課題名（タイトル）：

大規模並列アプリケーションの通信性能解析および高速化に関する研究

利用者氏名：　南里　豪志
所属　　：　本所　情報基盤センター

報告内容

１．　本課題の研究の背景、目的、関係するプロジェクトとの関係

本研究では，バッチ型で利用する共同利用の大規模並列計算機において，出来るだけ安定した並列処理性能を得るための技術について効果を検証する．近年，パーソナルコンピュータの普及，及び性能向上によって対話的に利用可能な計算能力は大幅に増大したが，最先端の科学技術計算おける中～大規模計算では，依然として複数の利用者がバッチ型で共同利用する大規模な計算機が必要である．現在利用されている大規模な計算機の構成としては，複数の計算ノードで構成された分散メモリ型の並列計算機が最も一般的である．この並列計算機をバッチ型で利用する場合，申請されたジョブを計算ノードに割り当てるジョブスケジューラの設定が，計算機の総合的な処理速度，及び個々のジョブにおけるプログラムの実行速度に大きく影響する．例えば，RICC において用いられているメタスケジューラは，出来るだけ空いている計算ノードが少なくなるようにジョブを割り当てて，システムにおけるジョブの充填率を上げることにより，総合的な処理速度の向上を図っている．

一方，個々のジョブにおけるプログラムの実行速度は，特に複数の計算ノードを利用する並列計算では，計算ノード間の通信性能に依存する．さらにこの通信性能は，通信する計算ノード同士の，ネットワーク上での距離による影響が大きい．計算ノード同士の距離が近い場合はほとんど性能が変動する要素が無いが，距離が遠い場合，到達するまでの通信遅延時間が長くなるだけでなく，他の通信と経路が競合することによる通信性能の大幅な低下が発生しやすくなる．

特に，並列プログラムにおける全プロセスによる総和の計算や全対全のコピー等の，集団通信と呼ばれる通信では，この転送性能の低下が大きな問題となる．集団通信には複数の実装アルゴリズムが用意されており，状況に応じて最適なアルゴリズムを選択して使用する．従来は，事前に十分な調査を行うことにより，使用するプロセス数と転送するメッセージサイズから最適なアルゴリズムを選択している．しかし通信性能が安定しない環境では，同じプロセス数，メッセージサイズでも事前の調査時とは最適なアルゴリズムが変化する可能性が高い．そのため，実行時の状況に応じて適応的にアルゴリズムを選択する仕組みが必要となる．

そこで本研究では，集団通信の実装アルゴリズムを実行時の状況に応じて選択する技術を有するソフトウェア STAR-MPI を用いることで，ジョブの充填率を重視したスケジューラによる並列計算機でも最適なアルゴリズムを選択できると予想し，実験によって効果を検証する．

STAR-MPI は，MPI を用いた並列プログラムにおいて，集団通信のアルゴリズムを実行時に自動選択する技術である．MPI とは，事実上の世界標準として広く用いられている並列プログラミングインタフェースであり，STAR-MPI は，この MPI の上に構築されている．利用者は，プログラム中の MPI の集団通信関数名を STAR-MPI における集団通信関数名に置き換えた上で，STAR-MPI のソースプログラムと自分のプログラムを通常の MPI プログラムと同様にコンパイルし，リンカで結合することにより,STAR-MPI の機能を利用できる．

STAR-MPI の処理は，以下の 2 つのフェーズに分けて行われる．

Learning フェーズ：

集団通信が呼ばれると，利用可能なアルゴリズムのうちの一つを用いて実行し，結果を返す．その際所要時間を計測し，記録する．全アルゴリズム

について規定回数の計測が終了するまで，各集団通信呼び出しに対してこのフェーズを実行する．全アルゴリズムの計測が完了すると，それらの所要時間の記録から最も高速なアルゴリズムを選出し，次の Monitoring フェーズで使用するアルゴリズムとする．なお，アルゴリズムの選出に用いる各アルゴリズムの所要時間としては，全プロセスの所要時間の平均値を用いる．

Monitoring フェーズ：
選出されたアルゴリズムを用いて集団通信を行う．このフェーズは Learning フェーズが終了してアルゴリズムが選択された後，各集団呼び出しに対して実行する．
実際の計算環境では実行中に状況が大きく変化することも考えられるため，定期的に集団通信の所要時間と Learning フェーズで得られた所要時間を比較する．もし，計測した時間が Learning フェーズ時の所要時間から大きく変動した場合，他のアルゴリズムの方が高速となった可能性があるので，再度 Learning フェーズに入り，アルゴリズムを選択する．

STAR-MPI には,選択対象の各アルゴリズムが MPI により実装されている．例えば Alltoall 通信のアルゴリズムとしては，初期設定時に Simple Spread(以下 Simple と表記)，Pairwise(Pair)，Pairwise with Light-Barrier(PairLB)，Pairwise with MPI-Barrier(PairMB)，Pairwise with One-Barrier(PairOB)，Ring(Ring)，Ring with Light-Barrier(RingLB)，
Ring with MPI-Barrier(RingMB)，Ring with One-Barrier(RingOB)から選択されるようになっている．さらに STAR-MPI は，実行環境における MPI ライブラリの MPI¥_Alltoall 関数も，選択肢として用いる．

2． 具体的な利用内容、計算方法

RICC における STAR-MPI の効果を検証するため，Alltoall 通信を 200 回呼び出すプログラムを用いて所要時間の計測を行った．このプログラムは STAR-MPI のベンチマークプログラムとして提供されているものである．
このプログラムを，プロセス数とメッセージサイズを変えながら RICC のメタジョブスケジューラに投入した．それぞれのプロセス数とメッセージサイズの組み合わせについて 10 回ずつジョブを投入し，回毎の各アルゴリズムの所要時間，選択されたアルゴリズム，全体的な平均所要時間，及びオーバーヘッドについて計測した．
RICC では計算ノードの単位でジョブを割り当てる．1 台の計算ノードには 8CPU コアが搭載されているので，例えば 64 プロセス時で 8 台，128 プロセス時で 16 台の計算ノードが割り当てられる．
なお，RICC のメタスケジューラには，16 台以下の計算ノードを使用するジョブについて，全部の計算ノードが 1 台のリーフスイッチに納まるような割り当てを指示する機能がある．この機能を使った場合，計算ノードが割り当てられるまでの待ち時間が長くなるが，計算ノードの相対的な位置が毎回ほとんど同じであるため，通信性能の変動が少ないと予想される．そこで，この機能を使った場合と使わなかった場合で計測結果を比較する．

3． 結果

128 プロセス，メッセージサイズ 1MB で，使用する全計算ノードを同じリーフスイッチに配置するよう指定した場合の結果を図 1 に，その指示を行わなかった場合の結果を図 2 に，それぞれ示す．図の横軸は投入したジョブの番号，縦軸は平均所要時間である．計測結果としては，最初の Learning フェーズにおける各アルゴリズムの平均所要時間(Pair～RingOB),MPI の MPI_Alltoall 関数の平均所要時間(MPI)，及び STAR-MPI の Monitoring フェーズにおける平均所要時間(STAR-MPI)である．
STAR-MPI は，最初の Learning フェーズで最も速かったアルゴリズムを選択するので，基本的に毎回，図中の各アルゴリズムのうち一番下のもので Monitoring フェーズを実行する．ただし，

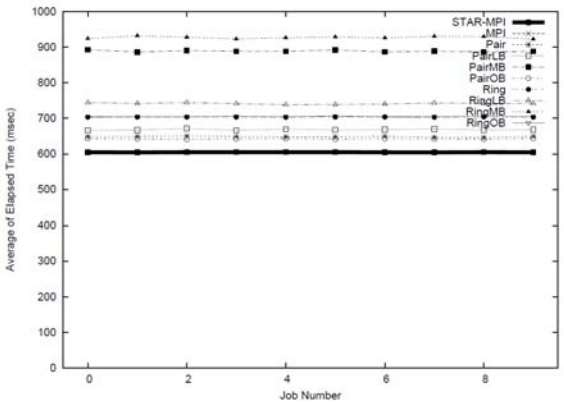

図 1　同一リーフスイッチ内のノードに割り当てた場合の性能（128 プロセス, 1MB）

Monitoring によるオーバヘッドや実行中の性能の変動，さらに，実行途中での再度の Learning フェーズによるアルゴリズム切り替え等により，選択したアルゴリズムの所要時間と STAR-MPI の所要時間は一致しない.

全計算ノードが同一スイッチに配置された場合，各アルゴリズムの性能は毎回ほとんど変動が無い．そのため STAR-MPI も，毎回ほとんど同じ選択をしている.

例えば図 2 では，MPI_Alltoall 関数を最適アルゴリズムとして選出している．このような状況であれば，STAR-MPI を利用しなくても，事前に十分な調査をすればプロセス数とメッセージサイズだけで最適なアルゴリズムを選択することができると考えられる.

一方，同一スイッチへの配置を指示しなかった場合，計算ノードが複数のスイッチに跨って配置されるため，通信時間が大きく変動する．その結果，図 1 に示す通り，各アルゴリズムの性能も大きく変動している.

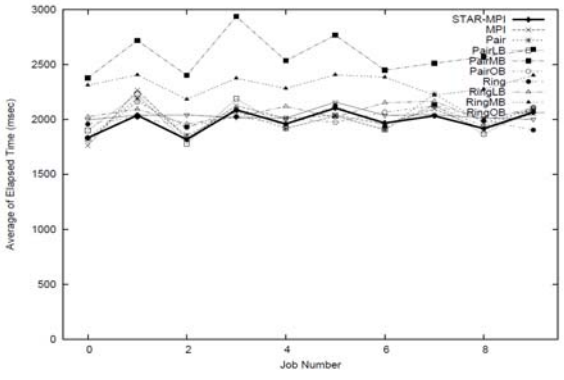

図 2　ノードの位置を考慮しないスケジューリングによる性能（128 プロセス, 1MB）

例えば Job Number 0 で最速だった MPI_Alltoall 関数は，Job Number 1 では最速アルゴリズム(Ring)よりも 10%程度遅くなっている．このように毎回通信性能が変動する場合，従来のプロセス数とメッセージサイズだけでは最適なアルゴリズムを選択できず，STAR-MPI のような動的な手法が必要となる.

4．まとめ

本稿では，充填率を重視してランク配置を行うジョブスケジューラを用いた環境で，STAR-MPI を用いることにより，通信性能の変動によらず最適なアルゴリズムが選択されることを確認した.

5．今後の計画・展望

今回の研究で試した STAR-MPI は，遅いアルゴリズムであっても実行中に性能を計測する必要があり、これが実行時のオーバヘッドとなる．集団通信では，あるアルゴリズムがほかのアルゴリズムに対して数倍遅い場合もあり，その場合、このオーバヘッドが無視できなくなる.

これに対して申請者らは，性能モデルによる各アルゴリズムの性能予測の結果を用いて，非常に遅いと予測されるアルゴリズムを計測対象から除外することによってオーバヘッドを低減する技術を提案している.

今後，RICC のメタスケジューラ上でこの技術を適用した場合の効果について検証を行う.

また，Alltoall 以外の集団通信における効果についても確認するとともに，集団通信を用いている科学技術アプリケーションプログラムでも実験を行い，実用性を検証する。

6．RICC の継続利用を希望の場合は、これまで利用した状況（どの程度研究が進んだか、研究においてどこまで計算出来て、何が出来ていないか）や、継続して利用する際に行う具体的な内容

今年度は，RICC のように積極的に空きノードを

活用してスループットを向上させるスケジューラでジョブ管理される並列環境における集団通信高速化の手段として，既存の動的アルゴリズム選択技術 STAR-MPI の効果を計測し，ある程度の実用性を確認した．今後，動的アルゴリズム選択におけるオーバヘッドの低減技術の適用と評価，及びこのような動的最適化技術の実アプリケーションでの効果に関する検証が必要である．

７． 一般利用で演算時間を使い切れなかった理由

８． 利用研究成果が無かった場合の理由

平成 22 年度　RICC 利用研究成果リスト

【国際会議などの予稿集、proceeding】

Y. Morie, T. Nanri and M. Kurokawa, Task Allocation Method for Avoiding Contentions by the Information of Concurrent Communications, in Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Networks, Feb. 2011, Insbruck, Austria.

【その他】

南里　豪志，黒川　原佳，充填率を考慮したジョブスケジューラによるランク配置に対する集団通信アルゴリズム自動選択技術の効果，日本応用数理学会環瀬戸内応用数理研究部会 第 14 回シンポジウム，Jan. 2011.