Leonardo Santos Paulucio

Relatório do 1º Trabalho de Processamento Paralelo e Distribuído

Vitória - ES

22 de Maio de 2018

Leonardo Santos Paulucio

Relatório do 1º Trabalho de Processamento Paralelo e Distribuído

Trabalho apresentado à disciplina de Processamento Paralelo e Distribuído do curso Engenharia da Computação da Universidade Federal do Espírito Santo como requisito parcial de avaliação.

Professor: João Paulo A. Almeida

Universidade Federal do Espírito Santo

Engenharia da Computação

Vitória - ES

22 de Maio de 2018

Sumário

1 Introdução

Nas últimas décadas houve uma crescente demanda na área de processamento de dados, o que exigiu o desenvolvimento de máquinas mais poderosas. Porém, apesar da grande capacidade existentes nos novos computadores quando comparados aos primeiros a serem produzidos e até mesmo a computadores de uma década atrás principalmente ao se analisar capacidade de processamento e memória, eles ainda não são capazes de atender essas demandas de processamentos sozinhos. Por esse motivo foram desenvolvidas várias técnicas de concorrência o que contribuiu para a criação dos Sistemas Distribuídos.

Esse trabalho tem por objetivo praticar programação paralela usando o *middleware* JavaRMI e realizar a análise de desempenho em um cluster de computadores. Ele consistirá na implementação de uma arquitetura mestre/escravo para realizar um ataque de dicionário em uma mensagem criptografada.

2 Implementação

2.1 Estrutura de Dados

2.2 Problemas Durante a Implementação

Durante o desenvolvimento do trabalho alguns problemas surgiram fazendo com que fosse necessário realizar algumas mudanças na estrutura de dados e na lógica de programação.

2.2.1 Cliente e Escravo

O cliente foi o primeiro a ser implementado. Devido a sua simplicidade não foram encontrados problemas durante sua implementação.

Já na implementação do escravo, que também não era muito complicada, um problema ocorreu, apesar de ter sido de fácil solução.

O problema que surgiu foi durante a implementação do método *startSubAttack*. Nos primeiros testes notou-se que os escravos não realizavam outros sub-ataques em paralelo, assim, ao analisar o código percebeu-se que não estavam sendo criadas *threads* no método para realizar o ataque, dessa forma o escravo ficava bloqueado durante a execução de um ataque não atendendo outras requisições que chegavam. Ao se criar *threads* esse problema foi solucionado.

2.2.2 Mestre

Os principais problemas que surgiram ocorreram durante a implementação do mestre, que é o responsável por gerenciar todos os escravos e todos os ataques que estão ocorrendo no momento.

3 Interoperabilidade

A interoperabilidade do trabalho foi testada com os seguintes grupos:

- Grupo do David e
- Grupo do André e Eric.
- Grupo do Eduardo e Gustavo. VER SE TIRA DEPOIS

3.1 Problemas Encontrados

falar da WIFI

4 Análise de Desempenho

4.1 Máquinas e Equipamentos Utilizados

Nesse trabalho foi utilizado um computador com processador *Intel Core i5-2410M CPU 2.30GHz x 4* com 6GB de memória RAM e o Sistema Operacional utilizado foi o *Linux Mint KDE 64-bit*.

Pegar informações dos PCs do LabGrad.

4.2 Uma Única Máquina

4.2.1 Centralizado

4.2.2 Paralelo com Vários Escravos

4.3 Várias Máquinas

5 Conclusão

Ao final desse trabalho é possível notar que um sistema distribuído é uma ferramenta muito poderosa pois permite aproveitar recursos de diferentes equipamentos para executar uma tarefa em comum.

Com a distribuição do serviço entre as máquinas é possível obter uma melhora significativa no tempo de resposta speedup. Observa-se ainda que não é possível atingir o speed up ideal segundo aLei de Amihdal devido ao overhead existente na rede, processamento, etc.

Outro ponto que é importante e que foi possível notar durante a implementação desse trabalho é a complexidade de funcionamento que existe na máquina responsável por realizar todo o gerenciamento, nesse caso representada pelo mestre. Ele tem que lidar: com o controle de máquinas ativas, com a distribuição das tarefas, com o controle do status das tarefas, com o controle de concorrência, redistribuição de tarefas, entre outros. Além de todas essas tarefas ele ainda tem que cuidar do acesso concorrente à variáveis de controle, que é um grande problema ao se trabalhar com muitas *threads*, assim, percebe-se a necessidade que sempre vai existir para que algum pedaço do programa não seja paralelizável.

falar das analises

Referências