Who hogs down my CPU?

Adi Oltean 21 Dec 2004 7:00 PM | 6

I always wanted to find a simple way to figure out the answer for this puzzling question. Usually I ended up opening Task Manager to find out what process eats all my CPU resources. But this doesn't work in most of the cases. For example, what about the case when the "System process" (i.e. a kernel-mode component) is eating all my CPU?

But I just found a true gem called Kernrate. This tool (available for free download here) does present accurate CPU statistics at API level! Even in Kernel mode.

All you have to do is to run Kernrate_i386_XP.exe from the Kernrates directory, and then press Ctrl-C after a while. You get all sorts of interesting statistics (context switches, interrupts per second, etc.) like the ones below. But the most interesting part is the CPU percentage spent in all the kernel-mode components, in decreasing order of consumed CPU time. Below, I ran Kernrate while doing a recursive DIR /S C: on a separate command shell:

```
C:\KrView\Kernrates>Kernrate_i386_XP.exe
 /==============================\
<         KERNRATE LOG           >
 \==============================/
Date: 2004/12/21   Time: 15:33:21
Machine Name: AOLTEAN-H4
Number of Processors: 1
PROCESSOR_ARCHITECTURE: x86
PROCESSOR_LEVEL: 6
PROCESSOR_REVISION: 0800
Physical Memory: 480 MB
Pagefile Total: 1125 MB
Virtual Total: 2047 MB
PageFile1: \??\E:\pagefile.sys, 720MB
OS Version: 5.1 Build 2600 Service-Pack: 2.0
WinDir: E:\WINDOWS

Kernrate User-Specified Command Line:
Kernrate_i386_XP.exe


Kernel Profile (PID = 0): Source= Time,
Using Kernrate Default Rate of 25000 events/hit
Starting to collect profile data

***> Press ctrl-c to finish collecting profile data
===> Finished Collecting Data, Starting to Process Results

------------Overall Summary:--------------

P0     K 0:00:13.656 (38.2%)  U 0:00:02.484 ( 7.0%)  I 0:00:19.578 (54.8%)  DPC
0:00:00.312 ( 0.9%)  Interrupt 0:00:00.296 ( 0.8%)
       Interrupts= 107928, Interrupt Rate= 3022/sec.


Total Profile Time = 35718 msec

                                       BytesStart          BytesStop
BytesDiff.
    Available Physical Memory   ,       107065344,       112259072,          5193728
    Available Pagefile(s)       ,       374460416,       371945472,         -2514944
    Available Virtual           ,      2132889600,      2131841024,         -1048576
    Available Extended Virtual  ,               0,               0,                0

                                  Total      Avg. Rate
    Context Switches     ,       395167,         11063/sec.
    System Calls         ,      1106131,         30968/sec.
    Page Faults          ,        20595,         577/sec.
    I/O Read Operations  ,         2346,         66/sec.
    I/O Write Operations ,         1102,         31/sec.
    I/O Other Operations ,        32185,         901/sec.
    I/O Read Bytes       ,       346338,         148/ I/O
    I/O Write Bytes      ,        83614,         76/ I/O
    I/O Other Bytes      ,      5832690,         181/ I/O

-----------------------------

Results for Kernel Mode:
-----------------------------

OutputResults: KernelModuleCount = 135
```

```
Percentage in the following table is based on the Total Hits for the Kernel

Time   13080 hits, 25000 events per hit --------
 Module                                Hits   msec  %Total  Events/Sec
amdk7                                  7381      35718    56 %     5166162
nv4_disp                               3834      35718    29 %     2683520
ntoskrnl                                819      35718     6 %      573240
win32k                                  341      35718     2 %      238675
hal                                     288      35718     2 %      201579
Ntfs                                    180      35718     1 %      125986
NVENET                                   64      35718     0 %       44795
USBPORT                                  38      35718     0 %       26597
atapi                                    20      35718     0 %       13998
ino_fltr                                 19      35718     0 %       13298
nv4_mini                                 17      35718     0 %       11898
usbohci                                  11      35718     0 %        7699
watchdog                                  9      35718     0 %        6299
tcpip                                     7      35718     0 %        4899
HIDPARSE                                  7      35718     0 %        4899
Npfs                                      5      35718     0 %        3499
HIDCLASS                                  4      35718     0 %        2799
NDIS                                      4      35718     0 %        2799
sr                                        4      35718     0 %        2799
ftdisk                                    4      35718     0 %        2799
usbhub                                    3      35718     0 %        2099
PCIIDEX                                   3      35718     0 %        2099
ACPI                                      3      35718     0 %        2099
mouhid                                    2      35718     0 %        1399
hidusb                                    2      35718     0 %        1399
mouclass                                  2      35718     0 %        1399
TDI                                       2      35718     0 %        1399
PartMgr                                   2      35718     0 %        1399
rdbss                                     1      35718     0 %         699
psched                                    1      35718     0 %         699
VIDEOPRT                                  1      35718     0 %         699
imapi                                     1      35718     0 %         699
CLASSPNP                                  1      35718     0 %         699

================================= END OF RUN ==================================
============================== NORMAL END OF RUN ==============================
```

Funny enough, in the code above we can see that a third-party driver on my machine - nv4_disp.sys - which is consuming 29% of my CPU. (**update**: this driver is related to my NVidia chipset). But anyway, this is not a truly relevant test because in a "dir /s" the bottleneck is not the CPU but the I/O needed to read the file system metadata into the cache.

Going back to our Kernrate - you can even zoom in a certain module to find more, if you use the "-z <module>" option. This command zooms into the Windows kernel (ntoskrnl.exe):

```
C:\KrView\Kernrates>Kernrate_i386_XP.exe -z ntoskrnl
 /==============================\
<         KERNRATE LOG           >
 \==============================/
Date: 2004/12/21   Time: 15:37:38
Machine Name: AOLTEAN-H4
Number of Processors: 1
PROCESSOR_ARCHITECTURE: x86
PROCESSOR_LEVEL: 6
PROCESSOR_REVISION: 0800
Physical Memory: 480 MB
Pagefile Total: 1125 MB
Virtual Total: 2047 MB
PageFile1: \??\E:\pagefile.sys, 720MB
OS Version: 5.1 Build 2600 Service-Pack: 2.0
WinDir: E:\WINDOWS

Kernrate User-Specified Command Line:
Kernrate_i386_XP.exe -z ntoskrnl


Kernel Profile (PID = 0): Source= Time,
Using Kernrate Default Rate of 25000 events/hit
CallBack: Finished Attempt to Load symbols for 804d7000
\WINDOWS\system32\ntoskrnl.exe

Starting to collect profile data

***> Press ctrl-c to finish collecting profile data
===> Finished Collecting Data, Starting to Process Results
```

```
------------Overall Summary:--------------

P0     K 0:00:01.406 (24.3%)  U 0:00:00.859 (14.8%)  I 0:00:03.531 (60.9%)  DPC
0:00:00.031 ( 0.5%)  Interrupt 0:00:00.062 ( 1.1%)
       Interrupts= 23804, Interrupt Rate= 4106/sec.


Total Profile Time = 5796 msec

                                       BytesStart          BytesStop
BytesDiff.
    Available Physical Memory   ,       117850112,       114122752,        -3727360
    Available Pagefile(s)       ,       370819072,       368578560,        -2240512
    Available Virtual           ,      2132889600,      2130681856,        -2207744
    Available Extended Virtual  ,               0,               0,               0

                                  Total      Avg. Rate
    Context Switches     ,       206440,         35612/sec.
    System Calls         ,       372915,         64330/sec.
    Page Faults          ,        14872,         2566/sec.
    I/O Read Operations  ,          186,         32/sec.
    I/O Write Operations ,          180,         31/sec.
    I/O Other Operations ,        19183,         3309/sec.
    I/O Read Bytes       ,        39296,         211/ I/O
    I/O Write Bytes      ,        11940,         66/ I/O
    I/O Other Bytes      ,      3000748,         156/ I/O

-----------------------------

Results for Kernel Mode:
-----------------------------

OutputResults: KernelModuleCount = 135
Percentage in the following table is based on the Total Hits for the Kernel

Time   1893 hits, 25000 events per hit --------
 Module                              Hits   msec  %Total  Events/Sec
amdk7                                1309       5796    69 %     5646135
ntoskrnl                              292       5796    15 %     1259489
hal                                    91       5796     4 %      392512
Ntfs                                   79       5796     4 %      340752
win32k                                 56       5796     2 %      241545
NVENET                                 28       5796     1 %      120772
ino_fltr                                8       5796     0 %       34506
atapi                                   6       5796     0 %       25879
CLASSPNP                                4       5796     0 %       17253
Npfs                                    3       5796     0 %       12939
PCIIDEX                                 3       5796     0 %       12939
watchdog                                2       5796     0 %        8626
nv4_mini                                2       5796     0 %        8626
sr                                      2       5796     0 %        8626
PartMgr                                 2       5796     0 %        8626
ftdisk                                  2       5796     0 %        8626
nv4_disp                                1       5796     0 %        4313
tcpip                                   1       5796     0 %        4313
USBPORT                                 1       5796     0 %        4313
NDIS                                    1       5796     0 %        4313

===> Processing Zoomed Module ntoskrnl.exe...


----- Zoomed module ntoskrnl.exe (Bucket size = 16 bytes, Rounding Down) --------
Percentage in the following table is based on the Total Hits for this Zoom Module

Time   292 hits, 25000 events per hit --------
 Module                              Hits   msec  %Total  Events/Sec
CcUnpinDataForThread                   32       5796    10 %      138026
KiDispatchInterrupt                    27       5796     8 %      116459
ZwYieldExecution                       18       5796     5 %       77639
FsRtlIsNameInExpression                14       5796     4 %       60386
KiIpiServiceRoutine                    13       5796     4 %       56073
SeUnlockSubjectContext                  9       5796     2 %       38819
NtAllocateVirtualMemory                 8       5796     2 %       34506
ObReferenceObjectByHandle               8       5796     2 %       34506
ExAllocatePoolWithTag                   8       5796     2 %       34506
NtRequestWaitReplyPort                  7       5796     2 %       30193
ExInterlockedPopEntrySList              7       5796     2 %       30193
SeDeleteAccessState                     6       5796     1 %       25879
ExAcquireResourceExclusiveLite          6       5796     1 %       25879
ExReleaseResourceLite                   6       5796     1 %       25879
```

```
NtOpenProcessTokenEx                    5       5796      1 %       21566
ObCreateObject                          5       5796      1 %       21566
ObfDereferenceObject                    5       5796      1 %       21566
wcstombs                                4       5796      1 %       17253
MmMapLockedPagesSpecifyCache            4       5796      1 %       17253
IoBuildPartialMdl                       4       5796      1 %       17253
RtlCopyUnicodeString                    4       5796      1 %       17253
```

This time nv4_dist was not a huge time-consuming module. but ntoskrnl was in the second place instead. We notice here KiDispatchInterrupt (which is probably the atapi!IdePortInterrupt interrupts). We also notice CcUnpinDataForThread, which denotes some cache manager data access (most likely for the cached NTFS metadata - the $MFT streams in special), and FsRtlIsNameInExpression which is the routine that matches file names to DOS pattern expressions in NTFS. In other words, it appears that we use FindFirstFile/FindNextFile which is not a surprise since the command that was run was DIR /S. It is however surprising that we spent 15% * 4% = 0.6% of kernel time when no real pattern was used in our DIR command.

There is also an Excel file called KrView.xls that allows you to generate all sorts of graphical summaries.

Anyway, this was only a quick tour. Download kernrate for yourself and enjoy!

P.S. There is also managed code support but you need a DLL which is not part of kernrate normally.

Comments

Damit 21 Dec 2004 9:49 PM #
Excellent post (and thanks for the link!) but I just wanted to comment that nv4_disp is not an antivirus driver, but rather the NVidia display driver.

IMO, it would make sense for the display driver to be using 29% of the CPU on a dir /s.

Adam 21 Dec 2004 11:13 PM #

Not to be too blunt but this tool requires a fair amount of knowledge to interpret correctly.

Btw: the amdk7 is there because the idle routines that the kernel uses when there are no runnable threads are in that binary.

余啊雷 22 Dec 2004 1:14 AM #

Adi Oltean 22 Dec 2004 2:33 AM #
>>> Not to be too blunt but this tool requires a fair amount of knowledge to interpret correctly.

Agree. But on the other side, there are some scenarios where this tool will give you immediately a feeling on where the "hot spot" is, for example if you have a certain type of bug in your driver that is causing performance problems. Say, for example, that you have a bogus while loop in that is consuming CPU - this type of tool will spot this bug right away.

余啊雷 3 May 2006 11:33 PM #
I've 3 new tricks to to add to a few of my earlier postings:
1.&amp;nbsp; In my batch file for VSTS profiling...

余啊雷 24 Mar 2008 11:41 PM #
PingBack from http://caferestaurantsblog.info/antimail-who-hogs-down-my-cpu/