MO401
IC - Unicamp

Trace Scheduling

Márcio Paixão Dantas
Junho, 2006

Roteiro

1. Introdução e Motivação
2. Trace Scheduling (TS)
3. Variações do TS
    1. Tree Compaction
    2. Singly Rooted Directed Acyclic Graph (SRDAG)
    3. Improved Trace Scheduling Compaction (ITSC)
4. Comparação entre algoritmos
5. Conclusão

Introdução

- Problema:
  - Otimização de microcódigo
- Importância:
  - Aumento do código --> dificuldade de identificar oportunidades de otimização
  - Desenvolvimento de Compiladores

--> Técnicas que produzam código eficiente

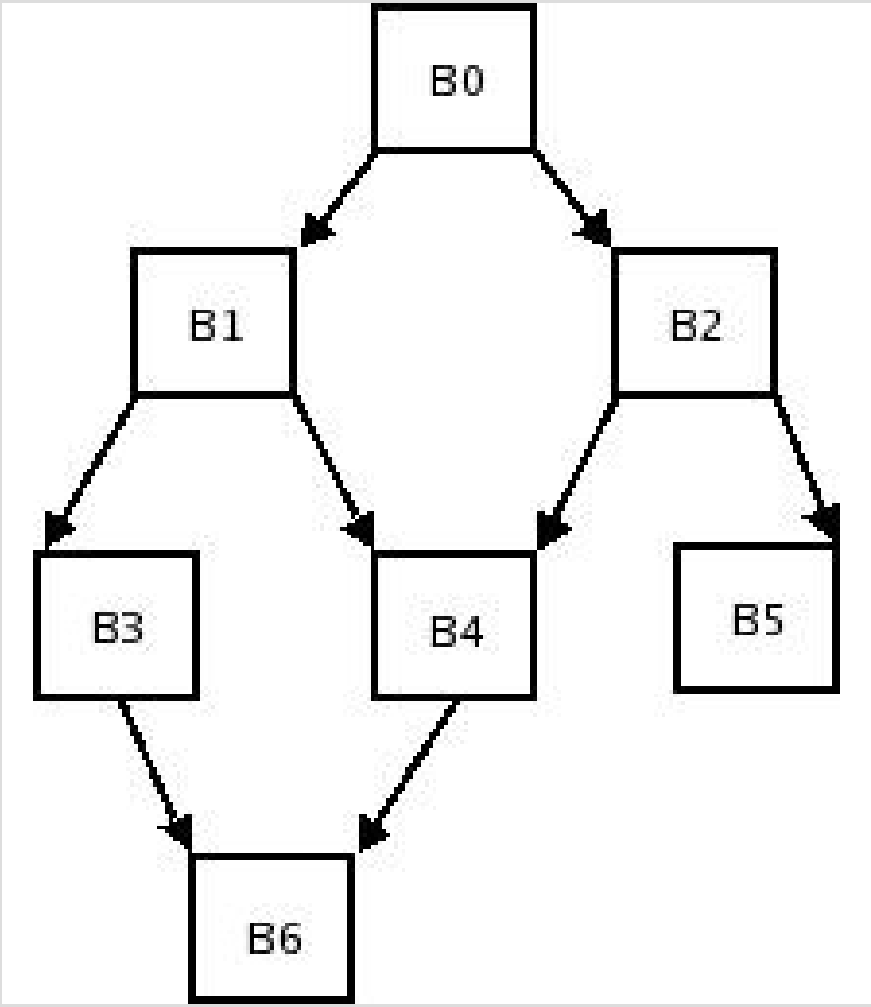
Introdução

Historicamente:

- Compactação Local
  - List Scheduling

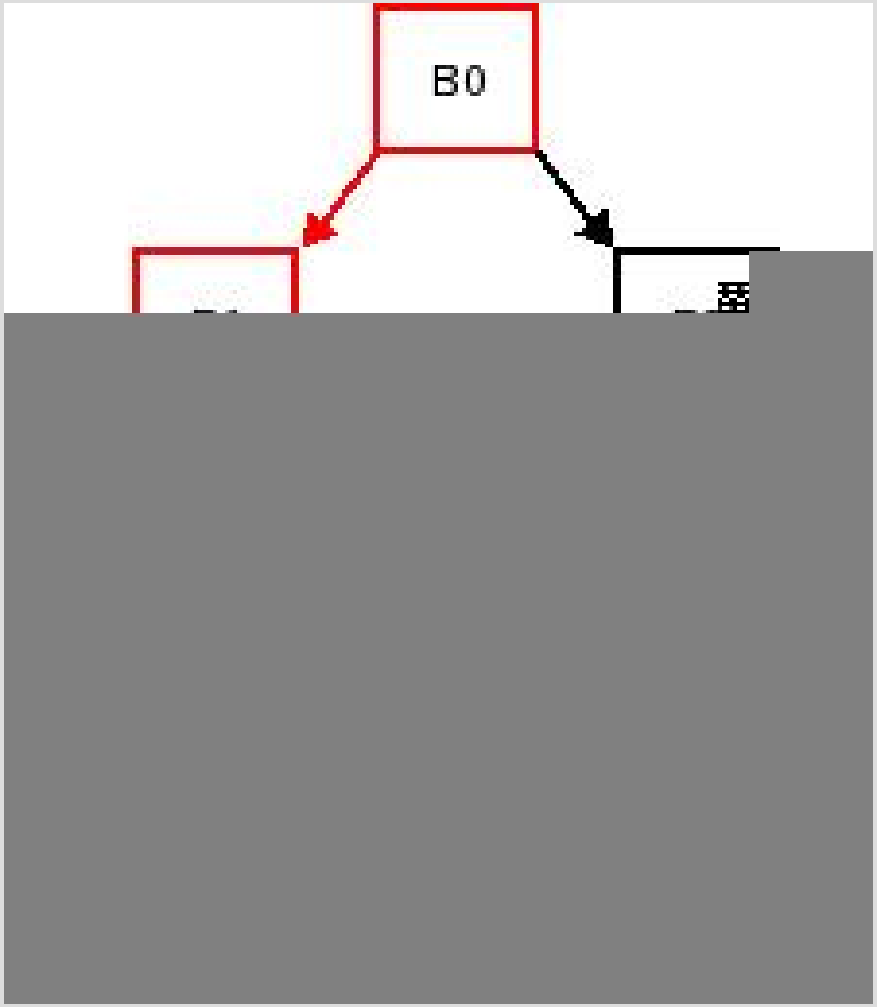
- Compactação global
  - Comp. Local + Movimentação entre blocos
  - Trace Scheduling (Fisher, 1981)

Trace Scheduling (TS)



Grafo de Fluxo de P

Trace Scheduling (TS)



Grafo de Fluxo de P

Funcionamento Básico:

- Escolha de *Traces* (caminhos), T, com maior probabilidade de serem executados
- Compactação de T
- *Bookkeeping*

Trace Scheduling (TS)

- Compactação de T
  - Definição de relação de precedência entre operações;
  - Construção de um DAG de precedência de dados;
  - Imposição das movimentações legais de MOP's no DAG;
  - Compactação de T como se fosse um bloco básico por *List Scheduling.*

Trace Scheduling (TS)

- Vantagens
  - Tempo de execução bem melhor que abordagens anteriores

- Problemas
  1. Cresc. tamanho de P pode ser exponencial;
  2. Estrutura do código otimizado pode ser bastante diferente da estrutura original;
  3. Desempenho ruim quando caminhos apresentam ciclos.

Tree Compaction

- Objetivo
  - Amenizar problemas 1 e 2.
- Idéia básica:
  - Realizar mesmas compactações feitas por TS, exceto aquelas que implicam cópia de blocos.

Tree Compaction

- Definições
  - *Top tree*
    - Árvore com um único nó com grau de entrada zero (raiz da árvore).
  - *Bottom Tree*
    - Árvore com um único nó com grau de saída zero (raiz da árvore).

Tree Compaction

Algoritmo

1. Particionar blocos de P em subconjuntos de *top trees*;
2. Aplicar *Trace Scheduling* em cada árvore;
3. Particionar blocos em *bottom trees*;
4. Fazer 2 até que todos os blocos estejam compactados.

Tree Compaction

- Vantagens
  - No pior caso, crescimento no tamanho do programa é da ordem de n^2, onde n é o número de desvios condicionais.
  - Complexidade computacional menor que TS;
  - Facilidade de implementação.

- Desvantagem
  - Tempo de execução pior que TS.

SRDAG

- Objetivos
  - Aumentar contexto de informação;
  - Escolher corretamente caminhos com maior probabilidade.

- Idéia básica
  - Usar um SRDAG para compactar cada bloco de P, ao invés de usar e compactar caminhos inteiros.

SRDAG

Algoritmo

Enquanto houver bloco não-compactado:

1. Seleciona raiz;
2. Seleciona SRDAG;
3. Compacta raiz;
4. Atualiza probabilidades e DAG original.

SRDAG

- Vantagens
  - Melhora tempo de execução e uso de memória em relação ao TS.
- Desvantagem
  - Maior complexidade computacional

ITSC

- Objetivos:
  - Reduzir espaço requerido;
  - Reduzir tempo de execução, sobretudo em caminhos contendo ciclos.

- Usa:
  - Conjunto modificado de regras de movimentação de MOP's entre blocos;
  - Algoritmo de TS melhorado;
  - Algoritmo de URCR (*unrolling, compacting, rerolling*).