doi:10.3772/j. issn. 1006-6748.2021.04.002

Research on optimization of virtual machine memory access based on NUMA architecture①

He Mujun (何牧君)②*, Zheng Linjiang*, Yang Kai**, Liu Runfeng**, Liu Weining*
(* College of Computer Science, Chongqing University, Chongqing 400044, P. R. China)
(** Dawning Information Industry Co., Ltd., Beijing 100084, P. R. China)

Abstract

With the rapid development of big data and artificial intelligence (AI), the cloud platform architecture system is constantly developing, optimizing, and improving. As such, new applications, like deep computing and high-performance computing, require enhanced computing power. To meet this requirement, a non-uniform memory access (NUMA) configuration method is proposed for the cloud computing system according to the affinity, adaptability, and availability of the NUMA architecture processor platform. The proposed method is verified based on the test environment of a domestic central processing unit (CPU).

Key words: cloud computing, virtualization, non-uniform memory access (NUMA) virtual machine, memory access optimization

0 Introduction

The advent of the artificial intelligence (AI) big data era has caused the scale of data centers to grow exponentially. Moreover, deep computing and high-performance computing have placed more stringent requirements on the performance of computer systems. Besides the transformation of the Internet Data Center (IDC) architecture, server architecture and accelerated computing components are also changing. With the development of this architecture, the virtualization platforms need to be designed in a way so that it is compatible with the characteristics of the processor's architecture, instruction set, clock frequency, and cache. Currently, mainstream processors all adopt a non-uniform memory access (NUMA) structure[1-7].

Based on the problems mentioned above, this work conducted extensive research to identify potential solutions. The paper is organized as follows. First, in view of virtualization scenarios, the motivation of virtual machine memory access optimization is introduced, then the key issues are analyzed. Subsequently, a series of virtual machine memory access optimization methods based on NUMA architecture are proposed, followed by discussion of its design and implementation[8-12]. Finally, the effect of the proposed optimization method is verified and analyzed.

1 Virtual machine memory access optimization technology of NUMA

1.1 NUMA architecture of modern processor

Since the 1980s, processing speeds have considerably increased. As a result, in the 1980s and 1990s, supercomputers were designed to provide high-speed memory access rather than faster processors, enabled computers to process large datasets at speeds that other systems could not do. Therefore, to reduce the cache miss rate, the processor designs a larger cache at that time. However, the size of the operating system and its applications also increased rapidly, making the improvement of cache processing much more difficult[13-15].

Around the same time, NUMA appears as a memory design structure for multi-core multiprocessors, where the memory access time depends on the memory location relative to the processor. In NUMA, a processor can access its own local memory faster than non-local memory (local memory of another processor), while for non-local memory, accessing the nearest neighbor is faster than the far end[16-17].

As shown in Fig. 1, the relationship between NUMA nodes of a multiprocessor can be described via geometric relationships, in which the number of nodes in the system is limited to 2^C, and C is the number of neighbor nodes owned by each other[18].

① Supported by the National Key Research and Development Program of China (No. 2017YFC0212100) and National High-tech R&D Program of China (No. 2015AA015308).
② To whom correspondence should be addressed. E-mail: tbabm. he@gmail. com
Received on Oct. 20, 2020

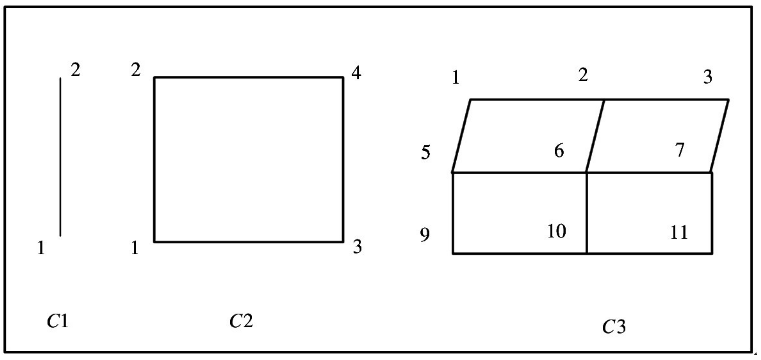


Fig. 1 Geometric diagram of node relationship

Taking $C = 3$ as an example, nodes 6, 2, 5, 7, and 10 are neighbor nodes, and 1, 3, 9, and 11 are remote nodes. In the current NUMA structure of the processor, node accesses to the memory overhead follow the order, local node, neighbor node, and remote node.

The NUMA hardware architecture system is shown in Fig. 2, which introduces the relationship between NUMA nodes. Despite the various designs for NUMA by different manufacturers, the principle stays the same.

In Fig. 2, the NUMA structure solves the forementioned memory access performance problem caused by cache miss by providing separate memory for each processor. On one hand, this prevents multiple processors from trying to address the same memory, which may reduce the performance. On the other hand, it also ensures the maximum design of the access memory bandwidth. Fig. 3 presents an example of a NUMA processor with 8 NUMA nodes in total. Among them, the access delay of the same node is the smallest, which is about 85 ns, followed by neighboring nodes of about 140 ns, and the remote node of about 240 ns as the largest.

In practical application, using NUMA structure can improve the memory access performance by more than double for the application type of data distributed (usually running on the server)[19].

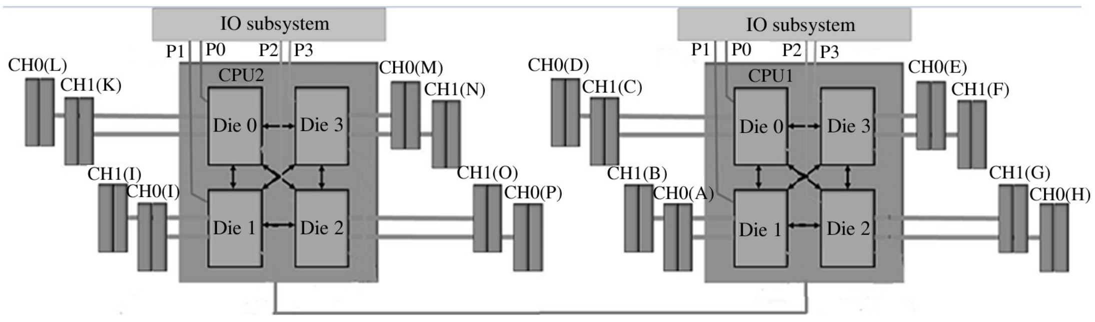
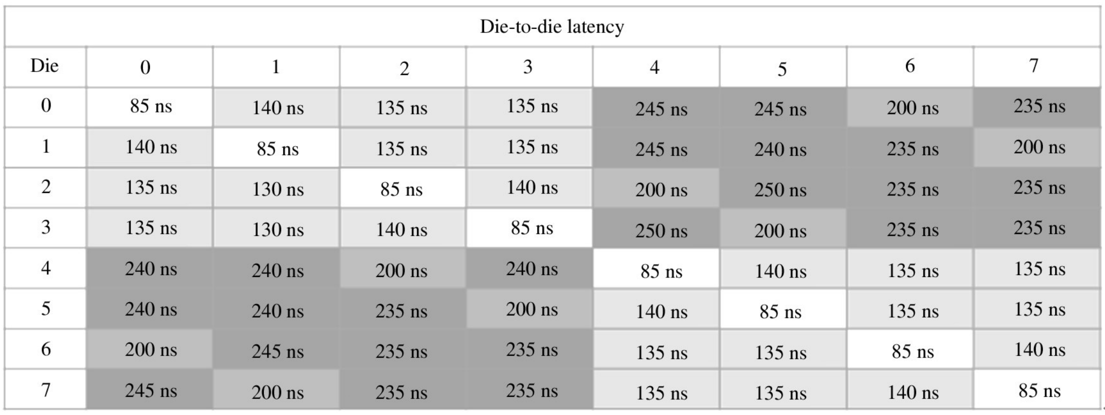


Fig. 2 NUMA hardware architecture



Die-to-die latency								
Die	0	1	2	3	4	5	6	7
0	85 ns	140 ns	135 ns	135 ns	245 ns	245 ns	200 ns	235 ns
1	140 ns	85 ns	135 ns	135 ns	245 ns	240 ns	235 ns	200 ns
2	135 ns	130 ns	85 ns	140 ns	200 ns	250 ns	235 ns	235 ns
3	135 ns	130 ns	140 ns	85 ns	250 ns	200 ns	235 ns	235 ns
4	240 ns	240 ns	200 ns	240 ns	85 ns	140 ns	135 ns	135 ns
5	240 ns	240 ns	235 ns	200 ns	140 ns	85 ns	135 ns	135 ns
6	200 ns	245 ns	235 ns	235 ns	135 ns	135 ns	85 ns	140 ns
7	245 ns	200 ns	235 ns	235 ns	135 ns	135 ns	140 ns	85 ns

Fig. 3 Communication delay between nodes of a NUMA processor

1.2 Virtual machine memory access optimization technology

This paper presents design directions to improve the memory access performance of virtual machine under NUMA architecture. These directions refer to the limitation of resources in terms of memory and processor[20-21].

By binding the memory and processor of a virtual machine, the cross-node memory access is avoided, and the performance and stability of memory access are improved. Furthermore, it provides a one-to-one binding of processor resources, which prevents the processor from switching within the node and improves the processor's unilateral computing power.

Due to the technical limitations of the above two levels, the virtual machine access bus communication by using this design exhibits a more stable, reliable, and overall better performance than other designs[22-23].

The research direction for further optimization is analyzed below. For the binding of memory and processor resources, the absolute availability of resources must be guaranteed. Therefore, a one-to-one resource usage principle is formulated, resulting in low resource utilization. If resources are bound, sufficient resources must be reserved and constrained by the principle of limited total resources, which is inevitable. When binding memory and processor resources are limited, the load of the virtual machine is unpredictable. Therefore, part of the load is high, and part of the load is low after resource binding, which will lead to the problem of unbalanced use of memory and the appearance of processor resources in the physical machine. To solve unbalanced resource utilization, a dedicated monitoring service module can be designed to regularly monitor the load of NUMA nodes with the physical machine. According to the threshold setting, the resource warning of computing node is triggered to prompt and record, and simultaneously, the switching service of the virtual machine memory and processor is designed to capture the resource alarm based on the monitoring service. In order to achieve the automatic balance of physical machine memory and processor resources in the computing node, the virtual machine of NUMA nodes triggers the processor switch and memory copy replication so as to improve the utilization of resources.

2 Key issues

In the cloud computing environment, abundant data intensive business occurs in the data center, such as artificial intelligence, deep learning, and high-performance computing. In order to improve the performance of such services, it is necessary to optimize the memory access bandwidth of the virtual machine where the service is located. This will face two main problems.

(1) Managing processor affinity. When the virtual machine runs on the host, it is inevitably affected by the host operating system. In the case of processor resource competition, the virtual machine process will increase the delay, while decreasing the memory access bandwidth. The direct result is that computing resource sensitive applications react slowly or will even pretend to be dead. To adapt to the memory architecture of the NUMA node, it is necessary to ensure that a process (virtual machine) on a given central processing unit (CPU) should run as long as possible on the host without being migrated to other processors.

(2) Adaptability of host and virtual machine memory access policy. In order to ensure the availability of resources, the remaining memory page and newly allocated memory page may not be on the same NUMA node during the virtual machine memory access process, leading to access of the virtual machine cross-node memory. However, simply assigning the NUMA node where the virtual machine is located cannot effectively solve this problem. For services in virtual machines, the access relationship between the newly allocated memory and allocated memory should be adapted to the host memory policy, otherwise, the service performance will be greatly affected.

3 Virtual machine memory access optimization scheme

Aiming at the main problems currently faced, this paper designs a set of virtual machine optimization schemes for multi-processor non-uniform memory access architecture hosts, which solves the problem of cross-node memory access through software. The main method is to limit the relationship between memory and the processor, so that the processor can access the memory and allocate it to the local node. Therefore, a technological breakthrough must be made from the allocation of memory and processor to solve this problem. After that, the performance and stability of memory access are guaranteed, and one-to-one allocation of virtual machine processors to host processors can be further performed to improve unilateral computing power, but the simple allocation is not unprincipled.

The basic idea of this method is as follows.

(1) The processor process binding of NUMA architecture is satisfied.

(2) Pre-allocate the virtual machine memory suit-

able for the host computer.

3.1 Satisfy the processor binding of NUMA architecture

The virtual machine runs on the host computer and is represented as a process, which is a presentation of the virtual machine on the physical machine. The process has its own independent memory space, and the virtual processor of the virtual machine appears as a thread of the process.

A virtual machine runs on the physical machine and is inevitably controlled by the operating system mechanism. First, when the processors of multiple virtual machines compete for the same physical processor resource, the acquisition process of time slice of the virtual machine will be limited by resources, inevitably causing certain delay. Second, there is a resource scheduling mechanism in the operating system, which will switch the virtual machine on the NUMA node with a higher load to the NUMA node with a lower load. When migration occurs, the access between processor and memory will cross NUMA nodes, resulting in an instantaneous drop in memory access bandwidth. For computing resource sensitive scenarios, the direct result is application performance degradation and even virtual machine crash. Therefore, additional mechanisms must be considered to ensure high computing performance of virtual machines. Since the virtual machine is a process, and the virtual processor is a thread in the process, redesigning the level of the physical machine operating system can solve the problem of processor switching.

In summary, to design a virtualized processor binding scheme with the purpose of improving the performance and stability of the virtualized system, three functions must be implemented, i. e. processor information collection, NUMA node range binding, and in-depth NUMA node processor one-to-one binding. The module calling relationship between them is shown in Fig. 4.

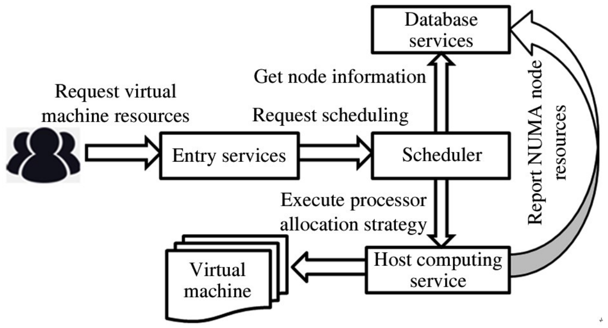


Fig. 4 Call relation of processor binding module

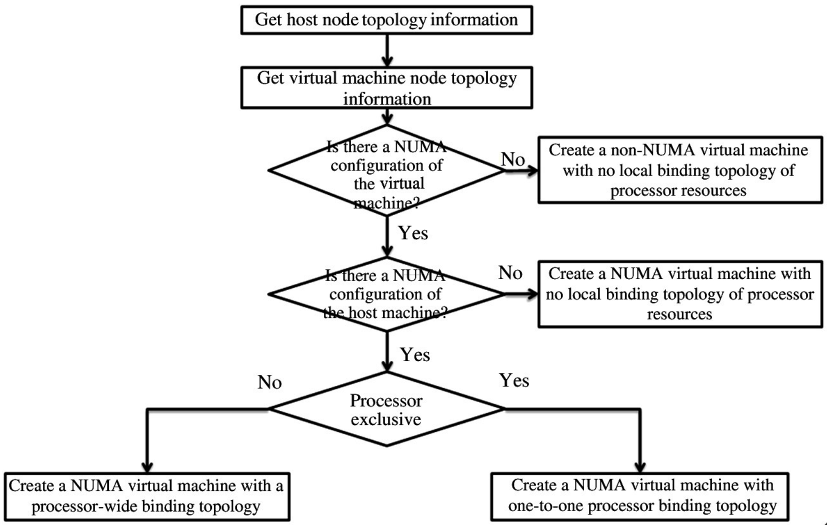
The process of processor resource binding can be described as four steps. The first step is to collect the underlying hardware information of the processor and save it to the database for regular update and synchronization. In the second step, the entrance service provides restful application programming interface (API) services, and users apply for the virtual machine through the portal service. The third is the scheduling service, which selects an appropriate physical machine to issue creation requests according to the latest NUMA node information of the physical machine and through the measurement of load and weight. The fourth step is the host computer computing service, which is mainly responsible for the execution of real virtual machine creation (including binding of processor resources).

When the NUMA node of the physical machine is selected, the instructions for creating virtual machine resources and binding will be sent to the physical machine. The target physical machine is bound to the node processor of the virtual machine using kernel-based virtual machine (KVM) virtual processor pinning technology. Using this technology, binding fixed node resources for the virtual machine processor is realized, where each virtual node is only bound to a physical node processor range for scheduling, avoiding node switching of the virtual processor and improving the memory access performance of the virtual machine. At the same time, one-to-one binding processor resources can be selected to avoid context switching caused by processor switching, reduce performance loss, and further increase the stability of computing power. Flow chart of NUMA node processor binding system is shown in Fig. 5.

The final goal of the solution is to bind the NUMA nodes and processor resources of the virtual machine. For computing sensitive industries, such as big data and AI, this technology is recommended to improve the memory access and computing capacity of virtual machines.

3.2 Pre-allocate the virtual machine memory suitable for the host computer

Memory virtualization is the process of converting virtual memory of a virtual machine into physical memory of a host, where the virtual machine still uses physical memory of the host. By adding an extended page table (EPT) register to the virtual machine, the page missing exception will be generated when the virtual machine has access to the page table and the address is empty. After the virtual layer catches the exception, it allocates physical addresses and establishes physical memory and virtual machine memory. During the next



Get host node topology information
Get virtual machine node topology information
Is there a NUMA configuration of the virtual machine?
No
Create a non-NUMA virtual machine with no local binding topology of processor resources
Yes
Is there a NUMA configuration of the host machine?
No
Create a NUMA virtual machine with no local binding topology of processor resources
Yes
No
Processor exclusive
Yes
Create a NUMA virtual machine with a processor-wide binding topology
Create a NUMA virtual machine with one-to-one processor binding topology

Fig. 5 Flow chart of NUMA node processor binding system

conversion, the direct query is converted according to the mapping record.

The processor and memory of a virtual machine are usually obtained from the same NUMA node. In order to ensure the availability of resources, the operating system will independently judge and select a virtual machine process to find alternative memory resources of other nodes to write data when resource competition occurs. It is inevitable that a certain amount of residual memory is retained in the old node, resulting in cross node memory access. Therefore, simply binding virtual machine nodes cannot effectively solve this problem, which requires synchronous allocation of node memory. This chapter mainly describes the memory orientation and priority allocation scheme of a virtual machine to avoid cross node memory access completely, or as far as possible, and improve the memory access performance and stability of the virtual machine. The memory allocation principle of a virtual machine is consistent with the kernel binding. The module call relationship of memory allocation is shown in Fig. 6.

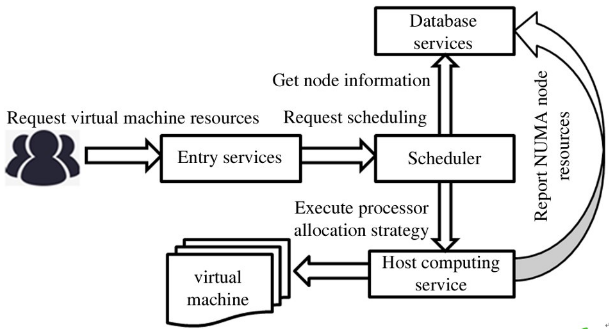
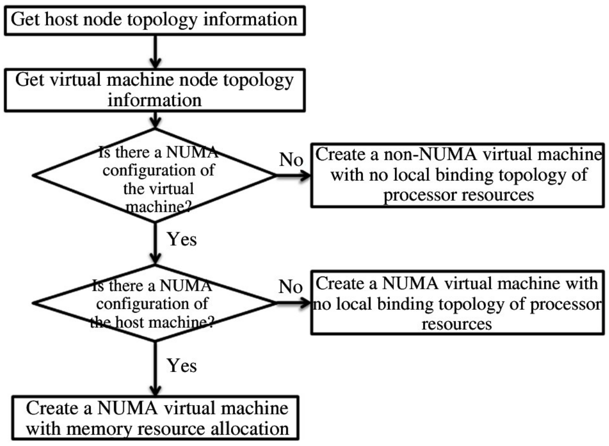


Fig. 6 Module call of node memory allocation system

The process of memory resource allocation can be described in four steps. The first is for the information collection system to collect the underlying hardware information of the processor and save it in the database, which is then updated and synchronized regularly. In the second, the entrance service provides restful API services, so that users can apply for the virtual machine. The third is the scheduling service, where a suitable physical machine is selected to issue the creation request according to the latest physical machine NUMA node and the measurement of load and weight. The fourth step is the host computing service, which is mainly responsible for the implementation of real virtual machine creation (including memory resource allocation). The memory allocation logic of computing services for virtual machines is shown in Fig. 7.



Get host node topology information
Get virtual machine node topology information
Is there a NUMA configuration of the virtual machine?
No
Create a non-NUMA virtual machine with no local binding topology of processor resources
Yes
Is there a NUMA configuration of the host machine?
No
Create a NUMA virtual machine with no local binding topology of processor resources
Yes
Create a NUMA virtual machine with memory resource allocation

Fig. 7 Flow chart of node memory allocation system

Based on the mechanism of the virtual machine as a process itself, the virtual memory allocation can be achieved by calling the process memory limit function of the self-control group. The principle of restriction is that the total memory on the physical node is greater than or equal to the total memory bound to the node. The memory of the virtual machine is strictly bound to the physical nodes corresponding to the virtual machine node. If a virtual machine restricts it to use only the memory of a NUMA node, it will not use the memory of other nodes when its node memory is exhausted and will start to use swap instead. If the machine is not set with swap, it will crash directly. Therefore, a compromised memory allocation strategy is adopted and the memory of nodes in the virtual machine is configured by using the principle of priority allocation. In single node memory competition, the memory of the corresponding physical nodes of other nodes in the virtual machine can be allocated, and there are two restrictions on the allocation of memory. The first one strictly limits the overall memory of the virtual machine that can only be obtained within the scope of some corresponding physical nodes. The second allows the necessary cross node allocation of the memory of NUMA nodes in the virtual machine but can only apply for resources within the range of nodes allowed by the overall memory of the virtual machine. This minimizes the occurrence of cross node memory access and ensures the high availability of resources.

Although super allocation fails using this strategy, it can ensure the high resource availability of the memory resources of the node to the virtual machine that allocates the memory and avoids memory competition among the virtual machines. When the physical NUMA node is short of memory, the memory resources cannot be obtained in the NUMA node normally nor can be switched after the allocation of memory resources. In view of swap and the virtual machine memory priority allocation strategy, the virtual machine memory will not starve to death, but will cause system failure of the virtual machine and a temporary decrease in memory access performance. However, in reality, this problem generally does not occur, but rather a few virtual machines are full of memory resources. There are realistically no services that consume a lot of resources on the host, thus meaning the virtual machine's memory resources will be sufficient.

The ultimate goal of the solution is to allocate or prioritize the memory resources of virtual machines. For industries that are sensitive to computing capabilities, such as big data, AI, etc., this technology is recommended to improve the memory access capabilities of virtual machines.

4 Experimental verification

The test environment and tools used in this work are listed in Table 1.

Table 1 Test environment and tools

Physical machine model	Number of processors	Number of NUMA node	Memory /GB	System version	Tool for testing memory access	Memory access test item	Tool for testing computing performance
Domestic CPU	32	4	128	centos 7.5	stream	copy	LMbench

4.1 Memory access performance verification

At present, there is no fixed binding between the memory and processor of normal virtual machines in the industry. The virtual machine memory and processor on the computing node are affected by the process scheduling of the operating system. In the case of virtual machine over-provisioning, it is easy to generate drift switching of NUMA nodes, causing the memory and processor of the virtual machine to be different NUMA nodes. As a result, the memory access performance will experience instant jitter and, thus, will be greatly reduced.

The virtual machine that binds the memory and processor resources uses the same NUMA node to calculate the memory and processor of the virtual machine on the NUMA node, which can effectively ensure the memory access performance of the virtual machine and prevent cross-NUMA node memory access.

This scheme tests the memory access conditions of cross-NUMA nodes, bound nodes, and memory. For applications with frequent service load jitter, this scheme provides the maximum performance optimization of data and tests the scheduling service bound by the memory and processor, so as to determine the effectiveness of resource protection of the scheduling service.

The binding method of the virtual machine NUMA node corresponding to the physical machine NUMA node is called node range binding, which corresponds to other two types of binding. The binding method of the virtual machine NUMA node memory corresponding to the physical machine NUMA node memory is called memory strict binding. Moreover, the priority binding mode of the virtual machine NUMA node memory corresponding to the physical machine NUMA node memory is called memory priority binding. These bindings are described in more detail as follows.

4.1.1 NUMA node range binding and memory strict binding

On the physical machine, the performance of memory bandwidth access rate of the virtual machine is improved by combining virtual machine node range binding and memory strict binding. The purpose is to test and attain the maximum memory bandwidth access rate, compare it with the maximum memory bandwidth of a non-memory and fixed processor bound virtual machine, then determine the performance optimization baseline data of memory bandwidth access rate. In this test, four test scenarios are set. In brief, Scenario 1 occurs after NUMA optimization, and Scenarios 2 – 4 simulate the real scenarios that customers may encounter for comparative testing.

Scenario 1 The memory accessed by the vCPU of virtual machine 1 is bound to nodeset = 2, by dividing NUMA nodes.

Scenario 2 The memory accessed by the vCPU of virtual machine 2 drifts from nodeset = 2 to the memory corresponding to nodeset = 1.

Scenario 3 The memory accessed by the vCPU of virtual machine 2 drifts from nodeset = 2 to the memory corresponding to nodeset = 3.

Scenario 4 The memory accessed by the vCPU of virtual machine 2 drifts from nodeset = 2 to the memory corresponding to nodeset = 4.

The strategy of virtual machine node range binding and memory strict binding ensures that the memory access location of a business virtual machine is local memory and that the occurrence of cross NUMA node memory access is eliminated to improve the memory access performance of the virtual machine. The key test indicators are affected by the following basic principles.

(1) The virtual machine that binds the processor and memory resources on the same NUMA node has no cross-NUMA node memory access, so the performance should be higher. In this case, it is called local node memory access.

(2) The virtual machine that binds the processor and memory resources to the neighboring NUMA node does not have memory access across NUMA nodes, so the performance should be normal. In this case, memory access is performed by neighbor node memory access.

(3) The virtual machine that binds the processor and memory resources to another neighboring NUMA node does not have memory access across NUMA nodes, so the performance should be normal, which is also called neighbor node memory access.

(4) The virtual machine that binds the processor and memory resources to the remote NUMA node does not have memory access across NUMA nodes, and the performance should be poor. In this case, it is called remote node accesses memory.

According to the test scenario, the results are described as follows. Memory access promotion rate can be calculated by Eq. (1).

$$Rate_{Promotion} = \frac{Speed_{binding} - Speed_{non\text{-}binding}}{Speed_{non\text{-}binding}} \quad (1)$$

where, $Rate_{Promotion}$ is memory access promotion rate, $Speed_{binding}$ is node range binding memory access rate, $Speed_{non\text{-}binding}$ is non-node fixed binding memory access rate.

Fig. 8 shows the comparison between Scenario 1 and Scenario 2. After testing, the local node memory access performance of Scenario 1 is better than that of the neighbor node access in Scenario 2, in terms of Copy, Scale, Add, and Triad. In the present study, the memory access bandwidth of Copy increased the most by nearly 40%, followed by Add and Triad with an increase of about 30%, then Scale with the least increase of nearly 20%.

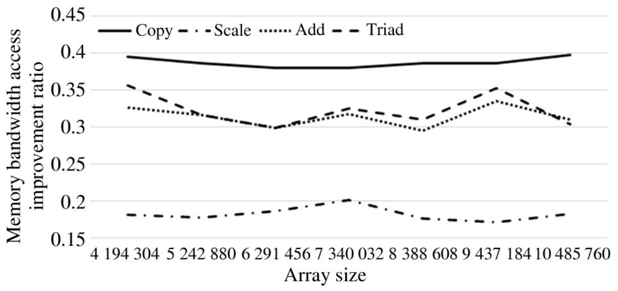


Fig. 8 Memory access comparison between Scenario 1 and Scenario 2

Fig. 9 shows the comparison between Scenario 1 and Scenario 3. The test tool can be used to demonstrate that the local node access performance of Scenario 1 is

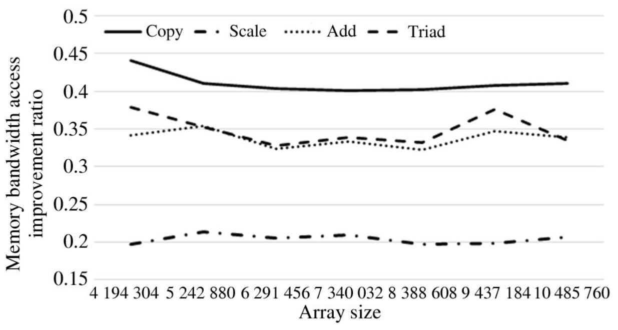


Fig. 9 Memory access comparison between Scenario 1 and Scenario 3

better than that of the neighbor nodes in Scenario 3, such as Copy, Scale, Add and Triad. The improvement amplitude is basically consistent with the bandwidth comparison of another neighbor node in the above figure, which shows that the memory access of neighbor nodes is consistent.

Fig. 10 compares Scenario 1 and Scenario 4. Using the test tool, it is revealed that the local node access performance of Scenario 1 is better than that of the remote node access of Scenario 4, in terms of Copy, Scale, Add and Triad. The memory access bandwidth of Add and Triad increased the most by more than 120%, followed by Copy and Scale with increases of nearly 100%.

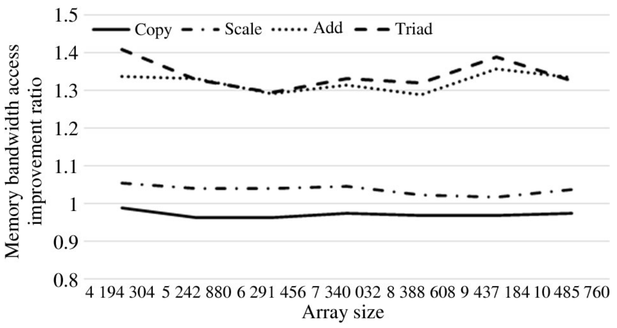


Fig. 10 Memory access comparison between Scenario 1 and Scenario 4

The test data results further reveal that after optimized processing of NUMA node range binding and memory strict binding, the memory access performance of the neighbor NUMA node improved by more than 30% (for the remote NUMA node, it increased by more than 100%). The scheme is shown to be effective in realistic scenarios, thus a high-performance virtual machine is necessary.

4.1.2 NUMA node range binding and memory priority-binding

On the physical machine, the performance of the virtual machine memory bandwidth access rate is improved by using the strategy of virtual machine NUMA node range binding and memory priority binding. The purpose of the test is to determine the maximum value of memory bandwidth access rate, compare it with NUMA node range binding and memory strict binding test results, then identify the difference between memory priority binding performance optimization baseline data and memory strict binding performance optimization baseline data.

In this test, the same four scenarios as those in subsection 4.1.1 were set, except memory priority binding was used here instead of memory strict binding.

The virtual machine NUMA node range binding and memory priority binding strategy ensure that the location of the business virtual machine access memory is local memory, avoiding cross NUMA node access memory and improving the memory access performance of the virtual machine.

According to the test scenario, the results are as follows. Fig. 11 shows that, through testing, the performance of local node access in Scenario 1 is better than that of neighbor nodes in Scenario 2. Among the functions, the memory access bandwidth of Copy increased the most by nearly 40%, followed by Add and Triad with increases of about 30%, then Scale with the least increase of nearly 20%.

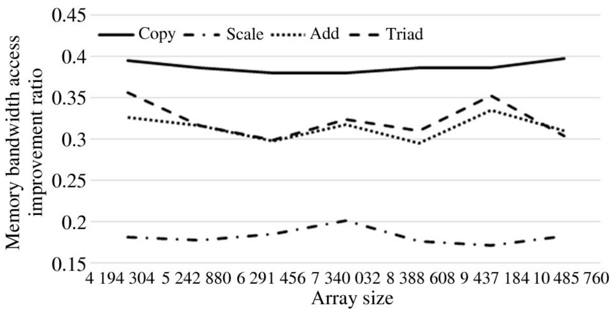


Fig. 11 Comparison of NUMA node range binding and non-NUMA node fixed binding to nodeset 1

Fig. 12 compares Scenario 1 and Scenario 3 by using the test tool, which indicates that the local node access performance of Scenario 1 is better than that of the neighbor node access of Scenario 3, in terms of Copy, Scale, Add, and Triad. The improvement amplitude is basically consistent with the bandwidth comparison of another neighbor node in the above figure, which shows that the memory access of neighbor nodes is consistent.

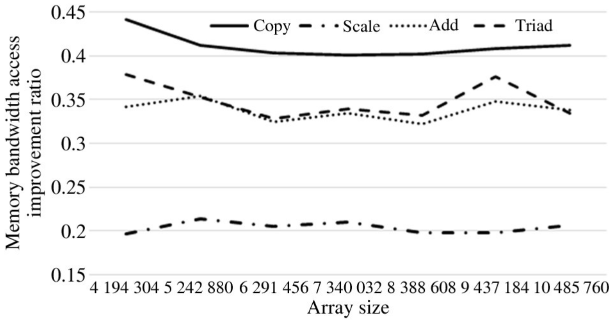


Fig. 12 Comparison of NUMA node range binding and non-NUMA node fixed binding to nodeset 3

Fig. 13 shows the comparison between Scenario 1 and Scenario 4, where the local node access performance of Scenario 1 is better than that of the remote node access of Scenario 4, in terms of Copy, Scale, Add, and Triad. Specifically, the memory access bandwidth of Add and Triad increased the most by more than 120% , followed by Copy and Scale with increases of nearly 100%.

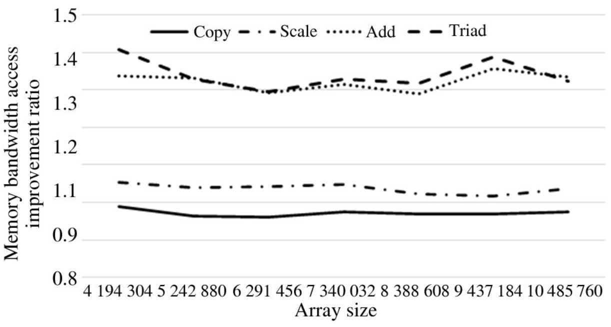


Fig. 13 Comparison of NUMA node range binding and non-NUMA node fixed binding to nodeset 4

The test results indicate that after optimization processing of NUMA node range binding and memory priority binding, the memory access performance of the neighbor NUMA node can be improved by more than 30% (for the remote NUMA node, it can be improved by more than 100%). The scheme is real and effective with no obvious difference in memory access performance between memory priority binding and memory fixed binding.

Understanding memory and computing performance in multi-core platforms is a prerequisite to perform optimizations. The state-of-the-art methods usually focus on the optimization method of specific scene. The method presented in this paper can be improved by more than 20% compared with the method in specific field in general operation[3].

4.2 Computing performance verification

For a virtual machine with one-to-one processor binding, the processor of the virtual machine will not switch, which can effectively ensure the computing performance of the virtual machine and prevent processor switching.

This test is aimed at processor exclusive binding and employs LMbench, a multi-platform open source benchmark used to evaluate the comprehensive performance of the system, as the test tool. For the application with frequent service load jitter, the maximum performance optimization data after the application of this scheme is provided.

On the physical machine, the virtual machine NUMA node processor exclusive binding strategy improves the performance of the virtual machine. The purpose is to test and attain the maximum value of the sustainable computing power, compare it with the maximum value of the non-processor exclusive binding virtual machine, then achieve the baseline data of computing performance optimization. In this test, two test scenarios are set. Scenario 1 occurs after NUMA node exclusive binding optimization, Scenario 2 occurs after NUMA node processor range binding optimization, and Scenario 3 simulates the real scenario that customers may encounter for comparative testing.

Scenario 1 The vCPU of virtual machine 1 is bound to the specific processor with nodeset = 1.

Scenario 2 The vCPU range mode of virtual machine 2 is bound to nodeset = 1.

Scenario 3 The vCPU range of virtual machine 3 is not bound by a nodeset.

The exclusive binding strategy of the virtual machine NUMA node processor ensures that the location of business virtual machine processor will not switch and will improve the computing performance of the virtual machine. The key indicators of the test are affected by the following principles.

(1) NUMA node processors exclusively bind the virtual machine with no processor switching, and thus, the performance should be higher.

(2) For the virtual machine bound by the NUMA node processor range, there is processor switching, and the performance is normal.

(3) For the virtual machine without fixed binding of the NUMA node processor, there is processor switching, and performance is normal.

Test results The overhead of context switching is about 2.7 – 5.48 μs.

Test summary By optimizing the exclusive binding of NUMA nodes, the performance loss of about 2.7 – 5.48 μs per processor switch can be avoided. This is recommended for computing performance sensitive industries, such as big data and AI.

5 Conclusion

The main purpose of this work is to introduce the memory and processor optimization of a virtual machine, which is divided into three parts.

The first part analyses the working principle of NUMA, the memory access performance, and stability of a virtual machine. In Sections 1 and 2, the current processor architecture, the relationship between the memory and processor, the existing form of virtual machines on physical machines, and the problems of virtual machines based on a physical machine architecture are introduced. Finally, the solutions to these existing issues are identified.

The second part designs a plan for the first part of the problem and provides a detailed elaboration and analysis. The plan mainly includes the resource binding and scheduling strategy of the memory and processor. Section 2 describes the main problems, and in Section 3, the problems are solved in two aspects, process NUMA binding and memory pre-allocation.

The third part verifies the creation of a virtual machine after optimization, including the impact of the binding strategy on the performance and effectiveness of the scheduling service on resource scheduling and protection. The verification includes two aspects, processor memory access performance and computing performance, which confirms the effectiveness of the solutions mentioned in the second part.

References

[1] Dokulil J, Benkner S. NUMA-aware CPU core allocation in cooperating dynamic applications[C] // Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops, New Orleans, USA, 2020:950-957

[2] Memarzia P, Ray S, Bhavsar V C. Toward efficient in-memory data analytics on NUMA systems[J]. *arXiv*: 190801860, 2019

[3] Khaleghzadeh H, Manumachu R R, Lastovetsky A. A hierarchical data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous multi-accelerator NUMA nodes[J]. *IEEE Access*, 2019, 8(78): 61-76

[4] Baruah T, Sun Y, Dinçer A T, et al. Griffin: hardware-software support for efficient page migration in multi-GPU systems[C] // Proceedings of the IEEE International Symposium on High Performance Computer Architecture, San Diego, USA, 2020: 596-609

[5] Sun Y, Baruah T, Mojumder S A, et al. MGPUSim: enabling multi-GPU performance modeling and optimization [C] // Proceedings of the 46th International Symposium on Computer Architecture, Phoenix, USA, 2019: 197-209

[6] Yan Z, Lustig D, Nellans D, et al. Nimble page management for tiered memory systems[C] // Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, USA, 2019: 331-345

[7] Jaleel A, Ebrahimi E, Duncan S. Ducati: high-performance address translation by extending TLB reach of GPU-accelerated systems[J]. *ACM Transactions on Architecture and Code Optimization*, 2019, 16(1): 1-24

[8] Rab M, Marotta R, Ianni M, et al. NUMA-aware non-blocking calendar queue[C] // Proceedings of IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications, Prague, Czech, 2020: 1-9

[9] Strati F, Giannoula C, Siakavaras D, et al. An adaptive concurrent priority queue for NUMA architectures[C] // Proceedings of the 16th ACM International Conference on Computing Frontiers, Alghero, Italy, 2019: 135-144

[10] Kim S, Zheng H, Venkatasubramanian R, et al. Adaptive CPU NUMA scheduling [P]. US Patent: 2016/0085571, 2019

[11] Jiménez-Peris R, Ballesteros F J, Kranas P, et al. NUMA-aware deployments for LeanXcale database appliance [C] // Proceedings of the International Conference on Cloud Computing and Services Science, Heraklion, Greece, 2019: 666-671

[12] Wu R, Zhang X, Kong X, et al. Evaluation of NUMA-aware scheduling in warehouse-scale clusters[C] // Proceedings of 2019 IEEE 12th International Conference on Cloud Computing, Milan, Italy, 2019: 475-477

[13] Wade J, Buenfil J, Collopy P. A systems engineering approach for artificial intelligence: inspired by the VLSI revolution of Mead & Conway[J]. *INSIGHT*, 2020, 23 (1): 41-7

[14] Langewiesche W. What really brought down the Boeing 737 Max[EB/OL]. https://www.nytimes.com/interactive/2020/02/27/opinion/2019-year-in-illustration.html: The New York Times, 2019

[15] Siebel T M. Digital Transformation: Survive and Thrive in An Era of Mass Extinction[M]. New York: Rosetta Books, 2019

[16] Rouse M. Definition: multi-core processor[J]. *TechTarget Retrieved March*, 2013, 6: 131-144

[17] Schauer B. Multicore processors—a necessity[J]. *ProQuest Discovery Guides*, 2008, 9: 1-14

[18] Manumachu R, Lastovetsky A L. Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution[J]. *Concurrency and Computation: Practice and Experience*, 2019, 31(4): 109-122

[19] Majo Z, Gross T R. Memory system performance in a NUMA multicore multiprocessor[C] // Proceedings of the 4th Annual International Conference on Systems and Storage, Haifa, Israel, 2011: 1-10

[20] Ganguly D, Zhang Z, Yang J, et al. Interplay between hardware prefetcher and page eviction policy in CPU-GPU unified virtual memory[C] // Proceedings of the 46th International Symposium on Computer Architecture, Phoenix, USA, 2019: 224-235

[21] Li C, Ausavarungnirun R, Rossbach C J, et al. A framework for memory oversubscription management in graphics processing units[C] // Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, USA, 2019: 49-63

[22] Kruse D M, Reuther L, Broas K M. Efficient programmatic memory access over network file access protocols [P]. US Patent:201410359114, 2019

[23] Wang Z, Nowatzki T. Stream-based memory access specialization for general purpose processors[C] // Proceedings of the 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture, Phoenix, USA, 2019: 736-49

He Mujun, born in 1982. He is a Ph. D candidate in College of Computer Science, Chongqing University. He received his M. S. degree from Institute of Process & Engineering of Chinese Academy of Science in 2008. He also received his B. S. degree from Zhejiang University in 2004. His research interests include cloud computing and cyber physical system.