ISSN 2047-3338

Performance Improvement of Information Exchange between the Virtual Machines on the Same Physical Host

Nguyen Tram Hong An, Nguyen Tan Cam and Cao Dang Tan

***Abstract*—Virtualization environment provides VMs which are isolated from each other. This isolation helps co-resident VMs use original transport protocols like UDP and TCP for information exchange. Using original protocols does not take advantage of being on the same host of VMs because of unnecessarily overheads. Running VMs are processes on physical host so that inter-VM communication should be inter-process communication to assure the profit of their exchange speed. It is very important to conduct new researches to find out the best solution to improving performance of inter-VM communication. Nowadays, there are many up-to-date mechanisms to solve this problem. However, most of related papers do not provide a general view for developers in term of choosing suitable mechanisms for a specific application with a specific group of data sizes nor any applications with any data sizes when they implement inter-VM communication. This article is focusing on analyzing and comparing main mechanisms of three approaches, which are being considered most now: Shared Memory, Unix Domain Socket and Pipes. The three inter-process communication tools: Unix Domain Socket (UDS), Shared Memory and Pipes are also implemented by authors for analysis and comparison purpose. The results of the experiments provide useful advices for developers when they want to choose a suitable inter-VM communication mechanism for applications depending on different data sizes.**

***Index Terms*— Inter-process Communication, Inter–VM Communication and Performance of Information Exchange**

I. INTRODUCTION

BASED on virtualization technology, Cloud Computing is growing fast and providing different benefits for users [1]. Along with this growth, remained aspects of virtualization are being studied widely. One of these important problems is information exchange between co-resident VMs.

One of the key features that virtualization environment provides is to assure the isolation barrier between VMs on the same host, called co-resident VMs. Co-resident VMs use original transport protocols (such as TCP, UDP etc) to do the information exchange [2]. Original protocols like TCP and UDP do not take advantage of being on the same host of these VMs because physical resources will be wasted by additional overheads such as header encapsulation, routing [3].

In fact, there are many mechanisms were created to improve information exchange performance between any two VMs on the same host by bypassing one or two last layers of TCP/IP protocol stack or bypassing protocol stack completely [3], [16].

Running VMs are processes which run on the same physical host so that information exchange between two co-resident VMs should be as inter-process communication (IPC) [3].

The approach of improving performance of information exchange between VMs on the same physical host is one of the main trends and is being widely studied. The target of these researches is creating new mechanisms to improve performance by reducing additional overheads in data information exchange and effectively utilizing physical resources of the running system.

Mechanisms mainly based on the Shared Memory approach are designed and implemented in different ways [4], [5], [6], [7], [8], [9]. In contrast, UDS and Pipes are mainly used in performance comparison [12], [13], [14]. Generally, all today mechanisms are used or designed based on IPC's concepts.

Up to now, performance of Shared Memory, Pipes and UDS are not directly compared with each other. Besides that, implementation of these IPCs were conducted on a few sizes of data. Previous articles do not provide an all-sided view about performance and features of newly developed mechanisms for developers.

This article implements Shared Memory, Pipes and UDS on different sizes of data and compares, evaluates, summarizes the performance and features of most-considered mechanisms, provides an all-sided view for developers in terms of choosing a good inter-VM mechanism or create a better one.

[1]Nguyen Tram Hong An: Ho Chi Minh City University of Science, Vietnam, (Email: annth2907@gmail.com)

[2]Nguyen Tan Cam: Hoa Sen University, Vietnam, (Email: camnguyentan@gmail.com)

[3]Cao Dang Tan: Ho Chi Minh City University of Science, Vietnam, (Email: tan@hcmus.edu.vn)

II. RELATED WORKS

A. Current State of the Mechanisms

Today, information exchange mechanisms between co-resident VMs belong to the three main approaches: Shared Memory, UDS and Pipes. However, the popularity and usability of Shared Memory is more than those of Pipes and UDS. This section is describing the design and implementation of mechanisms in the Shared Memory approach. How Pipes and UDS are used in the performance comparison is also being described in this section.

Shared Memory:

Mechanisms of the Shared Memory approach create a shared memory segment between sender and receiver. Whenever these VMs want to exchange information with each other, the sender writes data into the shared memory segment and the receiver can read the data from the segment immediately.

Two virtualization environments are usually used in this approach: Xen and KVM. Some mechanisms were implemented successfully on Xen such as Xen Loop [4], XWAY [5], Xen Socket [6] and IDTS [7]. Unlike Xen, KVM is a developing environment so that the numbers of mechanisms based on KVM are not as great as that of Xen: ZIVM [8], Inter Channel [9].

There are two ways to implement shared memory: use existing libraries (such as Inter Channel [9]) or only base on shared memory's concept [4], [5], [6], [7], [8].

Mechanisms of Shared Memory approach are designed and implemented as follows:

- Bypassing Network Protocol Stack completely by creating a shared memory segment between the sender VM and the receiver VM. Mechanisms using this design are XWAY [5], Xen Socket [6] and IDTS [7].
- Beside bypassing Network Protocol Stack, some mechanisms bypass one or two last layers of network protocol stack. This design is used in Xen Loop [4].
- Mechanisms such as XWAY [5] and Xen Socket [6] design their API in a the same way as socket API to help developers use their API easily.
- In case of MMNet [10], when the sender VM wants to communicate with the receiver VM, all Kernel Address Space of the sender VM is mapped to the receiver VM's address space. Next, the sender uses Event Channel, which is provided by Xen, to inform the receiver to read the data from the shared memory segment.
- IDTS [7] and Inter Channel [9] are based on remained aspects of I/O or default communication methods of hypervisor to introduce improvement and remedies.
- Most of Shared Memory mechanisms use two shared memory segments at the same time to speed up data exchange processes. A VM can be both a sender and a receiver at the same time, support bidirectional for data exchanging.

Unix Domain Socket (UDS) and Pipes:

UDS and Pipes are rarely used in practice because their implementation and libraries are not suitable for being applied directly to the virtualization environment (one of these reasons is the security problem which was mentioned in [2]). Besides, performance of UDS and Pipes is lower than that of Shared Memory in specific cases [3]. However, UDS has an API socket which is widely used by developers so that many mechanisms of Shared Memory simulate this UDS API socket to help developers use easily (XWAY [5], Xen Socket [6]).

B. Performance of New Inter-VM Mechanisms

Shared Memory Approach:

Today, there are many mechanisms belonging to Shared Memory approach, this article is comparing the main features of the mechanisms using shared memory to improve the communication throughput of co-resident VMs. Main features of these mechanisms are compared in Table I(a) and I(b).

TABLE I(a):
FEATURES OF XEN SOCKET, XWAY, XENLOOP AND ZIVM

Mechanisms / Features	XenSocket [2], [6]	XWAY [2], [5]	XenLoop [2], [4]	ZIVM [8]
User Transparency	×	✓	✓	✓
Kernel transparency	✓	×	×	✓
Transparent Live Migration	×	×	✓	✓
Location in Software stack	Below socket layer	Below socket layer	Below IP layer	User libs + syscall
Copying overhead	2 copies	2 copies	4 copies	0 copies
Standard prot. support	×	TCP	✓	✓
Autodiscovery and connection setup	×	×	✓	×
Distributed support	×	×	×	✓
Data size	>512KB & ≤100MB	≤ 32KB	≤ 32KB	>512KB & ≤100 MB

- *User transparancy:* User applications and libraries do not need to be rewritten against new APIs and system calls of new mechanisms.

- *Kernel transparency:* Code of Guest OS do not need to be modified and recompiled to run new mechanisms.
- *Transparent VM Live Migration:* Supporting Live Migration for VMs.
- *Location in Software stack:* Location of mechanism modules in software stack.
- *Copying overhead:* The number of copies needed for one time data transmission.
- *Supported protocol:* Protocols which are supported by mechanisms.
- *Autodiscovery and connection setup:* Supporting discover co-resident VMs and establishing connection between two VMs automatically.
- *Distributed support:* Ability to work on Cloud Computing environment.
- *Data size:* Suitable data sizes for mechanisms. This criteria show developers which sizes of data this mechanisms can work effectively. These sizes of data are based on announced figures of mechanisms [4], [5], [6], [7], [8], [9], [10], [11].
  - Sizes of data equal or smaller than 32KB are often used for message exchange such as TCP or UDP packets, instant messages, emails without attachments …
  - Sizes of data which is 512KB ≤ x ≤ 100MB (x is size of data) are usually used in file transfer or internet accessing.

TABLE I(b):
FEATURES OF SOCKET OUTSOURCING, MMNET, INTER CHANNEL, IDTS

Mechanisms / Features	Socket Outsourcing [11]	MMNet [2], [10]	Inter Channel [9]	IDTS [7]
User Transparency	×	✓	✓	✓
Kernel transparency	×	✓	×	×
Transparent Live Migration	✓	×	×	×
Location in Software stack	Socket layer	Below IP layer	User libs + Syscall	User libs + syscall
Copying overhead	2 copies	2 copies	2 copies	2 copies
Standard prot. support	×	✓	✓	×
Autodiscovery and connection setup	×	✓	×	Only Connection setup
Distributed support	×	×	×	×
Data size	≤ 32KB	≤ 32KB	≤ 32KB	≤ 32KB

Feature-wise, ZIVM is the best mechanism which is providing most of features for developers. However, ZIVM still has disadvantage in security ensuring for all VMs which are sharing the same shared memory segment. Xen Loop and MMNet are two runner-up mechanisms which also support many features but Xen Loop still has to improve copy overheads and MMNet has to ensure the memory isolation between VMs.

Regarding performance, based on figures from announced papers, new mechanisms have a higher bandwidth than that of UDS when data sizes are small (<1KB) but when the data sizes become bigger, UDS will have a higher bandwidth (XWAY [5]). In case of Xen Socket, Xen Socket bandwidth is lower than the UDS bandwidth when the data size smaller than 16KB but when data size is bigger than 16KB, the bandwidth of Xen Socket become better [6]. Generally, all new mechanisms have better bandwidth than original TCP and UDP.

In addition to comparing features of new mechanisms, this article is also summarizing all the performance figures in the announced papers of new mechanisms. The new mechanisms are compared with original TCP and KVM default.

TABLE II:
BANDWIDTH OF SHARED MEMORY MECHANISMS COMPARE WITH ORIGINAL TCP

	Bandwidth (Mbps)	TCP Bandwidth (Mbps)	Comparison Result
XenSocket (<16KB) [6]	9295	130	71.5 times.
XenSocket (≥16KB) [6]	6535	141	46.3 times.
XWAY [5]	7800	2000	3.9 times.
IDTS [7]	6600	4000	1.65 times.

TABLE III:
BANDWIDTH OF SHARED MEMORY MECHANISMS COMPARE WITH KVM DEFAULT

Mechanisms	Comparion with KVM Default
ZIVM	4.015 times
XenLoop	14.454 times
InterChannel	45.5 times
MMNet	9.855 times

Compared with original transport protocol TCP and hypervisor default mechanisms, the target of improving performance can be seen as being done very well by all new

mechanisms. They are faster than the original TCP and KVM default.

UDS and Pipes Approach:

In all papers which mention performance of UDS and Pipes. UDS and Pipes are compared with original protocols such as TCP and UDP. The results show that UDS and Pipes have a higher bandwidth than bandwidths of TCP and UDP. When compared with TCP, UDS has an outstanding bandwidth than that of TCP [12], [13]. The difference in bandwidth between TCP and UDS still high even though TCP is working in an ideal condition [14]. When compared with TCP and UDP, Pipes also has a higher bandwidth than that of TCP and that of UDP but the difference in their bandwidth between Pipes and TCP, UDP is lower than when we compare the difference among the bandwidths of TCP, UDP with UDS [7]. In all papers, UDS and Pipes are still used for performance comparison, not for wide implementation.

III. EXPERIMENTAL RESULTS

Three IPCs: Pipes, UDS and Shared Memory had been implemented to provide an all-sided view of performance and concordance of these IPCs with three groups of data sizes.

A. Test Model

The performance of these three IPC was evaluated on a machine equipped with Intel Pentium Dual Core E2200 of 2.2 Ghz, 2GB of RAM, 200GB of HDD and Ubuntu 12.04 LTS. All test programs were written in C.

In the testing model, there are a sender process and a receiver process. The sender sends an amount *x* Bytes of data to the receiver (value of *x* can be found in Table IV). When the receiver received all *x* Bytes of data, the progress of communication completes. The result of a data size of an IPC is an average result of 20 times of running the testing program.

All tested data sizes which are used in the experiment are based on common data sizes in usual information exchange and in the papers of all shared memory mechanisms.

TABLE IV:
DATA SIZES USED IN EXPERIMENT

Group of data sizes	Data Sizes	Remark
Small Data Sizes	16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB, 2MB.	Common data sizes when transferring small files, sending emails without attachment…
Average Data Sizes	100MB, 200MB, 300MB, 400MB, 500MB, 600MB.	Common data sizes when sending a CD or ISO file, sending video files…
Big Data Sizes	1GB, 2GB, 3GB, 4GB.	Common data sizes when sending DVD, sending ISO file, streaming video HD…

Bandwidth of UDS, Pipes and Shared Memory are shown in Fig. 1.

The graph shows that there are three ranges correlative with three groups of data sizes. The small data sizes group corresponds with the changing range. In this range, the bandwidth of all three IPCs has a wide range of fluctuations. The difference in bandwidth of a data size of the three IPCs is large. In this range, a sudden fall or rise in the bandwidth of the three IPCs happens frequently. The average data sizes group corresponds with the decreasing range. The bandwidth of all IPCs decreases when data sizes become bigger. The level of decrease can be small (in case of UDS and Pipes) or linear on the whole range (in case of Shared Memory). The difference among the three IPCs of a data size is smaller than in that of the changing range. The stabling range corresponds with the big data sizes group. In this range, bandwidth of all IPCs is at about 300 Mbps. Bandwidth does not decrease when size of data increases, the difference in bandwidth at a data size is not large.

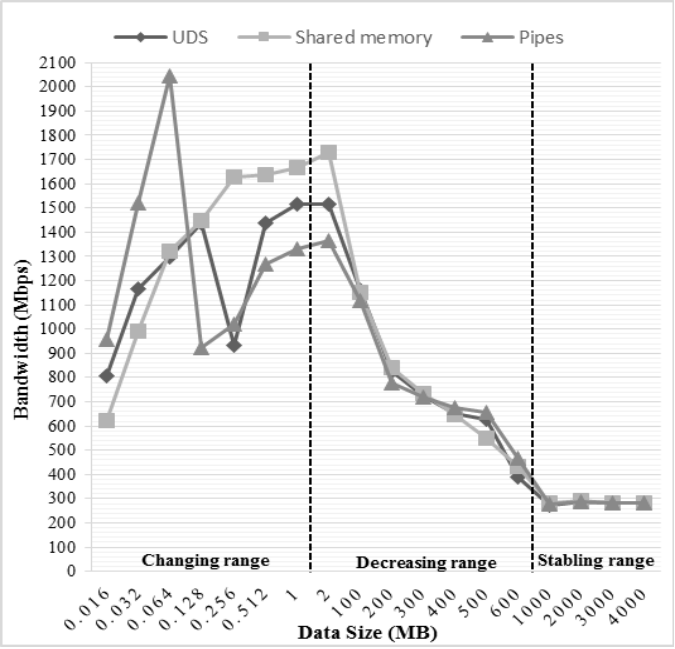

Fig. 1: Bandwidth of UDS, Shared Memory and Pipes

The results of experiment are shown as follows in Table V.

Based on the experimental results, Table VI shows the matching level of each IPC on the three data sizes groups.

Because the difference among the three IPCs in the average data sizes group is small, if developers need an IPC that can be used for many data sizes, then Shared Memory is the best choice, the second choice is UDS. With applications having data sizes in the small group data sizes such as DNS request, DNS response, music file transfer, text file, email without attachment, database query… Shared Memory will be the best selection. For all applications having data sizes in the average data sizes group such as videos file transfer, CD, TFTP …, developers should use Pipes. Shared Memory is also

a good choice in case of big data sizes such as FTP, streaming HD videos and transferring DVDs.

Some disadvantages of these IPCs:

- *UDS:* The size of UDS buffer is not as big as Shared Memory buffer, all the connection establishment and connection management manipulation are more complex than the others.
- *Shared Memory:* All the synchronization and mutual exclusion manipulation, which are used when transferring data, decrease the advantages of having a big buffer.
- *Pipes:* Buffer which is small-sized leads to increase in the number of reading and writing times.

TABLE V
BANDWIDTH OF SHARED MEMORY, PIPES AND UDS

Sizes (MB)	UDS (Mbps)	Shared Memory (Mbps)	Pipes (Mbps)
0.016	809.187820	620.791504	956.030671
0.032	1164.376361	989.435395	1521.859206
0.064	1297.750579	1320.761495	2046.689259
0.128	1438.782221	1449.634922	922.175598
0.256	930.850198	1629.027372	1018.145735
0.512	1439.323902	1639.157721	1266.550103
1	1514.535314	1666.517742	1332.025816
2	1514.199543	1727.163538	1365.058813
100	1162.956768	1151.480499	1118.280336
200	826.913941	838.702989	775.814498
300	725.788015	735.704989	718.129286
400	653.629809	647.515371	676.763066
500	626.976399	551.312155	654.772608
600	390.708846	431.000848	466.886044
1000	272.213147	281.198555	276.734680
2000	289.001141	290.171336	287.532941
3000	282.014764	284.038015	283.381890
4000	283.249123	283.283408	282.230225

TABLE VI
MATCHING LEVEL OF THREE IPCS IN EACH DATA SIZES GROUP

Data sizes / Matching Level	Small Data Sizes	Average Data Sizes	Big Data Sizes
I	Shared Memory	Pipes	Shared Memory
II	UDS	UDS	UDS, Pipes
III	Pipes	Shared Memory	-

IV. FUTURE WORKS

Extending the Research to Other IPCs:

The experimental results show that Shared Memory has the best performance and can be used for a wide range of data sizes. However, Pipes and UDS have a little difference in bandwidth with Shared Memory in specific cases. If UDS and Pipes are studied and developed deeply, they may show better performance.

Supporting Live Migration:

Supporting Live Migration can affect performance of new mechanisms because of connection status controlling. Supporting Live Migration without affecting their performance is a necessary research.

Supporting Real Time Protocol:

Hypervisors lack knowledge of real time applications running on VMs, so they cannot guarantee real-time for those applications. Today, there are some solutions such as AICT [2] to help hypervisors support real time protocols. However, it is still possible that processes running real time protocols within VMs cannot obtain a priority in CPU usage. For real time guarantees, the hypervisor's CPU scheduler and VMs CPU scheduler must coordinate with each other to meet real time requirements.

V. CONCLUSION

This article has analyzed and compared the most considered mechanisms of the three approaches: Shared Memory, UDS and Pipes. Based on this research, Shared Memory was found as the most popular approach while UDS and Pipes were mainly used for performance comparison. The article has also proposed the list of main useful features of the mechanisms in the Shared Memory approach. From the list, ZIVM, which is the mechanism of the Shared Memory approach, was selected as the best existing mechanism for developers. Three IPCs are implemented to provide an all-sided view about bandwidth for developers when they want to choose a suitable IPC to implement new inter-VM communication. From the experimental results, Shared Memory was proposed to be used for all data sizes. Finally, some open future researches are mentioned to improve inter – VM communication problems.

REFERENCES

[1] Sun Microsystems, *A guide to getting started with cloud computing*, 2009.

[2] Jian Wang, *Survey of State-of-the-art in Inter-VM communication Mechanisms*, Research Proficiency Report – Binghamton University, 2009.

[3] Nguyen Tram Hong An, Nguyen Tan Cam, Cao Dang Tan, *Research about performance improvement mechanisms for Inter – VMs Communication*, undergaduate thesis, 2013.

[4] Jian Wang, Kwame Lante Wright, Kartik Gopalan 2008: *XenLoop: A transparent high performance Inter-VM network loopback*, Proceedings of the 17th International Symposium on

High Performance Distributed Computing (HPDC), 2008, p. 109–118.

[5] Kangho Kim, Cheiyol Kim, Sung-In Jung, Hyun-Sup Shin, Jin-Soo Kim: *Inter-domain socket communications supporting high performance and full binary compatibility on Xen*, Proceedings of the fourth ACM, SIGPLAN/SIGOPS, 2008, Page 11-20.

[6] Xiaolan Zhang, Suzanne McIntosh, Pankaj Rohatgi, John Linwood Griffin, *XenSocket: A hight-throughput interdomain transport for Virtual Machine*, In Proceedings of Middleware, 2007.

[7] Dingding Li, Hai Jin, Yingzhe Shao, Xiaofei Liao, *A high-efficient Inter-Domain data transferring system for Virtual Machine*, ICUIMC '09, Proceedings of The 3rd International Conference on Ubiquitous Information Management and Communication, New York, 2009, Pages 385-390.

[8] Hamid Reza Mohebbi, Omid Kashefi, Moshen Sharifi, *ZIVM: A Zero-Copy Inter-VM communication mechanism for Cloud Computing*, Computer and Information Science, Volume 4, Number 6, 2011, Pages 18-27.

[9] Shengge Ding, Ruhui Ma, Alei Liang, Haibing Guan: *Optimization for Inter-VMs Network Performance.*

[10] Prashanth Radhakrishnan, Kiraan Srinivasan, *MMNet. An efficient Inter-VM communication mechanism*, Proceedings of Xen Summit, 2008.

[11] Hideki Eikaru, Yasushi Shinjo, Calton Pu, Younggyun Koh, Kazuhiko Kato, *Fast networking with Socket-outsourcing in Hosted Virtual Machine Environments*, Proceedings of the 2009 ACM symposium on Applied Computing, New York USA, 2009, Pages 310 – 317.

[12] Kwame Wright, Kartik Gopalan, Hui Kang, *Performance analysis of various mechanisms for Inter-process Communication*.

[13] Manoj Nambiar, Sricharan Smudrala, Sundar Narayanan, *Experiences with UNIX IPC for low latency messaging solutions*, The Proceedings of the Computer Measurement Group's 2009 Internation Conference, United States of America,2009.

[14] W. Richard Stevens, *TCP/IP Illustrated Volume 3: TCP for transactions, HTTP, NNTP and the UNIX domain protocols*, 13th printing, Addison Wesley, USA, 2003.

[15] Jonathan S. Shapiro, David J. Farber, Jonathan M. Smith, *The measured performance of a Fast Local IPC*, IWOOOS '96, Proceedings of the 5th International Workshop on Object Orientation in Operating System, Washington, 1996, Page 89.

[16] Nguyen Tan Cam, Huynh Van Tho Nguyen Hoang Sang, Cao Dang Tan, *nFTP: An Approach to Improve Performance of FTP Protocol on The Virtual Network Environment in the same physical host*, The International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE 2012), Bridgeport, Connecticut, USA, December 7-9, 2012.