**INQUIRY 3:** IS THERE A DIFFERENCE BETWEEN THE BUILD TIME OF OPTIMIZED AND CROSS PLATFORM IDES?

Sarah Huerta

Dr. Alice Winn / Logan Chalfant

19 November 2019

ISC3523C Fall 2019

**Abstract**

I investigated to see if there was a significant difference in the build time of a program on different Integrated Development Environments (IDE) when optimized for an operating system. I ran a program in four different IDE: XCode (Mac OS), Visual Studios (Windows), and CLion (cross platform software). Using a single factor ANOVA test, I found that there was a significant difference in the build times for the four different IDEs, *F (3,76) =663, p=2.15 x 10-54.* Using a post hoc comparison using a Tukey-Kramer test indicated that there was a significant difference when comparing Visual Studios (M=1016.5, SD= 120.8) to CLion on Windows (M=535.9, SD=31.3) which was significantly greater with a difference of 480.7 milliseconds, but not XCode (M=159.5, SD=51.4) to CLion on Mac OS (M=214.7, SD=21.2) which was significantly less with a difference of 55.15 milliseconds.

**Introduction**

In order to run outside programs, the user must download them onto the computer. One example is iTunes. iTunes, which is the Apple media software is available on Windows but is the default player for Mac OS. Since it is not originally meant for Windows, the behind the scenes software of the program had to be recreated for it to work. Users may notice a slower speed that it runs if it was compared to the speed that it runs in a Mac. iTunes will run faster on the Mac because the software was created for that specific type of computer. Not only are certain programs created for specific operating systems, but now certain programs are being created to be used with multiple operating systems. These are known as cross platform software.

In the world of programming, there are many different ways that a user can run a program that they write. One common way is the use of an Integrated Development Environments (IDE) for building and running programs. An IDE is a software used to compile, build, and run a high-level language program such as C++, Java, and Python. Research has showed that by using an IDE to run programs, there has been a significant increase in programmer productivity (Zayour & Hajjdiab, 2013). One issue that high-level programmers have with IDEs is the speed it takes to build the program. The build time, also referred to as the compile time of a program is the speed at which the computer converts the high level software language (C++, Java, and Python) into machine language (binary) for the computer to read, link any necessary files to the program, and get the program ready to execute for the user.

A highly debated topic in the world of programming is the use of IDEs. The topic is debated because it has also created issues with properly learning the programming language and debugging (Zayour & Hajjdiab, 2013). Debugging programs involves looking through code for any errors that the user might have made while writing. IDEs can do this for the user. At an educational level, this can be beneficial to beginner programs as it promotes learning opportunities within programming courses since students can focus less on the small mistakes and work on grasping larger concepts as a whole (Whitney, 2018).

At the college level, students who take the FSU Introduction to C++ are recommended to use an IDE as they are learning to first code. In order to allow the students to have access to an IDE they can use no matter the computer they own, the professors promote the use of CLion, a cross platform IDE that is available on Windows and Mac OS.

According to Dr. Liu, a professor in FSU Computer Science department, IDEs built for a certain operating system is going to have a better build time than a cross platform IDE (X. Liu, personal communication, November 4, 2019). Not just with IDEs but with applications in general, according to Dr. Liu, software is going to be optimized for working with that specific operating system if it was created for it. The software is written in a way that parallels the way the operating software is written and creates a faster and more efficient work space (Xu, 2018). This does not mean that cross platform software, programs created to work on any software, won’t work, it just means there is the chance that it will result in a slower build time of the program since it has to account for the different types of working environments that it may be used in.

In my programming experience, I have worked with Visual Studios, an IDE built for Windows OS, XCode, an IDE build for Mac OS, and CLion, a cross platform software. All three IDE software have been recommended to me by FSU faculty in my programming experience. Working with different IDEs on different operating systems made me wonder if there really was a difference in the build time of programs using IDEs built for operating systems, Visual Studios and XCode, and a cross platform software, CLion. Following the information given to me by Dr. Liu, I believe that there will be a significant difference when comparing the operating system IDEs to the cross-platform IDEs. I think that the third-party software will take more time on both of the operating systems when compared to its optimized counterpart.

**Experimental Design**

To address my question, I wrote a beginner level C++ program that I used to test each of the individual IDE’s as seen in Appendix B. To simplify the building process, I wrote my program in a way that does not use object-oriented programming. This allows the program to only work within the program, rather than having to call from outside functions. Eliminating this outside factorwill allow each of the four IDEs to build the function the same way, as different software may have different instructions for calling the outside files which could affect the build time.

To create a similar working environment for each IDE, I tested my program on the Mac OS and Window FSU computers at Paul Dirac Library. Using public computerswill eliminate any systematic bias that could occur by having excess files and programs on the disk space of the computer. Unlike personal computers, the library computers do not store files saved by users, meaning there are not any extraneous files that may be slowing down the run time of the computer.

I tested the program 20 times with each IDE. After running each program and getting the build time, I force quit the IDE and restarted the program to get a fresh working environment for each test run.

To collect more precise measurements for XCode, I had to use the “Time Profile Entire System” service which is an outside application included with the Mac OS. This could have created random bias in my estimations as the software itself was not measuring the program build time, but an outside application. If there was any delay in the programs interacting with each other this could have created a delay in measuring the build time which would cause a systematic error in my data collection.

**Analysis**

As seen in Figure 1, Visual Studios ran 480.7 milliseconds slower than CLion on Windows. XCode ran 55.2 milliseconds faster than CLion on Mac. I used a single factor ANOVA test to test the null hypothesis that there is no difference in build times among the I concluded that at an alpha value of .05, there is a significant difference among the build times of the IDEs, *F(3,76)=663, p=2.15 x 10-54,*

Since there was a significant difference among the build times, I furthered my analysis and performed a post hoc Tukey Kramer test to see which paring was statistically different from each other. Since there is only a significant difference with one of the pairs, it is hard to say if there truly a difference in whether cross platform IDE build times are different than optimized IDEs.

I also compared Visual Studios to XCode, to see if there could be a difference in the build time among the different operating systems. The test concluded that there was a difference between the build time of Visual Studios (M=1016.5, SD= 120.8) versus XCode (M=159.5, SD=51.4). This difference can also be seen in Figure 1, which shows the build times for all four IDEs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Paring** | **Null Hypothesis**  **Ho** | **Mean Difference** | **Minimum Significance Difference** | **Conclusion** |
| Visual Studio vs CLion on Windows | μ1=μ2 | 4.807 *x 102* | 5.673 *x 102* | Reject Ho |
| XCode vs CLion on Mac OS | μ3=μ4 | 5.515 *x 101* | 5.673 *x 102* | Fail to Reject Ho |
| Visual Studios vs XCode | μ1=μ3 | 8.57 *x 102* | 5.673 *x 102* | Reject Ho |

Table 1:Results of Tukey Kramer test showing a comparison of three mean build time pairings.

Figure 1: Bar graph representing the average build time of each IDE compared to its third-party software counterpart. The error bars show plus or minus one standard error of the mean.

**Conclusion**

With further research, I learned that the reason for a higher build time was seen with Visual Studios compared to CLion on Windows was due to the types of compliers that the programs use. CLion which is a newer software uses a complier that this written in a similar way to how a terminal builds a program. A computer terminal is a part of the operating system that allows the user to enter commands and directly translates it for the computer with no outside applications necessary. Visual Studios uses MSVC to compile its programs, which was created only to be run on a windows computer. CLion however uses Cygwin to build its programs. Since it has to be versatile on both Mac OS and Windows, Cygwin is modeled after the way a computer terminal works. The reason for XCode and CLion on Mac getting similar results is that XCode uses the Mac terminal to build the program, which is a similar process as Cygwin. But as Dr. Liu explained, XCode is going to have less commands it needs to run in order to build the program since it is optimized for its operating system. This way of building also accounts for why both Mac IDE’s were seen to be faster than the Windows (Melina Myers, personal communication, November 26th, 2019).

At this level of programming and the programming that my future students will be doing, the difference in speeds that I noticed would not be that relevant to our programing. However, these small differences could cost millions to an organization such as NASA. NASA uses programs tailored made for their operating systems that calculates space craft trajectory (Williams, Falck, & Beekman, 2018). These programs are being run in real time and need to have fast build times to keep up with the minute changes that are being made to the calculation. With each change in the variables that goes into calculating the trajectory, the program has to be rebuilt and needs to be quick. By only working with software that is optimized for the operating system they are using, NASA can focus more on optimizing the code they are writing rather than having to focus on program build time.

Despite having limitations of time, materials, and software availability, my results were consistent with the thinking that Dr. Liu explained to me. After conducting a single factor ANOVA test, I found that there is a significant difference between the build times. With further analysis using a Tukey Kramer test, my comparison of the paring Visual Studios compared to CLion were significantly different. However, comparing XCode to CLion there was not a significant difference. My results might suggest that there may be a difference in using cross platform IDEs rather than those created for the operating system. Looking at other cross platform IDEs, there could be a difference in the production of results that were seen.

**References**

Whitney, M. (2017). Embedding Secure Coding Instruction Into the IDE: Complementing Early and Intermediate CS Courses With ESIDE. Journal of Educational Computing Research, 56(3), 415–438.

Williams, J., Falck, R. D., & Beekman, I. B. (2018). Application of Modern Fortran to Spacecraft Trajectory Design and Optimization. 2018 Space Flight Mechanics Meeting.

Xu, S. *et al.* (2018).Taming the “Monster”: Overcoming Program Optimization Challenges on SW26010 Through Precise Performance Modeling. 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Parallel and Distributed Processing Symposium, 2018 IEEE International, IPDPS, 763.

Zayour, I., & Hajjdiab, H. (2013). How much Integrated Development Environments (IDEs) improve productivity? Journal of Software, (10), 2425.

**Appendix A**

Table A1

*20 run build time trials for the different IDEs in seconds.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Trial Number | CLion (Mac) | CLion (Windows) | XCode | Visual Studios |
| 1 | 193 | 515 | 277 | 1020 |
| 2 | 207 | 500 | 129 | 960 |
| 3 | 215 | 500 | 141 | 1020 |
| 4 | 207 | 485 | 135 | 990 |
| 5 | 218 | 546 | 132 | 1120 |
| 6 | 240 | 516 | 131 | 1010 |
| 7 | 224 | 516 | 276 | 1050 |
| 8 | 218 | 578 | 144 | 1000 |
| 9 | 200 | 516 | 124 | 970 |
| 10 | 203 | 516 | 154 | 1500 |
| 11 | 219 | 531 | 149 | 970 |
| 12 | 208 | 546 | 136 | 980 |
| 13 | 194 | 578 | 119 | 960 |
| 14 | 198 | 516 | 133 | 970 |
| 15 | 197 | 531 | 220 | 960 |
| 16 | 242 | 532 | 151 | 970 |
| 17 | 219 | 547 | 126 | 980 |
| 18 | 281 | 593 | 131 | 980 |
| 19 | 219 | 592 | 131 | 930 |
| 20 | 191 | 563 | 251 | 990 |

**Appendix B**

Table B1

*C++ program used to test IDE’s*

A screenshot of a social media post

Description automatically generated