Stash Directory: A Scalable Directory for Many-Core Coherence

Socrates Demetriades and Sangyeun Cho

University of Pittsburgh

Directory

- Chip multiprocessors with many cores.
- Coherence is needed across private Caches.
- Directory for scalable coherence solutions.
Directory: Energy VS Area

Duplicate-Tags [e.g., Piranha, NiagraT2]

Sparse [e.g., AMD Opteron]

L2 Cache
Core 0

L2 Cache
Core 1

L2 Cache
Core 2

L2 Cache
Core N
Directory: Energy VS Area

Duplicate-Tags [e.g., Piranha, NiagraT2]
Energy inefficient

L2 Tags
Core 0  Core 1  Core 2  Core N

Sparse [e.g., AMD Opteron]
Ways (L2 assoc. \times N-Cores)
Directory: Energy VS Area

Duplicate-Tags

Energy inefficient

Sparse [e.g., AMD Opteron]

Area Inefficient

How big is enough?

2x-4x over-provision

L2 Tags

Core 0

Core 1

Core 2

Core N
Sparse-based Directories

- Conventional Sparse (2-4x)
  [Gupta:isca90, Conway:micro10]
- Clever hashing (1.5x)
  [Ferdman:hpca’11, Sanchez:hpca’12]
- Course-grain set indexing
  [Alisafaee:micro12]
- Disabling coherence for private pages
  [Cuesta:isca11]
- Stash Directory

How big is enough?
Stash Directory

• Stash is allowed to not track all cached tags.
  – Entries that track private blocks can be silently removed from directory.
  – LLC and the coherence protocol are involved to discover unregistered blocks.

• Contribution:
  ✓ Power Efficient (low associatively)
  ✓ Space Efficient (as small as 0.25x provisioning size without performance impact)
  ✓ Transparent (no OS support, simple design).
  ✓ Scalable (largely independent to core count).
Outline

- Introduction
- Directory-Induced Invalidations
  - Stash Directory
  - Evaluation
- Conclusion
Directory-Induced Invalidations

Insert Dir Entry

Eviction

Directory Forces Invalidation

L2 Miss

Core 0  Core 1  Core 2  Core N
Conflict Rate \(\rightarrow\) Cache Miss Rate

[Benchmark: fluidanimte]
Forcing Invalidation on Private Blocks

Insert Dir Entry

Eviction

HOT Private Block

Core 0  Core 1  Core 2  Core N
(1) Invalidation of Hot Blocks
(2) Causing (unnecessary) Additional Misses
(3) ‘Polluting’ the Directory Set
Forcing Invalidation on Private Blocks

- PARSEC and SPLASH2 Workloads
- 1/4x Directory Size Provisioning

On average:
- **72%** of directory-induced invalidations target Private blocks
- **80%** of invalidated blocks will be re-loaded, causing misses.
Outline

- Introduction
- Directory-Induced Invalidations
- Stash Directory
- Evaluation
- Conclusion
Stash Directory: Overview

- Directory knows if an entry is tracking a private block.
- If evicted entry is private, then do not enforce invalidation.
- Private blocks remain hidden from the directory.
- LLC and the coherence protocol are involved to discover hidden blocks if necessary.
Stash Directory: Silent Eviction

Mark as Stash-hidden Block

Dir Eviction

Do not enforce invalidation
Stash Directory: Handling False Misses
Stash Directory: Handling False Misses

Shared LLC

Core 0

L2 Miss

Core 1

Found

Core 2

Core N
Outline

- Introduction
- Directory-Induced Invalidations
- Stash Directory
- Evaluation
- Conclusion
Evaluation Methodology

• **Workloads**
  - Multithreaded benchmarks from SPLAS2 and PARSEC.2.1.

• **Trace**
  - x86 traces generated using PIN.
  - Feed into cache/NoC cycle detailed model
  - 1-IPC in-order core model.

• **Simulated Machine Configuration**
  - 16-core tiled based CMP.
  - Distributed shared LLC 16MB,
  - L1/L2 private caches, inclusive. 4-way/8-way
  - 4x4 mesh NoC.
  - Distributed directory (same associativity as L2). Varying Size.
Comparison Schemes.

1. **Sparse**: Conventional Sparse Directory.

2. **PDC**: Deactivating Coherence for Private blocks [Cuesta:isca11].
   - course-grain classification of blocks into private/shared (page granularity).
   - If miss on a private block, do not invoke coherence protocol.
     => private blocks are not tracked by the directory.
   - Recover mechanism when page goes from private to shared.
   - OS-supported technique.

- All schemes use the same sharer-vector encoding.
- All schemes use the same associatively (same as L2).
Cache Size VS Miss Rate

![Graph showing the relationship between cache size and miss rate for different directory provisioning ratios. The graph includes lines for 'Sparse', 'PDC', and 'Stash'.]
Cache Size VS Miss Rate

![Graph showing the relationship between cache size and miss rate.

The graph plots % Miss Rate Change against Directory Provisioning Ratio for different cache sizes: 2x, 1x, 1/2x, 1/4x, 1/8x, and 1/16x. The x-axis represents the directory provisioning ratio, and the y-axis represents the % Miss Rate Change.

Legend:
- Blue: Sparse
- Red: PDC
- Green: Stash

The graph indicates that as the directory provisioning ratio decreases, the % Miss Rate Change increases significantly for all cache sizes, with Sparse showing the most pronounced increase at 1/16x.]
Cache Size VS Miss Rate

canneal

% Miss Rate Change

Directory Provisioning Ratio

Sparse
PDC
Stash
Cache Performance

- Miss Rate (Normalized)

2x, 1x, 1/2x, 1/4x
Cache Performance

For 1/4x Directory Size, improve execution time by 16% on average.
Similar in performance to Sparse-2x, while being 8 times smaller.
False misses are few (<6% of directory misses).
Scalability

**Area:**
Can Stash remain small? (1/4x)

**Bandwidth:**
Can Stash remain bandwidth efficient?
Conclusion

• Stash inherits the power efficiency of spares directories.

• Reduces the directory size requirements significantly.

• Provides a transparent optimization, independent of system software, core type and count.

• Leverages a shared, on-chip last level cache.
Thank you for your attention!