UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Multigrain Coherence Directories

Dr. Jason Zebchuk
Principal Engineer, Cavium

Prof. Andreas Moshovos, *University of Toronto*
Prof. Babak Falsafi, *EcoCloud, EPFL*

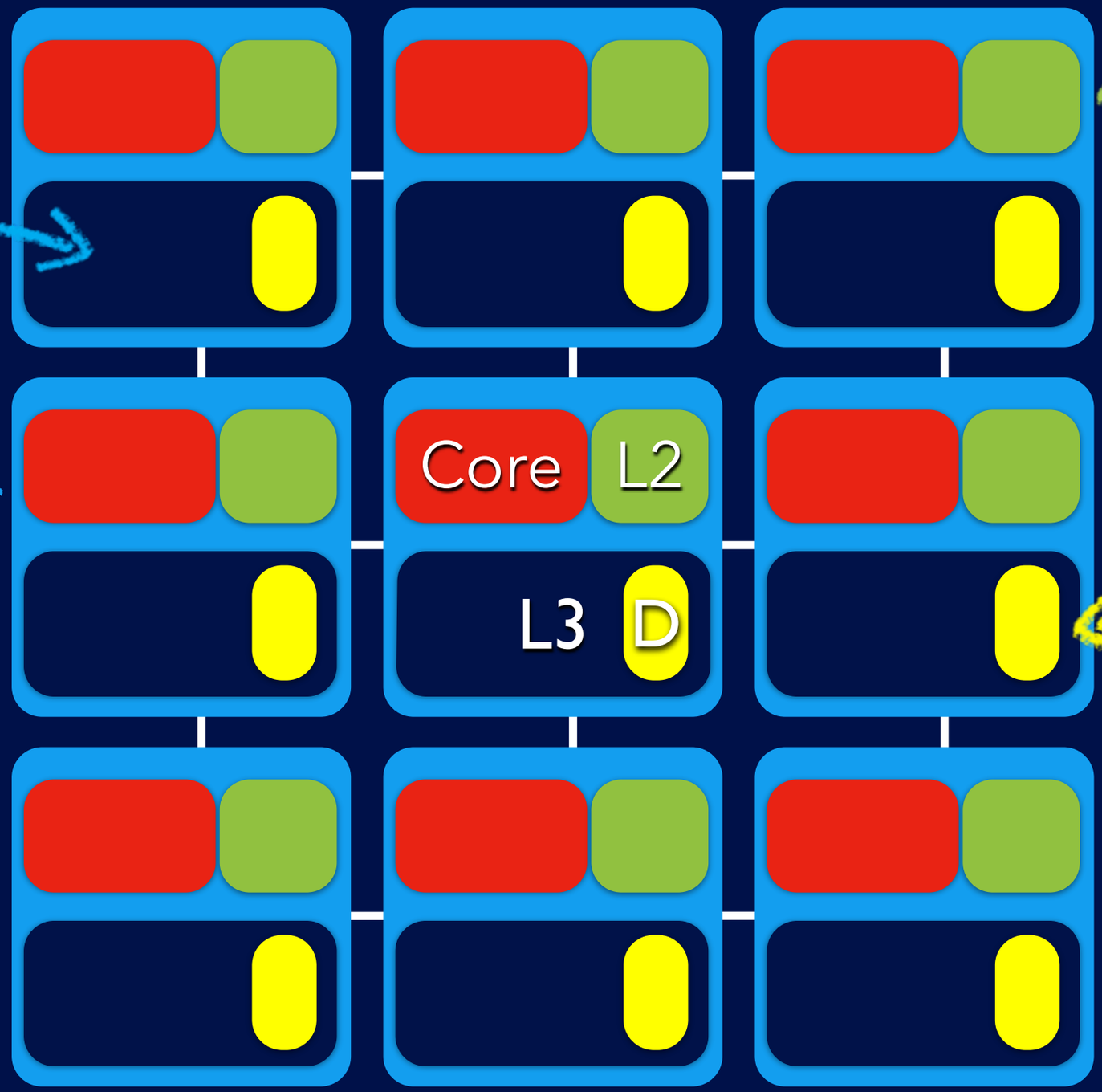
UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

Multigrain Coherence Directories

Dr. Jason Zebchuk
Principal Engineer, Cavium

Prof. Andreas Moshovos, *University of Toronto*
Prof. Babak Falsafi, *EcoCloud, EPFL*

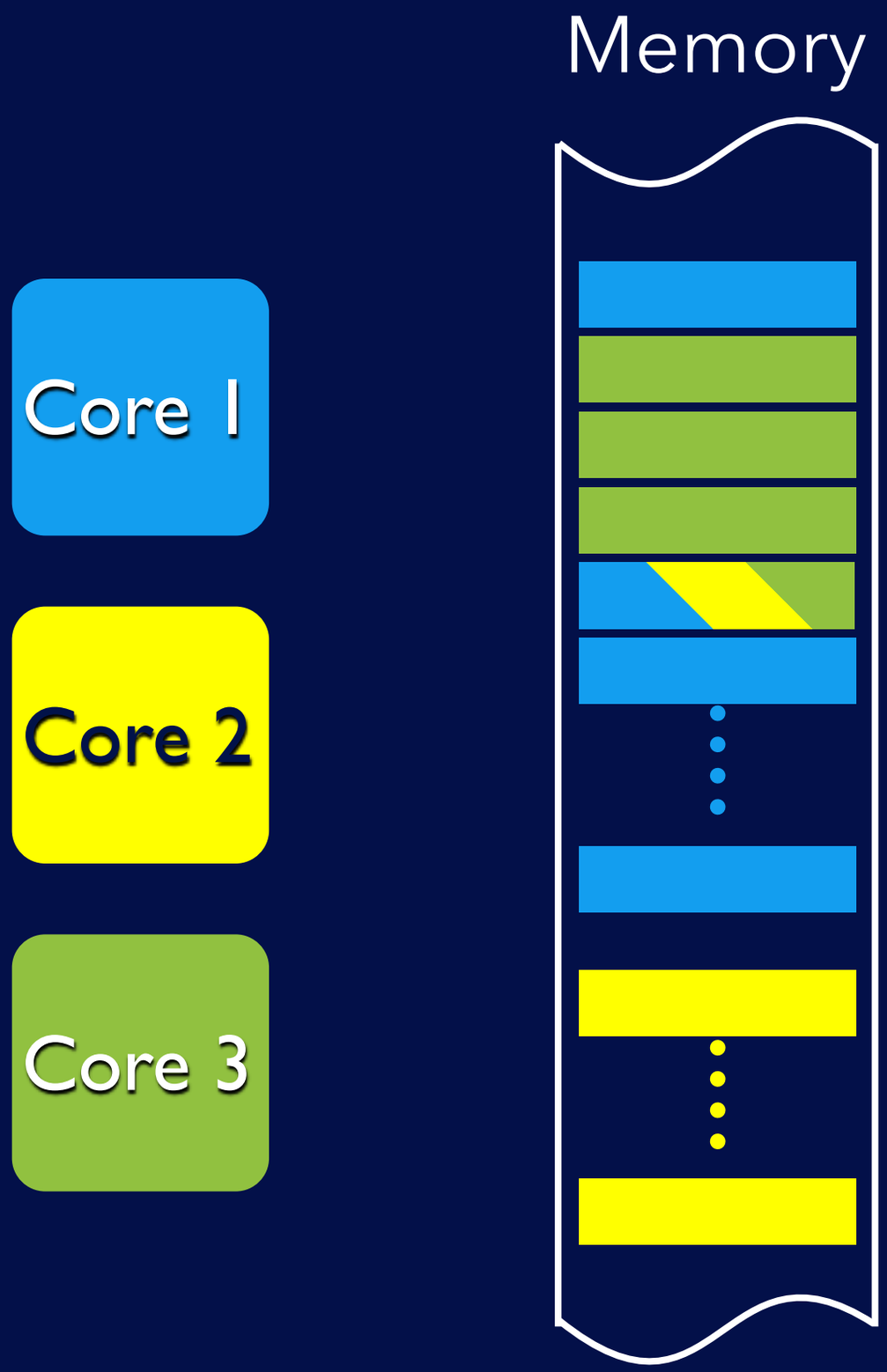
Many-Core Processor

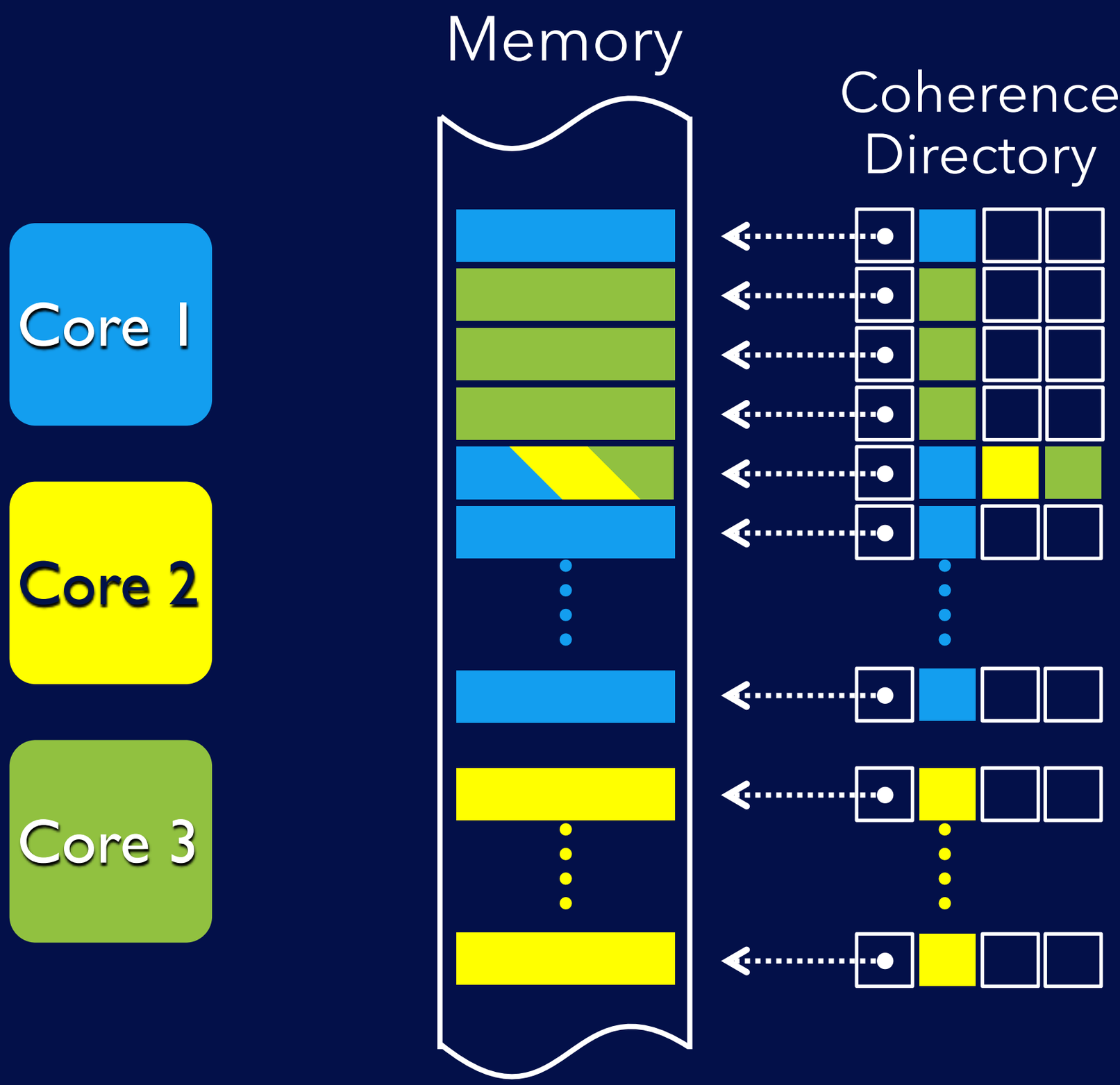


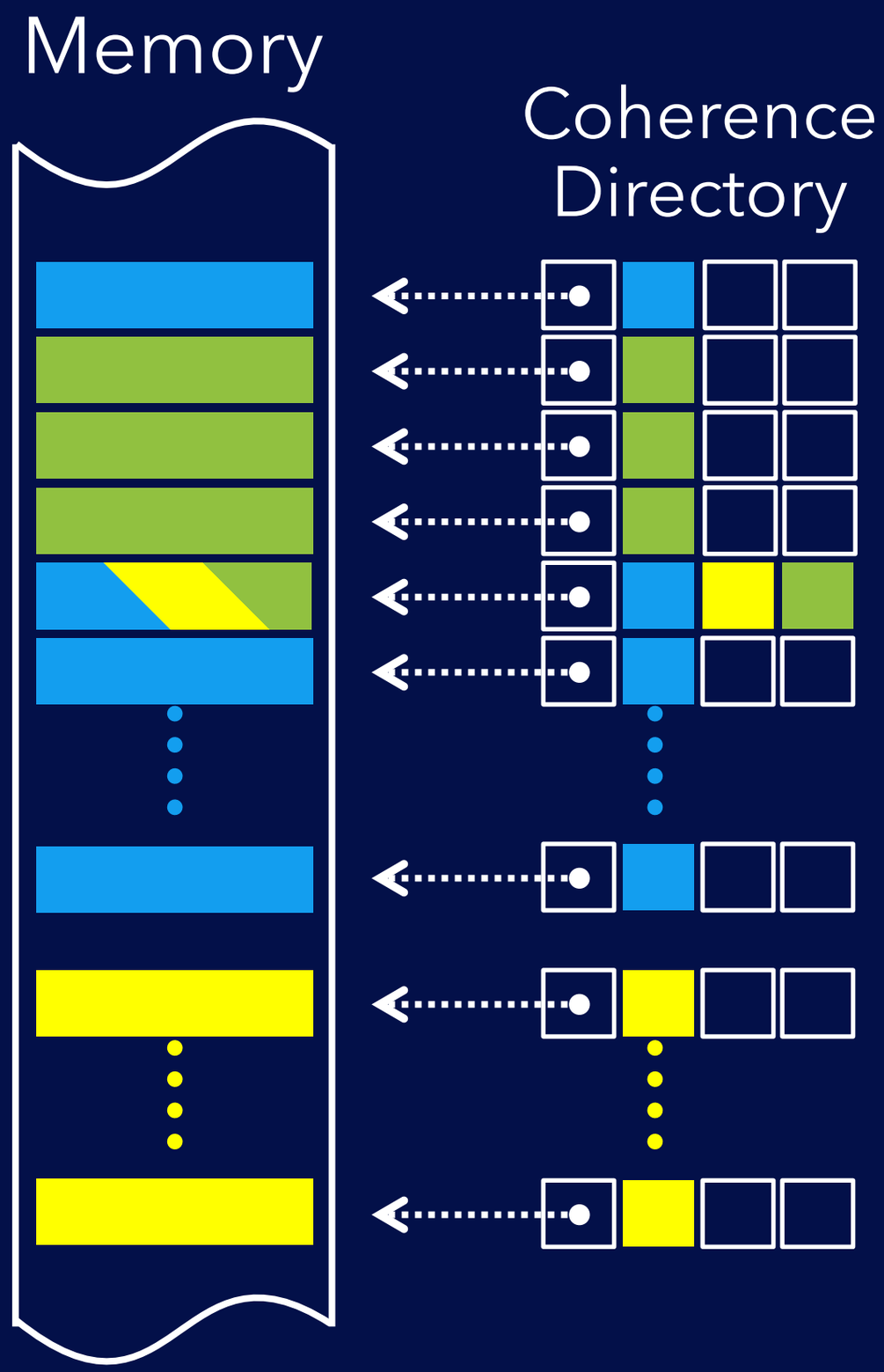
Private L2 Cache

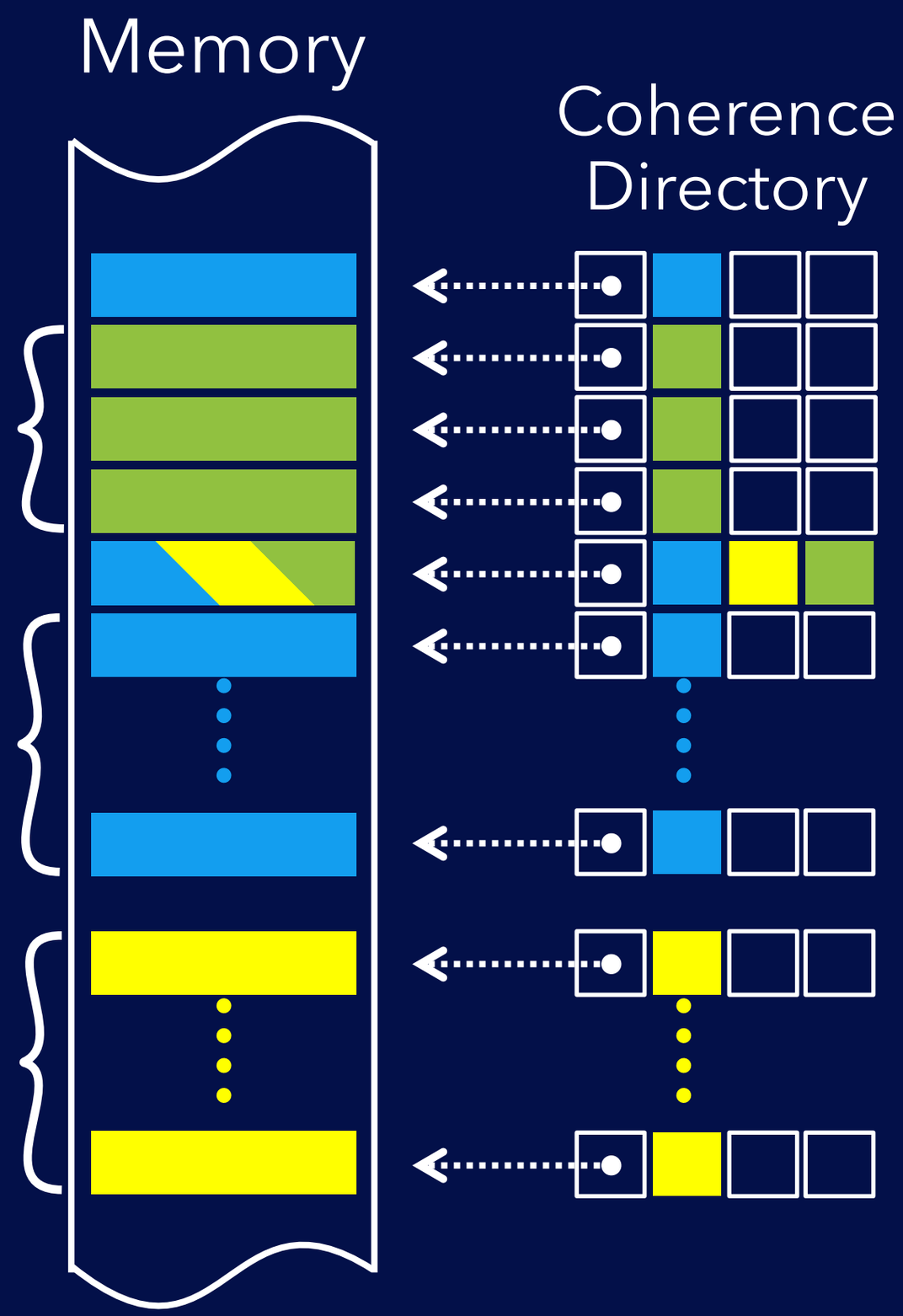
Shared Distributed L3 Cache

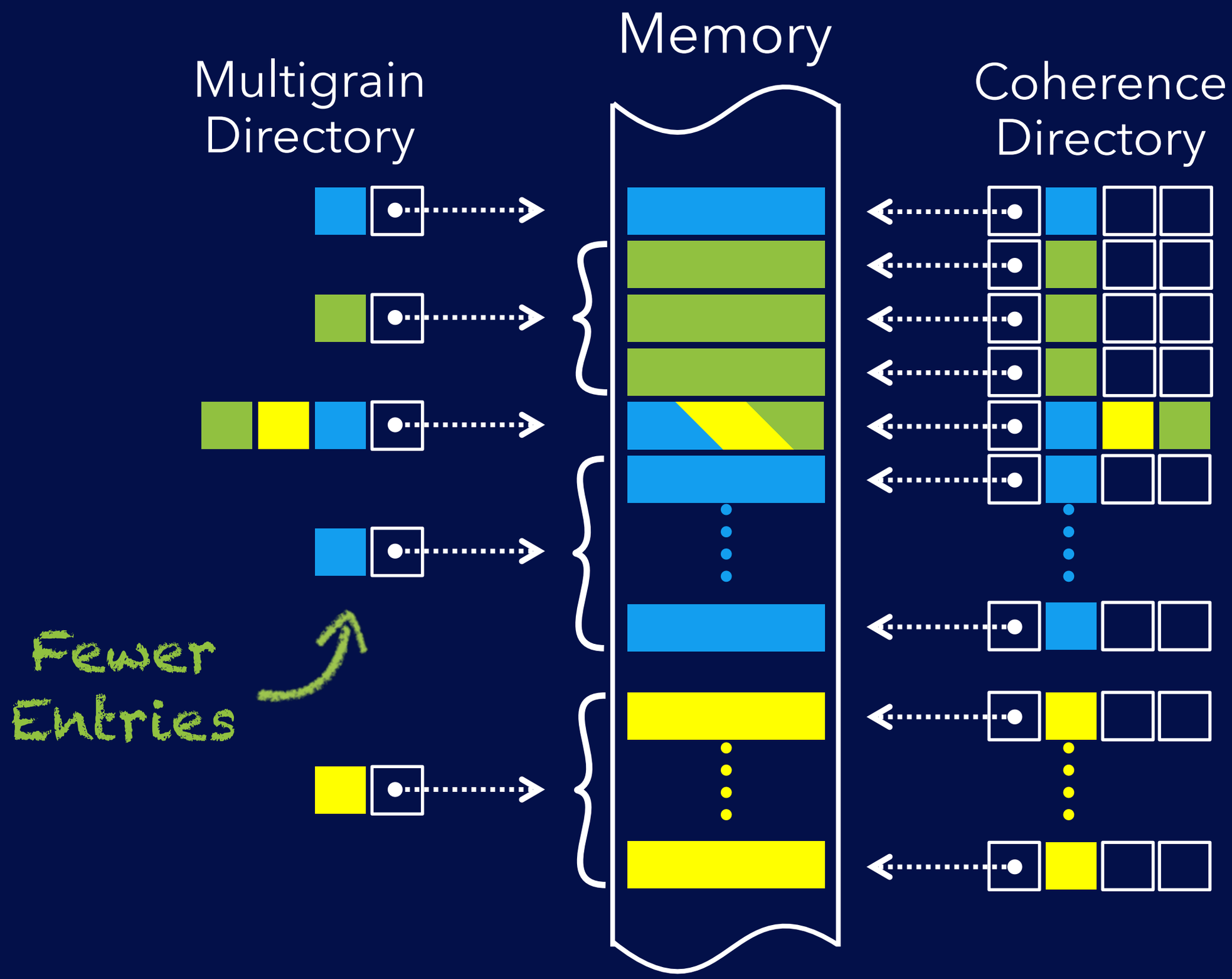
Coherence Directory!

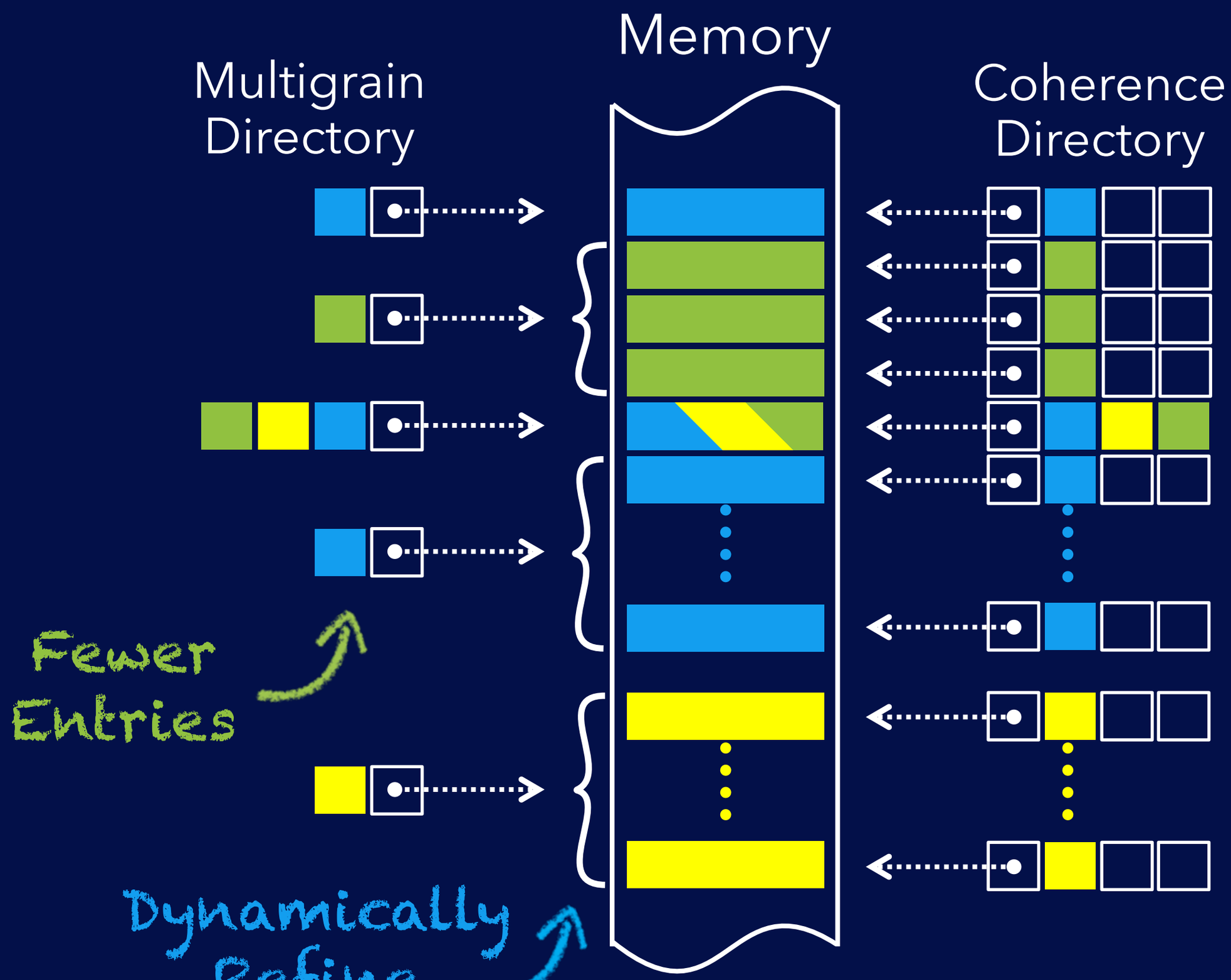

Memory
Core 1
Core 2
Core 3

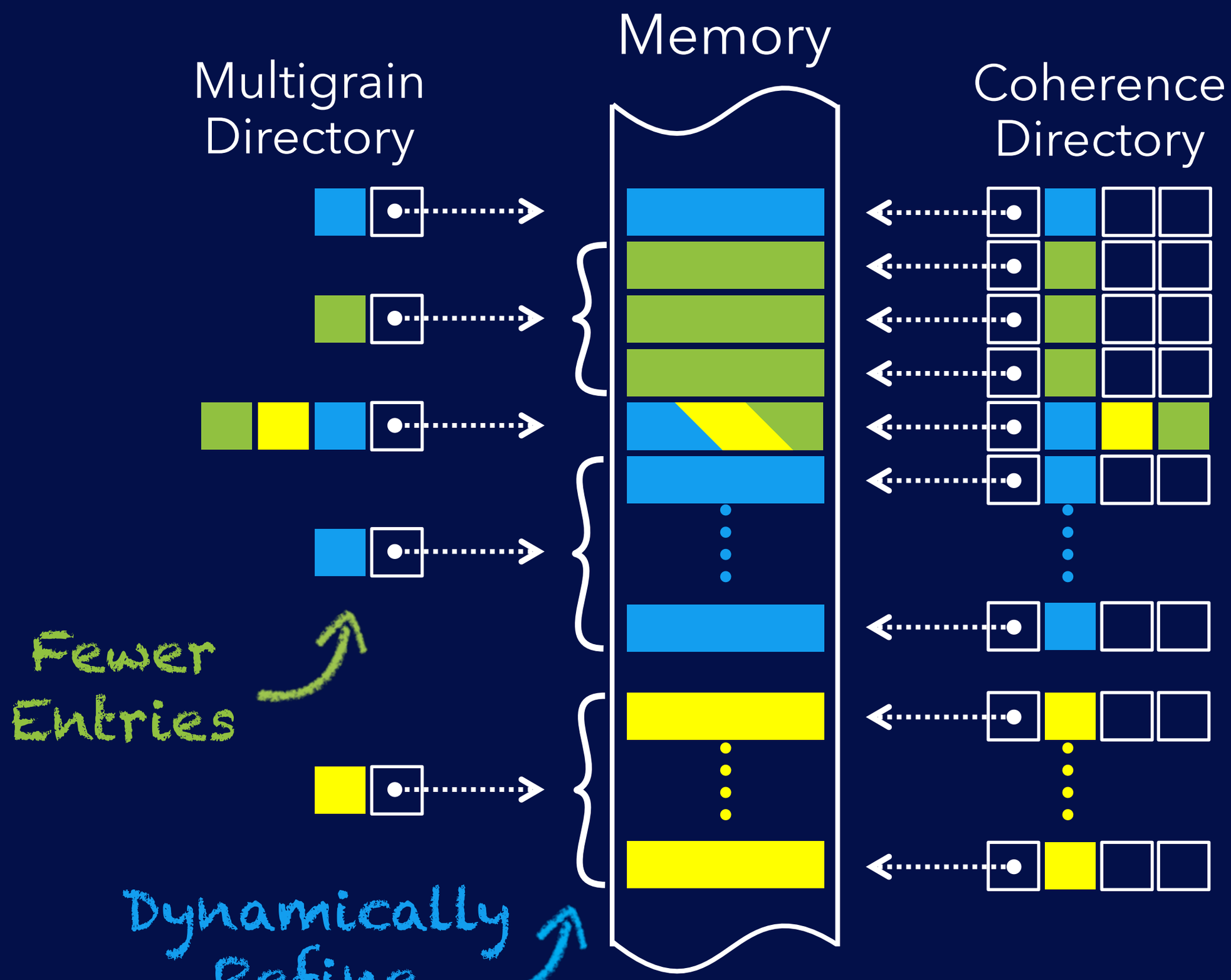

Memory
Coherence Directory
Core 1
Core 2
Core 3


Memory
Coherence Directory


Memory
Coherence Directory


Memory
Multigrain Directory
Coherence Directory
Fewer Entries


Memory
Multigrain Directory
Coherence Directory
Fewer Entries
Dynamically Refine Granularity


Memory
Multigrain Directory
Coherence Directory
Fewer Entries
Dynamically Refine Granularity

Multi-Grain Coherence Directory (MGD)

Conceptual MGD Directory:

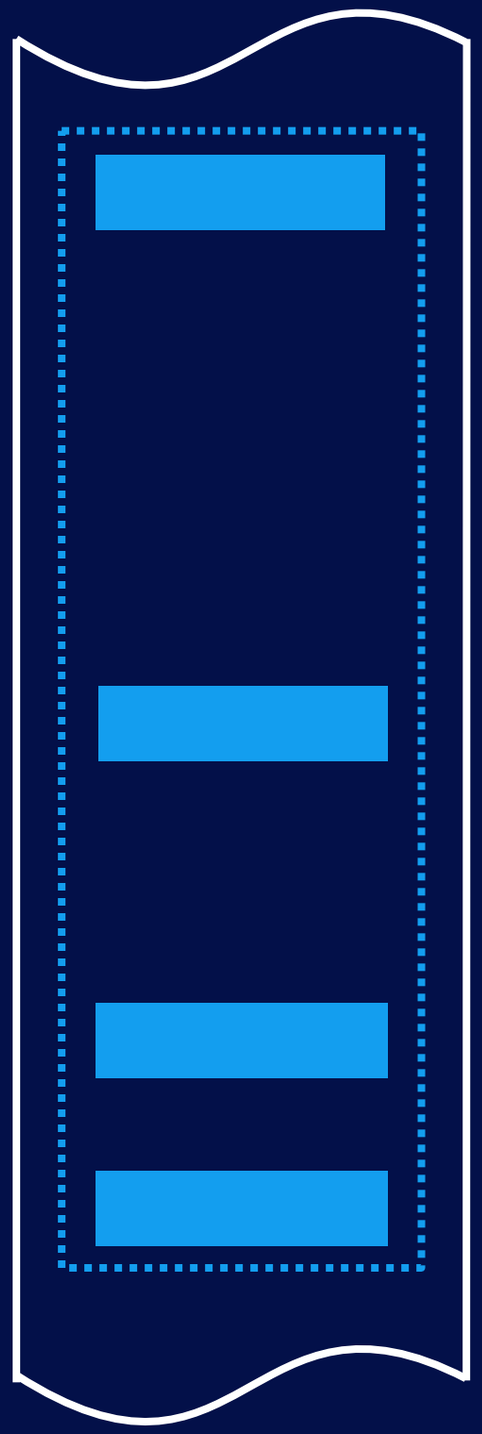
- ✓ Dynamically refine granularity of entries
- ✓ 78% fewer directory entries (on average)

Practical MGD Directory:

- ✓ Limited number of fixed granularities
- ✓ 41% less area
- ✓ No coherence protocol changes
- ✓ Robust performance

- **Original MGD concept**
- Potential benefits of MGD
- Making a practical design
- A recent competing design
- Some nice graphs
- Final Thoughts

Memory



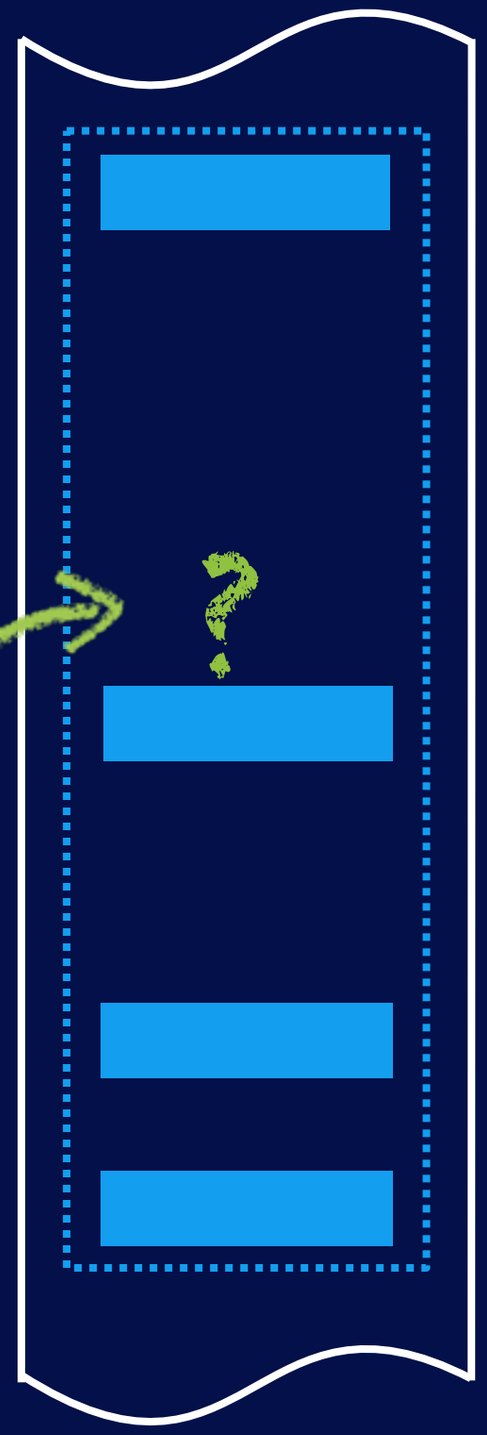
Core 1

Core 2

Core 3

Each entry tracks a private region of memory

Memory



Core 1

Core 2

Core 3

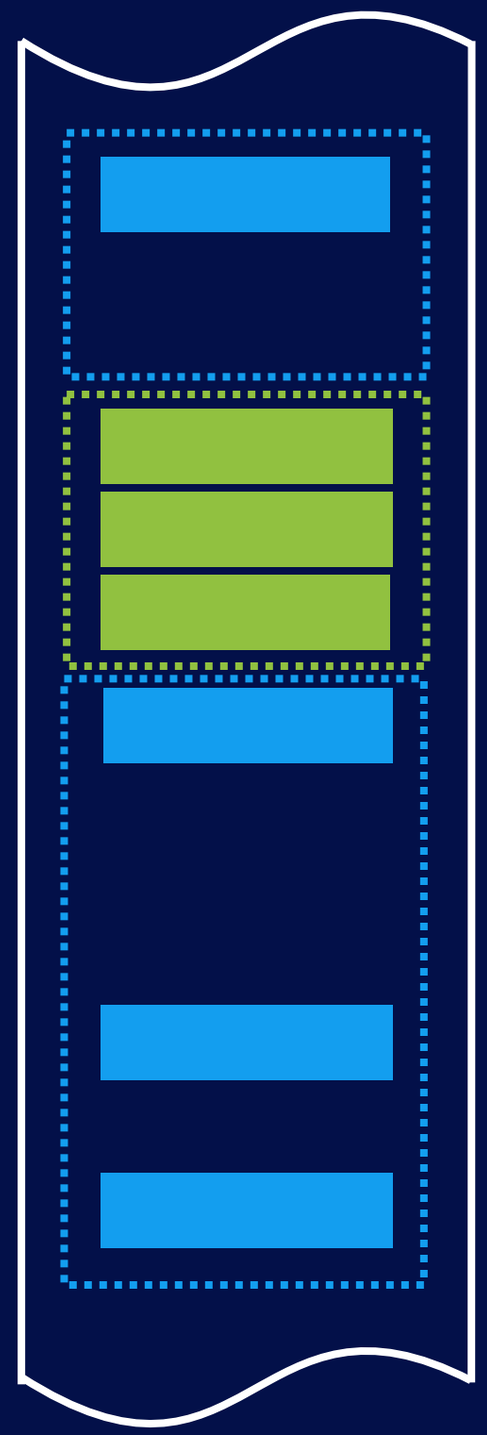
Each entry tracks a private region of memory

Memory

Core 1

Core 2

Core 3



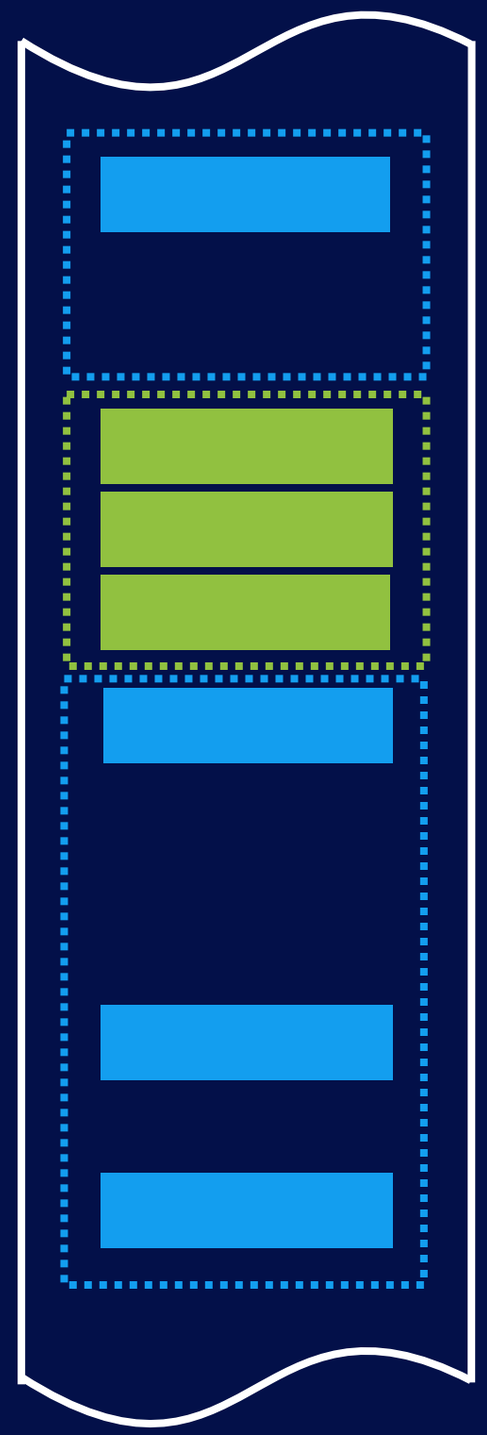
Dynamically adapt region size for each entry

Memory

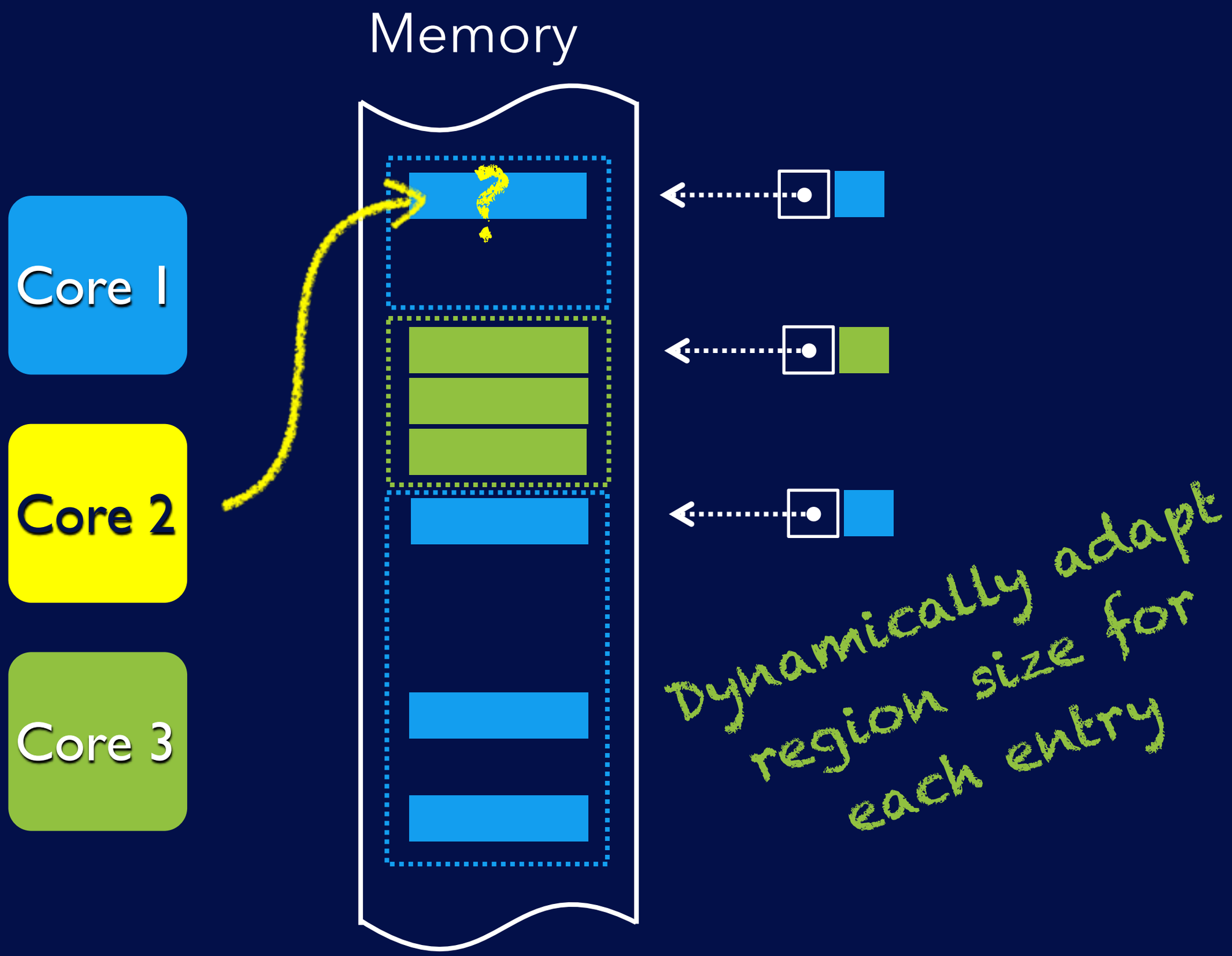
Core 1

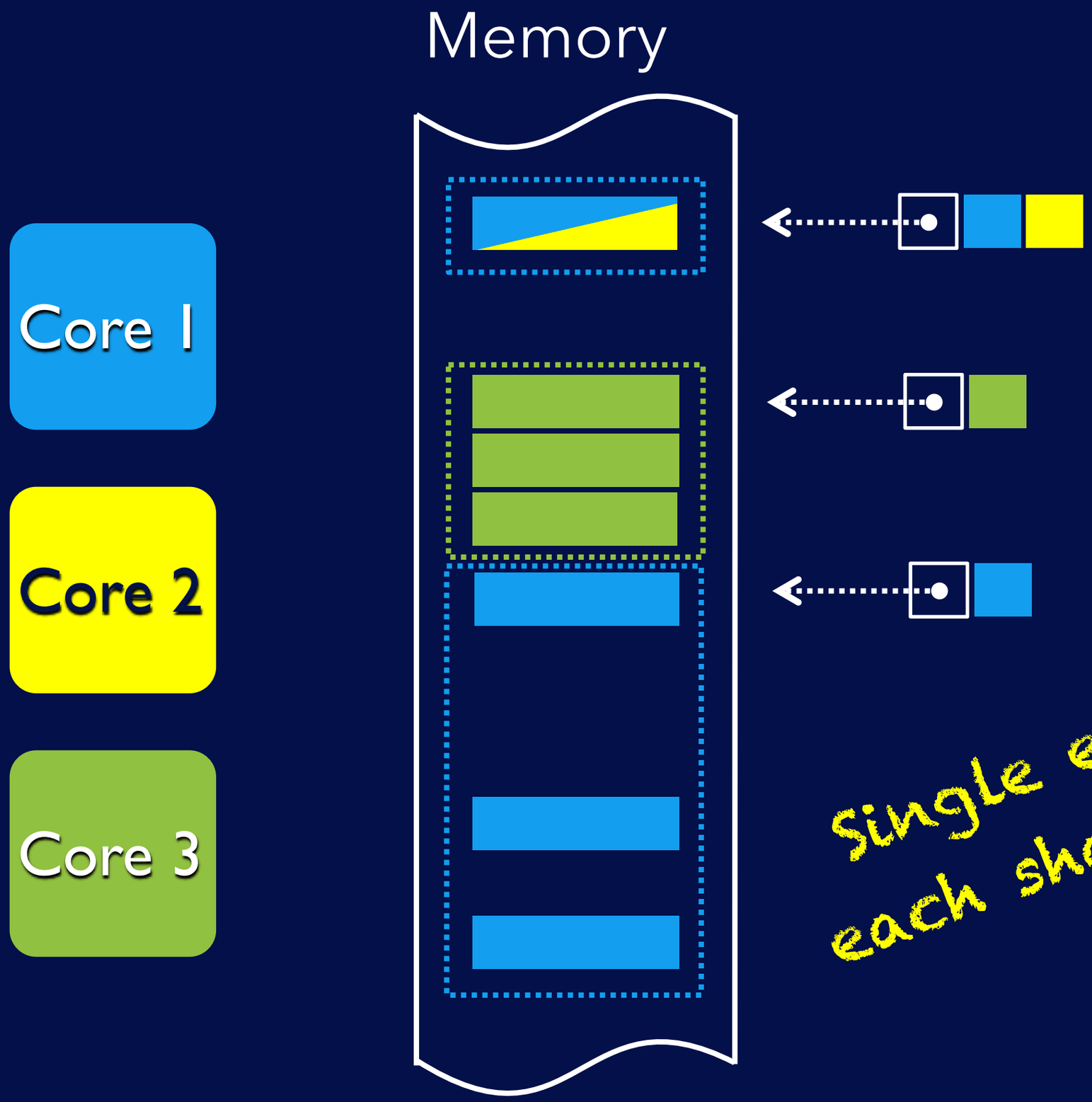
Core 2

Core 3



Dynamically adapt region size for each entry


Memory
Core 1
Core 2
Core 3
?
Dynamically adapt region size for each entry

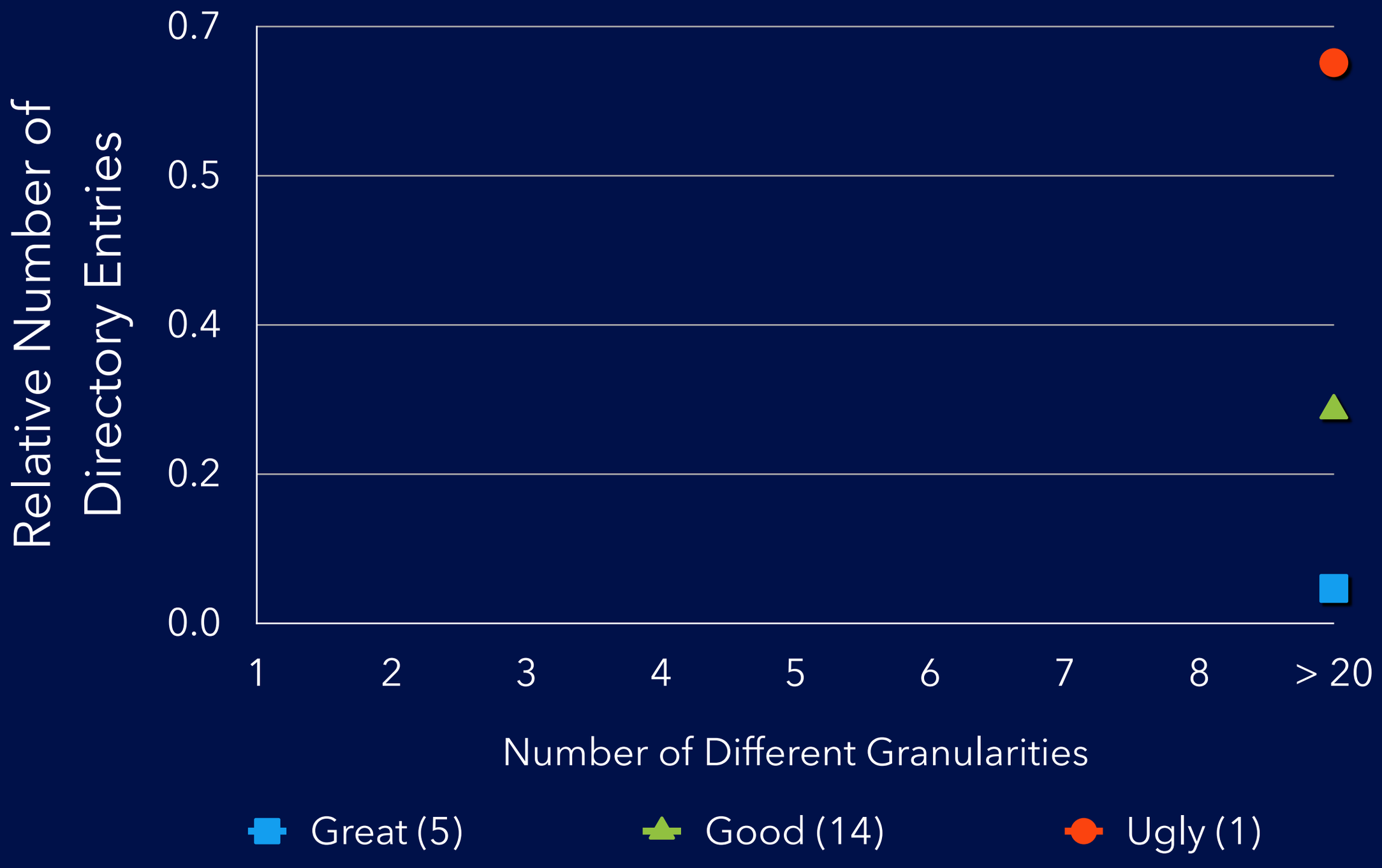

Memory
Core 1
Core 2
Core 3
Single entry for each shared block

Conceptual Multi-Grain Directory

- **Thought experiment, not a real design**
- Entries track *instantaneous* private regions
- Dynamically refine region granularity
  - *Maximize size of each private region*
  - *Limit to powers of two, properly aligned*
- Shared blocks assigned individual entries

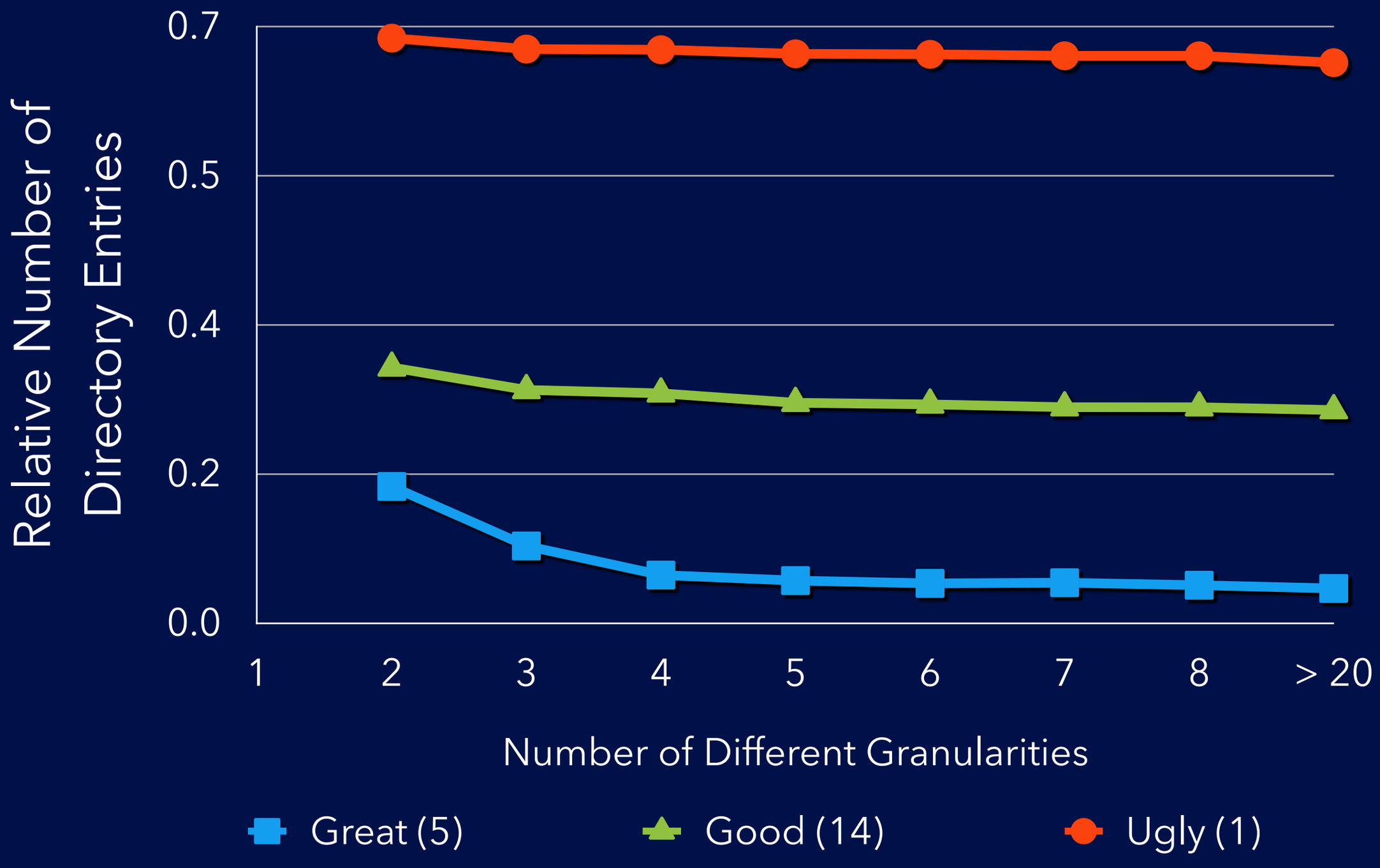
- Original MGD concept
- **Potential benefits of MGD**
- Making a practical design
- A recent competing design
- Some nice graphs
- Final Thoughts

Conceptual MGD Potential

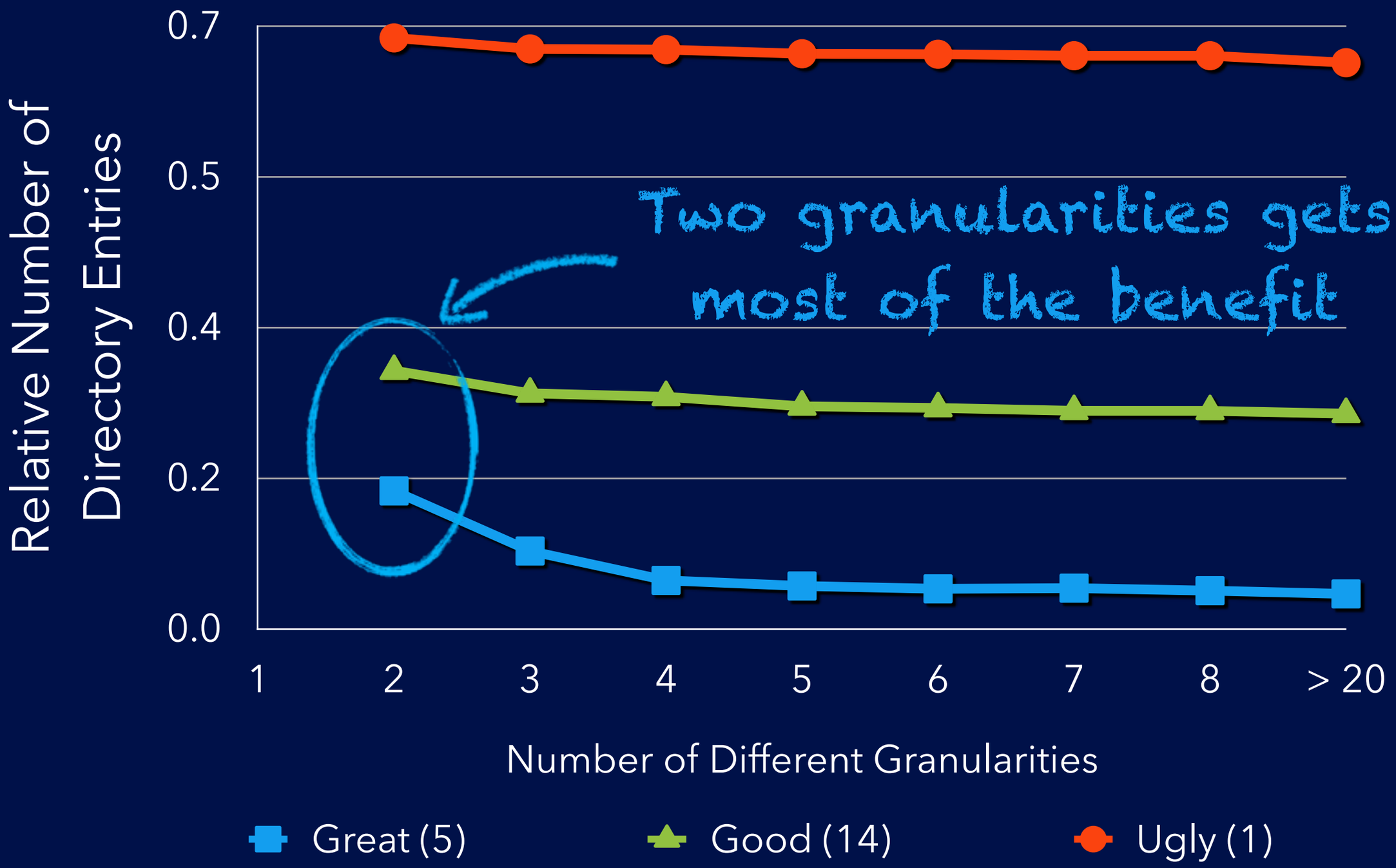


Conceptual MGD Potential

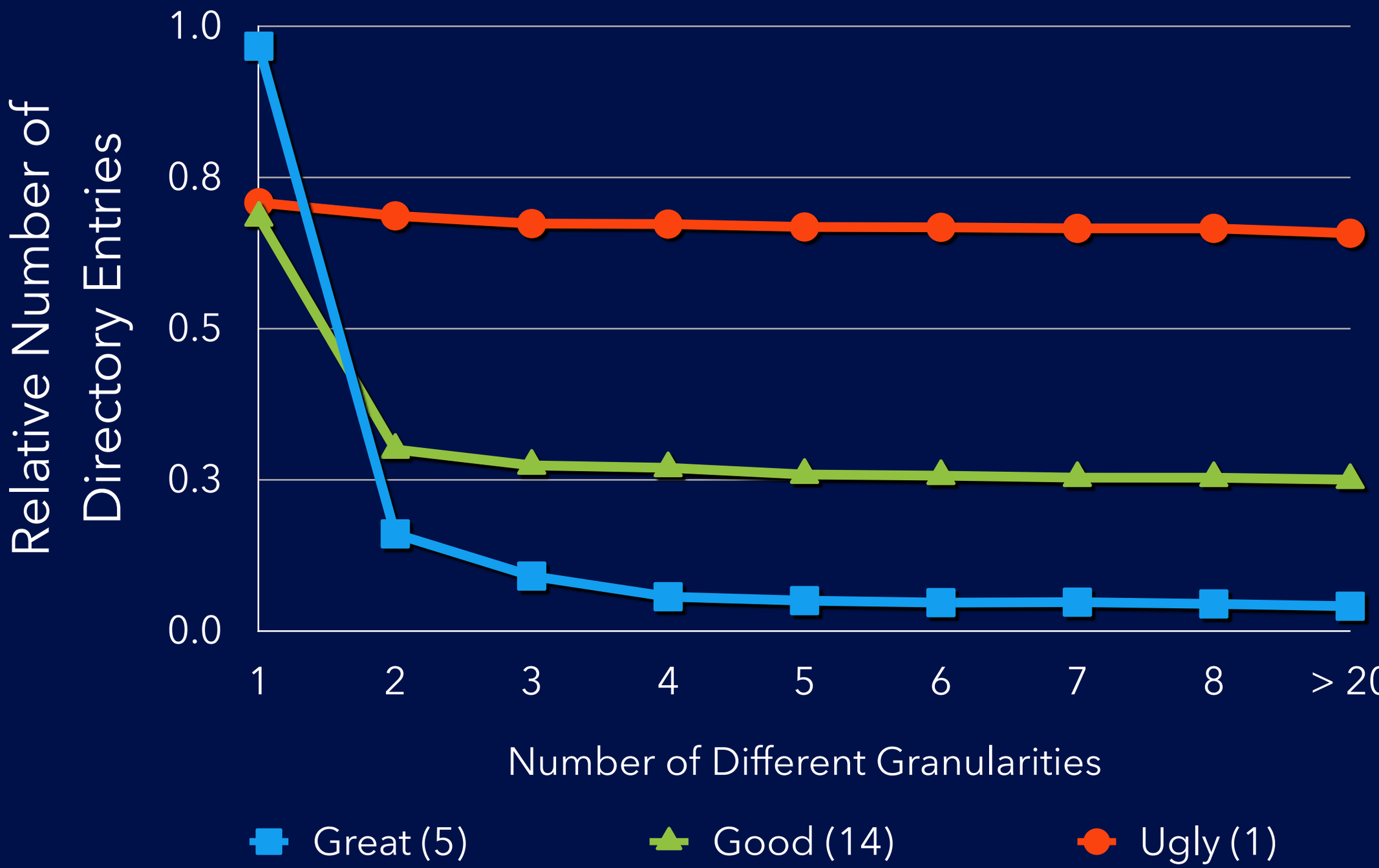
Conceptual MGD Potential



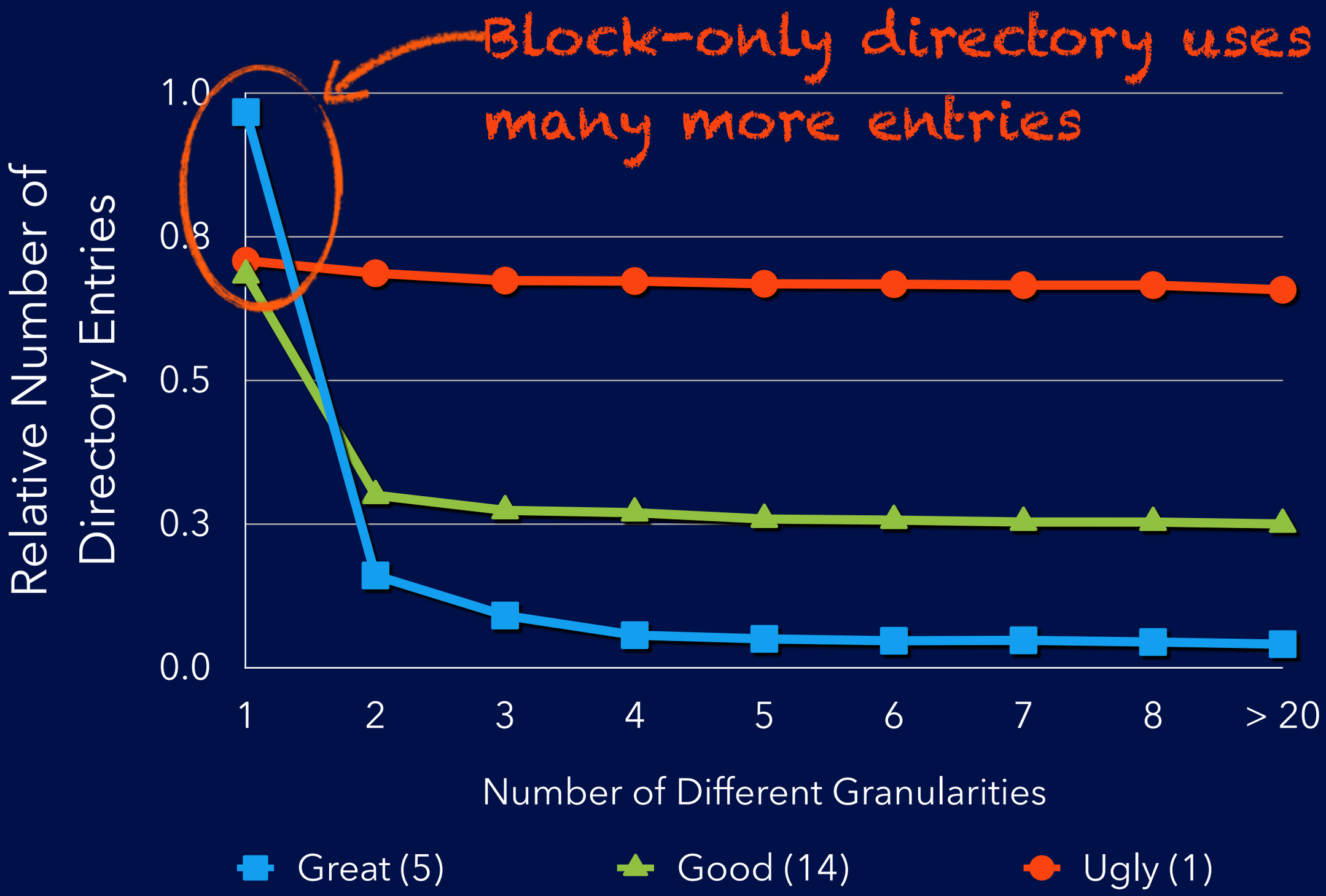
Conceptual MGD Potential



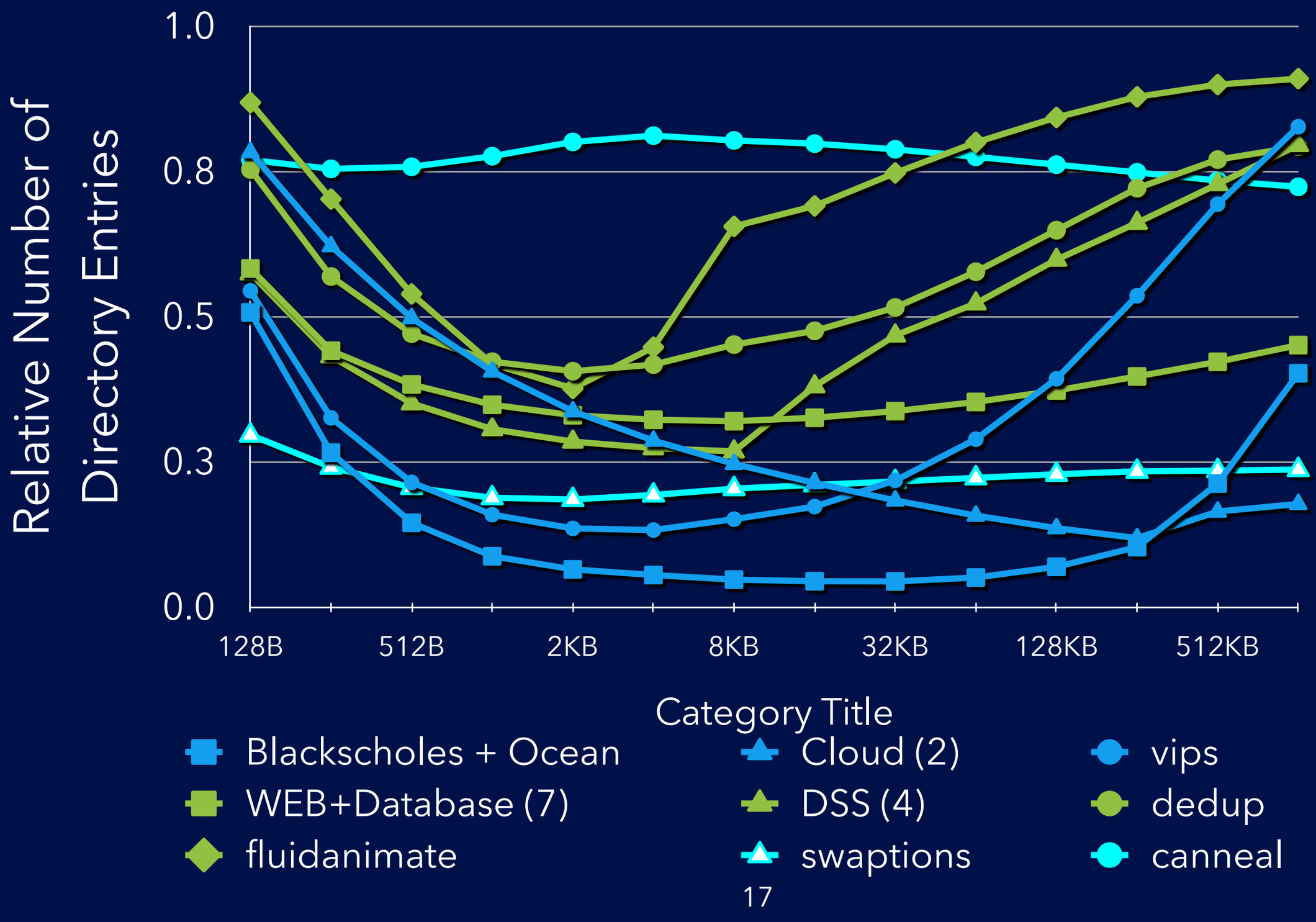
Conceptual MGD Potential



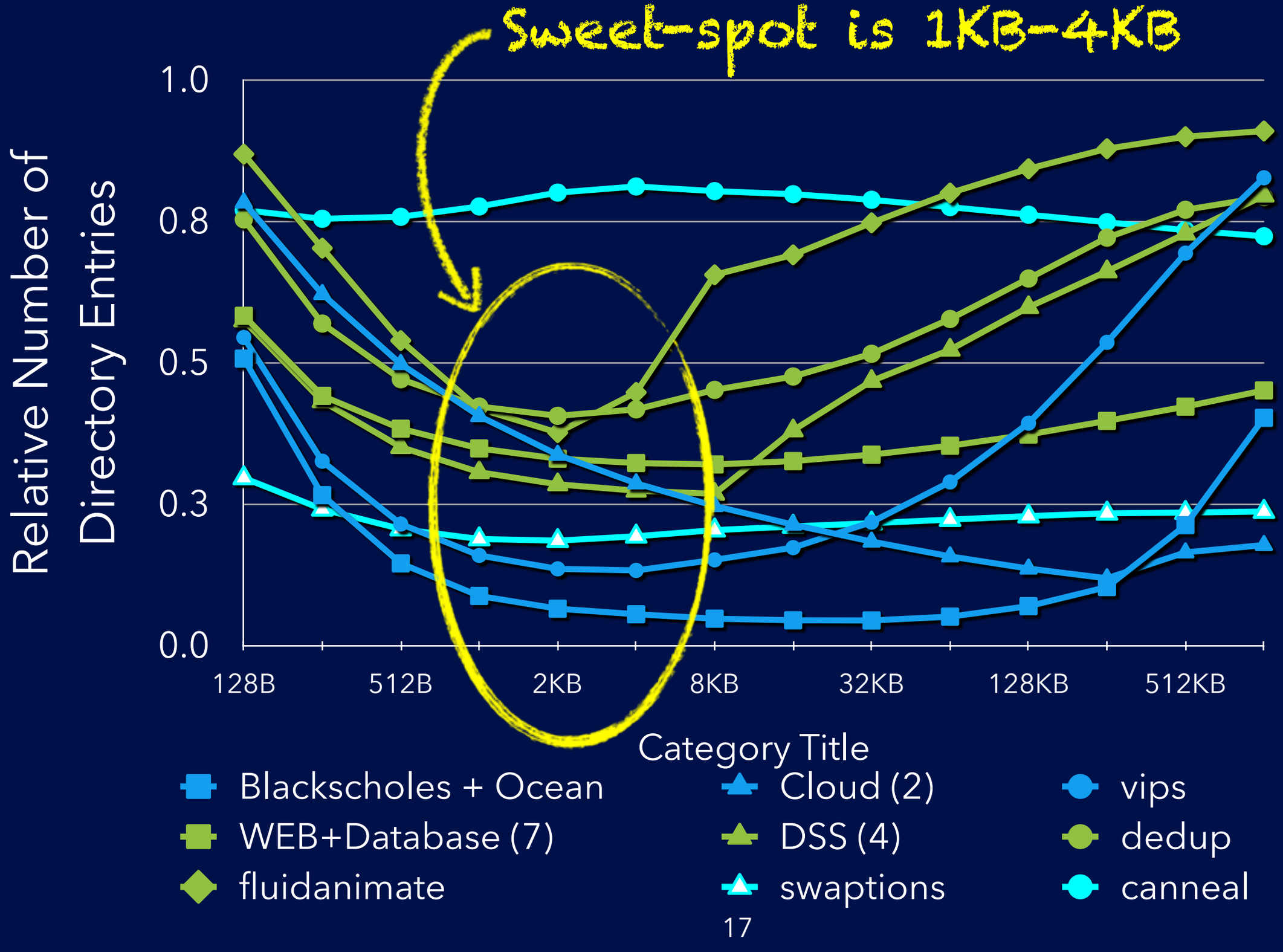
Conceptual MGD Potential



Region Size for Dual-Grain MGD



Region Size for Dual-Grain MGD



- Original MGD concept
- Potential benefits of MGD
- **Making a practical design**
- A recent competing design
- Some nice graphs
- Final Thoughts

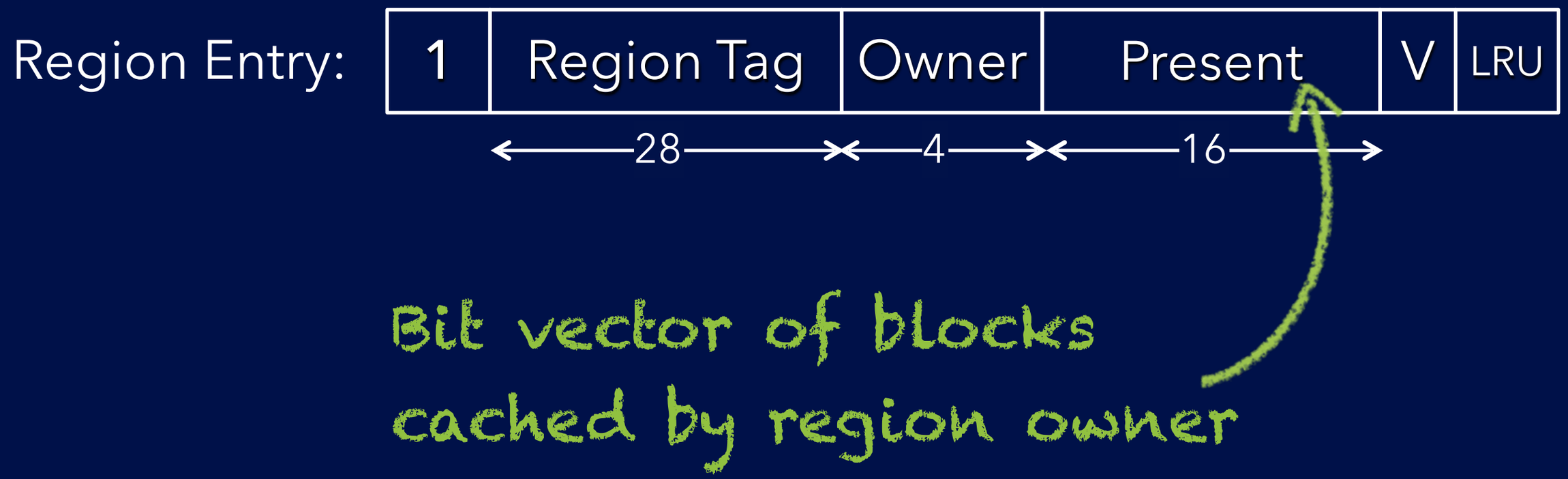
Practical Multi-Grain Directory

*A **Dual**-Grain Directory (DGD)*

- **Real design, not just a thought experiment**
- Track instantaneous private regions
- Use two fixed granularities
  - *individual cache blocks + regions of 1KB - 4KB*
- ***How will this work?***

What do Entries Look Like?

What do Entries Look Like?



What do Entries Look Like?

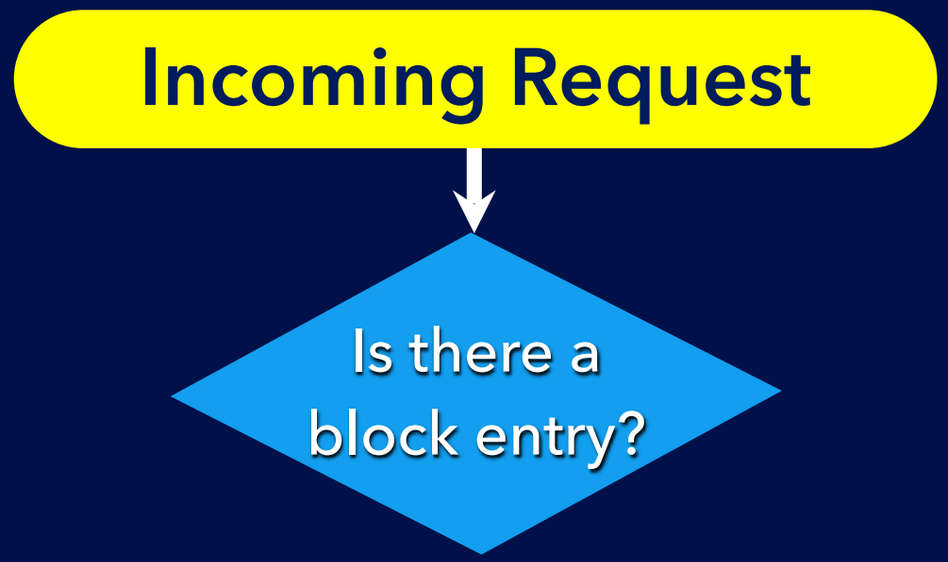
Both entry types are the same size

Bit vector of blocks
cached by region owner

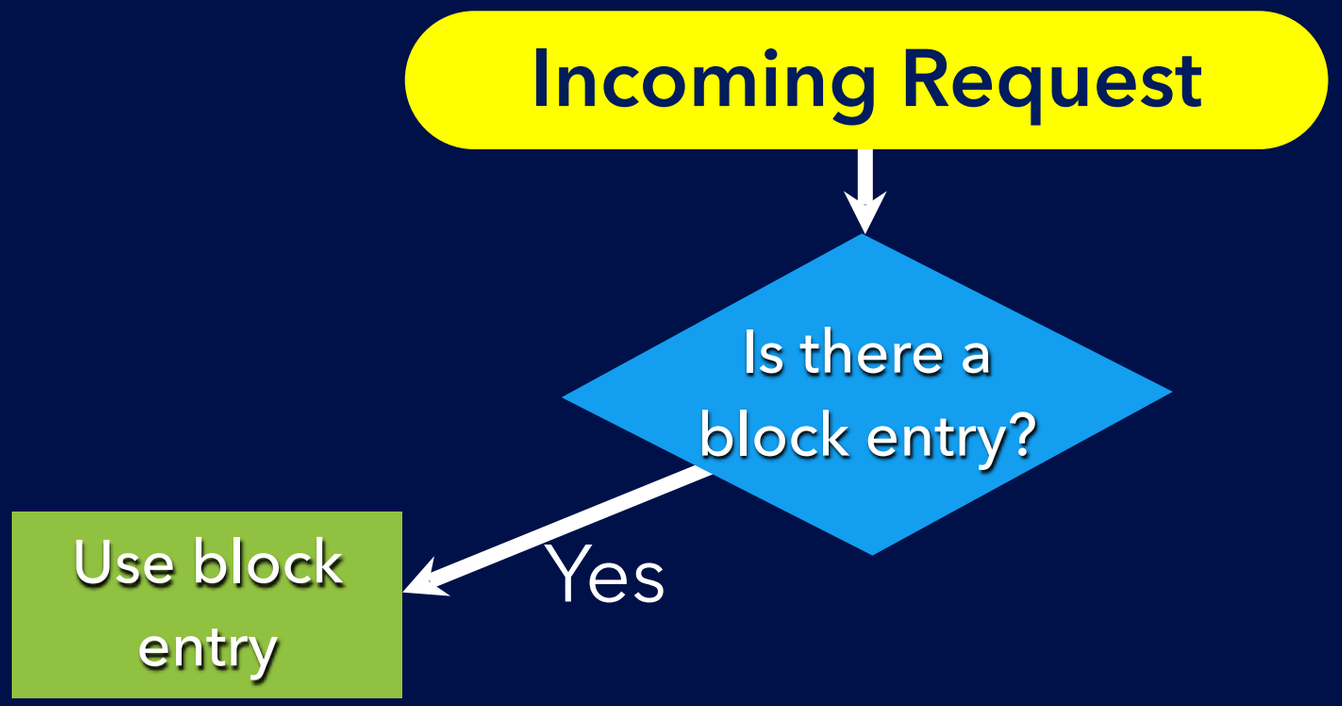
Dual-Grain Directory (DGD) Function

Incoming Request

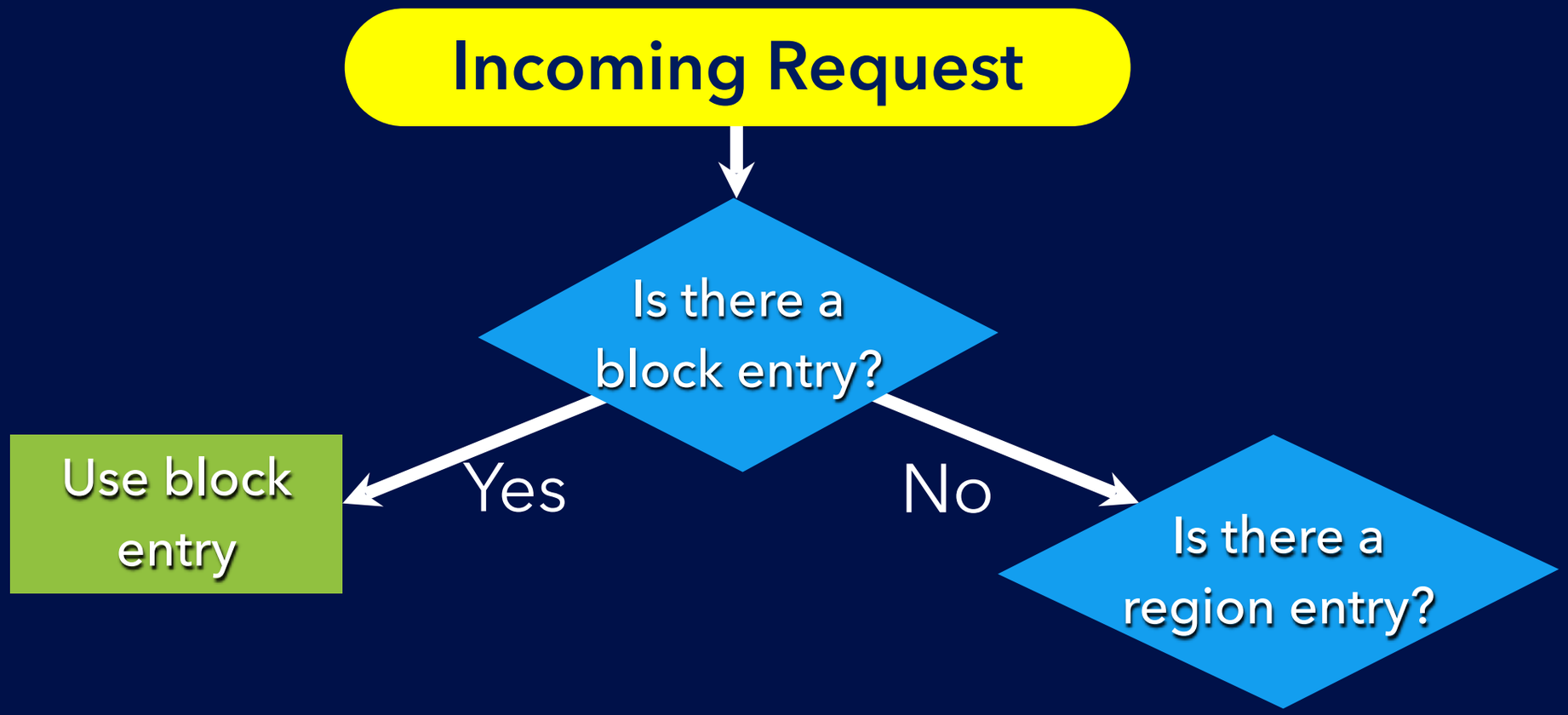
Dual-Grain Directory (DGD) Function



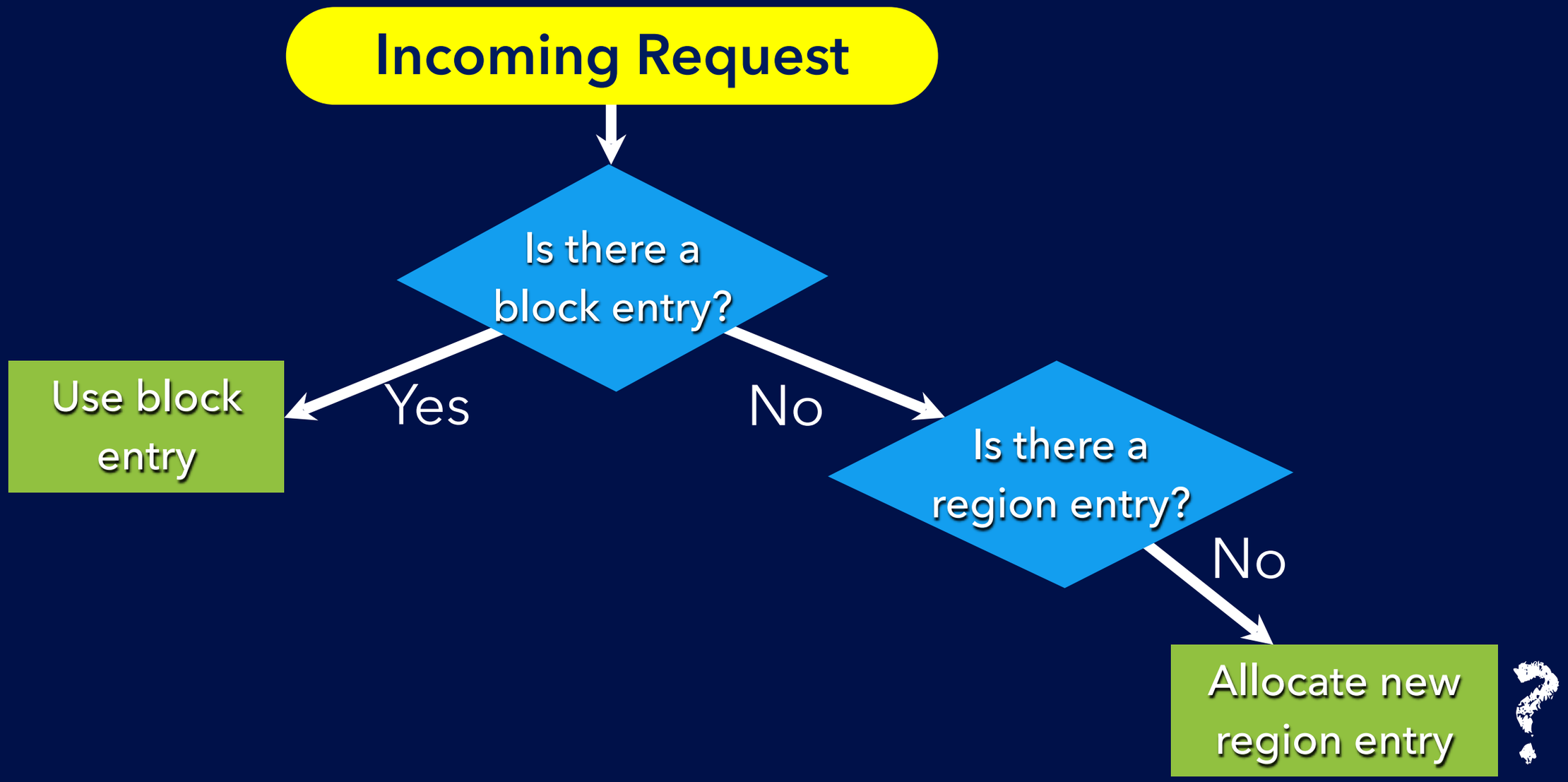
Dual-Grain Directory (DGD) Function



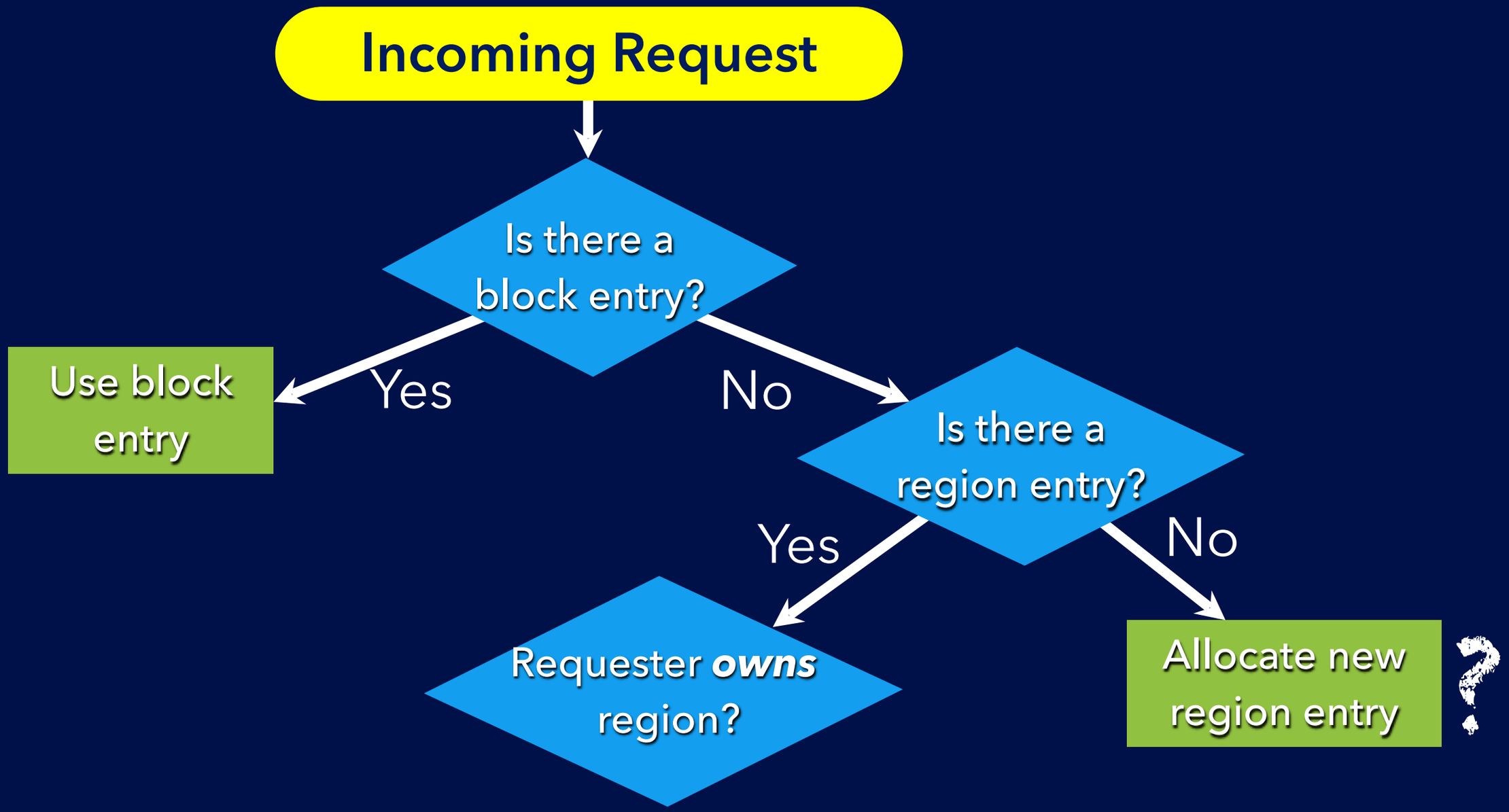
Dual-Grain Directory (DGD) Function



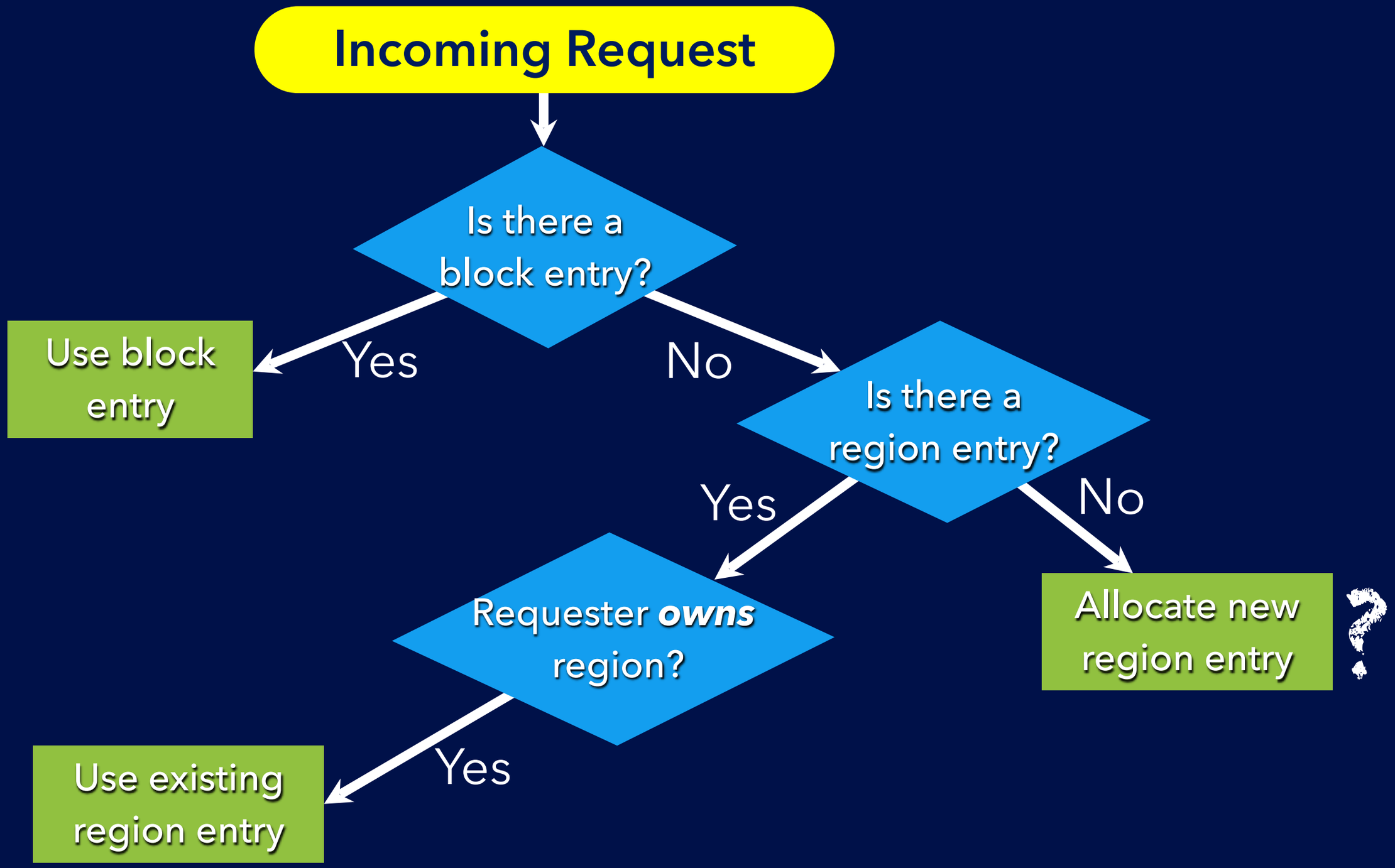
Dual-Grain Directory (DGD) Function



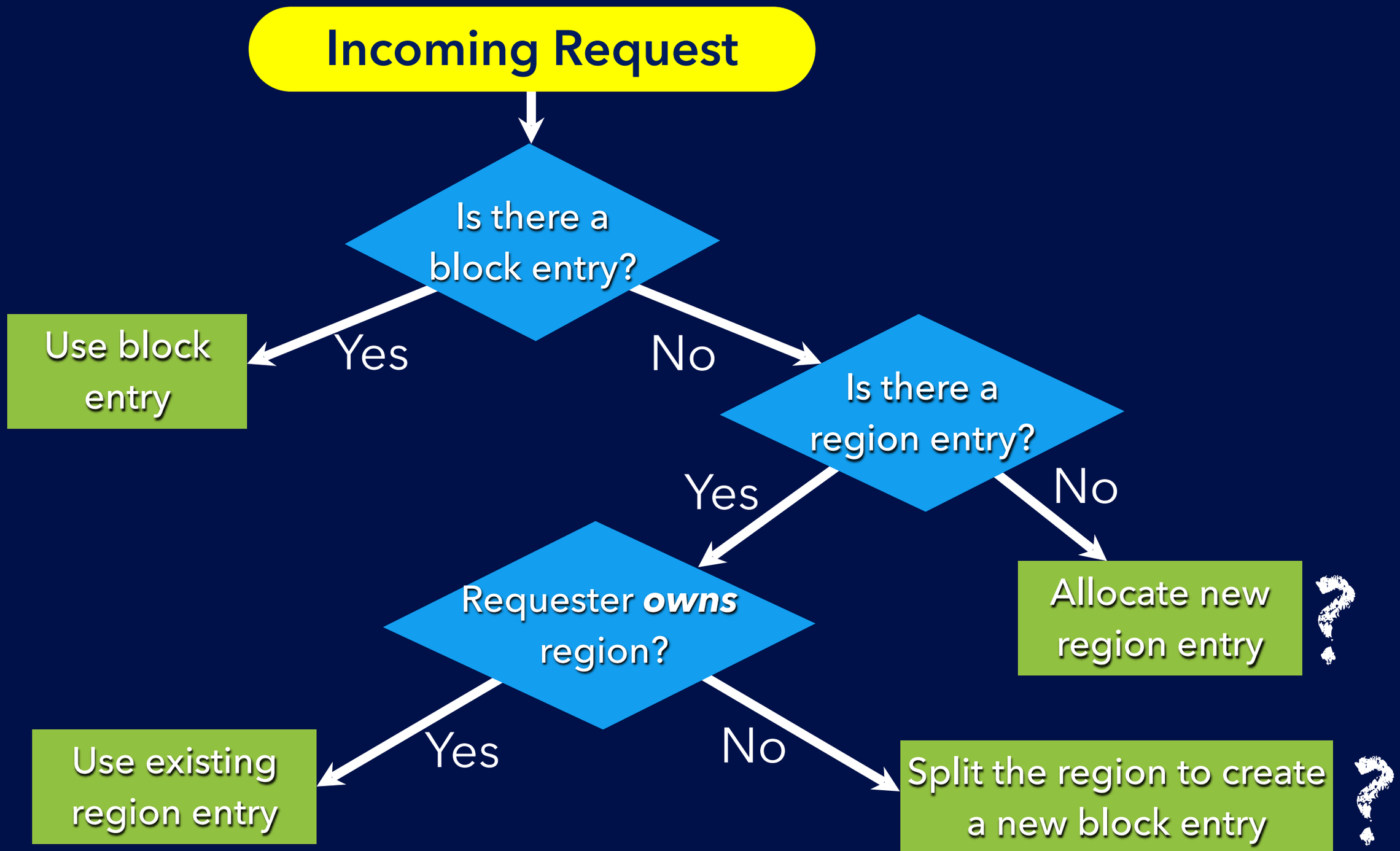
Dual-Grain Directory (DGD) Function



Dual-Grain Directory (DGD) Function



Dual-Grain Directory (DGD) Function



Memory

Region Entry

Core 1

Core 2

Core 3

Memory

Region Entry

Core 1

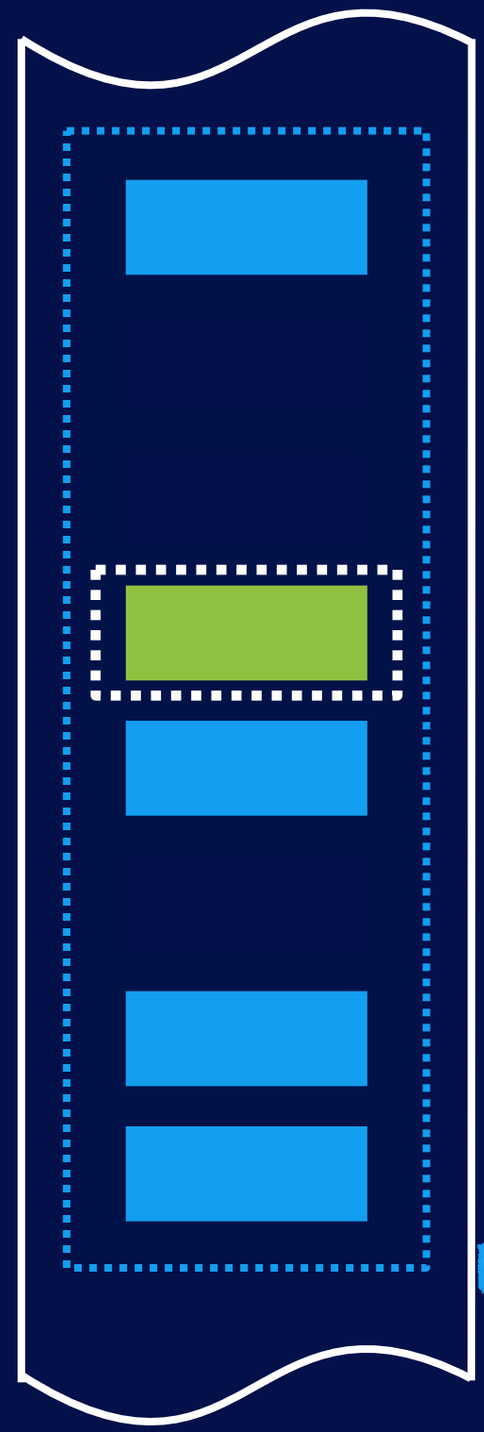
Core 2

Core 3

?

How do we split a region entry?

Memory



Region Entry

Block Entry

Core 1

Core 2

Core 3

Allow overlapping block and region entries

Block has precedence!

Memory

Region Entry

Block Entry

Core 1

Core 2

Core 3

?

Allow overlapping block and region entries

Block has precedence!

Shared blocks not tracked in region

Memory

Region Entry

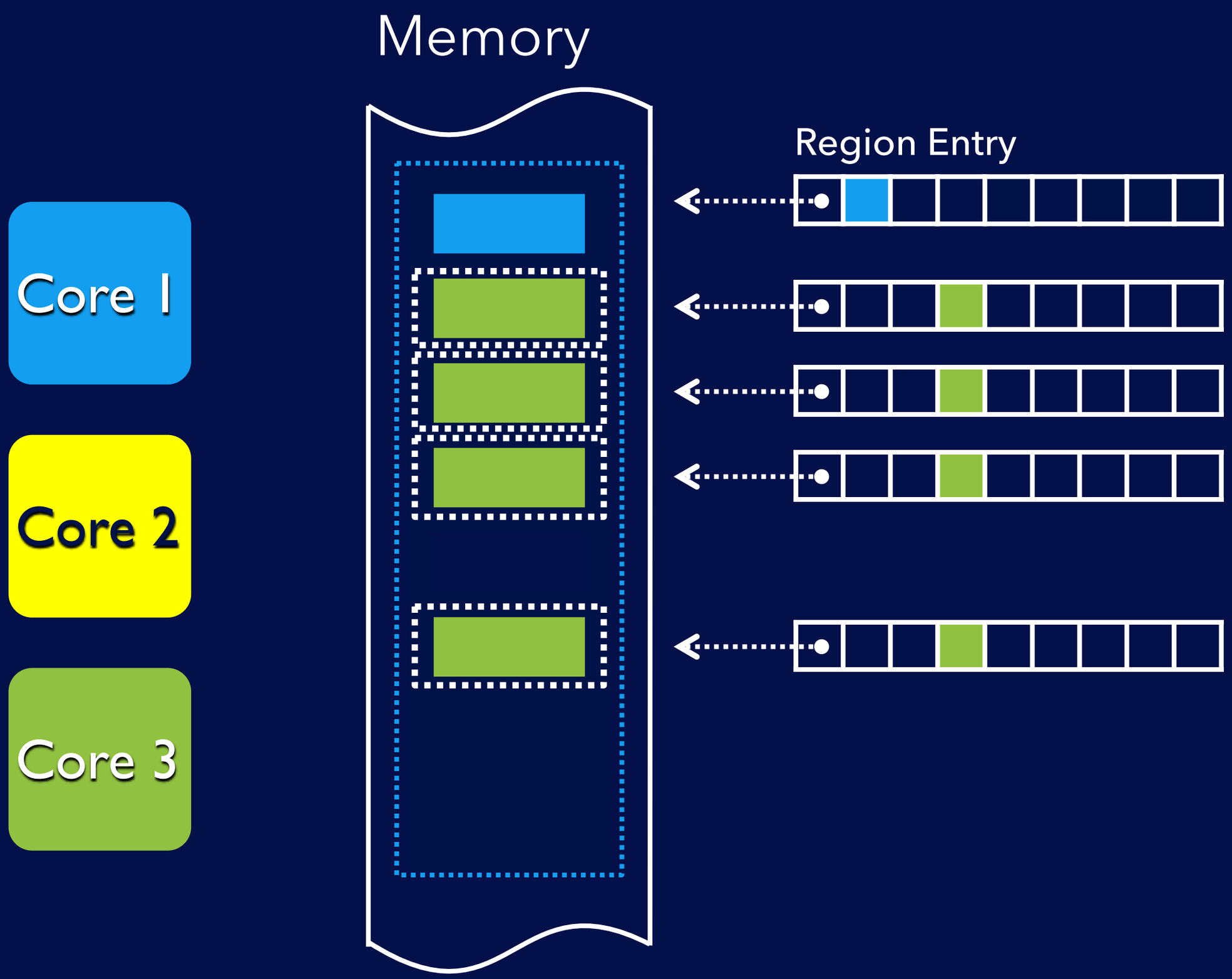
Block Entry

Block Entry

Core 1

Core 2

Core 3


Memory
Region Entry
Core 1
Core 2
Core 3

After eviction?

Core 1

Core 2

Core 3

Memory

?

Region Entry

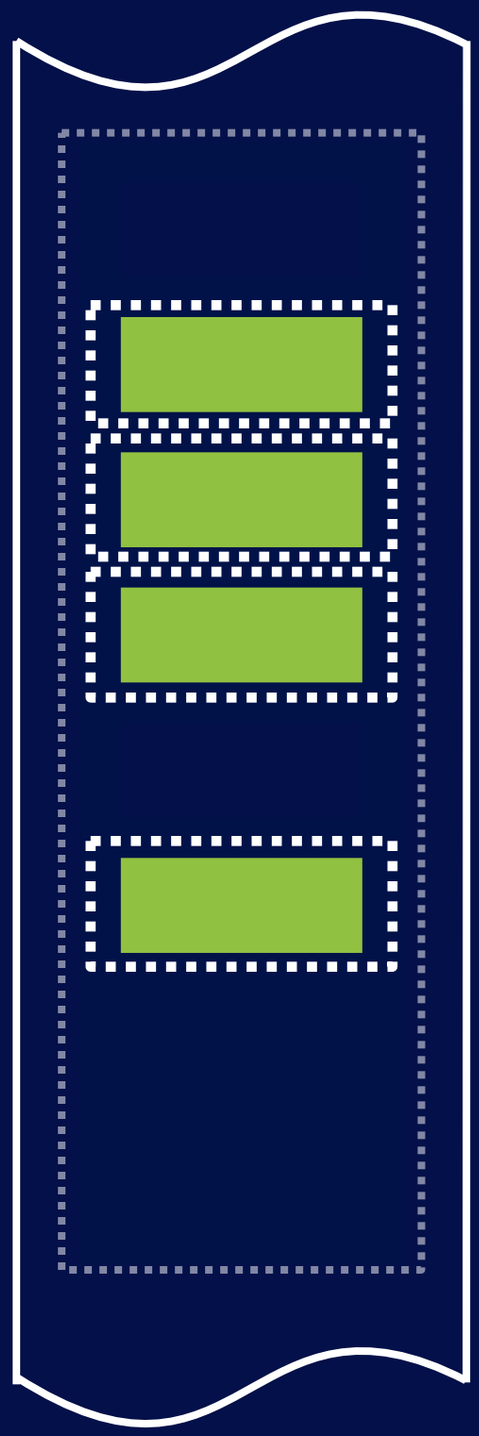
After eviction?

Core 1

Core 2

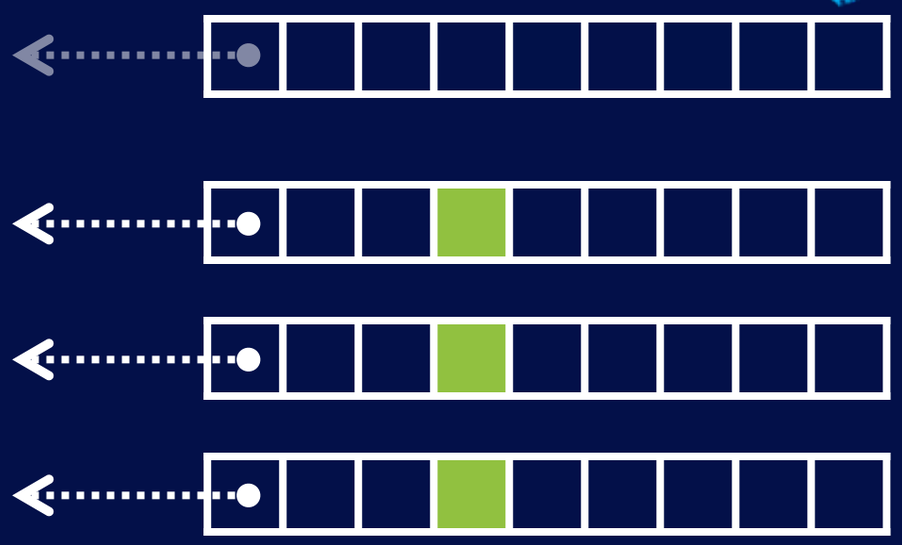
Core 3

Memory



Just invalidate region

Region Entry



Core 1

Core 2

Core 3

Memory

Re-allocation

Region Entry

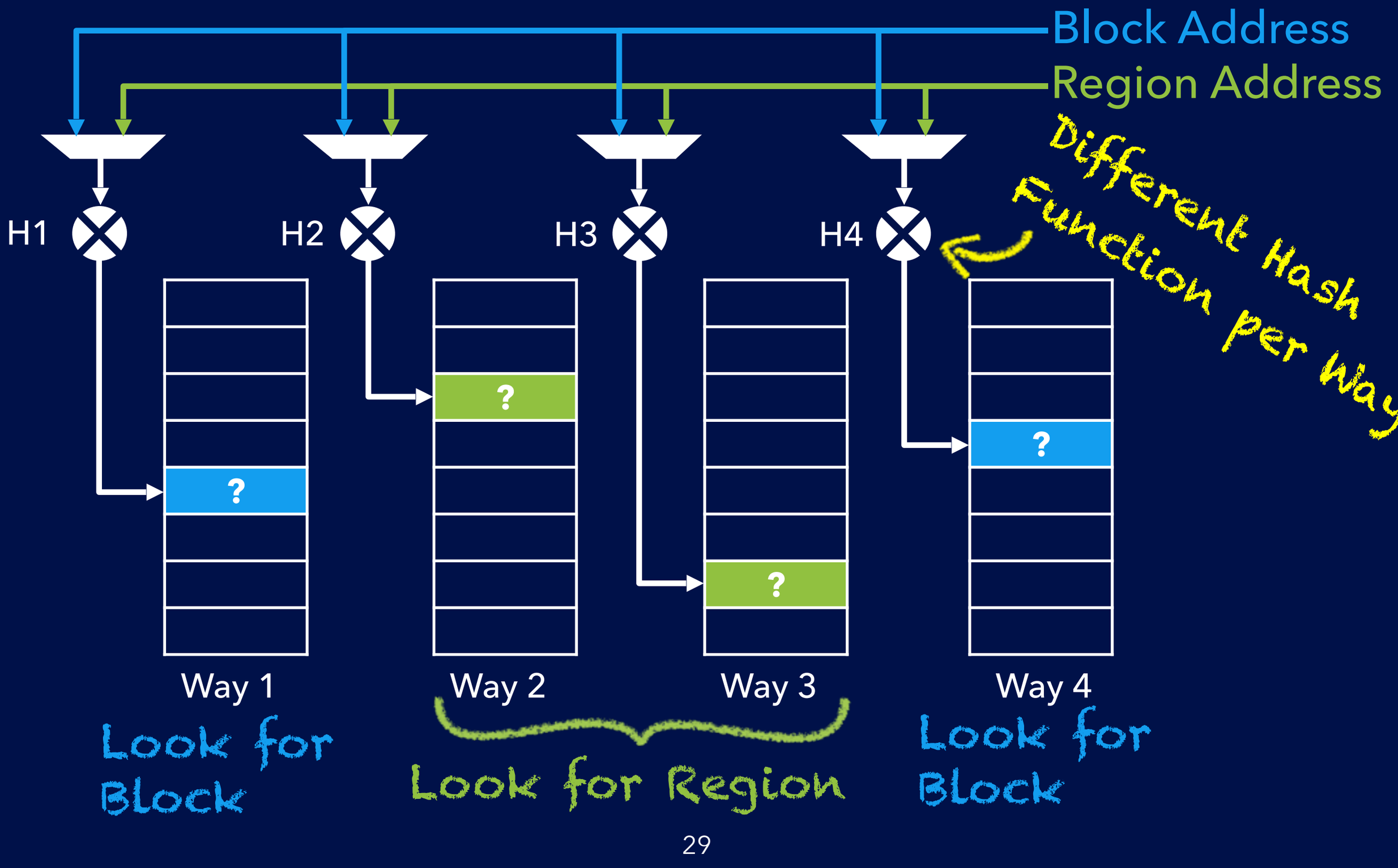
Lazy Merging!
Attempt merge if block entry gets evicted from directory.

DGD Directory Structure

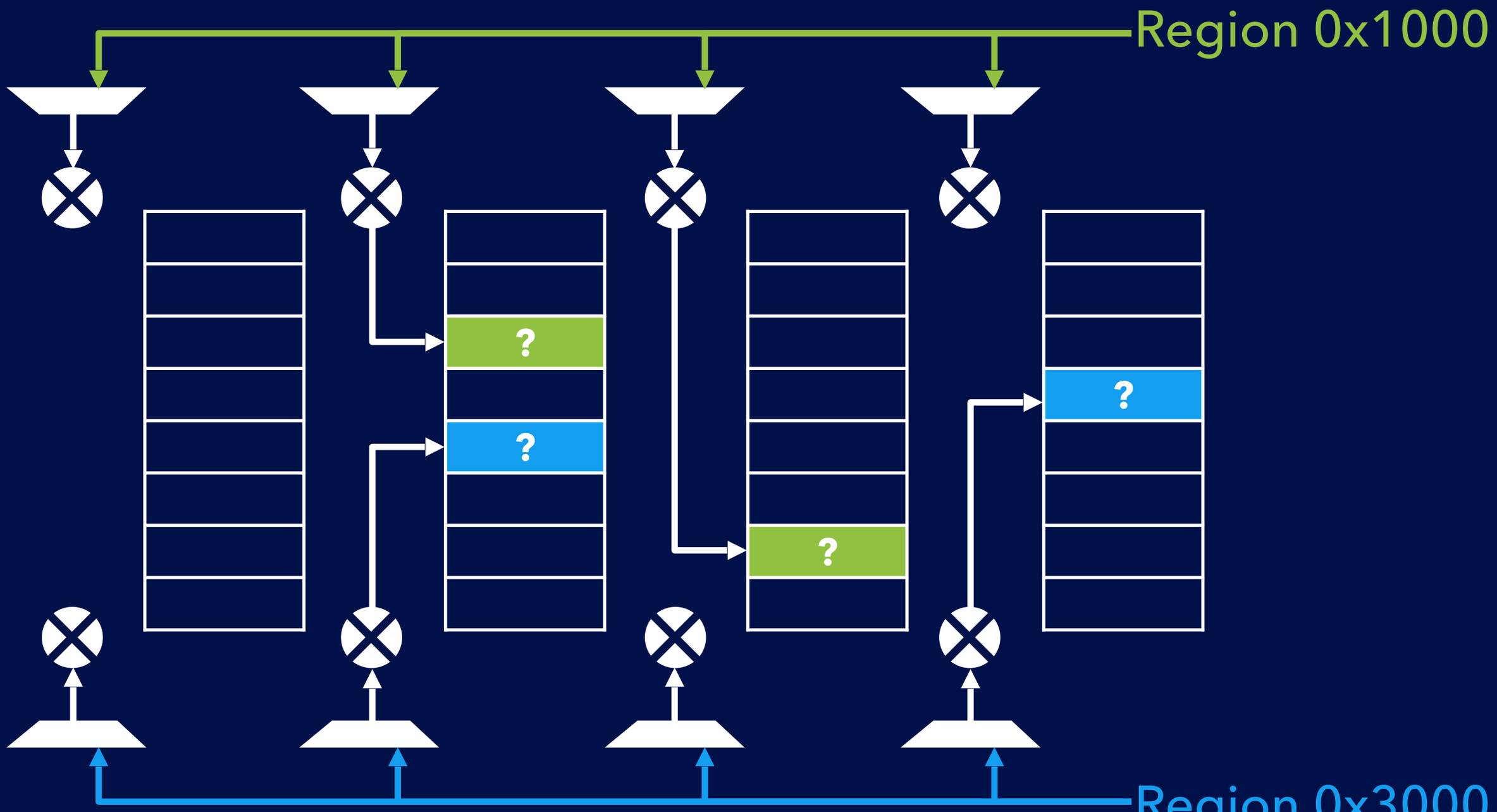
Objectives

- Single structure for block and region entries
- Find block and region entries with one lookup
  - *avoid multiple sequential lookups on critical path*
- Make lookups efficient
  - *allocation and replacement less critical*

DGD Structure

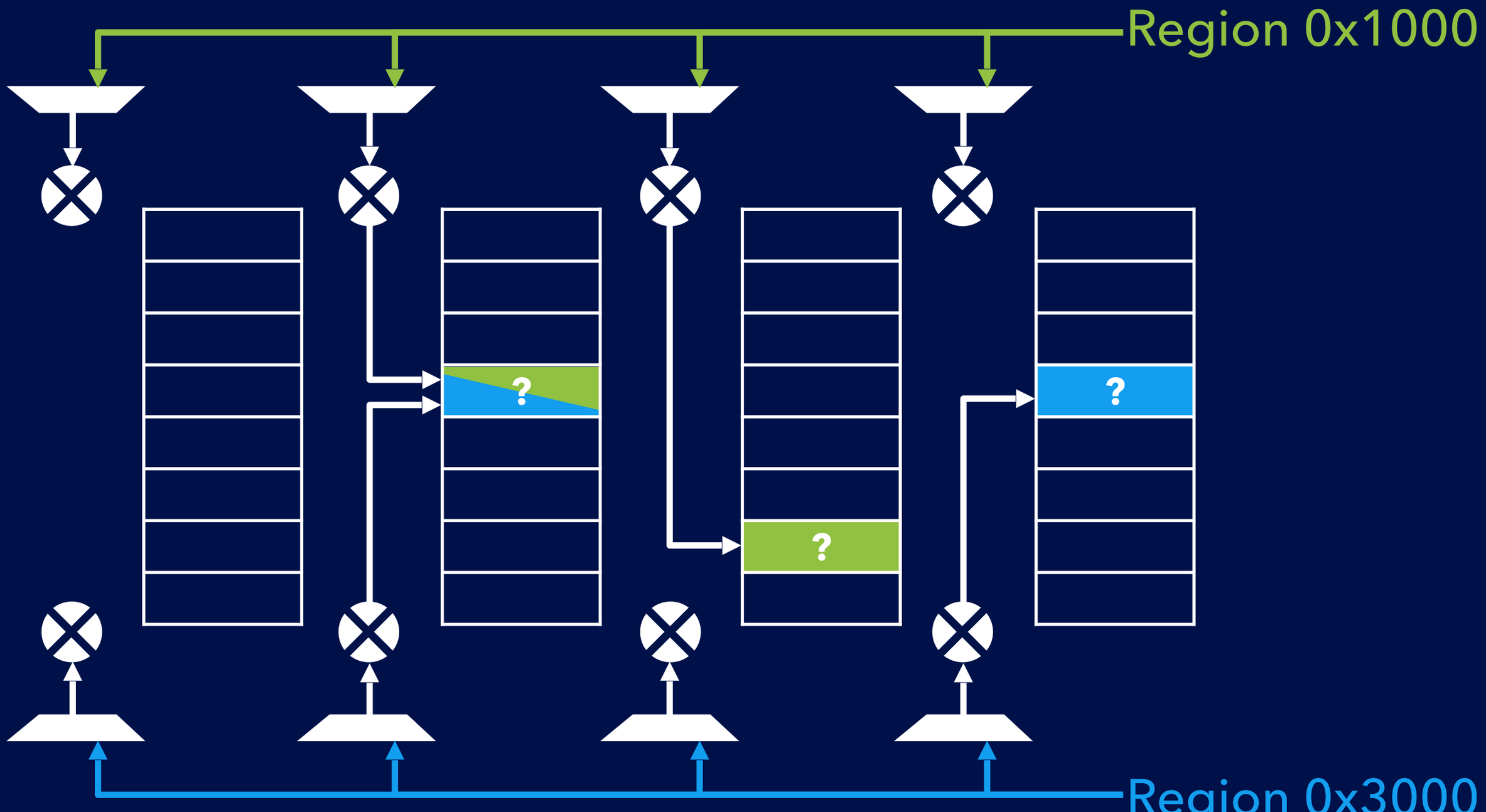


DGD Structure



Regions and blocks spread across all ways

DGD Structure



Complex options for resolving conflicts

Replacement Policy

- Skew-Associative with ZCache-style repalcement [Sanchez 2010]
  - *likely to find an unused entry*
  - *replacement can require multiple lookups*

- Equal preference to both block and region entries
  - *Invalidate cached blocks on directory eviction*
  - *Invalidated blocks allocated in shared cache if necessary*

- Not on the critical path
  - *Insertion/Eviction queue to delay actions*

- Original MGD concept
- Potential benefits of MGD
- Making a practical design
- **A recent competing design**
- Some nice graphs
- Final Thoughts

Spatiotemporal Coherence Tracking (SCT)

Mohammad Alisafaee, MICRO 2012

	DGD	SCT
Consider more than 2 granularities	✓	✗
No Protocol Changes	✓	✗
No New Race Conditions	✓	✗
Single Lookup for Each Access	✓	✗

- Original MGD concept
- Potential benefits of MGD
- Making a practical design
- A recent competing design
- **Some nice graphs**
- Final Thoughts

Simulation Details

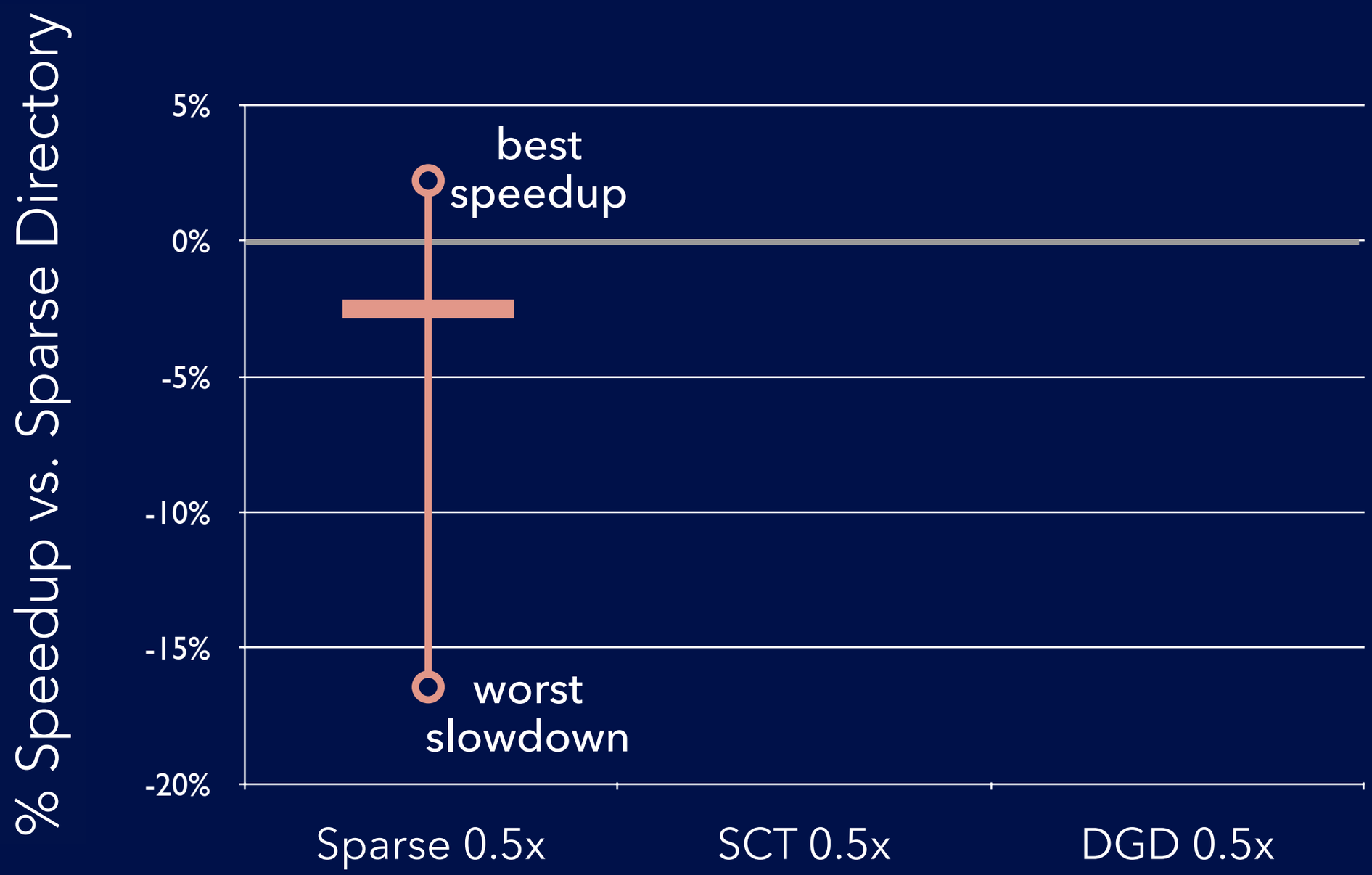
- 16-core chip-multiprocessor
- *64 KB L1 Instruction and Data caches*
- *256 KB private L2 cache*
- *16 MB distributed shared L3 cache*
- 8-way set-associative directory

- Simics + Flexus full-system simulation
- *SPARCv9 ISA, Solaris OS (version 8 or 10)*
- *Detailed timing of out-of-order cores and memory hierarchy*

- Selection of 18 diverse applications
- *SpecWeb, Parsec, TPC-C, TPC-H, CloudSuite*

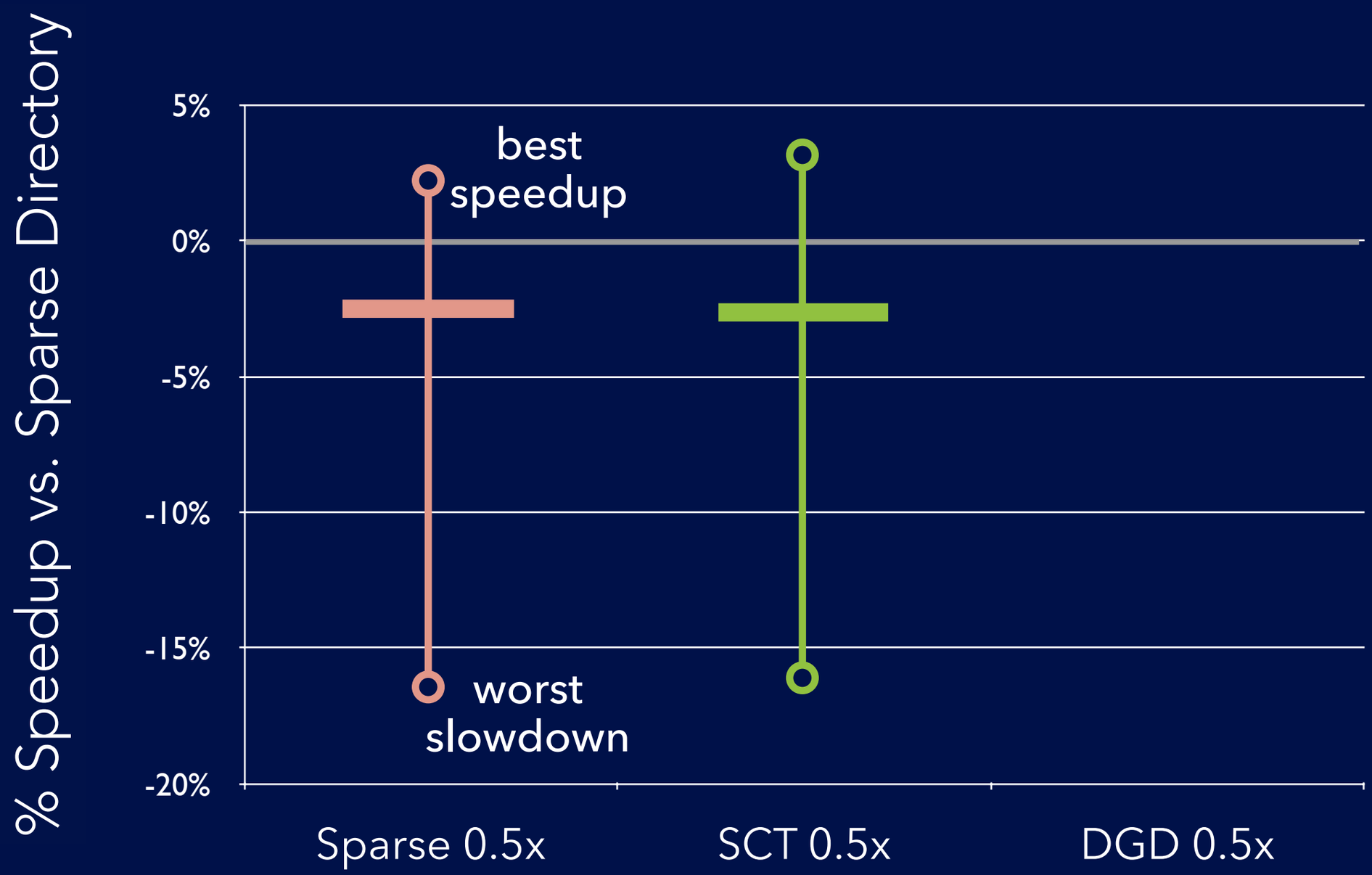
Dual-Grain Directory Performance

50% Entry Reduction vs. Baseline Directory



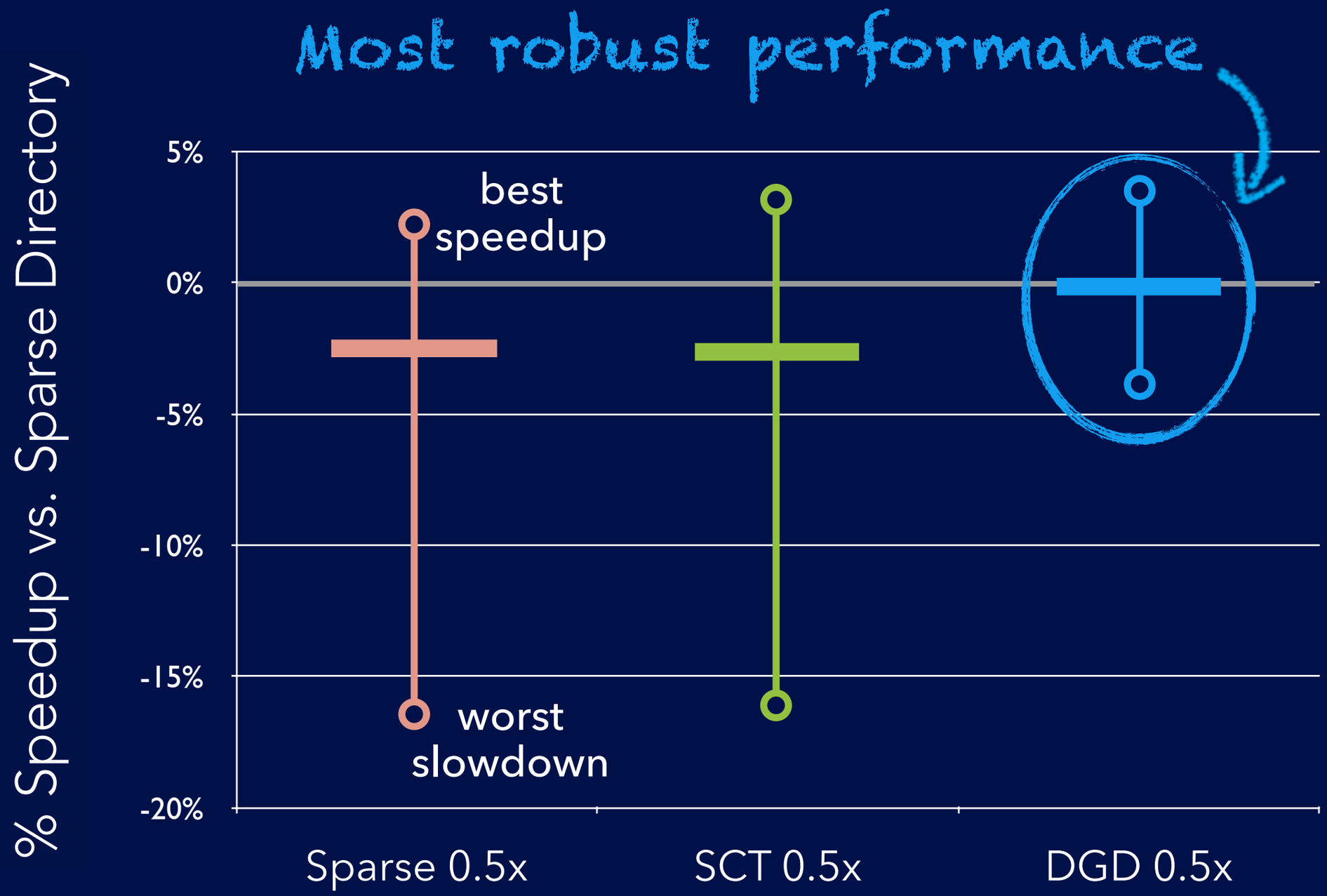
Dual-Grain Directory Performance

50% Entry Reduction vs. Baseline Directory



Dual-Grain Directory Performance

50% Entry Reduction vs. Baseline Directory



- Original MGD concept
- Potential benefits of MGD
- Making a practical design
- A recent competing design
- Some nice graphs
- **Final Thoughts**

Multi-Grain Coherence Directory

Ideal MGD Directory:

- ✓ Conceptual exploration
- ✓ Majority of benefit from just 2 granularities

Practical Dual-Grain Directory:

- ✓ Track private regions and shared blocks
- ✓ No coherence protocol changes
- ✓ **50% fewer entries**
- ✓ Robust performance, minimal losses