USENIX ATC '25

Accelerating Model Training on Ascend Chips: An Industrial System for Profiling, Analysis and Optimization

Yuhang Zhou[1], **Zibo Wang**[1], Zhibin Wang[1], Ruyi Zhang[1], Chen Tian[1], Xiaoliang Wang[1], Wanchun Dou[1], Guihai Chen[1], Bingqiang Wang[2], Yonghong Tian[2], Yan Zhang[2], Hui Wang[2], Fuchun Wei[3], Boquan Sun[3], Jingyi Zhang[3], Bin She[3], Teng Su[3], Yifan Yao[3], Chunsheng Li[3], Ziyang Zhang[3], Yaoyuan Wang[3], Bin Zhou[4], Guyue Liu[5]

[1] Nanjing University [2] Peng Cheng Laboratory [3] Huawei [4] Shandong University [5] Peking University

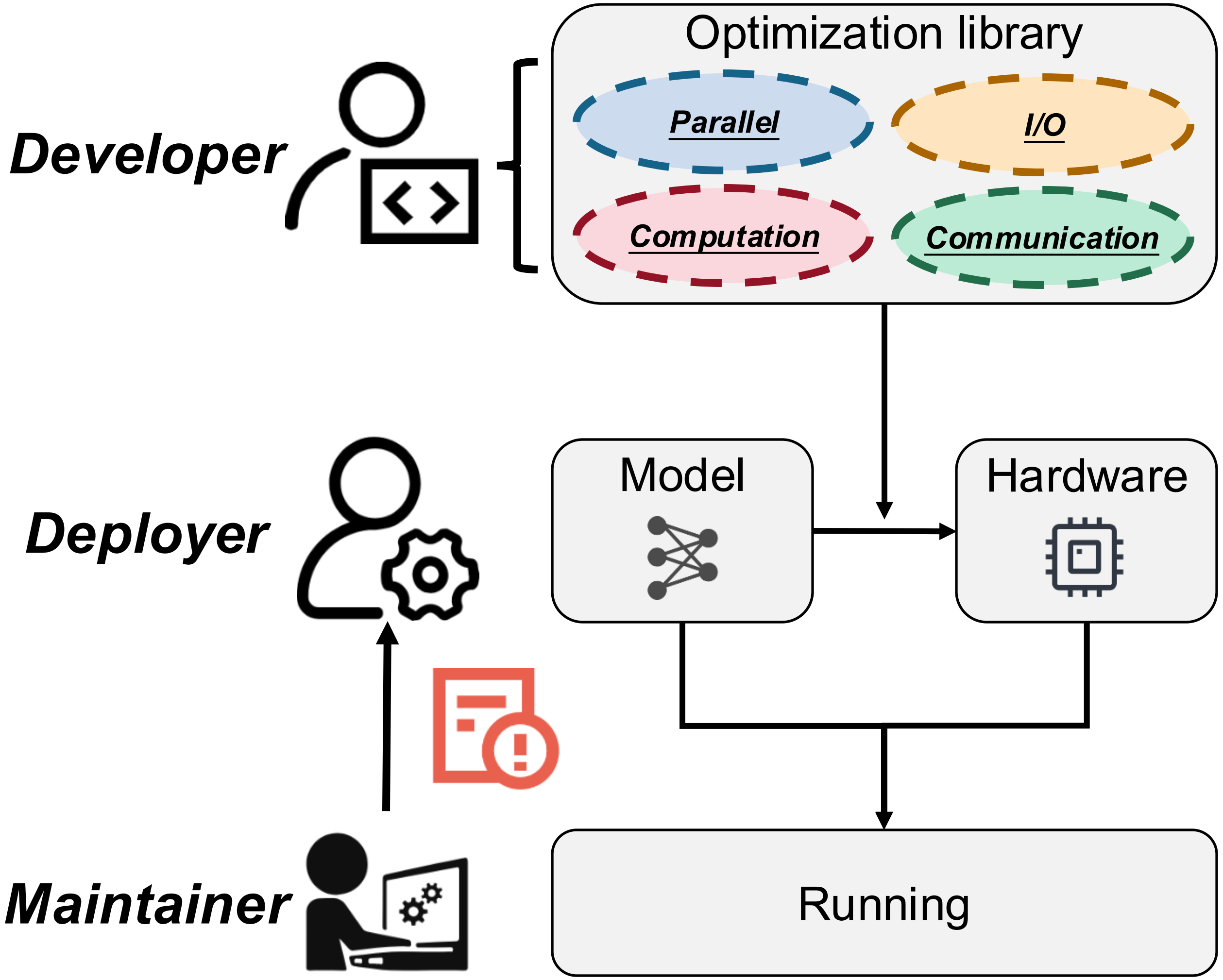
Outline

- Introduction
- Insights
- System Design
- Case Study
- Conclusion

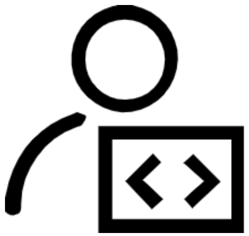
Outline

- Introduction
- Insights
- System Design
- Case Study
- Conclusion

Different Roles in Model Training



Developer



Optimization library

Parallel
DP/PP/TP/SP
Auto Parallel: Alpa, …

I/O
CPU processing: Pecan, …
Cache strategy: UGACHE, …
Data prefetch: Fastensor, …

Computation
Operator fusion: T3, …
Memory management: GMLake, …
Compiler： Cocktailer, …

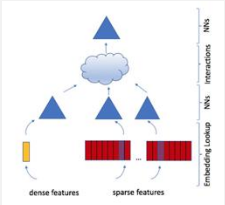
Communication
Scheduler: Syndicate, …
Topology： TopoOpt, …
Collectives： TCCL, …

Identify bottlenecks and develop optimizations

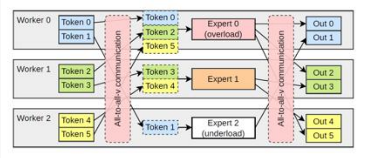
Deployer

Model

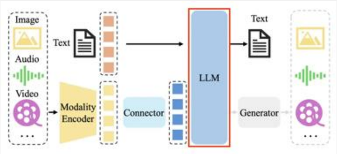
Transformer



DLRM

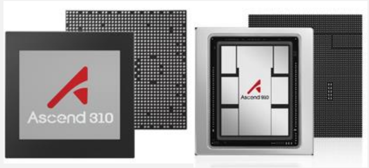


MoE

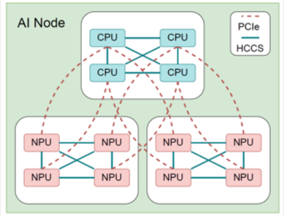


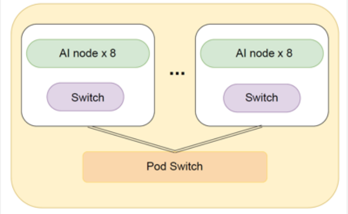
Multimodal

Hardware



GPU vs NPU



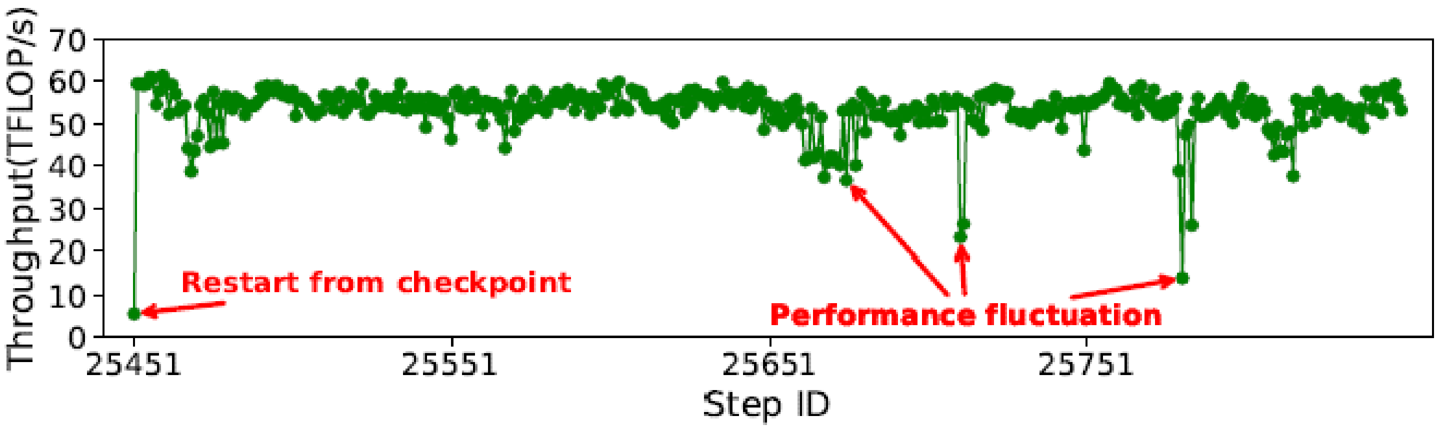


Single node vs Multi node

Select optimization for varying models and hardware

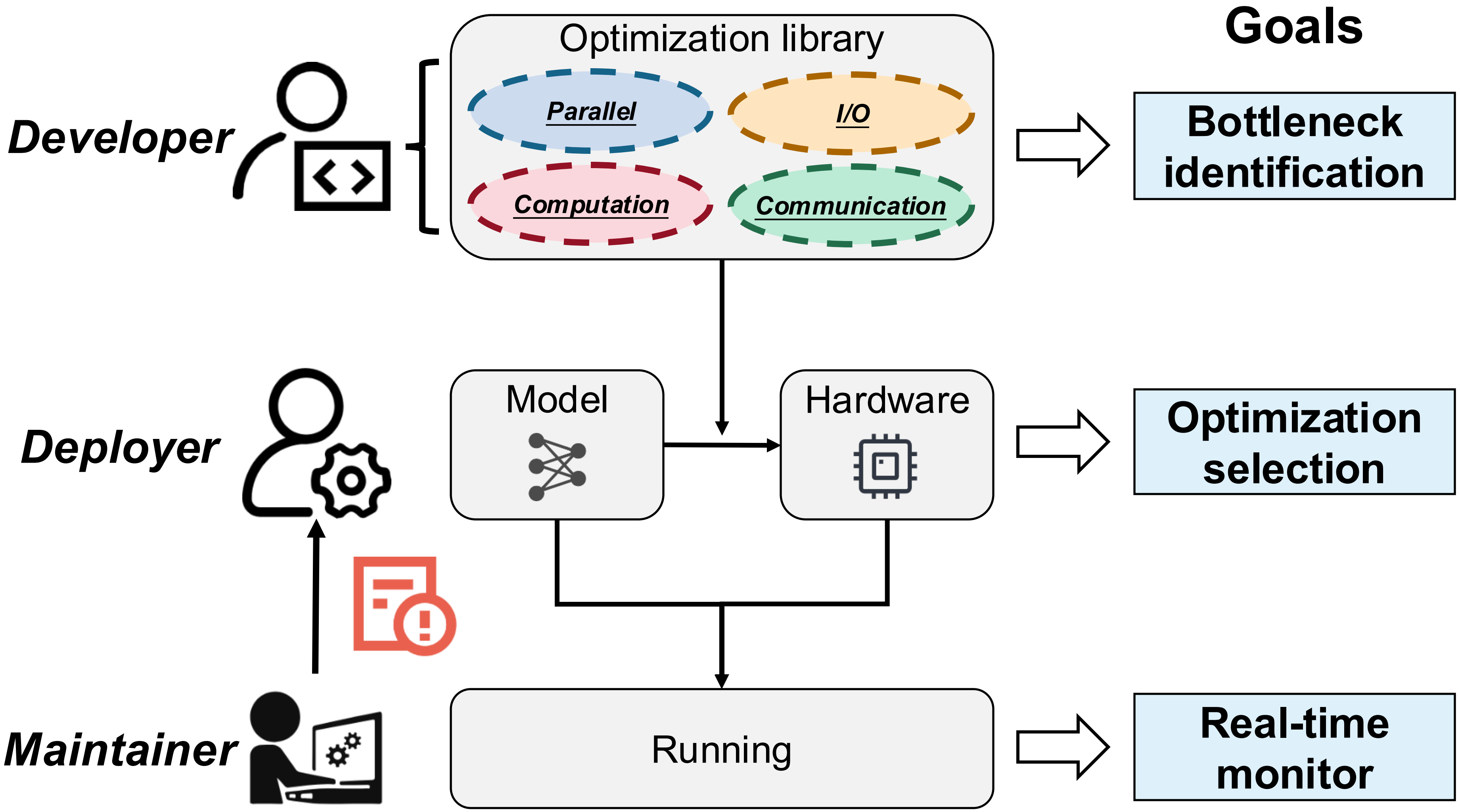
Maintainer

Random and unpredictable performance fluctuations



Real-time monitoring to capture performance fluctuations

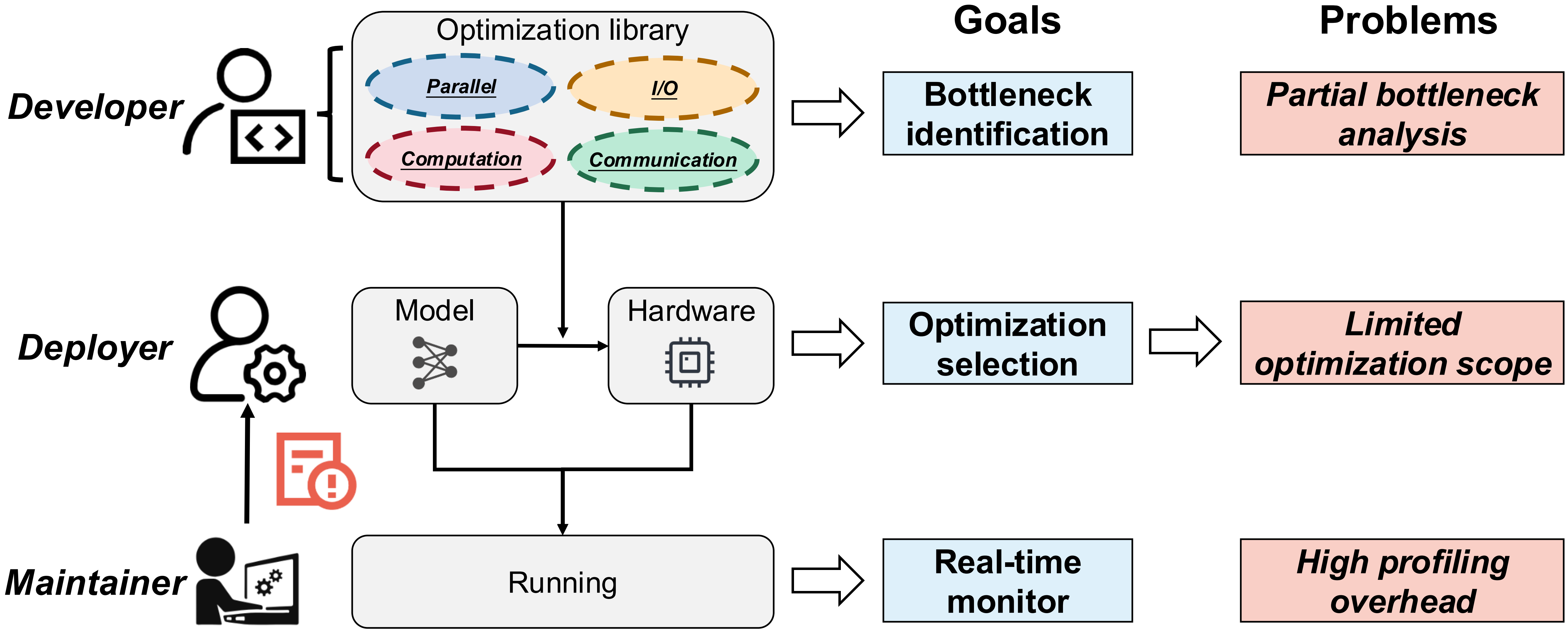
Different Roles in Model Training



Limitations

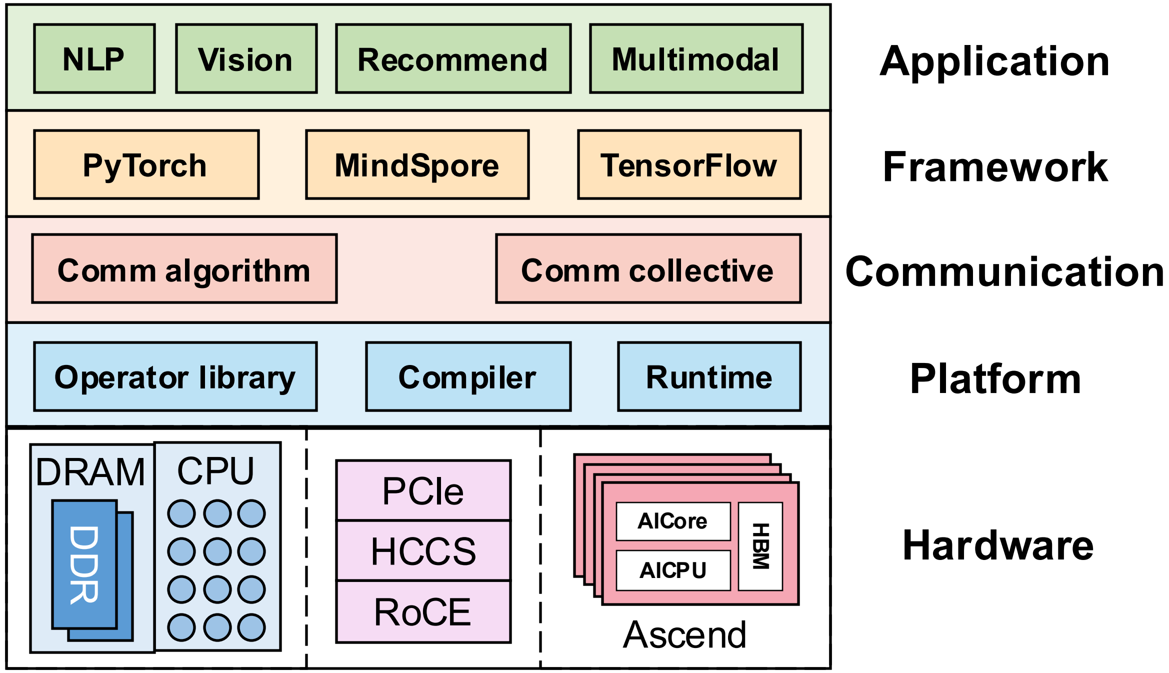
	Profiling	Analysis	Optimization
Bottleneck identification	Fine-grained profiling	Comprehensive analysis	
Optimization selection			Optimization guidance
Real-time monitor	Continuous profiling		
Limitations	***High profiling overhead***	***Partial bottleneck analysis***	***Limited optimization scope***

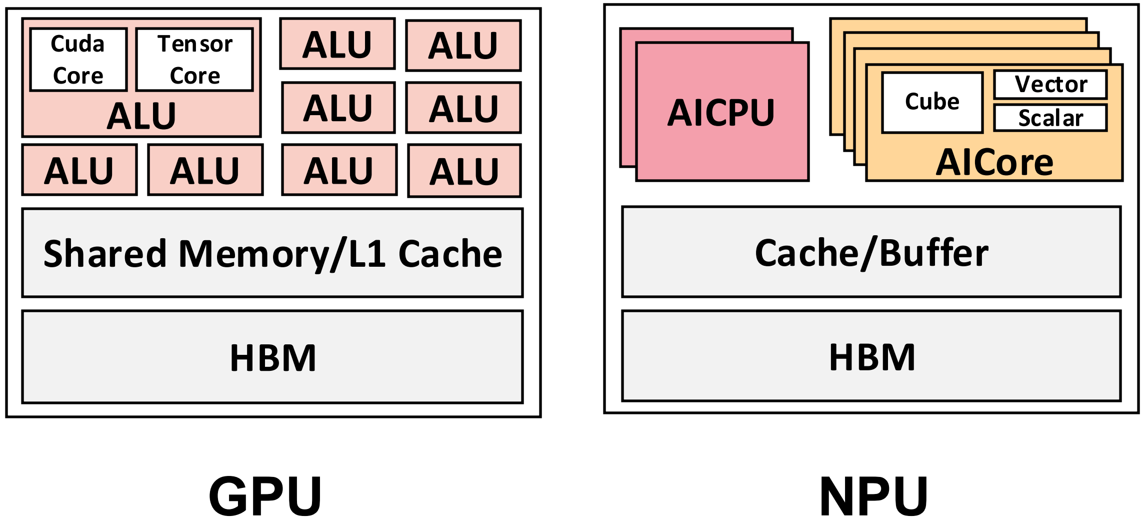
Different Roles in Model Training



Outline

Comparison of NPU and GPU





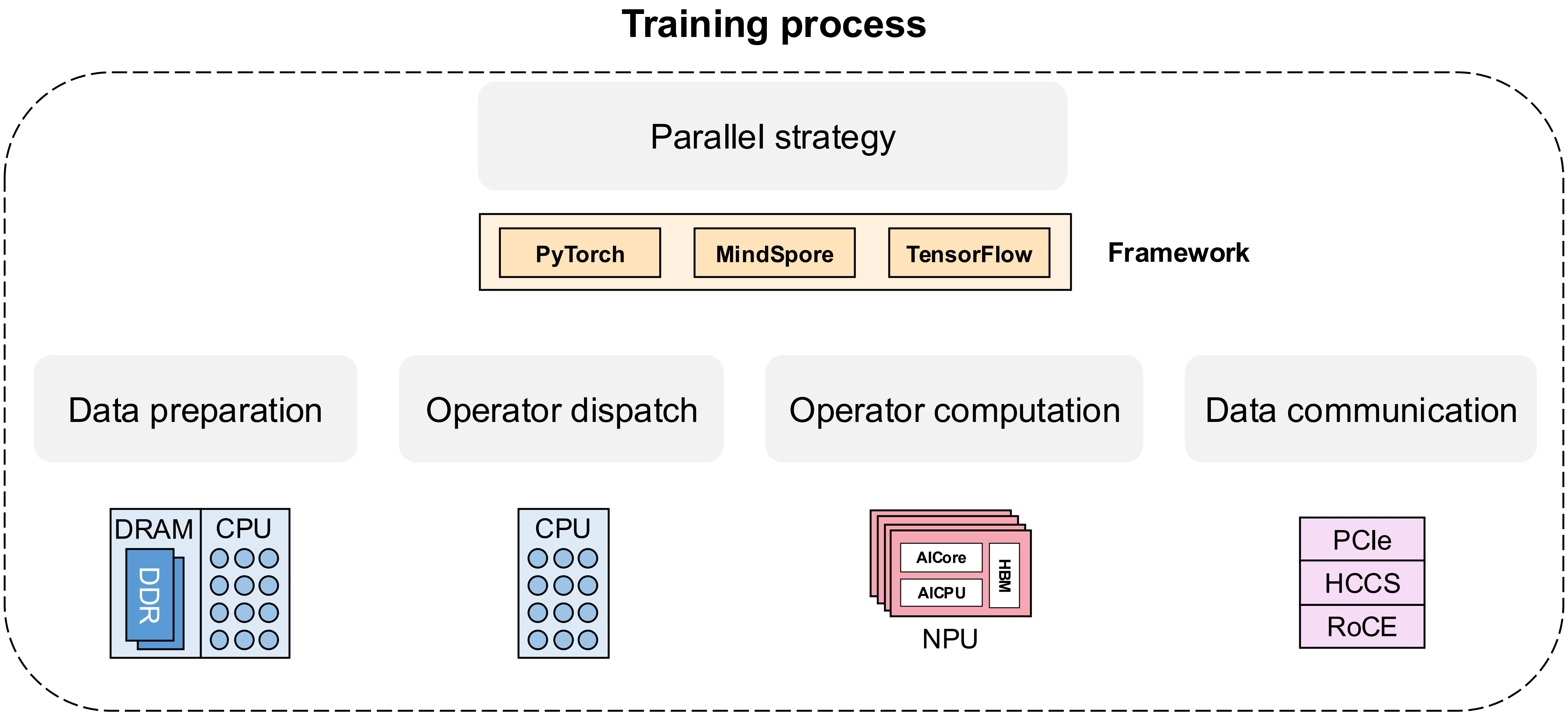
Same hierarchical training paradigm

Hardware-agnostic bottleneck

Differences in chip architecture

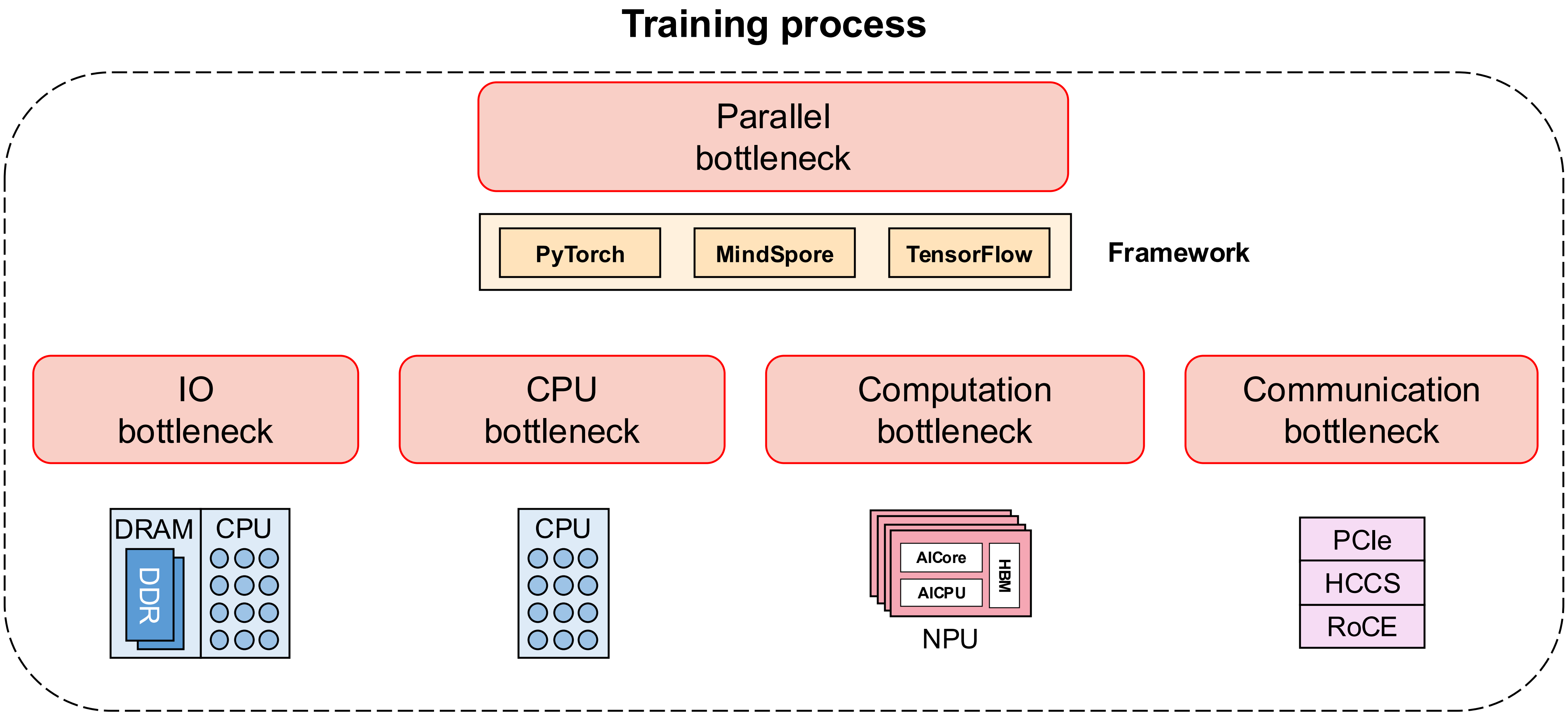
Hardware-specific bottleneck

Insights



Training process
Parallel strategy
PyTorch
MindSpore
TensorFlow
Framework
Data preparation
Operator dispatch
Operator computation
Data communication
DRAM
DDR
CPU
CPU
AICore
AICPU
HBM
NPU
PCIe
HCCS
RoCE

Insights

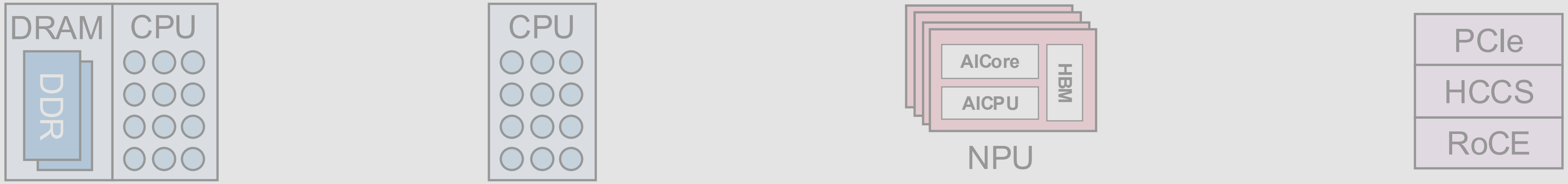

Training process
Parallel bottleneck
PyTorch
MindSpore
TensorFlow
Framework
IO bottleneck
CPU bottleneck
Computation bottleneck
Communication bottleneck
DRAM
DDR
CPU
CPU
AICore
AICPU
HBM
NPU
PCIe
HCCS
RoCE

Insights

Training process

Parallel

Hierarchical bottleneck analysis is feasible!



Outline

Introduction

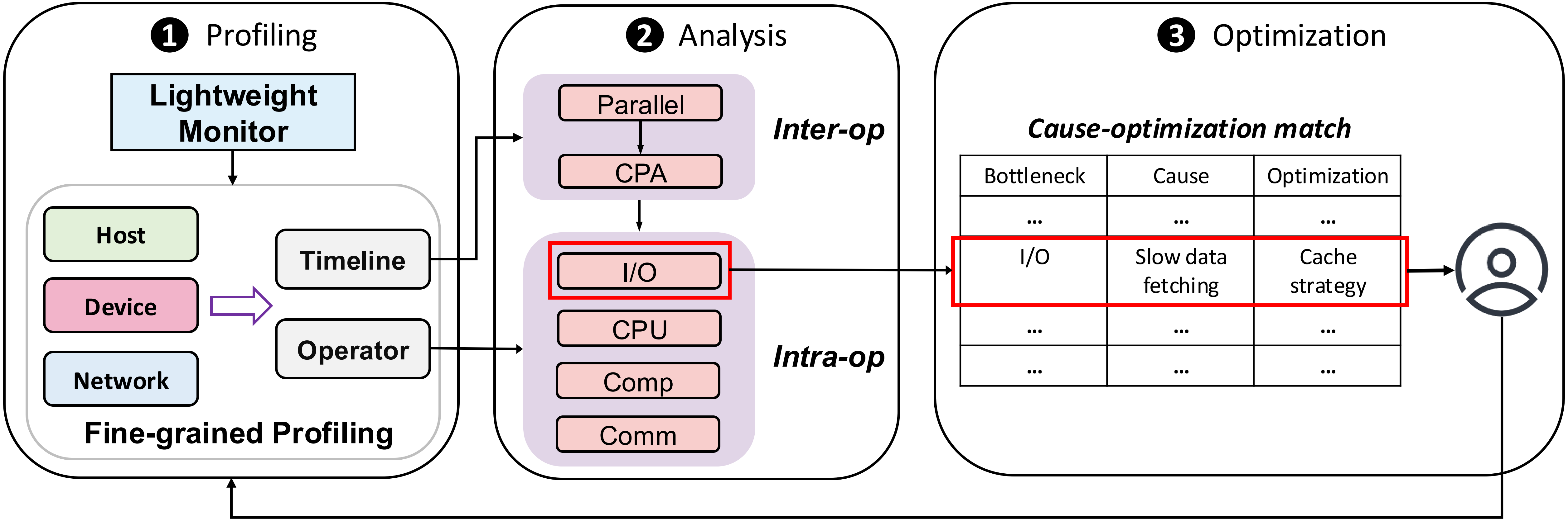
Insights

System Design

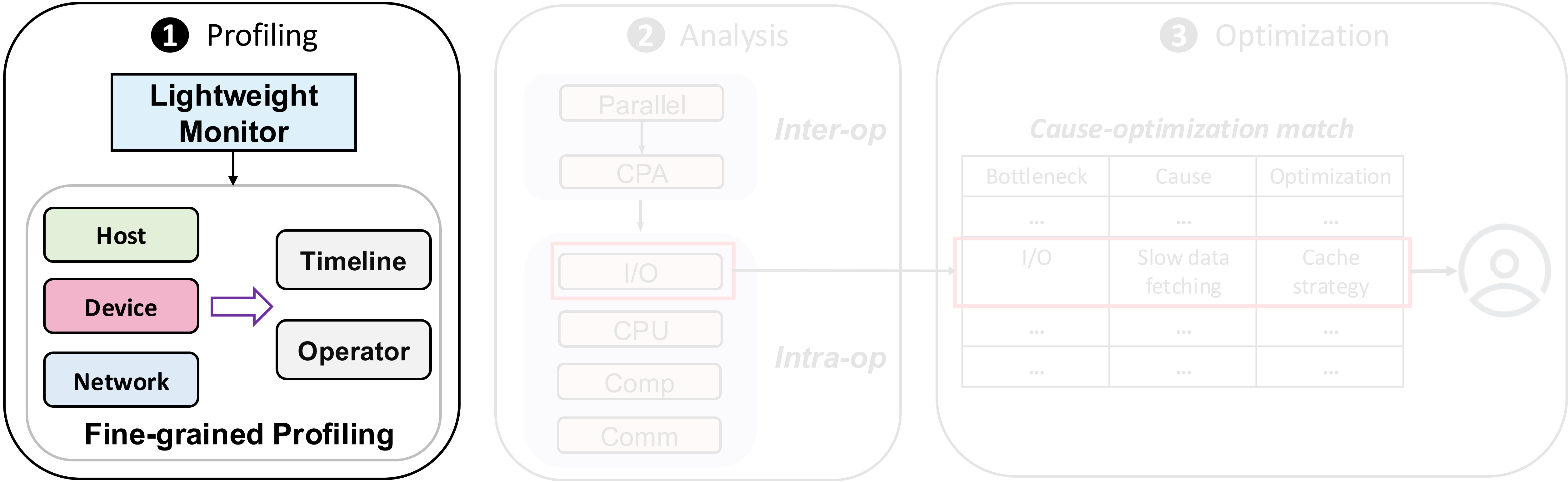
Case Study

Conclusion

Hermes System Design



Hermes System Design

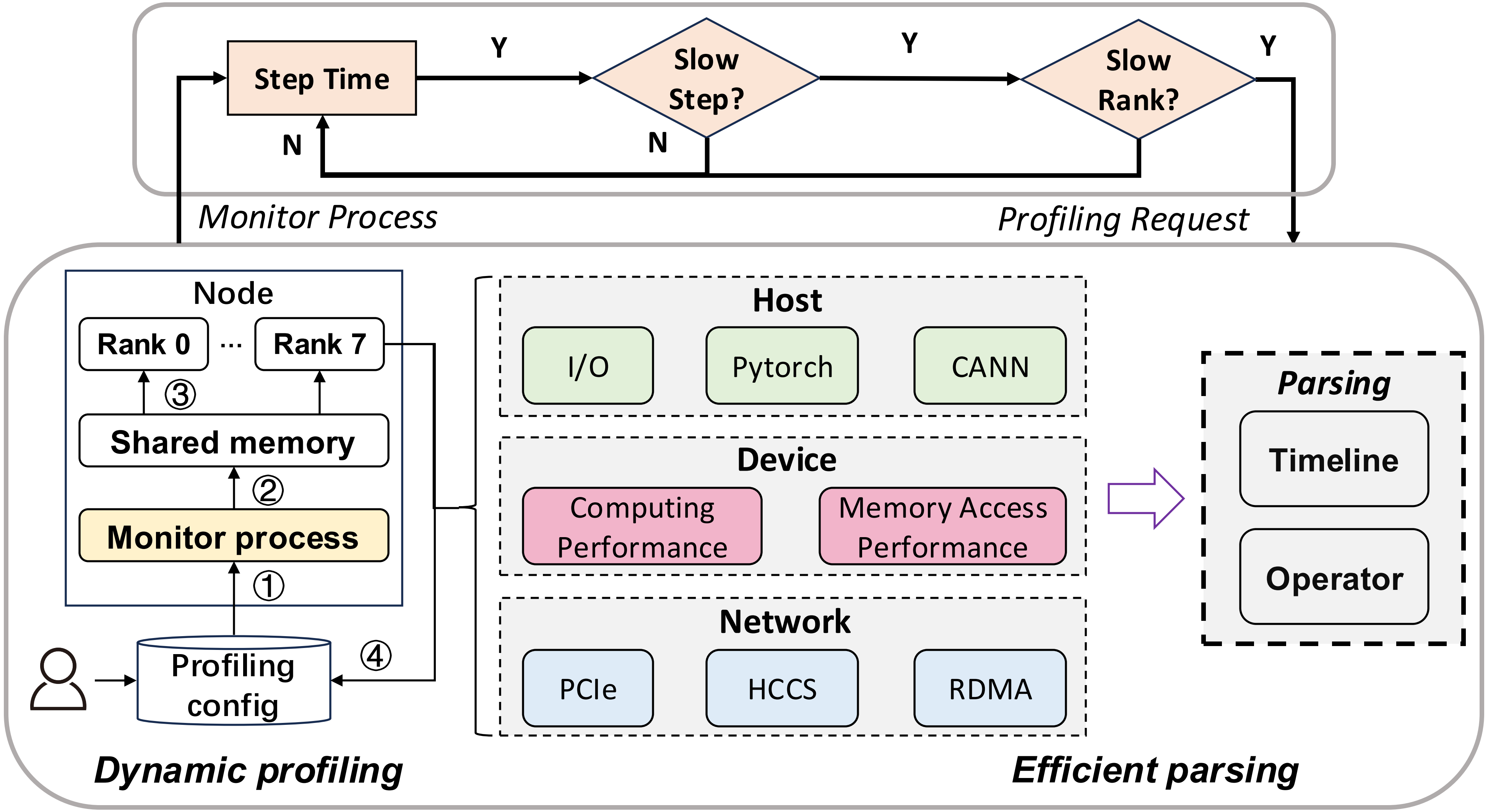


Coarse-to-fine profiling

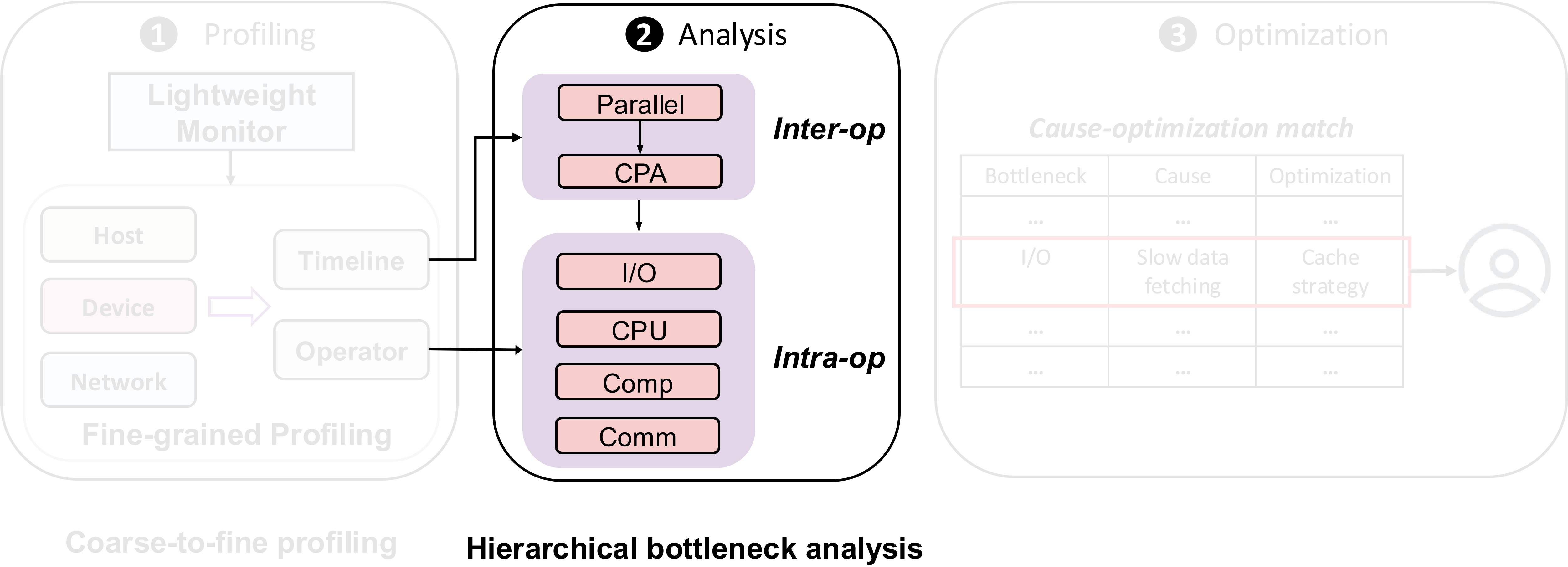
Coarse-to-fine Profiling

Lightweight Monitor

Fine-grained Profiling

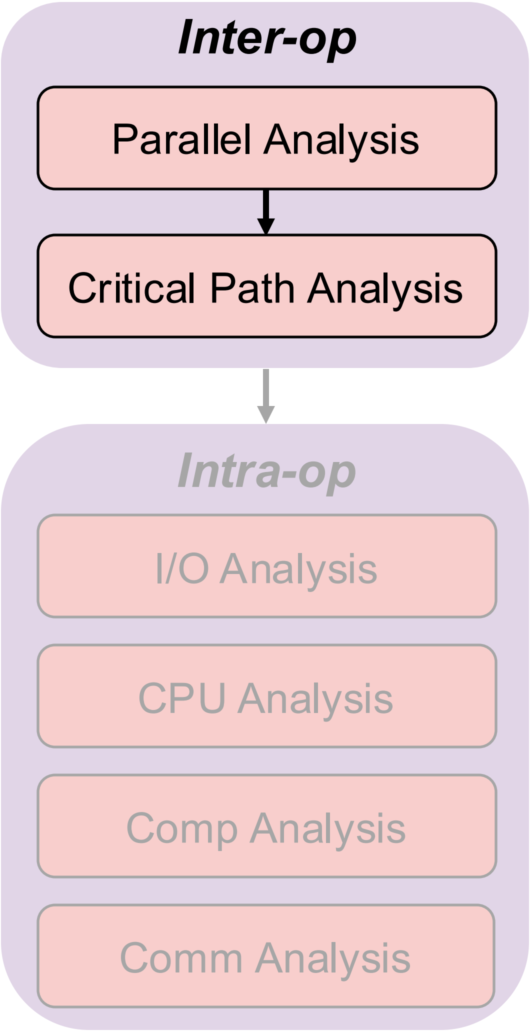


Hermes System Design

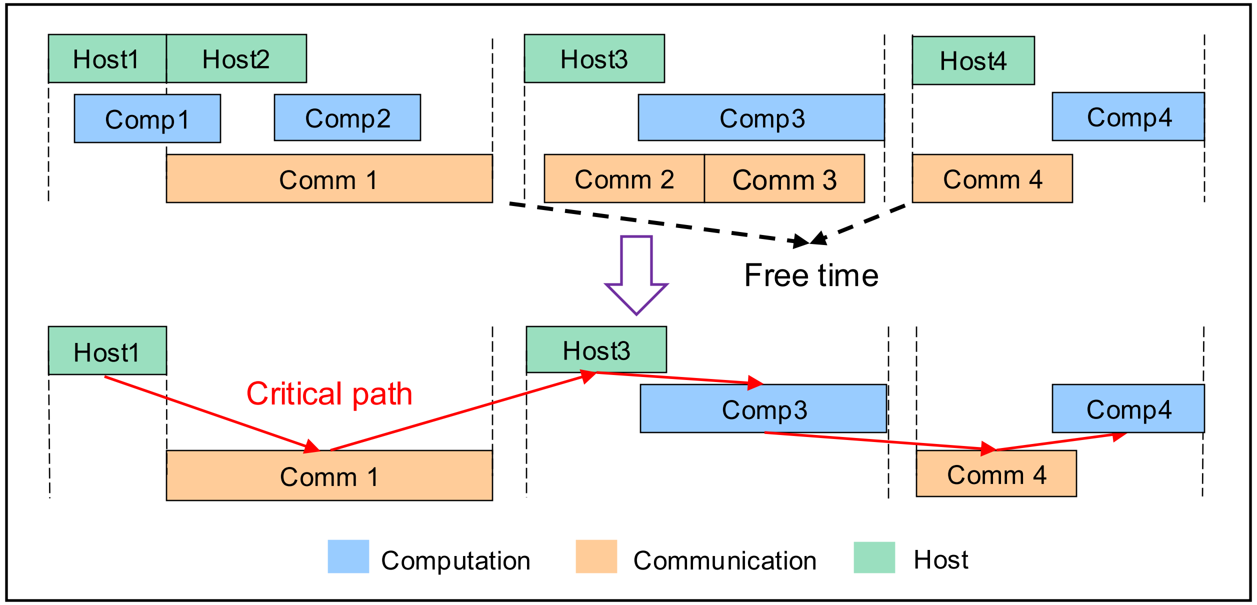


Hierarchical bottleneck analysis

Inter-operator Analysis



- Multi-component Parallel Analysis
  - Overlap, non-overlap computation/communication/host, free time.
- Critical Path Analysis
  - The bottleneck operators with most execution time on the critical path.



I/O Analysis

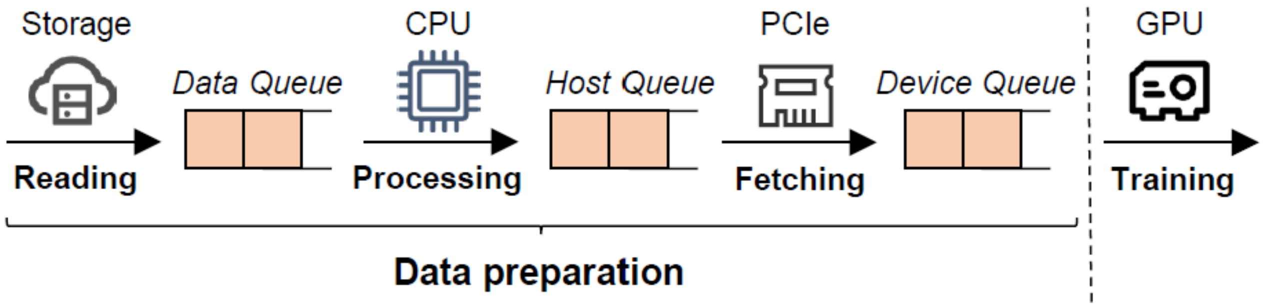
Inter-op

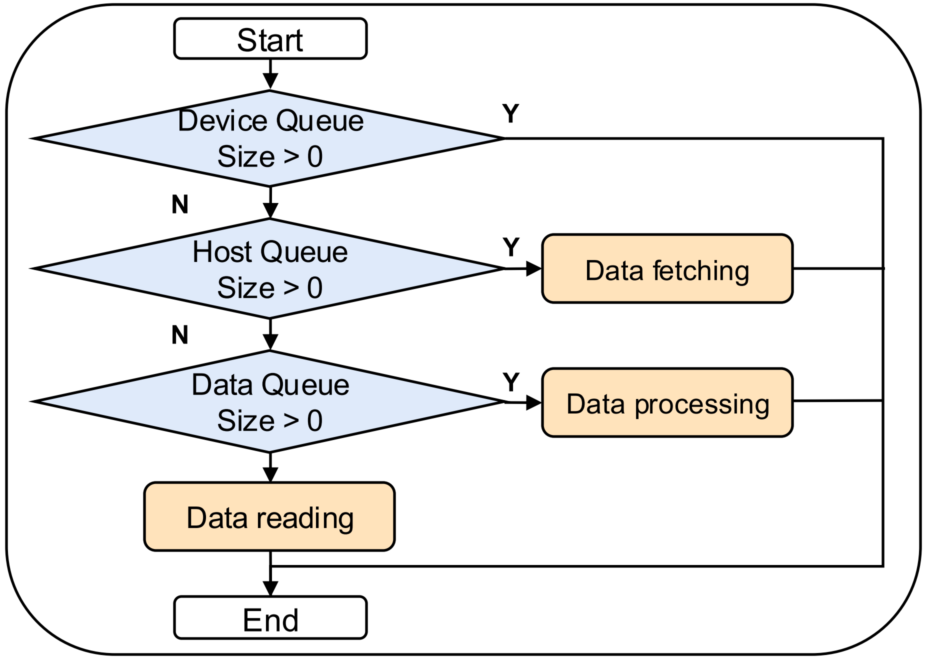
- Parallel Analysis
- Critical Path Analysis

Intra-op

- **I/O Analysis**
- CPU Analysis
- Comp Analysis
- Comm Analysis

- Queue-based I/O Analysis





CPU Analysis

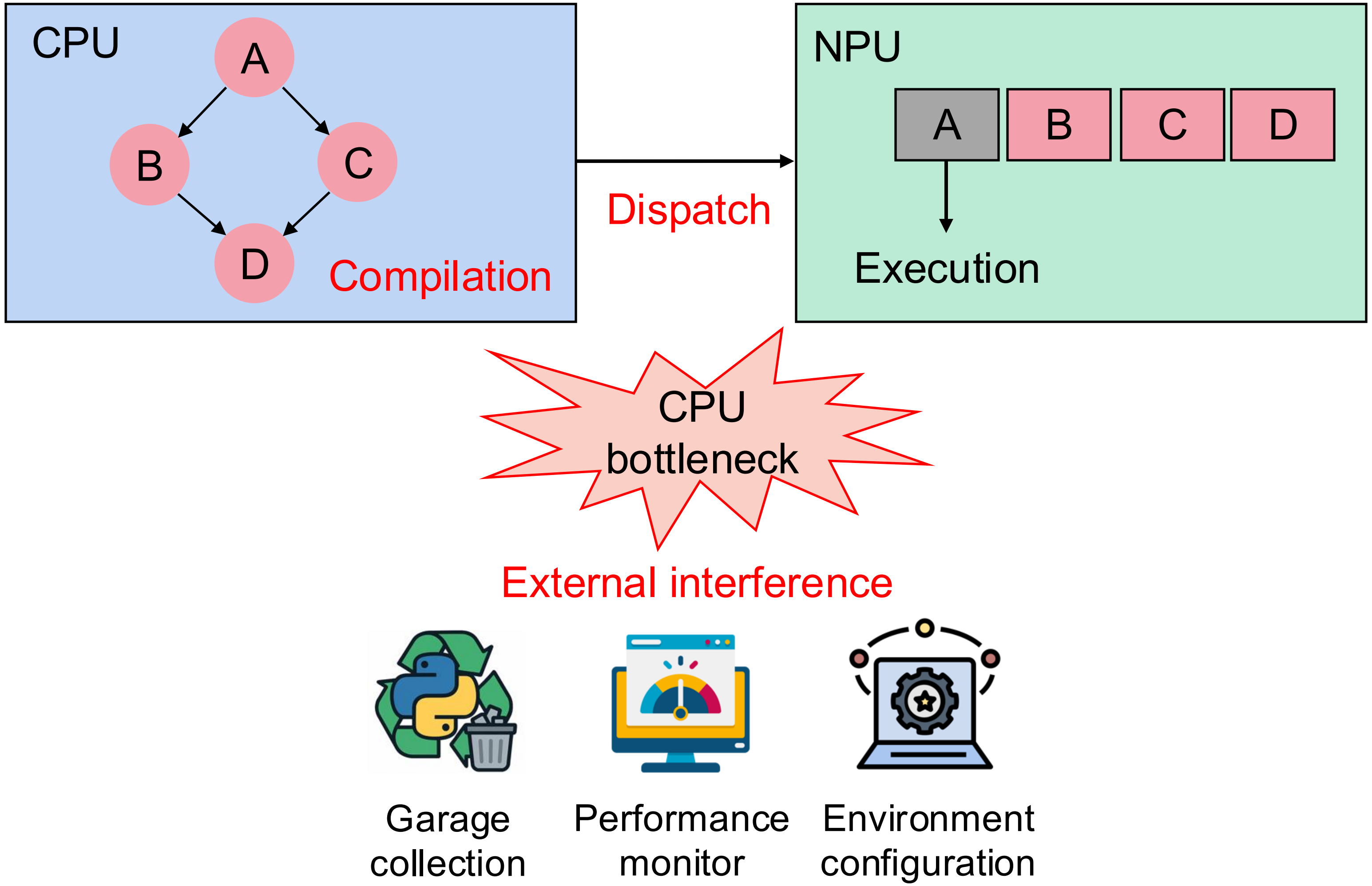
Inter-op

- Parallel Analysis
- Critical Path Analysis

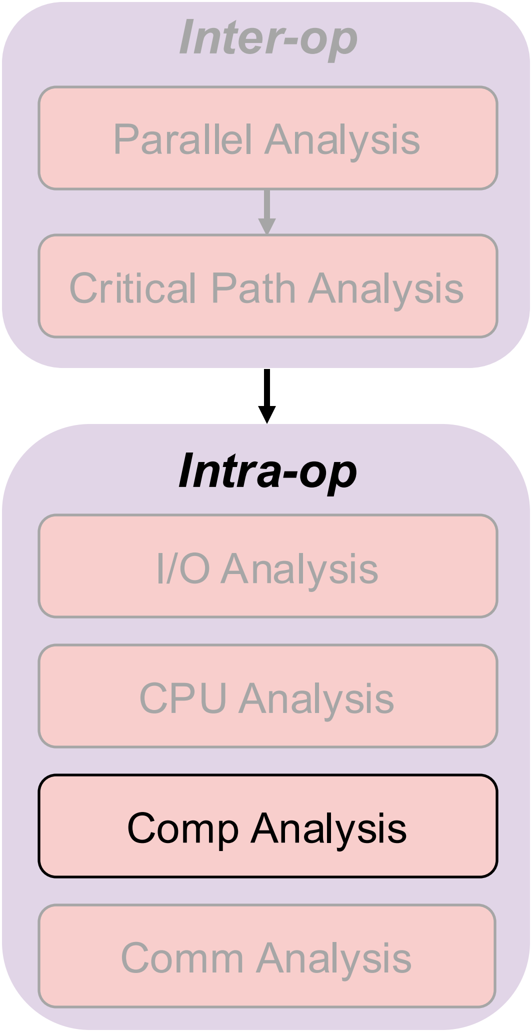
Intra-op

- I/O Analysis
- **CPU Analysis**
- Comp Analysis
- Comm Analysis

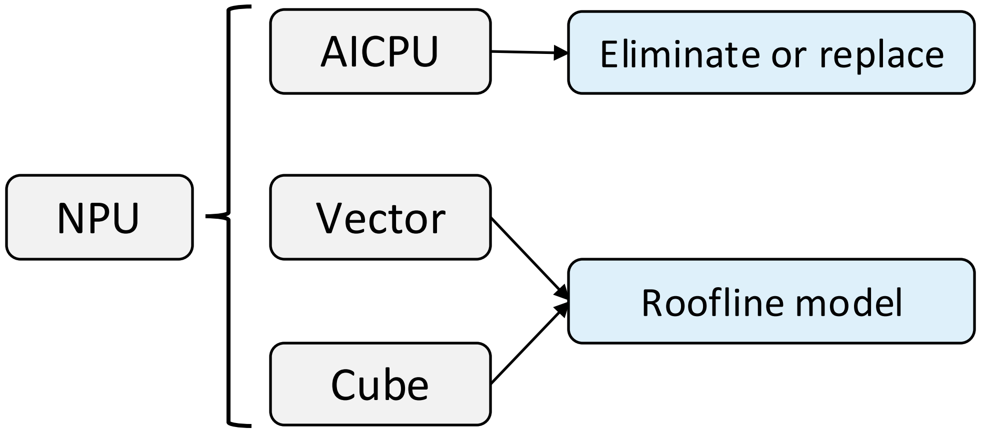
- CPU Bottleneck Causes

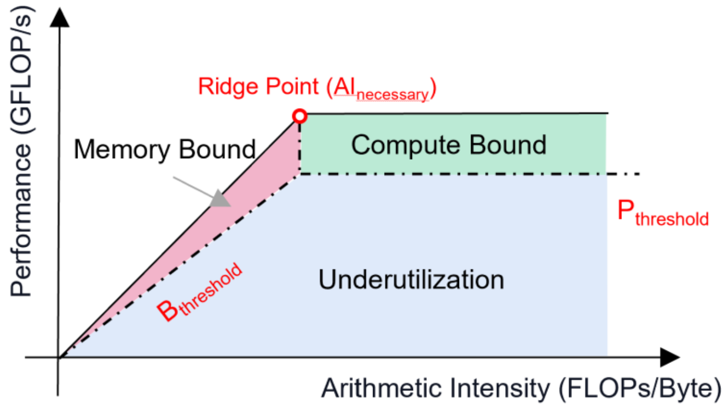


Computation Analysis



- Computation Bottleneck
  - Different compute units (AICPU, AICore Cube/Vector)
  - Roofline model analysis (arithmetic, memory)





Communication Analysis

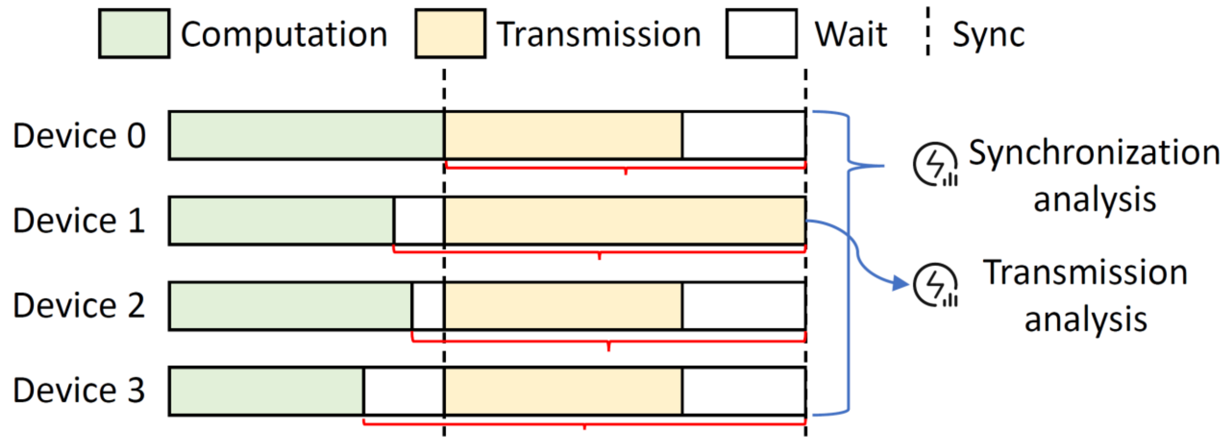
Inter-op

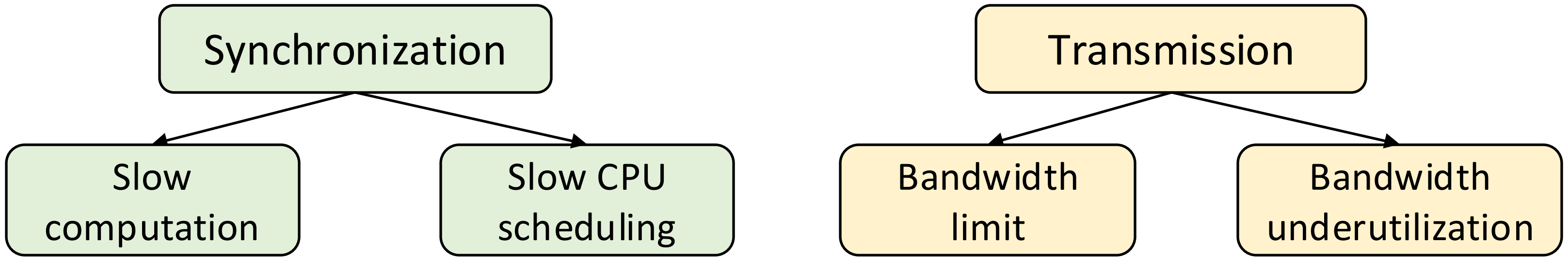
- Parallel Analysis
- Critical Path Analysis

Intra-op

- I/O Analysis
- CPU Analysis
- Comp Analysis
- **Comm Analysis**

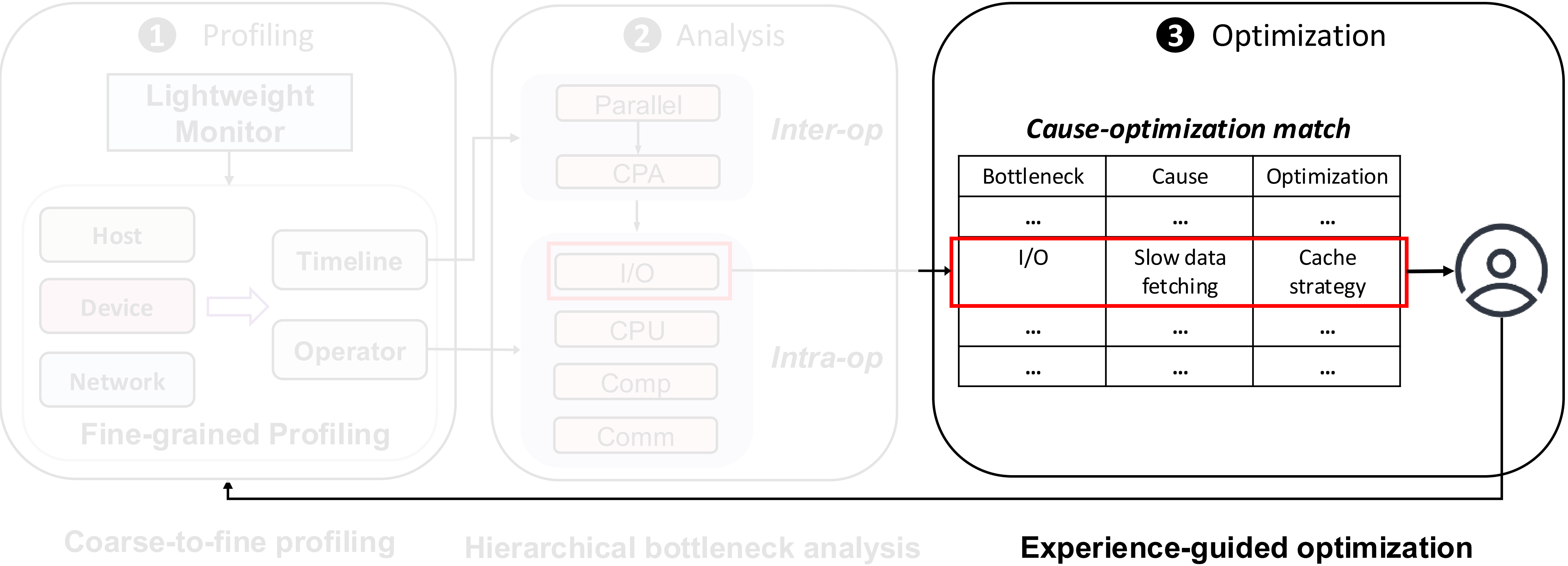
- Synchronization + Transmission





Detailed causes can be found in the paper.

Hermes System Design



Bottleneck	Cause	Optimization
...	...	...
I/O	Slow data fetching	Cache strategy
...	...	...
...	...	...

Bottleneck Cause-Optimization Match

Bottleneck	Cause	Optimization	Ratio
Parallel	Poor Parallelism	Auto hybrid parallel [67] / Multi-shard parallel	5.2%
I/O	Slow Data Reading	Increase I/O bandwidth / Remote to local storage	8.9%
	Slow Data Processing	Improve CPU parallelism (num_workers)	
		Avoid compression formats (zip, tar)	
		Cancel the taskset process binding [33]	
	Slow Data Fetching	Cache strategy (pin_memory, data prefetcher) [24, 66]	
CPU	Operator Complication	Replace dynamic shape operators / Disable JIT compilation	37.0%
	Operator Dispatch	Operator fusion [43, 68] / Eliminate synchronization operations	
	Garbage Collection	Disable gc / Increase gc threshold	
	CPU Resources Contention	Disable other CPU process	
	Environment Configuration	Align software versions / Reduce logging level	
Computation	Compute Bound	Avoid decreasing computing frequency / Isolate slow nodes	31.9%
	Memory Bound	Operator fusion [43, 68] / Quantization [38, 56, 63] / ZeRO [51, 52]	
	Underutilization	Eliminate AICPU operators	
		Replace operators with affinity APIs	
		Forbid private format	
Communication	Bandwidth Contention	Avoid bandwidth contention by re-scheduling operators	17.0%
	RDMA Retransmission	Adjust RDMA network configurations of switch and server	
	Small Packet	Increase batch size / Gradient fusion [26, 46] / Operator fusion	
	Byte Alignment	Align HCCS data size	
	Network Configuration	HCCL environment variables / Switch congestion control / UDP hashing collision	

CPU and computation bottlenecks dominate.

Lessons

CPU

CPU scheduling bottlenecks dominate but are often overlooked.

Computation

Underutilization dominates computation bottlenecks.

Communication

Contention between computation and communication.

Port flapping and link failures dominate network issues.

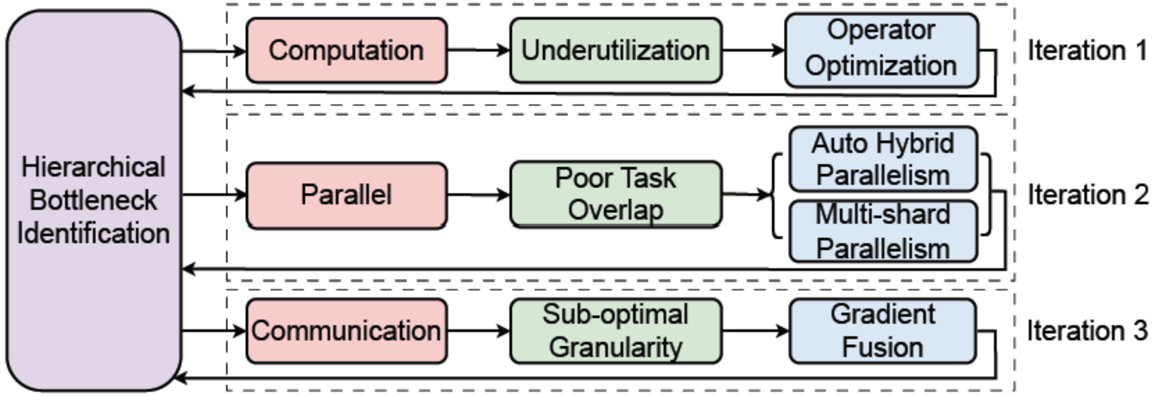
IO

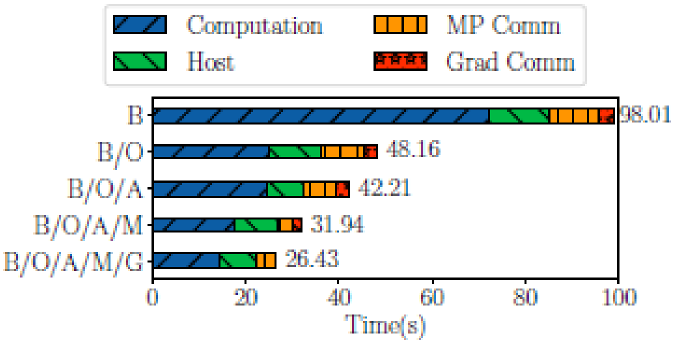
Minimize remote access to alleviate I/O bottlenecks.

Outline

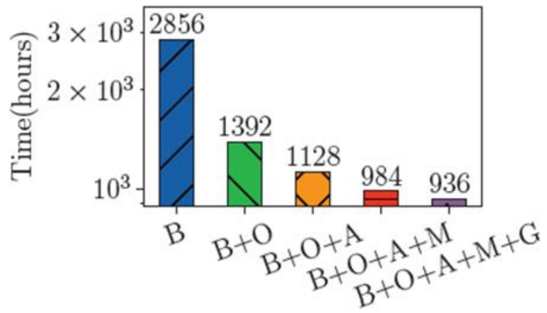
Iterative Optimization Development for PanGu-α

Device: 128 Ascend 910A
Workloads: 100B PanGu-α model training





Step time.



Overall training time.

The optimization of large model training often requires multiple iterations.

After three iterations of optimization, the *total time* speedup is 3.05x.

Deployment Optimization Experience

We summarize the speedups from optimization in different model deployments.

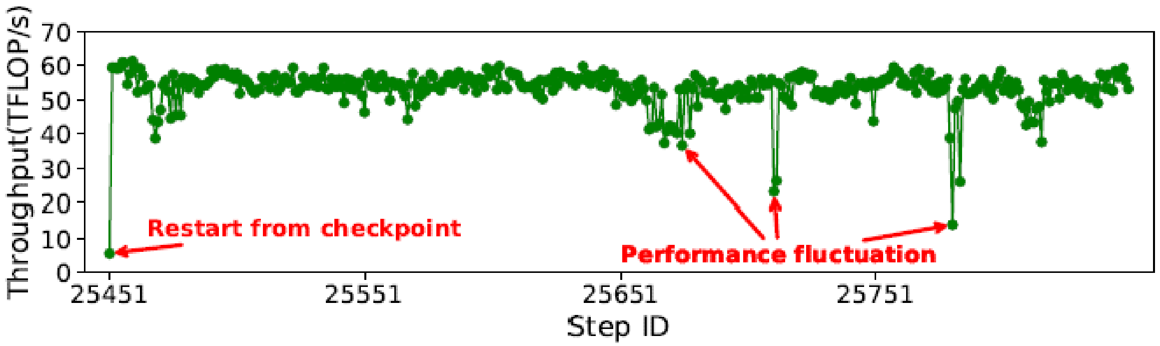
Type	Model	Parameter	Optimization Speedup (-: not optimizable)						# of NPUs	Dataset
			I/O	CPU	Para.	Compu.	Comm.	Total		
Vision	ResNet50	25.6M	5.03	-	-	1.02	1.04	5.34	8	ImageNet2012
	VGG16	138.4M	-	-	-	1.08	1.35	1.46		
	MobileNetV1-SSD	4.2M	-	1.37	-	-	-	1.37	1	VOC2012
			1.08	1.91	-	-	-	2.07	8	
NLP	Bert-Large	330M	-	-	-	1.63	1.38	2.49	8	Wiki
	PanGu-α	1.3B	-	-	-	1.18	1.02	1.20		
	GPT3-13B	13B	-	-	1.08	-	-	1.08		
Recommend	DeepFM	16.5M	-	-	-	-	1.08	1.08	8	Criteo
	DLRM	540M	-	-	-	-	1.17	1.17		

Our optimizations bring training speedups from 1.08-5.34× in vision, NLP, and recommendation models.

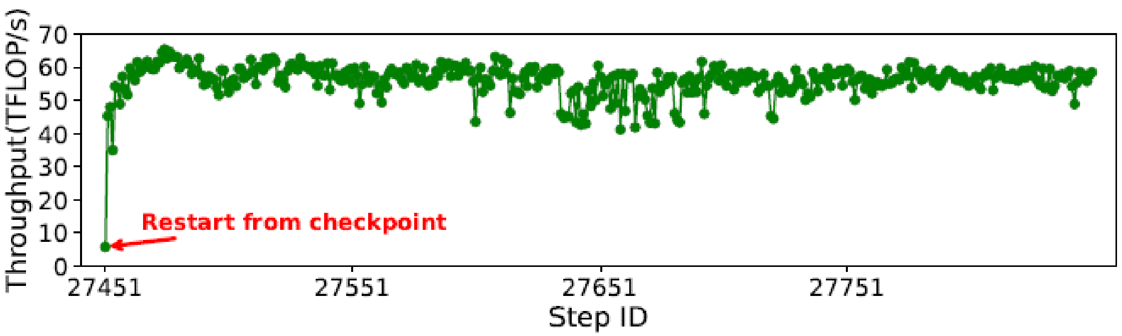
Detailed cases can be found in the paper.

Performance Fluctuation Optimization

9k-card MoE model training



(a) Performance before optimization.



(b) Performance after optimization.

(1) Increase Python garbage collection threshold.

(2) Active garbage collection when saving checkpoints.

Training time speedup is 1.06×.

Average throughput speedup is 1.05×.

Outline

Conclusion

1. We propose Hermes, a systematic training optimization system with lightweight profiling, hierarchical analysis, and automated optimization guidance.

2. We summarize insights from 135 real-world cases and demonstrate Hermes's effectiveness through extensive case studies.

Future Work

1. Expand Hermes to support emerging model training technologies like reinforce learning.

2. Improve Hermes's ability to handle more complex bottlenecks and situations.

3. Integrate training logs and even LLM-based agents to more accurate bottleneck analysis.

Thanks

Q&A

wangzb@smail.nju.edu.cn
yuhangzhou@smail.nju.edu.cn