2H-09

重力計算アプリケーションにおけるPEACH2へのオフローディング

鶴田 千晴 †　金田 隆大 †　三木 洋平 ‡　天野 英晴 †
†慶應義塾大学大学院理工学研究科　‡筑波大計算科学研究センター

1　はじめに

近年、HPC(High Performance Computing) の分野においてヘテロジニアスコンピューティングに大きな関心が集まっている。ヘテロジニアスな構成によるスーパーコンピュータの開発も盛んに行われており、性能や消費電力の観点で優れている。本研究ではヘテロジニアスな構成のスーパーコンピュータである HA-PACS/TCA における N 体シミュレーションの一部高速化について検討する。

2　HA-PACS/TCA

2.1　HA-PACS

ヘテロジニアス構成のスーパーコンピュータの一つとして HA-PACS/TCA があり、この HA-PACS/TCA とは筑波大学計算化学研究センターにて運営・開発がなされているスーパーコンピュータである [1]。また HA-PACS/TCA は GPU クラスタであり、内部の GPU を PEACH2 と呼ばれる FPGA ボードを用いて接続することで直接通信を可能とし、データの転送ボトルネックを解消することを目的に開発されている。

2.2　PEACH2

PEACH2[2] とは PCI Express Adaptive communication Hub ver2 の略であり、Altera 社が開発している FPGA ボード、Stratix IV を用いて実装された通信機構である。PEACH2 は PCIe のポートを 4 つ用いることで他ノードと 2 重のリングネットワークを構成している。

3　ターゲットアプリケーション

3.1　N 体シミュレーションとツリー法

N 体シミュレーションとは銀河や星団等の重力多体系の時間進化を数値的に調べる方法として一般的である。N 体シミュレーションの基礎方程式は、Newton の運動方程式であり、粒子の加速度は以下の式で与えられる。

$$\boldsymbol{a}_i = \sum_{j=0,j\neq i}^{N-1} \frac{Gm_j(\boldsymbol{x}_j - \boldsymbol{x}_i)}{(|\boldsymbol{x}_j - \boldsymbol{x}_i|^2 + \epsilon^2)^{\frac{3}{2}}}.$$

ここで $\boldsymbol{x}_i$、m_j はそれぞれ粒子 i の位置、質量であり、G は重力定数、N は粒子総数、ϵ はゼロ除算を避けるためのソフトニング長である。

重力計算を実行するにあたって効率的に演算を行うためにツリー法 [3] が広く用いられている。ツリー法とは粒子間の距離を計算し、ある閾値よりも遠い粒子に関しては荒い精度で計算を行うことで計算量を抑える手法である。

3.2　Locally Essential Tree (LET)

LET とは Locally Essential Tree の略であり、これはツリー法の中で GPU 間通信を行う際の前処理で作られるツリー構造である。各 GPU のプロセス間で粒子のデータを交換する際、すべてのデータを転送すると通信量が増大するが、遠方領域の詳細な粒子分布の情報は重力計算に必要ない。そこで通信量を削減するために LET[4] という手法を用いる。これは各プロセス間の通信量を削減するためにツリー構造の配列データを間引きする方法である。

4　実装：LET generator

4.1　LET 作成をオフロードする利点

現在実装中のツリー法による N 体シミュレーションでは GPU からツリーデータを CPU へ送信し、ホストの CPU にて LET が作成されている。しかし TCA の PEACH2 による GPU 間直接通信という特徴を十分に活かせていないため LET 作成のための自作モジュール、LET generator を PEACH2 上に実装する。この実装には 4 つの利点が存在する。

- LET 作成を PEACH2 を通して GPU 間通信の際に行うため、入力されたデータを on the fly に処理することができる。
- 現在実装中のツリー法のプログラムでは LET は CPU で作られているため、LET generator を用いることで LET 作成のために CPU を経由する必要がなくなる。そのため CPU-GPU 通信をなくし、通信時間の削減が期待できる。
- CPU の代わりに PEACH2 にて LET を作成するため、LET 生成中に LET 生成とは独立な計算を実行できる。
- PEACH2 の余剰リソースに実装を行うため、ハードウェア的に追加コストを必要としない。

4.2　LET generator

図 1 に LET generator における LET 作成の簡単な流れを示す。LET generator は入力としてツリーの x, y, z 座標、質量、子ノードのデータを受け取る。入力のデータを合わせると 192 bit になるが、Avalon バスの制約により LET generator の入出力幅は 128 bit になる。そのため、各データは 2 クロックに分けて入力される。その後 mask 配列を参照し、該当するノードに 1 が立っていれば LET に追加する。LET generator は主に距離判定のモジュールとツリーのデータ更新のためのモジュールの 2 つから構成される。距離判定モジュールの内部では単精度浮動小数点演算が行われてい

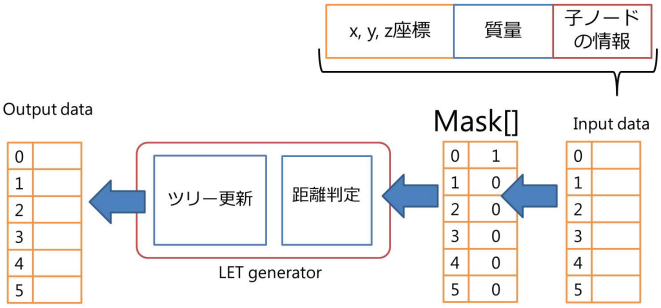

図 1: LET generator

るため、Altera 社によって提供されている浮動小数点演算用のメガファンクションを使用した。このモジュールでは 1 つのノードデータを処理するのに 136 クロックの遅延を必要とする。距離を計算した後、遠近の判定をする際の閾値には重力計算と同様に、Warren & Salmon によって提案された MAC[5][6] を採用している。遠近判定の結果、近いと判定された場合にはツリーの子ノードに関しても LET に追加をする必要があるため、mask 配列の更新を行う。この処理や、内部の変数の更新を行うために 2 つめのモジュールであるツリー更新モジュールが使われている。LET generator は mask 配列を参照してデータの投入の可否を決めているため、入力データがツリーの階層を跨ぐ場合にはツリー更新モジュールによる mask 配列の更新が終わるまで待つ必要がある。この場合ストールしてしまうが、それ以外に関して LET generator はパイプライン的に入力データを処理することができる。

5 評価

本章では LET generator に関する評価と N 体シミュレーションにおけるの LET 作成の高速化について検討する。本論文の実装は Verilog HDL を用いて設計され、Quartus II 13.1 にて論理合成を行った。その結果、ロジック使用率は 39%となり、PEACH2 単体のロジック使用率が 22%であることから 17%のオーバーヘッドで実装を行えたことが分かる。

表 1 に通信時間を測定した際の評価環境を示す。LET 作成に関して、自作モジュールと CPU での実行時間を表 2 に示す。自作モジュールの評価は RTL シミュレーションにて行い、比較対象となる CPU での実行は 10 回した実行時の平均実行時間である。LET 作成部分に関しては CPU と比較して 2.2 倍の高速化を達成した。

次に過去に測定した PEACH2 による通信遅延のデータを使用して計算をした PEACH2 による実行時間と、CPU で LET を作り、CPU から GPU へ値を送る実行時間を表 3 に示す。この表より CPU で LET を作り、送信するのに比べて 7.2 倍の高速化ができると予測している。

6 結論と今後の課題

本論文では TCA の通信機構として用いられている PEACH2 に重力計算アプリケーションの一部である LET 作成に関する処理を行う演算機構を実装した。自

表 1: 評価環境

FPGA	Stratix IV EP4SGX530NF45C2
CPU	Intel Xeon E5 2680
ICC	icc 13.1.3
MPI	Open MPI 1.6.5
CUDA	CUDA 5.5
粒子数	4096

表 2: LET generator の実行時間

	execution time
CPU	69.3 μsec
Our off-loading module	31.4 μsec

表 3: 通信時間を考慮した実行時間

	Execution time
CPU	245.7 μsec
Our off-loading module	33.7 μsec

作モジュールは一部制限があるがパイプライン化されている。そのためシミュレーション上の結果から LET を作成し、GPU に転送を行う範囲において 7.2 倍の高速化が見込まれる。今後の課題としては PEACH2 と GPU 間の通信部分を完成させ、協調動作させる予定である。後にツリー法のプログラムに組み込むことで、システム全体の性能評価を行っていきたい。

謝辞

本件研究は、JST-CREST 研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」、研究課題「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」による。

参考文献

[1] T. Hanawa, Y. Kodama, T. Boku, and M. Sato, “Interconnect for tightly coupled accelerators architecture,” ”IEEE 21st Annual Sympsium on High-Performance Interconnects (HOT Interconnects 21)”, 2013.

[2] Yuetsu Kodama, Toshihiro Hanawa, Taisuke Boku, Mitsuhisa Sato, “PEACH2: An FPGA-based PCIe network device for Tightly Coupled Accelerators,” HEART2014, pp.5–10, June 2014.

[3] Josh Barnes, Piet Hut, “A hierarchical O($NlogN$) force-calculation algorithm,” Nature, pp.446–449, Dec. 1986.

[4] Michael S. Warren, John K. Salmon, “Astrophysical N-body Simulations Using Hierarchical Tree Data Structures,” Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, pp.570–577, Sept. 1992.

[5] Michael S. Warren, John K. Salmon, “A Parallel Hashed Oct-Tree N-Body Algorithm,” Proceedings of the 1993 ACM/IEEE Conference on Supercomputing, pp.12–24, March 1993.

[6] Michael S. Warren, John K. Salmon, “Skeletons from the treecode closet,” Journal of Computational Physics (ISSN 0021-9991) vol.111, No 1, pp.136–155, March 1994.