A Practical Implementation of Clustered Fault Tolerant Write Acceleration in a Virtualized Environment

Deepavali Bhagwat, Mahesh Patil, Michal Ostrowski,
Murali Vilayannur, Woon Jung, Chethan Kumar

FAST 2015

I/O Acceleration in Virtualized Datacenters

- As virtualized datacenters scale, they experience I/O bottlenecks
  - Virtual Machine (VM) density increases
  - Cumulative VM I/O increases
  - J. Shafer, "I/O Virtualization Bottlenecks in Cloud Computing Today," WIOV'10

- Provisioning better/more storage
  - Disruptive, temporary fix
  - Buying capacity to solve performance

- Host-side flash
  - Locate application working set at the beginning of the I/O path to accelerate I/O
  - A non-disruptive approach

pernixdata

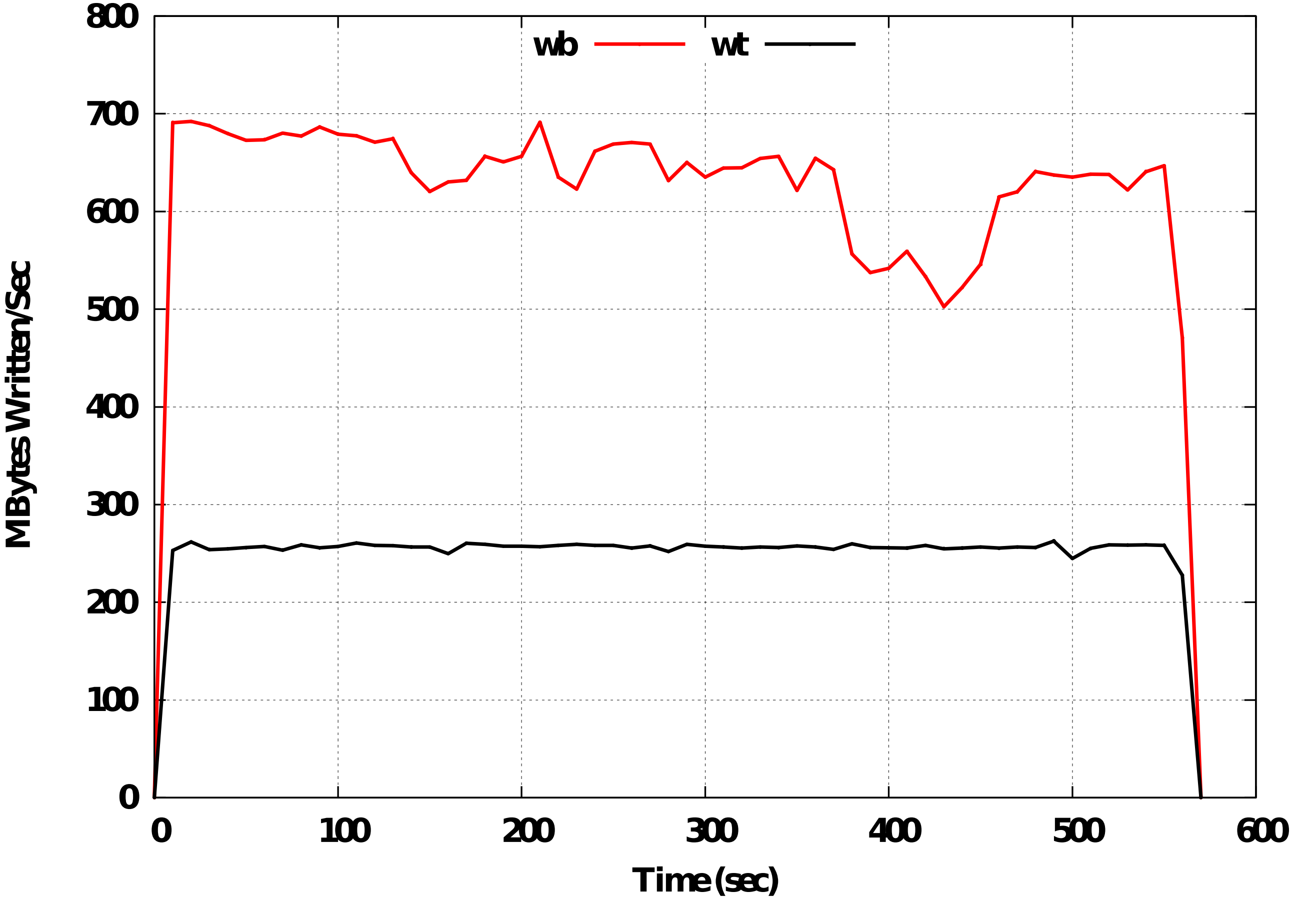
Host-side Flash for Accelerating I/O

- Host-side flash to accelerate reads alone (Write-Through)
  - Reads are issued first to flash and to SAN on a cache miss
  - Writes issued to flash and SAN and acknowledged after SAN completion
  - VM writes experience SAN latencies
  - Byan *etal*.,"Mercury: Host-side flash caching for the data center," Mass Storage '12
  - Qin *etal*.,"Reliable write-back for client-side flash caches," USENIX ATC '14

- Host-side flash to accelerate reads and writes (Write-Back)
  - Reads are cached as in Write-Through
  - Writes issued to flash only and acknowledged after flash completion
  - Writes periodically flushed to the SAN
  - VM writes experience flash latencies
  - Holland *etal*., "Flash caching on the storage client," USENIX ATC '13

pernixdata

Benefits of Using Host-side Flash for Write Acceleration



Cumulative Throughput of two VMs:

- Microsoft Exchange Server JetStress: 32K random reads/writes, 14K sequential writes
- fio: 64K sequential writes

Challenges using Host-side Flash for Write Acceleration

- Preserving VM mobility in virtualized environments
  - Resource Management
  - Power Management
  - High Availability

- Sustained writes

- Fault Tolerance

- We introduce FVP:
  - Seamless VM migration
  - Flow Control to gracefully handle sustained writes
  - Fault Tolerance by replicating VM writes to peers

pernixdata

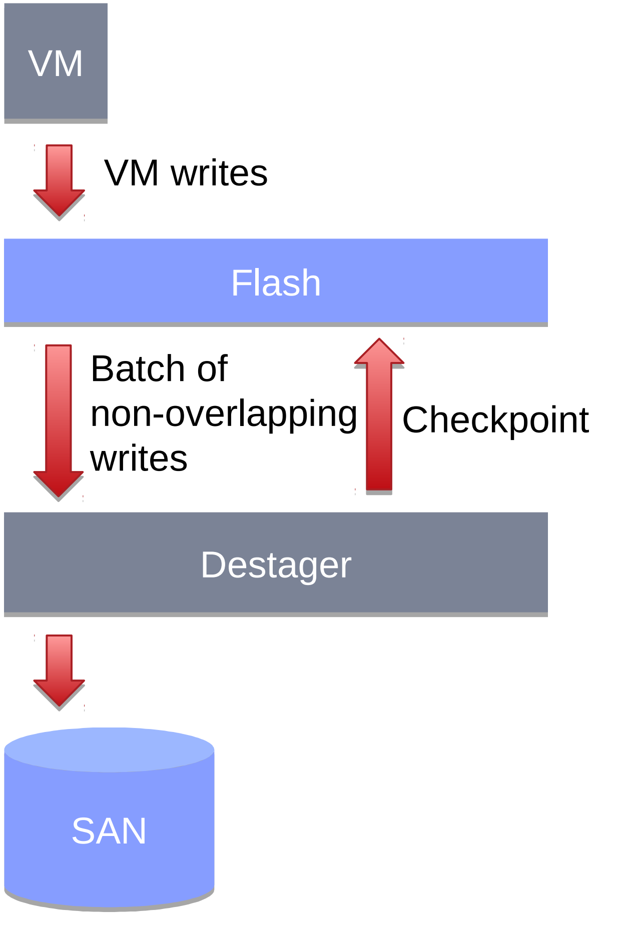
Acceleration Policies

- Write-Through (*wt*)
- Write-Back (*wb*)
- Write-Back with Peers (*wbp*)
  - Writes issued to flash and to peer hosts
  - Acknowledged to the VM after flash and peer completion
  - VM writes experience MAX(network, flash) latencies

pernixdata

The Destager

- Dirty VM writes periodically flushed to the SAN
- Each write is marked with a monotonically increasing serial number at entry
- Writes are batched
  - Writes in a batch do **not** overlap
  - Writes in a batch are issued concurrently to the SAN
- A checkpoint record is persisted to the flash after every write in a batch is complete
- To allow for fair share of SAN bandwidth
  - The destager cycles through VM write batches
  - The batch size is capped



What is a Checkpoint?

- A checkpoint consists of
  - Serial number
  - VM UUID

- For *wbp* VMs, the checkpoint is transmitted to peers.

- The primary and peer host persist checkpoint onto their host-side flash

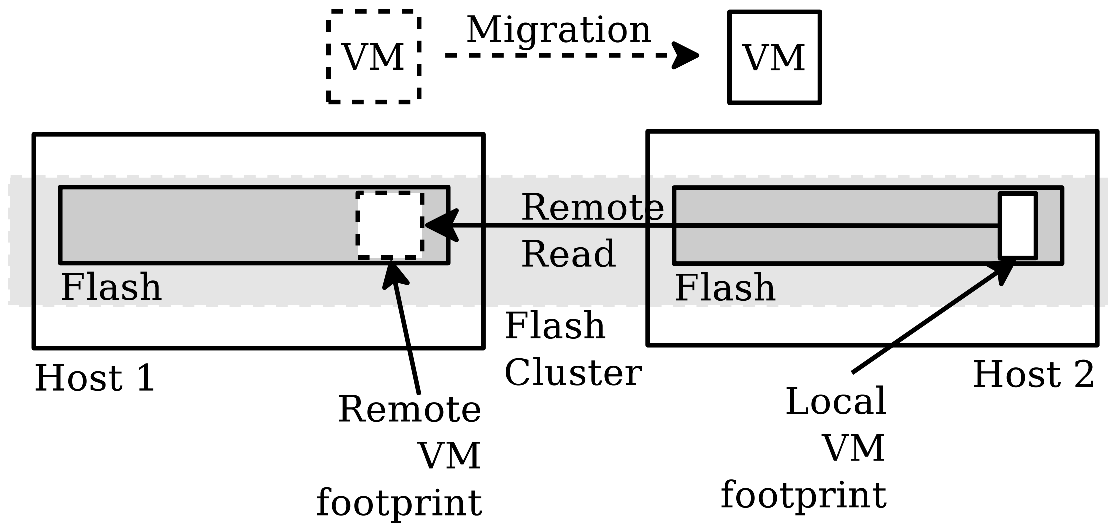
- The primary and peer hosts evict writes whose

  Serial number <= Checkpoint(serial number)

- Checkpoints reduce recovery time

pernixdata

Seamless VM Migration



- Every host has access to every other host's flash
- After migration VM reads are transmitted to the previous host
- The new host builds up the VM's footprint
- VM reads experience (network + flash) latencies
- After a certain period of time or after a certain number of cache misses, the new host stops transmitting reads to the previous host

Flow Control

- Sustained writes fill up flash space allocated to a VM
- VM has to be stalled, experiences degraded performance
- Flow Control:
  - Slow down VM to allow the destager to catch up
  - Avoid VM stall
- In the worst case, VM experiences SAN latency (which they would without FVP)
- FVP uses heuristics to trigger flow control
- Most applications are bursty, short write bursts gracefully absorbed

Fault Tolerance with Host-side flash

- In case of failure, SAN is in an inconsistent state with respect to the VM
- Affected VMs could be migrated to other hosts for High Availability
- Data corruption possible
- On-disk locks prevent hosts from corrupting VM data on the SAN
  - A lock for every VM, ownership arbitrated by the Virtual Machine File System
  - Only one host can acquire lock. That host is eligible to issue I/Os on behalf of the VM
  - Locks are persisted on the VM's datastore
- The lock contents
  - VM's last checkpoint (serial number, VM UUID)
  - VM acceleration policy
- A copy of the lock file is kept in a read only lock file

pernixdata

Fault Tolerance

- Flash Failure
- Host Failure
- Network Failure
- SAN Failure

pernixdata

Flash Failure

- Host relinquishes locks for affected VMs
- VM in *wt*: No data loss
- VM in *wb*: Recovery not possible. VM stalled.
- VM in *wbp*: VM stalled. Recovery via Online Replay
  - Peers periodically attempt to acquire locks. One peer succeeds.
  - The successful peer destages replicated writes from the last checkpoint.
  - Releases lock
- Migrated *wb/wbp* VM (HA)
  - New host acquires lock
  - Detects VM *wb/wbp* policy
  - Infers dirty writes still pending
  - Stalls the VMs
  - Releases locks
  - Periodically polls read only lock for waiting for a peer to complete online replay

pernixdata

Host Failure

- Virtual Machine File System releases locks as part of failure detection
- VM in *wt:* No data loss
- VM in *wb* via Offline Replay
  - On recovery, host acquires locks
  - FVP scans the flash device and destages all writes after the last checkpoint
  - Checkpoint is regularly updated in the lock file
  - VMs kept stalled until their writes are destaged
- VM in *wbp* via Online Replay
- Migrated *wb/wbp* VM: New host stalls the VMs until replay complete

pernixdata

Cascading Failures, Distributed Recovery

- Checkpoints persisted regularly to speed up recovery from cascading failures.
- FVP can recover from up to p flash failures.
- FVP can recover from multiple host failures. All metadata for writes + checkpoint persisted to flash.
- Recovery is distributed.

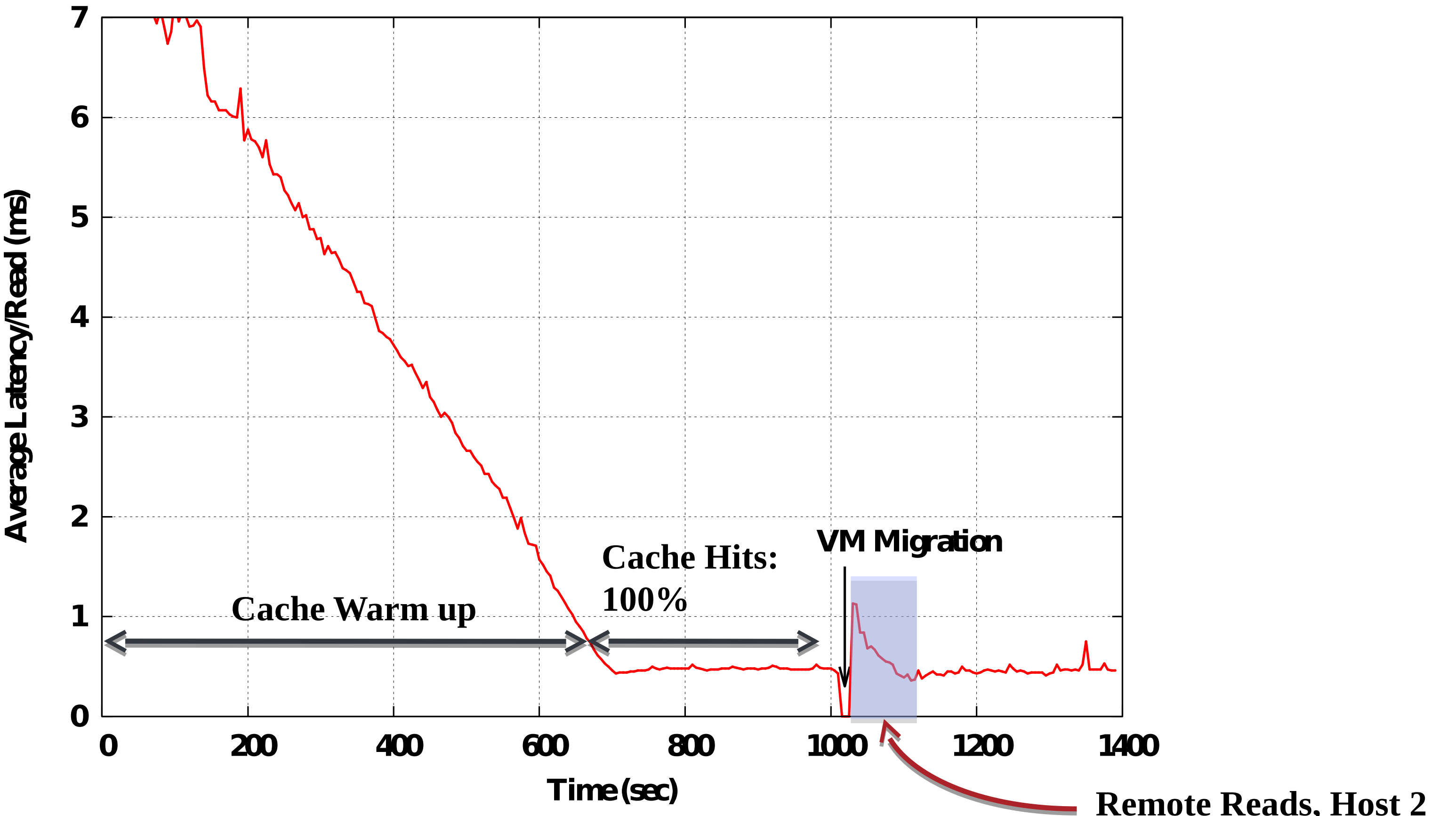
pernixdata

Experimental Evaluation

- VM migration
- How FVP absorbs short write burst.
- How FVP uses flow control to handle sustained write bursts.
- Fault Tolerance, Cost vs. Benefits.

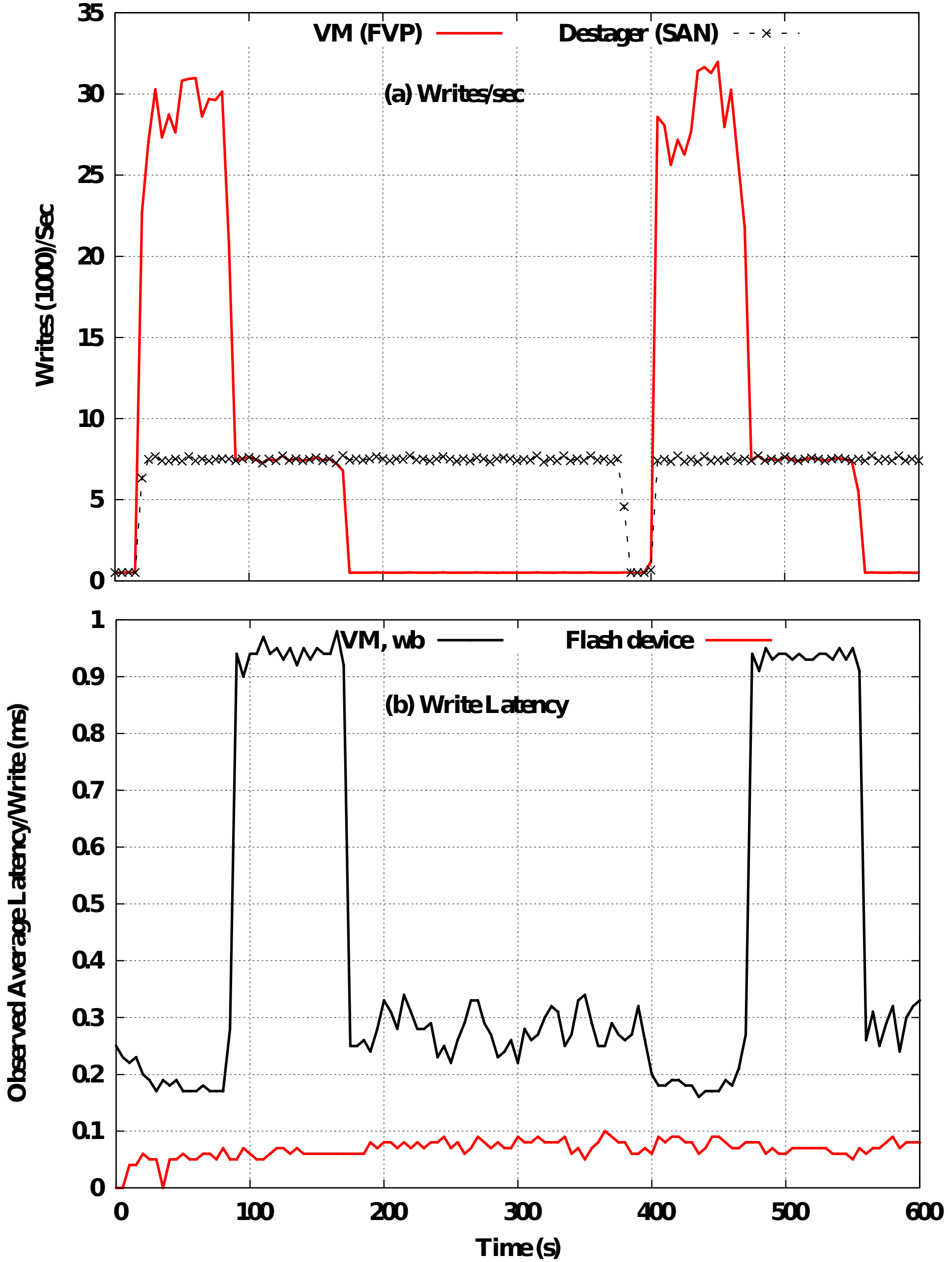
pernixdata

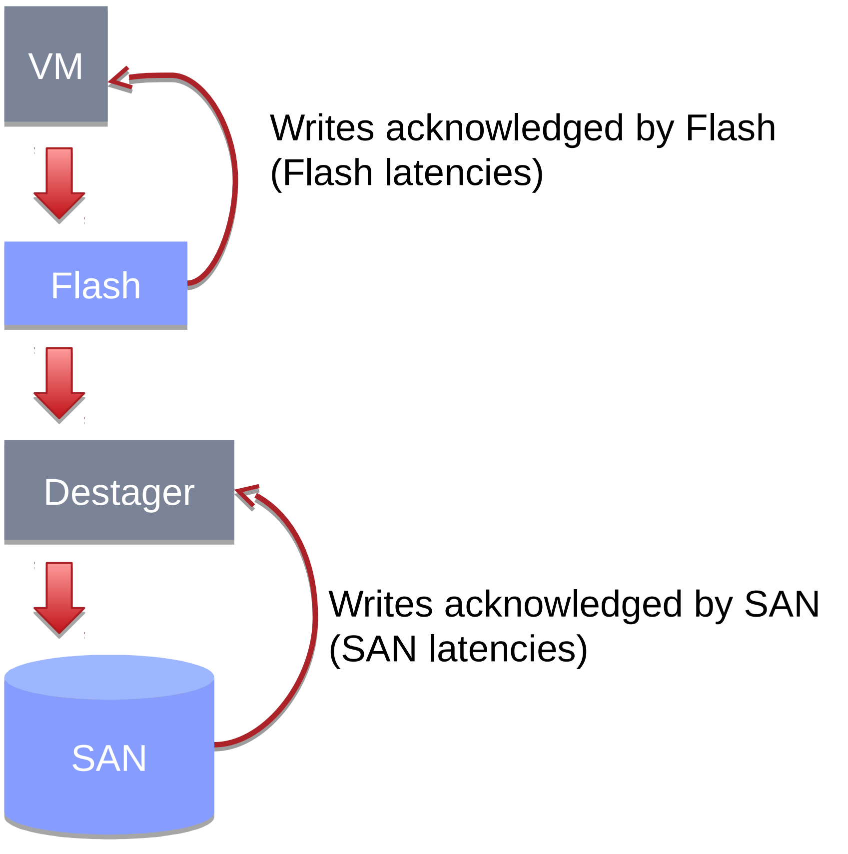
VM Migration



- Random 4K reads (Iometer)
  - Latency reduces as cache hits increase
  - After VM migration
    - Remote reads incur additional network latency
    - New host build VM footprint.
    - Latencies reduce as cache hits increase on new host.

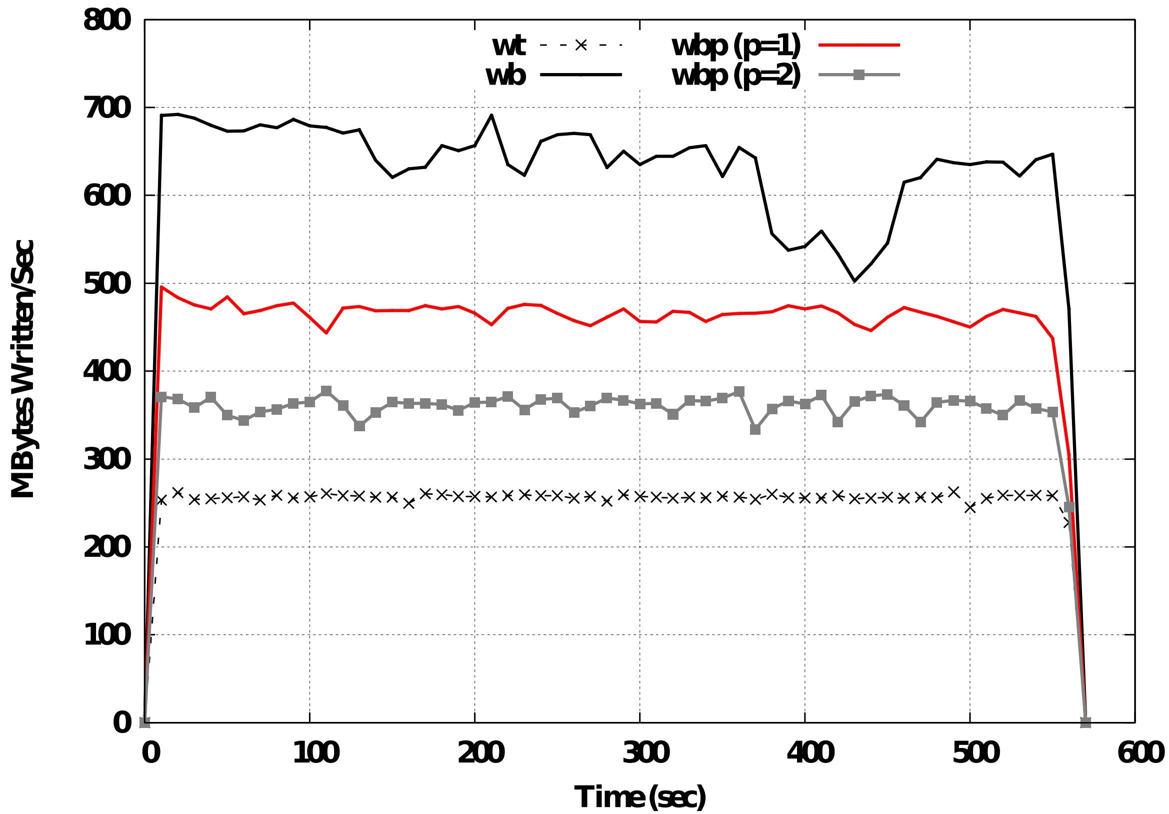
Sustained Write Bursts





pernixdata

Fault Tolerance Cost vs. Benefits



- Two VMs: Microsoft Exchange Server JetStress (reads and writes) and Ubuntu (writes)
- Replicating writes across peers incurs additional network latencies
- Peering reduces VM throughput, but protects against failures
- Throughput with peers is still better than only *wt*.
- Peering = fault tolerance + better acceleration than *wt*

Conclusions and Future Work

- FVP achieves seamless fault tolerant write back acceleration while preserving VM mobility (DRS, HA)
- Absorbs short write burst
  - Masks VMs from SAN latency spikes
  - Preserves VM performance predictability to help deliver on SLA objectives
- FVP handles sustained write bursts gracefully using Flow Control
- Future work: Building more intelligence/adaptability into FVP

A Practical Implementation of Clustered Fault Tolerant Write Acceleration in a Virtualized Environment

Deepavali Bhagwat, Mahesh Patil, Michal Ostrowski,
Murali Vilayannur, Woon Jung, Chethan Kumar

FAST 2015