



VMware vSphere Online Library
ESX 4.0 Update 1 and Later
vCenter Server 4.0 Update 1 and Later

[vSphere Resource Management Guide : Using NUMA Systems with ESX/ESXi](#) : VMware NUMA Optimization Algorithms and Settings

VMware NUMA Optimization Algorithms and Settings

This section describes the algorithms and settings used by ESX/ESXi to maximize application performance while still maintaining resource guarantees.

Home Nodes and Initial Placement

When a virtual machine is powered on, ESX/ESXi assigns it a home node. A virtual machine runs only on processors within its home node, and its newly allocated memory comes from the home node as well.

Unless a virtual machine's home node changes, it uses only local memory, avoiding the performance penalties associated with remote memory accesses to other NUMA nodes.

New virtual machines are initially assigned to home nodes in a round robin fashion, with the first virtual machine going to the first node, the second virtual machine to the second node, and so forth. This policy ensures that memory is evenly used throughout all nodes of the system.

Several operating systems, such as Windows Server 2003, provide this level of NUMA support, which is known as initial placement. It might be sufficient for systems that run only a single workload, such as a benchmarking configuration, which does not change over the course of the system's uptime. However, initial placement is not sophisticated enough to guarantee good performance and fairness for a datacenter-class system that is expected to support changing workloads.

To understand the weaknesses of an initial-placement-only system, consider the following example: an administrator starts four virtual machines and the system places two of them on the first node. The second two virtual machines are placed on the second node. If both virtual machines on the second node are stopped, or if they become idle, the system becomes completely imbalanced, with the entire load placed on the first node. Even if the system allows one of the remaining virtual machines to run remotely on the second node, it suffers a serious performance penalty because all its memory remains on its original node.

Dynamic Load Balancing and Page Migration

ESX/ESXi combines the traditional initial placement approach with a dynamic rebalancing algorithm. Periodically (every two seconds by default), the system examines the loads of the various nodes and determines if it should rebalance the load by moving a virtual machine from one node to another.

This calculation takes into account the resource settings for virtual machines and resource pools to improve performance without violating fairness or resource entitlements.

The rebalancer selects an appropriate virtual machine and changes its home node to the least loaded node. When it can, the rebalancer moves a virtual machine that already has some memory located on the destination node. From that point on (unless it is moved again), the virtual machine allocates memory on its new home node and it runs only on processors within the new home node.

Rebalancing is an effective solution to maintain fairness and ensure that all nodes are fully used. The rebalancer might need to move a virtual machine to a node on which it has allocated little or no memory. In this case, the virtual machine incurs a performance penalty associated with a large number of remote memory accesses. ESX/ESXi can eliminate this penalty by transparently migrating memory from the virtual machine's original node to its new home node:

- 1 The system selects a page (4KB of contiguous memory) on the original node and copies its data to a page in the destination node.
- 2 The system uses the virtual machine monitor layer and the processor's memory management hardware to seamlessly remap the virtual machine's view of memory, so that it uses the page on the destination node for all further references, eliminating the penalty of remote memory access.

When a virtual machine moves to a new node, the ESX/ESXi host immediately begins to migrate its memory in this fashion. It manages the rate to avoid overtaxing the system, particularly when the virtual machine has little remote memory remaining or when the destination node has little free memory available. The memory migration algorithm also ensures that the ESX/ESXi host does not move memory needlessly if a virtual machine is moved to a new node for only a short period.

When initial placement, dynamic rebalancing, and intelligent memory migration work in conjunction, they ensure good memory performance on NUMA systems, even in the presence of changing workloads. When a major workload change occurs, for instance when new virtual machines are started, the system takes time to readjust, migrating virtual machines and memory to new locations. After a short period, typically seconds or minutes, the system completes its readjustments and reaches a steady state.

Transparent Page Sharing Optimized for NUMA

Many ESX/ESXi workloads present opportunities for sharing memory across virtual machines.

For example, several virtual machines might be running instances of the same guest operating system, have the same applications or components loaded, or contain common data. In such cases, ESX/ESXi systems use a proprietary transparent page-sharing technique to securely eliminate redundant copies of memory pages. With memory sharing, a workload running in virtual machines often consumes less memory than it would when running on physical machines. As a result, higher levels of overcommitment can be supported efficiently.

Transparent page sharing for ESX/ESXi systems has also been optimized for use on NUMA systems. On NUMA systems, pages are shared per-node, so each NUMA node has its own local copy of heavily shared pages. When virtual machines use shared pages, they don't need to access remote memory.

Memory Page Sharing Across and Within NUMA Nodes

The `VMkernel.Boot.sharePerNode` option controls whether memory pages can be shared (de-duplicated) only within a single NUMA node or across multiple NUMA nodes.

`VMkernel.Boot.sharePerNode` is turned on by default, and identical pages are shared only within the same NUMA node. This improves memory locality, because all accesses to shared pages use local memory.

Note

This default behavior is the same in all previous versions of ESX.

When you turn off the `VMkernel.Boot.sharePerNode` option, identical pages can be shared across different NUMA nodes. This increases the amount of sharing and de-duplication, which reduces overall memory consumption at the expense of memory locality. In memory-constrained environments, such as VMware View deployments, many similar virtual machines present an opportunity for de-duplication, and page sharing across NUMA nodes could be very beneficial.