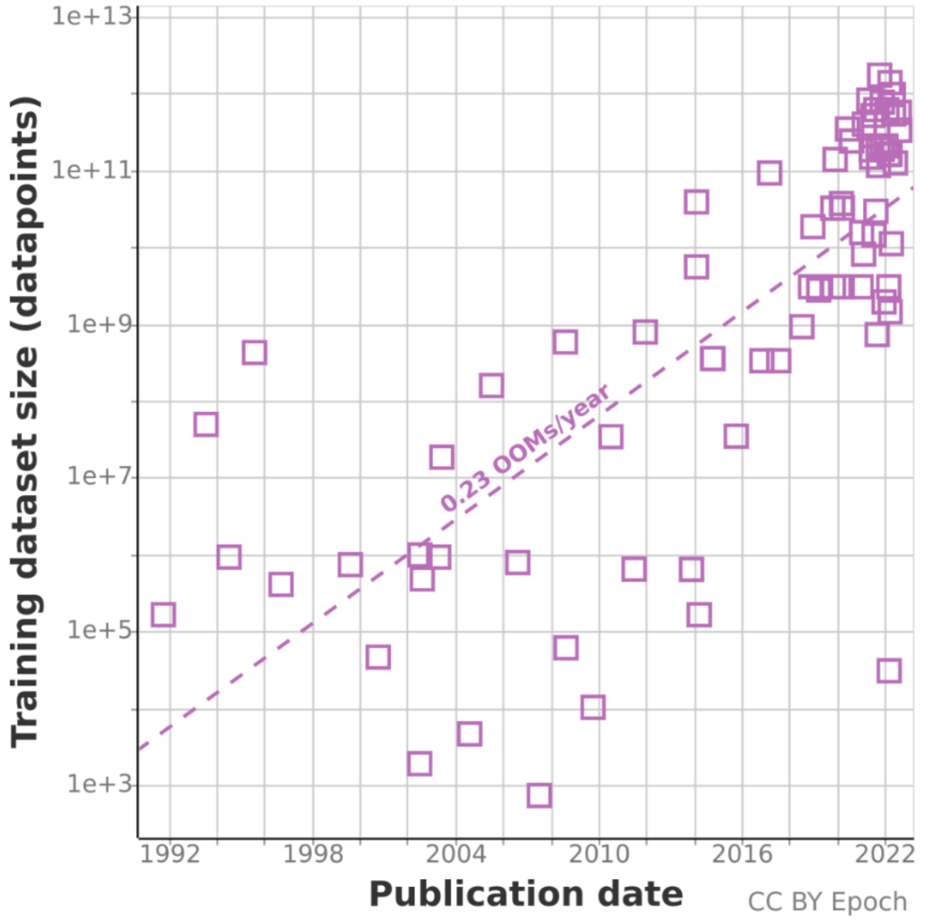
Supporting Secure Multi-GPU Computing with Dynamic and Batched Metadata Management

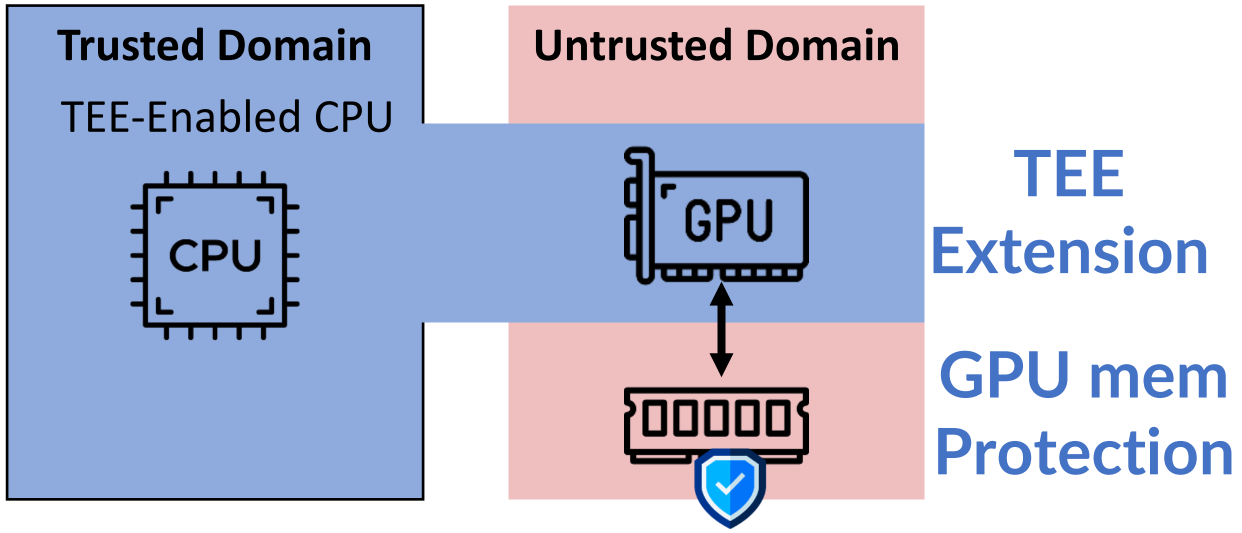
Seonjin Na[1], Jungwoo Kim[2], Sunho Lee[2], Jaehyuk Huh[2]

[1]Georgia Institute of Technology, [2]KAIST

Importance of Multi-GPU Computing



Secure GPU Computing Efforts in Academia and Industry

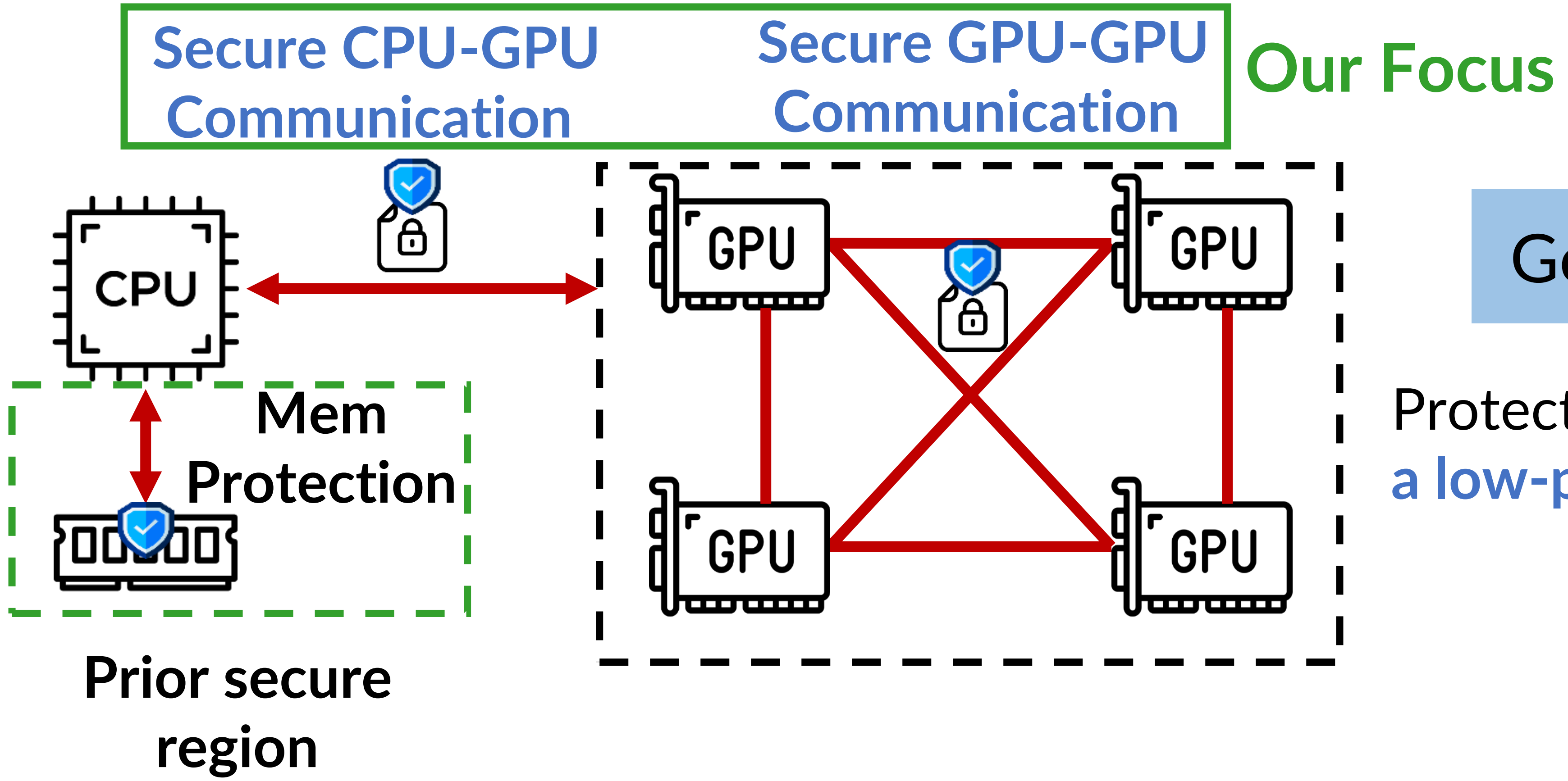


Graviton [OSDI ’18]
HIX [ASPLOS ’19]

Common Counters [HPCA’ 21]
SHM [HPCA ’22]
Plutus [HPCA ’23]

Lack of **data protection mechanism optimized for multi-GPU systems**

Our Goal: Efficient Data Protection for Multi-GPU System



Goal of this work

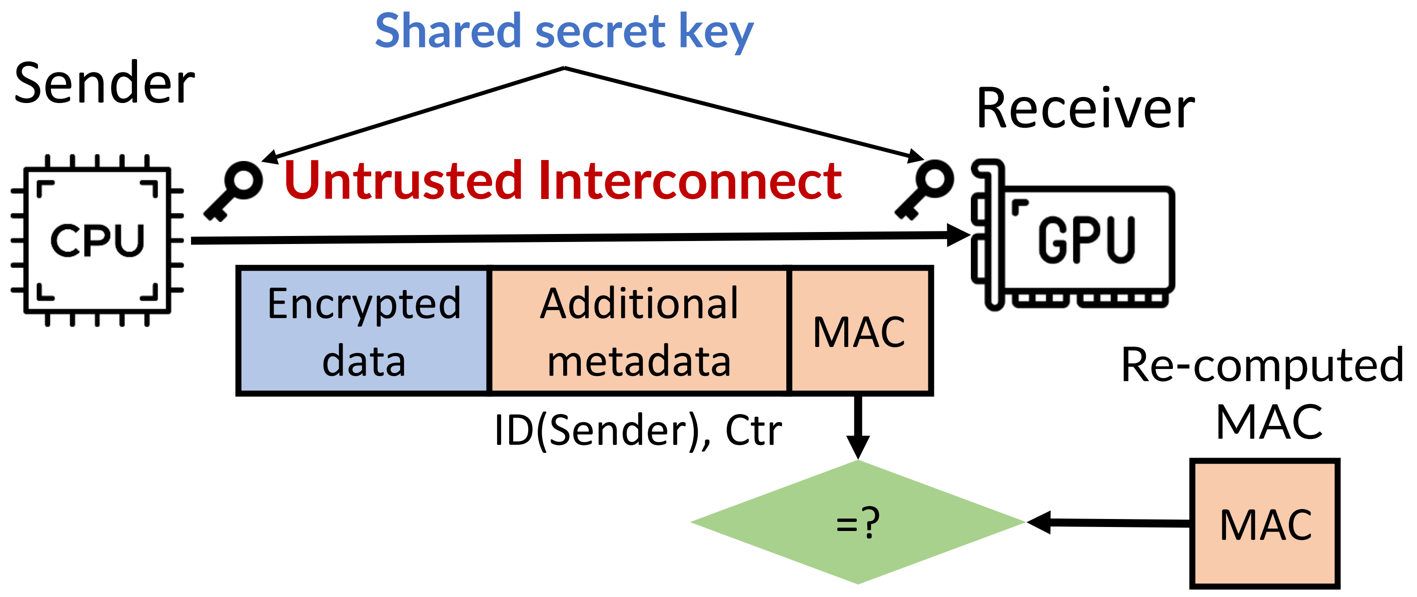
Protect transferred data with a low-performance overhead

Contents

Background: Protecting Transferred Data through Interconnect

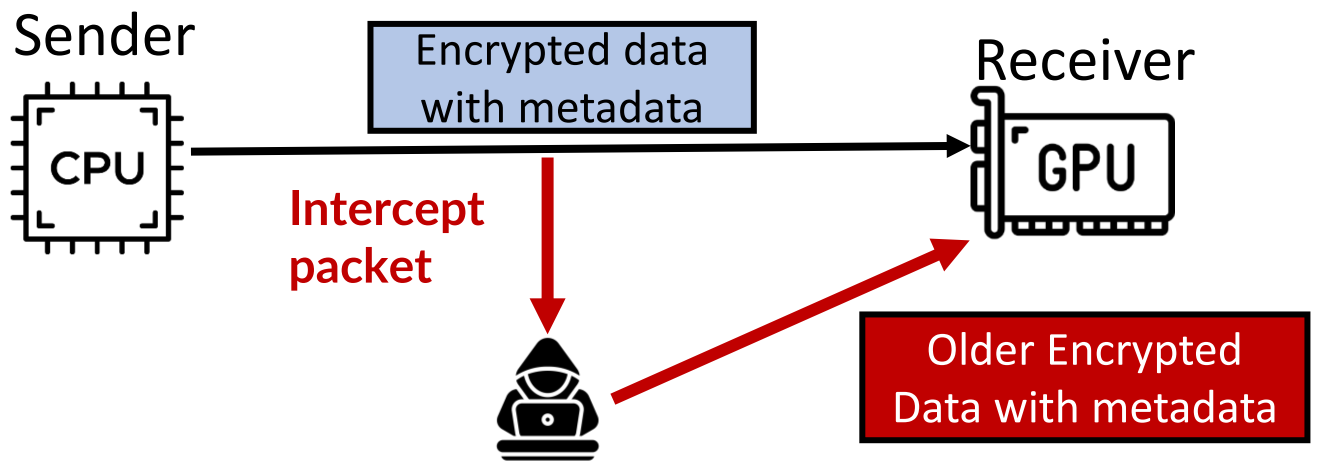
Confidentiality & Integrity

Authenticated en/decryption



Freshness

Replay attack protection

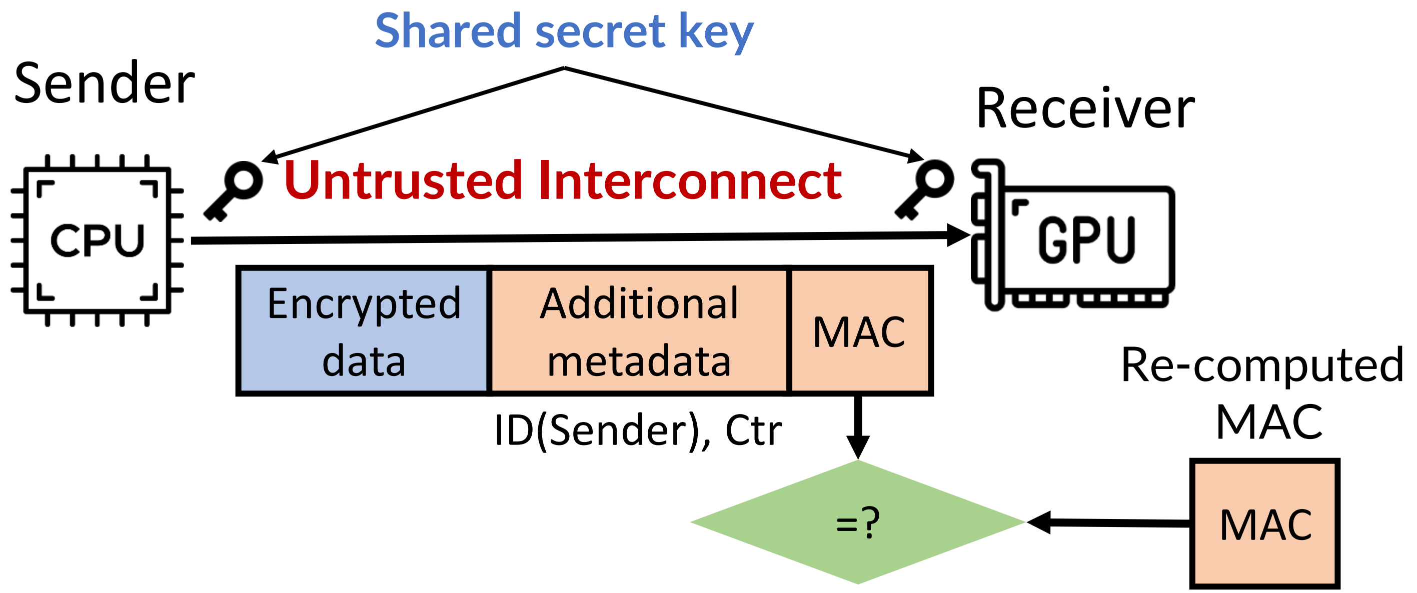


Sender
CPU
Encrypted data with metadata
Receiver
GPU
Intercept packet
Older Encrypted Data with metadata

Background: Protecting Transferred Data through Interconnect

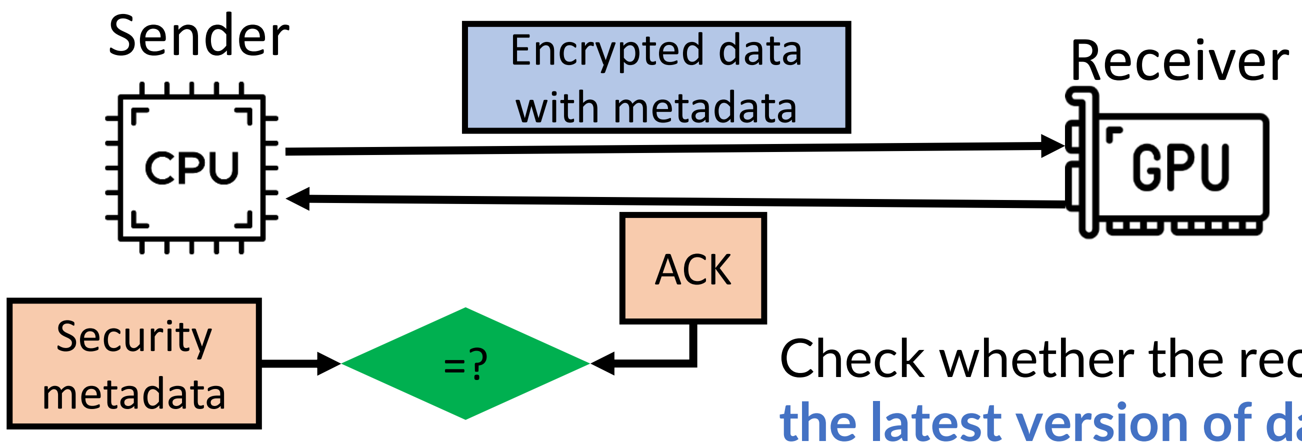
Confidentiality & Integrity

Authenticated en/decryption



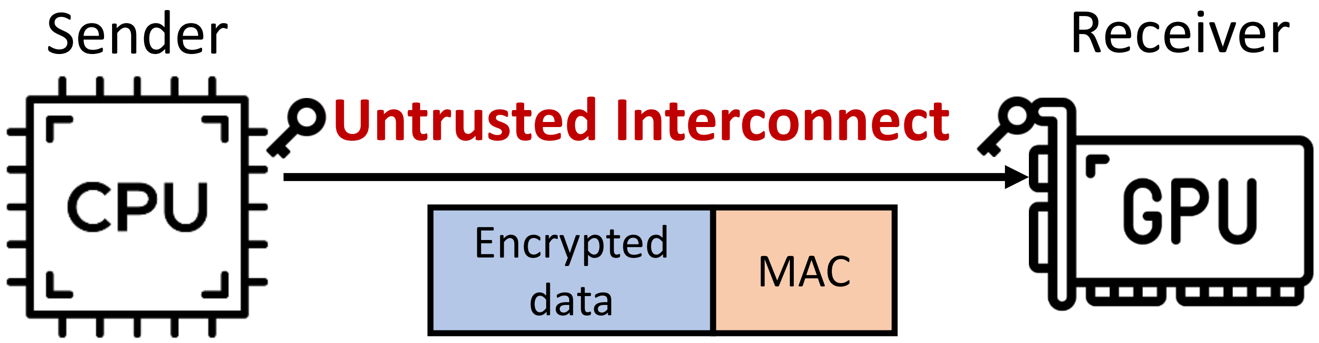
Freshness

Replay attack protection

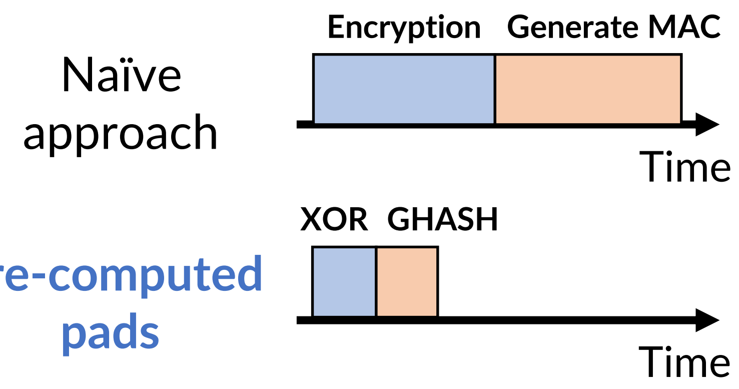


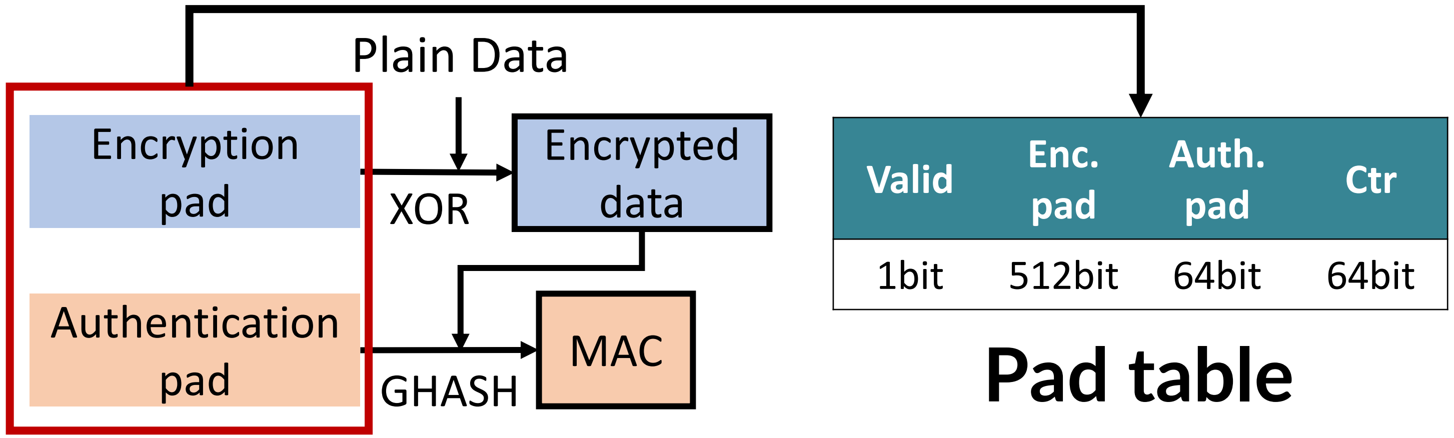
Check whether the receiver gets **the latest version of data**

Authenticated En/Decryption with Pre-Computation [1]



Store pre-generated Enc.pad/ Auth. pad





Valid	Enc. pad	Auth. pad	Ctr
1bit	512bit	64bit	64bit

Pad table

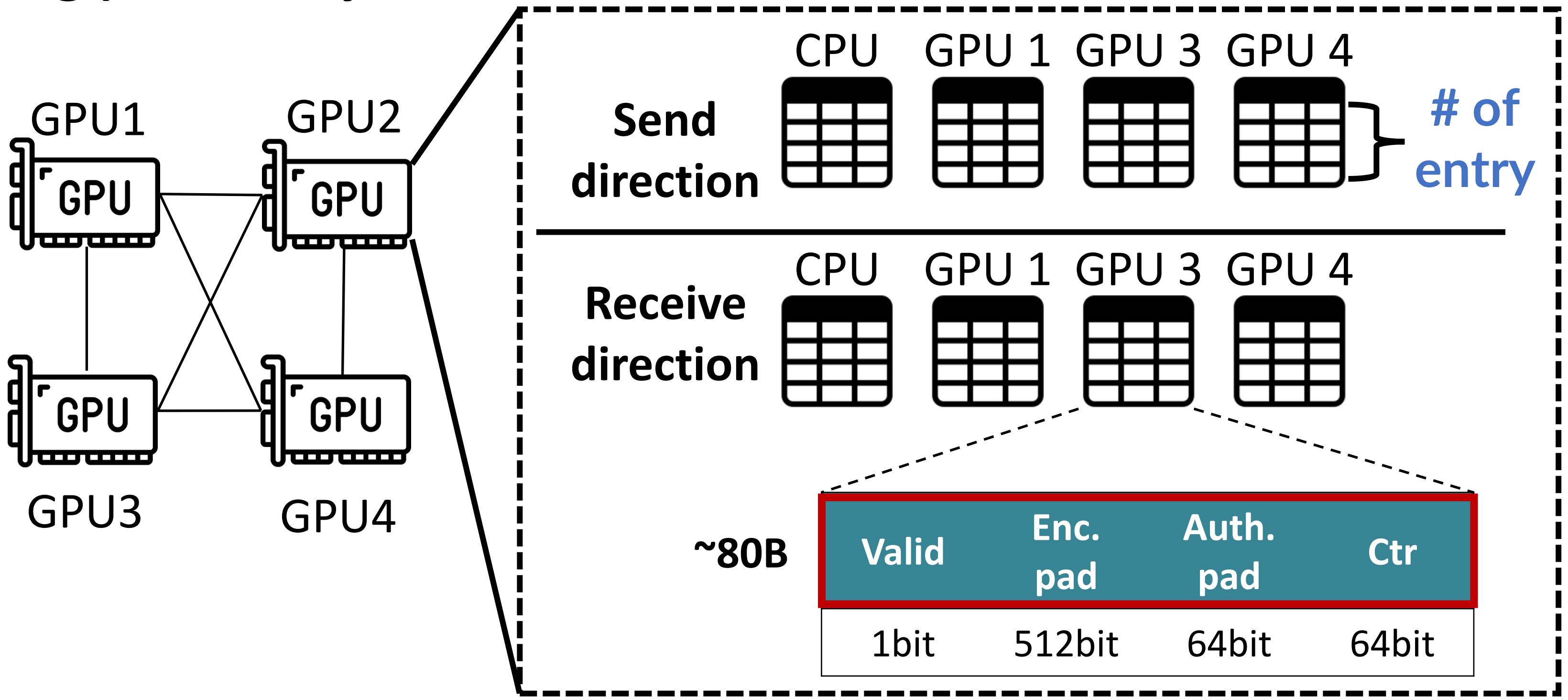
Stored in on-chip area

[1] Efficient data protection for distributed shared memory multiprocessors, PACT'06

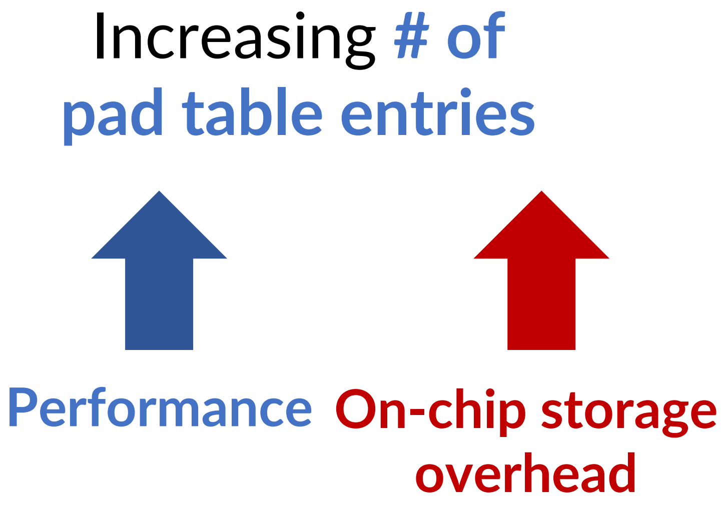
Prior Pad Table Management (Private) [1]

- Maintains **same # of pad entries** for all commu. pairs in a system

E.g.) 4-GPU System



Valid	Enc. pad	Auth. pad	Ctr
1bit	512bit	64bit	64bit

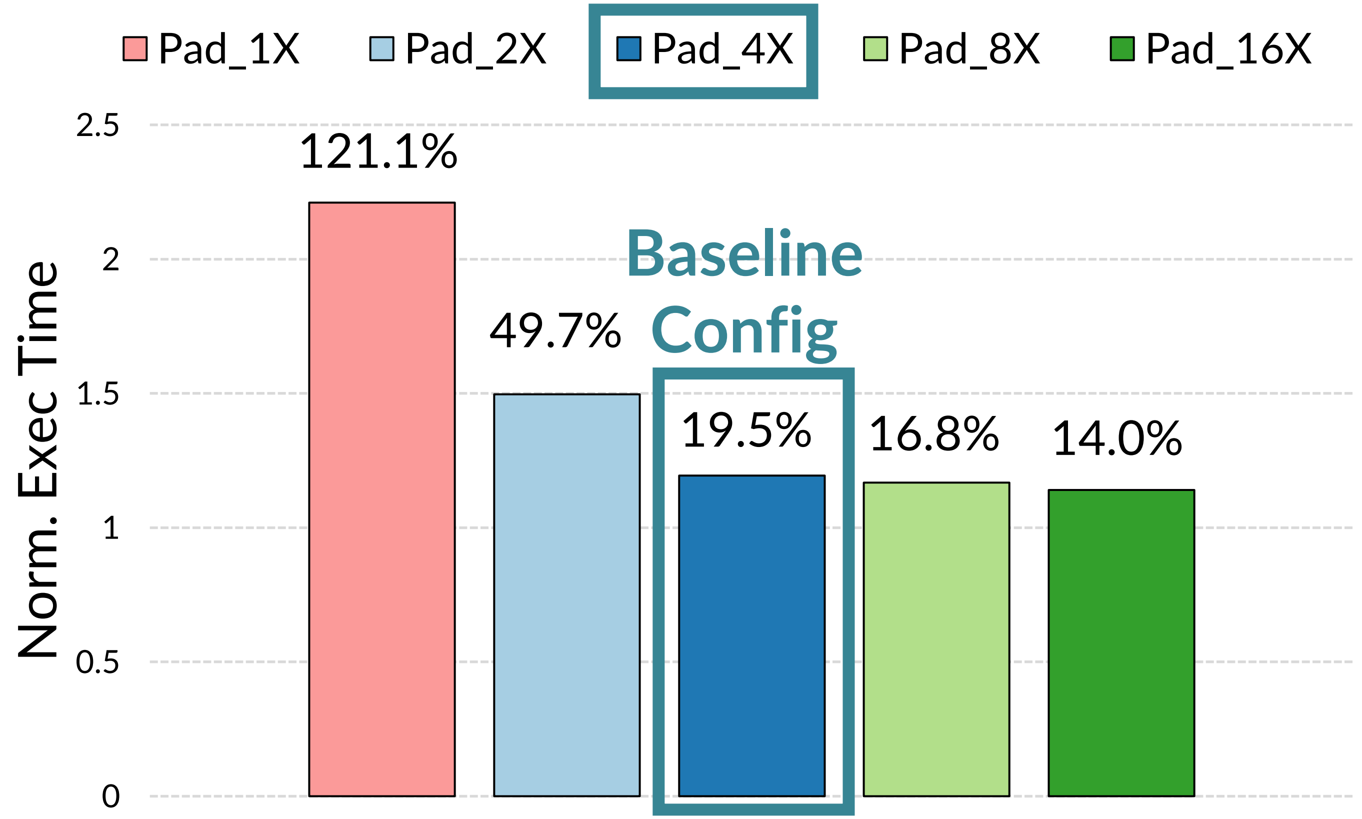


[1] Efficient data protection for distributed shared memory multiprocessors, PACT'06

Performance Impact of # of Pad Table Entries (Private) [1]

Baseline:
Unsecure 4 GPU

Use 4 pad table entries for all commu. pairs

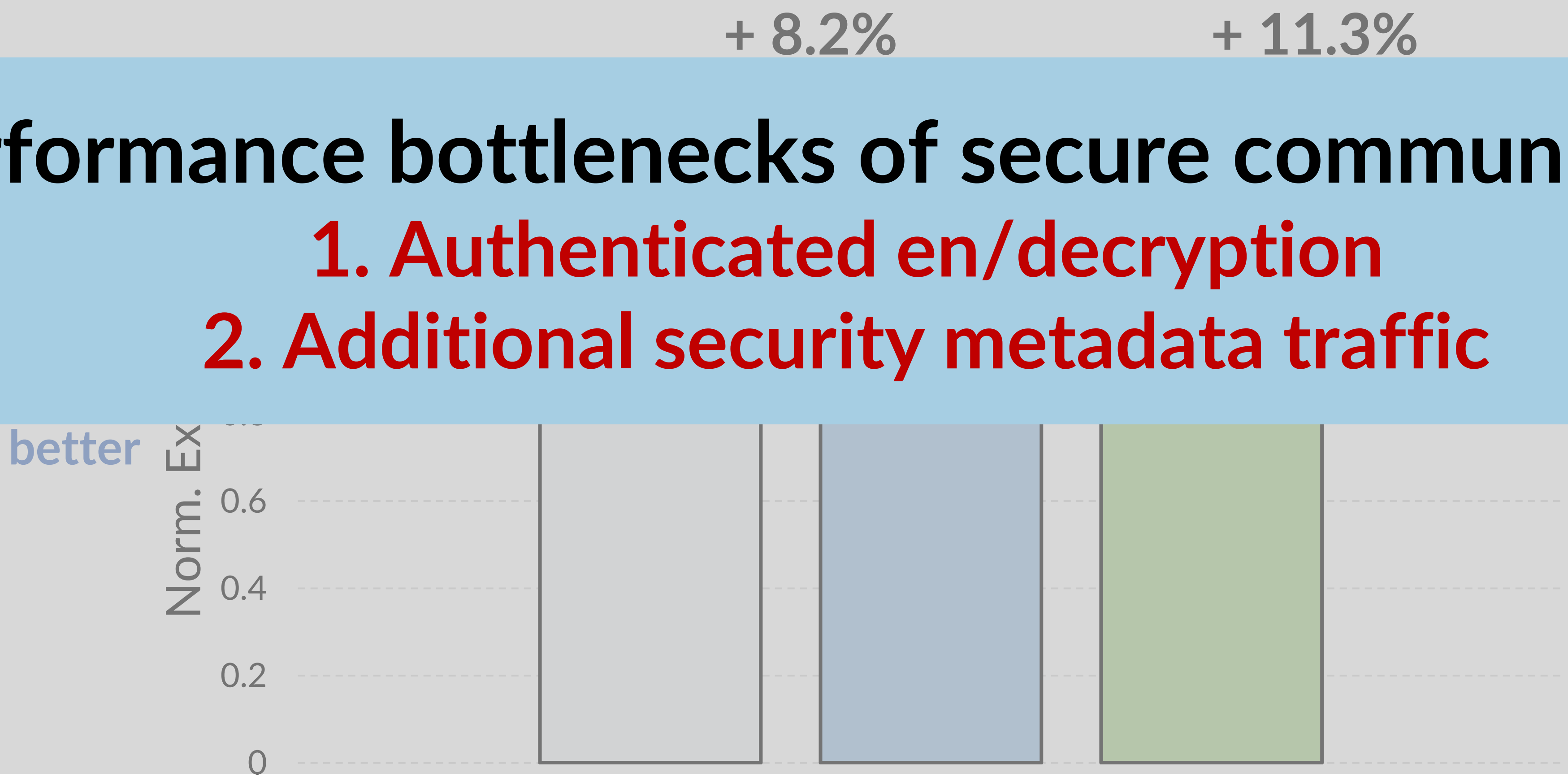


Lower is better

[1] Efficient data protection for distributed shared memory multiprocessors, PACT'06

Performance Breakdown Analysis

- Secure multi-GPU incurs average 19.5% performance degradation
  - Auth. en/decryption: 8.2% slowdown, Metadata traffic: 11.3% slowdown



Performance bottlenecks of secure communication

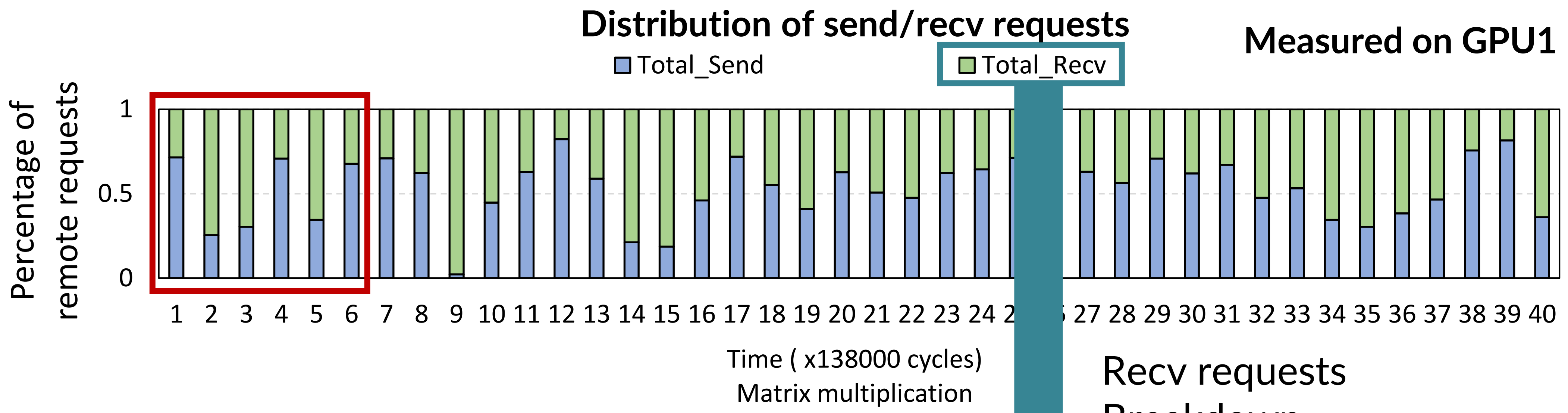
1. Authenticated en/decryption

2. Additional security metadata traffic

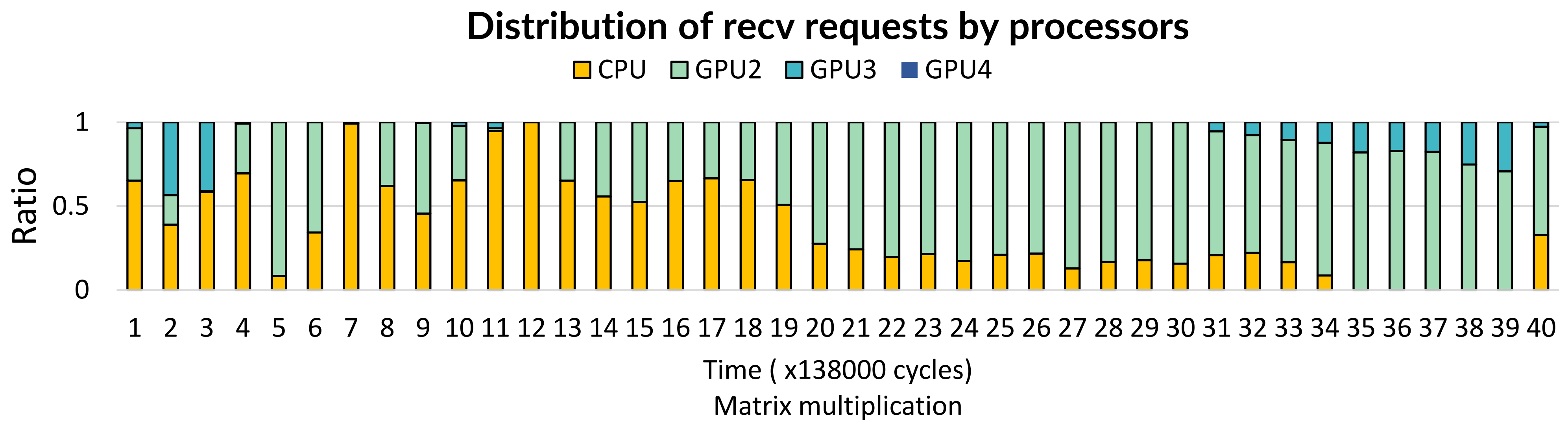
Contents

Key Insight 1: Dynamic Behavior of Communication Patterns

Measured on GPU1



Recv requests Breakdown



Key Insight 2: Burstiness of Communication in Multi-GPU System

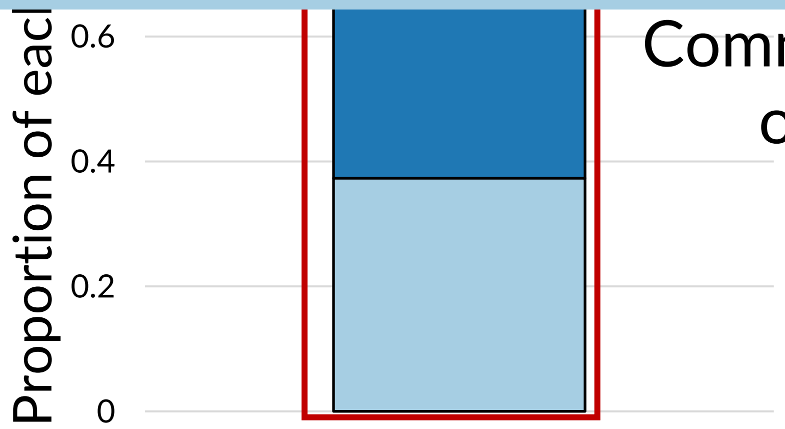
- Analyze distribution of cycles for gathering 16 transmitted data blocks

Cycle distribution

Our Key Observations

1. Dynamic behavior of communication patterns

2. Burstiness of communication



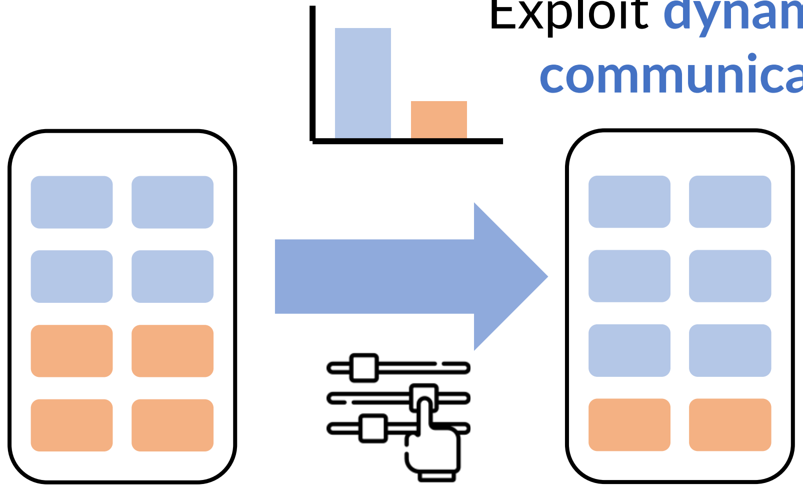
Communication between processors occurs **within a short period**

Main Idea of This Work

Challenge 1: authenticated en/decryption overhead

Dynamic pad table management

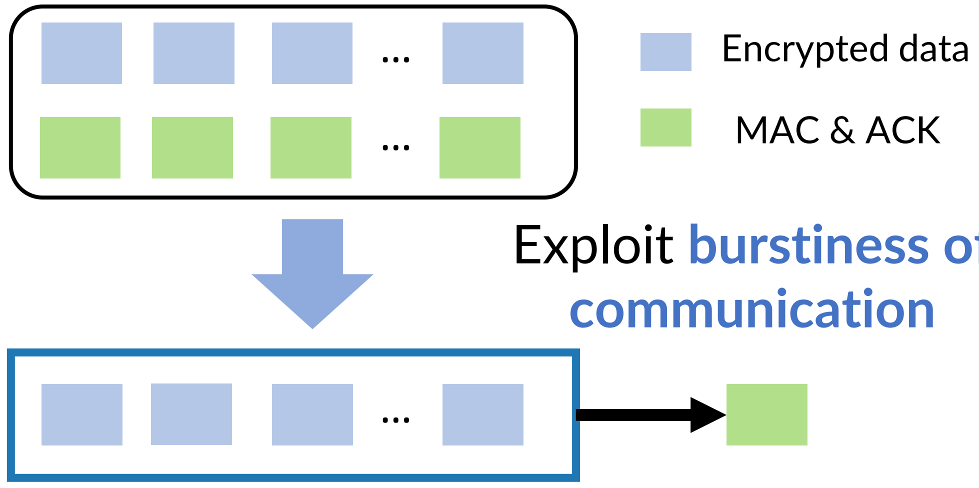
Exploit **dynamic behavior of communication pattern**



Increase opportunity to hide authenticated en/decryption latency

Challenge 2: additional bandwidth by security metadata

Batched MAC generation& verification

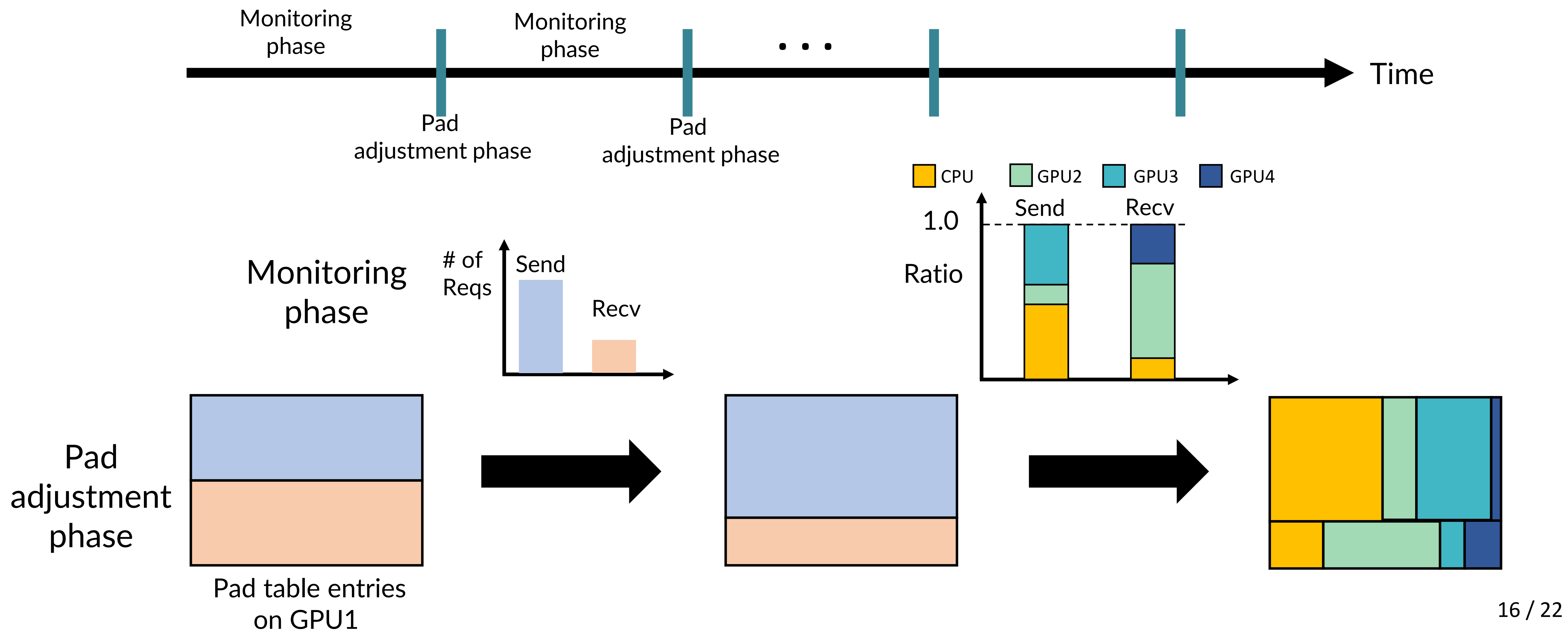


Exploit **burstiness of communication**

Reduce security metadata traffic

Dynamic Pad Table Management

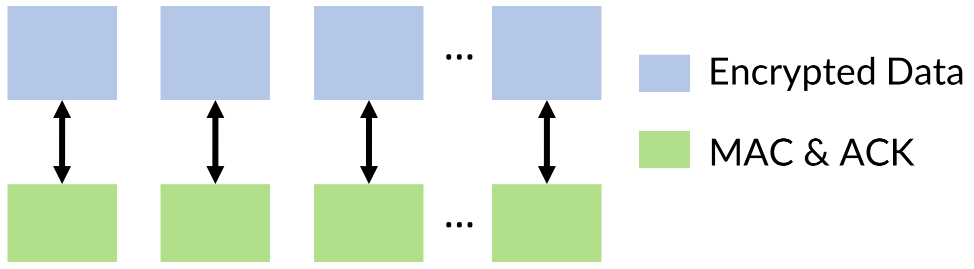
- **Dynamically adjust pad table entries** based on communication pattern



Batched MAC Generation & Verification

- Generate **coarse-grained MAC** to reduce metadata bandwidth

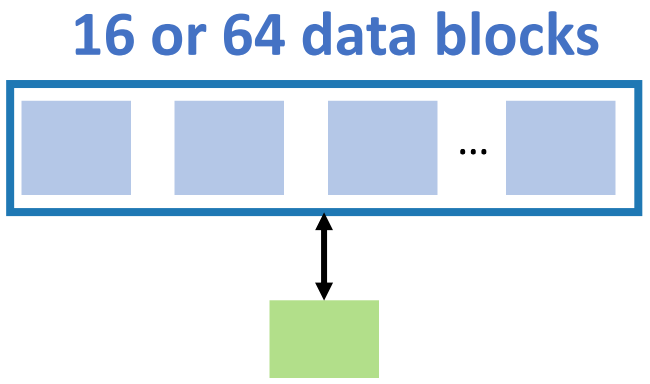
Prior approach



Example: **4KB page migration**

Traffic 64 Data + 64 Metadata (for decryption)
+ 64 MAC & ACK

Our batched MAC



Traffic 64 Data + 64 Metadata (for decryption)
+ 1 Batch info + 1 MAC & ACK

Contents

Evaluation Methodology

- Simulator: MGPUSim [ISCA '19]
- Workloads: 17 apps from various benchmark suites
  - AMD APP SDK, DNN Mark, Hetero Mark, Polybench, SHOC benchmark suites
- System configuration: Models 4 GPU system (AMD R9Nano GPU)

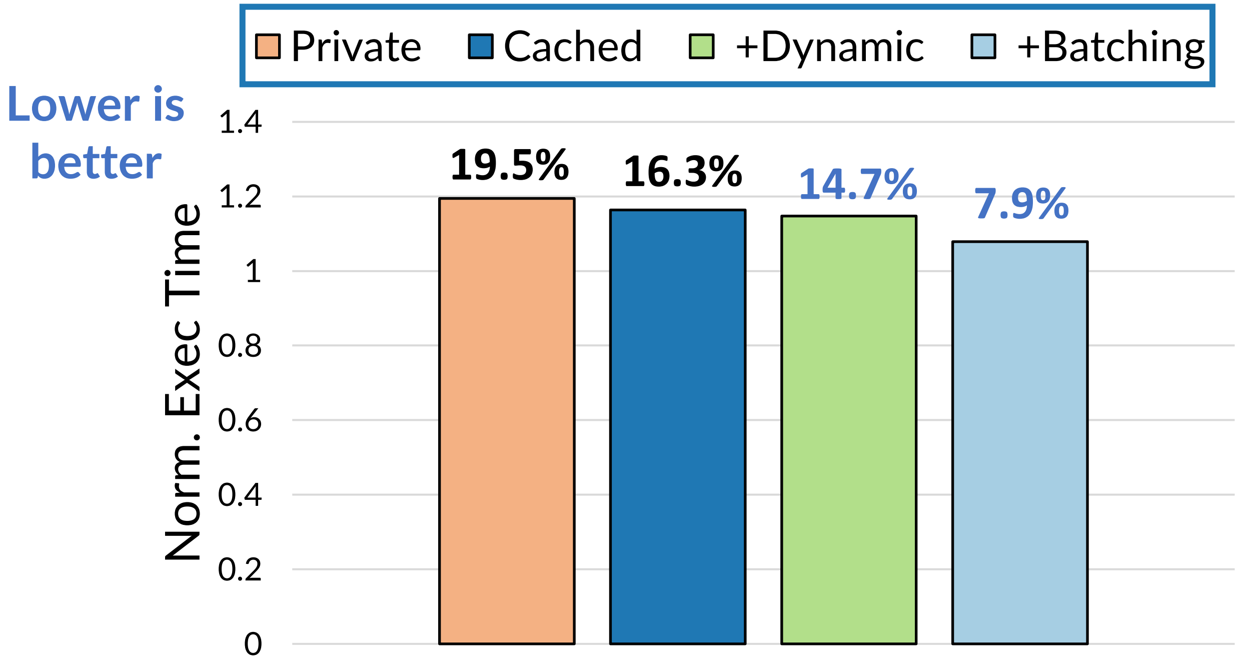
GPU Configuration	
Compute Unit	64 CUs per GPU, 1.0 GHz
L1 Inst / Vector / Scalar Caches Shared L2 Cache	16 KB / 32KB / 16KB 2MB
DRAM	4GB HBM Memory, 512 GB/s
CPU-GPU, GPU-GPU Interconnect	32 GB/s, 50 GB/s
Security Configuration	
Authenticated encryption/decryption	40 cycles [1,2]

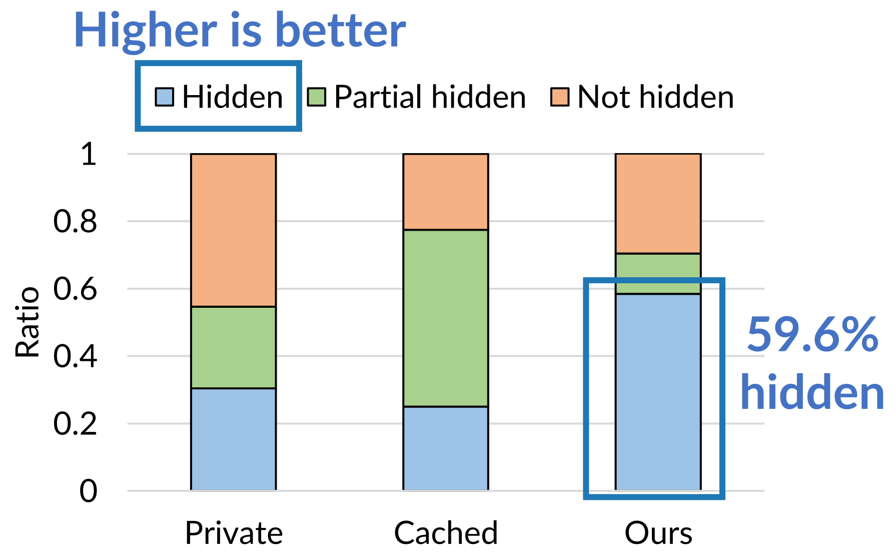
[1] Adaptive Security Support for Heterogenous Memory on GPUs, HPCA '22
[2] Plutus: Bandwidth-Efficient Memory Security for GPUs, HPCA'23

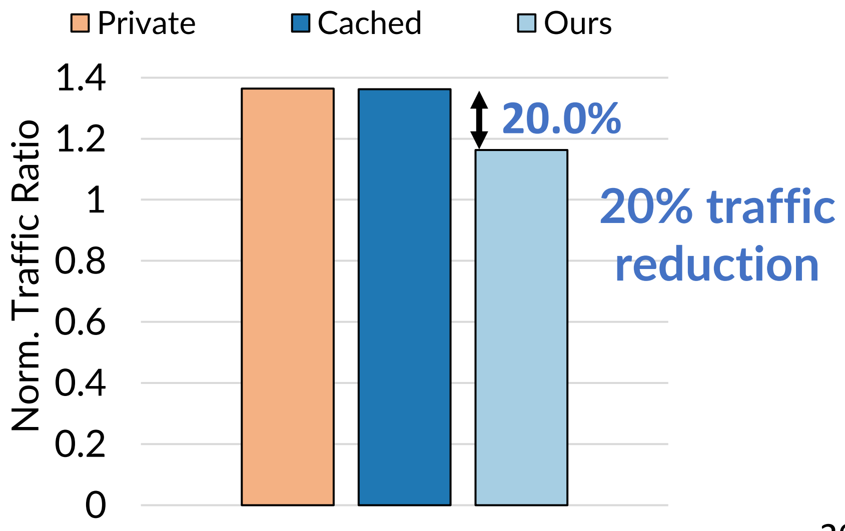
Performance Comparison Result

- Compared with two different mechanisms
  - **Private**[1]: Uses same number of pad entries for all pairs
  - **Cached**[1]: Allocates pad table entries like LRU cache

All configs have the **same total # of pad table entries**







[1] Efficient data protection for distributed shared memory multiprocessors, PACT'06

More Results in the Paper

- Scalability study to the number of GPUs
- Sensitivity study to authenticated encryption/decryption latency
- Hardware overhead of our design

Summary

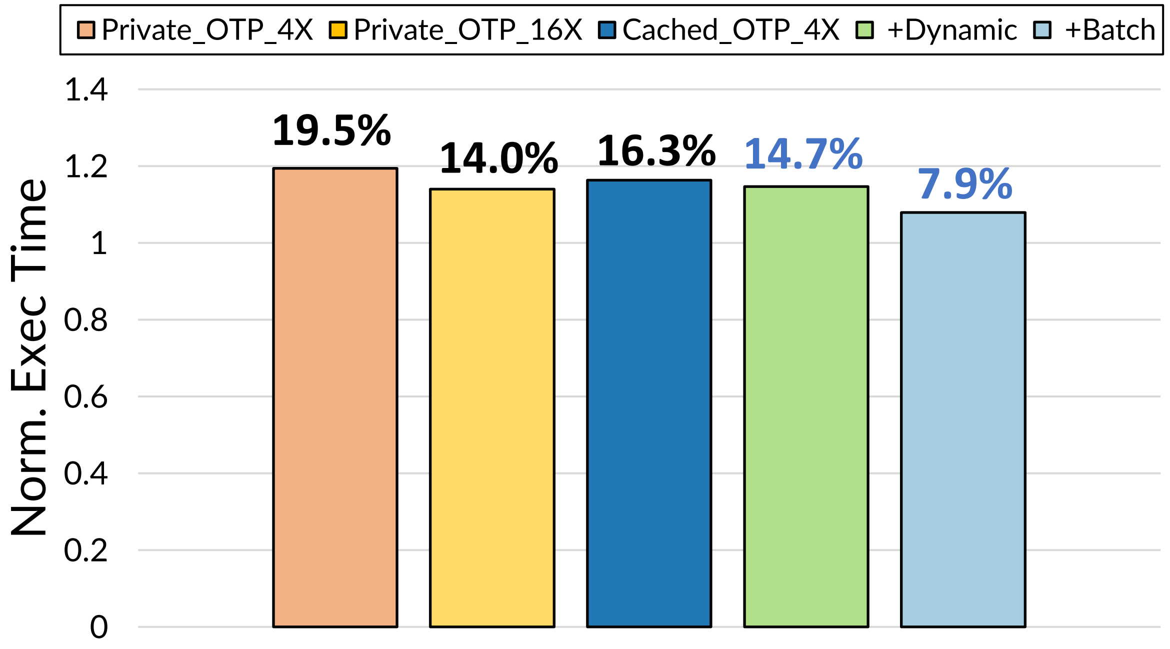
- **Problem**
  - Secure communication degrades multi-GPU system performance
- **Key Idea**
  - **Dynamic pad table management** exploit dynamic communication patterns
  - **Batched MAC generation** leverage burstiness nature of communication
- **Evaluation results**
  - Reduces perf. overhead by **11.6%**, **8.4%** compared to Private, Cached

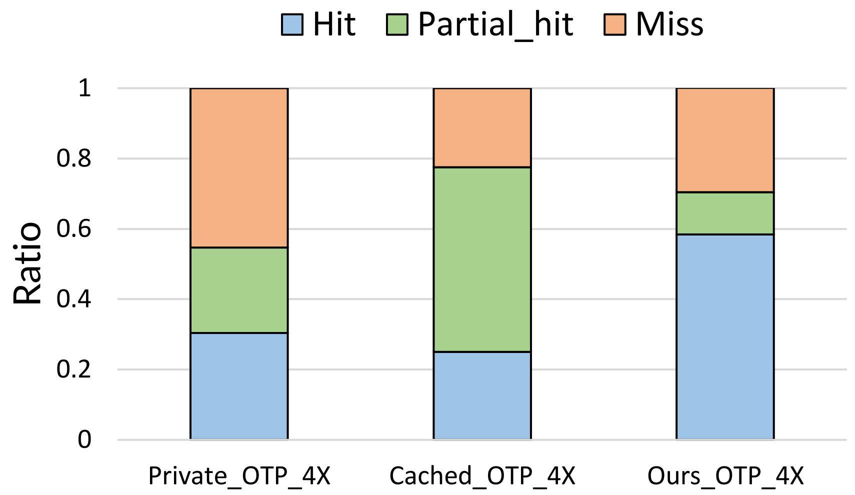
Backup Slides

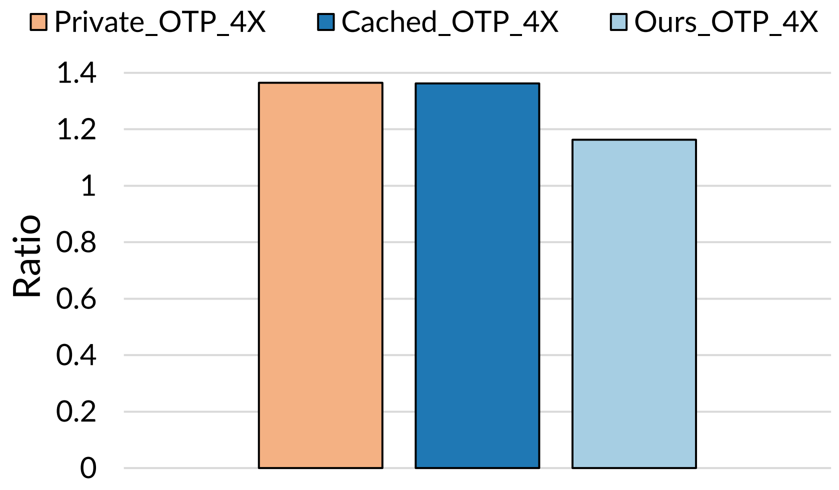
Performance Comparison Result

- Compared with two different mechanisms
  - **Private**: Uses fixed number of pad entries
  - **Cached**: Manages pad table entries like LRU cache

Lower is better


Private_OTP_4X
Private_OTP_16X
Cached_OTP_4X
+Dynamic
+Batch
Norm. Exec Time


Hit
Partial_hit
Miss
Ratio
Private_OTP_4X
Cached_OTP_4X
Ours_OTP_4X


Private_OTP_4X
Cached_OTP_4X
Ours_OTP_4X
Ratio