Estudo Comparativo de *Software* para Particionamento e Mapeamento de Grafos

Elias C. Carvalho
PPGC da UFRGS
Cesumar/Univale
elias@cesumar.br

Ricardo V. Dorneles
PPGC da UFRGS
cadinho@inf.ufrgs.br

Rogério L. Rizzi
PPGC da UFRGS
rizzi@inf.ufrgs.br

Tiarajú A. Diverio
PPGC da UFRGS
diverio@inf.ufrgs.br

Resumo

Aplicações e arquiteturas de computadores podem ser representadas por meio de grafos. O particionamento do grafo de uma aplicação e seu respectivo mapeamento sobre o grafo de uma arquitetura é uma forma de paralelizar o processamento. Nesse contexto, é importante considerar como o grafo da aplicação será particionado, pois uma distribuição desigual sobre um ambiente heterogêneo, certamente reduzirá a eficiência do processamento, pois alguns processadores terminarão sua tarefa antes e permanecerão ociosos até que os outros alcancem o ponto de sincronização para receberem novos dados e executarem novas tarefas. Este artigo apresenta a realização de experimentos de particionamento e mapeamento, representando o problema e a arquitetura através de grafos e utilizando ferramentas específicas para esse fim.

1. Introdução

Quando há a necessidade de processamento de alto desempenho, muitos pensam em grandes máquinas dedicadas (supercomputadores), que custam milhões de dólares, são difíceis de serem operadas e exigem salas superprotegidas. No entanto, hoje em dia, devido às tecnologias das redes de alta velocidade, é possível a construção de *clusters* de computadores agregando-se várias máquinas, o que viabiliza o uso do processamento de alto desempenho na solução de problemas em diversas áreas. Existe ainda a possibilidade nestas instalações de não se ter uma única sala com as máquinas, pois estas podem estar distribuídas em vários locais diferentes. A vantagem do custo é óbvia, pois computadores pessoais custam muito menos do que o mais barato dos supercomputadores.

Com *clusters* de computadores, tem sido possível resolver várias aplicações em ambientes acadêmicos sem a necessidade de ter um centro de supercomputação. Um exemplo disto é a modelagem do Lago Guaíba em Porto Alegre (mais conhecido por Rio Guaíba), que é um dos trabalhos desenvolvidos pelo Grupo de Matemática da Computação e Processamento de Alto Desempenho do Programa de Pós-Graduação em Computação da UFRGS, que tem o objetivo simular a hidrodinâmica e o transporte de substâncias em corpos de água [1, 2].

Inicialmente, os *clusters* são homogêneos, mas devido a necessidade de manutenção, de expansão e a idéia de fazer uso de todos os recursos computacionais disponíveis, inerentes aos meios acadêmicos ou empresariais, unem-se as máquinas mais lentas às mais recentes para extrair processamento de maior desempenho. Esse fator resulta na criação de *clusters* heterogêneos. A maior limitação para programar nesses ambientes surge da dificuldade de balancear a carga, pois quando se utiliza máquinas e redes de interconexão com características diferentes, os nós devem receber cargas proporcionais ao seu poder de processamento.

Em um ambiente heterogêneo, surge então a necessidade de mapear e balancear aplicações de forma a explorar seu potencial máximo. É nesse contexto que se situa esse trabalho, que tem por objetivo utilizar grafos para representar a arquitetura e a aplicação, durante a realização de experimentos com algumas ferramentas para particionamento e mapeamento de grafos.

2. O grafo da arquitetura

Um dos objetivos deste trabalho é modelar o *cluster* do Instituto de Informática da UFRGS por intermédio de um grafo denominado "grafo da arquitetura". Esse cluster é do tipo heterogêneo multiprocessado [3] e constituído por 10 nós interconectados por três redes de comunicação: *Fast-Ethernet*, *Myrinet* e SCI. Sua arquitetura pode ser observada na fig. 1 e detalhes dessa configuração são representados pela tab. 1. Por intermédio desses detalhes, percebe-se a heterogeneidade do *cluster*, uma vez que existem máquinas com um e dois processadores, processadores com velocidades diferentes e com diferentes capacidades de memória.

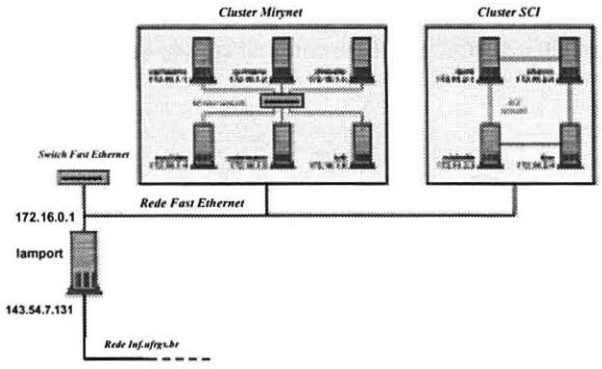

Figura 1– *Cluster* do Instituto de Informática da UFRGS

Tabela 1– Configuração dos nós do *cluster* do Instituto de Informática da UFRGS

Sq	Máquina	CPU (MHz)	RAM (Mb)
0	Lamport	Pentium III 600	620
1	Veríssimo	Dual Pentium Pro 200	128
2	Dionelio	Dual Pentium Pro 200	128
3	Euclides	Dual Pentium Pro 200	128
4	Quintana	Dual Pentium Pro 200	128
5	Luft	Pentium Pro 200	128
6	Ostermann	Pentium Pro 200	128
7	Kim	Pentium III 500 dual	256
8	Sharon	Pentium III 500 dual	256
9	Demi	Pentium III 500 dual	256
10	Michelle	Pentium III 500 dual	256

Com base em grafos de [4] que modelam uma arquitetura paralela representando uma rede completamente interconectada e uma rede em anel. Criou-se a versão inicial do grafo da arquitetura (fig. 2).

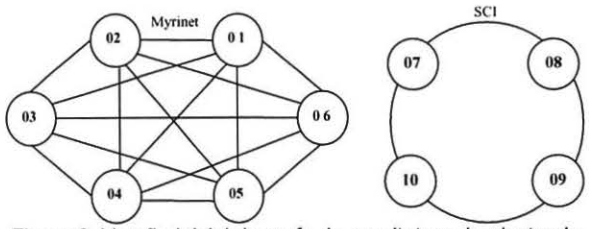

Figura 2 Versão inicial do grafo da arquitetura do *cluster* do Instituto de Informática da UFRGS

Após a criação da versão inicial do grafo, iniciou-se a fase de avaliação dos nós e da rede para atribuir pesos aos vértices e arestas do mesmo. Para isso, foram escolhidos três parâmetros, considerados relevantes para a aplicação a ser utilizada, dois deles expressam o desempenho de cada máquina do *cluster* e um representa o desempenho da rede.

Para avaliar o desempenho das máquinas, foram realizados experimentos com dois programas de *benchmark*: o *Flops* [5] que propõe um teste simples para avaliar a capacidade bruta de cálculos de ponto flutuante dos processadores e o *Stream* [6] que realiza medidas de desempenho da memória RAM para quatro tipos básicos de operações da álgebra linear comuns em computação científica.

Realizados os experimentos iniciais e obtidos os resultados, adotou-se o seguinte critério para elaboração dos pesos para os vértices e arestas do grafo da arquitetura: o *cluster* possui 10 máquinas e cada uma representa uma parte sobre a capacidade total de processamento do mesmo, de modo que a soma dessas partes resulta em 100% da capacidade total. Portanto, com base nos experimentos realizados, calculou-se quanto cada máquina representa dessa capacidade. Os detalhes de toda essa operação podem se observados na tab. 2. Nessa tabela, a coluna "1 Proc" representa o resultado obtido com o programa *Flops* para 1 processador, a coluna "2 Proc" apresenta o mesmo resultado para dois processadores descontando-se o tempo de contenção. A coluna "Copf" indica a capacidade de cada máquina para efetuar operações de ponto flutuante, considerando as 10 máquinas do *cluster*. A coluna "MB/s" mostra o resultado obtido com o programa *Stream*, a coluna "Camr" denota a capacidade da memória para cada máquina considerando as 10 máquinas do *cluster*.

Com os resultados referentes ao desempenho em operações de ponto flutuante e desempenho da memória,. foi calculado o valor referente ao poder computacional de cada máquina. Para este trabalho, entenda-se que o termo "*Poder Computacional*", representado pela coluna "Pc" na tab. 2, foi concebido, considerando a capacidade da máquina para realizar operações de ponto flutuante e a velocidade de sua memória RAM. Para representar essa métrica, foi elaborada a seguinte fórmula:

$$P_c = ((C_{opf} * \alpha) + (C_{amr} * \beta)) / \Delta$$

onde:

P_c = Poder Computacional;
C_{opf} = Capacidade de realização de operações de ponto flutuante;
α = Peso atribuído para C_{opf};
C_{amr} = Capacidade de acesso à memória RAM;
β = Peso atribuído para C_{amr};
Δ = Soma dos pesos α e β

Essa fórmula pode ser ajustada de acordo com o propósito da aplicação a ser particionada e mapeada em uma determinada arquitetura. Para isso, recomenda-se fazer uma análise da aplicação, para identificar as necessidades da mesma, como por exemplo: necessidade de acesso a disco, que pode ser avaliado através do programa de *benchmark* Iozone [7]. Em seguida, basta ajustar a fórmula na tentativa de obter o máximo de

precisão. Como exemplo, para uma aplicação que utilize muito mais o processador do que a memória, a fórmula mais aproximada deveria ser ajustada com os seguintes parâmetros:

$$P_c = ((C_{opf} * 8) + (C_{amr} * 2))/10$$

Com base neste exemplo e com os resultados obtidos por intermédio dos programas de *benchmark*, o poder computacional pode ser observado na tab. 2.

Tabela 2 – Capacidade para efetuar operações de ponto flutuante e velocidade da memória RAM

Sq	1 Proc	2 Proc	Copf	MB (s)	Camr	Pc
1	24,7339	48,1198	3,4%	312,5590	4,4%	3%
2	24,7398	48,1313	3,4%	404,3478	5,7%	4%
3	24,7545	48,1600	3,4%	418,6335	5,9%	4%
4	24,7373	48,1265	3,4%	384,2352	5,4%	3%
5	24,7247	24,7247	1,7%	413,4387	5,8%	2%
6	24,7226	24,7226	1,7%	283,3334	4,0%	2%
7	170,1511	295,7976	20,7%	1.222,8571	17,2%	21%
9	170,1222	295,7474	20,7%	1.196,1905	16,8%	21%
8	170,2607	295,9882	20,8%	1.249,5238	17,6%	20%
10	170,5568	296,5030	20,8%	1.222,8571	17,2%	20%

Para avaliar o desempenho da rede foi utilizado o programa de *benchmark Pingpong* [8] que fornece informações de largura de banda e latência da comunicação entre dois processos. O cálculo do peso referente à comunicação da rede, foi obtido calculando a média aritmética do valor referente à comunicação da máquina "A" para a máquina "B" e a comunicação da máquina "B" para máquina "A". Durante esse processo verificou-se que as máquinas da rede Myrinet conseguem se comunicar com as máquinas da rede SCI. Portanto, o grafo apresentado na fig. 2 foi alterado para uma nova versão, observada na fig. 3, na qual todas a máquinas se comunicam entre si, porém com pesos diferentes entre suas arestas para modelar a comunicação.

A diferença entre uma comunicação mais rápida ou mais lenta certamente influenciará a distribuição de carga. Dessa maneira, um nó receberá sua carga não só em função de seu poder computacional, mas também, da velocidade de sua comunicação com outros nós.

Os pesos das arestas não aparecem na fig. 3 para facilitar sua visualização, mas podem ser observados na tab. 3. Nela a primeira coluna mostra o número do vértice origem e as colunas subsequentes mostram o peso atribuído e o número dos vértices adjacentes ao vértice origem.

O fato de todos os nós poderem se comunicar entre si indica que é possível efetuar o mapeamento do grafo da aplicação sobre o grafo da arquitetura, utilizando-se todos os nós do cluster. No entanto a comunicação não é simultânea, pois os nós se comunicam via barramento.

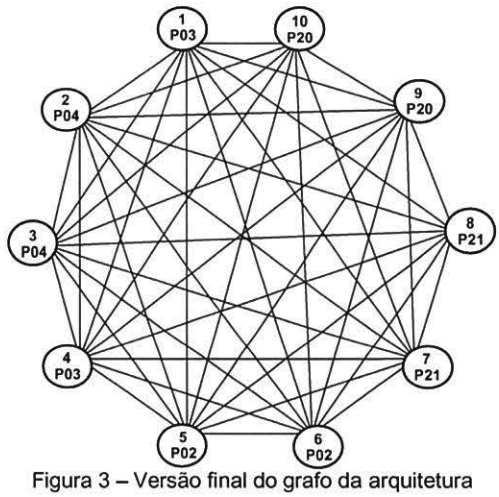

Figura 3 – Versão final do grafo da arquitetura

Tabela 3 – Peso das arestas entre os vértices

	Peso e número dos vértices adjacentes ao vértice origem																	
O	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N	P	N
0	4	1	4	2	4	3	4	4	4	5	1	6	1	7	1	8	4	9
1	4	0	3	2	4	3	4	4	4	5	1	6	1	7	1	8	4	9
2	4	0	3	1	4	3	4	4	4	5	1	6	1	7	1	8	4	9
3	4	0	4	1	4	2	4	4	4	5	1	6	1	7	1	8	4	9
4	4	0	4	1	4	2	4	3	4	5	1	6	1	7	1	8	4	9
5	4	0	4	1	4	2	4	3	4	4	1	6	1	7	1	8	2	9
6	1	0	1	1	1	2	1	3	1	4	1	5	1	7	1	8	1	9
7	1	0	1	1	1	2	1	3	1	4	1	5	1	6	1	8	1	9
8	1	0	1	1	1	2	1	3	1	4	1	5	1	6	1	7	1	9
9	4	0	4	1	4	2	4	3	4	4	2	5	1	6	1	7	1	8

3. O grafo da aplicação

Entre os trabalhos que vem sendo desenvolvidos pelo GMCPAD, destaca-se o modelo computacional paralelo do Lago Guaiba (mais conhecido como Rio Guaiba), do qual originou o projeto denominado "HIDRA", que tem o objetivo de simular a hidrodinâmica e o transporte de substâncias em corpos de água [1,2].

Vários modelos computacionais foram implementados, em linguagem C, utilizando o compilador gcc 2.91.60 sobre o sistema operacional Linux 2.2.1. A biblioteca de troca de mensagens utilizada foi o MPICH versão 1.2.1 [9], uma implementação do padrão MPI 1.0 desenvolvida no Argonne National Laboratory. A fig. 5 representa um desses modelos, que gera um grafo com 469 vértices e 1.692 arestas. Esse foi o grafo utilizado para a realização dos experimentos com particionamento e mapeamento deste trabalho. Para isso, o mesmo foi representado por arquivos que descrevem o formato do grafo, de acordo com os formatos exigidos pelas ferramentas de particionamento e de mapeamento utilizadas.

Figura 5 – Modelo computacional implementado para o Hidra

Parte do arquivo que descreve o grafo gerado pela aplicação HIDRA é apresentado na fig. 6 no formato de entrada para o *software* Scotch, onde grau do vértice representa o número de vértices adjacentes que ele possui.

Qtde. de Vértices	Qtde. de Arestas				
469	1692				
No. do Vértice	**Grau do Vértice**	**Vértices Adjacentes**			
0	2	1	5		
1	3	0	2	6	
2	3	1	3	7	
3	2	2	8		
4	2	5	10		
5	4	0	4	6	11
6	4	1	5	7	12
7	4	2	6	8	13
8	3	3	7	14	
9	2	10	17		
...					

Figura 6 – Descrição parcial do grafo gerado pelo HIDRA

4. Particionamento e mapeamento do grafo da aplicação

A forma de paralelização mais utilizada em modelos hidrodinâmicos como o HIDRA é a divisão do domínio entre diversos processadores. Nesse caso, é importante considerar como o domínio é particionado entre os processadores, pois uma distribuição desigual dos domínios entre eles, no caso de um *cluster* heterogêneo, por exemplo, certamente reduzirá a eficiência do processamento, fazendo com que alguns processadores terminem sua tarefa antes e permaneçam ociosos até que os outros alcancem o ponto de sincronização para receberem novos dados e executarem novas tarefas. A divisão desses domínios chama-se particionamento.

Depois de ser particionado, o grafo da aplicação, deve ser mapeado sobre o grafo da arquitetura. Aplicações paralelas geralmente são particionadas e realizam tarefas concorrentes que precisam executar algum tipo de comunicação entre si [10]. Antes das tarefas serem executadas, cada uma delas deve estar pré-determinada a ser executada em um processador específico. Nesse caso, um dos principais problemas é como alocar essas tarefas para determinados processadores, de forma que se possa diminuir o tempo de execução global da aplicação e, consequentemente, de comunicação. O assinalamento de tarefas para processadores chama-se mapeamento.

Ao descrever os experimentos cita-se a utilização de métodos globais e locais, que são métodos utilizados para realizar o particionamento dos grafos. O global efetua o particionamento baseado em algum algoritmo e o local recebe como entrada o grafo particionado pelo método global e visa melhorar a qualidade da partição, diminuindo o corte de arestas por intermédio do rearranjo dos vértices.

Para efetuar os experimentos com particionamento e mapeamento do grafo da aplicação, foram selecionados três ferramentas das quais apenas a terceira conseguiu efetuar o mapeamento, são elas: o Chaco [11], o Jostle [12] e o Scotch [13]. As próximas seções descrevem os experimentos realizados com cada uma delas.

4.1 Chaco

O Chaco 2.0 é um *software* para particionamento de grafos desenvolvido por Bruce Hendrickson e Robert Leland, no Sandia National Labs em Albuquerque, nos Estados Unidos. Esse *software* possui uma grande variedade de métodos e opções, muitos dos quais foram desenvolvidos pelos próprios autores. Alguns métodos exploram a geometria da malha, outros métodos a sua conectividade local ou a sua estrutura global.

As cinco classes de métodos de particionamento atualmente implementadas no Chaco utilizam os algoritmos de particionamento: simplificado [11], inercial ou RIB [14], espectral ou RSB [14], multinível e KL[15] e FM[16].

4.1.1. Experimentos com Chaco. O *software* Chaco foi executado 18 vezes sobre o grafo da aplicação. Em cada execução foram combinados vários parâmetros. Quando utiliza métodos multiníveis, o *software* permite que especifique em quantas partes dividir o grafo a cada estágio da decomposição recursiva.

A tab. 4 apresenta os resultados obtidos com experimentos realizados com o *software* Chaco. Essa tabela apresenta sete colunas. A primeira coluna apresenta o método global utilizado para particionar o grafo. A segunda coluna indica o método local utilizado para fazer o refinamento do grafo. A terceira coluna contêm o percentual de desbalanceamento de carga permitido durante o particionamento. A quarta coluna apresenta o autovetor utilizado durante a execução do método Espectral. A quinta coluna registra o número de reduções que o grafo sofreu durante a utilização dos métodos

locais. A sexta coluna exibe o número do corte de arestas. A sétima coluna representa o tempo gasto durante o particionamento.

Dos resultados obtidos e mostrados na fig. 7 observa-se:

- Experimentos com o método Espectral mostram que a utilização de um método local para refinamento do gráfico diminui significativamente o número de corte de arestas;
- Várias reduções podem melhorar a qualidade do particionamento. No entanto, há um certo limite, aonde não há mais melhorias no corte de arestas e tampouco no tempo de execução. Isso significa que o processo de refinamento do grafo chegou ao seu limite.

Tabela 4 – Dados referente aos particionamentos realizados com o Chaco

Método		Desb		Nº		Te
Global	**Local**	**Max**	**Av**	**Red**	**Ca**	**(s)**
Multinível	**KL**	**1%**		**5**	**79**	**0,05**
	KL	1%		10	91	0,04
	KL	1%		15	88	0,05
	KL	1%		20	84	0,05
	KL	1%		25	87	0,03
	KL	1%		30	85	0,06
Spectral	KL	1%	ML	5	83	0,10
	KL	1%	ML	10	92	0,11
	KL	1%	ML	15	83	0,10
	KL	1%	ML	20	84	0,10
	KL	1%	ML	25	93	0,11
	KL	1%	ML	30	94	0,16
		1%		5	102	0,09
		1%		10	100	0,09
		1%		15	98	0,09
		1%		20	97	0,10
		1%		25	100	0,10
		1%		30	100	0,10

O método Multinível-KL executado com 1% de desbalanceamento e apenas cinco reduções realizadas durante o refinamento do grafo, apresentou o melhor desempenho durante o particionamento (fig. 7), gerando 79 cortes de arestas. Com relação ao tempo de execução, esse mesmo algoritmo também apresentou o melhor desempenho, que pode ser observado na fig. 8.

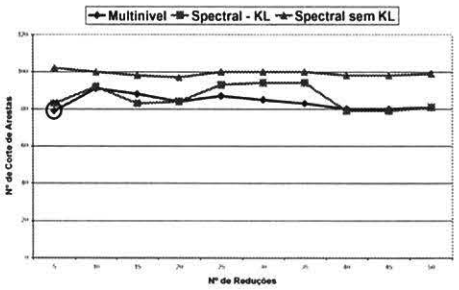

Figura 7 – Particionamento com Chaco

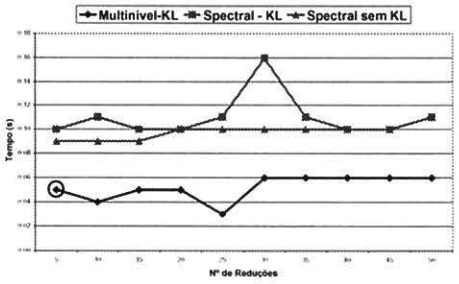

Figura 8 – Tempo de execução com Chaco

4.2 Jostle

O Jostle é um *software* para particionamento de grafos desenvolvido por Chris Walshaw na Universidade de Greenwich em Londres na Inglaterra e utiliza métodos de particionamento baseados em esquemas multiníveis descritos em [17].

O *software* Jostle utiliza o algoritmo multinível como método global e como método local permite a utilização de dois algoritmos: KL [15] ou *Greedy* [18].

4.2.1. Experimentos com Jostle. Foram realizados 14 experimentos sobre o grafo da aplicação e os melhores resultados alcançados são apresentados na tab. 5. Esta tabela descreve resultados com experimentos utilizando o método local *Greedy*.

Nessa tabela a primeira coluna apresenta o método global utilizado para efetuar o particionamento do grafo da aplicação. A segunda coluna indica o método local utilizado para refinar as partições geradas. A terceira coluna mostra o percentual de desbalanceamento informado. A quarta coluna exibe o percentual de desbalanceamento que o *software* conseguiu manter. A quinta coluna descreve o número de reduções executadas durante o refinamento. A sexta coluna exibe o número do corte de arestas. A sétima coluna finalmente apresenta o tempo de execução para cada particionamento realizado.

Os resultados apresentados na tab. 4 e exibidos na fig. 9 evidenciam que:

- o aumento do número de reduções durante o particionamento com método local KL contribuiu para a diminuição do corte de arestas. O método

Greddy conseguiu atingir um corte de arestas menor com menos reduções;

- o método Global Multinível em conjunto com o método *Greedy* executado com 3% de desbalanceamento e apenas uma redução realizada durante o refinamento do grafo, apresentou o melhor resultado durante o particionamento, gerando 81 cortes de aresta.

Tabela 5 – Dados referente aos particionamentos realizados com o Jostle

Método		Desbal.		Nº		Te
Global	Local	Max	Exec	Red	Ca	(s)
Multinível	KL	0%	0,0%	1	88	0,02
				5	87	0,02
				10	84	0,02
				15	95	0,02
				20	95	0,02
				25	95	0,02
				30	83	0,02
	Greedy	3%	2,1%	1	81	0,02
				5	83	0,02
				10	84	0,02
				15	90	0,02
				20	90	0,02
				25	90	0,02
				30	83	0,02

Considerando-se o tempo de execução, observado na fig. 10 esse *software* conseguiu manter sempre o mesmo tempo durante todos os experimentos.

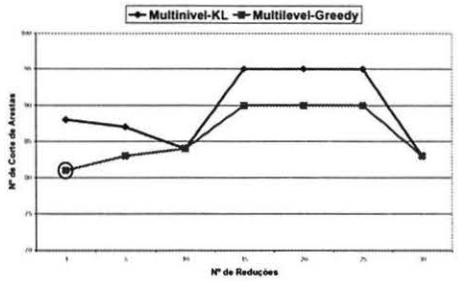

Figura 9 – Particionamento com Jostle

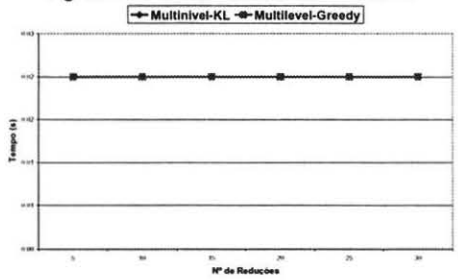

Figura 10 – Tempo de execução com Jostle

4.3 Scotch

O *software* Scotch foi desenvolvido por François Pellegrini no LABRI - Laboratoire Bordelais de Recherche en Informatique, de'lUniversité Bordeaux I, na França. O Scotch funciona um pouco diferente dos outros *software* estudados neste trabalho. Ele utiliza o algoritmo Bissecção Recursiva Dual para fazer o mapeamento e, em seguida, aplica um dos seus vários métodos de particionamento sobre a partição gerada.

4.3.1. Experimentos com Scotch. Foram realizados 18 experimentos com este *software*, combinando-se várias opções que ele oferece. Os resultados deste experimentos podem ser observados na tab. 6. Nessa tabela, a primeira coluna indica o modo como o grafo será percorrido durante o processo de particionamento e ou mapeamento. A segunda coluna mostra o método de particionamento utilizado para dividir um dos dois subdomínios gerados pelo mapeamento. A terceira coluna descreve o número de corte de arestas. A quarta coluna mostra o tempo gasto para se efetuar todo o processo.

Tabela 6 – Dados referente ao mapeamento e particionamento realizado com o Scotch

Esq. Exec	Particionamento	Ca	Te (s)
Adap	Fiduccia-Mattheyses	89	0,000006
	Gibbs-Poole-Stockmeyer	128	0,000000
	Greedy-Graph-Growing	79	0,000003
	Multilevel	598	0,000001
	Random	709	0,000001
	Exactifying	728	0,000001
Bread	Fiduccia-Mattheyses	89	0,000006
First	Gibbs-Poole-Stockmeyer	128	0,000000
	Greedy-Graph-Growing	78	0,000003
	Multilevel	598	0,000001
	Random	709	0,000001
	Exactifying	728	0,000001
Depth	Fiduccia-Mattheyses	90	0,000006
First	Gibbs-Poole-Stockmeyer	128	0,000000
	Greedy-Graph-Growing	77	0,000003
	Multilevel	598	0,000001
	Random	709	0,000001
	Exactifying	728	0,000001

Nesse sentido, este *software* apresentou o melhor resultado utilizando o algoritmo *Greedy*, alcançando 79 cortes de arestas (fig. 11).

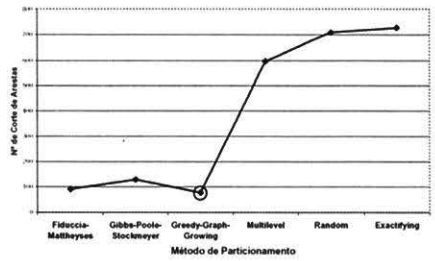

Figura 11 – Particionamento com Scotch

Este *software* apresentou tempos de execução menores que os outros *software* utilizados. Esses tempos são apresentados na fig. 12. O tempo de execução do melhor particionamento apresentou o valor de 0,000003 segundos.

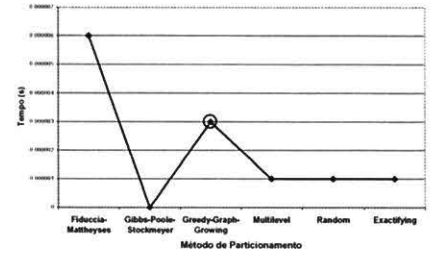

Figura 12 – Tempo de execução com Scotch

5. Comparação dos resultados

Para demonstrar a qualidade dos experimentos comparou-se os resultados obtidos com as três ferramentas. Para isso, o melhor resultado de cada experimento realizado e observados nas tabelas 4, 5 e 6, foi sintetizado na tab. 7. Nessa tabela, a primeira coluna apresenta o nome do *software* utilizado. A segunda e terceira colunas indicam o percentual de desbalanceamento máximo permitido e o percentual alcançado durante a execução do processamento, respectivamente. A quarta coluna mostra o número de cortes de arestas. A quinta coluna apresenta o tempo de execução alcançado pelo método.

Tabela 7– Melhores resultados observados entre Chaco, Jostle e Scotch

Software	Desb. Max.	Desb.	Ca	Te(s)
Chaco 2.0	1%	1,0%	79	0,050000
Jostle 2.2	3%	2,1%	81	0,020000
Scotch 3.1	1%	1,0%	77	0,000006

5.1 Qualidade do particionamento

Com relação à qualidade do particionamento, considerou-se a melhor ferramenta aquela que gerou o menor número de cortes de arestas e gastou o menor tempo durante a sua execução. Esses resultados são apresentados nas fig. 13 e fig. 14.

Entre as três ferramentas experimentadas, o Scotch revelou-se a melhor, pois conseguiu efetuar o particionamento com menor corte de arestas e em um tempo muito menor que as outras ferramentas. O Jostle apresentou um tempo de particionamento menor que o Chaco, no entanto perdeu no que se refere a qualidade do particionamento, pois não conseguiu gerar um corte de arestas menor.

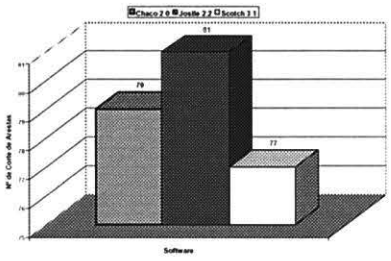

Figura 13 – Avaliação do particionamento

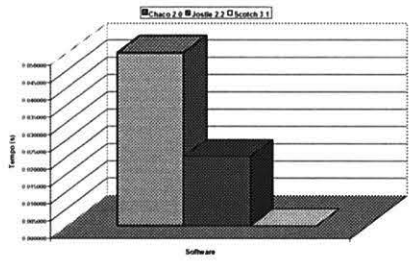

Figura 14 – Avaliação do tempo de execução

5.2 Qualidade do mapeamento

Com relação ao mapeamento do grafo da aplicação sobre o grafo da arquitetura, avaliou-se o comportamento do *software* Scotch, pois esse é o único *software* analisado que fez o mapeamento. Os resultados são apresentados na tab. 9. Essa tabela apresenta na primeira coluna o número da partição gerada, a segunda, terceira, quarta, quinta, sexta e sétima colunas indicam o número de vértices que a partição contém seguido do percentual que a mesma representa sobre o número total de vértices e a última coluna representa o poder computacional de cada nó do *cluster* modelado e apresentado na última coluna da tab. 2.

A fig. 15 ilustra os resultados obtidos e mostra a correspondência do grafo da aplicação para o grafo da arquitetura quando se trata do Scotch. Esse fator confirma a capacidade dessa ferramenta para efetuar o mapeamento de um grafo sobre outro. Com relação as ferramentas Chaco e Jostle, ambas não apresentaram capacidade de mapeamento, por outro lado demonstraram uma boa

capacidade para particionar o grafo em um ambiente homogêneo, pois conseguiram criar partições com praticamente o mesmo número de vértices (tab. 9).

Tabela 9 - Comparação de partições geradas

Partição	Chaco 2.0		Jostle 2.2		Scotch 3.1		Pc
1	47	10,02%	47	10,02%	15	3,20%	3%
2	47	10,02%	47	10,02%	19	4,05%	4%
3	47	10,02%	47	10,02%	19	4,05%	4%
4	47	10,02%	47	10,02%	19	4,05%	3%
5	47	10,02%	47	10,02%	13	2,77%	2%
6	46	9,81%	47	10,02%	10	2,13%	2%
7	47	10,02%	47	10,02%	94	20,04%	21%
8	47	10,02%	47	10,02%	94	20,04%	21%
9	47	10,02%	46	9,81%	94	20,04%	20%

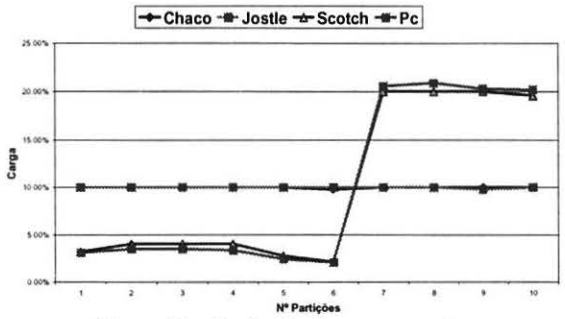

Figura 15 – Avaliação do mapeamento

6. Conclusões

Este artigo apresenta dois grafos, um representando uma arquitetura e outro representando uma aplicação. Em seguida descreve 3 ferramentas para particionamento de grafos e alguns experimentos realizados com essas ferramentas, combinando vários algoritmos, para particionar o grafo da aplicação e mapeá-lo sobre o grafo da arquitetura.

Após a realização dos experimentos, identificou-se o *software* Scotch como a melhor ferramenta, pois este apresentou os melhores resultados tanto na qualidade do particionamento quanto no tempo de execução. Isso indica que seus algoritmos foram implementados e otimizados para atingir seu potencial máximo.

Conclui-se que para se alcançar a bons resultados, torna-se necessário conhecer as características da aplicação e investigar as opções que as ferramentas de particionamento oferecem e combiná-las adequadamente.

7. Referências bibliográficas

[1]DORNELES, R. V., Particionamento de Domínio e Balanceamento Dinâmico de Carga em Arquiteturas Heterogêneas: Aplicação a Modelos Hidrodinâmicos e de Transporte de Massa 2-D e 3-D, PPGC da UFRGS, Porto Alegre, 2002.

[2] RIZZI, R. L., Modelo Computacional Paralelo para a Hidrodinâmica e para o Transporte de Massa 2-D e 3D, PPGC da UFRGS, Porto Alegre, 2002.

[3] CANAL, A. P., Paralelização de Métodos de Resolução de Sistemas Lineares Esparsos com o DECK em um Cluster de PCs, PPGC da UFRGS, Porto Alegre, 2002.

[4] ROOSTA, S., Parallel Processing and Parallel Algorithms: theory and computation, Springer, Berlin, 2000. 566p.

[5] ABURTO, A., Flops.c Version 2.0. Disponível em: <ftp://ftp.nosc.mil/pub/aburto>. Acesso em: 12 ago. 2001.

[6] McCALPIN, J. D., STREAM: Sustainable Memory Bandwidth in High Performance Computers, Dept of Computer Science. University of Virginia, Virgina. Disponível em: <www.cs.virginia.edu/stream>. Acesso em: 10 ago. 2001.

[7] Norcott, W. D., IOzone Filesystem Benchmark, Iozone Org. Disponível em: <www.iozone.org/>. Acesso em: 12 ago. 2001.

[8] Wominger, J., Pingpong - Measure effective bandwidth and latency, Lehrstuhl Fur Betriebssysteme University. Disponível em: <www.lfbs.rwth-aachen.de/~joachim/SCI-MPICH/ping pong_src.html>. Acesso em: 09 set. 2001.

[9] GROPP, W., et al. A high-performance, portable imple mentation of the MPI message passing interface standard, Parallel Computing, n.22, 1996. p.789-828.

[10] YANG, W., MAHESHWARI P., Mapping and Scheduling on Heterogeneous Systems, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, 1999. Disponível em: <www.cse.unsw.edu.au/school /people/info/weilaiy.html>. Acesso em: 13 abr. 2002.

[11] HENDRICKSON, B., LELAND, R, The Chaco User's Guide. Version 2.0. Disponível em: <www.cs.sandia.gov/CRF/ papers_chaco.html>. Acesso em: 22 mar. 2001.

[12] WALSHAW, C., The jostle user manual: Version 2.2. Disponível em: <www.gre.ac.uk/~jjg01/>. Acesso em: 14 jan. 2001.

[13] PELLEGRINI, F., SCOTCH 3.1 SCOTCH User's Guide, Disponível em: <dept-info.labri.u-bordeaux.fr/~pelegrin/scotch/ scotchgb. html>. Acesso em: 25 fev. 2001.

[14] DEMMEL, J., Applications of Parallel Computers. Berkeley, UC Berkeley, 1996. Disponível em: <www.cs. berkeley.edu/~demmel/cs267/>. Acesso em: 15 jul. 2001.

[15] KERNIGHAN, B., LIN, S., An Efficient Heuristic Procedure for Partitioning Graphs, The Bell System Technical Journal, n.49, v.2, 1970.

[16] FIDUCCIA, C.M., MATTHEYSES, R.M., A Linear Time Heuristic for Improving Network Partitions, Las Vegas, EUA, IEEE Computer Society, 1982. p.175-181.

[17] WALSHAW, C., CROSS, M., Mesh Partitioning: a Multilevel Balancing and Refinement Algorithm, London: University of Greenwich, 1998. Disponível em: <www.gre.ac.uk/jostle/>. Acesso em: 30 mar. 2001.

[18] SZEWCZYK, R, FERENCZ, A., WEINSTEIN, J., WILKENING, J., Graph Bisection, Disponível em: <www.cs.berkeley.edu/ szewczyk/CS270/>. Acesso em: 30 jum. 2001.