USO DA COMPUTAÇÃO PARALELA PARA MELHOR DESEMEPENHO DE PROCESSAMENTO

Carlos Alberto Barbosa Stenzel
carlosstenzel@hotmail.com

RESUMO

A computação paralela está presente em muitos locais, assim como bancos, onde o usuário final quer ter uma resposta rápida a ação executada, onde a não utilização da computação paralela ocasionaria em um maior tempo de resposta do comando executado. O uso da computação paralela permite que vários processos sejam executados ao mesmo tempo em paralelo, assim obtendo uma resposta mais rápida das funções executadas e sendo possível a utilização de Clusters e Grids podendo suportar ainda mais processos executando ao mesmo tempo, assim tendo um melhor proveito do poder do computador.

Para que seja possível o uso dessa tecnologia é necessário o uso da programação paralela, existem diversas formas para utilizar, como a utilização de Forks, MPI, OpenMP, CPAR, entre muitos outros existes para facilitar a utilização da computação paralela.

1 - INTRODUÇÃO

Desde o surgimento do primeiro computador digital eletrônico, ENIAC (*Electronic Numerical Integrator and Computer*), em 1946, a computação vem passando por um processo evolutivo intenso, tanto em nível de hardware quanto de software, sempre visando proporcionar um maior desempenho e tornar mais rápido o processante de funções.

Na década de 90, redes de computadores tornaram-se mais rápidas e confiáveis, o que possibilitou a interligação de computadores de maneiras mais eficientes formando os sistemas distribuídos.

Sistemas distribuídos têm sido utilizados para a execução de programas paralelos, em substituição às arquiteturas paralelas, em virtude de seu menor custo e maior flexibilidade.

Atualmente a computação paralela está presente em muitas coisas que usamos, pode-se não perceber o uso, mais está presente, por exemplo, nos caixas eletrônicos dos bancos, nos aplicativos

que são usados nos celulares ou computadores.

A computação paralela é utilizada para melhor desempenho em servidores para que o usuário final tenha uma resposta rápida ação solicitada, como por exemplo, retirar um extrato da sua conta em um caixa eletrônico, supondo que não fosse utilizado a computação paralela, então logo seus servidores não poderiam estar ligados em *cluster*, ou utilizando chips com múltiplos núcleos por exemplo que já estes mesmos são um exemplo do uso da computação paralela, então um servidor só pode executar uma função de cada vez, assim, quando um cliente solicitar um extrato, este pedido estará em uma fila para execução, então o usuário final, que nesse caso é um cliente de um banco que está tentando retirar seu extrato tem que esperar que todos os outros pedidos solicitados outros clientes que estejam na frente do seu pedido na fila de execução, sejam executados para que enfim o seu pedido de extrato solicitado seja processado e executado.

1.1 – Objetivos

1.1.1 – Objetivo geral

Mostrar como a utilização da computação paralela pode melhorar o desempenho de processamento em sistemas de grande porte que necessitem de rapidez e agilidade no processamento dos dados e funções.

1.1.2 – Objetivos específicos

- Apresentar os conceitos e o funcionamento da computação paralela.

- Mostrar de que forma paralelismo pode melhorar o tempo o desempenho de sistemas.

- Mostrar algumas técnicas utilizadas para a obtenção de maior desempenho.

- Efetuar uma comparação de alguns métodos utilizados na programação da computação paralela.

2.2 – Processadores Multi-core

A solução para o problema de aquecimento que estavam enfrentando com os processadores com alto *clock*, foi o uso de dois ou mais núcleos, está tecnologia é conhecida como *multi-core*, segundo ANDRÁS (2011, p.3) “[...] múltiplos núcleo ou *multi-core* são usados para descrever arquiteturas com um número elevado de núcleos.”, com essa tecnologia o processador reparte as tarefas entre os núcleos, o que faz que aumente a velocidade de processamento, segundo PERUZZO (2008, p.119) “isso possibilita que as instruções das aplicações sejam executadas em paralelo, numa

menor frequência e produzindo menos calor.”

Atualmente processares com arquitetura *multi-core* são encontrados na maioria dos computadores, celulares comercializados.

3 – Computação Distribuída

A necessidade de criar computadores que precisavam cada vez mais poder de processamento, fez surgiu a computação distribuída, a computação distribuída segundo COULOURIS (2011, p. 01) “é uma coleção de computadores autônomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dado”.

Um problema é divido em várias tarefas, cada um é resolvido por um ou mais computadores, que estão conectados formando um sistema distribuído.

4 – Cluster

Cluster pode ser definido segundo STALLINGS (2002, p.15) “como um grupo de computadores completos interconectados”, essas máquinas que estão conectada por cabos de rede compartilham recursos, segundo STALLINGS (2002, p.15) “como um recurso de computação unificado, que cria a ilusão de construir uma única máquina”.

Os *clusters* podem ser classificados segundo BUYYA (1999) como:

Balanceamento de Caga: Pode ser visto como uma solução abrangente na utilização de grandes redes, porque ele ocasiona um aumento na capacidade, melhorando assim o desempenho, já que o balanceamento será feito por vários servidores, todos os servidores mantém uma cópia integral de todos os dados onde os servidores precisarão dessa cópia para ler as informações que chegarem até ele, um software de controle se encarrega de sincronizar todas as informações, caso haja falha em algum servidor, ou algum precise ser desligado, os demais continuam trabalhando normalmente, ao voltar, o software de controle sincroniza o servidor com os demais para voltar os serviços normalmente.

Para MARCOS PITANGA (2003) “este modelo distribui o tráfego entrante ou requisições de recursos provenientes dos nodos que executam os mesmos programas entre as máquinas que compõem o *cluster*”.Todos os nodos são responsáveis em controlar os pedidos. Se um nó falhar, as requisições são distribuídas entre os nós disponíveis no momento. Este tipo de solução é normalmente utilizado em fazendas de servidores web (*web farms*).

Alta Disponibilidade: Sua principal função é manter os serviços ativos pelo maior tempo possível, onde é possível sua aplicação através de redundância de hardware e configuração de softwares específicos. Nesse caso os serviços não pertencem apenas a um computador, mas sim ao *cluster* como um todo. A desvantagem desse tipo de *cluster* é que caso um serviço venha a falhar, pode-se perder um pouco de tempo, pois seu principal objetivo é manter os serviços ativos. Diferente do balanceamento de carga que divide o processamento com os demais computadores interligados.

Para MARCOS PITANGA (2003) "estes modelos de *cluster* são construídos para prover disponibilidade de serviços e recursos de forma ininterrupta através do uso de redundâncias implícitas ao sistema. A ideia geral é que se um nó falhar (*failover*), aplicações ou serviços possam estar disponíveis em outro nó. Estes tipos de *cluster* são utilizados para base de dados de missões críticas, correios, servidores de arquivos e aplicações."

Alta Performance de Computação: Conhecido também por Cluster de Alto Desempenho, o objetivo desde tipo de *cluster* é desenvolver supercomputadores para processamento paralelo, trabalhando em conjunto de forma integrada, trazendo resultados satisfatórios, com serviços rápidos e confiáveis.

Para MARCOS PITANGA (2003) este modelo de *cluster* aumenta a disponibilidade e performance para as aplicações, particularmente as grandes tarefas computacionais. Uma grande tarefa computacional pode ser dividida em pequenas tarefas que são distribuídas ao redor das estações (nodos), como se fosse um supercomputador massivamente paralelo. Estes *clusters* são usados para computação científica ou análises financeiras, tarefas típicas para exigência de alto poder de processamento.

Cluster Combo ou Combinação: Combina as características dos *clusters* de alta disponibilidade e de balanceamento de carga, aumentando assim a disponibilidade e escalabilidade de serviços e recursos, Esse tipo de configuração de *cluster* é bastante utilizado em servidores de web, e-mail, entre muitos outros.

Cluster de Processamento Distribuído ou Processamento Paralelo: Este modelo de *cluster* aumenta a disponibilidade e o desempenho para as aplicações, particularmente as grandes tarefas computacionais. Uma grande tarefa computacional pode ser dividida em pequenas tarefas que são distribuídas as estações como se fossem um supercomputador massivamente paralelo. É comum associar este tipo de *cluster* ao projeto *Beowulf*.

5 – Computação Paralela e de alto desempenho

Segundo STALLINGS (2002) os primeiros supercomputadores surgiram nos anos 70. E até aquele momento segundo TANENBAUM (2007) as arquiteturas eram simples e o aumento da eficiência computacional estava limitado pelo desenvolvimento tecnológico, principalmente pelo fato dos processadores terem que terminar uma tarefa para iniciar outra.

Com esses fatos, pode-se perceber que a divisão de tarefas traria avanço significativos para os computadores dessa época quanto a questão de desempenho computacional. A partir dessa época surgiu dois caminhos seguidos para solucionar a divisão das tarefas, a arquitetura paralela e sistema distribuído.

Uma das grandes áreas que se dedica a propor tais melhorias é a computação paralela e de alto desempenho – HPC (*High Performance Computing*), que tem como base, várias subáreas da Ciência da Computação.

Não só no contexto científico é utilizado a computação de alto desempenho, pode-se observar que a uma explosão no volume de dados gerado a cada minuto.

Os *clusters* estão sendo usados em uma grande quantidade de aplicações, especialmente aquelas que exigem alto poder de processamento. Sua utilização inclui ainda qualquer tipo de aplicação crítica, ou seja, aplicações que não podem parar de funcionar ou não podem perder dados, como os sistemas de bancos, por exemplo.

3.1 - Métricas em Computação Paralela

Algumas métricas são utilizadas para medir o desempenho do uso da computação paralela em alguns sistemas, podemos dividi-las em 4 tipos:

- Tempo de execução
- Fator de aceleração ou *speedup*
- Eficiência
- Custo

6 – Utilização de Clusters e Grades na Computação Paralela

6.1 - Clusters

Segundo BUYYA (1999) *cluster* consiste de um conjunto de computadores interconectados trabalhando em conjunto para executar aplicações e integrando recursos computacionais. Seu objetivo é fazer com que todo o processamento da aplicação seja distribuído aos computadores, de forma que pareça um único computador.

Com isso, é possível realizar processamentos que até então somente computadores de alto desempenho seriam capazes de fazer. As características obtidas com a implementação de *clusters* são o aumento da confiabilidade, distribuição de carga e desempenho.

6.2 – Grades

A grade computacional segundo FOSTER (1999) é uma infraestrutura de gerenciamento de dados que fornece recursos eletrônicos para uma sociedade global em negócios, administração, pesquisa, ciência e entretenimento sem levar em conta a localização geográfica.

As grades segundo BERMAN (2003) integram redes, comunicação, processamento e informação com o objetivo de prover uma plataforma virtual para processamento e gerenciamento de dados da mesma forma que a Internet integra recursos para formar uma plataforma virtual para obter informações. As grades de larga escala são intrinsecamente distribuídas, heterogêneos e dinâmicos, eles têm transformado a ciência, as empresas, a saúde e a sociedade.

A essência da grade computacional pode ser resumida como Recursos sob-Demanda (RoD – *Resources on Demand*), ou seja, segundo YANG (2003) o fornecimento transparente de recursos da grade necessários as aplicações ou serviços que necessita de pouco ou nenhum atraso. A má qualidade de serviços na rede pode prejudicar significativamente o fornecimento eficiente de RoD.

7 – MPI

Usado para a troca de mensagens entre os nós do *cluster*, sendo mais avançada que a PVM (*Parallel Virtual Machine*), pois pode trabalhar com mensagens para todos os computadores ou para apenas um determinado grupo.

Segundo OTTO (1998) a Interface de Passagem de Mensagem (MPI), é um paradigma de programação amplamente usado em computadores paralelos, especialmente nos Computadores Paralelos Escalares (*Scalable Parallel Computers* – SPC) com memória distribuída e nas Redes de Estações de Trabalho (*Networks of Workstations* – NOW).

Segundo HUSS-LEDERMAN (1998) a MPI é um sistema padronizado portável de passagem de mensagem projetado por um grupo de pesquisadores da academia e da indústria para funcionar em computadores paralelos. O padrão define a sintaxe e a semântica de um conjunto de rotinas úteis para uma grande quantidade de usuários que desenvolvem programas portáteis de passagem de mensagem nas linguagens de programação Fortran, C ou C++.

A primeira versão do padrão, chamado de MPI-1, segundo OTTO (1998) foi liberada em 1994 e desde então a especificação MPI tornou-se o padrão de passagem de mensagem para computadores paralelos. A segunda versão do padrão, a MPI-2, segundo HUSS-LEDERMAN

(1998) foi disponibilizada em 1997.

No padrão MPI, uma aplicação é constituída por um ou mias processos que se comunicam, adicionando-se funções para envio e recebimento de mensagens entre os processos. Pode se observar na imagem abaixo a estrutura de um programa MPI, escrito na linguagem de programação C.

8 – Programação Paralela

A programação paralela é uma forma de computação em que o processamento de um algoritmo é divido em partes e executado ao mesmo mesmo tempo, de forma a aumentar o desempenho de processo e deixando mais rápido a execução de softwares.

O OpenMP (*Open Multi-Processing*) segundo PITANGA(2008) é uma API (Interface de Programação de Aplicações) que visa o desenvolvimento de aplicações *multitheads* de uma forma mais fácil em ambientes de programação C/C++ e Fortran. Ainda segundo PINTANGA (2008) é composto por diretivas de compilação de bibliotecas para programação *multitheads*, suportando o modelo de paralelismo dos dados, permitindo o paralelismo incremental e combina partes de código escrito na forma serial e paralela em um único código fonte.

O CPAR (Linguagem de programação paralela) foi criado pela Professora Doutora Liria Matsumoto Sato, e segundo SATO (1995) CPAR é uma extensão paralela da linguagem C que utiliza alguns elementos das linguagens C e Ada, tal como um modelo de programação de multitarefas. Ainda segundo SATO (1995) ela foi projetada visando oferecer construções simples para a exploração do paralelismo em múltiplos nives, permitindo uma melhor utilização da localidade de memória.

O CPAR oferece variáveis compartilhadas para o compartilhamento das informações entre os processos. A comunicação entre diversos elementos de processamento ocorre através das variáveis compartilhadas.

9 – CONCLUSÃO

Um dos objetivos é apresentar os conceitos e funcionamento da computação paralela, o que demonstrou que é atualmente essencial para sistemas computacionais, onde o tempo otimizado de uma resposta de processamento torna-se cada vez mais uma exigência. Apesar da grande evolução do hardware que permite um melhor desempenho do processamento e de baixo custo, se for utilizado de forma errada, apesar do alto potencial do hardware, pode acabar por obter um tempo maior de resposta, e a utilização errada do paralelismo no sistema também pode ter um desempenho prejudicado.

A computação paralela tem como objetivo dividir os processos para que seja possível uma redução do tempo de processamento assim como o esforço da infraestrutura. Atualmente ficou mais fácil essa implantação em diversos nível de sistemas, pois com a evolução do hardware, pode-se obter a utilização dos processadores multicores, assim tornando a utilização do paralelismo essencial para um melhor aproveitamento da infraestrutura, assim como a utilização de *clusters* e *grids*.

A utilização de *cluster* e *grids* ajudam e muito a obtenção de um melhor desempenho, onde o MPI se destaca quando utilizada dessas tecnologias que interligam computadores, para poder dividir seus processos entre eles.

Para realizar a aplicação em qualquer sistema deve-se ser feito uma análise para que assim saiba qual é a melhor opção para o sistema e o ambiente usado, para que assim possa-se obter um resultado melhor, alcançando o objetivo de diminuir o tempo de resposta da aplicação proposto pelo paralelismo.

Assim pode-se concluir que, o uso da computação paralela em diversos áreas, independente do tamanho do sistema e do uso, pode melhorar o desempenho do processamento, desde que seja feito um estudo para descobrir qual opção apresenta um melhor desempenho para o ambiente utilizado.

Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair. Distributed Systems: Concepts and Design (5th Edition), 2011.

András Vajda. Programming Many-Core Chips. Springer Science & Business Media, 2011.

STALLINGS W. Arquitetura e Organização de Computadores: projeto para o desempenho. 5. ed. Prentice Hall, 2002.

PITANGA, M. Computação em Cluste. Acessado em 06 de maio de 2015 – disponível através do endereço http://www.clubedohardware.com.br/artigos/153.

TANENBAUM, A. S. Organização Estruturada de Computadores. 5. ed. Prentice Hall, 2007.

PITANGA, M. - Construindo Supercomputadores com Linux. Brasport. 2008

BUYYA, R. High Performance Cluster Computing: architectures and systems, Volume 1. Prentice Hall, 1999.

FOSTER, I. KESSELMAN, C. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.

BERMAN, F., FOX, G. C., HEY, A. J. G. The open grid services architecture and data grids. Grid Computing: Making The Global Infrastructure a Reality. JohnWiley& Sons : Inglaterra. 2003.

YANG, K.; GUO, X.; GALIS, A. et al. Towards efficient resource on-demand in Grid Computing. ACM SIGOPS Operating Systems Review. v. 37, n. 2, p. 37-43, Apr. 2003.

SNIR, M., OTTO, S., HUSS-LEDERMAN, S. et al. MPI – The complete reference, Volume 1. 2. ed. The MIT Press, 1998.

SATO, L. M. Ambientes de programação para sistemas paralelos e distribuídos. Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, 1995.

SNIR, M., OTTO, S., HUSS-LEDERMAN, S. et al. MPI – The complete reference, Volume 1. 2. ed. The MIT Press, 1998.

GROPP, W., HUSS-LEDERMAN, S., LUMSDAINE, A., et al. MPI – The complete reference. Volume 2. 2. ed. The MIT Press, 1998.