TSUBAME 共同利用　平成 26 年度　学術利用　成果報告書

利用課題名　高性能・高生産性を達成する垂直統合型アプリケーションフレームワーク
英文：High Performance, Highly Productive Application Frameworks

丸山直也
Naoya Maruyama

理化学研究所計算科学研究機構
RIKEN Advanced Institute for Computational Science
http://mt.aics.riken.jp/

邦文抄録（300 字程度）　本課題ではポストペタスケールに向けた最重要課題である「並列性の克服」、「信頼性」、「低消費電力化」の解決に大きく貢献する高生産性垂直統合型ソフトウェアスタックの研究開発を行う。これはスケーラブルマルチスレッドランタイムを基盤としたドメイン特化型アプリケーションフレームワークであり、自動並列化、自動チューニング、耐故障性、電力最適化等の各種技術を透過的に内包する。我々は提案ソフトウェアスタック構成法を流体シミュレーション（CFD）および分子動力学法（MD）を対象として設計し、それらを最新の大規模ヘテロジニアススーパーコンピュータである TSUBAME2 を基盤として設計開発する。

英文抄録（100 words 程度）　We develop a vertically-integrated highly productive software stack for achieving parallel, resilient, and power-aware application development. We design the software stack as a domain-specific application framework built on top of a scalable multithreading runtime. More specifically, we develop frameworks for computational fluid dynamics and molecular dynamics applications using large-scale heterogeneous supercomputers such as TSUBAME2.

Keywords: 5つ程度　GPGPU, Programming model, high-level framework

背景と目的

GPU などのベクトルアクセラレータを共用したヘテロジニアスシステムの重要性は消費電力最適化の重要性と共に今日のペタスケールシステムにおいて広く認識されており、今日の最新 GPU は数千の SIMD コアが搭載されたメニーコアプロセッサであるが、このような多数コアの超高密度実装はポストペタスケール、さらにはエクサスケールの実現に向けてさらなる性能向上に不可欠な技術である。実際に主要プロセッベンダは小規模スカラーコアと大規模ベクトルアクセラレータを統合したヘテロジニアスプロセッサの設計開発を行っていると言われており、2015 年頃のポストペタスケール、2018 年頃のエクサスケールにおいて主流となることが予測される。

上述のアーキテクチャのためのソフトウェアスタックはヘテロジニアスプロセッサのプログラミング、性能最適化、スケーラビリティ、耐故障性、電力効率最適化などの種々の課題を解決し、またそのプログラミングモデルが将来に渡って連続性のある高い生産性を有したものでなければならない。今日におけるヘテロジニアスシステム向けソフトウェアスタックではシステムの性能を最大限に引き出すためには複雑かつ煩雑なプログラミングが必要とされ、また将来のアーキテクチャの革新に対する連続性を有しない。さらにソフトウェアスタック全体としては上述のプログラミングモデルの問題に加えて、通信オーバーヘッドの隠蔽や動的負荷分散によるスケーラビリティの達成、耐故障性、電力効率の最適化を実現しなければならないが、そのようなソフトウェアスタックは我々の知る限り存在しない。

本プロジェクトでは、上述したポストペタスケールシステムにおける課題を同時に解決する、新たな垂直統合型ソフトウェアスタックを提案する。ターゲットとするアプリケーションドメインを限定し、そのドメインに対して最適化したアプリケーションフレームワークをシステムソフトウェアおよびアプリケーション研究の専門家の密な協力により設計開発することで、上記課題を高い生産性のもと解決する。

概要

上述の目的を達成するために本課題では今年度は昨年度に引き続き流体シミュレーション向けフレームワークである Physis の設計、開発を中心に進めた。Physis は C 言語を拡張しステンシル計算を簡便に記述可能とするドメイン特化型言語およびそのランタイムから構成される。Physis にて記述したステンシル計算は Physis トランスレータによって各種実行アーキテクチャ向けコードに変換され、プログラマが明示的に並列化や GPU プログラミングを行う必要はない。本課題では Physis の設計の拡張やトランスレータの最適化等を進めた。また Physis の実アプリケーションへの適用評価を進めた。

また、昨年度から引き続き今年度も GPU プログラムの自動カーネル融合最適化についても研究を進めた。

結果および考察

カーネル融合最適化についてはカーネルの分割も含めた最適化へと発展させた。またプロファイリングや静的コード解析により最適化全体を自動化することに成功した。一方で自動化には不完全な処理もあるため、プログラマによる制御も可能とする設計とした。これにより従来までは限られたアプリケーションへの適用だったが、6 本のアプリケーションへ適用実験を拡大した。その結果最大 1.7 倍の性能向上を達成できることを確認した。

Before Fission

```
Kern_A<<<G, B>>>(R, S, T, Q, P, V, U, W, nz, c);

__global__Kern_A(R, S, T, Q, P, V, U, W, nz, c){
int i = blockIdx.x*blockDim.x + threadIdx.x;
int j = blockIdx.y*blockDim.y + threadIdx.y;
for(int k=0; k<nz;k++)  {
  R[i,j,k]= S[i,j,k] * (V[i-1,j,k]/V[i,j,k]);
  W[i,j,k]= V[i-1,j,k] - V[i,j,k];
  P[i,j,k]= c*(Q[i-1,j,k]+Q[i,j-1,k]+Q[i,j,k]+Q[i+1,j,k]+Q[i,j+1,k]);
  U[i,j,k]= T[i,j,k]-Q[i,j,k]*(Q[i-1,j,k]-Q[i,j-1,k]);
 }
}
```

After Fission

```
Kern_X<<<G, B>>>(R,S,V,W,nz);
Kern_Y<<<G, B>>>(P,Q,U,T,nz,c);

__global__ Kern_X(R,S,V,W,nz){
int i= //absolute index in X-Dir
int j= //absolute index in y-Dir
for(int k=0; k<nz;k++) {
 R[i,j,k]= S[i,j,k]
        * (V[i-1,j,k]/V[i,j,k]);
 W[i,j,k]= V[i-1,j,k]-V[i,j,k]);
} }
```

```
__global__ Kern_Y(P,Q,U,T,nz,c){
int i= //absolute index in X-Dir
int j= //absolute index in y-Dir
for(int k=0; k<nz;k++) {
 P[i,j,k]= c*(Q[i-1,j,k]+Q[i,j-1,k]
   +Q[i,j,k]+Q[i+1,j,k]+Q[i,j+1,k]);
 U[i,j,k]= T[i,j,k] - Q[i,j,k]
        * (Q[i-1,j,k]- Q[i,j-1,k]);
 }
}
```

図 1 カーネル分割例

まとめ、今後の課題

本課題では高性能・高生産性を達成するソフトウェアスタックの実現に向けて、その一要素技術であるステンシルフレームワーク Physis の設計、実装を進めた。また、さらなる最適化として大規模プログラムに適用可能なカーネル融合・分割アルゴリズムを開発し、その評価を行った。

今後は開発したカーネル融合・分割アルゴリズムの改良、他のアクセラレータアーキテクチャへの対応を検討中である。