7G-08

FPGAアクセラレーションを可能にするストリーミングフレームワークの評価

味曽野 智礼† 吉瀬 謙二‡
†東京工業大学 大学院情報理工学系研究科
‡東京工業大学 情報理工学院
misono@arch.cs.titech.ac.jp, kise@c.titech.ac.jp

1 はじめに

近年FPGAによるアクセラレータの研究が多く行われている。これは特定アプリケーションにおいてFPGAアクセラレータが消費電力を抑えながら高い性能を達成するためである。特にFPGAアクセラレータはストリーミング処理に向いていると言われており、リアルタイムデータ処理の高速化が期待できる。一方で近年ストリーミング処理を行なうソフトウェアフレームワークがいくつか提案されているが、それらからFPGAアクセラレータを簡単に使用することは出来ない。そこで本稿ではストリーミングフレームワークの1つであるHeronを拡張し、FPGAリソースを扱えるようにする。PCとFPGAボード間の通信はPCIeを利用する。そしてユーザーアプリケーションをCPUのみで実行した時と、FPGAを用いて実行した時の実行速度、消費電力を比較し、定量的に評価する。

2 背景

2.1 Heron

Heron[1]はTwitter社により開発されているオープンソースの分散ストリーミングフレームワークであり、クラスタ管理にMesos[2]を、ジョブスケジューラとしてAurora[3]を使用する。

Heronではユーザーアプリケーションを Topologyと呼ばれる非循環グラフとして記述する。図1にTopologyの例を示す。図中で丸で示されているノードはSpoutと呼ばれ、データ源としてデータを出力する。四角で示されているノードはBoltと呼ばれ、データを受け取りその処理を行なう。Boltは更に次のBoltにデータを出力しても良い。データの受け渡しの単位はユーザーが自由に定義することが可能である。

TopologyはJavaかPythonで記述される。それぞれのSpoutやBoltはクラスであり、Heronが提供する抽象クラスを継承する。例えばBoltクラスならば新たなデータが来る度に呼ばれるexecuteメソッドがあり、このメソッドをオーバーライドすることでユーザー独自の処理を記述する。全体のTopologyの定義はMain関数の中で行う。

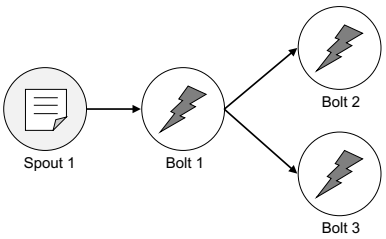

図1: Heron topologyの例

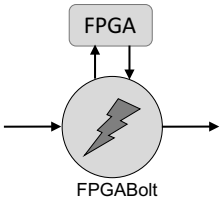

図2: 提案するFPGABoltクラス

2.2 ホストPC-FPGA間の通信

多くのFPGA評価ボードはPCIeインタフェースやEthernetインタフェースが付属している。本稿ではホストPCとFPGAがPCIeを介して接続されている環境を想定する。ここでPC-FPGA間の通信にはRIFFA[4]を用いる。これはオープンソースのPCIeインタフェース(FPGAのPCIeパケット処理回路およびデバイスドライバ)を提供し、研究目的に無償で使用することが可能である。RIFFAはXilinx、Alteraの様々なFPGAボードを対象としており、最大でGen2x8の通信を行なうことが出来る。

3 フレームワークの拡張

特定のHeronのBoltからFPGAを使用できるようにフレームワークの拡張を行う。主要な変更点は以下の通りである。

FPGAリソースの追加

CPUやメモリに加えて、FPGAを新たなリソースとしてフレームワークが扱えるようにする。Mesosにはカスタムリソース追加の機能があるためこれを使い、コードの修正は行わない。一方AuroraとHeronに関してはコードの一部の修正を行い、FPGAリソースの

Evaluation of Streaming Framework Which Enables FPGA Acceleration
Tomohiro MISONO† and Kenji KISE‡
†Graduate School of Information Science and Engineering Tokyo Institute of Technology
‡ Department of Computer Science Tokyo Institute of Technology
misono@arch.cs.titech.ac.jp, kise@c.titech.ac.jp

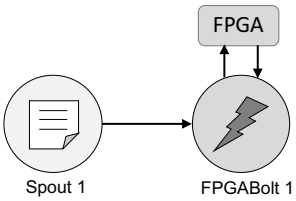

図 3: 評価 Topology

要求を行なうことができるようにする。

FPGABolt 抽象クラスの追加

新たな Bolt 抽象クラスとして FPGABolt 抽象クラスを追加し、FPGA へのデータ送信/受信を行なうメソッドを用意する (図 2)。このクラスを継承した Bolt がある場合、Heron は要求するリソースに自動的に FPGA を追加する。ユーザーは FPGA を使用する Bolt を定義する際にこの抽象クラスを継承させ、execute メソッドにおいて FPGA とのデータの送受信を行う。

FPGA との通信を行なうコードの追加

実際に RIFFA ドライバを用いて FPGA との通信を行なう部分はユーザーの Topology から見えない部分に記述する。これにより、将来的に FPGA 通信のバックエンドの変更がある場合の対応を容易にする。

今回の初期実装では 1 台の PC に 1 台の FPGA ボードのみ接続されることを前提にしており、ある FPGABolt が実行中はそれが FPGA を専有する。また簡単のため FPGA の動的コンフィギュレーションはサポートしておらず、事前に FPGA に対してコンフィギュレーションを行なうこととする。

4 評価

4.1 評価環境

拡張したフレームワークをインストールした 1 台の仮想マシン上で評価を行う。OS は Ubuntu 14.04、CPU は Xeon E5 2650 v2 (2.60GHz) であり、Xilinx Virtex-7 を搭載する vc707 評価ボード (PCIe Gen2x8) を 1 台接続する。CPU の VT-x および VT-d を有効にし、PCIe パススルーを用いて仮想マシンから FPGA ボードを使用する。

4.2 アプリケーション

評価アプリケーションとして 8 ビットグレースケール RAW 画像に対する 5 点メディアンフィルタを使用する。FPGA アクセラレータのコア回路の実装には Vivado HLS 15.4 を用い、プロジェクト全体は Vivado 15.4 で論理合成を行なう。

評価 Topology は Java で記述し、画像データを出力する Spout を 1 つ、メディアンフィルタの処理を行う Bolt1 つから構成する (図 3)。ただし Spout から出力する画像データは機械的に生成した値とする。

4.3 結果

メディアンフィルタの処理を CPU1 スレッドで行う場合と FPGA を用いる場合のそれぞれにおいて、10 分間動作させた時の、Bolt の execute メソッド 1 回分の平均実行時間を表 1 に示す。これはすなわち 1 枚の画像に対する平均処理時間である。 FPGA による実行結果とソフトウェアによる実行結果の値が一致する事を確認した。結果より、FPGA アクセラレータを用いることにより最大で約 370 倍の高速化が達成されることが分かる。メディアンフィルタでは各要素を中心とする領域毎にソートを行なうため CPU の処理コストが高く、各要素の値を並列に計算する FPGA アクセラレータが高い性能を達成したと考えられる。

表 1: メディアンフィルタ処理を行なう Bolt の execute メソッドの 10 分間の平均実行時間 (ms)

サイズ	512x512	1024x1024
CPU	151	688
FPGA	0.59	1.83

表 2: 1 枚分のメディアンフィルタ処理のみを行なう時の実行時間と消費エネルギー (サイズ:1024x1024)

	実行時間 [ms]	消費エネルギー [J]
CPU	671	22
FPGA	1.73	0.038

また参考として、Heron には組み込まずに 1024x1024 サイズのフィルタ処理 1 枚を行う時の実行時間と消費エネルギーを表 2 に示す。ただし消費エネルギーの FPGA の値はコンセントに接続したワットメーターの示す電力値 (22W) に、処理時間をかけ合わせることで求めた大まかな値であり、データ転送を行なう CPU で使用されるエネルギーは考慮していない。一方 CPU(1 スレッド) は Java プログラムであり、Intel Vtune 2017 の HotSpot を用いて対応する関数部の EnergyPack の値を計測した。表 2 より CPU、FPGA 共に実行時間は Heron 上で動作させたときとほぼ変わらないことが分かる。また、この場合の消費エネルギーの比較では FPGA の方が約 570 倍効率が良いことが分かる。

5 まとめ

ストリーミングフレームワークである Heron を拡張し、FPGA アクセラレータが使用できることを仮想マシン上で確認した。メディアンフィルタアプリケーションによる評価を行い、FPGA による高速化が実現できることを示した。今後の目標はクラスタ環境での評価や、システム全体の電力量の評価を行なうことである。

参考文献

[1] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy, and Siddarth Taneja. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 239–250, New York, NY, USA, 2015. ACM.

[2] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for fine-grained resource sharing in the data center. Technical report, 2010.

[3] Apache Aurora. http://aurora.apache.org/.

[4] M. Jacobsen and R. Kastner. Riffa 2.0: A reusable integration framework for fpga accelerators. In *2013 23rd International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8, Sept 2013.