ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Achieving low latency networks through high performance computing

Chandrika Prasad[1], Veena G.S[2], Chirag Agrawal[3], Robin Srivastava[4]

Department of CSE, MSRIT, Bangalore, Karnataka, India[1,2,3,4]

Abstract: We define latency as the time taken to deliver a unit of data from one point to another point in the system. Low latency networks refer to the networks where systems, their architecture, hardware and protocols are designed to bring down this latency. The question that why latency is so important can be answered by mentioning the fact that many applications such as voice transmission, networked gaming, and video transmission, interactive sessions solely depend on the latency of the network. The components of latency include Hardware: Every hardware comes with its own advantages and limitations. For example, some have fixed packet size whereas other may have variable size. Routers and Switches: All the networks components follow their own queuing strategies or congestion control strategies. Traditionally, processing of packets is dependent on the rate on incoming packets. System Latency: The packets to and fro between the application the network interface and this surely forms a part of the latency. Potentially, interruption by system can introduce infinite amount of latency.
OS Latency: The processing of the packets by the OS consumes time. It de-multiplexes the packets and sends them to their respective destinations.
Application Latency: The application need sufficient amount of CPU resources to perform the task.

Keywords: PCI, PCI-x, DLL, UNET

INTRODUCTION

In this paper we aim to study various ways to improve network latency using high performance computing techniques. We first discuss some techniques like UNET, Zero copy architecture and vertical partitioning of OS which had potential to address this problem directly but could not get worldwide attention due to various reasons. We then discuss the Shared Memory Model (here forth SMM) which addresses the core problem i.e. interrupt-driven architecture of the systems. The SMM is suffers with problem of Application Synchronization which then addressed with the solutions like VM Signaling and out-of-band signaling.

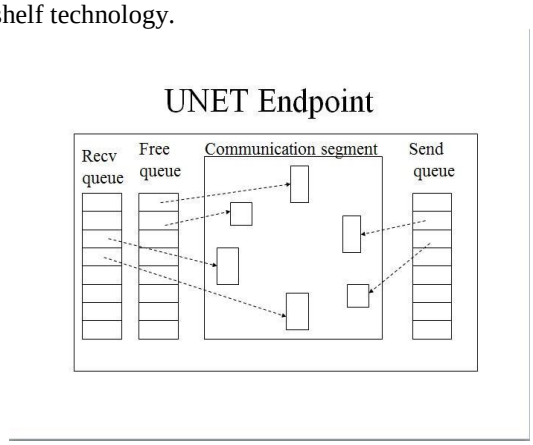
Survey

UNet: A User-Level Network Interface

The U-Net communication architecture provides processes with a virtual view of a network interface to enable User-Level access to high-speed communication devices. ATM communication hardware which is used as the architecture here, removes the kernel from the communication path, while stall providing full protection. U-Net uses per application message queues to send and receive data. But, this idea complicates what an application needs to do.

The U-Net architecture focuses on reducing the processing overhead required to send and receive messages as well as on providing flexible access to the lowest layer of the network. The intent is three-fold:
- help building local area setting with low latency,
- exploit the full network bandwidth even with small messages, and
- novel communication protocols are easily facilitated.

The three aspects that set U-Net apart from other techniques are:
- prioritizing high bandwidth and low latency with small messages,
- emphasis on protocol design and integration flexibility, and
- aim to fulfill the above two mentioned goals using off-the shelf technology.



Zero-Copy Architecture

Overview

In simple words, zero-copy architecture eliminates the need to copy data up to the application.

For e.g., if we read a file and then send it through a traditional network then four context switches and four data copies will be required. If we send the same file through zero-copy then the context switches will be reduces to two and either all or at least half of the CPU copies will be eliminated.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Current Bottlenecks-The need to Zero Copy

Contemporarily, the technology that has been deployed is Parallel Component Interconnect (PCI) and Extended Parallel Component Interconnect (PCI-X).The backbone of this technology is exclusive contention of the processor bus. Peripherals as end-points impersonate as bus master and are directly connected to its bus. Winning the contention of the bus leads to assigned address in processor's address space.

The main locality of bottleneck can be centered around connectivity between system processor , memory controller and the system I/O interconnects which forms the core of all domains of computing.

With the plethora of new and high speed end systems , PCI and extended PCI interconnects have become major performance hurdles. Passing data over conventional interconnects is achieved by forcing data through many paths using interrupts which leads to ideal state of the processor. To mitigate it, bridge chips are connected to peripheral devices and system Central processing unit. But it's implementation leads to trade off in fields of system development costs , increased power consumption and lower reliability.

Underutilization is due to the fact that processor bus is a parallel bus but exclusive contention per bus cycle leads to it being emulated as serial transmission. Hence, deteriorated performance and high latency in delivering service.

Probable Workaround – Hyper Transport Technology

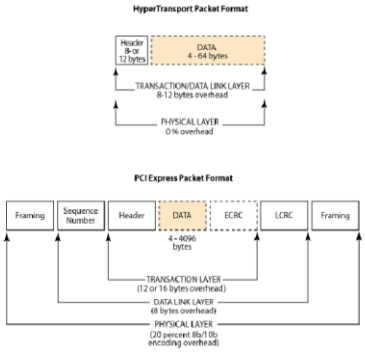
Backbone of this technology is direct peripheral to central processing unit interconnection. It deploys expansion cards which are directly plugged into HTX enabled end systems overlooking conventional end point bottlenecks.

Deployment of Hyper Transport Technology, allows system manufactures to design and launch an exhaustive motherboard platform that can easily blend with current high-performance market. It's integration is as easy as adding a HTX- based peripheral card into the legacy hardware.

Hyper Transport Technology – How edge over PCI and PCI-X

HTX is in-system chip to chip interconnect which enables plug-in subsystems to obtain direct connect performance facility. It can be combined with connector as PCI eliminating clock recovery circuit logic and introducing priority request interleaving. Thus, employing a super lean packet payload protocol.

The following diagrams explain how HTX leads to lower latency and faster response in comparison to its contemporaries.



The diagrams self explains the fact that PCI packet format has extra overheads due to introduction of various different headers like transaction layer which leads to overhead of 12-16 bytes ,DLL overhead of approximately 8 bytes and encoding overhead of 8 to 10 bits per packet. This phenomenon is optimized in HTX by encapsulating all encoding or data link functionalities into a single header which brings down the overhead to 8-12 bytes approximately.

To sum up , HTX is high bandwidth point to point link providing the lowest possible latency in chip to chip and board to board communications. It minimizes the count of buses in a system. It also gels well with legacy implementations by adding it's extension. Being packet-based protocol and deploying clock forwarding technique eliminates many control and command signals.

Implementing Zero-Copy Protocol – The master of all technology

Most primitive interpretation of zero copy protocol can be confined to the following two diagrams:

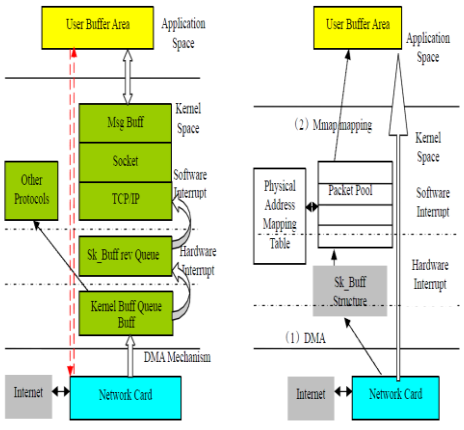


Figure 2. Compare traditional model with zero-copy model

Key implementation differences between conventional and zero copy protocol is the reduction in number of stages through which data is transferred before being available to the corresponding process. Through the process of Direct Memory Access the received data is stored in processor buffer which is readily available to be transmitted to the corresponding process for operation.

The next step is mapping the destination address of packet to the resulting process. In conventional systems, this phenomenon is carried out in the following steps:

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

All data is stored directly into kernel buff queue which prioritizes the transmission of data. When the data is readily available then it is forwarded to upper buffers like SK_Buff deploying protocols like FCFS etc. This introduces extra overhead. Now ,since the data has to be forwarded to corresponding process it has to be transected by lower layer protocols like TCP/IP and assigned a corresponding socket to be available to a process. All these data are stored in final level message buff which waits for scheduler to dispatch corresponding data to assigned user space and then to a user thread.

Zero-copy protocol as mentioned need not maintain copies in different level of buffers. All readily available packets are dumped into a centralized storage module known as packet pool. Since, all data packets has corresponding destination address .This address is mapped to the socket address after which it is promoted to user space sockets and scheduler assigns it a process to be executed. This minimized level of forwarding reduces number of software interrupts and other stalls like busy wait. Thus , reducing the overall latency in delivering the service.

Formal modeling – Conventional v/s Zero copy

This section mathematically proves the dominance of zero-copy protocol over other mentioned implementations. All latency concepts can be attributed to a single fact of time spent in moving data from kernel to user space and vice versa.

Let tot represents the total time taken to transmit a packet from application to kernel and vice versa. t1 represents time spent in passing data from user to kernel and t2 be the time of other transferoperations. R is the transfer rate which will determine the latency factor. It is governed by the equation as:

:

$$R = \frac{s}{t_{tot}} = \frac{s}{t_1 + t_2}$$

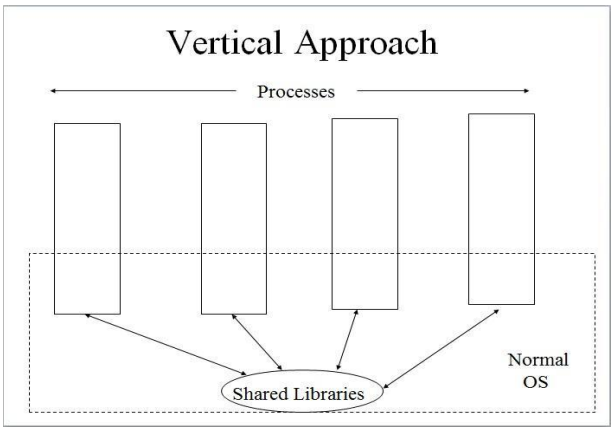
To obtain maximum rate we have to minimize the time spent so,

$$R_{max} = \lim_{t_1 \to 0} \frac{s}{t_1 + t_2} = \frac{s}{t_2}$$

As proved ,rate of transfer is inversely proportional to the time spent in other transfers which is reduced in Zero copy and increased due to extra buffer storage in conventional as depicted in figure 2.

Vertical Partitioning of OS

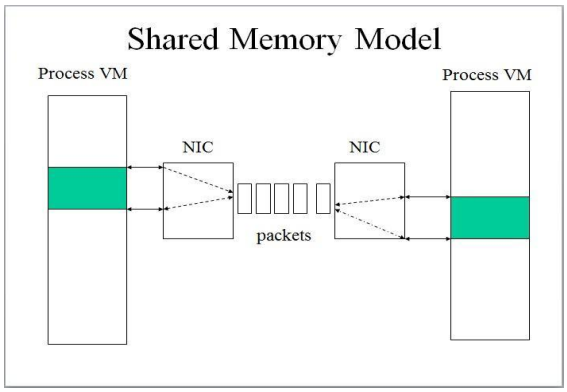
U-Net gave applications an abstract network card so there was less multiplexing of data. Now we go all the way and do more partitioning of OS resources. Even disk devices and file systems are carefully partitioned.



The prime reasons why all the above techniques did not catch up was because the real problem was that increased bandwidth bombarded the OS with interrupts and interrupts rate was phenomenal. Network traffic has become overwhelming for a interrupt driven system.

Shared Memory Model

In shared memory model data transfer is accomplished by writing to memory addresses in the local address space of the process. This data is captured by the local network card and serialized into packets which are transferred over the network to the remote machine which writes the data to remote addresses. A region of the local address space of the process is mapped to an IO region on the card. Standard memory-mapping techniques are used to make that mapping.



In simple words, this means that instead of physically sending it across the network, sending applications can tell receiving applications to get data from memory. This is usually done with a mechanism called Inter Process Communications (IPC), and it eliminates the latency and pitfalls of network connectivity. Deploying a large-scale multi-core machine with shared memory allows complex tasks to be completed more quickly.

Problems with SMM: It's difficult to inform the remote process/node that data is waiting to be read because there are no interrupts involved anymore. The major problem then becomes application synchronization.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

Application Synchronization

Application Synchronization means scheduling the relevant application to receive the data. Application Synchronization can be achieved in SMM through following ways:

Polling:

The receiver keeps polling certain addresses to see if a data transfer has occurred. This is expensive (wasting local CPU) and only relevant if there is a real chance of a data transfer. This technique could be used to provide to provide a form of distributed synchronization - spinning on a remote address.

VM Signaling:

In this technique page is only mapped locally when there is data to be read. If a page with no data is accessed, then a page fault occurs and I am blocked until the owner writes to the page.

Out-of-Band Signaling:

A separate channel is used outside the data transfer region to signal that data has been transferred. For example, writing to a special set of addresses would cause an interrupt to be generated at the remote end. So you would transfer the data by writing to your local address. After you then wrote to a special address associated with that memory region. An interrupt occurs on the other side and the OS works out which buffer you are referring to and wakes up the waiting process.

CONCLUSION

To conclude we can come up with the fact that although zero copy protocol and User-Level Network Interface has trade off like higher initial cost and increased power consumption but higher rate of processing and minimal latency compensates for the loss. However, if the system tends to become overwhelmed by the interrupts caused by network traffic then SMM might be the best technique to cope up.

REFERENCES

[1] U-Net - A User Level Network Interface Infrastructure - http://www.eecs.harvard.edu/~mdw/proj/old/unet/
[2] Implementing Parallel Processing - https://docs.oracle.com/cd/F49540_01/DOC/server.815/a67778/ch2_succ.htm
[3] Introduction to Parallel Programing -- https://computing.llnl.gov/tutorials/parallel_comp/#ModelsShared
[4] Parallel Programming Model - http://en.wikipedia.org/wiki/Parallel_programming_model#Shared_memory
[5] Polling - http://www.cs.umd.edu/class/ sum2003/cmsc311 /Notes/ IO/ polling.html
[6] Interrupts And Polling - http://dis-dpcs.wikispaces. com/6.5.3 +Interrupts +and+Polling
[7] http://en.wikipedia.org/wiki/Low_latency
[8] www.ciena.com/solutions/enterprise/ultra-low-latency-networking/
[9] www.scs.stanford.edu/~rumble/papers/latency_hotos11.pdf
[10] http://insidehpc.com/hpc-basic-training/what-is-hpc/
[11] http://www.hipc.org