2013 年 5 月 13 日 情報システム基盤学合同輪講発表資料

A Portable High-Productivity Approach to Program Heterogeneous Systems

著者： Zeki Bozkus, Basilio B. Fraguela

出典： *Int'l Heterogeneity in Computing Workshop (HSW 2012) with IPDPS 2012, pp.1-11, 2012.*

発表者： 高性能コンピューティング学講座 本多・近藤研究室　1353017 竹本拓未

1　はじめに

CPU と GPU などの異なった種類の演算デバイスが混在するヘテロジニアスシステムを活用することに注目が集まってきている。しかしそのためには、CPU 上でのプログラミングを行う際には必要としなかった多くの努力が必要となる。CPU 以外のデバイスで演算を行うためには、新しいプログラミング言語やツール、デバイス間での通信やメモリの管理などの手法を習得しなければならない [1]。さらに、これらの手法はベンダやデバイスに特有のものとなっているため、作成したプログラムの移植性が低くなってしまうという問題も存在する。

そこで本研究では、ヘテロジニアスシステムでのプログラミングのための、生産性と移植性を兼ね備えた新たなライブラリベースの手法を提案する。

2　OpenCL

OpenCL は、ヘテロジニアスシステムでの並列コンピューティングを実現するためのクロスプラットフォームなフレームワークである [2]。OpenCL は、CPU が各デバイスと通信するための API を備えている。また、OpenCL のコンパイラは各デバイス用のコンパイラを用いて、各デバイス上で実行可能なバイナリファイルを生成するため、作成するプログラムはベンダやデバイスに依存しない。これらの点から、現在 OpenCL はヘテロジニアス・コンピューティング手法のオープンスタンダードとなっている。

一方で、OpenCL のプログラミング・インタフェースは非常に複雑であるため、プログラマビリティが低いという問題が存在する。

3　提案手法

プログラムの移植性を損なうことなく、各デバイスの演算能力を活用できるように生産性の改善を図ることが目的である。そのために新たなライブラリ HPL（Heterogeneous Programming Library）を設計、実装する。ライブラリの効果は、プログラマビリティと実行性能、移植性について OpenCL と比較することで評価する。

4　HPL

4.1　HPL の概要

HPL は、ヘテロジニアスシステムでの並列コンピューティングを実現するための C++用ライブラリである。現在の実装では、HPL は OpenCL プログラムに変換され、OpenCL コンパイラによって各デバイスで実行されるバイナリファイルが生成される。そのため、HPL は OpenCL をサポートしている任意のデバイスを演算に使用することが可能である。

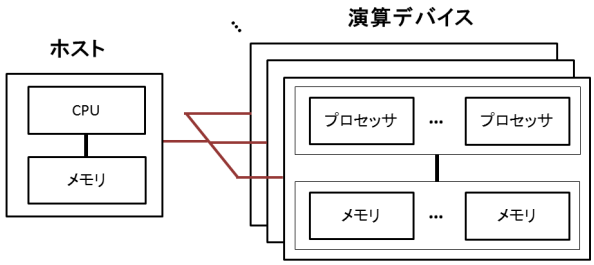

図 1: HPL のハードウェアモデル

図 1 は HPL のハードウェアモデルを示している。CPU とメモリを持つホストが一台有り、それにプロセッサとメモリを一つ以上持つ演算デバイスが複数台接続されている。このモデルは、移植性を向上させるために、OpenCL によって提案されたものをさらに簡略化している。さらに、HPL は演算デバイス上の 3 種類のメモリ（グローバルメモリ、コンスタントメモリ、ローカルメモリ）を区別してデータを割り当てることが可能である。これらの容量やアクセス速度の異なるメモリを使い分けることで、演算性能の向上が図れる。

演算デバイス上で処理を行うためにはカーネルと呼ばれる関数を呼び出す必要がある。カーネルは処理を実行するスレッドを処理によって必要となる数だけ生成する。これらのスレッドはカーネルと同じ次元数のユニークな識別子が割り当てられる。また、ユーザーは任意のスレッドにカーネルを分割し、グループ化をすることが可能である。6 × 6 個のスレッドを 2 分割して、ユーザーが指定した 2 つの演算デバイスにマッピングする例を図 2 に示す。

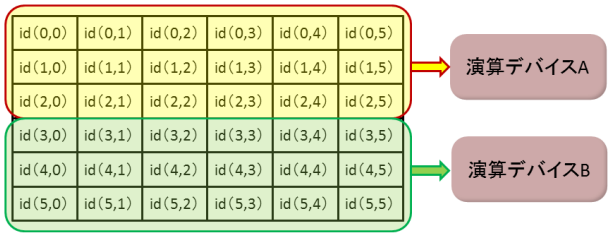

図 2: HPL カーネル内のスレッド分割・グループ化の例

```
1: #include “HPL.h”
2: using namespace HPL;
3:
4: int N = 1000;
5: double result[N];
6: Array<double,1,Global> x(N), y(N,result);
7:
8: void func(Array<double,1> y, Array<double,1> x, Double a) {
9:    y[id] = a * x[id] + y[id];
10: }
11:
12: int main() {
13:   Double a;
14:   // a、x、yに値を入れる処理
15:   eval(func)(y,x,a);
16: }
```

図 3: HPL によるベクトル計算のコード

4.2 HPL の構文

HPL の構文を用いたベクトル計算 $Y = aX + Y$ のコードの例を図 3 に示す。8 行目から 10 行目がカーネル関数部分である。このように HPL はカーネルのロードやコンパイルをカプセル化して、プログラマビリティを高めている。

5 評価

評価は、OpenCL と HPL で書いた 5 種類のベンチマークを用いて、プログラマビリティと実行性能、移植性について評価した。

表 1 は、コメントや空行を除いたコードの行数を計測したものである。表 1 から、HPL は OpenCL よりも最大で約 10 分の 1 にまで行数が短縮できていることが分かる。

表 1: SLOC

benchmark	OpenCL	HPL	Reduction
EP	1151	281	75.6 %
Floyd-Warshall	1170	107	90.9 %
Matrix transpose	455	52	88.6 %
Spmv	1637	517	68.4 %
Reduction	773	218	71.8 %

図 4 は、CPU による逐次実行との実行性能比を表している。最も実行性能差が生じたベンチマークは transpose で、4 %以下であった。

図 5 は、2 種類の異なったデバイスを使用して OpenCL と HPL で書かれたベンチマークを比較したもので、OpenCL に対する HPL のオーバーヘッドを表している。全てのベンチマークで、実行性能差は 2 ポイント以下であり、移植性が高いことが分かった。

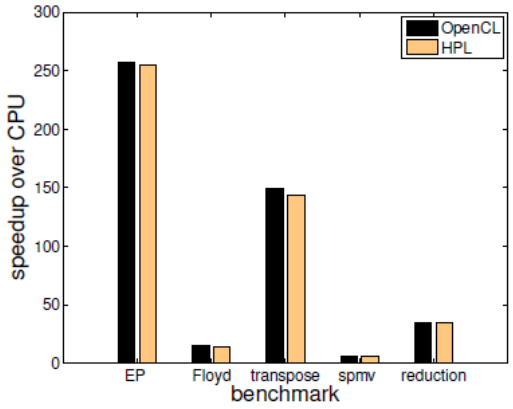

図 4: CPU による逐次実行との実行性能比

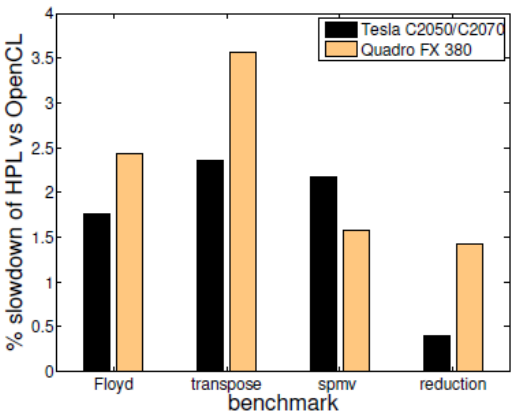

図 5: 異種デバイスでの OpenCL との実行性能比

6 おわりに

本研究では、ヘテロジニアス・コンピューティングのための移植性と生産性を兼ね備えた新たなライブラリ HPL を設計、実装した。これにより、OpenCL と比較して、実行性能の低下は 5 %以下に抑えたうえで、約 3 倍〜10 倍の生産性を実現できることが示された。

参考文献

[1] O. S. Lawlor, “Embedding OpenCL in C++ for Expressive GPU Programming,” in Proc. 5th Intl. Workshop on Domain- Specific Languages and High-Level Frameworks for High Performance Computing (WOLFHPC 2011), May 2011.

[2] The Khronos Group. OpenCL - the open standard for parallel programming of heterogeneous systems -, May 2013. http://www.khronos.org/opencl/.