様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　５ 年　 ６ 月 ２８ 日現在

機関番号：３２６２９
研究種目：基盤研究(C)（一般）
研究期間：2018～2022
課題番号：１８Ｋ１１３２７

研究課題名（和文）メニーノード多階層メモリを統合する高汎用メモリシステムの研究

研究課題名（英文）Seamless Unified Memory System over the Distributed Memories on Cluster Computing Nodes

研究代表者

緑川　博子（MIDORIKAWA, Hiroko）

成蹊大学・理工学部・客員研究員

研究者番号：００１９０６８７

交付決定額（研究期間全体）：（直接経費）　　3,300,000円

研究成果の概要（和文）：分散メモリ型の超並列計算システムにおいて，巨大な大域共有メモリ（大規模仮想アドレス空間）を実現するランタイムシステムmSMSを開発し，ローカル／リモートメモリの区別なくデータにアクセスできるプログラミング環境を構築した．典型的な科学技術計算では，従来のメッセージパッシング型(MPI)プログラムに比べ，プログラムの作成が容易な上，高い性能が得られる．また１つの計算ノードメモリには収まらない大規模N体問題では，この大域共有メモリに大規模グローバルツリーを作りBarnes-Hutアルゴリズムを実行可能で，MPIによる実装に比べ，プログラム開発の生産性を飛躍的に向上させた．

研究成果の学術的意義や社会的意義
スーパーコンピュータでは，複数の計算ノードをネットワークで接続し，並列計算により高速処理を行なっている，しかし，計算ノードはそれぞれ独立のメモリとアドレス空間を持ち，他の計算ノードのメモリにあるデータ（遠隔データ）をアクセスするには，特別なプログラムインターフェースを必要とする．多数の計算ノードを用いた並列処理では，プログラム記述は非常に複雑でプログラム開発は大きな負担となっている．このようなシステムにおいて，プログラム開発の生産性を高め，大域共有メモリ（大規模仮想アドレス空間）を実現するmSMSを開発し，ローカル／リモートメモリの区別なくデータにアクセスできるプログラミング環境を構築した．

研究成果の概要（英文）：We have developed a run-time system mSMS that realizes a huge global shared memory (large-scale virtual address space) in a distributed memory type massively parallel computing system. We have constructed a programming environment that can access data in the same manner regardless of local/remote memory. In typical scientific computing, the programs in mSMS are easier to create and have higher or equal performance than conventional message-passing (MPI) programs. For large-scale N-body problems that cannot fit in the memory of a single computing node, a large-scale global tree can be constructed in a mSMS huge shared memory and address-pointer access is available in typical Barnes-Hut programs. The programs in mSMS are much easier to develop compared to MPI programs.
The mSMS achieaved comparable or better performance compared to existing MPI programs, and it dramatically improved the program development productivity on a distributed memory computing system.

研究分野：高性能計算

キーワード：分散共有メモリ　遠隔メモリ　クラスタ　仮想共有メモリ　マルチノード並列処理　PGAS　分散メモリ　メモリアドレス空間

１．研究開始当初の背景

スーパーコンピュータ(スパコン)など計算機クラスタ利用による高性能計算では，現在も MPI （Message Passing Interface）が広く用いられ，分散メモリ型並列計算機におけるプログラム開発の低生産性が解決されたとは未だ言えない．これを軽減するため，PGAS（Partitioned Global Address Space）モデルと総称される様々な言語・API が提案されてきたが[1]，大域データ配列や大域インデックスを利用可能とするものはあるものの，可能なアクセス範囲がノード隣接データ領域に限られていたり，範囲を超えた大域データアクセスには MPI のような明示的な記述が必須であったり，定義できる大域データサイズに制限があり大規模計算には利用できないなど，多くの不自由さが存在する．

２．研究の目的

スパコンを構成する多数の計算ノードに分散するメモリの効率的利用を実現するための新たな枠組みを開発し，高性能実行とプログラム開発の高生産性を実現する．すなわち，ノード間通信方式，データ分散方式，データ一貫性モデル，並列実行モデルを含むシステムソフトウエアを開発し，大規模な分散共有メモリシステムを構築する．特定の応用処理パターンのみに特化したシステムではなく，グローバルデータへの様々なアクセスモード（同期・非同期，粗粒度・細粒度，プリフェッチ有無，メモリ一貫性の厳密・緩和，アクセス領域制限・無制限）を実装し，応用処理の特性やユーザ技量に応じた最適な記述手法やプログラミングインターフェースを提供する．

3. 研究の方法

本研究では，図１のように，クラスタシステムの全計算ノードプロセスに同一の共有大域アドレス空間を提供するランタイムシステム，分散共有メモリシステム mSMS（multithreaded Shared Memory System）を実現する．多くの PGAS が，それぞれの専用コンパイラを使って，大域データアクセスを静的なノード間通信プログラムなどに変換するのとは対照的に，mSMS では，大域アドレス空間へのアクセスは，mSMS ランタイムシステムがリアルタイムで処理する．ランタイムシステムは複数の SMS システムスレッドから成り（図２），効率的メモリページ転送，アクセス排他制御，同期処理などの様々な機構を備える．高移植性を保持するため，ノード間のシステムスレッドには高速 MPI 通信を用いる．

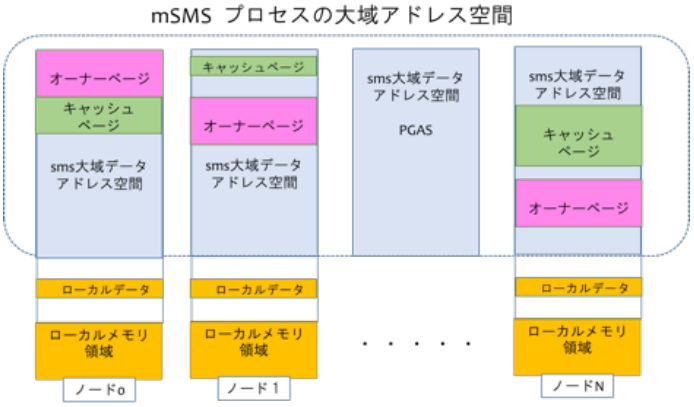


図１　クラスタ上で大域アドレス空間を実現するソフトウエア分散メモリ mSMS

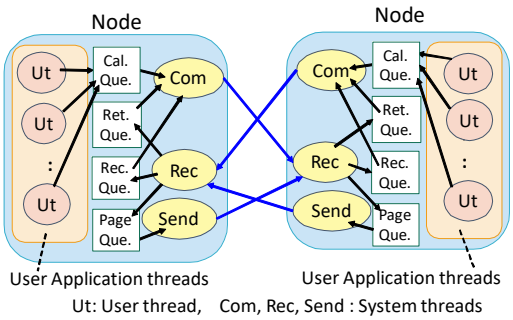


図２　mSMS の内部実装

４．研究成果

(1) スーパーコンピュータにおいて大規模共有メモリモデルによる楽な記述と高性能を両立

スパコン(東工大 Tsubame3.0)において，２から 180 の計算ノードのメモリを利用し，最大 22.5TB の共有データを定義し，典型的な科学技術計算の一つである７点ステンシル計算を行い，MPI プログラムと mSMS との性能を比較したところ，MPI と同等以上の性能を得た(図３)．従来の MPI のプログラムとは異なり，扱う大域配列データは shared 宣言を行うのみで，分散メモリ上にマッピングされて，一つの共有配列として扱える上，MPI における通信記述は不要となる．図４に示すように，通常の C プログラムと同等な記述でマルチコア並列(OpenMP)とマルチノード並列(mSMS)を実現した．

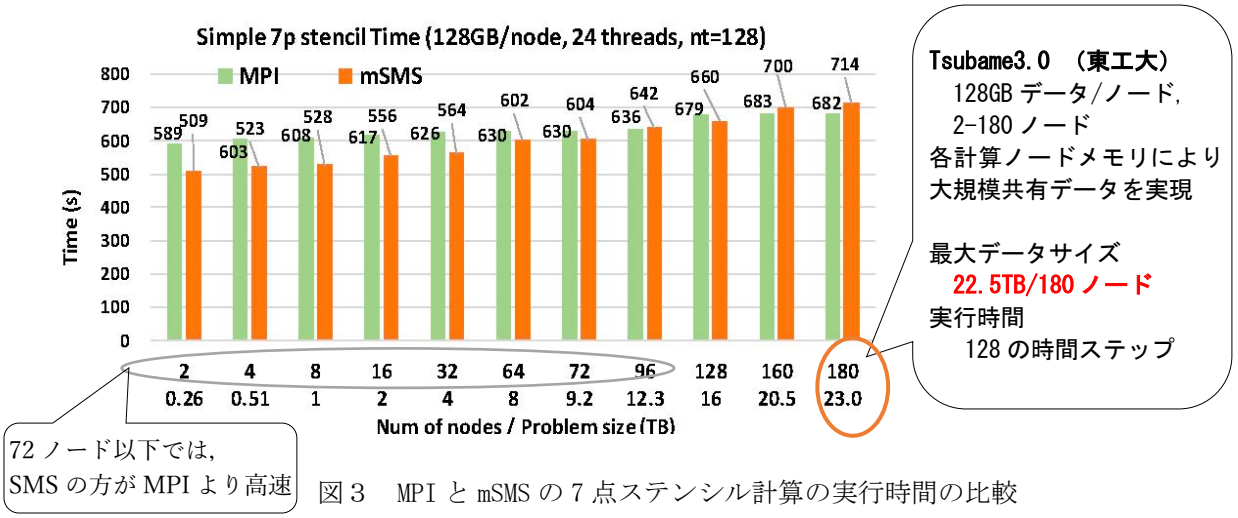


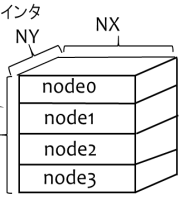
図３　MPI と mSMS の 7 点ステンシル計算の実行時間の比較

shared 指定子により，クラスタシステム上に．大域共有データ配列を定義

```
shared double A[NZ][NY][NX]::[NPROCS][1][1](0, NPROCS); // 大域データ配列
shared double B[NZ][NY][NX]::[NPROCS][1][1](0, NPROCS); // Z分割分散マップ
main()
{    double (*src)[NY][NX]; double (*dst)[NY][NX]; double (*tmp)[NY][NX]; // 3次元arrayを指すポインタ
     mpc_init(&argc,&argv);   // sms_startup() に変換
     nx = NX, ny = NY, nz = NZ;  // 配列サイズ
     bx = nx, by = ny; bz = nz / NPROCS;  // ブロックサイズ, Z次元分割
     // 各ノードの計算領域を始点，終点設定
     sx = 0; ex = bx;    sy = 0; ey = by;
     sz = MYPID * bz;  ez = (MYPID + 1) * bz;
          :
     for (z = sz; z < ez; z++)  for (y = sy; y < ey; y++)  for (x = sx; x < ex; x++) {A[z][y][x] = ?; B[z][y][x] = ?; } // 配列初期化
     mpc_barrier();  // データー貫性(ここでは通信トラフィックなし）と実行同期，sms_barrier()に変換
     src = A;  dst = B;
     for (t = 0; t < nt; t++) {   // 時間ステップ
     #pragma omp parallel for
       for (z = sz; z < ez; z++)  for (y = sy; y < ey; y++)  for (x = sx; x < ex; x++) { // 7点ステンシル計算
          dst[z][y][x]=0.4*src[z][y][x]+
                  0.1*(src[z-1][y][x]+src[z+1][y][x]+src[z][y-1][x]+src[z][y+1][x]+src[z][y][x-1]+src[z][y][x+1]);
       }
       sms_sync_drop();   //   実行同期，各ノードキャッシュページ廃棄
       tmp = dst;   dst = src;    src = tmp;      // srcとdstの交換
     }
     mpc_exit();   // sms_shutdown() に変換
}
```

7点単純ステンシル計算スケルトン
（MpCプログラム→ SMSライブラリ関数へ変換）

大域データ配列の分散マップ
Z分割4nodeへのマップ例



MYPID：MPI rankと同等
NPROCS：MPI 総プロセス数

実際は，omp parallel を利用して，各種変数を定義，
6重ループによる空間ブロック化を用いている

図４　mSMS を利用した７点ステンシル計算プログラム（拡張 C プログラム：MpC＋ OpenMP）
MpC：C に最低限の拡張（shared 共有データ分散マップ宣言を追加）を加えた言語

(2) スーパーコンピュータにおいて巨大グローバルツリーを構築し barnes-Hut 高速処理を実現

(1)のステンシル計算は，データアクセス領域とアクセスタイミングが静的かつ既知であり，従来の PGAS 言語でも，独自コンパイラにより事前に MPI 通信などを含むプログラムに静的変換することで対応ができる．しかし，いつどこの共有データがアクセスされるかわからない応用には対応できない．mSMS はランタイムシステムであるため，プログラムの動的なデータアクセス要求に応じて，従来の PGAS で設けられているアクセス領域やタイミングの制限を受けずに，共有データをアクセスできる．Barnes-Hut アルゴリズム（図 5）は，一つのグローバルツリーにアクセスすることにより，要求する精度に合わせて計算を省略して高速化でき，N 体問題など，共有メモリ型並列処理では広く用いられるが，分散メモリ向けの MPI では共有データが作成できないため，データの事前分割配分と再配分などを繰り返す複雑なアルゴリズムで実装するしかなかった．mSMS では，メモリアクセス局所性が高まるように各計算ノードにグローバルツリーの部分木を割り当て(図 6)，多くのアクセスがローカルメモリアクセスに留まるように工夫し性能向上を図っている．プログラム作成は容易で，マルチコアシステム向けのスレッドプログラムを手直しする程度で，一つの計算ノードのメモリには入らないような大規模な n 体問題を，mSMS の共有メモリ上のグローバルツリーを利用して効率よく解くことができる．

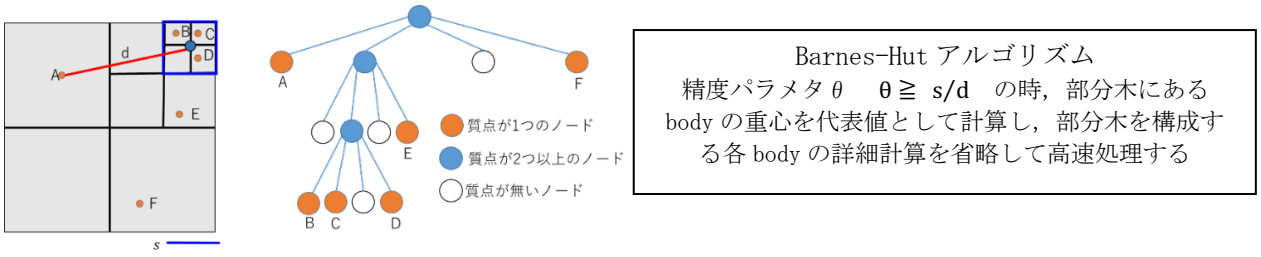


図５　Barnes-Hut アルゴリズムとツリーデータ構造

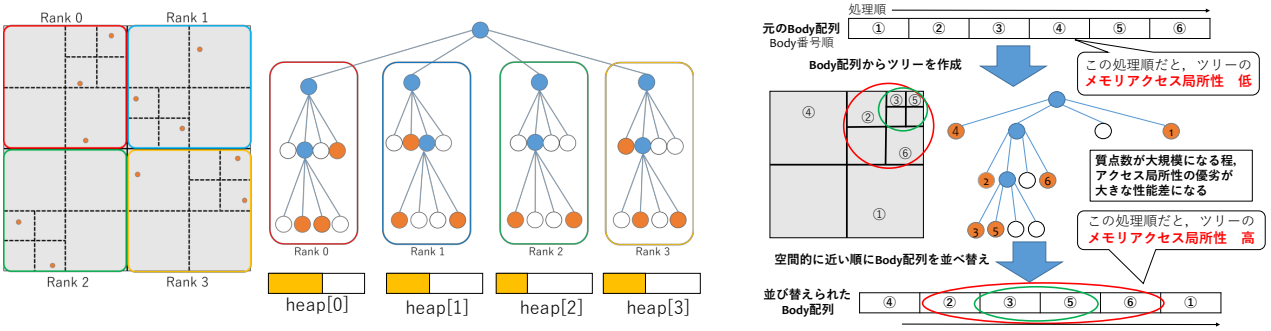


図６ (a)グローバルツリーの各計算ノードメモリへのマッピング　(b) メモリアクセス局所性を高めるデータ格納手法

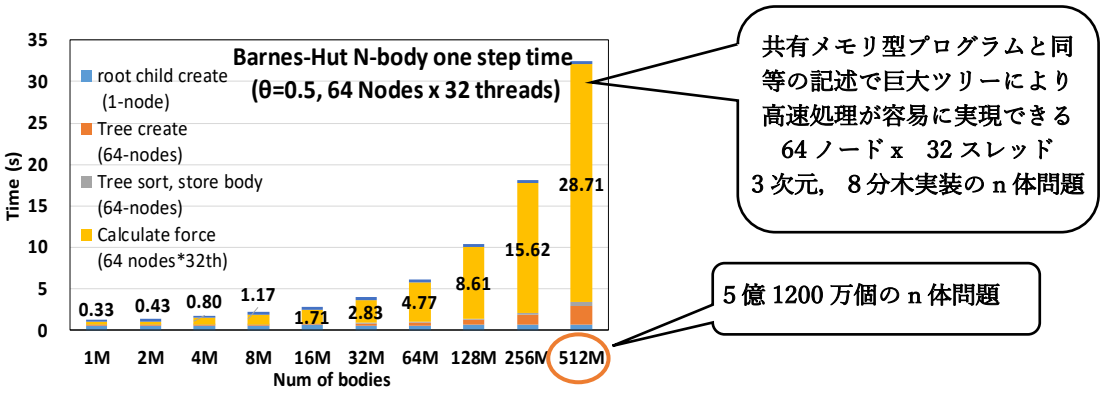


図７　３次元空間 N 体問題(質点：1M － 512M 個)

(3) プログラム開発の生産性を高めるグローバルビューに基づく sms 並列プログラミング環境の構築

mSMS を利用したプログラム開発環境は，図８に示すように階層構造で３つの API を提供する．ユーザの熟練度や応用の特徴に応じて自由に選択することができる．第１は，最も単純な API で，OpenMP や OpenACC と同様に，逐次コードにインクリメンタルに pragma 文の挿入する SMint API である（図 10）．OpenMP や OpenACC との共存も可能で，マルチノード，マルチ CPU，マルチ GPU を自由に組み合わせて選択し，逐次コードから容易に並列プログラム記述ができる(図 9)．第２は．分散ノードへの共有データ宣言 shared を利用できる MpC によるプログラミング(図４)，第３は，C と SMS ライブラリ関数を直接用いて記述する手法（図 11）である．いずれにおいても，実際には分散マッピングされた共有データをグローバルビューでプログラムできること

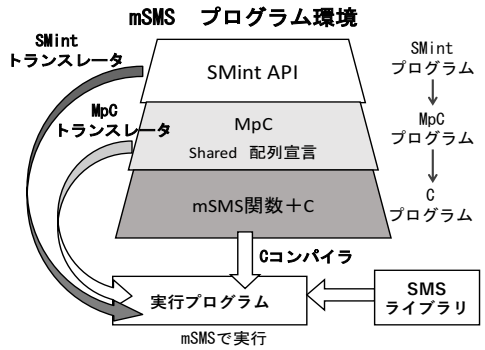


図８　３つの API（SMint，MpC，sms 関数）

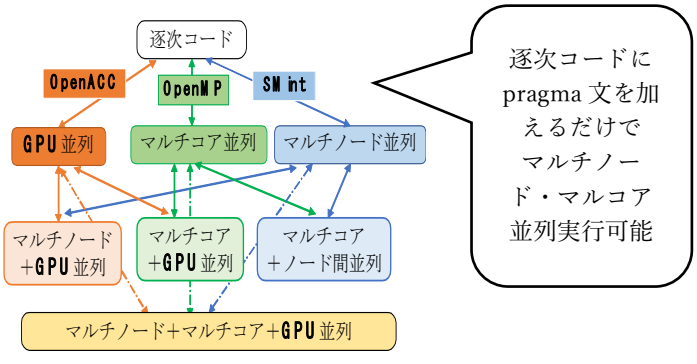


図９　SMint インクリメンタルプログラミング

が大きな特徴で，従来のローカルデータと通信を用いたプログラムとは異なり，プログラム開発生産性が高い．mSMS におけるプログラムでは，OpenMP，pthread，OpenACC，CUDA も併用可能である．

```
#include <smint.h>        // #pragma SMint 利用プログラム
#define N …
#pragma SMint shared ::[1](0,1) ;     // node0にマップ
double vec1[N] ;
#pragma SMint shared ::[](1,1) ;      // node1にマップ ,省略記述
double vec2[N] ;
#pragma SMint shared ::[NPROCS][](0,NPROCS) ;//全ノードに分散マップ
double array[N][N] ;

int main(int argc, char *argv[] )
{   int size, st, ed;              // 各ノードの担当領域

  #pragma SMint parallel for  // 全ノードで並列実行
  #pragma omp parallel for    //各ノードでマルチスレッド実行
   for( i=0; i<N; i++) {
     for(k=0; k<N; k++)
          vec2[i]= array[i][k] * vec1[k];    // 行列ベクトル積
   }
  // バリア同期は自動挿入
}
```

図 10　SMint API による行列ベクトル積

```
#include <sms.h>                  //  SMSライブラリ関数利用によるC プログラム
#define N …
int main(int argc, char *argv[] )
{  int size, st, ed;              //各ノードの担当領域
   double *vec1, *vec2; // 1次元配列 vec1[N], vec2[N] のためのポインタ
   double (*array) [N]; //2次元配列 array[N][N] のためのポインタ
   int dim [3]= {N, N,-1}, div[3]= {1, 1,-1}; // dim : 配列サイズ, div:分散マップ分割数

   sms_startup (&argc, &argv);

   vec1 = (double*)sms_alloc(sizeof(double), N, 0); // vec1[N] node0に割り付け
   vec2 = (double*)sms_alloc(sizeof(double), N, 1); // vec2[N] node1に割り付け
   div[0]= sms_nprocs;        //  arrayをバンド分割，全ノードに分散マッピング
   array= (double(*)[N]) sms_mapalloc(dim , div, sizeof(double), 0, sms_nprocs );

   size= N/sms_nprocs;
   st= size * sms_rank; ed= size * (sms_rank+1); //各ノード担当領域
  #pragma omp parallel for         //  各ノードでは，マルチスレッド実行
   for( i=st; i<ed; i++) {          // 全ノードで for(i=0; i<N; i++) を並列実行
     for(k=0; k<N; k++)   vec2[i]= array[i][k] * vec1[k];    //行列ベクトル積
   }
   sms_barrier();
   sms_shutdown();
}
```

図 11　SMS ライブラリ関数による行列ベクトル積

(4)他の PGAS 言語との記述性，性能の評価実験

代表的な PGAS 言語(UPC，XcalabeMP，MPI，SMint)を，東工大のスパコン(Tsubame3.0)にインストールし，プログラム生産性の一つの指標である記述性と性能を比較した．他の PGAS 言語の制限で小規模な 5 点ステンシル計算で比較した（表１）．各言語が最も高速だった最適スレッド数（事前調査）で実行した時の利用ノード数ごとの５点ステンシル計算時間を図 12 に示す．この結果，mSMS 利用による SMint で記述したプログラムは，MPI と同等以上の性能を得たばかりでなく，逐次コードとのプログラム行数の比較において，最も短いことが明らかになった．この結果，ステンシル計算のような最も単純な計算においても，SMint（mSMS）による記述では最も優れていることが明らかになった．mSMS ではアクセス領域やタイミングの制限がない上，(2)に述べたようなツリーをアドレスポインタ利用でアクセスする応用も可能で，さらに幅広い応用の記述が可能である．

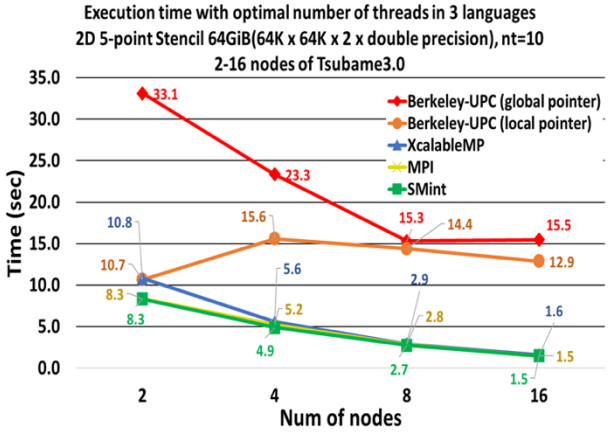


図 12 各種言語の５点ステンシル計算の性能比較

表１　各種言語のプログラム記述性と性能比較

各種言語	プログラムの記述性		性能	特徴
	行数	逐次に対する行数増加率	MPIに対する相対実行時間＊	扱えるグローバルデータサイズ、アクセス可能領域、記述の制限など。
UPC (Global view model)	29	1.07	4.0 ～ 10.4	グローバルデータサイズに制限有　小さい
UPC (Local view model)	57	2.11	1.3 ～ 8.9	MPIと同様のノードを意識したローカル記述
UPC (動的確保 upc_alloc)	71	2.63	9.4 ～ 77.0	コードが複雑
XcalableMP	40	1.48	1.0 ～ 1.3	グローバルデータのアクセス可能領域に制限有
SMint	31	1.15	0.9 - 1.0	グローバルデータのサイズ、アクセス制限なし
MPI	73	2.70	1	
Sequential (OpenMP)	27	1.00		

＊ノード、スレッド数により異なる

(5)音響解析並列計算における mSMS 利用の成果

これまで並列処理プログラム開発の経験がない音響解析の研究者に，mSMS を利用してもらい，音響解析 FDTD(2,4)の並列処理の記述と大規模スパコンでの実行を行ない，研究に役立ててもらった．この結果，シミュレーションで重要な境界条件の処理の記述や変更が簡単で非常に分かりやすく，大規模スパコンを利用し容易に並列処理による高速化ができるという評価を得た．紙面の制限で詳細は割愛するが，mSMS のプログラム開発における高生産性の一つの評価になると考えている．

＜引用文献＞

[1] M.D. Wael, et al.: “Partitioned Global Address Space Languages”, Journal of ACM Computing Surveys (CSUR), Vol.47, No.62, 2015

５．主な発表論文等

〔雑誌論文〕　計7件（うち査読付論文　3件／うち国際共著　0件／うちオープンアクセス　1件）

１．著者名 Ryoya Tabata, Rei Mastuda, Toshiaki Koiwaya, Sho iwagami, Hiroko Midorikawa, Taizo Kobayashi, Kinya Takahashi	４．巻 149
２．論文標題 Three-dimensional numerical analysis of acoustic energy absorption and generation in an air-jet instrument based on Howe's energy corollary,	５．発行年 2021年
３．雑誌名 The Journal of the Acoustical Society of America	６．最初と最後の頁 4000-4012
掲載論文のDOI（デジタルオブジェクト識別子） 10.1121/10.0005133	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

１．著者名 緑川博子，阪口裕悟	４．巻 2021-HPC-178
２．論文標題 分散共有メモリシステムmSMSにおけるマルチノード・マルチCPU・マルチGPUプログラミング	５．発行年 2021年
３．雑誌名 情報処理学会，研究報告ハイパフォーマンスコンピューティング	６．最初と最後の頁 1-10
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 阪口裕悟，緑川博子	４．巻 Vol.2020-HPC-173, No.7
２．論文標題 グローバルビュープログラミングをサポートするPGAS言語の記述性と性能の比較	５．発行年 2020年
３．雑誌名 情報処理学会，研究報告ハイパフォーマンスコンピューティング	６．最初と最後の頁 1 - 8
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 緑川博子，　柴山悠	４．巻 Vol.2019-HPC-170, No.43
２．論文標題 分散共有メモリシステムを利用したBarnes-Hutアルゴリズムの初期並列実装と性能評価	５．発行年 2019年
３．雑誌名 情報処理学会，研究報告ハイパフォーマンスコンピューティング	６．最初と最後の頁 1-8
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 阪口裕悟，緑川博子	４．巻 Vol.2019-HPC-170，No.41
２．論文標題 グローバルビュープログラミングをサポートするPGAS言語の記述性と性能の比較	５．発行年 2019年
３．雑誌名 情報処理学会，研究報告ハイパフォーマンスコンピューティング	６．最初と最後の頁 1-8
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 無
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Hiroko Midorikawa, Kenji Kitagawa, Yugo Sakaguchi	４．巻 2019
２．論文標題 mSMS : PGAS Runtime with Efficient Thread-based Communication for Global-view Programming	５．発行年 2019年
３．雑誌名 2019 IEEE International Conference on Cluster Computing (CLUSTER)	６．最初と最後の頁 1-2
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/CLUSTER.2019.8891009	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Yugo Sakaguchi,Hiroko Midorikawa	４．巻 2019
２．論文標題 The Programability and Performance of Global-View Programming API: SMint for Multi-node and Multi-core Processing"	５．発行年 2019年
３．雑誌名 2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)	６．最初と最後の頁 1-8
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/PACRIM47961.2019.8985126	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計5件（うち招待講演　0件／うち国際学会　2件）

１．発表者名 緑川博子
２．発表標題 高性能，高生産性を実現する大規模メモリ・並列処理システムソフトウエアの研究
３．学会等名 JHPCN 学際大規模情報基盤共同利用・共同研究拠点　第11回シンポジウム
４．発表年 2019年

１．発表者名
Ryoya Tabata, Hiroko Midorikawa, Ki'nya Takahashi

２．発表標題
Performance Evaluation of Acoustic FDTD(2,4) Method Using Distributed Shared Memory System mSMS

３．学会等名
2020 International Conference on High Performance Computing in Asia-Pacific Region（国際学会）

４．発表年
2020年

１．発表者名
阪口裕悟，西矢和生，緑川博子

２．発表標題
逐次プログラムからマルチコア・マルチノード並列処理への変換を容易にするディレクティブベースAPI SMint

３．学会等名
情報処理学会，ハイパフォーマンスコンピューティング研究会報告

４．発表年
2018年

１．発表者名
緑川博子

２．発表標題
ソフトウエア分散共有メモリシステムmSMSによる大規模マルチコアノードにおけるステンシル計算

３．学会等名
情報処理学会，ハイパフォーマンスコンピューティング研究会報告

４．発表年
2018年

１．発表者名
Hiroko Midorikawa

２．発表標題
mSMS: A New DSM System for HPC

３．学会等名
The International Conference for High Performance Computing, Networking, Storage, and Analysis 2018 (SC18), SC18 Tokyo Inst. of Tech. Booth-Poster（国際学会）

４．発表年
2018年

〔図書〕　計1件

１．著者名 Mitsuhisa Sato, Toshio Endo, Hiroko Midorikawa, Yukinori Sato, et,al.	４．発行年 2019年
２．出版社 Springer	５．総ページ数 317
３．書名 Advanced Software Technologies for Post-Peta Scale Computing	

〔産業財産権〕

〔その他〕

緑川博子　論文リスト
http://www.ci.seikei.ac.jp/midori/paper/paper.html

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

７．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

８．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関