TSUBAME 共同利用 平成 26 年度 学術利用 成果報告書

利用課題名 仮想 GPU を用いた分子動力学シミュレーションコードの開発と評価
英文：Development of Molecular Dynamics Simulation Program using Virtualized GPU

利用課題責任者 泰岡 顕治
Kenji Yasuoka

所属 慶應義塾大学理工学部機械工学科
Department of Mechanical Engineering, Keio University
URL http://www.yasuoka.mech.keio.ac.jp

邦文抄録（300 字程度）
前年度から継続してGPUを使用した分子動力学シミュレーションコードの大規模並列化に取り組んだ．GPU仮想化ミドルウェアを利用してGPU1,024台で並列計算を行ったときのデータ通信のパフォーマンスを解析した結果，シミュレーション内容に関与しないダミーのデータ転送処理を挿入したときの方が転送速度の向上が見られるケースがあった． GPU 仮想化ミドルウェアの機能にダミーデータ転送処理を追加した結果，追加しない場合と比べ総合的な計算速度の並列化効率が向上することを確認した． 今年度における当初予定に含まれていた冗長計算機能およびプロセスマイグレーション機能を含む耐故障性機能に関しては，それらの機能を実装し数台の GPU を含む小規模クラスタ環境で動作テストを行ったにとどまる．

英文抄録（100 words 程度）
We continued the development of largely parallelized program of molecular dynamics simulation using multiple GPU devices. We analyzed the performance of data communication between a client node and 1,024 GPUs using our GPU virtualization middleware, then we found that the communication speed improves by inserting a dummy data transfer between a client and GPUs. By implementing a routine that automatically inserts the transfer of dummy data into our GPU virtualization middleware, we obtained the improvement of parallel efficiency for the calculation of molecular dynamics simulation. The fault-tolerant mechanism such as “redundant calculation” and “process migration between GPU devices” which we planned has not been performed on TSUBAME, but tested on our small GPU cluster consisting of several GPU devices.

Keywords: virtualization, Graphics Processing Unit, CUDA, molecular dynamics simulation

背景と目的

近年の高性能計算機には，複数のプロセッサコアを同一パッケージあるいは独立したハードウェアアクセラレータとして搭載した計算ノードを高速なネットワークで相互に接続したものが多く見られる． 多数の計算ノードから構成される計算機クラスタ向けのアプリケーションを効率よく開発し，性能を引き出すことは困難さを伴うことが多い．GPU（Graphics Processing Unit)は，名前の示すとおり画像データ処理専用に開発されたプロセッサであるが，その並列演算処理性能が注目され，高性能計算機の分野に特化したGPGPU（General Purpose GPU)向けの製品が開発され，現在では数値演算用ハードウェアアクセラレータとして広く利用されるようになった． GPUを扱うプログラムを開発するためにはCUDA（Compute Unified Device Architecture)と呼ばれるC/C++またはFortranの拡張言語，あるいはOpenACCやOpenCLとよばれるフレームワーク等を利用することが多い． またネットワークで相互に接続されている複数の計算ノード上で並列計算を行うためにMPI（Message Passing Interface）を使用することが現在主流となっている．

GPUを含む計算機クラスタ上のアプリケーションを効率よく開発・利用するためにはこれらの開発環境に通じ，クラスタ全体の計算速度性能を引き出す手法も要求されることから，計算機それ自体を専門とはしない利用者にとって開発の敷居が低いとは言えない． また大規模化がより進むと予想される将来のexascale規模のスーパーコンピュータにおいて装置全体の平均故障間隔の短さが顕著な課題になると考えられ，アプリケーションレベルでの耐故障性機能も求められている．

本プロジェクトでは，GPU仮想化ミドルウェアを開発し，それらの機能をミドルウェア上の機能として実装しユーザから隠蔽することにより上記で述べたアプリケーション開発上の困難さを低減し，かつミドルウェアの機能の1つにアプリケーションレベルでの耐故障性機能を付加することを目指した．

概要

開発した GPU 仮想化ミドルウェアを利用し GPU 1,024 台で並列計算を行ったときのデータ通信スループットおよびレイテンシ特性を解析した結果，シミュレーション内容に関与しないダミーデータ転送のある方が転送効率の改善する場合があることが分かった．GPU 仮想化ミドルウェアの機能にダミーデータ転送処理を挿入した結果，挿入しない場合と比べ総合的な計算速度の並列化効率が向上するケースのあることを確認した． 当初の目的の１つであった「冗長計算機能」と GPU 故障発生を想定した「プロセスマイグレーション機能」を含む耐故障性機能に関しては，それらの機能を実装し数台の GPUを含む小規模クラスタ環境で動作テストを行ったにとどまった． また，Fast Multipole Method （FMM） を適用した分子動力学シミュレーションコードの開発に関しては，GPU 並列時の計算精度面の課題を解消できす性能評価まで至れなかった．今後も継続予定である．

結果および考察

図 1 に，クライアントノード上の PCI express インターコネクトに接続されたローカル GPU1 台に対して CUDA API 関数の 1 つである cudaMemcpy()を，実行間隔を変えながらその実行時間の平均値，最大値，最小値を計測した結果をそれぞれ”MEAN”，”MAX”，“MIN”で示した． グラフの横軸は実行間隔を表しており，概ね 10(msec)を境として，その前後で実行時間が不連続に増加する傾向をもつことを示している．

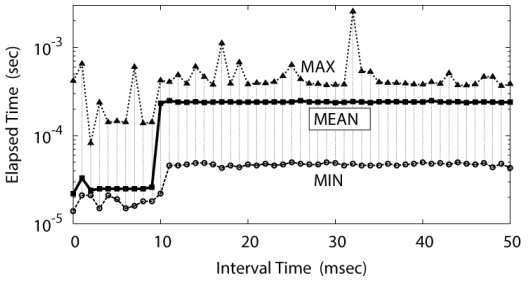

図 1 CUDA API 関数の1つである cudaMemcpy() の実行間隔への依存性の予備調査

図 1 に示したデータ転送処理時間の実行間隔への依存性を示した特性を考慮して，GPU 仮想化ミドルウェアの機能の一部に，データ転送の実行間隔が 10ms を超えないようなダミーデータ転送で挿入する変更を加えた． ダミーデータ転送の挿入の効果を確認するために，GPU 並列台数 1～1,024 台に対して巡回して繰り返し(1,2,...,n,1,2,...のように)転送を行った場合の転送1回あたりの平均実行時間を計測した結果を図 2 に示す．”(a)without internal”で示したグラフはダミーデータ転送を挿入せずにかつ実行間インターバルを開けなかった場合の結果である．GPU 台数が 32 台以上～64 台未満の範囲内に不連続に実行時間の増加する境界が見られた．これは図 2 図 1 で示した不連続点の影響と考えられる．”(b)with interval”グラフは，転送の実行間隔がいつも 10ms 以上空ける条件での実測結果である．図 1 に示した実行時間の遅い条件に相当するため(a)グラフの不連続境界の遅い方を反映した結果を表している． (c)グラフは，ダミーデータ転送を有効にした場合の結果である． ダミーデータ転送の挿入により図 1 の実行時間の短い領域で安定的に実行さえるため，(b)グラフで見られた実行時間の増加現象が解消されていることが分かる．

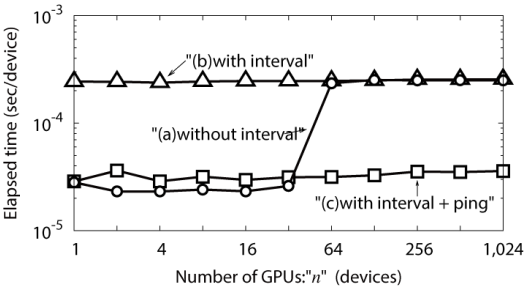

図 2 クライアントノードとリモート GPU 間の，cudaMemcpy(,, DeviceToHost) 関数の処理時間．ダミー転送の挿入有無の比較

ダミーデータ転送処理を有効にしたミドルウェアを使用して，クライアントノードと，1,024 台までのリモート GPU へのデータ転送スループットおよびレイテンシを計測した結果を，図 3 および図 4 に示す.

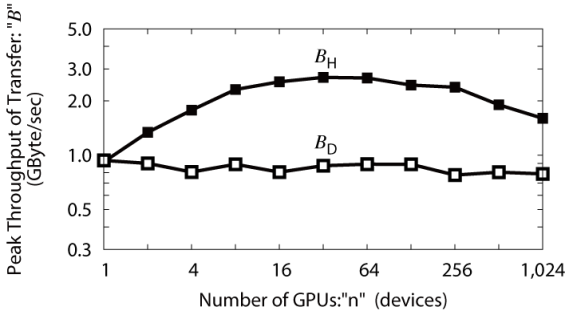

図 3 クライアントノードとリモート GPU 間のデータ転送スループットの実測結果

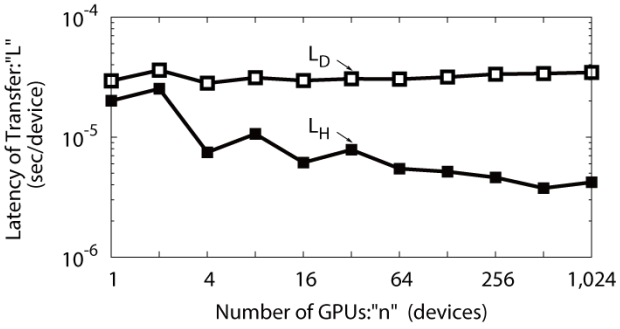

図 4 クライアントノードとリモート GPU 間のデータ転送レイテンシの実測結果

図 3 内の2つのグラフのうち，"B_H"で示したグラフがクライアントノードからリモート GPU への転送方向を，"B_D"が反対方向のリモート GPU からクライアントノードへの転送方向をそれぞれ表す． 同様に図 4 では"L_H"がクライアントノードからリモート GPU への転送方向を，"L_D"が反対方向のへの転送方向のレイテンシの実測結果を表す

ノード間はそれぞれ InfiniBand スイッチで接続され，理論最大スループット 4.0GByte/s に対して実測で最大 3.0GByte/s 程度の通信速度が出ている． "B_D"グラフには GPU 並列台数への依存傾向は見られないが，"B_H"グラフにのみ並列台数 16～64 付近で極大値を示す傾向が見られた．これは DS-CUDA がサポートする InfiniBand 通信の RDMA（Remote Direct Memory Access）機能を利用していることから，データ転送処理のオーバヘッドの一部がネットワークカード上のハードウェアにオフロードされ，ソフトとハード間で並列処理がなされたことによる転送効率が向上の影響と考えられる．一方の"B_D"グラフにその向上が見られないのは，"B_D"は常に同期的な(blocking)転送がなされることによる.

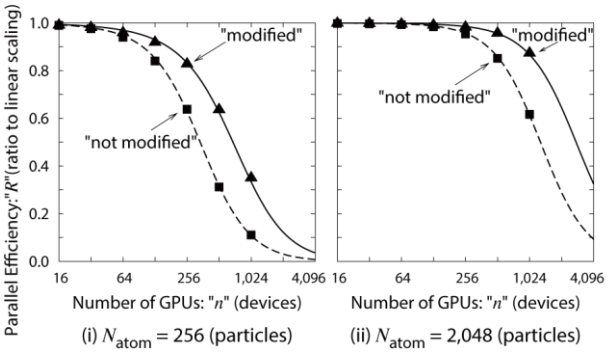

図 5 並列化効率．ダミーデータ転送処理の有無による比較

図 5 にダミーデータ転送処理の挿入有り・無しの場合において GPU を 16～1,024 台で並列化した場合の分子動力学シミュレーションの並列化効率の実測結果を示す． 左右2つのグラフはシミュレーションの計算量の小・大で2通りをそれぞれ計測したものである．最大 1,024 台の GPU 並列化のために DS-CUDA を使用している．図中の"not modified "ラベルで指し示したグラフが，ダミーデータ転送処理を挿入しなかった場合，"modified"ラベルで指し示したグラフがダミー処理を挿入した場合をそれぞれ表す．全体的な傾向として"modified"グラフの並列化効率の方が高いことから，ダミーデータ転送処理が有効なケースが存在することが分かった.

今回のケースでは，クライアントノードと GPU 間で実行されていたデータ転送の頻度が比較的低い場合には，データ通信の疎になる期間に適度なダミーデータ転送を挿入することである程度頻度を高くした方が全体的な総合的には並列化効率が向上する場合があることを示唆していると考えられる.

まとめ、今後の課題

GPU 仮想化ミドルウェアである DS-CUDA にダミーデータ転送処理を試験的に挿入することにより，並列化効率が向上するケースがあることを，実際の MD シ

ミュレーションを最大 1,024 台の GPU で並列化した場合で実験を行い確認した． 今年度当初の TSUBAME 利用目的であった GPU 仮想化ミドルウェアを使った耐故障性機能に関しては，大規模な動作テストを行うには至れず当研究室内の数台の GPU を用いたテストを行うにとどまった．

[参考文献]

[1] “1,024GPU を使用したレプリカ交換分子動力学シミュレーションの並列化”，老川稔，野村昴太郎，泰岡顕治，成見哲，情報処理学会論文誌コンピューティングシステム（ACS), 7(4), 1-14 (2014-12-16)