UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS DE RIO PARANAÍBA
SISTEMAS DE INFORMAÇÃO

RODRIGO DE OLIVEIRA COSTA
2746

TRABALHO PRÁTICO 1

RIO PARANAÍBA
2019

RODRIGO DE OLIVEIRA COSTA
2746

TRABALHO PRÁTICO 1

Trabalho Prático 1 apresentado à Universidade Federal de Viçosa como parte das exigências para a aprovação na disciplina SIN142 – Programação Concorrente e Distribuída

Orientador: João Batista Ribeiro

RIO PARANAÍBA

2019

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Nesse trabalho vai ser apresentado algoritmos e a teoria demostrar a diferença de desempenho dos problemas CPU bound, I/O bound, utilizando os métodos de programação sequencial e concorrente.

1.1 Programas, Processos e *Threads*

Um programa e um conjunto fixo e permanente de sequencias de instruções que pode ser executas pelo usuário ou diretamente por outro programa dentro do sistema operacional.Sendo assim um processo e um conjunto dinâmico de instruções criadas a partir de um programa no seu tempo de execução, um processo segue apenas um fluxo de execução que vai sendo incrementado a partir que suas instruções forem concluídas.Uma *Thread* pode ser considerada uma linha de execução de um processo. Essas *threads* pode ser criar uma ou varias linhas de execução para um processo, que faz o processo obter uma maior aproveitamento de uso de *CPU*.

1.2 Sequencial vs Concorrente

Um processo pode ser executado de duas formas **sequencial** ou **concorrente** sendo assim dando diferença no despenho e no aproveitamento dos recursos de hardware do equipamento a ser utilizado. No processo sequencial cada processo só começa quando o outro termina sendo assim deixando muito recurso parado como entrada e saída de dados quando e apenas necessário processamento para executar a função pedida por esse processo. No processo concorrente o sistema pode trabalhar com varias processos ao mesmo tempo sendo assim tendo um maior aproveitamento dos recursos e terminando cada tarefe mais rapida.

1.3 *CPU bound, I/O bound e Memory bound*

Nos casos dos seguintes algoritmos acontecem problemas de pendendo dos tamanhos das instancias a serem processadas para cada uma dessas unidades, criando assim o mais conhecido gargalo que nesse caso seria uma falta de recurso para um determinada função enquanto outras unidades estejam trabalhando com uma capacidade menor. Nesse caso são sugeridas varias formas de programação diferente que possam sanar ou diminuir esses caso no processo que estar sendo executado

***CPU bound*:** Processo em que se depende mas do uso do CPU podendo afetar as outras unidades do processador como por exemplo: as unidade de entrada e saída e unidades de memoria.

***I/O bound*:** Processo em que se depende mas do uso das unidades de entrada e saída podendo afetar as outras unidades do processamento como por exemplo: as unidade de CPU e unidades de memoria.

***Memory bound*:** Processo em que se depende mas do uso da memoria podendo afetar as outras unidades do processamento como por exemplo: as unidade de CPU e as unidade de entrada e saída.

1.4 Programação concorrente em Java

No java o sistema não consegue manipular as theads físicas mas sim ele faz uma simulação de *Thread* na maquina virtual do java usando a Classe *Thread* que contem todos métodos e funções necessárias para que seja usada na execução das *Treds*.

No Java existem dois modos de se fazer uma implementação usando a programação concorrente uma e fazer a classe da implementação implementar a classe *Runnable* e a outra e criar uma classe para organizar as *Thread* estendendo a classe *Thread* em todos os casos e necessário implementar a o método *Run* . depois que a classe *Run* estamimplentada basta criar as treds e usar os métodos *Star* ou *Join* dentre outros para execução necessária.

2 Desenvolvimento

Nesse trabalho foi feita a implementação de dois algoritmos um representando o problema de CPU bound e outro I/O bound.

Na implementação de CPU bound foi feito um algoritmo de ordenação baseado no algoritmo do *Select Sort* nesse algoritmo foi implementado de duas maneira essas sendo:

Sequencial: No algoritmo sequencial o usuário pode fazer vários teste com entradas de tamanhos diferentes após escolher os tamanhos o algoritmo gera dois vetores com valores aleatórios com o tamanho selecionado, e ordena sequencialmente um vetor depois o outro usando o algoritmo select sorte, e apresenta o tempo para o usuário.

Concorrente: Como no anterior o algoritmo o usuário pode fazer vários teste com entradas de tamanhos diferentes após escolher os tamanhos o algoritmo gera dois vetores com valores aleatórios com o tamanho selecionado,e ordena de forma concorrente e simultâneo aproveitando os recursos usando o algoritmo select sort.e apresenta o tempo para o usuário.

Na implementação de I/O bound foi feito um algoritmo de criação de arquivos nesse algoritmo foi implementado de duas maneira essas sendo:

Sequencial: No algoritmo sequencial primeiro criar uma pasta no C: do computador e depois verifica se tem algo dentro se tiver apaga todos arquivos.depois cria 10 mil arquivos isso na pasta temp e sequencialmente mas 10 mil aquivos na pasta temp1 e logo depois apresenta o tempo gasta para criação ao usuário.

Concorrente: No algoritmo concorrente semelhante ao anterior primeiro criar uma pasta no C: do computador e depois verifica se tem algo dentro se tiver apaga todos arquivos.depois cria concorrentemente 10 mil arquivos na pasta temp2 e temp 3. logo depois apresenta o tempo gasta para criação ao usuário.

2.1 Configuração dos experimentos

2.1.1 *Hardware e software* utilizados

Para o desenvolvimento e teste do algoritmo foi utilizado as seguintes configurações de Hardware e Software:

Hardware: Notebook ASUS,Processador I7 7500U, 8GB Memoria RAM, SSD 240GB, Placa de Video Geforce 930mx.

Software: NetBeans IDE 8.2,Sistema Operacional Windows 10 versão 19.03.

2.1.2 Manual de utilização dos programas

Para execução dos códigos vai ser necessário a instalação na maquina do NetBeans IDE 8.2 e que netbeans tenha acesso a pasta c: para criar as pagas para os teste da implementação do I/O bound.

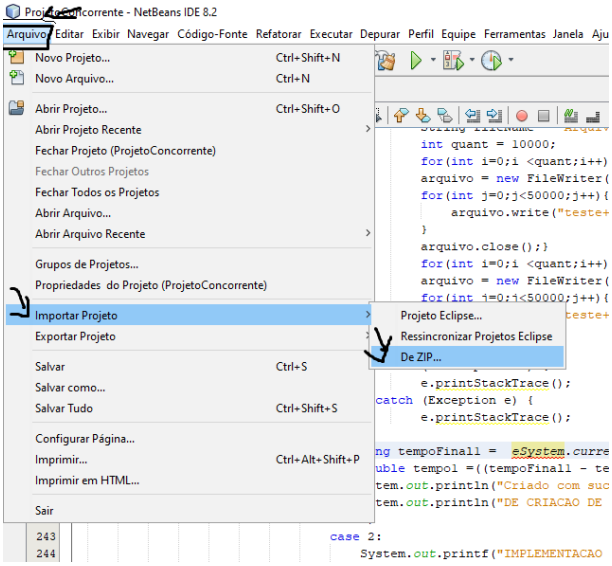

Figura 1 – Exemplo de como importa o projeto no NetBeans

Fonte: Próprio Autor

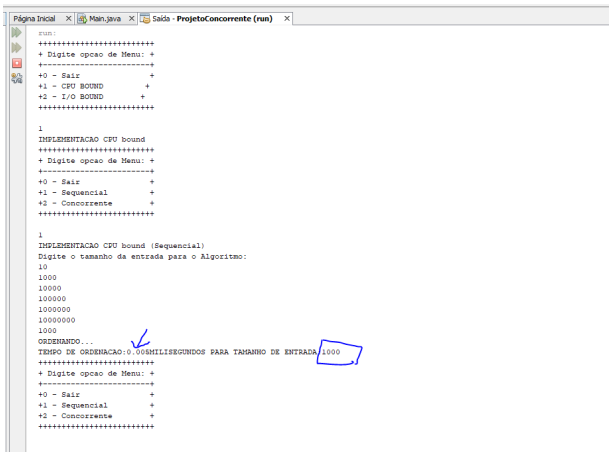

Figura 2 – Exemplo de como execuçãp o algoritmo no NetBeans

Passos para abrir o projeto: Abra o NetBeans, Va em Arquivo, Clica em Importa Projeto, depois Va em De ZIP, Seleciona o Arquivo, Clica em Importa.

Passos para executar o projeto: No NetBeans com o projeto aberto va em executar projeto.E siga as opções de Menu para cada algoritmo vai ter uma opção de menu com a implementação sequencial e concorrente.ao final de cada implementação tera um resultado de tempo de execução em medida de segundos.

2.2 Resultados e análises

2.2.1 Algoritmo *CPU Bound*

Para esse algoritmo foram feitos 5 teste de tamanho semelhantes para os 2 algoritmos que podemos ver no gráfico representado pela figura 3 e na Tabela 1 e analisar os resultados do algoritmo CPU Bound,veremos que no algoritmo concorrente obteve uma melhora de desempenho quase homogenia de 2,5 segundos para uma entrada de $1x10^5$.

Então pode-se conclui-se que para entrada de $1x10^5$ com duas Threads o algoritmo concorrente tem desempenho melhor que o algoritmo sequencial.

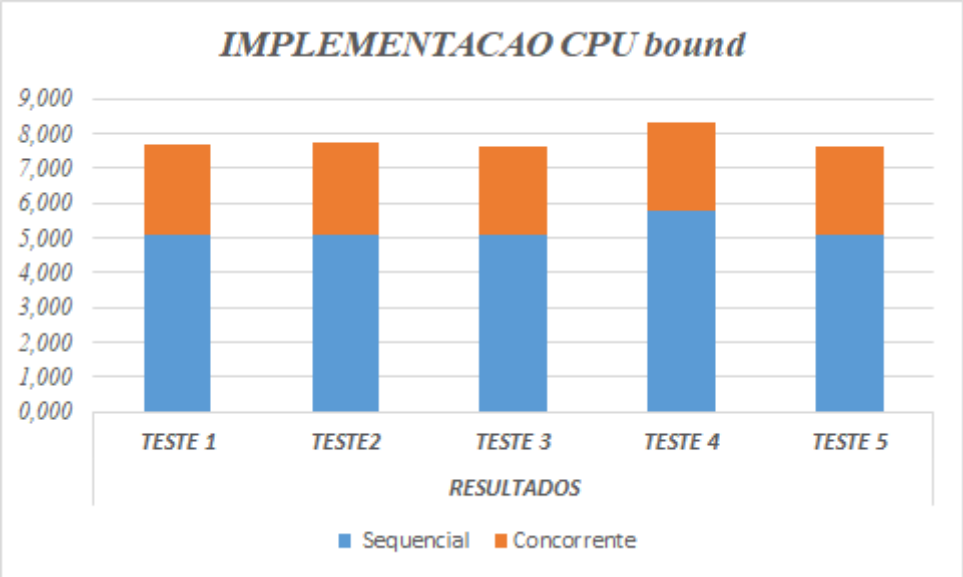

Figura 3 – Representação dos resultados apresentados pela implementação CPU *bound*

Fonte: Próprio Autor

IMPLEMENTAÇÃO CPU BOUND								
	RESULTADOS						*ANALISES*	
	TESTE 1	*TESTE2*	*TESTE 3*	*TESTE 4*	*TESTE 5*	*MEDIA*	*DESVIO PADRÃO*	*VARIÂNCIA*
SEQUENCIAL	5,108	5,076	5,092	5,796	5,077	5,230	0,317	0,1004
CONCORRENTE	2,546	2,640	2,543	2,531	2,541	2,560	0,045	0,0020

Tabela 1 – Tabela Resultados implementação CPU *bound*

2.2.2 Algoritmo *I O Bound*

Para esse algoritmo foram feitos 5 teste de tamanho semelhantes para os 2 algoritmos que podemos ver no gráfico representado pela figura 4 e na Tabela 2 e analisar os resultados do algoritmo *I O Bound*,veremos que no algoritmo concorrente obteve uma melhora de desempenho em media quase 100 segundos para o algoritmo sequencial.

Então pode-se conclui-se que para a instancia de $1x10^4$ com duas Threads o algoritmo concorrente tem desempenho melhor que o algoritmo sequencial.

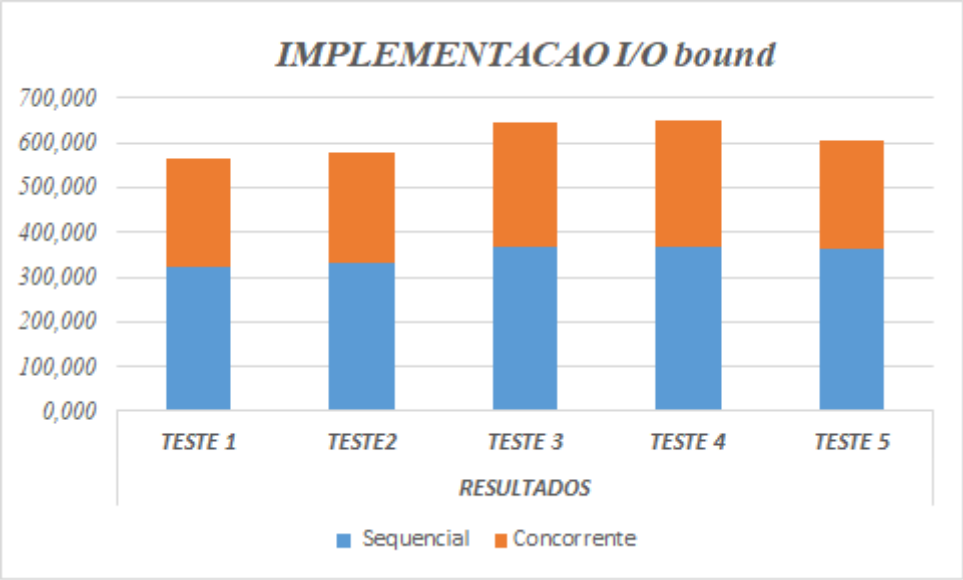

Figura 4 – Representação gráfica dos resultados apresentados pela implementação I/O *bound*

Fonte: Próprio Autor

IMPLEMENTAÇÃO I/O BOUND								
	RESULTADOS					***ANALISES***		
	TESTE 1	***TESTE 2***	***TESTE 3***	***TESTE 4***	***TESTE 5***	***MEDIA***	***DESVIO PADRÃO***	***VARIÂNCIA***
SEQUENCIAL	323,058	329,371	367,584	366,640	361,300	349,591	21,589	466,0914
CONCORRENTE	242,914	248,668	278,755	284,219	245,091	259,929	19,880	395,2272

Tabela 2 – Tabela Resultados implementação I/O *bound*

3 Conclusão

Concluíamos com esse trabalho que o uso de Thread para os algoritmos apresentados houve uma melhora significante no desempenho e no uso de hardware dos equipamentos utilizados.

Referências

BEN, M. **Principles of Concurrent and Distributed Programming, Second Edition**. Second. [S.l.]: Addison-Wesley, 2006. ISBN 9780321312839.

CONCURRENCY GLOSSARY. **Concurrent (order-independent) vs sequential**. Disponível em: <https://slikts.github.io/concurrency-glossary/?id=concurrent-order-independent-vs-sequential#:~:targetText=Concurrent\%20(order\%2Dindependent)\%20vs\%20sequential,step\%20to\%20produce\%20correct\%20results.> Acesso em: 17 nov. 2019.

PRODDIGITAL. **Thread e Processos – Quais as diferenças**. Disponível em: <https://proddigital.com.br/tecnologia/thread-e-processos-quais-as-diferencas/>. Acesso em: 16 nov. 2019.

WIKIPEDIA. **Thread (computação)**. Disponível em: <https://pt.wikipedia.org/wiki/Thread_(computacao)>. Acesso em: 16 nov. 2019.

(BEN, 2006) (PRODDIGITAL,) (WIKIPEDIA,) (CONCURRENCY GLOSSARY,)