Alleviating I/O Interference via Caching and Rate-Controlled Prefetching without Degrading Migration Performance

Morgan Stuart
Tao Lu
Xubin He

Storage Technology & Architecture Research Lab
Electrical and Computer Engineering Dept., Virginia Commonwealth University

Parallel **D**ata **S**torage **W**orkshop
November 16, 2014

Summary

1. Virtualization and Migration

2. Migration Induced Storage I/O Interference

3. Storage Migration Offloading
   a. Rate-Controlled storage read
   b. Caching the migrating VM’s accesses
   c. Prefetching bulk data

Migration Overview

- Virtual Machine (VM) adoption is huge
  - Flexibility for enterprise datacenters, HPC, and cloud

- Live Migration is a key enabler
  - Move a *running* VM **without** shutting down
  - Federate and increase manageability

Migration Data

- Early migration required **shared storage**
  Clark et al. (NSDI'05)
  - **Source** and **destination** could both access virtual disk
    - Only the memory and state required transfer

- Now capable of **full migrations** *Bradford et al. (VEE'07)*
  - Virtual disk must be moved as well
    - Much more data (Avg. ~60 GB Cloud vDisk *Birke et al. (FAST'14)*)

Progress in Migration Research

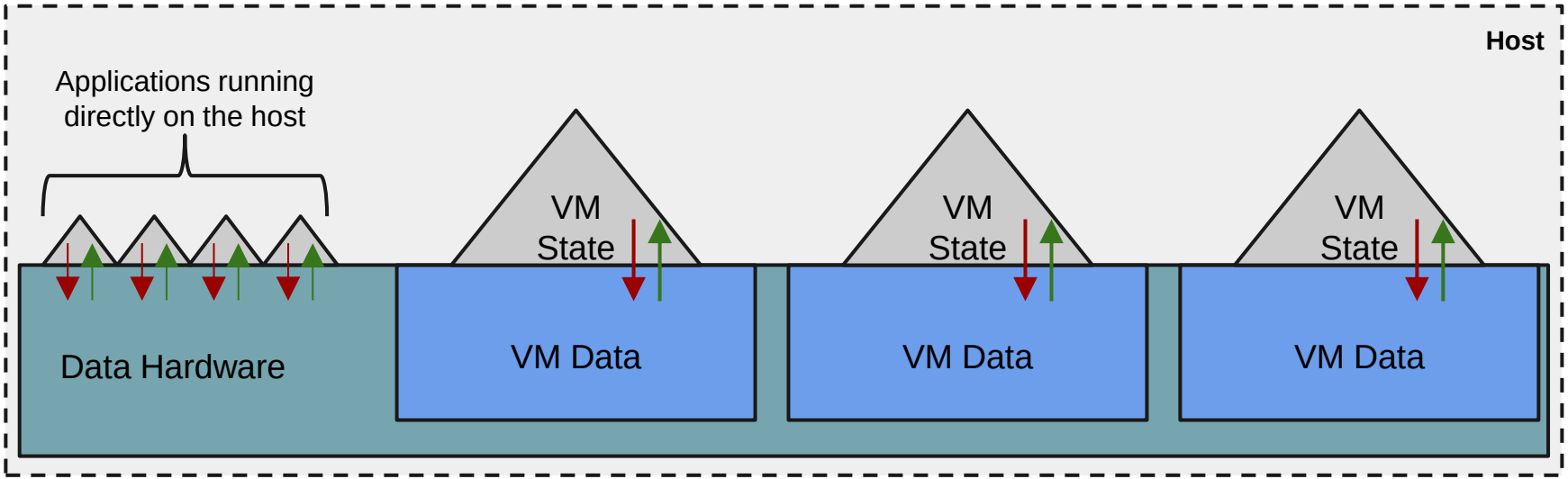
- Focus on **migration performance**
  - Reduce migration latency
    - Time between migration start and complete
  - Reduce migration downtime
    - The length of ***stop-and-copy***

- General strategy
  - Reduce amount of data to transfer

    Pierre et al. (Euro-Par'11), Al-Kiswany et al. (HPDC'11), Koto et al. (APSYS'12)
  - Avoid retransmissions
    - Zheng et al. (VEE'11)

Progress in Migration Research

Shared demand for a resource can create **interference**



Understanding Interference

- Fundamentally similar to any other interference
  - VMs contend for a resource...and the hypervisor can’t quite deliver
  - Leads to VM **performance degradation**

- Recent work has target VM interference
  - Primarily application level
    - *Chiang and Huang (SC’11), Mars et al. (MICRO-44), Nathuji (EuroSys’10)*

Migration Interference

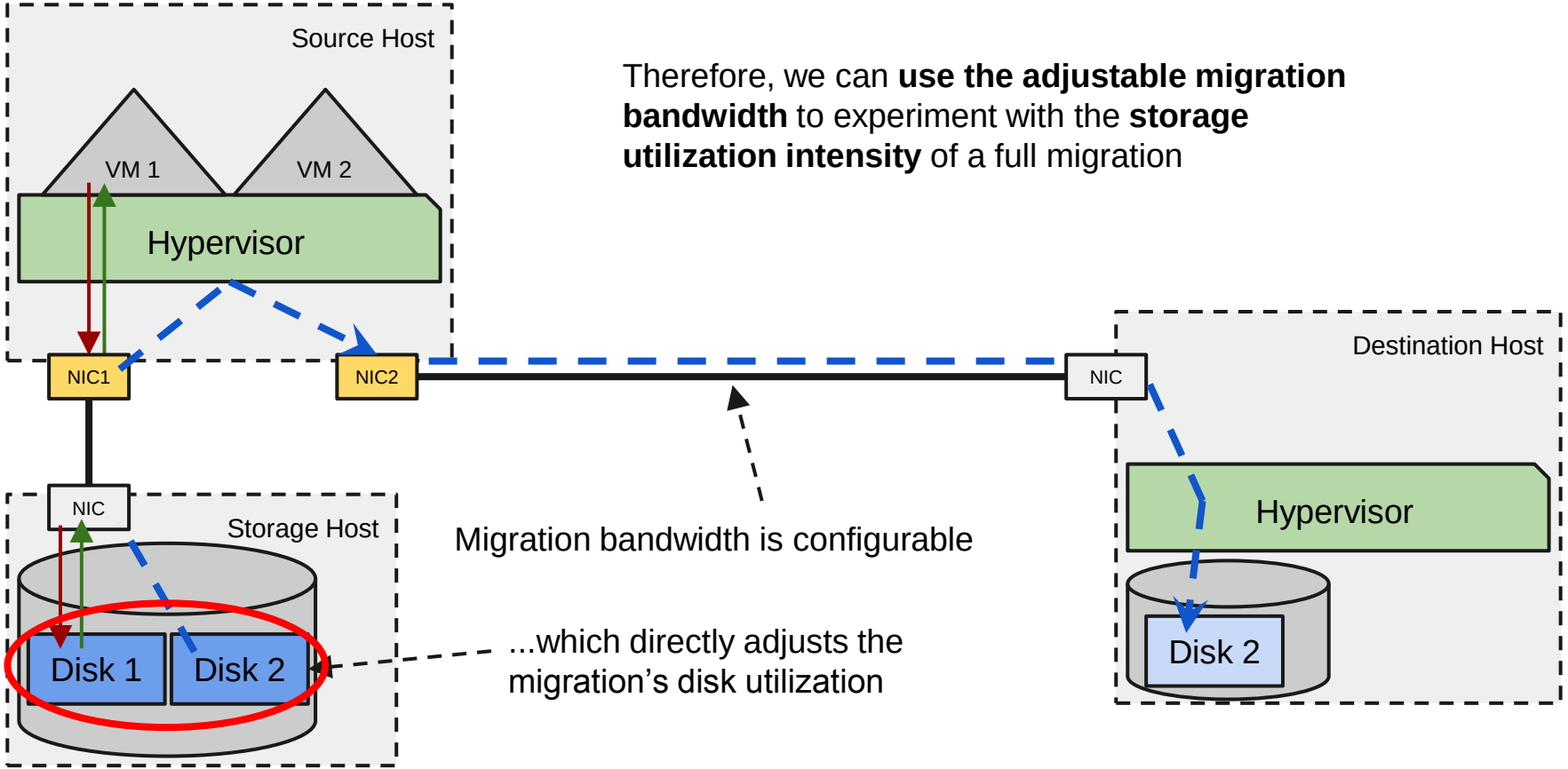
- Migration causes undeniable interference
  - Some work has addressed network, memory, and CPU
    - *Xu, Liu, et al. (Transactions on Computers, 2013)*

- Storage is often the performance bottleneck
  - How does storage migration impact its performance?

Migration Interference

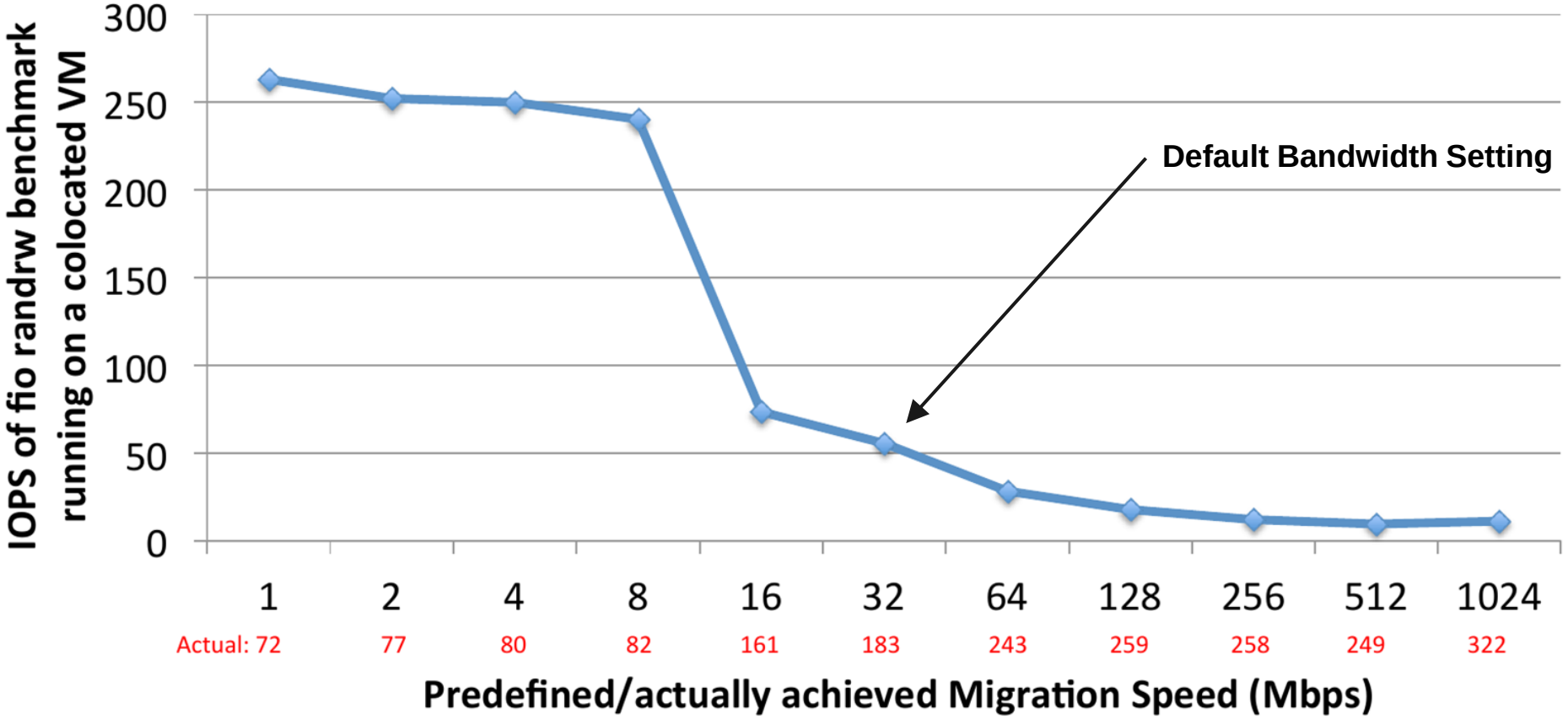
- Tests on KVM-QEMU
  - Two VMs located on the same **source host**
    - Virtual disks both placed on RAID-6 (8 disks) over **NFS**
      - *Migration traffic and NFS mounted on separate networks*
    - **1st VM** runs an IO benchmark
      - ***fio:*** *random R/W across a 1GB file @ 2MB/s*
    - **2nd VM** is idle
  - 2nd VM is migrated to **destination host**
    - Virtual disk is stored to a local drive here

Migration Interference



Therefore, we can **use the adjustable migration bandwidth** to experiment with the **storage utilization intensity** of a full migration

Migration Interference



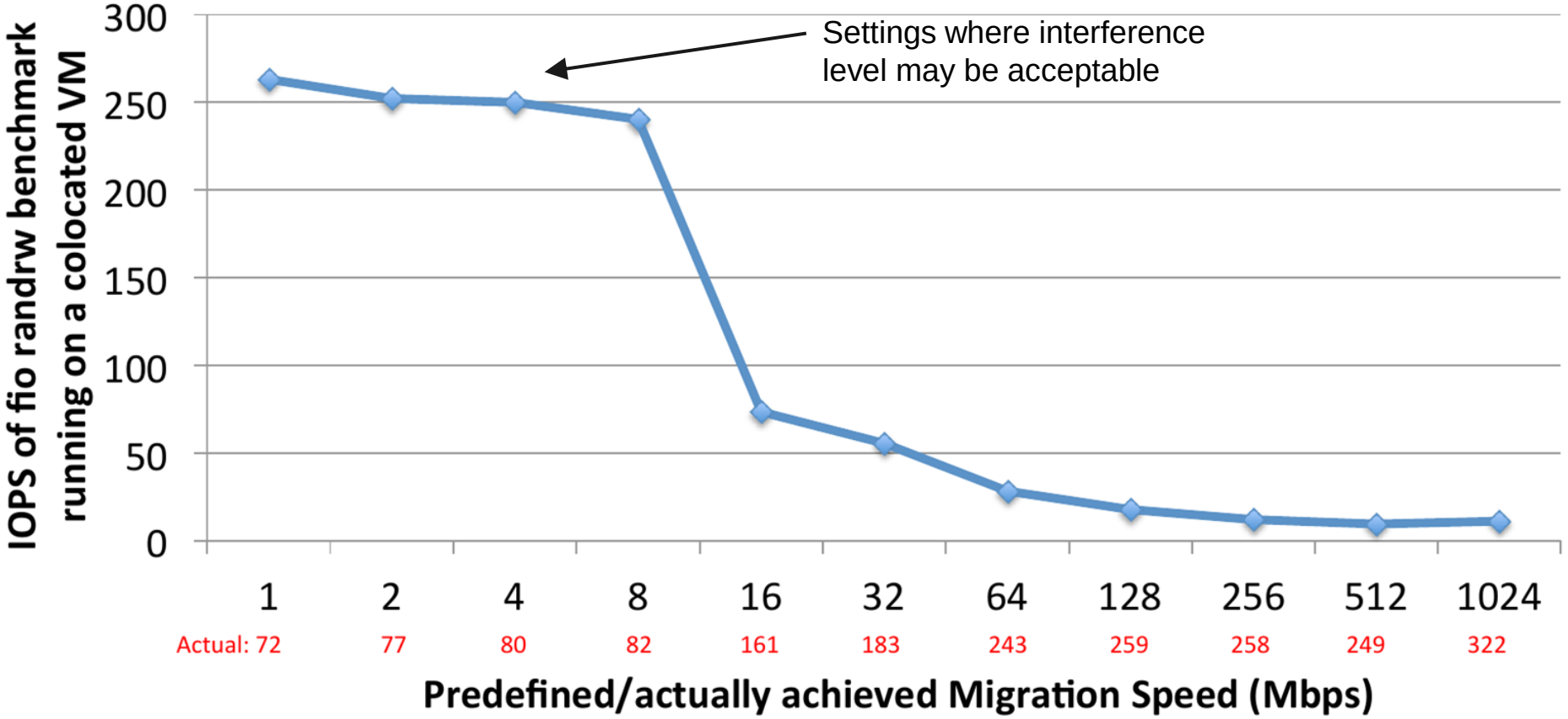
Storage Migration Offloading (SMO)

- **SMO Design goals**
  1. Maintain negligible interference throughout migration

  2. Don’t reduce the migration’s chance for convergence

  3. Avoid sacrificing the migration’s performance

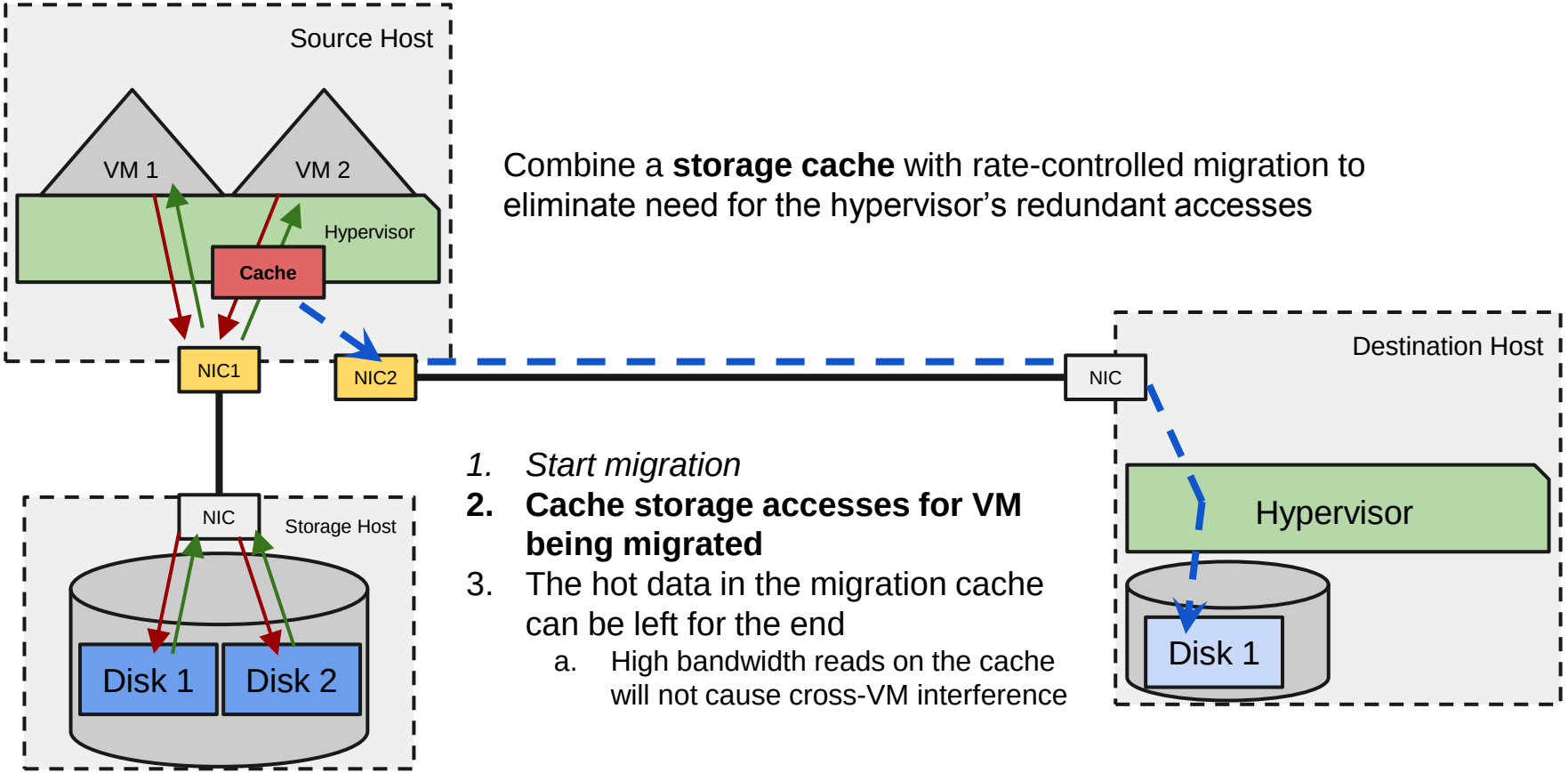
Migration Interference



SMO: Rate-Controlled Read

- **Rate Controlled** migration
  - Monitor perceived utilization/interference on disk
  - Adjust the migration read rate to avoid over-utilizing
    - Reduce interference
  - Problems still exist, just not as bad…
    - Improved latency vs. simple low static migration…
      - Could periods of low utilization be leveraged better?
    - Convergence and stop-and-copy could still suffer
      - Migration can still *fail*

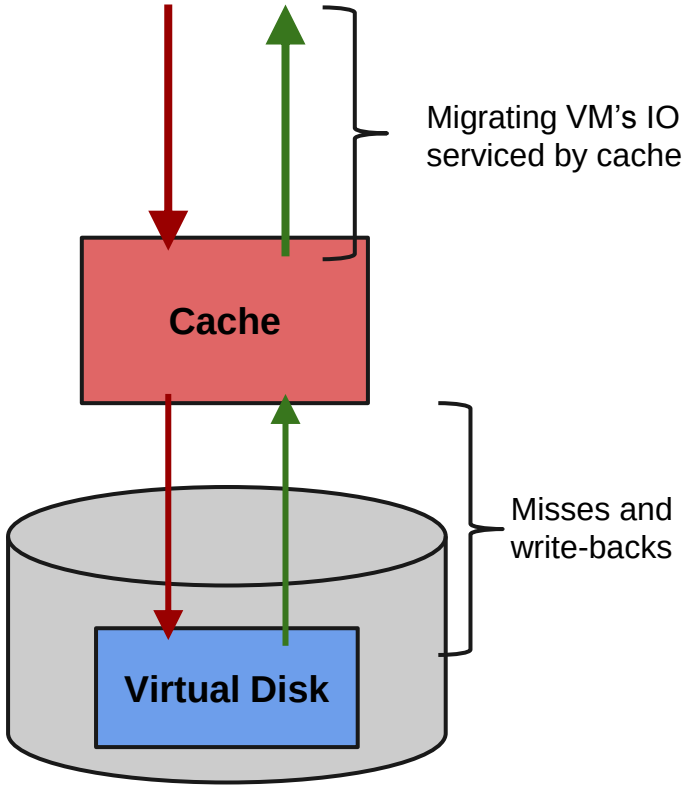
SMO: Caching



Combine a **storage cache** with rate-controlled migration to eliminate need for the hypervisor's redundant accesses

1. *Start migration*
2. **Cache storage accesses for VM being migrated**
3. The hot data in the migration cache can be left for the end
   a. High bandwidth reads on the cache will not cause cross-VM interference

SMO: Caching



For migration of IO-heavy VMs, this...

- Decreases shared storage utilization
  - Allowing increase in rate-controlled read
- Provides a low-interference store to get *dirtied* data
  - Data that makes it hard to converge

Can it be improved?

- Does not help if the migrating VM has low IO
- Workloads unlikely to access the ***entire*** disk
  - Some data will have to be read on behalf of the migration

SMO: Prefetch Data into the Cache

- Caching alone probably not enough
  - **Employ prefetching to *Buffer* data**
    - Get non-migrated data into the buffer whenever possible
  - **Prefetching should not cause extra interference**
    - Use excess disk bandwidth identified by rate controller
  - **Prefetched data can serve IO requests**

Disjoin sending data over the network
***from* reading data out of storage**

SMO: Transfer Rates

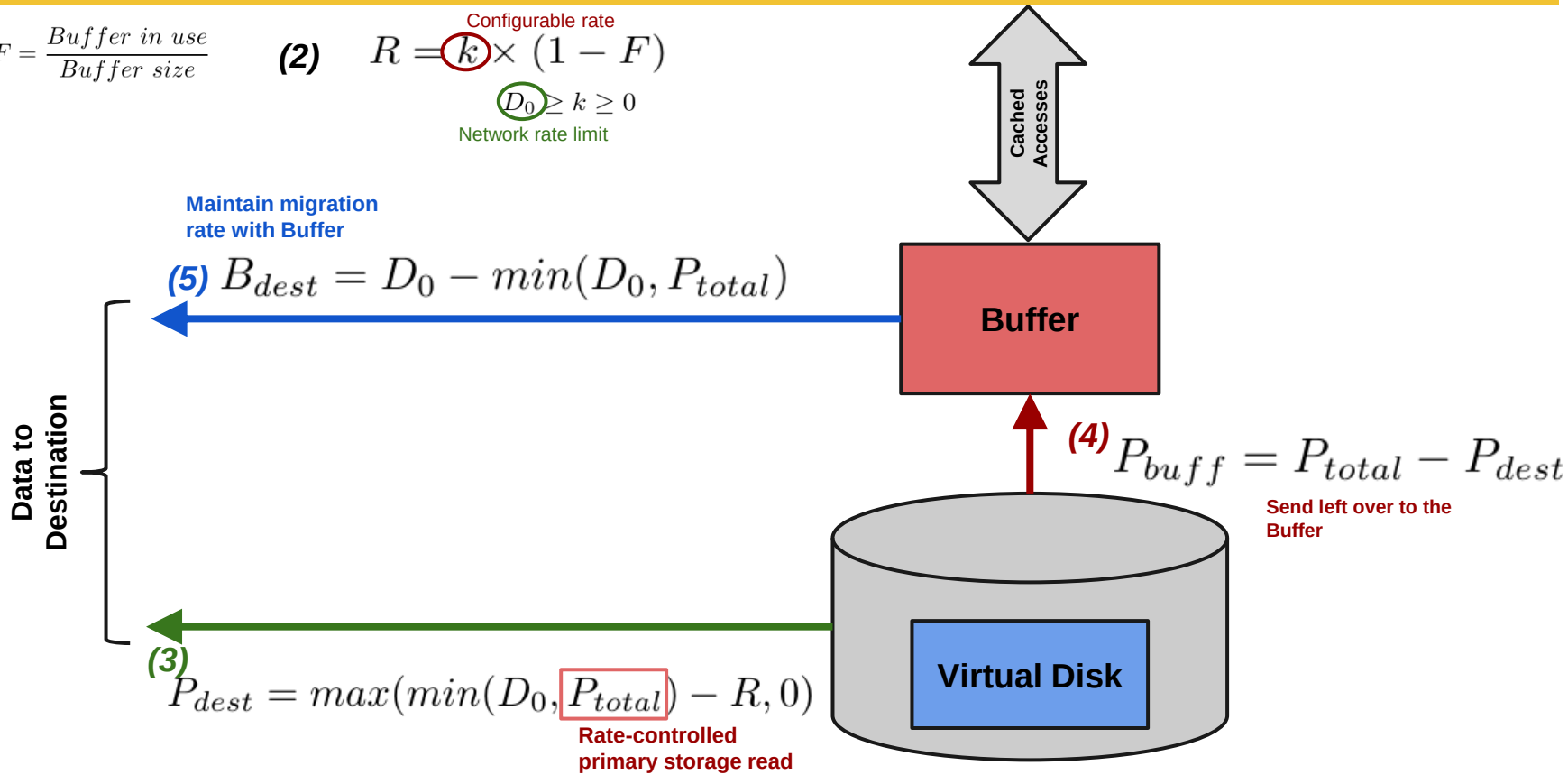
(1) $F = \frac{Buffer\ in\ use}{Buffer\ size}$

(2) $R = k \times (1 - F)$

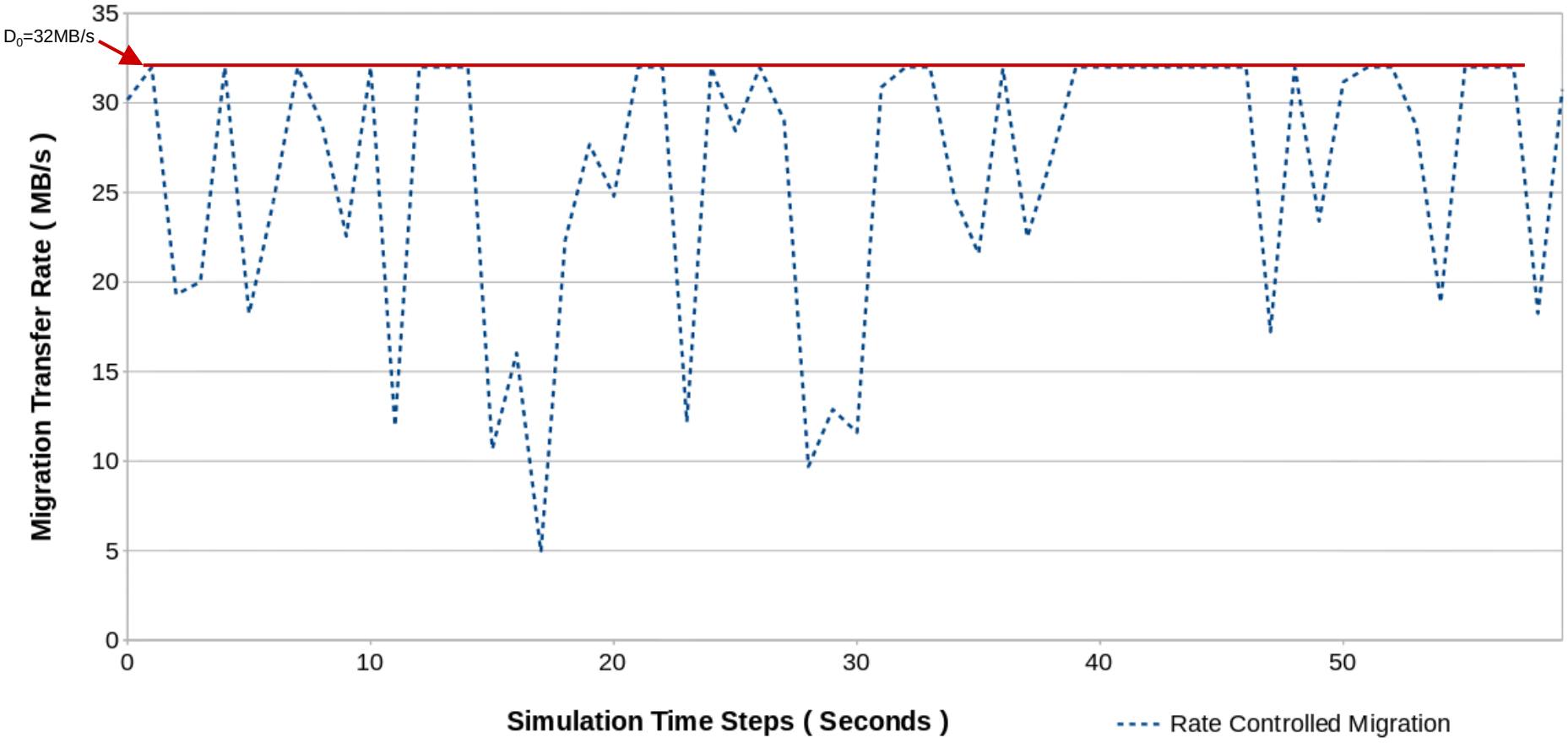
Configurable rate

$D_0 \geq k \geq 0$

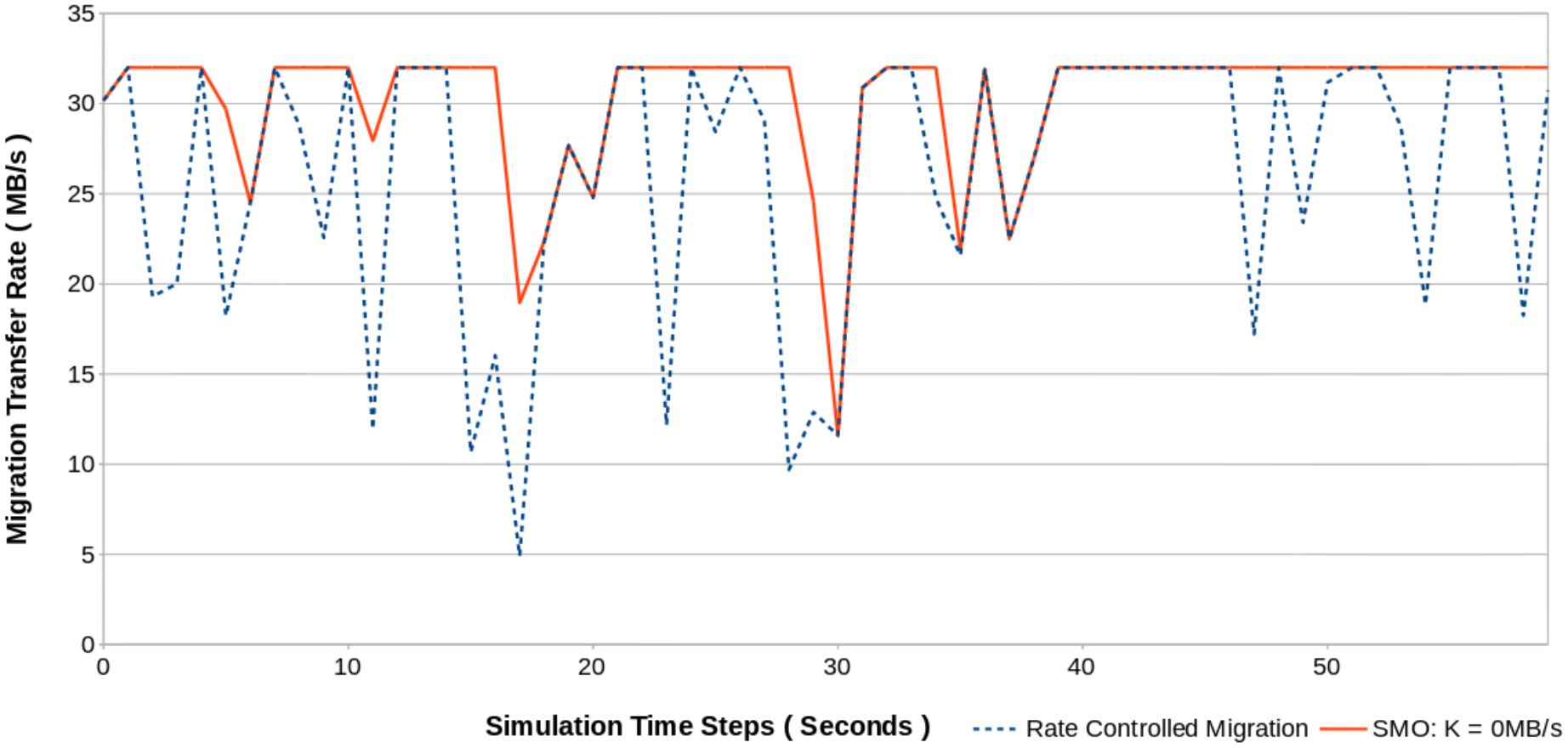
Network rate limit



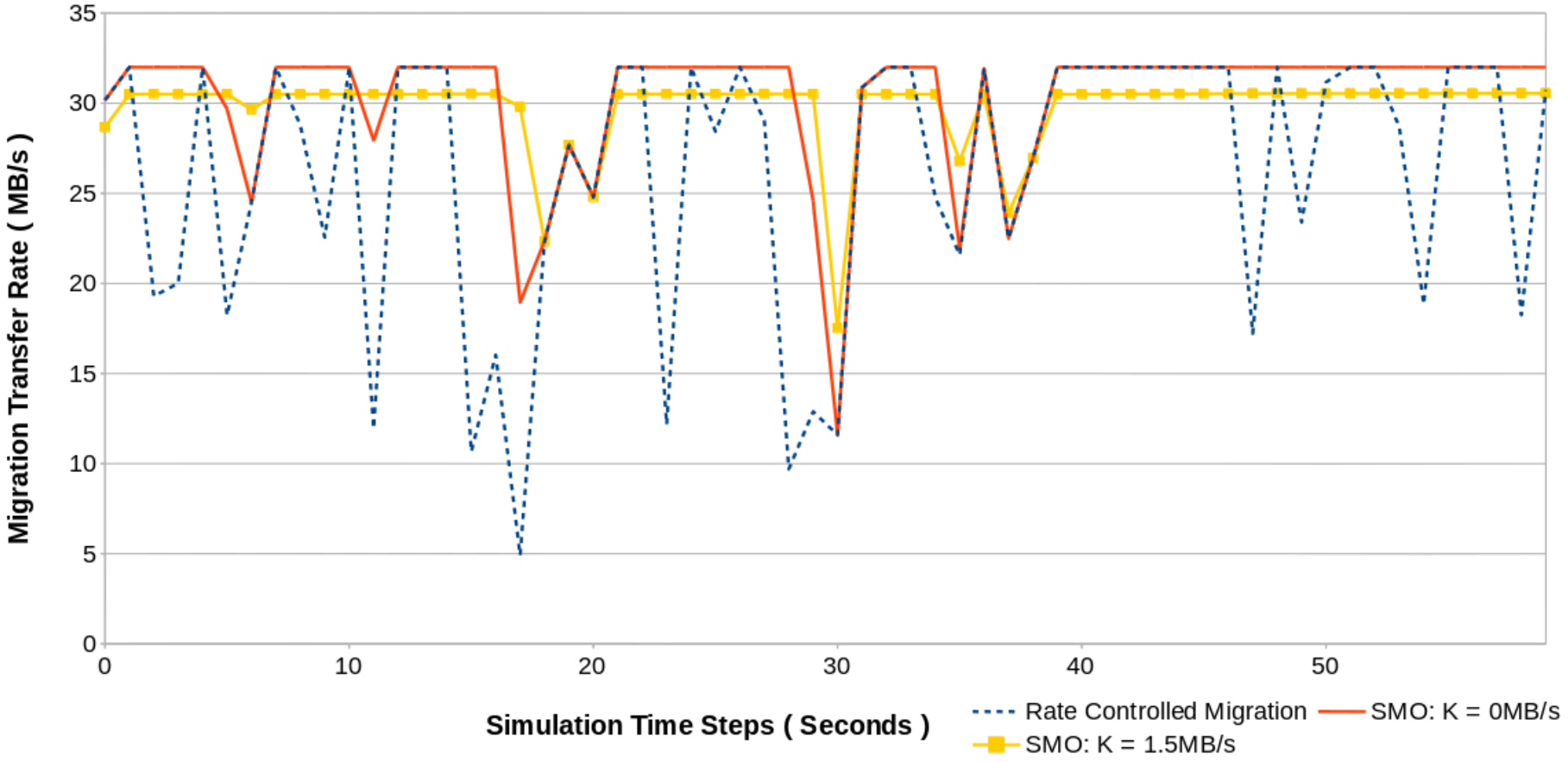
SMO: Analysis



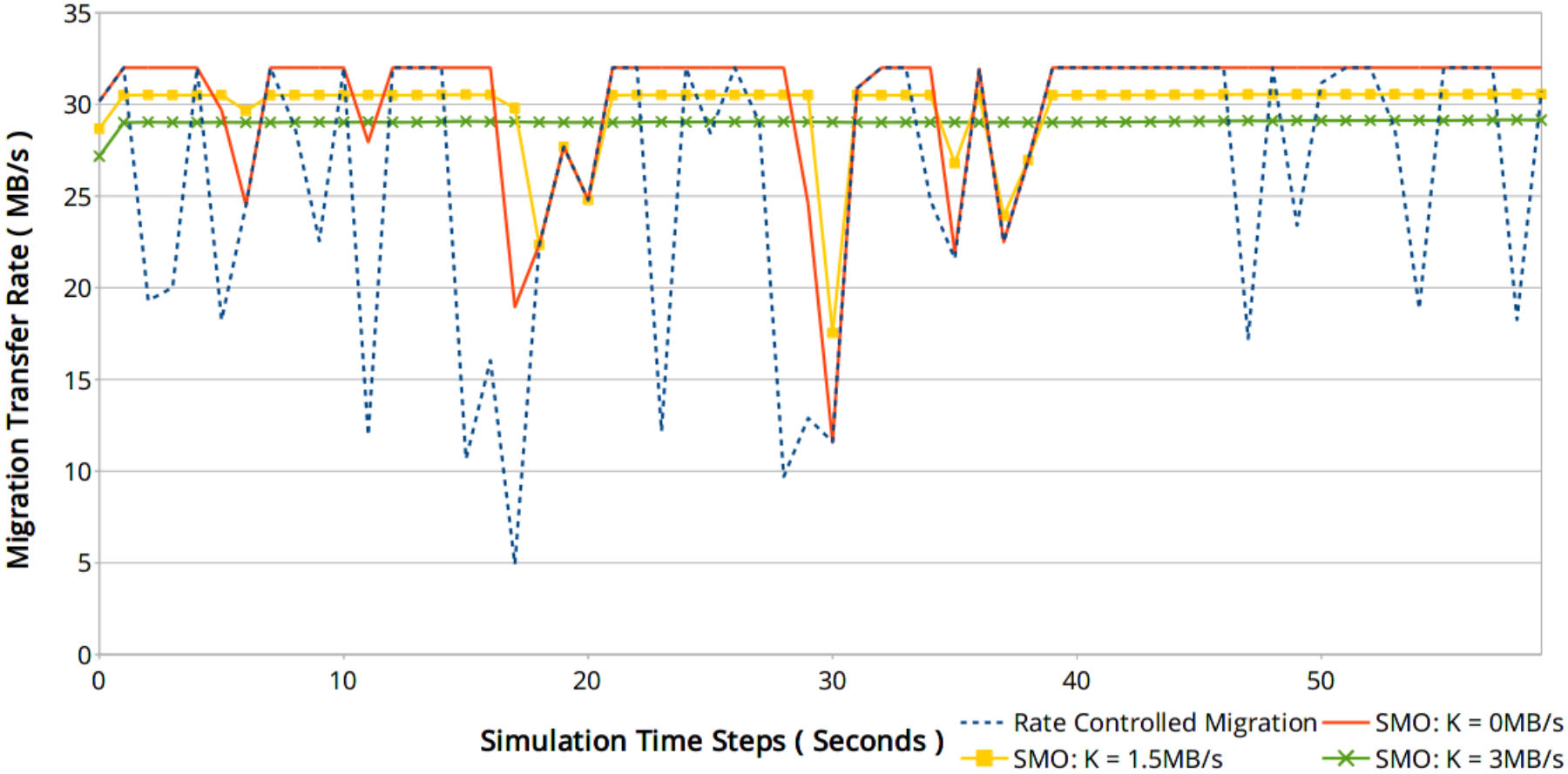
SMO: Analysis



SMO: Analysis



SMO: Analysis



SMO: Dynamic Caching Policy

- Basic assumptions
  - Non-volatile, Fully-associative, necessary meta-data to achieve consistency
  - Migrated data is usable, but considered *free*

- Create interplay with rate-controlled prefetching
  - Since cache policy dictates the primary IO levels

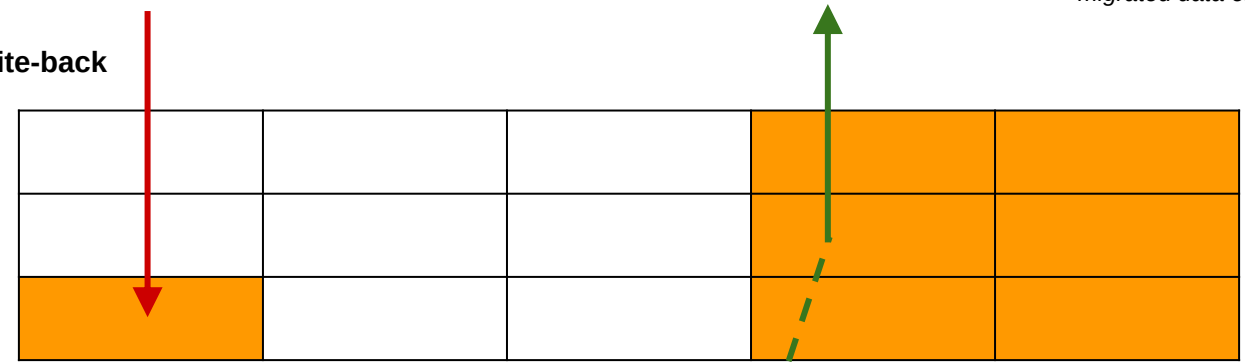
Storage Migration Offloading: Buffer States

- Two Properties (4 Combinations):
  - Space Available
    - **Under Capacity** *or* **At Capacity**
  - Status of Migration data
    - **Partially Offloaded**
      - Non-migrated data still on primary storage
    - **Fully Offloaded**
      - All remaining non-migrated data resides in buffer

Partially Offloaded & Under Capacity

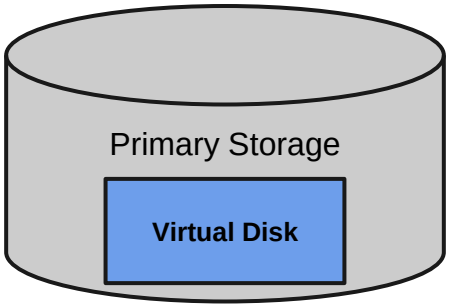
Buffer with space remaining, with non-migrated data on primary storage

Writes are issued as **write-back**



Buffer's cache lines

Misses are brought into the buffer



Goal: Drive primary utilization down while populating the buffer

Storage Migration Offloading: Buffer States

- **Partially Offloaded & Under Capacity**
  - *Drive primary utilization down while populating the buffer*
- **Partially Offloaded & At Capacity**
  - *Allow primary utilization to rise to normal levels, decrease buffer utilization*
- **Fully Offloaded & Under Capacity**
  - *Capture dirty data, decrease buffer utilization*
- **Fully Offloaded & At Capacity**
  - *Allow primary utilization to rise to normal levels, decrease buffer utilization*

Conclusion

- Storage migration interference impacts IO
  - Easily degrade basic IO over 90%
- Any full migration will require a full read of vDisk
  - Deduplication, compression, cold-data first, etc.
- Simple scheme: **Storage Migration Offloading**
  - Use secondary storage for buffering & caching
    - Offload data as quickly as possible
    - Leverage the workload’s IO through caching
    - Take advantage of extra disk bandwidth when possible

SMO: Future Work

- Subtleties of caching
  - State transitions, required tracking data, etc.
- Caching + Prefetching analysis
  - Benefit of interplay needs exploration
- Potential for migration *staging* phase?
  - Cache and offload prior to migration

Thank You!

Questions?

Acknowledgements

- Anonymous reviewers of PDSW 2014
- U.S. National Science Foundation (NSF), grants CCF-1102624 and CNS-1218960.

Backup

Buffer Device

- Considerations
  - Non-volatile keeps the preliminary design simple
    - Though, RAM or leveraging page cache is enticing
  - Size can be small (16GB - 32GB)
    - Assuming migration of ~60 GB disk
    - Stays cheap
  - SSD or High-performing HDD preferred
    - Should be able to maintain expected performance while prefetching+migrating

Cache Consistency

- Considerations
  - All data is expected to move to another store
    - Avoid writing through to storage, since the data is not required to be there
  - Must correlate buffer data to specific VM
    - Rebuilding in event of failure
    - Require several bits to uniquely identify the VM

Storage Migration Offloading: Caching

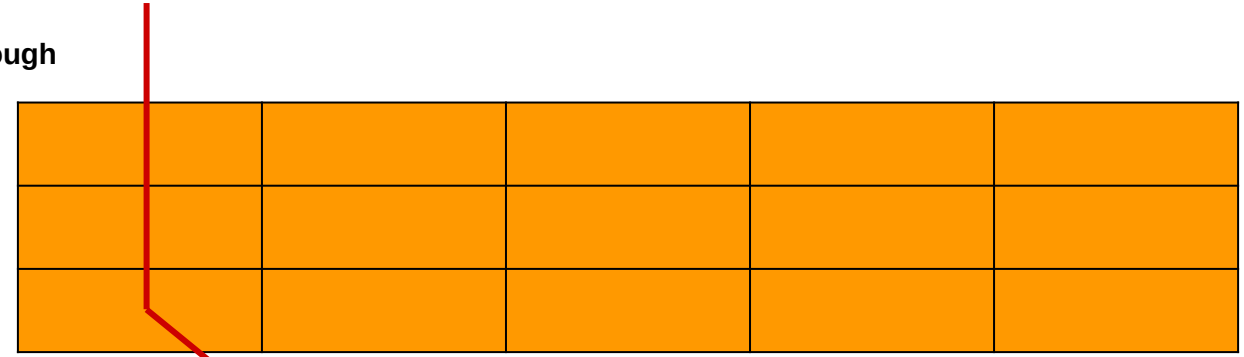
- Recognize redundant reads to storage
  - The running VM will likely read/write
  - The migration ***must*** read the entire vDisk
  - If the hypervisor reads **X** on behalf of the VM, the migration process should not have to read **X** again

Partially Offloaded & At Capacity

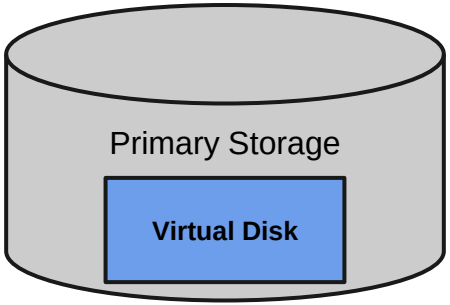
Buffer full, but non-migrated data on primary storage

Write hits are **write-through**

Write misses bypass the buffer



Reads bypass the buffer



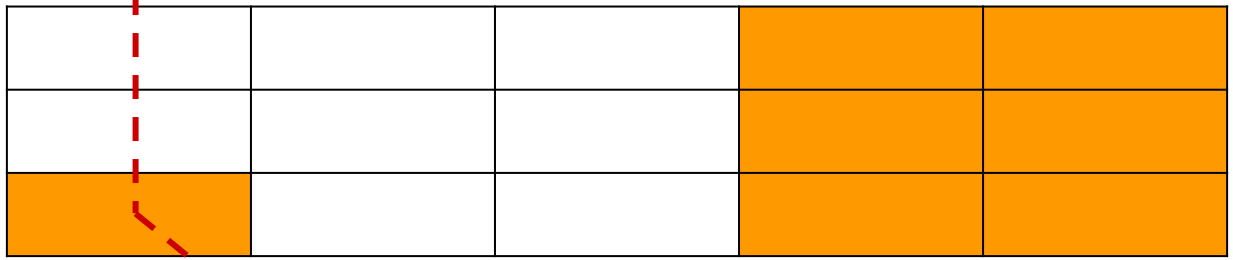
Goal: Allow primary utilization to rise to normal levels, decrease buffer utilization

Need to make space in the buffer → need to migrate data from the buffer

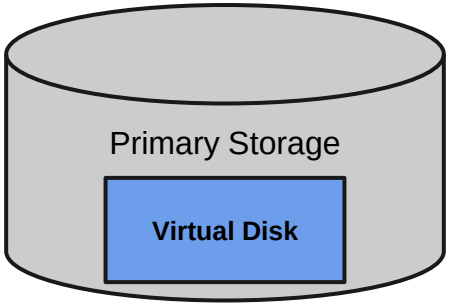
Fully Offloaded & Under Capacity

Buffer has space remaining, and all non-migrated data is on the buffer

Writes are **write-through**



Reads bypass the buffer



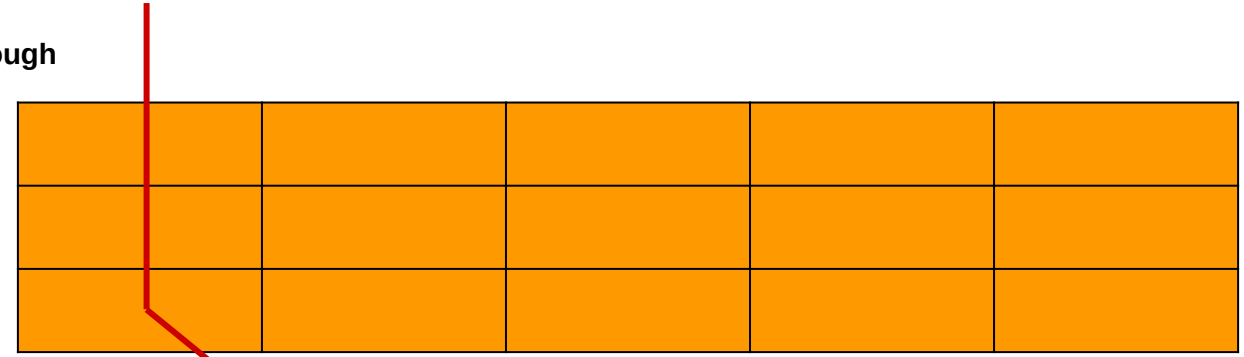
Goal: Capture dirty data, decrease buffer utilization

Fully Offloaded & At Capacity

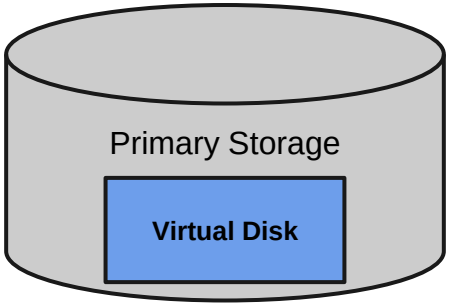
Buffer full, and all non-migrated data is on the buffer

Write hits are **write-through**

Write misses bypass the buffer



Reads bypass the buffer



Goal: Allow primary utilization to rise to normal levels, decrease buffer utilization

Need to make space in the buffer → need to migrate data from the buffer