CUDA 互換 GPU における高速なストリーム処理のための タスクスケジューリングアルゴリズムの検討

中野 瑛仁 [1]　伊野 文彦 [2]　萩原 兼一 [2]

[1] 大阪大学基礎工学部情報科学科　[2] 大阪大学大学院情報科学研究科

1　はじめに

GPU（Graphics Processing Unit）向けの開発環境である CUDA（Compute Unified Device Architecture）[1] は，ストリーム処理 [2] を記述するための機能を提供する．ストリーム中の i 番目の要素に含まれる命令は，ダウンロード D_i，カーネル K_i，リードバック R_i があり [1]，これらをタスク t_i で処理する．異なるタスク間においては，K を D および R とオーバラップできる．また，タスク間には実行順の依存関係が存在せず，任意の順で処理できる．ただし，同一タスク内における命令の実行順は変更できない．したがって，命令の呼び出し順を工夫することで資源の遊休状態を減らし，実行時間を最大限に削減できる．ただし，命令の呼び出し順を決定する作業は開発者の負担を増大させる．

この負担を軽減するために，中川ら [3] は動的なタスクスケジューリングを実現するミドルウェアを提案している．このミドルウェアは一定の基準に基づいてタスクを選択し，タスク内の命令を実行する．この基準により，原理的に隠蔽可能な命令集合 Con_t において，隠蔽される実行時間は増加する．したがって，Con_t の実行時間は縮小される．しかし，原理的に隠蔽不可能な命令集合 Con_f の実行時間は増加する．

そこで，本研究では中川らのミドルウェアにおけるタスクの新たな選択基準を提案する．提案手法は Con_f を最小化する．また，Con_t において，多くの場合に既存手法と同様に実行時間を短縮できる．

2　命令のオーバラップにおける問題

D_i，K_i および R_i に要する時間をそれぞれ T_{D_i}，T_{K_i} および T_{R_i} とおく．中川らは前提条件を $T_{K_i} \gg T_{D_i} + T_{R_i}$ とおき，実行時間の下限を $T_{opt} = \min(T_{D_1}..T_{D_n}) + \sum_{i=1}^{n} T_{K_i} + \min(T_{R_n}..T_{R_n})$ としている．なお，第 2 項は Con_t，第 1，3 項は Con_f の下限である．また，条件としてタスク間で T_D，T_K，T_R の比をタスク粒度の大小関係で与えている．さらに，タスク t_i の粒度 $|\ t_i\ |$ は D_i および R_i のデータ通信量の和で与える．

T を最小化するためには，まず $D_2..D_{n-1}$ および $R_2..R_{n-1}$ を $K_1..K_n$ でオーバラップする必要がある．さらに，粒度最小のタスクを実行順の最初および最後に処理する必要がある．

中川らは，T を最小化するために，ミドルウェアへ入力されるタスク列 $t_1..t_n$ を $|\ t_i\ |$ で昇順（降順）にソートし，実行順のタスク列 $t'_1..t'_n$ としている．このソートにより $\Sigma_{i=1}^{n-1}(||\ t'_i\ | - |\ t'_{i+1}\ ||)$ が最小となる．そのため，K_i を用いて R_{i-1} および D_{i+1} を隠蔽しきれる可能性が増加し，Con_t のオーバヘッドは減少する．しかし，タスクを昇順（降順）に選択する場合，T_{R_n}（T_{D_1}）の値が大きくなるので，Con_f のオーバヘッドは増加する．

3　提案手法

実行時間を短縮するためには以下の評価関数を最小化すればよい．

1. $\Sigma_{i=1}^{n-1}(||\ t'_i\ | - |\ t'_{i+1}\ ||)$
2. $T_{D_1} + T_{R_n}$

関数 1. は Con_t におけるオーバヘッドを削減するための条件である．また，関数 2. は Con_f におけるオーバヘッドを削減するための条件である．この 2 つの関数を同時に最小化することは不可能である．そのため，片方の関数は最小値に近似させることとなる．

中川ら [3] の手法では，関数 1. に重点がおかれている．それに対し本手法では，関数 2. を最小とするソート法を提案する．また，本手法は関数 2. と同時に関数 1. も可能な限り短縮することを目指す．

提案手法のアルゴリズムを以下に述べる．まず昇順にソートされたタスク $t'_1..t'_n$ を奇数番目と偶数番目に分割する．次に奇数番目のタスクを昇順に実行し，その後偶数番目のタスクを降順に実行する．

提案手法では，最初に t'_1，最後に t'_2 を処理するため，関数 2. を最小化する．また，この条件のもとで提案手法は関数 1. を最小限の値にする．

4　実験

提案手法の有用性を検証するために，昇順ソートおよび降順ソートとの比較を示す．実験では，CPU として Intel Xeon X5670，GPU として NVIDIA GeForce GTX 580 を持つ計算機を用いた．

入力として実行時間の異なる 8 つのタスクを与えた．全体の実行時間における T_k の割合 $r = \sum_{i=1}^{n} T_{K_i} / \sum_{i=1}^{n} (T_{D_i} + T_{K_i} + T_{R_i})$ を変えながら実験をした．

表 1: 各ソート法における実行時間（ミリ秒）

r	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
提案	41	41	41	41	43	65	97	173	379
降順	41	41	41	41	45	68	100	175	385
昇順	41	41	41	41	46	69	100	176	384

表 4 に試行 60 回の平均を示す．$r \geq 0.5$ のとき，提案手法は実行時間を 2〜6 ミリ秒ほど削減できた．したがって，提案手法は $r \geq 0.5$ のとき既存手法と比べ実行時間を短縮できている．

一方，$r \leq 0.4$ では，実行時間の差はなかった．この原因は前提条件 $T_{K_i} \gg T_{D_i} + T_{R_i}$ が満たされず，$T_{opt} = \sum_{i=1}^{n} (D_i + R_i)$ となったためである．この式のもとでは，条件 2. による実行時間の短縮は見込めない．

今後の課題としては，命令発行の優先度に関するアルゴリズムを検討し，速度向上を図ることが挙げられる．

参考文献

[1] NVIDIA Corporation. CUDA Programming Guide Version 4.0, June 2011.

[2] Brucek Khailany and *et al*. Imagine: Media processing with streams. *IEEE Micro*, 21(2):35–46

[3] Shinta Nakagawa and *et al*. A middleware for efficient stream processing in CUDA. *Computer Science - Research and Development*, 25(1/2):41–49