様式 C-19

科学研究費補助金研究成果報告書

平成２１年　３月３１日現在

研究種目：若手研究(B)
研究期間：2007～2008
課題番号：19700020
研究課題名（和文）
ヘテロジニアス・マルチコア時代の統一的ソフトウェア開発手法に関する研究
研究課題名（英文）
A study of a unified software development scheme in the heterogeneous multicore era
研究代表者
滝沢　寛之（TAKIZAWA HIROYUKI）
東北大学・大学院情報科学研究科・准教授
研究者番号：70323996

研究成果の概要：
本研究では，CPU と描画処理用ユニット(Graphics Processing Unit, GPU)を搭載している一般的な PC を想定し,その CPU と GPU 間での移植性を維持しつつ両者を効果的に利用可能な高級プログラミング言語処理系として SPRAT(Stream Programming with Runtime Auto-Tuning)を設計・実装・評価した．LU 分解および 2 次元非圧縮流体シミュレーションをアプリケーション例として SPRAT 言語で記述し，PC に搭載された CPU と GPU の性能差や問題サイズなどの実行時パラメータに応じて，SPRAT 実行環境がプロセッサを適切に自動切り替え可能であることを評価実験により示した．また，実行時間が最短になるようにプロセッサを切り替える手法や，アプリケーション実行に要するエネルギ消費を最小にするプロセッサ切り替えなどを試し，SPRAT 実行時環境がそれぞれの観点から適切にプロセッサを切り替え可能であることを示した．さらに,高水準の SPRAT 言語から GPU 向けの CUDA コードを生成する言語処理系(SPRAT コンパイラ)に 2 種類の自動最適化機能を実装し，それらの演算性能への影響を評価した．その結果，自動最適化機能により，SPRAT コンパイラによって自動生成されたコードを実行した際の性能を大幅に改善できることが示された．

交付額

（金額単位：円）

	直接経費	間接経費	合　計
平成 19 年度	1,100,000	0	1,100,000
平成 20 年度	2,000,000	600,000	2,600,000
総　計	3,100,000	600,000	3,700,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウェア
キーワード：ﾊｲﾊﾟﾌｫｰﾏﾝｽｺﾝﾋﾟｭｰﾃｨﾝｸﾞ　情報システム　自動チューニング　GPU ｺﾝﾋﾟｭｰﾃｨﾝｸﾞ

１．研究開始当初の背景

シングルスレッドの実行性能を損なうことなくマルチコア化の恩恵を享受できることから，特定用途のコプロセッサを混載するヘテロジニアス・マルチコア・プロセッサが注目されている．その一方で，ヘテロジニアス・マルチコア化が進むことによってプロセッサのハードウェア構成が多様化し，プログラマがそれぞれの構成を意識したプログラミングを強いられることが深刻な問題となりつつある．そのようなソフトウェア開発において異機種間のソースコードレベルの移植性を極力維持するためには，少なくとも個々のコプロセッサに依存する記述が他の部分と明確に切り離されている必要がある．しかし,多くのコプロセッサ用の Application Programming Interface(API)では，対象とするコプロセッサの効果的な抽象化のみが考

慮されており，異なる API 間での互換性は考慮されていない．このように，各コプロセッサをそれぞれ独自の方式で抽象化している限り，異機種間のソースコードレベルの移植性を維持するのは困難である．

しかしながら，マルチコア・プロセッサの各種コアや，近い将来プロセッサと統合されることが確実視されている各種演算コプロセッサでのデータ並列処理を統一的に記述するための検討はほとんどなされていない．このため，各種演算コプロセッサに対する統一的ソフトウェア開発手法を検討することによって，ヘテロジニアス・マルチコア化によるハードウェア構成の多様化に対応可能な，重要な知見をもたらすものと期待できる．

２．研究の目的

プロセッサのマルチコア化は従来のソフトウェア開発のスタイルを大きく変化させ，プログラマは常にハードウェア構成を意識して並列処理プログラムを開発することになる．複数のコアによる並列処理を明示的に記述し，適切に制御する技術がソフトウェア最適化・高速化の要となる．すなわち，プログラマは複数のコアにおける処理を明示的に記述して制御する必要がある．このため，個々のコプロセッサを適切に抽象化して利用するための技術がこれまで多く研究されてきた．しかしながら，同じ処理を実装する場合であっても，搭載しているコプロセッサの種類によって全く異なる API を利用する必要があり，またハードウェアの構成を充分に考慮した最適化が求められるという問題がある．

本研究の目的は，多種多様なデータ並列処理用コプロセッサを利用するための共通インタフェースを確立することである．ただし，各コプロセッサの機能と性能は大きく異なるため，全ての機能に対するインタフェースの共通化は困難である．このため用途とコード記述の柔軟性を制限し，その範囲内で各コプロセッサ用 API から共通性を見出すことによって，必要最小限の共通インタフェースを定義する．可能な限り共通インタフェースのみを用いてソフトウェアを開発することにより，ソースコードレベルの移植性を高く維持しつつ，必要に応じて個々のコプロセッサの性能を効果的に利用することが可能となる．

各種コプロセッサを独自の方式で抽象化している限りにおいては，異機種間でのソースコードレベルの移植性が犠牲となる．この問題の解決のために，データ並列処理用コプロセッサの統一的な抽象化手法を検討し，それらの共通インタフェースを設計することが本研究の特色である．ソースコードのうち，高速化や独自機能の利用のためにコプロセッサ特有の記述を必要とする部分と，コプロセッサの種類に依存しない部分を明確に区別することにより，高い移植性と実効性能を両立する．抽象化されたコプロセッサの共通インタフェースを規定することによって，複数の多様なコプロセッサを同時に利用するプログラミング方法の共通化が図られ，特にヘテロジニアス・マルチコア・プロセッサ向けのソフトウェア開発技術の進展に大きな役割を果たすものと考えられる．

３．研究の方法

用途やプログラミングモデルに一定の制約条件を設け，その条件の範囲内でデータ並列処理用コプロセッサ(アクセラレータ)を利用するための必要最小限の共通インタフェースを設計することにより，異機種間での移植性を維持しつつ各コプロセッサを効果的に利用することを目指す．このために，複数の種類のコプロセッサを統一的に利用可能なソフトウェア開発ツールを構築する．このためには，少なくとも以下の項目について検討する必要がある．

(1) 複数のコプロセッサによるデータ並列処理を統一的に記述するプログラミングモデル

ヘテロジニアス・マルチコア・プロセッサにおいては，複数のコプロセッサへ処理を適切に割り当てることが必要不可欠である．しかし，既存の抽象化手法ではそのような手段が提供されていない．このため，複数のコプロセッサを効果的に扱うためのプログラミングモデルが重要な検討課題である．

(2) メモリ参照の局所性やメモリ階層の抽象化手法

CELLプロセッサのSPE(Synergistic Processing Element)のように，高速にアクセス可能な固有のメモリ領域を持ったコプロセッサの場合，そのメモリ領域でのデータ管理が重要である．データの局所性を最大限に利用するために，データ並列処理用コプロセッサのメモリシステムの抽象モデルを確立する．

(3) 共通インタフェースをプログラマに提供する手段

既存のプログラミング言語を拡張すべきなのか，あるいは高級ライブラリとして提供すべきか等，開発ツールの記述性，保守性，および可搬性を検討する．

描画処理用ユニット(Graphics Processing Unit, GPU)は，アクセラレータとして現在最も一般的に用いられている．しかし，現在盛んに行われているGPUを用いた計算(GPUコンピューティング)では，NVIDIA 社の Compute

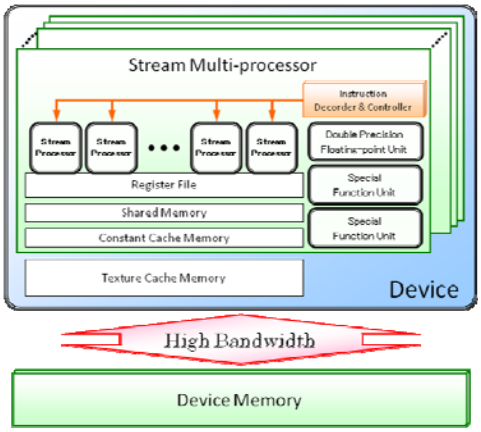

図 1 CUDA アーキテクチャ

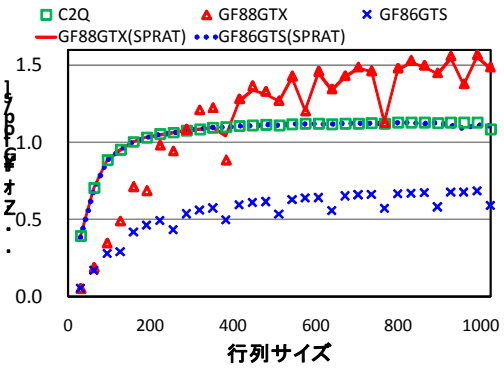

図 2 プロセッサ自動切替の効果

Unified Device Architecture(CUDA)などの API を用いて，図 1 に示す複雑なハードウェア構成を強く意識した抽象度の低いレベルでのプログラム開発が行われている．このため GPU の世代間の移植性を考えると，現在よりもさらに抽象度の高いプログラム開発環境が求められている．

本研究では，CPU と GPU を搭載している一般的な PC を想定し，その CPU と GPU 間での移植性を維持しつつ両者を効果的に利用可能な高級プログラミング言語処理系を構築する．

４．研究成果

本研究では，プログラマが個々のプロセッサの違いを意識しない，ハードウェアが高度に抽象化されたプログラム開発環境として SPRAT (Stream Programming with Runtime Auto-Tuning)を構築した．SPRAT では，異種複数のプロセッサを搭載するシステムで，各処理を実行するプロセッサを自動選択する．

本研究では，その最適なプロセッサ選択が実行時の様々な要因に依存していることを様々な計算カーネルにおける性能解析により明らかにした．その結果に基づき，SPRAT

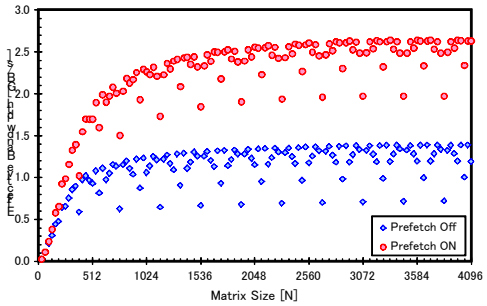

図 3 自動最適化の効果

には実行時性能予測に基づいてプロセッサを自動選択する機能を実現した．LU 分解および 2 次元非圧縮流体シミュレーションをアプリケーション例として SPRAT 言語で記述し，PC に搭載された CPU と GPU の性能差や問題サイズなどの実行時パラメータに応じて，SPRAT 実行環境がプロセッサを適切に自動切り替え可能であることを評価実験により示した．

実行時間が最短になるようにプロセッサを切り替える手法や，アプリケーション実行に要するエネルギ消費を最小にするプロセッサ切り替えなどを試し，SPRAT によるプロセッサ自動切替の有効性を評価した．その結果，SPRAT が問題サイズなどの実行時パラメータに応じて，それぞれの観点から適切にプロセッサを切り替え可能であることが示された．LU 分解の SPRAT コードに対して，実効演算性能に基づくプロセッサ自動切替を行った評価の結果を図 2 に示す．本評価で用いた LU 分解のコードは，GPU が不得意なメモリアクセスパターンを頻繁に行うため，評価に利用した NVIDIA GeForce 8600GTS(GF86GTS)や GeForce 8800GTX(GF88GTX)では高い性能を発揮することができない．その結果，図 2 に示すとおり，CPU(図中 C2Q)と GF88GTX の実効演算性能は行列サイズに依存して逆転している．このようなアプリケーションに対して，SPRAT では実効演算性能が高い方のプロセッサを適切に選択できている．

また，高水準の SPRAT 言語から GPU 向けの CUDA コードを生成する言語処理系(SPRAT コンパイラ)に 2 種類の自動最適化機能を実装し，それらの演算性能への影響を評価した．SPRAT コンパイラではデータの再利用性を解析し，再利用性の高いストリームデータを GPU のオンチップメモリへと複製(プリフェッチ)するコードを生成できる．図 3 にプリフェッチ自動最適化の効果を示す．再利用されるデータを高速なオンチップメモリ(図 1 中の Shared Memory)にプリフェッチすることにより，実効メモリ帯域幅が飛躍的に向上していることが分かる．ストリーム処理におい

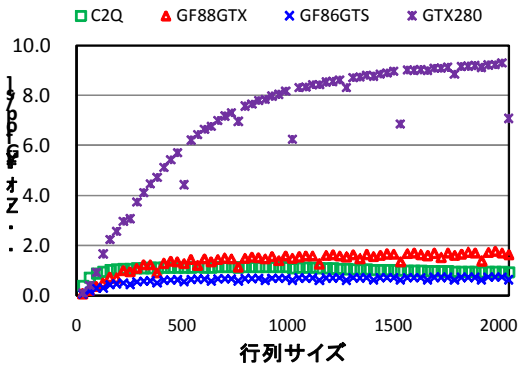

図 3 GPU の世代による相違

ては，実効メモリバンド幅が性能の律速条件になる場合が多いため，実効メモリバンド幅を高めることができる本自動最適化手法は極めて有効である．

また，同一のコードに対しても GPU の世代によって挙動が異なる．図 4 は図 2 の評価に使用した LU 分解のコードを新しい世代の GPU である NVIDIA GeForce GTX280（図中 GTX280）で実行した結果を示している．新しい世代の GPU では，不得意なメモリアクセスパターンを行ったときの性能低下が軽減されているため，GTX280 の性能は他の GPU と比較して非常に高くなっている．このように GPU の世代によって挙動が異なる場合や，異なる最適化技法が求められる場合にも，それを SPRAT コンパイラは自動的に判断して適切な最適化技法を GPU 用のコード生成時に適用できる．実際，SPRAT コンパイラは非効率的なメモリアクセスを 2 回の効率的なメモリアクセスで置き換えることが可能である．GPU の世代によっては，非効率なメモリアクセスパターンが効率的なメモリアクセスパターンの 10 倍近く遅いため，そのような置き換えによる最適化が効果的である．

これらの自動最適化機能により，SPRAT コンパイラによって生成されたコードを実行した際の性能を大幅に改善できる．

加えて，GPU を利用することによって飛躍的な計算性能向上を達成できることを実証するため，アプリケーション例として交差判定処理の効果的な GPU 実装の開発にも取り組んだ．

５．主な発表論文等

〔雑誌論文〕（計 3 件）

① H. Takizawa, K. Sato, and H. Kobayashi, “SPRAT: Runtime Processor Selection for Energy-aware Computing,” in Proceedings of the 2008 IEEE Conference on Cluster Computing, pp. 386—393, 査読有り, 2008.

② 滝沢寛之, 白取寛貴, 佐藤功人, 小林広明, “SPRAT:実行時自動チューニング機能を備えるストリーム処理記述用言語,” 情報処理学会論文誌：コンピューティングシステム(ACS), 1 巻 2 号, pp.207-220, 査読有り, 2008.

③ K. Komatsu, Y. Kaeriyama, K. Suzuki, H. Takizawa, and H. Kobayashi, “An Efficient Intersection Algorithm Design of Ray Tracing For Many-Core Graphics Processors,” The 10th IASTED International Conference on Computer Graphics and Imaging (CGIM 2008), pp.165-171, 査読有り, 2008.

〔学会発表〕（計 7 件）

① 佐藤功人, 滝沢寛之, 小林広明, “ストリーム処理記述言語のGPU向け自動最適化の検討,” 情報処理学会先進的計算基盤システムシンポジウム(SACSIS2009), 2009 年 5 月 29 日, 広島

② 小山賢太郎, 佐藤功人, 小松一彦, 滝沢寛之, 小林広明, “GPU向け線形代数ライブラリの性能評価,” 計算工学講演会, 2009 年 5 月 14 日, 東京大学

③ 佐藤功人, 滝沢寛之, 小林広明, “GPUを効率的に利用するための言語拡張と自動最適化手法,” 並列/協調/分散処理に関するサマーワークショップ(SWoPP2008), 2008 年 8 月 7 日, 佐賀.

④ 滝沢寛之, 佐藤功人, 小林広明, “GPUコンピューティングのためのストリーム処理記述言語,” 第 36 回可視化情報シンポジウム, 2008 年 7 月 23 日, 工学院大学.

⑤ 滝沢寛之, 白取寛貴, 佐藤功人, 小林広明, “SPRAT: 実行時自動チューニング機能を備えるストリーム処理記述用言語,” 情報処理学会先進的計算基盤システムシンポジウム(SACSIS2008), 2008 年 6 月 12 日, 筑波.

⑥ H. Takizawa, H. Shiratori, and H. Kobayashi, “Preliminary Evaluation for Runtime Auto-Tuning of GPGPU Applications,” The 2nd International Workshop on Automatic Performance Tuning (iWAPT2007), 2007 年 9 月 20 日, 東京大学.

⑦ 白取寛貴, 滝沢寛之, 小林広明, “実行時性能予測に基づくCPUとGPUへの動的タスク割当の検討,” 並列/協調/分散処理に関するサマーワークショップ(SWoPP2007), 2007 年 8 月 3 日, 旭川

６．研究組織

(1)研究代表者

滝沢　寛之（TAKIZAWA HIROYUKI）

東北大学・大学院情報科学研究科・准教授

研究者番号：70323996

(2)研究分担者
(　　)
研究者番号：
(3)研究連携者
(　　)
研究者番号：