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

平成　３０　年　　６　月　　４　日現在

機関番号：　１４４０１
研究種目：　挑戦的萌芽研究
研究期間：　2015～2017
課題番号：　１５Ｋ１２００８

研究課題名（和文）遅延隠蔽指向の記述モデルによるトライブリッドプログラミングの克服

研究課題名（英文）Overcoming Tribrid Programming with Latency Hiding Oriented Description Model

研究代表者
伊野　文彦（Ino, Fumihiko）

大阪大学・情報科学研究科・教授

研究者番号：９０３４６１７２

交付決定額（研究期間全体）：（直接経費）　　2,800,000円

研究成果の概要（和文）：GPU（Graphics Processing Unit）などのアクセラレータを装備する並列計算機を対象として，その複雑な並列プログラムを容易に記述するだけでなく高い性能を得るための遅延隠蔽指向型のプログラム記述方式を検討した．具体的には，GPUのプログラム記述方式を計算ノード間に展開する方式および計算ノード間のプログラム記述方式をGPU上に展開する方式に着目し，これらを実現するためのデータ分割手法や性能モデルを開発した．これらの有効性を，ステンシル計算などの実用的な応用を用いて複数GPUシステム上で評価した．

研究成果の概要（英文）：In this work, we explored latency hiding oriented description methods that not only achieve high performance but also facilitate complicated parallel programming for parallel computers equipped with accelerators such as the graphics processing unit (GPU). In more detail, we developed a data decomposition method and a performance model required for two program description schemes: one extends a GPU-based scheme for inter-node program description and the other extends an inter-node scheme for GPU program description. The effectiveness of the proposed methods was evaluated on multi-GPU systems with practical applications such as stencil computation.

研究分野：　高性能計算

キーワード：　並列プログラミング　並列処理　GPU　性能評価

１．研究開始当初の背景

近年，並列計算機を構成するハードウェアは多層化が進み，並列プログラミングに要する労力は増加の一途である．例えば，CPU のマルチコア化は，計算ノード内を共有メモリ向けの OpenMP 指示文，ノード間を通信仕様 MPI（Message Passing Interface）で記述するハイブリッドモデルを推し進めた．さらに，メモリ遅延を隠蔽できる GPU（Graphics Processing Unit）の登場は，CPU に対して 10 倍の加速を実現したが，GPU 特有の設計と記述が必要であり，超並列処理の敷居は一層高くなっている．

この敷居を除去するために，コンパイラに対する指示文を逐次プログラムに追加するだけで GPU プログラムを自動的に生成する試み OpenACC がある．しかし，処理できるデータの大きさや達成できる性能に関して制約がある．このように，高い性能を達成するだけでなく，ハードウェア層の全体に渡る，一貫した記述は実現できていないのが現状である．

２．研究の目的

本研究の目的は，GPU において有用な遅延隠蔽指向の記述モデルが，分散メモリ型並列計算機上で広く使われているメッセージ交換モデルに取って変わることができるか否かを明らかにすることである．つまり，GPU のプログラミングモデルを，最上位層のノード間までボトムアップ状に展開すれば，一貫した記述によるプログラミング労力の軽減だけでなく，遅延隠蔽指向の記述モデルによる計算の加速が期待できるのではないかと考えた．また，上記の記述モデルに対する対極的な方法として，メッセージ交換モデルを最下位層の GPU 内部までトップダウン状に展開する記述モデルを検討する．さらに，指示文に基づく記述モデルの制約，すなわち処理可能なデータの大きさに関する制約を緩和できる並列プログラム処理系を開発する．これらの記述モデルが達成できる実効性能，およびそのためのプログラミング労力を，GPU を装備するクラスタ計算機において定量的に明らかにする．

これらの目的を達成するために，本研究では以下の課題に取り組んだ．

(1) 複数 GPU 環境向けのデータ分割手法の開発
(2) GPU 主導のノード間通信を評価するためのツールの開発
(3) 大規模データを処理できる指示文とその処理系の開発

３．研究の方法

(1) 複数 GPU 環境向けのデータ分割手法の開発

ボトムアップ方式を実現するために，単一 GPU 向けの CUDA プログラムを，そのまま複数 GPU 向けの CUDA プログラムとして実行するためのデータ分割手法を開発した．開発したデータ分割手法は，Kim らの１次元データ向けの既存手法を拡張したものである．

具体的には，一部の GPU スレッドのサンプリング実行に基づいて GPU メモリに格納しきれない大規模な多次元データを小さなセグメントに分割する（図 1）．既存手法と比較して，提案手法は小さなセグメントサイズを実現し，GPU メモリの使用量を節約するとともに，CPU・GPU 間のデータ転送量を削減する．

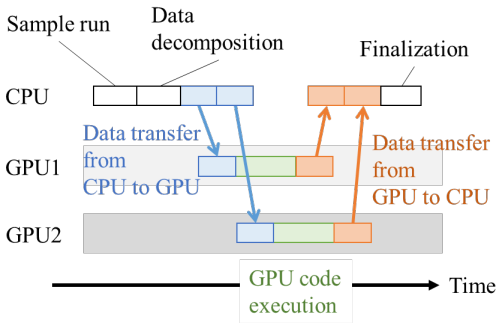

図 1:サンプル実行によるデータ分割

さらに，提案するデータ分割手法の有用性を評価するために，実用的なアプリケーションとしてコーンビーム再構成やボリュームレンダリングへの適用を試みた．

(2) GPU 主導のノード間通信を評価するためのツールの開発

メッセージ交換モデルを最下位層の GPU 内部までトップダウン状に展開する記述モデルとして，Gysi らは GPU が主体となって開始する通信手法（GPU 主導通信）およびそのためのフレームワークを提案している．

そこで，本研究では GPU 主導通信のための性能モデルや可視化ツールを構築し，GPU 主導通信の有用性を評価した．性能モデルでは，GPU 主導通信の性能上限を簡潔な式により求め，通信遅延を計算で隠蔽できるか否かを推測する．そのために，多数の並列スレッド上で実行される一連の通信命令を，命令キュー上の逐次的な通信処理としてモデル化する（図 2）．キューは GPU スレッドごとに存在し，1 つのスレッドがすべてのキューを巡回して命令を取り出すものとして，GPU 内部で発生する複雑な通信を抽象化する．

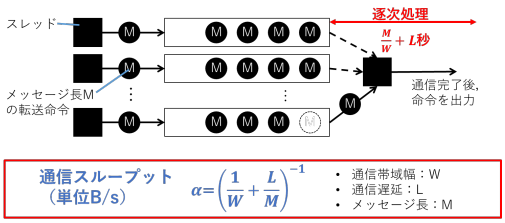

図 2: GPU 主導通信のモデル化

さらに，GPU クラスタ全体の性能ボトルネックを推測するために，よく知られたルーフラインモデルを拡張した（図 3）．典型的なル

ーフラインモデルでは，演算ならびにメモリ参照のスループットのいずれかが性能を支配する．一方，提案モデルは内部メモリ参照あるいはノード間通信が GPU クラスタの性能を支配するものとして，GPU クラスタ全体の性能上限を分析する．

このように，構築したルーフラインモデルは通信量に対する内部メモリ参照量（メモリ参照比）を因数として，GPU アプリケーションの性能ボトルネック，すなわちノード間通信あるいは内部メモリ参照のいずれかを推測する．

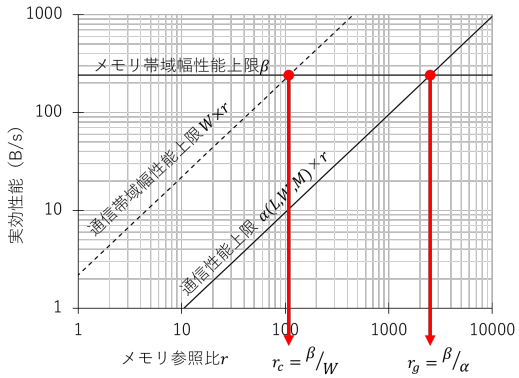

図 3:GPU 主導通信向けのルーフラインモデル

(3) 大規模データを処理できる指示文とその処理系の開発

GPU メモリ容量を超える大規模データに対するステンシル計算を指示文で実現するために，OpenACC を拡張した指示文ならびにその処理系を開発した．これにより，逐次プログラムに指示文を挿入するだけで，データ分割およびパイプライン実行を実現する GPU プログラムを自動的に生成できる．

さらに，CPU・GPU 間のデータ転送量を削減するために，データ分割およびパイプライン実行に加えて，テンポラルブロッキングを実現する指示文を開発した．ここで，テンポラルブロッキングは，キャッシュ上のデータを再利用することにより，キャッシュヒット率の向上を図る最適化手法である．CPU・GPU 間のデータ転送が全体性能を律速するステンシル計算では，テンポラルブロッキングによる転送量の削減が性能を向上させる．

４．研究成果

GPU 向けの開発環境 CUDA で記述された行列積プログラムに対し，提案するデータ分割手法ならびに既存手法を適用し，それらのメモリ使用量ならびに実効性能をマルチ GPU マシン上で評価した（図 4）．結果，提案手法は既存手法よりも GPU メモリ使用量を 80%削減し，CPU メモリに収まる程度の大規模行列を正しく処理できた．さらに，提案手法は行列の全体を CPU メモリに格納できれば，適切な分割数のもとで GPU メモリ容量を超える規模の行列を処理できた．このときの行列の大きさは，分割前と比べて 43 倍大きく，既存手法と比べて 29 倍大きかった．ただし，既存手法が処理できる規模の行列に対し，その実効性能は 28%低下した．この理由は，多次元分割に起因するインデックス変換にあり，GPU の記述モデルを最上位のノード間に効率よく展開するためには，GPU プログラムにおける変換オーバヘッドを削減する必要があることを明らかにした．

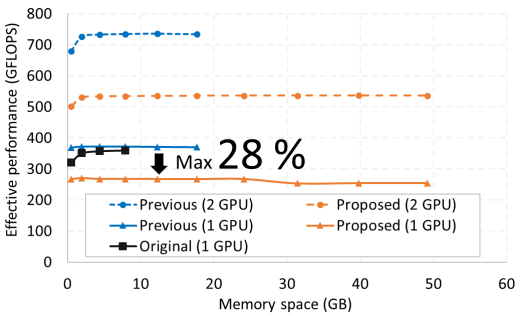

図 4:データ分割後の行列積の実効性能

次に，GPU 主導通信向けの性能モデルに関しては，Infiniband ネットワークで相互接続された 4 台の GPU クラスタにおいて提案モデルをステンシル計算に適用した．結果，提案モデルの示す性能上限に対して，実測値は 93%の高い実行効率に達することを確認した（図 5）．ただし，その高い実行効率を達成するためには，Infiniband Verbs のような低遅延の通信プロトコルを用いて通信と計算をオーバラップすることが不可欠であった．

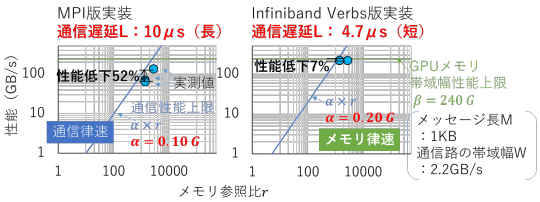

図 5:拡張ルールラインモデルの評価

メッセージ交換モデルを最下位層の GPU 内部にまで展開することの要点は，GPU プログラム内で通信命令を呼び出すことにあり，これにより通信と計算のオーバラップを GPU のスレッドスケジューラに任せることができる．結果，通信を計算とオーバラップさせるための CPU 上の複雑な記述は不要となり，GPU 関数の分割なしに，単一の GPU 関数のまま遅延を隠蔽できる．したがって，GPU 主導通信により，プログラムを簡潔に保ったまま，ノード間通信の遅延を隠蔽できる．

また，性能特性に関して GPU 主導通信を CPU 主導と比較した．性能モデルを用い，通信遅延を隠蔽するために必要なメモリ参照比を得た．具体的には，通信帯域幅ならびに GPU 主導通信スループットをそれぞれ W ならびに α とすると，GPU 主導通信は CPU 主導通信に比べ，W/α 倍のメモリ参照比が必要である．この差は Infiniband EDR 12X ならびに Tesla K80 の環境において約 22 倍の差となる．

さらに，メッセージ長，通信遅延ならびに通信帯域幅のクラスタ全体の性能に対する影響を性能モデルにより分析した．結果，典型的な環境において，通信遅延およびメッセージ長による影響が大きい一方，通信帯域幅の影響は小さいことが分かった．

最後に，拡張 OpenACC 指示文を用いて，姫野ベンチマーク，ヤコビ法および CIP 法などのステンシル計算を記述し，データの大きさを変更したときの実効性能を計測した（図 6）．結果，GPU メモリ容量の 12GB を超える，107GB のデータに対し，すべてのデータが GPU メモリ上に乗り切る場合と比べて 17%程度の性能低下に抑えることができた．

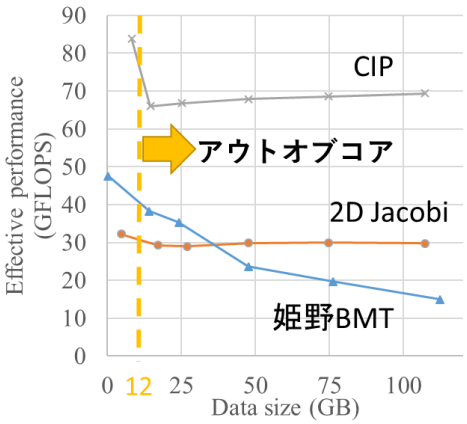

図 6:OpenACC 拡張指示文の性能評価

５．主な発表論文等

〔雑誌論文〕（計３件）

① Nobuhiro Miki, Fumihiko Ino, Kenichi Hagihara, PACC: A Directive-based Programming Framework for Out-of-Core Stencil Computation on Accelerators, Int. J. High Performance Computing and Networking, 査読有, 2018

② Yuji Misaki, Fumihiko Ino, Kenichi Hagihara, Cache-aware, In-place Rotation Method for Texture-based Volume Rendering, IEICE Trans. Information and Systems, 査読有, Vol.E100-D, No.3, 2017, pp.452–461, DOI: transinf.2016EDP7178

③ Yuechao Lu, Fumihiko Ino, Kenichi Hagihara, Cache-Aware GPU Optimization for Out-of-Core Cone Beam CT Reconstruction of High-Resolution Volumes, IEICE Trans. Information and Systems, 査読有, Vol.E99-D, No.12, 2016, pp.3060–3071, DOI: transinf.2016EDP7174

〔学会発表〕（計９件）

④ 酒井亮太郎，伊野文彦，GPU 主体のノード間通信を評価するための性能モデル，電子情報通信学会 2018 総合大会，2018

⑤ 箕畑宏宣，徳久三四郎，西口敏司，橋本渉，水谷泰治，並列プログラミングの学習において通信の可視化と性能測定を支援するツールの提案，電子情報通信学会 2017 総合大会，2017

⑥ Ryotaro Sakai, Fumihiko Ino, Kenichi Hagihara, Towards Automating Multi-dimensional Data Decomposition for Executing a Single-GPU Code on a Multi-GPU System, 4th Int. Workshop Computer Systems and Architectures, 2016

⑦ Nobuhiro Miki, Fumihiko Ino, Kenichi Hagihara, An Extension of OpenACC Directives for Out-of-Core Stencil Computation with Temporal Blocking, 3rd Workshop Accelerator Programming Using Directives, 2016

⑧ 三木脩弘，伊野文彦，萩原兼一，アウトオブコア・ステンシル計算に対する自動テンポラルブロッキングのためのアクセラレータ向けディレクティブ，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，2016

⑨ 酒井亮太郎，伊野文彦，萩原兼一，単一 GPU コードをマルチ GPU 環境で実行するための多次元データ分割手法の検討，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，2016

⑩ 田中寛章，藤井健太，磯淵郁也，水谷泰治，複数のハードウェアでの共通操作に着目した教育用並列プログラミング言語の提案，情報処理学会全国大会，2016

⑪ Ryotaro Sakai, Fumihiko Ino, Kenichi Hagihara, Preliminary Estimation on Automating Multi-dimensional Data Decomposition for Multi-GPU Systems, 2nd Annual Meeting on Advanced Computing System and Infrastructure, 2015

⑫ 三木脩弘，伊野文彦，萩原兼一，OpenACC を用いたアウトオブコア・ステンシル計算に対するテンポラルブロッキングの適用，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，2015

〔その他〕

ホームページ等

http://www-ppl.ist.osaka-u.ac.jp/

６．研究組織

(1)研究代表者

伊野　文彦（INO, Fumihiko）

大阪大学・大学院情報科学研究科・教授

研究者番号：９０３４６１７２

(2)研究分担者

水谷　泰治（MIZUTANI, Yasuharu）

大阪工業大学・情報科学部・准教授

研究者番号：１０４１１４１４