

複数台のスマートフォンによる計算の高速化の検証

コンピュータサイエンスコース 学籍番号:1411107 成見研究室 杉田陽亮

1 はじめに

3D 映像などを高速に処理する Graphics Processing Unit(GPU)を、高い計算能力を活かして様々な用途で利用する General Purpose computing on GPU(GPGPU)という分野が発展を続けている。GPUは複雑な条件分岐などを得意としないが、同一の処理を多数実行するのに長けており、近年では画像・音声処理を始め、ディープラーニング等にも利用されている。しかし、GPUはすべてのコンピュータに搭載されている訳ではない。

一方で、近年スマートフォンは人々に広く普及し、多くの場面で利用されている。スマートフォンには計算処理を行う System on a Chip(SoC)と呼ばれるチップが搭載されている。SoCには基本的な演算を行う CPUに加えて GPUなどが集約されており、性能が向上し続けている。ただし、PC向けプロセッサには性能が劣る。

本研究では、複数のスマートフォンを利用し、計算を分散させ、GPGPUを行うことによる高速化の検証を行う。まず、Android 上でのGPGPUを実現するOpenCLを利用する環境をAndroid Studio上でJava Native Interfaceを経由しながら確立する。次に、Wi-Fi技術を利用してPeer to Peer通信を可能とするWi-Fi Directを利用し、複数の端末を接続する。最後に2つの技術を利用した並列計算環境にて行列積計算を実行し、性能評価を行う。

2 既存研究・技術

2.1 OpenCL

OpenCL(Open Computing Language)はPCや組み込み環境、モバイル端末等を問わず、並列計算を可能とするフレームワークである[1]。プログラムはC・C++を元とした記述を行い、PC向けに書かれたソースコード資源や複数の言語によるラッパー等も利用できる。

James A. Rossらによって、モバイル端末におけるOpenCLを利用したGPUによる粒子計算の高速化が行われている[2]。粒子数変化などによる計算速度の変化などが研究されているが、全体を通して一端末によって実験が行われている為、本研究とは複数端末を扱う点で異なる。

2.2 Wi-Fi Direct

Wi-Fi Directは、Wi-Fiが利用可能な端末同士を、無線ルータ等への接続なく直接接続を可能とする技術である[3]。GoogleによってAndroidにおける実装例が公開されている[4][5]。

須永らはWi-Fi Directを利用したAndroidにおける電波不通地帯でのデータ通信基盤を開発している[6]。Wi-Fi Directにより端末間接続を確立し、そのネットワークを利用して各機能を実装する点で類似しているが、本研究とは計算の高速化に焦点を当てる点で異なる。

3 実装

3.1 Wi-Fi DirectとOpenCLを用いた計算手順

システムの概要図は図1である。Wi-Fi Directにて端末を接続すると、Groupと呼ばれるネットワークが作成され、端末はGroup Owner(GO)と非GOに分かれ、GOをServer、非GOをClientとして通信を行う。

計算はServerからClientに向けて計算データを送信し、各端末でOpenCLによるGPU演算を行った後、Serverへ集約し結果を出力する。

3.2 開発環境

3.2.1 OpenCLの実行

AndroidはJavaを主とする実行環境である為、C・C++にて記述されるOpenCLを直接実行できない。ただし、Java Native Interfaceを経由することでC・C++の関数を呼び出すことが可能となる。

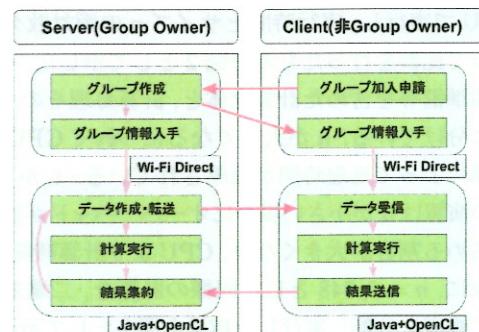


図1: Wi-Fi DirectとOpenCLを利用した計算の流れ

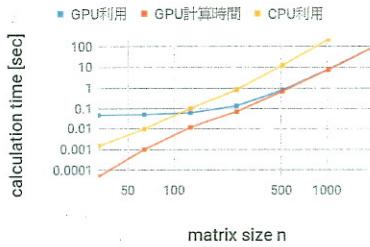


図 2: CPU と GPU の計算時間比較

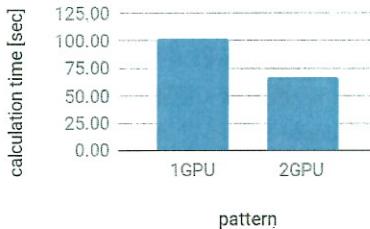


図 3: 複単端末での計算時間比較

OpenCL のコーディングはヘッダファイルに記載された関数を通して行う。コードに沿って各環境に存在するライブラリファイルが利用され、プログラムが実行される。OpenCL のライブラリファイルは libOpenCL.so であり、ビルド対象端末から取り出し、Android Studio のプロジェクト内へ格納する必要がある。

3.2.2 端末の双方向通信

Wi-Fi Direct を利用し Group を確立することでネットワークが構築される。Client は Server の IP アドレスを認識することが出来る為、これを用いて Socket を利用した TCP/IP 通信を実装出来る。

Socket 通信には byte 型配列を利用する必要がある。計算には float 型の配列を利用している為、byte 型配列へ変換するシリアル化/デシリアル化処理を行う。

4 評価

確立したシステムを利用したアプリを作成し、Nexus7 上で以下の評価を行った。

まず、一端末上で n 次正方行列の積演算を CPU と GPU で実行し、実行時間とサイズ n の両対数グラフによって比較を行ったものを図 2 として示した。GPU は事前準備等を含めた計算全体と、計算処理そのものの 2 つに分けています。 n が大きくなるにつれて GPU による計算によって処理時間が短縮されている。しかし、GPU での演算は n が小さい場合にオーバーヘッドが計算時間を占める割合が大きくなり、CPU より計算時間が劣る。

次に、 $n = 2,048$ とした同様の計算を、二端末を利用して処理を分割し実行した結果を図 3 として示した。また、この計算における Server 側端末の各処理にかかった時間の割合を図 4 として示した。図 3 より、2 端末に

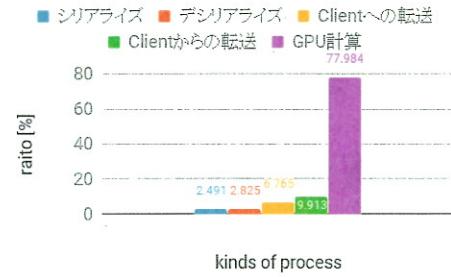


図 4: Server 内の処理にかかる時間の割合

処理を分割することで、1 端末と比較して約 1.5 倍の高速化を実現した。図 4 では、全体の計算時間において、データを他の端末へ転送する処理で約 17%、データを byte 型配列へ変換する処理で約 5% かかっている。

最後に、PC 向けの CPU である Intel Core i5-4460 にて $n = 2,048$ とした行列積演算を Java を利用したシングルスレッドにて行い、その結果と 2 端末にて計算した本システムを比較すると、CPU の約 1.1 倍の性能となることを確認した。

5 まとめと今後の課題

OpenCL と Wi-Fi Direct を利用した複数 Android 端末による GPGPU システムを実現し、その実行時間を測定した。データの転送・変換に多くの処理時間を要しているため、最適化することで更なる高速化が望める。端末通信の基盤となる環境を確立し、3 台以上利用した負荷分散も期待される。

GPU に向けたコードの最適化は行えていないことから、内部メモリを有効に扱うことで更なる高速化が期待できる。また、異なる端末を扱う場合に性能差を考慮した計算タスク割り振りや、更に多くの端末を扱う場合にブロードキャストを利用すること、転送データのシリアル化手法の改良等の検討が必要である。

参考文献

- [1] "OpenCL Overview", <https://www.khronos.org/opencl/> (最終アクセス日 2018 年 1 月 11 日)
- [2] J. A. Ross, D. A. Richie, S. J. Park, D. R. Shires and L. L. Pollock, "A case study of OpenCL on an Android mobile GPU," 2014 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, 2014, pp. 1-6.
- [3] "Wi-Fi Direct", <https://www.wi-fi.org/ja/discover-wi-fi/wi-fi-direct> (最終アクセス日 2018 年 1 月 11 日)
- [4] "Wi-Fi Peer-to-Peer", [https://developer.android.com/guide/topics/connectivity/wifip2p.html/](https://developer.android.com/guide/topics/connectivity/wifip2p.html) (最終アクセス日 2018 年 1 月 9 日)
- [5] "WIFIDirectDemo", <https://android.googlesource.com/platform/development/+/master/samples/WiFiDirectDemo?autodive=0> (最終アクセス日 2018 年 1 月 9 日)
- [6] 須永 宏、加藤 将太, "Wi-Fi Direct を利用した Android 間直通信のためのサービスコンピューティングプラットフォーム", 電子情報通信学会技術研究報告 113(496), pp.19-24, 2014