ANÁLISE E COMPARAÇÃO DE TEMPO DE ARMAZENAMENTO DAS ESTRUTURAS DE DADOS STRING, STRINGBUFFER E STRINGBUILDER

Ellen Tuane Silva Pinto, Amanda Evangelista Lima, Wagner dos Santos Clementino de Jesus.

Universidade do Vale do Paraíba / Faculdade de Engenharias, Arquitetura e Urbanismo, Avenida Shishima Hifumi, 2911, Urbanova - 12244-000 - São José dos Campos-SP, Brasil, ellentuanesp@gmail.com, amandaelima1@gmail.com, wagner@univap.br.

Resumo - A performance de dispositivos eletrônicos é medida pelo desempenho do aparelho em relação a aplicação utilizada, tornando-se assim um fator de importância na estruturação das aplicações. Concatenação de strings via “+” realizada pela classe String, pode ser conveniente para programadores, porém essa prática pode ser custosa em termos de memória e performance. Classes de estruturas de dados como StringBuffer e StringBuilder em Java, realizam funções similares ao que se refere a concatenação de strings, mas em suas diferenças tornam-se úteis para situações divergentes. Utilizando as estruturas de dados citadas, apresentamos uma análise e comparação de tempo de armazenamento das variadas formas de concatenação de strings. Refletindo uma aplicação real, o experimento demonstra semelhanças e divergências entre as variáveis utilizadas devido às suas maneiras de utilização da memória.

Palavras-chave: String. StringBuffer. StringBuilder. Java.
Área do Conhecimento: Engenharias/Engenharia da Computação

Introdução

Atualmente com a crescente demanda de tecnologia observa-se que dispositivos eletrônicos, sendo estes aparelhos celulares, computadores, relógios inteligentes e diversos outro, esse fenômeno proporciona um processo em que a tecnologia seja cada vez mais desfrutável, ocorra a utilização de diversas aplicações de modo simultâneo, podendo causar assim lentidão dos aparelhos e consequentemente menor desempenho de seu sistema operacional. Com isso, deve-se estimar como os dados que circundam a memória desses mecanismos podem interferir no desempenho destes aparelhos (Patterson, 2014).

A performance pode ser medida com a quantidade de aplicações acessadas simultâneas e o processamento da unidade de processamento central, CPU, compreendida no aparelho. Por intermédio desta informação pode-se afirmar que o processamento da CPU possui um valor padrão de acordo com o aparelho, a solução para obtenção de um melhor desempenho deve ocorrer na criação das aplicações. Em aplicações criadas e executadas na linguagem de programação de alto nível, como Java, há várias estruturas de armazenamento podendo estas serem específicas de numerais, palavras, letras, caracteres, listas, arrays e diversos outros. Três destas classes que são muito utilizadas são a String, StringBuffer e StringBuilder (Uzayr, 2022).

Quanto às similaridades e diferenças entre as classes estudadas neste trabalho, a classe String é representada por intermédio de uma sequência de caracteres imutáveis, possuindo um tamanho fixo, mas que possibilita a concatenação de strings recorrendo a criação de novos objetos quando realiza uma concatenação. As classes StringBuffer e StringBuider são similares a classe String em relação as suas funcionalidades, mas se diferem no modo em que armazenam seus caracteres ou substrings, eles podem ser inseridos ou concatenados durante ou ao final de sua execução e desta maneira apresenta crescimento dinâmico em seu comprimento, neste caso então, não necessitam de uma quantia elevada de memória para seu armazenamento. Contudo, a principal diferença entre as duas classes é devido ao fato de que os métodos da classe StringBuffer serem sincronizados (Niemeyer; Leuck, 2013).

Mediante as definições anteriores e diferenças apresentadas entre as classes, este trabalho propõe realizar a verificação de qual das classes é mais eficiente em seu armazenamento. Surgindo-se assim a ideia de criar uma codificação que realize a verificação do tempo de concatenação das estruturas de dados String, StringBuffer e StringBuilder, deste modo ao inserir uma quantia elevada de entradas temos o tempo de execução como métrica, tornando possível a realização de comparações entre as estruturas e assim analisar suas respectivas eficiências comparando-as ao realizarem o armazenamento.

Metodologia

Ao que se refere aos equipamentos e ferramentas dispostos para a realização do experimento, foi utilizado um computador portátil, com processador Intel Core i7, memória RAM de 8 gigabytes, sistema operacional Linux Ubuntu 20.04.5 LTS, enquanto o desenvolvimento do código para a realização da comparação entre as classes foi desenvolvido em linguagem Java versão 17.0.3+7 com auxílio da IDE de desenvolvimento Eclipse versão 2019-12(Liming, 2018).

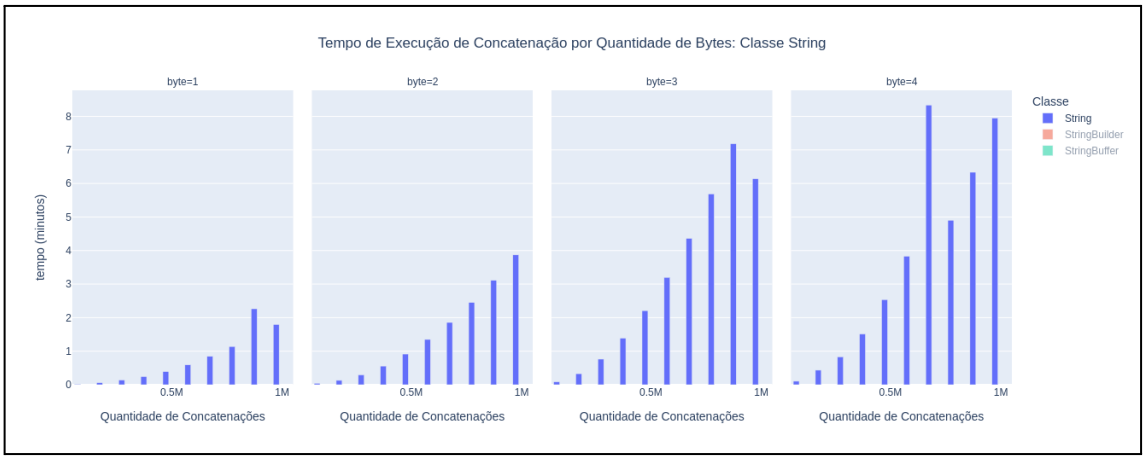
O experimento consistiu na criação de uma codificação que possibilita medir o tempo de concatenação para cada uma das classes isoladamente, isto é, a coleta do tempo inicial e tempo final de cada classe foi realizada no início e no fim de cada execução, desconsiderando qualquer outro código que venha antes ou depois, como atribuições e escritas no console, assim foram avaliados puramente a metodologia de concatenação de cada classe. Partindo disso, foi medido separadamente o tempo em que cada classe leva para concatenar 1 byte 100.000 vezes, o que completa um ciclo de coleta de dados para aquela quantidade de bytes. Mantendo o mesmo número de bytes, a quantidade de concatenações foi aumentada progressivamente de 100.000 em 100.000 até atingir o valor de 1.000.000, integrando assim 10 ciclos de coleta de dados. Ao final dos 10 ciclos, as variáveis strings atingiram um tamanho da ordem de megabyte. Em sequência, o mesmo foi feito para os valores de 2, 3 e 4 bytes. As informações como classe, quantidade de concatenações e quantidade de bytes foram armazenados em arquivo CSV (do inglês comma-separated values) juntamente com os respectivos dados de tempo obtidos ao final de cada ciclo de 1M. Com a finalidade de visualizar e analisar os resultados, foi utilizada a linguagem de programação Python e as bibliotecas Pandas e Plotly.

Importante destacar que o experimento foi realizado de modo que somente a aplicação do código estava sendo executado pela máquina sem nenhum outro processo acompanhando, somente os processos padrões do sistema operacional.

Resultados

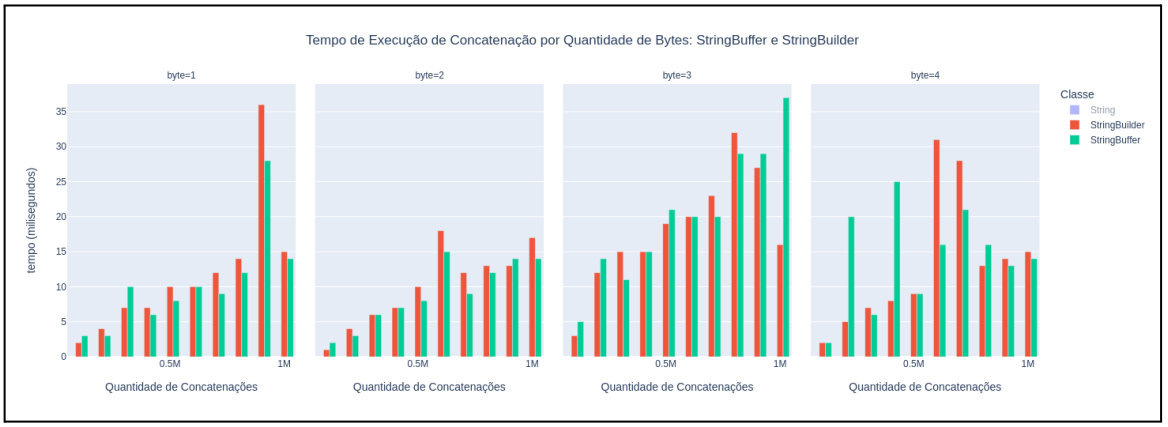
Os resultados apresentados na figura 1 e 2, tem como propósito expor as métricas que possibilitem a visualização do tempo de execução de ocorrência das concatenações de strings. Portanto, as figuras logo abaixo, mostram a disposição do tempo de execução para as concentrações de 0,1M a 1M vezes, para strings de 1, 2, 3 e 4 bytes.

Figura 1 - Gráfico Tempo de Execução de Concatenação por Quantidade de Bytes: Classe String.



Fonte: o autor.

Figura 2 - Gráfico Tempo de Execução de Concatenação por Quantidade de Bytes: Classes StringBuffer e StringBuilder.



Fonte: o autor.

O primeiro comportamento observado e esperado em comum para as três classes é que o tempo, apesar haver alguns valores atípicos, seguem uma crescente à medida que há um aumento no número de concatenações e de tamanho em bytes. O segundo, é que a partir da observação das figuras 1 e 2 é possível notar que há uma diferença considerável em relação ao tempo de concatenação da classe String em relação às outras duas classes. Uma evidência disso é que o tempo final está na escala de minutos, visto que se mantido em milissegundos os valores finais de tempo dificultariam a leitura e análise.

Para compreensão do comportamento do experimento como um todo, na tabela 1 são dispostas medidas estatísticas que representam cada classe. Importante ressaltar que os dados da classe String são apresentados na escala de minutos, enquanto os dados das outras duas classes em milissegundos. Partindo disso, é possível então apontar a principal diferença entre os resultados, a classe String tem mediana de 1.37 minutos e as classes StringBuffer e StringBuilder apresentam

mediana de 12.5 milissegundos. Embora a média e mediana das classes StringBuffer e StringBuilder apresentam resultados próximos, há uma diferença de 0.29 no desvio padrão entre ambos, favorecendo a classe StringBuffer.

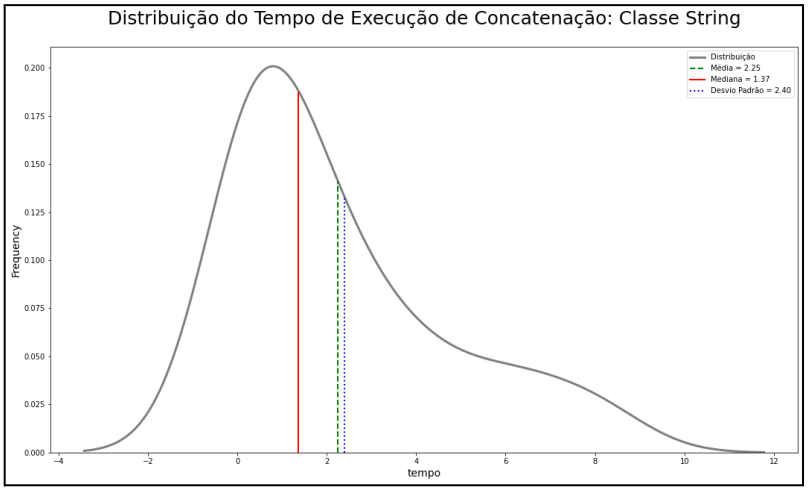
Tabela 1- Resultados Estatísticos das Classes.

Classe	Média	Desvio Padrão	Mediana	Variância
String (s)	2.50	±2.40	1.37	5.75
StringBuffer (ms)	13.40	±8.27	12.50	68.40
StringBuilder (ms)	13.30	±8.56	12.50	73.34

Fonte: o autor.

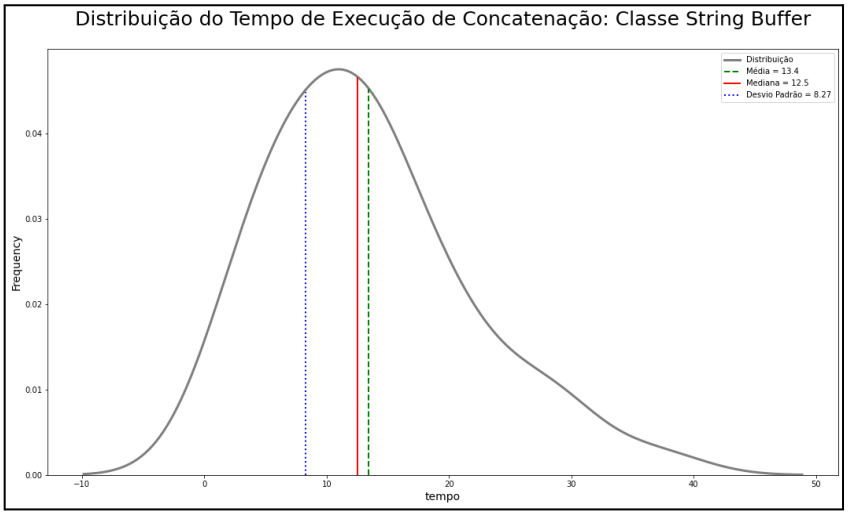
As figuras 3, 4 e 5, ilustram a distribuição dos dados que englobam o experimento como um todo e também os resultados estatísticos descritos na tabela 1 das classes String, StringBuffer e StringBuilder, nessa ordem. Entre os valores de média e mediana, entende-se que a mediana descreve melhor o comportamento dos dados, isto devido ao fato de que a média é sensível a dados extremos, tal como a diferença considerável de tempo de execução entre 0.1M e 1M, ou a valores atípicos, como por exemplo o resultado da concatenação de 4 bytes realizada pela classe String 0.7M vezes, observável na figura 2, que nos mostra um valor fora da tendência, sendo assim esses dados influênciam no valor final da média, que se encontra disposta deslocada para a direita na sequência de figuras. Uma medida descritiva importante a ser observada é a linha de desvio padrão em azul na figura de representação da Classe String, ela indica que há uma grande variabilidade nos dados considerando todos os experimentos, visto que ela está à direita do centro da distribuição. Isto representa na prática a variação do tempo de concatenação da classe, que vai de poucos milissegundos a 8 minutos, dados esses representados também na figura 2.

Figura 3 - Gráfico Distribuição do Tempo de Execução de Concatenação: Classe String.



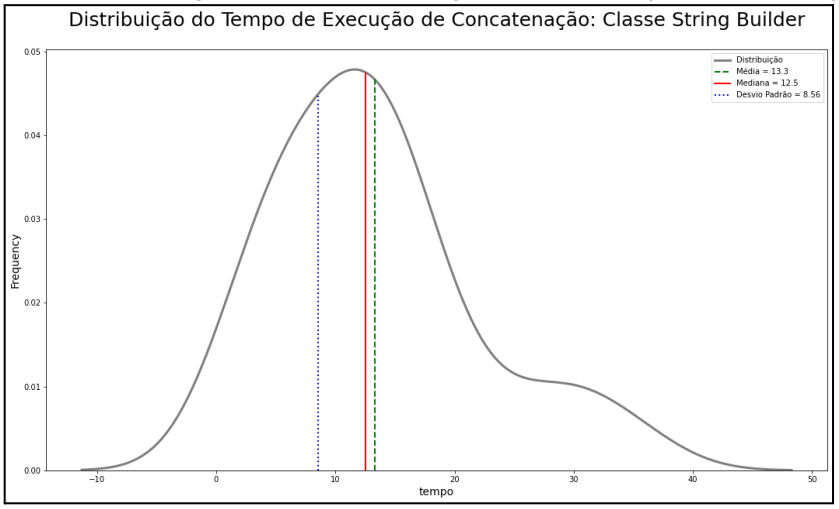
Fonte: o autor.

Figura 4 - Gráfico Distribuição do Tempo de Execução de Concatenação: Classe StringBuffer.



Fonte: o autor.

Figura 5 - Gráfico Distribuição do Tempo de Execução de Concatenação: Classe StringBuilder.



Fonte: o autor.

Discussão

Levando os resultados dispostos em consideração, para entendê-los tecnicamente comparamos uma amostra. O experimento que realizou 0.6M concatenações de 4 bytes na classe String leva aproximadamente 8 minutos, enquanto para as classes StringBuilder e StringBuffer o tempo para a mesma configuração é de 30 e 15 milissegundos respectivamente. A justificativa para este resultado é que a classe String possui um comportamento estático e imutável, ou seja, o valor não pode ser mudado após a primeira atribuição. Desta maneira, ao concatenar strings com diversas novas áreas de alocação de memória são criadas, o que claramente eleva o tempo para execução da tarefa, e as

strings antigas perdem referência, mas continuam ocupando o espaço na memória em sua reminiscência. Portanto, a concatenação dessa forma é considerada prejudicial à performance da aplicação.

Ao realizar a comparação dos resultados entre as classes StringBuffer e StringBuilder, podemos observar que na execução da mesma função, o Stringbuilder apresenta uma ligeira rapidez sendo evidenciada na média de 13.3 milissegundos contra 13.4 milissegundos da Stringbuffer, mas apresenta medianas idênticas, o que impossibilita a comparação entre ambas neste caso. Portanto, como citado anteriormente na seção dos resultados, sendo média sensível a valores extremos, considerando o desvio padrão, a StringBuffer apresenta uma vantagem, 68.40 contra 73.34 da StringBuilder. Estatisticamente, presumindo os dados do experimento como um todo, essa diferença de 0.29 indica que a classe StringBuffer variou menos em tempo para a concatenação das strings. Tecnicamente explicando essa pequena variação, a principal diferença entre as duas classes é que o StringBuffer opera seus métodos de modo sincronizado em relação a classe StringBuilder, em específico o método "append'" sempre adiciona esses caracteres no final do buffer, em um momento especificado, possibilitando assim maior eficiência na aplicação quando há diversas leituras ou modificação na mesma string. Portanto, a partir dessas evidências, pode-se dizer que a classe StringBuffer apresenta melhor desempenho quanto a concatenação de string.

Conclusão

O estudo contempla uma breve abordagem de comparação de consumo de tempo de concatenação utilizando as classes String, StringBuffer e StringBuilder em Java, visando a performance da aplicação. O experimento mostra a importância de otimizar a concentração de strings utilizando as classes em Java corretamente e com eficácia. Análises estatísticas foram combinadas junto com a disposição dos dados do experimento para providenciar informações importantes para a discussão do experimento e assim provar que os resultados experimentais mostram que a utilização da classe correta apresenta uma estratégia de otimização. O tempo de execução dos experimentos possibilitou também entender a importância da redução na alocação de memória dado à classe utilizada. Portanto, quanto à performance e otimização dos recursos de memória, os resultados encorajam a utilização da classe StringBuffer no uso direto de concatenação de strings.

Referências

Liming, Sean. Malin, John. **Java and Eclipse for Computer Science**, 2018.

Niemeyer, Patrick; Leuck, Daniel. **Learning Java.** 4 ed. 2013.

Patterson, D.A. **Organização e Projeto de Computadores: A Interface Hardware e Software.** Elsevier, 2014.

Uzayr, bin Sufyan. **Mastering Java.** 1 ed. 2022.