TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Cooperative I/O: A novel I/O semantics for energy-aware applications

Andreas Weißel, Björn Beutel, Frank Bellosa

Dresden, 2010-07-13

Motivation

- Energy-aware computing
  - Mobile devices / notebooks: enhance battery lifetime
  - Servers: limit energy cost
- Focus (<2002): OS-level improvements
  - Turn off devices / CPUs
  - Problems:
    - power management cost
    - advance knowledge needed

Power-saving hard disks

- ATA defines 4 power modes (active, idle, standby, sleep)
- Switching between modes costs energy
- Decision when to switch depends on the estimated power savings
  - Device-specific break-even point
    → makes OS-level decisions hard

Cooperative I/O

- Idea: applications give hints to OS
- Timeout: how long can I wait until the request must be started?
  - Example: video player filling a buffer
- Cancel: Drop request if timeout is reached?
  - Example: Auto-save in text editor

Whole-system integration

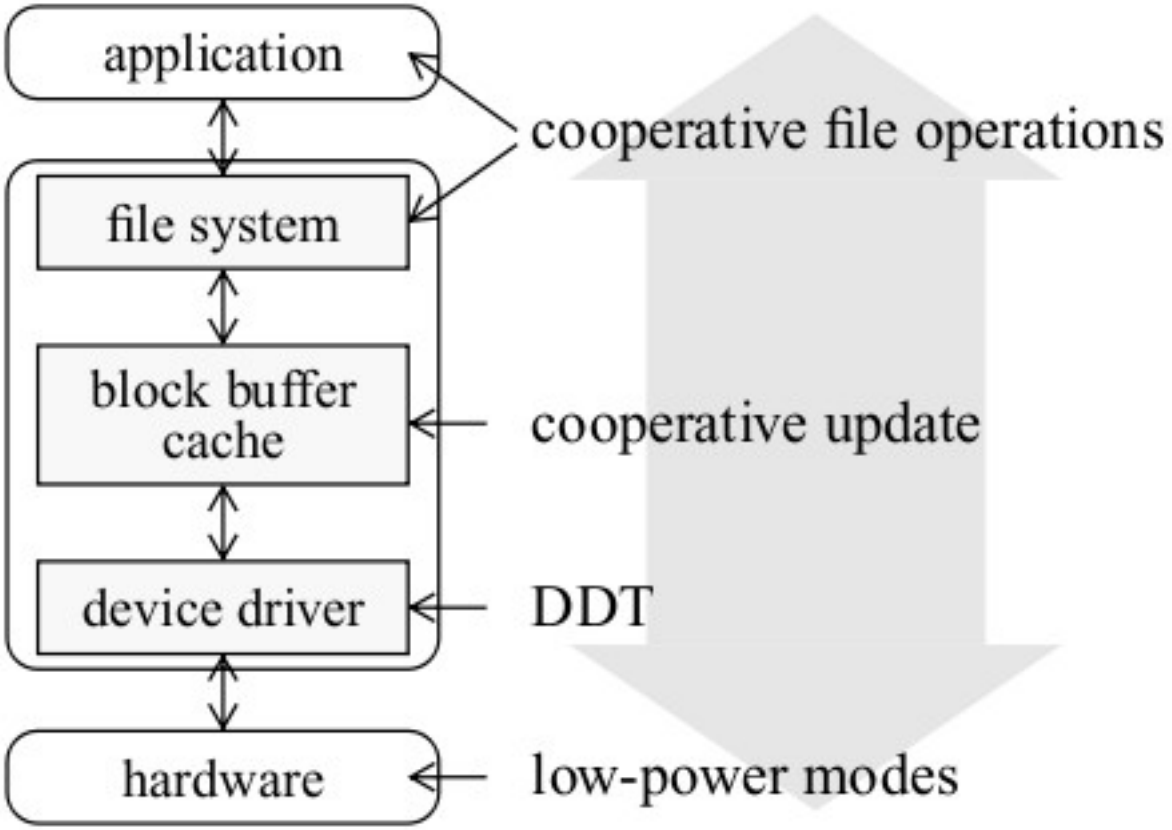

Figure 1: components of Coop-I/O

Cooperative file operations

- New Linux system calls:
  - coop_open
  - coop_read
  - coop_write
- Buffer cache updates easy for reading
  - If disk is in low-power mode, wait until timeout occurs or another request spins up the disk.
  - If timeout hits and cancel flag is set, cancel.

Cooperative writing

- write() usually works on buffer cache
- In-kernel update thread to write back dirty buffers to disk
  - Make this task cooperative

- 1st problem: writing may induce a read op

- 2nd problem: cancellation of a write-induced read after multiple modifications within the cache → *need transactional semantics*

Early commit strategy

- Instead of implementing transactions, try to commit write ops as early as possible
- Wait until another request spins up disk and then write back dirty buffers immediately
- If timeout hits, read cancel flag
  - Cancel
    → no other buffers? drop all dirty buffer
    → other buffers? write back w/ others
  - No cancel → write back all dirty buffers

Drive-specific cooperative update

- Write request batching
- Write back all dirty buffers every period
- Piggy-back on read requests
- Updates performed per drive
- Write back before switching to low-power mode

Implementation in Linux

- Modifications to
  - VFS
  - Ext2
  - Block layer
  - IDE driver

- Energy is a cross-cutting concern.

  OR

  Break-up all the nice abstractions.

Determining disk break-even

IBM Travelstar 15GN
break-even time = 8.7s

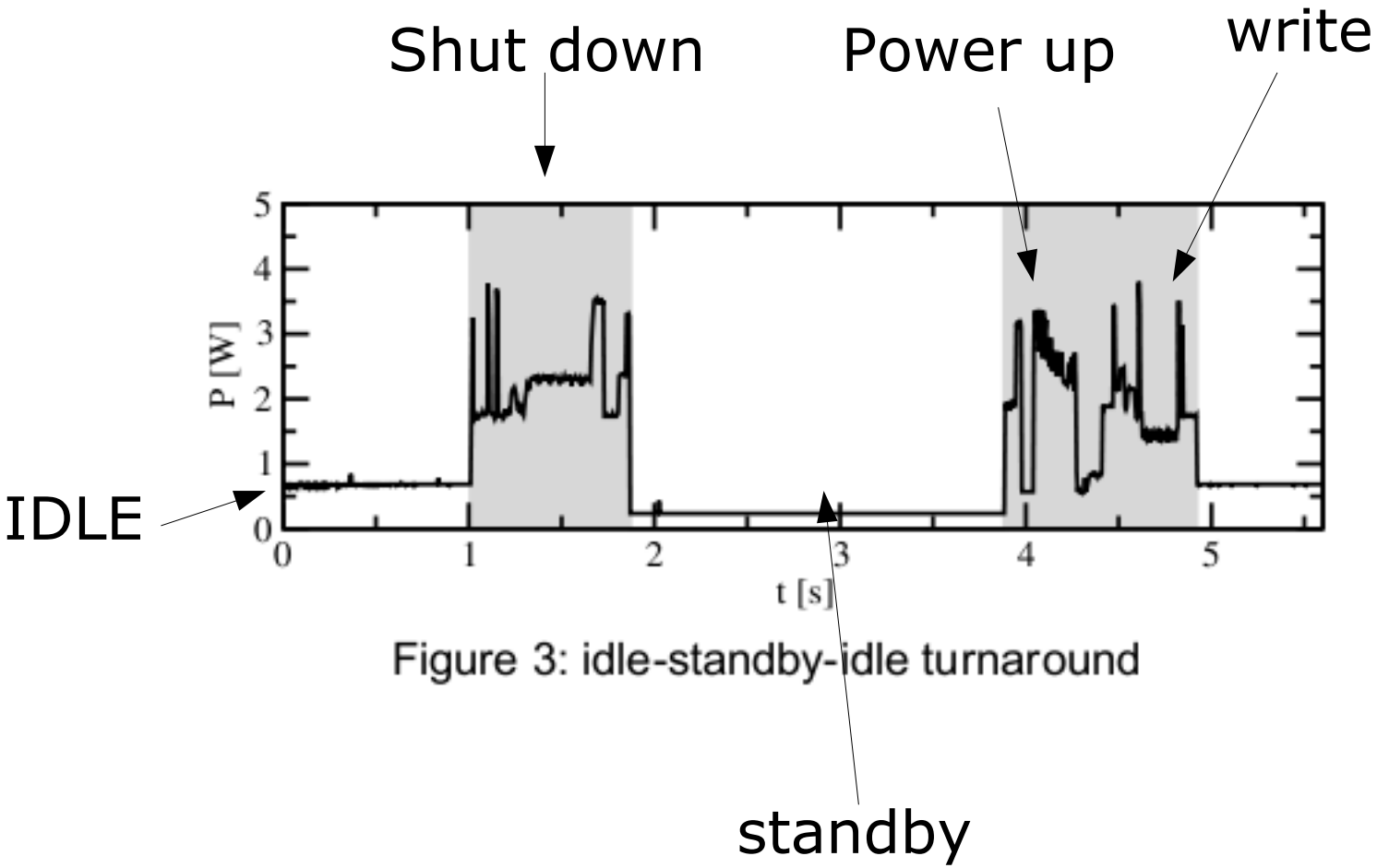

Figure 3: idle-standby-idle turnaround

TECHNISCHE UNIVERSITÄT DRESDEN

Power saving policies: MP3 player

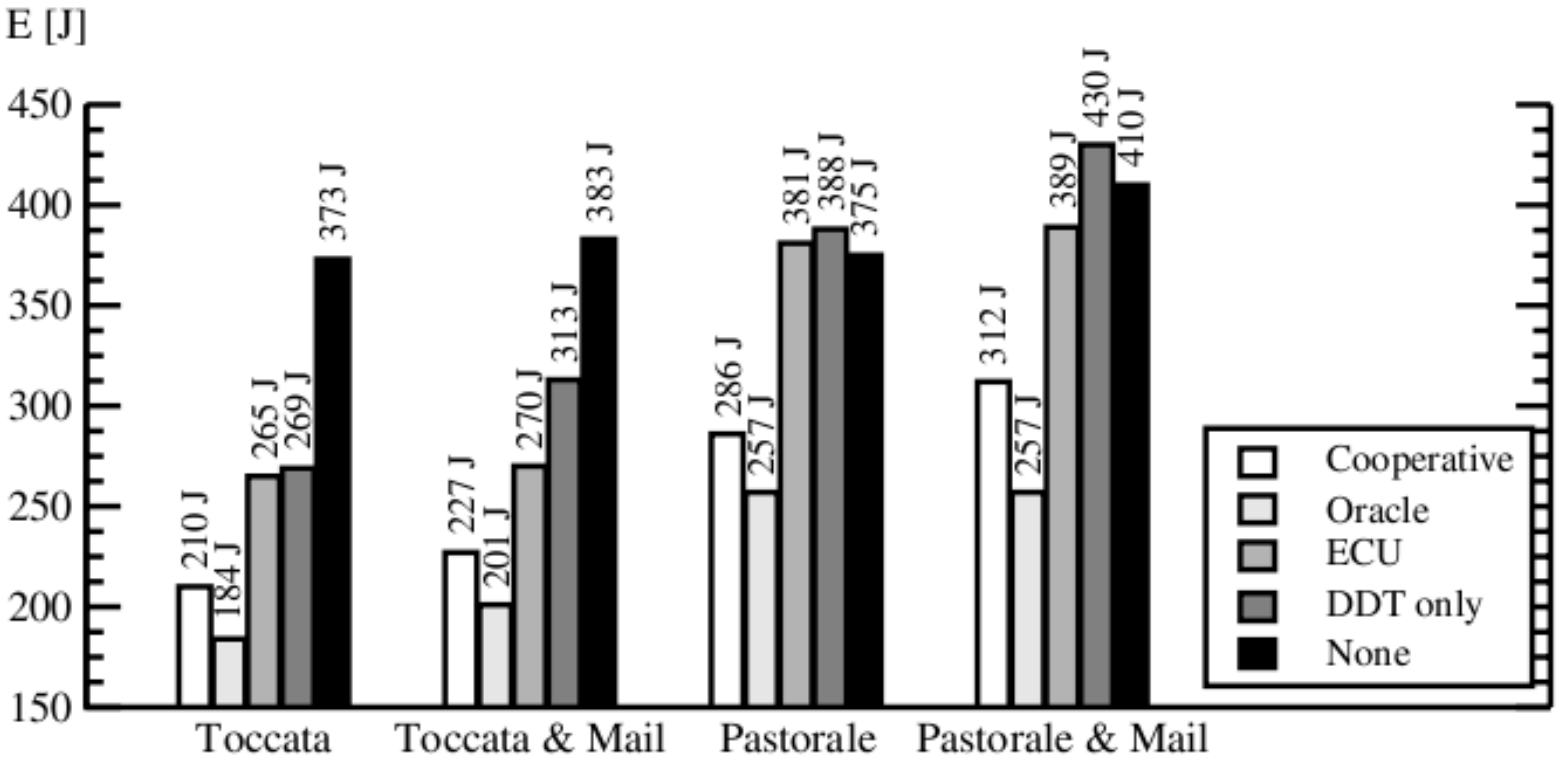

Figure 4: comparison of policies

Power saving policies: µ-benchmark

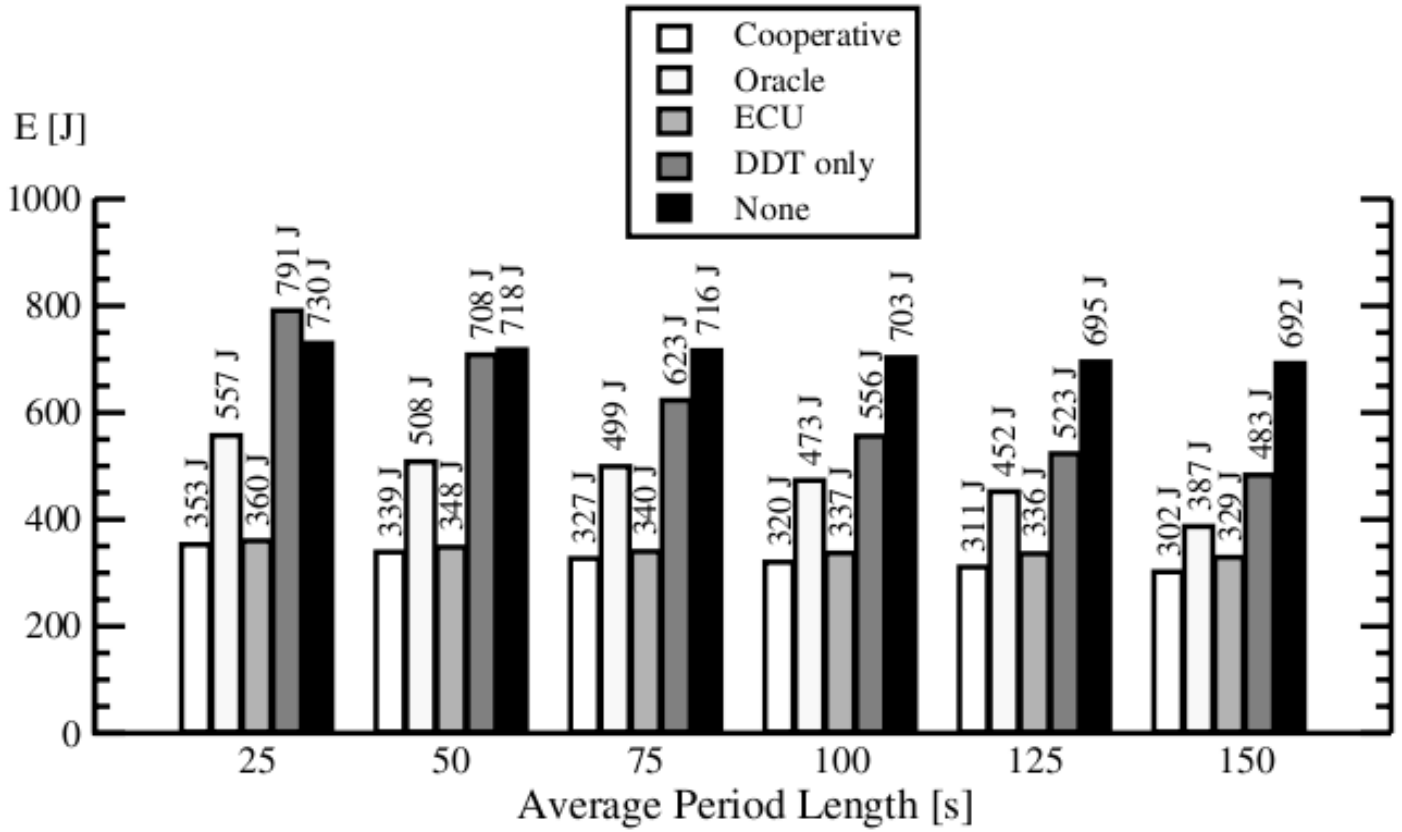

Figure 8: writes with varying average period length

Discussion

- Do the examples given require a solution as complex as this one?
  - Auto save → increase fsync period at user level?
  - Cron jobs etc. → i/o priorities?
- Are the examples valid?
  - Media player filling buffer → wants to specify when the request is completed, not when it is sent.
- Is this viable to implement for all available device classes and protocol stacks?