Esqueletos de Paralelismo em Erlang

Rodrigo Costa de Moura[1], André R. Du Bois[1]

[1]Centro Politécnico – Universidade Católica de Pelotas (UCPel)
Caixa Postal 402 – CEP 91.501-970 – Pelotas – RS – Brasil

{rodrigomoura,dubois}@ucpel.tche.br

1. Introdução

Cada vez mais as aplicações computacionais precisam de maior poder de processamento para serem executadas. Uma maneira de conseguir esse grande poder computacional é utilizando o processamento paralelo. Uma motivação para a utilização dessa técnica é a popularização das máquinas multicore e também a possibilidade de se trabalhar com clusters, que são um conjunto de máquinas interligadas, trabalhando como um computador de grande porte. Porém, desenvolver aplicativos que explorem o paralelismo é mais complicado que a programação convencional. Programas paralelos precisam se preocupar com a criação dos processos e também em como esses processos serão manipulados.

Tendo em vista os desafios de desenvolver programas paralelos e visando ajudar o programador, busca-se alternativas para facilitar essa programação. Uma alternativa é a utilização de Esqueletos de Paralelismo.

Com o objetivo de facilitar o desenvolvimento de aplicações paralelas, este artigo apresenta a implementação de dois esqueletos de paralelismo na linguagem Erlang, o esqueleto Map Paralelo e o esqueleto Divisão e Conquista. Para testar os esqueletos, foram desenvolvidas duas aplicações simples, o cálculo da aproximação do Pi com o método de Monte Carlo e o cálculo do Fibonacci. Os resultados da execução dessas aplicações em uma máquina com processador de oito núcleos são apresentados e discutidos.

2. Esqueletos de Paralelismo

Esqueletos de paralelismo [Cole, 1989] são funções de alta ordem que encapsulam padrões recorrentes de paralelismo. Eles são utilizados para resolver problemas rotineiramente encontrados quando se desenvolve programas para execução paralela. Utilizando esqueletos de paralelismo, o programador pode desenvolver programas paralelos sem se preocupar em como os processos serão criados e escalonados.

3. Erlang

Erlang [Armstrong 2005, Erlang 2008] é uma linguagem funcional desenvolvida pela Ericson para ser utilizada em aplicações distribuídas e tolerante a falhas. A linguagem se destaca pela facilidade de criar processos e fazer a comunicação entre eles utilizando troca de mensagens. Os processos são fortemente isolados uns do outros, pois não existe compartilhamento de recursos entre eles. As principais primitvas para criação de processos e comunicação em Erlang são:

- **spawn:** serve para criar processos, recebe como argumentos o módulo onde está definida a função que será executada, a função a ser executada e os argumentos que a função receberá. Como resposta, retorna o identificador do processo criado.

  Pid = **spawn**(Modulo, Função, Argumentos)

- **!:** para enviar uma mensagem para um processo utiliza-se o caracter '!' após o identificador do processo, e passa-se a mensagens entre chaves.

  Pid**!** {mensagem}

- **receive**: para que um processo possa receber uma mensagem, ele deve conter a primitiva receive. A mensagem será recebida entre chaves após a primitiva receive. Após a seta, a mensagem pode ser manipulada.

```
receive
        {mensagem} -> mensagem
end.
```

Por essa facilidade de criar e manipular processos, a linguagem funcional Erlang foi escolhida para desenvolver os Esqueletos de Paralelismo apresentados neste trabalho.

4. Esqueleto Map Paralelo

Ao se trabalhar com linguagens funcionais, uma função muito utilizada é a Map. Seus argumentos são uma função e uma lista de elementos. A Map aplica a função a cada elemento da lista e devolve uma lista de resultados.

```
%Função map seqüencial
map(_, []) -> [];
map(F, [H|T]) -> [F(H)|map(F,T)].
```

No esqueleto Map Paralelo é criada uma lista de processos, um para cada elemento da lista. Cada processo receberá a função e um elemento da lista, aplicará a função ao elemento, devolvendo o resultado da avaliação. Ao final, os resultados das avaliações são combinados em uma lista de respostas. (Figura 1).

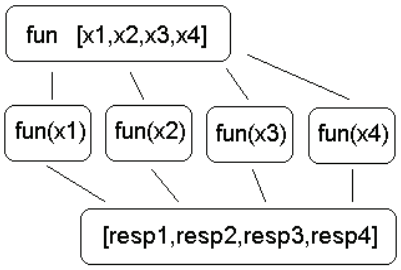

Figura 1. Divisão em processos e combinação das respostas em uma lista

4.1. Método de Monte Carlo

Para testes do esqueleto Map Paralelo, foi utilizado o cálculo da aproximação do Pi utilizando o método de Monte Carlo. Esse método consiste em gerar coordenadas aleatórias dentro de um quadrado com um círculo inscrito dentro dele (Figura 2).

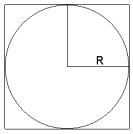

Figura 2. Forma onde são geradas coordenadas aleatórias

O cálculo do Pi é dado pela fórmula:
4*(pontos que incidem no círculo / total de pontos)

4.2. Testes

O esqueleto Map Paralelo foi testado no método de Monte Carlo para o cálculo da aproximação do Pi rodando em uma máquina com processador de oito núcleos. Foram feitos testes limitando o número de núcleos do processador, conseguindo ganhos de desempenho para todas as configurações (Tabela 1).

Tabela 1. Ganho de desempenho utilizando esqueleto Map Paralelo

Núcleos utilizados do processador	Speedup
2	1.89
4	3.55
8	6.05

5. Esqueleto Divisão e Conquista

A Divisão e Conquista consiste em pegar um grande problema com solução complexa e dividi-lo em vários pequenos problemas de fácil solução.

O esqueleto Divisão e Conquista testa se o problema é divisível. Se for, ele aplica uma função que cria uma lista de pequenos problemas. Então, utilizando o esqueleto Map Paralelo, é aplicada a função solução em cada problema que compõe a lista, gerando uma lista de soluções. Essa lista de soluções é combinada para gerar a solução final.

```
 -module(divCon).
 -export([divCon/5]).
divCon(Divisible,Split,Join,Fun,Prob)->
        case Divisible(Prob) of
                true  -> Join (mapp:main(fun(X)->Fun(X) end,Split(Prob)));
                false -> Fun(Prob)
        end.
```

5.1.Cálculo da Seqüência Fibonacci

No algoritmo utilizado para calcular o fibonacci de um número usando o esqueleto Divisão e Conquista, a função que divide o problema “abre” a árvore para o cálculo do

fibonacci até que a base tenha o número de elementos igual ao número de máquinas ou processadores em que a aplicação for rodar (Figura 3). Utilizando o esqueleto Map Paralelo é criado um processo para cada elemento da base e é executada a função fibonacci convencional em cada processo, retornando uma lista de resultados.

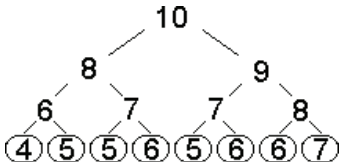

Figura 3. Árvore para o cálculo do fibonacci

Na lista de resultados, é aplicada uma função que une as respostas. Essa função soma as respostas aos pares recursivamente, até sobrar um elemento, que é a solução final.

5.2. Testes

O cálculo do fibonacci com o esqueleto Divisão e Conquista conseguiu um ganho de desempenho, mas devido ao desequilíbrio no tamanho dos processos, uns processadores ficaram ociosos enquanto outros ainda estavam trabalhando. Isso causou um speedup relativamente baixo: 1,18 com 2 processadores. É seqüência do projeto efetuar mais testes com variações do algoritmo para o cálculo do fibonacci, buscando um melhor equilíbrio no tamanho dos processos.

6. Conclusão

Com base nos resultados obtidos nos testes com os esqueletos de paralelismo implementados nesse trabalho, observa-se o ganho de desempenho na execução das aplicações do cálculo da aproximação do valor do Pi com o método de Monte Carlo e no cálculo do valor fibonacci em máquinas multicore, além de simplificar o desenvolvimento das aplicações testadas.

A linguagem Erlang, por sua sintaxe simples e facilidade na criação e manipulação de processos, se mostrou muito favorável na implementação de esqueletos de paralelismo para máquinas multicore.

A seqüência deste trabalho é desenvolver esqueletos de paralelismo direcionados a clusters, fazendo o escalonamento dos processos entre as máquinas que o compõem. Também, desenvolver os esqueletos de paralelismo em linguagens mais populares e de alto nível, como por exemplo, Java.

Referências

Armstrong, Joe. (2007). “Programming Erlang, Software for a Concurrent World”. Pragmatic Bookshelf.

Cole, Murray. (1989). “Algorithmic Skeletons: Structured Management of Parallel Computation”. Pitman.

Du Bois, A. R. (2008). “Memórias Transacionais e Troca de Mensagens: Duas Alternativas para Programação de Máquinas Multi-Core”, ERAD 2008.

Erlang (2008). Erlang. WWW Page, http://www.erlang.org/.