MPE による MPI 通信の可視化

松葉　浩也

東京大学情報基盤センター

１．はじめに

新スパコンである「T2K オープンスパコン（東大)」(本稿では各ノードを構成するマシンの製品名を取って「HA8000 クラスタシステム」と呼びます）は汎用部品から成るコンピュータ（ノード）を大量に使用することで高い性能を得るシステムです。各ノードの理論的な演算性能は SR11000 と同じですが、メモリバンド幅やキャッシュ容量などを加味した総合的なノード性能は SR11000 には及びません。その代わりに HA8000 クラスタシステムでは SR11000 よりも多くのプロセッサを使用することで性能を確保します。このようなシステムの性質を考慮して、HA8000 クラスタシステムでは並列化されていないアプリケーションやノード内並列化のみを行ったアプリケーションを実行するためのコースは準備しません。つまり、HA8000 クラスタシステムで実行するアプリケーションは MPI によって並列化されていることが必須となります。

このように HA8000 クラスタシステムでは MPI による通信が大変重要であるため、本稿では MPI のチューニングやデバッグに有効な通信の可視化システムの使い方を紹介します。本稿で紹介する可視化システムは MPE と呼ばれるツールであり、MPI を提案した米国アルゴンヌ国立研究所が作成したものです。MPI 通信ライブラリとして最も一般的な MPICH（HA8000 クラスタシステムでも使用します）と共に配布されており、MPICH をインストールしたシステムには通常は MPE も自動的にインストールされています。したがって、MPE は本センターの HA8000 クラスタシステムだけでなく、MPICH を使用するほとんどの並列計算機で使用できる汎用性の高いツールです。

２．MPE の使い方

通信の可視化は、ログ採取用のライブラリをリンクした MPI プログラムを実行して通信ログを採取し、そのログを可視化ソフトで確認するという手順になります。

２.１　可視化ライブラリのリンクと実行

MPE は MPI 通信関数に特殊なラッパ関数を挿入して通信ログを採取します。そのラッパ関数が書かれたライブラリを使用するためには、アプリケーションの再コンパイル（正確には再リンク）が必要です。MPE にはいくつかの種類のライブラリがあり、それらを選択して正しくリンクするのは少々複雑ですが、MPICH と共に使用する場合には mpif77 や mpicc などのコンパイル、リンク用スクリプトに MPE 用のオプションが準備されているため簡単に使用することができます。

通信の可視化のために使用するオプションは -mpilog です。次ページの例のように MPI プログラムをコンパイル、リンクしてください。ソースの変更は不要です。

（例）

```
$ mpif77 -mpilog -O3 program.f -o program
$ mpicc -mpilog -O3 program.c -o program
```

大規模なプログラムで Makefile を使用している場合、リンカに渡すオプションに -mpilog が入るように Makefile を修正してください（コンパイル時に付けても無害なのでコンパイラとリンカのコマンドを”mpif77 -mpilog”などにしてしまうのが最も簡単かもしれません)。

こうして作成した実行可能プログラムを通常の方法でバッチジョブシステムに投入し、実行します。実行に際しては特殊なオプションは不要です。終了後、実行ファイルと同じディレクトリに「実行ファイル名.clog」というファイルができていれば成功です。

＜注意＞
プログラムを単に -mpilog オプションでコンパイルした場合､すべての通信のログが採取されます｡一秒程度で終了する小さなプログラムでも 100MB 近いログになる可能性があります。最初に試すときは必ず小規模なプログラムで様子を見るようにして下さい。

2.2 可視化

採取したログを確認するためには、最初にログファイル形式の変換が必要です。

```
$ clogTOslog2 program.clog
```

これを実行すると 「プログラム名.slog2」というログファイルができます。これで準備が整ったので可視化ソフトを起動します (GUI プログラムなので X11 転送が有効になっている必要があります)。

```
$ jumpshot program.slog2
```

起動すると図 1 のような通信の状況を図示したウィンドウが開きます。図は CG 法のベンチマークプログラムの通信ログを可視化したものです。初期状態の画面ではログを採取した期間全体が表示されており、個々の通信は時間が短すぎて見えない状況です。この画面での矢印はたくさんの通信が発生していることを適当に図示しているのみで、これらはあまり有用な情報ではありません。有用な情報を得るためにはこのウィンドウの中から通信の状況を詳細に確認したい区間を選択します(マウスで始点から終点までドラッグする)。この状態が図 2 です。白い矢印が一回の通信を表します。矢印のない黒い部分が通信のない計算時間帯ということになります。白黒の本稿では見づらいですが、実際の画面では赤色で通信待ちの時間帯が目立つように表示されています。図 2 で示されているのは通信が効率よく行われるようにうまく書かれたベンチマークプログラムの通信ログであるため無駄な通信待ちは見られませんが、並列化の際の負荷の分散が不適切な場合や通信量に偏りがある場合などには一部のプロセスに長い通信待ち時間が見られるはずです。

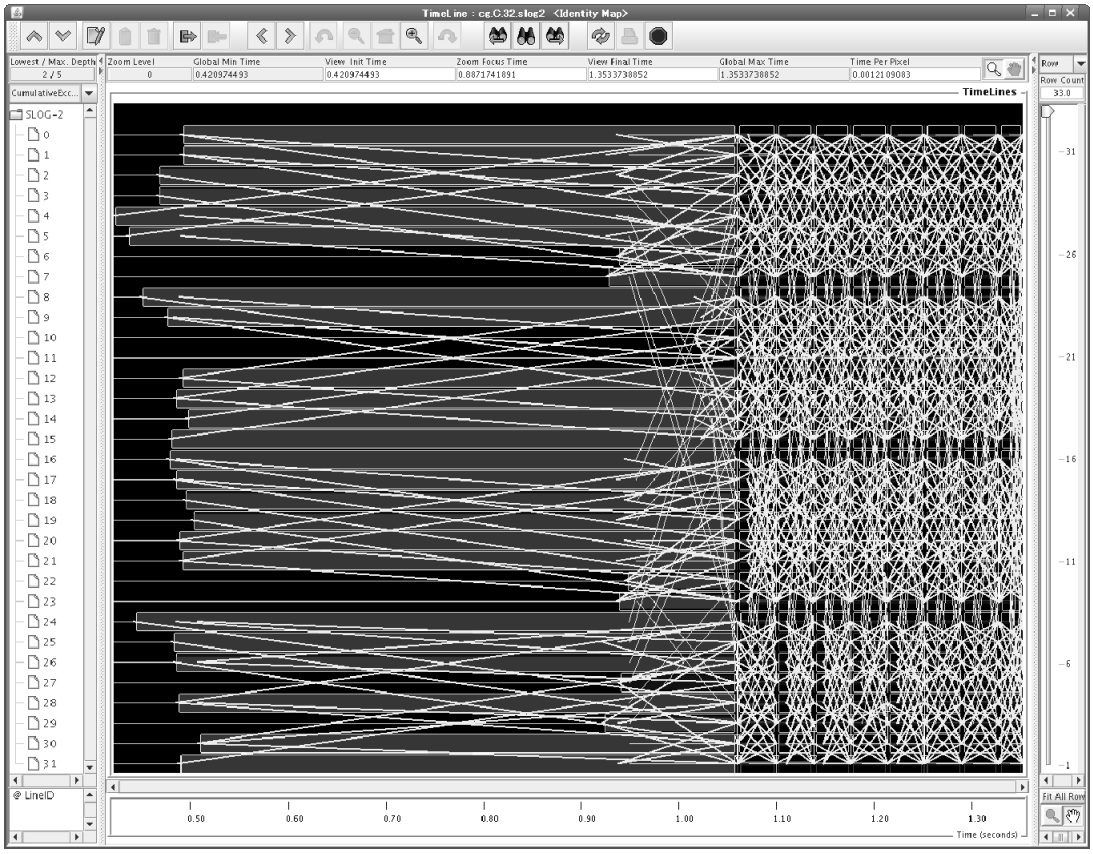

図 1 起動直後の画面

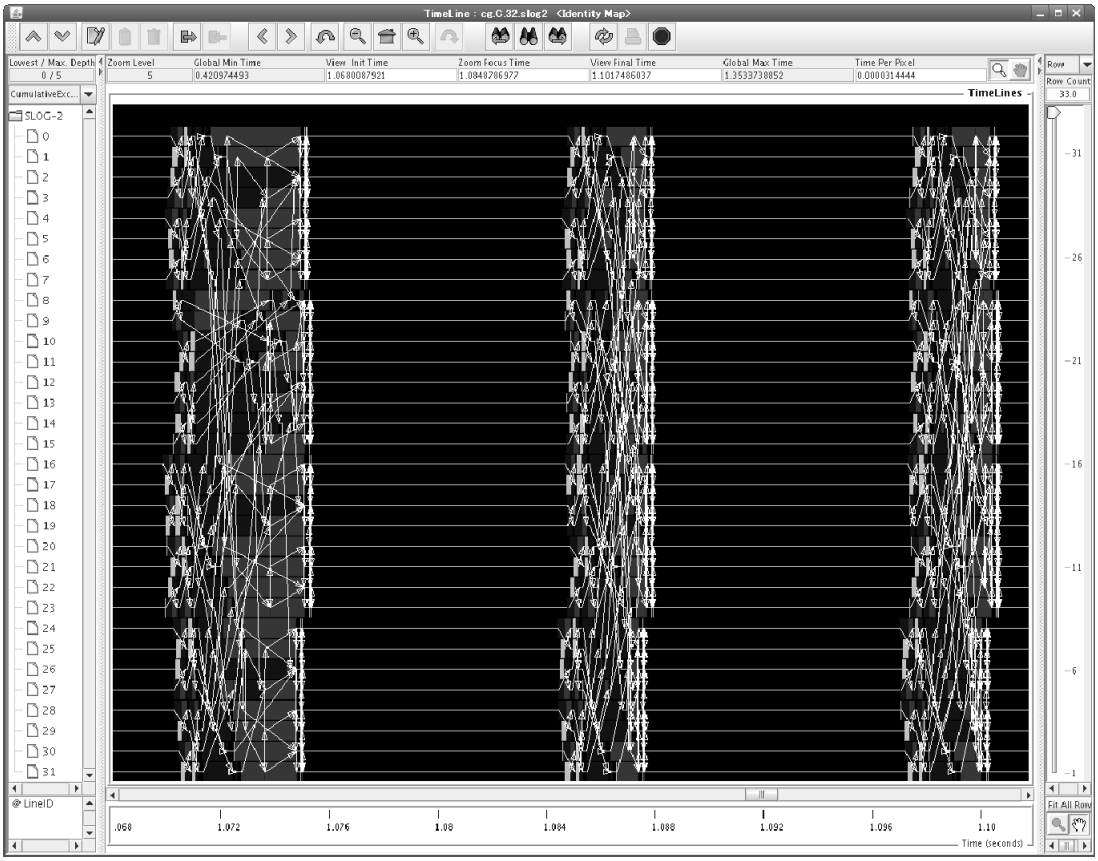

図 2 範囲選択後の画面

２.３　選択的なログの採取

ここまでで一通りの可視化の方法を紹介しましたが、ほとんど一瞬で終了するような小さなプログラムでも膨大な量のログになるため、実用上はチューニングに必要な部分に絞ってログを採取することが必須です。そのためにはプログラムに少し修正を加え、ログ採取の ON/OFF をライブラリに伝える必要があります。-mpilog オプションで作成したプログラムは MPI_Initialize 直後はログ採取が有効になっています。これを無効にするためには、プログラム中に以下の行を挿入します。

```
<FORTRAN>
   call mpi_pcontrol(0, ierr)
<C 言語>
   MPI_Pcontrol(0);
```

再びログ採取を有効にするには、上の例でゼロになっている引数を１にして同じ関数を呼び出します。これらを適宜プログラムの中に挿入することによって、必要な部分のみのログ採取が可能です。MPI_Pcontrol 関数は -mpilog オプションを付けない状態で呼び出しても無害なので、チューニング後の削除は必須ではありません。

３．MPE の仕組みとオーバヘッド

MPE は MPI の規格に含まれている profiling interface を使用しています。MPI の規格では MPI のライブラリ関数は weak symbol とすることが推奨されており、ユーザープログラムが独自の MPI_Send などの関数を定義した場合は、ユーザー定義のものが優先して呼び出されます。この場合、ユーザー定義の MPI 関数が本来の MPI 通信関数を呼び出す必要があることが多いですが、その場合には PMPI_Send など PMPI から始まる関数を使います。MPE もこれを使用しており、MPE のライブラリ中にログ採取機能を追加した MPI 関数群が定義されており、それらがログを採取した上で PMPI から始まる本来の通信関数を呼び出しています。

MPE のログは通常はメモリに書かれるため、通信のたびにディスクアクセスが発生するような大きなオーバヘッドはありません。前述した方法により、ログ採取を適切な範囲に絞っていれば、ログ採取による性能低下は無視できる程度に抑えられるものと思われます。

４．おわりに

通信は複数のノードにまたがって発生するイベントであるため、所要時間の計測やバグの発見などがやりづらいものですが、MPE を使えば実際に起こったことが簡単に視覚化できます。本稿では基本的な使い方のみを紹介しましたが、下記 URL より詳細な情報が入手できますので、より多くの機能を使用したい方はぜひご覧ください。

http://www-unix.mcs.anl.gov/perfvis/