Vol.2009-SLDM-140 No9
2009/5/21

OS レベルのプロファイリング情報を用いた携帯端末アプリケーションの消費電力モデリング

間嶋 崇[†1] 横山哲郎[†2] 曾剛[†1]
神山 剛[†3] 冨山宏之[†1] 高田広章[†1]

通信端末のシステムレベルの消費電力予測を行う手法を提案する．本手法はプロセッサと周辺デバイスの消費電力を OS レベルのプロファイル情報のみを用いて予測するため，オフラインとオンラインの両者に適応可能である．商用の通信端末において通信を伴うアプリケーションを実行した評価では，本手法による予測消費電力の実測値に対するフィッティング誤差は 10 % 以内に収まった．

Power Modeling of Applications in Wireless Communication Devices Using OS Level Profiles

Takashi Majima,[†1] Tetsuo Yokoyama,[†2]
Gang Zeng,[†1] Takeshi Kamiyama,[†3]
Hiroyuki Tomiyama[†1] and Hiroaki Takada[†1]

We propose a novel system level power estimation method for wireless communication devices. The method estimates the power consumption of processors and external devices by only using OS level profile information provided by system calls. The proposed method is applicable to on-line power estimation as well as off-line power estimation. Evaluations on a commercial wireless communication device showed that the method achieved an estimation error of less than 10 %.

1. はじめに

近年，携帯端末の使用可能時間を伸ばすことへのユーザの要望が高い．しかし，今後，電池容量が劇的に増加することは望めない．携帯端末の大きさ，重量，コストによる電池サイズの制約のため，携帯端末の電池容量は限られているからである．また，現在主流のリチウムイオン電池のエネルギー密度は理論的限界に迫っており[11)]，繰り返しの充放電により電池の満充電容量は低下していく．一方，アプリケーションの高機能化に伴って携帯端末の消費エネルギーはますます増加している．したがって，携帯端末においてアプリケーションの実行を制御することによって消費エネルギーを最適化することが課題となっている．

消費エネルギーの最適化手法は 2 種類に分類される．1 つは設計空間探索やコンパイラによるオフラインの手法，もう一方は実行時の測定・フィードバックによるオンラインの手法である．消費電力の実測は困難を伴うため，両手法において消費電力モデルが活用されることが多い．多くのオフラインの手法において消費エネルギー予測は複雑で時間のかかる命令またはアーキテクチャレベルのシミュレーションが用いられている．一方で，オンラインの手法における消費エネルギー予測手法は実行オーバヘッドの制限が厳しいため比較的少数である．既存のオンライン電力予測手法はハードウェア情報が必要であり，汎用の組込みプロセッサにそのまま適用することができない．さらに，既存手法のほとんどがプロセッサの消費エネルギーの予測に焦点を当て，通信デバイスなどを含むシステム全体の消費エネルギーの予測を目標としたものではない．したがって，これらの手法は，携帯端末において特に重要である，正確な電池使用可能時間の予測に直接用いることが困難である．

本稿では，通信を伴う携帯端末アプリケーションの消費電力モデリング手法を提案する．本手法は OS レベルのプロファイリング情報のみを用いてシステム全体の消費電力の予測を行う．消費電力モデルには線形式を用い，その係数は重回帰分析により推定する．

本稿における主な提案は以下の通りである．

- 提案手法では，アプリケーションのトレースログから取得が容易なシステムコール情報のみを用いて消費電力を予測する．したがって，提案モデルは計算時間の比較的かから

†1 名古屋大学 大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†2 南山大学 情報理工学部
Faculty of Information Sciences and Engineering, Nanzan University
†3 株式会社 NTT ドコモ 先進技術研究所
Research Laboratories, NTT DOCOMO, Inc.

ない軽量なモデルであり，オフライン時のみならずオンライン時の消費電力の最適化に活用できる．

- 提案手法は，プロセッサのみならず通信デバイスなどの周辺デバイスを含めたシステム全体の消費電力が対象である．したがって，システム全体の電池使用可能時間の予測に活用できる．
- Nokia 製の携帯端末 N810 において，提案手法の消費電力の予測値は実測値に対して平均誤差 10% 以内であった．

本稿の構成を述べる．2 章で関連研究を述べる．3 章で本稿で提案する消費電力モデルについて述べ，4 章でこのモデルの評価実験を行う．5 章でまとめと今後の課題について述べる．

2. 関 連 研 究

本章ではアプリケーションソフトウェアの消費電力予測の関連研究について述べる．この分野には多くの関連研究が存在するが，そのうちの代表的なものについて紹介する．

アプリケーションソフトウェアの消費電力の予測手法は大きく分けて 2 つ存在する．その 1 つが，プロセッサのパフォーマンスモニタユニット（PMU）を使った実行時での電力管理を目的するオンライン電力予測である．文献 1) では XScale プロセッサの消費電力予測用に既存の 5 つの PMU を用いた線形電力モデルが提案された．しかし，XScale プロセッサでは最大で 2 つの PMU しか同時に観測できないので予測を 1 回行うために，同一コードを複数回実行しなければならない．そこで，文献 2) では，設計時にプロセッサに一回観測できる PMU 数の上限を撤廃した．この手法は実行時の消費電力を効果的に予測できるが，ハードウェア情報を必要とし伴うコストが高いという欠点がある．

もう 1 つの手法は，シミュレーションによる予測手法である．この手法はさらに命令レベルとアーキテクチャレベルに分類される．命令レベルの消費電力の予測手法は各命令列の消費電力を調べるために網羅的なシミュレーションが必要である[3]．それに対して，文献 4) は消費電力への影響で命令をクラス分けし，各クラスの組み合わせのみを考慮することで必要なシミュレーション回数の削減させた．一方，Wattch[5] と Sim-Panalyzer[6] は SimpleScaler[7] と呼ばれるマイクロプロセッサの消費電力をアーキテクチャレベルのシミュレーションに基づいて計算する手法である．一般的に，アーキテクチャレベルのシミュレーションには時間がかかるという問題がある．シミュレーション時間を削減するために，ソフトウェアに対するマクロモデリングが提案された[8),9]．プロファイリング統計値とアルゴリズムの複雑度の情報がモデルのパラメータとして使用された．この手法は高速化が進んだ一方で，マクロモデリングの構成は困難でありパラメータの選択はソフトウェア依存である．よって，様々なアプリケーションのオンライン時の消費電力の予測には適していない．

既存手法に対し，提案手法は以下の特徴がある．第 1 に，特別なハードウェアを必要とせず，オフライン・オンラインの消費エネルギーの予測ができる．第 2 に，既存手法では主にプロセッサの消費電力を対象としているが，提案手法はプロセッサに加え周辺デバイスを含むシステムレベルの消費電力を対象としている．第 3 に，提案手法は命令やアーキテクチャに基づいたシミュレーションを必要とせず，OS レベルのプロファイリング情報のみを用いて電力予測を行う．

3. 消費電力モデル

本稿では，消費電力は，基本システムおよび周辺デバイスの消費エネルギーからなると仮定する．本稿では，RISC プロセッサを対象とする．DSP などでは各命令の単位時間あたりの消費エネルギーが大幅に異なるため，命令レベルの解析が有効である．しかし，RISC では，この差が小さいため，CPU への負荷や周辺デバイスの制御が消費電力を決定する．高負荷 CPU のアプリケーションが実行されたとき，プロセッサにおける消費電力は，周辺デバイスとの入出力や制御がないならば一定であると仮定する．周辺デバイスとの入出力や制御が行われたときは，その種類に応じてプロセッサの消費電力が下がり，周辺デバイスの消費電力が上がるものとする．

本稿では，OS から取得可能な情報の中からシステムコールを選定した．システムコールとは，あるタスクにおいて，必要なパーミッションを持たないプログラムを実行するときに，カーネルに行われる要求である．システムコールは，関数呼出しとして実行されるため，そのプロファイルは OS レベルで取得可能である．システムコールの選定理由は，第 1 に，周辺デバイスを制御するときには，システムコールが用いられるからである．したがって，システムコールのプロファイルから周辺デバイスの制御情報が抽出できることが期待される．第 2 に，システムコールのプロファイルは，粒度が適切であるためである．すべての関数呼出しのプロファイルを取得した場合は，取得に掛かるオーバヘッドが大きくなりすぎる．したがって，電力予測のためのシミュレーションに掛かる時間が膨大になり，アプリケーションの実行時に電力予測のための計算オーバヘッドによって行われる電力消費が無視できなくなる．一方，アプリケーションの種類だけをプロファイルした場合は，電力予測の誤差が大きくなりすぎる．なお，以降では，システムコールによって制御されるものを周辺デバイス，そうでないものを基本システムに対応させて考える．

Linux システムにおいて，システムコールのプロファイルは strace により取得可能である．オプション指定により各システムコールの引数や実行時間といった情報も取得可能である．本稿の実験では，strace をシステムコールのプロファイル取得に用いる．

本稿では，システムコールのプロファイルから得られた値の線形式によって消費電力を予測する．線形式を用いた理由は，モデルが簡単なため扱いが容易で，回帰分析により係数を推定できるためである．

線形モデル式において何を独立変数にするかを決定するために，システムコール発行回数を変数にして予備実験を行ってみた．通信を伴うアプリケーションにおいて，非常に大きなフィッティング誤差が得られた．したがって，単純にシステムコールの発行回数を変数にするだけでは消費電力予測がうまくいかないことが示された．この理由の 1 つとして，入出力をするデバイスが異なるにも関わらず，read と write の実行回数を計数していたことが考えられる．入出力デバイスを考慮するためにはシステムコールの引数の違いを考慮する必要がある．たとえば，通信を行う場合とフラッシュへファイル出力を行う場合は，同じ write が用いられても消費電力の傾向は大きく異なる．実際，N810 において，単位時間受信ファイル容量と消費電力の関係は，ある範囲では対数を取ったものに線形であった．これは詳しくは 4.2 節において議論する．以降では，システムコールの引数を考慮に入れる．以上を踏まえ，線形モデル式

$$\begin{aligned} P \;=\; & k_{\mathrm{const}} + k_{\mathrm{idle}} \times P_{\mathrm{idle}} + k_{\mathrm{receive}} \times \log_2 f(P_{\mathrm{receive}}) \\ & + k_{\mathrm{read}} \times P_{\mathrm{read}} + k_{\mathrm{write}} \times P_{\mathrm{write}} \end{aligned} \tag{1}$$

を定める．ただし，P を電力 (W)，k_x を x に関する定数係数，P_{idle} をシステムコール select による待機時間比率（$0 \leq P_{\mathrm{idle}} \leq 1$），$P_{\mathrm{receive}}$ を単位時間あたりの受信ファイル容量 (B/s)，P_{send} を単位時間あたりの送信ファイル容量 (B/s)，P_{read} を単位時間あたりのフラッシュからの読み込みファイル容量 (B/s)，P_{write} を単位時間あたりのフラッシュへの書き込みファイル容量 (B/s) とする．f は

$$f(x) = \begin{cases} c, & \text{for } x \leq b \\ x, & \text{for } b \leq x \end{cases} \tag{2}$$

のように，ある閾値 b までは定数関数，それ以上では恒等関数である．

4. 評価実験

4.1 実験環境

評価端末は，Nokia N810 Internet Tablet (**表 4.1**) を用いた．N810 は商用の携帯端末である．したがって，エンドユーザが実際に使用するような，様々なデバイスからなるシステムの消費エネルギーを計測することが可能である．Maemo は Debian GNU/Linux ベースの OS である．したがって，シェルを用いてコマンドライン入力が行える．本稿のすべての実験は，コマンドライン入力をバッチ処理で行うことで実現された．

以下の点を工夫することで実験の再現性を高めた．プロセッサの消費電力がアプリケーションの管理外で制御されないように，カーネルコンフィギュレーションにおいて，DVFS をオフに設定した．この設定でコンパイルしたカーネルを N810 で使用した．実験に不必要な Bluetooth およびディスプレイはオフに設定した．バックグラウンドプロセスはできるだけ停止した．N810 と無線 LAN 基地局の距離は常に 20 cm に保った．アプリケーションの実行は電力的に安定した状態で実行することを目標にした．このため各アプリケーション実行の前後に 5 秒間スリープ時間を設けた．N810 を長時間稼働させていると，対象アプリケーションを実行していないときでも，2.15 秒周期で消費電流値が 20-200 mA の範囲を上下に振動した．この場合，実験前に無線 LAN の基地局を再起動した．再起動後しばらくは，消費電流値は振動しなかった．このようにして，本稿における実験が行われる前には，N810 の定常状態で 20-30 mA の消費電流が観測されていた．

N810 には，付属の電池からではなく，**図 2** のように周辺機器と接続することで，電力を供給した．安定化電源には菊水の PMC18-5A を，電流測定器には横河電機の WT 1600 を用いた．電源において，電圧は N810 に付属の電池の仕様と同じ 3.70 V を，電流の測定レンジは 0.00-5.00 A を設定した．電流および電力の誤差はそれぞれ，読み値 0.1 %，測定レンジ 10 mA であった．測定器では，50 ms 毎に電力の測定を行った．消費エネルギーは，各測定電力に 50 ms を掛け，積算したものとした．測定器は，実験の 2 時間以上前に電源が入れられ，ウォームアップが行われた．対象アプリケーションを N810 で実行し，消費電力とシステムコールのログデータをそれぞれ取得した（**図 1**）．各データから電力予測に必要な値を抽出した．すなわち，電力のログデータからアプリケーション実行時の平均電力を，システムコールのログデータから注目しているシステムコールの統計値を抽出した．次に，それぞれを組合わせ，平均電力のベクトルとシステムコールに関する統計量の行列を作成した．消費電力を従属変数，システムコールに関する統計量を独立変数とした回帰分析によって線形モデル式 (1) の係数を推定した．ここで，目的に応じて線形モデル式 (1) の右辺の項は一部を消去したものを用いた．係数をもとにフィッティングの評価を行った．係数をもとに他のアプリケーションの消費電力を予測し，消費電力モデル式の評価を行った．

システムコールのログ取得には，簡単のため strace を用いた．このログ取得は電力計測

Vol.2009-SLDM-140 No9
2009/5/21

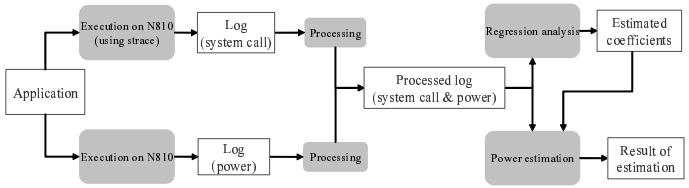

図 1 実験の流れ

表 1 Nokia N810 の構成

Processor	TI OMAP 2420, 400 MHz
Memory	DDR RAM 128 MB, Flash 256 MB
Storage	2 GB internal memory, 8 GB microSD memory card
WLAN	IEEE 802.11b/g
Bluetooth	Version 2.0 + EDR
OS	Maemo Diablo 4.1

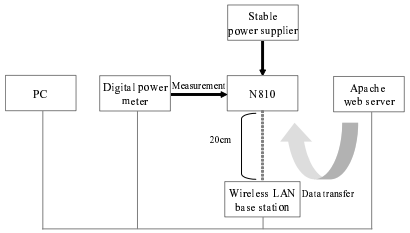

図 2 実験環境

とは独立に行った．これは，`strace` を用いた場合アプリケーションの時間オーバヘッドが 60 % 以上になることもあり，アプリケーションをそのまま実行したときと比べ消費エネルギーが変化してしまうためである．

実験の単位はアプリケーションとした．すなわち，アプリケーション全体を実行したときの時間，システムコール情報，平均消費電力などを解析対象の統計量とした．これは，`strace` によってアプリケーションを実行するときのオーバヘッドが無視できないほど大きく，消費電力のログと `strace` のログの照合が難しいためである．この改良は今後の課題である．

回帰分析には MATLAB を用いた．線形モデル式の回帰係数の推定には，最小二乗法を用いる関数 `regress` を使用した．入力データ間の相関係数は関数 `corrcoef` を用いて求めた．

4.2 結果と考察

高負荷 CPU のアプリケーションが実行されたとき，基本システムにおいて消費電力が一定であるという仮定の妥当性を評価した．組込みベンチマークスイート MiBench[10] から 20 種類のアプリケーションを選択し実験を行った．入力データは MiBench 付属のものを用いた．入力データが複数存在する場合は大きい（large）ものを選択した．実験はそれぞれの設定で 8 回行い平均消費エネルギーを求めた．2 秒以上の実行時間のアプリケーションでは，アプリケーションの種類によらずほぼ一定の消費電力を示した（**図 3**）．実行時間が

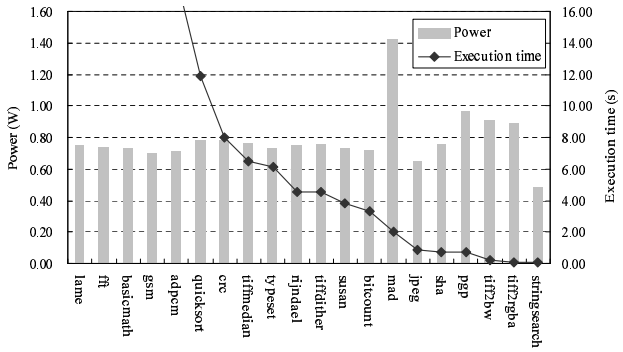

図 3 MiBench のアプリケーションの消費エネルギー

表 2 各アプリケーションの最大・平均フィッティングエラーと推定された係数

Application	Max. err.	Mean err.	k_{const}	k_{idle}	k_{receive}	k_{read}	k_{write}
MiBench	5 %	2 %	0.74	—	—	2×10^{-5}	-7×10^{-5}
GNU Wget-1	28 %	7 %	−0.21	−0.20	0.183	—	—
GNU Wget-2	48 %	9 %	−0.53	—	0.214	—	—
MiBench+Wget	38 %	9 %	0.73	−0.84	−0.113	6×10^{-5}	-3.0×10^{-4}
scp	30 %	10 %	0.85	−1.00	0.070	—	—

1 秒に満たない 6 種類のアプリケーションはサンプル数が少ないため，また mad は外れ値を示したため，以降では除外して考えることにする．これらを除外した残りのアプリケーションの平均消費電力は 0.74 A，分散は 5.8×10^{-4} であった．すなわち，消費電力のばらつきは低いという結果が得られた．**表 2** には，左から順に，アプリケーション名，最大フィッティング誤差，平均フィッティング誤差，線形モデル式の定数係数の値が示されている．MiBench の結果において，P_{read} と P_{write} の相関係数は 0.6 であった．$k_{\mathrm{read}} \times P_{\mathrm{read}}$ は，tiffmedian, crc, tiffdither において，それぞれ 0.07, 0.04, 0.03 W であり，残りのアプリケーションでは誤差の範囲であった．$k_{\mathrm{write}} \times P_{\mathrm{write}}$ は，tiffmedian において，絶対値が最大の −0.04 であった．このように，ファイル入出力の消費電力への影響は限られていた．

各命令と消費エネルギーを関連づけなくても，高負荷 CPU のアプリケーションの消費エネルギーは一定であることが示された．したがって，N810 において，高 CPU 負荷のアプリケーションの消費電力予測には命令の粒度の消費エネルギーの解析は必ずしも必要ない．

通信を伴うアプリケーションの消費エネルギーの予測を式 (1) で行ったときの精度を評

価した．対象アプリケーションには GNU Wget を用いた．Wget は，ウェブから非対話的にファイルのダウンロードを行うアプリケーションである．LAN 内のウェブサーバに異なるサイズのファイルを置き，N810 において Wget によりファイル取得を行った．通信帯域は，ウェブサーバ Apache のモジュールの設定によって制限を行った．ファイルは，コマンド `dd` によりゼロビットのみを含むものを用意した．設定通信帯域は 8 KiB から 2 MiB，取得ファイルサイズは 64 KiB から 16 MiB の範囲から，それぞれ 2 の倍数となる値を用いた．実験結果には，ファイルの取得時間が 2 秒以上 40 秒以下になるもののみを用いた．各ファイル取得は 10 回行った．解析に用いた統計値は，10 回の実験の平均値とした．消費電力は通信帯域が増加するにつれて緩やかに増加した（**図 4**）．この緩やかな増加のため，通信帯域を狭めることによる消費エネルギー最適化の効果は，Wget を対象とした場合は存在しないことが示された．通信帯域を変化させることによって電力のスケーラビリティは最大 8.0 倍であった．(2) における閾値 b は約 2^3 とした．すなわち 8 ($= 2^3$) bps 以下では，ほぼ一定の消費電力であり約 0.08 A であった．const, idle, receive の 3 係数の線形モデル式（GNU Wget-1）を用いた場合，P_{idle}, P_{receive} の相関係数は -0.9 であった．この相関係数が高いため，どちらか一方の項を線形モデル式から除いても，フィッティングエラーをそれほど悪化させることなく消費電力の予測が行えることが予測される．実際，const, receive の 2 係数の線形モデル式（GNU Wget-2）では，Wget-1 からの消費電力のフィッティング誤差悪化は 2 % にとどまった．最大フィッティング誤差は 20 % も悪化した．しかし，これは消費電力の絶対値が小さい通信帯域が 8 bps での値であった．消費電力の絶対値の差でみた場合，最大 0.09 W のフィッティング誤差でしかなかった．

実行時にオンラインで消費電力の予測をするとき，少しの精度悪化でプロファイリングのオーバヘッドを減らせたら，システムの消費電力の削減につながる．たとえば，上の結果ではプロファイリングの値を Wget-1 の 2 つから Wget-2 の 1 つにすることが可能である．これによってオンラインのオーバヘッドは半分になる．

Wget の実験結果のフィッティングによって得られた線形モデル式は通信を伴わないアプリケーションの予測には向かない．むしろ，Wget の消費電力予測に特化したモデルと言える．たとえば，MiBench のアプリケーションは通信を伴わず P_{idle} が 0 となる．この場合，Wget-1 と Wget-2 の実験から得られた係数を用いると，消費電力は，それぞれ 0.139 W と 0.112 W となる．しかし，これは MiBench の結果（図 3）とは大きく異なる．一方，図 4 では，最小二乗法による近似直線の誤差が小さかった．したがって，通信帯域と消費電力の相関関係は高く，一方から他方への予測は精度が高いことが示された．

取得ファイルサイズが変化しても，Wget において発行されるシステムコールは一部しか変化しなかった（**図 5**）．システムコール `select`，`read`，`write` は取得ファイルサイズが大きくなるにつれ発行回数が増加した．これらのシステムコールの発行回数はほぼ取得ファイルサイズに比例した．通信帯域が変化しても Wget において発行されるシステムコール `select`，`read`，`write` を除いて変化しなかった．受信ファイル容量の消費電力に与える影響は限られているため，`select`，`read`，`write` 以外のシステムコールは，消費電力の説明に有効ではないことが示された．これは，式 (1) において，限られたシステムコールで消費電力を説明を試みていることは，Wget に関しては適切であったことを示している．

MiBench と Wget を用いて推定した係数を用いた線形モデル式で別のアプリケーションの消費電力の予測を行った．MiBench からは，basicmath, fft, crc, lame, tiffdither, tiffmedian, typeset を用いた．これは，独立変数の係数推定を行うために，ファイルの入力か出力かいずれかに偏ったデータを用いれば充分であると考えたためであった．MiBench の P_{receive} と P_{idle}，Wget の P_{read} は常に 0 であった．表 2 における k_{const} の値は，MiBench の結果でパラメータ推定を行ったときとほぼ同じであった．Wget のみで係数推定を行った場合は，k_{const} の値は負であり，これとは異なる結果になった．消費電力の予測を行うアプリケーションには，scp を選定した．scp とは，ネットワークを通してファイルをホスト間でコピーするアプリケーションである．scp は，セキュアシェルを用いてデータ転送を行う．非対話的実験を行うため，認証時にキーボード入力が必要のない設定を行った．すなわち，認証には空白のパスフレーズを用いた．アプリケーションを 10 回実行し平均消費電力を求めた．scp の消費電力プロファイルは，開始から 6 秒ほどはファイルサイズによらずほぼ同じであった（**図 6**）．その後，一定の消費電力のプロファイルが続いた．この消費電力は設定された通信帯域によって，このプロファイルの長さは受信ファイルサイズによって異なった．このように，scp の消費電力プロファイルの特徴は Wget とは大きく異なっていた．scp のプロファイルにおいて，P_{write} に対して，P_{receive} と P_{idle} の相関係数は，それぞれ 0.6, -0.6 であった．それ以外の係数間の相関係数の絶対値は 0.5 以下であった．

すべてのアプリケーションにおいて，平均フィッティング誤差は 10 % 以内であった（表 2）．したがって，PMU などを用いなくても，OS レベルのプロファイリング情報だけを使ってもある程度の精度の予測が可能であることが示された．

5. おわりに

本稿では，携帯端末において通信デバイスなどを含むシステムレベルの消費電力モデリン

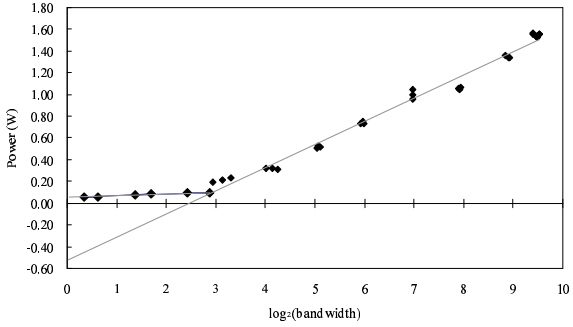

図 4 通信帯域を変化させたときの消費電力

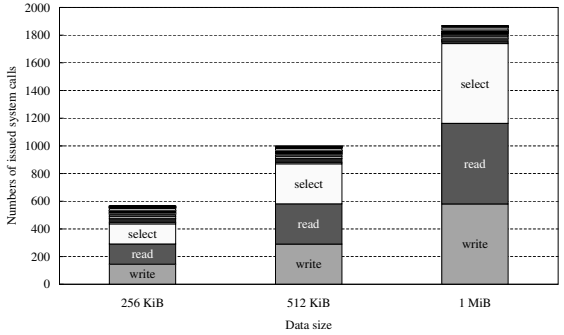

図 5 受信ファイル容量ごとの Wget 実行時システムコール発行回数

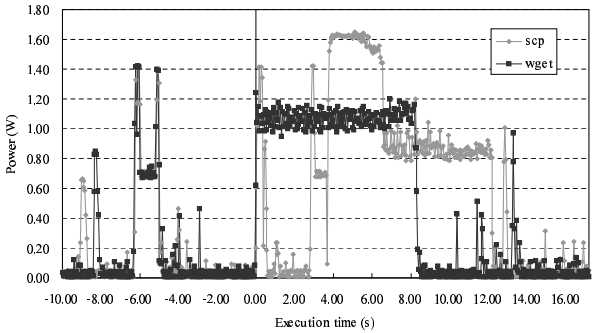

図 6 scp と Wget の消費電力プロファイル

グを行った．提案手法は，OS レベルのプロファイリング情報のみを用いて消費電力を予測していることを特徴とする．評価実験では，商用の通信端末を用いた．MiBench を用いた評価実験により，CPU が高負荷で通信を伴わないアプリケーションの消費電力はほぼ一定と仮定できることが検証された．したがって，システム全体の消費電力を予測するには周辺デバイスの制御を監視すればよい．我々は周辺デバイスの制御はシステムコールで行われることに着目し，このプロファイルを活用した．通信を伴うアプリケーションを解析することで，I/O 待ち，通信，ファイル I/O が最も消費電力に影響があることが判明した．このことから，評価実験ではこれらに関係するシステムコール情報をパラメータ化した線形モデル式によって消費電力の予測を行った．評価実験において，本手法による予測消費電力の実測値に対するフィッティング誤差は 10 % 以内に収まった．

謝　　辞

本研究を進めるに当たりご協力頂いた九州大学の石原亨准教授，奥平拓見氏に感謝します．

参 考 文 献

1) G. Contreras and M. Martonosi. Power prediction for Intel XScale® processors using performance monitoring unit events. In *Proc. ISLPED*, pages 221–226, 2005.
2) J.Peddersen and S.Parameswaran. CLIPPER: Counter-based low impact processor power estimation at run-time. In *Proc. ASP-DAC*, pages 890–895, 2007.
3) V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Trans. on VLSI Systems*, 2(4):437–445, 1994.
4) A. Sinha and A. P. Chandrakasan. JouleTrack: A web based tool for software energy profiling. In *Proc. DAC*, pages 220–225. ACM Press, 2001.
5) D.Brooks, V.Tiwari, and M.Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. ISCA*, pages 83–94. ACM Press, 2000.
6) The SimpleScalar-Arm power modeling project. `http://www.eecs.umich.edu/~panalyzer/`.
7) T.Austin, E.Larson, and D.Ernst. SimpleScalar: An infrastructure for computer system modeling. *Computer*, 35(2):59–67, 2002.
8) T. K. Tan, A. Raghunathan, G. Lakshminarayana, and N. K. Jha, High-level Software Energy Macro-modeling, In *Proc. DAC*, pages 605–610, ACM Press, 2001.
9) A. Muttreja, A. Raghunathan, S. Ravi, and N. K. Jha, Automated energy/performance macromodeling of embedded software, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 26(3):542–551, 2007.
10) M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *IEEE International Workshop on Workload Characterization*, pages 3–14, 2001.
11) T.L. Martin. *Balancing batteries, power, and performance: system issues in CPU speed-setting for mobile computing*. PhD thesis, Carnegie Mellon University, 1999.
12) Strace. `http://sourceforge.net/projects/strace/`