携帯端末における仮想機械での CPU 周波数抑制による消費電力の削減

橘田 頼之[1,a)] 鵜川 始陽[1,b)] 岩崎 英哉[1,c)]

概要：スマートフォンやタブレットなどのバッテリー駆動の携帯端末において，アプリケーション実行における消費電力を削減することは重要である．携帯端末に搭載される CPU の多くは動作周波数（以下，周波数）を変更することができる．周波数を下げれば消費電力を抑えることができる反面，アプリケーションの実行速度も遅くなるため，消費電力と実行速度はトレードオフの関係にある．我々は，メインメモリへのランダムアクセスの多い処理は，キャッシュが有効に働かないため，周波数を下げて実行しても実行速度に与える影響は大きくないという事実を利用し，アプリケーションの実行中に，実行速度の低下を抑えつつ消費電力を削減する手法を提案する．携帯端末ではごみ集め（以下 GC）の機構を持つ Java などの仮想機械上でアプリケーションが動作する．GC の処理では，計算の割合は少なくメインメモリへのランダムアクセスが主である．提案手法はこの性質を利用し，GC のみ周波数を下げて実行する．実装は，ARM 搭載のワンボードコンピュータ Pandaboard 上の Android 4.2.2 を対象として行った．この環境の下で，Java のメモリ管理のベンチマーク集である DaCapo の中から 6 つのベンチマークを実行した．その結果，6 つのうち 2 つのベンチマークについては提案手法の効果が確認できた．効果のあったベンチマークには，メモリ使用量が比較的大きいという性質があった．

キーワード：DVFS，消費電力，携帯端末，ごみ集め，Android

Reducing Energy Consumption in Virtual Machine for Mobile Devices by Lowering CPU Frequency

Yoriyuki Kitta[1,a)] Tomoharu Ugawa[1,b)] Hideya Iwasaki[1,c)]

Abstract: For mobile devices with batteries such as smartphones or tablets, reducing energy consumption by executions of applications is important. Most of CPUs on mobile devices are able to change the CPU frequency. The lower the CPU frequency is, the less the energy consumption. However, application's running speed also slows down. Thus, there is a trade-off between energy consumption and running speed. Memory bound tasks runs with less speed down by making CPU frequency low than CPU bound tasks. On the basis of this observation, this paper presents a method for reducing energy consumed by executions of applications with a small speed reduction. In mobile devices, applications are running on a virtual machine, such as a Java virtual machine (JVM). JVM has the garbage collection (GC) mechanism, which reclaims memory areas that is no longer used. GC is not a CPU bound tasks but a memory bound tasks. Thus, our method forces the CPU frequency to be low during GC is running. We used Android 4.2.2 on Pandaboard, a one-board computer with on ARM processor, as implementation target. We ran six benchmark programs in the DaCapo Benchmark Suite, which is a benchmark suite for Java memory management. As a result of our measurement, two benchmark programs that uses a lot of memory, showed good effects of proposed method.

Keywords: DVFS, Energy Consumption, Mobile Devices, Garbage Collection, Android

[1] 電気通信大学大学院情報理工学研究科情報・通信工学専攻
Department of Communication Engineering and Informatics, Graduate School of Informatics and Engineering, The University of Electro-Communications
[a)] ykitta@ipl.cs.uec.ac.jp
[b)] ugawa@cs.uec.ac.jp
[c)] iwasaki@cs.uec.ac.jp

1. はじめに

近年，スマートフォンやタブレットといった携帯端末が普及してきている．これらはバッテリー駆動であるため，アプリケーション実行において消費電力を削減することは重要である．また，携帯端末で複雑な処理を行うことも多くなってきており，消費電力の削減と共にアプリケーション実行速度も要求される．

消費電力の削減手法の一つとして DVFS（Dynamic Voltage and Frequency Scaling）がある．DVFS は，CPU の電圧と周波数を動的に制御する技術で，近年の携帯端末に搭載される CPU の多くに実装されている．これを用いて，電圧と周波数を下げて実行することにより，消費電力を削減することができる．その反面，アプリケーションの実行速度が低下してしまう．従って，消費電力と実行速度はトレードオフの関係にある．

CPU が計算する量が多い処理については，周波数を上げて実行すればそれだけ実行速度も向上する，しかし，メインメモリへのアクセスが多い処理の場合は．メモリアクセスに時間がかかるため，周波数を上げて実行しても実行速度はそれほど向上しない．そのため，メインメモリへのアクセスが多い処理については周波数を下げて実行した方が，CPU が計算する量が多い処理よりも実行速度の低下を抑えつつ消費電力の削減になる [1], [2], [3], [4] ．

メインメモリへのアクセスが多い処理の一つとしてごみ集め（以下 GC）がある．これは不要になったメモリ領域を解放して再利用を可能とする機構である．一般にアプリケーションの空きメモリ領域が不足すると GC を呼び，通常の処理（以下ミューテータ）を停止し，ヒープの全体にアクセスして不要なメモリ領域を解放する．GC のメモリアクセスは広範囲に不規則に行われる．そのため，キャッシュメモリの効果を受けにくく，メインメモリへのアクセスが多くなる．従って周波数を下げて実行しても実行速度に与える影響は少なくなる．

これに基づき本研究は，アプリケーションの実行中に，実行速度の低下を抑えつつ消費電力を下げる手法として，GC 時に周波数を下げて実行することを提案する．

この提案手法を Android の Dalvik VM 上に実装を行った．Android は携帯端末に多く用いられるプラットフォームであり，アプリケーションは Dalvik VM という仮想機械上で実行さる．GC はこの Dalvik VM が行う．

提案手法を実装した Dalvik VM の上で，メモリ管理のベンチマーク集である DaCapo ベンチマークのうち 6 つのベンチマークプログラムを実行し，評価を行なった．その結果，特にアプリケーションの使用するメモリ領域の大きいプログラムについて，提案手法の効果が確認できた．

以下，本稿の構成は次の通りである．2 章では，CPU における消費電力について述べ，3 章では，メモリアクセスによる消費電力への影響，4 章で Android におけるメモリ管理について述べる．5 章で提案手法について説明し，6 章で提案手法の具体的な実装と，評価について述べる．そして，7 章で関連研究について述べ，8 章で本稿をまとめる．

表 1 Pandaboard の特徴
Table 1 Characteristics of Pandaboard

CPU	OMAP 4430 (ARM アーキテクチャ)
CPU 周波数	300MHz, 600MHz, 800MHz, 1008MHz
メモリ	1GB
1 次キャッシュ	データ: 32KB, 命令: 32KB
2 次キャッシュ	1MB

2. CPU の消費電力削減

本章では，CPU の消費電力と，消費電力の削減手法について説明する．

2.1 CPU の消費電力

CPU の消費電力は，主にダイナミック電力とスタティック電力からなる [1]．ダイナミック電力は，回路のスイッチング動作により生じる消費電力である．スタティック電力は漏れ電力とも呼ばれ，スイッチング動作に関係なく消費される電力である．本研究では，ダイナミック電力を対象とする．

ダイナミック電力 P_d には，CPU の動作周波数を f，供給電圧を V としたとき，$P_d \sim V^2 f$ の関係がある．

2.2 DVFS

ダイナミック電力の削減手法の一つとして DVFS（Dynamic Voltage and Frequency Scaling）がある．これは動的に電圧と周波数を制御する技術であり，近年用いられている CPU の多くに実装されている [5], [6].

DVFS をユーザから扱うために，Linux では cpufreq という API が用意されている．cpufreq では動作ポリシーがいくつか用意されている．動作ポリシーには，負荷に応じて周波数を段階的に制御するもの，手動で周波数を制御するものなどがある．設定された周波数に応じて供給電圧も設定される．

3. メモリアクセスによる電力消費

広範囲なメモリ領域へのランダムアクセスはキャッシュミスを起こしやすい．キャッシュミスしたメモリアクセスは，メインメモリへアクセスするので，キャッシュにヒットした時に比べ大幅に時間がかかる．広範囲なメモリへのランダムアクセスは，キャッシュヒットしやすい狭い範囲へのメモリアクセスと比べ，周波数を下げて実行しても実行速度に与える影響が少ないということが知られている [1], [2], [3], [4] ．

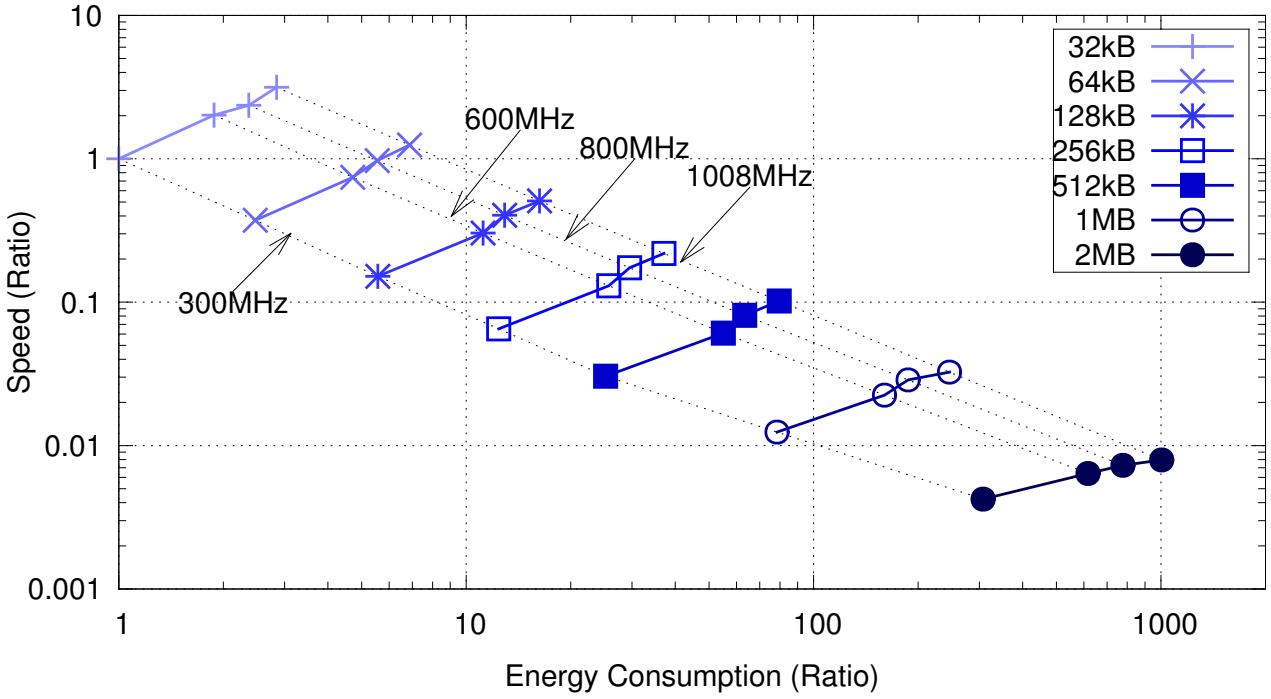


図 1 メモリアクセスによる消費電力量と実行時間の変化

Fig. 1 Energy consumption and running time according to range of memory access and frequency.

この事実を予備実験を行い検証した．予備実験では，メモリのある範囲に対してランダムアクセスを行うプログラムを用いた．アクセスする範囲の大きさを 32KB から 2MB まで変化させながら 4 段階の周波数で実行して，アクセスに要した時間と消費電力量を計測した．

実験環境としては，ワンボードコンピュータの Pandaboard [7] を用いた．Pandaboard はデュアルコア ARM プロセッサを搭載している．Pandaboard の特徴を**表 1** に示す．

図 1 に予備実験の結果を示す．図において縦軸は実行速度の比，横軸は消費電力量の比を示し，各軸とも対数である．周波数 300MHz でアクセスする範囲の大きさ 32KB の時の値が比の基準となっている．各点は周波数とアクセスする範囲の大きさを変えた時の値を示しており，同じ印の点で結ばれた線がアクセスする範囲の大きさが同じ実行の結果範囲である．

2 次キャッシュが 1MB であるので，アクセスする範囲が 2MB の時は多くのキャッシュミスが発生していると考えられる．

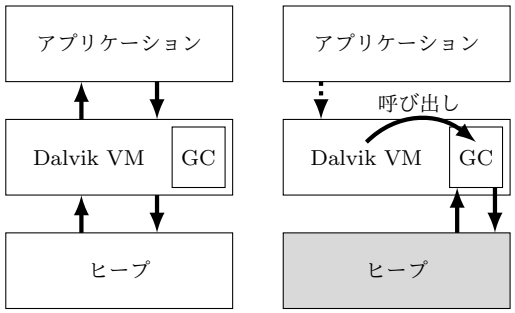
図を見ると，アクセスするメモリ領域の範囲が広くなるに従って，各点が右下にずれていっている．グラフから，周波数の変化による消費電力の最大と最小の比は大きくなり，実行速度の最大と最小の比は小さくなることが確認できる．従って，アクセスする範囲が広い時に低い周波数で実行することで，アクセスする範囲が狭いときに比べて，より少ない実行速度の低下で，より大きい消費電力の削減が出来ることが確認できる．

4. Android におけるメモリ管理

Android では，Dalvik VM という仮想機械により，Java バイトコードにコンパイルされた Java アプリケーションが解釈され実行される．アプリケーションがヒープ（動的に作られるデータのためのメモリ領域）を直接操作することはなく，**図 2(a)** のようにすべて Dalvik VM を経由してアクセスする．Dalvik VM がアプリケーションの実行と共にヒープにオブジェクトを割り付け，ヒープの空き領域が少なくなると Dalvik VM は GC を呼び出して（**図 2(b)**），不要になったメモリ領域を解放する．

ヒープの模式図を**図 3** に示す．オブジェクト同士は参照関係を持っている．ルート集合は，ヒープの外からヒープ内のメモリ領域への参照の集合であり，具体的にはスタックフレームやグローバル変数などである．ルート集合から参照関係を辿って到達できないオブジェクトをごみとして解放し回収する．図 3 では，灰色に塗られたオブジェクトがごみである．

Dalvik VM には，マークスイープというアルゴリズム



(a) アプリケーション実行時　(b) GC 実行時

図 2 Android でのヒープ操作
Fig. 2 Heap manipulation in Android

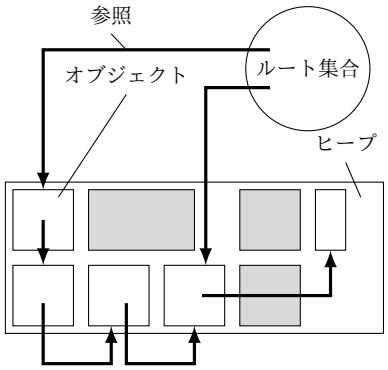


図 3 ヒープとオブジェクト
Fig. 3 Objects in heap

の GC が備わっている．マークスイープ GC は次の 2 つのフェーズにより実行される．

マークフェーズ 生きているオブジェクトにマークを付け，参照を辿る．

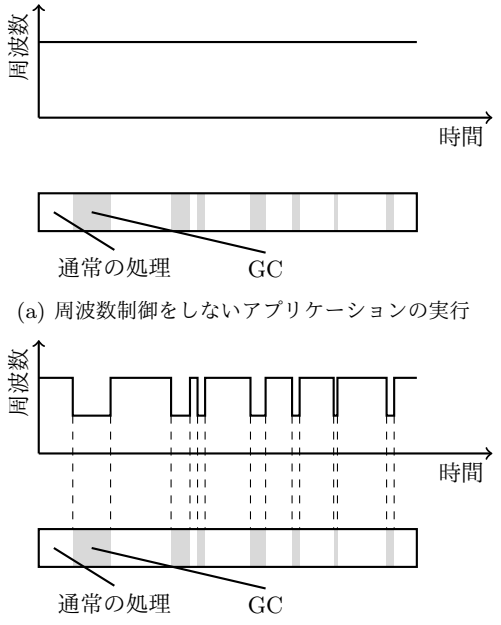
スイープフェーズ ヒープを端から見て行き，マークの付いていないオブジェクトを解放する．

マークフェーズで，オブジェクトからオブジェクトへ参照を辿るが，必ずしも近くのオブジェクトを参照しているとは限らない．従って，マークフェーズは，ランダムアクセスになる．

5. 提案手法

本研究では，GC 処理中は周波数を下げて実行することを提案する．低い周波数で GC を実行することにより，3 章で述べたように，より少ない実行速度の低下でより多くの消費電力の削減ができると期待できる．

アプリケーションの実行の流れを**図 4** に示す．通常の周波数制御を行わずにアプリケーションを実行したときの流れが**図 4(a)**，提案手法によりアプリケーションを実行したときの流れが**図 4(b)** である．図中の下側はアプリケーションの流れはミューテータ（図中白）と GC（図中灰色）から成ることを示している．図 4(a) では，アプリケーション実行中に周波数の制御は特に行わない．一方図 4(b) では，ミューテータを高い周波数で実行し，GC 開始時に周波数を下げて GC を実行する．GC が終了すると周波数を元の高い周波数に戻して，ミューテータの処理を再開する．



(a) 周波数制御をしないアプリケーションの実行

(b) 提案手法によるアプリケーションの実行

図 4 処理の種類による周波数の動的な変更
Fig. 4 Dynamic changing frequency according to kind of task

6. 実装と評価

提案手法を Android 上で動作する Java 仮想機械である Dalvik VM に提案手法を実装した．

周波数を変更する制御については，Android の一部である Linux カーネルにより提供されている API（cpufreq）を利用して，Dalvik VM の GC 開始時と終了時に周波数を変更する制御を追加した．Dalvik VM は，GC の処理中にミューテータの処理を止める停止型 GC と，GC の処理とミューテータの処理が並行して動作しうる並行 GC を持っており，起動オプションにより切り替えられる．今回の実装では，提案する周波数制御方式の基本的な効果を調べることが目的なので，停止型マークスイープ GC を対象とした．

評価対象として，予備実験と同じくワンボードコンピュータである Pandaboard を用いた．この Pandaboard 上で Android 4.2.2 を動作させ，その上で提案手法を実装した Dalvik VM を動かした．

消費電力は，Pandaboard への電源供給部に抵抗を挟み電圧降下をオシロスコープで測ることにより求めた．こうして計測した電圧降下から得られた消費電力から，周波数が 300MHz でアイドル状態に求めた消費電力を減じることで，アプリケーションの実行に要したダイナミック電力を求めた．

6.1 実験方法

提案手法の有効性を調べるために，GC 時に周波数を下げる実行（実行 A）と，実行 A と同じ時間だけかかるように GC の実行を考慮せずにある割合の時間だけ周波数を下げて行う実行（実行 B）について，消費電力量を比較した．これは，GC に着目することで，より多くの消費電力の削減が出来ることを示すために，GC を考慮した場合と考慮しない場合を比較するためこのような評価方法を用いた．実行 A では効果が分かりやすいように，ミューテータの実行周波数は 1008MHz で，GC の実行周波数は 300MHz とした．これらは Pandaboard の最高および最低周波数である．

実行したプログラムは，メモリ管理のベンチマーク集である DaCapo ベンチマーク [8] の実験環境で動作した 6 つのベンチマーク antlr，hsqldb，luindex，lusearch，pmd，xalan である．ベンチマークのヒープサイズはいずれも 256MB である．

実行 B の消費電力量は，全体を通して高い周波数での実行（実行 B_H），および，全体を通して低い周波数での実行（実行 B_L）の実行時間と消費電力量を計測し，これらから計算によって求めた．なお，この計算において周波数変更によるオーバーヘッドは無いものとした．実行 B の消費電力量 E_B は次のように求める．実行 B の高い周波数で実行する処理の量の割合を α とし，実行 A, B, B_H, B_L における実行時間をそれぞれ $T_A, T_B, T_{B_H}, T_{B_L}$，消費電力量をそれぞれ $E_A, E_B, E_{B_H}, E_{B_L}$ とすると，

$$T_A = T_B \tag{1}$$

$$T_B = \alpha T_{B_H} + (1-\alpha)\, T_{B_L} \tag{2}$$

$$E_B = \alpha E_{B_H} + (1-\alpha)\, E_{B_L} \tag{3}$$

が成り立つ．式 (1)，(2) から，

$$\alpha = \frac{T_A - T_{B_H}}{T_{B_H} - T_{B_L}} \tag{4}$$

となり，式 (3)，(4) を用いて，

$$E_B = E_{B_L} + (E_{B_H} - E_{B_L})\frac{T_A - T_{B_L}}{T_{B_H} - T_{B_L}} \tag{5}$$

となる．これにより，実行 B の消費電力量 E_B が求まる．

6.2 実験結果

6 つのベンチマークプログラムを実行した時の消費電力量の比を**図 5** に示す．図の縦軸は，実行 B を基準とした実行 A の消費電力量の比であり，これが 1 より小さいと，提案手法を用いた方が消費電力量が小さいことを示す．

ベンチマークプログラムのうち，antlr，luindex，lusearch，xalan は比 E_A/E_B が 1.0 前後であり，GC に着目して周波数を下げた効果はあまり見られない．一方，hsqldb，pmd は比がそれぞれ 0.71，0.85 であり，提案手法の効果が確認できる．

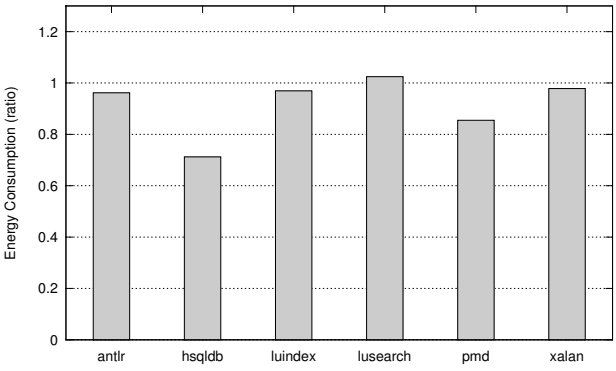


図 5 消費電力量の比
Fig. 5 Ratio of energy consumption

実行結果の詳細を**表 2** に示す．表の各平均 GC 時間は，1 回の GC にかかる時間の平均を表す．この表から，効果の見られなかった antlr，luindex，lusearch，xalan は平均 GC 時間が短く，効果の見られた hsqldb，pmd は，平均 GC 時間が長いことが分かる．従って，平均 GC 時間が長いプログラムは，本手法の効果が高いと考えられる．またこれは，使用しているメモリ領域が大きいことを意味する．

この結果を利用することでスマートフォンなどで動作するアプリケーションへの応用が期待できる．これらのアプリケーションは，複雑な処理を行ない，大量のメモリを使用し，頻繁にオブジェクトを生成するようなプログラムも増えてきている．こういったアプリケーションにおいては，アプリケーションの実行の GC の占める割合も大きくなり，提案手法による効果も期待できる．

7. 関連研究

cpufreqd [9] は，cpufreq を利用した Linux 上で動作するデーモンである．cpufreqd は，CPU の使用状況や温度等の状態と，ユーザが記述するポリシーファイルに従って動的に周波数を変更し，消費電力を削減することができる．本研究では，ユーザに記述を求めることなく消費電力の削減を図っている．

Marculescu [2] は，一定の期間についてキャッシュミスを調べそれに応じて CPU への供給電圧を制御することにより消費電力の削減を行っている．吉川 [3] は，キャッシュミスを検知すると CPU 周波数を下げ，バス周波数と同一にして実行することで消費電力を削減する手法を提案している．これらの手法には，キャッシュミスを検知するハードウェアが必要となる．更に，キャッシュヒット率を検知してから実際の CPU 動作周波数を変更するため，キャッシュミスの多い処理が実際に始まってから適用までに遅延が発生してしまう．本研究では，GC に着目し周波数の制御を行っているため，キャッシュミスを検知するハードウェアは不要であり，キャッシュミスの多いと考えられる

表 2 実行結果の詳細
Table 2 Details of result

ベンチマーク	antlr	hsqldb	luindex	lusearch	pmd	xalan
実行時間 (s)	243.073	152.175	201.488	387.263	439.342	261.13
総 GC 時間 (s)	84.6058	113.793	39.3871	272.316	279.977	68.8188
GC 回数	719	19	204	914	336	312
平均 GC 時間 (s)	0.118	5.99	0.193	0.298	0.833	0.221
実行 A の消費電力量 E_A (J)	204.837	61.912	196.722	227.413	224.657	313.081
実行 B の消費電力量 E_B (J)	212.999	86.910	202.953	222.059	262.813	320.491
E_A/E_B	0.961	0.712	0.969	1.024	0.854	0.978

GC の処理を遅延なしに消費電力の削減を図っている．

深津 [4] は，キャッシュヒット率を予測し，それに応じて事前に CPU 動作周波数を変更する手法を提案している．この手法は，OS のカーネル部において，システムコールや割り込み等の各イベントについて予測キャッシュヒット率をテーブルとして保持しておき，それに従ってキャッシュヒット率を予測する．さらに，動的に予測するだけではなくプログラムのコンパイル時に，中間コードに対して同様の予測を行い周波数を制御するコードを埋め込むことにより，静的に周波数の制御をする手法も提案している．これに対して，本研究は GC に着目している．GC は VM 上の処理であり，あらかじめまとまった処理時間を必要とし，キャッシュミスを起こしやすいことが分かっているが，OS 側からその事象を知ることはできない．そのため，OS カーネルによる予測で解決できない消費電力を削減できると考えられる．

吉田ら [10] は，仮想機械（言語仮想機械ではなく，計算機が動作する仮想機械）環境における，マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラである Accele スケジューラを提案し，実装している．これは仮想機械上の仮想 CPU スケジューラによりスケジューリングされるタスクの全てが低い周波数のタスクになるように，スケジューリングを行う．低い周波数のタスクを合わせることで物理 CPU の周波数を一挙に低くすることができ，それに伴い物理 CPU の供給電圧も下げることができる．それにより消費電力を削減できる．吉田らの研究と比較すると，本研究は対象とするレイヤーが異なる．提案手法を実装した Dalvik VM を吉田らの仮想機械上で実行することで，提案手法を単独で用いるより大きな降下が期待できる．

Cao ら [11] は，高機能なプロセッサと低機能なプロセッサを持つような AMP（Assymetric Multicore Processors）環境を用いて通常のアプリケーションの動作を高機能なプロセッサ，GC を低機能なプロセッサで処理することを提案し実装した．その結果，性能向上と消費電力の削減をすることができた．本研究では，一般的な SMP 構成や単一 CPU でも適用が可能である．

Griffin ら [12] は，Java 仮想機械の一つである KVM に参照カウンタ GC を実装し，KVM の既存実装のマークスイープ GC との比較を ARM プロセッサのシミュレータ上で行っている．その結果，1 次キャッシュミスが削減され，いくつかのアプリケーションで消費電力量を削減した．また，Velasco ら [13] は，マークスイープ GC や世代別 GC といった複数の GC アルゴリズムについて PowerPC プロセッサのシミュレータ上で実際のアプリケーションを動かし，どの GC が消費電力に配慮した GC であるか調査した．その結果，マークスイープ GC やコピー GC に対して世代別 GC が消費電力の削減に関して優れていること，メモリの階層構造が結果に多いに影響を与えていることを確認した．これらの研究に対して，本研究では，GC のメモリアクセスが多い処理が続き，キャッシュミスが起きやすいという特徴を利用して周波数の制御を行っている．従って上記の研究結果と併用して，より消費電力に配慮した GC アルゴリズムにおいて本手法を適用することも可能である．ただし，世代別 GC の場合，通常の GC とは別に，マイナー GC と呼ばれる GC も実施される．これは，GC の対象となるヒープの範囲が狭く処理時間も短いことから，この処理については別途考慮する余地があると考えられる．

8. おわりに

本稿では，Android の Java 仮想機械である Dalvik VM において，より小さい実行時間の増加でより大きい消費電力の削減を行うために，GC 時に周波数を下げて実行する機構を停止型マークスイープ GC に実装し，評価を行った．これにより，比較的 GC 時間の長いベンチマークにおいては特にこの効果が得られることが確認できた．

今後の課題としては，まず，適応的に周波数を制御することが挙げられる．6 章では，GC 時間の長いプログラムについて，提案手法による消費電力の削減効果が高いことを確認した．これより，長い時間が必要な GC の時のみ周波数を下げて実行することでより消費電力の削減が可能であると考える．例えば，直近の GC 時間を記録しておき，その GC 時間とあらかじめ定めておいた閾値よりも長い GC の時のみ周波数を下げるような機構にすることで，より消費電力の削減が可能ではないかと考えられる．

また，並行 GC への対応が挙げられる．今回の実験においては，本手法の効果を分かりやすくするために停止型 GC を対象とした．しかし，Android で動作するアプリケーションは通常並行 GC で動作する．また，携帯端末の CPU 自体もマルチコア化が進んできており，マルチコア環境においてより効率的に動作させるために並行 GC についての検証が必要となる．

更に，今回実験に使用したベンチマークプログラムは，GC の性能比較に用いられるベンチマークであるため，一般的な携帯端末上で動作するアプリケーションとは動作が異なる可能性がある．従って，携帯端末上で動作するアプリケーションの動作の傾向について調査を行い，それに見合ったベンチマークプログラムを用いて評価する必要がある．また，ベンチマークプログラムだけでなく，評価を行うアルゴリズムについても，キャッシュヒット率に応じて周波数を制御するもの等と比較を行うことで，より現実的な手法となることが期待される．

本研究ではマークスイープ GC を対象としたが，将来的には，他の GC 方式について検討を行い，提案手法を用いて，より消費電力削減の効果の高い GC 方式を検討することも考えられる．

謝辞 本研究は JSPS 科研費 25330080 の助成を受けたものです．

参考文献

[1] Venkatachalam, V. and Franz, M.: Power reduction techniques for microprocessor systems, *ACM Comput. Surv.*, Vol. 37, No. 3, pp. 195–237 (2005).

[2] Marculescu, D.: On the use of microarchitecture-driven dynamic voltage scaling, *Proceedings of Workshop on Complexity-Effective Design* (2000).

[3] 吉川恭史：情報処理装置，特許 4860104 号 (2012)．特許 4860104 号, 2012-01-25.

[4] 深津　元：情報処理装置，特許 4837456 号 (2011)．特許 4837456 号, 2011-12-14.

[5] Intel Corporation: Intel® 64 and IA-32 Architectures Software Developer's Manual. `http://www.intel.com/design/processor/manuals/253669.pdf`.

[6] Carlson, B. and Giolma, B.: SmartReflex™ Power and Performance Management Technologies: reduced power consumption, optimized performance. `http://focus.ti.com/pdfs/wtbu/smartreflex_whitepaper.pdf`.

[7] Pandaboard. `http://pandaboard.org/`.

[8] Blackburn, S. M. et al.: The DaCapo benchmarks: java benchmarking development and analysis, *Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, OOPSLA '06, New York, NY, USA, ACM, pp. 169–190 (2006).

[9] Mattia, D. and George, S.: Cpufreqd. `http://www.linux.it/~malattia/wiki/index.php/Cpufreqd`.

[10] 吉田哲也，山田浩史，佐々木広，河野健二，中村　宏：マルチコア CPU の電力消費特性を考慮した仮想 CPU スケジューラ，情報処理学会論文誌コンピューティングシステム（ACS），Vol. 4, No. 2, pp. 25–39 (2011).

[11] Cao, T., Blackburn, S. M., Gao, T. and McKinley, K. S.: The yin and yang of power and performance for asymmetric hardware and managed software, *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, Washington, DC, USA, IEEE Computer Society, pp. 225–236 (2012).

[12] Griffin, P., Srisa-an, W. and Chang, J. M.: An energy efficient garbage collector for java embedded devices, *Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems*, LCTES '05, New York, NY, USA, ACM, pp. 230–238 (2005).

[13] Velasco, J. M., Atienza, D., Olcoz, K., Catthoor, F., Tirado, F. and Mendias, J. M.: Energy characterization of garbage collectors for dynamic applications on embedded systems, *Proceedings of the 15th International Conference on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation*, PATMOS '05, Berlin, Heidelberg, Springer-Verlag, pp. 69–78 (2005).