

Forum:[Hack Forums](#)

Thread: [Optimization by reduction](#)

iconia - August 11, 2026, 3:03 pm

One thing that's changed the way I think about performance is realizing the fastest computation isn't a faster computation. It's the one you prove you don't have to do again.

People usually jump straight to threads, GPUs, SIMD, faster hardware, etc.

But if a workload has enough repeated structure, sometimes the bigger win is identifying what is actually invariant and only recomputing what changed.

Sounds obvious when you say it out loud, but I think a ridiculous amount of software still brute-forces basically the same work over and over again.

Been fascinated with this lately, has anyone been hyper-optimized with that line of process?

Terry A. Davis - August 11, 2026, 3:05 pm

Most people just understand that working is better than quick, unless you have a need for speed you're usually better to just get it working, then of course you remove the Thread.Sleep calls from your code ;)

iconia - August 11, 2026, 3:22 pm

(Aug 11, 2026 03:05 PM)Terry A. Davis Wrote: [\[CITE\]](#) Most people just understand that working is better than quick, unless you have a need for speed you're usually better to just get it working, then of course you remove the Thread.Sleep calls from your code ;)

totally agree, over preemptive optimization has hurt me more in that long run, get it working first, optimize second, collect my thread.sleep calls and speed up

Terry A. Davis - August 11, 2026, 3:35 pm

Pretty much - You shouldn't design for speed unless it's a need of the application.

Optimization comes from understanding the repeat process and how you either shortcut the process, or better still, see the way it works underlying so you can mitigate what steps you can.

iconia - August 11, 2026, 3:37 pm

yep, totally agree