C-025

Beowulf型クラスタシステムにおけるメッセージの振舞いの観測と分析
Analytical observation focused on the message behaviors in Clustered computer system

工藤　達矢†　　坂下　善彦‡
Tatsuya　Kudou　　Yoshihiko Sakashita

1．はじめに

近年では、高解像度画像の高速処理や気象予測、最適化問題のような大規模数値解析に基づくシミュレーションには膨大な計算が必要でスーパーコンピュータが用いられているが、最近ではコストパフォーマンスの良さで PC クラスタが注目されている。

MPI（Message Passing Interface）を用いた、並列処理はプログラムを同時に実行可能な複数の部分に分け、それぞれをプロセッサ上で動かす。プログラムを N 個のプロセッサに分けて実行すれば、1 つのプロセッサで動かすのに比べて N 倍速く実行できると考えられる。しかし、並列処理を行うことでプロセッサ数に比例して速度が向上するとは限らない、その要因にオーバーヘッドと負荷不均等がある[1]。

プロセッサ間のメッセージ通信を行うことで通信によるオーバーヘッドが加わり、計算時間の総和は一つのプロセッサで逐次処理を行った場合の計算時間を上回り、頻繁にプロセッサ間でメッセージ通信が行われる場合には、オーバーヘッドが大きくなり処理の高速化に繋がらない[2]。

また、クラスタは任意の構成が可能でスケーラビリティにも優れているが、プロセッサの能力が均一でない場合、負荷に不均等が発生する。プロセッサの負荷が均等でないと、1 つのプロセッサの処理が終えても、すべてのプロセッサの処理が終わるまではプログラムは終了しないので、プロセッサが待ち時間が生じているということになる[3]。

本研究では MPI[4]のメッセージをトレースし並列処理にかかる時間を解析し、同期と通信に焦点を当て処理フロー依存性を分析することで、オーバーヘッドの要因を明らかにしプログラムの修正を行い、メッセージ通信によるオーバーヘッドを少なくする。

2. Beowulf

2.1 クラスタの構成

本研究で使用したクラスタは一つのサブネット内で構築し、構成は表 1 の通りである。

表 1　クラスタの構成

ノード数	4 台
CPU	AMD Athlon 64 3500+ 2.2GHz
メモリ	1GB
OS	Ubuntu8.04
MPI	Mpich1.2.7p1

†湘南工科大学大学院情報工学専攻
Shonan Institute of Technology

‡湘南工科大学
Shonan Institute of Technology

2.2 MPI

MPI は、分散メモリ型の並列計算機の CPU 間のデータ通信を行うためのライブラリである。
分散メモリ型の並列計算機はそれぞれの計算機は独立したアドレス空間を持っていてメモリを共有していないので、メッセージ通信で CPU 間のデータのやり取りを行う。
複数の CPU がデータをメッセージとして送受信することで並列計算を行えるようにする。メッセージ通信はプロセスから他のプロセスにデータを明示的に送る方法で効率のよい並列プログラムを書くことができる。

3. 姫野ベンチ

姫野ベンチマークは情報基盤センター・センター長の姫野龍太郎氏が非圧縮流体解析コードの性能評価のために考えたものでポアッソン方程式解法をヤコビの反復法で解く場合に主要なループの処理速度を計るものであり、即座に実測速度を求めることが出来る[5]。

姫野ベンチは三次元行列に大量のデータを読み書きすることで、主に計算機のメモリバンド幅などの性能による効果が大きいベンチマークである。メモリバンド幅は計算機が一秒間に転送可能なデータの容量を示しており、これは主に計算機のメモリやプロセッサの処理能力などで決まる。コードは非常に短く簡単にコンパイル・実行できるので、即座に実測速度（MFLOPS）を求めることができるので並列計算ベンチマークとしてよく使われている。

PC では配列が大きくなりキャッシュから溢れるたびに MFLOPS 値が悪くなる、これから姫野ベンチマークでマシンの性能を測定する場合、すべての計算機で計算サイズを固定しないと公平な測定値が得られないことになる。姫野ベンチマークでは 5 通りの計算サイズを固定し、測定している。

表 2　姫野ベンチの計算サイズ

	i x j x k
XS	33 33 x 65
S	128 x 64 x 64
M	256 x 128 x 128
L	512 x 256 x 256
XL	1024 x 512 x 512

姫野ベンチはまず、ヤコビの反復法を 3 回行い、掛かった時間から 1 分間で実行できる反復回数を再度計算し、時間と反復回数から 1 秒間に実行した浮動小数点計算の回数を求める。本研究では、反復回数を固定することで実行時間の差などの性能を比較できるようにするため、計算サイズは S 反復回数を 7000 に固定して計測を行う。

姫野ベンチの計算の主な部分はｉｘｊｘｋの３重ループで計算し、計算の後に圧力を表す配列 p の値を更新して他のプロセスと交換し、それを繰り返すことで誤差を少なくする。圧力の交換はメッセージ通信で行われており、このメッセージをトレースする。

4. メッセージのトレース

姫野ベンチを実行し、MPI のコンパイルオプションである mpilog を使いメッセージのログを取り並列プログラム視覚化ツール jumpshot を使い可視化してメッセージのトレースを行った。

姫野ベンチで行われているメッセージ通信関数には非同期でメッセージの送信を行う MPI_Isend、非同期でメッセージの受信を行う MPI_Irecv、メッセージの送受信が終わるのを待つ MPI_Waitall、全てのプロセスの指定されたアドレスの値を集めて演算を行い結果を全てのプロセスに送信する MPI_Allreduce、全てのプロセスの同期を行う MPI_Barrier がある。図１は何も表示されてない間は計算時間、横棒の部分が左から MPI_Isend, MPI_Waitall, MPI_Allreduce を表している。

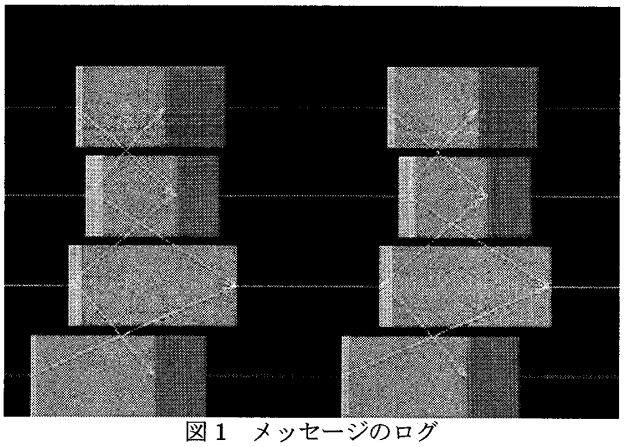

図１　メッセージのログ

図２で示すように４台で測定を行った際には処理時間の内48%をメッセージ通信が行われているため４台の測定結果は１台での測定結果の約２倍程度の MFLOPS 値しか得られなかった。

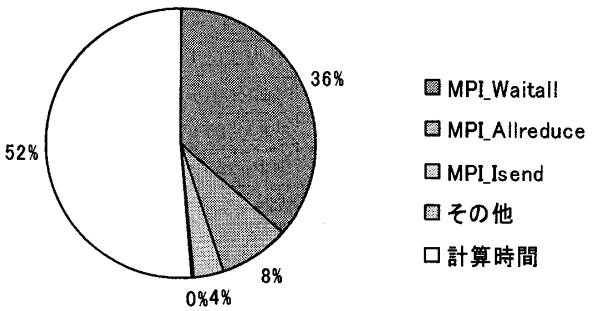

図２　４台で実行時の姫野ベンチの処理時間の割合

本研究で使用したクラスタでは姫野ベンチの処理時間の内8%は MPI_Allreduce が行われているが、姫野ベンチではこの MPI_Allreduce の結果はループごとに毎回０にリセットされているので、MPI_Allreduce を行う回数を減らすようにプログラムを修正することで、処理時間の内の MPI_Allreduce が行われている時間は 1%未満に短縮され、姫野ベンチの測定結果は図３で示すようにプロセッサ数が２つ以上の場合に約 8%の性能が向上した。

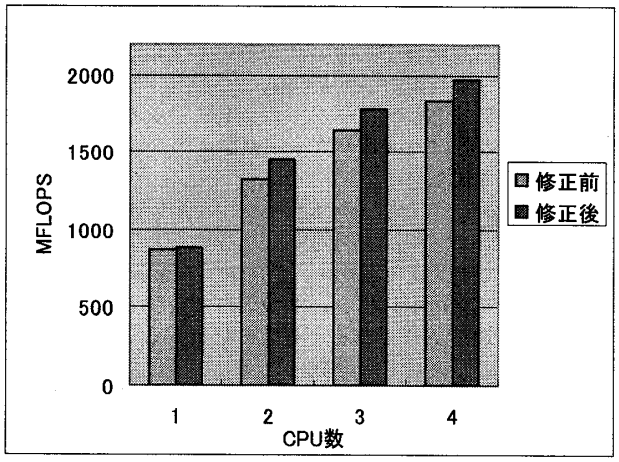

図３　姫野ベンチの測定結果

5. まとめ

本研究では Beowulf 型の分散メモリ型クラスタで姫野ベンチを実行して、MPI のメッセージをトレースし並列処理に掛かる時間を解析し、同期と通信に焦点を当て、処理フロー依存性を分析することで、MPI_Allreduce の実行回数を減らすことでメッセージ通信のオーバーヘッドを少なくすることで姫野ベンチの測定結果が約 8%向上することが確認できた。

並列処理の性能は通信で決まることから、オーバーヘッドの要因とその要因がプロセッサ数や計算性能、通信速度等にどのように関係しているかをメッセージの振舞いの観測と分析を行い明らかにすれば性能改善の手がかりが得られると考えられる。

参考文献

[1] トーマス・L・スターリング,“ PC クラスタ構築法 Linux によるベオウルフ・システム”
[2] William Gropp, Ewing Lusk, Thomas Sterling, “Beowulf cluster computing with Linux”.
[3] Nicholas Carriero, David Gelernter, “ 並列プログラムの作り方”
[4] Message Passing Interface Forum “MPI: A Message-Passing Interface Standard” http://www.mpi-forum.org/
[5] 姫野ベンチ http://accc.riken.jp/HPC/HimenoBMT/index.html
[6] 益口摩紀, 建部修見, 佐藤三久,関口智嗣,長嶋雲兵,“ 並列システム性能の視覚的解析とその評価” 情報処理学会研究報告.96－HPC-62（1999）
[7] 渡部善隆,南里豪志,藤野清次“ Himeno BMT によるハイパフォーマンスコンピュータの性能評価” 情報処理学会研究報告 2003－HPC-95（2003）
[8] 大森健児,“ 並列プログラミングの基礎 マルチプロセッサを指向した応用プログラム構成法