Performance of optimized software implementation of iSCSI

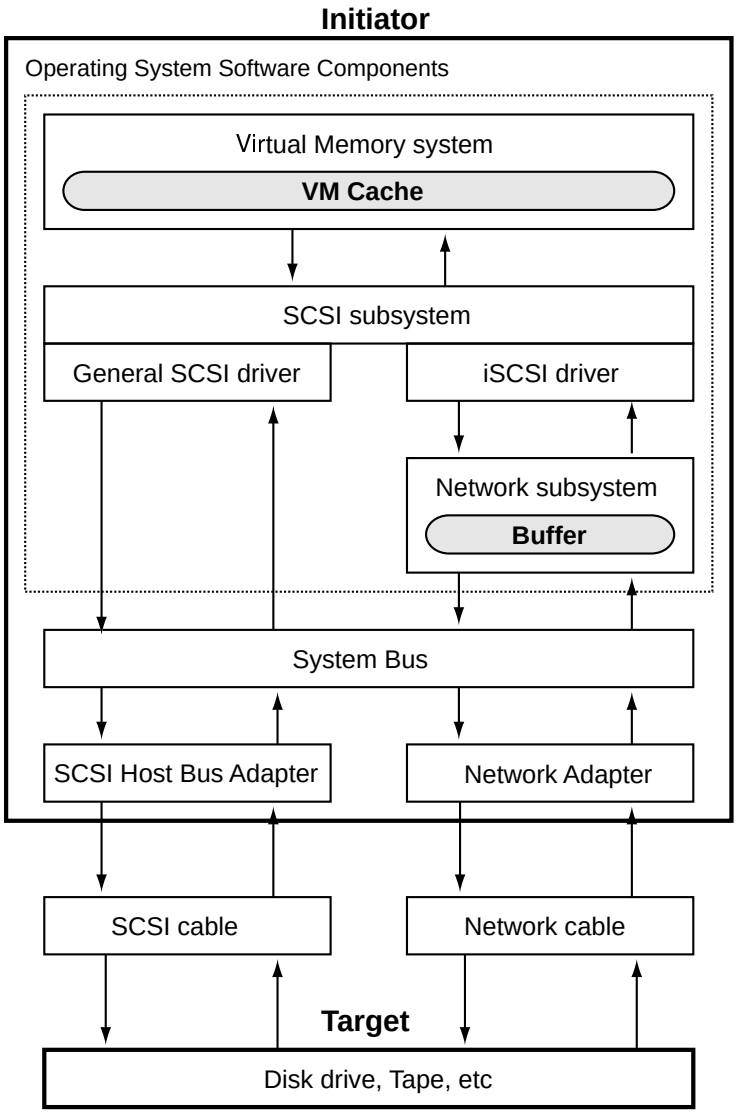
Fujita Tomonori

NTT Cyber Solutions Laboratories

tomof@acm.org

SNAPI'03, New Orleans, LA, USA

What is iSCSI



SCSI over TCP/IP

- SCSI commands are encapsulated into IP packets
- The host and storage systems are connected with Ethernet

New networked storage technology

- Today the dominant networked storage technology is Fibre Channel

Pros

- IP networks is inexpensive

Cons

- It suffers large overheads of processing TCP/IP protocol and data copying
  - ▷ *Specialized NICs for iSCSI*

Questions

Can iSCSI give high performance without specialized NIC?

- Are general network adapter features effective?
  - *Checksum offloading*
  - *Jumbo frames*
  - *TCP segmentation offloading (TSO)*
- Is the avoidance of data copy practical?

Where are the bottlenecks?

- Data copying and checksumming
- SCSI subsystem
- Network driver

General NIC features

Checksum offloading

- NICs calculate checksums instead of the host CPU
- Most gigabit NICs provide this feature

Jumbo frames

- It allows the host to use the larger MTU size (typically up to 9000 bytes)
- Most gigabit NICs provide this feature

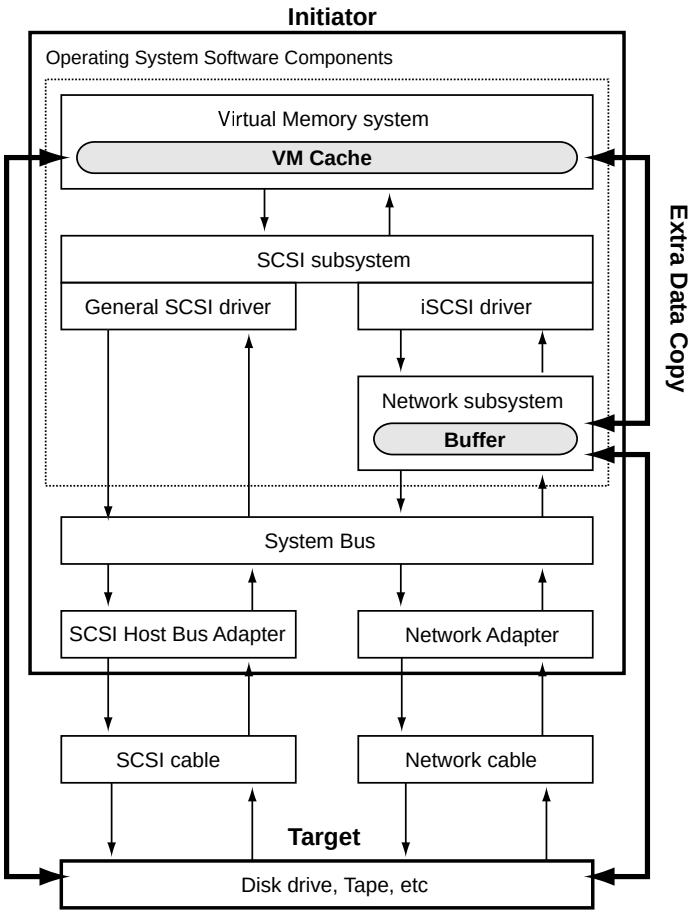
TCP segmentation offloading (TSO)

- NICs help to break a packet into smaller pieces at the IP layer upon writing
  - *The linux kernel can bypass the ip_fragment function*
  - *It works with large data (over the MTU size)*
- It has not become popular as yet
  - *3Com 3CR990, recent Intel NICs, and Broadcom Tigon3 chip (used by 3Com 3C996 etc).*

The Copying Problem

iSCSI copies data needlessly

- User/Kernel: between pages belonging to the user space and the page cache
- Kernel/Kernel: between the page cache and network buffers



We concentrate on avoidance of copy inside the kernel

- Interfaces of user/kernel copy avoidance are standardized
  - *Direct I/O : user processes bypass the page cache*
  - *Mmap : user processes directly access the page cache*

The status of open source iSCSI drivers for Linux

- All four drivers copy inside the kernel
  - *IBM, Cisco, Intel, and University of New Hampshire implementations*

Linux implementation issue (Write)

Zero-copy optimization requires no changes to the kernel

- Sendpage interfaces enable the NIC to transfer data directly from pages to the wire

Process of Writing

- The SCSI subsystem receives references to dirty pages
- The iSCSI driver hands over the references to the network stack
- The NIC transmits the contents of the referred pages by using DMA

iSCSI digest doesn't work with zero-copy optimization

- The iSCSI protocol defines 32-bit CRC digests on an iSCSI packet
- What if a process modifies a page between digest computing and transfer of the iscsi packest?
- The problem is that the Linux kernel permits modifying a page being written to a disk

Linux implementation issue (Read)

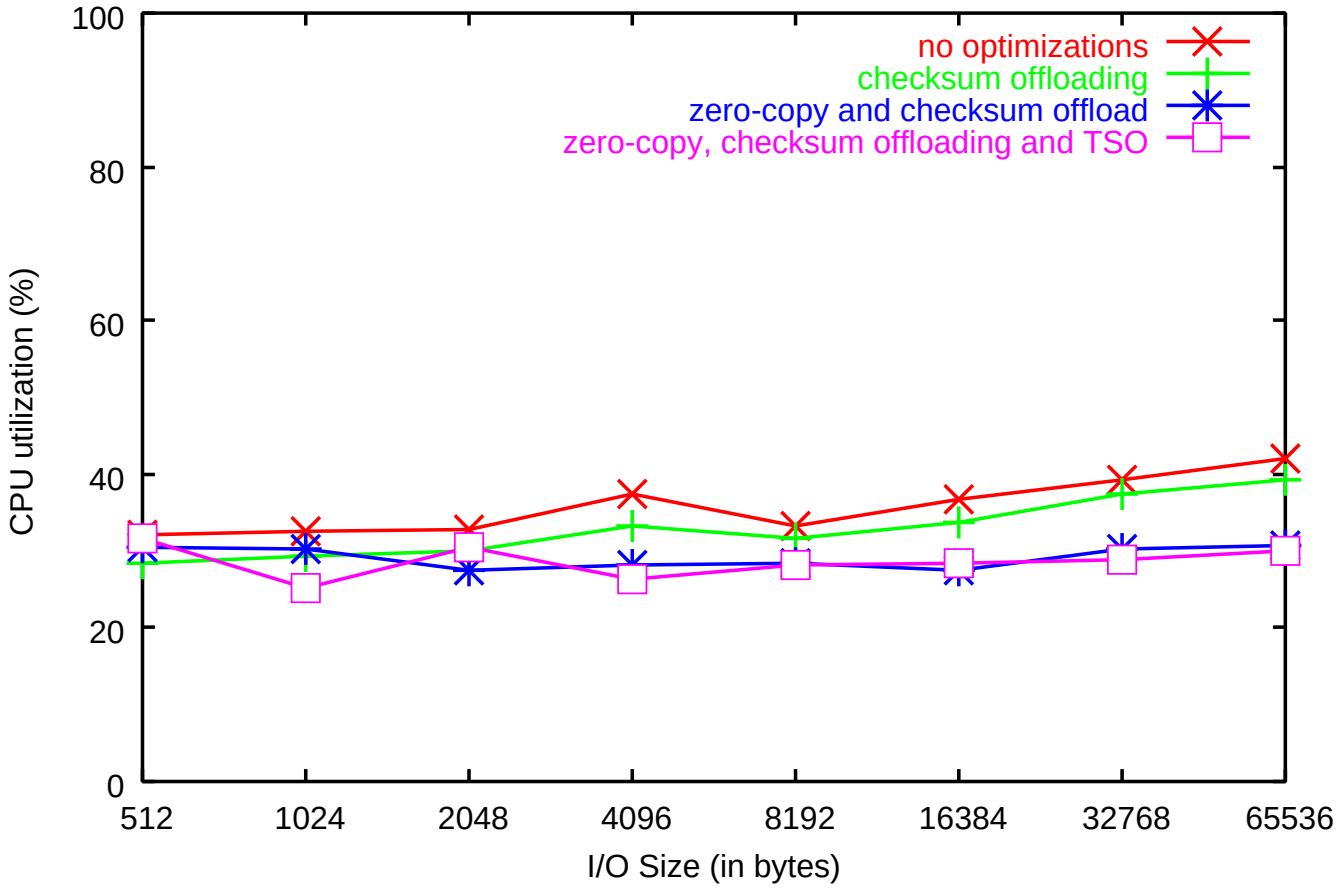
Zero-copy optimization requires changes to the kernel

- Page remapping changes address bindings between the page cache and network buffers

There are restrictions

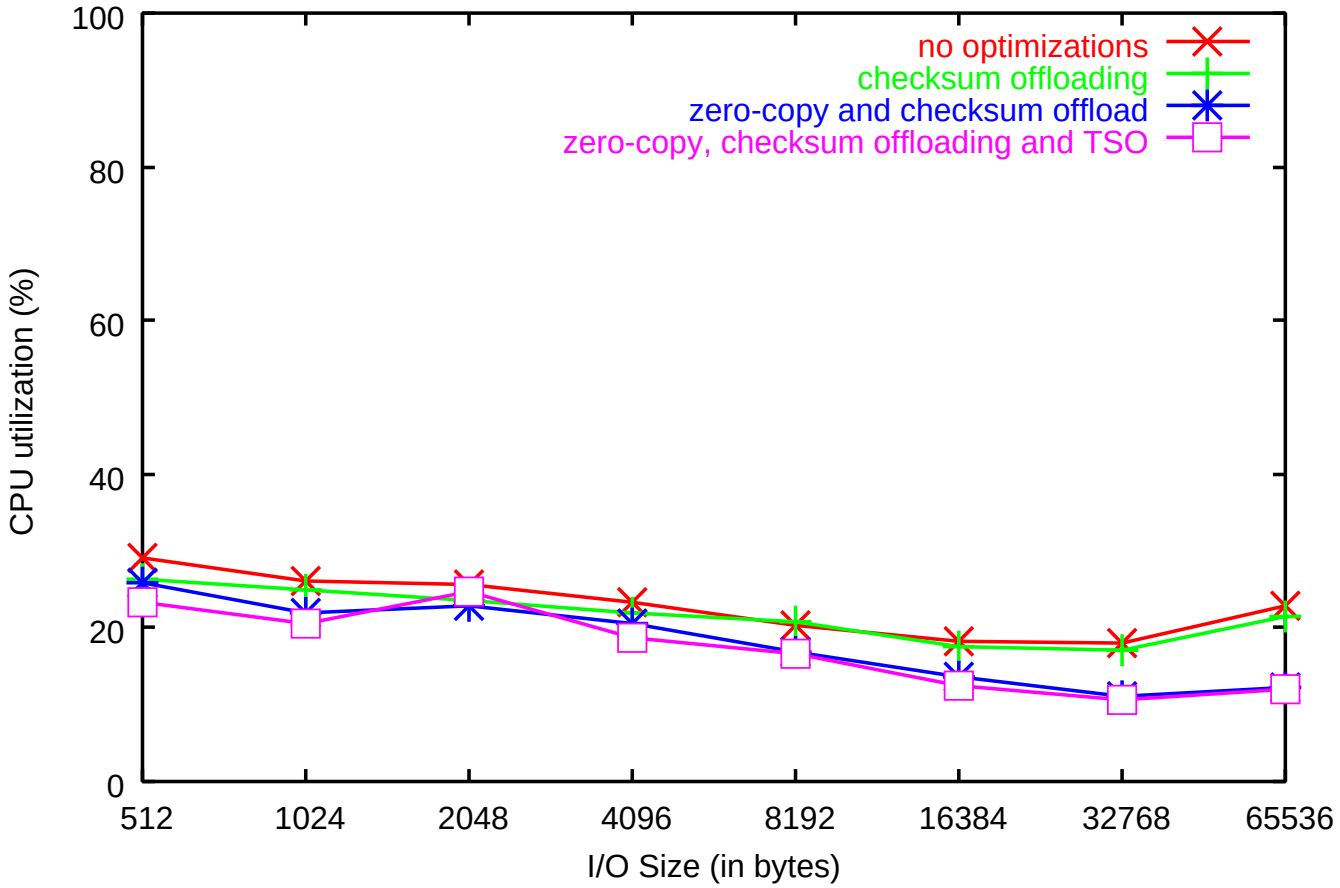
- The MTU size must be larger than the page size
  - ▷ *Jumbo frames are mandatory*
- It is not transparent to applications
  - ▷ *Buffers in the user space must be page-aligned with Direct IO*
- It suffers large overheads
  - ▷ *High-cost MMU operations are necessary*
- It can only be applied to the first TCP packet if one iSCSI packet consists of several TCP packets
  - ▷ *After removing the Ethernet, IP, TCP, and iSCSI headers, the data must be aligned on page boundaries*
  - ▷ *The length of the header of the first packet is different from that of the rest*

CPU utilization (write 1500-byte MTU)



- No clear gain from TSO
- Zerocopy produces better performance for the large I/O sizes.

CPU utilization (write 9000-byte MTU)



- Results are similar to those with the 1500-byte MTU

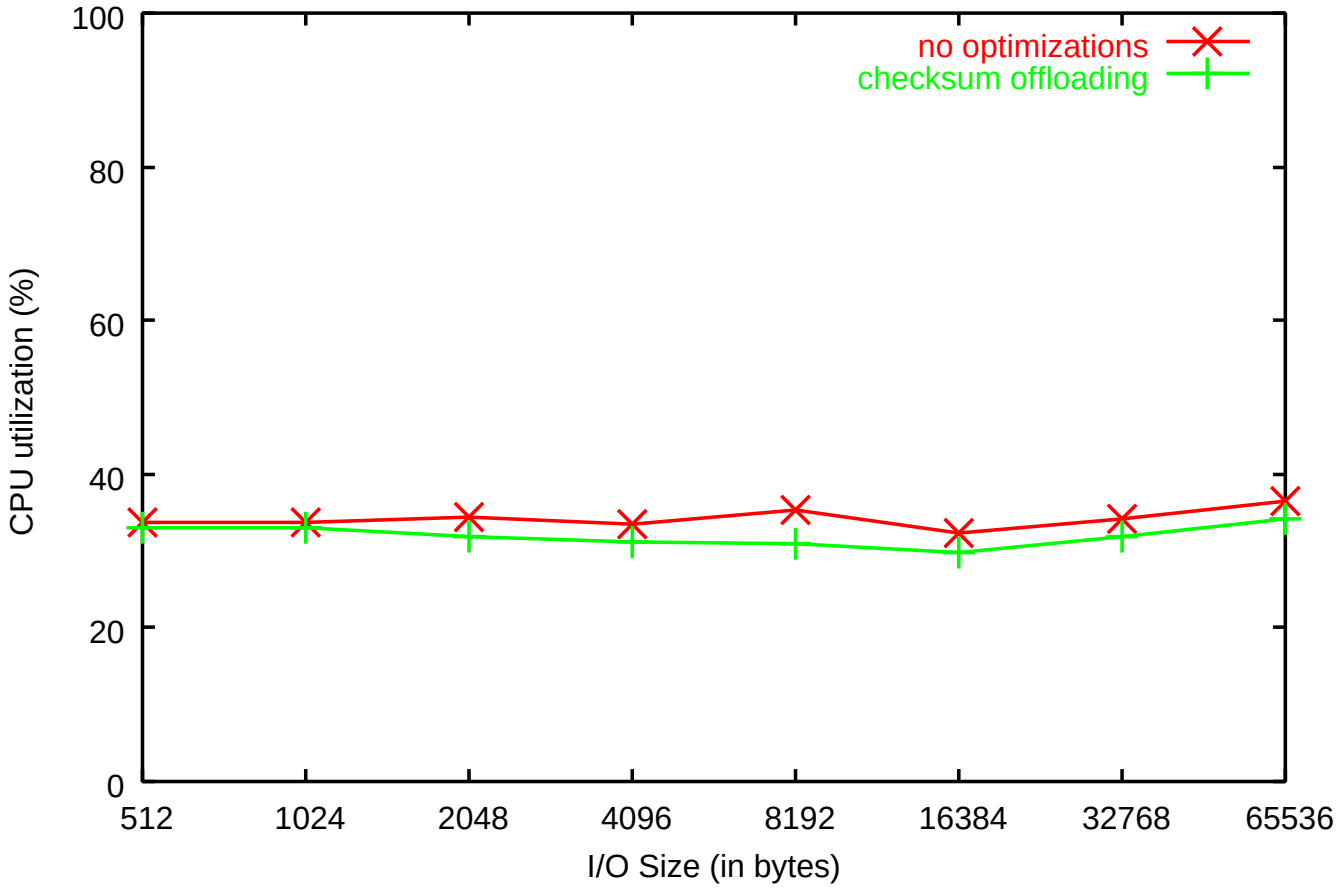
Distribution of CPU time (Write)

Distribution of CPU time during the write microbenchmarks for the 8192-byte I/O size.

	Percent Time					
	no optimizations		checksum offload		zero-copy	
MTU (byte)	1500	9000	1500	9000	1500	9000
Idle	66.81	79.85	68.35	79.23	71.86	83.32
iSCSI driver	1.52	1.34	1.53	1.41	1.32	1.30
SCSI subsystem	1.90	1.25	1.86	1.42	1.17	1.23
Copy & Checksum	3.76	3.40	2.93	2.41	0.27	0.27
NIC driver	2.12	0.96	2.27	1.07	2.60	1.06
TCP/IP	12.16	3.72	11.75	4.39	11.48	3.82
VM system	4.21	2.98	4.11	3.16	3.67	2.71

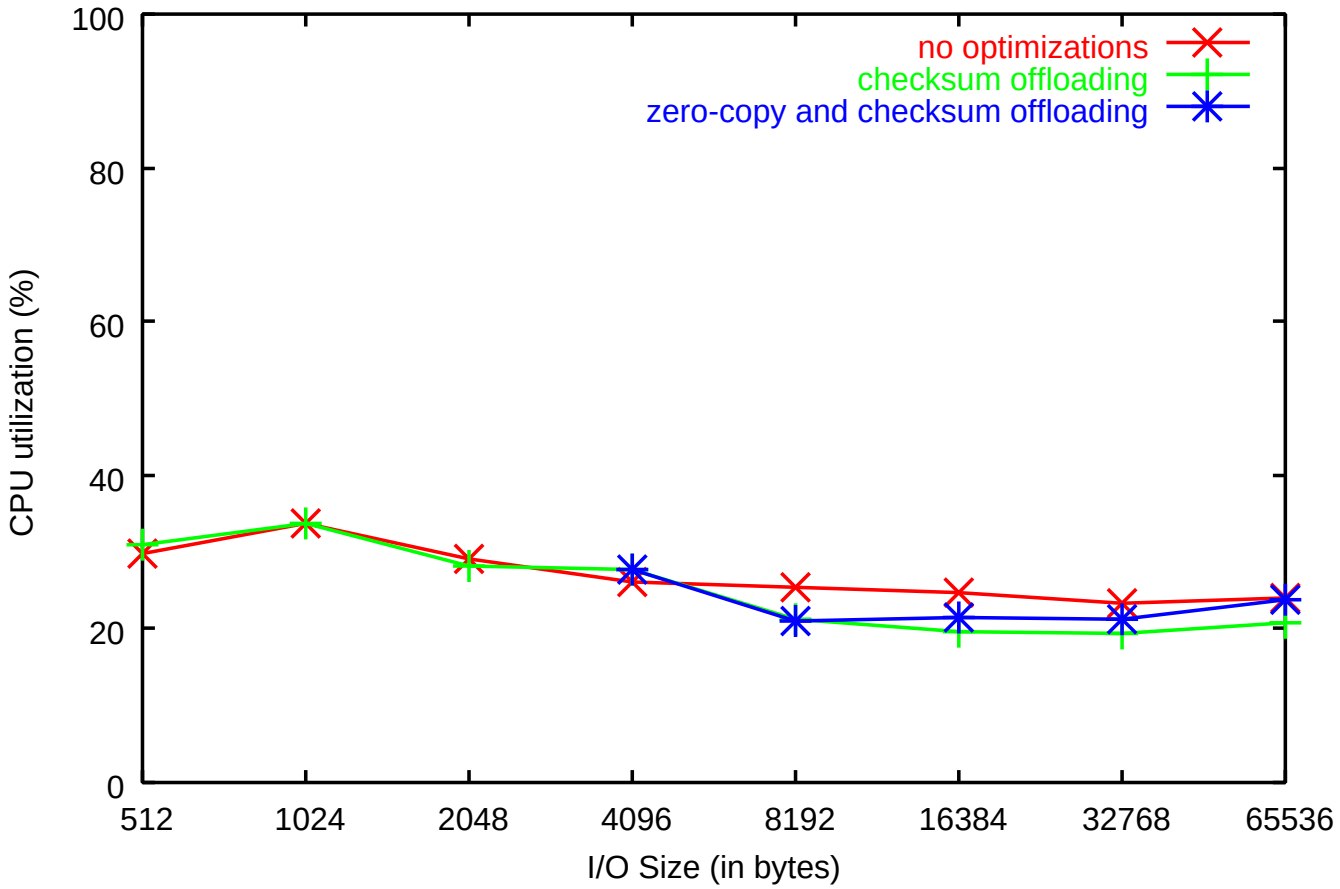
- The combination of checksum offloading and zero-copy is effective
- TCP protocol processing needs a lot of time
- SCSI and iSCSI processing is insignificant

CPU utilization (read 1500-byte MTU)



- zero-copy is impossible with the 1500-byte MTU

CPU utilization (read 9000-byte MTU)



- The largest gain is expected from zero-copy at 8K I/O size
  - ▷ *But no gain from zero-copy*

Distribution of CPU time (Read)

Distribution of CPU time during the read microbenchmarks for the 8192-byte I/O size.

	Percent Time					
	no optimizations		checksum offload		zero-copy	
MTU (byte)	1500	9000	1500	9000	1500	9000
Idle	64.84	74.77	69.08	78.74	n/a	79.89
iSCSI driver	2.01	1.99	1.93	1.79	n/a	2.12
SCSI subsystem	1.44	1.49	1.25	1.19	n/a	1.29
Copy & Checksum	6.78	6.65	3.81	3.77	n/a	0.16
NIC driver	2.28	1.92	2.14	1.93	n/a	2.18
TCP/IP	9.84	3.36	9.75	2.92	n/a	4.79
VM system	4.54	3.22	4.28	3.08	n/a	4.89

- Checksum offloading is effective
- Zero-copy optimization results in the worse performance of the VM system

Summary

Zero-copy improves the performance upon writing

- But it does not work with iSCSI digest

Zero-copy is not effective upon reading

- Page remapping is not effective with the 8192-byte MTU
  - *The gain form page remapping is smaller than the performance drop from it*
  - *The linux kernel benefits from the simplicity of the VM system very much*

The gain form TCP segmentation offloading is not clear

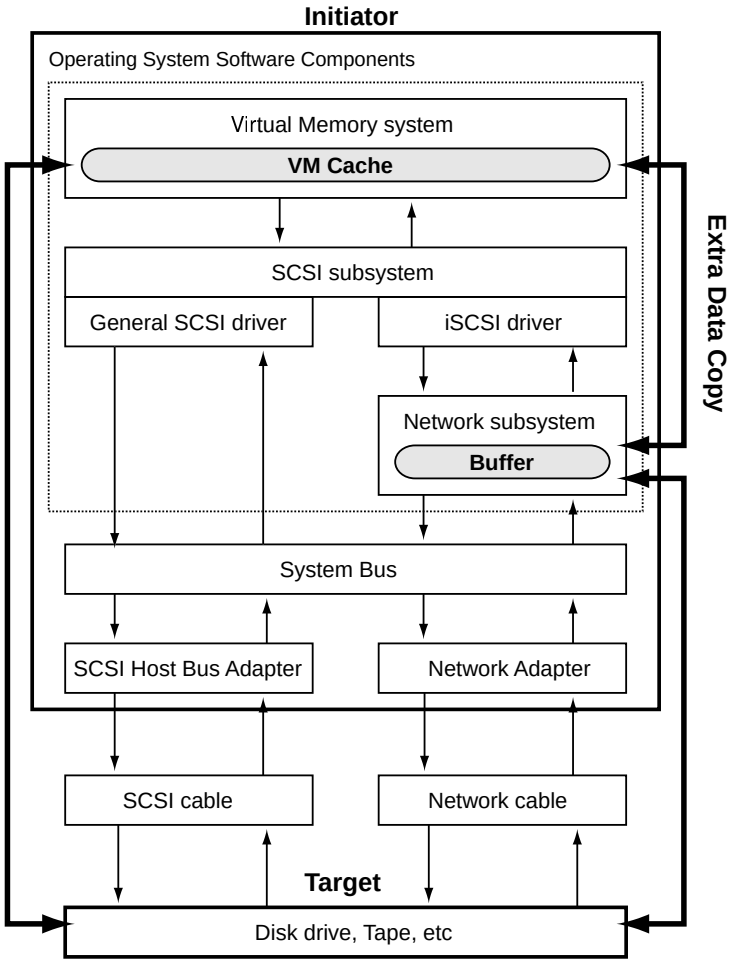
The overhead of iSCSI is mainly due to TCP/IP protocol processing

Backup

Outline

- What is iSCSI?
- Questions
- Useful general NIC features
- The copying problem
- Implementation issues
- Results of microbenchmarks

The straightforward implementation



Process of writing

- The SCSI subsystem receives references to dirty pages
- The iSCSI driver copies data from the dirty pages to network buffers
- The NIC transmits data from the network buffers by using DMA

Process of reading

- The NIC places incoming data in network buffers
- The iSCSI driver processes an iSCSI header and copies data from the network buffers to appropriate pages

Experiment environment

Initiator (Dell Precision Workstation 530)

- Modified version of Linux kernel 2.5.67
- Modified version of IBM's initiator driver
- Two 2 GHz Xeon processors
- 1 GB of PC800 RDRAM main memory
- Intel Pro/1000 MT Server Adapter

Target (IBM TotalStorage IP Storage 200i)

- Linux kernel 2.4.2
- Two 1.13 GHz Pentium III processors
- 1 GB of PC133 SDRAM main memory
- Intel Pro/1000 F network adapter

Gigabit Ethernet switch (Extreme Summit 7i)

Microbenchmark

Microbenchmark software

- It runs in kernel mode
- One I/O request is converted one iSCSI request
  - ▷ *Results are not affected by the page cache*
- All iSCSI commands request the operation on the same disk block address
  - ▷ *The effect of the target factors is kept to a minimum*

Configurations

- NO optimization
- Checksum offloading
- Zero-copy and checksum offloading
- Zero-copy, checksum offloading, and TSO

Measurement of the CPU utilization

- OProfile: Standard system-wide profiler for the Linux kernel