The CASE of FEMU:
Cheap, Accurate, Scalable and Extensible Flash Emulator

Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Matias Bjørling, Haryadi S. Gunawi

Why a new SSD Emulator?

<table>
<thead>
<tr>
<th>Platform</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator</td>
<td>Cheap; Easy; Time-saving</td>
<td>Trace-driven; Internal research only</td>
</tr>
<tr>
<td>Emulator</td>
<td>Cheap; Full-stack research support</td>
<td>Poor scalability; Poor accuracy</td>
</tr>
<tr>
<td>Hardware</td>
<td>Full-stack research support; Accurate</td>
<td>Expensive; Complex; Wear-out</td>
</tr>
</tbody>
</table>

Why Emulator?
- Get the benefits of both simulators and hardware platforms
- Enable wide range of SSD research, including SDF/Split-level Architecture and host-SSD co-designs, etc.

Why FEMU?
Bleak Status of Existing SSD Emulators
- FlashEmu: no longer maintained
- VSSIM: non-scalable; inaccurate<sub>app-level</sub>
- LightNVMe’s QEMU: only single-channel support; non-scalable

FEMU Use Case

- Supported Research:
  - Kernel changes
  - Interface changes
  - SSD FTL changes

FEMU Scalability: QEMU Virtual IO Optimization

> Eliminate VM-exits via Polling

> Customize QEMU AIO Path

Why bother optimizing QEMU?
A high performance base environment is needed to:
- Emulate NAND operations at ~100us level
- Emulate tens of parallel NAND flash chips

FEMU Accuracy

Delay Emulation:
- Endio queue: requests sorted according to completion time
- Periodic polling for request completion time expiration

Filebench

Latency Error: 11-47% ⇒ 0.5-38%

Future Work
- Further QEMU optimizations to support more scalability
- Improve accuracy by integrating more detailed SSD information
- Integrate well-implemented FTLs in popular SSD Simulators
- Multi-core support