氏名（本籍地）	佐藤（さとう） 功人（かつと）
学位の種類	博　士（情報科学）
学位記番号	情　博　第523号
学位授与年月日	平成24年3月27日
学位授与の要件	学位規則第4条第1項該当
研究科、専攻	東北大学大学院情報科学研究科（博士課程）情報基礎科学専攻
学位論文題目	Automatic Performance Tuning Methods for Heterogeneous Computing Systems （複合型計算システムにおける性能自動チューニング手法に関する研究）
論文審査委員	（主査）東北大学准教授　滝沢　寛之 東北大学教授　小林　広明　東北大学教授　青木　孝文 東北大学教授　中橋　和博（工学研究科）

論文内容の要旨

第1章　緒論

汎用プロセッサ(CPU)を組み合わせて構成される従来型の計算システムは、多数のプロセッサを組み合わせることで高い性能を達成している。近年、消費電力の制約から搭載するプロセッサ数を大幅に増加させることが難しくなっている。CPUの電力効率を大幅に向上させることも困難であるため、従来技術の延長で性能向上を続けることが難しくなっている。

複合型計算システムは、CPUとは異なる設計のプロセッサを搭載するシステムである。アクセラレータはCPUに比べて高い演算性能とメモリバンド幅を持っており、CPUと組み合わせて計算システムに搭載することで、一定の電力制約の中でより高い演算性能とエネルギ効率を達成することが可能となる。よって、複合型計算システムは科学技術計算の高速化に有望なシステム構成である。様々なアクセラレータの中でも、描画処理用プロセッサ(Graphics Processing Unit, GPU)は最も普及しているアクセラレータであり、通常のPCでも高い性能を容易に利用できる手段として注目を集めている。

複合型計算システムが持つ高い性能を利用するためには、プログラマが処理を適切にアクセラレータへ割り当てる必要がある。アクセラレータは処理内容によって実効性能が大きく変化する性質を持っている。不適切な処理をアクセラレータに割り当てた場合には、CPUに比べて実効性能が低くなるため、処理の割り当てを慎重に行わなければならない。しかし、熟練のプログラマであってもアクセラレータに割り当てるべき処理を決定することは難しく、実行時に処理内容が変化する場合には静的に最適な割り当てを決めることは困難である。常に最適な割り当てを実現するためには、実行時にアクセラレータに割り当てる処理を決める必要がある。

アクセラレータはCPUとは異なるメモリ階層を持っているため、アクセラレータに合わせた方法に従ってメモリにアクセスしなければ高い性能を達成することはできない。また、演算コアあたりに割り当てるスレッド数のように実行時に指定しなければならないパラメータが存在し、それらのパラメータは性能に大きな影響を与えるため、適切な値への調整が必要となる。アクセラレータの性能を引き出すようにプログラムを最適化することや、実行時パラメータを適切な値に設定するためにはアクセラレータについての

詳しい知識が必要となり、プログラマの試行錯誤による調整が要求される。

複数のアクセラレータを搭載している複合型計算システムの性能を引き出すためには、それぞれのアクセラレータに処理を適切に分散して割り当てる必要がある。適切な割り当てを実現するためには、アクセラレータごとに処理の実行時間を予測し、アクセラレータ間の演算負荷が均等になるように割り当てる必要がある。しかし、実行時に与えられるデータによって処理内容が変化する場合には、静的に最適な割り当てを決めることは困難である。また、複数のアクセラレータに処理を分散させるためには、プログラムの中から並列実行可能な部分を探し出し、並列に実行するようにプログラムを書き換える必要がある。

本論文は、複合型計算システムを利用する上で直面する以上の 3 つの課題に着目する。各章では、プログラマの判断において行っていた性能チューニングをプログラミングフレームワーク側で自動化するための手法を提案する。第 2 章では、プロセッサの実行時選択を自動化するためのプログラミング言語を提案する。第 3 章では、アクセラレータ依存のプログラムの最適化を自動化するための手法を提案する。第 4 章では、複数のアクセラレータに対しての負荷分散を自動化する手法を提案する。各章のそれぞれにおいて、プログラミングフレームワークが実行環境に対応して適切に自動性能チューニングを行うことが、複合型計算システムの性能を引き出すために効果的であることを定量的に評価し、複合型計算システム向けのプログラミングフレームワークに必要とされる要素技術を確立する。

第2章　ステンシル計算のためのプロセッサ実行時選択機能付きドメイン特化型言語

アクセラレータ上でプログラムを実行するためには、アクセラレータ用に定義された独自のプログラミング言語を用いてプログラムを記述する必要がある。そのため、実行時に使用するプロセッサを切り替えるためには、プロセッサごとにプログラムを用意しなければならない。本章では、実行時のプロセッサ切り替えを前提とした、プロセッサに依存しないプログラミング言語と実行時環境を組み合わせた SPRAT(Stream Programming-language with Runtime Auto-Tuning)を提案している。SPRAT 言語を用いて記述されたプログラムは、SPRAT が提供するコンパイラによって自動的に各プロセッサ用のプログラムへと変換される。そのため、計算システムに搭載されているプロセッサの種類に関係なくプログラムを記述することを可能としている。また、SPRAT の実行時環境は実行時に得られた処理対象のデータサイズなどの情報から、処理に最適なプロセッサを自動的に選択して割り当てる。

以上の提案手法について、流体シミュレーションと LU 分解のベンチマークを用いた性能評価の結果は、性能優先ポリシーと省電力優先ポリシーのどちらにおいても適切なプロセッサを自動的に利用可能であることを示している。また、ポリシーによって最適となるアクセラレータが異なる場合が存在することを示し、トレードオフが存在する場合であっても性能予測モデルに従って適切なプロセッサを選択可能であることを示した。

第3章　ドメイン特化型言語のための自動性能チューニング手法

アクセラレータの性能を最大限に引き出すためにはアクセラレータのアーキテクチャに合わせたプログラムの最適化を行う必要がある。本章では、実効メモリバンド幅を向上させるためにアクセラレータの一種である GPU のメモリ階層を適切に用いるための最適化を自動化する手法を提案する。また、GPU 上でプログラムを実行する上で必ず指定しなければならないコアあたりのスレッド数について、最適な値へと自動的に調整する手法を提案している。実効メモリバンド幅の向上手法では、第 2 章で定義した SPRAT 言語で記述されたプログラムに対して静的にメモリアクセスパターンの解析を行い、効率的なメモリアクセスになるようにプログラムを自動的に最適化する。コアあたりのスレッド数の自動設定では、短時間で

探索が完了するように、はじめにアクセラレータの特性や SPRAT 言語の性質を用いて最適値候補の絞り込みを行う。SPRAT コンパイラは絞り込まれた候補値を用いてアクセラレータ用のプログラムを生成し、プロファイリングに基づいて最適値を決定する。

これらの提案手法を LU 分解および流体シミュレーションのベンチマークの一つである姫野ベンチマークを用いて評価し、提案手法が実効メモリバンド幅の向上に効果があることを示した。また、同じベンチマークを用いてコアあたりのスレッド数の自動設定手法について評価し、どちらのベンチマークにおいても設定を自動的に調整可能であることを示した。

第4章　標準的プログラミング環境のためのオンラインタスクスケジューリング手法

複合型計算システムに搭載されている複数のアクセラレータを効率的に利用するためには、プログラマはアプリケーションが並列実行可能なタスクを多数持つようにプログラムを記述し、負荷分散を考慮してアクセラレータに割り当てなければならない。しかし、最適な負荷の分散は実行時に利用可能なアクセラレータの種類と数および入力データに依存するため、静的に最適な負荷分散状態を決定することは困難である。

本章では、標準的なプログラミング環境として提案されている OpenCL を用いたプログラムに対して、タスク間の並列性を可視化することでプログラムの並列化を支援する手法を提案し、プログラマの負担軽減を図っている。また、性能予測に基づいて MCT ポリシーに基づき動的にタスクスケジューリングする手法を提案している。効率的なスケジューリングを達成するためには高精度な実行時間予測が必要であるため、タスク実行時に与えられる引数などの値と実行時間の関係を分析し、高精度な性能予測モデルを構築する手法を提案している。これらの手法を併せて、並列化されたプログラムに対して実行時の環境に応じて自動的に最適な負荷分散を行う実行時環境を提案している。

モンテカルロ法を用いたベンチマークおよび 2 種類の実用的なプログラムを用いた評価により、提案手法は自動的に適切な負荷分散を実現可能であることを示した。また、実行環境によっては単純な均等負荷分散では性能低下する場合があり得ることを示すとともに、そのような場合には提案手法により負荷分散比を自動的に調整して性能低下を回避可能であることを示した。

第5章　結論

本論文では、複合型計算システムの高い性能を引き出す上で問題となっているプログラミング上の課題を解決するために、プログラミングフレームワークにおいて自動的に性能チューニングを行うための 3 つの手法を確立した。定量的な評価を通して、提案した自動性能チューニング手法がそれぞれの課題に対して効果的であることを示し、性能向上が可能であることを示した。提案手法をプログラミングフレームワークに組み込むことにより、複合型計算システムについて詳しい知識を持たないプログラマであっても、アクセラレータを意識せずにプログラミングすることが可能となり、アクセラレータが持っている高い演算性能を容易に利用することができる。

最終的に、本論文は将来の複合型計算システム向けのプログラミングフレームワークにおいて必要とされる要素技術について重要な知見を示した。

論文審査結果の要旨

複合型計算システムは汎用プロセッサとアクセラレータを搭載した高性能・高電力効率を実現可能なシステムアーキテクチャであり，科学技術演算の高速化に有望視されている．しかし，複合型計算システムで高い性能を得るためには，試行錯誤に基づく複雑な性能チューニングをプログラムに施さなくてはならない．本論文は，この性能チューニングを自動化する手法を確立し，複合型計算システムを容易に利用可能とする方法を論じたものであり，全編 5 章からなる．

第 1 章は緒論である．

第 2 章では，プロセッサの選択を自動化するために，SPRAT と名付けられたプロセッサ自動選択機能付きプログラミングフレームワークを提案している．SPRAT では，使用するプロセッサを考慮せずにプログラムを記述することを可能とする言語を提供している．また，データサイズと実行時間の関係から線形予測モデルを構築し，実行時間が最短となるプロセッサを自動的に選択する手法を実現している．本章で得られた知見は，言語に適切な制約を加えることで，煩雑なプロセッサ選択を自動化可能であることを示したものであり，大変有益である．

第 3 章では，アクセラレータごとに求められるプログラムの最適化を自動化するために，アクセラレータ特有の性質を適切に利用する手法と，実行時パラメータの値を決定する手法を提案している．SPRAT 言語の性質を利用してメモリアクセスを解析し，実効メモリバンド幅を向上させる最適化を自動化する手法と，アーキテクチャ的な観点から有望な設定候補を絞り込み，その中から適切な値を自動的に決定する手法を提案している．本章で得られた知見は，最適化に重要な情報を容易に解析できるように言語を設計することで，煩雑なアクセラレータ特有の最適化と実行時パラメータの決定を自動化可能であることを示したものであり，有益な成果である．

第 4 章では，複数のアクセラレータ間の負荷分散を自動化する手法を提案している．そのために，実行時情報を用いて複数の線形予測モデルを使い分ける高精度性能予測手法と，タスクの並列化支援のための依存関係解析手法を提案している．加えて，性能予測に基づくオンラインタスクスケジューリング手法を提案している．本章で得られた知見は，実行時間予測に基づいてタスクスケジューリングを行うことで，複合型計算システム向けプログラムの負荷分散を自動化できることを明らかにしたものであり，複数アクセラレータを自動的かつ適切に利用するために極めて有用な成果である．

第 5 章は，本論文を総括し，結論としている．

以上要するに本論文は，プログラム解析結果と実行時に得られる情報に基づき，複合型計算システムに搭載されているプロセッサ群を効率的に用いるための自動チューニング手法を確立し，定量的な評価を通して複合型計算システム向けのプログラミングフレームワークの構成法を論じ，まとめたものであり，情報基礎科学および計算機科学の発展に寄与するところが少なくない．

よって，本論文は博士（情報科学）の学位論文として合格と認める．