

Re: What will many cores mean to future Windows releases?

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.non-technical/2007-06/msg00302.html>

- *From:* "m. Th." <a@xxxxxx>
 - *Date:* Sun, 03 Jun 2007 12:19:33 +0300
-

Bob Dawson wrote:

"Paul Nichols [TeamB]" wrote

Views are to perform a consolidations of several tables to allow for viewing on joins that are normal operational processes. I do not see how that relates to a VM at all.

Technically you're right of course—there's no relation. But conceptually I think there is an analogy: each is a way of presenting a programmer with a simplified representation of the underlying reality against which (in theory <g>) he can program more productively.

Or if you don't like the view analogy one might think about an OPF (object persistence framework) an another sort of 'VM'—let the programmer write myObj.Save and an entire underlying layer is invoked. Does such a system put programmers in a box and retard their learning? Quite possibly. But well done, it also allows newbies right out of school to create perfect ACID transactions involving hundreds of objects—something they'd otherwise struggle with for weeks or months.

Agree with this. And this, imho, must happen. The programmers must concentrate on their human problem, not struggle with the technology. They must steer on **what** to do not on **how** to do it. But I don't think now, that a programming platform based on a VM stack is the response now (at least how .Net is implemented). Not at this so high performance price. Why one should choose to have (in your example) between myObj.Save and x86 machine code so many layers in plus which a VM implies? Isn't better to have a compiled code straight away?

However, if we had a VM caching mechanism that would read and write the core processes to a permanent store, after the first run, then VMs would be much more efficient.

Basically such a system would have to be able to detect system configuration changes and

Re: What will many cores mean to future Windows releases?

invalidate the realized code caches as required, but I see no reason why that shouldn't be possible. Though I'm not sure how much savings would actually result. Many of the programs I'm concerned with run for hours or days at a time—eliminating a first-run JIT really wouldn't achieve that much in our use profile.

In fact, these aren't VM anymore but, as you stated, JIT compilers. Cross-compilers are also a related family here.

But the main problem with VMs/JIT/Cross-platform are the famous 'least common denominator' problem. As much as general you are the lesser specific you are. Someone (Sir Isaac Newton) said that there are ones who tend to know lesser and lesser about more and more till they reach to know nothing about everything and others that tend to know more and more about lesser and lesser things till they reach to know everything about nothing. And this 'nothing' is a point. Is their point of view.

current OS performs. Does anyone truly believe that Vista makes good use of system resources?

not touchin' that ... <g>, but I'll make the point below that that's not so different than was the case with the first compilers.

advances in programming technology, but in the past few years, I see way too many so called programmers that do not even know how to perform a stack dump, write a batch file, or know how to run a debug command. This isn't advancement, this is kiddie scripting. :)

Yes and no—the old guard might say the same about me in that I never made a living or got really comfortable with ASM. Any climb in abstraction is accompanied by some loss of competence in the layers below—and that's been the case since the original 'comp sci.' departments evolved from mathematics or electrical engineering. Few programmers nowadays have any training in IC design—and neither do I; but I don't feel the lack of an EE degree as my biggest professional void. I don't think that anything can be said against VMs that couldn't equally be urged against any language higher than op code. Certainly as much was urged against the notion of an OS itself: writing to an OS is never going to be as efficient or elegant as just writing to the hardware. But OS's won anyway.

This puts developers into a box. Get them out of the box and they are lost. I personally feel you are not free to innovate and improve operations, if you do not have control over how the operations work. Maybe I am just an old timer :).

So to climb up several levels, I'm not sure that the VM puts programmers into a box any more than the VCL does: we've both seen programmers that don't have a clue how the VCL is actually structured despite Borland shipping the source, much less being comfortable with the

Re: What will many cores mean to future Windows releases?

underlying windows API. Is that Delphi's fault?

...But the box of VCL+WinAPI is much more larger than the box of a VM. Is the problem of 'least common denominator' from above. I felt this very much when I worked in FoxBase, FoxPro and VB. If you're on what the closed box offers you then it's ok. If not... (I remember now what struggle I must have for a binary AND in FoxPro... although a very nice product – I mean version 2.0–2.6, of course ;–))

Another problem here is the 'innovate and improve operations' that Paul said. Because every user wants from the programmer to build a space shuttle (this is, of course, normal due the human nature of every one from us) you must be prepared to build a space shuttle even if the specifications speak about a paper plane. I remember now the war which I had with some MS Access fans on RAD. We all started from scratch. In the first week they were the clear winners. But too bad for them that the customers didn't stop with 'can you add this?' after the first week... oh, well...). If one wants to work in the more comfortable environment of a highly abstract space and to let the abstraction layer to do the thinking for him that's ok, but don't force every one to work there only. Remember, in the past one of the main criticisms of Pascal were that it was too high level. And one of the main reasons that Delphi succeeded it was that it was a two-way tool and exposed the entire (well, almost) API to developer.

I've generally heard it said that a programmer should be competent one layer lower than he actually works, and at least passingly familiar with the layer under that. But that's generally not the actual case, and I doubt it ever was. If programmers are, on the whole, more incompetent than normal nowadays, it's probably a reflection of two things: the rate at which what's possible keeps expanding (giving us ever more to know), and the fact that, as you note, the reality of VMs lags their promise by some ways. But then again, it was years before compilers became smart enough to outcode most ASM writers—I don't think it's entirely unexpected that VM runtimes are equally 'un-optimised.'

1. Yeah, the programmers today don't want to focus. They are scared. ("What if the technology <x> which I use will become obsolete?") And that's why they tend to learn less and less about more and more (see above) which gives us the superficiality of the world we live. Also, generally speaking we don't work for the men who respect (their users/customers) (ie. personalities), we work for money (ie. cold thing). So our entire work flow (from thinking to post-sell support) is rather flawed (especially on quality aspects).
2. Theoretically you're right comparing (platform dedicated) compilers with VM runtimes but in practice the cross-platform issues (storage, protocols, different APIs, GUI issues also), different types of sand-boxes, resource consumption at runtime puts this comparison in a very different light imho.

On the general issue of programmer competence, I thought this interesting:
http://www.ftponline.com/vsm/2007_05/magazine/departments/guestop/

Opening line: "The point at which anyone on the planet is competent to write .NET applications passed sometime in the last year or so."

Re: What will many cores mean to future Windows releases?

Very nice. Also, my 2c from some old-timers: (;-)

The gap between the best software engineering practice and the average practice is very wide perhaps wider than in any other engineering discipline. A tool that disseminates good practice would be important.

Fred Brooks

Although leading-edge software-development practice has advanced rapidly in recent years, common practice hasn't. Many programs are still buggy, late, and over budget, and many fail to satisfy the needs of their users. Researchers in both the software industry and academic settings have discovered effective practices that eliminate most of the programming problems that were prevalent in the nineties. Because these practices aren't often reported outside the pages of highly specialized technical journals, however, most programming organizations aren't yet using them in the nineties. Studies have found that it typically takes 5 to 15 years or more for a research development to make its way into commercial practice (Raghavan and Chand 1989, Rogers 1995, Parnas 1999).

–Steve McConnell – Code Complete (2nd Edition)

Also regarding to .NET VM architecture now I remember an old Chinese story:

<< The greatest doctor from the Chinese empire was once asked by the Emperor:

– Who is the greatest doctor in our empire?

And he responded:

– When the one's illness is in a very advanced degree and the man is close to death, then I use all my knowledges and with big effort and in a very big amount of time – months, even a year – I manage to save the man. My skills are known in the entire empire. But, you see, my Emperor, we are in our small village three brothers. The one which is bigger than me, sees the ill from its first symptoms and cures the sick man in a week, maybe two. His skills are known only in our village. And the biggest brother knows that a man is sick even before a perceptible sign is to be shown outside. He cures the man in a day or two. His skills are known only by me. >>

Good discussion, BTW.

Yes, thanks.

bobD

Also I'm enjoying participation. Sorry for my musings. (hth).

m. th.

.

Re: What will many cores mean to future Windows releases?