SCD: A Scalable Coherence Directory with Flexible Sharer Set Encoding

Daniel Sanchez and Christos Kozyrakis
Stanford University

HPCA-18, February 27th 2012

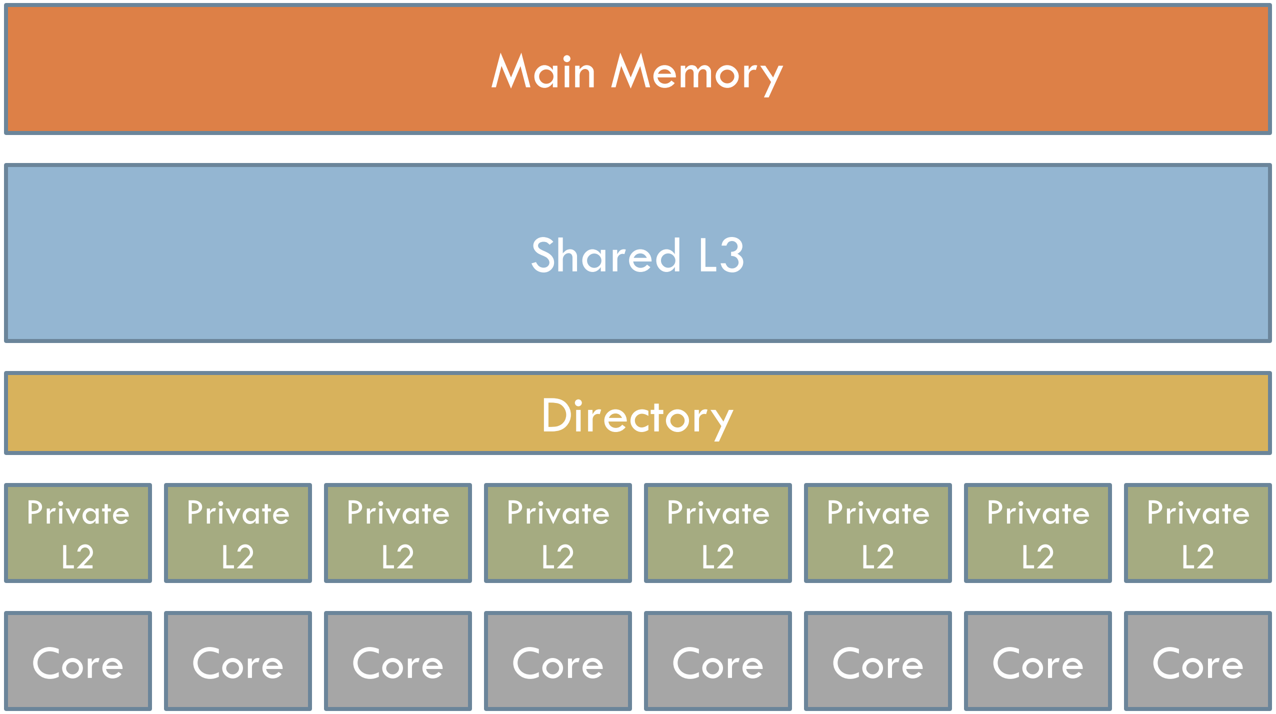
Executive Summary

- Directories are hard to scale, degrade performance
- SCD: A scalable directory with performance guarantees
  - Flexible sharer set encoding: Lines with few sharers use one entry, widely shared lines use multiple entries → Scalability
  - Use ZCache → Efficient high associativity, analytical models
    - Negligible invalidations with minimal overprovisioning (~10%)
  - At 1024 cores, SCD is 13x smaller than a sparse directory, and 2x smaller, faster, simpler than a hierarchical directory

Outline

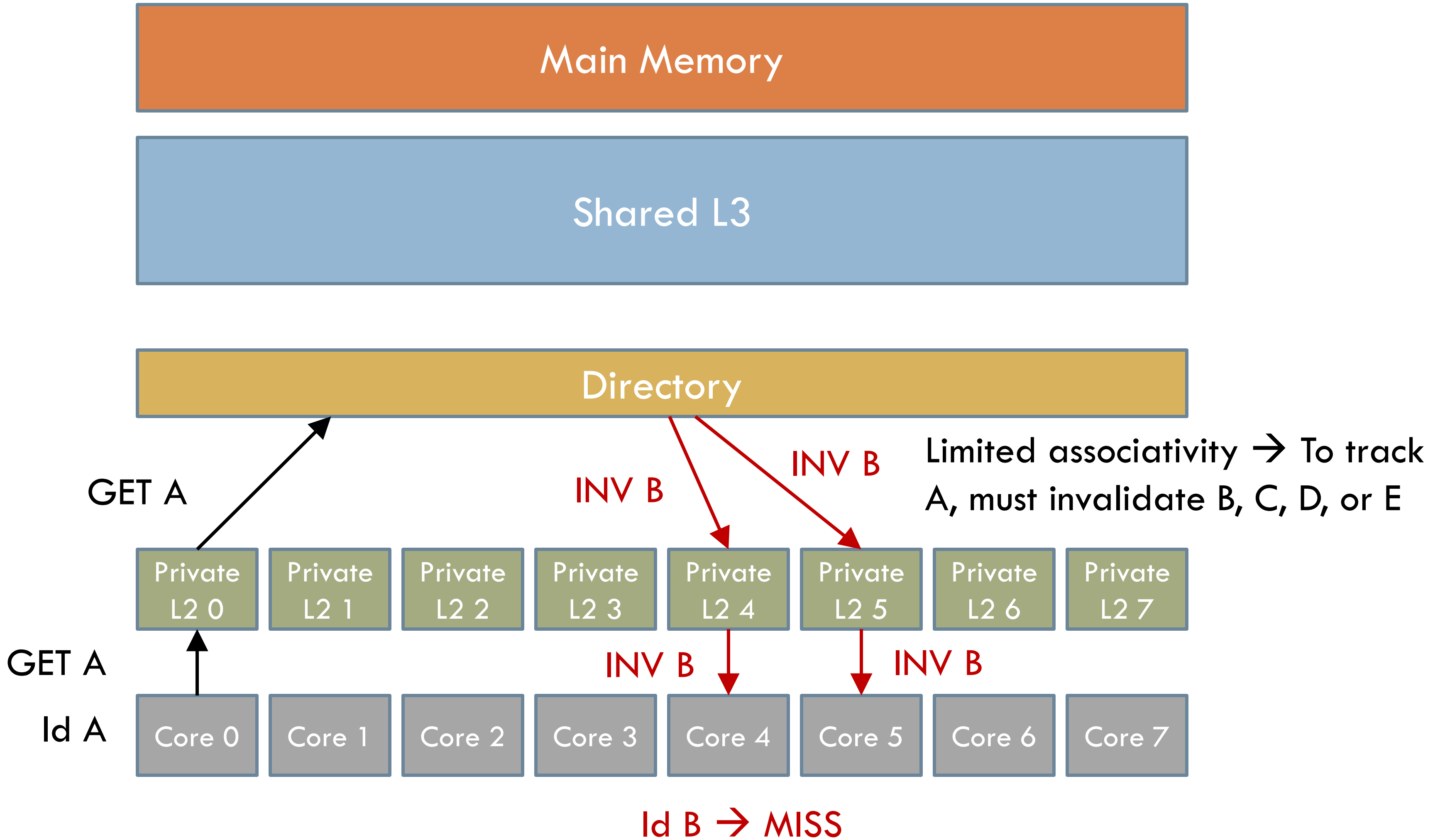
- Introduction
- SCD Design
- Analytical Bounds on Overprovisioning
- Evaluation

Directory-Based Coherence



- Scalable coherence protocols use a directory
  - Tracks contents of private caches
  - Ordering point for conflicting requests

Directory-Induced Invalidations

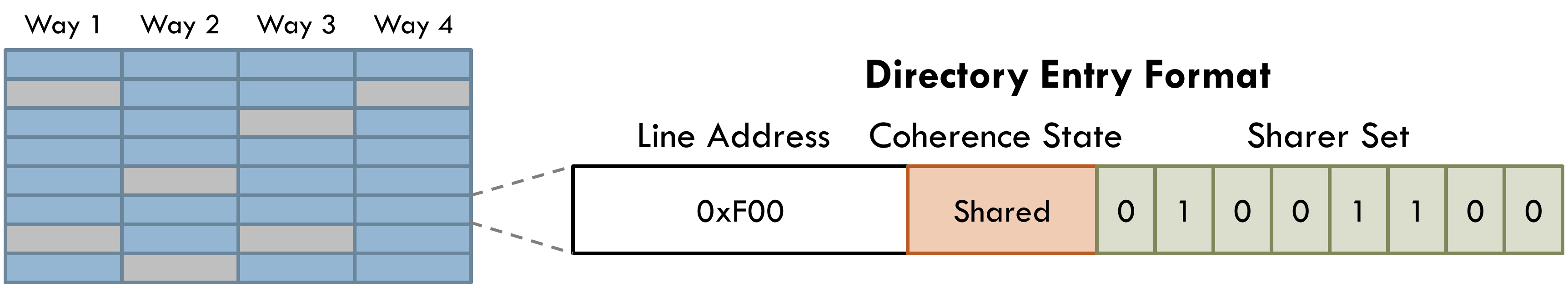


Desirable Directory Properties

1. Scalability
   - Latency, energy, area
   - Constant or log(cores) growth
2. Minimal complexity
   - No changes to coherence protocol
3. Exact sharer information
4. Negligible directory-induced invalidations
   - With minimal, bounded overprovisioning

Sparse Full-Map Directories

- Associative array indexed by address
- Sharer sets encoded in a bit-vector



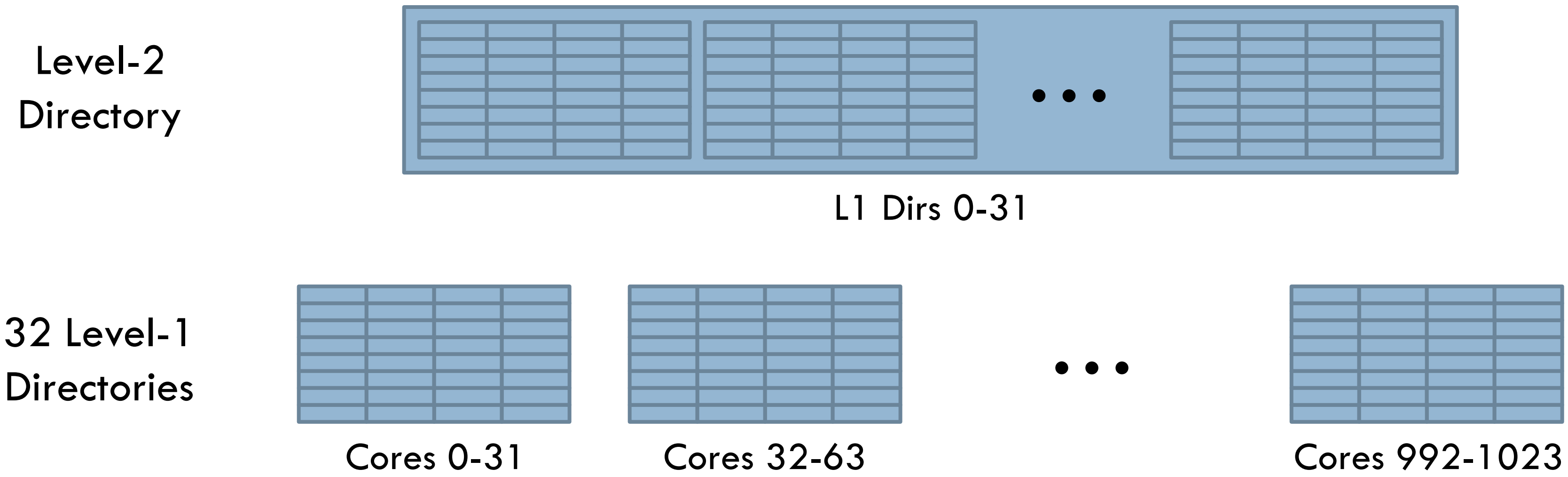
✓ Single lookup → Low latency, energy-efficient

✗ Bit-vectors grow with # cores → Area scales poorly

✗ Limited associativity → Directory-induced invalidations, overprovisioning (~2x)

Hierarchical Sparse Directories

- Multi-level hierarchy of sparse directories



✓ Small bit-vectors → Scalable area & energy

✘ Multiple lookups in critical path → Additional latency

✘ Needs hierarchical coherence protocol → More complexity

✘ Directory-induced invalidations more expensive

Single-Level Dirs with Inexact Sharer Sets

- Coarse-grain bit-vectors (e.g., 1 bit for every 4 cores)
- Limited pointers: Maintain a few sharer pointers, invalidate or broadcast on overflow
- Tagless [MICRO 09]: Encode sharers with Bloom filters
- SPACE [PACT 10]: De-duplicate sharing patterns

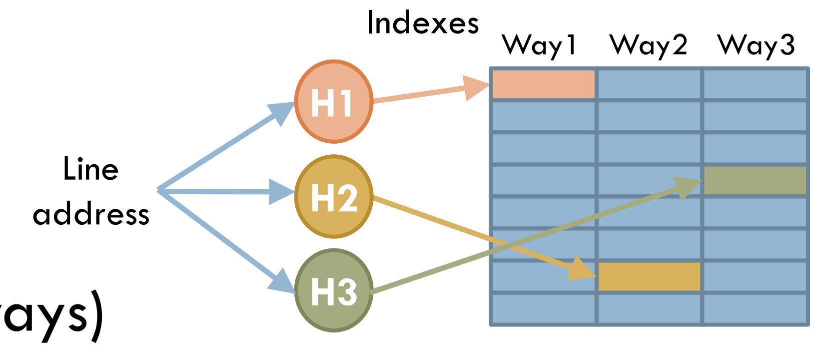
✓ Reduced area & energy overheads

✗ Overheads still not scalable

✗ Inexact sharers → Broadcasts, invalidations or spurious lookups

Efficient Highly-Associative Caches

- ZCache [MICRO 10]: High-associativity cache with few ways
  - Draws from skew-associativity and Cuckoo hashing
  - Hits take a single lookup
  - In a miss, replacement process provides many candidates
  - Provides cheap high associativity (e.g., 64-way associativity with 4 ways)
  - Described by simple & accurate analytical models



- Cuckoo Directory [Ferdman et al., HPCA 11]:
  - Apply Cuckoo hashing to sparse directories
  - Empirically show that smaller overprovisioning (~25%) eliminates most invalidations

Outline

- Introduction
- SCD Design
- Analytical Bounds on Overprovisioning
- Evaluation

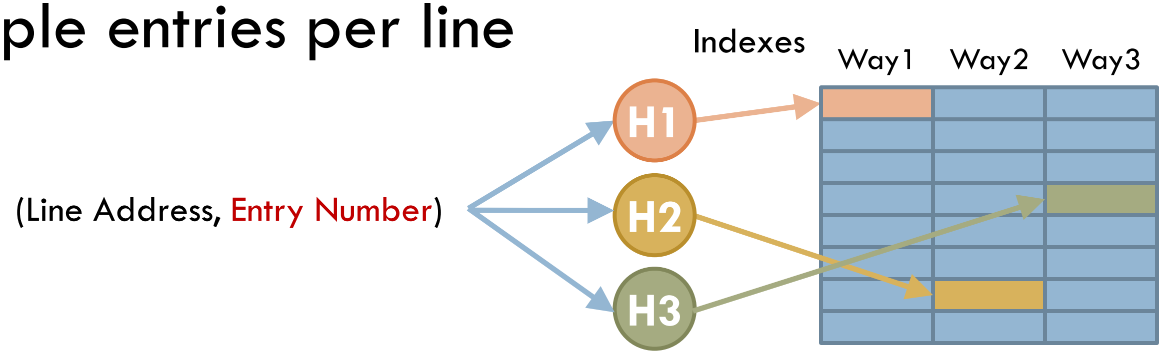
Scalable Coherence Directory: Insights

- Use ZCache
  - Cheap high associativity
  - Analytical models → Bounds on overprovisioning
    - Negligible difference with ideal directory regardless of workload
    - Validated in simulation

- Provision space per tracked sharer, not line
  - Flexible sharer set encoding: Lines with few sharers use a single entry, widely shared lines use additional entries

SCD Array

- ZCache array indexed by (Line Address, Entry Number)
  - Allows multiple entries per line



- Insertions walk array until an unused entry is found, or a limit of candidates (R) is reached, then invalidate one
  - Could use a replacement policy to decide victim
  - Evictions are negligible → no need for replacement policy

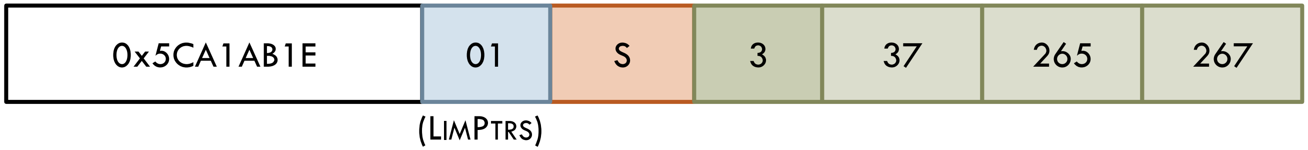
SCD Entry Formats

- Example: 1024 sharers

Line Address (44b)	Type (2b)	37b		
INVALID	0 0	Unused (37b)		
LIMITED POINTERS	0 1	Coherence State (5b)	#ptrs (2b)	3x 10-bit sharer pointers (30b)
ROOT BIT-VECTOR	1 0	Coherence State (5b)	Root bit-vector (32b)	
LEAF BIT-VECTOR	1 1	Leaf number (5b)	Leaf bit-vector (32b)	

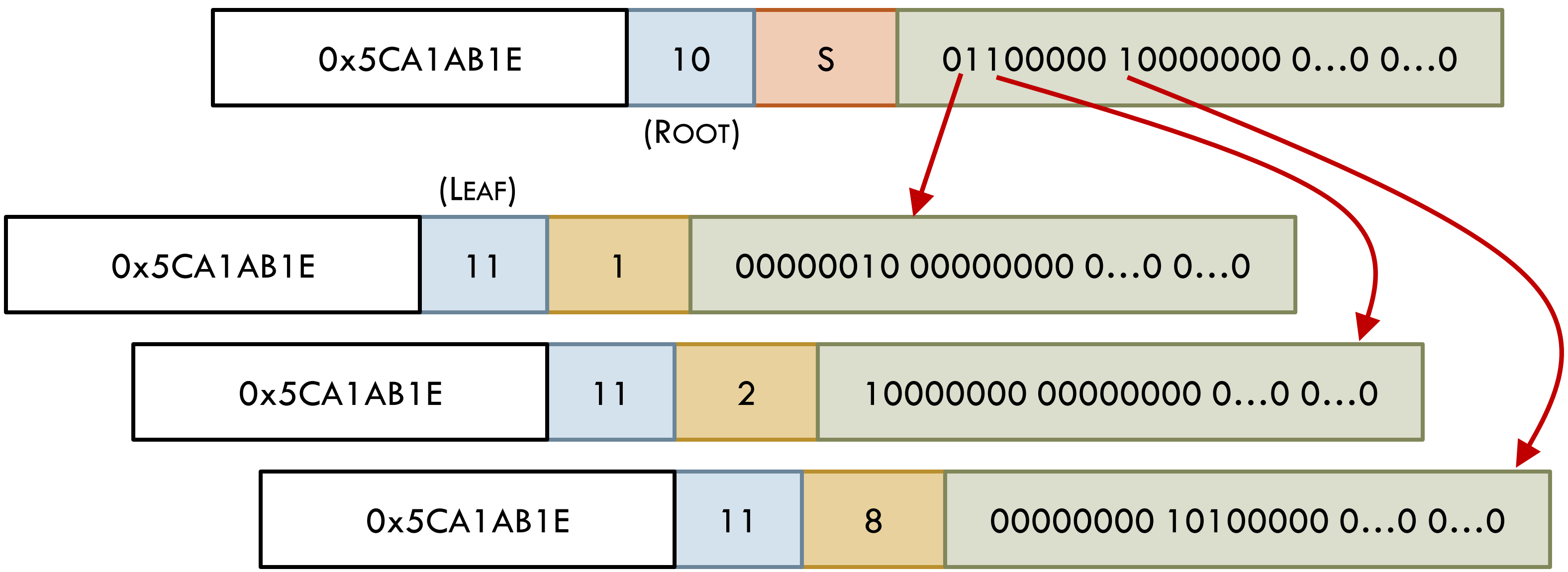
- Lines with one or few sharers use a limited pointer entry
- Lines with >3 sharers use root + leaves bit-vector entries

Example: Adding a Sharer



Add sharer *64* to address 0x5CA1AB1E :

1. Lookup (0x5CA1AB1E, 0), all pointers are used → switch to multi-entry format
2. Allocate entries (0x5CA1AB1E, leafNum+1) with leafNum=1,2,8
3. Write leaf bit-vectors
4. Write (0x5CA1AB1E, 0) as a root bit-vector



SCD & Desirable Properties

1. Scalability
   - Flexible sharer set encoding → Scalable energy and area
   - Coherence state stored in a single entry → Most operations have 1 lookup on critical path → Scalable latency

2. Minimal complexity
   - All entries in the same array → No coherence protocol changes

3. Exact sharer information

4. Negligible directory-induced invalidations
   - With minimal, bounded overprovisioning

Outline

- Introduction
- SCD Design
- Analytical Bounds on Overprovisioning
- Evaluation

Analytical Models

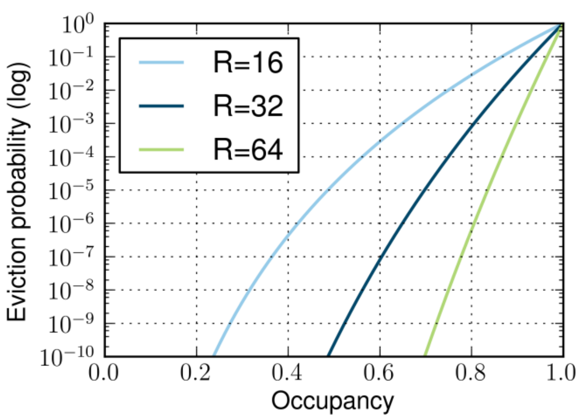
- Directories built with ZCache arrays can be characterized with simple, workload-independent analytical models

W	Ways
R	Replacement candidates
occ	Occupancy (fraction of used entries)

Fraction of insertions that cause a directory invalidation

Determines performance impact, interference

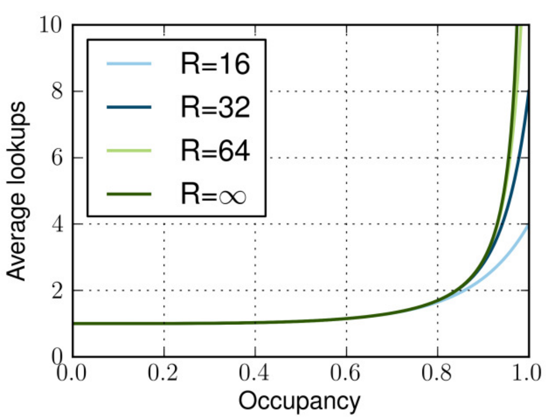
$$P_{inv} = occ^{R}$$



Average lookups per replacement

Determines replacement latency and energy

$$AvgLookups = \frac{1 - occ^{R}}{1 - occ^{W}}$$



Bounding Invalidations

- SCD bounds invalidations with minimal overprovisioning
  - Bounded worst-case behavior independent of workload
  - For Pinv=10^{-3} → W=4, R=64, 11% overprovisioning
    - Max directory occupancy 90%

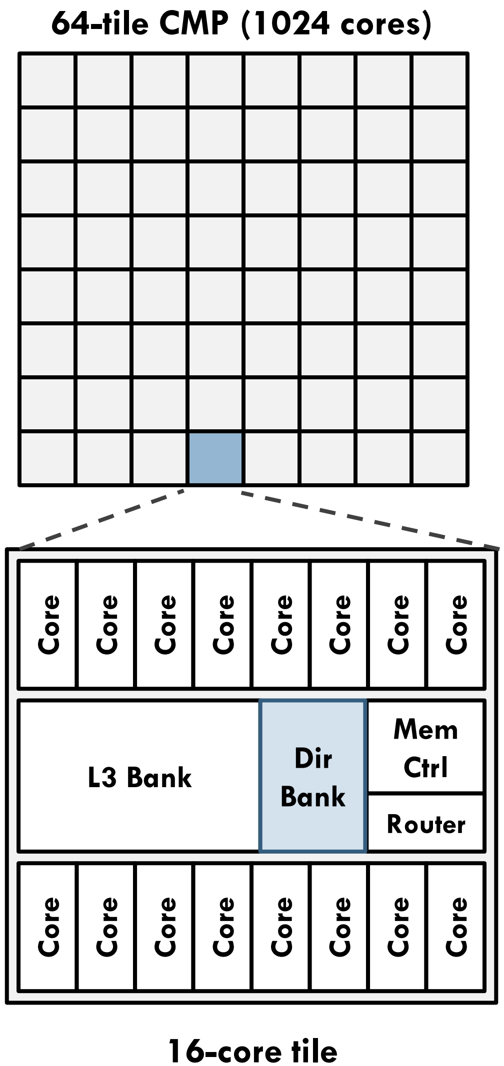
- Overprovisioning is:
  - Smaller than previous empirical results (25%-2x)
  - Bounded → Strict guarantees, no design-time uncertainty

Outline

- Introduction
- SCD Design
- Analytical Bounds on Overprovisioning
- Evaluation

Methodology

- Simulated system: 1024-core tiled CMP
  - In-order cores with split L1s
  - Private inclusive L2s, 128KB/core
  - Shared non-inclusive L3, 256MB
  - MESI directory protocol
- Directory implementations:
  - Sparse, 2-level Hierarchical, SCD
  - Directories 100%-provisioned for L2s
  - All directories use ZCache arrays → negligible invalidations
- 14 workloads from PARSEC, SPLASH2, SPECOMP/JBB, BioParallel suites

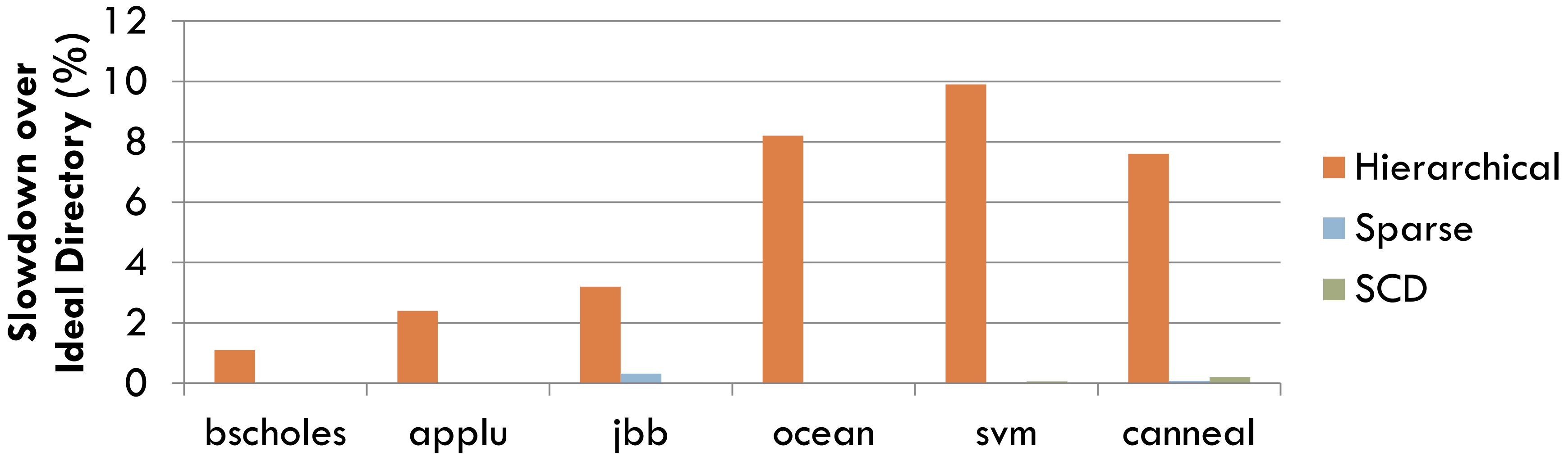


Area

Cores	Sparse	Hierarchical	SCD	Sparse/SCD	Hier/SCD
128	34.2%	21.1%	10.9%	3.12x	1.93x
256	59.2%	24.2%	12.5%	4.73x	1.94x
512	109.2%	27.0%	13.9%	7.87x	1.95x
1024	209.2%	30.9%	15.8%	13.22x	1.95x

- Area given as a percentage of L2 caches
- At 1024 cores, SCD is:
  - 13x smaller than Sparse
  - 2x smaller than Hierarchical
  - Takes ~3% of total die area

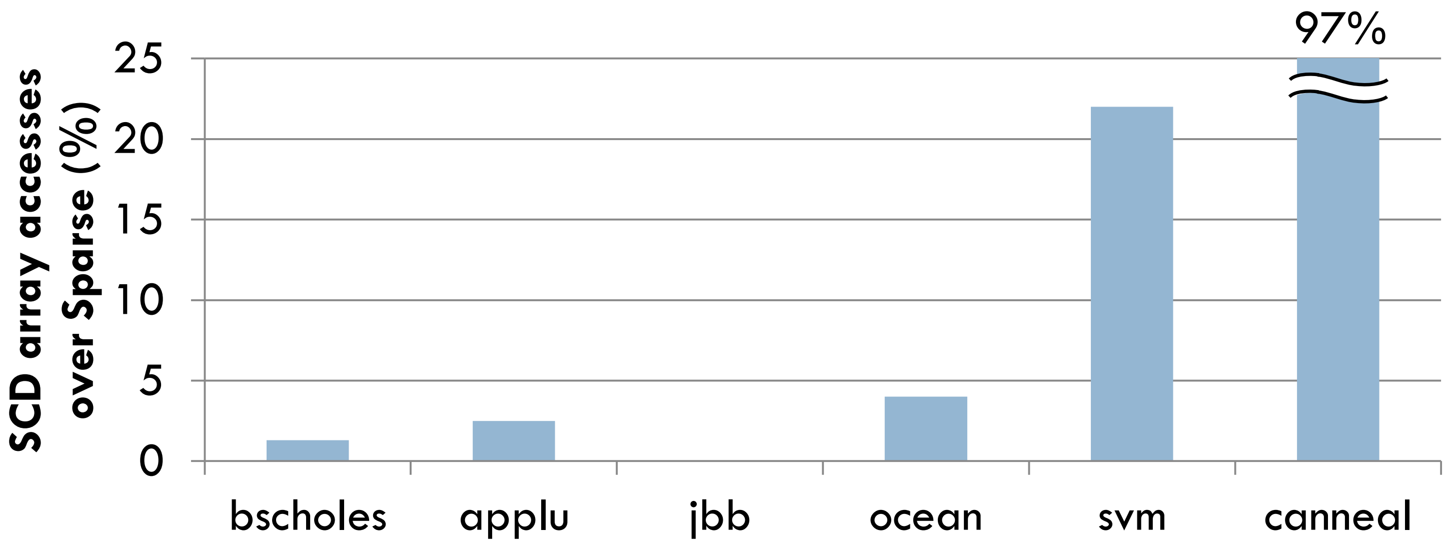
Performance



- Hierarchical up to 10% slower than Ideal
- Sparse has Ideal-like performance, but too expensive
- SCD as fast as Ideal & Sparse, cheapest

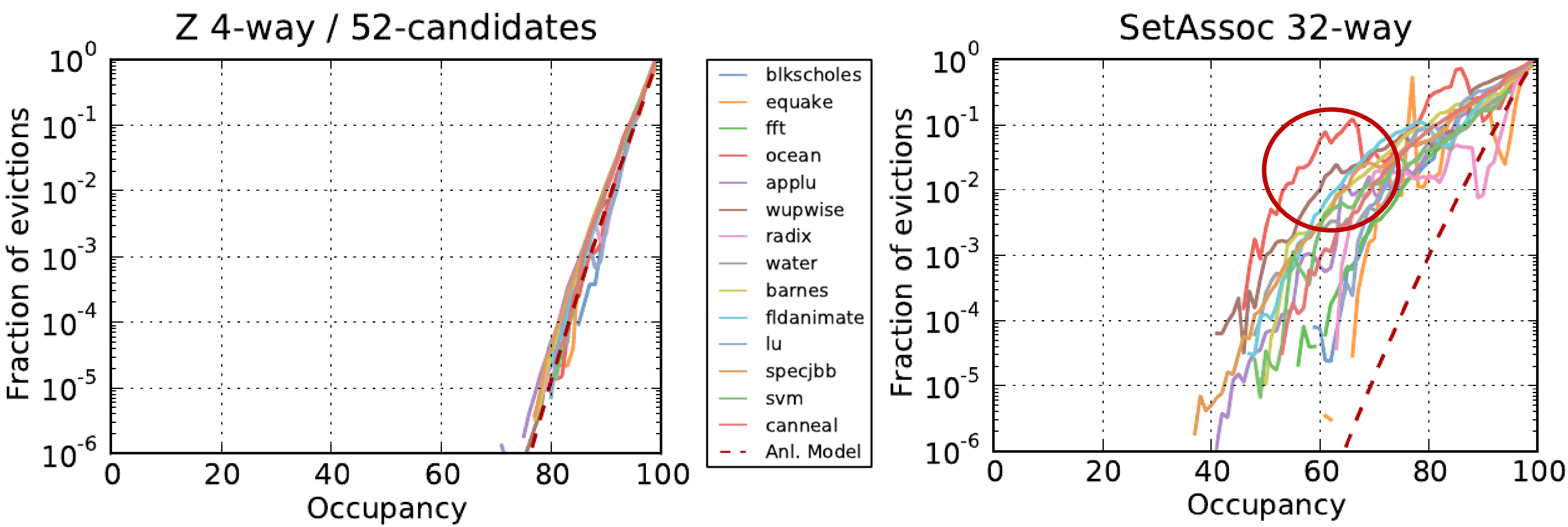
Energy Efficiency

- Directory energy = Accesses * Energy/access



- SCD performs slightly more accesses (lookups, writes) than Sparse
  - Some operations require multiple lookups
  - SCD has higher occupancy, replacements take longer
- SCD energy/access is smaller (narrow entries)

Analytical Models



- Empirical results on invalidations match analytical models
  - Bounds worst-case invalidations with minimal overprovisioning
  - Can provision directory using simple formulas
- Set-associative arrays do not meet analytical models
  - Need significant overprovisioning (~2x), no bounds
  - Similar results for Sparse & Hierarchical

Conclusions

- SCD insights:
  - Use a variable number of entries/line → Keep entries small
  - Use ZCache → High associativity + Analytical models
- SCD = Scalability + Performance guarantees
  - Scalable area, energy, latency
  - Simple: No modifications to coherence protocol
  - Negligible invalidations with bounded overprovisioning
  - At 1024 cores, SCD is 13x smaller than Sparse, and 2x smaller, faster and simpler than Hierarchical

THANK YOU FOR YOUR ATTENTION

QUESTIONS?