GeckoFTL
Scalable Flash Translation Techniques for Very Large Flash Devices

Niv Dayan, Philippe Bonnet, Stratos Idreos
SD Card | eMMC | SSD | Enterprise SSD

↑ performance
↓ power consumption
Block

Pages

4-32 KB
Constraints

- reads/writes at page granularity
- write latency > read latency
- sequential writes within a block
- block-erase before update
- limited erases per block

Block

Pages

4-32 KB
Constraints

reads/writes at page granularity
write latency > read latency
sequential writes within a block
block-erase before update
limited erases per block
Constraints

reads/writes at page granularity

write latency > read latency

sequential writes within a block

block-erase before update

limited erases per block

Block

Pages

4-32 KB
Constraints

- reads/writes at page granularity
- write latency > read latency
- **sequential writes within a block**
- block-erase before update
- limited erases per block
Constraints

- reads/writes at page granularity
- write latency > read latency
- sequential writes within a block
- block-erase before update
- limited erases per block

4-32 KB
Constraints

- reads/writes at page granularity
- write latency > read latency
- sequential writes within a block
- block-erase before update
- limited erases per block
Flash Translation Layer
(FTL)

Application

Flash
Out-of-place updates

Update logical page X

<table>
<thead>
<tr>
<th>valid</th>
<th>valid</th>
<th>valid</th>
<th>invalid</th>
<th>valid</th>
<th>valid</th>
<th>valid</th>
</tr>
</thead>
</table>

| valid | free | free | free | free | free | free |

5
Garbage-collection

Migrate

<table>
<thead>
<tr>
<th>valid</th>
<th>free</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid</td>
<td>free</td>
</tr>
<tr>
<td>invalid</td>
<td>free</td>
</tr>
<tr>
<td>valid</td>
<td>free</td>
</tr>
<tr>
<td>invalid</td>
<td>free</td>
</tr>
<tr>
<td>valid</td>
<td>free</td>
</tr>
<tr>
<td>invalid</td>
<td>free</td>
</tr>
</tbody>
</table>

- Migrate invalid entries to free.
## Garbage-collection

<table>
<thead>
<tr>
<th>Erase</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|  |  |  |  |  |
|---|---|---|---|
| valid | valid | valid | free |
| free  | free  | free  | free  |
Page Translation

Integrated RAM

Flash

Mapping table
User data

Mapping Cache
Page Translation

**Integrated RAM**

- Mapping Cache

**Flash**

- Mapping table

**Page Validity Bitmap (PVB)**

... 0 0 0 0 0 1 0 0 ...

**User data**

... valid valid valid valid invalid valid valid ...
Two concerns

1. Cost
2. Recovery

Integrated RAM

Mapping Cache

Page Validity Bitmap (PVB)

... 0 0 0 0 0 1 0 0 ...

Flash

Mapping table

User data

... valid valid valid valid invalid valid valid ...

...
Two concerns
1. Cost
2. Recovery

Integrated RAM

Mapping Cache

Page Validity Bitmap (PVB)
...
0 0 0 0 0 1 0 0 ...

Mapping table

User data
...
...

Flash
PVB $\propto$ device capacity

RAM

Recovery

\[ \begin{align*}
\text{Integrated RAM} & \quad 100 \text{ MB} \\
& \quad 1 \text{ GB} \\
& \quad 0 \\
& \begin{array}{c}
128 \text{ GB} \\
1 \text{ TB} \\
8 \text{ TB}
\end{array}
\end{align*} \]

\[ \begin{align*}
\text{Recovery time} & \quad 10 \text{ sec} \\
& \quad 100 \text{ sec} \\
& \quad 0 \\
& \begin{array}{c}
128 \text{ GB} \\
1 \text{ TB} \\
8 \text{ TB}
\end{array}
\end{align*} \]
PVB $\propto$ device capacity

PVB is the bottleneck.
Simple solution: store PVB in flash

- RAM
- Recovery
- Performance
- Device lifetime

Integrated RAM

Flash

PVB

... 0 0 0 0 0 0 1 0 0 ...

...
Simple solution: store PVB in flash

- RAM
- Performance
- Recovery
- Device lifetime

How to store PVB in flash without harming performance or device lifetime?
Solution:
GeckoFTL
Insight

PVB

... 0 0 0 0 0 0 0 0 0 ...

Block X
Insight

Page Invalidation $\leftrightarrow$ PVB

$\ldots 0 0 0 0 0 0 0 0 0 \ldots$

Block X
Insight

Page Invalidation ↔ PVB

... 0 0 1 0 0 0 0 0 0 ...

Block X
Insight

Page Invalidation $\xrightarrow{\quad} \text{PVB} \quad \begin{array}{cccccc}
... & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & ... 
\end{array}$

Block X
Insight

Page Invalidation $\rightarrow$ PVB $\rightarrow$ Garbage Collection

Block X

... 1 0 1 1 1 0 1 1 ...
Insight

Page Invalidation $\xrightarrow{\text{PVB}}$ Garbage Collection

... 0 0 0 0 0 0 0 0 ...  

Block X

PVB is update-intensive
Optimize for Updates

Buffer modifications in RAM
Reorganize later
LSM-tree variant

Integrated RAM

Flash

buffer

level 0

level 1

level 2

…
Page Invalidation

Block ID | Validity bitmap | Erase flag
---------|----------------|----------
4         | 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 |
Page Invalidation

Integrated RAM

Flash

buffer

Insert entry

level 0

level 1

level 2

…
Garbage Collection

Block ID  Validity bitmap  Erase flag

4  0 0 0 0 0 0 0 0 0 0 0 0 1

Reset bitmap
Flush & Merge

Integrated RAM

Flash

buffer

sort & flush

level 0

level 1

level 2

...
Flush & Merge

Integrated RAM

Flash

buffer

<table>
<thead>
<tr>
<th>level 0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>………….</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>level 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>………….</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>………….</td>
</tr>
</tbody>
</table>

…
Flush & Merge

Integrated RAM

Flash

buffer

level 0

level 1

…… 4 ……

…… 4 ……

level 2

sort-merge consolidate

Flush & Merge
Merge bitmaps

<table>
<thead>
<tr>
<th>Block ID</th>
<th>Validity Bitmap</th>
<th>Erase flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newer entry</td>
<td>4</td>
<td>0 0 1 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>Older entry</td>
<td>4</td>
<td>0 0 0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

OR

<table>
<thead>
<tr>
<th>Block ID</th>
<th>Validity Bitmap</th>
<th>Erase flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0 0 1 0 0 0 0 0 0 0 1</td>
<td>1 0 0 0 0</td>
</tr>
</tbody>
</table>
Merge bitmaps

<table>
<thead>
<tr>
<th>Block ID</th>
<th>Validity Bitmap</th>
<th>Erase flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newer entry</td>
<td>4 0 0 1 0 0 0 0 0 0 0 0 0 0 1</td>
<td></td>
</tr>
<tr>
<td>Older entry</td>
<td>4 0 0 0 0 0 0 0 0 0 1 0 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

IGNORE
Garbage Collection

Integrated RAM

Flash

buffer
…………

← search

level 0
…………

← search

level 1
………………

← search

Continue until finding entry with true erase flag

level 2
…………

…
Simulation

Baselines:

PVB in RAM

PVB in flash
Simulation

Baselines:
- PVB in RAM
- PVB in flash

Metrics
- RAM required
- Recovery time
- IO overheads
>90% RAM reduction

![Graph showing RAM reduction comparison between Baseline and GeckoFTL.](chart.png)
>50% recovery time reduction
>95% IO overheads reduction
Other Contributions

- Garbage collect FTL metadata
- Recovering RAM-resident FTL metadata
- Partition entries to maximize buffer utilization
- Wear leveling
Conclusion

GeckoFTL stores page validity metadata in flash

Buffer modifications
Reorganize later
Conclusion

GeckoFTL stores page validity metadata in flash

- Buffer modifications
- Reorganize later

- Integrated RAM
- Recovery time
- IO overheads
Conclusion

GeckoFTL stores page validity metadata in flash

Buffer modifications
Reorganize later

Integrated RAM
Recovery time
IO overheads

General applicability:

Store an update-intensive bitmap in secondary storage where adjacent bits are reset at the same time
Conclusion

GeckoFTL stores page validity metadata in flash

Buffer modifications
Reorganize later

Integrated RAM
Recovery time
IO overheads

General applicability:

Store an update-intensive bitmap in secondary storage where adjacent bits are reset at the same time

Thank you.