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　３ 年　 ５ 月 ３１ 日現在

機関番号：８２４０１
研究種目：基盤研究(C)（一般）
研究期間：2018～2020
課題番号：１８Ｋ１１３３１

研究課題名（和文）指示文とメタプログラミングによる性能可搬性に優れた高性能並列プログラミングモデル

研究課題名（英文）High-performance parallel programming model with performance portability using directives and metaprogramming

研究代表者
中尾　昌広（Nakao, Masahiro）

国立研究開発法人理化学研究所・計算科学研究センター・研究員

研究者番号：５０５８２８７１

交付決定額（研究期間全体）：（直接経費）　　3,400,000円

研究成果の概要（和文）：複数の高性能な計算機を用いて構成された並列計算機システムが、様々な分野で用いられている。そのようなシステムにおけるアプリケーション開発では、そのアーキテクチャによって異なるプログラミング言語が用いられる。そのため、システム間におけるコードの移植性が問題となっている。
本研究では、コードの移植コストを小さくするため、新しい並列プログラミング言語の設計を行った。具体的には、並列計算機システムの性能を引き出すことができる指示文およびメタプログラミングの設計を行った。そして、1つのプログラミング言語で記述されたソースコードで様々なシステムにおいて高速に動作するアプリケーションを開発できることを示した。

研究成果の学術的意義や社会的意義
将来の並列計算機システムでは、現在よりも複雑化・多様化すると考えられる。本研究成果を用いることで、そのようなシステムにおいても高速に動作するアプリケーションを簡易に開発できるため、コードの移植や性能チューニングに要するコストを大幅に削減することが期待できる。

研究成果の概要（英文）：Parallel computer systems, which are composed of multiple high-performance computers, are used in various fields. In developing applications for such systems, different programming languages are used depending on the architecture. Therefore, the portability of code between systems has become an issue.
In this research, we have designed a new parallel programming language to reduce the cost of code porting. Specifically, we designed directives and metaprogramming that can bring out the performance of parallel computer systems. We showed that source code written in a single programming language can be used to develop applications that run at high speed on a variety of systems.

研究分野：計算機工学

キーワード：高性能計算　コンパイラ　プログラミング　HPC

１．研究開始当初の背景

計算科学分野で行われる大規模シミュレーションには、複数の高性能な計算ノードをネットワークで接続したクラスタシステムが広く利用されている。クラスタシステムにおけるアプリケーションの作成では、クラスタシステムの構成（特にアクセラレータの有無とその製造元）によって異なるプログラミング言語が用いられることが一般的である。そのため、クラスタシステム間におけるコードの移植に要するコストが大きいことが問題となっている。また、例えば、同じ製造会社の CPU やアクセラレータであっても、それらの世代によってアーキテクチャは大きく異なる場合があるため、移植する際の性能チューニングに要するコストが大きいことも問題となっている。さらに、半導体の物理的な限界により、CPU やアクセラレータにおける１つの計算コアの性能は頭打ちになりつつあるため、将来の計算ノードの演算器は現在よりも複雑化・多様化すると考えられる。以上の動向により、将来のクラスタシステムでは、コードの移植と性能チューニングに要するコストは現在のクラスタシステムよりも大きくなると考えられる。

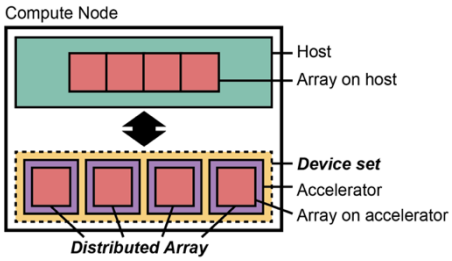
２．研究の目的

本研究の目的は、クラスタシステムにおいて、コードの移植および性能チューニングに要するコストを小さくするための新しいプログラミング言語を開発することである。研究代表者らはこれまでの研究において、高生産性と性能を両立させたクラスタシステムのための指示文ベースの並列言語 XcalableMP（XMP）を開発してきた。XMP を用いると、ユーザは簡易な指示文を用いるのみで、逐次コードから並列コードを生成することができる。また、XMP では既存の指示文ベースの並列言語 OpenMP との連携も可能である。XMP ではプロセスレベルでの並列性を、OpenMP ではスレッドレベルでの並列性をユーザに提供する。さらに、OpenMP の target 指示文を用いることで、アクセラレータに対するプログラミングも行うことができる。

XMP と OpenMP が提供する指示文は、ハードウェアの違いをある程度吸収できるが、性能可搬性（ハードウェアなどの計算機環境に依存せずに、あるコードが高性能となるように実行できる性質のこと）については考慮されていない。そこで、本研究では XMP と OpenMP に性能可搬性を実現する機能を追加した新しいプログラミング言語を開発する。このプログラミング言語を用いることで、様々なクラスタシステムにおいて高い性能を発揮する並列アプリケーションを簡易に作成できる。この特性により、将来のクラスタシステムにおいても、クラスタシステム間におけるコード移植と性能チューニングのコストを大幅に削減することが可能になる。

３．研究の方法

(1) OpenMP の拡張



計算ノード内の複数の計算資源を１つに抽象化し、それを扱えるように OpenMP の拡張を行った。その概念を右図に示す。この OpenMP 拡張では「デバイスセット（Device set）」と「分散配列（Distributed Array）」という概念を用いる。デバイスセットとはホストに接続されている複数のアクセラレータの組である。OpenMP 拡張ではデバイスセットに対して処理を記述することで、複数のアクセラレータを１つの計算資源のように扱うことができる。分散配列はデバイスセット上にマップされた配列であり、ホストメモリにある１つの配列を複数のアクセラレータのメモリに分散してマップすることが可能である。具体的には、OpenMP 拡張は次の機能を提供する。①デバイスセットと分散配列の定義。②ホストとデバイスセット間におけるデータ転送。③デバイスセット内のデバイス間におけるデータ転送。

上記の機能を実現するため、既存の OpenMP 指示文を次のように拡張した。①デバイスセットを扱うために、既存の device 節で array section を扱えるように拡張した。②タスクとデータの分割とマップを行うため、layout 節を新設した。③科学技術計算でよく現れる計算パターンであるステンシル計算を簡易に行うため、袖領域を定義する shadow 節と袖領域を同期するための target reflect 指示文を新設した。④デバイスセットに対する通信を行うため、target gmove 構文、target bcast 指示文、target reduction 指示文を新設した。

(2) XcalableMP の拡張

複数の計算ノードを利用した並列アプリケーションの性能可搬性の向上を実現するため、XMP の拡張を行った。具体的には、ステンシル計算を含む並列アプリケーションを効率よく実装するために次の機能を XMP が提供する指示文に追加した。①最適化手法の 1 つであるテンポラルブロッキング法を簡易に記述する記法。②袖通信に対する非同期化を簡易に記述できる記法。なお、これらの２つの機能を同時に用いることも可能である。また、①のテンポラルブロッキング法には袖幅というパラメータがあり、最適な袖幅はアプリケーションや計算機環境によって異なる。

そのパラメータチューニングを簡易に行うため、指示文に記載するパラメータだけを変更することで、袖幅を変更できるように設計している。

(3) メタプログラミングの利用

メタプログラミング的手法を取り入れることにより、既存のコンパイラシステムでは不可能な、アルゴリズムの変更を伴うような性能チューニングも可能にする。Fortran および C 言語をターゲットとしたメタプログラミングについて検討を行い、その言語の設計およびコンパイラの実装を行った. ここで言うメタプログラミングとは、ソースコードを処理対象とするプログラミングのことである。HPC 向けの高度な最適化をメタプログラミングの技法で記述することにより、複雑なコード変形をユーザプログラムから切り離すことが可能となる[3]。

(4) XMP と Python の連携

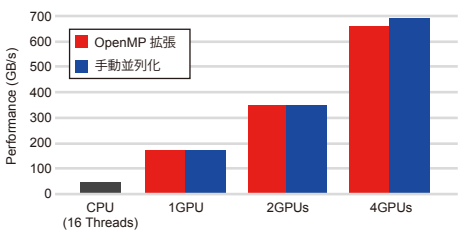
クラスタシステムにおけるアプリケーションの開発には, XMP のような高い生産性と性能を発揮できる並列言語がよく利用される。さらに、アプリケーションの開発を効率的に行うには、並列言語のみでアプリケーションを開発するのではなく, 並列言語と他の言語とを組合せて用いることも重要である。そこで、広く利用されている Python と XMP との連携機能を作成した。具体的には、XMP から Python を呼び出すインタフェース、および Python から XMP を呼び出すインタフェースを策定し、それらの実装を行った。

４．研究成果

(1) OpenMP 拡張に対する成果

OpenMP 拡張を用いて並列化した STREAM ベンチマークのプログラムの一部を右図に示す。2 行目と 3 行目の device 節は、device(0) から device(3)の 4 つのアクセラレータで構成されたデバイスセットを用いることを示している。2 行目の layout 節は、配列 a[], b[], c[] をブロック分割し、それらを各デバイスにマップすることを示している。3 行目の layout 節は、後に続く for 文のイテレーションをブロック分割することを示している。

```
1 double a[N], b[N], c[N], scalar = 1.0;
2 #pragma omp target map(to: b,c) map(from: a) device(0:4)
      layout(block)
3 #pragma omp teams distribute parallel for device(0:4)
      layout(block)
4 for(int i=0;i<N;i++)
5   a[i] = b[i] + scalar * c[i];
```

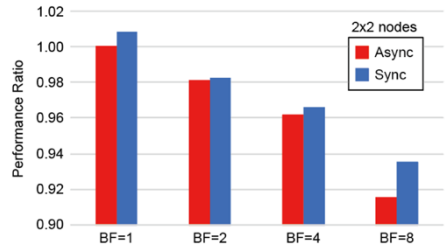


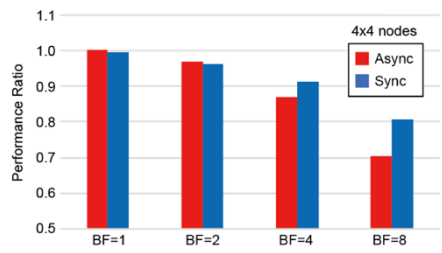
性能評価を行った結果を右図に示す。アクセラレータには、NVIDIA Tesla K80 を用いた。この結果より、手動で並列化を行った結果と OpenMP 拡張を用いた性能はほぼ同じであることがわかる。なお、ステンシル計算の典型例である姫野ベンチマークについても同様の実験を行い、そちらも性能がほぼ同じであることを確認している。ただし、手動で並列化を行った場合はコードが非常に複雑になるため、OpenMP 拡張を用いた方が開発に要するコストは小さいと言える[2,5]。

(2) XMP 拡張に対する成果

XMP 拡張を用いて並列化したステンシル計算の例を右図に示す。図中の BF は袖幅を意味している。3 行目の reflect 指示文で配列 a の袖領域を非同期通信している。4〜7 行目では、通信から得られるデータと依存関係のない箇所のみを先に計算し、通信後にそのデータと依存関係のある箇所を計算することを示している。9 行目以降は、それ以外の計算である[6,10]。

```
1  #pragma xmp shadow a[BF][BF], b[BF][BF]
2  for(int n=0;n<ITER;n+=BF){
3  #pragma xmp reflect (a) async
4  #pragma xmp loop on t[i][j] peel_and_wait(−1,−1)(BF−1,BF−1)
5    for(int i=1;i<N−1;i++)
6      for(int j=1;j<N−1;j++)
7        b[i][j] = (a[i+1][j+1] + ... + a[i−1][j−1])/8;
8
9  #pragma xmp loop on t[i][j] expand(BF−2,BF−2)(0,0)
10   for(int i=1;i<N−1;i++)
11     for(int j=1;j<N−1;j++)
12       b[i][j] = (a[i+1][j+1] + ... + a[i−1][j−1])/8;
13 }
```





ステンシル計算の典型例である姫野ベンチマークを用いて、理化学研究所が持つスーパーコンピュータ「富岳」上で性能評価を行った。その結果を下図に示す。利用ノード数は 2x2 と 4x4 ノードである。また、非同期通信と同期通信を用いたそれぞれの実験を、袖幅を変えながら実行した。縦軸は、非同期通信かつ BF=1 の結果を 1.0 とした場合の相対値であり、大きい方が良い値である。

この結果より、両方のノード数とも BF=1 が最も性能が高いことがわかる。しかしながら、2x2 ノードの場合は同期通信が、4x4 の場合は非同期通信を用いた方が性能は高いことがわかる。

この実験により、XMP 拡張を用いることで、ステンシル計算におけるパラメータチューニングを簡易に行えることがわかった。

(3) メタプログラミングに対する成果
　具体的な評価として、ループ・アンローリングやデータレイアウト変換をメタプログラミングにより簡易に適用できることを示した[3]。

(4) XMP と Python との連携に対する成果
　策定したインタフェースの一部を利用したコード例を右図に示す。(a)の 3 行目では XMP のモジュールをインポートしている。5 行目では XMP で作成した共有ライブラリをロードしており、7 行目では共有ライブラリ内の関数 test()を実行している。このように、簡易な手順で Python から XMP の関数を呼び出すことができる。例えば、高速に行う必要がある処理については XMP で記述し、それ以外の箇所（可視化や統計処理など）については Python を用いることを想定している。

```
1 from mpi4py import MPI
2 import ctypes
3 import xmp
4
5 lib = ctypes.CDLL('test.so')
6 xmp.init_py(lib, MPI.COMM_WORLD)
7 lib.test()
8 xmp.finalize_py(lib)
```

(a) Python のコード（test.py）

```
1 void test(){
2 #pragma xmp nodes p[*]
3   :
4 }
```

(b) XMP のコード（test.c）

　この機能を用いて、グラフ理論の問題である Order/Degree Problem に対するアプリケーションの作成を行った。Order/Degree Problem とは、与えられた頂点数とエッジ数を満たす中で、最小の平均頂点間距離を持つグラフを作成するという問題である。非常に時間を要する平均頂点間距離の計算には XMP を用いて、結果であるグラフの可視化には Python を用いるアプリケーションを作成した。このように、各言語が得意とする処理をそれぞれ作成し、それらを連携することで全体的な生産性向上に寄与できることを示した[1,7-9]

<引用文献>（研究代表者、研究分担者及び連携研究者には下線）

[1] Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato. ``A Method for Order/Degree Problem Based on Graph Symmetry and Simulated Annealing with MPI/OpenMP Parallelization'', HPC Asia, Guangzhou, China, Jan. 2019.
[2] Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato. ``Multi-accelerator extension in OpenMP based on PGAS model'', HPC Asia, Guangzhou, China, Jan. 2019.
[3] Hitoshi Murai, Mitsuhisa Sato, Masahiro Nakao, Jinpil Lee. ``Metaprogramming Framework for HPC based on the Omni Compiler Infrastructure'', 6th International Workshop on Large-scale HPC Application Modernization, Gifu, Japan, Nov. 2018.
[4] Masahiro Nakao. ``Development of parallel language XcalableMP 2.0 with high performance portability'', CCS International Symposium, Oct. 2020.
[5] Masahiro Nakao. ``Multi-Accelerator Extension in OpenMP based on PGAS Model'', CCS International Symposium, Tsukuba, Ibaraki, Japan, Oct. 2019.
[6] 中尾昌広. ``並列言語 XcalableMP に対する AT 機能の検討'', オートチューニングマイクロワークショップ, アイランドホテル浦島（愛知), 2019 年 10 月
[7] 中尾昌広, 村井均, 佐藤三久. ``MPI/OpenMP 並列によるグラフ対称性と Simulated Annealing を用いた Order/Degree 問題の一解法'', 第 167 回 HPC 研究会, 沖縄産業支援センター（沖縄), 2018 年 12 月
[8] 中尾昌広. ``Linkage of XcalableMP and Python languages for high productivity on HPC cluster system'', 第 6 回 XcalableMP ワークショップ, 筑波大学（茨城), 2018 年 11 月
[9] Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato. ``A Method for Order/Degree Problem Based on Graph Symmetry and Simulated Annealing, Graph Golf Workshop, Nov. 2018.
[10] 中尾昌広. ``AT 機能を搭載した並列プログラミング言語の提案'', オートチューニングマイクロワークショップ, 武雄温泉 ホテル春慶屋（佐賀), 2018 年 10 月
[11] 中尾昌広, 村井均, 佐藤三久. ``PGAS モデルによるマルチ GPU 対応 OpenMP コンパイラ'', 第 165 回 HPC 研究会, 熊本市国際交流会館（熊本), 2018 年 7 月

５．主な発表論文等

〔雑誌論文〕　計3件（うち査読付論文　3件／うち国際共著　3件／うちオープンアクセス　2件）

１．著者名 Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato	４．巻 0
２．論文標題 Multi-accelerator extension in OpenMP based on PGAS model	５．発行年 2019年
３．雑誌名 Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region	６．最初と最後の頁 18--25
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3293320.3293324	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 該当する

１．著者名 Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato	４．巻 0
２．論文標題 A Method for Order/Degree Problem Based on Graph Symmetry and Simulated Annealing with MPI/OpenMP Parallelization	５．発行年 2019年
３．雑誌名 Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region	６．最初と最後の頁 128--137
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3293320.3293325	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 該当する

１．著者名 Hitoshi Murai, Mitsuhisa Sato, Masahiro Nakao, Jinpil Lee	４．巻 0
２．論文標題 Metaprogramming Framework for HPC based on the Omni Compiler Infrastructure	５．発行年 2018年
３．雑誌名 2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)	６．最初と最後の頁 250--256
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/CANDARW.2018.00054	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 該当する

〔学会発表〕　計7件（うち招待講演　0件／うち国際学会　2件）

１．発表者名 Masahiro Nakao
２．発表標題 Multi-Accelerator Extension in OpenMP based on PGAS Model
３．学会等名 CCS International Symposium 2019（国際学会）
４．発表年 2019年

１．発表者名
中尾 昌広

２．発表標題
並列言語XcalableMPに対するAT機能の検討

３．学会等名
オートチューニングマイクロワークショップ2019

４．発表年
2019年

１．発表者名
中尾昌広

２．発表標題
MPI/OpenMP並列によるグラフ対称性とSimulated Annealingを用いたOrder/Degree問題の一解法

３．学会等名
情報処理学会 HPC研究会

４．発表年
2018年

１．発表者名
中尾昌広

２．発表標題
Linkage of XcalableMP and Python languages for high productivity on HPC cluster system

３．学会等名
第6回XcalableMPワークショップ

４．発表年
2018年

１．発表者名
中尾昌広

２．発表標題
AT機能を搭載した並列プログラミング言語の提案

３．学会等名
オートチューニングマイクロワークショップ2018

４．発表年
2018年

１．発表者名
中尾昌広

２．発表標題
PGASモデルによるマルチGPU対応OpenMPコンパイラ

３．学会等名
情報処理学会 HPC研究会

４．発表年
2018年

１．発表者名
中尾昌広

２．発表標題
A Method for Order/Degree Problem Based on Graph Symmetry and Simulated Annealing

３．学会等名
CANDAR Graph Golf Session（国際学会）

４．発表年
2018年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

【科研費を使用して開催した国内研究集会】
- 研究集会名：オートチューニングマイクロワークショップ
- 開催年：2018, 2019, 2020

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
研究分担者	村井　均 (Murai Hitoshi) (70590074)	国立研究開発法人理化学研究所・計算科学研究センター・上級技師 (82401)	

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