Vol.2018-ARC-232 No.20
2018/7/31

低レイテンシSSDアクセス時におけるCPU消費電力の削減

今村 智史[1,a)]　吉田 英司[1]

概要：
近年，従来の NAND フラッシュ SSD に比べ約 1/10 のレイテンシでアクセス可能な「低レイテンシ SSD」が登場している．低レイテンシ SSD に対する I/O アクセスでは，CPU コアで実行される OS 処理の時間が I/O レイテンシの大きな割合を占めるため，CPU コアの動作周波数が I/O レイテンシに大きな影響を与える．つまり，動作周波数を高く設定するほど I/O レイテンシは短縮される．その一方，低レイテンシ SSD への I/O アクセス時には CPU コアが深いスリープ状態に入ることができないため，動作周波数を高く設定するほど I/O 待ち時間中に CPU が消費する電力が高くなってしまう．
そこで本論文では，I/O アクセス時の CPU 消費電力を削減するために，ブロックデバイスの使用率を考慮した DVFS 手法を提案する．本手法では，対象のブロックデバイスの使用率を定期的に計測し，その使用率が極めて高い場合に CPU コアの供給電圧と動作周波数を低減する．この場合，実行アプリケーションの性能が CPU ではなくブロックデバイスにより律速されるため，その性能を低下させることなく CPU の消費電力を削減できる．Intel® Optane™ SSD を搭載した実サーバにおいて 28 種類の I/O インテンシブワークロードを用いて提案手法の効果を評価した結果，Linux の動作周波数制御ドライバに比べ性能をほとんど低下させることなく CPU の消費電力を平均 43.7%（最大 56.3%）削減できることが明らかになった．

1. はじめに

次世代メモリ技術を採用した低レイテンシ SSD が 2017 年頃から市場に登場し始めた [1]．そのデバイスレイテンシは約 10 マイクロ秒であり，従来の NAND フラッシュ SSD に比べ約 1/10 の長さである．この短いデバイスレイテンシにより高速な I/O アクセスが可能となった一方で，I/O アクセス毎に実行される OS 処理のオーバーヘッドが顕著になってきた．このオーバーヘッドを削減するためにこれまで数多くの最適化技術が提案されているが，I/O レイテンシ全体に対するその割合は依然として大きいままである [2]．

I/O アクセス時の OS 処理は CPU コアで実行されるため，その処理時間はコアの処理速度すなわち動作周波数に依存する．そのため，低レイテンシ SSD に対する I/O アクセスでは，コアの動作周波数が I/O レイテンシに大きな影響を与える．言い換えれば，コアの動作周波数を高く設定するほど I/O レイテンシが短縮される．そこで本研究では，低レイテンシ SSD に対する I/O アクセス時に Linux の intel_pstate ドライバ [3] によって CPU コアの動作周波数がどのように制御されるかを調査した．その結果，低レイテンシ SSD に対する I/O アクセス時には CPU コアの使用率が高くなるため，動作周波数が最大化されることが分かった．しかしながら，その一方で，CPU コアが I/O 待ち時間中においても深いスリープ状態に入ることができず，CPU が高い電力を消費することも明らかになった．

上記の場合，I/O 待ち時間中に DVFS（Dynamic Voltage and Frequency Scaling）を適用することで OS 処理を遅延させることなく CPU の消費電力を削減できる．しかしながら，低レイテンシ SSD の場合には I/O 待ち時間が約 10 マイクロ秒であることに対し，現存の CPU では動作周波数の変更に数十マイクロ秒を要する [4, 5]．したがって，このアプローチは現時点では現実的でない．そこで本論文では，ブロックデバイスの使用率を考慮した DVFS 手法（DU-DVFS: Device Utilization-aware DVFS）を提案する．本手法では，対象のブロックデバイスの使用率を定期的に計測し，その使用率が極めて高い場合に DVFS を適用する．この場合，CPU コアから発行された I/O リクエストが一時的に I/O キューに格納されるため，動作周波数の低下による OS 処理の遅延が隠蔽される．そのため，実行アプリケーションの性能を低下させることなく CPU の消費電力を削減できる．

この提案手法をユーザレベルのランタイムシステムとして実装し，Intel® Optane™ SSD [1] を搭載したサーバにおいて 28 種類の I/O インテンシブワークロードを用いてそ

[1] 株式会社 富士通研究所
[a)] s-imamura@jp.fujitsu.com

表 1: NAND フラッシュ SSD と低レイテンシ SSD の比較

	DC P3700 [6]	Optane™ 900P [1]
テクノロジ	MLC NAND	3D XPoint™
インターフェイス	PCIe NVMe	PCIe NVMe
容量	2 TB	480 GB
逐次読み出し	2,800 MB/s	2,500 MB/s
逐次書き込み	1,900 MB/s	2,000 MB/s
ランダム読み出し	450 KIOPS	550 KIOPS
ランダム書き込み	175 KIOPS	500 KIOPS
読み出しレイテンシ	**120 us**	**10 us**
書き込みレイテンシ	30 us	10 us

の効果の定量的評価を実施した．その結果，intel_pstate ドライバに実装された *powersave* ガバナに比べ，性能をほとんど低下させることなく CPU の消費電力を平均 43.7%（最大 56.3%）削減できることが明らかになった．

2. 背景

2.1 低レイテンシ SSD

低レイテンシ SSD は次世代メモリ技術を採用した新たなストレージデバイスであり，様々な分野での活用が期待されている．**表 1** に，従来の NAND フラッシュ SSD（Intel® DC P3700 SSD [6]）と低レイテンシ SSD（Intel® Optane™ 900P SSD [1]）の仕様の比較結果を示す． 低レイテンシ SSD の容量と逐次読み出しスループットは NAND フラッシュ SSD に比べ劣る一方，ランダム書き込みスループットとレイテンシは大変優れている．特に，読み出しレイテンシは NAND フラッシュ SSD に比べ 1/10 以上短縮されている．

2.2 OS 処理オーバーヘッドの顕著化

ハードディスクドライブや NAND フラッシュ SSD といった従来ストレージデバイスのレイテンシは 100 マイクロ秒〜数ミリ秒と長く，I/O アクセス毎に実行される OS 処理の時間は無視できるものであった．たとえば，近年主流になりつつある NVMe 規格対応の NAND フラッシュ SSD の場合，I/O レイテンシ全体に対して OS 処理時間の割合は約 5%である．これに対し，低レイテンシ SSD の場合にはこの割合が約 40%にも及ぶ [2]．したがって，低レイテンシ SSD の性能を最大限活用するためには，OS 処理オーバーヘッドの削減が大きな課題である．

このオーバーヘッドを削減するために，これまでに数多くの最適化技術が提案されてきた [2,7]．特に，I/O ポーリングはすでに実用段階にある有望な技術である．I/O ポーリングを適用することで CPU 使用率が高くなるものの，従来の割り込みベース I/O に比べコンテキストスイッチと割り込みハンドラの処理を排除できる [8,9]．しかしながら，I/O ポーリングを適用した場合においても OS 処理オーバーヘッドは依然として無視できるものではない．

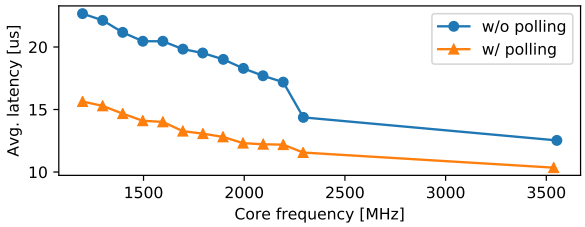

図 1: Optane 900P SSD に対する 4 KB ランダム読み出しの平均レイテンシと CPU コアの動作周波数の相関

3. 低レイテンシ SSD アクセス時における CPU 動作周波数の重要性

3.1 CPU 動作周波数と I/O レイテンシの相関

低レイテンシ SSD に対する I/O レイテンシの大きな割合を占める OS 処理の実行時間は，1 節で述べたように CPU コアの動作周波数に大きく左右される．そこで，表 1 に示した低レイテンシ SSD （Optane 900P SSD）を用いて CPU コアの動作周波数と I/O レイテンシの相関を調査する．本実験では，fio ベンチマーク [10] をダイレクト同期 I/O オプションを用いて 1 コアで実行し，I/O ポーリング非適用時と適用時それぞれの場合において 4 KB ランダム読み出しの平均レイテンシを計測する（実験環境の詳細は 5.1 節を参照）．**図 1** に示した結果から，I/O ポーリング適用の有無に関わらず，動作周波数を高く設定するほど I/O レイテンシが短縮されることが分かる．したがって，I/O レイテンシを最小化するためには動作周波数を最大化する必要がある．また，I/O ポーリングを適用することで I/O レイテンシを大幅に削減できることも分かるため，以降の実験では特に言及しない限り I/O ポーリングを適用する．

3.2 従来の DVFS 手法とその課題

最近の Linux カーネルでは，intel_pstate ドライバ [3] を用いて CPU コアの P ステート（供給電圧と動作周波数のレベル）を制御できる．なお，本論文では単純化のために動作周波数を P ステートとして使用し，動作周波数を変更する際には供給電圧も同時に変化するものとする．このドライバでは，稼働するコアの動作周波数を最大化する *performance* ガバナとコアの使用率に応じてその動作周波数を制御する *powersave* ガバナの 2 種類が用意されている．そこで，Optane 900P SSD に対する I/O アクセス時の両ガバナによる CPU コアの動作周波数制御とその制御が CPU 消費電力と I/O レイテンシに与える影響を調査する．**図 2** に，I/O ポーリング非適用時と適用時それぞれの場合における結果を示す．なお，実験方法は前節と同様であり，このグラフには表 1 に示した NAND フラッシュ SSD（DC P3700 SSD）を用いた場合の結果（I/O ポーリング非適用時）も参考のために含めている．

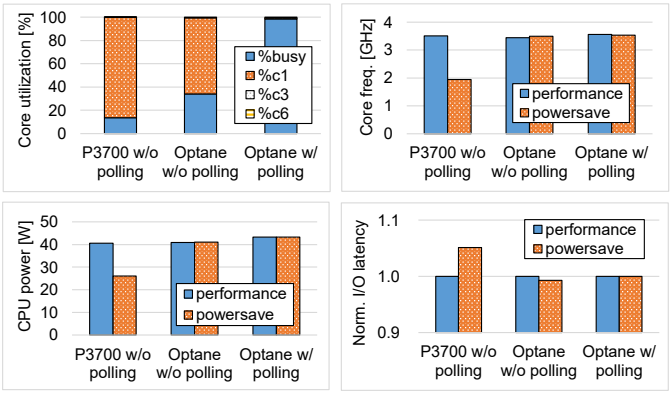

図 2: DC P3700 SSD および Optane 900P SSD に対する 4 KB ランダム読み出し時の動作周波数制御ガバナの評価

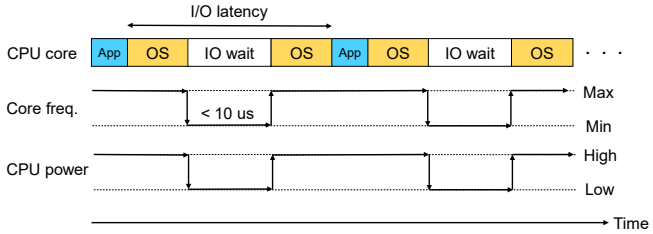

図 3: 低レイテンシ SSD アクセス時の理想的な DVFS 手法

DC P3700 SSD に対する I/O アクセスでは，OS 処理の時間に比べデバイスレイテンシが長いため CPU コアの使用率が 14%と極めて低くなっている．しかしながら，そのデバイスレイテンシはコアが C3 や C6 といった深いスリープ状態となるには短すぎるため，コアは I/O 待ち時間中に最も浅い C1 スリープ状態となる．この状態ではコアのクロックは停止される一方，供給電圧は遮断されずキャッシュメモリも停止しない [11]．そのため，動作周波数を最大化（3.6 GHz）する performance ガバナを適用した場合には CPU の消費電力が 40 W と高くなる．一方，powersave ガバナを適用した場合には，コアの低い使用率に基づいてその動作周波数が 1.9 GHz に低減される．これにより，I/O レイテンシは 5%ほど増大するが CPU の消費電力が 26 W まで削減される．

これに対し，Optane 900P SSD に対する I/O アクセスの場合には，コアの使用率がポーリング I/O 非適用時に 34%，適用時に 98%となる．この場合，両ガバナともコアの動作周波数を最大化し，その結果 I/O レイテンシが最小化される．しかしながら，その一方で，CPU は I/O 待ち時間中においても 40 W を超える高い電力を浪費してしまう．この際，コアの動作周波数を意図的に低減することで CPU の消費電力を削減できるが，図 1 で示したように動作周波数を低減するほど I/O レイテンシが増大してしまう．

3.3 理想的な DVFS 手法

前節の結果から，I/O レイテンシを最小化するためには動作周波数を最大化する必要がある一方，I/O 待ち時間中の CPU 消費電力を削減するためには動作周波数を最小化すべきことが分かった．**図 3** は，CPU コアから低レイテンシ SSD に対して同期 I/O リクエストを連続して発行する場合の理想的な動作周波数制御方法を表している．アプリケーションおよび OS 処理をコアで実行している際には動作周波数を最大化し，I/O 待ち時間中のみ動作周波数を最小化することで，I/O レイテンシを増大させることなく CPU の消費電力を削減できる．しかしながら，低レイテンシ SSD の場合の I/O 待ち時間はわずか 10 マイクロ秒ほどであるが，現存の CPU では動作周波数の変更に数十マイクロ秒を要するため [4,5]，図 3 のような動作周波数制御は現実的でない． 仮に各 I/O 待ち開始時に動作周波数を低減させる場合，その後の OS 処理を遅延させてしまう．

4. DU-DVFS

4.1 概要

前節で述べたように図 3 に示した DVFS 手法は現実的でないため，本論文ではデバイス使用率に基づく DVFS 手法（*DU-DVFS: Device Utilization-aware DVFS*）を提案する．この手法では，動作周波数の低減による OS 処理の遅延が実行アプリケーションの性能に影響しない場合に動作周波数を段階的に低減し，そうでなければ性能低下を防ぐために動作周波数を即座に最大化する． これにより，I/O インテンシブワークロードの実行時に性能を低下させることなく CPU の消費電力を削減できる．

4.2 デバイス使用率

提案手法では，動作周波数の低減による OS 処理の遅延が性能に影響しない状況を見極めるために，デバイス使用率（device utilization）と呼ばれる指標を用いる．これは，Linux においてブロックデバイスの I/O アクセス状況を監視する *iostat* ツールで使用される指標であり，一定時間内におけるブロックデバイスの稼働時間の割合（%）として定義される [12]． たとえば，対象のブロックデバイスが一秒間常に稼働していた場合には，その期間のデバイス使用率は 100%となる．iostat ツールでは，対象デバイスの稼働時間を計測するために/sys/block/⟨dev⟩/stat ファイルから取得可能な *io_ticks* と呼ばれる値を使用している．デバイスドライバは CPU コアから発行された I/O リクエストを一時的に格納するための I/O キューをコア毎に有しており，io_ticks は I/O リクエストがこのキューに格納されていた合計時間を計測している [13]．ブロックデバイスが先行の I/O リクエストを処理するために稼働している間は後続のリクエストがこのキューに格納されるため，io_ticks の値は対象デバイスがこれまで稼働していた総時間とみなすことができる．

図 4 に，8 KB および 64 KB ランダム読み出し時の Optane 900P SSD のデバイス使用率とスループットを示

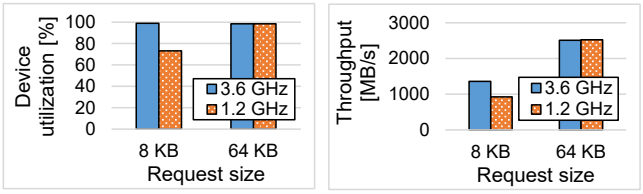

図 4: 8 KB ／ 64 KB ランダム読み出し時の Optane 900P SSD のデバイス使用率（左図）とスループット（右図）

Algorithm 1 DU-DVFS アルゴリズムの擬似コード

```
Input: dev, interval, dec_thr
  prev_io_ticks ← read_io_ticks(dev)
  while 1 do
    sleep(interval)
    curr_io_ticks ← read_io_ticks(dev)
    dev_util ← (curr_io_ticks - prev_io_ticks)/interval/10
    if dev_util > dec_thr then
      decrease_freq_onestep()
    else
      maximize_freq()
    end if
    prev_io_ticks ← curr_io_ticks
  end while
```

す． なお，デバイス使用率を 100%にするために，4 KB より大きなリクエストサイズを使用し fio ベンチマークを 2 コアを用いて実行している．このグラフでは，コアの動作周波数を最大化した場合（3.6 GHz）と最小化した場合（1.2 GHz）の結果を比較する．まず，リクエストサイズが 8 KB の場合には，動作周波数の低減によりデバイス使用率が 100%を下回り，スループットが低下することが分かる．これは，ブロックデバイスが稼働していない間に OS 処理の遅延により I/O レイテンシが増大したためである．これに対し，リクエストサイズが 64 KB の場合には，動作周波数を最小化した際にもデバイス使用率は 100%を維持しており，スループットが低下しない．なぜなら，ブロックデバイスが常に稼働してる間は CPU から発行された I/O リクエストがキューに溜まり，OS 処理の遅延が隠蔽されるためである．上記の結果から，動作周波数を低減させてもデバイス使用率を 100%に維持できる場合であれば，OS 処理の遅延が性能に影響しないことが分かる．

4.3 アルゴリズム

DU-DVFS 手法は，デバイス使用率を定期的に観測しその値を基に CPU コアの動作周波数を制御する．本研究では，I/O インテンシブワークロードが CPU 上の全てのコアを用いて低レイテンシ SSD に対し連続的に I/O リクエストを発行する状況を想定するため，全コアの動作周波数を同時に制御する．本手法は，対象のブロックデバイス（*dev*），デバイス使用率を観測する秒単位のインターバル（*interval*），動作周波数を低減するためのデバイス使用率の閾値（*dec_thr*）の 3 種類のパラメタを入力とする．

提案手法のアルゴリズムを Algorithm 1 に示す．まず，/sys/block/⟨dev⟩/stat ファイルから io_ticks の値（対象デバイスの総稼働時間）を取得し変数 *prev_io_ticks* に格納する．そして，指定されたインターバルの間待機した後，その時点での io_ticks の値を再度取得し変数 *curr_io_ticks* に格納する．その後，curr_io_ticks と prev_io_ticks の差をインターバルの値で割りデバイス使用率を算出する．なお，インターバルは秒単位であり，io_ticks はミリ秒単位であるため，パーセンテージで表されるデバイス使用率を算出するために上記の結果をさらに 10 で割る必要がある．このデバイス使用率が閾値 dec_thr を超えている場合には，CPU の消費電力を削減するためにコアの動作周波数を 1 段階低減する．そうでなければ，性能低下を防ぐために動作周波数を最大化する．最後に， curr_io_ticks の値を prev_io_ticks に代入し，上記の処理を繰り返し行う．

4.4 実装

本研究では，上記の DU-DVFS 手法をユーザレベルのランタイムシステムとして実装する．CPU コアの動作周波数制御は特定の MSR（model specific register）に値を書き込むことで行う．たとえば，Intel の CPU においては *IA32_PERF_CTL* レジスタが該当する [14]．また，本手法ではデバイス使用率を観測するインターバルと動作周波数を低減するためのデバイス使用率の閾値を入力パラメタによって変更できるため，5.3 節にて両パラメタに対するセンシティビティアナリシスを行う．

5. 評価

5.1 評価環境

提案手法の評価には，18 コアの Xeon® E5-2697 v4 プロセッサを 2 基，1.5 TB の DRAM，480 GB の Optane 900P SSD を搭載した PRIMERGY RX2540 M2 サーバを使用する．なお，安定した結果を得るために，Hyper-Threading テクノロジは無効にし 1CPU ソケットのみを用いる．CPU コアの動作周波数は 1.2 GHz から 2.3 GHz まで 0.1 GHz 単位で変更でき，さらに Turbo Boost テクノロジにより 1 コア稼働時には 3.6 GHz まで，全 18 コア稼働時には 2.8 GHz まで動作周波数を上昇できる．また，CPU の消費電力は Linux の *turbostat* ツールを用いて計測する．

本評価では，I/O インテンシブワークロードとして MSR Cambridge Traces [15] を選択し，*blkreplay* ツール [16] を用いてこれらのトレースデータのリプレイを行う．このツールはデータセンタにおけるブロックストレージデバイスの性能検証用に開発されたものであり，マルチスレッド同期 I/O エンジンを用いて実装されている [17] ．なお，ここでは低レイテンシ SSD に極めて高い負荷がかかる状況を想定するため，CPU 上のコア数と同数の 18 スレッドを用いて 100 万倍の速度でトレースのリプレイを行う．ト

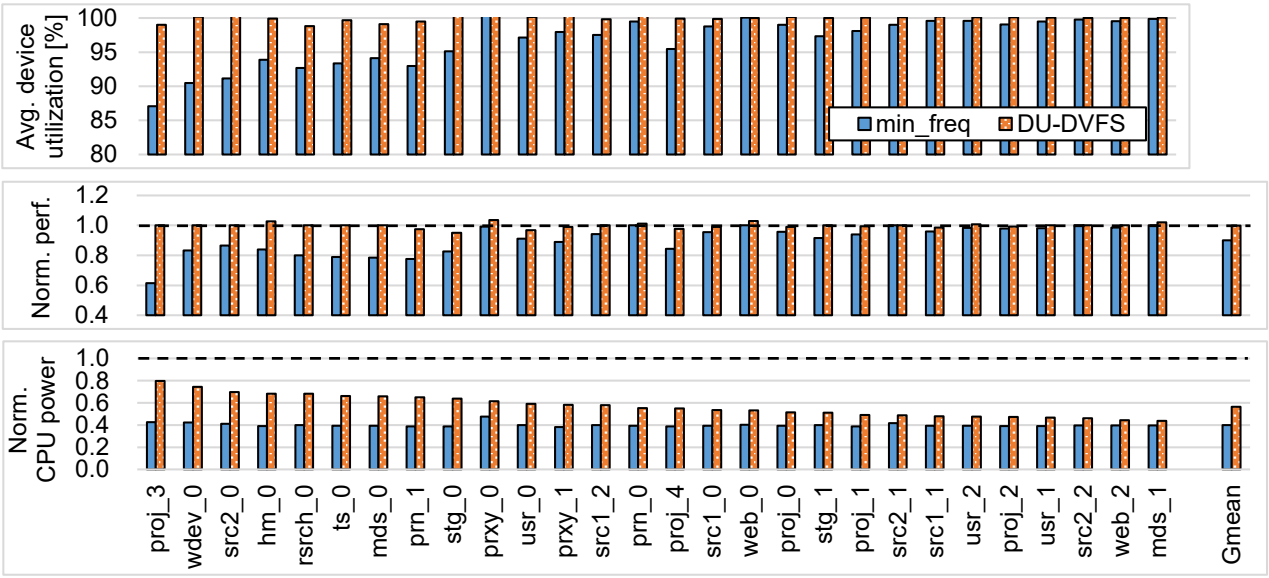

図 5: 28 種類の I/O インテンシブワークロードを用いた DU-DVFS 手法の評価結果

レースデータには 7 日（604,800 秒）間の I/O リクエストが含まれているため，この速度でのリプレイによりタイムスタンプを無視できる．さらに，先行研究と同様に当ツールの write-verify モードと write-protection モードを無効に設定する [17]．Linux のカーネルはバージョン 4.4.117 を使用し，ページキャッシュをバイパスするためのダイレクト同期 I/O オプションを用いて I/O ポーリングを適用する．なお，3 秒以内の短時間で終了するワークロードは評価から除外し，18 スレッドでの実行では大幅に性能が低下した prxy_0 ワークロードのみ 8 スレッドで実行する．

5.2 評価結果

図 5 に，28 種類のワークロードを用いた提案手法の評価結果を示す．ここでは，Optane 900P SSD に対する I/O アクセス時に CPU コアの動作周波数を自動的に最大化する powersave ガバナと最低動作周波数での実行（*min_freq*）を提案手法と比較する．上段のグラフは各ワークロードの実行全体を通してのデバイス使用率の平均値，中段のグラフは性能（実行時間の逆数），下段のグラフは CPU の消費電力をそれぞれプロットしている．なお，性能と CPU 消費電力の結果は，powersave ガバナを適用した場合の結果で正規化しており，提案手法のインターバルと閾値 dec_thr はそれぞれ 100 ミリ秒と 99%に設定している．

まず，図 5 内の左半分のワークロードに関しては，最低周波数での実行（min_freq）によりデバイス使用率が 100%を下回り性能が大きく低下してしまう．これは，動作周波数の低減による OS 処理の遅延が Optane 900P SSD のスループットに影響するためである．特に，proj_3 ワークロードでは powersave ガバナ適用時に比べ性能が 38.7%低下する．これに対し，DU-DVFS 手法ではデバイス使用率を 100%に維持できる範囲で動作周波数を低減するため，

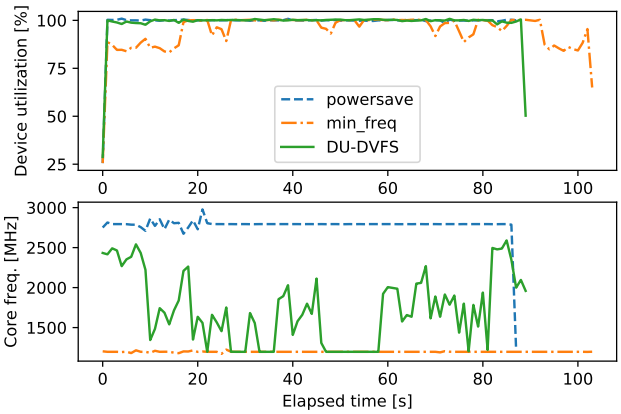

図 6: Proj_4 ワークロード実行時の詳細

性能を低下させることなく CPU の消費電力を削減している．**図 6** は，proj_4 ワークロードの実行中に DU-DVFS 手法により動作周波数がどのように制御されたかを示している．このグラフから，最低周波数での実行によりデバイス使用率が 100%を下回る際（たとえば，0〜20 秒の間）には提案手法が比較的高い動作周波数を選択していることが分かる．一方，最低周波数での実行でもデバイス使用率を 100%に維持できる際（たとえば，45〜60 秒の間）には最低周波数を選択している．

次に，右半分のワークロードに関しては，最低周波数での実行でもデバイス使用率を 100%に維持できるため，大きな性能低下は見られない．この場合，提案手法も最低周波数を選択するため，最低周波数での実行と同等の電力削減効果が得られる．特に，mds_1 ワークロードに対しては，提案手法により CPU 消費電力が 56.3%削減された．

28 種類のワークロードの幾何平均（*Gmean*）では，最低周波数での実行により CPU 消費電力が 60.1%削減される一方で性能が 9.9%低下した．これに対し，提案手法では

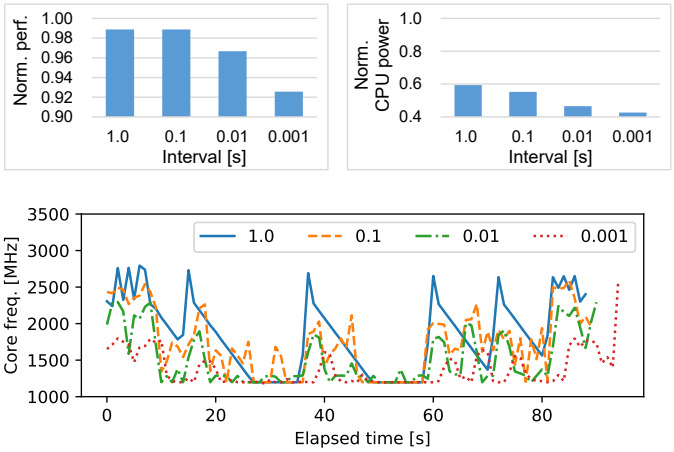

図 7: Interval パラメタを変更した場合の評価結果

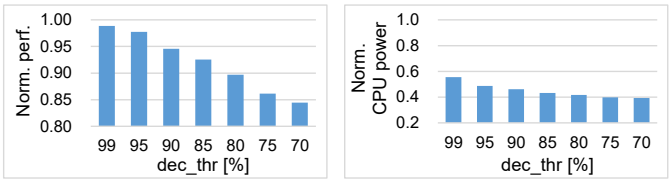

図 8: Dec_thr パラメタを変更した場合の評価結果

性能低下をわずか 0.2%に抑え 43.7%の CPU 消費電力削減を達成できた．

5.3 センシティビティアナリシス

DU-DVFS 手法では interval と dec_thr パラメタによりデバイス使用率を計測する頻度と動作周波数を低下するデバイス使用率の閾値を変更できるため，両パラメタを変更した場合の提案手法の挙動を分析する．なお，ここでは簡潔化のために proj_4 ワークロードの結果のみを記載する．

図 7 上段のグラフは，interval パラメタを 1 秒から 1 ミリ秒まで変更した場合の性能と CPU 消費電力を示している．各グラフの縦軸の値は，powersave ガバナ適用時の結果でそれぞれ正規化されている．これらのグラフから，インターバルが短くなるほど性能が低下する一方で，CPU 消費電力が低くなることが分かる．また，下段のグラフは，inteval パラメタの各値において proj_4 ワークロードの実行中に DU-DVFS 手法が選択した動作周波数をプロットしている．インターバルが長い場合には動作周波数が徐々に低減されていくのに対し，インターバルが短い場合には動作周波数が急激かつ即座に低減されていることが分かる．性能低下を極力避けつつ CPU 消費電力をより大きく削減するためには，interval パラメタは 100 ミリ秒が適切である．

次に，dec_thr パラメタを 99%から 70%まで変更した場合の評価結果を**図 8** に示す．DU-DVFS 手法はこのパラメタを低く設定するほどより積極的に動作周波数を低減するため，より大きな性能低下に伴い CPU 消費電力をより大きく削減できる．つまり，より大きな性能低下が許容できる場面では，より大きな電力削減効果を得ることができる．

6. 関連研究

DVFS は CPU のエネルギー効率を最適化するためのよく知られた技術であり，数多くの先行研究で適用されてきた．しかしながら，ストレージデバイスを考慮した DVFS 手法は下記のように大変限られている．

Lee らは，フラッシュベースのストレージデバイスに内蔵されたマイクロプロセッサやフラッシュコントローラといったハードウェアに対して DVFS を適用している [18]．彼らの手法は，ガベージコレクションやウェアレベリングといったバックグランド処理が指定時間内に完了するよう供給電圧と動作周波数のレベルを制御する．これに対し，本研究では低レイテンシ SSD に I/O リクエストを発行するホスト CPU を対象としている．

Ge らは並列計算システムにおいて I/O アクセスを監視しつつ DVFS を適用する I/O ミドルウェアを開発しており [19]，Manousakis らは I/O インテンシブワークロード向けに CPU，DRAM，ストレージデバイスの消費電力を包括的に監視するフィードバック DVFS 手法を提案している [20]．また， Mills らは，HPC システムにおいて I/O インテンシブなチェックポイント／リスタート処理に対して DVFS を適用している [21]．これらに共通するアイデアは，最大限の計算能力を必要としない I/O インテンシブフェイズにおいて，DVFS を適用することでシステム全体の消費電力もしくは消費エネルギーを削減することである．これらの先行研究ではハードディスクドライブや NAND フラッシュ SSD といったレイテンシの長い従来ストレージデバイスを想定しているため，CPU 動作周波数の低減が性能に与える影響は小さい．本研究では，低レイテンシ SSD に対する I/O アクセス時に図 1 に示すように動作周波数の低減が I/O レイテンシの増大を招くことに着目している．

Saito らも Mills らと同様に，HPC システムにおけるチェックポイント／リスタート処理に対して DVFS を適用する技術を提案している [22]．ただし，彼らはガーベジコレクションやウェアレベリング等の処理を CPU で行う PCIe 接続のフラッシュメモリを対象としており， CPU の動作周波数が I/O レイテンシに直接影響する状況を想定している．これに対し，本研究では CPU の動作周波数が低レイテンシ SSD に対する I/O アクセスのレイテンシに影響することを明らかにした．また，彼らの手法は性能と消費電力のトレードオフを考慮しシステム全体の消費エネルギーを削減しているのに対し，本研究の提案手法は性能を阻害することなく CPU の消費電力を削減することができる．

7. おわりに

本研究では，低レイテンシ SSD に対する I/O アクセスのレイテンシを最小化するためには CPU コアの動作周波数を最大化する必要があることを示し，その一方で動作周波数を最大化する場合には CPU が I/O 待ち時間中に高い電力を浪費することを明らかにした．そこで，ブロックデバイスの使用率が極めて高い場合に CPU コアの動作周波数を低減する DU-DVFS 手法を提案した．実サーバを用いた定量的評価の結果，Linux の powersave ガバナと比較して 28 種類の I/O インテンシブワークロードに対しほとんど性能を低下させることなく平均 43.7%（最大 56.3%）CPU の消費電力を削減できることが明らかになった．

参考文献

[1] Intel®: Intel® OPTANE™ SSD 900P SERIES, https://www.intel.com/content/www/us/en/products/memory-storage/solid-state-drives/gaming-enthusiast-ssds/optane-900p-series/900p-480gb-aic-20nm.html. Last accessed: June, 2018.

[2] Hady, F. T., Foong, A., Veal, B. and Williams, D.: Platform Storage Performance With 3D XPoint Technology, *Proceedings of the IEEE*, Vol. 105, No. 9, pp. 1822–1833 (2017).

[3] Intel®: Intel P-State driver, https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt. Last accessed: June, 2018.

[4] Mazouz, A., Laurent, A., Pradelle, B. and Jalby, W.: Evaluation of CPU Frequency Transition Latency, *Comput. Sci.*, Vol. 29, No. 3-4, pp. 187–195 (2014).

[5] Schöne, R.: A Unified Infrastructure for Monitoring and Tuning the Energy Efficiency of HPC Applications, PhD Thesis, Technischen Universität Dresden (2017).

[6] Intel®: Intel® SSD DC P3700 Series, https://ark.intel.com/products/79620/Intel-SSD-DC-P3700-Series-2_0TB-12-Height-PCIe-3_0-20nm-MLC. Last accessed: June, 2018.

[7] Huang, J., Badam, A., Qureshi, M. K. and Schwan, K.: Unified Address Translation for Memory-mapped SSDs with FlashMap, *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pp. 580–591 (2015).

[8] Yang, J., Minturn, D. B. and Hady, F.: When Poll is Better Than Interrupt, *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, p. 7 (2012).

[9] Le Moal, D.: I/O Latency Optimization with Polling (2017). Vault Linux Storage and Filesystems Conference.

[10] Axboe, J.: Flexible I/O Tester, https://github.com/axboe/fio. Last accessed: June, 2018.

[11] Fischer, W.: Processor P-states and C-states, https://www.thomas-krenn.com/en/wiki/Processor_P-states_and_C-states#cite_note-5. Last accessed: June, 2018.

[12] Godard, S.: Performance monitoring tools for Linux, https://github.com/sysstat/sysstat. Last accessed: June, 2018.

[13] The Linux Kernel Archives: Block layer statistics in /sys/block/dev/stat, https://www.kernel.org/doc/Documentation/block/stat.txt. Last accessed: June, 2018.

[14] Intel: Intel 64 and IA-32 Architectures Software Developer' s Manual Volume 3 (3A, 3B, 3C & 3D): System Programming Guide (2017).

[15] SNIA IOTTA Repository: MSR Cambridge Traces, http://iotta.snia.org/traces/388. Last accessed: June, 2018.

[16] Schöbel-Theuer, T.: blkreplay - a Testing and Benchmarking Toolkit, https://github.com/schoebel/blkreplay/wiki. Last accessed: June, 2018.

[17] Haghdoost, A., He, W., Fredin, J. and Du, D. H. C.: On the Accuracy and Scalability of Intensive I/O Workload Replay, *Proceedings of the 15th Usenix Conference on File and Storage Technologies*, FAST'17, pp. 315–327 (2017).

[18] Lee, S. and Kim, J.: Using Dynamic Voltage Scaling for Energy-Efficient Flash-based Storage Devices, *Proceeding of the 2010 International SoC Design Conference*, ISOCC '10, pp. 63–66 (2010).

[19] Ge, R., Feng, X. and Sun, X.-H.: SERA-IO: Integrating Energy Consciousness into Parallel I/O Middleware, *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '12, pp. 204–211 (2012).

[20] Manousakis, I., Marazakis, M. and Bilas, A.: FDIO: A Feedback Driven Controller for Minimizing Energy in I/O-intensive Applications, *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'13, p. 16 (2013).

[21] Mills, B., Grant, R. E., Ferreira, K. B. and Riesen, R.: Evaluating Energy Savings for Checkpoint/Restart, *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, E2SC '13, pp. 6:1–6:8 (2013).

[22] Saito, T., Sato, K., Sato, H. and Matsuoka, S.: Energy-aware I/O Optimization for Checkpoint and Restart on a NAND Flash Memory System, *Proceedings of the 3rd Workshop on Fault-tolerance for HPC at Extreme Scale*, FTXS '13, pp. 41–48 (2013).