DOI: https://doi.org/10.36910/6775-2524-0560-2024-55-18
UDC: 004.451

Melnyk Vasyl[1], Ph.D., Associate Professor
http://orcid.org/0000-0001-8282-6639
Bahniuk Nataliia[2], Ph.D., Associate Professor
https://orcid.org/0000-0002-7120-5455
Roiko Oleksandr[1], Ph.D.
https://orcid.org/0000-0001-8642-7707
Bortnyk Kateryna[2], Ph.D., Associate Professor
https://orcid.org/0000-0001-5282-099X
Kizym Svitlana[1],
https://orcid.org/0009-0003-1205-2086
[1]Volyn Professional College of the National University of Food Technologies, Lutsk, Ukraine
[2] Lutsk National Technical University, Lutsk, Ukraine

SOCKET PERFORMANCE INFLUENCE ON DATA PROCESSING INTENSITY IN A VIRTUAL MACHINE CLUSTER WITH HETEROGENEOUS CONDITIONS

Melnyk V., Bahniuk N., Roiko O., Bortnyk K., Kizym S. Socket performance influence on data processing intensity in a virtual machine cluster with heterogeneous conditions. The paper presents the dependences of data processing in heterogeneous conditions taking into account the impact of socket performance. As high-performance, sockets are proposed on the virtual interface architecture to use a component basis in order to support the application performance for intensive data processing. High-performance applications, using the traditional Linux-based TCP/IP interface, require guarantees of performance and its scalability to adapt for heterogeneous networks. In case of using highly effective protocols for their operation, some additional methods should be used at the user level, including for sockets on the virtual interface architecture. The limitation on the high-performance socket layers is tyed with the application developing completion to full performance salvation during the TCP/IP data transfer process. There also revealed that small-block load balancing has been found to increase heterogeneous network adaptation.

Keywords: virtual interface architecture, sockets, network performance, heterogeneous conditions.

Мельник В.М., Багнюк Н.В., Ройко О.Ю., Бортник К.Я., Кізим С.О. Вплив продуктивності сокетів на інтенсивність обробки даних на кластері віртуальних машин в гетерогенних умовах.У роботі наведено залежності обробки даних в неоднорідних умовах з урахуванням впливу продуктивності сокетів. Як високопродуктивні сокети пропонується на архітектурі віртуального інтерфейсу використовувати компонентну основу для підтримки продуктивності програми для інтенсивної обробки даних. Високопродуктивні програми, які використовують традиційний інтерфейс TCP/IP на базі Linux, вимагають гарантій продуктивності та масштабованості для адаптації до гетерогенних мереж. У разі використання високоефективних протоколів для їх роботи необхідно використовувати деякі додаткові методи на рівні користувача, в тому числі для сокетів на архітектурі віртуального інтерфейсу. Обмеження на рівні високопродуктивних сокетів пов'язане із завершенням розробки програми для повного збереження продуктивності під час процесу передачі даних TCP/IP. Також було виявлено, що балансування навантаження малих блоків збільшує адаптацію гетерогенної мережі.

Ключові слова: архітектура віртуального інтерфейсу, сокети, продуктивність мережі, гетерогенні умови.

Introduction. Nowedays, the research that is using high-performance data processing relies on analytical methods and their implementation for the network applications development. High-performance applications are used for clusters in technical, scientific, medical fields, in cosmonautics and others [1, 2]. Sensor technologies, for example, allow high-resolution support for multidimensional data captured by microscopes. On this basis, there appeares an interest in creating the high-performance network applications for data processing, which could interactively perform basic researches with the possibility of big data generalization and analysis [3]. In their architecture, computing clusters use conventional hardware, becoming the basis of supercomputers for the operation of various applications with intensive processing of large-scale data in the form of arrays, which require some powerful support from system platforms and resources. All these feautures are required for efficient movement of large data between the memory cells of the computing integrated processor. To achieve the cluster high performance, all operations in data exchange and processing must perform with high precision by managing means and their internal dynamics. High-performance applications also require a high level of guarantees for productive work with the ability to adapt to the heterogeneity of work environments and variations in the support system resources.

Network applications and their performance are related to data organization and basic structures that have a component structure [4-6] and are aimed at supporting their flexible operation on distributed computing platforms. Because of the basic structures, the network application integrates a set of interaction software components arranged according to computing resources. This placement architecture directly affects the flexibility of network applications and their productive optimization. In the parallel data processing process, there is also multicopying of data [4] around the storage cluster and computing nodes.

Another implementation method with increased productivity is a pipelining application in the computing cluster with high-speed data processing with redistribution of the data entire volume into user-defined blocks. Such processing can perform in a pipeline, taking into account their new defragmentation volume. Data delivery and processing are CPU-dependent and may overlap during operation, and the computing performance of the cluster will depend on the data blocks or thier transferred volume. In [7] was proved that small data forwarding blocks lead to an increase in the load balance and perfection of the pipeline. Due to this fact, a large number of messages should be transfered in blocks of small sizes after the defragmentation process, even smaller than the characteristic size of the payload message. However, the data delivery in the form of bulk blocks reduces the number of forwarding messages in the network, thereby increasing the bandwidth of the transmission channel with the possible manifestation of load imbalance and reduced pipeline.

Analysis of the latest research. The application on high-speed systems with intercomponent interaction in the process of exchanging messages for software applications was studied in [8,9] with the aim of creating new high-performance flowlines at the user level for exchange systems, which are aimed at reducing the parameters of the delay time and improving the throughput. However, some studies [10] aimed at the high-performance protocols standardization, which also includes the virtual interface architecture. The authors of works [11,12], where high-speed applications were focused on basic TCP or UDP protocols, used traditional socket interface. In order to support applications, there were socket connection approaches implemented at the user level with the high-performance protocols involvement. Applications written for the fundamental TCP/IP protocols for Linux kernel used normal communication. However, high-speed means own other supporting characteristics that affect the critical area of data exchange and its performance [13]. The influential parameter changes for the network application, such as the size of the defragmented blocks, provide an opportunity to obtain advantages in the productivity of working tools, increasing the speed of block processing and adaptability in a cluster system. Such actions and their application in the computing process also apply to sockets based on the virtual interface architecture in order to improve data exchange, increase the component system performance and its operation flexibility. Therefore, the data defragmentation mechanism of large blocks into sub-blocks with a system-specific volume, taking into account its component approach, leads to adaptation ensuring in the proces of data reception and transmission support [4] for applications with intensive processing integrated in the system.

The average delay time of data blocks with updating guarantee per every second under homogeneity conditions is studied in [14]. The first group of studies there concerns the improvement of frame-by-frame data processing level depending on the number of newly generated images or full updates per each second. During the request execution, the average delay time is determined for a partial image update per second, where "depending on the specified frame rate for a request with new images, the TCP socket puts a certain requirement to set the appropriate size of the data block to achieve the intended throughput". Research data obtined claim that for a fixed size of data fragmentation and the use of sockets under the virtual interface architecture there is ensured higher system performance, but from the side of sockets is a requirement to reduce the defragmentation sub-block size.

Our experimental results also confirm the performance advantage in the case of using the same sockets without computational costs presence on the processing nodes. Thus, for the case of using the TCP sockets, a higher update limit of 3.25 per processing second is not reached yet. The sockets based on the virtual interface architecture with data defragmentation into sub-blocks achieve the specified above rate without performance reducing. Received experimental results in [14] withot sub-block redistribution reveal an increase in performance ~3.5 times, and for availability of data block redistribution it goes up more than 10 times. Hance, in case of use the sockets using under the virtual interface architecture and the resizing of subblocks, the application shows significant performance improvements, scalability, and compliance with performance guarantees. It is also submitted that the application performance with data processing costs on the nodes depending from the sub-block size is linear. The values of the processing time for the partial visualization of taken image with using a digital microscopy application [15] and data defragmentation availability are on average ~18/byte of processing. The experiment data in the conditions of heterogeneity for sockets based on the virtual interface architecture with data defragmentation exceeded 3.25 updates/second due to the limitation of bandwidth. However, the experiment shows an overall performance improvement of 3.9-4.1 times for the case of non-defragmented data and ~12 times more with partitioning into sub-blocks.

The second part of the experiment in [14] concerns the update per second with delay time guarantees where the increase in the number of full updates for each image processing second is stated at a constant value of the delay time, which is the maximum for performing a partial update request. The results of application performance growth in homogeneous conditions for a constant value of delay time and TCP protocols reveal the performance dependence on the processing data block size, while in the same conditions, the performance improvement for sockets with a virtual interface architecture is achieved. The data refragmentation at fixed bandwidth and latency results significantly higher performance for these sockets. In the absence of processing costs and a delay time of ~100 μs, the processing performance for TCP sockets drops significantly, and for sockets with a virtual interface architecture it continues to increase up to peak values, achieving the performance improvement in almost 6 times without the participation of data defragmentation. The performance improvement of more than 8 times is observed for sockets with a virtual interface architecture during data block redistribution under homogeneity conditions. Due to the processing nodes overheads, the socket performance under the virtual interface architecture and TCP with high execution guarantee and constant latency is similar. For a processing cost value of ~18 ns/byte, the level of data processing for sockets based on the virtual interface architecture reaches some constant values, but for TCP sockets it is achieved faster. For the sockets proposed in this work, the framing level and the latency decrease are changing slightly, but the data processing performance increases by four times.

As described in [16], a high-performance socket used in a network interactive application to guarantee the productive operation of such a system in relation to the user due to the adaptability of the executable programs for productive processing in heterogeneity conditions. The results of the given data indicate that with the help of computing system reorganization with high-speed data processing, it is possible to achieve high performance of applications. Productivity growth should conduct high-speed computing guarantees and favorable stimulation of high scalability. At the same time, the involvement of an approach to data fragmentation and the structuring involvement to achieve a load balance in order to adapt additives to the environment heterogeneity, which as a result reduces resource variability, has a great impact.

Computing technology improvements, its power and memory have led to an increase to the potential for the network applications development and using in combination with the ability to store data in large volumes. To analyze the preservation of big data, the use of visualization codes and interactive visualization implemented the possibility of taking over a specific system from the inside. At the same time, the analysis and visual representation of data blocks can play a decisive role in many scientific studies. This mostly applies to programs implemented to work in an interactive high-speed way of executing requests and analyzing data blocks. The most frequently used high-performance application is the analysis of data taken from a microscope and their restoration with a high resolution [13] and digitization. Such an application uses relevant characteristics in its work, such as an instance of a video-captured study and a motivating script for data processing.

Powerful network applications should always work in batch mode with the ability to generate and process large-scale data arrays, the processing of which depends on external and internal high-speed parameters, which, for example, were studied in [13] on a cluster of virtual machines for high-speed processing. The cluster must exhibit binary compatibility for network applications with a traditional interface. With the use of the simplified communication mechanism in the system based on Xen 3.2 and the Linux kernel, an improvement of the external and internal speed parameters in the virtual machines communication was found, similar to the organization of UNIX DOMAIN communication. Bandwidth at the same time between machines using simplified communication became 2.12% greater than for traditional TCP/IP protocols, and the speed of message exchange improves gone up near by ~7.8%.

The operation of application support for searching, processing and saving digitized slides of studies made with a microscope, taking into account interactivity in the operating interval process of the microscope and behavioral displays are presented in [17]. The main difficulty that occurs in the big data processing of high-resolution digital images is proportional to the size of their volume. Such images can vary from hundreds of Mbytes to several hundreds of Gbytes, and the system and software must clearly display the current data of the working microscope with high accuracy, taking into account the continuous manipulation of its volume and frame-by-frame acquisition. The processing of such customer requests requires high-resolution projection of data on a selected resolution area, taking into account the pixel distribution at each resolution point. In this case, the visualization server of digital microscopy should be ready to receive and store typical requests in conditions of heterogeneity that will come from the client side.

In practice, the most frequent are requests for a complete update of the aggregate image, which after processing should present a completely new image. Requests for partial typical updates to display the image in panned or zoomed views are also common. The server must handle both types of requests. The digital processing of big data by query applications that modify a set of scientific data is quite often similar to an acyclic coarse flow in the form of defined blocks from one or more sources to processing nodes of clients. It can be one or more data sets with a corresponding distribution, corresponding to the system of their storage. The interest data for such a request is removed from the defined sets and is processed in the appropriate sequence of scheduled operations on the nodes. Therefore, the data of the digital microscopy frame processing application is combined during the addition of digital sub-blocks and the corresponding viewing [4,18], and next it is sent to the client.

Stored data sub-blocks contain partial indexed images or their fragments necessary to obtain a block, where a complete image can consist of multiple blocks. A partial update request may require some defined portion of blocks, and to execute them, the byte size and length of the block affect the amount of data that is important to exchange between processing nodes.

High-performance application support. To improve the high-speed system performance, one can take into account various effective influence directions, the first of which is the intermediate refragmentation introduction of data volumes in order to establish parallelism during the execution of I/O operations and obtaining data for the request execution. Due to the block pre-calculated matching after declustering, a query started for execution can occupy the maximum number of disks involved. On the other hand, the manifestation of the effective processing power of data blocks in the system affects the working application, developed with parallelism in mind. A third effect of improving the performance of applications working interactively with defragmented data sets is their pipelining, where the total time spent on data processing can be correspondingly reduced by defragmenting the data and pipelining it for processing, which in turn provides a final mechanism, forming the final version of their integration. Therefore, it is not necessary to wait for the completion of the request, but to keep track of the block results that are partially processed. This does not reduce the full time of processing the request, but it is effective in the work of interactive applications, taking into account the shift of the playback horizon. The parameter of the sub-blocks size should take into account the network bandwidth and the delay time in the process of their network exchange. The delay time also takes into account the transmission of the full processing message using the protocols used by the current system along the full transmission path from the sender to the receivers. The number of messages to transmit the same data volume decreases with the increase in its size due to defragmentation [14], and the system bandwidth or performance plays a more important role. As the size of the sub-block increases, the processing time for each of them increases, and as the number of separate or consecutive updates of the aggregate image decreases, the system loses sensitivity, and if the size of the sub-block decreases, then the number of them increases for the transmission of the same data amount over the network. Then the message waiting time can be dominant in the efficient operation of the network application. However, smaller subblocks result in better application load balancing, and performance may suffer slightly due to connection overhead.

According to previous studies [12,14] was found that increase in the size of the transmitted data blocks leads to an improvement in the parameter of the system response time for the complete request update, which is proportional to its bandwidth. In the implementation of a partial update request, there is an increase in costs for processing sub-blocks accompanied by an increase in the data amount. Conversely, as transmission block sizes decrease, a partial update request will receive less redundant data. However, a full update request in this case will suffer losses due to reduced system bandwidth. An increase in bandwidth and a decrease in system delay time with a decrease in the size of the message causes also increase in network bandwidth [16]. For example, when working with TCP/IP kernel sockets, the reduction in system latency during the transmission of messages with fixed volume is quite noticeable and leads to an increase in the performance of its work.

The load inhomogeneity of the due to heterogeneity can be argued by the hardware environment, the components of which can be machines with different power and memory capacity, as well as the resources distribution can overlap between other working applications. In such cases, CPU and memory resources can undergo dynamic changes, and the working application must be adapted to the heterogeneity of the environment during operation, optimized for the use of shared resources and for their changing availability and accessing. Here it is required to use its own adaptation mechanisms to balance the inter-node workload according to the computing capabilities of each involved work node. For this purpose, the distribution planning of predetermined output data and calculations should be implemented on the application processing nodes. The data fragmentation may be done so that processing and inter-node communication can be performed in pipeline mode. Data fragmentation can be performed taking into account the demand pattern, in which fast nodes could receive a larger volume of data. If such a high-speed node slows down in the data processing process due to the actions of other applications or other reasons, then the load balancing mechanism must quickly detect changes in the resources availability and/or their accessability.

Application developers need to take into account that in the course of work, component-based tools [5, 6] have the ability to ensure the efficiency of the working environment. Such tools can be located on different resources, but the execution of the task and the flow of parallel data processing can be significantly improved due to the pipeline of their multi-copies. In our work, we also use a component infrastructure for dividing data into blocks [4] in order to support the operation of high-intensity applications in distributed environments. For this purpose, an implemented highly efficient socket interface is developed according to the virtual interface architecture for TCP/IP applications, to provide an advantage in the implementation of increased data processing performance.

The data defragmentation method into blocks. Let's describe the means structure to divide the data into blocks [14] in order to implement a software filter-flow model for applications with intensive data processing and their development. It need to submit that according to this model, the application structure is implemented with components, where each of them is a filter exchanging data using the separation of their starting flow. The filter interface includes three functions: an initialization function for allocating and initializing memory for data structures and required resources; user-defined functions for processing operations on data elements; and initialization functions for completing and freeing resources. The filters connected by unidirectional data streams from the sender of mailings to the recipient perform the functions of reading them from the incoming streams and writing them to its output streams. A data buffer, represented by an array of elements, also works for transferring them between filters. A logical flow implementation divides data into fixed-size buffers and feeds them over a TCP connection between interaction points.

Therefore, the application structure is realized with groups of interconnected filters and logical flows. At a moment, any filter group is started to process a request, before executing it the operating system makes a socket connection between the filters on the remote hosts. Filters on a single machine represent separate threads, and the request of the running application is processing as a single module through the intended filter group. Data processing by one working module is carried out in the pipeline mode, in which different filters simultaneously work with different data. Processing by a single worker module occurs when the filtering process requests the function to initialize the filter, which is together with all the necessary resources [14]. On the next, the function of processing and reading data from input streams, the function of working with data to place them in buffers, and the function of writing data to output streams come into operation. After processing finish the last data buffer, a system marker transmits to submit the end of a distinct work module. After the processing end of the current work module, the finalization function is triggering that frees the involved resources and the workspace. To process the next single working module, the mentioned interface functions should be activated again.

The program model completes several types of distribution to facilitate performance optimization by performing distribution of logical input and output streams and a transparent filter copy, which can be implemented without affecting the filtered group semantics. In this case, the vision of the work block is unchanged regarding the transparent copies and their number. The transparent filter copies also provide the ability to implement data parallelism for a single query, and multiple filter groups allow the parallelism implementation between multiple queries.

To organize communication between sender and receiver logical filters at runtime, the filter system maintains a single point-to-point logical flow that is responsible for buffering the data flow between filter transparent copies. That is, if copy P1 performs a buffer write to the logical flow for the connection from filter P to filter Q, then, for example, the data buffer will be sent in copies to Host3 or Host4. During the distribution between transparent copies, the working system supports a cyclic migration mechanism and a demand management mechanism based on the level of buffer consumption. The demand management mechanism sends buffers to the designated filter for the fastest their processing. At the beginning of the data buffer processing received from the sending filter, the consumer filter informs its message that this buffer is being processed. For sending processing buffer data, the sender filter also selects a usage filter with the minimum number of unacknowledged buffers, achieving improved load balancing.

The research purpose and tasks. The study of the data processing intensity dependence from the socket performance with taking into account the environment heterogeneity in the processing cluster aimed to use a component infrastructure with a combination of secondary sub-block division for output data streams by the processing system to increase the performance of TCP applications in distributed environments together with a combination of a highly efficient socket interface created according to the virtual interface architecture. To achieve this goal, it is necessary to complete the following tasks:

– To implement a cluster system on personal computers using TCP-sockets and sockets based on the virtual interface architecture using a mechanism for dividing data into sub-blocks to study the heterogeneity parameters of message processing.

– Provide the performance indicators assessment for the message processing with using TCP- and virtual interface architecture sockets and combination of splitting data into sub-blocks in the research process by the mini-tests method for delay time and bandwidth parameters during direct and indirect influences on the application operation.

– to investigate the delay time average value for the processing of the data original block in heterogeneous conditions with a updating per second guarantee.

Theoretical foundations of virtual interface architecture sockets. Along with the applicatios development based on TCP and UDP protocols, nowadays there is a rapid development of protocols with high throughput or low latency at the user level. The socket interface based on the traditional TCP and UDP protocols is widely used in network applications, but the cLAN network is a clear example of the hardware implementation for the virtual interface architecture [14]. The socket implementation in the cLAN network is of two types, one of which is based on maintaining the imitation socket, immutability of TCP/UDP and IP levels. Also, an additional layer is introduced to connect the IP layer and the kernel layer of the virtual interface. There is introduced application of a LAN emulator [8] at the socket level for use from the IP level to the virtual interface level. During the final actions of system calls including kernel context switch, TLB processing, cache filling, actions of sublevel device handlers, etc., and also introduced multicopies in this implementation, the applications with using LANE did not achieve a clear advantage in performance increasing for the basic network [14].

To provide the socket interface in the cLAN network, there is another type based to use on the user level libraries that practice primitives based on the virtual interface architecture, and takes place in our work. According to the virtual interface architecture, there is a reference to the specified socket level. This implementation for sockets based on the architecture of the virtual interface is not the main one, and the micro-test results for them are given in [14, 19]. The paper will conduct mini-test studies of performance growth depending on the delay time and bandwidth using sockets based on the virtual interface architecture. To evaluate these parameters, the direct and indirect influences on the performance of high-production socket applications in the presence of splitting the output data into blocks will also be considered. The research was conducted on a personal computers cluster (420 nodes) connected by Giganet cLAN and Fast Ethernet networks using cLAN 1000 guest adapters and cLAN5300 cluster switches. Each node is subordinated to two Pentium III – 1 GHz processors on the Intel 840 chipset with four 32-bit 33 MHz PCI connectors, which are equipped with 512 MB SDRAM with 256 K L2 cache memory on the Linux-2.2.17 kernel.

Fig. 1 presents the theoretical and experimental dependences of the delay time for high-performance sockets, sockets based on the virtual interface architecture, and the dependence of the same parameter for traditional TCP sockets at their top on the message size [14]. To compare the received data, it can submit that the practically implemented socket level shows a low delay time value of 9.5 s., which is quite close to the theoretically calculated one for sockets based on the virtual interface architecture. From the dependencies shown in Fig. 1 is well observed that the delay time parameter increased by almost five times for traditional TCP/IP sockets to compare with high-performance ones. However, when the size of the message blocks increases from ~128 bytes with their further growth, the delay time monotonically increases sharply for all types of sockets, and it can be seen on the behavior of the dependencies shown in this figure.

Fig. 2 shows the theoretically calculated and experimentally obtained dependencies of the bandwidth achieved for high-performance sockets based on the virtual interface architecture and traditional cLAN-based one [14]. As you can see from the given comparative dependence curves, the throughput for high-performance sockets based on the virtual interface architecture reaches a peak behavior at a value of ~763 Mbytes/s, while for their theoretically calculated throughput curve, the peak occurs at a value of ~795 Mbytes/s, and for the traditional implementation of TCP – at a value of 510 Mbytes/s. Therefore, comparing the peak throughput for different sockets, it can be seen that it is improved by almost 50% for sockets with a virtual interface architecture compared to traditionally implemented TCP sockets.

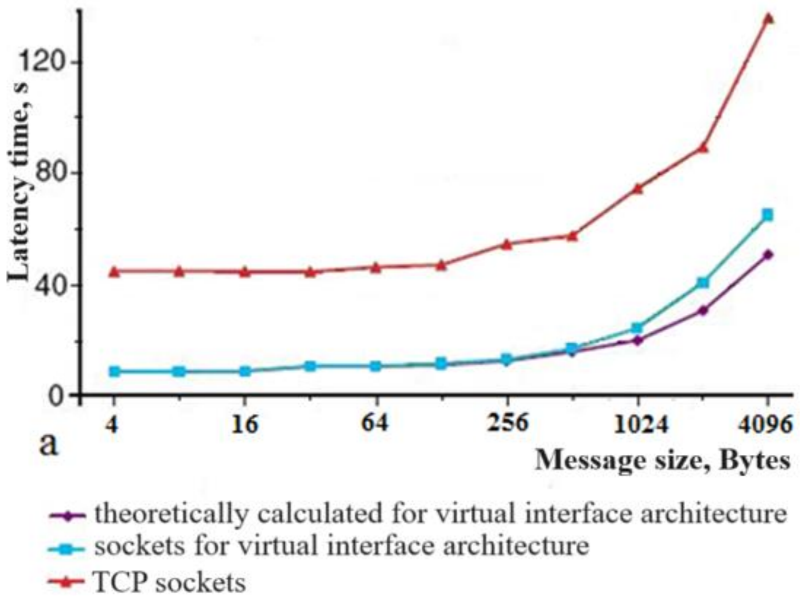

Fig. 1. Marks to compare the delay time dependence on the message size [14]

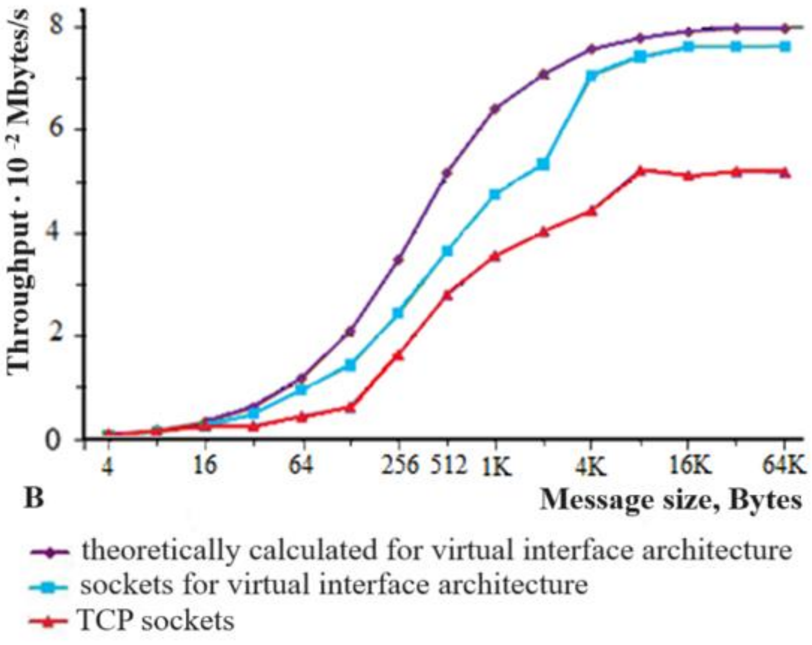

Fig. 2. Marks to compare bandwidth dependence on message size [14]

Experimental equipment. Two types of applications were used in the performance indicators study. The first type is intended for emulating the visualization server and with its help a four-stage conveyor mechanism is implemented programmatically, at the final stage of which the visualization filter is included. In order to improve throughput at the final stage of the pipeline, three copies of each filter were performed for each study. Visually, the layout of the filters is shown in Fig. 3 [13]. The image was reproduced by the user in the corresponding visualization node with the help of a filter, after which the necessary data was selected from the storage and transferred to other filters for the next work stages. Each of the following filters was placed in the system on another node included in the specified pipeline.

The image for consideration was 16 Mbytes of data for its acquisition and processing. Such predefined distribution block data will contain parts of the image, but the complete image for each distribution block will consist of several sub-blocks. To fulfill a request for a partial or full update of an integrated image, the system must deliver all the necessary blocks for their inclusion in it. It should be taken into account that each block must be delivered intact, which may require additional data initialization and their costs, including their transmission over the network to the place of unification into a single object.

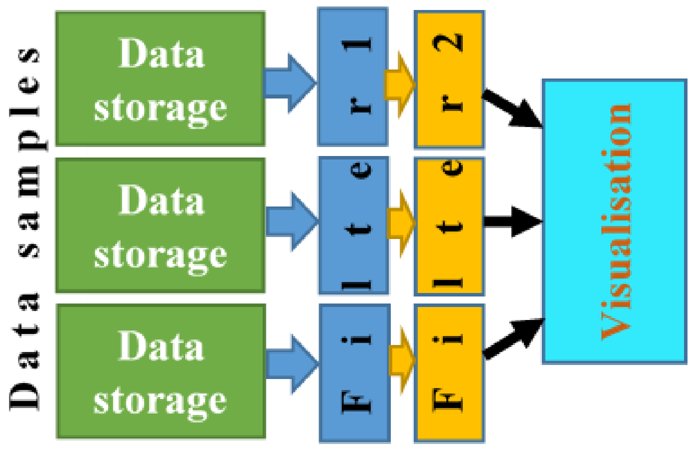

Fig. 3 – Experimental scheme [13]

Duering the research, two types of requests were implemented. The first one required a complete image renewing or a complete merging of all blocks involved in the composite image. This type of update from the side of practical implementation encourages the use with large block sizes and is bandwidth sensitive. Therefore, for the full update request with taking into account the corresponding speed, it is necessary to determine in advance the size of the blocks for storage and transmission, which would fulfill the user's conditions.

The completion of the second request or a partial image update takes place in the case of some image displacement rendering window or a change in its scale. For this request and its productive execution it is necessary to add only additional blocks to the existing blocks if possible in a small number compared to the full image reproduction. Productive execution of this request becomes dependent already on the delay time of additional blocks transmission. For partial recovery mapping fragments are usually extracted or narrowed, and the use of small data blocks is preferred in this case.

If the size of the blocks is relatively large in volume, then the partial update of the image will take a longer exposure time due to the modification of each block, even when a part of it is involved in the composite image. Using small block sizes during a full update will take longer due to changing a large number of blocks. If the application is configured to handle two types of mentioned requests, then it is necessary to correlate this agreement in the process of their execution. In works [13, 14] the improvement in the application performance using sockets based on the virtual interface architecture and guarantees of the requests execution for two types of updates was found, the results of which were compared with the results of the same studies on traditional TCP sockets.

Another implementation of the productive execution for the mentioned requests includes a balancing mechanism for distribution the load on the applications work due to the means of data fragmentation into blocks of appropriate sizes. For data processing by several nodes, the pipeline improvement will be achieved under the condition that the balanced load time goes to the time of data unit forwarding to the processing node and is equal to the time spent processing it on this node. Thus, taking into account the time intervals for processing and communication, a characteristic block size establishes for the application productive operation and achieving a better balance in the pipeline. In [14] there was taken into account the invariance of the computing nodes power during the application's operation time. However, due to the heterogeneity of the mentioned parameters the implementation of this approach in a dynamic heterogeneous environment is not possible.

According to the data obtained from the mentioned researches and their comparison, a high consistency for TCP/IP sockets during the pipeline implementation is achieved in case of breaking the initial data into sub-blocks of 16 Kbytes. For sockets based on the virtual interface architecture, such consistency is achieved during the refragmentation of the initial data into sub-blocks in size of 2 Kbytes. From such results follows that for TCP/IP sockets the characteristic block size is larger when for sockets with a virtual interface architecture it is smaller. During the information transmission through heterogeneous networks, the sizes of the blocks can be heterogeneous and quite large. In this case, anomalies and deviations from normal load balancing during the transfer of data blocks to a node, in which processing is slowed down, can lead to an increase in costs. Comparing the data presentation scheme below for the case of a heterogeneous cluster (Fig. 4), which was taken to analyze the impact of block heterogeneity on system performance, it is possible to predict cases of such growth.

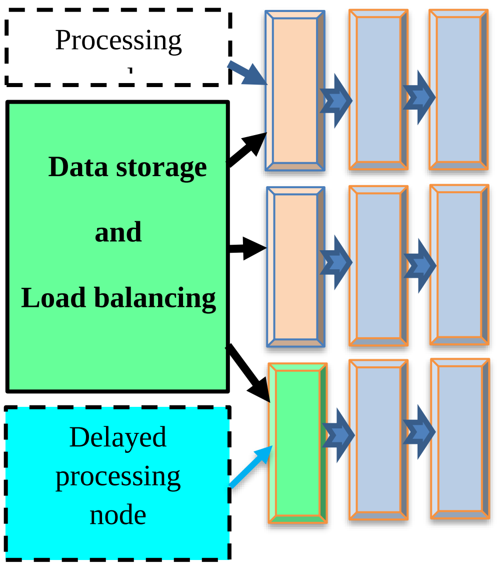

Fig. 4. Scheme of data management in heterogeneity conditions

Results and their discussion. In our conducted studies, the influence of virtual interface architecture sockets on the performance indicators of cluster computing nodes with a heterogeneous data set is observed. To gain this a slowed nodes emulation is made in the network, which, accordingly, force the data processing to be performed on other nodes more than once. For TCP protocols, the slowdown in data processing speed may take place due to a decrease in the speed of their exchange and an increase in their processing time. However, the following conditions are observed in the experiments: the communication time remains constant, and the data processing time may change in magnitude.

Also, the experiment investigates the impact of the cyclically migrating mechanism embedded in the buffer to perform data defragmentation on the performance for their processing. To compare the obtained results, the research was conducted using traditional TCP sockets and sockets based on the virtual interface architecture. The time of data transfer to the processing node under the compliance conditions with high pipeline should be the same as the time of their processing on each working node in particular. For this purpose during the experiment is monitored the balance of the load distribution on the active filters of the visualization application, or on the first nodes of the conveyor line (see Fig. 3). The data processing time on each of the filters depends linearly on the message volume and is approximately equal to 18 ns for each of its bytes. According to the obtained experimental data for traditional TCP sockets, the ideal pipeline was achieved for message sub-blocks with a size of ~16 Kbytes. In the case of the experiment for sockets based on the virtual interface architecture, the high co-verification rate was observed for message blocks with a volume of ~2 Kbytes. This proved that in the course of balancing the load and achieving high conveyance, it is necessary to analyze the influence parameters on them in more detail.

The dependence of the time amount for load balancing on nodes in conditions of heterogeneity with the growth of the network heterogeneity factor is shown in fig. 5. It can be seen from the given dependence that the value of this factor is affected by the processing speeds of the slowest and fastest processors. For the experiment with TCP, the message size is relatively large, which is ~16Kbytes, and an error occurs in the load balancing process. A block of data, which is transferring to process on the slower node leads to an increase in time costs. In general, this situation leads to an increase in load balancing time, which is an important factor for the system productive implementation. In studies with virtual interface architecture sockets, the characteristic value of the block size is smaller and the load balancing time required for its processing in a slow node, taking into account the error occurrence, is smaller to compare with TCP sockets. At the same time, the response time parameter for load balancing also decreases. Therefore, the obtained experiment results with sockets based on the virtual interface architecture show a decrease in the response time for load balancing compared to TCP sockets by approximately 8 times.

There was implemented cluster system in the experiment, which embodies the planned request management scheme (fig. 4). Based on such a system, the effect of heterogeneity on data processing speed, i.e., on the performance of the request control mechanism, is studied by performing the task of buffer planning with the division of data into blocks for TCP sockets and virtual interface architecture sockets. The result of high pipelining according to the cyclic scheduling scheme for sockets according to the virtual interface architecture was achieved by dividing the data into blocks of 2 Kbytes and for traditional TCP sockets the block size was 16 Kbytes. Fig. 6

shows dependence graphs for application execution time on the probability indicator of its slowdown. Note that the probability of slowing down is proportional to the heterogeneity parameter at the nodes. In the experiment, it was assumed that the dynamic value of the processing node should decrease over time. On the abscissa axis, as can be seen from the above graphs, was plotted the probable percentage of the slowdown. Analyzing the slowdown probability indicator equal to 20%, it can be understood that the data processing of 20% received by it is performed at a certain time with a slowdown while at the same time, the other received data processing of 80% takes place in its initial dynamics.

Figure 6 shows the dependencies for the two types of sockets described in the work with a variable heterogeneity index. From the analysis of the given dependencies it can be seen that the performance indicator of applications with traditional TCP protocols and the same coefficient of heterogeneity is quite close to applications with sockets based on the virtual interface architecture. Convergence in the magnitude of these indicators proves that, in accordance with the mechanism for managing requests to consumers and the data blocks sizes, processing nodes on less loaded processors are given the opportunity to perform a larger volume of work. At the same time, observing data pipelining ensures that it is forwarded and timed between the communication and the processing process.

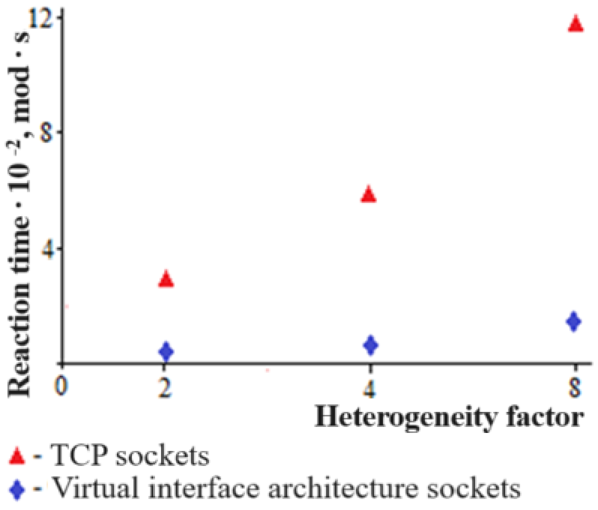

Fig. 5. The processing speed dependence during load balancing according to a cyclic planning scheme in conditions of heterogeneity

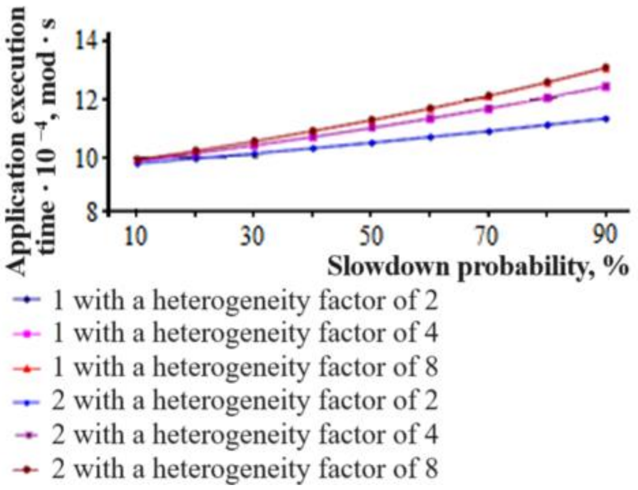

Fig. 6. The data processing speed dependence during load balancing according to the scheme of planned demand management: 1 – socket application for virtual interface architecture; 2 – application for TCP sockets

The obtained experimental results also confirm that in the case of using traditional sockets on the hardware operating configuration, the applications should be structured in such a way that it is possible to implement data pipeline for them in accordance with their dynamic planning. Also, in order to ensure the performance of applications and the guarantees for the waiting time given by them, the use of high-performance sockets is essential.

Conclusions and research perspectives. Based on personal computers the cluster system is implemented with the involvement of traditional TCP and virtual interface architecture sockets, and the data redistribution mechanism for heterogeneous processing. According to the results of the mini-tests, an approximate estimate of the maximum data processing performance for both types of sockets is given, based on the experimentally obtained values of the delay time and bandwidth parameters and taking into account the direct and indirect effects on the applications. Without taking into account the processing costs on the nodes for sockets based on the virtual interface architecture, there is a performance improvement of more than 3.5 times, and for the same sockets with available data block redistribution it takes more than 10 times. More than in 8 times the performance gains are seen when using block redistribution actions with latency-per-second guarantees.

Experimental analysis proves that intensive applications using the interface of traditional TCP/IP sockets based on Linux require a guarantee of performance, scalability and adaptation to heterogeneous networks. For the same applications, the use of high-performance protocols, which include sockets based on the virtual interface architecture, requires the involvement of a number of methods at the user level. Highly efficient socket layers are generally limited by applications written for performance-preserving TCP/IP connections. Due to the componentization and block fragmentation to ensure the operation of applications with intensive data processing, the obtained experiment results prove the increase in productivity conditioned by reorganization of their individual components. The restructuring approach stimulates the application scalability growth with performance guarantees. An increase in the adaptation of applications to heterogeneous networks is also observed in the course of small-block load balancing.

The obtained experimental results also prove that the parametric characteristics of the sockets according to the virtual interface architecture and the use of data fragmentation on the initial processing nodes lead to an improvement in performance, and in some cases by an order of magnitude. In this case, high-performance sockets with significantly lower data computing costs allow data processing applications to achieve significantly higher quality parameters. This fact can be openly applied on modern high-performance processing clusters and in the design of intensive data processing applications on nodes.

References

1. S. Iannucci, V. Cardellini, O. D. Barba, I. Banicescu. (2020) A hybrid model-free approach for the near-optimal intrusion response control of non-stationary systems. // Future Generation Computer Systems. 2020, V 109, p. 111-124.
2. Nagarjuna, Setti Vidya Sagar Appaj. A real-time communication strategy for wireless sensor Networks with differentiated services. / Industrial Engineering Journal. 2022. Vol51, Issue 03. P.36-52. ISSN: 0970-2555.
3. Howard F. What Is Quality of Service (QoS) in Networking? 2023. E-Resource. Available at: https://community.fs.com/article/what-is-quality-of-service-qos-in-networking.html.
4. E.R. Rhythm, R. A. Shuvo, H. K. Mehedi, S. Hossain, A. A. Rasel. Distributed Computing for Big Data Analytics: Challenges and Opportunities. / Computer Science and Engineering. Distributed Computing. Preprint. 2022. 5 p. DOI:10.13140/RG.2.2.12511.53927/1. Available at: https://www.researchgate.net/publication/366466213_Distributed_Computing_for_Big_Data_Analytics_Challenges_and_Opportunities
5. F. Isaila, J. Carretero, R. Ross. CLARISSE: a middleware for data-staging coordination and control on large-scale HPC platforms. CCGRID '16: Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. May 2016. P. 346–355. Available at: https://doi.org/10.1109/CCGrid.2016.24.
6. Morgan Fluhler. Effectively Handling Large Datasets. 2023. Dataiku. E-Resource. Available at: https://blog.dataiku.com/effectively-handling-large-datasets.
7. V. Melnyk, K. Melnyk, S. Lavrenchuk, I. Burchak, O. Kaganiuk. Influence of the message direct search mechanism based on the TCP protocols to the exchange process. East-European journal of Enterprise Technologies. Kharkov (Scopus DOI:10.15587/1729-4061.2019.167995). 2019. №3(2)99. p. 36-42.
8. Durgesh Pandey. Inter Process Communication (IPC). 2023. E-Resource. Available at: https://www.geeksforgeeks.org/inter-process-communication-ipc/.
9. Tom Tobul. Powering Advanced Creative Workloads with Entry-Level Workstations. 2021. Dell Blog. Available at: https://www.dell.com/en-us/blog/powering-advanced-creative-workloads-with-entry-level-workstations/
10. Virtual Interface Architecture and Support for SAN. 2021. E-Resource. Available at: https://learn.microsoft.com/en-us/windows-hardware/drivers/network/virtual-interface-architecture-and-support-for-san.
11. C. Shim, R. Shinde, M.Choi. Compatibility Enhancement and Performance Measurement for Socket Interface with PCIe Interconnections. / Hum. Cent. Comput. Inf. Science. 2019. Doi: https//:doi.org/10.1186/s13673-019-0170-0.
12. Shah H.V., Pu C., Madukkarumukumana R.S. High Performance Sockets and RPC over Virtual Interface (VI) Architecture. In the Proceedings of CANPC workshop (held in conjunction with HPCA Conference), p. 91-107, 1999.
13. Melnyk V., Melnyk K., Kuzmych O., Bahniuk N., Kraviez O. Internal and external communication performance improving research on a cluster of communicated virtual machines. / Scientific journal "CIT: education, science, production". 2020, N.39. P. 162–174.

14. Melnyk V., Kaganiuk O., Kozlenko M. et al. Data processing intensity dependence in the cluster on socket performance avoiding heterogeneity. / Scientific journal "CIT: education, science, production" 2020, № 40. P. 128-139.
15. M. Hortsch, N.K. Koney, A.M. Oommen, D.G. Yohannan, Y. Li, A.C. Rocha, V.C. Girao-Carmona. Virtual Microscopy Goes Global: The Images are Virtual and the Problems are Real. Advances in Experimental Medicine and Biology. / Part in the book series Biomedical Visualisation. 2023. Vol. 1421. P. 79-124.
16. Melnyk V., Bahniuk N., Melnyk K. Influence of High Performance Sockets on Data Processing Intensity. // Scientific Journal ″ScienceRise″. Kharkiv. 2015. V.6, № 2(11). P. 38-48.
17. R.C.N. Melo, M.W.D. Raas, C. Palazzi, V.H. Neves, K.K. Malta, T.P.Silva. Whole Slide Imaging and Its Applications to Histopathological Studies of Liver Disorders. / Frontiers in Medicine. 2020. 13 P. doi: 10.3389/fmed.2019.00310. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6960181/pdf/fmed-06-00310.pdf.
18. M. Abughazala, H. Muccini, M. Sharaf. Architecture Description Framework For Data-Intensive Applications. / Fourth International Conference on Intelligent Data Science Technologies and Applications (Conference Paper, 8 pages, October 24-26, 2023). ResearchGate. DOI:10.1109/IDSTA58916.2023.10317869. Available at: https://ieeexplore.ieee.org/document/10317869/metrics#metrics.
19. J. Rexford, I. Stoica. Socksdirect: datacenter sockets can be fast and compatible. / SIGCOMM '19: Proceedings of the ACM Special Interest Group on Data Communication (August 2019). P. 90–103. Available at: https://doi.org/10.1145/3341302.3342071.