JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Various techniques used for the optimization of Programming Code concerning Object-Oriented Programming

Krish Chaplot

B.E(Electronics & Communication) BITS Pilani K.K. BIRLA Goa Campus

Abstract: This paper aims to show various techniques by which programming code can be optimized. This is needed because an unoptimized code results in poor performance and heavy use of resources and also we can not obtain true compatibility of Code. Every code needs optimization and great efforts are spent during optimization, many languages are oops-based but some languages need more optimization and some need less as per the functionality available in the Language.

Keywords: Loop, Data Types , Switch , Binding , Variable, Function, Resources ,Constructor ,Header file , Program .

Introduction: There are different ways to code a given problem and all way leads to the desired solution. However, the code can be optimized for better performance like memory, processor, and internet bandwidth usage. We have to analyze that code to make it optimized so that we can efficiently manage resources. Most research papers on optimization include structure optimization .

Optimization Methods :

Optimization of Functions:

There are two main parts in the Function

a) Function Definition
b) Function Call

With the function, we should use the return value only if needed. We should prefer integer data type over char and short data type.

Different memory locations are used to store these definitions and calls. We need a way to correctly match each call to each function's definition because programs often require many functions to run smoothly.

Binding means joining a name to the operation. We have to optimize the binding since maximum time is utilized in the binding and if binding time is minimized then the overall efficiency of the program is increased. We have to reduce `less required dependencies and ensure that we minimize the nesting of function calls since nesting increases binding time and binding resources.

Optimization of Header Files

The optimum use of pre-compiled header files is also one of the criteria to improve the efficiency of binding. We have to hide the implementation details of the nesting function which is nested inside another function. We have to minimize unused code mostly that is for testing. Auto-generated codes must be minimized. Dead codes are executed but are not referenced during operation we have to identify such code and try to minimize it for the sake of increasing the speed of execution i.e. unnoticed code must be avoided. If possible cross compilation must be used it reduces memory requirements and lesser the memory higher the processing speed. we have to try to use the same code in different contexts.

Optimization using Alternate Programs :

We have to create alternate programs and we have to check the efficiency of the alternate program for similar sets of Input. We have to minimize those header files which are nested inside other header files and they are not usable. We have to minimize code duplication since duplicate code leads to problems in maintenance and code updating instead we have to develop reusable functions. Input-output operations must be minimized to avoid program performance degradation for this we can use the file-handling concept very efficiently.

Optimization using Multiple Processors.

We have to identify by finding those tasks which can be executed concurrently and then we have to assign concurrent tasks to the multi-core processors.

We have to plan functions in such a way that they should not modify variables that is they should produce similar outputs for similar sets of Inputs

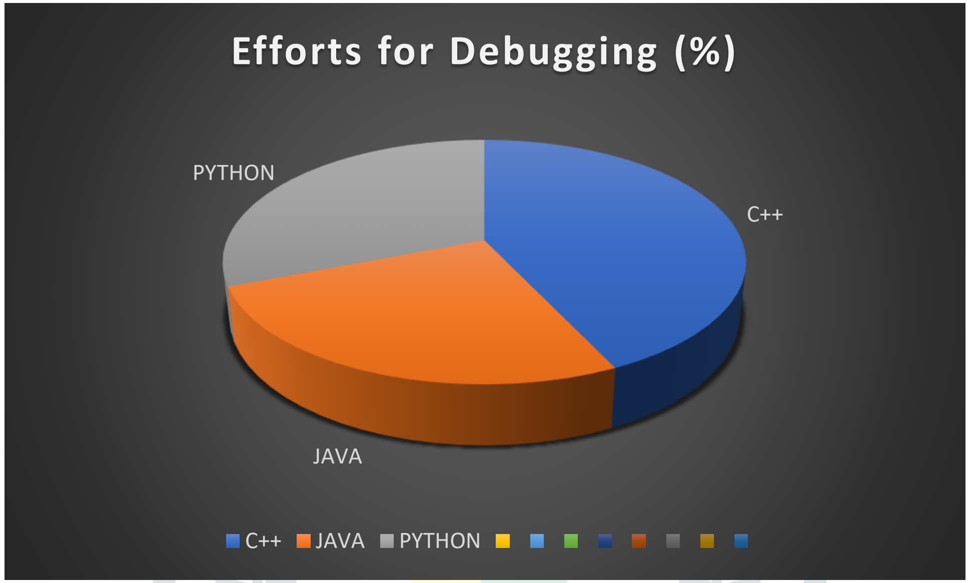
Optimization of Time spent in Debugging “

On average, a programmer creates 210 bugs per 3000 lines of code. 45 bugs per 3000 lines of code are noticed by the end users.

To remove a bug takes 60 times longer than writing a two lines of code More than sixty percent of the time of the developer is spent on code debugging.

In some countries, 100 billion dollars are spent on debugging annually So we have to optimize in such manner that minimize debugging .

	Efforts for Debugging(%)
C++	43
JAVA	26
PYTHON	31

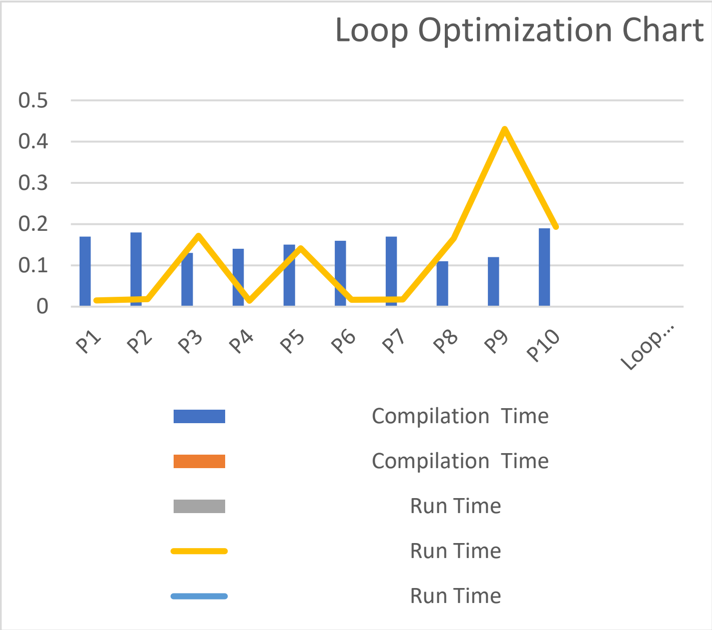


Optimization of Loops

We have to limit the number of loops and loops must be optimized. We should try to minimize the overhead associated with the loops and we can improve cache performance. Infinite loops must be used wherever needed, We have to reduce that code inside the loop which is not necessary. We should try to eliminate induction variables inside the loop. We have to replace the complex operator with the simple operator for example we can replace multiplication by successive addition. We should try to combine multiple loops if possible it will eliminate compile time. We can also divide single loop into multiple loops for efficiency We can also interchange the inner loop and outer loop to improve performance. Operations must be vectorized that is we should do operations on a vector rather than one value at a time.

	Compilation Time	Run Time
P1	0.17	0.0147
P2	0.18	0.0181
P3	0.13	0.1713
P4	0.14	0.0143
P5	0.15	0.1415

P6	0.16	0.0165
P7	0.17	0.0173
P8	0.11	0.1651
P9	0.12	0.4312
P10	0.19	0.1931



Optimization of Variables

We have to minimize local variables so that stack overhead can be minimized. We should declare local variables in most inner scopes.

If we have large data sets then they should be divided into chunks so that processing can be done in lower CPU components like memory, and speed.

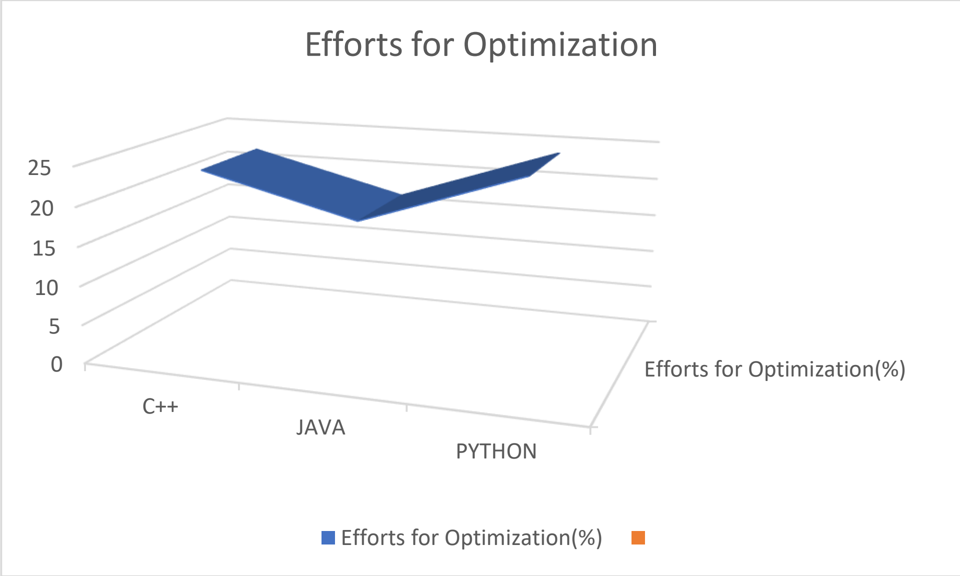
The slowest part of the program must be identified so that maximum efforts can be made there to improve performance.

Optimization of Constructor

We have to optimize the constructor since it will be called whenever an object is created so we have to keep the constructor weight light to improve compiler performance. We should try to convert small functions into inline it will improve performance to a very high degree , We should avoid function call cascading. Pre-increment operators must be preferred over post-increment operators.

In order to reduce the number of comparisons being performed we have to break larger and complex switch statements into multiple nested switch statements. Put frequently occurring case labels into one switch and keep the rest of the case labels into another switch which is the default leg of the first switch.

	Efforts for Optimization(%)
C++	23
JAVA	18
PYTHON	25



Result: The optimization of various structures and concepts was studied and it was found that both structures and concepts play equally important roles in the optimization of Code

Conclusion and Future Scope: Whenever we start coding any problem, we should not start immediately doing coding instead we have to perform code planning first i.e. we have to identify various structures that can be used for coding and then we have to select that structure that will provide optimum performance. Optimization is not uniform i.e. it may be applied differently in different conditions. Optimization is a spiral continuous process and has to be applied on every token of coding.

References :

[1]Alfred Aho, Ravi Sethi, Jeffrey D Ullman, “CompilersPrinciples, Techniques and Tools”, Pearson EducationAsia

[2] O.G.Kakde, (2008), “Compiler Design”, UniversitiesPress

[3] Carole Dulong, Rajiv Gupta, Robert Kennedy, JensKnoop, Jim Pierce (editors), (2000) “Code Optimization– Trends, Challenges, and Perspectives” Dagstuhl-Seminar-Report; 286, 17.9.–22.9.2000 (00381)

[4] Caspar Gries, (2009), “New Trends in the Optimizationof C-Code”,

[5] Kenneth Hoste Lieven Eeckhout,ELIS Department,Ghent University,” COLE: Compiler Optimization LevelExploration”

[6] Urban Boquist, “Code optimization Techniques for LazyFunctional Languages”, Thesis for the Degree of Doctor,Goteborg University,

[7] Stan Yi- Huang Liao, “Code Generation andOptimization for Embedded Digital Signal Processors”,Massachusetts Institute of Technology

[8] Neil Edward Johnson, “Code Size Optimization forEmbedded Processors”, Robinson College, Thesis for theDoctor of Philosophy at the University of Cambridge