

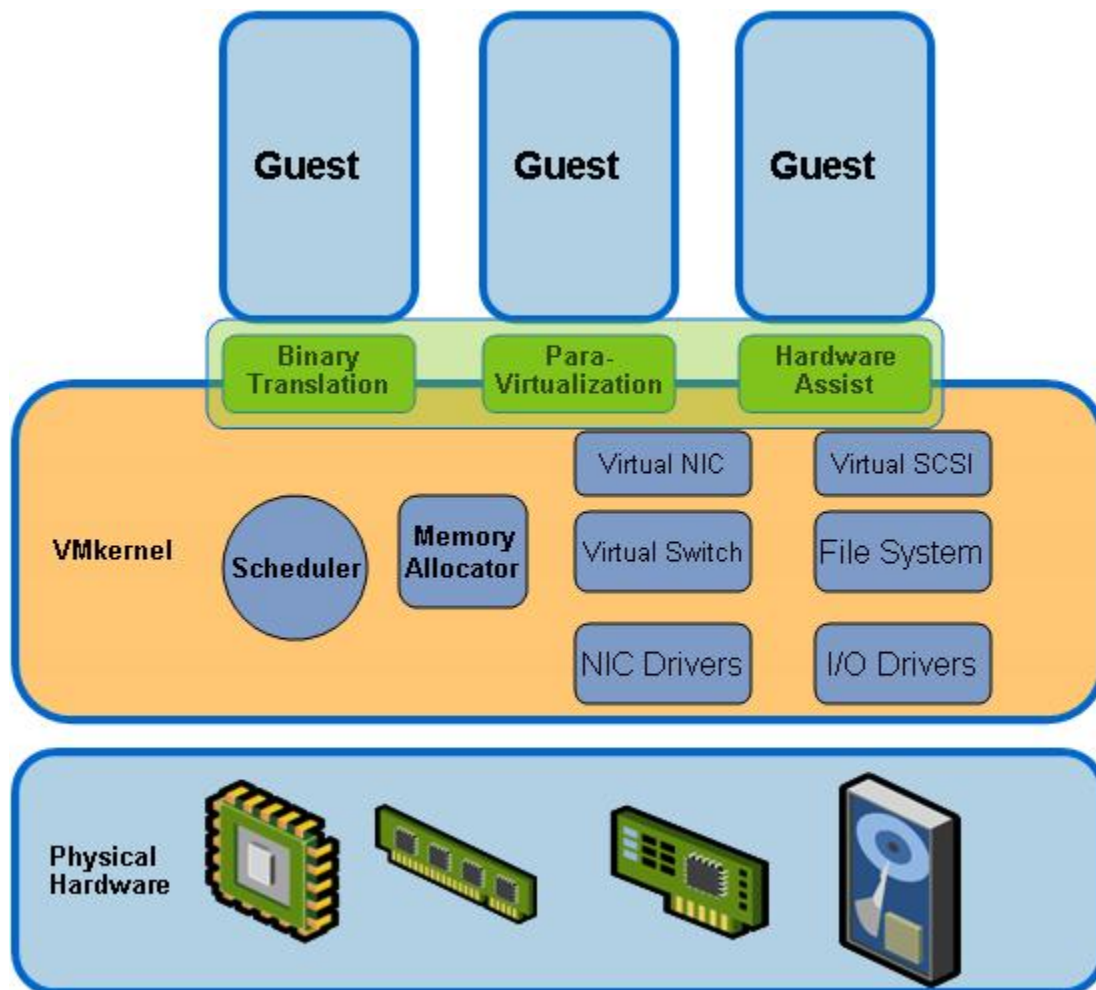
# VMkernel Scheduler

## Introduction

Details on the ESX Server scheduler are commonly requested when I engage customers and partners. People want to know more about how the scheduler works, when SMP should be used, and what the deal is with SMP co-scheduling. This page will answer these questions and others as they arise in the forum or the discussion portion of this page.

## Terminology and Architecture

In VMware parlance, the monitor is the part of our products that provides a virtual interface to the guest operating systems. The VMkernel is the part of our products that manages interactions with the devices, handles memory allocation, and schedules access to the CPU resources, among other things. This is shown in the following figure.



This document will provide information on one part of the VMkernel: the scheduler.

# Performance Scaling and the Scheduler

It is a critical requirement for enterprise deployments that an operating system provide fast and fair access to the underlying resources. As a critical part of this design, the scheduler has undergone countless engineer-years of development to guarantee that this requirement is met. We've now released dozens of papers showing linear scaling of workloads as vCPU count is scaled up within a single VM and VM count is scaled up within a single host. Here are a few such papers that contain supporting data.

1. [VM scaling](#) as demonstrated by Oracle databases.
2. [VM and vCPU scaling](#) under IBM DB2 load.
3. [VM and vCPU scaling](#) with SQL Server running in the VM.

The scheduler's ability to fairly scale up to and beyond totally committed CPU resources is no accident. In fact, in a conversation I had with a QA manager I was assured that the VMkernel's scheduler would fairly distribute CPU resources to all VMs at least up to 4x CPU overcommitment. Of course, on a system with the CPU over-committed by 4x each VM will only run at 1/4 native speed but the scheduler keeps the VMs running at that performance. Not one at 1/8 speed, one at 1/10 speed, and another at 1/4 speed.

## SMP and the Scheduler

As ESX Server supports uniprocessor (UP) and symmetric multiprocessor (SMP) VMs, the fair-and-fast requirement for the scheduler must be upheld in the presence of concurrently executing UP and SMP VMs. Internal testing of this requirement shows fair scheduling even in the presence of concurrently executing 1-way, 2-way, and 4-way VMs.

In fact, the ability to fairly execute under such environments is a very tricky problem for a scheduler. We've run analysis on competitors' products and found that the ability to fairly balance differently-sized VMs is something of which ESX Server alone is capable. Stay tuned in the coming months as we back this claim up with performance data.

## Cell Size

One construct that assists the scheduler in optimally placing VMs on a heavily utilized system is a cell. A cell is a logical grouping of a subset of CPU cores in the system. In ESX 3 versions the cell size is equal to four. Since the cell is statically assigned to physical cores, this means that each four-core processor is in exactly one cell. When only dual-core processors are present, a cell is comprised of two sockets. The most important thing to know about cells is the following:

*A VM cannot span more than one cell.*

This means that four-way VMs run on only one socket at a time in systems with quad-core CPUs. For this case, the number of options presented to the scheduler is equal to the number of sockets. In future versions of ESX we plan to increase the cell size to eight. In some cases (such as systems with hexa-core CPUs) a modification of the cell size can improve performance. See [KB article 1007361](#) for more information.

## UP or SMP?

When and if to use SMP is a common question from VMware users. The simple answer to this is to only use SMP when needed. Why only use SMP when needed? There are two reasons:

1. SMP schedulers are less efficient than UP schedulers. This is a simple experiment that can be confirmed with trivial benchmarks like Netperf or Passmark. On UP systems (either virtual or native) the UP hardware abstraction layer (HAL) will provide marginally better results than the SMP HAL.
2. Even when unused, virtualization of idle vCPUs requires resources by the kernel. Memory is needed to maintain data structures and CPU resources are needed to virtualize the idle system. The amount of work needed to support an idle CPU varies greatly but usually is in the realm of 1-2% of a single CPU core.
3. The work required to deliver timer interrupts increases quadratically with the number of vCPUs, like RHEL5, the number of timing interrupts delivered by the VMkernel can be quite high. See [Red Hat Enterprise Linux](#) for more information on this issue.

## What About Co-scheduling?

Back in the days of ESX Server 2.5, SMP VMs had to have their vCPUs co-scheduled at the same instant to begin running. Because only 2-way VMs were supported at this time, that meant that two CPU cores had to be available simultaneously to launch a 2-way VM. On a server with a total of only two cores, this meant that the VM could not be launched concurrently with any other process on the server. This would include the service console, the web interface, or any other process.

This requirement was reduced in ESX Server 3.0 through a process called relaxed co-scheduling. Effectively SMP VMs can have their vCPUs scheduled at slightly different times and idle vCPUs didn't necessarily have to be scheduled concurrently with running vCPUs. More details on this are available in the [Co-scheduling SMP VMs in VMware ESX Server](#) page.

## NUMA Considerations

Support for non-uniform memory access (NUMA) architectures was introduced in ESX Server 2. This meant that the scheduler became aware that memory was not uniform across each CPU. Each CPU node had access to its own local memory and a larger pool of remote memory (which was divided as local memory for the other CPU nodes.) Memory access to local memory is much faster than remote memory so the scheduler should favor the placement of processes on nodes that held the processes' memory.

Subsequent generations of ESX Server continued to optimize for the use of NUMA memory. This included placement of vCPUs next to needed memory and startup of VMs at NUMA nodes with resources available for execution. All of this is transparently handled by the scheduler but it should be noted that the newer your version of ESX Server, the better its NUMA scheduling is.