# GPUのPTXコードを用いた ランダムアドレスシフトの厳密評価

広島大学　藤田 徹

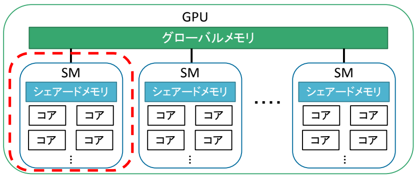
## 研究概要

GPUのアセンブリ言語であるPTXコードを用いて，メモリアクセスの高速化手法であるランダムアドレスシフトの厳密な性能評価を実施
- 命令をPTXコードで記述することによって，ランダムアドレスシフトの実行クロックサイクル数を計測

GPU（Graphics Processing Unit）
- グラフィックス処理に適したハードウェア
- 内部に多数のコアを搭載し，並列演算能力に優れる
- NVIDIA社が提供するGPUの統合開発環境CUDAを利用

## GPUアーキテクチャ

NVIDIA社のGPUには複数のストリーミングマルチプロセッサ（SM）が搭載されている




- 複数のコアとシェアードメモリから構成
- シェアードメモリは同じSM内のコアのみアクセス可能

各コアにはスレッドが割り当てられる
- ➢ 処理はスレッドを32個毎にまとめたワープ単位で実行される

今回は1つのSMのみに注目して計測を行う

## シェアードメモリとバンクコンフリクト

シェアードメモリは1ワードずつ32個のバンクによって構成されている

複数のスレッドが同じバンクの異なるアドレスに同時にアクセス
- ➢アクセスが逐次化され，スループットが低下 ➡ バンクコンフリクト

(例)バンク数4の場合

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| スレッド0 | スレッド1 | スレッド2 | スレッド3 |

衝突は発生しない

同時にアクセス

## シェアードメモリとバンクコンフリクト

シェアードメモリは1ワードずつ32個のバンクから構成されている

複数のスレッドが同じバンクの異なるアドレスに同時にアクセス
- ➢アクセスが逐次化され，スループットが低下 ➡ バンクコンフリクト

(例)バンク数4の場合

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| スレッド0 | スレッド1 | スレッド2 | スレッド3 |

衝突が発生

アクセスが逐次化

論文名：The Random Address Shift to Reduce the Memory Access Congestion on the Discrete Memory Machine
著者：Koji Nakano, Susumu Matsumae, and Yasuaki Ito
出典：International Symposium on Computing and Networking, pp. 95–103, Dec. 2013

## ランダムアドレスシフト

シェアードメモリに格納する要素をシフト数にしたがって行方向にサイクリックシフト

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |
| スレッド0 | スレッド1 | スレッド2 | スレッド3 |

アクセスの衝突回数：4

| シフト数 |
|---|
| 3 |
| 1 |
| 3 |
| 2 |

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 1 | 2 | 3 | 0 |
| 7 | 4 | 5 | 6 |
| 9 | 10 | 11 | 8 |
| 14 | 15 | 12 | 13 |
| スレッド0 | スレッド1 | スレッド2 | スレッド3 |

アクセスの衝突回数：2

アドレスシフトを行うことによってバンクコンフリクトが低減

## シフト数の決定方法

Random Address Shift
- ✓各行のシフト数は独立した乱数によって決定される

| シフト数 | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 3 | 1 | 2 | 3 | 0 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 2 | 14 | 15 | 12 | 13 |

Random Address Permute-Shift
- ✓ランダムに置換した数列に従ってシフトする
- ✓各行のシフト数はすべて異なる

| シフト数 | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 2 | 2 | 3 | 0 | 1 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 0 | 12 | 13 | 14 | 15 |

## アクセス時間の計測

GPU : GeForce GTX 680
- 1SM当たりのコア数 : 192
- 動作クロック : 1006MHz

次の4つのアクセスパターンについてワープ数を1から32まで変化させて1000回計測を行い，平均時間を算出
- Contiguous
- Stride
- Diagonal
- Random

アクセスに必要な実行クロックサイクル数を計測することで厳密な性能評価を実施

## PTXコードを用いたクロックサイクル数の計測方法

GPUのアセンブリ言語
- インラインアセンブラを用いてソースコード内に記述できる

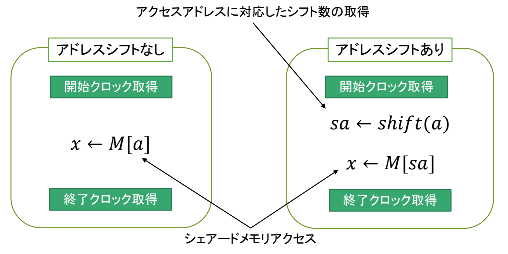
```
asm volatile (
        "mov.u32 %%a0, %3;¥n¥t"
        "mov.u32 %%a1, sh_mem;¥n¥t"
        "bar.sync 0;¥n¥t"
        "mov.u32 %0, %%clock;¥n¥t"
        "shl.b32 %%a2, %%a0, 3;¥n¥t"
        "add.u32 %%a2, %%a1, %%a2;¥n¥t"
        "ld.volatile.shared.u64 %2, [%%a2];¥n¥t"
        "mov.u32 %1, %%clock;¥n¥t"
        :"=r"(begin), "=r"(end), "=l"(temp)
        :"r"(AID[threadIdx.x])
);
```

この範囲のクロックサイクル数を計測

クロックレジスタの値を取得
- ➢ クロックカウンタの値が格納されている

測定したい命令をクロックカウンタの取得命令で挟み，その差分を求めることで命令の実行クロックサイクル数が計測できる

## 計測範囲




## 計測するアクセスパターン

1. Contiguous Access
   - 各スレッドは行方向にアクセス

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

アクセスは衝突しない

- Random Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 3 | 1 | 2 | 3 | 0 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 2 | 14 | 15 | 12 | 13 |

アクセスは衝突しない

- Permute Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 2 | 2 | 3 | 0 | 1 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 0 | 12 | 13 | 14 | 15 |

アクセスは衝突しない

## 計測するアクセスパターン

2. Stride Access
   - 各スレッドは列方向にアクセス

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

全てのスレッドでアクセスが衝突

- Random Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 3 | 1 | 2 | 3 | 0 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 2 | 14 | 15 | 12 | 13 |

一部のスレッドでアクセスが衝突

- Permute Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 2 | 2 | 3 | 0 | 1 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 0 | 12 | 13 | 14 | 15 |

アクセスは衝突しない

# 計測するアクセスパターン

3. Diagonal Access
   ◦ 各スレッドは斜め方向にアクセス

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

アクセスは衝突しない

- Random Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 3 | 1 | 2 | 3 | 0 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 2 | 14 | 15 | 12 | 13 |

一部のスレッドでアクセスが衝突

- Permute Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 2 | 2 | 3 | 0 | 1 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 0 | 12 | 13 | 14 | 15 |

一部のスレッドでアクセスが衝突

# 計測するアクセスパターン

4. Random Access
   ◦ 各スレッドはランダムにアクセス

| バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

一部のスレッドでアクセスが衝突

- Random Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 3 | 1 | 2 | 3 | 0 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 2 | 14 | 15 | 12 | 13 |

一部のスレッドでアクセスが衝突

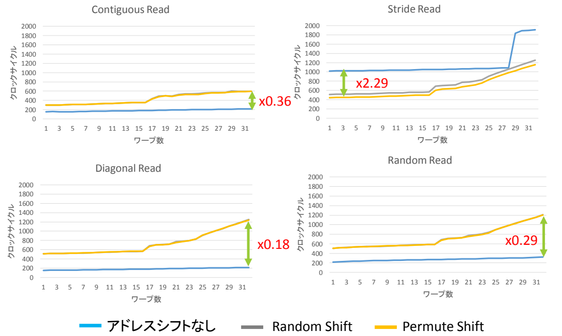
- Permute Shift

| | バンク0 | バンク1 | バンク2 | バンク3 |
|---|---|---|---|---|
| 2 | 2 | 3 | 0 | 1 |
| 1 | 7 | 4 | 5 | 6 |
| 3 | 9 | 10 | 11 | 8 |
| 0 | 12 | 13 | 14 | 15 |

一部のスレッドでアクセスが衝突

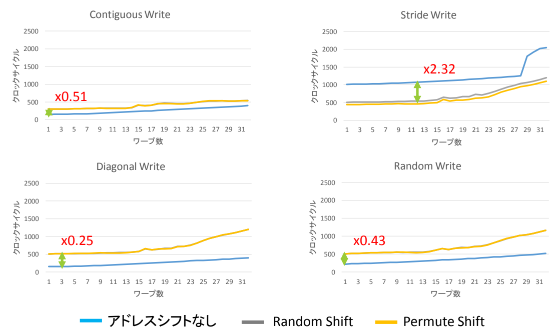
# 計測結果（read命令）

アドレスシフトを行わない場合に対する高速化率




# 計測結果（write命令）

アドレスシフトを行わない場合に対する高速化率




# 計測範囲の変更

アドレスシフトありの場合の計測範囲をメモリアクセスのみに変更する

**アドレスシフトあり**

開始クロック取得

$sa \leftarrow shift(a)$

$x \leftarrow M[sa]$

終了クロック取得

➡

**アドレスシフトあり**

$sa \leftarrow shift(a)$

開始クロック取得

$x \leftarrow M[sa]$

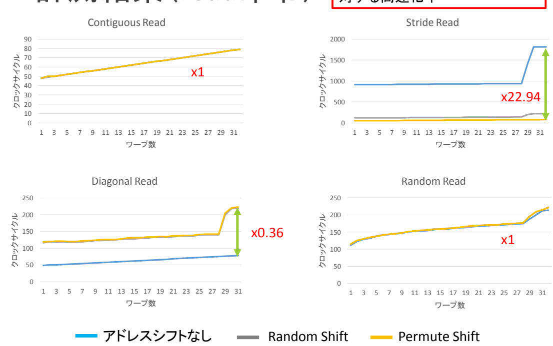
終了クロック取得

アドレスシフトがハードウェア化された場合を想定

➢ アドレスシフトの計算時間を無視できる

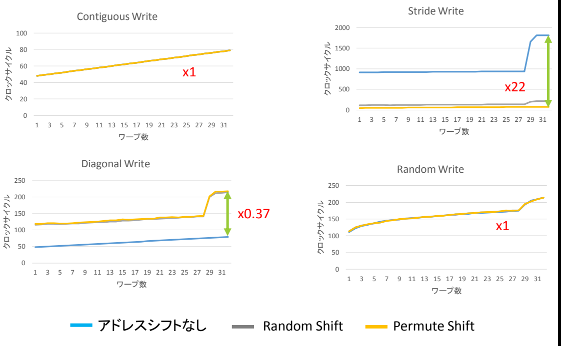
# 計測結果（read命令）

アドレスシフトを行わない場合に対する高速化率



# 計測結果（write命令）

アドレスシフトを行わない場合に対する高速化率




# まとめ

GPUのアセンブリ言語であるPTXコードを用いて，ランダムアドレスシフトの厳密な性能評価を行った

アドレスシフトを求める時間を含めた場合

- strideアクセスで約2.3倍の高速化となり，その他のアクセスパターンではアドレスシフトを行わない場合が高速となった

アドレスシフトを求める時間を含めない場合

- strideアクセスで約23倍の高速化となり，Contiguous，Randomアクセスではアドレスシフトを行わない場合と大きな差は見られなかった