第3章 プロセッサごとに最適な並列化手法が異なる

並列化×並列化…OpenMPとSIMDの効果を検証

藤井 裕也

リスト1　並列化の効果を測定するためのメモリ転送中心の処理

```
void twice(float* from, float* to, size_t size) {
#pragma omp parallel for
    for (int i=0; i<size; i++) {
        to[i] = from[i] * 2;
    }
}
```

ラズベリー・パイで試してみる

前章までにOpenMPによるスレッド並列化による高速化と，NEONによる高速化を紹介しました．では，スレッド並列とNEONを両方使えば，両方の高速化の恩恵を受けられるのでしょうか？

この章では具体的な処理と，並列化による実行時間の変化を見ていきます．実験は次の条件で行いました．

1) ラズベリー・パイ2 Model B（前期型，Cortex-A7）
- gcc 8.3.0
- GCCオプション -O3 -mfpu=neon-vfpv4 -fopenmp

2) ラズベリー・パイ3 Model B（Cortex-A53）
- gcc 8.3.0
- GCCオプション -O3 -mfpu=neon-fp-armv8 -fopenmp

検証1：メモリ転送が中心の処理

リスト1のプログラムについて，スレッド数とコンパイル・オプションを変えて速度を計測します．-ffast-mathを指定するかどうかでGCCがNEON命令を使うかどうかを制御します．

sizeは4000000（特に意味はないが大きめの数字）を設定します．

測定結果を表1に示します．

表1　メモリ転送の多い処理での測定結果

モデル	スレッド数	NEON	時間［ms］
ラズベリー・パイ2B	1	なし	36.6
		あり	33.5
	4	なし	25.4
		あり	27.7
ラズベリー・パイ3B	1	なし	20.2
		あり	12.7
	4	なし	15.4
		あり	15.5

● ラズベリー・パイ2の場合

OpenMPだけ使うのが最も速いようです．Cortex-A7のNEONはあまり性能が良くないようで，特にNEON命令の演算結果が利用可能になるまでの遅延が長いと言われています．

今回のケースだと，

スレッド数が増えてどのみちメモリ・ネックになる状態なら，遅延の長いNEON命令は使わない方が良い

ということだと思います．ただし速度差はそこまでないため，ループ内の演算量やデータの依存関係によっては簡単に逆転しそうです．

● ラズベリー・パイ3の場合

NEONだけ使うのが最も速いようです．これぐらいの処理であればコンパイラが自動でNEONを使ってくれて，OpenMPのディレクティブを指定しなくてもよいため，最も手軽に書いたものが最速で動作します．

NEON命令を使わないと演算器は遊んでいるだけですが，CPUコアを空けておくと他のプログラムを並行して走らせるときにも有用なので，NEONだけを使う方が単純な速度差以上に実用上有利と思われます．