p3cache： ネットブート型シンクライアント端末群を対象としたP2P型OSキャッシュ共有機構

深谷 健太[1,a)]　松原 克弥[1,b)]

概要：ネットブート型シンクライアントは，各端末の計算資源を有効活用しつつ，導入から運用までの総合コスト (TCO) を削減できるソリューションとして，企業や教育機関，施設等の大規模 PC 環境で広く採用されている．一方，各端末のディスクイメージをサーバに集約して起動時にネットワーク配信するため，サーバやその経路ネットワークへの負荷集中が課題となる．サーバ冗長化や端末のローカルディスクへのキャッシュなどの既存対処手法では，新たな設備の導入を要するため，TCO 削減との両立が難しい．本研究では，端末上で起動した OS が持つキャッシュに着目し，各端末のメモリ上にキャッシュされているディスクデータ片を必要とする近隣の他端末へ配信することで，サーバに対する負荷の削減を目指す．さらに，各端末の動作状況に応じて変化する OS キャッシュの効率的な分散管理を実現するために，ネットブートのディスクデータ要求毎に P2P 型のキャッシュ探索を行い最適なデータ保有端末を特定する機構を実現する．本稿では，Linux カーネルのページキャッシュ機構を対象として，P2P 型キャッシュ管理とディスクイメージ・サーバ機能を実現した p3cache の実装について述べる．また，複数台のシンクライアント端末群を用いた実験評価についても示す．

1. はじめに

ペーパレスや電子決済など，組織内の情報システムの進歩と ICT の利活用が進むにともない，PC の普及度合いは 1996 年度の 62%から，今日では 90%を超え，1 人 1 台，もしくは複数台の環境へと進んできている．そのような状況において，PC の保有にともなうコスト（TCO）を削減するソリューションとして，シンクライアント端末の採用が進んでいる．シンクライアントは，画面転送型（VDI）とネットブート型に大別できる．VDI は，クライアント端末に画面のみを転送することによって，計算資源をサーバに集約し，クライアント端末導入コストを下げ，さらに，仮想化技術により計算機環境をサーバに集約することで管理コストを削減できる．一方，集約されるサーバや端末までのネットワークに高い性能が求められ，負荷集中時のレスポンス低下などが PC の操作性に与える影響が大きい．一方，ネットブートは，サーバ上のディスクイメージを用いて端末上で OS 起動を行う技術である．同一仕様の PC 端末向けのディスクイメージをサーバで集約して一元管理することにより，PC の主な故障原因となるハードディスクを端末から排除し，さらに，ソフトウェア環境の一括管理による保守コストの削減を可能にする．RAM や GPU などの計算資源は，各端末に装備されたものを活用できるため，高性能な計算サーバを必要としないことも TCO 削減に貢献できる．

ネットブートにも，複数クライアント端末の同時起動の際に課題がある．多数のクライアント端末を同時に起動するような状況では，サーバへの負荷が一時的に集中する．その結果，サーバのネットワーク帯域がボトルネックとなり，OS 起動時間の増加やアプリケーションのレスポンスの低下などの問題が発生する．この課題へ対処するため，これまで以下に述べる 2 つの手法が用いられてきた．第 1 の対処法は，負荷のピーク時に求められる性能に合わせて，冗長サーバを追加する手法である．クライアント端末からのデータ要求を複数のサーバへ振り分けることで，1 台のサーバへの要求を減らすことが可能となる．また，クライアント側での対処手法として，クライアント端末上のローカルディスク上にディスクイメージのキャッシュを保持する手法がある [1], [2]．ネットブート時に，サーバから取得したデータを端末に装備したローカルディスクに保存する．データ保存後の OS 起動は，サーバへはデータを要求せず，自身のローカルディスク上のデータを用いて行われる．このように，ローカルディスクのデータを用いること

[1] 公立はこだて未来大学
Future University Hakodate.
[a)] g2118035@fun.ac.jp
[b)] matsu@fun.ac.jp

で，サーバへのデータ要求を減らすことが可能となる。しかし，これらの手法では，本来不要なクライアント端末のローカルディスクが必要となったり，サーバを追加したりなど，セキュリティ対策や TCO 対策削減に影響を及ぼす．

本研究では，ローカルディスクやサーバの追加を必要としない，ネットブート時のサーバの負荷分散手法を提案とする．本研究が提案するシステム p3cache は，起動済みのクライアント端末で動作する OS が RAM 上に持つファイルキャッシュに着目する．ファイルキャッシュは，Linux や Windows をはじめとする多くの OS が持つ機能であり，ネットブートを介して起動した OS においても，サーバ上のディスクイメージに対するファイルキャッシュを保有する．p3cache では，このファイルキャッシュを，近隣端末の起動に際してサーバの代わりに提供することで，サーバへの負荷集中を軽減する．特に，ネットブートにより起動するクライアント端末は，複数端末間で同様な環境が構築されるため，同じ内容のディスクデータを要求するため，他端末のキャッシュ再利用の効果が高いことが推測できる．

本手法は，起動が済んだクライアント端末もサーバの代わりとして機能するため，既存手法のようなローカルディスクやサーバの追加なしに，サーバへの負荷を分散することが可能である．加えて，RAM を用いるクライアント端末の方は，ローカルディスクを用いるサーバに比べ，より高速にデータ取得が可能である．ゆえに，クライアント端末がデータ転送を行う場合には，サーバより高速な応答が可能となり RTT(Round Trip Time) の削減も期待できる．

以降，本稿では，第 2 章で，p3cache の設計とシステム実現における技術課題と解決法について説明する．第 3 章で，ネットブートのプロトコルとして ATA over Ethernet(以降，AoE) を対象として，Linux カーネル内での p3cache の実装について述べる．第 4 章では，前章で述べた実装を用いた実験結果について示す．第 5 章で関連研究について紹介し，第 6 章でまとめと今後の課題について述べる．

2. p3cache の設計

2.1 システム構成

p3cache のシステム構成を図 1 に示す．p3cache は，端末とブートサーバから構成される．端末は，他の端末からのデータ要求に対して，自身が起動に用いたキャッシュデータをネットワーク転送する．ブートサーバは，起動中のクライアント端末の要求に応じて必要なデータを送信する．加えて，キャッシュ保有端末リストを作成し，各端末のキャッシュ状況を管理する．

p3cache の動作フローを図 1，図 2，図 3 に示す．まず，起動済みの端末が存在しない場合のフローを (図 1) 示す．最初に，端末 A は，ブートサーバとの通信を開始するための初期化処理時に，キャッシュ保有端末リストにキャッシュ保有端末として登録される (図 1 (1))．また，キャッシュ保有端末リストに登録された端末は，一旦，すべてのデータをキャッシュしているとみなされる．その後，端末 A は，通常通りブートサーバへデータ要求を行い，ブートサーバから起動に必要なデータを取得する (図 1 (2)，(3))．

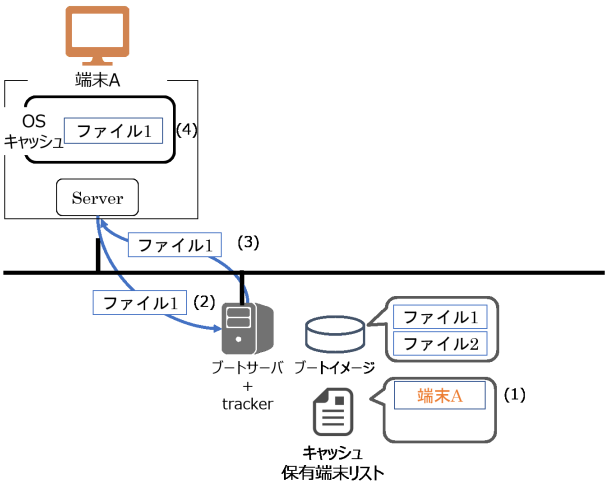

図 1 1 台目の起動

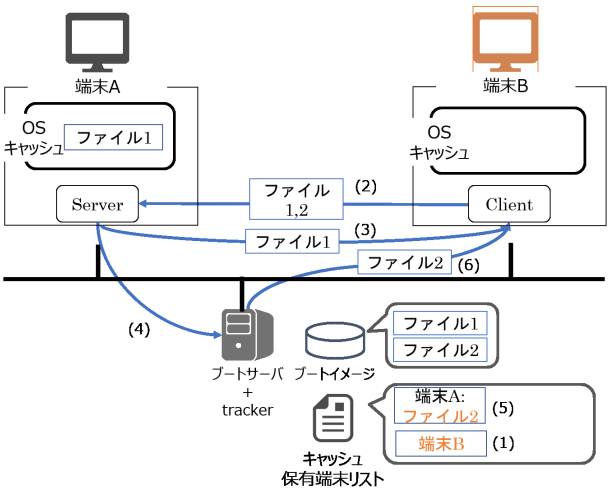

図 2 キャッシュミスが発生した時の起動

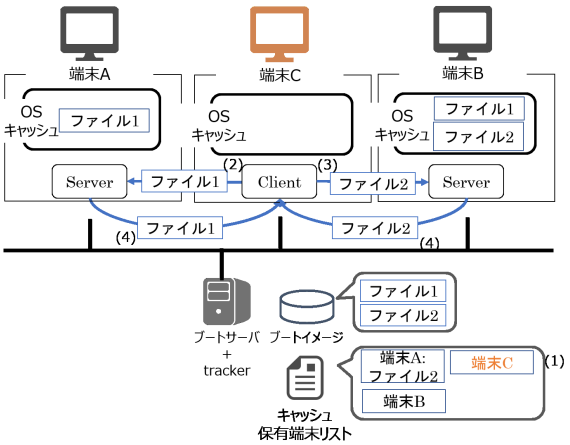

図 3 起動済み端末が複数ある場合の起動

この時，取得されたデータは OS キャッシュとして RAM 上にキャッシュされる (図 1 (4))．

次に，キャッシュミスが発生した場合のフローを (図 2) 示す．まず，端末 B の初期化時に，キャッシュ保有端末リストに端末を登録され (図 2 (1))，ブートサーバが保存しているキャッシュ端末保有リストを取得する．次に，端末 B は，キャッシュ保有端末リストを参照し，端末 A はすべてのキャッシュデータを保持していることを確認する．その後，端末 A に向けて，データ要求を行う (図 2 (2))．端末 A は，自身の OS キャッシュに保存されているファイル 1 を端末 B に送信する (図 2 (3))．しかし，端末 A は，端末 B から要求されたファイル 2 は保有していないので，端末 B としてファイルのデータの代理要求を行う (図 2 (4))．要求を受けたブートサーバは，端末 A がファイル 2 をキャッシュしていないことをキャッシュ保有端末リストに記録する (図 2 (5))．その後，ファイル 2 を端末 B に送信する (図 2 (6))．

最後に，起動済み端末が複数ある場合のフローを (図 3) に示す．まず，端末 C は，初期化時にキャッシュ保有端末リストに端末を登録され (図 3 (1))，キャッシュ保有端末リストを取得する．取得したキャッシュ保有端末リストを参照し，端末 A がファイル 1，端末 B はすべてのファイルをキャッシュしていることを確認する．その後，端末 A にファイル 1，端末 B にファイル 2 の要求を行う (図 3 (2)，(3)) 要求を受け取った端末 A・B は，それぞれの OS キャッシュからデータを取得し，端末 C に応答する (図 3 (4))．

2.2 ブートサーバ機能

他端末からのデータ要求に対して自身のキャッシュを用いて応答を行う，ブートサーバ機能をクライアント端末の OS 内に実現する必要がある．ブートサーバ機能実現するうえで，キャッシュデータの検索，キャッシュデータが書き換わることの検査，キャッシュデータを用いた応答の 3 つの課題を解決する必要がある．

2.2.1 キャッシュデータ検索の課題と解決

キャッシュデータを検索するための課題として，セクタを用いたキャッシュデータの検索がある．一般的な OS では，ファイル毎にキャッシュデータを管理していることが多い．一方，ネットワークブートのプロトコルには，ディスクイメージのデータ要求をセクタ単位で行うものがある．キャッシュデータにセクタ番号に関するデータを保持していない場合，他の端末の要求に対して，自身がキャッシュデータを保有しているかを判断することができなくなる．つまり，セクタ番号をキーとして OS キャッシュのデータを検索できるようにしなければ，他の端末からのデータ要求に対してキャッシュデータを利用することができない問題が発生する．

本課題を解決するため，ファイル I/O 時に，セクタ番号から OS キャッシュ上のファイルを検索するインデックス（以降，検索木）を作成する機能を追加することで対処する．そこで，取得したブロックデータをファイル情報に紐づける結合処理部に着目する．本結合処理部では，ブロックデバイスから取得して来たデータをファイルの一部であるページに結合する処理を行う．本結合処理の段階では，取得してきたデータのセクタ番号を保持しており，さらに，結合対象のファイルに関する情報も取得することが可能である．この時点で，セクタ番号とファイル情報を保存し検索木を作成することで，セクタ番号を用いた RAM 上のキャッシュデータ検索を実現する．

2.2.2 キャッシュデータ利用時の課題と解決

次に，キャッシュデータが書き換わることへの対処の必要性がある．p3cache は，他端末がファイルキャッシュを用いて応答する．しかし，他端末でファイルの書き込みが発生した場合，書き込まれたデータがファイルキャッシュに反映されてしまう．書き込み済みのデータを他の端末に送信してしまうと，同一な環境を構築できなくなってしまう．

そこで，ファイルキャッシュの書き込みフラグに着目する．通常，書き込みがあったキャッシュデータには，ディスクに書き出すため書き込みフラグが挿入される．前述の検索機能で，ファイルキャッシュが見つかった場合にも，書き込みフラグが挿入されている場合には，キャッシュが見つからなかったとみなすことで，この課題に対処する．

2.2.3 キャッシュデータ転送時の課題と解決

最後に，他端末から送られてきたデータ要求に対し，キャッシュデータを用いてサーバと全く同様な応答を作成する必要がある．

そこで，前述した，OS キャッシュ検索により該当ファイルキャッシュを検索し，要求の一部を書き換え，応答を作成し，ファイルキャッシュを封入することで行う．また，要求されたデータがキャッシュデータにない場合には，該当データを要求端末に送信するよう，ブートサーバへデータ要求を行う．

2.3 ブートクライアント機能

本提案機構では，キャッシュデータを保持する端末に対してデータ要求を送信する機能を追加する必要がある．他端末上の OS キャッシュを有効利用するため，複数台存在しうるクライアント端末の中から，必要なデータのキャッシュを保持しているクライアント端末へデータを要求するしなければならない．しかし，各クライアント端末は，他のクライアント端末が保持するキャッシュの情報を知るすべがない．加えて，端末のキャッシュ状態も常に変化する可能性があるため，各端末のキャッシュ状況を一元管理することは難しい．

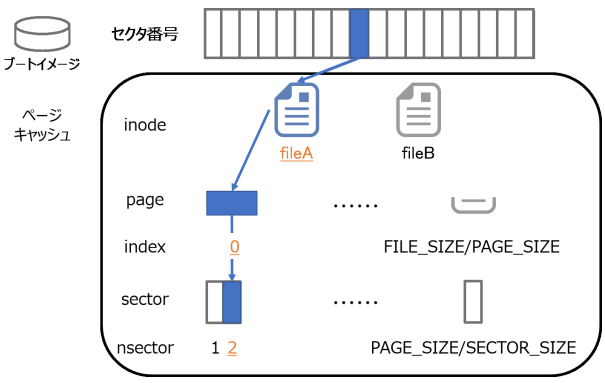

図 4 ページキャッシュの取得

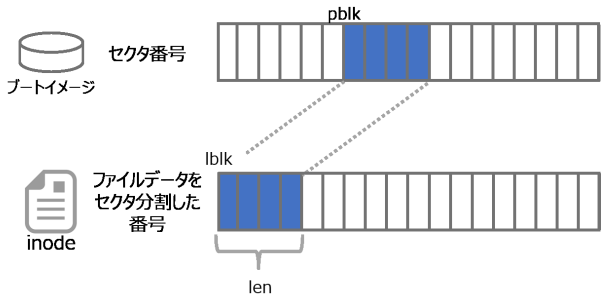

図 5 OS キャッシュ検索木の要素

この課題に対して，BitTorrent のような P2P 機能をブートクライアントに追加することで解決する．ネットブートを行う端末は，まずサーバから，キャッシュを持っている端末のリストを受け取る．それ以降は，サーバではなく，リスト中の端末に向けてすべてのデータ要求を行う．

2.4 tracker 機能

P2P 通信を行うために，ネットブートに参加した端末に対して，各端末のキャッシュデータの保有状態を保存する機能とそれをリストして送信する機能が必要である．しかし，キャッシュデータの管理のために，ネットワーク帯域を利用しすぎると，かえって負荷が集中してしまう．

本提案手法では，端末がキャッシュとして持っていないデータを管理・リスト化することで，データ量を減らす．通常の BitTorrent は，保有しているキャッシュデータを保存する．しかし，ネットブートを行った直後であれば，キャッシュミスが起こるデータのほうが少ないことが予想される．そこで，まず，ネットブートを行った端末はすべてのキャッシュデータを保存していると仮定する．キャッシュミスのため，サーバまでデータ要求が場合にのみ，キャッシュデータを持っていないことを保存しリストとして保存する．

3. p3cache の実装

本章では，Linux と AoE プロトコル [3] を用いた PXE ブートを対象として，p3cache の実現に必要な，ブートサーバ機能，ブートクライアント機能，tracker 機能の実装方法について述べる．

p3cache 実現のために，Linux のページキャッシュ機構にブートサーバ機能中のページキャッシュ検索木作成機構を追加した．加えて，AoE ドライバに追加，キャッシュ検索機能キャッシュを用いて応答する機能，AoE クライアント機能を実装した．さらに，AoE サーバに，tracker 機能を追加した．

3.1 ブートサーバ機能の実装

セクタ番号から，ページキャッシュのデータを取得するために，ページキャッシュ逆引き機能をページキャッシュ検索木により実現する．

3.1.1 ページキャッシュ検索木の構成

Linux では，OS キャッシュなどのファイルキャッシュをページキャッシュという機能で管理・保存している．Linux におけるセクタ番号とページキャッシュの関係を図 4 に示す．

ページキャッシュは主に inode，page によりデータの管理・保存が行われている．inode はファイルの情報や，キャッシュされたデータへの物理アドレスを保存した page へのリンクを持つ．また，ファイルサイズを FILE_SIZE とし，ページサイズを PAGE_SIZE とする，セクタサイズを SECTOR_SIZE とした時，ひとつの inode に対して，FILE_SIZE/PAGE_SIZE 個の page が対応している．更に，page には PAGE_SIZE/SECTOR_SIZE 個分のセクタ番号に対応したデータが保存されている．

p3cache で必要な，セクタ番号に対応したページキャッシュ上のデータを取得するためには，以下の 3 つの情報をたどる必要がある．

- inode： ファイルに対応したページキャッシュへのリンク
- index： ファイル内のデータをページのサイズで分割した時に何番目か
- nsector： ページをセクタのサイズで分割した時の何番目か

以上のデータを検索するために，検索木は以下の要素を持つ (図 5)．

- inode： ファイルに対応したアドレススペースへのリンク
- pblk： 先頭のセクタ番号
- lblk： ファイルデータをセクタ単位で分割した時に，pblk に対応する先頭セクタ番号
- len： pblk，lblk が何セクタ連続するか

検索木は出力として，inode，lblk を返す．検索木から得られた，データを用いたページキャッシュデータの取得は以下の方法で行う．また，1 つのページに含まれているセクタの数を PAGE_PER_SECTOR =

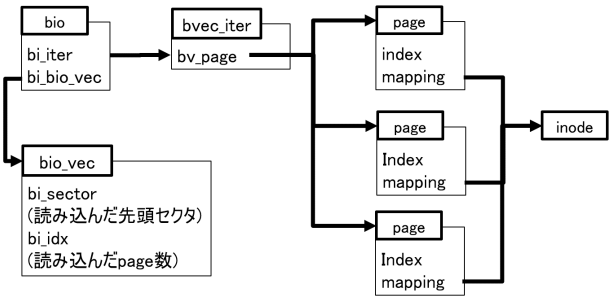

図 6 bio 構造体

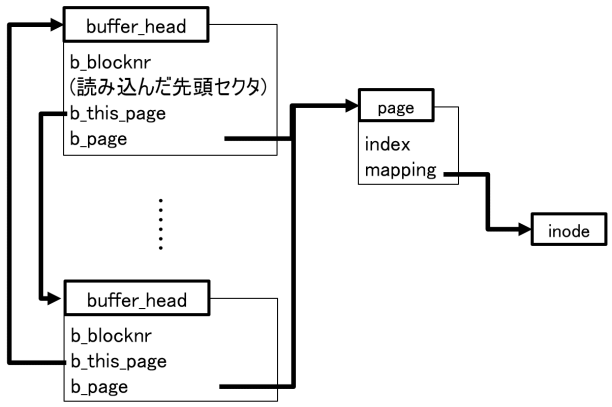

図 7 buffer_head 構造体

PAGE_SIZE/SECTOR_SIZE とする．

- inode： 検索木から返された inode をそのまま使う
- index： (lblk - 要求セクタ)/PAGE_PER_SECTOR
- nsector： (lblk - 要求セクタ)%PAGE_PER_SECTOR

また，検索木の要素の len はページキャッシュの取得には直接利用しないが，連続したブロックをまとめて扱うことで，検索木の要素を減らすために用いる．また，検索木の実装には，赤黒木を用いる．

3.1.2 ページキャッシュ検索木の作成

検索木の作成は，Linux カーネル内のブロック I/O コールバックハンドラに追加実装することで行う．ファイルの読み込み要求には，ページ内で連続したセクタの読み込みとページ内で不連続なセクタへの読み込みがある．

ページ内でセクタが連続している読み込み要求の場合，bio 構造体に結果が保存されコールバックされる．bio 構造体の構成を図 6 に示す．bio 構造体から取得可能なデータを用いて，以下の方法で木に要素を追加する．

- inode： page が持つ inode へのリンク
- pblk： bvec_iter 内の bi_sector そのまま利用する
- lblk： bio_内の先頭ページの index × PAGE_PER_SECTOR
- len： bi_vec 内に保存されている，bi_idx × PAGE_PER_SECTOR

次に，ページ内で不連続な領域への読み込み要求の場合には，page をブロックのサイズで分割して管理する buffer_head 構造体が返される．buffer_head 内のデータ (図 7) を用い，以下の方法で木に要素を追加する．

- inode： page が持つ inode へのリンク
- pblk： b_blkcknr をそのまま利用する．
- lblk： page の index × PAGE_PER_SECTOR+何番目の buffer_head か × SECTOR_SIZE
- len： buffer_head 構造体は，ブロック単位でのみデータを扱うので，必ずブロックの大きさ/SECTOR_SIZE になる．

3.2 書き込み済みキャッシュデータ転送回避の実装

本提案では，OverlayFS を用いることで，書き込み済みキャッシュデータの転送を回避した．OverlayFS は，2 つの FS を合わせて 1 つの FS に見せることができる機能である．ディスクイメージを読み込む専用の FS とユーザからの書き込みを保存する tmpfs を組み合わせた．読み込みと書き込みを FS システム単位で分けることで，ディスクイメージに書き込みが行われることはない．つまり，ネットブートした端末は，RAM に空きがある限りは，ディスクイメージのコピーを持ち続ける．書き込み済みデータを転送しないで済み，他端末に自身のキャッシュを転送する本システムと相性がいいため採用した．

3.2.1 キャッシュデータの一貫性の確保

p3cache は，キャッシュデータの書き込みによる，キャッシュ有効データの減少へ対処するために OverlayFS を利用する．OverlayFS は，2 つの FS を組み合わせて 1 つの FS に見せる技術である．

OverlayFS を用い，書き込みデータと読み込みデータに 1 つずつ FS を割当，それぞれを分離することで，キャッシュデータの一貫性を保つ．また，検索木は，読み込み時に木の要素を追加するため，分離した FS を意識することなく，ブートイメージと同じデータが保存されている読み込みデータの FS のみを検索対象とすることができる．

3.3 応答パケットの作成

AoE サーバ機能に必要な，他クライアント端末へのキャッシュデータを用いた応答は，AoE ドライバに追加の実装を行うことで実現する．通常 AoE ドライバには，応答パケットの処理の機能しかない．そこで，要求パケットから必要なデータを抽出し，ページキャッシュ検索木を用い，ページキャッシュを取得，キャッシュを用いた応答パケットを転送する機能を追加実装する．要求パケットから抽出するのは，セクタ番号を示す LBA で，この値を用いあらかじめ作成された検索木を利用する．要求されたデータがキャッシュされていない場合には，何もせず終了する．要求され

たデータが，キャッシュ済みである場合は，要求パケットの一部を書き換えることで，応答パケットを作成する．

起動済み端末でのキャッシュデータを用いた応答パケットの作成のために，要求パケットを書き換える箇所は，共通ヘッダ内の送信先 MAC アドレス，送信元 MAC アドレス，フラグ，ATA コマンドヘッダ内のセクタカウント，コマンド/ステータス，データである．送信先・送信元アドレスは，応答・要求で反転するため書き換える．フラグは，応答パケットであることを示す 0x18 に書き換える．セクタカウントは，応答時には，0 にする仕様になっているため，書き換える．コマンドは応答時には，ステータスを示す領域になるため，ATA デバイスが正常に動作していることを示す，0x40 に書き換える．データには，キャッシュしたデータを書き加える．

要求されたキャッシュデータを保有していない場合は，要求パケットの送信先 MAC アドレスをサーバのアドレスに書き換え，サーバに全く同じ要求パケットを送信する．以上のパケットの書き換えにより，ブートサーバ応答時と同様のパケットの作成を実現する，

3.4 AoE クライアント機能

AoE クライアント機能は，要求パケットの行き先 MAC アドレスを書き換える機能を AoE ドライバに追加することで実現する．書き換える MAC アドレスは，サーバから取得したキャッシュ保有端末リストをもとに行う．

3.5 tracker 機能

tracker の機能として，キャッシュ保有端末リストとキャッシュミス記録機能を追加した．

キャッシュ保有リストは，ネットブートを行った端末の MAC アドレスのみを保存する．MAC アドレスの追加は，端末がネットブートのためにブートサーバとの通信の初期化を行う時に行う．

キャッシュ記録機能は，キャッシュミスによって発生したサーバへの要求パケットのみを抽出し，記録することで行う．

4. 評価

4.1 キャッシュヒット率

他端末上の OS キャッシュを用いた応答の有用性を確認するため，ネットブート直後の端末上が，他端末に転送可能な OS キャッシュのデータ量を測定した．

実験環境は，端末 1 台，ブートサーバ 1 台をスイッチで接続した．それぞれのスペックを表 1，表 2 に示す．

実験の方法は，まず，端末 1 台が，通常通りネットワークブートを行う．この時，起動中端末からブートサーバへデータ要求されたセクタ番号をすべて保存する．次に，起動終了後，ブートサーバから端末に対してブート時に要求

表 1 端末のスペック

機種名	FMVA10029P
OS	Ubuntu sever 16.04
CPU	Intel(R) Core(TM) i3-4000M CPU @ 2.40GHx
メモリ	4GB

表 2 ブートサーバのスペック

機種名	Lenovo H330/H330
OS	Ubuntu sever 16.04
CPU	Intel(R) Core(TM) i5-2310 CPU
メモリ	8GB
AoE サーバ	vblade

表 3 ブートサーバへのキャッシュ対象パケットの到達数 [Mbyte]

有効	無効	合計
365.1447	165.6095	530.7542

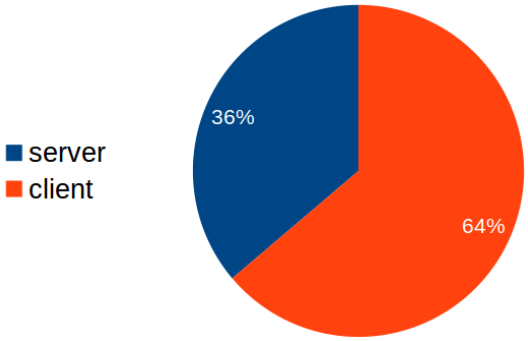

図 8 起動に利用したパケットの割合

されたセクタ番号すべてのデータ要求を行う．最後に，端末からブートサーバへ送られてきた応答内のデータとブートイメージを比較した．端末からの応答の中で，ブートイメージのデータと完全一致するものを有効，ブートイメージとは一致しないもの，応答が帰ってこなかったものは，無効とした．以上の手順を 10 回繰り返し，平均をとった．

実験結果を表 3 に示す．実験により，約 7 割のデータはブートイメージと同じ内容のデータを応答することができていることが分かった．起動直後であれば，約 7 割のデータが他の端末の起動にも有効であるため，ＯＳキャッシュを他端末に転送する本手法は有用であると考えられる．

4.2 ブートに用いられるキャッシュデータの割合

本システムの起動に対する影響を確認するため，本システム利用時の，起動時に用いられるサーバから送信されたデータと他端末から送信されたキャッシュデータの割合を調べる実験を行った．

実験環境は，2 台のクライアント端末と 1 台のブートサーバを用意した．それぞれのスペックは，実験 4.1 と同様である．実験の手順は，まず，1 台のクライアント端末を起動する．次に，1 台目の起動が終了した後にもう 1 台のクライアント端末の起動を行う．2 台目起動時に，起動に用

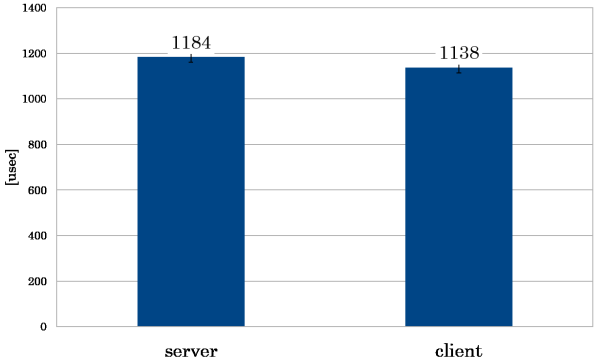

図 9 サーバ，他端末のパケット RTT

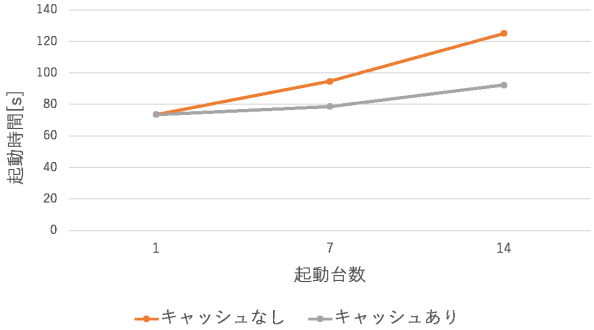

図 10 複数端末同時起動時の OS 起動時間

いられたパケットの送信元を確認することでデータの割合を測定した．起動に用いられたパケットの送信元を核にするため，AoE ドライバ内に，起動に利用したパケット数を数える機能を追加した．

実験結果を図 8 に示す．起動に対して，64%がクライアント端末から転送されたパケットを使用しており，ブートサーバへの負荷分散が期待できることが示された．

4.3 RTT の測定

サーバとクライアント端末内サーバ機構の機能差を確認するため，起動時に用いられた，サーバから送信されたパケットと他端末から送信されたパケットの RTT を測定した．

実験環境・手順は，実験 4.2 と同様で，RTT の測定には，起動している端末の AoE ドライバ内に，サーバ，他端末ごとに起動に利用したパケットの送信から受信までにかかった時間を合計する機能を追加実装し，合計をパケット数で割ることで行った．

実験結果を図 9 に示す．ファイルキャッシュを用いて，応答したほうが，サーバよりも早いという結果が得られた．これは，サーバは，ユーザ空間で動作しているアプリケーションであることに対して，AoE ブートサーバ機能は，カーネル空間で動作しているためだと考えられる．以上の結果より，端末の起動の高速化が期待できることが示された．

4.4 複数端末同時起動時の OS 起動時間

複数台の端末が一斉に起動したときに，OS 起動時間に対してどの程度貢献しているかを確認するために行った．

実験環境は，1,7,14 台の PC とブートサーバを一つのスイッチで接続した．実験の手順は，すべての端末を同時に起動したとから，最後の端末のログイン画面が表示されるまでの時間をストップウォッチ計測した．端末の起動は，Wake on Run パケットをブートサーバから送信することで行った．

実験結果を図 10 に示す．まず，本システムを適応しない場合には，起動する端末が増加するごとに，起動時間も増加することが分かった．これは，サーバへの要求が一時的に集中したためであると考えられる．一方，本提案システムを適応した場合，適応しないときと比べ，7 台の場合は約 17%，14 台の場合は，約 26%削減できていることが分かった．

5. 関連研究

浜本ら [4] は，172 台の教育用 PC をネットブートして，ネットブートの起動に影響を与える要因を調査した．影響を与える要因として，PC の台数，ブートサーバにおけるハードディスクの性能，ネットワーク構成，ブートサーバ数の 4 つを対象とした．調査の結果，教育用 PC の台数が増えると，PC の台数とネットワーク構成が起動時間に影響を与えることが分かった．ネットブートでは，ブートサーバへ要求パケットが早く届いた端末から順にネットワークの帯域を占領する．PC 台数が多い場合，起動が遅くなった PC には，ネットブートを行うために十分な帯域が確保されていない．帯域が確保されるまで，ブートサーバからのデータ取得ができないため，起動時間が遅くなる．ネットワーク構成も，ブートサーバへの経路にボトルネックとなる経路があるような場合には，PC 台数が多い時と同様に，十分な帯域が確保されず，起動時間が遅くなることが分かった．一方，ブートサーバを追加した場合は，各ブートサーバへの負荷が分散され，起動時間が短くなることが認められた．次に，ブートサーバのハードディスク性能は，ネットワークブート時間に貢献しにくいことが認められた．以上より，ネットワークブートにおける起動時間を短縮するためには，サーバへのボトルネックをなくす，ブートサーバを追加することが効果的であることが示された．ネットブートにおけるブートサーバへの負荷をローカルディスクへのキャッシュにより解決する手法として，八田ら [1]，表ら [2] の研究が挙げられる．まず．八田ら [1] は，クライアント端末のローカルディスクに独立したパーティションを作成して，ブートサーバから取得したデータのキャッシュを行っている．一方，表ら [2] は，仮想マシ

ンモニタが，得られたブートイメージをクライアント端末のローカルディスクに書き込むことで，キャッシュを行っている．これらの手法は，ブートサーバへの改造は行わないが，クライアント端末のローカルディスクを利用するため，PC 環境の変更が望まれない BYOPC では，利用しにくいという問題があげられる．また，端末同士がキャッシュデータの共有を行っていない点で，本研究とは異なる．

ネットブート時に，BitTorrent[5] を用いて他端末上のディスクデータを用いてネットブートを行う研究に，BootTorrent[6],mobbi[7] がある．BootTorrent は，まず tftp を用いて，軽量な OS をネットブートする．その後，軽量な OS は，実際に構築したい OS のイメージをサーバや起動済みの他端末から取得する．最後に，構築したい OS を qemu や kexec を用いて展開することで，統一された環境を構築する．mobbi は，Linux の起動時に，BitTorrent を用いてルートファイルシステムを取得するよう改造を加えることで，他端末とのディスクイメージ共有を実現している．BootTorrent，Mobbi ともに，BitTorrent を用いてサーバだけではなく他の端末からもディスクイメージを取得することで，複数台の端末が同時に起動する場合でも，起動時間を減らすことに成功している．本研究とは，BitTorrent を使うために，RAM 上に新たなキャッシュ領域を必要とする点で異なる．本研究手法は，OS が本来利用しているキャッシュ領域を利用して，データ共有を行うため，キャッシュデータの無駄がない．一方，BitTorrent を用いる場合，OS が有しているキャッシュ領域のほかに，キャッシュ領域を必要とする．加えて，すべてのディスクデータを取得し終わるまでファイルとしてみなすことができないため，必要以上に RAM を消費する可能性がある．

6. おわりに

本研究は，ネットブート型シンクライアント端末郡における，ディスクデバイスを必要としない，ブートサーバへの負荷分散を目的として，端末上で起動した OS が持つキャッシュに着目し，各端末のメモリ上にキャッシュされているディスクデータ片を必要とする近隣の他端末へ配信することでサーバに対する負荷の削減を行った．

本システムを実現するために課題とされる，セクタ番号を用いたファイルキャッシュの検索は，ファイルキャッシュ作成時に，必要なデータを取得し，ファイルキャッシュ逆引き木を作成することで解決した．また，キャッシュを保有している端末へのデータ要求は，BitTorrent like な機能を追加することで解決した．

今後の課題として，分散ハッシュテーブルを用いた，キャッシュ管理や，より早い段階で分散協調キャッシュを行う手法の検討が挙げられる．

参考文献

[1] 八田直樹，丸山　伸，松川正義，西村浩二，相原玲二：ネットブート環境における読み込みキャッシュ機構の改善による起動時間短縮の試み，研究報告インターネットと運用技術（IOT），Vol. 2012, No. 14, pp. 1–6（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/170000069962/⟩ (2012).

[2] 表　祐志, 品川高廣, 加藤和彦：仮想マシンモニタによる透過的ネットワークブート方式, 情報処理学会論文誌コンピューティングシステム（ACS），Vol. 4, No. 4, pp. 228–245（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/40019259272/⟩ (2011).

[3] Coile, B. and Hopkins, S.: The ATA over Ethernet Protocol, *Technical Paper from Coraid Inc* (2005).

[4] 浜元信州，三河賢治，青山茂義：教育用パソコンのネットワークブート起動時間に影響を与える要因の評価，学術情報処理研究，No. 15, pp. 46–52（オンライン），入手先 ⟨https://ci.nii.ac.jp/naid/40019778283/⟩ (2011).

[5] Cohen, B.: Incentives build robustness in BitTorrent, *Workshop on Economics of Peer-to-Peer systems*, Vol. 6, pp. 68–72 (2003).

[6] Trentini, A. and bruschi, d.: BootTorrent: peer-to-peer fast network boot (2015).

[7] McEniry, C.: Moobi: A Thin Server Management System Using BitTorrent., *LISA*, pp. 253–260 (2007).