A Preliminary Evaluation on Many-to-one Virtualization

Jin Zhang, Zhuocheng Ding, Yubin Chen,
Zhengwei Qi, Haibing Guan

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

ACM TURC 2019

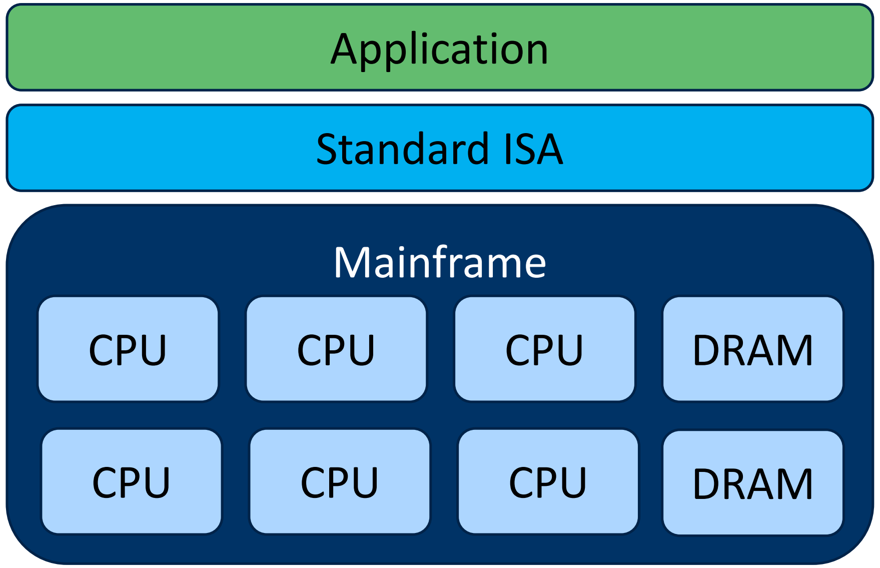
This work is supported by the National Key Research & Development Program of China 2016YFB1000502

CONTENTS

CONTENTS

Resource Aggregation

- **How can you aggregate resources in case of the end of Moore's Law?**

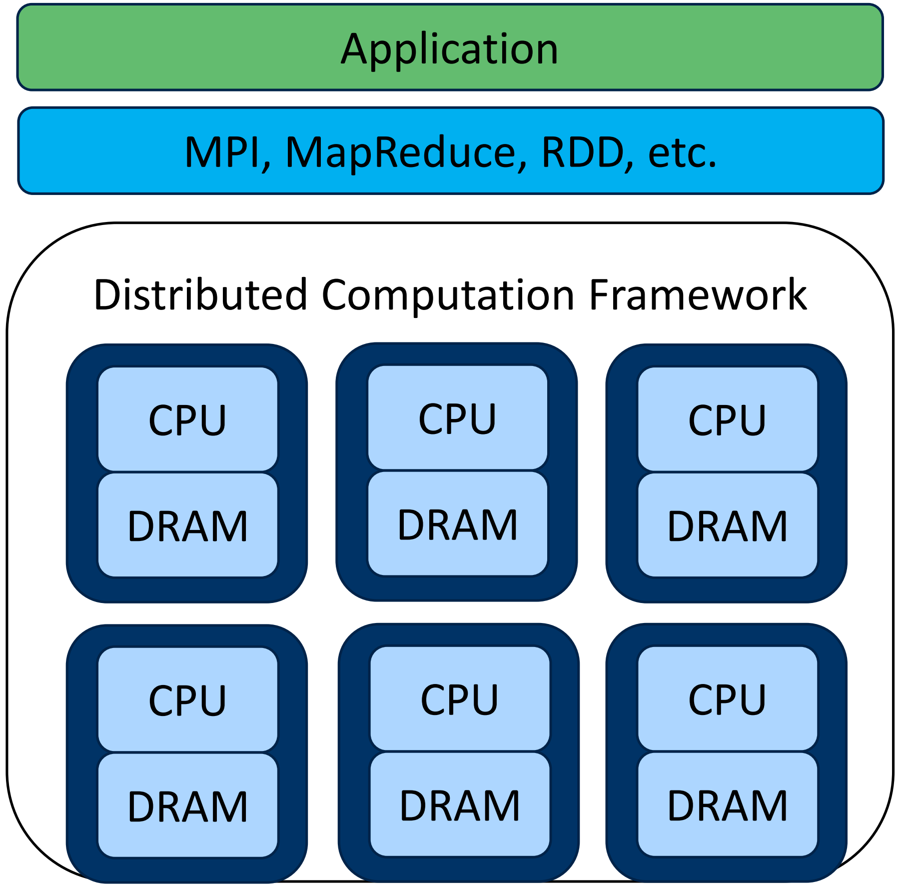


Scale-Up:
one machine with many CPUs, memory, etc.

+ No need to port existing software
– Expensive

Resource Aggregation

- **How can you aggregate resources in case of the end of Moore's Law?**

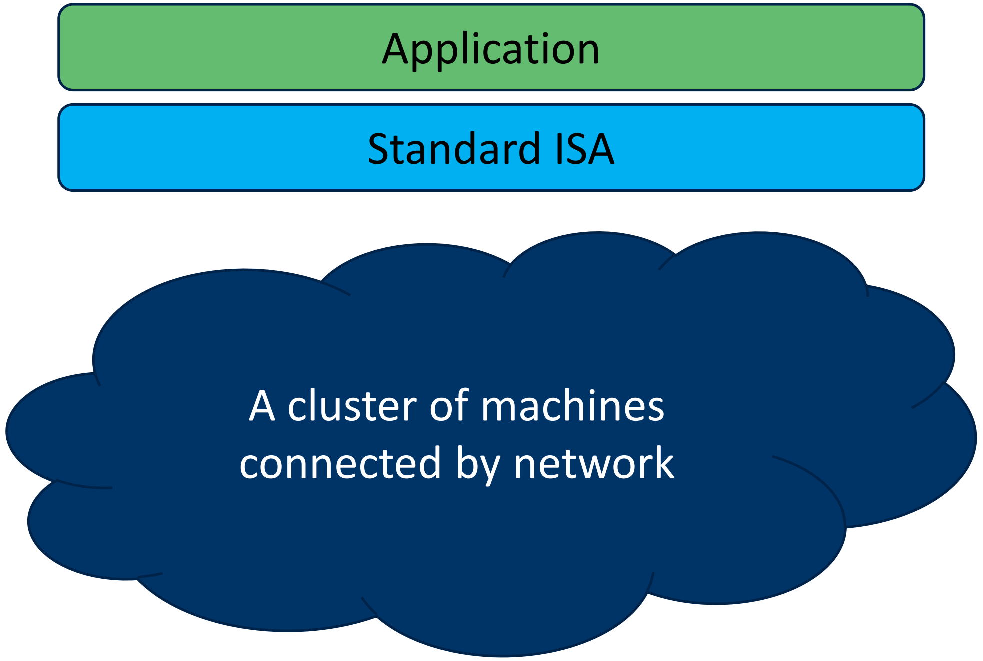


Scale-Out:
many machines with CPUs, memory, etc.

+ Affordable
− A huge engineering effort to port existing software

Single System Image

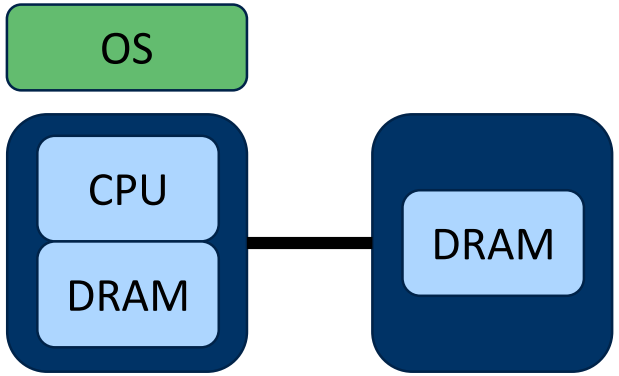
- **Single System Image (SSI)**
  - A single OS instance running on multiple machines.
  - No need to modify existing software.



Single System Image

- **Recently engaged by the high-speed network, such as RDMA, NVMe over Fabrics, Intel OmniPath, etc.**
- **Has different types.**

Memory aggregation

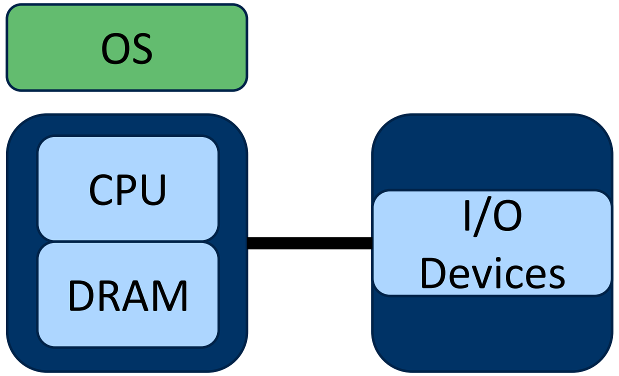


Infiniswap (NSDI'17),
Remote Regions (ATC'18),
Memory Blade (ISCA'09, HPCA'12)

Single System Image

- **Recently engaged by high-speed network, such as RDMA, NVMe over Fabrics, Intel OmniPath, etc.**
- **Has different types.**

I/O devices aggregation, including heterogeneous ones.

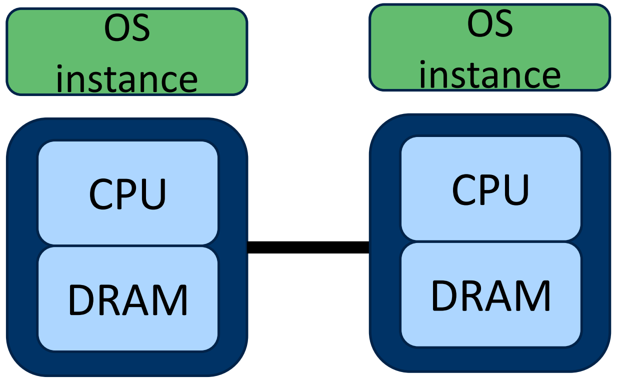


NFS, iSCSI, ReFlex (ASPLOS'17), Decibel (NSDI'17), PolarFS (PVLDB'18)

Single System Image

- **Recently engaged by high-speed network, such as RDMA, NVMe over Fabrics, Intel OmniPath, etc.**
- **Has different types.**

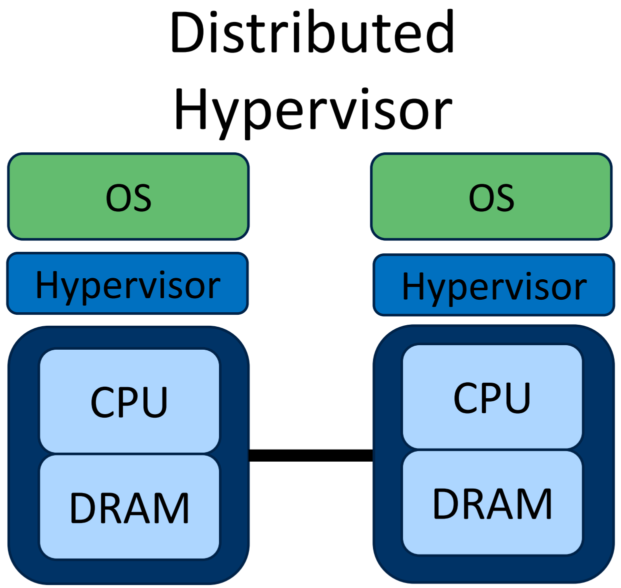
Distributed OS



HeliOS (SOSP’09), fos (SIGOPS’09),
Popcorn (EuroSys’15),
LegoOS (OSDI’18)

Single System Image

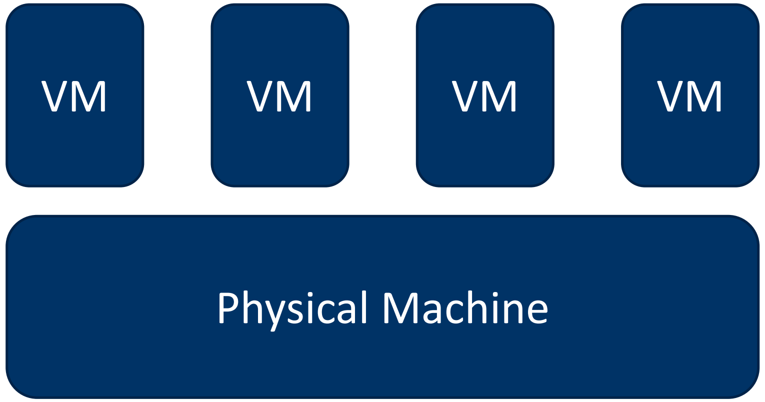
- **Recently engaged by high-speed network, such as RDMA, NVMe over Fabrics, Intel OmniPath, etc.**
- **Has different types.**



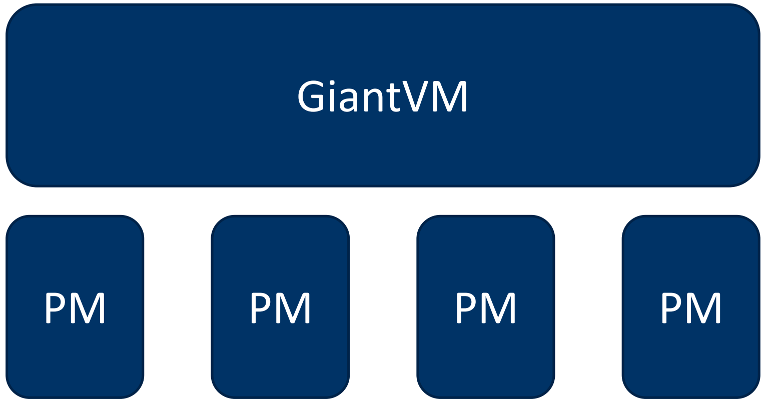
ScaleMP, TidalScale,
vNUMA (ATC'09),
GiantVM (Our work)

Rethink System Virtualization

One-to-many virtualization
(Classical virtualization)



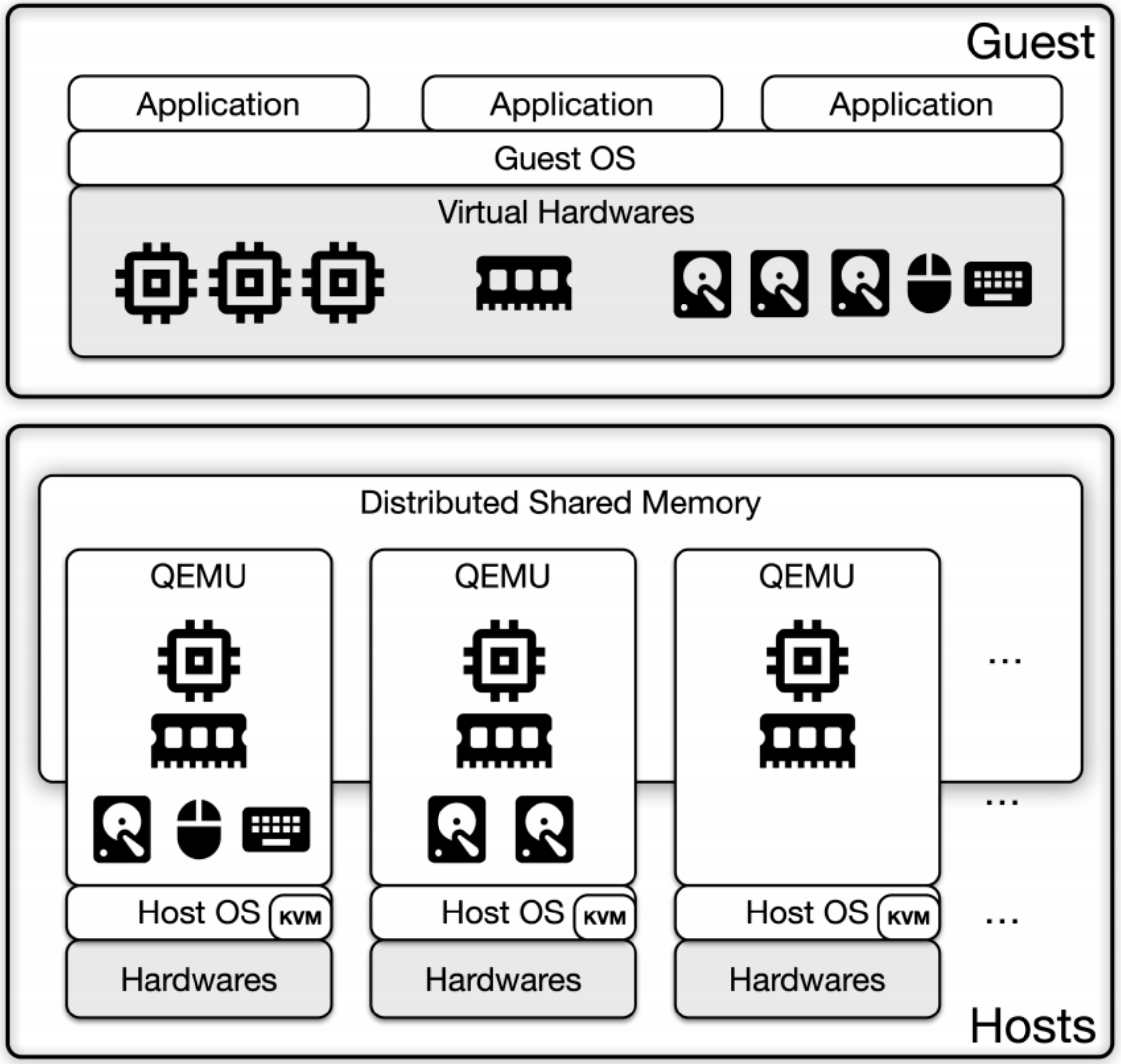
Our approach:
Many-to-one virtualization



Implement SSI by many-to-one virtualization

CONTENTS

GiantVM: Overview of Architecture



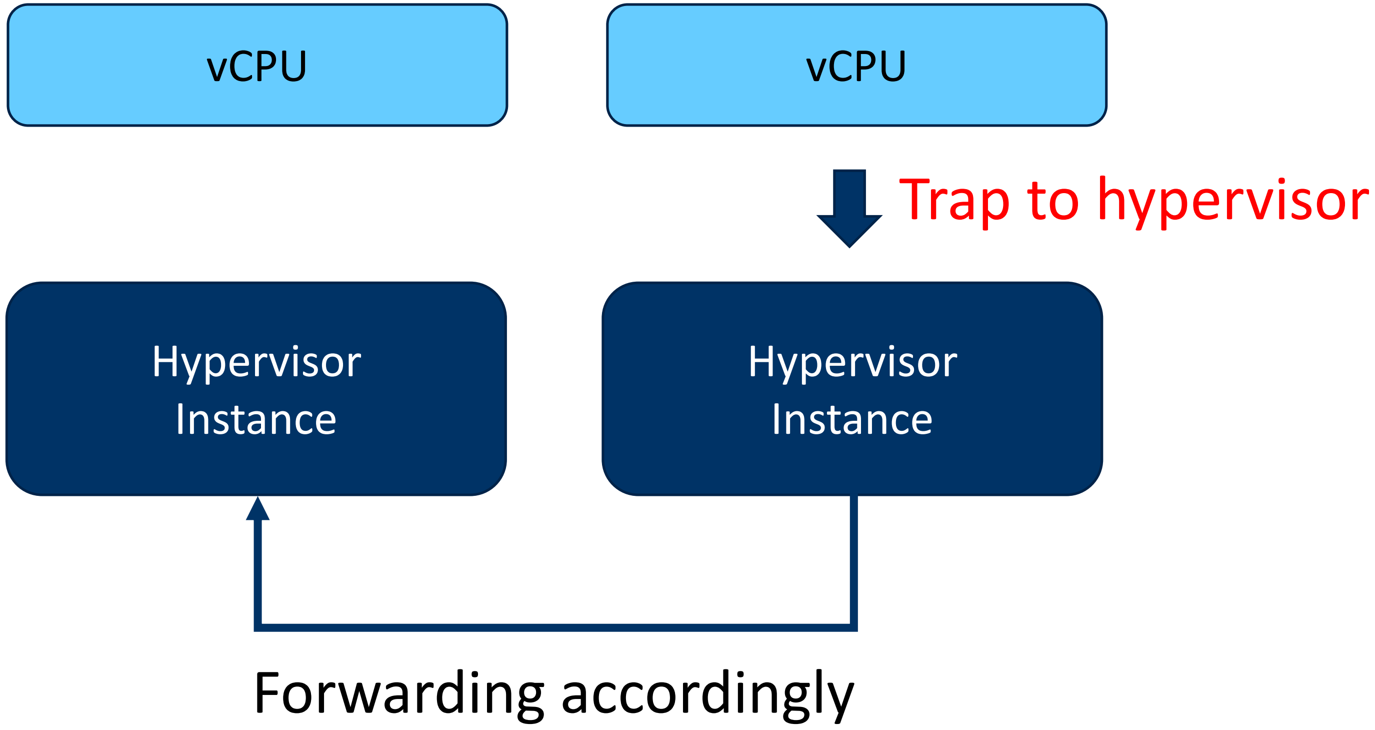
- **Each node runs with a hypervisor instance.**
- **Each instance is in charge of partial resources.**
- **The global consistent view seen by the guest is maintained by the instances.**

GiantVM: Design

- **The Von Neumann architecture consists of three components:**
  - CPU
  - I/O
  - Memory
- **And GiantVM should make these components distributed.**

- **CPU is distributed originally.**
- **I/O is centralized and usually multiplexed by device drivers.**
- **Distributed memory is also known as Distributed Shared Memory (DSM).**

GiantVM: CPU and I/O Virtualization



- **Only interactions between different nodes should be considered.**
  - E.g., inter-processor interrupts (IPI), memory-mapped I/O (MMIO), port I/O (PIO), etc.
- **These instructions are simulated by the hypervisor originally. GiantVM intercepts them by forwarding instructions to the proper remote node.**

GiantVM: Memory Virtualization

- **A straight-forward Ivy (K. Li, 1989) implementation on the EPT.**
  - Although a plethora of troubles when porting to QEMU-KVM…
- **Although Ivy only implements sequential consistency (an ancient and somewhat low-efficient memory model), there is no alternatives.**
  - Our target platform x86 follows x86-TSO.
  - In x86-TSO, write can be delayed until a ‘mfence’ is executed.
  - However, we have no way to capture execution of ‘mfence’. Thus we can only apply a conservative strategy.

Ivy Protocol in a Nutshell

- **Each page of memory space has one of three states:**

Privilege	State
Read and write	**M**odified
Read only	**S**hared
Cannot read or write	**I**nvalid

- **Some terminologies:**

Term	Description
Owner	The node holding the latest data.
Copyset	The nodes holding the valid data (owner is always included).

Ivy Protocol in a Nutshell

- **Two operations (read and write) on three states:**

	read	**write**
Modified	√	√
Shared	√	Invalid other copies. Change to the owner, *Modified*
Invalid	Ask owner for latest page. Change to the *Shared*	Ask owner for latest page Invalid other pages. Change to the owner, *Modified*

CONTENTS

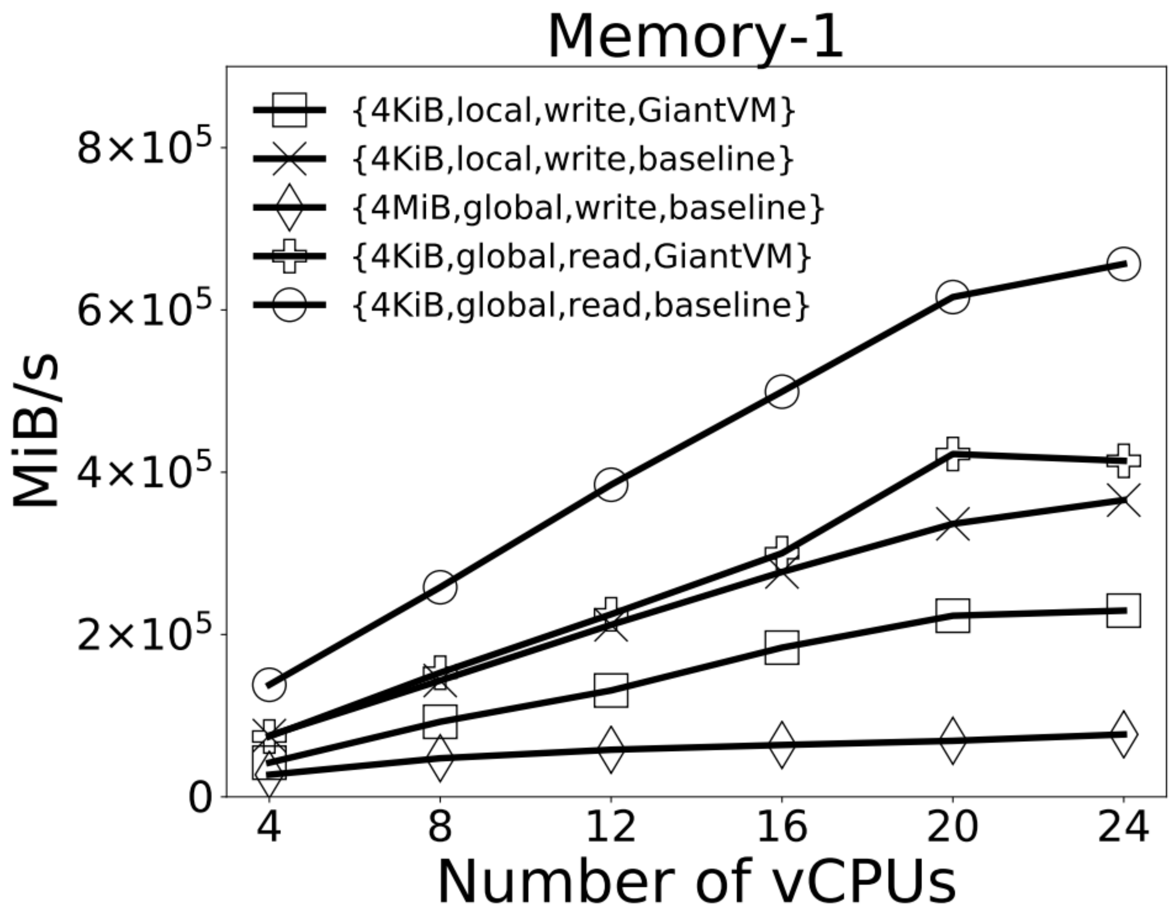
Experiment Setup

Cluster	
Number of Machines	4
CPU	8-core Intel Xeon E5-2620 v4
DRAM	128GB
Ethernet NIC	Broadcom NetXtreme BCM5720 Gigabit Ethernet
RDMA HCA	ConnectX-3 MCX354A-FCBT 56Gbps InfiniBand
OFED Version	Mellanox OFED v2.2-1
Disk	SEAGATE ST9300605SS
OS	Ubuntu 16.04
Control Group	
CPU	24-core Intel Xeon E5-2650 v4
DRAM	64GB
OS	Ubuntu 16.04
Guest Configuration	
DRAM	64GB
OS	Ubuntu 16.04

Baseline

Microbenchmarks

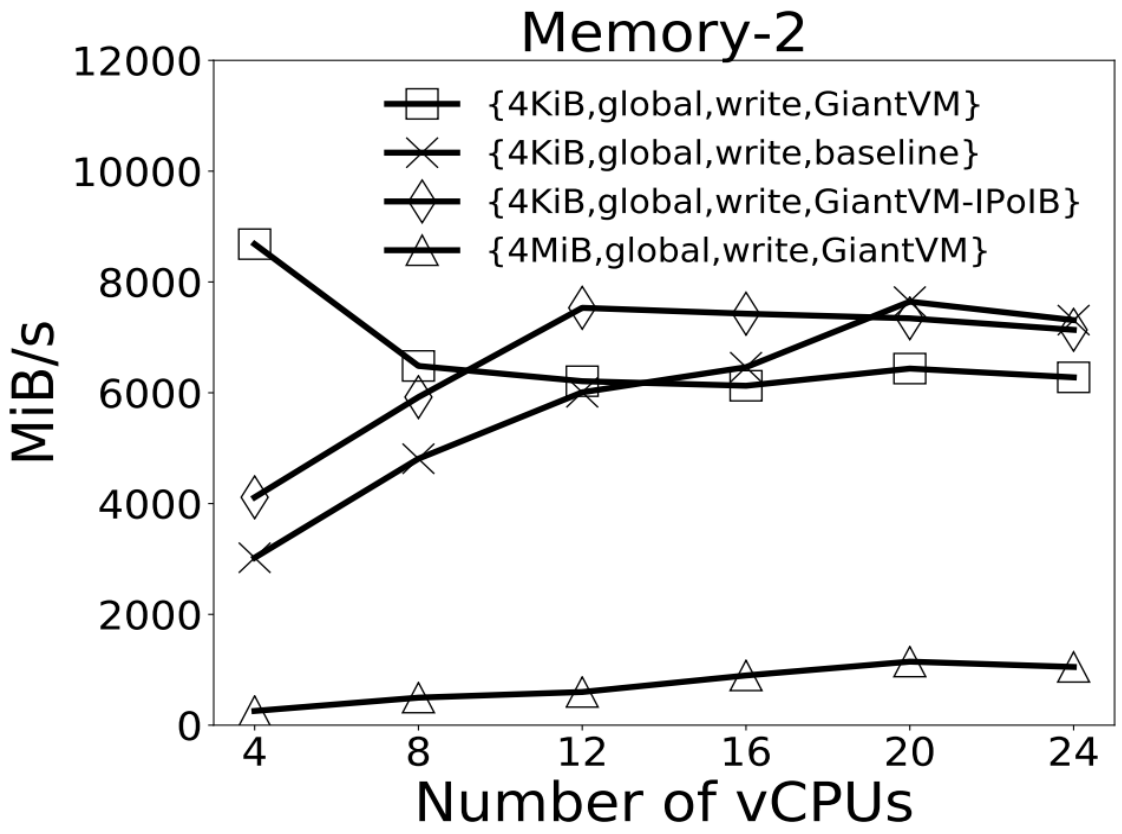
- **Sysbench-memory**
  - Best scenarios: read-only operations or write local pages.



- **Note the similarity between MESI or MOESI protocols used in cache system and Ivy protocol used in the DSM.**
  - Read-only or write-local are the optimum which leads no state transmissions.

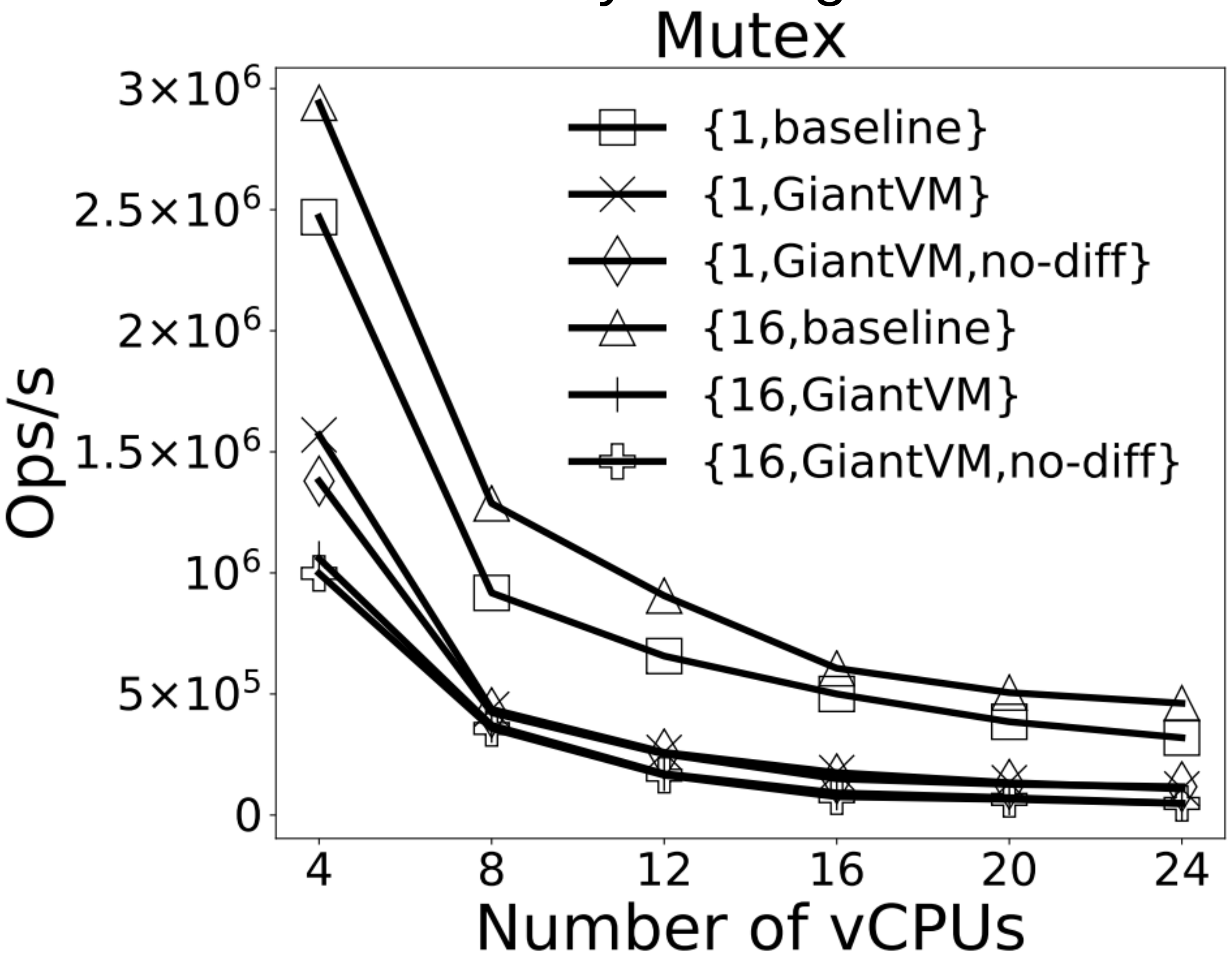
Microbenchmarks

- **Sysbench-memory**
  - Worst scenarios in theory: write global shared page.
  - However, there is an interesting “slower is faster” phenomenon.



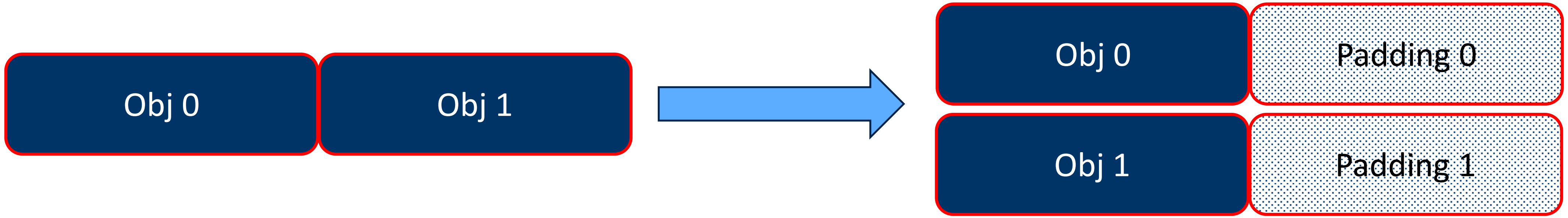
Microbenchmarks

- **Sysbench-mutex**
  - Another worst scenarios in theory: write global shared mutex(es).



Illustration

- **A *shared unit* means a cacheline in the cache system or a page in the DSM system.**
- **Thrashing**
  - One shared unit is written by multiple vCPUs. The shared unit therefore ping-pongs across different vCPUs.
- **False Sharing**
  - Multiple memory objects are allocated in one shared unit.
  - The access of one object of those brings another one together.
  - Sol. Allocate cacheline (page size) alignment memory objects.

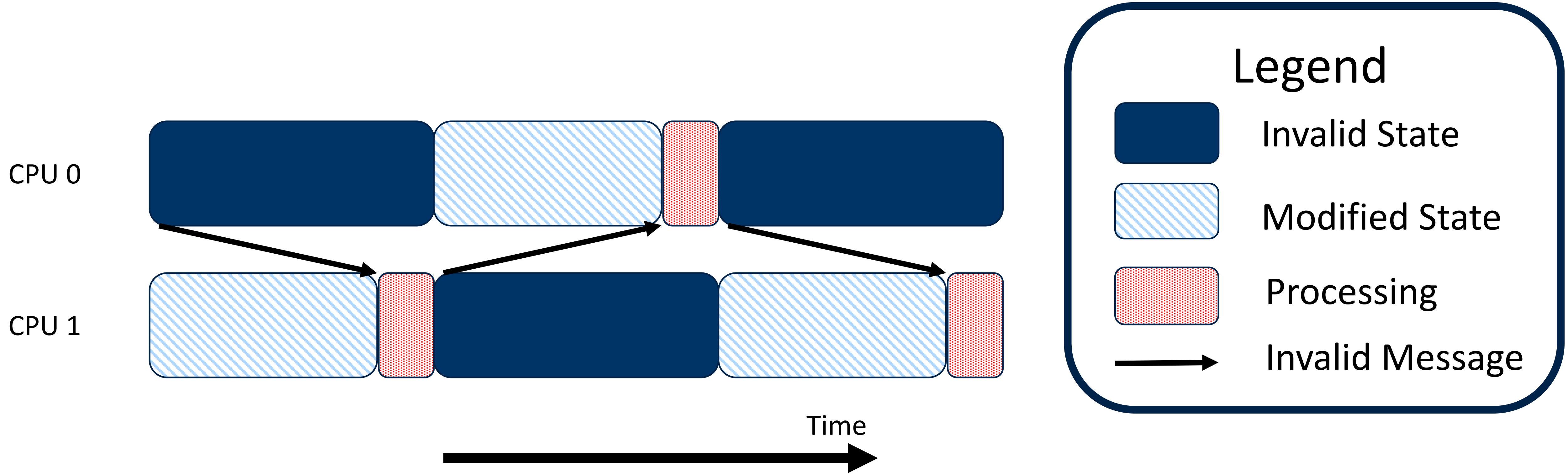


Illustration

- **Now back to write-global scenario.**
  - Sysbench-Memory and Sysbench-Mutex are different.
  - Memory is *throughput-oriented*, when the local vCPU can make progress if remote Invalid message has not arrived yet.
  - Mutex is *latency-oriented*, when the local vCPU cannot make progress if remote Invalid message has not arrived yet.

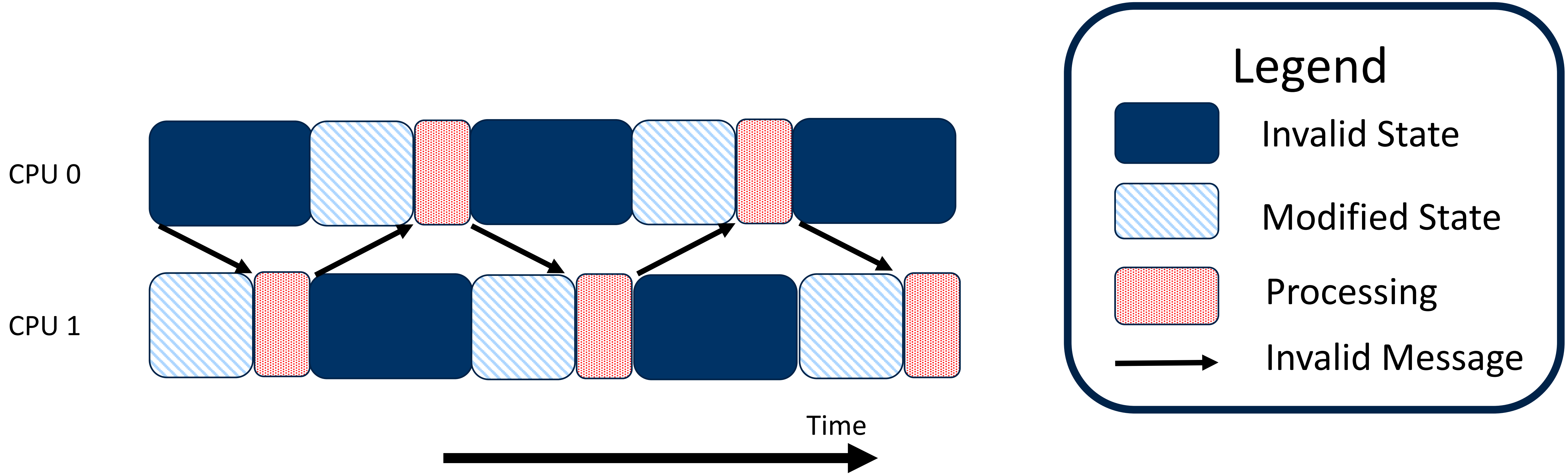
Illustration

- **Three scenarios**
  - **1. A slow Invalid message.**
  - 2. A fast Invalid message.
  - 3. The working set spans across the multiple shared units.



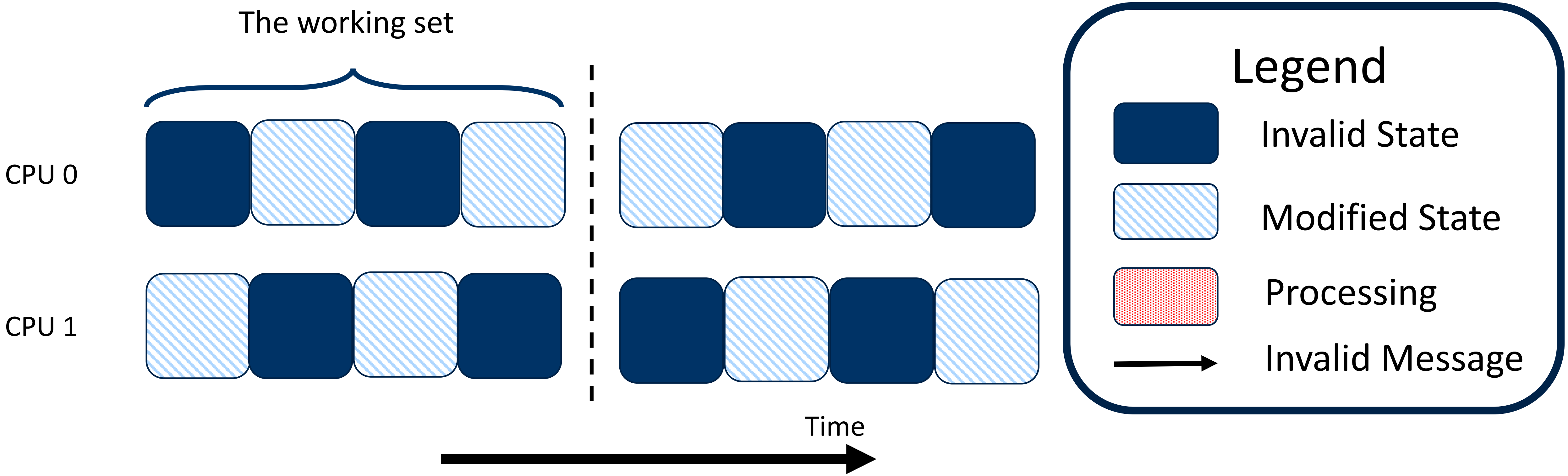
Illustration

- **Three scenarios**
  - 1. A slow Invalid message.
  - **2. A fast Invalid message.**
  - 3. The working set spans across the multiple shared units.



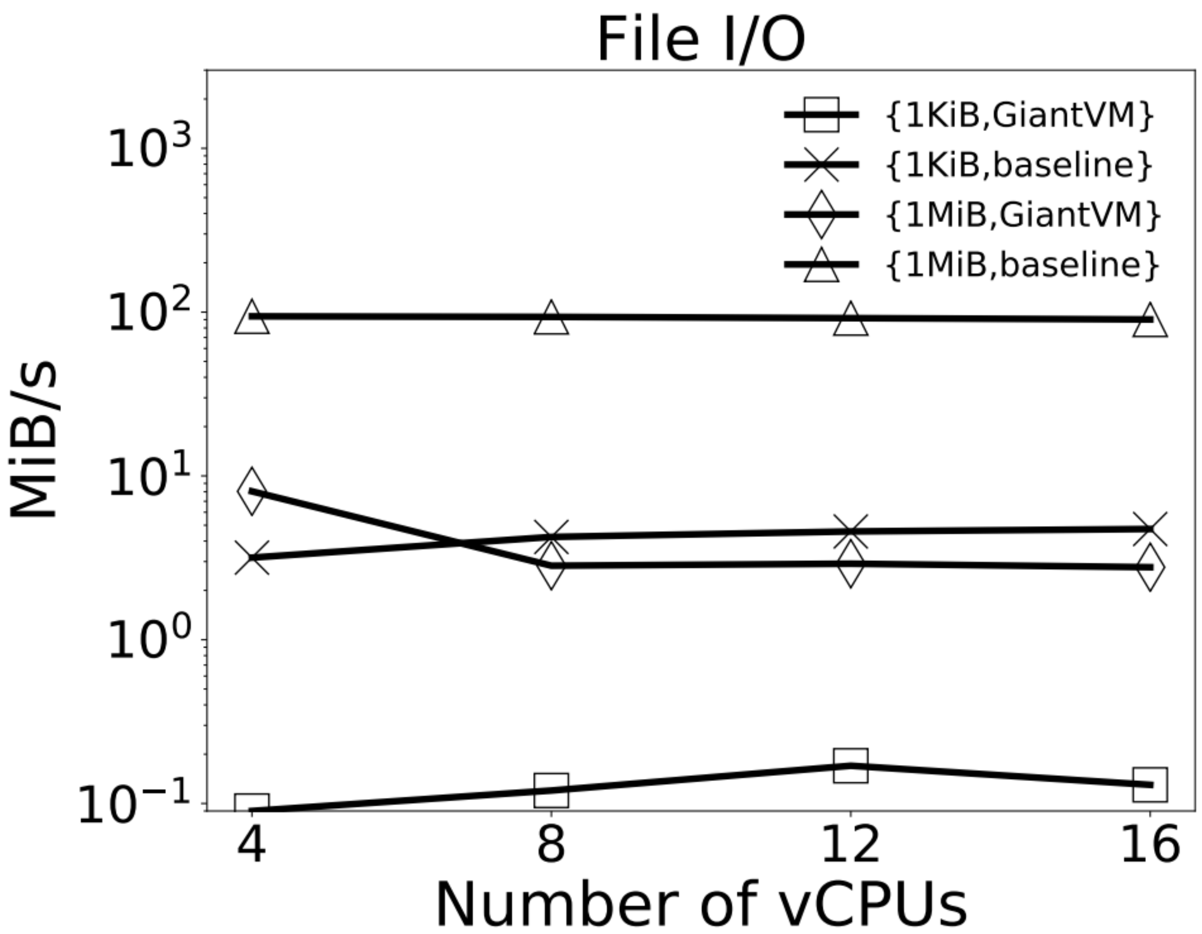
Illustration

- **Three scenarios**
  - 1. A slow Invalid message.
  - 2. A fast Invalid message.
  - **3. The working set spans across the multiple shared units.**



Microbenchmark

- **Sysbench-File I/O**
  - A catastrophic consequence of File I/O with 20x slow down.



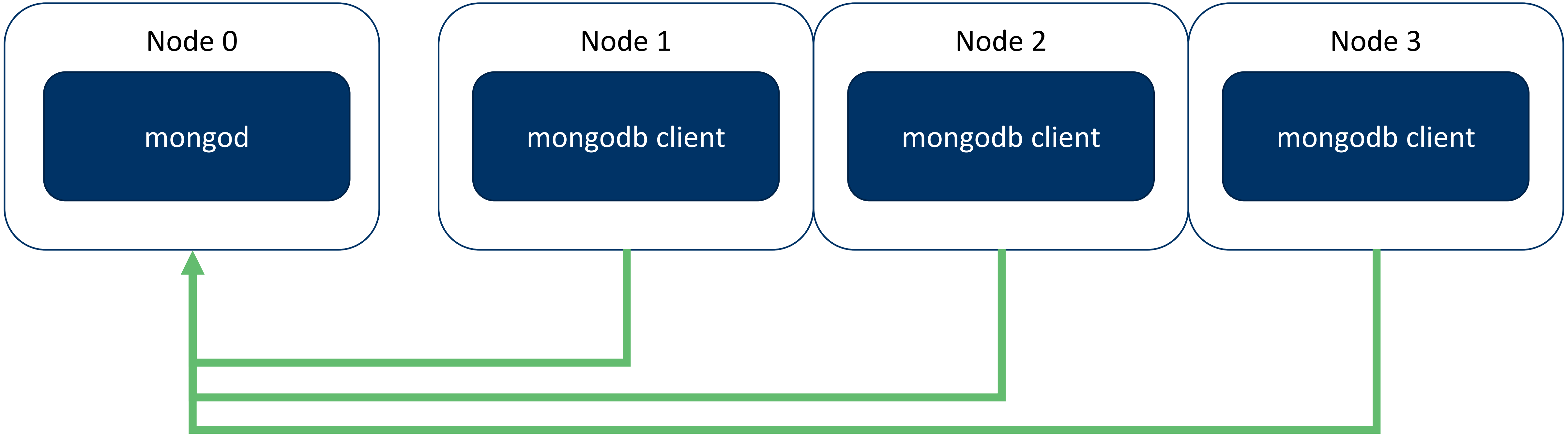
Microbenchmark

- **Direct usage of applications causes negative consequence.**
  - The I/O-intensive workload frequently traps to kernel and kernel pages are shared across all the vCPUs.
  - The shared pages lead to thrashing and false sharing.

Monolithic kernel like Linux may not be a good idea!

Application Evaluation: Data Colocation

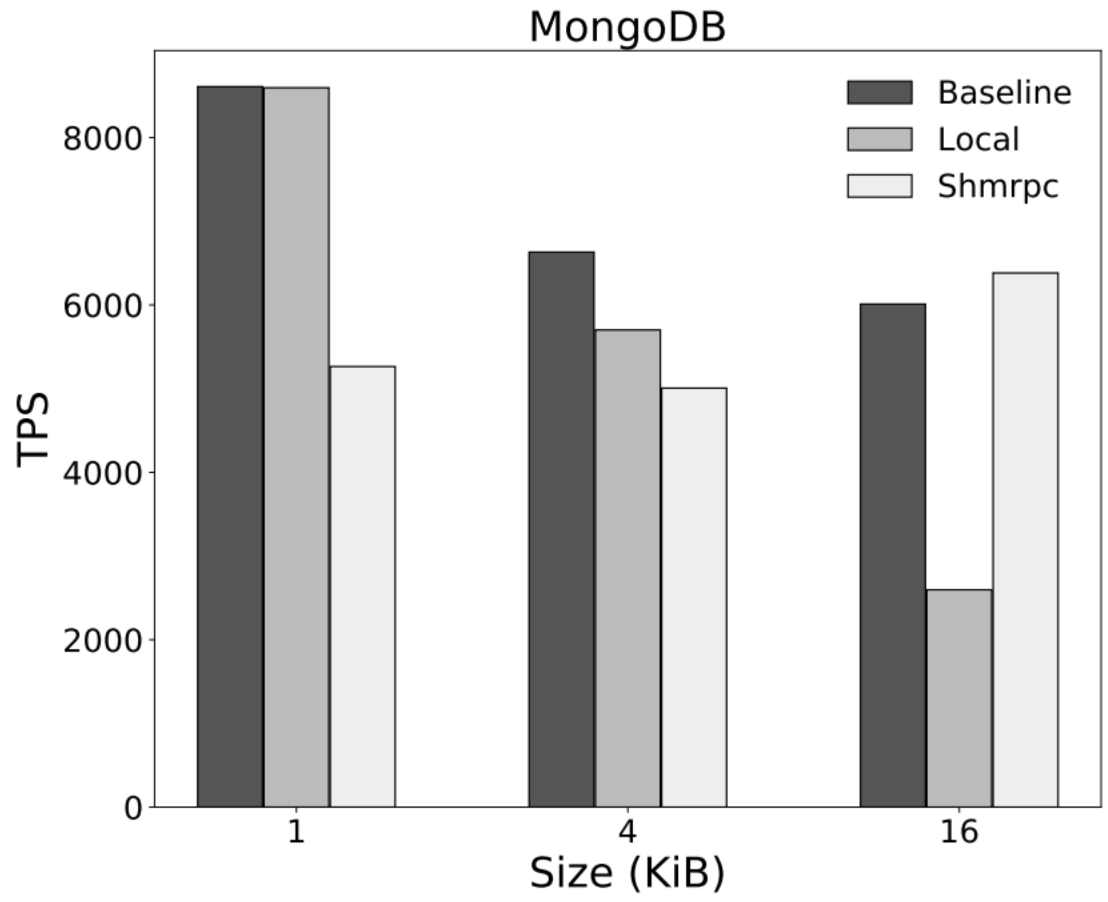
- **Some ideas borrowed from *Multikernel*:**
  - The applications as well as device drivers are binding to specific cores.
  - The communication between different cores is via shared-memory RPC.
- **Consider an example of MongoDB:**



Shared Memory RPC

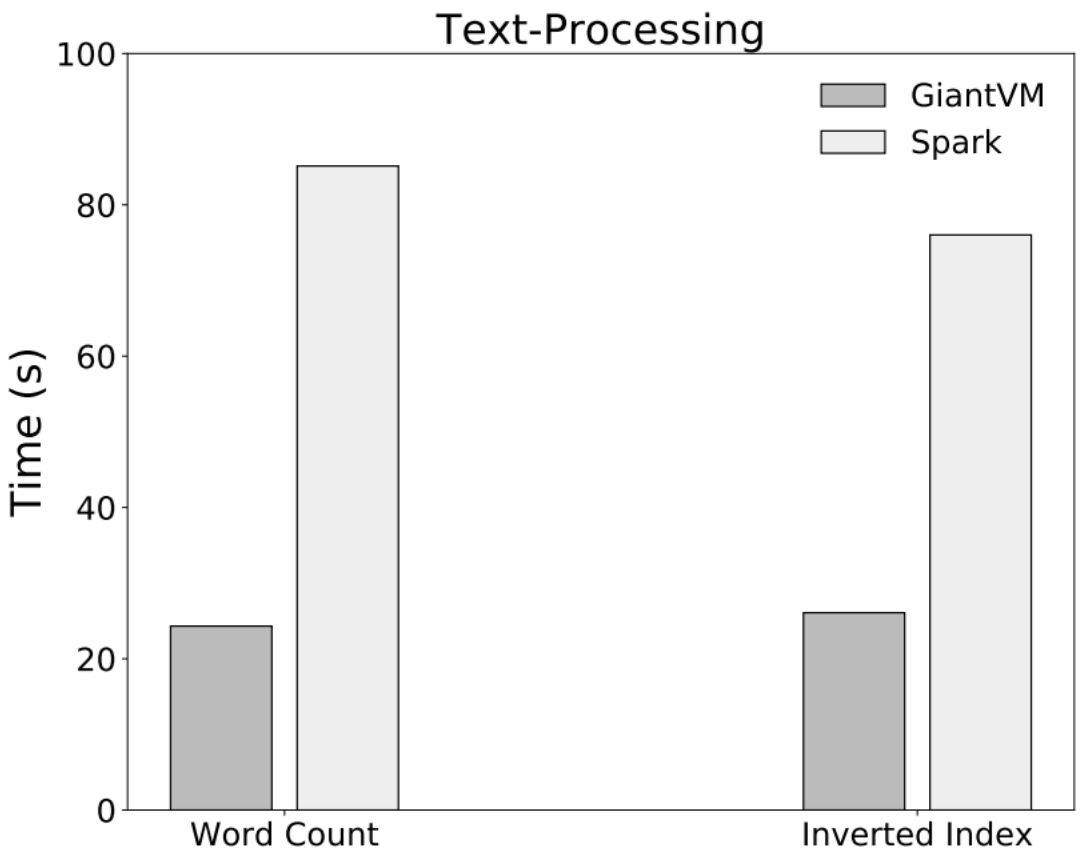
Application Evaluation: Data Colocation

- **MongoDB spins on inserting/removing randomly generated documents.**



Application Evaluation: Spark

- **Comparison of two text-processing programs with Spark:**



Future Work

- **The integration with multikernel (Barrelfish) is work in progress.**
  - Multikernel brings additional benefits, especially fault-tolerance.

- **Relaxing memory model.**
  - ‘mfence’ instructions can be simulated by para-virtualization.
  - Only in the kernel space.

Conclusion

- GiantVM verifies the viability of applying many-to-one virtualization to the SSI.

- There are some interesting features when running specific workloads in GiantVM.

- A monolithic kernel like Linux may not be the ideal architecture. Instead, ideas from multikernel brings the enhancement of the performance.

Q&A

Our website: http://tcloud.sjtu.edu.cn/

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY