様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２８　年　　６　月　　３　日現在

機関番号：　３２６６０
研究種目：　基盤研究(C)（一般）
研究期間：　2013～2015
課題番号：　２５３３００８９

研究課題名（和文）COINSコンパイラ共通基盤のGPU向け拡張

研究課題名（英文）Extension for GPU of CONS compiler infrastructure

研究代表者
滝本　宗宏（Takimoto, Munehiro）

東京理科大学・理工学部・教授

研究者番号：００３１８２０５

交付決定額（研究期間全体）：（直接経費）　　3,700,000円

研究成果の概要（和文）：最適化器を容易に記述できるコンパイラ共通基盤のCONISと，GPUのコードを生成できるLLVMの中間表現LIRとLLVM-IRの間に変換器を定義し，COINS上でGPU向け最適化器を実現できるようにした．また，GPU向け最適化の実例として，GPUに有効な通常最適化と，GPU特有の最適化を実現した．通常最適化として，配列参照を次元の一致度を考慮して集約し，キャッシュのヒット率を上げる手法や，任意の構造をもつループに対してスカラ置換を行う要求駆動型手法を実現した．GPU特有の最適化として，分岐先を，依存関係を考慮して，括り出したり，融合したりすることで，より多くの分岐発散を低減する手法を実現した．

研究成果の概要（英文）：In order to implement a new compiler infrastructure where optimizers for GPU can be easily implemented on simple intermediate representation LIR of COINS infrastructure, I have implemented a transformer between LIR and another intermediate representation LLVM-IR of LLVM, which can generate the code for GPU. Furthermore, to illustrate optimizations for GPU, I have proposed and implemented general optimizers effective for GPU, and GPU specific optimizers. As general optimizers, I have implemented a technique that enhances cache-hits through preceedingly aggregating the same array references with more same
indexes in higher dimensions. Also, I have implemented a technique that achieves scalar replacement for loops with any structures based on question propagation. As GPU specific optimizers, I have implemented a technique for effectively
suppressing branch divergence caused on GPU through code fusion or code factoring of branch paths based on data dependencies.

研究分野：　プログラミング言語処理系

キーワード：　コンパイラ　コード最適化　並列化　GPU　キャッシュ効率化　要求駆動型データフロー解析　静的単一代入形式　網羅型データフロー解析

１．研究開始当初の背景

グラフィックアクセラレータから発展したGPU（Graphics Processing Unit）は，その多くの演算器を用いて並列計算を行う構造から，画像処理以外の汎用目的の計算，GPGPU（General Purpose computing on GPU）に広く応用されるようになってきた．しかしながら，効率のよい並列計算を行うGPGPUプログラムを実現するためには，多くの知識と経験が要求される．　また，GPGPU プログラムは，CUDA のような高水準言語で記述されることが多く，低水準なコード最適化を実現するのが難しかった．

一方，仮想マシンの共通基盤として利用が進んでいるLLVM が，GPU 向けの実装によって拡張された．LLVM は，低水準表現LLVM-IR を入力として，Just-In-Time コンパイラ（以降，JIT と呼ぶ）の適用や，静的コンパイルを行うと同時に，コンパイル時，リンク時，実行時といった複数のタイミングで多くのコード最適化を適用する．GPU 向けの LLVM は，LLVM-IR の一部として，GPU 向けの低水準表現 NVVM-IR を備えている．すなわち，LLVM-IR 上で，GPU 向けの低水準なコード最適化器を実現できる．

本研究では，COINSのLIR をLLVM-IR に変換することによって COINS と LLVM を組み合わせた共通基盤を実現するとともに，LIR 上で CPU と GPU とが協調動作するプログラム向けの高度なコード最適化を実現する．最終的に，本研究で目指しているコード最適化は，次の 2 つである．

(1) CPU と GPU によって構成されるメモリ階層に基づいて，効率的なメモリアクセスを実現する．
(2) (1) の結果として，局所的なメモリアクセスが可能になった部分を抽出し，自動並列化を実現する．

２．研究の目的

本研究は，COINS と LLVM の組合せた共通基盤を実現するステップと，その共通基盤上で，GPU 向けのコード最適化を実現するステップに大別できる．

(1) COINS と LLVM の組合せで明らかにする点は，次のとおりである．
- **LIR の拡張と LLVM-IR への変換**
  LLVM-IR と NVVM-IR を調査し，LIR から直接変換できない部分は，LIR に拡張を加える．次に，COINSの生成コードの定義である TMD（Target Machine Description）に，LIR パタンとLLVM-IR パタンの対応を記述することによって，LIR，LLVM-IR 変換器を実現する．

(2) GPU 向けのコード最適化の実現では，これまでに，LIR 上で実現してきた手法の拡張を含め，LLVM と組み合わせたCOINS のLIR 上で，GPU 向けコード最適化手法を提案，実現する．拡張手法としては，次の手法を実現する．

① **コード移動に基づくロード命令集約の拡張**
コード移動に基づくロード命令集約は，同じ配列の高位次元にアクセスするロード命令が連続するように，優先的にコード移動させる手法であり，本手法を CPU と GPU のメモリ階層に拡張することによって，GPU から CPU の主メモリへのアクセスを低減できる可能性がある．

② **冗長性に基づくスカラ置換の拡張**
冗長性に基づくスカラ置換は，ループの繰返し間で添字は異なっているが同じアドレスにアクセスする配列参照をレジスタで置き換えるスカラ置換（scalar replacement）を，冗長性除去の観点から一般化したものである．本手法を，コード移動を用いて拡張することによって，GPU から CPU の主メモリへのアクセスをさらに低減させる．

③ **GPU 内の局所メモリにだけアクセスするプログラム部分の抽出と，自動並列化**
プログラム中から，メモリアクセスが GPU 内に局所化された部分を抽出する手法を確立し，COINS に備わっているループ並列化，SIMD 並列化，ソフトウェアパイプライニングを，抽出プログラムに適用できるように拡張することによって，自動並列化，あるいは，従来法より抽象的なディレクティブに基づく自動並列化を実現する．

３．研究の方法

基本的な流れは次のとおりである．

(1) LLVM-IR および NVVM-IR の仕様を調べるとともに，LLVM 用の CUDA を用いて，生成されるコードを観察し，LIR で表現できないコードパタンを整理する．
(2) LLVM-IR がもつ CUDA 特有のディレクティブを表現できるようにLIR を拡張する．
(3) LIR からLLVM-IR への変換器を実現する．
(4) CUDA のプログラムを，LLVM で拡張したCOINS 用に変形し，生成された実行コードの振舞いを検査する．
(5) 命令スケジューリングを用いた，CPU とGPU 間のメモリアクセスの効率化手法を中心に文献を調査する．
(6) 冗長なメモリアクセスを除去するとともに，同じ配列への参照を，高位次元が一致しているものから順に，連続するように並べ替え，キャッシュのヒット率を向上させる手法を実現する．
(7) 任意のループにっいて，繰返し間で同じアドレスにアクセスしている配列参照をレジスタで置き換えるスカラ置換を

実現する．
(8) GPU 実行に有効なメモリにアクセスしないプログラム部分を抽出し，COINS に備わっている並列化手法を適用することによって自動並列化を実現する．
(9) 時間に余裕があれば，実行効率と消費電力低減を両立する新しい手法を提案，実現する．
(10) 成果を発表する．

４．研究成果

(1) 配列次元に基づく大域ロード命令集約

現在の多くのコンピュータは，高速な CPU と低速な主メモリによって構成されており，プログラムの実行中に主メモリにアクセスするロード命令が実行されると，プログラムの実行効率が低減される可能性がある．そこで，多くの CPU には，キャッシュメモリが備えられており，ロード命令を実行する際は，まず，キャッシュメモリにアクセスアドレスの値が保持されていないかチェックする．一旦，主メモリへのアクセスが生じると，アクセス箇所の近傍の値がキャッシュメモリにコピーされる．次にロード命令が実行される際に，キャッシュメモリにアクセスアドレスの値が見つかりキャッシュがヒットすれば，ロード命令は，主メモリにアクセスすることなく効率的に実行できる．一方，キャッシュにアクセスアドレスの値が見つからずキャッシュミスすれば，主メモリへのアクセスが生じ，実行が非効率になる．

本手法は，同じ配列にアクセスするロード命令が連続して実行されるように集約し，キャッシュのヒット率を向上させる．このとき，高位次元の添え字が多く一致しているものを優先して集約する．ロード命令の集約は，部分冗長な式を解析する過程を，同じ配列にアクセスするロード命令を解析するように置き換えることで実現する．本手法は，Lazy Code Motion のアルゴリズムを採用しているので，集約が可能でなければ，プログラムの変形は行わない．また，集約が可能であった場合でも，不要な巻上げを避けるために巻き戻す．この際，最も多く高次元の添え字が一致した配列参照まで巻き戻すことによって，メモリ上でより近い配列参照を集約することができる．

本手法を COINS の最適化器として実現し，ベンチマークプログラムに適用して効果を調べたところ，キャッシュのヒット率を向上できることを確認した．

(2) COINS の LIR と LLVM の相互変換の実現

GPU 向けの LLVM は，LLVM-IR の一部として GPU 向けの低水準表現 NVVM-IR を備えている．この LLVM-IR と COINS の LIR の相互変換を実現することによって，COINS を GPU 処理系の一部として使用できるようになり，LIR 上で，GPU 向けの低水準なコード最適化器を実現できるようになる．

実現した変換器が与える影響を調べるために，特に，LIR から LLVM への変換部を用いて，SPEC ベンチマーク上での性能評価を行った．その結果，変換が正しく行われていることを確認したが，変換を行わず LLVM だけでコード生成したものと比較して，実行効率が低減することが分かった．

(3) 要求駆動型スカラ置換

ループの繰返しを越えて冗長になる配列参照をレジスタ参照に変更するコンパイラのコード最適化をスカラ置換という．本手法は，ループの１つの繰返しの中だけで冗長性を調べる部分冗長除去を，要求駆動型手法を用いて，異なる繰返しに適用できるように拡張する．要求駆動型の部分冗長除去では，各配列参照の冗長性について，質問を伝播するが，質問が，配列の添字の変数を定義している文に到達すると，添字の変数を文の右辺の式で置き換える．この代数変換によって，配列参照の表現を，以前の繰返しの変数を用いた形式に変換でき，繰返しを跨いだ冗長性の除去を実現する．

本手法の効果を確かめるために，SPEC ベンチマーク上で性能評価を行ったところ，スカラ置換の従来法と比べてプログラムの実行効率を最大で約 2.3%向上させることができることを確認した．

(4) 依存関係に基づく括出し法

SIMD 実行によって引き起こされる分岐発散は，GPU の性能低下を引き起こす主要な問題である．この分岐発散を改善する手法として，括出しと呼ばれる最適化が有効であることが知られている．しかし，従来の括出し法では，依存関係をもつ複数の計算を括り出す際，新たな分岐の挿入を複数回必要とする場合があった．本手法では，演算子の一致性だけに基づいた従来の括出し法を拡張し，括り出す計算の依存関係も考慮した括出し法を実現し，より大きな依存構造をもつ計算を優先して括り出すことによって，従来法よりも多くの括出しを，一度の分岐挿入で実現する．また，各分岐の訪問順序を考慮して括り出すことで，入れ子になった分岐からの括出しも可能になる．

本手法の有効性を示すために，SPEC と GPU の各ベンチマークを用いて実験を行ったところ，従来法と比較し，最大で 77%多くの括出しを実現し，最大で約 12%の実行速度向上が得られることを確認した．

(5) 依存関係に基づく分岐融合

GPU の SIMD 実行は，ウォープ内に異なる分岐経路に従うスレッドが存在するとき，分岐発散と呼ばれる効率の低下を生じる場合がある．この分岐発散を改善する有効な手段として，分岐融合が知られている．分岐融合は，分岐先の命令列を，同じ演算命令を共有するように組み合わせることで，分岐の実行コス

トを低減する．この命令列の組み合わせの際に，共有する命令のオペランドが異なっていたり，共有できる命令が存在しなかったりすると，分岐融合は，異なる変数を選択的に参照する選択命令を挿入することによって，プログラムの意味を保存する．このとき，命令列の並びによっては，選択命令の挿入が過剰になり，実行効率が低減する可能性がある．

本手法では，分岐融合の前処理として，依存関係に基づく命令の整列を行う．本手法は，演算子の一致性だけでなく，命令の依存関係も一致するように，命令を並べ替える．本整列手法を適用することによって，多くの共有する命令のオペランドが唯一になり，従来法に比べ，選択命令の挿入を抑制することができる．本手法の効果を示すために，本手法を Ocelot CUDA コンパイラ上に実装し，複数のベンチマークを用いて実験を行った．その結果，本手法によって，生成コードの実行効率が，最大で 11.3%以上向上することを示した．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 2 件）

① 澄川靖信，小島量，滝本宗宏：要求駆動型スカラ置換，日本ソフトウェア科学会論文誌コンピュータソフトウェア，査読有，Vol.32，No.2，2015，pp.93–113，http://ci.nii.ac.jp/naid/130005086348

② Yasunobu Sumikawa and Munehiro Takimoto, Effective Demand-driven Partial Redundancy Elimination, IPSJ Trans. Programming, 査読有, Vol.6, No.2, 2013, pp.33-44, http://id.nii.ac.jp/1001/00094920/

〔国際会議プロシーディングス〕（計 1 件）

① Yasunobu Sumikawa and Munehiro Takimoto, Global Load Instruction Aggregation Based on Array Dimensions, Proc. of Sixth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP2014), 査読有, 2014, pp. 123-129, 10.1109/PAAP.2014.43

〔学会発表〕（計 7 件）

① 那須 孝志，滝本 宗宏，依存関係に基づく Branch Fusion，日本ソフトウェア科学会第 18 回プログラミングおよびプログラミング言語ワークショップ（PPL2016），査読無，2016 年 3 月 8 日，渋川（岡山県）

② 那須孝志，滝本宗宏，依存関係に基づく括出し法，研究報告ソフトウェア工学（SE），査読無，Vol.2015-SE-189，No.9，2015 年 7 月 23 日，pp.1-6，札幌市教育文化会館（北海道）

③ Takashi Nasu and Munehiro Takimoto, Dependency Based Factoring, The 24th International Conference on Parallel Architectures and Compilation Techniques (PACT2015), 査読有, 2015 年 10 月 20 日, San Francisco (USA)

④ 澄川靖信，滝本宗宏，コンパイラのコード最適化に基づいたレジスタとキャッシュメモリの使用の効率化，情報処理学会第 77 回全国大会講演論文集，査読無，Vol.2015，No.1，2015 年 3 月 17 日，pp.295-297

⑤ 酒井宏城，澄川靖信，滝本宗宏，コンパイラ共通基盤 COINS の LLVM 向け拡張，情報処理学会第 55 回プログラミング・シンポジウム プログラム，2014 年 1 月 12 日，ラフォーレ倶楽部伊東温泉湯の庭（静岡県）

⑥ Yasunobu Sumikawa and Munehiro Takimoto, Improving Memory Hierarchy Performance Based on Code Motion, 12th Asian Symposium on Programming Languages and Systems (APLAS2014), 査読有, 2014 年 11 月 17 日,Singapore

⑦ Yasunobu Sumikawa and Munehiro Takimoto, Global Load Instruction Aggregation Based on Array Dimensions, 11th Asian Symposium on Programming Languages and Systems (APLAS2013), 査読有, 2013 年 12 月 10 日, Melbourne (Australia)

６．研究組織
(1)研究代表者

滝本　宗宏（TAKIMOTO MUNEHIRO）
東京理科大学・理工学部・教授
研究者番号：００３１８２０５