Subject: Re: Compiling IDL ... ever likey ?
Posted by Ken Knighton on Tue, 23 Jan 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Rick White <rlw@stsci.edu> wrote:
> Ken Knighton wrote:
>
>> Languages that compile to the host machine level are rapidly becoming
>> obsolete and investing programming time into turning a very efficient
>> pseudo-compiled, array oriented language like IDL into a true compiled
>> language is a waste of resources that could be better used to improve
>> other features of IDL. For example: I haven't heard anyone suggesting
>> that JAVA be turned into a compiled language.
>
> You're wrong -- there are definitely plans to produce Java compilers
> that turn Java binaries into native machine code.

No, I said that I haven't heard of it. Obviously, you are more in touch with the JAVA community than I am. But what you are saying simply reinforces the fact that languages of the old paradigm such as Fortran and C, are becoming obsolete. Actually, after thinking about it some more, I should have said:

"Traditional text-based, editor-entered, languages that are compiled directly to the machine level are becoming obsolete."

By the way, this doesn't mean that the billions of lines of COBOL, FORTRAN, C, C++, etc. are going anywhere soon. Heck, some of the computers we use are almost 20 years old. What I am saying is that new language systems that are successful will have to be multi-platform (JAVA, IDL, etc.), provide rapid application development tools (DELPHI, Visual BASIC, etc), or provide a short learning curve and ease of use to new users (IDL, MATLAB, Visual Basic, EXCEL). My inclusion of EXCEL shows that traditional approaches to using computers are losing ground to "End-User Programming".

> are seen by the Java community as crucial to getting good performance
> on computationally intensive applications.

Since you are a Java expert, does Java have built-in:

1. Vector based arithmetic and array manipulation functions
2. Plotting/imaging
3. Image processing and numeric functions
4. Mapping functions

These are mostly done at the machine level by IDL because they are atomic operations in IDL. Because of this, they wouldn't be improved much by

compilation. If they are not present in Java, then that explains why compiling Java is seen as crucial for computationally intensive applications. Perhaps what IDL need is more atomic level functions to perform common computationally intensive tasks.

>
> I think the Java approach could serve as a good model for an IDL
> compiler.

I am all for improving efficiency so long as it is not at the expense of removing portability and ease of use.

> Java gets compiled to an intermediate, machine independent
> pseudo-code which is (ordinarily) executed by the Java interpreter.
> The execution penalty of the interpreted code compared with C is
> about a factor of 20. The new compilers will replace the
> interpreter and will translate the Java pseudo-code into native machine
> instructions at execution time. This on-the-fly (or "just-in-time")
> compilation will produce code that executes at nearly the same speed
> as C (I haven't seen any hard numbers.)

And a different compiler will have to be written for every platform.
JAVA has a lot more people and resources behind it than does IDL.

>
> I think this approach could work very well for IDL.

I agree, but whether implementing it is in the best interest of RSI is another matter. They would have to do a cost-benefit analysis, and I think that this feature would not pay for itself.

> Perhaps someone should be working on an IDL-to-Java translator or even
> an IDL-to-Java pseudo-code compiler!

If you feel that it would make money, perhaps you should do it yourself.

Hey, just thought I'd liven things up a bit on our usually straight-laced and boring newsgroup. :)

Ken Knighton knighton@gav.gat.com knighton@cts.com
General Atomics
San Diego, CA (Sunny, 70 degrees, wore my shades into work)