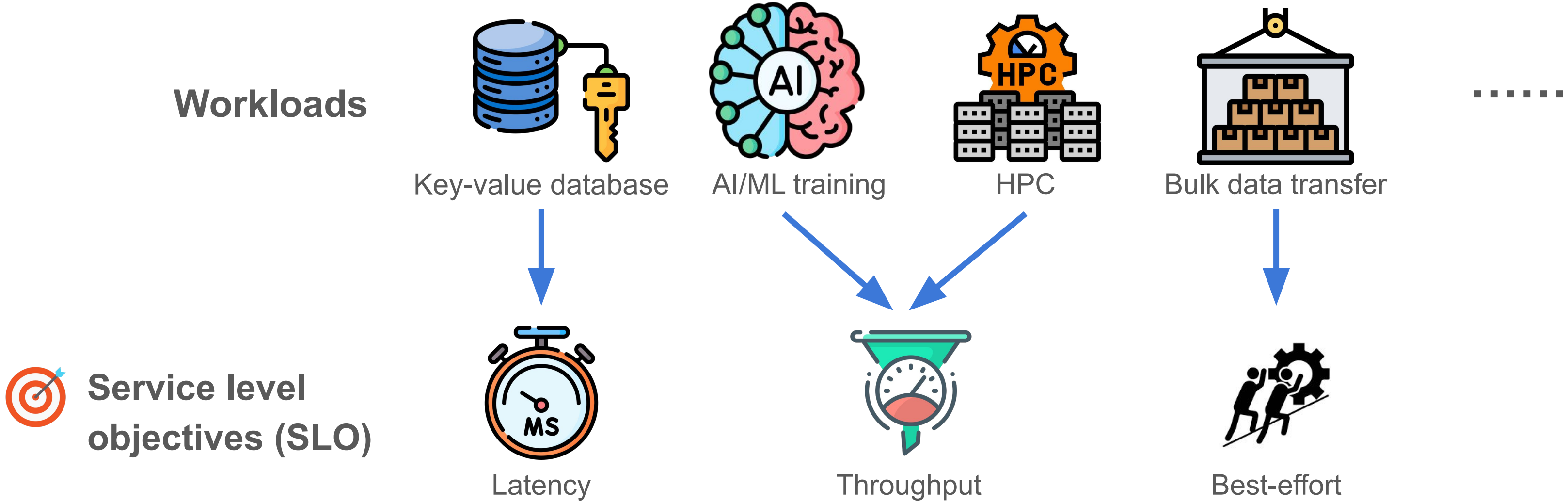
Exposing RNIC Resource for Software-Defined RDMA Scheduling

Yibo Huang, Yiming Qiu, Yunming Xiao, Archit Bhatnagar
Sylvia Ratnasamy, Ang Chen

RDMA for diverse workloads is critical in data centers

- Remote Direct Memory Access (RDMA):
    - low latency, high bandwidth, minimal overhead
    - widely deployed in data centers



Workloads

Key-value database

AI/ML training

HPC

Bulk data transfer

......

Service level objectives (SLO)

Latency

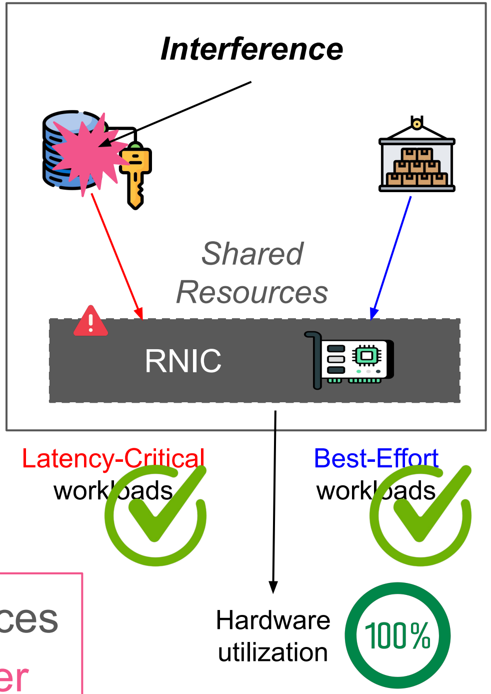
Throughput

Best-effort

Trend: co-located worklaods sharing RDMA hardware resources

- Ideal world:
  - Offer strong SLO guarantees
  - Operate hardware at 100% utlization
- Reality: colocation leads to “noisy neighbors” issues

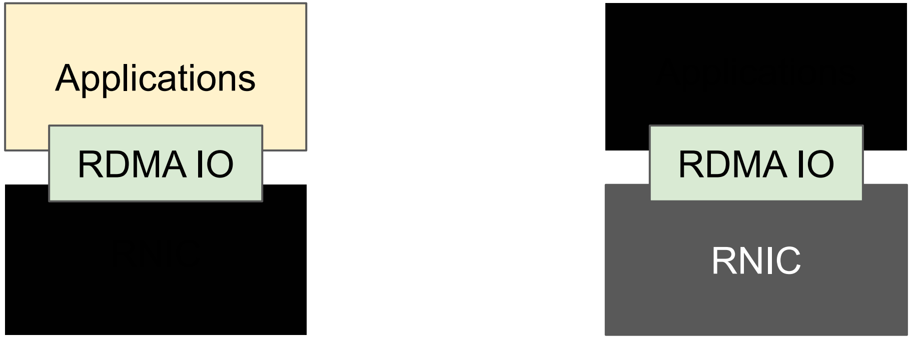
- Workloads on the same RDMA-enabled end host contend for shared RDMA NIC (RNIC) resources, e.g.,
  - RNIC port bandwidth
  - RNIC cache
  - ……



Requirement: workloads must share resources in a context-aware and cooperative manner

Sate-of-the-art RDMA frameworks

- Lack *principled* design to reason about the *interaction* between RDMA software and hardware layers



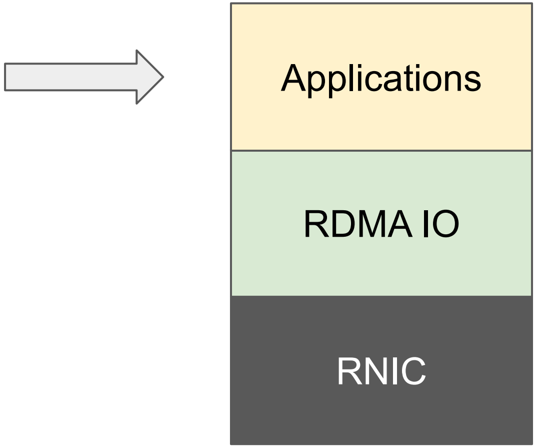
App developer’s view

RDMA hardware’s view

A big *context gap* between software requirements and RDMA hardware behavior

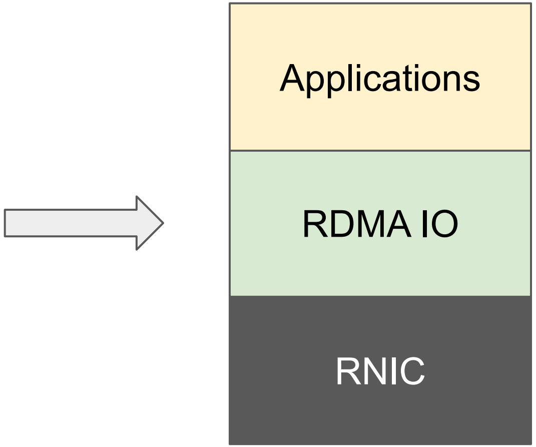
Existing efforts and limitations

- Solution 1: Design applications carefully
  - Lots of efforts
  - Unaware of other colocated applications



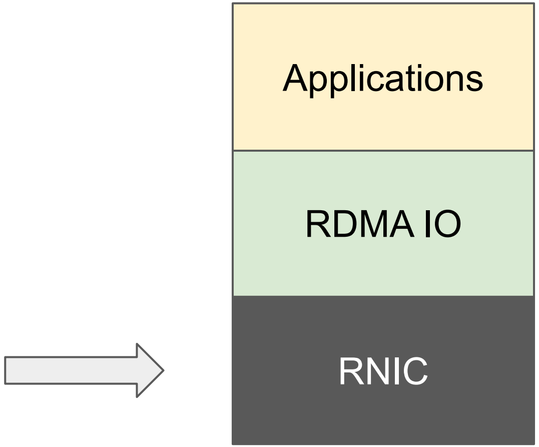
Existing efforts and limitations

- Solution 2: Redesign the RDMA IO
    - Not aware of application context & hardware resources



Existing efforts and limitations

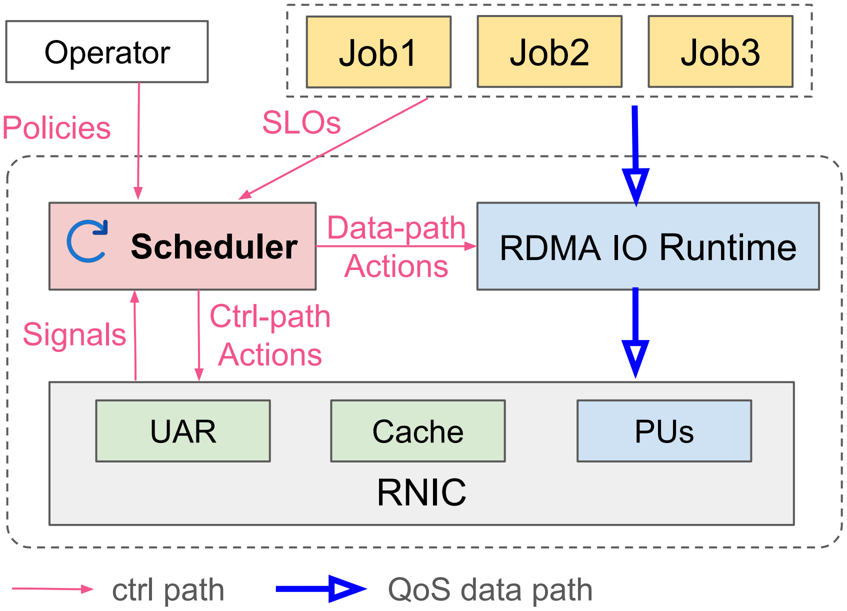
- Solution 3a: Redesign RNIC driver implementation
  - Require changes to existing RDMA practices (compatibility issues)
- Solution 3b: Leverage hardware programmability
  - Programmable hardware are not always available (availability issues)



Point solutions for specific cases

/ extensive changes to RDMA pipeline assumptions

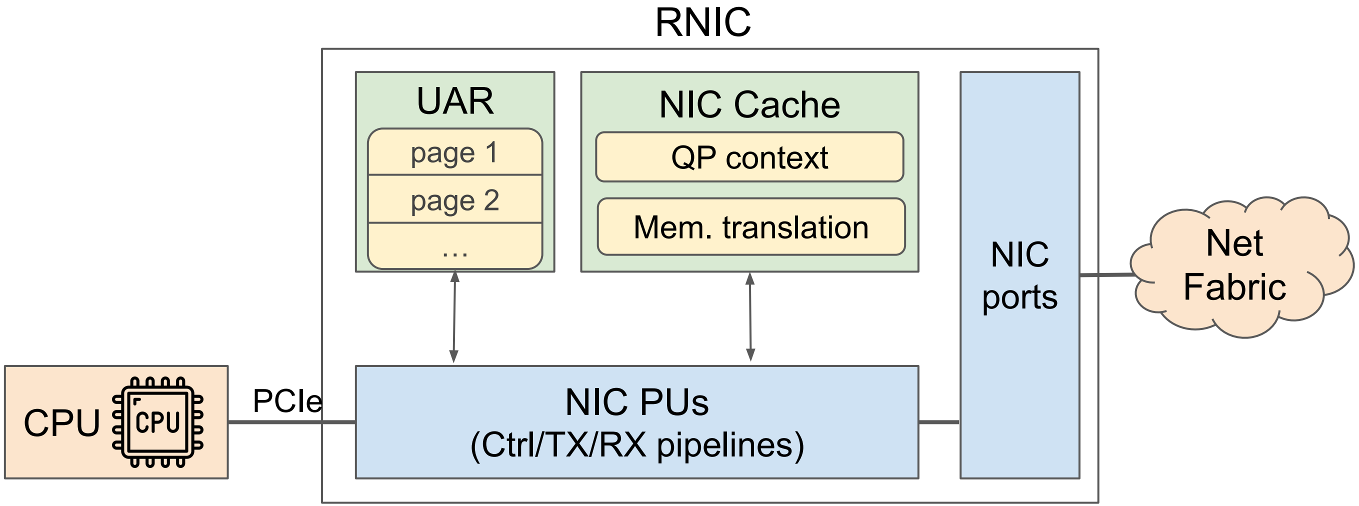
Solution: SwiftRDMA



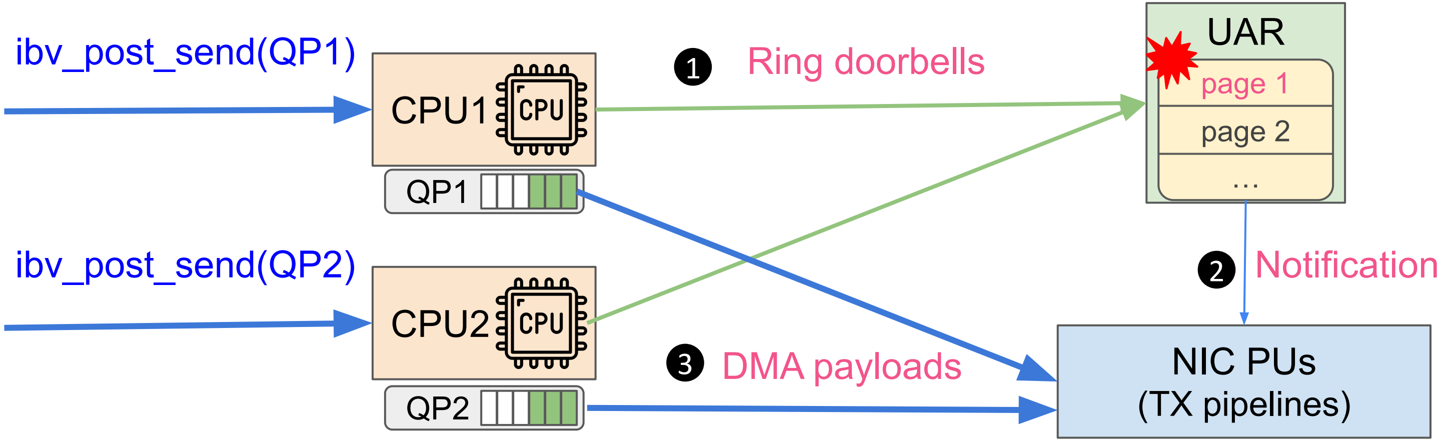
Domain-Specific Challenges

- Identifying root causes of interferences
  - Understand intrinsics of microarchitecture resources
  - Analyze different types of contentions/SLO violations
- Finding signals/actions as control knobs
  - Assume commodity RNICs, no way to customize
  - Unlike CPU core scheduling, no way to directly control
- Enforcing scheduling policies efficiently
  - Gather signals and take actions with low overhead
  - Scale well as the amount of workloads grows

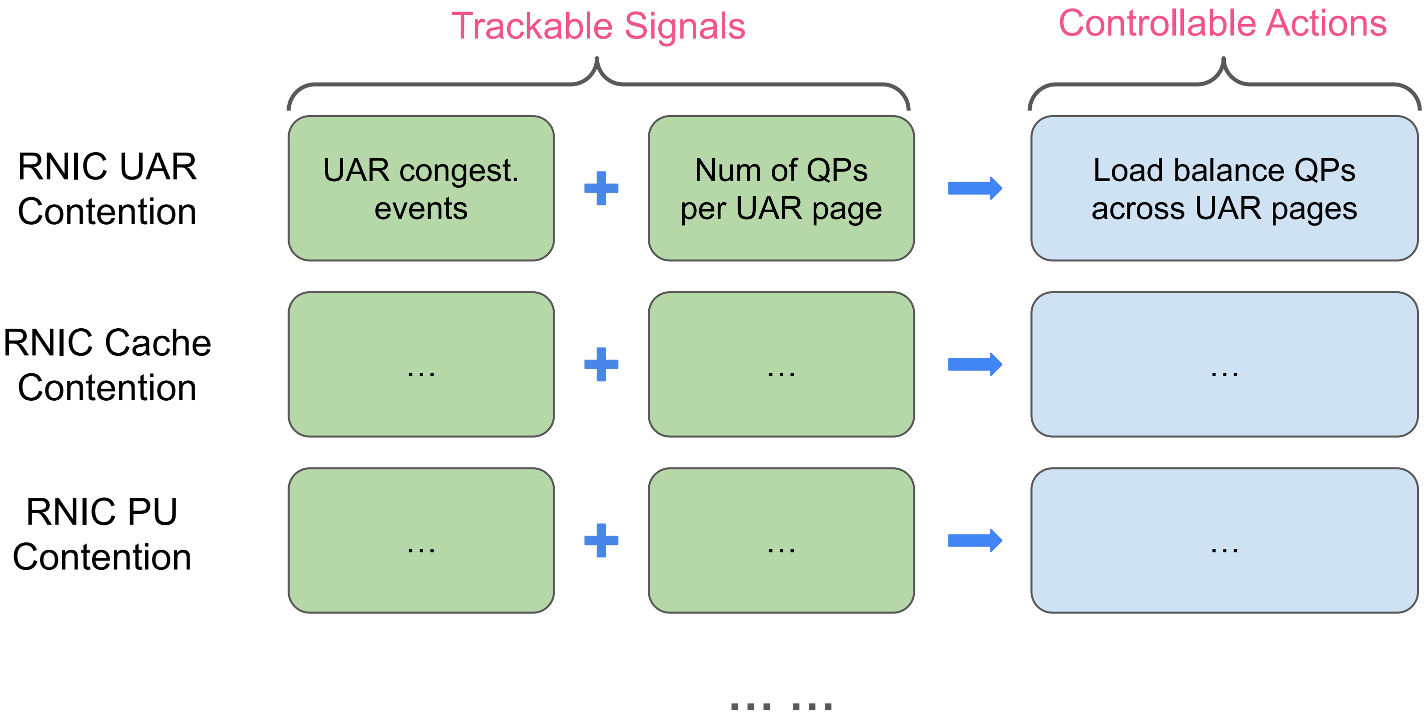
Identifying Contention Root Causes



Case study: RNIC UAR Contention



Leveraging Signals and Actions



	Trackable Signals				Controllable Actions
RNIC UAR Contention	UAR congest. events	+	Num of QPs per UAR page	→	Load balance QPs across UAR pages
RNIC Cache Contention	…	+	…	→	…
RNIC PU Contention	…	+	…	→	…

… …

Summary and Next Steps

- **Motivation**: RDMA is essential for co-located data center workloads, but sharing of RNIC resource could lead to contention and SLO violation.
- **SwiftRDMA**: exposing hardware resources for software-defined scheduling.
- **Key insight**: Bridging the context gap between RDMA hardware resource and application-level requitements.

Near-term goals

- Gathering signals/taking actions with low overhead.
- Enforcing policies with high efficiency and accuracy.

yiboh@umich.edu