2007－OS－105 (21)
2007／4／6

HTTP-FUSE-cloopによるP2Pモバイルシンクライアントシステム

金井　遵[†]　須崎有康[††]
八木豊志樹[††]　並木美太郎[†]

本稿では，DNSとP2P技術を用いた，モバイル可能な小型組込み機器とHTTP-FUSE-KNOPPIXによる，負荷分散機構や耐故障性を有するスケーラブルなシンクライアントシステムについて述べる．本研究では，キャッシュサーバ，RAID，DNSなどを用いた負荷分散，耐故障性向上技術に加えて，DNSにP2P技術を組み合わせることにより，小型機器によるサーバにおいても高いスケーラビリティを実現した．特にP2P技術により，シンクライアントシステムでボトルネックとなりやすいクライアントの同時起動時の性能低下を，クライアントが同時に起動用サーバとなることで緩和した．また，多くのハイブリッドP2Pシステムでボトルネックとなる検索システムを軽量化することで，組込み機器など資源が制約されるサーバ環境においても十分なスケーラビリティを実現した．これらの機能を実装した組込みシンクライアントサーバHTTP-FUSE-KNOPPIX-BOXにおいて，クライアント8台で起動時間98秒と多数のサーバを利用したシンクライアントシステムと遜色ない性能を実現した．

P2P Mobile Thin-Client System by "HTTP-FUSE-cloop"

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

This paper describes an implementation of mobile thin-client system with scalability and fault-tolerance by DNS and P2P technology for embedded hardwares and HTTP-FUSE-KNOPPIX. DNS-Balance with P2P technology realizes more scalability by embedded servers in addition to cache servers, RAID, and DNS. The P2P system improves bootstrap performance by clients which distribute block files for other clients when many clients boot simultaneously. The mobile embedded servers with limited hardware resources have sufficient scalability by lightweight search system. Evaluation of embedded HTTP-FUSE-KNOPPIX-BOX with the P2P system has shown that it boots up 8 clients in 98 seconds and its performance holds up to existing thin clients.

1. はじめに

近年，ネットワーク環境やPC性能の向上により，システムソフトウェアを含めたすべての情報をネットワークで一括管理するシンクライアントシステムを採用する機関が増加している．現在は企業や高等教育機関での利用が中心であるが，今後，システム管理の容易性から，High Performance Computing(HPC)やモバイル環境でのシンクライアントシステム，家庭内システム，教育用システム，情報端末など複数のクライアントに環境を提供する必要がある場合への要求が考えられる．しかし，従来のシステムは高性能で高コストなサーバ環境でのサービス提供が中心であり，全ての目的での利用に適するとは言い難く，普及を妨げている．

† 東京農工大学
Tokyo University of Agriculture and Technology
†† 独立行政法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

そこで，我々は先行研究において，軽量にシンクライアント環境の負荷分散・高速化や耐故障性を向上するさまざまな手法を提案してきた．本稿では，P2P技術およびDNSを利用したシンクライアントシステムの高速化・負荷分散手法を提案し，耐故障性を有しつつ，さらなるスケーラビリティを有するシステムを実現した．さらに，HTTPサーバ上のデータからブートするLinux，HTTP-FUSE-KNOPPIX[1)]をネットワーク経由で起動させ，シンクライアントを実現する低コストで持ち運び可能なサーバ，HTTP-FUSE-KNOPPIX-BOXを改良し，これらの機能を実装することで，小型サーバにおいても十分なスケーラビリティや耐故障性機能を実現した．本稿では，これらの設計，評価について述べる．

2. HTTP-FUSE-KNOPPIX

HTTP-FUSE-KNOPPIXは，ネットワークブロックデバイスHTTP-FUSE-cloopを利用したネットワークブート型のシンクライアントシステムである．従来

の NFS によるシンクライアントシステムと比べると，HTTP-FUSE-cloop は HTTP を介するため，ファイヤウォール越えが容易で多くの環境で動作可能であり，広域にサービスが提供可能である．また，HTTP 自体が軽量なプロトコルでサーバを圧迫しない，サーバの繋ぎ換えが容易といった利点がある．

HTTP-FUSE-cloop では，ディスクイメージを分割，圧縮し，HTTP サーバ上に置く．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 がブロックファイルをオンデマンドで読出し，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 または SHA1 により決定され，同一ファイル内容であれば同じファイル名になり，サーバのディスク容量の節約ができる．ブロックのディスク内位置とファイル名の対応は，HTTP サーバ上のインデックスファイルが利用される．

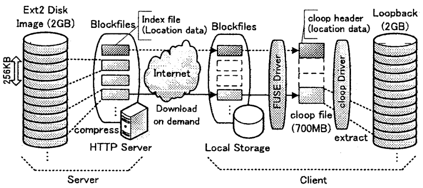

図 1　HTTP-FUSE-cloop

3. 先行研究と問題分析

シンクライアントシステムにおいては，スケーラビリティの実現が重要である．また，シンクライアントシステムにおいてはサーバが単一障害点となるため，サーバ自体を冗長化し，故障の発生時には故障検出や切り替え，復帰などを迅速に行い，可用性を高める必要がある．これらのため，従来のシンクライアントシステムは大規模化していた．そこで，我々は先行研究[2)3)4)]において，OS などを含めたソフトウェアデータを配信する複数のキャッシュサーバを用意し，図 2 のように DNS(DNS-Balance 改) とソフトウェア RAID を介して，定期的にクライアントの接続サーバを変更することで動的な負荷分散を行い，軽量にスケーラビリティと耐故障性を実現する手法を提案してきた．また，これらの機能を組み込んだ小型の組込み機器をサーバ化し，シンクライアントシステムの小型化，低コスト化，モバイル化などを実現した．

我々のシステムにおいて，小型シンクライアントサーバを複数用意し，負荷分散を行うことでスケーラビリティを実現していた．サーバ 4 台で 50 台ほどのクライアントによるシンクライアントシステムを実現したが，さらなるスケーラビリティや高速化を実現するためには，サーバ数をさらに増やす必要があり，モバイル化，低コスト化の妨げになる．モバイル性を実現するためには，サーバを増やさずにスケーラビリティを実現する必要がある．

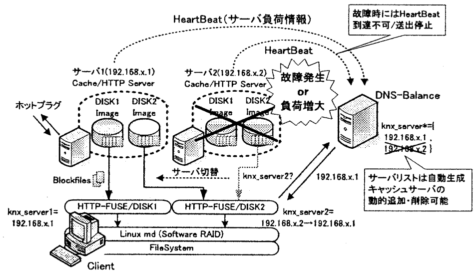

図 2　DNS と RAID によるスケーラビリティと耐故障性

4. 本研究の目的

本研究では，モバイル可能な組込み機器のような低スペックなハードウェアおよびネットワーク接続環境で，一定台数以上にサーバを増やさずに十分なスケーラビリティや耐故障性を持ったシンクライアントシステムを実現することを目的とする．このことは，会議などでのモバイル環境でのシンクライアントシステム，家庭内のシステム，教育用システム，情報端末，HPC 環境[6)] など複数のクライアントに環境を提供する場合のほか，インターネットから OS などを入手する情報家電などのユビキタスへの応用，さらには高性能，高信頼なハードウェアの利用が常識であった従来のシンクライアントシステムの小型化，低消費電力化，省コスト化を実現できる．そこで，本研究では P2P 技術と DNS による負荷分散・高速化，耐故障性向上手法を提案し，実際にモバイル可能な小型組込みシンクライアントシステムを実現し，評価を行う．

5. P2P によるスケーラビリティの実現

我々はキャッシュサーバと DNS，RAID を組み合わせた高速化手法について提案してきた[2)4)]．この手法でサーバの負荷分散を行えるものの，さらなるスケーラビリティを実現するためには，P2P のようなサーバにトラフィックを集中させない手法が有効である．この手法は，特に本研究で想定するような低スペックなサーバで効果を発揮すると予想できる．そこで，クライアントにおいてもキャッシュサーバによるブロックファイル配信を可能にし，DNS-Balance 改に P2P ノードの割り当て機能を実装することで，ハイブリッド P2P システムに似たトラフィック分散を実現した．

5.1 概　　要

多くのシンクライアントシステムにおいてボトルネックとなるのはクライアント起動時のトラフィック[14)] である．特に，クラスタシステムやキオスク端末などでは，あらかじめ実行するソフトウェアが決まっており，起動後のディスクアクセスのトラフィックは少なく，この傾向は顕著である．つまり，起動時にサーバに集中するトラフィックをさらに分散させることで，サーバシステムを簡略化する事ができ，特に本研究で想定するような組

込みサーバなど，資源が貧弱な環境でも十分なスケーラビリティを実現できる．

そこで，我々は HTTP-FUSE-KNOPPIX によるシンクライアントシステムにおいて，クライアント側では起動時に多くの同一ブロックファイルをサーバから取得することに着目した．先行して起動しているクライアントが取得したブロックファイルを，ほかの起動中のクライアントに対して配布することで，サーバの負荷を軽減することが可能である．これは，ファイルの位置情報のみをサーバが管理するハイブリッド P2P システムに近い．

5.2 設　　計

今回，P2P による負荷分散機能を他の負荷分散・耐故障システムと併せて利用できるように，Squid による多重キャッシュサーバと DNS-Balance を利用して，P2P システムを実現した．本システムでは，P2P システムにおける検索システムを DNS-Balance が，クライアントにおけるデータ配信サーバ機能を Squid による多重キャッシュサーバが担う．

DNS-Balance は，図 3 のようなクライアント同士の接続関係を表す，木構造を動的に生成する．この接続関係に従いクライアント同士でブロックファイルのやりとりを行うことで負荷分散を行う．接続関係の生成は DNS-Balance がクライアントからハートビートを受け取ることにより行う．クライアントでは DNS-Balance に対して定期的に，上位クライアントの名前解決を行い，ブロックファイルの取得先を得る．さらに，ローカルのキャッシュサーバを介して，上位クライアントのキャッシュサーバに対してブロックファイル取得を試みる．キャッシュミスした場合には，さらに上位のクライアントまたは負荷分散用キャッシュサーバに対してブロックファイル取得を試みる．多くの P2P システムと違い，キャッシュサーバの特性上，キャッシュミスした場合でも下位のクライアントは意識する必要なしに，上位のキャッシュサーバにファイルを取りに行くため，厳密にどのクライアントがどのブロックファイルを持っているかを管理する必要はない．これにより，ハイブリッド型 P2P システムでボトルネックになりやすい検索システムを簡略化できるほか，クライアント側でも検索のための問合わせ回数を大幅に削減できる．しかし，起動時間を短縮し，トラフィック総量を抑えるためには，できるだけ適切な接続関係を構築し，キャッシュヒット率を上げる必要がある．そのための木構造の生成ルールとして以下のような指針を設けた．

- 先行起動クライアントからブロックファイルを取得する
  クライアントの起動段階を 10 段階に分け，起動段階が先行しているクライアントからブロックファイルを取得することにより，キャッシュヒット率を上げることができる．基本的に起動終了までは起動時に利用したブロックファイルを削除しないようなキャッシュサイズを指定するが，大きなキャッシュサイズを用意できないマシンのために，キャッシュミスを防ぐため，起動段階の近いクライアントに優先して接続するようにする．起動終了後には，他のデータ転送を行い，起動時のブロックファイルについてはキャッシュミスの可能性が上がるので，リストから外す．
- 可能な限りトラフィックが偏らないようにする
  クライアントは基本的には起動が先行しているクライアントに接続するが，接続可能クライアント数の上限を設定し，一定数以上のクライアントが接続され性能が落ちることがないようにする．このとき，各クライアントの下位接続クライアント数は可能な限り一定になるようにする．接続可能な先行クライアントが無い場合には，同起動段階のクライアントに対して接続を行う．この場合，キャッシュミスの可能性は増えるが，後ほど親ノードになったクライアントや他の子ノードもそのファイルを利用する可能性が高く，その時にはキャッシュにヒットするので無駄にならない．なお，同段階のクライアントの接続最大数の制限は設けず，2 ホップ先には起動が先行しているクライアントが存在するようにし，キャッシュヒット率を上げる．
- 経由ノード数は最小にする
  上位のクライアント A でキャッシュミスが起きた場合，そのクライアント A が，さらに上位のクライアント B に対してデータの取得を行う．クライアント B でも，キャッシュミスが起きた場合には，さらに上位のクライアントには取りにいかず，直接負荷分散用サーバからデータを取得する．つまり最大 3 ホップで，必ずデータを取得できるようにする．これは，トラフィック発生を抑え，ブロックファイル取得のレイテンシを小さくするためである．また，データ改ざんノード検出を容易にする意味もある．

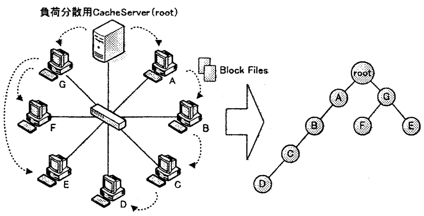

図 3　クライアント同士の接続関係

上記の指針に基づき，クライアント割り当て木構造の生成は一定時間毎に DNS-Balance が行う．各クライアントの起動段階の管理は各クライアントから受け取るハートビートデータを利用する．木構造生成のアルゴリズムを図 4，接続ノード決定アルゴリズムを図 6 に，木の生成例とデータの流れを図 5 に示す．

木構造の生成の計算量は最悪 $O(n^2/2)$ である．特に，接続ノードを決定する get_connect_node() では同段階のノードが多い場合に計算量が増え，n 台のノードがすべて同段階ノードの場合，計算量が最悪値となる．

```
#負荷分散用サーバ数= r, 負荷分散用サーバ接続可能数= s
#各クライアント接続可能サーバ数= c, 各段階のクライアント数= L(n)

#Level ≧ lv で接続可能数最大ノード取得関数
#type=NODE_ALL:全ノード探索 type=NODE_PARENT:親ノードのみ探索
(Level, Nodenum) = get_connect_node(lv, type)

#負荷分散用キャッシュサーバデータ
node(99, 1..r) ← serv
i ← r * s          #上位段階の接続可能ノード数
for j = 10..0      #各起動段階について
  a = L(j) * c     #同段階の接続可能ノード数
  for k = 1..L(j)
    if i > 0 then #上位段階接続可能ノード有：自分は同段階内の親ノード
      (lv, m) = get_connect_node(j + 1, NODE_ALL)
      node(j, k).type ← PARENT
      i ← i - 1
    else           #上位段階接続可能ノード無：自分は同段階内の子ノード
      (lv, m) = get_connect_node(j, NODE_PARENT)
      node(j, k).type ← CHILD
      if node(lv, m).nchild >= 0
        a ← a - 1
      endif
    endif
    node(j, k).parent ← node(lv, m)
    node(j, k).nchild ← c
    #同段階親ノードでは負数可
    node(lv, m).nchild ← node(lv, m).nchild - 1
  i ← i + a
```

図 4 割り当て木構造生成アルゴリズム

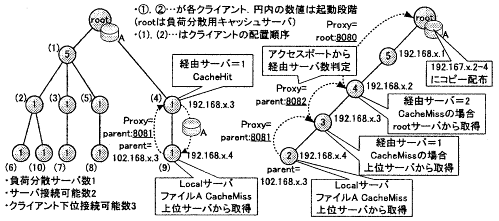

図 5 木構造の生成と経由サーバ

なお，本システムでは HTTP-FUSE-KNOPPIX の起動段階を表 1 に示す 11 段階に分割した．起動に用いられるスクリプト（linuxrc，knoppix-autoconfig，xsession など）中から/tmp/hb.state ファイルに対して起動段階を書きこむ．ハートビート送信プログラムが更新を検出し，DNS-Balance に対して起動段階の更新を反映する．

5.3 不正ノードの検出

本方式では，クライアントがサーバともなるため，データ改ざん等，悪意を持ったノードの検出が必要である．データの改ざん検出については各クライアントが行う．HTTP-FUSE-cloop では，ブロックファイル名はファイルのハッシュ値により決定され，ディスク内の位置とブロックファイル名の関係はインデックスファイルに記述されている．インデックスファイルに関しては，信頼できる負荷分散用キャッシュサーバから取得し，他のクライアントからブロックファイルの取得を行った際に，ファイルのサイズ，ハッシュ値の計算を行い，インデックスファイルのデータと比較する．改ざんが検出された場合，DNS-Balance に対して改ざんノードの検出メッセージの送信を行い，DNS-Balance ではそのクライアントが割り当てられているクライアントと，そのもう一つ上位のクライアントをノードリストから削除し，次回の木生成以降利用しない．クライアント側では，定期的に DNS の再問い合わせを行うので，データ改ざんが起きた場合でも，下位のノードでは意識せずに動作の継続が可能である．

表 1 起動段階

起動段階	段階 0 からの経過時間例 [s]	累計通信量例 [MB]	設定箇所
0	0	0.0	NIC 設定完了/ハートビート送出開始
1	10	0.2	HTTP-FUSE-cloop 起動終了
2	16	8.4	ディレクトリ再構成完了
3	22	24.0	初期設定コピー完了
4	26	25.0	ハードウェア自動検出開始
5	40	26.6	ハードウェア自動検出中 (1)
6	48	30.4	ハードウェア自動検出中 (2)
7	55	34.7	ハードウェア自動検出完了．X 起動開始
8	65	41.4	KDE 起動開始
9	72	50.3	KDE スプラッシュウィンドウ表示
10	94	65.1	起動終了

```
function get_connect_node(level, mode)

if mode = NODE_ALL then
  steplist = (10..level, 99)
else
  steplist = (level)
endif

r      ← -10000    # 暫定接続先接続数
r_node ← (-1, -1)  # 暫定接続先

for step in steplist
  for i = 1..L(step)
    if (node(step, i).nchild >=r && mode = NODE_ALL) ||
       (node(step, i).nchild >=r && mode = NODE_PARENT &&
        node(step, i).type = PARENT) then
      # 前回接続サーバを優先するコードも入る
      r ← node(step, i).nchild
      r_node ← (step, i)
    endif

return r_node
```

図 6 接続ノード決定アルゴリズム

5.4 耐故障性

故障ノードや接続断が発生した場合には，ハートビート送出が停止され，木が再構成される．そのため，DNS の再問い合わせにあわせて接続サーバが切り替わり，下位のノードでは意識せずに動作の継続が可能である．また，負荷分散用キャッシュサーバの故障については，複数の負荷分散用キャッシュサーバを用意することで解決できる．

ここで，ルートキャッシュサーバや，ノード故障・電源断時の接続切り替えによるクライアントの通信停止時間 T_{stop} について考察する．図 7 のように，$T_{tout} = T_{ttl}$ かつ，$T_{tout} > T_{svdel} + T_{gentree}$ のとき，DNS 再リクエスト時に木構造が最新に更新されているため，$T_{stop} = T_{tout}$ となる．

5.5 システムの構築例

本システムでは，DNS リクエストに比べブロックファイルの転送量が圧倒的に多いため，LAN 内で大規模なシンクライアントシステムを構築する場合，スイッチなどのネットワーク機器に集中するトラフィックがボトルネックになると考えられる．大規模なシステムを実現し

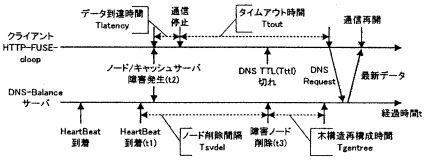

図 7 キャッシュサーバ・ノード故障時のタイミングチャート

たい場合，図8のようにネットワークセグメントをわけることでトラフィックを明確に分離し，大元のイメージ配信用サーバと，複数のルータ機能を持つDNS-Balance兼キャッシュサーバにより運用するような方法が最適である．このため，一つ一つのサーバが実現するスケーラビリティは本システムのようなハイブリッドP2Pシステムで数十台〜百台程度を実現すれば十分であると考える．

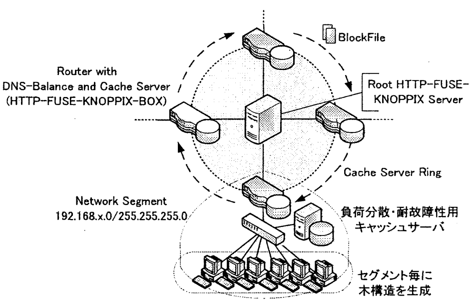

図 8 大規模なシステムの運用例

本システムは，WANで運用することも可能である．しかし，WANでPXEブートは不可能であること，キャッシュサーバを近くに置いた方が性能的に有利であること，システム自体が安価であること，セキュリティの問題などから，広域でシステムを運用する場合には，一次サーバのみをWAN側に設置し，LAN内にP2P用の小型サーバを置くような運用が好ましい．

また，不正ノードの検出で述べたように，インデックスファイルのみは信頼できる負荷分散用ルートキャッシュサーバから取得する．インデックスファイルのキャッシュ期限を短く設定することで，更新を迅速に伝搬することが可能である．

5.6 スケーラビリティに関する考察

本システムにおいては，スケーラビリティの実現を妨げる以下のような項目について解決する必要がある．

- DNS自体のスケーラビリティ
- サーバに集中するトラフィック
- 各ノードに集中するトラフィック

このうち，DNSの負荷については，DNSリクエストに対する応答性および割り当て木構造の生成時間が重要である．この評価については，評価の章において実測値を計測し，詳しく述べる．

また，サーバに対するトラフィックについては，サーバに直接接続するノードのトラフィックに加え，キャッシュミスによるトラフィックの発生がある．キャッシュミスが起きるのは，機種依存のドライバなどのファイルリクエストなどに加えて，起動段階の逆転によって，木構造と実際の起動段階の不一致が発生した場合である．つまり，最大経由キャッシュサーバ数が3で，接続関係上での起動段階がノードA>ノードB>ノードCだった場合に，実際の起動段階がノードC>ノードB>ノードAのように逆転した場合，ノードCが共通ブロックファイルのリクエストを行うと，ノードA，ノードBともにキャッシュを持たないため，ルートキャッシュサーバからデータを取得することになる．特に，図9のような構成で起動段階の逆転が起きた場合にキャッシュミスのトラフィックが最大となり，最大で(1)式のトラフィック発生があり得る．ここで，n_{step}は起動段階分割数，n_{via}を最大経由キャッシュサーバ数，n_{root}をルートキャッシュサーバの下位接続接続最大数，aを1台のクライアント起動時に発生する平均トラフィック量，bをクライアント毎に固有の平均トラフィック量，Nをクライアント数とし，$N/n_{root} > 2n_{step}$とする．

$$Tr_{all} = (INT((n_{step} \times 2 - 1)/n_{via}) + 1) \times n_{root} \times a + N \times b \qquad (1)$$

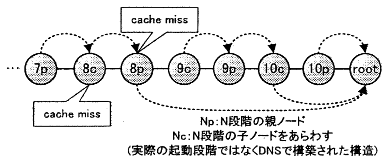

図 9 サーバに最もトラフィックが集中する構成

このうち，システムに存在するマシンが異なる場合を想定し，機種依存のトラフィックが占める割合を計測した結果を表2に示す．表2の結果は，ノートPC(PentiumIII 866MHz)，自作PC（Athlon 1800+）と，構成の大幅に異なる構成の計算機を利用しているが，トラフィック全体に占める機器依存のトラフィック量は4%以下と少ない．(1)式とこの値より，サーバに集中するトラフィック量は図10のように予想できる．

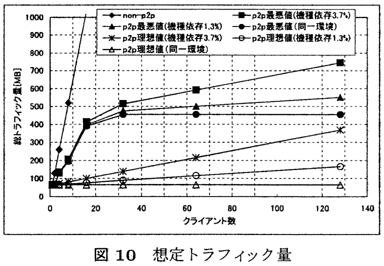

図 10 想定トラフィック量

機種依存ファイル以外のキャッシュミスによるトラフィック発生を抑えるためには，起動段階が逆転した場合にはそれを速やかに木構造に反映し，クライアント側もそれを検知する必要がある．よって，起動段階の逆転

が発生しやすい箇所では段階分割の粒度を細かくすることや，DNS の TTL を短くすることが有効である．実験より，同程度のマシン構成において逆転が発生しやすい箇所は，ハードウェアの自動検出であり，表 1 の起動段階では実際に細かく分割している．これにより，起動段階数 11，最大経由サーバ数 3 の実験環境においては最大で上位ノードでキャッシュミスが発生しないの理想的な状態の 8 倍のトラフィック発生があり得るが，予備実験では，実際に起動段階逆転によるトラフィック発生はほとんどなかった．つまり，理想値である，

$$Tr_{all} = n_{root} \times a + N \times b \qquad (2)$$

に近い数値となっている．特にクライアントが 100 台程度のとき，固有ファイルが 4%程度有った場合でも，起動時にサーバに集中するトラフィックは 300MB 以下と，P2P を用いない場合の 4 台程度のトラフィックと同等となり，十分なスケーラビリティを実現できると予想できる．

クライアントに集中するトラフィックについては，$a >> b$ かつ $N/n_{root} >> n_{client}$ より，起動段階が同段階に集中する場合に最悪になると考えられ，その場合の親ノードのトラフィックは，

$$Tr_{all} = (N/n_{root} - 1) \times (a + b) \qquad (3)$$

で表すことができる．しかしながら，$n_{root} = 4$ 程度の場合でもクライアント 100 台で親ノードに接続する子ノードは 24 台程度と，一般的なシンクライアントサーバに集中するトラフィックに比べると小さい．

6. HTTP-FUSE-KNOPPIX-BOX

我々は，HTTP-FUSE-KNOPPIX を起動可能にする小型シンクライアントサーバとして，HTTP-FUSE-KNOPPIX-BOX を開発している．HTTP-FUSE-KNOPPIX-BOX は，クライアントをネットワークブートさせるための TFTP サーバ，DHCP サーバ，HTTP サーバまたはキャッシュサーバ，負荷分散，耐故障性を実現する DNS-Balance などからなり，通常の PC から組込み向け環境など，Linux が動作する環境であれば動作可能である．今回，従来機能に加え，P2P 機能を加えた DNS-Balance を HTTP-FUSE-KNOPPIX-BOX に組み込んだ．

今回，評価で主に利用した HTTP-FUSE-KNOPPIX-BOX は SH4-266MHz を搭載しており，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．利用方法は従来と変わらず電源を投入し，クライアントを LAN 接続するのみで利用可能である．複数の HTTP-FUSE-KNOPPIX-BOX による負荷分散も，電源を投入するのみで自動的に P2P 機能と併せて利用でき，移動先でアドホックに負荷分散機構・耐故障性を持つシンクライアントシステムを構築するような利用方法も可能である．利用用途として，ハイパフォーマンスコンピューティング，会議や展示会などモバイル環境での一時的なシンクライアントシステム，家庭内のシステム，教育用システム，情報端末などとしての利用が考えられる．

表 2　機器依存トラフィック（キャッシュミスのトラフィック）

	総量 (a)	固有ブロックファイル (b)	割合 (b/a)
マシン A	634 ファイル/63.5MB	8 ファイル/0.8MB	1.3%/1.3%
マシン B	645 ファイル/65.1MB	19 ファイル/2.4MB	2.9%/3.7%

7. 評　　価

HTTP-FUSE-KNOPPIX-BOX による P2P シンクライアントシステムの評価を行った．本章ではこれらの詳細について述べる．

7.1 P2P DNS-Balance の基礎評価

P2P DNS-Balance の基礎評価として，DNS リクエストに対する応答性能，クライアントの接続関係を表す木構造の生成時間の測定を行う．特にハイブリッド P2P システムでは検索システムの負荷がボトルネックになることが多く，スケーラビリティの考察で述べたように，スケーラビリティを実現するためにはこれらの値が重要である．

まず，DNS の応答性能については，クライアント台数を変えた場合を想定し，同時に複数の DNS リクエストを発行し，応答時間の計測を行った．結果を図 11 に示す．また，専用スレッドで定期的に行われる木構造の生成については，想定されるクライアント台数を変え，実際に構築にかかる時間を計測した．構築には，クライアントの起動段階の分布で計算量が変化するため，起動段階の分布を同一にした場合（計算量が最悪になる場合）と，ランダムにした場合について計測を行った．結果を図 12 に示す．

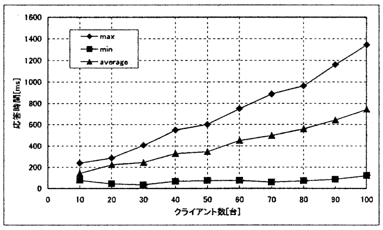

図 11　DNS 応答時間

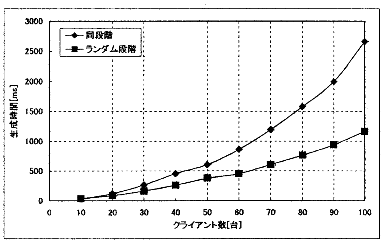

図 12　構成生成時間

1 つの DNS リクエストの処理時間はクライアント数

にかかわらず $O(1)$ であるが，リクエスト集中により，クライアント増加による性能低下が起きている．また，構成の生成時間は $O(n^2)$ であり，クライアント数の増加と共に構成時間が増加している．

ここで，構成の生成時間については DNS の TTL 以下であることが望ましい．これは，DNS の再リクエスト毎に，構成を最新状態に更新しておくためである．また，DNS 応答時間が TTL を上回ると，リクエストの飽和を起こす可能性があるので，DNS 応答時間が TTL 以下である必要がある．先行研究では性能面，故障発生時の復帰時間などの条件から，TTL は 3 秒に設定していたが，クライアント 100 台程度までであれば，応答時間の最悪値，生成時間ともに 3 秒を下回っており，十分なスケーラビリティを実現できると考えられる．

なお，現在，DNS-Balance 改の実装はスクリプト言語 Ruby により行っているが，C などネイティブに実行できる言語を利用することで，応答時間，生成時間ともに，著しい性能改善を実現できることを確認しており，さらなる性能改善とスケーラビリティの実現が可能である．

7.2 P2P DNS-Balance によるスケーラビリティの評価

実際に複数のクライアントを利用し，P2P によるスケーラビリティ実現の評価を行った．サーバとして SH4 による小型組込みモバイル HTTP-FUSE-KNOPPIX-BOX を用意し，クライアント 8 台を，同時起動，10 秒毎に 1 台ずつ起動，30 秒毎に 1 台ずつ起動し，P2P 有効の場合と無効の場合（サーバ数は 1 台），さらに従来のサーバ 4 台による負荷分散方式（P2P 機能は無効），従来方式との比較として NFS による KNOPPIX（これのみ，サーバは 800MHz の CPU による x86 機）で起動時間の計測を行った．クライアントの起動時間（最良値，最悪値）の計測結果を図 13 に，サーバのトラフィックを図 14，図 15 に示す．負荷分散キャッシュサーバの接続最大数を 1，クライアントの下位ノード最大接続数を 5 としている．

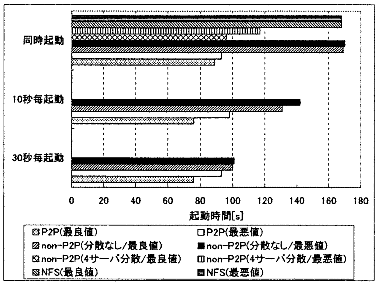

図 13 P2P 利用時の起動時間

P2P を用いると起動時間の最悪値が 98 秒と，P2P を用いない場合の 170 秒に比べて大幅に高速化できた．クライアント 1 台のみでの起動時間は 95 秒であり，従来性能が落ちやすいクライアントの同時起動でも十分な性

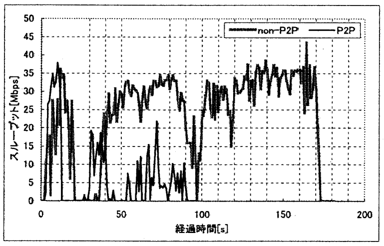

図 14 P2P 利用時のサーバトラフィック

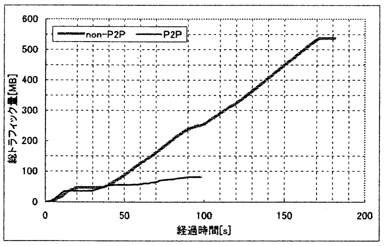

図 15 P2P 利用時のサーバトラフィック（総量）

能を実現可能である．さらに，従来のサーバを複数利用した負荷分散方式に比べても高速な起動時間を実現している．また，従来の NFS を利用したシンクライアントシステムよりも十分高速である．負荷分散キャッシュサーバの接続最大数が 4 台まではほとんど起動時間は変わらないため，クライアント数をさらに増やした場合でも，起動性能はほとんど落ちないと予想できる．以上より，組込みサーバにおいてもスケーラビリティを発揮することを確認できた．負荷分散用キャッシュサーバを併せて複数利用することで，さらなるスケーラビリティが実現可能であるほか，耐故障性を実現できる．

8. 関 連 研 究

高速化・負荷分散機構，耐故障性を持つファイル共有システムとして，Lustre，BTSlave，CFS 等が存在する．また，既存のシンクライアントシステムとして，STRAGEX，VIVER 等が存在する．これらと本手法についての比較を行う．

- Lustre[12]
  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り，本研究で利用を想定している小型サーバには適さないため，モバイル化が困難である．
- BTSlave[11]
  BTSlave は BitTorrent[9] プロトコルを利用する分

散ファイルシステムである．BitTorrent プロトコルは分散ハッシュテーブルを利用し，スケーラビリティを実現しているが，ランダムなブロックの読み込みに適さず性能向上が難しい．特に多数のクライアントにシンクライアントシステムを提供するような場合，ブロックファイル転送のトラフィックがボトルネックとなるため，本手法で効率的かつ十分なスケーラビリティを実現できる．

- CFS[10)]
  CFS は分散ハッシュテーブル（DHT）を実現するアルゴリズムである Chord を利用した P2P ファイルシステムである．CFS は WAN ベースで利用でき，PureP2P 形態のためスケーラビリティの面で優れるが，実験的な実装であり性能や安定性に難がある．本システムでは検索，ブロックファイルの通信ともに，オーバヘッドが小さく高速なシンクライアントシステムを実現できる．
- 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．しかし，HTTP-FUSE-cloop はファイアウォールを容易に越えることができるという利点がある．また，iSCSI ストレージは専用ハードウェアが必要な場合が多くモバイルにも向かないが，HTTP-FUSE-KNOPPIX は特殊な専用ハードウェアが必要なく安価で，持ち運びも可能である．
- VIVER[15)]
  VIVER は LiveCD を用いた，専用サーバを必要としない Linux シンクライアントシステムである．ディストリビューション非依存，プラグイン方式による柔軟性といった特徴を持つ．手軽にシンクライアントシステムが構築できる点では本研究と同様だが，本研究はサーバの負荷分散が可能，耐故障性が確保できる，シンクライアントサービス提供に PC が必要ないといった利点がある．

9. 今後の課題

今回，P2P 機能は起動時のトラフィック分散を主眼において設計したが，様々なアプリケーションがランダムに起動されるような環境では，分散ハッシュテーブル（DHT）などを利用し，PureP2P の形態でアプリケーションのトラフィック分散まで行うことが望ましい．また，木構造生成のアルゴリズムについては非常に単純な実装になっており，さらに改善が可能であるほか，評価に関してもさらに台数を増やした場合の評価が必要である．また，セキュリティに関して，悪意のあるノードがある場合にはシステムの性能を落とすことができる．本システムは LAN 内での運用を基本としているため，性能低下のみで致命的な問題にならないことが多いと思われるが，対処が不十分である．これらを今後の課題とする．

10. おわりに

本論文では，耐故障性やスケーラビリティを有する P2P モバイルシンクライアントシステムを組込みシンクライアントサーバ HTTP-FUSE-KNOPPIX-BOX を開発することにより実現した．P2P 技術を導入した結果，資源に制限の大きい組込みシステムにおいても十分なサーバ性能を実現することができた．これにより，モバイル用途はもちろん，従来の高性能なシンクライアントサーバシステムにも本システムを適用することができ，低コスト化を実現できる．

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 (2005).
2) 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).
3) 金井, 須崎, 八木, 並木：モバイル環境向け HTTP-FUSE-KNOPPIX BOX のキャッシュによる高速化, DICOMO2006, pp.765-768 (2006).
4) 金井, 須崎, 八木, 並木：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, インターネットコンファレンス 2006 (2006).
5) Kitagawa, Tan, Abe, Chiba, Suzaki, Iijima, Yagi: File System (Ext2) Optimization for Compressed loopback device, Linux-Kongress 2006 (2006).
6) 小西, 石井, 浜野, 大木, 福田, 小長谷: KNOPPIX クラスターによるインスタント・ハイスループットコンピューティング環境の設計と構築, Computer System Symposium (2005).
7) Squid http://www.squid-cache.org/
8) Patterson, Gibson, Katz: A Case for Redundant Arrays of Inexpensive Disks (RAID), Proc. of the International Conference on Management of Data (1988.6)
9) Cohen: Incentives Build Robustness in BitTorrent, First Workshop on Economics of Pexr-to-Peer Systems (2003).
10) Dabek, Kaashoek, Karger, Morris, Stoica: Wide-area cooperative storage with CFS, Proceedings of SOSP'01, pp.202–215 (2001).
11) Cox: BTSlave: http://btslave.sf.net/
12) Lustre http://www.lustre.org/
13) DNS Balance http://openlab.jp/dns_balance/
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).
15) 古橋: 統合ディスクレスネットワーク基盤システム, 第 48 回プログラミング・シンポジウム, (2007).