AMD 64 bit Dual Core Optimization

Asked 16 years, 3 months ago Modified 10 years, 1 month ago

Viewed 1k times

0

We have a graphics intensive application that seems to be experiencing problems on AMD 64 bit Dual Core platforms that are not apparent on Intel platforms.

Running the application causes the CPU to run at 100%, in particular when using code for shadows and lighting (Open GL).

Does anyone know of specific issues with AMD processors that could cause this or know where to track down the problem, and/or ways to optimize the code base to avoid these issues?

note, the application generally works well on mid range hardware, my dev machine has an nvidia gtx260 card in, so lack of power should not be an issue

opengl graphics 64-bit cpu amd-processor

Share

Improve this question

Follow

edited Nov 18, 2014 at 0:31

Pokechu22

5,050 ● 9 ● 38 ● 64

asked Sep 19, 2008 at 10:24

Ian Wilson
61 ● 3 ● 7

6 Answers

Sorted by: Highest score (default)

2

Note that AMD64 is a NUMA architecture - if you are using a multi-processor box you may be running lots of memory accesses across the hypertransport bus which will be slower than the local memory and may explain the behaviour.

This will not be the case between cores on a single socket so feel free to ignore this if you are not using a multiple-socket machine.

Linux is NUMA aware (i.e. it has system services to allocate memory by local bank and bind processes to specific CPU's). I believe that Win 2k3 server, 2k8 and Vista are NUMA aware but XP is not. Most of the proprietary unix variants such as Solaris have NUMA support as well.

Share Improve this answer Follow

answered Sep 22, 2008 at 12:35

ConcernedOfTunbridge Wells
66.5k ● 15 ● 148 ● 198

1

Late answer here.

Dunno if this is related, but in some win32 OpenGL drivers, SwapBuffers() will not yield the CPU while waiting for vsync, making it very easy to get 100% CPU utilisation.

The solution I use to this is to measure the time since the last SwapBuffers() completed, which tells me how far away the next vsync is. So before calling SwapBuffers(), I take short Sleep()s until I detect that vsync is imminent. This way SwapBuffers() doesn't have to wait long for vsync, and so doesn't hog the CPU too badly.

Note that you may have to use timeBeginPeriod() to get sufficient Sleep() precision for this to work reliably.

Share Improve this answer Follow

answered Sep 26, 2008 at 20:39
Mike F

1

Depending on how you've done your shadows and other graphics code, it possible that youve "fallen off the fast path" and the graphics driver has started doing software emulation. This can happen if you have complicated pipelines, or are using too many conditionals (or just too many instructions) in shader code.

I would make sure that this particular graphics card supports all the features you are using.

Share Improve this answer Follow

answered Nov 21, 2008 at 17:28
luke
14.8k ● 5 ● 49 ● 57

0

I would invest in profiling software to trace down the actual cause of the problem.

On linux, Valgrind (which contains Cachegrind & Callgrind) + KCacheGrind can make working out where all the heavy function calls are going on.

Also, compile with full debug symbols and it can even show the assembley code at the slow function calls.

If you're using an Intel Specific compiler, this may be part of your problem (not definate tho), and try the GCC family.

Also, you may want to dive into OpenMP and Threads if you haven't already.

Share Improve this answer Follow

answered Sep 19, 2008 at 10:26

Kent Fredric
57.3k ● 14 ● 111 ● 151

0

Hm - if you use shadows the GPU should be under load, so it's unlikely that the GPU renders the frames faster than the CPU sends graphic data. In this case 100% load is ok and even expected.

It could simply be a borked OpenGL driver that does burns CPU-cycles in a spinlock somewhere. To find out what's exactly going on I suggest you run a profiling tool such as Code Analyst from AMD (free last time I've used it).

Profile your program a couple of minutes and take a look where the time is spent. If you see a big peak in the opengl drivers and not in your application get a new driver. Otherwise you at least get an idea what's going on.

Btw - let me guess, you're using an ATI card, right? I don't want to offend any ATI fans out there, but their OpenGL-drives are not exactly stellar. If you're unlucky you may even used a feature that the card does not support or that is disabled due to a silicon bug. The driver will fallback into software rasterization mode in this case. This will slow down things a lot and give you a 100% CPU-Load even if your program is single-threaded.

edited Sep 19, 2008 at 10:44

answered Sep 19, 2008 at 10:33

Nils Pipenbrinck
86.2k ● 33 ● 155 ● 223

0

Also the cache is not shared, which might cause a lack of performance when sharing data among multiple threads.

Share Improve this answer Follow

answered Sep 22, 2008 at 12:46

Ronny Brendel
4,835 ● 5 ● 37 ● 55