ソフトウェアトレースにおけるレジスタ再割り当てアルゴリズムの検討

請園　智玲 (北陸先端大)・田中　清史 (北陸先端大)

1　はじめに

近年，ダイナミックリンクライブラリやダイナミッククラスローディング技術がソフトウェア開発において一般的に使用されている．これら技術はソフトウェアの差分配布や別プログラム間のコード共有化に有用であるが，実行中にプログラムロード行うため，関数のインライン展開やコード配置最適化などモノリシックプログラムに適用できた広いスコープの最適化を行うことが難しい．また，通常のコンパイラによる最適化では，実際の実行において通るパスとその頻度を知ることができず，この情報を利用した最適化を行うことも同様に難しい．これら問題を解決するために動的最適化が注目され，研究されている [3][4]．我々は動的最適化環境を軽量に実現するためのハードウェア／システムソフトウェア機構とその機構を利用した最適化アルゴリズムであるソフトウェアトレース生成アルゴリズムを提案した [1][2]．本稿では提案した最適化アルゴリズムより生成されるソフトウェアトレースに適用するレジスタ再割り当てアルゴリズムを検討する．

2　ソフトウェアトレース

本節では既に提案したソフトウェアトレース [2] の概要を説明する．スーパースカラプロセッサは 1 クロックに複数の命令をフェッチし，実行する機能を持つ．現代のプロセッサは動作周波数が向上し，命令フェッチに充てる時間が減少したことから，1 クロックでフェッチ可能な命令は 1 つの命令キャッシュブロックの連続した領域に限られる．このため，分岐命令によりメモリアドレス空間上連続しない命令フェッチを行う場合にはフェッチ能力自体が低下する．

ソフトウェアトレースはプログラム上のループ構造に着目し，動的にループ内のパスプロファイリングを行うことにより，最も実行されやすいパスを推測し，そのパスをトレースとしてオリジナルのプログラムに付加し，ループ内の条件分岐命令の taken 実行確率を減少させ命令フェッチ能力を向上させる最適化である．

ソフトウェアトレースの付加的利点として，ループ内のトレースのコピーをオリジナルとは別に主記憶上に配置しオリジナルコードと干渉しない点が挙げられる．動的最適化は実行時に実行中コードを最適化するため，オリジナルコードの命令修正，追加，削除を行う場合，PC 相対分岐命令のリロケーション解決をプログラム全体に施さなければならない．これは動的最適化にとって大きなオーバヘッドとなる．しかしながら，ソフトウェアトレースに対して最適化を行う場合は，リロケーション解決の範囲はソフトウェアトレース内部に限定されるため，オーバヘッドは軽減する．

3　レジスタ再割り当て

プログラムから指定できるデータ格納領域は通常，汎用レジスタと主記憶の 2 つ存在する．汎用レジスタは命令実行パイプラインの最も近い位置に存在するデータ格納領域であるため，最もアクセスの速いデータ格納領域であるが，代表的な RISC アーキテクチャに見られるように多量に用意することは難しい．そのため，直近で使用されるデータや再利用性の高いデータを優先して配置すべきである．高級言語でプログラムを作成する際に，コンパイラはこのことを考慮してレジスタ割り当てを行う．しかしながら，プログラムにはその制御構造により実行パスが存在する．実行パスは実行時に入力データにより変動するため，コンパイラは 1 本のパスに全てのレジスタ資源を充てることができずに，特に再利用性の観点から見た優先度の低いデータはメモリに退避する．本手法が対象とするパスの違いによるレジスタ空き資源変化を図 1 に示す．

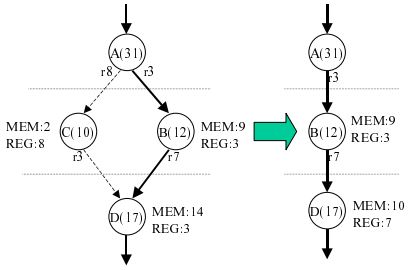

図 1: レジスタ再割り当て

図はグラフによりパスを表現しており，節が基本ブロック，枝が分岐方向を示す．それぞれの基本ブロックは A〜D の名前がある．名前の右の括弧で示される数はその基本ブロックで必要とされる変数の数，枝に付随する r 付きの数は枝の発生元基本ブロックの実行後，解放されるレジスタの数である．(r8 ならばレジスタ番号 0〜7 が解放) 基本ブロックの横に示される MEM と REG に付随する数はその基本ブロックで必要な変数がレジスタとメモリに割り当てられた数である．A は全てのレジスタを使い切っており，そこからの解放数が枝が指す先の基本ブロックが唯一利用可能なレジスタ数であるとする．左のグラフがオリジナルコード，右がソフトウェアトレース上のパスを示す．オリジナルコード中段の B と C はそれぞれ解放数が異なるが，D は解放数の少ない C の解放領域でレジスタ割り当てを行わなければならない．これは D 以降の処理で 3〜6 のレジスタを利用する可能性が有るためであり，D の時点ではこれを上書きすることができないためである．このため，D のレジスタ割り当て数は 3 となっている．一方，これをソフトウェアトレース生成によって選ばれた ABD のパス上では 7 つのレジスタ全てを D のレジスタ割り当てに使用している．しかしながら，ソフトウェアトレースで生成したパスは予測に基づいており，プログラム実行上予測が外れる場合がある．この時の正常な処理を実現するため，トレース実行前にあらかじめレジスタ値を主記憶に退避し，予測ミス時にレジスタ値を復帰できるようにソフトウェアトレースは生成されている．

4　最後に

本稿ではソフトウェアトレースの特性として得ることが可能なレジスタ空き資源を利用するためのレジスタ再割り当て手法を示した．

参考文献

[1] 請園 智玲，田中 清史，”ハードウェア解析システムによるバイナリコードの動的最適化”，情報処理学会研究報告，ARC, Vol.2005, No.120, pp.7–12, 2005.

[2] 請園 智玲，田中 清史，”ソフトウェアトレース生成による動的最適化の予備評価”，情報処理学会研究報告，ARC, Vol.2006，No.88, pp.169–174, 2006.

[3] Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia. Dynamo：A Transparent Dynamic Optimization System. SIGPLAN Conference on Programming Language Design and Implementation, pages 1–12, 2000.

[4] Alex Ramirez, Josep L. Larriba-Pey, Carlos Navarro, Josep Torrellas, Matero Valero. Software Trace Cache. Proceedings International Conference on Supercomputing, pages 119–126, 1999.