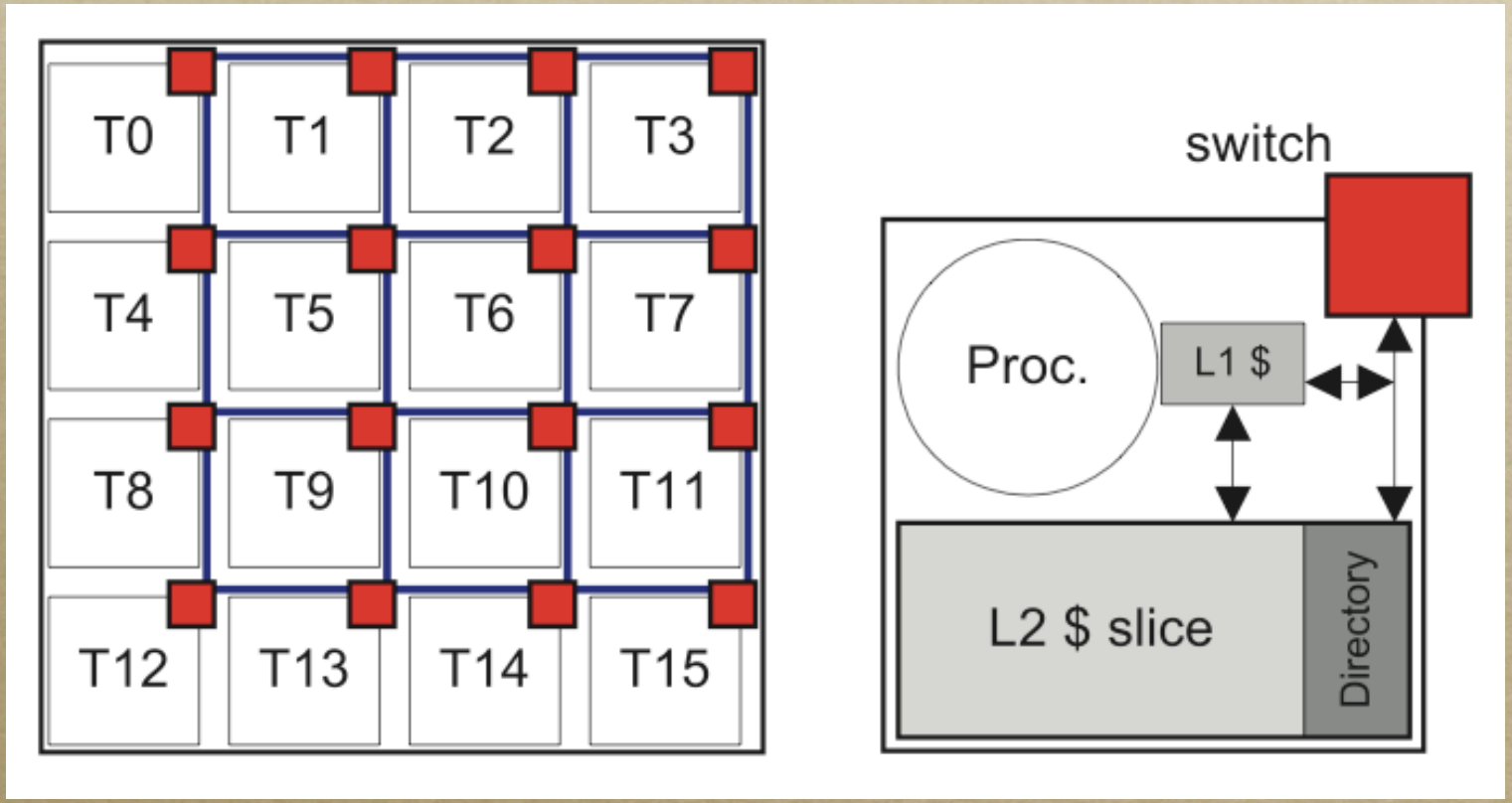
Managing Distributed, Shared L2 Caches through OS-Level Page Allocation

Sangyeun Cho
Lei Jin
(Micro 2006)

Claims

- *Manage L2 cache through OS-level page allocation*
- *Flexible without complex hardware support*
- *Dynamically control data placement and cache sharing*

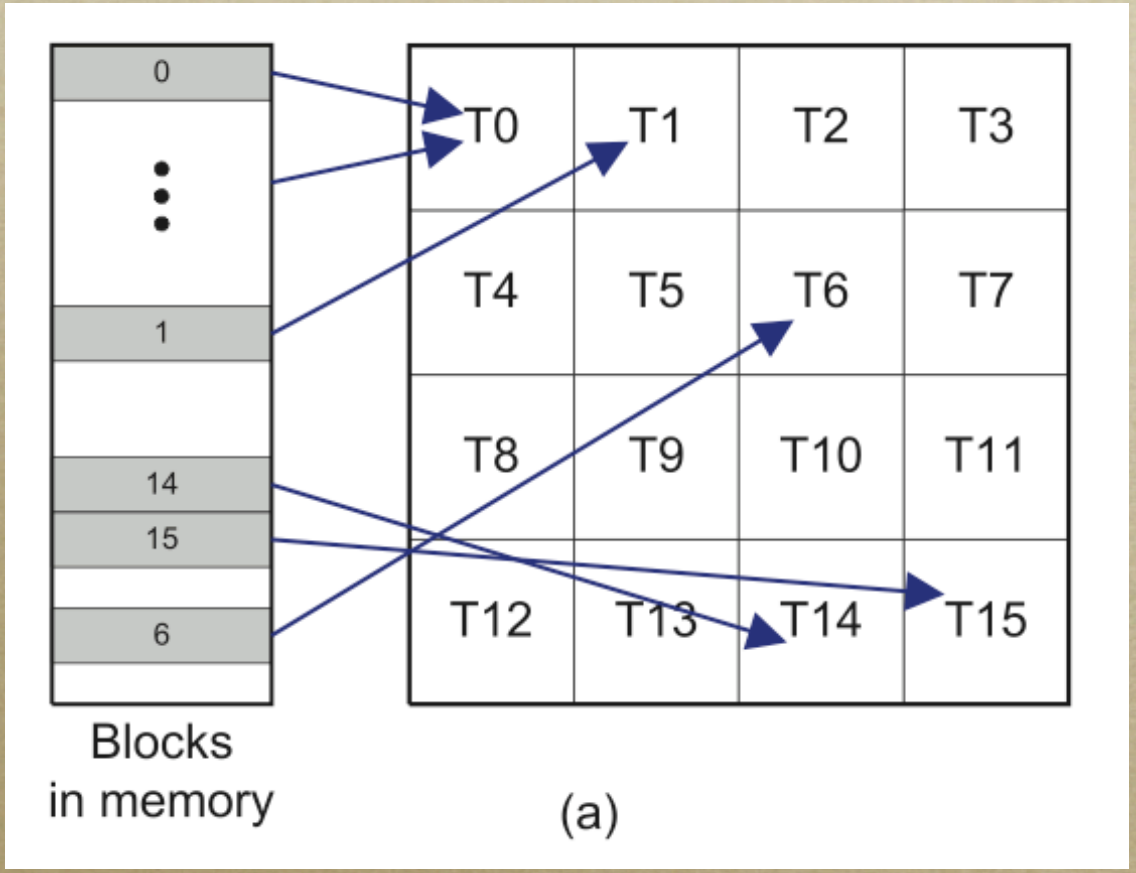
Example Chip and Tile (Core)



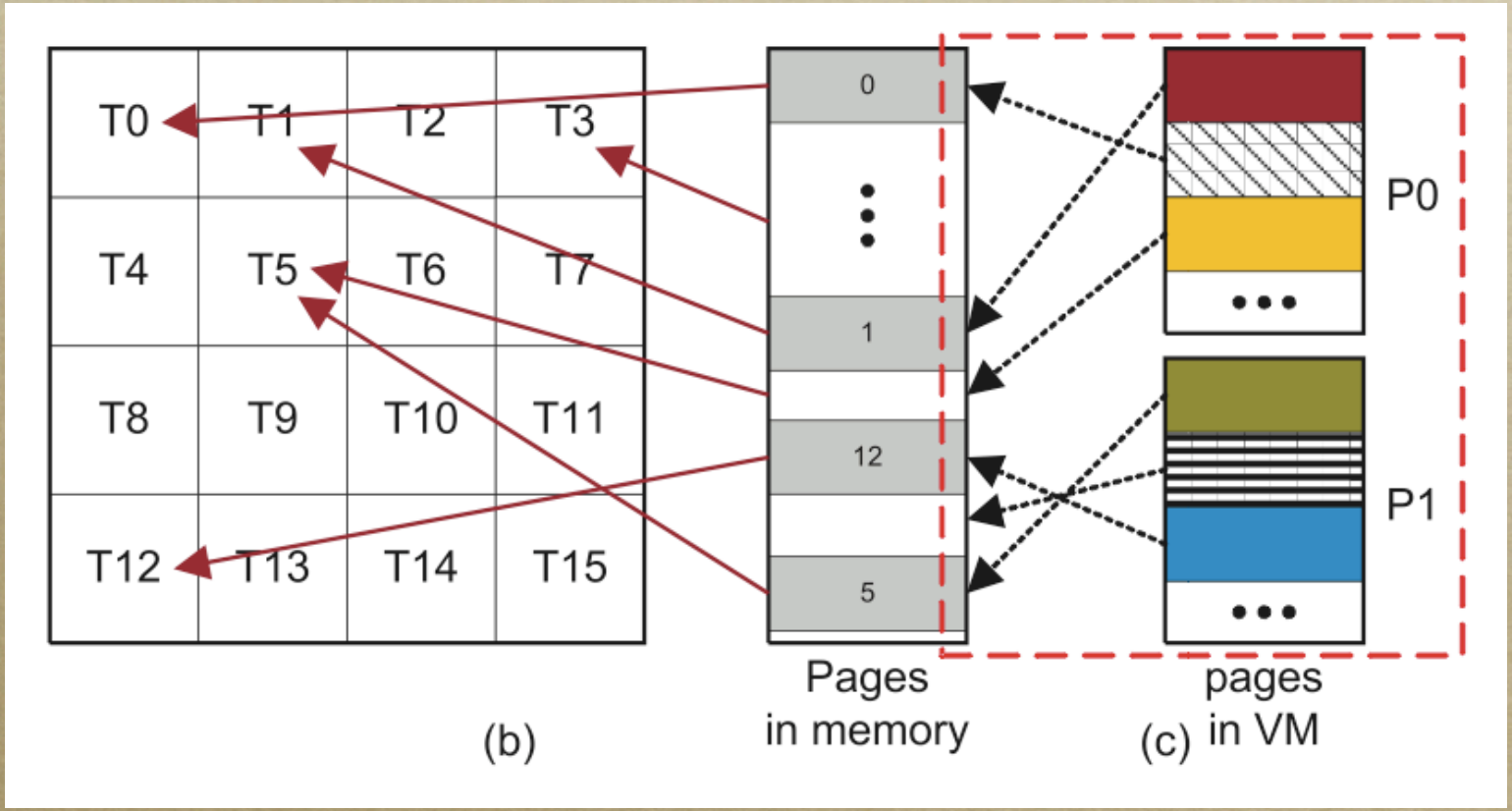
L2 Cache Allocation

- *Traditionally S = A mod N*
- *Proposed change to S = PPN mod N*
- *Allows the OS to chose the virtual to physical mapping (PPN choses the slice)*

Line Granularity



Page Granularity



Congruence Group

- *CGi = {phys page (PPN=j)|pmap(j) = i}*
- *Used to map a physical page to a core.*
- *Convenient to use modulo-N on PPN for pmap*

Caching Schemes

- *Private caching*
  - *OS allocates private pages for Pi running on core i from CGi*
- *Shared caching*
  - *Pages allocated from all congruence groups {CGi} (0<i<N-1)*
  - *Round robin or Random*

Hybrid Caching Scheme

- *Partition {CGi} into K groups (K<N)*
- *Allocate pages from that group for a core within that group*
- *Allows sharing within a group*

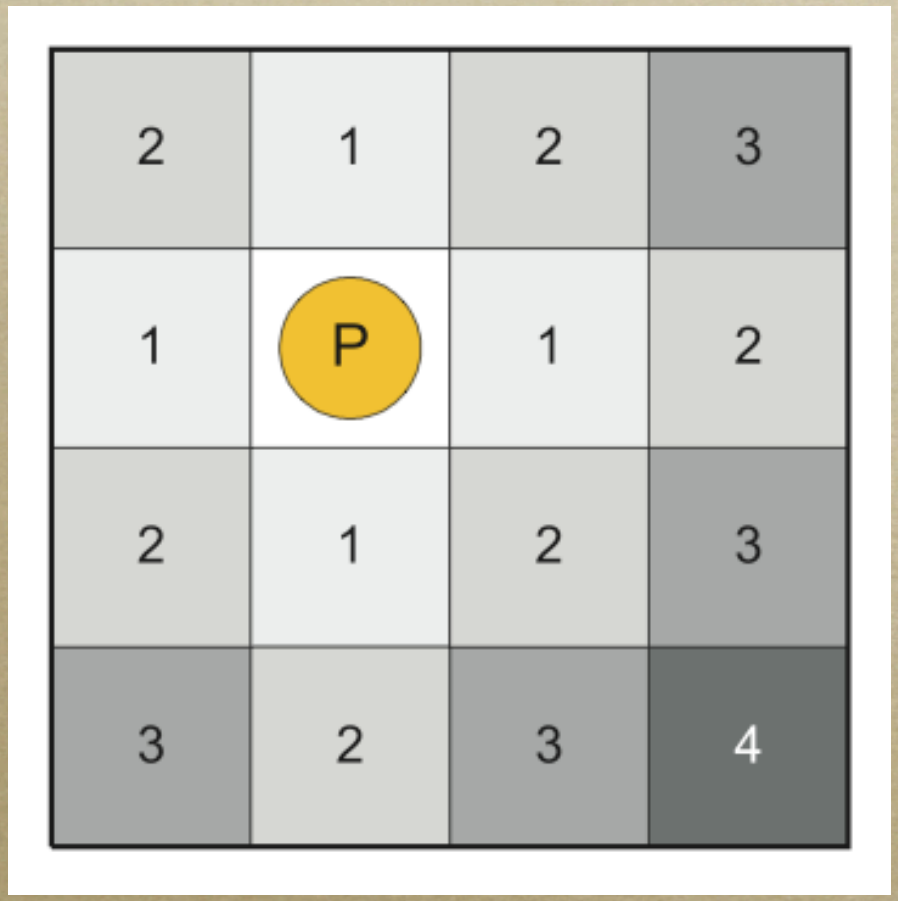
OS Modifications

- *N free lists instead of a single free list*
  - *Depends on the cache scheme*
- *Must consider existing data mappings*
  - *Makes allocation more complex*

Page Spreading

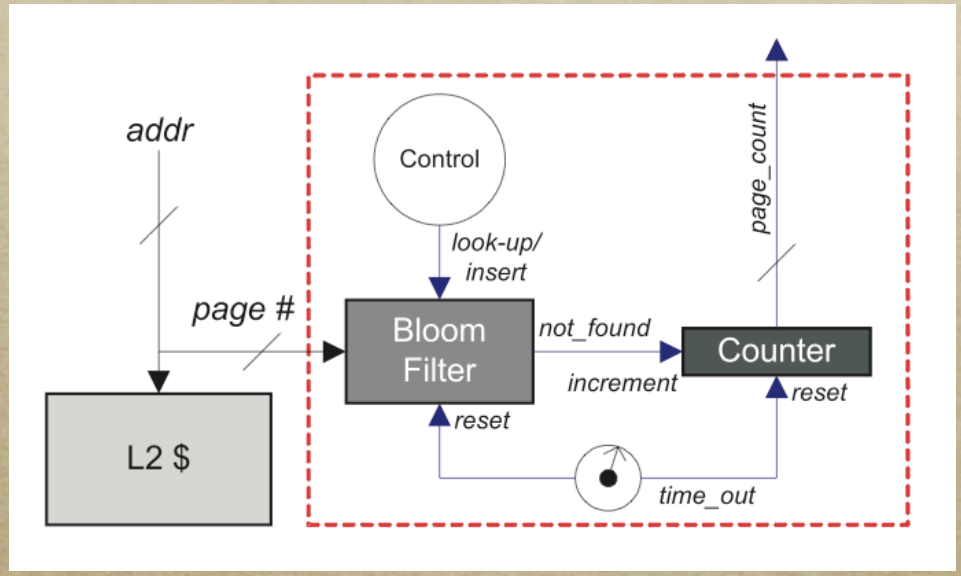
- *When the local L2 slice is too small for the working set*
- *Need to consider data proximity to reduce the number of network hops*
- *Also must consider cache pressure*
  - *number of accessed pages/cache size*

Data Proximity

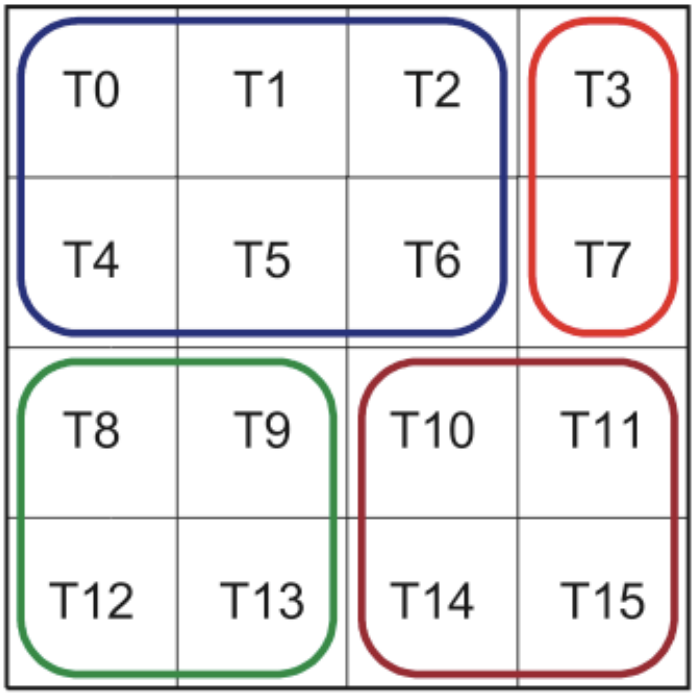


Bloom Filter Monitor

- *Keeps track of pages accessed*
- *Low overhead*
  - *512-kB cache slice*
  - *8-kB page*
  - *512-byte filter*
  - *<0.5% false positive*



Virtual Multicore!



G0 = {T0,T1,T2,T4,T5,T6}

G1 = {T3,T7}

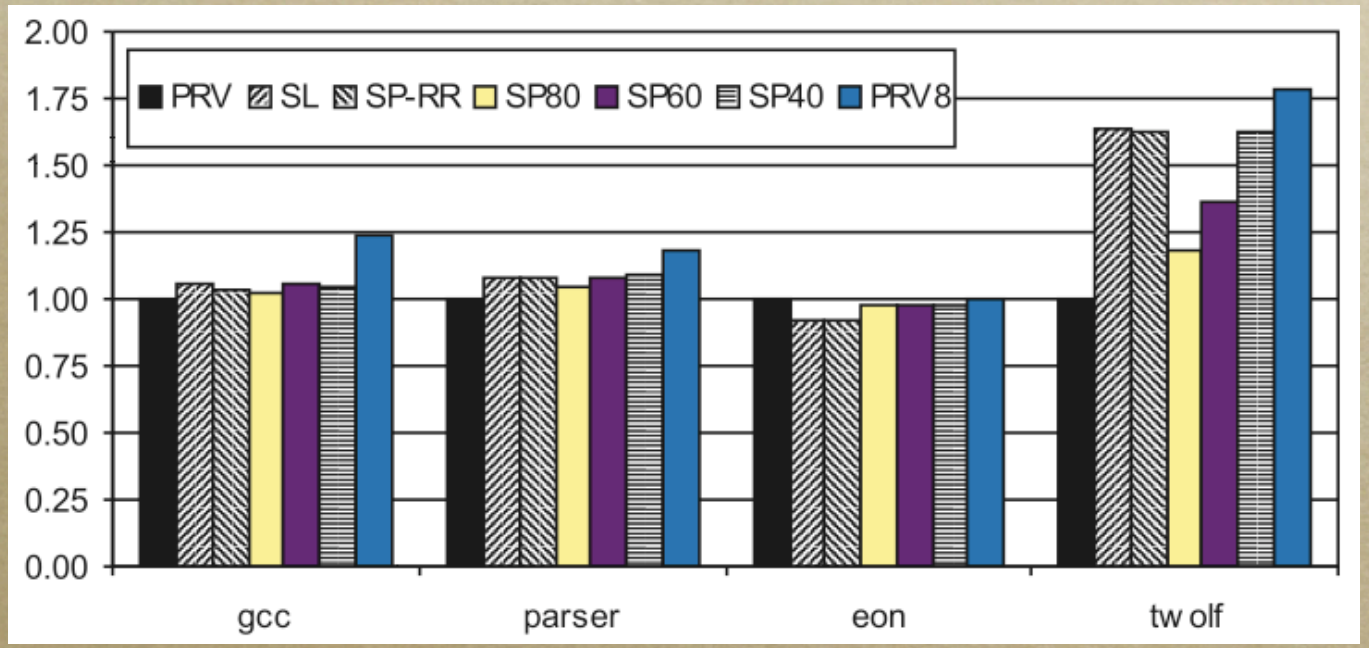
G2 = {T8,T9,T12,T13}

G3 = {T10,T11,T14,T15}

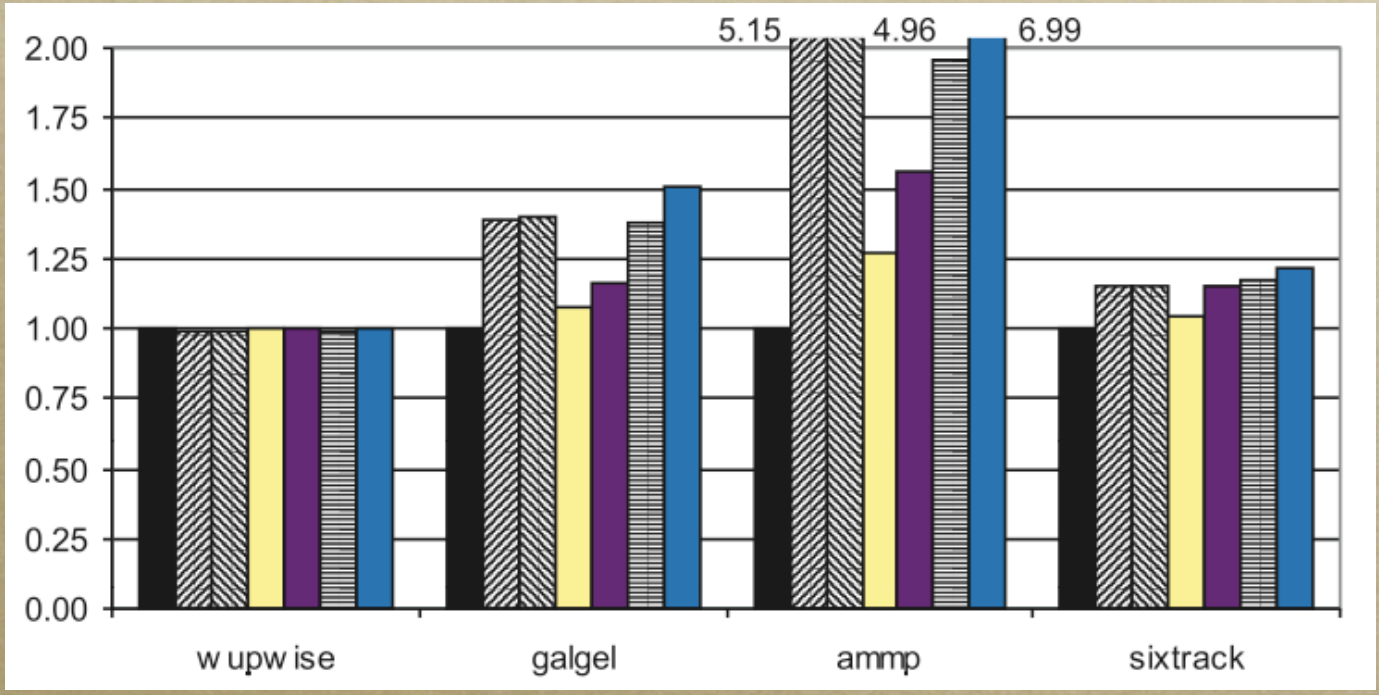
Simulator Setup

- *SimpleScalar*
- *16 tiles (4x4 mesh) (2 cycle hop)*
- *Single issue*
- *16kB L1 I/D caches (1 cycle)*
- *512kB L2 cache slice (8 cycles)*
- *2GB main memory (300 cycles)*

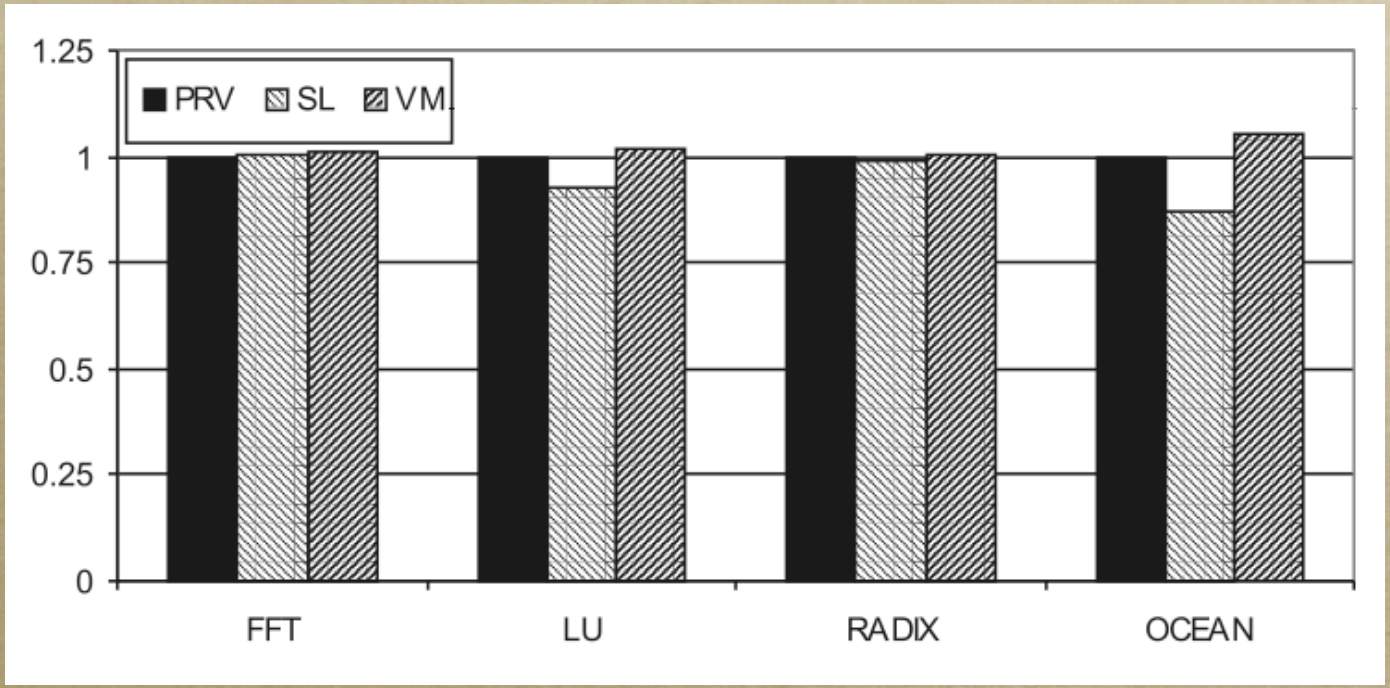
Results


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
PRV
SL
SP-RR
SP80
SP60
SP40
PRV8
gcc
parser
eon
tw olf

Results


5.15
4.96
6.99
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
w upw ise
galgel
ammp
sixtrack

Parallel Workloads



How to Kill Cache Coherence

- *Goal: Reduce overhead of cache coherence*
  - *Also some of the messiness*
- *Granularity issue*
- *OS independent*
- *Lower storage overhead and communication*