Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Scalable Parallel Module for Capturing Multiple Video Streams in Real Time

Omar Antonio Hernández Duany[1], Caridad Anías Calderón[1], Cornelio Yáñez-Márquez[2*], Roberto Sepúlveda Lima[1], Fernando de la Nuez García[1]

[1]*Technological University of Havana, Cuba*
[2]*Centro de Investigación en Computación, Instituto Politécnico Nacional, México*
**Corresponding author: cyanez@cic.ipn.mx*

ARTICLE INFO

Received: 30 Dec 2024

Revised: 15 Feb 2025

Accepted: 25 Feb 2025

ABSTRACT

The process of capturing multiple IP video streams in real time is of great importance for the development of heterogeneous computer vision systems, because it is the first stage for the management of various processes that base their potential on visual analysis for decision making in different organizational environments. This process demands the execution of a high number of operations that guarantees that several concurrent video streams are managed without losses and under time restrictive conditions. In this sense, a parallel and scalable software module has been designed to optimize the use of computational resources in order to capture a higher number of concurrent video streams. The parallel algorithm is based on shared-distributed memory programming paradigms and improves the efficiency of the use of the network adapters architecture and integrates the potential of all the processing cores available in the computational node that supports the process, based on the cooperation between processors and computational nodes on a high-performance computational infrastructure with the purpose of promoting the scalability of the system. It is important to point out that the video capture module has transversal applicability in several application domains and makes it possible to increase the number of video streams to be processed concurrently in correspondence with the available hardware architecture.

Keywords: Video Capture, Real Time, Opencv, Parallel Programming, Openmp.

INTRODUCTION

Nowadays, many organizations from different sectors of society have associated intrinsic processes in which it is required to manage real-time video content for decision making, based on the analysis of previously defined requirements. It is important to consider that specifically in the field of telecommunications, the process of capturing real-time video streams is applicable to various domains [1].

Many technological solutions in heterogeneous organizational contexts require the realization of the capture process prior to the visual analysis of the contents as a resource to achieve the effectiveness and accuracy of various decision-making processes. It is important to consider that at the current stage video capture has transversal applicability, since it is indispensable for the deployment of a large number of technological solutions based on pre-trained artificial intelligence models, which emulate the capabilities of the human brain in terms of making increasingly accurate decisions based on visual analysis. This approach enables real-time processing of an increasing number of IP video streams to which it is possible to incorporate other functionalities such as noise reduction techniques and frame detail enhancement where required.

Among the application domains of the capture module are the functions of object recognition, evaluation and classification, among others. Behavioral analysis of production processes, quality control, visual inspection systems, unsupervised analysis, robotic solutions, control of unmanned vehicles and autonomous driving, video surveillance systems, vehicular traffic control, control components of smart cities, smart villages, among many other uses.

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

OBJECTIVES

The development of any computer vision system is based on the classical model composed of the sub-processes shown in Fig 1. The development of this capture module has been specifically focused on the video capture sub-process, which integrates the first phase of the system and is the prelude to the remaining phases. The peculiarity that distinguishes this design is the integration of an efficient method for optimizing the utilization of the network adapters according to the processing of IP video streams, as well as the optimization of the processing capacity of the hosting computational nodes [3].

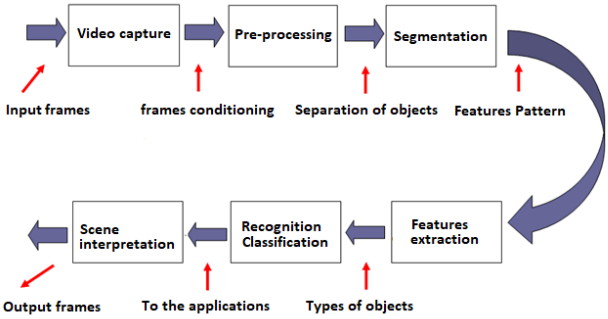

Figure 1. Sub-processes of the computer vision system.

The scalable parallel capture module is part of a modular software infrastructure that can be harmoniously scaled horizontally or vertically, supporting technological solutions that require advanced content processing of real-time video streams.

The module for capturing multiple real-time video streams from IP cameras is deployed in the first phase of the process with the primary objective of accelerating the processing of the frame sequence without loss of quality by maximizing the efficiency of use of the network adapter architectures and the computational node that supports them. The modular design of the infrastructure [4] contributes to the compatibility between each of the modules that compose it.

The reduction of the computational node processors' usage time is essential, because the less execution time consumed by the sub-process of capturing multiple video streams, the possibilities of increasing the processing complexity of the modules required for decision making are expanded, and other processing modules can be added successively according to the requirements of the applications. This capability contributes to the extensibility of the potential of any computer vision system for making increasingly accurate decisions in real time [5].

METHODS

The capture module designed and implemented works in backend on a server node that guarantees the realization of the capture process of multiple IP video streams minimizing its execution time considering the technical characteristics of the IP cameras that provide the data sources. IP cameras capture live video and audio signals that are transmitted over the network for processing and display, using TCP/IP and UDP protocols [6].

Each instance of video stream processing is associated a priori to a processing core, resulting in establishing a proportionality to the number of streams to be processed with respect to the number of processing cores available per node. The hardware architecture of the host node may vary according to the number of CPU cores, its clock frequency, the amount of RAM memory available, the external storage capacity of the disks and the data transfer rates. It cannot be overlooked that the access times to each of these devices is an important aspect to achieve the proposed objective.

The two factors that determine the effectiveness of the capture process are closely linked to the efficient use of the computational performance of each of the processing cores of the CPUs of the computational nodes and the optimization of the use of the hardware architecture of the network adapters involved in the capture. Therefore, the

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

estimation of the computational cost of the algorithm contributes to determine in what proportion the capture process should be optimized with a view to achieve its execution under time constrained conditions [7].

The designed capture software module is designed to operate continuously 24/7/365, using mainly the shared memory programming paradigm, which enables the organic storage of video frames in an area in RAM memory accessible to all CPU cores, thus facilitating access to the information contained in the frames for processing by other modules. It is important to point out that capturing video streams requires at least one connection with the same bandwidth as the transmission rate of the service, which must be processed by optimizing the computational performance of the processing nodes.

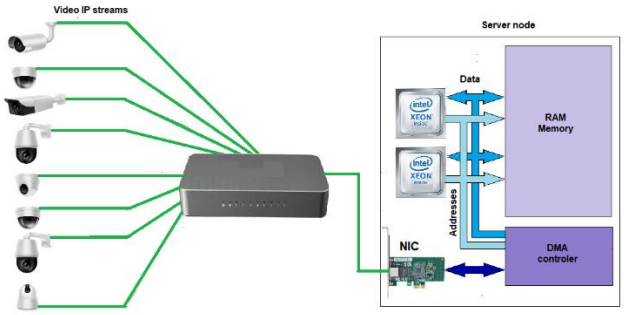

Figure 2. Capturing module of multiples video streams.

The design also allows access to individual frames in RAM for more efficient performance of other visual analysis operations associated with other processing modules. As the processing cores have a higher clock frequency, a higher number of operations per second can be performed. The video capture module has been implemented using C++ language. It creates N instances of the VideoCapture object, which make it possible to manage a certain number of IP video streams coming from heterogeneous cameras. As can be seen in Fig. 2, a description of the capture process is made on a server with 2 processors and 8 processing cores.

To achieve the operation of the capture process based on the shared memory paradigm, the OpenMP parallel programming library is used, which allows the processing cores to share a common memory area allowing the simultaneous management of the various video streams coming from each of the capture devices.

The execution of the videocapture instances are also extensible for execution on other computational nodes based on the distributed memory paradigm through the MPI message passing library. This functionality makes it possible to increase the number of cameras to be managed indeterminately according to the computational performance of these nodes.

Management of 8 cameras with a single server with 2 CPUs and 4 processing cores each, has been implemented. Each of the cameras have associated successive index 0,1,2,3...7. This module work in continuous form and it is able to realize the video streams transfer that transit through the network card using Direct Memory Access (DMA) with the purpose to reduce the processing density of the processing cores. By means of DMA is possible to fill out the buffers with the received data of the capturing process that are assigned during the initialization phase and these are released for further processing.

DMA is an alternative method for input/output port communication that is essential in this design to allow devices of different processing speeds to communicate with each other, preventing each CPU from receiving a massive interrupt overhead. It is important to take into consideration that most recent network adapters (NICs) have the ability to use DMA ensuring that the copying of memory blocks is done directly from the network adapter to the RAM and also vice versa. In this case the functions of the capture module manage only the input for the construction of frame lists in RAM from the network device without CPU intervention.

The DMA controller (DMAC) is a specialized circuit or a dedicated microprocessor that functions as a bus controller simultaneously with the CPU and ensures that data does not pass through the CPU [8]. In practical terms by means

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

of DMA transfer memory blocks are moved from an external memory to a faster internal one by employing the system buses (data and address bus). For the implementation of the solution a strategy has been employed that helps to avoid the bus being held captive only by the DMAC controller.

In this design strategy the DMA implements the x86 architecture with several DMA channels, each of which is used by a specific device. Each DMA address is assigned to a capture device to avoid conflicts. The data structure of the NICs is reused using a circular buffer called Buffer Ring that contributes to the optimization of the DMA process. It is worth noting that the Linux operating system kernel has included since version 2.5 the NAPI (new API) method. Interrupts are not generated by received packets, instead, packets are stored in RAM and the operating system kernel periodically polls them to retrieve them. This method also helps to improve performance in cases where high traffic rates occur.

The module has been developed on Linux operating system, specifically being validated on Ubuntu 20.04.4 LTS (Focal Fossa) and Ubuntu 22.04.4 LTS (Jammy Jellyfish) distributions. It has been configured on virtual machines on a private cloud platform that can be extensible based on the software as a service (SaaS) model. The method described below performs the process of configuring the operating system kernel to enable the management of traffic received through the network adapter.

The kernel enables NAPI in the network interface driver, so that they are disabled in high traffic circumstances. The network driver must be compiled to include the NAPI function that is supported by the operating system kernel.

Incoming packets are placed in the FIFO RX of the NIC. When this fills up it transfers them directly to a memory slot by means of DMA. This action frees the CPU from transfer execution. The space reserved in RAM memory is called DMA buffer, or buffer ring. Packets arriving at the NIC are stored in the buffer ring.

At this point NAPI generates a list of packets in a Poll list (poll_list) and immediately issues a notification to the operating system kernel through the softirq_pending register, also called interrupt handler schedules, which is assigned to each CPU core for the purpose of notifying that the corresponding packets received from the DMA buffer can now be collected.

Using the ksoftirqd algorithm, the kernel periodically checks for any pending softirqs in the softirq_pending register (which NAPI recorded in the previous operation). When a softirq is found, the kernel goes to the DMA buffer to retrieve the packets, and transfers them to the TCP/IP stack. The kernel ksoftirqd processes are executed on each CPU in the system and are logged during startup.

When there is no more pending work, the NAPI subsystem is disabled and the device IRQs are re-enabled prior to the completion of the execution of this process.

The process repeats continuously from step 2.

The kernel libraries that are used for the configuration of this procedure are:

netdevice.h

busy_poll.h

When a network device is activated in the operating system with the ifconfig command, the function associated with the ndo_open field of the net_device_ops structure is called. The kernel library responsible for this task is: netdevice.h which has a specific location in the directory structure corresponding to the version of the operating system in use.

- The ndo_open function performs the following tasks.
- Enable NAPI.
- (netif_napi_add) allocate memory to the RX and TX queue.
- Register an interrupt handler.
- Enable hardware interrupts.

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

In the execution of this task the device driver must indicate to the operating system the memory region that will be used by the network adapter. Once the RAM region has been reserved, the hardware is informed of its physical location. From then on the incoming data will be written to the reserved memory location from where the module will collect and process it [9].

Before starting the process, it is required to identify the make and model of the network adapter for which the command is used:

```
#lscpi | egrep -i --color 'network|ethernet'
```

The determination of the network adapter model can also be checked by reviewing the datasheet. In order to ensure that the operating system kernel has as much time as possible for the evacuation of all packets from RAM, it is recommended to set the maximum possible buffer size of the card in the computational nodes that have to process several video streams, so as to avoid data loss due to overwriting.

To reserve the required space in RAM for the network adapter, the ethtool tool is used to adjust the size of the buffer ring, up to the maximum capacity allowed by the NIC. For example, with the Dell PowerEdge T310 server broadcom 5709 PCI-E Gigabit Ethernet G218C [XX] NIC model, a dual port network adapter is available, which has a transfer rate of 300 mbps. The extended configuration space determines the maximum size of the buffer ring and its setting range is between 256 to 4096.

It is desirable for the deployment of the module to reserve the maximum possible buffer size for the NIC, which in this case is 4096 bytes. For its configuration, the command is executed as follows:

```
#ethtool -G eth0 rx 4096 tx 4096
```

It is important to consider that the use of the DMAC driver may have associated cache coherency problems [10] due to failure to update the correct cache data during processing. If the CPU is using external memory. In that case the workaround alternative has been to flush the cache lines before starting outgoing DMA transfers [11].

On the other hand, to avoid packet loss, use has been made of side scaling reception (RSS) which is also known as Multi-Queue Receive (Multi-Queue Receive). This method of network controller allows the most effective distribution of the reception processing performed in this case through DMA, from the network making use of several CPUs in a multiprocessor system [12]. Using this method, the network controller processes the received data by employing an interrupt service function of the miniport controller that schedules a deferred procedure call (DPC).

DPC is an operating system interrupt handling mechanism to enable controllers to reference during the execution of certain processes. They are executed with a lower priority than normal interrupts. A DPC indicates all data received within the DPV call. Therefore, all receive processing associated with the interrupt is executed on the CPU where the receive interrupt occurs [13].

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

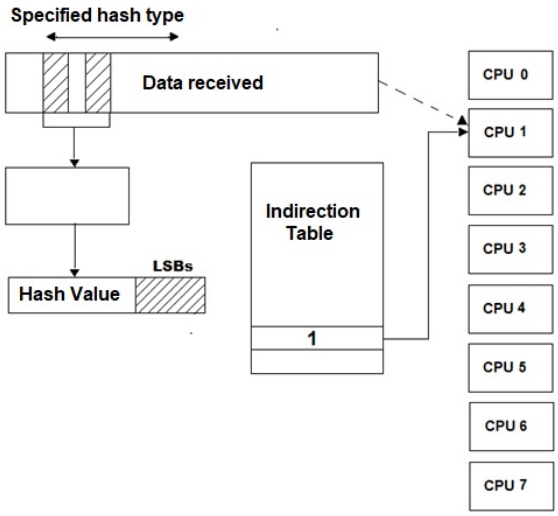

Figure 3. Allocation of data packets received from the capturing devices.

RSS enables the NIC and the miniport controller to schedule the reception of DPCs on other processors. The RSS design ensures that the processing of a specific network connection, in this case associated with a video stream coming from a specific IP camera, remains associated with a specific processing core (CPU) previously assigned. The NIC implements a hash function and the resulting hash value helps to select a specific CPU or processing core.

The NIC employs a hash function to compute a hash value in a defined area within the received data [14]. The defined area may not be contiguous, however, the least significant bits (LSBs) of the hash value are used to index the indirect addressing table. The values in this table are used to assign the received data to a specific CPU. In support of Message Signaled Interrupt (MSI) a NIC can also interrupt the associated CPU.

FIGURE 3. describes how the process of allocating the data received from the memory buffer to the corresponding CPU takes place. Although the generic term CPU is used, reference is made in practice to processing cores considering the architecture that has been implemented, which was previously described.

There are three levels of hardware support for RSS [15]. In the design of the capture module, the hash method with multiple receive queues was employed in which the NIC maps the received data buffers to the queues associated with the CPU in charge of monitoring the received packets using a 32-bit hash value. When the receive scale setting (RSS) is enabled, all processing of receive data for a given TCP connection is executed on the corresponding processor. Connections are shared among multiple processors or processing cores to achieve more efficient use of cache memory by sharing the processing load.

RSS contributes to the improvement of system performance by minimizing delays by distributing the NIC receive processing among multiple CPUs. The distribution of receive processing helps to ensure load balancing associated with this process. RSS provides distributed processing of receive indications from a specific NIC in DPC to multiple CPUs to achieve these performance improvements in a secure environment. In-order processing preserves the sequence in which received packets are delivered for each network connection and has a specific device associated with it that is managed from the capture module.

Pseudo-code that performs the general description of the capture module functions

Configure the characteristics of the capture device (network adapter) to ensure data transfer by DMA and reserve the RAM memory area allocated to this process. The resolution of the acquisition devices is based on configurable parameters according to their technical characteristics. The value of the number of cameras is a configurable parameter and is associated with the number of capture devices.

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Create N RAM indexed lists of the video frame sequence associated to the capture devices. The lists constitute temporary storage buffers. List 1 is associated to capture device 1 (cam 1) and so on up to list N which is associated to capture device N (cam N). These lists will contain the video sequences.

The processing of each list is dynamically assigned to a specific processing core; although they could also be processed by any available processor if required. The module has been configured so that any processing core can access the frames of any of the frame lists as long as it has "free time" for execution. Dynamic access to the lists is handled by the task queue manager. Launch other pre-processing or processing modules where required for the purpose of parallel processing of RAM-accessible frames prior to display.

Continuously store pre-processed video streams using various formats that guarantee access to other off-line multimedia content processing modules if necessary. Supports writing operations in formats such as mp4, mov, avi, among others. These formats can then be played back using GStreamer achieving automatic transcoding of multimedia content between the supported codecs, which include among others the following: VP8, H.264, H.263, AMR, OPUS, Speex, G.711.

Close the open windows of the display once the capture process is completed and release the reserved virtual windows from the video memory that was associated with each of the capture devices, in cases where the execution of other real-time processing modules is not required.

The capture module attempts to offload the CPU from the data transfer process to take over primarily the processing of the capture. This method contributes to the improvement of the overall performance of the system by allowing of the number of video streams to be processed by means of two alternatives:

Adding computing nodes with a greater number of processing cores (vertical scaling).

Increasing the number of interconnected nodes in a high-performance computing infrastructure (horizontal scaling).

The capture module can be managed from local or remote sites, in cases where remote accessibility from any geographical location is required, access to the computer infrastructure via IP must be guaranteed. The deployment process of the module must be preceded by an analysis for the prevention of vulnerabilities and intrusions that conspire against the security functions of the system from the remaining technological resources of the system, including cameras.

RESULTS AND DISCUSSION

With the development of the module, it has been possible to optimize the use of the hardware architecture in an efficient way in terms of capturing multiple IP video streams in real time. This has a favorable impact on obtaining a transversal solution for different visual analysis processes in terms of decision making and constitutes an optimization of the first phase of any computer vision system.

FIGURE 4 shows an analysis of the utilization of the processing cores of each of the processing cores by the capture module and illustrates the utilization percentage of the processing cores by the module that performs the lossless capture process, using only between 22% and 29% of the computational capacity of each of the processing cores involved in the capture process.

The variation in the percentage of utilization is mainly associated with the resolution of the IP cameras that perform the capture because they deliver frames with specific resolutions and also with the number of frames per second (fps) that are transferred.

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

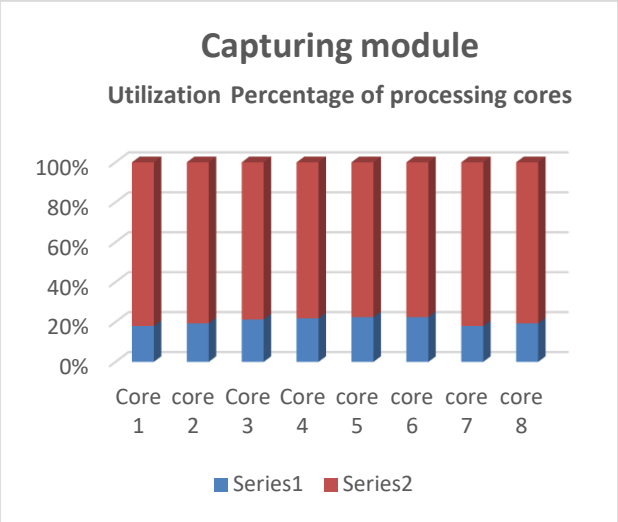

Figure 4. Utilization percentage of processing cores by the capture module.

It is important to point out that the capture module contributes to optimize the efficiency of the hardware architecture that supports it for different purposes. With this design, as can be seen, approximately 70% of the computational capacity of each of the processing cores is available for reuse by other pre-processing and processing modules. Its scalability is essential to articulate computer vision systems that must manage an undetermined number of IP cameras on which complex operations can be performed.

CONCLUSIONS

The module for capturing multiple IP video streams in real time is based on a parallel design that optimizes the use of the hardware architectures of the network adapters and the computational nodes that support them.

This module is highly effective for managing concurrent real-time video streams and optimizes the efficient use of available computational resources. It has been demonstrated that a single server node with 2 processors and 4 processing cores can capture and manage 8 IP video streams coming from heterogeneous capture devices through a network adapter.

The process of capturing 8 concurrent IP video streams uses between 22% and 29% of each of the processing cores. In this case, approximately 70% of the computational capacity of each processing core is available for the execution of other modules for pre-processing and processing of the video stream content according to organizational demands.

The capture module constitutes the first phase of any computer vision system and in turn provides the data sources for processing by other modules responsible for solving heterogeneous problems based on visual analysis for decision making in various organizational contexts.

The module is scalable and generates capacities to increase the number of flows to be processed as a higher number of cores and processing nodes are available when an HPC computing infrastructure is used.

This process has transversal applicability in application domains that require making use of visual analysis for decision making. It contributes to obtaining a better cost-performance ratio due to the efficient use of the hardware architectures that support it.

REFERENCES

[1] Mayur Akewar, Video Processing Aproaches: A review, TechRxiv, 2024
[2] Wenhao Wang, Zhenbing Liu, Haoxiang Lu, Rushi Lan, Zhaoyuan Zhang, "Real-Time Video Super-Resolution with Spatio-Temporal Modelling and Redundancy Aware Inference", 2023

Journal of Information Systems Engineering and Management
2025, 10(38s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[3] Shelby Lockhart, Amanda Bienz, William D. Gropp, Luke N Olson, "Characterizing the performance of node-aware strategies for irregular point-to-point communication on heterogeneous architectures, Parallel Computing", 2023
[4] O A. Hernandez Duany, Caridad Anias Calderón, "A model to build High Performance Infrastructures for Telecommunications organizations on cloud computing", 2024
[5] Guo, M., Xu, T., Liu, J., Liu, Z., Jiang, P., Mu, T. Zhang, S., Martin, R.R., Cheng, M., Hu, S., "Attention mecanisms in computer vision: a survey". Comput. Vis Media 8(3), 2022.
[6] "Considerations when designing and implementing video over IP", white paper, Harman, 2018.
[7] "Screen and Video Capture Software Market Trends", Share, Global Industry Outlook, 2023
[8] Ding Li, Weije Zhang, Mianxiong Dong, Kaoru Ota, "DMA-Assisted I/O for Persistend Memory", IEEE Transactions on Parallel and Distributed Systems, 2024.
[9] Paolo Pazzaglia, "Optimal Memory Allocation and Scheduling for DMA Data Transfer under the LET Paradigm", IEEE, 2021.
[10] Akur Changela, "Direct Memory Access (DMA) Controller", JETIR, 2018.
[11] Altaf Ahmed, Abdullah A, Aljumah, M Gulam Ahmad, "Design and Implementation of a Direct Memory Access Controller for Embedded Applications, International Journal of Technology", 2019.
[12] "Guides-Configuring Multi-Queue NICs, Akamai", 2023.
[13] Arubhav Choudhary, "Introduction to receive side scaling (RSS)", Medium, 2019.
[14] F Martin Fernández, P Caballero-Gil, Analysis of the new standard hash function, 2022.
[15] Lei Yan, Yueyan Pan, Diyu Zhou, Sanidhya Kashyap, "RSS++ algorithm to reassign load between cores. Problem dimension, 2023.