Gleaner: Mitigating the Blocked-Waiter Wakeup Problem for Virtualized Multicore Applications

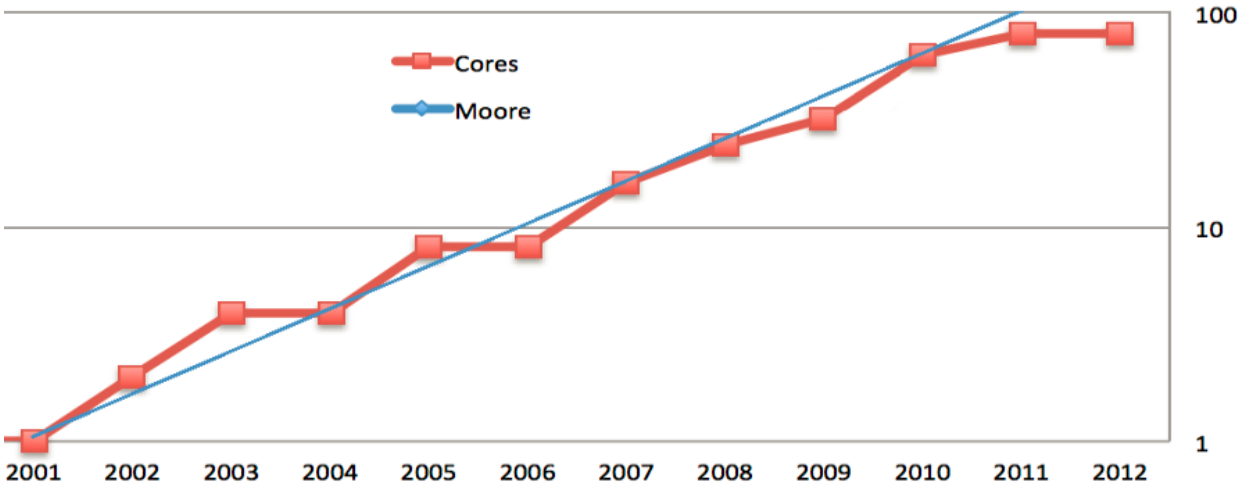
Xiaoning Ding Phillip B. Gibbons
Michael A. Kozuch *Jianchen Shan*

New Jersey Institute of Technology

Intel Labs Pittsburgh

Background

- **Number of cores in a computer keeps increasing**



* Number of cores in the x86 system with highest performance in SPECint Rate benchmark reports.
Adapted from John Appleby 's blog page *HANAlgorithmics - Efficiency by Design with SAP HANA*

- **Number of virtual computing units (i.e. vCPUs) in a virtual machine also keeps increasing**

Amazon EC2 Instance Types

	media	large	xlarge	2xlarge	4xlarge	8xlarge
# vCPUs	1	2	4	8	16	32

Mismatch between vCPU abstraction and pCPU behavior

- Can applications effectively take advantage of more vCPUs in each VM?
- vCPUs are schedulable execution entities.
  - E.g. KVM vCPUs are normal threads scheduled by Linux scheduler.

State	pCPU	vCPU
Busy	Can make continuous progress in parallel	May be suspended without notification

Application expectation **Reality**

Performance indication --- Synchronization

vCPUs may be suspended

Busy waiting

Lock holder preemption problem (LHP)

- **The vCPU holding a spin-lock is preempted**
- vCPUs waiting for the lock spins for a long time
- LHP reduces scalability and efficiency
- Has been well studied. Hardware solution in CPU.

Idle vCPUs resume execution slowly

Blocking

Blocked-waiter wakeup problem (BWW) *

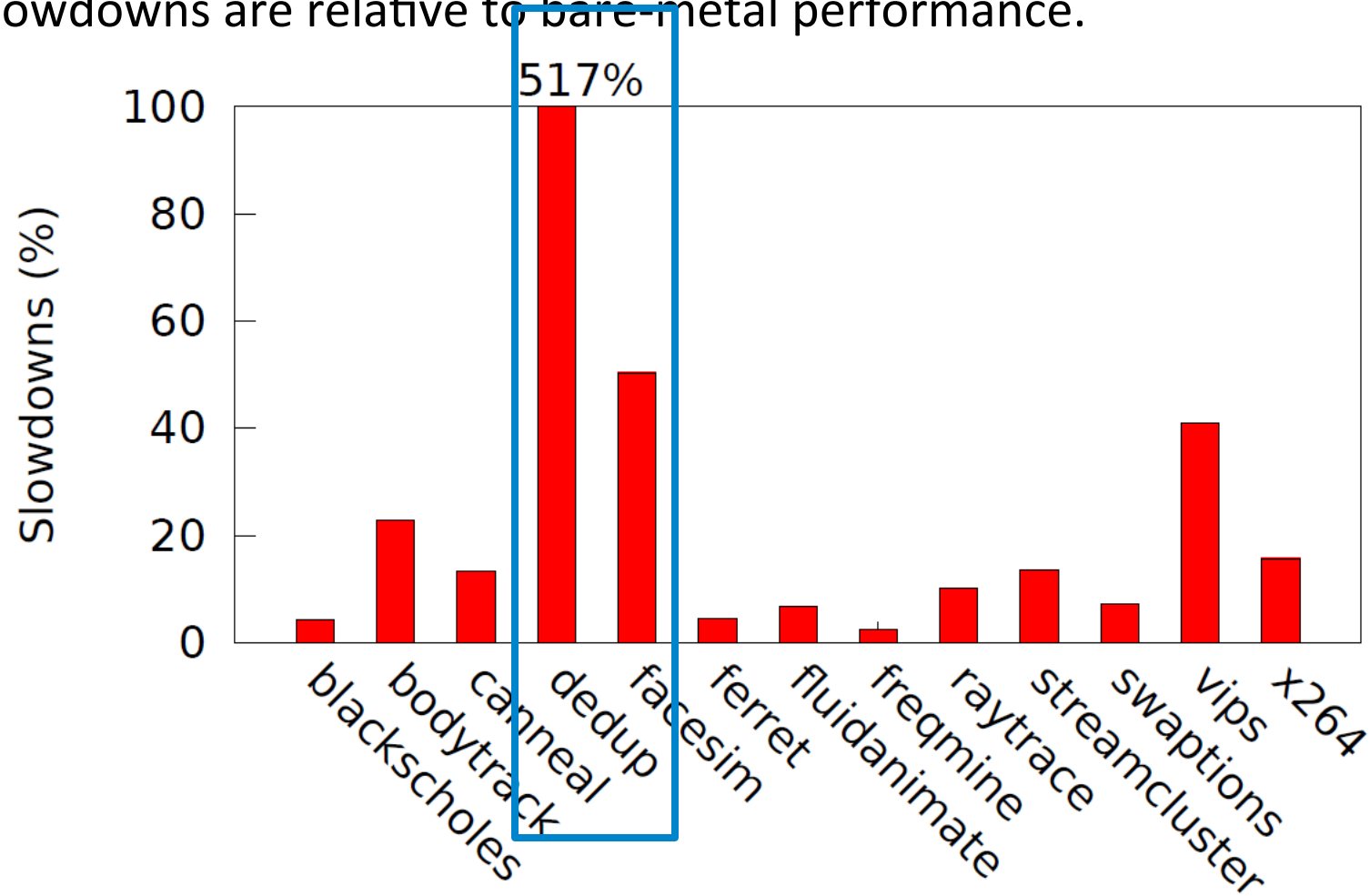
*[TR FDUPPITR-2010-002]

Blocked-waiter wakeup problem (BWW)

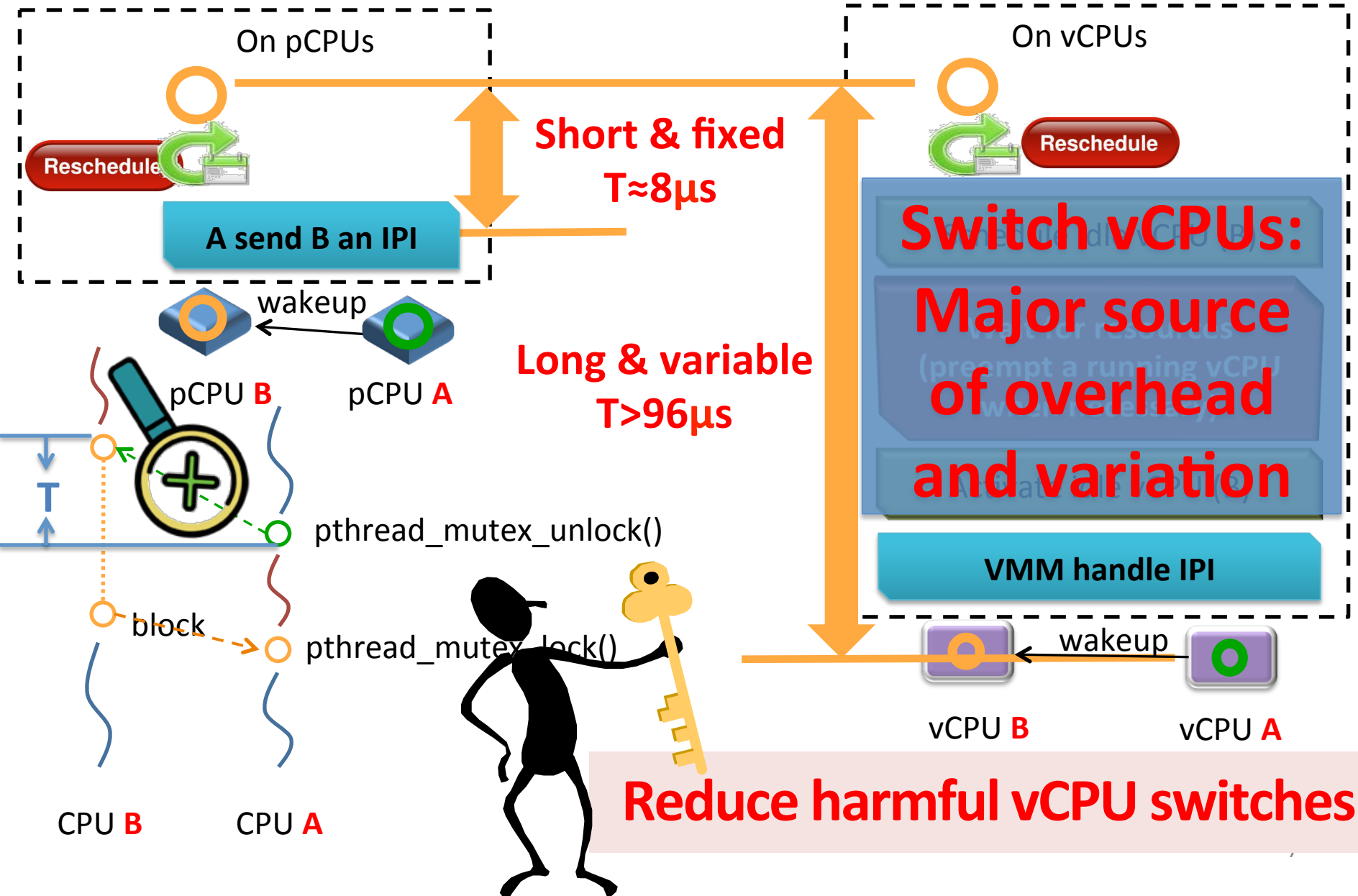
- BWW: waking up blocked threads takes long time on idle vCPUs
  - Idle vCPUs must be rescheduled before they run any threads
    - VMM suspends vCPUs when they became idle (due to lack of runnable threads).
  - How long? --- Depends only the availability of hardware resources and the cost of vCPU switches
- BWW degrades performance of synchronization intensive applications.
  - Computation is delayed → Increase of execution time in VMs
  - Unpredictable/variable delay → Unpredictable performance
  - Extra overhead of vCPU switches → Reduced overall system performance

BWW causes serious performance degradation even on dedicated hardware

One vCPU on each physical core.
PARSEC benchmarks run in the VM.
Slowdowns are relative to bare-metal performance.



Waking up a thread on an idle CPU



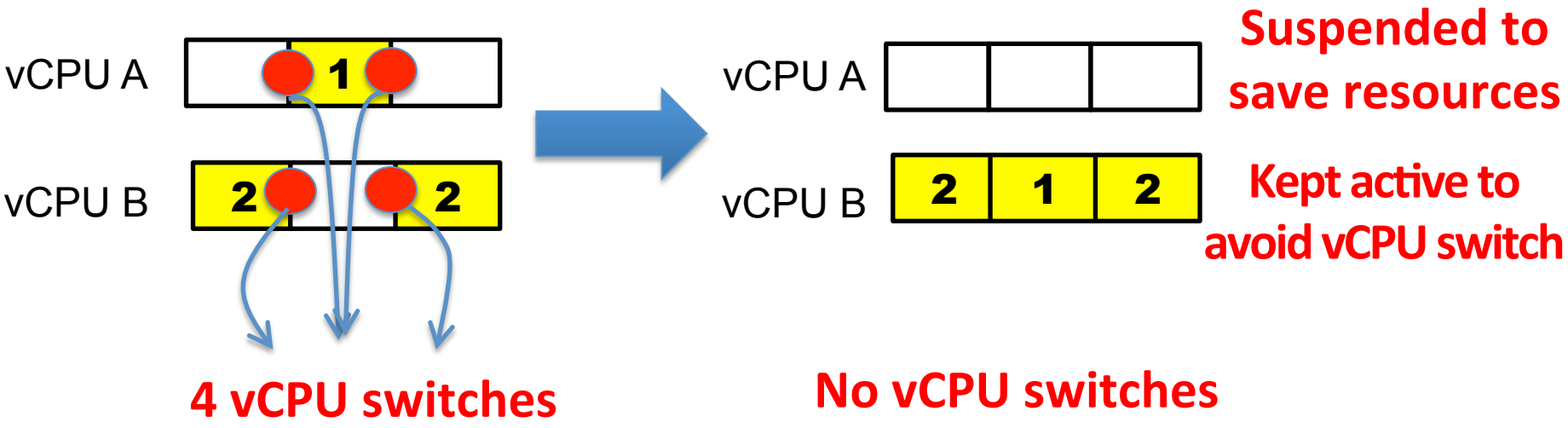
Reducing vCPU switches---- resource retention

- Prevent idle vCPU from being suspended
  - Idling operating, e.g. an idle vCPU spins instead of yielding hardware resources.

- May cause resource under-utilization
  - System throughput is reduced when vCPUs are idle for a significant amount of time.
  - Must balance the cost of spinning and vCPU switch
    - Set a timeout for spinning.
    - Pay both overhead (spinning + vCPU switches) for long idle periods causing timeouts

Reducing vCPU switches
--- Consolidation scheduling

Consolidate busy periods and coalesce idle periods on vCPUs

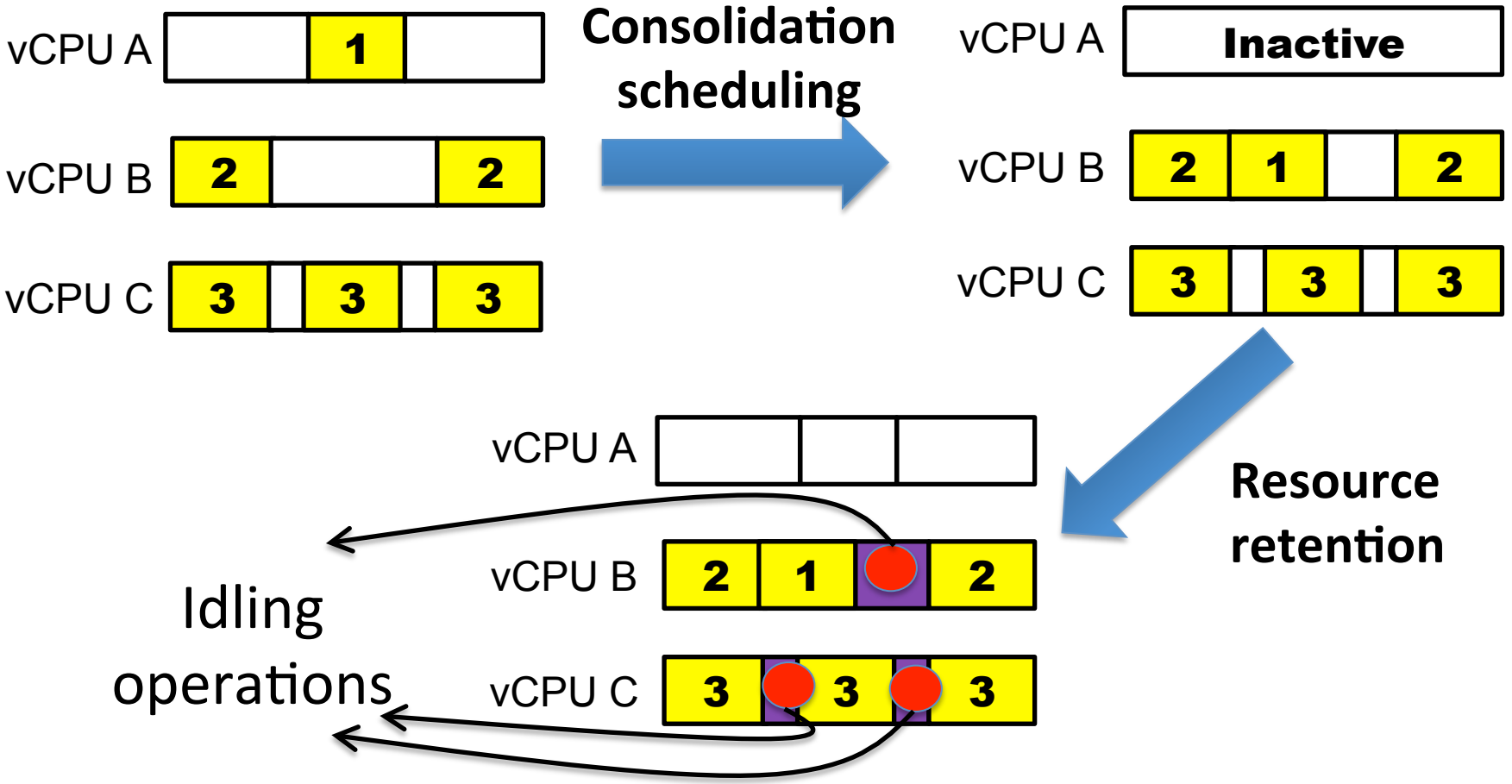


- Problem: active vCPUs may be overloaded → low performance

Gleaner --- basic idea

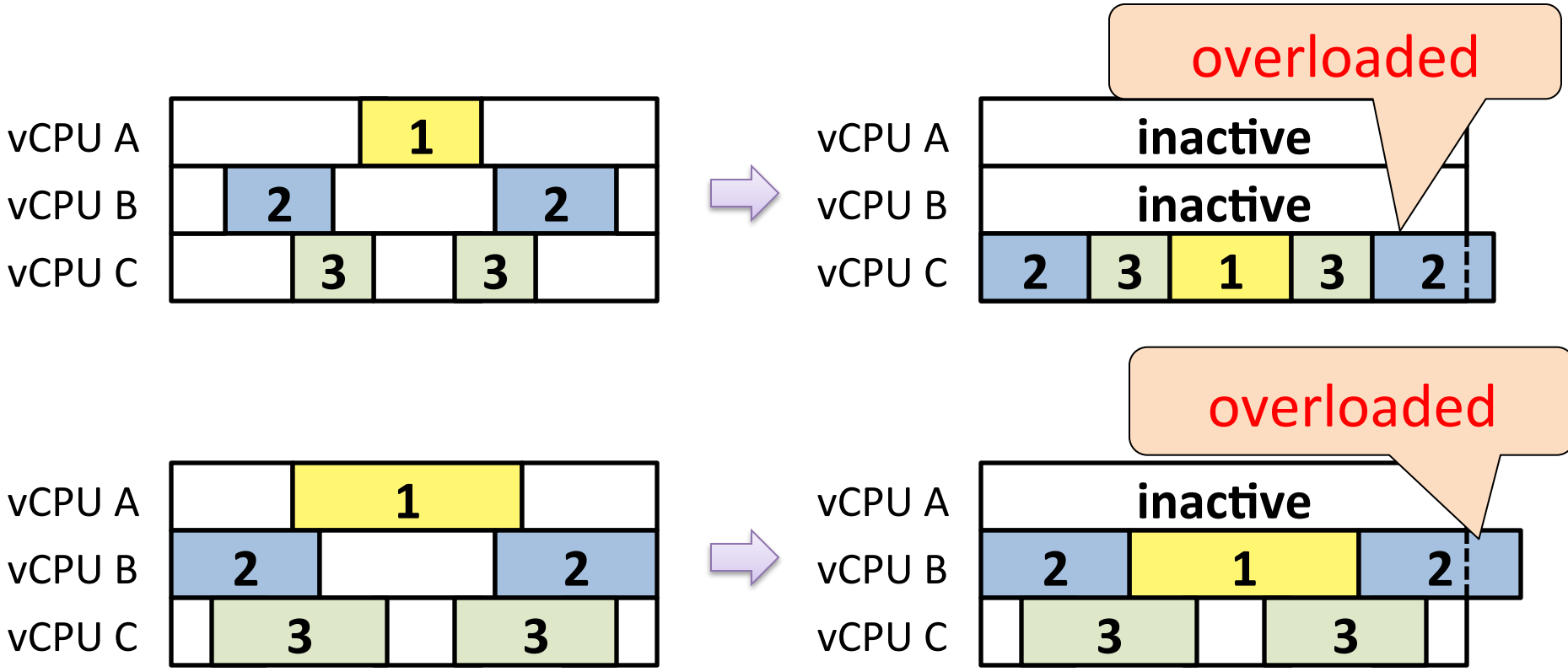
consolidation scheduling + resource retention

(reduce long idle periods) (manage short idle periods)



Causes of overloading

- Workload is too heavy for active vCPUs
- Workload cannot be evenly distributed among active vCPUs
  - e.g. long computation periods



Overloading prevention

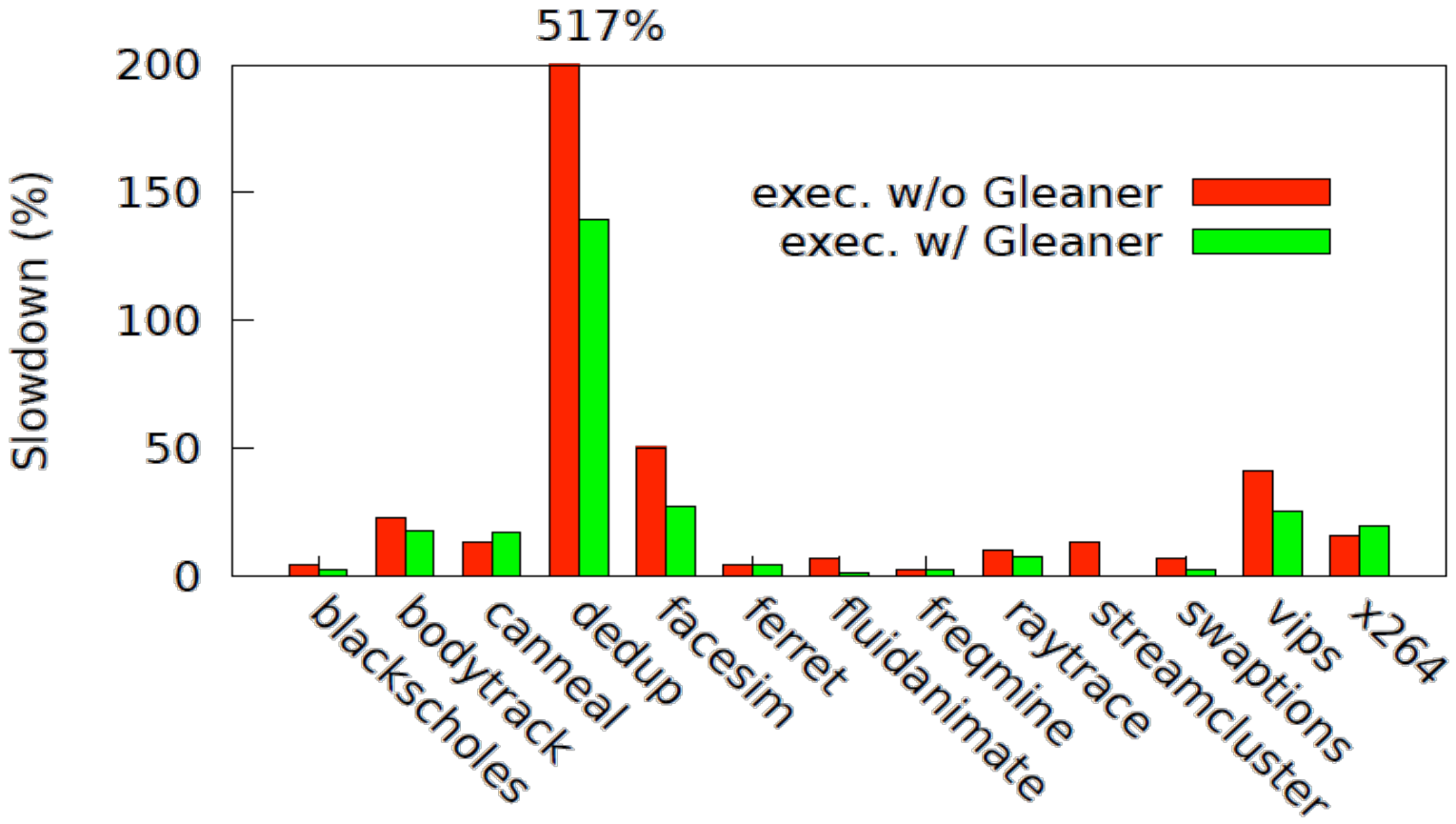
- Gleaner gradually consolidates workload threads
- Gleaner only proceeds if
  - vCPU utilization would not be too high after consolidation
  - Workload can be evenly distributed among active vCPUs
- Gleaner stops consolidation when it observes throughput starts to reduce
- Refer to the paper for details

Experiment Setup

- A prototype implementation at user level
- Dell PowerEdge server with 16 cores
- Ubuntu Linux 12.04 with kernel updated to 3.9.4
- VMM is KVM in Ubuntu distribution.
- Each VM has 16 vCPUs
- Benchmarks:
  - PARSEC 3.0 suites
  - MySQL driven by SysBench
  - MatMul (Matrix Multiplication)

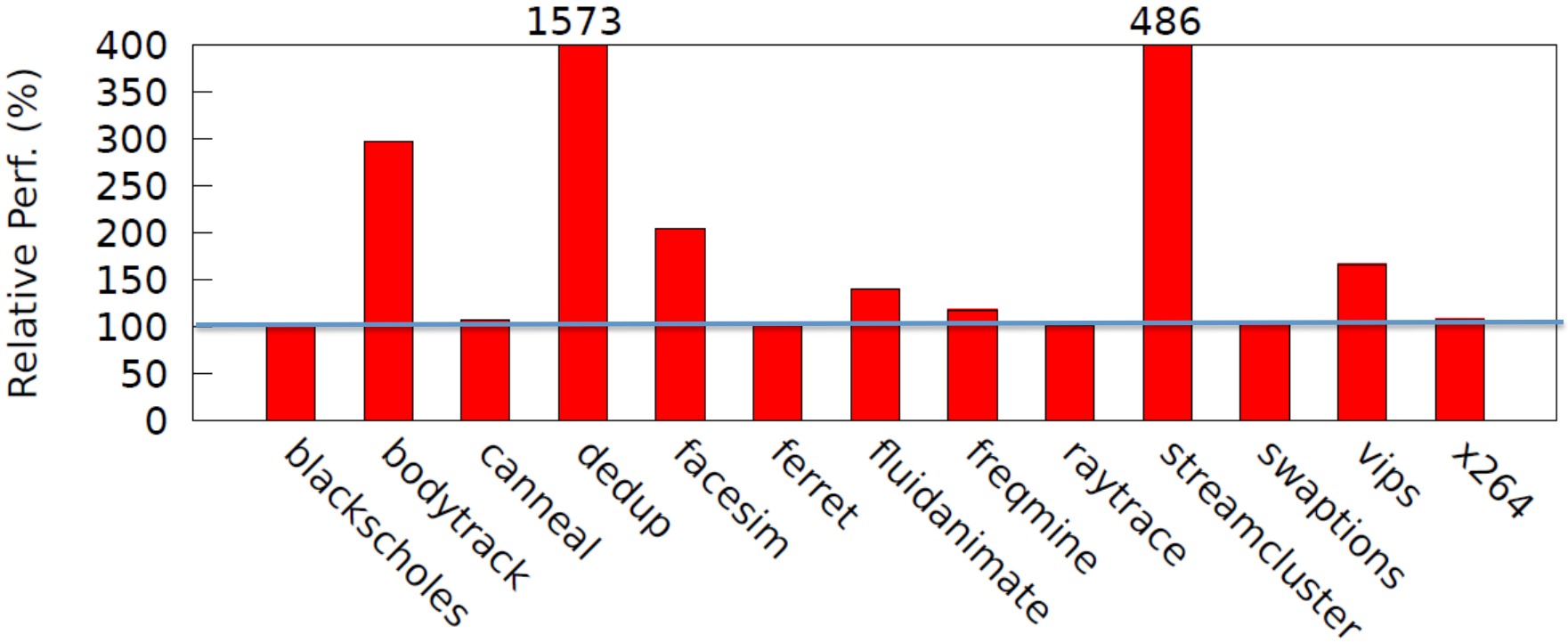
Application Performance on a VM with dedicated hardware

One vCPU on each core (16 vCPUs on 16 pCPUs).
Slowdowns are relative to bare-metal performance.



Application performance on oversubscribed system

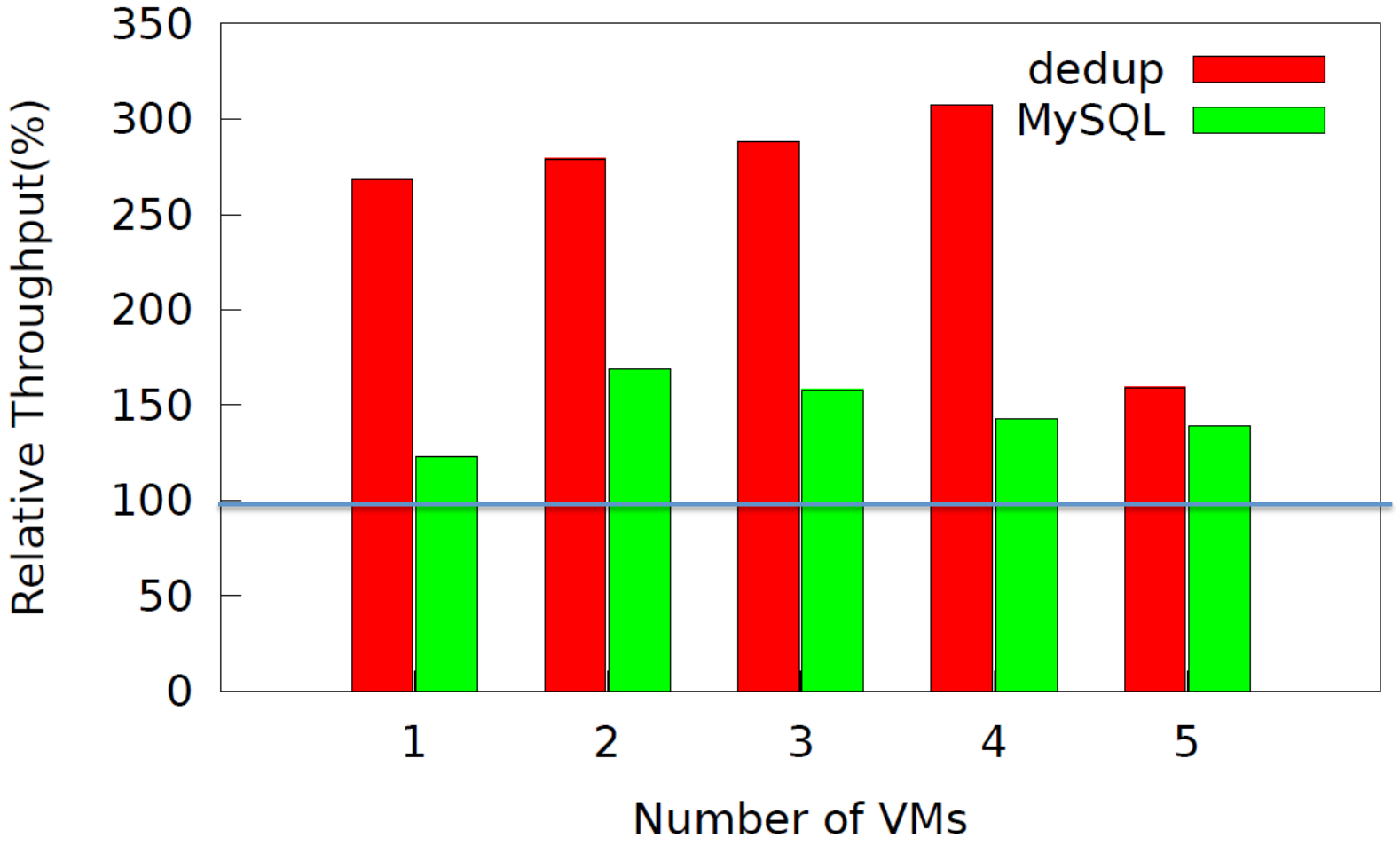
- Run a PARSEC benchmark in a VM and *matmul* in the other VM.
- 32 vCPUs (16 in each VM) on 16 cores
- Performance is relative to the stock system (without gleaner)



Gleaner improves application performance significantly (16x) on oversubscribed systems

Gleaner improves overall throughput

- Run multiple identical VMs (16 vCPUs each)
- Throughput is relative to stock systems without gleaner

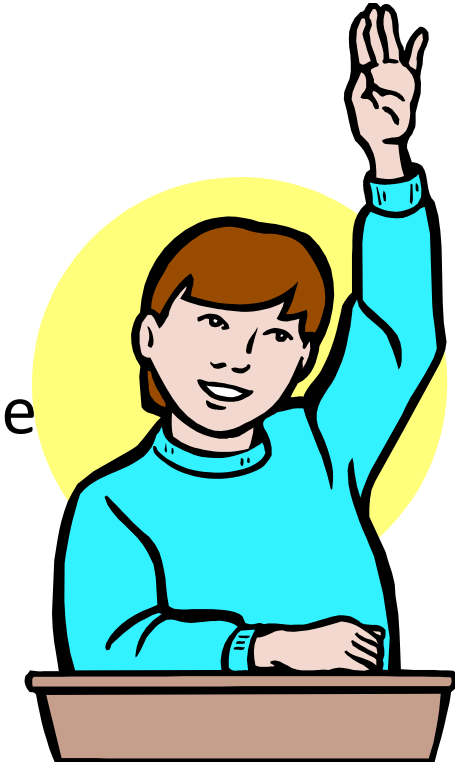


Conclusion

- BWW is caused by waking-up blocked threads being delayed by switching back idle vCPUs
- BWW significantly reduces application performance in VMs and overall throughput
- Key: reduce harmful vCPU switches due to idling
- Gleaner combines two methods --- resource retention and consolidation scheduling
- Gleaner can improve application performance by up to 16x and system throughput by 3x.

Conclusion

- BWW is caused by waking-up blocked threads being delayed by switching back idle vCPUs
- BWW significantly reduces application performance in VMs and overall throughput
- Key: reduce harmful vCPU switches due to idling
- Gleaner combines two methods --- resource retention and consolidation scheduling
- Gleaner can improve application performance by up to 16x and system throughput by 3x.



dingxn@njit.edu

http://www.njit.edu/~dingxn