Protection and Utilization in Shared Cache Through Rationing

Raj Parihar, Jacob Brock*, Chen Ding*, Michael C. Huang

Dept. of Electrical & Computer Engineering

*Department of Computer Science

University of Rochester, Rochester, NY
Motivation

- Last level cache sharing among competing programs is common
  - Significant performance loss due to co-run interference: up to 65%

- Baseline (naive) partitioning/sharing policies: two extremes
  - **Hard partition**: every program gets equal cache share
  - **Free-for-all**: programs can use any portion of cache

- Rationing: To achieve protection and utilization both
  - Every prog is assigned a portion of cache, known as **ration**
  - A prog can exceed ration only if another prog is not using its ration
  - If a prog is using its ration, it can not be taken away by peer progs
Hardware Support to Enable Rationing

- **Access-bit**: To detect unused ration (inspired from OS), 1 per blk
- **Ration counter**: To track the current usage of a prog, 1 per core
- Additional storage overhead: <1% of total cache storage
Comparison with Baseline Policies

<table>
<thead>
<tr>
<th>Time</th>
<th>Core1: xy xy xy ...</th>
<th>Core2: ab abc abc ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>y x</td>
<td>b a</td>
</tr>
<tr>
<td>3</td>
<td>y x</td>
<td>c b</td>
</tr>
<tr>
<td>4</td>
<td>x y a</td>
<td>c b y a</td>
</tr>
<tr>
<td>5</td>
<td>y x b a</td>
<td>y a x</td>
</tr>
<tr>
<td>6</td>
<td>y x b a</td>
<td>y b x</td>
</tr>
</tbody>
</table>

### Communist

<table>
<thead>
<tr>
<th>Time</th>
<th>Core1: xy xy xy ...</th>
<th>Core2: ab abc abc ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>y x y</td>
<td>b y a x</td>
</tr>
<tr>
<td>3</td>
<td>y x c b</td>
<td>c b y a</td>
</tr>
<tr>
<td>4</td>
<td>x y a c</td>
<td>a x c b</td>
</tr>
<tr>
<td>5</td>
<td>y x b a</td>
<td>b y a x</td>
</tr>
</tbody>
</table>

### Capitalist

<table>
<thead>
<tr>
<th>Time</th>
<th>Core1: xy xy xy ...</th>
<th>Core2: ab abc abc ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>b y a x</td>
<td>b y a x</td>
</tr>
<tr>
<td>3</td>
<td>c b y a</td>
<td>c b y a</td>
</tr>
<tr>
<td>4</td>
<td>a x c y</td>
<td>a x c y</td>
</tr>
<tr>
<td>5</td>
<td>b y a x</td>
<td>b y a x</td>
</tr>
</tbody>
</table>

### Rationing

- **Data array**: 
- **Access bits**: 11100
- **Ration cntr**: 11

**Resource Protection**
## Comparison with Baseline Policies

<table>
<thead>
<tr>
<th>Time</th>
<th>Communist</th>
<th>Capitalist</th>
<th>Rationing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x a</td>
<td>a x</td>
<td>a x</td>
</tr>
<tr>
<td></td>
<td>y x b a</td>
<td>b y a x</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td></td>
<td>y x c b</td>
<td>c b y a</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>x y a c</td>
<td>a x c b</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>y x b a</td>
<td>b y a x</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

**Resource Protection**

<table>
<thead>
<tr>
<th>Time</th>
<th>Communist</th>
<th>Capitalist</th>
<th>Rationing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x a</td>
<td>a x</td>
<td>a x</td>
</tr>
<tr>
<td></td>
<td>x b a</td>
<td>b a x</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td></td>
<td>x c b</td>
<td>c b a x</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>x a c</td>
<td>a x c b</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>x b a</td>
<td>b a x c</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

**Capacity Utilization**


Experimental Analysis

**Capacity utilization:** co-run with a low-pressure peer (*eon*)

![Graph showing IPC Norm. to solo run w/ 512KB cache for various benchmarks such as gzip, apsi, etc., with normalized values for Communist, Capitalist, Rationing, etc., and a note on SPEC CPU 2000: Co-run with *eon* (2 cores, 1 MB L2 cache).]
Experimental Analysis

- **Capacity utilization**: co-run with a low-pressure peer (eon)

- **Resource protection**: co-run with a high-pressure peer (mcf)
Summary

- Rationing protects the ration of each program while at the same time finds and utilizes unused ration

- Rationing support can be added on top of existing cache architecture with minimal additional hardware

- Rationing provides an integrated design for cache sharing and allows software-hardware collaboration