CS 744: SPLIT-FS

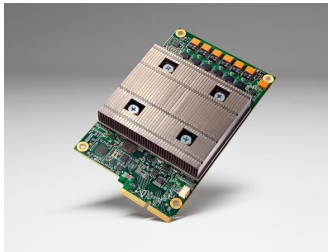
Shivaram Venkataraman

Fall 2021

ADMINISTRIVIA

- Course Project: Check in: Today!
- Midterm 2 next week!

Serverless Computing



Compute Accelerators

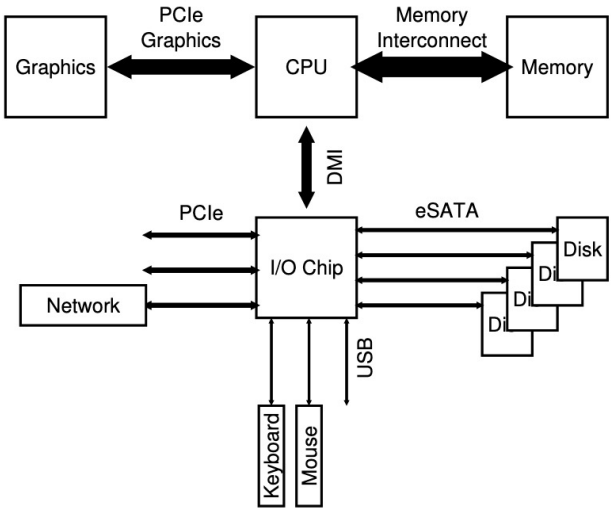
Infiniband Networks

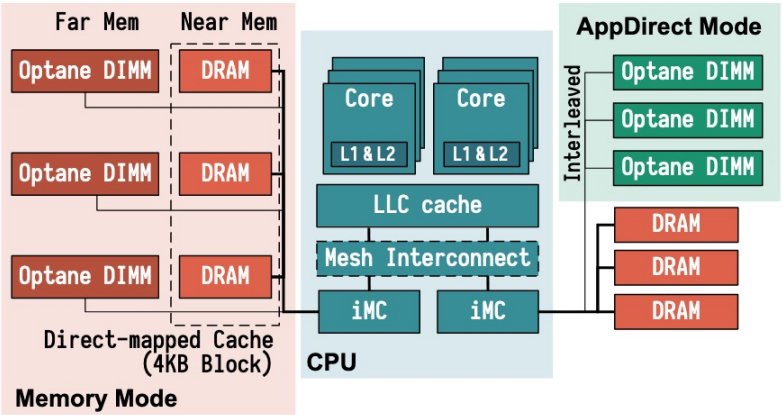
Non-Volatile Memory

PERSISTENT MEMORY

Property	DRAM	Intel PM
Sequential read latency (ns)	81	169 (2.08×)
Random read latency (ns)	81	305 (3.76×)
Store + flush + fence (ns)	86	91 (1.05×)
Read bandwidth (GB/s)	120	39.4 (0.33×)
Write bandwidth (GB/s)	80	13.9 (0.17×)

WHAT IS DIFFERENT?





(a) Optane Platform Modes (Memory and AppDirect)

BACKGROUND: FILE SYSTEM API

```
int fd = open(char *path, int flag, mode_t mode)
read(int fd, void *buf, size_t nbyte)
write(int fd, void *buf, size_t nbyte)
close(int fd)

rename(char *old, char *new)

fsync(int fd)
```

MOTIVATION: OVERHEADS

DAX: mmap pages into virtual memory.

No page caches!

File system	Append Time (*ns*)	Overhead (*ns*)	Overhead (%)
ext4 DAX	9002	8331	1241%
PMFS	4150	3479	518%
NOVA-Strict	3021	2350	350%
SplitFS-Strict	1251	580	86%
SplitFS-POSIX	1160	488	73%

SPLIT-FS: GOALS

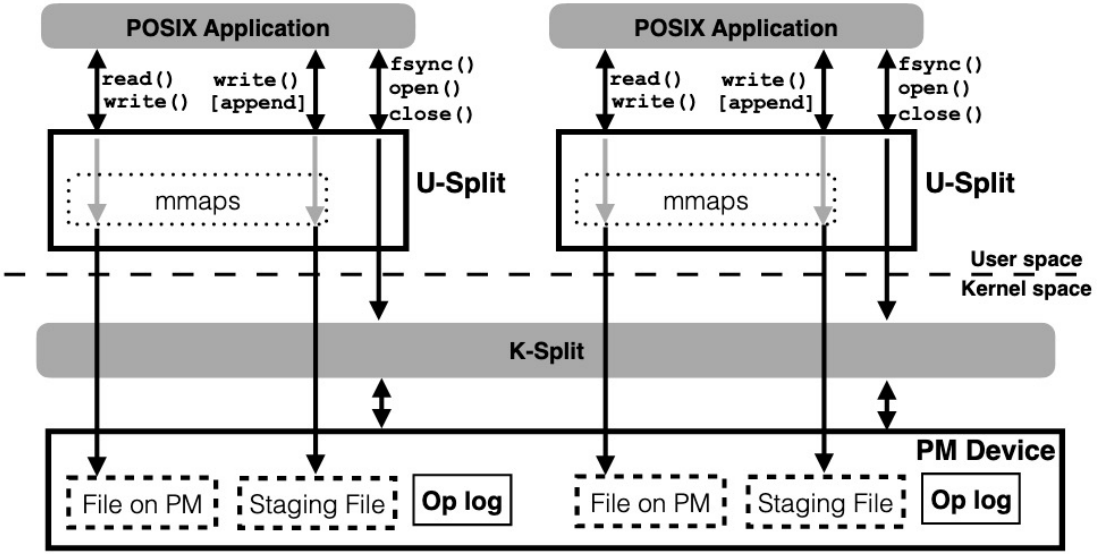
Low software overhead

Transparency

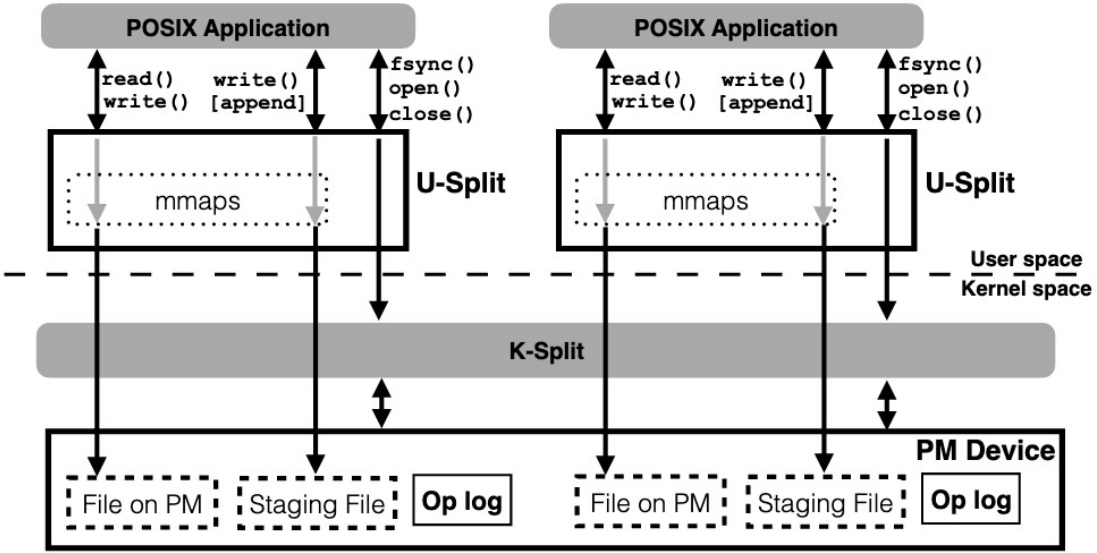
Minimal data-copy/IO

Flexible semantics

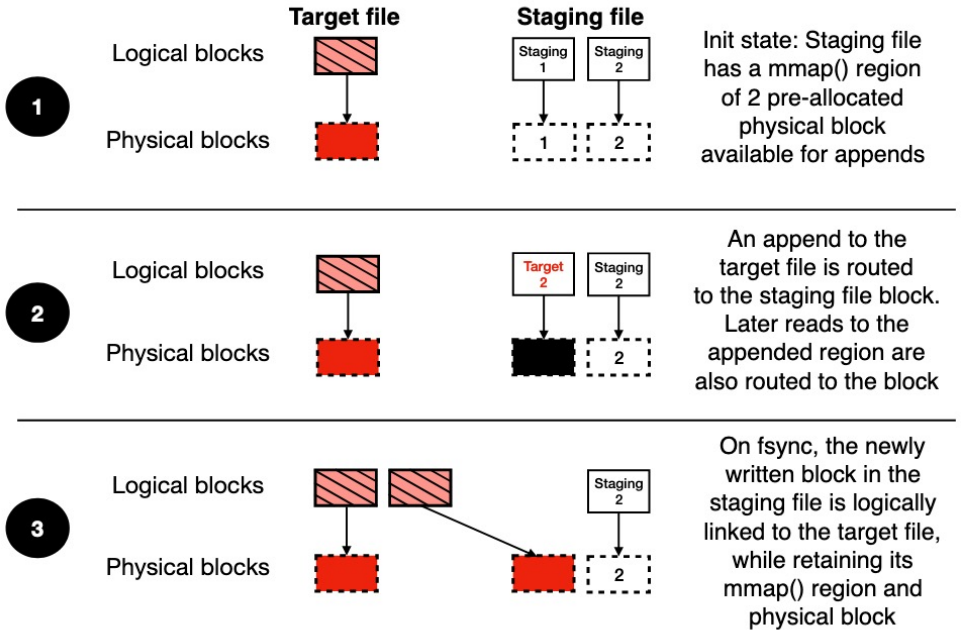
SPLIT FS DESIGN: READ/WRITES



SPLIT FS DESIGN: APPEND



RELINK


Target file
Staging file
1
Logical blocks
Physical blocks
Staging 1
Staging 2
1
2
Init state: Staging file has a mmap() region of 2 pre-allocated physical block available for appends
2
Logical blocks
Physical blocks
Target 2
Staging 2
2
An append to the target file is routed to the staging file block. Later reads to the appended region are also routed to the block
3
Logical blocks
Physical blocks
Staging 2
2
On fsync, the newly written block in the staging file is logically linked to the target file, while retaining its mmap() region and physical block

SPLIT-FS MODES

Mode	*Sync. Data Ops*	*Atomic Data Ops*	*Sync. Metadata Ops*	*Atomic Metadata Ops*	*Equivalent to*
POSIX	✗	✗	✗	✓	ext4-DAX
sync	✓	✗	✓	✓	Nova-Relaxed, PMFS
strict	✓	✓	✓	✓	NOVA-Strict, Strata

SPLIT-FS: LOGGING

Logical redo logging

- Log entry: 64B in size! 4B checksum!
- sfence to ensure ordering

Fixed length log: 128 MB per-application

Replay entire log on recovery!

SUMMARY

Persistent Memory: New opportunities, new challenges

Split-FS: split Pipelining to use CPU, GPU

Partition buffer, BETA ordering

DISCUSSION

https://forms.gle/8TwGgqXhVyuiRCpx8

System call	Strict	Sync	POSIX	ext4 DAX
open	2.09	2.08	1.82	1.54
close	0.78	0.69	0.69	0.34
append	3.14	3.09	2.84	11.05
fsync	6.85	6.80	6.80	28.98
read	4.57	4.53	4.53	5.04
unlink	14.60	13.56	14.33	8.60

Table 6: SplitFS system call overheads. The table compares the latency (in us) of different system calls for various modes of SplitFS and ext4 DAX.

In what ways can SplitFS improve performance of Big Data frameworks like MR/Spark?

NEXT STEPS

Next class: TPU

Project check-ins tonight!

DISCUSSION

Staging files in DRAM?

Page faults are expensive on open()