Domain-Specialized Cache Management for Graph Analytics

Priyank Faldu, Boris Grot, Jeff Diamond

This research is partially supported by a grant from Oracle Labs.
Cache management in the age of big data

Variety of application domains

Data Analytics  Graph Analytics  Machine Learning

Working set size much larger than typical SPEC benchmarks
- Vastly different cache access patterns across domains
Cache management in the age of big data

Variety of application domains

- Data Analytics
- Graph Analytics
- Machine Learning

Working set size much larger than typical SPEC benchmarks
- Vastly different cache access patterns across domains

Yet, cache management mechanisms are “domain-agnostic”
- Assumption: one size fits all
Cache management in the age of big data

Variety of application domains

Working set size much larger than typical SPEC benchmarks

Vastly different cache access patterns across domains

Yet, cache management mechanisms are “domain-agnostic”

Assumption: one size fits all

Graph Analytics

A case for domain-specialized cache management
Domain-agnostic techniques for graph analytics

SHiP-MEM  Hawkeye  Leeway

Graph Workloads (5 applications x 5 datasets)
Domain-agnostic techniques for graph analytics

HPCA'20

Winner of the latest cache replacement championship

Graph Workloads (5 applications x 5 datasets)
Domain-agnostic techniques for graph analytics
Domain-agnostic techniques for graph analytics

Speed-up over RRIP

Graph Workloads (5 applications x 5 datasets)

SHiP-MEM  Hawkeye  Leeway

Slowdown
Domain-agnostic techniques for graph analytics

Speed-up over RRIP

SHiP-MEM  Hawkeye  Leeway

1-15% geomean slowdown
Outline

➢ Performance of domain-agnostic cache management

➢ Graph analytics

➢ GRASP: domain-specialized cache management
  - Software-guided reuse-prediction
  - Hardware-enforced cache management

➢ Performance evaluation
Applications of graph analytics

Extract meaningful information out of complex many-to-many relationships among objects

Community Analysis
- Identify customers with similar interests
Applications of graph analytics

Extract meaningful information out of complex many-to-many relationships among objects

Community Analysis
- Identify customers with similar interests

Connectivity Analysis
- Find weakness in a network

Path Analysis
- Route optimization for distribution and supply chain

Centrality Analysis
- Most influential people and information in social media

And many others …
Real-world graphs & power-law degree distribution

Small fraction of vertices have high connectivity – **hot vertices**

Large fraction of vertices have low connectivity – **cold vertices**

Prevalent in many domains – e.g., Twitter user-follower graph
Real-world graphs & power-law degree distribution

Small fraction of vertices have high connectivity – hot vertices

Large fraction of vertices have low connectivity – cold vertices

Prevalent in many domains – e.g., Twitter user-follower graph

Average User

~700
Real-world graphs & power-law degree distribution

Small fraction of vertices have high connectivity – hot vertices

Large fraction of vertices have low connectivity – cold vertices

Prevalent in many domains – e.g., Twitter user-follower graph

Average User: ~700

Donald Trump: ~72M
Real-world graphs & power-law degree distribution

Small fraction of vertices have high connectivity – hot vertices

Large fraction of vertices have low connectivity – cold vertices

Prevalent in many domains – e.g., Twitter user-follower graph

How does connectivity influence cache locality?
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Cache Accesses in Time
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Cache Accesses in Time

Vertex Properties

HPCA'20
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Cache Accesses in Time
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Vertex Properties

Cache Accesses in Time
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Vertex Properties

Cache Accesses in Time
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Vertex Properties

V₀ → P₀
V₁ → P₁
V₂ → P₂
V₃ → P₃
V₄ → P₄
V₅ → P₅

Key observation: vertex reuse is proportional to its degree

Cache Accesses in Time
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Key observation: vertex reuse is proportional to its degree

Cache Accesses in Time
A canonical example of graph analytics

Computes property for a vertex based on its neighbors' properties

Example Graph

Vertex Properties

V₀
V₁
V₂
V₃
V₄
V₅

P₀
P₁
P₂
P₃
P₄
P₅

Key observation: vertex reuse is proportional to its degree

Hot vertices $\rightarrow$ Small footprint + High reuse
Challenging to identify hot vertices in hardware

Domain-agnostic techniques rely on purely hardware mechanisms
Challenging to identify hot vertices in hardware

Domain-agnostic techniques rely on purely hardware mechanisms.

Example Graph

Vertex Properties

Cache Accesses in Time
Challenging to identify hot vertices in hardware

Domain-agnostic techniques rely on purely hardware mechanisms

Example Graph

Vertex Properties

Reason 1 Irregular Accesses

Cache Accesses in Time
Challenging to identify hot vertices in hardware

Domain-agnostic techniques rely on purely hardware mechanisms

Example Graph

Reason 1 Irregular Accesses

Reason 2 Long Reuse Distances

Cache Accesses in Time

HPCA'20
Challenging to identify hot vertices in hardware

Domain-agnostic techniques rely on purely hardware mechanisms

Example Graph

Vertex Properties

Reason 1 Irregular Accesses

Reason 2 Long Reuse Distances

Idea: Leverage domain-knowledge for reuse prediction
Proposal: GRASP – a software-hardware co-design

Software aids hardware in identifying hot vertices

Hardware preferentially caches hot vertices
Outline

➢ Performance of domain-agnostic cache management

➢ Graph analytics

➢ GRASP: domain-specialized cache management
  - Software-guided reuse-prediction
  - Hardware-enforced cache management

➢ Performance evaluation
GRASP: Software-guided reuse-prediction

Task: Let software aid hardware in identifying hot vertices
GRASP: Software-guided reuse-prediction

Task: Let software aid hardware in identifying hot vertices

Challenge: Non-trivial due to sparse distribution of hot vertices in memory

Vertex Properties

- $P_0$
- $P_1$
- $P_2$
- $P_3$
- $P_4$
- $P_5$

Hot

Hot
GRASP: Software-guided reuse-prediction

Task: Let software aid hardware in identifying hot vertices

Challenge: Non-trivial due to sparse distribution of hot vertices in memory

Idea: Leverage prior graph reordering optimization
Optimization: skew-aware graph reordering

Vertices are ordered in memory based on their assigned IDs

Changing vertex order to improve cache locality [IISWC’19]
Optimization: skew-aware graph reordering

Vertices are ordered in memory based on their assigned IDs

Changing vertex order to improve cache locality [IISWC’19]
Optimization: skew-aware graph reordering

Vertices are ordered in memory based on their assigned IDs

Changing vertex order to improve cache locality [IISWC’19]

Original Vertex Order

Graph is unchanged

Degree-based Sort

Hot

Hot

Hot
Optimization: skew-aware graph reordering

Vertices are ordered in memory based on their assigned IDs

Changing vertex order to improve cache locality [IISWC’19]

Graph is unchanged

Degree-based Sort

Hot vertices are placed in a contiguous region
Optimization: skew-aware graph reordering

Vertices are ordered in memory based on their assigned IDs

Changing vertex order to improve cache locality [IISWC’19]

Graph is unchanged

Degree-based Sort

Hot vertices are placed in a contiguous region

Easy to communicate the region boundary to hardware
GRASP: Region-based lightweight interface

1 Preprocessing: Software applies skew-aware reordering
GRASP: Region-based lightweight interface

1 Preprocessing: Software applies skew-aware reordering

Architecturally exposed configuration registers
GRASP: Region-based lightweight interface

1. Preprocessing: Software applies skew-aware reordering

2. Initialization: Software populates configuration registers

Architecturally exposed configuration registers
GRASP: Region-based lightweight interface

1. Preprocessing: Software applies skew-aware reordering

2. Initialization: Software populates configuration registers

3. Initialization: Hardware logically partitions the Property Array

Architecturally exposed configuration registers
GRASP: Region-based lightweight interface

1. Preprocessing: Software applies skew-aware reordering

2. Initialization: Software populates configuration registers

3. Initialization: Hardware logically partitions the Property Array

Architecturally exposed configuration registers

Hot Vertices

Cold Vertices

Region Start

Region End

High Reuse Region

Low Reuse Region

LLC size
GRASP: Region-based lightweight interface

1 Preprocessing: Software applies skew-aware reordering

2 Initialization: Software populates configuration registers

3 Initialization: Hardware logically partitions the Property Array

Software involvement is limited to initialization
GRASP: Reuse prediction at runtime

Cache Access

Does it belong to High Reuse Region?

Yes ➔ High Reuse Hint

No ➔ Low Reuse Hint
GRASP: Reuse prediction at runtime

Cache Access

Does it belong to High Reuse Region?

Yes → High Reuse Hint

No → Low Reuse Hint

Prediction is entirely done in hardware
Outline

➢ Performance of domain-agnostic cache management

➢ Graph analytics

➢ GRASP: domain-specialized cache management
  - Software-guided reuse-prediction
  - Hardware-enforced cache management

➢ Performance evaluation
GRASP: Hardware-enforced cache management

Task: Preferentially cache hot vertices

Challenge: LLC capacity is limited
- Not all hot vertices can fit
GRASP: Hardware-enforced cache management

Task: Preferentially cache hot vertices

Challenge: LLC capacity is limited
- Not all hot vertices can fit

Hot vertices but predicted to have low reuse due to limited LLC capacity
GRASP: Hardware-enforced cache management

Task: Preferentially cache hot vertices

Challenge: LLC capacity is limited
- Not all hot vertices can fit

Hot vertices but predicted to have low reuse due to limited LLC capacity

Requirement: Keep cache management flexible
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

<table>
<thead>
<tr>
<th>LRU Cache management Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Highest Priority</strong> (MRU)</td>
</tr>
<tr>
<td>Way 1</td>
</tr>
<tr>
<td><strong>Lowest Priority</strong> (LRU)</td>
</tr>
<tr>
<td>Way 4</td>
</tr>
</tbody>
</table>
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

LRU Cache management Technique

- Way I
- Way 2
- Way 3
- Way 4

Insertion
GRASP: Preferential but flexible cache management
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

LRU Cache management Technique

Way 1 — Way 2 — Way 3 — Way 4

Insertion

Hit Promotion

Eviction
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

GRASP policies for High Reuse Region

GRASP policies for Low Reuse Prediction

High Reuse

Low Reuse

Cache Access

Hit Promotion

Eviction

Insertion

Way 1

Way 4
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

GRASP policies for High Reuse Region

Hit Promotion

Eviction

Insertion

GRASP policies for Low Reuse Prediction

Insertion
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

GRASP policies for High Reuse Region

- Way 1
- Way 4
- Insertion
- Hit Promotion
- Eviction

GRASP policies for Low Reuse Prediction

- Way 1
- Way 4
- Insertion
- Hit Promotion
GRASP: Preferential but flexible cache management

4-Way Set Associative Cache

GRASP policies for High Reuse Region

GRASP policies for Low Reuse Prediction

High Reuse

Low Reuse

Cache Access

Eviction

Hit Promotion

Insertion
GRASP is simple!

Software
- Off the shelf skew-aware reordering optimization
- Compatible with multiple skew-aware reordering techniques
GRASP is simple!

Software
- Off the shelf skew-aware reordering optimization
- Compatible with multiple skew-aware reordering techniques

Lightweight Interface
- Software configures a pair of registers at initialization
- No software dependency after initialization
GRASP is simple!

Software
- Off the shelf skew-aware reordering optimization
- Compatible with multiple skew-aware reordering techniques

Lightweight Interface
- Software configures a pair of registers at initialization
- No software dependency after initialization

Hardware
- Lightweight address comparison logic to infer the reuse hint
- Trivial policy changes
- Minimal modifications to cache structure – no additional metadata
GRASP is simple!

Software
- Off the shelf skew-aware reordering optimization
- Compatible with multiple skew-aware reordering techniques

Lightweight Interface
- Software configures a pair of registers at initialization
- No software dependency after initialization

Hardware
- Lightweight address comparison logic to infer the reuse hint
- Trivial policy changes
- Minimal modifications to cache structure – no additional metadata

Accelerating graph analytics at minimal cost
Outline

➢ Performance of domain-agnostic cache management

➢ Graph analytics

➢ GRASP: domain-specialized cache management
  - Software-guided reuse-prediction
  - Hardware-enforced cache management

➢ Performance evaluation
Evaluation methodology

Evaluated 25 benchmarks (5 applications x 5 graph datasets)
- Graph applications from the Ligra framework [PPoPP’13]
- Graph datasets are 0.3GB – 8GB in Compressed Sparse Row (CSR) format

Datasets are reordered using DBG [IISWC’19]
- Degree-Based Grouping is state-of-the-art skew-aware reordering

Evaluated on the Sniper simulator [TACO’14]
- 8 Out of Order cores
- 16MB shared LLC (2MB per core)
Domain-agnostic techniques vs GRASP

Graph Workloads (5 applications x 5 DBG-applied datasets)
Domain-agnostic techniques vs GRASP

Graph Workloads (5 applications x 5 DBG-applied datasets)
Domain-agnostic techniques vs GRASP

- No Slowdown
- Up to 10.2% speed-up

Graph Workloads (5 applications x 5 DBG-applied datasets)
More results in paper

Evaluation of pinning-based techniques

Evaluation of GRASP on low-/no-skew graph datasets

Evaluation of GRASP on top of other reordering schemes

… and more
Key take away: one size does NOT fit all

Look beyond domain-agnostic cache management
Thank You

Priyank Faldu

Source code  https://github.com/faldupriyank

Personal website  www.faldupriyank.com

I am on the job market