様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成 24 年　6 月 28 日現在

機関番号：84304
研究種目：若手研究(B)
研究期間：2008 年度～2011 年度
課題番号：20700039
研究課題名（和文）　静的型付関数型言語の SIMD 演算・並列化拡張の型に基づく高性能な実装に関する研究
研究課題名（英文）　Research on type-based efficient implementation of SIMD and parallel extensions for the statically typed functional programming language
研究代表者　吉田　信明（YOSHIDA NOBUAKI）
財団法人京都高度技術研究所・研究部　副主任研究員
研究者番号：00373506

研究成果の概要（和文）：Standard ML 言語に対し，SIMD 演算に必要なベクトルデータ型および操作と，GPGPU による並列演算機構を，必要に応じて言語プリミティブや型を拡張して導入し，シームレスに静的型付関数型言語から並列演算を駆動できるようにした．これらの拡張は，現在実験的に個別に SML#言語処理系に実装されているが，研究期間終了後にリリースされた SML#処理系の正式版に対して今後移植する予定である．

研究成果の概要（英文）：We realized parallel computation mechanisms on the statically-typed functional programming language by implementing SIMD operations with vector data types and GPGPU parallel processing extensions on “SML#” compiler. Currently, these extensions are experimental, and each functions are implemented on previous versions of SML# compiler independently;　these will be ported to the latest version of the compiler.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
2008 年度	900,000	270,000	1,170,000
2009 年度	700,000	210,000	910,000
2010 年度	900,000	270,000	1,170,000
2011 年度	700,000	210,000	910,000
年度			
総　計	3,200,000	960,000	4,160,000

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:ソフトウェア学，言語処理系，並列計算,ハイパフォーマンスコンピューティング

１．研究開始当初の背景

PC などの分野において，単一のプロセッサコアの性能向上が限界に達しつつあり，マルチスレッド化・並列化などのソフトウェアも含めた最適化が必要になったため，汎用プロセッサと特定の用途向けの異なるアーキテクチャのプロセッサを組み合わせたマルチプロセッサ演算環境が主流となりつつある．

製品としては，単一パッケージに異種の複数のプロセッサを集積したヘテロジニアス・マルチコアプロセッサ(Cell Broadband Engine(SCE，東芝，IBM) など）や，汎用 CPU とグラフィックプロセッサ組み合わせて高速な演算を実現する GPGPU (CUDA(NVidia)など)が普及しつつある．このような環境では，ベクタなどのデータ列に対して，細粒度の並

列演算を実行する機能を提供しており，また，Single Instruction Multiple Data(SIMD，単一命令による複数データ処理）演算機能も提供されている．この機能は，2〜4 要素程度の整数や浮動/固定小数点数の数ワード長のベクタに対する加減乗除などの演算を 1 命令で行う機能である．このような SIMD 演算はマルチメディア処理や 3 次元処理など，現在のアプリケーションの性能を高めるには必須の機能となっている．

これらの環境向けの開発は，C 言語などの手続き型言語が主流であり(CUDA (NVidia)，など)，言語を上記のベクタ型と並列化構文で拡張するなどして機能を提供している．その一方で，SIMD 演算や並列処理のプログラミングは複雑であり，C 言語では実行時の型安全性などの信頼性・安全性は保証されない．このような，言語処理系によるサポートのない煩雑なプログラミングは，開発者にとって負荷の高い作業である．

２．研究の目的

この研究の目的は，上に述べたようなプログラマにとって煩雑で負荷の高い，SIMD や GPGPU などを用いたプログラミングに対して，信頼性・安全性の高いプログラミング環境を提供することを目的としている．ここで実現されるプログラミング環境を用いることで，プログラマは，C 言語などでしばしば障害となるポインタ処理の不備による実行時のメモリエラーなどを回避できる．

並列計算・並列処理は，今後，社会的ニーズの高い重要な分野であると考えられる．この研究は，この分野におけるソフトウェア開発の生産性向上に貢献することを目的としている．

具体的な目標として，この研究では，異なるアーキテクチャを持つプロセッサが混在するマルチプロセッサ環境における，SIMD 演算を用いた，高性能かつ安全性・信頼性の高い並列演算が可能な静的型付関数型言語 (Standard ML) 処理系の実現をめざす．この処理系の中では，従来の Standard ML の型と同様に，ベクタ型や GPGPU 上のデータへのポインタを取り扱うことができるようにし，これに基づき，型安全性を保証する．また，GPGPU 上で駆動されるプログラムについても，Standard ML 言語における通常の関数と同様の取り扱いができるようにすることを目指す．

これにより，開発者は，型安全性の下で，効率的な並列演算を利用することが可能となる．

３．研究の方法

二つの手法により研究を進めた．研究開始後の状況から，SIMD 演算よりも， GPGPU に向けた処理の拡張のほうが今後，社会的ニーズが高いと考えられたため，特に，(2)に注力して研究を進めた．

これらを実装するにあたっては，必要に応じて Standard ML の型を拡張して行った．実装は，東北大学電気通信研究所大堀研究室で開発が進められている，SML#処理系に対して必要な拡張を施すことにより行った．SML#処理系は，C 言語などで記述された関数を直接呼び出す強力な機構を持っているため，本研究の基礎に適している．

(1) Standard ML への SIMD 演算に必要なデータ型と操作の導入
現在広く用いられている汎用プロセッサでは，2〜4 要素のベクトルデータに対して，同時に同じ演算実行する命令・操作が実装されている．また，GPGPU 上でも同様のデータ型に対して演算を行うための操作が準備されている．
そこで，このような機能を利用するのに必要なベクトルデータ型で Standard ML 言語を拡張し，そのデータ型に対する加算・減算などを行うプリミティブを，Standard ML 言語に導入した．実装は，インテル社の汎用プロセッサを対象として行った．

(2) Standard ML への CUDA の導入
GPGPU 環境，特に NVidia 社の CUDA 環境を用いて，Standard ML 言語で記述されたプログラムで並列計算を駆動できるようにすることを目指して，以下を行った．

① SML#用 CUDA ライブラリの作成
CUDA を駆動するのに必要なライブラリを作成した．CUDA では，GPU 上に別途用意したメモリにデータとプログラムを転送し，それを駆動することで演算を行う．これらに必要な API やデータ型を，Standard ML から使用できるようにした．

② SML#の CUDA プログラム埋め込みのための拡張
CUDA C で記述された，GPU 用のプログラムを Standard ML 言語に埋め込み，Standard ML の関数として呼び出せるようにするための拡張を SML#処理系に施した．以下のようにして実現した．

- プログラムを埋め込むための文法の追加
  “_cudacfun()”という文法を追加した．括弧内にリテラルの文字列としてCUDA Cを埋め込めるようにした．
- 埋め込まれたCUDA Cプログラムからのネイティブコード生成機構の実装
  コンパイラを拡張し，前項の文法で埋め込まれたCUDA Cコードに対し，内部でCUDA Cコンパイラを呼び出してライブラリを生成する機構を導入した．
- 埋め込まれたCUDA Cプログラムからの SML#用の型およびコード生成
  前項で生成したライブラリを用いて並列計算を駆動するための，SML#用の内部コードを生成する機構を導入した．この機構は，CUDA Cの文法で拡張した(Standard MLで記述された)C言語パーサを使用して型などを取りだし，それに基づいてSML#用の型安全なスタブコードを生成する．

これらを実装するにあたっては，必要に応じてStandard MLの型を拡張して行った．実装は，東北大学電気通信研究所大堀研究室で開発が進められている，SML#処理系に対して必要な拡張を施すことにより行った．SML#処理系は，C言語などで記述された関数を直接呼び出す強力な機構を持っているため，本研究の基礎に適している．

４．研究成果

（主な成果）

Standard ML 言語の中から，安全にGPGPUを用いた並列処理を行えるようになった．現状では，実験的な実装であり，型に制約があったり，GPU用のプログラムはCUDA Cで記述する必要が依然としてあったりするが，CUDA Cプログラムについても，SML#コンパイラの中で統合的に扱っているので，今後，Standard MLに近い文法のDSL (Domain Specific Language)を設計して置き換えるなどのことも可能である．

具体的には，以下のことを実現した．

(1) Standard MLへのSIMD演算に必要なデータ型と操作の導入
SML#処理系に対し，int3(3要素の整数ベクタ)型などの型で拡張を行った．併せて，これらの型のデータ同士の演算を行うプリミティブを実装した．これにより，2〜4要素のベクトル型に対するSIMD演算が安全に行えるようになった．

(2) Standard MLへのCUDAによる並列演算機構の導入
プログラムに埋め込まれた並列計算用のプログラムを，Standard ML言語の中で他のStandard ML関数と同様に，安全に使用できるようになり，静的型付き関数型言語からの安全なGPGPUにおける並列処理の駆動が行えるようになった．

①Standard ML向けのCUDAライブラリの実装
SML#処理系で利用可能な，CUDAライブラリを実装した．次項でのCUDA C言語の埋め込みのためには，必要なライブラリである．
このライブラリは，CUDA環境で提供される駆動用のライブラリを内部で使用して，主要なAPIについて，関数型言語から使用しやすいインターフェースにした上で，同様の機能を提供している．
GPU上のメモリの割り当てやデータの転送，並列計算の起動などを，Standard ML言語から行うことができる．

②Standard MLにCUDA Cプログラムを安全に埋め込むための拡張の実装
CUDA Cで記述されたプログラムをStandard MLの中に埋め込み，Standard MLの関数のようにして並列計算を駆動できる拡張をSML#処理系に施した．CUDA Cで記述されたプログラムは，SML#コンパイラによって解析され，SML#の型が生成される．あわせて，埋め込まれたCUDA Cのプログラムから，必要なネイティブコードと，呼び出しのためのスタブコードが生成される．この一連の機構により，Standard MLからシームレスにGPGPUによる並列演算を駆動できるようになった．

（位置づけ）

この研究を開始した当初に比べ，並列計算においては，GPGPUによる並列計算クラスタが広く主流として用いられるようになっており，主要なスパコンの多くをしめるようになっている．このようなスパコンでの開発生産性を高めるための基盤となる研究である．

(今後の展望)

今回の研究で実装した拡張は，現状では実験的な実装であり，それぞれ個別に実装しているため，研究期間終了後にリリースされたSML#処理系の正式版に今後移植していく予定である．また，より Standard ML の文法で記述できるようにするための CUDA 用 DSL 設計への取り組みや，メモリ管理の自動化なども必要である．

前述のように，GPGPU クラスタの普及は進んでいるので，このような環境への拡張も考える必要がある．このような環境で計算性能を向上するためには，クラスタ内でのメモリモデルを考慮することが重要である．今回の研究で対象とした単一ホスト上でのメモリモデル，すなわち，汎用 CPU 向けのメモリ空間が 1 つと，GPGPU 向けのメモリ空間の間のデータ転送を考慮するだけではなく，クラスタを構成するノード間についても，考慮しなければならない．

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）

〔学会発表〕（計　1 件）

① 吉田信明，Standard ML への GPGPU 関数の埋め込み手法，2012 年電子情報通信学会総合大会，2012 年 3 月 22 日，岡山大学．

６．研究組織

(1)研究代表者

吉田　信明（YOSHIDA NOBUAKI）

財団法人京都高度技術研究所　研究部

副主任研究員

研究者番号：00373506

(2)研究分担者

なし

(3)連携研究者

なし