B-014

CPU 及び GPU 混載環境における 自動チューニングとセルフスケジューリングの比較

Evaluation of hybrid parallel optimization implementations on CPU with integrated GPU

山下 将嗣[1)]　　若谷　彰良[2)]
Masatsugu Yamashita　Akiyoshi Wakatani

1　はじめに

半導体技術の進歩により CPU のアクセラレータとして GPU が用いられるようになっているが、GPU を用いる際には異なるアーキテクチャーを活かした処理が必要となる。しかし、CPU と GPU が混載されているプロセッサにおけるハイブリッド並列処理において、それらの負荷分散の最適な決定方法が課題として挙げられる。

本研究では、組合せ最適化問題として代表的なアプリケーションの 1 つである巡回セールスマン問題を対象とし、タスク割り当ての手法の 1 つであるセルフスケジューリングに着目し、OpenCL 環境にてハイブリッド並列処理を行うプログラムを実装し、その性能評価を行う。

本研究で対象とする GPU 内蔵プロセッサとは、CPU 内部に GPU の機能を搭載したもので「iGPU(Integrated GPU)」とも呼ばれる。GPU には、「内蔵 GPU」と「分離 GPU」の 2 種類に分類され、「分離 GPU」はグラフィックボードのことを指す。内蔵 GPU は、分離 GPU に比べて消費電力が少なく、コストパフォーマンスが高いという利点がある。

また巡回セールスマン問題 (travelling salesman problem, TSP) とは、都市の集合と各 2 都市間の移動コストが与えられたとき、全ての都市をちょうど一度ずつ巡り出発地に戻る巡回路のうちで総移動コストが最小のものを求める組合せ最適化問題である。問題の大きさは、都市の数で表される。この問題は、計算複雑性理論において NP 困難と呼ばれる問題のクラスに属する。

なお本研究では、総当たり法を用いて最短経路を求める。

2　関連研究

川崎らの研究は [1]、GPU の実行フレームワークとして OpenACC に着目し、OpenACC では GPU 実行中に CPU の待機状態による無駄を解消する仕組みがないことから、OpenMP/OpenACC を用いたハイブリッド並列化のためのコード変換フレームワークの提案を行っている。ここで提案されるコード変換フレームワークでは、ハイブリッド並列化としてチャンクサイズ一定のセルフスケジューリング法 (動的負荷分散方式) を扱っているが、最適なチャンクサイズを求める方法は未知である。

また Chronopoulos らは [2]、ヘテロジニアスシステムにおける負荷分散について、各種セルフスケジューリングを検討し、trapezoidal self-sheduling の新しい方法を提案している。一方、我々のシステムでは、CPU と GPU という最小のヘテロジニアスシステムであり、その構成に適したセルフスケジューリングを提案する。

1)　甲南大学大学院　自然科学研究科
2)　甲南大学　知能情報学部

3　Static self-scheduling の評価

3.1　実験環境

今回の実験で、CPU に Intel Core i5 - 10210U，GPU に Intel@ UHD Graphics for 10th Gen Intel@ Procrssors、OS に Windows 10 Home を用いたコンピュータ上で、Microsoft Visual Studio 2022 環境内で OpenCL 3.0 を用いて実行し、性能評価する。

3.2　実験

static self-scheduling とは、１つのタスクを割り当てる際のチャンクサイズを一定にして行うセルフスケジューリングのことである。まず TSP を対象に総当たり法で最短経路を求めるプログラムにおいて static self-scheduling におけるチャンクサイズによる影響を調べる。図 1 に、都市数 13 の場合の実行時間の比較を示す。

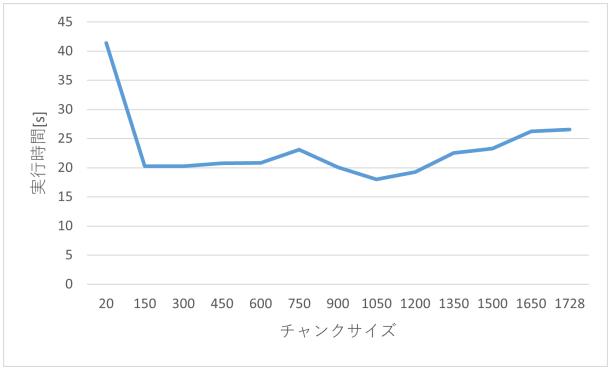


図 1: チャンクサイズを変更した場合の結果

図 1 で示すように、チャンクサイズによって実行時間への影響があることが分かった。この図からわかることとして、チャンクサイズが特に小さい場合に実行時間が長くなってるのは CPU/GPU に対する割り当て回数が多くなりタスク割り当てを行う際に起こるオーバーヘッドが大きくなったためだと考えられる。

したがって、最速となるスケジューリング単位を見つけることは重要ではあるが非常に困難であり、static self-scheduling の最大の欠点であるといえる。

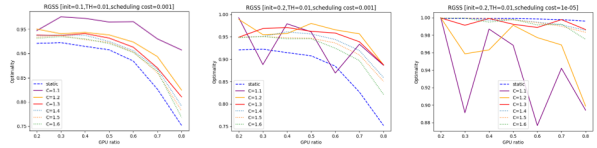
次に、研究 [3] の予備実行を行う自動チューニングプログラムとの性能比較を行った。予備実行を行う自動チューニングプログラムとは、ループの内の数パーセントを各スレッド (CPU/GPU) で予備実行を行い、その結果から CPU 及び GPU に対する負荷を事前決定し実行するプログラムである。都市数 13 の場合の結果は、セルフスケジューリングが 17.9 秒、予備実行を行う自動チューニングが 22.1 秒となった。また、自動チューニングにおいて予備実行の時間を抜いた時間が 15.7 秒であった。このことから最適なチャンクサイズが分かれ

ば、予備実行による自動チューニングよりもセルフスケジューリングの方が実行時間は速いという結果になった。しかし、図 1 からも分かるように最適なチャンクサイズを見つけることは容易ではない。

4 Relaxed guided self-scheduling の提案

前章で述べたように、static self-scheduling においては、チャンクサイズを適切に選ぶことにより、性能において最大 130 %の差が発生し、その選択は重要であるが、事前にその値を決定することは難しい。そこで、CPU と GPU という並列度 2 に適したセルフスケジューリングとして、guided self-scheduling[2] や trapezoidal self-scheduling[2] のように、大きめのチャンクサイズから徐々にサイズを小さくする手法を検討する。ここでは、チャンクサイズの初期値を INIT とし、パラメータ C を用いてチャンクサイズを xx=xx/C で徐々に小さくし、TH まで小さくなればサイズを変更しないスケジューリング手法 (RGSS: relaxed guided self-scheduling) を提案する。 この手法においては、最適な C や INIT のパラメータを決定することが必要となるので、シミュレーションによってその値を決定する。シミュレーションにおいては、GPU と CPU の最適な負荷分散をしたときの GPU の割合 (GPU ratio) に対して optimality(最適度) がどのようになるかを評価する。例えば、GPU と CPU のスピード比が 3:1 であれば、GPU に全体の 3/4、CPU に全体の 1/4 を担当させれば最適となるので、その時の GPU の負荷の割合を GPU ratio と呼ぶ。なお、この割合で静的に負荷分散させたときの実行時間を best time とする。また、optimality とは、RGSS で実行した時の実行時間と best time との比を表しており、例えば、best time が 0.25 で、RGSS での実行時間が 0.29 であれば、optimality は、0.86(= 0.25/0.29) と計算する。1.0 に近い方が望ましいが、1.0 を超えることはない。

シミュレーションでは、問題サイズは 1.0 で固定とし、スケジューリングすることにかかるコスト (s-cost) を定義することで、問題サイズの大きさをコントロールし、そのコストが 0.001 の場合を中程度の問題サイズ、0.00001 の場合を問題サイズが大きい場合としている。さらに、INIT を、0.1、0.2、0.3 とし、C の値を 1.1〜1.6 で変化させ、TH は 0.01 とする。比較対象の static self-scheduling のチャンクサイズは 0.01 とする。シミュレーション結果の代表的なものを図 2 に示し、図 2(a) と図 2(b) は中程度の問題サイズ、図 2(c) は問題サイズが大きい場合の結果である。図 2(a), 図 2(b) から分かるように、中程度の問題サイズでは、チャンクサイズの初期値を 0.1 とし、C を 1.1 とする場合が、GPU ratio に関わらず、概ね、optimality が 0.95 を超えており、最適な組み合わせであることが分かる。また図 2(c) は問題サイズが大きい場合であるが、RGSS よりも、static self-scheduling で十分な optimality を得ることができている。問題サイズが大きい場合は、スケジューリングに要するコストが相対的に小さく、比較的小さい 0.01 のチャンクサイズでスケジューリングしても、そのコストは無視できるので、static self-scheduling が最適に近づきやすいと考えられる。しかし、問題サイズが大きいかどうかは相対的なもので、明確に区別しにくく、問題サイズによって手法を切り替えるのは難しい。その点で、問題サイズに関わらず、安定して高い optimality を示しているのは、INIT=0.2 のチャンクサイズを初期値とした C=1.3 の場合と考えられ、optimality は 0.95 前後を示している。今後の検討では、C=1.3、 INIT=2.0 の RGSS をベースに、各種アプリケーションおよびシステムの性能評価を行っていく。



(a) INIT=0.1, s-cost=0.001　(b) INIT=0.2, s-cost=0.001　(c) INIT=0.2, s-cost=0.00001

図 2: RGSS の見積もり

5 おわりに

本研究では、まず巡回セールスマン問題を対象とし、CPU および GPU 混載環境における static self-scheduling の評価を行った。性能評価では、チャンクサイズによる実行時間への影響が非常に大きくチャンクサイズによっては最大 130 %の差が出た。また、予備実行を行う自動チューニングプログラムとの性能比較では、今回の研究における最適値であるチャンクサイズのときは、実行時間で優位になったが、適切でないチャンクサイズでは遅くなってしまうこともあるため、最適なチャンクサイズを見つけることが重要であることが分かった。

しかし、最適なチャンクサイズを見つけることは容易ではないため、大きめのチャンクサイズから徐々にサイズを小さくし、一定まで小さくするとそれ以降はチャンクサイズを固定とするスケジューリング方法である RGSS を提案した。

今後は、RGSS においてシミュレーションによって得られた C=1.3、 INIT=2.0 をベースに各種アプリケーションおよびシステムの性能評価を行っていく。

謝辞

本研究の一部は甲南大学デジタルツイン研究所経費, JSPS 科学研究費 (基盤研究 (C) 23K02671) 及び私立大学等経常費補助金別補助「大学間連携等による共同研究」による.

参考文献

[1] 川崎 真之, 大島 聡史, 八巻 隼人, 三輪 忍, 本多 弘樹, “OpenMP/OpenACC ハイブリッド並列化のためのコード変換フレームワークの提案”, 情報処理学会研究報告, Vol. 2022-HPC-187, 7 pages, 2021.

[2] AT Chronopoulos, R Andonie, M Benche, “D Grosu A class of loop self-scheduling for heterogeneous clusters, Proceedings 2001 IEEE International Conference on Cluster Computing”, DOI: 10.1109/CLUSTR.2001.959989, 2001.

[3] 若谷彰良, 中谷秋栄, “再帰関数を含むアプリケーションに対する OpenCL のハイブリッド並列”, 甲南大学紀要 知能情報学編,vol. 16, no. 1, pp. 1-6, 2023.