情報処理学会論文誌 Vol. 50 No. 2 691–700 (Feb. 2009)

キャリアグレードOSのためのディスクWrite処理方式

池邉 隆[†1] 内田直樹[†2]
平澤将一[†3] 本多弘樹[†3]

今日電話サービスに代表されるネットワークサービスのセッション制御を行う交換システムはATCAやCarrier Grade Linuxに代表されるコストパフォーマンスに優れるデファクトスタンダード製品を適用することが要望されている．デファクトスタンダード製品を適用した交換システムではサービス処理である呼処理の状態遷移にともない，各セッションの状態を同期Writeシステムコールを用いてハードディスクドライブに記録することが求められる．呼処理には実時間処理が求められ，この記録は高優先で処理されるが，IOアクセスが輻輳する場合，遅延する問題がある．これはLinuxのIO優先制御が十分でないために生じ，高優先度の同期Writeシステムコールの遅延はサービス品質を劣化させ，さらに著しい遅延はサービス障害をも引き起こす．本論文では，高優先の同期Writeシステムコールの遅延を短縮させるため，カーネルのIOスケジューラとファイルシステムのジャーナリング処理にIO優先度を考慮した処理を提案する．更に実験を通じて提案方式の有効性を示す．

A Study of Disk Write Mechanism for Carrier Grade Operating System

Takashi Ikebe,[†1] Naoki Uchida,[†2] Shoichi Hirasawa[†3] and Hiroki Honda[†3]

Today, call control servers use COTS (commercial off-the-shelf) components such as Advanced Telecom Computing Architecture (ATCA) and Carrier Grade Linux. However, the adoption of COTS components causes an increase in the maximum response time of high-priority write system call of recording call control transaction whenever IO operation is congested by normal-priority IO requests. Call control is a real-time operation; the delay of state transition makes service quality worse and sometimes causes fail-over of service. The delays may occur due to failure of software and hardware. However, even if there are no failures, the response time of high-priority write system call sporadic delays due to insufficient consideration about the IO prioritization by the OS. We present reduction of bottlenecks of the file system operation and IO scheduler mechanism in the kernel to reduce the maximum response time of high-priority write system call. The evaluation demonstrates that the presented approach shortens the maximum response time of high-priority write system call with sufficient reliability.

1. はじめに

今日電話サービスをはじめとし，様々なネットワークサービスが，IPネットワーク上で提供されている．この電話サービスはSIP[1]に代表されるセッション制御プロトコルを用い，交換システムによって提供される．この交換システムはOCAF[2]，SAF[3]，CGL[4]，ATCA[5]に代表される標準化アーキテクチャに準じた，コストパフォーマンスに優れるデファクトスタンダードな既製コンピュータの適用が要望される．

交換システムのサービス処理である呼処理は，処理上重要なデータを状態遷移にともないトランザクションログに記録する．このトランザクションログの記録は高優先度で実施され，一定の時間以内に終了される必要がある．既製コンピュータを交換システムに適用する場合，このトランザクションログをハードディスクドライブ（以降ディスクと呼ぶ）に記録することが求められる．しかしディスクへのIOアクセスが輻輳する場合，高優先で処理されるべきトランザクションログ記録のためのディスクIOアクセスが遅延する．

大半の処理がソフトリアルタイムである多くのエンタプライズシステムにおいて，この事象は問題とはならない．しかし交換システムはファームリアルタイムシステムであり，交換システムに適用されるキャリアグレードOSでは，この課題を克服することが求められる．

上記の課題を解決するために，本論文ではIO優先度を考慮したIOスケジューリングとファイルシステムのジャーナリング処理を行うことで，高優先の同期Writeシステムコールに要する最大レスポンスタイムを短縮する方式を提案し，実験を通じて提案手法の有効性を示す．なお既製OSとしてLinuxを，そのファイルシステムとして信頼性と実績に優れ

†1 日本電信電話株式会社NTTネットワークサービスシステム研究所
NTT Network Service Systems Labotatories, NIPPON TELEGRAPH AND TELEPHONE CORPORATION
†2 日本電信電話株式会社NTTサービスインテグレーション基盤研究所
NTT Service Integration Labotatories, NIPPON TELEGRAPH AND TELEPHONE CORPORATION
†3 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications

る EXT3[6] ファイルシステムを使用する．またハードウェアとして汎用マルチプロセッサである x86 アーキテクチャCPU の SMP を使用する．

2. 交換システムのディスクアクセスの要求条件

本章は，既製コンピュータを用いる交換システムのハードウェアの要件と，交換システムのディスクアクセスの要件を示す．

2.1 ハードウェアの要件

交換システムに適用される既製コンピュータは ATCA に代表される標準化アーキテクチャに従った，ブレード型サーバが使用され，典型的な ATCA の CPU ブレードは単一のディスクを実装する．本論文もこのシステム構成を前提とする．

2.2 同期 Write システムコールの要件

従来の交換システムは専用のデュプレックスシステムを用い，高優先で記録すべきトランザクションログはハードウェアレベルで冗長化された専用メモリに記録し，処理遅延の発生を最小限に抑えていた．しかし前述の既製コンピュータには専用メモリは存在しない．

トランザクションログは確実な記録のため，ディスクへの記録を行うことが要求される．さらにトランザクションログの記録は，ディスクに確実にデータが書き込まれるまで処理をブロックする同期 Write システムコールを用いることが要求される．

2.3 競合する Write 処理の要件

交換システム上ではトランザクションログを記録するための，高優先度の Write 処理，また，他の Write 処理として，バックアップ処理や異常系処理時のログ収集処理といった複数の通常優先度の Write 処理が同時に存在する．

特にバックアップ処理は，記録されたトランザクションログやシステムファイルを定期的に保存しており，トランザクションログ記録の Write 処理と同一ディスクに対するアクセスが競合する Write 処理となる．このような競合状況において，トランザクションログ記録のための高優先度の同期 Write システムコールが一定の時間以内に処理されることが要求される．このタイムアウト値はサービス仕様および処理内容によって異なるが，おおむね 100 ミリ秒がソフトリミットとして使用される．

交換システムはレスポンスタイムがソフトリミットを連続して何回か超過する場合，タイムアウトであるハードリミットを超過し，呼処理プロセスは該当のセッションが異常状態にあると判断し，該当のセッションをリセットする．さらにレスポンスタイムが 1 秒を超過するような場合，交換システムでは呼処理プロセス全体や，システム全体が異常な状態に陥っていると考え，より広範囲のソフトウェアの初期化・再実行や，待機系サーバへ処理を引き継ぐ等の障害処理が行われ，サービスが一時的に停止する．

一般に呼処理のトランザクションでは数百ミリ秒単位のタイムアウト値を使用する．本論文ではトランザクションログ記録のタイムアウト値として，3 回連続のソフトリミット超過である 300 ミリ秒を使用する．

本論文で対象とする交換システムには，40 万 BHCA（Busy Hour Call Attempt）程度の呼処理性能が求められる．これを実現するには，トランザクションログ記録のため 4 Kbyte 程度のデータを高優先の同期 Write システムコールを用いて，平均的に毎秒 111 回程度，レスポンスタイムのタイムアウト値以内で処理することが必要となる．

2.4 信頼性の要件

交換システムは年間 99.999%以上の可用性でサービスを提供することが求められる[7]．交換システムの個々の構成要素においても，十分な信頼性をもって処理を継続することが求められる．高優先の同期 Write システムコールのレスポンスタイムも上述のタイムアウト値を十分な信頼性をもって超過しないことが求められる．

3. 既存の Write 処理に関する研究動向

本章では，Linux カーネルの同期 Write システムコールの概要と，既存の Write 処理に関する研究と課題について示す．

既存の Write 処理を高速化する研究として，Trail[8),9)] に代表されるように用途ごとに複数のディスクを使い分ける方式が存在する．しかしこれら方式は交換システムに適用される標準的なアーキテクチャ，ハードウェア構成に沿わず，実システムへの適用は困難である．

標準的なアーキテクチャ，ハードウェア構成に合致した Write 処理を高速化する方式として，IO スケジューラを用いる研究[10)–15)] が行われており，Linux カーネル等，広く一般に使用されている．しかしこれら方式を交換システムに適用する場合，IO アクセスが輻輳する場合に遅延が発生する．この課題について以下で説明する．

またファイルシステムの処理では，同期 Write システムコールの処理時にファイルシステムの整合性を保つジャーナリング処理が行われる．しかしこの処理において IO 優先度を考慮した処理が行われず，IO アクセスが輻輳する場合に遅延が発生する．この課題についても以下で説明する．

3.1 Linux カーネルの同期 Write システムコールの処理概要

プロセスがディスクへ同期 Write システムコールを行う際のカーネル処理概要を説明す

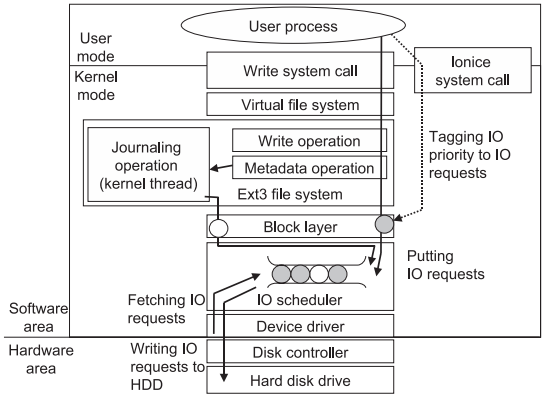

図 1　Linux の同期 Write システムコールの概要
Fig. 1　Abstract of synchronous write mechanism of Linux kernel.

る（**図 1**）．

プロセスは Write システムコールを同期モードで発行する．その後，同期 Write システムコールはファイルシステムの処理を呼び出す．EXT3 ファイルシステムでは，ファイルの更新と同時に信頼性を高めるジャーナリング処理を行う．

その後ブロックレイヤに処理を引き継ぐ．ブロックレイヤでは，書き込むべきデータを IO リクエストに変換し，IO リクエストを IO スケジューラのキューに登録し，ディスクコントローラのデバイスドライバに，ディスクへの記録を指示する．同期 Write システムコールを発行したプロセスはこの時点で IO Wait の CPU スケジューリング状態になり，IO リクエストがディスクに書き込まれるまで待合せを行う．

ディスクコントローラのデバイスドライバは割込み契機で動作し，IO スケジューラのキューの先頭から次に処理すべき IO リクエストを取り出し，ディスクへデータを記録する．ディスクへの記録が終了すれば，デバイスドライバは処理が終了したことをプロセスに通知し，同期 Write システムコールを発行したプロセスは待合せを終了し処理を継続する．

3.2　IO スケジューラの IO リクエスト順序制御

IO スケジューラは，まだデバイスドライバで処理されていない IO リクエストをキューイングする．このキューはディスクごとに存在する．Linux では No-operation（NOOP），Deadline[11]，Anticipatory[16]，Complete Fair Queuing（CFQ）の 4 つの IO スケジューラを選択でき，それぞれのポリシに従って IO リクエストが操作される．

- NOOP IO スケジューラ
  NOOP IO スケジューラはいっさいの IO リクエストの並べ替えを行わない．新たな IO リクエストはキューの先頭か末尾に追加され，キューの先頭から IO リクエストが処理される．
- Deadline IO スケジューラ
  Deadline IO スケジューラは IO リクエストが長い間処理されないことを防ぐことを目的とし，ディスクのセクタ順にソートされた IO リクエストのキューと，IO リクエストが登録された順にソートされた期限付きキューを持つ．
  通常時，ディスクのセクタ順にソートされたキューを処理することで，ディスクのシーク回数を削減する．もし期限付きキューで長時間処理されない IO リクエストが存在すれば，期限付きキューを優先的に処理する．
  この期限付きキューを処理する閾値は，Write の IO リクエストに対して 5 秒である．この期限付きキューはディスク単位で存在しており，IO リクエストの優先度を考慮した処理は行われない．
- Anticipatory IO スケジューラ
  Anticipatory IO スケジューラは Deadline IO スケジューラと同様にディスクのセクタ順にソートされた IO リクエストのキューと，IO リクエストが登録された順にソートされた期限付きキューを持つ．
  Anticipatory IO スケジューラの期限付きキューの閾値は，Write の IO リクエストに対しては 250 ミリ秒である．この期限付きキューは Deadline IO スケジューラ同様ディスク単位で存在しており，IO リクエストの優先度を考慮した処理は行われない．
- CFQ IO スケジューラ
  CFQ IO スケジューラはあらかじめいくつかのクラスに分かれた優先度に応じた IO リクエストのキューを有し，IO リクエストの優先度単位で IO スループットを割り当てるポリシを持つ．この IO 優先度の識別には ionice[17] による優先度割当てを使用する．
  CFQ IO スケジューラは，IO リクエストが IO スケジューラに登録された際にその IO リクエストを優先度ごとに存在するキューに移動する．次に，IO リクエストを処理するため，定期的に各キューに登録された IO リクエストを優先度に応じて重み付けられた一定時間，処理用のキューへラウンドロビンで移動する．
  一方，デバイスドライバは処理用のキューから IO リクエスト取り出しその処理を実行する．優先度に応じた一定の時間，各キューから処理用のキューへ IO リクエストが移

動されるが，この処理時間の長さによって，各キューの IO スループットが調整されることとなる．

CFQ IO スケジューラは IO 優先度を IO スループットの制御のために使用するが，レスポンスタイムに対して制御は行われない．

3.3　IO スケジューラの課題

IO スケジューラにおいて高優先の同期 Write システムコールが遅延する原因は，NOOP IO スケジューラを除いた他の IO スケジューラで共通的に，ディスク全体での IO スループットを向上させることを重視しており，交換システムのように複数の優先度の Write 処理のための IO リクエストが存在する状態で，高優先の同期 Write システムコールの遅延に対する考慮がされていないためである．

IO スループットを向上させる，既存の IO スケジューラの代表的な動作は，ディスクの連続するセクタへアクセスする複数の IO リクエストを 1 つの IO リクエストにマージ（結合処理）すること，またキューイングされている IO リクエストをアクセスするセクタ順にソート（順序制御）することである．この動作によりアクセス時に生じるシーク回数を削減する．

これらの処理は，IO 優先度ではなくディスクのセクタを考慮して処理されているため，**図 2** に示すように，新たに登録された IO リクエストが高優先であっても，この IO リクエストがアクセスするセクタが現在処理されている IO リクエストのセクタから離れた位置であれば，すぐには処理されない．そればかりか，現在処理されているセクタのより近傍へアクセスする低優先の IO リクエストが新たに登録された場合，この IO リクエストが先に処理されてしまう．

さらにマージ処理により，1 つの IO リクエストが有するブロックサイズがしばしば肥大化し，IO リクエストあたりのディスクでの処理時間が長くなり，遅延が発生する．

3.4　ファイルシステムでのジャーナリング処理

EXT3 ファイルシステムはジャーナリングファイルシステムであり，同期 Write システムコールの処理としてファイルの更新とジャーナルの更新を行う．ファイルの更新処理では，更新されたデータ部分であるデータブロックの記録を行い，ファイルサイズや，アクセスタイムスタンプ等の更新をメタデータである inode に記録する．その後，ファイルシステムはジャーナル情報が更新されるまで待合せを行う．この待合せによりクラッシュ時にもメタデータの整合性を保ち，ファイルシステムの信頼性を高める．

ジャーナルの更新処理ではパーティションごとに Kjournald カーネルスレッドを生成し，

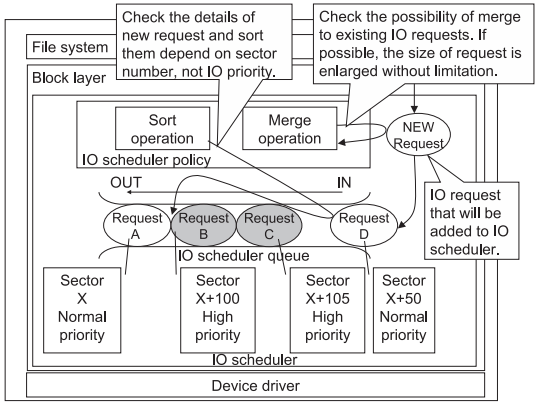

図 2　IO スケジューラの問題
Fig. 2　The problem of IO scheduler.

規定の動作では各ファイルのメタデータ更新をジャーナルデータとしてシーケンシャルにディスクへ書き込む．

3.5　ファイルシステムの課題

同期 Write システムコールの遅延は IO スケジューラだけではなく，ファイルシステムにおいても生じる．具体的には EXT3 ファイルシステムの，ジャーナル記録処理を行う Kjournald カーネルスレッドの実行がユーザプロセスの実行と独立していることに原因がある．

Kjournald はカーネルスレッドであり，同期 Write システムコールを発行したユーザプロセスの実行とは独立したタスクとして実行される．すなわち Kjournald はユーザプロセスの IO 優先度とは関係なく処理を行い，あらゆるジャーナルの更新を通常の IO 優先度を使用してディスクへ書き込む．一方，ファイルシステムが同期 Write システムコールに関する処理を終了するためには，ジャーナル更新の完了を待つ必要がある．

図 3 に示すように，ユーザプロセスが高優先の同期 Write システムコールを発行している場合，データブロックの更新処理は高優先度で処理されるが，これにともなうメタデータ更新のためのジャーナル更新を行う Kjournald カーネルスレッドはこの優先度を認識することができないため，当該ジャーナル更新は通常優先度で処理されることとなってしまう．このため，IO アクセスが輻輳する場合，ジャーナル更新には一定の処理時間を要することとなり，長時間待合せが発生し，高優先の同期 Write システムコールが遅延してしまう．

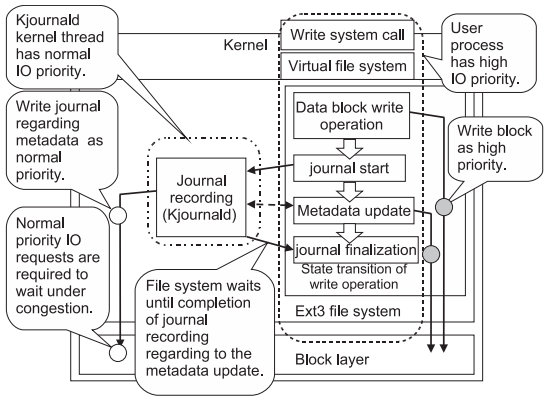

図 3　ファイルシステムのジャーナル処理における問題
Fig. 3　The problem of journaling mechnism of filesystem.

4. 提 案 方 式

本章では上述した遅延を改善する方式を提案し，実装について説明する．

4.1　IO スケジューラでの優先度処理

筆者は IO スケジューラでディスクのセクタ順ではなく，IO 優先度を第 1 の優先順位としてソートし，同一優先度間では FIFO でソートする IO スケジューラの処理を提案する．提案方式により IO スケジューラはつねに最も優先度が高い IO リクエストを，登録された順にデバイスドライバに渡すことが可能となる．

さらに提案方式では，IO 優先度が同一の IO リクエストに限って IO リクエストのマージを許容するとともに，マージ回数上限値を設け，IO 優先度が高優先でない場合にはマージ可能な回数を制限することとしている．これによりディスクの IO スループットが向上する範囲内でのマージを可能とするとともに，IO スループット向上に貢献しない不必要なマージを制限することができる．これにより高優先の IO リクエストのレスポンスタイムがタイムアウト値を超えないようにしつつ，全体の IO スループットを向上させることができる．

なお本値は，高優先の IO リクエストと低優先の IO リクエストを競合させた試験等により，ディスクの IO スループットが向上する範囲で，高優先の IO リクエストのレスポンスタイムがタイムアウト以下に抑えられるよう設定されるべきと考える．

4.2　ファイルシステムでの優先度処理

筆者はファイルシステムの課題を解決するために Kjournald の処理に，同期 Write システムコールを発行したユーザプロセスの IO 優先度を一定区間引き継ぐことで，Kjournald とユーザプロセスの IO 優先度を同一とすることを提案する．

具体的には，ユーザプロセスの IO 優先度をジャーナル記録開始時に EXT3 ファイルシステムの Kjournald カーネルスレッドへ通知する．その後 Kjournald カーネルスレッドは，与えられた情報を自身の IO 優先度として設定し，ジャーナリング処理を開始する．ジャーナル処理終了時には EXT3 ファイルシステムがジャーナル記録の待合せを終了する際に，設定した IO 優先度をリセットする．

4.3　提案方式の実装

提案方式の IO スケジューラを，すでに IO 優先度の概念を持つ CFQ IO スケジューラをベースに新たな IO スケジューラとして実装を行った．この際，提案方式の実現性を確認するため最も簡易な実装である，単一の IO リクエストキューを用い，IO リクエストの登録ごとに優先度かつ，同一優先度間では登録順である FIFO となるようソートを行った．また非高優先度の IO リクエストに対してはマージ回数上限値を超過するマージは行わない処理とした．

IO スケジューラのマージ処理部では cfq_merge() 処理を変更し，非高優先度の IO リクエストに対して設定された一定回数を超過するマージを行わない処理とした．またキューイング部分においては，cfq_insert_request() 処理において，高優先の IO リクエストが登録される場合に，CFQ のプロセス・優先度別のキューではなく，優先度・登録順にソートして IO リクエストキューに登録する処理に変更した．

また EXT3 ファイルシステムの Kjournald への IO 優先度通知のため，ジャーナル記録開始処理である journal_start 関数の引数に渡すパラメータである journal 構造体のメンバとして，同期 Write システムコールを発行したプロセスの IO 優先度を追加し，journal_start 関数を呼び出す際に IO 優先度を設定する処理とした．

Kjournald カーネルスレッド側では主処理であるジャーナルの記録を journal_commit_transaction 関数のループで行う．この journal_commit_transaction 関数の先頭で journal 構造体の IO 優先度のメンバをチェックし，設定された IO 優先度を引き継ぎ，ジャーナルデータの更新処理を行う．

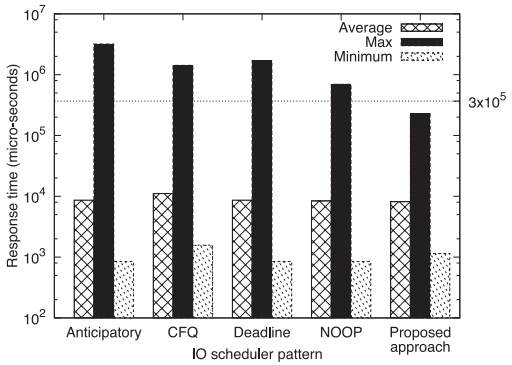

図 4　高優先の同期 Write システムコールを使用した Write システムコールのレスポンスタイム
Fig. 4　Response time of Write system calls tagged as high priority.

5. 実験による提案方式の評価

提案方式による高優先の同期 Write システムコールのレスポンスタイムの計測結果を**図 4** に示す．本実験では計測用の擬似呼処理プロセスと，Linux カーネルのバージョン 2.6.21 を使用し，既存の 4 つの IO スケジューラと提案方式に対して計測を行った．評価環境を**表 1** に示す．

呼処理プログラムのトランザクションログの処理では，呼処理上の状態遷移の中でも課金にかかわるような特に重要な状態遷移の状態をディスクへ記録する．評価に用いた擬似呼処理プロセスは，この記録処理部分のみを抽出したもので，高優先の同期 Write システムコールを用いて 4 Kbyte データの書き込みを，45 万 BHCA 程度の呼処理に相当するよう連続的に実施するものである．ハードウェア構成，サービス種別にも依存するが，本論文で対象とする交換システムには，前述のとおり 40 万 BHCA 程度の処理性能が求められ，性能評価に用いる負荷としては十分であると考える．

また，背景負荷として RSYNC[18] によるバックアップ処理を使用した．RSYNC は一般に広く使用されているバックアップ処理であり，本論文で対象とする交換システムにおいても使用されている．交換システムでは十分なメモリを搭載し，呼処理中にトランザクションログ記録以外のディスクアクセスが不用意に発生しないよう，メモリ設計，プログラム設計が行われている．トランザクションログ記録と競合するディスクアクセスは，バックアップ処理時のディスクアクセス，各種プログラムの走行ログ記録のためのディスクアクセス，オペレーション契機によるユーザデータの取得・変更処理等である．これら競合するディスクアクセスは，バックアップ処理に比べると，その他のディスクアクセスは，きわめて微小なディスクアクセスであり，呼処理によるディスクアクセスへの影響は少ない．

表 1　評価環境
Table 1　Evaluation environment.

CPU	Pentium Xeon 2.8 GHz ×2
Memory	2 GB
ディスク	160 GB SATA-1，7200 RPM，8 MBytes キャッシュ，同期マウント
ディスクコントローラ	Intel 6300ESB SATA
カーネル	2.6.21
マージ可能な回数	16
計測対象	擬似呼処理プロセスが発行する Write システムコールのレスポンスタイム．4 KByte 単位の連続した高優先度の同期 Write システムコールを実施．
背景負荷	RSYNC により通常 IO 優先度でシステムデータのバックアップを実施．
計測回数	100 万回計測を実施

なお本評価ではマージ回数上限値を 16 とした．この値は評価に先立ち次の計測を行うことにより求めた．すなわち，擬似呼処理プロセスとバックアップ処理プロセスが稼動している状況でマージ回数上限値とディスクの IO スループット，高優先 IO リクエストのレスポンスタイムの関係を計測し，マージ回数上限値を大きくしてもそれ以上 IO スループットが向上しなくなり，かつ，高優先 IO リクエストのレスポンスタイムが，タイムアウト以下である点を求めた．本評価環境ではマージ回数上限値が 16 の時点で IO スループットの向上がおおむね飽和し，高優先 IO リクエストのレスポンスタイムはタイムアウト値以下であった．

図 4 の X 軸は IO スケジューラのパターンを示し，Y 軸は高優先の同期 Write システムコールのレスポンスタイムをログスケールで示している．本結果から既存の IO スケジューラでは最大レスポンスタイムがきわめて大きく，高優先の同期 Write システムコールが遅延していることが分かる．それぞれの最大値は Anticipatory IO スケジューラで約 3.2 秒，CFQ IO スケジューラで約 1.4 秒，Deadline IO スケジューラで約 1 秒，NOOP IO スケジューラで約 700 ミリ秒を要しており，タイムアウト値を大きく超過している．

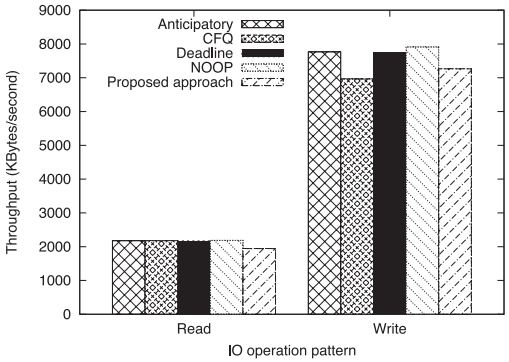

図 5　実験中の平均 IO スループット
Fig. 5　IO throughput average during experiment.

一方，提案方式の最大レスポンスタイムは 220 ミリ秒程度である．この値はタイムアウト値以内に収まっており，かつ他の IO スケジューラより最大で 14 分の 1 に短縮できていることが分かる．平均レスポンスタイムはいずれの IO スケジューラでもおおよそ 8 ミリ秒程度である．また最小レスポンスタイムはいずれの IO スケジューラでもおおよそ 1 ms 程度である．

また，計測中のディスクの IO スループットを**図 5** に示す．本結果は計測中の毎秒の各オペレーションの値の平均値を示している．X 軸は処理種別を示し，Y 軸は IO スループットを示す．提案方式においてはおおよそ 7.3 MBytes/second の Write 処理の IO スループットを達成しているが，この値は Anticipatory，Deadline，NOOP IO スケジューラより遅い値である．一方 CFQ IO スケジューラよりは高速な値である．読み込み処理はおおよそ 2 MBytes/second であり，他の IO スケジューラよりも 10%程度低下している．

6. 考　　察

提案方式の高優先の同期 Write システムコールの平均レスポンスタイムはおおむね 8 ミリ秒であり，これは擬似呼処理プロセスからの Write 処理は 500 KBytes/second で実施されていることを示す．一方，計測中の Write 処理の IO スループットはおおむね 7.3 MBytes/second であり，大半の Write 処理はバックアップ処理によって実施されていることが分かる．

NOOP IO スケジューラは既存の IO スケジューラの中では最も高速な最大レスポンスタイムである 700 ミリ秒を記録している．NOOP IO スケジューラは IO 優先度を考慮しないが，特別なマージ処理およびソート処理をいっさい行わない．このため高優先の IO リクエストは，バックアップに比べて Write 処理量が少ない分の，相対的な遅延が生じているだけにとどまっていると考える．

ディスクのセクタ順のキューを持つ Anticipatory および Deadline IO スケジューラでは，特にバックアップからの Write 処理が多数を占めるような場合，いったんバックアップ処理の IO リクエストが処理されると，相対的に Write 処理量が少ないことによる遅延に加えて，新たに登録されるバックアップ処理の IO リクエストが高優先の IO リクエストより先に処理され，高優先の同期 Write システムコールが遅延すると考える．

さらにこれら IO スケジューラの期限付きキューは IO 優先度の考慮がないため，大半の Write 処理が通常 IO 優先度のバックアップから生成される状況では，高優先の IO リクエストが期限付きキューの後半につながれ，遅延の短縮につながらないと考える．

CFQ IO スケジューラでは IO 優先度を使用しているが，最大レスポンスタイムは大きく，1.4 秒ほどを要している．CFQ IO スケジューラは，優先度に応じたキュー単位で一定時間 IO リクエストのキューイング処理を行い IO スループットの調整を行うため，IO リクエスト単位での処理が遅延すると考える．

提案方式では，既存の方式に比べて最大レスポンスタイムを 220 ミリ秒と大幅に短縮することを可能としている．これは提案方式が高優先の IO リクエストを優先的に処理し，デバイスドライバが高優先の IO リクエストを先に処理することに成功していると考える．

提案方式の Read 処理の IO スループットは他の IO スケジューラに比べて若干低い．これは提案方式では高優先の同期 Write システムコールによる IO リクエストを優先的に処理するためと考える．他の IO スケジューラでは Read 処理と Write 処理の IO リクエストが同等に IO リクエストキューにキューイングされるが，提案方式では高優先度の Write 処理に関する IO リクエストを優先的に IO リクエストキューの先頭にキューイングするため，相対的に Read 処理の IO リクエストが処理される頻度が低下し，結果的に Read 処理の IO スループットが低下したと考える．

一方，Read 処理，Write 処理の IO スループットを合算した値では，Anticipatory，Deadline IO スケジューラは 9.9 MBytes/second 程度，CFQ IO スケジューラでは 9.1 MBytes/second 程度，提案方式では 9.2 MBytes/second 程度を記録しており，提案方式の総合的な IO スループットが他方式と比べて著しく劣るわけではない．

また提案方式では 1 つ 1 つのデータアクセスを行うセクタが，他方式に比べて連続ではなくシークが多数発生する．このため IO スループットの向上には不利なアクセス方法となる．

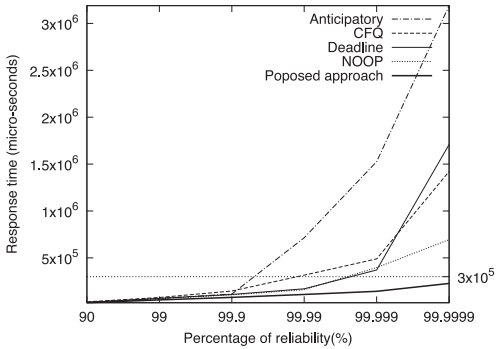

図 6　レスポンスタイムのパーセンタイルグラフ
Fig. 6　Percentile graph of responses.

しかし交換システムに代表される実時間処理を重要視するシステムでは，IO スループットよりも安定的なレスポンスタイムの方が重要である．このような要求条件を持つシステムには，提案方式が有用であると考える．

また実験結果である 100 万回のレスポンスタイムの結果の分布をパーセンタイルグラフで**図 6** に示す．X 軸はレスポンスタイムが一定の時間以内に収まる信頼性のパーセントを示し，Y 軸はレスポンスタイムを示す．図 6 のグラフは提案方式による最大レスポンスタイムが 300 ミリ秒のタイムアウト値以下であることを 99.9999%の信頼性を持っており，他方式に比べて十分な信頼性を持っていることが分かる．

交換システムでは呼処理にともなうトランザクションログ記録に競合するディスクアクセスとして，バックアップ処理，各種プログラムの走行ログ記録，オペレーション契機でのユーザデータの取得・変更処理等が実施される．本論文で示した手法により呼処理のリアルタイム性を向上させることが可能となるが，他のディスクアクセス処理のレスポンス性能は悪化する．交換システムでは，他のディスクアクセスが遅延してもシステムとして致命的な問題にならないよう，運用上の対処がなされている．たとえばオペレーション契機の作業は，呼処理の最繁時間帯を避け，場合によっては夜間に作業を実施し，呼処理によりこれら処理の遅延が少なくなるよう，またこれら処理が呼処理に影響が最小限になるよう考慮されている．

7. おわりに

交換システムは実時間処理を必要とする．交換システムに既製コンピュータを適用する場合，呼処理のトランザクションログをディスクへ高優先で同期 Write システムコールを用いて記録し，このレスポンスタイムがタイムアウト値以内に安定的に収まることが要求される．

既製 OS である Linux では呼処理による Write 処理と，他の Write 処理が競合する場合，呼処理から発行された高優先度の同期 Write システムコールが遅延する．これは Linux カーネルや既存の研究においては，ディスクの IO スループットを最大化することを重視しており，IO の優先度に応じた処理遅延を考慮できていないためである．

交換システムに適用されるキャリアグレード OS ではこの課題を解決すべきであり，本論文ではこの課題を IO スケジューラとファイルシステムで IO 優先度に応じた処理を行う方式を提案した．また実験を行い，提案方式は高優先度の同期 Write システムコールの最大レスポンスタイムを十分な信頼性を保って，既存の方式に比べて最大で 14 分の 1 に短縮できることを示した．

今後の課題として，IO スケジューラのキューイング処理の改善と他ファイルシステムへの対応があげられる．提案方式の IO スケジューラ部分の実装は簡易化のため単一のキューを使用し優先度と登録順によるソート処理を行っている．本実装ではキューに多くの IO リクエストがキューイングされる際，ソート処理時間に課題があると考える．また現在の実装では EXT3 ファイルシステムのみにしか対応しておらず，他ファイルシステムへの対応も重要である．

謝辞　本研究を進めるにあたって検討にご協力いただいた日本電信電話株式会社，日野村聡氏，野口昌彦氏，岩田哲弥氏，河原崎裕朗氏，中野宏朗氏，NTT コムウェア株式会社，森下徹氏に感謝いたします．

参考文献

1) Rosenberg, J., et al.: SIP: Session Initiation Protocol, RFC3261 (June 2002). http://www.ietf.org/rfc/rfc3261.txt?number=3261
2) ITU-T: Open Communications Architecture Forum (OCAF) Focus Group. http://www.itu.int/ITU-T/ocaf/
3) Service Availability Forum. http://www.saforum.org/home
4) Linux Foundation: Carrier Grade Linux. http://www.linux-foundation.org/en/

Carrier_Grade_Linux
5) PCI Industrial Computer Manufacturers Group: Advanced TCA PICMG 3.0 Short Form Specification (Jan. 2003). http://www.advancedtca.org/
6) Theodore, Y. Ts'o. and Tweedie, S.: Planned Extensions to the Linux Ext2/Ext3 Filesystem, *Proc. 2002 Usenix Technical Conference*, FREENIX Track (2002).
7) Serlin, O.（著），渡辺榮一（編）：フォールト・トレラント・システム，pp.33–56, マグロウヒル出版，東京 (1990).
8) Chiueh, T. and Huang, L.: Trail: A Fast Synchronous Write Disk Subsystem Using Track-Based Logging, Technical Report, Paper ID:360, Computer Science Department State University of New York at Stony Brook (1999).
9) Chiueh, T.: Trail: A Track-Based Logging Disk Architecture for Zero-Overhead Writes, *Proc. 1993 IEEE International Conference on Computer Design*, Cambridge, pp.339–3443 (1993).
10) Bruno, J., et al.: Disk scheduling with quality of service guarantees, *IEEE ICMCS* (June 1999).
11) Chen, S., et al.: Performance evaluation of two new disk scheduling algorithms for real-time systems, *Journal of Real-Time Systems*, Vol.3, No.3, pp.307–336 (Sep. 1991).
12) Golubchik, L., et al.: Evaluation of tradeoffs in resource management techniques for multimedia storage servers, *IEEE ICMCS* (June 1999).
13) Huang, L. and Chiueh, T.: Implementation of a rotation latency sensitive disk scheduler, Technical Report ECSL-TR81, SUNY, Stony Brook (Mar. 2000).
14) Iyer, S. and Druschel, P.: The effect of deceptive idleness on disk schedulers, Technical Report CSTR01-379, Rice University (June 2001).
15) Jacobson, D. and Wilkes, J.: Disk scheduling algorithms based on rotational position, Technical Report HPL-CSP-91-7rev1, Hewlett-Packard (Feb. 1991).
16) Iyer, S. and Druschel, P.: Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O, *SOSP 2001* (Aug. 2001).
17) Axboe, J.: ionice(1)—Linux man page. http://linux.die.net/man/1/ionice
18) samba project: rsync. http://rsync.samba.org/

(平成 20 年 4 月 24 日受付)
(平成 20 年 11 月 5 日採録)

池邉　隆

1977 年生まれ．2000 年電気通信大学電気通信学部電子工学科卒業．2002 年電気通信大学大学院情報システム学研究科修士課程修了．2008 年電気通信大学情報システム学研究科博士課程修了．2002 年日本電信電話（株）入社．以来，ネットワークサービスシステムの研究開発に従事．現在，日本電信電話（株）ネットワークサービスシステム研究所所属，電子情報通信学会会員．博士（工学）．

内田　直樹

1959 年生まれ．1985 年電気通信大学大学院電気通信学研究科修士課程修了．同年日本電信電話（株）入社．以来，ネットワークサービスシステムの研究開発に従事．2004〜2005 年電子情報通信信学会通信ソサイエティ総務幹事．現在，日本電信電話（株）サービスインテグレーション基盤研究所主席研究員．（工学博士）．

平澤　将一（正会員）

2000 年東京大学理学部情報科学科卒業．2002 年同大学大学院理学系研究科情報科学専攻修了．2005 年同大学大学院情報理工学系研究科コンピュータ科学専攻博士課程単位取得退学．2006 年電気通信大学大学院情報システム学研究科助手．2007 年同助教．高性能計算システム，プログラミング言語処理等．ACM，IEEE CS 各会員．

本多　弘樹（正会員）

1984 年早稲田大学理工学部電気工学科卒業．1991 年同大学大学院理工学研究科博士課程修了．1987 年より同大学情報科学研究教育センター助手．1991 年より山梨大学工学部電子情報工学科専任講師，1992 年より同助教授．1997 年より電気通信大学大学院情報システム学研究科助教授．2007 年より同教授．並列処理方式，並列化コンパイラ，並列計算機アーキテクチャ，グリッド等の研究に従事．工学博士．IEEE-CS，ACM 各会員．平成 15 年度情報処理学会山下記念研究賞受賞．