ネットブートにおけるクライアント間での
ルートファイルシステム共有

--- ネットワーク対応した CAS (Content Addressable Storage) ---

渡辺義人† 北川健司† 阿部大将† 濱野裕樹† 千葉大作†
須崎有康‡ 八木豊志樹‡

概要 クライアント管理が容易に行えることからネットブート方式が注目されている.この方式で起動されるコンピュータはサーバから OS やアプリケーションのファイルを取得して動作を行う．そのため，サーバ障害が発生すると動作中のクライアントにも影響する方式となっている.筆者らは,各クライアントがサーバから OS やアプリケーションのデータの異なる部分を取得し,クライアント間で共有することでサーバに依存しない形で冗長化･負荷分散を行うことが可能となるネットワーク対応した CAS の実装を行った.

Sharing of Root Filesystem for Network Booting Clients

--- CAS(Content Addressable Storage) for Network Booting ---

YOSHIHITO WATANABE† KENJI KITAGAWA† DAISUKE ABE† HIROKI HAMANO† DAISAKU CHIBA†
KUNIYASU SUZAKI‡ TOSHIKI YAGI‡

1 はじめに

近年，多数のコンピュータを保有する企業や公共機関，教育機関では，コンピュータ管理コストの削減や，情報保護のためのセキュリティポリシーの徹底等の対応に迫られている．また，高速な LAN の普及や安価なコンピュータの高性能化による影響もあり，コンピュータをネットワーク経由で起動するネットブート方式が注目されている.

この方式では,ネットブートサーバ(以下,サーバ)が OS やアプリケーションのファイル（以下，システムファイル）を持ち，ネットブートされるコンピュータ（以下，クライアント）がサーバから必要なデータを取得し動作を行う．また，システムファイルがサーバ上にあるため，アップデートやセキュリティポリシーに従った設定等の管理作業を一括して即時に行うことが可能である.その一方で，サーバやサーバへのネットワークに障害が発生すると，クライアントはシステムファイルの取得が行えず動作を継続することができなくなってしまう（single point of failure）．そのため，ネットブート方式の安定性はサーバにクリティカルに依存しており，安定性向上のために冗長化や負荷分散を行わなければならない．しかし，これらの対策はサーバの大規模化を招き，構築・運用費用の増加につながってしまう．そこで，筆者らは Linux においてシステムファイルに該当するルートファイルシステムを分割し，各クライアントが分割されたルートファイルシステムの異なる部分を保持し，お互いに補完しあうことで，サーバに依存することなく，冗長化・負荷分散が可能であると考えた.

実装はネットブート用に開発された HTTP-

†株式会社アルファシステムズ
‡独立行政法人産業技術総合研究所

FUSE KNOPPIX[1][2][3]がルートファイルシステムのマウントに使用しているHTTP-FUSE cloopを拡張し実現した．なお，HTTP-FUSE cloopは最近LB-CASと改称されているため，以後は新しい名称のLB-CASを用いる．

本稿ではLB-CASを使用したネットブートにおけるクライアント間でのルートファイルシステム共有についての検討，実装及び実験結果について報告する．

2 LB-CAS とHTTP-FUSE KNOPPIX

2.1 LB-CASとは

CAS(Conten-Addressable Storage)とは，内容を元にアドレッシングを行うストレージシステムである．最もポピュラーなシステムとしてはPlan9で使われているVenti[4]がある．

LB-CAS（LoopBack-CAS）は，指定したサーバ上にあるブロックファイルとインデックスファイルから，読み取り専用のループバックマウント可能なディスクイメージファイルを仮想的に復元するシステムである． LB-CASの動作イメージを図１に示す．

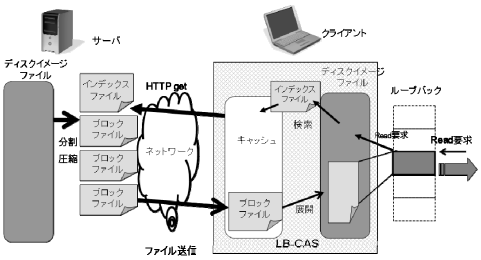

図1 LB-CASの動作イメージ

ブロックファイルは，ディスクイメージファイルを分割，圧縮したファイルである．ファイル名はブロックファイル自身のハッシュ値（SHA1）が付けられている．インデックスファイルは元のディスクイメージファイルのアドレスとブロックファイル名（SHA1値）の対応付けが記録されている.クライアントは，データ読込みが発生した際にインデックスファイルを元に必要なブロックファイルを決定しサーバからHTTPでブロックファイルを取得する．

HTTP-FUSE KNOPPIXはLB-CASで仮想的に復元されたKNOPPIXのディスクイメージファイルをルートファイルシステムにマウントし起動するKNOPPIXである．

2.2 既存の負荷分散手法

この節では既に提案されているサーバの負荷分散手法とその問題点について検討する．

・DNSとプロクシを利用した手法[5]

この手法では，DNSの名前解決を利用して，各クライアントで起動しているプロクシサーバがキャッシュミス時にアクセスする上位のプロクシサーバ（クライアント）を制御し，クライアントをツリー状に構造化している．そして，ブロックファイルの取得はプロクシサーバを通して階層的に行われている．そのため，起動時の様に同じブロックファイルが各クライアントから要求される場合，プロクシサーバによるキャッシュの効果によりサーバへのトラフィックが軽減される．しかし，DNSによる名前解決を頻繁に行わなければ木構造を制御できないため，TTLを短くしてDNSを利用している．そのため，DNS障害の影響を受けやすいと考えられる．

・HTTPリダイレクトを利用した方法［6］

この手法は，クライアントからのブロックファイル取得要求をサーバがDBに記録している．そして，同じブロックファイルを要求した別のクライアントには，先にブロックファイを取得したクライアントへHTTPリダイレクトを行う.HTTPリダイレクトによりクライアント間でブロックファイルの転送が行われる．そのため，トラフィックの大部分を占めるブロックファイルの転送が分散されサーバのトラフィックが軽減される．しかし，リダイレクト先のクライアントの異常終了や，ブロックファイルの削除が発生すると，リダイレクトされたクライアントはブロックファイルの取得ができず停止してしまう．また，ブロックファイルの転送要求は必ずサーバを経由するため，サーバが停止するとクライント間の転送も行えなくなる．

いずれの場合も，サーバに機能追加を行っ

ており，サーバへの依存という点では逆に強くなっていると考えられる．

3 提案手法の概要

3.1 サーバへの依存の軽減

HTTP-FUSE KNOPPIX で必要なものは，インデックスファイルとブロックファイルを配信し，KNOPPIX のルートファイルシステムを提供する HTTP サーバである．この HTTP サーバの役割を完全にクライアントに移管することができれば，一時的なサーバ停止に影響されない安定した動作や，サーバの負荷分散を行うことができると考えられる．

3.2 ルートファイルシステムの共有

提案手法では，サーバを利用せずにルートファイルシステムをクライアントが提供するために，ルートファイルシステム全体を分割し，各クライアントが異なる部分を保持し共有している．共有はピュア P2P の形で行うためサーバ本体に変更を加える必要はない．

懸念点としては，キャッシュの共有とは異なり，提案手法は一定量のブロックファイルを保持するためにメモリを消費する点がある．しかし，KNOPPIX ベースであるため圧縮した段階で 700MB 程度に収まり，クライアント 10 台で分割した場合 1 台当たり必要なメモリは 70MB 程度となる．最近のコンピュータに搭載されているメモリ量と比較すると十分に小さいと考えられる．

クライアントの台数不足によってルートファイルシステムが冗長化できない可能性も懸念されるが，ネットブートにより管理を効率化する組織ではある程度のクライアント数が確保できると考えられる．そのため，クライアント不足によりルートファイル全体を保持できない状況は発生しにくいと考えられる．

4 LB-CAS の機能拡張

追加実装している機能は大きく分けて，ブロックファイルのクライアントへの分散機能とクライアント間での共有機能である．

LB-CAS の場合，ルートファイルシステムの実体はブロックファイルであるため，この二つの機能を利用してルートファイルシステムの共有を行うことができる．

4.1 ブロックファイルの分散

企業や教育機関といった組織では，クライアントが使用される時間帯や台数が決まっていることが予想される．そのため，一般的な P2P と異なり，ファイル共有を行うノードが動的に変化することは少ないと考えられる．そこで，提案手法ではブロックファイルのクライアントへの動的な分散は行わないことにした．あらかじめブロックファイルを複数のグループに分け，実行時にはブロックファイの取得のみを実施している．

4.1.1 ブロックファイルの分散戦略

例えば，合計 9 個のブロックファイルを，3 台のクライアントに分散させることを考える．ブロックファイルの分散戦略として，図 2 の様にブロック単位でサイクリックに分散する方法(以下，サイクリック分散方式)や図 3 の様に一定サイズのブロック毎に分散する方法（以下，ブロック分散方式），または，この二つを組み合わせる方法が考えられる．

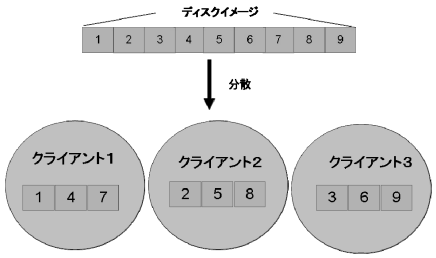

図 2 サイクリック分散方式

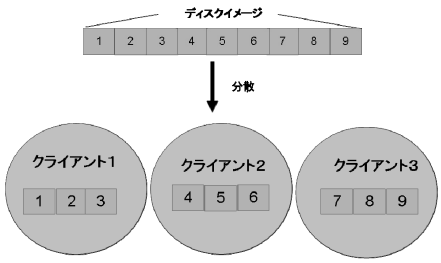

図 3 ブロック分散方式

図ではそれぞれのブロックファイルについて元のディスクイメージファイルの先頭から順に 1，2，3.... と番号をつけて記載しているが，実際はハッシュ値（SHA1）がファイル

名になっている.

提案手法では，サイクリック分散方式を採用している．理由は，一般的にファイルシステムはディスクアクセスの効率を考慮して,1つのデータは物理的なディスク上でも極力連続するように配置するためである.

HTTP-FUSE KNOPPIX のディスクイメージファイルは ext2 でフォーマットされており，1つのデータはディスク上でなるべく連続するように配置される．そのため，図3の様にブロック分散方式を採用すると，特定のクライアントへアクセスが集中してしまう可能性が考えられる.

4.1.2 ブロックファイルの分割と配布

全てのブロックファイルがクライアントへ分散されている状態に分割を行うためには，次の条件を満たす範囲で分割数を決定する必要がある．分割数をD，同じブロックファイルをもつクライアントの台数，言い換えるとブロックファイルの冗長度をRとすると

D×R<クライアント台数

ブロックファイルの合計サイズ÷D<空きメモリ量

となる．また，クライアントの一時的な停止を考慮してRは2以上であることが望ましい．しかし，クライアントの台数やブロックファイルの合計サイズ，空きメモリ量は使用する環境に依存するため，実験を行い最適な値を決定しなければならない.

ブロックファイルの配布は，ブート時にクライアントが使用するリストを指定しクライアントがリストにあるブロックファイルを取得することで行っている．ブロックファイルのリストは，ブロックファイルの分割数と分割戦略に従ってインデックスファイルから生成される．ユーザ操作やKNOPPIX の起動の妨げとならないように，ブロックファイルの取得は，KNOPPIX 起動後にバックグラウンドで行っている.

4.2 ブロックファイルの共有

4.2.1 ブロックファイル共有の設計方針

ピュア P2P の形式でブロックファイル共有を行うために，ブロックファイルを持っているクライアントを探すブロックファイル検索機能と，ブロックファイルの取得先として存在するクライアントを探すクライアント発見機能が必要である.

P2P ファイル共有ソフトウェアではよく大規模化/大容量化の為に比較的オーバヘッドが大きいアルゴリズムが利用されるが，本提案はネットブートを利用するため，LAN 内部での利用が前提条件にある．そのため，小・中規模を想定しユーザへの応答性能を重視した単純な機能で実装している.

4.2.2 クライアント発見機能

LAN 環境による利用であることから，マルチキャスト通信を利用し，マルチキャストメッセージに対する応答により行っている．マルチキャストを用いることで，同一セグメント内での効率よく通信ができ，ネットワークの設定により規模の拡大にも柔軟に対応することができる.

4.2.3 ブロックファイル検索機能

ブロックファイル検索では，取り扱うブロックファイルの数は数千個程度かつクライアント数も小・中規模であるため，ブロックファイルとブロックファイルを持っているクライアントを対応付ける管理情報（以下，管理情報）は十分に小さいと考えられる.

実装では管理情報は個々のクライアントがそれぞれ保有し，クライアント発見の度に管理情報の取得と追加を行っている．検索時には各クライアント内の管理情報を参照し外部には検索クエリを発行しないことで，検索の応答速度を上げている.

4.2.4 ファイルの取得機能

ブロックファイル検索の結果から，ブロックファイルを持っていると判明したクライアントまたはサーバのいずれかから直接 HTTP で取得する．ブロックファイルのダウンロードに失敗した場合は，そのアクセス先を検索対象から除外し別のサーバかクライアントからダウンロードを行うようにしている．サーバからの取得が失敗した場合は，一時的に利用不可のフラグを付けて別途サーバの復帰確認を行うようにしている.

4.2.4 クライアンの起動/停止確認

各クライアントでは，ブロックファイルのダウンロードに失敗したアクセス先を検索対

象から外している．しかし，CPU 負荷が高かったために応答がない場合や接続上限に達している場合等がある．そのため，失敗した場合は必ずしもクライアントが停止しているとは言えない．そこで，一定時間ごとにクライアントの起動/停止を確認する同期処理を行い，クライアント全体で検索対象のリフレッシュを行っている．同期処理は，クライアントの IP を元に取りまとめ役となるクライアントを決定し，取りまとめを行った後，起動しているクライアントのリストをマルチキャストで送信している．

5 実験

実験は表１に示す環境で，機能追加を行った LB-CAS を用いて行い，トラフィックと CPU 使用率を計測した．実験は，ネットブート後のブロックファイル分散が完了してから一斉に OpenOffice.org の起動を行った．また，表 2 にクライアントとサーバの詳細なスペックを示す．

表１ 実験環境

サーバ台数	1 台
クライアント台数	40 台
ネットワーク	100Mbps
KNOPPIX のバージョン	5.1.1
総ブロックファイルサイズ	740MB
OpenOffice.org のバージョン	2.1

表2 マシンスペック

サーバ スペック	
CPU	Core2 Duo U7500
メモリ	1024MB
Swap	1024MB
クライアント スペック	
CPU	Athron 2400+
メモリ	512MB
Swap	1024MB

5.1 ブロックファイルの分散方式によるアクセスパターンの違い

4．1 で触れたブロックファイルの分散方法によるアクセスパターンの違いを確認するため，ブロックファイルを各クライアントに 1/20 ずつ配布した状態で，サイクリック分散方式による場合(図 2)とブロック分散方式による場合(図 3) とブロックファイル共有を行わない場合の 3 つの場合で実験を行った．表 3 にそれぞれの場合で tcpdump を用いて取得したクライアントの HTTP のパケット数の最大と最小を示す．

表3 tcpdump によるパケット数

	ブロック分散	サイクリック分散	共有なし
最大	476,807	91,368	51,300
最小	50,066	74,300	50,022

単位:個

ブロック分散方式では，パケット数が最小と最大で 10 倍近く異なり，特定のクライアントにアクセスが集中していることが読み取れる．また最小値は共有を行っていない場合とほぼ同じであり，他のクライアントへブロックファイルを転送していないことが分かる．サイクリック分散方式では，最小と最大の差は 20%程度であり，ブロック分散方式に比べて偏りは大幅に少なくなっている．

5.2 ブロックファイルの共有機能による冗長化と負荷分散

提案手法を用いた場合に，全クライアントへブロックファイルを分散後サーバを停止し，クライアントで OpenOffice.org 等のアプリケーションが起動できることを確認した．また，サーバが動作した状態で，提案手法を用いてブロックファイルを 10 分割してクライアントに共有させた場合（以下，10 分割），ブロックファイルを 20 分割してクライアントに共有させた場合(以下，20 分割) と提案手法を用いない場合（以下，共有なし）において，OpenOffice.org を起動した場合のネットワークトラフックと各マシンでの CPU 利用率を測定した．

5.2.1 サーバのトラフィックと CPU 負荷

図 4 に，共有無し，20 分割，10 分割のそれぞれの場合で，tcpdump で取得したサーバの

トラフィックを示す．20 分割の場合は，全体が 40 台なので各クライアントから見るとあるブロックファイルを持ったサーバがネットワーク上に 3 台（元のサーバ＋クライアント 2 台）あるように見える．同様に，10 分割の場合は，5 台のサーバがあるように見える．また，図 5 にサーバで vmstat によって取得した CPU の負荷を示す．サーバの CPU 負荷にはまだ余裕があることが分かる．そのため，サーバのボトルネックはネットワーク帯域であることが読み取れる．10 分割と 20 分割の場合でグラフが途中で切れているのは，OpenOffice.org の起動が完了して，データ計測を終了したためである．

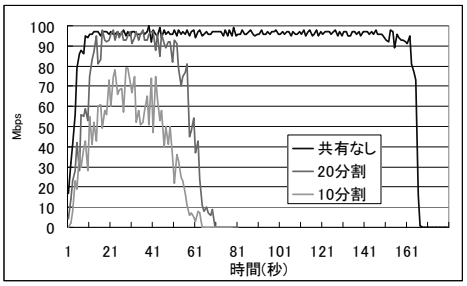

図 4 サーバのトラフィック

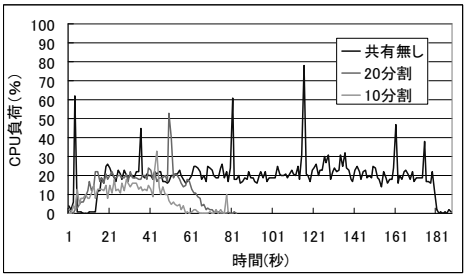

図 5 サーバの CPU の負荷

5.2.2 クライアントのトラフィックと CPU 負荷

図 6 に共有無しの場合，図 7 に 20 分割の場合，図 8 に 10 分割の場合における，クライアント 40 台の総トラフィックを示す． input は各クライアントが受信した総トラフィック，output は送信した総トラフィックとなっている．また，図 9 に 40 台のクライアントのうちのある 1 台のクライアントにおける CPU 負荷を示す．また，図 10 は同じクライアントでの CPU の I/O 待ちの割合を示す．

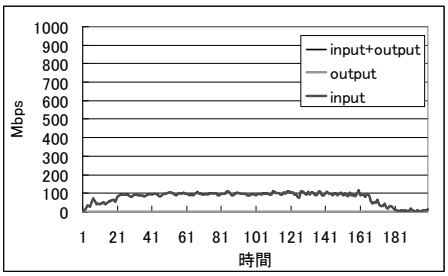

図 6 クライアントの総トラフィック（共有なし）

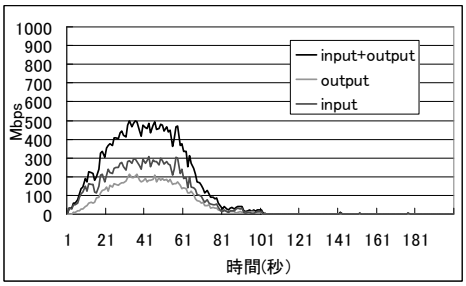

図 7 クライアントの総トラフィック(20 分割)

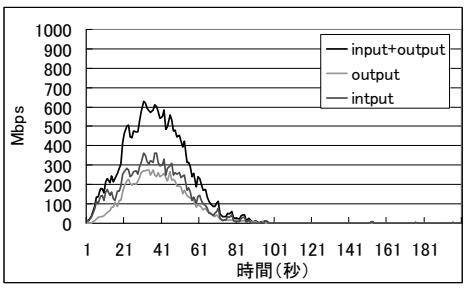

図 8 クライアントの総トラフィック(10 分割)

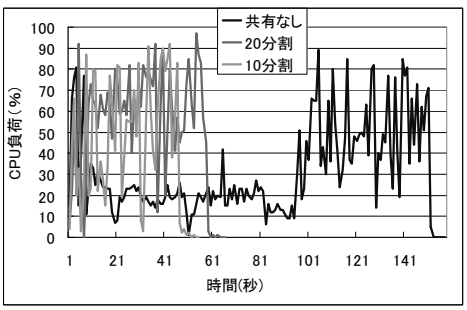

図 9 クライアントの CPU 負荷

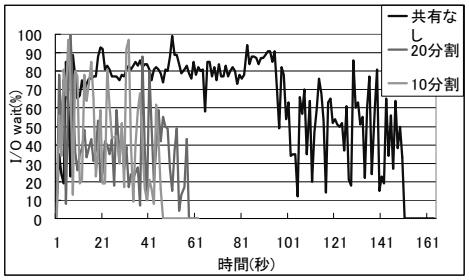

図10 クライアントのCPUのI/O待ちの割合

図6，図9，図10から，共有なしの場合はCPUに余裕がある状態であり，ネットワークトラフィックが100Mbpsとなっていることから，クライアントにおいてもボトルネックがネットワーク帯域にあることが読み取れる．これは，全てのクライアントがサーバに接続するためサーバまでの通信経路が混雑するためと考えられる．図7と図8ではネットワーク帯域の上限100Mbpsを大きく超えているが，この点については後ほど考察する．

6 考察

6.1 サーバトラフィックの軽減

5.2節の実験から，共有なしの場合はサーバとクライアント間のネットワーク帯域がボトルネックとなっている．

提案手法の場合はクライアント間でブロックファイルが転送されたことにより，サーバのトラフィックが軽減されている．また，クライアントの総トラフィックは帯域の上限を超えている．理由は，ブロックファイルの転送がクライアント間で行われたため，通信の分散がおこりスイッチングハブの異なるポート間で通信が発生したためと考えられる．そのため，スイッチングハブの特性を生かして負荷分散が行えていると考えられる．

クライアントのCPUの負荷状況を見ると，提案手法の利用の有無に限らずCPU負荷は100%近くになり，その持続時間も同程度である．そのため，提案手法はクライアントに影響が出るほど負荷を与えていないと考えられる．

6.2 ブロックファイルの分散

5.1節の実験から，ブロックファイルの分散方法によっては，特定のクライアントに負荷が集中することが示された．負荷の集中は新たなボトルネックの発生や，そのクライアントを利用するユーザへの応答速度の低下を招くと考えられる．

6.3 サーバの総トラフィック

サーバの総トラフィックについても検討を行った．

クライアントの台数 *Nclient*
ブロックファイルの合計サイズ *Vblocks*
KNOPPIX起動時のダウンロードサイズ *Vboot*
アプリ起動時のダウンロードサイズ *Vapp*
ブロックファイルの分割数 *D*
クライアントへの割当てサイズ $Vclient = Vblocks/D$

と置くことでサーバの総転送量は以下の式で表すことができる．

共有なしの総転送量

$= (Vboot + Vapp) \times Nclient$

提案手法の総転送量 *(Nclient > Dの場合)*

$= Vboot \times Nclient$ **1*
$+(Vblocks-Vboot) \times (Nclient/D)$ **2*
$+(Vapp-Vclient/D) \times Nclient/(Nclient/D+1)$ **3*

**1は起動にかかる転送量*
**2はブロックファイルの分散にかかる転送量*
**3はアプリケーション起動時のサーバの転送量*

表4に特定のアプリケーションの起動に必要なブロックファイルの合計サイズを示す．また，表4でアプリケーションを起動しているものについて共有あり，10分割，20分割についてサーバの総転送量を計算し，その推移を図10にまとめた．

表4 ブロックファイルサイズ

起動の状態	サイズ
KNOPPIXのみ	129
KNOPPIX＋OpenOffice．org	178
KNOPPIX＋OpenOffice．org ＋Firefox	190
KNOPPIX＋OpenOffice．org ＋Firefox＋wireshark	199

単位：MB

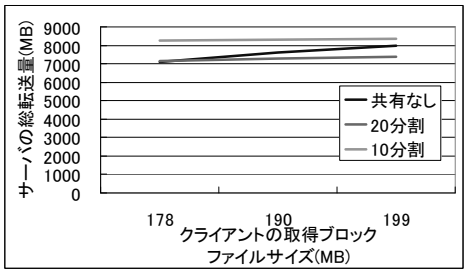

図 10 サーバの総転送量

20 分割の場合は，共有なしの場合よりもサーバの総転送量は減り，10 分割の場合は増えている．これは，最初にブロックファイルを各クライアントに配布する転送量が，アプリケーションの起動時に負荷分散によって軽減される転送量より多いためである．また，負荷分散による総転送量の軽減は，アプリケーションの起動時に発生する．そのため，複数のサイズの大きいアプリケーションが利用される場合に提案手法がより有効となる．

7 今後の課題

・ルートファイルシステムの肥大化への対応

ユーザのニーズによっては大量のアプリケーションをインストールして利用することが考えられる．その場合，ルートファイルシステムが肥大化し総ブロックファイルサイズが増加する．そのため，提案手法で必要とするクライアントのメモリも増加していくが，メモリの容量はどこかで限界を超えることが考えられる．対策としては，すべてのブロックファイルの分散をあきらめ，優先度の高いアプリケーションや，システムに関連するブロックを優先して分散させることが考えられる．

・ネットワーク負荷の検討

提案手法を利用した場合，考察で述べたようにネットワーク全体で通信が行われている．これは，ネットワークに負荷を与えていると考えることもできる．そのため，ネットワークを利用するアプリケーションを使用する場合を考慮して，必要に応じて占有する帯域の上限値を設けるような制御が必要となると考えられる．

8 関連研究

・viver[7]

viver はネットブート環境を作るためのフレームワークである．その中にサーバが提供するディスクイメージファイルを各クライアントで分割して共有する機能がある．viver では動的にディスクイメージファイルの一部を各クライアントにブロック分散方式で割り当てている．また，他にも多くの点が動的に構成されるように設計されている．

9 おわりに

LB-CAS にブロックファイルの分散・共有機能の追加実装を行い，クライアント間でのルートファイルシステムの冗長化と負荷分散を実現した．これにより，スイッチングハブの特性を生かした負荷分散が可能となり，ネットワークの有効利用が可能となった．また，クライアントはサーバ障害に影響されにくくなり，ネットブート方式の安定性が向上したと考えられる．

参考文献

[1] Kuniyasu Suzaki, Toshiki Yagi, Kengo Iijima, Nguyen Anh Quynh,"OSCircular: Internet Client for Reference",USENIX-LISA(21st LargeInstallation System Administration Conference), pp.105-116 (2007.11)

[2] 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-FUSECLOOP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 (2005).

[3] 八木，須崎，後藤，飯島，丹：HTTP を用いた広域分散ストレージ， 第 4 回情報科学技術フォーラム(2005)

[4] Sean Quinlan and Sean Dorward, "Venti: a new approach to archival storage", USENIX FAST02 (Conference on File and Storage Tchnologies) (2002.1)

[5] 金井， 須崎， 八木， 並木：HTTP-FUSE-KNOPPIX-BOX によるモバイルシンクライアントシステムの実現，電子情報通信学会論文誌「ユビキタス時代の情報基盤技術」特集号,Vol.J90-D, No.6, pp.1383-1393 (2007.6)

[6] 後藤，北川，須崎，：HTTP-FUSE KNOPPIX におけるキャッシュを利用した起動手法の提案，情報処理学会研究報告.［システムソフトウェアとオペレーティング・システム］, vol. 2007, pp. 103-110,

[7] http://viver. sourceforge. jp/