計算システム工学分野

低消費電力化のための並列処理タスクへのプロセッシングエレメントの動的な配分機構

学籍番号 23413531 氏名　榊原　宏章

指導教員名　岩田　彰

1　はじめに

アプリケーションはプロセッサ抽象化 API を用いることにより，端末に搭載されている複数のプロセッサのリソースを細やかに指定して使用できる．Windows 上でのパフォーマンス向上のためにバイナリに変更を加えずに使用プロセッサの動的な変更を可能とする研究として PACUE[1]がある．プロセッサの中の計算ユニットの単位であるプロセッシングエレメントをタスクに適切に割り当てることで省電力化が期待できる．タスクがリソースを適切に使用するには，まず，リソースの動的な配分を可能とする環境の構築が必要となる．本論文では，Linux 系 OS において，バイナリに変更を加えることなく，プロセッサ抽象化 API 利用アプリケーションが使用するプロセッシングエレメントを動的に変更できるリソース配分機構の実現方式を提案し，提案方式の効果を確認する．

2　関連研究

PACUE は，API フックによりプロセッサ抽象化 API への介入を行うことによって，バイナリに変更を加えずに方式の利用を可能としている．PACUE は Windows が搭載された PC を対象としているが，消費電力や占有面積への制限という観点から携帯端末でのリソースの有効活用に対する需要が高いことから，Linux 系 OS での実現を図る．また，多くのプロセッサ抽象化 API では，PE へのタスクの割り当てを指定することができる．これを利用して，動的に PE の配分を行うことでプロセッサの選択をするだけよりも個々の端末性能を活かした処理を行うことが期待できる．そのためには，まず，リソースの動的な配分を可能とする環境の構築が必要となる．

3　提案方式とその実装

提案方式の概要を図 1 に示す．提案方式は PE へのタスクの割当をタスクへの PE の配分と捉える．提案方式は PE 割当機構と PE 選択機構の 2 つの機構を持つ．PE 割当機構は，API フックによりプロセッサ抽象化 API の呼出へ介入し，タスクが割り当てられる PE を変更した上で同じ API を呼び出すことにより，PE への割当の変更を行う．PE 選択機構は，取得した端末情報や実行情報などの利用状況データを元に，タスクへ割り当てられる PE の選択を行う．PE の動的な配分は，PE 選択機構の結果に従って，タスクが割り当てられる PE を PE 割当機構が変更することによって実現する．また，提案方式は PACUE の構成を継承しているため，使用プロセッサの動的な変更も可能である．

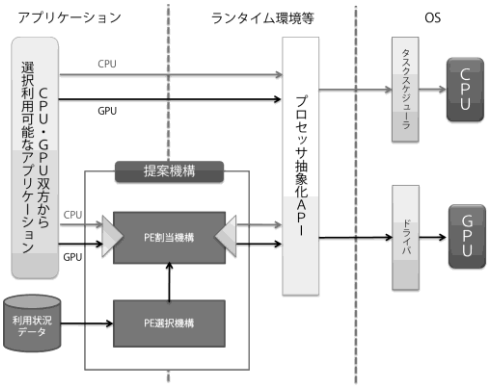

図 1：提案方式の概要

提案方式を Linux および Android において，OpenCL[2]を用いて実装した．実装の環境を表1に示す．本実装では，並列化したアルゴリズムの持つループのネスト数がデータの次元数×2である場合に，タスクが割り当てられるPEを変更すると正しい計算を行えないため，変更を行わない．

4　評価

まず，互換性の確認のため，文献やOpenCLフレームワーク実装ベンダが提供しているサンプルコードを用いて，動作検証を行った．いずれのコードに

表 1：実装の環境

	Android	Linux
本体型番	Sony Erricsson Xperia P(LT22i)	HP Folio 13 -1009TU
CPU	1 GHz Dual-core ARM Contex-A9	Intel Core i5 -2467M
チップセット	ST-Erricsson NavaThor u8500	Intel HM65 Express
OS	Android 2.3.7	Ubuntu 10.04 LTS 32bit
実行環境およびコンパイラ	PGCL 12.8	AMD APP SDK 2.7

ついても，提案方式によるタスクが割り当てられるPEの変更が可能であった.

次に，提案方式を導入した際の消費電力の変化を確認した．提案方式を用いずプロセッサ抽象化 API を利用して処理を行った場合と，提案方式を用いてタスクが割り当てられる PE の変更を行った場合について，電力が継続して消費される連続した処理を行うアプリケーションを想定した評価用コードにより消費電力の比較を行った.

評価用コードは，ループのネスト数がデータの次元数である FFT を並列化したコード(コードA)と，ループのネスト数がデータの次元数×2 である DCT を並列化した, 提案方式による PE の動的な配分が行えないコード(コード B)を用いた．コード A では 512×512 [pixel]の画像, コード B では 64×64 [pixel]の画像を入力として評価用コードを実行する.

評価用コードの処理量の目安としてデータ入力速度を考え, 一定の FPS(Frame Per Second)で画像データを評価用コードに入力し，消費電力を調べる．計測は，バッテリを 100%まで充電した状態から 5 分間スリープさせたのちに評価用コードを 10 分間実行し，3 秒ごとに消費電力を計測する．FPS に対して処理が間に合わない場合には計測を中止する．Linux カーネルが提供している power supply class を利用して，端末全体の消費電力を計測する.

タスクが割り当てられる PE の初期値は使用するサンプルコードの初期値を用いた．また，タスクが変更される PE の選択は，予め評価環境におけるサンプルコードの値より高速な値を調査して用いた.

結果を図 2，図 3 に示す．横軸が FPS，縦軸が実行中に計測した消費電力の平均である．表中の消費電力は，評価用コードを実行したときに計測した値から，評価用コードを実行していない検証環境の消費電力を引いた値であり，評価用コードを実行したことによる増加分を表す. Linux の結果の単位は W, Android の結果の単位は VA である．以降，提案方式を利用しない場合をデフォルト状態と呼ぶ.

コード A においては，提案方式を用いることで PE に割り当てる処理量が調節され，全体として消費電力が低減できた. Linux では最大で約 69.07%, Android では最大で約 49.16%削減された．また，コード B においては， 提案方式を用いることで消

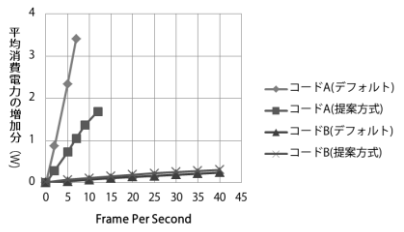

図 2：データ入力速度に対する平均消費電力（Linux）

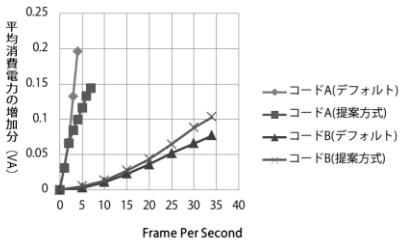

図 3：データ入力速度に対する平均消費電力（Android）

費電力が増加しているが，コード A での消費電力の削減量に対して，コード B での消費電力の増加量は比較的小さいことがわかった.

5 まとめ

本論文では，Linux 系 OS において，バイナリに変更を加えることなく, プロセッサ抽象化 API 利用アプリケーションが使用する PE を動的に変更できるリソース配分機構の実現方式を提案した．Linux および Android 上で OpenCL を用いて実装し，提案方式による PE の割当の変更の効果を確認した.

提案方式は新たなランタイム環境の導入やアプリケーションの改変が不要で利用可能であること，提案方式はネスト数がデータの次元数であるループを並列化したコードにおいて省電力効果があることがわかった.

今後の課題としては，一般的な利用状況においても省電力化ができるように，PE の配分に適したアルゴリズムが含まれているかを常駐アプリケーションで識別することと，PE の配分を決定するアルゴリズムの検討を行いたいと考えている.

6 参考文献

[1] Tetsuro Horikawa, Michio Honda, Jin Nakazawa, Kazunori Takashio and Hideyuki Tokuda, ”PACUE: Processor Allocator Considering User Experience”, Euro-Par 2011, Vol. 7156, pp. 335-344, Aug-Sep, 2012.

[2] Khronos group, “OpenCL - The open standard for parallel programming of heterogeneous systems”, http://www.khronos.org/opencl/