Project 2

Cache device
Project overview

• Implement a cached storage system
  – A smaller device caches frequently used blocks in a larger device
  – Use device-mapper

• Due: 20th May 23:59
• Write through
  – Write req. returns when BS finishes update

• Write back
  – Write req. returns when cache updated

• Write around
  – Cache is filled when data are read
Replacement policy

- **Fundamental rule**
  - Before replacing a data block in cache, it must have been updated to the backing device.

- **FIFO**: simplest approach → sequential replacement

- **LRU**: in a unit of cache chunk (for efficiency)
Cache meta-data management

- **Objective:** address translation
  - Acquiring the block number of cached block

- **Data structure**
  - Hash table
  - Search tree (B-Tree)
  - Journal (logging)

- **Synchronization** of in-memory and on-disk meta-data
  - Write-through caching simplifies meta-data management
Other factors to consider

- **Caching unit:** sector, page, block, ...

- **Sequential I/O**
  - HDDs show good sequential I/O performance
  - Caching small I/O requests only
### Existing caching solution in Linux kernel

<table>
<thead>
<tr>
<th></th>
<th>Flashcache</th>
<th>Bcache</th>
<th>DM-cache</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer</strong></td>
<td>Facebook (Mohan Srinivasan)</td>
<td>Google (Kent Overstreet)</td>
<td>RedHat (Joe Thornber)</td>
</tr>
<tr>
<td><strong>Allocation</strong></td>
<td>Set-relative</td>
<td>Global</td>
<td>Global</td>
</tr>
<tr>
<td><strong>Metadata</strong></td>
<td>Set associative array</td>
<td>B-tree</td>
<td>B-tree</td>
</tr>
<tr>
<td><strong>Replacement</strong></td>
<td>FIFO, LRU</td>
<td>FIFO, LRU, RANDOM</td>
<td>Multi-Queue, Clean</td>
</tr>
<tr>
<td><strong>Cache modes</strong></td>
<td>WB / WT / WA</td>
<td>WB / WT / WA</td>
<td>WB / WT</td>
</tr>
<tr>
<td></td>
<td>WB : writeback,</td>
<td>WT : writethrough,</td>
<td>WA: writearound</td>
</tr>
<tr>
<td><strong>Sequential I/O size</strong></td>
<td>Configurable (512 blocks)</td>
<td>Dynamic (Sequential_io_avg)</td>
<td>Configurable (512KB)</td>
</tr>
</tbody>
</table>
Project goal

• **Base-line**
  – Write through caching
  – FIFO replacement
  – No on-disk meta-data management

• **Extension** (bonus)
  – Write back, write around caching
  – LRU replacement (or other policy)
  – On-disk meta-data management
  – Or other ideas of your own!