



Firebird and disk I/O

Dmitry Yemanov
dimitr@firebirdsql.org

Firebird Project
www.firebirdsql.org

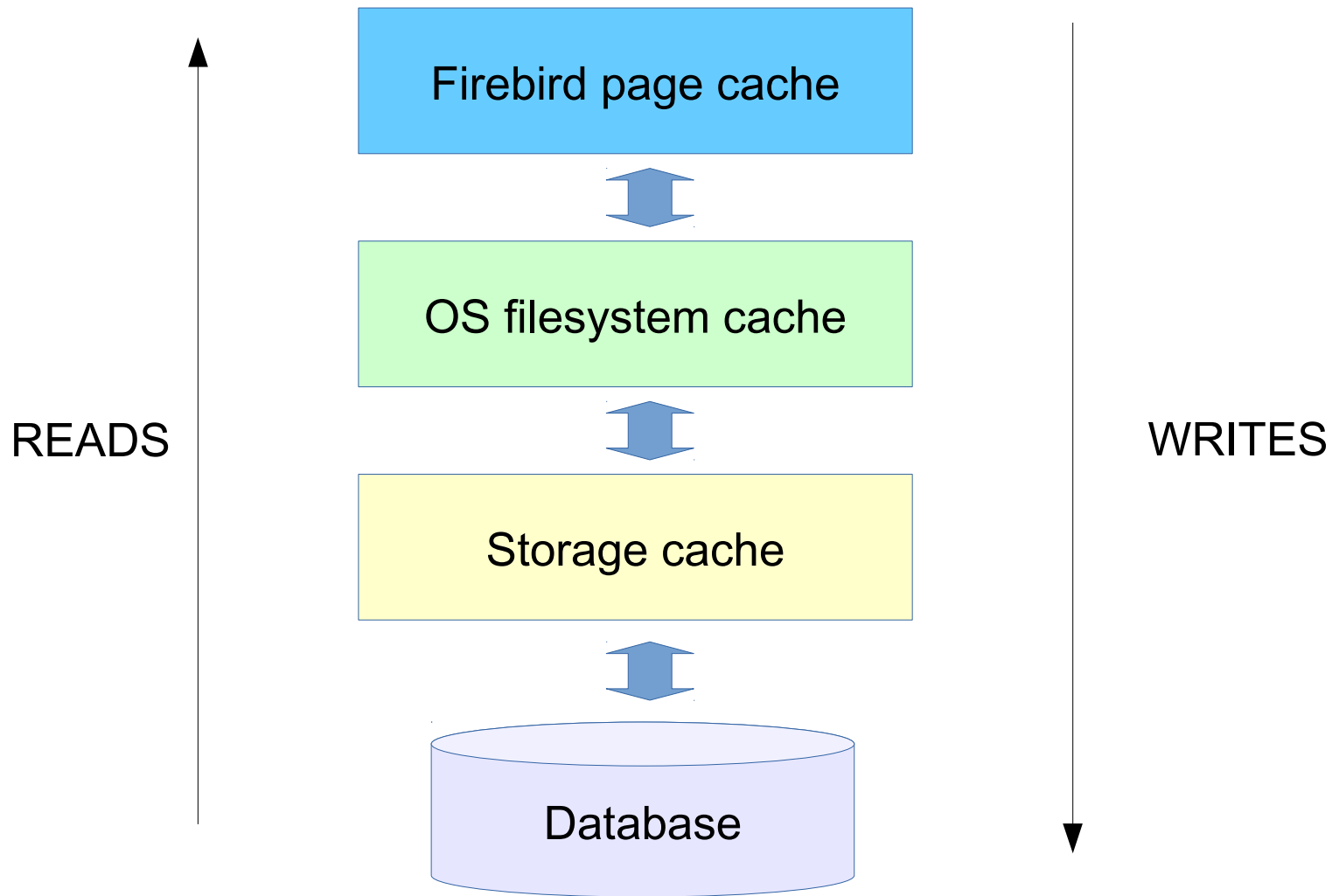
Page I/O operations

- Page reads
 - ♦ Single page is read from disk and copied to the page cache
 - ♦ Attempted to be sequential / uni-directional
 - ♦ Random under load
- When happens
 - ♦ Cache miss (all architectures)
 - ♦ Page was modified by other process (CS / SC)

Page I/O operations

- Page writes
 - Single page is written from the page cache to disk
 - Attempted to be uni-directional (partially)
 - Random under the load
- When happens
 - Transaction commit or rollback (all architectures)
 - Cache full of dirty pages (all architectures)
 - Other processes need modified pages (CS / SC)

I/O and caching





I/O optimization

- Tuning for faster I/O
 - DefaultDbCachePages / gfix -buffers
(bigger might be better or worse!)
 - TempCacheLimit / RAM disk
 - Reserve RAM for the OS filesystem cache
(up to 50%)
 - Fast storage, SSD preferred
 - RAID is a good idea

Durability

- Careful Writes
 - ♦ Page relationships (inter-page pointers)
 - ♦ Page dependency graph
 - ♦ Related pages are written one before another
 - record before index
 - backversion before record
 - fragment before record
 - data page before pointer page
 - etc
 - ♦ Disk image is consistent at any point of time
 - ♦ Orphan pages

Durability

- Careful Writes
 - ♦ Page relationships (inter-page pointers)
 - ♦ Page dependency graph
 - ♦ Related pages are written one before another
 - ♦ Disk image is consistent at any point of time
 - ♦ Orphan pages
- Recovery
 - ♦ Immediate, nothing special should be done
 - ♦ GFIX allows reuse of orphan pages

Durability

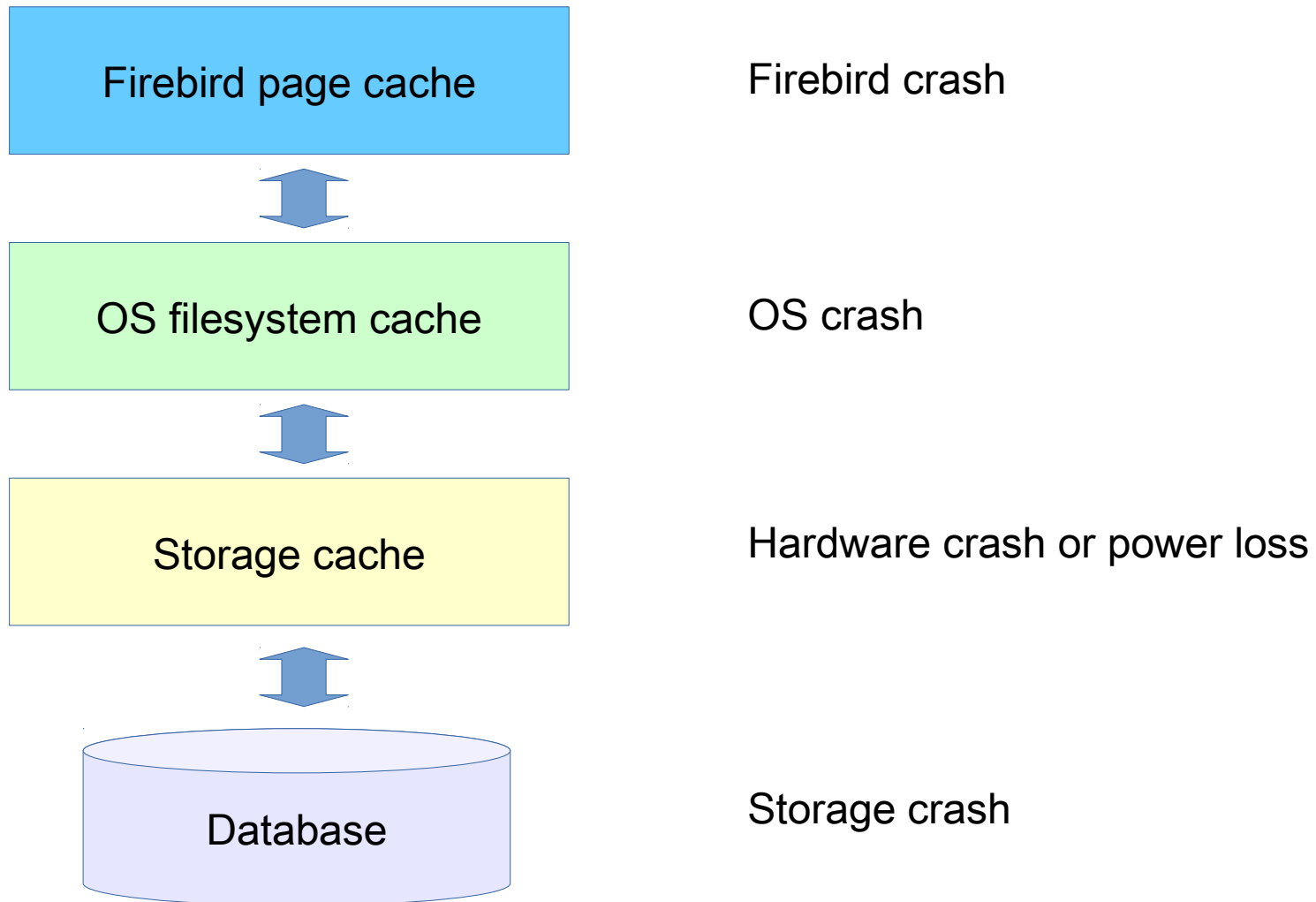
- Careful Writes
 - Order of writes is absolutely important
 - Writes must reach the disk

Durability

- Careful Writes
 - ♦ Order of writes is absolutely important
 - ♦ Writes must reach the disk

- Any problems?
 - ♦ Yes, caching may play against durability!
 - ♦ Write-through cache is OK
 - ♦ Write-back cache is BAD

Durability and caching





Filesystem cache

- OS filesystem cache
 - Often useful for reads (cold cache)
 - Extremely important for Classic
 - May dramatically affect performance for writes
 - Can be write-through or write-back
 - Flushed in background (async) or by request (sync)

Filesystem cache

- How to control?
 - Forced writes = ON → write-through
 - Forced writes = OFF → write-back
 - MaxUnflushedWrites: how many transactions can be lost
 - MaxUnflushedWriteTime: how much time of work can be lost

Filesystem cache

- Sample configurations
 - `MaxUnflushedWrites = -1, MaxUnflushedWriteTime = -1`
 - `MaxUnflushedWrites = 100, MaxUnflushedWriteTime = 5`
 - `MaxUnflushedWrites = -1, MaxUnflushedWriteTime = 1`
 - `MaxUnflushedWrites = 1, MaxUnflushedWriteTime = -1`

Filesystem cache

- Sample configurations
 - MaxUnflushedWrites = -1, MaxUnflushedWriteTime = -1
 - MaxUnflushedWrites = 100, MaxUnflushedWriteTime = 5
 - MaxUnflushedWrites = -1, MaxUnflushedWriteTime = 1
 - MaxUnflushedWrites = 1, MaxUnflushedWriteTime = -1
- Any problems?
 - Yes, OS can reorder writes!
 - Careful Writes idea is violated, data corruption is possible
 - UPS and storage w/BBU somewhat help, but...
 - OS crash may cause a disaster

Filesystem cache

- Durability again
 - Either FW=ON (write-through cache, synchronous writes)
 - Or FW=OFF with explicit cache flushes (MaxUnflushed*)

Filesystem cache

- Durability again
 - Either FW=ON (write-through cache, synchronous writes)
 - Or FW=OFF with explicit cache flushes (MaxUnflushed*)
- Any problems?
 - Yes, sync/flush requests can be ignored by lower levels!
 - Some storage interfaces do not support sync'ing
 - Smart storage caches play against durability

Filesystem barriers

- Barriers
 - ♦ Split sets of writes into ordered groups, down to page level granularity
 - ♦ Play in favor of the Careful Writes idea

Filesystem barriers

- Barriers
 - Split sets of writes into ordered groups, down to page level granularity
 - Play in favor of the Careful Writes idea
- Any problems?
 - Yes, the cost is very high!
 - Performance vs durability: choose only one

Performance vs durability

- Cached writes (FW=OFF)

barrier=0



49200 tpmC

barrier=1



42500 tpmC

Performance vs durability

- Cached writes (FW=OFF)



- Cached writes with forced flushes (FW=OFF + MaxUnflushed*)



Performance vs durability

- Cached writes (FW=OFF)



- Cached writes with forced flushes (FW=OFF + MaxUnflushed*)



- Uncached writes (FW=ON or mount=sync)



Performance vs durability

- Summary
 - FW=ON w/barriers provides absolute durability, but terribly slow
 - FW=ON w/o barriers is faster, but not 100% reliable
 - FW=OFF w/flushes with/without barriers is even faster, but allows some data loss and may violate Careful Writes during soft crashes
 - FW=OFF w/o flushes is the fastest, but absolutely unreliable

Performance vs durability

- Recommendations
 - Barriers may be avoided if BBU storage is used
 - Software RAID → better use barriers
 - SSD with/without Power Loss Protection
 - FW=ON is generally safe (see above)
 - FW=OFF may be used if
some predicted data loss is acceptable
or
some kind of redundancy is enforced
 - Using Windows w/o BBU or redundancy is questionable
 - Firebird on Windows PDC is a bad idea



Questions?