Efficient file sharing between host and unikernel

DIPLOMA THESIS

Fotis Xenakis

Supervisor: Nectarios Koziris

30/10/2020

Cloud computing & virtualization

In the beginning there was the **cloud**:

- Utility computing
- Everything as a Service
- Large scale

Relying on **virtualization**:

- Elasticity for the provider
- Isolation for the user

Operating systems

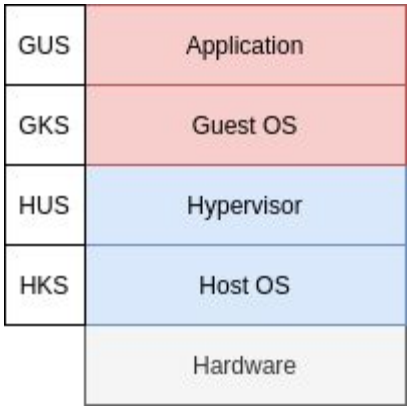
General-purpose guest OS:

- Imposes isolation

In practice:

- Only runs a **single application**

Could be more **efficient**. Bloat!



Unikernels

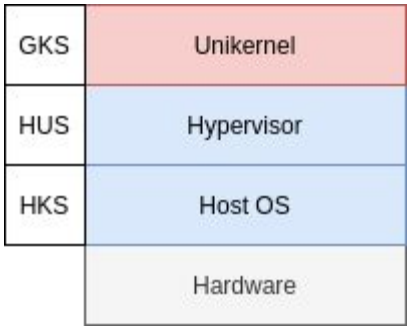
“OS” + application:

- Library OS
- Single address space

Thus:

- **Efficiency** (size, memory, time)
- Performance(?)

OSv: Linux compatibility, many features, many(!) file systems



File systems & virtualization

Traditionally:

- Guest: **disk** (block device)
- Host: **file** (opaque)
- **Indirection**

Access to the host fs:

- High level
- **Easier** for the guest
- Access from the host **too**

Virtio-fs

Until now:

- NFS, SMB, VirtFS...
- **Network**

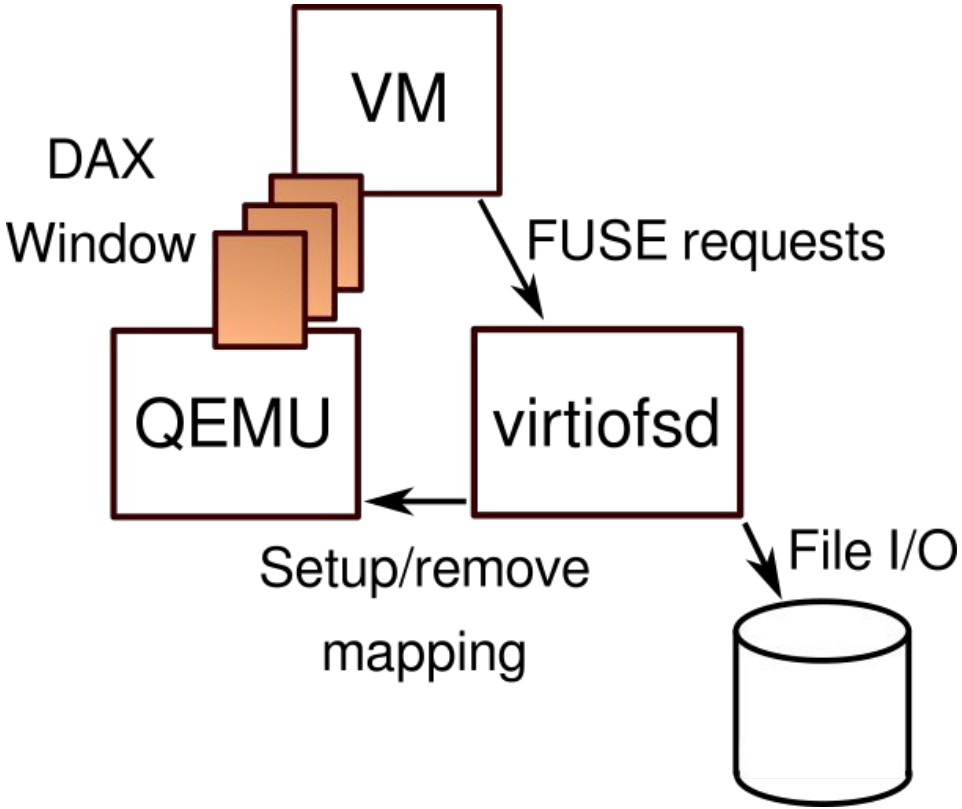
virtio-fs:

- Virtualization-native
- **virtio, FUSE**
- **virtiofsd**

Virtio-fs DAX window

Shared guest⇔host memory:

- **mmap()**
- No VMEXIT
- Better performance



Thesis goals

Technical:

- virtio-fs **DAX** window in **OSv**
- **Read-only**
- Boot from virtio-fs

...or not:

- **Contribution** to FOSS project
- Practical work ⇒ practical value

Virtio-fs in OSv

Pre-existing, elementary implementation

Driver:

- Low-level PCI device handling
- **Agnostic** to FUSE

File system:

- High-level fs operations
- Knows **FUSE**

DAX window in OSv

Driver:

- PCI

File system:

- map alignment
- FUSE_SETUPMAPPING
- FUSE_REMOVEMAPPING

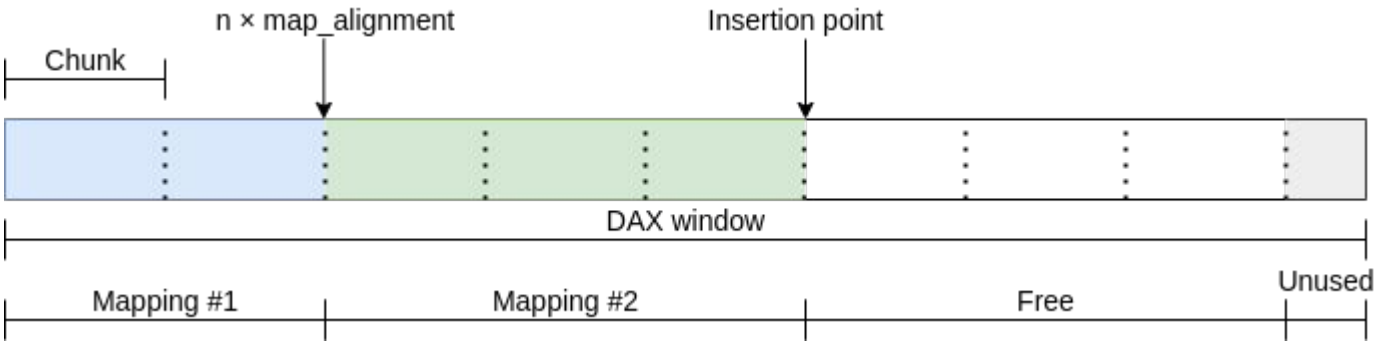
DAX window manager

Problem:

- Multiple mappings

Solution:

- Chunks
- Stack (LIFO)



Boot from virtio-fs

Initially:

- **Boot** fs: always ramfs
- **Root** fs: dynamically rofs ⇒ ZFS ⇒ ramfs

Eventually:

- **Optional** kernel command line option
- Root fs: rofs, ZFS, **virtio-fs**, ramfs
- Backwards compatible

Changes to kernel and tools

Evaluation

Various scenarios:

- **Microbenchmark**
- **Startup** time
- Real-world **application**

Microbenchmark

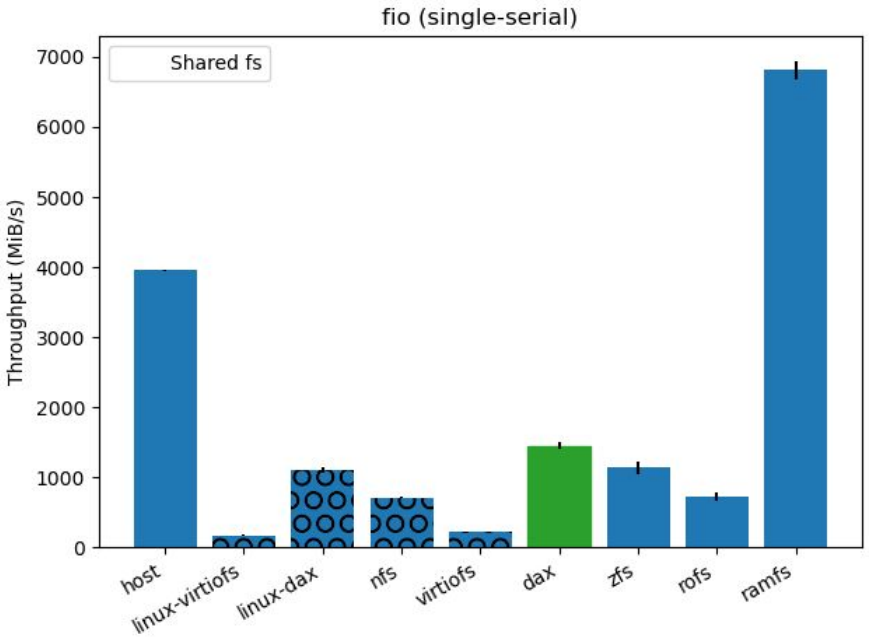
Flexible I/O tester (fio)

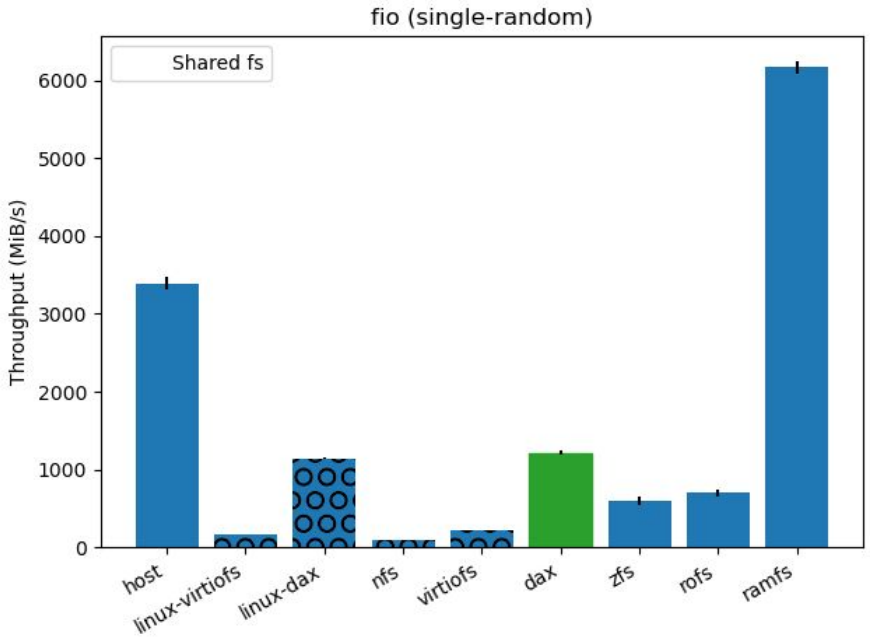
- 1 file / multiple files
- Total size 1 GiB
- Serial / random reading
- **Throughput**

Comparison:

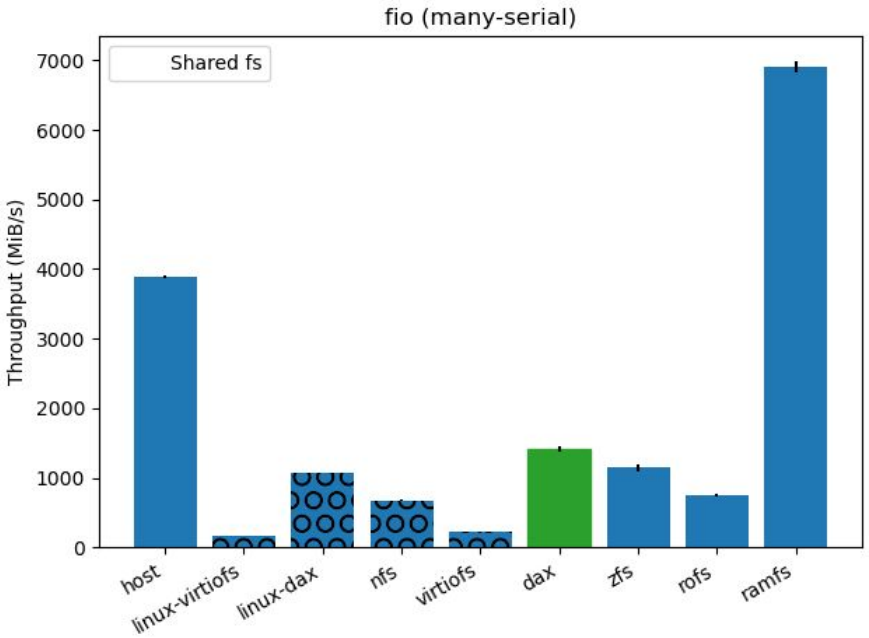
- Host (baseline)
- Linux: virtio-fs, virtio-fs-DAX
- OSv: virtio-fs, virtio-fs-DAX, ZFS, rofs, ramfs, NFS

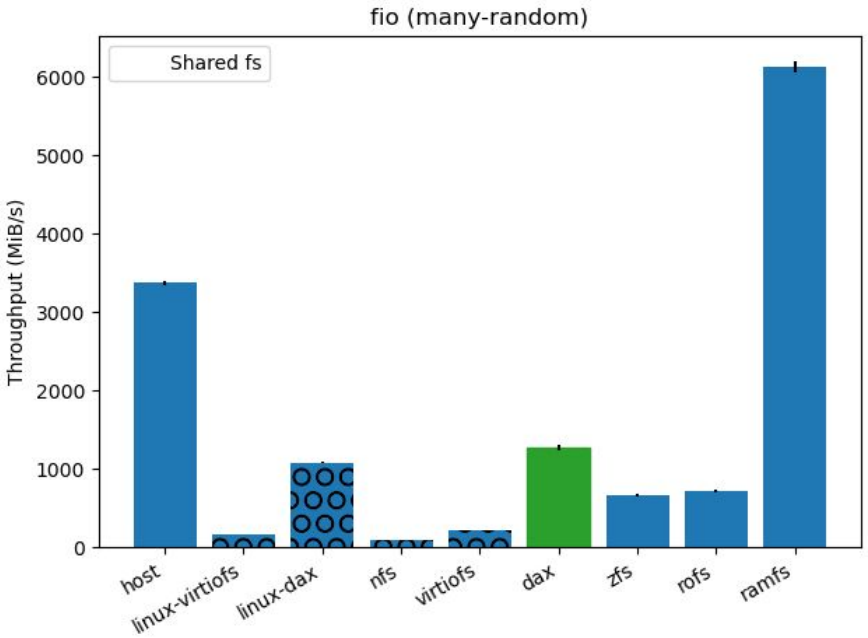
Microbenchmark results (single file)





Microbenchmark results (multiple files)





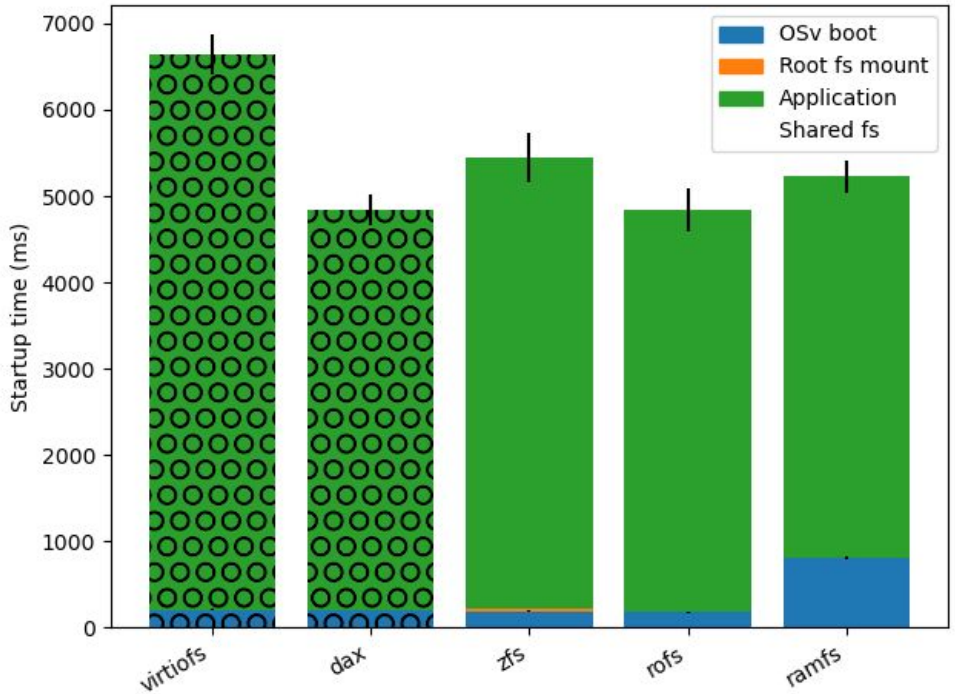
Startup time

Simple **web** (spring) **application**

- OSv boot
- Root fs mount
- Application initialization

Comparison:

- OSv: virtio-fs, virtio-fs-DAX, ZFS, rofs, ramfs



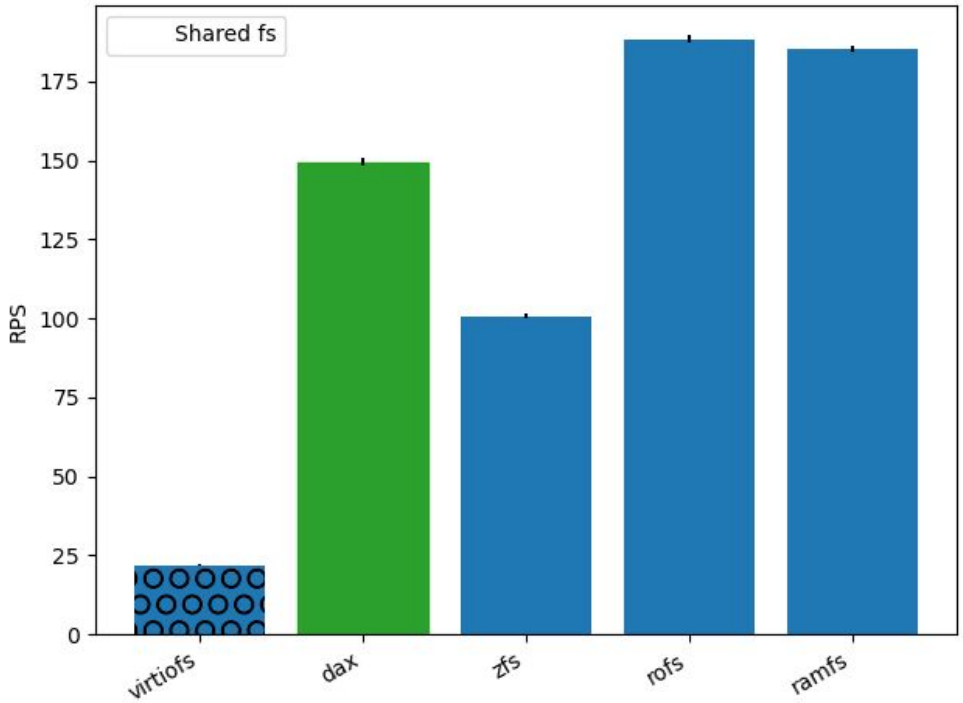
Application benchmark

Nginx static **web server**:

- Many files ~MiB
- **Vegeta HTTP** load client
- **Requests / second**

Comparison:

- OSv: virtio-fs, virtio-fs-DAX, ZFS, rofs, ramfs



Summary

Technical conclusions:

- virtio-fs in OSv **useful** (changes without rebuilding)
- DAX window ⇒ excellent **performance**
- Pleasant **development** on OSv (like application development)

Non-technical conclusions:

- Open, supportive **communities**
- Contribution

Contribution


```
You are welcome to write a guest post on the QEMU blog
(https://qemu.org/blog/) that gives an overview of OSv and the virtio-fs
device interface. I imagine people would be interested in learning about
both these topics.

The git repo for the QEMU website and blog is here:
https://gitlab.com/qemu-project/qemu-web

You can create a new blog post by dropping a Markdown file into _posts/.
Images can be added to screenshots/ directory.

The blog post can be sent to the QEMU mailing list using
git-send-email(1) with Thomas Huth <thuth redhat com> CCed. Please also
CC virtio-fs redhat com and we'll review it.

Stefan
```

Thank you!

Questions?