CPU throttling in C++

Asked 16 years, 4 months ago Modified 1 year, 6 months ago

Viewed 6k times

49

I was just wondering if there is an elegant way to set the maximum CPU load for a particular thread doing intensive calculations.

Right now I have located the most time-consuming loop in the thread (it does only compression) and use `GetTickCount()` and `Sleep()` with hardcoded values. It makes sure that the loop continues for a certain period and then sleeps for a certain minimum time. It more or less does the job, i.e. guarantees that the thread will not use more than 50% of CPU.

However, behavior is dependent on the number of CPU cores (huge disadvantage) and simply ugly (smaller disadvantage).

Any ideas?

c++ performance cpu throttling

Share

Improve this question

Follow

edited Jun 1, 2023 at 12:05

digito_evo

3,652 ● 3 ● 19 ● 52

asked Aug 5, 2008 at 7:11

Piotr Tyburski
651 ● 6 ● 11

2 What visible behavior do you want to achieve? That is, what does this watchdog want from your threads? Should they by no means use more than, say, 80% of the CPU? Can setting process base priority to Idle possibly calm the WD down?
– wordmonger Aug 11, 2008 at 19:52

5 Answers

Sorted by: Highest score (default)

22

I am not aware of any API to do get the OS's scheduler to do what you want (even if your thread is idle-priority, if there are no higher-priority ready threads, yours will run). However, I think you can improvise a fairly elegant throttling function based on what you are already doing. Essentially (I don't have a Windows dev machine handy):

Pick a default amount of time the thread will sleep each iteration. Then, on each iteration (or on every nth iteration, such that the throttling function doesn't itself become a significant CPU load),

1. Compute the amount of CPU time your thread used since the last time your throttling function was called (I'll call this dCPU). You can use the GetThreadTimes() API to get the amount of time your thread has been executing.
2. Compute the amount of real time elapsed since the last time your throttling function was called (I'll call

this dClock).

3. dCPU / dClock is the percent CPU usage (of one CPU). If it is higher than you want, increase your sleep time, if lower, decrease the sleep time.
4. Have your thread sleep for the computed time.

Depending on how your watchdog computes CPU usage, you might want to use GetProcessAffinityMask() to find out how many CPUs the system has. dCPU / (dClock * CPUs) is the percentage of total CPU time available.

You will still have to pick some magic numbers for the initial sleep time and the increment/decrement amount, but I think this algorithm could be tuned to keep a thread running at fairly close to a determined percent of CPU.

Share Improve this answer Follow

answered Sep 5, 2008 at 23:33

Matthew Xavier
2,118 ● 1 ● 13 ● 18

If your goal is to avoid wasting CPU time, a cheaper heuristic is probably a better choice. Depending on how much of a problem it is for your thread to starve itself under high system load, you could *just* check elapsed wall-clock time. On x86, this is very cheap, because time functions based on `rdtsc` don't even need to enter kernel mode. Making multiple system calls even `n` iterations is worse than making just one, unless it lets you increase `n` by a lot and still get the behaviour you want. – Peter Cordes May 11, 2016 at 7:57

5

On linux, you can change the scheduling priority of a thread with nice().

Share Improve this answer

Follow

answered Aug 5, 2008 at 8:03

Mark Harrison

304k ● 131 ● 350 ● 489

Other platforms have similar features, see also: stackoverflow.com/questions/18884510/… I think this can be a nice solution to the problem, albeit with different semantics - i.e. no 50% CPU consumption guarantee – milianw Jan 8, 2018 at 9:03

3

The problem is it's not normal to want to leave the CPU idle while you have work to do. Normally you set a background task to IDLE priority, and let the OS handle scheduling it all the CPU time that isn't used by interactive tasks.

It sound to me like the problem is the watchdog process.

If your background task is CPU-bound then you want it to take all the unused CPU time for its task.

Maybe you should look at fixing the watchdog program?

Share Improve this answer

Follow

answered Sep 6, 2008 at 9:01

Douglas Leeder

53.3k ● 9 ● 99 ● 138

1 It's very reasonable to want an idle CPU. Maybe you want to do some computation but don't care how fast it gets done, as long as it does not spin up the CPU fan on your laptop. – Ringding Sep 21, 2009 at 10:32

@Ringding: In that case you'd want the CPU frequency "governor" to leave the CPU frequency lowish while running only this thread; that would give better work per energy than alternating sleep/wake, especially on CPUs that can turbo to high clocks for short bursts. But I guess that might not work if there is higher-prio work to do on other cores, if the system can't run some cores at low frequency while others are at max frequency. (I think that's the case on most Intel laptop/desktop CPUs, other than "turbo" above the rated max all-cores sustained frequency.) – Peter Cordes Jun 1, 2023 at 16:14

2

I can't think of any cross platform way of what you want (or any guaranteed way full stop) but as you are using GetTickCount perhaps you aren't interested in cross platform :)

I'd use interprocess communications and set the intensive processes nice levels to get what you require but I'm not sure that's appropriate for your situation.

EDIT: I agree with Bernard which is why I think a process rather than a thread might be more appropriate but it just might not suit your purposes.

Share Improve this answer edited May 23, 2017 at 12:17

Follow

Community Bot
1 ● 1

answered Aug 5, 2008 at 7:23

sparkes
19.5k ● 5 ● 40 ● 46

0

You may be able to change the priority of a thread, but changing the maximum utilization would either require polling and hacks to limit how many things are occurring, or using OS tools that can set the maximum utilization of a process. However, I don't see any circumstance where you would want to do this.

Share Improve this answer

Follow

answered Jan 3, 2018 at 18:06

Aashishkebab
341 ● 4 ● 10