Fine-grain Memory Deduplication for In-memory Database Systems

Heiner Litz, David Cheriton, Pete Stevenson

Stanford University

Memory Capacity Challenge

- In-memory databases
  - Limited by memory capacity
- Technology
  - DRAM scaling
  - Pin limitation
  - Power
- Datasets grow > Moore's Law

Memory represents increasing portion of system cost

Our Approach: HICAMP

Fine grain memory deduplication:

1. Search memory for duplicates
2. Replace copy with Reference
3. Virtually increase capacity

Reduces cost & power, increases performance

Potential Savings

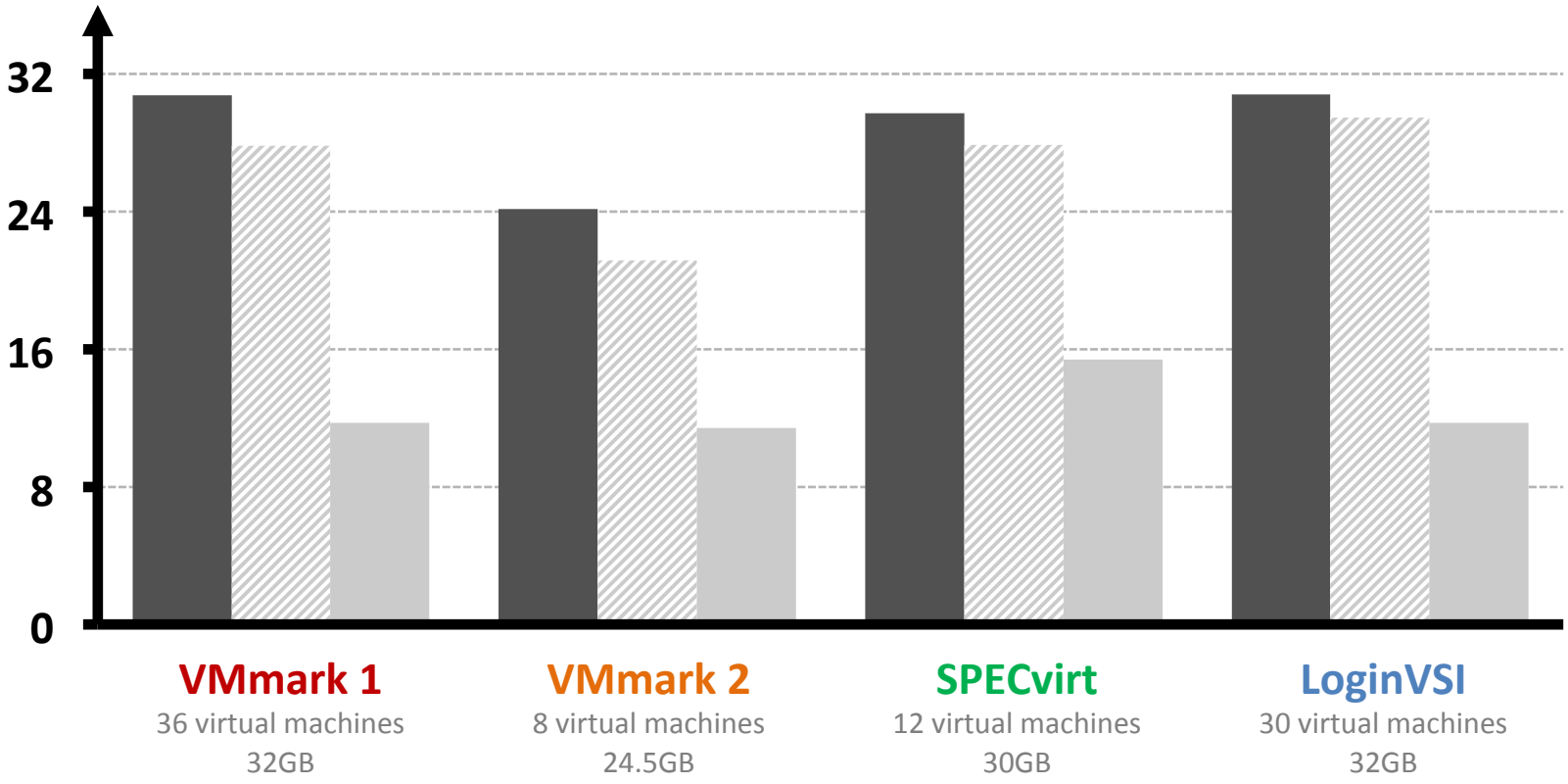
Company	Workload	Dataset	Memory	Raw Dedup	+Overhead
Lightminer	Benchmark	TPCH	256 GB	1.76x	1.54x
SAP	SAP-Hana One	Private	1024 GB	1.94x	1.68x
LinkedIn	Profile Page	Professional Profiles	48 GB	2.74x	2.28x
Quantifind	Data Mining	Social web data	64 GB	2.91x	2.40x
Arista	Build server	Private	128 GB	3.10x	2.53x
UC Berkeley	Shark	Conviva Server	68 GB	3.19x	2.59x
NHN	Memcached	Private	8 GB	3.74x	2.95x
Yelp	Hadoop (EMR)	Private	7 GB	3.79x	2.99x

2x-3x Deduplication worth doing!

Page sharing versus Dedup

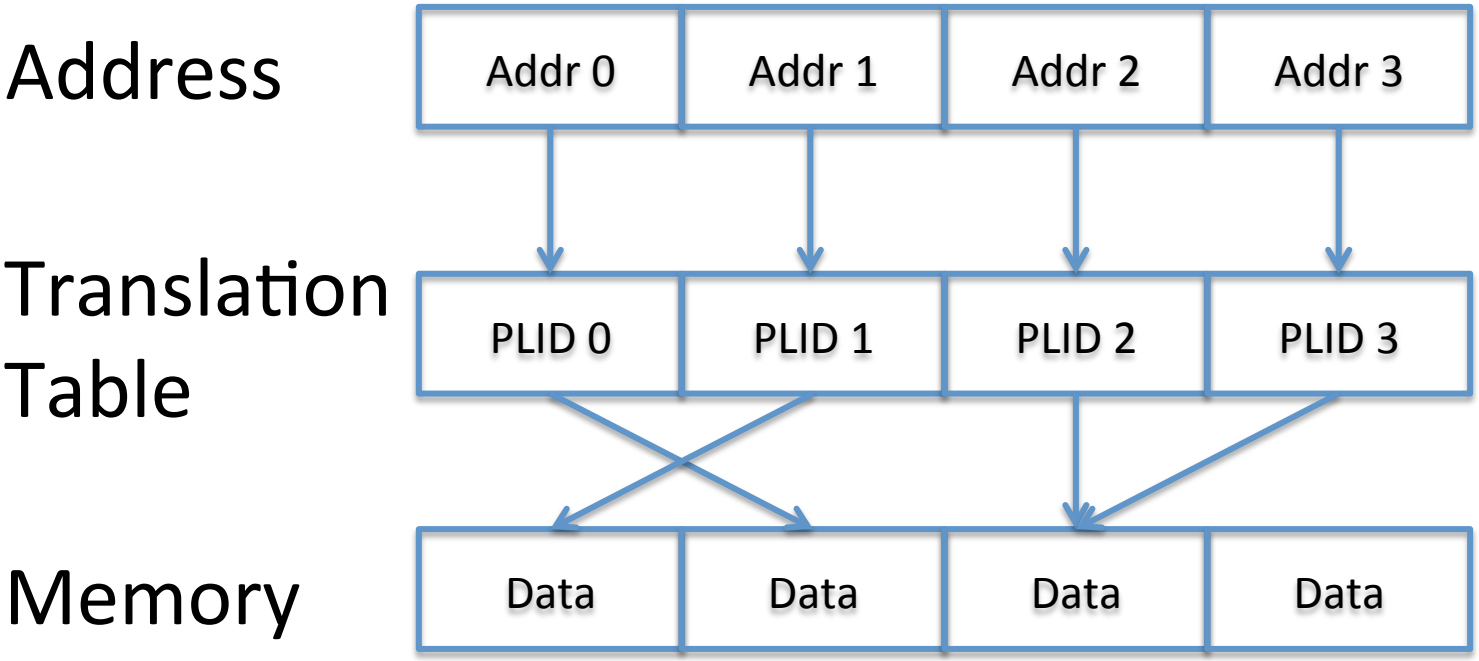
memory usage

(GB)

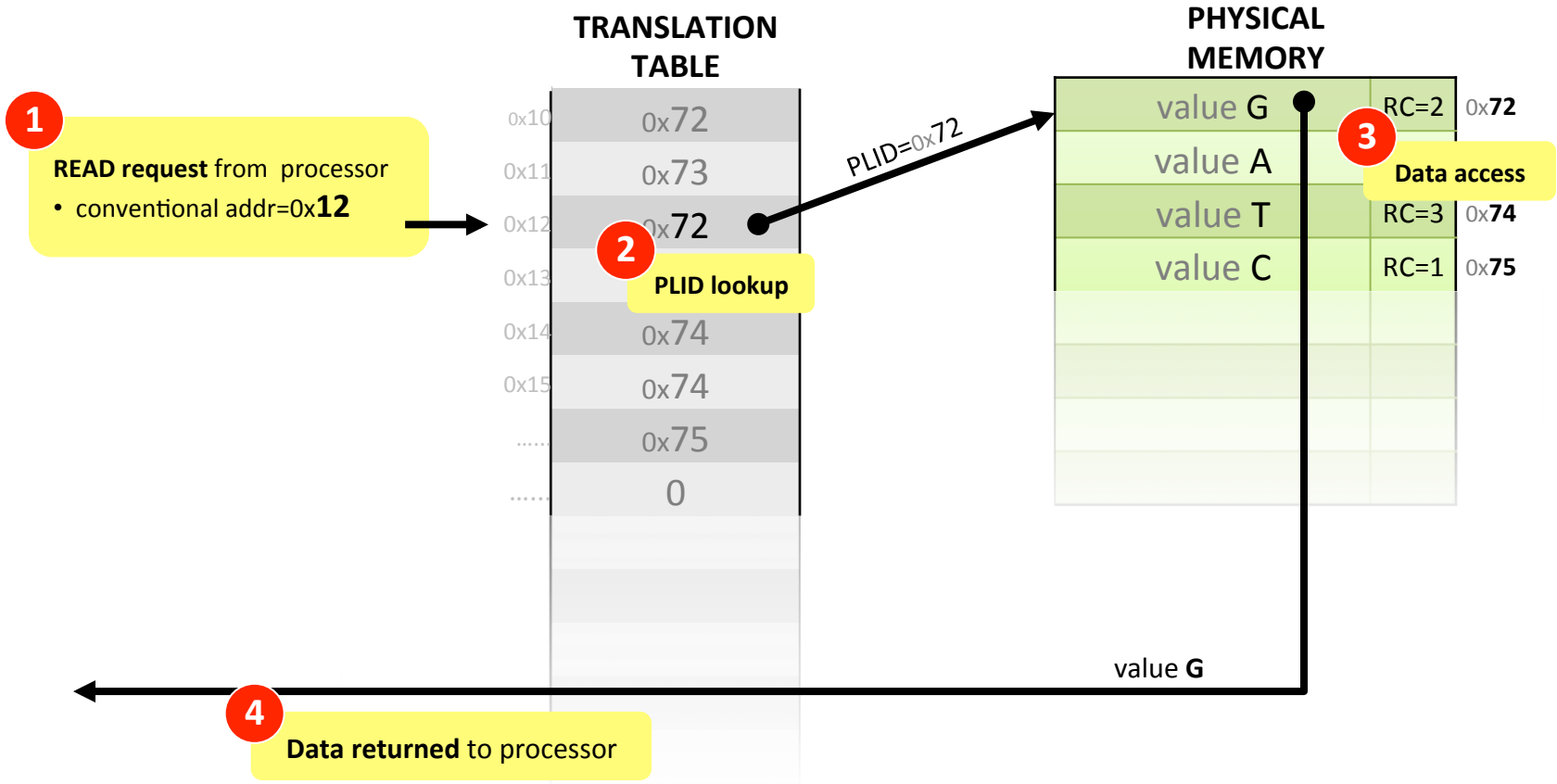


Architecture

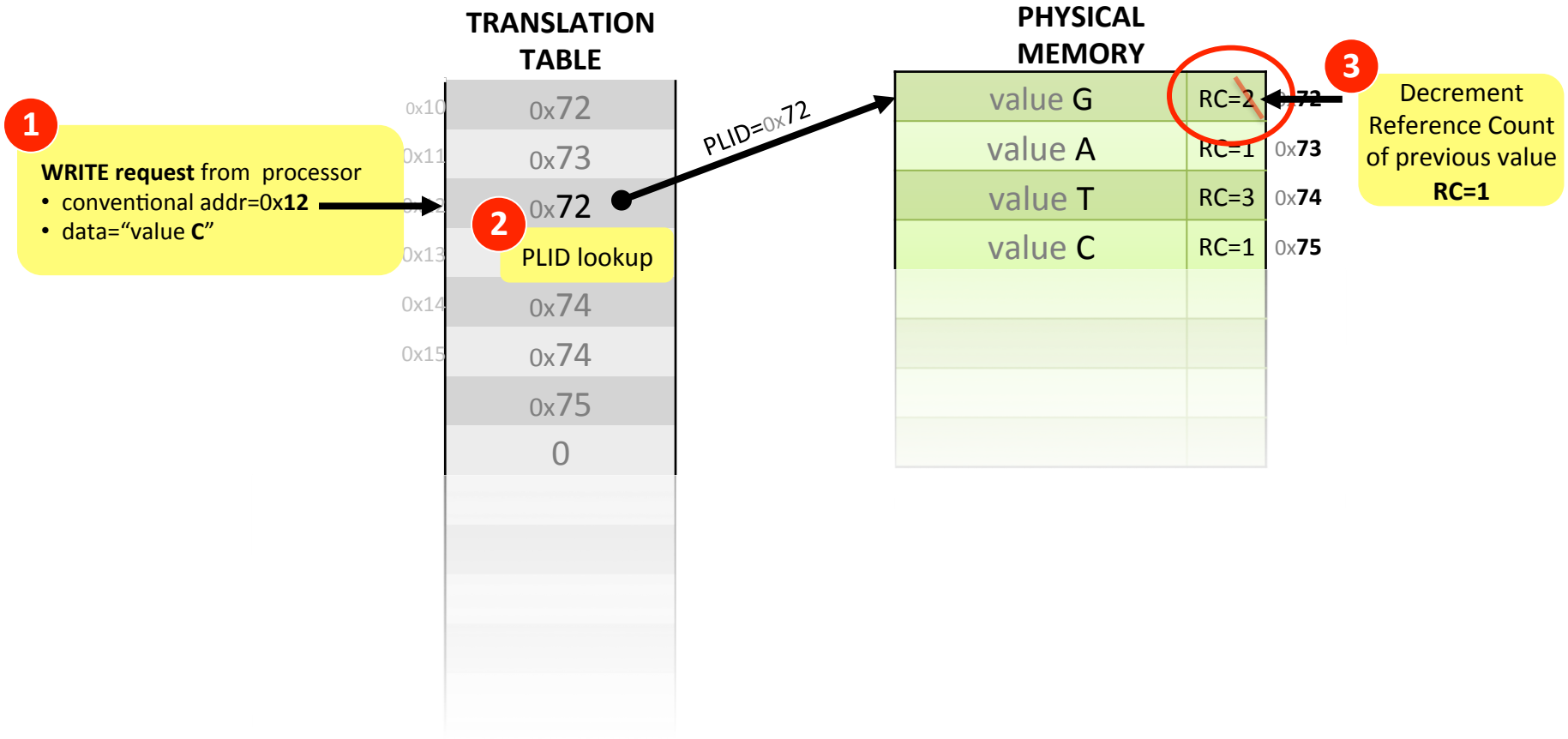
- New Indirection Layer



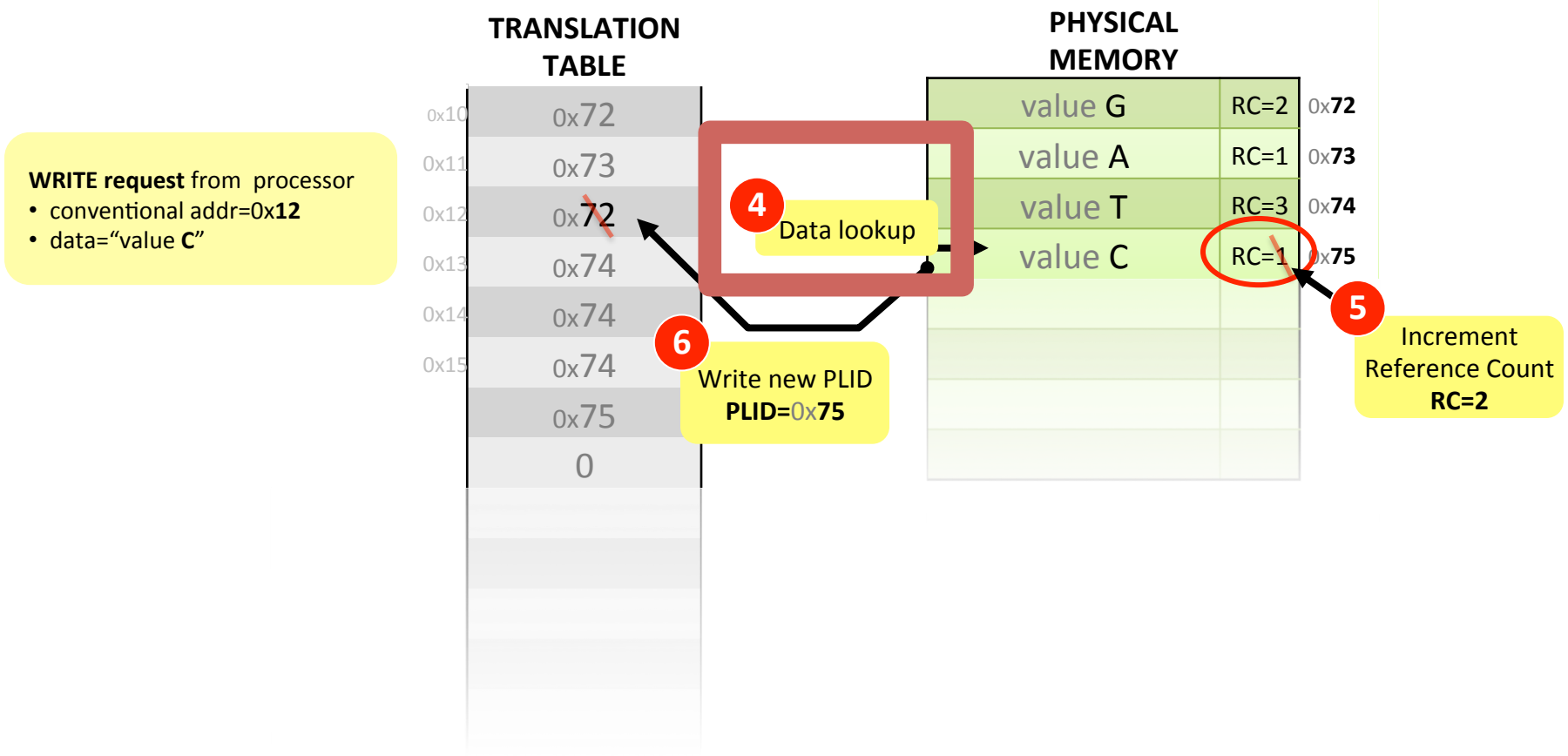
HICAMP Read


TRANSLATION TABLE
PHYSICAL MEMORY
1
READ request from processor
• conventional addr=0x12
0x10
0x11
0x12
0x13
0x14
0x15
0x72
0x73
0x72
0x74
0x74
0x75
0
PLID=0x72
2
PLID lookup
value G
value A
value T
value C
RC=2
RC=3
RC=1
0x72
0x74
0x75
3
Data access
value G
4
Data returned to processor

HICAMP Write 1/2



HICAMP Write 2/2



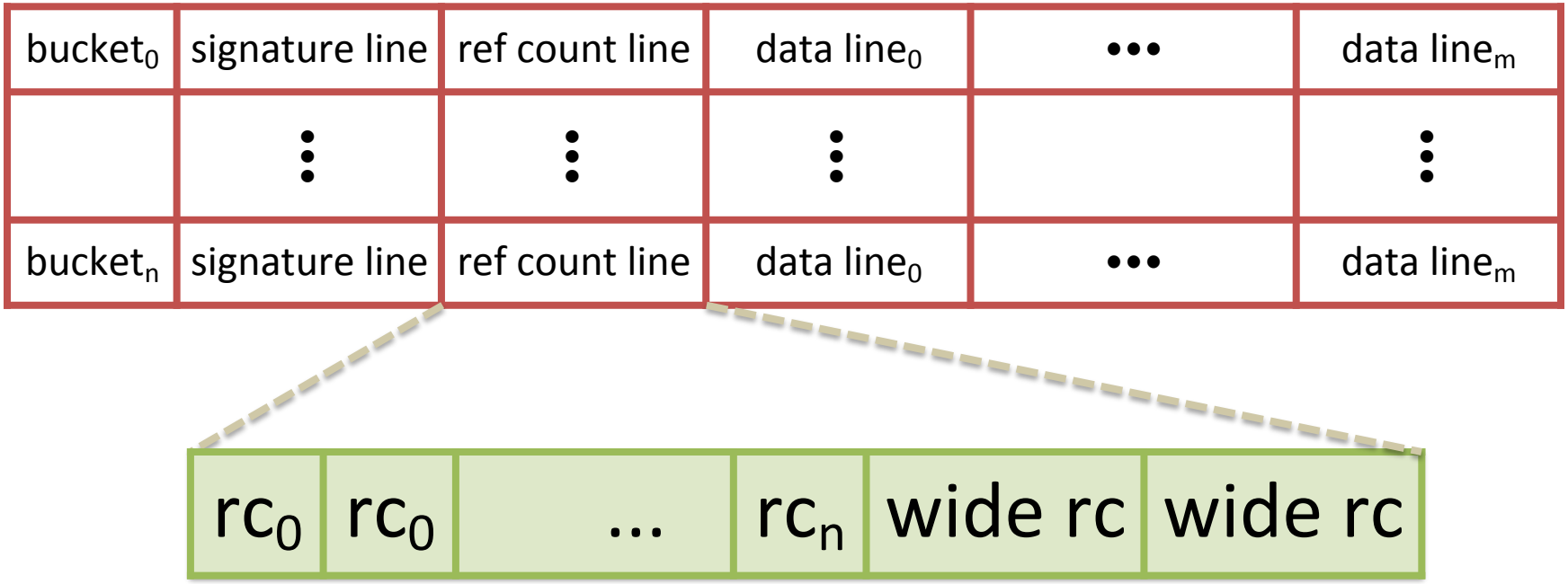
How to perform efficient data lookup?

Global Content Search

Only 2 Mem Ops!

Reference Counting

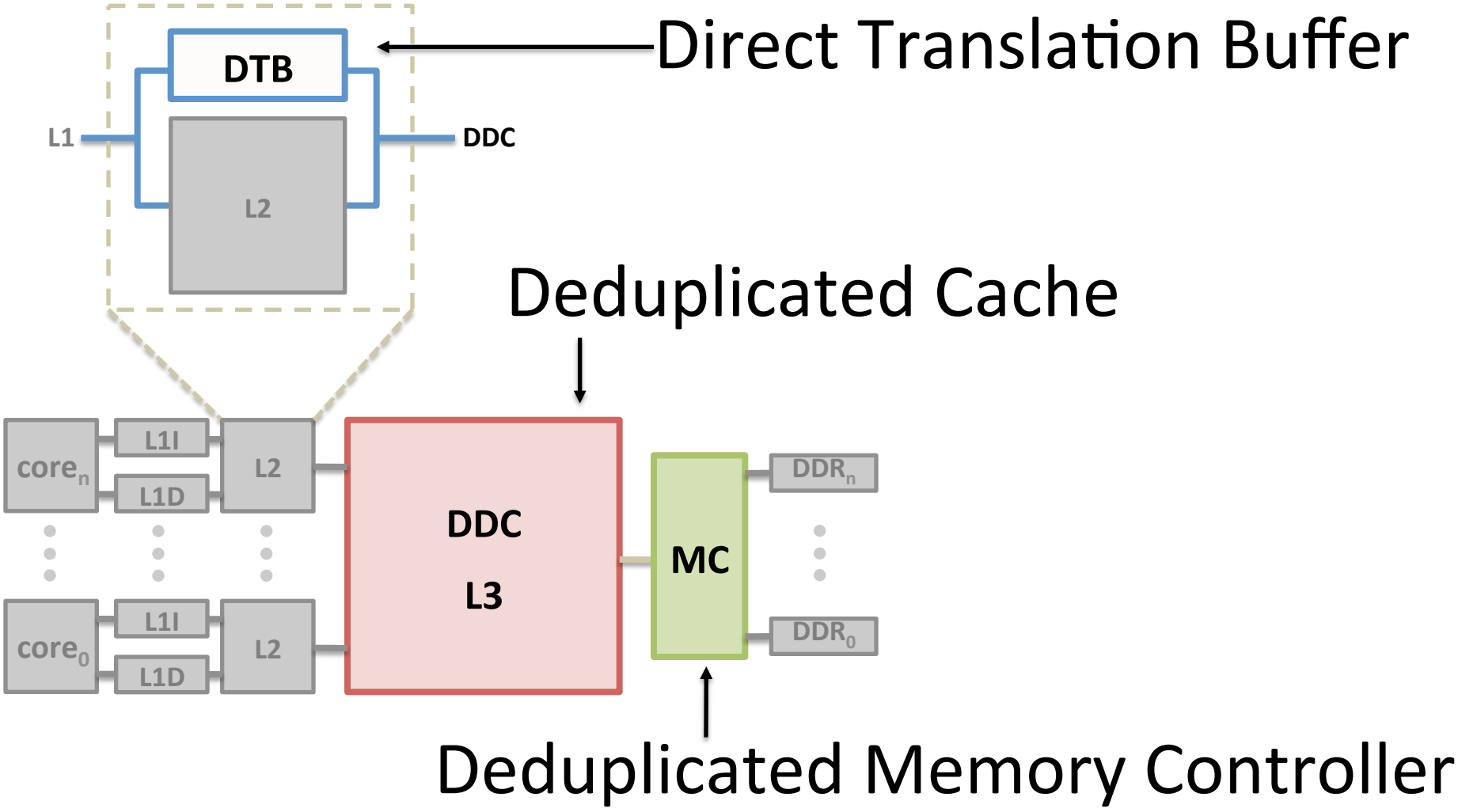
- Multiple references per line
- Garbage collection: When to free lines?



Overheads

- Translation Table
  - 64 Byte data lines
  - 32 bit PLIDs
  - Overhead: 1/16th
- Signatures
  - 1 byte per data line (1/64th)
- Reference Counts
  - 1 byte per data line (1/64th)

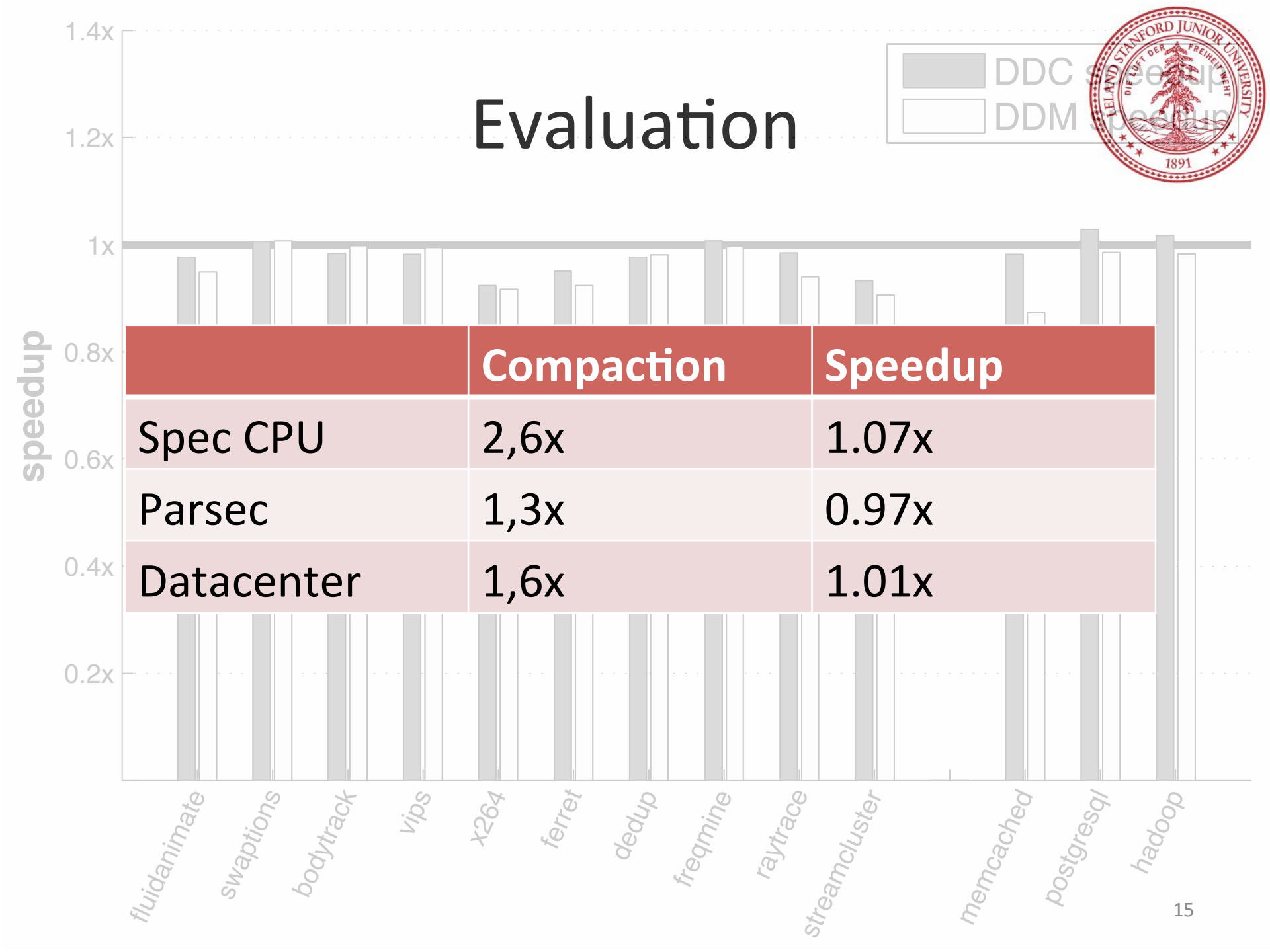
DDC System


Direct Translation Buffer
DTB
L1
DDC
L2
Deduplicated Cache
core$_n$
L1I
L1D
L2
core$_0$
L1I
L1D
L2
DDC
L3
MC
DDR$_n$
DDR$_0$
Deduplicated Memory Controller

Evaluation Methodology

- Cycle accurate simulator (ZSIM)
- Functional + timing models for
  - Deduplicated L3
  - Cache Coherency
  - Replacement strategy
  - Translation cache
  - In cache allocation
- Multi Workload Analysis

Evaluation



	Compaction	Speedup
Spec CPU	2,6x	1.07x
Parsec	1,3x	0.97x
Datacenter	1,6x	1.01x

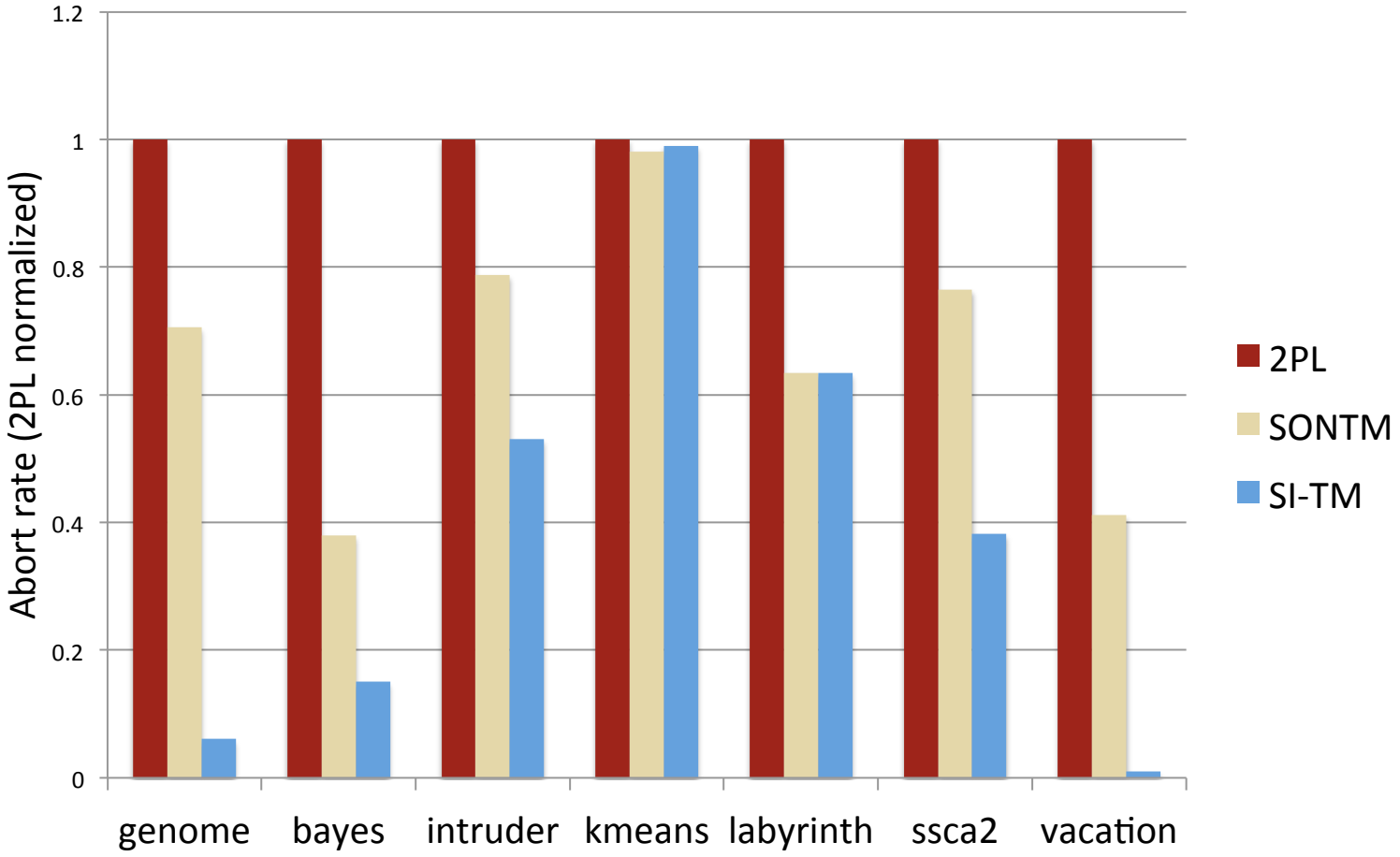
Compression vs. Dedup

- Both depend on redundancy/entropy
- Local vs. global
  - Runlength encoding
  - Huffman
  - 1KB buffer for Ziv-Lempel
- Compression is compute intensive
- Latency
  - Deduplication: 2 Mem Ops
  - Variable, depends on technique and buffer size
- Both can be combined (delta encoding)

Other uses of Indirection: SI-TM

- Snapshot isolation based Transactional Memory
- Extend translation layer to support versioning
- Efficient Copy on Write
- Concurrent reads & writes
- Read-only transactions always succeed
- Aborts only on write-write conflicts

Abort Rate Comparison



Concluding Remarks

- Increasing demands on memory capacity
  - Faster, simpler, more predictable
- Deduplication
  - Increases capacity by 2x
  - Amortizes cost of indirection layer
- Level of indirection opens the door to many other optimizations
- Compelling processor extension

Questions?

heiner.litz@stanford.edu

Outlook II - New Memory Technologies

- Non-Volatile
  - Flash DIMM (Diablo), RRAM (Crossbar), PCM
  - Deduplication (↑ capacity)
  - Fine grain wear leveling (↑ resilience)
  - Fine grain chipkill (↓ spare capacity)
- Hybrid Memory Cube
  - Heterogeneous Access Latency
  - Data remap to benefit from Locality (↑ performance)

Enabled by Translation Layer

Indirection Layer II

- Memory error remapping (Chip kill)
- Fine grain wear leveling (NVRam)
- Traffic steering (exploit locality/heterogeneity)
- Memcopy
  - Copy PLIDs instead of data