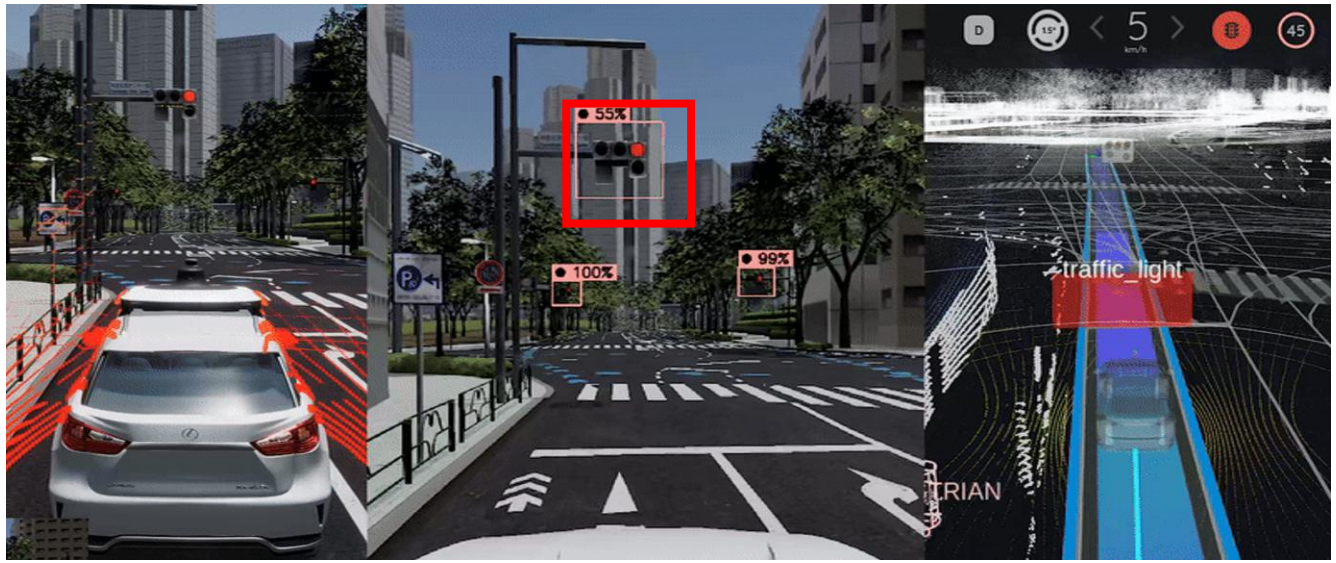
Background: Autonomous Driving Systems

Background: Autonomous Driving Systems

Camera

Lidar

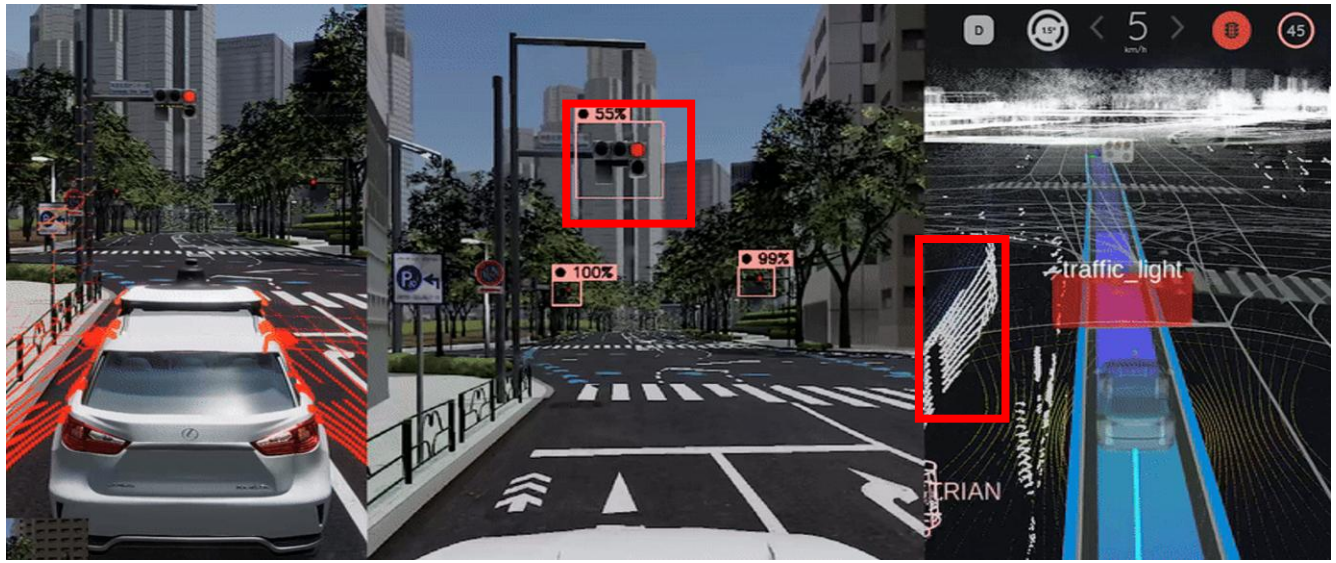
Background: Autonomous Driving Systems



Camera  Traffic light

Lidar 

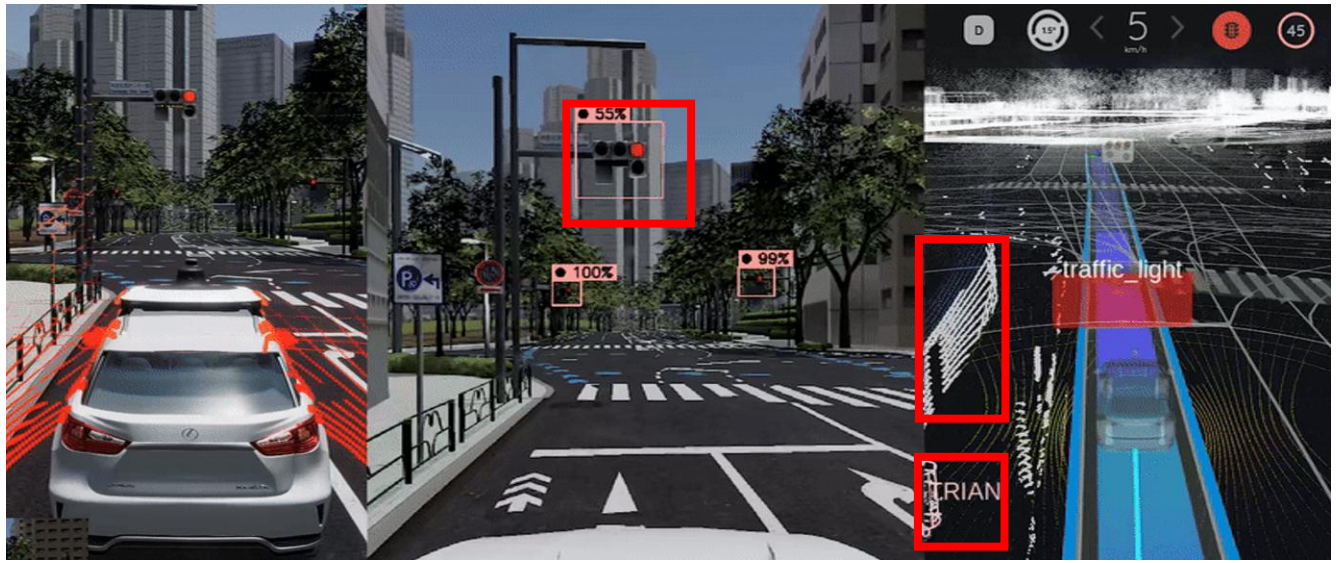
Background: Autonomous Driving Systems

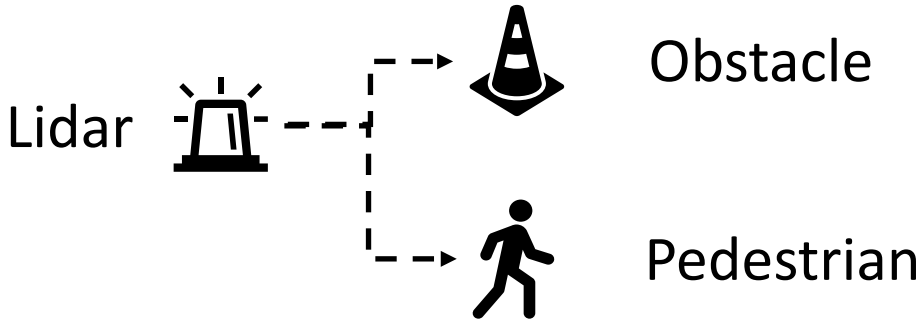


Camera ------> Traffic light

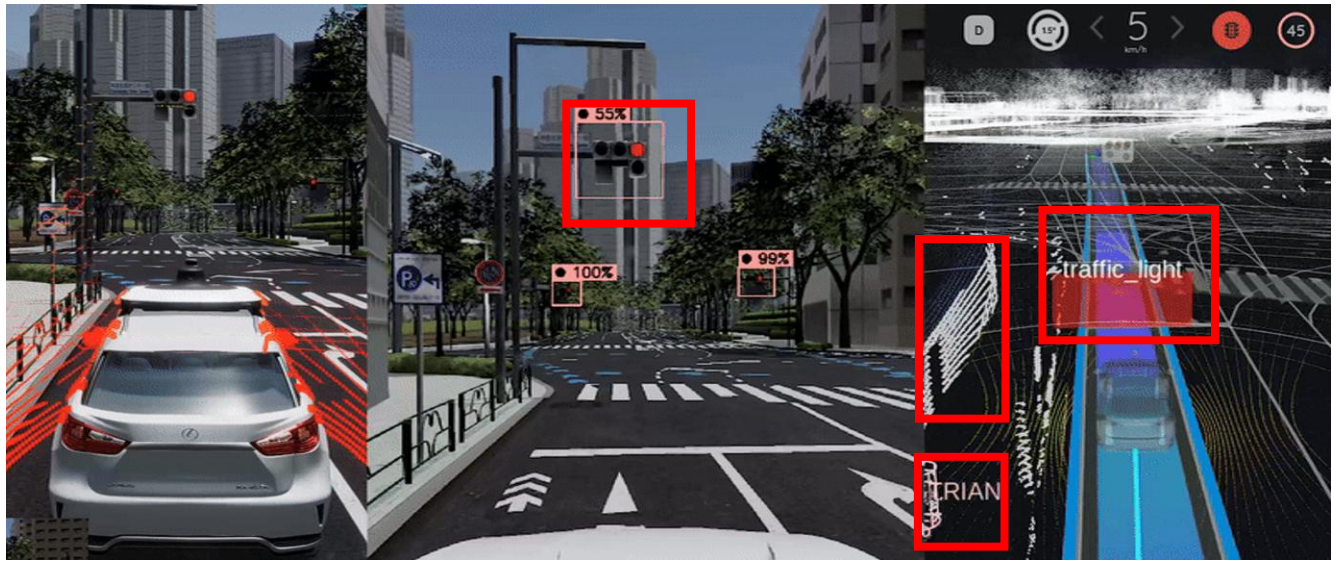
Lidar - - -> Obstacle

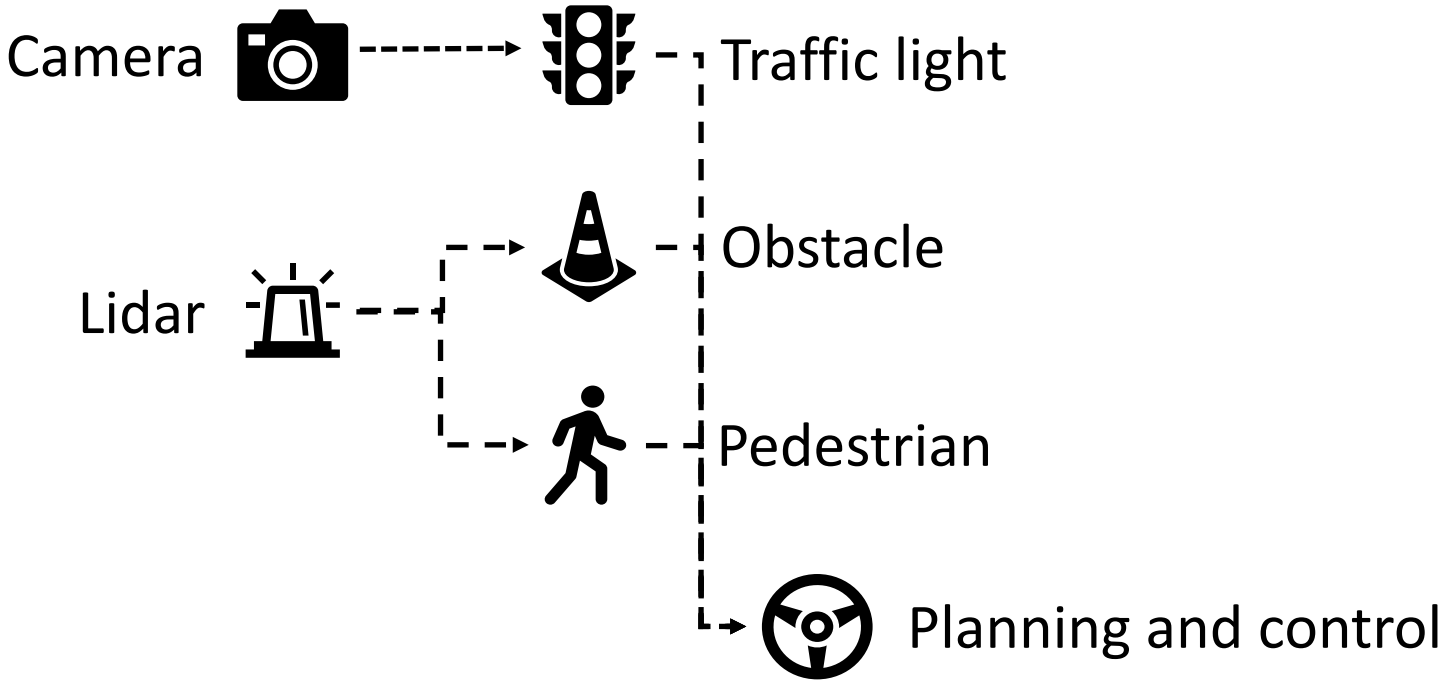
Background: Autonomous Driving Systems



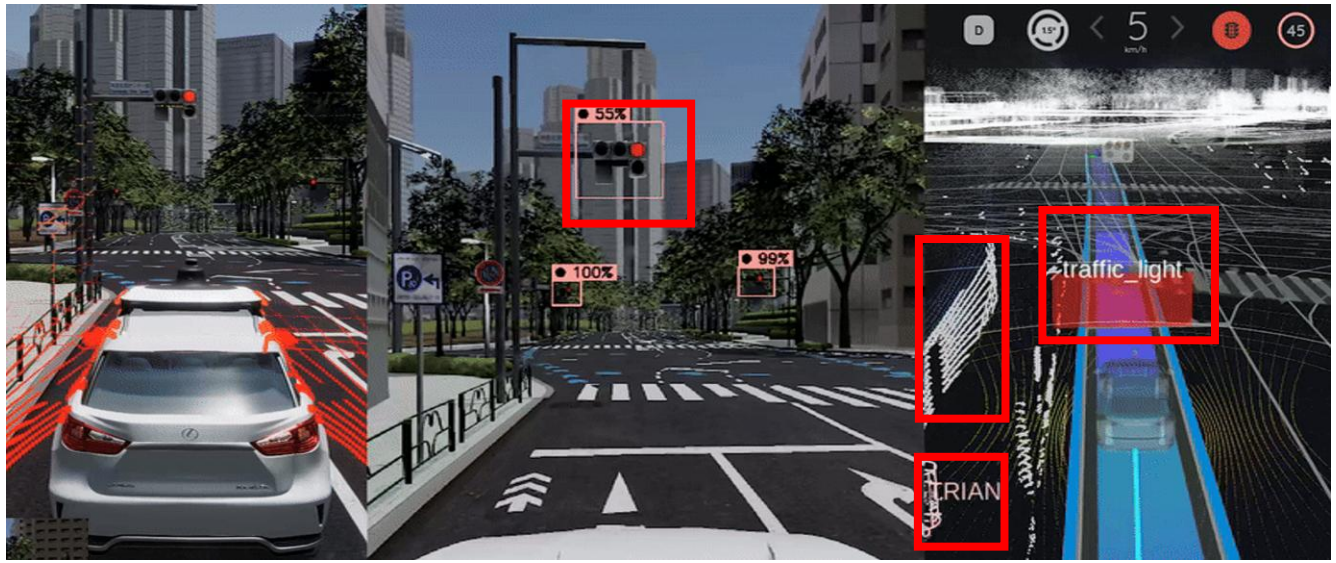


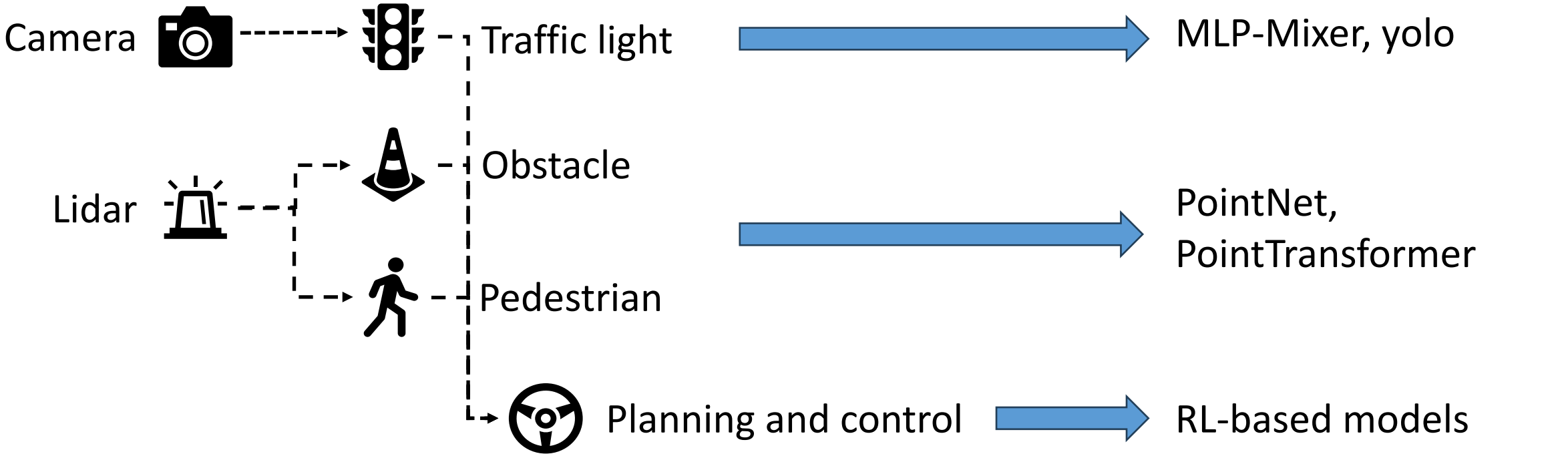
Background: Autonomous Driving Systems



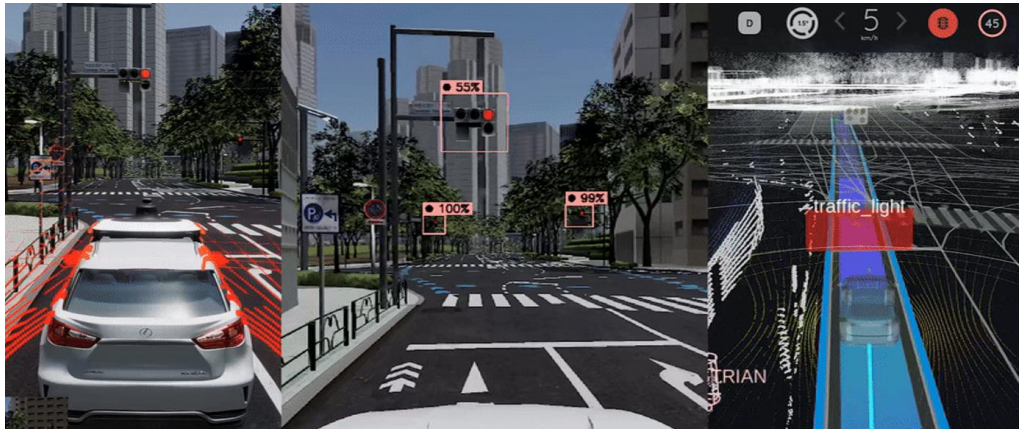


Background: Autonomous Driving Systems



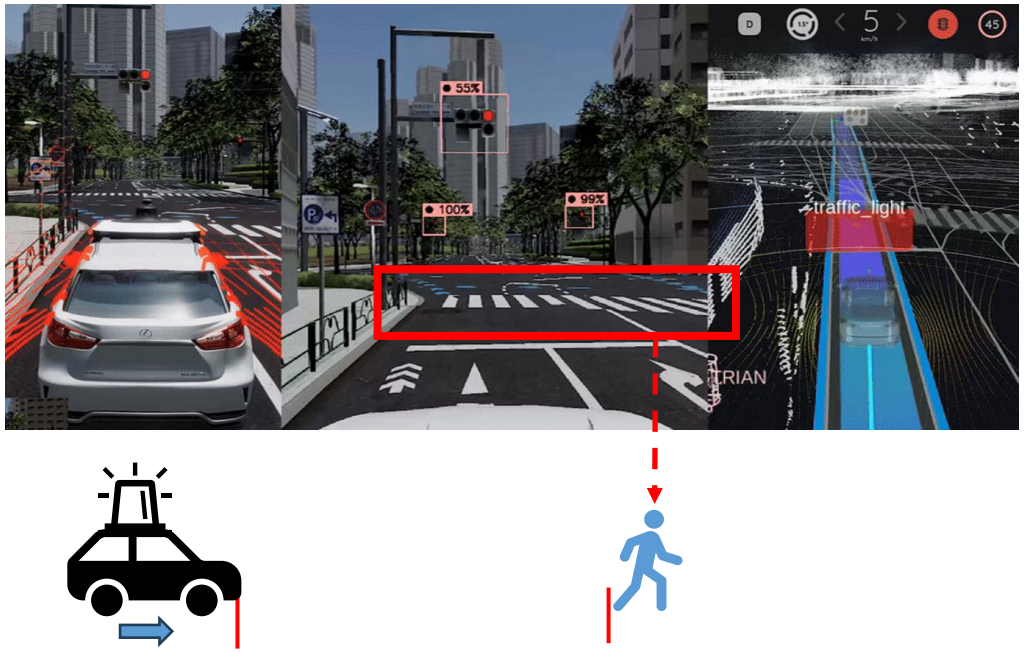


Background: The need for Autonomous Driving System



Safety is the most important!

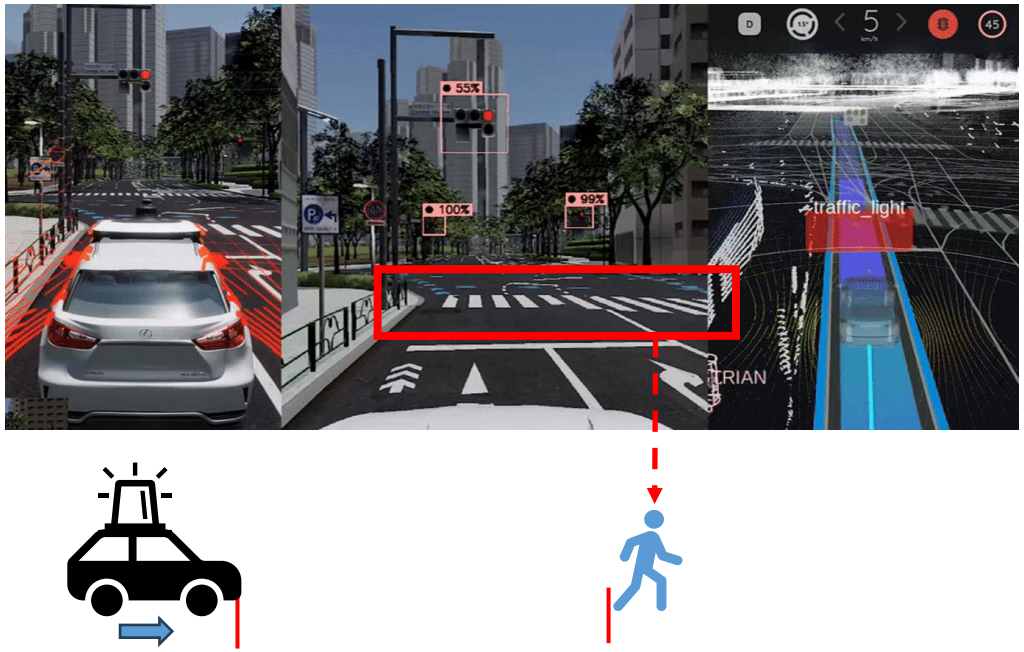
Background: The need for Autonomous Driving System



Safety is the most important!

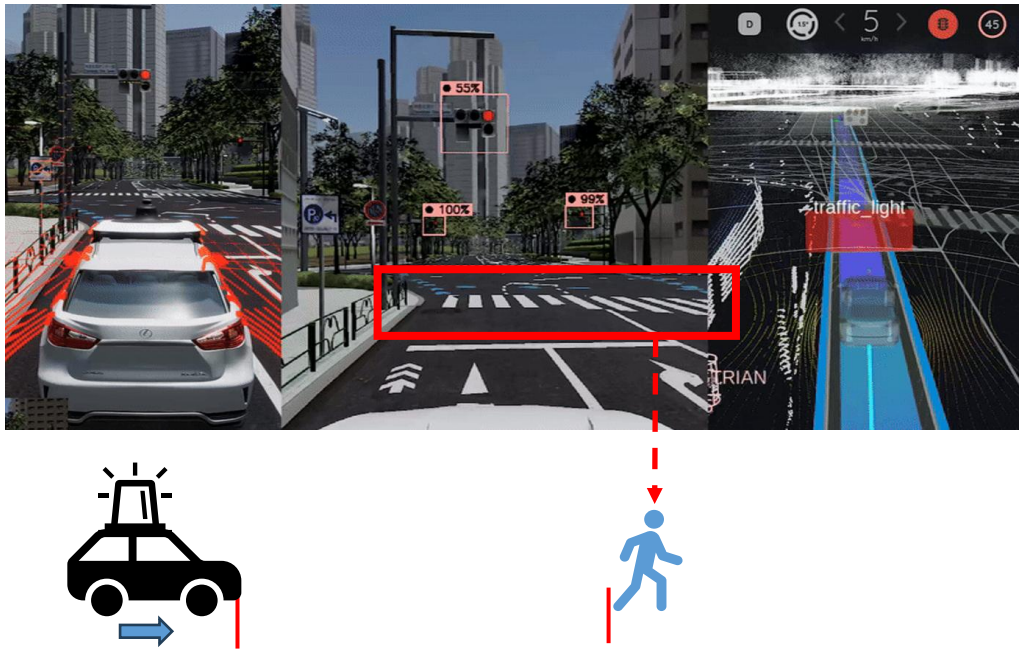
t

Background: The need for Autonomous Driving System



Safety is the most important!

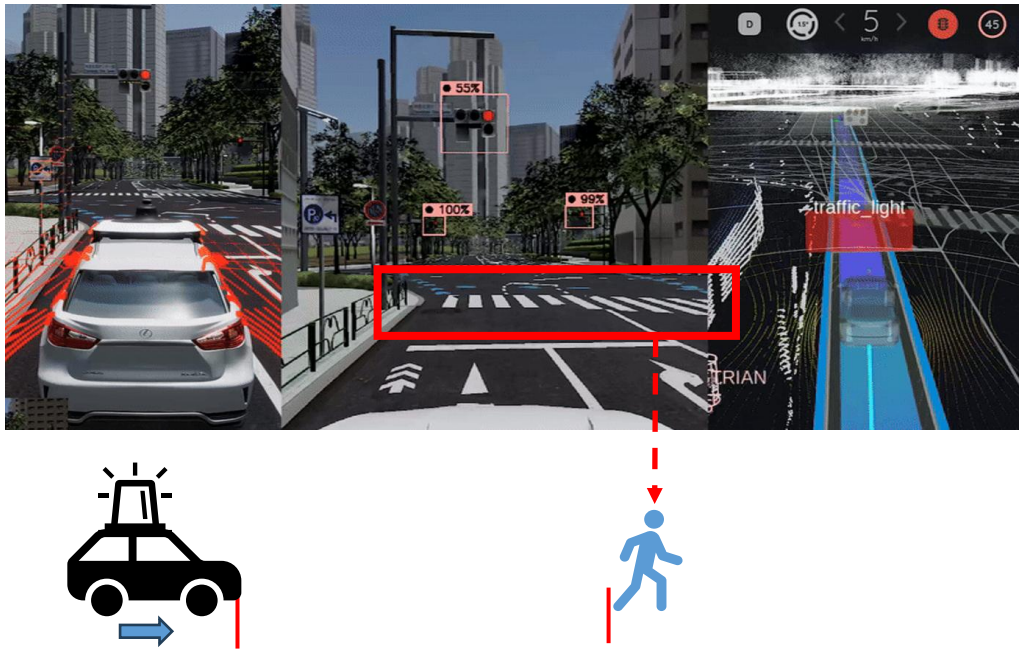
Background: The need for Autonomous Driving System



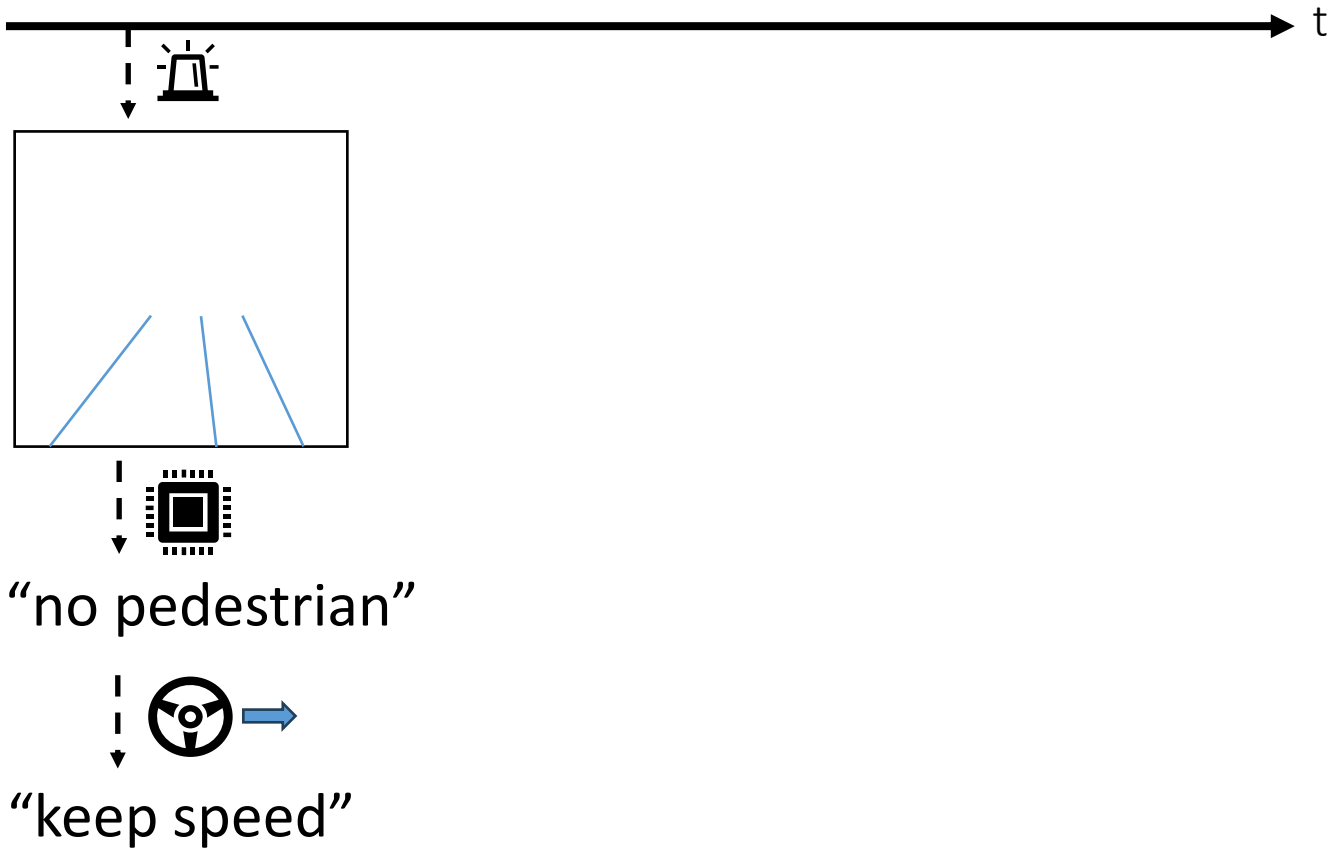
Safety is the most important!

“no pedestrian”

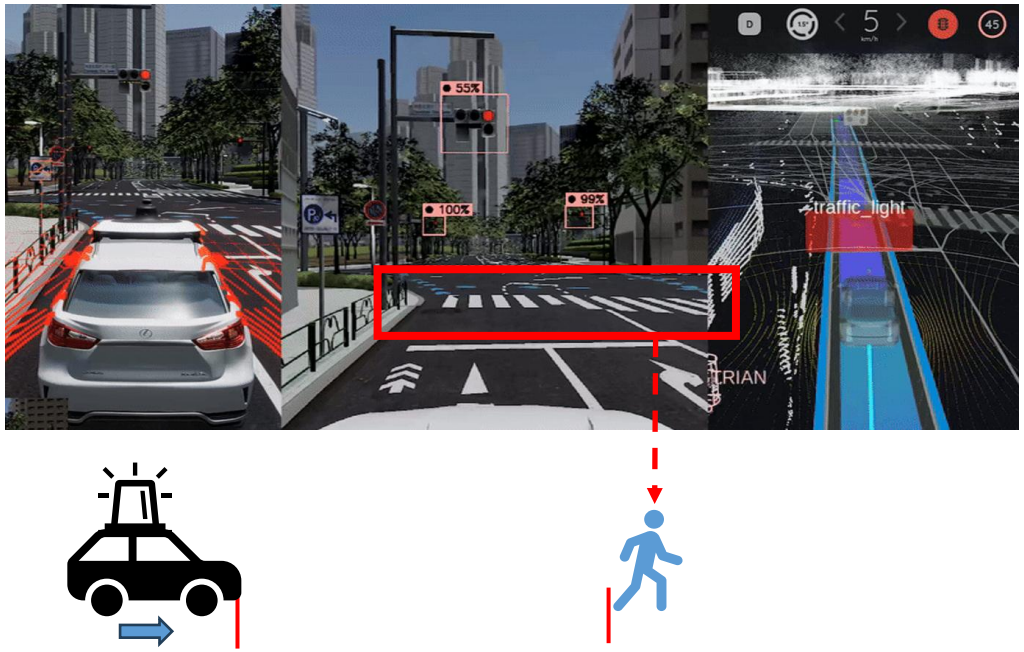
Background: The need for Autonomous Driving System



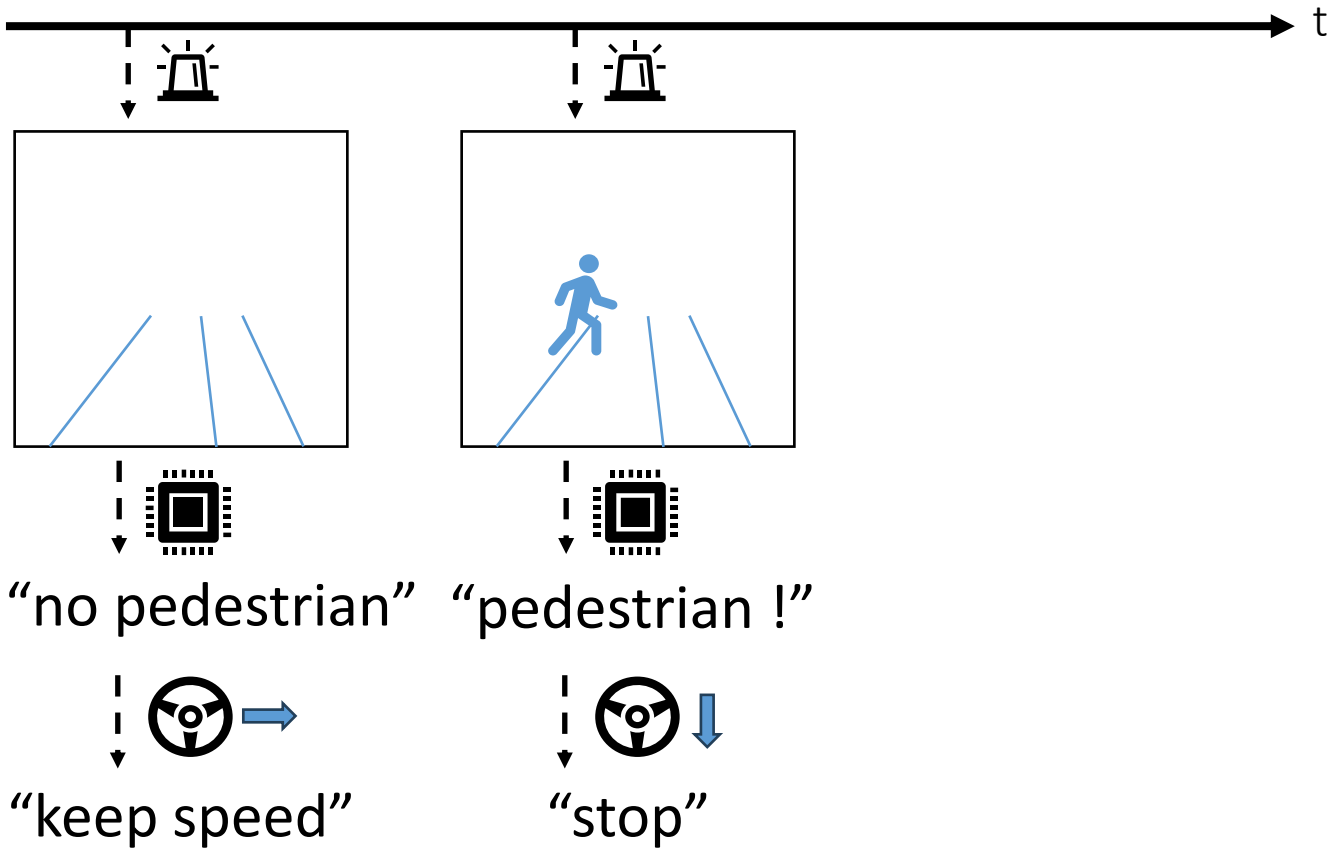
Safety is the most important!



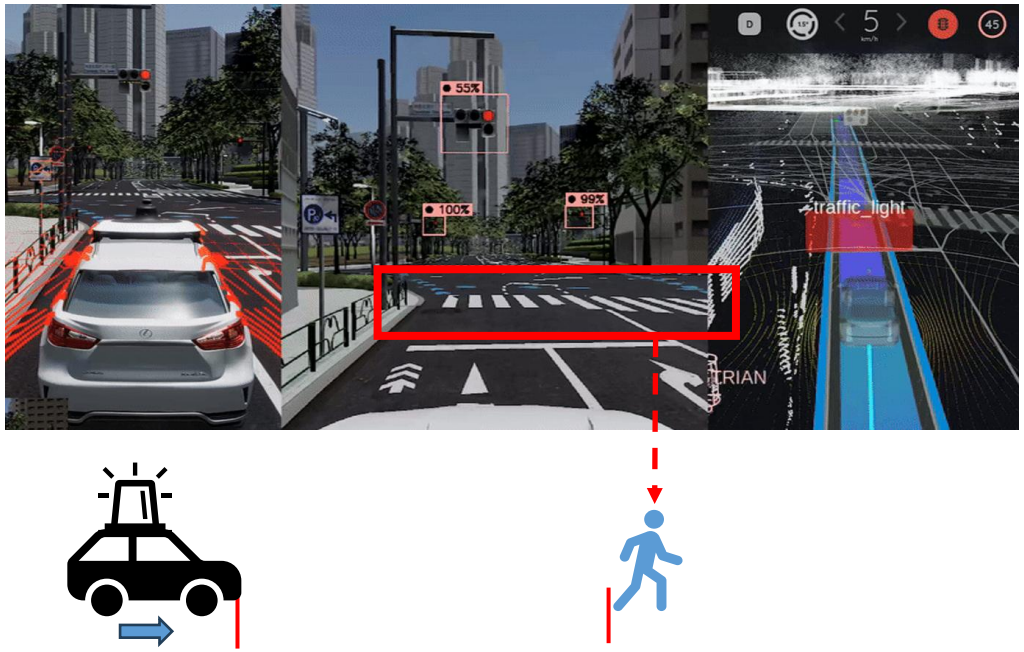
Background: The need for Autonomous Driving System



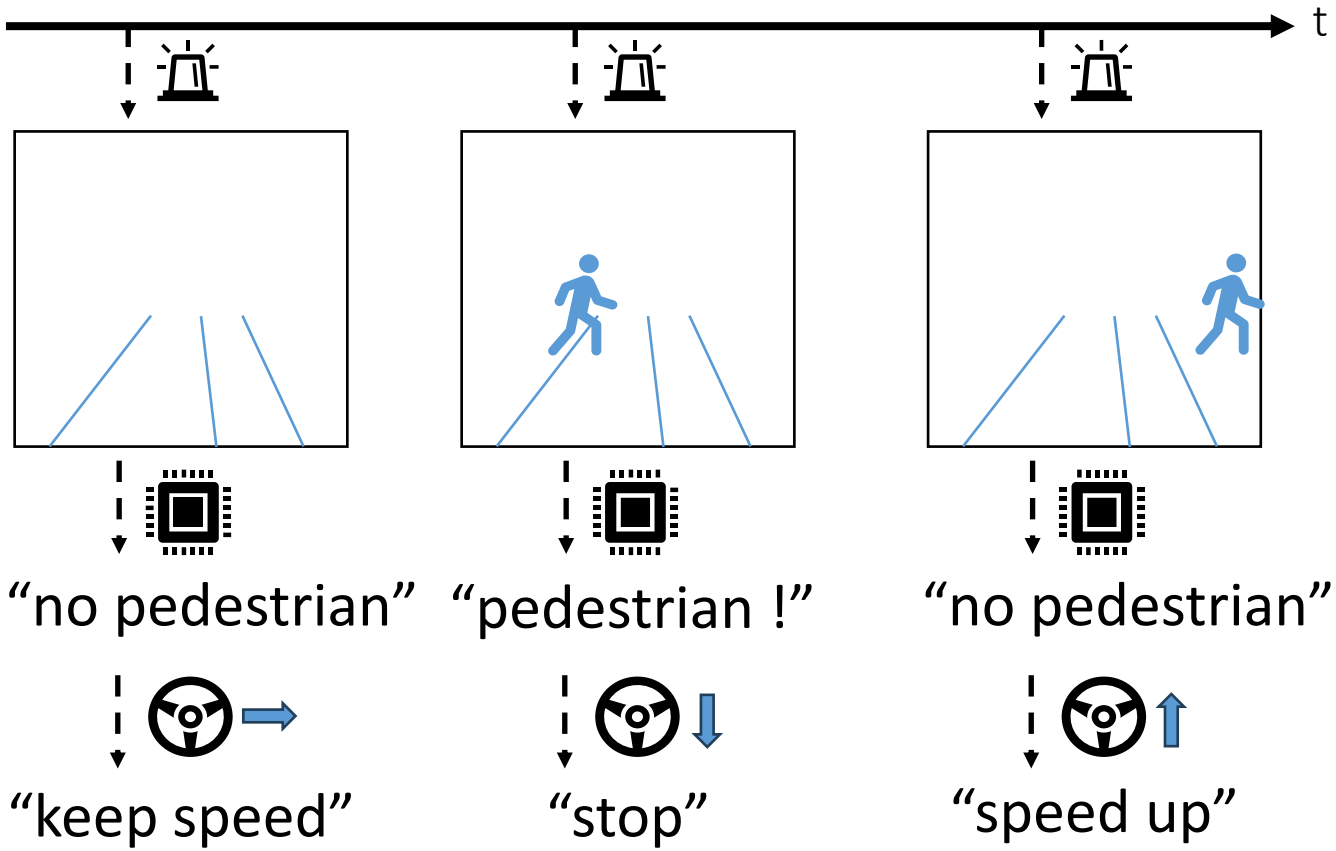
Safety is the most important!



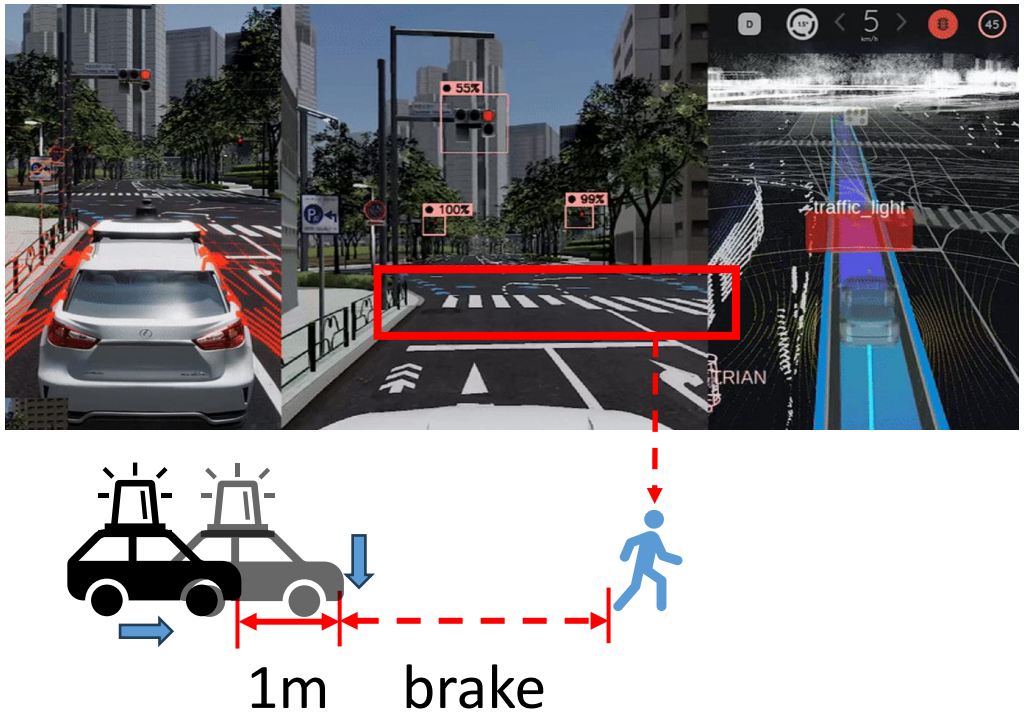
Background: The need for Autonomous Driving System



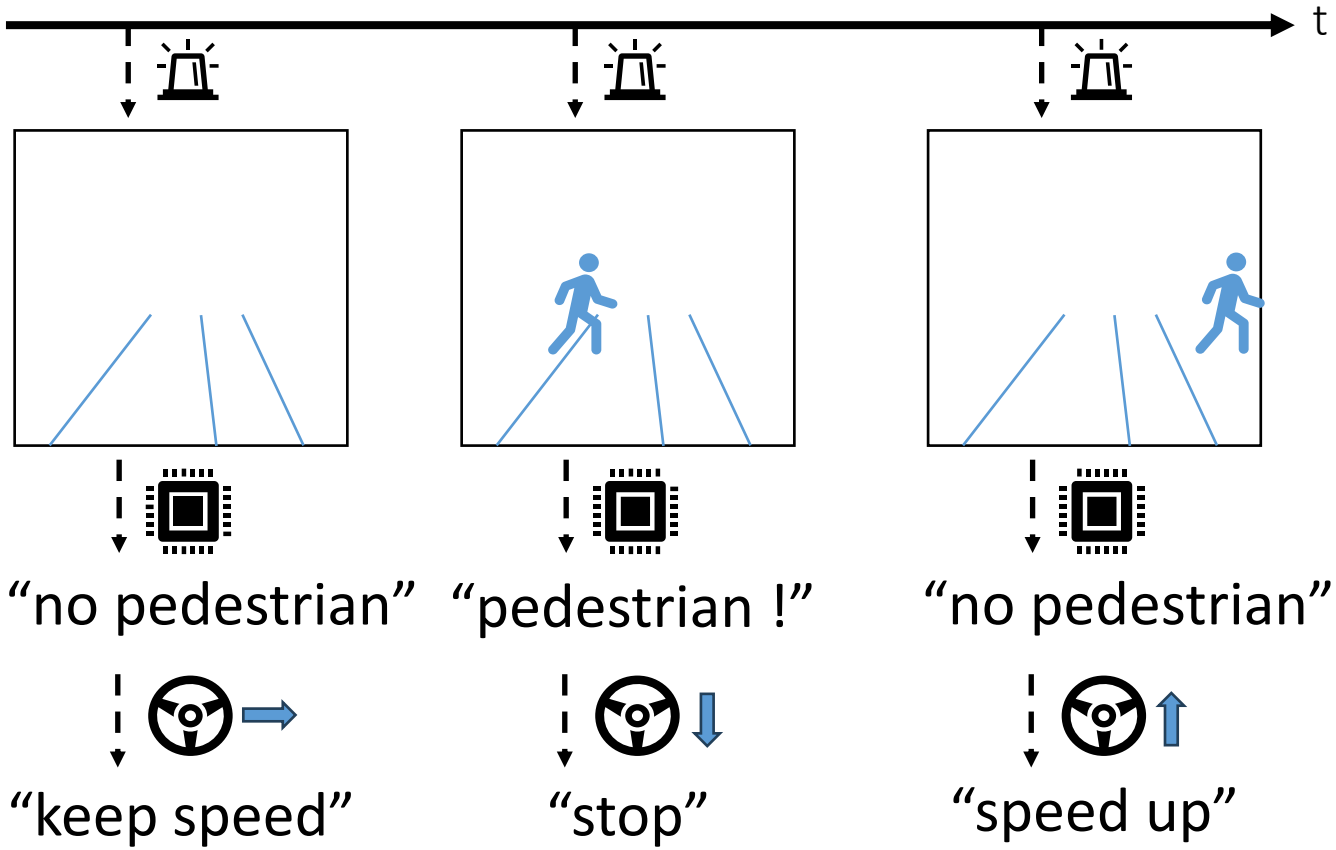
Safety is the most important!



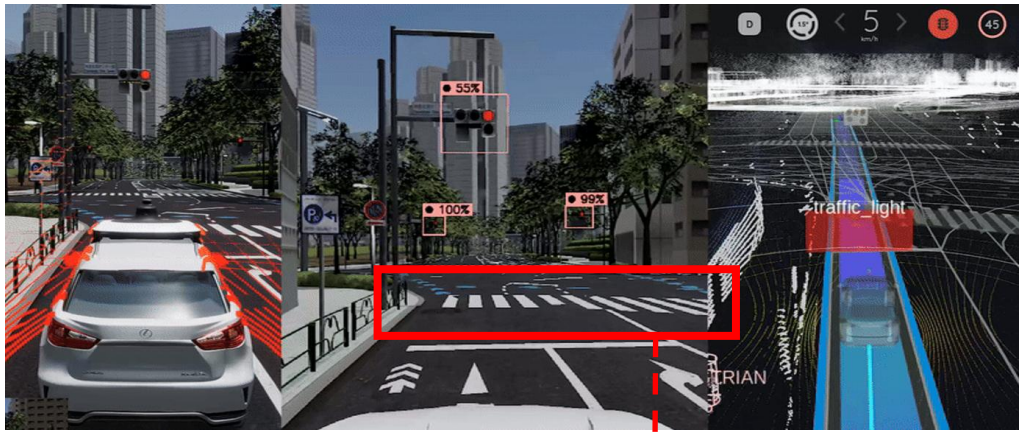
Background: The need for Autonomous Driving System



Safety is the most important!

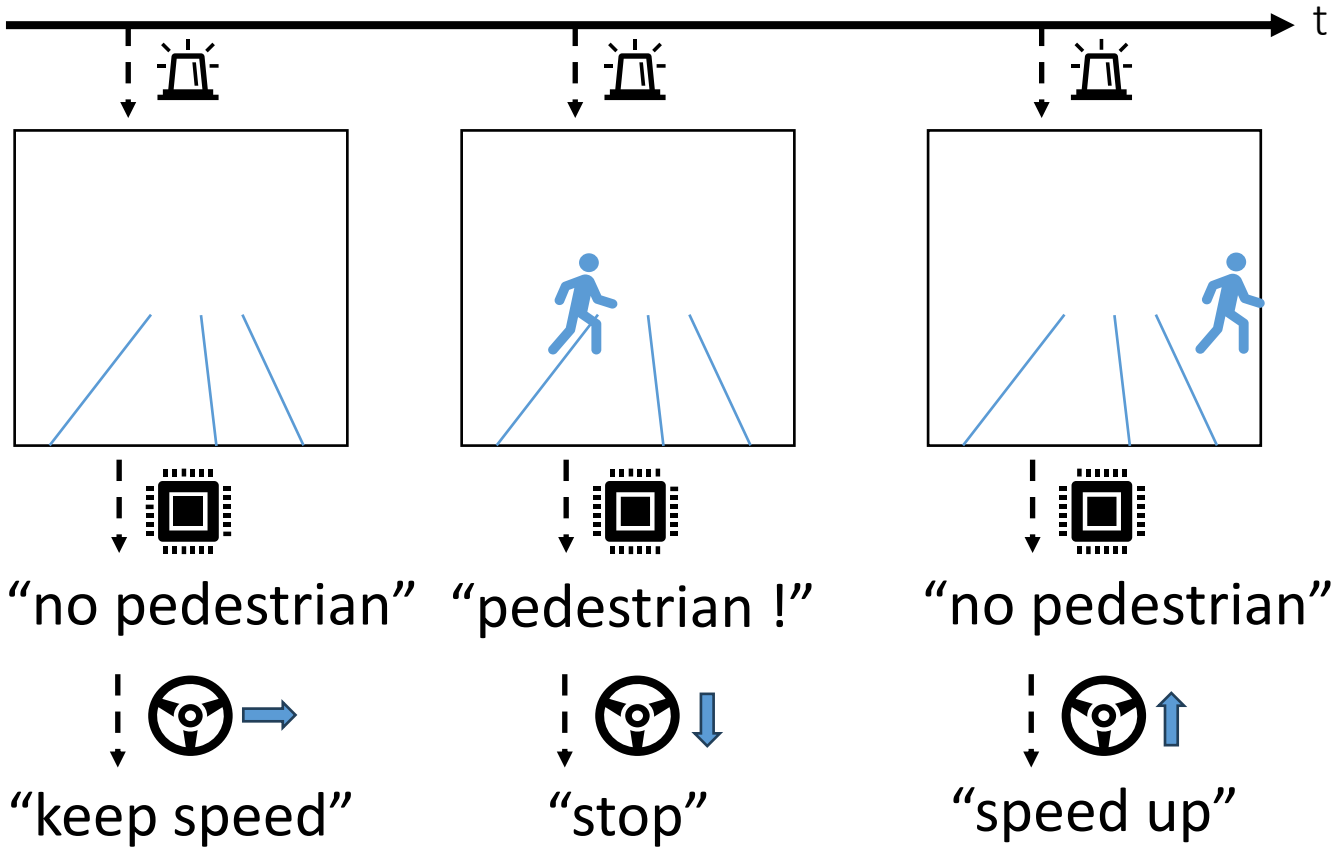


Background: The need for Autonomous Driving System

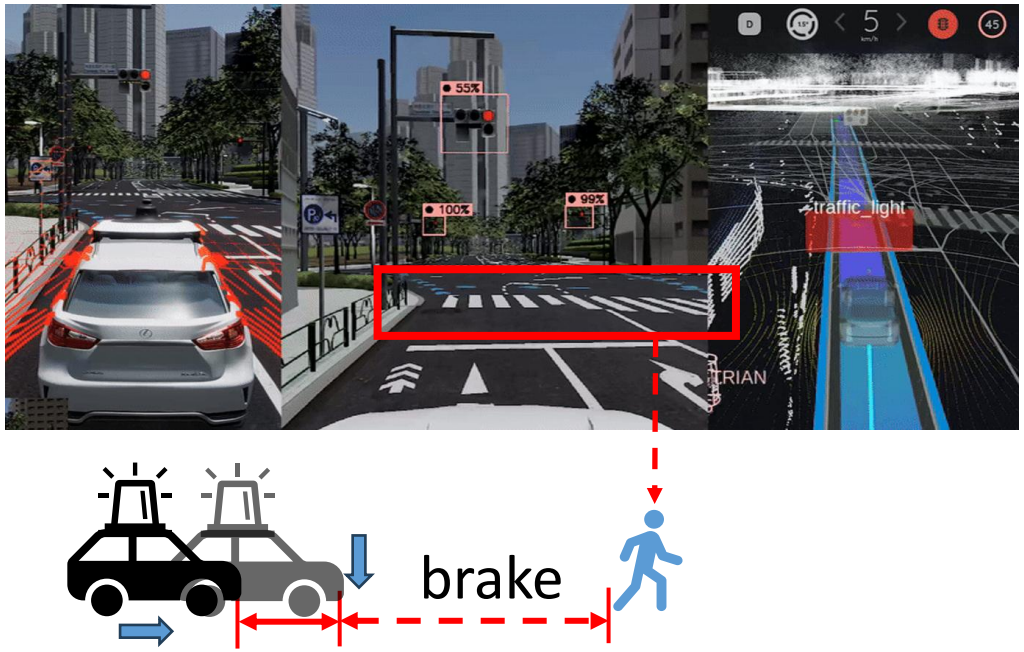


- If 120km/h (75 mile/h, 33.3 m/s)
- Gap : 30 ms

Safety is the most important!



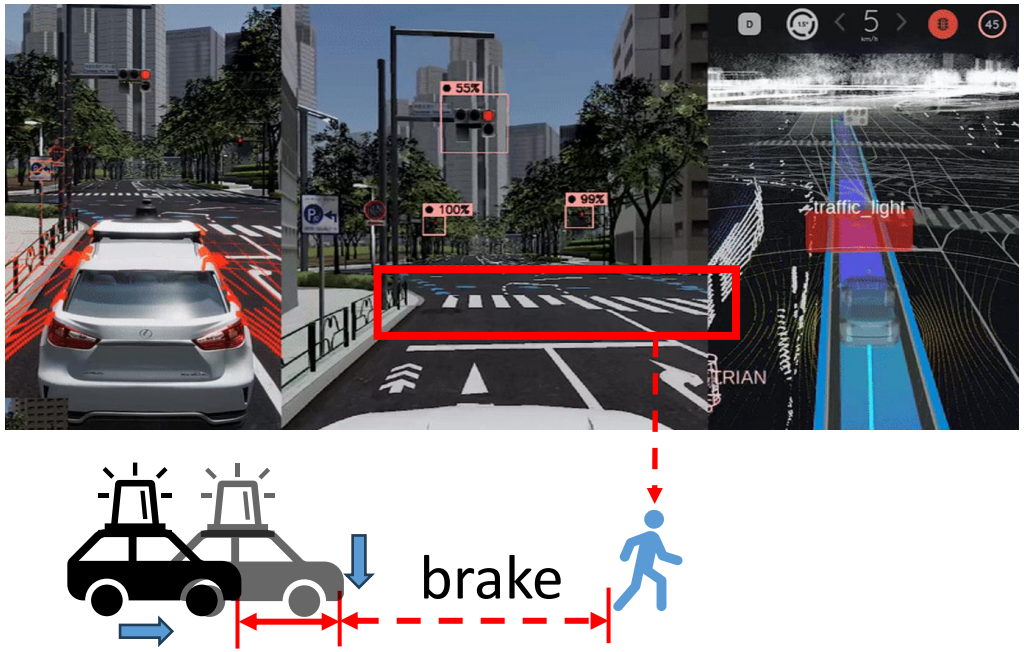
Background: The need for Autonomous Driving System



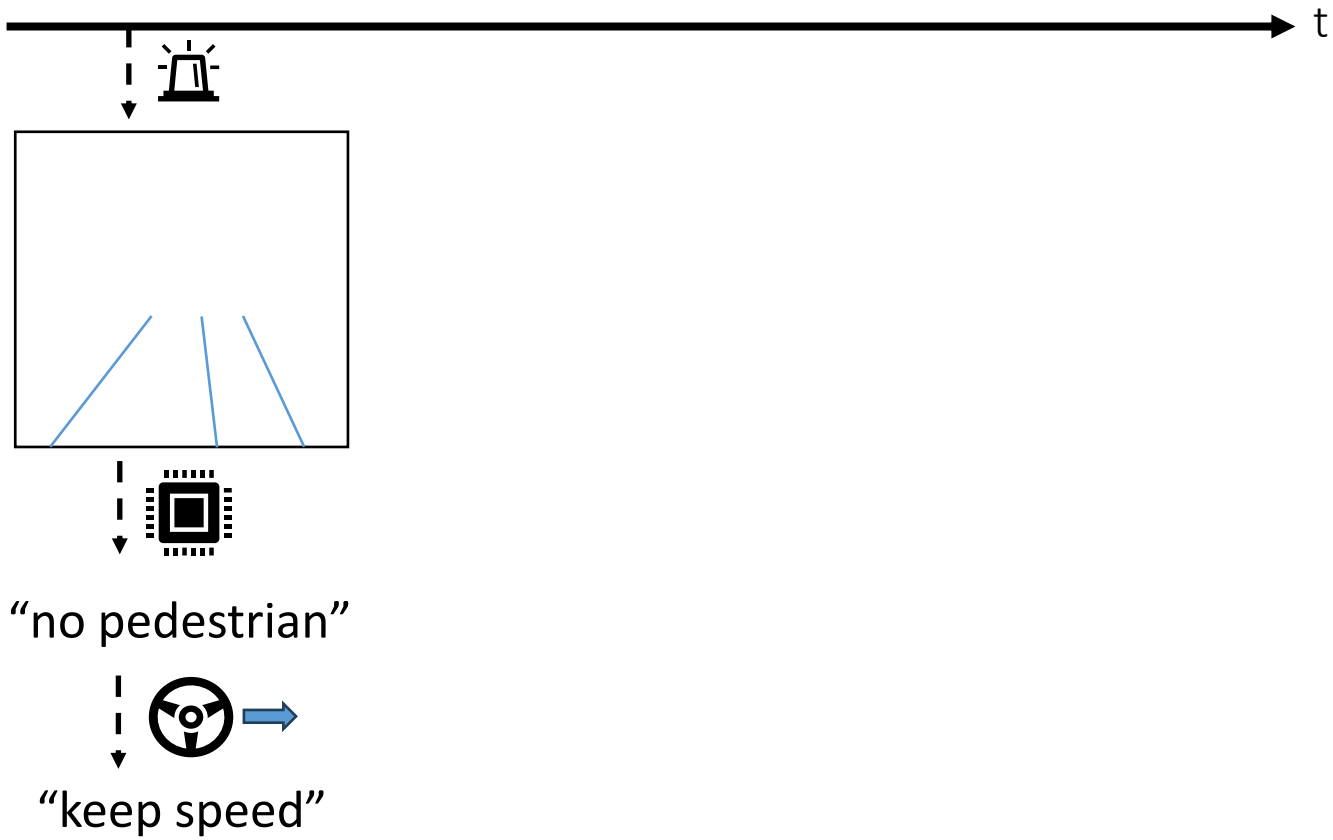
Safety is the most important!

t

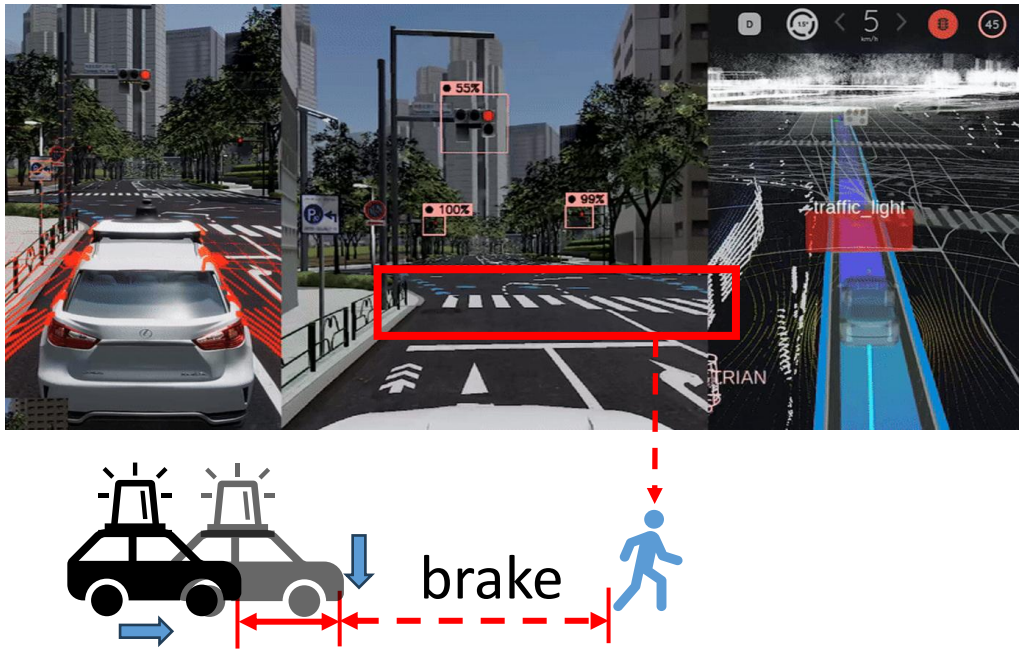
Background: The need for Autonomous Driving System



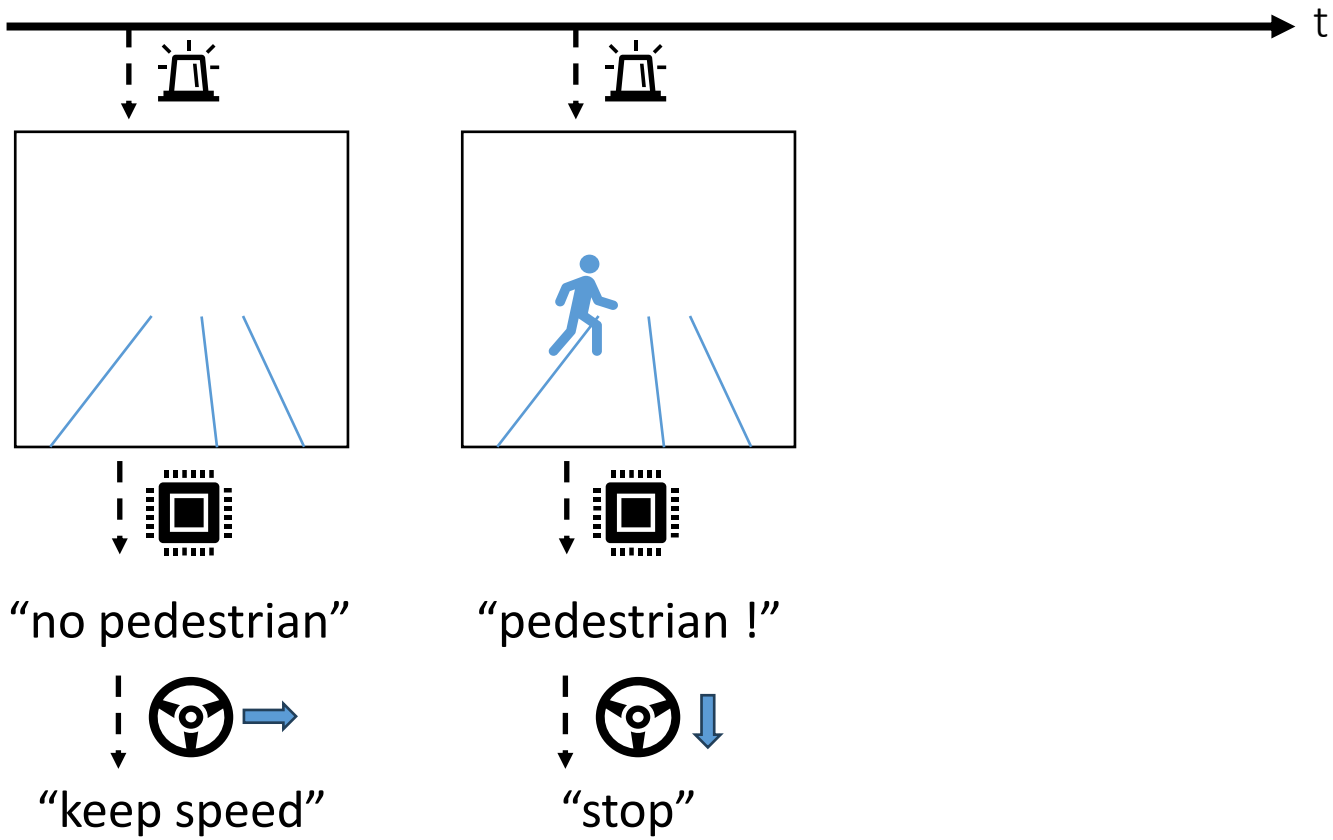
Safety is the most important!



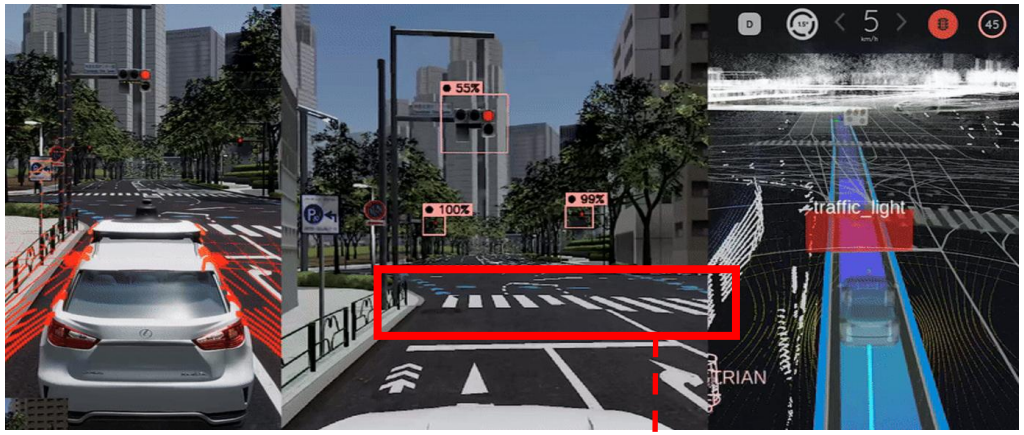
Background: The need for Autonomous Driving System



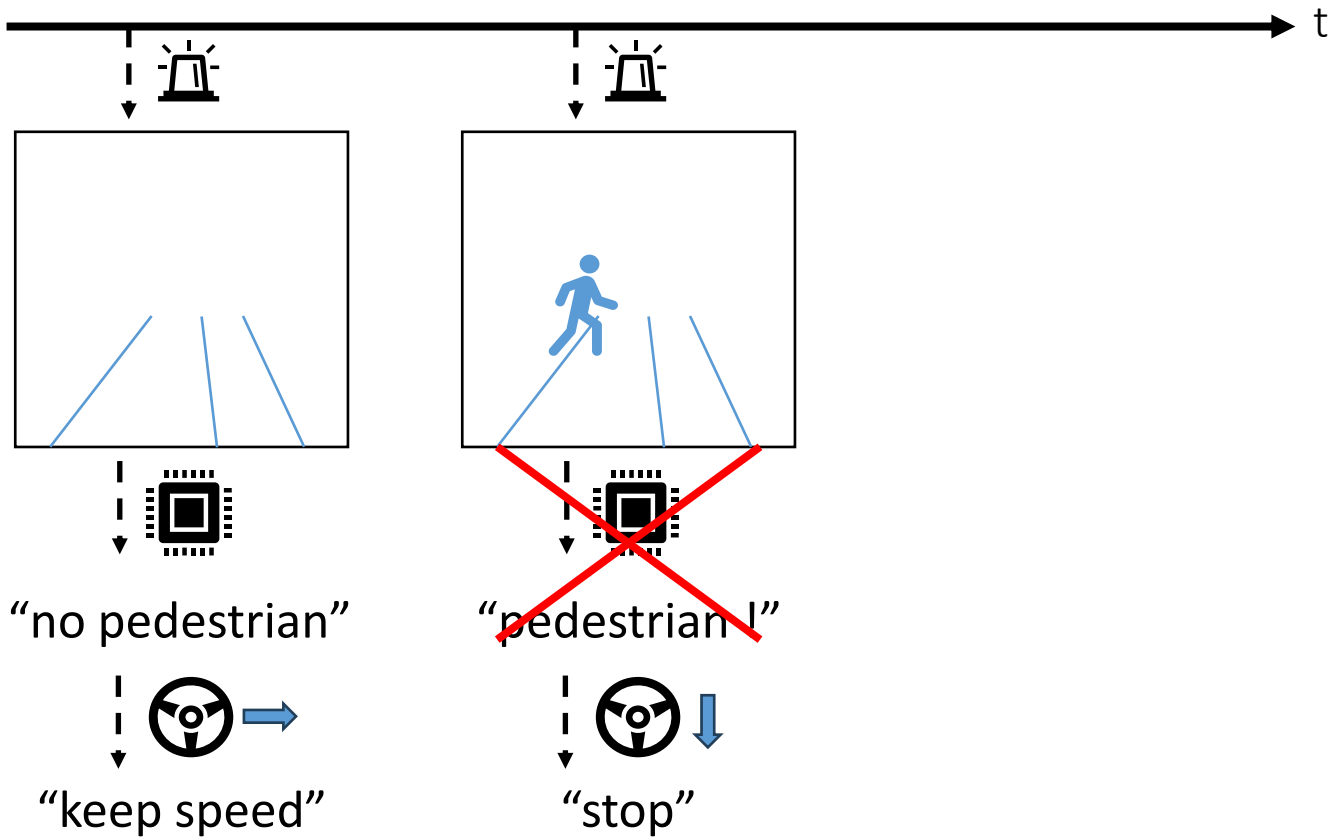
Safety is the most important!



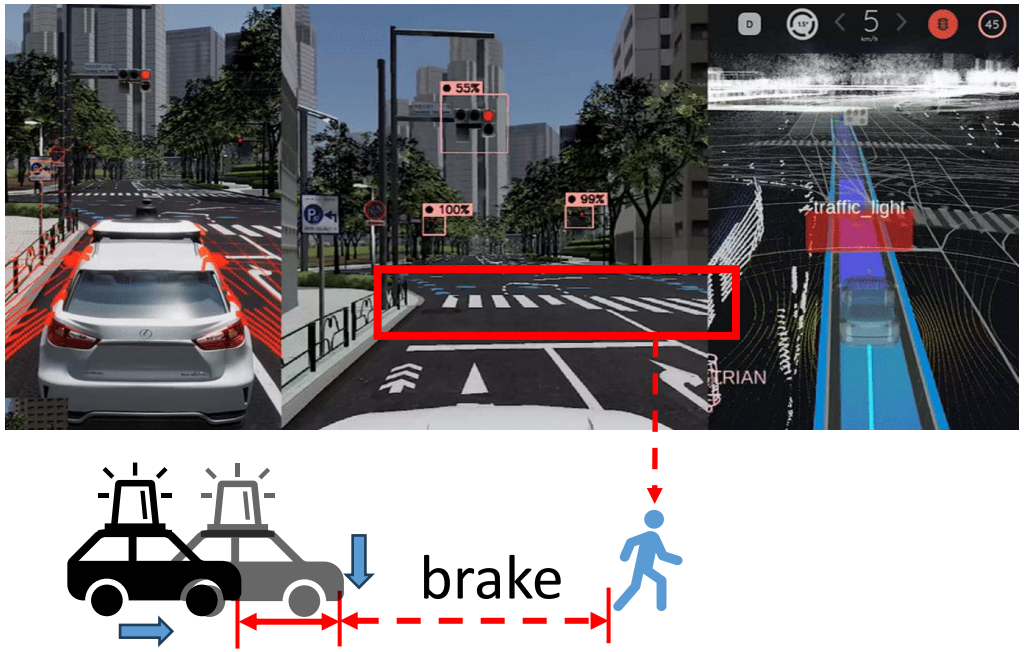
Background: The need for Autonomous Driving System



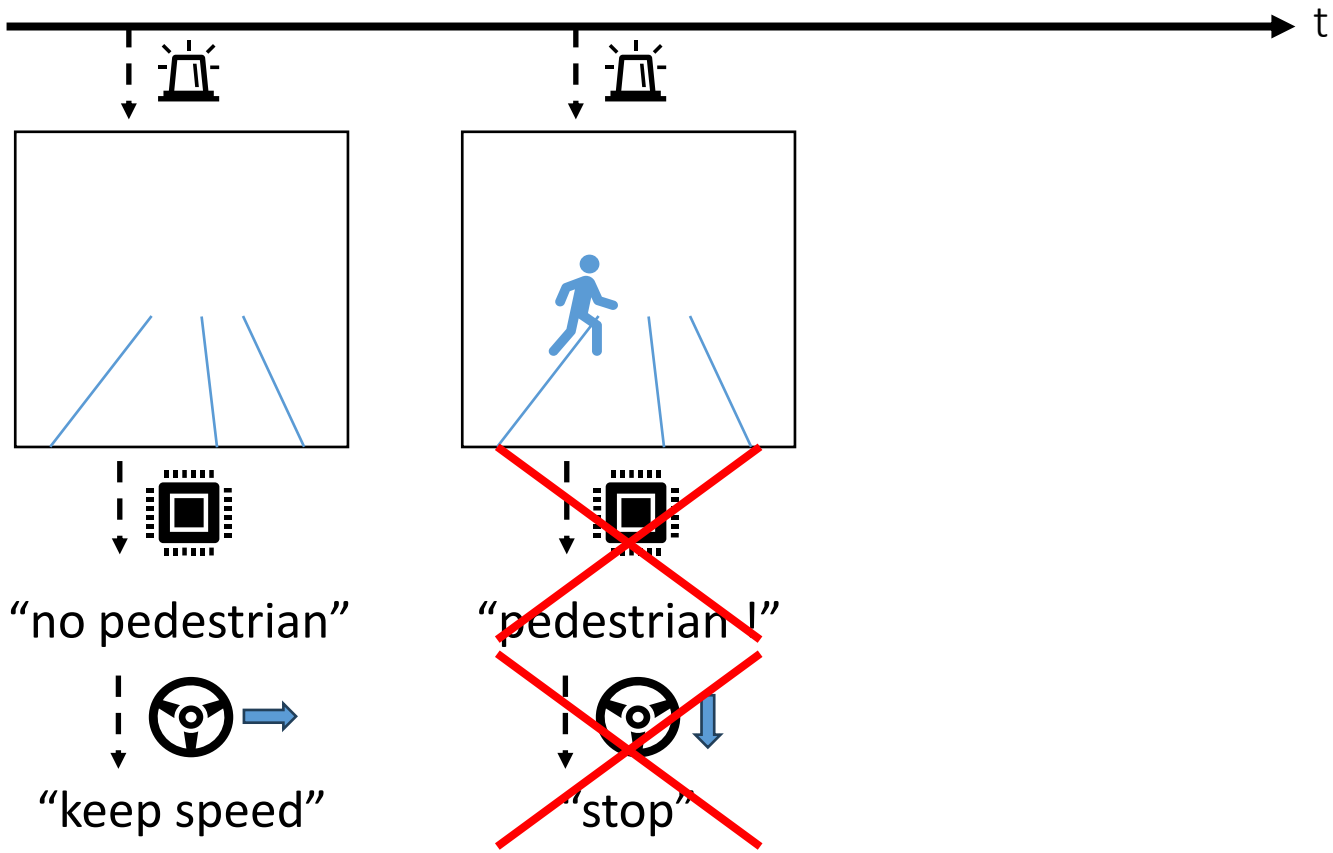
Safety is the most important!



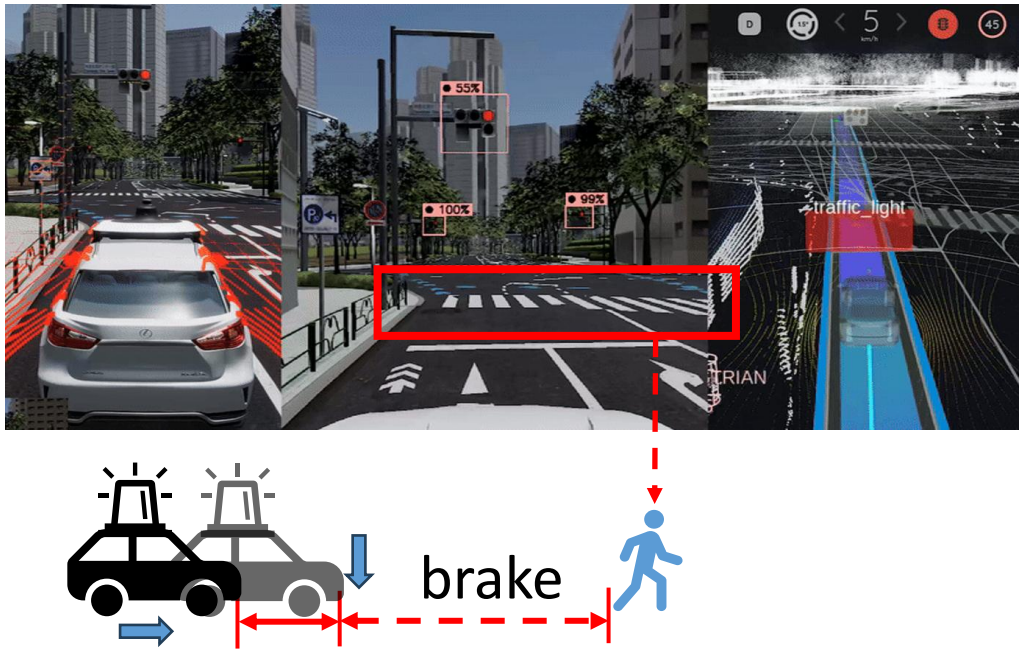
Background: The need for Autonomous Driving System



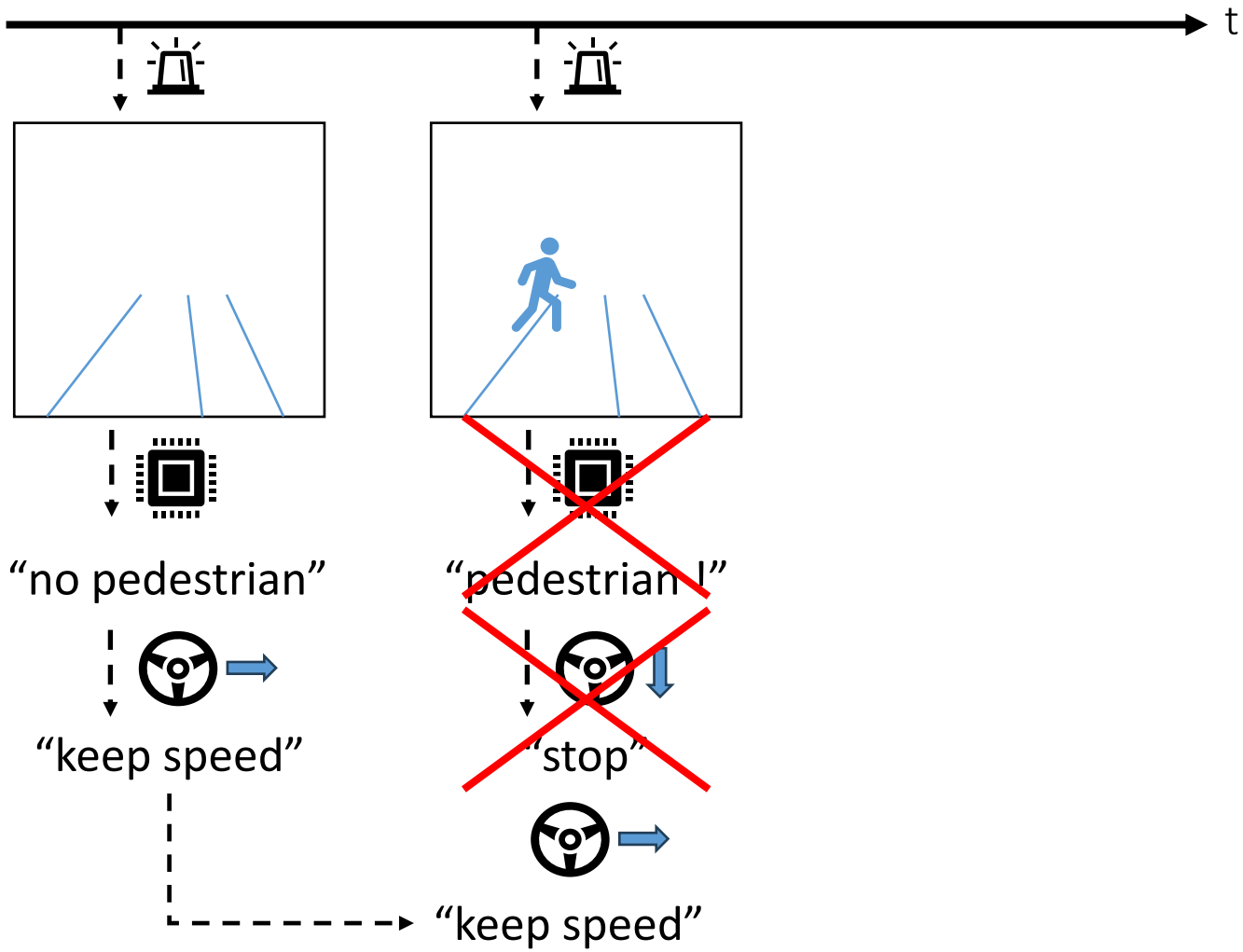
Safety is the most important!



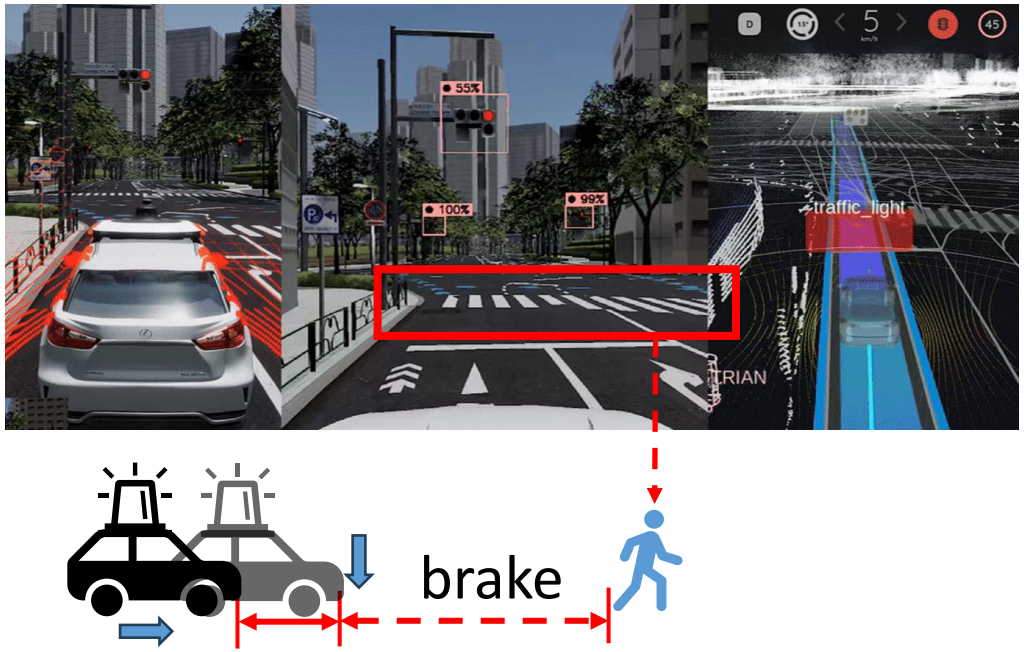
Background: The need for Autonomous Driving System



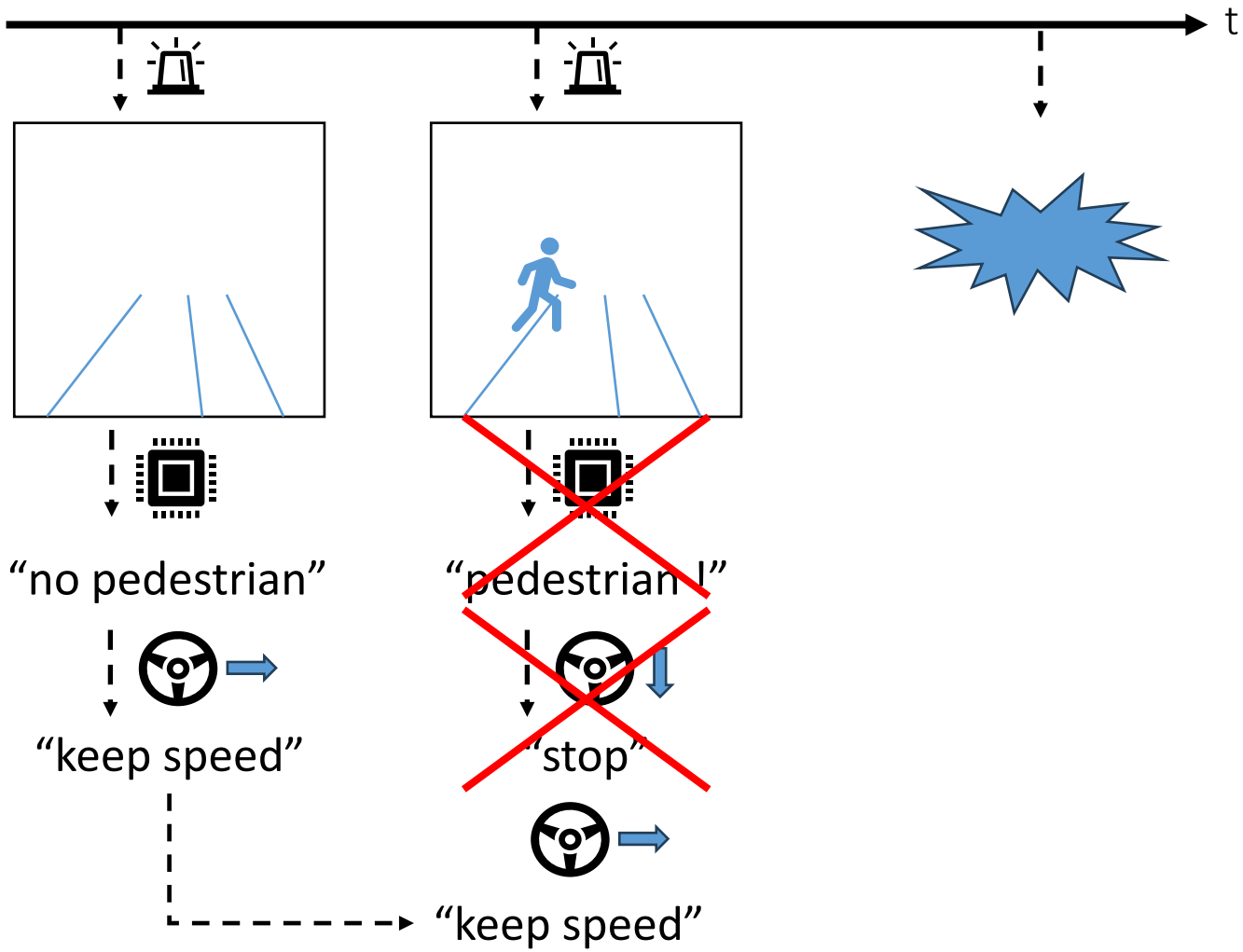
Safety is the most important!



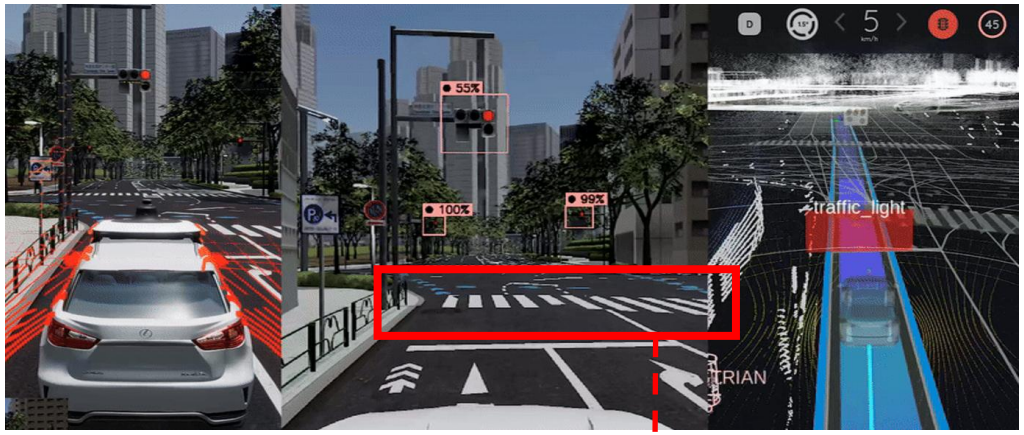
Background: The need for Autonomous Driving System

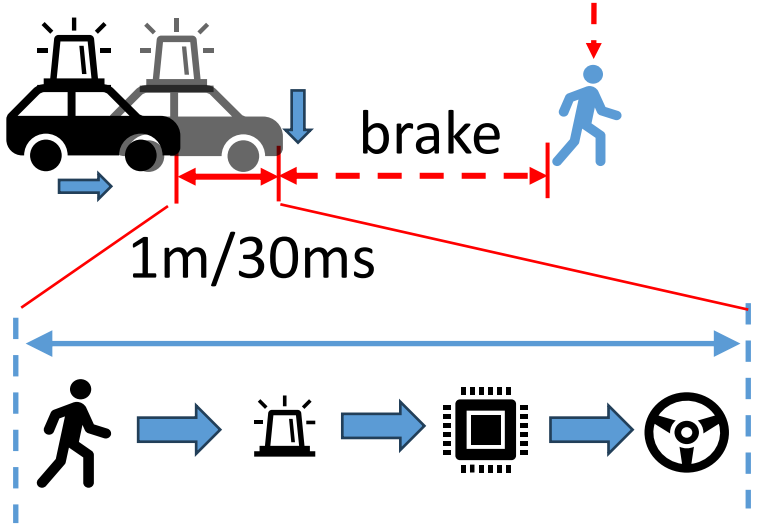


Safety is the most important!

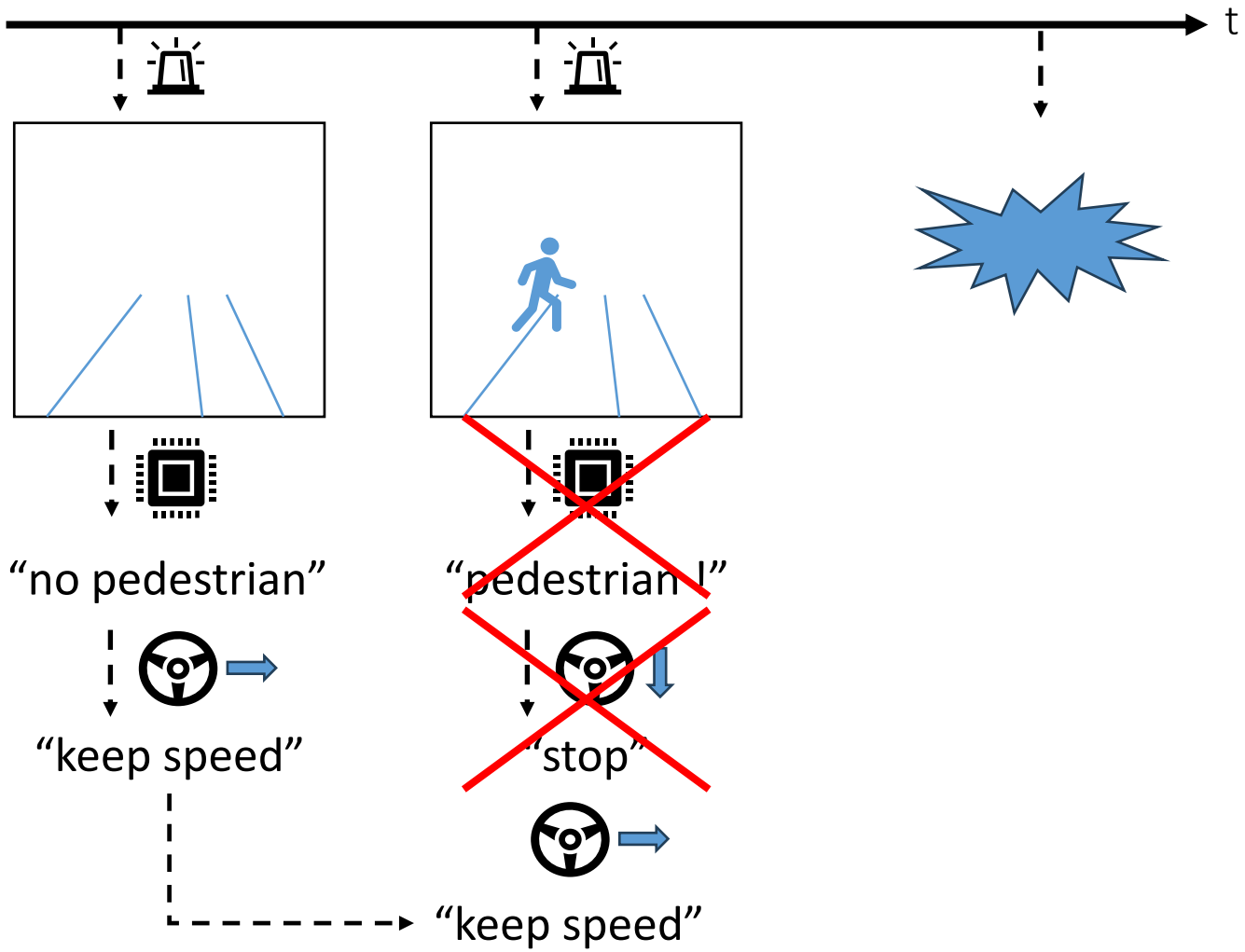


Background: The need for Autonomous Driving System

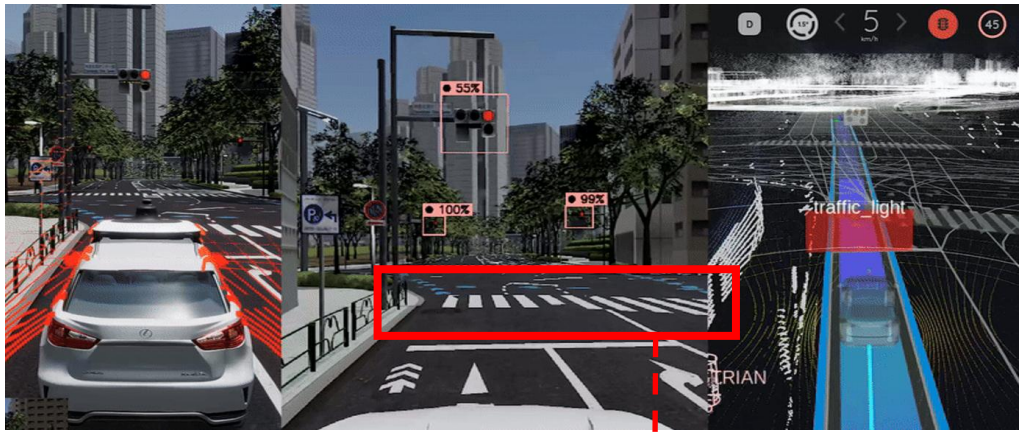


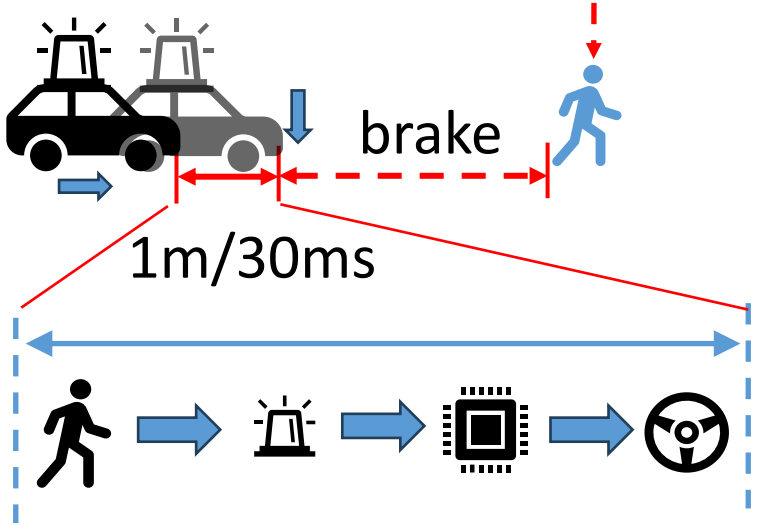


Safety is the most important!



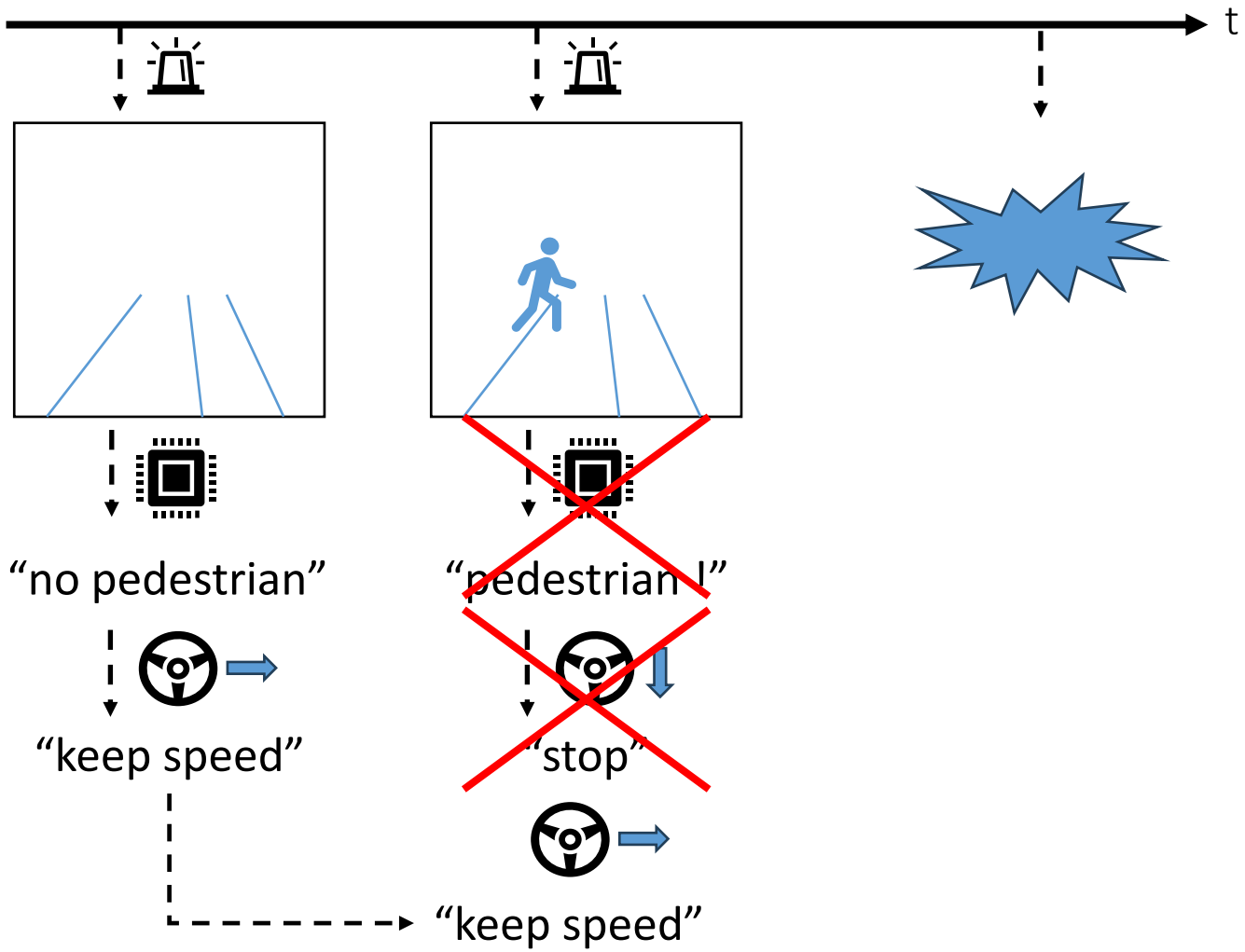
Background: The need for Autonomous Driving System



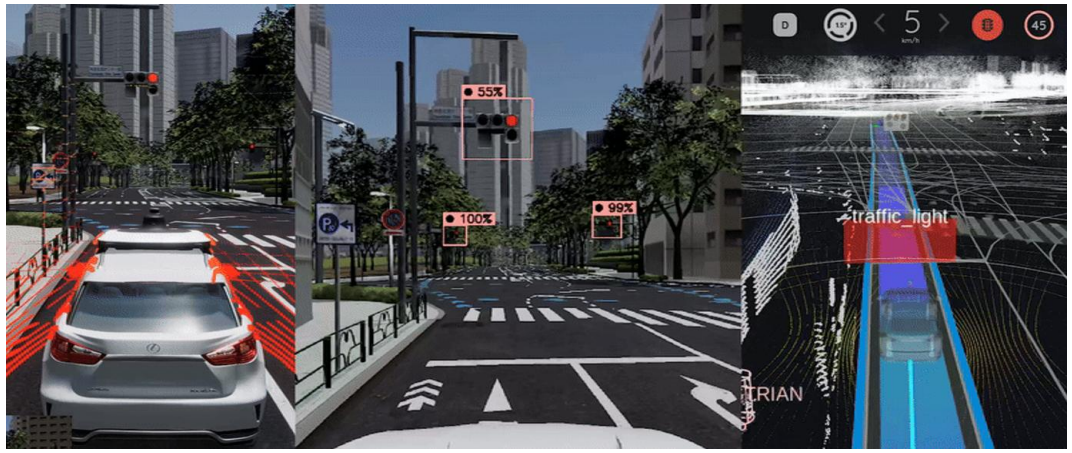


Latency bounded ➡ System predictable

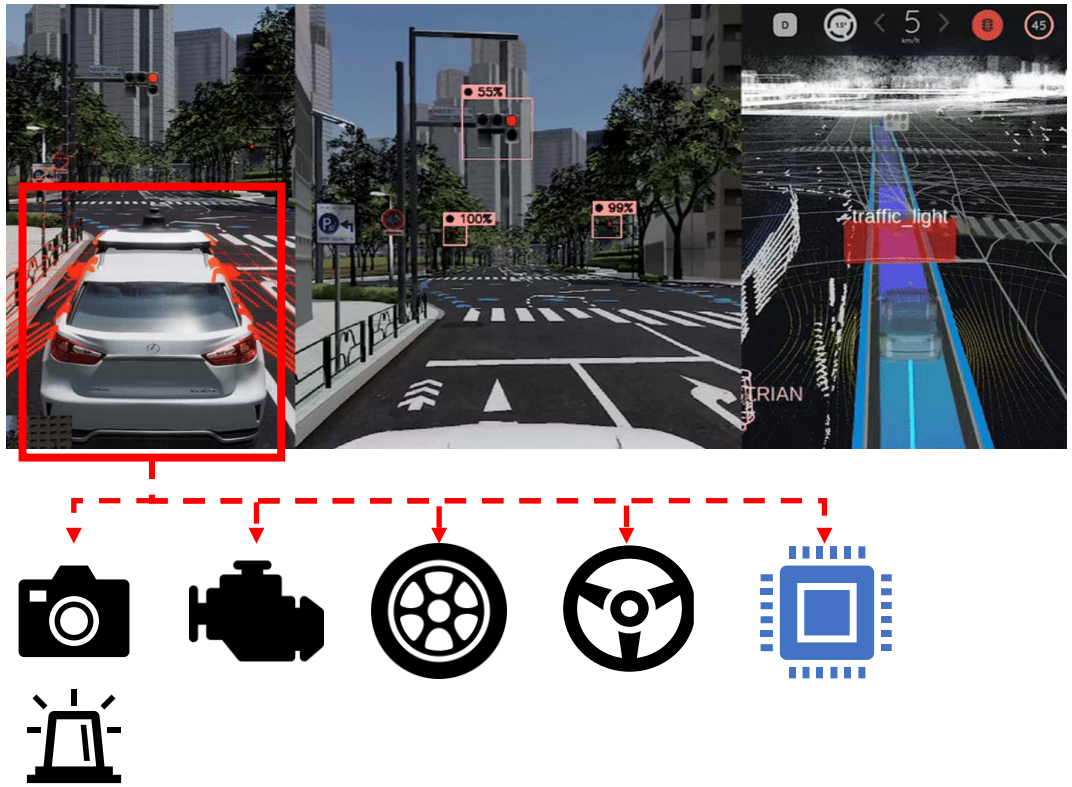
Safety is the most important!



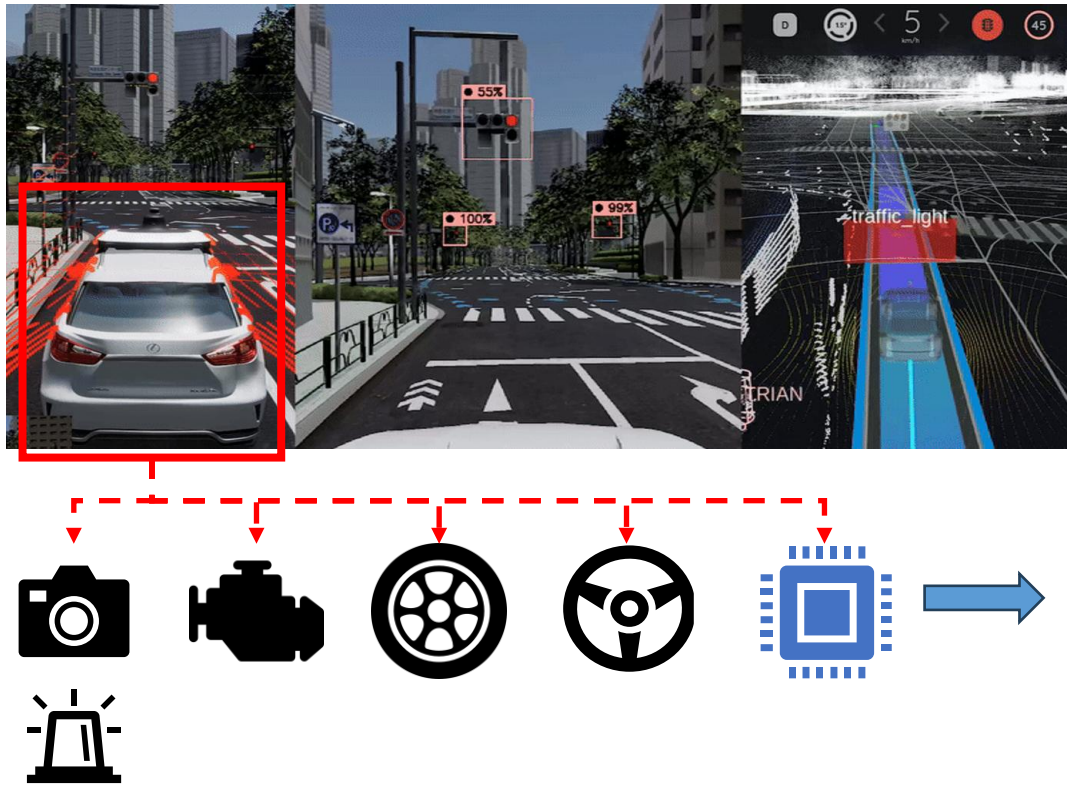
Background: The Hardware Computing Platform



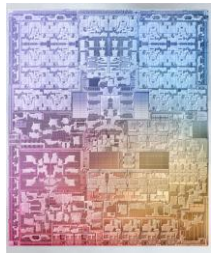
Background: The Hardware Computing Platform



Background: The Hardware Computing Platform



Apple M4



CPUs

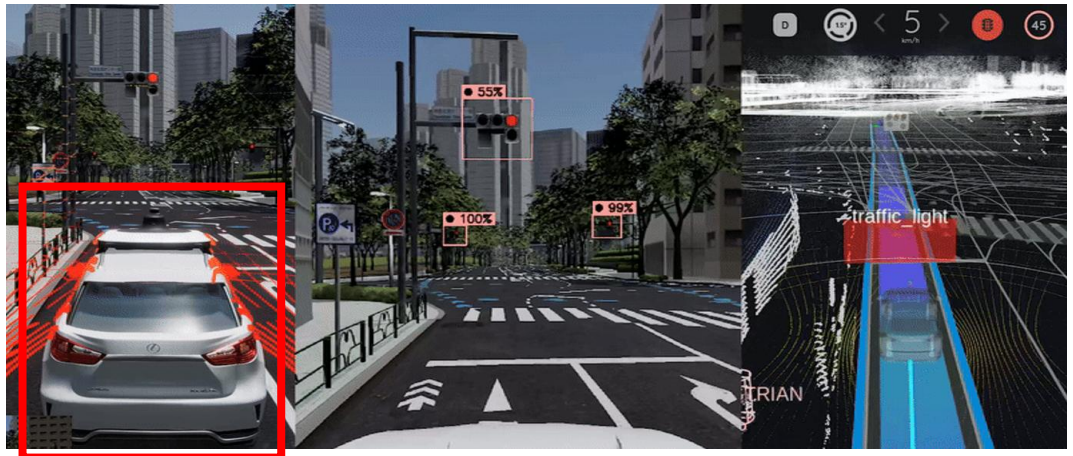
Nvidia A100

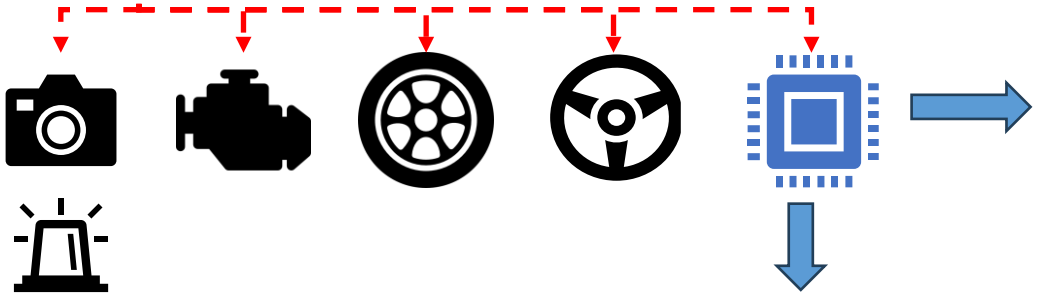
GPUs

Groq LPU

Customized
AI accelerator

Background: The Hardware Computing Platform

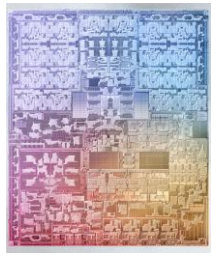




Battery powered

Space limited

Apple M4



CPUs

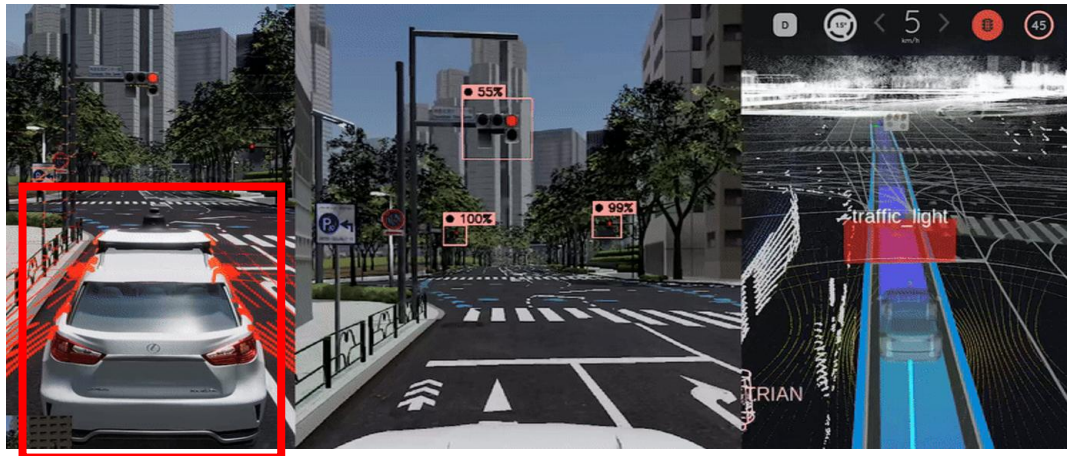
Nvidia A100

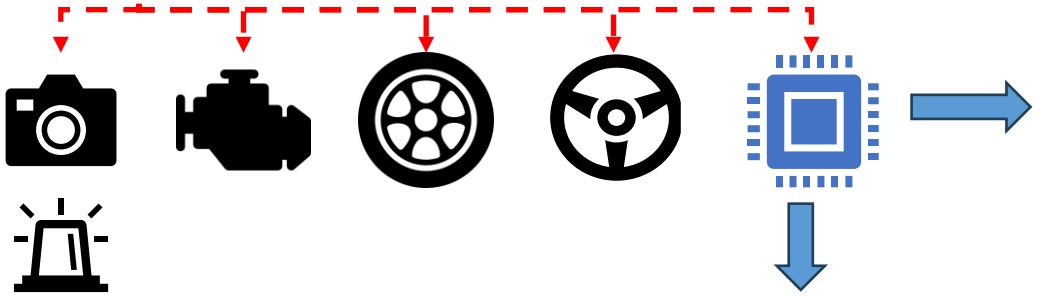
GPUs

Groq LPU

Customized
AI accelerator

Background: The Hardware Computing Platform



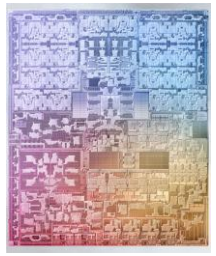


Battery powered

Space limited

Different Tasks → same accelerator

Apple M4



CPUs

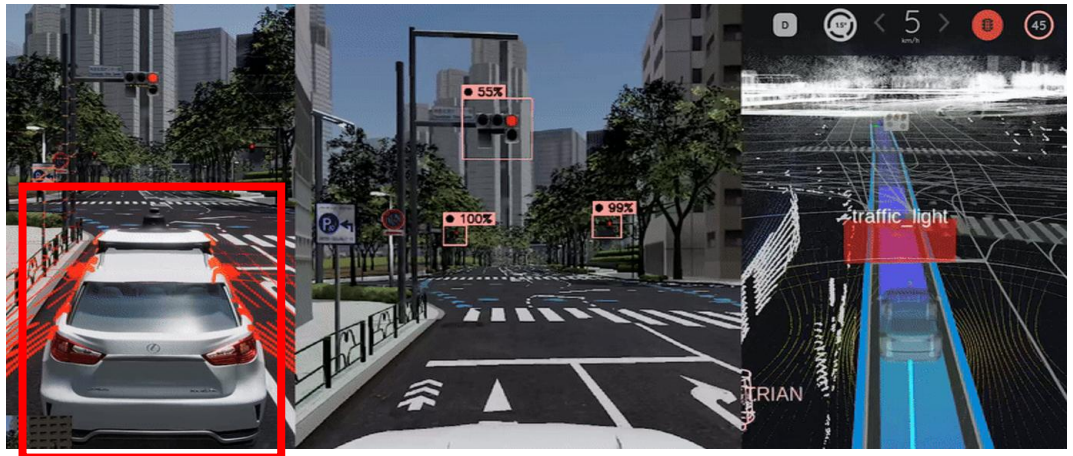
Nvidia A100

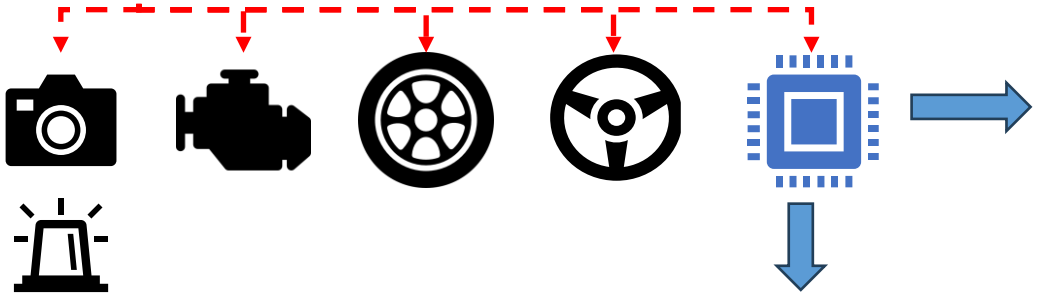
GPUs

Groq LPU

Customized
AI accelerator

Background: The Hardware Computing Platform

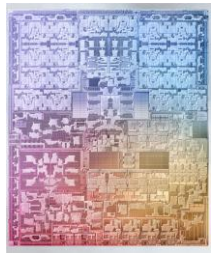




Battery powered

Space limited

Apple M4



CPUs

Nvidia A100

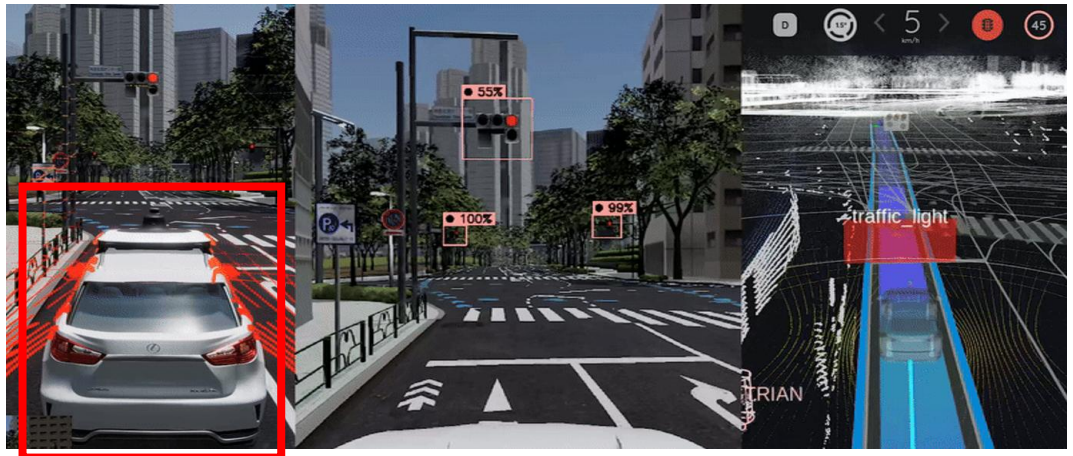
GPUs

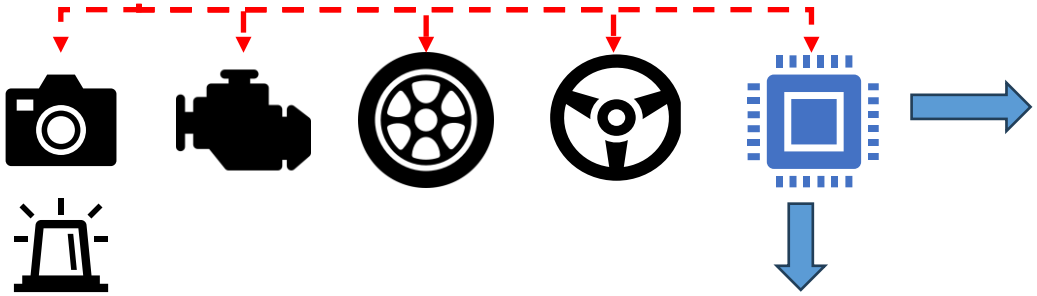
Groq LPU

Customized
AI accelerator

Different Tasks → same accelerator

Background: The Hardware Computing Platform

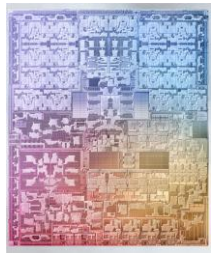




Battery powered

Space limited

Apple M4



CPUs

Nvidia A100

GPUs

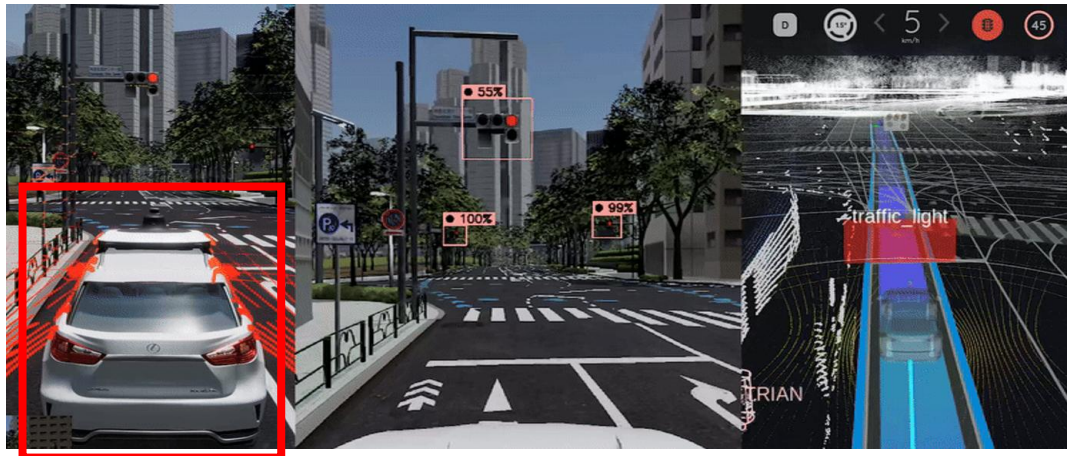
Groq LPU

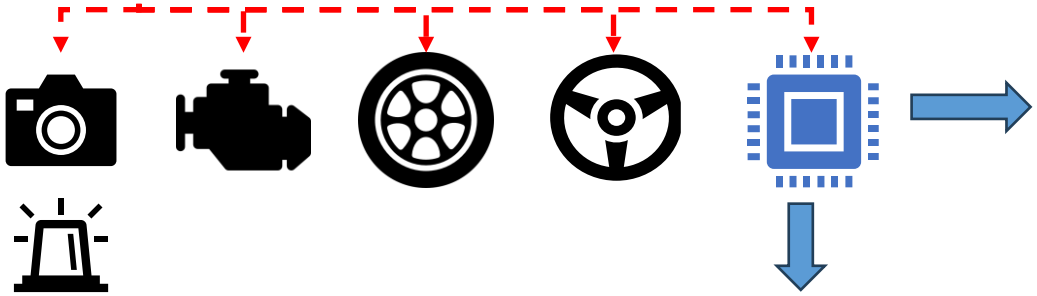
Customized
AI accelerator

Different Tasks → same accelerator

Dynamic Scheduling + Preemption

Background: The Hardware Computing Platform





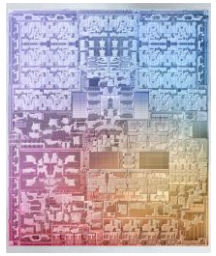
Battery powered

Space limited

Different Tasks → same accelerator

Dynamic Scheduling + Preemption

Apple M4



CPUs

Nvidia A100

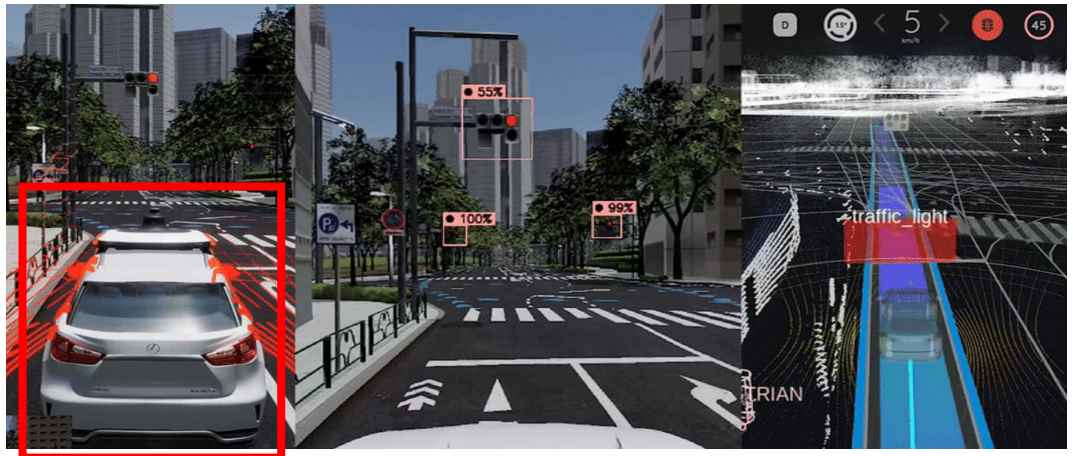
GPUs

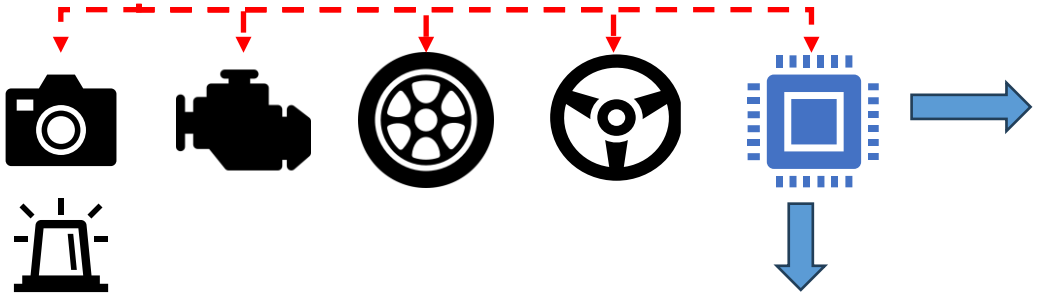
Groq LPU

Customized AI accelerator

? **What architectural features are needed**

Background: The Hardware Computing Platform





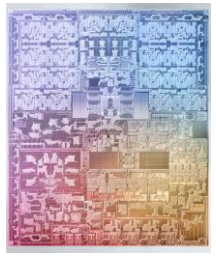
Battery powered

Space limited

Different Tasks → same accelerator

Dynamic Scheduling + Preemption

Apple M4



CPUs

Nvidia A100

GPUs

Groq LPU

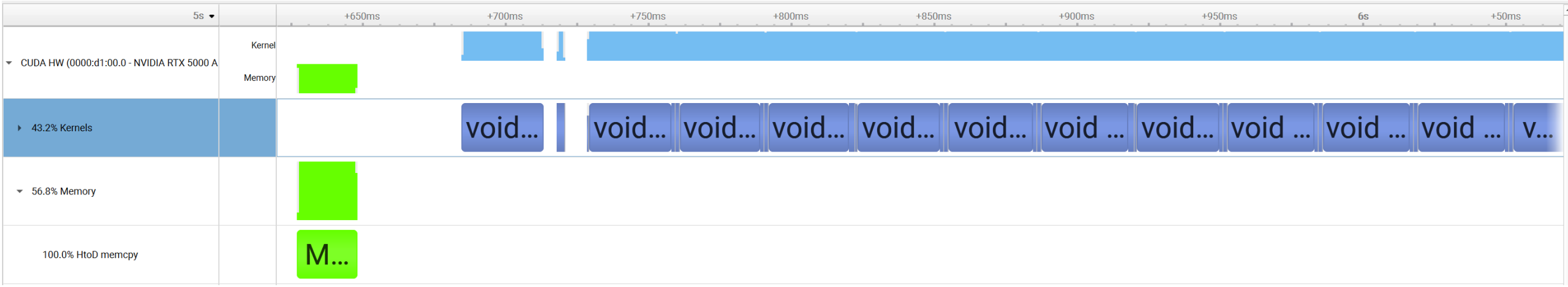
Customized AI accelerator

What architectural features are needed

How are the DNNs executed

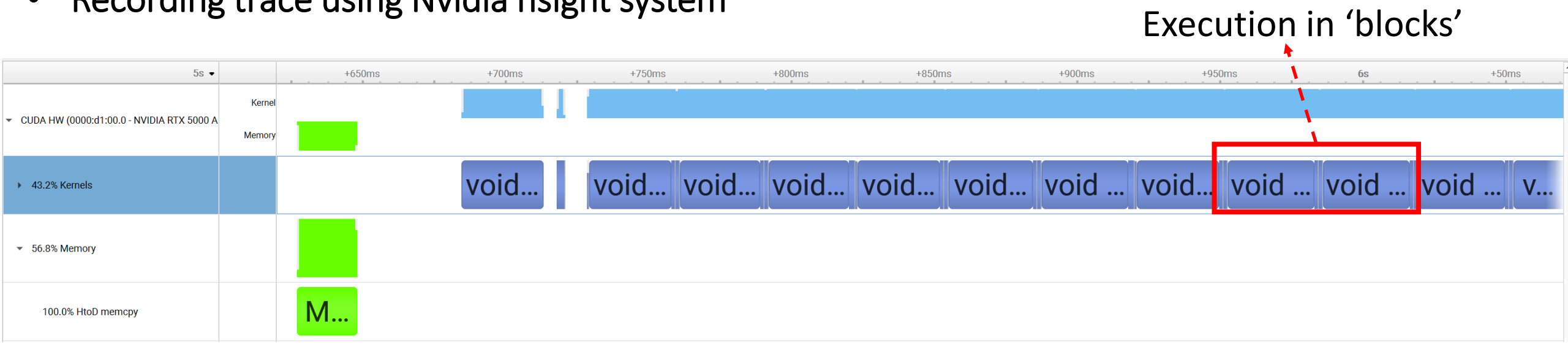
Background: Layerwise Scheduling and Execution Pattern

- Profile a MLP model on GPU using Pytorch
- Recording trace using Nvidia nsight system



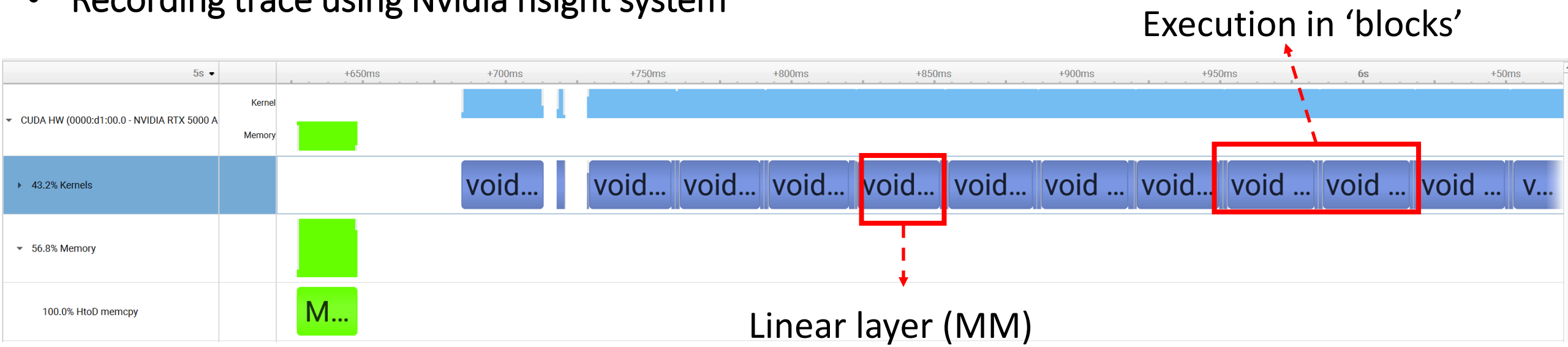
Background: Layerwise Scheduling and Execution Pattern

- Profile a MLP model on GPU using Pytorch
- Recording trace using Nvidia nsight system



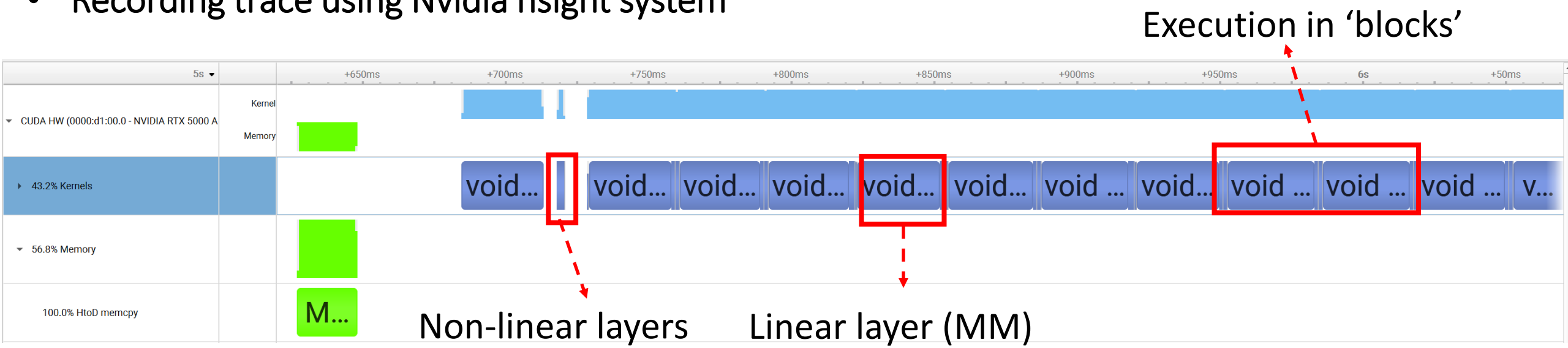
Background: Layerwise Scheduling and Execution Pattern

- Profile a MLP model on GPU using Pytorch
- Recording trace using Nvidia nsight system



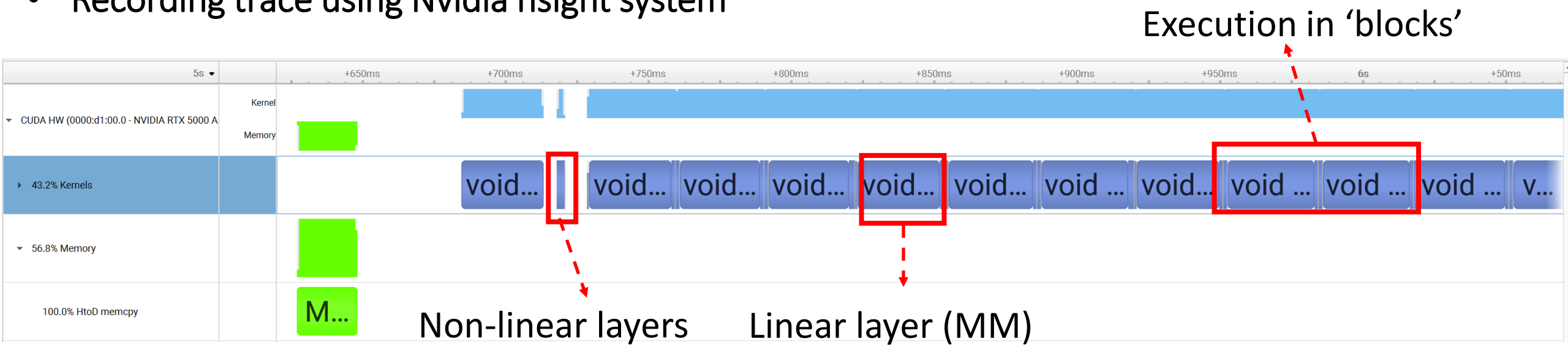
Background: Layerwise Scheduling and Execution Pattern

- Profile a MLP model on GPU using Pytorch
- Recording trace using Nvidia nsight system



Background: Layerwise Scheduling and Execution Pattern

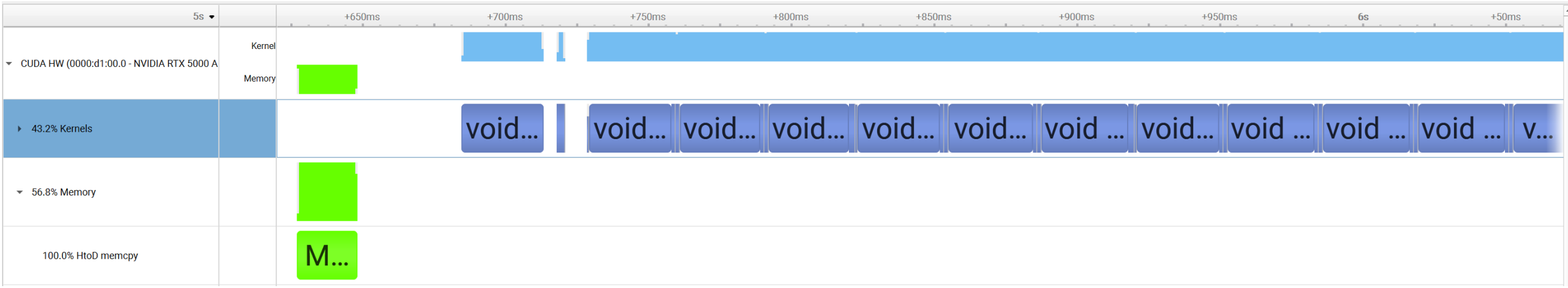
- Profile a MLP model on GPU using Pytorch
- Recording trace using Nvidia nsight system



- DNN models are composed of DNN layers (CNN, MM, non-linear,...)
- GPU schedule and execute the layers one-by-one in the form of kernels

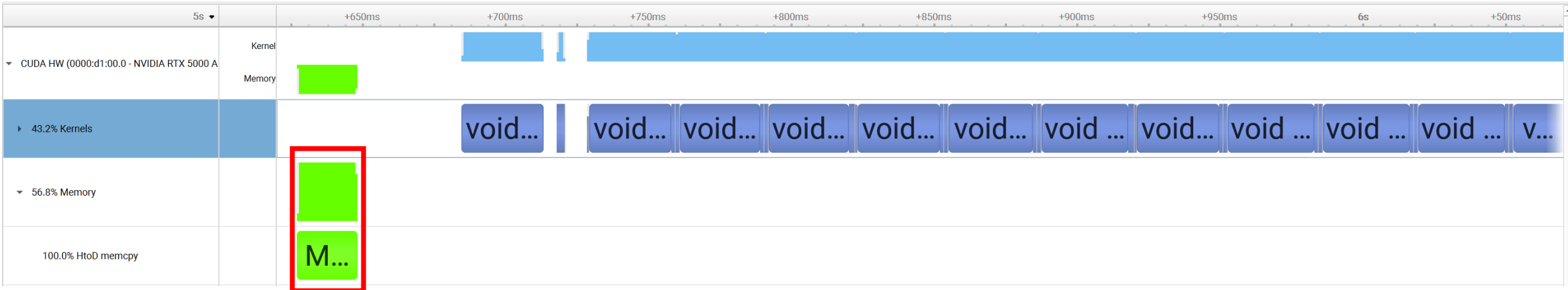
Background: Layerwise Scheduling and Execution Pattern

- W/t CPU scheduler, can preempt between two layers



Background: Layerwise Scheduling and Execution Pattern

- W/t CPU scheduler, can preempt between two layers

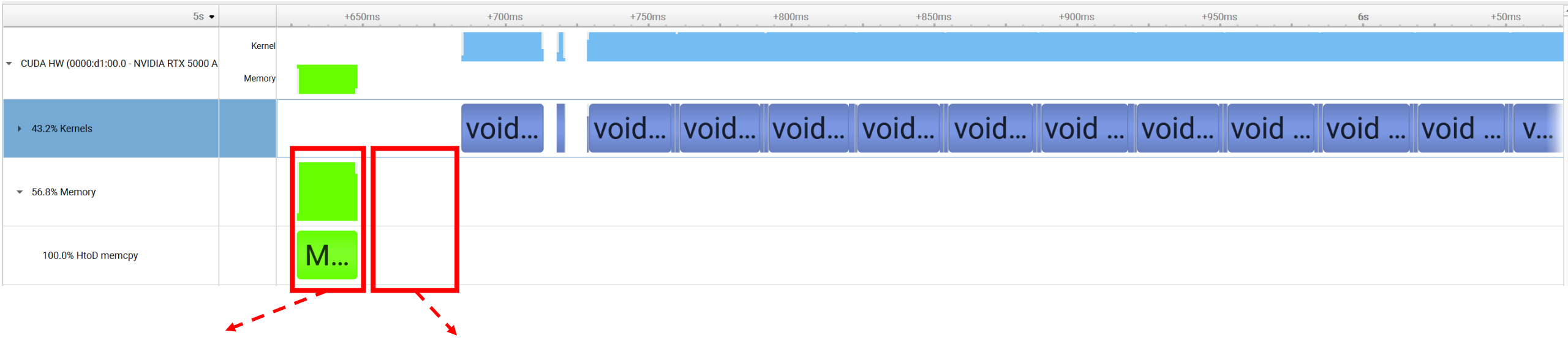


HtoD memcpy

20.4 ms

Background: Layerwise Scheduling and Execution Pattern

- W/t CPU scheduler, can preempt between two layers



HtoD memcpy

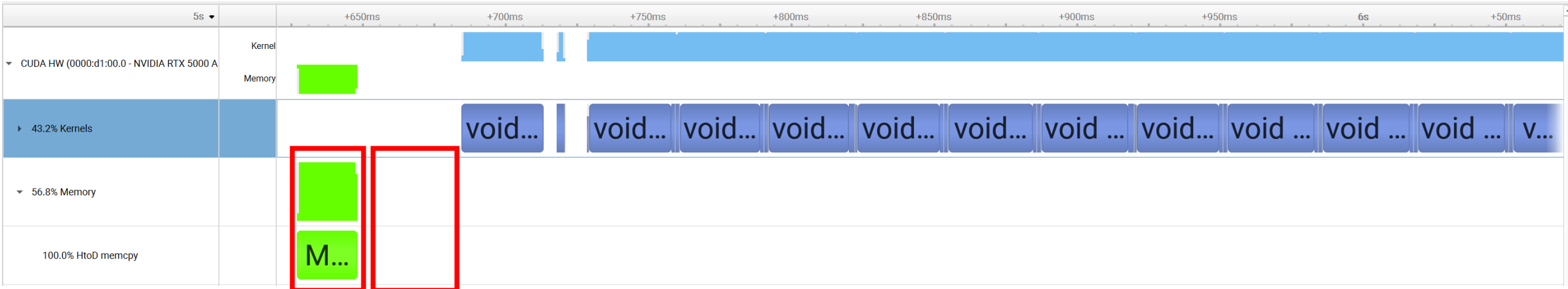
Other overhead

20.4 ms

31.3 ms

Background: Layerwise Scheduling and Execution Pattern

- W/t CPU scheduler, can preempt between two layers



HtoD memcpy

Other overhead

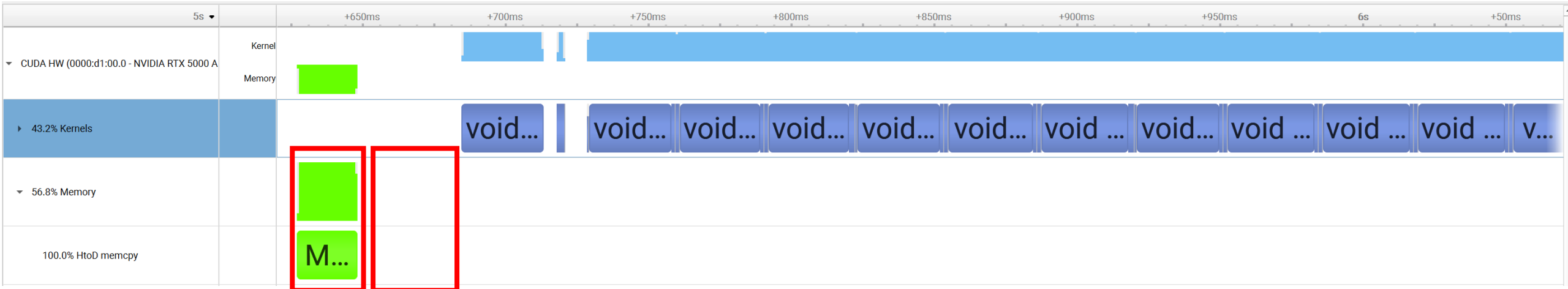
20.4 ms

31.3 ms

- the overhead in preemption could also be an issue

Background: Layerwise Scheduling and Execution Pattern

- W/t CPU scheduler, can preempt between two layers



HtoD memcpy

Other overhead

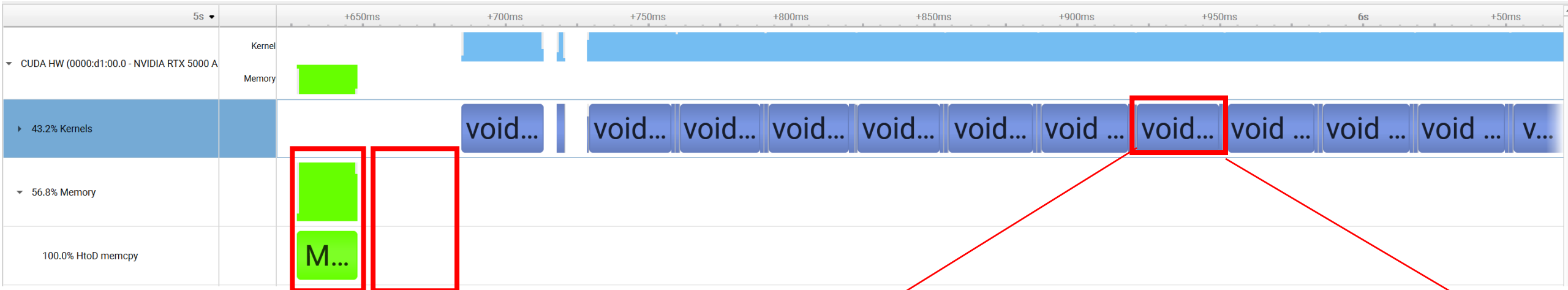
20.4 ms

31.3 ms

- the overhead in preemption could also be an issue ? Can we schedule on accelerator

Background: Layerwise Scheduling and Execution Pattern

- W/t CPU scheduler, can preempt between two layers

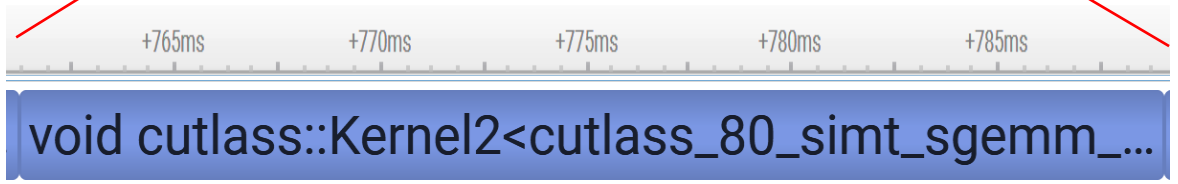


HtoD memcpy

Other overhead

20.4 ms

31.3 ms



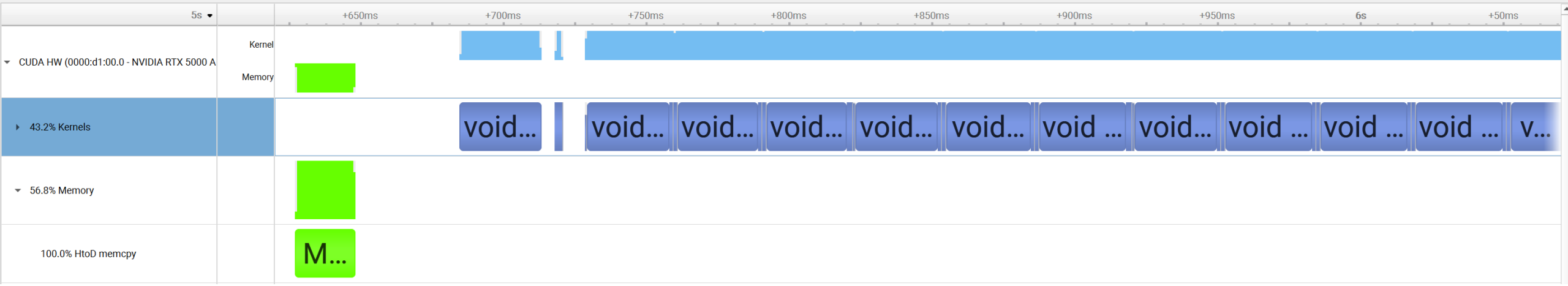
27.8 ms

- the overhead in preemption could also be an issue
- If cannot preempt within a layer → long block time

? Can we schedule on accelerator

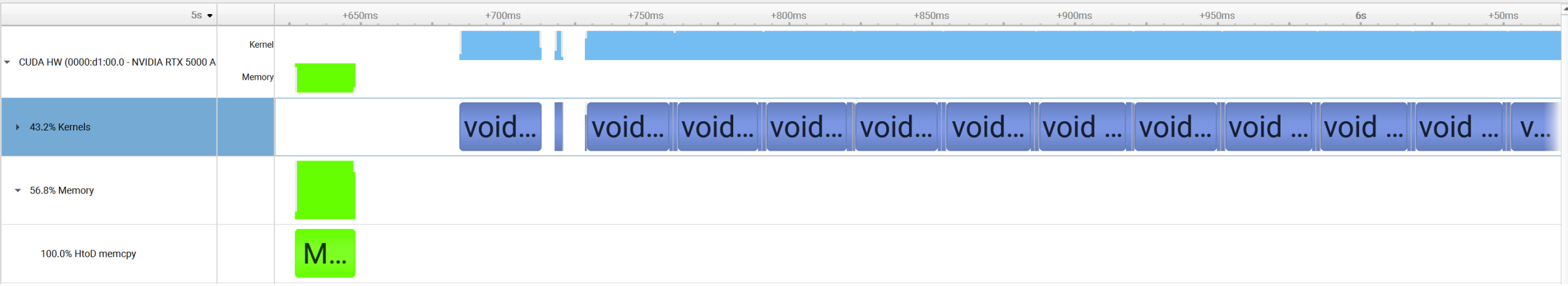
Background: Layerwise Scheduling and Execution Pattern

- W/t changing low-level Software/hardware, may preempt within a layer



Background: Layerwise Scheduling and Execution Pattern

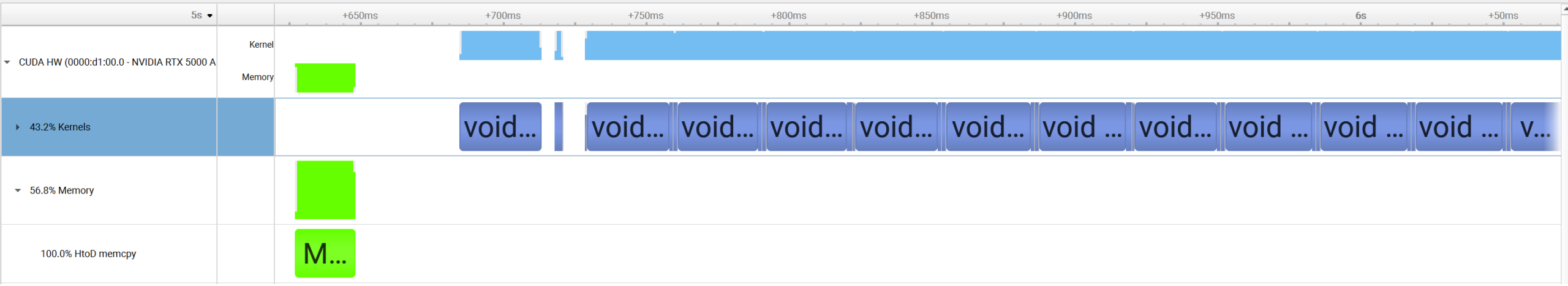
- W/t changing low-level Software/hardware, may preempt within a layer



- Not Native support --> may lack flexibility

Background: Layerwise Scheduling and Execution Pattern

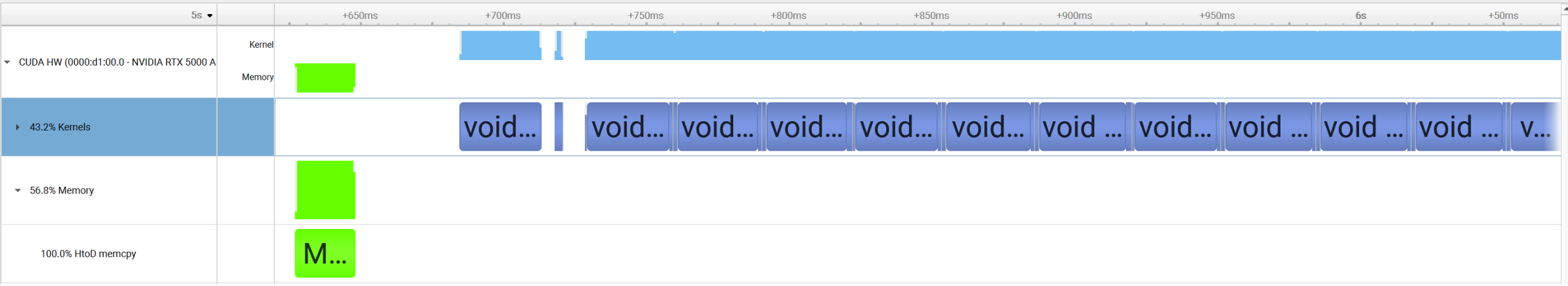
- W/t changing low-level Software/hardware, may preempt within a layer



- Not Native support --> may lack flexibility
- Low-level control API not provided

Background: Layerwise Scheduling and Execution Pattern

- W/t changing low-level Software/hardware, may preempt within a layer



- Not Native support --> may lack flexibility
- Low-level control API not provided
- Large programming effort

Summary: The Need For Real-time Safety-Critical Systems

- Reduce scheduling overhead

- Preempt within a layer

Summary: The Need For Real-time Safety-Critical Systems

- Reduce scheduling overhead → **Hardware-implemented EDF scheduler on FPGA chip**
- Preempt within a layer → **Intra-layer preemptive flexible dataflow**

Summary: The Need For Real-time Safety-Critical Systems

- Reduce scheduling overhead → **Hardware-implemented EDF scheduler on FPGA chip**
- Preempt within a layer → **Intra-layer preemptive flexible dataflow**

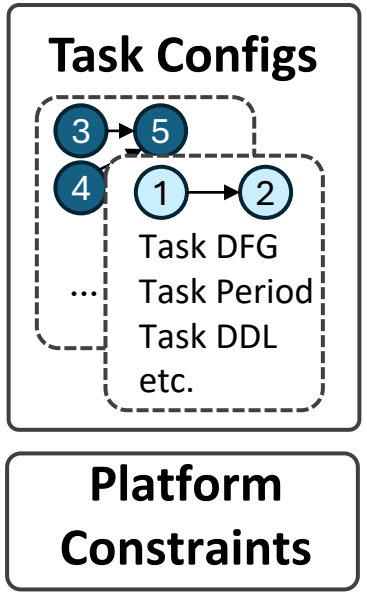
FPGA platform:
Bit-level reconfiguration
Low level hardware control

→ **Hardware support**

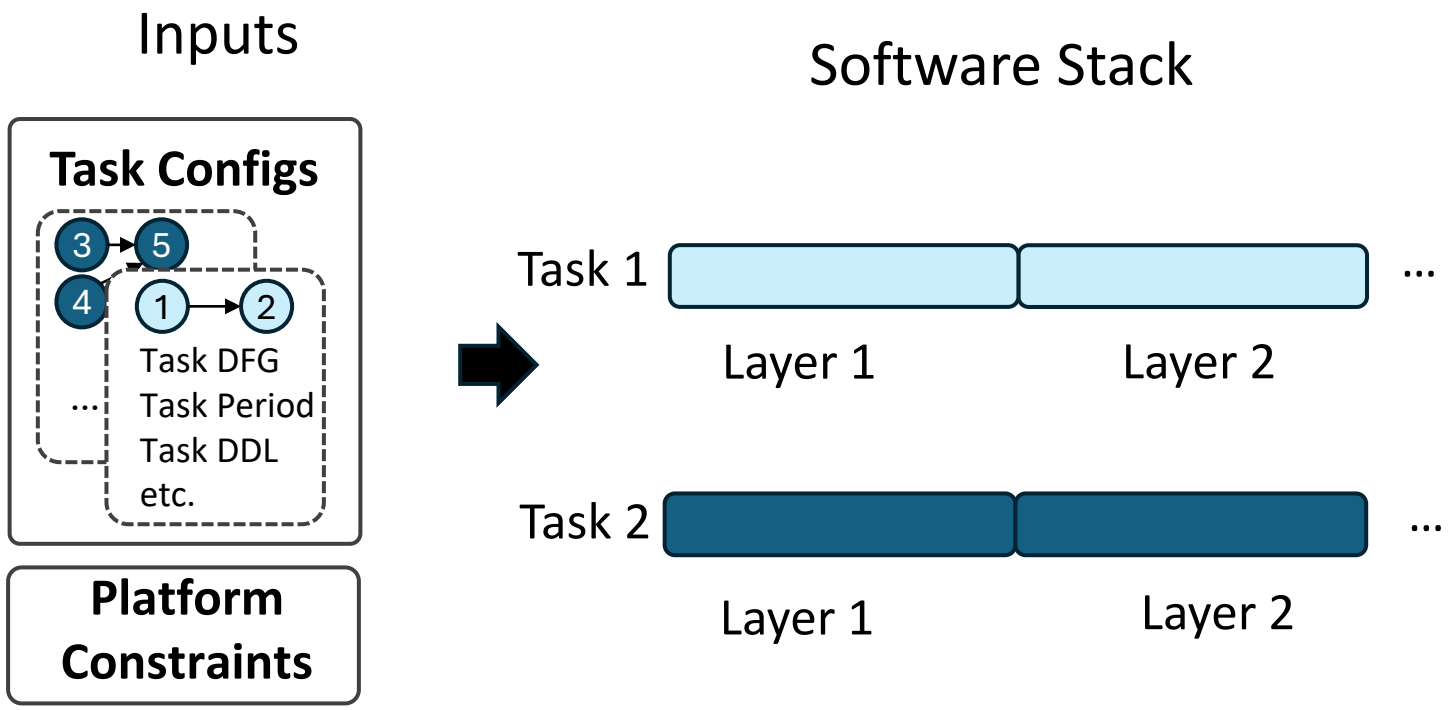
DERCA: Workflow Overview

DERCA: Workflow Overview

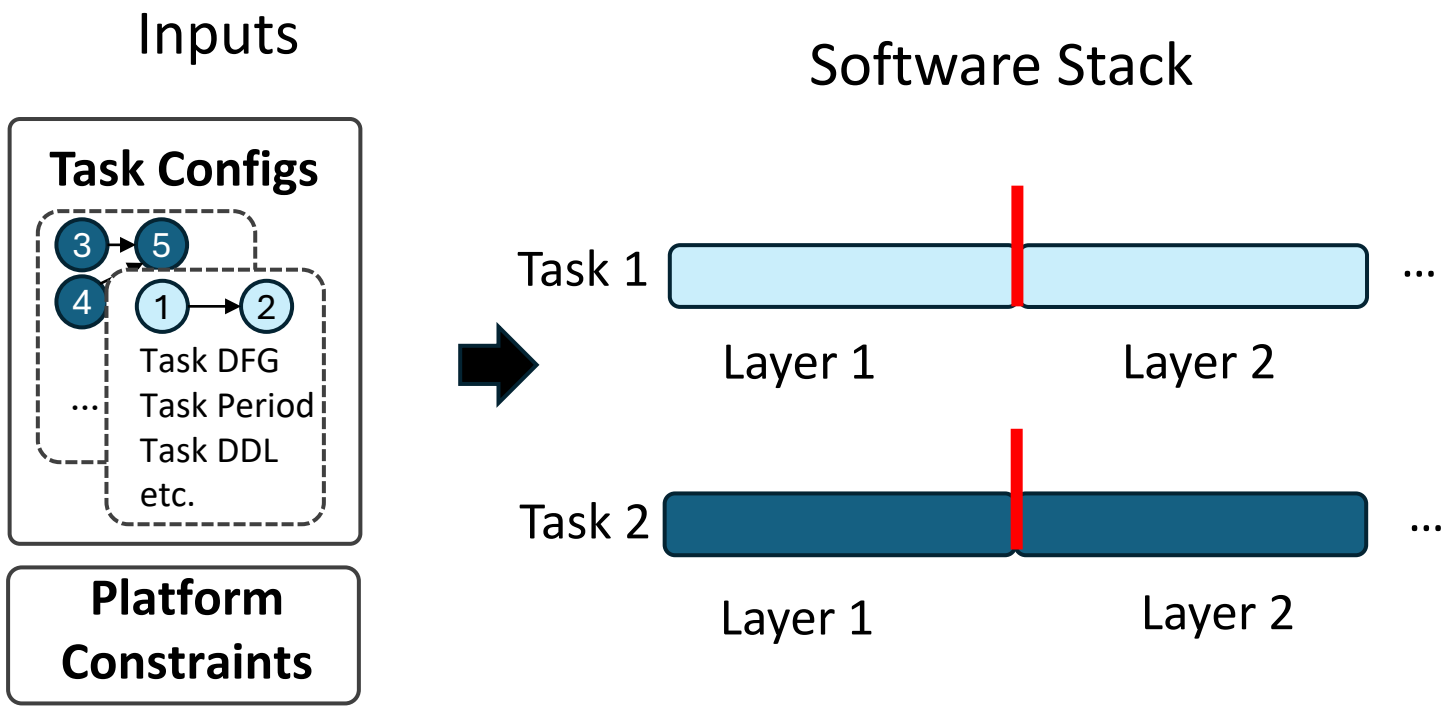
Inputs



DERCA: Workflow Overview

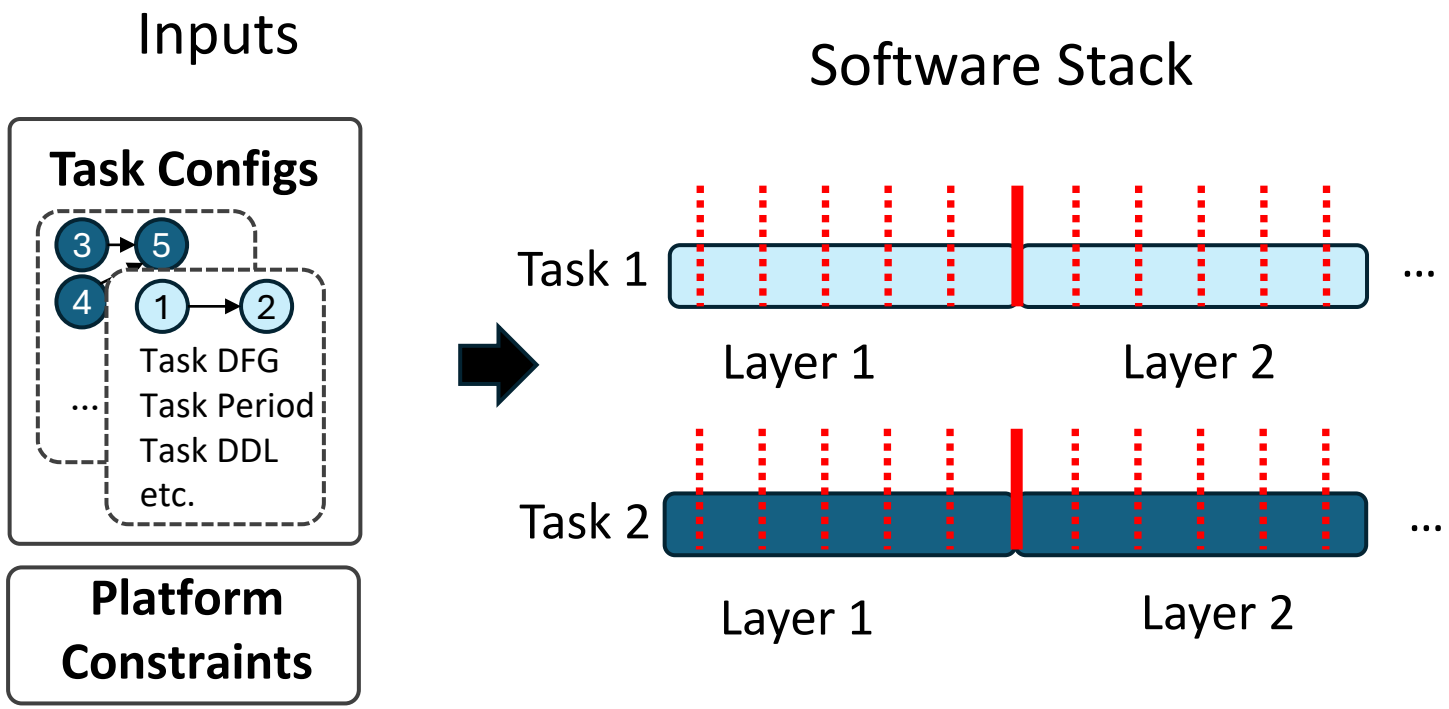


DERCA: Workflow Overview



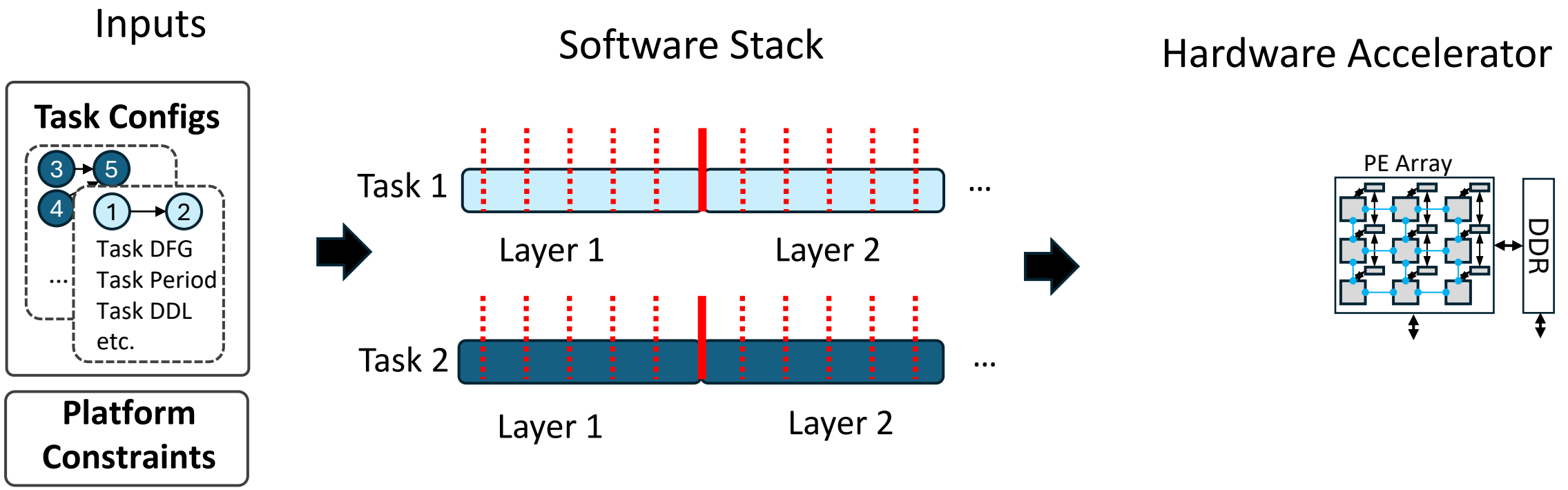
- Inter-layer preemption point (PP)

DERCA: Workflow Overview



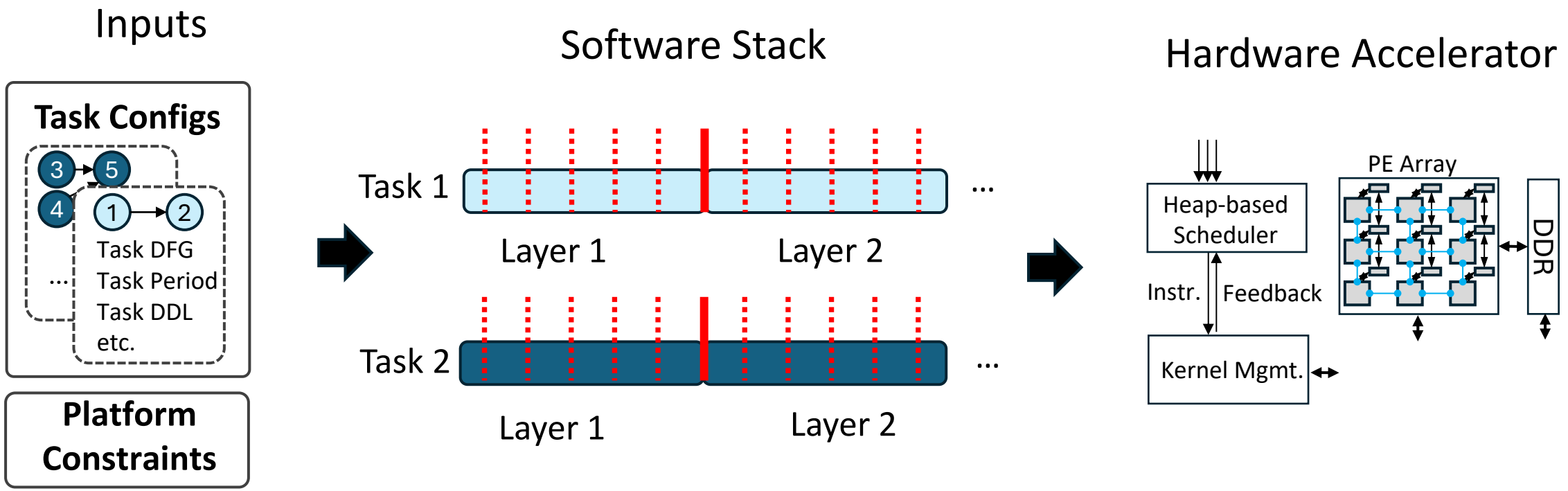
- Inter-layer preemption point (PP)
- Intra-layer PP
- HW operation → overhead optimization

DERCA: Workflow Overview



- Inter-layer preemption point (PP)
- Intra-layer PP
- HW operation → overhead optimization

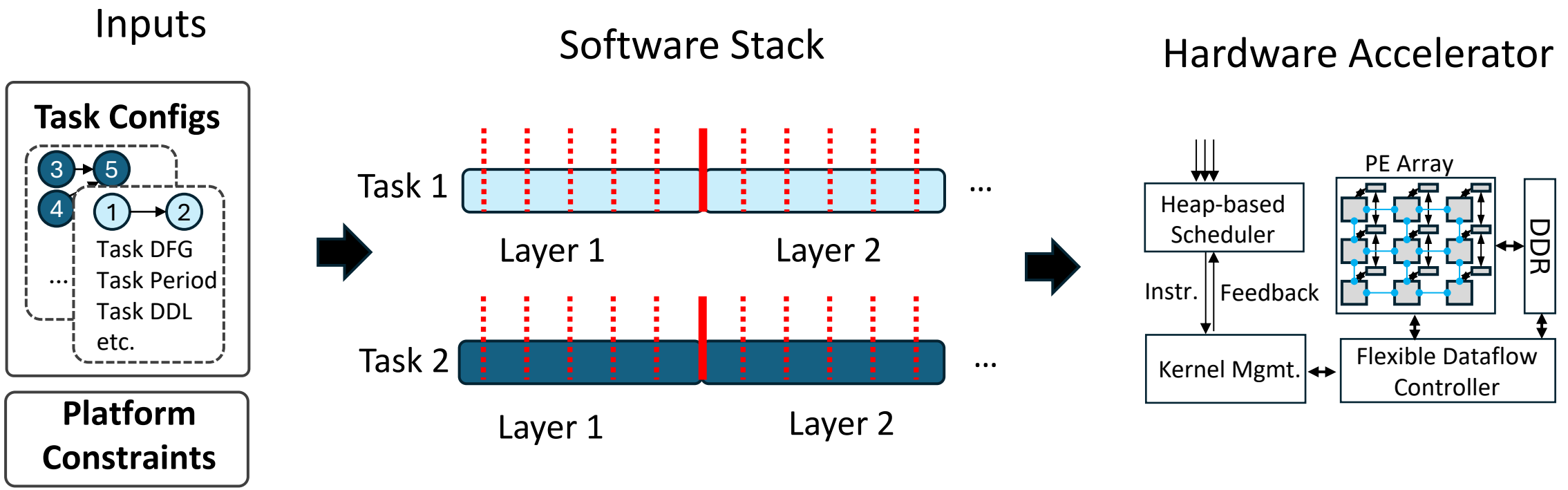
DERCA: Workflow Overview



- Inter-layer preemption point (PP)
- Intra-layer PP
- HW operation → overhead optimization

- Dynamic scheduling

DERCA: Workflow Overview

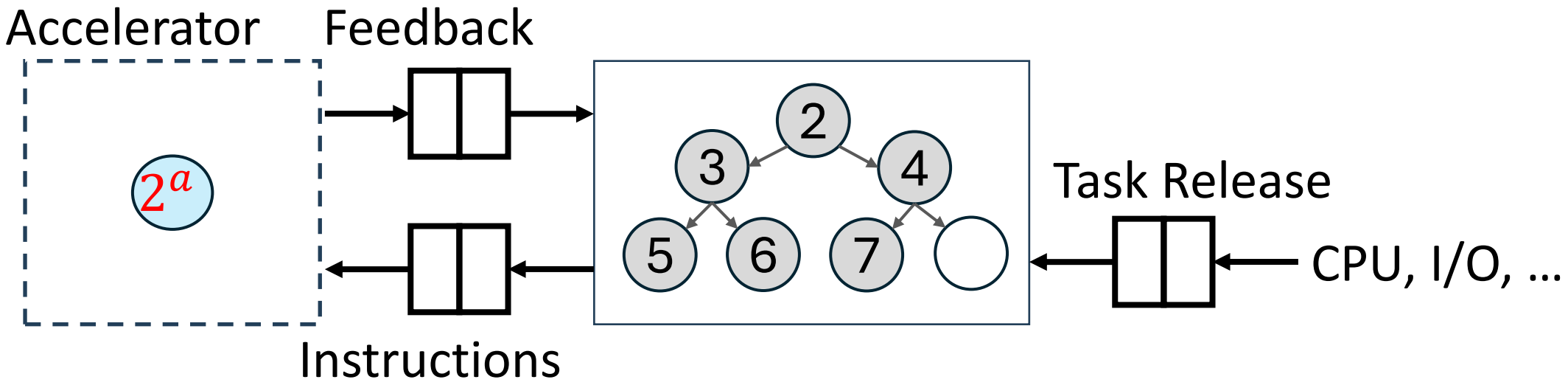


- Inter-layer preemption point (PP)
- Intra-layer PP
- HW operation → overhead optimization

- Dynamic scheduling
- Fine-grained preemption

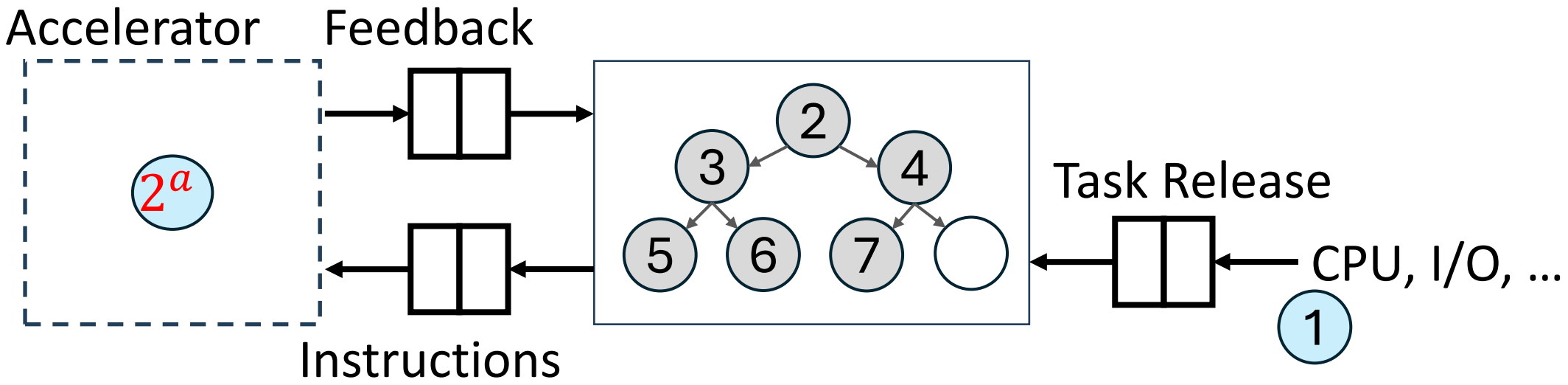
DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



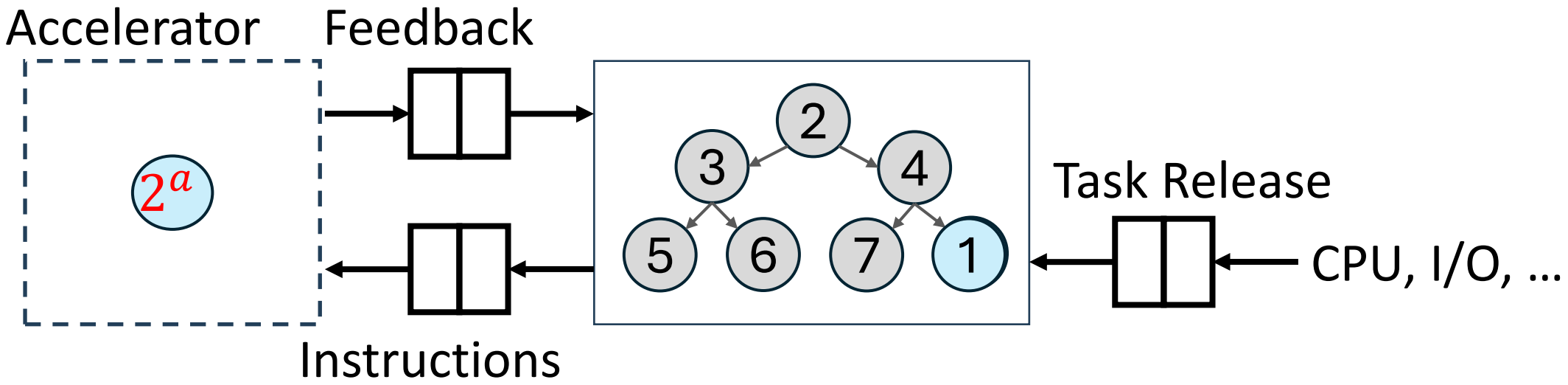
DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



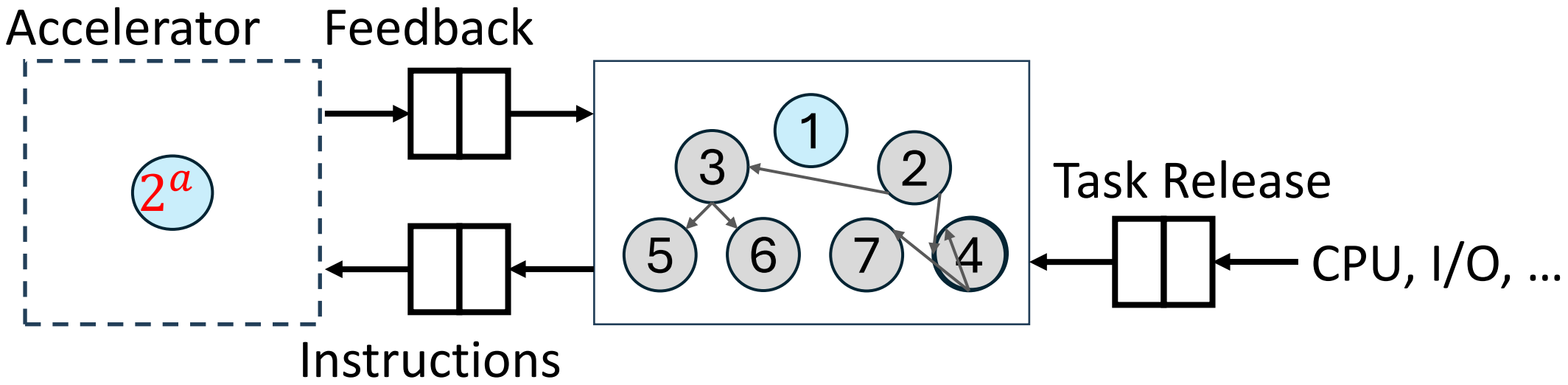
DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



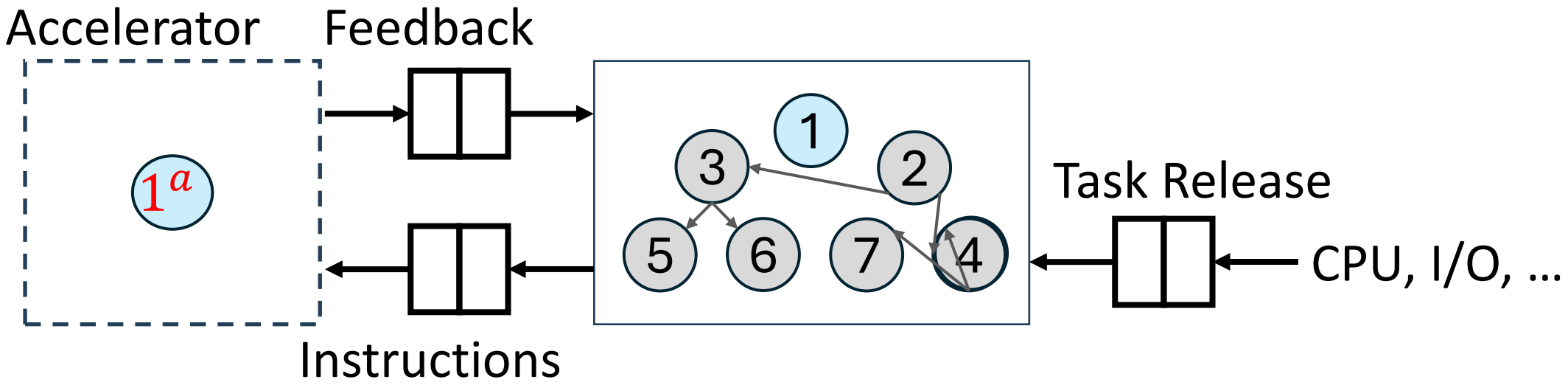
DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



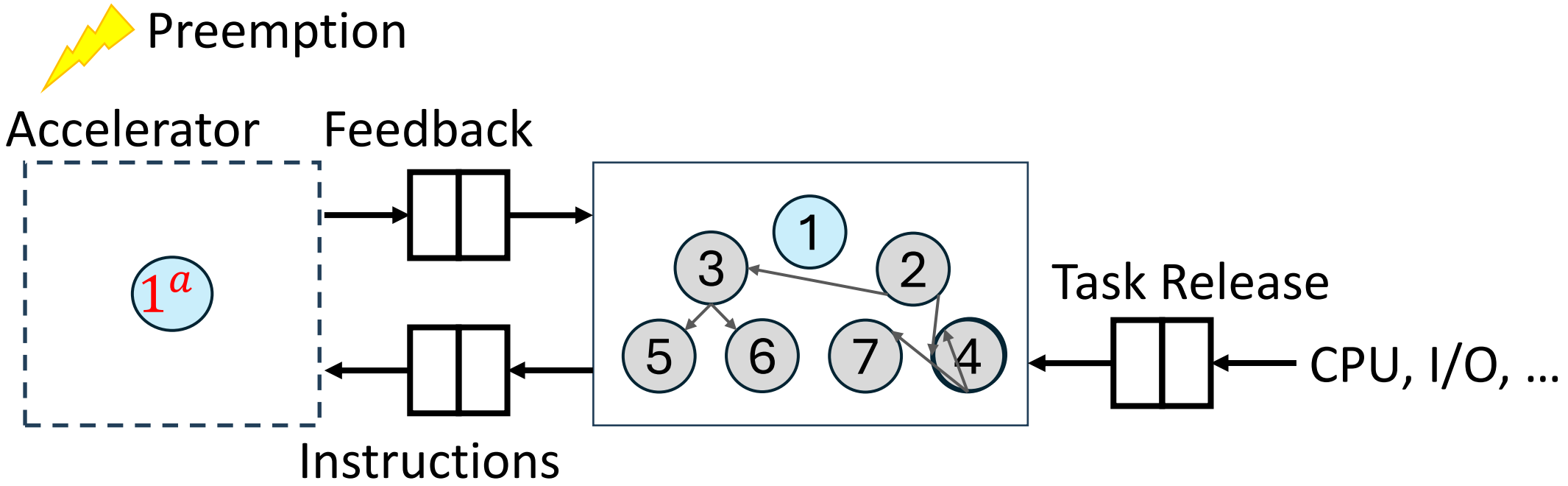
DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



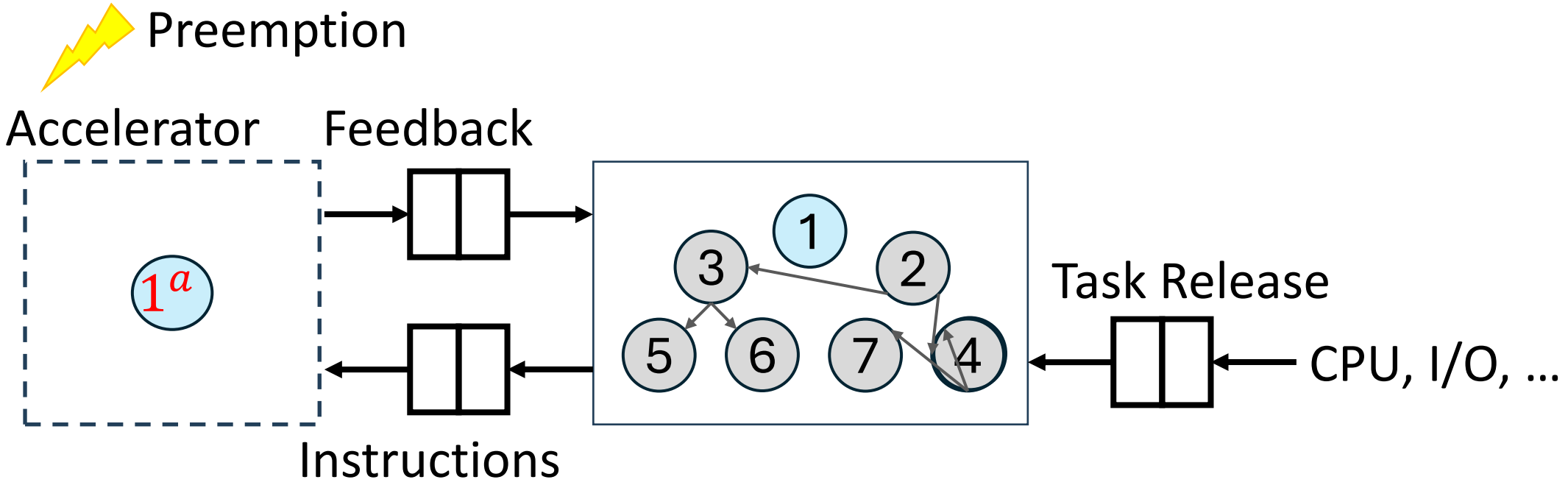
DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



DERCA: On-chip EDF Scheduler

Core: hardware-implemented heap



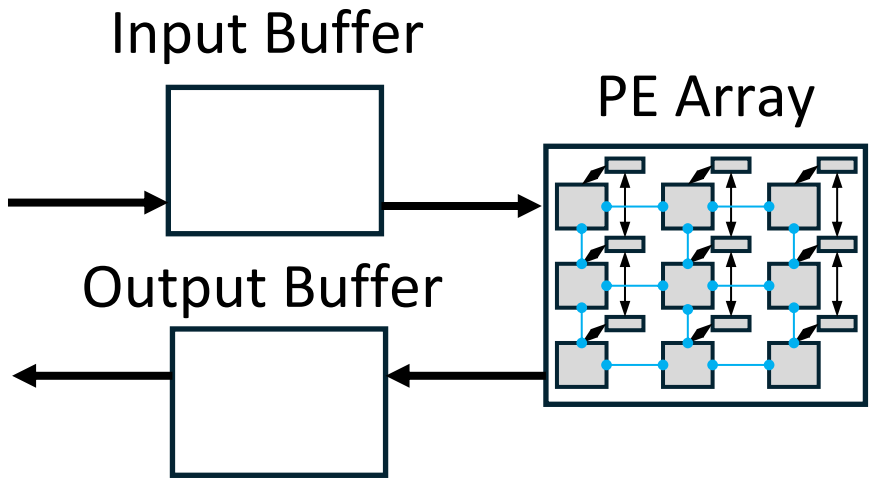
Update the queue when:

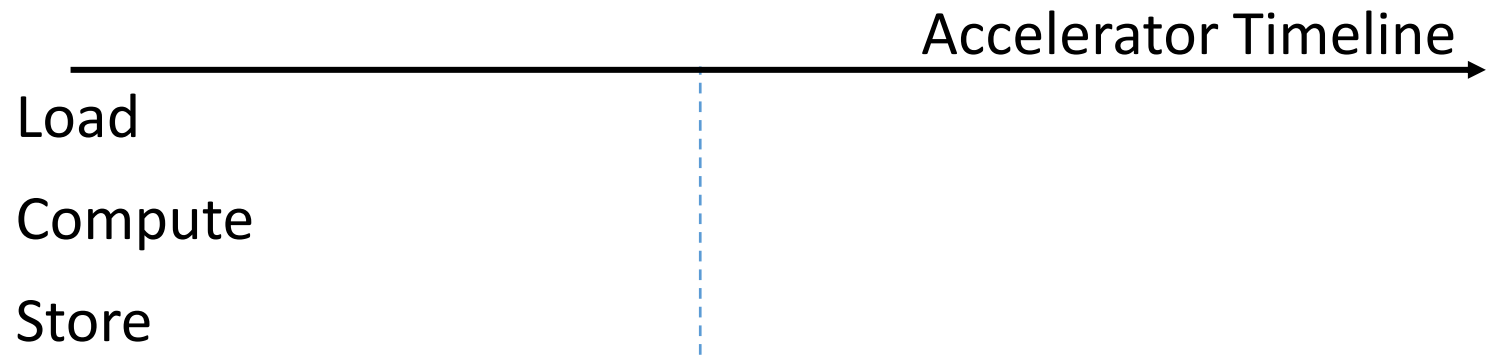
- New job release
- Finish a segment

DERCA: Intra-Layer Preemptive Accelerator Design

RHS

LHS



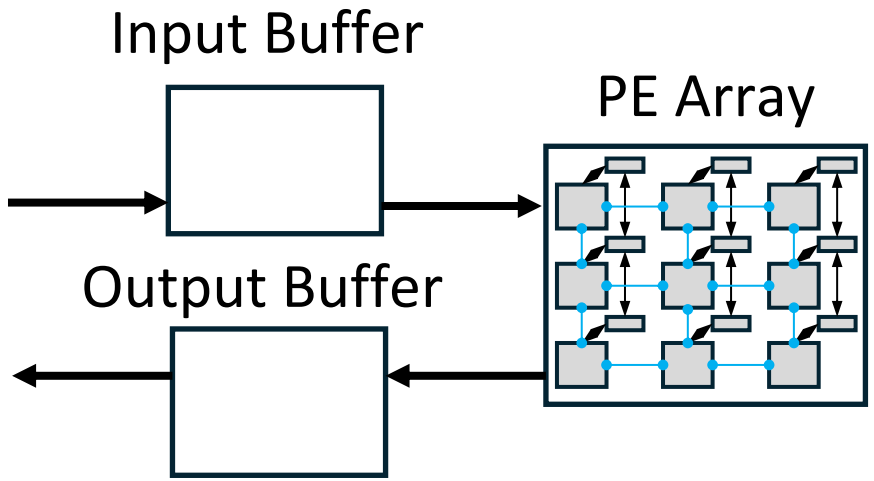


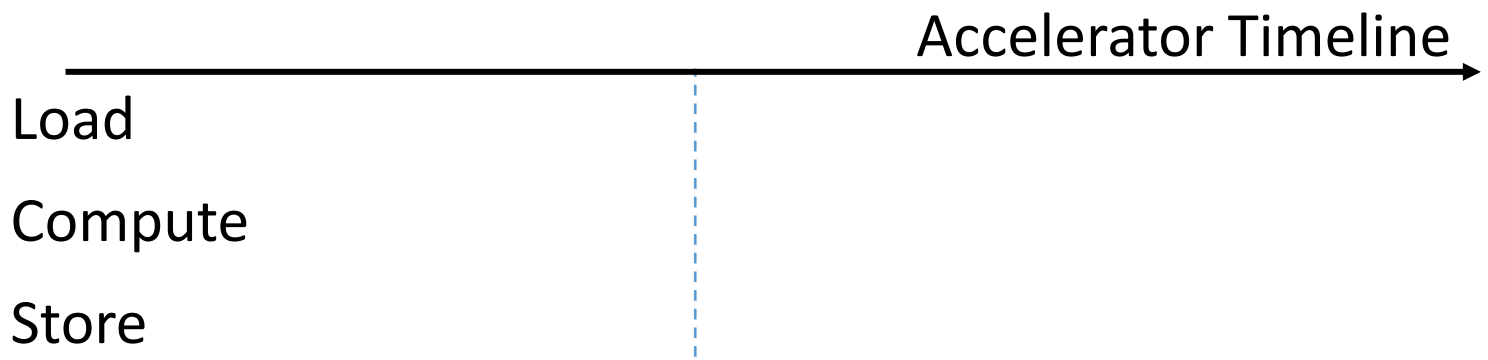
DERCA: Intra-Layer Preemptive Accelerator Design

Where should we partition the DNN layer? → execution procedures

RHS

LHS





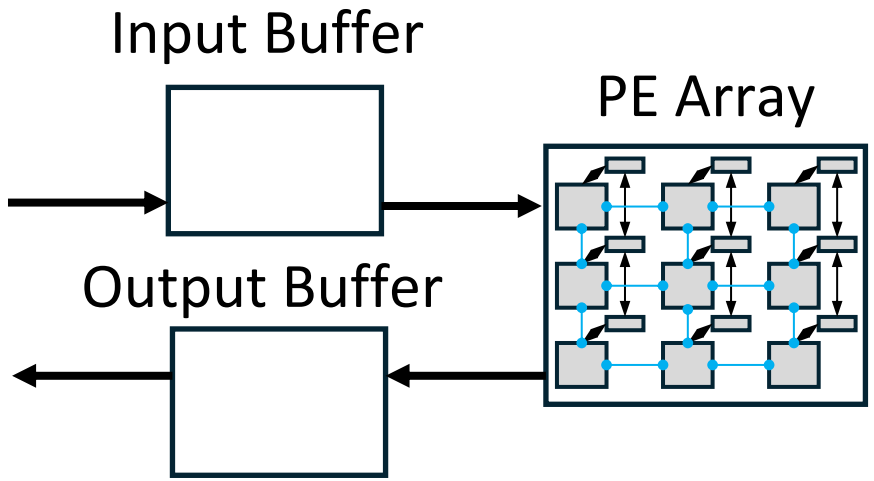
DERCA: Intra-Layer Preemptive Accelerator Design

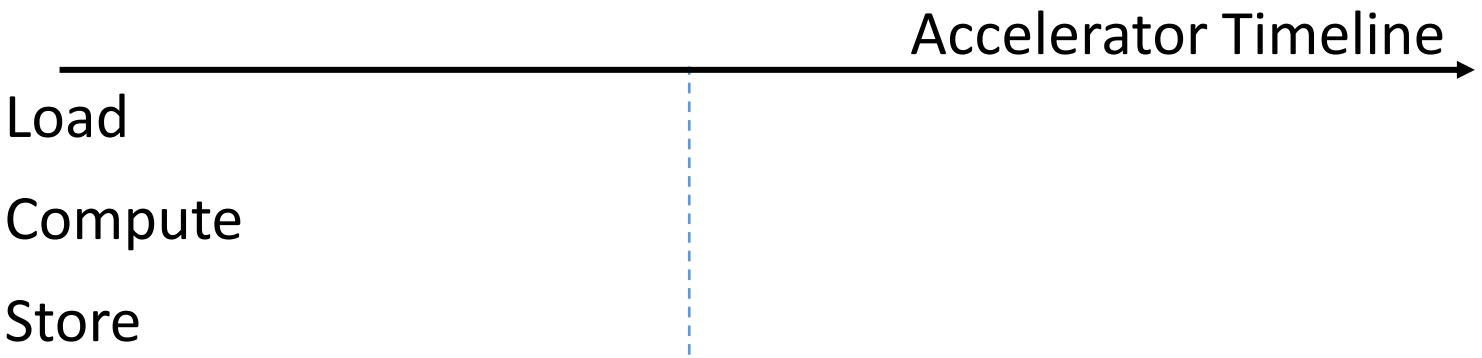
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs

RHS

LHS

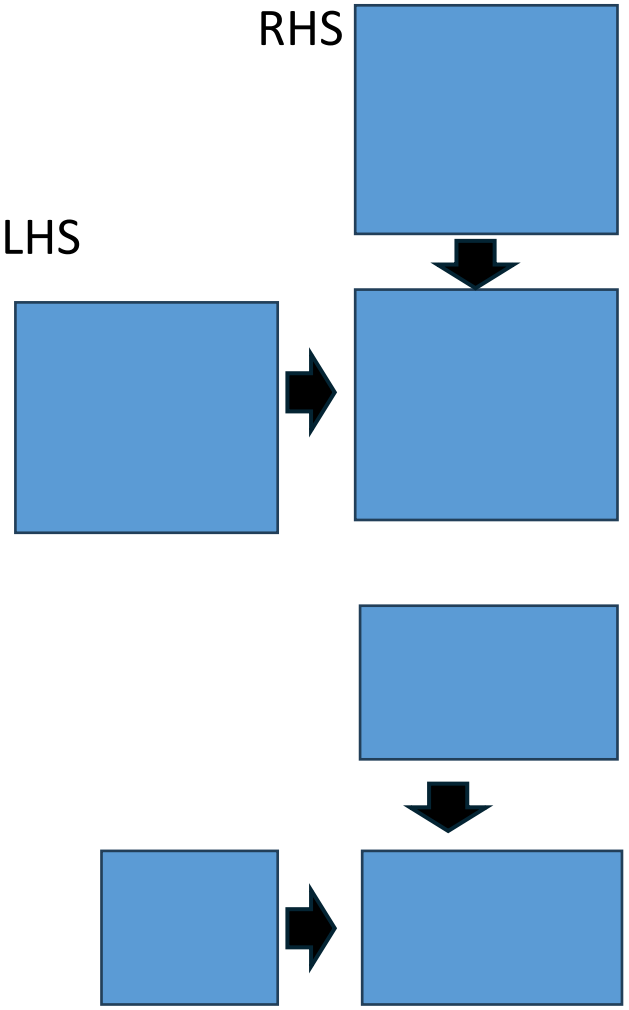


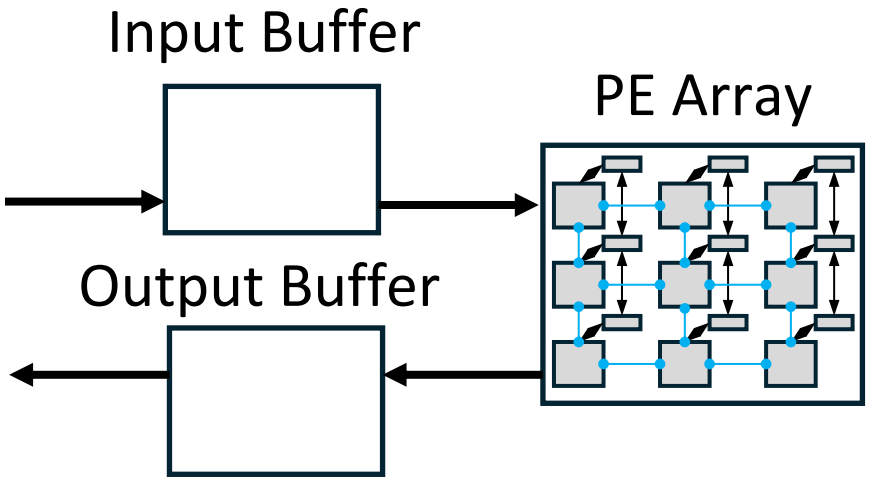


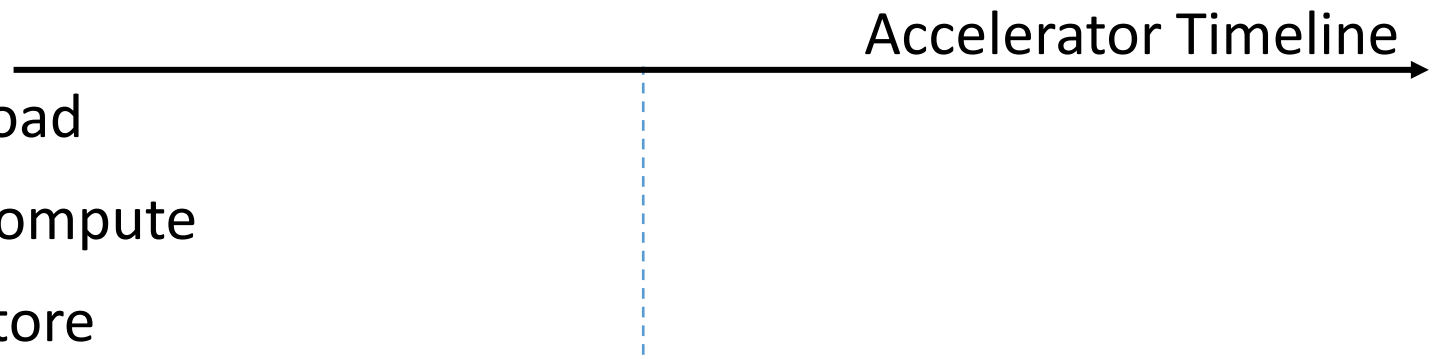
DERCA: Intra-Layer Preemptive Accelerator Design

Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



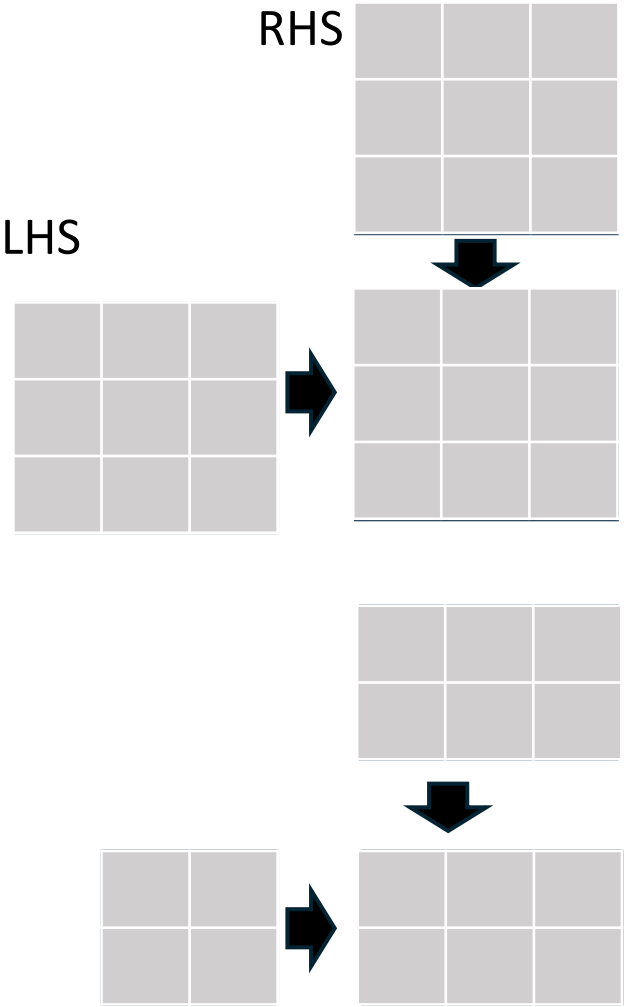




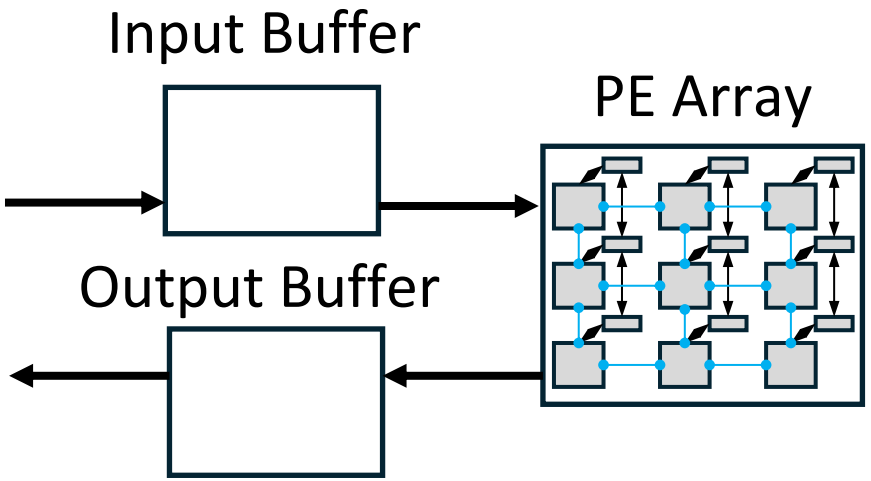
DERCA: Intra-Layer Preemptive Accelerator Design

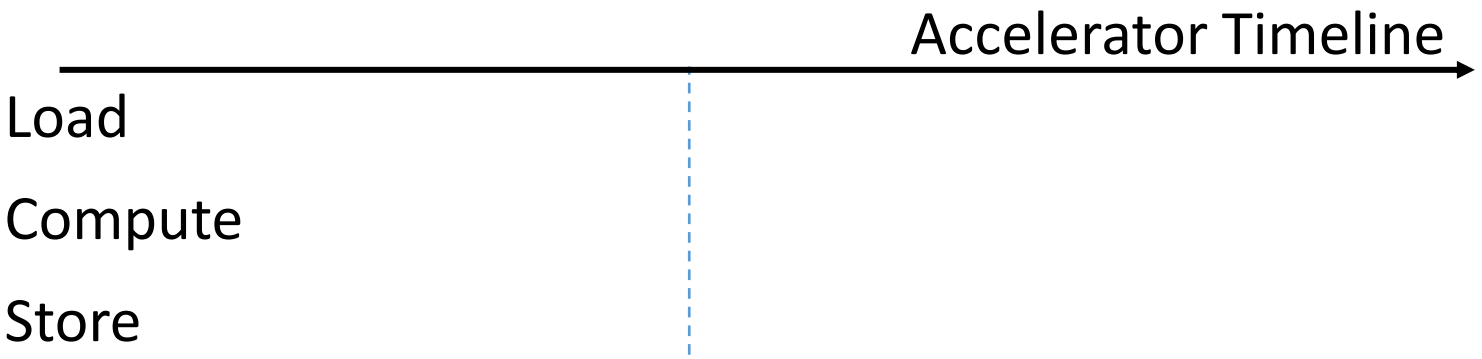
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

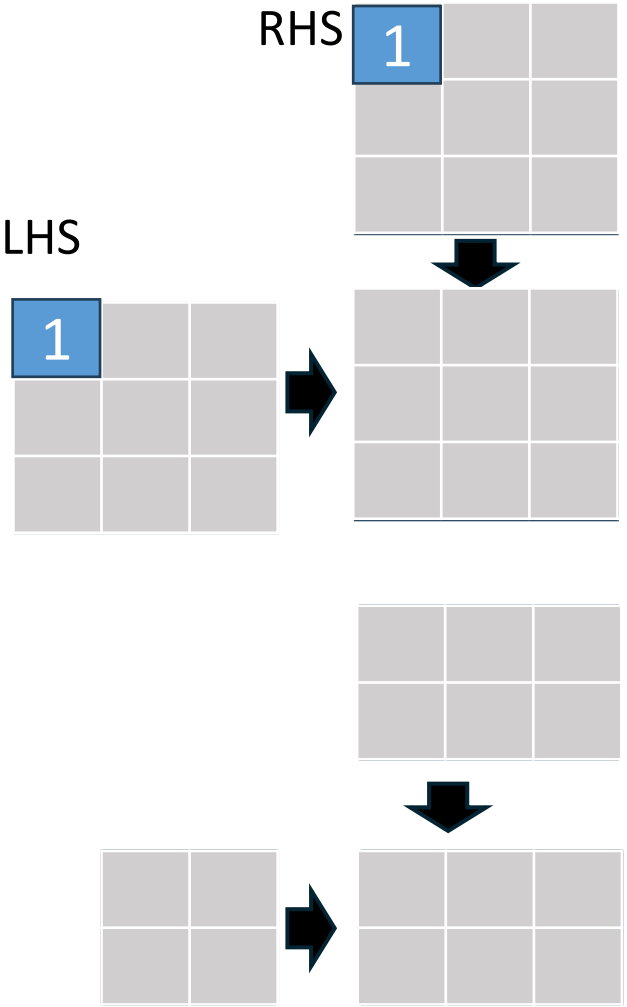




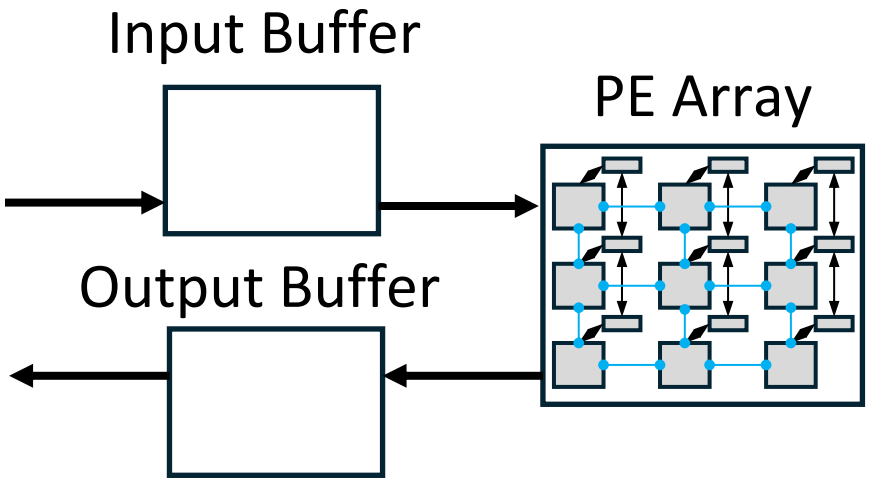
DERCA: Intra-Layer Preemptive Accelerator Design

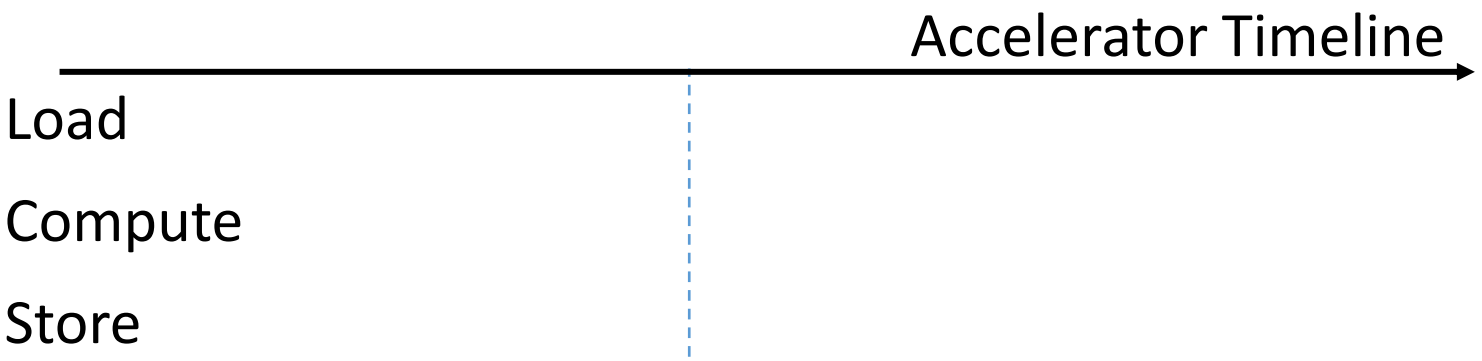
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

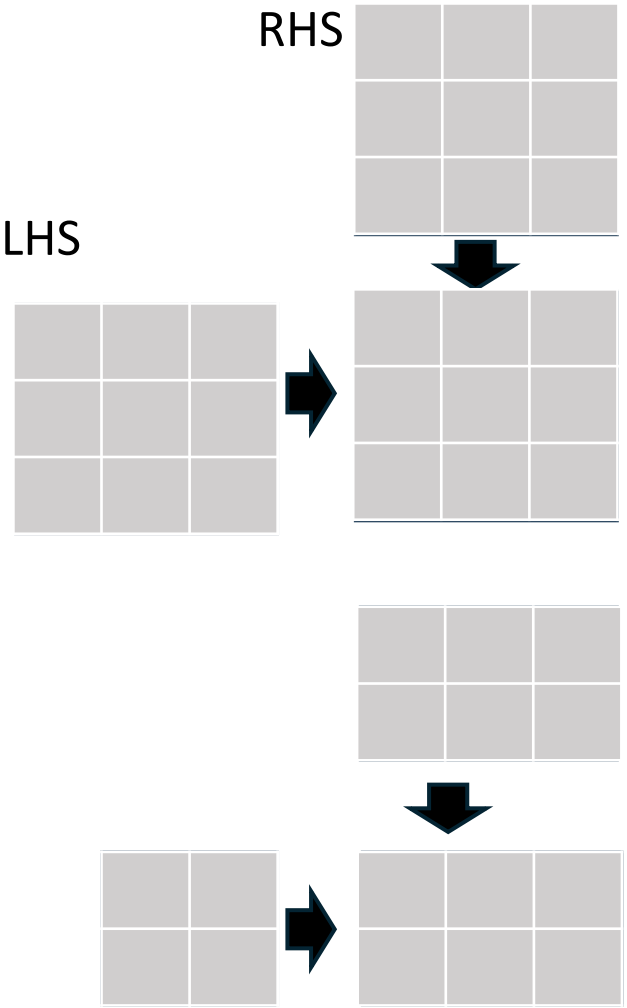




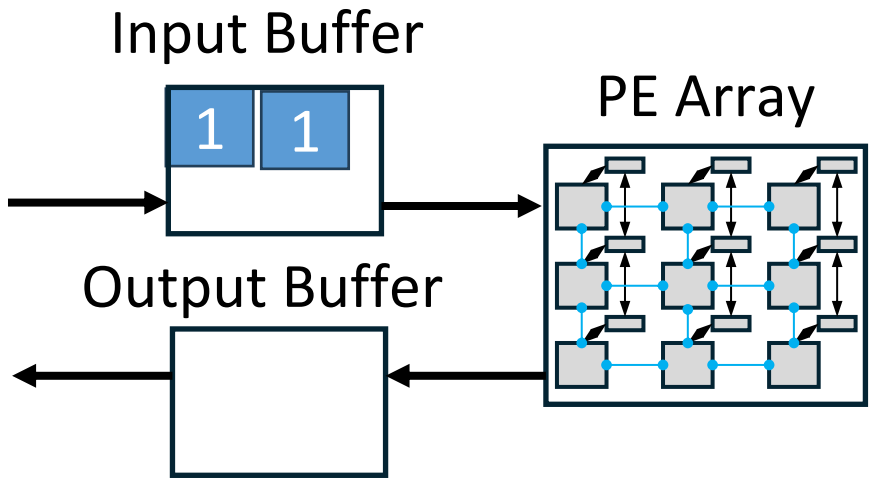
DERCA: Intra-Layer Preemptive Accelerator Design

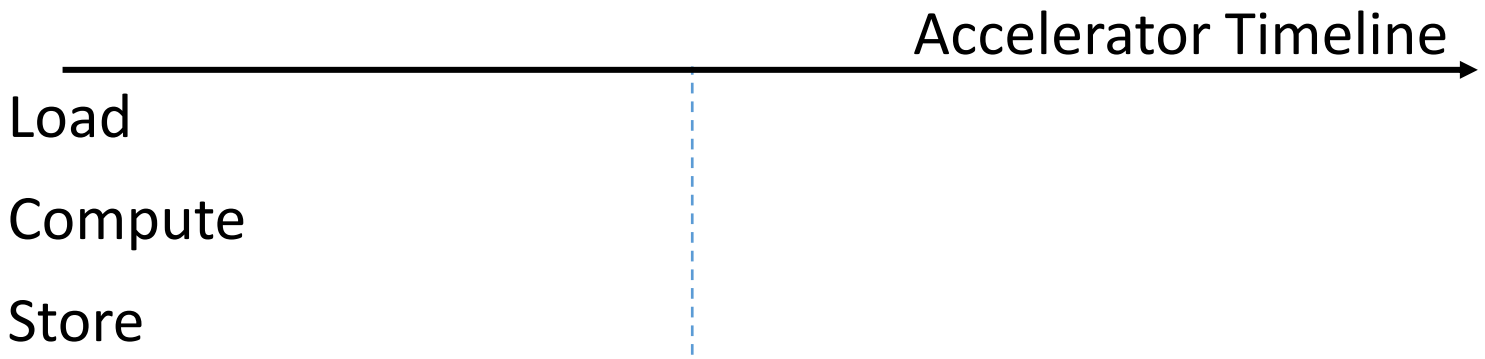
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

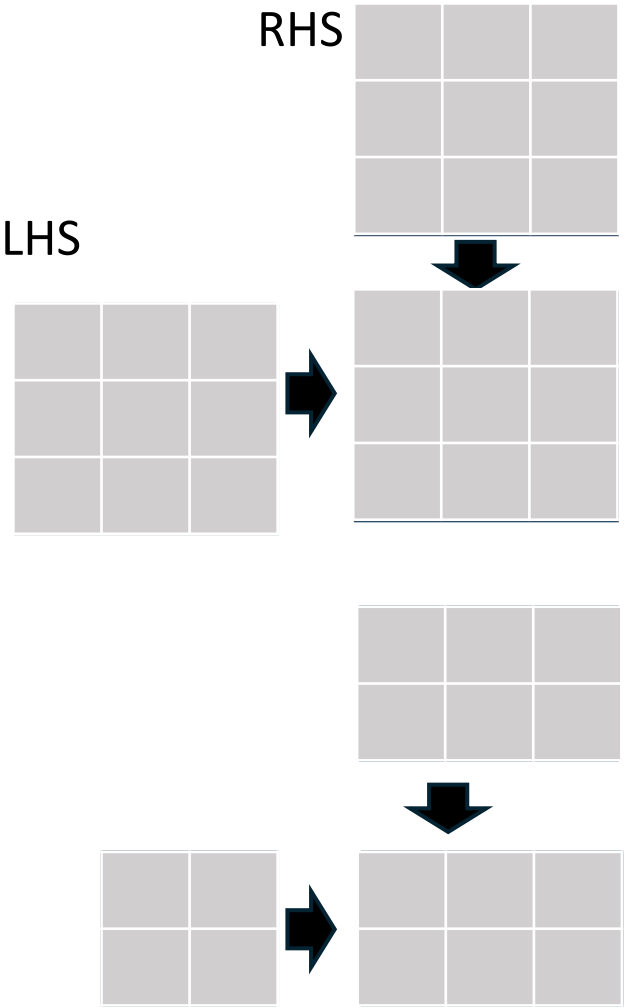




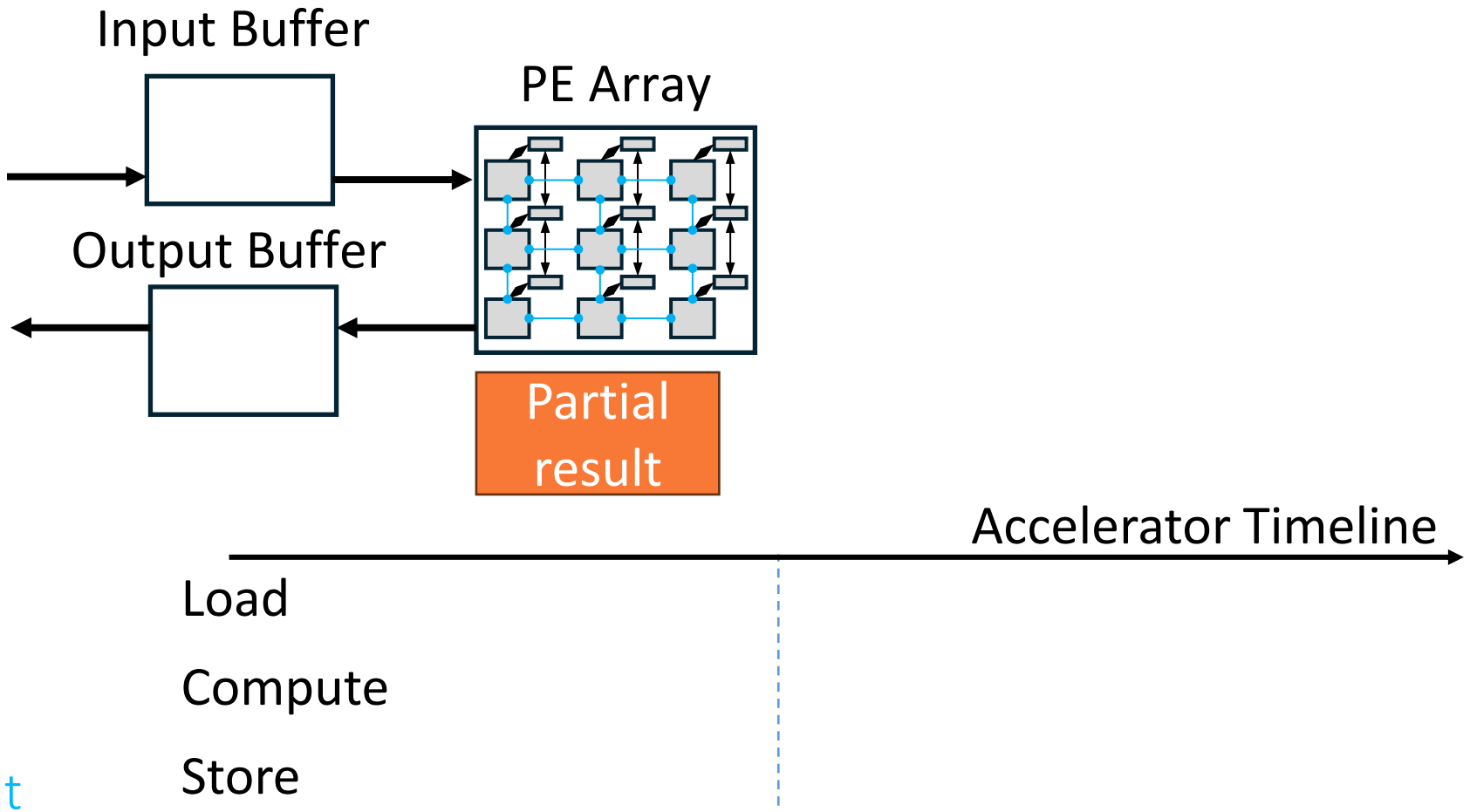
DERCA: Intra-Layer Preemptive Accelerator Design

Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



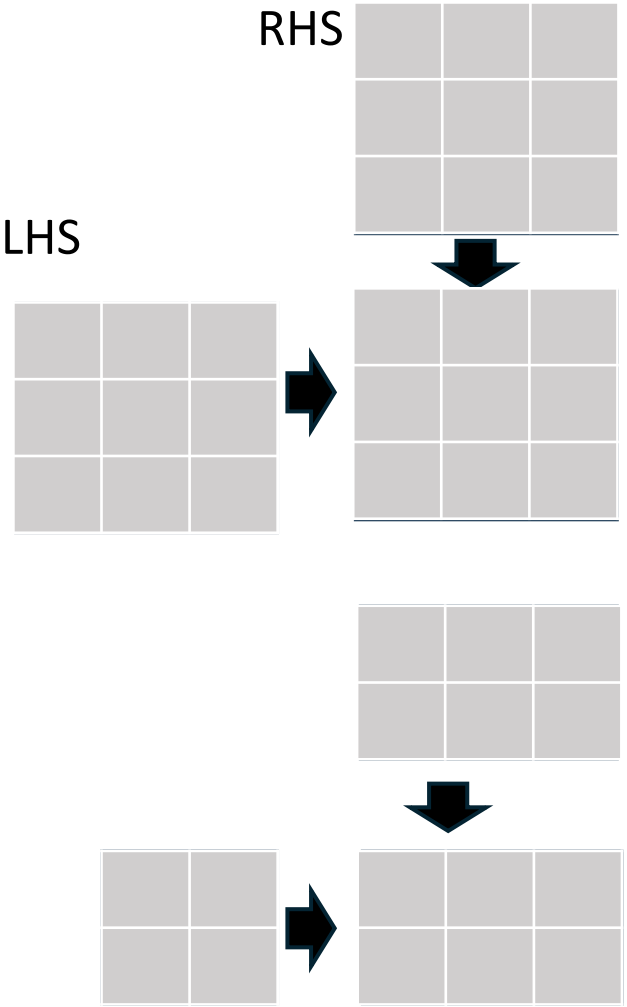
Same procedure, different #tiles (loop iterations)



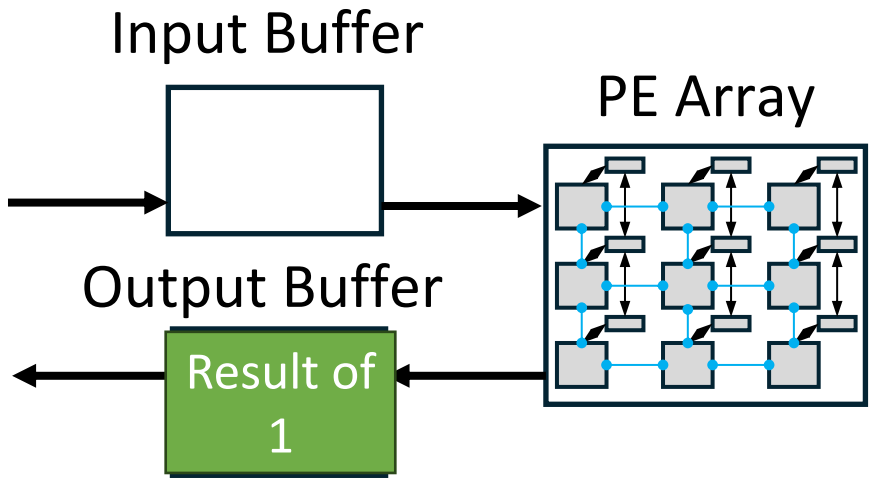
DERCA: Intra-Layer Preemptive Accelerator Design

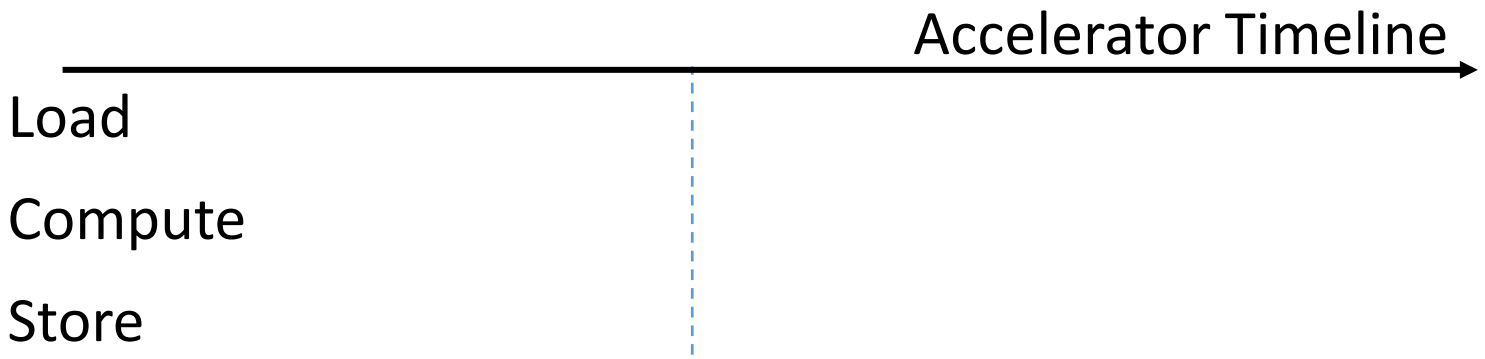
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

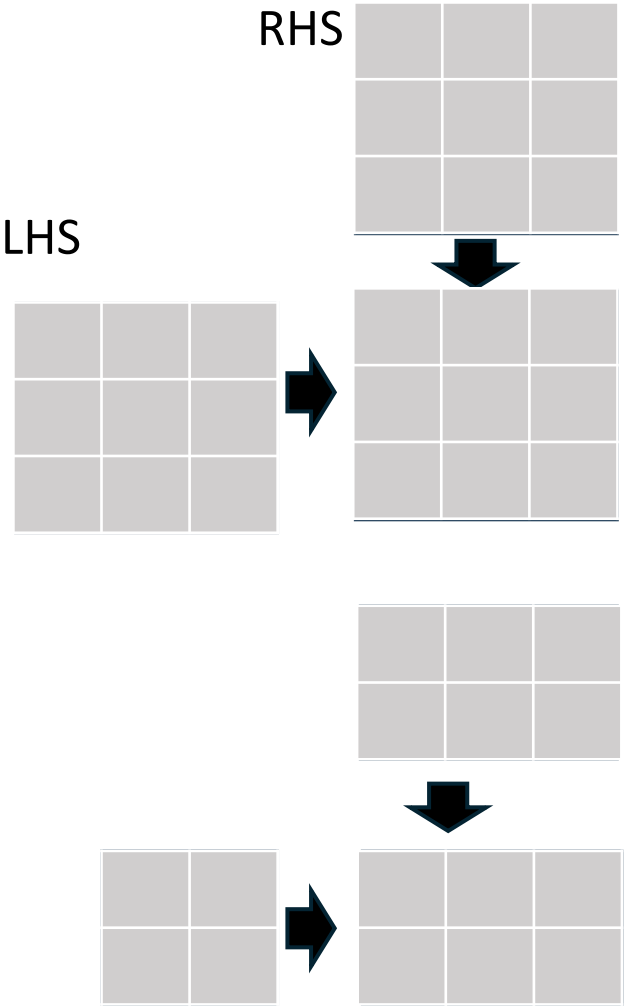




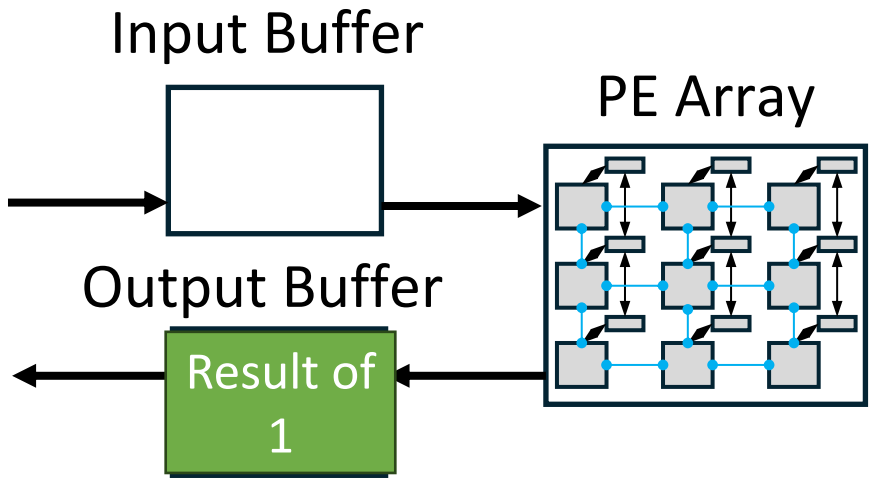
DERCA: Intra-Layer Preemptive Accelerator Design

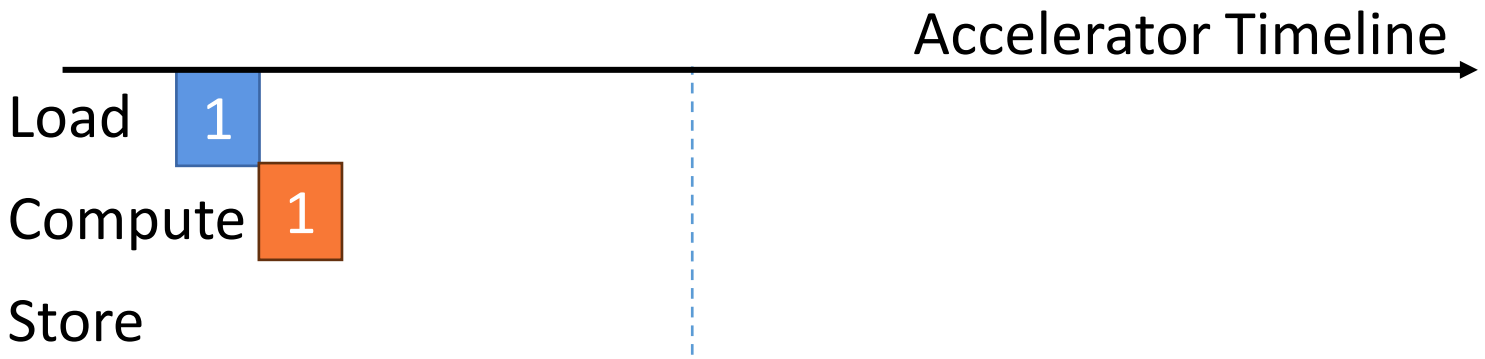
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

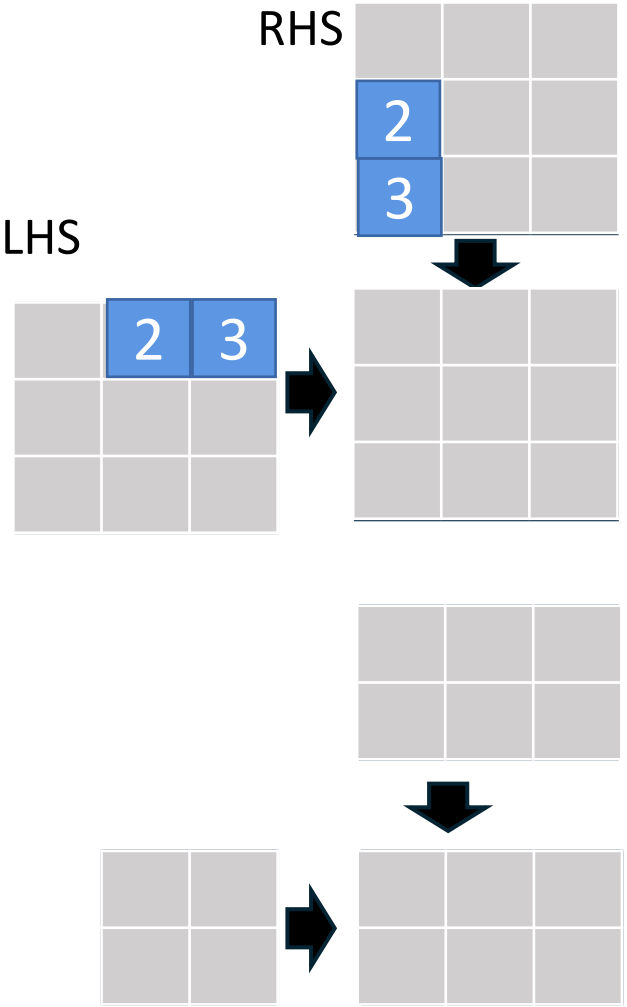




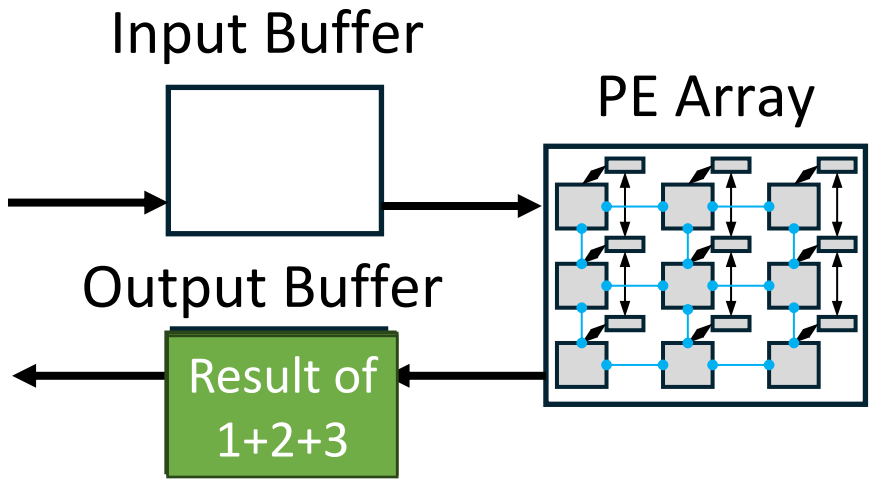
DERCA: Intra-Layer Preemptive Accelerator Design

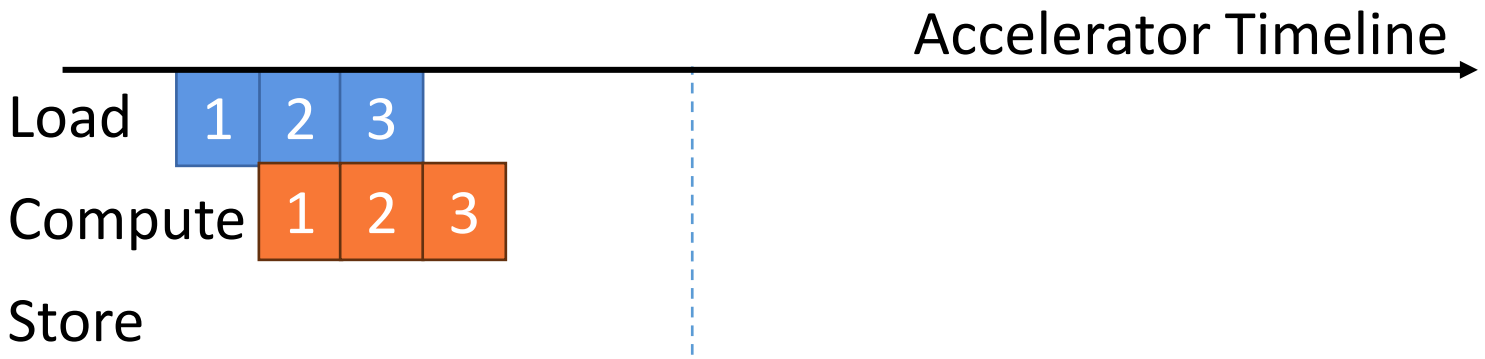
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

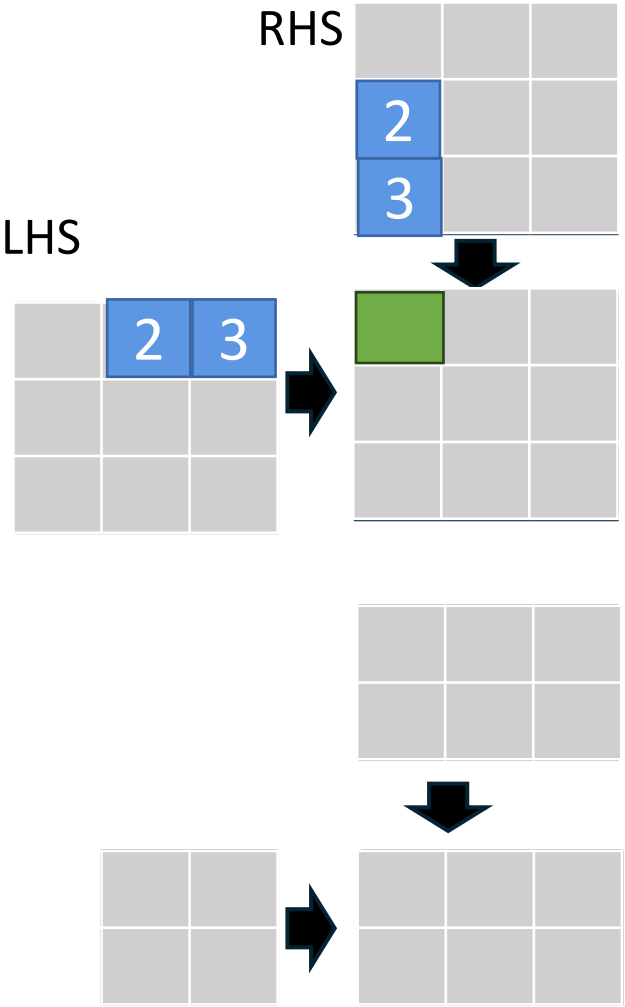




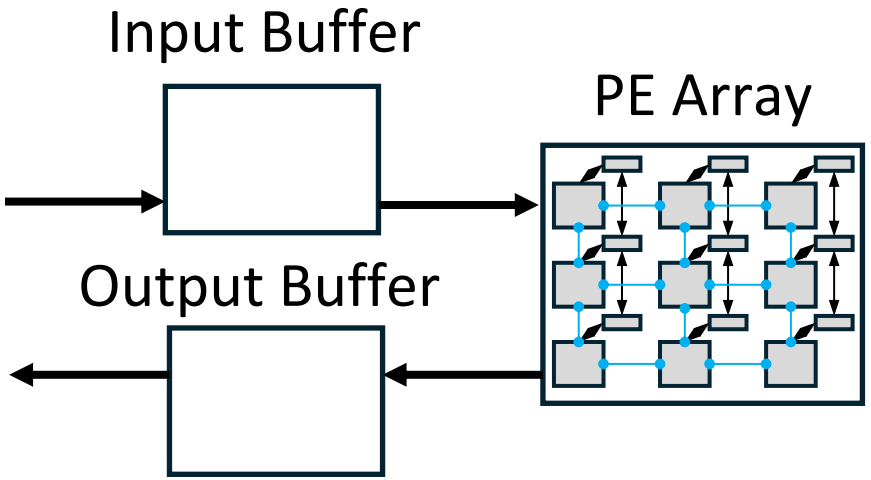
DERCA: Intra-Layer Preemptive Accelerator Design

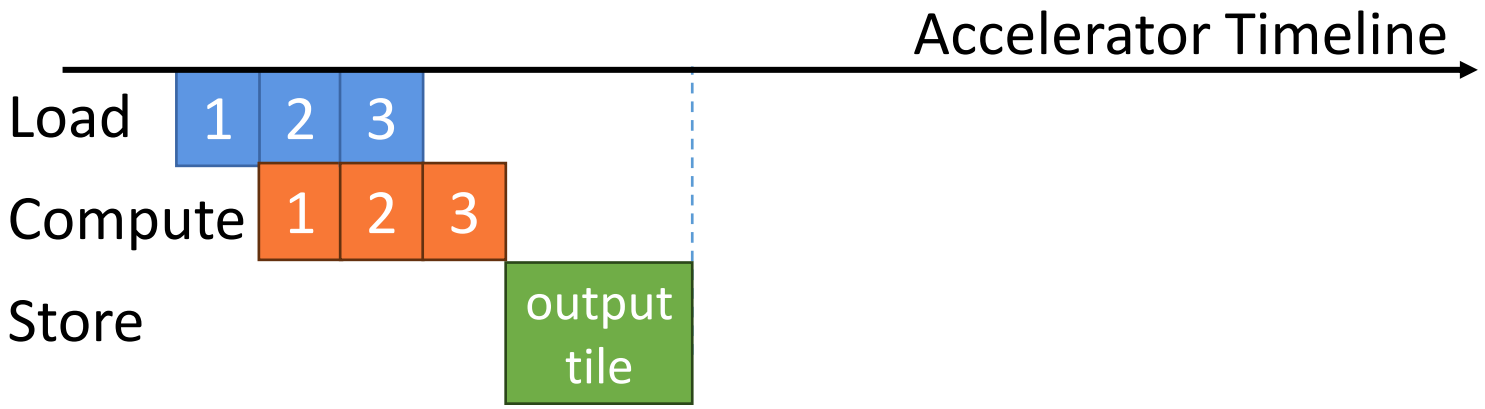
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

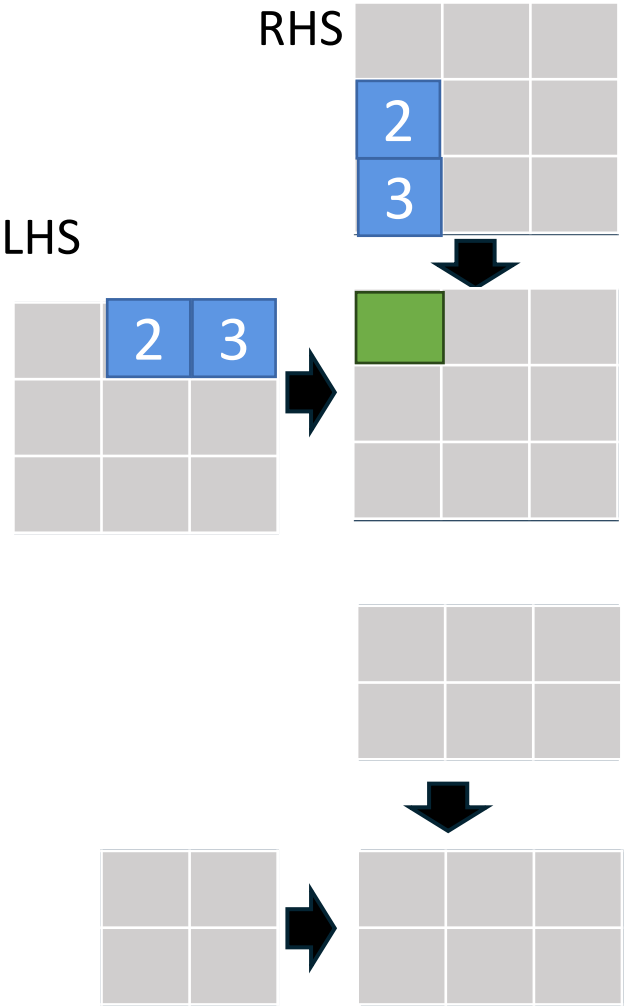




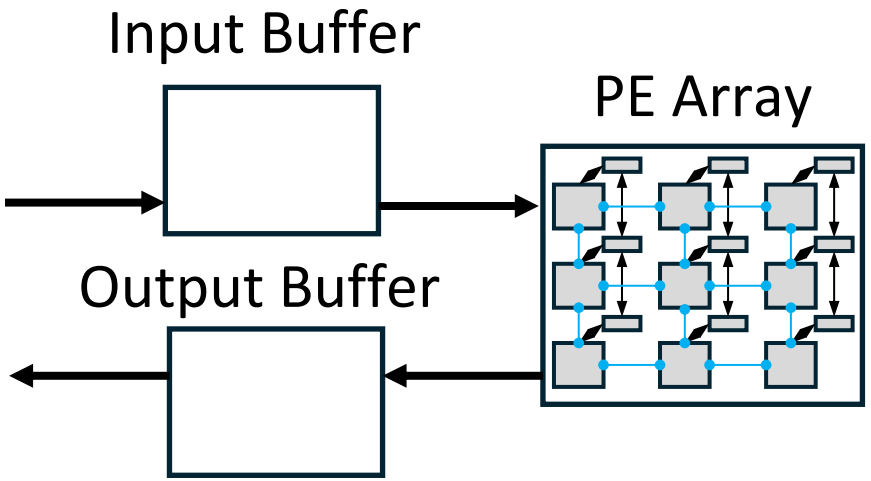
DERCA: Intra-Layer Preemptive Accelerator Design

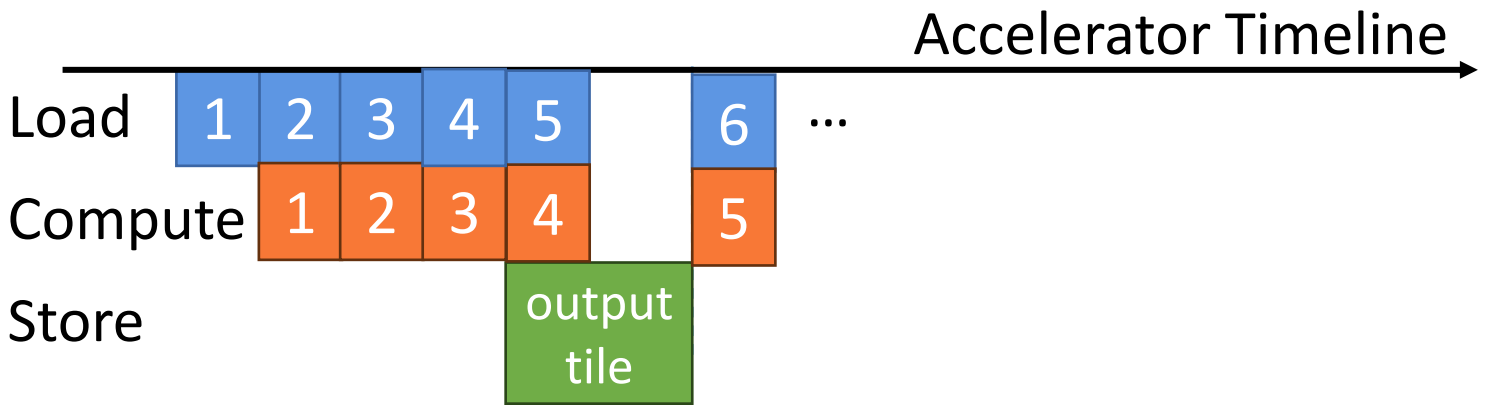
Where should we partition the DNN layer? → execution procedures

→ same acc + different configs



Same procedure, different #tiles (loop iterations)

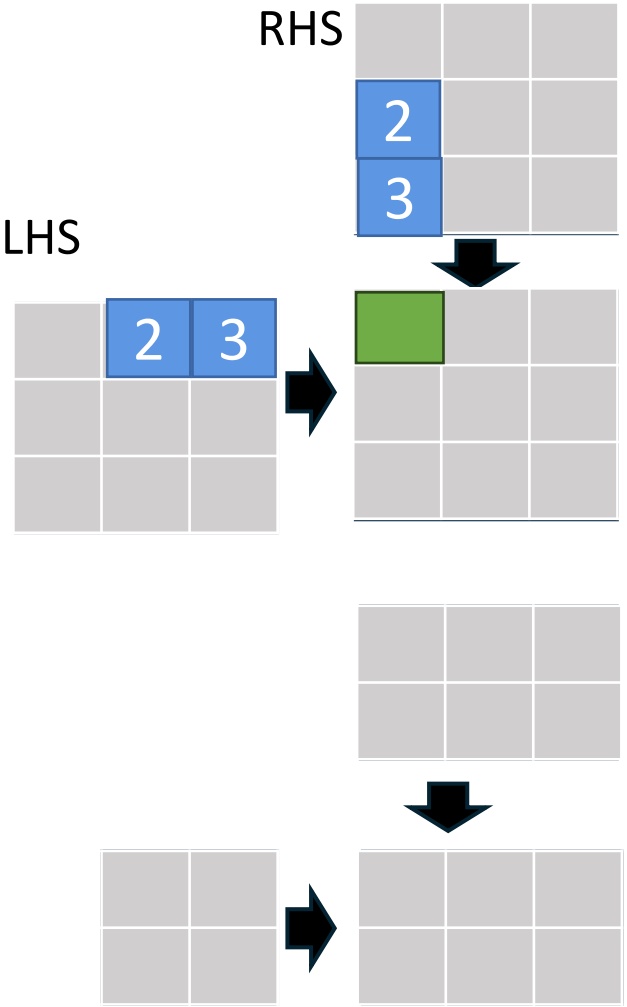




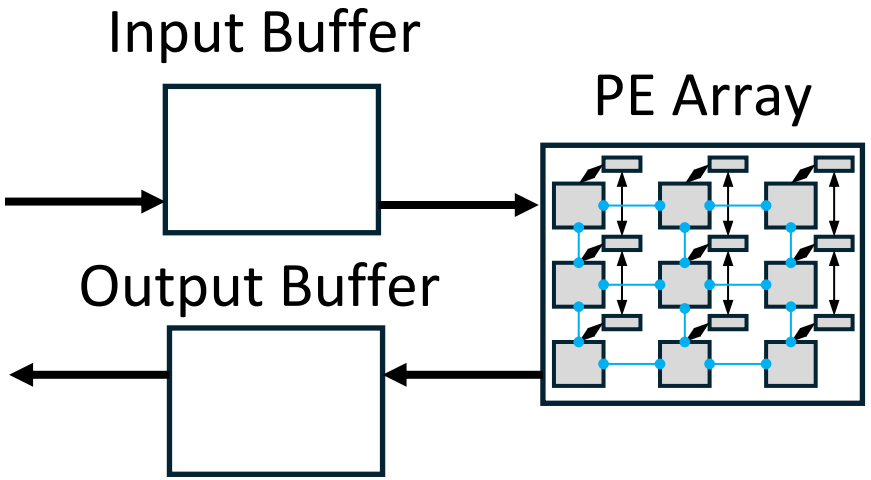
DERCA: Intra-Layer Preemptive Accelerator Design

Where should we partition the DNN layer? → execution procedures

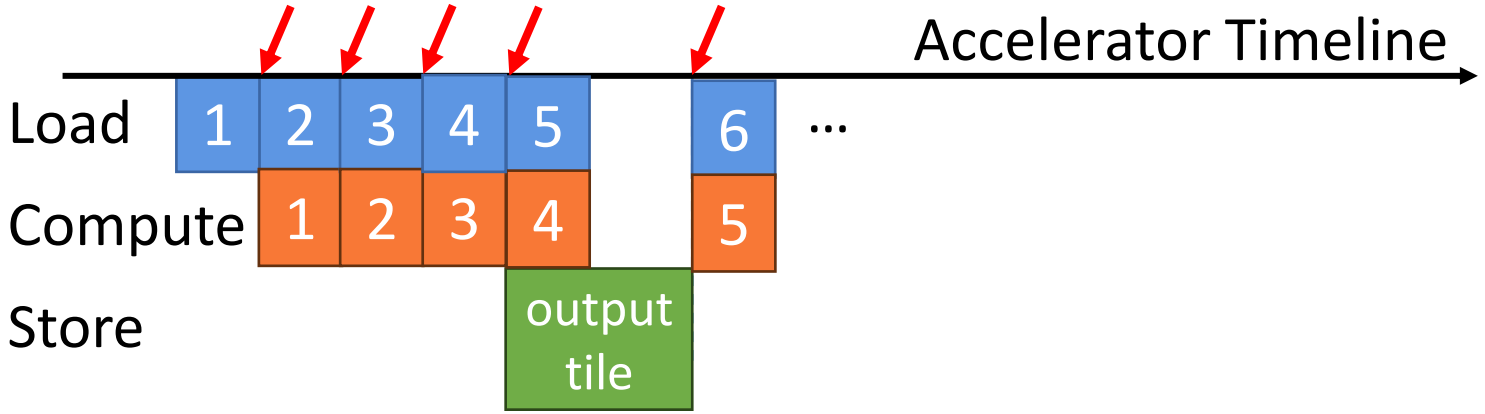
→ same acc + different configs



Same procedure, different #tiles (loop iterations)



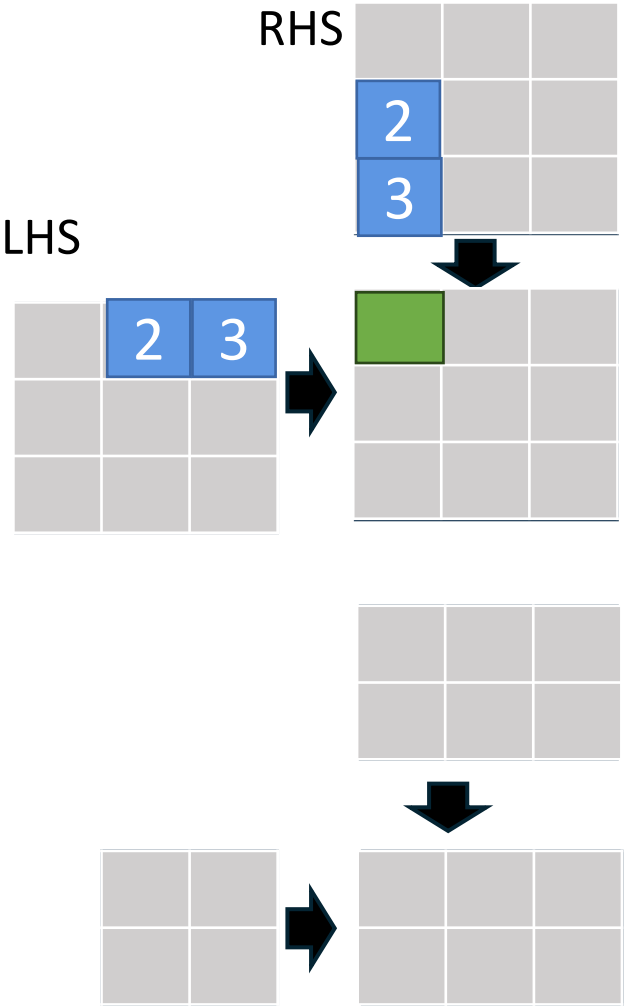
Potential PPs: gap between tiles



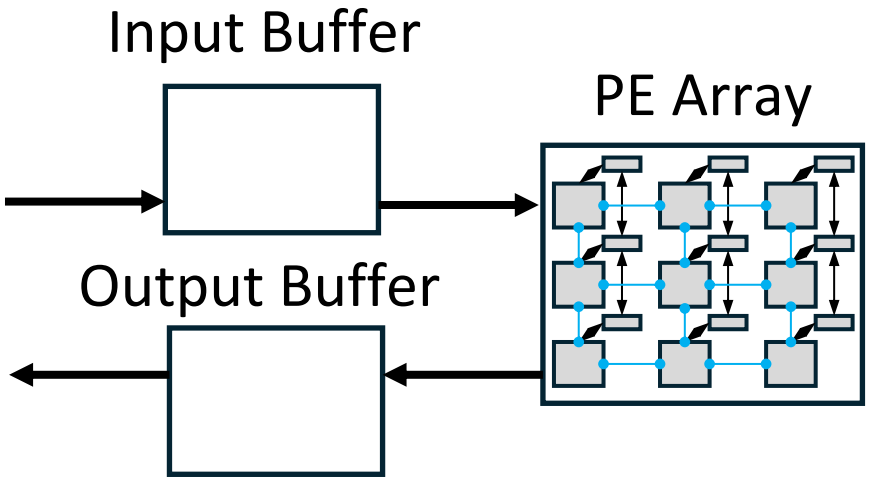
DERCA: Intra-Layer Preemptive Accelerator Design

Where should we partition the DNN layer? → execution procedures

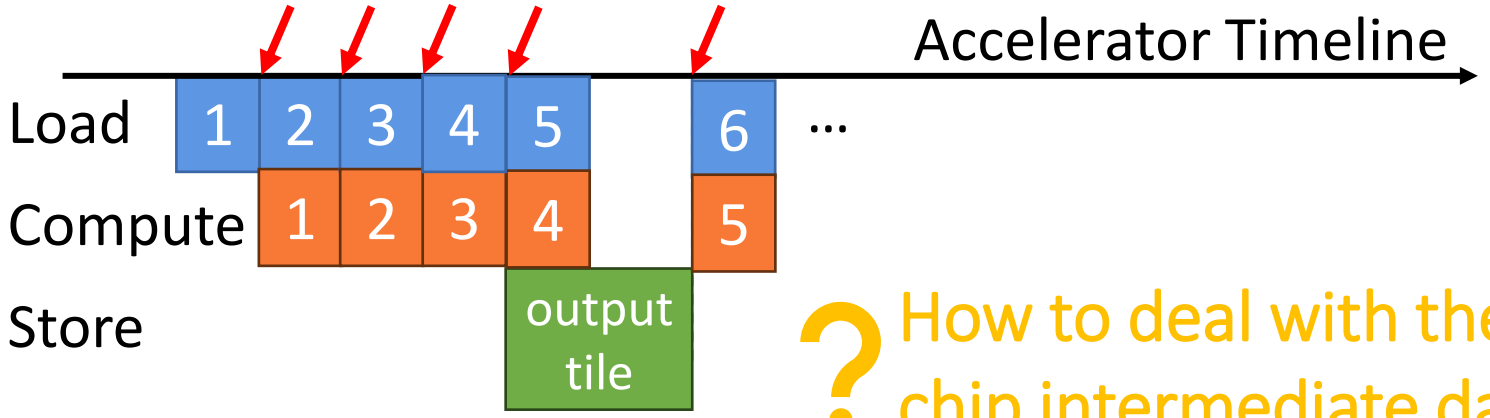
→ same acc + different configs



Same procedure, different #tiles (loop iterations)

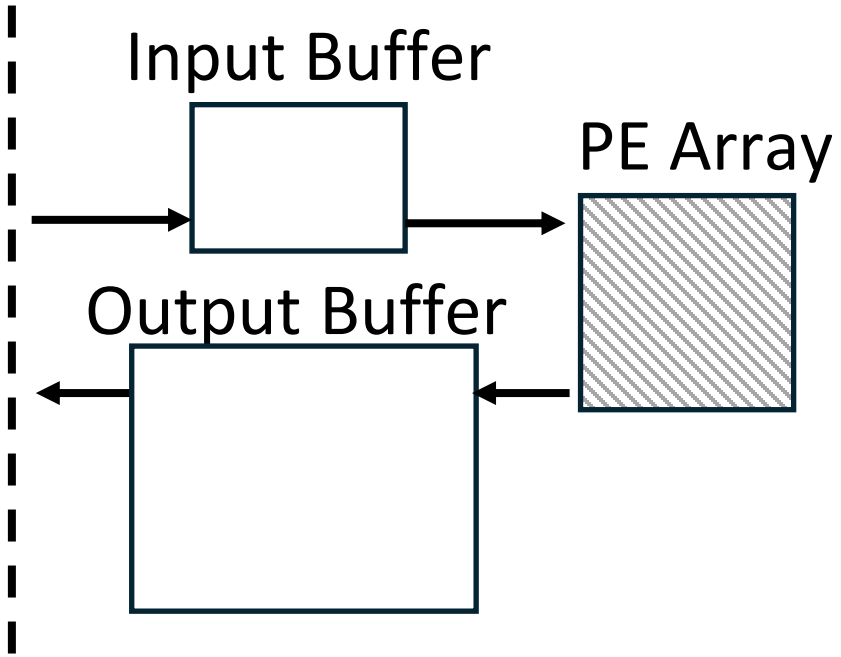


Potential PPs: gap between tiles

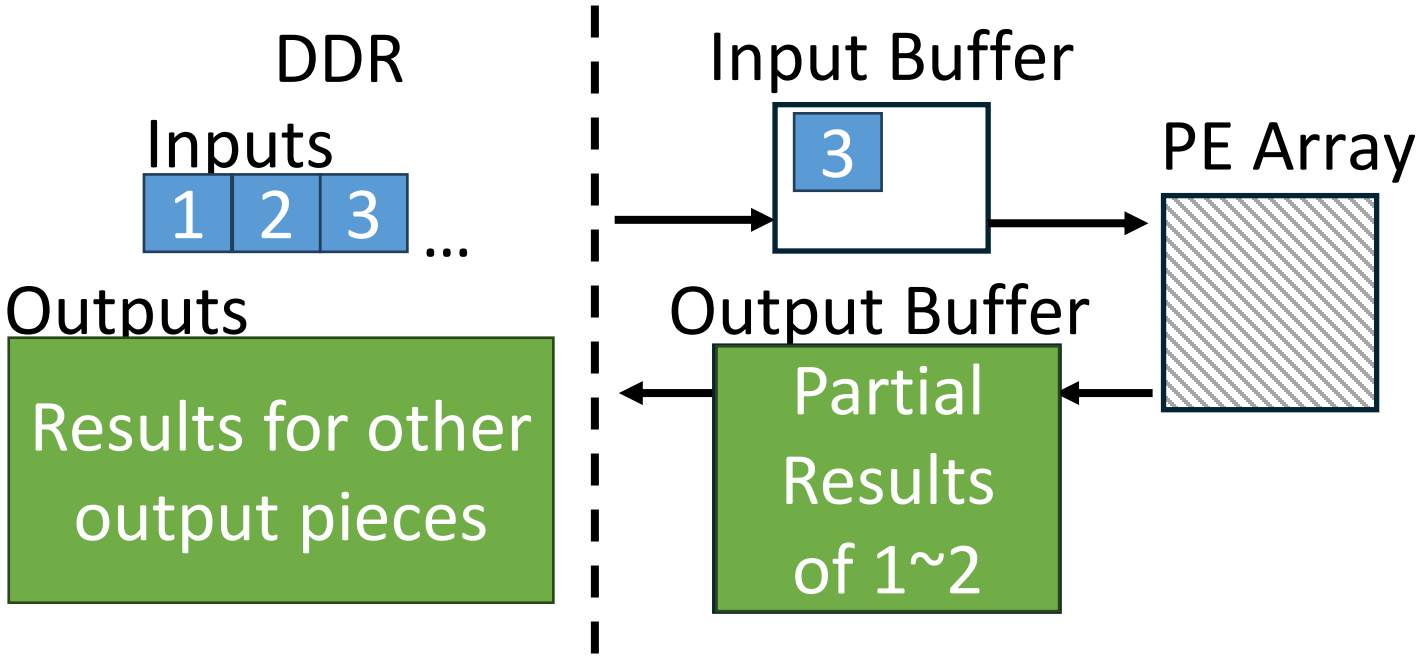


? How to deal with the on-chip intermediate data

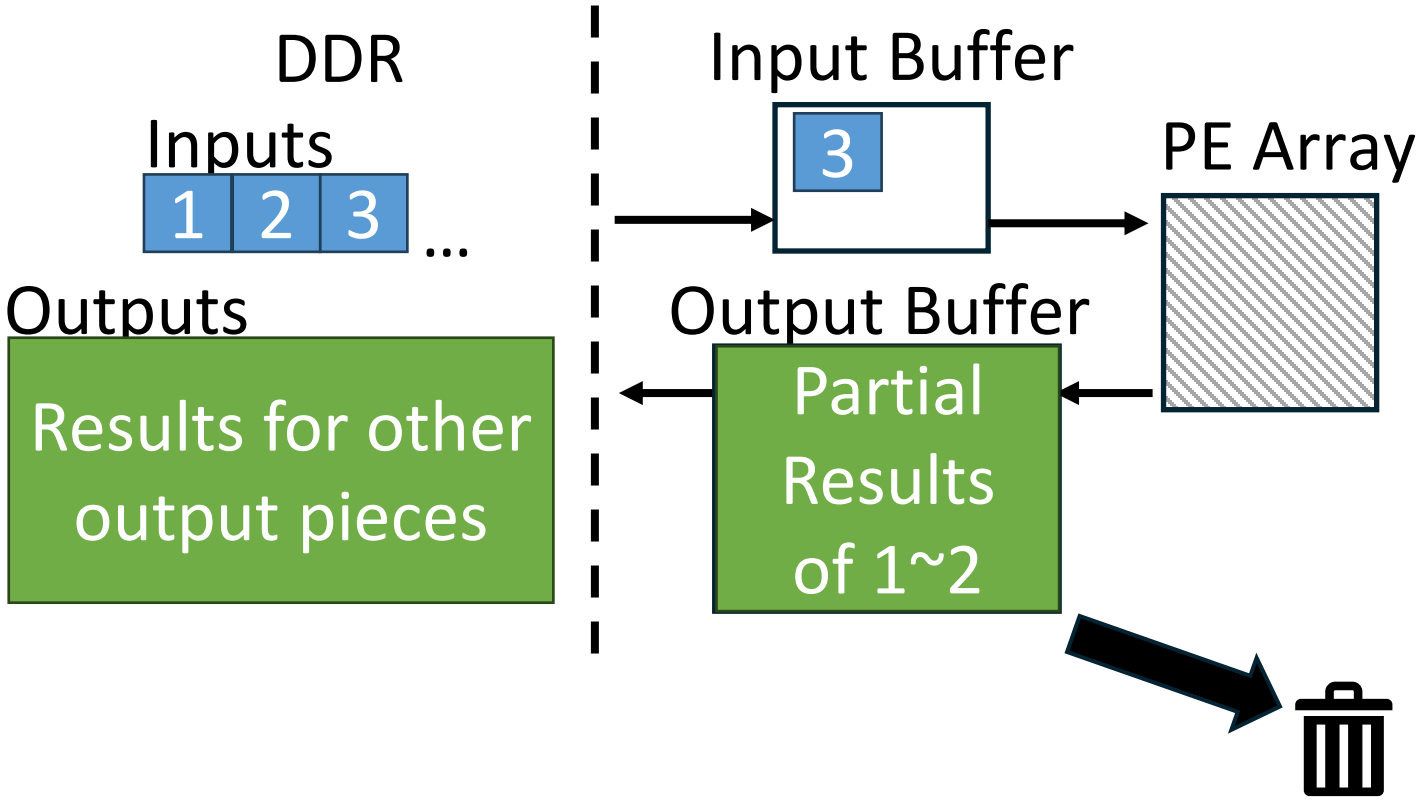
DERCA: Two Ways of Dealing with Intermediate Data



DERCA: Two Ways of Dealing with Intermediate Data

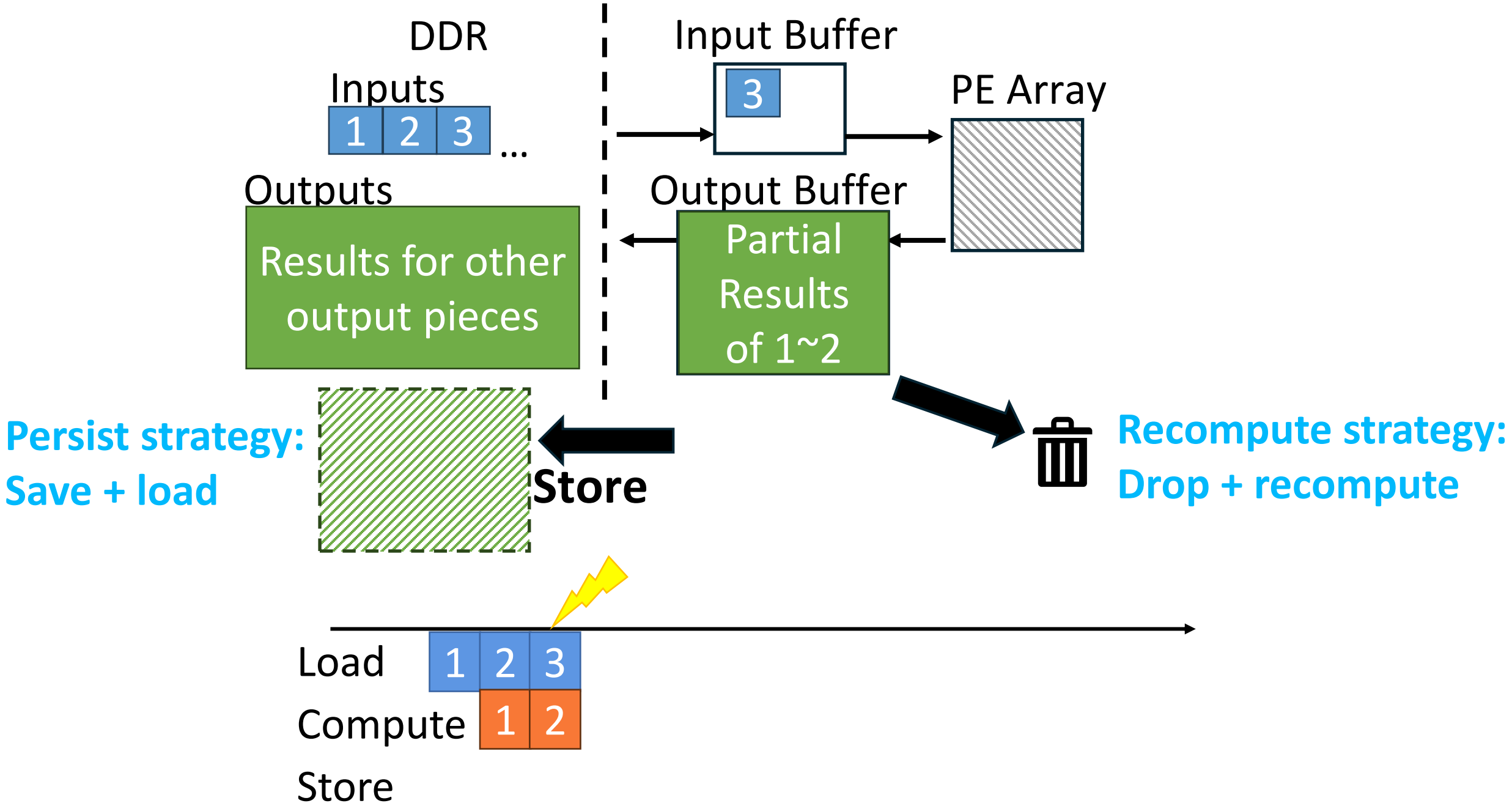


DERCA: Two Ways of Dealing with Intermediate Data

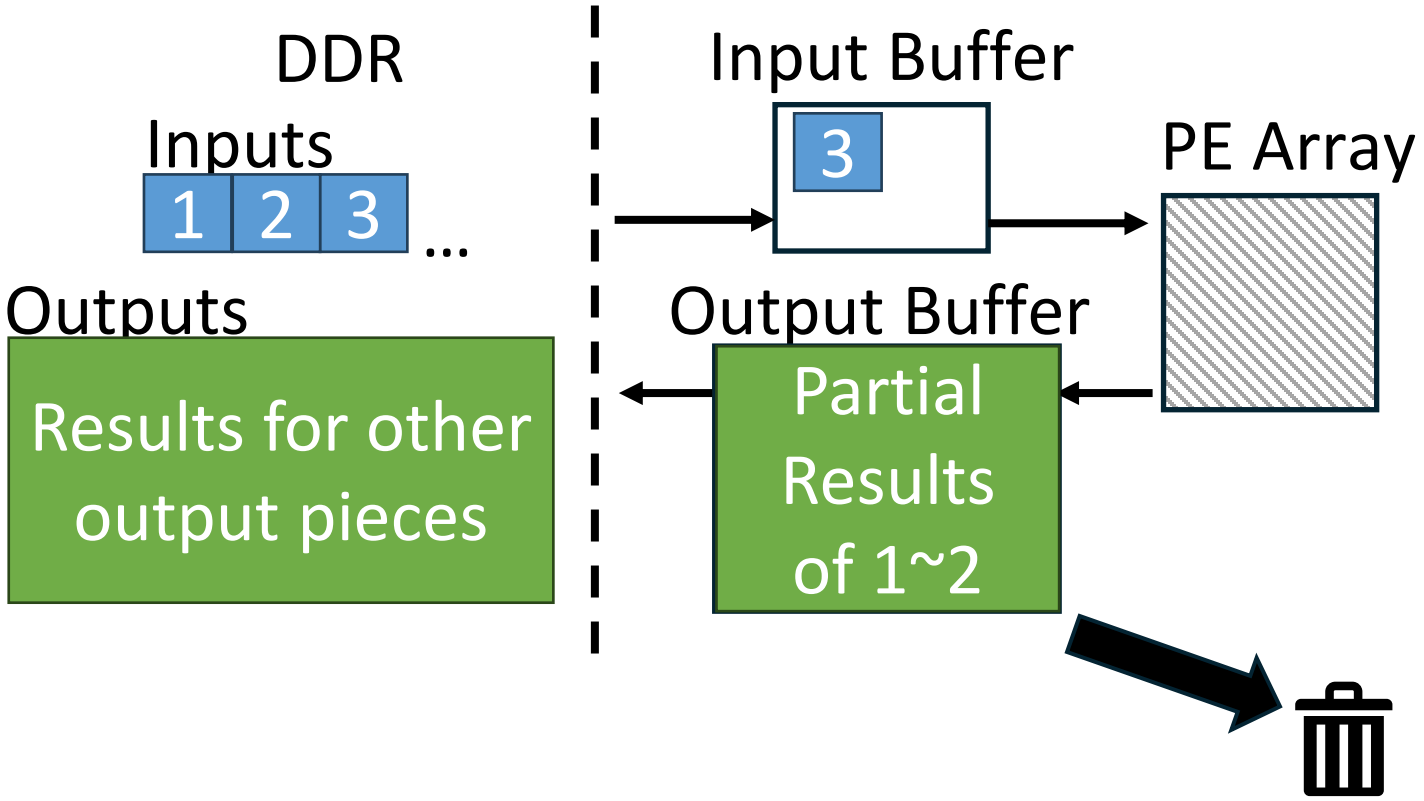


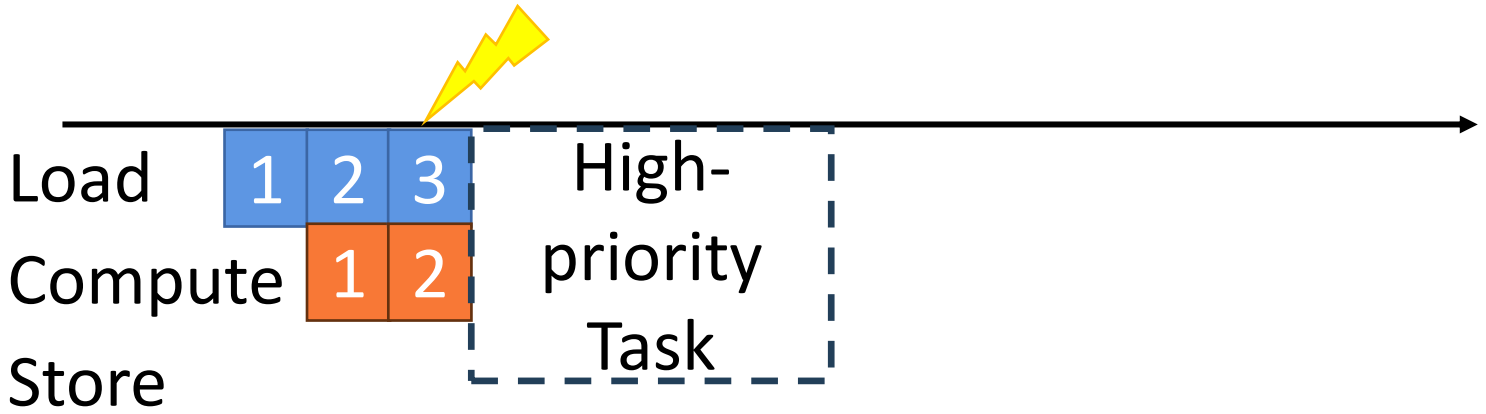
Recompute strategy:
Drop + recompute

DERCA: Two Ways of Dealing with Intermediate Data

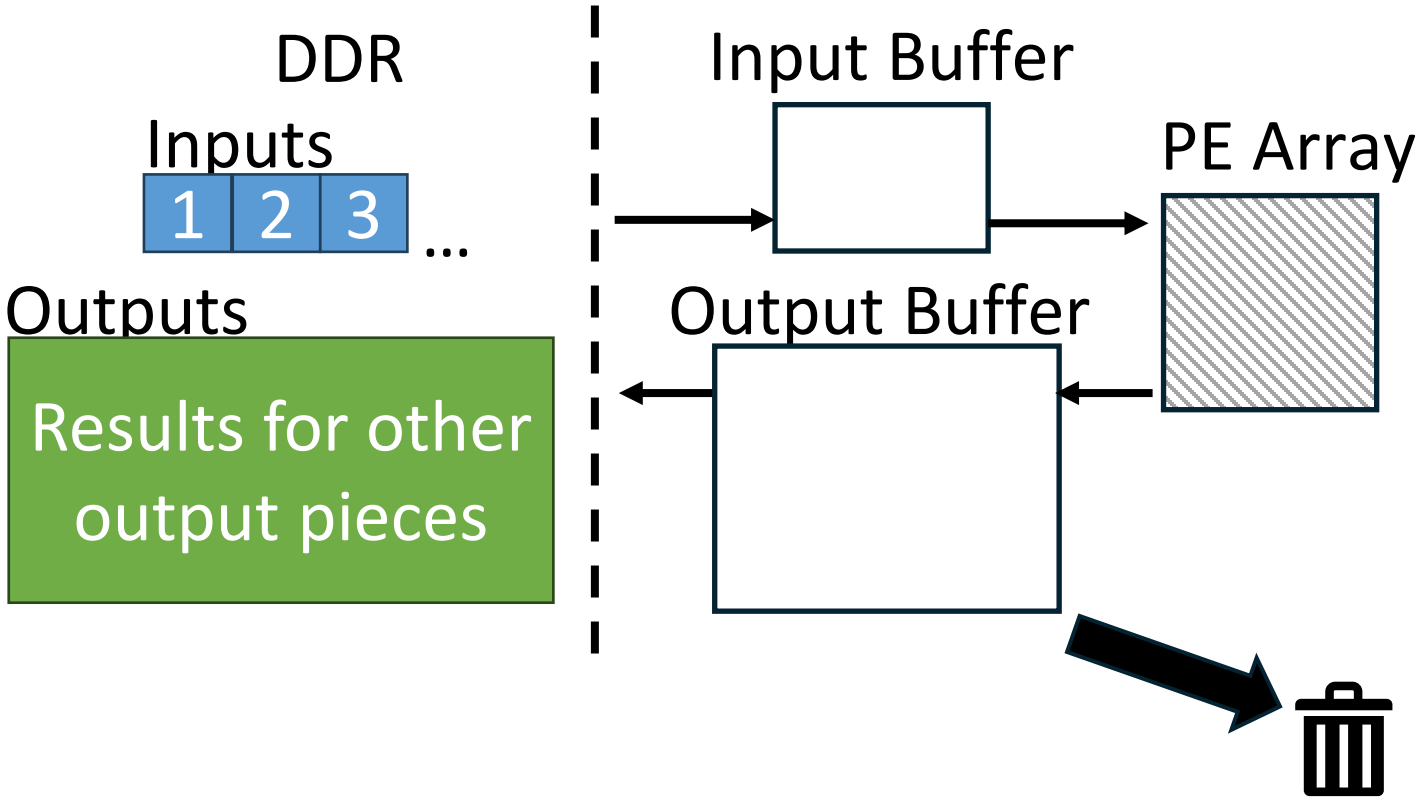


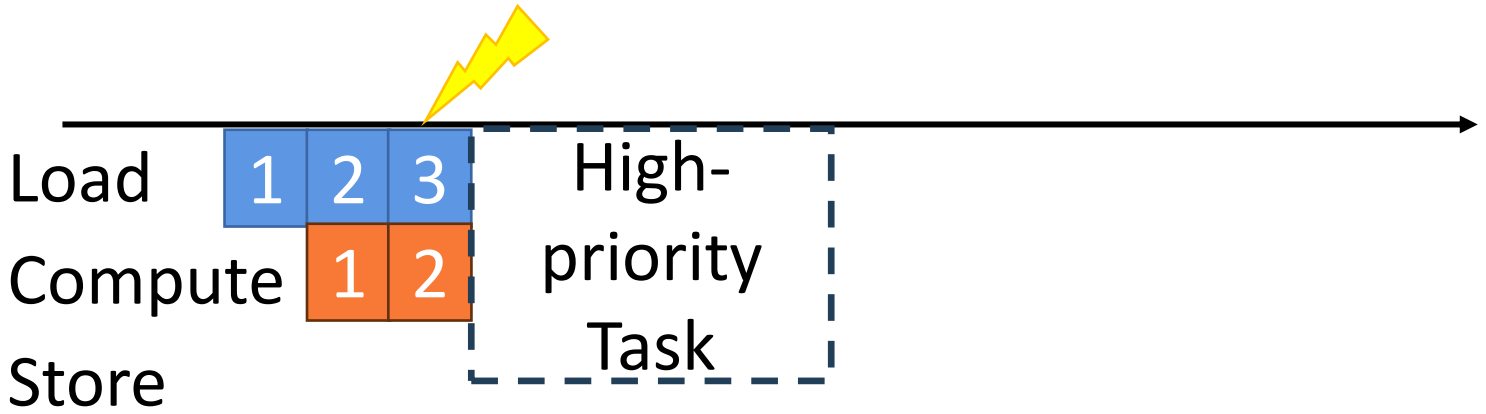
DERCA: Recompute strategy



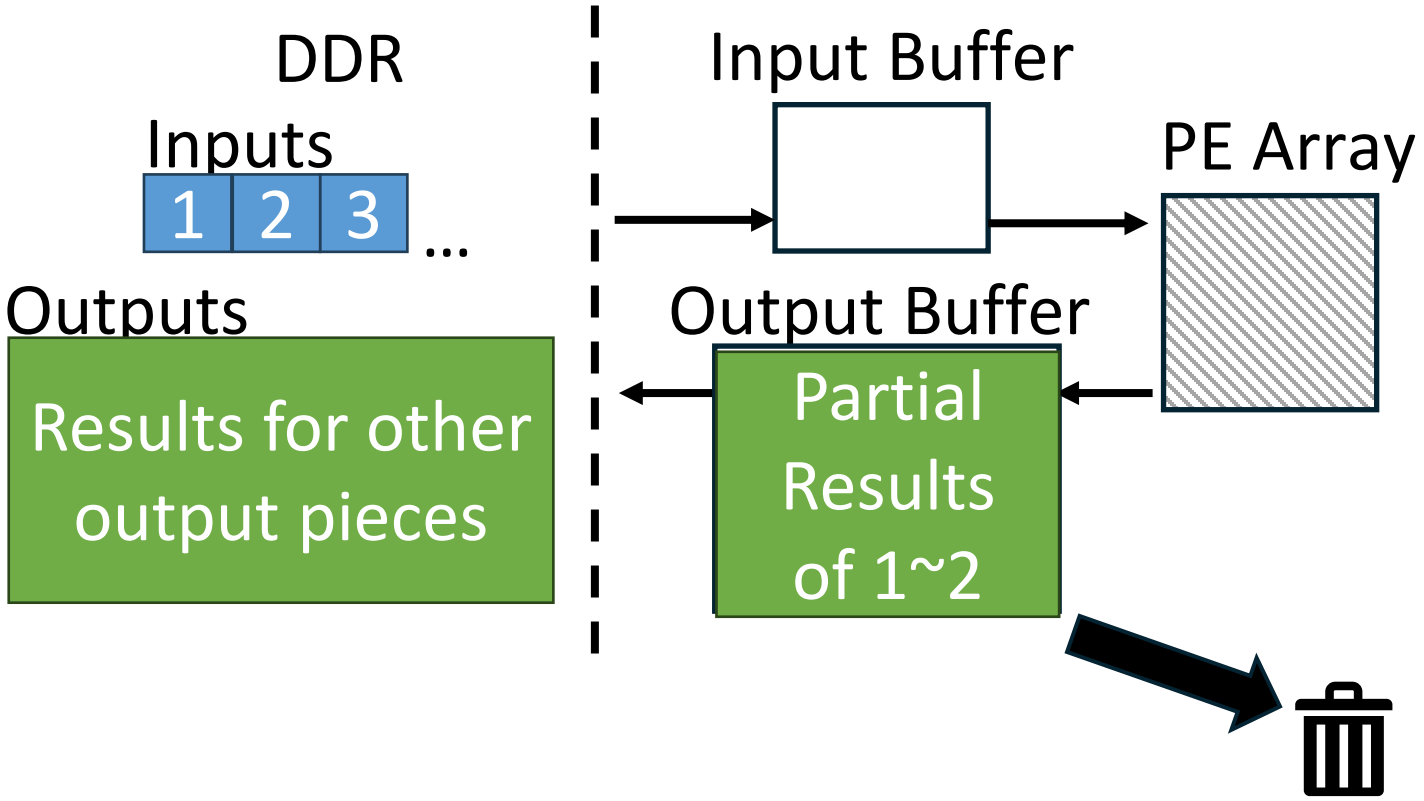


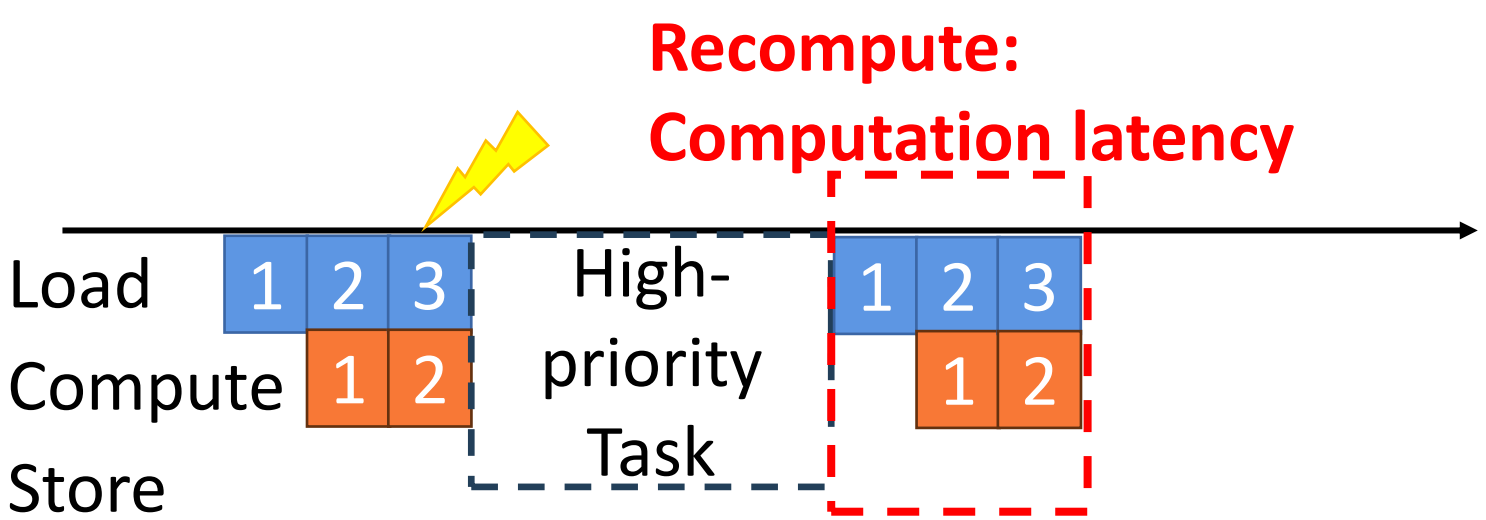
DERCA: Recompute strategy



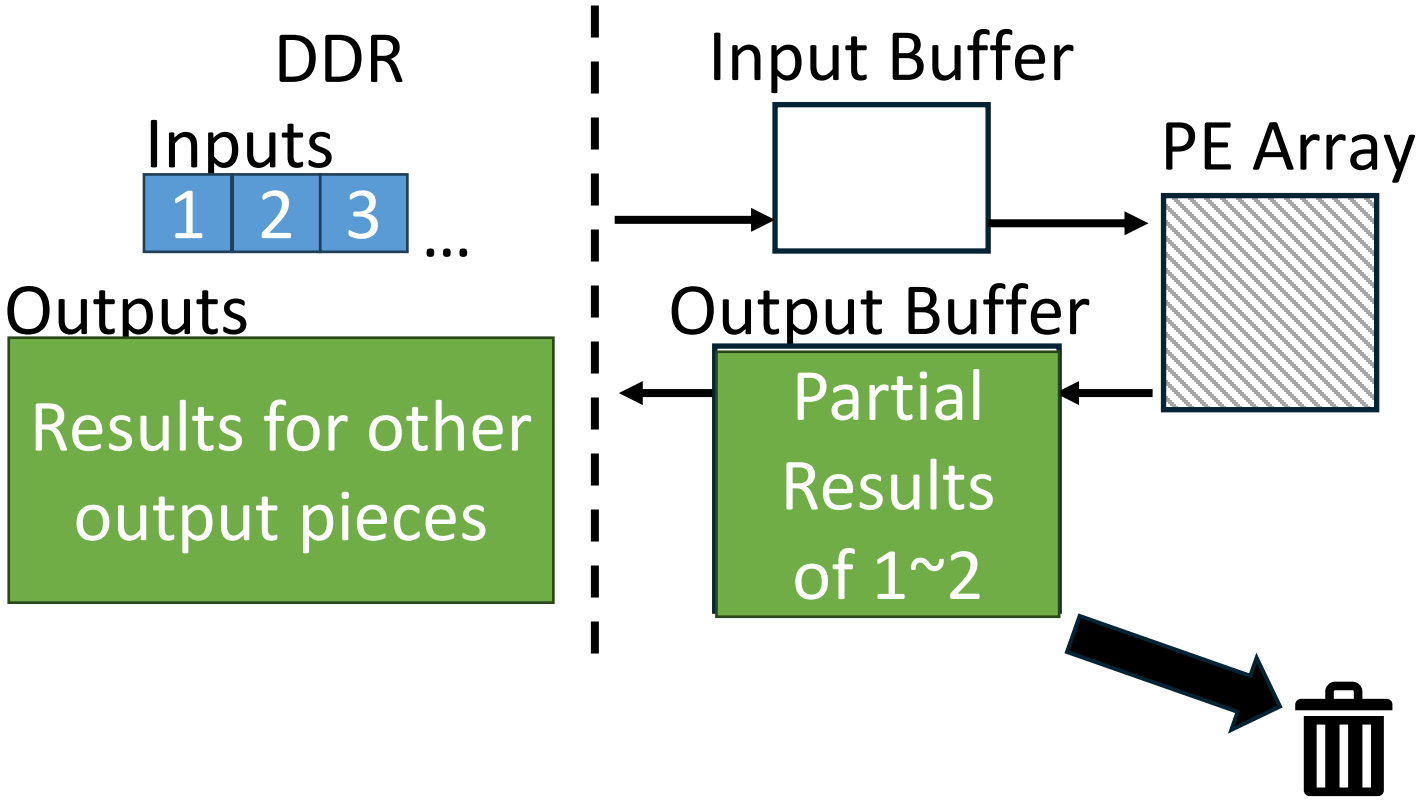


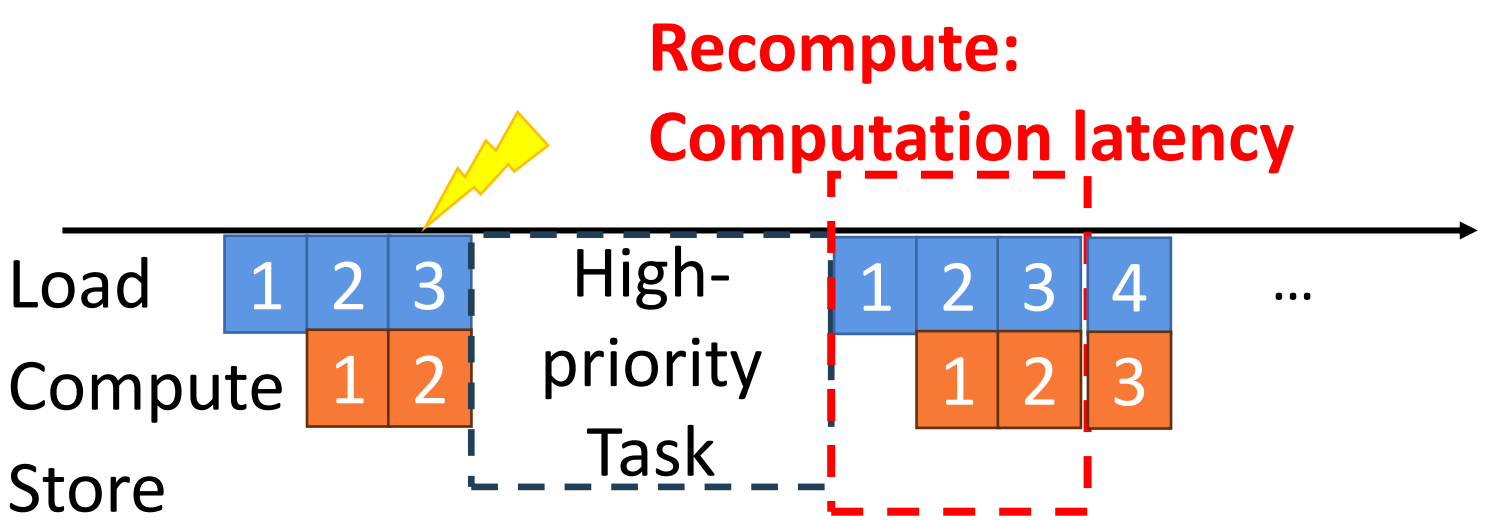
DERCA: Recompute strategy



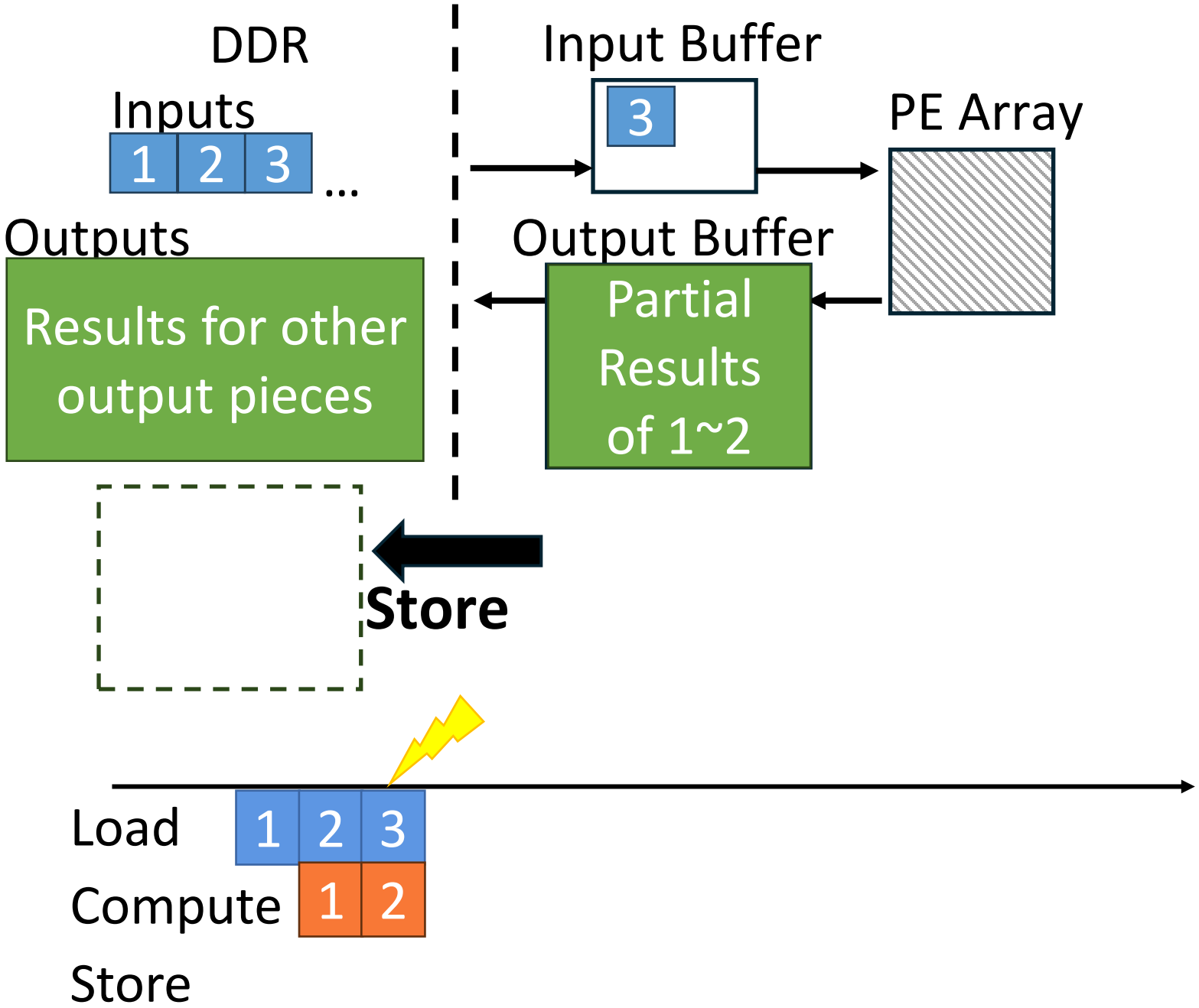


DERCA: Recompute strategy

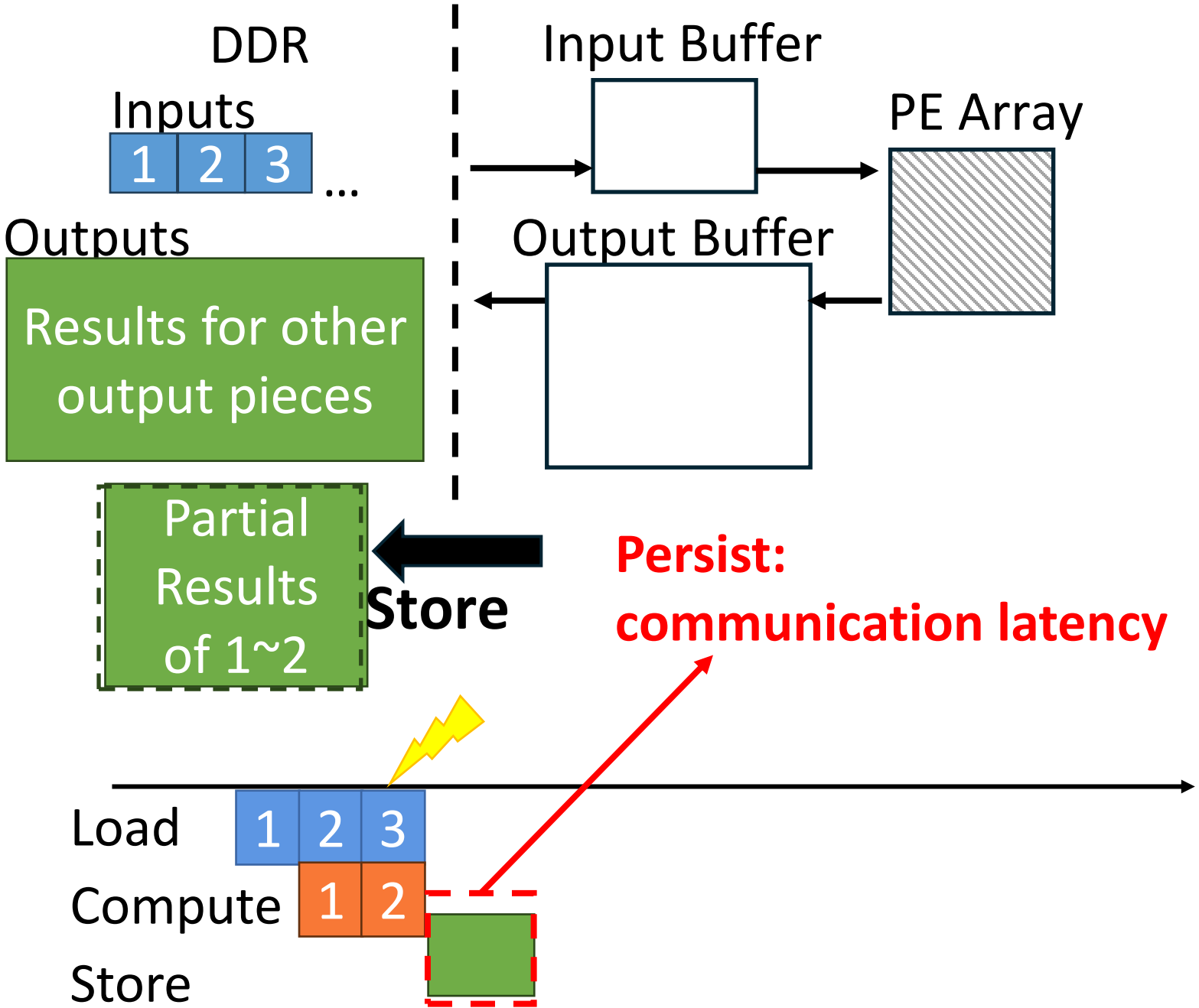




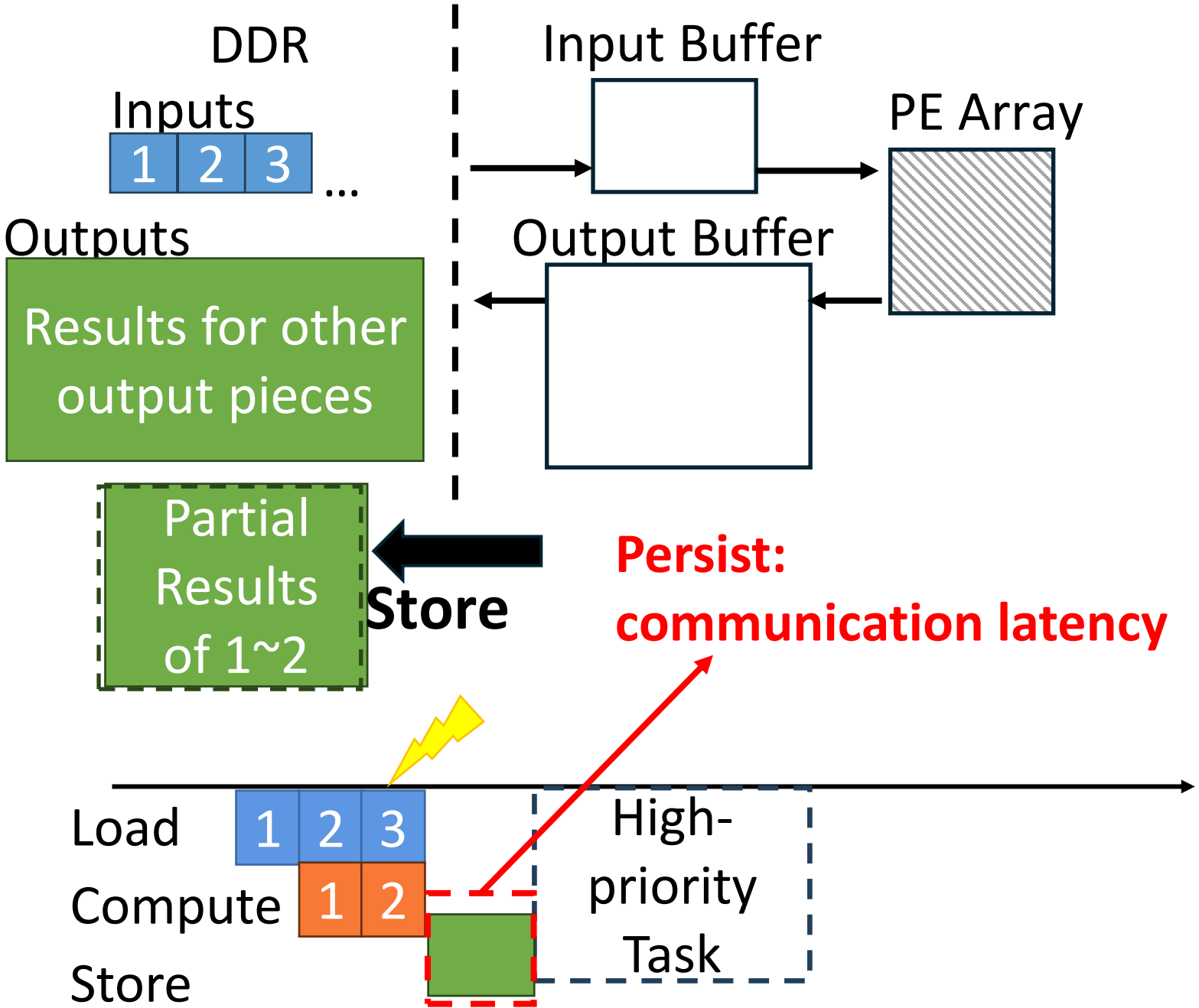
DERCA: Two Ways of Dealing with Intermediate Data



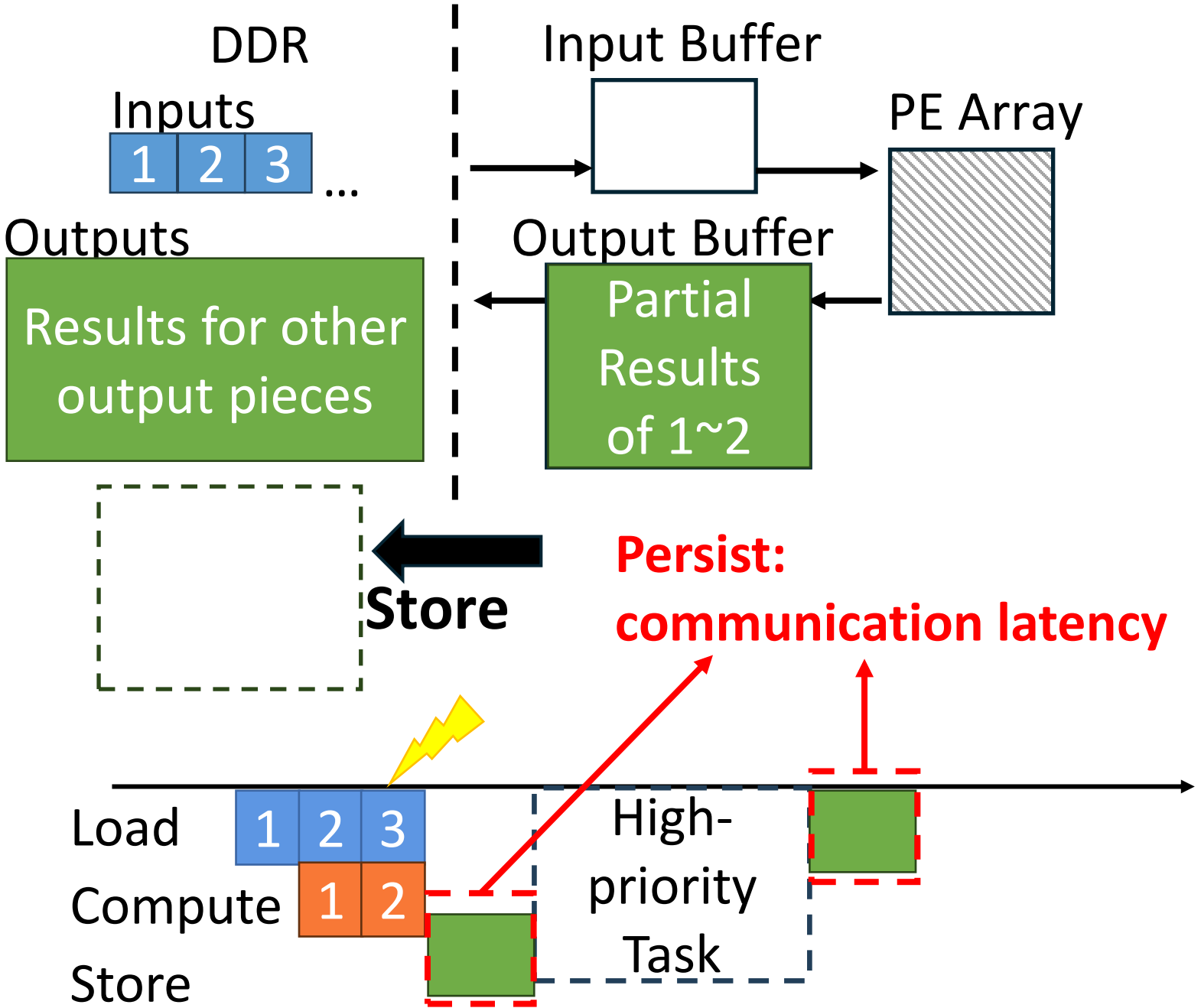
DERCA: Two Ways of Dealing with Intermediate Data



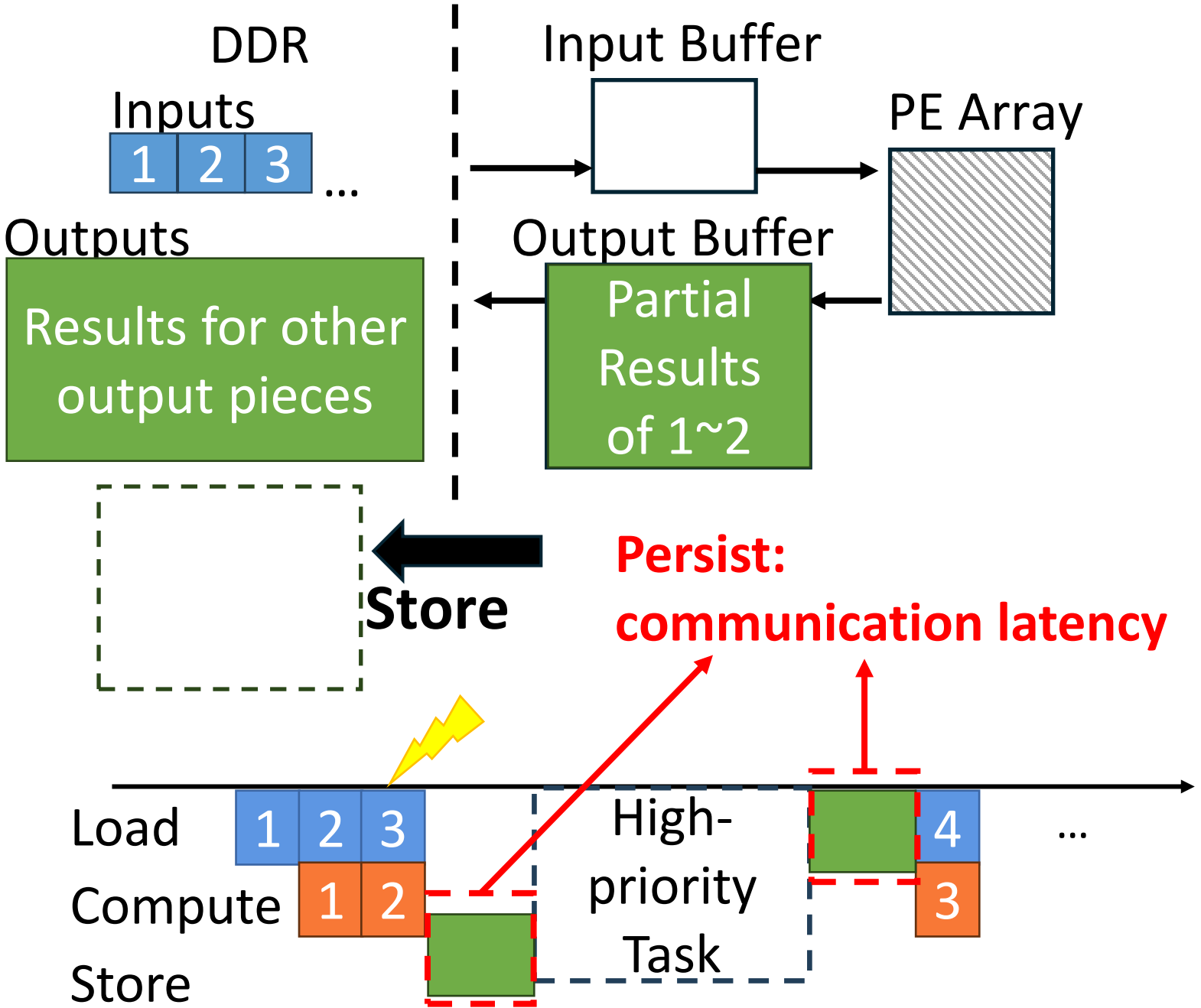
DERCA: Two Ways of Dealing with Intermediate Data



DERCA: Two Ways of Dealing with Intermediate Data

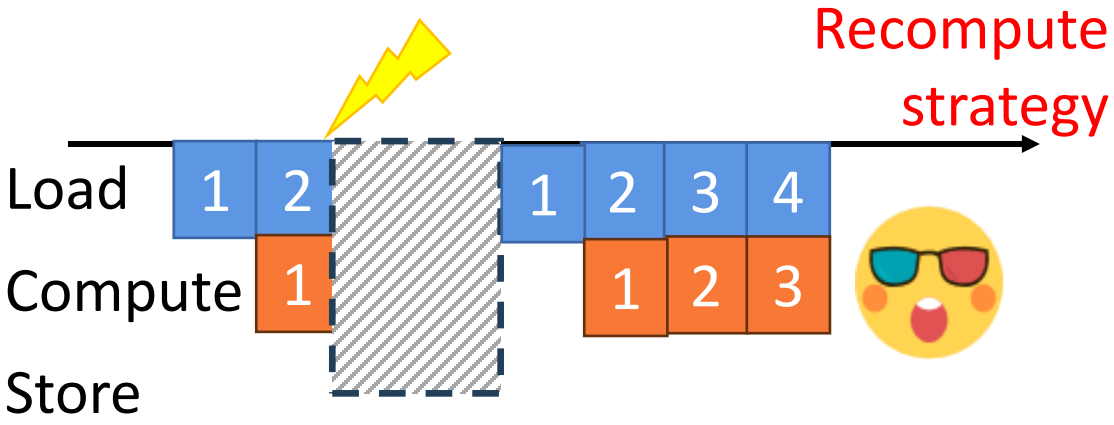


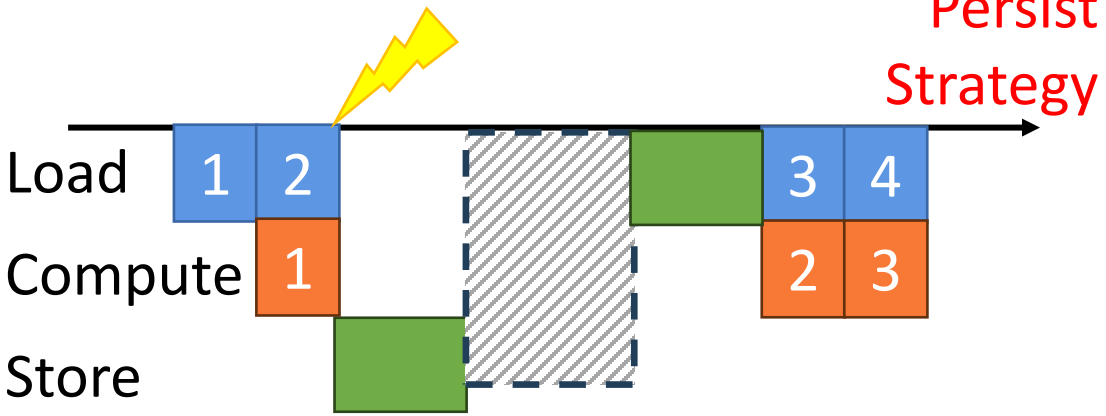
DERCA: Two Ways of Dealing with Intermediate Data



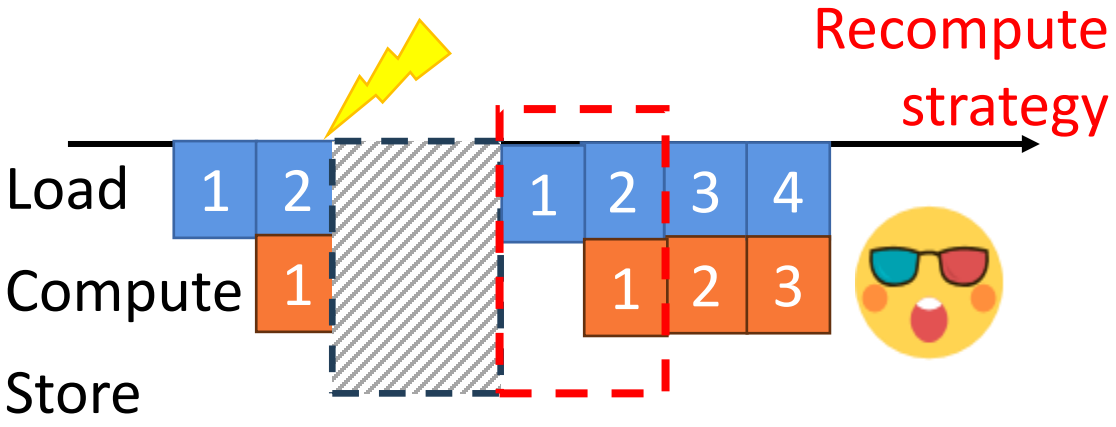
DERCA: Design Tradeoffs

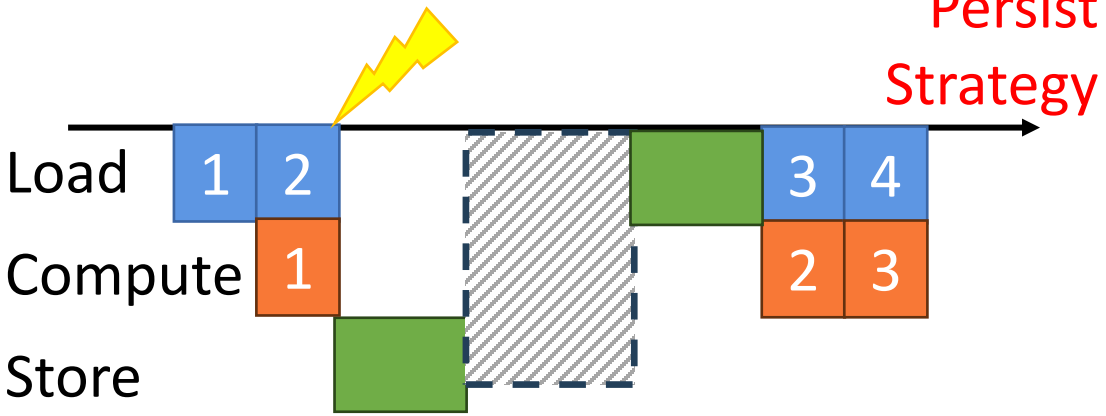
DERCA: Design Tradeoffs



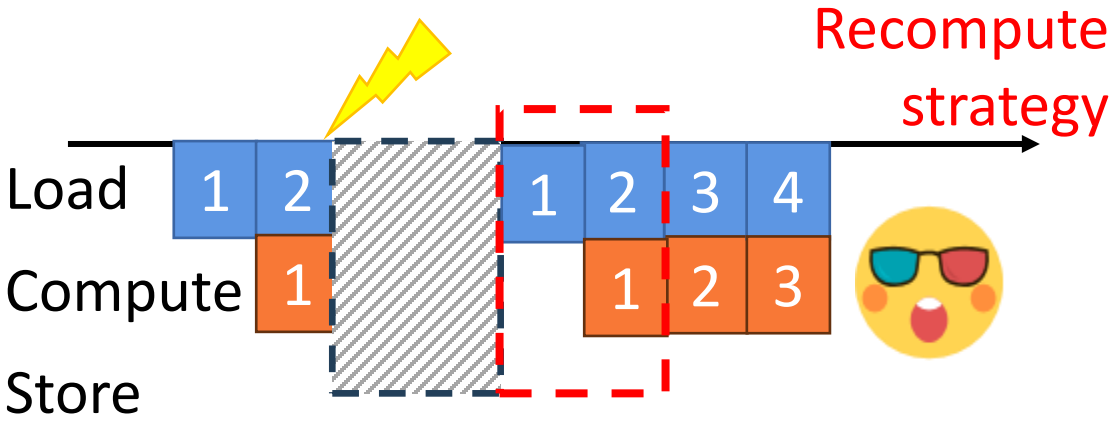


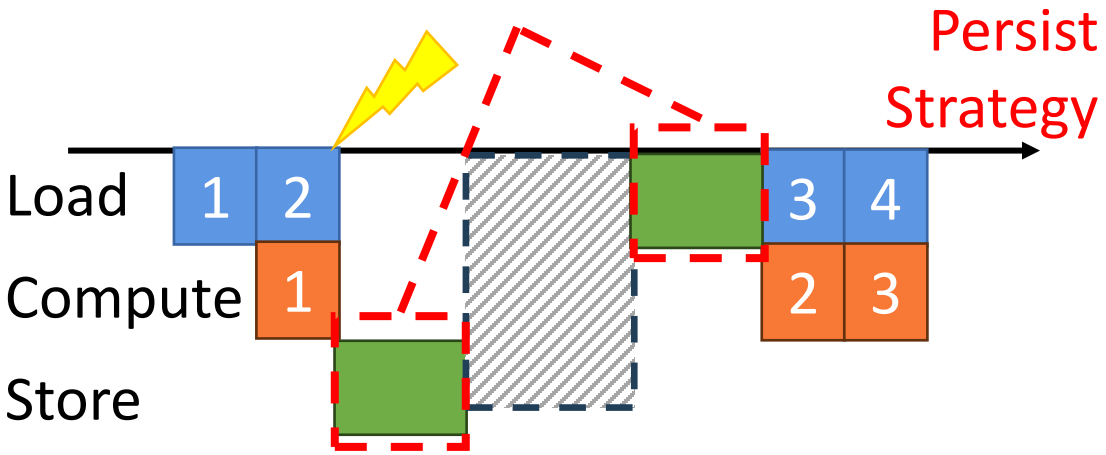
DERCA: Design Tradeoffs



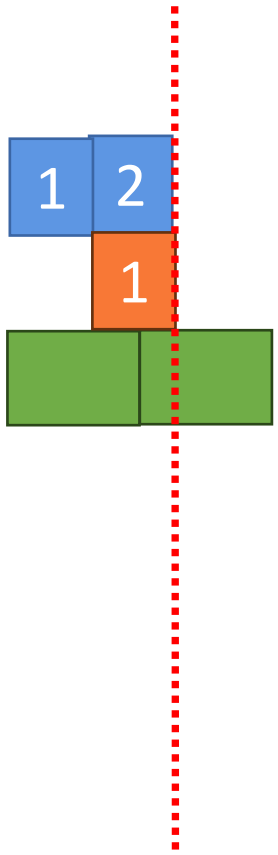


DERCA: Design Tradeoffs





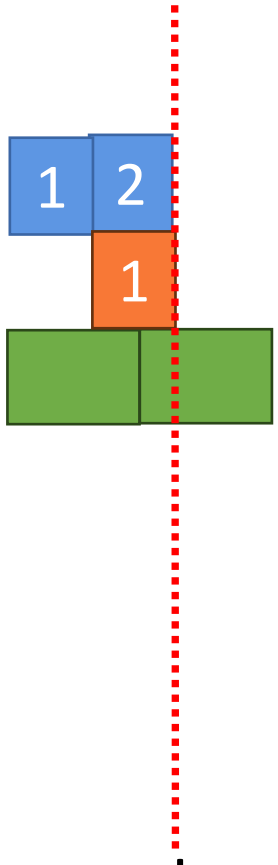
DERCA: Design Tradeoffs

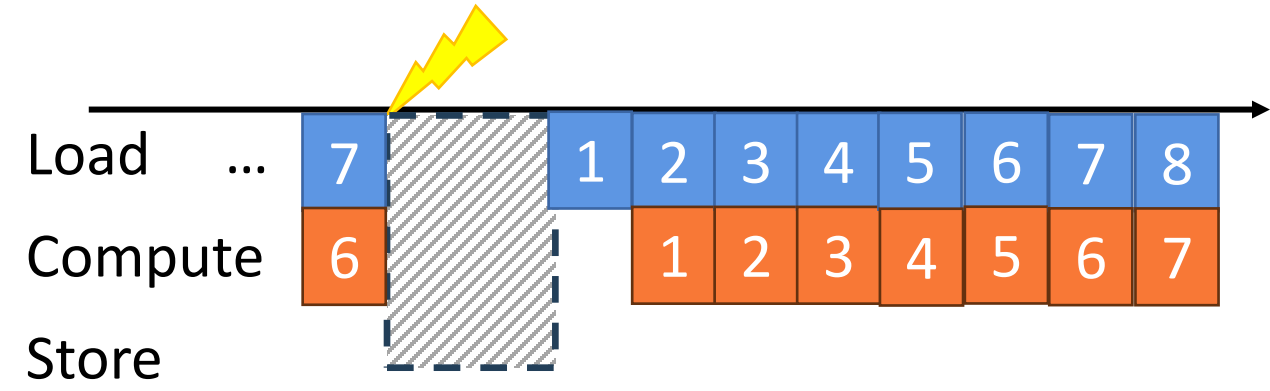


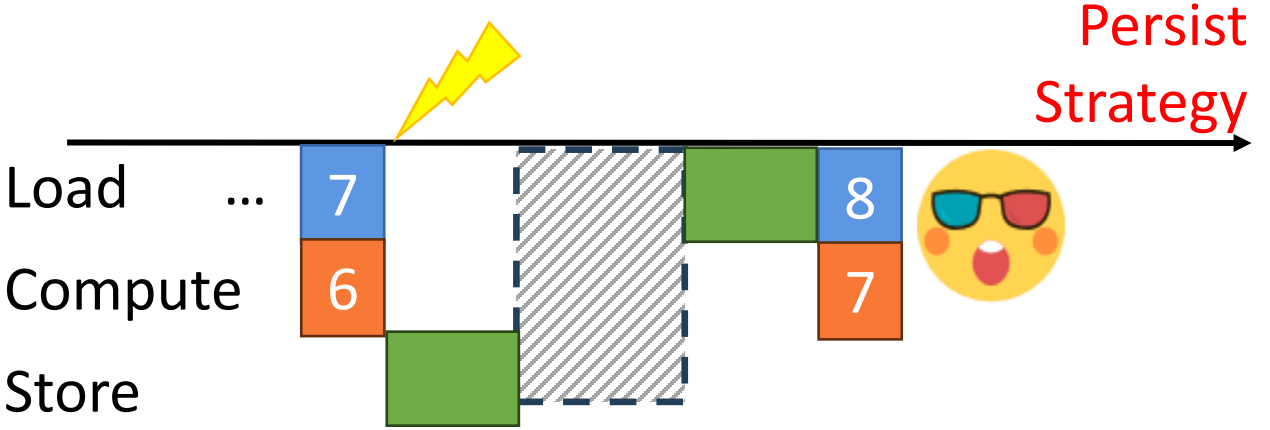
Small # tiles accumulated:

Store + Load > Recompute(1)

DERCA: Design Tradeoffs



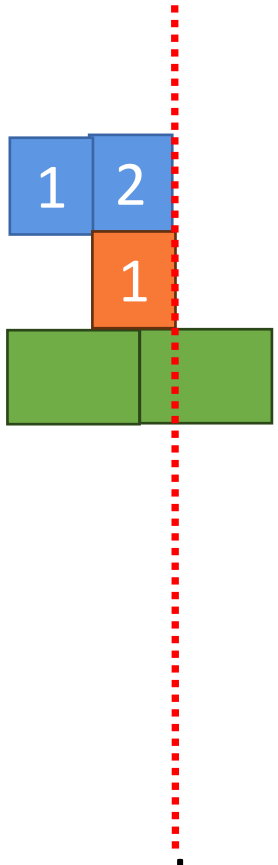




Small # tiles accumulated:

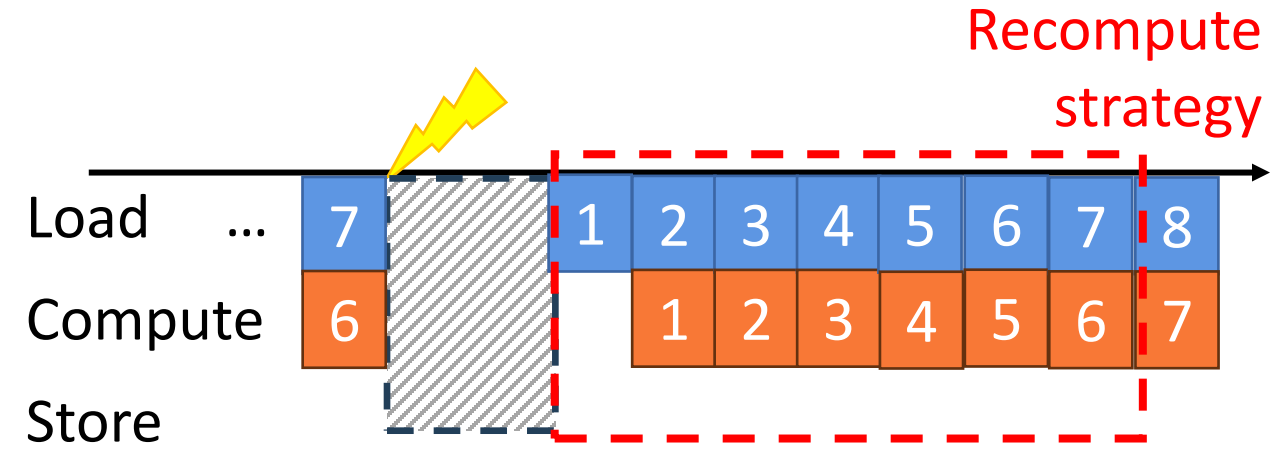
Store + Load > Recompute(1)

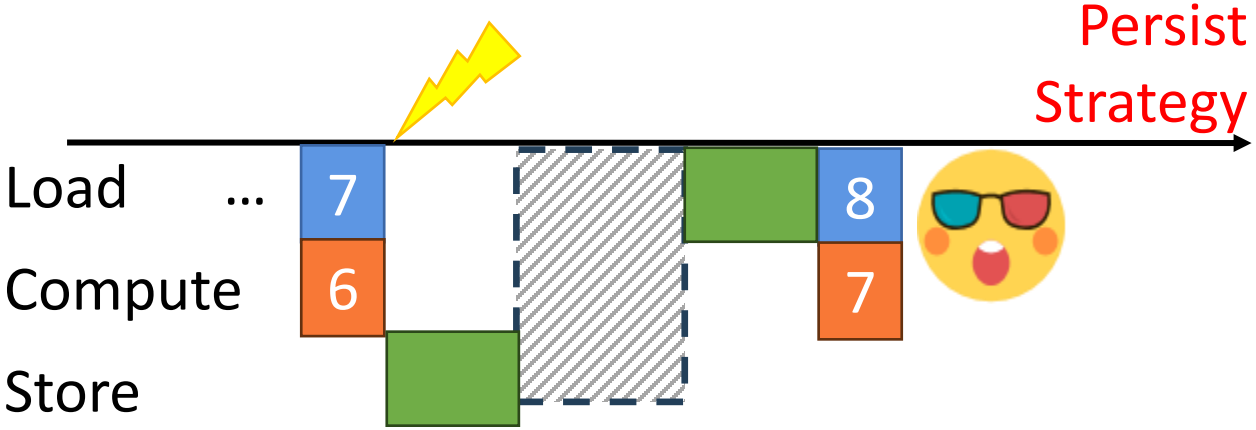
DERCA: Design Tradeoffs



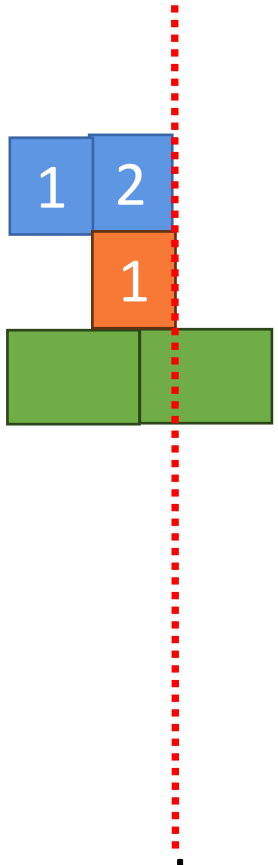
Small # tiles accumulated:

Store + Load > Recompute(1)



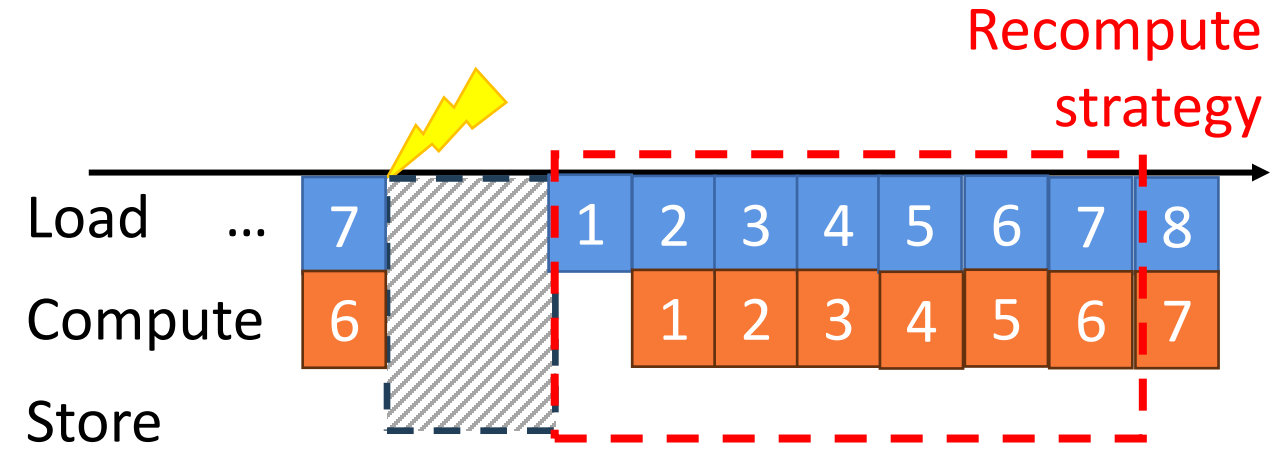


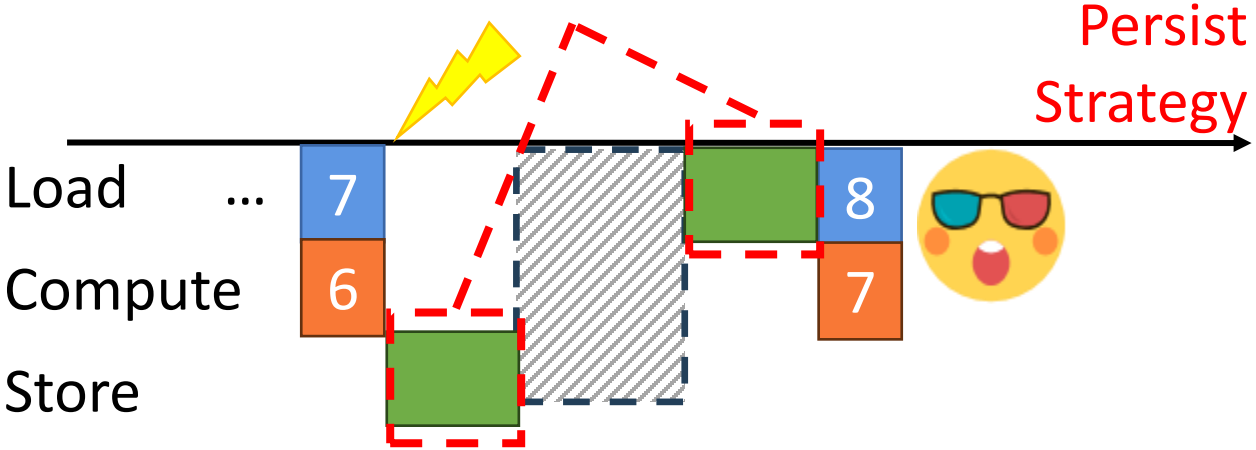
DERCA: Design Tradeoffs



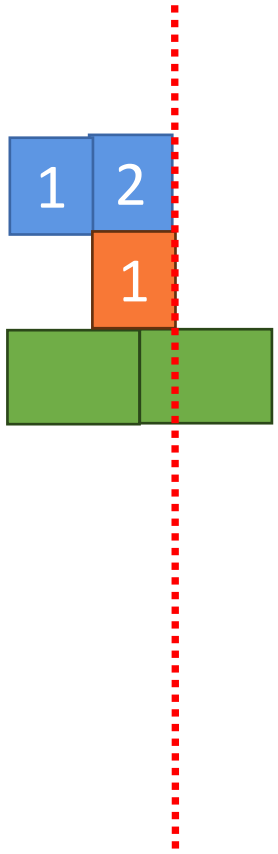
Small # tiles accumulated:

Store + Load > Recompute(1)



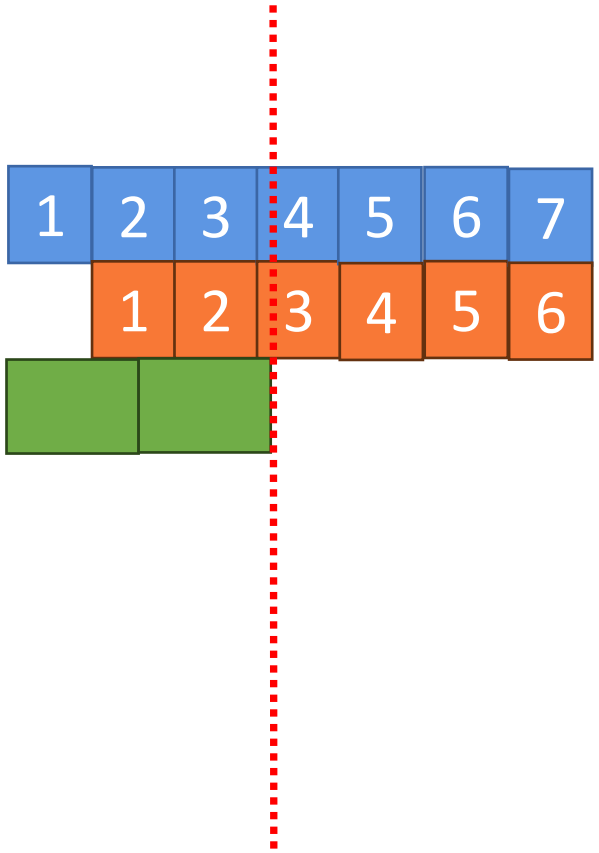


DERCA: Design Tradeoffs



Small # tiles accumulated:

Store + Load > Recompute(1)



Large # tiles accumulated:

Store + Load < Recompute(1-7)

DERCA: modeling and overhead optimization

...

[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

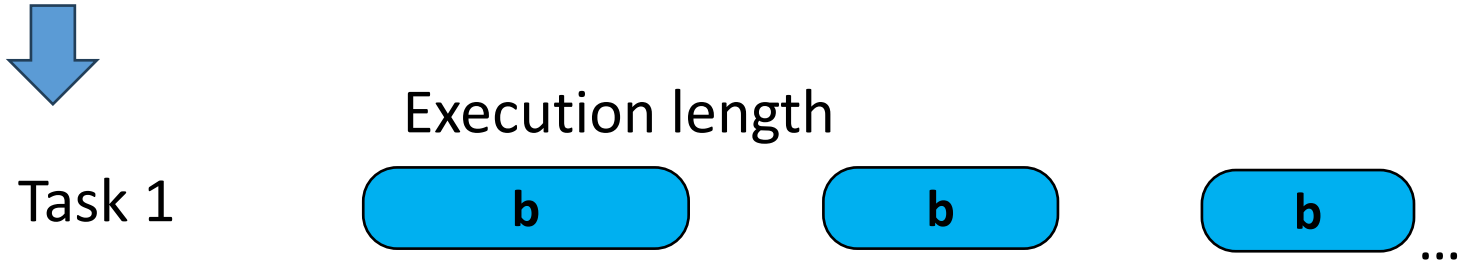
Hardware operations (load/comp/store/scheduling)

...

[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

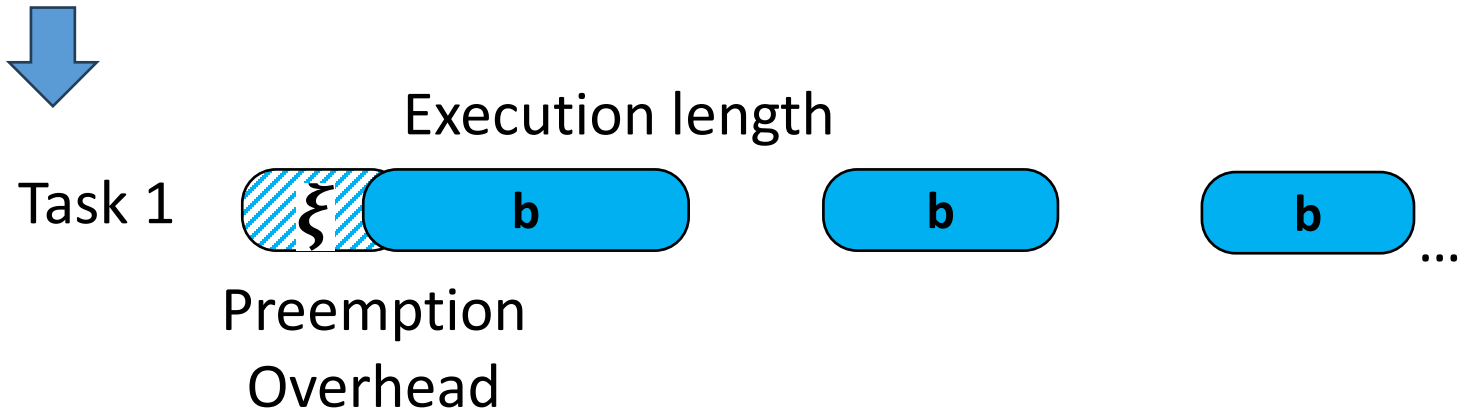
Hardware operations (load/comp/store/scheduling)



[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

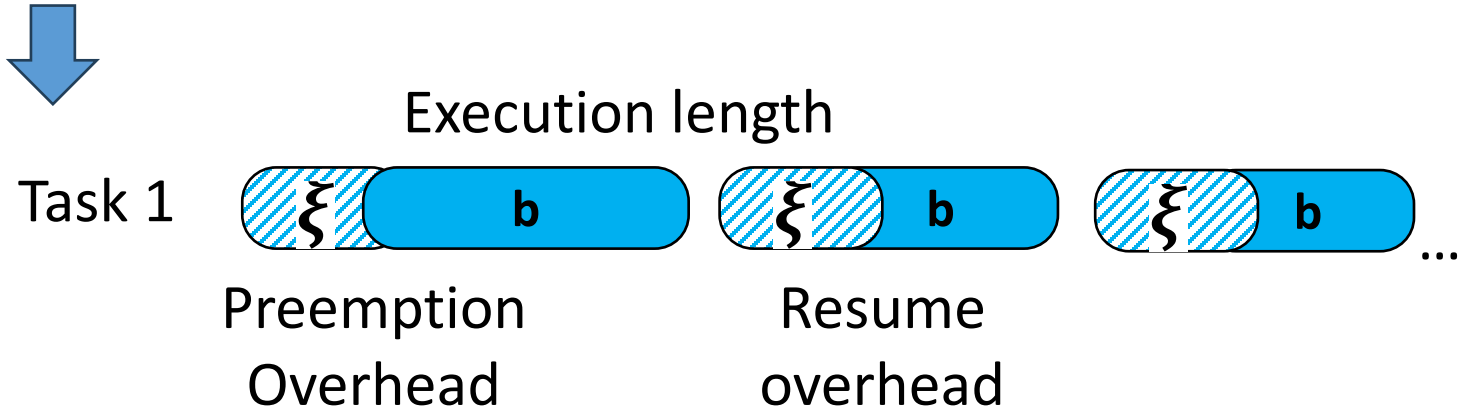
Hardware operations (load/comp/store/scheduling)



[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

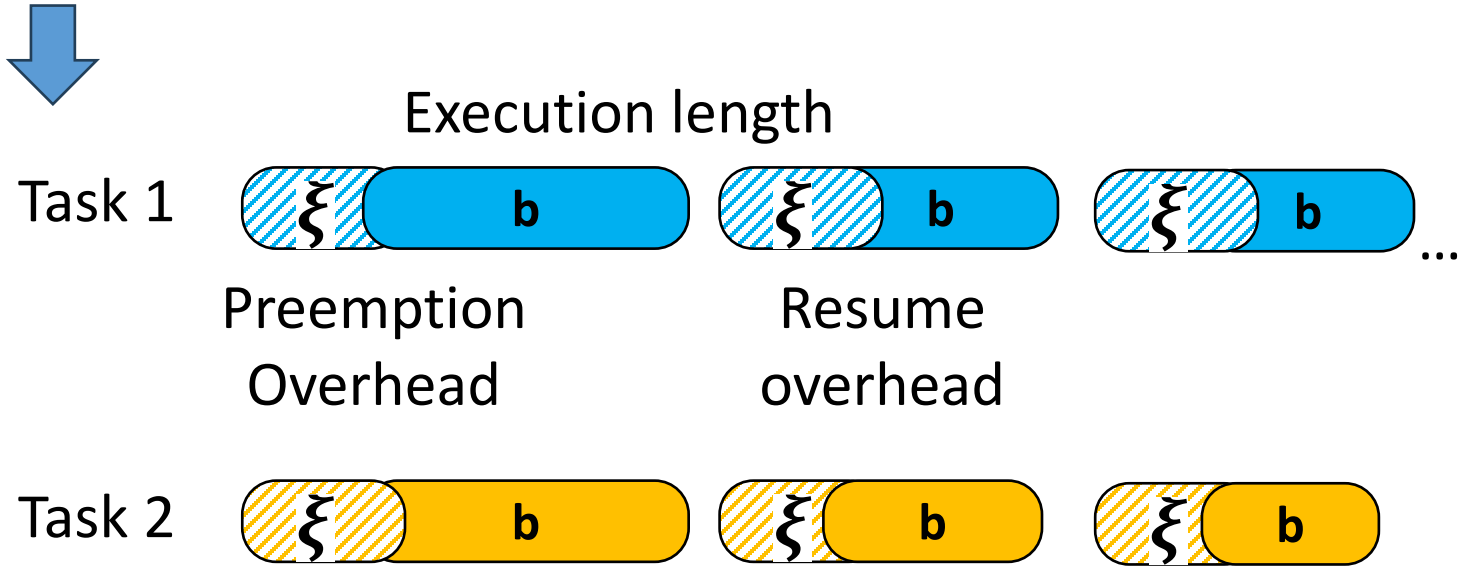
Hardware operations (load/comp/store/scheduling)



[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

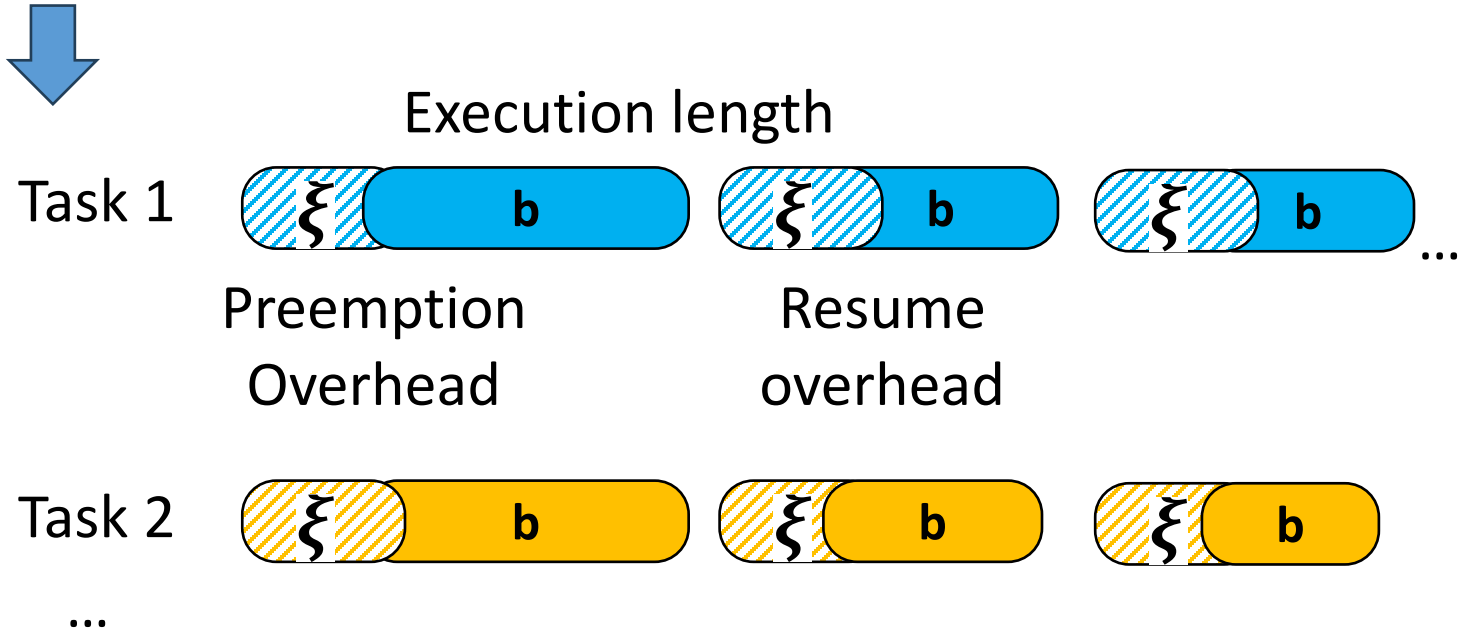
Hardware operations (load/comp/store/scheduling)



[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

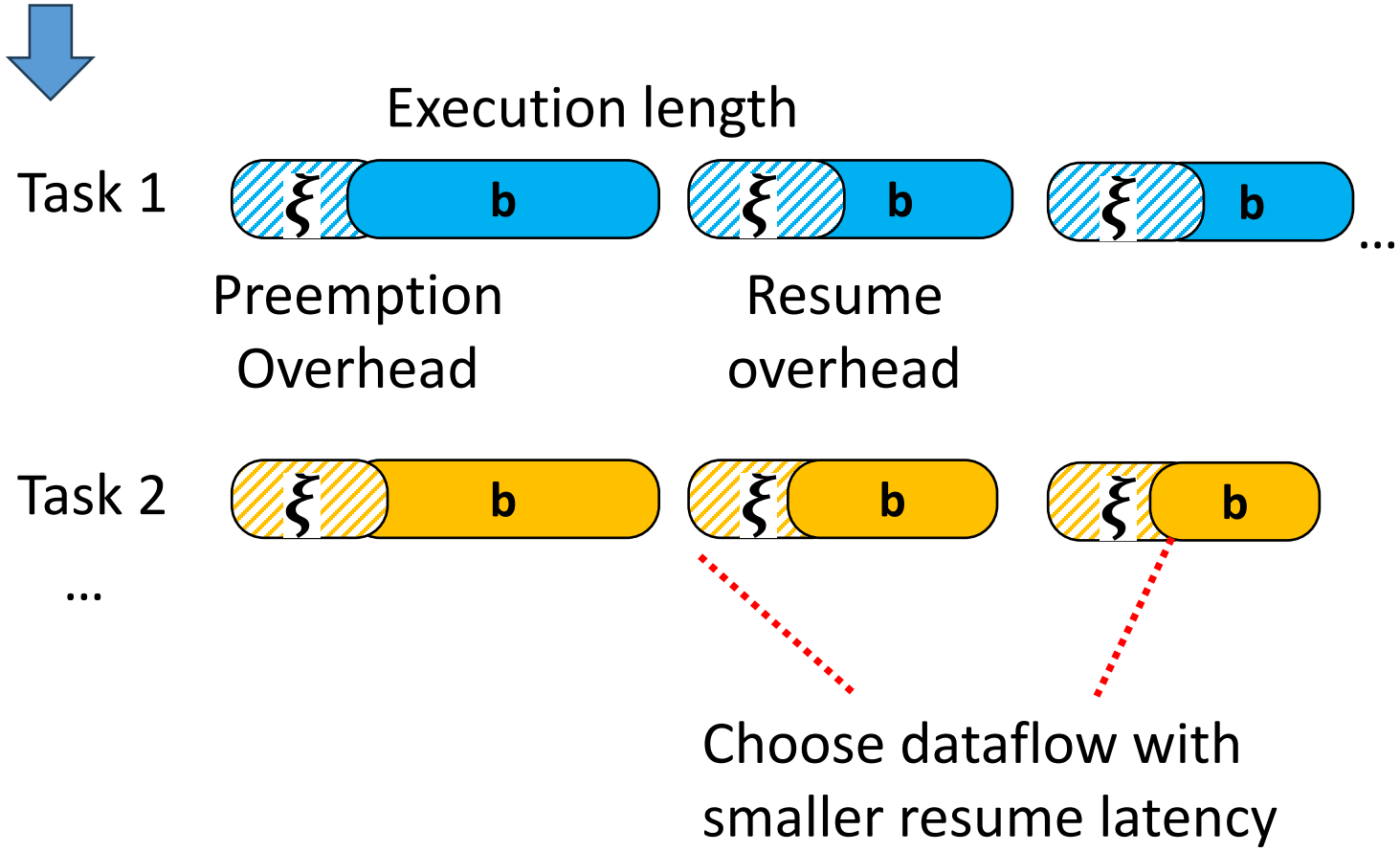
Hardware operations (load/comp/store/scheduling)



[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

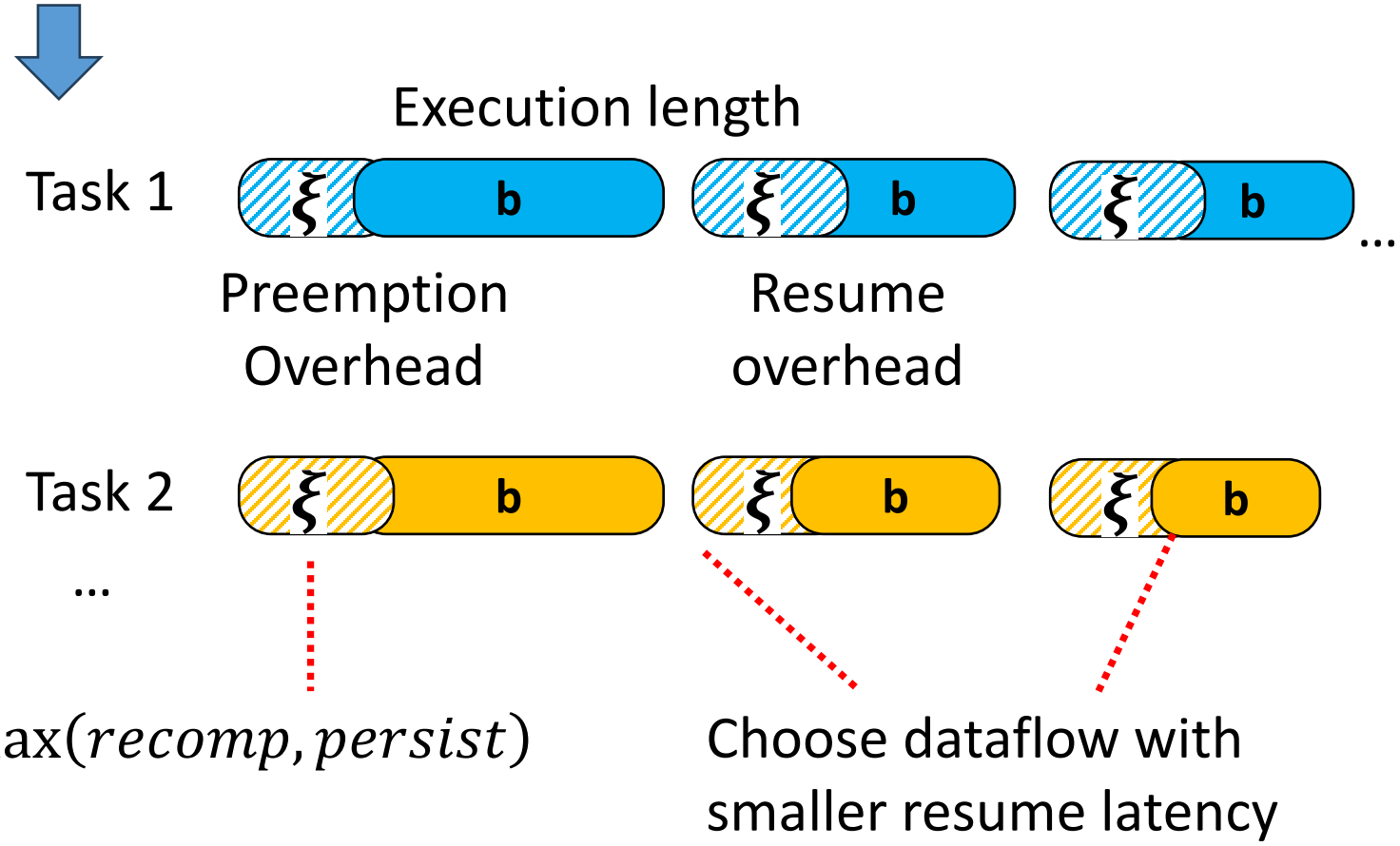
Hardware operations (load/comp/store/scheduling)



[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.
[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

Hardware operations (load/comp/store/scheduling)

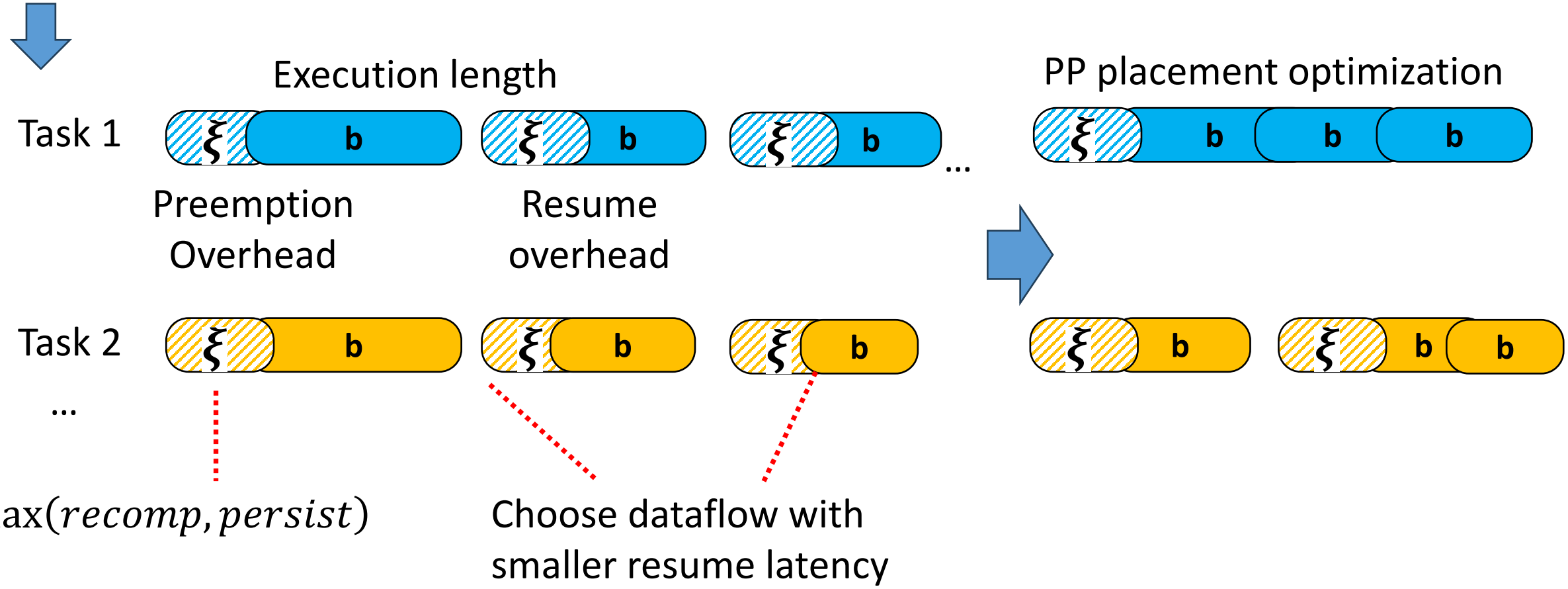


[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.

[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

Hardware operations (load/comp/store/scheduling)

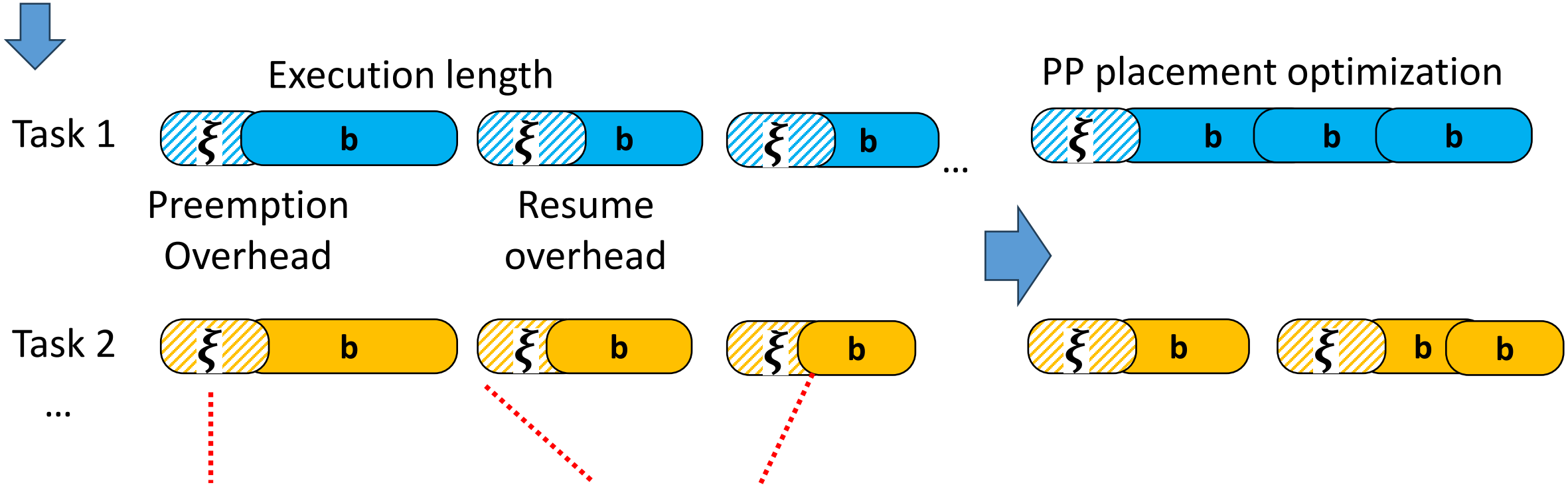


[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.

[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: modeling and overhead optimization

Hardware operations (load/comp/store/scheduling)



$\max(recomp, persist)$

Choose dataflow with smaller resume latency

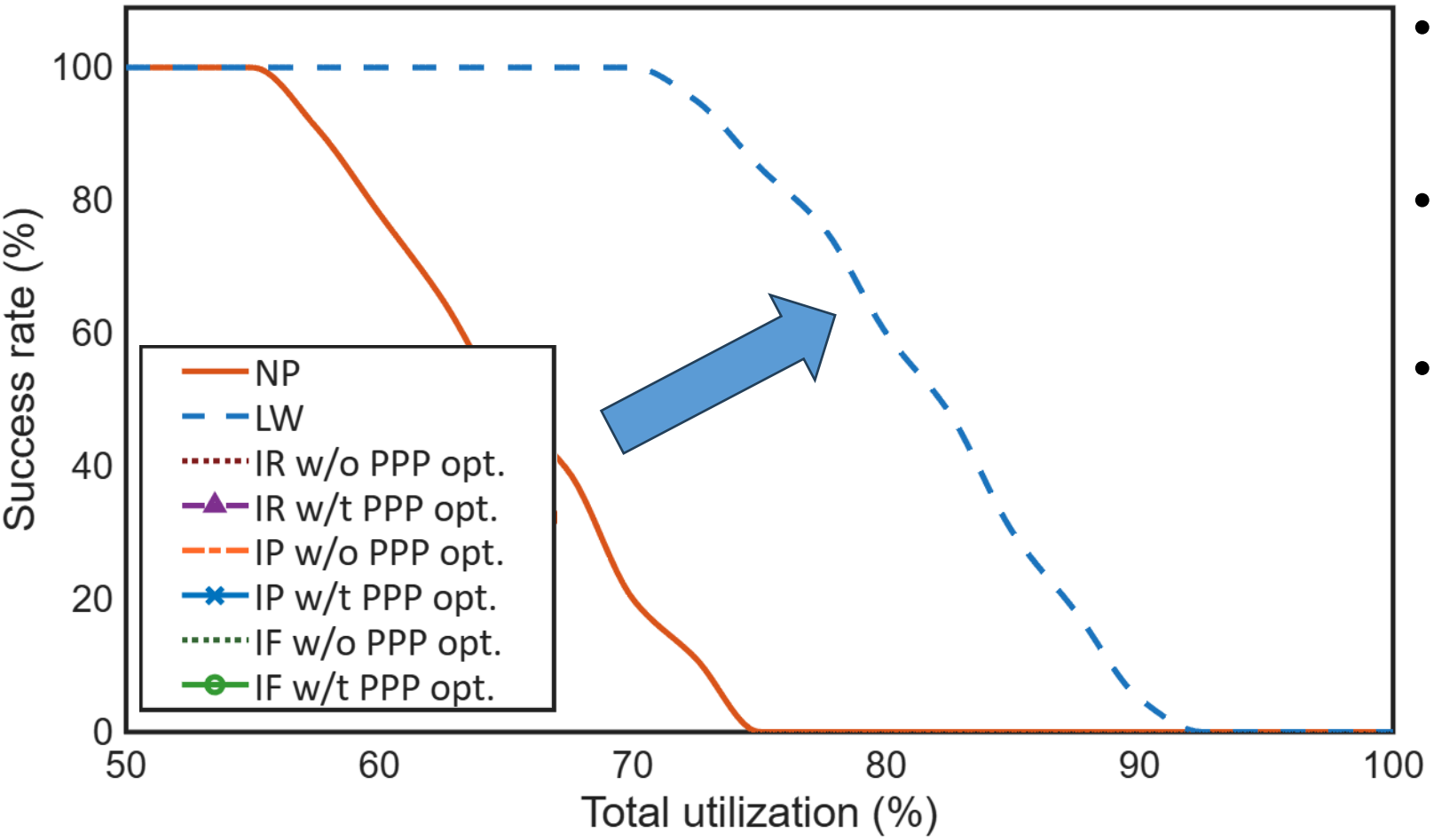
- **safe + tight estimation**
- **optimize for overhead**

[1] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito and M. Caccamo, "Preemption Points Placement for Sporadic Task Sets," *2010 22nd Euromicro Conference on Real-Time Systems*, Brussels, Belgium, 2010, pp. 251-260, doi: 10.1109/ECRTS.2010.9.

[2] Standaert, B., Raadia, F., Sudvarg, M., Baruah, S., Chantem, T., Fisher, N., & Gill, C. (2024). A Limited-Preemption Scheduling Model Inspired by Security Considerations

DERCA: Evaluation

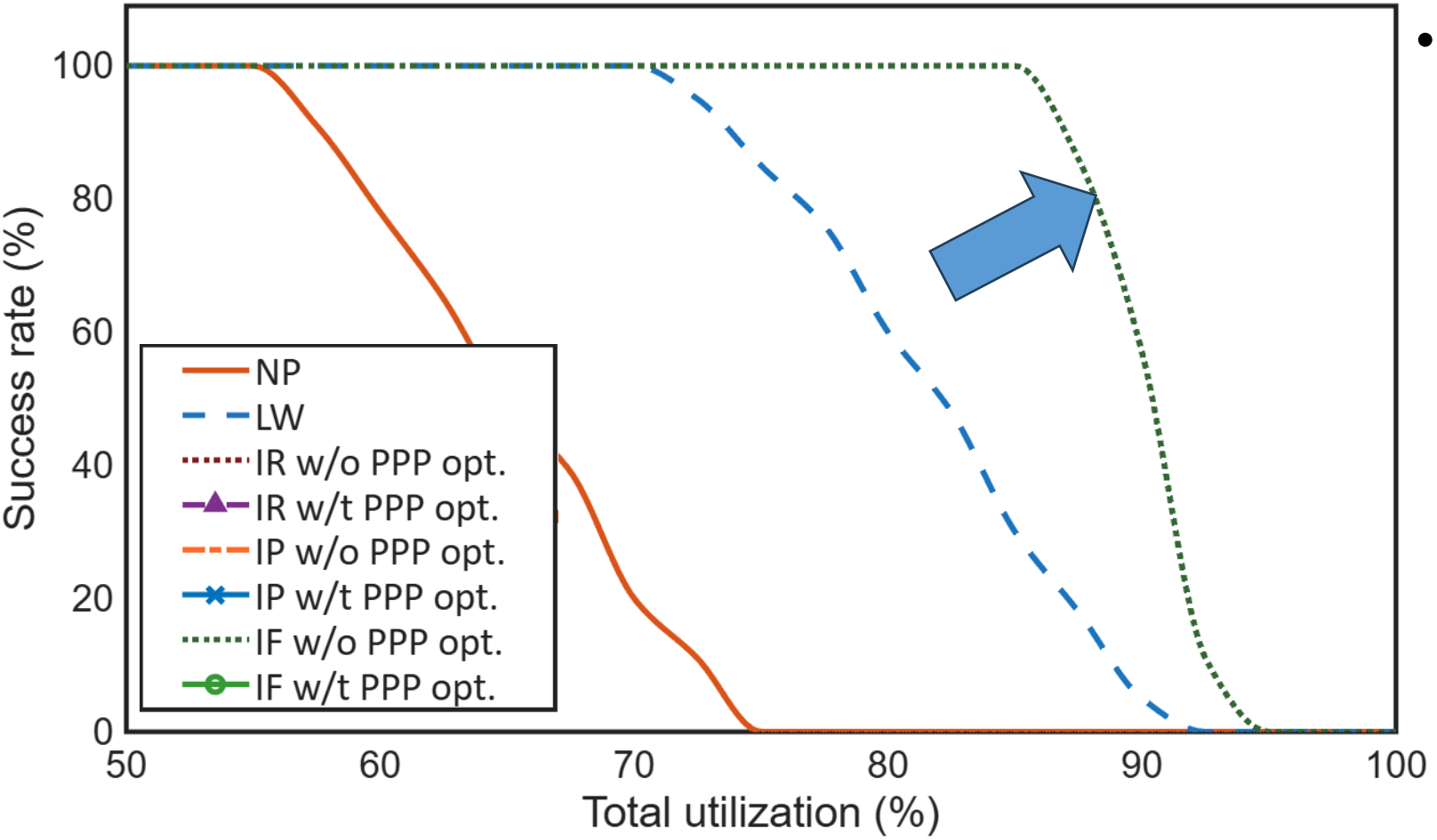
- Compare on synthetic workloads
- Task1: (1k x 8k x 1k), (1k x 8k x 1k)
- Task2: (1k x 8k x 1k), (1k x 8k x 1k)



- Randomly generate periods for each task to get a taskset
- Higher success rate → better schedulability
- Baselines: Low success rate

DERCA: Evaluation

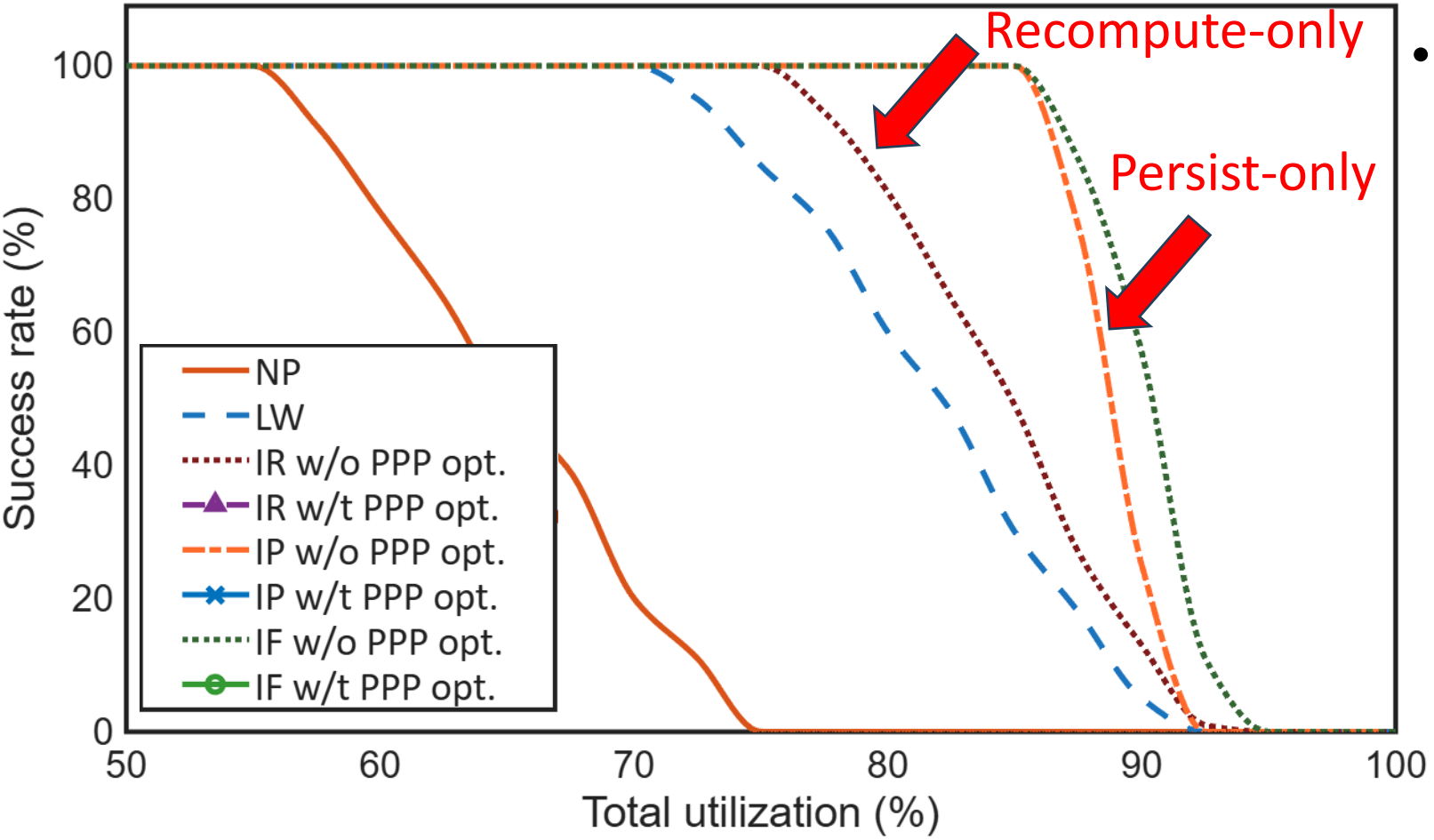
- Compare on synthetic workloads
- Task1: (1k x 8k x 1k), (1k x 8k x 1k)
- Task2: (1k x 8k x 1k), (1k x 8k x 1k)



- Intra-layer flexible dataflow: best schedulability

DERCA: Evaluation

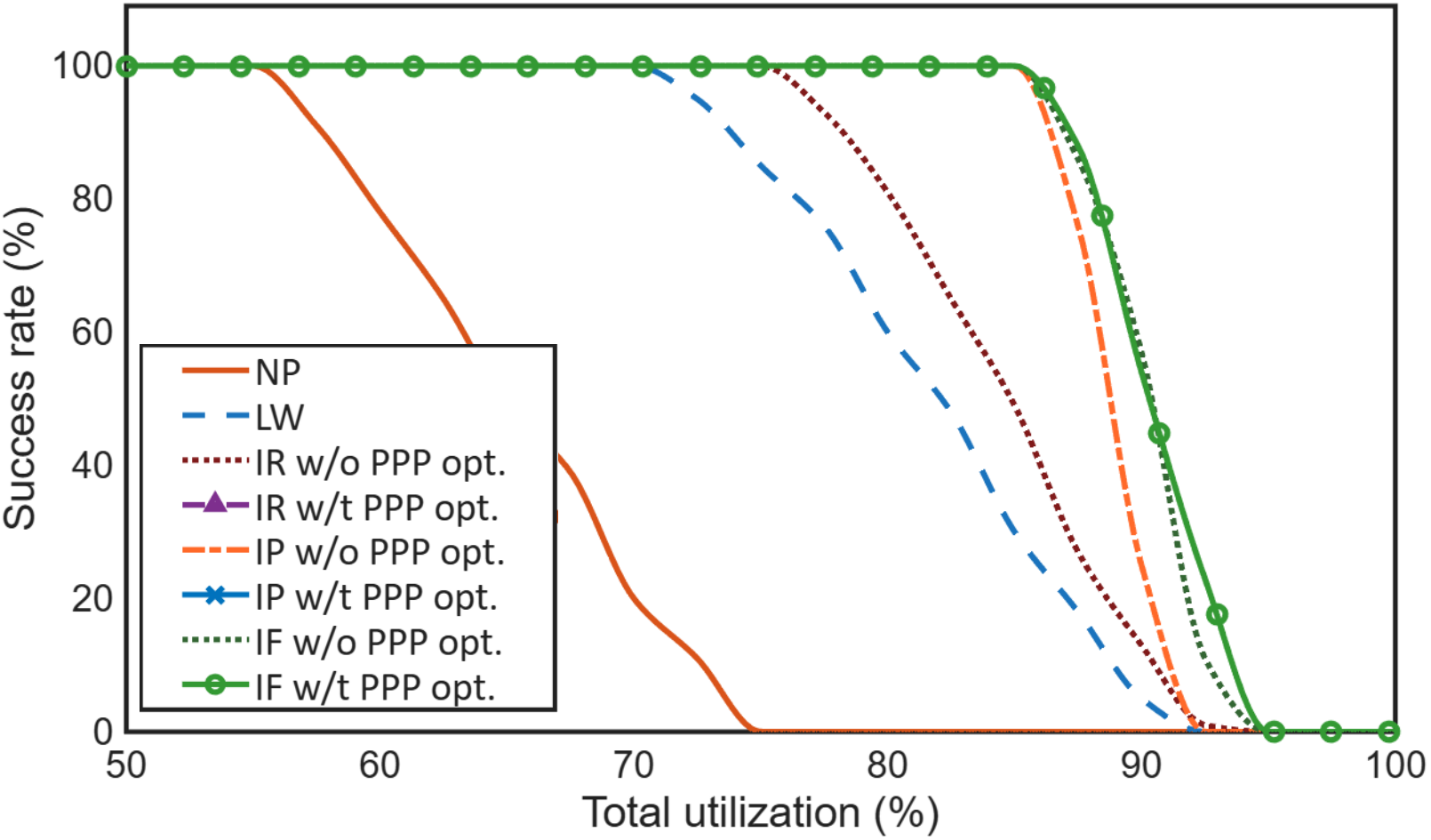
- Compare on synthetic workloads
- Task1: (1k x 8k x 1k), (1k x 8k x 1k)
- Task2: (1k x 8k x 1k), (1k x 8k x 1k)



- only one strategies → high overhead, worse schedulability

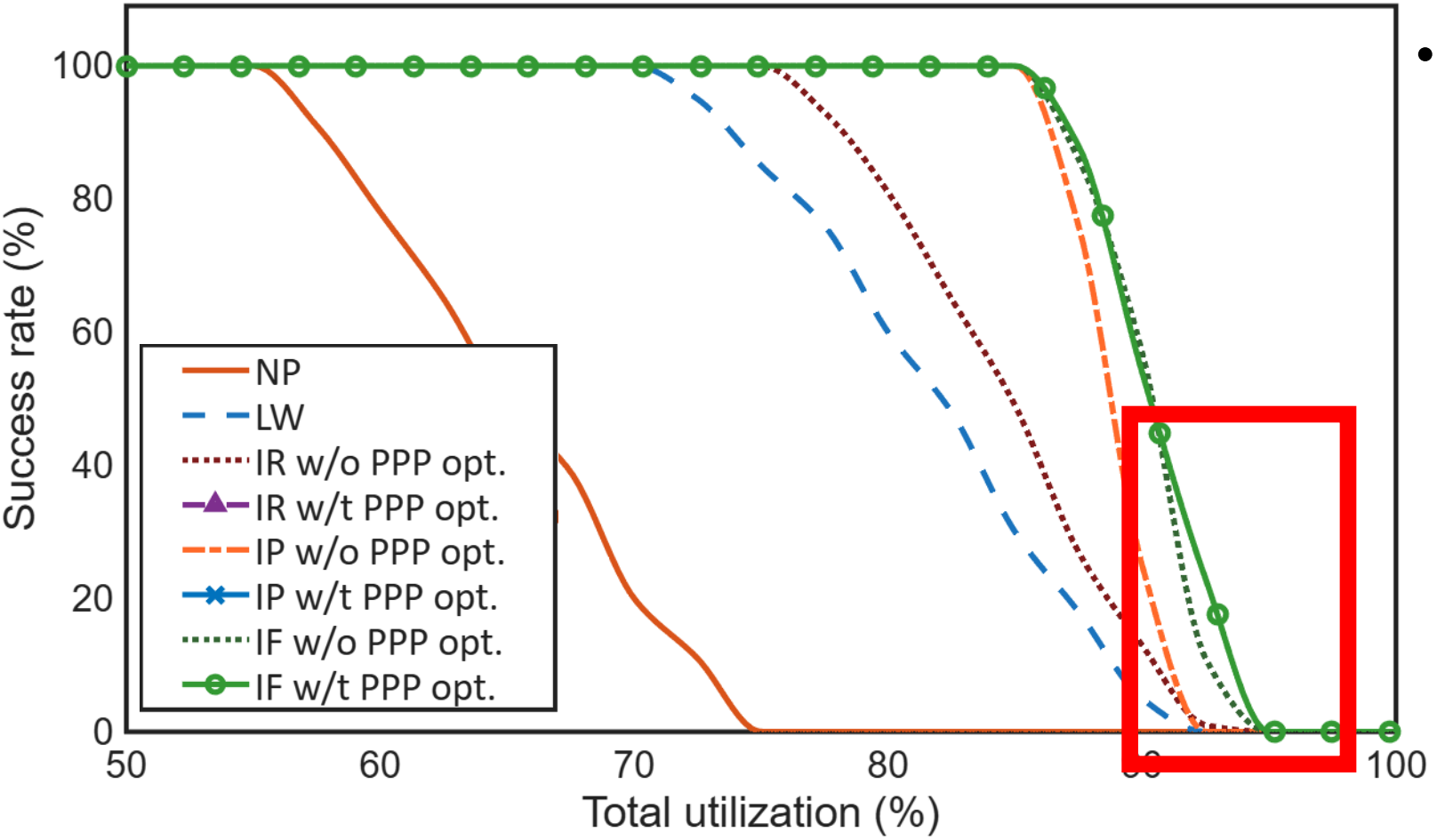
DERCA: Evaluation

- Compare on synthetic workloads
- Task1: (1k x 8k x 1k), (1k x 8k x 1k)
- Task2: (1k x 8k x 1k), (1k x 8k x 1k)



DERCA: Evaluation

- Compare on synthetic workloads
- Task1: (1k x 8k x 1k), (1k x 8k x 1k)
- Task2: (1k x 8k x 1k), (1k x 8k x 1k)

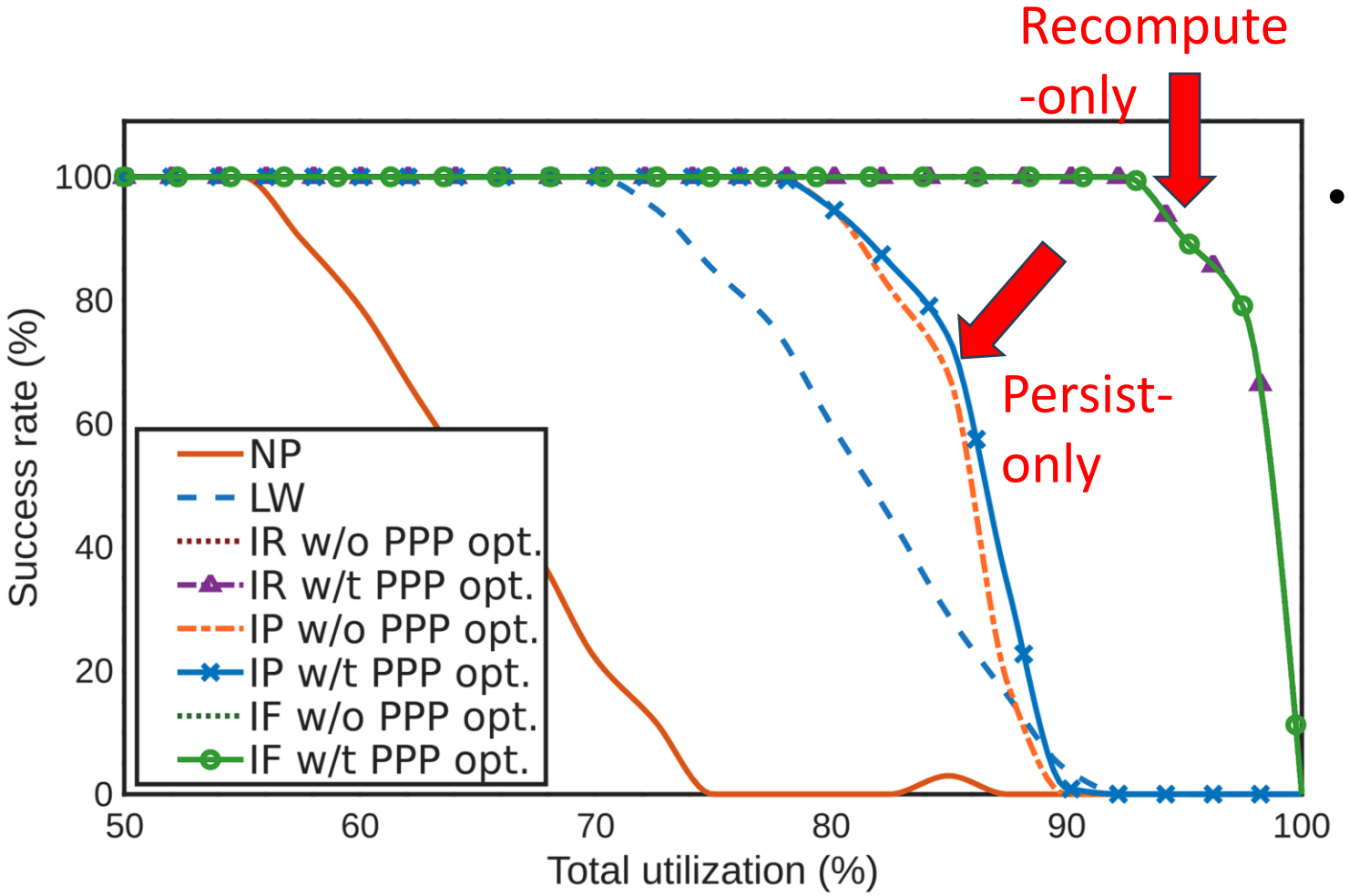


- PP placement optimization → further reduce overhead

DERCA: Evaluation

- Success rates on other workloads

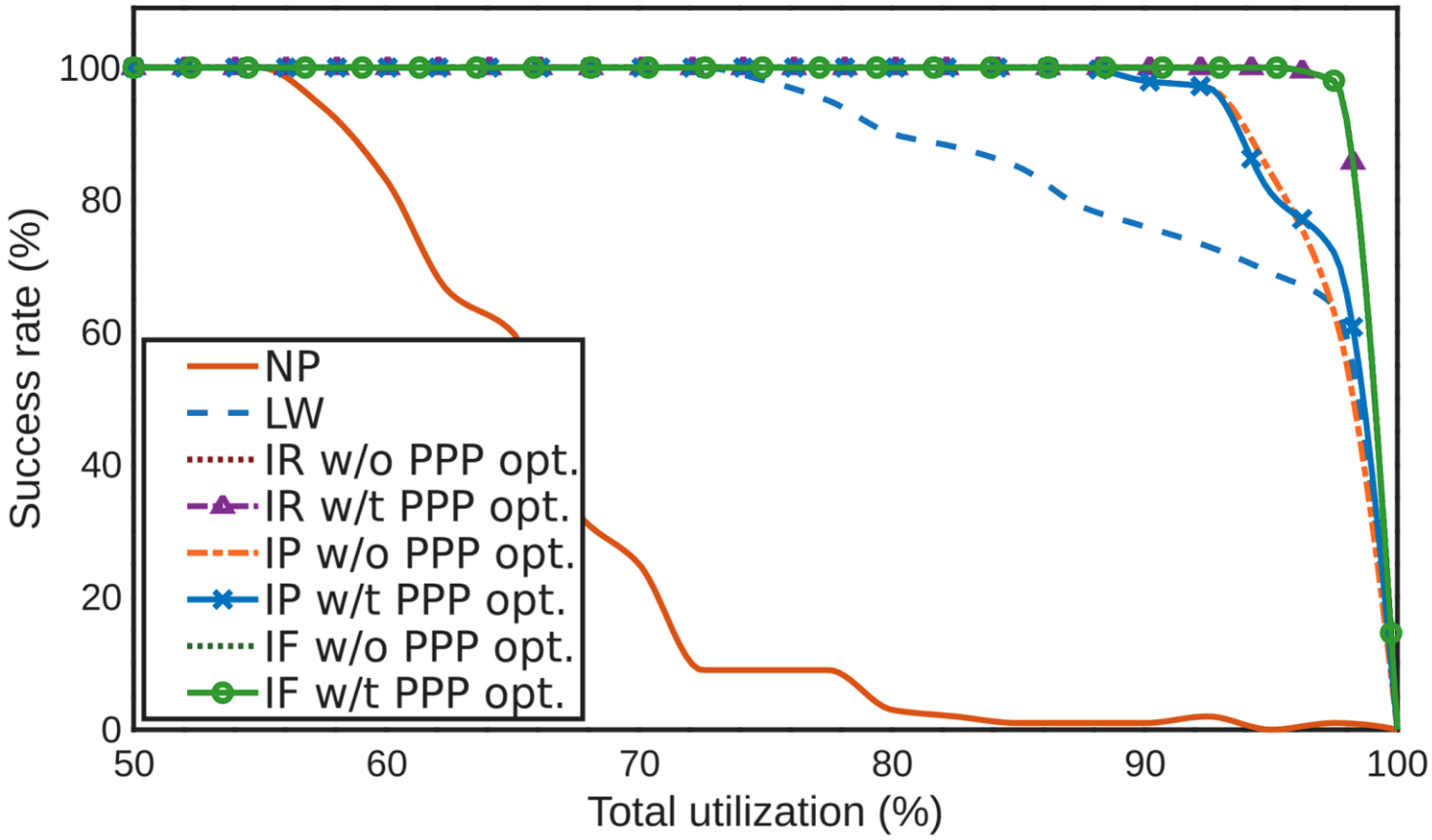
- Task1: (2k x 128 x 2k), (2k x 128 x 2k)
- Task2: (2k x 128 x 2k), (2k x 128 x 2k)



- Small reduced dimension (K) → Recompute is better

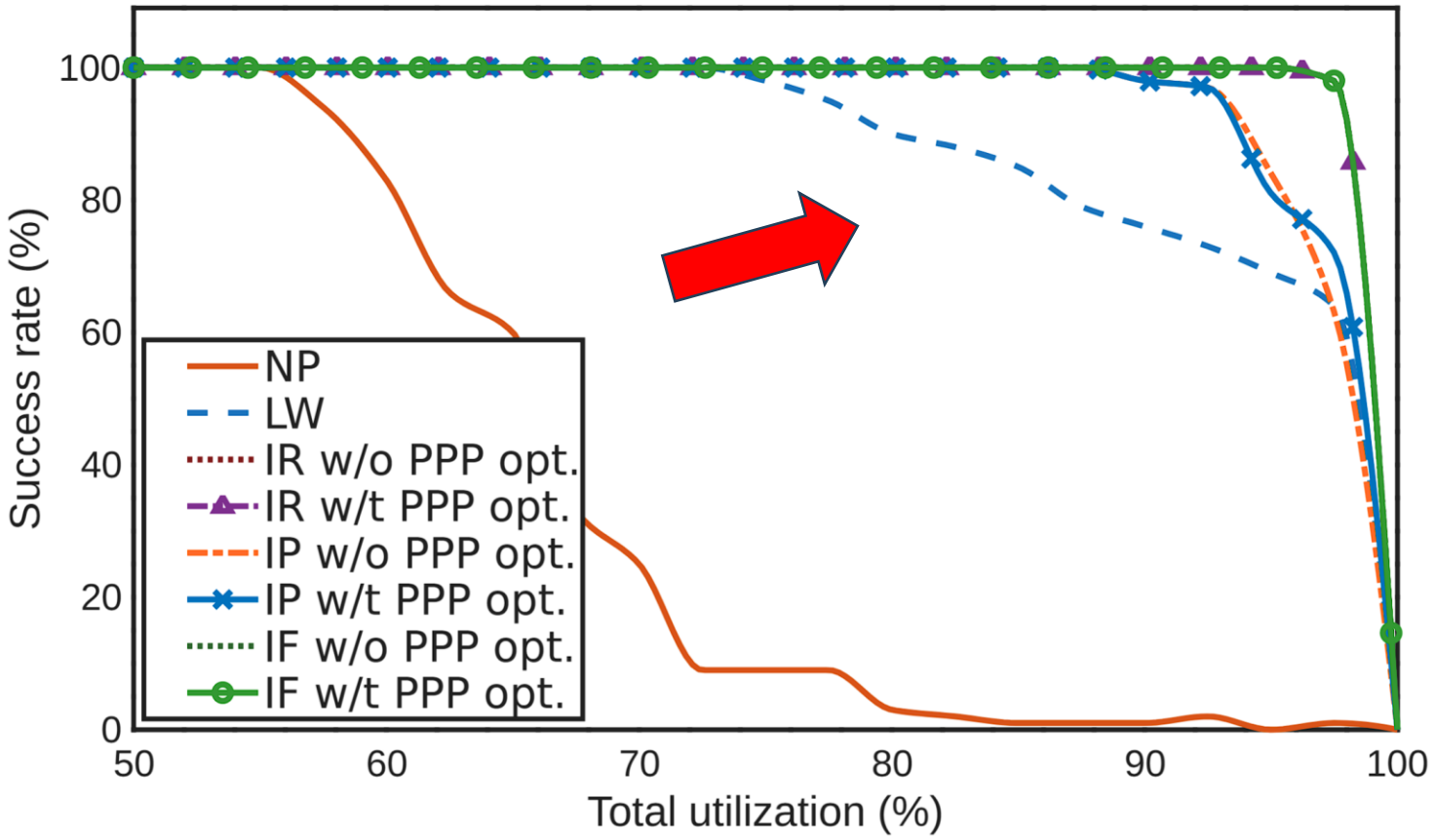
DERCA: Evaluation

- Success rates on other workloads
- Realistic workloads: DeiT-T, BERT-tiny,BERT-mini, PointNet, and MLP-Mixer



DERCA: Evaluation

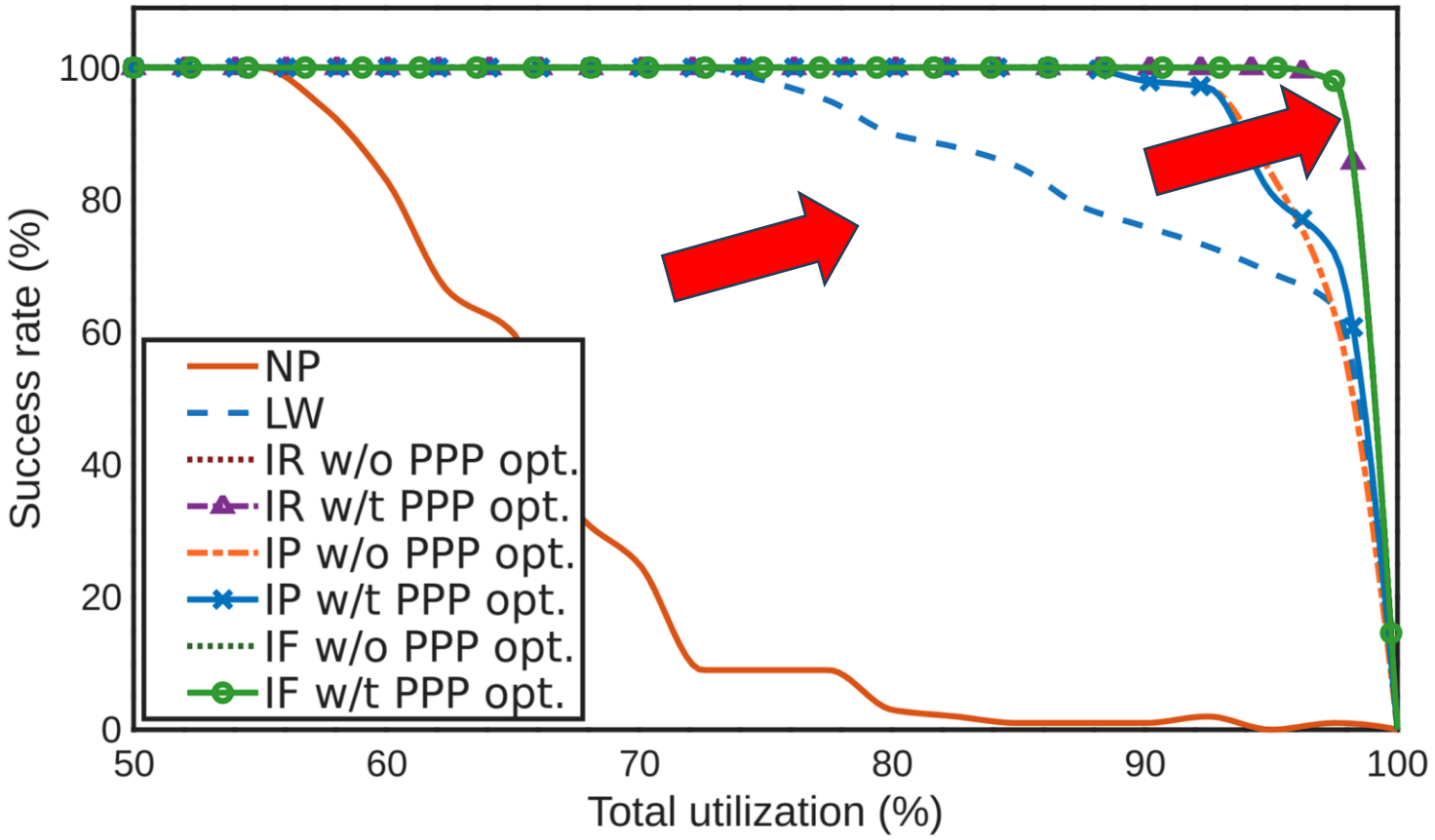
- Success rates on other workloads
- Realistic workloads: DeiT-T, BERT-tiny,BERT-mini, PointNet, and MLP-Mixer



- More layers → layerwise gets better than synthetic workloads

DERCA: Evaluation

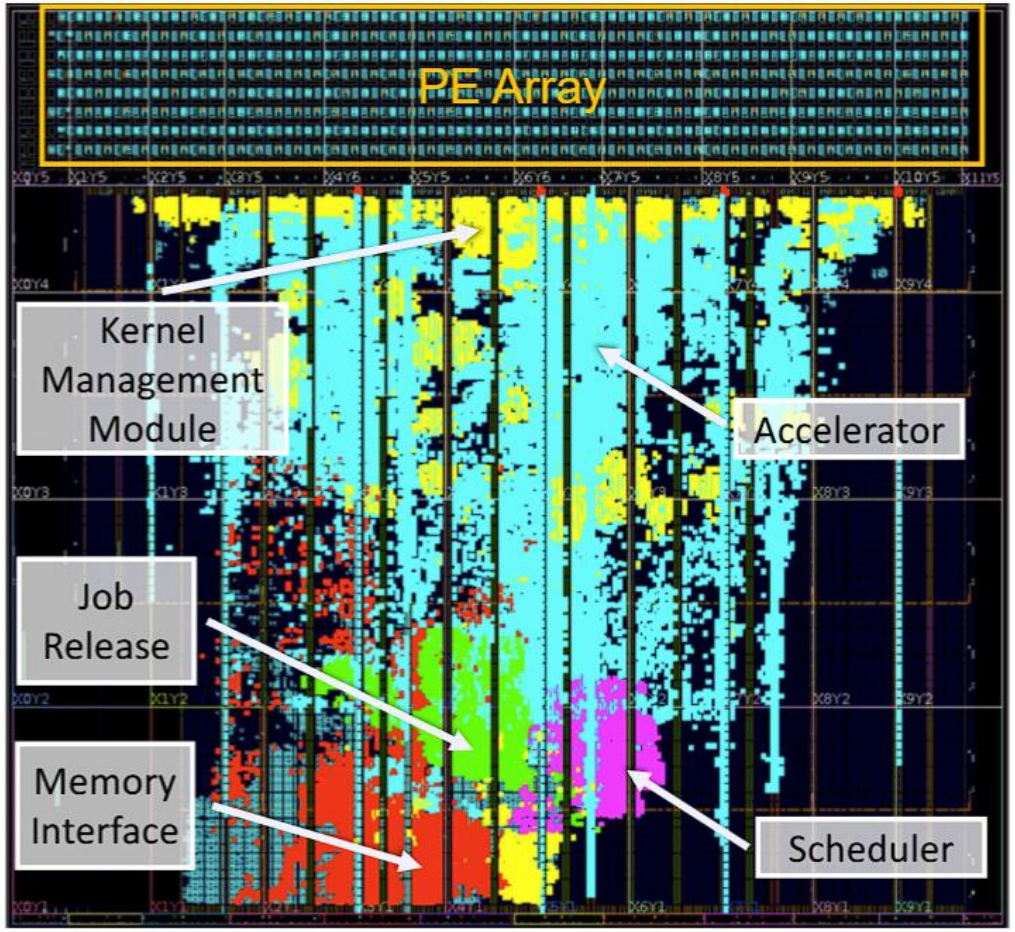
- Success rates on other workloads
- Realistic workloads: DeiT-T, BERT-tiny,BERT-mini, PointNet, and MLP-Mixer



- More layers → layerwise gets better than synthetic workloads
- Still, DERCA flexible dataflow has a better success rate, especially in high utilizations

DERCA: Evaluation

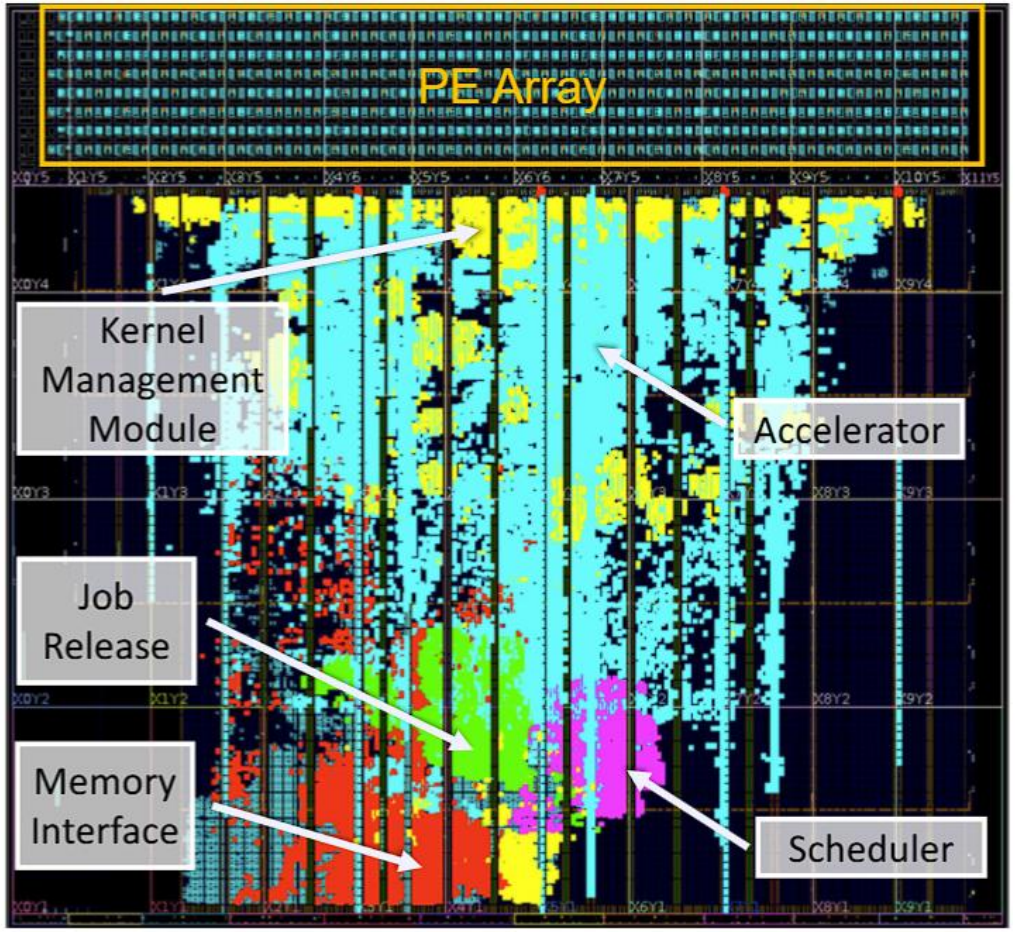
- Prototype on AMD VCK190 platform
- CHARM as baseline accelerator for PE array



DERCA: Evaluation

	LUT	REG	BRAM	URAM	DSP	AIE
Scheduler	9177 (1.02%)	8132 (0.45%)	1 (0.10%)	0 (0%)	0 (0%)	0 (0%)
Job Release	15135 (1.68%)	32646 (1.81%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Memory Interface	14517 (1.61%)	23218 (1.29)	0 (0%)	0 (0%)	37 (1.88%)	0 (0%)
Kernel Management	11898 (1.32%)	25053 (1.39%)	6.5 (0.67%)	0 (0%)	1 (0.05%)	0 (0%)
Accelerator	105060 (11.68%)	112324 (6.24%)	787.5 (81.44%)	384 (82.84%)	102 (5.18%)	384 (96.00%)
Total	155787 (17.31%)	201373 (11.19%)	795 (82.21%)	384 (82.94%)	140 (7.11%)	384 (96.00%)

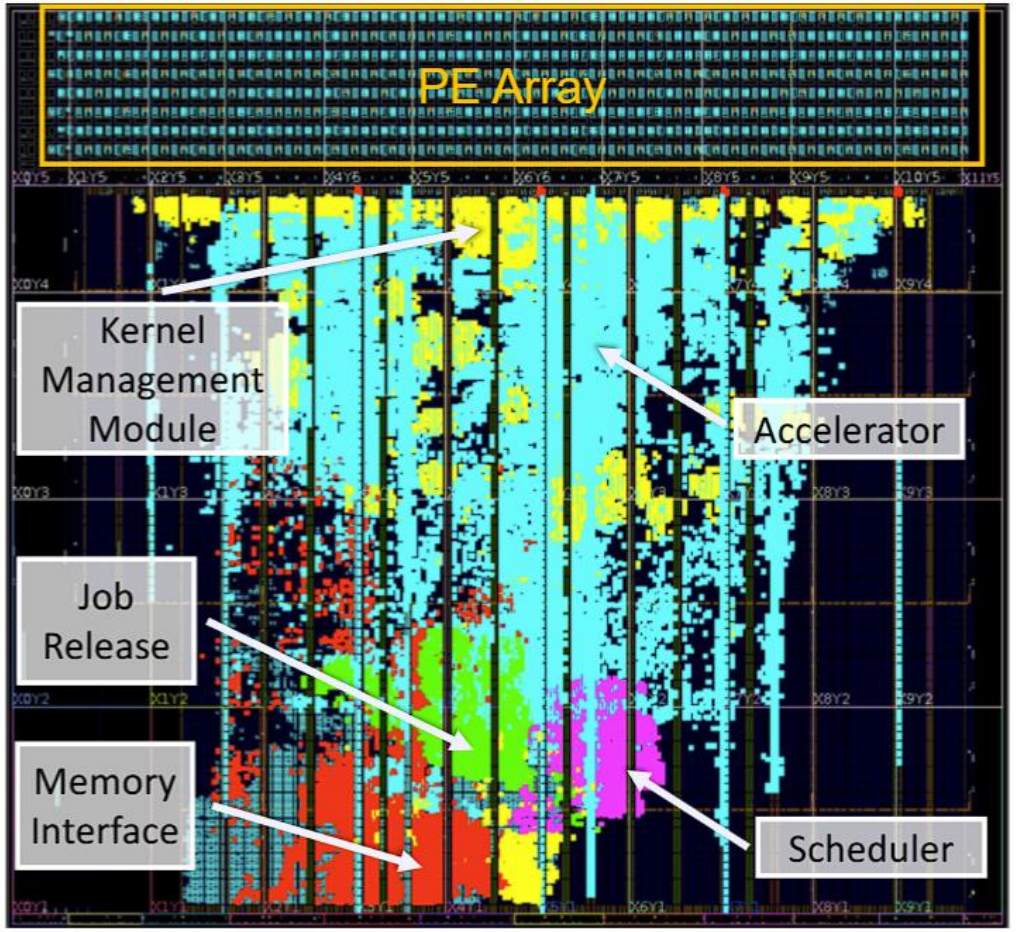
- Prototype on AMD VCK190 platform
- CHARM as baseline accelerator for PE array



DERCA: Evaluation

	LUT	REG	BRAM	URAM	DSP	AIE
Scheduler	9177 (1.02%)	8132 (0.45%)	1 (0.10%)	0 (0%)	0 (0%)	0 (0%)
Job Release	15135 (1.68%)	32646 (1.81%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Memory Interface	14517 (1.61%)	23218 (1.29)	0 (0%)	0 (0%)	37 (1.88%)	0 (0%)
Kernel Management	11898 (1.32%)	25053 (1.39%)	6.5 (0.67%)	0 (0%)	1 (0.05%)	0 (0%)
Accelerator	105060 (11.68%)	112324 (6.24%)	787.5 (81.44%)	384 (82.84%)	102 (5.18%)	384 (96.00%)
Total	155787 (17.31%)	201373 (11.19%)	795 (82.21%)	384 (82.94%)	140 (7.11%)	384 (96.00%)

- Prototype on AMD VCK190 platform
- CHARM as baseline accelerator for PE array
- DERCA is lightweight: <5% additional resource usage



Thank You

DERCA: DetERministic Cycle-level Accelerator on Reconfigurable Platforms in DNN-Enabled Real-Time Safety-Critical Systems

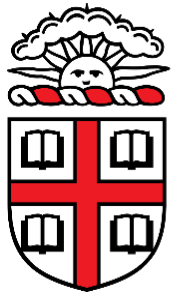
Shixin Ji*, Zhuoping Yang*, Xingzhen Chen*, Wei Zhang*, Jinming Zhuang*,

Alex K. Jones§, Zheng Dong†, Peipei Zhou*

Brown University*; Wayne State University† ; Syracuse University§

shixin_ji@brown.edu
peipei_zhou@brown.edu

https://peipeizhou-eecs.github.io/



BROWN

WAYNE STATE UNIVERSITY

National Science Foundation

Syracuse University

AMD
XILINX