Efficient Techniques for Sharing On-chip Resources in CMPs

Ruisheng Wang

PhD Oral Defense

2017-05-09

amazon web services

CRM as a Service

Google Cloud Platform

salesforce

IBM Cloud

Maching Learning as a Service

Storage as a Service

ORACLE CLOUD

Neural Network Learning

vmware

Dropbox

Database as a Service

openstack CLOUD SOFTWARE

Functions as a Service

Payments as a Service

rackspace

Email as a Service

Microsoft Azure

f

twitter

SendGrid

amazon web services

CRM as a Service

salesforce

Google Cloud Platform

IBM Cloud

Machine Learning as a Service

Storage as a Service

ORACLE CLOUD

> "Overall cloud workloads will more than triple from 2015 to 2020."
>
> Cisco Global Cloud Index

Database as a Service

Neural Network Learning

vmware

Dropbox

openstack CLOUD SOFTWARE

rackspace

Functions as a Service

Payments as a Service

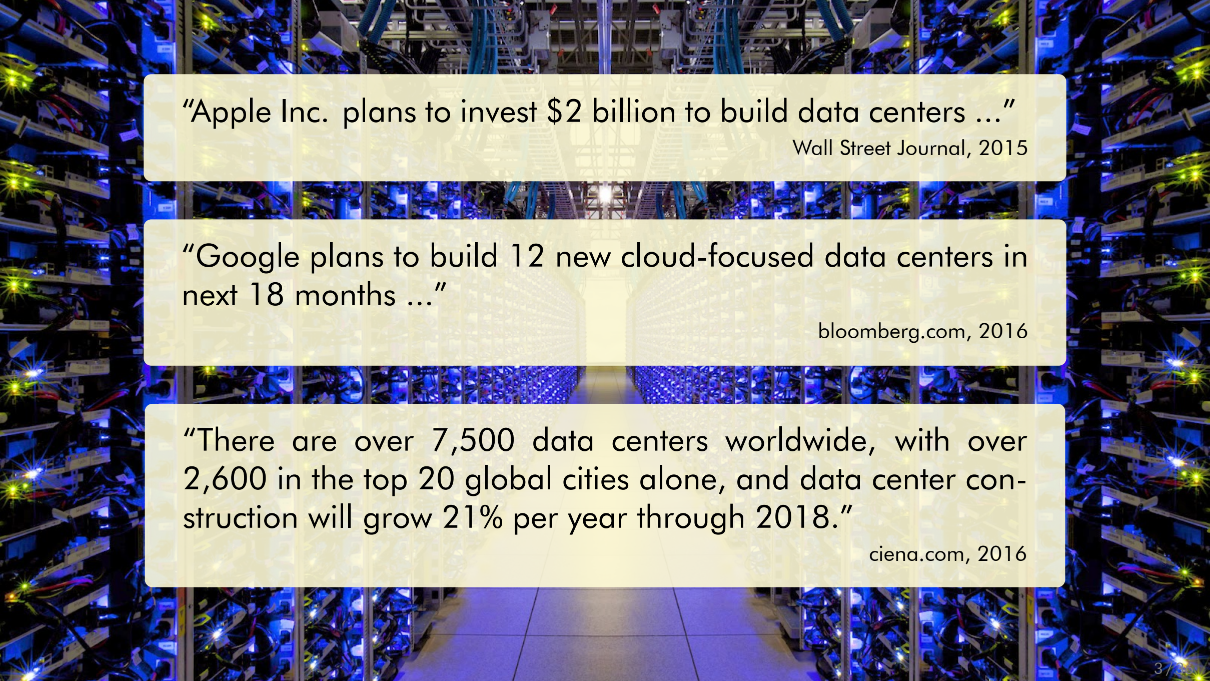
Email as a Service

Microsoft Azure

f

twitter

SendGrid



"Apple Inc. plans to invest $2 billion to build data centers ..."

Wall Street Journal, 2015

"Google plans to build 12 new cloud-focused data centers in next 18 months ..."

bloomberg.com, 2016

"There are over 7,500 data centers worldwide, with over 2,600 in the top 20 global cities alone, and data center construction will grow 21% per year through 2018."

ciena.com, 2016

> “Various analyses estimate industry-wide utilization is between 6% and12%.”

“Reconciling High Server Utilization and Sub-millisecond Quality-of-Service” by Jacob Leverich and Christos Kozyrakis, 2014

> “Such WSCs tend to have relatively low average utilization, spending most of (their) time in the 10%–50% CPU utilization range.”

“Data Center as a Computer” by Luiz Andre Barroso, Jimmy Clidaras, and Urs Holzle, 2013

"Various analyses estimate industry-wide utilization is between 6% and12%."

"Reconciling High Server Utilization and Sub-millisecond Quality-of-Service" by Jacob Leverich and Christos Kozyrakis, 2014

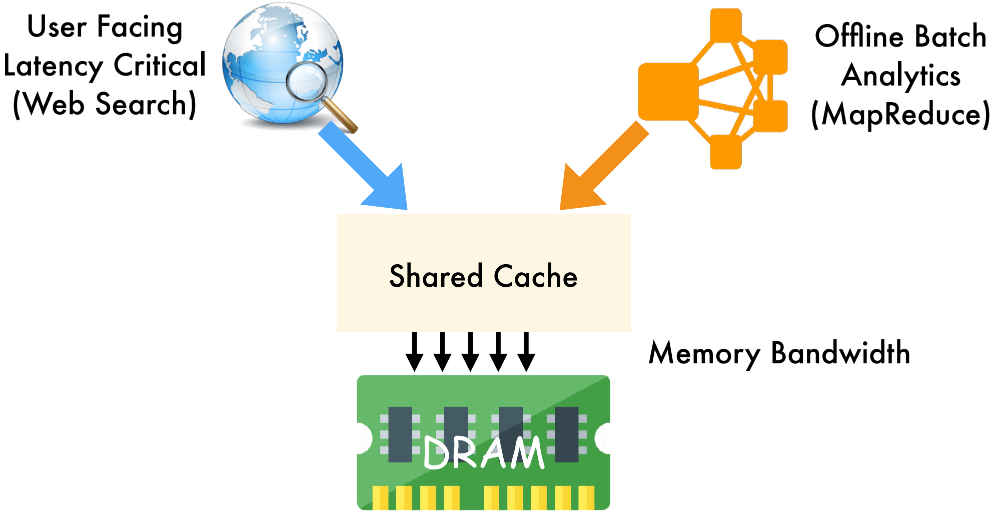
"Such WSCs tend to have relatively low average utilization, spending most of (their) time in the 10%–50% CPU utilization range."

"Data Center as a Computer" by Luiz Andre Barroso, Jimmy Clidaras, and Urs Holzle, 2013

Overprovisioning!!!

Workload Interference on Shared On-Chip Resources

Resource Interference (Uncontrolled Sharing)



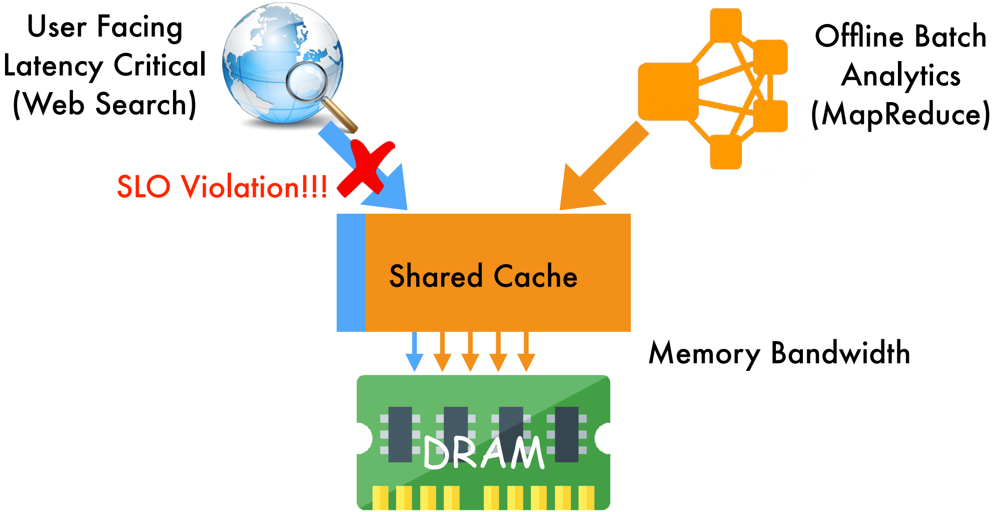
User Facing
Latency Critical
(Web Search)

Offline Batch
Analytics
(MapReduce)

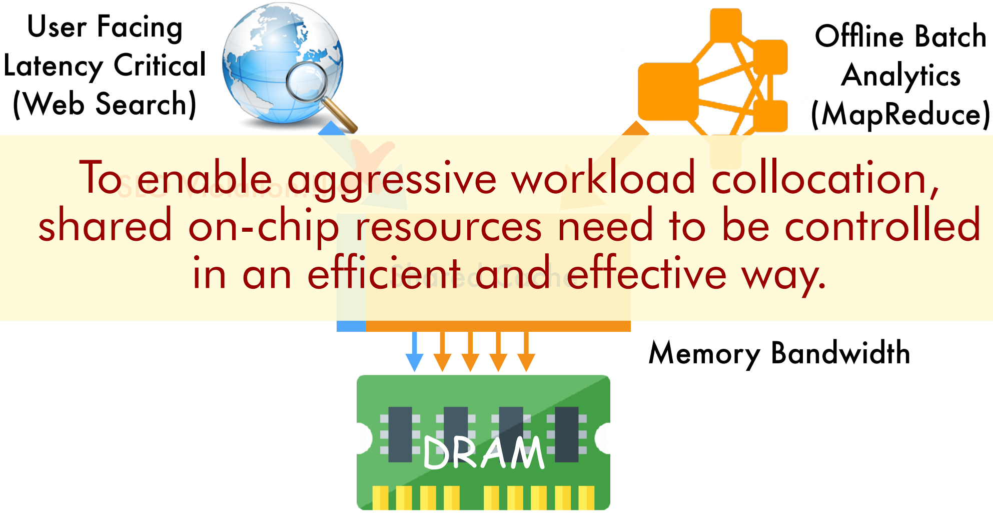
Shared Cache

Memory Bandwidth

Resource Interference (Uncontrolled Sharing)



Resource Interference (Uncontrolled Sharing)



User Facing
Latency Critical
(Web Search)

Offline Batch
Analytics
(MapReduce)

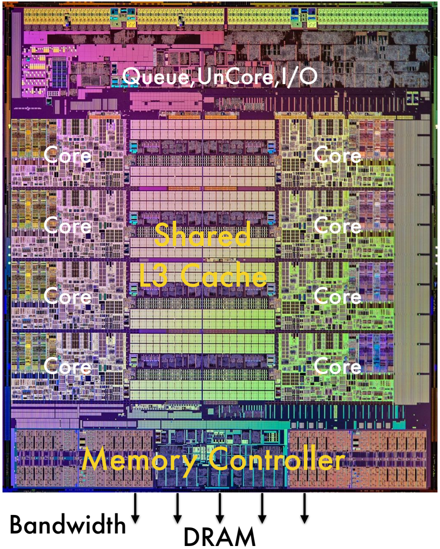
Memory Bandwidth

To enable aggressive workload collocation, shared on-chip resources need to be controlled in an efficient and effective way.

Shared On-chip Resources

Last-Level Cache

- Partitioning-induced associativity loss
- Unpredictable miss rate curve



Intel Core i7-5960X

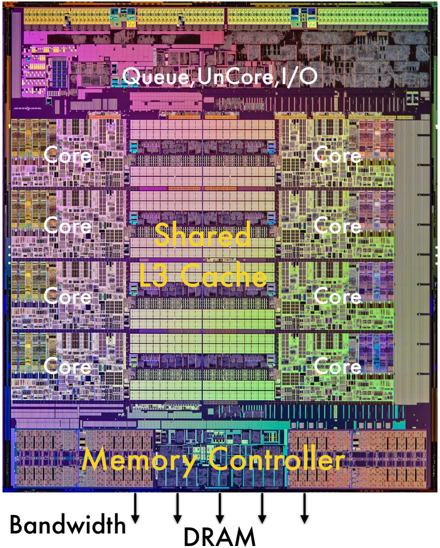
Shared On-chip Resources

Last-Level Cache

- Partitioning-induced associativity loss
- Unpredictable miss rate curve

Off-Chip Memory Bandwidth

- Unfair/Unreasonable memory bandwidth allocation



Intel Core i7-5960X

Shared On-chip Resources

Last-Level Cache

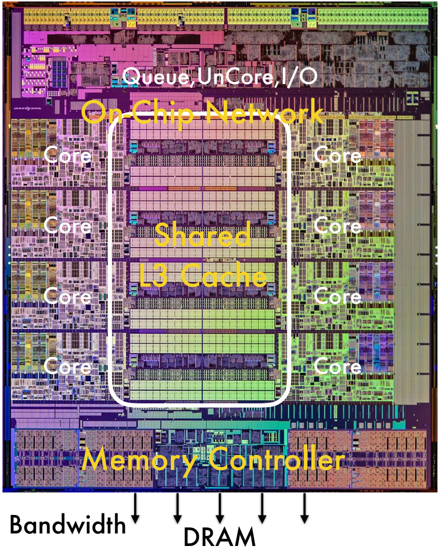
- Partitioning-induced associativity loss
- Unpredictable miss rate curve

Off-Chip Memory Bandwidth

- Unfair/Unreasonable memory bandwidth allocation

On-Chip Network

- Expensive deadlock avoidance



Intel Core i7-5960X

My contributions

Efficient techniques for sharing last-level cache, off-chip memory bandwidth and on-chip network

My contributions

- Last-level Cache
  - Futility Scaling: High-Associativity Cache Partitioning (MICRO 2014)
  - Predictable Cache Protection Policy (under preparation for submission)
- Off-chip Memory Bandwidth
  - Analytical Model for Memory Bandwidth Partitioning (IPDPS 2013)
- On-Chip Network
  - Bubble Coloring: Low-cost Deadlock Avoidance Scheme (ICS 2013)

My contributions

Efficient techniques for sharing last-level cache, off-chip memory bandwidth and on-chip network

My contributions

- Last-level Cache
  - Futility Scaling: High-Associativity Cache Partitioning (MICRO 2014)
  - Predictable Cache Protection Policy (under preparation for submission)
- Off-chip Memory Bandwidth
  - Analytical Model for Memory Bandwidth Partitioning (IPDPS 2013)
- On-Chip Network
  - Bubble Coloring: Low-cost Deadlock Avoidance Scheme (ICS 2013)

An Analytical Performance Model for Memory Bandwidth Partitioning

Shared Memory Bandwidth Management

Focus on fairness

- Fair Queue Memory System – divide the memory bandwidth equally for each application [Nesbit et al., 2006]

Focus on throughput

- ATLAS – prioritize the applications that have attained the least service over others [Kim et al., 2010a]

Focus on both throughput and fairness

- Thread Cluster Memory Scheduler – improves both system throughput and fairness by clustering different types of threads together [Kim et al., 2010b]

Shared Memory Bandwidth Management

Focus on fairness

- Fair Queue Memory System – divide the memory bandwidth equally for each application [Nesbit et al., 2006]

Focus on throughput

- ATLAS – prioritize the applications that have attained the least service over others [Kim et al., 2010a]

What are the best memory bandwidth partitioning schemes for different system performance objectives?

Focus on both throughput and fairness

- Thread Cluster Memory Scheduler – improves both system throughput and fairness by clustering different types of threads together [Kim et al., 2010b]

Model for Memory Bandwidth Partitioning

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \mathit{SystemObjectiveFunction}(x) \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i \leq B, i = 1, \ldots, N \end{aligned}$$

Model for Memory Bandwidth Partitioning

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \mathit{SystemObjectiveFunction}(x) \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i \leq B, i = 1, \ldots, N \end{aligned}$$

Common System Performance Objectives

Throughput-oriented:	Weighted Speedup / Sum of IPCs
Fairness:	Minimum Fairness (Lowest Speedup)
Balancing throughput and fairness:	Harmonic Weighted Speedup

Single Application Performance Model

$$IPC_{shared,i} = \frac{APC_{shared,i}}{API_i} = \frac{x_i}{API_i}$$

- IPC: Instructions Per Cycle
- APC: memory Accesses Per Cycle
- API: memory Accesses Per Instruction

Single Application Performance Model

$$IPC_{shared,i} = \frac{APC_{shared,i}}{API_i} = \frac{x_i}{API_i}$$

- IPC: Instructions Per Cycle
- APC: memory Accesses Per Cycle
- API: memory Accesses Per Instruction

Example

Assume an application takes 10,000 cycles to execute 1,000 instructions, during which it generates 100 memory accesses

- IPC = 1,000/10,000 = 0.1
- API = 100/1,000 = 0.1
- APC = 100/10,000 = 0.01

Harmonic Weighted Speedup

$$\underset{X}{\text{maximize}} \quad H_{sp} = \frac{N}{\sum_{i=1}^{N} \frac{IPC_{alone,i}}{IPC_{shared,i}}} = \frac{N}{\sum_{i=1}^{N} \frac{APC_{alone,i}}{x_i}}$$

$$\text{subject to} \quad \sum_{i=1}^{N} x_i \leq B, i = 1, \ldots, N$$

- Optimal Partitioning — ***Square_root***

$$\frac{x_i}{x_j} = \frac{\sqrt{APC_{alone,i}}}{\sqrt{APC_{alone,j}}}$$

Fairness

$$\frac{IPC_{shared,i}}{IPC_{alone,i}} = \frac{IPC_{shared,j}}{IPC_{alone,j}} \implies \frac{x_i}{APC_{alone,i}} = \frac{x_j}{APC_{alone,j}}$$

- Optimal Partitioning — ***Proportional***

$$\frac{x_i}{x_j} = \frac{APC_{alone,i}}{APC_{alone,j}}$$

Weighted Speedup

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & W_{sp} = \frac{1}{N}\sum_{i=1}^{N}\frac{IPC_{shared,i}}{IPC_{alone,i}} = \frac{1}{N}\sum_{i=1}^{N}\frac{x_i}{APC_{alone,i}} \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i \leq B, i = 1, \ldots, N \end{aligned}$$

- Optimal Partitioning — ***Priority_APC***
  - A fractional Knapsack problem
  - The optimal memory request scheduling is to always prioritize the requests from an application with a lower APC_{alone} over the ones from an application with a higher APC_{alone}
  - Similarly, the optimal partitioning for sum of IPCs is ***Priority_API***

Relationship between Performance Objectives and Memory bandwidth Partitioning

□ Application 1

■ Application 2

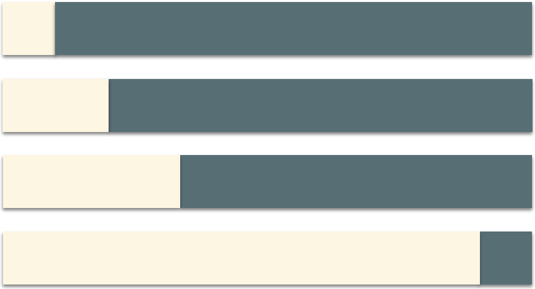
$$\frac{APC_{alone,1}}{APC_{alone,2}} = \frac{1}{4}$$

Uncontrolled Sharing

Best Fairness
Proportional (1:4)

Best Harmonic Weighted Speedup
Square_root (1:2)

Best Weighted Speedup
Priority_APC



Relationship between Performance Objectives and Memory bandwidth Partitioning

□ Application 1

▨ Application 2

$$\frac{APC_{alone,1}}{APC_{alone,2}} = \frac{1}{4}$$

No One-Size-Fits-All

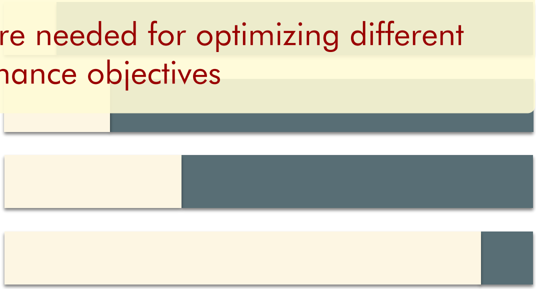
Different partitioning schemes are needed for optimizing different system performance objectives

Uncontrolled Sharing

Best Fairness
Proportional (1:4)

Best Harmonic Weighted Speedup
Square_root (1:2)

Best Weighted Speedup
Priority_APC



Evaluation Methodology

Full system simulator (Gem5) + Memory subsystem simulator (DRAMSim2)

System Configuration

Cores

- Four out of order cores

Caches

- L1 I-cache/D-cache
  - 32KB, 2-way, 1 ns, 64B line
- Private unified L2
  - 256KB, 8-way, 5 ns, 64B line

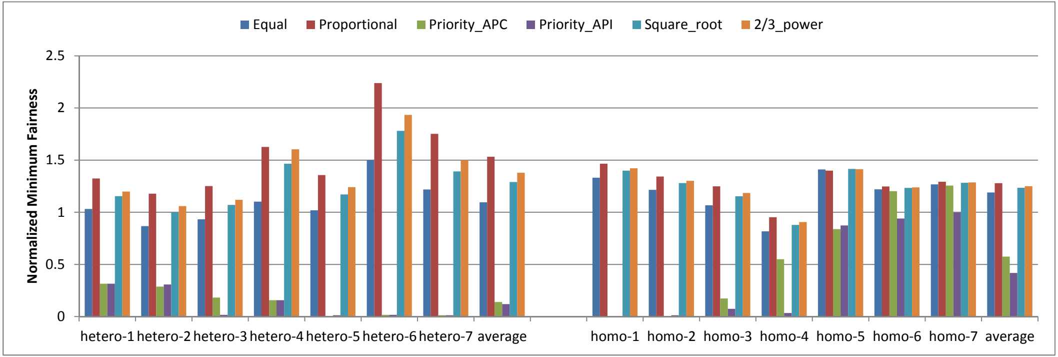
Memory

- DDR2-400
- tRP-tRCD-CL: 12.5-12.5-12.5ns

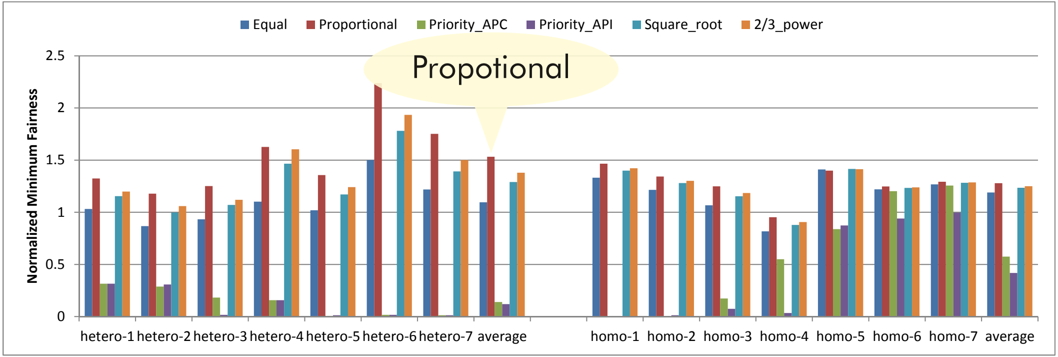
Workloads

- Benchmark: SPEC CPU 2006
- 14 Workloads
  - Mix 4 benchmarks
- RSD: Relative Standard Deviation of APC_{alone}s of co-scheduled applications
- 7 Heterogeneous
  - RSD $>$ 30
- 7 Homogeneous
  - RSD $<$ 30

Results: Fairness

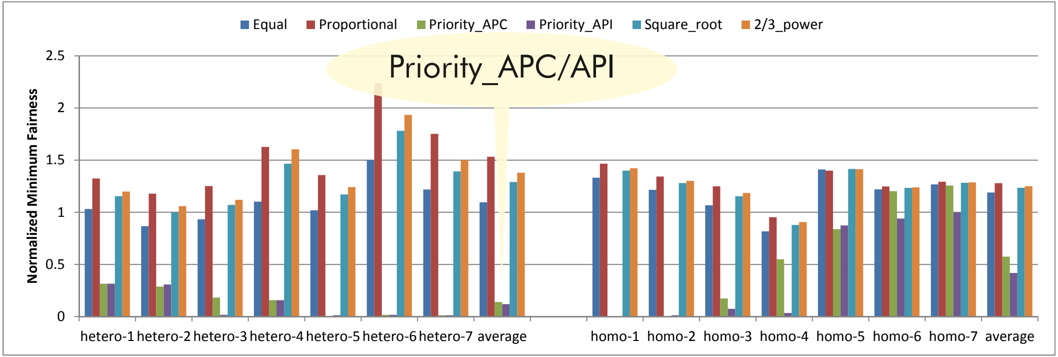


Results: Fairness



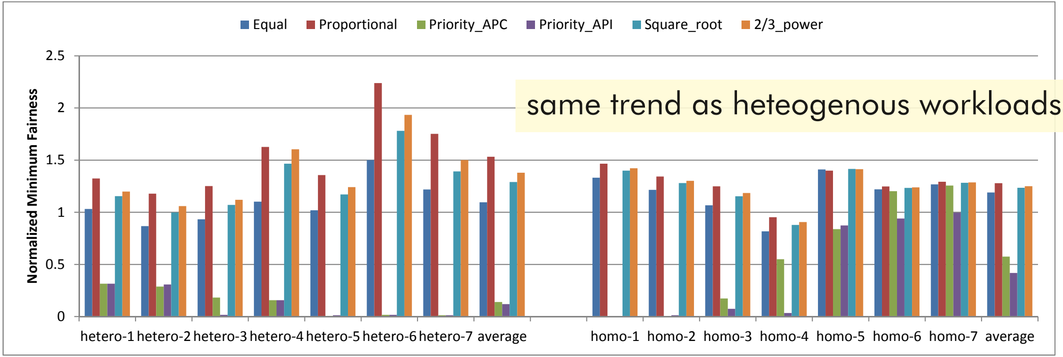
Proportional scheme achieves highest minimum fairness (> 50% improvement over ***No_partitioning***)

Results: Fairness



Proportional scheme achieves highest minimum fairness (> 50% improvement over ***No_partitioning***)

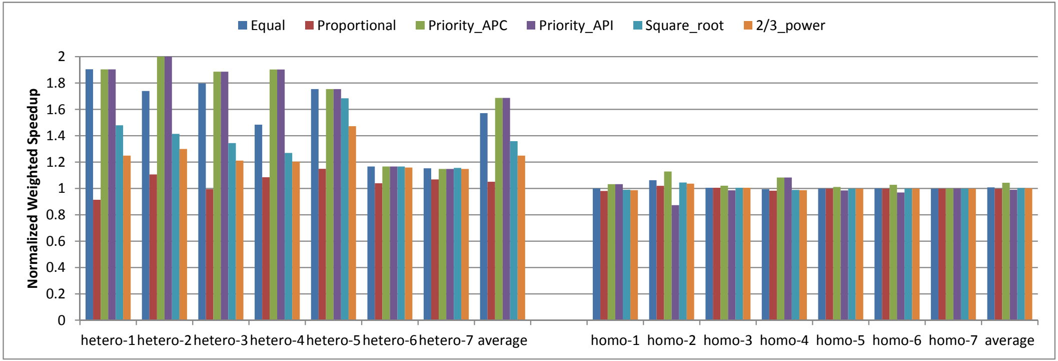
Results: Fairness



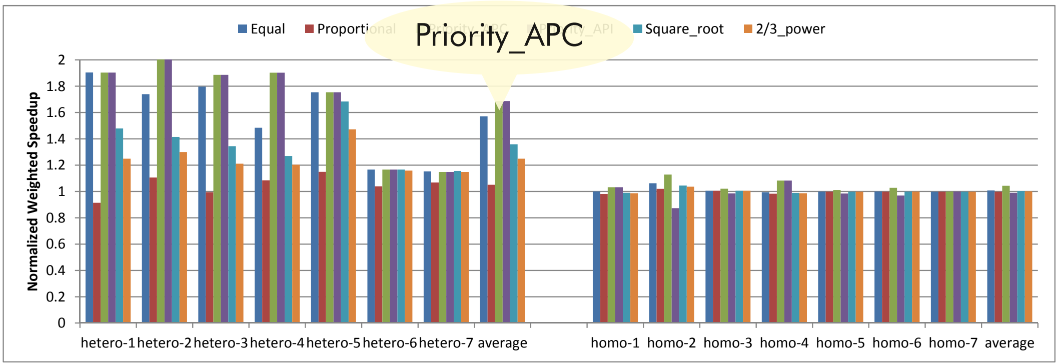
same trend as heteogenous workloads

Proportional scheme achieves highest minimum fairness (> 50% improvement over ***No_partitioning***)

Results: Weighted Speedup

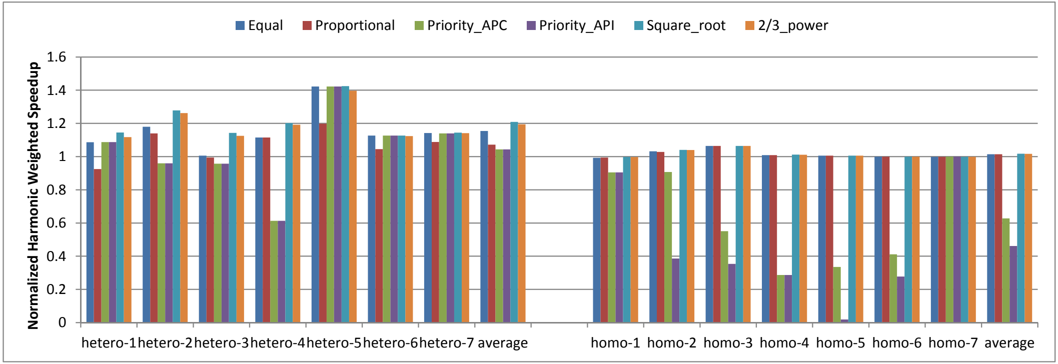


Results: Weighted Speedup

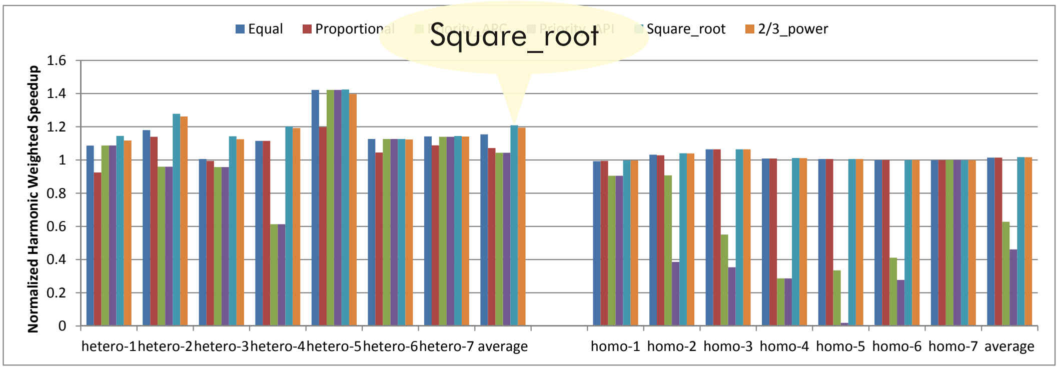


Priority_APC achieves highest Weighted Speedup (64.2% improvement over *No_Partitioning*)

Results: Harmonic Weighted Speedup



Results: Harmonic Weighted Speedup



Square_root scheme achieves highest Harmonic Weighted Speedup (20.3% improvement over ***No_partitioning***

Summary of Bandwidth Partitioning Model

- Analytical model that establishes the relationship between memory bandwidth partitioning schemes and system performance objectives
- No one-size-fits-all
  - Based on the model, different optimal partitioning schemes for different performance objectives are derived

Summary of Bandwidth Partitioning Model

- Analytical model that establishes the relationship between memory bandwidth partitioning schemes and system performance objectives
- No one-size-fits-all
  - Based on the model, different optimal partitioning schemes for different performance objectives are derived
- Extension for cache partitioning

$$IPC_{shared,i} = \frac{APC_{shared,i}}{API_{shared,i}} = \frac{memory\_bandwidth\_share_i}{F_i\left(cache\_capacity\_share_i\right)}$$

Summary of Bandwidth Partitioning Model

- Analytical model that establishes the relationship between memory bandwidth partitioning schemes and system performance objectives
- No one-size-fits-all
  - Based on the model, different optimal partitioning schemes for different performance objectives are derived
- Extension for cache partitioning

$$IPC_{shared,i} = \frac{APC_{shared,i}}{API_{shared,i}} = \frac{memory\_bandwidth\_share_i}{F_i\left(cache\_capacity\_share_i\right)}$$

Predictable cache miss rate curve

Predictable Cache Protection Policy

Overview of Cache Protection Policies

Insertion based Policy

What fraction of incoming lines will be protected? ⇒ insertion ratio ρ

Bimodal Insertion Policy (BIP[1])

- 1/32 (ρ) of incoming lines are inserted to MRU position
- The rest of incoming lines are inserted to LRU position

Protecting Distance based Policy

How long will existing lines be protected? ⇒ protecting distance d_p

Protecting Distance based Policy (PDP[2])

- An inserted/reused line is protected for d_p accesses before its eviction
- An incoming line will bypass the cache if no unprotected candidates available

[1] M. Qureshi, et al. “Adaptive insertion policies for high performance caching” ISCA 2007

[2] N. Duong, et al. “Improving cache management policies using dynamic reuse distances” MICRO 2012

Overview of Cache Protection Policies

Insertion based Policy

What fraction of incoming lines will be protected? ⇒ insertion ratio ρ

Bimodal Insertion Policy (BIP[1])

- 1/32 (ρ) of incoming lines are inserted to MRU position
- The rest of incoming lines are inserted to LRU position

Protecting Distance based Policy

How long will existing lines be protected? ⇒ protecting distance d_p

Protecting Distance based Policy (PDP[2])

- An inserted/reused line is protected for d_p accesses before its eviction
- An incoming line will bypass the cache if no unprotected candidates available

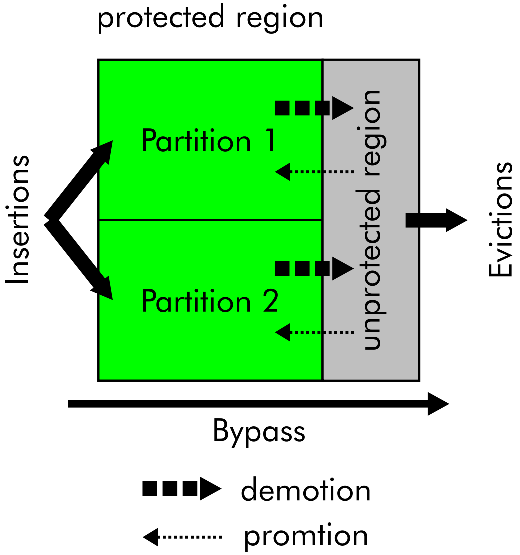
Why do we need predictability?

1. Help the cache controller to enforce better d_p or ρ.
2. Help the resource allocation algorithm to make intelligent decisions to share the cache.

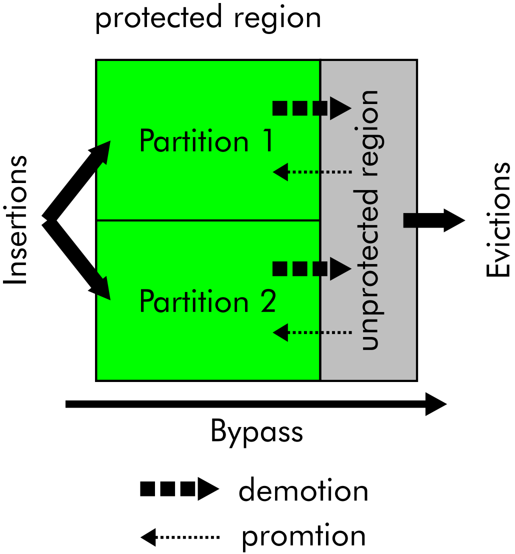
[1] M. Qureshi, et al. "Adaptive insertion policies for high performance caching" ISCA 2007

[2] N. Duong, et al. "Improving cache management policies using dynamic reuse distances" MICRO 2012

Predictable Cache Protection Policy (PCPP)



Predictable Cache Protection Policy (PCPP)



Operations

On a hit

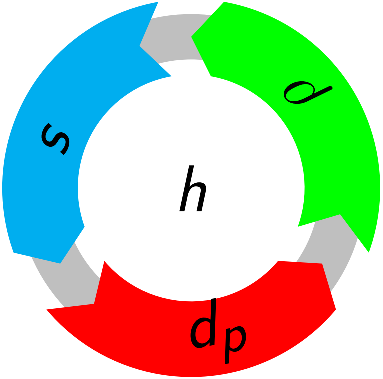
1. reset the hit line's age to zero
2. ***promote*** if the line is unprotected

On a miss

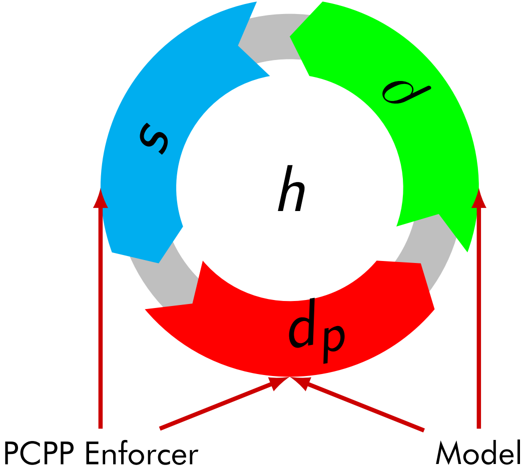
1. ***demote*** if candidate's age > d_p
2. if ① # of protected lines < s and ② unprotected candidates exist
   - ***insert*** the incoming line
   - ***evict*** an unprotected candidate

   otherwise → ***bypass*** (ρ=1-bypass_rate)

Model Overview



Model Overview

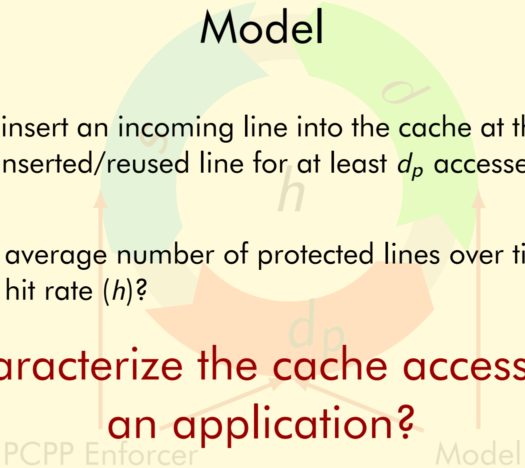


Model Overview

Model

- Inputs (ρ, d_p)
  1. On a miss, insert an incoming line into the cache at the probability of ρ
  2. Protect the inserted/reused line for at least d_p accesses
- Outputs (h, s)
  1. What is the average number of protected lines over time (s)?
  2. What is the hit rate (h)?

How to characterize the cache access pattern of an application?

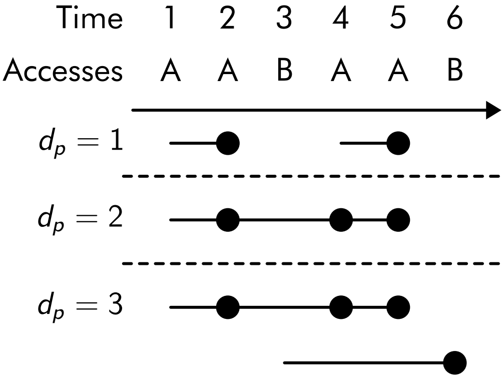


Reuse Streak

- A d_p-protected reuse: an access whose reuse distance $\leq d_p$
- A d_p-protected reuse streak: a sequence of consecutive d_p-protected reuses

Reuse Streak

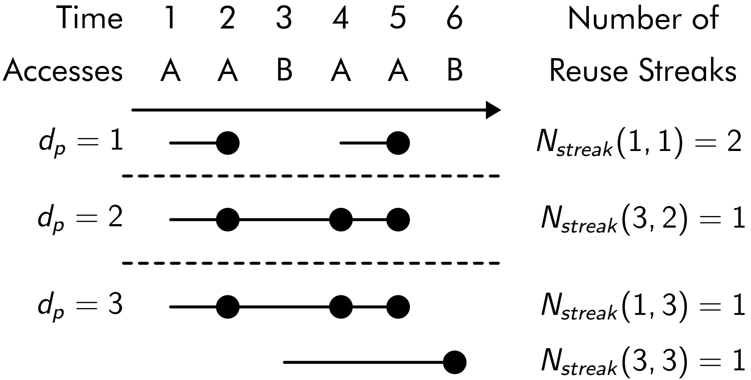
- A d_p-protected reuse: an access whose reuse distance $\leq d_p$
- A d_p-protected reuse streak: a sequence of consecutive d_p-protected reuses



Time	1	2	3	4	5	6
Accesses	A	A	B	A	A	B

Reuse Streak

- A d_p-protected reuse: an access whose reuse distance $\leq d_p$
- A d_p-protected reuse streak: a sequence of consecutive d_p-protected reuses
- $N_{streak}(l, d_p)$: number of d_p-protected reuse streaks whose length is l



Time 1 2 3 4 5 6

Accesses A A B A A B

Number of Reuse Streaks

$d_p = 1$ $N_{streak}(1,1) = 2$

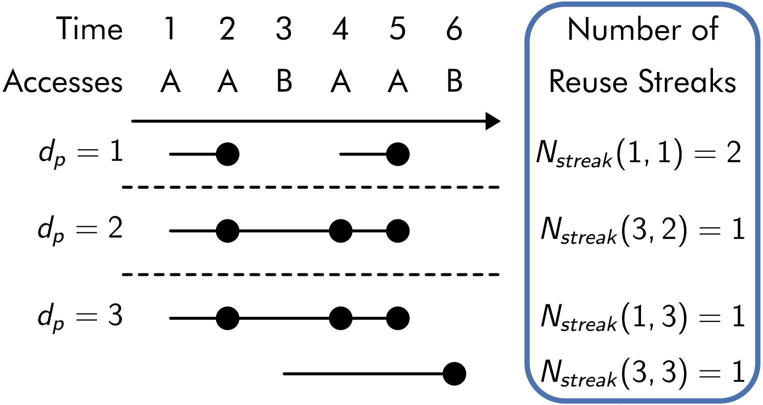
$d_p = 2$ $N_{streak}(3,2) = 1$

$d_p = 3$ $N_{streak}(1,3) = 1$

$N_{streak}(3,3) = 1$

Reuse Streak

- A d_p-protected reuse: an access whose reuse distance $\leq d_p$
- A d_p-protected reuse streak: a sequence of consecutive d_p-protected reuses
- $N_{streak}(l, d_p)$: number of d_p-protected reuse streaks whose length is l



Time	1	2	3	4	5	6
Accesses	A	A	B	A	A	B

$d_p = 1$

$d_p = 2$

$d_p = 3$

Number of Reuse Streaks

$N_{streak}(1, 1) = 2$

$N_{streak}(3, 2) = 1$

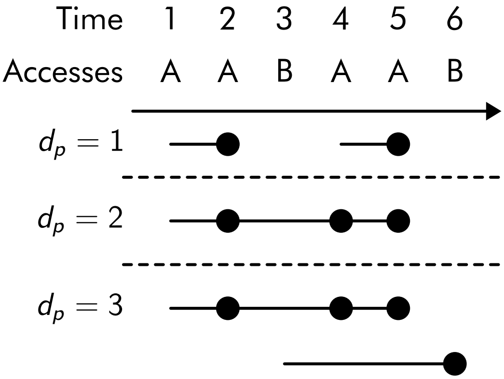
$N_{streak}(1, 3) = 1$

$N_{streak}(3, 3) = 1$

full information

Reuse Streak

- A d_p-protected reuse: an access whose reuse distance $\leq d_p$
- A d_p-protected reuse streak: a sequence of consecutive d_p-protected reuses
- $N_{streak}(l, d_p)$: number of d_p-protected reuse streaks whose length is l

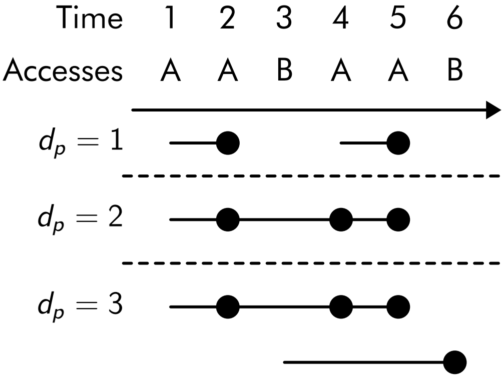


Time	1	2	3	4	5	6
Accesses	A	A	B	A	A	B

	Number of Reuse Streaks	Average Reuse Streak Length
$d_p = 1$	$N_{streak}(1, 1) = 2$	$\overline{L}(1) = 1$
$d_p = 2$	$N_{streak}(3, 2) = 1$	$\overline{L}(2) = 3$
$d_p = 3$	$N_{streak}(1, 3) = 1$ $N_{streak}(3, 3) = 1$	$\overline{L}(3) = 2$

Reuse Streak

- A d_p-protected reuse: an access whose reuse distance $\leq d_p$
- A d_p-protected reuse streak: a sequence of consecutive d_p-protected reuses
- $N_{streak}(l, d_p)$: number of d_p-protected reuse streaks whose length is l



Time	1	2	3	4	5	6
Accesses	A	A	B	A	A	B

$d_p = 1$

$d_p = 2$

$d_p = 3$

Number of Reuse Streaks

$N_{streak}(1, 1) = 2$

$N_{streak}(3, 2) = 1$

$N_{streak}(1, 3) = 1$

$N_{streak}(3, 3) = 1$

Average Reuse Streak Length

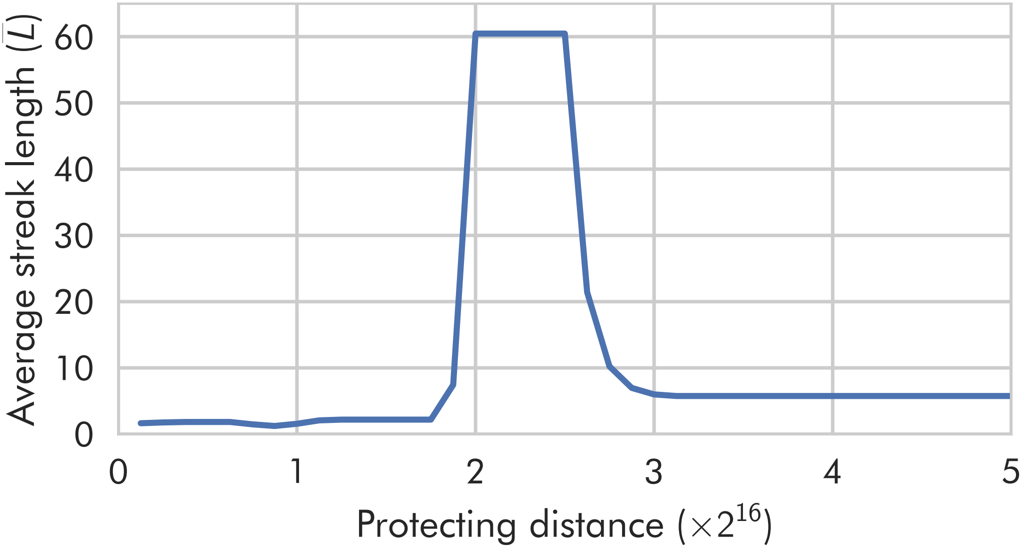
$\overline{L}(1) = 1$

$\overline{L}(2) = 3$

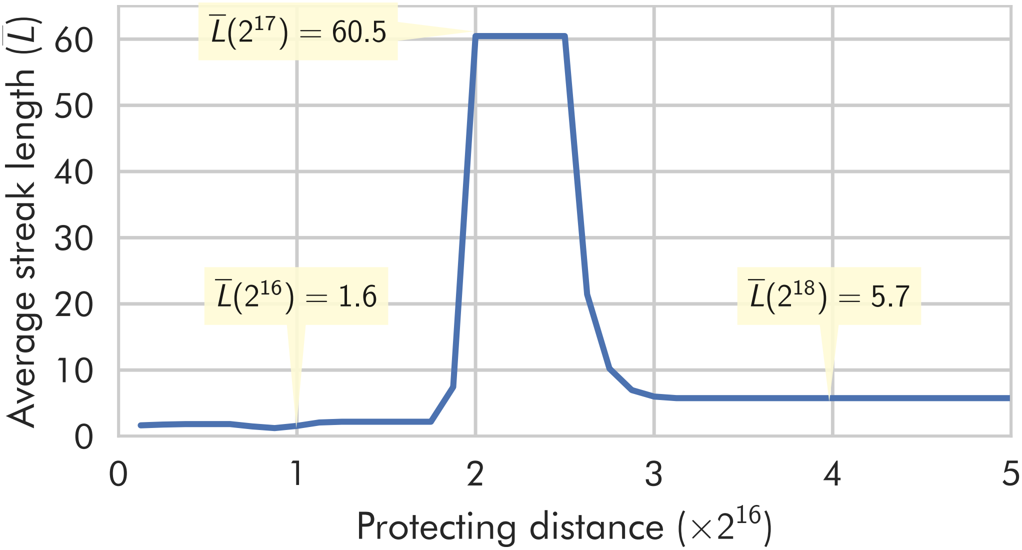
$\overline{L}(3) = 2$

approximate

Average Reuse Streak Length (cactusADM)



Average Reuse Streak Length (cactusADM)



Hit Rate of a Single Streak

Assumption: the insertions of incoming lines are independent.

$$\begin{aligned} h_{streak}(l, \rho) &= l - E\left(N_{failures}\right) \\ &= l - \frac{(1-\rho)\left(1-(1-\rho)^{l}\right)}{\rho} \\ &\gtrapprox l + 1 - \frac{1}{\rho} \quad (\text{when } l \to \infty) \end{aligned}$$

Hit Rate of a Single Streak

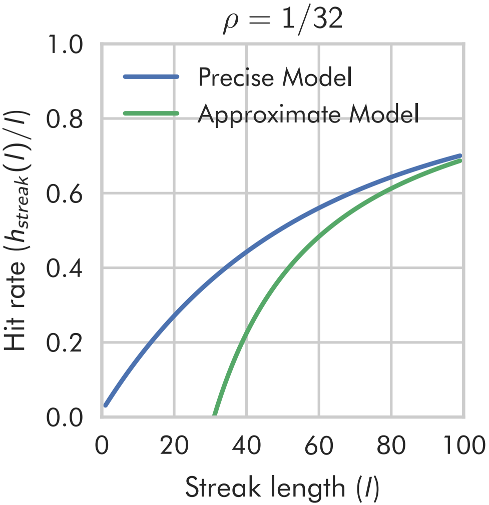
Assumption: the insertions of incoming lines are independent.

$$
\begin{aligned}
h_{streak}\left(l,\rho\right) &= l - E\left(N_{failures}\right) \\
&= l - \frac{(1-\rho)\left(1-(1-\rho)^{l}\right)}{\rho} && \longleftarrow \text{Precise} \\
&\gtrapprox l + 1 - \frac{1}{\rho} \quad (\text{when } l \to \infty) && \longleftarrow \text{Approximate}
\end{aligned}
$$

Hit Rate of a Single Streak

Assumption: the insertions of incoming lines are independent.

$$
\begin{aligned}
h_{streak}\left(l,\rho\right) &= l - E\left(N_{failures}\right)\\
&= l - \frac{(1-\rho)\left(1-(1-\rho)^{l}\right)}{\rho}\\
&\gtrapprox l + 1 - \frac{1}{\rho} \quad (\text{when } l \to \infty)
\end{aligned}
$$



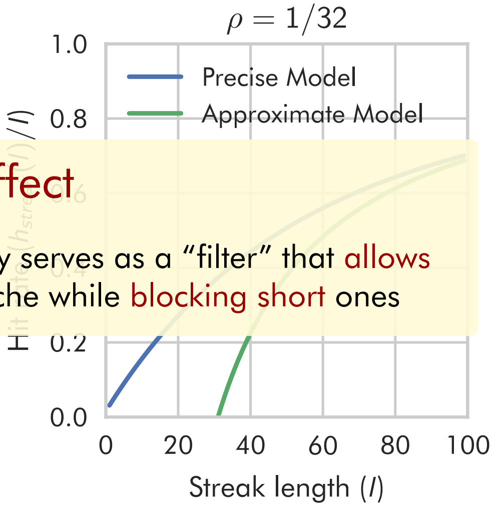
Hit Rate of a Single Streak

Assumption: the insertions of incoming lines are independent.

$$h_{streak}(l,\rho) = l - E\left(N_{failures}\right)$$

$$= l - \frac{(1-\rho)\left(1-(1-\rho)^{l}\right)}{\rho}$$

$$\gtrapprox l + 1 - \frac{1}{\rho} \quad (\text{when } l \to \infty)$$



Streak Effect

When $\rho \ll 1$, a cache protection policy serves as a “filter” that allows long reuse streaks to occupy the cache while blocking short ones

Model

- Hit model $h(\rho)$

$$h(\rho) = \frac{\text{total hits}}{\text{total accesses}} = \frac{\sum_{l=1}^{\infty} N_{streaks}(l) \times h_{streak}(l)}{\text{total accesses}}$$
$$\gtrapprox H_{max}\left(1 + \frac{1}{\overline{L}} - \frac{1}{\rho\overline{L}}\right) = H_{max} - \frac{H_{max}}{\overline{L}}\left(\frac{1-\rho}{\rho}\right)$$

- Size model $s(\rho)$

$$s(\rho) = \frac{\text{lifetime of all lines}}{\text{total accesses}} = \frac{\text{total hits} \times \overline{D} + \text{total evictions} \times d_p}{\text{total accesses}}$$
$$= \frac{\text{total hits}}{\text{total accesses}} \times \overline{D} + \frac{\text{total insertions}}{\text{total accesses}} \times d_p = h(\rho)\overline{D} + \rho(1 - h(\rho))d_p$$

Model

- Hit model $h(\rho)$

$$h(\rho) = \frac{\text{total hits}}{\text{total accesses}} = \frac{\sum_{l=1}^{\infty} N_{streaks}(l) \times h_{streak}(l)}{\text{total accesses}}$$

$$\lessapprox H_{max}\left(1 + \frac{1}{\overline{L}} - \frac{1}{\rho\overline{L}}\right) = H_{max} - \frac{H_{max}}{\overline{L}}\left(\frac{1-\rho}{\rho}\right)$$

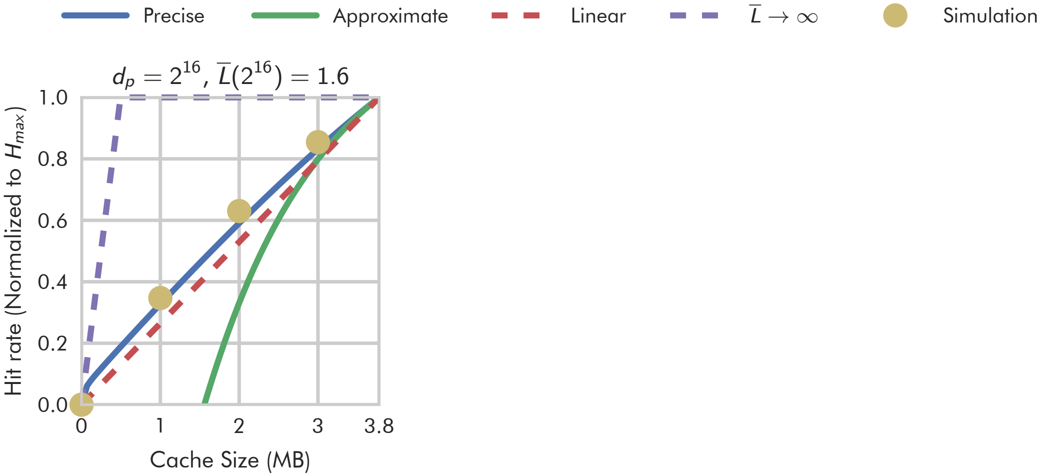
Model	Required Information
Precise	full reuse streak pattern
Approximate	average reuse streak length ($\overline{L}$)

- Size model $s(\rho)$

$$s(\rho) = \frac{\text{lifetime of all lines}}{\text{total accesses}} = \frac{\text{total hits} \times \overline{D} + \text{total evictions} \times d_p}{\text{total accesses}}$$

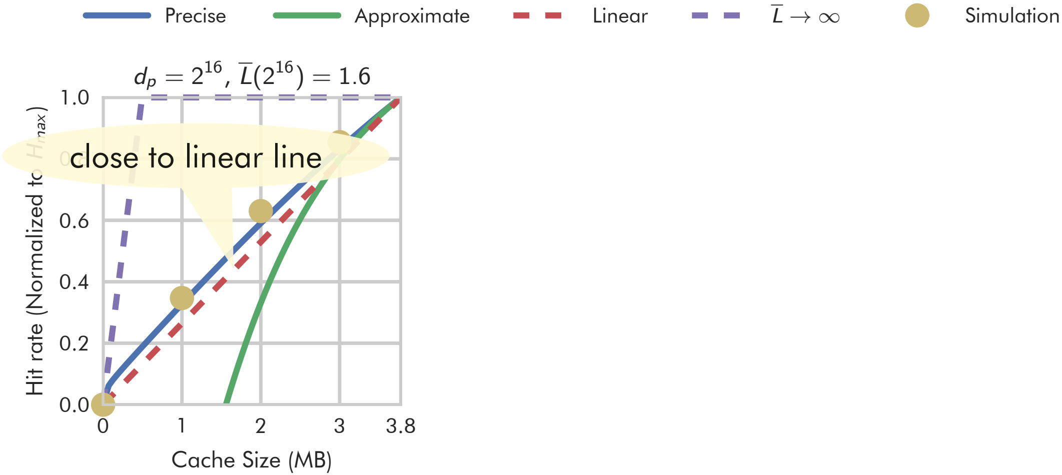
$$= \frac{\text{total hits}}{\text{total accesses}} \times \overline{D} + \frac{\text{total insertions}}{\text{total accesses}} \times d_p = h(\rho)\overline{D} + \rho(1 - h(\rho))d_p$$

Model Validation (cactusADM)



Short Streaks

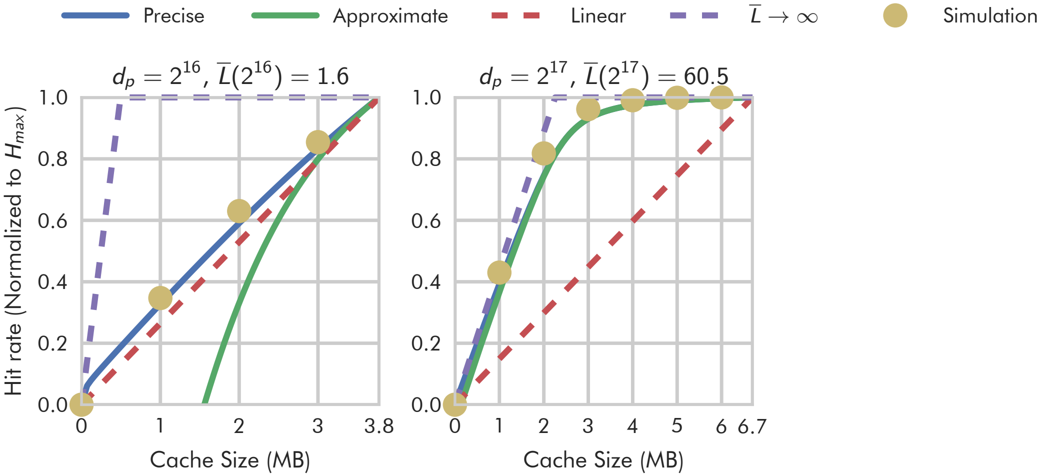
Model Validation (cactusADM)



close to linear line

Short Streaks

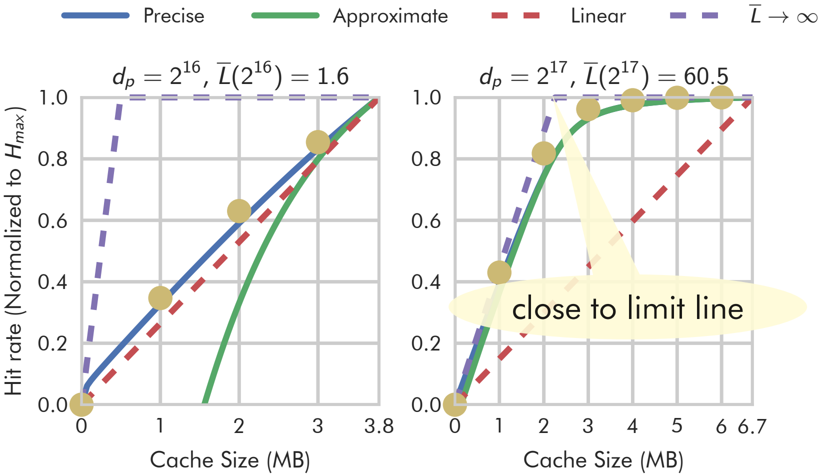
Model Validation (cactusADM)



Short Streaks

Long Streaks

Model Validation (cactusADM)

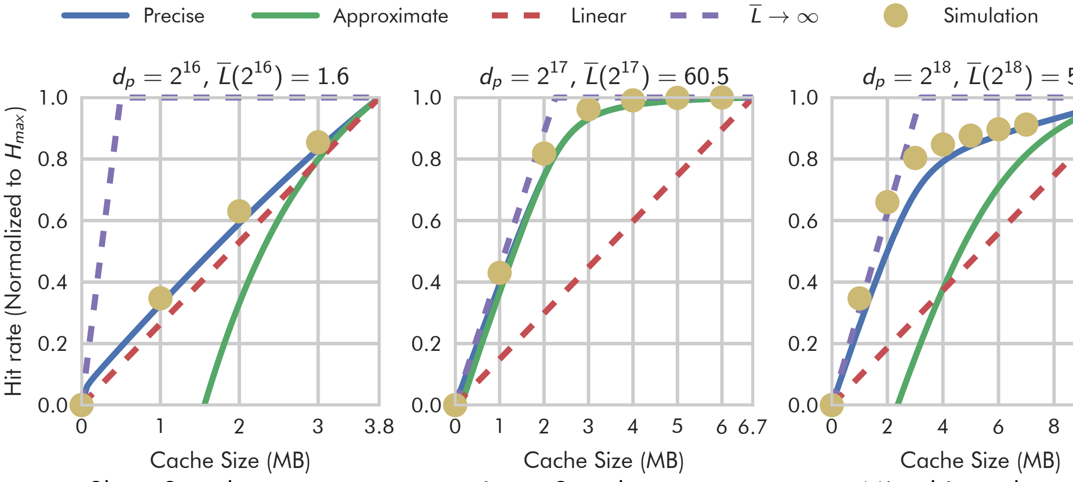


close to limit line

Short Streaks

Long Streaks

Model Validation (cactusADM)



Short Streaks

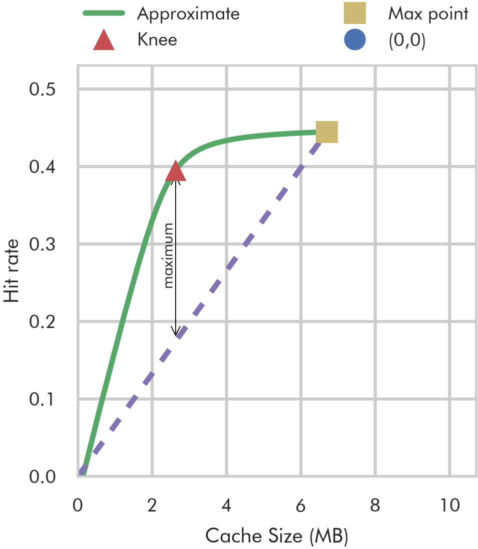
Long Streaks

Mixed Length

Hit Rate Curve Construction

"Knee": the point on the approximate curve that has the maximum distance from the linear reference line

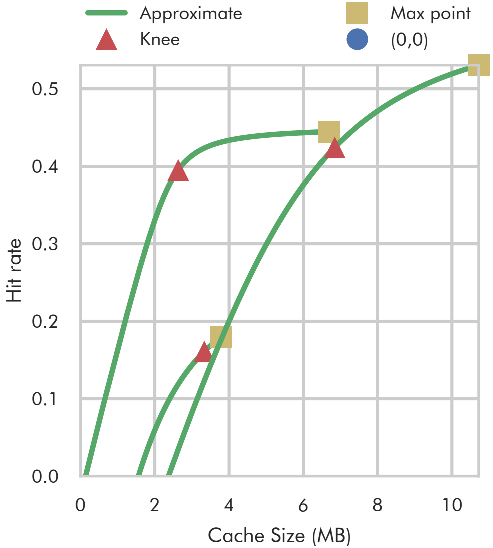
$$\rho_{knee}(d_p) = \frac{1}{\sqrt{\overline{L} - \frac{H_{max}}{\overline{L}(1-H_{max})}}} \approx \frac{1}{\sqrt{\overline{L}}}$$



Hit Rate Curve Construction

“Knee”: the point on the approximate curve that has the maximum distance from the linear reference line

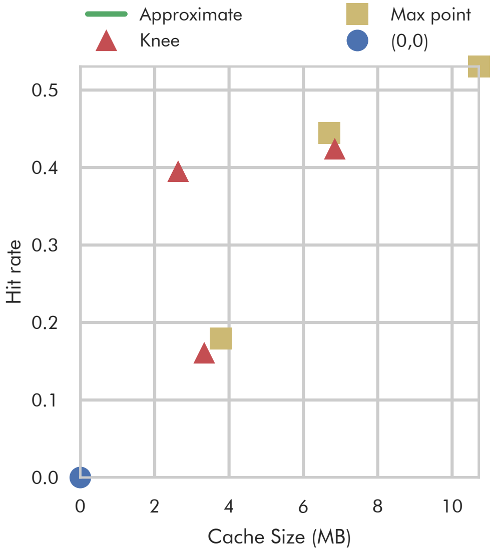
$$\rho_{knee}(d_p) = \frac{1}{\sqrt{\overline{L} - \frac{H_{max}}{\overline{L}(1-H_{max})}}} \approx \frac{1}{\sqrt{\overline{L}}}$$



Hit Rate Curve Construction

"Knee": the point on the approximate curve that has the maximum distance from the linear reference line

$$\rho_{knee}(d_p) = \frac{1}{\sqrt{\overline{L} - \frac{H_{max}}{\overline{L}(1-H_{max})}}} \approx \frac{1}{\sqrt{\overline{L}}}$$



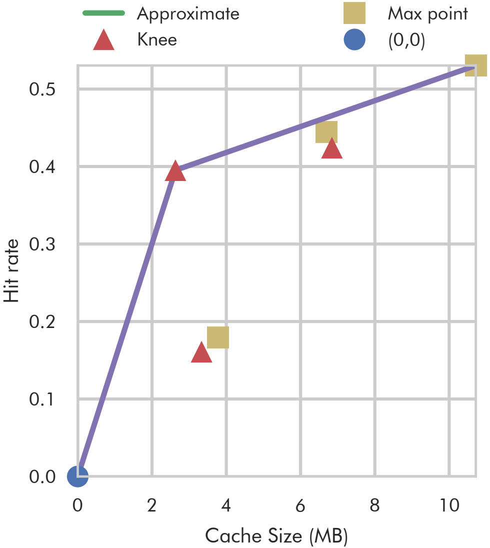
Hit Rate Curve Construction

"Knee": the point on the approximate curve that has the maximum distance from the linear reference line

$$\rho_{knee}(d_p) = \frac{1}{\sqrt{\overline{L} - \frac{H_{max}}{\overline{L}(1-H_{max})}}} \approx \frac{1}{\sqrt{\overline{L}}}$$

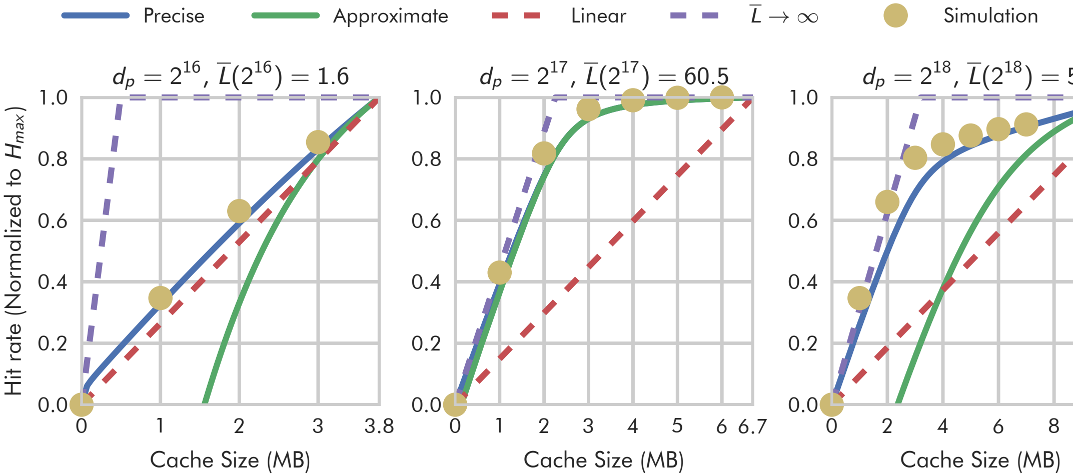
Talus[3]: yield a hit rate curve that traces out the convex hull of a set of points

Apply Talus technique on (0,0), Knee points, Max points



[3] N. Beckmann and D. Sanchez."Talus: A simple way to remove cliffs in cache performance." HPCA 2015

Model Validation (cactusADM)



Short Streaks

Long Streaks

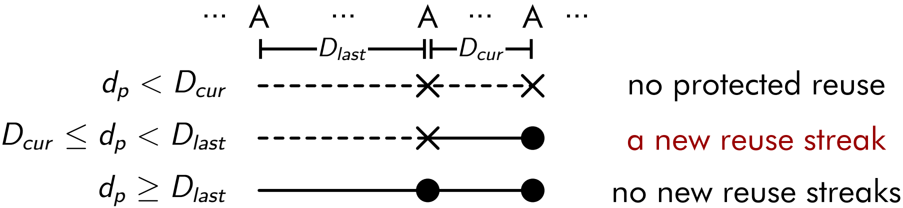
Mixed Length

Profiling Average Reuse Streak Length

$$\overline{L} = \frac{\text{total reuses}}{\#\text{ of reuse streaks}} = \frac{\text{total reuses}}{\#\text{ of streak starts} - \#\text{ of streak ends}}$$

Profiling Average Reuse Streak Length

$$\overline{L} = \frac{\text{total reuses}}{\text{\# of reuse streaks}} = \frac{\text{total reuses}}{\text{\# of streak starts} - \text{\# of streak ends}}$$



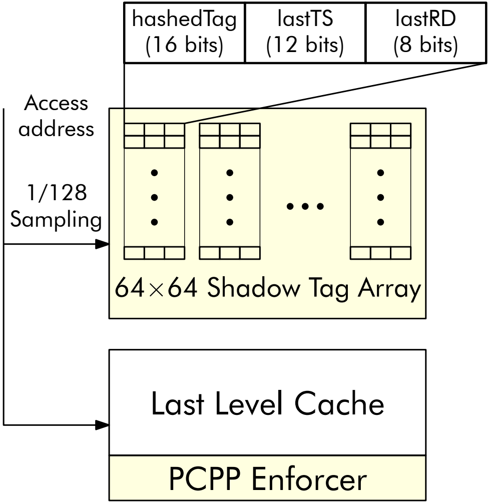
$d_p < D_{cur}$ no protected reuse

$D_{cur} \leq d_p < D_{last}$ a new reuse streak

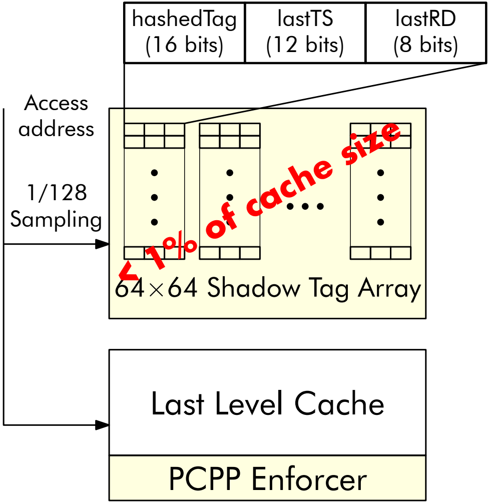
$d_p \geq D_{last}$ no new reuse streaks

Detecting the start of a reuse streak

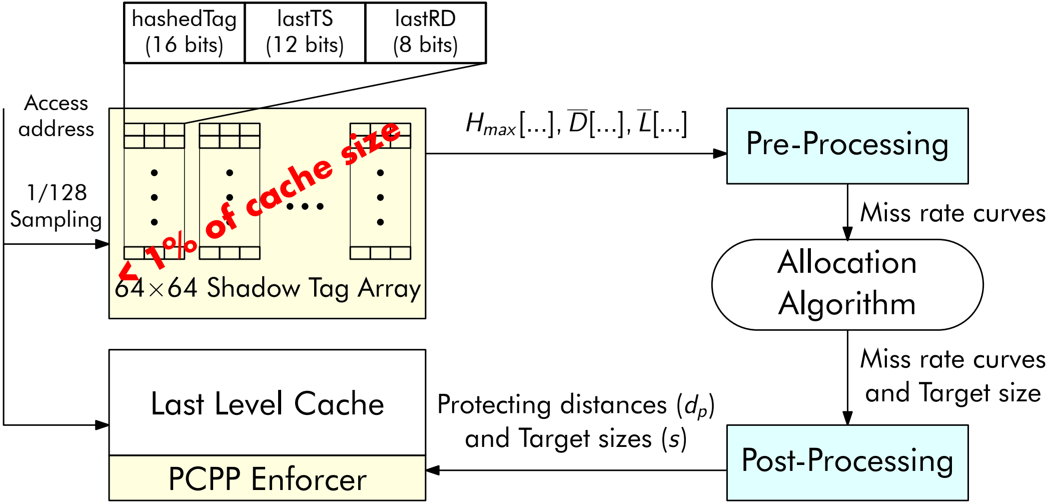
Implementation



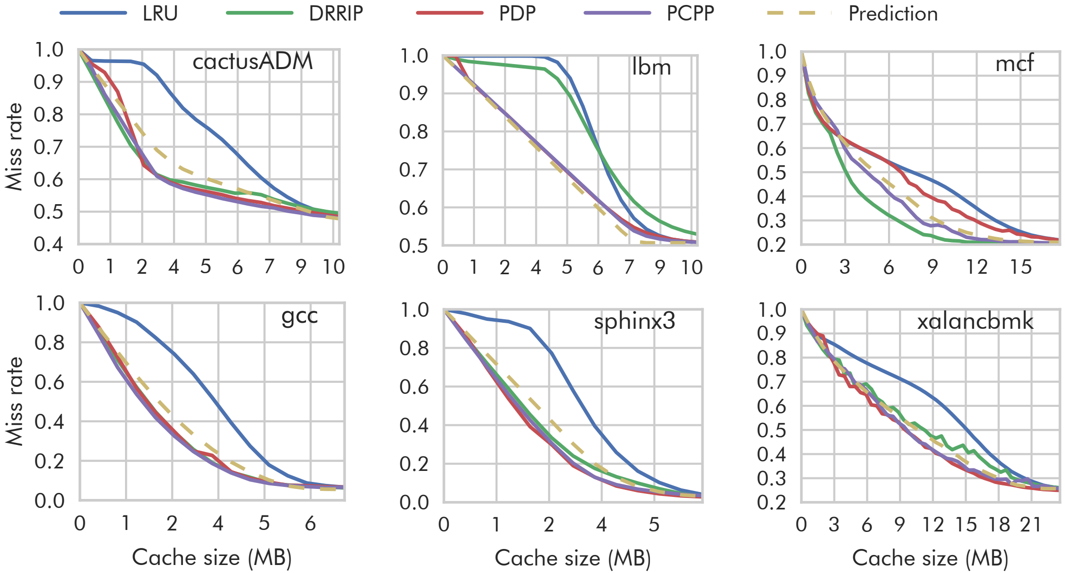
Implementation



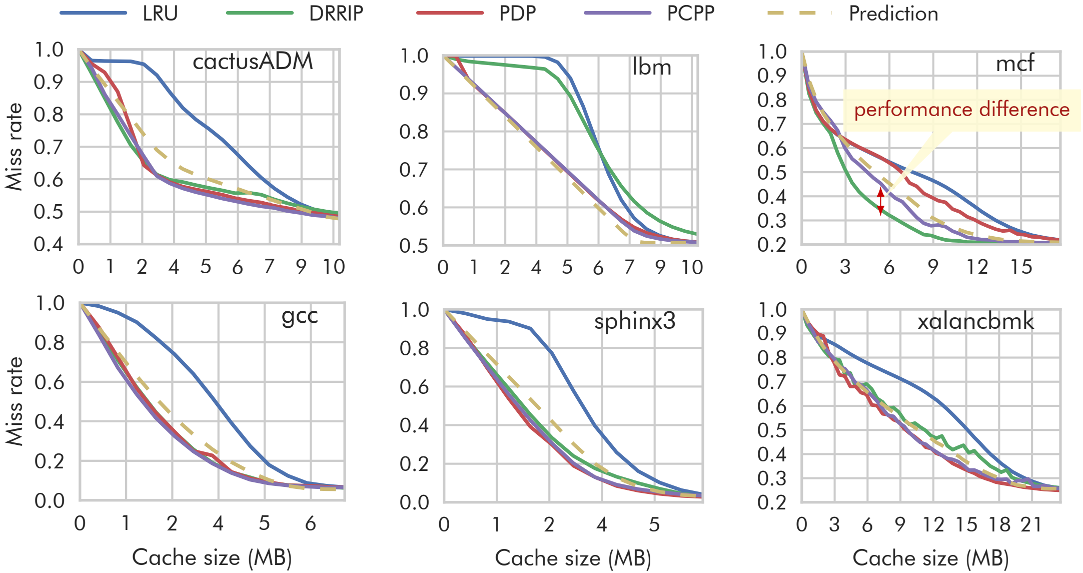
Implementation



Results


LRU
DRRIP
PDP
PCPP
Prediction
cactusADM
lbm
mcf
gcc
sphinx3
xalancbmk
Miss rate
Miss rate
Cache size (MB)
Cache size (MB)
Cache size (MB)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0 1 2 4 5 6 7 9 10
1.0
0.9
0.8
0.7
0.6
0.5
0 1 2 4 5 6 7 9 10
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0 3 6 9 12 15
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 4 5 6
1.0
0.8
0.6
0.4
0.2
0.0
0 1 2 4 5
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0 3 6 9 12 15 18 21

Results


LRU
DRRIP
PDP
PCPP
Prediction
cactusADM
lbm
mcf
gcc
sphinx3
xalancbmk
performance difference
Miss rate
Miss rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.2
0.0
0 1 2 4 5 6 7 9 10
0 3 6 9 12 15
0 1 2 4 5 6
0 1 2 4 5
0 3 6 9 12 15 18 21
Cache size (MB)
Cache size (MB)
Cache size (MB)

PCPP Summary

- The reuse streak concept and the streak effect that explains the behaviors of a cache protection policy
- A precise and an approximate model to predict the performance of cache protection policy based on reuse streak information
- A runtime profiler for average reuse steak length and a practical cache protection policy that produces predictable miss rate curves

Conclusions

To enable aggressive workload collocation on a chip, shared on-chip resources needs to be managed in an efficient and effective way.

- Last-level cache
  - High-associativity cache partitioning
  - Predictable high-performance cache policy
- Off-chip memory bandwidth
  - Goal-oriented memory bandwidth allocation
- On-chip network
  - Low-cost deadlock avoidance

My Publications

- Ruisheng Wang and Lizhong Chen, “Futility Scaling: High-Associativity Cache Partitioning”, in Proceedings of the 47th IEEE/ACM International Symposium on Microarchitecture (MICRO), December 2014
- Lizhong Chen, Lihang Zhao, Ruisheng Wang and Timothy Mark Pinkston, “MP3:MinimizingPerformance Penalty for Power-gating of Clos Network-on-Chip”, in Proceedings of the 20th IEEE International Symposium on High-Performance Computer Architecture (HPCA), February 2014
- Ruisheng Wang, Lizhong Chen and Timothy Mark Pinkston, “Bubble Coloring:Avoiding Routing- and Protocol-induced Deadlocks with Minimal Virtual Channel Requirement”, in Proceedings of the 27th International Conference on Supercomputing (ICS), June 2013
- Ruisheng Wang, Lizhong Chen and Timothy Mark Pinkston, “An Analytical Performance Model for Partitioning Off-Chip Memory Bandwidth”, in Proceedings of the 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS), May 2013
- Lizhong Chen, Ruisheng Wang and Timothy Mark Pinkston, “Critical Bubble Scheme: An Efficient Implementation of Globally-aware Network Flow Control”, in Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2011)
- Yuho Jin, Ruisheng Wang, Woojin Choi and Timothy Mark Pinkston, “Thread Criticality Support in On- Chip Networks”, in Proceedings of Third International Workshop on Network on Chip Architectures (NoCArc 2010), held in conjunction with the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-43)

Thank You For Listening!

Questions?