MPI 向け動的メモリ割当分析ツール（DMATP-MPI）

秋元 秀行†‡　安島 雄一郎†‡　安達 知也†　岡本 高幸†‡　三浦 健一†‡　住元 真司†‡

1. はじめに

我々はポストペタスケールスパコンに対応する性能を犠牲としない省メモリ型の通信ライブラリの開発を JST-CREST のプロジェクトの一つとして進めている [1]．省メモリ化に着目した場合，通信ライブラリ単体のメモリ使用量を定量的に測定する必要がある．しかしながら既存のメモリ使用量の測定・評価方法はプロセス単位が主体であり，ライブラリ単体を定量的に評価することは容易でない．

一方 MPI ライブラリの省メモリ化を進めるにあたっては，その削減対象を絞り込む必要があるが，ソースファイルの規模から静的に解析する手法は非現実的である．動的解析についてもメモリ使用量に着目した分析ツールは見当たらない．そこで，我々はこれらの目的に使用可能な MPI 向け動的メモリ割当分析ツール（DMATP-MPI: Dynamic Memory Allocation Tracing Profiler for MPI）を開発したので，その詳細を報告する．

2. 動的メモリ使用量の評価・分析ツールの要件および開発方針

我々は評価・分析ツールの要件として以下の二つを定義した．第一に開発した省メモリ型の通信ライブラリを評価する側面で，メモリ使用量を容易かつ定量的に評価・測定できること．第二に省メモリ化を効率的に進めるための解析・分析ツールとしての側面で，ライブラリ内のメモリ使用量の大きい部分を特定できることである．これらの要件を実現するため以下に述べる方針のもとに，開発を進めた．

- 非測定プログラム・ライブラリに改変を加えることなく動的に解析することにより，容易かつ定量的な評価を実現する．
- 測定粒度としてプロセス以下のライブラリおよびその内部関数別に詳細な分類・集計を行い，省メモリ化対象の絞り込みに使用可能とする．分類は非測定プログラムから最初に呼ばれたライブラリ・関数を基に分別する．

3. DMATP-MPI の詳細および実装

DMATP-MPI は動的ライブラリとして提供し，非測定プログラムの実行時に LD_PRELOAD する．図 1 はメモリ使用量の動的解析における非測定プログラムおよび DMATP-MPI の処理フローである．初期化処理として“malloc”, “memalign”, “realloc”, “free”の動的メモリ獲得・開放関数をフックし，分類・集計機能付きのフック関数に置き換える [2]．加えて非測定プログラム終了時に集計結果を表示させるための結果表示関数の登録を行う．非測定プログラムの動作中は全ての malloc 系関数呼び出しにおいて，3.1 に述べる分類･集計処理を動的に行い，非測定プログラムの終了時に結果を出力する．

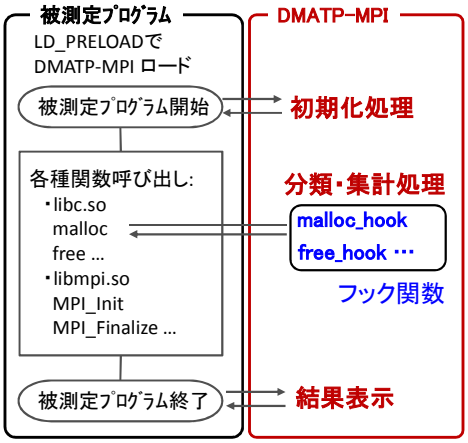

図 1. DMATP-MPI を用いた分類・集計処理の概略フロー

† 富士通株式会社　次世代テクニカルコンピューティング開発本部
Fujitsu Limited, Next Generation Technical Computing Unit
‡ (独)科学技術振興機構　戦略的創造研究推進事業
Japan Science and Technology Agency, Core Research for Evolutional Science and Technology

[コールトレース(backtrace_symbols)の例]

```
0: ./libdmatpmpi.so(malloc_hook+0x91)[0x7f42a4315ad3]
1: /home/akimoto/OpenMPI/lib/libmpi.so.1(opal_class_initialize+0x8f)[0x7f42a408af4f]
2: /home/akimoto/OpenMPI/lib/libmpi.so.1(opal_output_init+0x1a6)[0x7f42a408d266]
3: /home/akimoto/OpenMPI/lib/libmpi.so.1(opal_init_util+0x39)[0x7f42a408a569]
4: /home/akimoto/OpenMPI/lib/libmpi.so.1(ompi_mpi_init+0x6c)[0x7f42a3fd8fdc]
5: /home/akimoto/OpenMPI/lib/libmpi.so.1(MPI_Init+0xf0)[0x7f42a3fee540]  ← ".so"検出
6: ./IMB-MPI1(main+0x2f)[0x4033bf]     ← ユーザプログラムのmain関数
7: /lib64/libc.so.6(__libc_start_main+0xfd)[0x3b2ac1ec5d]
8: ./IMB-MPI1[0x4032d9]                ← ユーザプログラム
```

関数呼び出し順 ↑　　検索対象　　固定部

図 2 malloc_hook 関数呼び出しに至ったコールトレース例（斜体は説明のための追加文字列）

3.1 malloc 系フック関数内の分類・集計処理

動的メモリの獲得・開放量は GNU libc の拡張関数である“malloc_usable_size”を用いて取得する．本関数は malloc 系関数で割り当てたメモリアドレスを引数に呼び出すことで，実際に割り当てた動的メモリ量を返す関数である．これによりメモリアドレスを引数に取る“realloc”や“free”関数における開放メモリ量の取得が大幅に簡略化される．

動的メモリを獲得・開放したライブラリおよびその内部関数への分別は，backtrace 関数群 [2]で得られるフック関数の呼び出しに至るまでのコールトレース情報を用いて行う．図 2 はその一例であり，各行は“ProgLib(Func+Offset)[Addr]”の形式を持ち，ProgLib”はプログラムまたは動的ライブラリ名，“Func”はその内部関数名である．実際のライブラリ・関数の呼び出しは要素 6 から 0 の順であり，“ProgLib”から“.so”を検索することで最初に呼び出された動的ライブラリ，同一行の“Func”からその内部関数を特定する．すなわち図 2 のメモリ獲得は“libmpi.so”の“MPI_Init”によるものとして分別し，メモリ量と合わせて分類・集計する．この時，メモリ使用量については最大・最小値，malloc 系の各関数の呼び出し回数を合わせて集計する．

4. MVAPICH2, Open MPI の評価結果

図 3 は MVAPICH2, Open MPI の関数別の動的メモリ使用量を測定・比較した結果である．測定は Intel MPI Benchmarks の 128 B Exchange[3]を 32 並列で実行し，Rank: 0 で行った．両 MPI 共にメモリ開放は基本的に MPI_Finalize まで行わない作りであることが分かる．一方, MVAPICH2 MPI は MPI_Init 時に必要なメモリを一度に獲得し，Open MPI は通信発生時に必要に応じてメモリを割り当てる実装であることが分かる．

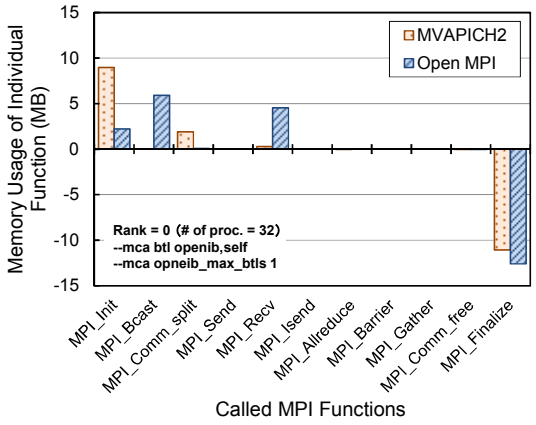

図 3 MVAPICH2, Open MPI のメモリ使用量
Intel MPI Benchmarks 128 B Exchange

5. まとめ

低遅延・省メモリ型の通信ライブラリの開発に必要な動的メモリ割当分析ツール（DMATP-MPI）を提案・実装した．本ツールは非測定プログラム・ライブラリに変更を与えることなく動的にメモリ使用量を解析し，詳細に分類・集計可能である．これにより，通信ライブラリ単体の動的メモリ使用量の定量評価やライブラリ内のメモリ削減対象の絞り込みに有用である．

参 考 文 献

[1] 三浦健一 他, “エクサスケールコンピューティングに向けた省メモリ通信ライブラリの検討”, 情報処理学会研究報告, 2012-HPC-133(14), pp. 1-6, Mar., 2012.

[2] Loosemore S., et al., “The GNU C Library Reference Manual for version 2.16”, Free Software Foundation, Inc., 2012.

[3] Intel, “Intel MPI Benchmarks 3.2.3”, (2011). http://software.intel.com/en-us/articles/intel-mpi-benchmarks