Aplicação ParallelPassBreak: Modelagem e Implementação

Ibero C. K. Benítez, Renata H. S. Reiser, Adenauer C. Yamin

Centro Politécnico – Universidade Católica de Pelotas (UCPel)
Caixa Postal 402 – CEP 91.501-970 – Pelotas – RS – Brazil

{ibero,reiser,adenauer}@ucpel.tche.br

Abstract. *This work considers the modeling and implementation of the ParallelPassBreak algorithm, a parallel aplication characterized as requiring a high-rate tasks processing during its execution.*

Resumo. *Este trabalho considera a modelagem e implementação do algoritmo ParallelPassBreak, uma aplicação paralela caracterizada por requerer alta taxa de processamento para a sua execução.*

1 Introdução

Técnicas de programação paralela e/ou distribuída estão sendo cada vez mais estudadas, devido à existência de aplicações que demandam grande poder de processamento para serem executadas [Leopold, 2000]. A proposta deste trabalho é o desenvolvimento de uma aplicação paralela e distribuída, caracterizada por exigir uma elevada taxa de processamento para a sua execução. O trabalho apresenta a modelagem e implementação de uma aplicação paralela e/ou distribuída com grande carga de processamento. Considera-se a aplicação *ParallelPassBreak*, para quebra de senhas por força bruta.

Na seção 2, descreve-se as principais etapas metodológicas da aplicação *ParallelPassBreak*, incluindo em suas subseções a arquitetura, o funcionamento, a implementação e os resultados obtidos com sua execução. A seguir, tem-se as considerações sobre a continuidade do trabalho.

2 Aplicação *ParallelPassBreak*

A aplicação ParallelPassBreak utiliza o método exaustivo de busca em espaço finito (alfabeto) para a quebra de senhas. Neste contexto, as senhas são geradas com o uso do algoritmo de *hashing MD5 (Message-Digest Algorithm 5)*, codificado com 128 bits. Este algoritmo se caracteriza por ser unidirecional, ou seja, após a aplicação da função de *hashing* MD5 em um valor de entrada, será impossível deduzir o valor inicial com base na saída da função. Essa característica permite que o algoritmo seja usado em autenticações (*login*), ou ainda, para verificar a integridade de arquivos em programas P2P, por exemplo. O MD5 possui ainda uma variação no padrão dos valores de saída (independente do valor de entrada) caracterizando-se como um algoritmo instável, isto é, pequenas alterações nos valores de entrada acarretam em resultados bem diferentes na saída.

2.1 Funcionamento da Aplicação ParallelPassBreak

A aplicação *ParallelPassBreak* consiste em quebrar 15 senhas de 6 caracteres. Para alcançar tal objetivo, as tarefas de modelagem e implementação são distribuídas da seguinte maneira:

(i) um cliente requisita uma tarefa ao servidor;

(ii) o servidor envia a esse cliente uma combinação de 3 caracteres (os 3 primeiros da senha) e os MD5 das senhas ainda não quebradas;

(iii) se a senha é quebrada, esta é enviada para o servidor com seu respectivo MD5;

(iv) ou, se após testar todas as combinações iniciadas com estes três caracteres, a senha não for quebrada, cliente os devolve ao servidor;

(v) o servidor retira do conjunto de tarefas pendentes cada uma das combinações testadas quebrando ou não uma senha;

(vi) verifica se todas as senhas foram quebradas e encerra a execução.

Salienta-se que o servidor sempre executa o item (v) independente se o cliente executou (iii) ou (iv). Caso uma tarefa seja concluída sem quebra da senha, essa é enviada ao servidor que a retira da fila de tarefas pendentes.

2.2 Arquitetura da Aplicação ParallelPassBreak

A aplicação foi implementada de acordo com a arquitetura cliente-servidor, mostrada na Figura 1. Neste caso, tem-se um servidor que pode receber pedidos de tarefas de diversos clientes para a execução das mesmas. A implementação foi desenvolvida na linguagem Java, a qual facilitou a execução do algoritmo em diferentes plataformas (*hardware*) utilizando uma configuração de grade heterogênea, composta por computadores multicores e máquinas sequenciais.

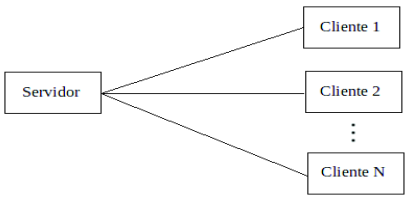

Figura 1. Arquitetura Cliente-Servidor

2.3 Implementação da Aplicação *ParallelPassBreak*

Nesta subseção estão descritos, através de uma linguagem estruturada, o principal módulo do programa, que implementa a arquitetura do cliente e servidor para a aplicação *ParallelPassBreak*. Veja Figura 2.

2.4 Resultados Alcançados

A aplicação foi testada em uma grade com configuração heterogênea, composta por 214 máquinas (clientes) para o processamento das tarefas. O resultado é razoável tendo em vista o número de processos, a heterogeneidade das unidades computacionais usadas e o bom funcionamento da aplicação. Em relação ao desempenho, pensa-se em organizar as tarefas em grupo ao enviá-las aos clientes, para assim, tentar diminuir o custo de comunicação entre cliente e servidor. As máquinas usadas possuíam duas configurações de hardware: uma configuração com um processador Intel(R) Celeron(R)

CPU 2.80GHz e 1 GiB de RAM, e outra com Intel(R) Pentium(R) Dual Core CPU. 1.60GHz e 1GiB de RAM. O tempo total de processamento para a quebra de todas as senhas de 6 caracteres foi de 6 horas e 33 minutos.

```
// Implementação da Classe Servidor

for(int i=0; i<85; i++){
     for(int j=0; j<85; j++){
          salvaEstado(i, j, tarefasEmExecução);
          for(int k=0; k<85; k++){
               aguardaConexão();
               msg = leMensagem();
             if(éUmPedidoDeTarefa(msg) || éUmaTarefaConcluída(msg))
              {
     tarefa = “”+alfabeto[i]+alfabeto[j]+alfabeto[k]+” “+md5DasSenhas;
                    criaThredParaEnviar(tarefa);
                    tarefasEmExecução.add(tarefa);
                    if(éUmaTarefaConcluída(msg))
                         tarefasEmExecução.remove(msg);
               }
               if(éUmaSenhaQuebrada(msg))
                    arquivo.write(msg);
          }
     }
//Implementação Clientes
pedeTarefa();
stringInicial = recebeTarefa();
for(int i=0; i<85; i++){
     for(int j=0; j<85; j++){
          for(int k=0; k<85; k++){
               s =stringInicial + alfabeto[i] + alfabeto[j] + alfabeto[k]+"";
               sMD5 = md5(s);
               if(compara(sMD5))
                    enviaProServer(sMD5+" = "+s+'\n'); //senha //quebrada

          }
     }
}
enviaProServer(stringInicial); //aviso de tarefa concluída
```

Figura 2. Implementação das Funções Cliente e Servidor

3. Continuidade do Trabalho

A aplicação *ParallelPassBreak* mostra-se uma boa proposta para a validação do ambiente de execução paralela. Neste sentido, a continuidade do trabalho considera a análise do algoritmo de forma que possa contribuir como uma aplicação para a validação dos componentes em desenvolvimento no Projeto D-GM (*Distributed Geometric Machine*). Este Projeto tem como princípio integrar duas linhas de pesquisa da UCPel: Grupo de Matemática e Fundamentos da Computação (GMFC) e o grupo de Pesquisa em Processamento Paralelo e Distribuído (G3PD).

Busca-se a execução da aplicação *ParallelPassBreak* na VirD-GM (*Virtual Distributed Geometric Machine Model*), a partir das especificações obtidas com uso das facilidades do VPE-GM (*Visual Programming Environment for the Geometric Machine Model*), que disponibiliza construtores de processos e configurações de memória baseada nas abstrações da programação visual e fundamentadas no modelo GM [Munhoz,2009].

A partir da identificação das principais abstrações, como a definição de processo elementar e sua implementação por estruturas recursivas denominadas envelopes, o trabalho visa adequar o programa ao ambiente de execução distribuído VirD-GM, incluindo uma análise do escalonamento e da distribuição efetiva das tarefas no ambiente de execução VirD-GM.

De acordo com a Figura 3, tem-se a modelagem dos três processos que definem a aplicação *ParallelPassBreak* no VPE-GM:

(i) *PassD,* expressão correspondendo ao processo elementar que gera as tarefas, e escreve na posição de memória (3);

(ii) *PassC*, processo gerado pelo construtor interativo paralelo, indicando um dos 25 processos que lê, a partir de uma memória, as operações e as configurações dos dados de entrada de uma tarefa para realizar a correspondente execução, colocando o resultado em uma outra posição (de 4 a 29) predefinida da memória.
(iii) *SumP*, processo que realiza a concatenação das senhas obtidas na etapa anterior, escrevendo o resultado final na posição de memória (30).

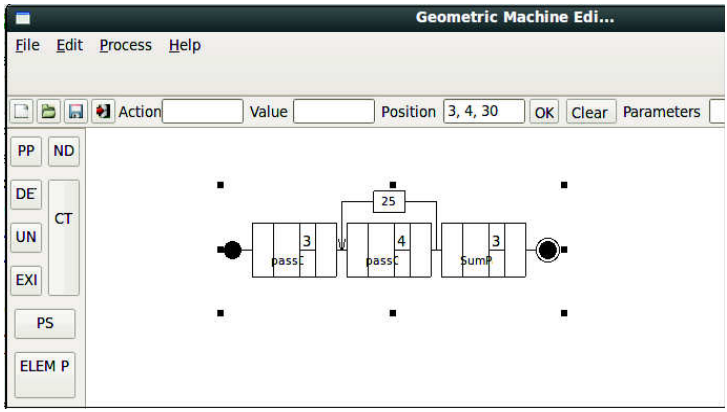

Figura 3. Modelagem da Aplicação ParallelPassBreak no VPE-GM

Estão em desenvolvimento os correspondentes testes para execução paralela e distribuída da aplicação *ParallelPassBreak* no VirD-GM, considerando diferentes arquiteturas multiprocessadas.

Referências

MUNHOZ, FELIPE (2009)"Expandindo o VirD-GM para Suporte às Demandas do Projeto D-GM", CC-CP/ UCPel, 81 p. (Monografia do Projeto de Graduação),2009.

LEOPOLD, CLAUDIA, "Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches",Wiley-Interscience, 2000.