When should I consider changing thread priority

Asked 16 years, 3 months ago Modified 16 years, 3 months ago

Viewed 5k times

7

I once was asked to increase thread priority to fix a problem. I refused, saying that changing it was dangerous and was not the root cause of the problem.

My question is, under what circumstannces *should* I conider changing priority of threads?

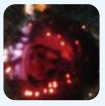
multithreading language-agnostic

Share

Improve this question

Follow

asked Sep 18, 2008 at 18:53



theschmitzer

12.8k ● 11 ● 42 ● 49

5 Answers

Sorted by: Highest score (default)

6

When you've made a list of the threads you're using and defined a priority order for them which makes sense in terms of the work they do.

If you nudge threads up here and there in order to bodge your way out of a problem, eventually they'll all be high



priority and you're back where you started. Don't assume you can fix a race condition with prioritisation when really it needs locking, because chances are you've only fixed it in friendly conditions. There may still be cases where it can fail, such as when the lower-priority thread has undergone priority inheritance because another high-priority thread is waiting on another lock it's holding.

If you classify threads along the lines of "these threads fill the audio buffer", "these threads make my app responsive to system events", "these threads make my app responsive to the user", "these threads are getting on with some business and will report when they're good and ready", then the threads ought to be prioritised accordingly.

Finally, it depends on the OS. If thread priority is completely secondary to process priority, then it shouldn't be "dangerous" to prioritise threads: the only thing you can starve of CPU is yourself. But if your high-priority threads run in preference to the normal-priority threads of other, unrelated applications, then you have a broader responsibility. You should only be raising priorities of threads which do small amounts of urgent work. The definition of "small" depends what kind of device you're on - with a 3GHz multi-core processor you get away with a lot, but a mobile device might have pseudo real-time expectations that user-level apps can break.

Keeping the audio buffer serviced is the canonical example of when to be high priority, though, since small

under-runs usually cause nasty crackling. Long downloads (or other slow I/O) are the canonical example of when to be low priority, since there's no urgency processing this chunk of data if the next one won't be along for ages anyway. If you're ever writing a device driver you'll need to make more complex decisions how to play nicely with others.

Share Improve this answer Follow

edited Sep 18, 2008 at 19:12

answered Sep 18, 2008 at 19:00

Steve Jessop

279k ● 40 ● 469 ● 709

2

Not many. The only time I've ever had to change thread priorities in a *positive* direction was with a user interface thread. UIs must be extremely snappy in order for the app to feel right, so a lot of times it is best to prioritize painting threads higher than others. For example, the Swing Event Dispatch Thread runs at priority 6 by default (1 higher than the default).

I do push threads down in priority quite a bit. Again, this is usually to keep the UI responsive while some long-running background process does its thing. However, this also will sometimes apply to polling daemons and the like which I *know* that I don't want to be interfering with anything, regardless of how minimal the interference.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:56

Daniel Spiewak

55.1k ● 14 ● 111 ● 120

1

Our app uses a background thread to download data and we didn't want that interfering with the UI thread on single-core machines, so we deliberately prioritized that lower.

Share Improve this answer

Follow

answered Sep 18, 2008 at 19:00

Bob King

25.8k ● 7 ● 57 ● 66

1

I think it depends on the direction you're looking at changing the priority.

Normally you shouldn't ever increase thread priority unless you have a *very* good reason. Increasing thread priority can cause your app's thread to start taking away time from other applications, which probably isn't what the user wants. If your thread is using up a significant amount of CPU it can make the machine hard to use, as some standard UI threads may start to starve.

I'd say the only times you should increase priority above normal is if the user explicitly told your app to do so, but even then you want to prevent "clueless" users from doing so. Maybe if your app doesn't use much CPU normally, but might have brief bursts of really really important activity then it could be OK to have an

increased priority, as it wouldn't normally detract from the user's general experience.

Decreasing priority is another matter. If your app is doing something that takes a LOT of CPU and runs for a long time, yet isn't critical, then lowering the priority can be good. By lowering the priority you allow the CPU to be used for other things when it's needed, which helps keep the system responding quickly. As long as the system is mostly idling other than your app you'll still get most of the CPU time, but won't take away from tasks that need it more than you. An example of this would be a thread that indexes the hard drive (think google desktop).

Share Improve this answer

Follow

answered Sep 18, 2008 at 19:01

Herms

38.7k ● 13 ● 79 ● 104

0

I would say when your original design assumptions about the threads are no longer valid.

Thread priority is mostly a design decision about what work is most important. So for some examples of when to reconsider: If you add a new feature that might require its own thread that becomes more important, then reconsider thread priorities. If some requirements change that force you to reconsider the priorities of the work you are doing, then reconsider. Or, if you do performance testing and realize that your "high priority work" as specified in your design do not get the required performance, then tweak priorities.

Otherwise, its often a hack.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:58

Doug T.

65.5k ● 28 ● 141 ● 205