Vol.2013-GN-86 No.37
Vol.2013-CDS-6 No.37
2013/1/17

フラッシュメモリ搭載スマートフォンにおける I/O スケジューリング

中村優太[†1]　服部拓也[†2]　永田恭輔[†3]　野村駿[†4]　山口実靖[†5]

スマートフォン，タブレット型 PC が急速に普及し，その基盤ソフトウェアである Android の重要性が高まっている．Android は Linux を元に開発されたソフトウェアプラットフォームであり，データベースなどのサーバコンピュータ向けソフトウェアも搭載されている．Android 端末ではこれらを用いて高度なデータ処理を行うことも可能であり，その場合は多数の I/O 要求を高速に処理する能力が重要となる．本研究では，フラッシュメモリ搭載の Android 端末の I/O のスケジューリングに着目し，フラッシュメモリ搭載端末における基本 I/O 性能の評価，I/O スケジューリングの影響の評価，I/O スケジューラの性能の評価を行った．基本 I/O 性能の評価では，シーケンシャルリード性能とランダムリード性能の比較を行い，フラッシュメモリ搭載の端末であっても両性能に大きな差があることを確認した．I/O スケジューリングの影響の評価では，I/O 要求群をアドレスでソートした順に処理した場合と，ソートせずに処理した場合の性能を比較し，フラッシュメモリ搭載端末でもソートを行った方が性能が高く，I/O スケジューリングに効果があることを確認した．最後に，I/O スケジューラの性能比較を行い，初期設定で選択されている CFQ スケジューラより，Deadline スケジューラの方が性能が高いことを確認し，その理由について考察を行った．

A Performance Study on I/O Scheduling in Smartphones using Flash Memory Storage

YUTA NAKAMURA[†1]　TAKUYA HATTORI[†2]　KYOSUKE NAGATA[†3]
SHUN NOMURA[†4]　SANEYASU YAMAGUCHI[†5]

As smartphones and tablet PCs gets popular, Android is increasing its importance. Android is an OS which is derived from Linux OS. Thus, Android has highly functional data processing software, such as RDBMS. For complex data processing, Android is required to provide high I/O processing performance. In this paper, we focus on I/O scheduling on Android devices with flash memory and evaluate basic I/O performance, effect of I/O scheduling on performance, and performance of I/O scheduler.
First, we evaluated sequential read performance and random read performance of Android devices. From the results, we found that sequential read performance and random read performance significantly differ even though these devices had flash memory. Second, we compared times to process I/O requests which are sorted by access address and request which are not sorted. The results showed that time to process sorted requests are shorter than that of non-sorted request. It indicated that I/O scheduling had effect on I/O performance in flash memory devices as well as terminals with HDD. Third, we compared performance of I/O schedulers in Android OS and we found that deadline scheduler provided better performance than CFQ scheduler, which is standard I/O scheduler of Android OS.

1. はじめに

近年，スマートフォンやタブレット PC が普及し，それらの携帯端末で動作する組み込み機器ソフトウェアプラットフォームとして Android OS が注目されている．2012 年第 2 四半期において，日本では全携帯電話利用者の約 4 分の 1 にあたる 2400 万人以上がスマートフォンを利用しており，2011 年末に比べて 43%の増加となっている[1]．スマートフォンにおける Android OS の世界シェアは急速に増加しており，2012 年第 2 四半期には 68 パーセントを超え[2]，現在も増加中である．

Android 搭載端末は，従来の携帯電話とは異なり Linux ベースの OS を搭載しているため，RDBMS などの高度なデータ処理も可能となっている．そのため，今後はスマートフォンやタブレット PC でも従来のサーバ計算機と同じように I/O 量の多いアプリケーションが実行されることが可能となり，Android プラットフォームにおける I/O 性能の向上は重要な課題の一つと考えられる．しかし，これまでフラッシュメモリにおいては I/O スケジューリングには効果がなく，フラッシュメモリ搭載 Android 端末の I/O スケジューラとしては NOOP が良いとされ，Android 端末の I/O スケジューラの研究はほとんどされてこなかった．

本研究では，フラッシュメモリ搭載 Android 携帯端末における I/O 基本性能の評価, I/O スケジューリングの効果の評価を行い，I/O スケジューリングの性能を述べる．そして，フラッシュメモリ搭載 Android 端末の I/O スケジューラの評価と解析を行い，スケジューラの変更による I/O 性

†1 工学院大学工学部情報通信工学科
Kogakuin University Department of information and Communications Engineering, Kogakuin University
†2 工学院大学大学院 工学研究科 電気・電子工学専攻
Kogakuin University Electrical Engineering and Electronics, Kogakuin University Graduate School
†3 工学院大学大学院 工学研究科 電気・電子工学専攻
Kogakuin University Electrical Engineering and Electronics, Kogakuin University Graduate School
†4 工学院大学工学部情報通信工学科
Kogakuin University Kogakuin University Department of information and Communications Engineering, Kogakuin University
†5 工学院大学工学部情報通信工学科
Kogakuin University Kogakuin University Department of information and Communications Engineering, Kogakuin University

ⓒ 2013 Information Processing Society of Japan

能向上を示す.

本論文の構成は以下の通りである. 2 章で Android OS と，そのカーネルおよび I/O スケジューラについて紹介する. 3 章ではフラッシュメモリ搭載 Android 端末の I/O 基本性能評価を行う. 4 章では, I/O スケジューリングの効果の検証，I/O スケジューラの有効性について示す．5 章で関連研究を紹介する．6 章で考察を行い，第 7 章にてまとめを述べる.

2. Android OS

2.1 Android

Android OS はアプリケーション，アプリケーションフレームワーク，ライブラリ，Android ランタイム，Linux カーネルでコンポーネントされている[3].

アプリケーションフレームワークは Android 向けのソフトウェア開発をする際に共通して必要とされる命令や関数の集合である．この機能利用することで，開発に要する時間を減らすことが可能となる．アプリケーションフレームワークには“Content Provider”などが含まれており, “Content Provider”はアプリケーション(たとえば Mail アプリケーション)が，別のアプリケーションのデータ（たとえば，連絡先アプリケーションのデータ）にアクセスすること実現したり，アプリケーションデータの共有を実現したりするためのソフトウェアである.

Android には Android システムの様々なコンポーネントで使用される C/C++ライブラリのセットが含まれている．これらの機能は開発者に公開されており，多くの場合アプリケーションフレームワークを介して使用される．コア・ライブラリには System C library や SQLite などが含まれている．System C library は組み込み Linux ベースのデバイス用に調整された BSD 由来の標準的な C システムライブラリ（libc）の実装である．SQLite は軽量なリレーショナル・データベース・エンジンの一つであり，“Content Provider”のようなアプリケーション間のデータ共有の仕組みの中でも利用されている.

Android ランタイムは，仮想マシン“Dalvik VM”と，基本的な API を提供するコア･ライブラリで構成されている．Dalvik VM は Android プラットフォームのために開発された仮想計算機である．低メモリ環境に対して最適化されており, オペレーティングシステムによるプロセス間の分離，メモリ管理，スレッドのサポートを用いて複数の VM インスタンスが同時に動作できるよう設計されている. Android アプリケーションは一般に Java 言語で書かれ，Java ソースファイル(`.java` ファイル)は Java コンパイラによって Java VM 互換の `.class` ファイルへコンパイルされ，`.class` ファイルは Dalvik VM 互換の `.dex` ファイルへ変換される．`.dex` ファイルはリソースファイルと一緒にアプリケーションパッケージファイル（apk）へ圧縮される．そして，アプリケーション起動時に DavlikVM プロセスが `.dex` ファイルを読み込み実行する[4].

Linux カーネルはセキュリティ，メモリ管理，プロセス管理，ネットワークスタック，ドライバモデルなどのコアシステムサービスを利用するために使用されている．Android OS のバージョンによって，搭載される Linux カーネルのバージョンは異なるが，対応したバージョンの大部分が変更されることなく搭載されている．ただし，HDD 搭載の Linux 計算機では HDD は SATA 規格のものも含めて SCSI サブシステムを用いて制御されるが，Android スマートフォンではフラッシュメモリは MMC ドライバによりされる.

2.2 I/O スケジューラ

I/O スケジューラはプロセス群から発行される I/O 要求群を適切な順に並び替え，高性能や高公平性などを実現する OS 内のソフトウェアである．新しい I/O 要求が追加されたときに I/O スケジューラが呼び出され，新しい I/O 要求をキューのどの位置に追加するかを決定する.

HDD 搭載の計算機にておいて I/O スケジューラが I/O 性能に大きな影響を与えることが確認されており[5]，Android においても Linux と同様に NOOP, DEADLINE, CFQ の 3 種類の I/O スケジューラが搭載されている．NOOP はスケジューリングにおける並び替えは行わずに単純に I/O 要求を到着順に処理する．そのため，スケジューリングによる負荷が小さく，一般にフラッシュメモリなどのランダムアクセスが高速なハードウェアに適しているとされる．DEADLINE は現在処理している I/O 要求の近隣の I/O 要求を優先して処理することで HDD ヘッドの移動を削減する．近隣ではない I/O 要求は保留されてしまうが，これには限界値(deadline)が用意されており限界値より長く保留された I/O 要求は優先的に処理される．レイテンシの上限が保障されているため，リアルタイムアプリケーションやデータベース管理システムなどに適しているとされている. CFQ は，プロセスから発行される I/O 要求をプロセス毎のキューに割り振っていくスケジューラであり，処理対象のキューを切り替えながら I/O 要求を処理していく．処理対象キューを一定時間で変更するため，I/O 要求はプロセス単位で公平に処理される．CFQ にも DEADLINE と同様に待ち時間の長い I/O 要求を優先的に処理する機能が備わっている．Android においては，初期状態では CFQ が使用されている.

3. I/O 基本性能評価

フラッシュメモリ搭載 Android 端末における I/O 基本性能を測定するために，シーケンシャルリードとランダムリードの性能評価と，I/O スケジューリングの効果検証を行った.

表 1 測定端末の仕様

端末名	使用 OS	CPU	メモリ	FS
NexusS	4.0.3	CPU cortexA8 (Hummingbird) Processor 1GHz	512MB	VFAT
Nexus7	4.1.2	NVIDIA® Tegra® 3 mobile processor 1.3GHz	1GB	FUSE

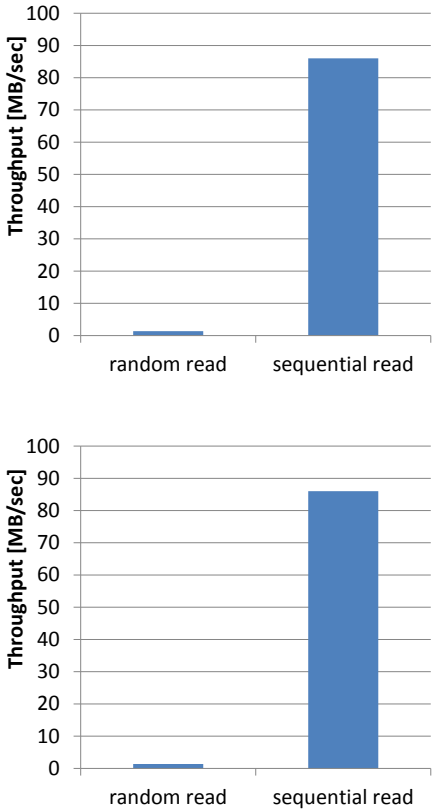

図 1 ランダムリードとシーケンシャルリード性能
(上段が NexusS，下段が Nexus7)

3.1 基本性能評価

フラッシュメモリ搭載 Android 端末のシーケンシャルリード，ランダムリードの性能を図 1 に示す．測定に使用した端末の仕様は表 1 の通りである．測定はフラッシュメモリ内に 1GB のファイルを作成し，それに 4KB の I/O 要求を 1000 回発生させ, 所要時間を計測することにより行った．I/O スケジューラは CFQ を用い，シングルスレッドで行った．一般にフラッシュメモリはランダムアクセスを高速に処理できるとされているが, 図 1 より Android OS を搭載した端末ではシーケンシャルリード性能がランダムリード性能を大幅に上回ることが分かる．この差には OS の先読み処理による影響も含まれていると考えられる．

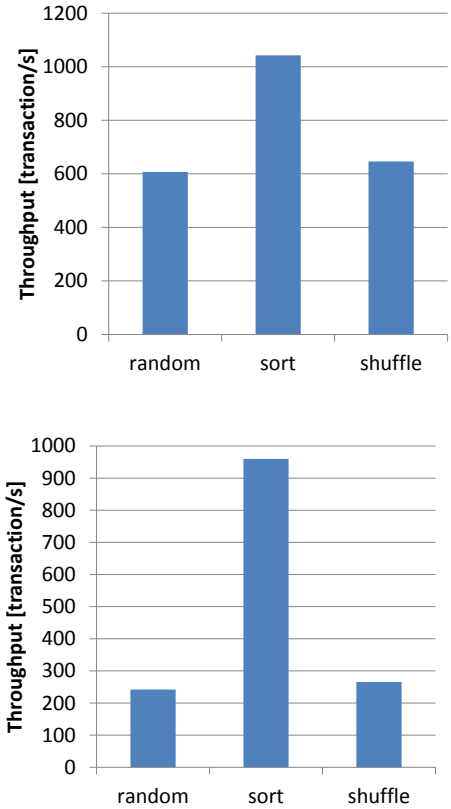

図 2 I/O スケジューリングの効果検証
(上段が NexusS，下段が Nexus7)

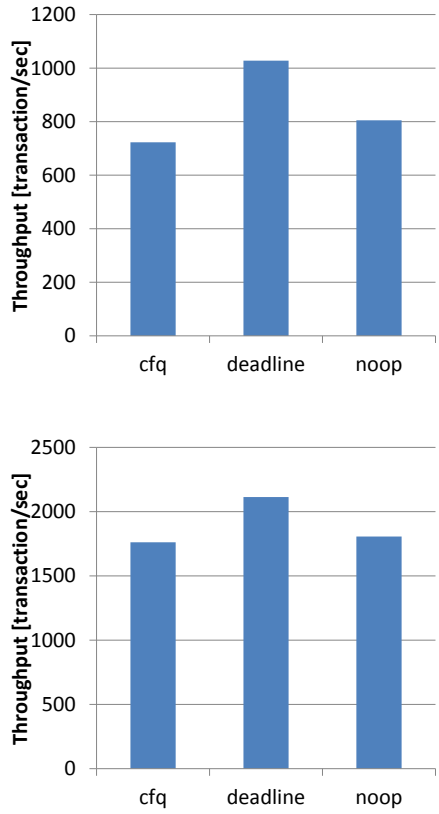

図 3 マイクロベンチによる I/O スケジューラの性能比較
(上段が NexusS，下段が Nexus7)

ⓒ 2013 Information Processing Society of Japan

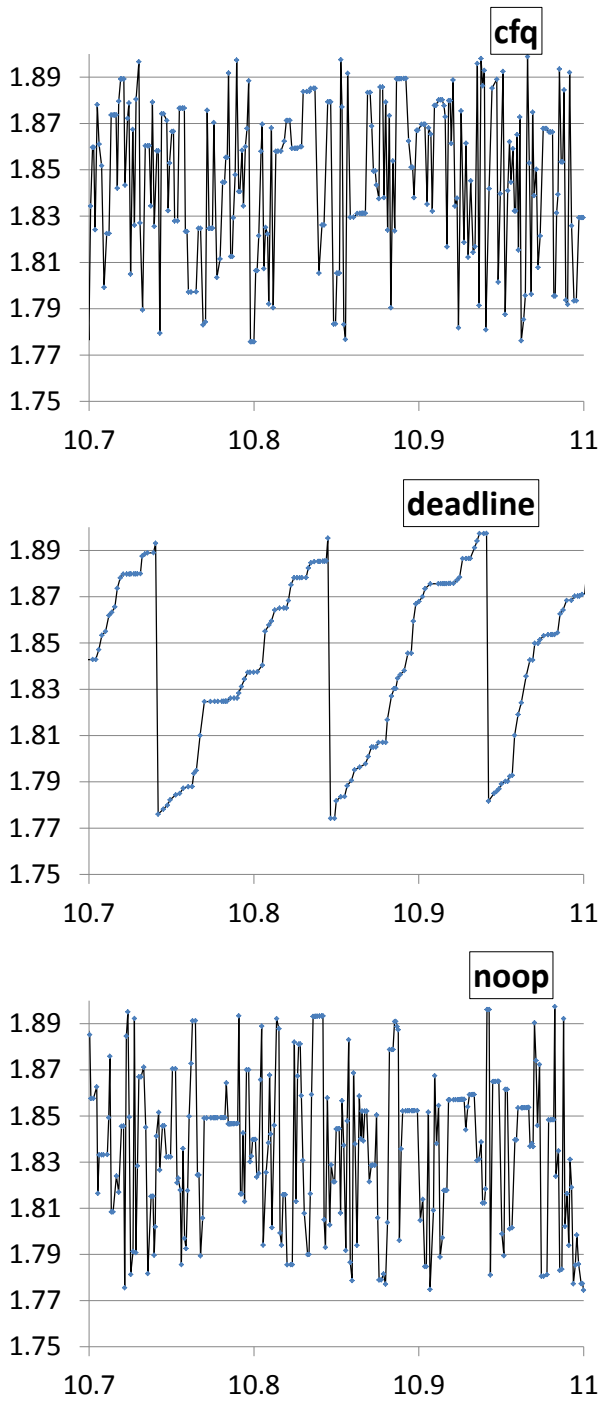

図 4　I/O スケジューリングの様子

3.2 I/O スケジューリングの効果検証

フラッシュメモリ搭載の Android 端末におけるスケジューリングの効果の有無を調査するために，ソート済みアドレスへの I/O 要求と，非ソートアドレスへの I/O 要求の性能を比較する実験を行った．測定に使用した端末は表 1 の通りである．測定は，1GB のファイルに対して，1 バイトの read 要求を 1000 回発行することにより行った．“sort”と“shuffle”では，1GB のファイルを 1MB のブロック 1024 個に分割し，各ブロックの先頭の 1 バイトに対して read 要求を行った．この処理では，各読み込みアドレスは最低でも

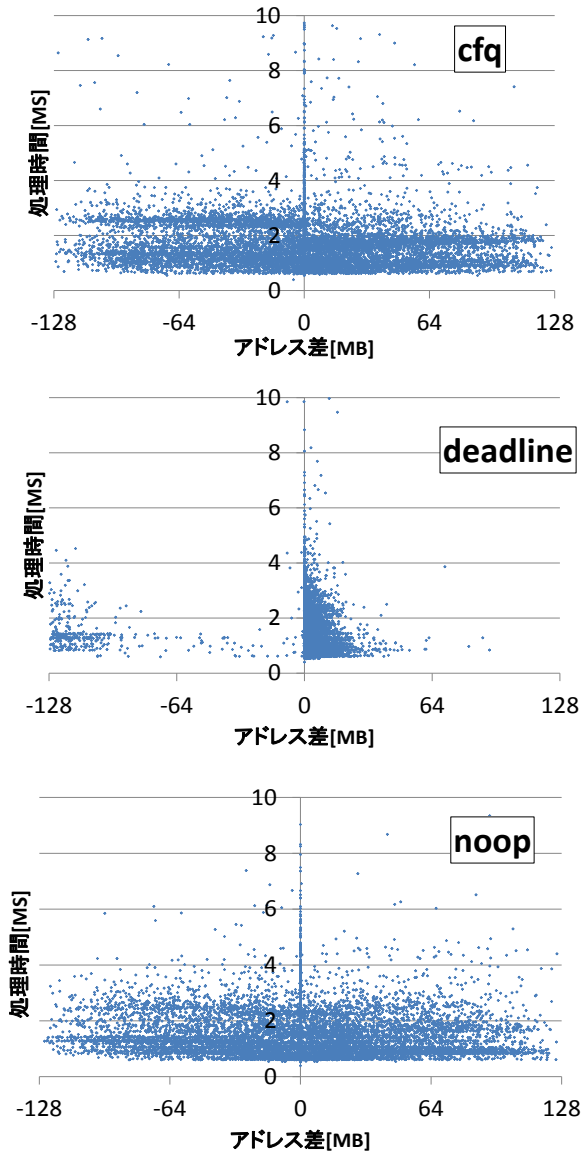

図 5　MMC ドライバ観測結果(NexusS)

1MB は離れることになり，OS による先読み処理の効果はあらわれない．“sort” では先頭ブロックから後方ブロックに対してアドレスが昇順になるように読込を行い，“shuffle”ではランダム順に並び替えて読込を行った．“rand”ではブロック分割をせず 1GB ファイル内のすべてのバイトから対象を決定し読込を行った．測定結果を図 2 に示す．図より，“sort” が”shuffle”よりも性能が高いことがわかる．同じアドレスへの I/O 要求群を異なる順で処理した“sort”と“shuffle”では性能が異なっていることより，フラッシュメモリ搭載 Android 端末においても I/O スケジューリングにより I/O 性能が変化することが分かった．

4. I/O スケジューラの性能評価

マルチスレッド環境下における I/O スケジューラの性能を評価するために，マイクロベンチマークによる性能評価

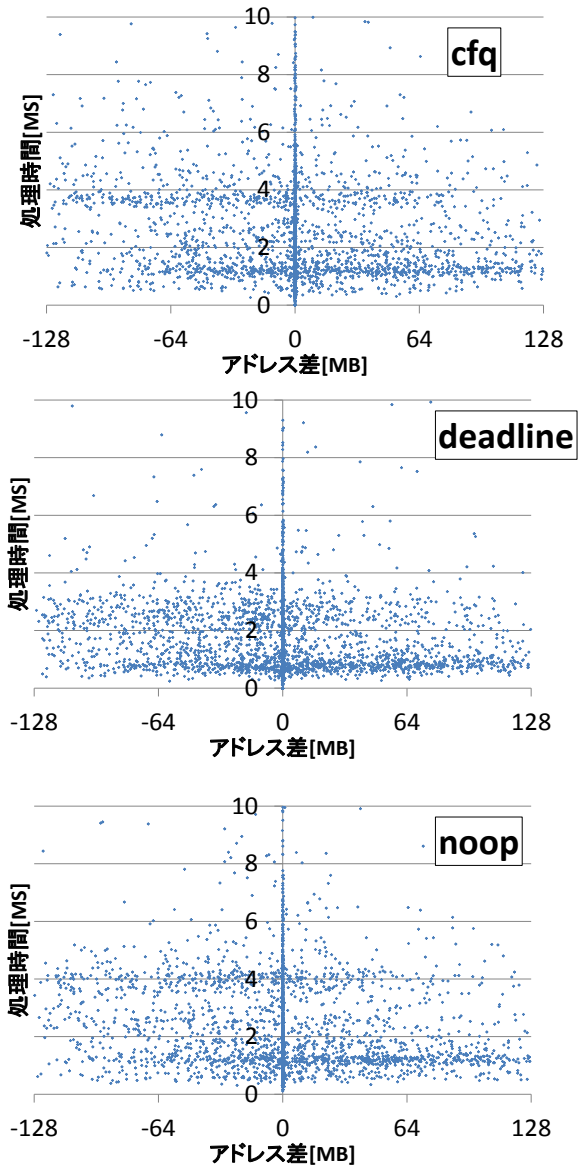

図 6　MMC ドライバの観測結果(Nexus7)

と，アプリケーションベンチマークによる性能評価を行った．

4.1 マイクロベンチマークによる性能評価

マルチスレッド環境における I/O スケジューラの性能を図 3 に示す．測定に使用した端末は表 1 の通りである．性能は，読み込み対象ファイルの 128MB のアドレス範囲からランダム選んだ 1 バイトに対して読込を行う実験を行い測定した．スレッド数は 16 である．Android アプリケーションランタイムの影響を排除するために，ベンチマークプログラムは C 言語で実装し，実行はコマンドラインシェル上にて行った．

図 3 より，フラッシュメモリ搭載 Android 端末においても I/O スケジューラによる性能の違いがあり，DEADLINE が NOOP や初期設定の CFQ より高性能であることがわかる．

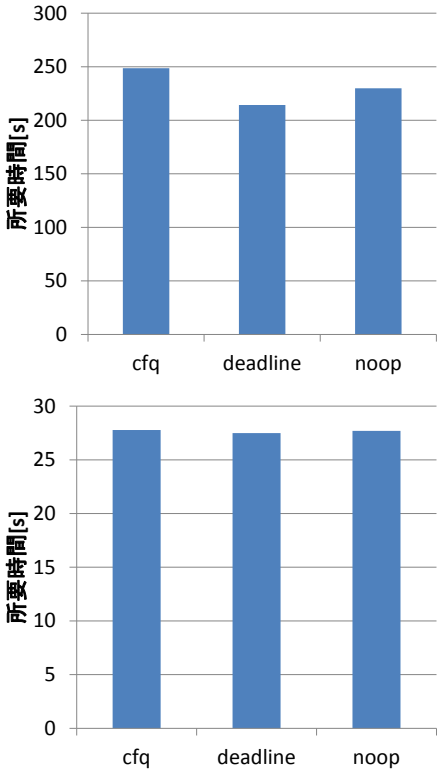

図 7　アプリベンチによる I/O スケジューラの性能比較
(上段が NexusS，下段が Nexus7)

この結果を解析するために，MMC ドライバにて命令が発行された時刻，読み込み先アドレスをカーネルモニタ[6][7]を使用して取得した．

図 4 に各 I/O スケジューラにおける MMC ドライバに対する I/O 要求発行時刻と, I/O アドレスの関係を示す. また, 図 5 と図 6 に観測された I/O 要求のアドレス差(今回のアクセスアドレスと前回のアクセスアドレスの差)と, 処理時間の関係を示す．図 4 より，DEADLINE においては下位アドレスから上位アドレスに移動しながらアクセスを行うことが多く, CFQ, NOOP においてはそのような特徴がないことが分かる. また, 図 5, 図 6 よりアドレスの差が正である(アドレス増加方向に移動する)場合はアドレスの差が負であった場合よりも処理時間が短いことがわかる．また，DEADLINE はアドレスの差が正方向となる場合が多く, 結果として性能が高くなったと考えられる．

4.2 アプリケーションベンチマークによる性能評価

図 7 に，アプリケーションベンチマークによるマルチスレッド環境下における I/O スケジューラの性能評価を示す. 測定に使用した端末は表 1 の通りである．ベンチマークプログラムは Android アプリケーションとして Java 言語で実装され，16 スレッドで 1GB の読み込み対象ファイルのランダムに選んだアドレスに対する 1 バイトの読込を 10000 回行い, 処理に要した時間を測定する. 図より, Nexus S においては DEADLINE が，CFQ，NOOP よりも性能が優れることが分かり，Nexus 7 においては全 I/O スケジューラと

もにほぼ同程度の性能となった．以上より，フラッシュメモリ搭載の Android 端末においても多くの場合で I/O スケジューラが I/O 性能に影響を与え，DEADLINE が初期設定である CFQ の同等以上の性能を出せることが分かった．

5. 関連研究

I/O スケジューラの提案に関する既存の研究としては，Iyer らや Andrew による Anticipatory スケジューラの提案[8][9]がある．I/O スケジューラの性能に関する研究としては，Pratt らによる性能評価[10] や，Axboe による性能評価[11] がある．また，2 階層の I/O スケジューリングを考慮した研究として Cello[12] が，自己学習型（Self-Learning）ディスクスケジューラの研究として Zhang らによる研究[13] がある．

また，仮想化環境における I/O スケジューラの研究として，Boutcher の性能評価[14] ，Kesavan らの仮想ディスクの振る舞いと通常の物理ディスクの振る舞いの違いを調査[15], Xu らのプロセスの I/O 要求の局所性を利用した新しいスケジューラを提案[16]，IBM による巨大ストレージ仮想計算機環境における I/O スケジューラのチューニングに関する報告[17]などがある．

しかし，これらはフラッシュメモリや SSD を考慮したものではなく，近年普及しているスマートフォンにおける I/O スケジューラの考察に必ずしも有用ではない．

SSD の振る舞いを考慮した研究として，Wang らの評価[18][19]がある．これらでは，SSD の振る舞いやファイルシステム，アプリケーションの振る舞いを考慮して性能に関する深い考察を行っているが，トランザクション処理を主たる目標として，スマートフォンの I/O 性能について考察する本研究とは目的が異なる．

Linux カーネルをモニタリングするための既存研究として，カーネルモニタリングツールの FTrace[20][21]SystemTap[22][23]LTTng[24][25]，OProfile[26]がある．これらのツールを用いることにより，Linux カーネルの内部処理の観察が可能になる．しかし，それらは Linux 用に構築されているため Android の解析には適さず，MMC ドライバの解析ができない．また，我々の提案手法[6], [7]は，オーバーヘッドが非常に小さいが，アプリケーション起動時間に関する考察として，文献[27]や，電力と性能の考察として文献[28]がある．しかし，これらにおいては I/O 性能に関する詳細な考察はなされていない．

Android I/O 性能に関するものとしは以下のものがある．文献[29]において，Android OS における演算性能と I/O 性能の評価と解析がされており，現状では多くの例において Linux が Android よりも性能において優れていることが確認されている．また，文献[6]において，Android 端末上における RDBMS の問い合わせ処理性能が示され，そのボトルネックの発見方法の提案と性能向上手法に関する考察が行われている．これはら汎用的 I/O 性能の向上の考察として有益であるが，I/O スケジューリングに特化しておらずスケジューリングに関する考察はなされていない．

6. 考察

6.1 フラッシュメモリコントローラ

本稿では，フラッシュメモリにおける I/O 要求のアドレス差と応答時間の関係に着目し，I/O 性能の向上手法についての考察を行った．この両者の関係は，HDD の場合はその物的構造により決定され，ほぼすべての製品において同様のものとなると期待される．これに対してフラッシュメモリの場合は，両者の関係はフラッシュメモリのコントローラの実装にも大きく依存していると予想される．しかし，4 章で述べたように，2 台の端末においてほぼ同等の結果が得られており，現状や近い将来の多くのスマートフォンやタブレット端末においてはこれらの考察は有効であると考えられる．今後，フラッシュメモリコントローラに大きな変更が加えられた場合は，それらの端末に対して再度本稿と同様の考察を行う必要があると考えられるが，その様な場合でも本稿で用いた手法(アドレスソートの有無による性能の比較，MMC ドライバ層における観察)は有効であると考えている．

今後はフラッシュメモリコントローラにより多くの最適化が施され，さらに複雑な振る舞いをするようになることも考えられ，I/O スケジューラの最適化の重要性はますます高まっていくと期待される．

6.2 ファイルシステム

ファイルシステムの実装が本稿の実験結果に与えた影響について考察する．

図 4 から図 6 の計測は，ファイルシステム，ブロック層，I/O スケジューラより下層に行われているため，ファイルシステムなどの影響は受けておらず，他のファイルシステム実装などを用いて同一の結果が得られると考えられる．

一方，図 1 から図 3 および図 7 の計測はユーザ空間からファイルシステムを介して行われているため，ファイルシステムなどの実装の影響を受けていると予想される．特に図 1 の計測はブロック層の先読みの影響を受けており，他の OS を用いた場合は本稿と大きく異なる結果が得られる可能性が考えら得る．しかし，OS に Android を用いる場合はいずれのバージョンを用いてもブロック層や先読みの実装に大きな違いがないため，ほぼ同一の結果が得られると予想される．また，先読みと比較してその影響の程度は小さいと考えられるが，ファイルシステムの実装により巨大ファイルのシーケンシャルアクセスの性能やランダムアクセス性能に違いがあるためファイルシステムを変更すると図 1 の影響に変化が現れると予想される．

図 2 および図 3 の実験ではランダムアクセスを行っており，ファイルシステムのランダムアクセス性能の影響が含

まれていると予想される．たとえば，Nexus S で用いられている vfat は ext2 などの他のファイルシステムと比較してランダムアクセス性能が高くなく，ランダムアクセスがより高速なファイルシステムを用いた場合はすべてのすべての性能に向上がみられることが考えられる．その場合はデバイスの振る舞いの違いがより顕著に性能に影響を与え，本稿で行ってきた考察(デバイスの振る舞いを考慮したスケジューリング)がより明確に適用可能になると予想される．

図 7 の計測はファイルシステムを介して行っているので，ファイルシステムなどの実装の影響を受けていると思われる．しかし，本実験は Android のアプリケーションランタイムの影響も受けており，ファイルシステムなどの影響は図 4 のマイクロベンチマークの結果と比較して小さいと予想される．

6.3 カーネルモニタリングのオーバーヘッド

本稿の MMC ドライバ層における I/O 要求の観測は，著者らが開発したカーネルモニタリングツール[6]を用いて行われた．このモニタリングのオーバーヘッドを計測するため, モニタリング 1000 回に要する時間を計測したところ 850 マイクロ秒であり，モニタリング 1 回あたりのオーバーヘッドは約 0.85 マイクロ秒であった．これはフラッシュメモリアクセスにより生じる遅延時間と比較して十分に小さく，十分に正確にモニタリングできていると言える．

7. まとめ

本論文では，フラッシュメモリ搭載 Android 端末における I/O 性能基本性能の評価を行い，フラッシュメモリ搭載 Android 端末において，シーケンシャルリード性能がランダムリード性能よりも極めて高い性能であることを示した．次に，同じアドレス群をソートして読み込みを行った場合と，非ソートで読み込みを行った場合では，ソートをして読み込みを行った場合が高い性能を示したことから，フラッシュメモリ搭載 Android 端末において, I/O スケジューリングが効果を発揮することを示した．さらに，マイクロベンチマークによる I/O スケジューラの性能評価を行い，CFQ, DEADLINE, NOOP の中では DEADLINE が高い性能であることを示した．そしてその結果の解析を行い，フラッシュメモリ搭載 Android 端末では I/O 要求のアドレス差が正方向である場合の方が負方向である場合よりも応答時間が短く高性能であることを示した．

今後はフラッシュメモリ搭載 Android 端末に適したスケジューリングシステムについて考察していく予定である．

謝辞

本研究は JSPS 科研費 22700039, 24300034 の助成を受けたものである．

参考文献

1) コムスコア・ジャパン株式会社，日本における携帯電話利用者の最新状況に関するレポート
http://www.comscore.com/jpn/layout/set/popup/Insights/Press_Releases/2012/8/Japan_Smartphone_Surge
2) Android and iOS Surge to New Smartphone OS Record in Second Quarter, According to IDC
http://www.idc.com/getdoc.jsp?containerId=prUS23638712
3) What is Android?|Android Developers:
http://developer.android.com/guide/basics/what-is-android.html
4) Dalvik (software)
http://en.wikipedia.org/wiki/Dalvik_%28software%29
5) 新居健一，山口実靖，"I/O スケジューラの改善による仮想計算機環境における I/O 性能の向上",WebDB Forum 2011
6) 服部拓也 新居健一 山口実靖，“Android 端末 におけるデータベースのアクセス性能に関する考察”，分散，協調とモバイル(DICOMO2012)シンポジウム, pp.91-96(2012)
7) 三木香央理，山口実靖，小口正人，“Android 端末におけるカーネルモニタの導入”，第 22 回コンピュータシステム・シンポジウム，2010 年 11 月
8) S. Iyer and P. Druschel, ”Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O,” SOSP’01:Proceedings of the eighteenth ACM symposium on operationg systems principles. ACM, 2001.
9) Andrew Morton, ”Linux: Anticipatory I/O Scheduler,” http://kerneltrap.org/node/567
10) S. Pratt and D. Heger, ”Workload Dependent Performance Evaluation of the Linux 2.6 I/O Schedulers,” In Proceedings of the Linux Symposium, 2004
11) Jens Axboe, ”Linux Block IO - present and future,” In Proceedings of the Ottawa Linux Symposium, pp.51-61
12) P. Sheoy and H. Vin, ”Cello: A disk scheduling framework for next generation operating systems,” In Proceedings of ACM SIGMETRICS Conference, pp.44-555, 1997
13) Y. Zhang and B. Bhargava, ”Self-learning disk scheduling,” IEEE Trans, On knowl. And Data Eng.,21(1), pp.40-65, 2009
14) D. Boutcher and A. Chandra, ”Does Virtualization Make Disk Scheduling Passe?,”SOSP Workshop on Hot Topics in Storage and File System(Host Storage ’09)
15) M. Kesavan, A. Gavilovska and K. Schwa, ”On Disk I/O Scheduling in Virtual Machines,” WIOV’10, May 2010
16)Y. Xu and S. Jiang, ”A Scheduling Framework that Makes any Disk Schedulers Non.work.conserving solely based on Request Characteristics,” FAST2011
17)Kernel Virtual Machine (KVM) Best practices for KVM http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaat/liaatbestpractices pdf.pdf
18) Yongkun Wang, Kazuo Goda, Miyuki Nakano and Masaru Kitsuregawa,Performance Evaluation of Flash SSDs in a Transaction Processing System, IEICE TRANS. INF. & SYST., E94-D, 3, (2011.03)
19) Yongkun Wang, Kazuo Goda, Miyuki Nakano and Masaru Kitsuregawa,Early Experience and Evaluation of File Systems on SSD with Database Applications, Proceedings of the 5th IEEE International Conference on Networking, Architecture, and Storage (IEEE NAS2010), 467-476 (2010.07)
20) Steve Rostedt. ftrace tracing inftrastructure. http://lwn.net/Articles/270971/.
21) T. Bird, “Measuring Function Duration with Ftrace,” in Proc. of the Japan Linux Symposium, 2009
22) SystemTap http://sourceware.org/systemtap
23) Frank Ch. Eigler. “Problem solving with systemtap,” In Proceedings of the Ottawa Linux Symposium 2006, 2006
24) LTTng Project http://lttng.org/
25) M. Desnoyers, M. R. Dagenais, "The LTTng Tracer: A low impact performance and behavior monitor of GNU/Linux" Proceedings of Ottawa Linux Symposium 2006, 2006, pp. 209-223.
26) J. Levon and P. Elie. Oprofile: A system profiler for linux. http://oprofile.sf.net, September 2004.
27) Kyosuke Nagata, Saneyasu Yamaguchi, "An Android Application Launch Analyzing System ," 8th ICCM: 2012 International Conference on Computing Technology and Information Management,

ⓒ 2013 Information Processing Society of Japan

28) Nagata Kyosuke, Saneyasu Yamaguchi, Hisato Ogawa, "A Power Saving Method with Consideration of Performance in Android Terminals," The 9th IEEE International Conference on Autonomic and Trusted Computing (IEEE ATC 2012), ATC7-3, (2012/09/07)
29) 服部拓也 新居健一 山口実靖, “Android OS における演算性能と I/O 性能の評価と解析 マルチメディア”, 分散, 協調とモバイル(DICOMO2011)シンポジウム, pp.1399-1406 (2011)

ⓒ 2013 Information Processing Society of Japan