CST-221 Operation Systems

Joe Leon

Virtualization

12-15-19

Github: <https://github.com/SkieXI/CST-221-Final.git>

1. Compiled programs are generally lower level and run faster since once they are converted into machine language, they are compiled all at once and can be configured to run on specific hardware, and in turn, get the most advantages out of what that hardware has to offer. This is due to the nature of how a compiled program works, once a piece of software called a complier is ran, for example GCC, the C file is then converted into machine language that is specific to the hardware that the code is being ran on. By comparison, interpited languages are conveted into machine language one at a time by their own compliers and are executed as they are needed. This results in a number of differences in performances, though neither is particaully strong or weaker than the other. And with advancing hardware and VM techniques, these differences have been shrinking even more in recent years

Interpreted languages, while slower, only tend to compile what is needed at the given time. This can be useful for programmers who have to maintain code and expand upon later updates. Interpreted languages also tend to be higher level code that is easier understood by humans. Another aspect to take into account is that interpreted code can also be changed during its execution, and this has a number of security concerns since new code can be injected into an application by skilled hackers who wish to exploit the functionality.

Another advantage to interpreted languages is that they can be ran on any major OS, so long as the language is installed in it. C and C++ programs do not run natively in Windows environments. And would require a separate complier to be installed as oppose to being able to run simply with a GCC command in Linux. However, for most applications that are household utensils or utilities, most of them are written in C since the moment you turn them on, all of their functionality. Another noteworthy detail is that interpreted languages take up less memory overall since they are compiled in parts. Compiled code on the other hand, requires more memory to function since the whole program is being converted to machine language at once. Also, Compiled code will not run at all until every line of code has been debugged or make sure that it is functional beforehand.

For having a program that is being ran on a VM machine, I would say that a compiled code would be much more beneficial since it will be able to take advantage of smaller storage sizes and the exactly CPU that the application is be ran on. However, in the end, a lot of the differences that each approach brings to the table have been diminished over the years, or the differences between the two are now more minuscule. This is made even more evident with how powerful modern hardware has gotten over the years. Though with VMs, this can become an issue if each machine is only given a few hundred MB of RAM to work with.

Work cited:

Haas, J. (2019, December 8). What Are the Benefits of Using Compiled Languages. Retrieved December 15, 2019, from <https://www.lifewire.com/compiled-language-2184210>

Khillar, S. (2019, June 10). Difference Between Compiled and Interpreted Language. Retrieved December 15, 2019, from <http://www.differencebetween.net/technology/difference-between-compiled-and-interpreted-language/>

What's the difference between a compiled and an interpreted language? (n.d.). Retrieved December 15, 2019, from <https://www.programmerinterview.com/general-miscellaneous/whats-the-difference-between-a-compiled-and-an-interpreted-language/>