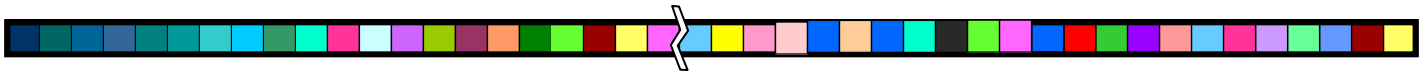
A Framework for Coarse-Grain Optimizations in the On-Chip Memory Hierarchy

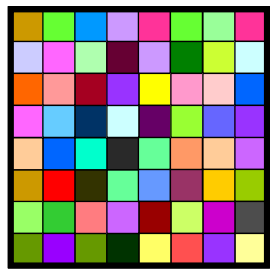
Jason Zebchuk, Elham Safi, and Andreas Moshovos
{**zebchuk**,elham,moshovos}@eecg.toronto.edu

AENAO Research Group
Department of Electrical and Computer Engineering
University of Toronto

Conventional Block Centric Cache

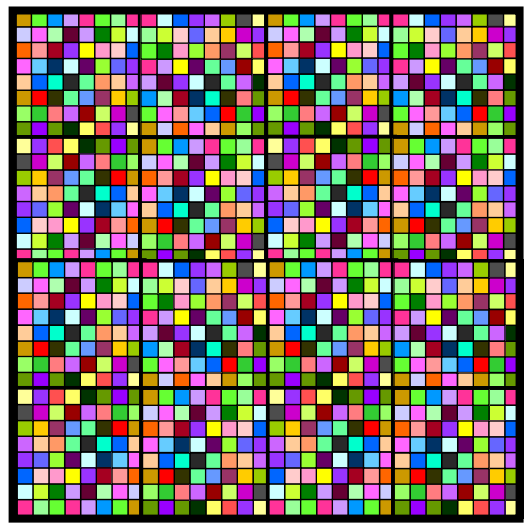
Fine-Grain View of Memory





L2 Cache

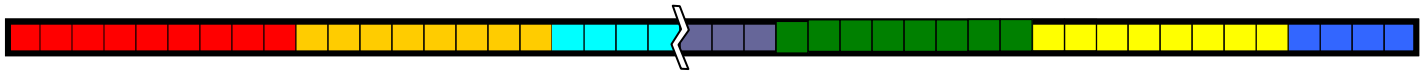
- "Small" Blocks
  - Optimizes Bandwidth and Performance
- Large L2/L3 caches especially

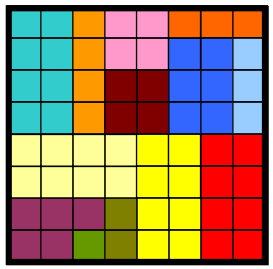


Big Picture Lost

"Big Picture" View

Coarse-Grain View of Memory

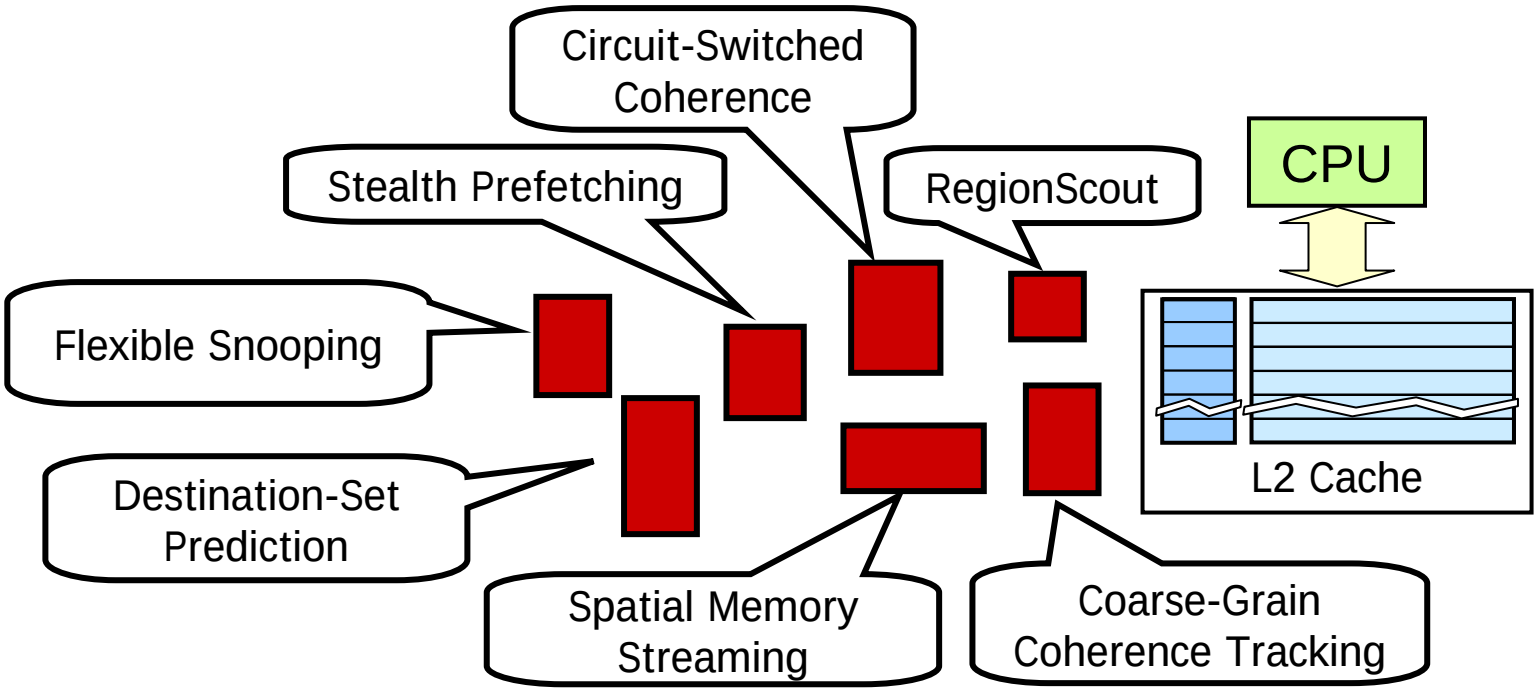




L2 Cache

- **Region**: 2^n sized, aligned area of memory
- **Patterns and behavior exposed**
  - **Spatial locality**
- **Exploit for performance/area/power**

Coarse-Grain Framework

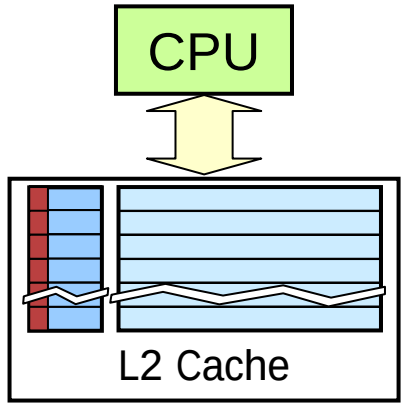


- Many existing coarse-grain optimizations
- Add new structures to track coarse-grain information

Hard to justify for a commercial design

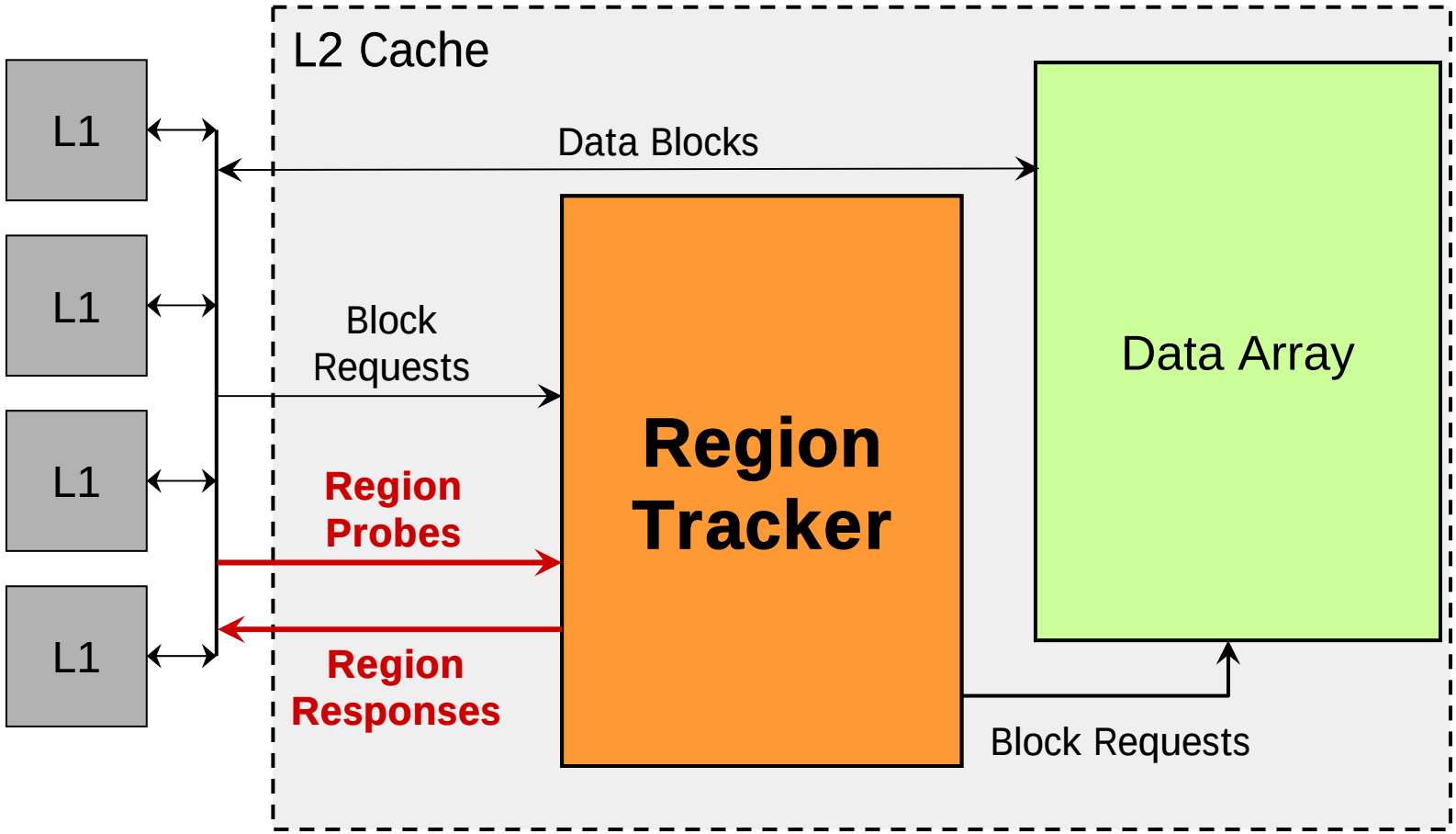
Coarse-Grain Framework

- Embed coarse-grain information in tag array
- Support many different optimizations with less area overhead



Adaptable optimization FRAMEWORK

RegionTracker Solution



Manage **blocks**, but also track and manage **regions**

RegionTracker Summary

- Replace conventional tag array:
  - 4-core CMP with 8MB shared L2 cache
  - Within 1% of original performance
  - Up to 20% less tag area
  - Average 33% less energy consumption

- **Optimization Framework**:
  - Stealth Prefetching: same performance, 36% less area
  - RegionScout: 2x more snoops avoided, no area overhead

Road Map

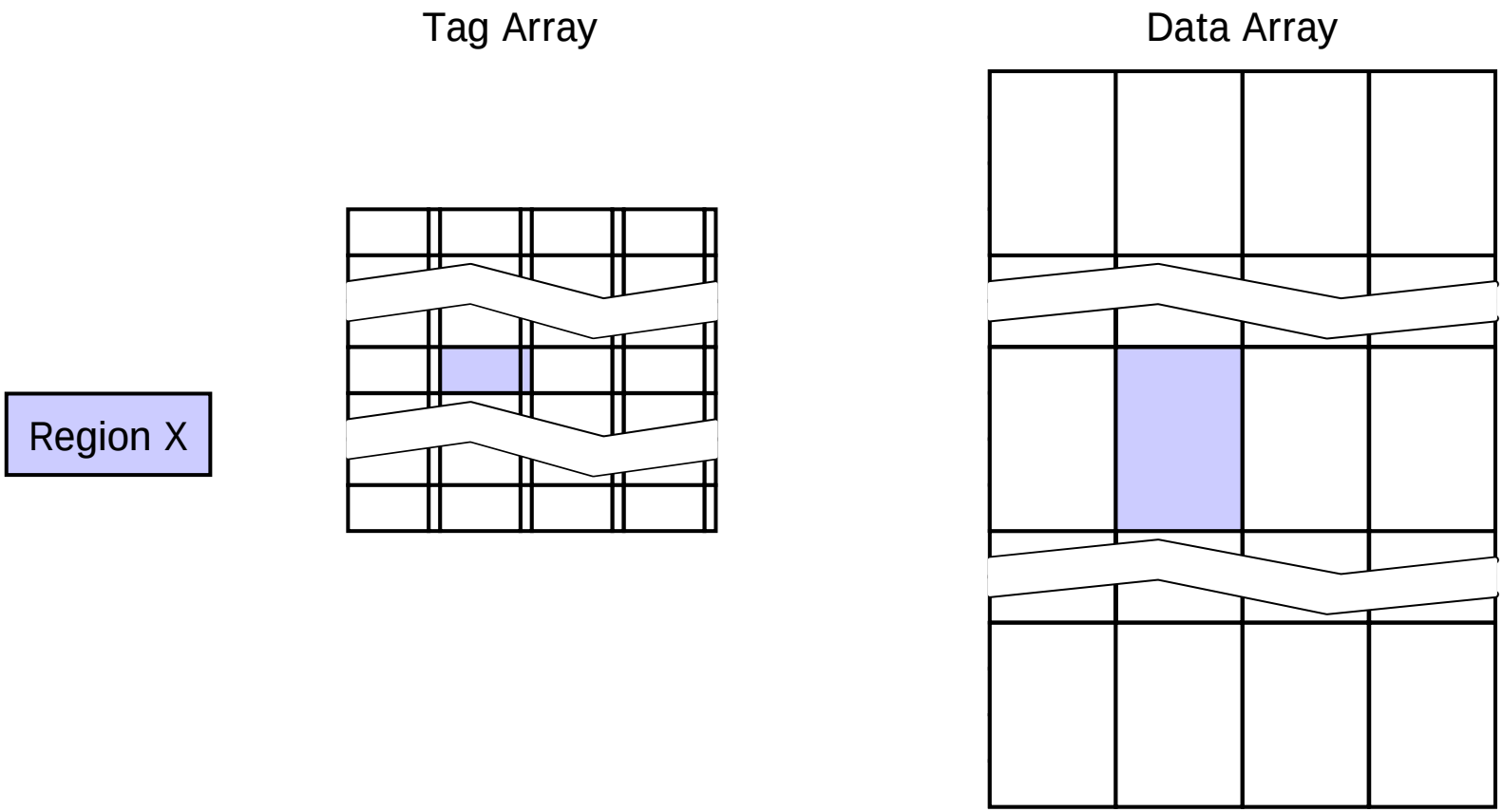
- Introduction
- **Goals**
- Coarse-Grain Cache Designs
- RegionTracker: A Tag Array Replacement
- RegionTracker: An Optimization Framework
- Conclusion

Goals

1. Conventional Tag Array Functionality
   - Identify data block location and state
   - Leave data array un-changed

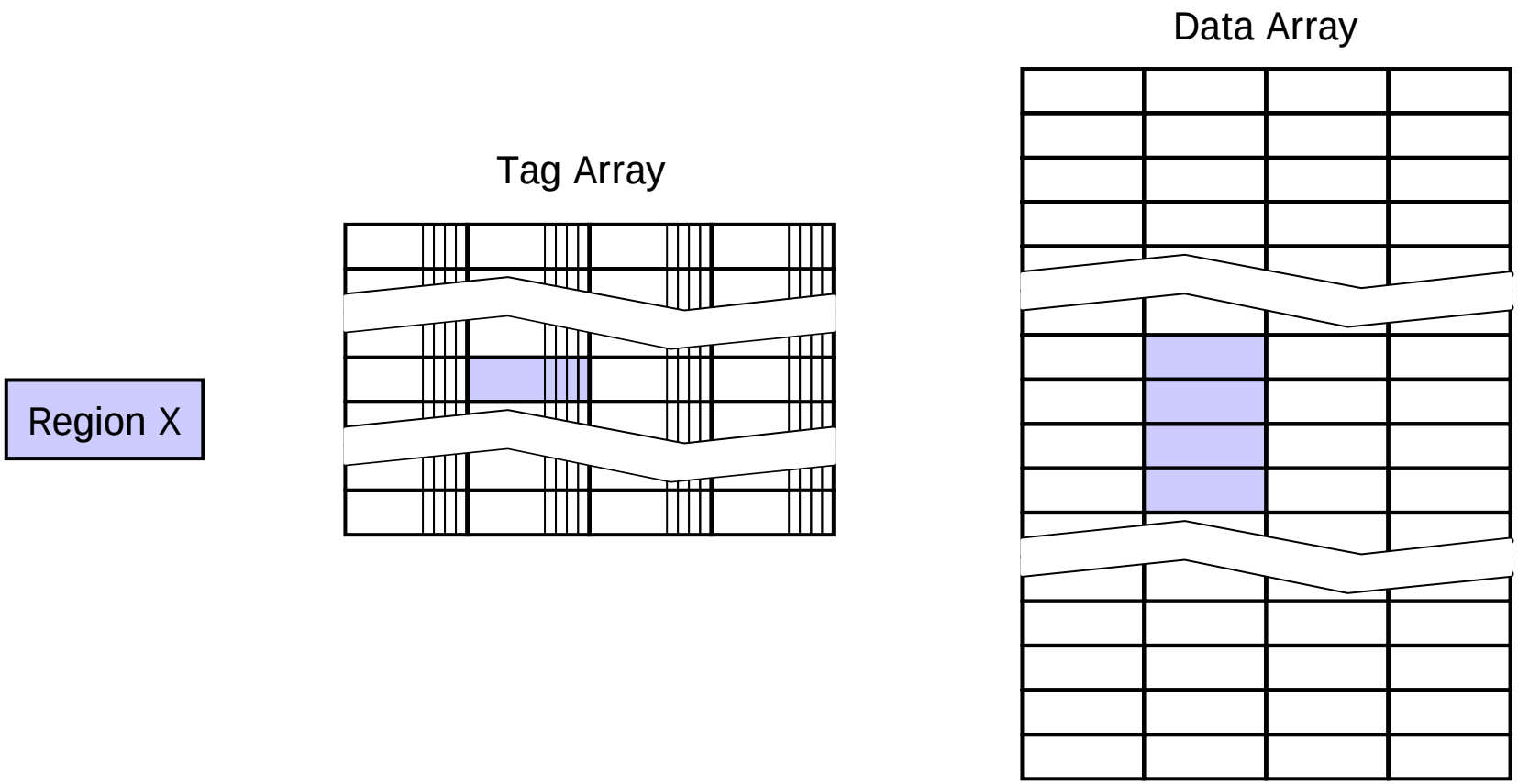
2. Optimization Framework Functionality
   - Is Region X cached?
   - Which blocks of Region X are cached? Where?
   - Evict or migrate Region X

Large Block Size



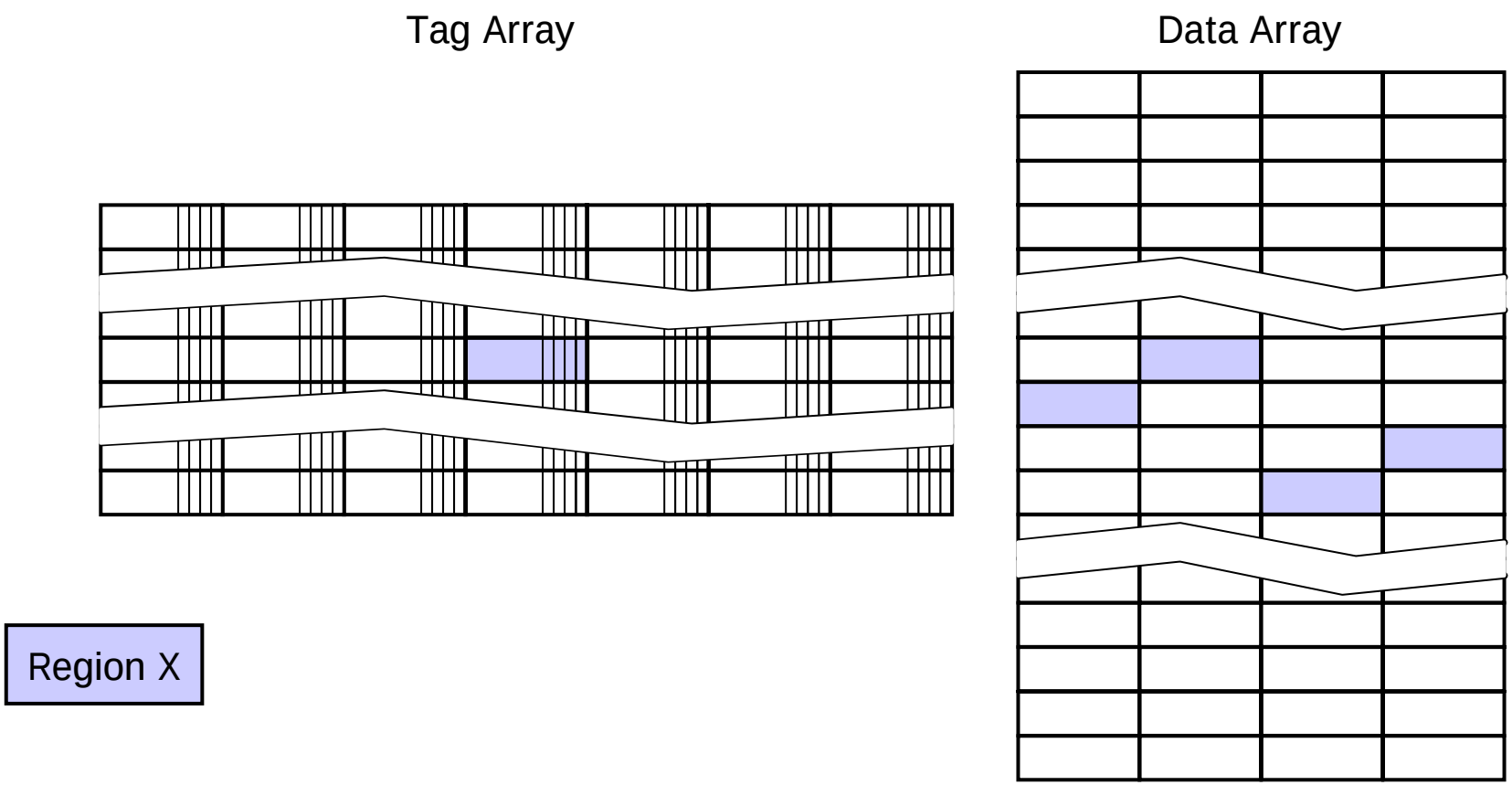
- **Increased BW, Decreased hit-rates**

Sector Cache



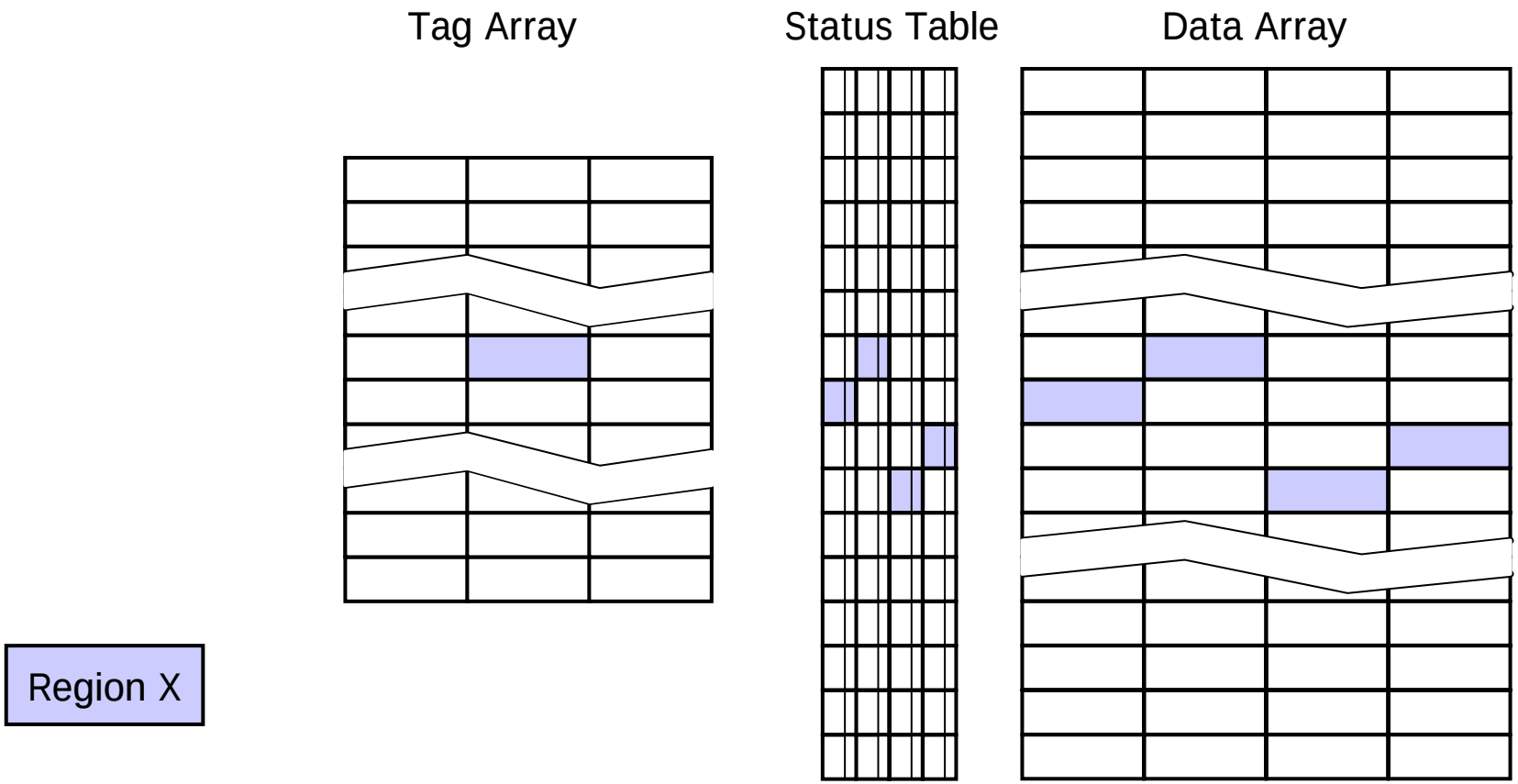
- **Decreased hit-rates**

Sector Pool Cache



- **High Associativity (2 - 4 times)**

Decoupled Sector Cache



- **Region information not exposed**
- **Region replacement requires scanning multiple entries**

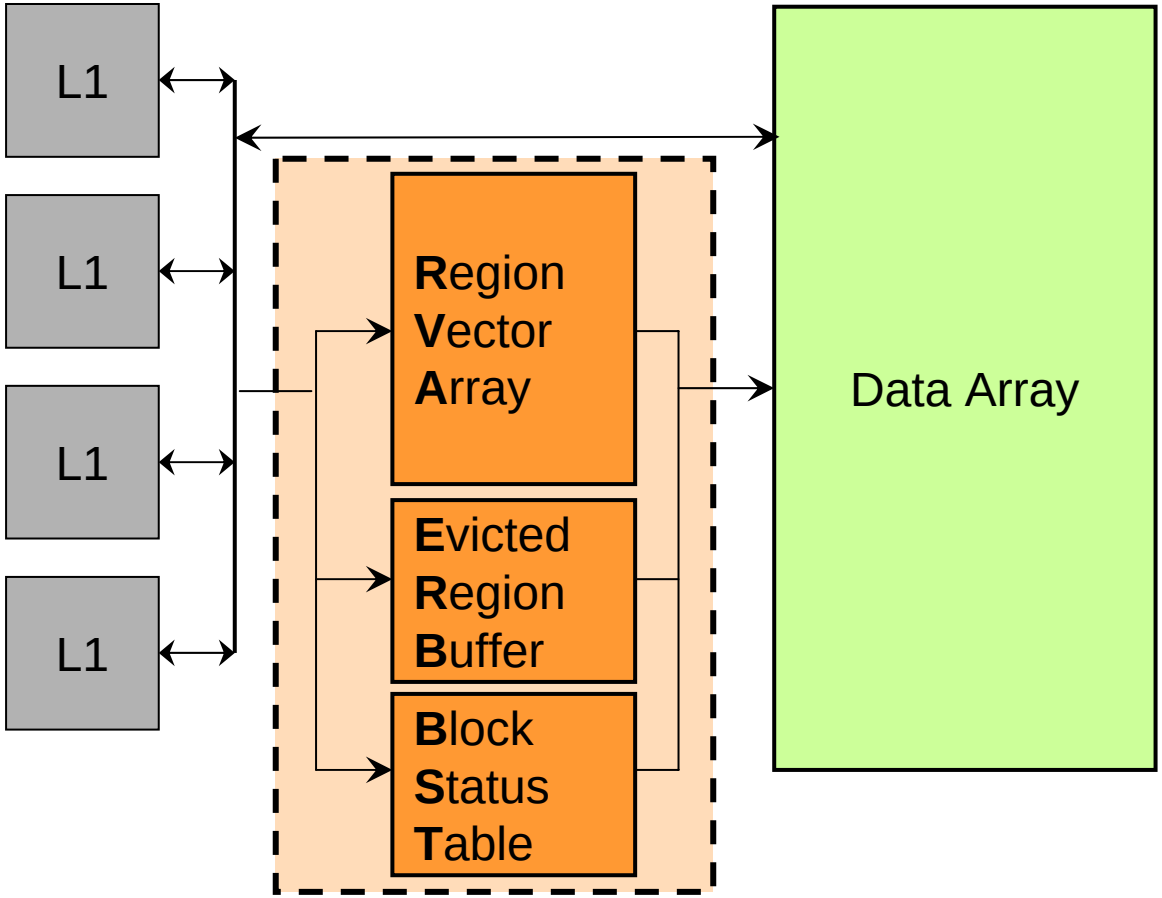
Design Requirements

- Small block size (64B)
- Miss-rate does not increase
- Lookup associativity does not increase
- No additional access latency
  - (i.e., No scanning, no multiple block evictions)

- Does not increase latency, area, or energy
- Allows banking and interleaving

- **Fit in conventional tag array "envelope"**

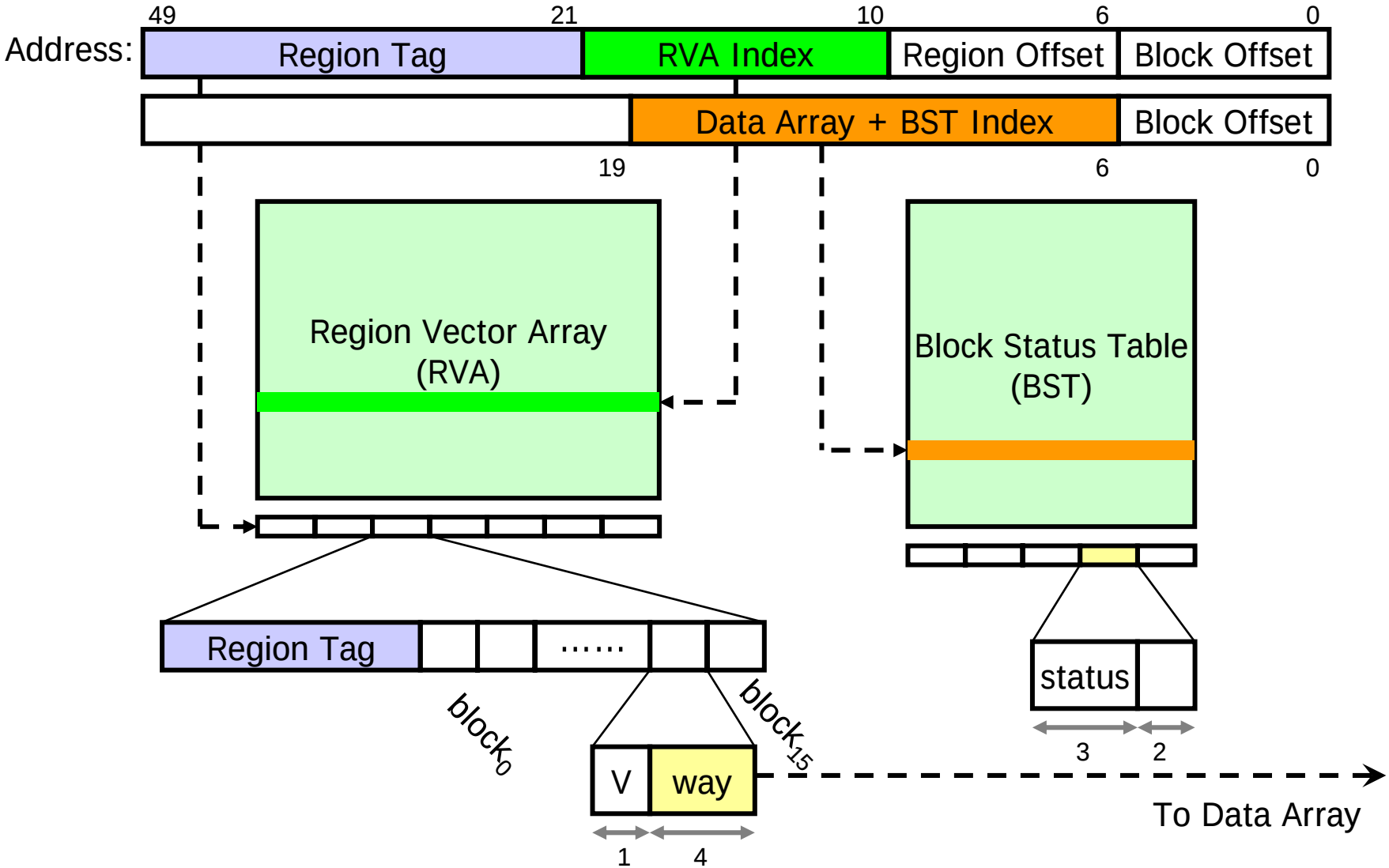
RegionTracker: A Tag Array Replacement



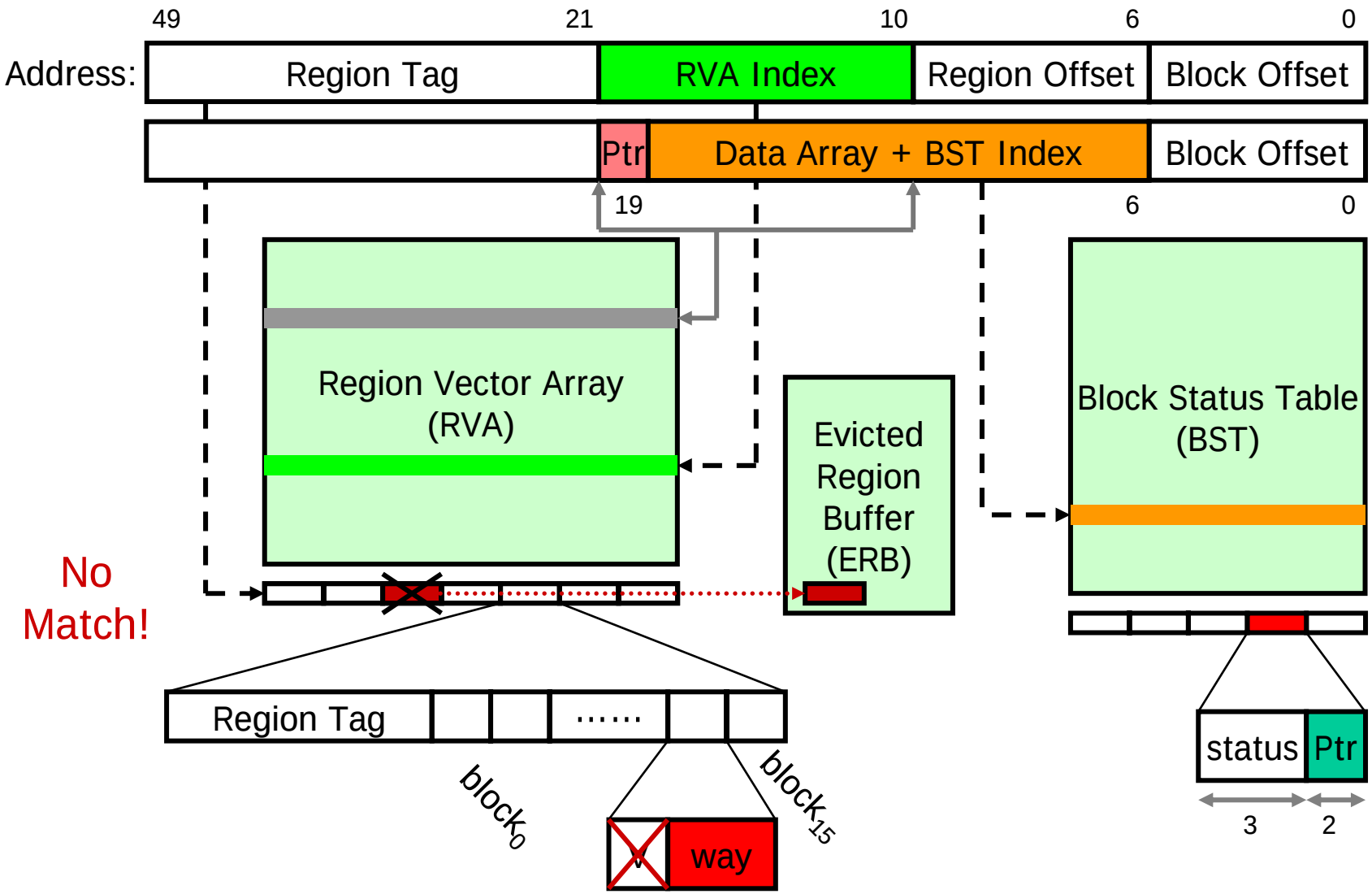
- **3 SRAM arrays, combined smaller than tag array**

Common Case: Hit

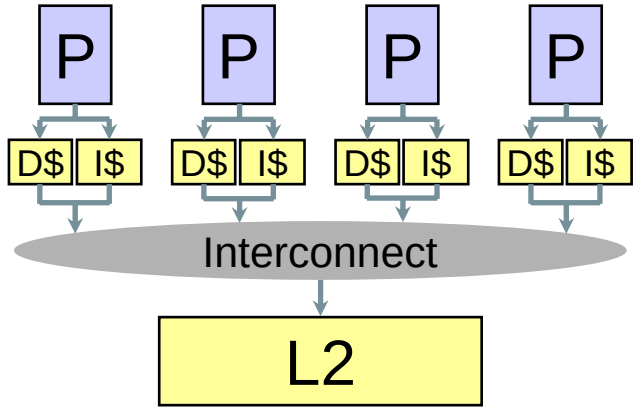
Ex: 8MB, 16-way set-associative cache, 64-byte blocks, 1KB region



Worst Case (Rare): Region Miss

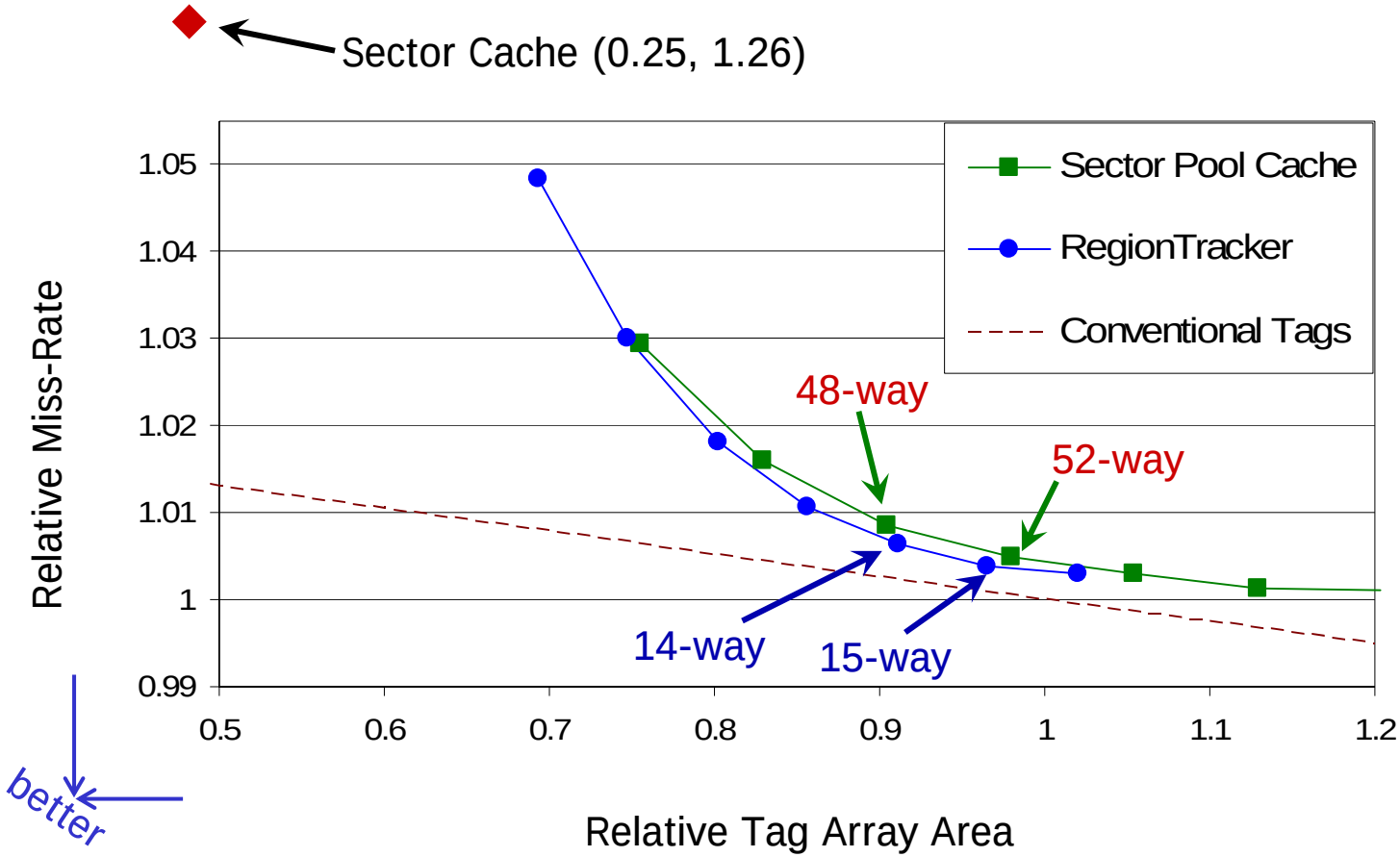


Methodology



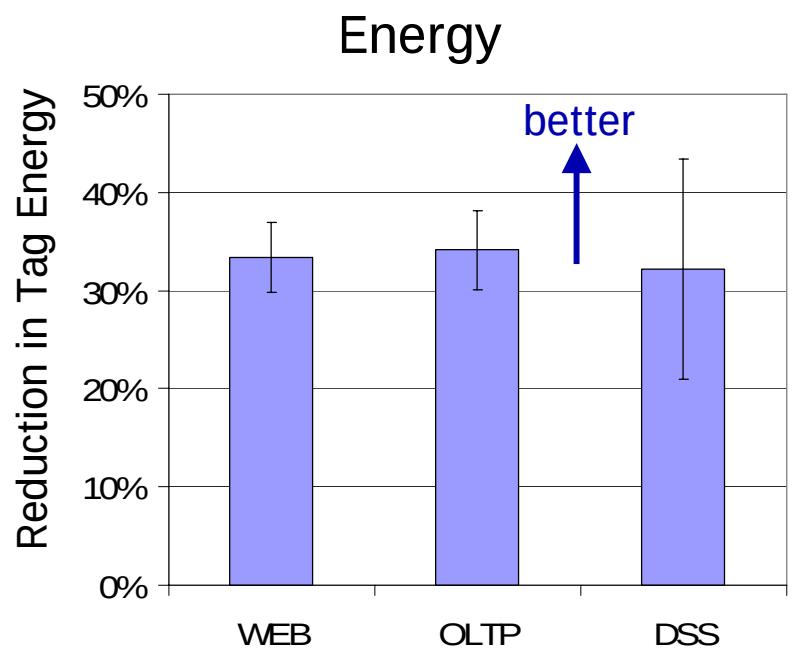
- Flexus simulator from CMU SimFlex group
  - Based on Simics full-system simulator
- 4-core CMP modeled after Piranha
  - Private 32KB, 4-way set-associative L1 caches
  - Shared 8MB, 16-way set-associative L2 cache
  - 64-byte blocks
- **Miss-rates**: Functional simulation of 2 billion instructions per core
- **Performance and Energy**: Timing simulation using SMARTS sampling methodology
- **Area and Power**: Full custom implementation on 130nm commercial technology
- 9 commercial workloads:
  - WEB: SpecWEB on Apache and Zeus
  - OLTP: TPC-C on DB2 and Oracle
  - DSS: 5 TPC-H queries on DB2

Miss-Rates vs. Area



- Sector Cache: 512KB sectors, SPC and RT: 1KB regions
- Trade-offs comparable to conventional cache

Performance & Energy



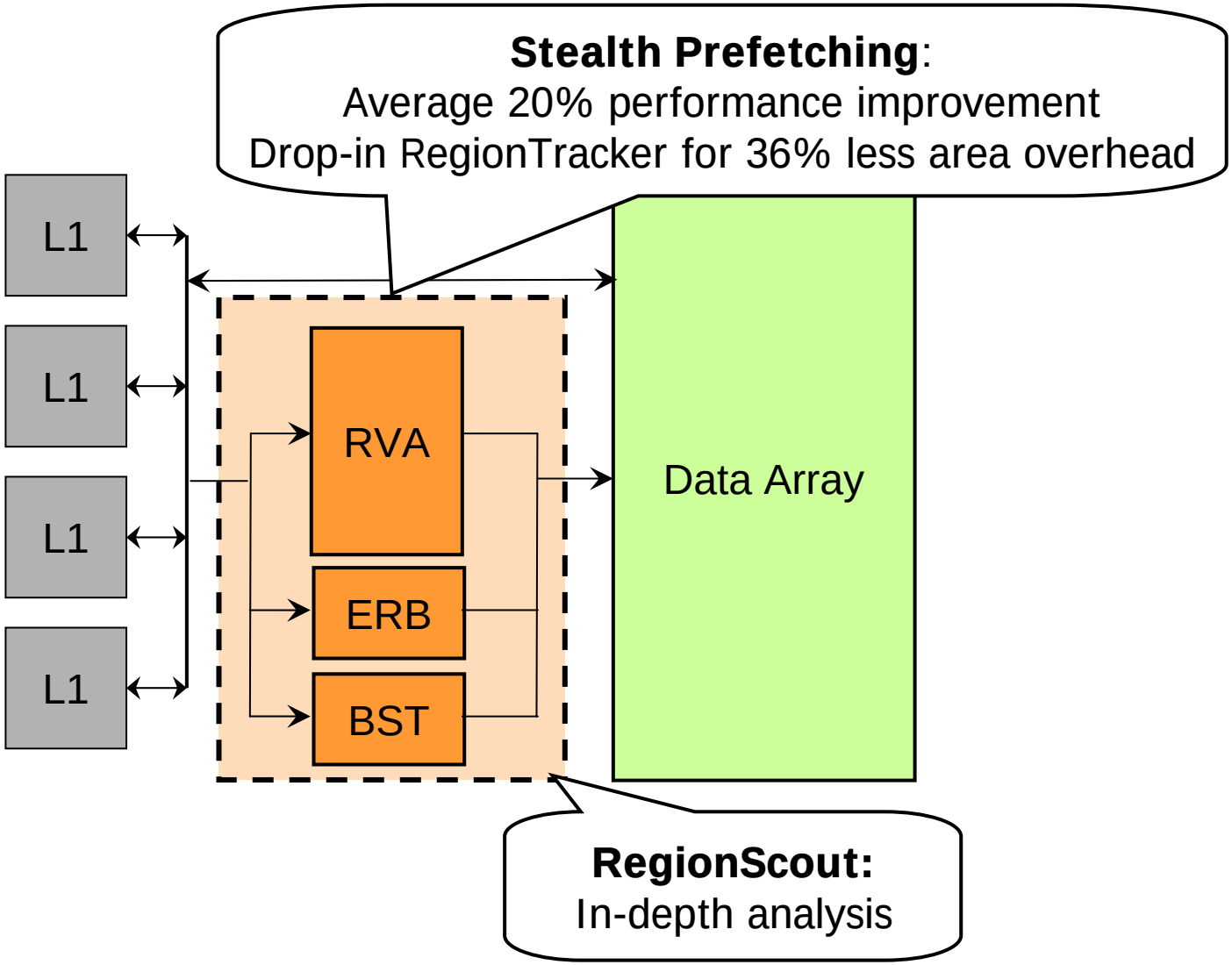
- 12-way set-associative RegionTracker: 20% less area
- Error bars: 95% confidence interval

- Performance within 1%, with 33% tag energy reduction

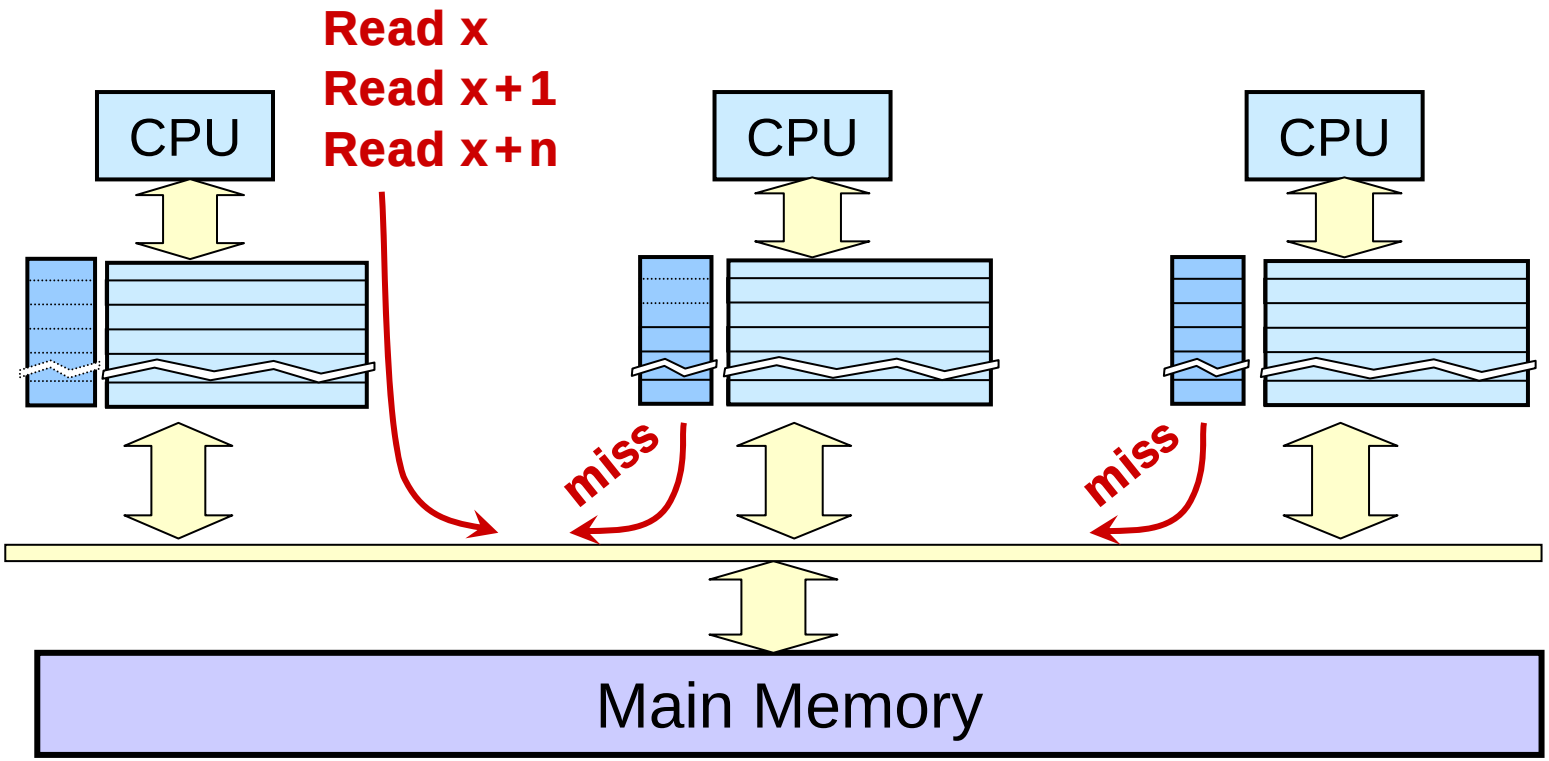
Road Map

- Introduction
- Goals
- Coarse-Grain Cache Designs
- RegionTracker: A Tag Array Replacement
- **RegionTracker: An Optimization Framework**
- Conclusion

RegionTracker: An Optimization Framework

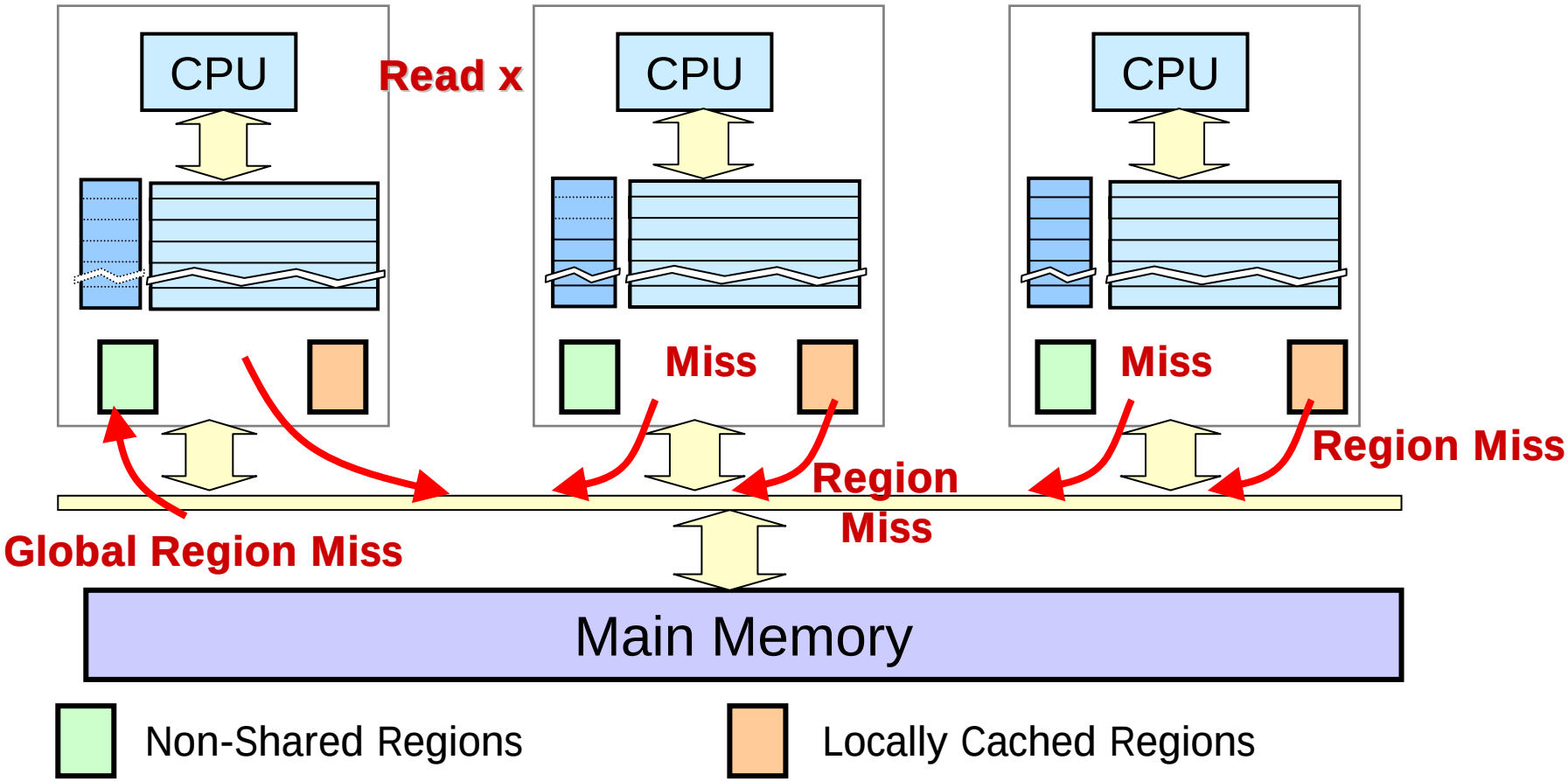


Snoop Coherence: Common Case



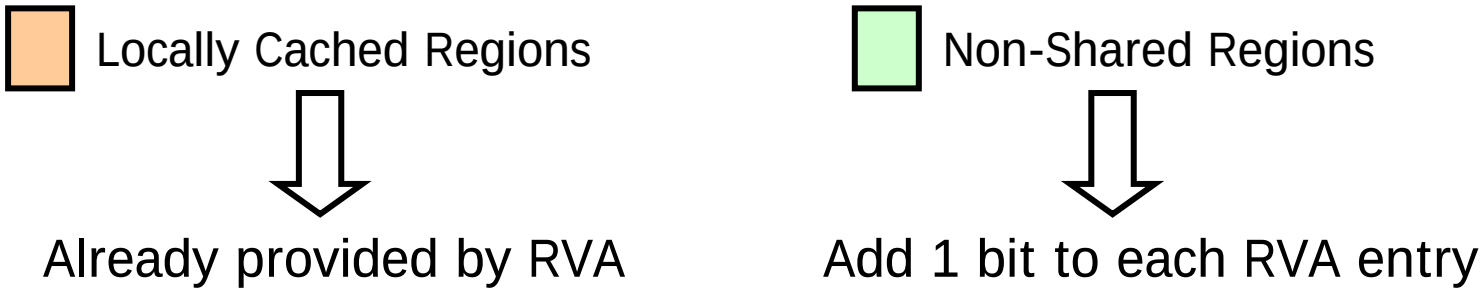
Many snoops are to non-shared regions

RegionScout



Eliminate broadcasts for non-shared regions

RegionTracker Implementation



- Minimal overhead to support RegionScout optimization
- **Still uses less area than conventional tag array**

RegionTracker + RegionScout

- 4 processors, 512KB L2 Caches
- 1KB regions

New optimization possible with RegionTracker

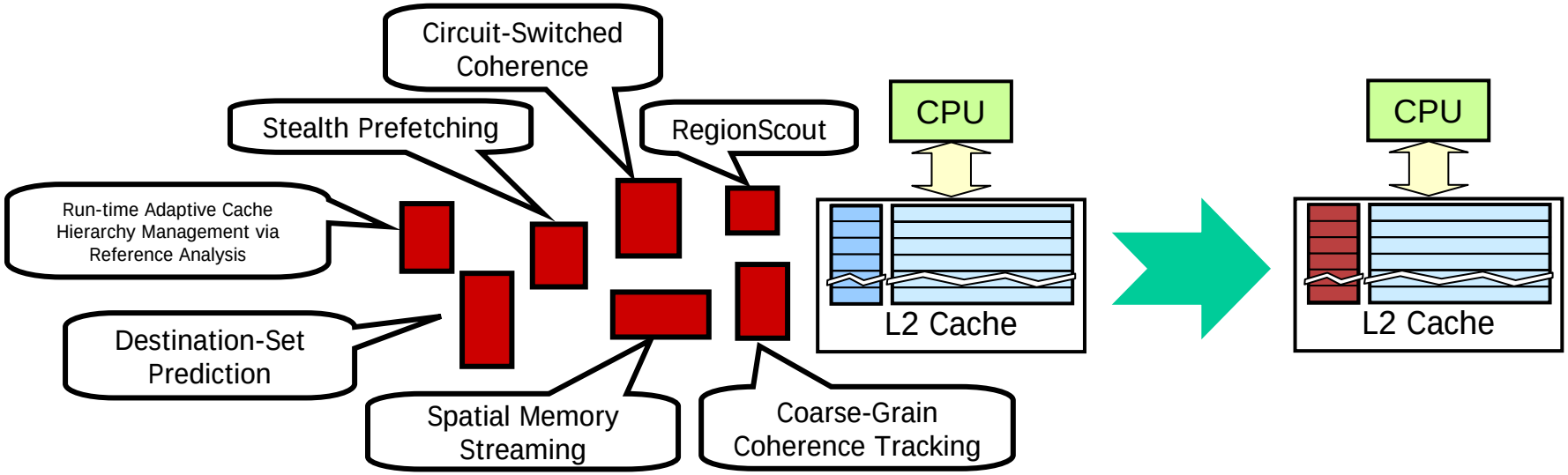
Avoid 41% of Snoop Broadcasts, no area overhead compared to conventional tag array

Result Summary

- Replace Conventional Tag Array:
  - 20% Less tag area
  - 33% Less tag energy
  - Within 1% of original performance

- Coarse-Grain Optimization Framework:
  - 36% reduction in area overhead for Stealth Prefetching
  - Filter 41% of snoop broadcasts with no area overhead compared to conventional cache

Conclusion



RegionTracker framework makes coarse-grain optimizations more attractive

A Framework for Coarse-Grain Optimizations in the On-Chip Memory Hierarchy

Jason Zebchuk, Elham Safi, and Andreas Moshovos
{**zebchuk**,elham,moshovos}@eecg.toronto.edu

AENAO Research Group
Department of Electrical and Computer Engineering
University of Toronto