GPU 向け協調マルチタスキングにおけるオーバヘッド削減の検討

岡　陽介[†]　伊野文彦[†]　萩原兼一[†]

1. はじめに

近年，GPU の持つ高い浮動小数点演算性能に着目し，ネットワーク上の計算資源として共有する試みがある．ボランティア型計算システムの計算資源として利用する環境下では，資源の提供側で動作するホストアプリケーション（HA）と資源に投入されるゲストアプリケーション（GA）が同時に実行されうる．現在の GPU はプリエンプティブマルチタスキングに対応しておらず，GPU 上で呼び出した複数のプログラムが逐次実行される．そのため，HA と GA を同時に実行すると，GA の終了まで HA の実行が待たされ，HA のフレームレートが低下する問題がある．

この問題に対して，既存研究[1] は GA を複数のタスクに分割して，GPU の負荷が高い場合もしくは低い場合に応じた 2 種類の実行モードを使い分けることにより，HA のフレームレートを維持しつつ GA を実行している．ここで，カーネルとは GPU 上で動作するプログラムのことである．負荷推定に要するオーバヘッドを低くするために，GPU サイクルを消費しない空カーネルを用いる．空カーネルを GPU 上に呼び出してから終了するまでの時間（実行終了時間）を閾値と比較し，負荷を推定する．問題として，大量に空カーネルを実行するような状況において，オーバヘッドの総和が増大する．

本研究は，オーバヘッドの削減を目的として，空カーネルの替わりに GA カーネルを用いる．GA カーネルの実行終了時間は GPU の負荷が高いと長くなる．この時間の長短により GPU の負荷を推定し，実行モードを選択する．提案手法では HA と GA のみを GPU 上に呼び出すため，GA を GPU に割り当てる機会が増加する．

2. 提案手法

負荷を推定するために，GA カーネルの実行終了時間 t を測定する．また，HA が動作していない状態で GA カーネルの実行終了時間を T とする．T の最大値を実行モード選択のための閾値 T_s とする．HA が動作している場合，HA の描画処理が終了するまで，GA カーネルを GPU に割り当てることができないため，GA カーネルの実行終了時間は長くなる．すなわち，$t > T_s$ となる可能性が高い．したがって，$t > T_s$ となった場合に GPU は高負荷であると推定する．さらに，低負荷であると誤って推定することを回避するため，r 回以上 $t < T_s$ となった場合に GPU は低負荷であると推定する．r は実験により求めた値を使用する．

提案手法が用いる実行モードは既存手法と同一である．GPU の負荷が低い場合は GA のスループットを向上させるために，GA カーネルを連続して実行する．(**図 1**) 一方，GPU の負荷が高い場合は GA カーネルを一定間隔の待ち時間を挟み実行し，HA に描画処理の機会を提供する．(**図 2**)

次に，GA を実行する処理の流れを説明する (**図 3**)．GA の実行を開始する時点では GPU の負荷は不明である，したがって，HA のフレームレート低下を抑制するため，高負荷向けの実行モードで GA カーネルを実行する．2 回目の GA カーネル実行以降は，前回の GA カーネル実行より得られた t を用いて負荷を推定

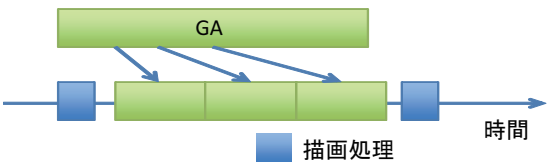

図 1　低負荷向け実行モード

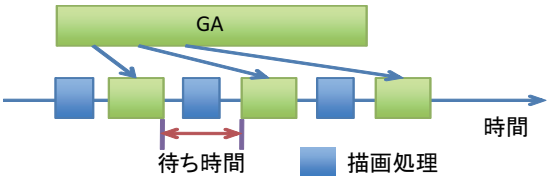

図 2　高負荷向け実行モード

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

```
入力： GA タスク，閾値 T_s，試行回数 r
 1: count := 0; t := 0; T_s := 0;
 2: while 全てのタスクが終了していない do
 3:    if t < T_s then
 4:       count := count + 1;
 5:    else
 6:       count := 0;
 7:    end
 8:    T_0 := 現在時刻;
 9:    if count < r then // 高負荷と推定
10:       高負荷向け実行モードで GA カーネル実行;
11:       待ち時間を挿入;
12:    else // 低負荷と推定
13:       低負荷向け実行モードで GA カーネルを実行;
14:    end
15:    t := 現在時刻 −T_0;
16: end
```

図 3 GA カーネルの実行アルゴリズム

し，実行モードを決定する．この処理を GA タスクが全て終了するまで繰り返す．

3. 評 価 実 験

負荷推定時のオーバヘッドを削減できていることを確認するために，HA としてフレームレートが毎秒 60 回である，描画処理のベンチマークを実行している状態で，提案手法および，既存手法を適用した GA を実行した．さらに，提案手法による弊害がないことを確認するために 4 種類の HA 実行状態において GA を実行し，HA のフレームレートと GA の性能を既存手法を比較した．HA 実行状態とは，GPU が遊休状態，および HA として Word で文書を作成，YouTube で動画を視聴，OpenGL レンダラを実行のいずれかである．いずれの実験も GA として CUDA SDK 3.2 に含まれる行列積を用いた．行列のサイズは 3072×3072 であり，GA の実行時間とは行列積を 300 回繰り返し実行した時間と，高負荷向け実行モードにおける待ち時間の合計時間である．また負荷推定の試行回数は $r = 4$ とした．実験環境は Intel Xeon 2.80 GHz と NVIDIA GeForce GTX 470 である．

図 4 にベンチマークと同時に実行した GA の実行時間を示す．提案手法は既存手法と比べ，約 12% 実行時間を短縮できている．既存手法の GA 実行時間において空カーネル実行に要した時間も約 12%であるため，空カーネル実行におけるオーバヘッドを削減できている．HA のフレームレートは提案手法および既存手法において毎秒 60 回であり，フレームレートを維持できた．

表 1 および**表 2** に各 HA と同時に GA を実行した場合の，HA のフレームレート f および GA のスループット比 P を示す．行列積を専有実行した実行時間を t_l，各 HA 実行状態における GA 実行時間を t_m とすると，$P = t_l/t_m$ である．OpenGL を除いて，提案手法は f を維持しつつ，既存手法と同等の P を得ることができ，提案手法における弊害が無いことが確認できた．しかし，OpenGL においては f が毎秒 5 回低下し，P が約 10% 向上している．この原因は既存手法に比べて提案手法では，低負荷向け実行モードが多く選択されたためである．

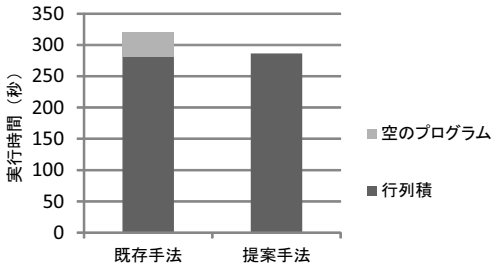

図 4 ベンチマークソフト実行時の GA 実行時間

表 1 各 HA のフレームレート（回／秒）

HA 実行状態	単独実行	提案手法	既存手法
Word	12.0	12.0	11.9
YouTube	32.3	32.2	31.8
OpenGL	60	56.5	59.1

表 2 各 HA 実行状態における GA のスループット比（%）

HA 実行状態	提案手法	既存手法
遊休状態	98.0	98.1
Word	97.7	97.2
YouTube	95.8	94.8
OpenGL	42.1	33.8

今後の課題として，動的にフレームレートが変動する HA に対する GA の性能向上が挙げられる．

謝　　辞

本研究の一部は，科学研究費補助金基盤研究（B）（2330007）の補助による．

参 考 文 献

1) Fumihiko Ino and *et al.* Cooperative multitasking for GPU-accelerated grid systems. In *Proc. CCGrid'10*, pp. 774–779, May 2010.