

**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: 20<sup>th</sup> May 23:59



# Caching policy

- **Write through**

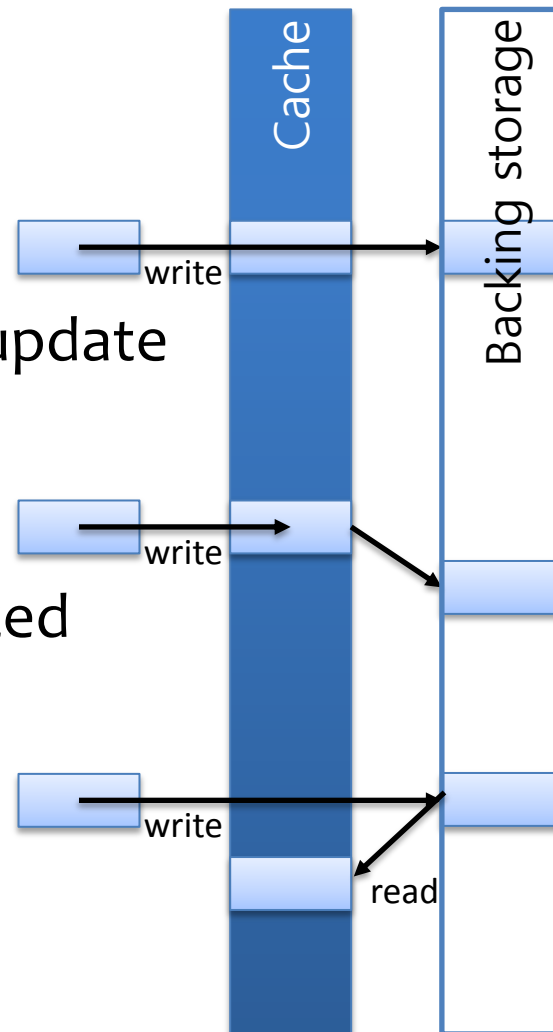
- Write req. returns when BS finishes update

- **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



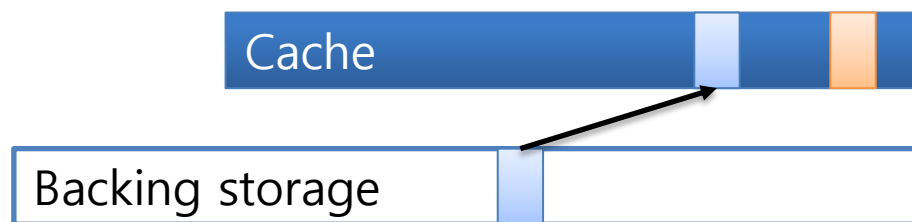
- **LRU**: in a unit of cache chunk (for efficiency)





# 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



# Example

- Existing caching solution in Linux kernel

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



# 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!