Implementing Enterprise-Level Quality of Service Using Variable Resource Allocation in the Xen 4 Hypervisor

James Cook
jcook908@vt.edu

Michael Lowman
mdlowman@vt.edu

Kyle Morgan
knmorgan@vt.edu

Conor Scott
conscott@vt.edu

CS4284 Systems & Networking Capstone. Spring 2011. Faculty advisor: Ali R. Butt

Introduction

- Increasing popularity of cloud offerings
- Infrastructure-as-a-Service provides an abstract concept of computing resources
  - CPU, disk, memory, etc.
- Single server should handle many VMs
- Currently no means to guarantee resources allocated to VMs running on a hypervisor
- Major customers are given dedicated servers for important VMs
  - Guarantees availability
  - Inefficient, expensive overprovisioning
- Acceptable means for CPU and memory QoS currently available
  - Core → CPU mappings
  - Memory ballooning
- No acceptable means for disk monitoring

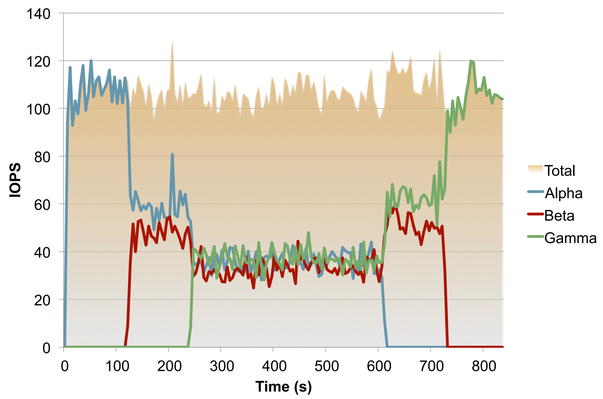
Motivation

- Cloud service providers would benefit greatly from enhanced QoS in the I/O path
- Saving money by consolidating servers
  - Doesn't degrade VM performance
- In-place mechanisms not sufficient
  - Many workarounds; no comprehensive solutions
  - Missing element to compartmentalize performance

Implementation

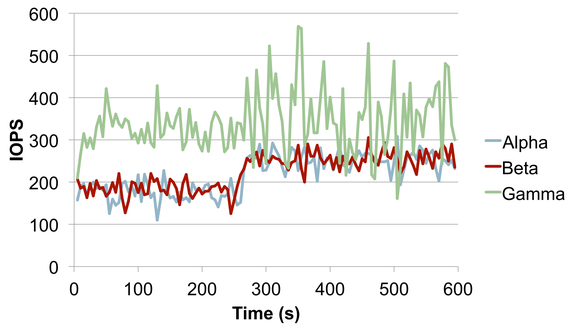
- Xen is open source: modifiable
- Enterprise-class standard solution
- Used by major cloud providers
- Test platform of Debian server
  - Three Debian VMs (alpha, beta, and gamma)
  - Linux 2.6.32 with Debian Xen pvops patchset
  - AMD Sempron™ Processor 3800+
  - 2 GB memory

Figure 2: Read IOPS with Staggered Benchmarks



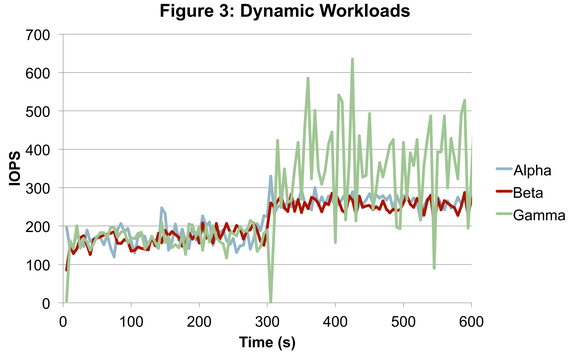
- Monitor I/O by polling sysfs data structures and parsing relevant data
- Ran suite of benchmarks to find scheduling cases with poor performance
  - Used Flexible File System Benchmark (ffsb) to create I/O workloads with varying characteristics
    - Sequential read/writes
    - Random read/writes
    - Small file workload (mail server)
    - Large file workload (database)

Figure 1: Total IOPS



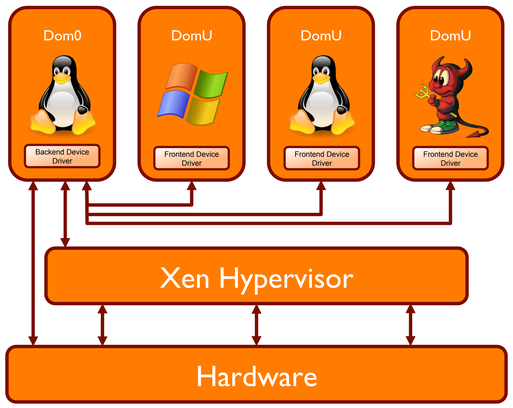
- Figure 1 depicts a machine with a variety of workloads running in different VMs
  - Database workloads on Alpha and Beta
  - Backup workload on Gamma
- Figure 2 depicts a 10-minute random-reads workload run on each VM
  - Each benchmark staggered by two minutes
- Figure 3 depicts a dynamic workload
  - All VMs initially run database benchmark
  - Gamma changes workload to backup after five minutes

Figure 3: Dynamic Workloads



Xen I/O Path

- Scheduler provides no QoS guarantees: nondeterministic
- DomU creates virtual I/O requests to blkback driver
- blkback driver sends requests to ring buffer in Dom0
- Dom0 services request via physical disk driver
- DomU is notified when request completes



Future Work

- Counter-based hard limiting
  - In a given timeslice, only allow N I/O operations
  - After N operations in timeslice are performed, pretend ring buffer is full
- DomU-based daemon inserts tagged requests
  - Variable based on current I/O load and priority policy
  - Fills ring buffer if approaching maximum allowed throughput
  - Dom0 discards tagged requests

Sponsored by Rackspace US, Inc.
Special Thanks to Gabe Westmaas