様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　　８　日現在

機関番号：　１２２０１
研究種目：　基盤研究(C)（一般）
研究期間：　2013～2016
課題番号：　２５３３００５５

研究課題名（和文）CPU/GPU混載プロセッサのためのソースレベル自動並列化システムの研究開発

研究課題名（英文）Source-level parallelization system for CPU/GPU combined heterogeneous architecture

研究代表者

馬場　敬信（Baba, Takanobu）

宇都宮大学・オプティクス教育研究センター・教授

研究者番号：７００９２６１６

交付決定額（研究期間全体）：（直接経費）　　3,800,000円

研究成果の概要（和文）：本研究は、マルチコアCPUとGPUを内蔵する非均質なアーキテクチャの普及を見据えて、それぞれ特徴的なアーキテクチャを活用してプログラムの高性能化を行うための自動並列化システムの実現を目指すものである。このため、ユーザには逐次的なプログラム記述を可能としつつ、実行時の履歴情報を活用してソースレベルでの自動並列化を目標として研究を行った。本研究における成果としては、Valgrindを用いて実行パス、データ依存、メモリアクセスのプロファイリングシステムを構築すると共に、これをベンチマークプログラムに適用して、各プログラムの実行特性に応じた並列化の方法を決定する手順を実験的に明らかにしたことである。

研究成果の概要（英文）：The increase of transistor count allows us to implement multicore-CPU and GPU combined heterogeneous architecture on a single chip. We have designed a source-level loop parallelization system framework for the heterogeneous architecture. Following the designed framework, we have developed a path and data-dependency profiling tool using Valgrind. By applying the system to benchmark programs, we have clarified the characteristics of each loop of the programs and showed that we can utilize the profiled results to determine the parallelizing scheme on heterogeneous architecture

研究分野：　計算機システム

キーワード：　CPU/GPU混載プロセッサ　実行プロファイリング　自動並列化

１．研究開始当初の背景

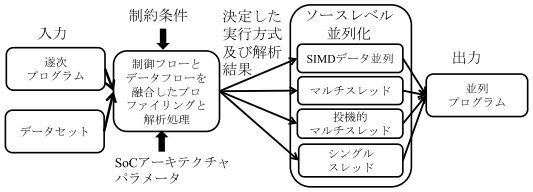
よく知られたムーアの法則に従って１つのチップ上のトランジスタ数は増大の一途をたどっており、今や１チップ上に 10^9 規模のトランジスタが搭載される時代となっている。この多数のトランジスタを活用して、1チップ上には複数の CPU コアをもったマルチコア CPU が実装されると共に、さらにグラフィックス処理の高速化を目的に開発された GPU(Graphics Processing Unit)が共存するヘテロジニアス（非均質）アーキテクチャが一般的になりつつある。

このようなヘテロジニアスアーキテクチャの性能を十分に発揮するには、マルチコア CPU と GPU それぞれのアーキテクチャ上の特徴を活用した並列化が重要な鍵を握る。即ち、マルチコア CPU は、複数のタスクを同時に実行する所謂タスク並列を前提とした機能が強化されているのに対し、GPU では多数の依存関係のないデータを並列処理するデータ並列を前提とした機能が強化されているという大きな違いがある。

多くのユーザにとっては、並列プログラムを作成すること自体が大きな障壁であるだけでなく、さらにヘテロジニアスアーキテクチャのもつこのような特徴を考慮してプログラムを作成することは極めて困難な作業となる。このため、ユーザには使い慣れた逐次的なプログラミング言語によるプログラム作成を可能としつつ、そのプログラムの部分部分の特性に応じたアーキテクチャの選択と選択したアーキテクチャを対象とした並列化を自動的に行うシステムの実現が切望される。

２．研究の目的

このような背景のもとに、本研究においては、**図１**に示すように、実行プロファイルに基づいてホットループを検出し、さらに検出したホットループの実行パス、実行パス間のデータ依存、及びメモリアクセスパターンなどのプロファイル情報を求めることにより、制御フローとデータフローを把握し、対象ループごとに適したアーキテクチャを自動的に選択し、さらに選択したアーキテクチャ上での並列化をソースプログラムレベルで自動的に行う処理系を研究開発することを研究目的とする。



マルチコアCPU/GPUを搭載したSoCを前提として、プロファイリング結果を解析することにより最適な実行方式を決定し、さらに解析結果を用いて、ソースレベルで並列化を行う

図１ ソースレベル自動並列化システムの概念

３．研究の方法

上記の研究目的に沿って、本研究においては、以下の研究項目に沿って研究を実施した。

(1) CPU/GPU混載システムにおける並列化の課題抽出

マルチコア CPU と GPU のそれぞれについて、並列化に際しての課題を抽出する。特に GPU 上での並列化については、並列実行されるスレッド間のデータ依存関係やメモリアクセスの高速化等、GPU アーキテクチャに固有の最適化が求められるため、小規模のベンチマークプログラムに留まらず、CGH(Computer Generated Holography)計算などの大規模の実アプリプログラムを対象に実験的に GPU での並列化を行って課題抽出に努めた。

(2) 並列化に際し必要となるプロファイル情報の特定

(1)で抽出した課題を念頭に、これに対処して並列化を行うのに必要なプロファイル情報の特定を行った。従来より我々は実行パス（以下単にパス）に着目したプロファイル情報の収集とその解析についての研究を重ねており、本研究でもパスを制御のプロファイリングの基本とし、さらにパス間のデータ依存関係をデータ依存プロファイリングの基本とすることとした。

(3) Valgrind を用いたプロファイリングシステムの設計及び試作

プログラムを再コンパイルすることなく実行プロファイルを収集するため、動的バイナリ変換を行うこととし、そのためのツールとして Valgrind を用いた。Valgrind は主要なプロセッサの命令セットに対応するとともに、プラグインツールによってユーザが独自のプロファイリング機能を付加することが可能である。これに(2)で定めたプロファイリング情報の取得機能を装備することでシステムの試作を行うこととした。

(4) 試作したプロファイリングシステム実験的な適用とその結果検証、及びシステム改善

試作したプロファイリングシステムを、各種のベンチマークプログラムに実験的に適用して、各ベンチマークプログラムのホットループを検出、検出したホットループのパス情報、パス間のデータ依存関係等を抽出した。さらに、抽出された情報に基づいて、対象とするホットループが、マルチコアによる並列化、GPU による並列化、あるいは逐次処理のいずれに適合するのかを判定する方法を検討した。また、この判定結

果の妥当性について検討した。もし、試作したプロファイラの出力する情報だけでは十分な判定ができないと判断された場合は、どのような情報があればより正確な判定が可能なのか、を見極めて、その取得に必要な機能をプロファイリングシステムに補って改善を行うこととした。

４．研究成果

本研究の成果は、CPU/GPU 混載システムにおける自動並列化を目標として、以下のように整理することができる。

(1) パスベースプロファイリングシステムの試作

Valgrind をベースにプロファイリングシステムを構築し、ループを対象に、パスプロファイリングとパス間のデータ依存関係の抽出が行える機能を実装するとともに、正しく動作することを確認した。データ依存関係については、レジスタ変数だけでなくメモリ変数も扱えるようにしている。

一例として、N クイーン問題を解くプログラムのメインループのアセンブリコードを**図２**に、そのループを対象としたプロファイル結果を**図**３に示す。図３に示す通り、プロファイルには、ループ内のパスの経路（基本ブロックの系列）と経路毎の実行回数、パスに含まれる命令数、ループ内のパスの総実行回数、及びパス間のデータ依存関係などが含まれている。

(2) 試作したプロファイリングシステムの実験的な適用とその改善

試作したプロファイリングシステムをベンチマークプログラムに適用すると共に、ループの性能改善のために欠かせないプロファイリング機能を明らかにして、システムの改善を行った。

具体的には、NAS 並列ベンチマークプログラムから CG, MG, EP, SP を選んでプロファイラを適用した。この結果の一例を図４に示す。これらのプログラムのホットな最内ループにおいて実行されるパス数は 1-3 本と少なく、同時にパス間の依存関係もあまりないため、基本的にはデータ並列処理とみなすことができる。しかし、同時にパスの命令数が少ないため、並列処理のオーバヘッドが大きくなる傾向がある。従って、データ並列処理であるから GPU が適合、と単純に決定することはできず、並列処理のオーバヘッドを補う十分な処理の粒度が必要であること、データ依存関係だけでなくメモリのアクセスパターン等のプロファイルが必要であるとの判断に至った。

```
400a6f: 41 8b 06              mov   (%r14),%eax
400a72: 85 c0                 test  %eax,%eax
400a74: 74 59                 je    400acf <try+0x5ff>

400a76: 48 63 bc 24 c4 00 00  movslq 0xc4(%rsp),%rdi
400a7d: 00
400a7e: 83 3c bd c0 17 60 00  cmpl  $0x0,0x6017c0(,%rdi,4)
400a85: 00
400a86: 48 89 bc 24 e0 00 00  mov   %rdi,0xe0(%rsp)
400a8d: 00
400a8e: 74 3f                 je    400acf <try+0x5ff>

400a90: 48 63 84 24 c0 00 00  movslq 0xc0(%rsp),%rax
400a97: 00
400a98: 83 3c 85 00 18 60 00  cmpl  $0x0,0x601800(,%rax,4)
400a9f: 00
400aa0: 48 89 84 24 d8 00 00  mov   %rax,0xd8(%rsp)
400aa7: 00
400aa8: 74 25                 je    400acf <try+0x5ff>

400aaa: 48 63 44 24 54        movslq 0x54(%rsp),%rax
400aaf: 83 7c 24 54 07        cmpl  $0x7,0x54(%rsp)
400ab4: 8b 94 24 b8 00 00 00  mov   0xb8(%rsp),%edx
400abb: 89 14 85 80 17 60 00  mov   %edx,0x601780(,%rax,4)
400ac2: 0f 85 82 00 00 00     jne   400b4a <try+0x67a>

400ac8: 83 05 a1 0c 20 00 01  addl  $0x1,0x200ca1(%rip)      # 601770 <solution>

400acf: 83 84 24 b8 00 00 00  addl  $0x1,0xb8(%rsp)
400ad6: 01
400ad7: 49 83 c6 04           add   $0x4,%r14
400adb: 83 84 24 c4 00 00 00  addl  $0x1,0xc4(%rsp)
400ae2: 01
400ae3: 83 ac 24 c0 00 00 00  subl  $0x1,0xc0(%rsp)
400aea: 01
400aeb: 83 bc 24 b8 00 00 00  cmpl  $0x8,0xb8(%rsp)
400af2: 08
400af3: 0f 85 76 ff ff ff     jne   400a6f <try+0x59f>
```

図２ N クイーン問題メインループのアセンブリコード

```
Path0
400a6f 400acf
Count = 2388, Instruction = 9
Path1
400a6f 400a76 400a90 400acf
Count = 222, Instruction = 17
Path2
400a6f 400a76 400acf
Count = 420, Instruction = 13
Total Count = 3030
Path0 → Path0
r14 Write = 400ad7 ( 4) → Read = 400a6f ( 0)
Path0 → Path1
r14 Write = 400ad7 ( 4) → Read = 400a6f ( 0)
Path0 → Path2
r14 Write = 400ad7 ( 4) → Read = 400a6f ( 0)
Path1 → Path0
r14 Write = 400ad7 (12) → Read = 400a6f ( 0)
Path1 → Path1
r14 Write = 400ad7 (12) → Read = 400a6f ( 0)
Path1 → Path2
r14 Write = 400ad7 (12) → Read = 400a6f ( 0)
Path2 → Path0
r14 Write = 400ad7 ( 8) → Read = 400a6f ( 0)
Path2 → Path1
r14 Write = 400ad7 ( 8) → Read = 400a6f ( 0)
Path2 → Path2
r14 Write = 400ad7 ( 8) → Read = 400a6f ( 0)
```

図３　プロファイル結果

```
Loop #1
Start Address = 400797
Path0
Block Seq. = 400797
Count = 730041600
# of Inst. = 8
# of Iter. Per Loop = 292
Total Count = 730041600
Reduction
Reg = xmm0  Inst. Addr = 4007aa

Dependency
Path0 → Path0
rax   Write = 4007ae (  5) → Read = 400797 (  0) // Induction
Overlap Inst. = 8
```

図４ CG 中のループのプロファイル結果

これを受けて、(i)メモリアクセスパターンをプロファイル可能とすること、(ii) 当初最内ループに限定して行っていたプロファイリングの対象を外側ループにまで拡げ

ること、の２点についてシステムの改善を行った。特に(ii)に関しては新たにループブロックの概念を導入して、ループを１まとまりのブロックとみなすことにより任意の多重ループをプロファイル可能としている。

図５に、CG プログラム中でプロファイル対象とした２重ループのソースコードを示す。また、図６に、改善後のプロファイラを用いて、外側ループまでプロファイルを行った結果を示す。loop1 と loop2 がそれぞれ内側と外側の 2 つの for ループのプロファイル結果を示す。loop２のプロファイル結果に示すように、40121f で始まる内側（最内）ループが一つの基本ブロックとして扱われていることが分かる。また、ループ突入後の繰り返し実行回数が、内側ループで 125 回、外側ループで 1398 回であることが分かる。これによって、内側ループだけでは十分な繰り返し回数がなく並列化による性能向上が見込めない場合でも、外側ループまで対象とすることで大幅な性能向上が期待できることが分かる。

```
for (j = 1; j<= lastrow-firstrow+1; j++) {
  sum = 0.0;
  for (k = rowstr[j]; k < rowstr[j+1]; k++) {
    sum = sum + a[k]*p[colidx[k]];
  }
  w[j] = sum;
}
```

図５ CG プログラム中のプロファイル対象ループ（ソースコード）

```
**Loop Number 1**
--loop 1--
Loop's start address         = 4012ef
Number of Iteration Per Loop = 125
Execution Count              = 3767400
------------Path0------------
Block Seq. = 4012ef
Number of Execution   = 3767400
Number of Instruction = 31
-----------------------------
Dependency -- mem --
Path0 -> Path0
ffefffda8 Write = 40134a ( 22) -> Read = 4012ef (  1)
   8d6080 Write = 401342 ( 21) -> Read = 401336 ( 19)
Dependency -- reg --
Path0 -> Path0
 No
--loop2--
Loop's start address         = 4012c8
Number of Iteration Per Loop = 1398
Execution Count              = 69899
------------Path0------------
Block Seq. = 4012c8-4012ef-(lb:4012ef)-4012ef-40136d
Number of Execution   = 69899
Number of Instruction = 3978
-----------------------------
Dependency -- mem --
Path0 -> Path0
ffefffdac Write = 4013a9 (3971) -> Read = 4012d4 (  3)
Dependency -- reg --
Path0 -> Path0
 No
***************
```

図６　CG プログラム中の２重ループのプロファイル結果

当初の研究目的から見ると、ここまでの成果は、プロファイリング機能を中心とするものである。このため、今後の課題としては、プロファイリング結果を活用した CPU/GPU 混載システムにおける最適な並列実行方式の決定法、および決定された実行方式に即したソースレベル並列化の実現がある。

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 1 件）

1. Yuan-ming Zhang, Xiao Gang, Takanobu Baba, "Accelerating sequential programs on commodity multi-core processors," Journal of Parallel and Distributed Computing, Elsevier, Vol.74, Issue 4, pp.2257-2265, Apr. 2014. DOI 10.1016/j.jpdc.2013.12.009

〔学会発表〕（計 11 件）

1. Takanobu Baba, Kanemitsu Ootsu, ″Two-Level Controlled Parallel Reconfigurable Architecture,″ First Workshop on Pioneering Processor Paradigms (WP3), 4 February, 2017 / Austin, TX, USA http://wp3workshop.website/

2. 菊池 祐貴, 大津 金光, 馬場 敬信, 横田 隆史, 大川 猛, “多重ループの動的挙動解析のためのループブロックを導入したパスプロファイラの実現”, 信学技報, CPSY2016-121, pp.103-108, 2017 年 1 月.

3. Takanobu Baba, Boaz Jessie Jackin, Shinpei Watanabe, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota, Yoshio Hayasaki, Toyohiko Yatagai, ″Object decomposition method for acceleration of large-scale hologram calculations on GPU-clusters,″ Accepted for ICCP Poster, IEEE International Conference on Computational Photography (ICCP 2016), May 13-15, Northwestern University, Evanston, IL.

4. 菊池 祐貴, 大津 金光, 馬場 敬信, 大川 猛, 横田 隆史, “ヘテロジニアスメニーコアプロセッサにおける最適並列処理の決定方式に関する検討”情報処理学会 第 78 回全国大会 講演論文集, pp.1-111～1-112, (2016.03.12).

5. Kanemitsu Ootsu, Yutaka Matsuno,

Takeshi Ohkawa, Takashi Yokota, Takanobu Baba, "Empirical Performance Study of Speculative Parallel Processing on Commercial Multi-core CPU with Hardware Transactional Memory," Second Workshop on Software Engineering for Parallel Systems (SEPS), pp.57-65, Oct.

6. Takanobu Baba, Kazuki Ohshima, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota, "Consideration of Loop Parallelization on Heterogeneous Multicore Architecture Using Path and Data Dependence Profiling," The First Workshop on Software Engineering for Parallel Systems (SEPS), Mon 20 - Fri 24 October 2014, Portland, Oregon, United States.

7. 大島 一輝, 大津 金光, 馬場 敬信, 大川 猛, 横田 隆史, “プログラム実行パス間のデータ依存を解析するためのパスプロファイラの実現”, 信学技報, Vol.114, No.155, pp.203-208 (CPSY2014-44), 2014 年 7 月 30 日.

8. 大島 一輝, 馬場 敬信, 大津 金光, 大川 猛, 横田 隆史, “Valgrind を用いたループイテレーション間依存を検出するプロファイラの開発”, 情報処理学会 第 76 回全国大会講演論文集, pp.1-99～1-100, 2014 年 3 月 12 日.

9. Takanobu Baba, Hiroaki Miyata, Boaz Jessie Jackin, Takeshi Ohkawa, Kanemitsu Ootsu, Takashi Yokota, Yoshio Hayasaki, Toyohiko Yatagai, "Interpolation-Based Object Decomposition and Parallel Computation Method for Large-Scale Computer-Generated Hologram," Proc. 12th IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2014), pp.200-207, Feb. 2014. DOI: 10.2316/P.2014.811-020

10. Hiroyoshi Jutori, Takanobu Baba, Kanemitsu Ootsu, Takeshi Ohkawa, Takashi Yokota, "Exploration of Highly Accurate Path Prediction Mechanism using Detailed Path History," Proc. 1st International Symposium on Computing and Networking — Across Practical Development and Theoretical Research — (CANDAR'13), pp.582-586, Dec. 2013. (WANC'13 short paper) DOI: 10.1109/CANDAR.2013.104

11. 金海 和宏, 大津 金光, 大川 猛, 横田 隆史, 馬場 敬信, “ループにおけるパス予測と分岐予測の関連性についての考察”, 信学技報, Vol.113, No.169, pp.115-120 (CPSY2013-29), 2013 年 8 月 2 日.

〔図書〕（計 1 件）

1. 馬場敬信： コンピュータアーキテクチャ（改訂 4 版）, オーム社、p. 420(2016)

６．研究組織

(1)研究代表者

馬場 敬信（BABA Takanobu）
宇都宮大学・オプティクス教育研究センター・教授
研究者番号：70092616

(2)研究分担者

大津 金光（OOTSU Kanemitsu）
宇都宮大学・大学院工学研究科・准教授
研究者番号： 00292574

(3)連携研究者

横田 隆史（YOKOTA Takashi）
宇都宮大学・大学院工学研究科・教授
研究者番号： 90334078

大川 猛 （OHKAWA Takeshi）
宇都宮大学・大学院工学研究科・助教
研究者番号： 80392596