特集 科学技術計算におけるソフトウェア自動チューニング

< ソフトウェア自動チューニング技術の応用 >

9 MPI 通信ライブラリの自動チューニング

今村　俊幸　電気通信大学

MPI チューニング概要

PC クラスタが誰にでも簡単に導入できるようになり，並列計算は珍しくなくなってきた．MPI（Message Passing Interface[1]）は分散メモリ型計算機におけるメッセージパッシングモデルの仕様であり，1994 年の第 1 版の規格発表から 10 年以上が経ち，すでにメッセージパッシングの標準規格の地位を確立している．MPI は並列プログラミングに欠かすことのできないプログラミングツールの 1 つである．

並列プログラミングをする際，わずかな非並列化部分や並列化オーバヘッドの影響により性能向上の上限が定まってしまう問題が存在する．プログラム作成者はできる限り通信オーバヘッドを削減する努力を行うが，通信ライブラリ自身の性能は開発者任せの部分が大きい．したがって，できるだけ高速な MPI 実装を利用したいと考える．実際，実行性能にシビアな計算機センタなどではネットワークデバイスやシステム規模に合わせた最適な実装系を選択し導入している．

1 万プロセスを超えるような次世代スパコン（所謂ペタコン）環境は，多くのプログラム開発者には未経験のプロセス数であり，正しく動作するプログラムすら開発困難となることが予想される．また，ペタコンでは，チップ上に集積されているコア数が増加し，レジスタから主記憶までの階層構造が深くなるに伴い，各プロセスが所有するデータ間の距離が大きくなり通信の遅延が増大する．プロセッサ性能やネットワーク帯域の強化は不可欠ではあるが，並列化の副作用でもある通信オーバヘッドの削減にもっと目を向けなくてはならない．国内にも数千〜数十万規模の高速計算ノードを相互結合するインターコネクト技術に関するプロジェクト[2]において，本課題の重要性が認識されている．プログラム開発者にのみ負担を強いることなく，MPI ライブラリ自体が性能チューニングの機能を備えることによりプログラム開発者は半自動的に性能チューニングが行えるようになり，高性能な並列プログラム開発の負担は軽減されるだろう．本稿では，MPI_Bcast 関数を例にとり MPI の性能チューニングの一例を説明したい．

集団通信のチューニング

MPI の通信は大きく 1 対 1 通信（point-to-point communication）と集団通信（collective communication）に分けられる．前者は，文字通り起点（Sender）と終点（Receiver）のプロセスを指定する通信で MPI の基本である．

一方，後者の集団通信は複数プロセスからなるグループで行われる特定の意味を持った通信形態であり，1 対 1 通信のみでプログラムを書いた場合と比較して複数の通信を抽象化して扱えるため可読性や保守性を高めることにつながる．代表的なものとして，プロセスの同期制御を行うバリア同期（MPI_Barrier）やブロードキャスト（MPI_Bcast），プロセス間に分散するデータに 2 項演算（たとえば総和計算など）を行うリダクション計算（MPI_Reduce）が挙げられる．

集団通信のアルゴリズムは Vadhiyar[3] や Faraj[4] の論文によくまとめられている．1 つの集団通信には多くのアルゴリズムが存在しており，MPI 実装はこの中のいずれかを使用している．集団通信のコストは，（アルゴリズム，参加プロセス数，メッセージサイズ）の 3 つで決定される場合が多く，適切な「アルゴリズム」を選択することで，通信性能が大きく変わる．そのため，集団通信部分にチューニングの努力を注ぐ MPI 実装系が多く存在する．

集団通信の内部通信には 1 対 1 通信が利用され，多くの場合，受信したメッセージを直ちに送信することが多い．送受信が同時に可能なネットワークを使用している場合は，メッセージを分割して（分割したメッセージの長さを「セグメント」と呼ぶ），受信済みメッセージの送

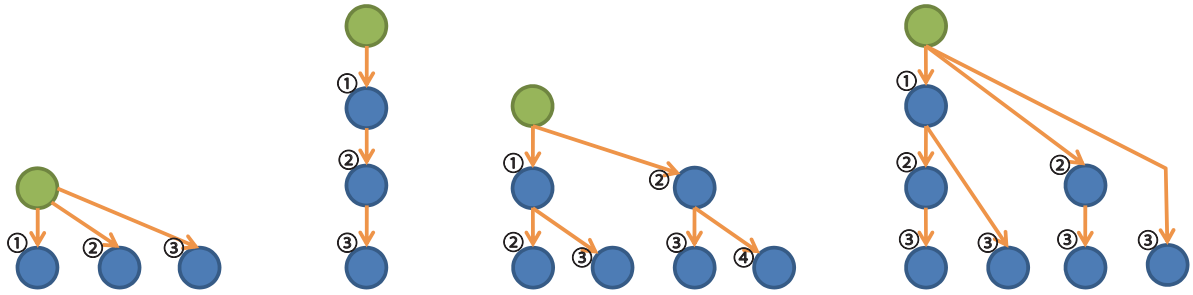

図 -1　各アルゴリズムでのデータの流れ（左から , Sequential, Chain, Binary, Binomial．緑色の丸がルートプロセスを表し，矢印に付した番号順にメッセージを送信する）

信操作と次のメッセージ受信操作を同時進行させることができる．このような送受信方式を「パイプライン方式」と呼び，ネットワークを最大限利用しつつ通信オーバヘッドの削減を行うことができる．

このようにアルゴリズムを含めたさまざまなパラメタ調整が集団通信のチューニングということになる．

MPI_Bcast

代表的なブロードキャストアルゴリズム「Sequential」，「Chain」，「Binary」，「Binomial」の 4 種類は，図 -1 に示すような接続トポロジを持つ．ルートと呼ばれるプロセス（図中の緑色の○）から矢印で接続された 1 つの子プロセス（青色の○）に送信する．さらに矢印を持つプロセスは同様に送信を続ける．1 度に 1 回の送受信のみが行われると仮定すると，図 -1 の矢印に付した番号のタイミングで送信がされる．全参加プロセスへのデータ送受信が終了するまでの時間を MPI_Bcast のコストとすれば，これらのアルゴリズムのコストは以下のように計算される．ここで，メッセージ長 x の 1 対 1 通信のコストが $\alpha+\beta x$ の線形関数で表現されるものと仮定する．また，集団通信に参加するプロセス数を P，メッセージ長を X と示した．

$$T_{Sequential}=(P-1)(\alpha+\beta x)$$
$$T_{Chain}=(P-1)(\alpha+\beta x)$$
$$T_{Binary}=2(\lceil \log_2(P+1)\rceil-1)(\alpha+\beta x)$$
$$T_{Binomial}=\lceil \log_2 P\rceil(\alpha+\beta x)$$

α，β が定数であると仮定すれば，上式から通信コストを最小にする適切なアルゴリズムが選択できる．しかしながら，α，β はプロセスの配置やネットワークスイッチなどの状況によって変動する．つまり，アルゴリズムごとに定まる関数 $\alpha(P, X)$, $\beta(P, X)$ となる．サンプリング等によってこれらを推定する必要があるが，α，β など項別の推定は困難となる．こういった場合は，定数と見なした α，β による 1 次推定を行った上で，高精度な推定には関数によらない方法を用いるのが適当である．実際は，サンプリングによってコスト関数の概形を決め，統計的な手法によりコスト推定を行う．

OpenMPI でのチューニングの試み

MPI 実装系の中で精力的にチューニング機構を導入しているのは，OpenMPI であろう [5]．OpenMPI は MCA（Modular Component Architecture）に基づいて開発されており，個々の集団通信関数モジュールを動的に選択できる設計となっており，その設計に基づき集団通信関数に動的なアルゴリズム決定機構を提供している．

以下，OpenMPI version 1.3 に基づいて解説する．OpenMPI の集団通信関数はソースファイル ompi/mca/coll に収納されている．そのディレクトリを見れば basic, sm, tuned などが存在する．basic は基本的な線形アルゴリズムと対数アルゴリズムを，sm は共有メモリ向けの実装，tuned はその他各種アルゴリズムの実装が納められている．MPI_Bcast には次の 6 種類が実装されている．1, 2, 5, 6 がそれぞれ図 -1 の Sequential，Chain, Binary, Binomial に対応している．

1. basic_linear
2. chain
3. pipeline ☆1
4. split_bintree ☆2

☆1 pipeline : OpenMPI での pipeline アルゴリズムは chain においてメッセージを分割し，送受信を同時に実施するようにしたアルゴリズムを指す．

☆2 split_bintree : split_bintree は bintree の一種である．ルートに接続する 2 つの 2 分木にメッセージの前半，後半の別々を担当させ，最後に前後半をマージするアルゴリズムである．

```
% mpirun -np 19 -mca coll_tuned_use_dynamic_rules 1 -mca coll_tuned_bcast_algorithm 6 –mca coll_tuned_bcast_segmentsize 4096 ./a.out
```

図 -2　OpenMPI における集団通信アルゴリズム指定の例（例はアルゴリズム 6(binomial) をセグメントサイズ 4096 バイトで実行するよう指定している）

```
# MPI_Bcast 用の rule file (# 以下はコメント )
1   # ルール設定対象の集団通信の個数
7   # MPI_Bcast の ID (coll_tuned.h 内の enum COLLTYPE_T の定義による )
1   # 設定対象のコミュニケータパターンの数
32  # 以下コミュニケータサイズが 32 のパターンでの設定
5   # メッセージサイズによる場合分けの個数
# 以下の行は 4 フィールドからなる ( コンマではなくスペースがセパレータ )
#   メッセージサイズ下限 ( バイト ), アルゴリズム , faninout パラメタ ( 今回は使用せず ), セグメントサイズ
      0   6   0   1024 #    0  <= msg.  < 8kB の場合はアルゴリズム 6, セグメントサイズ 1024
   8000   6   0   2048 #    8  <= msg.  < 16kB 〃       アルゴリズム 6, セグメントサイズ 2048
  16000   6   0   4096 #   16  <= msg.  < 40kB 〃       アルゴリズム 6, セグメントサイズ 4096
  40000   5   0   4096 #   40  <= msg.  < 96kB 〃       アルゴリズム 5, セグメントサイズ 4096
  96000   5   0   8192 #   96  <= msg.       〃       アルゴリズム 5, セグメントサイズ 8192
```

図 -3　測定から作成した MPI_Bcast 関数の最適パラメタ指定のルール一例

5. bintree
6. binomial

Open MPI ではこれらアルゴリズムの選択方法として，1）based rules, 2）forced rules, 3）fixed rule set のいずれかが利用できる．1）は各種パラメタに応じてどのアルゴリズムを選択するかのルールをファイルにより設定するものである．2）は単にアルゴリズムを指定するもので，**図 -2** のようにコマンドラインで MCA へのオプション指定ができる．3)はデフォルトの選択ルール関数に従うというものである．

次に示すような PC クラスタにおいて，flat-MPI[☆3] によってノード数以上のプロセスによる並列処理を想定する．そして，MPI_Bcast 関数の自動チューニングを行ってみよう．

Intel Xeon X3330（Quad core）× 8 ノード
Gigabit Ether（BCM95722）
OS : Fedora9（kernel 2.6.27.9-73）

i）まず，メッセージサイズ 1000 から 10000 まで 1000 刻みで 100 試行分の平均実行時間から各アルゴリズムそれぞれに適切なセグメントサイズを決定する．ただし，セグメントサイズは 2^iK バイト（i=0, 1, …, 8）に限定する．

ii）次に，それぞれのメッセージサイズで通信コストを最小化する（アルゴリズム，セグメントサイズ）の組を決定する．

iii）ii）の結果をもとに，**図 -3** のようなルールファイルを自動生成する（ただし，実際に生成されるルールはセグメントサイズが激しく変動するが，図 -3 は簡単化したものである）．このルールファイルをコマンドラインオプション -mca coll_tuned_dynamic_rules_filename によって指定することで，最適なアルゴリズムが選択されるようになる．

この i) から iii) の操作を自動で行うことで MPI_Bcast の自動チューニングが達成される．OpenMPI の現バージョンにはこのような自動的なチューニングの機能はなく，今回は簡単なシェルスクリプトを作成して自動チューニングを行った．

図 -4 のグラフは，各種アルゴリズムとチューニングを施した MPI_Bcast の実行時間を示したものである．チューニングによりデフォルトの 3 分の 2 まで通信コストが削減された．

グラフから分かるように測定範囲では，デフォルトは split_bintree を利用している．実際，ソースコードを見ると，2048 バイト以下は binomial を選択し，さらに 370728 バイト以下でセグメントサイズ 1KB の split_bintree を選択するよう書かれている．

☆3 flat-MPI：flat-MPI は PC クラスタなどですべての MPI プロセスがシングルスレッドで実行されるモデルである．基本的に並列性が MPI のみの 1 階層で記述される（インストラクションレベルや SIMD レベルの並列性は除く）．

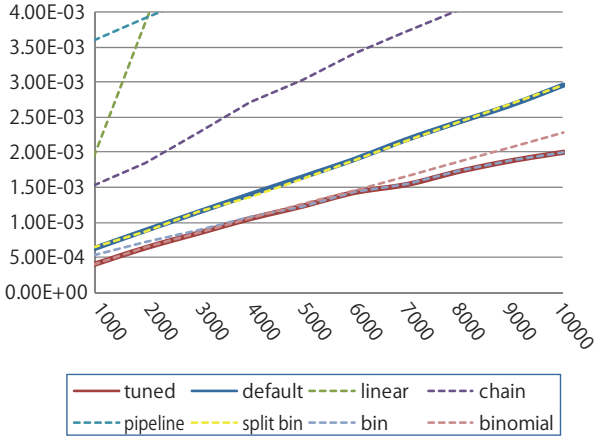

図 -4　P=32 の場合の MPI_Bcast の平均実行時間 [秒]（横軸はメッセージサイズ）

A. インストール時に，最速と考えられるアルゴリズム，送受信方式，機構を固定する．
B. インストール時に，適当なサンプリングを行い最良の組合せを利用する．
C. インストール時に，考えられるパラメタの全探索もしくは過去の探索結果から候補探索を行い，それを反復し精度を上げ最良のものを利用する．
D. インストール時には上記のものを利用し，実行と同時にパラメタ探索を実施し，動的に変更を行う．

図 -5　タイミングで分類したチューニング戦略

OpenMPI のデフォルトのアルゴリズム選択は図 -5 に示す A のチューニング戦略であり，環境に即した高速化ができない場合がある．一方，ここで紹介した方法は B のチューニング戦略である．デフォルトよりも明らかに正確となり，より環境に即したものとなる．一方，サンプリングの網にかからないパラメタが存在するため，ギリギリまでチューニングを必要とする場合は C の戦略を行わなくてはならない．しかしながら，A ＜ B ＜ C の順にサンプリングのコストが必要となるので，どの戦略を選択するかはコストパフォーマンス次第であろう．

また，実際のアプリケーションでは使用する範囲で最大の性能が発揮されることが望ましい．プログラムの実行中に（または実行するたびに）サンプリングを同時に行い，ピンポイントで推定精度を高めるといった D の戦略がより重要となる場合もある．

OpenMPI では，プロファイリング情報の動的フィードバックを可能とする各種インフラはすでに整備されている．（メッセージサイズ，プロセッサ数）に応じた最良パラメタへの対応関係（文献 6 では「マップ」と呼ばれている）を実際の並列プログラムを動作させながら精密にしていくことで，実行時環境に即したアルゴリズム推定の精度を上げることができる．研究レベルではあるが，このような動的なアルゴリズム選択機構の開発が進んでおり，公開版への取り込みが期待される．

新たな方向性について

自動チューニングによる通信オーバヘッドを削減するアプローチは，今後の大規模なマルチコア・マルチプロセッサ計算機では必須のものと考えられる．ダイヤルをひねるイメージで性能を調整することができれば，ペタスケールの高性能計算機からその先のエクサスケール計算機までの通信最適化の展望が見えてくる可能性がある．現在の実装系にはすでにチューニングのためのインフラが整備されており，将来的にはその機構が威力を発揮するときが来るであろう．

また，本稿では説明ができなかったが，コンパイラによりプログラム中の MPI 関数呼び出しの意味を解析し，ノンブロッキング関数への書き換えや，集団通信を 1 対 1 通信に置き換えるなどのプログラム変換を行う研究が進んでいる [7]．これにより，通信ライブラリ単体のチューニングにとどまらずプログラム全体で通信コストを削減するチューニング手法の確立がなされるものと期待される．

参考文献

1) MPI Forum, MPI : A Message-Passing Interface Standard, version MPI-2.1. http://www.mpi-forum.org/docs/mpi21-report.pdf (2008).
2) PSI（Petascale System Interconnect）プロジェクト，http://www.psi-project.jp/
3) Vadhiyar, S. S., Fagg, G. E. and Dongarra, J. J. : Towards an Accurate Model for Collective Communications, Intl. J. of High Performance Cumputing Applications, Vol.18, No.1, pp.159-167 (2004).
4) Faraj, A. and Yuan, X. : Automatic Generation and Tuning of MPI Collective Communication Routine, proceedings of ICS'05 (2005).
5) Open MPI, http://www.openmpi.org/
6) Grbovic, J. P., Fagg, G. E., et al. : Decision Trees and MPI Collective Algorithm Selection Problem, Proceedings of Euro-Par2007 Parallel Processing, LNCS 4641, pp.107-117 (2007).
7) Danalis, A., Pollock, L. and Swany, M. : Automatic MPI Application Transformation with ASPhALT, Proceedings of IPDPS 2007, pp.1-8 (2007).

（平成 21 年 3 月 26 日受付）

今村　俊幸（正会員）　imamura@im.uec.ac.jp

電気通信大学准教授．HPC，特に数値計算ライブラリの自動チューニングの研究に従事．博士（工学）．平成 11 年日本応用数理学会論文賞，平成 18 年度本会山下記念研究賞，2005, 2006 両年 Gordon Bell Finalist. 日本応用数理学会，SIAM 各会員．