実行パスの静的解析による自動メモ化プロセッサの高速化手法

佐藤 裕貴*，藤井 政圭，津邑 公暁（名古屋工業大学）

Static Execution Path Analysis for Auto-Memoization Processor

Yuki Sato, Masayoshi Fujii, Tomoaki Tsumura (Nagoya Institute of Technology)

1. はじめに

プロセッサの高速化手法として，プログラムの並列性に着目した手法が注目されている．一方，これとは全く異なる概念である計算再利用と呼ばれる高速化手法がある．本研究では，この計算再利用の実行モデルの 1 つである**自動メモ化プロセッサ**[1]において，実行対象プログラムを静的解析することによって，実行パスを考慮した再利用適応度を算出し，高速化を図る手法を提案する．

2. 自動メモ化プロセッサ

自動メモ化プロセッサは，関数およびループを計算再利用の対象区間と見なし，実行時にその入力と出力の組を再利用表と呼ばれる表へ登録する．その後，再び同じ再利用対象区間に侵入する際，再利用表を参照し現在の入力の組と過去の入力の組を比較する．もし，現在の入力の組が再利用表上のいずれかの入力の組と一致する場合，その入力の組に対応する出力の組をレジスタやキャッシュに書き戻すことで，再利用対象区間の実行を省略する．しかし，再利用を適用する際，再利用表に対するアクセスに伴いオーバヘッドが発生する．これらのオーバヘッドが大きい場合，計算再利用を適用することでかえって性能が低下してしまう可能性がある．そのため，自動メモ化プロセッサは計算再利用を適用することで削減できるサイクル数と発生するオーバヘッドから再利用適応度を算出し，再利用による効果が得られないと判断すると，再利用の適用を中止し性能低下を抑制する．

3. 提案手法

前章で述べたように，自動メモ化プロセッサは再利用を適用するか否かを判定するために，削減できるサイクル数と発生するオーバヘッドを用いて再利用適応度を算出する．この削減できたサイクル数と発生したオーバヘッドは，命令区間の実行終了時に入力と出力の組を再利用表に登録する際に，同時に登録される．そして，再び同一命令区間を通常実行する際に，再利用表に記憶してある削減できたサイクル数と発生したオーバヘッドを用いて，再利用適応度を求める．しかし，この削減できたサイクル数と発生したオーバヘッドは条件分岐によって変化する実行パスごとに異なる．そのため，過去に通った実行パスと直近の実行パスが異なる場合，正しい再利用適応度を算出できず，再利用の効果が得られない実行結果を再利用表に登録してしまう可能性がある．そこで，条件分岐を考慮して，各実行パスに対応した再利用適応度を正確に算出するソフトウェア解析手法を提案する．提案手法では，プログラムを静的解析することで条件分岐によりスキップされる区間を取得し，その区間を考慮した再利用適応度を含むヒント命令を実行対象プログラム内に挿入する．そして，挿入したヒント命令に含まれる解析情報を利用することで，再利用の効果が得られない実行パスを通った際の入力と出力の組を再利用表に登録せず，命令区間の実行を終了する．これにより，再利用の効果が得られる実行パスを通った際の入力と出力の組をより多く再利用表へ登録することができる．その結果，従来手法では再利用を適用できなかった命令区間に対しても再利用を適用することが可能となる．

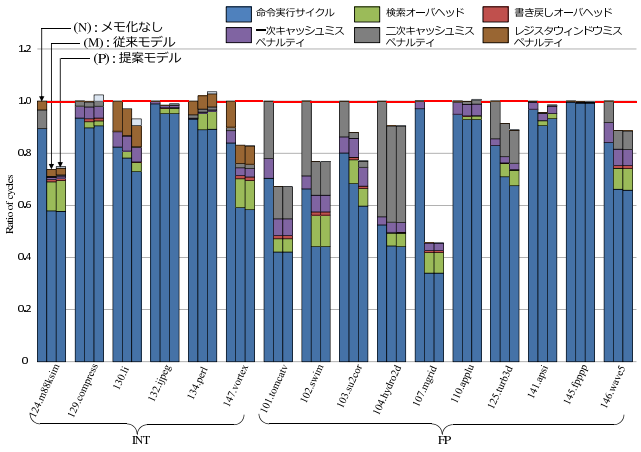

Fig.1　Evaluation Results

4. 評価

評価には，汎用ベンチマークプログラムである SPEC CPU95 を用いた．評価結果を Fig.1 に示す．結果は，メモ化無し (N)，従来モデル (M)，提案モデル (P) を示しており，メモ化無し (N) を 1 として正規化している．評価の結果，130.li 等いくつかのプログラムでは，命令実行サイクル数が削減できていることがわかる．これは，提案手法により再利用の効果が得られる実行パスを通った際の入力と出力の組を，従来モデル (M) より多く再利用表へ登録することができ，それらの結果の再利用に成功したためであると考えられる．提案手法の効果が最も大きかった 103.su2cor において，メモ化無し (N) と比較し，従来モデル (M) では 11.9 %のサイクル数を削減したのに対し，提案モデル (P) では 22.9 %のサイクル数を削減し，性能向上を確認した．

文　献

(1) Tsumura, T.,et.al.: Design and Evaluation of an Auto-Memoization Processor, *Proc. Parallel and Distributed Computing and Networks*, pp. 245–250 (2007)