第6章 データ・コピー処理を例にデータ入力/出力の処理を最適化する

最適化①… 処理ベクタ長のチューニング

本橋 弘臣

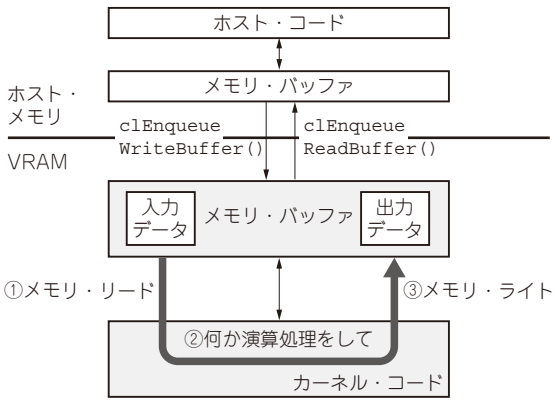

図1 カーネル・コード実行時のデータの流れ

● カーネル・コードの実行時の基本動作

図1にカーネル・コード実行時のデータの流れを示します．GPUによる演算処理を単純して考えると，次の基本動作の組み合わせになります．

① VRAMから入力データを読み出す
② GPU内のプロセッサ・エレメント（PE）で演算処理を行う
③ VRAMに出力データを書き込む

全ての処理を最適に高速化することができれば，GPU演算処理全体が最短の時間で実行可能になります．

ここで②で実行する処理の中で最も軽い処理は何なのかを考えてみると，GPUでは何も演算処理は行わずにVRAMから読み出した入力データをそのまま出力する，言い方を変えるとデータ・コピー処理になります．

● データ入力/出力の処理を最適化することが重要

GPU演算処理を最適化する際には，カーネル・コードでの演算アルゴリズムを工夫したり，共有メモリの使い方などを改善したりすることも重要です．しかし，仮に②の処理を最適化できたとしても，①と③の処理が最適化されていないままだと，①/③の部分がボトルネックとなってしまい，GPU演算処理全体としては高速になりません．つまり，まずは足回りとなる①と③の処理を高速化/最適化しておく必要があるということです．

そこで，本稿ではデータ・コピー処理を行うカーネル・コードを題材として取り上げます．OpenCLで利用可能なさまざまな仕組みを活用することで，処理性能（ここではデータ・コピー速度）が大きく変化します．

コード①…データ・コピー処理

・データ型：uint，・データ個数：8

● uintデータを8個コピーするプログラム

カーネル・コードでuintのデータを8個コピーするプログラムです．プログラムはchap6/opencl-copy-uintに格納されています．

▶カーネル・コードの内容

リスト1(a)にカーネル・コードを示します．X方向のグローバルIDを参照して，各スレッドごとにコピーするべきデータを決定しています．

▶ホスト・コードの内容

ホスト・コードのうち，主要となる部分を**リスト1**(b)に示します．GPUからの出力データをホスト・メモリ側にコピーした後で，GPUへの入力データと出力データを比較して完全一致していることをチェックしています．

● 実行結果

実行結果を次に示します．

```
$ ./main↵
     :
[Kernel] dest[0] <- src[0]
[Kernel] dest[1] <- src[1]
[Kernel] dest[2] <- src[2]
[Kernel] dest[3] <- src[3]
[Kernel] dest[4] <- src[4]
[Kernel] dest[5] <- src[5]
```