Uma comparação entre dois modelos de soluções para problemas paralelos

Caio Dias Valentim
Departamento de Informática, PUC-Rio

14 de outubro de 2010

Resumo

Este documento visa comparar duas visões para criação de estratégias de paralelização de problemas. Uma descrita no livro do Foster e a outra descrita em artigo de Nicholas Carriero e David Gelernter.

1 Introdução

O documento está devidido da seguinte forma: Nas duas primeiras seções os textos são apresentados de forma bastante resumida, na seção **Contrastes** é caracterizada a principal diferença entre as abordages e, por fim, na **Conclusão**, uma visão geral do exposto é apresentada.

2 Foster

No segundo capítulo do livro [2], Foster mapea o processo de design de um programa paralelo em quatro etapas. Cada etapa é caracterizada por objetivos específicos e por uma papel na solução geral. Desta forma, se todas etapas forem concluídas de forma adequada, espera-se ter uma solução paralelizável de boa qualidade para um problema qualquer. Abaixo estão enumeradas e descritas as etapas.

1. *Particionamento.* É a parte em que deve-se pensar sobre como dividir o problema em pedaços menores. Nesse estágio são decididas uma ou várias formas de particionar o problema. São consideradas as possibilidas de decomposição funcional ou por domínio e são recomendadas algumas posturas, como, por exemplo, que os subtarefas sejam pequenas para facilitar uma distribuição igual das mesmas depois.

2. *Comunicação.* Nessa etapa é discutido como as tarefas definidas no passo anterior vão colaborar entre sim, ou seja, como elas vão enviar e receber mensagens para produzir seu trabalho. Ele divide o a comunicação em quatro grupos. Cada um com duas opções: local/global, estrutura/não-estruturada, estática/dinâmica e síncrona/assíncrona. Além disso, ele tenta metrificar o custo geral da comunicação no sistema.

3. *Aglomeração.* Revisitando os passos anteriores, busca-se agrupar um conjunto de pequenas tarefas em uma maior. No agrupamento ele considera a comunicação entre as partes maiores, bem como o tamanho das partes e requisitos práticos como quantidade de processadores na arquitetura destino.

4. *Mapeamento.* Por fim, no último passo, ele discute como deve ser feito o mapeamento das tarefas para os processadores disponíveis. Alguns algoritmos de balanceamento de carga e agendamento de tarefas são discutidos.

Para mostrar a aplicabilidade do modelo criado, Foster discute três aplicações onde mostra o passo-a-passo dos estágios de construção da solução.

3 N. Carriero e David Gelernter

No paper [1] é proposta uma maneira de projetar programas paralelos baseado em três tipos de paradgimas de paralelismo (tradução livre): Paralelismo do resultado, paralelismo de estágios e paralelismo de especialidades. Os autores do artigo acreditam que, de forma prática, todas soluções paralelas podem ser enquadrar em um dos modelos (templates) acima. Além disso, eles consideram questões práticas de implementação e mostram como esses se relacionam com os paradigmas citados. Os modelos propostos são resumidos abaixo.

3.1 Modelos

1. *Paralelismo do Resultado.* Essa classe de solução paralela é caracterizada pela partição do resultado pretendido (a saída do programa) em partes menores. Cada uma das partes é então construída de forma paralela e no final basta, se for necessário, juntar todas. Um exemplo prático desse modelo é a soma de dois vetores A e B de n posições. Cada posição i do resultado final pode ser calcula de forma independente por um ou mais processos paralelos.

2. *Paralelismo de Estágios.* Nesse modelo o problema é divido em estágios que devem ser completados para chegar a solução. Para cada estágio todos os processos estão disponíveis. O objetivo é usar o recurso disponível para completar os estágios e, quando todos estágios forem completados, chegar a solução do problema. Um exemplo claro desse tipo de solução é um design com pipelines.

3. *Paralelismo de Especialidades.* Já aqui o interessante é dividir os diferentes processos por especialidade e colocá-los para executar simultaneamente. Ou seja, cada processo sabe resolver um pedaço do problema e todos são colocados para executar. Naturalmente alguns processos terão que esperar para poder avançar, dependendo da estrutura do problema. O problema do produtor/consumidor se enquadra aqui, uma vez que temos duas entidades especializadas: O produtor responsável por gerar insumo, e o consumidor cuja função é consumir o insumo gerado.

4 Contrastes

A principal diferença entre os dois textos é a abordagem do processo de criação de soluções paralelas. No texto do Foster, é proposta uma série de etapas a serem seguidas para se criar um modelo paralelo. Ou seja, deve-se seguir a linha de racícionio, se preocupando com cada faceta do problema, uma por vez.

Já no artigo do N.Carriero e David o proposto são templates de solução que, presumidamente, funcionam na maioria dos casos. O processo de projetar uma solução, nesse modelo, começaria por identificar qual dos três tipos de soluções melhor se enquadra ao problema e então usá-lo.

Novamente, o artigo do Carriero é marcado pelos templates para soluções recorrentes em paralelismo, enquanto o livro do Foster apresenta um framework de pensamento para chegar a uma solução. Isso reflete uma postura distinta nos dois textos. No livro, parece que o autor não acredita em uma forma fechada de soluções que enquadrem todos os tipos de problemas. Sendo assim, ele busca estabeler pontos cruciais ao processo de *construção* da solução, delimintando etapas de *pensamento* e dando conselhos de como avaliar se cada etapa está boa ou não.

Já na outra abordagem os autores parecem crer ser possível enquadar as soluções paralelas em soluções comuns e quem podem ser usadas sempre. Assim, eles deixam claro o formato de uma solução paralela: Identificar o template que mais se enquadra para o problema e usar o que é conhecido para uma boa solução seguindo esse template.

Note que os templates não precisam ser exclusivos, ou seja, algumas soluções podem se utilizar de uma ou mais templates.

5 Conclusão

Os dois textos são muito bons para criar intuição sobre problemas paralelos. Apesar de abordagens diferentes, as dicas e estruturas de pensamento descritas em ambos são de grande utilidade.

Além disso, como mostrado, cada texto apresenta uma visão diferente das soluções paralelas e essas visões são tratadas em detalhes.

Referências

[1] Nicholas Carriero and David Gelernter. How to write parallel programs: A guide to the perplexed. *ACM Computing Survey*, 21:323–357, 1989.

[2] Ian Foster. *Designing and Building Parallel Programs.* Addison-Wesley, 1995.