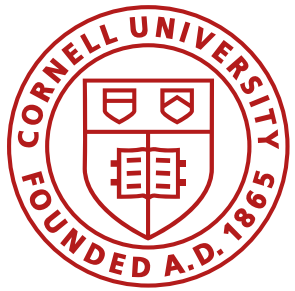
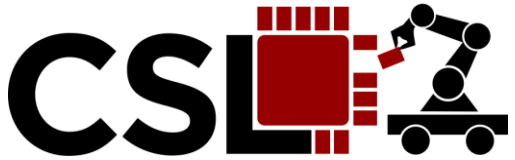


XRT: An Accelerator-Aware Runtime for Accelerated Chip Multiprocessors

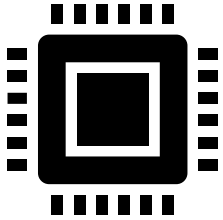
Neel Patel, Mohammad Alian (Cornell University)

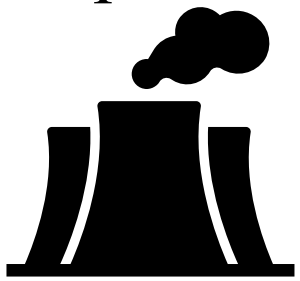
CornellEngineering
Electrical and Computer Engineering



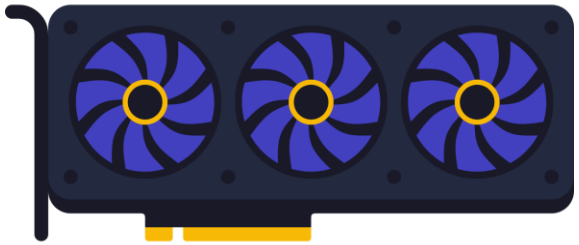
The Need for Hardware Specialization

Datacenters demand compute and energy



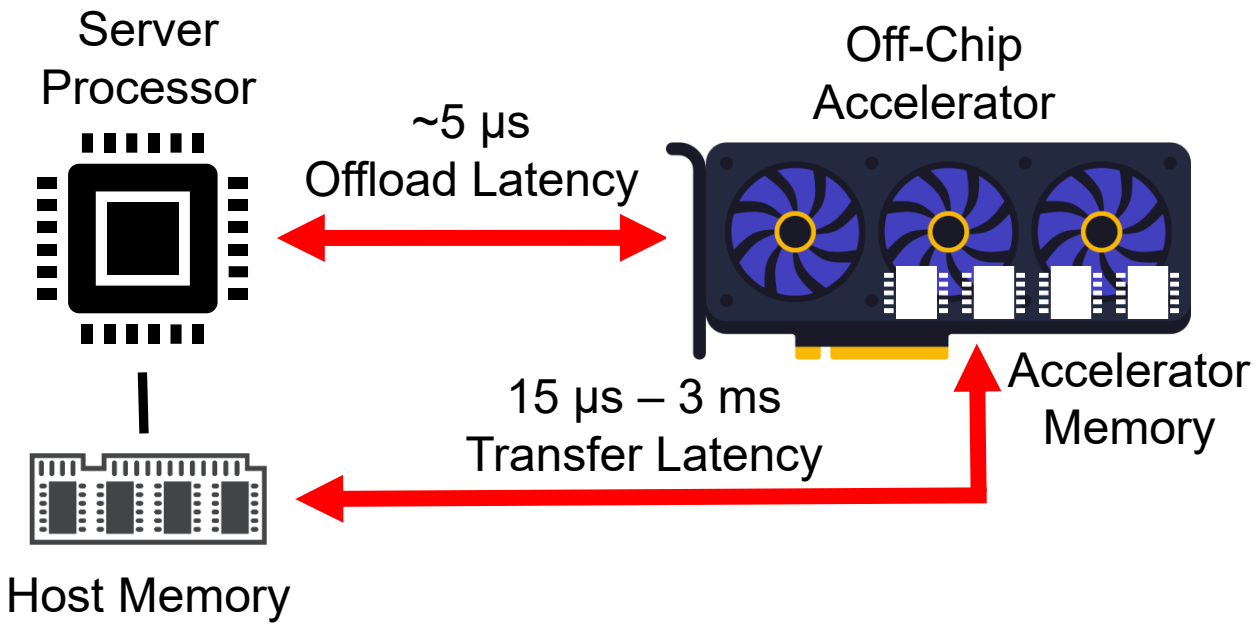


Slowing of Moore's Law hinders improvement of general-purpose cores

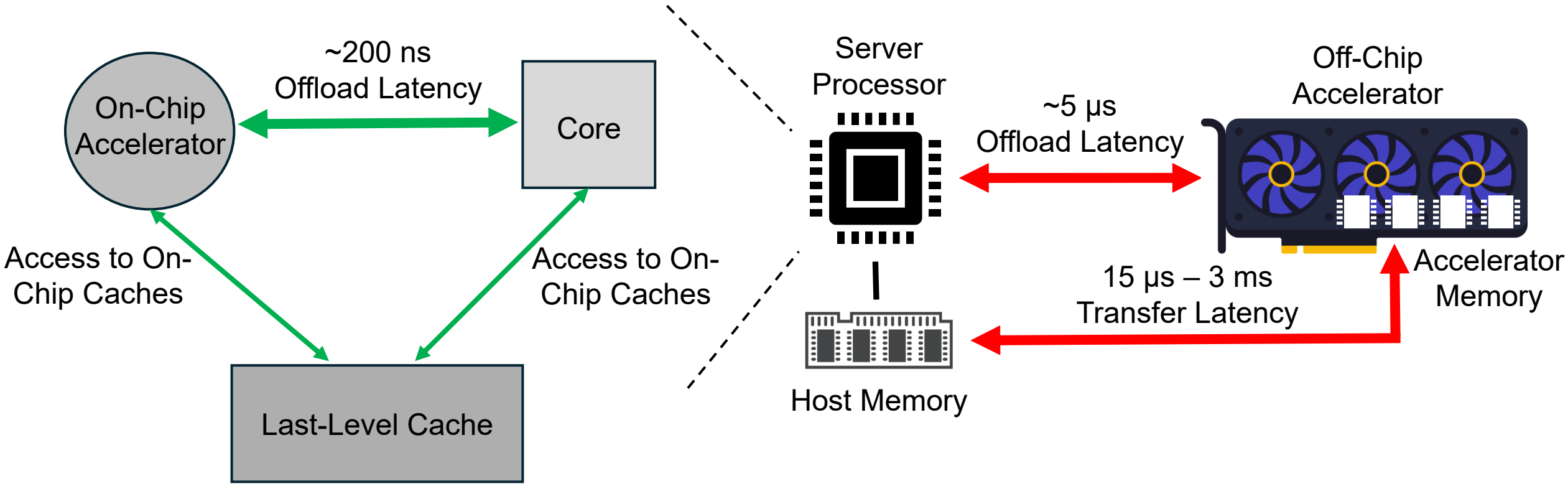


Accelerators provide higher performance and energy efficiency than general-purpose cores

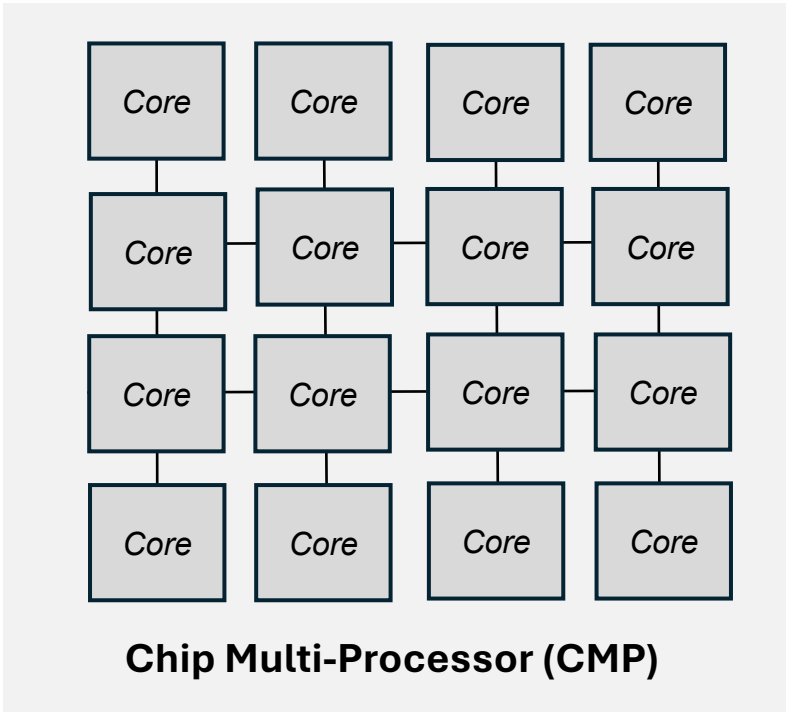
The Benefits of On-Chip Acceleration

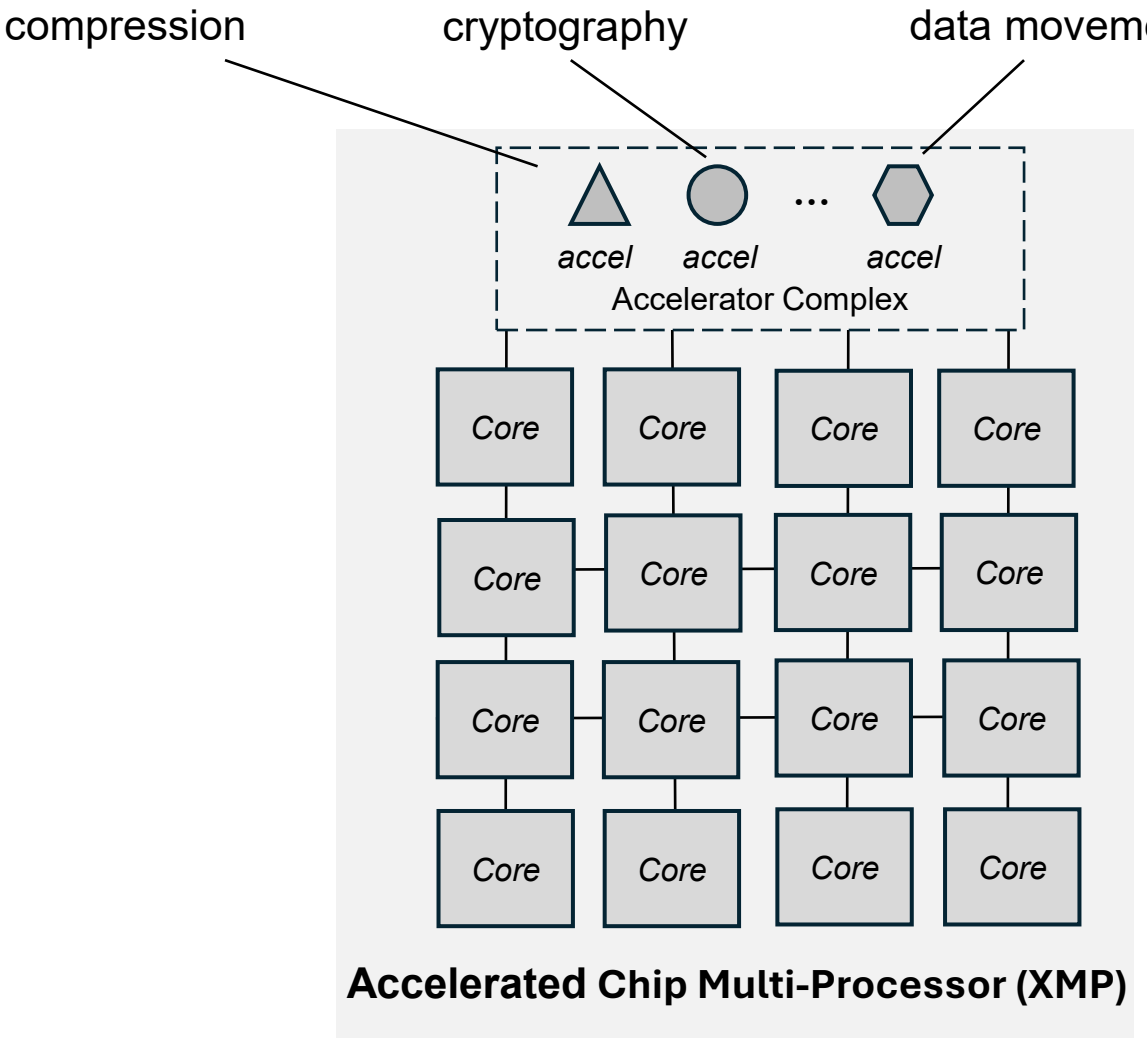


The Benefits of On-Chip Acceleration



On-Chip Acceleration on Server Processors





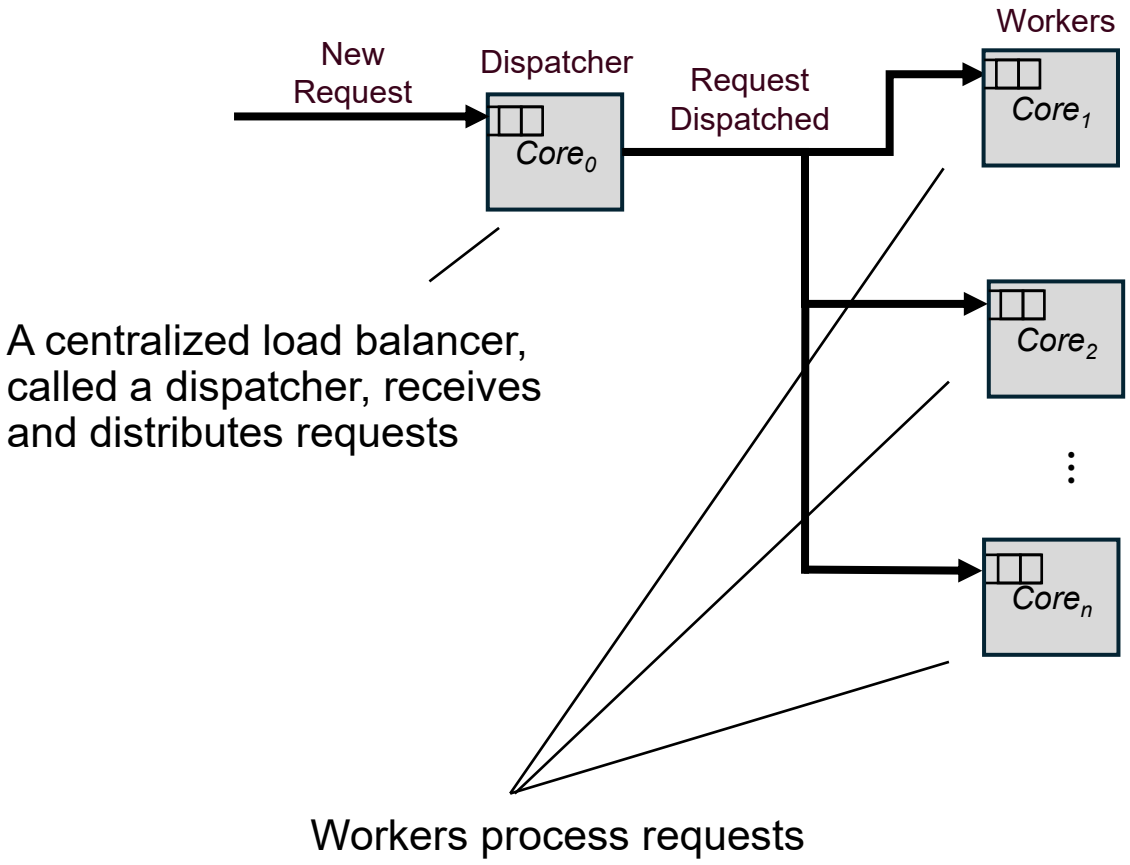
Goal and XRT's Contributions

Goal: Enable microsecond-scale datacenter applications to get the performance and efficiency benefits of on-chip acceleration

XRT's Contributions: A runtime for XMPs

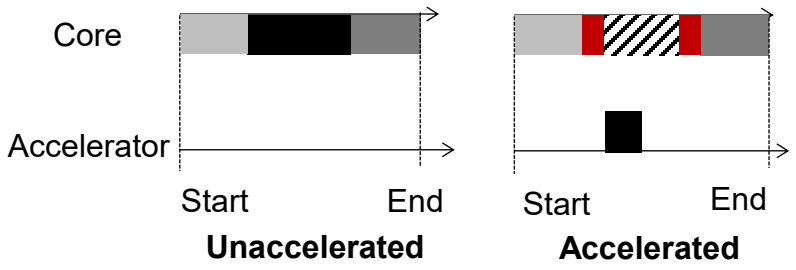
- **Distributes overhead of accelerator-to-core communication** across many cores
- Avoids stalls due to overloaded accelerators by **falling back to software** when necessary
- **Eliminates unnecessary context switches** when threads are blocked on an accelerator

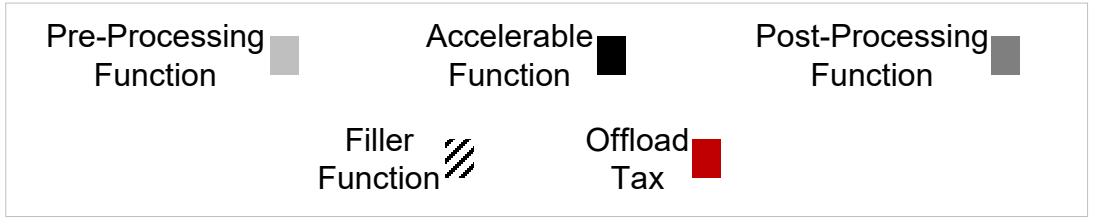
How are CMP Runtimes Designed?

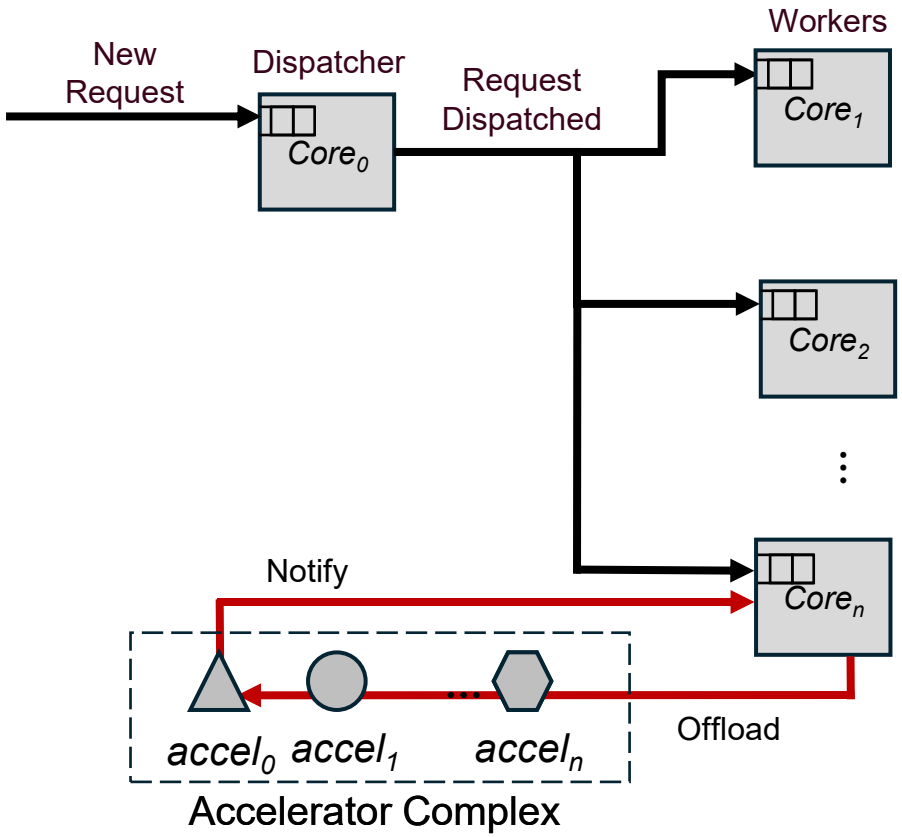


Request Processing on an XMP

On an XMP, requests execute on both the cores and the accelerators:







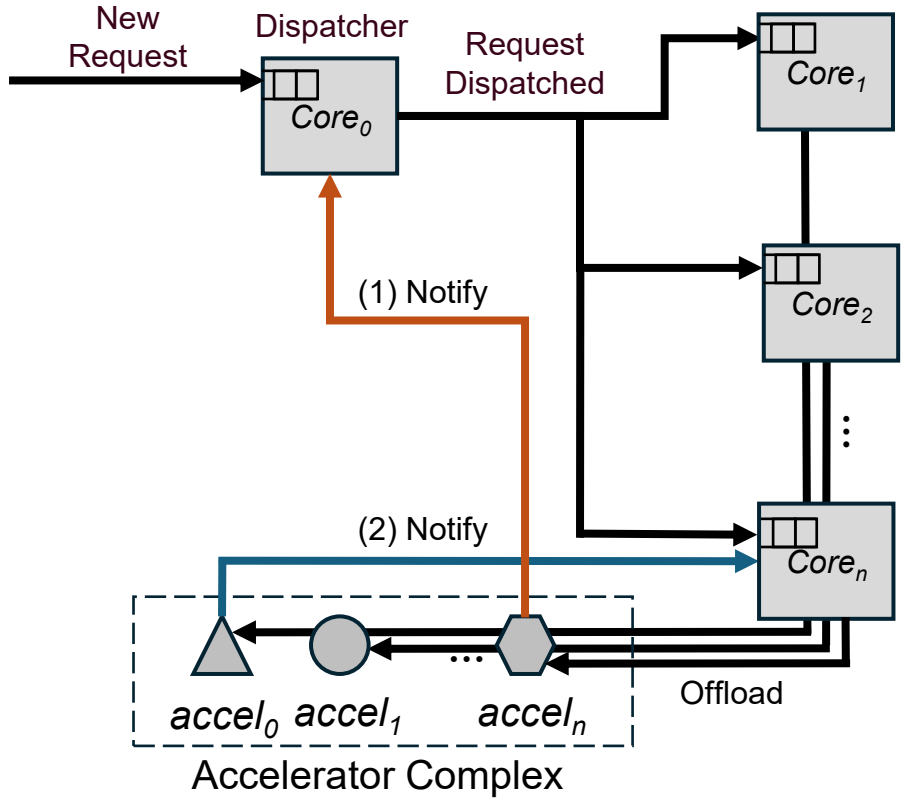
Where to Process Notifications?

Dispatcher-centric: Accelerators notify the centralized dispatcher

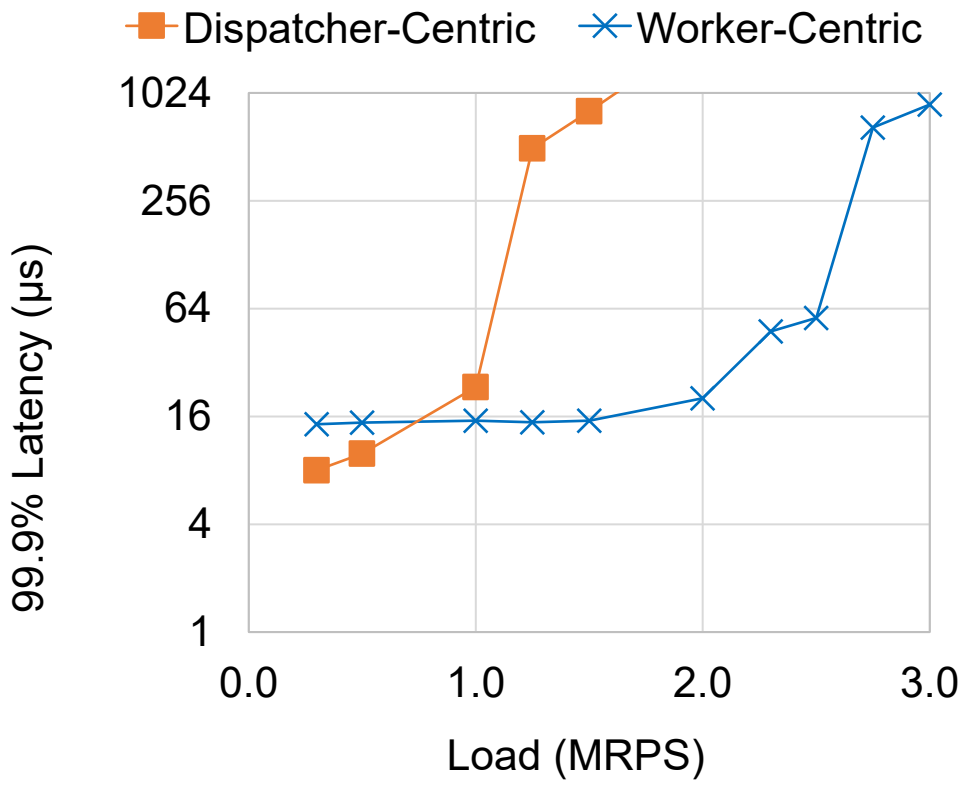
- Benefit: Global view when making scheduling decisions

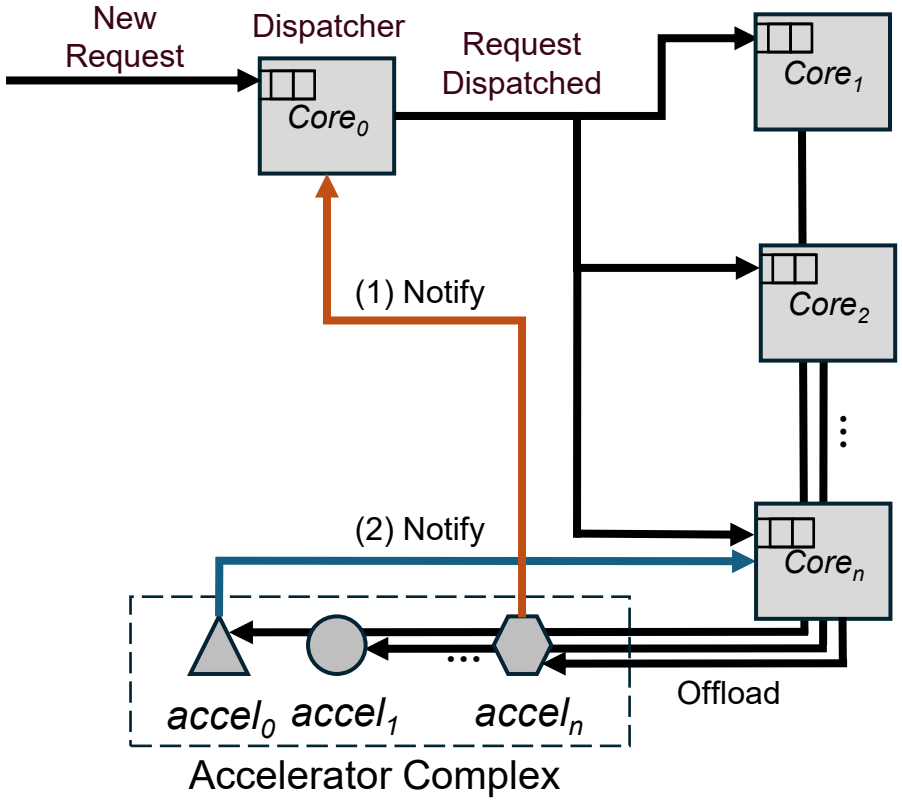
Worker-centric: Accelerators notify the offloading worker core

- Benefit: Offloads load balancing to accelerators



How to Design an XMP Runtime?

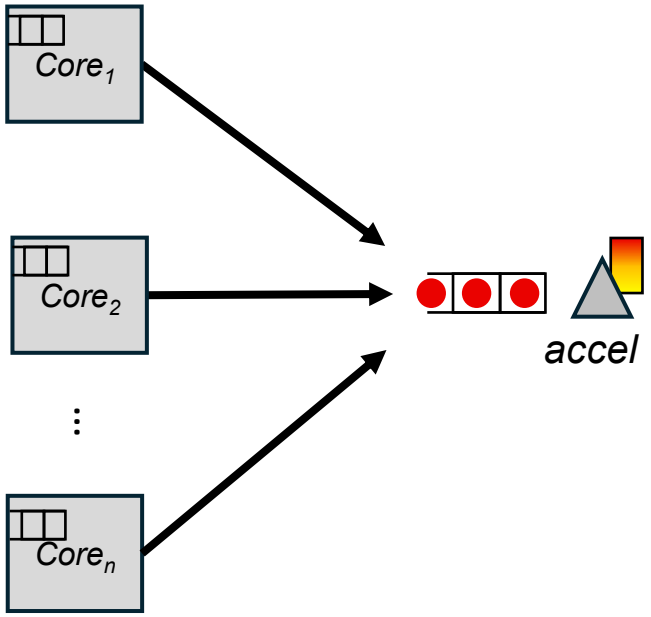


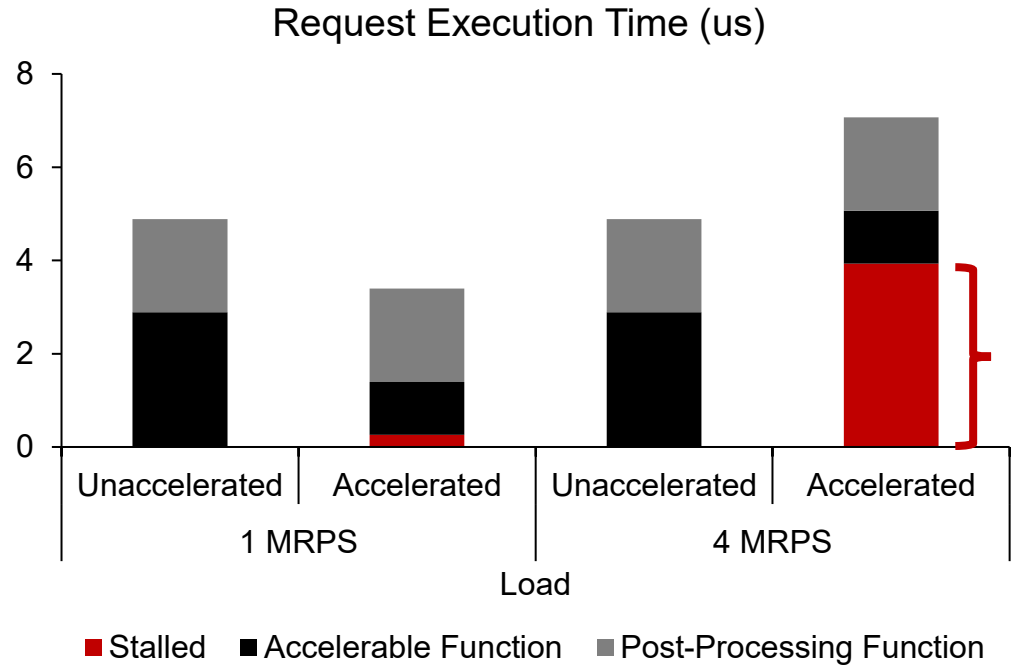


Notification signals from accelerators turn the dispatcher into the system bottleneck.

Challenge #1: Overloaded Accelerators

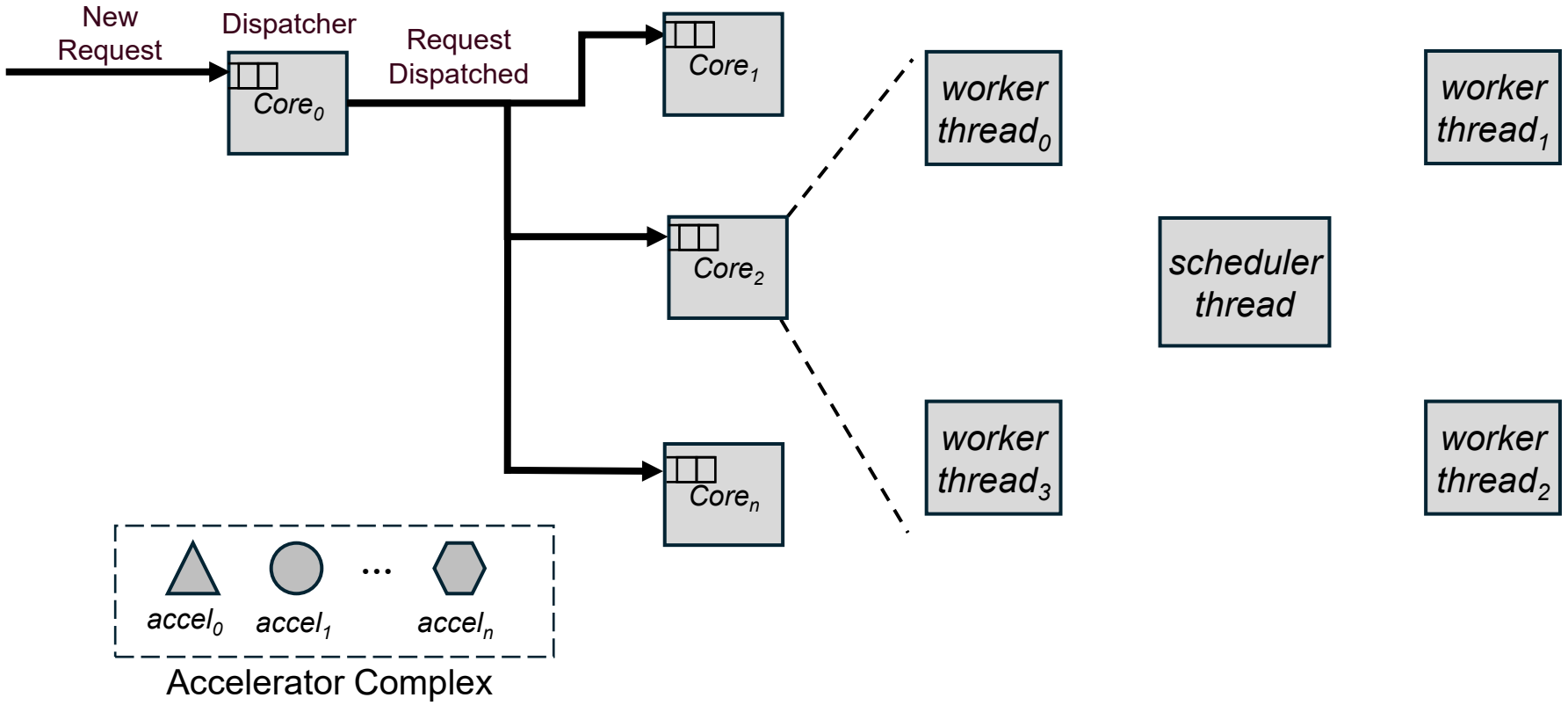
Overloaded accelerator stalls upstream cores



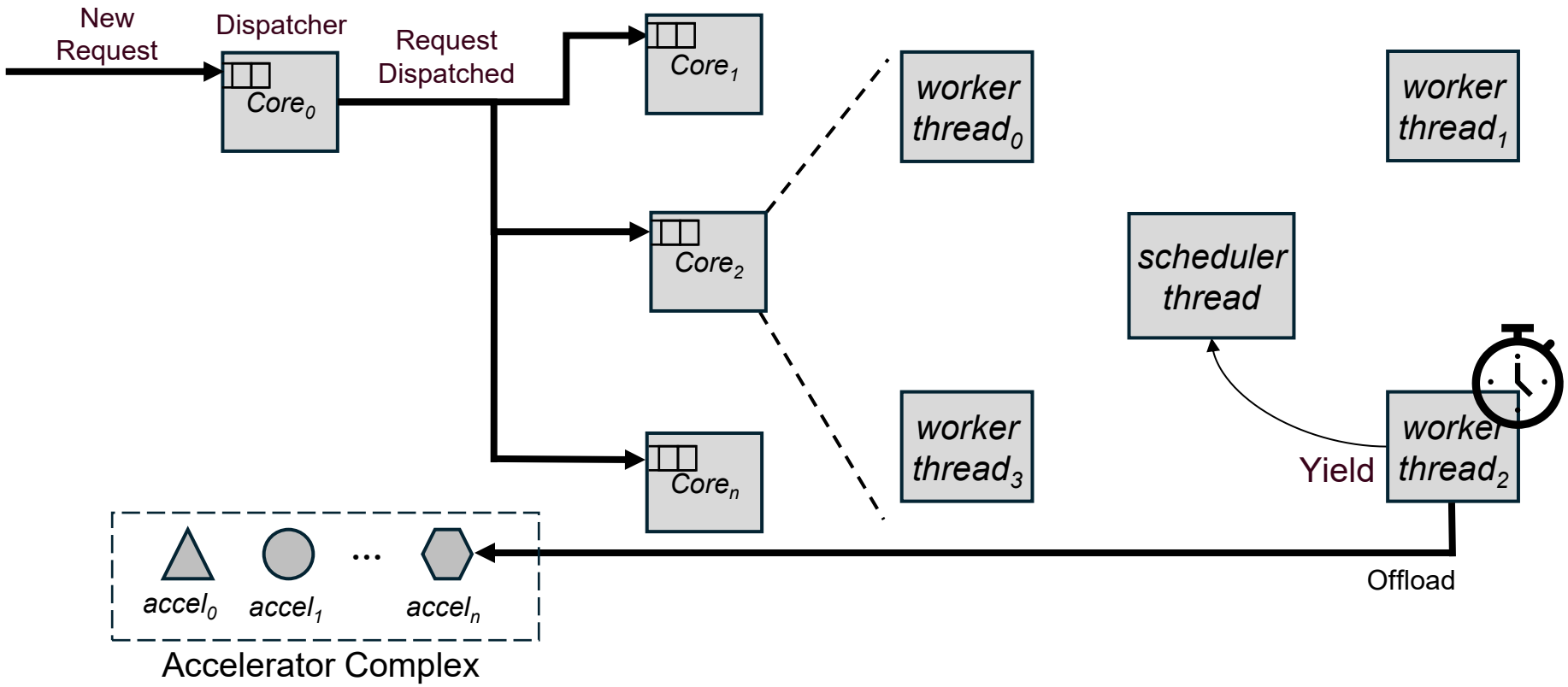


55% of CPU cycles spent stalled, waiting to offload to an accelerator

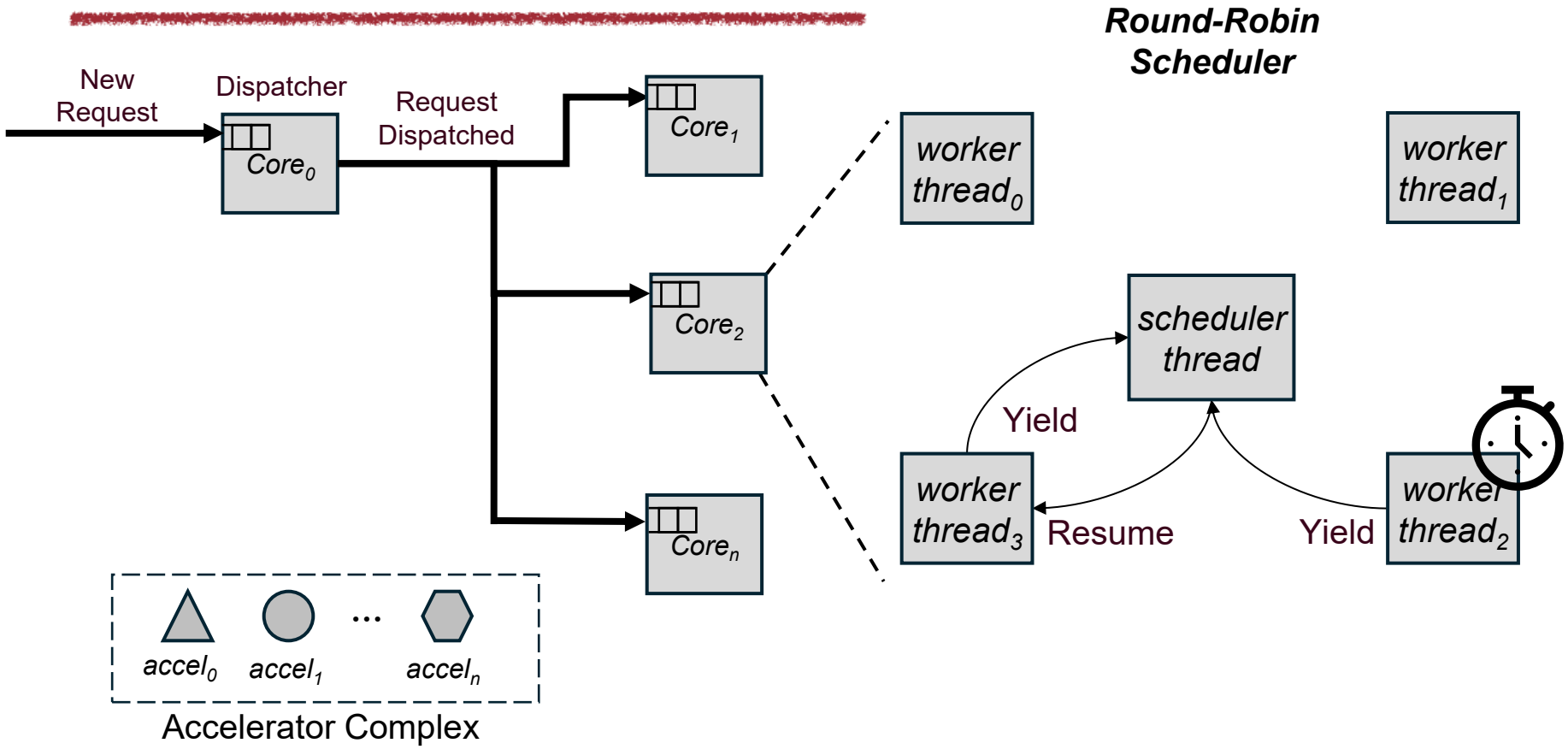
Challenge #2: Unnecessary Context Switches



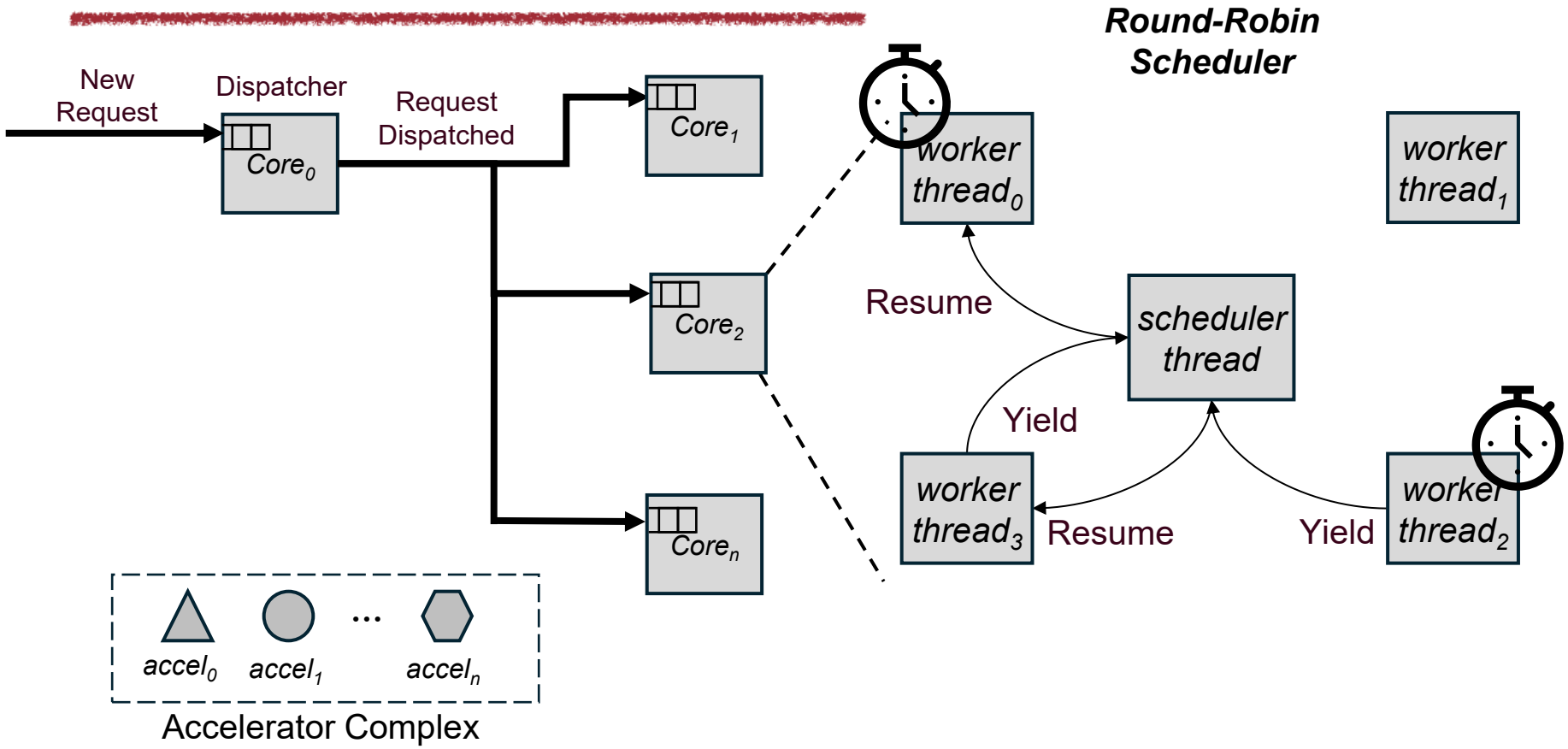
Challenge #2: Unnecessary Context Switches



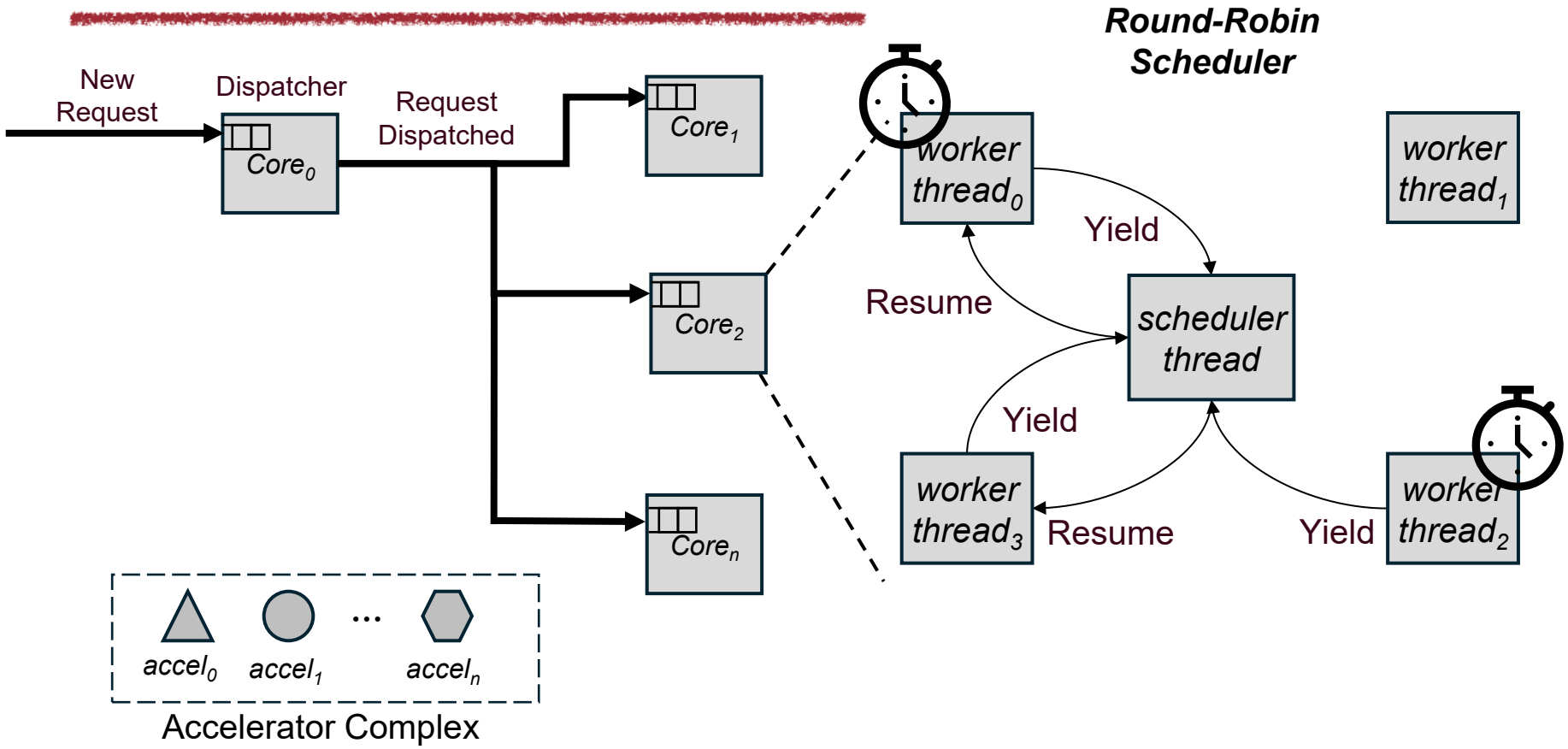
Challenge #2: Unnecessary Context Switches



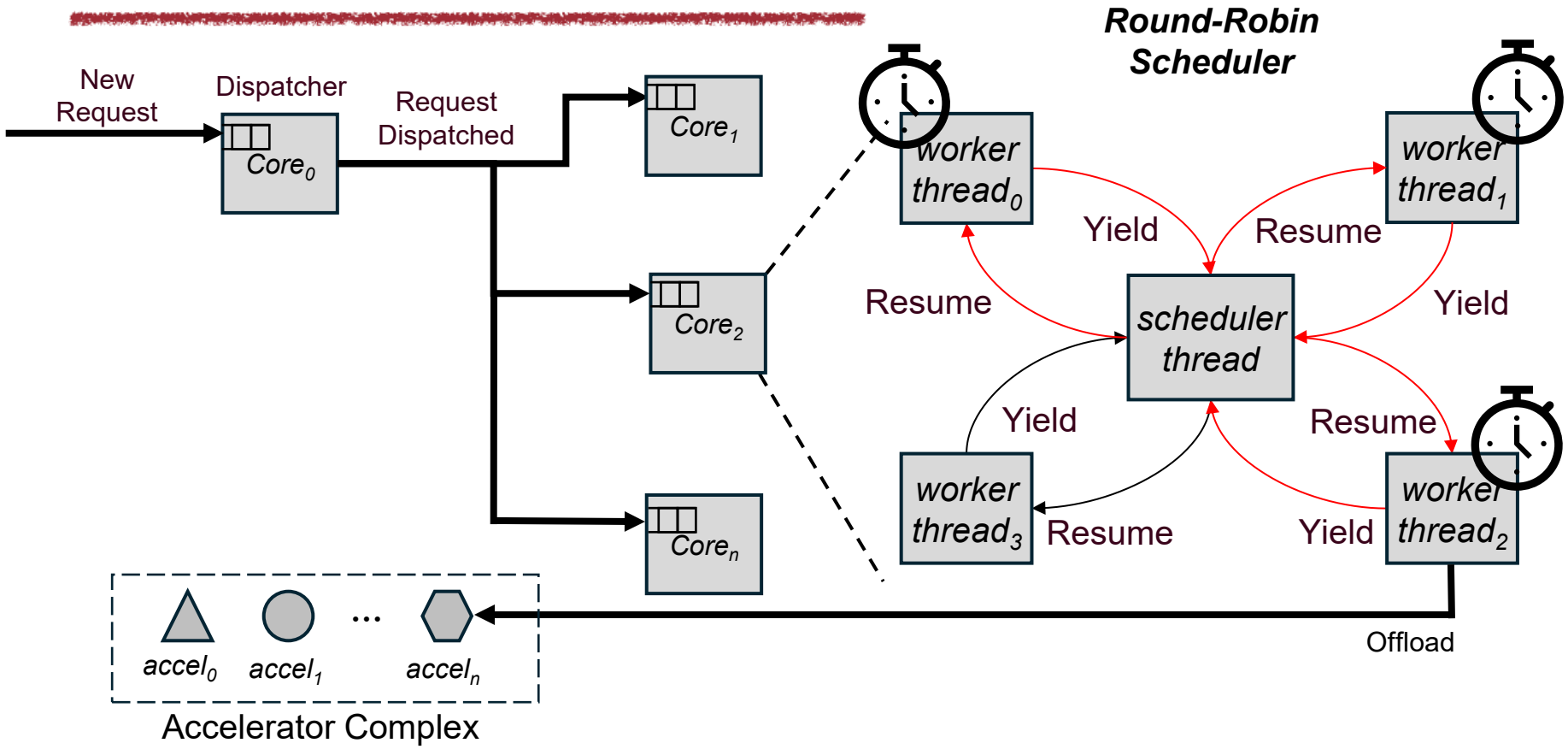
Challenge #2: Unnecessary Context Switches



Challenge #2: Unnecessary Context Switches

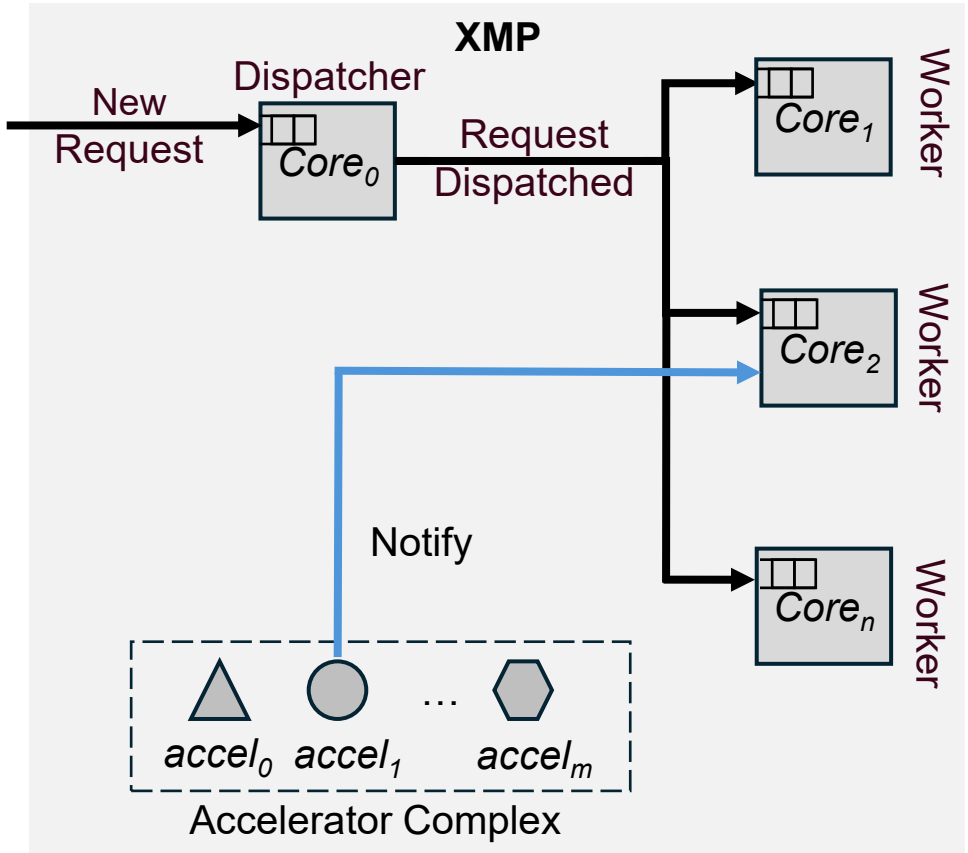


Challenge #2: Unnecessary Context Switches



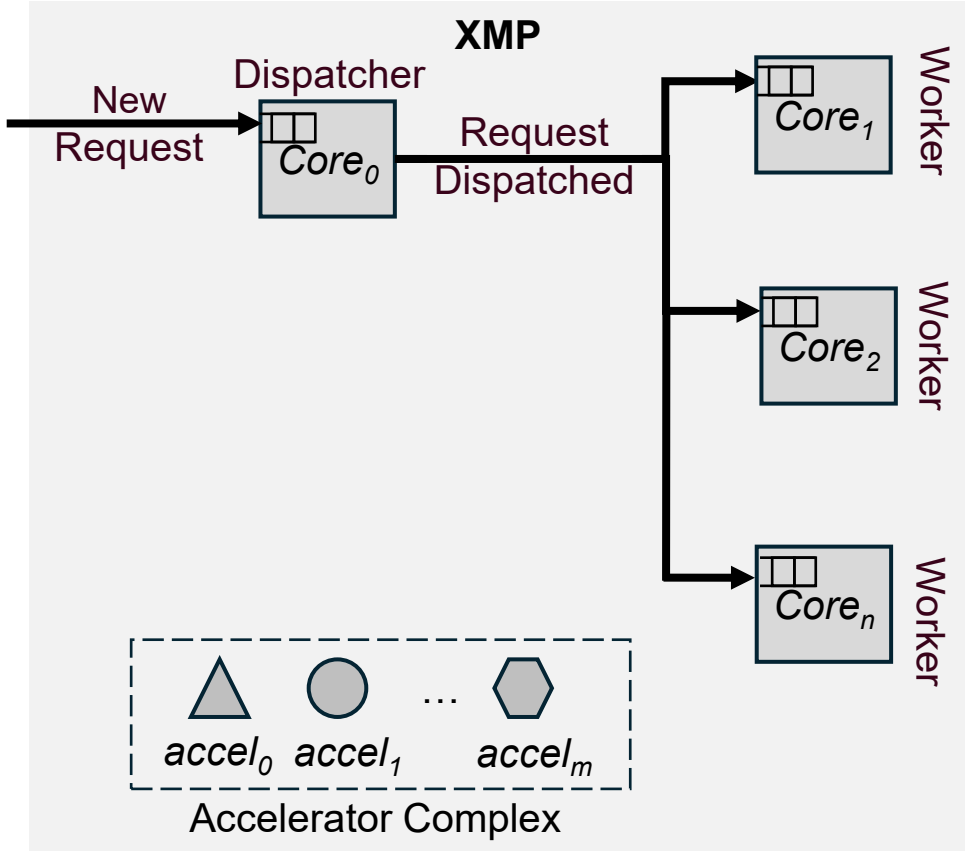
Average (mean) of ~**21 unnecessary resumptions per request** reducing throughput by ~13%

XRT: An Accelerator-Aware Runtime

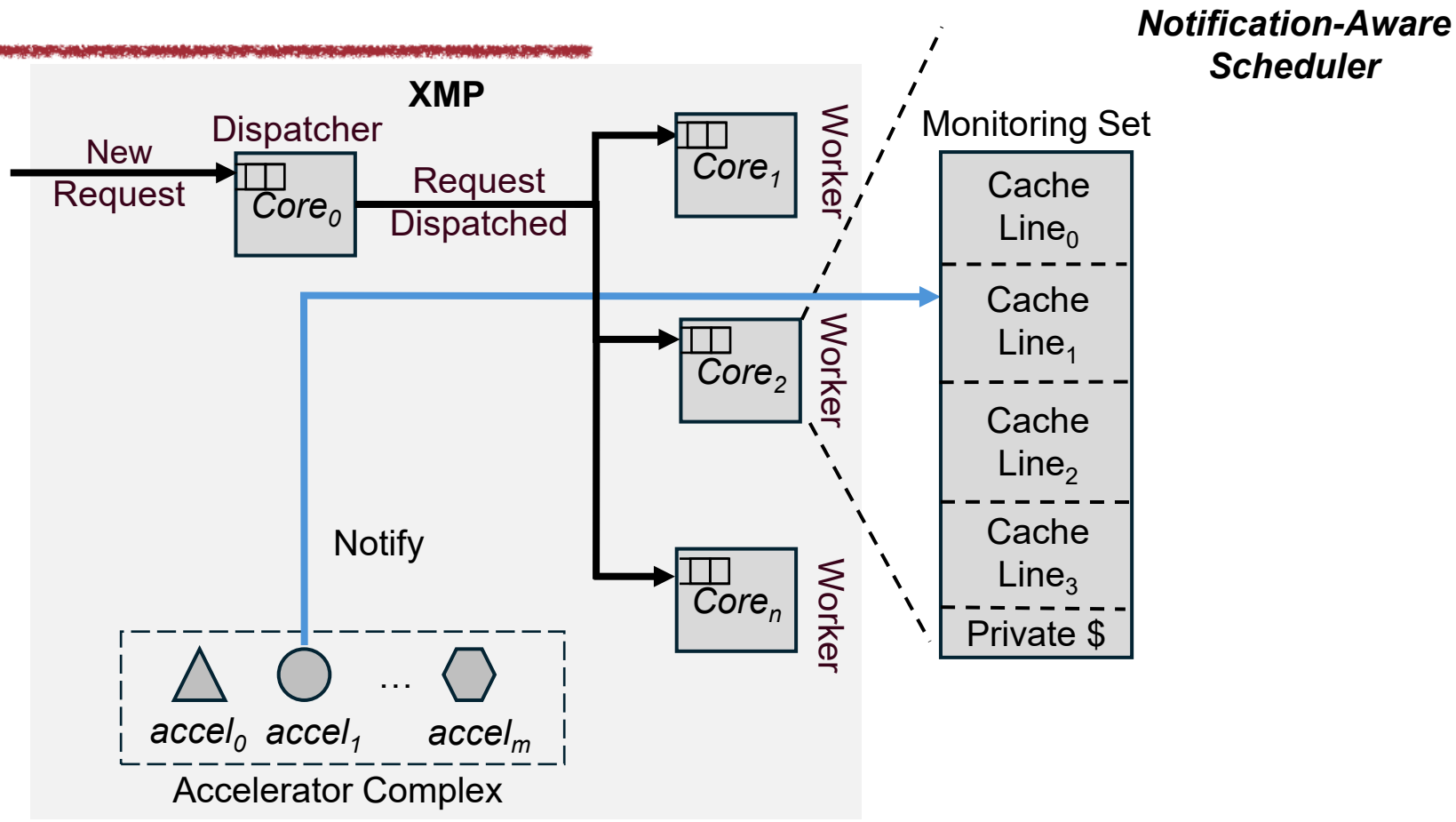


XRT: An Accelerator-Aware Runtime

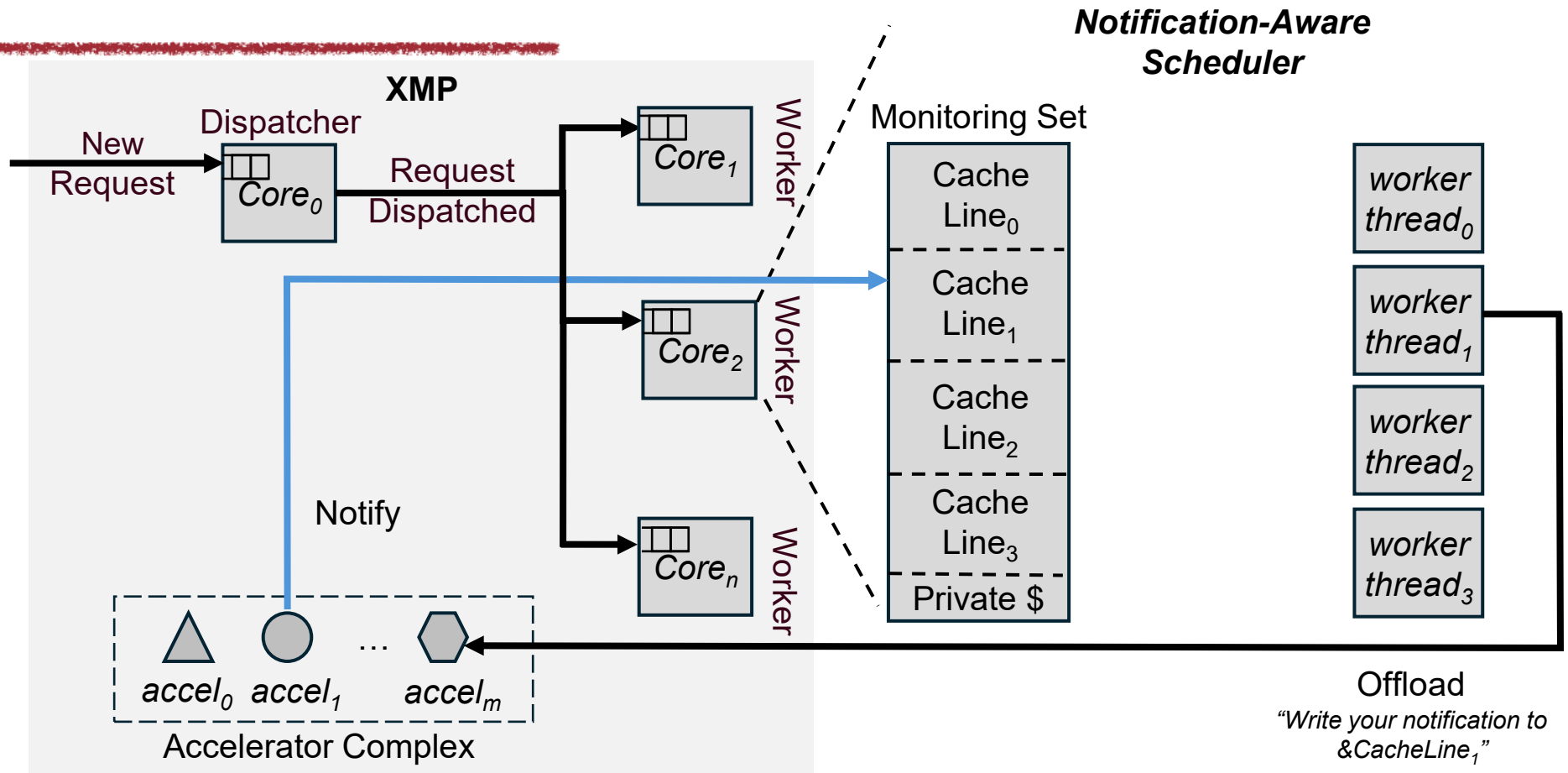
Notification-Aware Scheduler



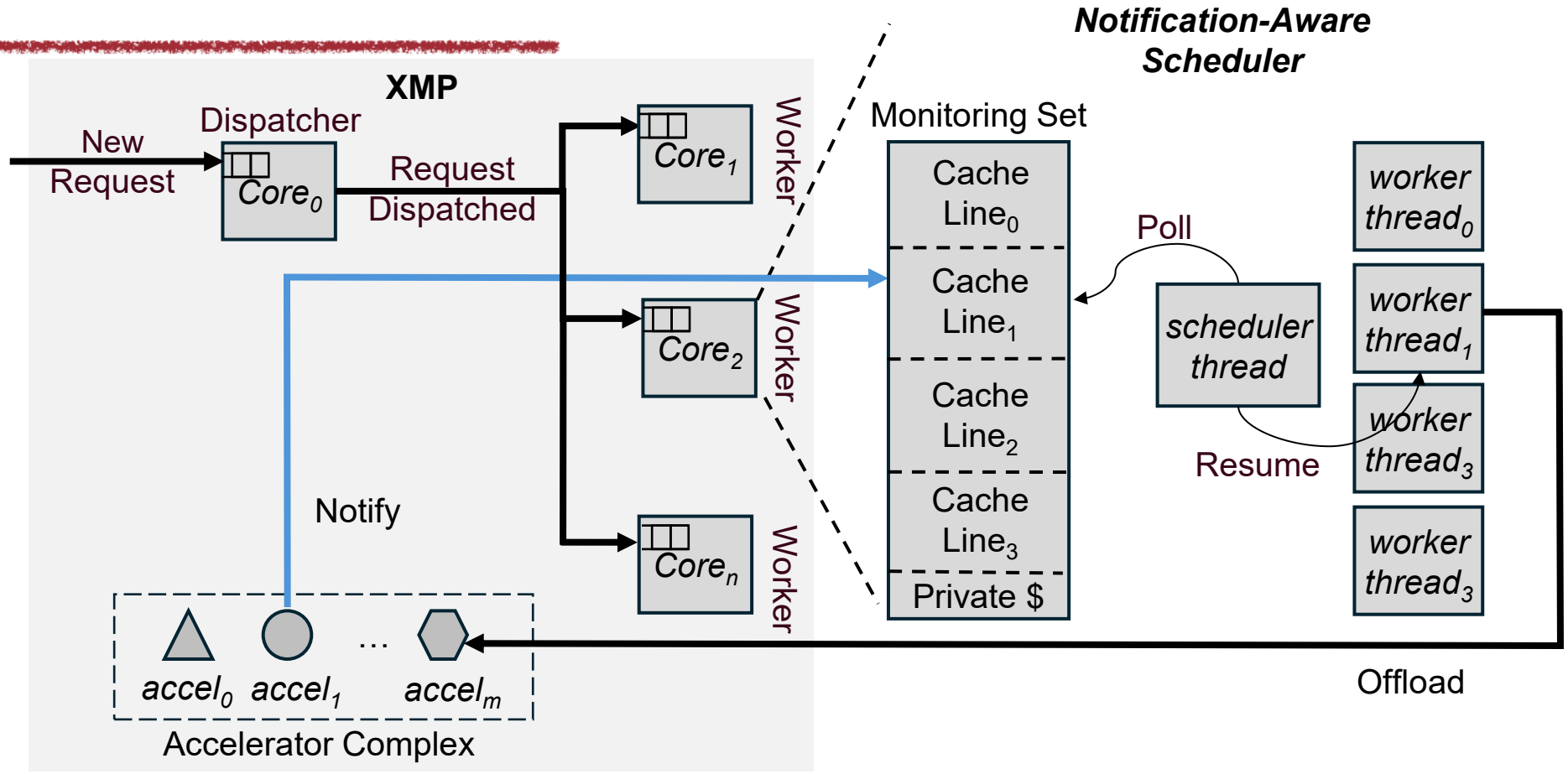
XRT: An Accelerator-Aware Runtime



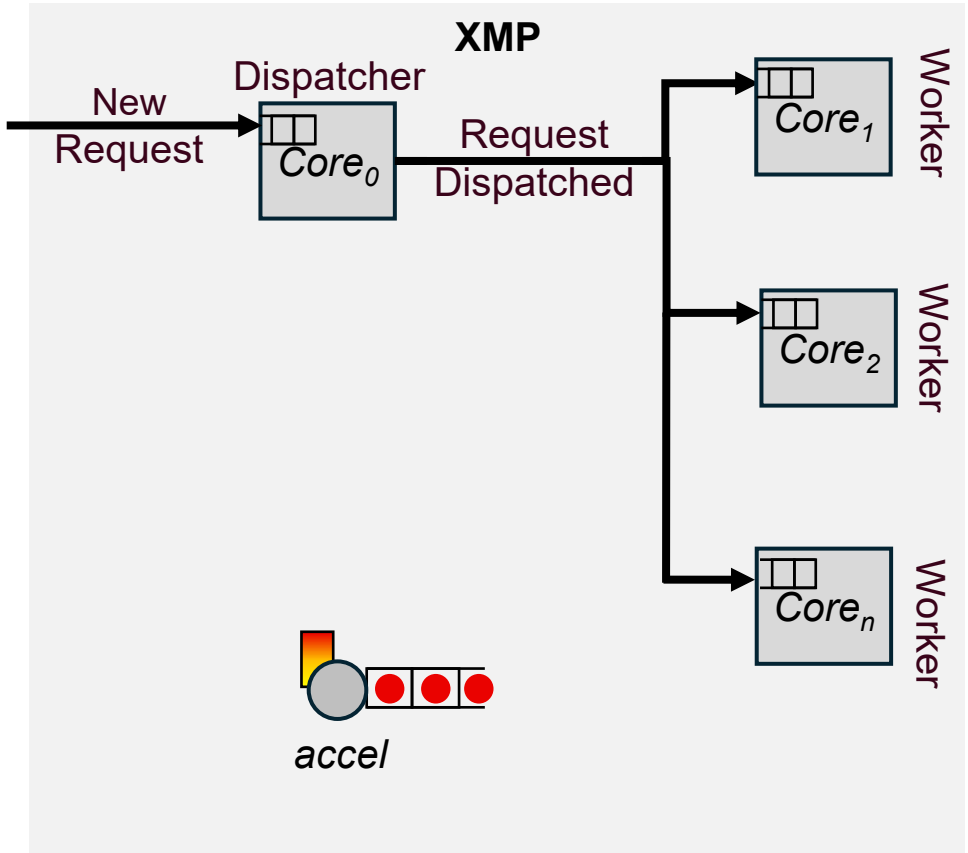
XRT: An Accelerator-Aware Runtime



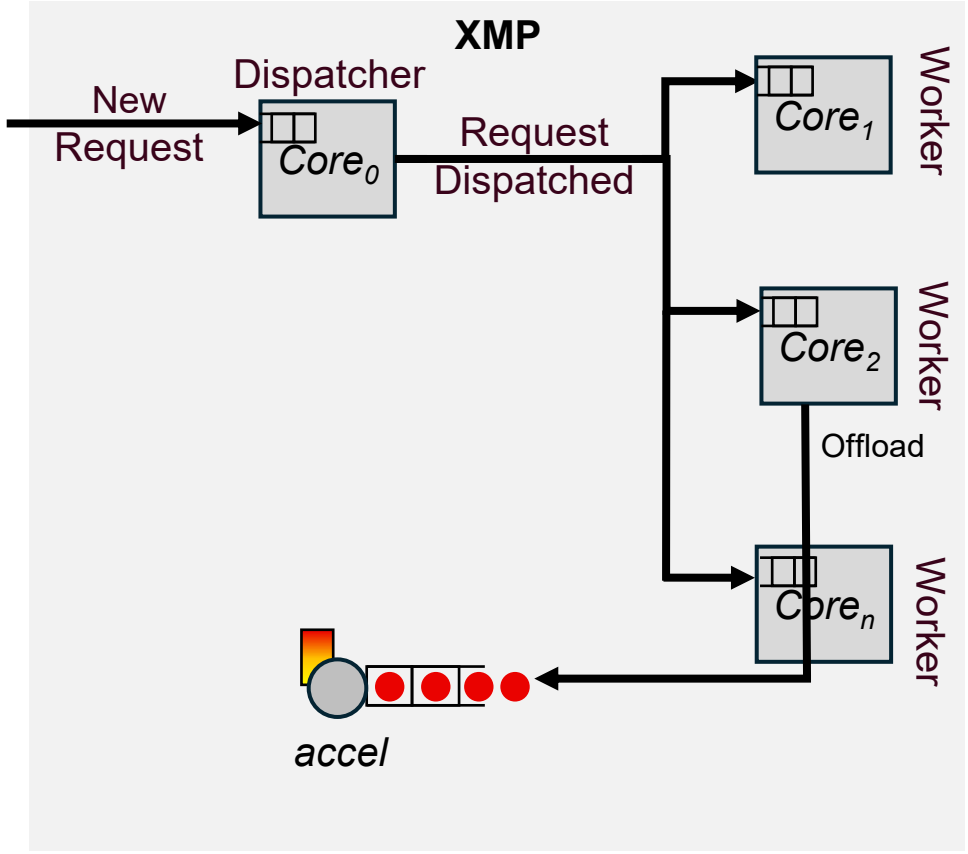
XRT: An Accelerator-Aware Runtime



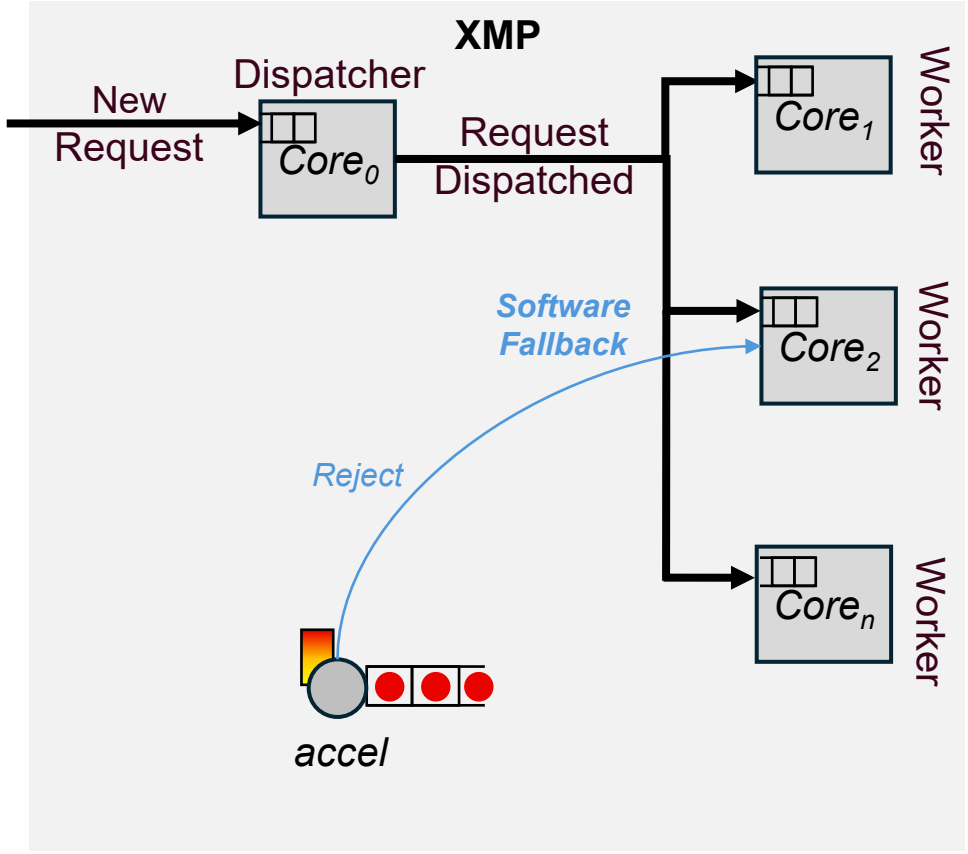
XRT: An Accelerator-Aware Runtime



XRT: An Accelerator-Aware Runtime



XRT: An Accelerator-Aware Runtime



Experimental Setup

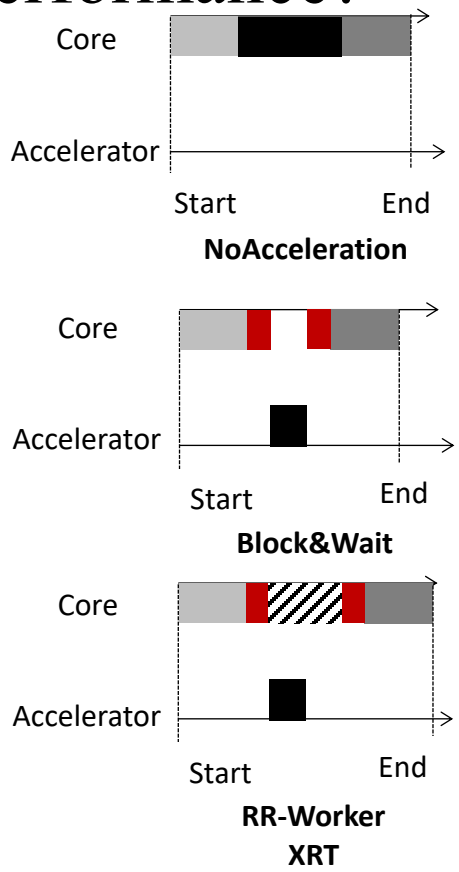
- Server: Intel Xeon 5th Gen Processor w/ 26 cores, 2 Data Streaming Accelerators (DSAs), 2 In-memory Analytics Accelerators (IAAs)
- Load generator produces Poisson arrivals
- Three-phase workloads with varying computational intensities, access patterns, and use cases for the DSA and IAA accelerators:

Workload	Pre-Processing Function	Accelerable Function	Post-Processing Function
Message Processing	Deserialize	Decompress	Hash
Similarity Scoring	Decrypt	Memcpy	DotProduct
Graph Processing	-	Decompress	PointerChase
Database	-	Memcpy	PointerChase
Data Preprocessing	MatMul	Memfill	Principal Component Analysis
Data Analytics	Update	Filter	Histogram

Evaluation Question

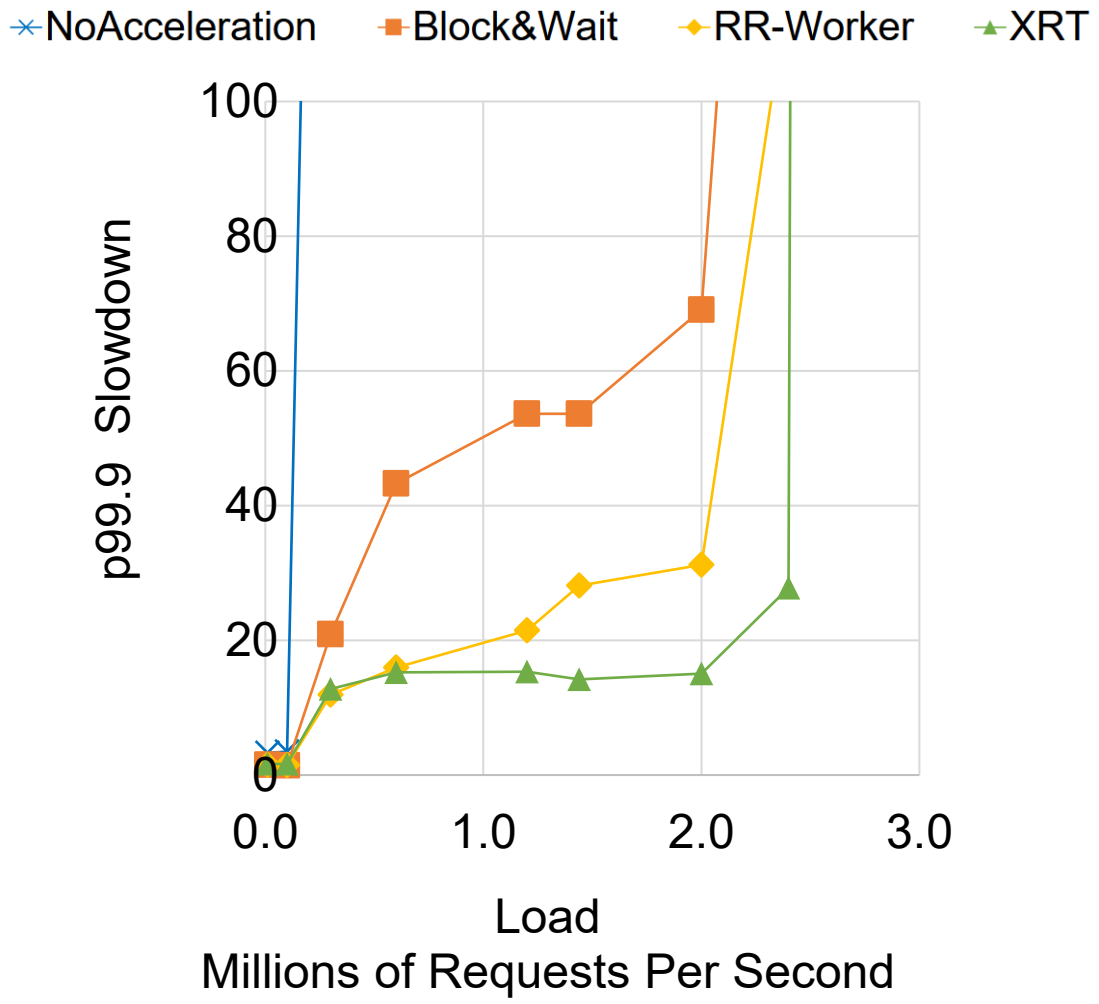
- XRT's performance compared to systems without optimizations for XMPs?
- How do XRT's optimizations contribute to its overall performance?
- Compared Systems:

System	Uses Accelerators	Yields the Core	Notification-Aware Scheduler	Software Fallback
NoAcceleration	✗	✗	✗	✗
Block&Wait	✓	✗	✗	✗
RR-Worker	✓	✓	✗	✗
XRT	✓	✓	✓	✓



Case 1: Highly Accelerable

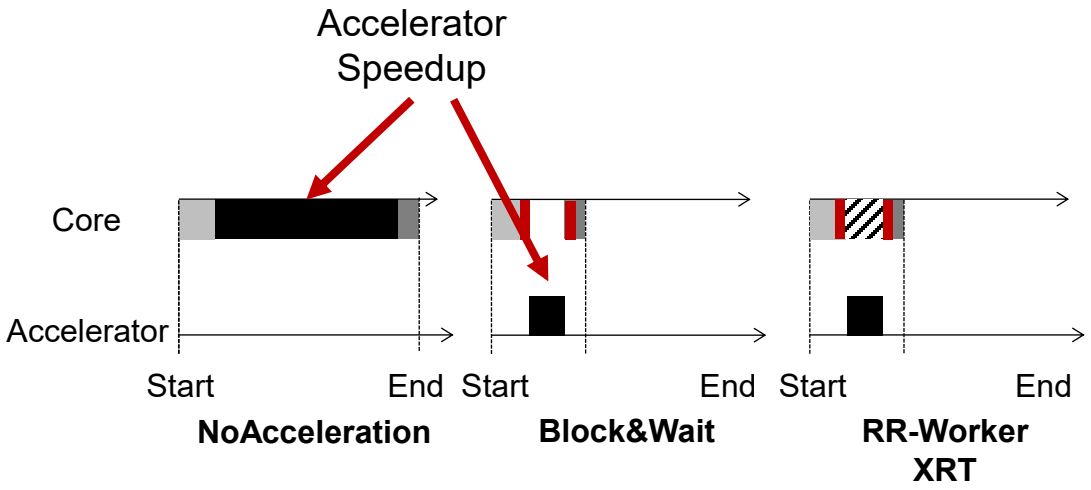
$$slowdown = \frac{Response\ Time}{Unloaded\ Accelerated\ Execution\ Time}$$

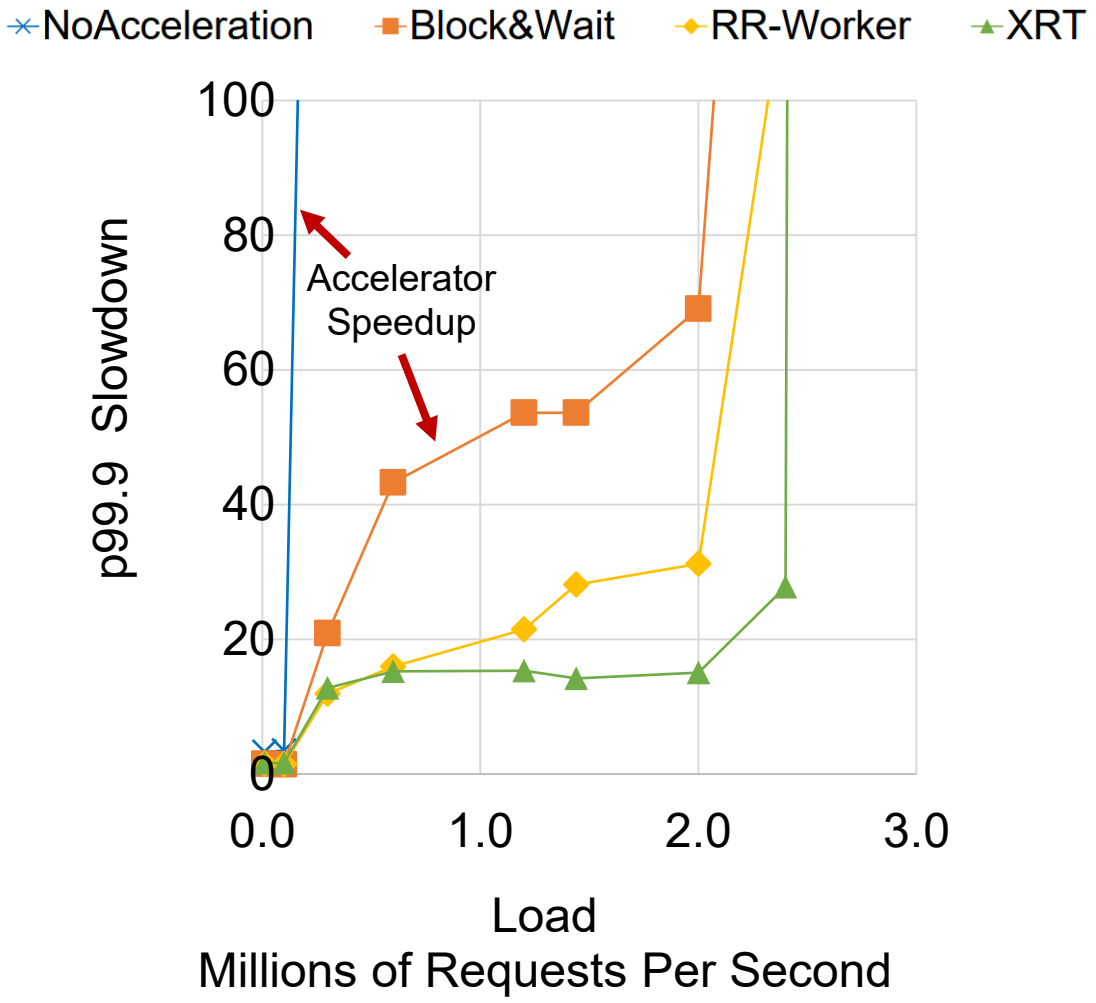


Accelerable → **Deserialize-Decompress-Hash**

Case 1: Highly Accelerable

$$slowdown = \frac{Response\ Time}{Unloaded\ Accelerated\ Execution\ Time}$$

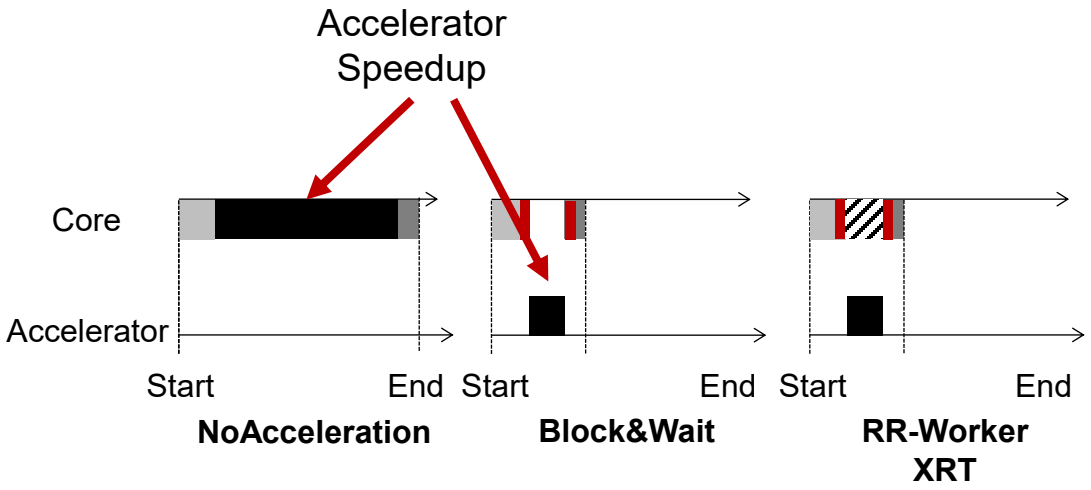


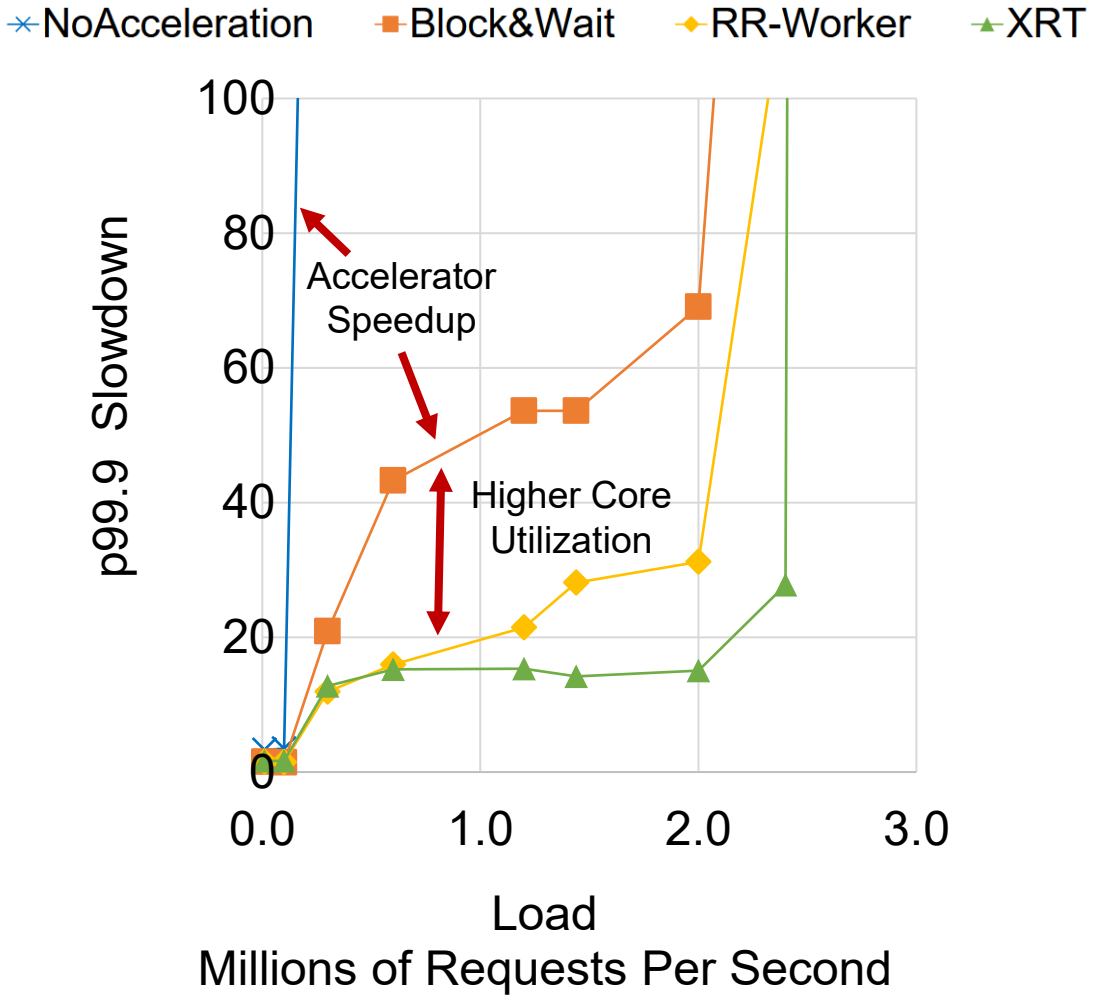


Deserialize-
Decompress-Hash

Case 1: Highly Accelerable

$$slowdown = \frac{Response\ Time}{Unloaded\ Accelerated\ Execution\ Time}$$

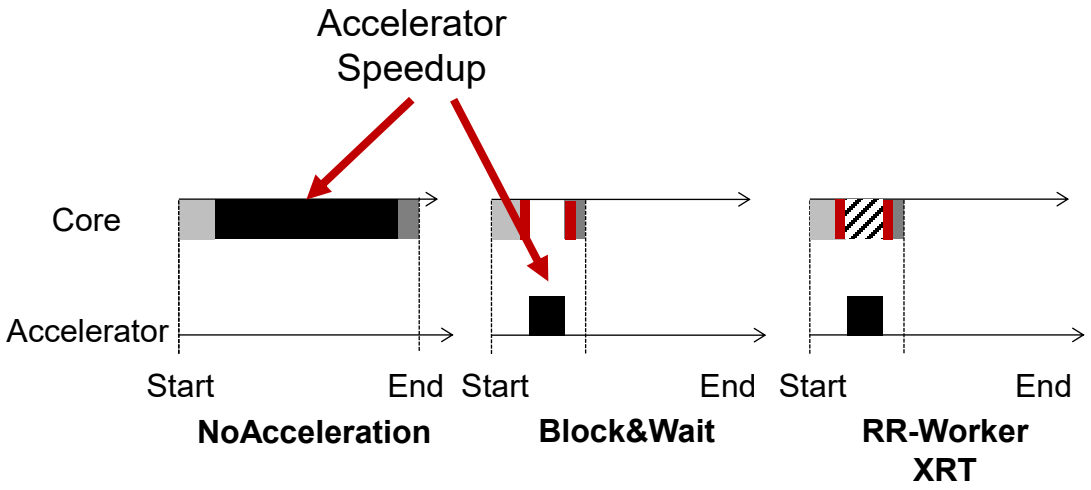


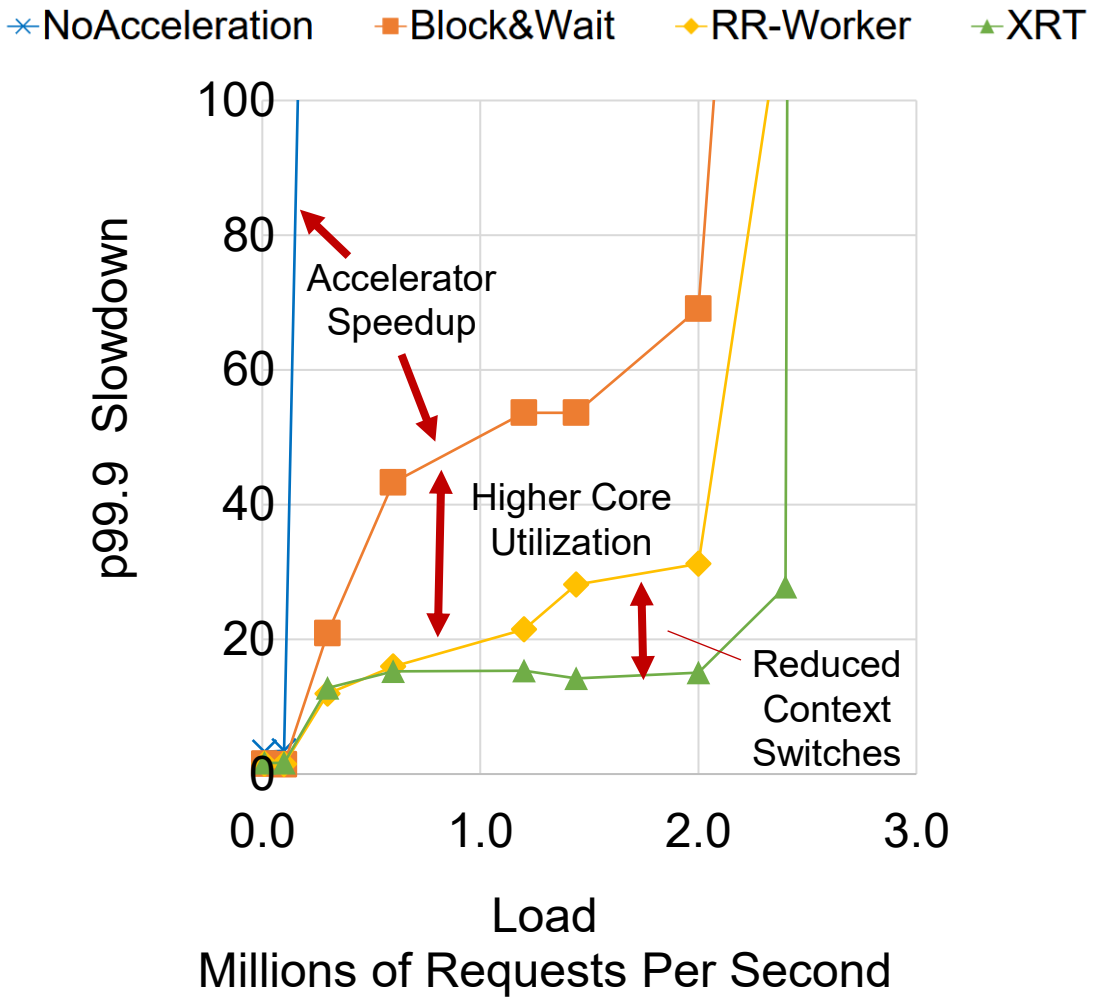


Deserialize-
Decompress-Hash

Case 1: Highly Accelerable

$$slowdown = \frac{Response\ Time}{Unloaded\ Accelerated\ Execution\ Time}$$

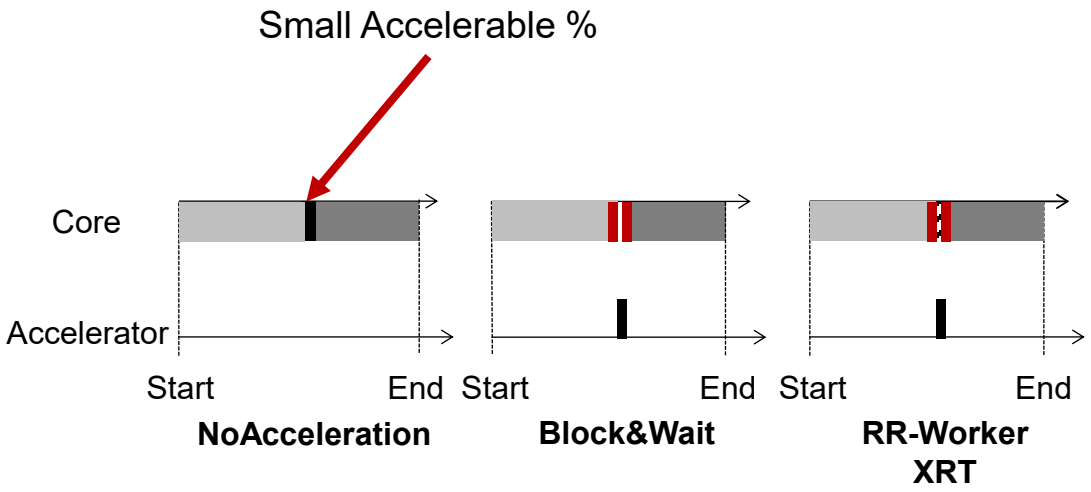


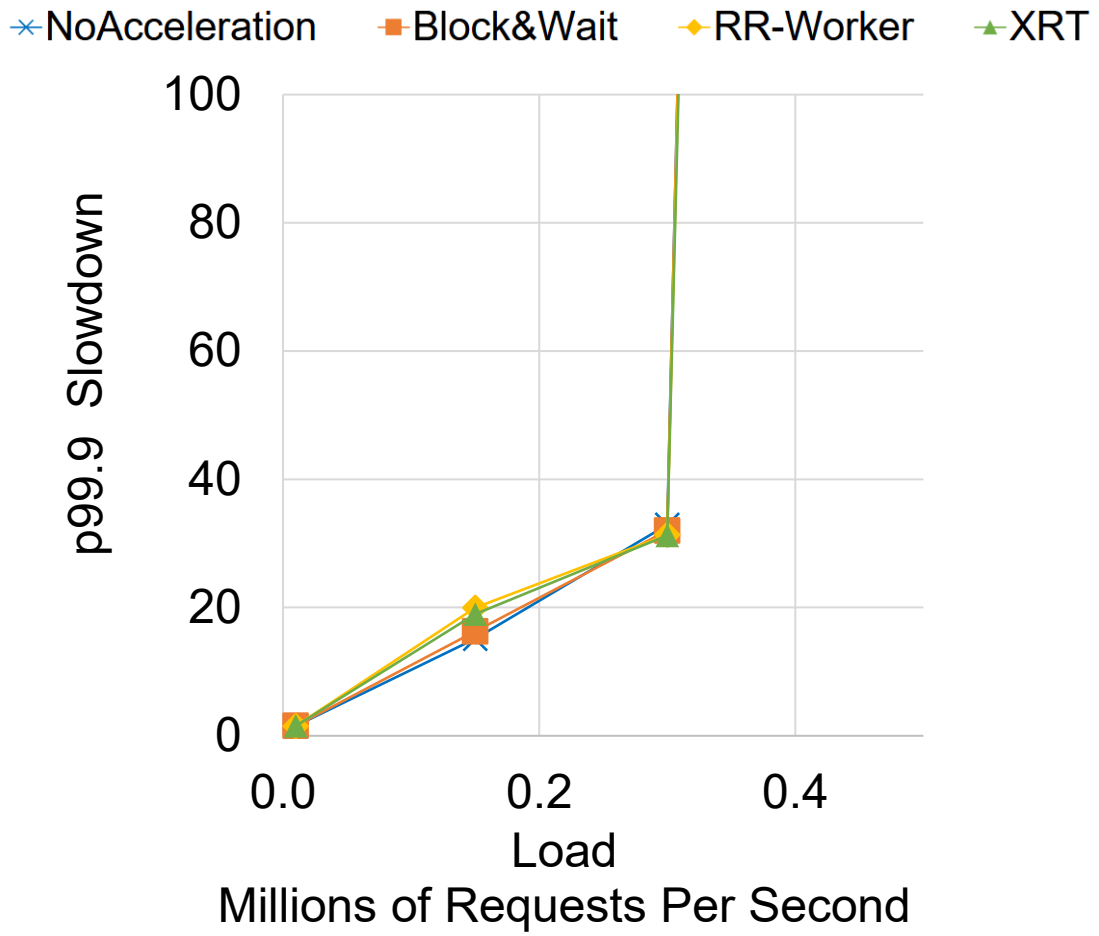


Deserialize-
Decompress-Hash

Case 2: Not Accelerable

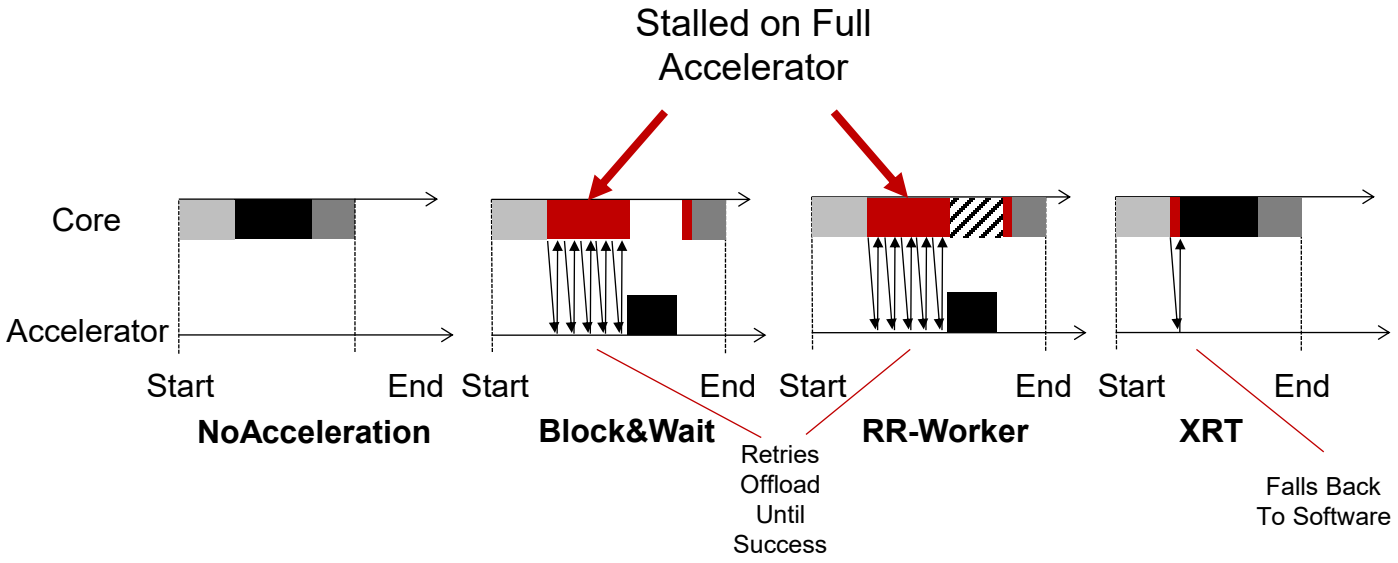
Dominated by general-purpose core processing

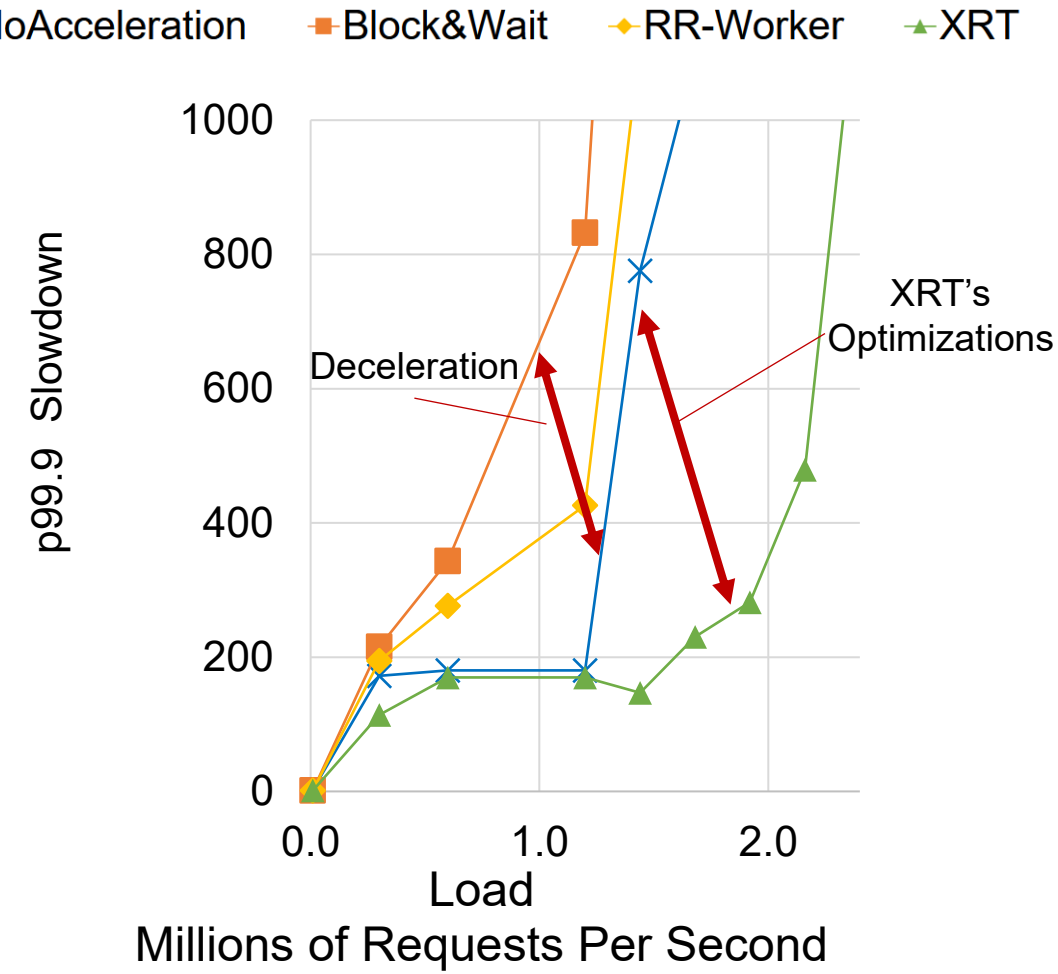




Matmul-Memfill-PrincipalComponentAnalysis

Case 3: Stalled On Accelerator





Memcpy-PointerChase

Contributions and Conclusion

- XRT: An accelerator-aware runtime for XMPs
  - Scales to large XMPs by distributing accelerator notifications
  - Falls back to software when accelerators are overloaded
  - Notification-aware scheduler eliminates unnecessary context switches
  - Achieves up to 3.2x increase in throughput under service-level constraints and never experiences slowdowns

For more information visit Alian Research Group at
https://arg.csl.cornell.edu

CornellEngineering
Electrical and Computer Engineering

CSL