

多倍長型 MPI プログラムの自動生成に向けた 多倍長型通信ライブラリの実装と評価

愛沢菜穂^{†1} 斯波柁^{†1} 菱沼利彰^{†2} 田中輝雄^{†1} 藤井昭宏^{†1} 平澤将一^{†3}
工学院大学^{†1} 筑波大学^{†2} 東北大学^{†3}

1 はじめに

大規模な計算では誤差の低減を図るため多倍長計算を用いることがある[1]. 我々は多倍長計算を手軽に扱うため, 倍精度計算を行うコードから GNU Multi-Precision Library (GMP) [2]を用いた多倍長計算を行うコードを自動生成する機構 Xev-GMP[3] [4]の開発を進めている.

本研究では Xev-GMP において MPI 通信を含むコードの変換の実現を目的とした. GMP の多倍長型は後述する理由により MPI での通信が困難であった. GMP の多倍長型の MPI 通信を行う手法を示した先行研究[5]があるが, 一度の通信のために複数の関数を呼ぶ必要があり MPI 関数の呼び出しと同等のインタフェースではない. 本研究ではそれらを MPI 関数と 1対1で対応するようにライブラリ化することで, 上記の目的の達成を図った.

2 Xev-GMP

GMP で扱う多倍長型である `mpf_t` 型は初期化, 宣言, 代入, 開放や演算の際に専用の関数を呼ぶ必要があり, 手書きでコードを書くにはコストが高い. これを解消する目的で開発を進めているのが Xev-GMP である.

Xev-GMP は C プログラムにおいて倍精度計算を `mpf_t` 型での計算に自動変換する機構である. ユーザーは各多倍長変数の精度を指定するための数行のディレクティブをコードに追記するのみでよい. 現在 Xev-GMP は MPI 通信を含むコードの変換が課題となっている.

3 `mpf_t` 型の通信

MPI 通信を含むコードの変換が課題となる理由

Implementation and Evaluation of Multiple-precision Communication Library toward Automatic Multiple-precision MPI Code Generation

Nao Aizawa^{†1}, Masaki Suwa^{†1}, Toshiaki Hishinuma^{†2}, Teruo Tanaka^{†1}, Akihiro Fujii^{†1}, Shoichi Hirasawa^{†3}

^{†1} Kogakuin University

^{†2} University of Tsukuba

^{†3} Tohoku University

として, `mpf_t` 型が構造体であることや, そのメンバに含まれるポインタが仮数部を指しており, 精度によってデータサイズが変わることが挙げられる.

幸谷らの MPIGMP Library [5]では以下の3つの手順でこの問題を解決している.

1. 新しく確保したバッファに対し, `mpf_t` 型内の必要な情報 (仮数部含む) を順にコピーする処理 (`pack`) を行う.
2. 集団通信の場合はバッファを派生データ型として定義したのち通信を行う.
1対1通信の場合はデータサイズを通信したのち, 実データの通信を `BYTE` 型として行う.
3. `pack` と逆の処理 (`unpack`) を行う.

MPIGMP Library ではこの手順に不可欠な `pack` や `unpack`, バッファサイズの取得等を行う関数が提供されている.

本研究では, 以上の手順を関数化し, MPI 関数と 1対1対応させる方針とした.

4 実装した関数と実装手法

MPI の基本的な関数に対応させるため, 各関数と同じ機能を果たす `mpf_t` 型通信用の関数を実装した. 今回は `Send`, `Recv`, `Bcast`, `Reduce`, `Allreduce`, `Gather`, `Scatter`, および `Alltoall` に対応させた.

例として `Xev_GMP_MPI_Send` という関数について述べる. この関数は `MPI_Send` で `mpf_t` 型の通信を行う. 内部では前章で述べた手順に加え, 受信側と送信側の精度が異なる場合の処理を行う. 引数は `MPI_Send` とほぼ同様である.

変換において Xev-GMP は倍精度型を引数にとる `MPI_Send` に対して, `Xev_GMP_MPI_Send` に書き換え, 引数を修正する処理のみ行えばよい.

5 性能計測

実装したライブラリの性能を東京大学の Reedbush-U[6]上で評価した.

5.1 `Send`, `Recv` の通信性能

はじめに、実装した Xev_GMP_MPI_Send と Xev_GMP_MPI_Recv の内部の各処理の割合を評価した。2プロセス間で配列長 10^5 の mpf_t 型の 1 対 1 片方向通信を行った際の各処理にかかる実行時間と、比較対象として配列長 10^5 の mpf_t 型に対する四則演算の実行時間を図 1 に示す。mpf_t 型は仮数部の精度が 64, 128, 256, 512, 1024, 2048, および 4096 bit のとき、通信する 1 要素当たりのデータサイズはそれぞれ 32, 40, 56, 88, 152, 280, および 536 Byte である。凡例のその他は通信に用いるバッファの確保や、実データ通信を行うためのデータサイズの通信を含む。

図 1 では pack, unpack, および実データ通信の割合が大きく、その他の処理は全体の 0.05% 未満である。データサイズの増加に対し通信と加減算の時間の増加は緩やかであるが、乗算と除算時間は急である。

5.2 疎行列ベクトル積の性能

次に、連立一次方程式の計算の核となる疎行列とベクトルの積 (SpMV) の実行時間を評価した。SpMV では隣接プロセスとベクトルデータを通信するため、Xev_GMP_MPI_Send と Xev_GMP_MPI_Recv を使用する。

行列は 3次元拡散方程式を問題領域 $100 \times 100 \times 100$ で離散化し、1 行あたり 27 個の非零要素を持つものを用いる。仮数部の精度が 64 bit の結果を図 2 に、4096 bit の結果を図 3 に示す。どちらの精度の場合も有効桁数まで乱数を代入した。図 3 での通信時間はどのプロセス数でも図 2 の通信時間の約 10 倍であった。図 1 で仮数部 64 bit と 4096 bit の通信時間を比べた場合も約 10 倍である。一方、図 3 での加算、乗算の時間は図 2 の時間の約 4~5 倍であるのに対し、図 1 で仮数部 64 bit と 4096 bit の加算の時間を比べると約 7 倍、乗算の場合は約 89 倍だった。

6 おわりに

Xev-GMP に MPI 通信を含むコードの変換に対応させるためのライブラリを開発し、基本的な mpf_t 型用 MPI 通信関数を実装した。これらの関数が正常に動作することが分かったため、今後は Xev-GMP においてこのライブラリを用いて実際に変換を行う機能の実装が課題となる。

実用化する上ではふたつの課題が挙げられる。第一は pack と unpack の処理の高速化である。mpf_t 型の仮数部において値が入っていない領域については処理を省いて時間を短縮する方法やスレッド並列化を行う方法が挙げられる。第二に、大規模計算を高速化する上で用いられやすい非同期通信を実装することである。

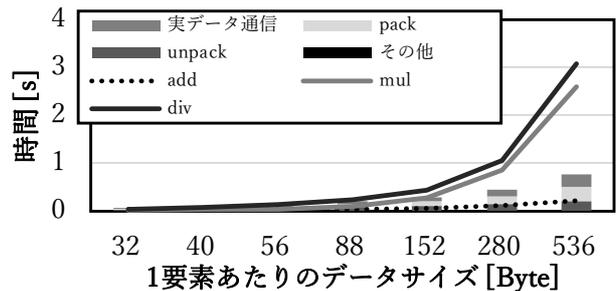


図 1 関数 Xev_GMP_MPI_Send の各処理時間

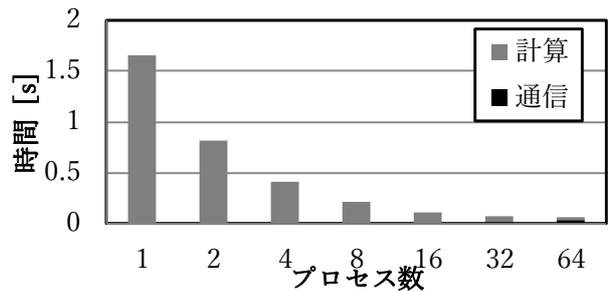


図 2 SpMV の通信、計算時間 (仮数部 64 bit)

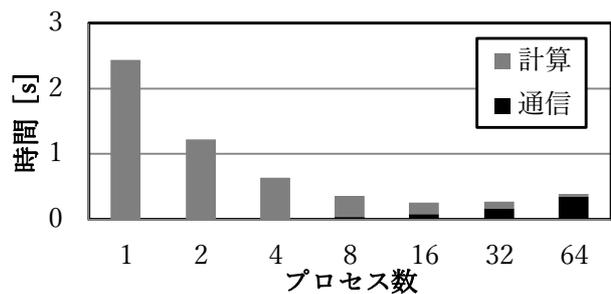


図 3 SpMV の通信、計算時間 (仮数部 4096 bit)

謝辞 本研究の一部は JST CREST「進化的アプローチによる超並列複合システム向け開発環境の創出」の支援、ならびに JSPS 科学研究費 25330144 の助成を受けて行われた。

参考文献

- [1] Tomonori Kouya, A Highly Efficient Implementation of Multiple Precision Sparse Matrix-Vector Multiplication and Its Application to Product-type Krylov Subspace Methods, IJNMA, Vol. 7, Issue 2, pp. 107-119, 2012.
- [2] The GNU MP Bignum Library, <https://gmplib.org/>.
- [3] Xev-GMP : Automatic Code Generation for GMP Code from C Code - Advanced Software HPC labs, <http://hpcl.info.kogakuin.ac.jp/lab/software/xev-gmp>.
- [4] T. Hishinuma, T. Sakakibara, A. Fujii, T. Tanaka, and S. Hirasawa. Xev-GMP: Automatic Code Generation for GMP Multiple-Precision Code from C Code, CSE2016, pp.1-4, 2016.
- [5] MPiGMP Library, <http://na-inet.jp/na/bnc/>.
- [6] Reedbush スーパーコンピュータシステム[東京大学情報基盤センタースーパーコンピューティング部門], <http://www.cc.u-tokyo.ac.jp/system/reedbush/>.