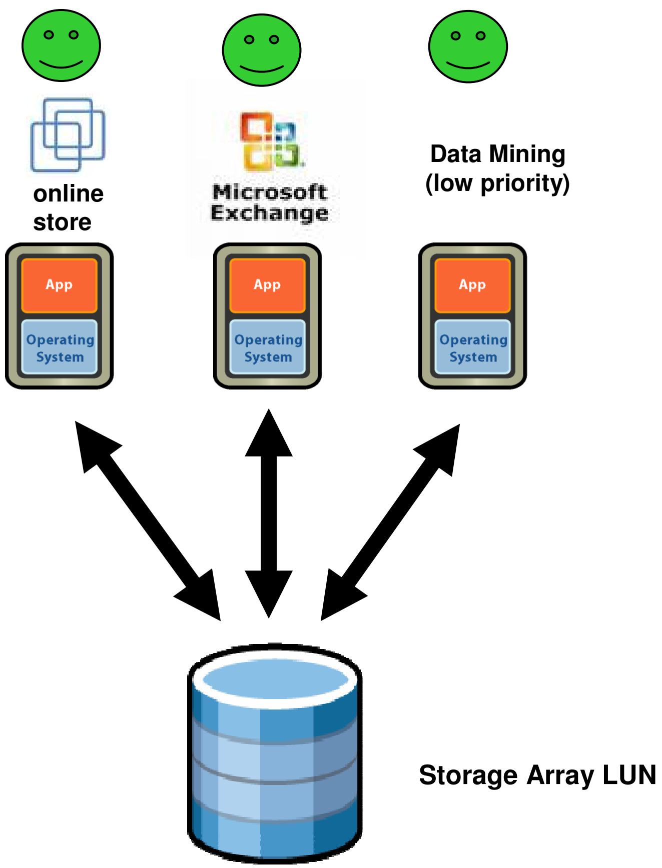
PARDA: Proportional Allocation of Resources for Distributed Storage Access

Ajay Gulati, Irfan Ahmad, Carl Waldspurger

Resource Management Team
VMware Inc.

USENIX FAST 09 Conference – February 26, 2009

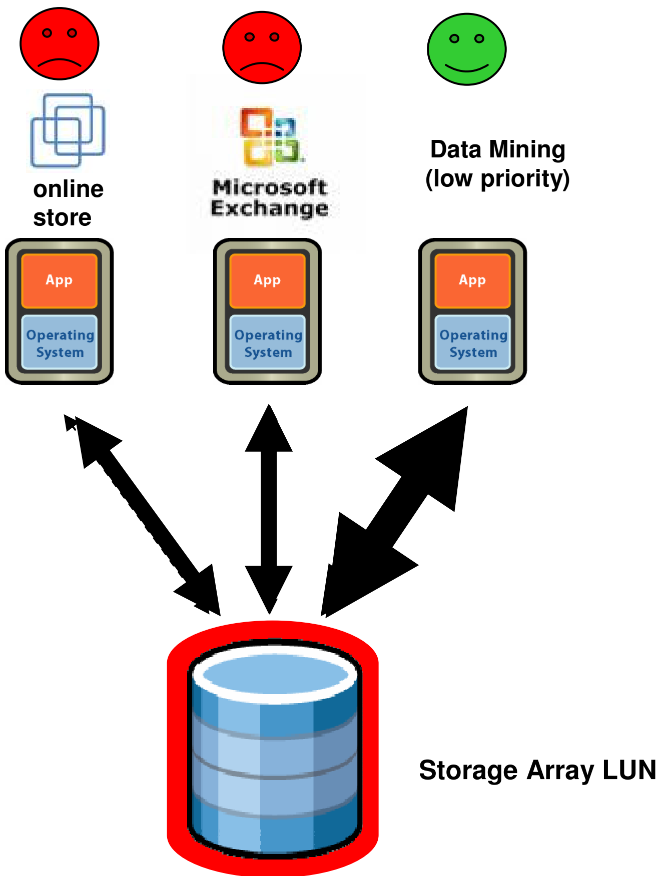
The Problem



online
store
Microsoft
Exchange
Data Mining
(low priority)
App
Operating
System
App
Operating
System
App
Operating
System
Storage Array LUN

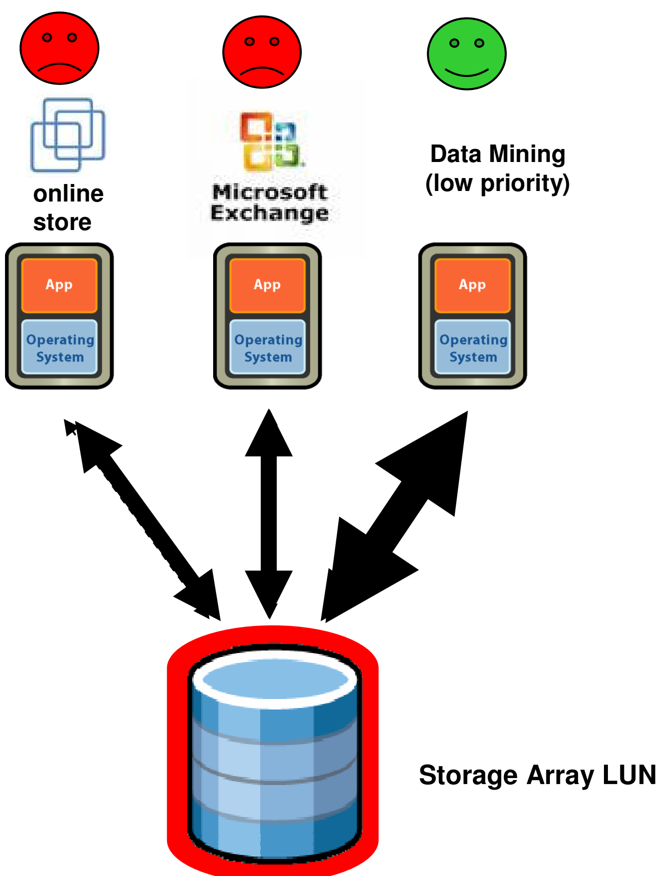
The Problem

What you see

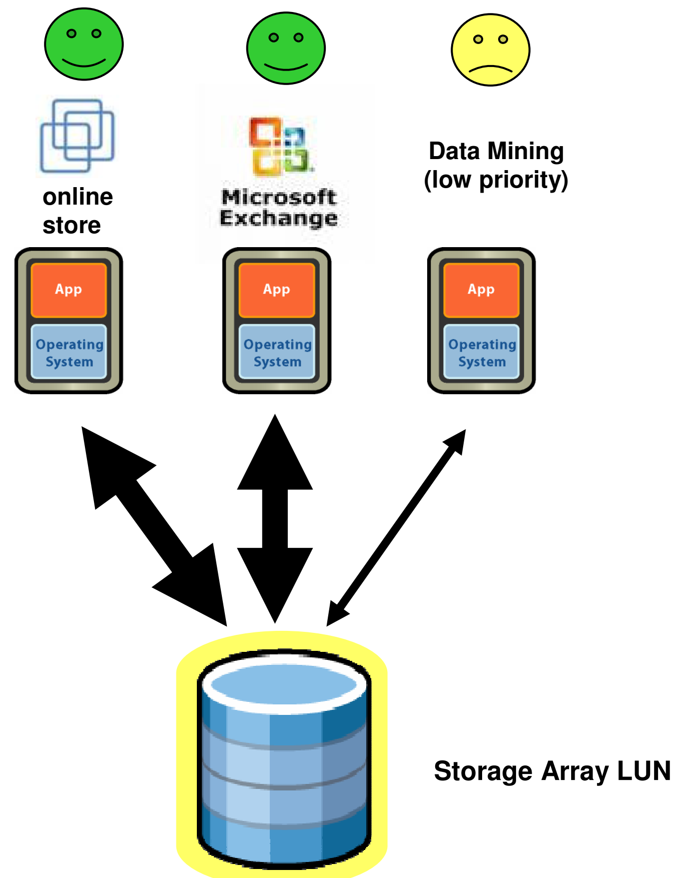


The Problem

What you see



What you want to see



Distributed Storage Access

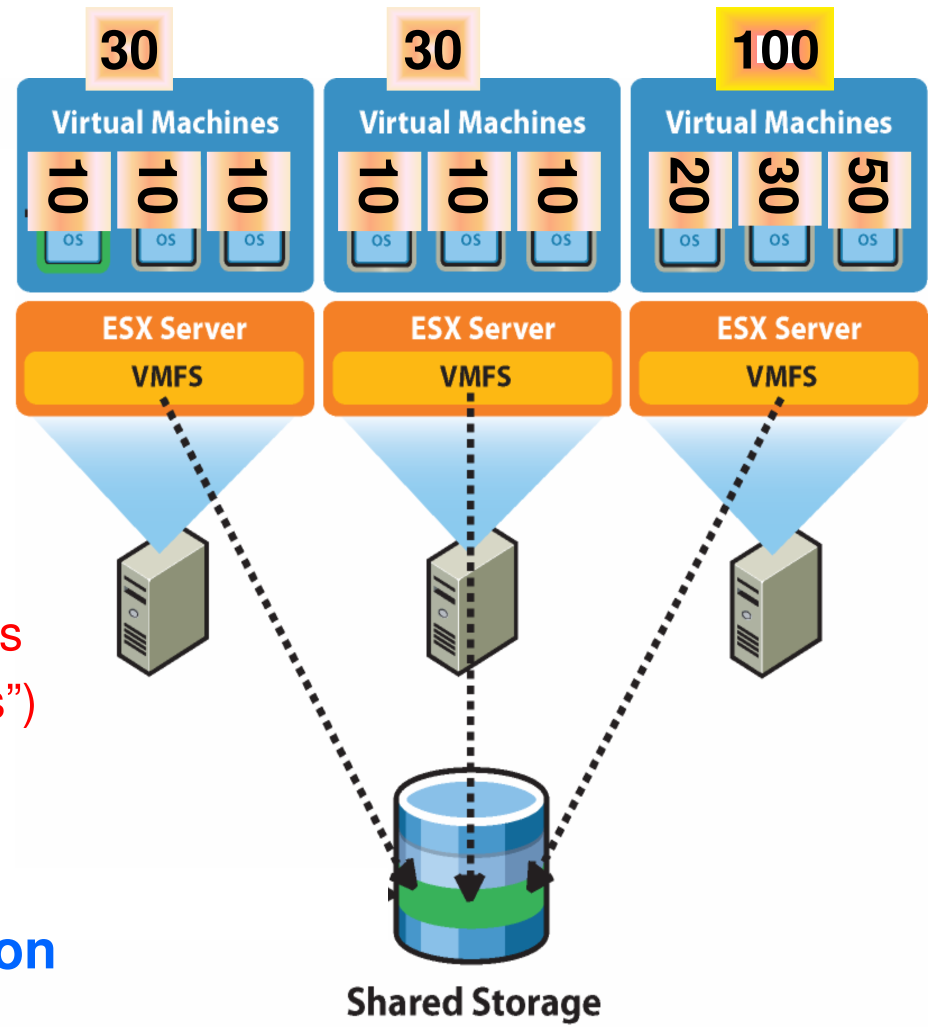
Setup

- VMs running across multiple hosts
- Hosts share LUNs using a cluster filesystem
- No central control on IO path

Issues

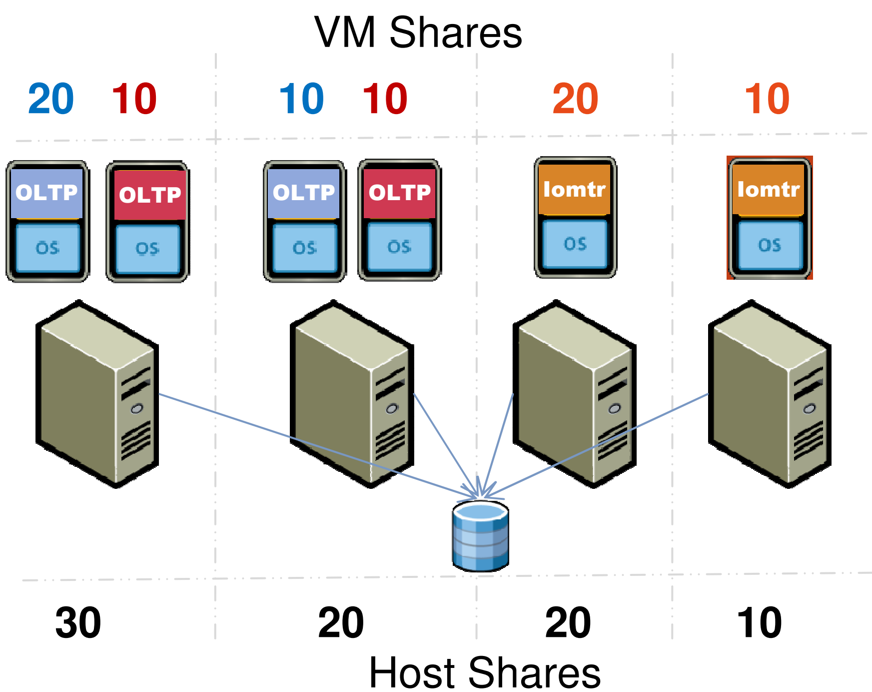
- Hosts adversely affect each other
- Difficult to respect per-VM allocations
  - Proportional shares (aka "weights")
  - Specify relative priority

Goal: **Provide isolation while maximizing array utilization**

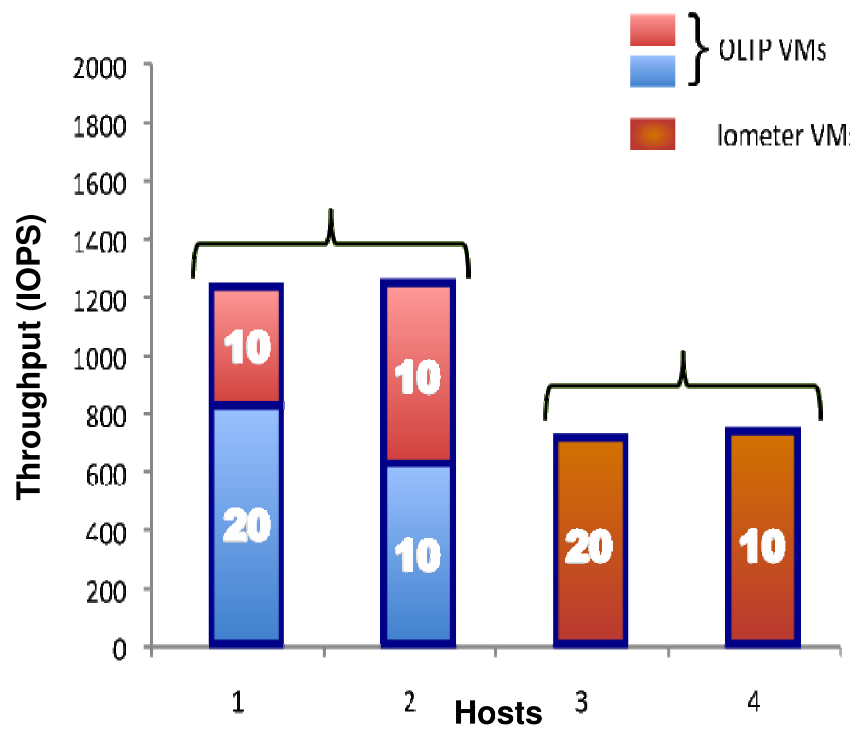


vmware

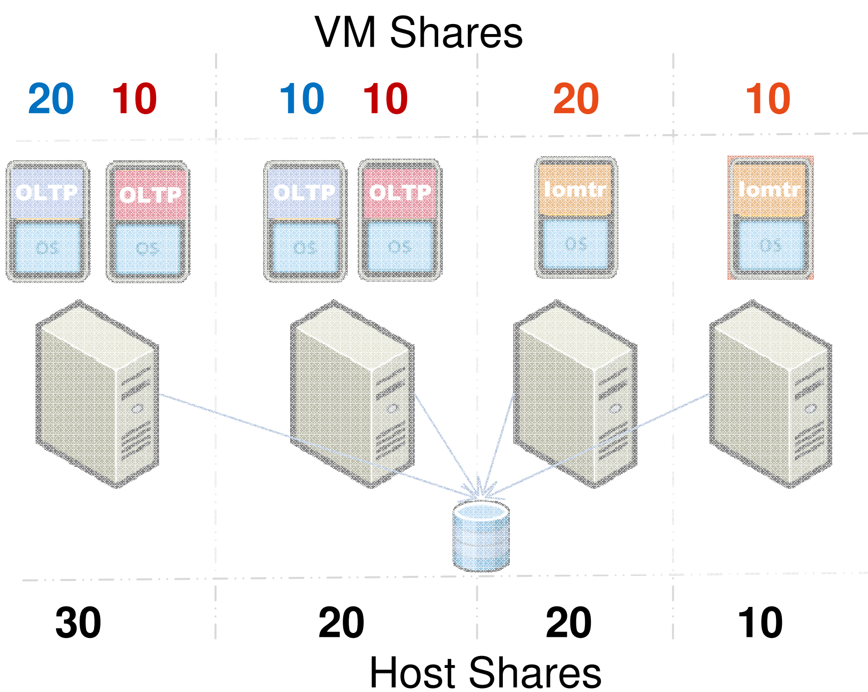
Host-local Scheduling Inadequate



- Local SFQ schedulers respect VM shares BUT not across hosts



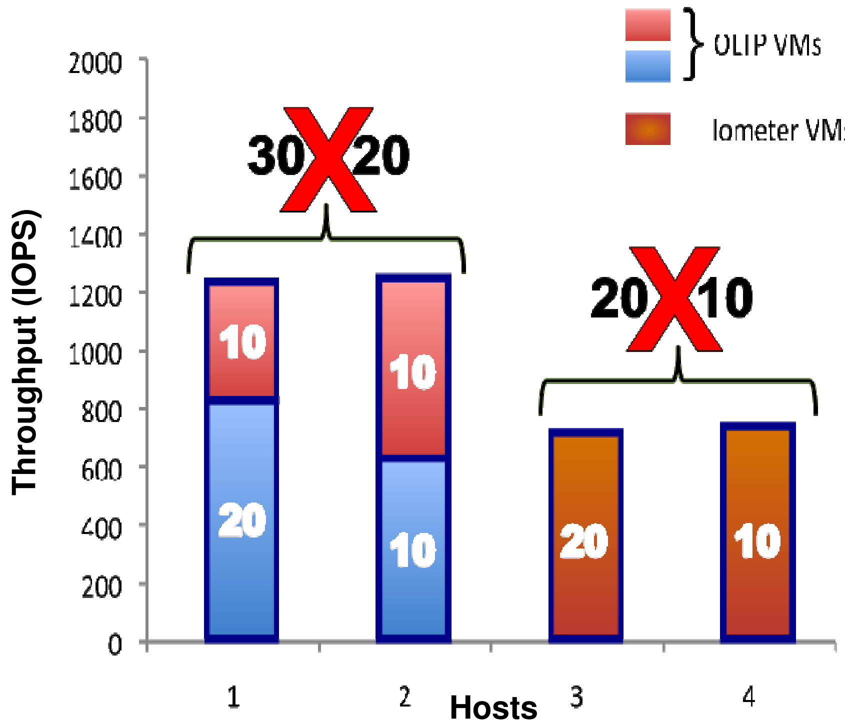
Host-local Scheduling Inadequate



- Local SFQ schedulers respect VM shares BUT not across hosts

- Hosts get equal IOPS

⇒ IOPS dependent on VM placement!



vmware

Outline

- Problem Context
- **Analogy to Network Flow Control**
- Control Mechanism
- Experimental Evaluation
- Conclusions and Future Work

Analogy to Network Flow Control

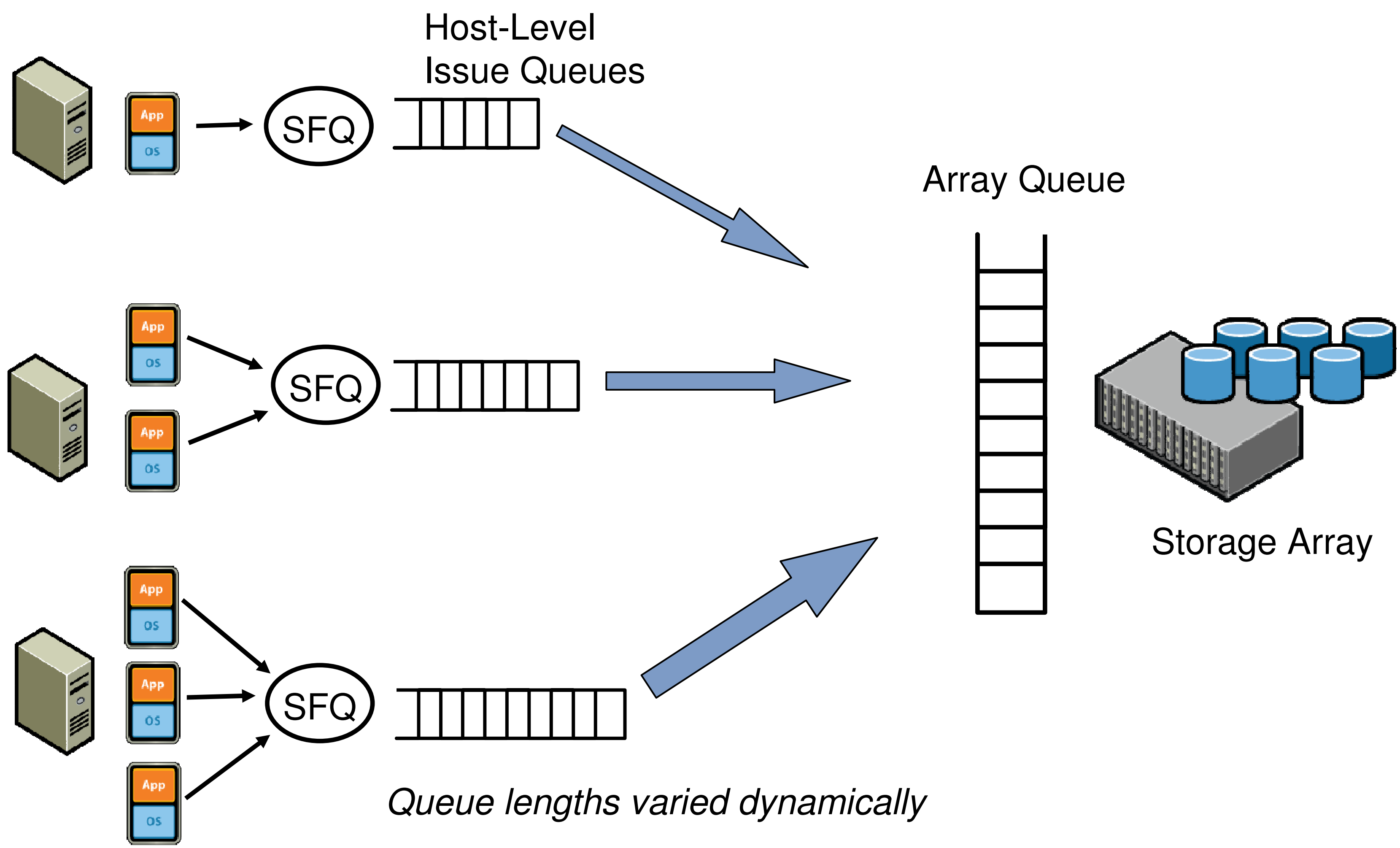
Networks	Storage
> Network is a black box	> Array is a black box
> Network congestion detected using RTT, packet loss	> Estimate array congestion using IO latency > Packet loss very unlikely
> TCP adapts window size	> Adapt number of pending IO requests (a.k.a. window size)
> TCP ensures fair sharing > FAST TCP* proportional sharing	> Adapt window size based on shares/weights

* Low *et. al.* FAST TCP: Motivation, Architecture, Algorithms, Performance. *Proc. IEEE INFOCOM '04*

Outline

- Problem Context
- Analogy to Network Flow Control
- **Control Mechanism**
- Experimental Evaluation
- Conclusions and Future Work

PARDA Architecture



Per-Host Control Algorithm

$$w(t+1) = (1-\gamma)w(t) + \gamma\left(\frac{\mathcal{L}}{L(t)}w(t) + \beta\right)$$

- Adjust window size ***w(t)*** using cluster-wide average latency ***L(t)***
  - $\mathcal{L}$: latency threshold, operating point for IO latency
  - β :proportional to aggregate VM shares for host
  - γ : smoothing parameter between 0 and 1
- Motivated by FAST TCP mechanism

Control Algorithm Features

$$w(t+1) = (1-\gamma)w(t) + \gamma\left(\frac{\mathcal{L}}{L(t)}w(t) + \beta\right)$$

- Maintain high utilization at the array
  - Overall array queue proportional to Throughput x $\mathcal{L}$
- Ability to allocate queue size in proportion to hosts' shares
  - At equilibrium, host window sizes are proportional to β
- Ability to control overall latency of a cluster
  - Cluster operates close to $\mathcal{L}$ or below

Unit of Allocation

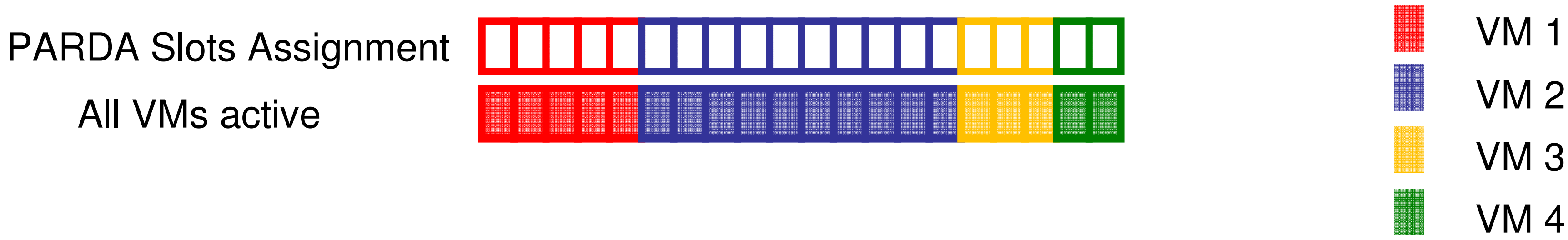
- Two main units exist in literature
  - Bandwidth (MB/s)
  - Throughput (IOPS)
- Both have problems
  - Using bandwidth may hurt workloads with large IO sizes
  - Using IOPS may hurt VMs with sequential IOs
- PARDA: allocate queue slots at array
  - Carves out array queue among VMs
  - Workloads can recycle queue slots faster or slower
  - Maintains high efficiency

Storage-Specific Issues

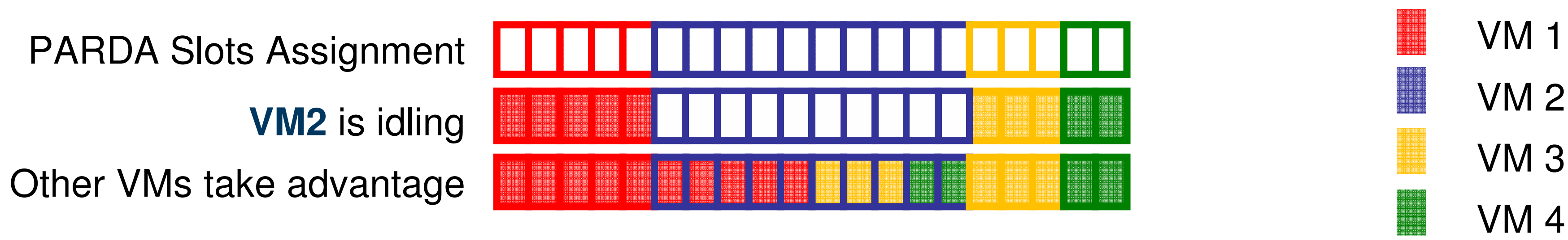
- Issues
  - Throughput varies by 10x depending on workload characteristics
  - IO sizes may vary by 1000x (512B – 512KB)
  - Array complexity: caching, different paths for read vs. write
  - Hosts may observe different latencies

- PARDA Solutions
  - Latency normalization for large IO sizes
  - Compute cluster-wide average latency using a shared file

Handling Bursts—Two Time Scales

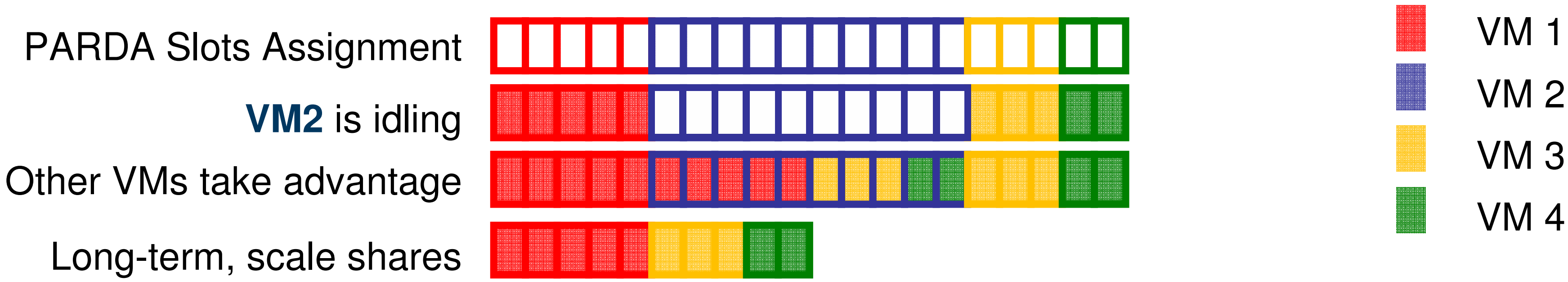


Handling Bursts—Two Time Scales



- Workload variation over short time periods
  - Handled by existing local SFQ scheduler
  - No strict partitioning of host-level queue

Handling Bursts—Two Time Scales



- Workload variation over short time periods
  - Handled by existing local SFQ scheduler
  - No strict partitioning of host-level queue
- VM idleness over longer term
  - Re-compute β per host based on VM activity
  - Effectively scale VM shares based on its utilization
  - Utilization computed as
    (# outstanding IOs) / (VM window size)

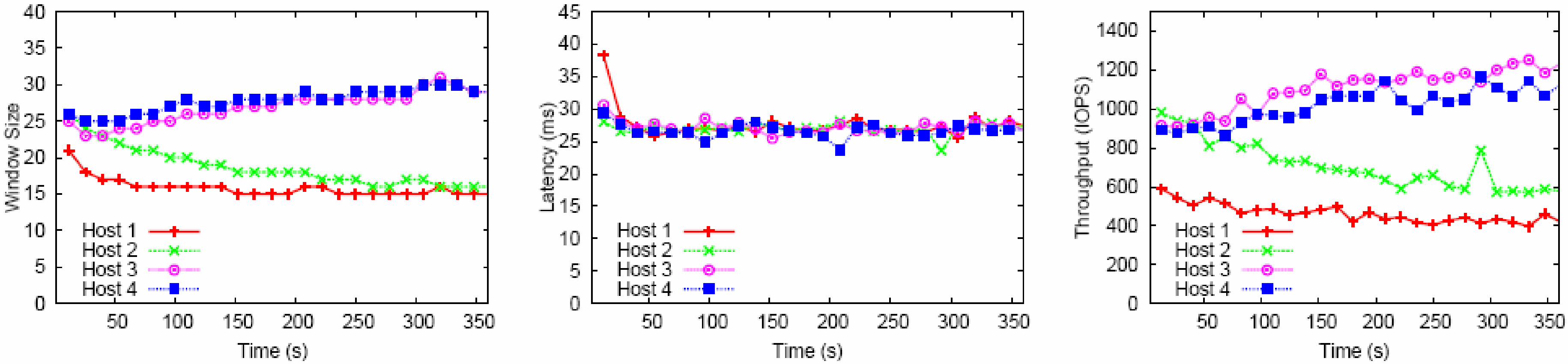
Outline

- Problem Context
- Analogy to Network Flow Control
- Control Mechanism
- **Experimental Evaluation**
- Conclusions and Future Work

Experimental Setup

- VMware Cluster
  - 1-8 hosts running ESX hypervisor
  - Each host: 2 Intel Xeon dual cores, 8 GB RAM
- FC-SAN attached Storage
  - EMC CLARiiON CX3-40 storage array
  - Similar results on NetApp FAS6030
- Two volumes used
  - 200 GB, 10 disks, RAID-0 (striped)
  - 400 GB, 5-disk, RAID-5

PARDA: Proportional Sharing



(a) Window Size (b) Latency (ms) (c) Throughput (IOPS)

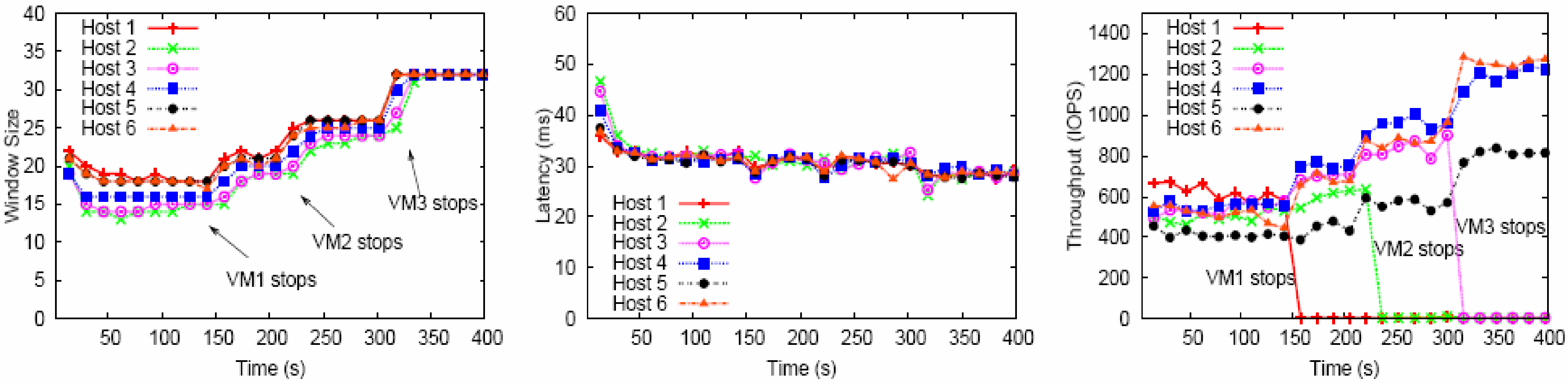
Window sizes are in proportion to shares	Latency close to 25 ms	IOPS match shares but affected by other factors

Aggregate IOPS with/without PARDA (3400 vs 3360)

Setup:

- 4 Hosts, shares – 1 : 1 : 2 : 2
- Latency threshold $\mathcal{L}$ = 25ms
- Workload – 16KB, 100% reads, 100% random IO

PARDA: Dynamic Load Adaptation



(a) Window Size (b) Latency (ms) (c) Throughput (IOPS)

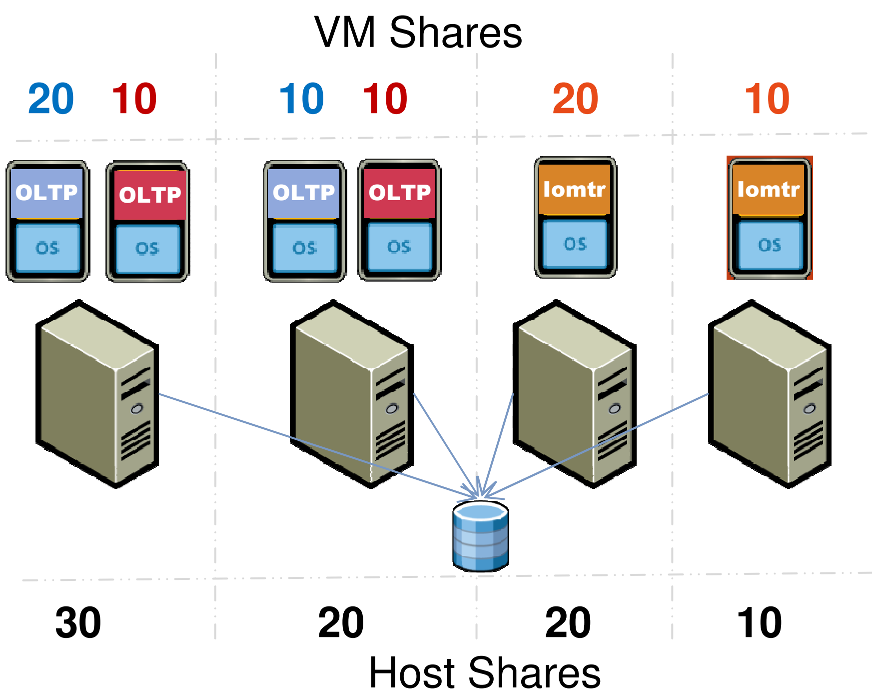
PARDA adapts to load	Latency close to 30 ms	IOPS increase with increase in window size

Aggregate IOPS with/without PARDA (3090 vs 3155)

Setup:

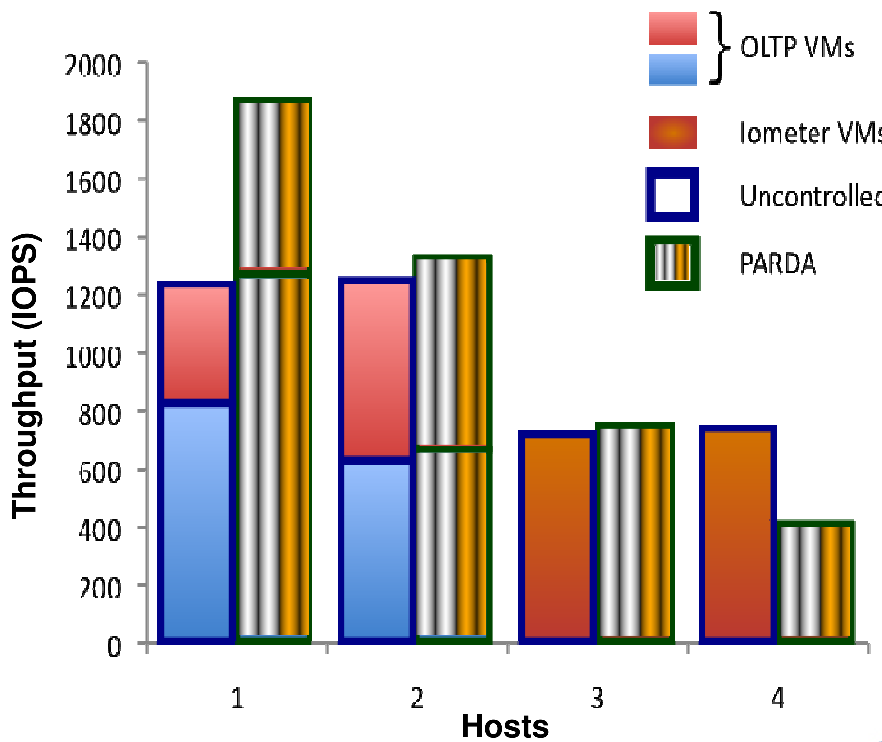
- 6 Hosts, equal shares, uniform workload
- Latency threshold $\mathcal{L}$ = 30ms
- Three VMs are stopped at 145, 220 and 310 sec

PARDA: End-to-End Control



Setup:

Latency threshold $\mathcal{L}$ = 25ms

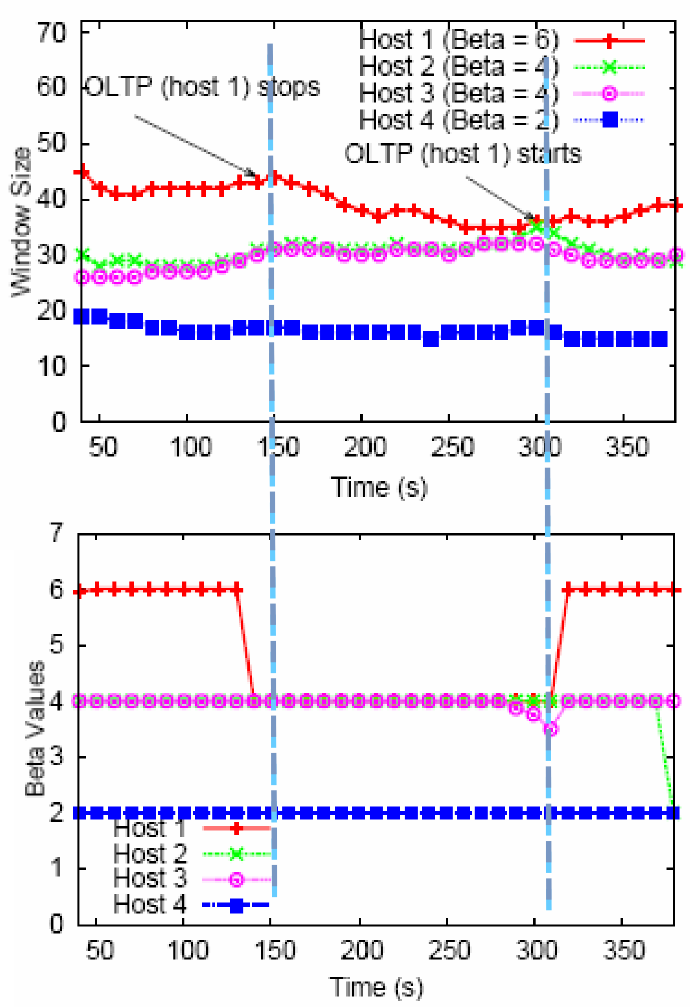


With PARDA:

Shares are respected

independent of VM placement

PARDA: Handling Bursts



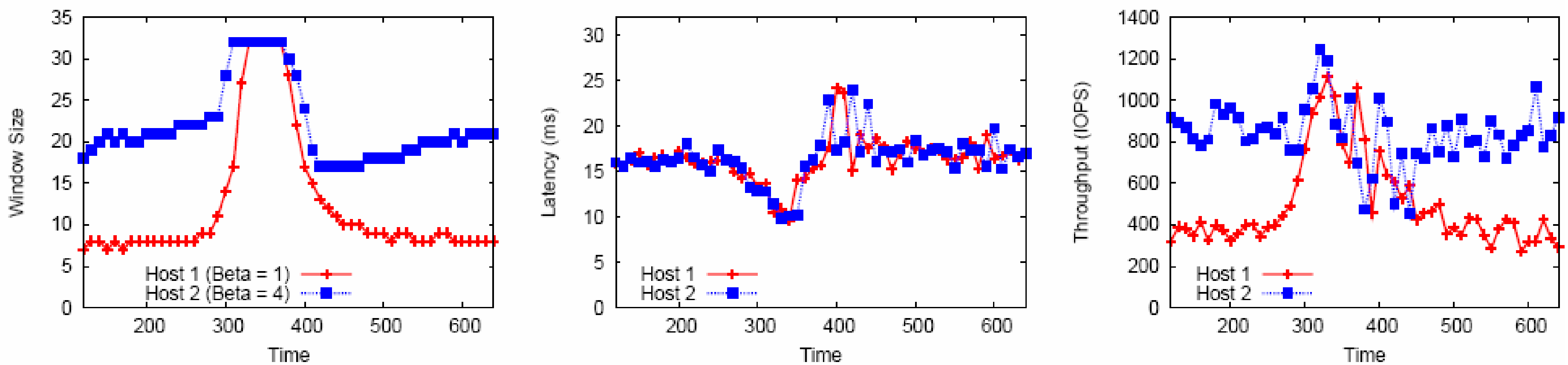
Setup:

- One VM idles from 140 sec to 310 sec

Result:

- PARDA is able to adjust β values at host
- No undue advantage given to VMs sharing the host with idle VM

PARDA: SQL Workloads



(a) Window Size (b) Latency (ms) (c) Throughput (IOPS)

Setup:

- 2 Hosts, 2 Windows VMs running SQL server (250 GB data disk, 50 GB log disk)
- Shares 1 : 4
- Latency threshold $\mathcal{L}$ = 15ms

		Uncontrolled			PARDA		
Host	VM Type	*OPM*	*Avg Lat*	T_h, L_h	β_h	*OPM*	*Avg Lat*
1	SQL1	8799	213	615, 20.4	1	6952	273
2	SQL2	8484	221	588, 20.5	4	12356	151

Without PARDA:

- **No Fairness Across hosts**
- **Both hosts get ~600 IOPS**

With PARDA:

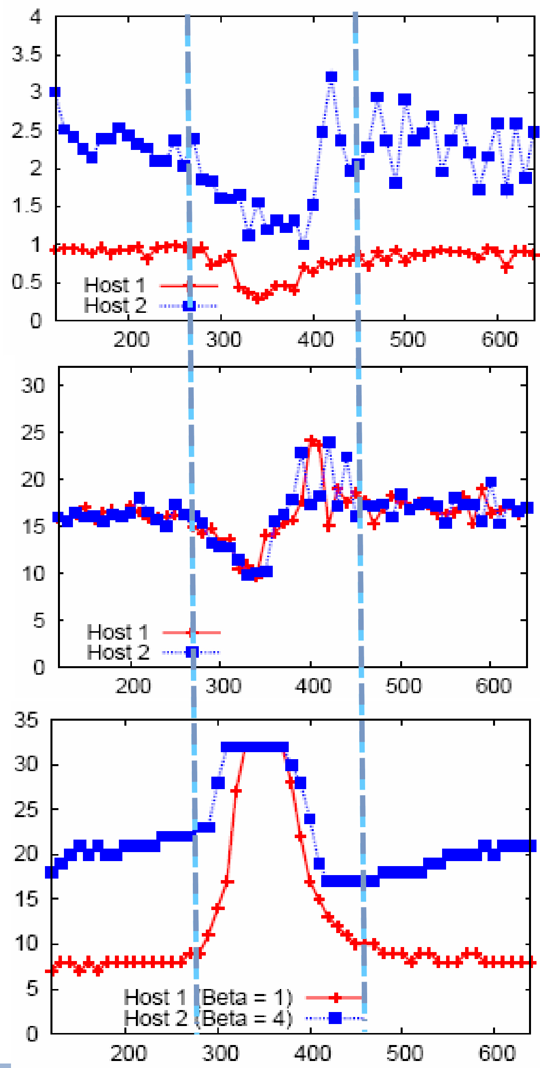
- **Shares are respected *across hosts***
- **Host 1,2 with shares 1:4 get 6952 and 12356 OPM**

PARDA: Burst Handling

Beta values

Latency

Window size



Result:

- PARDA adjusts β values at host with the change in pending IO count
- VM2 with high shares is unable to fill its window
- IOPS and OPM are differentiated based on β values

Evaluation Recap

- Effectiveness of PARDA mechanism
  - Fairness
  - Load or throughput variations
  - Burst handling
  - End-to-end control
  - Enterprise workloads
- Evaluation of control parameters (without PARDA)
  - Latency variation with workload
  - Latency variation with overall load
  - Queue length as control mechanism for fairness

Outline

- Problem Context
- Analogy to Network Flow Control
- Control Mechanism
- Experimental Evaluation
- **Conclusions and Future Work**

Conclusions and Future Work

- PARDA contributions
  - Efficient distributed IO resource management – without any support from storage array
  - Fair end-to-end VM allocation, proportional to VM shares
  - Control on overall cluster latency
- Future work
  - Latency threshold estimation or adaptation?
  - Detect and handle uncontrolled (non-PARDA) hosts
  - NFS adaptation

Questions ...

{agulati,irfan,carl}@vmware.com

Storage in Virtualization BoF tonight @7:30 pm