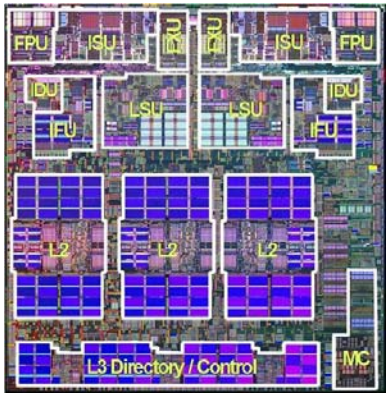
Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors

Michael Zhang & Krste Asanovic

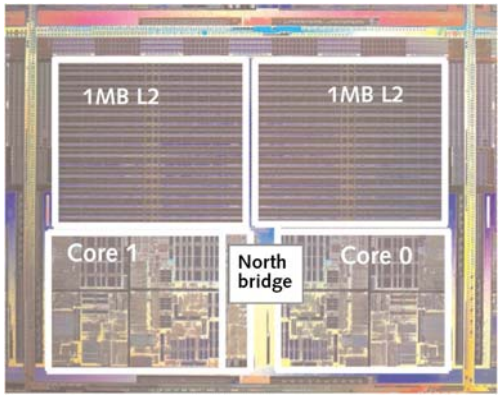
Computer Architecture Group

MIT CSAIL

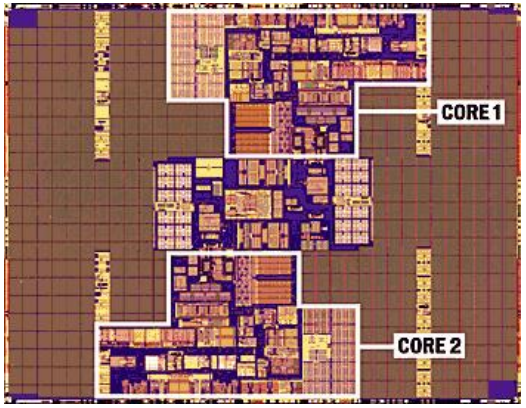
Chip Multiprocessors (CMPs) are Here



IBM Power5
with 1.9MB L2



AMD Opteron
with 2MB L2



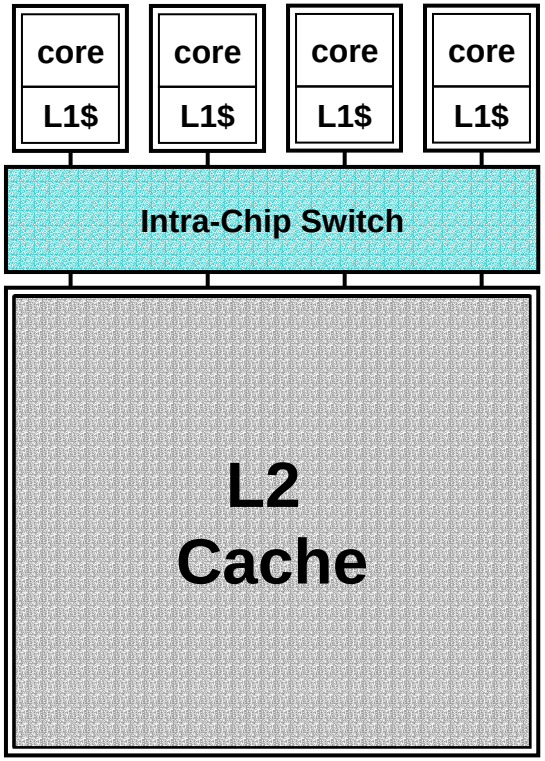
Intel Montecito
With 24MB L3

- **Easily utilizes on-chip transistors**
- **Naturally exploits thread-level parallelism**
- **Dramatically reduces design complexity**

- **Future CMPs will have *more* processor cores**
- **Future CMPs will have *more* cache**

CSAIL

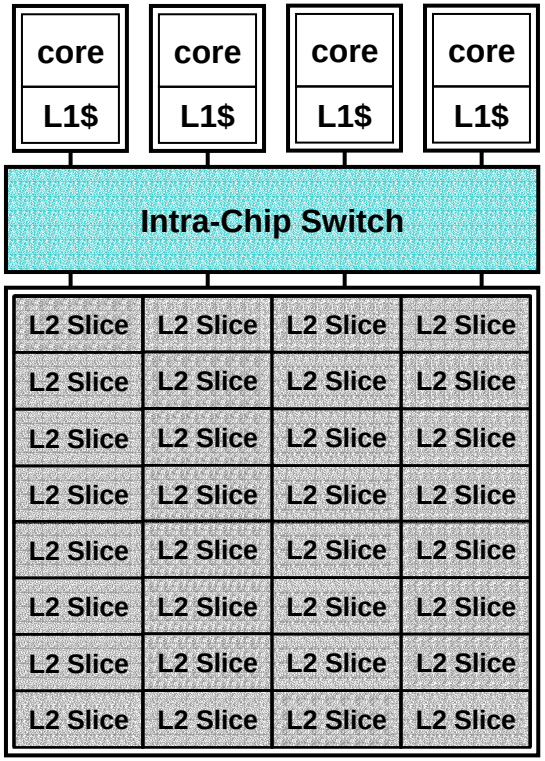
Current Chip Multiprocessors



A 4-node CMP with a large L2 cache

- ***Layout*: "Dance-Hall"**
  - *Core + L1 cache*
  - *L2 cache*
- ***Small L1 cache:* Very low access latency**
- ***Large L2 cache:* Divided into slices to minimize access latency and power usage**

Current Chip Multiprocessors

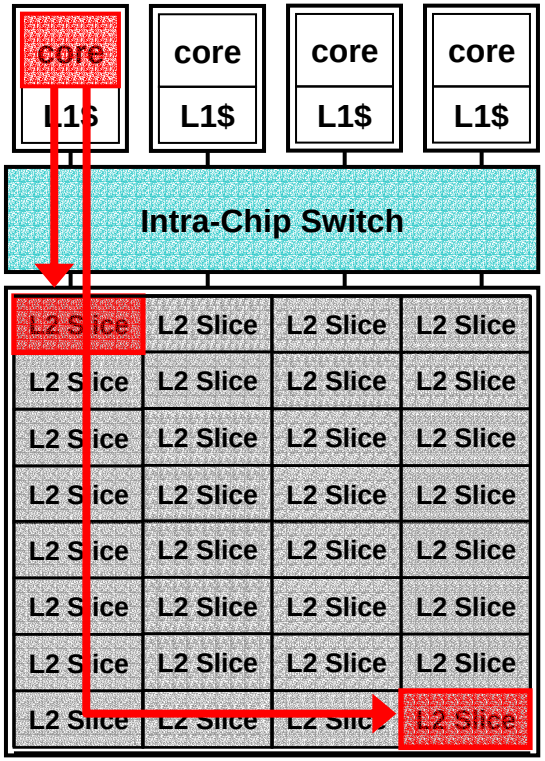


A 4-node CMP with a large L2 cache

- ***Layout*: "Dance-Hall"**
  - *Core + L1 cache*
  - *L2 cache*
- ***Small L1 cache:* Very low access latency**
- ***Large L2 cache:* Divided into slices to minimize access latency and power usage**

CSAIL

Increasing CMP Cache Capacities lead to Non-Uniform Cache Access Latency (NUCA)



A 4-node CMP with a large L2 cache

- ***Current:*** **Caches are designed with (long) uniform access latency for the worst case:**

  Best Latency == Worst Latency

- ***Future:*** **Must design with non-uniform access latencies depending on the on-die location of the data:**

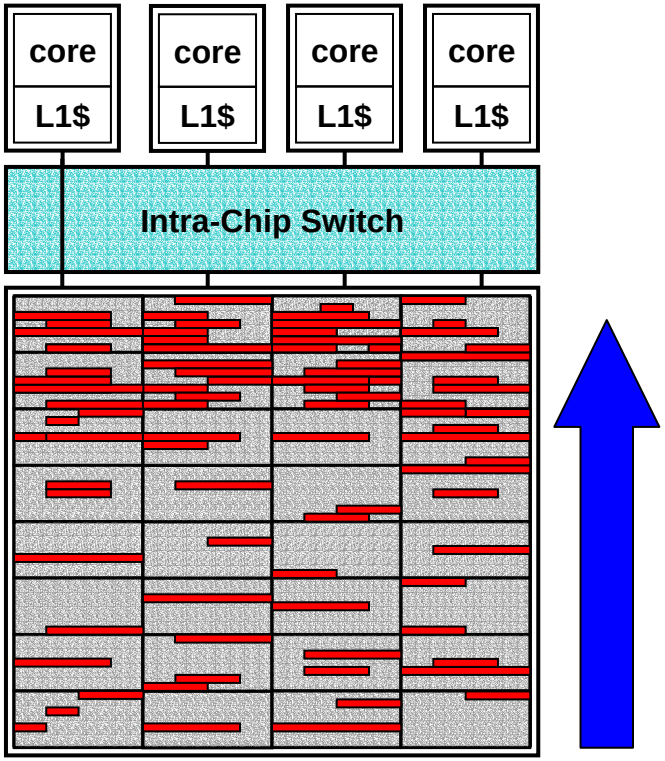
  Best Latency << Worst Latency

- ***Challenge:*** **How to minimize average cache access latency:**

  Average Latency → Best Latency

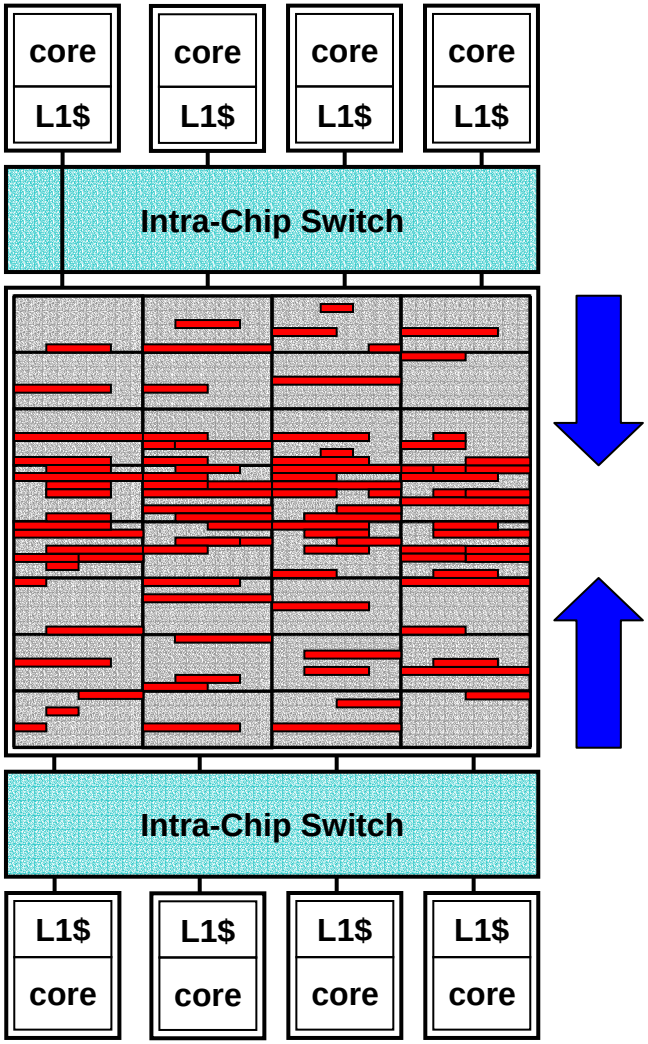
CSAIL

Current Research on NUCAs



- **Targeting uniprocessor machines**
- ***Data Migration*: Intelligently place data such that the active working set resides in cache slices closest to the processor**
  - *D-NUCA* [*ASPLOS-X, 2002*]
  - *NuRAPID* [*MICRO-37, 2004*]

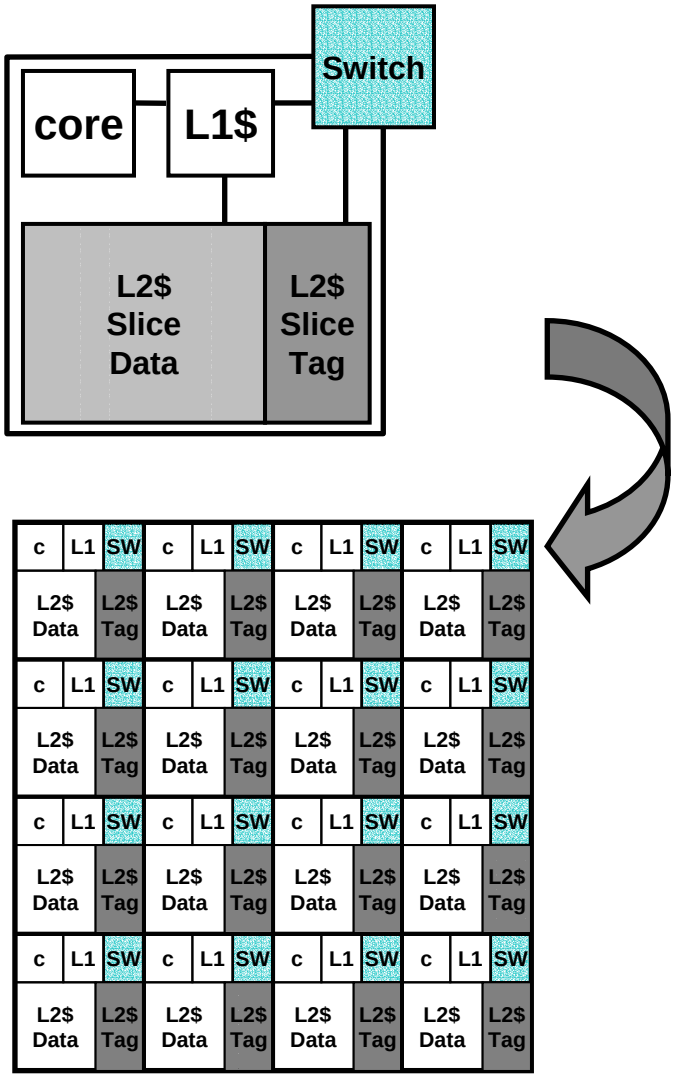
Data Migration does not Work Well with CMPs



- ***Problem:*** **The unique copy of the data cannot be close to all of its sharers**
- ***Behavior:*** **Over time, shared data migrates to a location equidistant to all sharers**
  - *Beckmann & Wood [MICRO-36, 2004]*

CSAIL

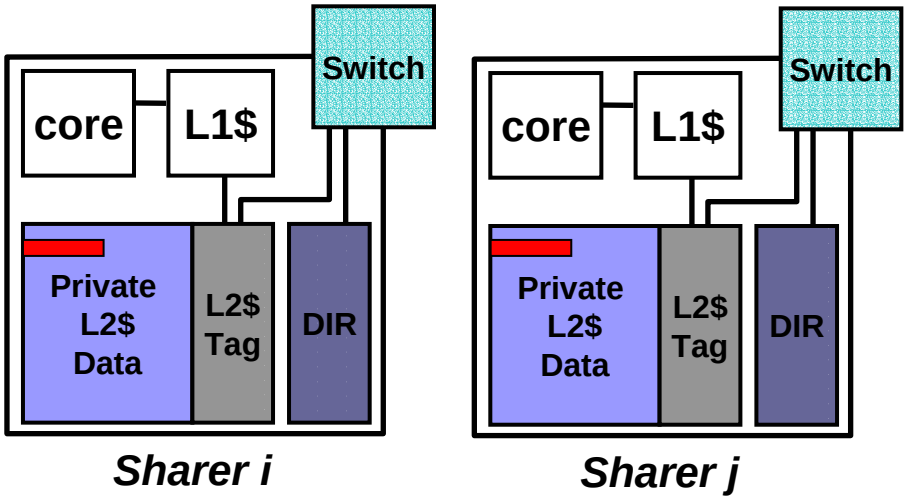
This Talk: Tiled CMPs with Directory-Based Cache Coherence Protocol



- **Tiled CMPs for *Scalability***
  - *Minimal redesign effort*
  - *Use directory-based protocol for scalability*
- **Managing the L2s to minimize the effective access latency**
  - *Keep data close to the requestors*
  - *Keep data on-chip*
- **Two baseline L2 cache designs**
  - *Each tile has own private L2*
  - *All tiles share a single distributed L2*

CSAIL

Private L2 Design Provides Low Hit Latency

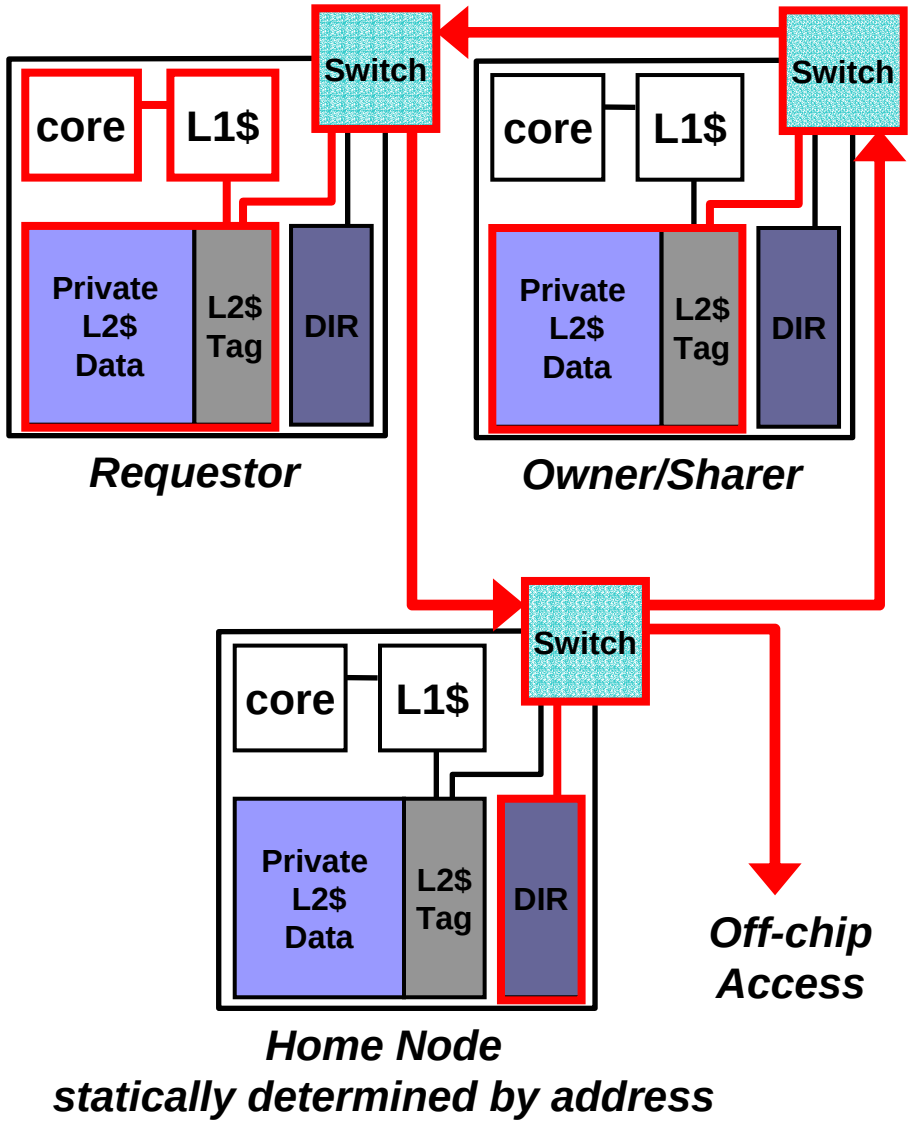


- **The local L2 slice is used as a private L2 cache for the tile**
  - *Shared data is duplicated in the L2 of each sharer*
  - *Coherence must be kept among all sharers at the L2 level*

- **On an L2 miss:**
  - *Data not on-chip*
  - *Data available in the private L2 cache of another chip*

CSAIL

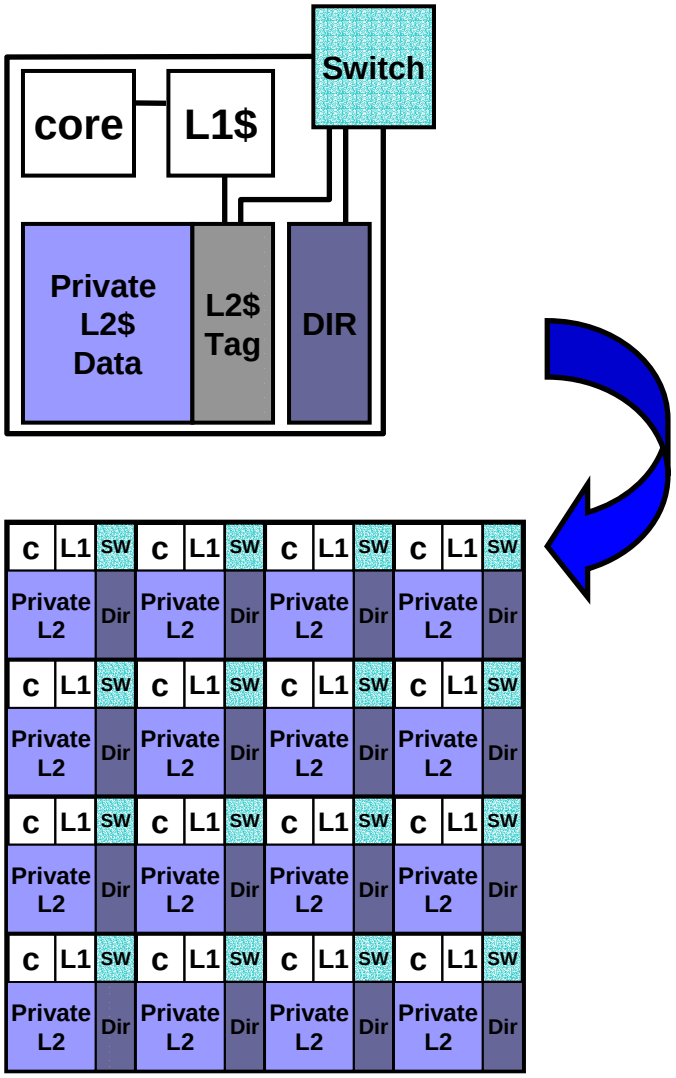
Private L2 Design Provides Low Hit Latency



- **The local L2 slice is used as a private L2 cache for the tile**
  - *Shared data is duplicated in the L2 of each sharer*
  - *Coherence must be kept among all sharers at the L2 level*
- **On an L2 miss:**
  - *Data not on-chip*
  - *Data available in the private L2 cache of another tile (cache-to-cache reply-forwarding)*

CSAIL

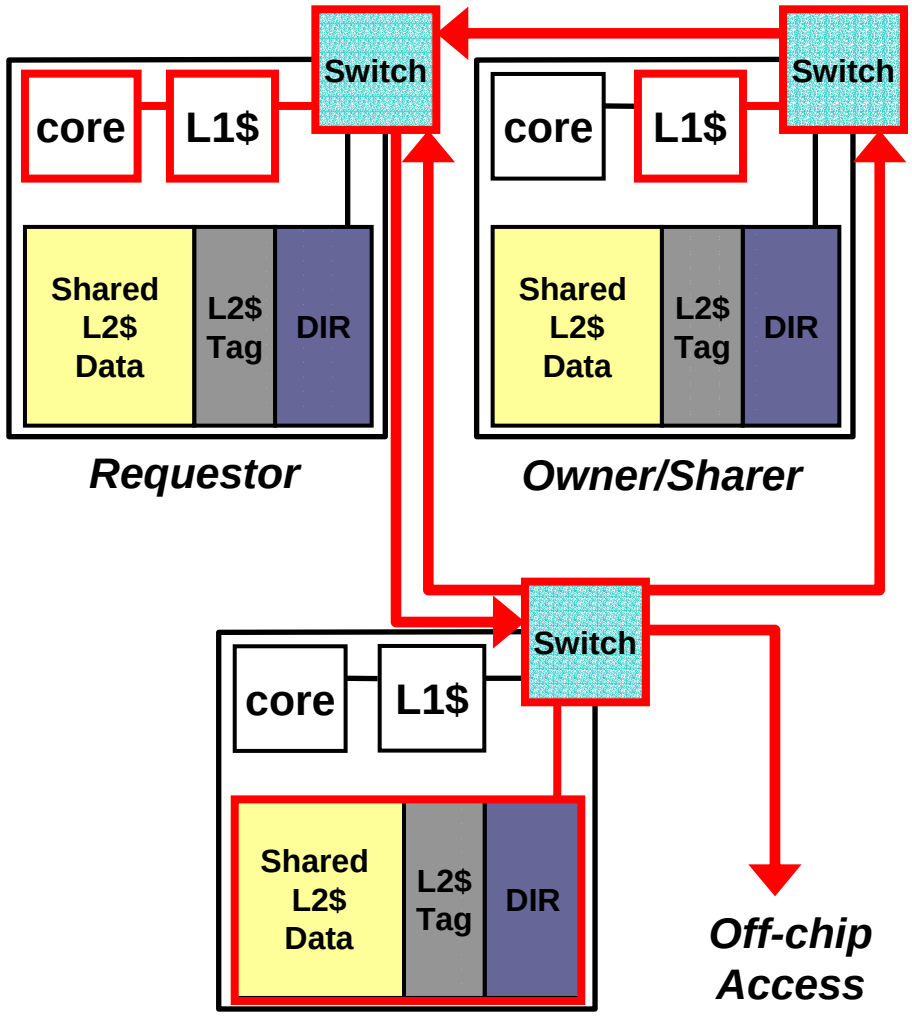
Private L2 Design Provides Low Hit Latency



- **Characteristics:**
  - *Low hit latency to resident L2 data*
  - *Duplication reduces on-chip capacity*
- **Works well for benchmarks with working sets that fits into the local L2 capacity**

CSAIL

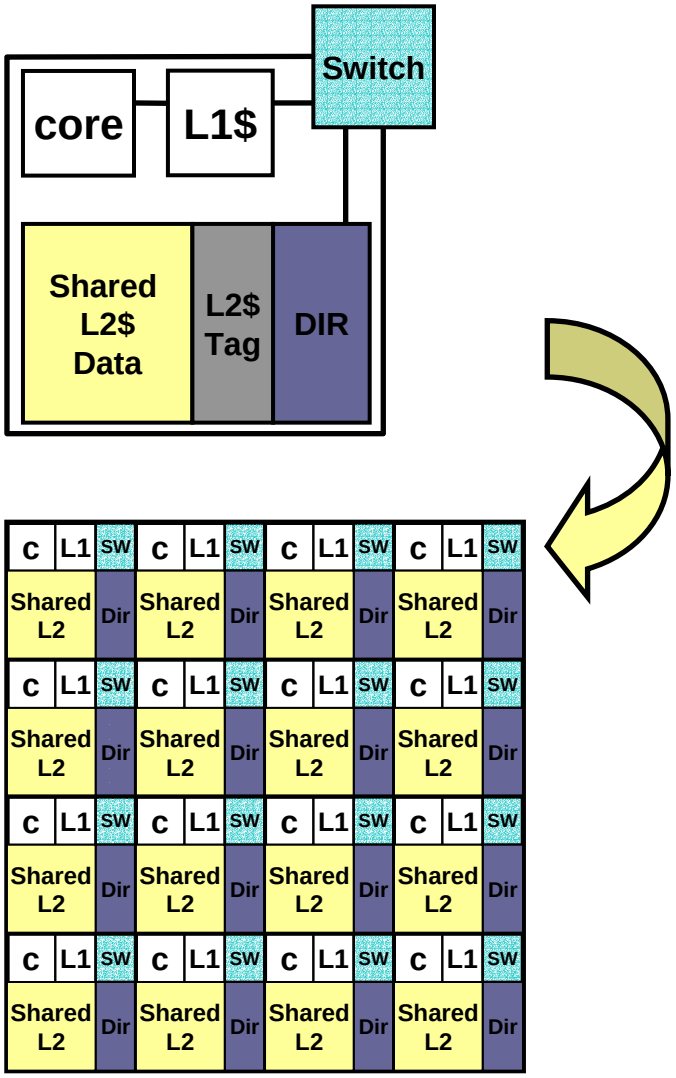
Shared L2 Design Provides Maximum Capacity



- **All L2 slices on-chip form a distributed shared L2, backing up all L1s**
  - *No duplication, data kept in a unique L2 location*
  - *Coherence must be kept among all sharers at the L1 level*
- **On an L2 miss:**
  - *Data not in L2*
  - *Coherence miss (cache-to-cache reply-forwarding)*

CSAIL

Shared L2 Design Provides Maximum Capacity



- **Characteristics:**
  - *Maximizes on-chip capacity*
  - *Long/non-uniform latency to L2 data*
- **Works well for benchmarks with larger working sets to minimize expensive off-chip accesses**

CSAIL

Victim Replication: A Hybrid Combining the Advantages of Private and Shared Designs

- **Private design characteristics:**
  - *Low L2 hit latency to resident L2 data*
  - *Reduced L2 capacity*
- **Shared design characteristics:**
  - *Long/non-uniform L2 hit latency*
  - *Maximum L2 capacity*

CSAIL

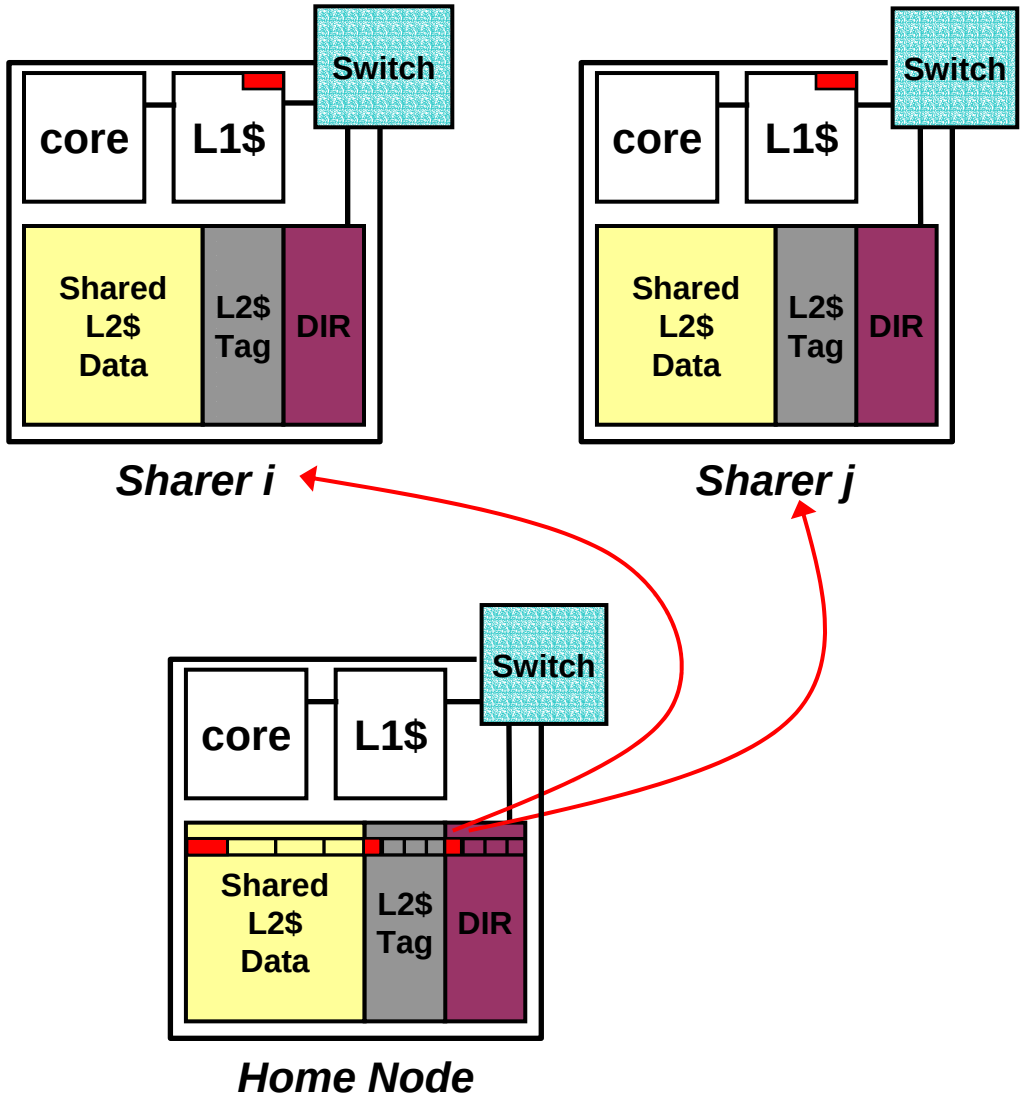
Victim Replication: A Hybrid Combining the Advantages of Private and Shared Designs

- **Private design characteristics:**
  - ➔*Low L2 hit latency to resident L2 data*
  - *Reduced L2 capacity*
- **Shared design characteristics:**
  - *Long/non-uniform L2 hit latency*
  - ➔*Maximum L2 capacity*

Victim Replication: Provides low hit latency while keeping the working set on-chip

CSAIL

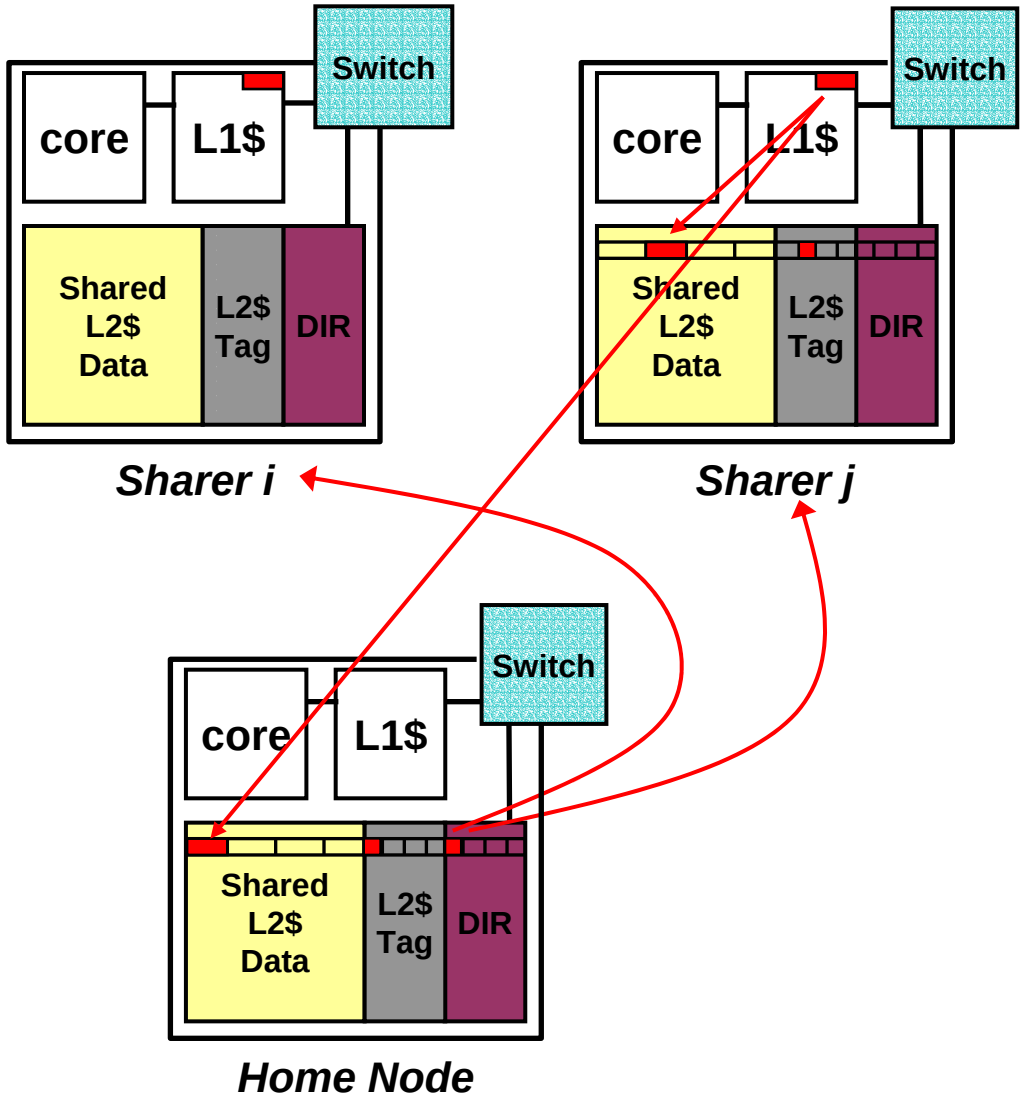
Victim Replication: A Variant of the Shared Design



- ***Implementation***: **Based on the shared design**
- ***L1 Cache***: **Replicates shared data locally for fastest access latency**
- ***L2 Cache***: **Replicates the L1 capacity victims → *Victim Replication***

CSAIL

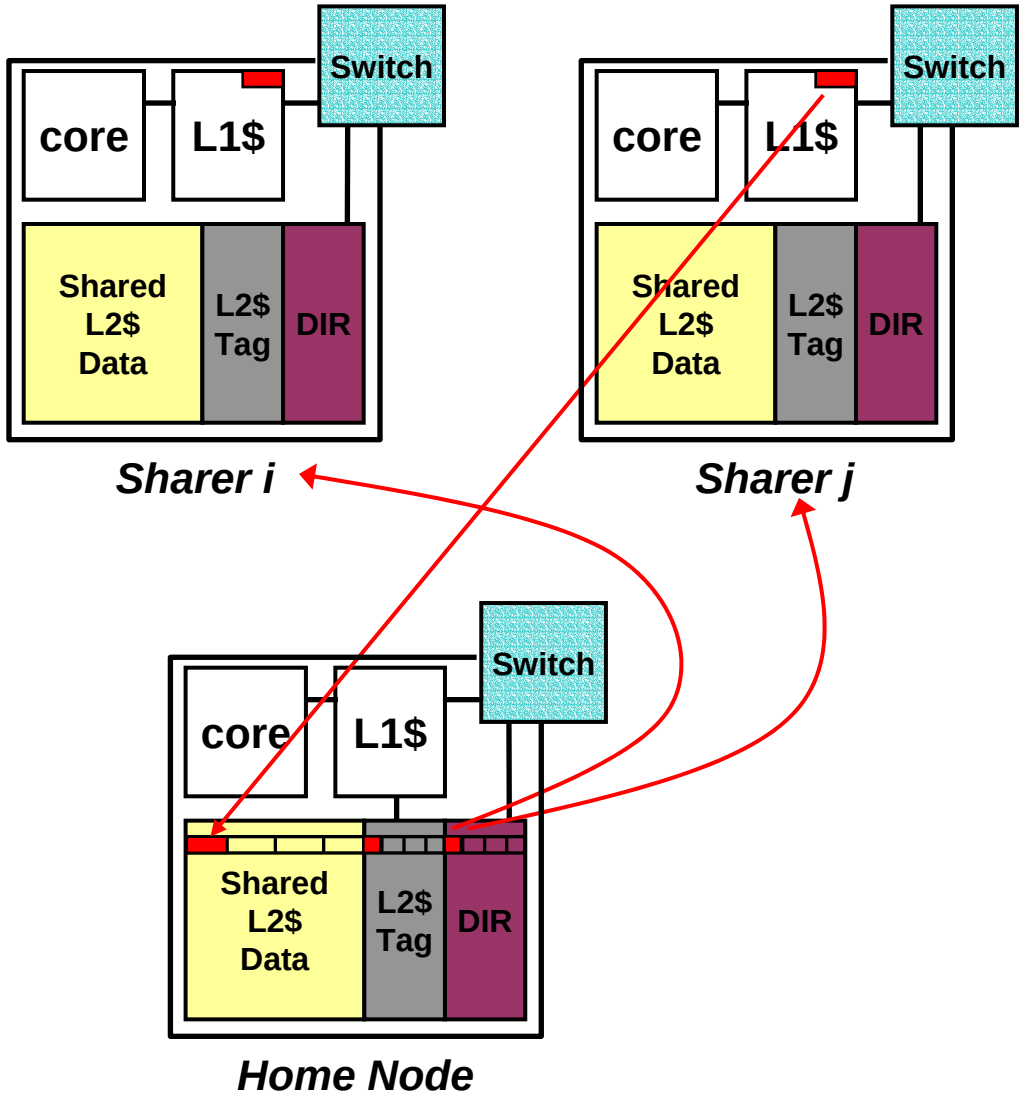
Victim Replication: The Local Tile *Replicates* the L1 *Victim* During Eviction



- ***Replicas*: L1 capacity victims stored in the *Local* L2 slice**
- ***Why?* Reused in the near future with fast access latency**
- **Which way in the target set to use to hold the replica?**

CSAIL

The Replica should *NOT* Evict More Useful Cache Blocks from the L2 Cache



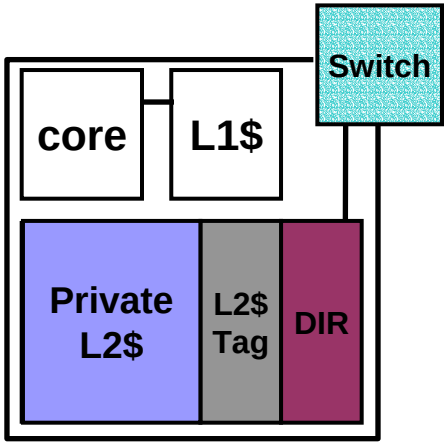
Replica is NOT always made

1. ***Invalid blocks***
2. ***Home blocks w/o sharers***
3. ***Existing replicas***
4. ***Home blocks w/ sharers***

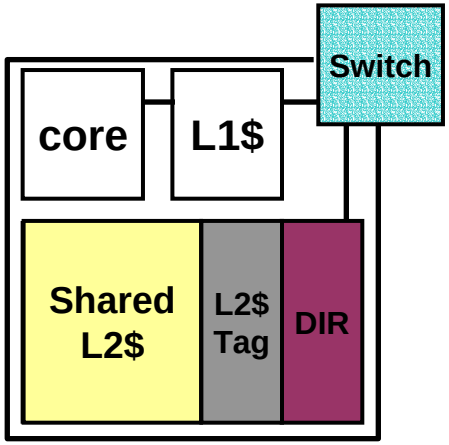
Never evict actively shared home blocks in favor of a replica

CSAIL

Victim Replication Dynamically Divides the Local L2 Slice into Private & Shared Partitions

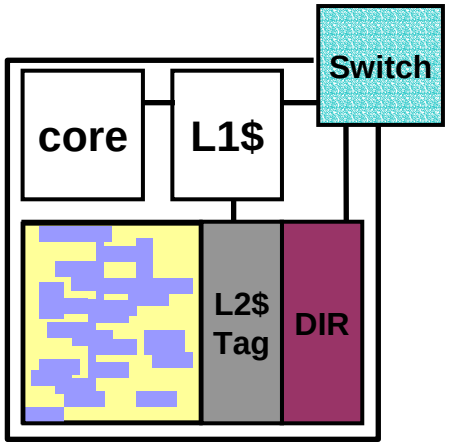


Private Design

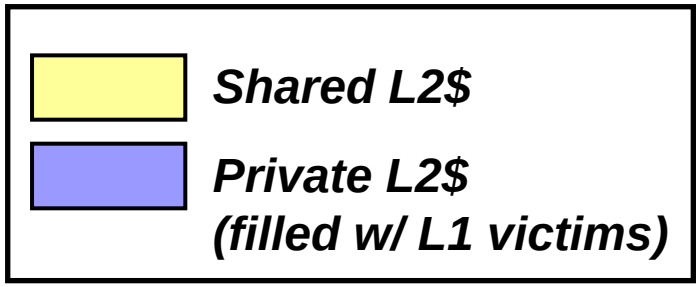


Shared Design

Victim Replication dynamically creates a large local private, victim cache for the local L1 cache



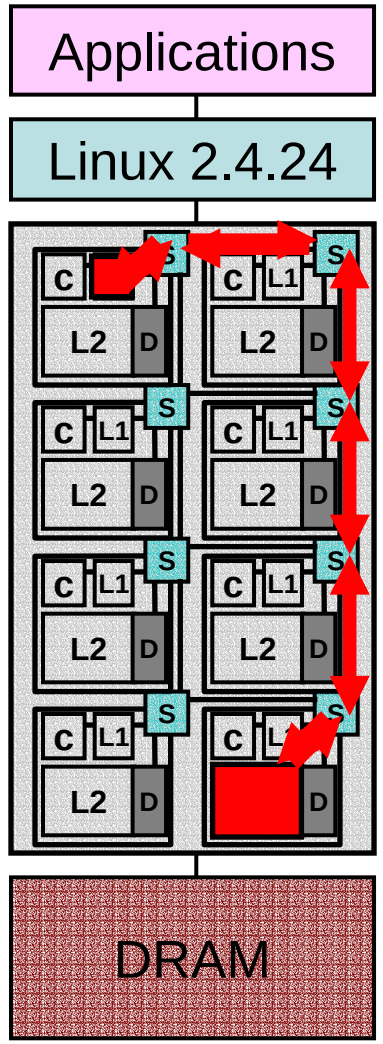
Victim Replication



CSAIL

Experimental Setup

- **Processor Model: *Bochs***
  - *Full-system x86 emulator running Linux 2.4.24*
  - *8-way SMP with single in-order issue cores*
- **All latencies normalized to one *24-F04 clock* cycle**
  - *Primary caches reachable in one cycle*
- **Cache/Memory Model**
  - *4x2 Mesh with 3 Cycle near-neighbor latency*
  - *L1I$ & L1D$: 16KB each, 16-Way, 1-Cycle, Pseudo-LRU*
  - *L2$: 1MB, 16-Way, 6-Cycle, Random*
  - *Off-chip Memory: 256 Cycles*
- ***Worst-case cross chip contention-free latency is 30 cycles***



CSAIL

The Plan for Results

- **Three configurations evaluated:**
  1. *Private L2 design → L2P*
  2. *Shared L2 design → L2S*
  3. *Victim replication → L2VR*

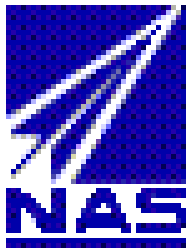
- **Three suites of workloads used:**
  1. *Multi-threaded workloads*
  2. *Single-threaded workloads*
  3. *Multi-programmed workloads*

- **Results show Victim Replication's Performance *Robustness***

CSAIL

Multithreaded Workloads

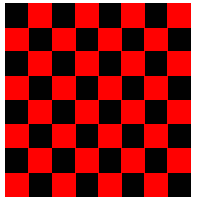
- **8 NASA Advanced Parallel Benchmarks:**
  - *Scientific (computational fluid dynamics)*
  - *OpenMP (loop iterations in parallel)*
  - *Fortran: ifort –v8 –O2 –openmp*



- **2 OS benchmarks**

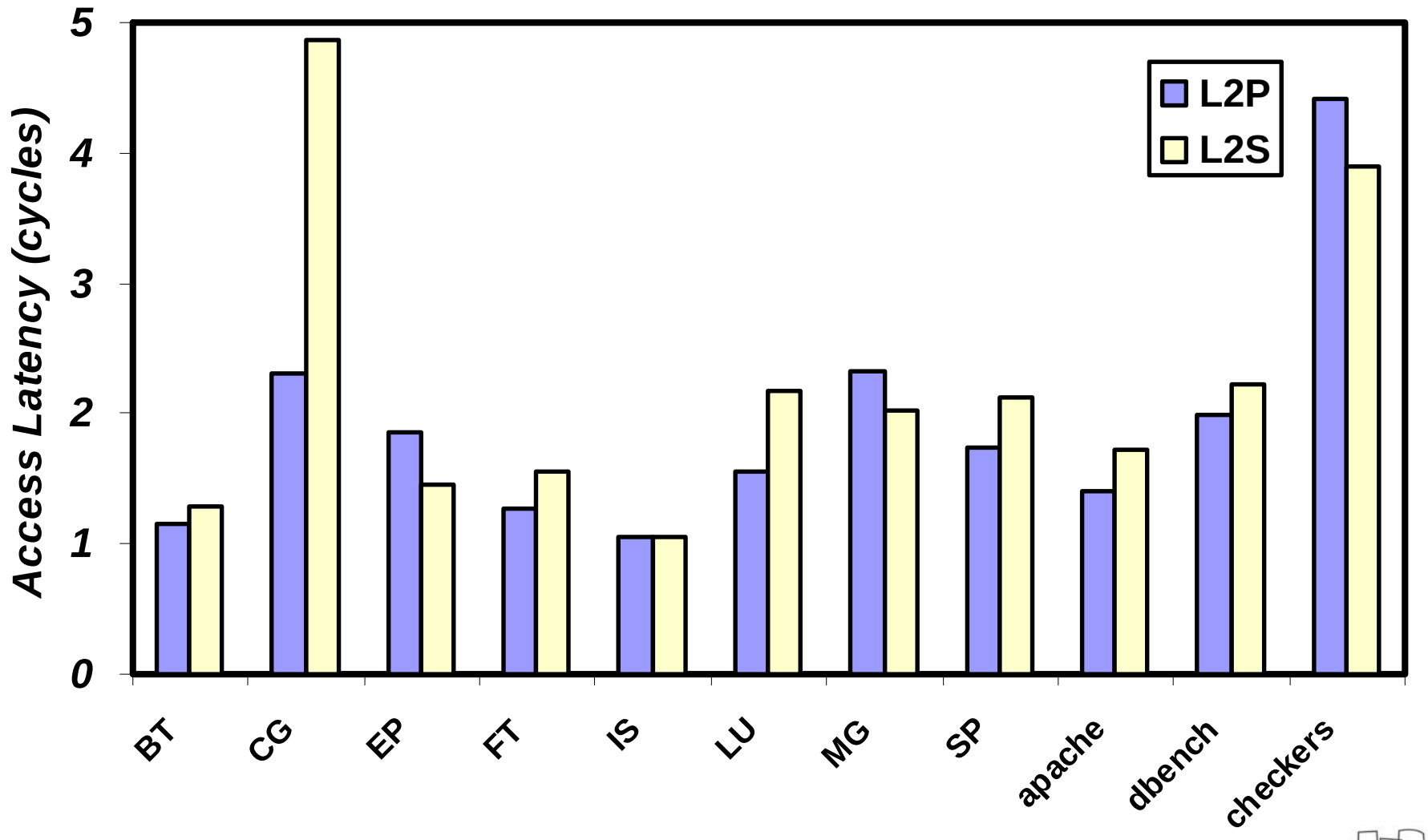
  - *dbench: (Samba) several clients making file-centric system calls*
  - *apache: web server with several clients (via loopback interface)*
  - *C: gcc 2.96*

- **1 AI benchmark: Cilk checkers**
  - *spawn/sync primitives: dynamic thread creation/scheduling*
  - *Cilk: gcc 2.96, Cilk 5.3.2*



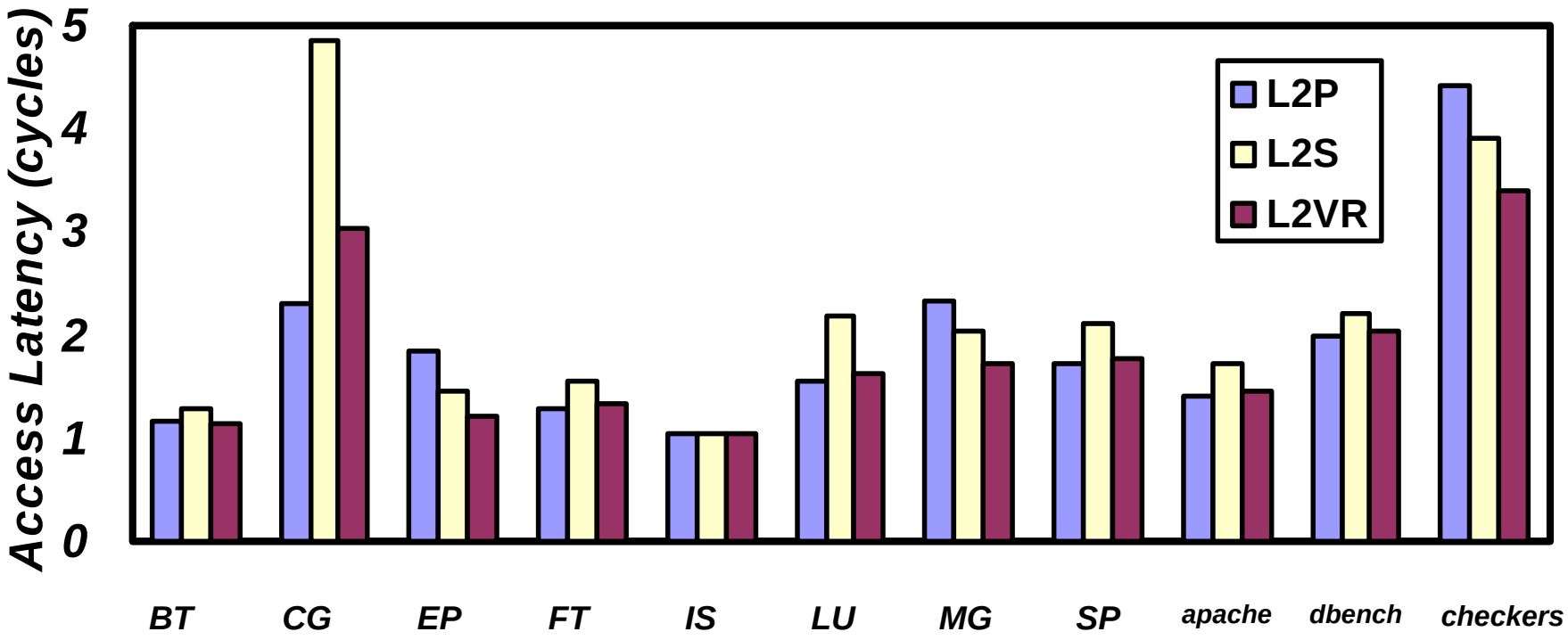
CSAIL

Average Access Latency



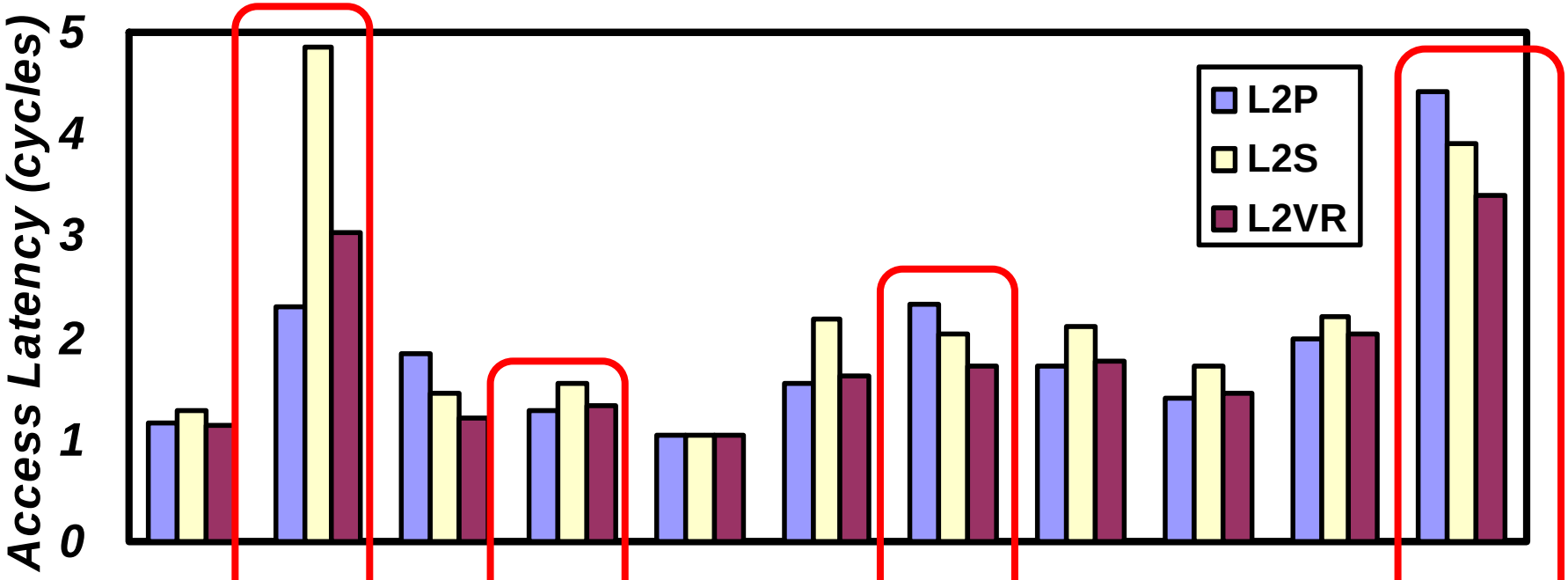
CSAIL

Average Access Latency, with Victim Replication



CSAIL

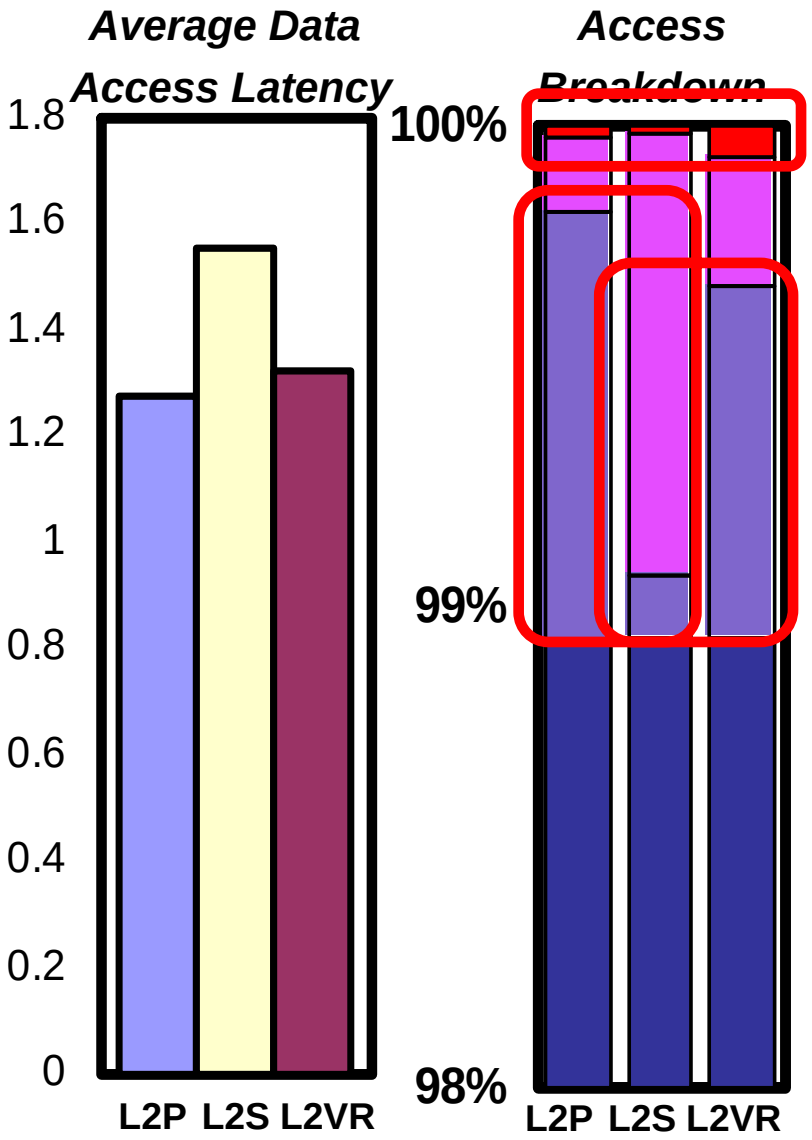
Average Access Latency, with Victim Replication



	BT	CG	EP	FT	IS	LU	MG	SP	apache	dbench	checkers
1st	L2VR	L2P	L2VR	L2P	Tied	L2P	L2VR	L2P	L2P	L2P	L2VR
2nd	L2P 0.1%	L2VR 32.0%	L2S 18.5%	L2VR 3.5%	Tied	L2VR 4.5%	L2S 17.5%	L2VR 2.5%	L2VR 3.6%	L2VR 2.1%	L2S 14.4%
3rd	L2S 12.2%	L2S 111%	L2P 51.6%	L2S 21.5%	Tied	L2S 40.3%	L2P 35.0%	L2S 22.4%	L2S 23.0%	L2S 11.5%	L2P 29.7%

CSAIL

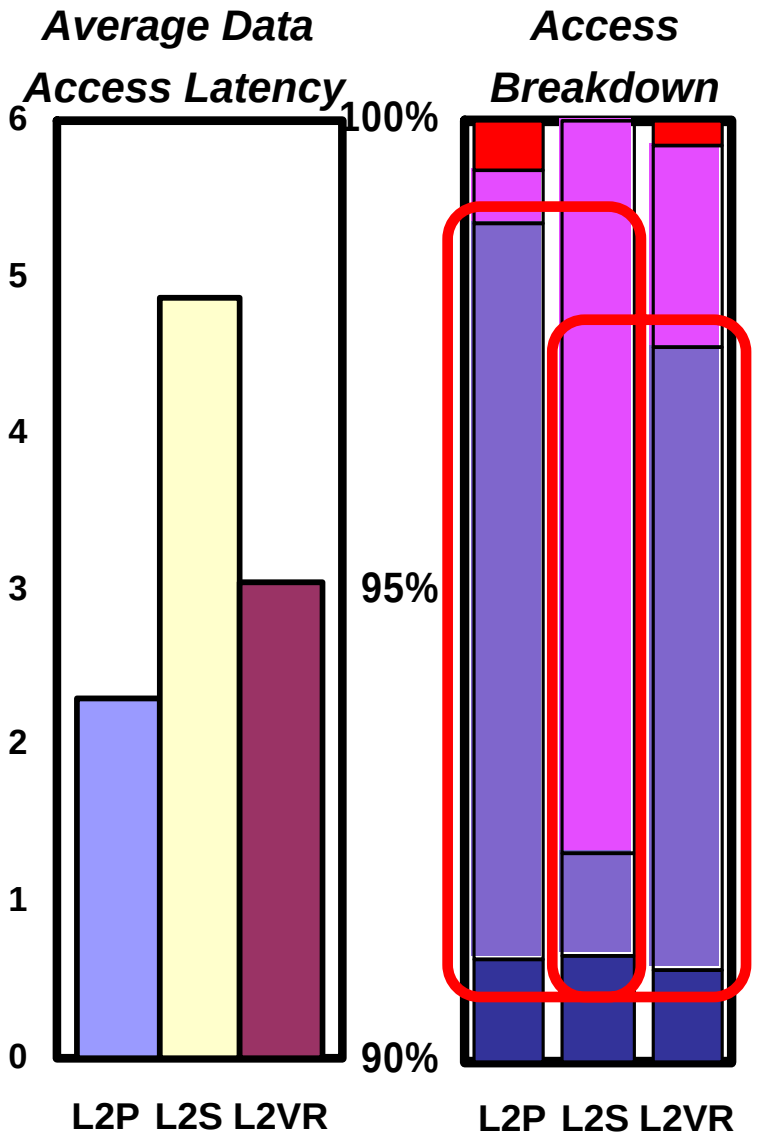
FT: Private Design is the Best When Working Set Fits in Local L2 Slice



- ***The large capacity of the shared design is not utilized as shared and private designs have similar off-chip miss rates***
- ***The short access latency of the private design yields better performance***
- ***Victim replication mimics the private design by creating replicas, with performance within 5%***

CSAIL

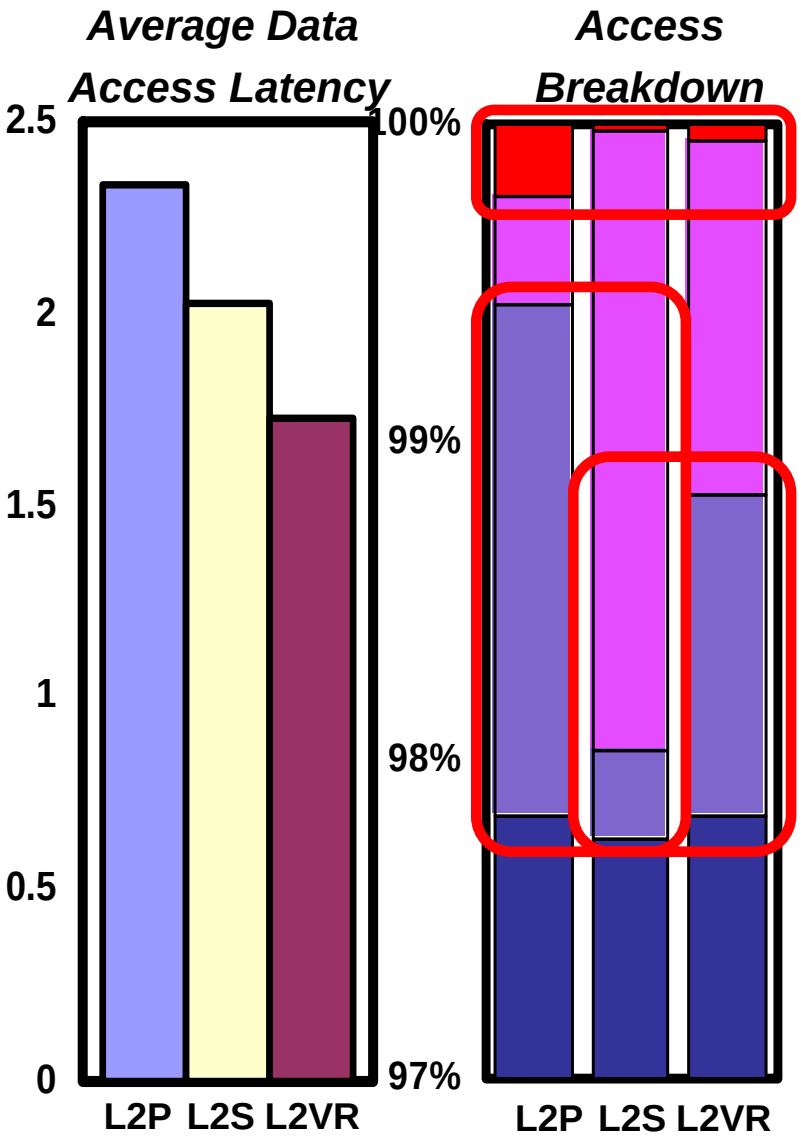
CG: Large Number of L2 Hits Magnifies Latency Advantage of Private Design



- *The latency advantage of the private design is magnified by the large number of L1 misses that hits in L2 (>9%)*
- *Victim replication edges out shared design with replicas, by falls short of the private design*

CSAIL

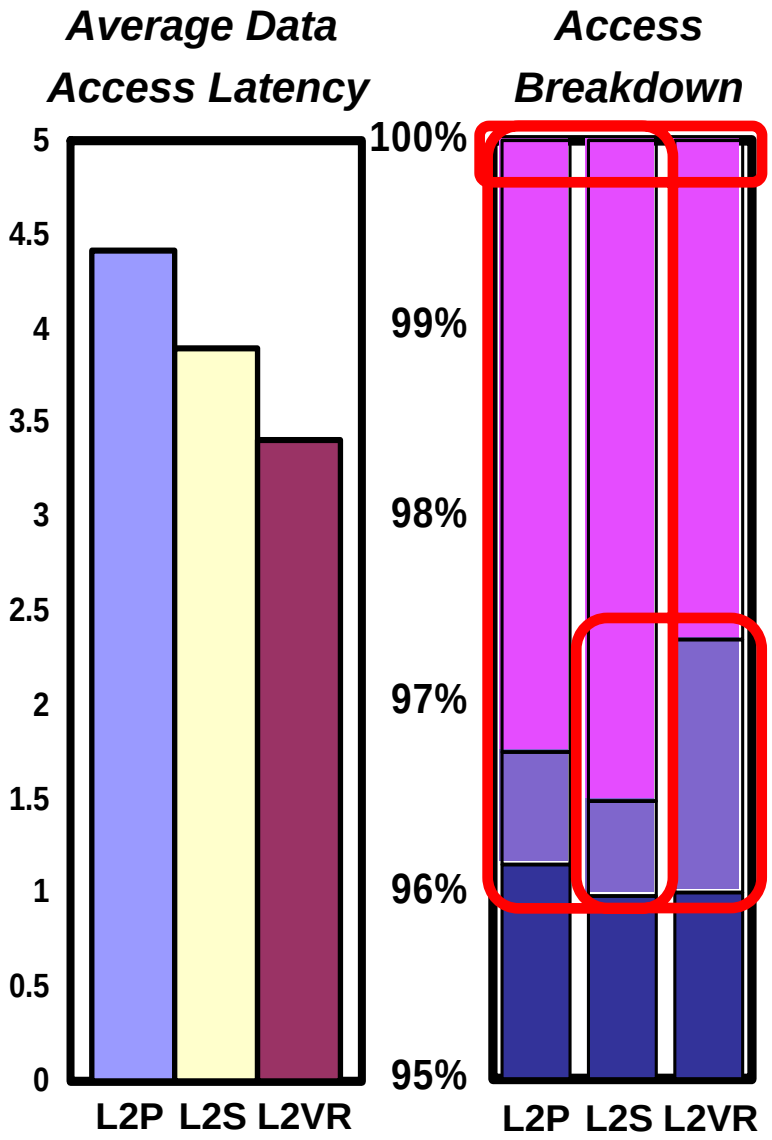
MG: Victim Replication is the Best When Working Set Does not Fit in Local L2



- *The capacity advantage of the shared design yields many fewer off-chip misses*
- *The latency advantage of the private design is offset by costly off-chip accesses*
- *Victim replication is even better than shared design by creating replicas to reduce access latency*

CSAIL

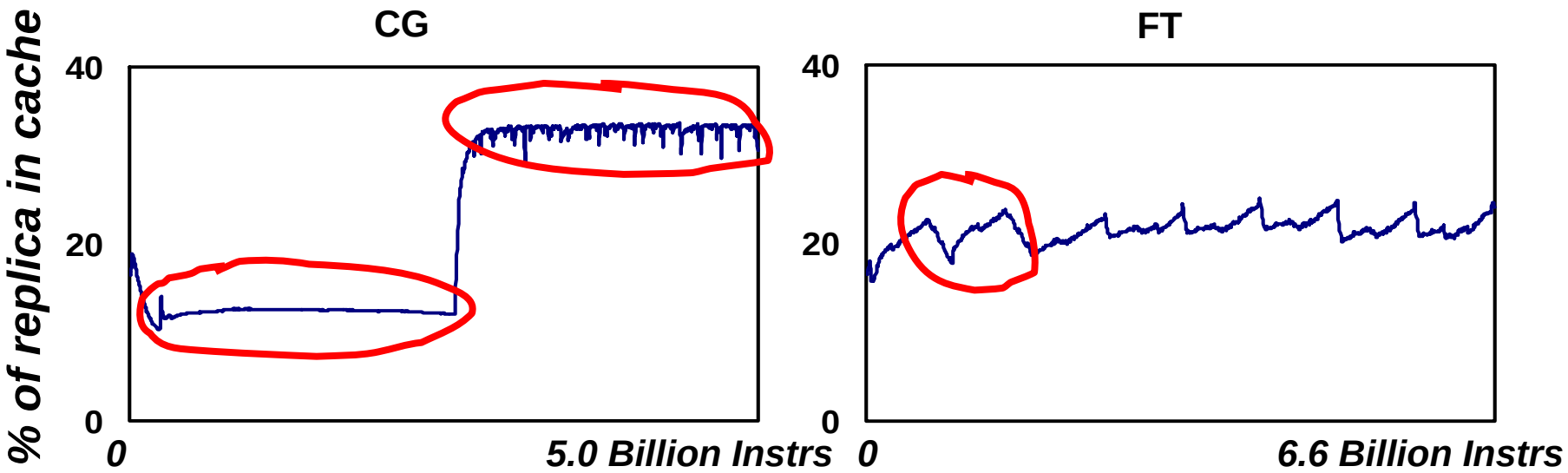
Checkers: Dynamic Thread Migration Creates Many Cache-Cache Transfers



- *Virtually no off-chip accesses*
- *Most of hits in the private design come from more expensive cache-to-cache transfers*
- *Victim replication is even better than shared design by creating replicas to reduce access latency*

CSAIL

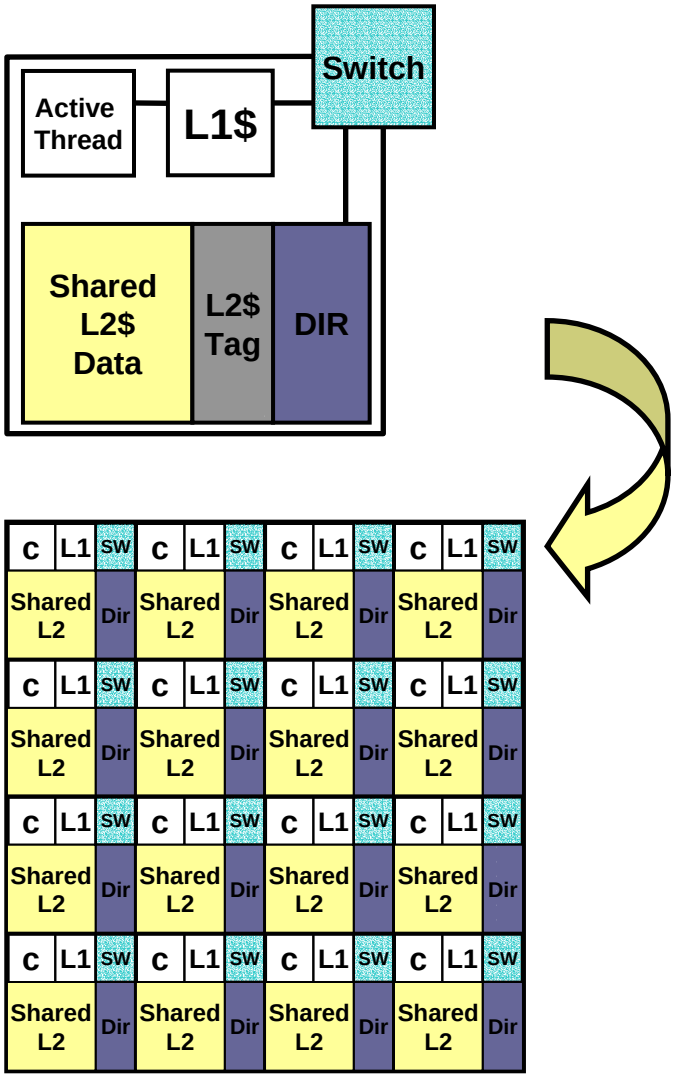
Victim Replication Adapts to the Phases of the Execution



Each graph shows the percentage of replicas in the L2 caches averaged across all 8 caches

CSAIL

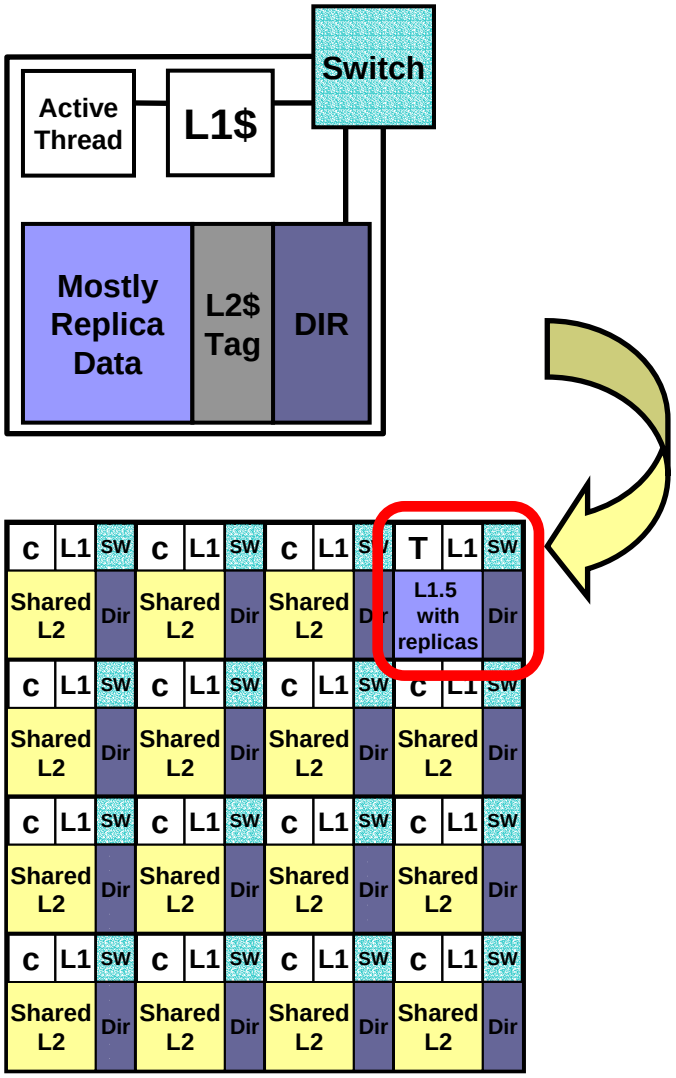
Single-Threaded Benchmarks



- **SpecINT2000 are used as Single-Threaded benchmarks**
  - ***Intel C compiler version 8.0.055***
- **Victim replication automatically turns the cache hierarchy into *three* levels with respect to the node hosting the active thread**

CSAIL

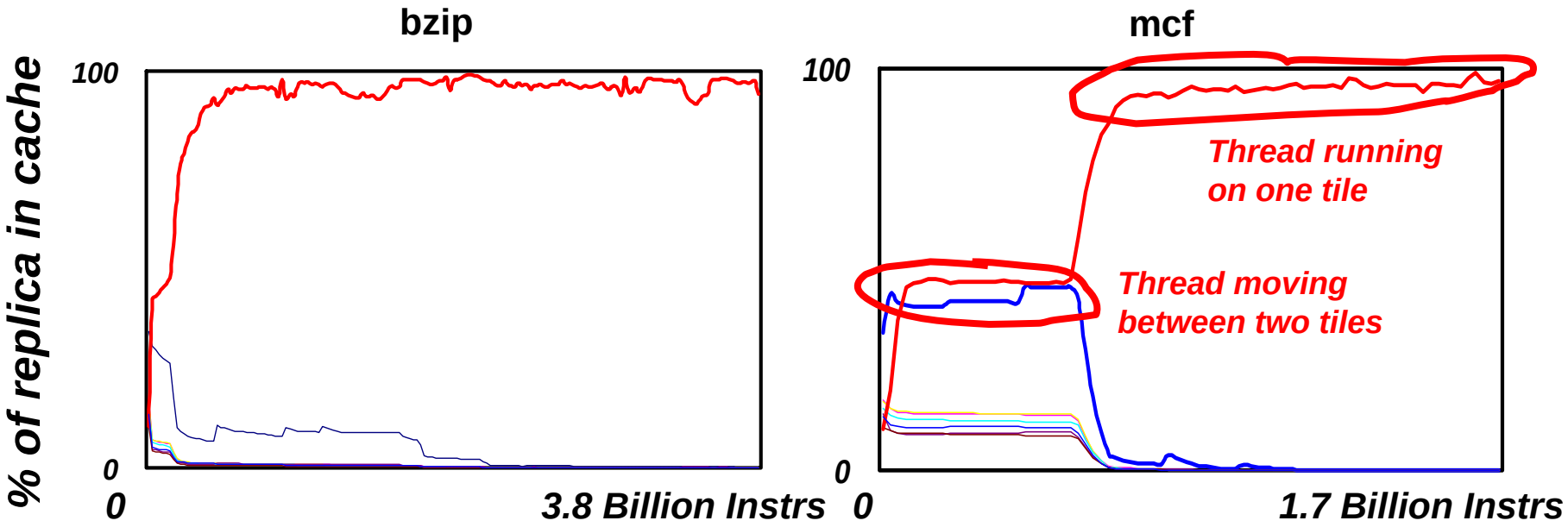
Single-Threaded Benchmarks



- **SpecINT2000 are used as Single-Threaded benchmarks**
  - ***Intel C compiler version 8.0.055***
- **Victim replication automatically turns the cache hierarchy into *three* levels with respect to the node hosting the active thread**
  - *Level 1: L1 cache*
  - *Level 2: All remote L2 slices*
  - ***"Level 1.5":*** *The local L2 slice acts as a large private victim cache which holds data used by the active thread*

CSAIL

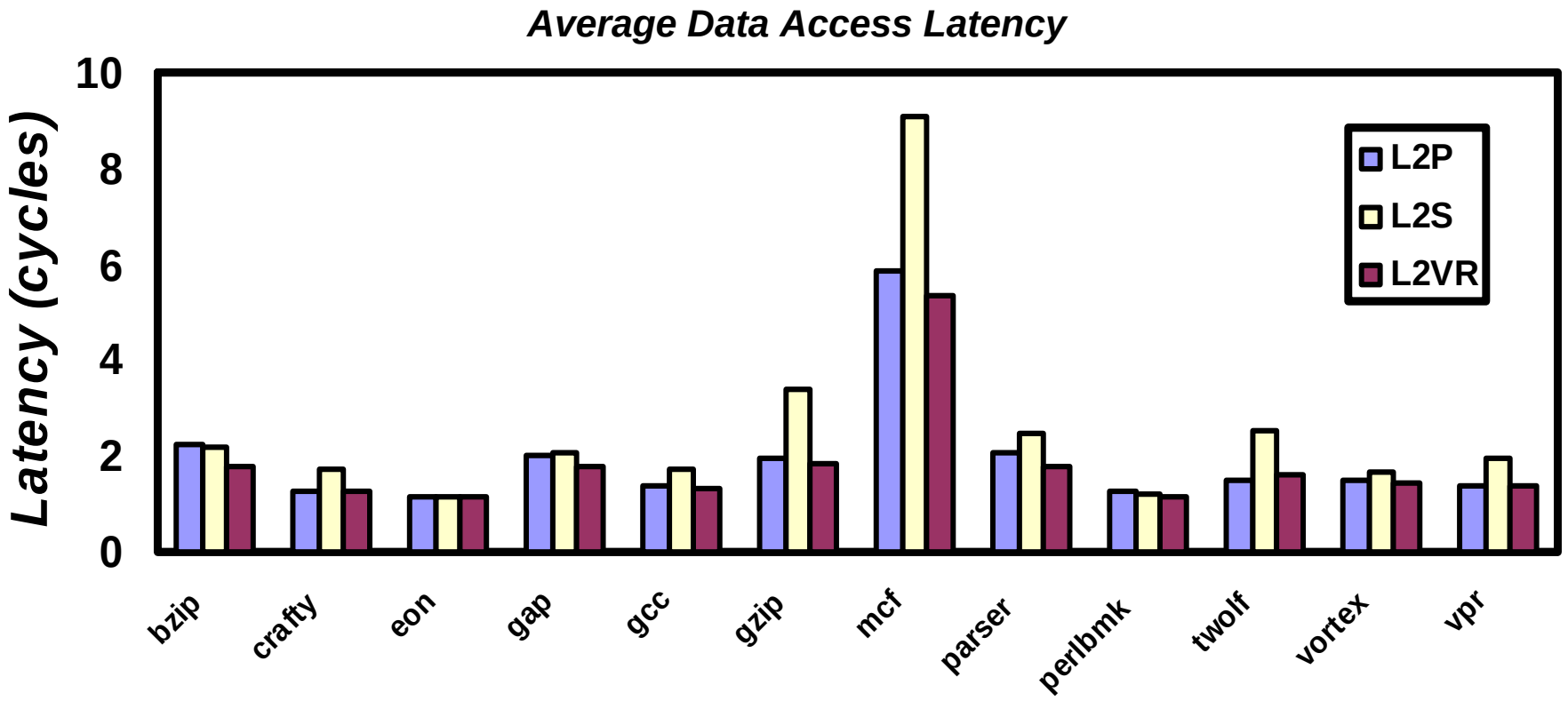
Three Level Caching



Each graph shows the percentage of replicas in the L2 caches for each of the 8 caches

CSAIL

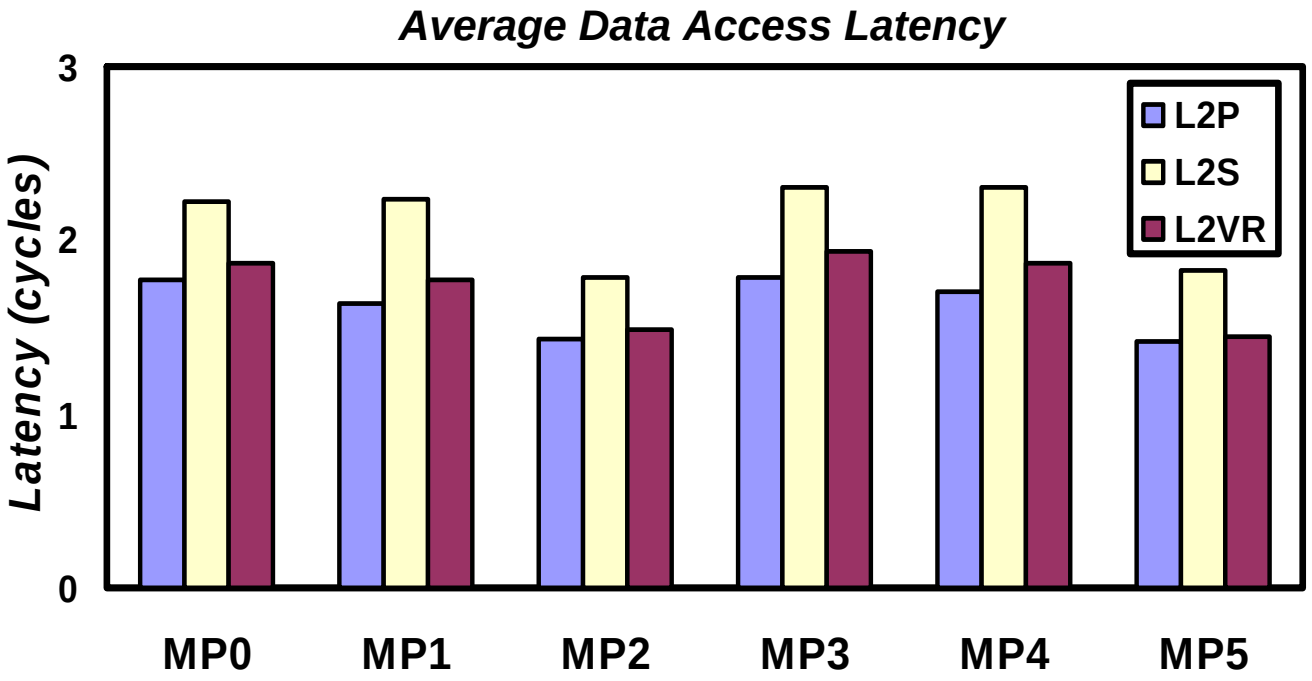
Single-Threaded Benchmarks



Victim replication is the best policy in 11 out of 12 benchmarks with an average saving of 23% over shared design and 6% over private design

CSAIL

Multi-Programmed Workloads



Created using SpecINTs, each with 8 different programs chosen at random

1st : Private design, always the best
2nd : Victim replication, performance within 7% of private design
3rd : Shared design, performance within 27% of private design

CSAIL

Concluding Remarks

Victim Replication is

- ***Simple**: Requires little modification from a shared L2 design*

- ***Scalable**: Scales well to CMPs with large number of nodes by using a directory-based cache coherence protocol*

- ***Robust**: Works well for a wide range of workloads*
  1. *Single-threaded*
  2. *Multi-threaded*
  3. *Multi-programmed*

Thank You!

CSAIL