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

Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminatahan Sundararaman*, Matias Bjørling†, Haryadi S. Gunawi
What SSD platforms are used?
What SSD platforms are used?

- Simulator
- Emulator
- Hardware Platform
### What SSD platforms are used?

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Emulator</th>
<th>Hardware Platform</th>
</tr>
</thead>
</table>

2

FEMU @ FAST ’18
What SSD platforms are used?

- Simulator:
  - DiskSim+SSD
  - SSDSim
  - FlashSim

- Emulator

- Hardware Platform
What SSD platforms are used?

- **Simulator**
  - DiskSim+SSD
  - SSDSim
  - FlashSim
  - Simple
  - Time-saving

- **Emulator**

- **Hardware Platform**
What SSD platforms are used?

Simulator
- DiskSim+SSD
- SSDSim
- FlashSim

Emulator

Hardware Platform

- Simple
- Time-saving

57%
What SSD platforms are used?

- Simulator:
  - DiskSim+SSD
  - SSDSim
  - FlashSim
  - Simple
  - Time-saving
  - Trace driven
  - Internal-research only

- Emulator

- Hardware Platform

57%
What SSD platforms are used?

Simulator
- DiskSim+SSD
- SSDSim
- FlashSim

Emulator

Hardware Platform

- Simple
- Time-saving
- Trace driven
- Internal-research only

57%

Trends
- Software-Defined Flash
- Split-Level Architecture
Simulator

- DiskSim+SSD
- SSDSim
- FlashSim

Emulator

- Simple
- Time-saving
- Trace driven
- Internal-research only

Hardware Platform
Simulator

- DiskSim+SSD
- SSDSim
- FlashSim

Emulator

- Simple
- Time-saving
- Trace driven
- Internal-research only

Hardware Platform

- OpenSSD
- OpenChannel-SSD
**Simulator**
- DiskSim+SSD
- SSDSim
- FlashSim
- ✔ Simple
- ✔ Time-saving
- ✗ Trace driven
- ✗ Internal-research only

**Emulator**

**Hardware Platform**
- OpenSSD
- OpenChannel-SSD
- ✔ Full-stack Research
- ✔ Accurate
**Simulator**
- DiskSim+SSD
- SSDSim
- FlashSim

- Simple
- Time-saving
- Trace driven
- Internal-research only

**Emulator**

**Hardware Platform**
- OpenSSD
- OpenChannel-SSD

- Full-stack Research
- Accurate

20%
Simulator

- DiskSim+SSD
- SSDSim
- FlashSim

- Simple
- Time-saving
- Trace driven
- Internal-research only

Emulator

Hardware Platform

- Full-stack Research
- Accurate
- Expensive
- Complex to use
- Wear-out

20%
Simulator

- DiskSim+SSD
- SSDSim
- FlashSim

- Simple
- Time-saving
- Trace driven
- Internal-research only

Emulator

- Full-stack Research
- Accurate
- Expensive
- Complex to use
- Wear-out

Hardware Platform

- OpenSSD
- OpenChannel-SSD

20% Single SSD

19% Single SSD
**Simulator**

- DiskSim+SSD
- SSDSim
- FlashSim

- Simple
- Time-saving
- Trace driven
- Internal-research only

**Emulator**

- 20%

  - 19% Single SSD
  - 1% Distributed SSDs

**Hardware Platform**

- Full-stack Research
- Accurate
- Expensive
- Complex to use
- Wear-out

- OpenSSD
- OpenChannel-SSD
**Simulator**
- DiskSim+SSD
- SSDSim
- FlashSim

- Simple
- Time-saving
- Trace driven
- Internal-research only

**Emulator**
- LightNVM’s QEMU
- VSSIM
- FlashEm

**Hardware Platform**
- OpenSSD
- OpenChannel-SSD

- Full-stack Research
- Accurate
- Expensive
- Complex to use
- Wear-out
**Simulator**
- DiskSim+SSD
- SSDSim
- FlashSim
- ✔ Simple
- ✔ Time-saving
- ✗ Trace driven
- ✗ Internal-research only

**Emulator**
- LightNVM’s QEMU
- VSSIM
- FlashEm
- ✔ Fullstack Research
- ✔ Cheap

**Hardware Platform**
- OpenSSD
- OpenChannel-SSD
- ✔ Full-stack Research
- ✔ Accurate
- ✗ Expensive
- ✗ Complex to use
- ✗ Wear-out
**Simulator**
- DiskSim+SSD
- SSDSim
- FlashSim
- Simple
- Time-saving
- Trace driven
- Internal-research only

**Emulator**
- LightNVM’s QEMU
- VSSIM
- FlashEm
- Fullstack Research
- Cheap
- Poor Scalability
- Poor Accuracy

**Hardware Platform**
- OpenSSD
- OpenChannel-SSD
- Full-stack Research
- Accurate
- Expensive
- Complex to use
- Wear-out
The “CASE” of FEMU

FEMU: QEMU/Software based Flash Emulator
The “CASE” of FEMU

FEMU: QEMU/Software based Flash Emulator

- Cheap: $0, https://github.com/ucare-uchicago/femu
The “CASE” of FEMU

FEMU: QEMU/Software based Flash Emulator

- **Cheap:** $0, https://github.com/ucare-uchicago/femu
- **Accurate:** 0.5-38% error rate in latency
  - 11% average at microsecond level
The “CASE” of FEMU

FEMU: QEMU/Software based Flash Emulator

- **Cheap**: $0, https://github.com/ucare-uchicago/femu
- **Accurate**: 0.5-38% error rate in latency
  - 11% average at microsecond level
- **Scalable**: support 32 channels/chips
The “CASE” of FEMU

FEMU: QEMU/Software based Flash Emulator

- **Cheap**: $0, https://github.com/ucare-uchicago/femu
- **Accurate**: 0.5-38% error rate in latency
  - 11% average at microsecond level
- **Scalable**: support 32 channels/chips
- **Extensible**
  - modifiable interface
  - modifiable FTL
What is FEMU?

Typical Fullstack Research          FEMU Fullstack Research
What is FEMU?

Typical Fullstack Research

- App
- Host OS
- Hardware Platform

FEMU Fullstack Research
What is FEMU?

Typical Fullstack Research

FEMU Fullstack Research

App

Host OS

Hardware Platform

FEMU

QEMU
What is FEMU?

Typical Fullstack Research

- App
- Host OS
- Hardware Platform

FEMU Fullstack Research

- App
- Guest OS
- VM
- FEMU
- QEMU
What is FEMU?

Typical Fullstack Research  
- App
- Host OS
- Hardware Platform

FEMU Fullstack Research  
- App
- Guest OS
- VM
- FEMU
- QEMU

Supported research:
What is FEMU?

Typical Fullstack Research

- App
- Host OS
- Hardware Platform

FEMU Fullstack Research

- App
- Guest OS
- VM

FEMU Fullstack Research

- FEMU
- QEMU

Supported research:

- FTL changes

FEMU Fullstack Research
What is FEMU?

Typical Fullstack Research

- App
- Host OS
- Hardware Platform

FEMU Fullstack Research

- App
- Guest OS
- VM
- NVMe
- FEMU
- QEMU

Supported research:
- Kernel changes
- Interface changes
- FTL changes
QEMU Scalability

Guest OS

QEMU

IO  IO  IO
QEMU Scalability

Guest OS

QEMU

IO

IO

IO

Diagram showing IO latency (us) vs. number of threads.
QEMU Scalability

Guest OS

QEMU

IO

...
QEMU Scalability

IO
IO
...
IO

IO Latency (us)

Guest OS
QEMU

# of threads
QEMU Scalability

![Graph showing IO latency (us) vs. number of threads. The expected latency is indicated.]
QEMU IDE Scalability

- 1 IO thread
- Guest OS
- QEMU

Graph showing IO Latency (us) vs. # of threads. Expected latency remains constant as the number of threads increases.
Guest OS

QEMU

2 IO threads

Expected

IO Latency (us)

# of threads

1 2 4 8 16 32 64
Expected

![Graph showing IO latency vs. number of threads]

- IO
- Guest OS
- QEMU

# of threads

IO Latency (us)
Guest OS

QEMU

![Graph showing IO Latency vs. # of threads]

- Expected
QEMU

Guest OS

IO

IO Latency (us)

0 50 100 200 300 400

1 2 4 8 16 32 64

# of threads

Expected

Represent

VSSIM
QEMU NVMe Scalability

IO

Guest OS

QEMU

IDE

NVMe

Expected

IO Latency (us)

# of threads

1 2 4 8 16 32 64
QEMU NVMe Scalability

Represent LightNVM’s QEMU

Guest OS

QEMU
QEMU Scalability

![Graph showing the scalability of QEMU](image-url)

- **Average Latency**

  - IO Latency (us)
  - # of threads

  - IDE
  - NVMe
  - Virtio
  - dp
QEMU Scalability

QEMU and existing emulators are NOT Scalable!
QEMU Scalability

QEMU and existing emulators are NOT Scalable!

FEMU is Scalable!
Scalability Root Causes & Solutions (I)

App
Guest OS
NVMe driver
Scalability Root Causes & Solutions (1)

Diagram:
- App
- Guest OS
- NVMe driver
- Submission Queue
Scalability Root Causes & Solutions (1)

- App
- Guest OS
- NVMe driver
- Submission Queue
- Tail DoorBell
- QEMU NVMe Emulation
Scalability Root Causes & Solutions (1)

- App
- Guest OS
- NVMe driver
- Submission Queue
- Tail DoorBell
- QEMU NVMe Emulation

VM-exit
Scalability Root Causes & Solutions (I)

- App
- Guest OS
- NVMe driver

Submission Queue

Tail DoorBell

QEMU NVMe Emulation

thousands of cycles interrupt **overhead**
Scalability Root Causes & Solutions (I)

1. **FEMU @ FAST '18**

2. **Guest OS**
   - **App**
   - **NVMe driver**
     - Submission Queue
     - Completion Queue
     - Tail Doorbell
     - Head Doorbell
     - QEMU NVMe Emulation

3. **VM-exit**

4. **thousands of cycles interrupt overhead**
Scalability Root Causes & Solutions (1)

QEMU NVMe Emulation

Tail DoorBell

Completion Queue

Submission Queue

thousands of cycles

interrupt overhead

Guest OS

NVMe driver

App

Guest OS

NVMe driver

App

Submission Queue

Completion Queue

Tail DoorBell

Head Doorbell

thousands of cycles

interrupt overhead
Scalability Root Causes & Solutions (I)

- Guest OS
- App
- NVMe driver

Submission Queue
Completion Queue

Tail DoorBell
Head DoorBell
QEMU NVMe Emulation

thousands of cycles interrupt overhead

Submission Queue
Scalability Root Causes & Solutions (I)

QEMU NVMe Emulation

Submission Queue

Completion Queue

Tail DoorBell

Head DoorBell

thousands of cycles interrupt overhead

VM-exit

Guest OS

App

NVMe driver

FEMU @ FAST '18
Scalability Root Causes & Solutions (I)

QEMU NVMe Emulation

Guest OS

App

NVMe driver

Tail DoorBell

Head Doorbell

Submission Queue

Completion Queue

thousands of cycles interrupt overhead

VM-exit

FEMU @ FAST '18

Submission Queue

Polling

Shadow DoorBell

QEMU NVMe Emulation

Queue

Completion
Scalability Root Causes & Solutions (1)

QEMU NVMe Emulation

Guest OS

App

NVMe driver

Submission Queue

Completion Queue

Tail DoorBell

Head DoorBell

thousands of cycles
interrupt overhead

QEMU NVMe Emulation

Guest OS

App

NVMe driver

Shadow DoorBell

Submission Queue

Completion Queue

polling

Tail DoorBell

Head DoorBell

QEMU NVMe Emulation
Scalability Root Causes & Solutions (1)

QEMU NVMe Emulation

Guest OS

App

NVMe driver

Submission Queue

Completion Queue

Tail DoorBell

Head DoorBell

thousands of cycles interrupt overhead

ZERO VM-exit

Submission Queue

Completion Queue

Shadow DoorBell

polling

QEMU NVMe Emulation

Guest OS

App

NVMe driver

Shadow DoorBell

Tail DoorBell

Head DoorBell
Scalability Root Causes & Solutions (2)

NVMe Emulation
Scalability Root Causes & Solutions (2)

- NVMe Emulation
- Block Driver
- Image Format Driver
- Raw Device Driver
Scalability Root Causes & Solutions (2)
Scalability Root Causes & Solutions (2)

- NVMe Emulation
- Block Driver
- Image Format Driver
- Raw Device Driver
- AIO Queue
- Thread Pool
Scalability Root Causes & Solutions (2)

- NVMe Emulation
- Block Driver
- Image Format Driver
- Raw Device Driver
- AIO Queue
- Thread Pool
- Host File System
Scalability Root Causes & Solutions (2)
Scalability Root Causes & Solutions (2)

- NVMe Emulation
  - Block Driver
    - Image Format Driver
      - Raw Device Driver
        - AIO Queue
        - Thread Pool
          - Host File System
            - Host Block IO Layer
              - Host Device Driver
Scalability Root Causes & Solutions (2)
Scalability Root Causes & Solutions (2)

- NVMe Emulation
- Block Driver
- DMA Emulation
- Image Format Driver
- Raw Device Driver
- AIO Queue
- Thread Pool
- Host File System
- Host Block IO Layer
- Host Device Driver

NVMe Emulation
Scalability Root Causes & Solutions (2)
Scalability Root Causes & Solutions (2)

- NVMe Emulation
- Block Driver
- DMA Emulation
- Image Format Driver
- Raw Device Driver
- AIO Queue
- Thread Pool
- Host File System
- Host Block IO Layer
- Host Device Driver

- DMA from/to heap storage
- NVMe Emulation
- FEMU Heap Storage
- DMA Emulation
Scalability Root Causes & Solutions (2)

- DMA Emulation
- NVMe Emulation
  - Block Driver
  - DMA Emulation
    - Image Format Driver
    - Raw Device Driver
    - AIO Queue
    - Thread Pool
    - Host File System
    - Host Block IO Layer
    - Host Device Driver

More than 20us latency reduction

DMA from/to heap storage
FEMU Accuracy

FEMU

OpenChannel-SSD
FEMU Accuracy

FEMU

OpenChannel-SSD
FEMU Accuracy

App

FEMU

OpenChannel-SSD
FEMU Accuracy

App

FEMU

$L_{femu}$

OpenChannel-SSD

$L_{oc}$
FEMU Accuracy

Error = \frac{|L_{femu} - L_{oc}|}{L_{oc}}
NAND \rightarrow \text{Data Register} \rightarrow \text{RAM}

\text{time}

T_R + T_{\text{transfer}}
\[ T_R + T_{\text{transfer}} \Rightarrow \text{queueing delay} \]
\[ T_R + T_{\text{transfer}} \rightarrow \]
Single-Register model (S-Reg)

NAND → Data Register → RAM → NAND → Data Register → RAM

$T_R + T_{transfer} + T_R + T_{transfer}$

queueing delay

time
Single-Register model (S-Reg)

\[ T_R + T_{\text{transfer}} \]

queueing delay

\[ T_R + T_{\text{transfer}} \]

OLTP

Error (%) vs. S-Reg

- Error: 20%
- S-Reg: 40
Single-Register model (S-Reg)

NAND $\rightarrow$ Data Register $\rightarrow$ RAM $\rightarrow$ NAND $\rightarrow$ Data Register $\rightarrow$ RAM

$T_R + T_{transfer} + T_R + T_{transfer}$

time

queueing delay

OLTP

Error (%)
Single-Register model (S-Reg)

![Diagram showing data flow between NAND, Data Register, RAM, and Queueing Delay]

\[ T_R + T_{\text{transfer}} + T_R + T_{\text{transfer}} \]

OLTP

Error (%) vs. S-Reg
Single-Register model (S-Reg)

NAND \rightarrow Data Register \rightarrow RAM \rightarrow NAND \rightarrow Data Register \rightarrow RAM

\[ T_R + T_{transfer} \rightarrow \text{queueing delay} \rightarrow T_R + T_{transfer} \]

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure.png}
\caption{Diagram showing the single-register model (S-Reg) with NAND, Data Register, RAM, and additional delays.}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{graph.png}
\caption{Graph showing OLTP and error percentage.}
\end{figure}
Single-Register model ($S$-Reg)

- NAND → Data Register
- Data Register → RAM
- RAM → NAND
- NAND → Data Register
- Data Register → RAM

Time:
- $T_R$ + $T_{\text{transfer}}$
- Queueing delay
- $T_R$ + $T_{\text{transfer}}$

Faster

Graph:
- OLTP
- Error (%)
- S-Reg
Single-Register model (\textit{S-Reg})

\begin{align*}
\text{NAND} & \xrightarrow{\text{Req1}} \text{Data Register} \xrightarrow{\text{Req1}} \text{RAM} \xrightarrow{\text{Req2}} \text{NAND} \xrightarrow{\text{Req2}} \text{Data Register} \xrightarrow{\text{Req2}} \text{RAM}
\end{align*}

\[ T_R + T_{\text{transfer}} + T_R + T_{\text{transfer}} \]

Queueing delay

Double-Register model (\textit{D-Reg})

\begin{align*}
\text{NAND} & \xrightarrow{\text{Req1}} \text{Data Cache Register} \xrightarrow{\text{Req1}} \text{RAM} \xrightarrow{\text{Req2}} \text{NAND} \xrightarrow{\text{Req2}} \text{Data Cache Register} \xrightarrow{\text{Req2}} \text{RAM}
\end{align*}

\[ \text{faster} \]

\[ \text{OLTP} \]

\begin{axis}[
width=8.4cm,
height=6cm,
xtick=data,
symbolic y coords={0,20,40},

\addplot[fill=gray!30] coordinates {
(1,0)
(2,40)
};

\end{axis}
Single-Register model (S-Reg)

Double-Register model (D-Reg)

- **NAND** → **Data Register** → **RAM**
- **NAND** → **Data Register** → **RAM**

\[ T_R + T_{\text{transfer}} \]

Queueing delay

\[ T_R + T_{\text{transfer}} \]

 Faster

**OLTP**

<table>
<thead>
<tr>
<th>D-Reg</th>
<th>S-Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>40</td>
<td>60</td>
</tr>
</tbody>
</table>
FEMU Accuracy

Filebench

<table>
<thead>
<tr>
<th>Application</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Server</td>
<td>40</td>
</tr>
<tr>
<td>Network FS</td>
<td>20</td>
</tr>
<tr>
<td>OLTP</td>
<td>60</td>
</tr>
<tr>
<td>Varmail Video Server</td>
<td>10</td>
</tr>
<tr>
<td>Web Proxy</td>
<td>30</td>
</tr>
</tbody>
</table>

Single Register Model
**FEMU Accuracy**

**Latency Error:** 11-57%

Single Register Model (**S-Reg**)
FEMU Accuracy

Latency Error: 11-57%

Single Register Model (S-Reg)
FEMU Accuracy

Latency Error: 11-57% => 0.5-38%

Single Register Model (\textit{S-Reg})  Double Register Model (\textit{D-Reg})
FEMU Accuracy

Latency Error: 11-57% \Rightarrow 0.5-38%
FEMU Accuracy

Latency Error: 11-57% \Rightarrow 0.5-38%

- Single Register Model (S-Reg)
- Double Register Model (D-Reg)

Filebench

Latency (ms)

OpenChannel-SSD

FEMU

Similar!

X: # of channels
Y: # of planes per channel
FEMU Limitations

- Further optimizations to support higher parallelism (more scalable)
- Accuracy can be improved
- Not able to emulate large-capacity SSD
- No persistence
Conclusion

- Cheap
- Accurate
- Scalable
- Extensible

https://github.com/ucare-uchicago/femu
Conclusion

Installing and using FEMU can cause side effects including headache, nausea, agitation, and depression. If your research condition does not improve after using FEMU for a week, please talk to us, your advisor, or your doctor immediately.

- Cheap
- Accurate
- Scalable
- Extensible

https://github.com/ucare-uchicago/femu
Thank you!

Questions?

FEMU: https://github.com/ucare-uchicago/femu

Huaicheng Li
huaicheng@cs.uchicago.edu

http://ucare.cs.uchicago.edu