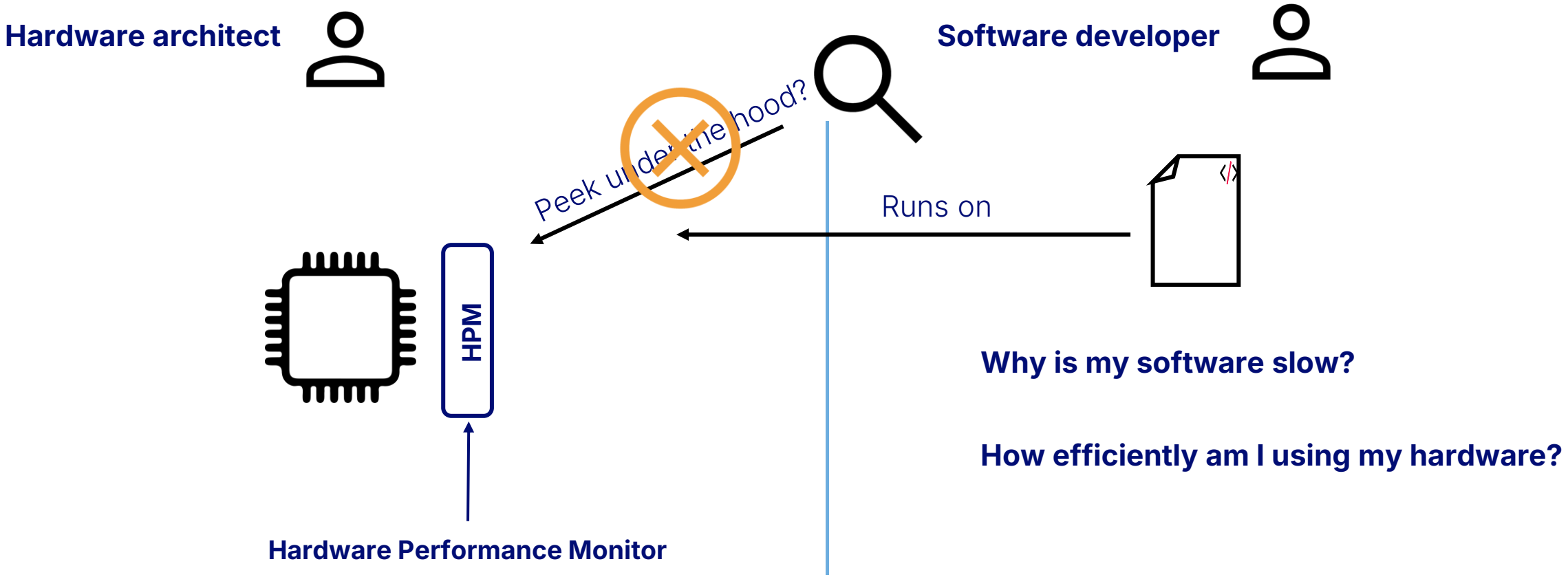
Towards Open-Source and Automatic Performance Characterization Hardware

Matthew Edwin Weingarten

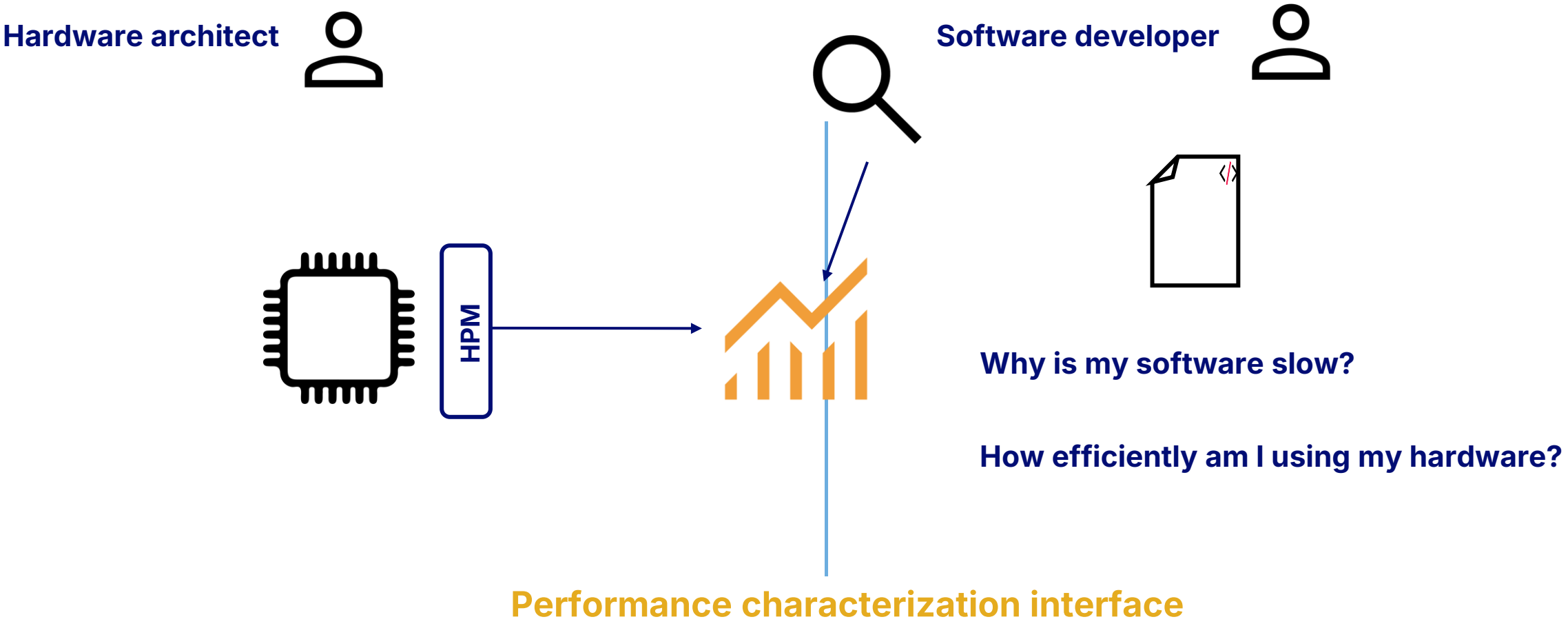
COLUMBIA ENGINEERING

RISC-V®

Why do we need performance characterization?

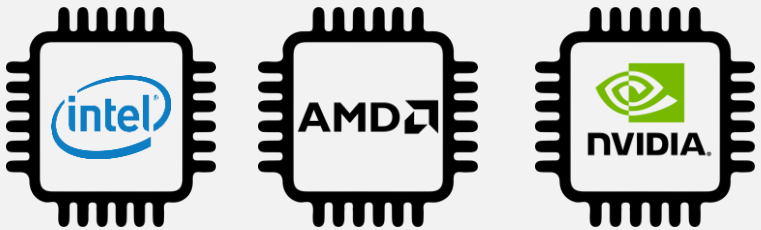


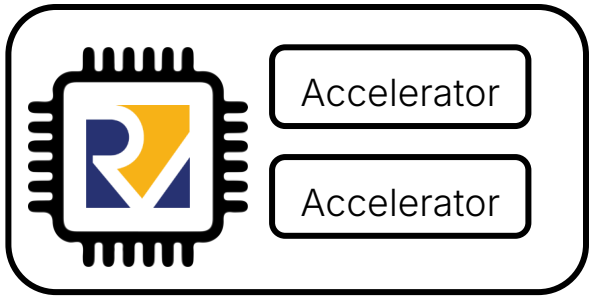
Why do we need performance characterization?



What are the trends in hardware design?

Performance Characterization



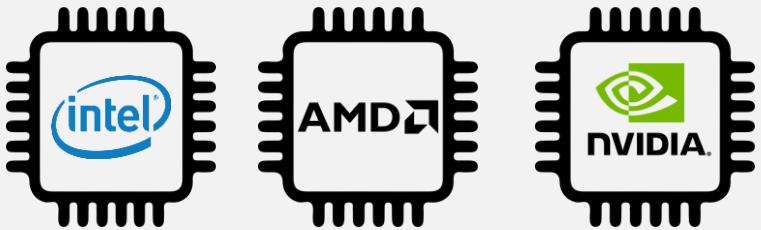


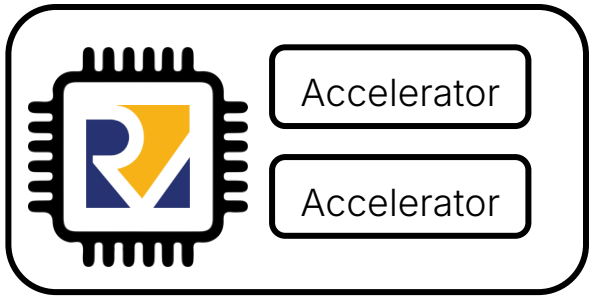
GPU, TPU, NVDLA, Gemini, NPU, Vortex

Proto-acc, CDPU,PIM,..

What are the trends in hardware design?

Performance Characterization

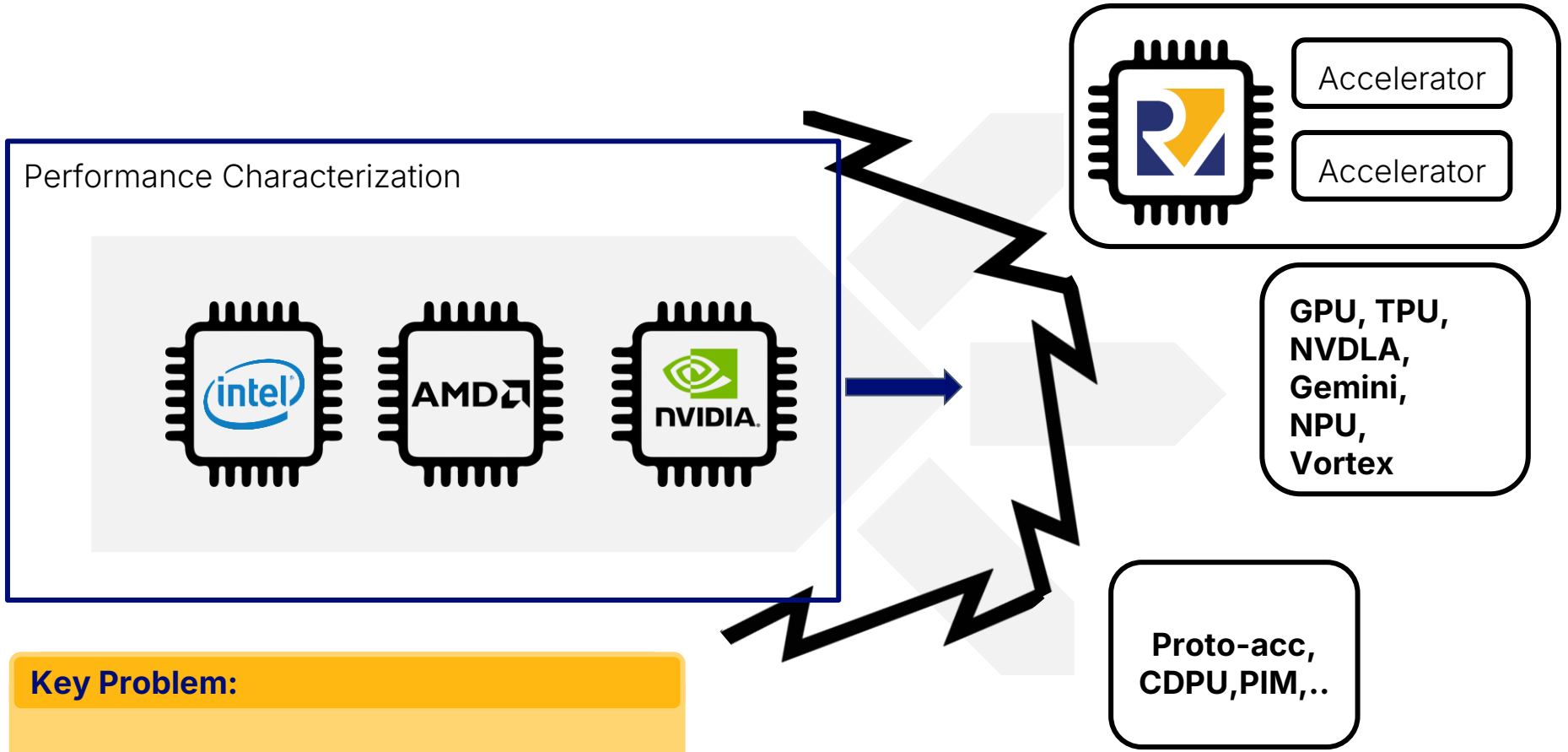




GPU, TPU, NVDLA, Gemini, NPU, Vortex

Proto-acc, CDPU,PIM,..

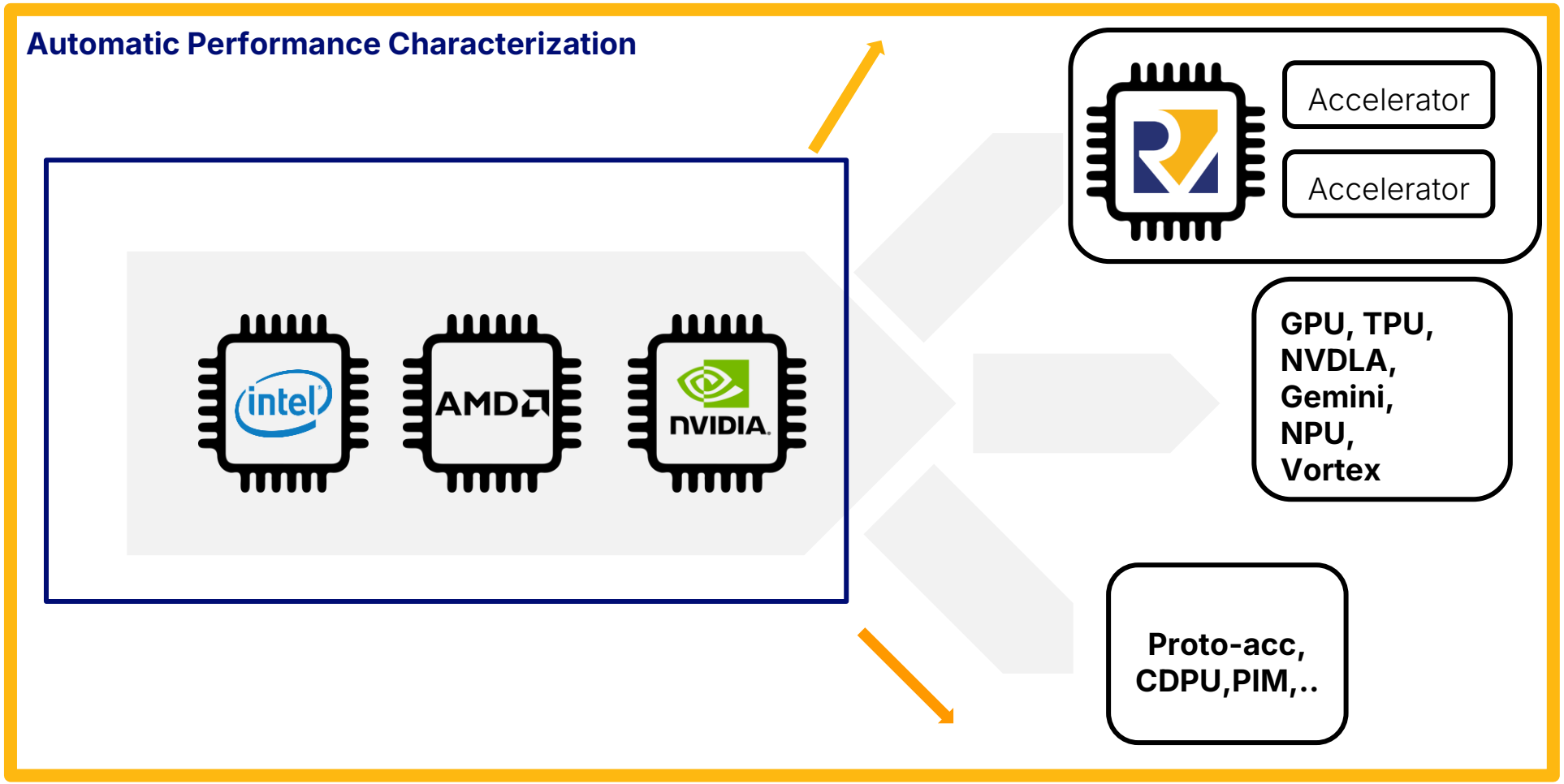
What are the trends in hardware design?



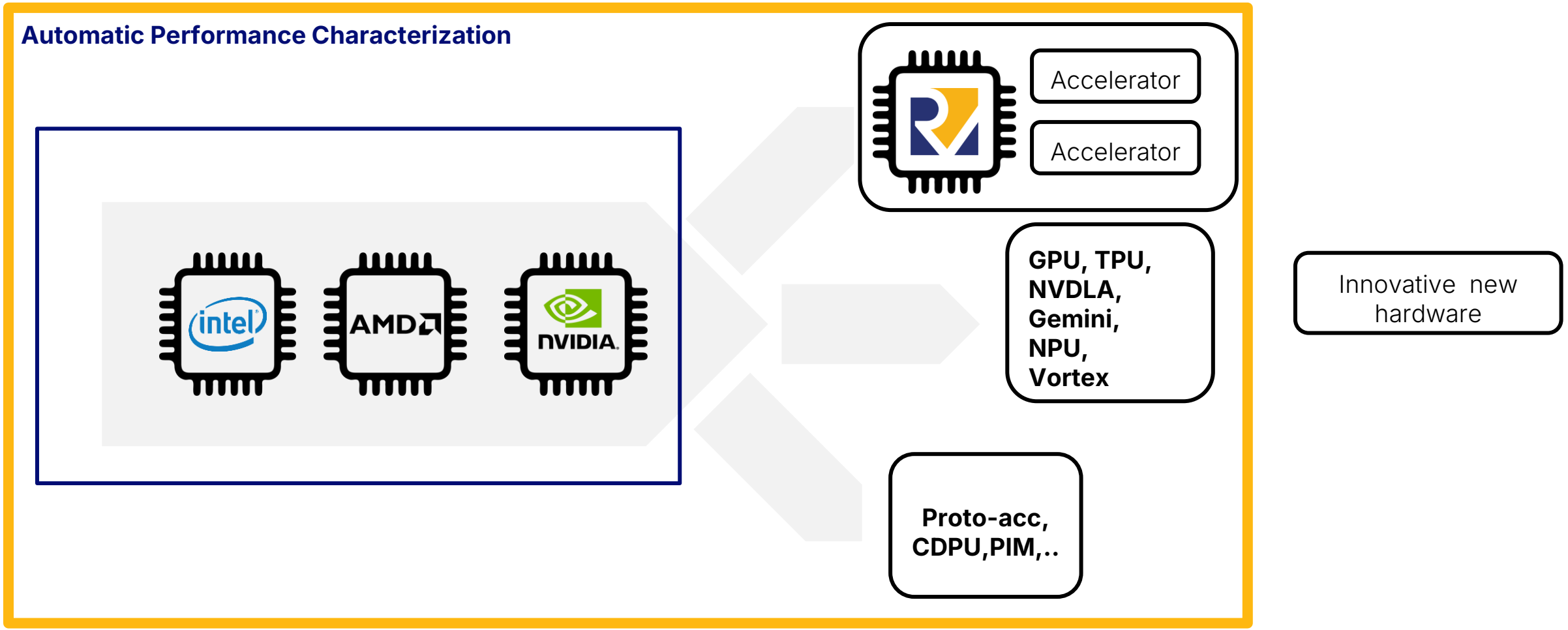
Key Problem:

Performance characterization requires **lots of effort, hard to get right**

Our vision: Open-source and automatic characterization

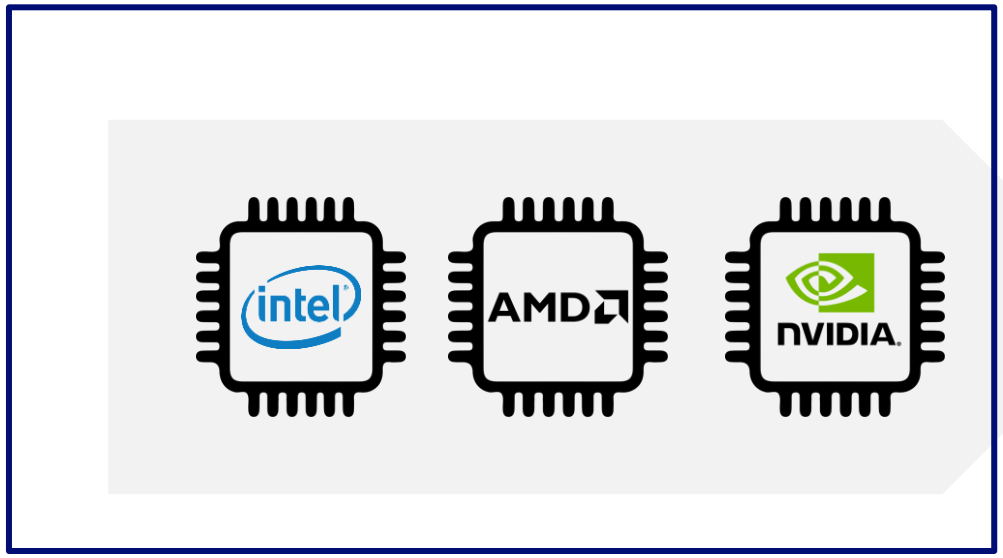


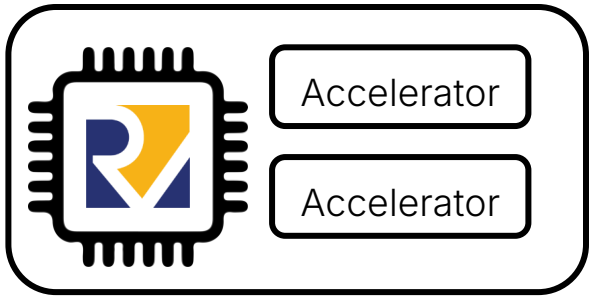
Our vision: Open-source and automatic characterization



Our vision: Open-source and automatic characterization

Automatic Performance Characterization



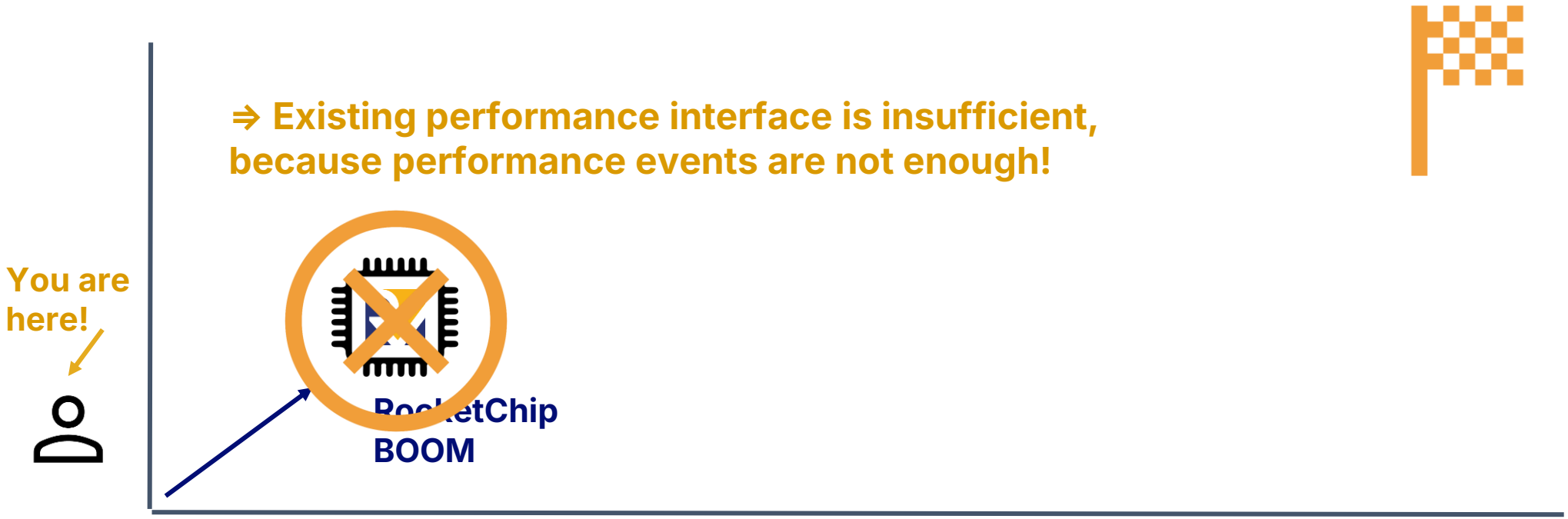


GPU, TPU, NVDLA, Gemini, NPU, Vortex

Proto-acc, CDPU,PIM,..

Innovative new hardware

Our vision: Open-source and automatic characterization



Key Vision:

Fully **automatic** and **generalized** performance characterization

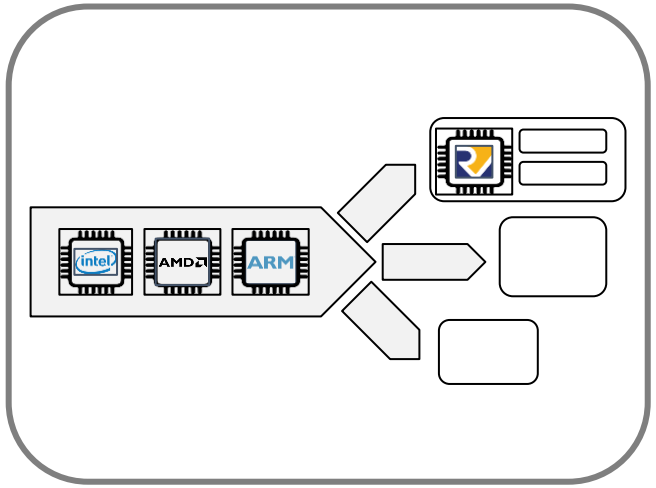
Key Problem:

Performance characterization requires **lots of effort, hard to get right**

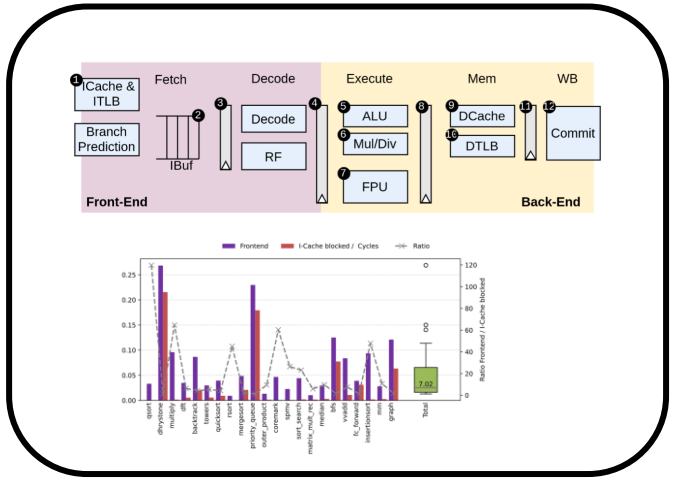
Key Contribution:

Performance characterization HPM events for **RocketCore** and **BOOM**

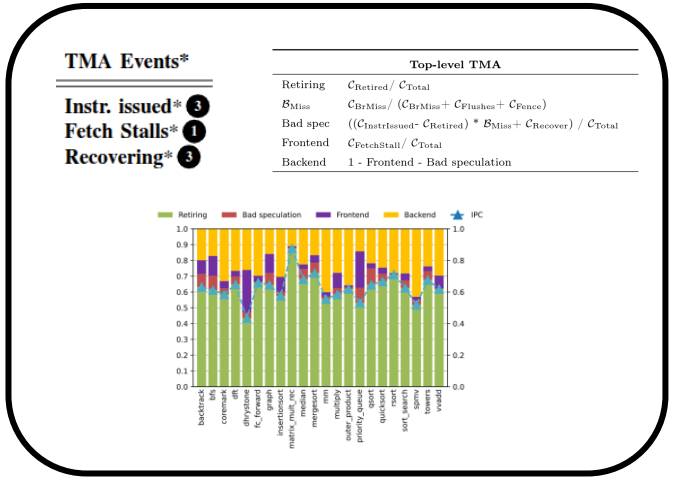
Outline



Introduction & motivation



Performance characterization on RISC-V

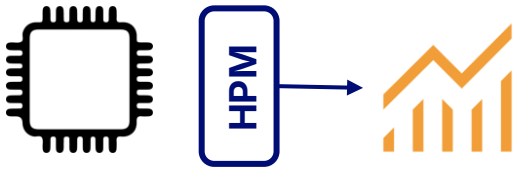


Methodology & results

COLUMBIA ENGINEERING

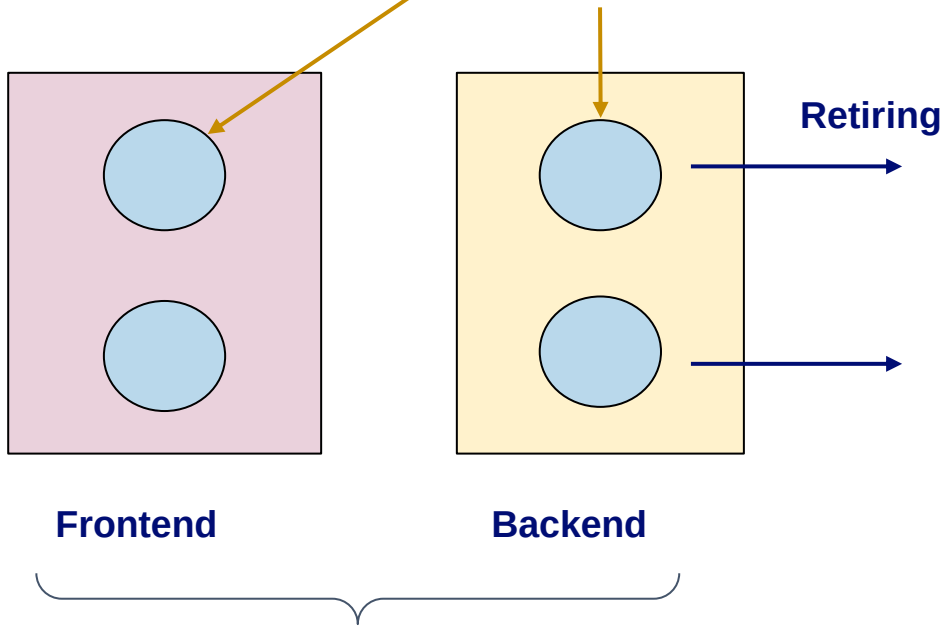
ICE

What is Top-Down Microarchitectural Analysis (TMA)?



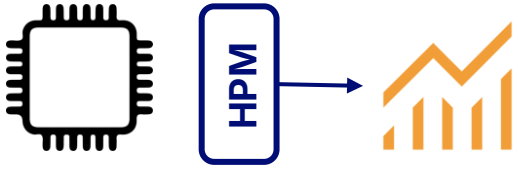
Total slots = cycles x pipeline width

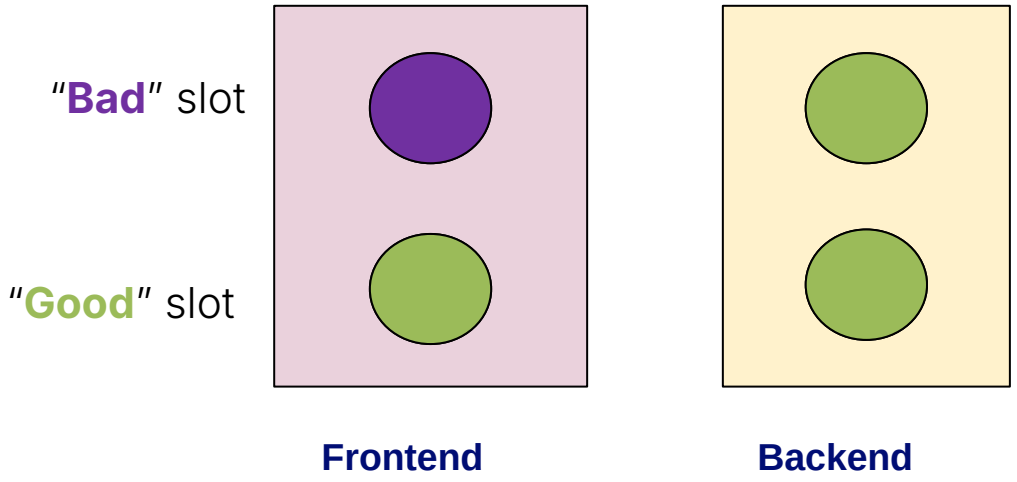
Slot = Abstraction for “work” in the pipeline



Semantic grouping of pipeline

What is Top-Down Microarchitectural Analysis (TMA)?

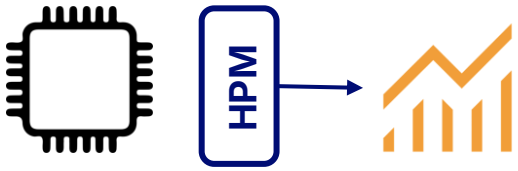


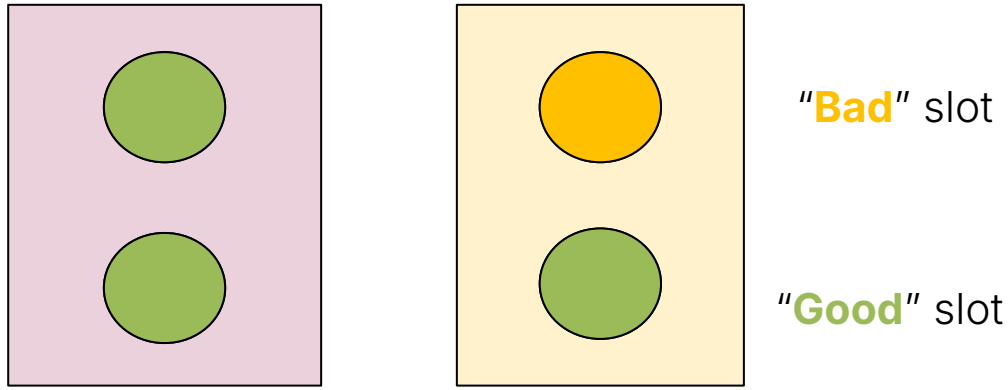


TMA

⇒ Categorize slots

What is Top-Down Microarchitectural Analysis (TMA)?

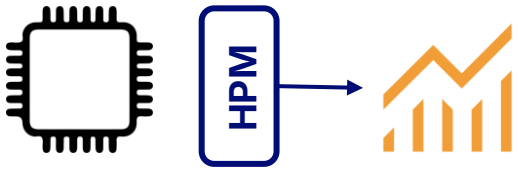


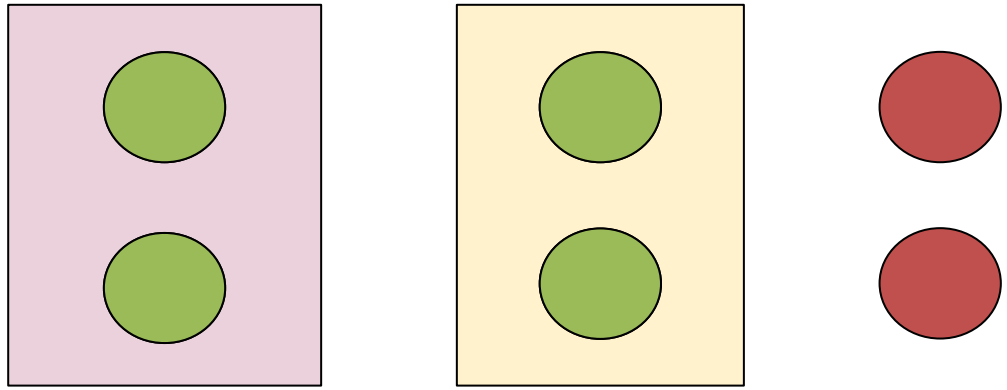


TMA

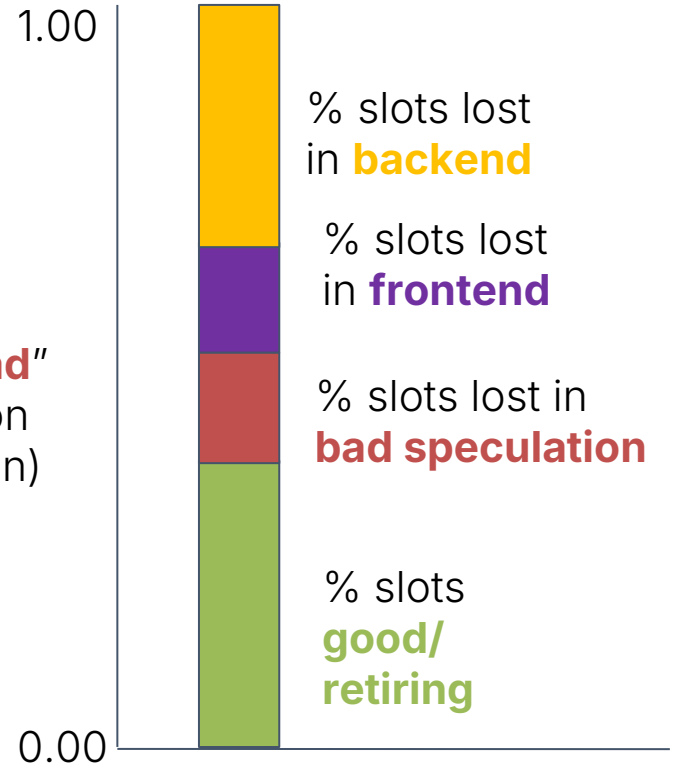
⇒ Categorize slots

What is Top-Down Microarchitectural Analysis (TMA)?





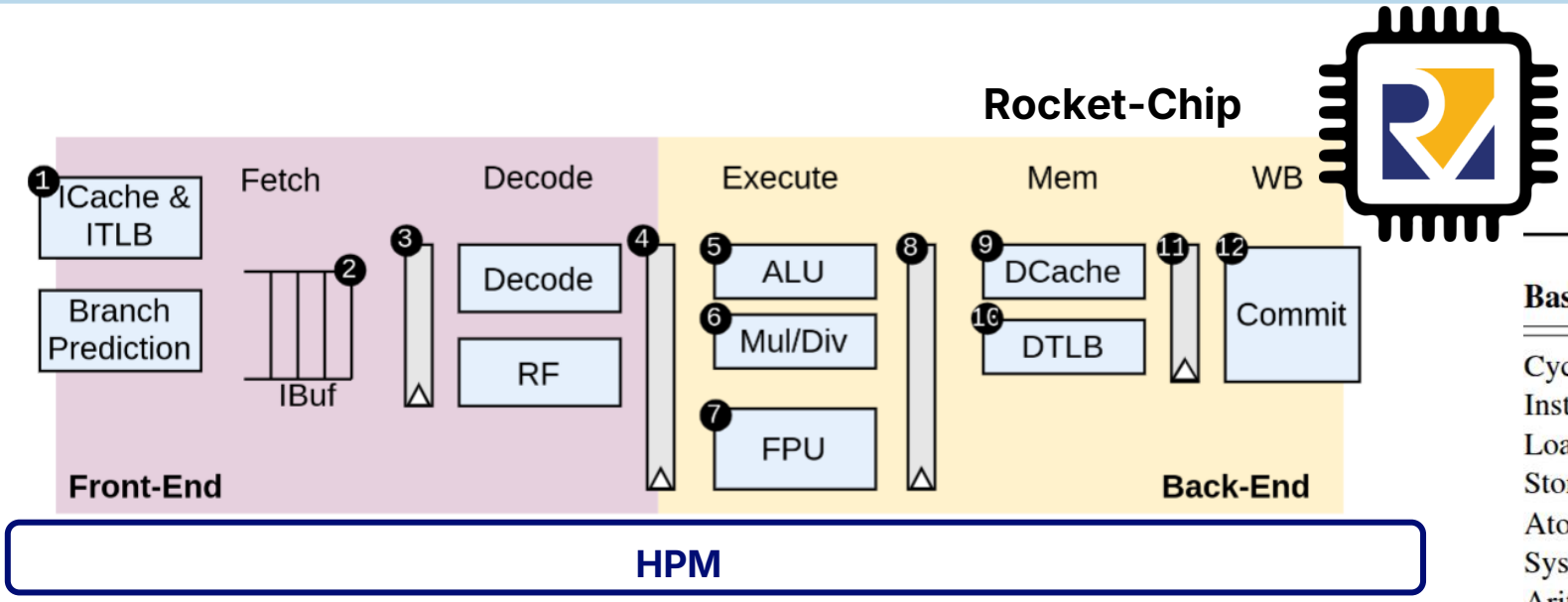
"**Good**" slot turns into "**Bad**" slot due to bad speculation (e.g., branch misprediction)



TMA

⇒ Categorize slots

What is the current state of performance characterization?

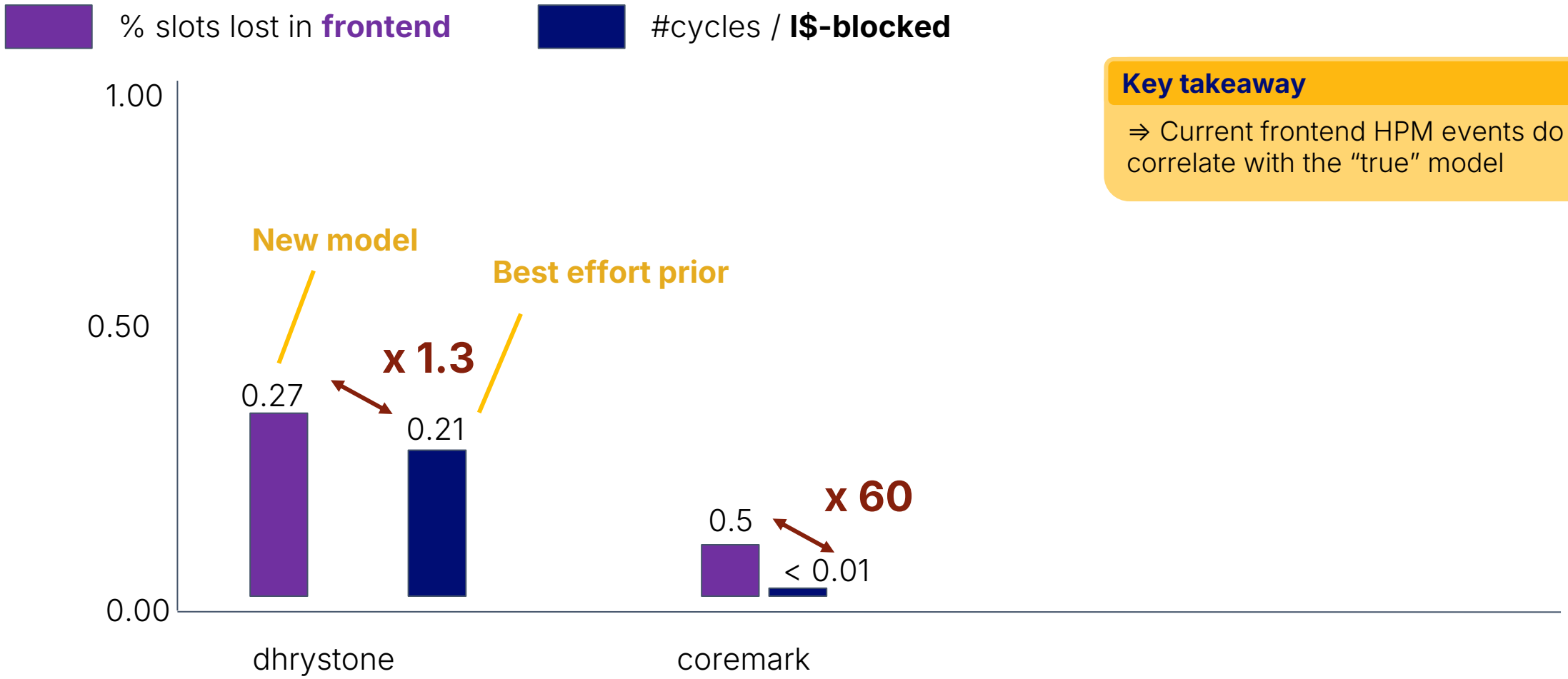


Basic	Micro-architect.	Memory
Cycles	Load-Use inter.	I$ miss 1
Instr. R.	Long-latency inter.	D$ miss 8
Load	Csr inter.	D$ release
Store	I$-blocked 1	ITLB miss 1
Atomic	D$-blocked 9	DTLB miss 10
System	Br mispred.† 8 7	L2 TLB miss
Arith	CF target mispred.† 8	
Branch	Flush† 11	
Fence†	Replay 11	
...	Mul/Div interlock 6	
	FP inter. 7	

Question:

Is RocketChip HPM interface expressive enough to give us accurate performance interface?

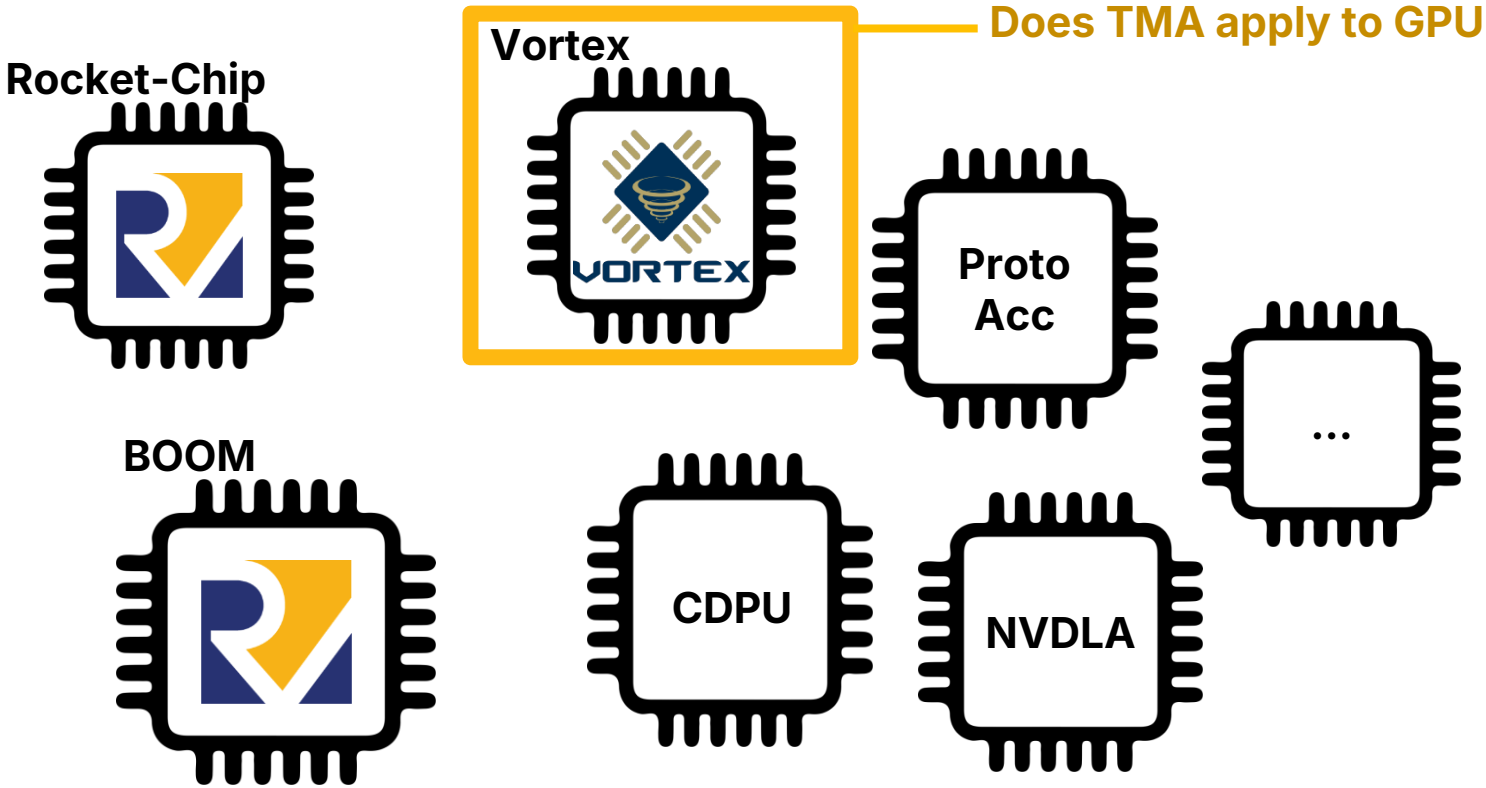
Is the current HPM implementation enough?



Key takeaway

⇒ Current frontend HPM events do not correlate with the "true" model

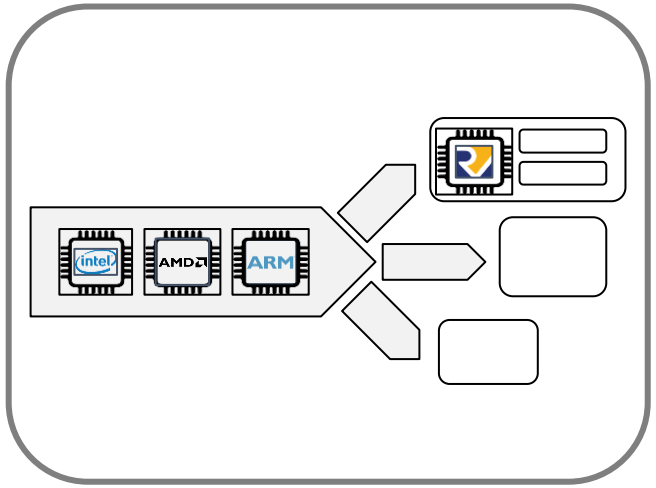
Is this an isolated incident?



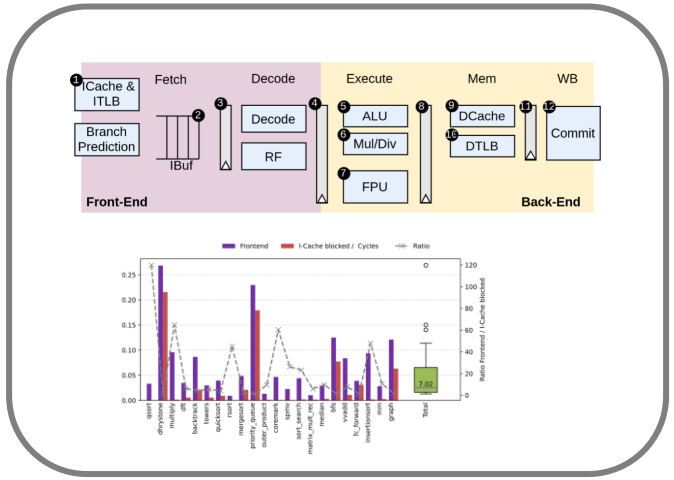
Key takeaway

⇒ Performance interfaces for most open source hardware are insufficient!

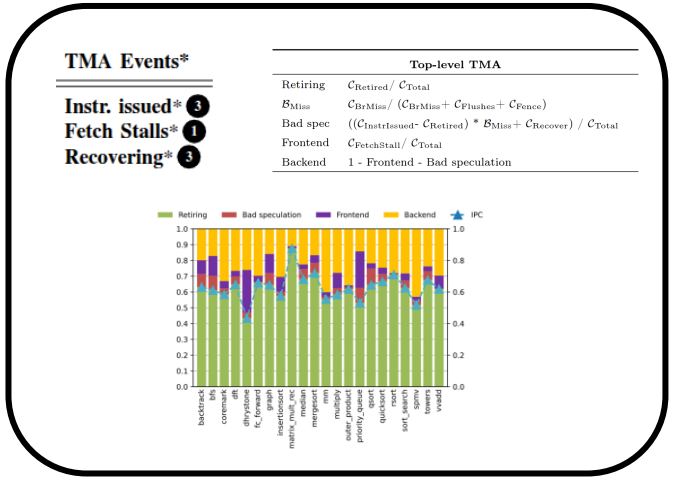
Outline



Introduction & motivation



Performance characterization on RISC-V

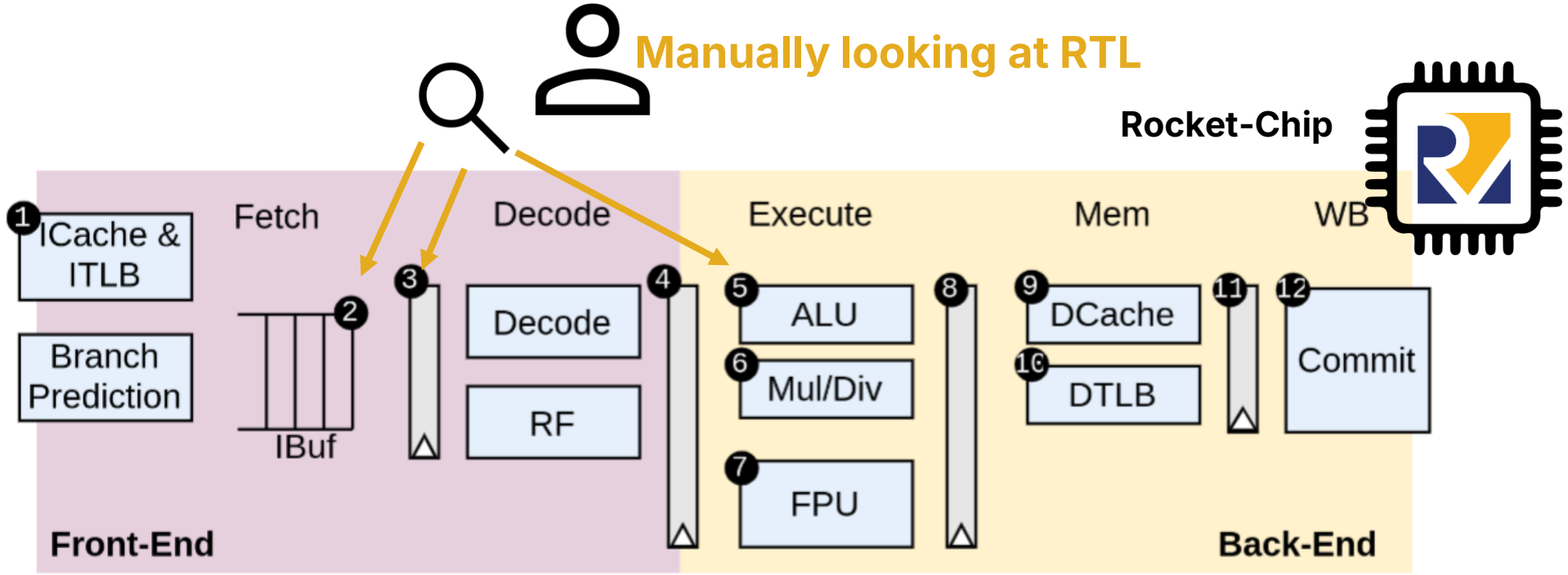


Methodology & results

COLUMBIA ENGINEERING

ICE

What is our current methodology?



TMA Events*
Instr. issued* 3
Fetch Stalls* 1
Recovering* 3

Top-level TMA	
Retiring	$\mathcal{C}_{\text{Retired}} / \mathcal{C}_{\text{Total}}$
$\mathcal{B}_{\text{Miss}}$	$\mathcal{C}_{\text{BrMiss}} / (\mathcal{C}_{\text{BrMiss}} + \mathcal{C}_{\text{Flushes}} + \mathcal{C}_{\text{Fence}})$
Bad spec	$((\mathcal{C}_{\text{InstrIssued}} - \mathcal{C}_{\text{Retired}}) * \mathcal{B}_{\text{Miss}} + \mathcal{C}_{\text{Recover}}) / \mathcal{C}_{\text{Total}}$
Frontend	$\mathcal{C}_{\text{FetchStall}} / \mathcal{C}_{\text{Total}}$
Backend	1 - Frontend - Bad speculation

Key takeaway

⇒ This process takes multiple 2-3 months of work

What methodology is used to evaluate correctness?

⇒ **Case studies to show:**

(1) Can we accurately identify bottlenecks?

(1) Can we accurately predict speedups after applying optimizations?

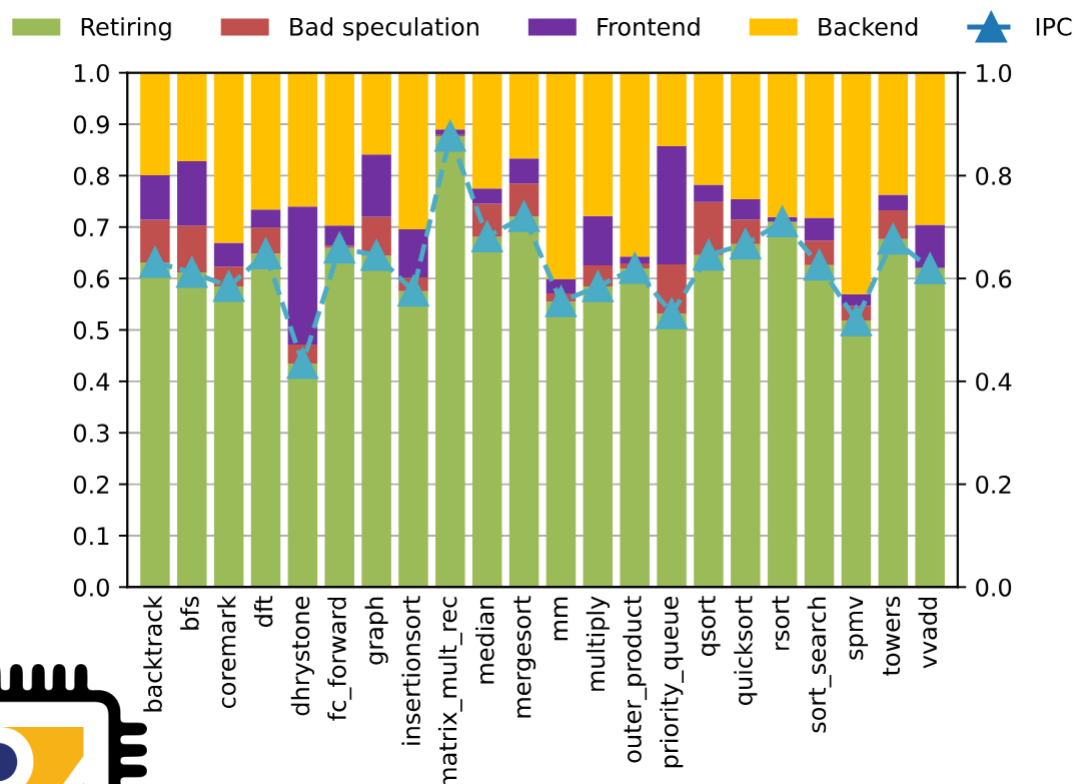
Synthetic benchmark

```
2   beq a5,zero,.L1
3   nop
4   j .L2
5   nop
6   j .L2
7   .L2:
8   beq a5,zero,.L3
```

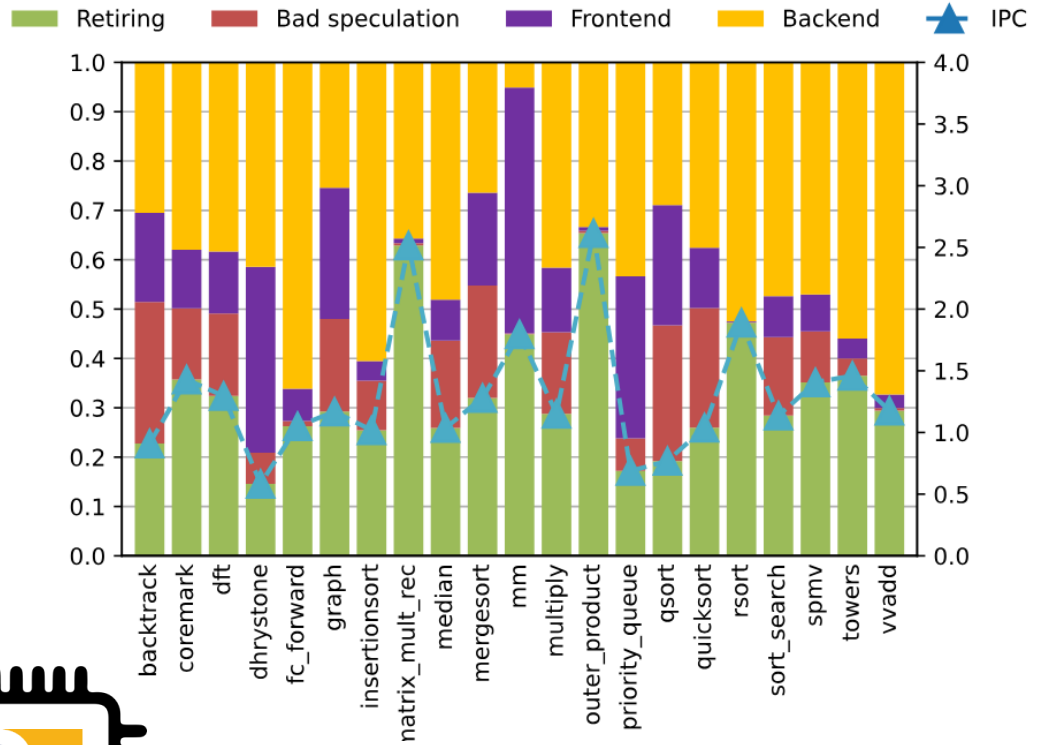
Always mispredict

Always predict correctly

What are the results? —> This is the Baseline!



Rocket-Chip

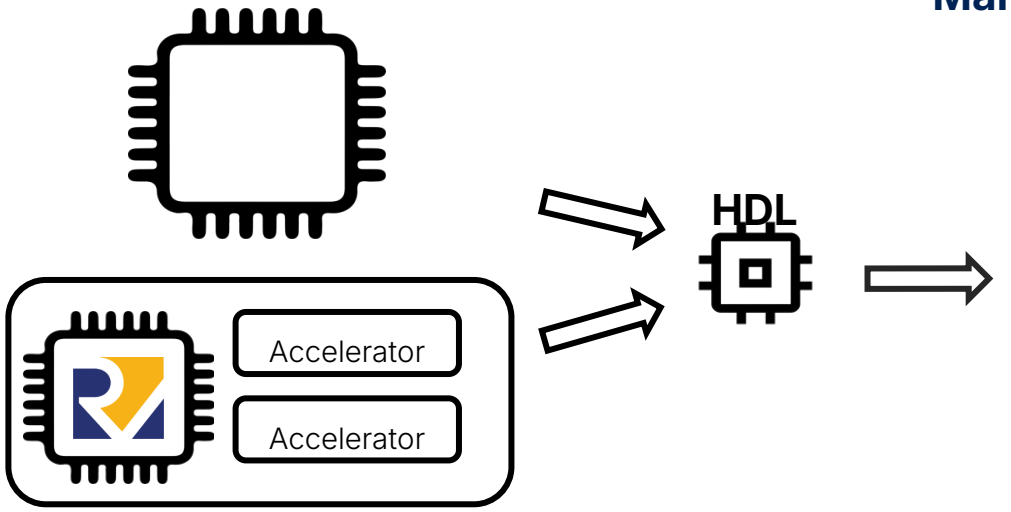


BOOM

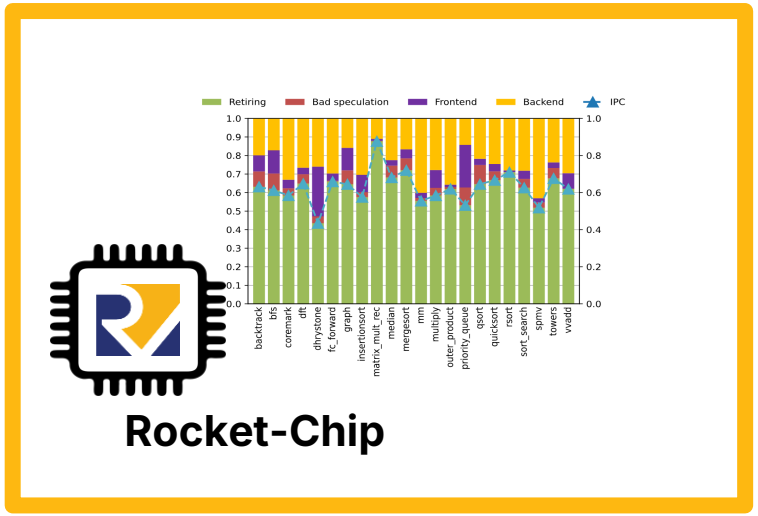
Key takeaway

⇒ (1) Creating a baseline
=> (2)Leverage insights from this design

Call for feedback: Automatic performance characterization



Manual implementations as evaluation reference



Key Problem:

Performance characterization requires **lots of effort, hard to get right**

Key Vision:

Fully **automatic** and **generalized** performance characterization

Key Contribution:

Performance characterization HPM events for **RocketCore** and **BOOM**

mattweingarten.github.io/

Contact: matthew.weingarten@columbia.edu