Analysis of iSCSI Target Software

FUJITA Tomonori

NTT Cyber Solutions Laboratories

tomof@acm.org

SNAPI'04, Antibes Juan-les-Pins, France

What is iSCSI

SCSI over TCP/IP

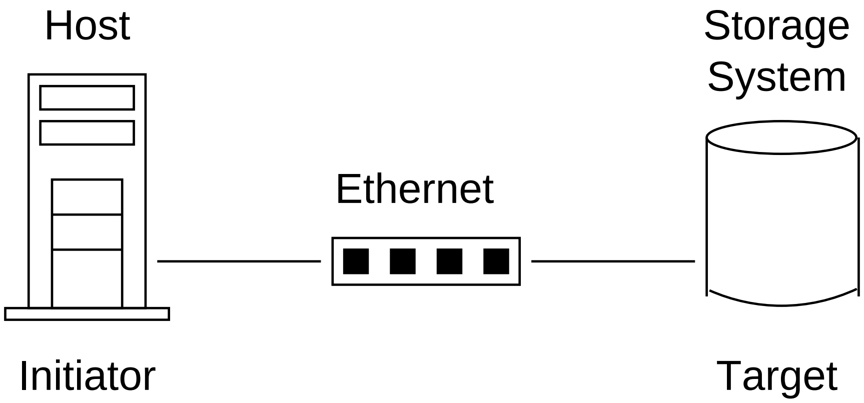
- SCSI commands are encapsulated into IP packets
- A host and a storage system are connected with Ethernet

New networked storage technology

- Today the dominant networked storage technology is Fibre Channel

Advantages

- IP networks infrastructure is inexpensive
- We are familiar with IP network and SCSI technology



Various iSCSI storage systems

Commercial solutions

- Optimized operating systems and specialized hardware
  - ▷ *iSCSI NIC, NVRAM, etc*
- Many vendors sell iSCSI storage systems
  - ▷ *IBM, EMC, Hitachi, NetApp, etc*

Open source solutions

- Linux and commodity hardware
  - ▷ *PC and Gigabit NIC*
- Two well-known open source iSCSI targets
  - ▷ *UNH - New Hampshire Univ. and HP*
  - ▷ *Ardis - Ardis Technologies Corp.*

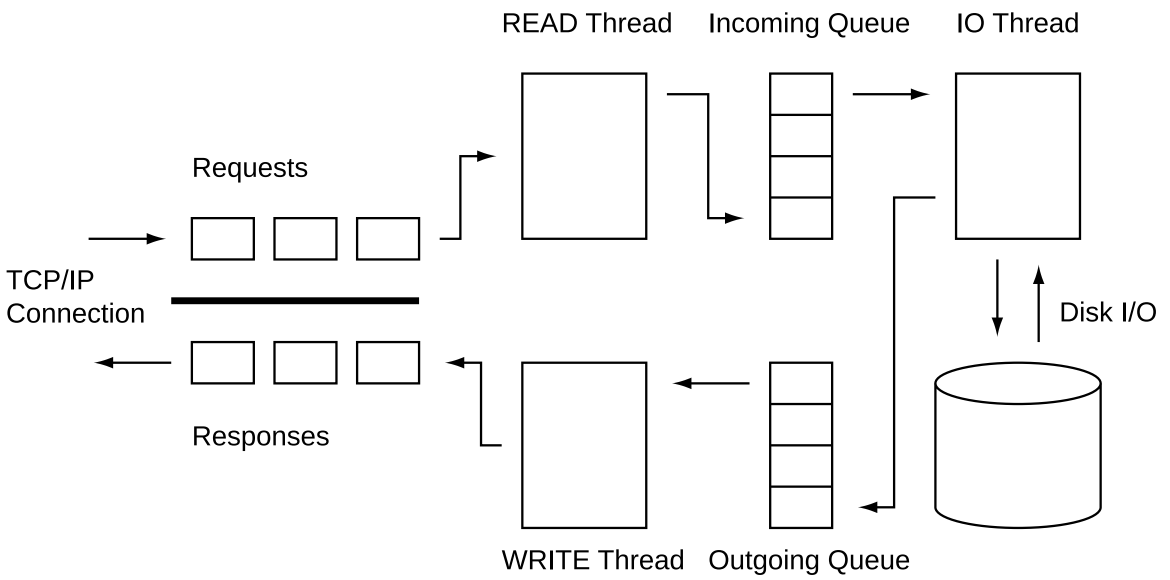
Questions

What are the differences among open source targets?

- UNH-disk
  - ▷ *UNH iSCSI target configured as DISKIO mode*
- UNH-file
  - ▷ *UNH iSCSI target configured as FILEIO mode*
- Ardis
- Threaded-Ardis
  - ▷ *Our new implementation based on the Ardis code*

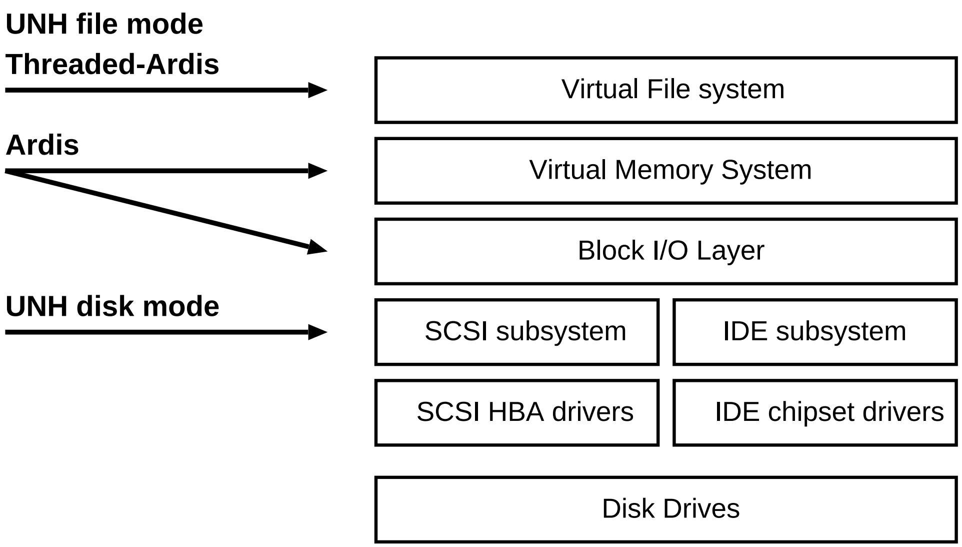
Are open source solutions comparable to commercial products?

Architecture overview



- Three kernel threads provides the major part of the iSCSI target functionality
  - ▷ *Read thread receives data from initiators*
  - ▷ *I/O thread performs I/O operations*
  - ▷ *Write thread sends responses to initiators*
- There are major differences in I/O thread design

I/O thread designs



Linux kernel provides several interfaces for I/O operations

- Virtual file system (vfs) interface
  - ▷ *UNH-file and Threaded-Ardis*
- Modified Virtual Memory subsystem and block I/O layer
  - ▷ *Ardis*
- SCSI subsystem interface
  - ▷ *UNH-disk*

Assessment criteria - kernel modifications

Ardis modifies the source code of Linux kernels

- Ardis implements own I/O functions by using a modified kernel
  - ▷ *Improved performance because Linux standard functions are not optimized for iSCSI targets*
- More complicated code
  - ▷ *Increased cost for implementation*
  - ▷ *Increased cost for maintenance of the code*
  - ▷ *Lowered stability*

Assessment criteria - Interoperability

UNH-file breaks interoperability with direct-attached storage

- UNH-file sends the response of WRITE commands before the data are written to disk
  - *Improved performance due to delayed disk I/O operations*
- It require software changes in the existing operating system or application
  - *Data integrity protection techniques like journaling don't work*
  - *Data corruption may happen after a system crash*

Delayed write is different from disk drive cache

- Delayed write aggressively reorders write operations
  - *It breaks data integrity protection techniques*

Assessment criteria - Disk management

UNH-disk suffers poor disk management features

- UNH-disk design can use only SCSI disk drives
  - ▷ *The SCSI subsystem is lower than the block I/O layer providing disk management features*
- UNH-disk doesn't supports various block devices
  - ▷ *IDE*
  - ▷ *Serial-ATA*
- UNH-disk doesn't supports virtual block devices
  - ▷ *Flexible storage management (LVM)*
  - ▷ *Redundancy (software RAID)*

Assessment criteria - Performance

UNH-disk suffers poor read performance

- UNH-disk design cannot use page cache minimizing disk I/O by storing data
  - ▷ *The SCSI subsystem is lower than VM system providing page cache functionality*
- Every SCSI read command invokes a disk I/O
  - ▷ *It leads to poor read performance*

UNH-file suffers poor performance

- UNH-file design cannot perform SCSI commands simultaneously
  - ▷ *The vfs interface works synchronously - The I/O thread sleeps until I/O completions*
- All I/O operations are serialized unnecessarily
  - ▷ *It leads to poor performance*
- Threaded-Ardis uses multiple I/O threads to avoid this problem
  - ▷ *This design enables Threaded-Ardis to handle SCSI commands synchronously*

System Comparison

Examined targets

- Ardis
- Threaded-Ardis
- UNH-disk
- UNH-file
- UNH-file-sync
  - ▷ *Modified UNH-file providing interoperability about WRITE commands*
- Entry-class commercial iSCSI target system
  - ▷ *The product details are confidential*

Benchmark software

- Microbenchmarks
  - ▷ *Sequential write*
  - ▷ *Sequential read*
- Macrobenchmarks
  - ▷ *Postmark - write intensive*
  - ▷ *Network server benchmark - read intensive*

Experimental infrastructure

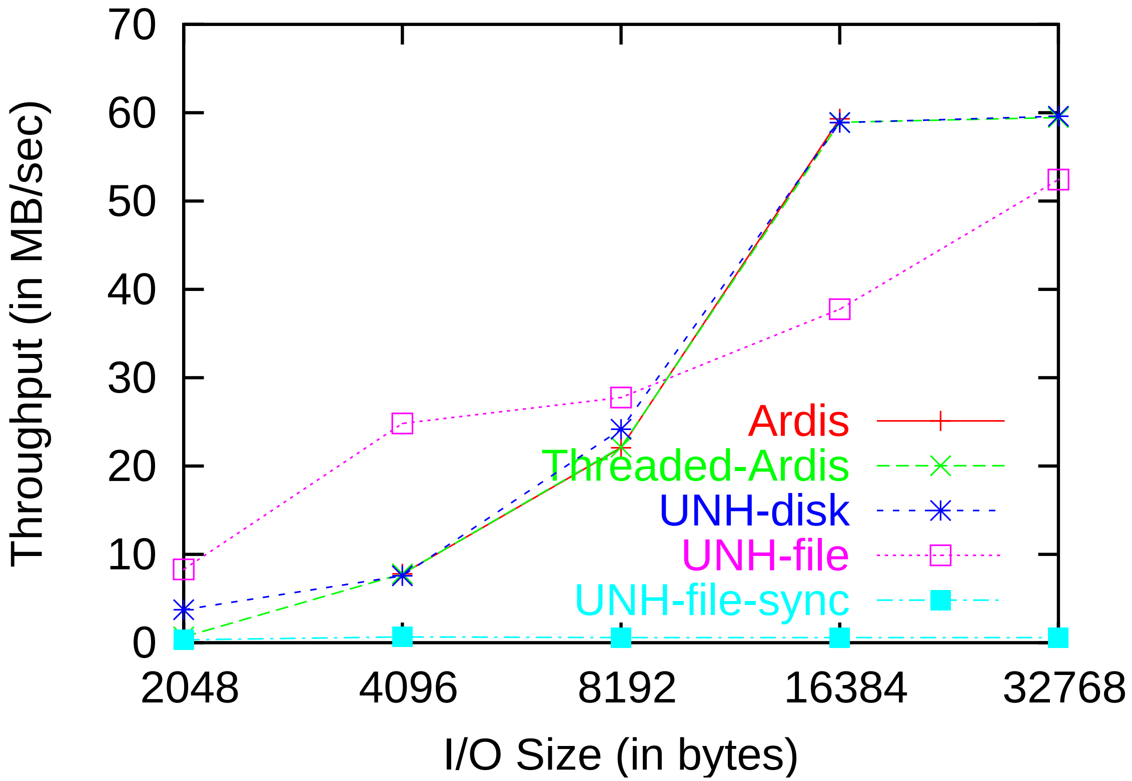
Initiator

- Linux kernel 2.6.4
- 2 GHz Xeon processor
- 1 GB main memory
- Cisco initiator 4.0.1.1

Target

- Linux kernel 2.4.25
- 2 GHz Xeon processor
- 2 GB main memory
- Maxtor Atlas 10K, 36.7 GB 10,000 RPM SCSI disks
- LSI Logic 53C1030 Ultra320 SCSI chip

Microbenchmarks - Write

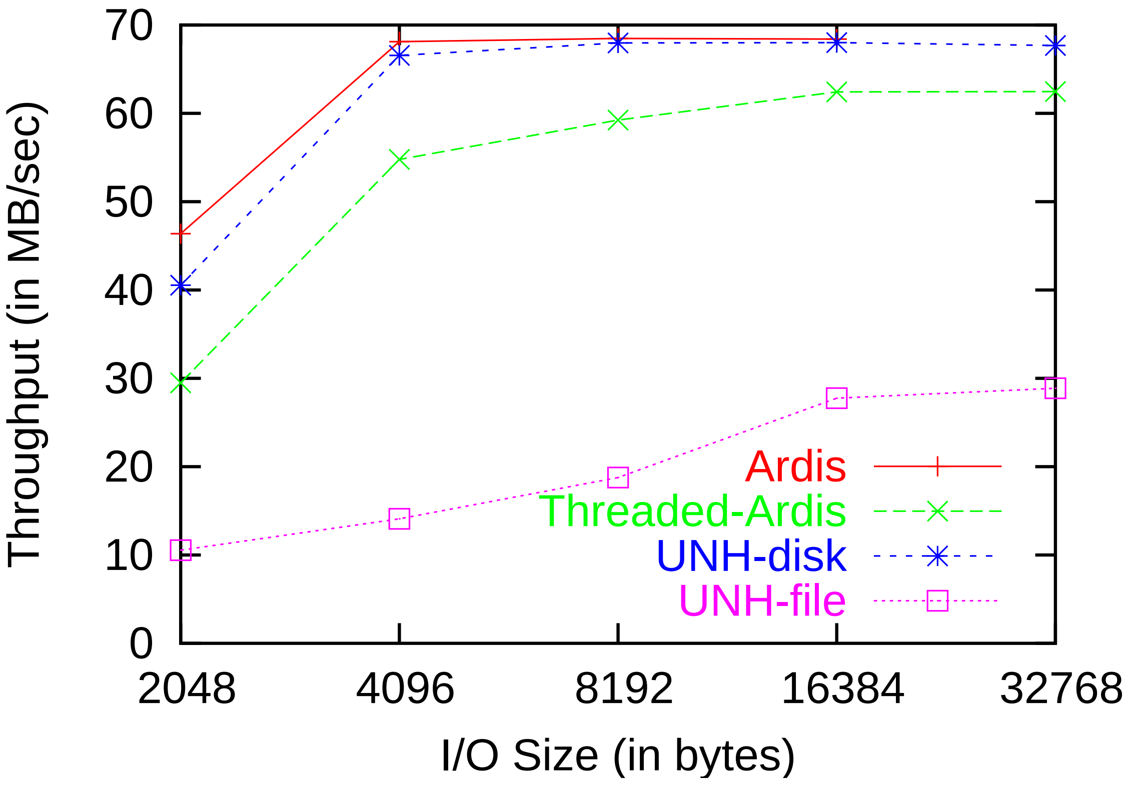


- UNH-file produces better performance due to delayed write
  - ▷ *No I/O operations are performed up to 8 KB*

- No clear gain from specialized functions

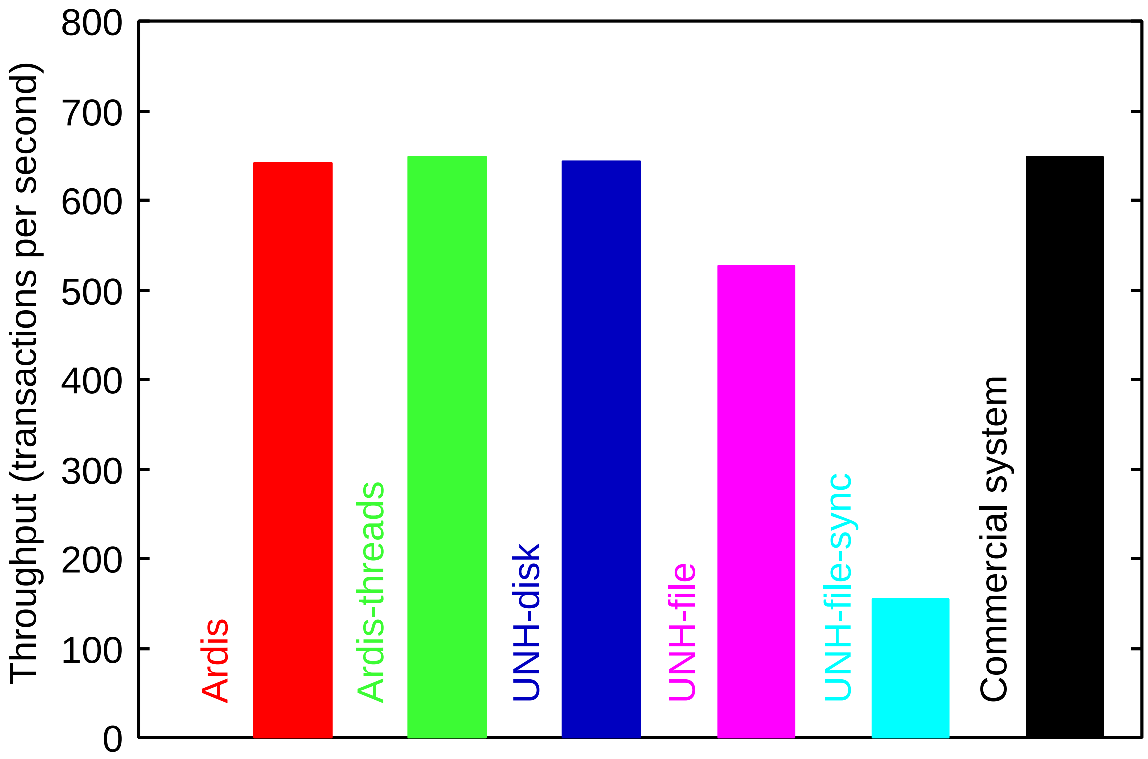
- Ardis couldn’t complete the benchmarks with 2KB or 32KB due to its bugs.

Microbenchmarks - Read



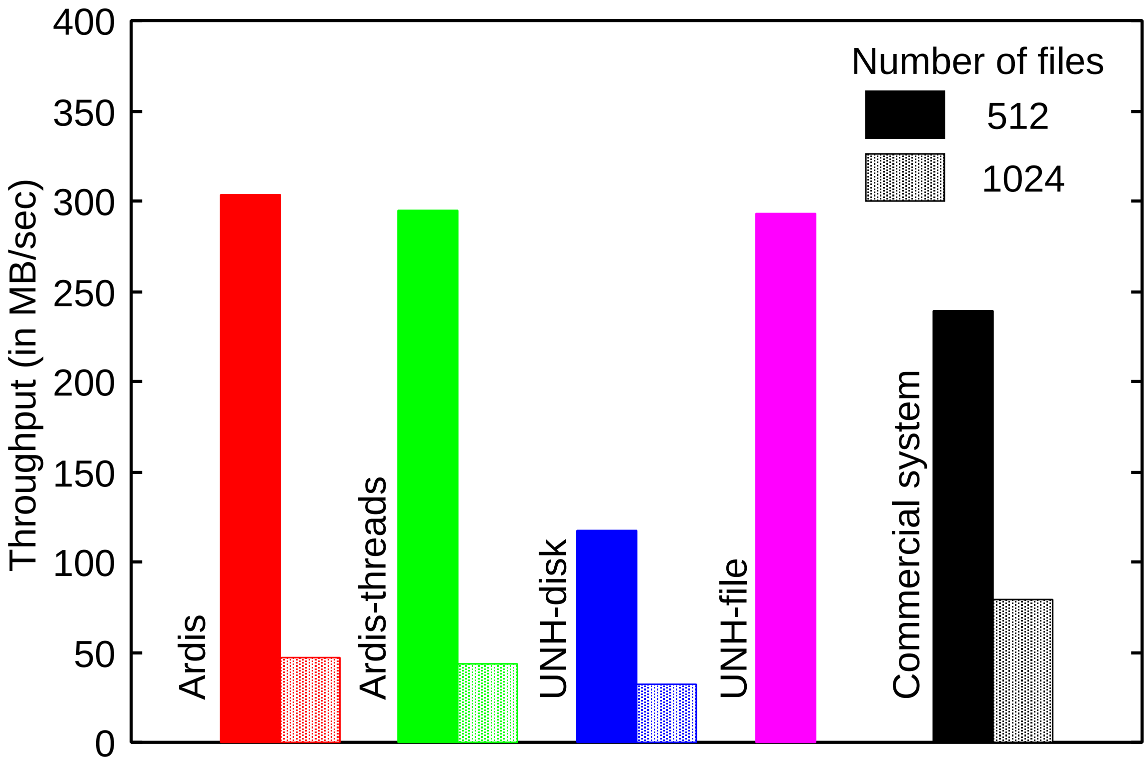
- Ardis performance is slightly better than the others
- UNH-file performance is poor
  - ▷ *It needs to perform I/O operations*
  - ▷ *It cannot handle them simultaneously*

Postmark



- Postmark is designed to measure performance in the ephemeral small-file workloads seen by ISP
  - ▷ *20,000 files, 50,000 transactions, and file sizes of between 512 bytes and 16 KB*
- Comparable throughputs except for the UNH-file-sync and UNH-file targets

Network Server benchmark



- The benchmark repeatedly reads many files in one directory in random order
  - ▷ *512, 2 MB files 4,096 times in total*
  - ▷ *1,024, 2 MB files 4,096 times in total*

- Large effects of page cache on the performances
  - ▷ *Page cache hit rate of 85.4% and 25.4% respectively at the target*

Summary

Our design advantages

- Low cost
  - ▷ *No kernel modification*
  - ▷ *No changes to existing software*
- Rich disk management features
  - ▷ *Exploiting kernel virtualization of block devices*
- Comparable performances in common workloads
  - ▷ *The iSCSI target modifying a Linux kernel*
  - ▷ *Entry-class commercial iSCSI target system*

Current development status

- The code was released publicly at May 25, 2004
  - ▷ **http://sourceforge.net/projects/iscsitarget/**
- The code has been actively maintained by several developers