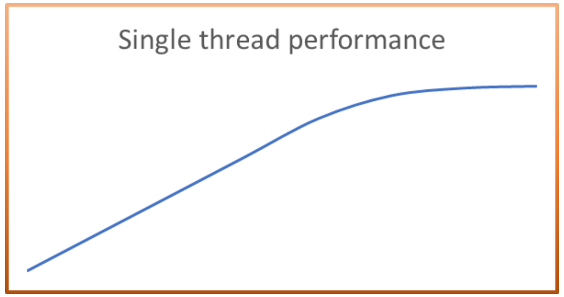
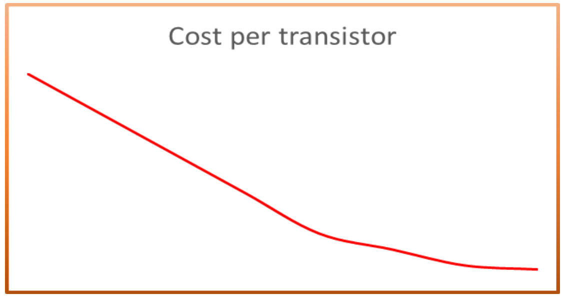
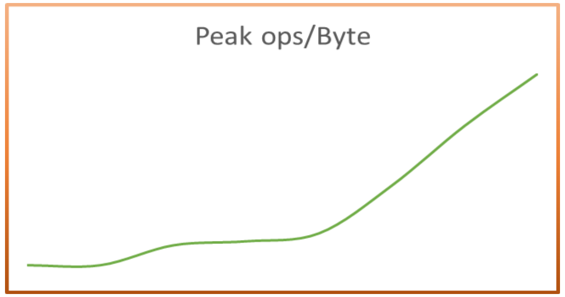
Memory Performance Optimization

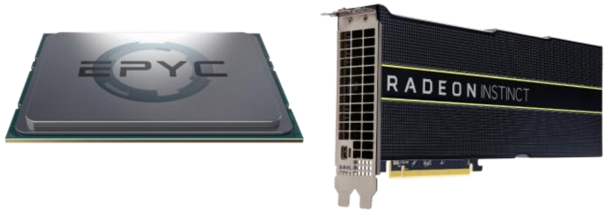
Nuwan Jayasena

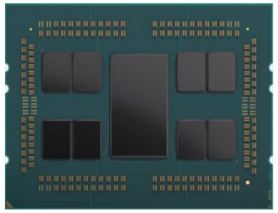
(Impending) Scaling Challenges in "Traditional" Technologies have Led to a Golden Age of Innovation

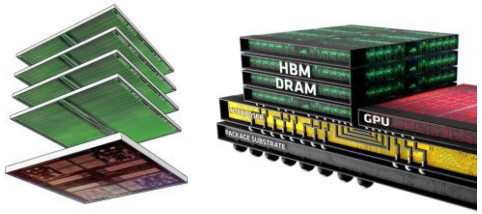












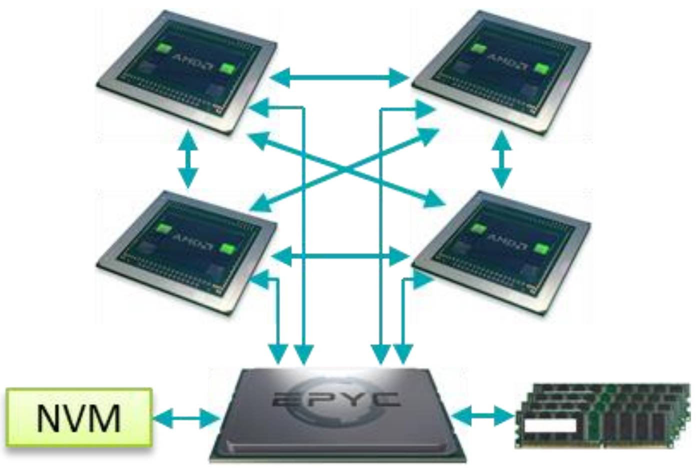
Many-core processors, accelerators etc.

Partitioned designs, advanced packaging etc.

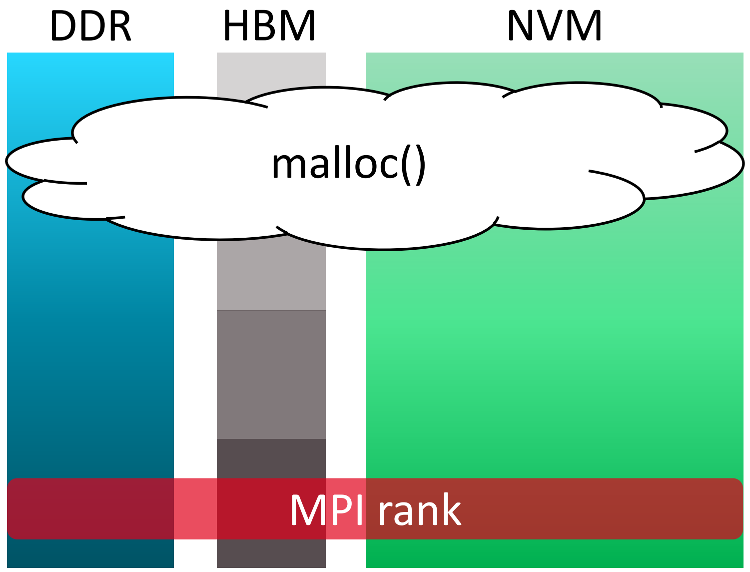
In-package memory, processing near memory etc.

AMD

Advanced Node Designs Challenge Traditional Views of Memory

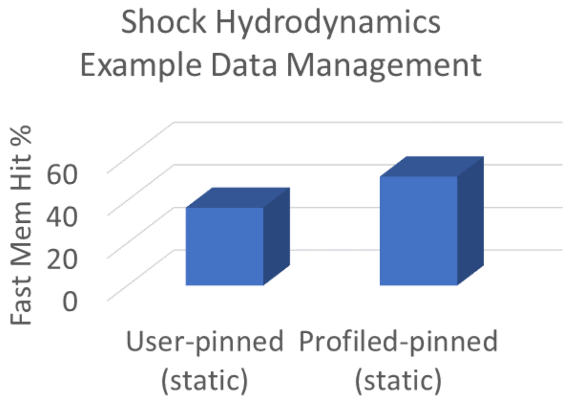


Node Organization

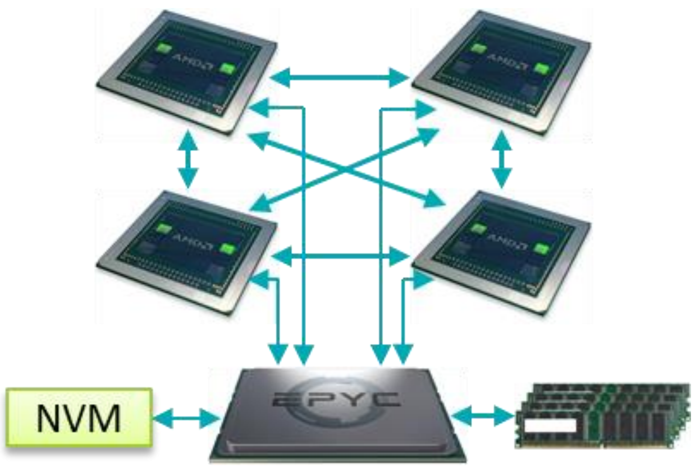


Memory View

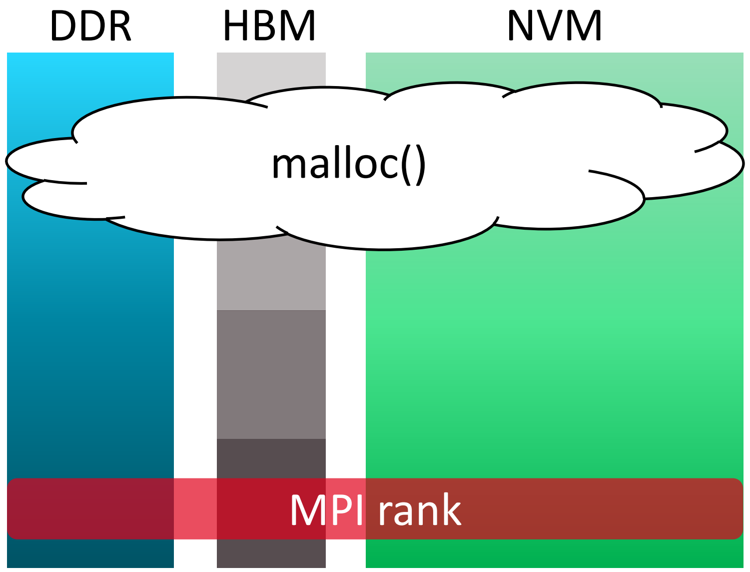
- Intra-node non-uniform memory pools
  - Exacerbated by growing application memory capacity needs
  - Increased reliance on efficient use of caches and interconnects
- Manually choosing memory pools is often challenging for developers
  - Static placement can be sub-optimal
  - Also hampers code portability



Advanced Node Designs Challenge Traditional Views of Memory

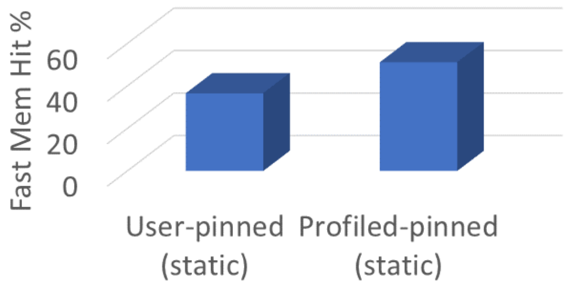


Node Organization



Memory View

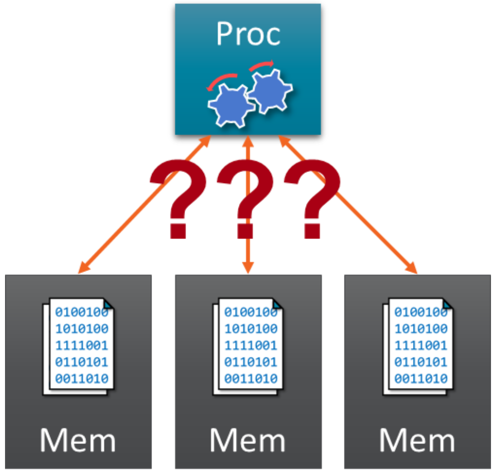
- Intra-node non-uniform memory pools
  - Exacerbated by growing application memory capacity needs
  - Increased reliance on efficient use of caches and interconnects
- Manually choosing memory pools is often challenging for developers
  - Static placement can be sub-optimal
  - Also hampers code portability



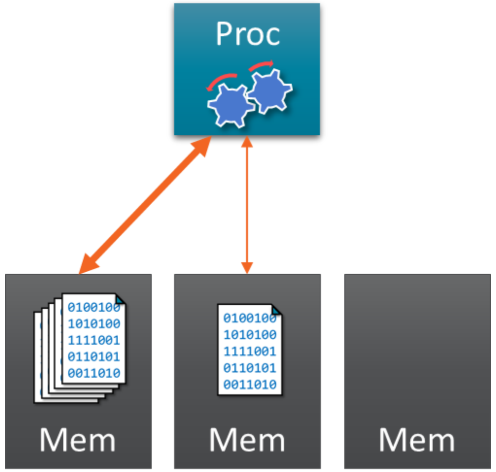
Innovations in data management are necessary to realize full benefits of aggressive node organizations

Locality Management Requires Hardware-Software Collaborative Solutions

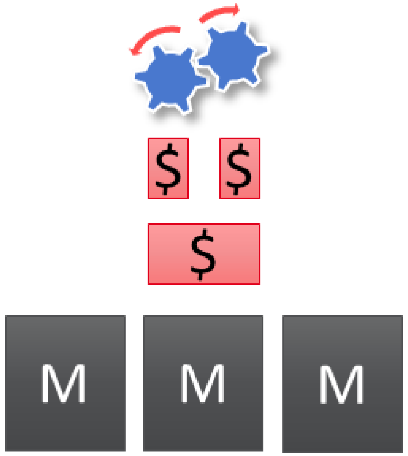
Understanding Data-Compute Affinity



Enabling Software-guided Data Placement Control



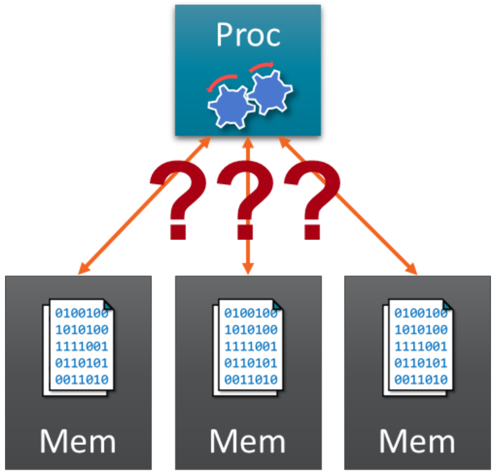
Memory-aware Data Structures and Algorithms



AMD

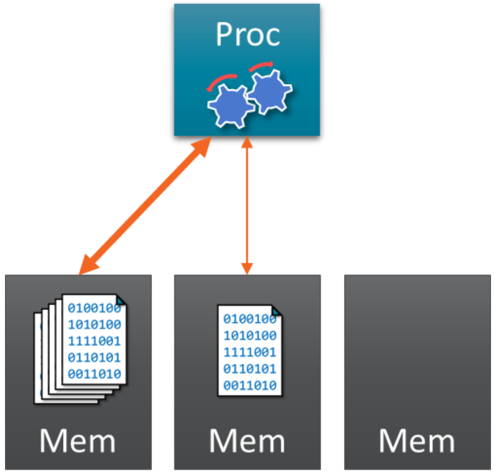
Locality Management Requires Hardware-Software Collaborative Solutions

Understanding Data-Compute Affinity

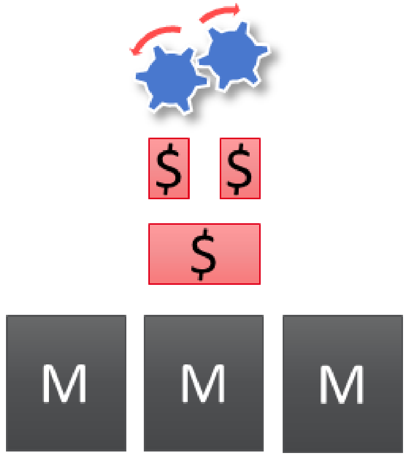


E.g., Memory access tracking with *Majority Element Algorithm*

Enabling Software-guided Data Placement Control



Memory-aware Data Structures and Algorithms



E.g., *Morton Filter* approximate set membership data structure

POINT SOLUTION EXAMPLES

Memory Access Tracking: Identifying Frequently Accessed Pages

- Software-only solution: periodically sample “accessed” bits in page table entries
  - Coarse-grain, statistical information only

- Naïve hardware solution: access counter associated with each OS page
  - Incurs storage for counters and overhead for updating them
  - System software has to periodically scan the counts and find frequently-accessed pages (usually by sorting)

- Wide range of solutions proposed in literature

- An elegant approach: majority element algorithm (MEA) [1][2] from data mining
  - Heuristic for finding most frequently occurring elements in a data stream
  - More precisely, finds a set of $\lfloor 1/\theta \rfloor$ elements that includes all elements that occur θN or more times
    - Where $0 > \theta > 1$ and N is the length of the data stream
    - E.g., if $\theta = 0.05$ & $N = 100$, finds $1/0.05 = 20$ elements that includes all elements that occur $0.05\text{x}100 = 5$ times or more

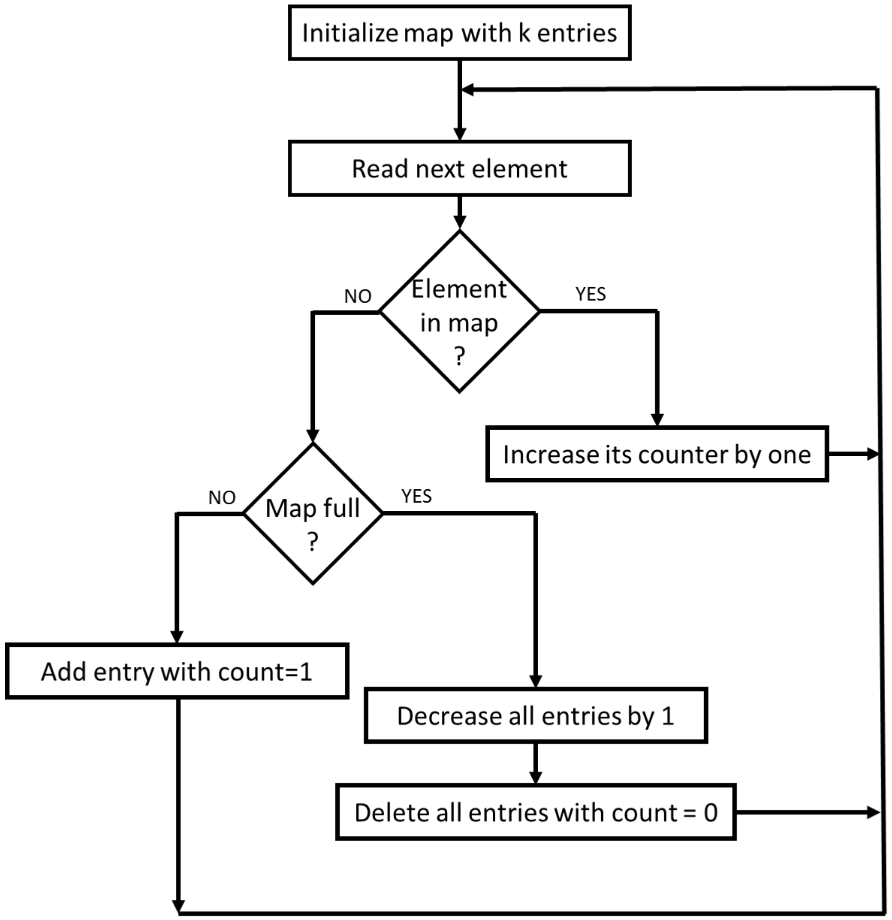
[1] R.M. Karp and S. Shenker, ACM TODS, vol. 28, no. 1.

[2] M. Charikar *et al.*, Theoretical Computer Science, vol. 312, no. 1.

Applying MEA for Memory Affinity Tracking

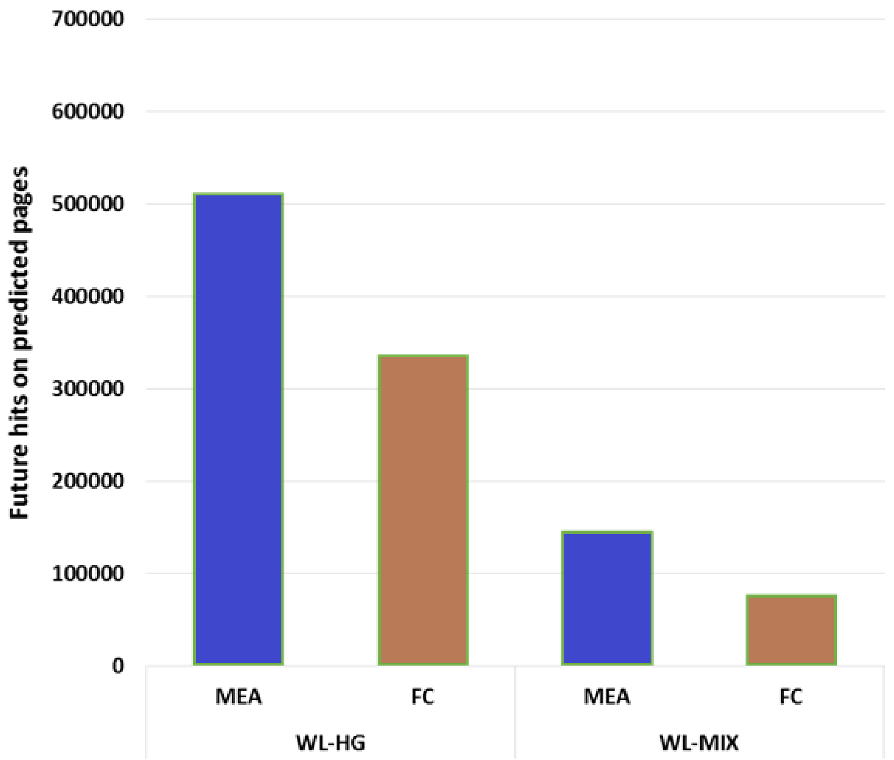
- Treat the sequence of pages accessed by a processor as a data stream
- Find the most frequently occurring page numbers in that stream

- Simple and low-overhead hardware implementation
  - Only as many counters (K) as the number of top memory pages to be discovered
  - No separate pass to sort and find frequently accessed pages



MEA: A Free Lunch that You Can Have and Eat Too!

- Recency bias: MEA makes better predictions of future memory accesses than (an impractical) approach that counts accesses to every OS page



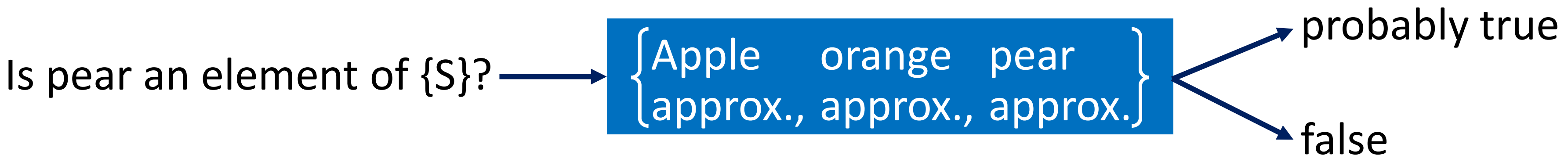
FC: full counters (counter per OS page) with full sorting and selection

WL-HG: homogeneous workloads

WL-MIX: mixed workloads

See Prodromou *et al.*, HPCA 2017

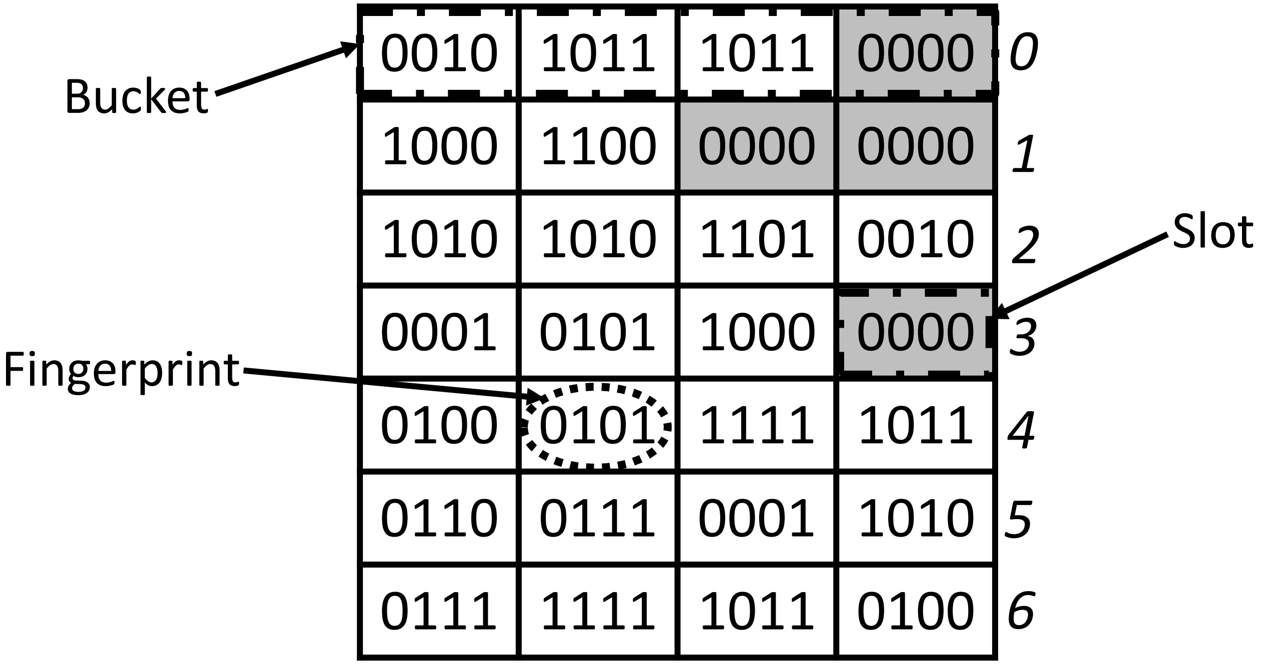
Memory-aware Data Structures: Approximate Set Membership



- Approximate set membership data structures (ASMDSs) trade precision for space
  - Tunable false positive rate ϵ where increasing the bits per item reduces ϵ
  - ϵ is dependent on the bits per item in the approximation not the original data size
    - E.g., encoding 4B items and 200MB items takes the same amount of space for a given ϵ

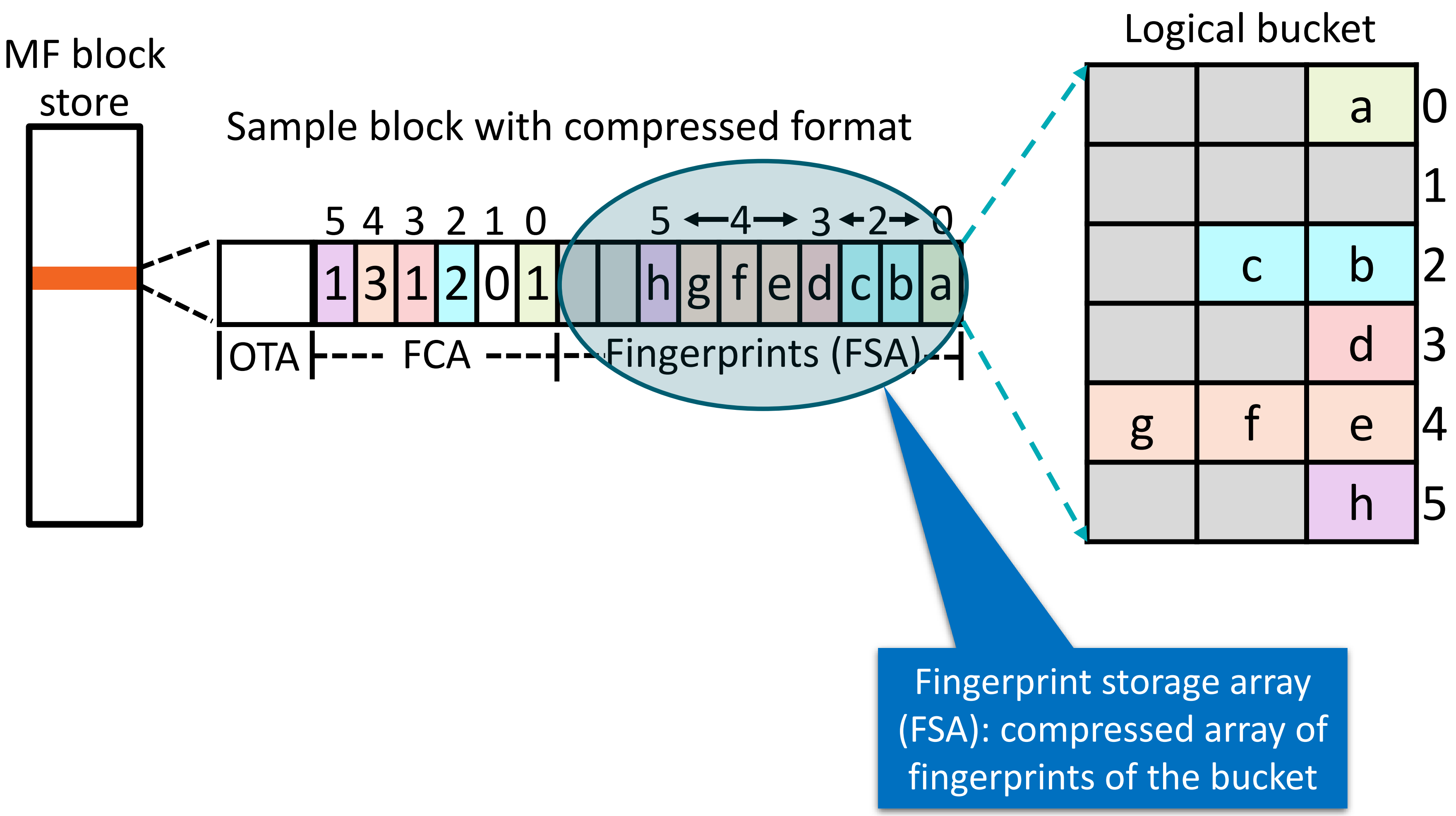
- Wide-ranging usage
  - Genome sequencing, relational databases, file systems, key-value stores, web caching, networking etc.

Background: Cuckoo Filters

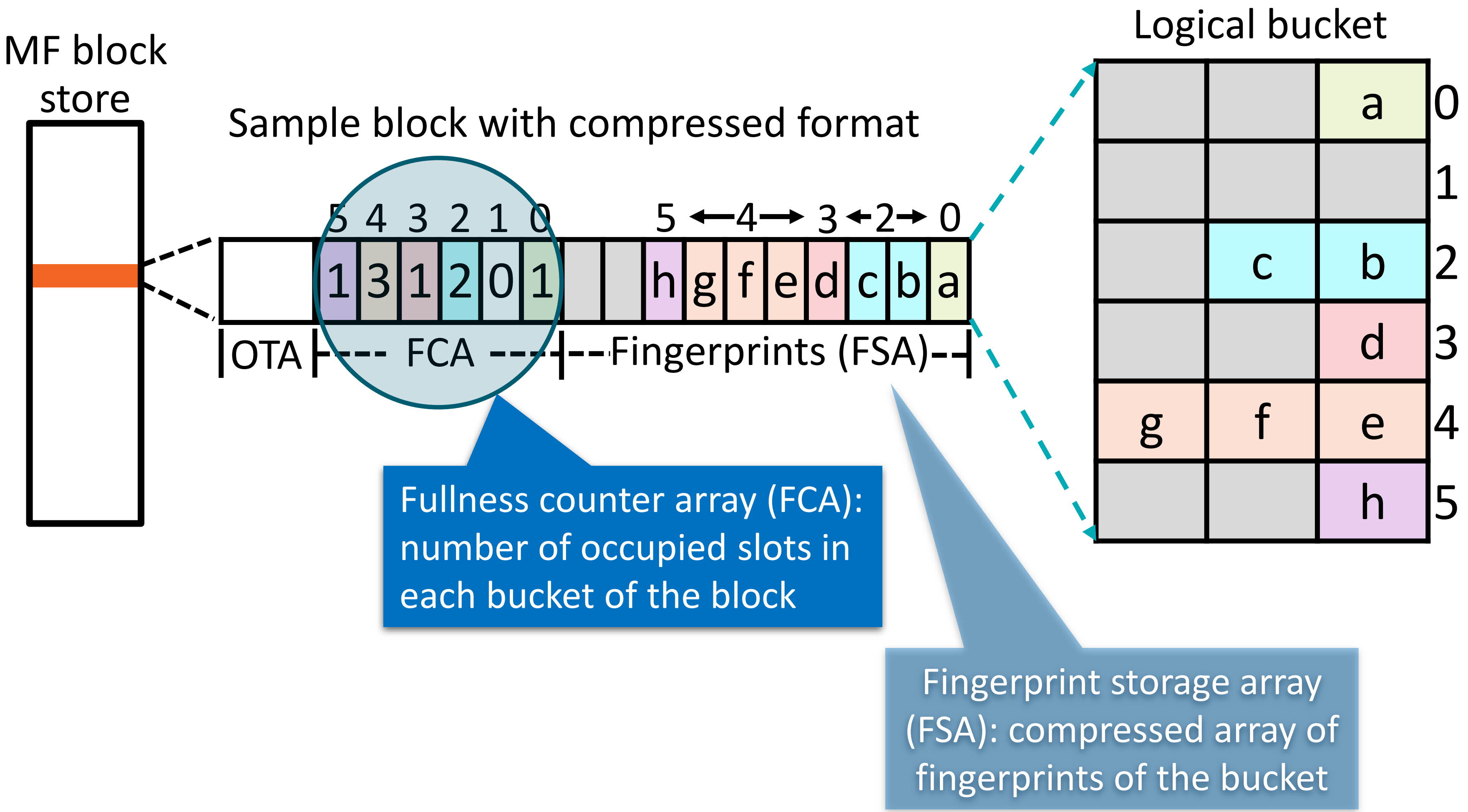


- Buckets are associative collections of slots (similar to cuckoo hash tables)
- Each slot stores a “fingerprint”, a short hash of a single data element
- Multiple (typically 2) hash functions to map data items to candidate buckets

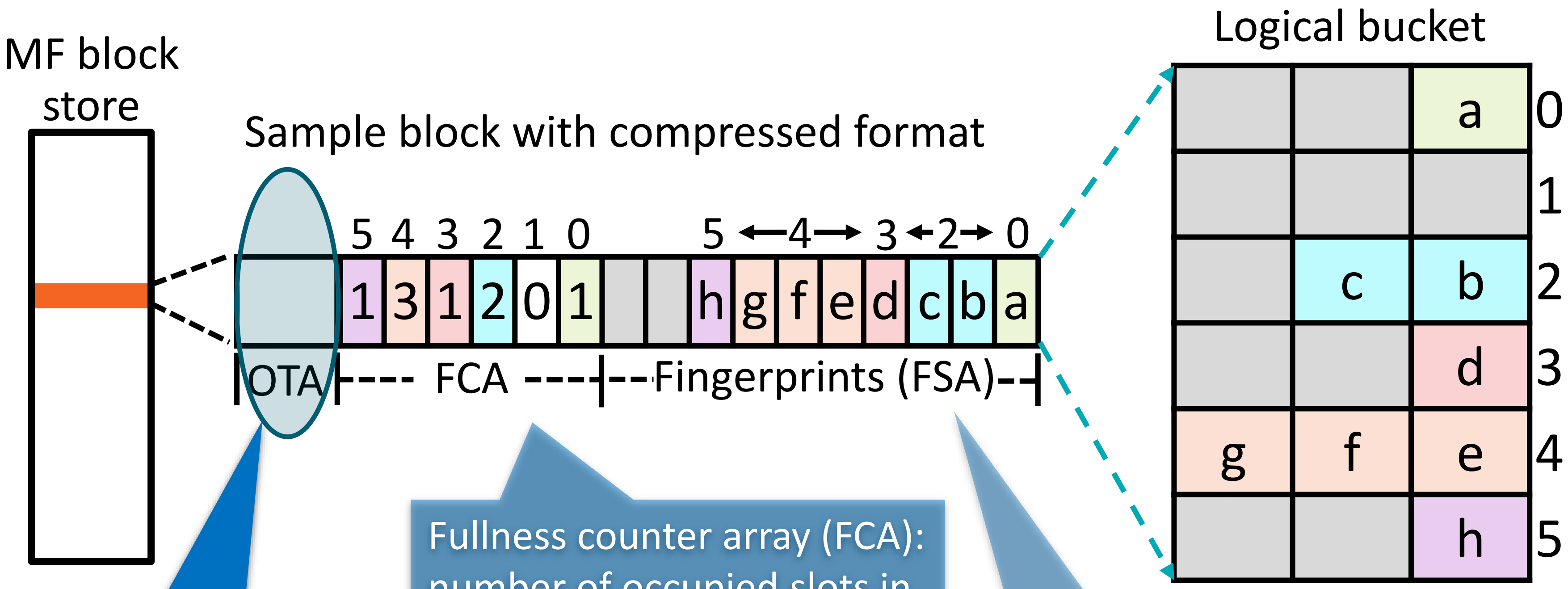
Morton Filters: Decoupling Memory Storage from Logical Structure



Morton Filters: Decoupling Memory Storage from Logical Structure



Morton Filters: Decoupling Memory Storage from Logical Structure



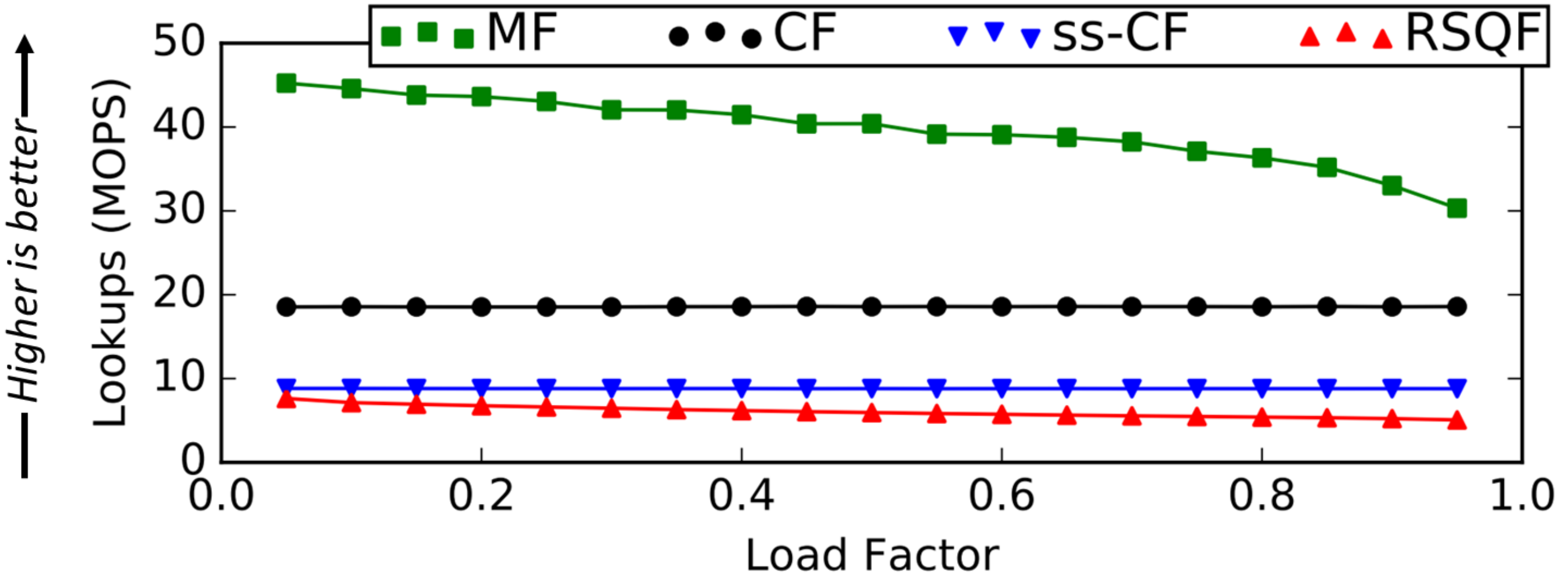
Fullness counter array (FCA): number of occupied slots in each bucket of the block

Overflow tracking array (OTA): bit-vector indicating primary bucket overflows

Fingerprint storage array (FSA): compressed array of fingerprints of the bucket

AMD

Memory-centric Optimizations Provide Significant Benefits



MF: Morton Filter

CF: Cuckoo Filter

ss-CF: Semi-sorting Cuckoo Filter

RSQF: Rank and Select Quotient Filter

Morton Filters outperform state-of-the art ASMDSs by > 1.6x-2.4x for positive lookups at similar false positive rates

Additional Metrics	Morton Filters Improvement over CF
Negative Lookup Throughput	1.3x to 2.5x
Insertion Throughput	0.9x to 15.5x
Deletion Throughput	1.3x to 1.6x

See Breslow & Jayasena, VLDB 2018

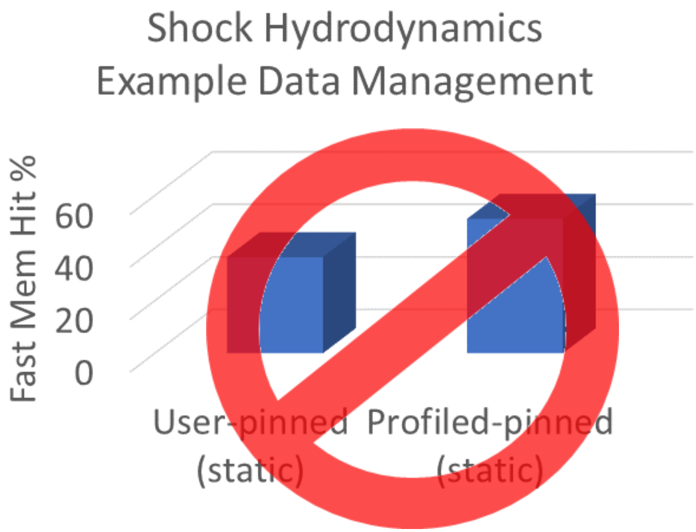
AMD

CALL FOR GENERALIZED SOLUTIONS

Capture Application Knowledge About Data Access Behaviors

- Need APIs to capture domain/application knowledge about data access behaviors
  - Along with better access to high-level hardware metrics

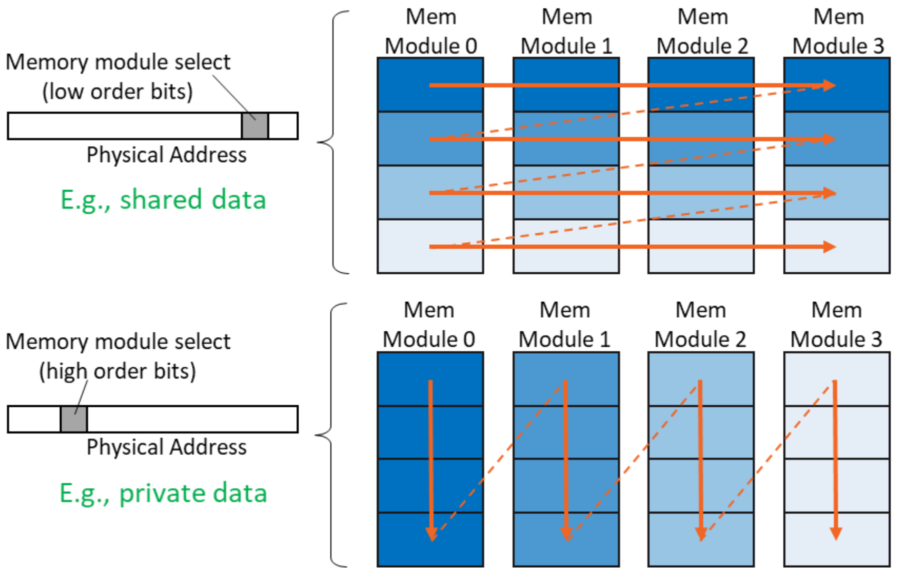
- Which data structures are likely to have frequent reuse?
  - Can we be even more nuanced and capture reuse distance expectations?
  - More portable than dictating which cache levels to bypass
- Which data has predictable access patterns? Which ones don't?
  - Can drive allocation, placement, and access optimizations
  - More general than prefetch hints
- Which data structures have high spatial locality?
  - Which ones have dense access patterns?
  - What is the degree of sparsity?
- Which data structures are accessed concurrently?
  - Can we reduce or eliminate interference in memory accesses?



AMD

Need System Capabilities to Exploit Domain Knowledge

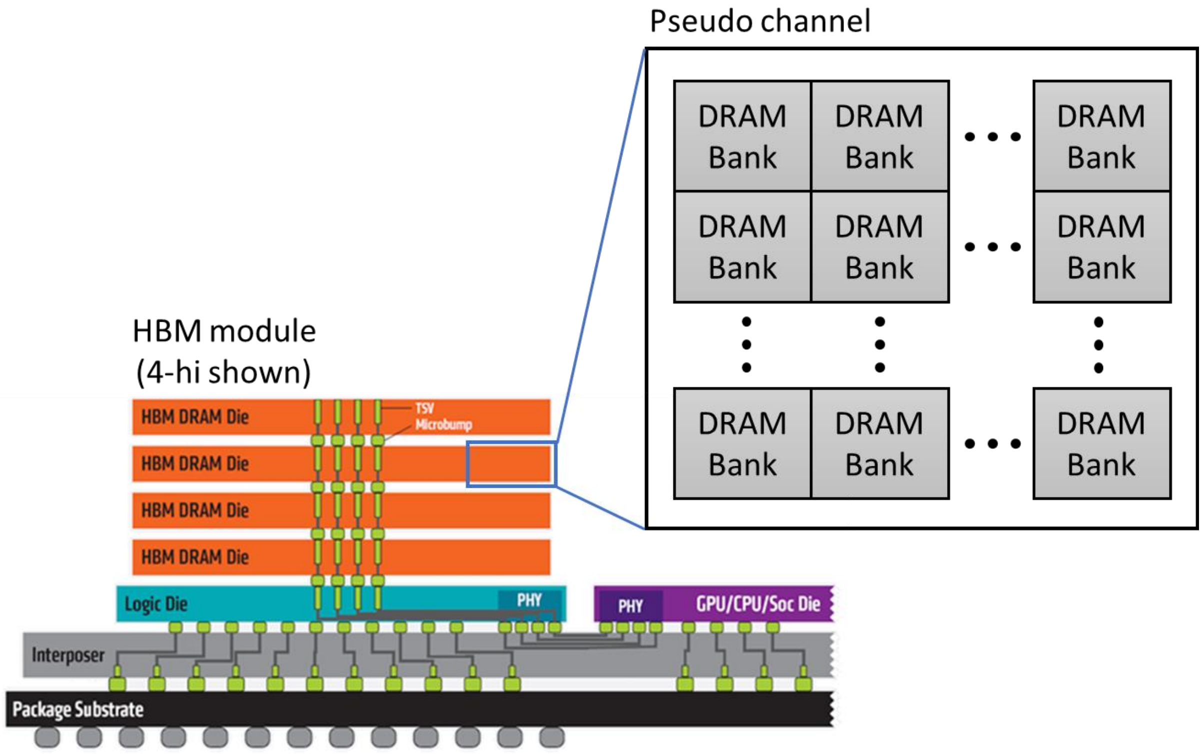
- System software and hardware intelligence and capabilities to exploit domain/application information
- Different memory pools within the same memories with different characteristics
  - Different degrees of cacheability, prefetch aggressiveness etc.
  - Optimized for different access patterns
    - Different OS page sizes
    - Tailored distribution granularity among memory channels, banks, etc.
- Placement of concurrently accessed data structures to reduce interference
  - Partition groups of memory channels, banks etc.



AMD

Making Software (More) Aware of Memory Organization

- Memory pool awareness is already available up in practical forms
- Advanced node architectures can benefit from software awareness of more detailed memory organization characteristics
  - Help hardware memory schedulers
  - Reduce hot spots and distribute traffic in hardware
  - Reduce hardware cost of over-designing memory system
- Query hardware memory organization at boot time (firmware)
- Data layout optimizations at allocation time (system software)
- Data placement and addressing optimizations at run-time (system software and hardware)
  - Hardware support for software distribution of accesses across memory channels



Summary

- Hardware scaling trends are driving an explosion of new solutions
  - Many of which introduce new memory considerations
- It’s time for software to take a more active role in optimizing memory performance
- Specific point solutions have demonstrated significant solutions
- Major research opportunity for hardware-software collaborative solutions for data management with potentially huge payoffs

DISCLAIMER AND ATTRIBUTIONS

Disclaimer:

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD