GRFS

A new I/O subsystem for modern data-intensive workloads

Aldo Tali

Advisor: Charalampos Mainas

Chair of Decentralized Systems Engineering

https://dse.in.tum.de/

22.04.2022 – 21.10.2022

Motivation: Research context

- Data intensive workloads
  - Modern (ML/AI)
  - Traditional (databases)

- Performance Bottleneck
  - move big volumes
  - CPU-centric architecture

- Near Data Processing (NDP)

State-of-the-art

- Recent developments in market ready low-power processing units provide stronger computational power
  - SoCs, FPGAs, ASICs, … [1,2,3]

- NDP: advantages provide promising foundation for an interface of generic workloads
  - project invariant
  - practical solution

[1] J. Kwak, et al. "Cosmos+ OpenSSD: Rapid Prototype for Flash Storage Systems"
[2] "NGD Systems heralds in-situ processing for NVMe SSDs"
[3] Xilinx. "Samsung SmartSSD"

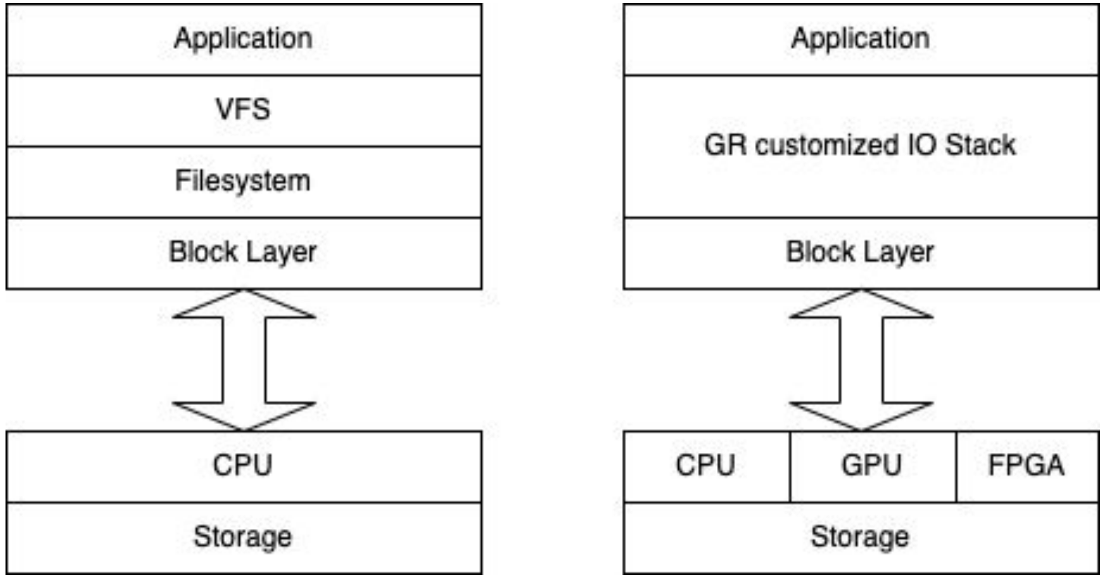
Research gap

- Prior solutions are application centric
  - Limited performance or flexibility
  - High programming effort
  - Lack of crucial system support

- NDP devices will equip cloud servers
  - Heterogeneity constraint
  - Multi-tenant constraint

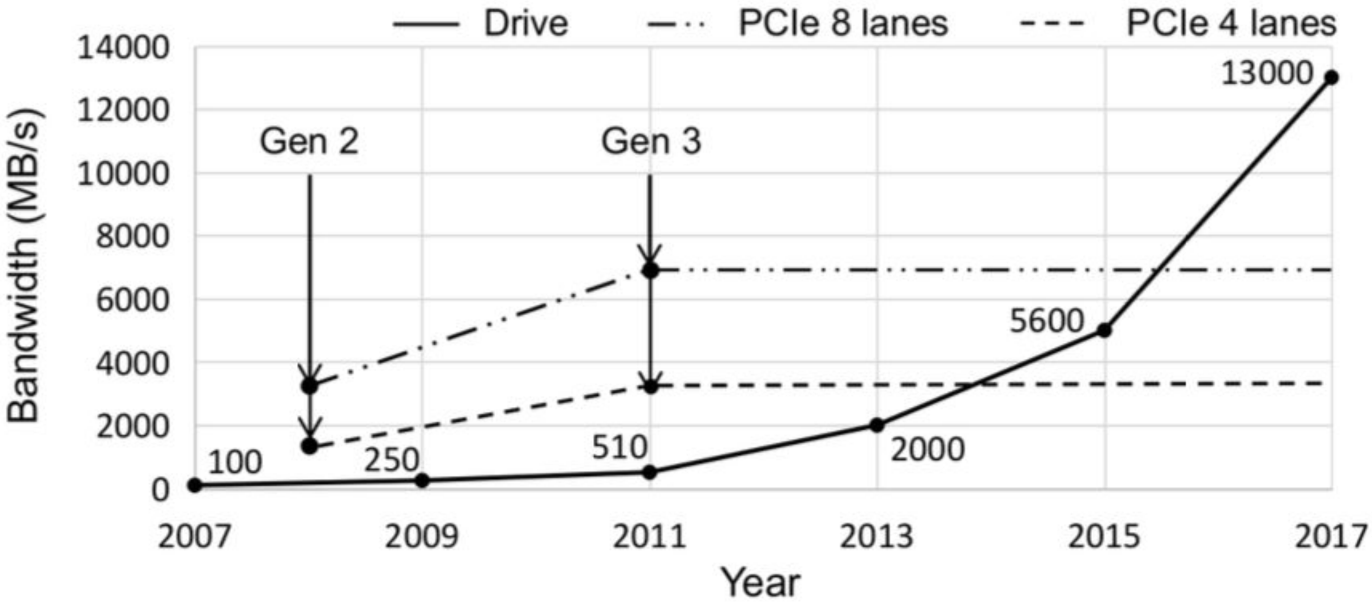
GRFS

- A new I/O subsystem, which can efficiently manage and provide a generic-enough interface for user applications



Background

- Data Volumes are steadily increasing, compute is not
- Computational Storage bandwidth is advancing



[1] Figure Taken from: Z. Ruan, et al. "INSIDER: Designing In-Storage Computing System for Emerging High-Performance Drive"

I/O subsystem: A new approach to modern data-intensive workloads

Virtual File System is bypassed to offload operations as direct calls to NVMe.
A POSIX compliant approach to abstract the use of the NDP units.

System design goals:

- Efficiency
- System Abstraction
- System Support

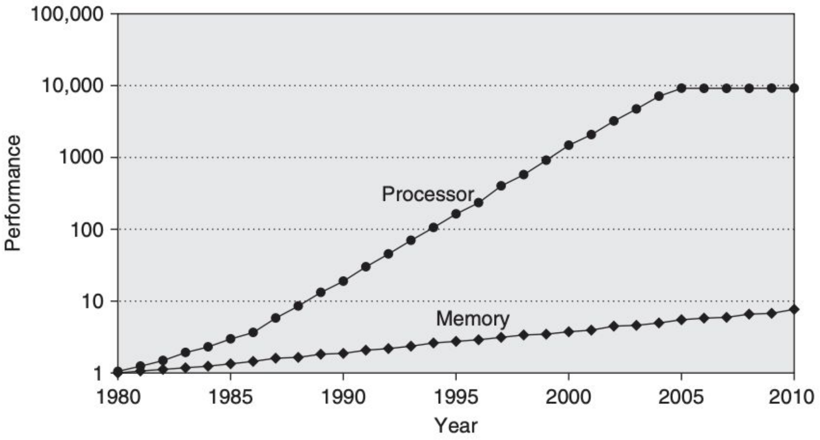
Outline

- ~~Motivation~~
- Background
  - State of Data Intensive Workloads
  - State of System on Chip
  - Accelerators in ML/DL
- Design
- Implementation
- Evaluation

State of Data intensive Workloads

- Data Operations on increased volumes requires data to move
  - CPU handles **operations**
  - Memory and storage handle **volumes**

- Evaluate performance based on the 4 V's
  - Volume, Velocity, Variety, Veracity



[1] E. Mollick, "Establishing Moore's Law," in IEEE Annals of the History of Computing, vol. 28, no. 3, pp. 62-75,
[2] J. Henessy et.al "Computer Architecture: A Quantitative Approach" pp 73

State of System on Chip

- Compute in Storage
- Decrease Data Transfers
  - Remove network dependency, consume less energy
- Adoption suffers
  - Lack of interface, security, usability, only stateless operations

Accelerators in ML/DL

- Identify independent chunks

- Offset them to NDP

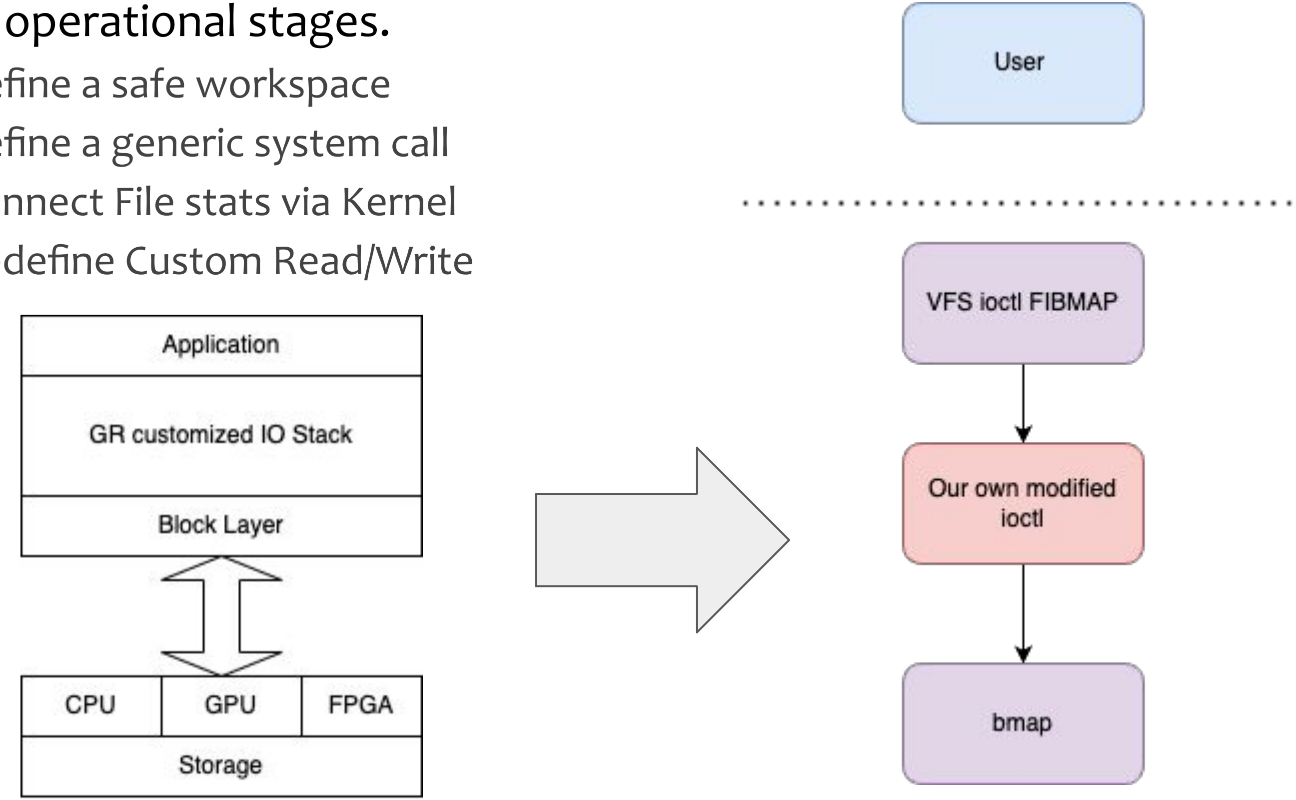
```
SELECT   avg(metric1),
         max(metric2),
         sum(metric3)
FROM     my_table;
```

Outline

- ~~Motivation~~
- ~~Background~~
- Design
    - The proposed subsystem
- Implementation
- Evaluation

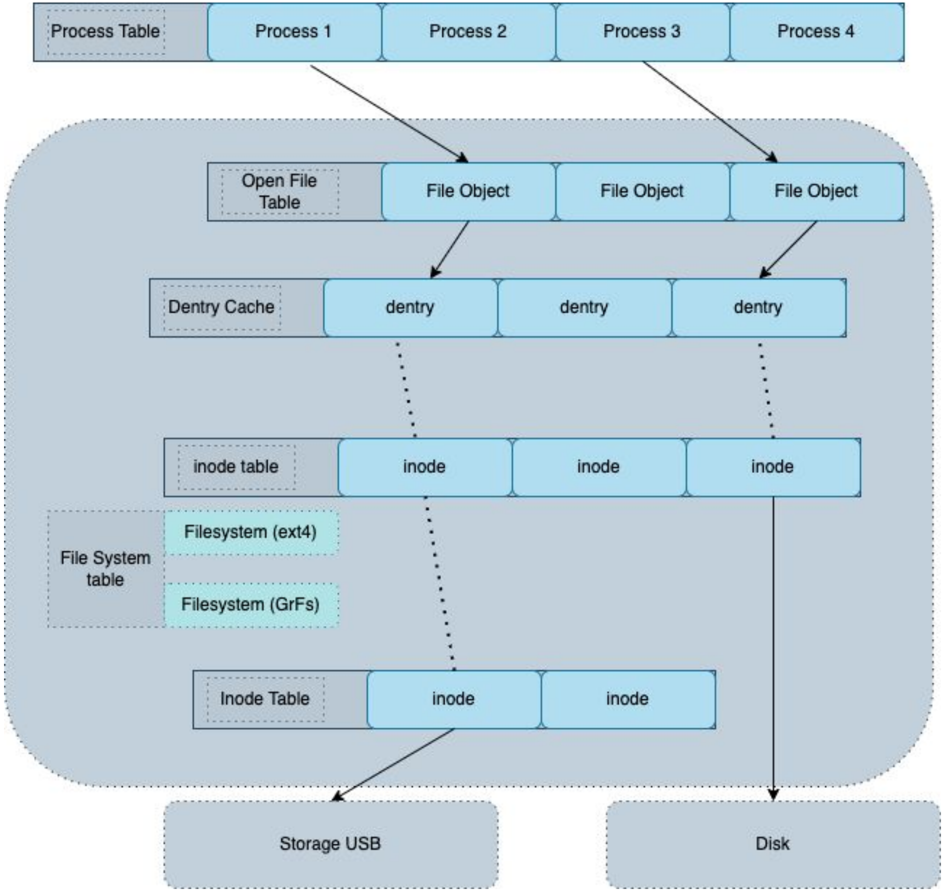
System overview

- 4 main operational stages.
  - Define a safe workspace
  - Define a generic system call
  - Connect File stats via Kernel
  - Redefine Custom Read/Write



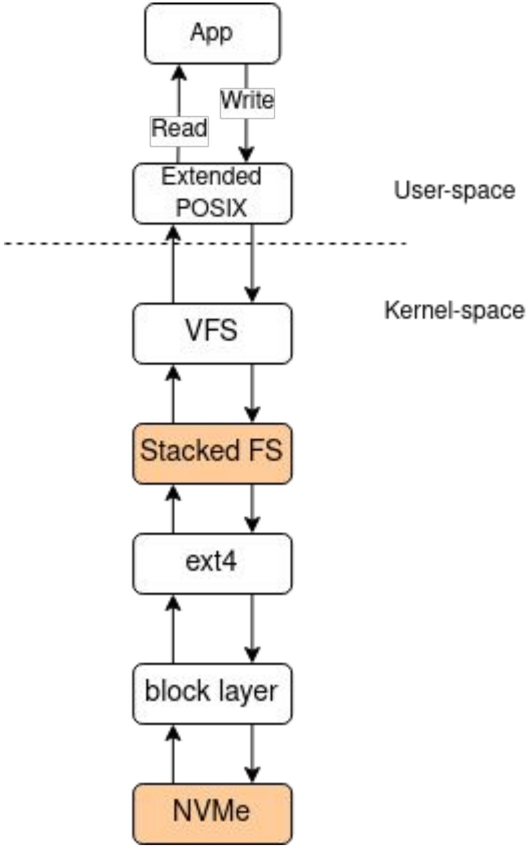
Design overview

GRFS maps **dentry, inode, file object** and **super operations** seamlessly into the host fs via a POSIX compliant integration.



The architectural design

- Override VFS calls
- Operate on custom VFS operations
- Skip the orchestrator overhead



Outline

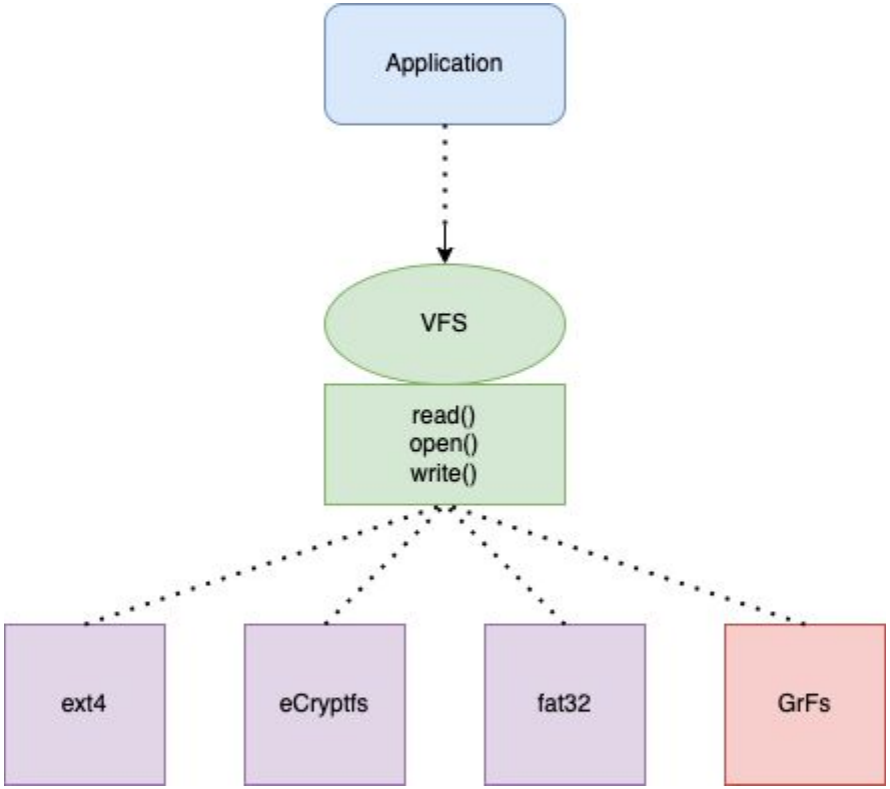
- ~~Motivation~~
- ~~Background~~
- ~~Design~~
- Implementation
- Evaluation

Implementation

GRFS is build on C by overriding VFS POSIX compliant read/write operations.

- VFS overrides use FIBMAP and bmap
- User - Kernel switch done via FS segment – intact!
- Stacked File System Model
- Uses guidelines from SimpleFS and ecryptFs

GrFS workflow



Outline

- ~~Motivation~~
- ~~Background~~
- ~~Design~~
- ~~Implementation~~
- Evaluation

Evaluation

- What are integration overheads of GrFs?
  - Local to read/write operations, incomplete VFS support

- How robust is GRFS in integrating in different NDP's?
  - Custom VFS implementation revision is required

Summary

Current Von Neumann architecture is not optimized for data intensive workloads

- frequent data movements on compute-storage
- file system overheads
- heterogeneity of accelerations not interfaced

GrFS:

- abstracts read/write operations
- integrates seamlessly in kernel
- bypasses VFS