FICE 5ª FEIRA DE INICIAÇÃO CIENTÍFICA E EXTENSÃO 15 e 16 de Setembro

DESENVOLVIMENTO DE AMBIENTES COMPARATIVOS ENTRE TÉCNICAS DE CLUSTERIZAÇÃO E PROCESSAMENTO EM GRAPHIC PROCESSOR UNIT

Amanda Serafini[1]; Ricardo Kohler [2]; Diego Ricardo Krohl[3] ; Angelita Rettore Araujo Zanella[4]; Wagner Carlos Mariani[5]

INTRODUÇÃO

Este projeto apresenta um referencial comparativo de desempenho entre aglomerados de computadores (clusters) e processamento em GPU (Graphic Processor Unit). O propósito desta comparação é decidir em qual destes ambientes obtém-se um melhor desempenho ao calcular Desigualdades Matriciais Lineares (LMIs).

Largamente utilizadas na área da teoria do controle, as Desigualdades Matriciais Lineares são problemas de otimização onde se deseja minimizar ou maximizar uma função linear com restrições também lineares. Sob este aspecto, consideram-se as Desigualdades Matriciais, problemas de programação linear (PPL). (DANTZIG, 1951).

As matrizes a serem exploradas, com este intuito, podem ser de diversos tamanhos, e do ponto de vista matemático podem ter dimensões onde mesmo sendo processadas em computadores modernos, levem um tempo considerado inaceitável.

Normalmente os problemas de desigualdades matriciais são apresentados nos trazendo uma ou mais matrizes quadradas dadas, uma operação matemática que envolve estas matrizes, uma matriz a ser descoberta e finalmente o resultado esperado no final desta operação. De acordo com o problema pode-se ter como

[1] Aluna do Instituto Federal Catarinense, Videira. Curso de Ciência da Computação. E-mail: amandaserafini1@hotmail.com
[2] Técnico Administrativo no Instituto Federal Catarinense, Videira. E-mail: ricardo.kohler@ifc-videira.edu.br
[3] Professor do Instituto Federal Catarinense, Videira, Curso de Ciência da Computação. E-mail: diego.krohl@ifc-videira.edu.br
[4] Professora do Instituto Federal Catarinense, Videira, Curso de Ciência da Computação. E-mail: angelita@ifc-videira.edu.br
[5] Professor Orientador do Instituto Federal Catarinense, Videira. Curso de Ciência da Computação. E-mail: wagner.mariani@ifc-videira.edu.br

objetivo encontrar os menores valores possíveis na matriz a ser descoberta, ou então descobrir os maiores valores possíveis nesta mesma matriz, seguindo a regra:

$$\begin{aligned} &\mathbf{A}^t.\mathbf{P} + \mathbf{P}.\mathbf{A} < 0 \\ &\mathbf{P} > 0 \\ &\min \mathbf{P} \end{aligned} \qquad (1)$$

Um exemplo de uma LMI pode ser visto na Equação (1). Onde dada a Matriz quadrada **A**, desejamos encontrar menor matriz quadrada **P**, tal que o resultado da equação seja menor que zero (determinante negativo) e a matriz **P** seja maior que 0 (determinante positivo).

Assim como outras LMIs este problema pode ser atacado sob dois aspectos, sendo estes; a otimização do software que busca a resolução do problema, e o desempenho do hardware onde este software será executado.

Quanto a otimização de software uma importante ferramenta é a da programação concorrente, onde o programa é executado simultaneamente por diversos processos que cooperam entre si.

A programação concorrente é mais complexa que a programação sequencial, mas mesmo com todas as suas complexidades, em sistemas nos quais existem vários processadores é possível aproveitar este paralelismo explicitamente e acelerar a execução do programa. (OLIVEIRA, 2004).

Uma maneira de estender a programação concorrente, com o objetivo de maximizar o desempenho geral na execução de um programa é agregar diversos equipamentos por intermédio de redes de computadores, formando clusters. Gerando assim uma nova categoria de estudos da computação denominada computação distribuída de alto desempenho. (DANTAS, 2005).

Motivado pela alta demanda computacional exigida por aplicações gráficas, nos anos de 1980 surgiram os aceleradores gráficos não programáveis, que em meados da mesma década passaram a integrar todos os elementos essenciais em um único chip permitindo assim a computação em unidades de processamento gráfico, GPUs, que são compostas nativamente de múltiplos núcleos, não tão

otimizados como núcleos de CPU, mas que pela grande quantidade, pode torná-las interessantes para o processamento matemático paralelizado. (NVIDIA, 2016).

Porém, os computadores, comportam na maioria dos casos no máximo 2 GPUs enquanto o Cluster possui poucos limites de expansão.

Neste trabalho ficou evidente a importância de uma única memória RAM compartilhada. O cluster teve dificuldades de desempenho até mesmo para superar o desempenho de um único computador usando apenas seu CPU local, pois utilizando um único computador, o processamento é realizado, com uma única cópia das matrizes envolvidas sendo manipuladas por ponteiros nas diversas Threads processadas pelo GPU, ou pelo processador principal (CPU).

O Cluster de alto desempenho, no entanto, precisa primeiro transportar pela rede para cada computador remoto, também chamado de nó, uma cópia destas matrizes, ou ao menos trechos delas. O tempo de transferência acaba sendo comparável ao tempo de processamento local em si. Ao aumentarmos o número de computadores de Cluster além de aumentarmos o desempenho, aumentamos também a necessidade de transferência de dados, esta transferência acaba por reduzir em muito o ganho ao agregar nós. O Processamento em GPU no entanto, independe de transferência de dados por rede, tornando-se mais indicado para problemas que envolvem manipulação intensa de matrizes.

PROCEDIMENTOS METODOLÓGICOS

O desenvolvimento do projeto foi realizado no período entre Julho de 2015 e Junho de 2016, no laboratório de informática no Campus de Videira, do Instituto Federal Catarinense.

Este laboratório possui computadores equipados com processadores AMD Phenom X4B95 com quatro núcleos de 3 GHz e 4GB de memória RAM, e também uma placa de vídeo com GPU modelo AMD Radeon 4550 de 80 núcleos.

O Cluster deste laboratório foi montado fazendo uso de uma rede ethernet de 100 mbps usada exclusivamente para este fim, com o objetivo de evitar interferências de outros fluxos de dados, senão os envolvidos na operação dos testes. Estes testes envolveram até 8 computadores, portanto 32 núcleos.

Para a criação do Cluster de alto desempenho, fez-se o uso do sistema operacional Linux Pelican HPC, versão 3.2.0, uma distribuição Linux, gratuita e livre que que permite a configuração fácil de um cluster, bem como a utilização da biblioteca Open MPI (Message Passing Interface).

De acordo com (CREEL 2015) uma das vantagens do uso do Pelican HPC é ser suportado em uma mídia ISO híbrida, que pode ser inicializada a partir de um Live-CD ou uma unidade USB, facilitando a criação de um cluster, tornando o processo de configuração transparente e rápido para o usuário.

Outra vantagem é que esta mesma distribuição do sistema operacional, pode ser usada para suportar um único computador, e realizar os testes de desempenho em processamento local. O Aplicativo matemático Octave já está presente, o que facilita o uso de scripts pré-existentes para a solução de LMI locais em GPU.

A biblioteca Open MPI permite ao programador, ao desenvolver o aplicativo, enviar instruções e valores para que sejam processados em um núcleo, inclusive remoto, e receber posteriormente os resultados. Os aplicativos a serem executados, precisam ser adaptados usando esta biblioteca, informando inclusive a quantidade de nós do cluster.

O resultado do cálculo de LMIs depende, com muita frequência da multiplicação de matrizes, representado abaixo:

Figura 1 – Multiplicação de matrizes

$$\begin{pmatrix} \boxed{2 \quad 3 \quad \cdots \quad 8} \\ a_{21}\ a_{22} \cdots a_{2j} \\ \vdots \quad \vdots \quad \quad \vdots \\ a_{i1}\ a_{i2} \cdots a_{ij} \end{pmatrix} \cdot \begin{pmatrix} \boxed{5} & a_{12} \cdots a_{1j} \\ \boxed{6} & a_{22} \cdots a_{2j} \\ \boxed{\vdots} & \vdots \quad \quad \vdots \\ \boxed{1} & a_{i2} \cdots a_{ij} \end{pmatrix} = \begin{pmatrix} 2.5+3.6+\ldots+8.1 & \\ & \end{pmatrix}$$

Fonte: autor (2016)

É possível subdividir os cálculos entre os diversos núcleos enviando trechos para que sejam calculados de maneira independente. Na Figura 1, podemos observar as operações matemáticas necessárias, destacadas por um quadro, para obter o valor resultante de uma multiplicação em uma posição de uma matriz. Se

desejarmos, podemos enviar os valores destacados dentro de um quadro para serem multiplicados e somados em núcleos separados e paralelamente.

No processamento local, podemos solicitar, no que cada um dos núcleos do CPU ou que os núcleos do GPU executem em paralelo o cálculo de cada uma das posições da matriz resultante. Todos os núcleos irão então obter os dados das mesmas matrizes, que já estão em memória RAM, em uma única cópia.

Se desejarmos, no entanto, processar em núcleos que se encontram em outro computador, os dados precisam ser enviados pela rede para o computador remoto. Ainda no exemplo da Figura 1, os valores destacados pelos quadros devem ser enviados pela rede para o nó que realizará o cálculo. O mesmo acontece para qualquer outra posição a ser calculada remotamente.

Desconsiderando a capacidade de processamento e sim a capacidade de transferência de dados, dada a configuração de nosso laboratório quando consideramos a tecnologia Hypertransport AMD 3.0, que realiza conexões entre dispositivos, ampliando consideravelmente velocidade de clock e largura de banda. (HYPERTRANSPORT, 2016) e que está presente nestes computadores, é capaz de transferir dados a 41.600 MB/s (megabytes por segundo), entre a memória RAM e o processador, enquanto a tecnologia de rede fast ethernet transfere os dados, de acordo com (IEEE STANDARD 2016), a uma taxa de 100Mb/s (megabits por segundo), entre os diferentes nós do cluster, e considerando ainda que 1 byte possui 8 bits a rede de computador é 3.328 vezes mais lenta que o barramento hypertransport que é usado para transferir os dados internamente em um único computador, sem levar em conta que tanto o protocolo TCP/IP quanto o protocolo Ethernet, possuem dados de controle (payload).

Portanto o tempo apenas para a transferência dos dados para um processador remoto, normalmente é superior ao tempo de processamento local, e estes valores acompanham em proporção tanto os nós de um cluster quanto o tamanho das matrizes envolvidas. Esta característica, não é típica apenas dos computadores envolvidos no experimento. O alto desempenho na transferência de dados entre a memória RAM e os processadores, é um aspecto muito valorizado no projeto de todos os computadores modernos, considerando que a velocidade do

processador cresce bem mais significativamente do que dos dispositivos de armazenamento. (MCCALPIN, 1995).

Na tentativa de diminuir esta problemática, na programação para o cluster, visando diminuir o tempo gasto com transferência de dados pela rede, a estratégia de subdividir a matriz foi abandonada, e enviamos a matriz inteira para cada nó remoto, e posteriormente executamos todo o algoritmo que descobre um valor que satisfaça os requisitos do LMI em cada nó. O primeiro nó do cluster a descobrir uma solução válida cancela o processamento nos demais.

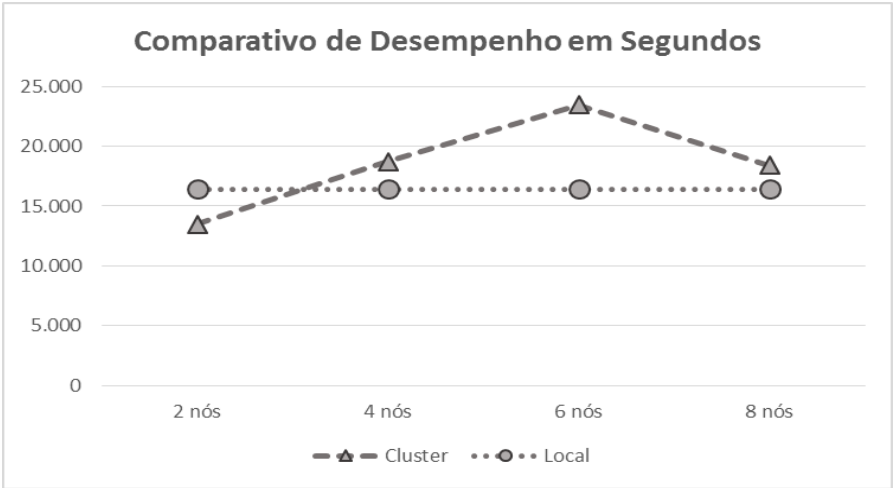
Como o tempo de transferência é o maior gargalo no cluster, esta estratégia maximiza o desempenho do Cluster para solucionar LMIs, enviando uma única vez uma cópia de todos os dados a serem trabalhados. No entanto, como no processamento local, não temos de transferir nada pela rede, podemos subdividir o cálculo, principalmente da multiplicação de matrizes entre diversos núcleos.

Para que pudéssemos comparar o mesmo algoritmo em Cluster e localmente adotamos a mesma abordagem no processamento local. Esta decisão, no entanto, subutiliza as características da GPU.

RESULTADOS E DISCUSSÕES

As experiências realizadas nos laboratórios, utilizaram a LMI já expressada na Equação 1. A Matriz **A** dada para o problema, era de ordem 10. Espera-se da matriz **P** uma solução para a LMI de ordem 10 com o menor valor possível.

Gráfico 1 – Resultados dos testes



O experimento foi repetido 10 vezes para cada nó, gráfico acima mostra o tempo médio para encontrar a solução, desprezando 10% dos piores e melhores resultados. Como pode-se observar o melhor desempenho do Cluster foi com 2 nós, conseguido superar o processamento local em CPU. Ao acrescentarmos mais 2, ficando com 4 nós, e posteriormente com 6 nós, o desempenho cai, devido a maior carga de transferências pela rede, conseguindo recuperar-se apenas parcialmente com 8 nós.

Apenas no melhor caso, com 2 nós o cluster conseguiu calcular a LMI em menos tempo que apenas um computador local, que, para efeitos comparativos evitou aproveitar a possibilidade de dividir a multiplicação de matrizes. Ou seja, o cluster com 2 nós só consegue superar um único computador pois este último não tira proveito do uso massivo de múltiplos núcleos da GPU, que como citado acima, foi subutilizado para que as propriedades do algoritmo não interferissem no comparativo.

CONSIDERAÇÕES FINAIS

Ao observarmos o gráfico, fica claro, que o cluster de alto desempenho enfrenta dificuldades, quanto a transferência de dados entre os nós que o compõe.

Em nosso experimento, claramente, o cluster é inferior ao processamento local, seja por GPU ou mesmo por CPU. Sendo, portanto, desaconselhado.

Isto é devido pois enquanto o processamento em múltiplos núcleos locais, beneficia-se dos barramentos internos, tais como o hypertransport, que, justamente por trafegar dados a curta distância somente dentro dos próprios circuitos do computador, pode atingir velocidades altíssimas, quando comparados a transferência uma rede local convencional.

O problema do desempenho na transferência de dados pela rede, poderia ser minimizado, dentro das redes ethernet, poderíamos citar o uso de equipamentos que suportem a união de canais tais como o etherchannel, que consiste em vários canais físicos unidos a uma única porta lógica, buscando aumento de banda. (CISCO, 2010).

Ou ainda, pode-se substituir o protocolo ethernet por outras tecnologias de transferência, tais como Myrinet, uma rede baseada na ideia de processamento paralelo (FELDERMAN, 1995) e Infiniband, um padrão desenvolvido pela Trade InfiniBandSM Association, que propicia melhor desempenho em vários aspectos (PFISTER, 2001). Ambas são mais rápidas que a ethernet.

Os equipamentos de rede próprios para estas tecnologias, no entanto, são mais caros que os equipamentos para ethernet.

Este custo pode justificar a adoção do processamento em GPUs, que vem tornando-se progressivamente mais populares e baratas.

O Cluster ainda pode ser indicado, para situações onde muitas operações devem ser realizadas com o mesmo conjunto de dados. A grande capacidade computacional atual está sendo buscada cada vez mais as mais diversas áreas, procurando resolver problemas relacionados a astrofísica, modelos de clima, e simulações biológicas por exemplo, que podem beneficiar-se desta tecnologia. (DEL ROSARIO, 1994).

Enfim, quando nos deparamos com equações muito grandes, e que usam matrizes, podem beneficiar-se de clusters, que processam em GPUs em seus nós.

O caso de LMIs as equações podem envolver matrizes de ordem muito grande, porém, a equação em si é relativamente pequena. Estas características do problema, tornam um típico caso onde os benefícios do processamento em GPU superam o cluster.

REFERÊNCIAS

CISCO. **Catalyst 2960 and 2960-S Software Configuration Guide, 12.2(53)SE1 - Configuring EtherChannels and Link-State Tracking [Cisco Catalyst 2960 Series Switches]**. Disponível em: <http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_53_se/configuration/guide/2960scg/swethchl.html#wp1275535>. Acesso em: 13 jul. 2016.

CREEL, M. **Pelicanhpc.** Grecs Computer Code, Research Group in Computation and Simulations (GRECS), p. 24, 42, 2015.

DANTZIG, G. B. **Maximization of a linear function of variables subject to linear inequalities.** Activity Analysis of Production and Allocation, (Ed.). KOOPMANS, T. C. New York, John Wiley, p. 339-347, 1951.

DANTAS, Mario. **Computação distribuída de alto desempenho: redes clusters e grids computacionais**. Rio de Janeiro: Axel Books do Brasil, p. 278, 2005.

DEL ROSARIO, Juan Miguel, and Alok N. Choudhary**. "High-performance I/O for massively parallel computers: Problems and prospects."** *Computer* 27.3 (1994): 59-68.

FELDERMAN, Robert E., et al. **"Myrinet: A gigabit-per-second local area network."** *IEEE micro.* February (1995): 29-36.

HYPERTRANSPORT. **HyperTransport Specifications.** Disponível em: <http://www.hypertransport.org/default.cfm?page=HyperTransportSpecifications>. Acesso em: 13 jul. 2016.

MCCALPIN. "**Sustainable memory bandwidth in current high performance computers"**, Technical Report, Silicon Graphics, October 1995.

IEEE STANDARDS. **Medium Access Control (MAC) parameters, Physical layer, Medium Attachment Units, and Repeaters for 100 Mb/s Operation, Type 100Base-T** (1995).

NVIDIA. **O QUE é computação com GPU.** 2015. Disponível em: <http://www.nvidia.com.br/object/what-is-gpu-computing-br.html>. Acesso em: 21 abr. 2016.

OLIVEIRA, R.; CARISSIMI, A.; TOSCANI, S. **Sistemas operacionais**. 3. ed. Série Didática do II-UFRGS, p. 27-57 2004

PFISTER, Gregory F. **"An introduction to the infiniband architecture."** *High Performance Mass Storage and Parallel I/O* 42 (2001): 617-632.