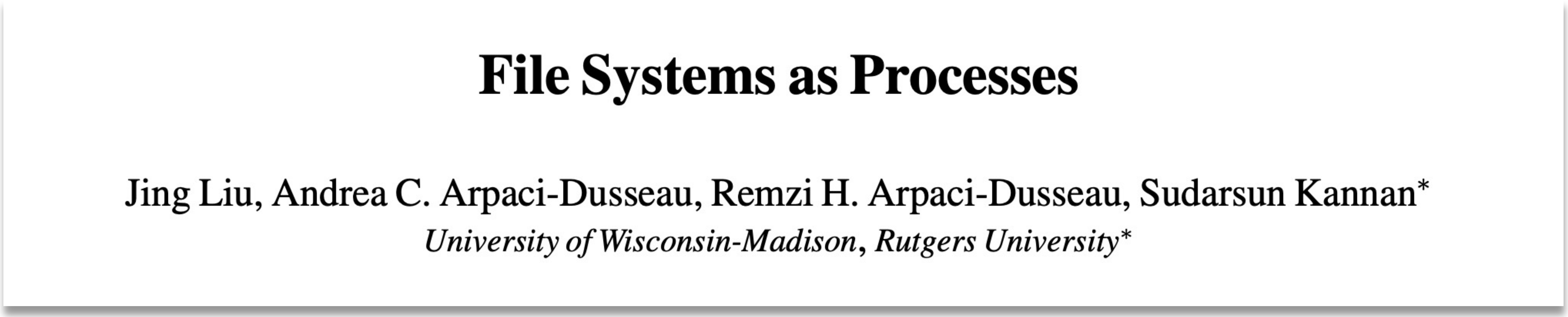


File Systems as Processes

Jing Liu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Sudarsun Kannan*
*University of Wisconsin-Madison, Rutgers University**

File Systems as Processes (FSP)

ATLAS KAAN YILMAZ

Main Problems

Main Problems

- Faster access latency on newer generations of SSDs

Main Problems

- Faster access latency on newer generations of SSDs
- The traditional FS design hinders performance gain

Main Problems

- Faster access latency on newer generations of SSDs
- The traditional FS design hinders performance gain
- Kernel trap overhead is a dominant cost

Main Problems

- Faster access latency on newer generations of SSDs
- The traditional FS design hinders performance gain
- Kernel trap overhead is a dominant cost
- FSP builds a direct-access FS as a user process

Advantages of FSP Architecture

Advantages of FSP Architecture

- Developer velocity

Advantages of FSP Architecture

- Developer velocity
- Ensure integrity, concurrency, consistency as trusted computing

Advantages of FSP Architecture

- Developer velocity
- Ensure integrity, concurrency, consistency as trusted computing
- Easier cluster management

Advantages of FSP Architecture

- Developer velocity
- Ensure integrity, concurrency, consistency as trusted computing
- Easier cluster management
- FSP delivers high performance

FSP Architecture

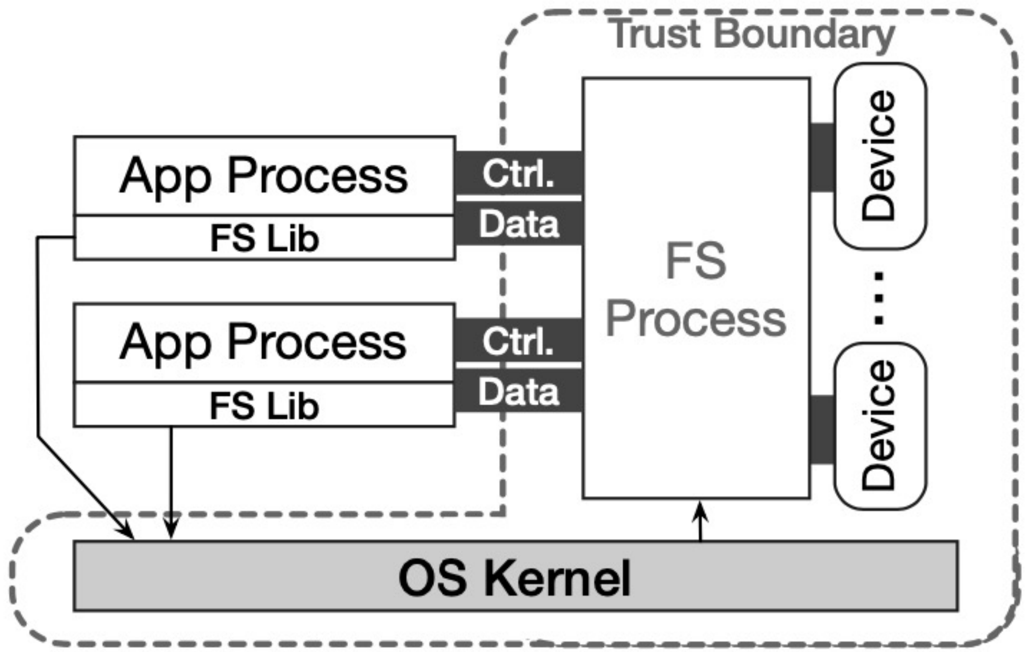
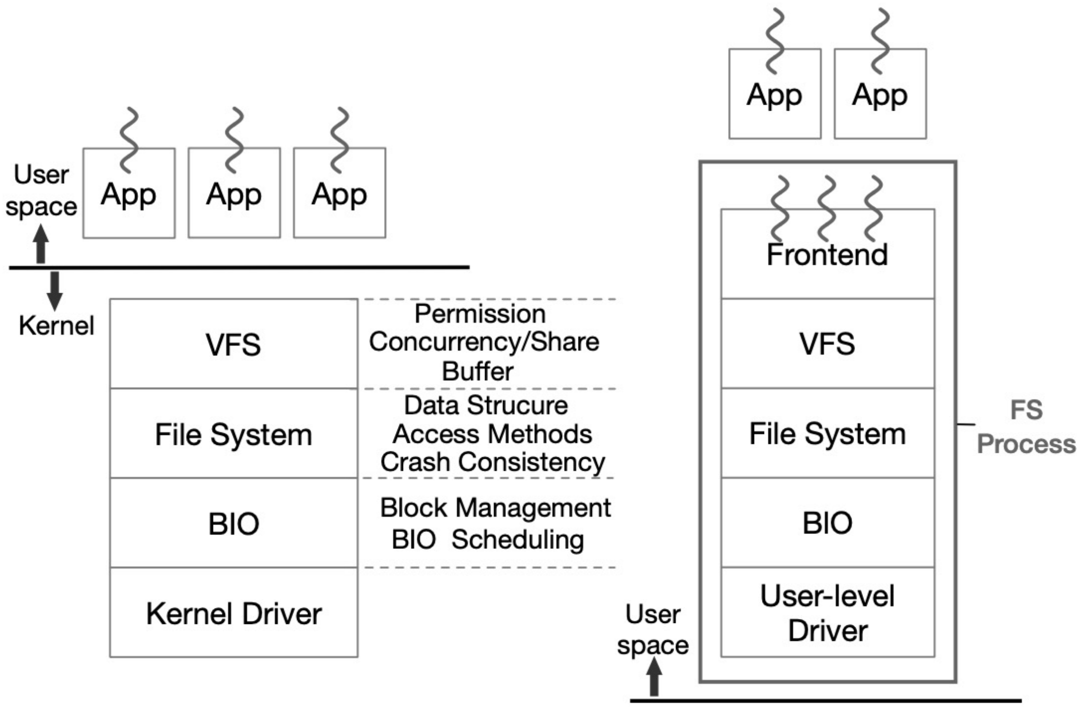


Figure 1: **File Systems as Processes Architecture**

In-Kernel VS FSP



(a) Kernel Storage Stack (b) Process Storage Stack

Figure 2: **Anatomy of the Storage Stack**

Challenges of FSP

Challenges of FSP

- Efficient Communication – IPC, multi-cores

Challenges of FSP

- Efficient Communication – IPC, multi-cores
- Frontend Thread Model – management, locks, request collection

Challenges of FSP

- Efficient Communication – IPC, multi-cores
- Frontend Thread Model – management, locks, request collection
- Process to IO Connection – TCB, secure comm, clean-up, forks

Challenges of FSP

- Efficient Communication – IPC, multi-cores
- Frontend Thread Model – management, locks, request collection
- Process to IO Connection – TCB, secure comm, clean-up, forks
- Handling Requests – interrupts, polling, buffers, scheduling

Challenges of FSP

- Efficient Communication – IPC, multi-cores
- Frontend Thread Model – management, locks, request collection
- Process to IO Connection – TCB, secure comm, clean-up, forks
- Handling Requests – interrupts, polling, buffers, scheduling
- Legacy Design – modern devices, multi-layer arch, limiting defects

DashFS Prototype Results

DashFS Prototype Results

- Faster write, read, direct access

DashFS Prototype Results

- Faster write, read, direct access
- Sub-microsecond latency on IPC

DashFS Prototype Results

- Faster write, read, direct access
- Sub-microsecond latency on IPC
- Comm channel scales well with number of threads

Cheers!

Atlas Kaan Yilmaz