ファイル受信の検知を用いたバックアップの中断による ファイル送信時間の増加の抑制

高橋 風太[1]　串田 高幸[1]

概要：ファイルサーバがユーザからのファイル受信と同時にバックアップサーバへのバックアップを行う場合，ハードディスクが書き込み速度の上限に達する．ここでの課題は，ユーザのファイル送信時間が増加することである．本研究の提案手法では，ファイルサーバへのファイル受信を検知しバックアップを一時的に中断させることで，ハードディスクが書き込み上限に達することを抑制する． 評価として，提案手法を用いずにバックアップしたものと，提案手法によるバックアップとのファイル送信が完了するまでの時間を比較する．結果として，10GB の動画ファイルにおいて，提案手法を用いない場合と比較して，提案手法を用いたファイル送信時間は 24 秒減少した．ファイル送信時間の増加を抑制できたことから，提案手法によるサーバの書き込み速度の低下の抑制が期待できる．

1. はじめに

背景

今日，安全保管サービスを必要とする電子形式の大量のデータが生成されており，デジタルデータの重要性と社会的価値が継続的に高まっているため，データのバックアップとディザスタリカバリの問題がネットワークの基本になりつつある [1]．既存の自動バックアップサービスではバックアップ間隔を曜日単位で指定できるが，具体的なバックアップ時刻は明確にされていない．毎日行われるバックアップの例では増分バックアップによる方法を用いており，初期バックアップ後，ネットワーク帯域幅への負荷を軽減し，データバックアップの速度を向上させている．データトラフィックを削減することも，サービスエクスペリエンスを向上させるための重要な手段である [2]．OIS[*1]のイベント派生エンジンのパフォーマンス要件には，99.99%の平均可用性が含まれている [3]．よって，サービスを継続したままバックアップが行われる企業や組織が存在する．例として，デルタ航空会社が運営する OIS が存在する．

課題

ファイルサーバがユーザからのファイル受信と同時にバックアップを行う場合， ハードディスクが書き込み速度の上限に達しユーザのファイル送信が完了するまでの時間が増加することが課題である．Jaguar スーパーコンピュータの場合では，I/O 負荷の最大 75%が書き込みであると推定されており，CPU リソースは十分に活用されていない [4]．バックアップ時間を短縮させつつファイルを受信する際のバックアップにより発生するハードディスクの書き込み回数を減少させる方法として増分バックアップが存在する．しかし，この方法では動画ファイルをバックアップの対象とする場合は定期的に一定量更新されるデータベースとは性質が異なり， 一度にバックアップする容量が一定でない．そのため，動画ファイルを対象とするバックアップでは，ハードディスクの書き込み処理を軽減できない場合がある．図 1 に，ファイルサーバのバックアップとファイル受信による書き込み速度の低下を示す．

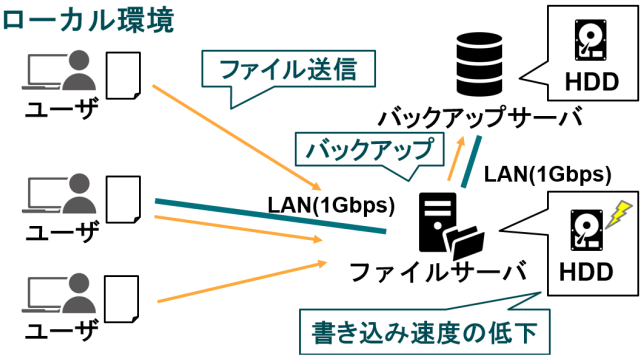

図 1　ファイルサーバのバックアップとファイル受信による書き込み速度の低下

図 1 では，ユーザがファイルサーバへファイルを送信

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1

[*1] 企業または組織の日常業務を継続的にサポートする大規模な分散システム

し，ファイルサーバがバックアップサーバへバックアップしている．通信には 1Gbps の有線 LAN を使用し，ファイルサーバ及びバックアップサーバの処理はハードディスク (HDD) により行われる．ファイルサーバはユーザからファイルを受信しつつバックアップを行うことになるため，書き込み処理が重複する．これにより，ファイルサーバの書き込み速度は低下する．

各章の概要

第 2 章では関連研究について述べる． 第 3 章では提案方式について述べる． 第 4 章では提案方式をもとに開発したシステムの実装と実験方法について述べる． 第 5 章では本研究の提案手法の評価手法と分析手法について述べる．第 6 章では第 5 章で述べた各手法に関する議論について述べる．第 7 章では本研究の結果から得られた成果とまとめについて述べる．

2. 関連研究

サービスの I/O パフォーマンスを保証する研究がある．この研究では，アプリケーションユーザが I/O アクセスとストレージスペースの事前の時間ベースの予約を明示的に行える分散ストレージシステムを提案している [5]．アプリケーションユーザが後でアクセスできるように開始時間と終了時間のパラメーターを使用して，ストレージシステムの望ましいパフォーマンスを事前に明示的に予約できるようにする分散ストレージシステムにすることで，予約された時間中のスループットを保証するものである．しかし，バックアップにおいてはユーザがストレージの開始時間と終了時間を把握できないため，パフォーマンスが保証が為されるものではない．

EC2 (Elastic Compute Cloud) において，仮想 I/O ワークロード間の競合によって引き起こされる新しいタイプのセキュリティの脆弱性を示すことを課題とする研究がある [6]．ハードドライブのスループットやネットワーク帯域幅といった，データ集約型のアプリケーションにとって重要な I/O リソースに重点を置いており，パフォーマンスの低下はワークロードあたりのコストを直接増加させるとしている．この課題に対して，ワークロードを使用して，最小限のリソース消費で対象のアプリケーションと仮想マシン (VM) に大幅な遅延を発生させるフレームワークを設計および実装している．この研究により，I/O のパフォーマンス低下が現実に起こり得ることが示され，コストの増加につながることが示されているといえる．しかし，この研究はサーバがクラウドにあることを前提にしているため，ローカル環境での課題の解決はできない．

仮想マシンのパフォーマンス干渉に対するハイパーバイザー I/O スケジューリングの影響についての研究がある [7]．この研究では，物理マシンを異なるテナントが所有する複数の仮想マシン（VM）で共有できるようにする仮想化技術において，悪意のあるテナントによるサイドチャネル攻撃やパフォーマンス干渉攻撃への扉が開かれる可能性を指摘している．I/O ベースのパフォーマンス攻撃は，特にデータ集約型アプリケーションの場合，依然として大きな脅威であるとしており，適切に設計された測定フレームワークが仮想 I/O スケジューリングの研究に役立ち，そのような知識を潜在的に適用して，基盤となる I/O リソースの使用法を活用できることを示している．この研究から，I/O のパフォーマンスの重要性が示されている．しかし，本研究では VM への攻撃を前提にしたものではないため，この研究の提案で本研究の課題を解決できない．

仮想化システムにおける I/O の予測可能性の定量化と改善についての研究がある [8]．この研究では，多くのユーザーが一貫したパフォーマンスを望んでいる可能性があり，IaaS プロバイダーは，既存の（予測可能性を無視する）サービスに加えて，予測可能なパフォーマンスのサービスのクラスを提供する必要があると主張している．そして，予測可能な VM パフォーマンスを提供するストレージシステムである VirtualFence を提案している．また，ストレージ I/O の予測可能性は，主にハードディスクドライブ（HDD）の機械的な制限のために，達成するのが最も困難であるとしている．この研究から，本研究における課題であるファイルサーバのパフォーマンス低下が，ユーザへのサービスのパフォーマンスの非一貫性を招くことに影響していることが示される．また，本研究の課題の，HDD のストレージ I/O の予測による解決が困難であることを示している．

クラスタの一元化されたファイルサーバのための新しいヒントベースの I/O メカニズムについての研究がある [9]．この研究では，中小規模のクラスタシステムでの集中型ファイルサーバにおいて，複数の並列アプリケーションが同時に共有ストレージにアクセスすると，異なるクライアントからの I/O 要求の干渉により，I/O パフォーマンスが大幅に低下することを課題としている．この研究の功績として，3 つの項目を挙げている．1 つ目は，並列要求のヒント情報によってファイルサーバのパフォーマンスを向上させるために，集中型ファイルサーバの必要性を提示していることである．2 つ目は，United-FS では，2 つのレベルのメカニズムが設計および実装されており，ディスク上のデータの局所性を最適化するために，書き換えメカニズムが提案されていることである．3 つ目は，提案された I/O メカニズムは，クライアント側のコードを変更することなく，NFS を含む任意の集中型ファイルサーバで採用できることである．この研究から，ファイルサーバのパフォーマンスの重要性が示されている．しかし，この研究は並列アプリケーションを前提にした提案であるため，本研究の課題を解決することはできない．

並列ファイルシステムのためのサーバレベルの適応型データレイアウト戦略についての研究がある [10]．この研究では，並列ファイルシステムは，高度な I/O 並列処理を提供して，I/O とメモリ速度の間のギャップを隠すために広く使用されているが，アプリケーションのデータアクセスパターンが複雑なため，ピーク I/O パフォーマンスが達成される頻度が少ないことを課題としている．提案では，小さなリクエストの I/O パフォーマンスはリクエストサービスレートによって制限されることが多く，大きなリクエストのパフォーマンスは I/O 帯域幅によって制限されるという観察に基づいて，両方の要因を考慮し，サーバレベルの適応データを提案している．この研究から，ファイルサーバの I/O パフォーマンスは理論値を出す頻度が低く，パフォーマンス向上が重要であることが示されている．しかし，この研究はへ入れるファイルシステムを前提にした提案であるため，本研究の課題を解決することにはつながらない．

3. 提案方式

基礎実験

基礎実験の実験環境には仮想マシン（VM）を用いる．VM の作成には ESXi を用いる [11]．ユーザ，ファイルサーバ，バックアップサーバをそれぞれ異なる VM として作成する．図 2 に実験環境の全体図を示す．

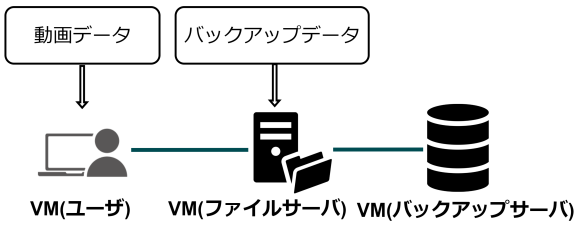

図 2 実験環境の全体図

ユーザが送信する動画データとバックアップデータの両方には，100MB の容量で MP4 形式の動画ファイルを 10GB になるように 102 個複製したものを 1 つのディレクトリに入れたものを用いる．本研究で行う実験で用いる動画ファイルには，Youtube 上にアップロードされているクリエイティブ・コモンズ・ライセンスが適用されている動画を用いる[*2]．この動画の容量は 100MB であったが，提案内容が活用されるユースケースはより長く容量の大きい動画を想定するため，今回の基礎実験では，1 時間 12 秒で 10GB の容量である IEEE CCEM 2021 Student Project Showcase の録画を参考にした．よって，100MB の動画が 10GB になるように複製している．動画データはユーザとして扱う VM 内に移動し，バックアップデータはファイルサーバとして扱う VM 内に複製する．実験では，動画データはファイルサーバに送信し，バックアップデータはバックアップサーバとして扱う VM に送信する．下記に，基礎実験に用いる各 VM の構成情報を示す．

- VM 構成情報 (ユーザ)
  OS Ubuntu-20.04.2
  vCPU 1 コア
  RAM 1GB
  HDD 128GB

- VM 構成情報 (ファイルサーバ)
  OS Ubuntu-20.04.2
  vCPU 1 コア
  RAM 1GB
  HDD 64GB

- VM 構成情報 (バックアップサーバ)
  OS Ubuntu-20.04.2
  vCPU 1 コア
  RAM 2GB
  HDD 64GB

基礎実験では，2 つの方法を比較することでバックアップによるファイル送信完了までの時間の増加を確認する．2 つの方法をそれぞれ T_1 と T_2 とする．図 3 に，ファイル送信完了までの時間の比較のグラフを示す．

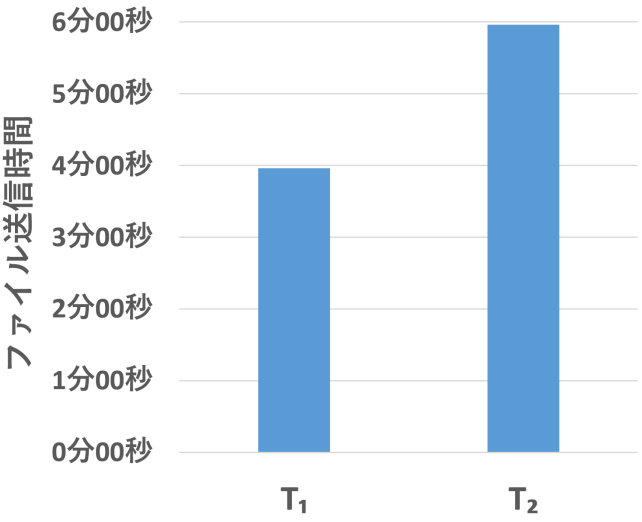

図 3 ファイル送信完了までの時間の比較

下記に，T_1 と T_2 の具体的な内容を示す．

T_1 ユーザからファイルサーバへファイルの送信を行う際の，ファイル送信完了までの時間

[*2] 「Big Buck Bunny 60fps 4K - Official Blender Foundation Short Film」https://www.youtube.com/watch?v=aqz-KE-bpKQ

T_2　ユーザからファイルサーバへのファイルの送信と同時にファイルサーバからバックアップサーバへバックアップを行う際の，ユーザとファイルサーバ間のファイル送信完了までの時間

即ち，T_1 はバックアップがファイル送信時間に影響しない場合のファイル送信完了までの時間を表し，T_2 はバックアップがファイル送信時間に影響する場合のファイル送信完了までの時間を表す．

実験の結果，T_1 は 3 分 58 秒であり，T_2 は 5 分 58 秒となった．ファイルサーバがバックアップを行いながらユーザからのファイルを受信する場合，バックアップを行わない場合と比較してファイル送信時間が 2 分増加していることが示される．

提案方式

本研究の提案手法では，ファイルサーバがユーザから送信されてきたファイルの受信している間，バックアップを一時的に中断させることでハードディスクが処理する量を軽減させ，ハードディスクが書き込み上限に達することを抑制する．図 4 に提案手法の流れを示す．

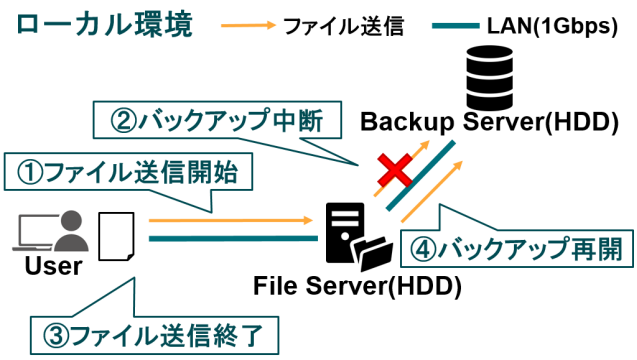

図 4　提案手法の流れ

図 4 の①から④までの流れを以下に示す．

① ファイル送信開始：ユーザにより動画ファイルの送信が開始される．

② バックアップ中断：ファイルサーバがディレクトリ内のファイル作成を検知し，ファイルの受信開始の判定とする．この判定により，バックアップサーバへのバックアップを中断する．

③ ファイル送信終了：ユーザによる動画ファイルの送信が終了する．

④ バックアップ再開：ファイルサーバがディレクトリ内のファイルの書き込みの終了を検知し，ファイルの受信終了の判定とする．この判定により，バックアップサーバへのバックアップを再開する．

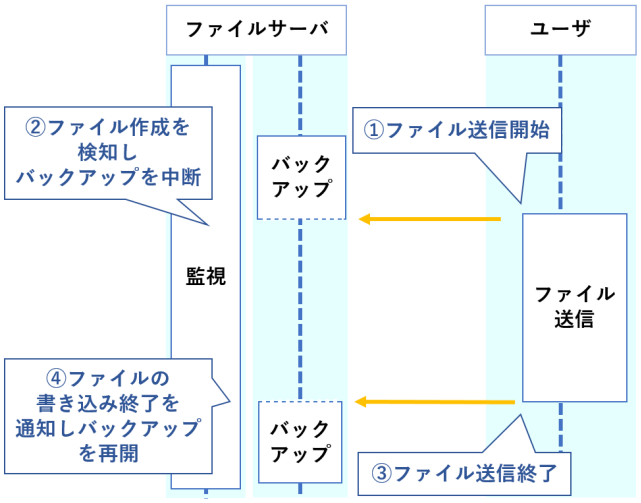

図 5　提案手法のシーケンス図

図 5 に提案手法のシーケンス図を示す．

①でユーザのファイル送信が開始されると，②でファイルサーバがファイル作成を検知し，バックアップを中断する．③でユーザのファイル送信が終了すると，④でファイルサーバが書き込み終了を通知し，バックアップを再開する．ファイルサーバ内の提案手法を取り入れたソフトウェアによって監視は常に行われ，ファイルサーバがユーザからファイルを受信している間にバックアップが中断される．中断されたバックアップの進捗は保持されるため，中断の度に初めからバックアップを行う事象を発生させることはない．

ユースケース・シナリオ

映画作成会社の映画ファイル管理をユースケースとして想定している．映画ファイルにおける漏洩リスクが存在するため，映画作成会社は社内のサーバを用いる．映画ファイルの収容にはファイルサーバを使用し，ファイルサーバ内には提案手法を用いたアルゴリズム配置する．また，同様に映画ファイル保護の観点から映画ファイルのバックアップを行うことを想定する．バックアップデータがファイルサーバ内に存在する場合ファイルサーバの故障が原因で映画ファイルを損失する機会を発生させるため，バックアップする映画ファイルを保存するためのバックアップサーバを構築することを想定している．映画作成会社の社員は映画ファイルをファイルサーバに送信する．ファイルサーバは指定された時刻や設定に従い映画ファイルをバックアップサーバに送信しバックアップをするが，映画ファイルの受信の検知した場合は映画ファイルの送信を一時停止する．この作業により，映画ファイルのファイルサーバへの送信の際にハードディスクの書き込み速度が上限に達することを抑制させ，送信時間の増加を抑制させることが可能になる．

4. 実装と実験方法

実装

実装は本研究の提案手法を実現するソフトウェアを新たに作成する．全体の実装構成図を図 6 に示す．

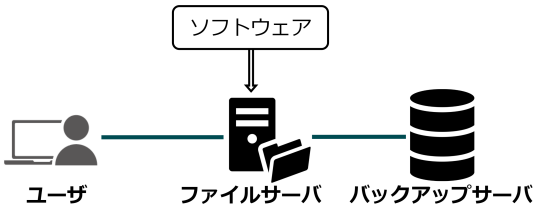

図 6 全体の実装構成図

ソフトウェアはファイルサーバ内に配置する．実装は，図 6 のユーザ，ファイルサーバ，バックアップサーバを前提にする．

ソフトウェアの機能を以下に示す．

(1) ユーザがファイルを送信すると，API である inotify によりファイル作成と書き込み終了を検知する

(2) inotify がファイル作成を検知した場合, kill コマンドでファイル送信に用いているコマンドを停止させ，バックアップサーバへのバックアップを中断する

(3) inotify がファイルの書き込み終了を検知した場合，バックアップサーバへのバックアップを再開する．

図 7 にソフトウェアに実装する機能のフローチャートを示す．

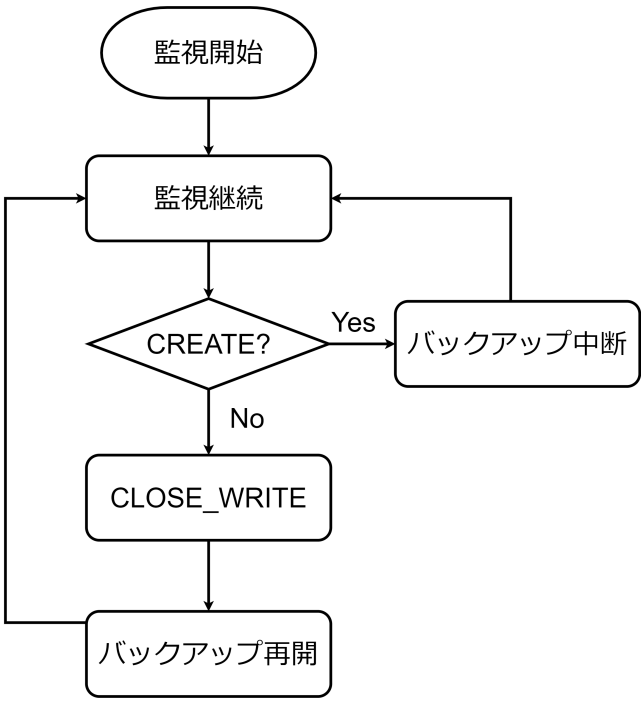

図 7 ソフトウェアに実装する機能のフローチャート

CREATE はファイルの作成の検知を表す．ユーザからのファイルの受信開始をトリガーにしてバックアップの中断を行う．CLOSE_WRITE はファイルの書き込み終了の検知を表す．ユーザからのファイルの受信完了をトリガーにしてバックアップの中断を行う．

実験環境

実験環境は，基礎実験と同様に仮想マシン (VM) を用いる．それぞれユーザ，ファイルサーバ，バックアップサーバを異なる VM として作成し，本研究の提案手法を取り入れたソフトウェアをファイルサーバ内に配置する．ユーザが送信する動画ファイルとバックアップデータには 10GB の容量で MOV 形式の動画ファイルを用いる．基礎実験では 100MB の動画ファイルを 10GB 分ディレクトリに入れた動画データを用いたが，実験段階でバックアップが再開しなかった．よって，基礎実験で用いた 10GB の動画ファイルで代用する．ファイル送信時間の計測には，コマンドの実行時間を計測する time コマンドを用いた．実験環境の全体図を図 8 に示す．

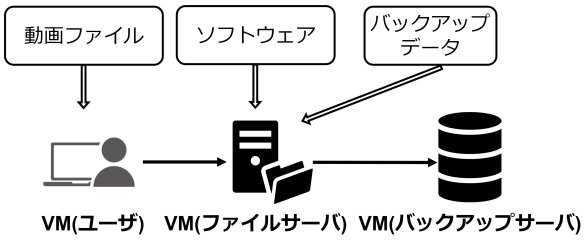

図 8 実験環境の全体図

ユーザは動画ファイルを用意し，ファイルサーバ内にソフトウェアを配置する．バックアップデータをファイルサーバ内に用意し，バックアップサーバへバックアップする．

5. 評価と分析

バックアップがされていないときのファイルの送信，課題となるファイルの送信，提案手法によるファイルの送信の，3 つの方法での時間の比較により評価を行う．これらをそれぞれ，T_1，T_2，T_3 とする．これらの計測は，ファイルサーバがバックアップを行いつつユーザから 10GB の MOV 形式の動画ファイルの受信が完了するまでの時間を計測する．

計測の結果，T_1 は 1 分 33 秒，T_2 は 1 分 57 秒，T_3 は 1 分 33 秒となった．図 9 に，基礎実験による結果と実験によって出た結果の比較を示す．

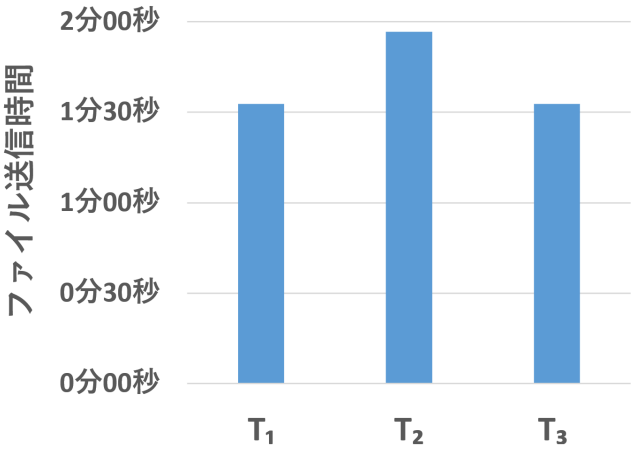

図 9 実験結果

下記に，T_1，T_2，T_3 の具体的な内容を示す．

T_1 バックアップがされていないときのファイルの送信：ユーザからファイルサーバへファイルの送信を行う際の，ファイル送信完了までの時間

T_2 課題となるファイルの送信：バックアップを行いつつ，ユーザがファイル送信を完了するまでの時間

T_3 提案手法によるファイルの送信：提案手法を用いてバックアップを行いつつ，ユーザがファイル送信を完了するまでの時間

図 9 より，T_3 は T_2 と比較して 24 秒減少している．また，T_1 と T_3 を比較すると，ファイル送信時間に差がない．これらは，提案手法を用いることで，ファイル送信時間がバックアップに影響を受けず，ファイル送信時間が増加していないことを表している．これにより，提案手法がファイルサーバの書き込み速度の低下を抑制できたことが示されている．

6. 議論

本研究で実装したソフトウェアでは，ディレクトリでの受信の検知に対応していないため，基礎実験で用意されていたディレクトリでの検証をすることができない．本研究の提案手法ではバックアップを完全に一時停止するため，バックアップが完了するまでの時間を増加させることに繋がっている．これは，バックアップを完全に停止せずに帯域幅を制限することで改善が可能である．また，バックアップ中断の判断にユーザからのファイル受信状況だけでなくハードディスクの書き込み速度の監視を加えることで，バックアップの中断の頻度を減少させ，バックアップ完了までの時間が増加する頻度を減少させることができる．

バックアップ中にバックアップファイルが更新された場合，バックアップ前のファイルとバックアップ後のファイルの整合性が取れない可能性がある問題がある．これは，バックアップ中またはバックアップ完了時にファイルの整合性を確認し，整合性が取れない場合は再度バックアップをすることで解決できる．

7. おわりに

ファイルサーバがユーザからのファイル受信と同時にバックアップサーバへのバックアップを行う場合，ハードディスクが書き込み速度の上限に達する．ここでの課題は，ユーザのファイル送信時間が増加することである．提案では，ユーザからのファイルの受信を検知しバックアップを一時的に中断させることで一度にハードディスクが処理する量を軽減させ，ハードディスクが書き込み上限に達することを抑制した．評価では，提案手法を用いない場合と用いる場合のファイル送信時間の比較を行った．結果として，10GB の動画ファイルにおけるファイル送信時間は 24 秒減少した．本研究の提案により，サーバの書き込み速度の低下の抑制が期待できる．

参考文献

[1] Suguna, S. and Suhasini, A.: Overview of data backup and disaster recovery in cloud, *International Conference on Information Communication and Embedded Systems (ICICES2014)*, pp. 1–7 (online), DOI: 10.1109/ICICES.2014.7033804 (2014).

[2] Jian-hua, Z. and Nan, Z.: Cloud Computing-based Data Storage and Disaster Recovery, *2011 International Conference on Future Computer Science and Education*, pp. 629–632 (online), DOI: 10.1109/ICFCSE.2011.157 (2011).

[3] Gavrilovska, A., Schwan, K. and Oleson, V.: A practical approach for 'zero' downtime in an operational information system, *Proceedings 22nd International Conference on Distributed Computing Systems*, pp. 345–352 (online), DOI: 10.1109/ICDCS.2002.1022272 (2002).

[4] Xie, B., Chase, J., Dillow, D., Drokin, O., Klasky, S., Oral, S. and Podhorszki, N.: Characterizing output bottlenecks in a supercomputer, *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–11 (online), DOI: 10.1109/SC.2012.28 (2012).

[5] Tanimura, Y., Hidetaka, K., Kudoh, T., Kojima, I. and Tanaka, Y.: A distributed storage system allowing application users to reserve I/O performance in advance for achieving SLA, *2010 11th IEEE/ACM International Conference on Grid Computing*, pp. 193–200 (online), DOI: 10.1109/GRID.2010.5697948 (2010).

[6] Chiang, R. C., Rajasekaran, S., Zhang, N. and Huang, H. H.: Swiper: Exploiting Virtual Machine Vulnerability in Third-Party Clouds with Competition for I/O Resources, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 26, No. 6, pp. 1732–1742 (online), DOI: 10.1109/TPDS.2014.2325564 (2015).

[7] Yang, Z., Fang, H., Wu, Y., Li, C., Zhao, B. and Huang, H. H.: Understanding the effects of hypervisor I/O scheduling for virtual machine performance interference, *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pp. 34–41 (on-

line), DOI: 10.1109/CloudCom.2012.6427495 (2012).

[8] Li, C., Goiri, c., Bhattacharjee, A., Bianchini, R. and Nguyen, T. D.: Quantifying and improving I/O predictability in virtualized systems, *2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*, pp. 1–6 (online), DOI: 10.1109/IWQoS.2013.6550269 (2013).

[9] Chen, H., Xiong, J. and Sun, N.: A novel hint-based I/O mechanism for centralized file server of cluster, *2008 IEEE International Conference on Cluster Computing*, pp. 194–201 (online), DOI: 10.1109/CLUSTR.2008.4663771 (2008).

[10] Song, H., Jin, H., He, J., Sun, X.-H. and Thakur, R.: A Server-Level Adaptive Data Layout Strategy for Parallel File Systems, *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, pp. 2095–2103 (online), DOI: 10.1109/IPDPSW.2012.246 (2012).

[11] Walters, J. P., Younge, A. J., Kang, D. I., Yao, K. T., Kang, M., Crago, S. P. and Fox, G. C.: GPU Passthrough Performance: A Comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL Applications, *2014 IEEE 7th International Conference on Cloud Computing*, pp. 636–643 (online), DOI: 10.1109/CLOUD.2014.90 (2014).

