

Re: True Memory Use

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2005-02/1241.html>

From: Joseph M. Newcomer (*newcomer_at_flounder.com*)

Date: 02/14/05

Date: Mon, 14 Feb 2005 01:40:45 -0500

THIS is a completely different question. What you really have here is a question of whether or not a single processor can sustain a massive influx of data (for example, what happens at peak data rates). We had to actually measure our server (it can handle 1300 packets a second, at which point it saturates). So the real metric is to be able to create a simulated load and note when the system saturates. That's the real measure (that's what we did; I had four other machines on the network pumping packets at maximum rate). Trying to figure this out a priori by some sort of static analysis of the program is not going to be informative. You have to measure it under load. That's the only way to answer your question. If the computer can handle a given load, in a given configuration, the answer is "yes"; if it can't the answer is "no". And it means you have to assume that the end-user machine has no other load on it. If you need to run IIS, or Apache, or SQL Server, or you are just a networking router, that's a different environment. If you double the RAM, that's a different environment. If you put a faster disk on, that's a different environment. But no amount of desktop analysis is going to tell you that you can run n copies of a server with particular load conditions on a particular machine configuration; only live measurement (ideally with a simulated load that you can control to simulate any load you want) is going to answer the question.

Then, if you get a "no" on the configuration you want, you start doing work on, then you have to start asking questions about how to improve the performance. At that point, you start looking at working set size, paging behavior, and the like; once that is factored out, start looking at optimizing the code itself, starting with the algorithms and then working down to lines of code. The last time I did something like this where we needed to tune the performance, I spent several months working on it. I would say that at least one month of effort, spread over that interval, was building the load simulator, and instrumenting the program to do things like measure the packet traffic and response time (example: every packet was timestamped when it arrived, the timestamp was carried through the system, and upon the completion of the request, the time taken was computed. I then had to track the mean and standard deviation as well as max and min times to get a decent statistical analysis so I knew if I could trust my simulation).

Bottom line: you're asking the wrong question. The question is how to measure an actual set of instances of the program under simulated load conditions, and measuring the performance of the program itself. No amount of theorizing about what the performance might look like is going to answer this sort of question.

Note that you have to expect 70 seconds for a reverse DNS lookup; you need to do this asynchronously. I did a whole lot of work to guarantee that this ran without interfering

microsoft.public.vc.mfc: Re: True Memory Use

with other threads; I had several threads devoted just to doing the DNS lookup. And what's going to happen if you take near max time? A 70-second delay on a 2GHz machine is a whopping lot of delay, something over 11 orders of magnitude over the basic CPU clock cycle. A database update could be 5 or 6 orders of magnitude slower than a CPU clock. So worrying about code size isn't going to tell you about the impact of these. A page fault is six orders of magnitude delay over an L2 hit, so if you have paging traffic, you're going to get far worse performance. This means that the working set is going to be the killer (it usually is). If the working set of each program is so large that they cannot co-exist in the memory, given the very heavy load you've just described you are going to be dominated by third-order effects, which static analysis cannot reveal.

Database update is going to affect your performance, so measuring your program itself isn't going to tell you much. And the GUI performance is going to have an impact as well, so you have to drive that. Our monitor also had database updates and GUI updates running, so I kept the database updates running in a background thread with a queue so they did not slow down the GUI. But of course the GUI **does** consume cycles, so only load simulation is going to tell you anything meaningful.

Ultimately, you have I/O bandwidth (to disk, to network, to screen), compute bandwidth (how much work do you have to do), network latencies (your reverse DNS lookup), and as a component of the I/O bandwidth, paging bandwidth. You can't get any answers to such a complex interaction without real loads to tell you what is happening. Any one of these could dominate the others by several orders of magnitude and be the ultimate gating of your performance, but unless you have real data, you will have no idea. The DNS lookup is going to be hard to simulate unless you first can model the actual DNS lookup time, then factor that into your simulation (for example, by creating a DNS proxy server on a simulation machine which will do a Sleep() of a random interval based on the expected network delay times before returning the DNS record, to simulate those delays; otherwise a local simulation won't give you realistic data.

joe

On Sun, 13 Feb 2005 05:23:35 -0000, "Mark Randall" <markyr@gmail.com> wrote:

> *"Joseph M. Newcomer" wrote:*

>> *Surprising, particularly because I built a server monitoring tool that can*

>> *monitor up to*

>> *256 servers at a time, and the limitation was arbitrary because we had*

>> *allocated only an*

>> *8-bit field for the server ID.*

>>

>> *However, running 50 copies of most reasonable programs is not a burden.*

>> *The working set is*

>> *going to be the major performance limitation. The peak memory usage*

>> *probably doesn't*

>> *matter much, because it would include all the standard libraries which are*

>> *shared anyway.*

>> *joe*

>>

>

> *Monitoring the servers themselves is only part of the task – the servers are*

> *Active Worlds world servers, and each contains usually up to 100 users*

Re: True Memory Use

microsoft.public.vc.mfc: Re: True Memory Use

>(although they have been known to reach into the 1000s) of user clients, and
>every event caused by the client needs to be tracked in real time, added to
>that each world server has a database of 'objects' (3D objects that form the
>environment, Active Worlds is kinda like there, but AW is 10 years old,
>There is only 1 or 2) with potentially over 100 million records, all of
>which need to have their events monitored by my tool when sent by the world
>server.
>
>In addition to this I need to do things such as IP and reverse DNS lookups
>of every client connected, as well as update a database, coupled with a
>rather complex GUI I have included a link to a screenshot of in a simple
>configuration, and for each of these events / GUI features it needs to
>respond in some way to the world server, the tool is a Bot, just like IRC
>bots – but a lot more complex, it has to track clients in virtual 3D, space,
>time, and their full interaction with the 3D environment.
>
>http://zetech.swehli.com/software/evo/img/evo_11.png
>
>Detailed description of the proggy anyway. As for running 50 Im glad it
>shouldnt be a problem RAM wise, processor wise I hope my code is efficient
>enough to handle the data should it go to a huge scale.
>
>- Mark R
>

Joseph M. Newcomer [MVP]
email: newcomer@flounder.com
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm