Log-Structured KV-Stores

- RocksDB
- Amazon DynamoDB
- riak
- WT
- cassandra
- Accumulo
- HBase
- Couchbase
- BigTable
- levelDB
Log-Structured KV-Stores
Why Log-Structured KV-Stores?
Why Log-Structured KV-Stores?

fast writes
Why Log-Structured KV-Stores?

memory

storage
Why Log-Structured KV-Stores?
Why Log-Structured KV-Stores?
Why Log-Structured KV-Stores?

byte-addressable

block-addressable
write data
write data
write data
In-Place Writes

write data
In-Place Writes

write data

SSD

B-trees
In-Place Writes

write data

SSD

B-trees
Log-Structured Writes
Log-Structured Writes

buffer writes
Log-Structured Writes

buffer writes
Log-Structured Writes

buffer writes
Log-Structured Writes

buffer writes
Log-Structured Writes

buffer writes

SSD
Log-Structured KV-Stores

fast writes
Log-Structured KV-Stores

- fast writes
- fast reads
- massive data
Background
Background

The Log-Structured Merge-Tree
Background

LSM-tree
buffer
writes

↓

buffer
key value pairs

buffer
key
Sherlock: a fictional detective
Waldo: an inconspicuous traveler

value

buffer
buffer gets full
buffer

sort & flush

SSD

sorted runs
buffer

exponentially increasing capacities

level 1 → SSD
level 2
level 3
where's Waldo

buffer

binary searching
where’s Waldo

down

buffer

pointers

one I/O per run

SSD
where’s Waldo

buffer

Bloom filters

true
negative

pointers

SSD

where's Waldo
where’s Waldo → buffer

Bloom filters

true negative
false positive

pointers

SSD
where’s Waldo

buffer

Bloom filters

true negative

false positive

ture positive

pointers

SSD

Bloom filters

true negative

false positive

ture positive
buffer

Bloom filters

pointers

SSD

merging frequency
merging

writes

reads
merging

writes

reads
Tiering
write-optimized
cassandra

Leveling
read-optimized
RocksDB

merging
Tiering
write-optimized

Leveling
read-optimized
Tiering
write-optimized

gather

Leveling
read-optimized
Tiering
write-optimized

Leveling
read-optimized

gather

merge & flush ↓
Tiering
write-optimized

gather

Leveling
read-optimized
Tiering
write-optimized

Leveling
read-optimized

gather

merge
Tiering
write-optimized

gather

Leveling
read-optimized

merge

flush
Tiering
write-optimized

gather

Leveling
read-optimized

merge
Log$_R(N)$

Tiering
write-optimized

Leveling
read-optimized
Tiering
write-optimized

Leveling
read-optimized

\[ \log_R(N) \]

size ratio
Tiering
write-optimized

Leveling
read-optimized

\[ \log_R(N) \]

size ratio

\( R \) runs per level

1 run per level
Tiering
write-optimized

Leveling
read-optimized

$R$ runs per level

$1$ run per level

size ratio $R$
Tiering
write-optimized

Leveling
read-optimized

size ratio $R \gg$
Tiering
write-optimized

Leveling
read-optimized

\( T \) runs per level

\( 1 \) run per level

\( \overset{\approx}{\text{size ratio}} \ R \)
Tiering
write-optimized

O( N ) runs per level
log

Leveling
read-optimized

1 run per level
sorted array

size ratio $R \uparrow$
Tiering

Leveling

log

sorted array
Tiering

Leveling

size ratio $R$

sorted array

log
Tiering Leveling

log

size ratio $R$

Leveling

$R$ sorted array
Monkey: Optimal Navigable Key-Value Store  SIGMOD17
Monkey: Optimal Navigable Key-Value Store

Niv Dayan
Manos Athanassouliis
Stratos Idreos
Monkey: Optimal Navigable Key-Value Store SIGMOD17
data

Bloom filters

bits/entry

×

×

×
data

Bloom filters

bits/entry

×

×

×
Bloom filters

false positive rate

$O(e^{-x})$

$O(e^{-x})$

$O(e^{-x})$
Bloom filters

false positive rate

$O(e^{-x})$

$O(e^{-x})$

$O(e^{-x})$

$= O(e^{-x} \cdot \log_R(N))$
Bloom filters

false positive rate

\[ O(e^{-x}) \]

\[ O(e^{-x}) \]

\[ O(e^{-x}) \]

\[ \cdots \log_R(N) \]

\[ = O(e^{-x} \cdot \log_R(N)) \]
Bloom filters

most memory

false positive rate

$O(e^{-x})$

$O(e^{-x})$

$O(e^{-x})$
Bloom filters

most memory

saves at most 1 I/O!

false positive rate

$O(e^{-x})$

$O(e^{-x})$

$O(e^{-x})$
reallocate
same memory - fewer false positives

reallocate
relax

false positive rates

\[ 0 < p_0 < 1 \]
\[ 0 < p_1 < 1 \]
\[ 0 < p_2 < 1 \]
false positive rates

\[ 0 < p_0 < 1 \]
\[ 0 < p_1 < 1 \]
\[ 0 < p_2 < 1 \]

relax

read cost

\[ \text{model} \]

\[ \text{read cost} = f(p_0, p_1 \ldots) \]

memory footprint

\[ \text{memory footprint} = f(p_0, p_1 \ldots) \]
false positive rates

\[ 0 < p_0 < 1 \]
\[ 0 < p_1 < 1 \]
\[ 0 < p_2 < 1 \]

model

read cost \[ = \sum_{i=1}^{L} p_i \]

memory footprint \[ = -\sum_{i}^{L} \frac{N}{T^{L-i}} \cdot \frac{\ln(p_i)}{\ln(2)^2} \]
false positive rates

\[ 0 < p_0 < 1 \]
\[ 0 < p_1 < 1 \]
\[ 0 < p_2 < 1 \]

model

\[ \text{read cost} = \sum_{i=1}^{L} p_i \]

\[ \text{memory footprint} = - \sum_{i=1}^{L} \frac{N}{T^{L-i}} \cdot \frac{\ln(p_i)}{\ln(2)^2} \]

in terms of \( p_0, p_1 \ldots \)

relax
false positive rate

\[ p_0 \approx O(e^{-x/R^2}) \]

\[ p_1 \approx O(e^{-x/R_1}) \]

\[ p_2 \approx O(e^{-x/R_0}) \]
false positive rate

\[ O(e^{-x/R^2}) \]
\[ O(e^{-x/R^1}) \]
\[ O(e^{-x/R^0}) \]

\hspace{0.5cm} \text{geometric progression} \hspace{0.5cm} = \hspace{0.5cm} O(e^{-x})
\( O(e^{-x} \cdot \log_R(N)) \quad > \quad O(e^{-x}) \)
\[ O(e^{-x} \cdot \log R(N)) \]

\[ O(e^{-x}) \]
\[ O(e^{-x} \cdot \log_R(N)) \]

Graph showing read latency (ms) vs. number of entries (log scale) with curves for RocksDB and Monkey, indicating different time complexities.
I/O overheads with leveling

point

long range

short range

writes
false positive rates

exponentially decreasing

$O(e^{-x/R})$

$O(e^{-x/R^2})$

$O(e^{-x})$
false positive rates

\[ O(e^{-x}/R^2) \]
\[ O(e^{-x}/R) \]
\[ O(e^{-x}) \]

point \rightarrow \text{largest level}
largest level
$O(e^{-x})$

point

long range

writes

short range
long range

target range

$O\left(\frac{s}{R^2}\right)$

$O\left(\frac{s}{R}\right)$

$O(s)$

(target key range)
target key range

O(\(s/R^2\))

O(\(s/R\))

O(\(s\))

largest level

long range
point
largest level
$O(e^{-x})$

long range
largest level
$O(s)$

short range

writes
short range

target range

1
1
1
1
short range

target range

\[ O(\log_R(N)) \]

all levels

Levels:

1

1

1
largest level
$O(e^{-x})$

point

largest level
$O(s)$

long range

all levels
$O(\log_R(N))$

short range

writes
writes

exponentially more work
writes

exponentially more work

exponentially less frequent
writes

all levels
merge 1

writes
write-amplification

\[ O(R) \]

\[ O(R) \]

\[ O(R) \]

\[ O(R) \]
writes

\[ O(R) \]

\[ O(R) \]

\[ O(R) \]

\[ \mathcal{O}(R \cdot \log_R(N)) \]
\[ \text{point} \rightarrow \text{long range} \rightarrow \text{short range} \rightarrow \text{writes} \]

\begin{align*}
\text{largest level} & : O(e^{-x}) = O(e^{-x}/R^2) + O(e^{-x}/R) + O(e^{-x}) \\
\text{largest level} & : O(s) = O(s/R^2) + O(s/R) + O(s) \\
\text{all levels} & : O(\log_R(N)) = 1 + 1 + 1 \\
\text{all levels} & : O(R \cdot \log_R(N)) = O(R) + O(R) + O(R)
\end{align*}
\( O(e^{-x}) \) \rightarrow \ O(s) \rightarrow \ O(R) \rightarrow \ O(e^{-x}) \)

\( \text{point} \rightarrow \text{long range} \rightarrow \text{writes} \rightarrow \text{all levels} \)
point

long range

largest level

largest level

\( O(e^{-x}) \) \( \Rightarrow \) \( O(s) \)

\( O(R) \) \( \Rightarrow \) \( O(R) \) \( \Rightarrow \) \( O(R) \) \( \Rightarrow \) \( O(R) \) 

writes

all levels

superfluous
for point lookups and long range lookups

merging at smaller levels is superfluous
worse as data grows!
poor performance
poor performance

lower device lifetime (on SSD)
Dostoevsky

SIGMOD18
Dostoevsky: Space-Time Optimized Evolvable Scalable Key-Value Store
very write-optimized

Doostoevsky: **Space-Time Optimized Evolvable Scalable Key-Value Store**
Tiering
write-optimized

Leveling
read-optimized
Tiering
write-optimized

Lazy Leveling
mixed-optimized

Leveling
read-optimized
Lazy Leveling

\[ \text{Tiering} \leftarrow \text{Leveling} \]
Lazy Leveling

\[
\begin{align*}
\Rightarrow & \text{ merge when level fills} \\
\Rightarrow & \text{ merge to have at most 1 run}
\end{align*}
\]
point

long range

short range

writes
false positive rates

\[ O\left(\frac{e^{-x}}{R^3}\right) \]

\[ O\left(\frac{e^{-x}}{R^2}\right) \]

\[ O\left(\frac{e^{-x}}{R}\right) \]

\[ O\left(e^{-x}\right) \]
false positive rates

\[ O(e^{-x}/R^3) \]
\[ O(e^{-x}/R^2) \]
\[ O(e^{-x}) \]
false positive rates

\[ O(e^{-x}/R^3) \]

\[ O(e^{-x}/R^2) \]

\[ O(e^{-x}) \]
point

O(e^{-x})
Point

$O(e^{-x})$

$O(\log R(N) \cdot R \cdot e^{-x})$

with uniform FPRs

$\begin{align*}
O(e^{-x}) \\
O(e^{-x}) \\
O(e^{-x}) \\
O(e^{-x})
\end{align*}$
point

$O(e^{-x})$

long range

short range

writes
long range

target range

$O(s/R^2)$

$O(s/R)$

$O(s)$

target key range
long range

target range

$O(s/R^2)$

$O(s/R)$

$O(s)$

largest level

target key range
point

O(e^{-x})

long range

O(s)

short range

writes
\[ O(R) \cap O(1 + R \cdot (\log R(N) - 1)) \]

short range

target key range
point $O(e^{-x})$

long range $O(s)$

short range $O(1 + R \cdot (\log_R N - 1))$

writes
write-amplification

\[ O(1) \]

\[ O(N) \]

\[ O(R) \]
write-amplification

\[ O(1) \]
\[ O(1) \]
\[ O(R) \]

\[ O(R + \log_R(N)) \]
\( O(e^{-x}) \) \hspace{1cm} \text{point} \hspace{1cm} \text{long range} \hspace{1cm} \text{short range} \hspace{1cm} \text{writes} \\

\( O(s) \) \hspace{1cm} \hspace{1cm} \hspace{1cm} O(1 + R \cdot (\log_R(N) - 1)) \hspace{1cm} O\left( R + \log_R(N) \right) \)
Lazy Leveling

Leveling

\[ O(e^{-x}) \quad O(s) \quad O(1 + R \cdot (\log_R(N) - 1)) \quad O( R + \log_R(N) ) \]

\[ O(e^{-x}) \quad O(s) \quad O(\log_R(N)) \quad O( R \cdot \log_R(N) ) \]
**Tiering**  
\[ O(R \cdot e^{-x}) \quad O(R \cdot s) \quad O(R \cdot \log_R(N)) \quad O(\log R(N)) \]

**Lazy Leveling**  
\[ O(e^{-x}) \quad O(s) \quad O(1 + R \cdot (\log_R(N) - 1)) \quad O(R + \log R(N)) \]

**Leveling**  
\[ O(e^{-x}) \quad O(s) \quad O(\log_R(N)) \quad O(R \cdot \log_R(N)) \]

- **point**  
- **long range**  
- **short range**  
- **writes**
Leveling writes

Lazy Leveling

Tiering

point

writes
Leveling

long range

Tiering

Lazy Leveling

Leveling

writes
Tiering  Lazy Leveling  Leveling
Tiering writes

Lazy Leveling

Leveling
Tiering
writes

Lazy Leveling

Leveling
short range
Tiering writes

Lazy Leveling writes & point

Leveling short range
Lazy Leveling

Fluid

Tiering

Leveling
Fluid LSM-Tree
Lazy Leveling

Fluid LSM-Tree
Lazy Leveling

\[
\begin{align*}
\text{point} & \quad \text{long range} & \quad \text{short range} & \quad \text{writes} \\
\{ & \quad R \text{ runs} & \{ & 1 \text{ runs} \\
& \quad \text{graphical elements} & & \text{graphic elements}
\end{align*}
\]
Lazy Leveling

point  long range  optimize  short range  writes

\[ \{ R \text{ runs} \} \]
\[ \{ 1 \text{ runs} \} \]
Lazy Leveling

- point
- long range
- optimize
- short range
- writes

2 runs
1 runs
Leveling

optimize

short range

writes

long range

point

1 runs

1 runs

1 runs
Leveling

- long range
- short range

optimize writes

1 runs

1 runs
point  long range  short range  optimize  writes

Lazy Leveling

\[ \begin{align*}
\{ & R \text{ runs} \\
& 1 \text{ runs} \\
\end{align*} \]
Lazy Leveling

point

long range

short range

optimize writes

R runs

2 runs
point long range short range optimize writes

Tiering

\[
\begin{align*}
&\{R \text{ runs} \\
&\{R \text{ runs}
\end{align*}
\]
optimize

point

long range  short range  writes

Tiering

R runs

R runs
Lazy Leveling

- $1$ runs
- $R$ runs

- point
- long range
- short range
- writes
Lazy Leveling

optimize

point

long range

short range

writes

$R$ size ratio

$R$ runs

1 runs
optimize

point

long range  short range  writes

Lazy Leveling

$R$ size ratio

$\{ R \text{ runs} \}$

$\{ 1 \text{ runs} \}$
Fluid LSM-Tree

- $R$ size ratio
- $K$ runs at smaller levels
- $Z$ runs at largest level
Fluid LSM-Tree

Tiering

Lazy Leveling

Leveling
The graph illustrates the normalized throughput (ops/s) as a function of point lookups and updates. As the number of lookups/updates increases, the throughput also increases. There is a noticeable leveling off at higher throughput levels, indicating a saturation point.
Conclusion
Conclusion

Bloom filters

LSM-tree
Conclusion

Bloom filters

optimizes memory allocation

LSM-tree
Conclusion

Bloom filters optimizes memory allocation

LSM-tree removes superfluous merging
Conclusion
Conclusion

tiering (1997)

leveling (1996)
Conclusion

- Tiering (1997)
- Leveling (1996)

little memory
Conclusion

ample memory

tiering (1997)

leveling (1996)
Conclusion

ample memory

tiering (1997)

leveling (1996)
Conclusion

ample memory

tiering (1997)
leveling (1996)
Conclusion

ample memory

tiering (1997)

leveling (1996)
Conclusion
Conclusion
Conclusion
Conclusion