4º Trabalho Prático

Tiago Alves Macambira < tmacam@dcc.ufmg.br >

4 de Agost de 2003

1 Os algorítmos

Nesse trabalho utilizamos 2 dos 3 algoritmos estudados no 3º trabalho prático: o shit-and exato e o shift-end aproximado. Ambos utilizam uma idéia simples, a de usar um autômato finíto não-determinístico (NDFA) pare realizar a busca de um padrão no texto. Contudo, a implementação desses algoritmos é feita com tal geniosidade que temos um algoritmo não somente eficiente mas relativamente fácil de entender. Esses dois algoritmos utilizam o paralelismo de bits das palavras de um computador, de forma muito engenhosa. tanto para representar as transições possíveis entre estados como para representar o próprio estado do autômata (estados ativos).

Uma explicação mais detalhada do funcionamento de ambos os algoritmos pode ser obtido no trabalho prático anterior e em [1].

2 Paralelismo

Nesse trabalho prático pedia-se que fosse projetado "*um algoritmo paralelo para recuperar ocorrências de padrões em arquivos constituídos de documentos e distribuídos através de uma rede de estações de trabalho*"[2].

Para tanto, foram implementados 2 programas:

- um programa ESCRAVO, responsável por fazer a busca por padrão;
- um programa MESTRE que gerenciava os programas escravos, enviando a estes requisições de busca por padrões.

Ambos os programas foram feitos utilizando sockets em ambiente unix.

2.1 Mestre

Este programa é o responsável por receber, através da linha de comando, requisições de busca por casamento de padrões em texto e enviá-las para os vários programas escravos, para que estes fizessem a pesquisa. Após receber os resultados dos escravos, esse programa sumariza essas informações e as retorna para o usuário.

Algorithm 1 Master

```
Processa a linha de comando
Conecta-se com os escravos
Envia as requisições aos escravos
Processa o resultado dos escravos
```

Diferentemente do que foi solicitado em [2], não estamos informando as linhas onde ocorreram os casamentos, mas apenas a quantidade total de linhas onde encontramos ocorrências. Apesar de ser uma vesão mais simples do problema, não diminuímos o esforço computacional realizado, apenas simplificamos o processo de comunicação.

A estrutura geral do programa *master* é mostrada em no Algorítmo1. Esse programa utiliza, no passo onde ocorre o processamento dos resultados dos escravos a função SELECT para gerenciar a comunicação entre os vários clientes de forma eficiente.

2.2 Escravos

Este programa é o responsável por fazer, sob pedido do programa mestre, buscas por casamentos de padrões. Ele fica num *loop* infinito a espera de requisições e, sempre que uma for recebida, ele a processa, retorna o resultado para o mestre e volta a esparar por uma nova requisição. Dessa forma, pode-se utilizar um mesmo escravo várias vezes.[1]

Para cada requisição, o programa escravo retorna

- o número de linhas com ocorrências encontradas sua respectiva partição dos dados;
- o tempo que ele levou para realizar a consulta.

Com esses dados, o mestre pode, ao somar os valores relatados por cada escravo individualmente, informar o número total de linhas com ocorrências existentes nos arquivos pesquisados. Pode também observar quais escravos trabalham mais lentamente[2].

3 Resultados

3.1 Considerações Iniciais

Antes de iniciarmos uma discursão sobre os resultados obtidos, alguimas considerações devem ser feitas para que possamos observar de forma mais clara o significado dos resultados, o motivo pelo qual alguns resultados inesperados foram obtidos, etc.

[1] mas não concorrentemente.

[2] Tal informação pode ser útil para observar problemas com os escravos ou com as máquinas onde os escravos estão rodando.

3.1.1 Sobre a implementação

O algoritmo de busta implementado no programa SLAVE (slave.c e search.c) realiza um *loop* no qual ele lê, linha a linha, através da função FGETS, o conteúdo do arquivo. Cada linha é individualmente passada para a função de busca e casamento de padrões que retorna tão logo seja encontrado um casamento. Essa função por sua vez, a cada chamada, realiza um processo de inicilização para o processo de casamento. Esse processo é mais custoso no algoritmo para casamento aproximado.

Devido a essa estrutura, intuitiva mas ineficiente, os valores de tempo de execução do algoritmo são consideravelmente altos quando comparados com o tempo que uma execução, com os mesmos parâmetros, do programa *agrep*.

3.1.2 Sobre o processo de execução das consultas

Esse processo foi automatizado utilizando-se uma script chamada FAZ_TUDO.SH. Essa script realiza todas as chamadas necessárias ao programa master e relata algumas informações sobre o andamento do processo de execução na tela. Estas informações, juntamente com a saída das várias instâncias do programa master são posteriormente condensadas num arquivo CSV através de uma outra script, a PARSE_ESTATISTICAS.PL. Esse arquivo CSV pode ser lido em praticamente qualquer planilha eletrônica e desta forma permite uma fácil extração dos dados.

3.1.3 Sobre o ambiente

Durante o processo de execução das consultas, foi verificado que a máquima WOLVERINE, utilizada tanto para os testes com 2 e com 4 escravos, estava inacessível. Devido a isso, nos resultados abaixo, quando conveniente, utilizaremos a média de tempo das outras máquinas como sendo o valor de tempo utilizado pela máquina wolverine.

Observou-se também que, em paralelo à execução do programa escravo, durante o tempo destinado a esse experimento, na máquina CICLOPE também era executado o seguinte programa: *"tcsh -c /usr/bin/nice -1 perl /home/pos/aldo/processatp2.pl"*. Esse script, rodando com prioridade acima da normal, executava um programa escrito em java. Devido a isso, para as consultas utilizando 4 escravos, encontaremos discrepâncias entre os tempos de execução consumidos nos escravos, o que refletirá negativamente no tempo de execução total dessas consultas e, conseqüentemente no *speed-up* conseguido com 4 escravos.

Para ilustrar esse fato, observe a tabela 1, onde consta o tempo de relógio utilizado em cada escravo para o processamento de uma consulta e o tempo total reportado pelo programa mestre para essa consulta:

Observa-se que o tempo de execução, nessa instância, na máquina ciclope, foi 6 vezes mais lento do que nas outras máquinas, que possuem tempos próximos. Esses comportamento não ocorreu isoladamente, sendo na verdade a regra para todo o processo de execução das consultas utilizando 4 escravos. Fossem as máquinas iguais em condições de processamento no instante dos testes, os resultados obtidos seriam mais significativos para os propositdos do trabalho.

Máquina	Tempo no escravo	Relativo ao nó mais veloz	Relativo ao tempo no mestre
vampira	32,625158s	1	0.16363
ciclope	199,375697s	6.111	0.99996
fera	35,628149s	1.092	0.17869
Total	199,382762	6.111	1

Tabela 1: Busca exata pelo pedrão "State Department" com 4 escravos

3.2 Saída do Programa

A saída da execução do programa contra-se em anexo e disponível ná versão eletrônica deste documento, no arquivo SRC/TYPESCRIPT. O arquivo com dados condensados da saída das consultas encontra-se em src/estatiticas.csv. Nesse arquivo, para a execução com 4 escravos, os tempos do escravo n° 0,1,2 e 3 referem-se, respectivamente, aos tempos reportados pelos escravos rodando nas máquinas vampira, ciclope, fera e wolverine[3].

3.3 Avaliação empírica

3.3.1 Speedup

Define-se speedup como sendo:

$$S = \frac{T(1)}{T(n)}$$

onde $T(1)$é o tempo levado pela versão seqüêncial do algorítmo e $T(n)$ como o tempo levado pelo do algorítmo paralelo usando n processadores.

Nos nossos experimentos conseguimos os seguintes valores pra *speedup* (na média):

N° de Erros	N° de escravos 1	2	4
0	1	2,13	0,67
1	1	2,07	0,64
2	1	2,05	0,64
3	1	2,03	0,64

Observe que o speedup obtido com 4 máquinas é completamente inesperado: a performance cai para níveis inferiores aos das execuções seqüênciais. Isso deve-se a máquina CICLOPE, que, como mencionado na seção 3.1.3, estava dividindo seu tempo de processamento com outras tarefas.

Observe também que quando contabilizamos os tempo de execução para 2 escravos estamos na verdade contabilizando o tempo utilizado apenas para uma máquina, uma vez que a máquina WOLVERINE estava inacessível.

Se levarmos em consideração que

[3] Lembrando: A máquina wolverine encontrava-se inacessível durante os testes, motivo pelo qual todos os tempos reportados para essa máquina serão iguais a 0.

- o tempo gasto pela parte serial desse algoritmo é muito pequeno frente ao tempo gasto na parte paralela;
- os tempos da máquina CICLOPE estão muito acima do esperado e que tal comportamento não seria encontrado num ambiente paralelo dedicado;
- que nos tempos obtidos nas execuções com 4 escravos, os resultados da máquina FERA são muito próximo da máquina vampira

talvez seja interessante observar apenas o *speedup* médio conseguido pela máquina VAMPIRA. É fato que esses valores não refletirão o tempo gasto com a parte serial do algoritmo (enviar requisições, tratar requisições e respostas, etc) mas, pelo exposto acima, julga-se que tais valores darão uma idéia aproximada do que consigiríamos em condições ideiais[4]:

N° de Erros	N° de escravos		
	1	2	4
0	1	2,14	4,11
1	1	2,14	4,11
2	1	2,14	4,11
3	1	2,14	4,11

3.3.2 Tempos

Na tabela abaixo, mostramos a soma dos valores de tempo em μsegundos reportados pelo processo mestre, variando-se o número de escravos e o número de erros permitidos:

N° de Erros	N° de escravos		
	1	2	4
0	403496181	188662236	600745342
1	833135954	401834900	1295250003
2	1079122466	524683586	1693878512
3	1330272313	653271495	2088663245
Total	3646026914	1768452217	5678537102

4 Versão Eletrônica

A versão eletrônica desse documento bem como os fontes dos programas, scripts, saídas dos programas, planilhas, gráficos, etc encontram-se disponíveis em:

http://www.dcc.ufmg.br/~tmacam/PAA/PAA4/

[4] Ou pelo menos em condições mais favoráveis

Referências

[1] "Projeto de Algoritmos com implementações em Pascal e C". Ziviani, Nívio. 5a./6a. edição

[2] http://www.dcc.ufmg.br/~nivio/cursos/pa03/tp4/pa03tp4.html