1B－2

マルチプロセッサSNAILのネットワークに関する性能評価と支援ハードウエアの実装*

安川 英樹　笹原 正司　寺田 純　塙 敏博　天野 英晴†
慶應義塾大学 理工学部‡

1 はじめに

現在、SSS(Simple Serial Synchronized) 型 MIN (Multistage Interconnction Network)[2] を用いた 16 プロセッサからなるマルチプロセッサ SNAIL を実装中である [1].SNAIL は、その開発目的の一つに SSS-MIN の評価が挙げられる. その為には、ハードウエアの各部をモニタし、統計を取るパフォーマンスモニタが必要であり、既に実装済みである. 今回はこのパフォーマンスモニタを用いて、12 プロセッサの構成での評価を行なうことができたので、これを報告し、検討する.

2 評価

2.1 評価条件

今回の評価は 12 プロセッサ 4 メモリモジュールで行なった. 評価に用いたアプリケーションは以下の 3 種類で, いずれも, 各プロセッサはほぼ同様の処理を行なう性質を持つ. プロセッサ間の同期は Test&Set を用いたバリア同期を用いている. MIN に対する最悪の負荷を想定するため, 同期変数に対して各プロセッサは busy wait を行なっている.

- Four coloring
  隣接する領域が同色とならないように各領域を塗り分ける問題をニューラルネットワークを用いたアルゴリズムにより解くプログラム. 本評価ではアメリカ合衆国の 48 州を用いた.
- N-Queen
  $n \times n$ のチエス盤上に n 個のクィーンを互いに取り合うことのない位置に配置する問題をニューラルネットワークを用いたアルゴリズムにより解くプログラム. 本評価では $n = 32$ を用いた.
- Jacobi
  連立一次方程式をヤコビ法を用いて解くプログラム. 200 元の三重対角型の帯行列を用いている.

これらアプリケーションはローカルメモリの有効利用に関する最適化が施されており, コード・ローカルデータ・初期化後変更されないデータ等は各プロセッサのローカルメモリに配置されている. 以下に台数効果と Message Combining の効果について述べる.

2.2 実行時間の評価

2.2.1 台数効果

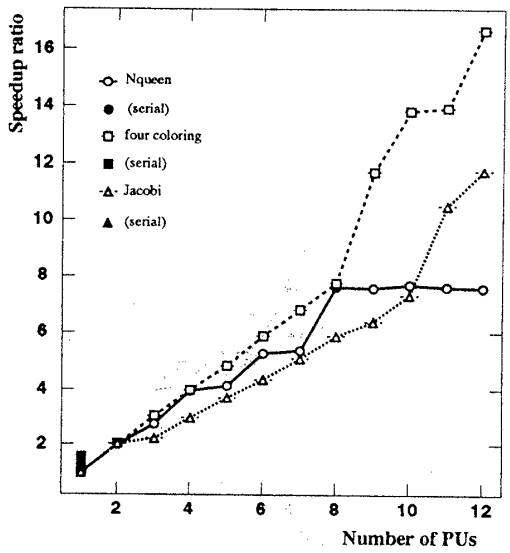

図 1: 台数効果

各アプリケーションにおけるプロセッサ数の増加による性能の向上を図 1に示す. この図は 1 プロセッサで並列プログラムを実行した場合の実行時間を基準としているが, 同じ問題を逐次処理で解いた場合の実行時間 (serial) も併せて図中に示している. この逐次プログラムは, 全てのデータをローカルメモリ上におくと共に, 同期操作の部分等並列化のロスを省いている.

この図によると,1 プロセッサで実行した場合は, 全ての問題について全てをローカルメモリにおいた場合の方が 40%から 50%高速である. N-queen については並列性の限界により,8 プロセッサ以上では性能の向上が見られないが,four coloring では, 並列探索により逐次処理による無駄な探索が減るため, 並列処理による高速化がなされている. これに対し Jacobi 法はアルゴリズムの性質上, 共有メモリを頻度にアクセスするため, 並列化の損失がやや大きい.

2.2.2 Message Combining の効果

上記のプリケーションについて, Message Combining の効果を調べた所、どれひとつとしてその効果は観測できなかった. そこで, 共有メモリに対するアクセスが最も頻繁な Jacobi について, 扱う問題のサイズを小さくする事により全体の同期操作の頻度を増し,Message Combining の効果を観測した. 図 2は Jacobi において, 行列のサイズをプロセッサ数に等しくとって実行時間を調

*Performance Evaluation about Network of Multiprocessor SNAIL and Implementation of Support Hardware
†Hideki YASUKAWA, Masashi SASAHARA, Jun TERADA, Toshihiro Hanawa, Hideharu AMANO
‡Keio University

べた結果である. この結果によると Message Combining により約 1%の性能向上が観測された.

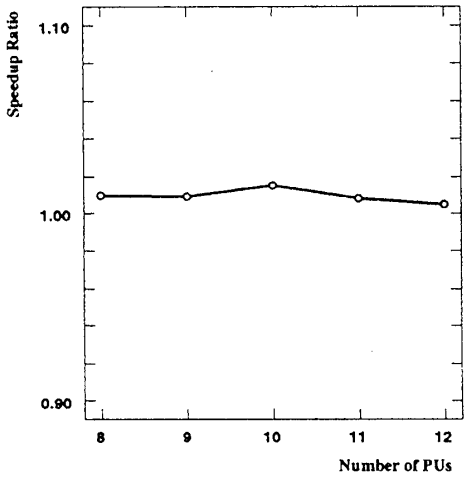

図 2: コンバインの効果 (1)

Jacobi 法では, 共有メモリ上のベクトル要素に対するアクセス頻度自体はサイズに依存せず大きい. このことから, Message Combining は同期操作に対してのみ効果を発揮し, 行要素自体のアクセスに対する効果はほとんどないことがわかる.

次に同期操作を Message Combining 可能な Test&Set を用いたものから,Message Combining の行なわれない Fetch&Dec を用いたものに変えて試みたが, 実行時間の変化はほとんどなかった. これにより,Message Combining が実際に行なわれているのが, バリア同期を行なう場合の変数に対する同期命令ではなく, 同期変数の変化を認識するための busy wait 中の読み出しアクセスであることがわかる. Jacobi 法では, 全プロセッサの足並みがほぼ揃ってしまうため,busy wait のループを何度も回ることがない. このことが Message Combining の効果が小さいひとつの原因になっている.

さらに,Message Combining の効果を調べるため, オンチップキャッシュは用いずに, 共有メモリ上の命令コードを全プロセッサで同時に実行する実験を行なった. 図 3にその結果を示す. プロセッサ数が 3 以下では, アクセスの衝突が起きないため, 実行時間にはほとんど変化がない.4 を越えると,Message Combining がない場合は, アクセス競合により性能が台数にほぼ比例して低下する. これに対し,Message Combining を行なう場合は, 各フレームで完全に同期して一つづつ命令が実行されるため, 性能の低下が起こらない. この場合, プログラムは疑似的に SIMD 動作していることになる. この性質は同期動作を行なう SNAIL 独自のものであり,Message Combining は非常に有効に働く. このような動作は現実性が低いが, ローカルメモリ上にコードを置いた場合でもスケジューリングを完全に行なうことができれば, フレームに対して同様の同期アクセスを行ない, データのマルチキャストを高速に行なうことが期待できる.

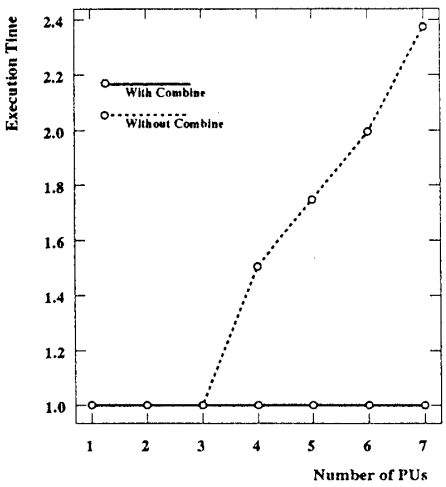

図 3: コンバインの効果 (2)

3 おわりに

SNAIL の実装と評価を通し,SSS 型 MIN がアクセス遅延, アクセス競合については十分な性能を持っていることが示され, Message Combining に関しても小規模ながら実際的な評価を取ることができた.

現状ではプロセッサ台数も少なく (12 台), アプリケーションも, 共有メモリに対するアクセスパターンが単純なものであり, さらに SSS 型と従来型の Message Combining は性質が異なるが, 以下の点が確認された.

(1) 実際のプログラムでは Messag Combining が有効になるのは主として同期操作である.

(2) Read 同士の Message Combining は, 同期変数が書き換わったことを検出するための busy wait に対して有効であるが,Jacobi 法のようにプロセッサの足並みが揃うような問題では大きな効果はない.

(3) SSS 型 MIN における Message Combining はハードウエア量の増加は 20%程度であり, スケジュールとの併用によりデータのマルチキャスト等に効果を発揮できる可能性がある.

現在, 共有メモリ型マルチプロセッサ用ベンチマーク Splash の中からいくつかのアプリケーションを移植しており, また,busy wait を用いない同期についても評価中である. この結果を踏まえ,Message Combining の効果についてさらに検討を進める予定である.

参考文献

[1] 笹原正司、寺田純、大和純一、天野英晴、: SSS 型 MIN に基づくマルチプロセッサ SNAIL の実装と評価, 信学技報,CPSY93-29,pp.9-16 (1993).

[2] Amano, H. Zhou, L. Gaye, K.:*SSS (Simple Serial Synchoronized)-MIN: A novel multi stage interconnection architecture for multiprocessors*, Proc. of IFIP Congress 92, Sept. pp. 571-577, (1992).