Представлено створення та реалізацію механізму спрощеної комунікації (МСК) для віртуального кластера в високопродуктивному cloud'і, який є бінарно сумісним для додатків зі стандартним сокетним інтерфейсом. МСК реалізований на базі Xen 3.2 та ядра Linux 2.6.18. і представляє процес, подібний для сокетів UNIX DOMAIN

Ключові слова: механізм спрощеної комунікації, високопродуктивна обробка даних, бінарна сумісність, кластер

Представлено создание и реализация механизма упрощенной коммуникации (МУК) для виртуального кластера в высокопроизводительном cloud'е, который является бинарно совместимым для приложений со стандартным сокетным интерфейсом. МСК реализован на базе Xen 3.2 и ядра Linux 2.6.18, представляющий процесс, подобный как и для сокетов UNIX DOMAIN

Ключевые слова: механизм упрощенной коммуникации высокопроизводительная обработка данных, бинарная совместимость, кластер

UDC 004.451
DOI: 10.15587/1729-4061.2017.98896

IMPLEMENTATION OF THE SIMPLIFIED COMMUNICATION MECHANISM IN THE CLOUD OF HIGH PERFORMANCE COMPUTATIONS

V. Melnyk
PhD, Associate Professor*
E-mail: melnyk_v_m@yahoo.com
N. Bahnyuk
PhD, Associate Professor*
E-mail: bagnjuk_n@rambler.ru
K. Melnyk
PhD, Associate Professor*
E-mail: ekaterinamelnik@gmail.com
O. Zhyharevych
Assistant*
E-mail: oz_lutsk@mail.ru
N. Panasyuk
PhD, Associate Professor**
E-mail: Natalyalutsk85@hotmail.com
*Department of Computer Engineering***
Department of Computer Technology*
***Lutsk National Technical University
Lvivska str., 75, Lutsk, Ukraine, 43018

1. Introduction

At present, the development of high-performance computations (HPC) has gained considerable traction in different fields of its application. However, as the obtained scientific data confirm [1–4], it still suffers from many technical shortcomings: low throughput and network performance, both externally and within the cluster of HPC, as well as increased time delay. Addressing these deficiencies was carried out in the implementation of various technical and programming mechanisms. Thus, in order to improve the inter-domain communication, authors of article [1] proposed the com-sockets, which contributed to improving the network throughput and reducing the time of message delay. In paper [2], authors, in order to solve a similar problem of data exchange between clouds in the scientific medicine, employed java-sockets with a single copy of the protocol when reusing the fulfillment threads and special libraries of quick messaging, which contributed to the improvement of network characteristics in data exchange. Other works investigated the issue of the influence of high-performance sockets on the intensity of data processing and the implementation of analyzer for detecting the interaction between java- and other applications and the network. All these studies emphasized, for HPC and network interaction, the network costs, use of memory, the scalability and reliability of data transfer. However, one of the most relevant issues was improving the performance efficiency (throughput) of the network on its different sections.

Today, from the point of view of end users or system administrators, sometimes there is a question how to bypass or "ruin" already working traditional models of applying HPC, with the purpose of understanding and creating more flexible and more efficient management and the use of known resources. Their implementation for HPC is required by increasingly sophisticated needs of practical use, and, at the same time, by the growing interest of scientists to address this range of issues.

There is also growing interest to the high-performance cloud computation and the systems, which are quite actively implemented in different areas of human activity. Articles [3, 4] emphasize the issue of improving the network performance, delivery and fast data processing to increase the working ability of applications in science and medicine. In order to solve the above problems, more and more researchers recommend and choose cloud-computing as a new high-speed tool for data management. These papers pay attention to the effective use of HPC resources. All this range of research is aimed at making this communication link highly effective between the infrastructures of HPC and their applications.

Constructing the HPC of clouds that employ traditional HPC resources not only opens up new benefits associated with solving the above-mentioned problems but also brings with it a lot of interesting studies and tasks. One of these tasks is to reduce the load on the network of a virtual cluster in HPC cloud.

2. Literature review and problem statement

By the research conducted and the results obtained, the authors of articles [5–7] clearly proved that the Linux Xen guest domain demonstrates considerably lower network performance than the native Linux. This problem is addressed in paper [8] under conditions when the application on a virtual machine communicates with another virtual machine on another physical machine. Authors of the aforementioned studies have found a marked decrease in productivity by a factor of 2–3 for the input load. A similar reduction by a factor of 5 was observed also for the output load on the network. They also prove that the TCP/IP protocol, developed for the comprehensive Internet environment, is not applicable for the environment of a virtual cluster.

Present work set the task of implementation and discussion of two types of examples of communication between virtual machines in a virtual cluster (Fig. 1). It was necessary to implement an inter-domain communication as communication between virtual machines deployed on a single physical machine. Fig. 1 shows this communication between virtual machines 1 and 2 and between virtual machines 3 and 4. Extra-domain communication link should be implemented as a link between virtual machines deployed on different physical machines. This link is shown between virtual machines 1 and 3 or between virtual machines 2 and 4 (Fig. 1). In order to solve the above-specified problem, we also propose here a mechanism of simplified communication (SCM) for network performance in the environment of a virtual cluster. We also set the task to verify complete binary compatibility with applications that use the interface of standard sockets.

First of all, we would like to introduce a simplified communication protocol, adapted to the environment of network of a virtual cluster, as opposed to TCP/IP, in order to improve its performance efficiency. This protocol is characterized by not only a low time delay and high throughput of communication network, but also a full dual compatibility for the applications written using the interface of standard sockets. There is no need to rewrite or recompile acting applications to ensure productivity of the protocol of simplified extra-domain communication. It should be clearly noted that this protocol is used solely for the internal non-extra-domain communication, inside a virtual cluster. However, any virtual machine in a virtual cluster that is communicating with a machine (physical or virtual) outside the virtual cluster still requires the TCP/IP protocol.

At the same time, we also set the task to integrate the XenLoop [9] into machine with simplified communication with the aim of improving the performance of an *inter-domain* network. In order to obtain a better performance of communication in an inter-domain network, the XenLoop intercepts outgoing network packets before the network level. It then detects those of them that are meant for virtual machines, united by a high-speed channel of distributed memory and, at the same time, bypasses the interface of a virtualized network.

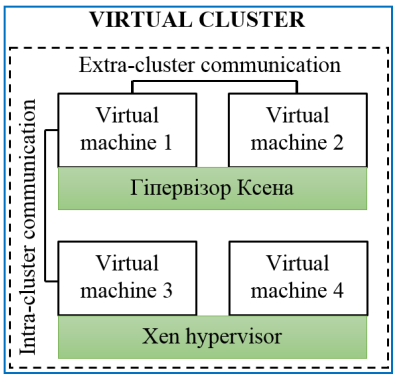

Fig. 1. Example of communication between virtual machines in a virtual cluster

In the course of analysis of the scientific studies, we discovered materials that describe the reduction of overhead expenses of network input/output virtualization, which affects the performance of intensive network applications. One of the most affordable ways is to launch a high-speed communication channel between the domains. In particular, in article [10], high performance is achieved through the circumvention of TCP/IP-stacks, bypassing the costs of turning pages and providing a direct boosted communication channel between the domains in the same machine. High-performance cross-domain communication mechanism, referred to as XWAY, intercepts related calls between the INET- and TCP-level. It has a switch in order to determine whether the destination domain is located in the same physical machine, or in the other. If it is in the same machine, the switch tries to create an XWAY Canal and link it to the XWAY-socket. In other words, from the level of switch, it sends the INET-request to the TCP level. External communication in the network is not affected as the XWAY-channle does not exist for it. In addition, XWAY supports full binary compatibility for applications written for the sockets with a standard interface, however, it is meant only for the intra-domain communication. Many network intensive applications in virtual clusters, such as, for example, those designed for high-performance parallel computation, use an intra-domain example of hybrid communication, rather than extra-domain communication. Thus, according to the Amdahl's law [11], the optimization of communication only between the domains will not improve significantly performance of the system as a whole.

The mechanism of XenLoop [9] is an open source project under the GPL-law. Similar to XWAY, the XenLoop mechanism is aimed at obtaining better communication performance in the environment of virtual machines inside a domain. It is sensitive to the problem of transparency at the level of the user, and still achieves high productivity of communication between virtual machines – co-residents. It checks incoming network packets on the network level and tracks packets, sent for the virtual machines deployed on a single physical machine. This is done via a high-speed channel of distributed memory between the given virtual machines, bypassing a virtualized network interface. Using the XenLoop mechanism, a virtual machine-guest can transparently migrate between machines without violating the current network communication and flawlessly switch between the usual network path and the XenLoop channel.

The entire mechanism of XenLoop is realized in the form of Linux kernel modules in Xen-domain "0" and guest

domains so that any of the virtual machines in the domain is capable of loading the XenLoop dynamically in operating mode. The assessment conducted through several non-modifying tests proved that XenLoop can reduce the time delay of the full round-trip between virtual machines by 5 times and increase the throughput by 6 times [9]. However, the drawback is that by using the presented XenLoop mechanism, it is possible to optimize communication performance only inside the domain, similar to using the XWAY. As XenLoop on the network level intercepts outgoing network packets that arrive, it testifies to the additional utilization of CPU resources for the TCP/IP processing. Fig. 2 shows architecture of the XenLoop mechanism in the domain of the user.

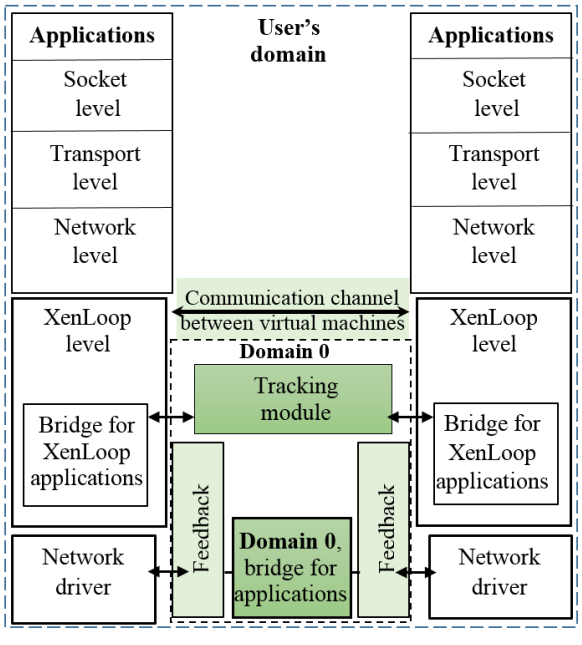

Fig. 2. Architecture of the XenLoop mechanism

Another approach to optimize network performance, proposed in article [12], implies the monitoring of packet transfer between virtual machines. For this purpose, models of virtualization for the input/output devices in the environment of virtual machines require the introduction of monitoring of virtual machines (MVM) and/or preferred virtual machines. Monitoring is conducted for each performed input/output operation, which can cause difficulties in the network performance for systems with high input/output requirements. These include systems that are equipped with high speed modem channels between them, such as, for example, InfiniBand. Monitoring the exchange of input/output between virtual machines expands the passage concept for the operating system, which comes from the notion of user-level communication. This allows time-critical input/output operations to be enabled directly on the guest virtual machines without monitoring between virtual machines or preferred virtual machines. By using the embedded intelligence in modem high speed network interfaces, it is possible to use the monitoring of exchange between virtual machines. This can greatly improve input/output performance and the communication efficiency between them without compromising security or separation. Fig. 3 shows architecture of monitoring the passage between virtual machines.

However, monitoring the passing between virtual machines can be implemented only in an environment whose networking equipment supports the passing of an operating system. And only those applications that are written in the interface of passing the operating system can be satisfied by the performance of monitoring of run between virtual machines.

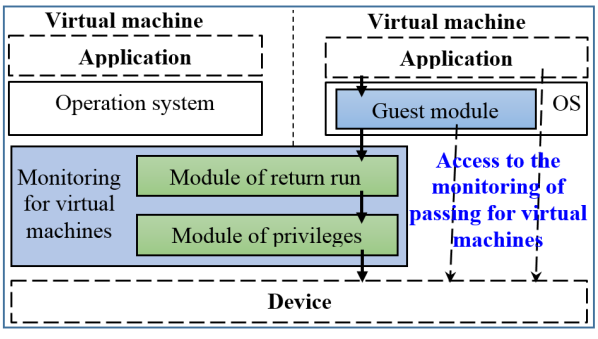

Fig. 3. Architecture of monitoring the passing between virtual machines

Remote memory access protocol (RMAP) is a specifically designed window-like reliable datagram protocol of simplified communication (SCP), designed to implement a remote access to the memory in the same subnet. This is exactly the module described in articles [13, 14]. It is a fully transparent distributed system in the Xen environment for virtual machines, which coordinates the use of wide-cluster memory resources to support its large volume and intensive loading for the input/output. In order to effectively transmit a memory page, RMAP removes such functions of TCP as stream-byte abstraction in the order of delivery, control of overloading and functionality of IP-routing in the system of a single subnet.

Thus RMAP bypasses the stack of TCP/IP protocols and communicates directly with the driver of a network device. RMAP focuses on transmitting the memory pages in such a way so that ordinary socket programs may not obtain performance improvement from it.

The project was realized on Xen 3.2 with Linux kernel 2.6.18. Before we show the design, let us present a brief Xen-review and a review of the input/output architecture for a Xen-network (Fig. 4).

Xen is a virtual x86 machine monitor, which makes it possible for the multidirected operating systems to distribute common hardware equipment among themselves. This allocation is done under a safe and resource-efficient mode, but without any obstacle for the work or functioning of the network. This is achieved by providing an idealized abstraction of virtual machines according to which operating systems such as Linux, BSD, and Windows can be deployed with minimal effort. When using Xen, several operating systems can run on one machine at the same time. Each operating system is referred to as a guest domain, while there is one privileged hosting domain among them to control software at the level of applications, which is referred to as domain 0.

In such Xen system, there are two types of virtualization. Pair-virtualization represents the abstraction of hardware equipment. It is similar, but not identical, to the modification of subordinate physical hardware for the guest operating systems. That is why certain operating systems, such as Windows, cannot function under this mode. Full Xen-virtualization provides complete abstraction of subordinate hardware equipment that requires a processor support.

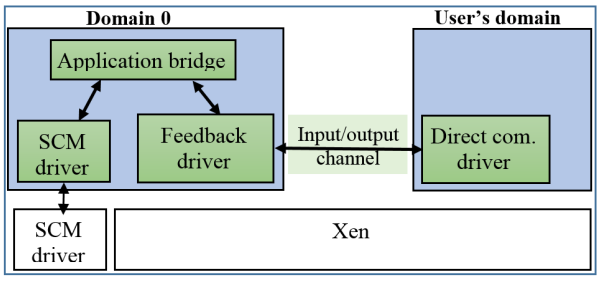

Fig. 4. Input/output architecture of a Xen-network

All network communications under Xen are executed through the use of a virtual network interface. In the domain of the user (Fig. 5), a *network input driver* (front driver) must operate, and in domain 0 – a *network output driver* (feedback driver). Each interface of a *network input* in the guest domain is connected to the corresponding interface of feedback or a *network source end* in domain 0. The drivers of the source end are connected to a physical network card through a software bridge. Network applications in the guest operating systems send data to the input driver. The input driver (front driver) sends a message to the source side (reverse) driver with a copy of the data or scrolling the pages.

As evidenced by the scientific data we already analyzed, one can conclude that up to now reducing the additional network costs particularly in a virtual cluster has been addressed only in separate scientific studies. A question of improving the productivity of operation of the applications for HPC in the cloud of virtual machines has not been investigated up until now and it remains a topical issue. At the same time, still unresolved are the tasks arising from the research itself, such as the introduction of effective mechanisms of communication and their impact to improve the operational performance of intra-cluster and extra-cluster sections of a network, as well as the binary compatibility of applications.

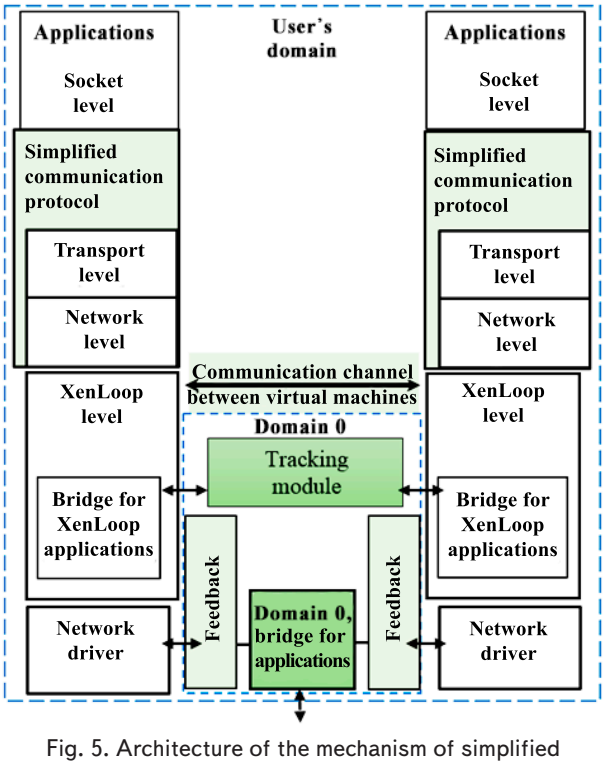

Fig. 5. Architecture of the mechanism of simplified communication (SCM)

3. The aim and tasks of the study

The aim of present work is to represent a mechanism of simplified (light) communication (SCM) for a virtual cluster in the cloud of HPC that supports a binary compatibility for the applications written using a standard socket interface.

To achieve the set aim, the following tasks had to be solved:

– to design a simplified mechanism of communication (SCM) and confirm the improvement of network performance in a virtual cluster of HPC. SCM should use the proposed protocol for simplified communication and support compatibility with network applications;

– in order to improve the performance of intra-domain communication in the proposed system, to integrate XenLoop to intercept packets at the level below that of the network. To examine the throughput between two virtual machines using SCP, it is required to apply a reference test with a set of NAS-tags [20] and make sure that a packet should be sent through the FIFO channel, launched by XenLoop;

– to prove experimentally that the throughput of inter-domain communication improves for SCP and the protocol of remote memory access (RMAP).

4. Design and implementation of the system

Design of the system.

SCM is composed of two modules. The first one is a *simplified communication protocol module*, which is responsible for the extra-domain communication relative to a virtual cluster. The second one is a *XenLoop module*, which launches high-speed channel between virtual machines with the aim of improving the intra-domain communication in a virtual cluster.

A simplified communication protocol module (SCP) is the main one and contains three sub modules. The first sub-module is a section of the kernel. In order to intercept network messages from the applications of top-level, while maintaining the programming interfaces stable [15, 16], it is necessary to make some changes in the Linux kernel of the guest operating systems. The second sub-module is a simplified communication protocol (SCP). As is known, a TCP/IP protocol consists of addressing, multiplexing, control over connections, control of passing, overload prevention, packet fragmentation/recombination, etc. It operates well in a complex network environment, such as the Internet. However, in such environment as a virtual cluster for the clouds of HPC, where network infrastructure has much higher productivity than the Internet, the functioning of the TCP/IP protocol becomes complicated for processors.

These circumstances gave rise to the idea of developing SCP, which, instead of the TCP/IP protocol, operates as a Linux kernel boot module in a guest operating system in compliance with extra-domain communication. It includes the initialization of the protocol, connection set-up, buffering of data, algorithm of transfer and disconnection. Since the Linux kernel of a guest virtual machine was corrected and improved, the proposed protocol can intercept requests when the top-level applications trigger the socket function for setting up a TCP connection. Similar to TCP, in order to ensure reliable data transmission, SCM also employs a timeout and a mechanism for repeat-

ed data transfer. Before sending a packet, a simplified communication protocol computes a data checksum and puts the result in the packet.

When a package of SCP is received, SCM first checks its checksum. If the checksum does not match, SCM would send a message in order to inform the sender of a second dispatch of this packet. In order to improve the throughput of network connection, SCP will receive multiple packets before sending its confirmation. The number of sent packets before a confirmation depends on the actual network environment. Several experimental runs were executed to identify and confirm to a certain extent the most accurate value. If the sender does not receive a confirmation packet after a certain set time, the timeout feature is enabled. The sender will resend the unconfirmed data. By employing the SCP transfer algorithm, data can be transferred effectively and reliably to a remote machine. Therefore, the given protocol simplifies the routing, prevents overloading and slow-start of the TCP protocol, as well as implements an efficient and reliable transmission of datagrams and increases the productivity of extra-domain communication.

The third sub-module, a SCP module, is the module (similar to the *daemon* module), which is in the user's space of the guest domain. At the stage of setting up a connection with the help of SCP between two virtual machines, there must be an exchange of some information. In this case, a TCP channel is used to transmit such information as the channel number, MAC-addresses, and after that, while the connection is being set up, it is instantly destroyed. *Daemon* listens to a certain TCP port and the inbound requests to connect with SCP to exchange the necessary information over the TCP channel. Fig. 6 shows the sequence of setting up a connection when SCP is engaged in this operation.

The following is a description of the stages of establishing a connection in the system with a simplified communication protocol (SCP).

1. An application from a virtual machine triggers the socket function to set up a TCP connection.

2. An add-on to the Linux kernel intercepts triggering such a function and transmits a specific task for the function of determining to identify whether the location of the virtual machine is another physical machine.

3. The determining function returns a certain result.

4. If the source and the destination of virtual machine are deployed on different physical machines, the function of setting up a connection with the help of SCP with the isolation of the required data structures will be enabled.

5. The function of setting up a connection through SCP returns the appropriate result.

SCP is aimed at optimizing the performance of extra-domain communication network in a virtual cluster. In order to obtain a better performance for the intra-domain communication, we integrated the *XenLoop module* into the proposed SCM, which intercepts outgoing network packets below the network level.

If the destination of the packet relates to the coresident virtual machine, it bypasses the virtual network interface and is transmitted via a high-speed channel of distributed memory between virtual machines. XenLoop works as a Linux kernel module and provides full compatibility for the top-level applications. It can be typically downloaded at the same time when the system is under working condition. This module contains a bridge in the form of specialized software for the guest machine, which is able to intercept each outgoing package from the network level. It is also used to check the title of the packet to determine the place of its destination. By using the information provided by the recognition module, the XenLoop module can identify the packets that belong to intra-domain communication. Every time two guest virtual machines within a single physical machine establish active network traffic, they dynamically create a bidirectional data-transmission channel between themselves using the "handshaking" protocol on both sides. Packets that are sent to a coresident virtual machine run through a high-speed FIFO channel between virtual machines. Fig. 7 shows in detail a diagram of the sequence of intra-domain communication.

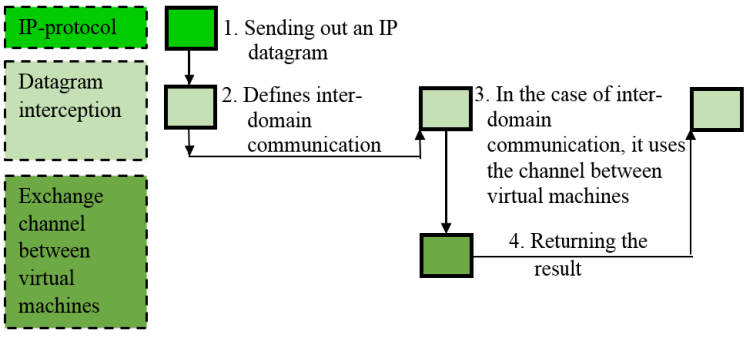

Fig. 7. Sequence diagram of intra-domain communication

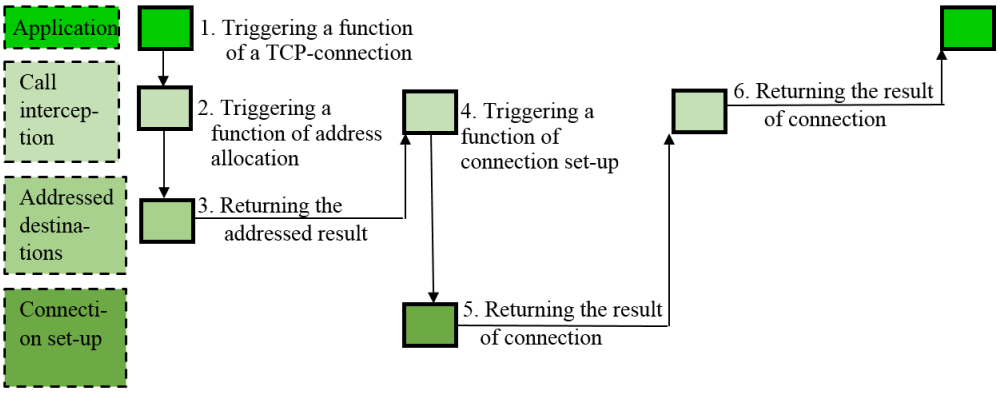

Fig. 6. Sequence diagram of setting up a connection in the presence of SCP

1. The IP protocol sends an IP datagram, which is intercepted by the function of datagram interception.

2. The function of datagram interception, using the information provided by a tracking message module, determines whether its transfer relates to the intra-domain communication.

3. If the datagram belongs to the communication inside the domains, then it will be sent and through the channel between virtual machines created by XenLoop.

4. Returning the appropriate result.

Implementation of the system.

We shall next focus on the details of implementing the module of SCP, since other remaining modules for SCM are described in detail in article [15]. Similar to XWAY, a SCP (communication) module consists of three basic modules: an application to the Linux kernel, loading module for the kernel and the level of using *daemon*. The application to the kernel is used to implement the binary compatibility with socket applications and collaborates with the loading module for kernel in the process of transferring the data using SCP. The loading module for the kernel includes SCP while *daemon* at the user's level helps establish a connection with this protocol.

Application of the kernel mostly modifies two Linux kernel C-structures: *inet_stream_ops* i *tcpyrot*. They both contain pointers that indicate the functions that implement basic operations of TCP network contingent events, such as connection, receiving, sending, and receiving and so on. These pointers were replaced with the above functions. There is always the assertion in each function that determines whether the current operation should be processed by SCP or by the original function of the kernel. If it is a simplified communication, then it will be processed by SCP, otherwise it will be processed by the TCP/IP original protocol.

Define C-structure called as *lp_sock* in the application to the kernel, which consists of the usual structure *TCP_sock* and a pointer that refers to the structure named *IP_ring*. The structure *IP_ring* contains many data elements that are needed for the simplified communication, such as information about the address of the other end of a message, buffers for sending and receiving, channel number and information on the status of communication. We modified the field '*.objsize*' of the structure *tcp_prot* with field size *lp_sock* instead of *tcp_sock*. At the stage of creating a TCP-socket, field *lp_sock* will be created instead of *tcp_sock*. The application to the kernel also includes the other two C-files called *lproto.c* and *lproto.h*. The file *lproto.c* includes mainly determining the structures *lp_sock* and *lp_ring*. The file *lproto.c* includes the functions that are defined by the fields of two modified structures. The kernel boot module is responsible for establishing the connection, data transmission and shutdown of connection. With the help of *daemon*, at the user's level, the connection between two kernel modules is set up. They communicate with each other and exchange data through the protocol of bilateral "handshake". First, two nodes equal in level share MAC-addresses, channel number, IP and information about the ports. Each module must also select own structures *lp_ring* for the pointer to the structure *lp_sock* of the *p_desc* member. They use information received from one another in order to initialize the member-fields of the structure *lp_ring*. After that, the address *lp_ring* is assigned to the *p_desc* member of appropriate structure *lp_sock*.

The user-level *daemon* that helps establish a simplified connection is a socket software. It listens to the TCP port and detects when SCP is triggered. If SCP receives a request for a connection, it first checks the target machine whether there is a virtual machine on another physical machine. If it is required to employ SCP to send and receive data in the communication, then this protocol first launches a connection through its own TCP channel to the user-level *daemon* on the target virtual machine. After the user-level *daemon* accepts a connection request, it will prescribe a file descriptor that represents this connection to the device of a virtual character called *lp_cdev* and implemented in SCM. Due to this device of virtual character, the file descriptor can be transferred to the loading module in kernel space. After that, these two kernel modules may share information necessary for setting up a SCP connection via own TCP channel.

Each datagram of SCP contains a set of fields in a specific order so that the reader knows how to decipher and read the stream of data obtained. The title of the simplified communication protocol datagram contains eight fields. The format of a datagram is given in the form of Fig. 8.

Checksum	**Sequence number**	**Initial index**	**Length of useful part of a datagram**
Serial number of confirmation	Index number of confirmation	Type	Channel number
Useful part of a datagram			

Fig. 8. Format of a datagram for SCP

Fig. 8 shows:

– Checksum – allows SCP to detect datagrams with damaged headers and useful part, and then to discard them.

– Sequence number that refers to the serial number of the first data byte of a datagram.

– Initial index that refers to the index of a useful part of a datagram in the sender's buffer.

– Length of the useful parts of a datagram is the length of the useful part of the message (submessage) in bytes.

– Serial number of confirmation is the next serial number, which the sender of datagram aims to receive.

– Index number of confirmation is a verification number that arrives after a start index in the sender's buffer, then the initial index is displaced to the location, indicated by the number of the confirmation.

– Type is used to identify the type of a datagram. SCP has two types of datagrams: one contains data while the other contains control information.

– Channel number is the number of the buffer channel to connect using SCP.

In order to ensure reliable transmission of datagrams, the proposed SCP employs a window-like algorithm for data transfer, which is in detail described in paper [17].

5. Discussion of results of examining SCM in a cluster of HPC

We shall estimate the realization of a mechanism of simplified communication (SCM) using data obtained from the experiments described in article [18]. First, we shall describe the experimental environment and the workload,

and then we shall present the experimental results. We shall add to them the results of comparison of performance for the intra-domain and extra-domain sections of a communication network with a protocol of remote memory access (RMAP) and a binary compatible SCM.

In the experiment, we used two identical physical machines. We executed experimental verification of the performance of extra-domain communication between these two machines and the performance of intra-domain communication on a single machine.

Each physical machine was equipped with a two-core processor of 2.5 GHz, memory 4 GB, SATA-drive, the network card Gigabyte and the Gigabyte Ethernet network communication. The same software was deployed on both machines: CentOS 5.0 with Linux kernel 2.6.18.8 and Xen 3.2.0. Virtual machine is equipped with the following: 2 VCPU and 512 MB RAM.

Network performance evaluation was carried out with a parallel set of NAS-tags and NETPERF-tags [19].

A *test with a parallel set of NAS-tags* [20, 21] is a set of tags, which represents the CPU, cache, memory, and the workload of the input/output system in a wide range of actual applications. A unit with a parallel set of NAS-tags consists of five cores (EP, MG, CG, FT, IS) and three simulated complex dynamically operated applications (BT, SP, LU).

For the tests, we chose a problem the size of class A. Parallel set of *NAS*-tags illustrates a large variety for the communication loads within the system. Data are obtained starting from the exchange between the nearest neighboring points to the exchange between points with coarse communication patterns considering all points of communication. The module of tags NPB3.3-MPI was compiled in Open-MPI-l.4.2 based on GCC 4.1.2.

Fig. 9 shows that the relative operation time for the tags using SCP is smaller than for the TCP/IP protocol.

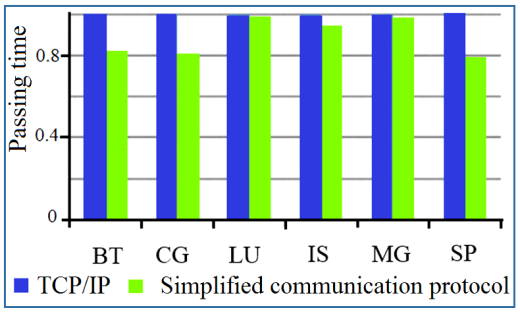

Fig. 9. Test run time with a parallel set of NAS-tags for class A

The *test with a set of Netperf-tags* was conducted in the following way. The first virtual machine on a physical machine worked as a *Netperf* server. The second virtual machine on another physical machine sent a large amounts of data to the first virtual machine in the fastest possible way.

We ran several tests with different time of measurement [22]. Result of the test, given in Fig. 10, indicates that the throughput of SCP is somewhat higher than that for the TCP/IP protocol. It is believed that the TCP/IP protocol is rather well executed in such environment and there are not so many "places" to optimize its operation. Improvement in the throughput amounted to approximately 2.1 %. It is believed that the improvement is mainly due to reducing the time of launching and smaller size of the title packet.

A performance test of intra-domain network connection. In order to improve the throughput of intra-domain network communication, we integrated XenLoop into the system of SCM, which consists of two modules: a detection module and a XenLoop module. We launched several virtual machines on a single physical machine, then downloaded a detection plugin in domain 0 and the module of XenLoop on each of the virtual machines.

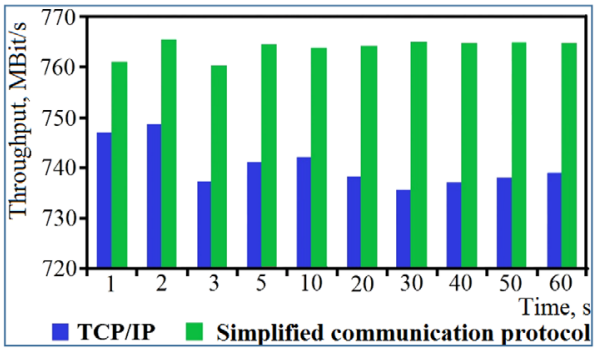

Fig. 10. Comparison of throughput for the protocols of SCP and TCP/IP

First, let us compare the throughput of TCP protocol between two virtual machines that employ *Netperf* between the incoming and outgoing drivers of communication mechanism and the channel of XenLoop. Next, we check the throughput of the local loop channel (described in the literature as loopback) and compare with the XenLoop channel throughput.

Result shown in Fig. 11 demonstrates that the throughput to transmit large amounts of data through a TCP protocol improves by about one and a half. We should also note that the throughput between virtual machines deployed on a single physical machine is slightly better than the throughput for the local loop channel, loopback.

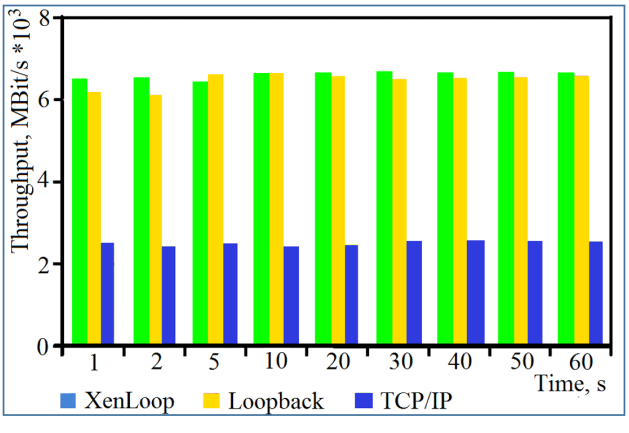

Fig. 11. Throughput of XenLoop, Loopback, and TCP/IP

Next, we shall perform a *comparison of the obtained experimental data for a system with SCP and the system with a protocol of remote memory access (RMAP).* SCP and RMAP are used separately to transmit a specified amount of data from a virtual machine on one physical machine to another virtual machine on another physical machine. Result shown in Fig. 12 proves that the throughput in the case of applying SCP improves compared with the use of RMAP with a growth by approximately 7.8–7.9 %.

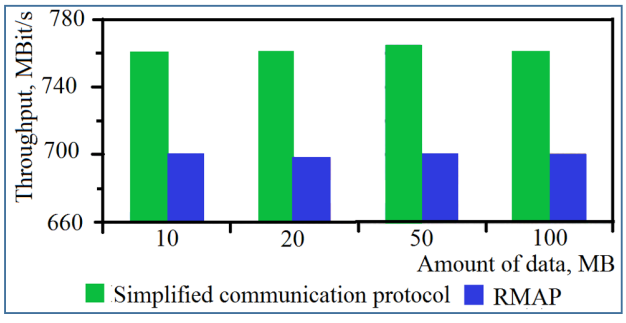

Fig. 12. Throughput of SCP and RMAP

Although SCP and RMAP send data in one packet to their maximum capacity, but there is a difference among them. RMAP-protocol is designed to transfer a page of memory one by one. The final packet with the size of maximum page for transmission over the network cannot reach the final point of destination. That is why the protocol of RMAP compared to SCP must send several pages to transmit a specified amount of data. And this proves the result of improvement in the case of applying the protocol of SCM.

The system with SCM not only provides a better performance (throughput), but also offers a binary compatibility with existing network applications. In order to check binary compatibility with the examined applications, it is necessary to run many common network applications in the examined system with SCM. To realize the study into compatibility of applications, we launched such of them as SCP, APACHE, WGET, NETPERF, MYSQL, VSFTPD, TELNET and some others. All applications have passed the test. However, those applications that do not employ a standard socket interface cannot pass it.

7. Conclusions

1. The work presented a mechanism to optimize network performance for a virtual cluster of HPC, referred to as the mechanism of simplified communication (SCM), which can improve both the intra-domain and extra-domain (external) network throughput, offering at the same time compatibility with the existing network applications. SCM makes it possible to avoid the slow start phase of the TCP protocol and to use a smaller header for a data packet.

2. Experiments prove that the throughput between two virtual machines using SCP is about 2.1 % higher than that for TCP/IP while relative time of a reference test with a set of NAS-tags [20] is lower. In order to improve the performance of intra-domain communication, the system with SCM integrated XenLoop, which can intercept a packet at the level below the network and check whether the purpose of the packet is a coresident virtual machine. If the packet is sent between cooperating virtual machines, then it will be sent through the FIFO channel opened exactly by XenLoop.

3. Result of the study proves that the throughput of the intra-domain communication improves by about one and a half. We compared the indicators for SCP and a protocol of remote memory access (RMAP) that demonstrates that SCP is 7.8–7.9 % faster than RMAP.

References

1. Melnyk, V. Design and implementation of inter-domain communication mechanism for high performance data processing [Text] / V. Melnyk, P. Pekh, K. Melnyk, N. Bahnyuk, O. Zhyharevych // Eastern-European Journal of Enterprise Technologies. – 2016. – Vol. 1, Issue 9 (79). – P. 10–15. doi: 10.15587/1729-4061.2016.60629
2. Melnyk, V. Influence of high performance sockets on data processing intensity [Text] / V. Melnyk, N. Bahnyuk, K. Melnyk // ScienceRise. – 2015. – Vol. 6, Issue 2 (11). – P. 38–48. doi: 10.15587/2313-8416.2015.44380
3. Melnyk, V. High production of java sockets (HPJS) for health clouds in science [Text] / V. Melnyk, O. Zhyharevych, K. Melnyk // Proceedings of National Aviation University. – 2015. – Vol. 64, Issue 3. doi: 10.18372/2306-1472.64.9041
4. Melnyk, V. M. Significance of the socket programming for the laboratory with intensive data communications [Text] / V. M. Melnyk, P. A. Pekh, K. V. Melnyk, O. K. Zhyharevych // Computer-integrated technologies: education, science and industry. – 2015. – Issue 20. – P. 67–71.
5. Barham, P. Xen and the art of virtualization [Text] / P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho et. al. // Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles – SOSP '03. – 2003. doi: 10.1145/945445.945462
6. Pratt, I. Xen Virtualization [Text] / I. Pratt // Linux world 2005 Virtualization BOF Presentation. – 2007.
7. Chisnall, D. The Definitive Guide to the Xen Hypervisor [Text] / D. Chisnall. – 2-nd ed. – Prentice Hall, 2007.
8. Menon, A. Optimizing network virtualization in Xen [Text] / A. Menon, A. L. Cox, W. Zwaenepoel // In 2006 USENIX Annual Technical Conference. – Boston, Massachusetts, USA, 2006. – P. 15–28.
9. Wang, J. XenLoop: a transparent high performance inter-VM network loopback [Text] / J. Wang, K.-L. Wright, K. Gopalan // Cluster Computing. – 2009. – Vol. 12, Issue 2. – P. 141–152. doi: 10.1007/s10586-009-0079-x
10. Kim, K. Inter-domain socket communications supporting high performance and full binary compatibility on Xen [Text] / K. Kim, C. Kim, S.-I. Jung, H.-S. Shin, J.-S. Kim // Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments – VEE '08. – 2008. doi: 10.1145/1346256.1346259
11. Amdahl's Law [Electronic resource]. – Available at: http://home.wlu.edu/~whaleyt/classes/parallel/topics/amdahl.html
12. Liu, J. High Performance VMM-Bypass I/O in Virtual Machines [Text] / J. Liu, W. Huang, B. Abali, D. K. Panda // USENIX Annual Technical Conference archive. – 2006.
13. Hines, M. R. MemX [Text] / M. R. Hines, K. Gopalan // Proceedings of the 3rd International Workshop on Virtualization Technology in Distributed Computing – VTDC '07. – 2007. doi: 10.1145/1408654.1408656
14. Deshpande, U. MemX: Virtualization of Cluster-Wide Memory [Text] / U. Deshpande, B. Wang, S. Haque, M. Hines, K. Gopalan // 2010 39th International Conference on Parallel Processing. – 2010. doi: 10.1109/icpp.2010.74

15. Kim, J.-S. Design and implementation of a user-level Sockets layer over Virtual Interface Architecture [Text] / J.-S. Kim, K. Kim, S.-I. Jung, S. Ha // Concurrency and Computation: Practice and Experience. – 2003. – Vol. 15, Issue 7-8. – P. 727–749. doi: 10.1002/cpe.721
16. Son, S. SOP: A Socket Interface for TOEs [Text] / S. Son, J. Kim, E. Lim, S. Jung // In Internet and Multimedia Systems and Applications. – 2004.
17. Clark, D. D. Window and acknowledgement strategy in TCP [Text] / D. D. Clark // RFC 813. Internet Engineering Task Force. – 1982. doi: 10.17487/rfc0813
18. Menon, A. Diagnosing performance overheads in the xen virtual machine environment [Text] / A. Menon, J. R. Santos, Y. Turner, G. (John) Janakiraman, W. Zwaenepoel // Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments – VEE '05. – 2005. doi: 10.1145/1064979.1064984
19. Network bandwidth testing [Electronic resource]. – Available at: http://semenushkin.ru/2010/07/01/тестирование-пропускной-способности
20. Bailey, D. H. The Nas Parallel Benchmarks [Text] / D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum et. al. // International Journal of High Performance Computing Applications. – 1991. – Vol. 5, Issue 3. – P. 63–73. doi: 10.1177/109434209100500306
21. Overview of some cluster performance measurement systems [Electronic resource]. – Available at: http://www.ixbt.com/cpu/cluster-benchtheory.shtml
22. Netperf: A Network Performance Benchmark. Revision 2.0 [Electronic resource]. – Hewlett-Packard Company. – Available at: http://www.netperf.org/netperf/training/Netperf.html

Запропоновано концепцію і розроблено альфа-версію геоінформаційної системи для виявлення і розв'язання урбаністичних проблем. Вона пропонується як допоміжний інструмент для управління розвитком пострадянських міст Східної Європи. Від аналогічних систем її відрізняє орієнтація на виявлення проблем і встановлення протиріч на території міста. Описано її структуру, перелік функцій та інформаційне наповнення. В результаті її випробувань на тестових ділянках визначені зони з характерними урбаністичними процесами: джентрифікацією, комерціалізацією і ревіталізацією території міста. Ці дані наведено у картографічній формі

Ключові слова: геоінформаційна система, урбаністика, розумне місто, просторова трансформація, міський простір, джентрифікація, комерціалізація, ревіталізація, карта

Предложена концепция и разработана альфа-версия геоинформационной системы для выявления и решения урбанистических проблем. Она предлагается как вспомогательный инструмент для управления развитием постсоветских городов Восточной Европы. От аналогичных систем она отличается ориентацией на выявление проблем и определение противоречий на территории города. Описана ее структура, набор функций и информационное наполнение. В результате ее испытаний на тестовых участках определены зоны с характерными урбанистическими процессами: джентрификацией, коммерциализацией и ревитализацией территории города. Эти данные представлено в картографической форме

Ключевые слова: геоинформационная система, урбанистика, умный город, пространственная трансформация, городское пространство, джентрификация, коммерциализация, ревитализация, карта

UDC 912.43
DOI: 10.15587/1729-4061.2017.98809

DEVELOPMENT AND USE OF A GEOINFORMATION SYSTEM FOR REVEALING URBAN PROBLEMS

A. Oreshchenko
PhD
Department of economic and social geography
Taras Shevchenko National University of Kyiv
Volodymyrska str., 60, Kyiv, Ukraine, 01033
E-mail: logograd@ukr.net
I. Nesterchuk
PhD, Associate professor
Department of Tourism
Zhytomyr National Agroecological University
Staryi blvd., 7, Zhytomyr, Ukraine, 10008
E-mail: nester_geoek@ukr.net

1. Introduction

Transformations in areas of human life activity are a subject matter of Social and Economic Geography studies [1]. These changes help determine the human impact on the environment and identify problems involved and caused by this impact [2].

Research is more detailed if it deals with a variety (rather than intensity) of factor impact on a territory,