Int. j. inf. tecnol. (February 2021) 13(1):239–250
https://doi.org/10.1007/s41870-020-00483-z

ORIGINAL RESEARCH

VCSP: virtual CPU scheduling for post-copy live migration of virtual machines

Narges Jalaei[1,2] · **Faramarz Safi-Esfahani**[1,2]

Received: 7 December 2019 / Accepted: 19 May 2020 / Published online: 26 May 2020

Abstract The live migration of virtual machines among physical machines aims at efficient utilization of resources, load balancing, maintenance, energy management, fault tolerance, sharing resources and mobile computing. There are several methods for the live migration of virtual machines. In the post-copy approach, a virtual machine starts working in a target host, when data that is not present in the target machine is requested, a page fault occurs and the processing stops in the target until the arrival of the requested information. Certainly, the number of stops and lags negatively affects the system downtime. The reduction of physical or virtual CPU frequency is one of the techniques recommended to efficient migration of virtual machines. The modification of the frequency of physical and/or virtual CPU has already been used in the improvement of the pre-copy method to manage the speed of changes in the source machine. This paper presents the virtual CPU scheduling for post-copy (VSCP) framework that reduces the speed of the virtual CPU in post-copy to make a balance between processing speed of pages in the target machine and their transmission speed in the source machine. The VCSP is also compared with the baseline methods of the post-copy-prefetching and post-copy. In the experiments, the system downtime, total migration time, total pages transferred and throughput of system are evaluated. The results indicate that the proposed method improves the system downtime up to 8.17%, total migration time up to 30.33%, total pages transferred up to 54.49% and throughput of system up to 23.65%.

Keywords Live migration · Virtual machine · Post-copy · CPU frequency

1 Introduction

Live virtual machine migration is one of the achievements of virtualization that provides the possibility of a virtual machine displacement from one physical machine to another physical machine with the aim of server maintenance, load balancing and turning off the working machine, efficient utilization of resources, power management, fault tolerance, resource sharing and mobile computing [1–3].

The post-copy [4] is one of the migration techniques that combines pre-copy and lazy-copy algorithms [5]. In this method, the virtual machine is stopped in source and the CPU state is transferred to the target physical machine while the memory status is still in source physical machine. The virtual machine starts its operation in the target physical machine and processes the memory pages. When the data that are not present in the memory are not available, a page fault happens. The requested page is transferred from the source machine to the target machine. In the Post-copy method, the high speed of the target machine's processor makes the page faults occur at a higher speed. When a page fault occurs, the processor remains idle until the arrival of the requested page that results in the following problems: first, the downtime status of the system is longer; second, the migration takes more time; third, the rate of the transferred data increases and the

✉ Faramarz Safi-Esfahani
fsafi@iaun.ac.ir

Narges Jalaei
n.jalaei@sco.iaun.ac.ir

[1] Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran

[2] Big Data Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran

system throughput decreases. The total migration time, downtime, total pages transferred and throughput are the key criteria in the quality of the Post-copy live migration [4, 6, 7].

There are various methods for virtual machine migration and one of the methods for optimization of live virtual machine migration is regulating the physical or virtual machine CPU frequency. As far as the processor has a higher speed than other sectors such as memory, storage, etc., this difference in speed influences the migration process. The decrease of CPU frequency leads to a relative balance between the processing speed of pages and their transmission speed. By reducing the speed of the physical or virtual processor, the pre-copy [8] method balances the creation speed of dirty pages and reduces both downtime and total migration time. It is worth noting that in physical CPU scheduling, the decrease of the CPU speed affects all operations of the physical machine. However, in virtual CPU scheduling, the virtual machine performance is originated from the physical processor [6, 9, 10].

In this paper, the framework of virtual CPU scheduling for post-copy (VSCP) is recommended for the optimization of the post-copy method. The main idea is to decrease virtual CPU speed and try to make a relative balance between the processing speed of pages in the target machine and their transmission speed from the source machine. Reduction of virtual CPU speed leads to the transmission of more pages with lower page fault and consequently reduction of downtime, total migration time and lower transmitted data and improvement of the system efficiency. For the implementation of this idea, the kernel-based virtual machine (KVM) hypervisor [11] is used for virtualization and Quick EMUlator (QEMU) emulator [12] is used to control virtual CPU scheduling. The experiments consider the specifications of the physical machine, workload rate, network bandwidth, number of virtual machines and allocated resources to the virtual machine and the efficiency parameters including system downtime, total migration time, total pages transferred and throughput of the system have been evaluated.

This paper is organized as follow: Sect. 2 represents the concepts and related works to the applications of real and virtual CPU scheduling in virtual machine migration. Sect. 3 deals with VCSP proposed method, Sect. 4 describes experimental environment, setup and evaluations, and finally in Sect. 5 conclusion and recommendations for future works are explained.

2 Concepts and related works

2.1 Virtual machine migration

Virtual machine migration is the status change of a virtual device such as memory, CPU, I/O devices between physical machines during the migration process. There are two kinds of migrations: live migration and non-live migration. In live migration, the source machine participates in the migration process and the migration happens such that the source machine continues its activity and after the migration is done, we turn off the source machine; however, in non-live migration, first the source machine activity will be stopped and then migration will occur.

The virtual migration operation from one physical machine to another for system maintenance provides the possibility of load balancing and turning off the working machine. The aim of this operation is the efficient utilization of resources, power management, fault tolerance, resource sharing, and mobile computing. The local manager examines the system status and orders the global manager to migrate several virtual machines if required. In response, the global manager sends a message for the migration of the virtual machine to the hypervisor [1–3]. The live migration achievements include energy management, fault tolerance, resource sharing, system maintenance, load balancing, and mobile computing applied in research activities such as [13–26].

2.2 The post-copy live migration method

Post-copy is one of the migration algorithms based on a combination of the pre-copy and lazy-copy algorithms [5]. The function of this method is such that the virtual machine is stopped in source and the system status is transferred to the target physical machine; this is while the memory status is still in source physical machine. The virtual machine starts its activity in the target physical machine. When the processor reaches a missed page in the target machine, a page fault happens and then the memory page is demanded from source physical machine and transferred to the target physical machine, subsequently [2]. The pre-paging, active pushing, demand paging, and prefetching techniques are used for the transmission of pages. Demand paging is the simplest and slowest technique and functions in this way that the active page will be transferred from the source physical to target physical machine, then, virtual machine refers to required pages in target physical machine, when the referred page doesn't exist in target physical machine, page fault happens and it transfers the related page from source physical machine to target physical machine. By default, post- copy method transmits the pages in this way.

The prefetching technique is a page transmission technique for post-copy which classifies and sort data in a package including N memory pages for transfer. In this method, by any request, N adjacent pages will be selected and transferred to the target machine. Although the time spent for prefetching pages is considerable in the real environment with a high volume of data, using the prefetching technique is considered as a good idea. The active pushing technique is a predictive technique where along with the requested page; a few adjacent pages are also transferred to make less page fault. The pre-paging is also a predictive technique however smarter, i.e. it guesses what pages are more probable to be referred to in the future and transfers those pages. The other technique is dynamic self-ballooning (DBS) which is not for the post-copy method and has application in other methods. This technique is for the prevention of transferring free pages to which nothing is allocated, since the transferring of these free pages is a useless task and leads to traffic in the transmission system and unnecessary implementation in target physical machine [4, 7, 27].

In this paper, both demand paging and prefetching techniques have been used to page transfer according to the pseudo-codes Algorithms 1 and 2.

Algorithm1. post-copy migration algorithm[4]

1. p← pages of VM Memory
2. Transfer CPU to destination host
3. Resume VM on destination host
4. B← Array[] (Bitmap in the destination)
5. If B_i==0 then page fault
6. While B_i==0
7. Request p from the source
8. Transfer p to destination host
9. End while

Algorithm2. post-copy-prefetching migration algorithm[4]

1. p← pages of VM Memory
2. A← Array[] (Bitmap in the source)
3. Transfer CPU to destination host
4. Resume VM on destination host
5. B← Array[] (Bitmap in the destination)
6. If B_i==0 then page fault
7. While B_i==0
8. Request p from the source
9. For (i==0 ; i<N ; i++)
10. If A_i==1 then p← A_i
11. Transfer p to destination host
12. End while

transmission speed, the migration would be failed. By changing physical or virtual CPU speed, this method balances the formation speed of dirty pages in the pre-copy-based methods. Therefore, the downtime of the system and total migration time is considerably decreased. In the methods based on the lazy-copy technique, if the processing speed of the virtual machine's pages is higher than their transmission, then the number of page faults would be increased. The scheduling method can improve the downtime, and the total migration time by improving the number of page faults. Figure 1 shows the mind-map of both pre-copy and post-copy live migration techniques based on CPU scheduling, where the mainstream of this research has been highlighted that focuses on the post-copy.

The authors in [9] proposed the FMC which is a combined method for the improvement of the pre-copy method. This is a self-adaptive method that performs the transmission operation between both source and target physical machines by tracking the input/log file and its response instead of changed memory pages. In the first stage, the method is divided into providing log files and replaying them in the target. When the bandwidth is very narrow, the CPU is scheduled to reduce the size of input files. The

2.3 Related works to both virtual and real CPU scheduling in the live migration of virtual machines

Any type of transferring and moving data and code is considered as migration [28]. The scheduling of physical or virtual machines improves the live virtual machine migration. As far as CPU has a higher speed than memory, storage, and other devices, this difference in speed may result in several problems. If the speed of dirty page formation or producing log files is higher than their

experiments have shown that the FMC reduces migration overhead in a quick bandwidth compared to the pre-copy method. Moreover, in low bandwidth, FMC has shorter downtime than cr/tr. Moreover, in a narrow bandwidth, the overhead has an acceptable result. The advantages of this self-adaptive method are the high speed, low overhead, reduced CPU speed of the virtual machine in certain conditions, short total migration time and short downtime. The disadvantages of this method are the release of replay in the standby time of the virtual machine and being confined to local spaces.

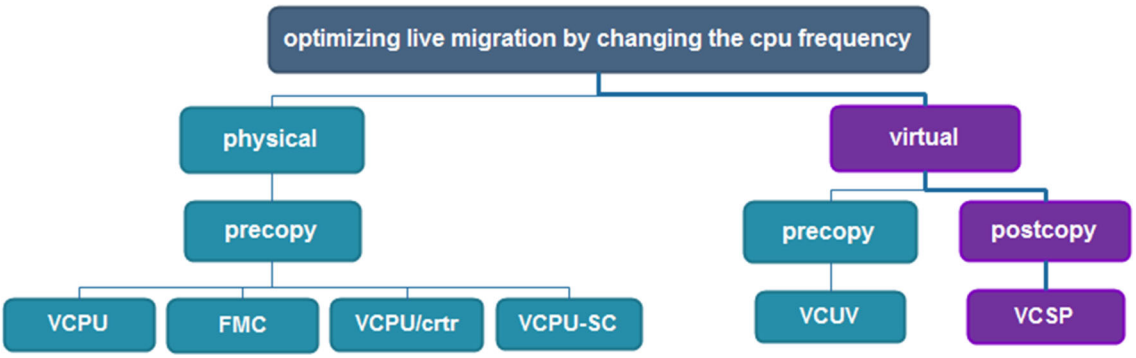

Fig. 1 The mind-map of both pre-copy and post-copy live migration techniques based on CPU scheduling

The researchers in [10] proposed the VCPU method to improve the pre-copy method. The base of this algorithm is the reduction of frequency of source physical machine CPU that by reducing the processing speed of virtual machine decreases the formation speed of dirty pages, thus, the formation speed of dirty pages would be closer to the transmission speed of pages and in this way, migration procedure would improve. The advantages of this method are good performance in a confined situation, optimum accessibility range, short downtime, and avoidance of software change. The disadvantages of this method are a failure in supporting parameters such as writable tasks, lack of control over bandwidth and failure in controlling all programs.

The research work [6] presented a method to improve the cr/tr method that reduces the formation speed of log files by reducing the frequency of physical machine CPU which consequently leads to a decrease in downtime and total migration time of virtual machine compared to the pre-copy method. The advantages of this method are high efficiency, high tolerance process, high speed, transparent migration, acceptable total migration time and short downtime. The disadvantages of this method include inefficiency in nonhomogeneous and complex environments such as multiprocessors.

The research article [29] has presented a method that reduces the total migration time of the virtual machine and downtime of the system by presenting an optimum scheduling method for the physical machine CPU and the pre-copy method. Moreover, this algorithm leads to energy saving. The credit of one of the used scheduling algorithms is XEN hypervisor which functions based on weight and capacity parameters. The capacity is the allocated percentage of CPU to any virtual machine and weight is allocated schedule to any virtual machine. The main idea of this paper is that it makes the formation speed of dirty pages desirable by creating a reduction in the percentage rate allocated to the virtual machine (cap). The advantages of this method include decreased downtime and total migration time, a considerable reduction of transmitted pages, energy-saving, increased computation speed and capability in the virtualization of hardware resources. The disadvantage of this method is the uncertainty of optimum efficiency and good implementation.

The authors in [20] presented a method for the improvement of the traditional pre-copy algorithm. By reducing the frequency of virtual machine CPU, this method tries to make a balance between the speed of CPU and transmission of pages between the network and therefore improvement of virtual machine migration. In this method, the virtual machines of source servers that do not require migration will continue their operation without any disturbance in their function. The advantages of this method are reduced downtime and total migration time, a considerable decrease in the transferred pages and balancing the speed of CPU and transmitted pages in the network. The disadvantages of this method are failure to support some parameters such as writable tasks and lack of control over the bandwidth of the network.

Table 1 shows the comparison of the most related works with the proposed method in this current research to show their difference in terms of the hypervisor, the way of changing CPU frequency, the type of data exchange, and the baseline technique to improve.

3 Virtual CPU scheduling for post-copy (VCSP) proposed method

Figure 2 presents the framework of live migration through virtual CPU scheduling for the post-copy method that includes components of the source and target machines. The buffer is a virtual memory composed of several VM pages. A virtual CPU (VCPU) is allocated to any virtual machine that is the share of virtual machines from the physical CPU. The memory management unit (MMU) is responsible for providing the data requested from the CPU. The bit map is an array that shows the presence status of

Table 1 Classification of live migration optimization techniques based on CPU scheduling

Migration technique	Hypervisor	Frequency change type	Transition data type	Effective item	Improved technique
FMC [1]	XEN	Physical CPU	Logfile	Dirty pages	PRE-COPY CR/TR
VCPU [2]	XEN	Physical CPU	Page	Dirty pages	PRE-COPY
VCPU CR/TR [3]	XEN	Physical CPU	Logfile	Dirty pages	CR/TR
VCPUV [4]	XEN	Virtual CPU	Page	Dirty pages	PRE-COPY
VCPU-SC [5]	XEN	Physical CPU	Page	Dirty pages	PRE-COPY
POST-COPY [6]	XEN	–	Page	Page fault	–
VCSP (our approach)	KVM	Virtual CPU	Page	Page fault	POST-COPY

memory pages in source and target machines with 0 and 1. The array component with value 1 shows that the requested page is available and whenever the corresponding bit of the page in the bit map is 0, it indicates the page fault and a request for transferring the page from the source to the target will be issued that is due to the page fault. Whenever the requested page is transferred from the source to the target, the value of the corresponding bit of the page changes from zero to 1. The data exchange points between the source and target machines are called the Stub connected through the network. The stub is responsible for receiving requests in target and transferring them to source machine and in the same way, receiving the pages sent from the source to the target machine in response to the requests received. The migration agent is responsible for retrieving the requested page from the virtual memory of the source machine based on the requested page number. In the proposed structure, both VCSP and VCSP-prefetching algorithms are considered for the migration control algorithm (MCA). In fact, the migration method specifies how is the response given by the source machine to the page request of the target machine and how many pages should be transferred to the target machine.

3.1 The virtual CPU scheduling for post-copy (VCSP) method

The main idea of the VCSP framework is the reduction of CPU speed of the target virtual machine and making a relative balance between both processing speed and page transmission speed. In the post-copy approach, page faults are the most effective reason in reducing the quality of migration. Reducing the processing speed results in

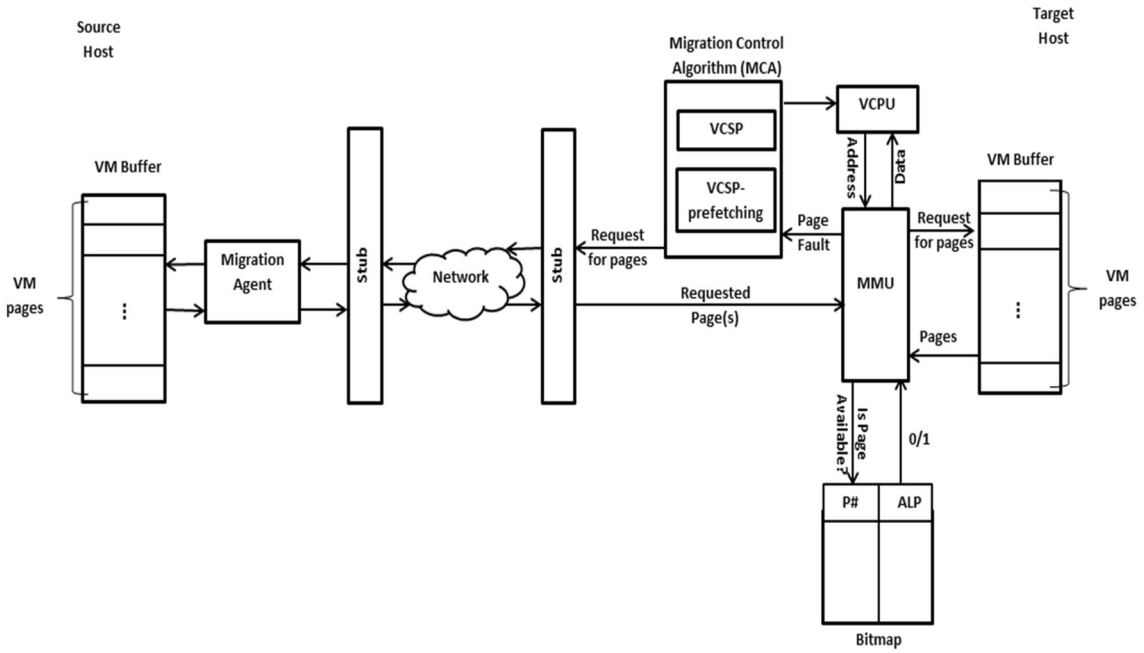

Fig. 2 The framework of virtual CPU scheduling for post-copy (VSCP)

reducing the number of page faults that is equal to improving the migration.

First, the CPU status is transferred to the target machine and the virtual machine starts working in the target host. The speed of the virtual CPU in the target machine would decrease to half by using this idea. The basis for reducing CPU speed is the obtained feedback from the runtime environment such as the number of fault pages, system efficiency, response time, output, or a combination of them. In VCSP, CPU sends its requests to the memory management unit (MMU) that is responsible for providing the data (pages) required by the CPU. The bit map is checked and if the requested page is available in the target machine, it is processed; however, whenever, the requested page does not exist in the target buffer, a page fault occurs and the processing is stopped in destination until the arrival of the requested page. The requested data are present in the source machine; thus, a request is sent to stub and the memory pages are requested from the source machine. In the source machine, the requested page is taken from memory and sent to stub and after transmission, they are processed in the target machine. This process continues until completing the transmission of pages from the source to the target machine. It is noteworthy that post-copy transfers pages by the demand-paging technique as default. Algorithm 3 shows the pseudo-code related to the VCSP method.

Algorithm3. VCSP migration algorithm

```
1.  p← pages of VM Memory
2.  Transfer CPU to destination host
3.  Schedule E% of the CPU time to execute VM
4.  Resume VM on destination host
5.  B← Array[] (Bitmap in the destination)
6.  If Bi==0 then page fault
7.  While Bi==0
8.          Request p from the source
9.          Transfer p to destination host
10. End while
```

3.2 The VCSP-prefetching method

The VCSP-prefetching is an extension of the VCSP method that transfers pages through a prefetching technique. So far in demand-paging technique, one page is transferred by any request, the decrease in the speed of the processor leads to an increase in total migration time and network traffic. A prefetching technique is proposed for overcoming this problem. In the prefetching method, one n-page package is transferred by any request Therefore, more pages are fetched in the destination host that reduces the traffic of the network. Applying the VCSP-prefetching and reducing the processing speed, results in improving the transmission time of pages. Algorithm 4 shows the pseudo-code related to the VCSP-prefetching method.

Algorithm4. VCSP-prefetching migration algorithm

```
1.  p← pages of VM Memory
2.  A← Array[] (Bitmap in the source)
3.  Transfer CPU to destination host
4.  Schedule E% of the CPU time to execute VM
5.  Resume VM on destination host
6.  B← Array[] (Bitmap in the destination)
7.  If Bi==0 then page fault
8.  While Bi==0
9.          Request p from the source
10.         For (i==0 ; i<N ; i++)
11.             If Ai==1 then p← Ai
12.         Transfer p to destination host
13. End while
```

4 Experimental environment, setup, and evaluations

4.1 Experimental environment

The experimental environment includes two hosts with 8G RAM, Intel xeon e5-2650, 8 cores 2.00 GHz CPU and 1 GB/s bandwidth. The size of each VM is 2 GB as well. KVM is installed on the hosts to support virtualization; however, KVM does not support CPU scheduling and applies the Linux scheduler. Therefore, QEMU ver. 2.5 is used for scheduling the virtual CPUs. The installed operating system on the virtual machine is Debian 7.8 arm/arch and on the physical machine is Ubuntu 14.04. Moreover, the size of each page is 4 kb, and each time 10 pages are fetched as well. The tests are performed on various workloads 8, 16, 32, …, 512 (MB) and the implementation results of the post-copy, post-copy-prefetching, VCSP, and VCSP-prefetching methods are then compared.

In order to implement the experiments, a stress test program applied in [4] is used that begins with a low workload and is increased consecutively. This change of workload is controlled by a working window log file that determines an interval of data volume for processing. In this part, VCSP-prefetching, VCSP, post-copy-prefetching, and post-copy methods are implemented in the test

condition and they are compared and evaluated in terms of downtime of the system, total migration time, total transferred pages, and throughput of the system.

In this paper, KVM open source virtualization along with Linux for virtual CPU scheduling is used. If the KVM scheduler is used to reduce the virtual machine CPU speed, Linux codes should inevitably be changed. It raises two issues: first, there is no priority control in Linux, and second, by changing the source of Linux, a new Linux is produced that should be installed and is analogous to an unpleasant situation for users. To resolve the issues, the QEMU emulator [12] that is based on a full virtualization approach is used. The QEMU schedules virtual machines and allows Linux to implement the post-copy algorithm while the speed of the virtual processor is decreased. The Telnet connects to the QEMU and issues commands. The migration command by default uses the pre-copy method. Thus, both post-copy and VSCP extension codes should be added to both source and destination to determine that the post-copy algorithm should migrate.

In post-copy, first, a bit map is created for all pages and the transferable pages are specified. Then, the CPU context that includes the content of registers is transferred. Immediately, after the transmission, CPU starts its work in the target machine. If during the processing, a page is requested that is not transferred yet, then, the first-page fault will happen and change the bit map that is initialized by zero at the beginning. Any page that is transferred, its corresponding value changes from zero to one. The bit map is continuously checked and the zero value indicates a page fault. This procedure continues until completing the page transfer.

In the source machine, first, a buffer is defined that plays the role of memory. Until the completion of the migration, any page fault that occurs in the target machine this memory is looked for the requested page. This buffer is associated with physical memory through the prefetching method such that it fetches the requested page to physical memory, then packs and sends the page to the stub. After completion of the migration process, the buffer is released. In prefetching technique, a bit map, to illustrate the adjacent pages will be checked and among zero and ones that indicate the transmitted and non-transmitted pages, those components with the bits related to value 1 will be prepared for transmission in form of a package of N members and those bits of zero value will be rejected. Although rejecting zero bits leads to a decrease in speed for a high volume of information especially at the end of operation when most pages are transmitted, prefetching is a good idea.

In destination host, the processor of virtual machine accesses to a bit map. In this part, a quasi MMU is defined because QEMU is an emulator and does not simulate the real software, completely. While CPU sends its request to MMU, it checks the bitmap, if the demanded page is number 1 in the bitmap, it receives the page and processes it; otherwise, a page fault occurs. There is a component called page fault detector inside MMU that in case of fault page, creates a request including the page number and sends to stub that is responsible for receiving and sending requests. Stubs are associated with each other in both machines.

4.2 Experimental setup and evaluations

In this paper, the evaluation criteria are downtime, total migration time, transmitted data, and system efficiency parameters. In the post-copy method, downtime includes those times when CPU remains idle for achieving data. This time is calculated in terms of milliseconds. Total migration time is the time it takes for the migration process to be completed. In the post-copy method, total migration time is from the moment when the CPU status is transferred to the target machine to the time when the last page is transferred from memory and is then calculated in terms of the second. The parameter total page transferred, is the amount of data (KB) transferred from the source machine to the target machine during migration. In the post-copy method, this interesting point is that the memory pages are transferred through how many requests. The throughput of the system is the rate of transferred data in the time unit and calculated in terms of MB/s. In the post-copy method, downtime negatively affects the throughput of the system. Table 2 shows the configuration of tests during experiments.

4.2.1 Experiment 1: The evaluation of downtime

This experiment aims at examining the system downtime in the VCSP-prefetching method compared to the post-copy method. The occurrence of a page fault results in an idle CPU until the demanded page becomes available. Therefore, the number of page faults greatly influences system downtime. According to results in Fig. 3, system downtime increases when the workload increases. In the post-copy, increasing the workload is analogous to increasing the number of requests for fetching pages that negatively increases the downtime of the system. In VCSP, downtime is increased however, it is decreased when the workload increases. In the post-copy, increasing the workload is analogous to increasing the number of requests for fetching pages that negatively increases the downtime of the system. After increasing the number of requests to access pages, the reduction of CPU speed results in decreasing the number of page faults. When the prefetching technique is used, the time spent for prefetching pages is added to system downtime. According to Fig. 3, in workload 256 and

Table 2 Configuration of tests

Experiment no.	Evaluated variables	Comparing algorithms	Methods	Workload	Transition data rate per request	CPU frequency (%)
1	Down time	Post-copy	Demand paging	8, 16, 32, …, 512	One page	100
		Post-copy	Prefetching	8, 16, 32, …, 512	Ten pages	100
		VCSP	Demand paging	8, 16, 32, …, 512	More than one page	50
		VCSP	Prefetching	8, 16, 32, …, 512	More than 10 pages	50
2	Total migration time	Post-copy	Demand paging	8, 16, 32, …, 512	One page	100
		Post-copy	Prefetching	8, 16, 32, …, 512	Ten pages	100
		VCSP	Demand paging	8, 16, 32, …, 512	More than one page	50
		VCSP	Prefetching	8, 16, 32, …, 512	More than ten pages	50
3	Total pages transferred	Post-copy	Demand paging	8, 16, 32, …, 512	One page	100
		Post-copy	Prefetching	8, 16, 32, …, 512	Ten pages	100
		VCSP	Demand paging	8, 16, 32, …, 512	More than one page	50
		VCSP	Prefetching	8, 16, 32, …, 512	More than ten pages	50
4	Throughput	Post-copy	Demand paging	8, 16, 32, …, 512	One page	100
		VCSP	Prefetching	8, 16, 32, …, 512	More than ten pages	50

higher, most diagrams are ascending which shows that when the workload is low, prefetching time is more acceptable than the number of page faults. Although in vitro condition and with the low workload, the proposed VCSP method considerably improves system downtime when pages are transferred by prefetching technique; however, these results are not valid for the real condition with the high workload. As seen in Fig. 3, by increasing the workload, the VCSP downtime is decreased. The system downtime through VCSP-prefetching is improved by 8.17% compared to the post-copy method.

4.2.2 Experiment 2: The evaluation of total migration time

This experiment aims at examining the total migration time in the VCSP-prefetching method compared to the post-copy method. According to Fig. 4, total migration time for the post-copy, VCSP, post-copy-prefetching and VCSP-prefetching methods in various workloads are calculated. After the test, despite the expectation, it becomes clear that the VCSP proposed method has longer total migration time than the post-copy method. This is the reason the waiting time (that itself is the result of reducing CPU speed) is augmented to the total migration time; thus, the prefetch operation is added to the post-copy for removing the problem of total migration time of the proposed method. Prefetching has a considerable effect on reducing total migration time in the post-copy status since more data are transferred by any request. In this condition, if processing speed is decreased, the reduced number of requests is effective in improving total migration time. In fact, the VCSP-prefetching method has obtained much better results in terms of total migration time. This improvement and optimization become more highlighted in high workload. Accordingly, total migration time in the VCSP-prefetching method is improved by up to 30.33% compared to the post-copy method.

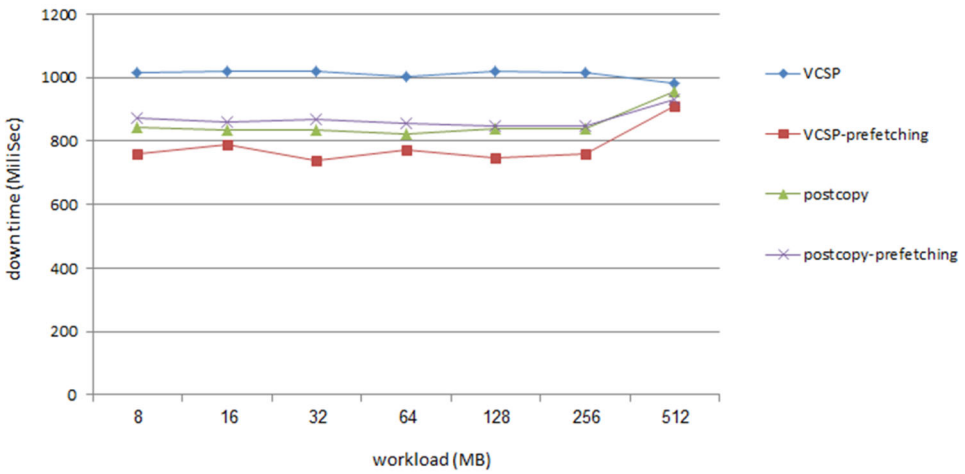

Fig. 3 Downtime of the main algorithm and VCSP proposed method with various workloads

4.2.3 *Experiment 3: The evaluation of total pages transferred*

This test aims to examine the rate of transferred data in the VCSP-prefetching method compared to the post-copy method. According to Fig. 5, with an increased workload, the total pages transferred will increase. In the post-copy method, by total pages transferred, it is meant counting page faults and page crowd in the transmission network. Based on this, with the decrease of CPU frequency, this data rate increases greatly since although the number of requests is less than the post-copy method, the crowd of pages in the network increases due to decreased speed of processing. When the prefetching technique is used for transferring pages since the size of each page is considered 4 KB, the buffer could include more pages and transfers more pages, with the decrease of CPU frequency, the number of requests also decreases and thus, the total pages transferred decreases. The results of the test show that in the VCSP-prefetching technique, especially when the workload increases, the total pages transferred decrease. Thus, according to the tests, the total number of transferred paged is improved in VCSP-prefetching up to 54.49% compared to the post-copy method.

4.2.4 *Experiment 4: The evaluation of system efficiency*

This test aims to examine the throughput of the system of the VCSP-prefetching method compared to the post-copy method. The throughput of the system is tested in 0.1 s periods. According to the obtained results as shown in Fig. 6, when migration is done through the VCSP method the throughput of the system considerably improves when the pages are transferred through the prefetching technique. During system downtime, the throughput of the system considerably decreases. When the page fault occurs, the system has downtime and as seen in Fig. 6, the throughput of the system at these points is reached to its minimum. During 40–44 s, the VCSP-prefetching method has 4 s fall in throughput of the system due to continuous page faults

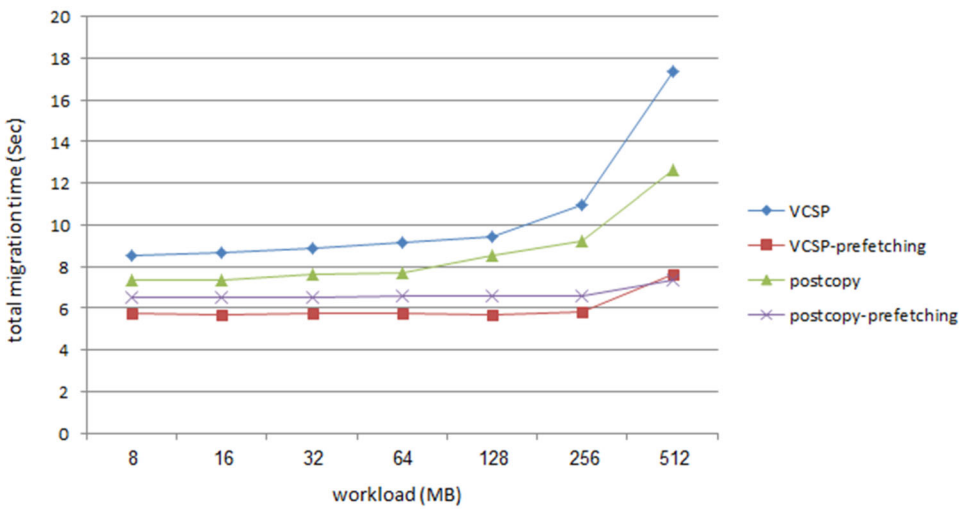

Fig. 4 Total migration time for post-copy and VCSP with various workloads

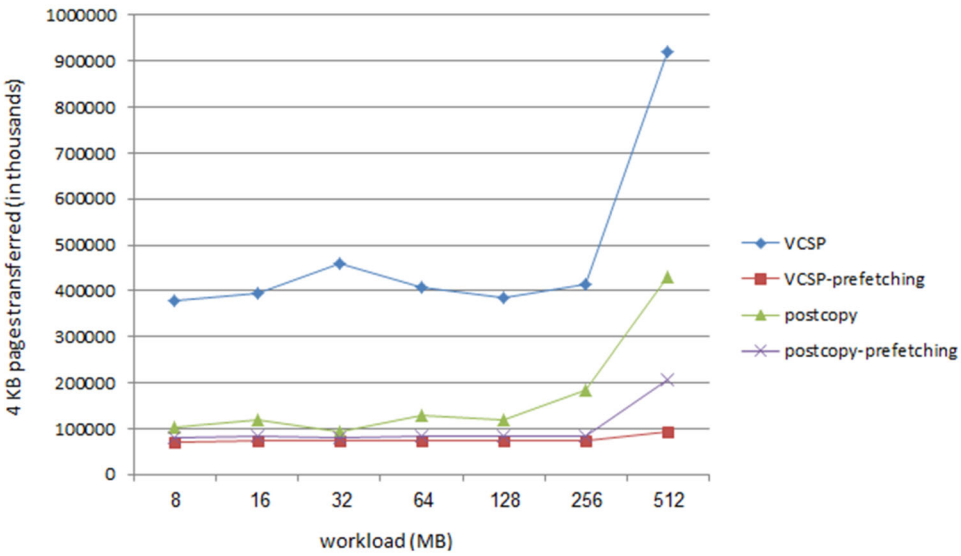

Fig. 5 The rate of transferred data for the main algorithm and proposed VCSP method by various workload

and in sum, it has 4.7 s downtime and the throughput of the system is 6.7 s for the post-copy method. The results show that even at system downtime, the VCSP-prefetching method has better throughput. According to the tests, the throughput of the system has improved in the VCSP-prefetching method up to 23.65% compared to the post-copy method.

4.3 Experimental discussion

Table 3 shows the comparison results of the migration methods. Each number shows the amount of improvement or downgrading of the migration methods in comparison to the post-copy. After the evaluation and comparison of the post-copy, VCSP, post-copy-prefetching, and VCSP-prefetching methods in terms of downtime, total migration time, total pages transferred and throughput of the system, it could be concluded that although the decreased CPU speed is effective to balance CPU speed and memory, decreasing processing speed is a negative action. Moreover, since the demand-paging technique transfers one page per request, the negative effect of decreasing the frequency of CPU is more than its positive effect. The results show that the prefetching technique is more effective for transferring pages in the post-copy technique. As can be seen in the Figures, VCSP-prefetching method has considerably improved all quality criteria. This optimization is more highlighted in total migration time with 30.33% improvement and total pages transferred with 54.49% improvement; moreover, it has considerable effect in throughput of system with 23.65% improvement. As this idea has been successful in improving the pre-copy method, it has optimized the post-copy method up to 29.16%.

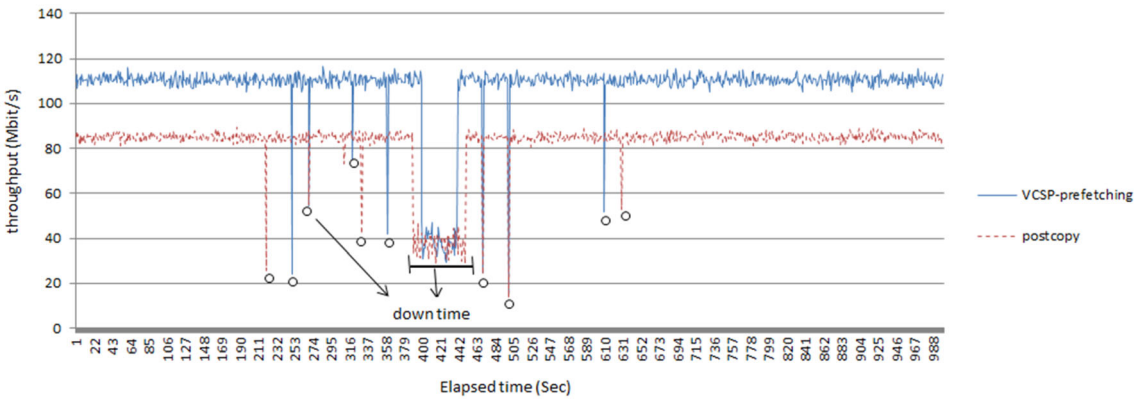

Fig. 6 The comparison of system efficiency for the main algorithm and proposed VCSP method

Table 3 The comparison of tests' results

Comparing algorithms vs. post-copy	Downtifme (%)	Total migration time (%)	Total page transferred (%)	Throughput (%)
Post-copy-prefetching	− 2	23	41	4
VCSP	− 16	− 17	− 65	10
VCSP-prefetching	8.17	30.33	54.49	23.65

5 Conclusion and recommendations for future works

Virtual CPU scheduling has been recommended for improving the post-copy classic method in terms of system efficiency. In the post-copy method, in case of page fault occurrence, the processing was stopped until the arrival of the requested page. High processing speed results in changing the memory frequently and leads to the increased number of page faults and thus increased system downtime. VCSP method reduces the speed of virtual machine CPU to make a relative balance between processing speed of pages and their transmission speed so that more pages will be transferred with lower fault page, thus, quality and efficiency factors improved in the post-copy method. However, it is worth noting that reducing the frequency of CPU is not suitable for all methods of transferring pages in the post-copy. Therefore, the prefetching technique was used to decrease the number of requests for pages that results in decreasing CPU frequency and improving efficiency. The experiments evaluated the post-copy, VCSP, post-copy-prefetching, and VCSP-prefetching methods in terms of system downtime, total migration time, total pages transmitted and throughput of the system. The results indicate that VCSP-prefetching has the best result compared to other methods such that compared to the post-copy, it has improved system downtime up to 8.17%, total migration time up to 30.33%, total pages transferred up to 54.49% and throughput of system up to 23.65%. According to evaluations, the post-copy method has improved up to 29.16%.

Several research activities are recommended for future works: (1) The application of queuing theory and technique can be studied in managing pages. (2) The application of the pre-paging technique for transmission of pages can also be investigated. (3) The predictive selection of pages may improve the efficiency of the post-copy. (4) The study of how CPU scheduling could improve the post-copy method can be another direction. (5) The application of the self-ballooning technique to study the effect of transmitting free pages on the efficiency of the method.

References

1. Ahmad RW, Gani A, Hamid SHA, Shiraz M, Xia F, Madani SA (2015) Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues. J Supercomput 71(7):2473–2515. https://doi.org/10.1007/s11227-015-1400-5
2. Kaur S, Pandey PV (2015) A survey of virtual machine migration techniques in cloud computing. Comput Eng Intell Syst 6(7):28–34
3. Shah SAR, Jaikar AH, Noh S-Y (2015) A performance analysis of precopy, postcopy and hybrid live VM migration algorithms in scientific cloud computing environment. In: 2015 international conference on high performance computing & simulation (HPCS), pp 229–236
4. Hines MR, Deshpande U, Gopalan K (2009) Post-copy live migration of virtual machines. ACM SIGOPS Oper Syst Rev 43(3):14–26
5. Zarrabi A (2012) A generic process migration algorithm. Int J Distrib Parallel Syst 3(5):29
6. Liu W, Fan T (2011) Live migration of virtual machine based on recovering system and CPU scheduling. In: 2011 6th IEEE joint international information technology and artificial intelligence conference, vol 1, pp 303–307
7. Sahni S, Varma V (2012) A hybrid approach to live migration of virtual machines. In 2012 IEEE international conference on cloud computing in emerging markets (CCEM), pp 1–5
8. Patel PD, Karamta M, Bhavsar MD, Potdar MB (2014) Live virtual machine migration techniques in cloud computing: a survey. Int J Comput Appl 86(16):18–21
9. Zhao J, Hu L, Xu G, Chang D, Ding Y, Fu X (2013) A fast live migration algorithm of virtual machine with CPU scheduling. In: Proceedings of the international conference on grid, cloud, and cluster computing (GCC), p 115
10. Jin H, Gao W, Wu S, Shi X, Wu X, Zhou F (2011) Optimizing the live migration of virtual machine by CPU scheduling. J Netw Comput Appl 34(4):1088–1096
11. Kivity A, Kamay Y, Laor D, Lublin U, Liguori A (2007) KVM: the Linux virtual machine monitor. In: Proceedings of the Linux symposium, Canada, pp 225–230
12. Bellard F (2005) QEMU, a fast and portable dynamic translator. In: USENIX annual technical conference, FREENIX Track, vol 41, p 46
13. Alaei N, Safi-Esfahani F (2017) RePro-active: a reactive–proactive scheduling method based on simulation in cloud computing. J Supercomput. https://doi.org/10.1007/s11227-017-2161-0
14. Motavaselalhagh F, Safi Esfahani F, Arabnia HR (2015) Knowledge-based adaptable scheduler for SaaS providers in cloud computing. Hum Centric Comput Inf Sci. https://doi.org/10.1186/s13673-015-0031-4
15. Salimian L, Safi Esfahani F, Nadimi-Shahraki M-H (2016) An adaptive fuzzy threshold-based approach for energy and performance efficient consolidation of virtual machines. Computing. https://doi.org/10.1007/s00607-015-0474-5

16. Khorsand R, Safi-Esfahani F, Nematbakhsh N, Mohsenzade M (2017) ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering run-time circumstances in cloud computing environments. J Supercomput. https://doi.org/10.1007/s11227-016-1928-z
17. Fadaei Tehrani A, Safi-Esfahani F (2017) A threshold sensitive failure prediction method using support vector machine. Multiagent Grid Syst. https://doi.org/10.3233/MGS-170263
18. Haratian P, Safi-Esfahani F, Salimian L, Nabiollahi A (2017) An adaptive and fuzzy resource management approach in cloud computing. IEEE Trans Cloud Comput. https://doi.org/10.1109/TCC.2017.2735406
19. Meshkati J, Safi-Esfahani F (2019) Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing. J Supercomput 75(5):2455–2496
20. Amiri PAD, Rad SZ, Isfahani FS (2016) Providing a solution to improve pre-copy method for migrating virtual machines in cloud infrastructure. J Theor Appl Inf Technol 92(2):225–234
21. Kamalinasab S, Safi-Esfahani F, Shahbazi M (2019) CRFF. GP: cloud runtime formulation framework based on genetic programming. J Supercomput 75(7):3882–3916
22. Hemasian-Etefagh F, Safi-Esfahani F (2019) Dynamic scheduling applying new population grouping of whales meta-heuristic in cloud computing. J Supercomput 75(10):6386–6450
23. Torabi S, Safi-Esfahani F (2018) A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. J Supercomput 74(6):2581–2626
24. Salimian L, Safi F (2013) Survey of energy efficient data centers in cloud computing. In: Proceedings of the 2013 IEEE/ACM 6th international conference on utility and cloud computing, pp 369–374
25. Salimian L, Safi-Esfahani F (2018) Energy-efficient placement of virtual machines in cloud data centres based on fuzzy decision making. Int J Grid Util Comput 9(4):367–384
26. Shojaei K, Safi-Esfahani F, Ayat S (2018) VMDFS: virtual machine dynamic frequency scaling framework in cloud computing. J Supercomput 74(11):5944–5979
27. Kapil D, Pilli ES, Joshi RC (2013) Live virtual machine migration techniques: survey and research challenges. In 2013 3rd IEEE international advance computing conference (IACC), pp 963–969
28. Ibtihal M, Hassan N (2020) Homomorphic encryption as a service for outsourced images in mobile cloud computing environment. In Cryptography: breakthroughs in research and practice, IGI Global, pp 316–330
29. Amani A, Zamanifar K (2014) Improving the time of live migration virtual machine by optimized algorithm scheduler credit. In: 2014 4th international conference on computer and knowledge engineering (ICCKE), pp 346–351