Running vApus virtualised

Uit Sizing Servers Wiki

Inhoud

Introduction

Running time-sensitive software virtualised is problematic, since measured time on a VM can skew from the wall-clock time. Especially if the underlying hardware is under heavy load.

Luckily there is a solution, for a virtualised Windows that is. (I did not research Linux solutions)

In the first chapter I tell you what you must do to run vApus virtualised without explaining anything. Next I give pointers to proof and to more details that this approach works (according to the internet).

This, however, is not yet tested. We must run a vApus physical and virtualised and cross-reference stresstest results and behaviour to check if they are equal to one-another. And this under heavy load, for example ASAP, high concurrency, distributed with a lot of slaves,...

Documented details are about virtualising using VMware software. Running on Xen or Hyper-V should not be a problem since the TSC Invariant MSR is available through the Hypervisor[1][2].

Practical

The VM must be a Windows 8.1 or a Windows Server 2012 or newer.

Download CoreInfo from SysInternals[3] and run it from cmd on the VM (Administrator rights required).

The Machine Specific Register TSC-INVARIANT must be available (A * means available)[4]. Make sure that you've installed the VM...tools (vmware tools for example).

Install vApus and happy testing.

In depth

Windows

From Microsecond Resolution Time Services for Windows[5] and confirmed by Microsoft[6]:

> *Windows 8.1 goes back to the roots; it goes back to 156250. The TSC frequency is calibrated against HPET periods to finally get proper timekeeping.*

And

> *Since Windows 7, the operating system runs tests on the underlying hardware to see which hardware is best used for timekeeping. When the processors Time Stamp Counter (TSC) is suitable, the operating system uses the TSC for timekeeping. If the TSC cannot be used for timekeeping the operating system reverts to the High Precision Event Timer (HPET). If that does not exist it reverts to the ACPI PM timer. For performance reasons it shall be noted that HPET and ACPI PM timer cause IPC overhead, while the use of the TSC does not. The evolution of TSC shows a variety of capabilities:*
>
> - Constant: The TSC does not change with CPU frequency changes, however it does change on C state transitions.
> - Invariant: The TSC increments at a constant rate in all ACPI P-, C- and T-states.
> - Nonstop: The TSC has the properties of both Constant and Invariant TSC

This means that the Stopwatch class used for time measurements in vApus 's Connection Proxies, when vApus runs on Windows 7 / 8 can be correct and on => Windows 8.1, always work.

The Stopwatch class uses the QueryPerformanceCounter win32 API function internally. QPC uses TSC if the TSC-INVARIANT MSR (Machine Specific Register) is available, or falls back to HPET or PM timer (outdated, slower and not correct on VMs obviously)[6].

If we need more precision we can use the GetSystemTimePreciseAsFileTime function on Windows 8.1[5]. But I think that the Stopwatch class should be precise enough. This depends on the hardware [5][7].

> *Windows 8 introduces the function GetSystemTimePreciseAsFileTime() "with the highest possible level of precision (<1us)". This seems the counterpart to the linux gettimeofday() function.*

(Virtualized) Hardware

From Microsoft[6]:

> …you can determine whether your processor has an invariant TSC by using one of these: the Coreinfo.exe utility from Windows Sysinternals checking the values returned by the CPUID instruction pertaining to the TSC characteristics the processor manufacturer's documentation The following shows the **TSC-INVARIANT** info that is provided by the Windows Sysinternals Coreinfo.exe utility (www.sysinternals.com). An asterisk means "True".

The TSC-INVARIANT MSR (a hardware counter for CPUs) must be available. This must be supported by the used virualisation software[8]. Presumably you must have the VM...tools installed for correct virtualization.

Check if TSC-INVARIANT is available using CoreInfo[3].

References

1. Magenheimer, D. Xen 4.3 TSC_MODE HOW-TO (http://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt). Retrieved on 2015, January, 2
2. Microsoft (2013, June,19). Hypervisor Top-Level Functional Specification 3.0a: Windows Server 2012 (http://www.microsoft.com/en-us/download/details.aspx?id=39289)
3. Russinovich, M (2014, August, 18). CoreInfo v3.31 (http://technet.microsoft.com/en-us/sysinternals/cc835722.aspx)
4. Achtemichuk, M (2013, October, 18). Microsoft Operating System Time Sources and Virtual Hardware 10 (http://blogs.vmware.com/vsphere/2013/10/microsoft-operating-system-time-sources-and-virtual-hardware-10.html)
5. Lentfer, A (2014, October). Microsecond Resolution Time Services for Windows, v1.80 (http://www.windowstimestamp.com/description)
6. Briggs, E (Presumed) (2014, September, 30). Acquiring high-resolution time stamps (http://msdn.microsoft.com/en-us/library/windows/desktop/dn553408%28v=vs.85%29aspx)
7. Microsoft. Stopwatch.Frequency Field, .Net Framework 4.5 (http://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch.frequency(v=vs.110).aspx)
8. Vinschen, C (2012, June, 15). GetSystemTimePreciseAsFileTime on pre-Windows 8? (http://social.msdn.microsoft.com/Forums/windowsdesktop/en-US/b0df67d7-8151-4ee3-9c58-343004243c1b/getsystemtimepreciseasfiletime-on-prewindows-8?forum=windowsgeneraldevelopmetissues)

vApus virtual vs vApus physical in numbers

Introduction

One physical Windows client / one or more virtual Windows clients where used to run vApus. This to check if the above is actually correct. The virtual client(s) run in our own production to simulate hardware usage in the real world (e.g. Amazon), where hardware is shared among a lot of VMs. Or on AWS.

Local

Each test was executed twice.

Test setup

Virtual client:

- Windows Server 2012 R2 @ [Istanbul (production) (ESXi 5.5)
- 2 vCPUs, AMD Istanbul CPUs, TSC-Invariant MSR available
- 6 GB RAM

Physical client:

- Windows Server 2012 R2 @ dell-r815

- 2 CPUs, 32 cores AMD Opteron 6276, TSC-Invariant MSR available
- 32 GB RAM

Test 1:

- Tales Anziplast (Liquifi)
- Users: 5, 5, ..., 100 in steps of 5
- 2 runs
- Delay: 900 - 1100 ms
- The client is self-monitored to make sure that it is not the bottleneck

Used vass: Bestand:Tales Anziplast.zip

Test 2:

- A Break on First Run Sync Distributed test
- 3 tile stress tests: benchdb, phpbb and nieuws.be
- Users: 5, 5, ..., 100 in steps of 5
- 2 runs
- Delay: 900 - 1100 ms
- The client is self-monitored to make sure that it is not the bottleneck

Used vass: Bestand:Dev and testing 20141027.zip

Test results on their own are irrelevant: The result with the virtual vApus must match (+/-) those with the physical vApus.

Results

Test 1

Both tests gave consistent results. Specifically the **throughput** for the concurrency where the server was not yet saturated was compared: Results for a saturated server will fluctuate more.

	vApus physical	vApus virtual
Execution 1 Concurrency 15	68,87	67,24
Execution 2 Concurrency 15	72,19	71,4

Bestand:VApus virtual vs physical Test 1.zip

Test 2

Both tests gave consistent results. Specifically the **throughput** for the concurrency where the server was not yet saturated was compared: Results for a saturated server will fluctuate more.

Results of the slowest test:

	vApus physical	vApus virtual
phpbb Execution 1 Concurrency 45	341,04	334,17
phpbb Execution 2 Concurrency 45	332,31	340,31

Bestand:VApus virtual vs physical Test 2.zip

AWS

Test setup

A more simple approach was chosen since there is already enough jitter on AWS: www.sizingservers.be was tested using only a single vApus instance.

Test 2 was executed trice to ensure that there are no big differences in the results.

Virtual client:

- Windows Server 2012 R2 @ AWS c3.large Ireland

Physical client:

- Windows Server 2012 R2 @ dell-r815

Test 1:

- www.sizingservers.be
- Users: 5, 10, 15
- 2 runs
- Delay: 900 - 1100 ms
- The client is self-monitored to make sure that it is not the bottleneck

Test 2:

- www.sizingservers.be
- Users: 5, 10, 15
- 2 runs + ACTION DISTRIBUTION (all actions times 10)
- Delay: 900 - 1100 ms
- The client is self-monitored to make sure that it is not the bottleneck

Used vass: Bestand:VApus v2 AWS.zip

Test results on their own are irrelevant: The result with the virtual vApus must match (+/-) those with the physical vApus.

Results

As you can clearly see in the spreadsheets results are as they should be. Testing from AWS is a bit slower because there is a latency of 22 ms to www.sizingservers.be.

www. sizingservers.be is located in our own network. You can find this latency back in the average response times.

Excel spreadsheets:

Bestand:From Dell-r815.xlsx

Bestand:From AWS.xlsx

Conclusion

It is safe to say that vApus can be used on VMs and in the cloud.

Overgenomen van "https://wiki.sizingservers.be/index.php?title=Running_vApus_virtualised&oldid=15775"

- Deze pagina is het laatst bewerkt op 15 jul 2015 om 09:55.