MATTHEW SACKS

Linux kernel resource allocation in virtualized environments

Matthew Sacks works as a systems administrator at Edmunds.com. His focus is network, systems, and application performance tuning.

matthew@matthewsacks.com

THE BEHAVIOR OF THE LINUX KERNEL and its resource allocation methods are an art and science, more the former than the latter. When working with Linux in a virtualized environment, the complexities of the Linux kernel's resource allocation algorithms increase. New performance issues may arise and proper functionality can come to a halt—especially on overutilized systems. The Linux kernel behaves differently on a virtualized platform in comparison to bare-metal. Why Linux behaves differently on a virtual platform and how to address performance and stability issues when it starts to malfunction or degrade in performance relate, but do not depend on, the environment. The solution presented here is not that of changing the environment, but, rather, that of making adjustments to the Linux kernel to coalesce with the hypervisor.

NOTE: VMware's ESX platform is certainly not the be-all and end-all of virtualization. However, it is widely used and accepted. The same principles used in this example may apply to other virtual platforms. The tuning methods provided in this article are not intended as a replacement for good capacity planning.

The Environment

The virtual environment comprised 6 VMware ESX 3.01 servers running approximately 11 virtual machines per server. Each ESX server had Red Hat Enterprise Linux 4 Update 4 machines running a wide array of application and Web servers. The environment was intended to simulate a high-volume, high-traffic production Web site by simulating load tests on the virtual servers. The phenomenon experienced was the Linux Kernel's OOM-Kill function, which would trigger and kill processes that were consuming the most resources. How the ESX server interacts with this Linux kernel in allocating resources is the starting point.

VMWARE ESX SERVER RESOURCE ALLOCATION

The VMware ESX server adds another layer of abstraction between the Linux server's physical and virtual memory and the real memory of the ESX server. The ESX server creates additional memory overhead in managing the virtual devices, CPUs, and memory of the virtual machine. It can be thought of as virtual memory that manages virtual memory: a new set of resources that must be managed on top of the guest operating system's own virtual memory management algorithms. Resources can run thin quickly and resource allocation issues tend to increase faster on a virtual server than on a bare-metal server.

For example, consider an ESX server with 1 GB of memory, running two virtual machines with 256 MB of "physical" memory allocated for each virtual server. The amount of free resources available is approximately 170 MB. The service console uses approximately 272 MB, the VMkenel uses somewhat less than that, and, depending on how many virtual CPUs and devices are added to each virtual server, the memory overhead increases.

ESX uses a proprietary memory ballooning algorithm to adjust and allocate memory to virtual servers. ESX loads a driver into the virtual server which modifies the virtual server's page-claiming features. It increases or decreases memory pressure depending on the available physical resources of the ESX server, causing the guest to invoke its own memory management algorithms. When memory is low the virtual server's OS decides which pages to reclaim and may swap them to its virtual swap.

However, sometimes pages cannot be reclaimed fast enough or memory usage grows faster than can be committed to swap; then the Linux OS kills processes and "Out of Memory" errors appear in the syslog.

The Linux Out of Memory Killer

The `out_of_memory()` function is invoked by `alloc_pages()` when free memory is very low and the Page Frame Resource Allocation Algorithm has not succeeded in reclaiming any page frames. The function invokes `select_bad_process()` to select a victim and them invokes the `oom_kill_process()` to kill the process that is utilizing the most resources. Typically, the `select_bad_process()` function chooses the process that is not a critical system process and is consuming the largest number of page frames. This is why, when running a resource-intensive application or Web server on a virtual environment, the application or Web server may begin crashing frequently. Check the logs for the "Out of Memory" errors to see if the `oom_kill_process()` function is being called by the Linux kernel. The `oom_kill_process()` function comes into play because of how Linux is allocating memory into lower memory zones.

Memory Zones and the Dirty Ratio

By default, the Linux kernel allows addressing of memory in the lower zone called ZONE_DMA. This zone contains page frames of memory below 16 MB. In high workloads, once the ZONE_NORMAL (normal memory zone) and ZONE_HIGHMEM have been exhausted by an application, it will begin to allocate memory from ZONE_DMA and the requestor of the application will pin them, thereby denying access to these zones by other critical system processes. The `lower_zone_protection` kernel parameter determines how aggressive the kernel is in defending the lower memory allocation zone.

The dirty ratio is a value expressed in percentage of system memory at which limit processes generating dirty buffers will write data to disk rather than relying on the pdflush daemons to perform this function. The pdflush kernel thread scans the page cache looking for dirty pages (pages that the kernel has set to be swapped to disk) and then ensures that no page remains dirty for too long. ZONE_DMA can be protected from being utilized by applications and the dirty ratio can be adjusted by tuning the Linux kernel.

Tuning the Linux Kernel for Virtualization

The /etc/sysctl.conf file allows modification of select kernel settings without recompilation of the kernel. The /etc/sysctl.conf file is used to adjust behaviors of the Linux kernel to address issues with resource allocation. A set of virtual memory tunable parameters is available for tuning from within this file. Two tunable virtual memory parameters in particular will solve the "Out of Memory" problems and most other problems with memory allocation in a virtual Linux server.

To protect the lower zones of memory from being utilized by the applications on a virtual Linux server, edit the /etc/sysctl.conf file to include the following parameter:

```
lower_zone_protection = 100
```

To increase the ratio by which the pdflush kernel thread scans the page cache to look for dirty pages to 5 percent of the system memory, edit /etc/sysctl.conf to include the following parameter:

```
dirty_ratio = 5
```

Reboot the system or type the command `sysctl –p` so that the new kernel settings will take effect. Now most memory resource allocation issues should be resolved in a virtualized Linux environment. Tuning these few settings provides a small insight into how tuning the Linux kernel can solve performance-related problems in a virtualized environment. As a result of the tuning changes, "Out of Memory" errors are reduced dramatically in scope and frequency, and virtual memory is utilized more effectively.

Conclusion

There are numerous algorithms at work with VMware's ESX server and within the Linux kernel itself. In a low-volume environment the standard configurations and settings may be sufficient. In a high-volume, high-performance environment where load tests are constantly making requests against application and Web servers, the defaults are typically insufficient. To squeeze the maximum amount of performance out of a system, an understanding of the underlying algorithms and behaviors of the ESX server is essential before tuning the guest operating system's kernel. The end result is maximal performance on an otherwise overutilized or poorly performing virtual environment. The key is to understand which algorithms need to be changed and to set them to the right values. This is where kernel tuning becomes more of an art than a science.

ACKNOWLEDGMENTS

I want to acknowledge the Systems Administration Team at Edmunds.com, Safdar Husain, David Wolfe, David Morgan, Nikki Sonpar, and Eric Theis, who all contributed in some way to this article.

REFERENCES

[1] D. Bovet and M. Cesati, *Understanding the Linux Kernel* (Sebastopol, CA: O'Reilly & Associates, 2005).

[2] B. Matthews and N. Murray, "Virtual Memory Behavior in Red Hat Linux A.S. 2.1," Red Hat white paper, Raleigh, NC, 2001.

[3] N. Horman, "Understanding Virtual Memory in Red Hat Enterprise Linux 4," Red Hat white paper, Raleigh, NC, 2005.