RECU: Rochester Elastic Cache Utility

Chen Ding

Professor
Department of Computer Science
University of Rochester

Joint work with Jacob Brock and Chencheng Ye. Publication in NPC 2015 and upcoming in IJPP.


Rent Me

Computer Rentals

- IBM Bluemix
  - instant access, launch quickly, scale with demand
- Amazon AWS
  - burstable (T2), balanced (M3), computer optimized (C4)
    - a user may rent 1, 2, 4 or more vCPUs
  - an on-demand instance
    - fixed price, immediately available
  - a spot instance
    - dynamically assigned to the highest bidder

- Jelastic
  - a unit is a Cloudlet
  - CPU usage measured by the number of CPUs per hour

Cache Memory

- Memory on modern multicores
  - most expensive operations
  - most transistors on-chip
- Multicore cache
  - a mixture of private/shared caches
    - IBM Power 8 512KB private L2, 96MB shared ERAM L3
    - Intel Haswell 256KB private L2, up to 20MB shared L3
  - cache is fast memory "operating system"
- Dilemma of sharing
  - maximize or equalize
  - consistency needed for
    - performance tuning and optimization

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

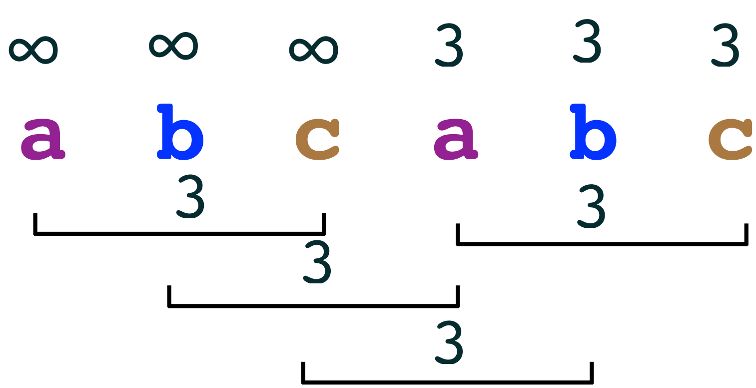
Whitepaper

NVIDIA's Next Generation CUDA™ Compute Architecture:

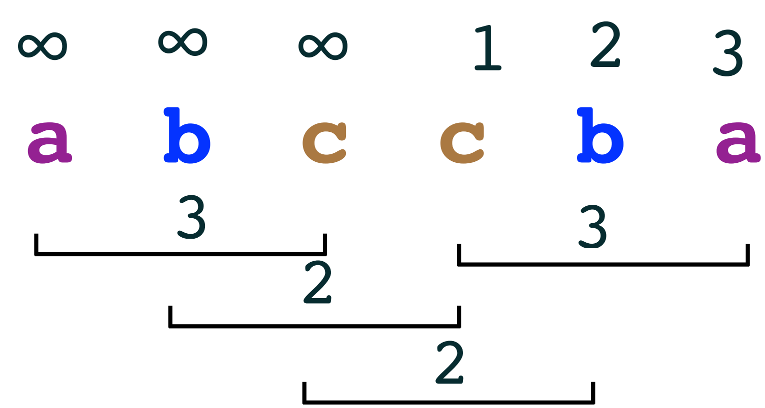
Fermi™


Help!

- *An access*
  - shorter reuse distance -> better locality
- *An execution window*
  - smaller WSS -> better locality
- *A timescale (window length)*
  - smaller footprint -> better locality



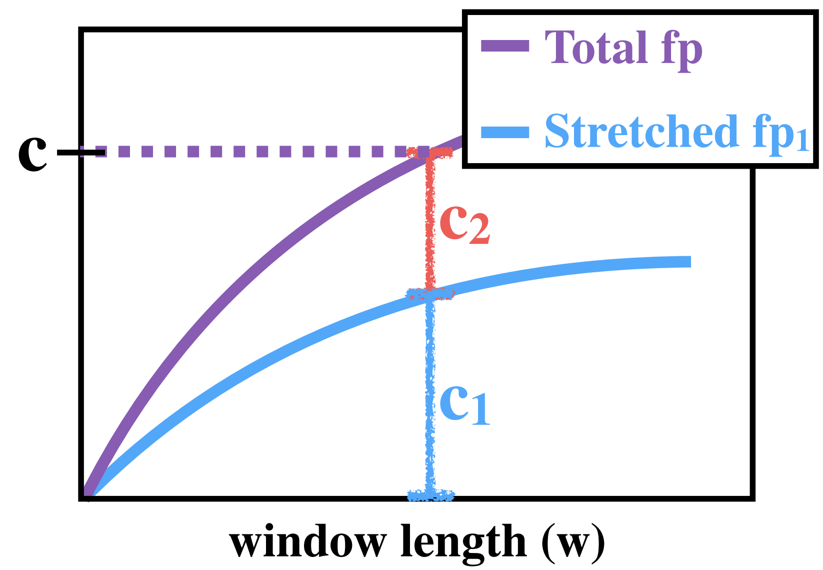
fp(3) = 3



fp(3) = 10/4 = 2.5

High-order Theory of Locality (HOTL)

- Footprint [PPOPP 2008/2011, PACT 2011]
  - average working-set size in time w
- Shared cache
  - composable analysis
  - $fp_{p1,p2}(w) = fp_{p1}(w) + fp_{p2}(w)$
- Miss ratio curve [Xiang+ ASPLOS 2013]
  - derivative of the footprint



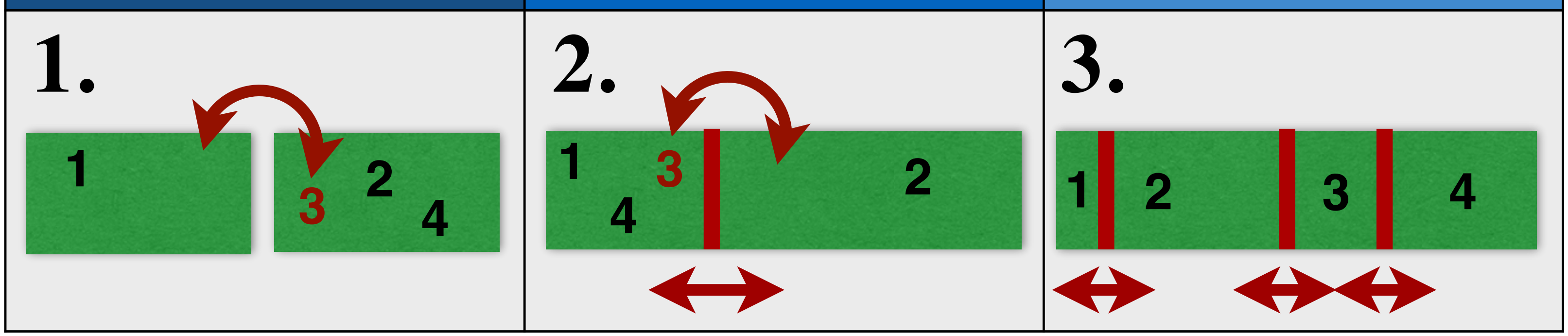
- Xiaoya was a CAS student while doing this research


S
U
P
P
O
R
T

Optimal Cache Sharing

- Social choice [Xie and Loh, 2008]
  - communist (equal partitioning), capitalist (free-for-all), utilitarian
- Economics/game theory [Zahedi and Lee, 2014]
  - sharing incentive, envy free, Pareto optimal

Sharing Only Multiple Caches	Partition-Sharing Single Cache	Partitioning Only



Optimal Partition Sharing [Brock+ ICPP 2015]

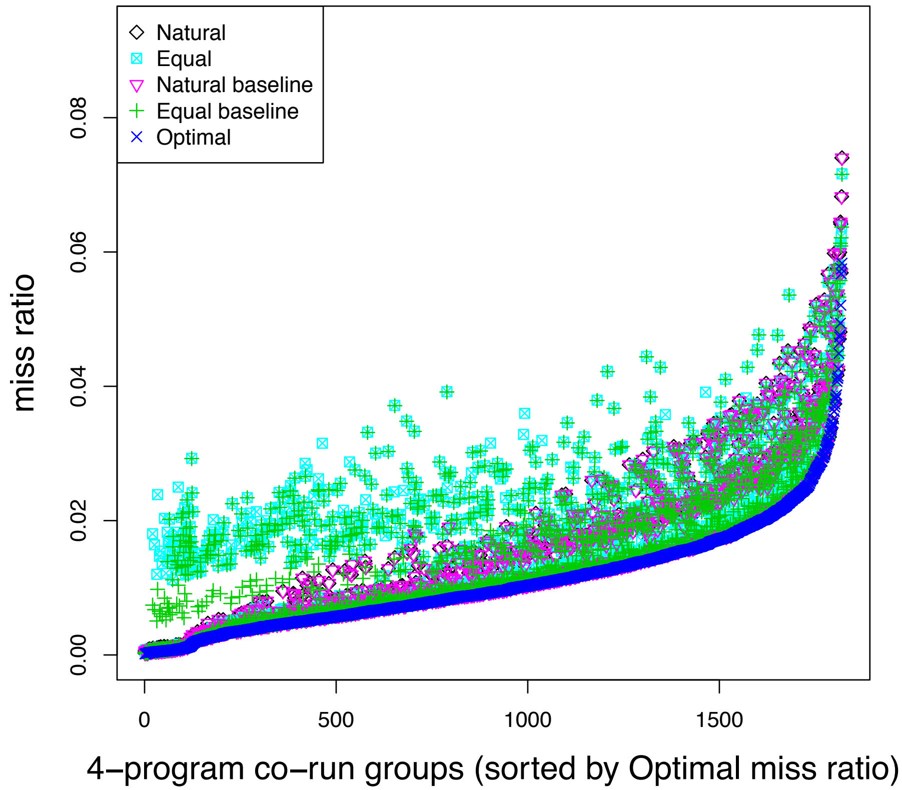
- Footprint theory
  - cache sharing is equivalent to natural cache partitioning
- Optimal partition
  - implies optimal partition sharing
- Dynamic programming
  - to find the optimal partition
  - generalizes previous work
    - Stone et al. 1992, convex miss ratios
    - Suh et al. 2004, piecewise convex
  - supports baseline optimization

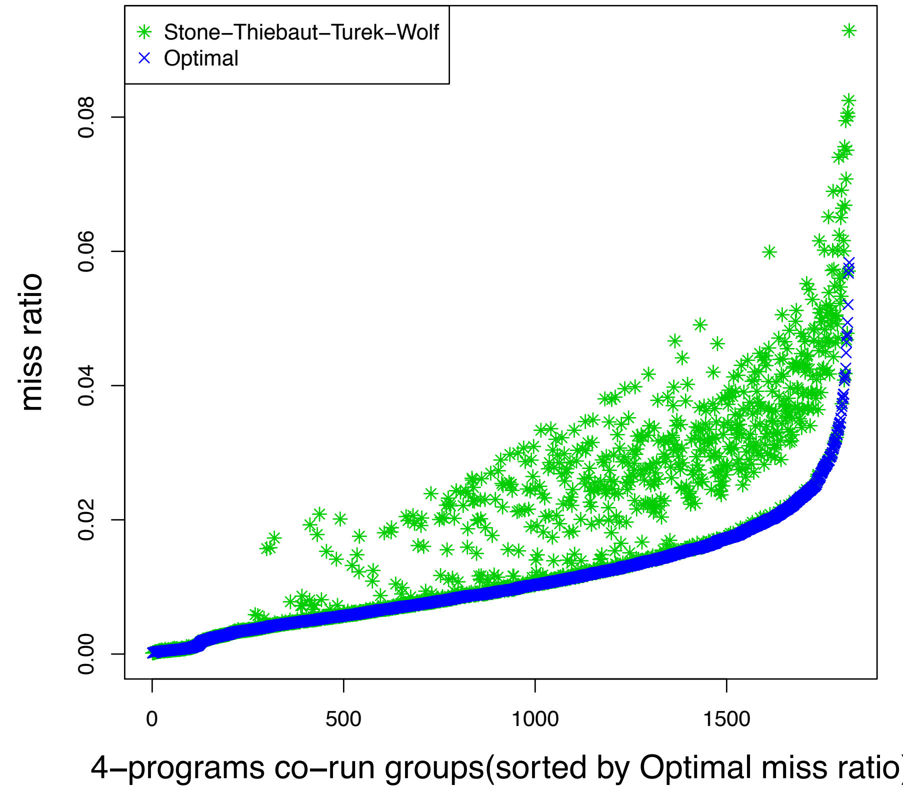
Baseline Optimization

- Rochester elastic cache utility (RECU)
  - a way to combine fairness and synergy
- Equal baseline
  - no higher miss ratio than equal partition
- Natural baseline
  - no higher miss ratio than free-for-all sharing
- Elastic baseline
  - no more than x% higher
- RECU
  - equal or natural baseline
  - miss ratio or cache space

Cache Space based Elasticity [NPC'15, IJPP]

- No elasticity
  - equal partition
- 100% elasticity
  - optimal partition
- Intermediate baselines
  - not as effective as miss ratio elasticity



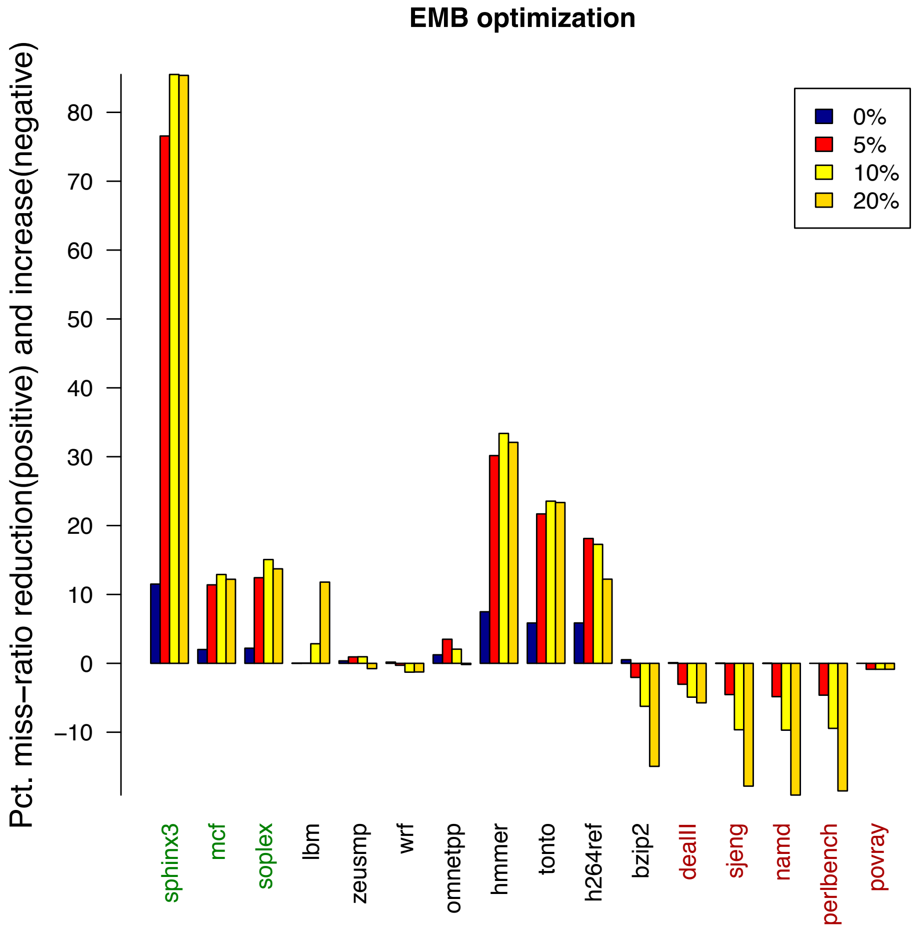


- 16 prog. in SPEC 2006, 1820 4-prog groups
    - 0.9 sec for sampling per program
- 8MB shared cache, 1024 8KB units
- over 45 billion ways to partition for each group
    - 0.2 sec for optimization

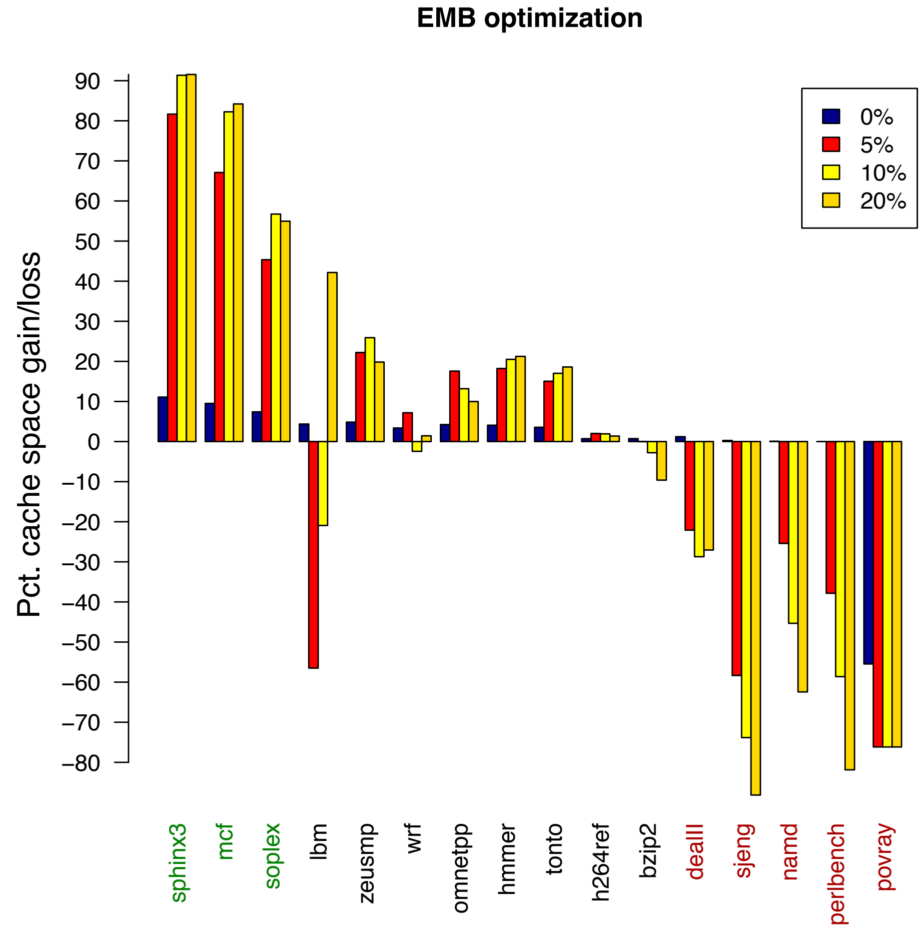
Methods of cache allocation	Overall performance improvement		Individual performance loss			
			Percent prog. degraded by			Worst loss
	Avg	Median	$> 0\%$	$\geq 5\%$	$\geq 10\%$	
RECU with elastic miss ratio baseline						
strict (0%)	6.01%	0%	0%	0%	0%	0%
5%	52.7%	6.63%	61.2%	0%	0%	4.99%
10%	74.4%	9.32%	16.4%	42.4%	0%	9.99%
20%	94.2%	12.5%	58.8%	43.7%	37.2%	19.9%
50%	108%	16.0%	59.1%	52.3%	46.0%	49.9%
100%	115%	22.5%	59.3%	53.3%	47.0%	99.7%
alternatives to RECU (2 types of losses: miss ratio and cache space)						
optimal caching	131%	43.7%	59.1%	53.2%	45.7%	31623%
			59.0%	57.7%	56.8%	98.8%
free-for-all sharing	102%	20.5%	63.9%	58.3%	52.9%	68735%
			64.0%	62.9%	61.6%	98.9%

efficiency

quality of service guarantee



(a) Miss ratio reduction (positive) vs. increase (negative)



(b) Cache space gain vs. loss

RECU Benefits

- Cloud provider
    - baseline optimization
        - 6%, 53%, 108% improvements for 0%, 5%, 50% concession
    - at 50% elasticity
        - half the misses overall
        - close to optimal throughput (108% vs. 131%)
        - at most 50% increase individually
    - free-for-all sharing
        - similar throughput but as high as 32X miss-ratio increase for individual tasks
- Cloud user
    - cost saving from greater cloud efficiency
    - bounded increase in miss count