FAST'25

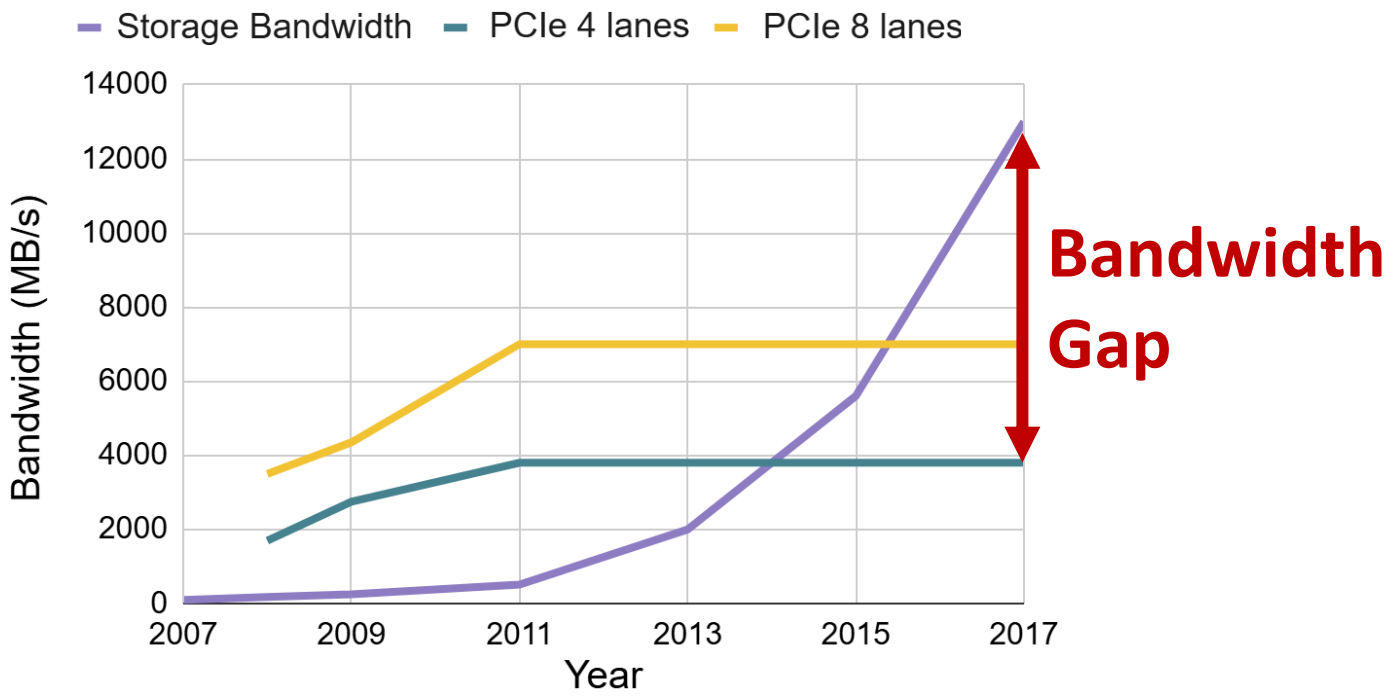
# Selective On-Device Execution of Data-Dependent Read I/Os

*Chanyoung Park, Minu Chung, Hyungon Moon*

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

# First Issue from the Evolution of Storage Devices

- The bandwidth between a host and a driver becomes a bottleneck due to the bandwidth gap between PCIe and storage internals

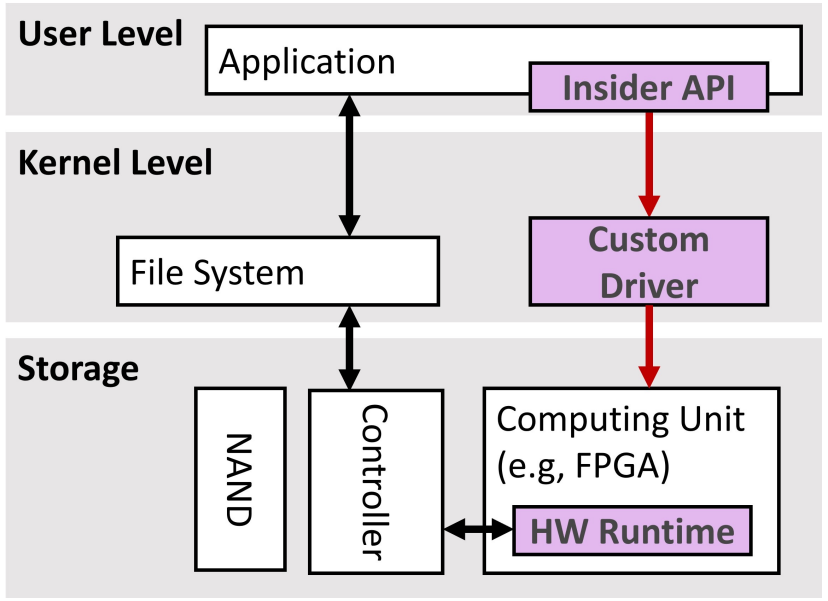



*[ Bandwidth Evolution of Storage Devices ]*

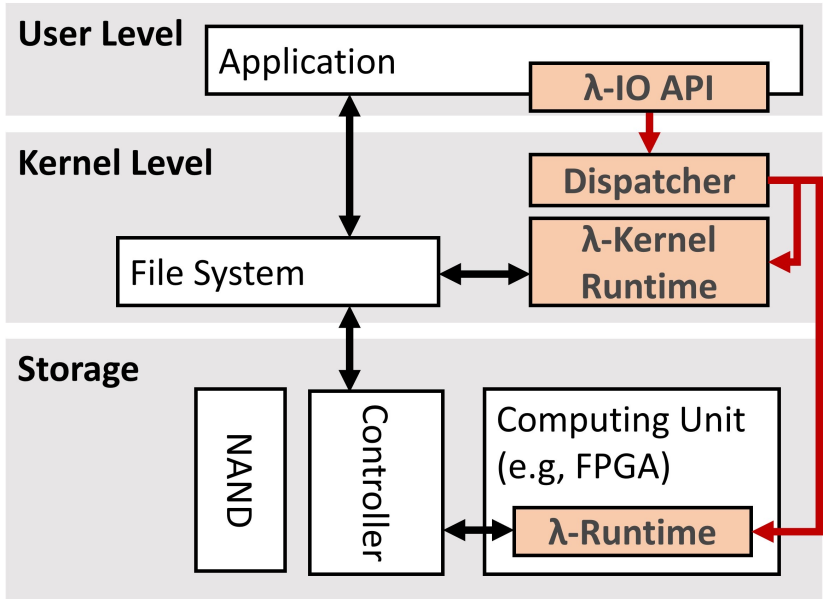
# In-Storage Computing (ISC)

- ISC also offloads data-intensive computations into the storage device to reduce data transfer
- ISC leverages the higher internal bandwidth of SSDs for throughput-centric workloads

*Example 1: Insider [ATC 19]*

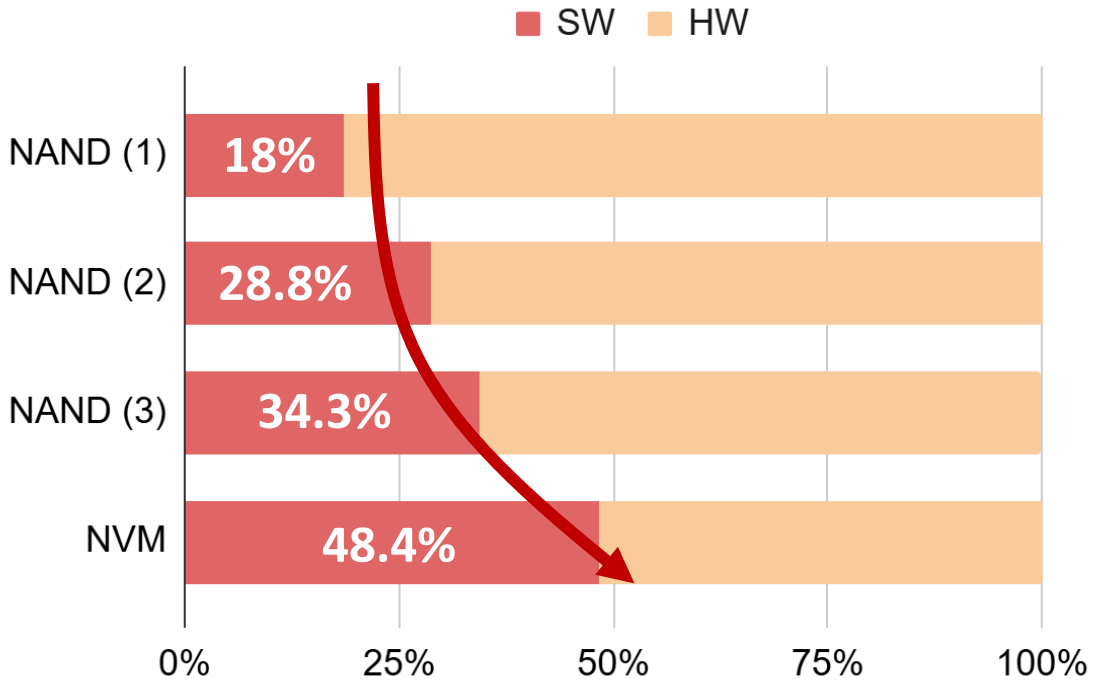



*Example 2: λ-IO [FAST 23]*



# Second Issue from the Evolution of Storage Devices

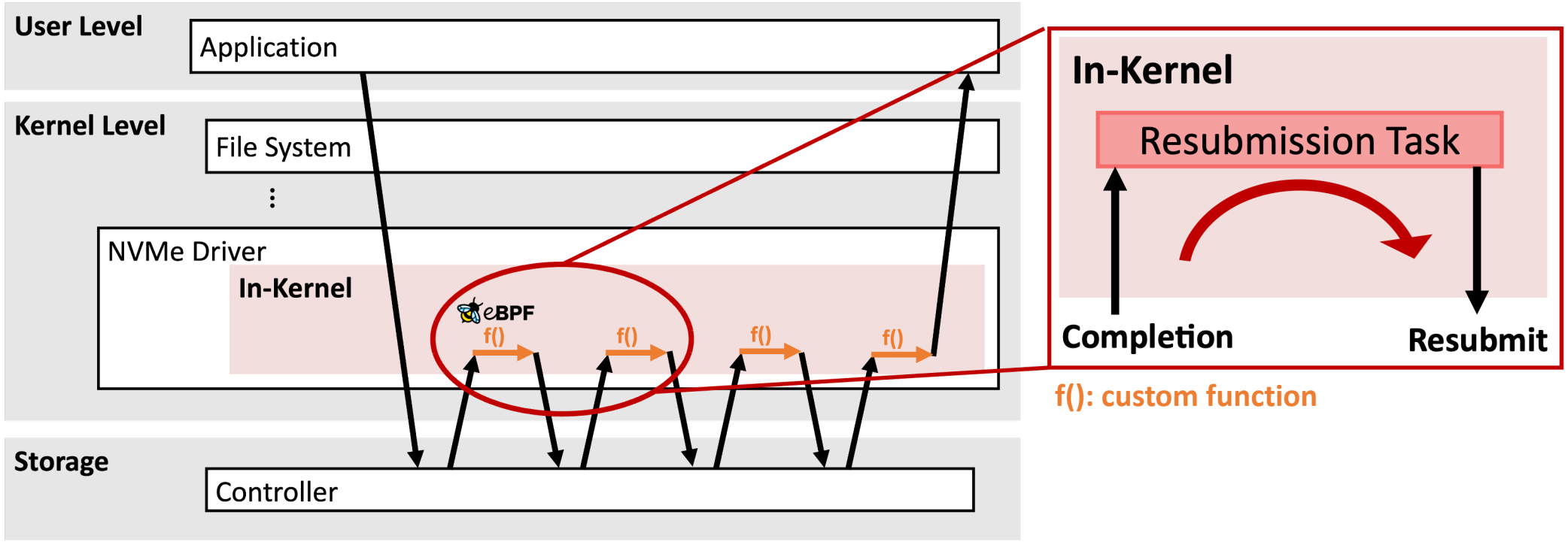
- Kernel software stack becomes a bottleneck due to the high speed of modern storage devices




*[ The ratio of Kernel SW stack latency about a 512B read() ]*

# In-host Near Storage Computing (NSC)

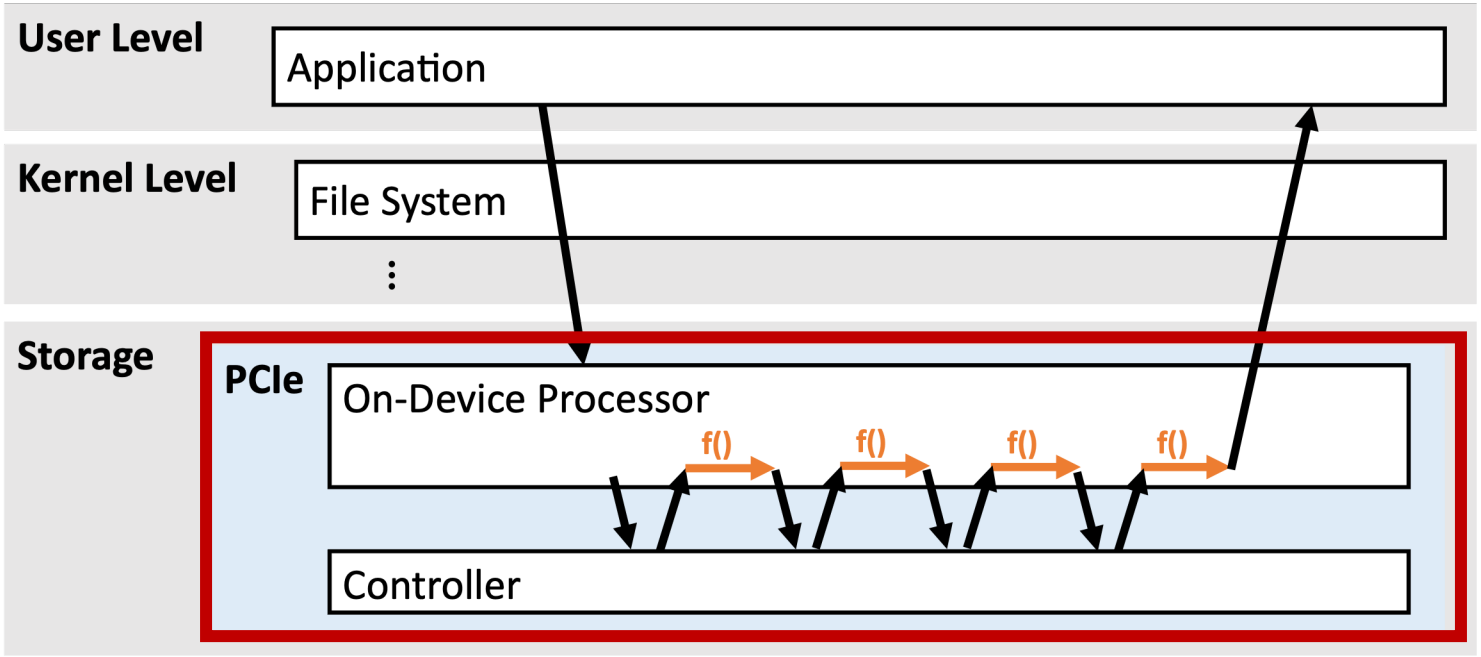
- In-host NSC offloads latency-critical computations closer to the storage device, allowing it to bypass the kernel software stack
- Example: XRP [OSDI 22] accelerates data-dependent read I/Os with eBPF
  - In the NVMe driver, XRP executes user-defined eBPF and resubmits the next read request



# Unexplored Question: ISC for Data-Dependent Reads?

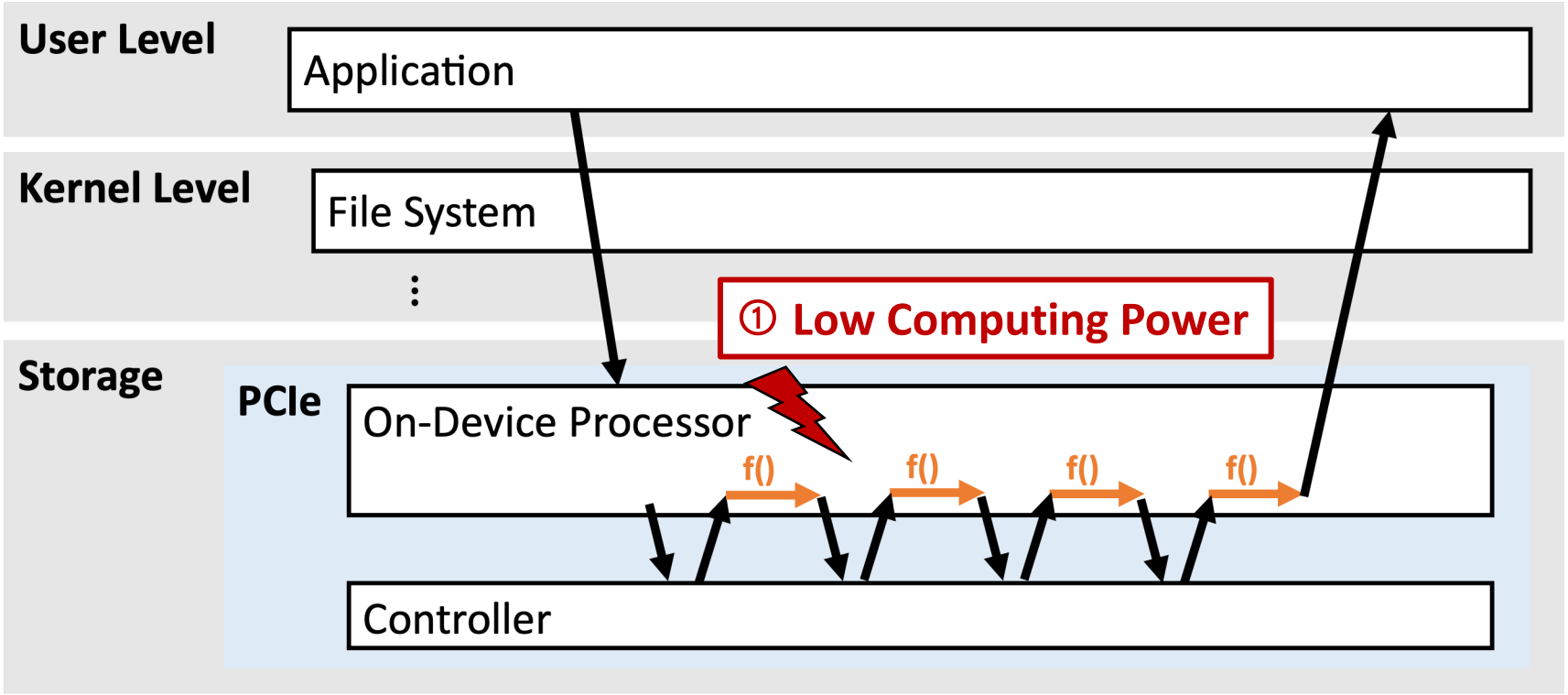
- We explore whether in-storage computing can expedite data-dependent read I/Os as an example of latency-critical cases

→ We must understand the characteristics of in-storage computing when moving resubmission tasks into the storage



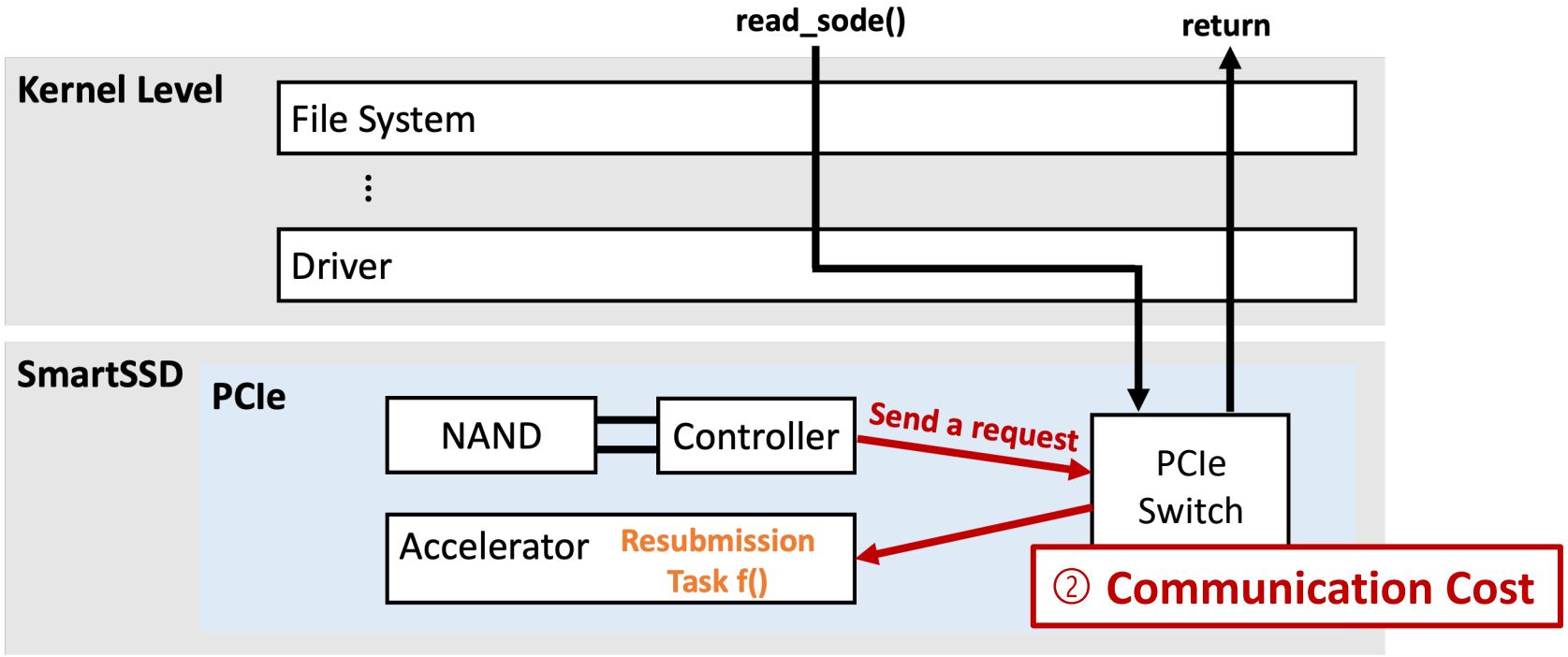
# Characteristic 1: Low Computing Power of Storage Devices

1. The on-device processor has relatively low computing power
   - Each operation experiences higher latency, reducing the advantage of lower data access latency



# Characteristic 2: Disadvantages of Access Latency

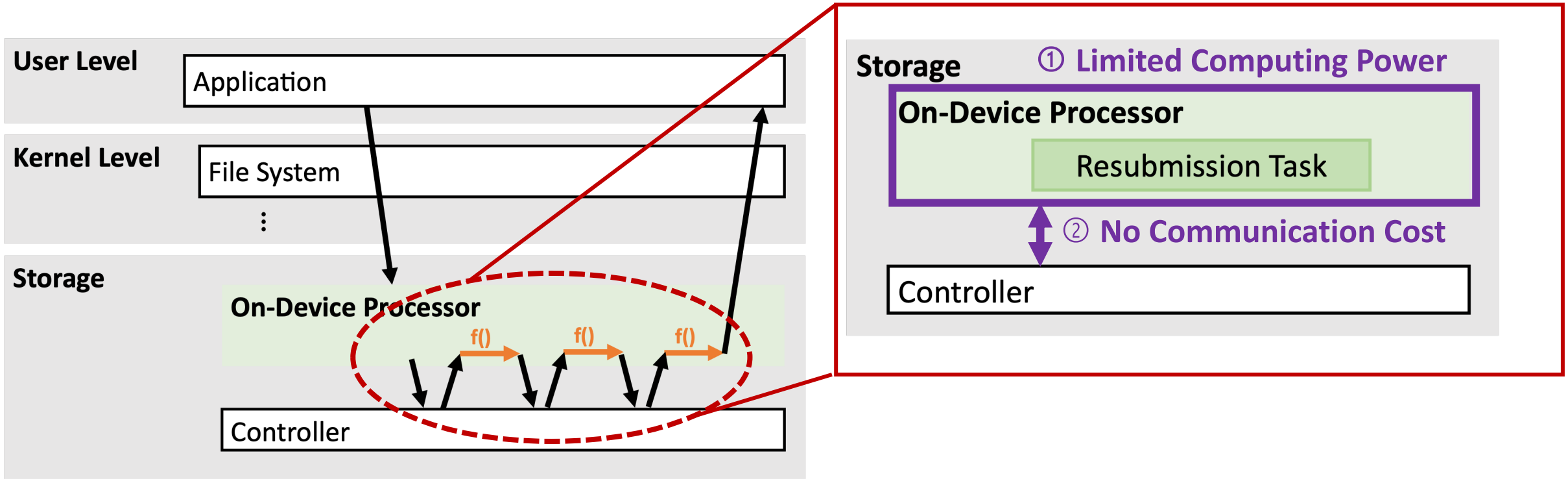
2. Prior commercial computational storage devices are not designed for the potential advantage of access latency
   - ➢ Their access latency is not significantly lower than that from the host processor




*Example: Samsung SmartSSD*

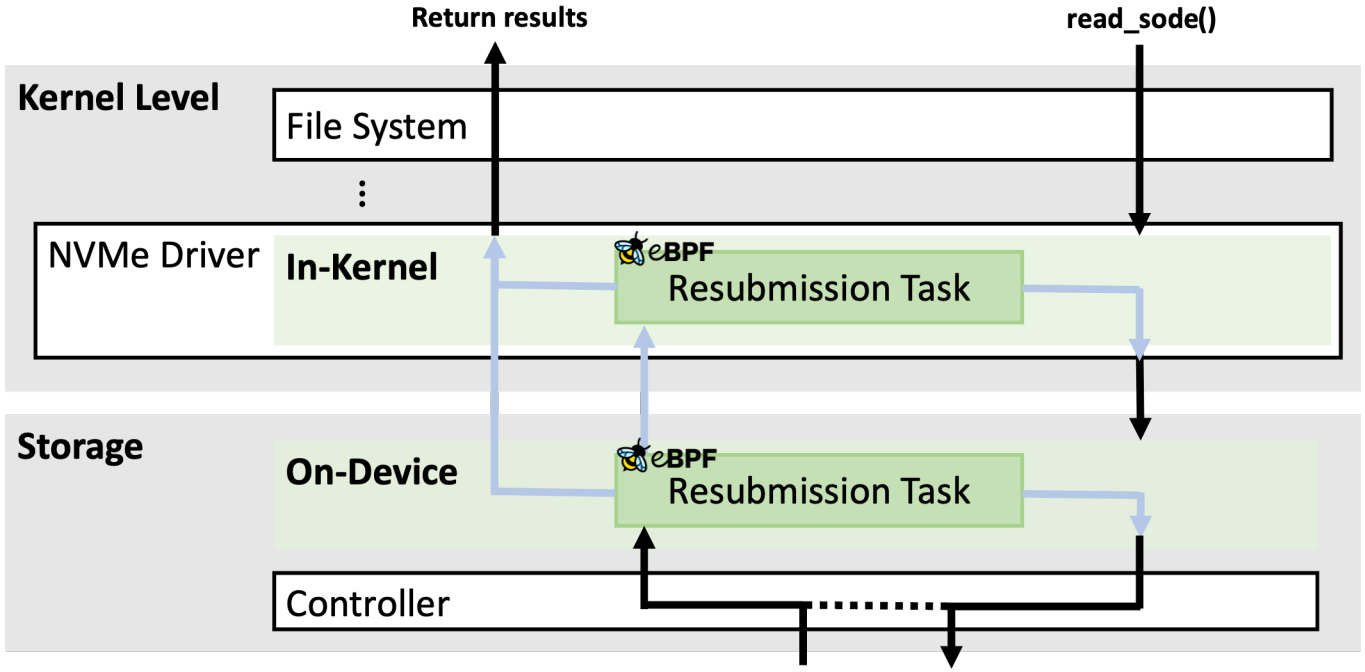
# Constraints from Characteristics of ISC

- These characteristics cause **two constraints** for latency-critical workloads
  1. Considering the **limited computing power** of storage devices
  2. The on-device processor must be located possibly **closer to the controller**



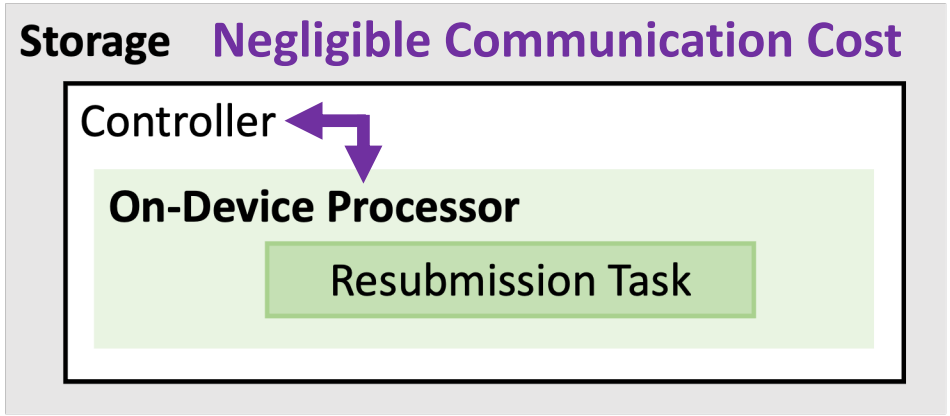
# Selective On-Device Execution (SODE)

- SODE explores on-device computing in data-dependent read I/Os, where the reduced data transfer is not beneficial
- SODE introduces **hybrid resubmission** that selectively balances on-device and in-storage executions
  - From Constraint 1: Considering the limited computing power of storage devices

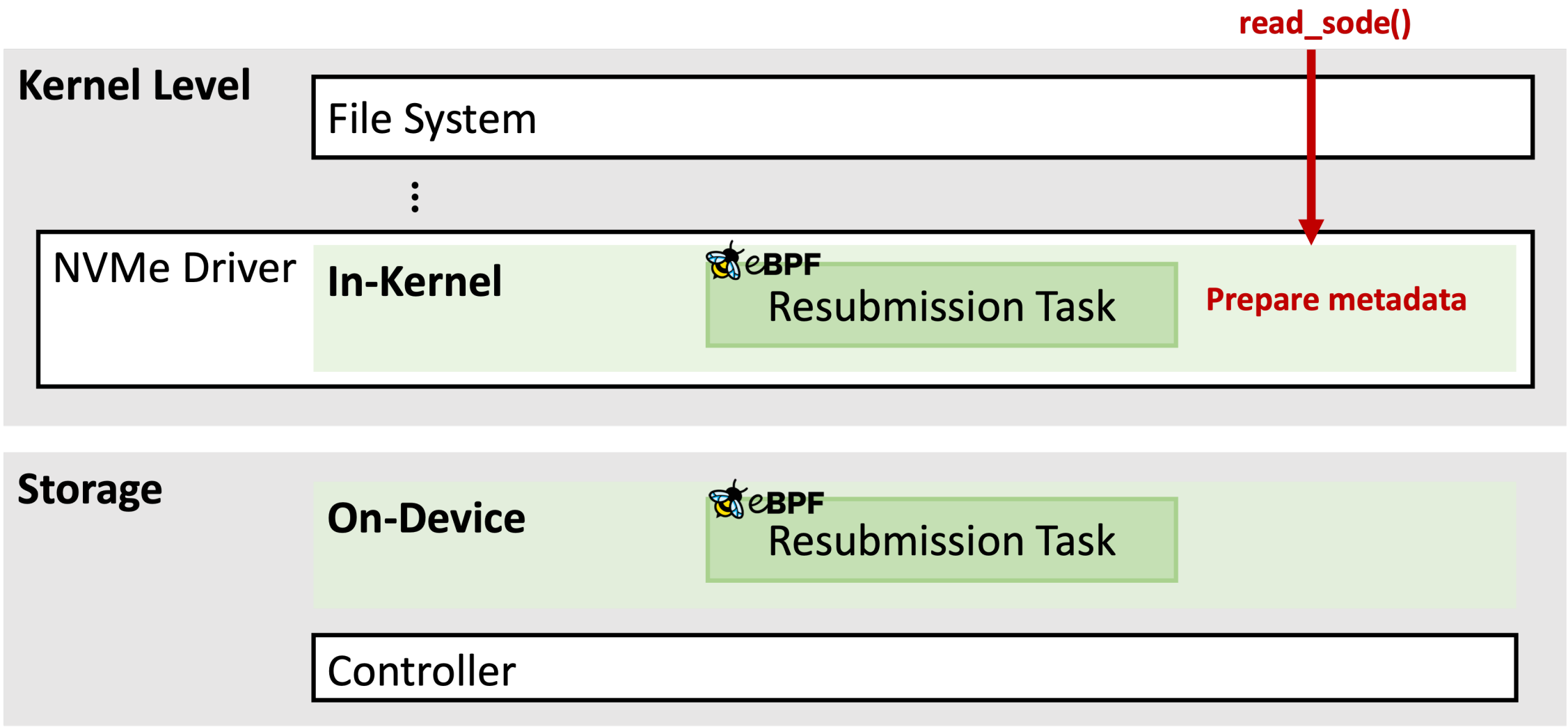


# Selective On-Device Execution (SODE)

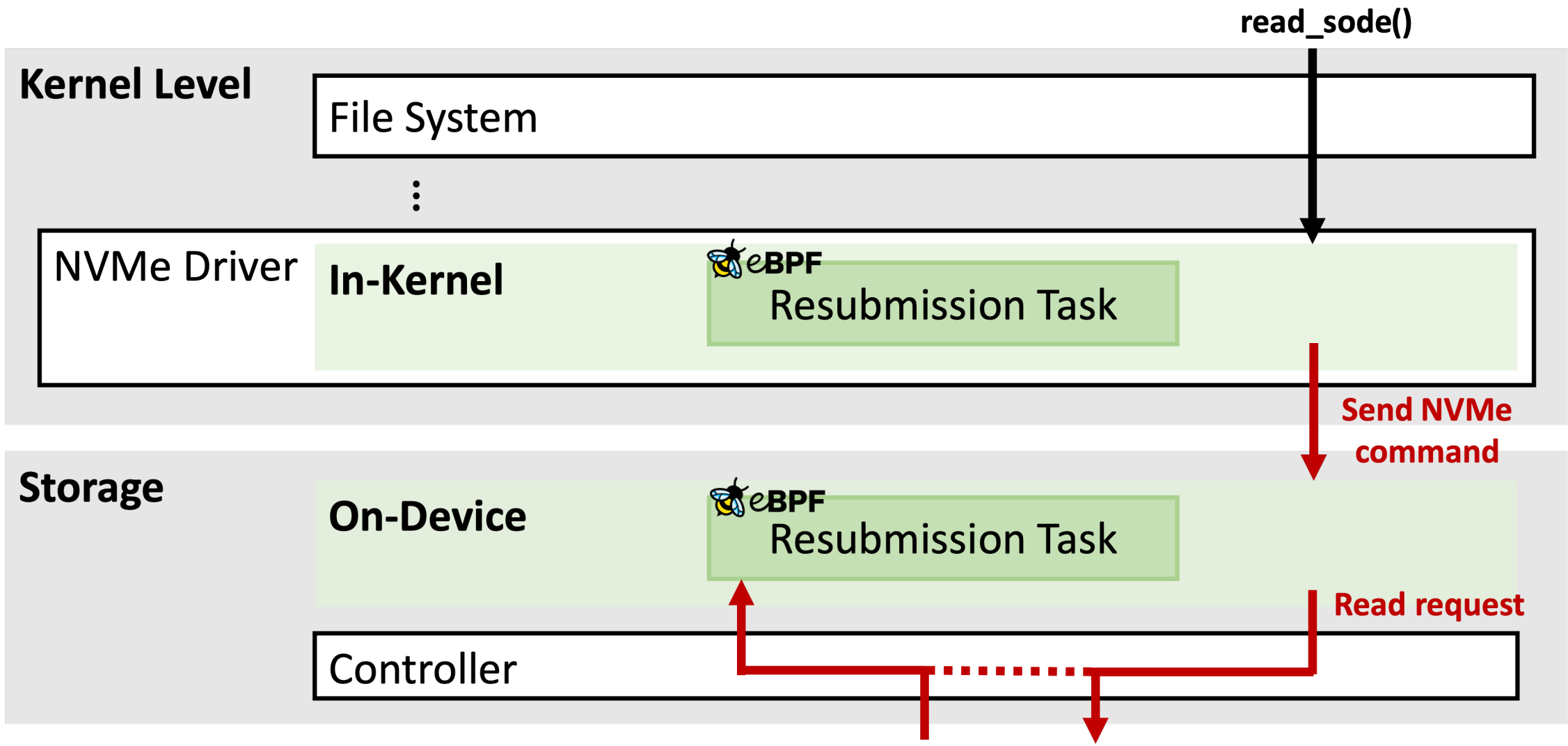
- SODE explores on-device computing in data-dependent read I/Os, where the reduced data transfer is not beneficial
- SODE introduces **hybrid resubmission** that selectively balances on-device and in-storage executions
  - From Constraint 1: Considering the limited computing power of storage devices
- On-device processor is located **possibly within the same System-on-Chip (SoC)** as a storage controller
  - From Constraint 2: The on-device processor must be located closer to the controller



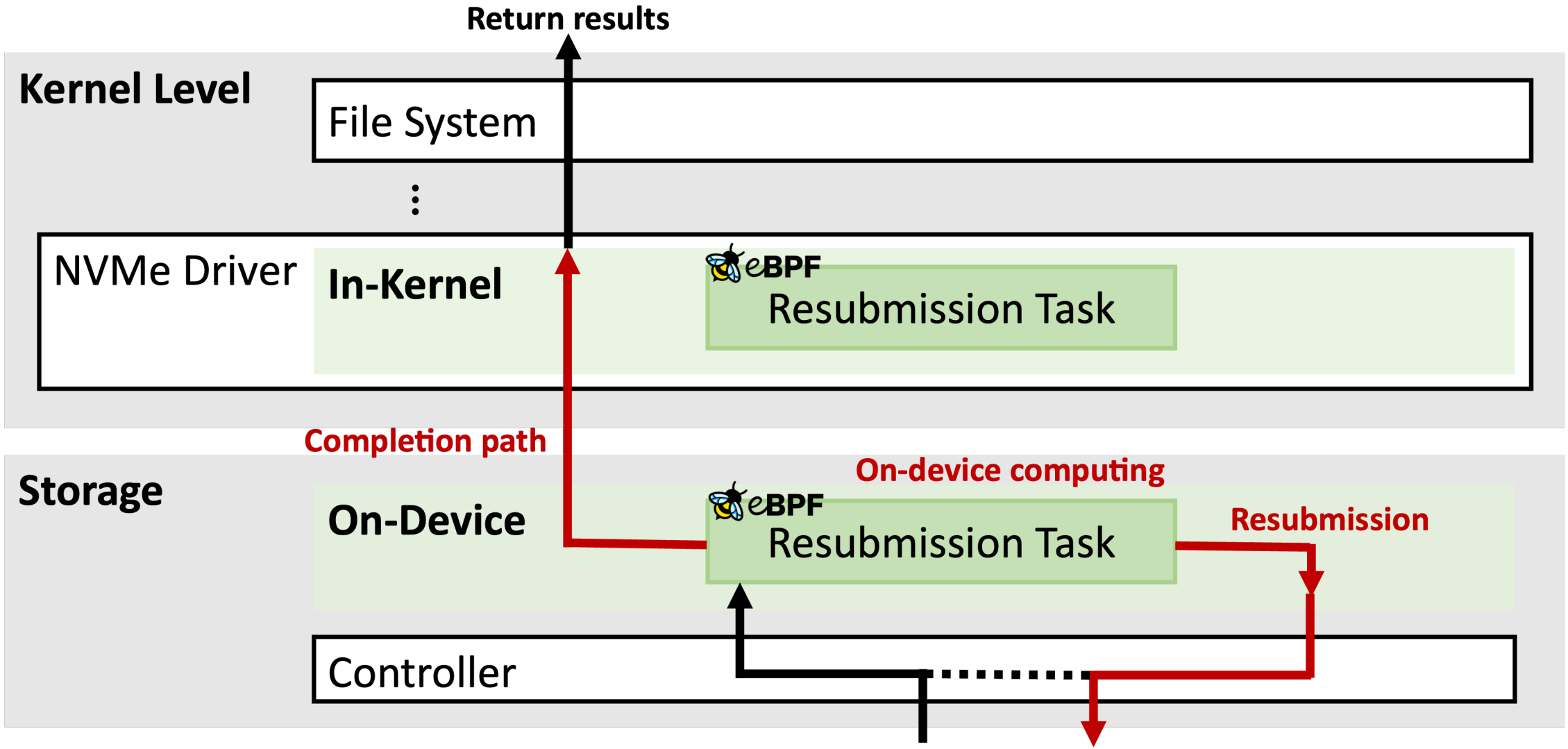
# Hybrid Resubmission



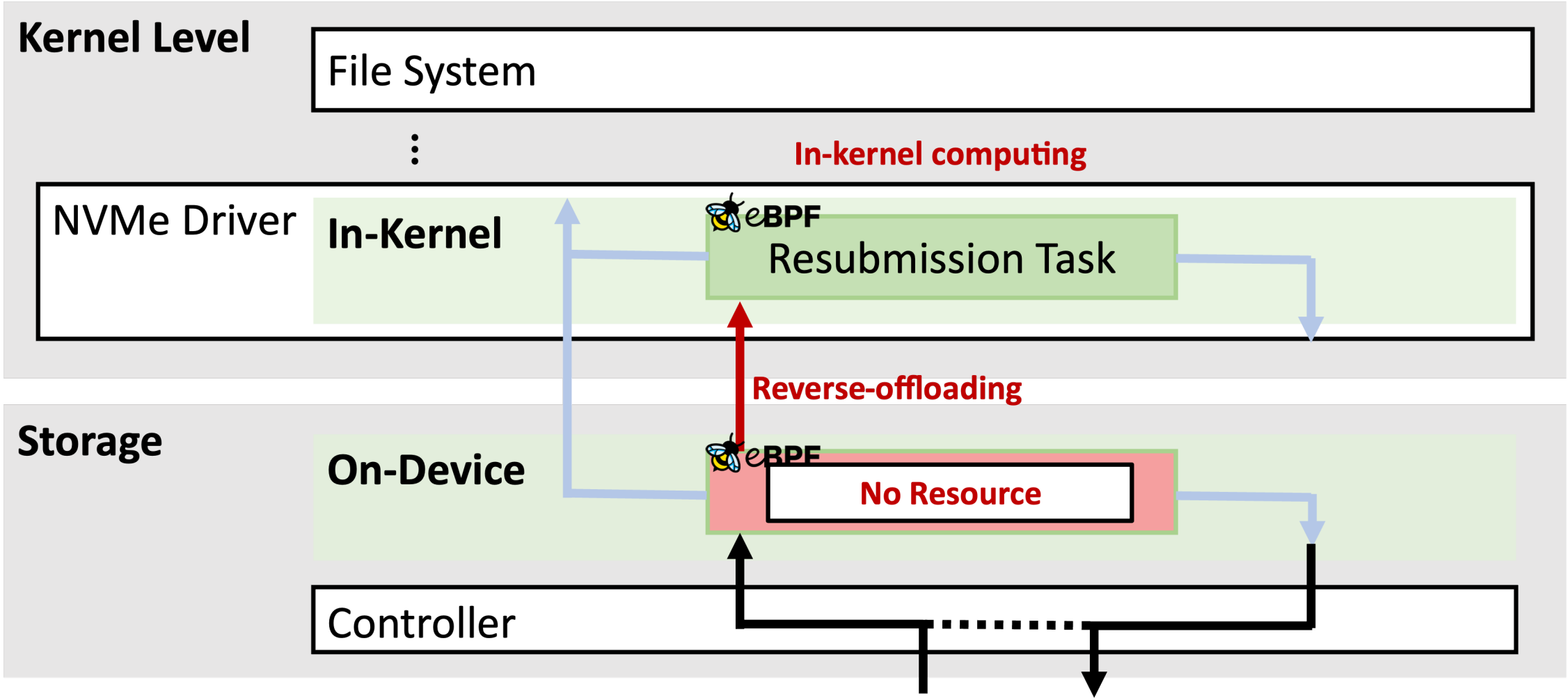
# Hybrid Resubmission



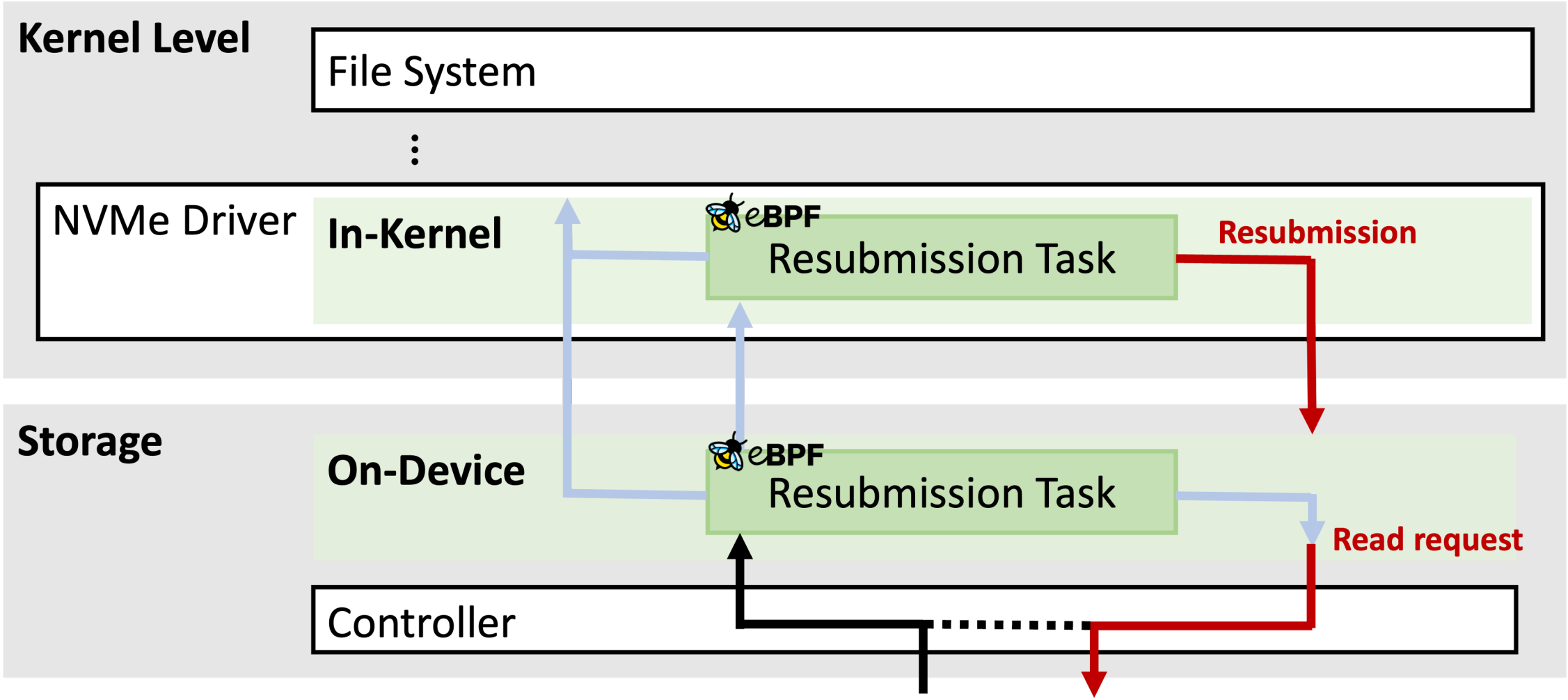
# Hybrid Resubmission



# Hybrid Resubmission: Reverse-offloading

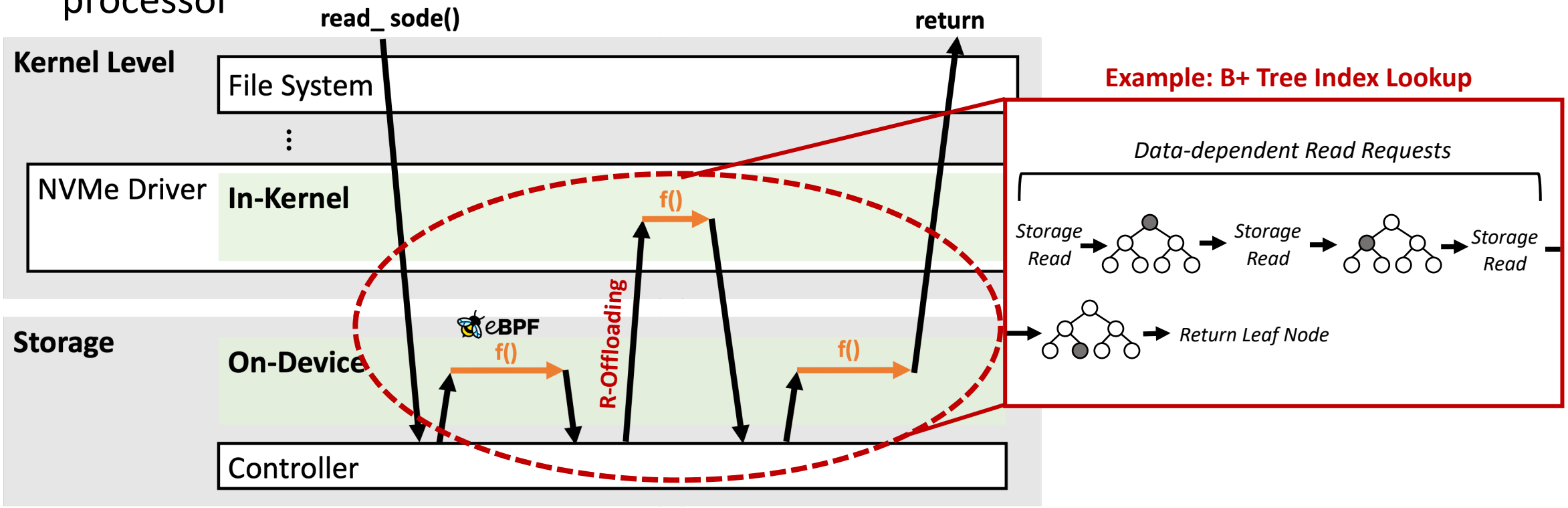


# Hybrid Resubmission



# Hybrid Resubmission: Example

- As a example of data-dependent read I/Os, SODE can traverse read-only large on-disk data structures such as LSM trees and B-trees
- The reverse-offloading complements the low computing power of the on-device processor



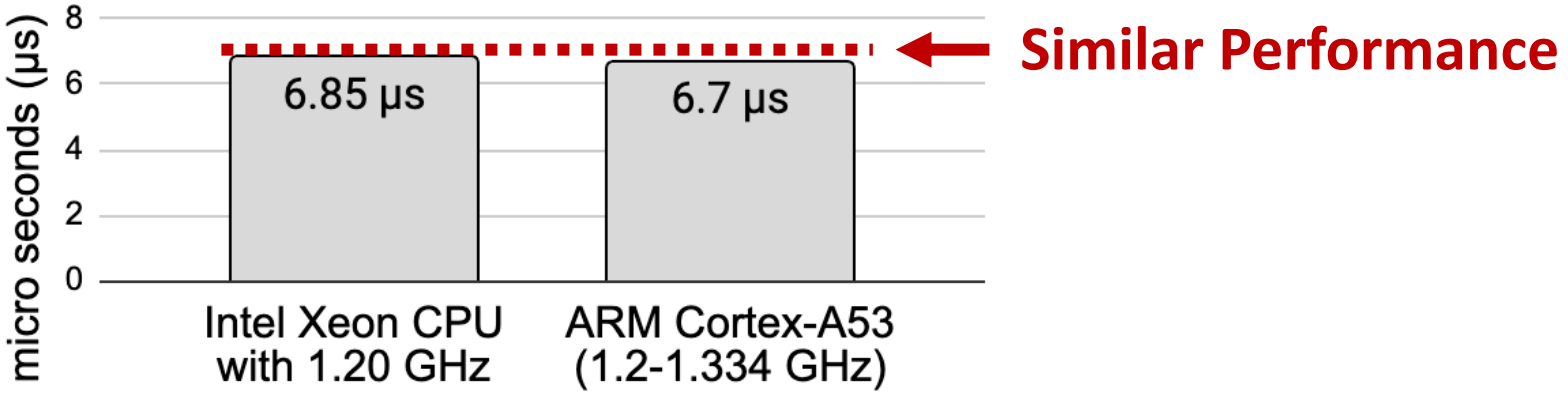

f(): custom function

# Further Design Details

- Translating file offsets in on-device and in-kernel
  - Cached file metadata
- Supporting heavy on-device resubmission tasks
  - Parallel resubmission tasks
- Sandboxing on-device resubmission tasks
  - Emulating Memory Protection Key (MPK) based sandboxing overhead

# SODE Prototype

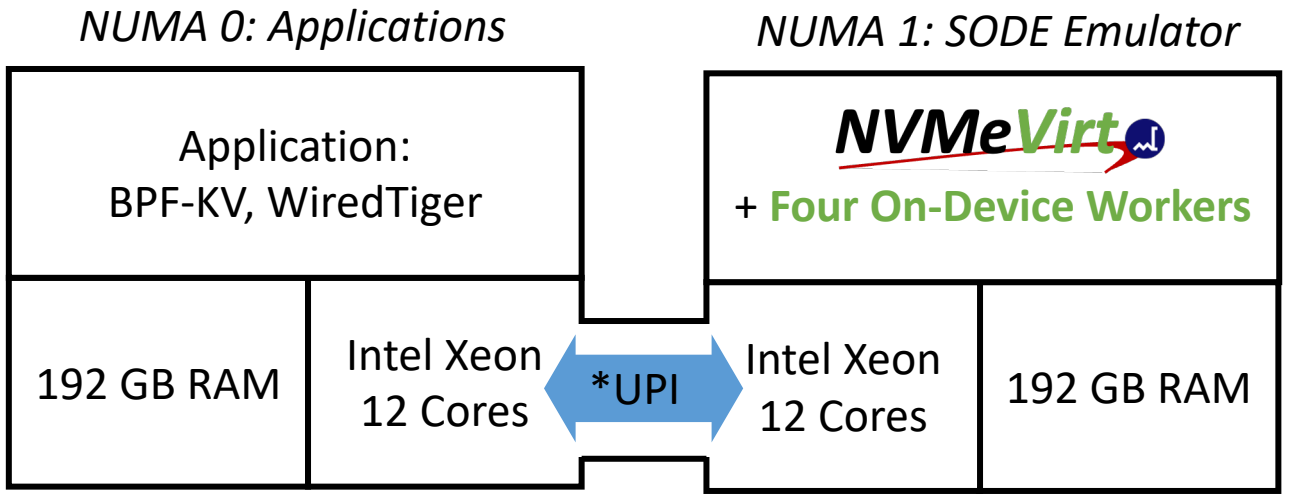
- We implement a prototype of SODE by extending NVMeVirt
- NVMeVirt is a software-based emulator for NVMe-based storage devices
- We select the “Intel OptaneDC SSD” model to use low-latency storage devices
- ARM Cortex-A53 is a reference for our on-device processor.
- We limit CPU performance for on-device computing to emulate wimpy on-device cores




*[ Microbenchmark results on different CPUs ]*

# Experimental Setup

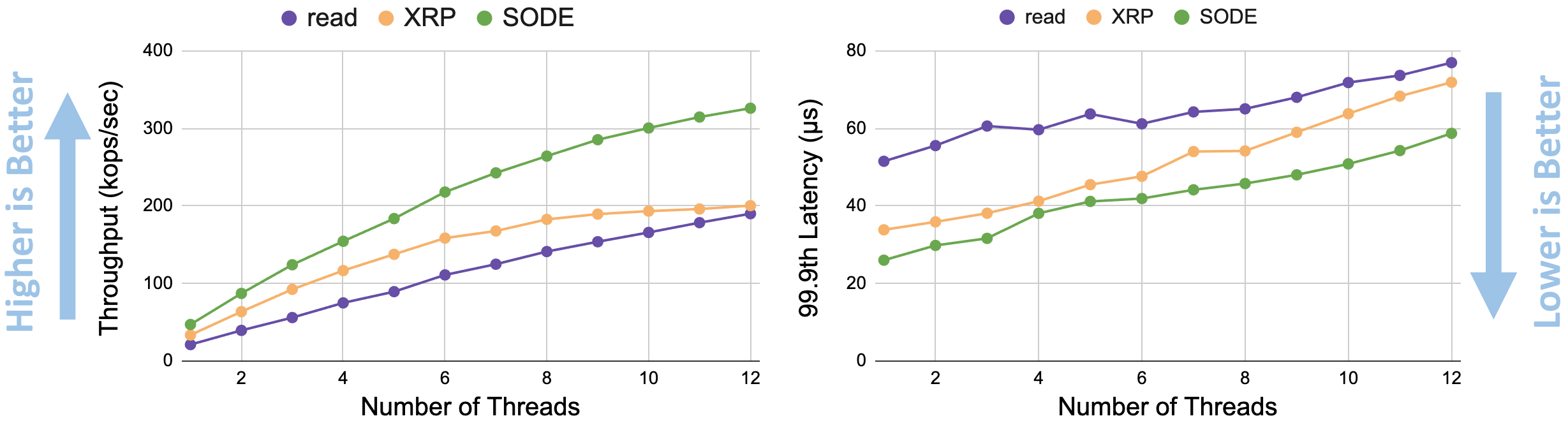
- Ubuntu 18.04 with Linux 5.12.0 kernel
- Two Intel Xeon Gold 6136 (12 cores, 1.20 - 3.00 GHz)
- 192GB RAM for applications + 192GB RAM for SODE storage emulation
- We bind four cores, similar to the four-core ARM Cortex-A53
- We assign one resubmission worker to each core




**UPI: Intel Ultra Path Interconnect*

# SODE Eliminates Kernel Software Stack Overheads and PCIe Roundtrip

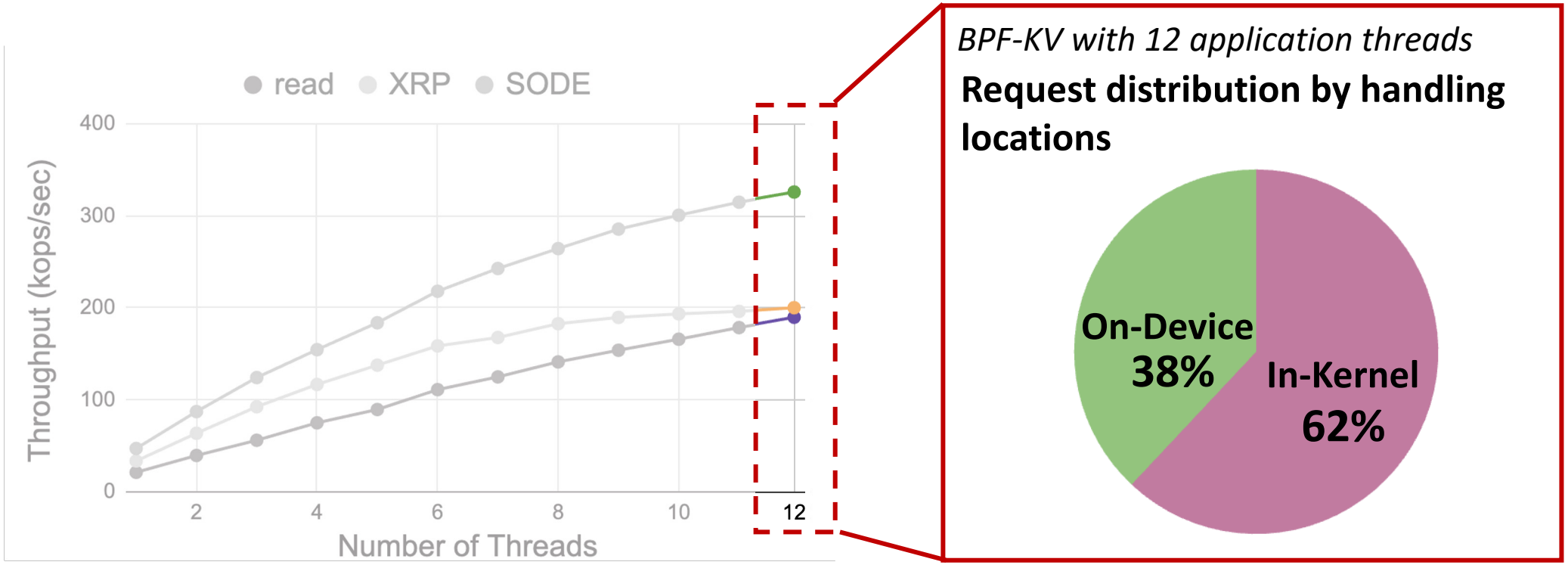
- SODE outperforms read() syscall and XRP regardless of the number of the application thread




*[ Multi-threaded results in **BPF-KV** with uniform random 512B read and Index depth 6 ]*

# SODE Eliminates Kernel Software Stack Overheads and PCIe Roundtrip

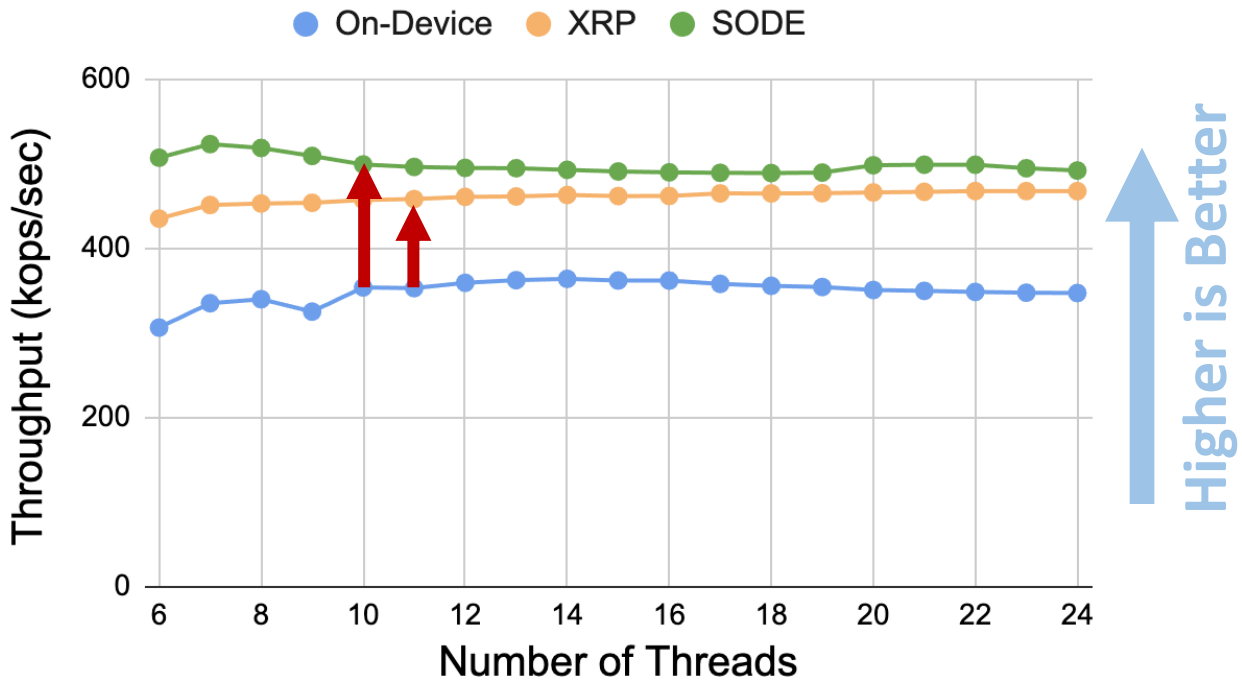
- SODE outperforms read() syscall and XRP regardless of the number of the application thread




*[ Multi-threaded results in **BPF-KV** with uniform random 512B read and Index depth 6 ]*

# SODE is Better Than On-device Only or In-Kernel Only

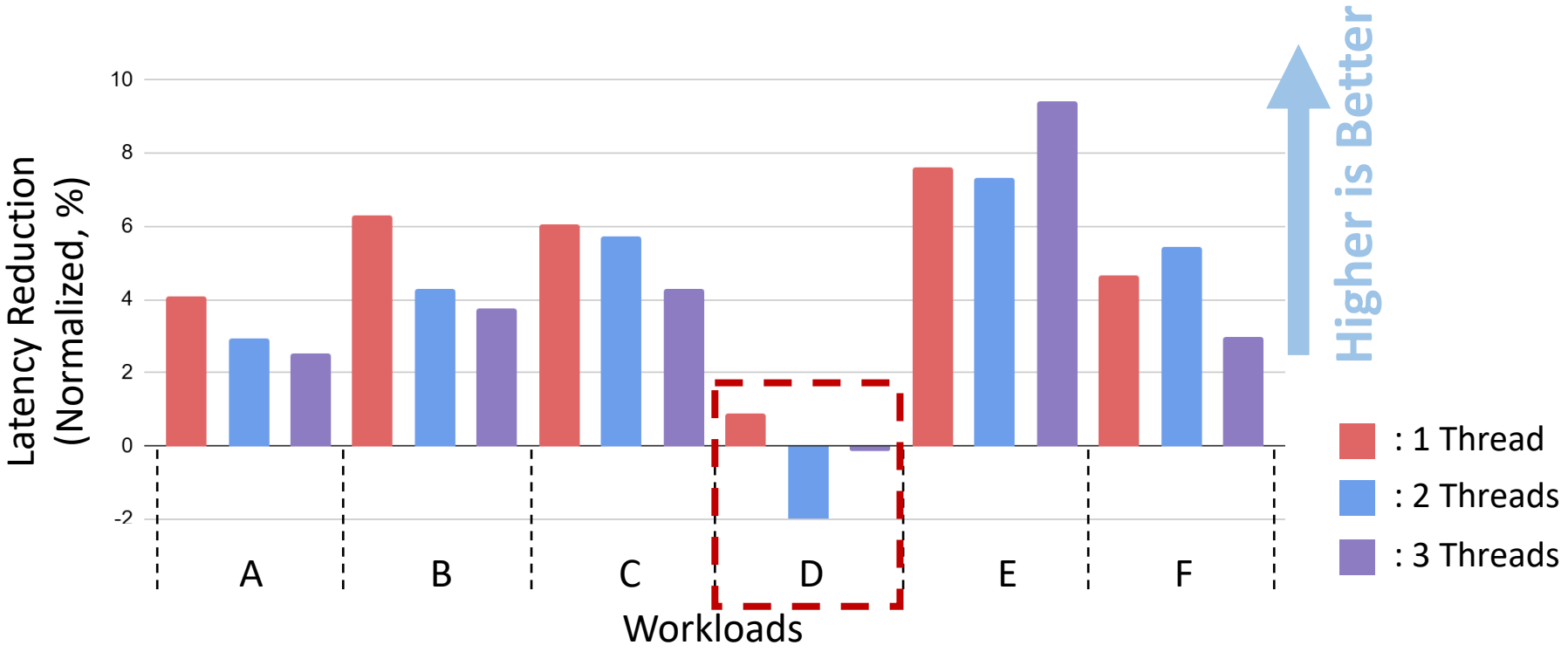
- Using on-device computing alone cannot achieve better performance than XRP, which is similar to in-kernel execution alone
- SODE has about 10.5% throughput higher than that of XRP




*[ **BPF-KV** results with an open-loop load generator and with varying threads ]*

# SODE Reduce Resubmission Latency on Real World Application

- On **WiredTiger** with YCSB workloads, we evaluated resubmitting system call latency of SODE compared to XRP
- SODE can decrease resubmission system call latency up to 9.4% compared to XRP
- Workload D incurs a small number of resubmissions by shallow on-disk tree depth



# Conclusion

- The ecosystem of in-storage computing does not fit in latency-critical cases
  - The limited performance of on-device processors can lower the advantage of data access latency
  - Commercial computational storage devices can diminish the potential advantage of access latency
- SODE employs hybrid resubmission, which selectively utilizes on-device and in-kernel computing
- SODE achieves better performance than approaches that entirely rely on either in-kernel or on-device for data-dependent read I/Os

Our code is available on GitHub