

USO DE PARALELISMO DE DADOS PARA MAIOR EFICIÊNCIA DE ALGORITMOS DE PROCESSAMENTO DE IMAGENS

Pedro Ribeiro Mendes Júnior

Orientadora: Lucília Camarão de Figueiredo

Orientador: David Menotti

UFOP - Universidade Federal de Ouro Preto

ICEB - Instituto de Ciências Exatas e Biológicas

DECOM - Departamento de Computação

BCC390 - Monografia I

31 de março de 2012

Introdução

- Processamento paralelo em algoritmos de PDI.
- Estrutura altamente paralela das GPUs.
- Paralelismo de dados.
- *Nested Data Parallelism* (NDP).
- Área de PDI parece fértil para explorar NDP.

Justificativa

- Paralelismo é uma questão atual e importante.
- Em Haskell os modelos de programação paralela são deterministas.
- Estudar eficiência do NDP implementado no DPH.

Objetivos

O objetivo geral do projeto é explorar oportunidades de uso de NDP para a implementação eficiente de algoritmos de PDI.

- Identificar algoritmos de PDI paralelizáveis.
- Obter implementações paralelas eficientes.
- Obter evidências a favor ou contra o paralelismo em Haskell.

Metodologia

- Estudo dos diversos modelos de paralelismo.
- Seleção de aplicações interessantes de PDI.
- Instalação e configuração de plataformas de desenvolvimento.
- Implementação de algoritmos selecionados.
- Comparações entre implementações.
- Avaliação dos resultados obtidos.

Desenvolvimento

- Implementação de algoritmos de PDI utilizando DPH, uma extensão do GHC que utiliza NDP.
- Problema de gerar Árvore de Componentes Conectados (CCT - *Connected Component Tree*)

Descrição do problema

A CCT é representada como uma *Point tree*.

Existe três principais classes de algoritmos sequenciais para computar a CCT:

- *Flooding-based algorithms*
- *Emerging-based algorithms*
- *1-D algorithms*

Descrição do problema

- Gerar a CCT para uma dimensão;
- Realizar o junção entre as soluções.

Descrição do modelo de paralelismo

- Blelloch desenvolveu uma linguagem para ensinar e implementar algoritmos paralelos.
- Descrições em alto nível de algoritmos paralelos.

As seguintes características são importantes de se conseguir:

- Um modelo de desempenho baseado na linguagem;
- Suporte para construções de NDP.

Exemplo da utilização do DPH em Haskell

```

1 quicksort :: [Int] -> [Int]
2 quicksort a = if length a <= 1 then a
3               else sa !! 0 ++ eq ++ sa !! 1
4   where m   = a !! 0
5         lt  = [e | e <- a, e < m]
6         eq  = [e | e <- a, e == m]
7         gt  = [e | e <- a, e > m]
8         sa  = [quicksort e | e <- [lt, gt]]

```

```

1 quicksort :: [Int:] -> [Int:]
2 quicksort a = if lengthP a <= 1 then a
3               else sa !: 0 +:+ eq +:+ sa !: 1
4   where m   = a !: 0
5         lt  = [:e | e <- a, e < m:]
6         eq  = [:e | e <- a, e == m:]
7         gt  = [:e | e <- a, e > m:]
8         sa  = [:quicksort e | e <- [:lt, gt]::]

```

Problemas

- *Framework* ainda se encontra em desenvolvimento.
- Não tem as mesmas facilidades de se programar.
- Incapacidade de misturar código vetorizado com não vetorizado.

Instalação dos softwares

- O DPH está disponível como uma extensão do GHC 7.2.
- Disponível na forma de um pacote cabal.
- Necessita de um Prelude de propósito especial.

Implementação

- Imagem representada como VP de VP de Word8.
- Resultado representado como um VP de VP de (Int, Int).
- A construção da CCT para uma dimensão não pode ser paralelizável.
- Construído sobre PArray.
- Necessidade de uma função *unwrapper*.

Trabalhos futuros

- Comparação com outras implementações.
- Comparação entre LLVM e C.
- Implementação de outros algoritmos de PDI.
- Criação de uma biblioteca com enfoque em paralelismo.
- Utilização das GPUs em implementações em Haskell.

Cronograma de atividades

Atividades	Mar	Abr	Mai	Jun	Jul
Terminar implementação	X				
Seleção de aplicações		X	X	X	
Estudo das bibliotecas	X	X	X	X	
Implementações	X	X	X	X	
Comparações		X	X	X	
Avaliação dos resultados		X	X	X	
Redigir a monografia				X	X
Apresentação do trabalho					X

Tabela: Cronograma de Atividades.

Fim

Perguntas?