Comparativo de desempenho na execução entre Linguagens de Programação

Diego Batista da Fonseca, Wellington B. Rodrigues, João Roberto U. da Cruz, Alann Kelly Pirchiner Perini

Tecnologia em Análise e Desenvolvimento de Sistemas - Centro Universitário Anhanguera de São Paulo - Campo Limpo

Estrada do Campo Limpo, 3677 - Jd. Bom Refúgio - Campo Limpo - São Paulo -

Cep. 05777-001- Brasil

dgbfonseca@gmail.com, wellington.rodrigues@anhanguera.com, joaor.cruz@anhanguera.com, alann.perini@anhanguera.com

Abstract. *Computer architecture has come a long way over the last few decades. Consequently, programming languages have evolved alongside with computers. This research aims at assessing and analyzing the runtime and use of computational resources, besides investigating whether paradigm features of programming languages have been kept in current architectures, considering the evolutionary leap of the last years. Such quantitative analysis will be based on the results obtained from the implementation of the array sorting algorithms Bubble Sort and Quicksort in the structured and object-oriented paradigms, through the use of C and Java programming languages.*

Resumo. *A arquitetura dos computadores evoluiu bastante no decorrer das décadas e, consequentemente, as linguagens de programação de que eles fazem uso também. O objetivo desta pesquisa é avaliar e analisar o tempo de execução e o consumo de recursos computacionais, além de investigar se as características dos diferentes paradigmas de linguagens de programação se mantêm em arquiteturas atuais, dado o grande salto evolutivo dos mesmos nos últimos anos. Esta proposta de análise quantitativa será baseada em resultados obtidos com a implementação dos algoritmos de ordenação de vetores* Bubble Sort *e* QuickSort *nos paradigmas estruturado e orientado a objetos, utilizando as linguagens de programação C e Java.*

1. Introdução

Um paradigma de programação, está intrinsicamente ligado ao modelo de pensamento de um programador, e a forma com a qual o programador analisou, abstraiu, e solucionou determinado problema (Ascencio & Campos, 2014). É através do paradigma de programação, que serão estipuladas as técnicas de programação utilizadas, com as quais os problemas do mundo real serão apresentados, modelados, e solucionados dentro da perspectiva computacional.

Os paradigmas de programação são classificados pela estrutura em que se baseiam, e a respeito dos diversos paradigmas de programação existentes, podemos citar: O paradigma estruturado (Ascencio & Campos, 2014) e o paradigma orientado a objetos (Deitel & Deitel, 2010).

Este trabalho está organizado em seções. Na Seção 2, apresentaremos o conceito da linguagem estruturada. Na Seção 2.1 a linguagem C. O conceito da linguagem orientada a objetos, na Seção 3. Na Seção 3.1apresentaremos a linguagem Java. A Seção 4, introduz os algoritmos de ordenação de vetores. Nas Seções 4.1 e 4.2 os algoritmos de ordenação de vetores *Bubble Sort* e *Quick Sort* respectivamente. Na Seção 5, explanaremos sobre a execução dos experimentos numéricos. Por fim, na Seção 6 as considerações finais e trabalhos futuros.

2. Linguagem Estruturada

A linguagem estruturada, como o próprio nome sugere, tem seu alicerce fundamentado no paradigma estruturado. A metodologia do paradigma estruturado, expressa que qualquer problema pode ser subdividido em problemas menores, que consequentemente, reduzem a complexidade do problema original tornando-o de mais fácil compreensão e resolução. Essas subdivisões na linguagem estruturada, são denominadas de sub-rotinas, ou funções (Ascencio & Campos, 2014).

O funcionamento do paradigma estruturado, dá-se da seguinte maneira: As sub-rotinas ou funções recebem valores variáveis. Esses valores são submetidos a um ou mais processos. Esse bloco de instruções formam uma estrutura, estabelecendo um fluxo de sequências lógicas, na qual cada passo dado gera valores de saída com o intuito de solucionar um determinado problema.

2.1. Linguagem C

Nos laboratórios Bell, Dennis Ritchie criou o C, uma linguagem oriunda das linguagens BCPL e B. Em 1972, a linguagem C foi implementada, tornando-se conhecida mundialmente como a linguagem de desenvolvimento dos sistemas operacionais. No momento, praticamente todos os grandes sistemas operacionais estão escritos em C e/ou C++ (Deitel & Deitel 2011).

A criação e o funcionamento de um programa em C pode ser dividido em cinco passos: o primeiro a Edição (O código fonte é criado e armazenado em disco); o segundo o Pré-Processamento (O código fonte passa por um pré-processador, no qual são executadas algumas tarefas como: Expansão de macros, junção de linhas separada por sequência de escape, remoção de comentários); o terceiro passo é a Compilação (O compilador pega o resultado do pré-processamento, e trata cada arquivo fonte como sendo uma unidade de compilação gerando um arquivo objeto); o quarto passo o *Linking* (O *Linking* é responsável por agrupar todos os arquivos objetos e gerar um arquivo executável); e o último a Execução (após o *Linking* o arquivo executável é criado estando pronto para utilização).

Em 1989, a linguagem C foi padronizada pelo *American National Standards Committee on Computers and Information Processing*. O documento que contém as normas de padronização do C é o ANSI/ISO 9899:1990 (Deitel & Deitel, 2011).

3. Linguagem Orientada a Objetos

A linguagem orientada a objetos, como o próprio nome sugere, tem seu alicerce fundamentado no paradigma orientado a objetos ,que visa o mundo de forma diferenciada da linguagem estruturada. Nesta a resolução dos problemas enfatiza o que deverá ser feito,

ou seja, as ações que precisam ser executadas, e como o problema será subdividido, a fim de reduzir sua complexidade (Ascencio & Campos, 2014).

Na linguagem orientada a objetos, a ênfase é voltada para a identificação dos objetos que compõe determinado problema, e como estes se relacionam entre si. Um objeto é uma estrutura de dados que contém atributos, que são as características do objeto e métodos, que são os comportamentos do objeto (Ascencio & Campos, 2014).

3.1 A Linguagem Java

Em 1991, a empresa *Sun Microsystems* (Deitel & Deitel, 2010) deu início a um projeto que tinha como objetivo criar interatividade entre diversos objetos utilizando C++, uma linguagem baseada em C que segundo o fabricante possuía suporte a orientação a objetos, mas observou-se durante o desenvolvimento do projeto que a linguagem C++ não possuía recursos suficientes, dando origem a criação de uma nova linguagem, o Java.

A linguagem Java é multiplataforma (Oracle, 2017) devido a Java *Virtual Machine* (JVM) um aplicativo de *software* que simula um computador, mas oculta o sistema operacional e os *hardwares* adjacentes dos programas que interagem com ela. Sendo assim um programa em Java pode ser executado em qualquer plataforma de *hardware* que contenha uma JVM.

A criação e o funcionamento de um programa em Java pode ser dividido em cinco passos sendo o primeiro a Edição (O código fonte é criado e armazenado em disco cujo nome termina com a extensão ".java"); o segundo a Compilação (O compilador Java é utilizado para compilar o arquivo ".java", gerando um novo arquivo que recebe a extensão ".class" que contém os *bytecodes* do programa); o terceiro o Carregamento (O carregador da Java *Virtual Machine* (JVM) lê o arquivo ".class" e carrega os *bytecodes* na memória principal); o quarto a Verificação (O verificador da JVM verifica se os *bytecodes* são válidos e se não violam as restrições de segurança do Java); e o último a Execução (A JVM lê os *bytecodes* e os compila para a linguagem de máquina na qual o computador entende, e executa as tarefas requisitadas).

4. Algoritmos de Ordenação de Vetores

Na perspectiva computacional, uma classe de algoritmos que são comumente utilizados são os algoritmos de ordenação. Um algoritmo de ordenação tem por finalidade organizar determinada lista, seja ela numérica ou alfabética, de acordo com regras pré-estabelecidas, que dependem das necessidades que foram apresentadas.

4.1. *Bubble Sort*

O *Bubble Sort*, é um algoritmo de método simples de ordenação por troca (Friend, 1956).

Um algoritmo de método simples, tende a usar uma quantidade maior de comparações em sua estrutura, possui código enxuto e complexidade de implementação reduzida, quando comparado com um algoritmo de método sofisticado. O *Bubble Sort* apresenta fraco desempenho em dados aleatórios (Astrachan, 2003).

Ao utilizar o *Bubble Sort*, todos os dados armazenados em um determinado vetor serão comparados entre si. Partindo do princípio de que, cada elemento ocupa uma determinada posição dentro de um vetor, a comparação ocorre quando esse determinado elemento é comparado ao elemento adjacente. Então, a regra de comparação pré-definida é analisada

e a permuta de posição entre os elementos comparados ocorre, caso os elementos comparados estejam fora de ordem.

Todo esse processo, acontece utilizando duas estruturas de repetição incorporadas uma a outra. Serão feitas tantas passagens dentro dessa estrutura, quanto forem necessárias para suprir o tamanho do vetor. O algoritmo só terá fim quando não houverem mais elementos a serem permutados (Ascencio & Araújo, 2011).

Podemos dividir o funcionamento do *Bubble Sort* em três etapas.

1ª Etapa: O vetor é percorrido comparando cada elemento ao seu elemento adjacente (em pares).

2ª Etapa: Quando a regra de comparação é saciada, a permuta dos elementos ocorre.

3ª Etapa: Por fim, executa-se o primeiro e o segundo passo até que não existam elementos a serem permutados.

4.2. *Quick Sort*

O *Quick Sort*, é um algoritmo de método sofisticado de ordenação criado por (Hoare, 1961) que preconiza a ordenação fundamentada na técnica de divisão e conquista, na qual um determinado vetor é dividido em dois, por meio de um procedimento recursivo.

Podemos dividir o funcionamento do *Quick Sort* em três etapas:

1ª Etapa: Dividir: Um vetor é dividido em dois subvetores não vazios. É escolhido um elemento no meio do vetor que recebe o nome de pivô. Os elementos são ordenados de maneira que os que ficarem à esquerda do pivô são menores ou iguais ao pivô, e os elementos à direita do pivô são maiores ou iguais ao pivô.

2ª Etapa: Conquistar: Os dois subvetores são ordenados recursivamente.

3ª Etapa: Combinar: Durante o processo recursivo, os elementos são ordenados no próprio vetor, não havendo nenhum processamento nesta etapa.

5. Experimentos Numéricos

A arquitetura de *hardware* que contemplará os experimentos numéricos, contará com um a seguinte configuração: Processador Intel® Core™ i5 3210M de 2.50 GHz, 6GB de memória DDR3 SDRAM 1600 MHz, armazenamento 2.5" SATA com espaço de 500GB com 5400 RPM e sistema operacional Linux Ubuntu (16.04.2).

Serão implementados os algoritmos de ordenação, *Bubble Sort* e *Quick Sort*. A proposição destes dois algoritmos dá-se pelo fato de apresentarem complexidade distintas, na qual o tempo de execução do *Bubble Sort* no pior caso é $O(n^2)$, e o tempo de execução do *Quick Sort* no pior caso é $O(n.log\ n)$. Pretendemos concluir que as duas linguagens de programação apresentam desempenho inferior, ou superior em cenários distintos de complexidade, ou que uma das linguagens apresenta desempenho superior a outra independente da complexidade do algoritmo. Essa implementação, ocorrerá em duas linguagens de programação; uma orientada a objetos e uma estruturada, sendo as linguagens escolhidas Java e C respectivamente.

Utilizaremos como base de dados para os experimentos numéricos uma implementação própria desenvolvida em linguagem C, que possua como suporte a função *rand* para geração de vetores com números aleatórios para possibilitar os testes de ordenação.

Faremos ensaios nos quais o primeiro vetor será da grandeza numérica de dez elevado a terceira potência, e os demais seguirão um padrão de grandeza no qual o exponente adjacente será aumentado em três.

Com intuito de analisar a distribuição dos tempos e eliminar possíveis *outliers*, cada ensaio será executado cinco vezes, utilizando a mesma massa de dados em cada etapa do experimento. O experimento chegará ao fim, no momento em que chegarmos a um valor no qual a ordenação desse vetor ultrapasse cinco horas para ser concluído utilizando o algoritmo *Quick Sort* em C.

6. Considerações Finais e Trabalhos Futuros

A arquitetura dos computadores evoluiu bastante no decorrer das décadas (Tanenbaum, 2010) e, consequentemente, as linguagens de programação de que eles fazem uso também (Deitel & Deitel, 2010). Essa afirmação nos leva ao seguinte questionamento: ainda há diferenças significativas entre implementações feitas com diferentes paradigmas de linguagens de programação?

Os próximos passos deste estudo buscarão responder se essas diferenças podem ser relacionadas as questões semânticas, do processo de escrita do código fonte, ou mesmo no desempenho do processamento de soluções. Posteriormente iremos implementar os mesmos testes em plataformas de *hardware* mais potentes para averiguar a diferença de desempenho de processamento em arquiteturas atuais.

7. Referências

Ascencio, Ana, F. G. & Araújo, Graziela, S. (2010). Estruturas de Dados: Algoritmos, analise de complexidade e implementações em Java e C/CC++". 1.ed. São Paulo: Editora Pearson. ISBN 978-85-7605-881-6

Ascencio, Ana, F. G. & Campos, Edilene, A. V. (2014). Fundamentos da Programação de Computadores: Algoritmos, Pascal e C++. 3.ed. São Paulo: Editora Pearson

Deitel, Paul & Deitel, Harvey. (2010), Java Como Programar. 8. ed. São Paulo: Editora Pearson, ISBN 9788576055631

Deitel, Paul & Deitel, Harvey. (2011), Como Programar em C. 6.ed. São Paulo: Editora Pearson, ISBN 9788576059349

Friend, E. (1956), Sorting on Electronic Computer Systems, Journal of the ACM, vol. 3 Issue 3, July 1956, Page 134-135. DOI 10.1145/320831.320833

Hoare, C.A.R. (1961), Algorithm 64: Quicksort. Magazine Communications of the ACM, vol. 4 Issue 7, July 1961, Page 321. DOI 10.1145/366622.366644

Oracle, (2017), Java Platform, Standard Edition (Java SE), Disponível em: <https:// www.oracle.com/br/java/technologies/java-se.html>. Acesso em: 04 jul. 2017

Tanenbaum, Andrew, S. (2010). Sistemas Operacionais Modernos, 3. ed. São Paulo: Editora Pearson, ISBN 9788576055631, ISBN: 8576052377 e ISBN13: 9788576052371