Cache Partitioning and Offchip Bandwidth Partitioning/Scheduling

Dr. A. Sahu
CSE, IIT Guwahati

Outline
- Multiprocessor Cache Prefetching
- S/W Helper thread based prefetching
- Cache Partitioning
- Cache and BW Partitioning
- Pre-fetch Aware memory scheduling

LSPS with multiple level
- L2 is shared (LPS)
  - All processor get equal shared depend on request rate
  - Interference with each others
- Equal Share ?

LPPS: Equal Share ?
- Logical partitioning remove interference
- Partition equally in-term of size
  - Fair in-term of size
  - 10 kg Rice per Family
- Partition non equally but depend on requirement/request rate
  - Should be fair in-term of requirement ?
  - Rice Per Family depends on Family size

What are Shared resources ?
- Resources
  - Cache, Memory/BUS Bandwidth
- When to partition
  - Static/Dynamic

Fair in terms of Size
- Logically Partitioned Physically Shared (LPPS)

Resources
- Static
- Dynamics
- Not Required

May be
- Static
- Dynamics
- Not Required
**Cache Partitioning**

- Moinuddin K. Qureshi and Yale N. Patt of UT Austin *Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches*, MICRO 2006

---

**Static Partitioning**

- Profile the applications to be run on the multiprocessor
- Calculate the access behavior or required cache of each application
- Allocate cache before running the program
  - Some require 1MB cache, some require less cache
  - Saturating application: allocating more cache space is no use for them, working set size
  - Cache non-friendly application
- Still aggregate sharing the capacity
Cache Sharing

- CMP and shared caches are common
- Applications compete for the shared cache
- Partitioning policies critical for high performance
- Traditional policies:
  - Equal (half-and-half) ➔ Performance isolation. No adaptation
  - LRU ➔ Demand based. Demand # benefit (e.g. streaming)

Utility of Cache Share: Way Based

Utility \( U^{ab} = \text{Misses with } a \text{ ways} - \text{Misses with } b \text{ ways} \)

Cache and BW Partition

- Weight speedup of four applications running on a 4 core CMP

Motivation behind partitioning

Num ways from 16-way 1MB L2

improve performance by giving more cache to the application that benefits more from cache
Cache and BW Partition

- Weight speedup of four applications running on a 4 core CMP

- Cache and BW partitions can give better speedup

Off-chip BW Partitioning

- Yan Sohlin, Understanding how off-chip Memory Bandwidth Partitioning in Chip multiprocessor Affects System Performance, HPCA 2010

BW partitioning: CPI Model

- Average L2 miss/instruction= h_m
- Average L2 miss latency = t_m
- Extra queuing delay due to contention = Δt_m

\[
CPI = CPI_{L2,m} + h_m t_m
\]

\[
CPI_i = CPI_{L2,m,i} + h_m (Δt_m + t_{m,i})
\]

CPI and H_m

- h_m,i is different for different apps running on diff processor
- It affect the individual CPI_i (thread/app Specific)
  \[
  h_m = (M.A.CPI) / f_{ck}
  \]
  Where  M=miss rate, A=Access frequency
- By combining with
  \[
  CPI_i = CPI_{L2,m,i} + h_m (Δt_m + t_{m,i})
  \]
  \[
  CPI = CPI_{L2,m,i} + h_m (Δt_m + t_{m,i})
  \]
- The sensitivity of Apps performance to queuing delay depends
  - Miss Frequency (MA), CPI_{L2,m,i} and Average Penalty t_{m,i}
**System with out BW partitioning**

- Let $\lambda$ denotes arrival rate of cache miss from all core, So $\lambda = \sum_{i=1}^{P} \lambda_i$, for $i=1$ to $P$
- Little law: Average queue length(N) = Arrival Rate($\lambda_i$) * Service time(T) $N = \lambda_i T$
- Total available BW=B byte/sec, Cache block size =K, service time of a miss = K/B Sec
- So $N = (K/B) \cdot \sum \lambda_i$
- Average waiting time for newly arriving request $W = f_{clk}(K/B).N = f_{clk}(K^2/B^2) \cdot \sum \lambda_i$

**System with BW partitioning**

- Total BW =B, $\beta_i$ =fraction of BW allocated to Proc$_i$
- $\sum \beta_i$=1, BW allocated to Proc$_i$ =B.$\beta_i$
- Service time for request from Proc$_i$ = $T_i = K/B.\beta_i$
- Arrival rate from Proc$_i$ = $\lambda_i = M_i A_i$
- Avg waiting time=$N_i = \lambda_i T_i = M_i A_i (K/B.\beta_i)$
- Exp waiting time $\Delta t_{m,i} = N_i T_i f_{clk} = M_i A_i K^2 f_{clk} / (B \cdot \beta_i^2 B^2)$
- CPI$_i$ is $CPI_i = CPI_{L2M,i}/(1 - [M_i A_i t_{m,i}/f_{clk}] - [K^2 \cdot \sum \lambda_i A_i^2]/B^2)$
- is Special of Prev CPI$_i$ with $\beta_i = (M_i A_i)/\sum \lambda_i A_i$

**Extra queuing delay to request of thread**

- As request arrive according to arrival rate from diff processor
- Probability of request from thread, is shared with total arrival rate $= (M_i A_i / \sum \lambda_i A_i)$
- So expected waiting time $\Delta t_{m,i} = W / ((M_i A_i / \sum \lambda_i A_i))$
  - $= (f_{clk}(K^2/B^2) \cdot \sum \lambda_i A_i / ((M_i A_i / \sum \lambda_i A_i))$
  - $= (\sum \lambda_i A_i \cdot f_{clk} K^2)/M_i AB^2$
- CPI$_i$ is $CPI_i = CPI_{L2M,i}/(1 - [M_i A_i t_{m,i}/f_{clk}] - [K^2 \cdot \sum \lambda_i A_i^2]/B^2))$

**Optimum BW partitioning**

- WeightedSpeedup $= \sum IPC_i/IPC_{alone, i} = \sum CPI_{alone, i}/CPI_i$
  - $= \sum CPI_{alone, i}/CPI_{L2M,i} \cdot (1 - [M_i A_i t_{m,i}/f_{clk}] - [K^2 \cdot \sum \lambda_i A_i^2]/B^2))$
- How to optimize?
  - Minimize third term $K^2 \cdot \sum \lambda_i A_i^2 / \beta_i \cdot B^2$
  - With $\sum \beta_i = 1$
- Mathematical manipulation optimum $\beta_i$ is

$$\beta_i = (M_i A_i)^{2/3}/(\sum \lambda_i A_i)^{2/3}$$

For detail please refer paper

---

**Partition Method**

Diagram showing observed, calculated, and allocated cache and BW partition method.