# XMP-dev に基づく CPU/GPU ハイブリッド負荷分散システム

小田嶋 哲哉[†] 李 珍泌[†]
朴 泰祐[†,††] 佐藤 三久[†,††]

## 1. はじめに

近年，高い演算性能及びメモリバンド幅をもつ GPU を画像処理以外の汎用計算に用いる GPGPU（General-Purpose computation on GPU）が注目されている．特に，NVIDIA 社が提供するプログラミング環境 CUDA（Compute Unified Device Architecture）によって CPU で行うプログラミングに近い形でのプログラミングが可能になったことで，HPC の様々なアプリケーション分野で GPGPU への対応が進んでいる．これに伴い，GPU を搭載した PC クラスタ（以下「GPU クラスタ」と略す）が数多く出現し，広く利用されるようになった．しかし，現在の PC クラスタは，すでに MPI や OpenMP など直交するモデルを組み合わせているため，複雑なプログラミングが必要である． GPU クラスタでは CUDA などによる GPU プログラミングが加わることで，プログラミングコストの増加が問題になっている．

また GPU クラスタでは，GPU を一種の非常に高速な計算加速装置とみなして，CPU から計算するデータを送り，計算が終わったらデータを受け取るという機能分散的なプログラミング手法が一般的である．しかし，これでは CPU の計算リソースを GPU と並行して有効に使用することができない．また，CPU のコア数は年々増加しており，GPU と CPU の潜在的な演算能力は徐々に近づきつつあると言える．

我々は従来より，分散メモリシステムを対象とした，PGAS 並列プログラミング言語 XcalabmeMP（以下「XMP」と略す）の開発を進めている．この GPU 向けの拡張仕様として XcalableMP acceleration device extension[1]（以下「XMP-dev」と略す）があるが，これは指示された特定のループ処理をすべて GPU だけで計算するもので，ハイブリッド処理は対象としていない．そこで，XMP-dev の枠組みで CPU と GPU を負荷分散の対象として同時に利用するハイブリッド処理を検討する．我々は，INRIA で開発されている StarPU[2] システムに着目し，XMP-dev との統合化を行う．ランタイムレベルで CPU と GPU へのタスクのスケジューリングをすることが可能であり，これを XMP-dev に組み込むことで言語レベルでの CPU/GPU 協調計算を行うことが可能になる．これによって，GPU と CPU へのデータの分散及び負荷分散をし，計算リソースを最大限活用できるプログラミングの支援を行う．

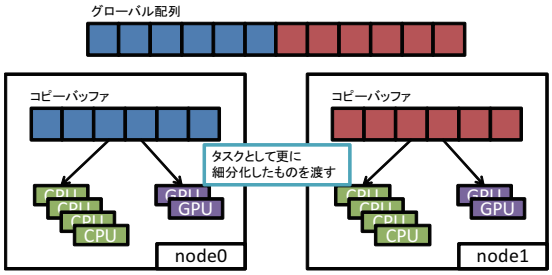


**図 1** タスク割り当てのイメージ

## 2. XMP-dev/StarPU の提案

我々は，StarPU を用いて CPU と GPU での協調計算を言語レベルで行うために，XMP-dev の拡張仕様である XMP-dev/StarPU を提案している．ベースである XMP-dev に関しては文献[1] に詳しいが，ここでは簡単な説明をする．XMP-dev は逐次のプログラムに directive を挿入することで，その部分について分割・GPU による計算を行える．これによって，プログラミングのコストをおさえることができる．本実装では，XMP-dev が device として GPU を対象としていたように，device として StarPU のタスクスケジューラを呼び出すように．これによって，ノード間のデータの分散や，同期処理は従来通りの動作をし，ノード

† 筑波大学大学院 システム情報工学研究科
Graduate school of Systems and Information Engineering, University of Tsukuba
†† 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba



ⓒ 2012 Information Processing Society of Japan

**表 1** ノード構成

| | |
|---|---|
| 構成 | 4 ノード（16CPU コア/ノード） |
| CPU | AMD Opteron 6134 2.3GHz |
| メモリ容量 | 16GB DDR3 |
| GPU | NVIDIA Tesla C2050 |

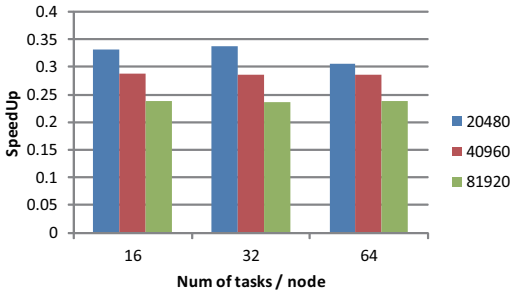


**図 2** XMP-dev に対する相対評価

内における計算リソースへのタスクのスケジューリングは StarPU が行うように制御する．

StarPU では，配列データを細かく分割し，計算に必要なものをまとめた単位をタスクの演算対象と定義する．これは，GPU が CPU と独立なメモリ空間を持っており，各タスクのデータに独立性が必要なためである．**図 1** に示すように，XMP-dev をこれに対応させるためにノード間で分割されたローカル配列に対し，コピーバッファを作る．このバッファをノード内でさらに細かく分割することでタスクとして渡せるようにする．この配列に関するメモリ操作や計算リソースへのタスクの割り当ては StarPU の API を用い，ノード間のデータの交換などはコピーバッファから XMP で分割したローカル配列にコピーバックをし，ローカル配列に対して従来の XMP のノード間の分散，同期を行う．我々の提案では，コピーバッファに対する分割数，つまりタスクの数は指示文のオプションとしてユーザが指定する．

## 3. 性能評価

XMP-dev/StarPU コンパイラはまだ開発中であるため，本評価には XMP-dev/StarPU が生成するコードを想定したものを用いる．評価に用いるベンチマークは N 体問題とし，粒子数である問題サイズ $N$ を 20480，40960，81920 とする．それぞれに対し，タスク数を変えた時，XMP-dev の実行時間に対して想定したコンパイラの相対性能を**図 2** に示す．評価に用いたクラスタは**表 1** の通りである．

図 2 は，縦軸に計算時間，横軸にノード内におけるタスクの数を示している．各問題サイズ，タスク数において XMP-dev に対して約 3 割程度の性能し

**表 2** 問題サイズ 40960 における演算性能 [GFLOPS]

| タスク数 | 16 | 32 | 64 |
|---|---|---|---|
| GPU Perfomance | 4.06 | 2.05 | 1.07 |
| CPU Perfomance | 2.59 | 3.43 | 4.09 |

か出ていないことがわかる．これは，StarPU のタスクスケジューリングルールが関係していると考えられる．StarPU のスケジューリングは，空いている計算リソース（CPU または GPU）にタスクを順次割り当てる．**表 2** より，粒度が大きいタスク数のときには CPU が計算のボトルネックとなってしまい，逆に細粒度のタスクではタスク実行時に通信が頻発してしまうことによるボトルネックが影響して性能が低下している．タスクのサイズが一定である本実装ではどちらかのリソースがボトルネックとなってしまい，性能が向上していないと考えられる．このことから，XMP-dev/StarPU では StarPU の各タスクサイズを調整する必要があり，CPU と GPU の演算性能に応じた最適なタスクサイズを選択することが性能向上の鍵であると推察する．

## 4. おわりに

本稿では XMP-dev を拡張した XMP-dev/StarPU を提案し，想定されるコードにおいて GPU と CPU の協調計算を行った．これによって，タスクのサイズによって性能が出にくいという問題がわかった．

これを解決するために，各リソースに応じてタスクサイズを変えることが必要であると考えられる．GPU を用いる場合，タスクの起動時に毎回通信が発生する．また，一度に大量のデータを処理させることによって本来の GPU の性能を引き出すことができる．一方 CPU において，タスクの粒度を細かく，多くすることで StarPU のスケジューリングが柔軟になり常に CPU コアが動作しているようにする必要がある．

**謝辞** 本研究の一部は，戦略的国際科学技術協力推進事業（日仏共同研究）「ポストペタスケールコンピューティングのためのフレームワークとプログラミング」による．

## 参考文献

1) 李珍泌, チャントゥアンミン, 小田嶋哲哉, 朴泰祐, 佐藤三久．PGAS 並列プログラミング言語 XcalableMP における演算加速装置を持つクラスタ向け拡張仕様の提案と試作．情報処理学会論文誌, Vol. 5, No. 2, pp. 33–50, Mar. 2012.
2) StarPU Handbook for StarPU 0.9.2 . `http://runtime.bordeaux.inria.fr/StarPU/`.


ⓒ 2012 Information Processing Society of Japan