# Preserving I/O Prioritizations in Virtualized OSes

**Kun Suo*, Jia Rao*, Luwei Cheng◆, Xiaobo Zhou* and Francis C. M. Lau^**

* The University of Colorado, Colorado Springs ◆ Facebook ^ The University of Hong Kong

## Introduction

A program running in a virtualized environment should exhibit a behavior essentially identical to that in a physical environment [Popek and Goldberg'74]

• OSes do not preserve the property of I/O prioritization when running as virtualized guest Oses and in shared environments.

• Without prioritization, I/O-bound workloads are significantly affected by co-running compute-bound workloads, leading to degraded and unpredictable I/O performance.

• We identify two types of priority inversion issues due to time discontinuity caused by CPU multiplexing.

• We design *xBalloon*, a simple approach to preserving static and dynamic priorities in Linux.

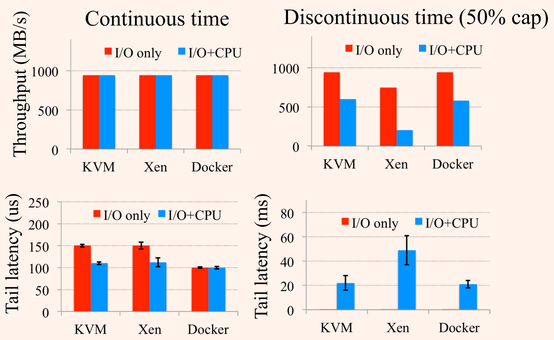
## I/O Prioritization in Linux Guests

Linux prioritizes I/O-bound tasks over compute-bound tasks in two ways. Users can assign an elevated *static* priority, e.g., real-time priority SCHED_RR, to I/O tasks to guarantee I/O performance. Linux CFS scheduler uses dynamic priorities based on virtual runtime to prioritize tasks that have short CPU bursts and long I/O wait time.

## Time Discontinuity in Guest OSes

It is a common practice in the cloud to share CPU among multiple VMs or limit CPU usage via capping. CPU availability becomes discontinuous to guest Oses running inside VMs. Most OSes synchronize with the hypervisor to avoid time drift, thereby seeing discontinuous passage of time.

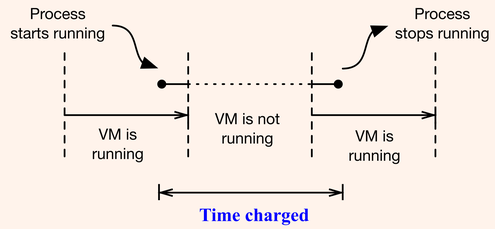
## Violations of I/O Prioritization




*I/O prioritization guarantees that I/O performance is not affected by co-running compute task under* ***continuous*** *time. Guest OS does not preserve I/O prioritization under* ***discontinuous*** *time, leading to significantly degraded and wildly unpredictable I/O performance*

## Short-term Priority Inversion

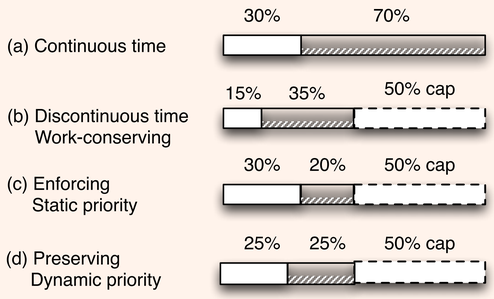
Linux's CFS uses virtual runtime (vruntime) to track CPU usage and prioritizes tasks with small vruntimes. Short-term priority inversion happens when the vruntimes of I/O tasks are mistakenly inflated under discontinuous time so that CFS fails to prioritize them.



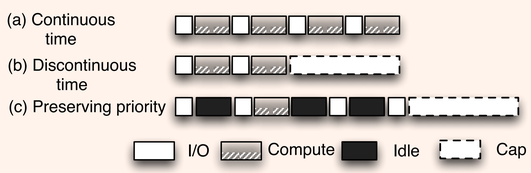

*Time discontinuity causes inaccurate CPU accounting to processes in the guest OS and I/O-bound tasks can appear to be CPU-bound*

## Long-term Priority Inversion

Work-conserving scheduling, which is designed for dedicated systems under continuous time, causes the long-term priority inversion in guest OSes with constrained CPU allocation and discontinuous time.




*Work-conserving scheduling in the guest OS does not preserve priorities between the high priority (white bar) and low priority task (shaded bar) under discontinuous time*




I/O requests arrive at discrete time and work-conserving scheduling in the guest OS allows the compute task to use the CPU that should be used by the I/O task in the future under constrained resource allocation

## xBalloon Design

The keys to preserving priorities are (1) making the guest OS aware of discontinuous time and (2) helping it differentiate the scheduling of I/O and compute tasks considering discontinuous CPU availability.

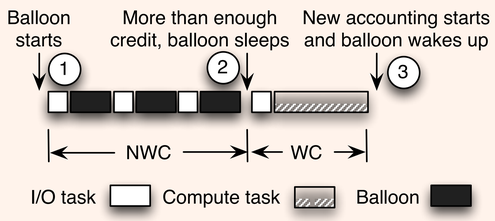
### Differential clocks

xBalloon equips the Linux guest with two clocks, one absolute clock synchronizing with the host clock in the hypervisor and one relative clock ticking only when the VM is running. Accurate CPU accounting is achieved and short-term priority inversion is prevented by using the relative clock for task scheduling.

### CPU balloon

Inspired by memory ballooning, a CPU balloon represents the CPU time a VM voluntarily gives up and reserves for future use. It is a user space process that pauses the VM whenever scheduled to run.

### Semi-work-conserving (SWC) scheduling




*SWC allows the guest OS to autonomously manage the CPU allocated to the VM so that it can truly differentiate task scheduling based on the availability of CPU.*

① The guest OS starts with NWC mode at the beginning of each accounting period. The balloon is active.

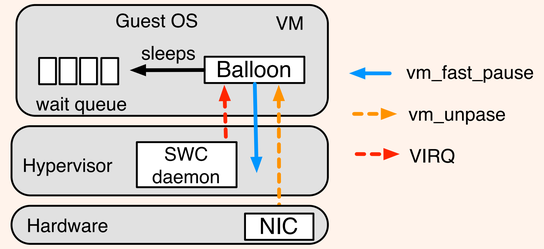
② The guest switches to WC mode if the VM has satisfied resource constraint and the low priority task is free to run.

③ The guest switches to NWC mode and wakes up the balloon when the next accounting period starts.

### Preserving static and dynamic priorities

- Enforcing static priority Low priority tasks use the absolute clock and high priority tasks and the balloon use the relative clock. I/O tasks are assigned the real-time priority while other tasks including the balloon are under normal priority.
- Preserving dynamic priority Treat the balloon as a normal task and rely on Linux CFS to prioritize tasks with small virtual runtimes and enforce fair shares among tasks.

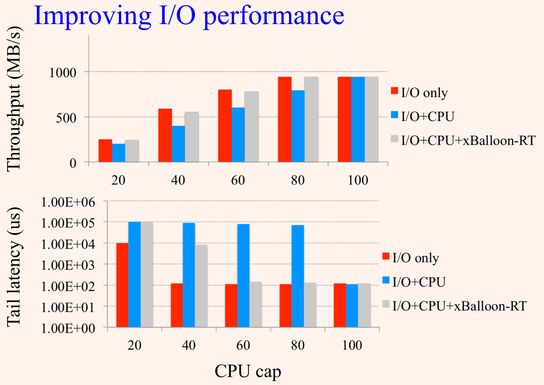
## Implementation



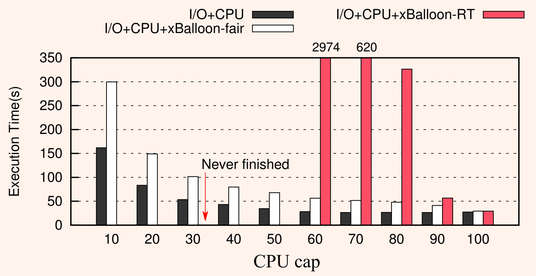

The implementation consists of ~500 LOC in Linux 3.18.21 and Xen 4.5.0

## Evaluation

### Improving I/O performance




### Preserving priority




*xBalloon starves the compute task when the CPU cap is lower than the I/O demand and allows the compute task to use slack CPU when the cap increases.*

## Acknowledgement

This work was supported by the U.S. National Science Foundation under grant CNS-1320122 and CNS-1422119

UCCS University of Colorado Colorado Springs