4M-3

GPGPU を用いた低消費電力タスク・スケジューリングの組合わせ最適化解法

吉田 晴紀＊，畠山 弘樹 , 朝倉 宏一（大同大学）

1. はじめに

我々は、サーバ計算機の消費電力削減を目的とした、低消費電力スケジューリング・アルゴリズムを開発している[1,2]。プログラム全体の処理時間を増加させずに消費電力を削減させるため、通信処理などに伴うタスク間の処理待ち時間に着目し、タスクを消費電力の低い P-State で実行させる。このとき、どのタスクにどの P-State を割り当てるかを組合せ最適化問題として定義し、解を抽出する。しかし組合せが膨大で処理に時間がかかるという問題点がある。本稿では、タスクと P-State の組合せ問題を GPGPU を用いて計算することで、処理時間の短縮を図る。

2. 低消費電力タスク・スケジューリング

低消費電力タスク・スケジューリングは複数のタスクで構成されるプログラムを、複数のプロセッサを持つ計算機で実行させるとき、どのタスクをどのプロセッサに割り当て、どの P-State で実行させるかを解決する問題である。P-State とはプロセッサの動作周波数と動作電圧の組のことである。近年のプロセッサには複数組の P-State が用意されており、高速だが消費電力が多い状態から低速だが消費電力が少ない状態まで存在する。本稿では P-State の状態を数値で表し、P-State の値が大きくなるほど処理時間が遅く、消費電力は少なくなるとする。プロセッサが P-State を切り替えるための仕組みが DVFS (Dynamic Voltage and Frequency Scaling)である[3]。低消費電力タスク・スケジューリングでは処理待ちが発生するタスクの P-State を変化させて処理時間を増加させずにプロセッサの無駄な消費電力の削減を行う。Fig.1 に DVFS を適用し P-State を変更した例を示す。Fig.1(a)のタスク n2 の処理コストは 30 である。n2 から n4 へ処理が移る際に n3 から通信処理待ち時間が 30 発生している。Fig.1 (b)は n2 に DVFS を適用した結果である。プロセッサの処理速度を 1/2 に落として n2 の処理コストが 60 に増加したがプログラム全体の処理時間には影響を及ぼさない。Fig.1(b)では Fig.1(a)に比べて n2 の処理にかかる消費電力を 75%削減することができる。 N個のタスクと P-State を P種類持つプロセッサにスケジューリングする場合を考えると解空間が P^Nとなるため P-State の数とタスク数が多い場合は現実的な時間で最適解を求めることができない。

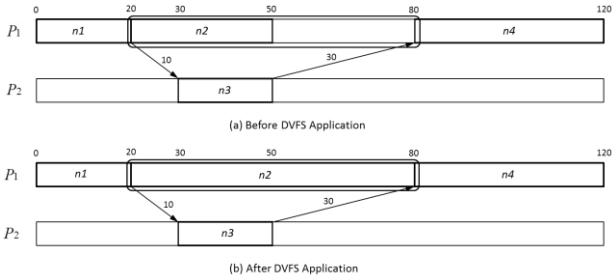

Fig.1 An application of DVFS

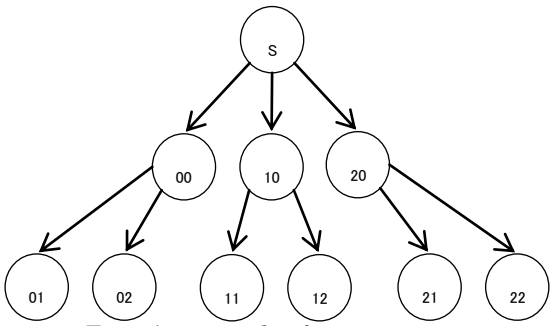

Fig.2 An example of a tree structure

3. アプローチ

最適な P-State の組の計算には組合せ最適化アルゴリズムである分枝限定法[4]を適用して厳密解の計算を効率的に行う。まず、タスク全体の処理時間と消費電力の計算をノードとして、タスクの P-State の全ての組で木を作成する。Fig.2 にタスク数が 2、P-State が 3 段階の場合の木の例を示す。Fig.2 のノードの中の数字はタスクの P-State の組である。たとえば、「10」はタスク 1 の P-State を 1 に、タスク 2 の P-State を 0 に、それぞれ設定することを表している。P-State は 0 が一番高速で消費電力が高い状態を表している。次に木の根から順に、ノードを処理していき最小消費電力量とその P-State の組を保存する。また、あるノードでプログラム全体の処理時間が増加するとき、処理時間が増加したノードより下のノードでも同様に処理時間が増加するため、そのノードの下にあるノード群の処理を行わない。これにより不必要な計算を省略することができる。

4. CUDA を用いた実装

NVIDIA 社が提供する CUDA (Compute Unified Device Architecture) は GPGPU のための統合開発環境である[3]。CUDA のプログラミングモデルでは最小の実行単位は thread として表現される。そして、thread を集約した処理単位として block が定義される。block 内の thread 数は Fermi コアであれば 1024 である。さらに block を集約した処理単位として grid が定義される。このようなモデルにより、CUDA では非常に多くのスレッドを階層的に管理し、並列実行することができる。

CUDA プログラムで Fig.2 の木の根から幅優先探索でノードを thread に割り当て、並列処理により最適解を求める。Fig.3 に Fig.2 の木をスレッドに割り当てた例を示す。Fig.3 では最初に「00」、「10」、「20」の P-State の組の結果を並列に計算を行っている。計算終了後、プログラム全体の処理時間が増加しなかった場合、そのときの消費電力量を block 内のすべての thread から参照可能な共有メモリに格納する。各スレッド 2 回目以降は、共有メモリに格納された親ノードの結果を元にして計算し、より消費電力量が少ない結果が得られたら共有メモリを更新する。これを木の高さだけ繰り返すことにより、共有メモリに最適解の候補が得られ、最後に集約演算により最適解を導出する。

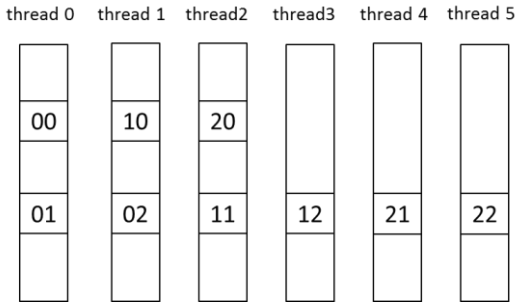

Fig.3 an example of program structure

5. まとめ

本稿では、GPGPU を用いた低消費電力タスク・スケジューリングの組合わせ最適化解法の開発について述べた。処理時間が木の高さに依存する本手法を用いた場合、分枝限定法による計算量の削減が効果を発揮しにくい。そのため、より GPGPU に適した計算アルゴリズムを開発することが、今後の課題として挙げられる。

文 献

[1] 畠山弘樹, 朝倉宏一：「DVS 適用による消費電力削減のための効率的なタスク選択機構の提案」, 平成22年度電気関系学会東海支 部連合大会, D5-6 (2010).

[2] 森祐一郎, 朝倉宏一, 渡邊豊秀：”マルチ・プロセッサ環境におけるDVSを用いた消費電力削減アルゴリズムの構築 ”, 電気学会論文誌C, Vol.131, No.4, pp.926-933 (2011).

[3] Hisa Ando：“プロセッサを支える技術”, 技術評論社 (2011).

[4] 今野治, 鈴木久敏：“整数計画法と組合せ最適化”, 日科技連出版社 (1982)

[5] David B. Kirk, Wen-meo W. Hwu：CUDAプログラミング実践講座, ボーンデジタル (2010).