Analyzing and Optimizing Linux Kernel for PostgreSQL

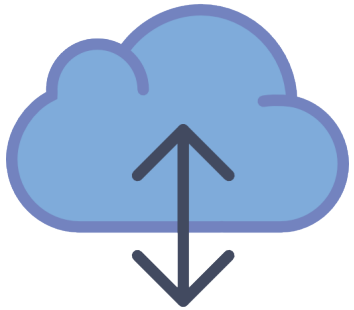
Sangwook Kim

PGConf.Asia 2017

apposha

Sangwook Kim

- **Co-founder and CEO @ Apposha**
- Ph. D. in Computer Science



- **Cloud/Virtualization**
- SMP scheduling [ASPLOS’13, VEE’14]
- Group-based memory management [JPDC’14]

- **Database/Storage**
- Non-volatile cache management [USENIX ATC’15, ApSys’16]
- Request-centric I/O prioritization [FAST’17, HotStorage’17]

What We Do

Amazon Aurora

Fully-managed &
performance-optimized for
MySQL and PostgreSQL

for

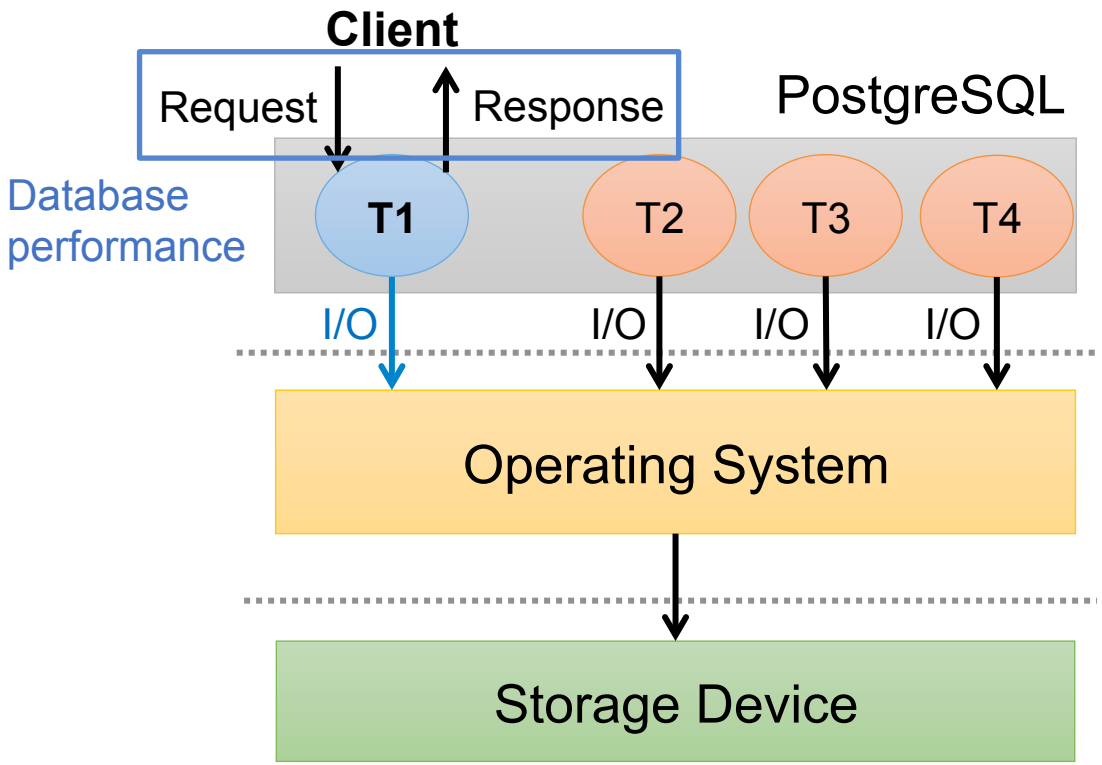
mongoDB®

Fully-managed &
performance-optimized for MongoDB
No special H/W support
No code changes for DB engine

Contents

- **Background tasks**
- Full page writes
- Parallel query

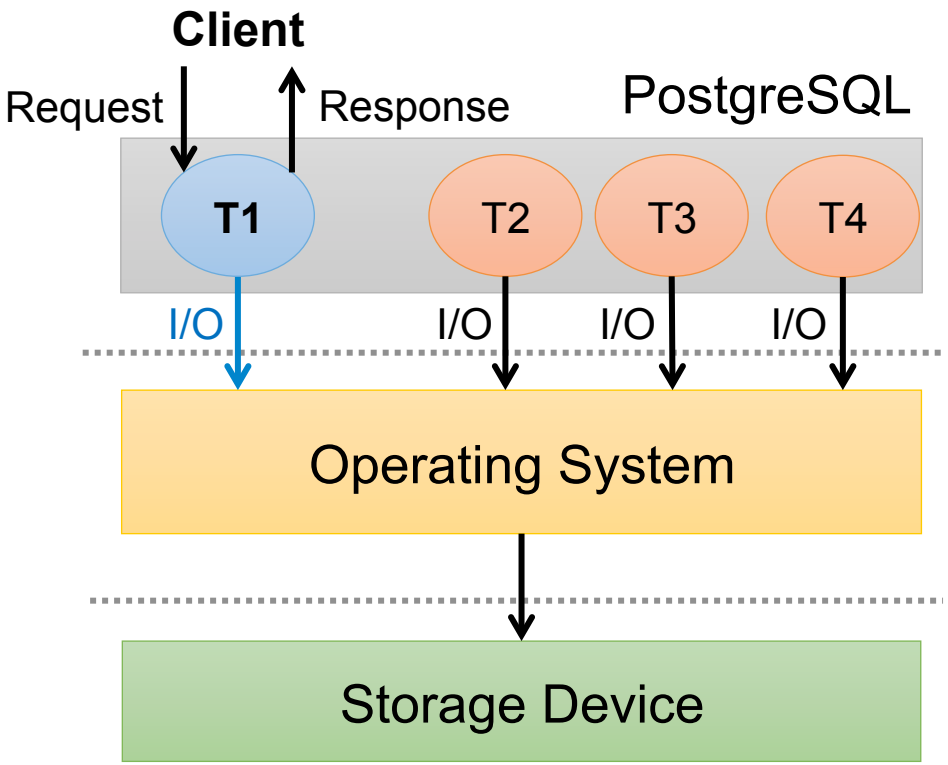
PostgreSQL Architecture



*** PostgreSQL's processes**

- **Backend (foreground)**
- **Checkpointer**
- **Autovacuum workers**
- **WAL writer**
- **Writer**
- **…**

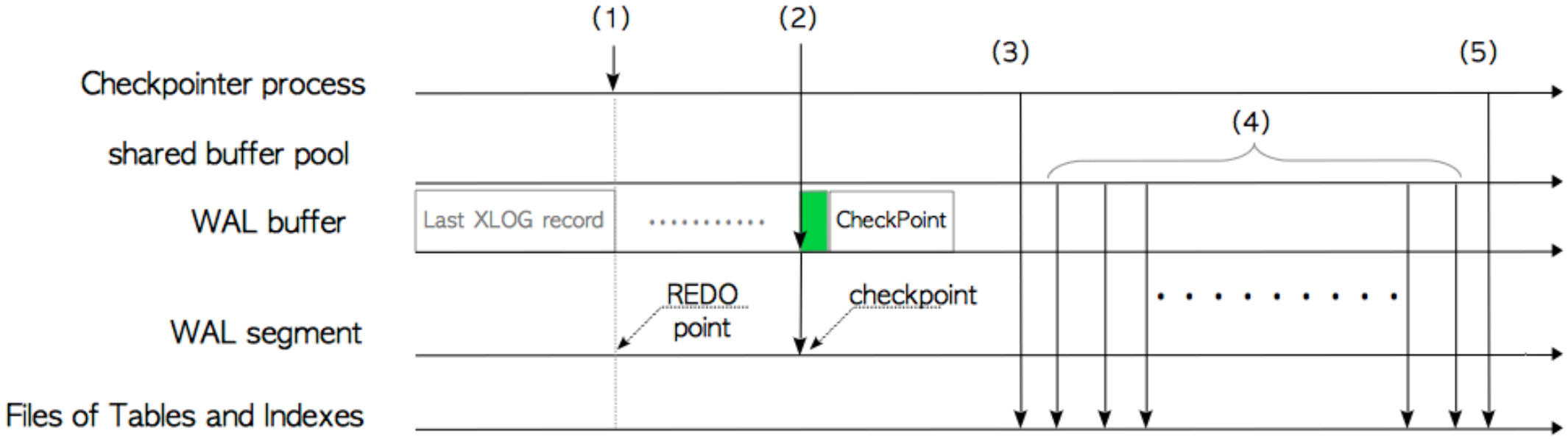
PostgreSQL Architecture



*** PostgreSQL's processes**

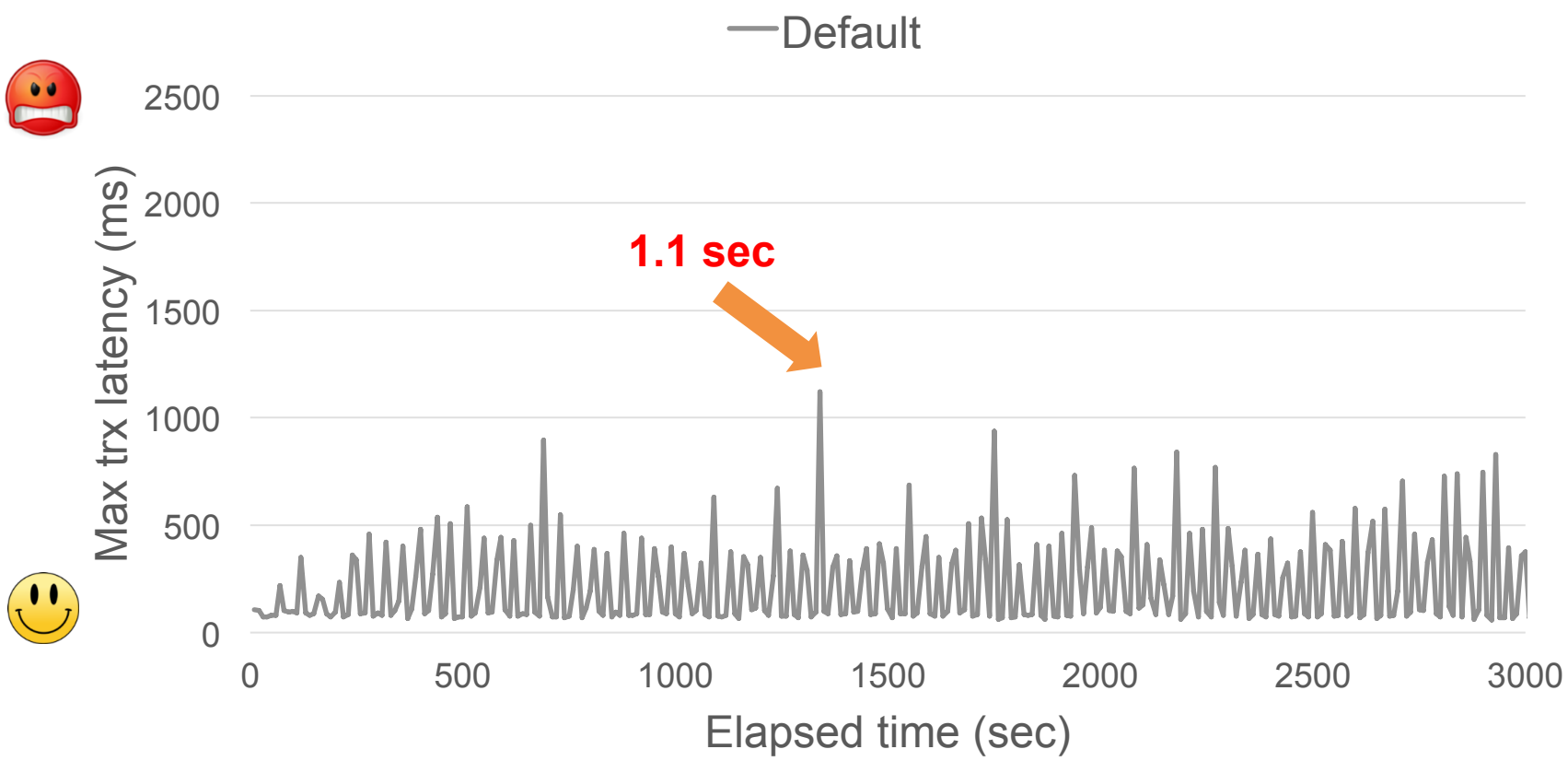
- **Backend (foreground)**
- **Checkpointer**
- **Autovacuum workers**
- **WAL writer**
- **Writer**
- **…**

PostgreSQL Checkpoint



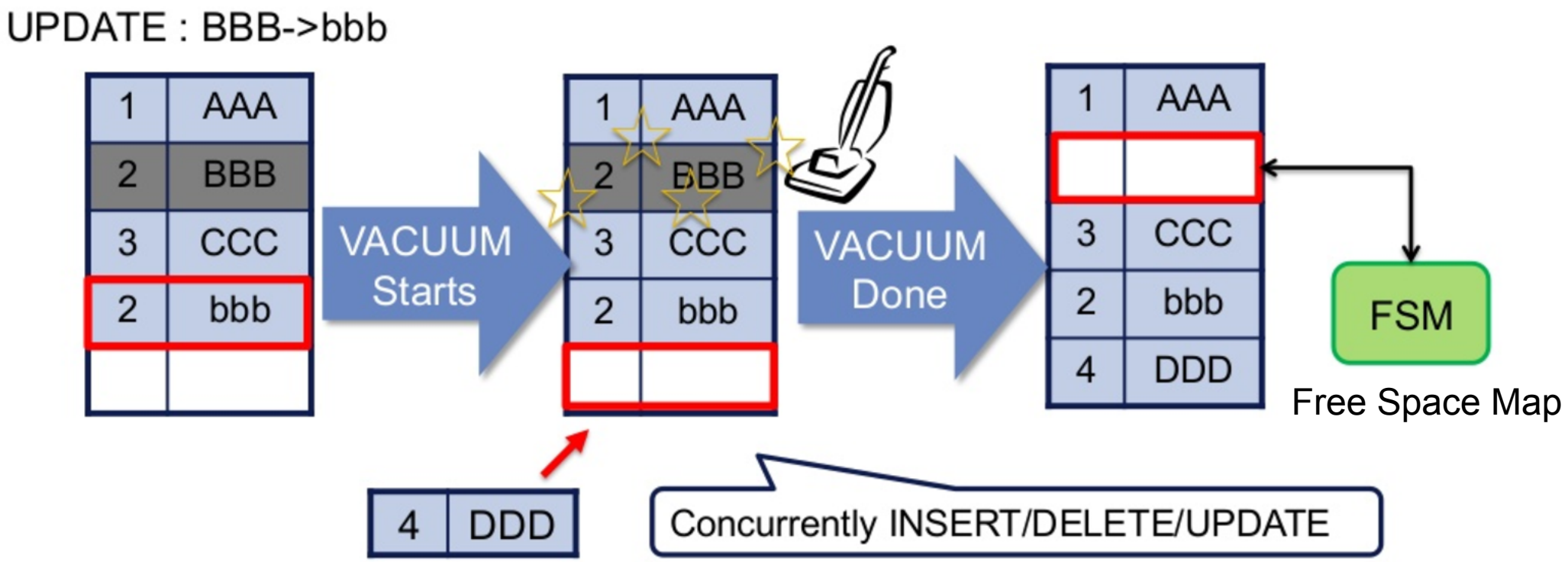
http://www.interdb.jp/pg/pgsql09.html

Impact of Checkpointer



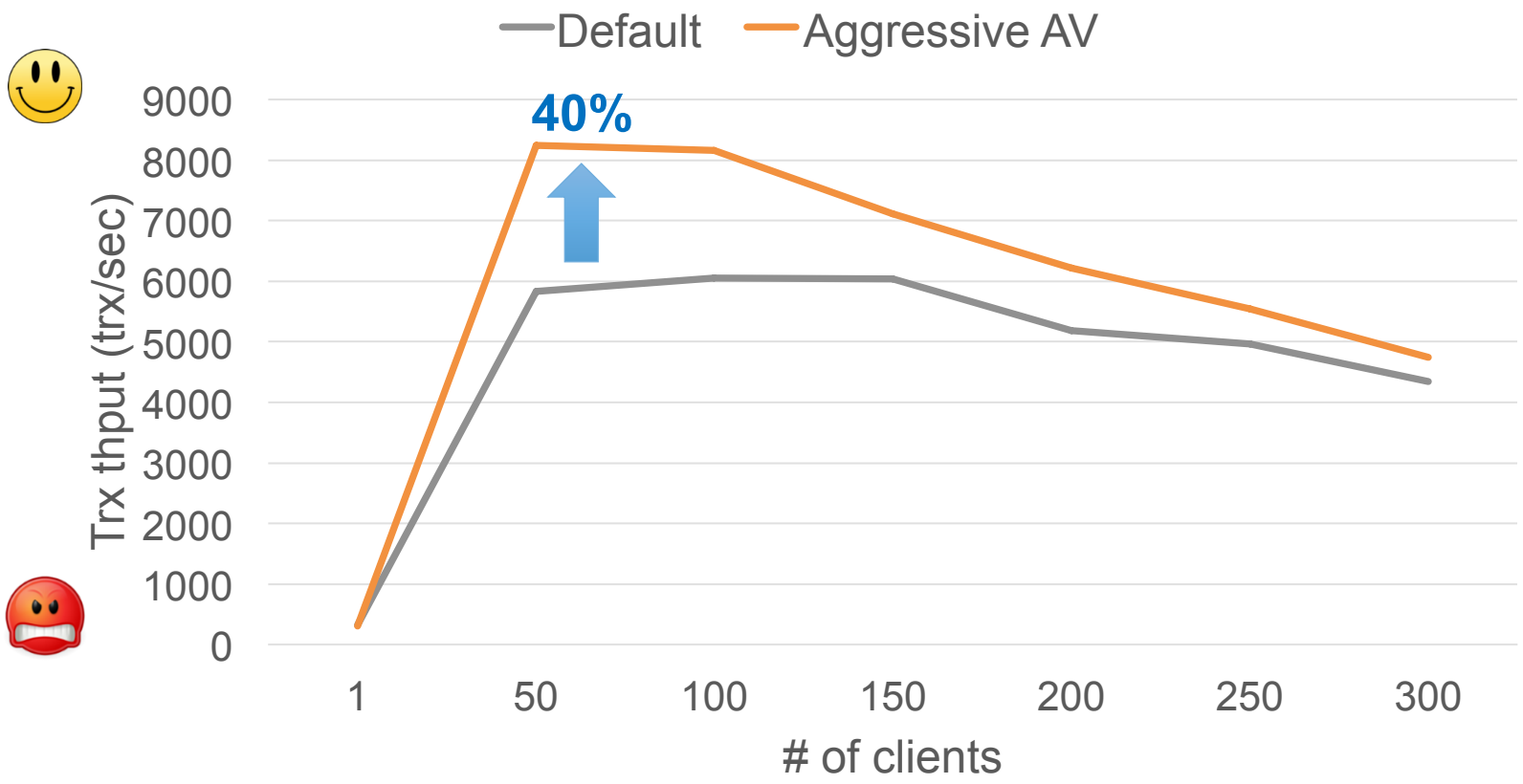
- Dell Poweredge R730
  - 32 cores
  - 132GB DRAM
  - 1 SAS SSD
- PostgreSQL v9.5.6
  - 52GB shared_buf
  - 10GB max_wal
- TPC-C workload
  - 50GB dataset
  - 1 hour run
  - 50 clients

PostgreSQL Autovacuum



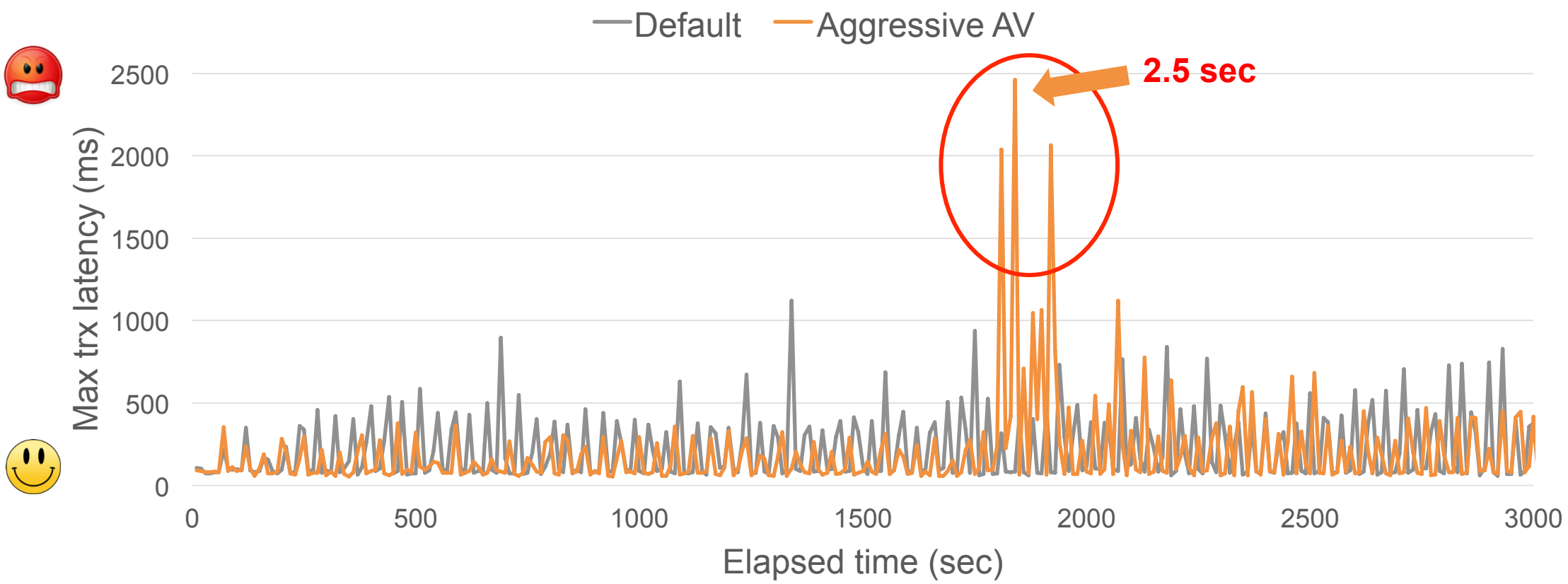
http://bstar36.tistory.com/308

Impact of Autovacuum Workers



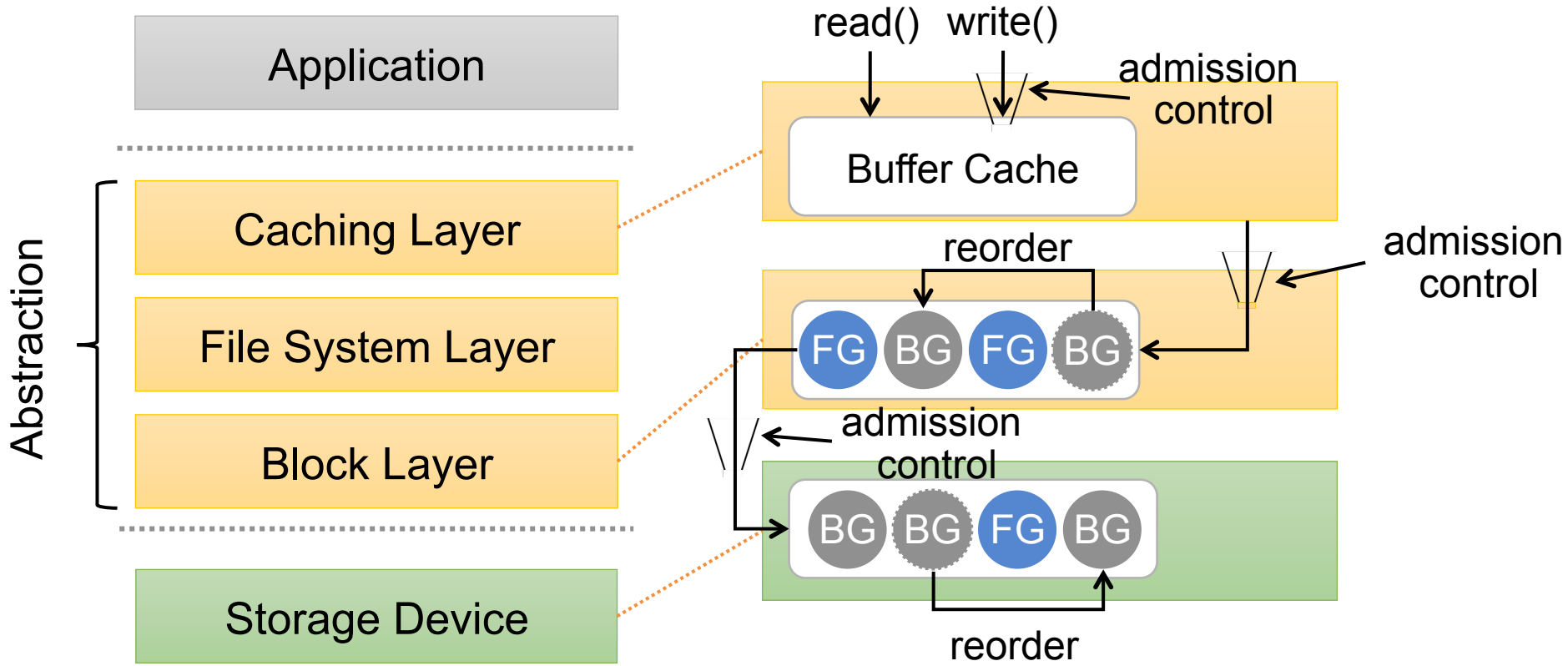
autovacuum_max_workers
3 => 6
autovacuum_naptime
1min => 15s
autovacuum_vacuum_threshold
50 => 25
autovacuum_analyze_threshold
50 => 10
autovacuum_vacuum_scale_factor
0.2 => 0.1
autovacuum_analyze_scale_factor
0.1 => 0.05
autovacuum_vacuum_cost_delay
20ms => -1
autovacuum_vacuum_cost_limit
-1 => 1000

Impact of Autovacuum Workers



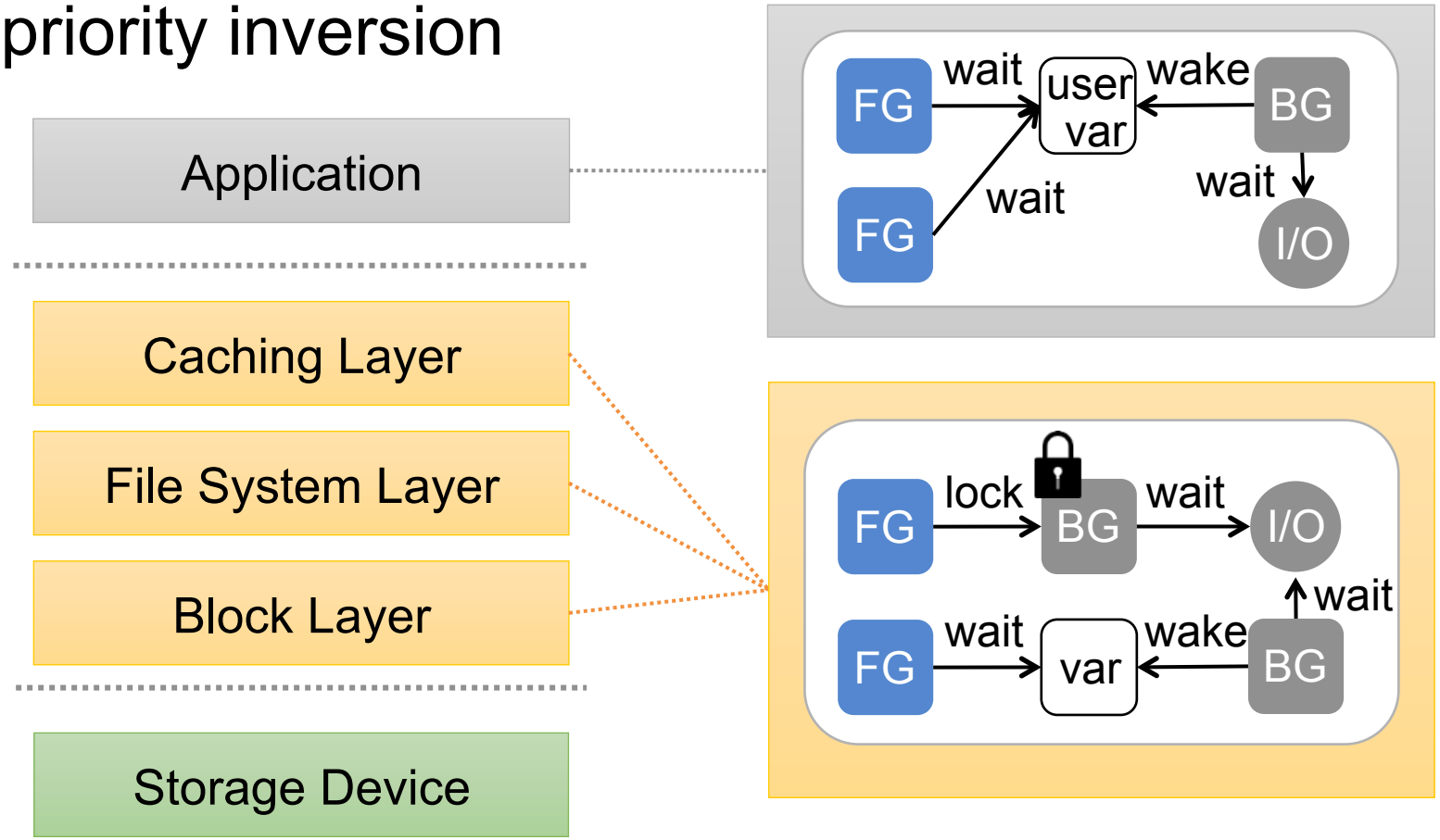
Why Background Tasks Matter

- Multiple independent layers



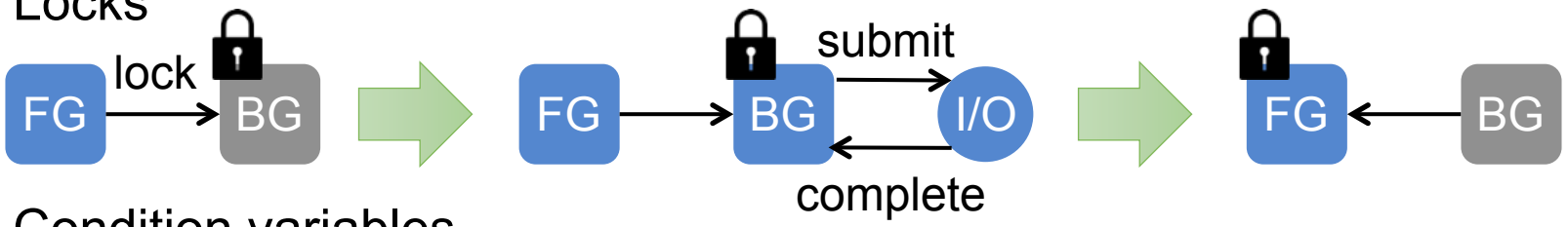
Why Background Tasks Matter

- I/O priority inversion

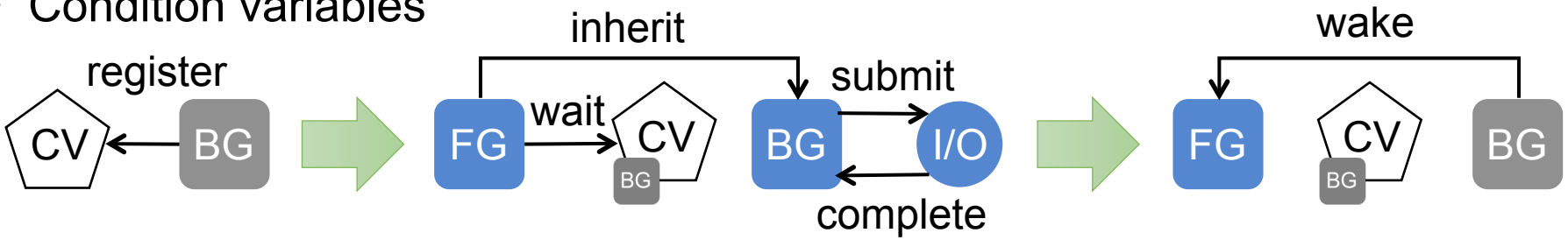


Request-Centric I/O Prioritization

- Solution v1 (reactive)
  - Request-aware I/O processing in the I/O path [FAST'17]
  - I/O priority inheritance [USENIX ATC'15, FAST'17]
    - Locks



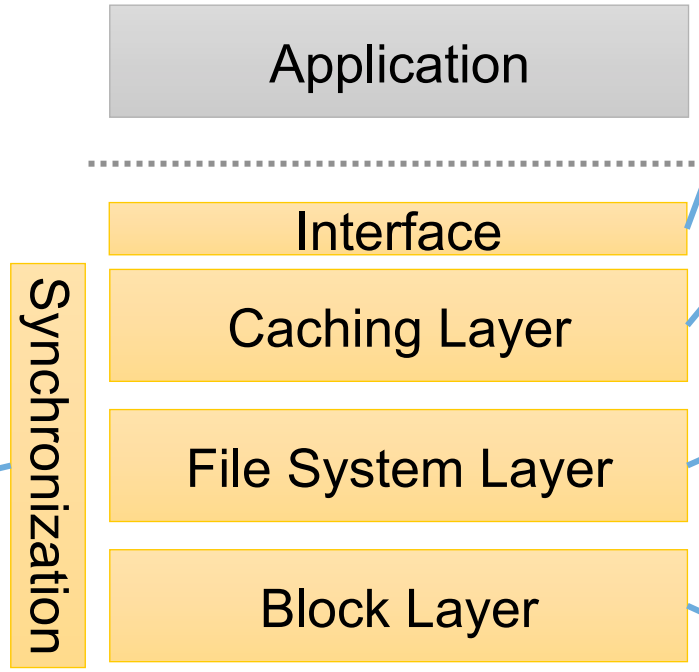
    - Condition variables



[USENIX ATC'15] Request-Oriented Durable Write Caching for Application Performance
[FAST'17] Enlightening the I/O Path: A Holistic Approach for Application Performance

Request-Centric I/O Prioritization

- Solution v1 (problem)



linux/kernel/fork.c
linux/kernel/sys.c
linux/kernel/sysctl.c
linux/include/linux/sched.h

linux/include/linux/writeback.h
linux/mm/page-writeback.c
linux/include/linux/mm_types.h
linux/fs/buffer.c

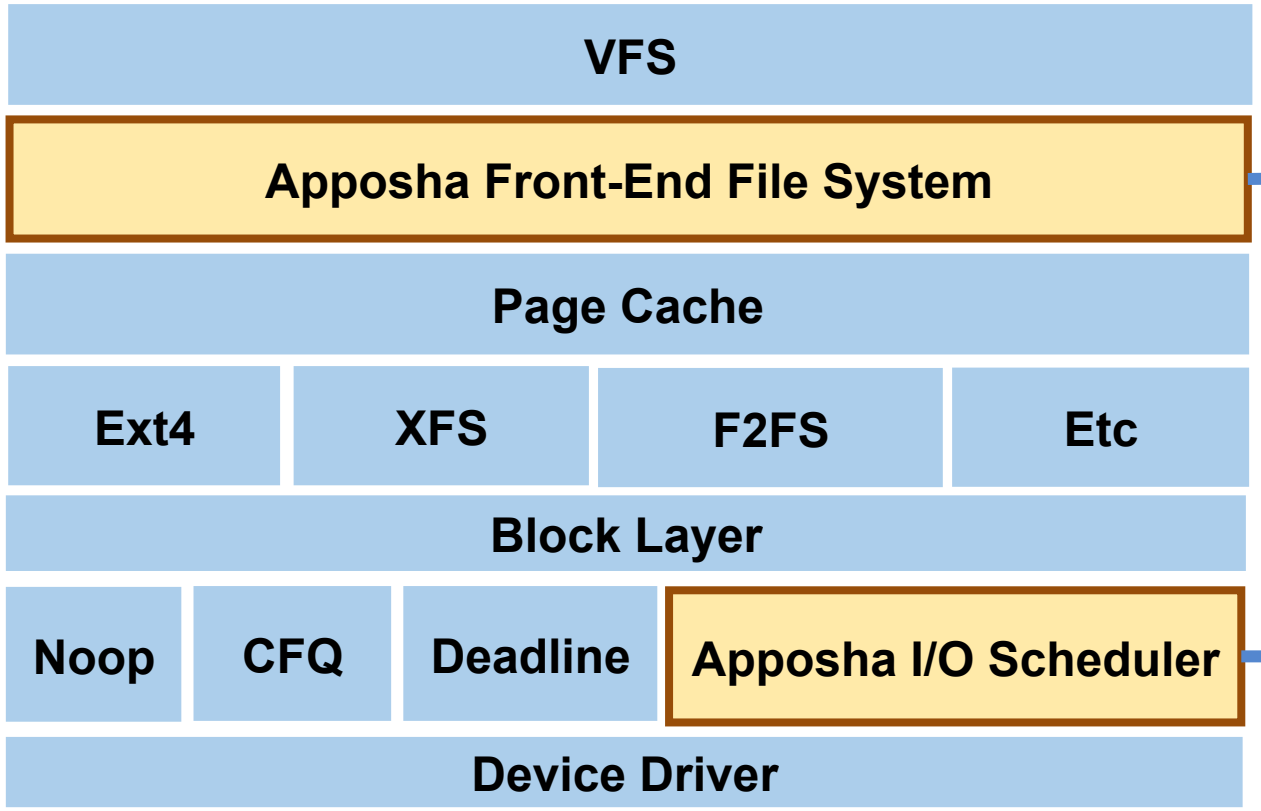
linux/fs/buffer.c
linux/fs/ext4/extents.c
linux/fs/ext4/inode.c
linux/include/linux/jbd2.h
linux/fs/jbd2/commit.c
linux/fs/jbd2/ journal.c
linux/fs/jbd2/ transaction.c

linux/include/linux/blk_types.h
linux/include/linux/blkdev.h
linux/include/linux/buffer_head.h
linux/include/linux/caq.h
linux/block/blk-core.c
linux/block/blk-flush.c
linux/block/blk-lib.c
linux/block/blk-mq.c
linux/block/caq-iosched.c
linux/block/cfq-iosched.c
linux/block/elevator.c

linux/include/linux/mutex.h
linux/include/linux/pagemap.h
linux/include/linux/rtmutex.h
linux/include/linux/rwsem.h
linux/include/uapi/linux/sem.h
linux/include/linux/wait.h
linux/kernel/sched/wait.c
linux/kernel/locking/rwsem.c
linux/kernel/futex.c
linux/kernel/locking/mutex.h
linux/kernel/locking/mutex.c
linux/kernel/locking/rtmutex.c
linux/kernel/locking/rwsem-xadd.c

Request-Centric I/O Prioritization

- Solution v2 (proactive)



Linux I/O Stack

- Control writes at the upper layer
- Tag low-priority for background I/Os

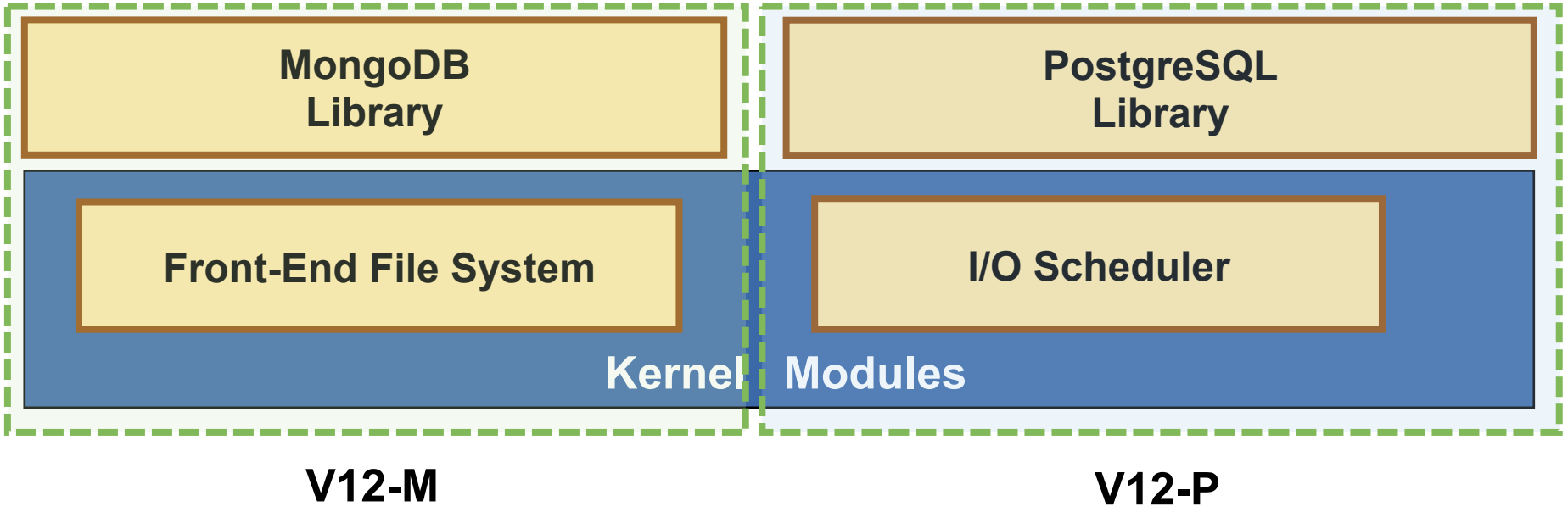
- Priority-aware I/O scheduling
- Control device-level congestion

Request-Centric I/O Prioritization

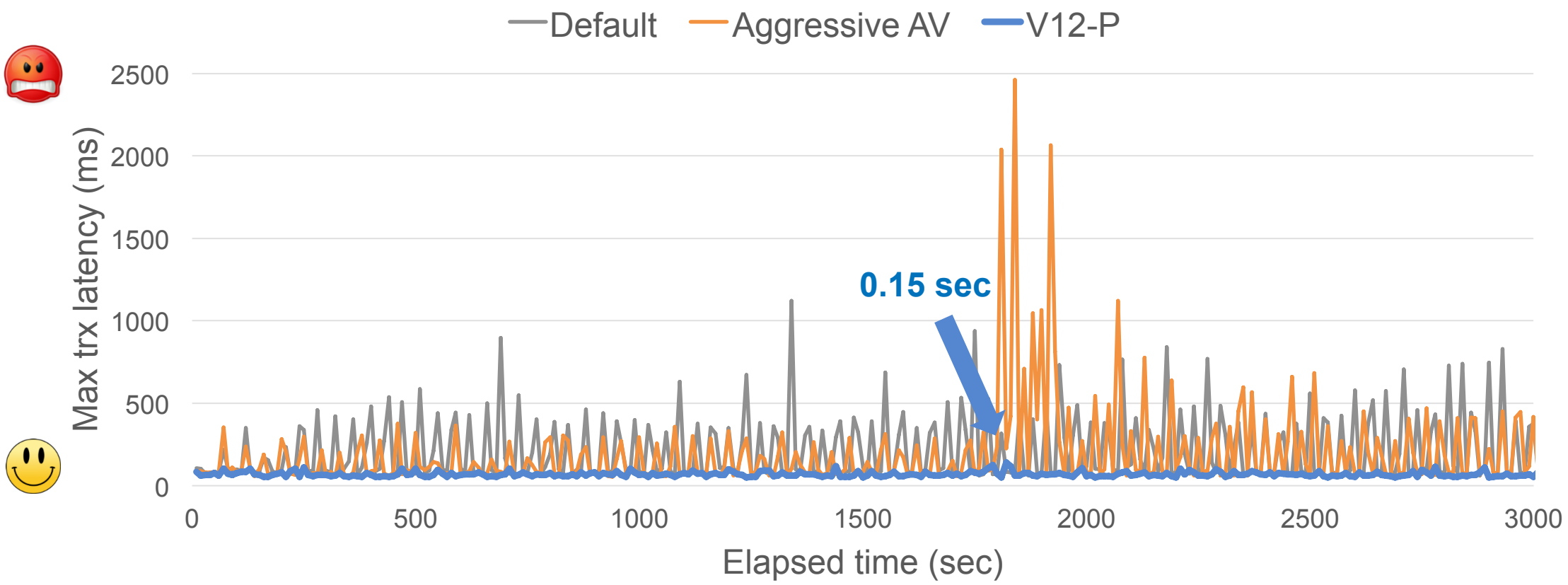
- V12 Engine

User-level library

- Classify task priority, optimize WAL accesses



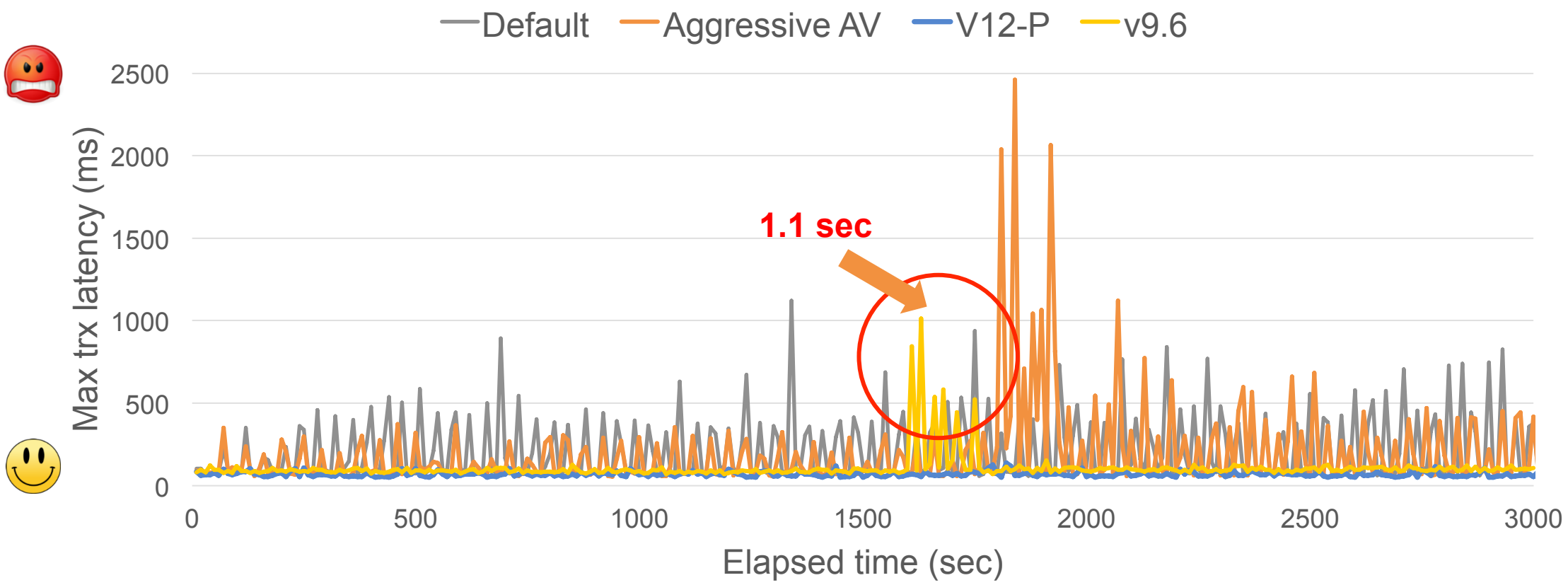
Request-Centric I/O Prioritization



V12-P: V12 Engine for PostgreSQL

Request-Centric I/O Prioritization

New feature since v9.6:
checkpoint_flush_after

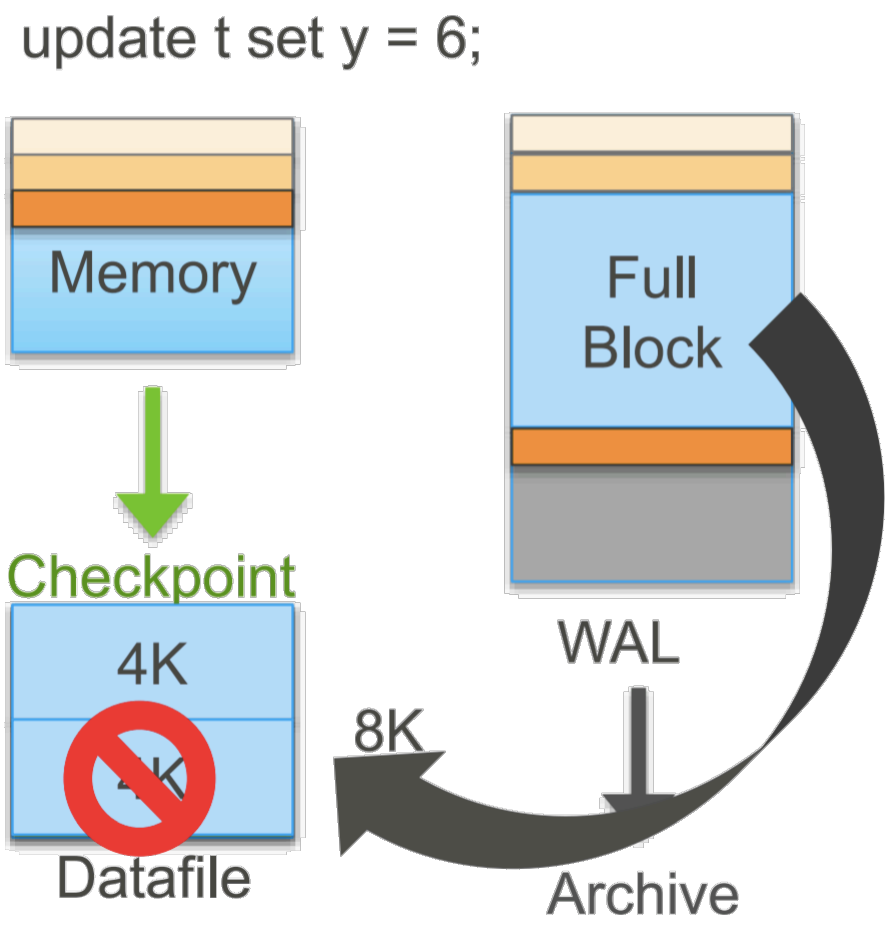


V12-P: V12 Engine for PostgreSQL

Contents

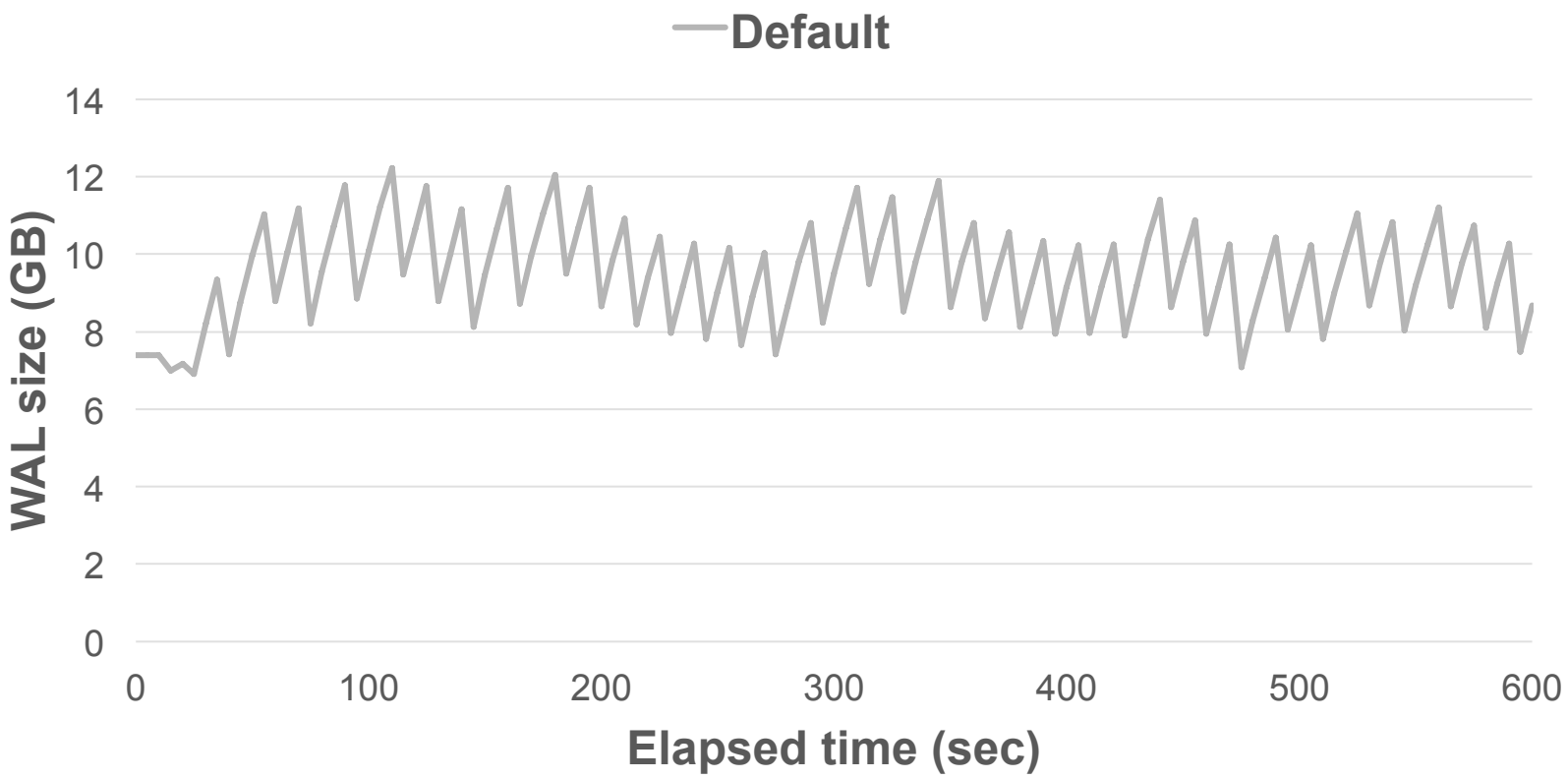
- Background tasks
- **Full page writes**
- Parallel query

Full Page Writes



Full Page Writes

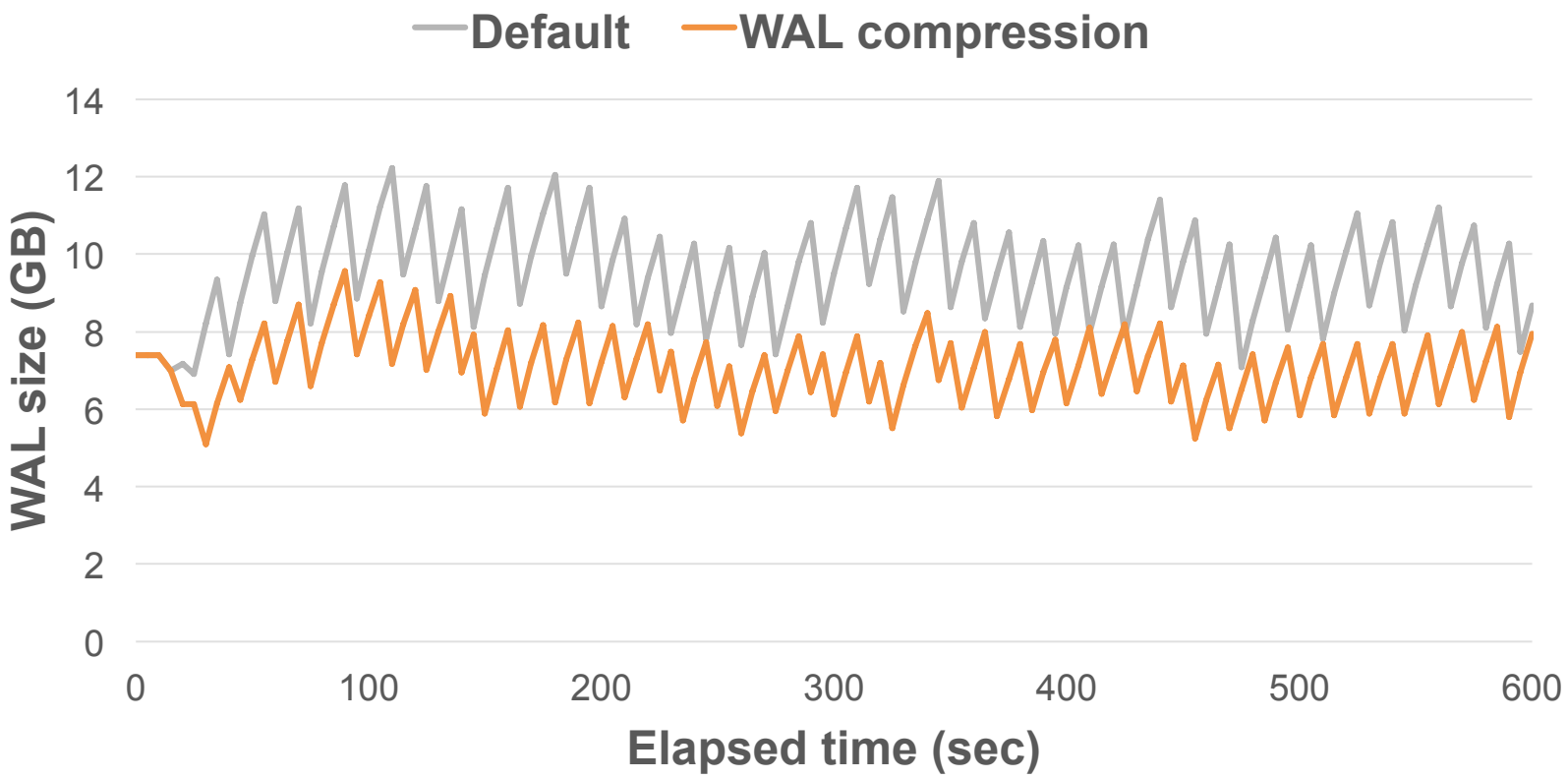
- Impact on WAL size



- PostgreSQL v9.6.5
  - 52GB shared_buffer
  - **1GB max_wal**
- TPC-C workload
  - 50GB dataset
  - 10 min run
  - 200 clients

WAL Compression

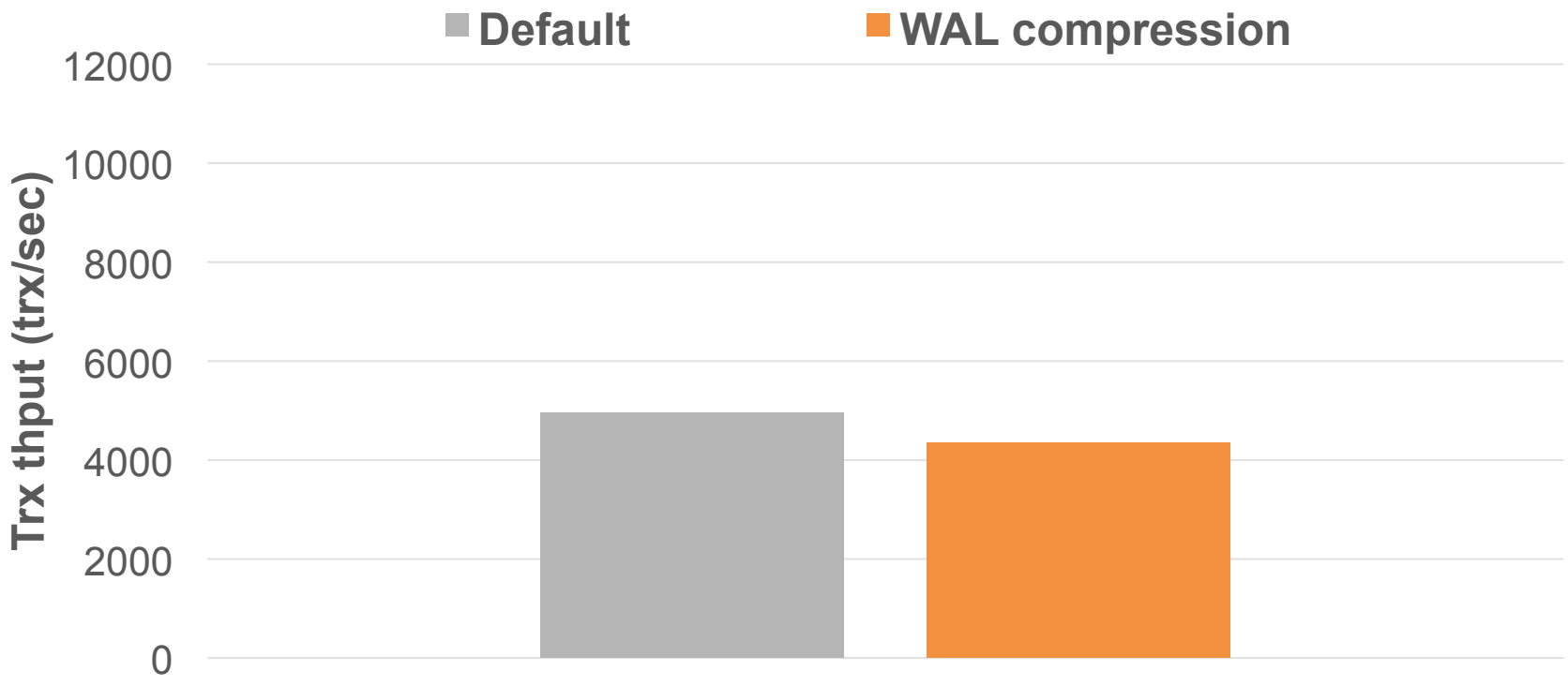
- Impact on WAL size



- PostgreSQL v9.6.5
  - 52GB shared_buffer
  - **1GB max_wal**
- TPC-C workload
  - 50GB dataset
  - 10 min run
  - 200 clients

WAL Compression

- Impact on performance

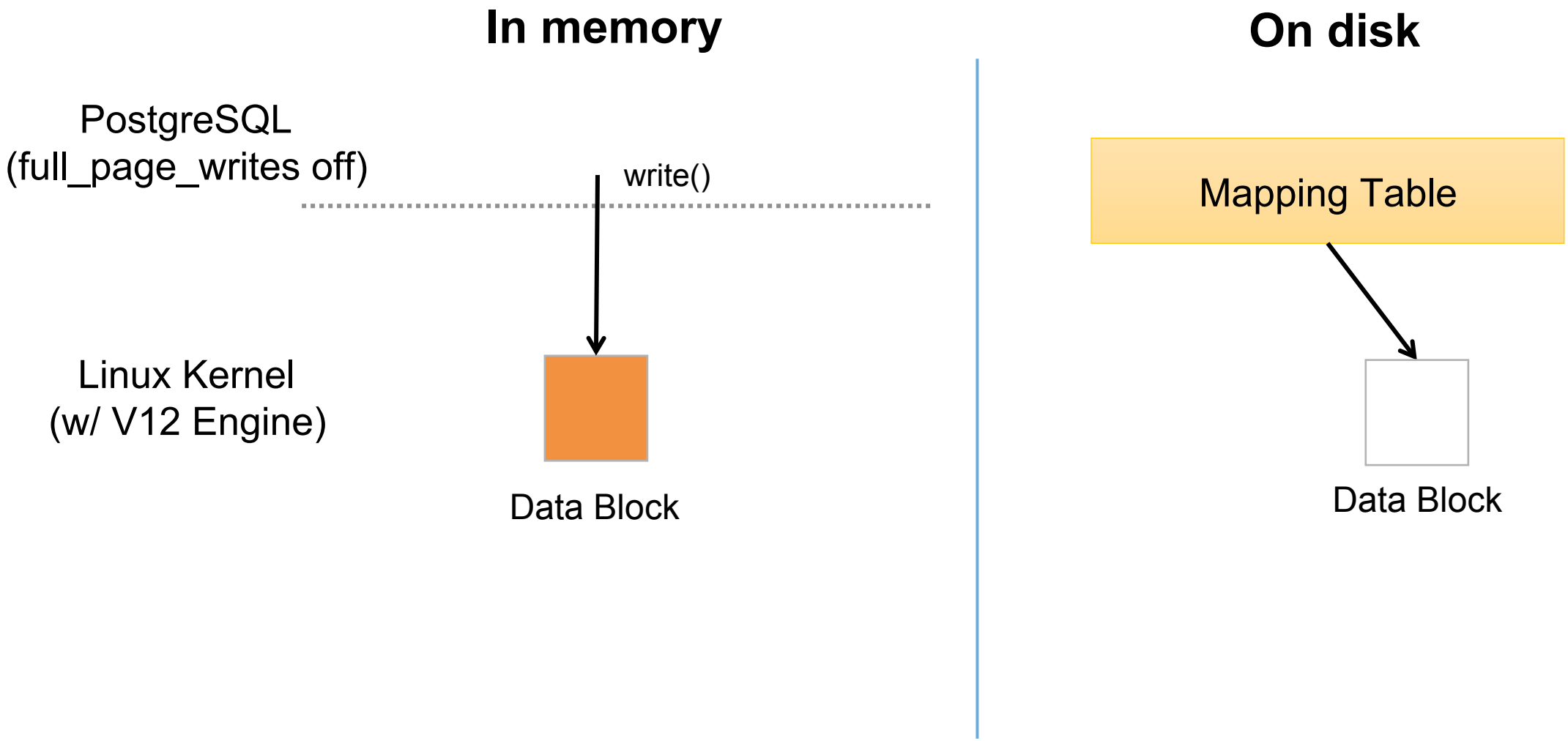


WAL Compression

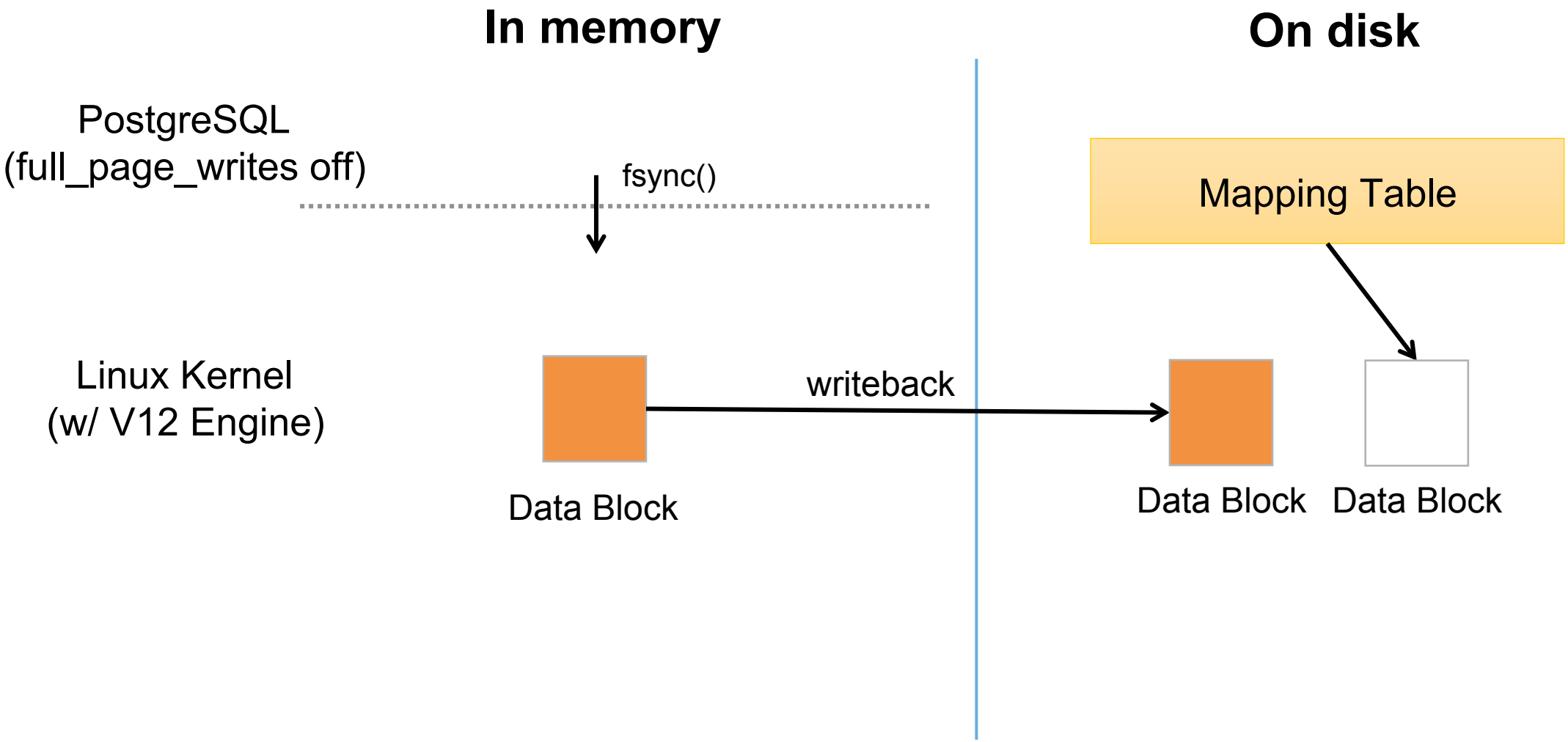
- Impact on performance

What if we can safely disable full_page_writes w/o special H/W support?

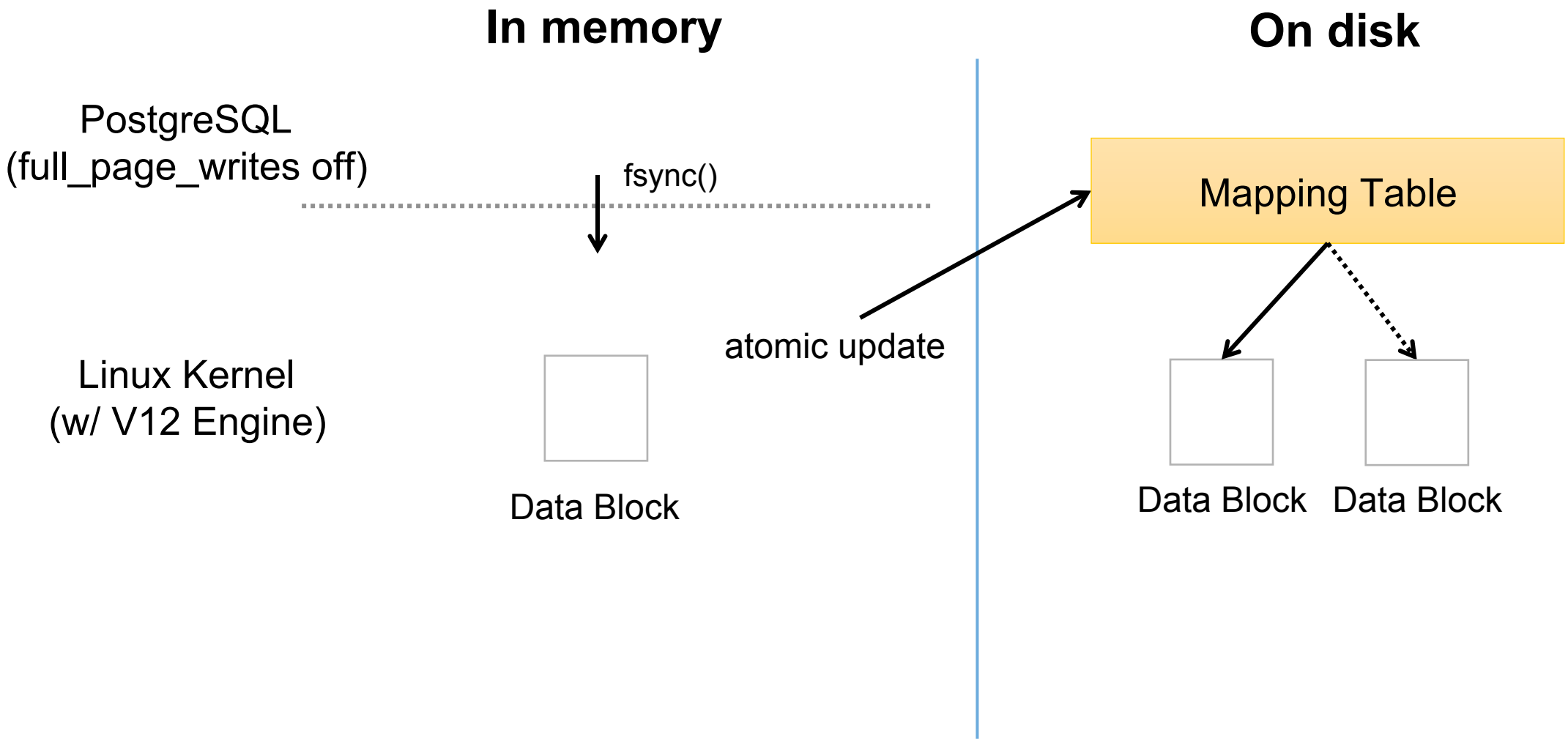
Atomic Write Support



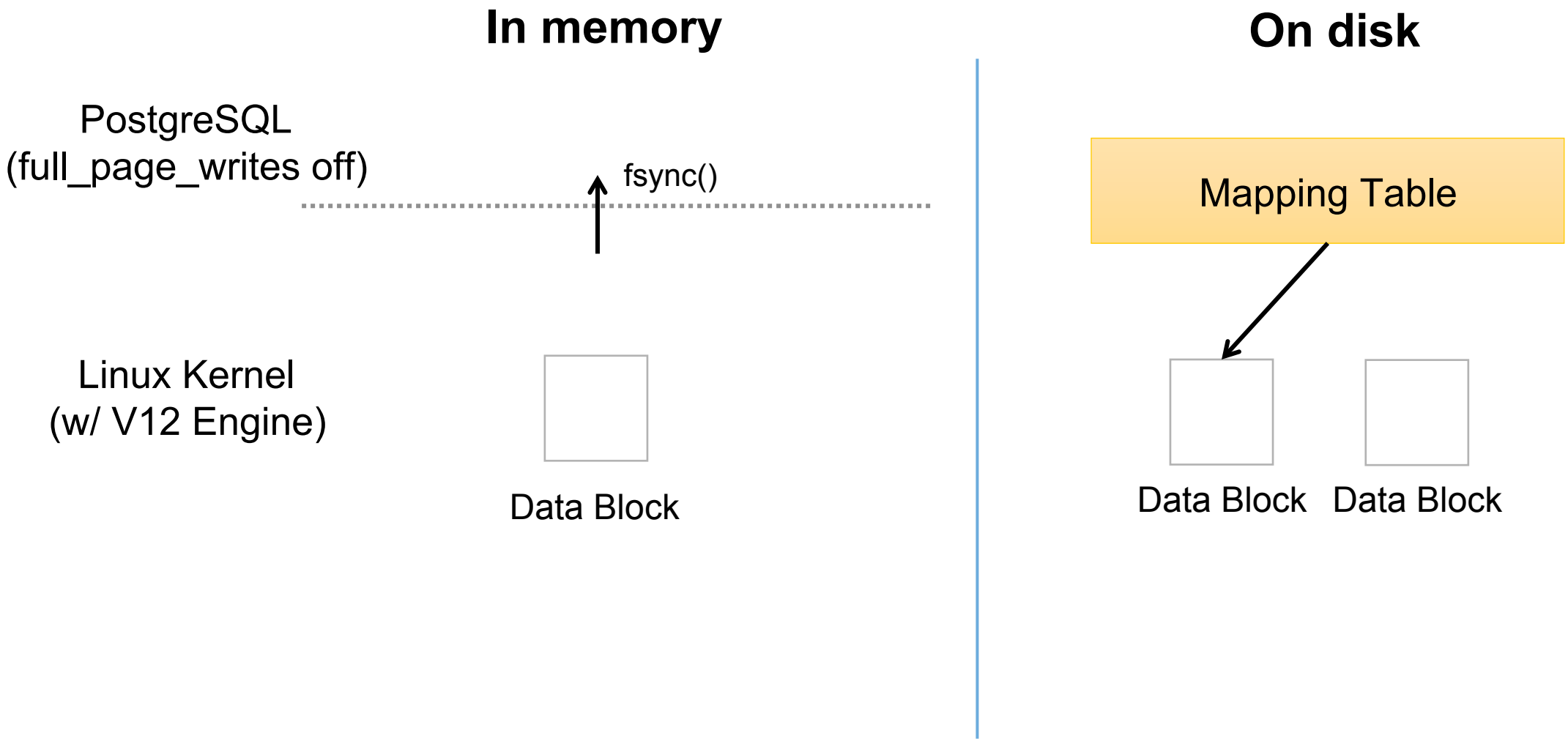
Atomic Write Support



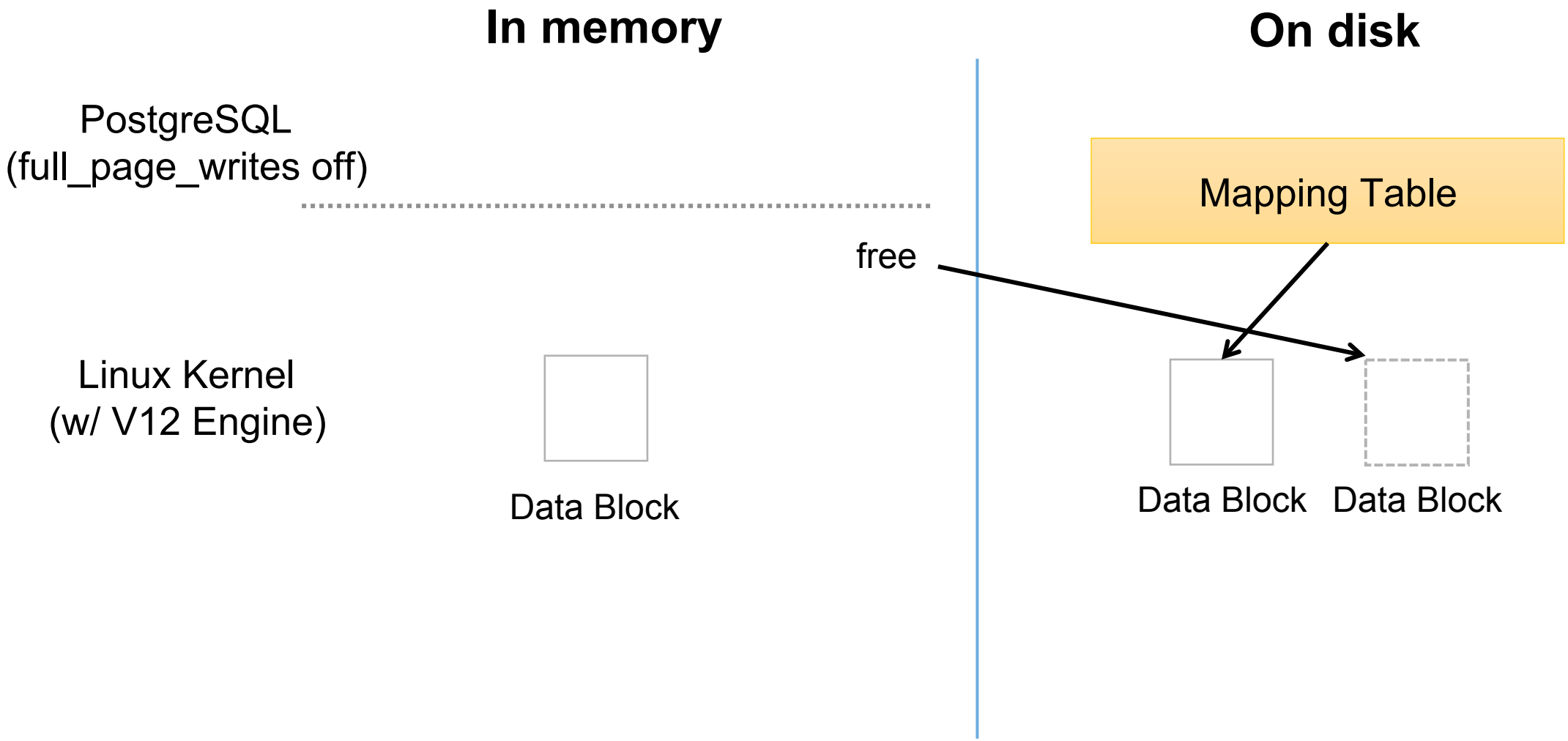
Atomic Write Support



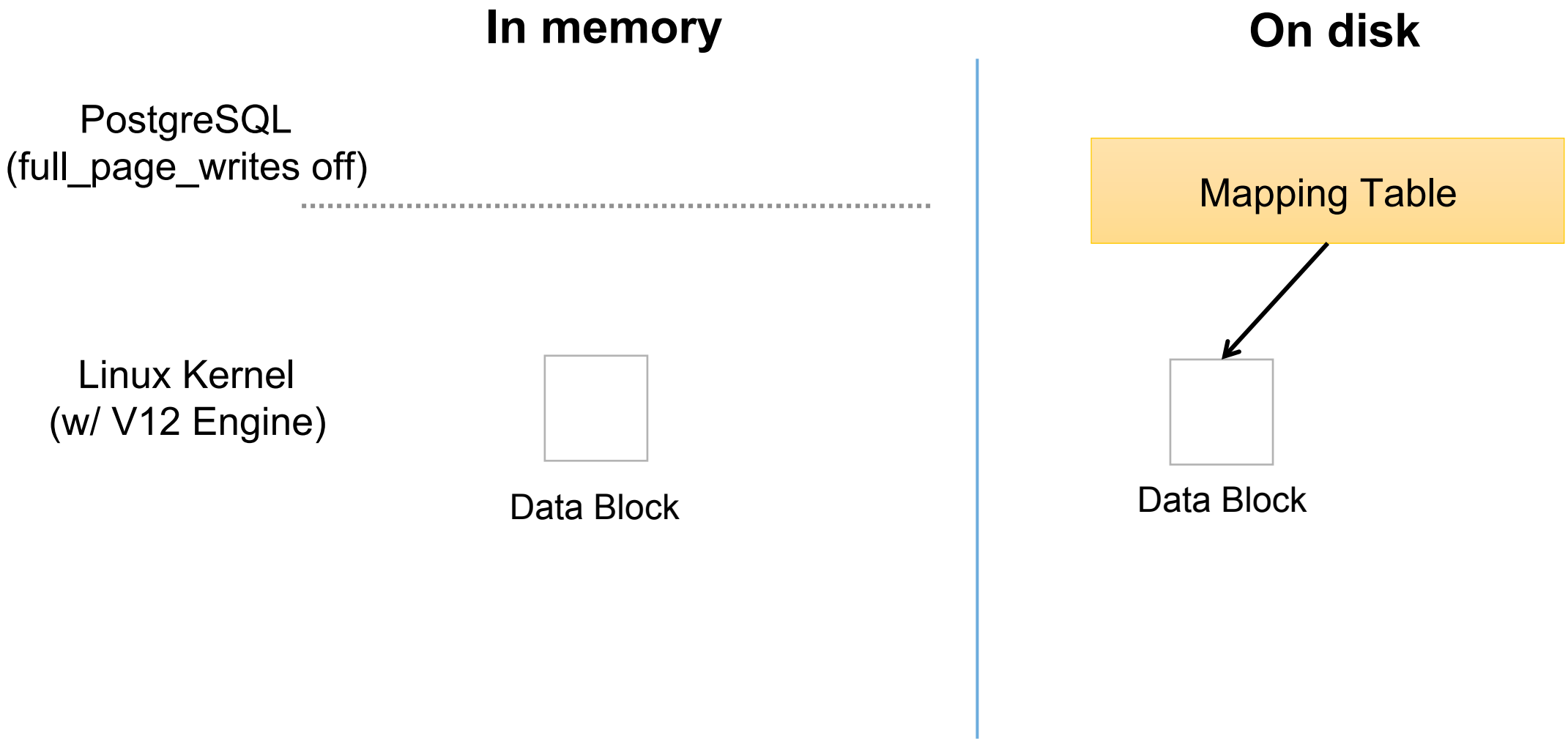
Atomic Write Support



Atomic Write Support

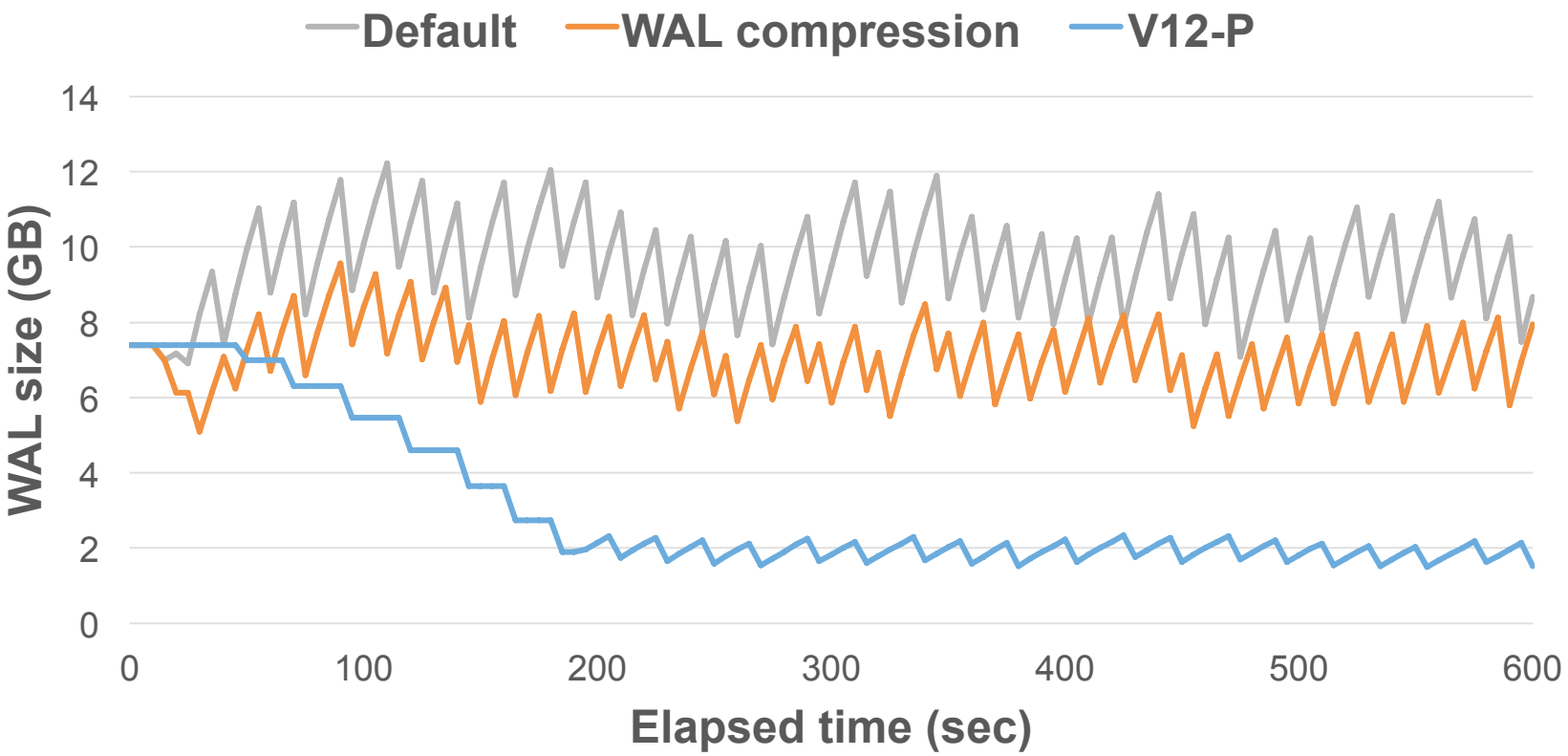


Atomic Write Support



Atomic Write Support

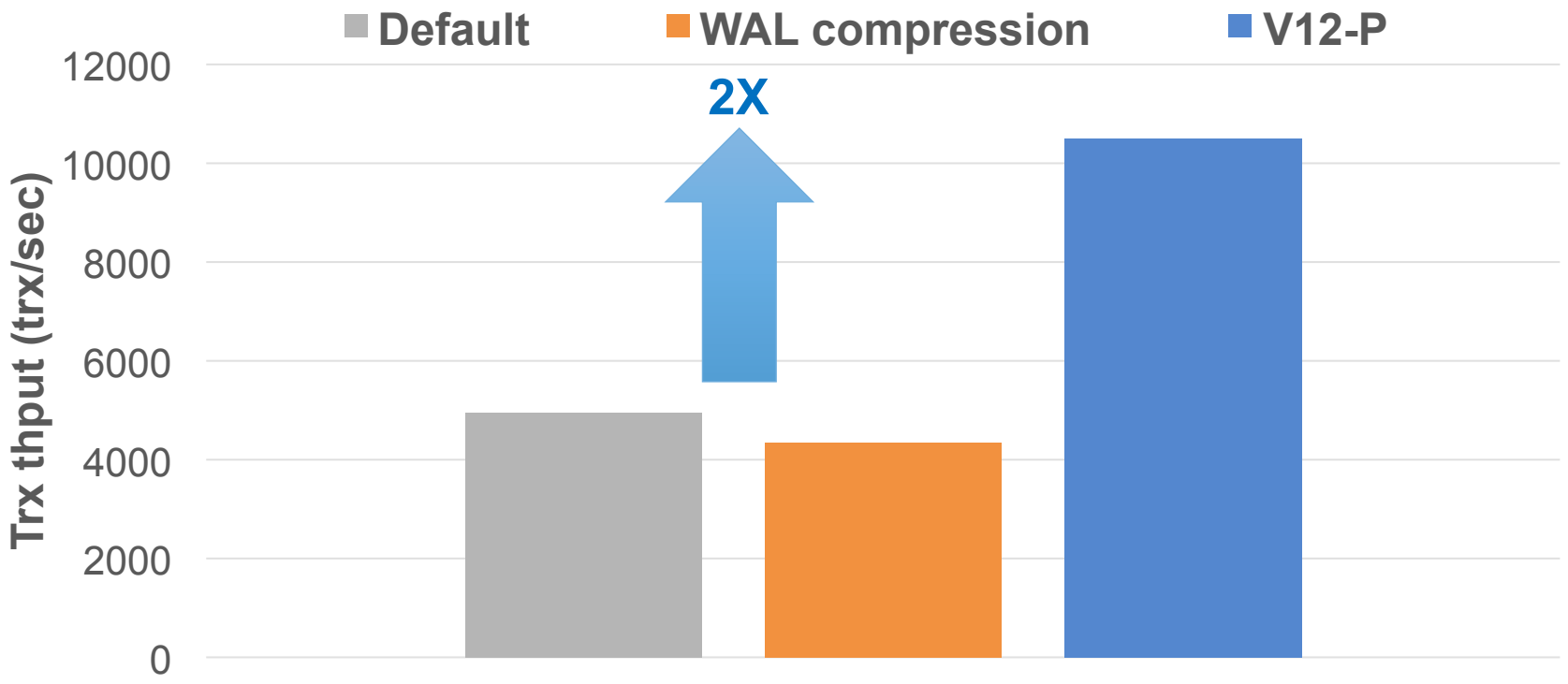
- Impact on WAL size



V12-P: V12 Engine for PostgreSQL

Atomic Write Support

- Impact on performance

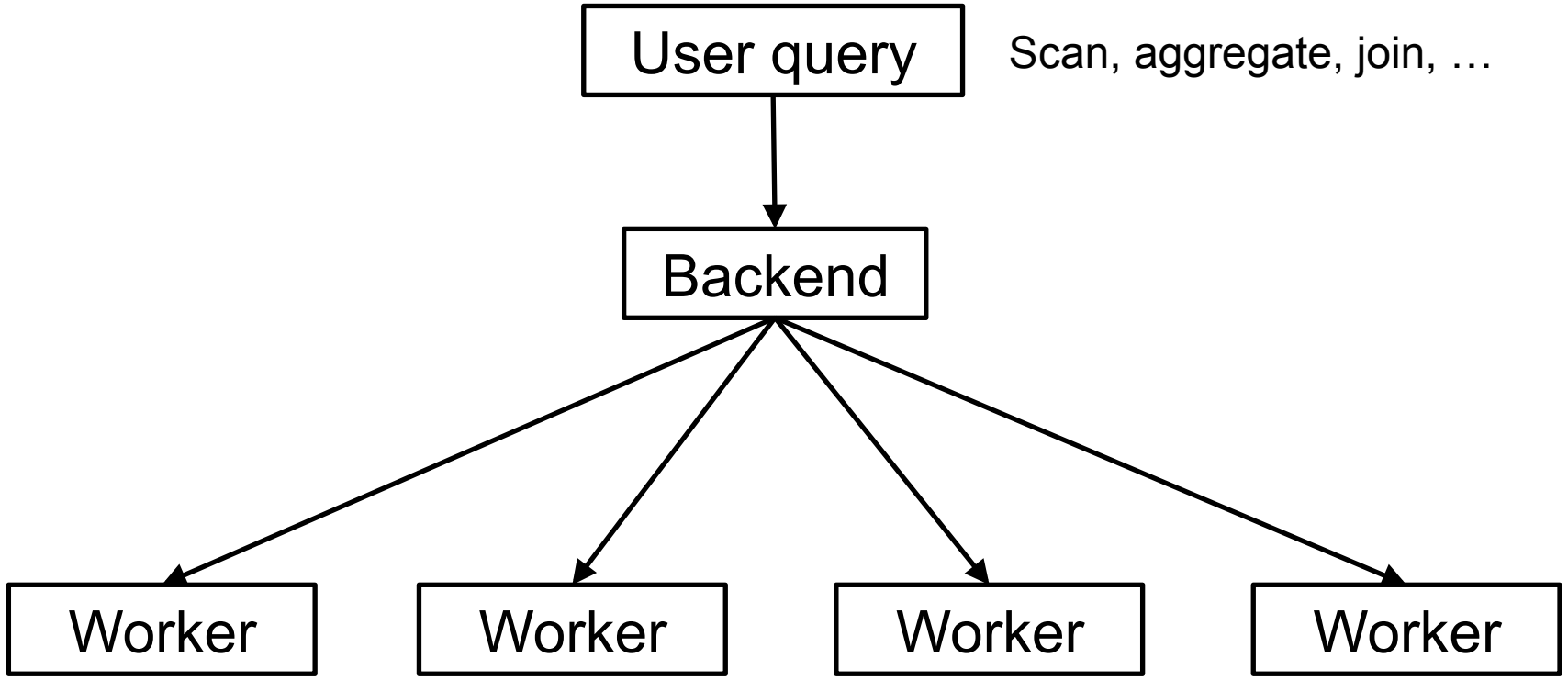


V12-P: V12 Engine for PostgreSQL

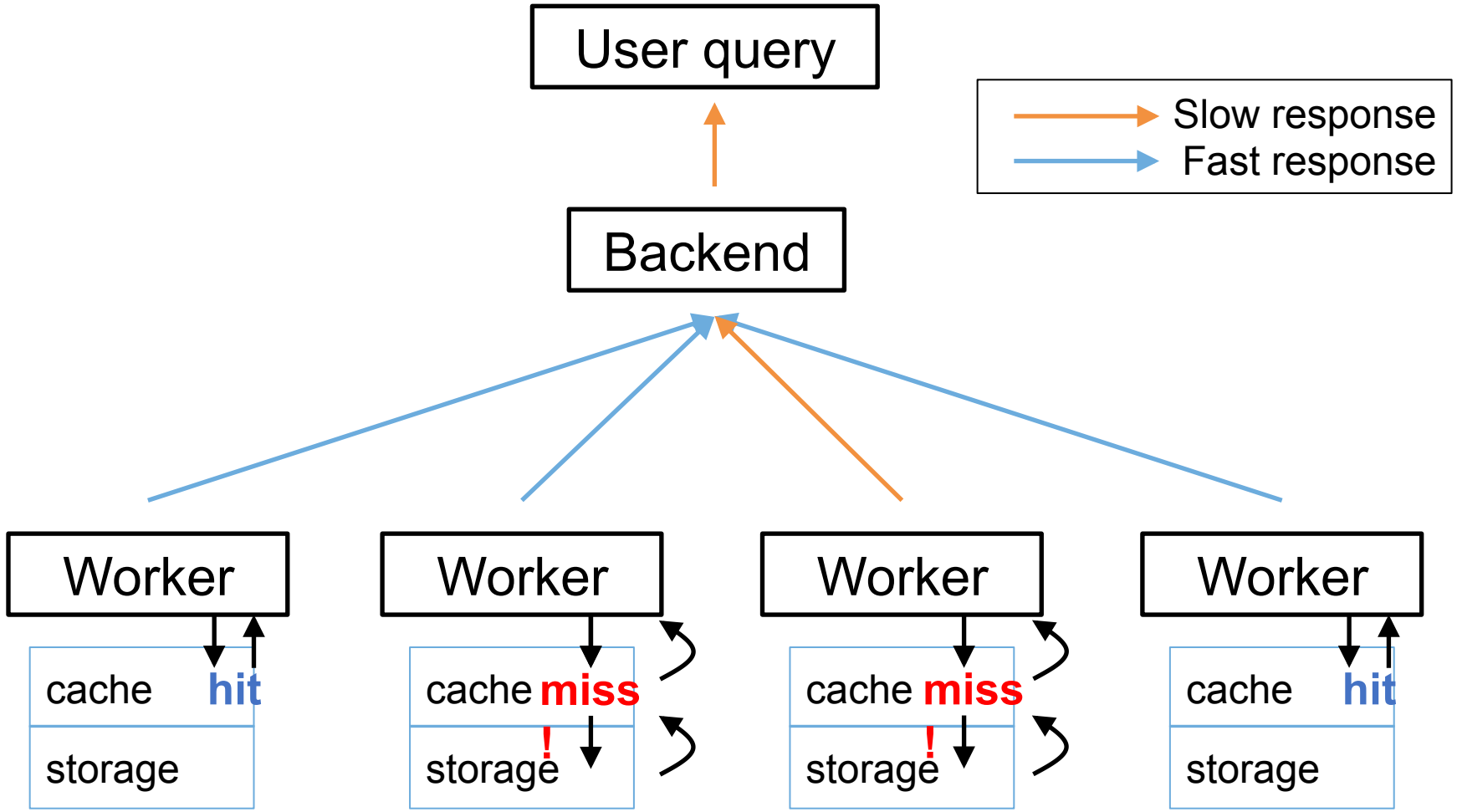
Contents

- Background tasks
- Full page writes
- **Parallel query**

Parallel Query

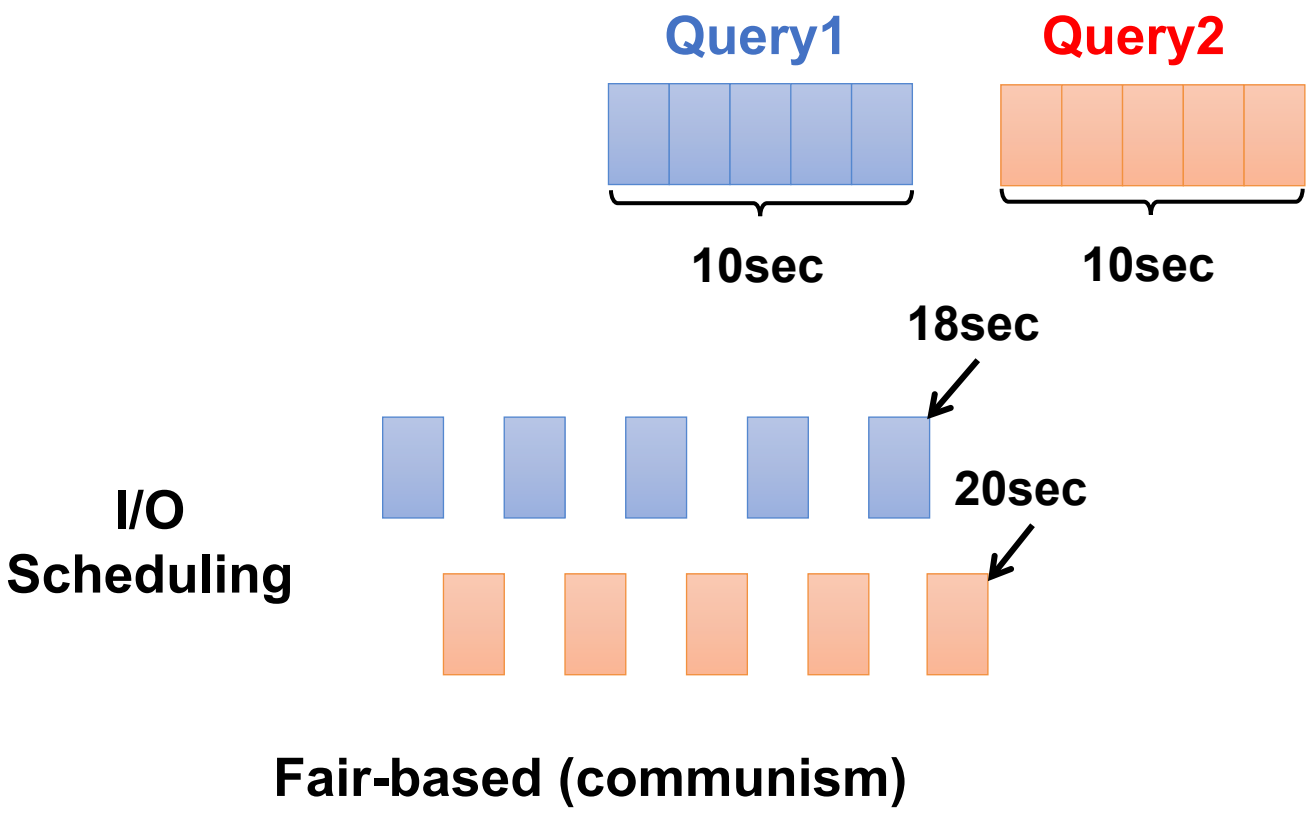


Parallel Query



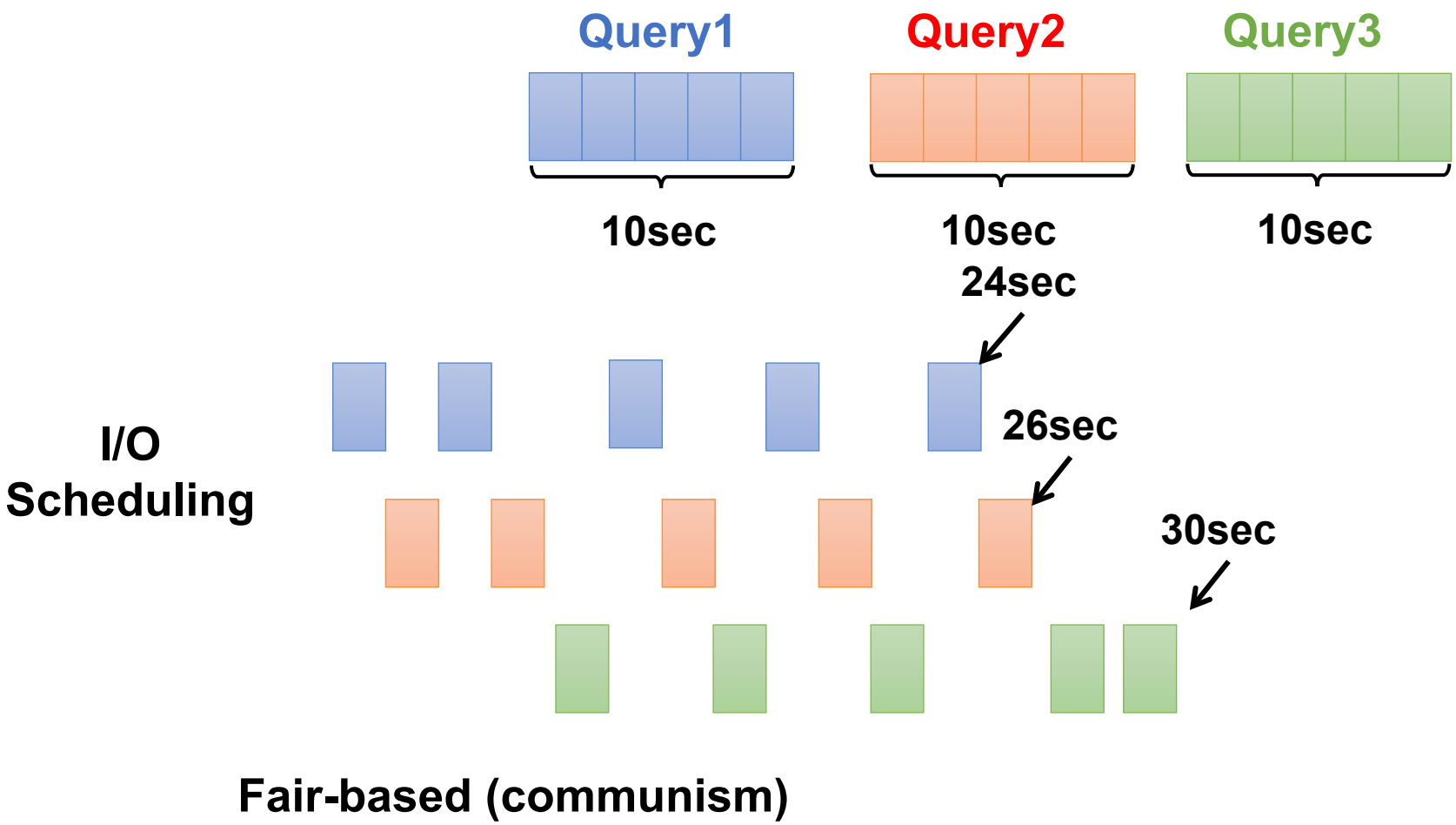
Parallel Query

- Problem inside Linux kernel



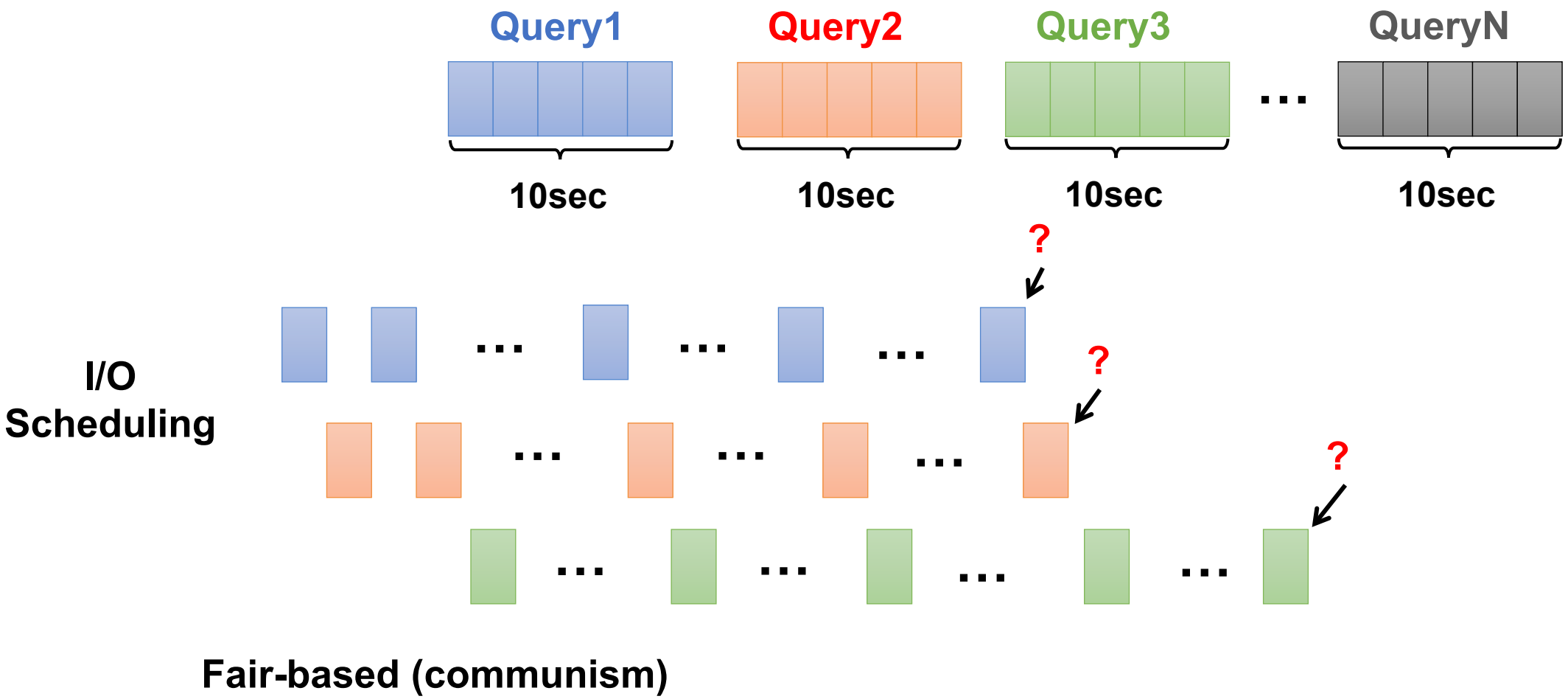
Parallel Query

- Problem inside Linux kernel



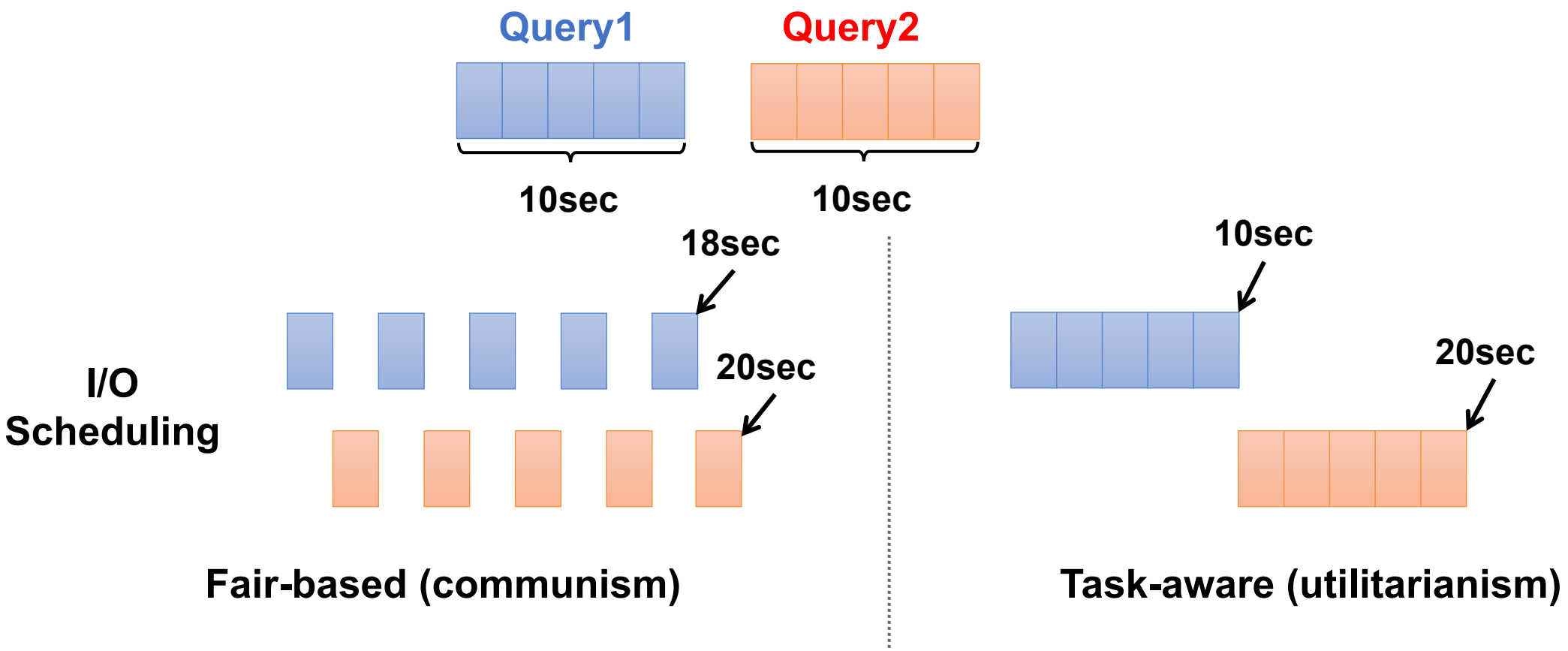
Parallel Query

- Problem inside Linux kernel



Parallel Query

- Problem inside Linux kernel



Optimizing Parallel Query

- Illustrative experiment

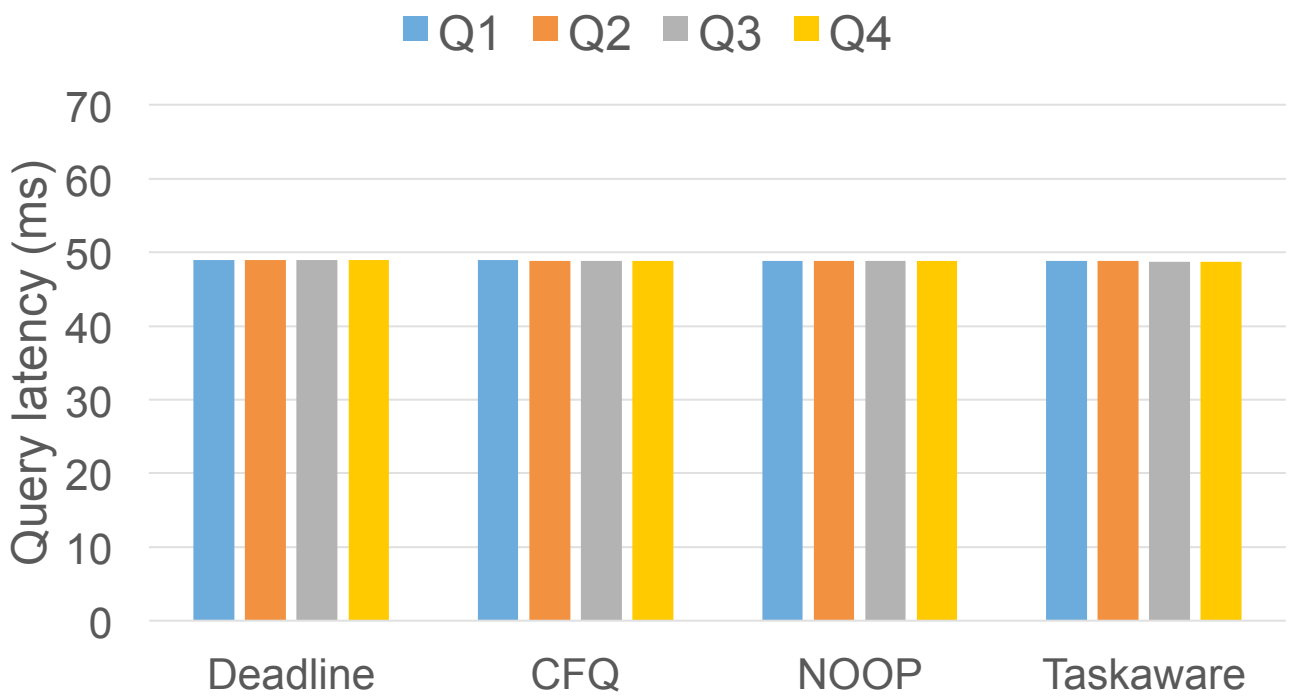
pgbench –i –s 1000 test1
pgbench –i –s 1000 test2

Q1: SELECT * FROM pgbench_accounts WHERE filler LIKE ‘%x% -d test1
Q2: SELECT * FROM pgbench_accounts WHERE filler LIKE ‘%x% -d test2

```
 QUERY PLAN
----------------------------------------------------------------------------------
 Gather  (cost=1000.00..1818916.53 rows=1 width=97)
   Workers Planned: 7
   ->  Parallel Seq Scan on pgbench_accounts  (cost=0.00..1817916.43 rows=1 width=97)
         Filter: (filler ~~ '%x%'::text)
```

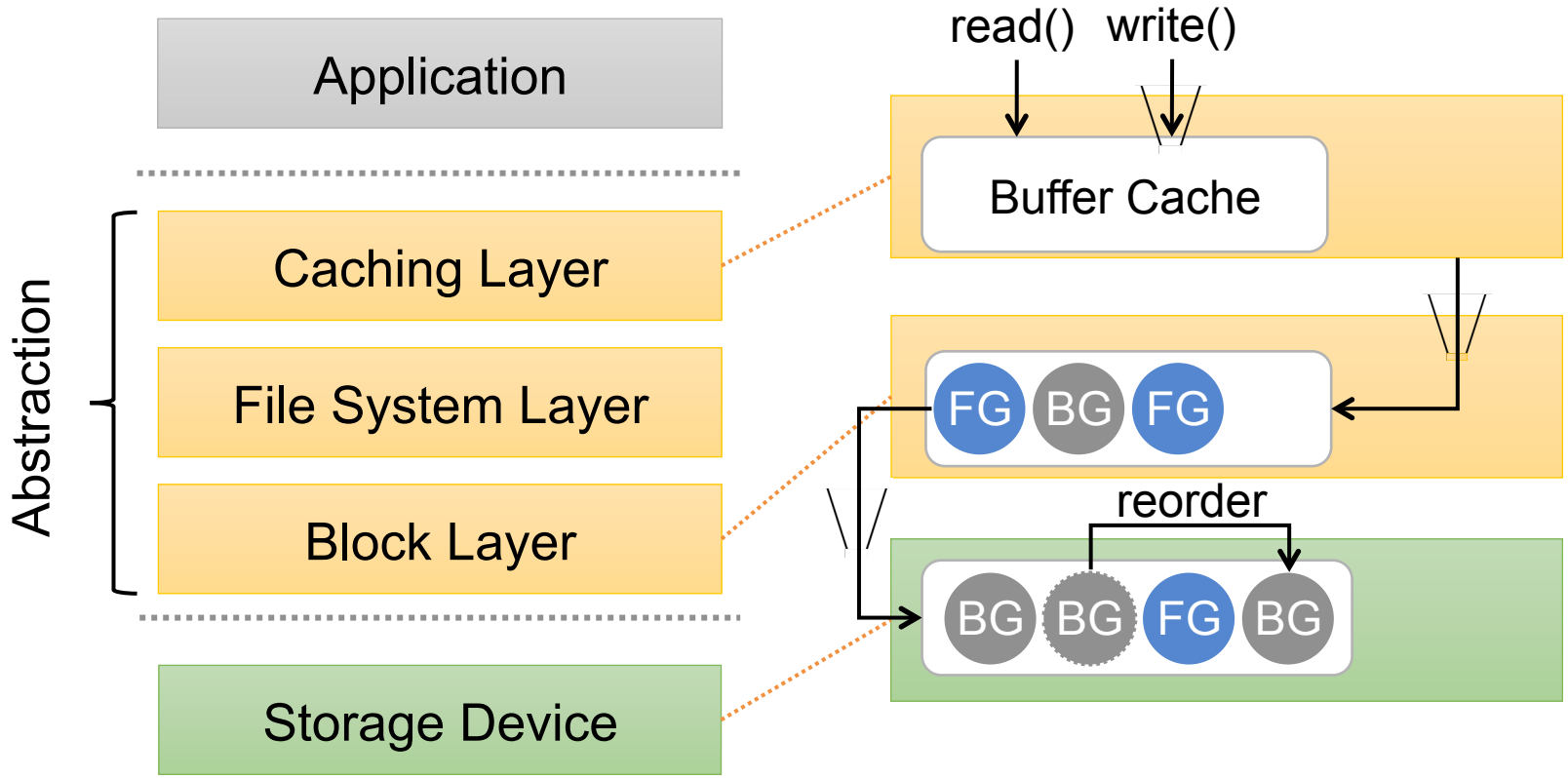
Optimizing Parallel Query

- Illustrative experiment



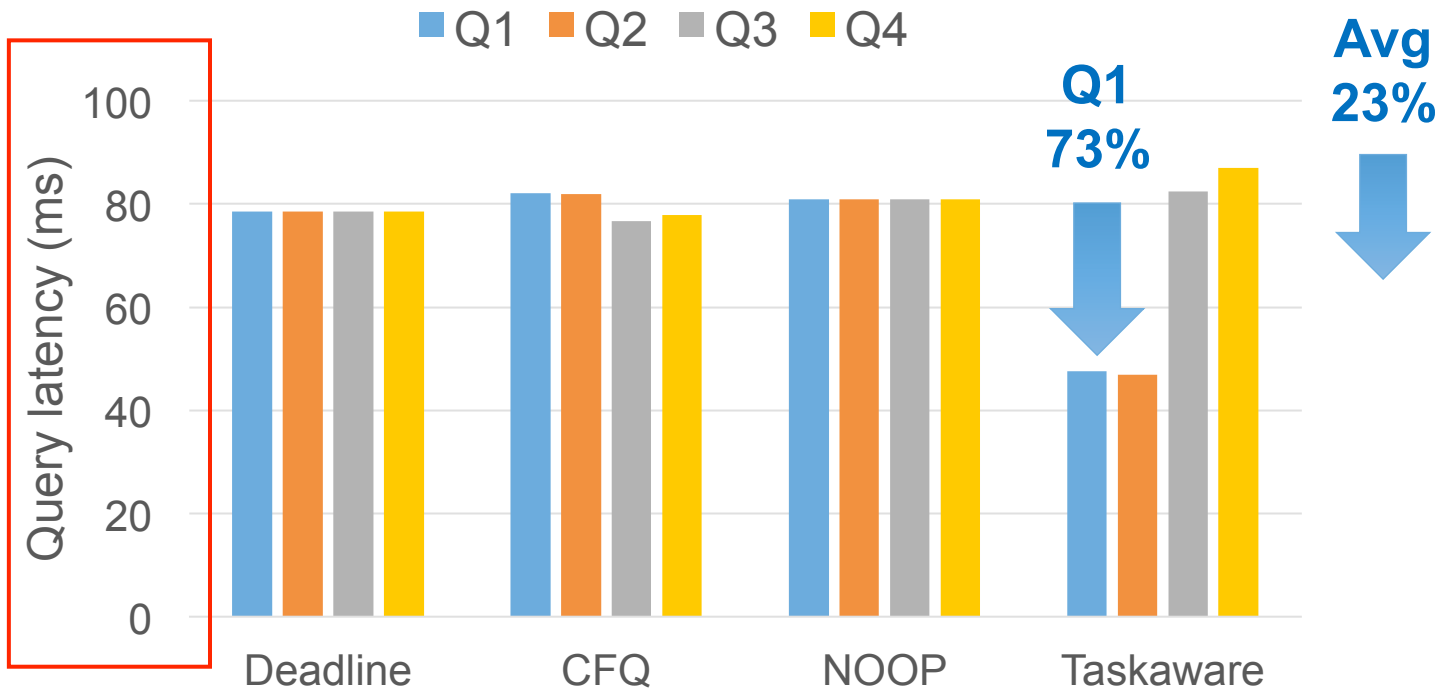
Recall

- Multiple independent layers



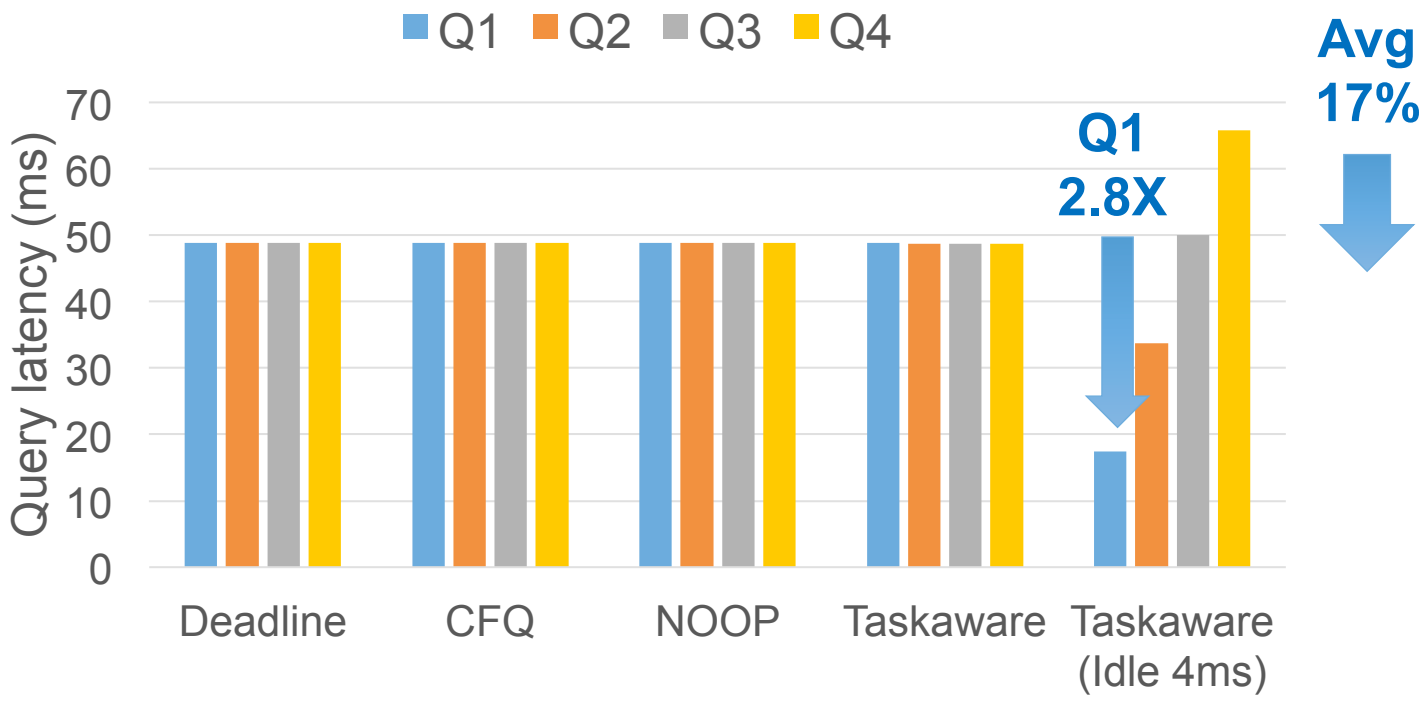
Optimizing Parallel Query

- Illustrative experiment (device queue off)



Optimizing Parallel Query

- Illustrative experiment (4ms idle_slice)



Optimizing Parallel Query

- Ongoing work
  - Handling heavy tasks
  - Extending to CPU scheduling

apposha

Solutions Cloud Service Contact Careers My Account

Unleash your database

Experience unparalleled performance with our V12 Engine

H http://apposha.io
F www.facebook.com/apposha
M sangwook@apposha.io

mongoDB PostgreSQL MySQL elastic