Caching Writes

On a write, some applications assume that data makes it through the buffer cache and onto the disk

➡ As a result, writes are often slow even with caching

OSes typically do write back caching

- Maintain a queue of uncommitted blocks
- Periodically flush the queue to disk (30 second threshold)
- If blocks changed many times in 30 secs, only need one I/O
- If blocks deleted before 30 secs (e.g., /tmp), no I/Os needed

✓ Unreliable, but practical

- On a crash, all writes within last 30 secs are lost
- Modern OSes do this by default; too slow otherwise
- System calls (Unix: fsync) enable apps to force data to disk

Read Ahead

Many file systems implement "read ahead"

- FS predicts that the process will request next block
- FS goes ahead and requests it from the disk
  - This can happen while the process is computing on previous block
  - Overlap I/O with execution
  - When the process requests block, it will be in cache
  - Compliments the disk cache, which also is doing read ahead

✓ For sequentially accessed files can be a big win

- Unless blocks for the file are scattered across the disk
- File systems try to prevent that, though (during allocation)