Toward NUMA-aware Multi-VM Cooperative Resource Scheduling

Sungying Chiang
The University of Tokyo
Tokyo, Japan

Ryo Hayashi
The University of Tokyo
Tokyo, Japan

Masanori Misono
Technical University of Munich
Munich, Germany

Takahiro Shinagawa
The University of Tokyo
Tokyo, Japan

EXTENDED ABSTRACT

The rapid growth of data-intensive applications, such as high-performance computing and artificial intelligence, has significantly increased the demand for memory capacity, driving the widespread adoption of the non-uniform memory access (NUMA) architecture. NUMA facilitates increasing memory capacity by partitioning memory into local and remote memory, effectively addressing the scalability, bandwidth, and contention challenges associated with the uniform memory access architecture. NUMA is also becoming increasingly prevalent in cloud environments that host data-intensive applications.

While NUMA offers significant advantages, non-uniformity requires operating systems (OS) to implement effective resource scheduling to maintain optimal memory access performance. For instance, OSs need to ensure that processes and their associated memory are located on the same NUMA node as much as possible to avoid remote memory access. However, virtualization technologies commonly used in cloud environments often hide the NUMA topology from guest OSs, making it challenging to achieve optimal resource scheduling based on NUMA.

Several previous studies have attempted to address the issue of leveraging NUMA in virtualized environments. Voron et al. [3] provide a simple interface for the guest OS to inform memory release and policy choices so that the hypervisor can implement appropriate NUMA policies. XPV [1] provides a paravirtualization interface that exposes virtual NUMA topologies and allows guests to appropriately realize NUMA policies. CPS [2] provides a cooperative para-virtualized scheduling framework to facilitate the proactive exchange of timely scheduling information between the hypervisor and guest VMs. However, balancing the flexible allocation of NUMA memory among multiple guest OSs while allowing each guest OS to perform resource scheduling that leverages the NUMA topology remains challenging.

We propose a NUMA-aware cooperative resource scheduling scheme for multiple virtual machines (VM). Our key idea is to have all VMs exactly replicate the host machine's NUMA topology, enabling the guest OSs to fully exploit the underlying physical NUMA topology. This allows guest OSs to take advantage of dynamically allocated NUMA memory even if they do not support NUMA reconfiguration. Additionally, to facilitate resource sharing among guest OSs, memory resources are managed cooperatively to avoid unnecessary over-commitment and to maximize the performance of the guest OSs.

We design three components to achieve our proposal scheme. The first is a binding manager that replicates the NUMA topology of the host machine and exposes it to the guest OS. The host OS creates the same number of virtual CPUs as physical CPUs in each VM and binds them one-to-one so that the guest OS can recognize which NUMA node the virtual CPUs belong to. The host OS also creates the same amount of virtual memory as physical memory for each VM and exposes the NUMA topology to the guest OS in the same configuration as the physical machine, so that the guest OS can recognize which NUMA node a memory region belongs to. This one-to-one correspondence in both CPU and memory allows the guest OS to properly handle the NUMA topology without having to support dynamic NUMA configuration changes.

The second is a balancing manager that accommodates available resources between VMs. The balancing manager aggregates information from each guest OS and monitors the overall resource utilization of the physical machine. Each guest OS runs a resource management daemon to notify the balancing manager of the resource utilization status of the guest OS. The balancing manager communicates with the resource management daemon to understand the resource utilization status of each VM and instructs it to increase or decrease resource usage. This coordinated resource accommodation between host and guest makes it possible to control the actual amount of resources used by each VM, while making all resources of the host machine visible to the VMs.

The third is a custom memory allocator in the guest OS. This allocator interprets memory allocation requests to enforce NUMA node-specific allocation, prioritizing the NUMA node where the allocating task is running. Additionally, the allocator initiates a 'cleanup' process when the overall system load is assumed to be low, based on information provided by the balancing daemon, to facilitate the migration of memory allocated on remote nodes to the local node.

We are currently implementing these components and plan to evaluate their performance in the future.

REFERENCES

[1] Bao Bui et al. 2019. When eXtended Para - Virtualization (XPV) Meets NUMA. In *Proceedings of the Fourteenth EuroSys Conference 2019 (EuroSys '19)*. 15 pages. https://doi.org/10.1145/3302424.3303960

[2] Yuxuan Liu et al. 2024. CPS: A Cooperative Para-virtualized Scheduling Framework for Manycore Machines. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '23)*. 43–56. https://doi.org/10.1145/3623278.3624762

[3] Gauthier Voron et al. 2017. An interface to implement NUMA policies in the Xen hypervisor. In *Proceedings of the Twelfth European Conference on Computer Systems (EuroSys '17)*. 453–467. https://doi.org/10.1145/3064176.3064196