

POR QUE ESCREVER PROGRAMAS PARALELOS?

ALEXANDRE DA COSTA SENA*

RESUMO

O objetivo deste artigo é mostrar que, apesar de todas as dificuldades de se implementar e executar programas paralelos, eles são essenciais para as pesquisas atuais, principalmente nas áreas de computação, física, biologia e engenharia.

Palavras-chaves: Paralelismo; Programas Seqüenciais; Grandes Desafios.

ABSTRACT

The main goal of this article is to show that despite all difficulties in implementing and executing parallel programs, they are essential to the current researches, mainly in the computation, physic, biology and engineering areas.

Keywords : Parallelism; Sequential Programs; Great Challenges.

* Mestre em Ciência da Computação pela UFF. Professor da UNILASALLE (Niterói/ RJ).

INTRODUÇÃO

Escrever um programa paralelo é mais difícil e mais trabalhoso para se testar. Além disso, nem sempre um programa paralelo é mais eficiente que um programa seqüencial. Então, por que devemos (necessitamos) escrevê-lo? Para responder a essa pergunta podemos fazer uma analogia com a resolução de um problema qualquer. Por exemplo, se quisermos levantar um bloco de aço que uma pessoa sozinha não consegue (não tem força suficiente) temos duas possibilidades para resolver o problema: chamar uma pessoa mais forte ou chamar várias pessoas. O mesmo ocorre na computação. Se quisermos resolver um problema mais rapidamente também temos duas abordagens: comprar uma máquina mais rápida ou dividir o problema entre várias máquinas.

Porém, se escolhermos a primeira solução, teremos um limite bem claro para a melhoria de desempenho do nosso programa, ou seja, o tempo que ele demora para executar na melhor máquina disponível (ou qual é o peso máximo que homem mais forte do mundo consegue levantar). Além disso, as máquinas mais rápidas são em geral muito caras e não estão prontamente disponíveis para uso, o que também inviabiliza o uso da primeira opção (força bruta).

Se escolhermos a segunda opção, dependendo do grau de paralelismo do programa, o limite para melhorar o desempenho do programa é o número de máquinas que conseguirmos disponibilizar para executarmos o nosso programa (ou o número de pessoas que conseguirmos chamar para levantar o bloco de aço). Atualmente, com o uso da internet é possível disponibilizar um número grande de máquinas para executarmos um programa. Porém, apesar de todo o potencial de melhora da segunda opção, ela é muito mais difícil de ser implementada e muito mais sujeita a erros.

O primeiro problema a ser resolvido é paralelizar o nosso programa. Escrever um programa paralelo é muito mais difícil do que escrever um programa seqüencial. Além disso, nem todo programa é possível de ser

paralelizado de maneira eficiente. O próximo passo é depurar os erros, que são muito mais difíceis de serem encontrados em programas paralelos do que nos programas seqüenciais. Agora, só temos que executar. Porém, para que o nosso programa tenha um bom desempenho um fator crucial é a distribuição das tarefas que compõem o programa entre as máquinas disponíveis. Se essa distribuição não for bem feita o programa pode não melhorar muito ou até ter um desempenho pior do que o programa seqüencial.

A distribuição das tarefas de um programa paralelo entre os recursos (máquinas) disponíveis é um problema em aberto (ainda não foi encontrado um algoritmo que consiga encontrar a solução ótima em tempo polinomial) muito estudado pelos pesquisadores da área de computação (chamado de escalonamento de tarefas).

Por último, colocar um programa paralelo para executar nas máquinas disponíveis não é uma tarefa fácil. Mesmo que todas estas máquinas estivessem em um único local, interconectadas através de uma rede, executar um programa paralelo nesse ambiente ainda é complicado. Talvez, com as novas pesquisas, em pouco tempo essa tarefa possa se tornar mais simples através das grades (*grids*) computacionais (FOSTER, 1999).

Logo, como podemos ver, apesar das vantagens de se escolher a segunda opção, executar programas em paralelo ainda é bastante complicado e praticamente apenas acessível a pesquisadores. Será que vale a pena todo o esforço de se executar um programa paralelo? Será que é esse o caminho a se seguir?

EXEMPLOS DE PROBLEMAS QUE PODEM SER PARALELIZADOS

A cada dia que passa os programas necessitam de maior capacidade de processamento. Dois exemplos desses programas são: o projeto genoma e os

grandes desafios. O projeto genoma foi criado com o objetivo de mapear todo o genoma (conjunto de cromossomos) do ser humano assim como seu código genético. Já os chamados grandes desafios são problemas científicos de larga escala, nas áreas de ciências e engenharia, com grande impacto econômico, político ou científico e que só podem ser tratados através da computação de alto desempenho.

Apesar de a capacidade de processamento ainda estar aumentando, esse crescimento é cada vez menor e mais próximo do limite físico (velocidade da luz). Por outro lado, o tamanho dos problemas tem aumentado mais rapidamente que o desenvolvimento tecnológico. Logo, a única opção a ser seguida é a paralelização dos programas para que os problemas sejam resolvidos em um tempo aceitável. A seguir veremos alguns desses problemas e por que eles demoram tanto para executar.

O PROJETO GENOMA

O projeto Genoma Humano é um esforço multinacional, iniciado em 1988, que tem como objetivo produzir o mapa físico completo de todos os cromossomos humanos, assim como a seqüência do DNA humano (SETÚBAL, 1999). Como parte desse projeto, genomas de outros organismos (tais como bactérias, fungos, moscas e ratos) também estão sendo estudados.

Apesar de tanto esforço, isto não é uma tarefa fácil, pois o genoma humano é composto de mais de três bilhões de pares de bases (nucleotídeos).

Este projeto contempla algumas das seguintes fases: usar técnicas de laboratório (por exemplo, canhão de ar) para pegar um DNA e partir em pedaços microscópicos; identificar esses pedaços (de no máximo 800 pares de base) em laboratório; pegar milhões desses pedaços (podendo conter pedaços repetidos ou não) e tentar juntar até formar o genoma humano. Em uma segunda fase,

identificar o código genético humano. Um gene é uma determinada seqüência de nucleotídeos responsáveis por produzir uma proteína específica.

Até pouco tempo acreditava-se que apenas 2 % do genoma humano era considerado útil, isto é, apenas a parte que contém o código responsável pela produção de proteínas (genes); o resto era considerado refugo. Porém, há pouco tempo os cientistas descobriram que esse refugo também tem um papel importante no desenvolvimento e características de todo o organismo (GIBBS, 2003).

Logo, apesar do projeto genoma ter praticamente conseguido decifrar o código genético humano (através de programas paralelos sofisticados e redes de computadores poderosas) muito trabalho ainda tem que ser feito. Imaginem se todo o seqüenciamento do genoma humano tivesse que ser feito à mão. Seria muito pior do que montar um quebra cabeça de três bilhões de peças, pois teríamos peças repetidas e, pior do que isso, apenas pedaços de peças repetidas.

O projeto genoma como podemos ver é um bom exemplo em que o uso do paralelismo teve um papel essencial no resultado final do trabalho, em termos de desempenho e qualidade.

OS GRANDES DESAFIOS

Em 1987, William Graham apresentou uma estratégia de cinco anos para o suporte federal à pesquisa e ao desenvolvimento da computação de alto desempenho. Dentro desse plano foi apresentada uma lista contendo os problemas chamados de grandes desafios. Esses problemas são, atualmente, com freqüência citados como protótipos dos tipos de problemas que demandam a força de um supercomputador.

A seguir será mostrada uma breve lista de alguns desses problemas (FOSDICK, 1995):

Previsão do tempo, clima e mudanças globais – O objetivo é entender o sistema atmosfera-oceano para que possam ser possíveis previsões de longo prazo sobre o seu comportamento.

Desafios na ciência dos materiais – A computação de alto desempenho fornece uma assistência imprescindível na melhoria do conhecimento sobre a natureza atômica dos materiais.

Projeto de semicondutores – À medida que materiais mais rápidos, tal como *gallium arsenide*, são usados em componentes eletrônicos, um conhecimento mais aprofundado é necessário para saber como eles operam e como mudar suas características.

Supercondutividade – A descoberta da supercondutividade em alta temperatura em 1986 permitiu o estudo da tecnologia de transmissão de força eficiente (econômica), instrumentos ultra-sensíveis e novos aparelhos.

Projetos de remédios – Previsões das arquiteturas das proteínas e das moléculas de RNA através das simulações em computador são uma ferramenta indispensável no projeto de novos remédios.

Astronomia – O volume de dados gerados atualmente pelos telescópios inunda os recursos computacionais disponíveis. Um poder computacional maior certamente trará avanços significativos nas descobertas sobre os outros planetas e o universo em geral.

Transporte – Contribuições substanciais podem ser feitas no desempenho dos veículos através de simulações nos computadores.

Esses foram apenas alguns poucos exemplos dos problemas que existem e que apesar de toda tecnologia e recursos existentes ainda não foi possível tratá-los de forma adequada. Para que isso seja possível é necessário aumentarmos bastante o poder computacional existente e a maneira de conseguir usar esse poder.

CONCLUSÃO

Construir programas paralelos é muito mais difícil e trabalhoso do que programas sequenciais. Porém, para resolver problemas complexos, apenas uma máquina (mesmo a mais rápida do mundo) não é suficiente. Logo, a utilização dos algoritmos paralelos é imprescindível. Como exemplos de problemas complexos em termos de desempenho e dados, foram mostrados o projeto genoma e os grandes desafios.

Apesar dos vários problemas mostrados para se desenvolver programas paralelos, recentes pesquisas mostram avanços no desenvolvimento de *middleware* (ambiente) para execução de tais programas. Esses ambientes permitem que o programa paralelo seja executado usando as máquinas disponíveis através de uma rede de computadores de maneira transparente ao usuário. Quem sabe, em poucos anos, construir e executar um programa paralelo seja uma tarefa tão comum quanto executar um programa sequencial hoje.

Referências bibliográficas

FOSDICK, Lloyd; JESSUP, Elizabeth. *An Overview of Scientific Computing*. Disponível em: <<http://www.citeseer.ist.psu.edu/article/fosdick95overview.html>>. Acesso em: jan. 2004.
FOSTER, I.; KESSELMAN, C. *Computational Grids. The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

GIBBS, W. Wayt. *O genoma oculto, além do DNA*. Scientific American Brasil, Duetto, ano2, n.19, p.82-89; jan. 2004.

SETÚBAL; MEIDANIS. *Introduction to Computational Molecular Biology*. PWS publishing Company, 1999.