Enlightening the I/O Path:
A Holistic Approach for Application Performance

Sangwook Kim\textsuperscript{13}, Hwanju Kim\textsuperscript{2}, Joonwon Lee\textsuperscript{3}, and Jinkyu Jeong\textsuperscript{3}

Apposha\textsuperscript{1}
Dell EMC\textsuperscript{2}
Sungkyunkwan University\textsuperscript{3}
Data-Intensive Applications

Relational

Document

Key-value

Column

Search

mongoDB

CouchDB

MySQL

PostgreSQL

redis

riakKV

Apache

Cassandra

Elastic

Solr
Data-Intensive Applications

• Common structure

* Example: MongoDB
  - Client (foreground)
  - Checkpointer
  - Log writer
  - Eviction worker
  - …
Data-Intensive Applications

• Common structure

Background tasks are problematic for application performance

- Log writer
- Evict worker
- ...

Storage Device
Operating System
Client

I/O
I/O
I/O
I/O

Example: MongoDB - Server (client) - Checkpointer - Log writer - Evict worker - ...
Application Impact

- Illustrative experiment
  - YCSB update-heavy workload against MongoDB
Application Impact

• Illustrative experiment
  • YCSB update-heavy workload against MongoDB

![Graph showing operation throughput and elapsed time]

- CFQ: 30 seconds latency at 99.99th percentile
- Regular checkpoint task
Application Impact

- Illustrative experiment
  - YCSB update-heavy workload against MongoDB

I/O priority does not help
Application Impact

- Illustrative experiment
  - YCSB update-heavy workload against MongoDB

State-of-the-art schedulers do not help much
What’s the Problem?

- Independent policies in multiple layers
  - Each layer processes I/Os w/ limited information

- I/O priority inversion
  - Background I/Os can arbitrarily delay foreground tasks
What’s the Problem?

• Independent policies in multiple layers
  • Each layer processes I/Os with limited information

• I/O priority inversion
  • Background I/Os can arbitrarily delay foreground tasks
Multiple Independent Layers

- Independent I/O processing
Multiple Independent Layers

- Independent I/O processing

Abstraction

Application

Caching Layer

File System Layer

Block Layer

Storage Device

Buffer Cache

read() write()

admission control
Multiple Independent Layers

• Independent I/O processing
Multiple Independent Layers

- Independent I/O processing
Multiple Independent Layers

• Independent I/O processing
Multiple Independent Layers

- Independent I/O processing
What’s the Problem?

• Independent policies in multiple layers
  • Each layer processes I/Os w/ limited information

• I/O priority inversion
  • Background I/Os can arbitrarily delay foreground tasks
I/O Priority Inversion

• Task dependency

Diagram:
- Application
- Caching Layer
- File System Layer
- Block Layer
- Storage Device
- Locks
- Condition variables
I/O Priority Inversion

• Task dependency

Application

Caching Layer

File System Layer

Block Layer

Storage Device

Condition variables

FG → BG → I/O

lock

wait
I/O Priority Inversion

• Task dependency
I/O Priority Inversion

- Task dependency
I/O Priority Inversion

- I/O dependency

Diagram:
- Application
- Caching Layer
- File System Layer
- Block Layer
- Storage Device
- Outstanding I/Os
I/O Priority Inversion

• I/O dependency

For ensuring consistency and/or durability
Our Approach

• Request-centric I/O prioritization (RCP)
  • Critical I/O: I/O in the critical path of request handling
  • Policy: holistically prioritizes critical I/Os along the I/O path

100 ms latency at 99.99th percentile
Challenges

• How to accurately identify I/O criticality

• How to effectively enforce I/O criticality
Critical I/O Detection

• Enlightenment API
  • Interface for tagging foreground tasks

• I/O priority inheritance
  • Handling task dependency
  • Handling I/O dependency
I/O Priority Inheritance

• Handling task dependency

• Locks

• Condition variables
I/O Priority Inheritance

- Handling I/O dependency
I/O Priority Inheritance

• Handling I/O dependency
I/O Priority Inheritance

• Handling I/O dependency

Q admission stage

Sched queueing stage

Block Layer
I/O Priority Inheritance

• Handling I/O dependency

Q admission stage

Sched queueing stage

Block Layer

Non-critical I/O tracking

PER-DEV ROOT

NCIO

Descriptor
Location
Resolver
Sector #
I/O Priority Inheritance

• Handling I/O dependency

I/O

allocate

Q admission stage

Sched queueing stage

Block Layer

Non-critical I/O tracking

Descriptor
Location
Resolver
Sector #

PER-DEV
ROOT

NCIO

NCIO

NCIO

NCIO
I/O Priority Inheritance

- Handling I/O dependency

I/O Priority Inheritance

- Handling I/O dependency

Block Layer

Q admission stage

Sched queueing stage

Non-critical I/O tracking

Descriptor Location Resolver Sector #

PER-DEV ROOT

NCIO

NCIO

NCIO

NCIO
I/O Priority Inheritance

• Handling I/O dependency

---

I/O

Q admission stage

Sched queueing stage

Block Layer

Non-critical I/O tracking

Descriptor
Location
Resolver
Sector #

PER-DEV
ROOT

NCIO

NCIO

NCIO

NCIO
I/O Priority Inheritance

- Handling I/O dependency

Block Layer

Q admission stage

Sched queueing stage

I/O

Non-critical I/O tracking

Descriptor
Location
Resolver
Sector #

PER-DEV ROOT

NCIO

NCIO

NCIO

NCIO
I/O Priority Inheritance

- Handling I/O dependency

![Diagram of I/O Priority Inheritance](image)

- Q admission stage
- Sched queueing stage

Non-critical I/O tracking

- PER-DEV ROOT
- NCIO
- NCIO
- NCIO
- NCIO

Descriptor Location Resolver Sector #
I/O Priority Inheritance

• Handling I/O dependency

Block Layer

Q admission stage

Sched queueing stage

Non-critical I/O tracking

Descriptor Location Resolver Sector #

delete on completion
Handling Transitive Dependency

• Possible states of dependent task
Handling Transitive Dependency

• Recording blocking status

FG → BG → BG

- Blocked on task

FG → BG → I/O

- Blocked on I/O

FG → BG

- Blocked at admission stage
Handling Transitive Dependency

• Recording blocking status

- Task is recorded
- Blocked on I/O
- Blocked at admission stage
Handling Transitive Dependency

• Recording blocking status

- Task is recorded
- I/O is recorded
- Blocked at admission stage
Handling Transitive Dependency

- Recording blocking status

- Task is recorded
- I/O is recorded
- I/O is recorded
- retry
Challenges

• How to accurately identify I/O criticality
  • Enlightenment API
  • I/O priority inheritance
  • Recording blocking status

• How to effectively enforce I/O criticality
Criticality-Aware I/O Prioritization

• Caching layer
  • Apply low dirty ratio for non-critical writes (1% by default)

• Block layer
  • Isolate allocation of block queue slots
  • Maintain 2 FIFO queues
  • Schedule critical I/O first
  • Limit # of outstanding non-critical I/Os (1 by default)
  • Support queue promotion to resolve I/O dependency
Evaluation

• Implementation on Linux 3.13 w/ ext4

• Application studies
  • PostgreSQL relational database
    • Backend processes as foreground tasks
    • I/O priority inheritance on LWLocks (semop)
  • MongoDB document store
    • Client threads as foreground tasks
    • I/O priority inheritance on Pthread mutex and condition vars (futex)
  • Redis key-value store
    • Master process as foreground task
Evaluation

• Experimental setup
  • 2 Dell PowerEdge R530 (server & client)
  • 1TB Micron MX200 SSD

• I/O prioritization schemes
  • CFQ (default), CFQ-IDLE
  • SPLIT-A (priority), SPLIT-D (deadline) [SOSP’15]
  • QASIO [FAST’15]
  • RCP
Application Throughput

- PostgreSQL w/ TPC-C workload

Transaction throughput (trx/sec) vs. dataset size:

- 10GB dataset
- 60GB dataset
- 200GB dataset

Comparison between CFQ and CFQ-IDLE.
Application Throughput

- PostgreSQL w/ TPC-C workload
Application Throughput

- PostgreSQL w/ TPC-C workload
Application Throughput

- PostgreSQL w/ TPC-C workload

![Graph showing transaction throughput for different dataset sizes and queueing disciplines]
Application Throughput

- PostgreSQL w/ TPC-C workload
Application Throughput

• Impact on background task

![Graph showing application throughput](image-url)
Application Throughput

• Impact on background task
Application Throughput

• Impact on background task

Our scheme improves application throughput w/o penalizing background tasks
Application Latency

- PostgreSQL w/ TPC-C workload

![Graph showing CCDF (Complementary Cumulative Distribution Function) for transaction latency. The graph compares different queue policies (CFQ, CFQ-IDLE, SPLIT-A, SPLIT-D, QASIO). The x-axis represents transaction latency in milliseconds, and the y-axis represents the CCDF P[X<=x]. Notably, the graph shows that at the 99.9th percentile, transaction latencies exceed 2 seconds for all queue policies except QASIO. The legend points out that the red line, representing QASIO, is the only one that remains below the 2-second mark at the 99.9th percentile.]

Over 2 sec at 99.9th
Application Latency

- PostgreSQL w/ TPC-C workload

Our scheme is effective for improving tail latency

Over 2 sec at 99.9th

300 msec at 99.999th
Summary of Other Results

• Performance results
  • MongoDB: 12%-201% throughput, 5x-20x latency at 99.9th
  • Redis: 7%-49% throughput, 2x-20x latency at 99.9th

• Analysis results
  • System latency analysis using LatencyTOP
  • System throughput vs. Application latency
  • Need for holistic approach
Conclusions

• Key observation
  • All the layers in the I/O path should be considered as a whole with I/O priority inversion in mind for effective I/O prioritization

• Request-centric I/O prioritization
  • Enlightens the I/O path solely for application performance
  • Improves throughput and latency of real applications

• Ongoing work
  • Practicalizing implementation
  • Applying RCP to database cluster with multiple replicas
Thank You!

- Questions and comments

- Contact
  - sangwook@apposha.io