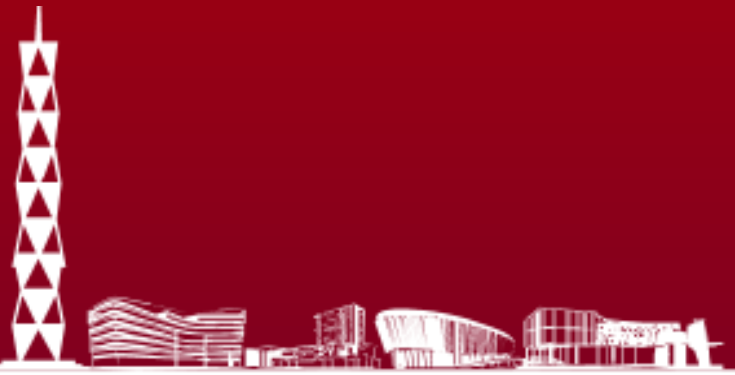
HotStorage 2025
The 17th ACM Workshop on Hot Topics in Storage and File Systems
July 10-11
Boston, MA

上海科技大学
ShanghaiTech University

Bridging a Shortcut to Exempt the Software Tax Charged on Logging I/Os

Author: **Yanpeng Hu**, Yunxin Yang, Li Zhu and Chundong Wang
ShanghaiTech University



立志成才报国裕民

Outline

- Introduction
- Background
- Motivation
- Design
- Experiments
- Conclusion

Introduction

- Logging (WAL) ensures database durability but causes high I/O overhead
- fsync/fdatasync syscalls for WAL are slow, bottlenecking databases like OceanBase
- High speed SSDs reduce I/O time, but software tax dominates (60.2% latency)
- Existing solutions (SPDK, kernel-bypass) need complex software/hardware changes

Introduction

- **Key Insight**:
- Many databases use **preallocated** logs
- Preallocated log files have stable layouts
  - No resizing/permission changes.
- **Éxitos**: Uses eBPF to bypass software layers
  - Maps file offsets to disk block LBAs efficiently
  - Redirects I/O via ioctl to SSD driver

Introduction

- **Key Insight**:
- Many databases use **preallocated** logs
- Preallocated log files have stable layouts
  - No resizing/permission changes.
- **Éxitos**: Uses eBPF to bypass software layers
  - Maps file offsets to disk block LBAs efficiently
  - Redirects I/O via ioctl to SSD driver
- **Benefits:**
  - No app/kernel intrusive changes; POSIX-compatible
  - Works with SATA/NVMe SSDs.
  - 2.3× faster than vanilla I/O stack

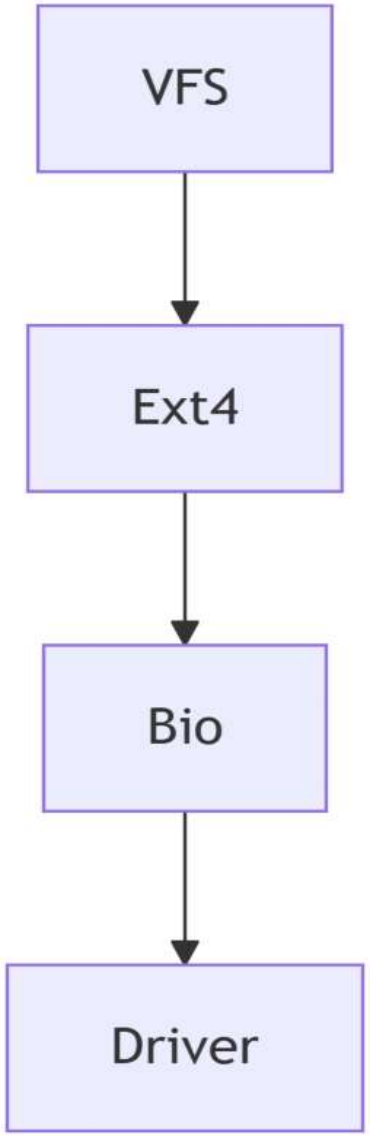
Outline

- Introduction
- Background
- Motivation
- Design
- Experiments
- Conclusion

Background

上海科技大学
ShanghaiTech University

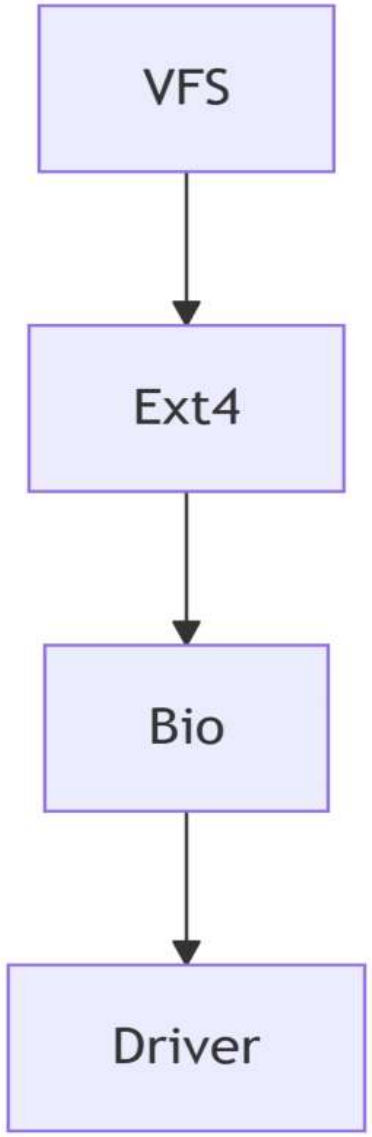
- **Software tax:** OS/filesystem overhead during I/O
- **Preallocation for** database logs:
  - Reserves contiguous disk blocks upfront
  - Avoids runtime allocation/journaling



Background

上海科技大学
ShanghaiTech University

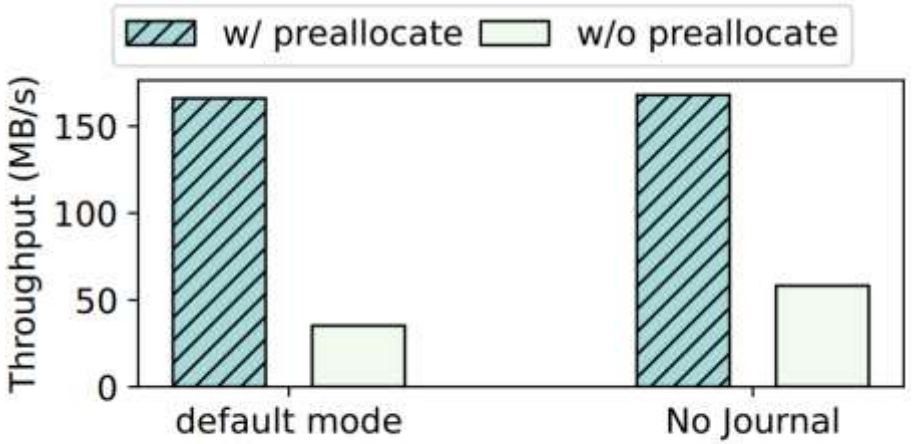
- **Software tax:** OS/filesystem overhead during I/O
- **Preallocation for** database logs:
  - Reserves contiguous disk blocks upfront
  - Avoids runtime allocation/journaling
- **Key insight**:
  - Preallocated log files = **stable layouts**
- **eBPF** (OS kernel tool):
  - Safely runs custom code in kernel
  - Hooks syscalls (e.g., file write, fdatasync)



Outline

- Introduction
- Background
- Motivation
- Design
- Experiments
- Conclusion

Motivation - Why Optimize Logging?



(a) The Impact of Pre-allocation

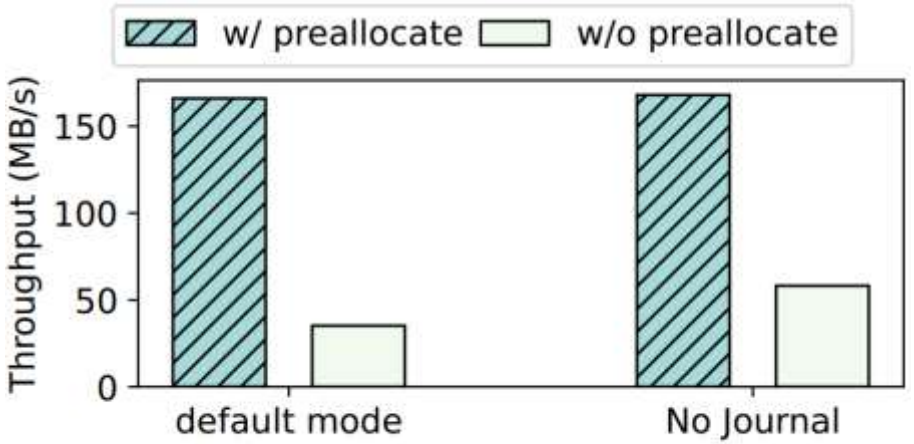
O1: Preallocation boosts performance

①Throughput ↑ **4.7×** on NVMe SSD

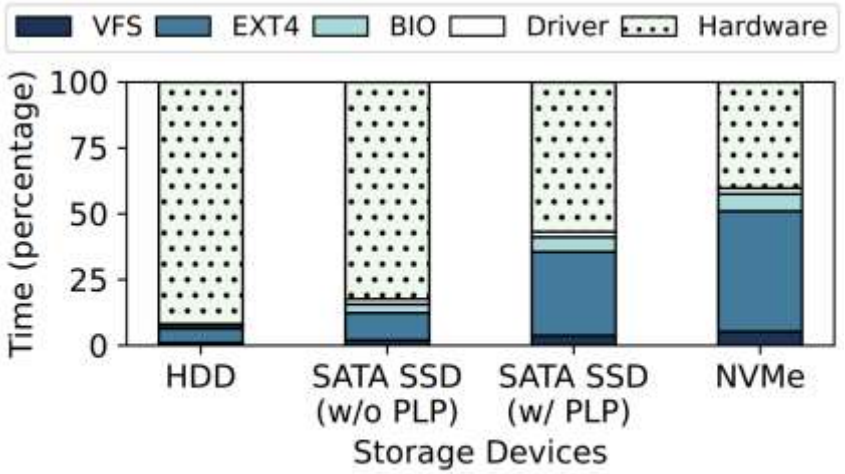
②Reason: removes ext4 journaling and fragmentation overhead

=>Databases use preallocated logs

Motivation - Why Optimize Logging?



(a) The Impact of Pre-allocation



60.2%

(b) The Breakdown Cost on the I/O Path

O1: Preallocation boosts performance

①Throughput ↑ **4.7×** on NVMe SSD

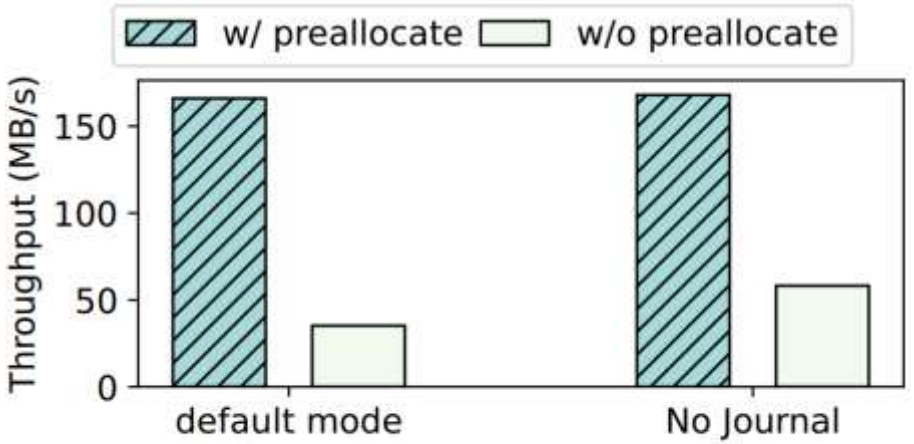
②Reason: removes ext4 journaling and fragmentation overhead

=>Databases use preallocated logs

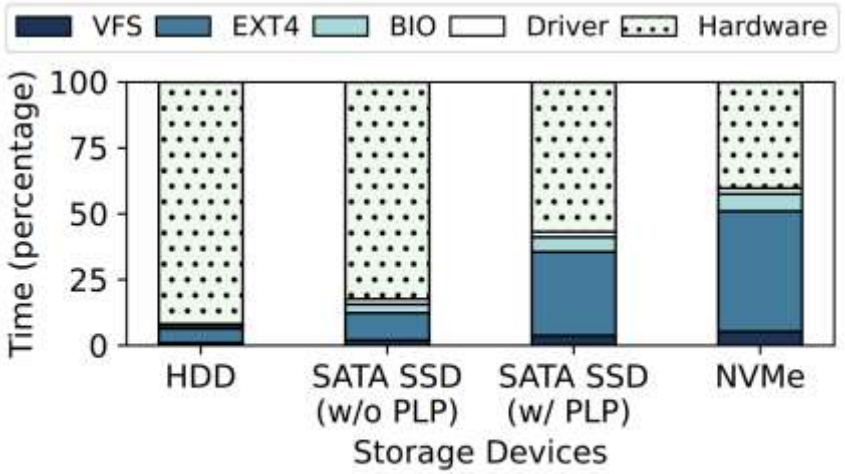
- **O2: But software tax still dominates:**
  - ①**60.2%** of I/O latency on fast NVMe SSDs for **software tax**

Motivation - Why Optimize Logging?

上海科技大学
ShanghaiTech University



(a) The Impact of Pre-allocation



46.3%

(b) The Breakdown Cost on the I/O Path

O1: Preallocation boosts performance

①Throughput ↑ **4.7×** on NVMe SSD

②Reason: removes ext4 journaling and fragmentation overhead

=>Databases use preallocated logs

O2: But software tax still dominates:

- ①**60.2%** of I/O latency on fast NVMe SSDs for **software tax**

②**Filesystem** => 46.3% of total time

立志成才报国裕民

Motivation - Why Not Existing Solutions?

上海科技大学
ShanghaiTech University

	No Hardware Change	Non-Intrusive Kernel	Transparent Support for DBs	Filesystem Compatibility	Supported SATA SSD
SPDK	√	√	x	x	√
NVMeDirect	√	x	x	x	x
Moneta-D	x	x	√	√	√
BypassD	x	x	√	√	x
Éxitos	√	√	√	√	√

O3: State-of-the-art limitations

SPDK/NVMeDirect: Break POSIX, need app rewrites I/O layer

BypassD/Moneta-D: Need intrusive kernel changes + special hardware

Motivation - Why Not Existing Solutions?

上海科技大学
ShanghaiTech University

	No Hardware Change	Non-Intrusive Kernel	Transparent Support for DBs	Filesystem Compatibility	Supported SATA SSD
SPDK	√	√	x	x	√
NVMeDirect	√	x	x	x	x
Moneta-D	x	x	√	√	√
BypassD	x	x	√	√	x
Éxitos	√	√	√	√	√

Éxitos opportunity:

Preallocated logs = **stable layouts for database logs in filesystems and disks**

→ Safe to bypass OS without intrusive changes in kernel and hardware

Outline

- Introduction
- Background
- Motivation
- Design
- Experiments
- Conclusion

Éxitos Architecture

上海科技大学
ShanghaiTech University

- **3** main components:

- ①Maco:
  - Map file offset → LBA during initial step

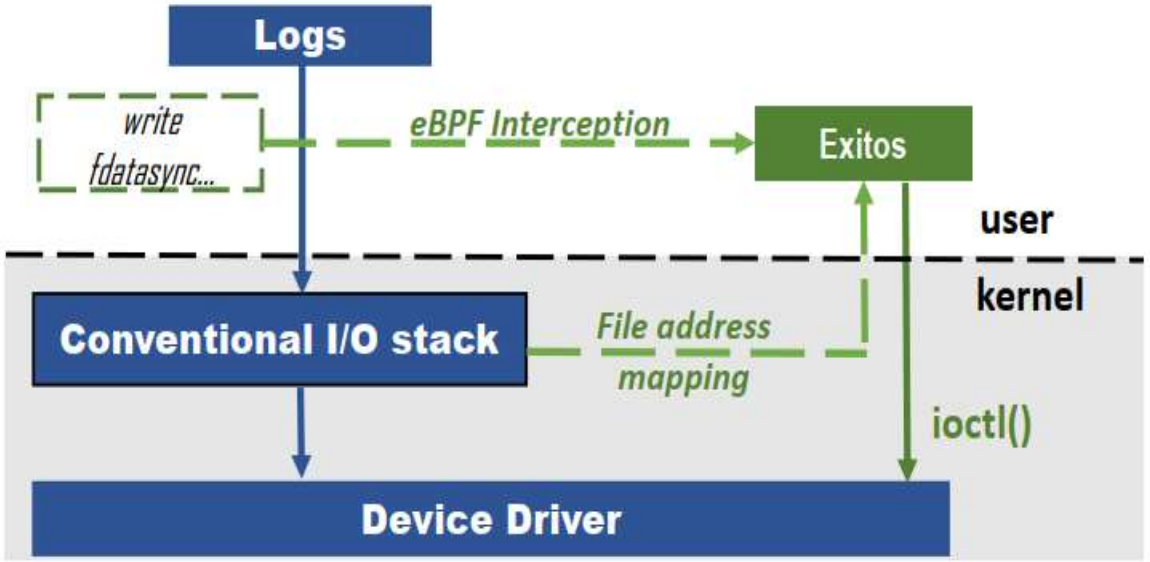


Figure 2: The Architecture of Éxitos

Éxitos Architecture

上海科技大学
ShanghaiTech University

- **3** main components:

- ①Maco:
  - Map file offset → LBA during initial step

- ② eBPF Hooks:
  - Trap write/fdatasync of the App process like OceanBase

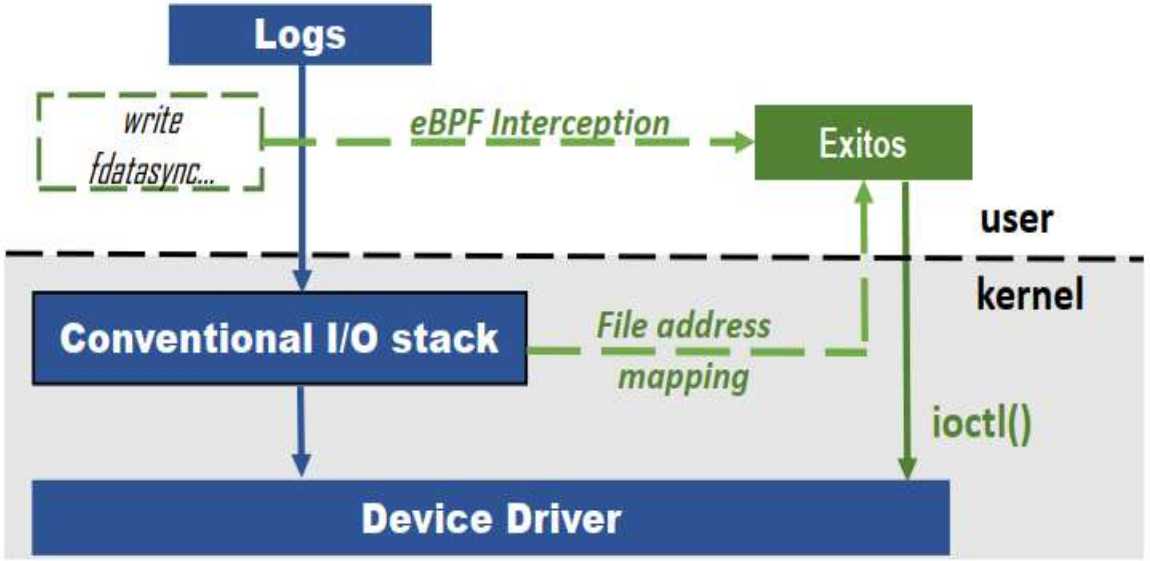


Figure 2: The Architecture of Éxitos

Éxitos Architecture

上海科技大学
ShanghaiTech University

- **3** main components:

- ①Maco:
  - Map file offset → LBA during initial step

- ② eBPF Hooks:
  - Trap write/fdatasync of the App process like OceanBase

- ③Direct Dispatch: ioctl to SSD driver

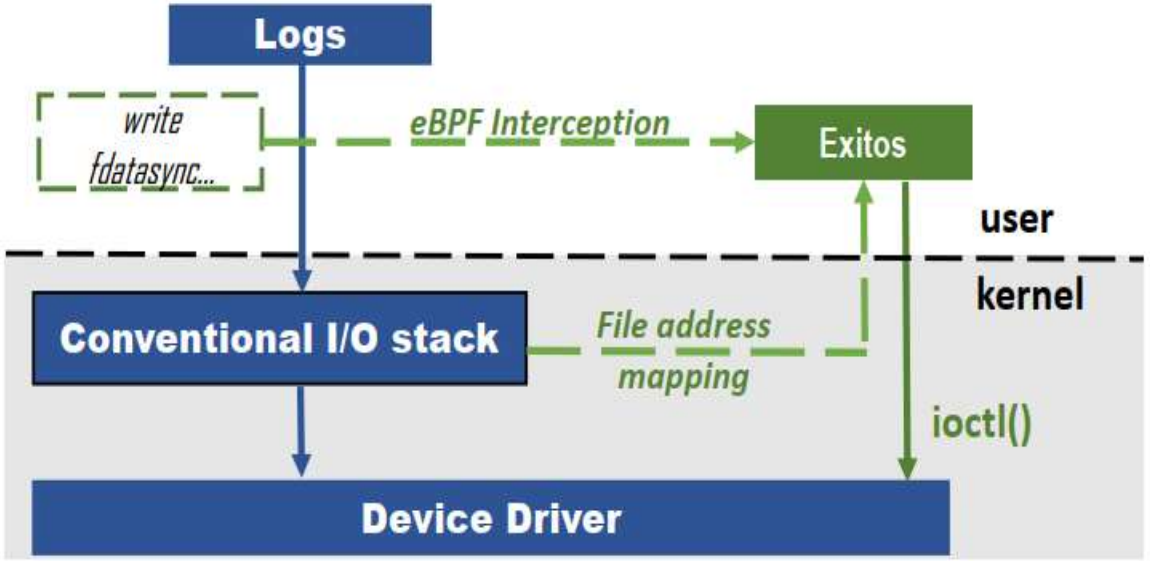


Figure 2: The Architecture of Éxitos

Metadata Magic with Maco

上海科技大学
ShanghaiTech University

Maco (Metadata Collector):

Stores **{file descriptor, offset → SSD LBA}** in eBPF map

Built when opening preallocated log files

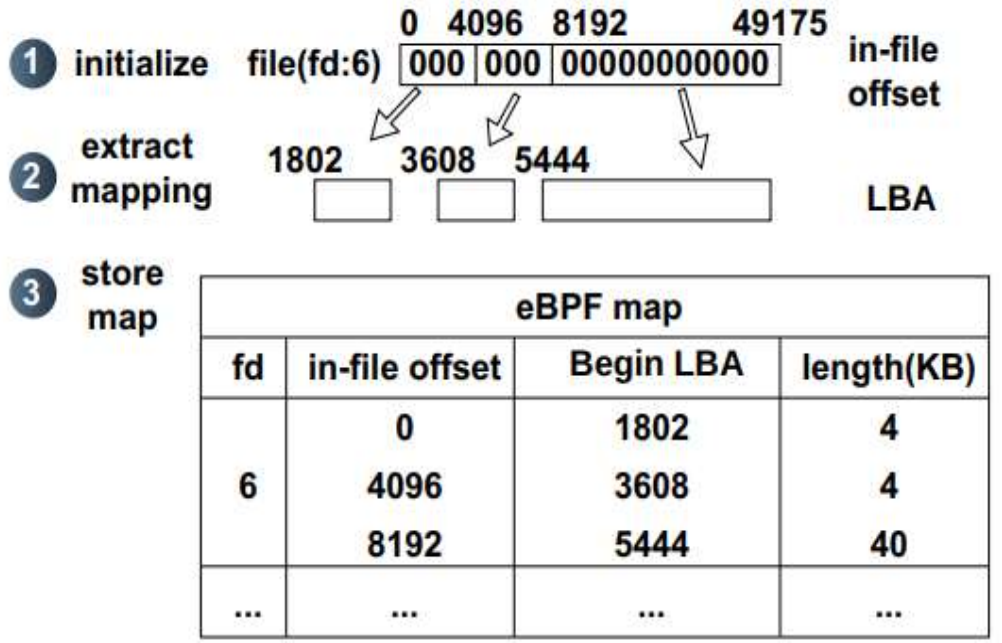


Figure 3: Address Mapping Extraction for WAL in Maco

Metadata Magic with Maco

上海科技大学
ShanghaiTech University

Maco (Metadata Collector):

Stores **{file descriptor, offset → SSD LBA}** in eBPF map

Built when opening preallocated log files

Time complexity?

O(1) because mapping is stable

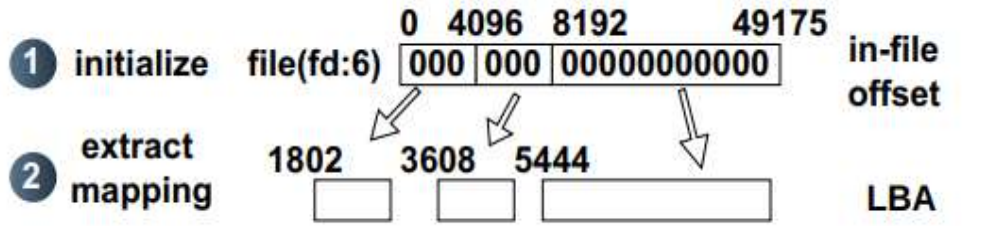
Additional memory usage?

Preallocation → contiguous LBAs → few records for one big file

Extra memory overhead is low

Crash-Safety?

Rebuilt from filesystem metadata after reboot



eBPF map			
fd	in-file offset	Begin LBA	length(KB)
6	0	1802	4
	4096	3608	4
	8192	5444	40
...	...	...	...

Figure 3: Address Mapping Extraction for WAL in Maco

Bypassing I/O & Permissions

上海科技大学
ShanghaiTech University

- **For write:**
  - Use Maco mapping → send data via ioctl to bypass software layers
- **For fdatasync:**
  - Send NVMe-flush (NVMe) command to ensure durability
- **Dual Permission Modes:**
  - Fast mode(Éxitos): Checks permissions only at file open
  - Strict mode(Éxitos-S) : Validates permissions per I/O request

Outline

- Introduction
- Background
- Motivation
- Design
- Experiments
- Conclusion

Evaluation Setup

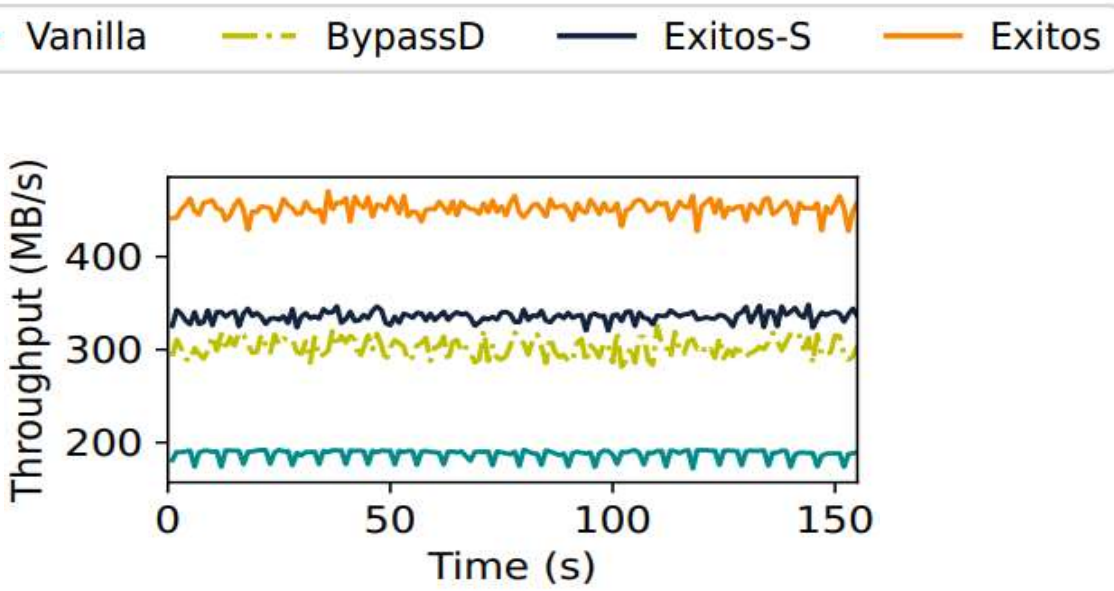
上海科技大学
ShanghaiTech University

- **Hardware:**
  - HP Z2 G4: Intel i9-9900K (16c), 64GB RAM
  - SSD: Samsung PM1725a (NVMe with PLP)
- **Software:**
  - Ubuntu 22.04.1, Linux 6.6.5
  - OceanBase v4.3.3
- **Baselines:**
  - **Vanilla:** Standard I/O stack
  - **BypassD:** With software-emulated IOMMU
  - **Exitos/Exitos-S**
- **Why BypassD?**
  - Only competitor with official **software simulation** for DBs
  - Others:
    - Kernel/database code intrusive changes
    - Special hardware
    - No implementation for the most of the DBs

Micro-Benchmarks - Fio Simulating Logging

We chose Fio to simulate logging behavior in the database as Micro-benchmarks.

- Test **(a): 4**KB block size writes + fdatasync
  - Éxitos: **2.4×** throughput vs. vanilla

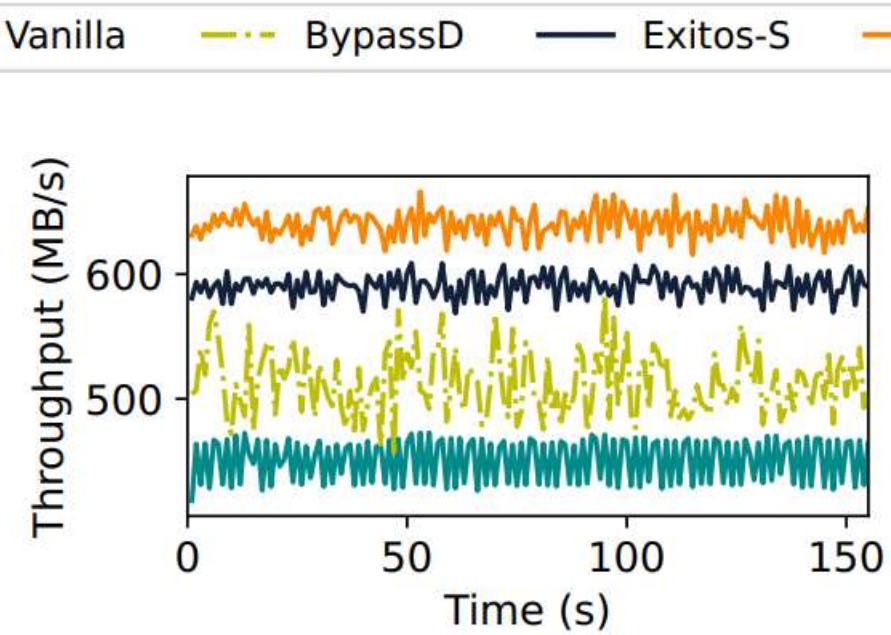


(a) Single-Threaded 4KB Writes.

Micro-Benchmarks - Fio Simulating Logging

We chose Fio to simulate logging behavior in the database as Micro-benchmarks.

- Test (a): **4**KB block size writes + fdatasync
  - Éxitos: **2.4×** throughput vs. vanilla
- Test **(b)**: **16**KB block size writes + fdatasync
  - Éxitos still beats every baseline
  - — even under the SSD throughput ceiling



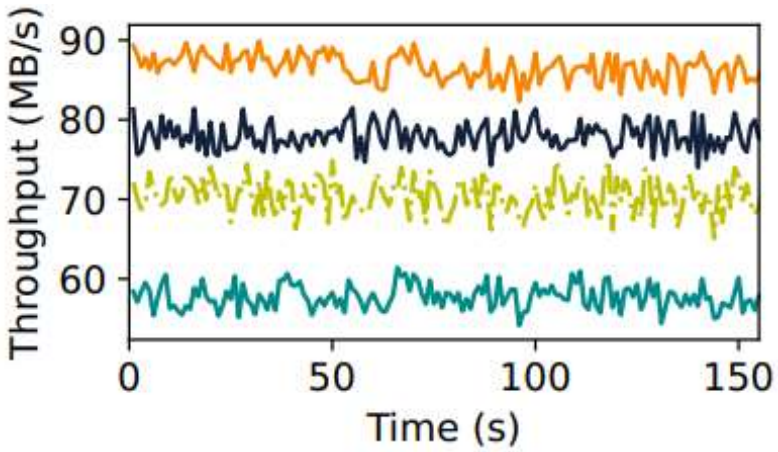
(b) Single-Threaded 16KB Writes.

Micro-Benchmarks - Fio Simulating Logging

上海科技大学
ShanghaiTech University

We chose Fio to simulate logging behavior in the database as Micro-benchmarks.

- Test (a): **4**KB block size writes + fdatasync
  - Éxitos: **2.4×** throughput vs. vanilla
- Test (b): **16**KB block size writes + fdatasync
  - Éxitos still beats every baseline
  - — even under the SSD throughput ceiling



(c) Multi-Threaded 4KB Writes.

- Test **(c): 4**KB block size writes + **16 threads** + fdatasync
  - Éxitos achieves near-linear scalability under 16 threads
  - Outperforms every competitor—zero hardware or kernel modifications required.

Macro-Benchmarks - OceanBase with SysBench

上海科技大学
ShanghaiTech University

Real production workload

Measured directly on OceanBase under three OLTP workloads:

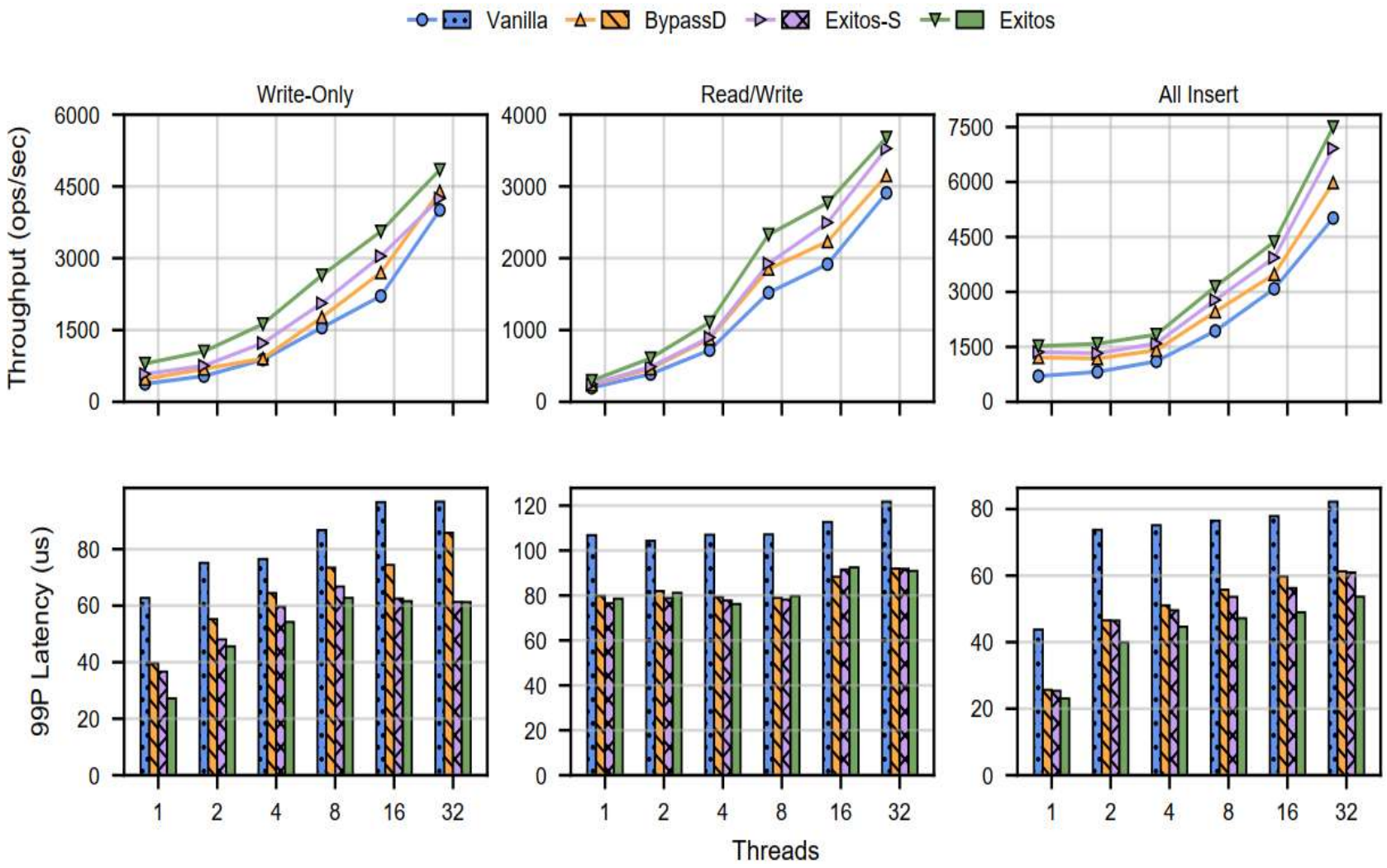
- Write-Only
- Read/Write ·
- All-Insert

Throughput

• 1 thread (write-only):

2.3 × more ops than vanilla

+31 % average vs. BypassD



立志成才报国裕民

Macro-Benchmarks - OceanBase with SysBench

上海科技大学 ShanghaiTech University

Scalability

near-linear scalability to **32 threads**

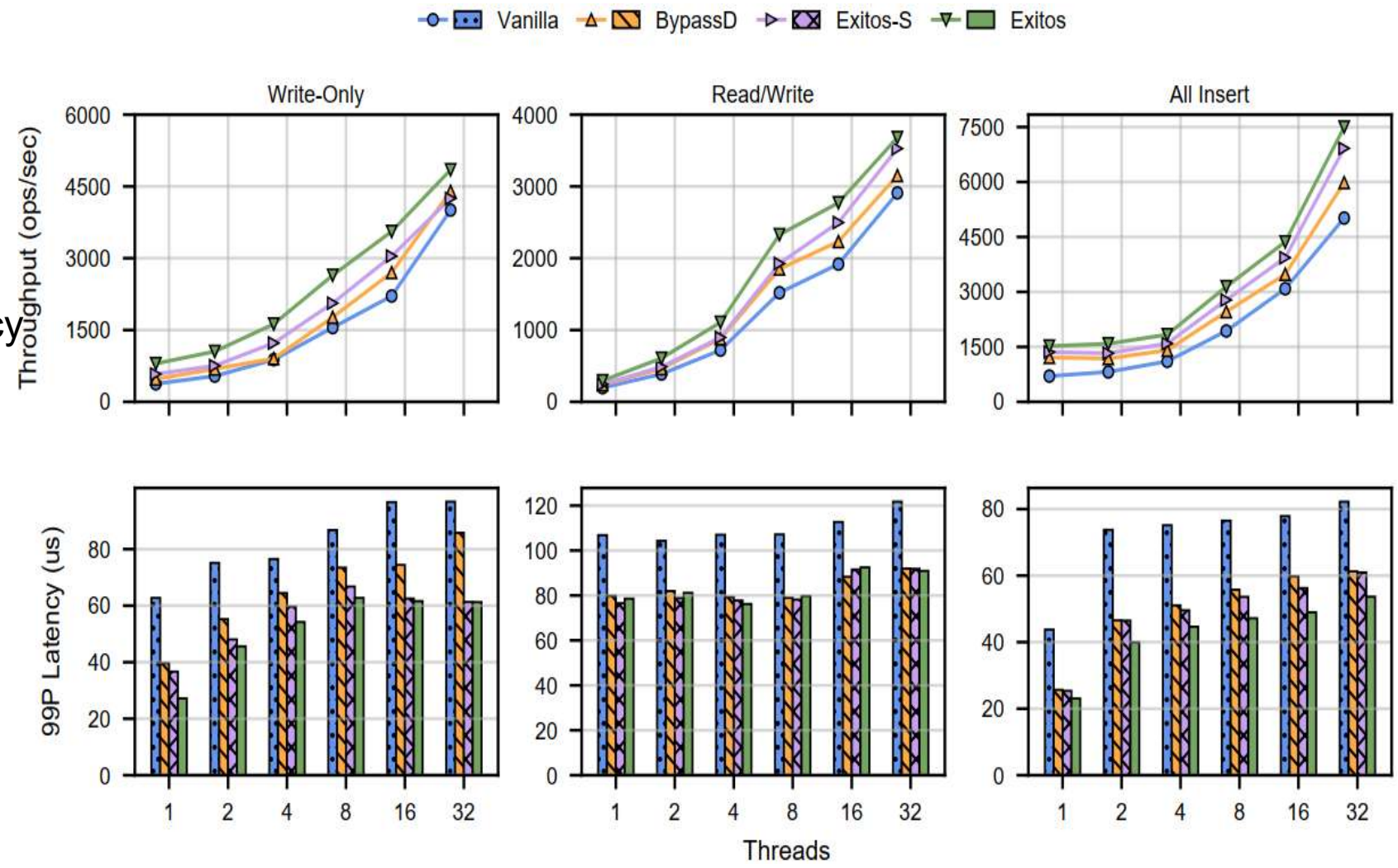
Latency

32 clients: **-62 %** 99-percentile latency

Strict-mode

Only a small overhead for

Éxitos -S (Strict mode)



立志成才报国裕民

Outline

- Introduction
- Background
- Motivation
- Design
- Experiments
- Conclusion

Conclusion - Exitos: Zero-Tax Logging

上海科技大学
ShanghaiTech University

- **Problem**: Logging I/Os bottlenecked by 60.2% software tax on fast SSDs.
  - Solution: Exitos – eBPF-driven shortcut for preallocated logs:
  - Bypasses OS layers via stable LBA mapping (Maco).
  - Direct ioctl dispatch to SSD.

Conclusion - Exitos: Zero-Tax Logging

- **Problem**: Logging I/Os bottlenecked by 60.2% software tax on fast SSDs.
  - Solution: Exitos – eBPF-driven shortcut for preallocated logs:
  - Bypasses OS layers via stable LBA mapping (Maco).
  - Direct ioctl dispatch to SSD.

- **Impact**:

  - 2.3× faster OceanBase throughput.
  - Software tax slashed to 4.1%.
  - No app/kernel/hardware changes

- **Future**: Adapt to other stable-file workloads.

Q&A

上海科技大学
ShanghaiTech University

- Thanks for your time.