UNIVERSIDADE FEDERAL DE LAVRAS

PPD – Trabalho 2

Relatório

Responsáveis:

João Paulo Costa
Renato Carvalho Alvarenga
Vinícius Salles Pereira

Lavras - Março 2017

1. O PageRank

GCC 126 – Programação Paralela e Distribuída
Professora Marluce Rodrigues Pereira
Departamento de Ciência da Computação - DCC
Universidade Federal de Lavras - UFLA

PageRank é um algoritmo utilizado pela ferramenta de busca Google para posicionar websites entre os resultados de suas buscas. O PageRank mede a importância de uma página contabilizando a quantidade e qualidade de links apontando para ela.

Na construção da métrica de PageRank, a web é vista como uma rede de citações, cada nó corresponde a uma página e cada ligação corresponde a uma referência de uma página para outra (hiperligação). A métrica atribuí um valor a cada nó (página) da rede, um valor maior corresponde a um nó mais importante na rede. Do ponto de vista da teoria das redes, PageRank é uma métrica de centralidade. Esta métrica tira partido da estrutura de hiperligações na web para produzir o valor para cada página da rede. Uma hiperligação a uma página conta como um “voto” de suporte.

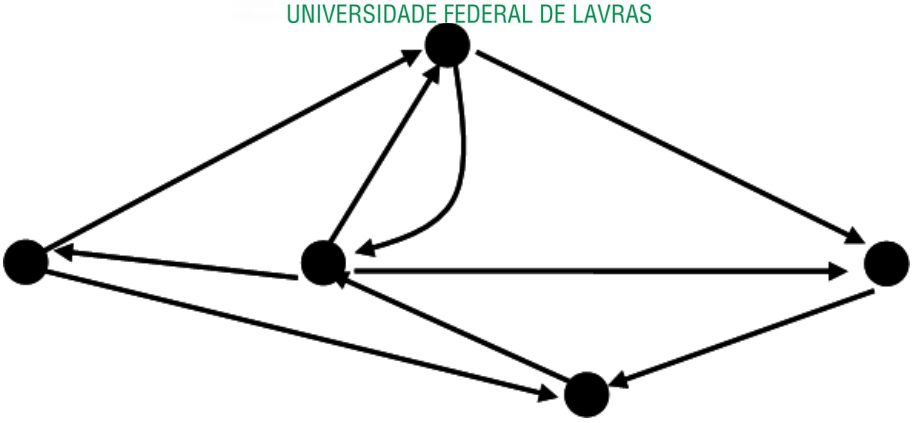
Embora PageRank seja o nome dado ao algoritmo cujo a empresa Google tem o direito de uso, existem diferentes variações descritas para este mesmo algoritmo, isso acontece pois, o algoritmo pode ser modelado de diferentes formas, permitindo o desenvolvedor definir o que implica em um maior valor de PageRank. Existem variações que priorizam o número de ligações que uma página recebe, e outras variações que priorizam o número de ligações que uma página realiza em outras páginas.

Portanto o cálculo do PageRank pode ser definido de acordo com o propósito da aplicação ao qual ele está inserido.

2. Decisões de projeto (Construção do Algoritmo)

O algoritmo de PageRank possui diversas variantes, onde não apenas considera-se os backlinks(links que apontam para um nó) e fowardlinks(links que são apontados por um nó). No caso de situações mais avançadas é comum se deparar com códigos complexos que tentam ao máximo simular o comportamento de um usuário ao navegar pela internet. Dumping é uma das abordagens que tentam simular um ação de um usuário ocioso na web, indentificando as possibilidades de fluxo.

Diante dessas considerações criamos uma própria versão do Algoritmo PageRank, para sermos efetivos em executar o conceito da aplicação sem aprofundarmos em uma possível complexidade que dificultaria a aplicação. Nesta versão cada nó possui o mesmo valor percentual para dividir entre os nós ao quais ele faz ligações de encaminhamento, no caso os fowardlinks.



No exemplo acima, cada nó possui 20% (100%/n, sendo n o número de nós) para serem distribuídos entre os nós que estão ligados através dos fowardlinks, pois inicialmente os nós são alocados igualmente com um pré pagerank.

Para se realizar o cálculo do PageRank é verificada a condição se o nó em questão aponta para outro nó, se sim, ele divide a valor que ele possui para distribuir entre ele e os demais nós apontados por ele (no exemplo ele divide os 20%). Caso o nó não aponte para algum outro nó, ou seja ele não possui fowardlinks, então ele divide o valor que ele possui(pré pagerank) entre todos os nós do grafo. Isso acontece para que possamos simular de uma maneira mais apropriada a navegação do usuário pela web, em que numa situação de deriva na web o fluxo continue de maneira aleatória.

3. Paralelizando o código

A paralelização do código foi feita tanto a nível de threads, quanto a nível de processos, sendo que, a nível de threads foi utilizado o openmp e a nível de processos o mpi.

O OpenMP (do inglês Open Multi-Processing, ou Multi-processamento aberto) é uma interface de programação de aplicativo(API) para a programação multi-processo de memória compartilhada em múltiplas plataformas. Permite acrescentar simultaneidade aos programas escritos em C, C++ e Fortran sobre a base do modelo de execução fork-join. Está disponível em muitas arquiteturas, incluindo as plataforma Unix e Microsoft Windows. É constituída por um conjunto de diretivas de compilador, rotinas de biblioteca, e variáveis de ambiente que influenciam o comportamento do tempo de execução.

Message Passing Interface (MPI) é um padrão para comunicação de dados em computação paralela. Existem várias modalidades de computação paralela, e dependendo do problema que se está tentando resolver, pode ser necessário passar informações entre os vários processadores ou nodos de um cluster, e o MPI oferece uma infraestrutura para essa tarefa.No padrão MPI, uma aplicação é constituída por um ou mais processos que se comunicam, acionando-se funções para o envio e recebimento de mensagens entre os processos. Inicialmente, na maioria das implementações, um conjunto fixo de processos é criado. O objetivo de MPI é prover um amplo padrão para escrever programas com passagem de mensagens de forma prática, portátil, eficiente e flexível.

Escolheu-se usar a abordagem de threads através do openmp para paralelizar todos os laços *for* utilizados na implementação. Utilizamos essa prática pois se tratava de uma técnica exaustivamente utilizada na disciplina. Por outro lado o MPI foi usado para que processos fossem criados afim calcular cada um separadamente o pagerank através da matriz que armazena os nós e fowardlinks. As linhas da matriz são divididas igualmente entre os processos que se comunicam entre si através de uma sequência de MPI_Send e MPI_Receive, em que as submatrizes são novamente congregadas por um processo mestre; àquele que possui o seu rank correspodente à 0.

4. Avaliações

Os testes foram realizados em dois computadores diferentes, onde cada um executou o código com 1, 2, 4 e 8 processos, assim como consta na descrição do trabalho. O primeiro computador utilizado foi um desktop de um dos laboratórios do Departamento de Ciência da Computação, cujas especificações são: Intel(R) Core(TM) i3-370M com 2 núcleos físicos e 4 núcleos lógicos. O segundo computador utilizado foi um notebook de um dos membros da equipe, cujas especificações são: Intel(R) Core(TM) i7- 4510U com 2 núcleos físicos e 2 núcleos lógicos. Ambas máquinas possuem a mesma quantidade de memória ram, 8 GB.

Outro aspecto a ser colocado em evidência é o tamanho da entrada para a solução do problema. Nos testes utilizamos entradas de 1000, 10000 e 100000 nós, onde esses representam diretamente páginas na web.

Processador i3 - dois núcleos físicos e quatro processadores lógicos(1000 nós):

PC UFLA (1000 nós)	1 processo	2 processos	4 processos	8 processos
	0.024454	0.020910	0.080677	0.075621
	0.023298	0.018517	0.068497	0.093239
	0.022811	0.024948	0.063846	0.112390
	0.022968	0.020976	0.070434	0.128974
	0.023838	0.021678	0.067763	0.118027
	0.021706	0.021794	0.072120	0.109247
	0.023622	0.021755	0.067489	0.137473
	0.023724	0.023603	0.072350	0.132682
	0.022462	0.026012	0.067337	0.084881
	0.022738	0.028163	0.076588	0.086180

Processador i3 - dois núcleos físicos e quatro processadores lógicos(10000 nós):

PC UFLA (10000 nós)	1 processo	2 processos	4 processos	8 processos
	0.103733	0.069703	0.088996	0.092798
	0.109476	0.070354	0.103077	0.123386
	0.109155	0.066514	0.085507	0.138298
	0.107797	0.068024	0.105969	0.108204
	0.102623	0.069873	0.086386	0.095516
	0.098833	0.063604	0.089444	0.086143
	0.107357	0.067175	0.089661	0.092618
	0.098446	0.069236	0.099702	0.087773
	0.101208	0.067008	0.083583	0.144215
	0.103573	0.067721	0.094200	0.130152

Processador i3 - dois núcleos físicos e quatro processadores lógicos(100000 nós):

PC UFLA (100000 nós)	1 processo	2 processos	4 processos	8 processos

	4.746843	2.435007	1.367267	0.783456
	4.797456	2.418942	1.340040	0.804929
	4.730945	2.439218	1.326215	0.821707
	4.859981	2.456890	1.336217	0.750601
	4.704787	2.188053	1.361491	0.807930
	4.819361	2.426310	1.374734	0.797561
	4.837848	2.373175	1.346093	0.767063
	4.792086	2.451480	1.356921	0.870992
	4.802260	2.400074	1.361765	0.833250
	4.830092	2.380792	1.380908	0.809946

Processador i7 - dois núcleos físicos e dois processadores lógicos(1000 nós):

PC Equipe (1000 nós)	1 processo	2 processos	4 processos	8 processos
	0.018797	0.035105	0.064651	0.140118
	0.018797	0.032634	0.062571	0.131890
	0.020724	0.032634	0.065938	0.156667
	0.034951	0.027453	0.053582	0.189138
	0.016954	0.036353	0.050472	0.137646
	0.019541	0.075556	0.052025	0.092242
	0.021298	0.037101	0.071452	0.129593
	0.017911	0.038289	0.088683	0.125104
	0.034160	0.031774	0.051003	0.088358
	0.017084	0.060300	0.092489	0.099611

Processador i7 - dois núcleos físicos e dois processadores lógicos(10000 nós):

PC Equipe (10000 nós)	1 processo	2 processos	4 processos	8 processos
	0.116265	0.097881	1.204029	2.334026
	0.111289	0.065897	1.309997	2.551253
	0.119152	0.068471	0.783353	2.191182
	0.121917	0.095594	1.560102	3.712476
	0.119330	0.073012	1.608450	2.313615
	0.113693	0.060919	1.131817	2.036091

	0.116465	0.058926	1.373397	3.680549
	0.114626	0.078863	1.092968	1.953733
	0.117487	0.062850	1.589882	3.424987
	0.105130	0.069649	0.354037	2.149008

Processador i7 - dois núcleos físicos e dois processadores lógicos(100000 nós):

PC Equipe (100000 nós)	1 processo	2 processos	4 processos	8 processos
	3.611872	1.827689	341.509541	----------
	3.625605	1.752509	491.533694	----------
	3.461967	1.663035	390.343576	----------
	3.682101	1.681506	----------	----------
	3.669137	1.665487	----------	----------
	3.567029	1.638655	----------	----------
	3.594351	1.647284	----------	----------
	3.522400	1.758525	----------	----------
	3.645451	1.758572	----------	----------
	3.533907	1.713563	----------	----------

Considerando o desktop i3, verifica-se um comportamento peculiar nos testes. Ao utilizarmos um número de processos acima de quatro temos um aumento substancial no tempo de execução. Acredita-se que isso ocorra devido ao tempo gasto na comunicação dos processos (MPI_Send e MPI_Receive), ainda mais evidente com um número maior de entrada. Tal fato nos remete que existem casos onde o uso de threads, onde temos memória compartilhada na paralelização, seja mais eficiente.

Considerando o notebook i7, tem-se a mesma peculiaridade do caso anterior. Um número maior de processor exige gastos na comunicação e isso interfere diretamente no tempo de execução. Entretanto os testes realizados nessa máquina na entrada de 100000 nós foram demasiadamente inferiores se compararmos o computador anterior, em que testes com 4 e 8 processos se tornaram praticamente impraticáveis devido a demora em se encontrar a solução.

É visível que os núcleos lógicos a mais do processador i3 proporcionaram um melhor desempenho em relação ao processador i7. Este resultado é importante pois reforça a premissa que um processador i3 pode sim ser mais eficiente que um processador i7 dependendo da arquitetura utilizada.

4. Conclusão

Após os testes realizados com os dois diferentes tipos de processadores, com a variação do tamanho do problema (número de nós) e das diferentes abordagens na solução(1, 2, 4 e 8 processos), pudemos verificar que quanto maior o número de processos, maior gasto na comunicação entre os mesmos foi observada, fator que influenciou diretamente no tempo de execução do algoritmo.

Outro ponto importante a se salientar é que por conta do problema de cálculo de PageRank ser uma abordagem complexa, tornou-se necessário simplificar o algoritmo para que a implementação não fosse uma barreira para o desenvolvimento do projeto. É possível ainda inferirmos que problemas diferentes demandam soluções diferentes e que muitas vezes o uso do MPI seja mais efetivo na utilização de um cluster de computadores, pois como na situação do nosso problema revelou, a utilização de um processo que possui quatro threads seria mais eficiente do que a utilização de quatro processos que necessitam de gastos substanciais para comunicação entre eles.

Finalmente mas não menos importante, se foi possível reafirmar premissas de que um processador i3 pode ser sim mais produtivo do que um processador i7 dependendo da sua arquitetura implementada.