OpenCSD, simple and intuitive computational storage emulation with QEMU and eBPF

Corné Lukken & Animesh Trivedi

fosdem@dantalion.nl

https://github.com/Dantali0n/OpenCSD
Who am I

Corne Lukken - Dantali0n
Parallel & Distributed Systems
Academia
  Bachelor Software Engineering (AUAS)
  Master Computer Science (VU / UvA)
Experience
  Health Technology @ AUAS
  Openstack @ CERN
  Computational Storage @ VU
  SCADA @ ASTRON
Why do we need it

200 Zettabytes
Why do we need it

- Von Neumann architecture
- Memory bottlenecks
- Interconnect bottlenecks
- Energy efficiency
Why do we need it

4.5x bandwidth gap with 64 SSD storage server (2021)

Solution: prevent moving data from flash SSD to host

---

What is Computational Storage

- Fit compute & memory on storage device
- User submits programs to run on device (*kernels*)
- Only return results from user programs
- Reduce data movement
- Improve energy efficiency
- Specialized hardware (ASIC / FPGA)
State of Current Prototypes (September 2022)

Impediments:

1. API between host and device
2. Keep filesystem synchronized
3. Stick to existing interfaces

Shortlist:

BlockNDP (2020) ²
Metal FS (2020) ³
INSIDER (July 2019) ⁴

² https://doi.org/10.1145/3429357.3430519
³ https://doi.org/10.1145/3342195.3387557
⁴ https://www.usenix.org/conference/atc19/presentation/ruan
OpenCSD & FluffleFS

- Entirely runs in user-space
- Using pre-existing system calls
- Concurrent access to same file
- Built using existing open-source libraries
- Use and experiment without any additional hardware
The design and implementation of a Log-structured File System
Design: Zoned NameSpaces (ZNS) 1/2

Conventional SSDs

- The (traditional) block interface
- Linear writes
- Large erase units
- Flash Translation Layer (FTL)
- Logical vs Physical Block Address (LBA / PBA)
Design: Zoned NameSpaces (ZNS) ²

Zoned Namespaces SSDs

- Fit interface to nand flash behavior
- Divide erase units in zones
- Require each zone is linearly written
- Perfect match for LFS filesystems
- Host operating system manages FTL

---

Design: ZNS + LFS

Synchronizing host & device filesystem
  Ensure file immutable for kernels
  No host communication during kernel execution
  Unblock regular access
  Check & understand kernel behavior

Snapshot consistency model
Design: Architecture Independent Kernels

Define system calls in ABI header
Leave implementation to VM (vendor)
Compile once use everywhere
eBPF ISA trivial to implement in VM
Pre-existing FOSS eBPF VMs (uBPF)
Shannon Entropy Demo - Background

Quantify randomness, distribution of possible values

High value, very random, typically between 0/1
Normalize for bytes
Used in background / filesystem compression
Store count in 256 bins (array), histogram

\[ H(x) = - \sum_{i=1}^{n} p(x_i) \log_b p(x_i) \]
eBPF very small stack size
Get heap pointer, manually offset
System calls provided by eBPF
Helper functions & data structures provided by filesystem
Can we make user programs agnostic to filesystem?

```c
void *buffer;
bpf_get_mem_info(&buffer, &buffer_size);

uint32_t *bins = (uint32_t*)(buffer + sector_size);
for(uint16_t i = 0; i < 256; i++) {
    bins[i] = 0;
}

while(buffer_offset < data_limit) {
    lba_to_position(*cur_data_lba, zone_size, &zone, &sector);
    bpf_read(zone, sector, 0, sector_size, buffer);

    for(uint64_t j = 0; j < bytes_per_it; j++) {
        bins[*((byte_buf + j))] += 1;
    }

    buffer_offset = buffer_offset + sector_size;
    next_data_lba(&cur_data_lba);
}

bpf_return_data(bins, sizeof(uint32_t) * 256)
```
Operating Principle Details

Step by step system calls
Stream vs Event kernels
When to take snapshots?
  Upon `setxattr`
Isolate by filehandle, pid, inode?
  By pid
Shannon Entropy Demo - Execution

How to submit I/O requests to execute kernels?

Stride requests (FUSE page limit)

Inode from kernel file as value for extended attribute Key

Different keys for different types of offloading

Return data from `os.pread` less than request size.
Limitations and Considerations

Filesystem (FluffleFS) is solely proof of concept!
eBPF endian conversions and datastructure layouts
No floating point in eBPF
Kernel performance not representative of microcontrollers
Only, read stream kernel fully supported
Problematic event kernel performance
Filesystem agnostic kernels
Further Reading

Try it today! https://github.com/Dantali0n/OpenCSD

OpenCSD: Log-Structured Filesystem enabled Computational Storage Device platform - https://tinyurl.com/opencsd-thesis
