Design, Implementation and Evaluation of a High Performance I/O Subsystem in Linux

Pramodh Mallipatna

Masters Thesis M.S. Computer Engineering University of Kansas July 18, 2000

Thesis Committee:

Dr. Douglas Niehaus, Chair Dr. Joseph B. Evans Dr. Gary Minden

Presentation Outline

- Motivation
- Related Efforts
- Implementation of FlexIO
 - Disk-Disk I/O
 - Disk-Network I/O
 - Network-Network I/O
 - WildForce I/O
- Evaluation
- Conclusions & Future Work

Motivation

- Problem Statement
 - Modern systems use faster CPUs, hardware and more main memory
 - Fast I/O applications
 - E.g. Streaming audio, video and some customized applications
 - Possible bottle-necks for fast I/O applications
 - Software I/O operations
 - conventional software I/O APIs
 - Data flow crosses address space boundaries
 - Multiple data copies during I/O
- Approach
 - Re-design of the I/O subsystem
- Our focus
 - Minimize the copy operations
 - Let the data flow remain in the kernel

Related Efforts

• The Hardware Approach

- Using ASICs, FPGAs etc.

• ASICs for Packet Forwarding, Buffer Management

• Direct Memory Access (DMA)

– Data transfer without the involvement of CPUs

• High speed devices - Hard disk controllers, Ethernet and ATM cards

• Efficient management of Caches

– LRU/LFU, Second chance algorithm

• Early Eviction LRU, an adaptive page replacement algorithm

Implementation (FlexIO)

- FlexIO A pseudo device driver for high performance I/O in the kernel
- Features
 - Built into the kernel or a Linux loadable module
 - Enables user processes to describe and control data flows among kernel components
 - Currently supports
 - Disk-Disk I/O (Ext2 files and Raw Partitions)
 - Disk-Network I/O (Ethernet and ATM networks)
 - Network-Network I/O
 - WildForce I/O
 - Supports Memory Mapped I/O

Disk-Disk I/O using FlexIO

- Data flow remains in the kernel
- Supports I/O between
 - Ext2 files
 - I/O using kernel buffers
 - I/O at the Buffer Cache level
 - Memory Mapped I/O
 - Raw Partitions







Disk-Network I/O using FlexIO

- Data flow remains in the kernel
- Supports Ethernet and ATM
- Runs directly over Link layer
- No error correction incorporated - Lost packets are lost - We do not lose any





- Own protocol type defined
- Protocol Type 0x0A0A
- FlexIO header
 - Port
 - Packet Information



Disk-Network I/O using FlexIO...

- Network Bottom-Half hands the packet to FlexIO
- FlexIO Queue at the receiver
- FlexIO Task Queue to service the queue
 - Identifies flow specific information
 - Writes the data to the disk





Network I/O between three hosts with the intermediate host just forwarding the packets

- End hosts Same functionality as before
- Intermediate hosts Forward the packets

WildForce I/O using FlexIO

- WildForce
 - PCI card housing FPGAs
 - Has on-board memory
 - Used for digital data processing
 - Device driver and user level APIs to interact with the board
 - No access to the driver source
- I/O using FlexIO
 - Accesses Character Device Table using WildForce Major number
 - WildForce configuration still done using WildForce user level APIs
 - Data flow remains in the kernel
 - Supports Memory Mapping user and kernel memory segments



- Maps user buffers to • kernel space
- Calls WildForce • driver's entry points
- WildForce driver still • copies data across address spaces





•

•

•

Evaluation of FlexIO

- FlexIO driver was tested by
 - Transferring various sets of data using FlexIO
 - Disk-Disk I/O
 - Disk-Network I/O
 - Network-Network I/O
 - WildForce I/O
 - Compared with conventional I/O mechanisms
 - File System and Socket APIs for Disk and Network I/O
 - WildForce user level APIs for WildForce I/O

Disk-Disk I/O Evaluation

- FlexIO compared with a user process, both doing Disk-Disk I/O between
 - Ext2 files on same and different partitions of an IDE disk
 - Ext2 files on same and different partitions of a SCSI disk
- I/O using FlexIO
 - Using a kernel buffer
 - I/O at the Buffer Cache level
 - Memory Mapped I/O
- User Process I/O
 - Using a user buffer
 - Memory Mapped I/O













Disk-Network I/O Evaluation

• FlexIO compared with a user process using UDP sockets



Disk-Network I/O Results

- UDP offers better throughput for small file transfers - with larger send and receive buffers
- Unable to transfer large files at high rates
- FlexIO offers better sustained throughput than UDP for large files



Throughput of Disk-Network I/O



WildForce I/O Evaluation

- FlexIO compared with a user process using WildForce APIs for I/O
- Used to perform I/O for SAR processing





Conclusions

- Conventional I/O APIs are not suited for all kinds of applications
- I/O flow need not always cross address space boundaries
- FlexIO
 - Lets user processes define and control data flows in the kernel
 - Memory Mapped I/O supported
 - Supports
 - Disk-Disk I/O
 - Disk-Network I/O
 - Network-Network I/O
 - WildForce I/O
 - Disk-Disk I/O: Offers the same performance as that of File System APIs
 - Disk-Network I/O and Network-Network I/O: Offers higher throughput than UDP based transfers
 - WildForce I/O: Offers the same performance as that of WildForce APIs

Conclusions...

- Framework for
 - generalized high performance I/O subsystem
 - Assembling data flow for computations across sets of machines
- Demonstrates how I/O among dissimilar input and output entities can be controlled from within the kernel
 - E.g. Disk-Network I/O, File-WildForce I/O, Network-WildForce I/O

Future Work

- Streaming real-time audio and video data
- Use of Memory Mapping functionality for WildForce I/O
- Integration with the higher layer network protocols
- Flow control in FlexIO
- Testing the driver under real-time