Write Atomicity and NVM Drive Design

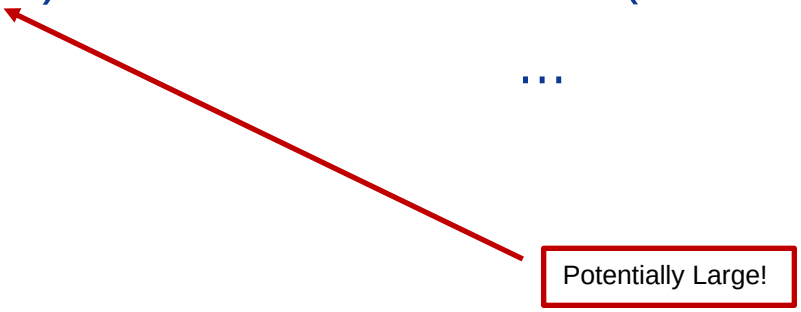
Andy Rudoff
Datacenter Software
Intel

Definitions

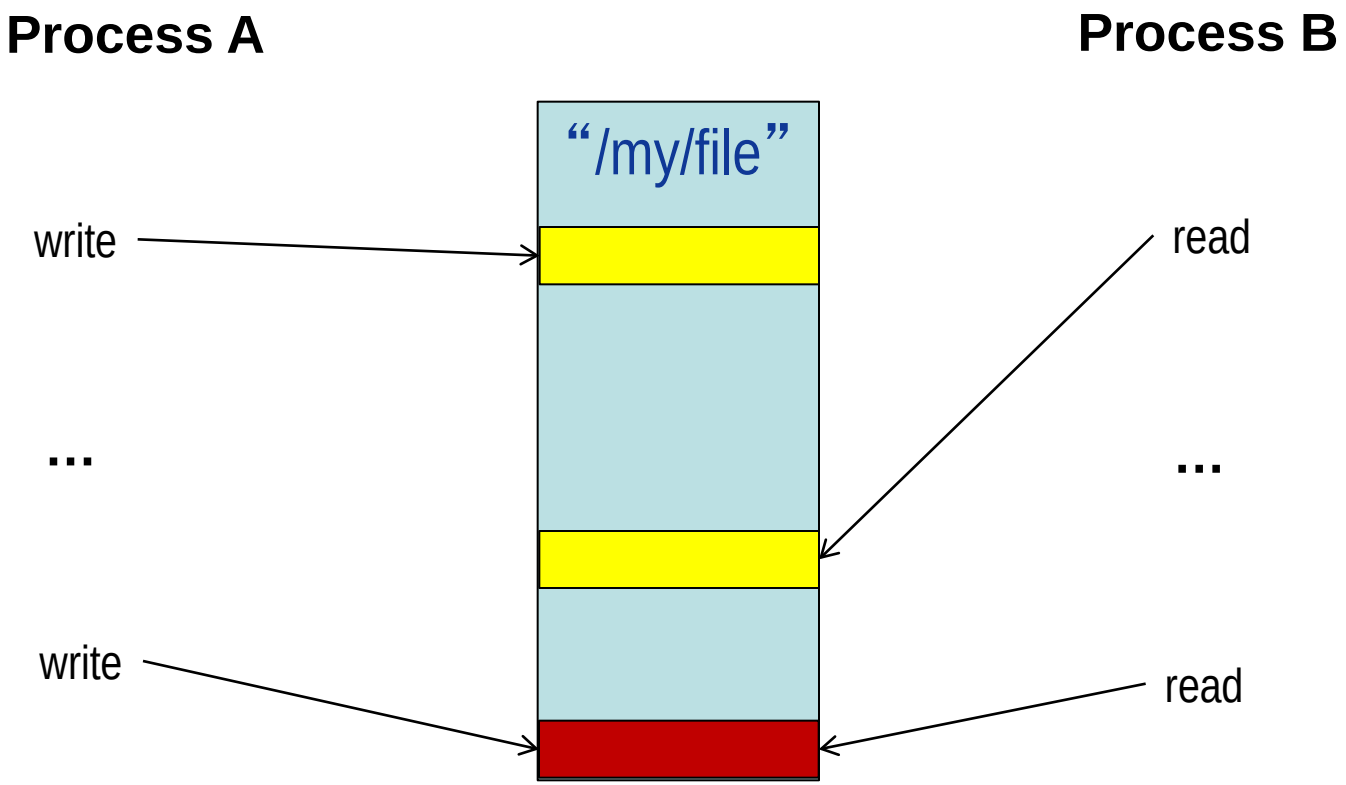
- Write Atomicity (non-powerfail)
  - At what granularity do writes become visible to other accesses
    - Block I/O
    - File I/O
    - Stores
- Powerfail Write Atomicity
  - At what granularity are writes interruptible by power failure
    - Block I/O
    - File I/O
    - Stores

Some History: POSIX

Process A	Process B
fd = open("/my/file", O_RDWR);	fd = open("/my/file", O_RDWR);
…	…
write(fd, buf, len);	read(fd, buf, len);
…	…
write(fd, buf, len);	write(fd, buf, len);
…	…
write(fd, buf, len);	read(fd, buf, len);
…	…



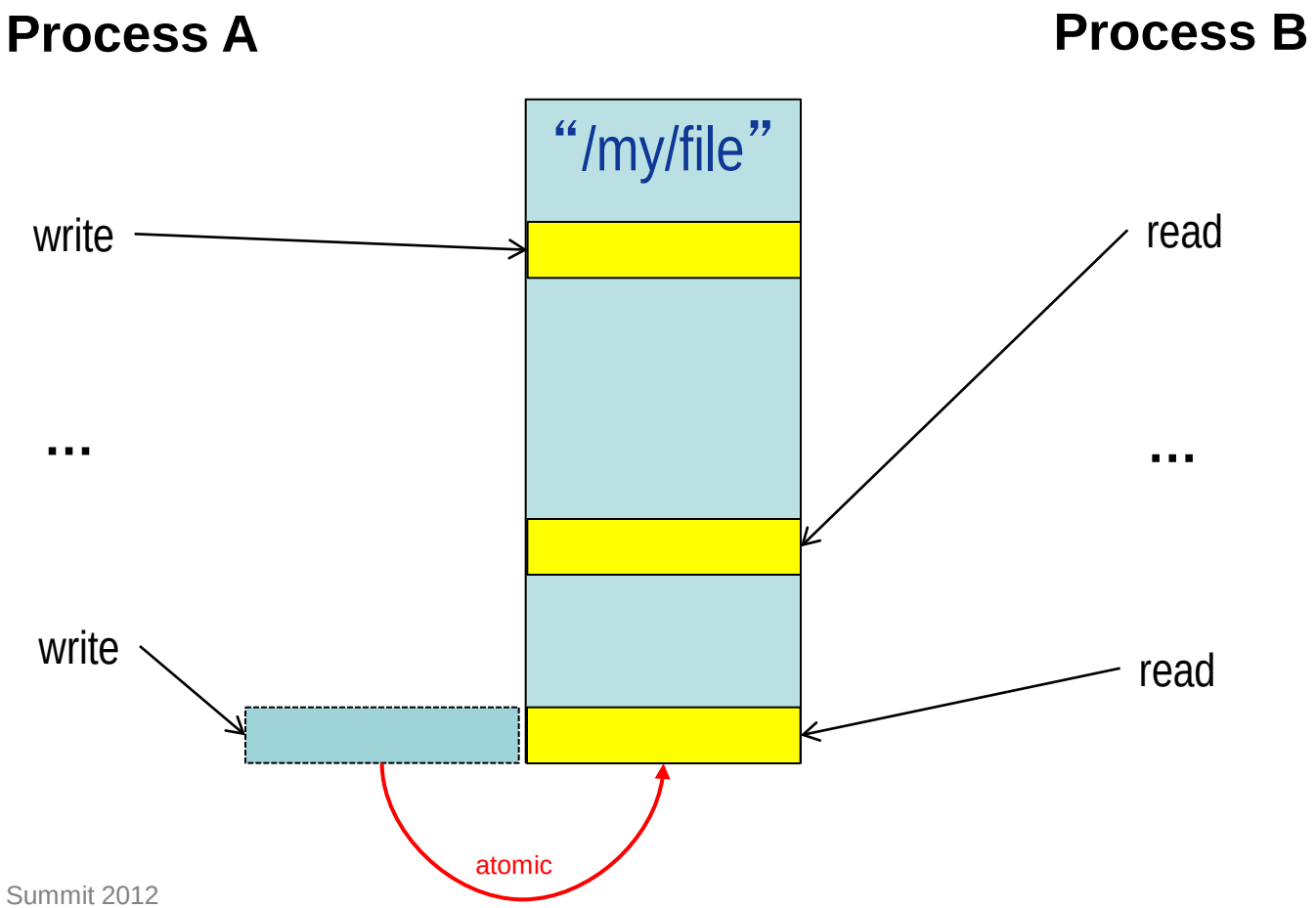
POSIX: Range Locking



Some History: File Access

- POSIX File Access
  - Requires write() must appear atomic to any readers accessing the file
  - Most commonly implemented with range locking
  - NFS implementations never fully complied with this (!)
  - More recent file systems and APIs are relaxing this
  - …but there are also newer COW filesystems to consider

Allocate-On-Write

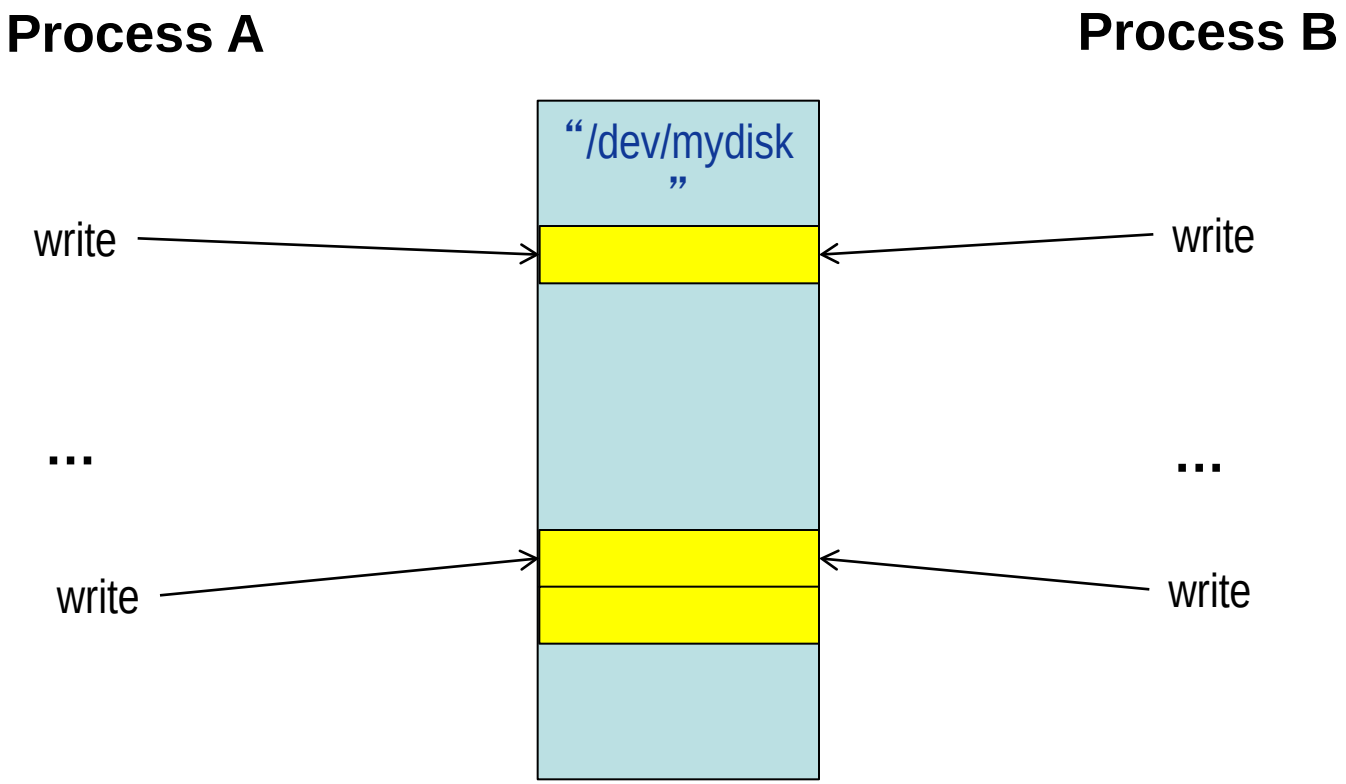


More History: File Access

- Windows WriteFile
  - Documents “single sector” atomicity
  - Provides CreateFileTransacted()

Although a single-sector write is atomic, a multi-sector write is not guaranteed to be atomic unless you are using a transaction (that is, the handle created is a transacted handle; for example, a handle created using CreateFileTransacted).

Block Device Atomicity



Some History: Block Access

"We consider it a bug for two threads to issue I/O to the same block on a raw device."

"Our application divides up accesses with locking & per-thread responsibilities."

"There are devices out there that do not ensure concurrent operations to the same sector don't interfere with each other."

Some History: Block Access

- "Every Enterprise SCSI drive provides 64k powerfail write-atomicity. We depend upon it and can silently corrupt data without it."
- "We require devices that will not tear 32k writes."
- "Disk drives use rotational energy to complete their writes on power loss."
- "Disk drives lie, saying a write is durable when it is actually in a non-powerfail-safe cache."
- "We depend on zero write-atomicity and use checksums on every block."

Some Observations: Block Access

- Many applications depend on 1 block powerfail WA
- Some applications (like mysql) can benefit from higher WA
  - innodb_doublewrite=0
- Many file systems depend on 1 block powerfail WA
  - Some require "fsck" to fix torn writes if no WA
- Some API documentation promises 1 block atomicity (doesn't specify powerfail)
- Many drivers do not provide non-powerfail WA at all
- Testing the powerfail case is hard

- Conclusion: there's lots of buggy SW out there

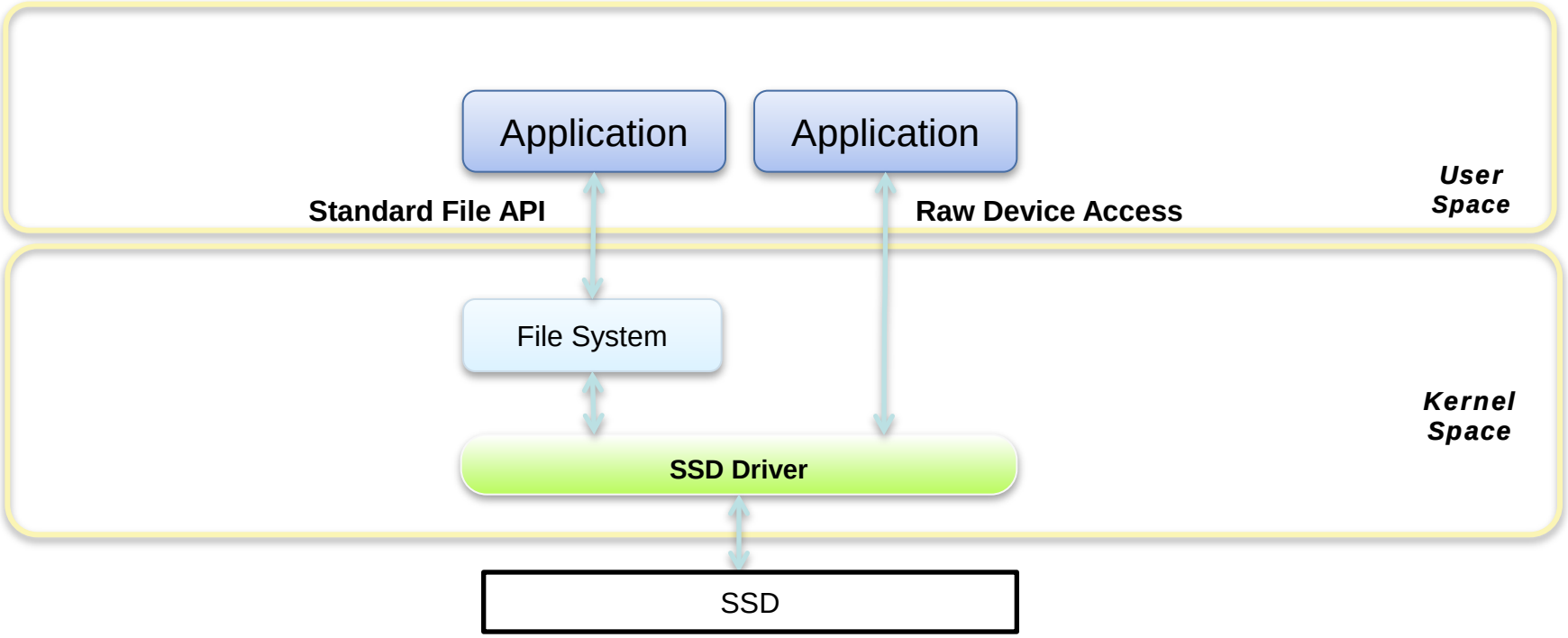
Software Implementation

- Applications (and kernel modules) can increase their *non-powerfail write atomicity* using locks between cooperating threads
  - Virtually everyone who cares takes this approach
- All applications (informally) surveyed agreed:
  - They do not depend on HW to prevent interleaving data between writes to the same block
  - Again, this could be because they don't realize they are depending on it

Software Implementation

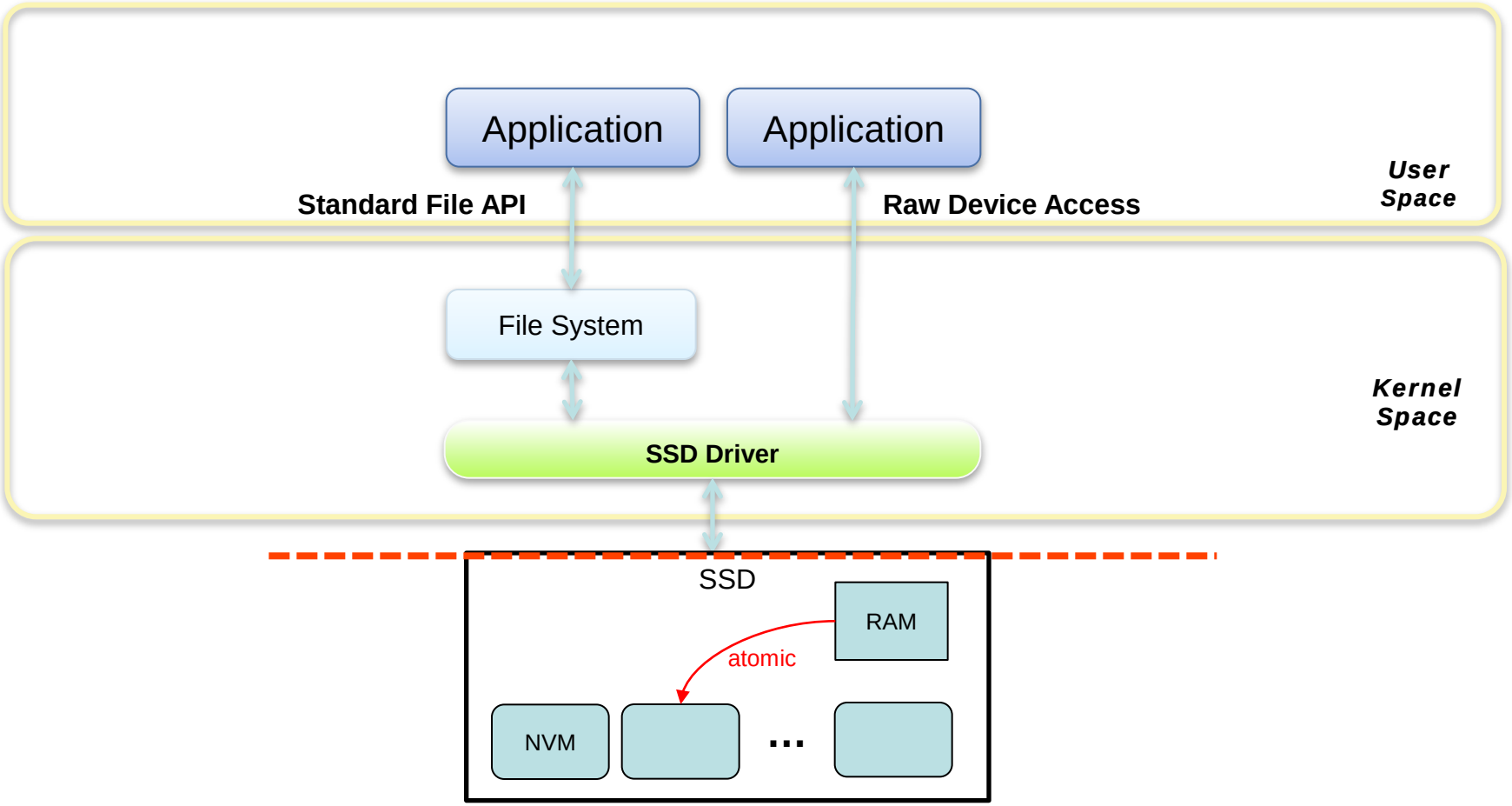
- Locking itself doesn't do anything to help *powerfail write atomicity*. There are two commons options here:
  - Write-ahead logging (WAL)
    - Software-only
    - HW-assisted
  - Shadowing
    - Software only
    - HW-assisted

The SSD Data Paths

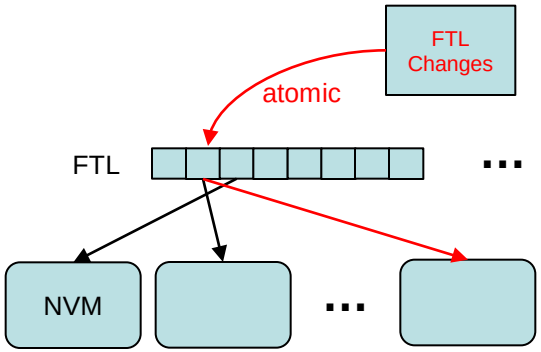


- Powerfail Write Atomicity could be implemented:
  - In the SSD Driver
  - In the SSD
- Interesting thought experiment
  - An SSD build entirely from battery-backed RAM…

Defining the Powerfail Safe Domain



Leveraging the FTL



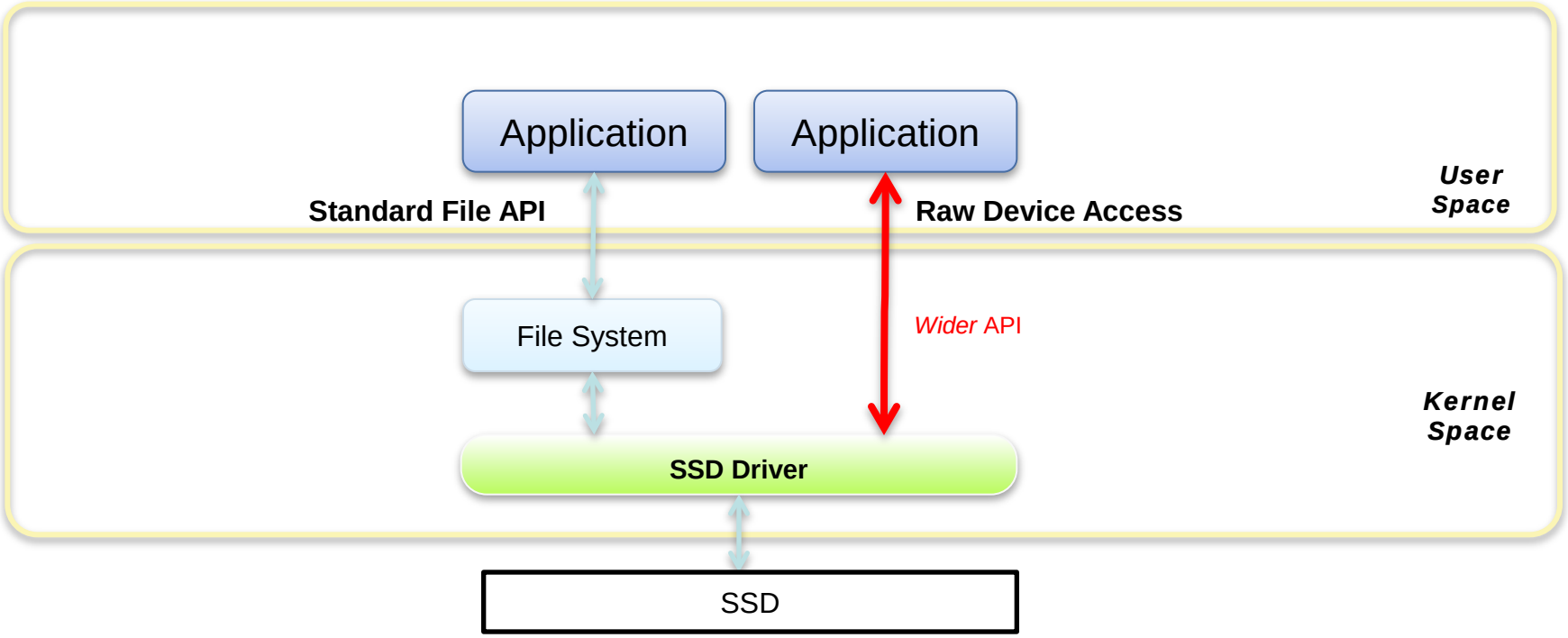
Discovering Atomicity

- Applications have no idea what atomicity they can depend on without:
  - An error-prone administrative configuration step
  - Locking into a vendor-specific API
  - Making a usually-right but sometimes-wrong assumption (!)
- Solutions are starting to surface

NVMe Identify Controller

527:526	M	**Atomic Write Unit Normal (AWUN):** This field indicates the atomic write size for the controller during normal operation. This field is specified in logical blocks and is a 0's based value. If a write is issued of this size or less, the host is guaranteed that the write is atomic to the NVM with respect to other read or write operations. A value of FFFFh indicates all commands are atomic as this is the largest command size. It is recommended that implementations support a minimum of 128KB (appropriately scaled based on LBA size).
529:528	M	**Atomic Write Unit Power Fail (AWUPF):** This field indicates the atomic write size for the controller during a power fail condition. This field is specified in logical blocks and is a 0's based value. If a write is issued of this size or less, the host is guaranteed that the write is atomic to the NVM with respect to other read or write operations.

Exposing Capabilities



An API to Expose Capabilities

```
len = nvm_capability(fd, NVM_PFWRITE_ATOMICITY);
```

The atomicity of a single write with respect to power failure. Writes of this size in bytes or smaller cannot be “torn” by system crash or loss of power – either the entire write takes place or the entire write does not take place.

SNIA NVM Programming TWG

- Founding members
  - Dell, EMC, Fujitsu, HP, IBM, Intel, NetApp, Oracle, QLogic, Symantec
- Charter: Develop specifications for new software "programming models" as NVM becomes a standard feature of platforms
  - Scope:
    - In-kernel NVM programming models
    - Kernel-to-application programming models
  - Programming models specify the exact technical behavior, up to (but not including) the OS specific API semantics
- APIs
  - Each OSV codes the programming models to specific to OS
  - Linux Open Source project underway to provide the Linux implementation of this effort
  - Ex: SNIA NVM Programming TWG + Linux open source project provides the full solution for Linux.

Summary: For **SSD** Products

- Atomic writes (non-powerfail)
  - Desirable, many examples where they don't exist
  - Many examples where apps don't depend on it
  - A place for IHVs to add value
- Powerfail Atomic Writes
  - Strongly recommend 1 block minimum
  - Higher is better
  - An even *better* place to add value
- Discovery
  - Cultivate an ecosystem where drivers and applications can discover the properties of NVM