RAIL: Predictable, Low Tail Latency for NVMe Storage

Heiner Litz    Ana Klimovic
Christos Kozyrakis
Flash in the Datacenter

• Leverage low latency for user facing apps

• Disaggregate to improve utilization

• Share SSDs among multiple tenants

• High fan out → Predictable, low tail latency
Flash Performance

Flash Performance (Average)
– 1M IOPS (4 GB/s)
– Read latency: 80 µs
– Write latency: 2 ms
– Erase latency: 5 ms
• Flash read performance degrades with increasing % write
Read – Write Interference

- SSDs deploy parallel NAND chips as the logical unit (LUN) of parallelism

- Single outstanding operation per LUN

- Order ~100 LUNs per device

- Read latency is caused by reads getting stuck behind write or erase on same LUN
RAIL

• **Redundant Array of Independent LUNs**

• Use redundant data to reduce tail read latency

• When reading from a busy LUN, instead re-compute data from parity

• **GOAL:** Minimize probability that reads conflict with writes.
Read conflict probability (128 LUNs)

21x Reduction
Implementation: LightNVM and Open-Channel SSDs
Results

99% Read

95% Read

90% Read

80% Read
Software FTLs

• Enable new exciting opportunities
  – Full control over HW, resource allocation, scheduling

• Application specific trade-offs
  – RAIL: Tail latency vs. capacity & IOPS
  – GC: Spare capacity vs. erases & writes
  – QoS: Latency sensitive vs. batch

• Challenges
  – Incorrect SW can wear out devices quickly
  – Which operations should be in HW/SW?
  – Monitoring/feedback from HW to SW?
Conclusion

- RAIL reduces tail latency by 5x

- At the cost of write bandwidth and capacity

- Software FTL enables user configurable tradeoffs:
  - Latency
  - Write bandwidth
  - Write & read amplification
  - Fault tolerance
  - Capacity
Backup
Non-Conflicting Placements

<table>
<thead>
<tr>
<th>Number of LUNs</th>
<th># of conflict-free k-tuples</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>128</td>
<td>256</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
</tr>
<tr>
<td>512</td>
<td>4096</td>
</tr>
<tr>
<td>1024</td>
<td>16384</td>
</tr>
<tr>
<td>256</td>
<td>65536</td>
</tr>
</tbody>
</table>

- RAIL optimal k=3
- RAIL optimal k=4
- RAIL optimal k=5
- RAIL greedy k=3
- RAIL greedy k=4
- RAIL greedy k=5
- Conventional SSD