MultiLanes: Providing Virtualized Storage for OS-level Virtualization on Many Cores

Junbin Kang, Benlong Zhang, Tianyu Wo, Chunming Hu, and Jinpeng Huai

Beihang University

夏飞
20140904

Outline

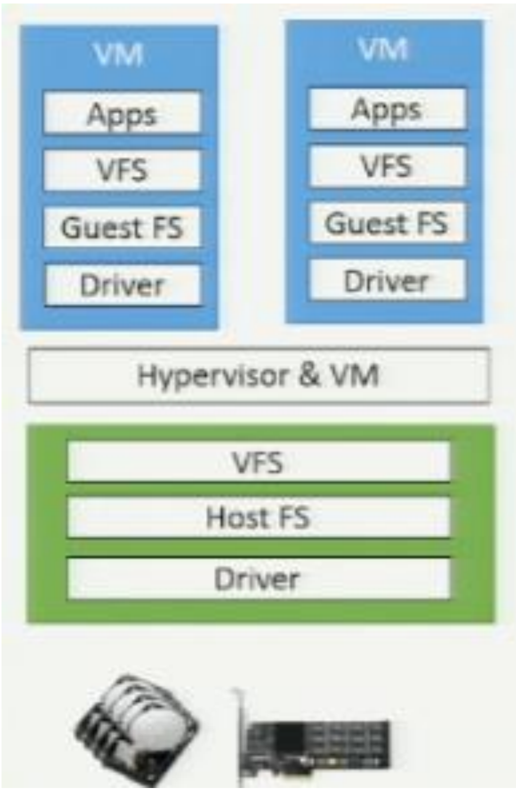
- Background
- MultiLanes Design
- Evaluation
- Related Work
- Conclusion

Background

- Many-core architecture
  - Common in modern processor
  - Requests consolidation for high performance
- Virtualization
  - Efficient method to improve performance and utilization of hardware.
- Non-Volatile Memory
  - Higher percentage of software overhead

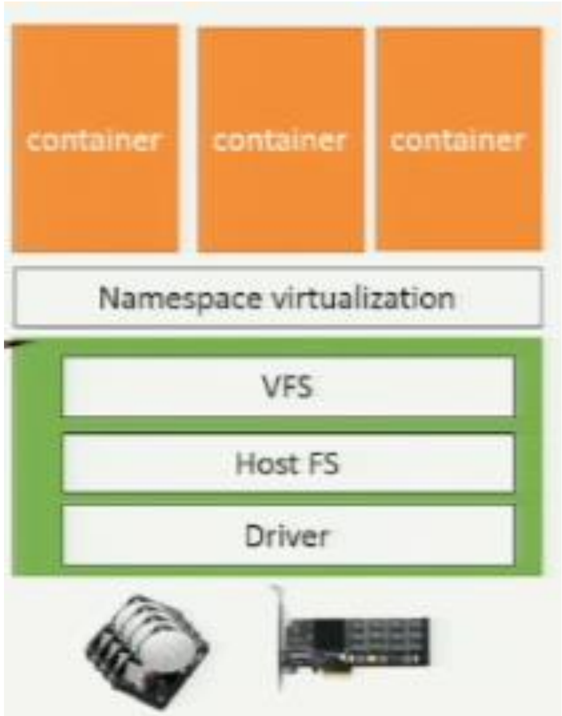
Hypervisor-based Virtualization vs. OS-level Virtualization

- Hypervisor-based Virtualization
  - E.g., Xen, KVM



Deep IO stack -> Poor performance

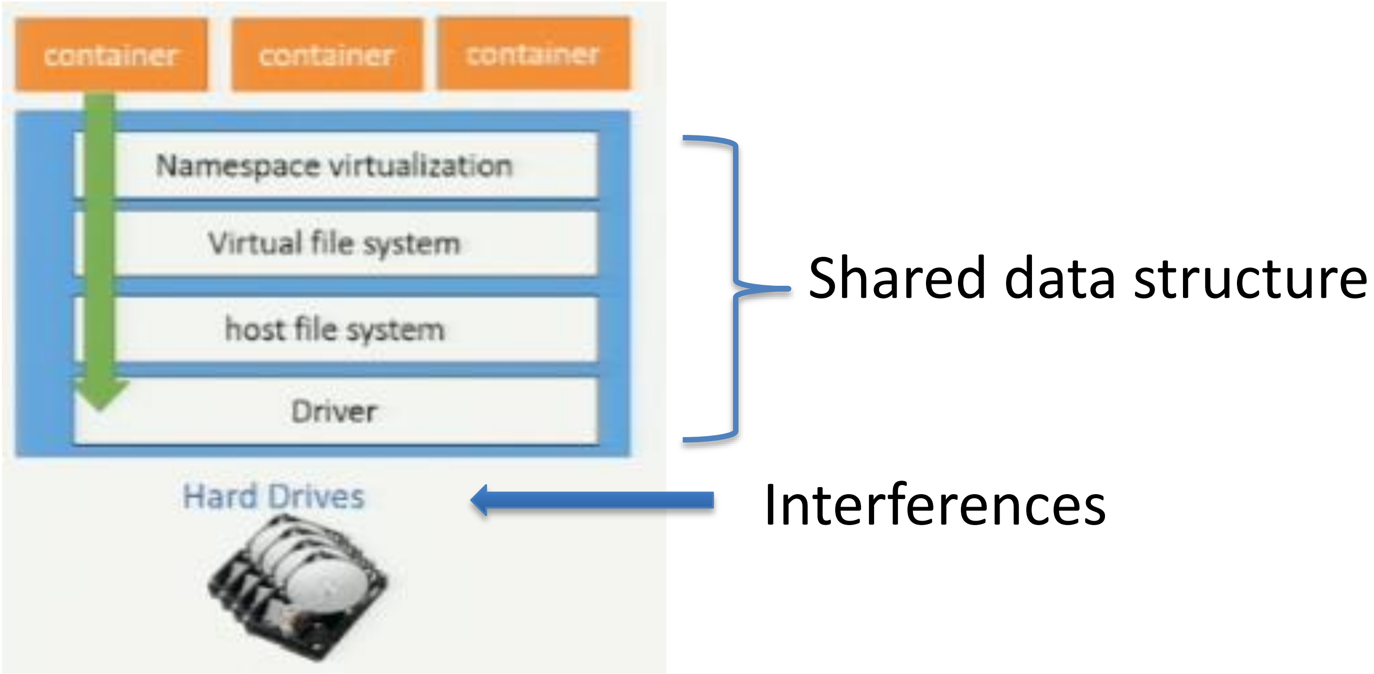
- OS-level Virtualization
  - E.g., VServer, LXC



Container: A virtualized environment (VE)

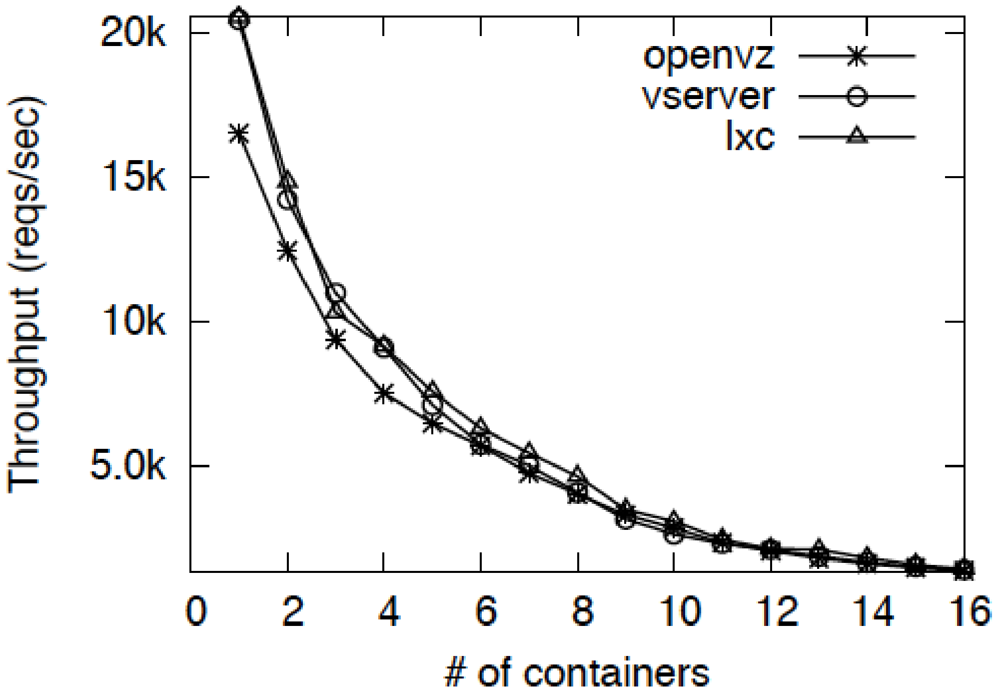
OS-level Virtualization

- Performance bottleneck of software
  - Especially for NVM-based fast storage



Motivation

- Scalability issues
  - FS: Ext3



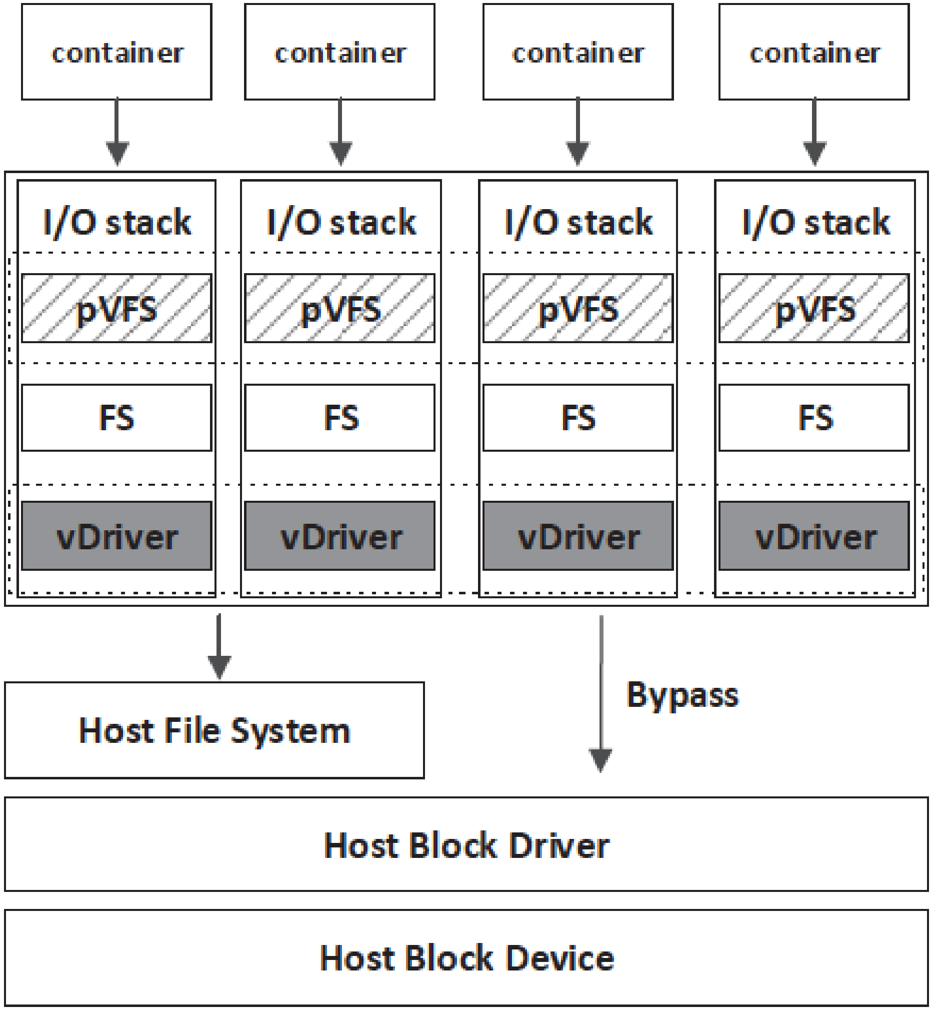
MultiLanes Design

- MultiLanes
  - A storage system for OS-level virtualization that addresses the I/O performance interference between multiple VEs on many cores.

- Design Goals:
  - Simple, self-contained, transparent to applications and FS
  - Good scalability
  - Low virtualization overhead on fast storage

MultiLanes Design

- MultiLanes Architecture



Container: a set of processes (actually)

pVFS: partitioned VFS

Different FSs

vDriver: virtualized device driver

Virtualized Storage: a file in host FS

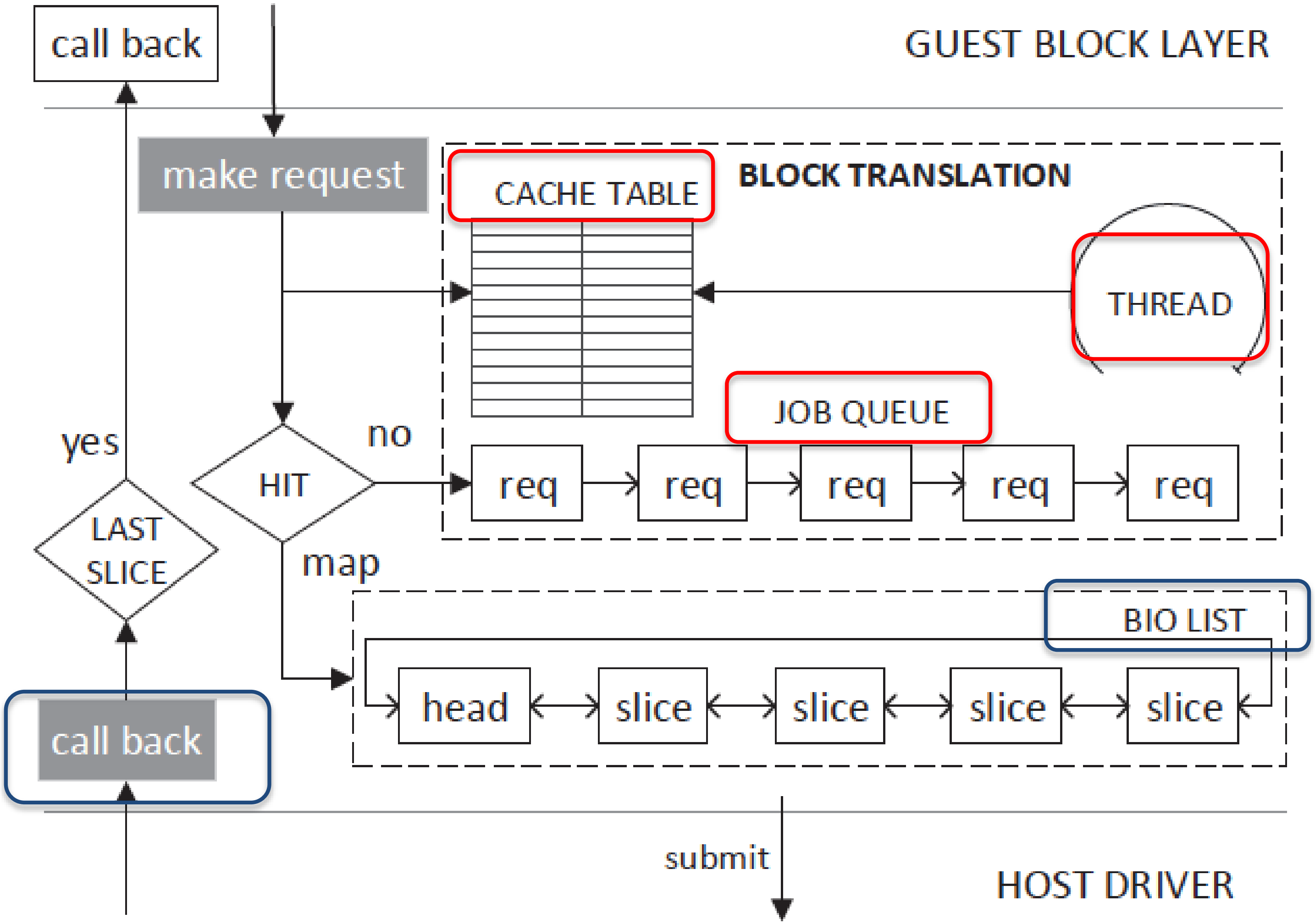
Virtualized Storage

- Traditional OS-level virtualization
  - Stores VM’s data on the host FS directly
  - Results in contention on shared data structure and locks
- MultiLanes
  - Maps a regular file in host FS as a virtualized block device
  - Incurs overhead

vDriver

- Key work
  - Mapping virtualized resources to physical ones.
- Two components
  - Block translation: maps a logical block of a file to the physical blocks on the host device
  - Request handling: handles IO requests of virtualized device
    - A single IO request of virtualized device may be translated into multiple IO requests of host device.

vDriver Structure



Slice: a IO request on the host block device.

Block Translation

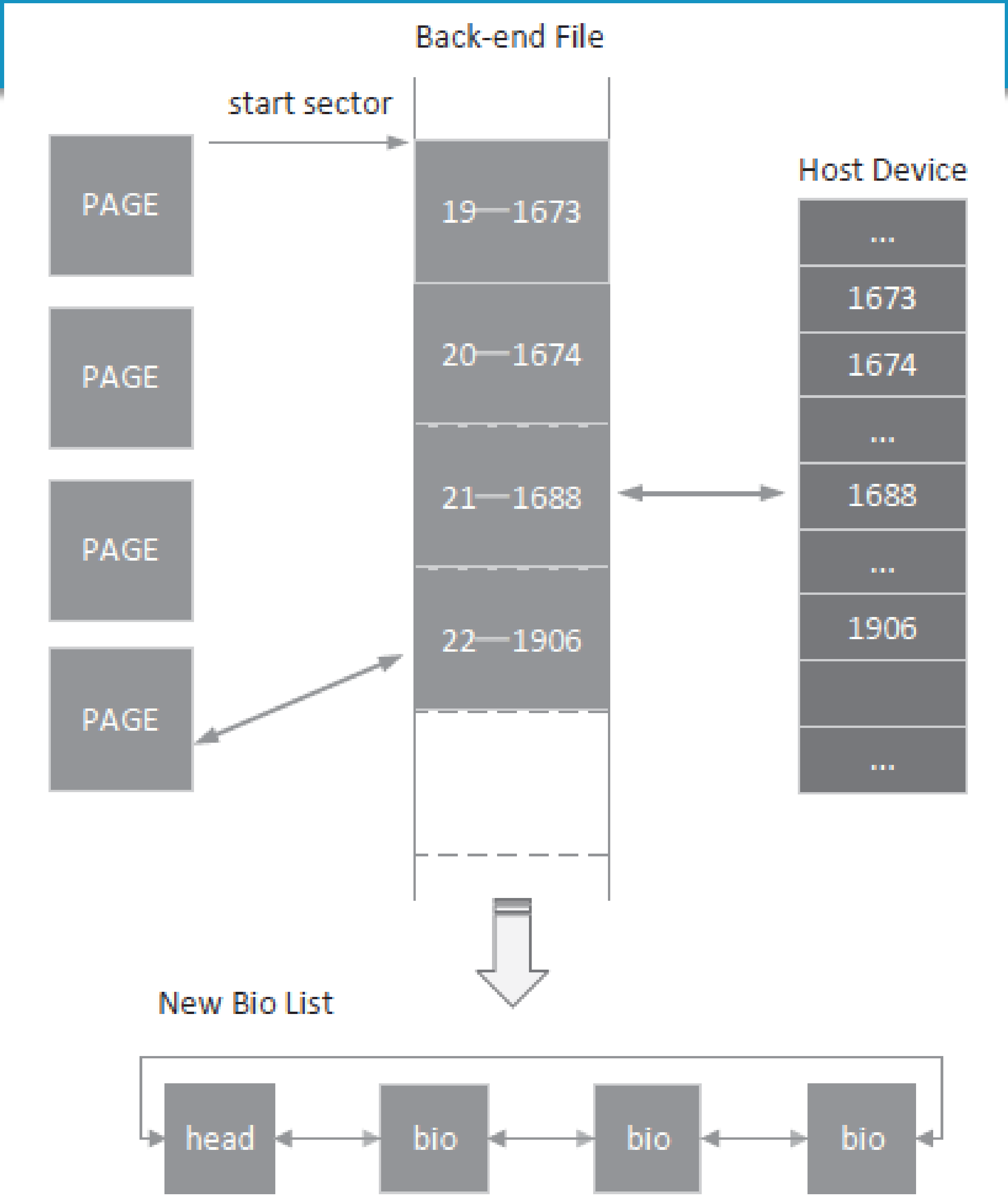
- Cache Table
  - Maintains the mapping between logical blocks and physical blocks
- Job Queue
  - Stores translation job
- Translation Thread
  - Block translation
    - Invokes the mapping interface of host FS to get the target physical block
    - Stores a new mapping entry in the cache table
    - Wakes up the container thread

Request Handling

- Slice
  - A new block I/O request on the host block device
- BIO list
  - Doubly-linked list
  - Each slice is submitted to host device driver in sequence
- I/O completion
  - Offer a I/O completion callback to the host driver

Example

- Request Mapping



Partitioned VFS

- Hot VFS Locks

#	Hot VFS Locks	Hot Invoking Functions
1	inode_hash_lock	insert_inode_locked() __remove_inode_hash()
2	dcache_lru_lock	dput() dentry_lru_prune()
3	inode_sb_list_lock	evict() inode_sb_list_add()
4	rename_lock	write_seqlock()

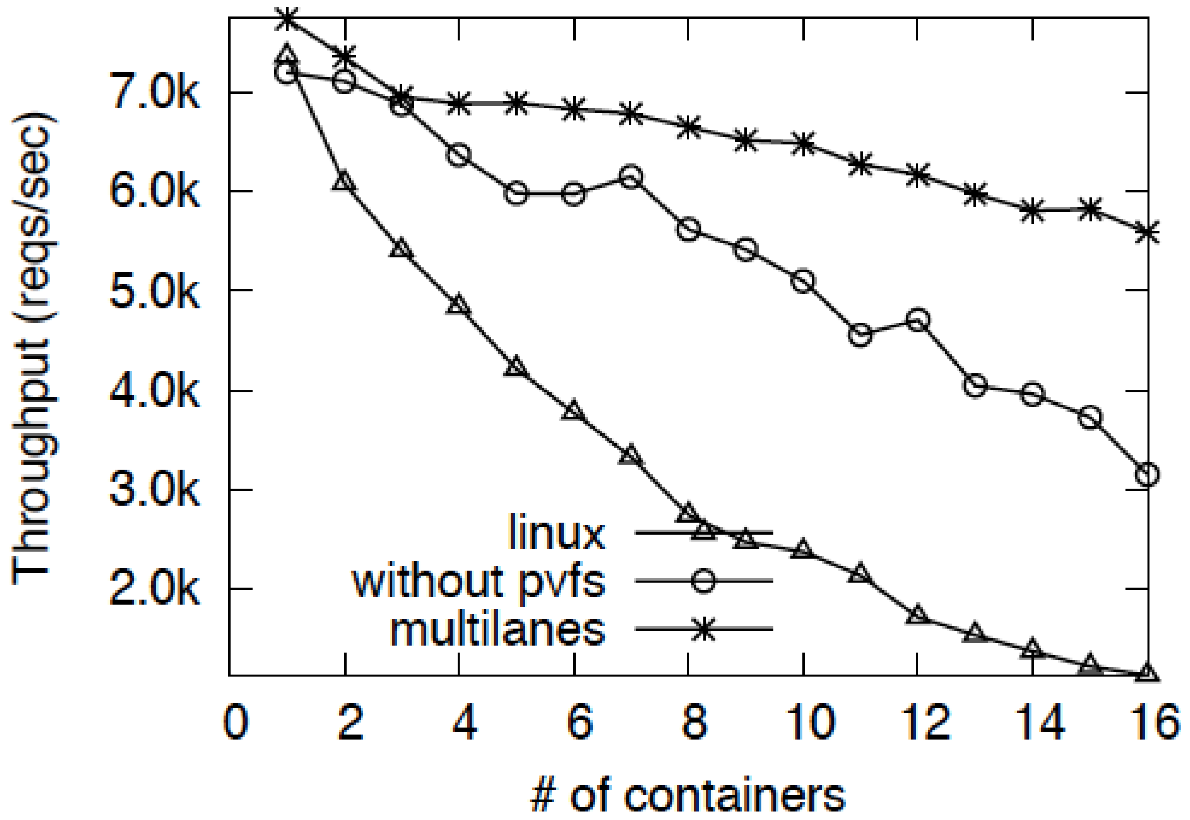
- Method in MultiLanes
  - Allocate an inode hash table and a dentry hash table for each container
  - Use separate locks for guest FS

Evaluation

- Experimental Setup
  - Intel 16-core machine with 64GB memory
  - 16 LXC containers
  - 40GB Ram disk as fast storage device
  - FS: Ext3, Ext4, XFS, Btrfs
- Microbenchmarks
  - Ocrd: runs 64K transactions (create, rename, and delete files)
  - IOzone: writes 4KB data to a file that ends up with 256MB size
- Macrobenchmarks
  - Filebench: mail server, file server
  - MySQL

Results

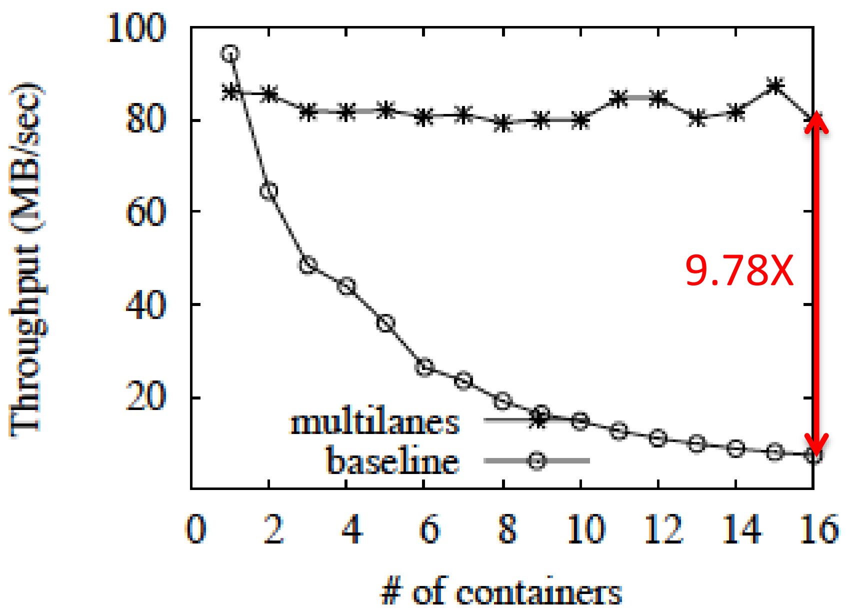
- Ocrd
  - Ext3



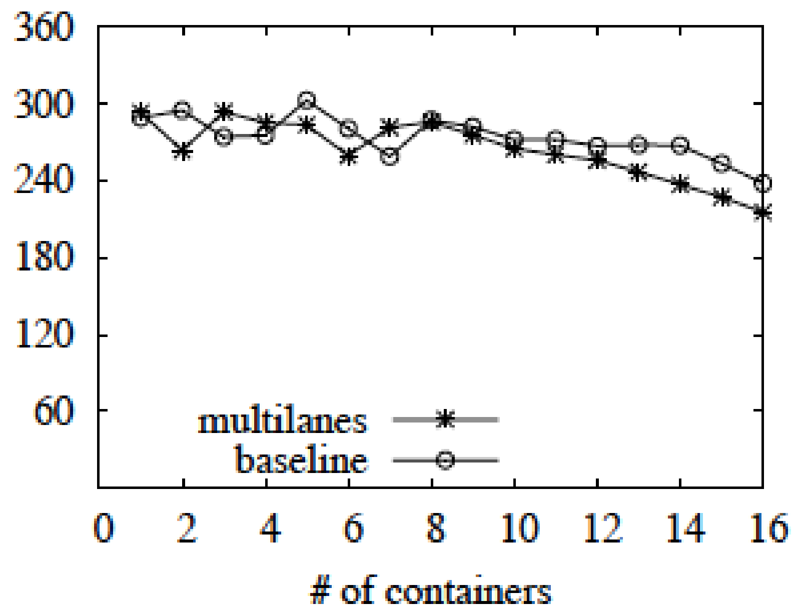
Lock	Ext3	Ext4	XFS	Btrfs
inode_hash_lock	1092k	960k	114k	228k
dcache_lru_lock	1023k	797k	583k	5k
inode_sb_list_lock	239k	237k	144k	106k
rename_lock	541k	618k	446k	252k

Table 3: Contention Bounces. *The table shows the contention bounces using MultiLanes without pVFS.*

- IOzone
  - Sequential writes



(a) Buffered write on Ext3

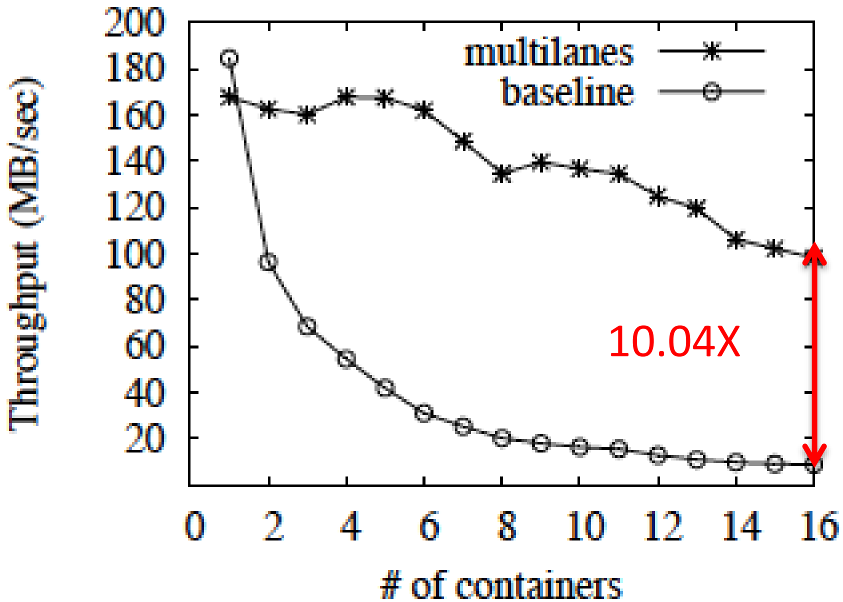


(c) Buffered write on XFS

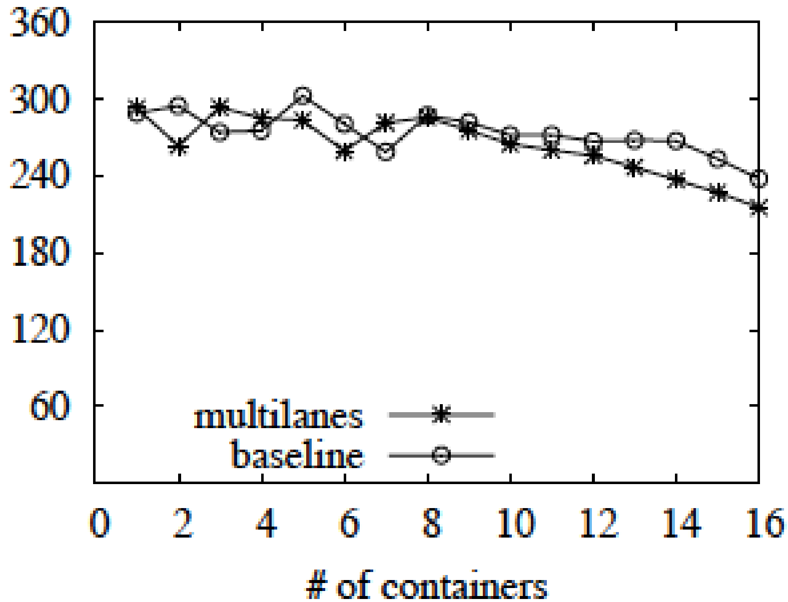
XFS: delays block allocation and metadata journaling

Results

- IOzone
  - Random writes



(a) Buffered write on Ext3



(c) Buffered write on XFS

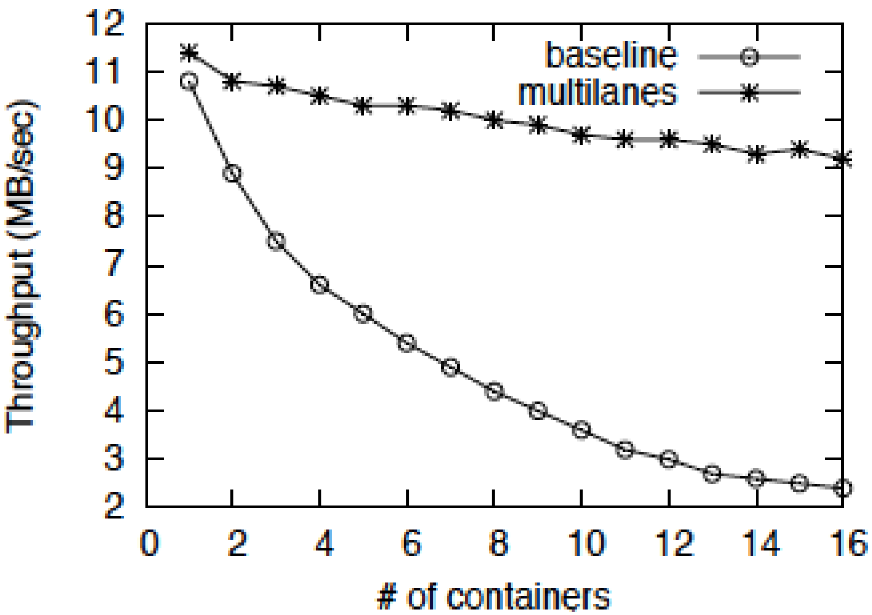
XFS: competitive

Results

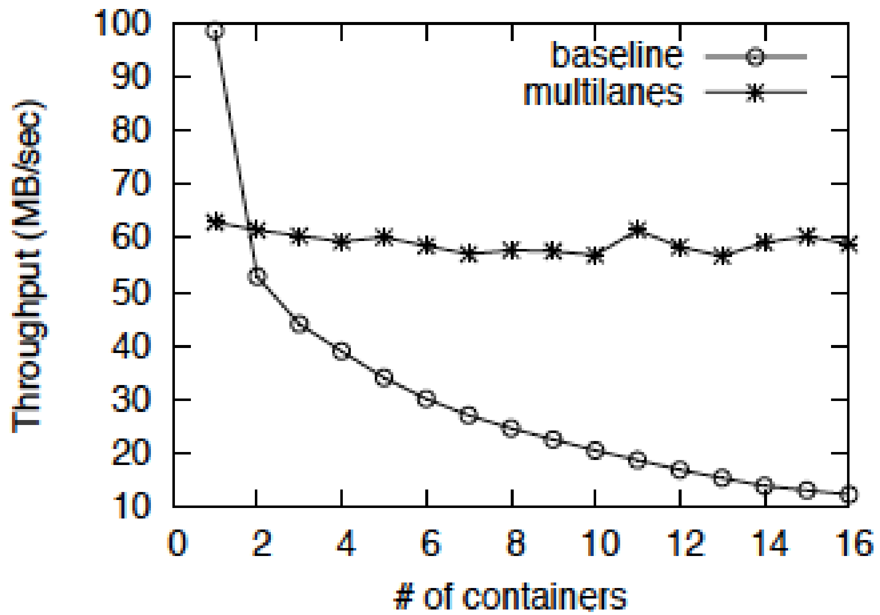
- Filebench

Workload	# of Files	File Size	I/O Size	Append Size
Varmail	1000	16KB	1MB	16KB
Fileserver	2000	128KB	1MB	16KB

Table 4: **Workload Specification.** *This table specifies the parameters configured for Filebench Varmail and Fileserver workloads.*



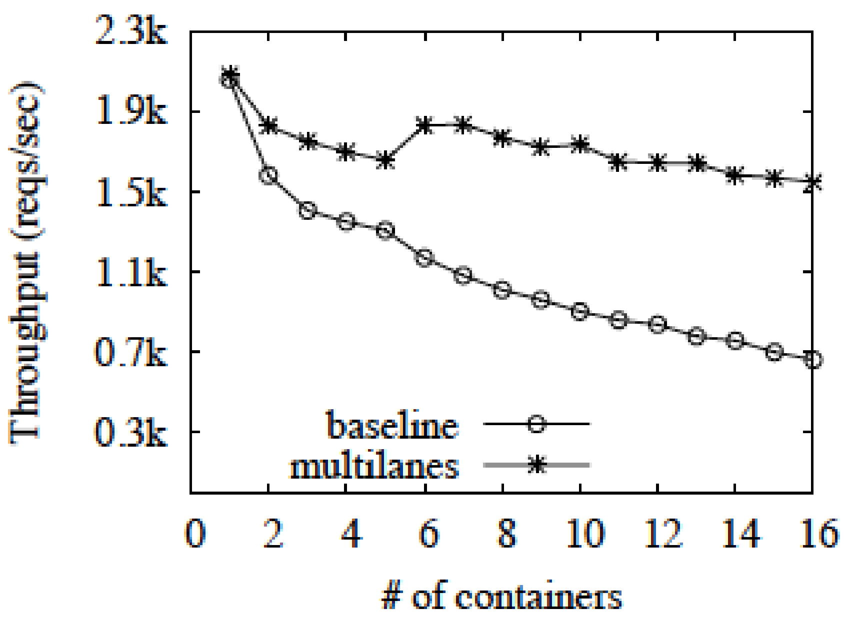
(a) Mail server on Ext3



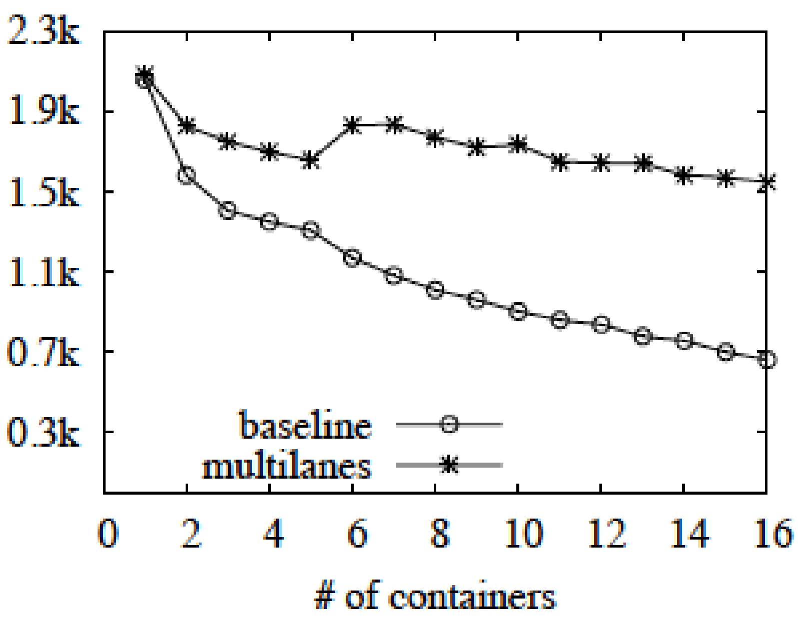
(e) File server on Ext3

Results

- MySQL



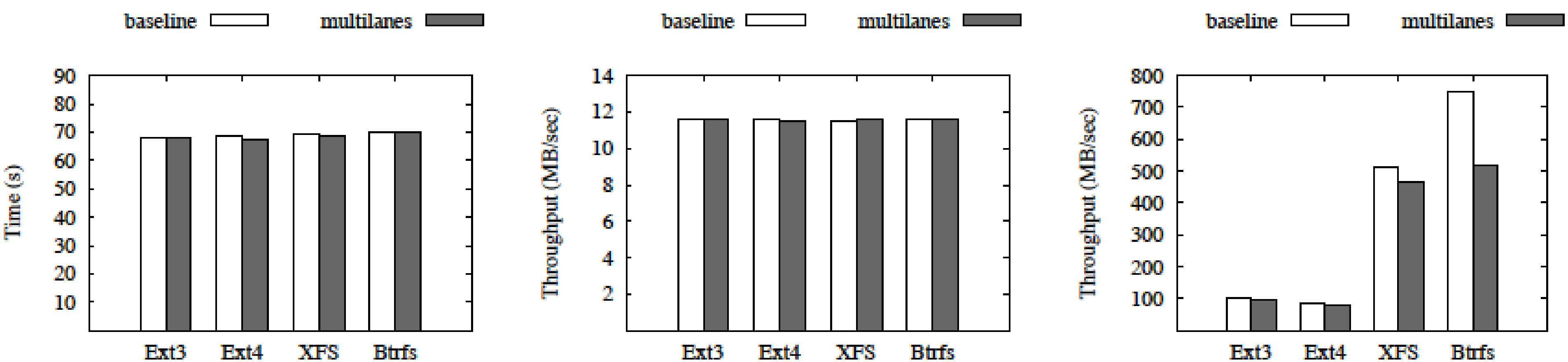
(a) MySQL on Ext3



(b) MySQL on Ext4

Overhead Analysis

- Single container
- Benchmarks
  - Apache Build: I/O less intensive
  - Webserver: read intensive
  - Streamwrite: write intensive



(a) Apache Build (b) Webserver (c) Streamwrite

Related Work

- Performance isolation
  - Space partitioning or time multiplexing hardware resources (e.g., CPU, memory, disk)
    - E.g., VSever (allocating and scheduling physical resources), Cgroup (limit, account, and isolate resource usage of process groups)
- Kernel scalability
  - Partitioning hardware resources
    - Hive , Barrelfish (multi-kernel model)
  - Virtualization layer
    - Diso (running multiple VMs on shared-memory multiprocessors)
- Device virtualization
  - Device emulation, e.g., network cards, SCSI devices
  - Para-virtualization, e.g., KVM, Xen
  - Direct device assignment

Conclusion

- OS-level virtualization suffers from storage performance interference, especially for fast storage device.
- MultiLanes
  - Provides an isolate I/O stack to eliminate contentions between multiple VEs on many cores.