HyperV_Mon

A Free Tool From TMurgent Technologies
Version 1.6

Introduction

HyperV_Mon is a GUI tool for viewing CPU performance of a system running Hyper-V from Microsoft. Virtualization adds a layer of complexity to understanding true CPU usage of a virtual machine. This tool provides a simple to view set of graphs to depict actual processor utilization, including "overhead" processing due to the use of the Hypervisor.

HyperV_Mon is **not** intended to be a regular monitoring tool that you use every day. This tool is designed to help you understand what is going on when you need to know it. The tool requires that you have .the NET 3.5 Framework installed on the machine that you wish to run the GUI on. If you didn't install Server Core, you can place it on the root partition machine, but typically you want to place this on a desktop that is not on the Hyper-V server to minimize impact on the server itself.

The tool leverages the Root Partition WMI provider – the same interface as is used by SCVMM and third party tools. So as long as the remote GUI user has WMI access to the Root Partition Server you are good to go.

Some of the new features in HyperV_Mon also need WMI access to the VMs themselves, but will reasonably operate if access to the VMs are not available.

*The term "**Overhead**" is used in this document and in the product as a shorthand to capture the concept of processing that is being performed by the Hypervisor itself.*

Technically, this is not "overhead" in the sense of processing that would not be occurring if you didn't virtualize. While some of this processing truly represents overhead, some of it represents processing that would have occurred in drivers of the non-virtualized system.

The tool has no installer. Just an exe with a configure button.

Setting Up The Tool

After copying the exe onto a system, start it and mouse click the “Configure” button. The Configure dialog is shown in Figure 1.

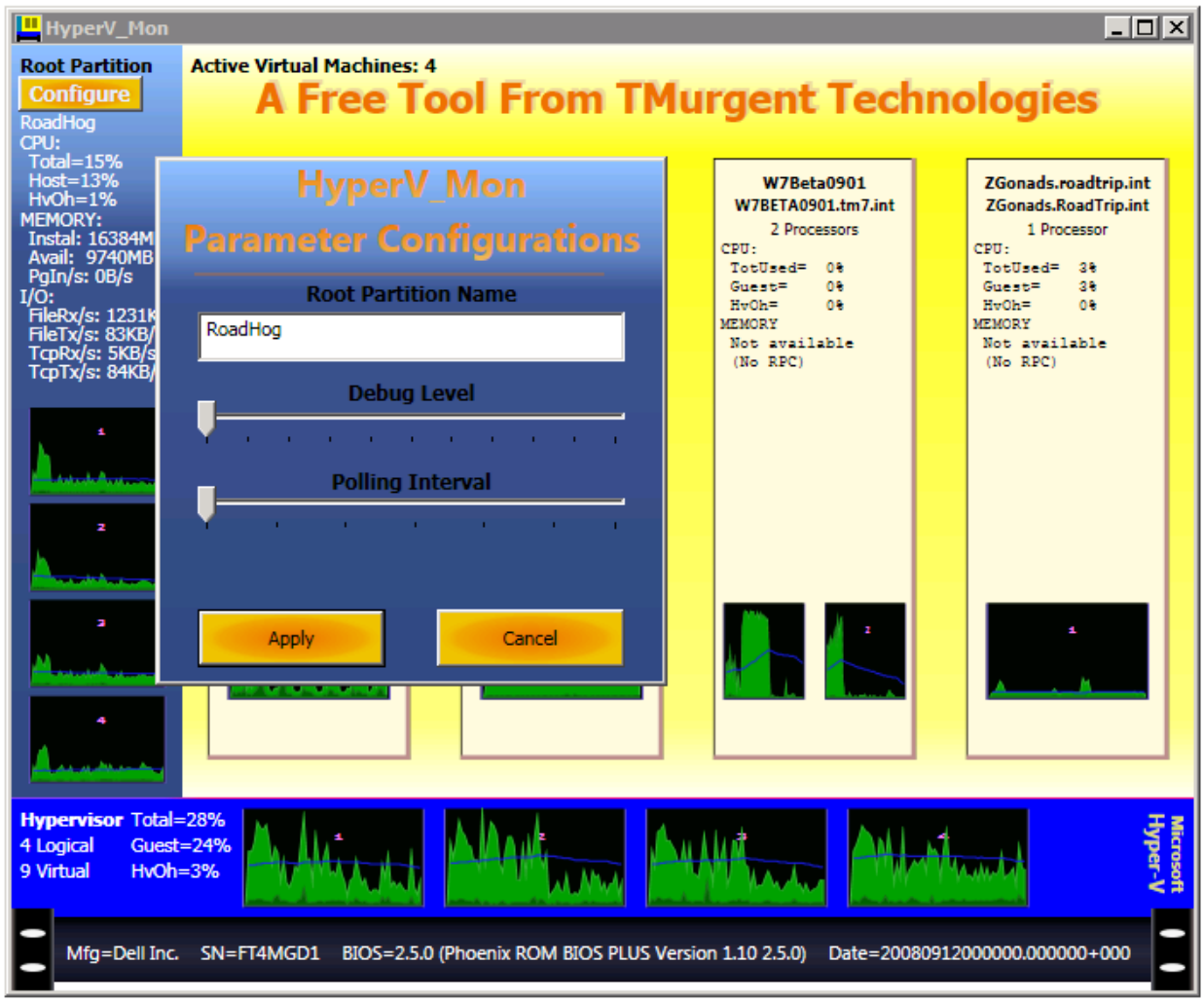

Figure 1 - Configure Screen

You must enter a qualified name for the host in the root partition of the server to be monitored. Any resolvable name may be used.

The Debug level defaults to 0 – meaning no debugging. You can move the slider to the right to enable debugging and configure how much information to log. Level 1 provides for errors only. The log file will be appended to in the current working folder.

The polling interval defaults to 1 second. Move the slider to the right to slow down poling.

The three Configuration items are written to the HKEY_CURRENT_USER hive.

HyperV_Mon requires access to the WMI providers of the root partition of the target machine. This can prove to be a problem unless the machine (and user account you are running under) that the GUI is on is in the same domain as the Root Partition. See the blog of Microsoft's John Howard http://blogs.technet.com/jhoward/default.aspx (especially Hyper-V Remote Management Parts 1-5) if you are having difficulty. Basically, if you can run wbemtest on your client machine and connect to the root partition machine you will be good to go.

Using The Tool

Figure 2 shows an example of the GUI in action.

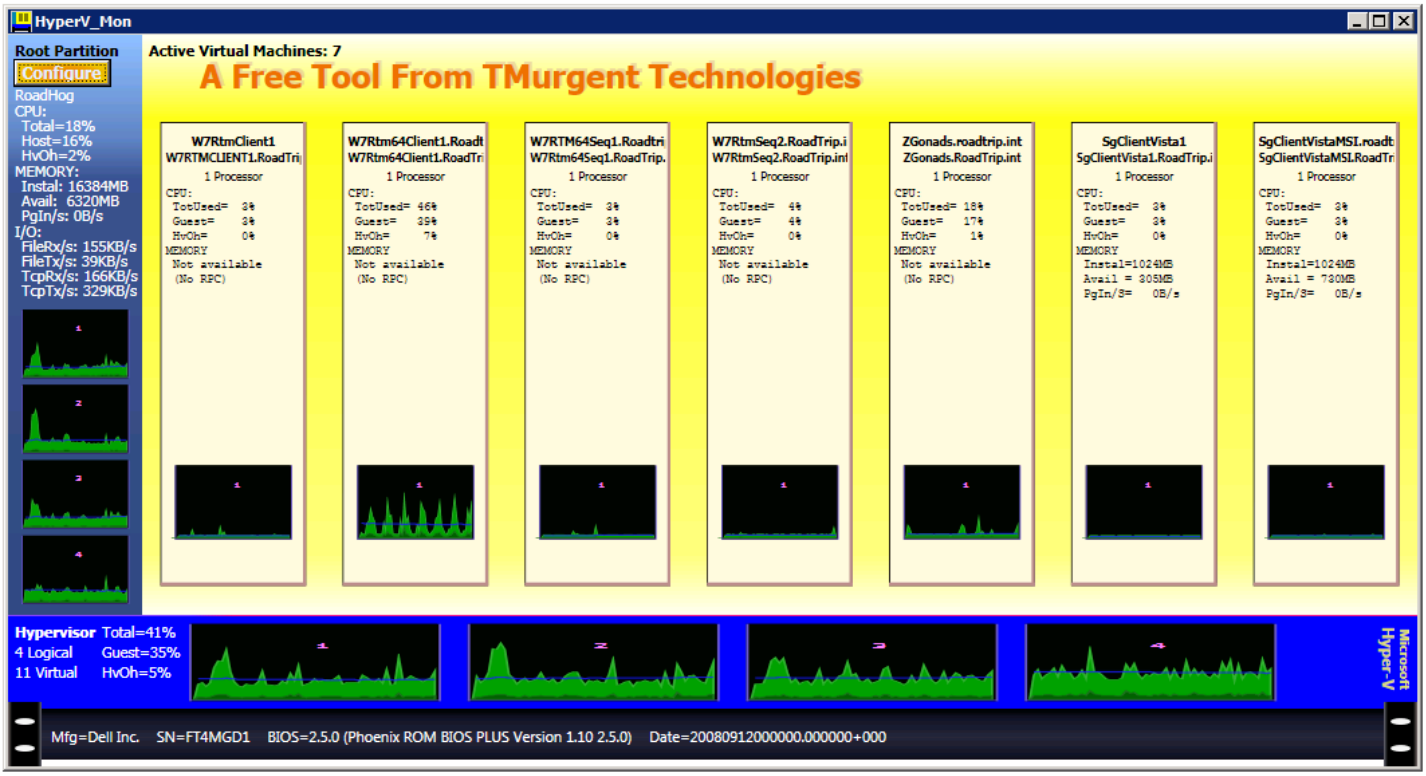

Figure 2 - A Sample of the GUI in Action!

At the bottom layer of the display is the hardware. Not much to talk about there. It doesn't matter what your hardware looks like, the graphic will look like a 1U rack server from my lab. Some information is displayed identifying the hardware.

The next layer up is the Hypervisor. Only Microsoft Hyper-V, and Hyper-V R2 are supported. The number of "Logical CPU" physically installed is detected and depicted. Actual CPU usage at the physical layer is reported. In our case, this was a single quad-core processor. In the example, almost all of the CPU is related to Guest usage and is shown in a lighter shade of green. CPU usage tied to Hypervisor

calls is shown at the lower edge in a darker green color. It is easiest to think of the Hypervisor calls as "hypervisor overhead", although technically some of that "overhead" is processing that would have been performed the guest was on raw metal hardware. The last 60 interval measurements are displayed (1 second each, by default) with the most recent interval displayed on the right of the graph. The blue spline (curvy line) represents a running longer term average (each dot depicts the most recent 60-interval average total CPU at the time of the interval) for trending.

Above the Hypervisor layer the display has two major parts, situated left and right.

On the left is the Root Partition OS. This is typically where your Server Core is running. In the example this was a full blown 2008 with AD, DNS, IIS, Terminal Services, and a host of other things to be interesting. The Root Partition has "Virtual Processors", and Hyper-V will supply one for one for each "Logical Processor" in the Hypervisor. The Hypervisor will schedule the Root Partition virtual processors (along with guest virtual processors) into logical processors as it sees fit. Think of this as round-robin scheduling. If ran a single threaded app on the root partition and used processor affinity within the Root partition OS, you would see all of the CPU for that app running in one of the root partition virtual processors – just as you would see from the task manager inside the root partition. But looking at the Hypervisor layer you will see that the load was actually spread out among the logical processors!

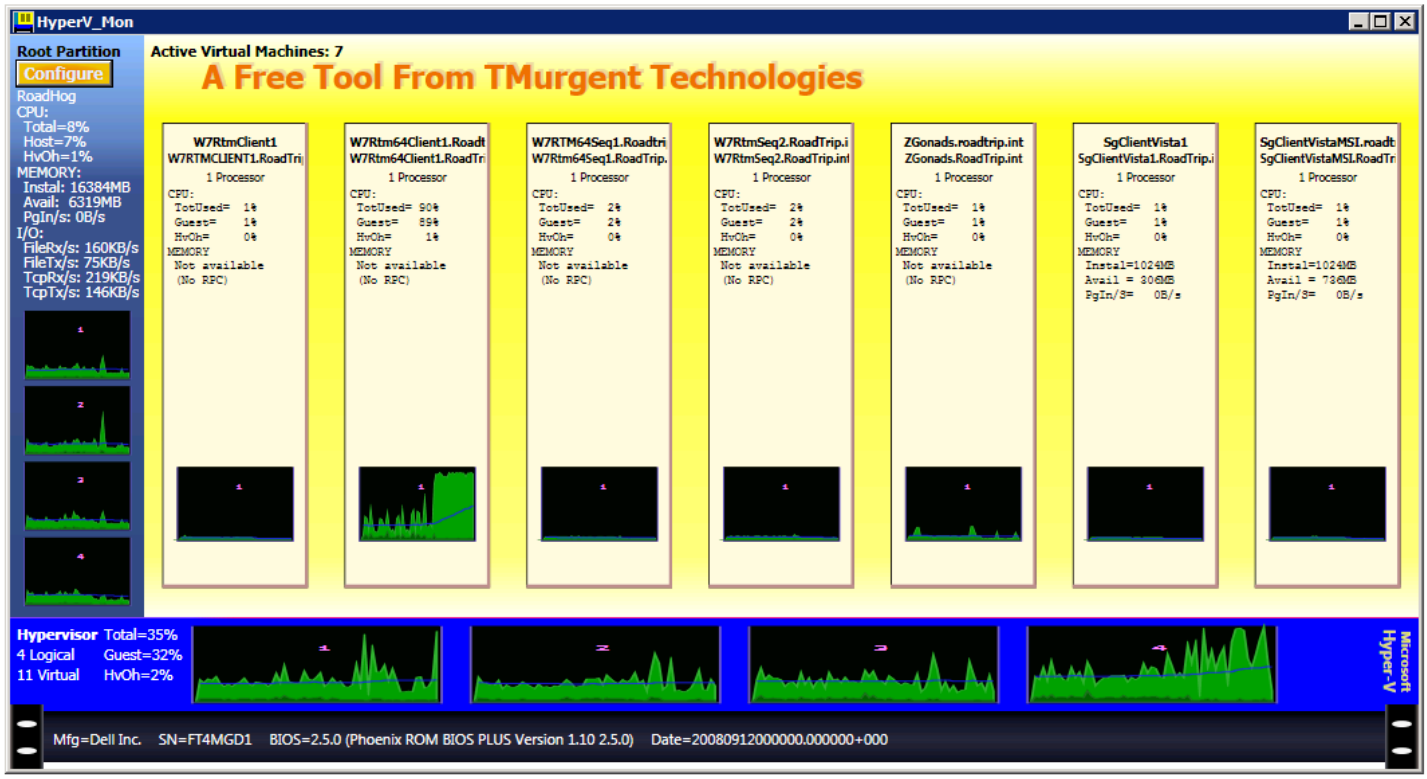

Figure 3 - Example showing Hypervisor calls

On the right, in the yellow area, are the active Guest Virtual Machines. These also have Virtual Processors. Both the Root Partition and Guest VMs virtual processor displays show CPU usage broken between appropriate OS and Hypervisor usage. See Figure 3 for an example of this

You should note that **the task manager inside guest OS machines lies.** This isn't the task manager's fault. In the example shown in Figure 3, the Windows Task Manager within the guest os would be showing 100% CPU utilization where our snapshot shows only 90%. Microsoft refers to this as clock slew, but basically the Hypervisor is pulling the rug out from under the OS's nose (de-scheduling the virtual processor) and the OS doesn't know it. This is why we must pull information out of the hypervisor to know what is really going on.

Also note how the heavy processing inside of one VM is being spread between multiple virtual processors. This will occur even if processor affinity is used in the VM, as is in the example Figure 4.

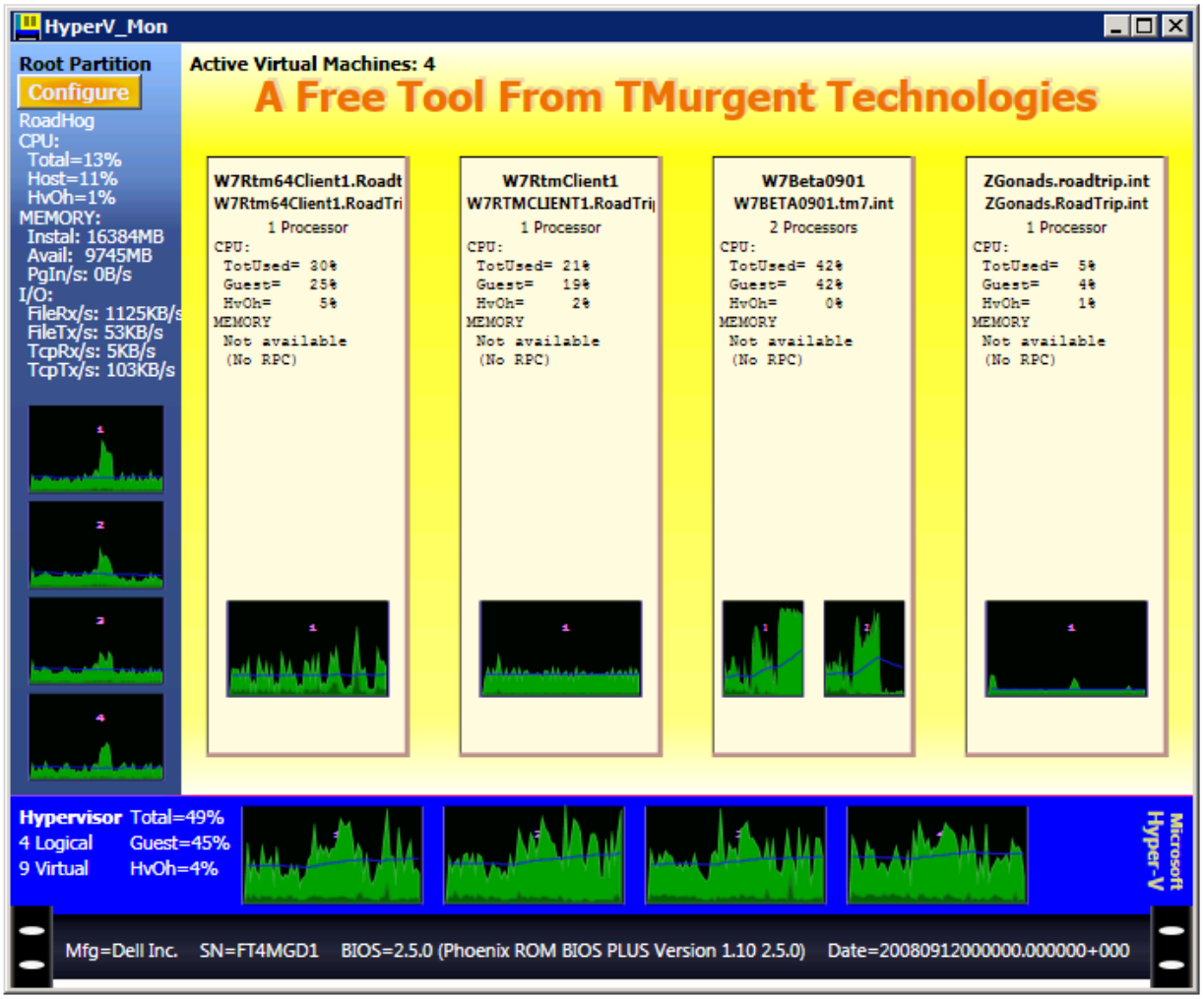

Figure 4 - Example with Processor Affinity

In this case, a fully consuming single threaded process started in the VM w7Beta-9-1, a dual-processor VM. After running for about 10 seconds, we applied processor affinity to the single-threaded process

inside the VM, assigning it to (virtual) processor 0. As can be seen in the VM graphs, the load changes from being split between the VMs virtual processors to consuming only virtual processor 0, and yet at the hypervisor level we see no effect on the loading of physical (logical) processors.

This has implications on running Hyper-V on a NUMA (Non-Uniform Memory Access) system. If you are using a NUMA hardware platform you might want to consider reading up on appropriate techniques to associate VMs and VM memory to specific NUMA groups (c.f. "*Windows Server 2008 Hyper-V Resource Kit"*, Microsoft Press).

With this tool, it is easy to understand why Windows XP and earlier OSs significantly underperforms on Hyper-V in comparison to Windows Server 2003, or Vista, or above. You will see significantly more hypervisor "overhead" processing with the older operating systems because Microsoft made modifications to the newer operating systems to be hypervisor (and processor VT) aware. Those newer OSs do not display such behavior. It is all in the enlightenments! In my setup, it is common to see heavy hypercall CPU usage with heavy IO usage in the XP VMs as is seen in Figure 5. The amount of overhead you see with Windows XP is not likely to be as high as is shown in this example. This is under investigation (there may be a CPU/chip revision level involved in my setup), and is why I made this tool in the first place!

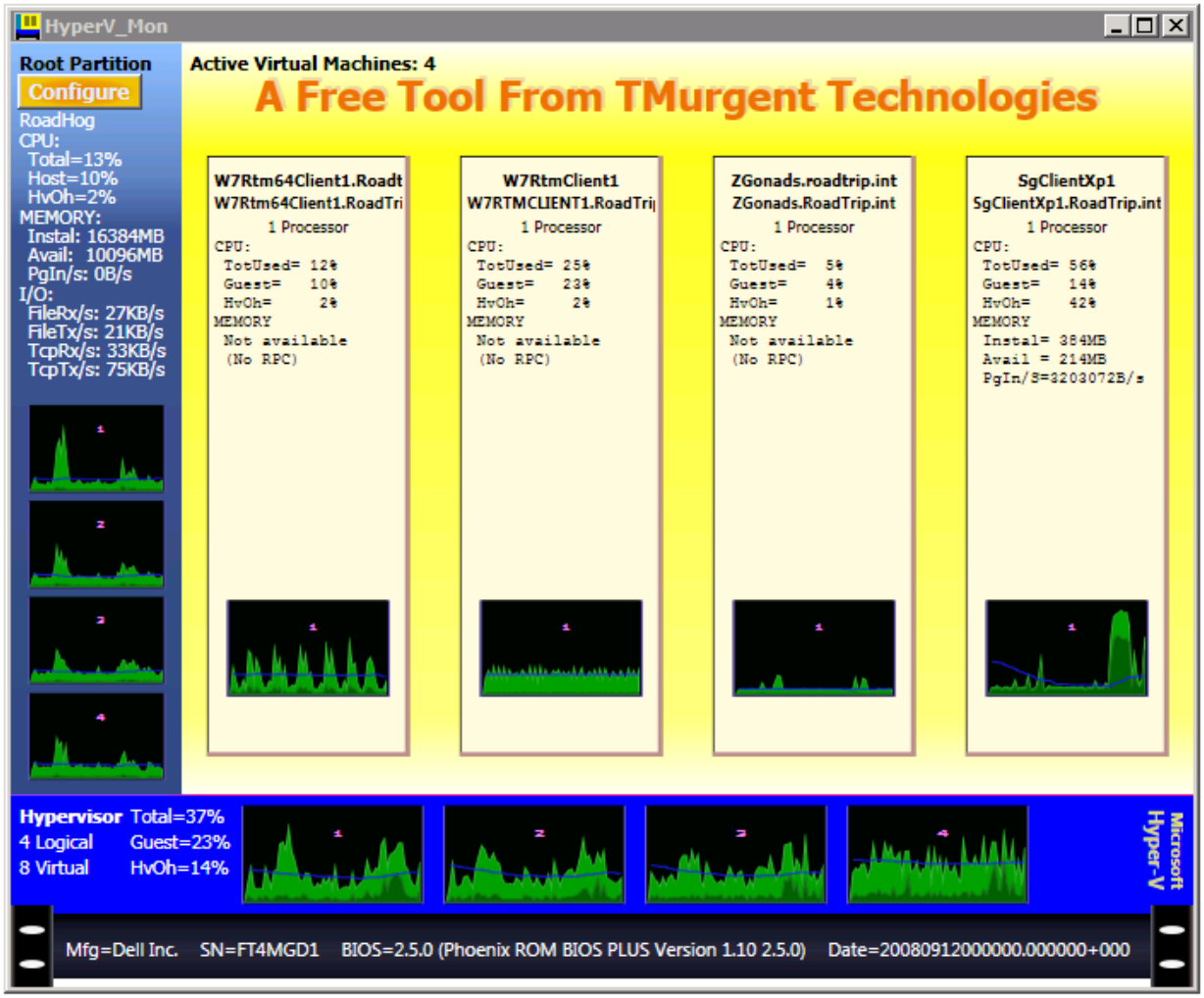

Figure 5 - Windows XP Hypervisor Load Example

New Stuff

In addition to making the graphs show historical information and the rolling average blue line, you can now hover the mouse over both graphs and the many text sections to see more detail information in "tooltips".

Memory related information has been added, both for the root partition and the VMs (when WMI/RPC access is allowed). This information is focused on three key values to help you decide if you have too much or too little memory installed in the VM. Install indicates the number of Megabytes physically installed in the case of the root partition, or allocated to the VM in the case of VMs. Available memory is the portion of the installed memory considered "available" by the OS (the technical definition of available memory varies in some OSs, but consider it as memory available to an application on demand without requiring existing memory to be paged out). Note that Memory allocated to running VMs are also shown as consumed memory in the host partition OS. The host and VM partitions also show the key statistic "PgIn/S". This represents the number of bytes per second read in from the disk due to hard page faults. Heavy paging activity in this counter indicates the need for more memory.

Finally, some File and TCP I/O counters have been added to the root partition display. File and network I/O together represent the third major performance concern for the server (after CPU and Memory).

Legalese

The product is free; has no support; is not warranted for any use. Feedback is encouraged. Gifts accepted gratefully.