Developing iSCSI Storage Systems on Linux

State of Linux iSCSI support

Tomonori Fujita

NTT Cyber Solutions Laboratories

tomof@acm.org

http://zaal.org/

Ottawa Linux Symposium, 2005/07/21, Ottawa, Canada

Outline

- What is iSCSI ?
- Initiator implementations
- Targets implementations
- Performance
- Remaining issues

What is iSCSI ?

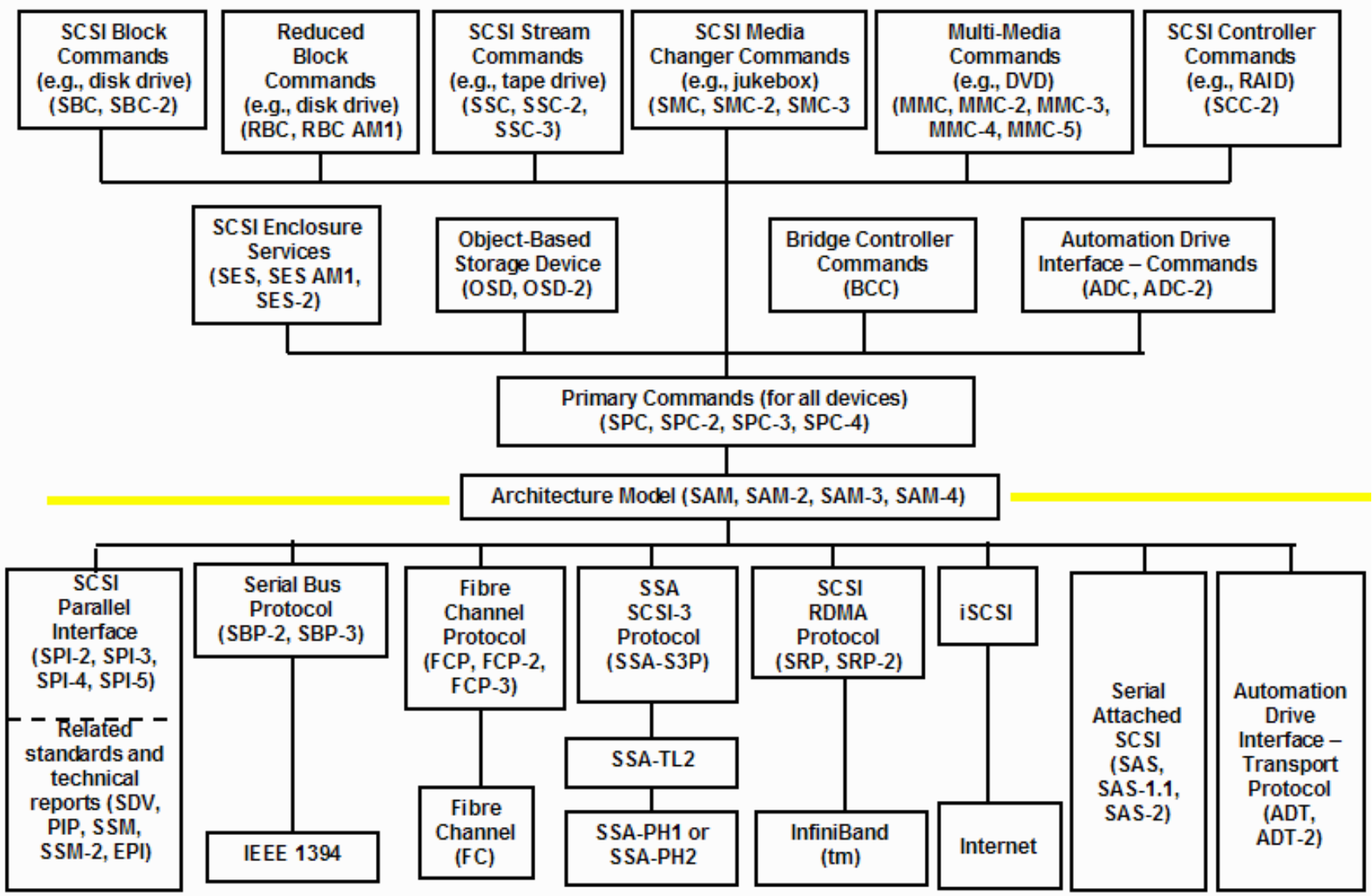
SCSI over TCP/IP

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

TCP/IP is used as one of SCSI transports

- SCSI is a layered architecture
  - *Device type specific command sets (disk, tape, etc)*
  - *Shared command sets (for all device types)*
  - *Transport protocols*
- Various transports are supported
  - *Parallel SCSI (SPI) - Parallel cable*
  - *Fibre Channel (FC) - Fibre Channel*
  - *Serial Attached SCSI (SAS) - Serial cable*
  - *iSCSI - TCP/IP*

SCSI Standard Architecture



taken from `http://t10.org/`

Why is iSCSI ?

New networked storage technology - Storage Area Network (SAN)

- Today the dominant networked storage technology is Fibre Channel (SAN)
  - *Specialized expensive hardware*
- iSCSI can build inexpensive SAN (IP-SAN)

Why not NFS ?

- Some applications does not like NFS
  - *Database*
- iSCSI works with NFS nicely (it does not replace NFS)
  - *A large NFS server can have lots of disk drives by using SAN*

iSCSI initiator implementations in Linux

Initiator

- It issues SCSI commands to request services (hosts)

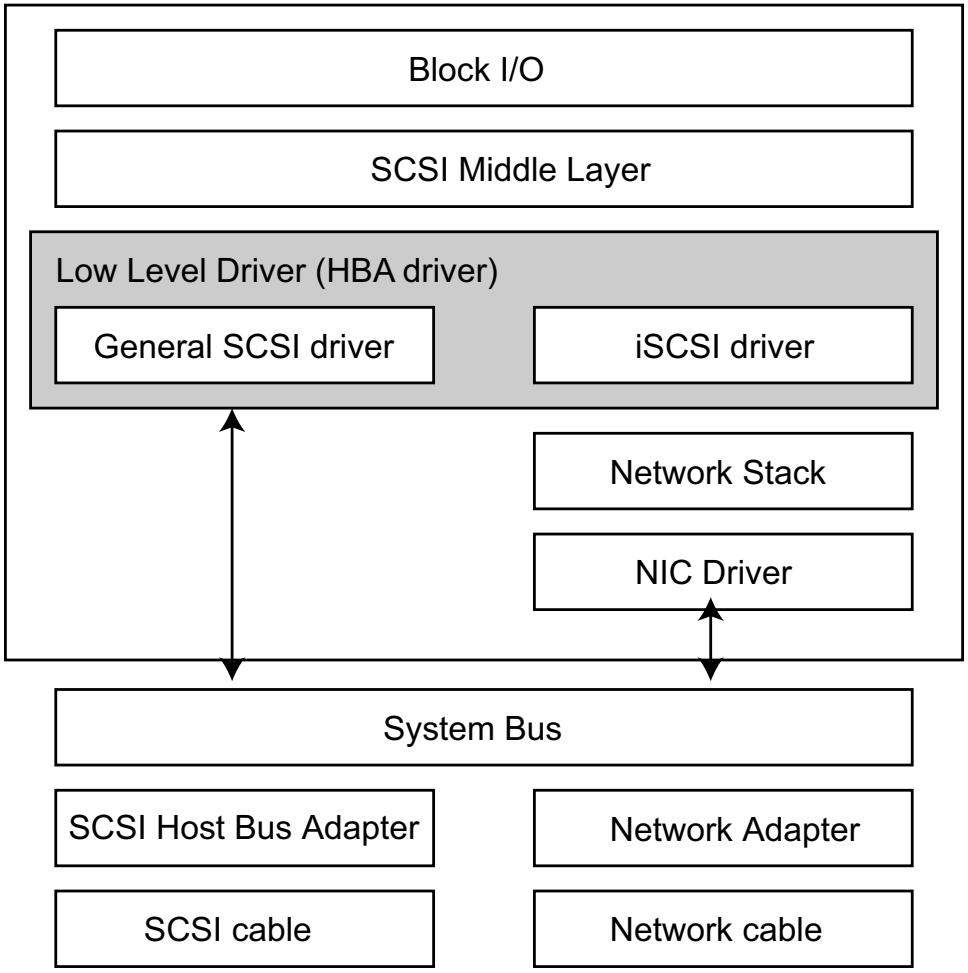
We have several implementations under GPL license

- UNH
- linux-iscsi (sfnet)
- core-iscsi
- open-iscsi

The common design of initiators in the Linux kernel

Implemented as a SCSI Host Bus Adapter (HBA) driver

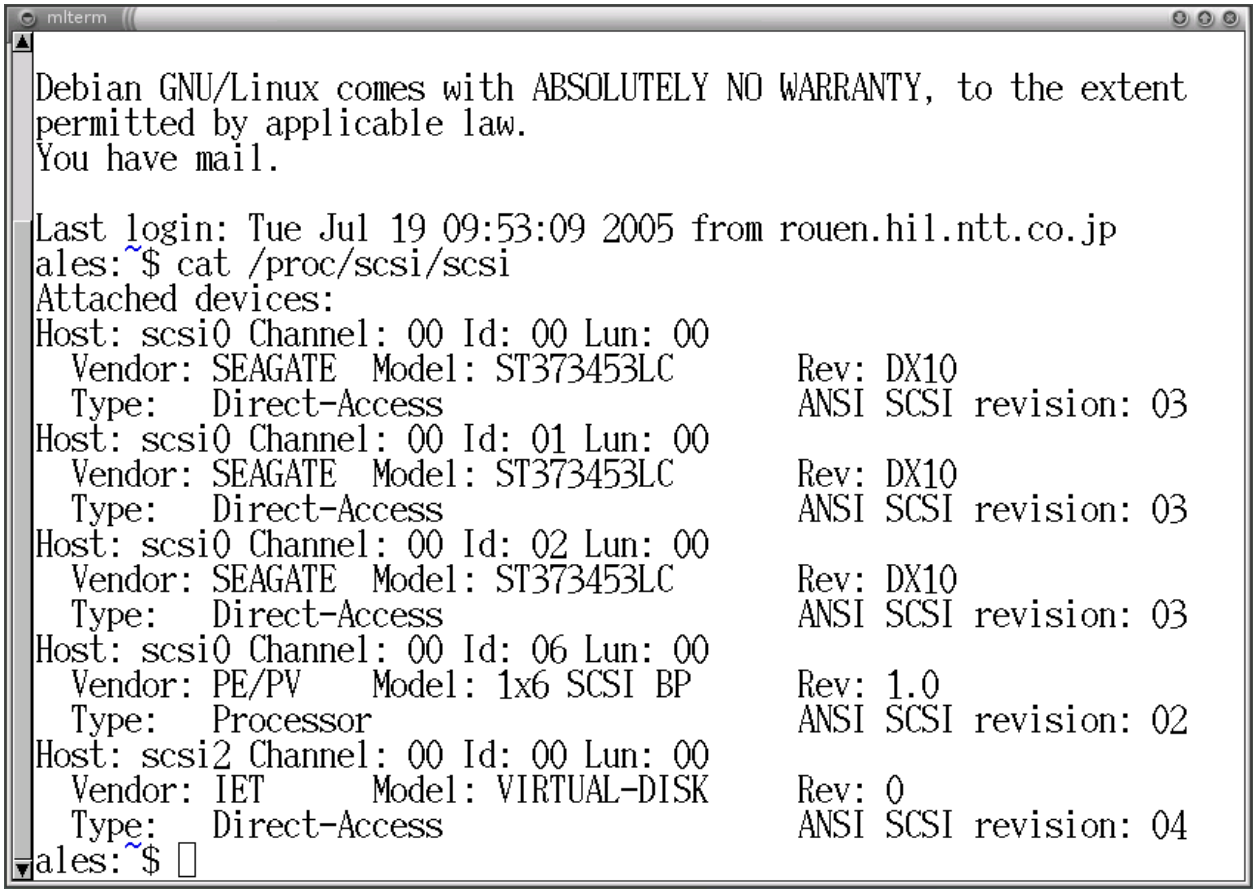
- General LLDs communicate with its HBA using DMA.
- iSCSI initiators communicate with the network stack.



What does iSCSI disk look like ?

No difference between iSCSI and general disk

- Three DAS (direct attached storage) disk drives (SEAGATE ST37345LC)
- One iSCSI disk drive (IET VERTUAL-DISK)



UNH initiator - Initiators in Linux (1)

Things to know

- Implemented by University of New Hampshire people
- Maintained by HP and University of New Hampshire people
- The first iSCSI initiator implementation in Linux
- Aimed for a reference implementation

Advantages

- Most of features in the iSCSI RFC are supported (though some of them are not so useful)

Disadvantages

- Not tested intensively in production environments
- Configuration scheme is not handy
- Not designed properly from Linux kernel perspective
  - ▷ *It doesn't use* `SCSI Transport attributes` *feature in the Linux kernel*
  - ▷ *Everything in kernel space*
  - ▷ *Wrong coding style*

Linux-iscsi (sfnet) - Initiators in Linux (2)

Things to know

- Implemented by Cisco
- Maintained mainly by Cisco and major storage vendor people
- It WAS considered as the standard iSCSI initiator implementation in Linux
- Some distributions include this driver by default
- Note that linux-iscsi 5.x is not sfnet (it's open-iscsi)
- 165 mailing list subscribers

Advantages

- Tested intensively in production environments
- Supported by lots of commercial iSCSI storage systems
- Probably the most robust
- Decent configuration scheme

Disadvantages

- Not more development though still maintained
  - ▷ *It cannot be compiled with 2.6.11 or the newer versions*

open-iscsi - Initiators in Linux (3)

Things to know

- Implemented and maintained by Dmitry Yusupov and Alex Aizman
- It will be merged to the mainline kernel (hopefully in the near future)
- It's also known as linux-iscsi version 5.X
  - ▷ *linux-iscsi version 4.X and 3.X refer to sfnet*
- 111 mailing list subscribers

Advantages

- Designed properly from Linux kernel perspective
  - ▷ *A large portion in user space*
- The most active development
  - ▷ *Many iSCSI developers are focusing on this implementation*

Disadvantages

- It needs more tests in real (production) environments
- Some important features are not implemented yet

core-iscsi - Initiators in Linux (4)

Things to know

- Implemented and maintained by PyX Technologies, Inc
- It had been distributed with PyX Technologies target systems (Not GPL)

Advantages

- Tested intensively with PyX Technologies target systems
- Decent configuration scheme

Disadvantages

- Not active development
  - *It cannot be compiled with 2.6.12-rc1 or the newer versions*
- Not designed properly from Linux kernel perspective
  - *It doesn't use* `SCSI Transport attributes` *feature*
  - *Everything in kernel space*

Summary - Initiators in Linux

- Linux iSCSI initiators are ready for most production environments
- Try open-iscsi in your environment now because it will be merged into the mainline kernel
- sfnet would be a safe choice if you cannot use the latest kernel

How to implement iSCSI storage systems (target)

Target

- It provides services to initiators (storage systems)

Commercial solutions

- Commercial iSCSI storage system use optimized operating systems and specialized hardware
  - *iSCSI specialized NIC, NVRAM, etc*

Open source solutions

- Linux and commodity hardware
  - *PC and Gigabit NIC*
- Linux provides many features useful for iSCSI target systems
  - *Software RAID*
  - *LVM*
- You can implement whatever you want
  - *Management interfaces*
  - *Access control*

iSCSI target implementations in Linux

We have several implementations under GPL license

- UNH
- Ardis
- iSCSI Enterprise target software (aka IET)

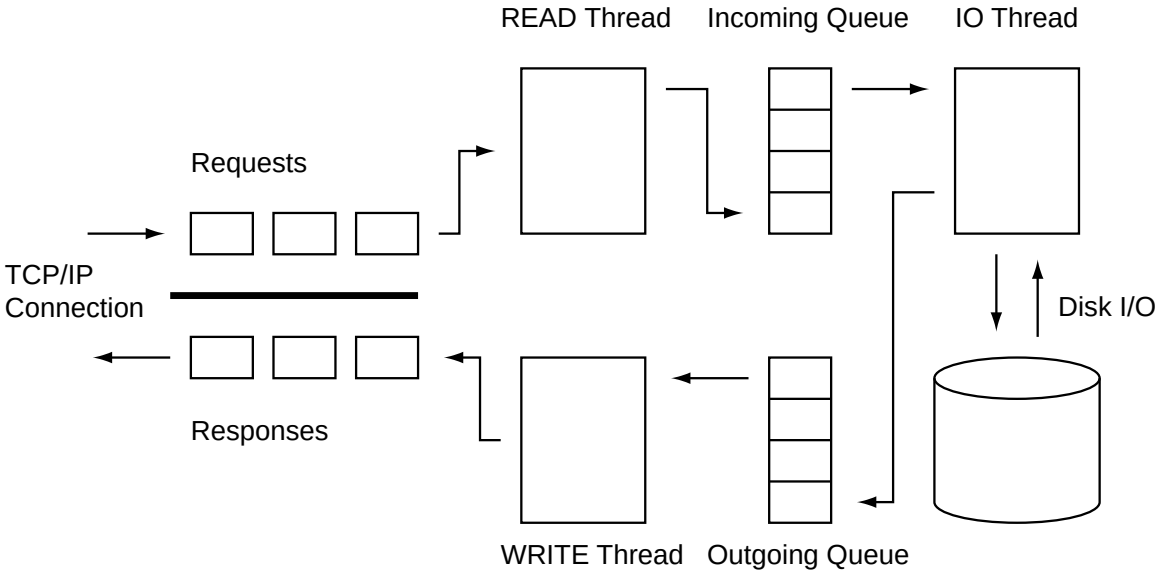
Features

- Virtual devices
  - *Exporting files (including block device files) as disk to initiators*
  - *Exporting iso image files as cdrom drives to initiators*
  - *Exporting files (including block device files) as tape drives to initiators*
- Real devices
  - *Exporting real devices like tape and cdrom drives*

The common design of targets in the Linux kernel

All the implementations run in Linux space

- They need the full control of page cache
  - ▷ *Intelligent caching like commercial storage systems*
- They interact with hardware like NVRAM



UNH target - targets in Linux

Things to know

- Implemented by University of New Hampshire people
- Maintained by HP and University of New Hampshire people
- Probably the first iSCSI target implementation in Linux
- Aimed to debug UNH initiator implementation

Advantages

- Most of features in the iSCSI RFC are supported

Disadvantages

- Unnecessarily complicated design
  - ▷ *Designed for multiple transport protocols at the beginning*
- Configuration scheme is not handy
  - ▷ *No dynamic changes to configurations*
  - ▷ *Very complicated*

Ardis target - targets in Linux

Things to know

- Implemented by Ardis technologies
- It keeps unchanged for already one and a half years

Advantages

- Decent design

Disadvantages

- Probably dead
- Support only 2.4 kernels
- Need a kernel patch
- Configuration scheme is not handy
  - ▷ *No dynamic changes to configurations*
- Buggy

iSCSI Enterprise Target (IET) - targets in Linux

Things to know

- Started as a fork from Ardis
- Maintained by me
- The most part of Ardis code has gone (especially in kernel space)
- 203 mailing list subscribers

Advantages

- Decent configuration scheme
  - *Very simple*
  - *All configurations are dynamically changeable*
  - *Access control based on initiator address and target name patterns*
- Tested heavily
  - *Some brave people use it in production use*

Disadvantages

- Some important features are not implemented yet

The detailed designs of initiators

Things to know

- open-iscsi integrates iSCSI processing into the network stack
  - ▷ *A bit complicated but effective code*

UNH, sfnet, core-iscsi

- tx thread writes to sockets
- rx thread read from sockets
- queuecommand function links SCSI commands to an internal link
- tx thread unlinks SCSI commands, builds iSCSI PDUs, and send them
- rx thread calls cmnd->done

open-iscsi

- queuecommand function builds iSCSI PDUs and send them
- sock->data_ready calls cmnd->done

The performance comparison of initiators

Things to know

- There is no performance difference due to the bottleneck of target's disk in most cases
- All initiators are fast enough for 1 Gigabit Ethernet and normal slow target's disk
- You can see the difference if you use the combination of 10 Gigabit Ethernet and a really fast target like RAM disk

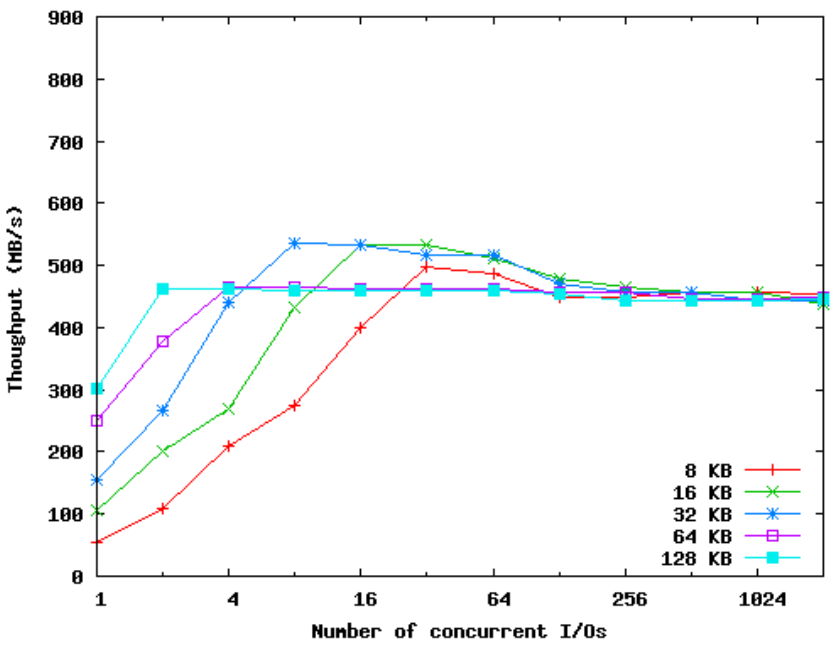
core-iscsi vs. open-iscsi

- Both claim that it's a high-performance iSCSI initiator
- core-iscsi 1.6.2.0-rc1 / open-iscsi 0.3rc6-363

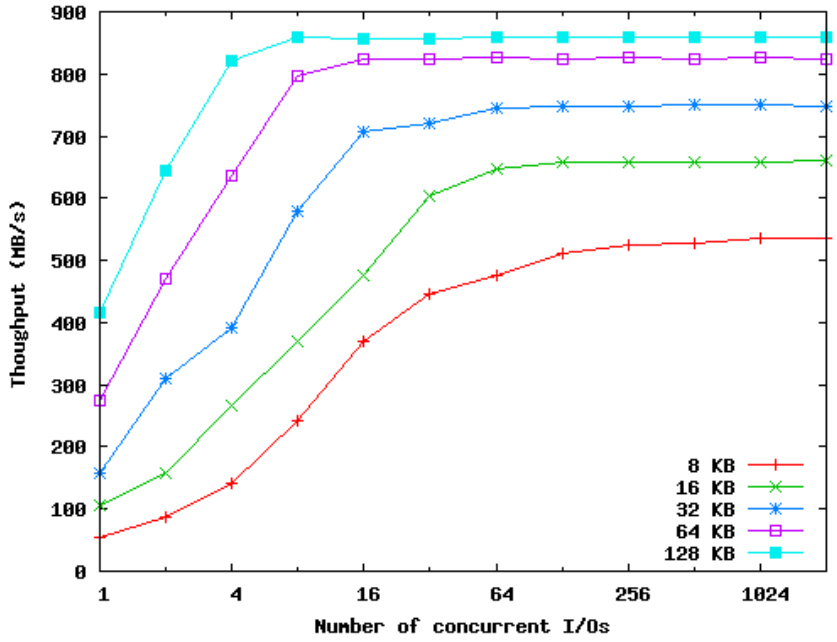
Experiment environment

- Two Opteron (2.4 GHz) boxes are connected directly
- Chelsio T110 10GbE NIC (full TOE NIC, all TCP functionality is implemented on NIC chip)
- Benchmark software runs in kernel space (uses bio functions)
- IET runs in nullio mode (dumps received data and sends meaningless data)

Sequential read



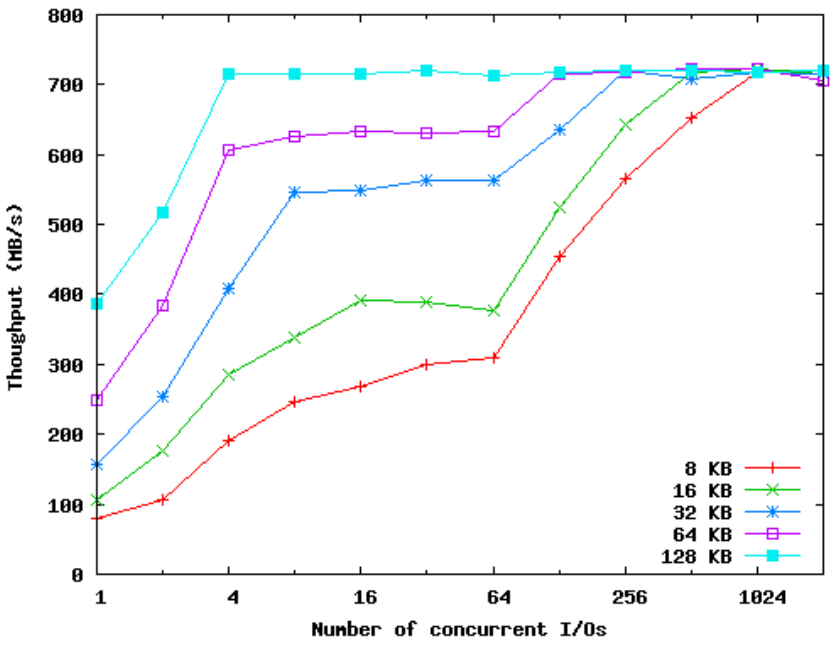
core-iscsi



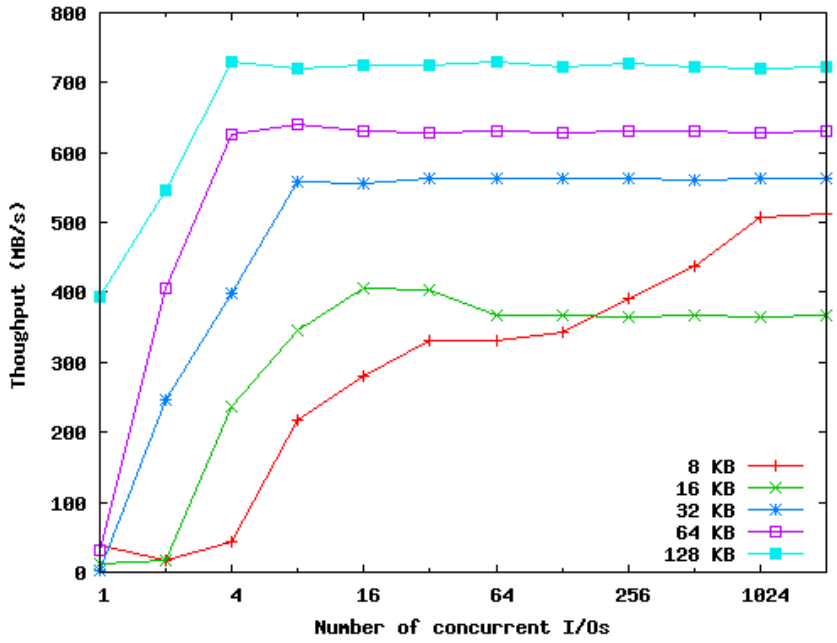
open-iscsi

- open-iscsi is much faster (due to the iSCSI processing integrated into the network stack)

Sequential write



core-iscsi



open-iscsi

- Both can achieve 700 MB/s
- open-iscsi performs terribly with single concurrent I/O (since the combination of open-iscsi and Chelsio network stack leads to some bugs)

Remaining issues with initiators (1)

System locks in out-of-memory situations

- Symptoms
  - *Suppose that the system is in an OOM situation due to lots of dirty pages*
  - *The system tries to write dirty pages to iSCSI disk to reclaim memory*
  - *The network stack fails to allocate memory since the system is already in an OOM situation*
  - *The system cannot write dirty pages thus it cannot reclaim memory*
- Problem
  - *You cannot preallocate some memory in the network stack for an OOM situations*
  - *Even in an OOM situations, you must send and receive several packets to write one write request at least*
- Status
  - *The developers cannot do nothing (New features in the network stack are necessary)*

Remaining issues with initiators (2)

Unnecessarily I/O errors happens under heavy load

- Symptoms
  - *iSCSI initiators have some time limit for some operations*
  - *Suppose that the target sends a ping thus the initiator must respond immediately*
  - *The initiator cannot send since the system is under heavy load*
  - *The target judges the initiator dead and closes the connection*
- Problem
  - *iSCSI initiator needs some real-time stuff (scheduling and network processing)*
  - *The issue gets complicated especially with open-iscsi because the most parts are in user-space*
- Status
  - *open-iscsi use mlock to avoid page out*
  - *New feature in the network stack are necessary again*

Remaining issues with targets (1)

Unnecessary I/O error happens under heavy load

- Symptoms
  - *iSCSI targets have some time limit for some operations*
  - *Suppose that the initiator sends a request thus the target must respond immediately*
  - *The target cannot send since the system is under heavy load*
  - *The initiator judges the target dead and aborts the request*
- Problem
  - *iSCSI target needs some real-time stuff (scheduling and network processing)*
- Status
  - *New features in the network stack are necessary again*

Remaining issues with targets (2)

Write back vs. write through (disk drive cache algorithm)

- Symptoms
  - *The target uses a file with the vfs interface to export it as disk to initiators*
  - *With writeback, the target doesn't need to sync h dirty page*
  - *The amount of dirty page cache increases too much (several hundreds mega bytes) compared with normal disk cache (several mega bytes)*
- Problem
  - *The target cannot control the amount of dirty page cache*
- Status
  - *IET supports only write through*