平成 23 年 9 月 30 日

GPU 仮想化による自動冗長計算システムの開発

電気通信大学 情報工学科 成見研究室
0821019 吉川 和幸

1 研究背景

近年、画像処理演算装置である GPU を多目的に利用する GPGPU 技術が広がりを見せている。特に最近は科学技術計算やシミュレーションのみでなく、フォトレタッチソフトや動画エンコードソフト等の一般向けソフトウェアにも用いられるようになってきており、今後ますます GPGPU の需要は拡大すると思われる。

しかし、コンシューマ向け GPU ではまれに計算ミスが発生することがわかっている。2009 年に長崎大学の濱田らが行った研究では、使用した 421 枚の GPU の内およそ 1 割にあたる 41 枚で計算ミスが発生した。[1]

本来の画面表示用途では計算ミスは次の計算結果によってすぐに置き換えられるため表面化することはほぼないが、前のステップの結果を次のステップの計算に用いるシミュレーションなどでは、以降ずっとミスが含まれた値を使用することになってしまう。

そこで本研究では、コンシューマ向け GPU を使用した GPGPU の信頼性を向上させるシステムの開発を目指す。

2 既存研究及び技術

2.1 ECC メモリ

誤った値の記録を検出し、訂正する機能がついたメモリモジュールを ECC(Error Check and Correct) メモリという。NVIDIA 社の GPGPU 専用製品である Tesla シリーズはこの ECC メモリを採用しているが、ECC メモリ未搭載機種である GeForce シリーズに比べ数倍～数十倍の価格となっており、かなり高価である。

2.2 rCUDA

NVIDIA 社が提供する GPGPU 開発環境である CUDA のリモート実行を可能とするミドルウェアとして、Antonio らの開発チームによる rCUDA[2] がある。これは CUDA ランタイムライブラリをリモート実行用のものに置き換え、ネットワークを介してサーバマシンの GPU でコードを実行するというものである（図 1）。最新のアルファ版である rCUDA3.0a では CUDA4.0 のランタイムライブラリ及び CUBLAS ライブラリのリモート実行に対応しているが、利用するには既存のソースコードを書き換える必要があり、また冗長計算などの信頼性向上手法は取り入れられていない。

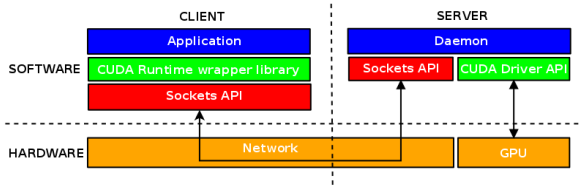

図 1 rCUDA のプログラム構成

3 研究手法

3.1 冗長計算、及びその自動化

複数の GPU に同じ計算をさせ（冗長計算）、その結果を比較することで計算ミスを検出し、信頼性を向上させる。また、ユーザがプログラムを書くなくても自動的に冗長計算を行える仕組みをシステムに盛り込む。

3.2 GPU の仮想化

実際には複数のリソースがあっても、仮想的には単一のリソースであるように見せかける手法を仮想化という。この研究では GPU を仮想化することで、通常通り一台の GPU を使用するコードを書くだけで複数の GPU を利用できるようにする仕組みを提供する。

3.3 Remote CUDA の使用

GPU の仮想化には K&F Computing Research の川井らが作成した Remote CUDA を用いる。rCUDA 同様 CUDA のリモート実行を可能とするミドルウェアであるが、既存のソースコードを変更することなく利用できるという利点がある。また自動化されてはいないが冗長計算機能も搭載されている。図 2 に Remote CUDA を用いた CUDA コードのコンパイルの概要を示す。

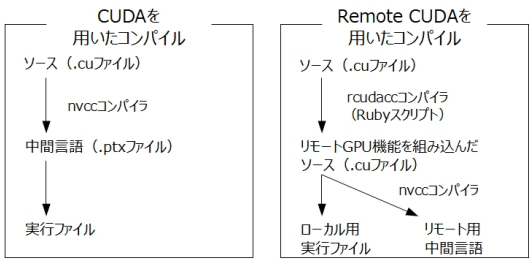

図 2 Remote CUDA でのコンパイル概要

本研究ではこの Remote CUDA を基にシステムの開発を目指す。

4 これまでの研究

自動冗長計算のために使用する Remote CUDA は、自動冗長計算機能を組み込む前にも様々な問題点がある。具体的には、ネットワークを介して実行するため実行時間が余計にかかる、新しいバージョンの API に対応していない、CUFFT や CUBLAS といった外部ライブラリの API に対応していない、などが挙げられる。そこでまずこれらの問題点について評価・改善を行った。

4.1 実行速度

Remote CUDA を使用すると、使用しない場合と比べてデータ通信にかかる時間が加わり、全体の実行時間は増加することが確認された。さらに、冗長計算をする場合は通信データ量が最低でも 2 倍になるため、さらなる実行時間の増加が予想される。

しかし、サーバでのカーネルコードの実行時間が長ければ全体の実行時間に対する通信時間の比率が下がるため、規模の大きいコードをリモート実行する際にはそこまでのパフォーマンス低下は見られないのではないかと考えられる。

4.2 CUFFT ライブラリ API の対応

高速フーリエ変換を GPGPU によって行うライブラリである、CUFFT ライブラリへの対応を試みた。現在の API 対応状況は表 1 のようになっている。

表 1　CUFFT ライブラリ API の対応状況

関数名	状況
cufftPlan1d	対応済み
cufftPlan2d	対応可能
cufftPlan3d	対応可能
cufftPlanMany	未対応
cufftDestroy	対応済み
cufftExecC2C	仮想化は完了、未動作
cufftExecR2C	未対応
cufftExecC2R	未対応
cufftExecZ2Z	未対応
cufftExecD2Z	未対応
cufftExecZ2D	未対応
cufftSetStream	未対応
cufftSetCompatibilityMode	対応可能

cufftPlan1d、cufftDestroy、cufftExecC2C の 3 つの関数があれば CUFFT のサンプルプログラムが動作するため、まずはこれらの対応を優先させた。cufftPlan1d は一次元の FFT 計算をどのように行うかという設定（プランと呼ぶ）を作成する関数で、cufftDestroy は cufftPlan 系関数で作成したプランを破棄する関数である。これら 2 つの関数は既に対応済みである。残る cufftExecC2C 関数が実際に FFT 計算をする関数であるが、GPU メモリに割り当てられたデータを使用して GPU カーネルを動作させる部分でエラーが発生してしまうため未動作である。しかしクライアント-サーバ間での計算データのやりとりをする部分については既に動作を確認しているため、原因が特定できればすぐにでも対応可能と思われる。

5 まとめと今後の研究予定

信頼性の低いコンシューマ向け GPU でも信頼できる GPGPU 計算を行うため、Remote CUDA を用いた自動冗長計算システムの開発を目標とした。現在の Remote CUDA の様々な問題点を改善するため、まずは CUFFT ライブラリへの対応を試みた。まだ GPU カーネルは動作しないが、計算データのやりとりは行えている。

今後の研究では以下を行っていく。

5.1 新たな API への対応

まず中途になっている CUFFT ライブラリの完全な対応を行う予定である。また、現状の Remote CUDA は最新バージョンである CUDA4.0 のランタイム API には未対応であるため、こちらも順次対応させていく。

5.2 冗長計算の自動化

Remote CUDA には既に冗長計算機能が組み込まれているが、計算ミスが発生した際の処理は全てユーザが記述しなければならない。他のサーバへ再接続する、再計算させるなどのエラー処理を自動化し、ユーザは既存のソースコードから何も変更しなくても自動冗長計算システムを利用できる形に改良していく。

参考文献

[1] T.Hamada, et al., 42 TFlops Hierarchical N-body Simulations on GPUs with Applications in both Astrophysics and Turbulence, 2009.

[2] Antonio J. Pena, rCUDA, 2011 http://www.hpca.uji.es/rCUDA