HTTP-FUSE-cloop のソフトウェア RAID による高速化と耐故障性の実現

金井 遵[†] 須崎有康[††]
八木豊志樹[††] 並木美太郎[†]

概要 本稿では，ソフトウェア RAID による HTTP-FUSE-cloop の高速化と，Domain Name Server（DNS）を用いた耐故障性の向上手法を提案する．HTTP-FUSE-cloop はサーバ上の分割ブロックファイルを再構成する仮想ブロックデバイスである．本手法では，ソフトウェア RAID を用いて複数のサーバから同時にブロックファイルをダウンロードすることにより，HTTP-FUSE-cloop の高速化を実現した．また，耐故障性の確保は，RAID の冗長性と HTTP-FUSE-cloop で利用する HTTP がセッションレスのプロトコルであることに着目し，DNS ソフトウェアである DNS-Balance に手を加えることで，障害が起きたサーバを自動的に検知し，クライアント起動中においてもサーバ切り替えを可能にし，耐故障性の向上を実現した．評価より，従来手法に比べて，4 サーバからの同時データ取得を行った場合では，最大 4 倍近くの HTTP-FUSE-cloop ディスク性能の高速化を確認できた．

Implementation of Software RAID for HTTP-FUSE-cloop with Fault-Tolerance and Efficient Performance

Jun Kanai ,[†] Kuniyasu Suzaki,[††] Toshiki Yagi [††]
and Mitaro Namiki [†]

1. はじめに

近年，システムソフトウェアを含めた全ての情報をネットワークで一括管理を行うシンクライアントシステムを採用する企業や学校が増加している．また，管理の容易性から，企業や学校などだけではなく，モバイル環境でのカスタマイズされた統一環境の提供，家庭内など少数のマシンへのシンクライアント環境提供などの潜在的な要求もあると考えられる．HTTP-FUSE-KNOPPIX[1] は CD/DVD ブート可能な Linux である KNOPPIX[10] をベースとし，HTTP サーバ上のブロックファイルからルートファイルシステムを再構成する仮想ブロックデバイス HTTP-FUSE-cloop を使った Linux シンクライアントである．

しかしながら，HTTP-FUSE-cloop はインターネット上からブロックファイルを取得するため，性能の問題があった．これに対して，種々の高速化手法が提案されているが，サーバの性能限界などの根本的なディスク性能自体を改善する手法は少ない．そこで本研究では，ソフトウェア RAID により，複数のサーバからブロックファイルを取得することにより，ディスク性能の改善を試みた．また，RAID のバックアップデータが複数あるという特長を生かして，DNS に手を加えることにより，クライアント起動中でも利用サーバ切り替えを可能にし，耐故障性の向上を実現する．本稿ではこれらの設計，実装，評価について述べる．

† 東京農工大学 工学府 情報工学専攻
Tokyo University of Agriculture and Technology
†† 独立法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

2. HTTP-FUSE-cloop

本章では，本研究で用いる HTTP-FUSE-cloop の仕組みと，既提案の高速化，耐故障性向上手法および問題点について述べる．

2.1 HTTP-FUSE-cloop

HTTP-FUSE-cloop では，ファイルシステムのイメージを分割，圧縮し，HTTP サーバマシン (HTTP-FUSE-KNOPPIX サーバと呼ぶ) 上に置く．このファイルをブロックファイルと呼ぶ．図 1 のように，クライアント側で HTTP-FUSE-cloop がブロックファイ

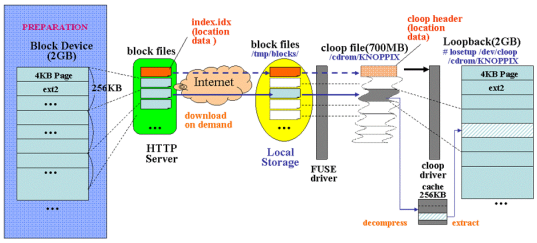

図 1 HTTP-FUSE-cloop

ルをオンデマンドで読み出し，ルートファイルシステムとすることで OS を起動する．ブロックファイル名は MD5 により決定され，ファイルの内容が同一であれば同一ファイル名になるため，サーバのディスク容量の節約が可能である．ブロックのディスク内位置とファイル名の対応決定には，同じく HTTP サーバ上に置かれたインデックスファイルが利用される．

2.2 既提案の手法と問題点

HTTP-FUSE-cloop では，ネットワーク上のブロックファイルを取得するため，ローカルのストレージから読み出す場合に比べて，レイテンシやバンド幅などの速度低下要因が存在した．また，スケーラビリティの実現のために，サーバを多数用意した場合には，耐故障性や負荷分散の実現が重要になる．そのため，次のような高速化手法，耐故障性，負荷分散の実現手法が提案，導入されている．

- ブロックファイルのキャッシュによる高速化
  HTTP-FUSE-cloop では，一度取得したファイルを RAM ディスクまたはクライアントのハードディスク等に保存し，次回以降のアクセスで利用することができる．しかしながら，ブロックファイル更新時にはインターネット上からファイルを再取得する必要があるほか，シンクライアントシステムにおいてはハードディスク中にデータを保存することができないため，マシンの起動毎にブロックファイルを取得し直す必要がある．
- DLAHEAD（DownLoad Ahead）[1)]
  HTTP-FUSE-cloop では，マシン起動時に必要なファイルを要求時ではなく，先読みする機能が備わっている（DLAHEAD 機能）．各種処理と並行して DLAHEAD は行われるため，特にレイテンシが大きい環境において効果を発揮する．しかし DLAHEAD は，OS 起動時のみに行われるため，起動後のアプリケーション起動などには効果がない．
- Ext2Optimizer[2)]
  HTTP-FUSE-cloop では，ブロックファイル中の一部分が必要な場合でもファイル全体を取得する必要がある．これは無駄が大きいため，Ext2Optimizer では，Ext2 でフォーマットされたディスクで，プロファイル情報を元に故意にフラグメントを発生させる．必要な領域を隣接させることで，無駄な領域のダウンロードを少なくするようになっている．しかしながら，全てのアプリケーションにおいて Ext2Optimizer で最適化を行うのは不可能であり，ディスク性能自体を高速化するものではない．
- キャッシュサーバによる高速化
  先行研究[1)] によれば，HTTP-FUSE-KNOPPIX の起動時間はネットワークのレイテンシに大きく左右される．一般に，インターネットからファイルを取得する際のレイテンシは，LAN 環境のレイテンシに比べて極端に大きい．そのため，筆者らは先行研究[3)] において，LAN 側に HTTP のキャッシュサーバを置くことにより，レイテンシを改善し，HTTP-FUSE-KNOPPIX の高速化を実現した．しかしながら，常に 1 台のサーバからデータを取得するため，サーバの性能限界には対処が不可能である．
- DNS によるトラヒック分散
  HTTP-FUSE-cloop では，一定時間毎に DNS を介してサーバの名前解決を行うため，サーバを複数用意し，返すサーバの IP アドレスを動的に変更することで，自動的にトラヒック分散を行うことができる．高速化のほか，これを応用して耐故障性を高めることができる．しかしながら現状では DNS 側でサーバの正常動作の確認を行っていないなど，耐故障性において不十分な面があった．

以上より，現状の HTTP-FUSE-cloop における問題点は，HTTP-FUSE-cloop の根本的なディスク性能の改善が行われていないことにあると考える．また，上記高速化手法は OS 起動を中心として考えられ，アプリケーション起動に関する高速化手法は少ない．一方，耐故障性についても不十分な点が多い．そこで本研究では，この HTTP-FUSE-cloop ディスク性能の向上と，不十分であった耐故障性の向上を行うことを目的とする．

3. ソフトウェア RAID による高速化と冗長性

本章では，ソフトウェア RAID を利用した HTTP-FUSE-cloop の高速化手法と冗長性について述べる．

3.1 概　　要

通常の物理的なストレージディスクの高速化手法として，Redundant Arrays of Inexpensive Disks(RAID) がある．RAID は，複数の物理ディスクを単一のディスクとして認識させることで，信頼性の向上や，高速化を行う技術である．Linux では md（multi disk）を利用することで，ソフトウェアによる RAID が構築できる．従来の HTTP-FUSE-cloop では，読み出しリクエストを 1 つずつしか受け付けられなかったが，md を用いて複数の HTTP-FUSE-cloop を RAID アレイ化し，複数のサーバからのブロックファイルの取得・展開を並列化することで，効果が期待できる．

RAID には RAID0〜RAID6，それに RAID0 と RAID1 を組み合わせた RAID0+1，RAID1+0 などが存在する．本研究では，主にディスクからの読み込み性能の高速化を目的とするため，RAID0 と RAID1 を利用する．それぞれの RAID 方式の特徴を次に示す．

- RAID0
  ストライピングと呼ばれ，複数台のハードディスクにデータを分散して保存する．ディスク性能はリード，ライトともに単体ドライブと比較して非常に改善される．
- RAID1
  ミラーリングと呼ばれ，複数台のハードディスクに同じ内容を保存する．ディスク性能は複数ドライブに同一内容を書き込むため，ライトでは落ちるが，リードでは複数ドライブから同時に読み込むことで改善される．

3.2 HTTP-FUSE-cloop への導入

HTTP-FUSE-cloop はループデバイスを介して，通常の物理ストレージデバイスと同じくストレージディスクとして認識される．そのため，md によるソフトウェア RAID 化が可能である．特に HTTP-FUSE-cloop を RAID アレイ化することで，次のような利点があると考えられる．

- KNOPPIX 自体の高速化
  OS 起動においても RAID によるディスク性能向上に伴う，高速化が期待できる．また，ソフトウェア RAID は DLAHEAD や Ext2Optimizer，キャッシュサーバ導入など他の HTTP-FUSE-KNOPPIX 高速化手段と同時に導入が可能であるため，相乗効果が期待できる．
- アプリケーションの高速化
  従来の HTTP-FUSE-KNOPPIX の高速化手法は，OS 起動に関するものが中心であった．本方式はディスク性能自体の向上が可能であるため，アプリケーションの高速化も期待できる．OS 起動と違い，アプリケーション起動は突発的にトラヒックが発生する．OS 起動後はサーバの帯域にも余裕が有ると考えられ，最大限クライアントの帯域を利用することが重要である．ソフトウェア RAID はこのためにも有用であると考えられる．
- 複数のプロセスからアクセス
  複数プロセスからディスクの読み出し要求があった場合，ダウンロードが並列化される．従来方式では逐次読み込みだったものが並列読み込みされ，高速化が期待できる．現在の PC において，ディスク性能は体感速度に大きく影響し，この傾向は複数アプリケーション動作時に顕著であるため，体感上での速度向上も期待できる．

本研究では，冗長性は複数サーバ内に RAID アレイを構成する物理ディスクイメージを分割した HTTP-FUSE-cloop の圧縮ブロックファイルを置くことにより確保される．よって，どのサーバからも RAID アレイを構成する全てのディスク内容を読み取ることができるようにするため，全てのサーバに全てのディスクイメージのブロックファイルを置く．これにより，図 2，図 3 のように，RAID0，RAID1 ともに全てのサーバが全ての物理ディスク内容を持つことになり，どのサーバからも RAID アレイを構成する全てのディスク内容を読み取れるため，冗長性を持つ．RAID アレイを構成するそれぞれのディスクを読み取るサーバを後述の DNS-Balance により選択し，md により RAID アレイを構築する．DNS-Balance により，ディスクイメージを読み取るサーバを必要に応じて切り替えれば耐故障性が向上できる．また，RAID1 では，md 自体の耐故障機能より，HTTP-FUSE-cloop 自体が問題を起こしても，他の HTTP-FUSE-cloop を利用できる．また，HTTP-FUSE-cloop によって物理的なネットワークインタフェースを変えることができるため，クライアントサイドでのネットワークトラブル時の耐故障性を確保できる．

これにより，RAID1 では実際の物理ディスク容量で，従来方式の RAID アレイに参加するディスク数倍のイメージ容量をサーバ上に置くことになる．しかし，HTTP-FUSE-cloop では，ブロックファイルのファイル名は MD5 により管理される．RAID1 ではディスクごとに異なる箇所は少数であり，ほとんどが同一ファイルとなるため，ディスクの必要容量は従来方式とほぼ変わらない．また，RAID0 においても従来方式とほぼ同じ容量になるほか，サーバによって読み取れるディスクを限定すれば，ディスク容量を節約でき，二

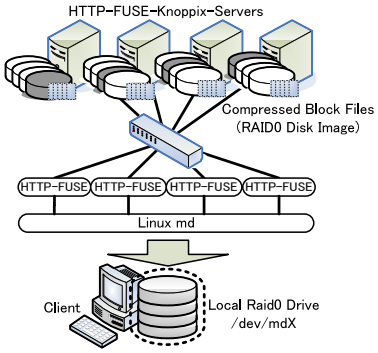

図 2 HTTP-FUSE-cloop と RAID0(ストライピング)

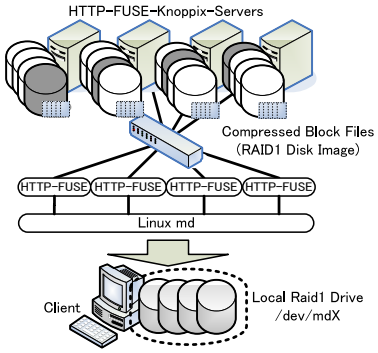

図 3 HTTP-FUSE-cloop と RAID1(ミラーリング)

次記憶容量が少ない環境でも導入が可能である．

4. DNS-Balance 改によるトラヒック自動分散と耐故障性の向上

本章では，DNS-Balance に手を加えた，DNS-Balance 改を用いたトラヒックの自動的な分散と耐故障性の向上方法について提案し，実際の設計・実装について述べる．

4.1 概　　要

RAID1 では複数のサーバにデータがミラーリングされる．また，RAID0 においても全てのディスクのイメージを 1 台のサーバ上に置くことで，擬似的に RAID1+0 のようにストライピングとミラーリングを組み合わせることが可能である．この RAID の特徴を利用し，1 台のサーバが障害を起こしたとき，他のサーバにクライアントが接続し直せば，耐故障性の向上が可能である．本研究では，DNS ソフトウェアに手を加え，クライアントの HTTP-FUSE-cloop に対して返す HTTP-FUSE-KNOPPIX サーバのアドレスを変えることで，耐故障性の実現及び，トラヒックの自動的な分散を実現した．

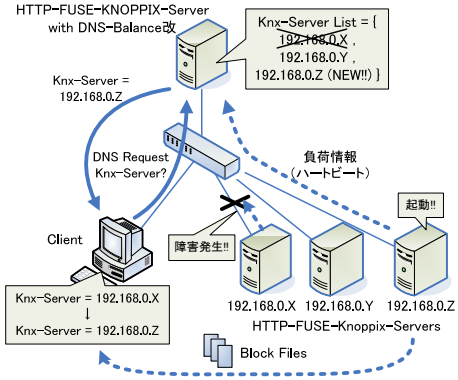

図 4 DNS-Balance と耐故障性

4.2 設計と実装

筆者らは先行研究[3] において，DNS によるサーバ分散を行うソフトウェアである DNS-Balance[11] に手を加えたものと，独自方式のロードバランサにより，HTTP-FUSE-cloop の耐故障性およびトラヒックの分散を実現した．しかしながら，両者ともプロトタイプであったため，DNS-Balance では，サーバの生存確認ができないため，障害が起きたサーバを割り当てる可能性があることや単純にラウンドロビンでサーバを割り振っていた問題があった．また，独自方式では，クライアントが起動中のサーバ変更が行えないことなど不十分な点が多かった．一つのサーバに障害が発生した場合，直ちに他のサーバにクライアント割り当てを変更する必要がある．そこで，本研究では DNS-Balance にさらに手を加え，問題点を解決するとともに，独自方式の機能を統合した．具体的には次の機能を実現した．

- サーバ負荷分散機能
  本機能ではクライアントの割り当て状況や負荷状況などによって，DNS が返す HTTP-FUSE-KNOPPIX サーバの IP アドレスを変更することにより，サーバの負荷分散を行う．クライアント側では，従来機能により HTTP-FUSE-cloop が図 4 のように HTTP-keep-alive および TTL の期限切れごとに DNS に対して名前解決を行う．これにより，割り当てサーバが変更される．サーバの分散方式として次の 2 方式を用意した．
  – 起動時決定方式
    クライアントの起動から終了まで同一のサーバを使う方式である．ただし，サーバに異常が発生した場合はこの限りではない．本方式ではサーバのトラヒックの上限が見積りやすい．従来，独自方式で実装されていた機能を

統合した．

– 動的決定方式
サーバの負荷状況に応じて，クライアントに対して返す IP アドレスを決定し，割り当てサーバを変更する方式である．本方式ではトラヒックに応じて，サーバの柔軟な運用が可能である．

トラヒック分散を目的とした，クライアントのサーバ割り当てルールとして，以下のような指針を設けた．

– ネットワーク負荷の高いサーバは避ける
– サーバの能力（手動設定）に応じて柔軟に割り当てを可能にする
– 同一マシン内の各 HTTP-FUSE-cloop の同じサーバへの割り当ては避ける
– 各サーバのクライアント割り当て台数が均等になるようにする
– 各 HTTP-FUSE-cloop は同一サーバを使い続ける

優先度はほぼ上記の順序になっている．これを定式化した (1) 式により，クライアントはサーバに割り当てられる．ここで，各サーバの優先度 P は，各 HTTP-FUSE-KNOPPIX サーバのネットワーク負荷を $LoadNW$（10 段階），サーバの優先度を Pri，既割り当てのクライアント台数を $nCli$，既に同一マシン内の HTTP-FUSE-cloop がそのサーバに割り当てられているかを $sFuse$（0 または 1），前回 DNS リクエスト時にそのサーバに自分自身が割り当てられたかを $oFuse$（0 または 1）として，

$$P = LoadNw + nCli/(0.2 \times Pri) + 5 \times sFuse - oFuse \quad (1)$$

で算出され，最も P の値が小さいサーバにクライアントが割り当てられる．Pri の係数 0.2 は，手動設定の優先度によって $LoadNw$ と $nCli$ のバランスを取ることが可能なことから決定している．Pri を大きくすることで，割り当てクライアント台数を増やせる．また，$sFuse$ の係数 5 は，極力同一マシン内の複数の HTTP-FUSE-cloop を同じサーバに割り当てないためにこの値になっている．ただし，既割り当てのサーバに非常に余裕があり，倍のネットワーク負荷がかかっても問題なく，さらに他のサーバとネットワーク負荷の差が非常に大きい場合は，同一マシン内の複数の HTTP-FUSE-cloop を同じサーバに割り当てても性能低下は非常に小さいと考えられ，割り当てるためにこの係数になっている．

また，各 HTTP-FUSE-KNOPPIX サーバは割り当てられるクライアントの IP アドレスの範囲を指定することができる．これにより，ネットワーク形態に応じた負荷分散を半自動化することができる．

- 耐故障性機能
クライアントからの HTTP-FUSE-KNOPPIX サーバの名前問い合わせに対して，正常なサーバのみを返すようにし，耐故障性を確保した．具体的には，DNS-Balance サーバ側と，HTTP-FUSE-KNOPPIX サーバ側で以下のような障害検出・回復を行っている．
  – ハートビート検出
  HTTP-FUSE-KNOPPIX サーバは DNS-Balance サーバに対して，負荷状況を一定時間ごとに送信する．この負荷状況はハートビートを兼ねている．DNS-Balance は一定時間負荷状況の報告がなかったサーバをサーバリストからはずす．また初めて報告があったサーバは動的にサーバリストに追加される．
  – HTTP サーバの正常動作確認
  HTTP-FUSE-KNOPPIX サーバ側では，自サーバ内で動作している HTTP サーバに対して定期的にファイル取得を試みる．ファイル取得ができない場合，サーバが異常状態であると見なし，ハートビート送出を停止する．これにより，DNS-Balance サーバ側でサーバリストから該当 HTTP-FUSE-KNOPPIX サーバを削除する．さらに，HTTP サーバを再起動することにより，障害回復を試みる．

  図 4 のようにクライアントの HTTP-FUSE-cloop はサーバからブロックファイルが取得できない場合，再度 DNS に対して名前解決を行うので，ネットワーク全体をダウンさせることなく，クライアントの起動中でも動的なサーバの追加削除が可能である．
- DNS フォワーダ機能
耐故障性の向上や高速化に直接寄与する機能ではないが，DNS リクエストを他のサーバに投げる，DNS のフォワーダ機能を追加した．これは，DNS-Balance がマシン起動後も DNS として利用されるため，HTTP-FUSE-KNOPPIX サーバ以外の名前解決も行う必要があるためである．

図 5 USL-5P による HTTP-FUSE-KNOPPIX-BOX

5. HTTP-FUSE-KNOPPIX-BOX への実装

筆者らは HTTP-FUSE-KNOPPIX を起動可能にする小型サーバとして，HTTP-FUSE-KNOPPIX-BOX を開発している[3)]．先行研究[3)] では，x86 マシンに組み込まれた NFS ベースのシンクライアントサーバに比べて，各種組み込みハードウェアによる HTTP-FUSE-KNOPPIX-BOX でも十分な性能が得られた．HTTP-FUSE-cloop は NFS ベースのシンクライアントサーバに比べて負荷が小さく，クライアント台数を増やした場合の性能低下も小さい．HTTP-FUSE-KNOPPIX-BOX は管理の容易性，低負荷，低コストであることなどから，モバイル環境などのほか，家庭内，教育機関，情報キオスク端末などへの利用が考えられる．また，モバイル環境だけでなく従来のシンクライアントサーバ環境の小型化や省電力化，低コスト化も可能である．

HTTP-FUSE-KNOPPIX-BOX は，HTTP サーバまたはキャッシュサーバ，TFTP サーバ，DHCP サーバなどからなり，通常の PC サーバから組み込み向け環境など，Linux が動作する環境であれば動作可能である．貧弱なサーバのハードウェア環境では，ソフトウェア RAID と DNS-Balance 改によりリクエストが分散されるので，特に本手法が効果を発揮すると思われる．本研究では実際に，I・O データ社の小型 USB サーバである USL-5P を HTTP-FUSE-KNOPPIX-BOX 化し，RAID アレイ化したイメージを組み込み，サーバを多重化した（図 5）．USL-5P は，大きさは文庫本サイズ，重量は 200g 以下と小型軽量であり，十分にモバイルが可能である．

従来の HTTP-FUSE-KNOPPIX-BOX に対して行うべき変更点は少ない．ソフトウェア RAID に関しては，RAID イメージのブロックファイルを置けばよいので，特にソフトウェアなどの変更の必要はない．サーバの自動的な分散や，耐故障性を確保するためには，DNS-Balance にハートビートを送るクライアントソフトウェアを導入する必要がある．しかしながら，これは非常に小さなソフトウェアであり，CPU，メモリリソースはほとんど消費しない．

また，HTTP-FUSE-KNOPPIX-BOX には，キャッシュサーバが導入されている．キャッシュサーバを介することで，ディスクイメージのコピーの自動的な配布が可能になる．また，本手法と併せて，トラヒックを DNS-Balance により自動的に分散させることも可能である．これにより，HTTP-FUSE-KNOPPIX サーバを多数用意した場合でもイメージに関するメンテナンスの手間が必要なくなる．

6. 評　　価

ソフトウェア RAID を導入した HTTP-FUSE-cloop について，RAID 方式，サーバ数，HTTP-FUSE-cloop のブロックファイルサイズなどを変えた際のディスク性能の計測や，実際のアプリケーション起動時間の計測を行った．本章では，これらの詳細について述べる．

予備実験より，組み込み環境では多くの場合，ブロックファイルを読み取るためのディスクバンド幅がボトルネックになることが確認されている．そのため，HTTP-FUSE-KNOPPIX サーバとしては前述の USL-5P を最大 4 台用意した．また，クライアントとしては Athlon64x2 による SMP マシンを利用した．サーバとクライアント間は GbE による LAN により接続されている．サーバと，クライアント環境の諸元を表 1 に示す．

クライアントマシンには，4 つの HTTP-FUSE-cloop を導入し，md により RAID0 または，RAID1 でまとめて一つのディスクとする．さらに，HTTP-FUSE-KNOPPIX サーバの 1 台の USL-5P に DNS-Balance 改を導入し，名前解決はこのサーバに対して行うようにする．全ての HTTP-FUSE-KNOPPIX サーバにはハートビートを送信するソフトウェア，HTTP サーバを導入し，RAID アレイを構成する全てのディスクイメージを HTTP-FUSE-cloop のブロックファイルに変換したものを置いた．よって，HTTP-FUSE-cloop が DNS-Balance に対して名前解決を行うが，どのサーバでも全てのディスク内容を読み取れるようになっている．さらに，これらのサーバおよびクライアントを一つのギガビットスイッチングハブに接続して評価を行った．

ディスク（ext2 でフォーマット）の基本性能を計測

するベンチマークプログラムは，ディスクに対して任意の粒度でランダムアクセスを行うプログラムを作成した．ベンチマーク時に読み込まれるディスク上に置くファイルの内容は，ランダムに生成し，ブロックファイルは元のファイルサイズとほぼ同じサイズと，圧縮率が悪い状況になっている．また通常，HTTP-FUSE-cloop は高速化のために，ローカルにダウンロードしたブロックファイルのキャッシュファイルを置き利用するが，ボトルネックになると考えられるネットワークからダウンロードした際の性能を測るため，キャッシュしないようにして計測している．同様に，ファイルシステムによるキャッシュ利用を極力避けるため，ディスクイメージおよびベンチマークアクセス用ファイルのサイズは 300MB とし，そのうち最大 100MB をベンチマークにより，アクセスするようになっている．今回，RAID のチャンクサイズはブロックファイルサイズの 2 倍としている．これは，1 つのリクエストに対して，極力同じブロックファイルを複数のサーバから取得しないようにするためである．

6.1 ソフトウェア RAID の方式による評価

最初に，最適な RAID の方式を調べるために，RAID 方式の違いによるディスク性能の計測を行った．ディスクイメージを RAID0，RAID1，RAID なし (従来方式) で作成し，ブロックファイルサイズ 128KB で分割圧縮し，HTTP-FUSE-KNOPPIX サーバ 4 台の上に置く．これをマウントし，ランダムアクセスプログラムによりディスク性能を計測した．また，単一プロセスによりアクセスを行った場合と，8 プロセスにより並列アクセスを行った場合の性能を計測した．単一プロセスの場合の結果を図 6，表 2 に，8 プロセスの場合の結果を図 7，表 3 に示す．

8 プロセスではランダムアクセスの粒度にかかわらず，RAID0，RAID1 ともに，従来方式に比べてサーバ 4 台で 4 倍近くの高速化になっており，アプリケーションなどの同時起動にも強いと考えられる．RAID0 と RAID1 での比較では，若干 RAID1 の方が速いという結果になった．これは，RAID0 の場合は，md による冗長性はないため，アクセスすべきサーバがあらかじめ決まる．つまり，ディスクの読む位置によって，1 つのサーバにアクセスが集中することになると，リクエスト待ちが起きてしまうためだと考えられる．RAID1 の場合には，ディスクの位置ではなく，アクセス順序によってアクセスするサーバが決まるため，このような症状は起きない．

一方，1 プロセスにおいても，従来方式に比べて性能向上を達成した．これは，ランダムアクセスサイズが大きい場合，並行してデータがリードされるためだと考えられる．1 プロセスの場合は 8 プロセスの場合と逆に RAID0 が RAID1 の性能を上回るという結果になった．各サーバのトラヒック状況を見ると，RAID1 では逐次アクセスに近い状況になっていたため，md の読み込みアルゴリズムに関する性能の違いだと考えられる．

また，Ext2Optimizer による最適化を併せて行うことで，アクセス回数が最適化されるため，HTTP-FUSE-KNOPPIX のさらなる高速化が可能である．

6.2 サーバ数による評価

続いて，サーバを複数台用意することによる効果を

表 1 テスト環境

	サーバ (USL-5P)	クライアント
CPU	SH4-240MHz	Athlon64x2 3800+
メモリ	64MB	1GB
二次記憶	USB メモリ (1GB)	HDD
ネットワーク	100Mbps	GbE
サーバ	HTTP(thttpd)	–
重量	190g	–

表 2 RAID 方式によるディスク性能差（単一プロセス性能）

アクセスサイズ [KB]	RAID0	RAID1	従来方式
1	0.0209	0.0144	0.0167
10	0.107	0.148	0.129
100	1.06	0.807	0.710
1000	5.64	3.67	2.69
10000	10.8	6.44	4.26

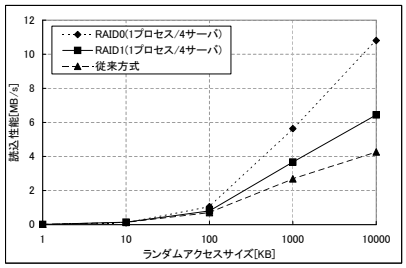

図 6 RAID 方式によるディスク性能差（単一プロセス性能）

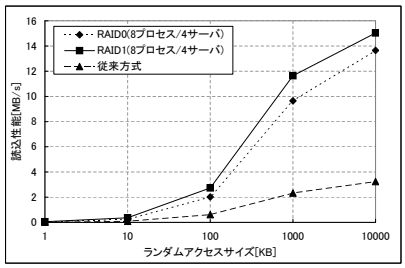

図 7 RAID 方式によるディスク性能差（複数プロセス性能）

表 3 RAID 方式によるディスク性能差（複数プロセス性能）

アクセスサイズ [KB]	RAID0	RAID1	従来方式
1	0.0534	0.532	0.0131
10	0.233	0.364	0.0907
100	2.02	2.73	0.614
1000	9.64	11.6	2.33
10000	13.6	15.0	3.24

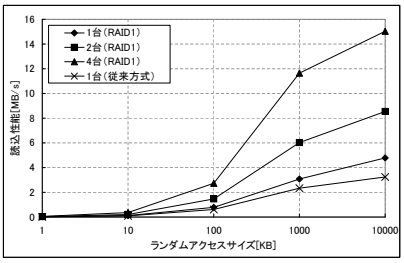

図 8 サーバ数によるディスク性能差

表 4 サーバ数によるディスク性能差

サイズ [KB]	1 台	2 台	4 台	従来方式
1	0.0136	0.0313	0.0532	0.0131
10	0.117	0.209	0.364	0.0907
100	0.785	1.47	2.73	0.614
1000	3.07	6.03	11.6	2.33
10000	4.79	8.54	15.0	3.24

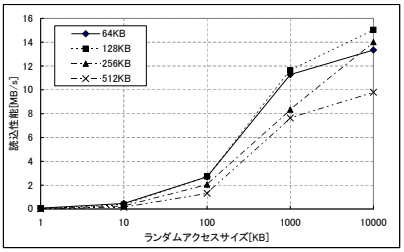

図 9 ブロックファイルサイズによるディスク性能差

調べるために，サーバ数の違いによる評価を行った．サーバの台数を 1，2，4 台と変え，RAID1 と従来方式によりディスク性能の計測を行った．ブロックファイルサイズは 128KB，同時読み込みプロセス数は 8 としている．計測結果を図 8，表 4 に示す．

結果より，サーバ 1 台に対して，4 台の場合は約 3 倍以上の性能になっている．貧弱なサーバ環境では，サーバの台数を増やすことによる，アクセスの分散は効果が大きい．また，同じサーバ 1 台の場合でも，従来方式とソフトウェア RAID 利用の場合で，ソフトウェア RAID を利用した方が速いという結果になった．これは，HTTP-FUSE-cloop によるファイル取得・展開処理が並列化されたためだと考えられる．

6.3 ブロックファイルサイズによる評価

続いて，イメージファイルを分割するブロックファイルのサイズの最適値を調べるため，ブロックファイルを 64KB，128KB，256KB，512KB と変えてディスク性能の計測を行った．RAID 方式は RAID1，同時読み込みプロセス数は 8，サーバ数は 4 台である．計測結果を図 9，表 5 に示す．

ランダムアクセスサイズが小さい場合は 64KB が最適，大きい場合には 128KB が最適という結果になった．しかしながら今回の計測では，同一のブロックファイルに対してアクセス回数が少数の場合について計測している．Ext2Optimizer により必要な部分を同一ブロックファイル内にまとめ，不必要なブロックファイルのダウンロードを減らすことが可能であり，このときブロックファイルサイズが小さいことの利点は少なくなる．また，ブロックファイルサイズが小さいと，HTTP サーバに対するリクエストが多くなる．またファイル探索の計算量も多くなるため，負荷が大きい．そこで，ブロックファイルサイズは 128KB または，256KB 程度が最適であると考えられる．

表 5 ブロックファイルサイズによるディスク性能差

サイズ [KB]	64KB	128KB	256KB	512KB
1	0.0738	0.0532	0.0326	0.0196
10	0.458	0.364	0.251	0.140
100	2.70	2.730	2.06	1.31
1000	11.3	11.6	8.34	7.64
10000	13.3	15.0	14.0	9.80

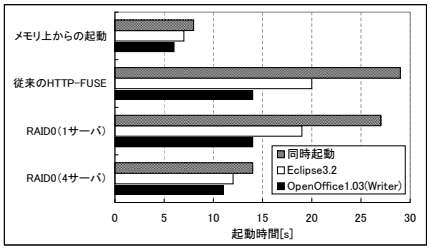

図 10 アプリケーション起動時間の比較

6.4 アプリケーション起動時間の評価

続いて，実際の利用場面においての本方式の有用性を調べるため，応用評価として本方式を適応したディスクにアプリケーションのプログラムやデータを置き，アプリケーションの起動時間の計測を行った．従来方式と RAID0 で計測を行っている．ブロックファイルサイズは 128KB，サーバ数は 1 台と 4 台とした．アプリケーションとしては，起動時間が長いアプリケーションとして OpenOffice.org1.0.3(Writer)，Eclipse3.2 を選んだ．計測結果を図 10 に示す．

実際の利用場面においても，ソフトウェア RAID による高速化手法が有用であることがわかった．メモリからの起動に比べると遅いものの，サーバ 4 台では従来方式に比べ，Eclipse では 1.7 倍，OpenOffice.org でも 1.3 倍と高速な起動を実現している．また，体感速度に大きく影響すると思われるアプリケーションの同時起動にも非常に強く，2.6 倍以上の起動速度を実現している．

6.5 耐故障性の評価

最後に，DNS-Balance による耐故障機能を利用し，サーバが問題を起こした場合を想定した耐故障性の評価を行う．HTTP-FUSE-KNOPPIX サーバを 2 台用

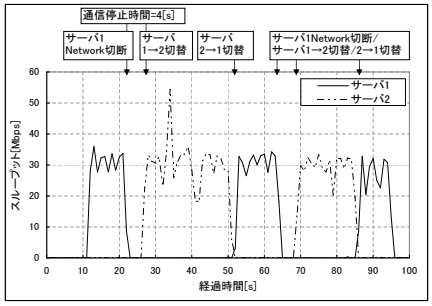

図 11 従来方式（RAID なし）の場合のトラヒック状態

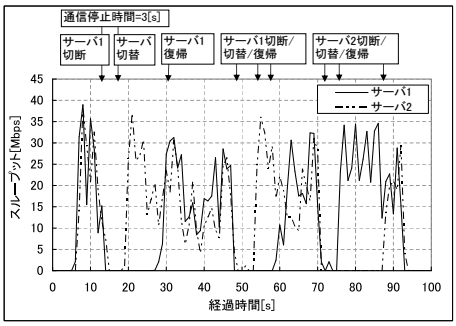

図 12 RAID0 の場合のトラヒック状態

意し，サーバ上に置いた，従来方式と RAID0 を使ったイメージをクライアントサイドで HTTP-FUSE-cloop を介してマウントする．さらに，dd による巨大なファイル読み込み中にサーバサイドのネットワークを物理的に切断，復帰を繰り返し，クライアントサイドでプログラムが異常を起こすことなく，サーバが自動的に切り替わることを確認する．確認はサーバサイドでトラヒック状況を確認することにより行った．従来方式のトラヒック状態を図 11 に，RAID0 のトラヒック状況を図 12 に示す．HTTP-FUSE-cloop の接続のタイムアウト時間を 3 秒，DNS 問い合わせ結果のキャッシュ時間を 3 秒，ハートビート送信間隔を 1 秒，ハートビート送信が無い場合にサーバリストから該当サーバから削除されるまでの最大時間を 3 秒とする．

結果より，実際にサーバが切り替わっていることが確認できる．今回，パラメータより，サーバからの通信停止からサーバ切り替えまでの最大時間の理論値は 6 秒であるが，実際にその範囲内に収まっていることが分かる．また，RAID0 ではサーバ復帰後に割り当てサーバが元のサーバに切り替わり，トラヒックが再び分散されていることが確認でき，耐故障性とともにトラヒック分散が実現できた．

6.6 考　　察

ソフトウェア RAID とサーバ分散によるディスク高速化の効果を実証できた．特に実際の利用場面では，単一プロセスでも高速化可能な RAID0 と，イメージサイズは 128KB か 256KB を組み合わせるのが良いと思われる．今回ベンチマークに用いたファイルは圧縮率が最低と，最も状態が悪い状況でのアクセスであり，通常のファイルに対してはクライアント側のスペックが高い場合にはさらに高速化を望むことができると考えられる．

なお，今回ボトルネックになっているのは，サーバ側のディスクのバスバンド幅および，クライアント側のブロックファイル展開性能である．サーバ，およびクライアントをさらに高性能なものにすると，FUSE の性能限界は 30MB/s ほどであるため，FUSE がボトルネックになると考えられる．

7. 今後の課題

今後の課題や，現状の問題点として次のような課題を挙げることができる．

- ソフトウェア RAID のルートファイルシステムへの導入と相乗効果の検証
  本方式はアプリケーションだけで無く，OS 起動でも高速化を期待できる．実際に HTTP-FUSE-KNOPPIX のルートファイルシステムにソフトウェア RAID を導入し，評価を行う必要がある．また，HTTP-FUSE-KNOPPIX にはソフトウェア RAID 以外にもさまざまな高速化手法が存在する．これらを本手法とともに導入し，評価を行う必要がある．
- DNS サーバの問題
  DNS-Balance 改による耐故障性は，DNS-Balance が動作しているサーバが正常に動作していることにより保証される．このため，DNS-Balance のサーバに異常が発生した場合，正常なサービスが行えなくなる．さらに耐故障性を向上させるためには，DNS サーバに異常が発生した場合に，他の HTTP-FUSE-KNOPPIX サーバが DNS サーバを起動する仕組みなど，対策が必要である．

8. 関 連 研 究

高速にファイルを取得できるファイルシステムとして，NRFS，Lustre，WAFS などが存在する．また，既存のシンクライアントを実現するシステムとして，Ardence，STRAGEX が存在する．これらと本手法についての比較を行う．

- NRFS[7)]
  NRFS は NFS をベースとして，ソフトウェア

RAID を利用可能にした分散ファイルシステムである．ホットスワップに対応し，信頼性を実現している．しかしながら，ミラーリングのみの対応であることや，NFS ベースであるため，WAN への拡張が難しいことや性能の問題がある．

- Lustre[9)]
  Lustre ファイルシステムとは大規模クラスタなどのために開発されている分散ファイルシステムである．Lustre ではメタデータと実際のデータを分離することで高速化を実現している．また，ミラーリングによる高信頼性も実現している．Lustre はクラスタマシンでの利用を前提とし，メタデータサーバやストレージサーバなどを別々に用意する必要があるため，サーバ規模が大規模になる．
- WAFS[12)]
  WAFS は WAN に対応した高速なファイルシステムである．WAFS では，LAN 側に置いたキャッシュサーバの利用およびプロトコルの効率化により高速化を行っている．しかし，サーバが大規模化し，ファイアウォール越えも難しい．HTTP-FUSE-KNOPPIX-BOX では，広域にサービスを提供できるほか，ハードウェア規模も小さくでき，キャッシュサーバによる高速化の他にも様々な高速化手法を組み込んでいる．
- Ardence[13)]
  Ardence は仮想ディスクを利用して，Windows や Linux のシンクライアントを実現するシステムであり，OS によらずシンクライアントシステムを利用できるという利点がある．一方で，HTTP-FUSE-cloop は Linux 専用であるが，HTTP を利用しファイアウォールを容易に越えることができ広域にシンクライアントシステムを提供できる，マシン毎にディスクイメージの作成の必要がない，ブロックファイル圧縮，サーバ多重化，負荷分散による高速化などの利点がある．
- STRAGEX[14)]
  STRAGEX は iSCSI を利用したスケーラブルなシンクライアントシステムであり，WAN 化も視野に入れて開発されているのは HTTP-FUSE-cloop と同様である．しかし，HTTP-FUSE-cloop は HTTP および，DNS，TFTP，DHCP の他には特殊なサービスが必要なく，ファイアウォールを容易に越えることができるため，HTTP によるイメージの配布が容易という利点がある．ハードウェアも iSCSI ストレージのような特殊なハードウェアが必要なく，安価である．

9. おわりに

以上より，本稿ではソフトウェア RAID による HTTP-FUSE-cloop の高速化と，DNS-Balance を組み合わせた耐故障性の向上を実現した．また，実際に評価より，HTTP-FUSE-cloop の高速化の確認ができた．特に，ハードウェア性能が制限される小型 LinuxBox で，実用的な性能が確認できたことは，HTTP-FUSE-KNOPPIX-BOX のモバイル化，低コスト化に繋がり意義深い．また，HTTP-FUSE-cloop は KNOPPIX ばかりでなく汎用のネットワークブロックデバイスであり，今後応用を広げていきたい．

参 考 文 献

1) 須崎，八木，飯島，北川，田代：ネットワークに対応した分割圧縮ループバックデバイス HTTP-FUSE-CLOOP とそれから起動する Linux，インターネットコンファレンス 2005.
2) 北川, 丹, 阿部, 千葉, 須崎, 飯島, 八木：圧縮ブロックデバイスにおけるファイルシステム最適化, Linux Conference 2006 (2006.6).
3) 金井, 須崎, 八木, 並木:HTTP-FUSE-KNOPPIX-BOX のキャッシュ導入による高速化と評価, SIGOS, Vol.2006-OS-102, pp.45-52 (2006.5).
4) Suzaki, Yagi, Iijima, Kitagawa, Tashiro: HTTP-FUSE Xenoppix, Ottwa Linux Symposium 2006
5) Kitagawa, Tan, Abe, Chiba, Suzaki, Iijima, Yagi: File System (Ext2) Optimization for Compressed loopback device, Linux-Kongress 2006
6) 佐々木, 安立, 田村, 安田, 横井: HTTP-FUSE-KNOPPIX を基盤としたサーバ学習環境システムの開発, 教育システム情報学会 第 6 回研究会「情報教育の実績と新しい展開」
7) 松本：ネットワーク RAID ファイルシステム, http://www.ssspc.org/nrfs/files/nrfs-linux.pdf
8) D. A. Patterson, G. A. Gibson, R. H. Katz: A Case for Redundant Arrays of Inexpensive Disks (RAID) Proceedings of the International Conference on Management of Data (SIGMOD), June 1988.
9) Lustre http://www.lustre.org/
10) KNOPPIX http://unit.aist.go.jp/itri/knoppix/
11) DNS Balance http://openlab.jp/dns_balance/
12) Cisco Wide Area File Services（WAFS）http://www.cisco.com/japanese/warp/public/3/jp/product/hs/storage/fewafs/
13) Ardence http://www.vci.com/
14) 市川，岡，鷲坂: iSCSI を活用したシンクライアント PC システム STRAGEX, IPSJ, Vol.47, No.SIG12(ACS15), pp.377–386(2006.5).