システムソフトウェアと
オペレーティング・システム 78－20
（1998．5．8）

移動型計算機のためのファイルシステム KFSの設計と実装

岩本 健嗣　　望月 祐洋　　徳田 英幸
慶應義塾大学 政策・メディア研究科　慶應義塾大学 政策・メディア研究科　慶應義塾大学 環境情報学部

計算機の小型化に伴い移動型計算機環境への需要は急速に高まっている．しかし，現在のネットワーク環境と異なる移動型計算機環境では，既存のネットワークファイルシステムを利用すると性能や計算機の資源，移動透過性の面で問題が発生する．本稿ではこうした問題点を解決し，移動型計算機のためのファイルシステムの設計について述べ，ユーザレベルでの実装を用いて行なった測定について述べる．

KFS: Design and Implementation of Network File System for Mobile Computing Environment

Takeshi IWAMOTO[1], Masahiro MOCHIZUKI[1] and Hideyuki TOKUDA[2]

[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University

Desktop computers are being replaced by portable computers in network computer environment. But usage of these portable computers differs from that of desktop computers, several problems occur when it is used in today's network computing environment. To solve these problems, this paper describes the design and implementation of network file system for mobile computing environment,named KFS.

1 背景

近年のコンピュータの小型化，またPDAと呼ばれるさらに小型化されたデバイスの登場によって，人々が携帯型コンピュータを移動しながら利用することが現実となりつつある．さらに，こうした小型コンピュータをネットワークに接続して利用することも可能となっている．

しかし，移動型計算機環境は，既存のネットワーク環境と比べて次に述べるようにさまざまな点で異なっている．例えば，広域ネットワークの利用や計算機の資源が欠如していること，また，計算機の移動に伴ない利用するネットワークが変化すること，ネットワークが突然切断される可能性といった特徴がある．

こうした環境の相違を踏まえると，移動型計算機のための通信プロトコルや，ネットワークファイルシステムなどのミドルウェアの分野においてもいまだ研究過程であり，包括的な移動型計算機のためのネットワーク環境の構築が必要不可欠となる．特に既存のネットワークファイルシステムを移動型計算機で利用するとさまざまな問題が生じる．

本論文では，移動型計算機のためのネットワークファイルシステムKFSの設計を行いこうした問題を解決した．またユーザレベルの実装を用いてKFSの基本性能の測定を行った．

2 既存のネットワークファイルシステム

まず本論文で想定する移動型計算機環境の定義を行い，次にその環境下で既存のネットワークファイルシステムを利用した場合の問題点について触れる．最後に実際にいくつかのネットワークファイルシステムについて説明し，それぞれの問題点を述べる．

2.1 移動型計算機環境

移動型計算機環境は，以下の点で既存のネットワーク環境と大きく異なる．既存のネットワーク

ファイルシステムの評価を行う前に，特に本論文で想定する移動型計算機環境を定義する．

- 計算機がネットワークに接続するポイントが変化する．
  移動型計算機の物理的な移動に伴ない，接続するネットワークが変化する．
- 帯域が狭いネットワークや広域ネットワークを利用する．
  移動型計算機は，Ethernetのような比較的帯域の広い有線ネットワークだけではなく，帯域の狭い電話回線や無線ネットワークなども利用し通信を行う．また，移動先が物理的に大きく離れている場合，応答時間の長い広域ネットワークを利用することがある．
- 計算機資源の欠如．
  本論文では移動型計算機として小型のPDAなどを利用する場合を考慮する．これらは従来のコンピュータなどと比較してメモリやディスクなどの計算機資源が欠如している．
- 通信が突然切断されることがある．
  無線ネットワークや電話回線を利用しているため，通信が突然切断されることがある．

2.2 既存のネットワークファイルシステムの問題点

既存のネットワークファイルシステムを前節で述べたような移動型計算機環境で利用すると次のような問題が生じる．

(1) 広域分散環境での性能の低下

移動型計算機環境では，ファイルサーバとクライアントが広域ネットワークを利用して通信することが考えられる．例えば，物理的に遠く離れた場所に移動し，移動先から移動元のファイルサーバを参照する場合などが挙げられる．こうした場合，ファイルサーバとクライアント間の通信の応答速度の低下に伴ない，通常のネットワークファイルシステムでは応答性が極端に低下してしまう．

(2) 移動型計算機の資源の消費

今日のコンピュータの小型化によりラップトップコンピュータの性能は格段に向上した．しかしPDAと呼ばれるより小型化された計算機では通常のデスクトップのコンピュータと比較して計算機資源，特に2次記憶が欠如している．

大規模なキャッシュが必要なネットワークファイルシステムは，ディスクを大量に消費するためこうした小型計算機で利用することは困難である．

(3) 移動透過性

ファイルシステムを利用しているクライアントがが移動し，他のネットワークに接続した場合，IPアドレスの変更が行われる．クライアントの認証をIPアドレスで行っているファイルシステムでは，クライアントが移動した場合に同じファイルサーバを参照し続けることができない．

(4) 切断時のファイル操作

移動型計算機環境では，クライアントは無線ネットワークや携帯電話などを利用することが考えられるため，ネットワークが一時的に切断されることがある．有線ネットワークやデスクトップコンピュータでの利用を前提とした既存のネットワークファイルシステムでは，切断時にまったくファイル操作できなくなる．

2.3 既存のファイルシステムの評価

既存のネットワークファイルシステムの評価を表1にまとめる．表中の“○”はその点について考慮されていることを示し，“×”は考慮されていないことを示す．また，“△”を記したものは条件付きで考慮されている．

NFS[1][2]を除くネットワークファイルシステムでは広域ネットワーク利用時のパフォーマンスに関しては改善されているが，そのためにクライアントに大きなキャッシュを持たせることが前提となっており，計算機資源の面で移動型計算機環境に適していない．また，移動透過性についてはPFS[10]を除くファイルシステムで全く考慮されていないため，クライアントが移動した場合にサーバの再設定などが必要である．切断時のファイル操作はCoda[6]，Little Work[9]，PFSでクライアントにキャッシュとして保持されているファイルにのみ限って可能である．

3 設計

3.1 設計目標

KFSは移動型計算機環境で利用するために次に挙げる機能を満たすことを目標として設計を行う．

- 広域ネットワーク上での良好な応答性
- 計算機資源の保護
- 移動透過性の考慮
- ネットワークの切断時のファイル操作

3.2 基本設計

本節では，KFSのソフトウェア構造について述べ，3.1節で述べた目標のそれぞれの実現方法について説明する．

表 1: 既存のネットワークファイルシステムの問題点

名前	広域ネットワーク利用時の性能	計算機資源の保護	移動透過性	切断時のファイル操作
NFS	×	○	×	×
AFS	○	×	×	×
Coda	○	×	×	○
Little Work	○	×	×	○
PFS	○	×	△*1	○

*1 VIP を利用

3.2.1 KFS のソフトウェア構造

KFS の基本構造を図 1に示す．KFS はファイルサーバ，キャッシュサーバ，クライアントの3つのコンポーネントからなり，各コンポーネント間の通信は KFS プロトコルによって行われる．また，キャッシュサーバとファイルサーバ間の通信は，キャッシュを扱う CMP プロトコルを利用する．

クライアントはファイルサーバ上のファイルをネットワークを経由して共有する．共有されたファイルはクライアントのファイルシステム上にマウントされローカルディスク上のファイルと同様にアプリケーションやユーザから透過的に扱うことができる．

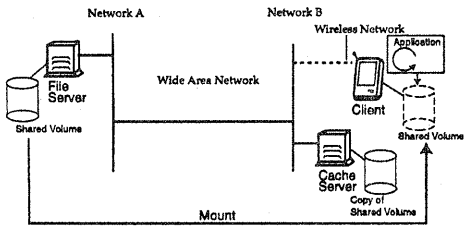

図 1: KFS の基本構造

ファイルサーバは NFS などのネットワークファイルシステムと同様の機能を提供し，通常のファイルに対する操作 (read や write，lookup など) の要求に応答する．

キャッシュサーバは KFS 独自のコンポーネントで，ファイルサーバ・クライアント間で要求されたファイルの内容をキャッシュしている．1 ネットワークセグメントにつき，1 つのキャッシュサーバが存在し，そのセグメントに接続されている複数のクライアントが同じキャッシュサーバを利用する．

クライアントはキャッシュサーバに対してファイル操作の要求を行い，キャッシュサーバはそのファイルのキャッシュを保持していれば，キャッシュの内容で応答する．もし存在しなかった場合はキャッシュサーバがファイルサーバに要求を行い，ファイルを取得する．その後のクライアントからの要求にはそのキャッシュを利用して応答する．またキャッシュサーバはキャッシュを用いる際に，ファイルサーバに対してキャッシュの有効性の確認を行わない．そのため，広域ネットワークを通した通信を極力減らすことができる，

クライアントが移動した際は，移動先のキャッシュサーバと通信を行い，移動先のキャッシュサーバがファイルサーバに対して要求を仲介する．図 2では，ネットワーク B からネットワーク C にクライアントが移動を行い，移動先のネットワーク上のキャッシュサーバと新たに通信を行ない，ファイルサーバ上のファイルを共有している．

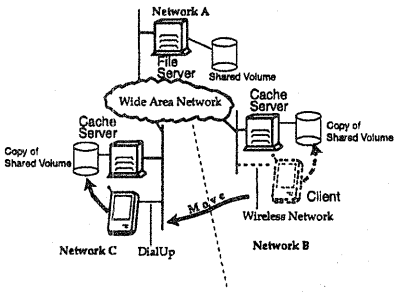

図 2: KFS でのクライアントの移動

3.2.2 広域ネットワーク上での良好な応答性

KFS ではキャッシュサーバがキャッシュを持つため，実際のファイルサーバとの通信を極力減らすことができる．これにより，遅延時間の大きい広域ネットワークを介する通信を避けられる．

また，3.3節で説明するように，キャッシュサーバとファイルサーバ間でのキャッシュの有効性の確認はクライアントの動作と非同期に行われるため

に，クライアントがその応答時間を待つ必要がない．そのため，遅延時間の大きい広域ネットワークで利用した時でも高速な応答性を提供する．

3.2.3 計算機資源の保護

KFS では，基本的にクライアントにキャッシュを行わない．これにより移動型計算機のメモリやディスクを圧迫することなくファイルの共有を行うことができる．

3.2.4 移動透過性の考慮

移動型計算機環境では，計算機がネットワークを越えて移動を行う．通常のネットワークファイルシステムでは，移動してクライアントの IP アドレスの変更が行われると通信できなくなる場合が多い．

KFS では，ネットワークプロトコルとして Mobile-IP[12] を利用することによって，移動透過性の保証をネットワーク層に期待する．Mobile-IP により計算機が移動しても連続した通信を確保することができる．また，移動先でも同じ IP アドレスを利用することができるため，ファイルサーバでの認証をクライアントの IP アドレスを用いて行っても，サーバ側での再設定や再起動の必要がない．

3.2.5 ネットワークの切断時のファイル操作

2.3節で述べたように，NFS や AFS などのネットワークファイルシステムでは通信が切断されるとファイル操作が全く行えなくなる．KFS では移動型計算機のネットワークが切断されても，ユーザが指定したファイルに限ってクライアントにキャッシュを持たせることで，操作を可能にする．

3.3 キャッシュの一貫性

一般的にファイルシステムにキャッシュを持たせる場合，キャッシュの一貫性の問題が生じる．つまり，オリジナルのファイルとその複製であるキャッシュとの間にずれが起こらないようにしなければならない．例えば図 3では，あるファイルをクライアントが読み込んだ場合，そのクライアントが接続しているキャッシュサーバ上にキャッシュが読み込まれるが，このファイルに別のクライアントが変更を加えた場合キャッシュサーバ上のキャッシュとファイルサーバ上のファイルの間でキャッシュの一貫性が損なわれる．KFS では次に説明する機構によって一貫性を保証する．

クライアントの要求を受けたキャッシュサーバは，キャッシュの一貫性が保たれているかどうかの確認をファイルサーバに行わなければならないが，KFS ではクライアントへ応答したあとでこの確認を行う．この機構により，クライアントはキャッシュサーバが一貫性の確認を終了するのを待たずに，要求に対する応答を受けとれるので応答性が向上する．しかし逆に，クライアントが無効なキャッシュを利用してしまう可能性もある．こうした場合，KFS ではキャッシュサーバがファイルサーバに対して一貫性の確認を行った後，そのキャッシュが無効であった場合，クライアントが利用しているキャッシュを同じファイルの別のバージョンとして切り替える機構を用意する．ユーザは後にそれらのファイルを自由に統合できる．

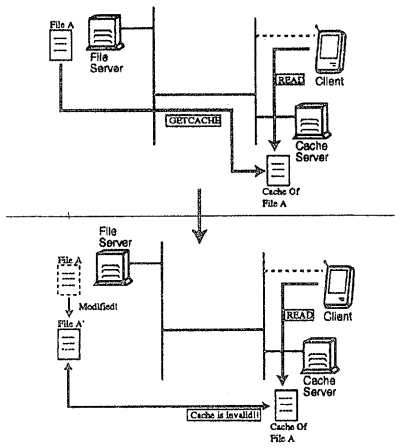

図 3: キャッシュの一貫性の破れ

この設計ではあるクライアントが参照しているファイルが，他のクライアントによって頻繁に書き換えられると，次々と異なるバージョンのファイルが作成されてしまう．また，いつでも最新のファイルを見たいという要求を満たすことはできなくなってしまう．しかし，一般的に一つのファイルを複数のユーザが同時に参照し変更を加える可能性は低いため [13]，こうした場合については特に考慮せず一貫性の処理を行なう．

例えば，図4では，クライアントがキャッシュサーバに対してファイルの read 要求を行っているが，ファイルサーバ上ですでに別のクライアントによってそのオリジナルのファイルが書き換えらえている．この場合キャッシュサーバ中にあるキャッシュはすでに無効であるが，キャッシュの有効性の確認はクライアントに対する応答の後に行われるため，クライアントは無効なキャッシュの内容を read することになる．クライアントに応答した後，キャッシュが無効であったことを確認したキャッシュサーバはクライアントに対してその旨を通知し，現在ク

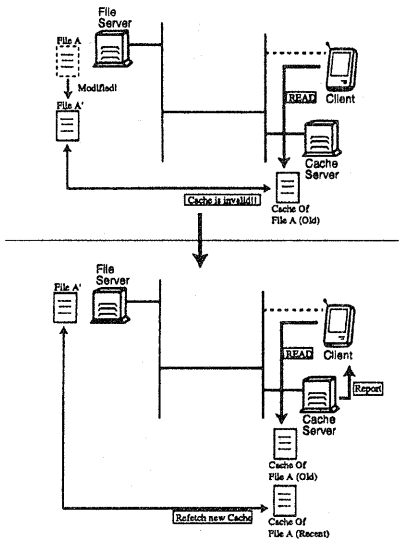

図 4: キャッシュの一貫性の保証

ライアントが利用しているキャッシュを同じファイルの別のバージョンとして切り替え，最新のキャッシュをファイルサーバから取り込む．

また同様に書き込みの場合も，要求を受けたキャッシュサーバはキャッシュに書き込みを行い，クライアントに即座に応答してからファイルサーバにキャッシュの有効性の確認を行う．

3.4 KFS プロトコル

KFS で用いられるプロトコルはクライアントとキャッシャサーバ間で用いられる "KFS-MH プロトコル" とキャッシュサーバとファイルサーバ間でやりとりされる "CMP (Cache Management Protocol) プロトコル" に分けられる．また，これらのプロトコルは Sun RPC 上のプロトコルとして実装される．

3.4.1 KFS ファイルハンドル

KFS プロトコルでは，クライアントやキャッシュサーバがサーバ上のファイルを指定するのに KFS ファイルハンドルと呼ばれる識別子を利用する．クライアントは，lookup 要求をサーバに送ることによりそのファイルの識別子を得ることができる．それ以降のファイルの読み込み，書き込みその他の操作はこの KFS ファイルハンドルを用いて行う．

この識別子は，サーバ上でのみ意味を持つのでクライアントで偽造，改変を行うことを防いでいる．またクライアントがファイルサーバ側と異なる名前空間のファイルシステムを利用している場合でもこの識別子を用いることで透過的にファイル操作を行うことができる．

4 実装

本節では，KFS のユーザプロセスによる実装について述べる．KFS はクライアント，キャッシュサーバ，ファイルサーバの 3 つのコンポーネントからなるが，クライアントは，簡単なファイルシステムを提供する部分と測定評価用プログラムからなり，キャッシュサーバ，ファイルサーバはそれぞれ一つののサーバとして実装した．

4.1 RPC を用いた KFS プロトコルの実装

KFS プロトコルは RPC を用いて各手続きの実装を行なった．またプロトコルコンパイラとして OS 付属の rpcgen を用いた．

4.1.1 KFS-MH プロトコル

以下に KFS-MH プロトコルで実装した 4 つの手続きを示す．

- KFS_MHREAD
- KFS_MHREADDIR
- KFS_MHLOOKUP
- KFS_MHOPEN

(1) KFS_MHREAD

この手続きはクライアントがファイルを読み込むときに利用する．引数として読み込みたいファイルのファイルハンドル，位置，サイズを渡し，読みこんだバッファが戻り値として返される．

(2) KFS_MHREADDIR

この手続きはディレクトリの内容，つまりディレクトリのエントリを読み込むために利用する．引数として読み込むディレクトリのファイルハンドルを渡し，戻り値としてディレクトリエントリの数とディレクトリエントリの kfsnode 構造体の配列が返される．

(3) KFS_MHLOOKUP

KFS_MHLOOKUP はクライアントがファイルまたはディレクトリのファイルハンドルを得るために利用する．引数には親ディレクトリのファイルハンドルとファイルハンドルを得たいファイル，またはディレクトリの名前を渡し，戻り値としてファイルハンドルが返される．

(4) KFS_MHOPEN

この手続きはクライアントがファイルを open する時に利用される．キャッシュサーバはこの要求を受けた時点でキャッシュを調べ該当するファイルのキャッシュが無ければファイルサーバに要求を行う．引数はファイルハンドルを渡し，成功か不成

功を示すフラグが戻り値として返される．ファイルサーバにキャッシュの要求を行う場合でもその終了を待たずに関数は値を返す．

4.1.2 CMP プロトコル

キャッシュサーバ・ファイルサーバ間で利用される CMP プロトコルでは，以下に示す 2 つの手続きの実装を行なった．

- KFS_GETCACHE
- KFS_GETDIRENT

(1) KFS_GETCACHE

キャッシュサーバがファイルサーバに対してキャッシュの要求を行う手続きで，引数としてファイルのファイルハンドル，読み込み位置，サイズを渡す．実装ではキャッシュはファイル単位で行なわれるが，プロトコルでは一定のサイズに区切って転送が行なわれる．

(2) KFS_GETDIRENT

この手続きはキャッシュサーバがファイルサーバ上のディレクトリのエントリを得るために利用する．引数はディレクトリのファイルハンドルで，戻り値はエントリの数とエントリの kfsnode 構造体の配列が返される．

4.2 クライアント

クライアントは KFS-sh(KFS-shell) というユーザに簡単なファイルの操作を提供するシェルの部分と，その操作を KFS プロトコルに変換しキャッシュサーバとの通信を行う KFS-cif(KFS-client interface) に別れる．クライアントのソフトウェア構造を図 5に示す．

4.2.1 KFS-sh

KFS-sh では表 2に示すように，cat や ls，cd などの Unix でのファイル操作コマンドの一部が利用できる．

表 2: KFS-sh で利用できるコマンド一覧

コマンド名	機能
cd	ワーキングディレクトリの移動
pwd	ワーキングディレクトリの表示
ls	ディレクトリエントリの表示
cat	ファイル内容の表示

KFS-sh はユーザからのコマンドと引数を解析し，適当な KFS-cif のルーチンを呼び出す．その後，KFS-cif の処理が終了を待ち返ってきた結果を整形しユーザに表示する．

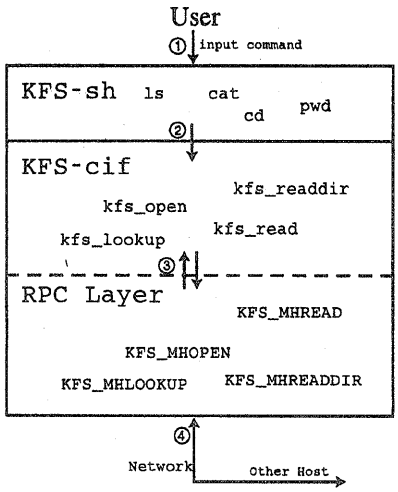

図 5: KFS クライアントのソフトウェア構造

4.3 キャッシュサーバ

キャッシュサーバはユーザレベルのサーバプロセスとして実装されており，実行されると，クライアントからの KFS-MH プロトコルによる要求を待ち続ける．main 関数は rpcgen の作成する KFS-MH プロトコル側のサーバ用ソースコードを利用しており，今回の実装では各手続きを処理する関数の作成を行なった．

4.4 ファイルサーバ

ファイルサーバもキャッシュサーバと同様にユーザプロセスのサーバとして実装されており，キャッシュサーバからの CMP プロトコルによる要求に対して応答する部分と，SFS(Simple File System) と呼ばれる実際のファイルシステム (UFS) にアクセスする部分からなる．main 関数は rpcgen の生成する CMP プロトコルのサーバ側ソースコードを利用している．

キャッシュサーバからの要求によって，ファイルの内容やディレクトリのエントリを返す．

5 評価

ユーザプロセスによる KFS の実装を用いた通信性能を測定し，NFS との比較を行った．また，計算機資源の保護，移動透過性，切断時のファイル操作の点についてその他のネットワークファイルシステムと比較し定性的評価を行った．

5.1 測定環境

測定は図 6に示した環境で行なった．

キャッシュサーバとファイルサーバ間はSLIPを用いて接続を行う．転送速度は19200bpsであり，遅延の大きい広域ネットワークを擬似的に再現している．クライアントとキャッシュサーバ間は10MbpsのEthernetを利用して接続を行った．

また，比較のために3台とも同一のEthernet上に接続した環境でも測定を行なった．

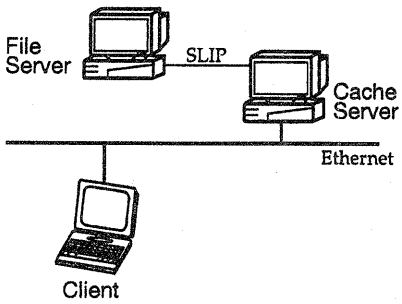

図 6: SLIPを用いたKFSの測定環境

5.2 読み込み速度

クライアントでファイルサーバ上の3種類の大きさのファイルの読み込みを行い，KFS，NFSでそれぞれにかかる時間を測定した．また比較のために3台を同一のEthernetで接続した環境でも測定を行なった．

表 3に，SLIP を用いた実験環境で 1KByte，10KByte，100KByteの3種類のファイルの読み込みを行った結果を示す．2回目の読み込みは，1回目の直後に行ない，3回目の読みこみは2回目の60秒後に行った．

表 3: KFS と NFS の読み込み速度 (19200bps SLIP)

ファイルサイズ	回数	KFS	NFS
1KB	1回目	0.34	1.05
1KB	2回目	0.05	0.16
1KB	3回目	0.05	0.31
10KB	1回目	2.01	15.14
10KB	2回目	0.09	0.16
10KB	3回目	0.1	0.31
100KB	1回目	19.15	138.01
100KB	2回目	0.89	0.20
100KB	3回目	0.91	0.35

単位は秒

表 4: KFS と NFS の読み込み速度 (10M Ethernet)

ファイルサイズ	回数	KFS	NFS
1KB	1回目	0.10	0.01
1KB	2回目	0.01	0.01
1KB	3回目	0.01	0.01
10KB	1回目	0.18	0.03
10KB	2回目	0.07	0.01
10KB	3回目	0.07	0.01
100KB	1回目	0.68	0.18
100KB	2回目	0.64	0.05
100KB	3回目	0.64	0.05

単位は秒

1回目のファイルの読み込みでは，KFS，NFSともSLIP上のファイルサーバから読み込みを行うため時間がかかっている．2回目以降の読み込みでは，KFS，NFSともキャッシュを利用するため1回目の読み込みより高速になっているが，NFSはメモリ上のキャッシュを利用する前に一貫性の確認応答を行うため，KFSより読み込みに時間がかかることがわかった．ただし，100Kのファイルを読み込んでいる場合は，ファイルのキャッシュをネットワーク上のキャッシュサーバからクライアントに転送する時間がかかるため，メモリ上のキャッシュを利用するNFSに比べて速度が劣っている．

また，3回目の読み込みは，2回目の読み込みから60秒の時間を置いているため，NFSではディレクトリの最終更新時間のキャッシュが無効になっており，ファイルのキャッシュを利用する前の確認応答の手順が増えている．このため，NFSでは2回目に比べて性能がさらに悪化している．これに対しKFSでは2回目の読み込み時間と3回目の読み込み時間に殆ど性能の低下は認められない．

5.3 Ethernet上での測定との比較

表4には10M Ethernet上でのKFSとNFSの測定の結果を示す．表3と比較すると，NFSは回線速度の低下に伴ない大幅に性能が低下しているのに対して，KFSではそれほどの低下はみられない．特に，2回目以降の読み込みでは，KFSは殆ど性能が低下していないことが分る．

5.4 KFSの定性的評価

表5に定性的評価の結果をまとめる．表中の“○”はその機能が提供されていることを示し，“△”は制限つきで利用できることを表す．また，は全くその機能について考慮されていない場合は“×”を

表 5: KFS の定性的評価

名前	計算機資源の保護	移動透過性	切断時のファイル操作
KFS	○	△*1	△*2
NFS	○	×	×
AFS	×	×	×
Coda	×	×	○
Little Work	×	×	○
PFS	×	△*3	○

*1 Mobile-IP を利用

*2 ユーザが指定したファイルのみ可能

*3 VIP を利用

記した．この結果から，KFS は移動型計算機環境でのネットワークファイルシステムの問題点を解決しており，特にクライアントとなる計算機の資源が欠如している場合に向いていることが分った．

6 まとめ

移動型計算機を利用する環境は，次のような特徴を持つ．

- 計算機の移動に伴ない接続するネットワークが変化する．
- 広域ネットワークや帯域の狭いネットワークを利用することがある．
- 計算機資源が欠如している．
- 通信が突然切断されることがある．

本論文ではこの移動型計算機環境でネットワークファイルシステムを利用した場合の問題点を解決する KFS の設計と実装を行った．ユーザプロセスによる実装を用いて行なった NFS との比較では，最大で約 6 倍の読み込み速度を得られた．この点から広域ネットワークでも，良好な応答性が期待できる．また，定性的な評価では KFS は以下のような機能を持ち，移動型計算機環境での利用に適したネットワークファイルシステムであることが明らかになった．

- 計算機資源の保護
- 移動透過性の考慮
- ネットワークの切断時のファイル操作

参考文献

[1] SUN Microsystems,
"NFS White Paper"
http://www.sun.com/sunsoft/solaris/desktop/nfs.html

[2] Sun Microsystems,
"NFS: Network FIle System Protocol Specification," 1989, RFC1094

[3] Sun Microsystems,
"RPC: Remote Procudure Call Protocol Specification Version 2," 1988,RFC1057

[4] Sun Microsystems,
"XDR: External Data Representation Standard, " 1987, RFC1014

[5] Morris,j.,
Satyanarayanan,M., Conner,M.H.,Howard,J.H., Rosenthal, D.S., Smith,F.D.(1986).
"Andrew:a distributed personal computing environment" Comms.ACM,

[6] James J.Kistler And M.Satyanarayanan
"Disconected Operation in the Coda File System," In Proceedings of the Thirteenth Syposium on Operations Systems Principles, Oct., 1991

[7] M.Satyanarayanan, James J.Kistler, Lily B. Mummert Maria R.Ebling, Puneet Kumar and Qi Lu,
"Experience with Disconnected Operation in a Mobile Computing Environment" In Proc. of USENIX Mobile and Location-Independent Computing Symposium, pp.11-28, Aug., 1993

[8] M.Satyanarayanan,
"Fundamental Challenges in Mobile Computing" In 15th ACM Symposium on Principles of Distributed Computing, May., 1996.

[9] Peter Honeyman, Larry Huston, Jim Rees, and Dave Bachmann,
"The LITTLE WORK Project,"
http://www.citi.umich.edu/u/rees/wwos3.ps

[10] WIDE プロジェクト DFS ワーキンググループ,
"広域分散ファイルシステム" WIDE プロジェクト研究報告書 1994

[11] Fumio Teraoka, Keisuke Uehara, Hideki Sunahara and Jun Murai,
"VIP: A Protocol Providing Host Mobility," Communications of the ACM, Vol.37, No.8, pp.67-75, Aug., 1994

[12] Charles Perkins, editor.
IP mobility support. Internet Request For Comments RFC 2002, October 1996.

[13] Satyanarayanan, M.
"A Study of File Sizes and Functional Lifetimes," Proc. of the 8th Symposium on Operating Systems Principles, ACM, pp.96-108,1981