Virtual Machine Storage – often overlooked optimizations

Dilip Naik
DilipN@Niriva.com
Niriva LLC

Overview of this talk

- Applies to all production VMs
  - Any hypervisor - Vmware, Hyper-V, Citrix etc
  - Any OS in guest – Windows, Linux, etc
- Applies to all storage –
  - DAS, thin provisioned storage arrays, etc
- Shows you how to eliminate at least 60% of disk I/Os during full backups; VM movements
  - Assumes 30% free space inside guest VM
  - Avoid reading all NTFS unused parts of child VM

"SPARSE STORAGE"

- Save space by not storing zeros
  - At the file system layer e.g. NTFS Sparse (SQL)
  - At the block layer e.g. storage arrays that support thin provisioning
- We kind of forgot about the I/O savings and that is what this talk is about
  - Don't store & don't copy/drag them around either!
    - *Yes you thin provisioned storage guys are missing something!*
  - Even if you store them, don't drag them around!

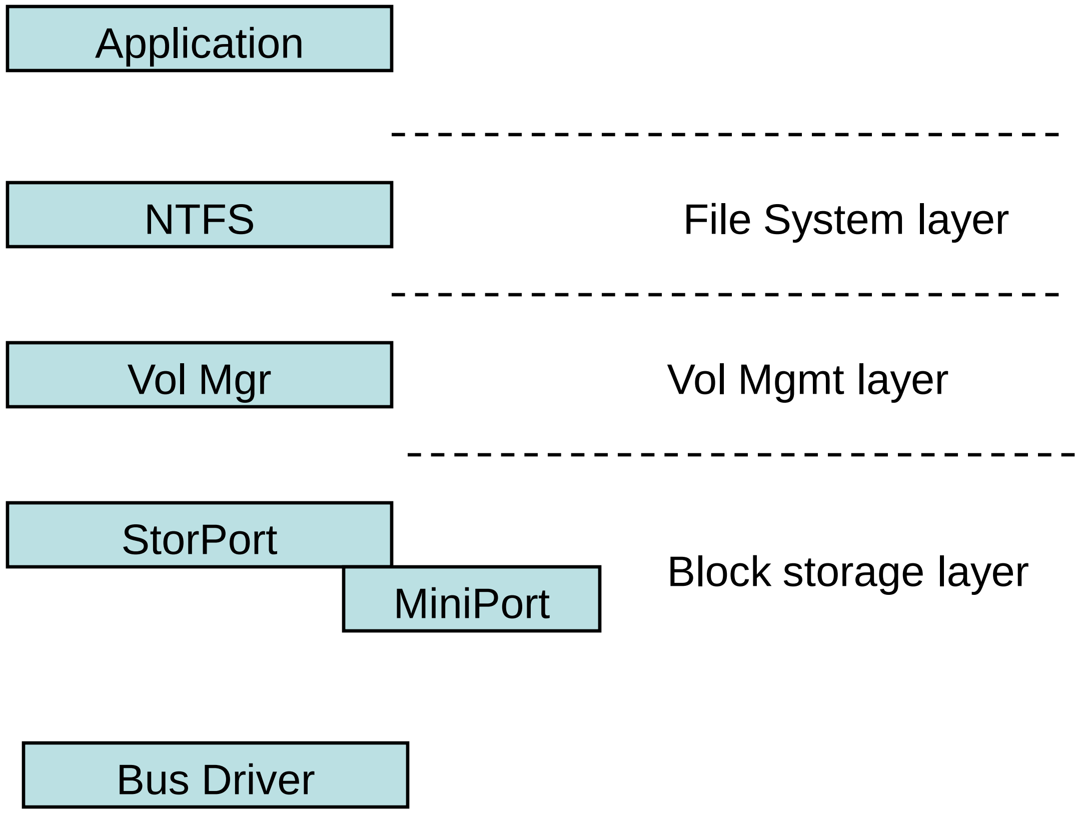
NTFS Sparse & block thin provisioned

	NTFS Sparse	Thin provisioned storage
Data	Zero data in units of 16kb not stored by either NTFS or by block storage	Zero data units in units decided by storage vendor not stored; NTFS believes it is storing data
Non sparse aware App issues read	NTFS does not issue reads to block storage, NTFS returns zero data to app; zeros do not cross PCI bus	NTFS issues reads to block storage, block storage returns zeros to NTFS; zeros MAY cross PCI bus; zero filled buffers dragged around
Apps can be aware of storage provisioning	Yes, NTFS publishes APIs; but Hyper-V does not use those APIs; SQL does	No, block storage vendors do not publish APIs; even if they do, NTFS does not use them
Poorly written copy or backup app will thick provision the file	Yes, e.g. xcopy or copy.cmd copying an NTFS sparse file	No, thanks to storage vendors who unprovision blocks containing zeros; app still drags the zero filled buffers around •and the non zero useless data

SSDs & Sparse Storage

- SSDs want to turn unused blocks into stored zeros
- Thin provisioned arrays want to turn unused blocks into non stored zeros
- Issue : How to detect NTFS unused blocks?
  - Win7 – if SSD & ATA & NTFS & default registry values then issue TRIM command
  - Relevant committees still finalizing for SCSI devices (UNMAP command)

Windows storage stack overview



At what layer does your thin provisioned storage array return zero filled buffers? Can you do better? What if the zero filled buffers never cross the PCI bus?

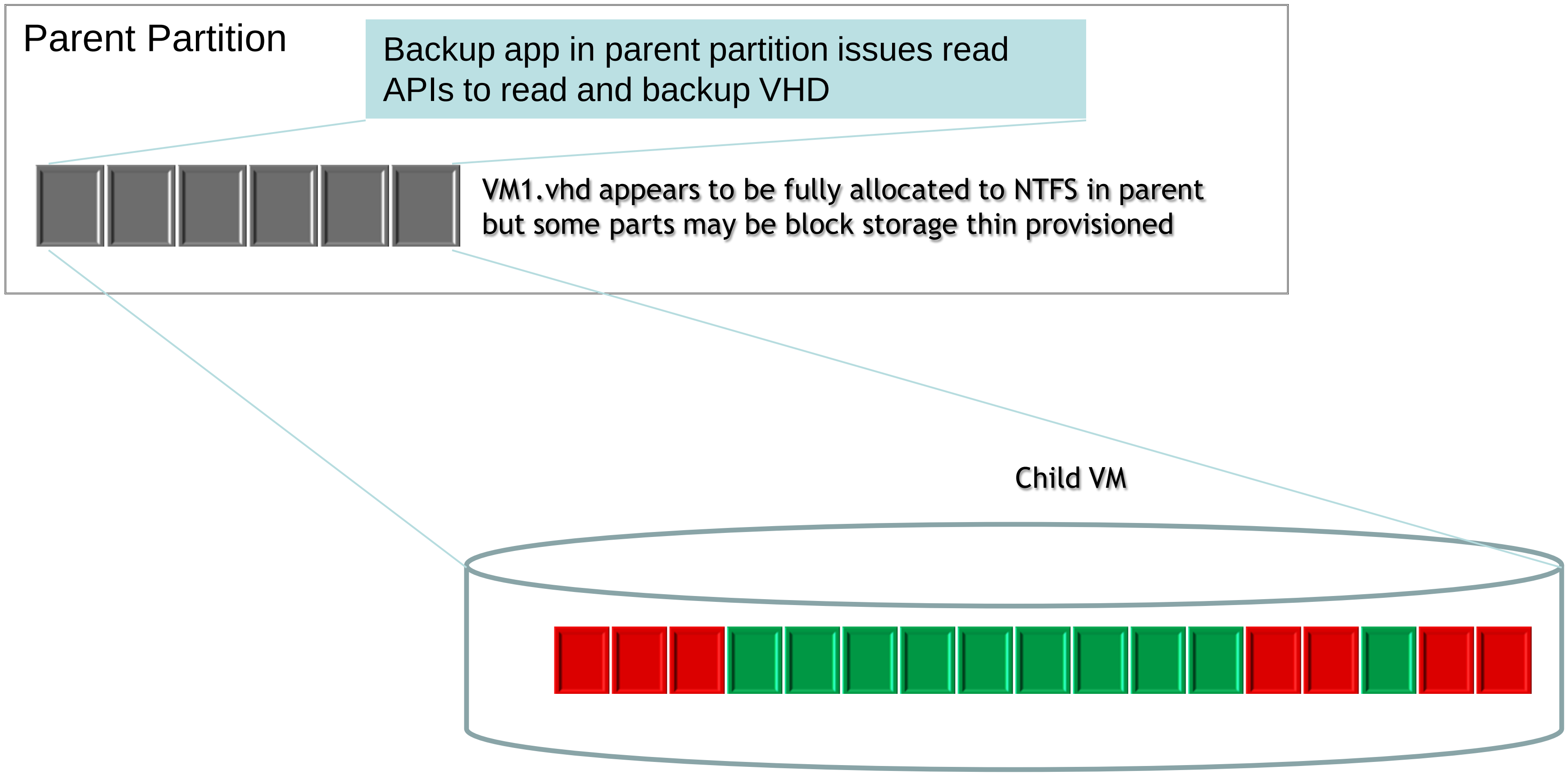
VM file formats

- All major vendors (VMware, Microsoft, Citrix) support two types of VM file formats
  1. Fixed size aka flat – a 40 GB VM occupies 40.0X GBs – all space pre-allocated
     - Best practice for production VMs
  2. Dynamically expanding aka sparse – file grows as needed

Typical VM usage scenarios

- Full backup of a VM
- Copy a non running VM
  - From one test server to another
  - From test to production server
  - From local cloud to remote cloud
- ***All of these are typically performed in a highly inefficient manner!***

Figure 1 – VHD file from parent & child perspectives



Parent Partition
Backup app in parent partition issues read APIs to read and backup VHD
VM1.vhd appears to be fully allocated to NTFS in parent but some parts may be block storage thin provisioned
Child VM

Assume green block thin provisioned
Run a full backup in parent partition
- What happens when backup app reads the green parts of the file?
- Who generates the zeros, what happens in cache mgr and NTFS buffers?
- And if green blocks are NOT thin provisioned, things are MUCH worset
- NTFS will scribble so need a way to maximize the green blocks BEFORE you run backup

🟩	Free space
🟥	Block in use

Demo 1 – full backup operation

Demo 2 – VHDCopy operation

VHDCopy, VHDBackup, VMDKCopy

- Avoid reading useless parts of VHDs/vmdks
- Scriptable – Take a VSS snap, mount the snapped vol, and VHD inside is source for copy or backup
- No need to zero fill source – dest is zero filled & compresses much better – especially so with transactional s/w like SQL, Exchange
  - Applies particularly to NTFS scribble
- With /Secure option, dest VHD is zero filled, but zeros don't traverse LAN or PCI bus!
- Built as highly portable C library

Summary

- Traditional full backups need help
  - Run Sdelete or some soln to zero fill VHD
  - Even then, a full backup "reads" the zero blocks
  - Zero blocks cause disk & CPU overhead
- VHDCopy, VHDBack, VMDKCopy
  - Do the equivalent of zero fill VHD
  - Do not read the "zero filled" blocks
  - Considerable savings in disk I/O & CPU
  - Backup files smaller – more compressed zeros
  - Written as a library of APIs portable to any OS

NTFS “scribble”

1. Create new file, write 4MB, close…delete
   - Turns some thin provisioned block storage into thick
2. Repeat …the new file occupies diff blocks
   - Turns more thin provisioned blocks into thick
3. Transactional s/w such as SQL, Exchange etc make the op described in above bullets realistic
4. Storage vendors have worked hard
   1. Provide an applet to run periodically
   2. Hook NTFS operations
   3. Trick NTFS into sending Trim command ? Love

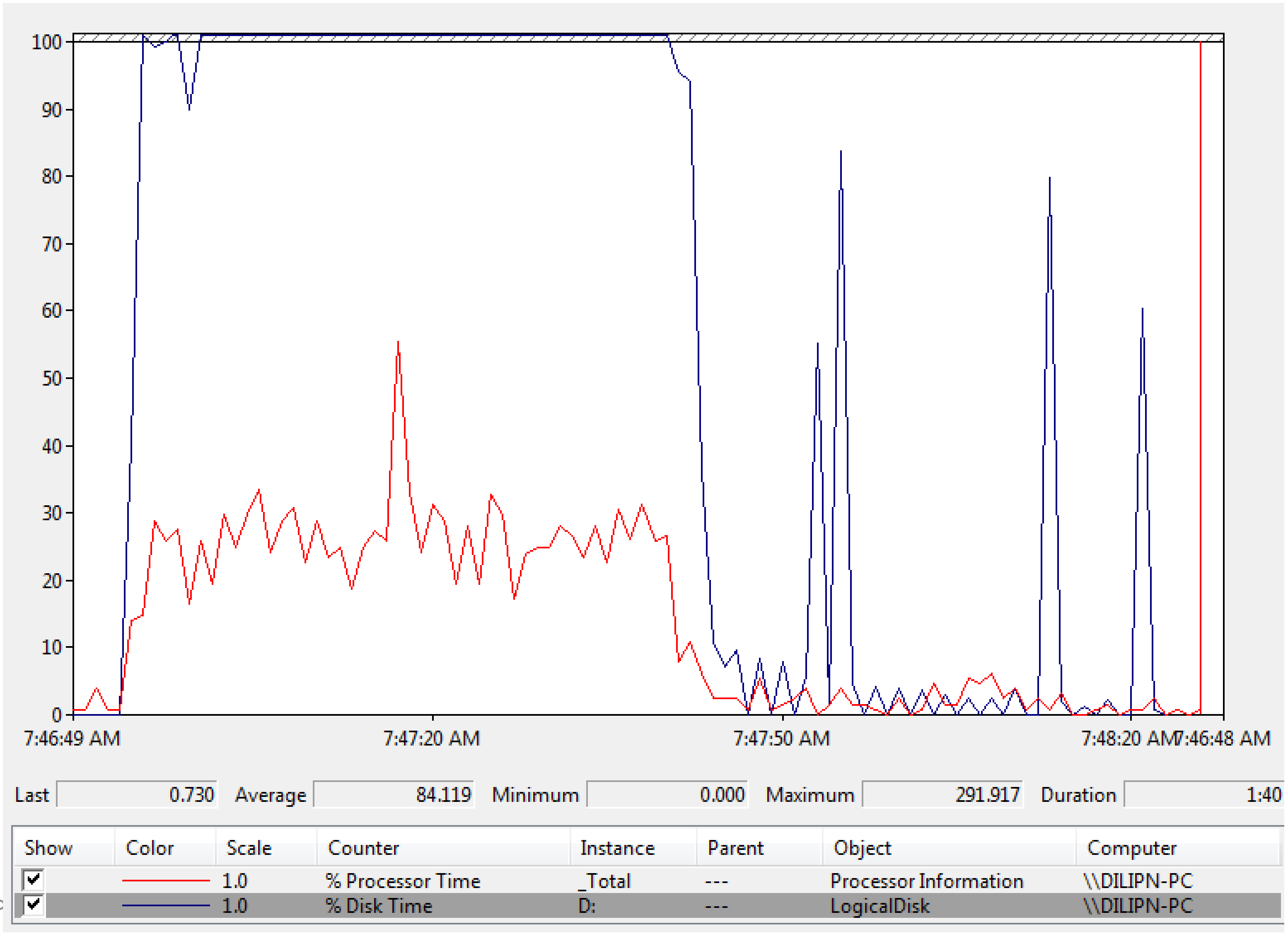
References

- Microsoft VHD file format specification
  - http://technet.microsoft.com/en-us/virtualserver/bb676673.aspx
- VMDK file format specification
- ATA/Trim delete notification support in Win 7 by Neal Christiansen at SNIA SDC 2009

Backup Slides

CPU & Disk I/O with xcopy (cmd)

Disk I/O(blue) and CPU (red) graph while copying a 4GB mostly empty VHD file with xcopy cmd. Note disk I/O hitting a sustained 100 %

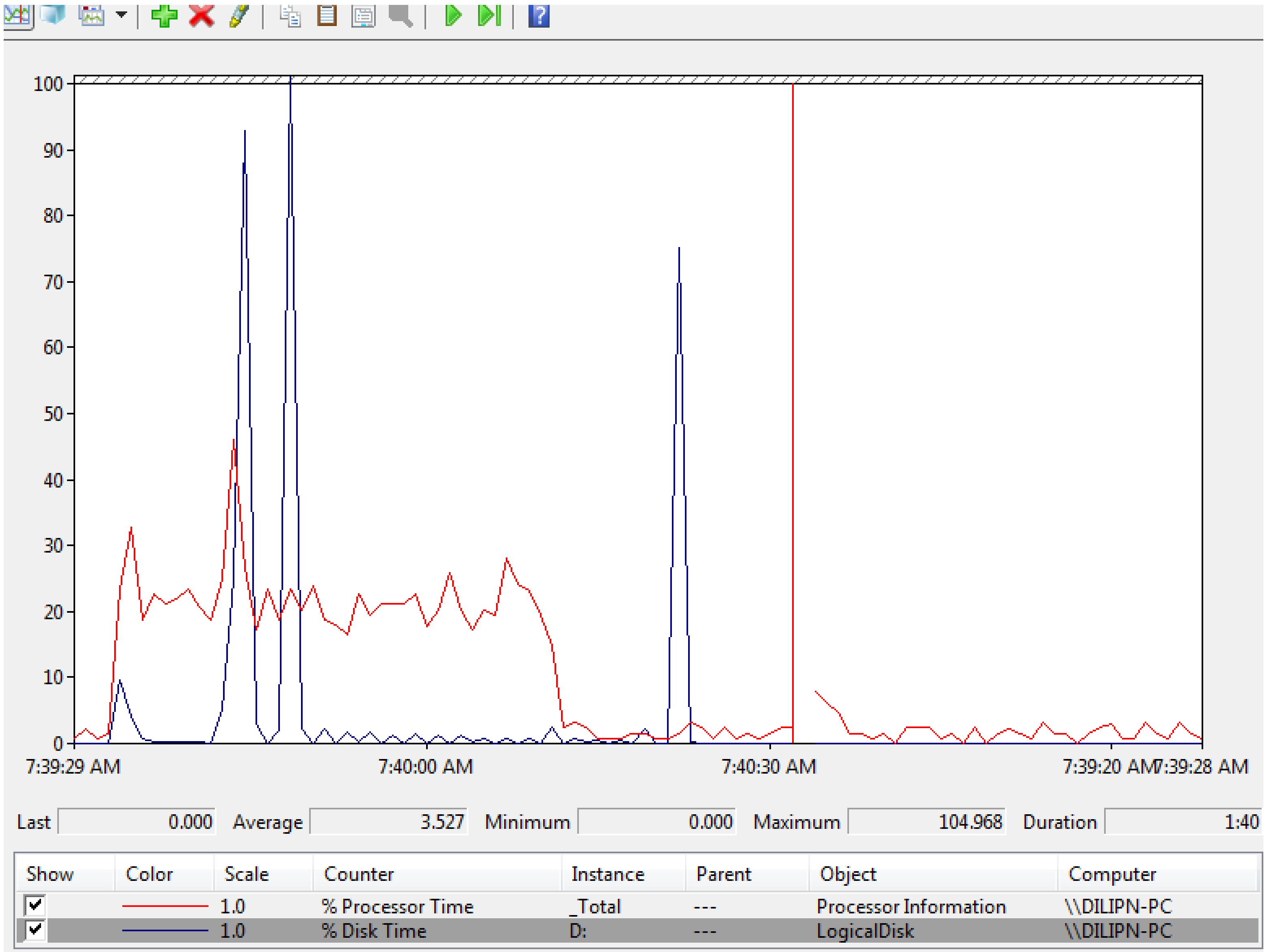


Last 0.730 Average 84.119 Minimum 0.000 Maximum 291.917 Duration 1:40

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	1.0	% Processor Time	_Total	---	Processor Information	\\DILIPN-PC
✓	🟦	1.0	% Disk Time	D:	---	LogicalDisk	\\DILIPN-PC

CPU & Disk I/O with VHDCopy

Disk I/O(blue) and CPU (red) graph while copying a 4GB mostly empty VHD file with VHDCopy. Note reduction in both disk I/O AND CPU

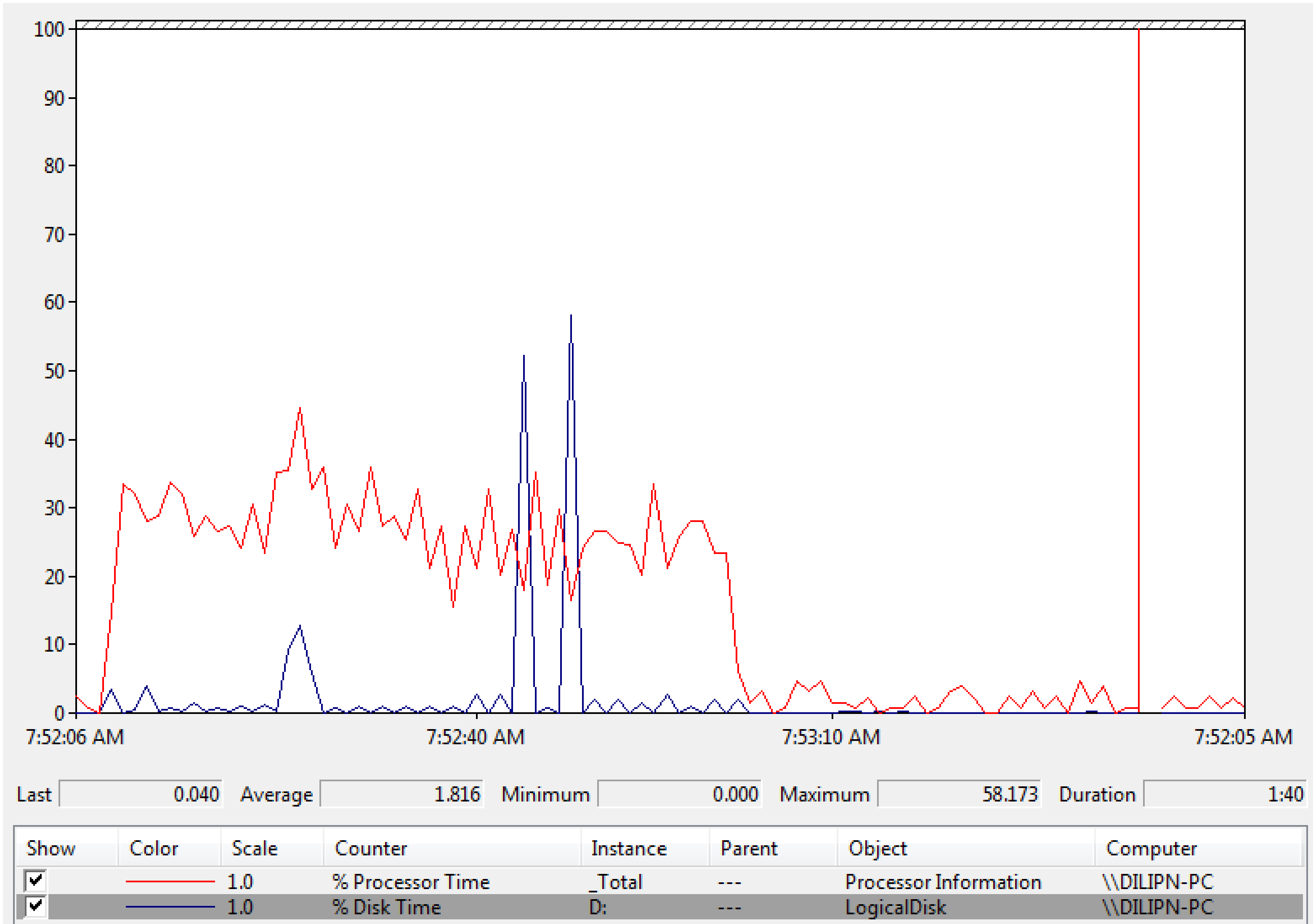


Last	Average	Minimum	Maximum	Duration
0.000	3.527	0.000	104.968	1:40

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	1.0	% Processor Time	_Total	---	Processor Information	\\DILIPN-PC
✓	🟦	1.0	% Disk Time	D:	---	LogicalDisk	\\DILIPN-PC

CPU & disk I/O graph with VHDBack (LZ compressed)

- Note the even further reduction in disk I/O and CPU I/O
- With production VMs, especially so running transactional software, the savings will be much bigger (think SQL, NTFS)
- Note the CPU usage (with LZ compression) versus pure xcopy(no LZ) in slide 12



Last	Average	Minimum	Maximum	Duration
0.040	1.816	0.000	58.173	1:40

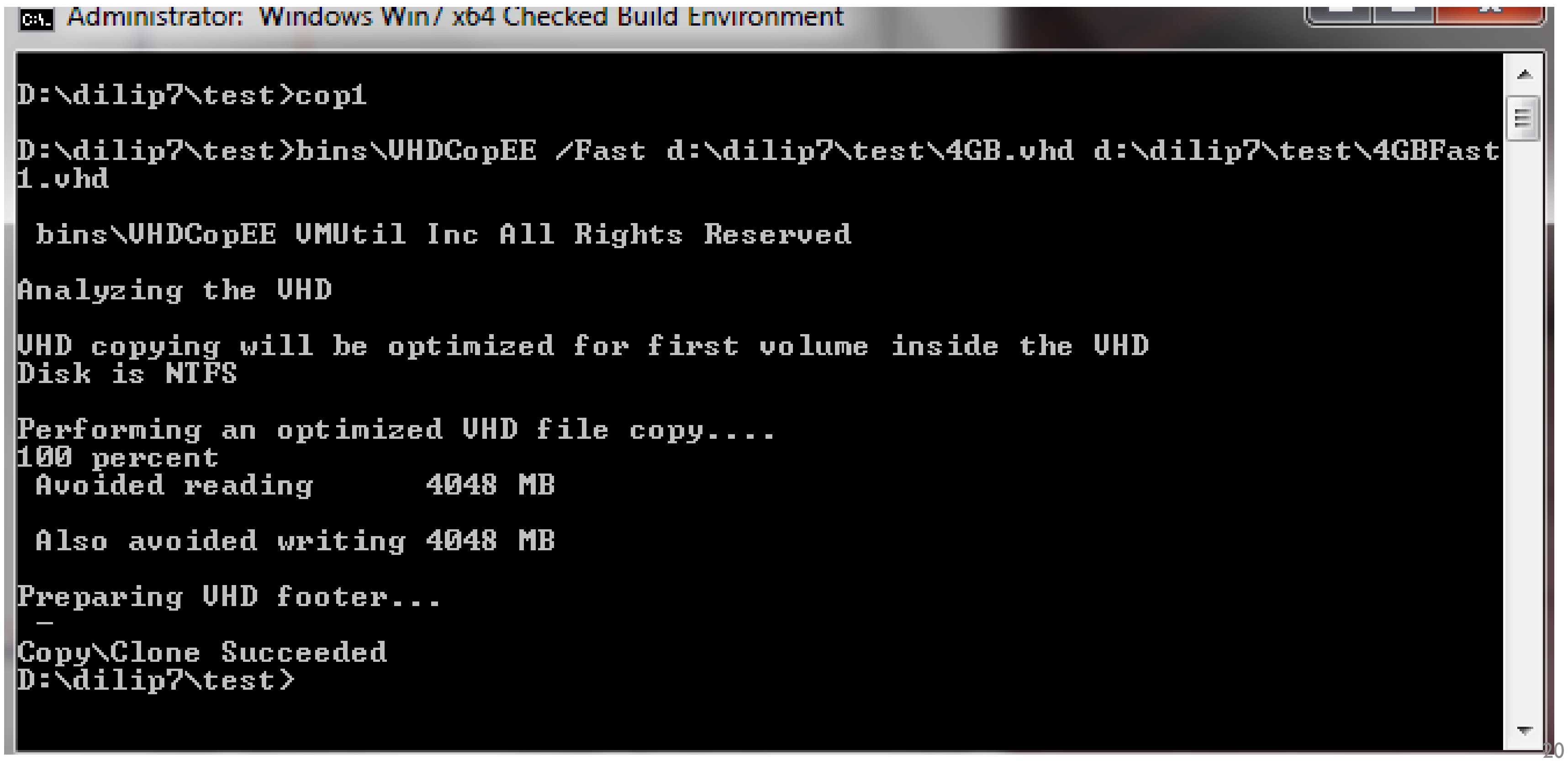
Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	1.0	% Processor Time	_Total	---	Processor Information	\\DILIPN-PC
✓	🟦	1.0	% Disk Time	D:	---	LogicalDisk	\\DILIPN-PC

VHDCopy command line tool

•Note that a mostly empty 4096 MB (4GB) resulted in a 48MB read

•These savings WILL scale even with production VMs

•Please repeat this experiment with your thin provisioned storage!



Administrator: Windows Win7 x64 Checked Build Environment

```
D:\dilip7\test>cop1

D:\dilip7\test>bins\VHDCopEE /Fast d:\dilip7\test\4GB.vhd d:\dilip7\test\4GBFast
1.vhd

 bins\VHDCopEE VMUtil Inc All Rights Reserved

Analyzing the VHD

VHD copying will be optimized for first volume inside the VHD
Disk is NTFS

Performing an optimized VHD file copy....
100 percent
 Avoided reading      4048 MB

 Also avoided writing 4048 MB

Preparing VHD footer...
 -
Copy\Clone Succeeded
D:\dilip7\test>
```

VM backup

- Backup from outside VM
  - Will drag zeros
    - Backup App issues read to parent NTFS
    - Parent NTFS issues read to SAN storage which returns zeros (before or after PCI bus)
    - Zeros traverse PCI bus? Ask your storage vendor
  - But VHDBack, VHDCopy will not drag zeros!

- Backup from inside VM
  - Costly – pay for agents
  - Enumerate files and backup files
    - Slower; will avoid dragging zero buffers; no bare metal
  - Block level backup
    - Will drag zeros
    - Complicated
    - File level recovery?