1Z－6

MKngプロジェクトにおけるモーバイル計算機環境: Codaファイルシステムにおけるクライアント間キャッシュミス補償機能の設計†

稲村 浩　　南部 明　　徳田 英幸

NTT 情報通信研究所　　慶應義塾大学 環境情報学部

1 はじめに

慶応大学を中心に行なわれている次世代マイクロカーネル(MKng)研究プロジェクトでは，Real-Time Machマイクロカーネルと Lites UNIXサーバ[3]上で，モーバイルコンピューティング環境のためのファイルシステムであるCodaファイルシステム[2]を利用可能にした．

Codaファイルシステムは米カーネギーメロン大学で研究開発されている分散ファイルシステムである．クライアントがネットワークから切り離された状態でもサービスの継続が可能な，ディスコネクトオペレーションという動作モードを持っているのが特徴である．この動作モードはCodaクライアントがローカルディスクに個々のファイル全体をキャッシュするという性質を利用している．楽観的複製制御に基づき，更新の衝突は再統合フェイズで利用者の助けを借りて解決する．クライアントとサーバは役割がはっきりしているため，それぞれ専用ノードで運用されることが多い．

2 問題

複数のCodaクライアントがサーバから隔離された状況(図1参照)を考える．現状のCodaクライアントはサーバに接続できなくても動作できる[1]が，サーバに接続せずにクライアント同士ではファイルのやり取りはできない．例えば複数のCodaの利用者が外国に出かけたとする．持っているラップトップ同士は通信できるものの，本国のサーバには通信コストの点で接続するのが難しいとしよう．あるクライアントがアプリケーションの実行に必要なライブラリをキャッシュしていなかったが，他のクライアントが保持しているということが起こり得る．Codaクライアント同士が，接続の難しいサーバに依存することなくファイルのやり取りができれば，この状況に対応できる．

モーバイル環境でデータの転送に利用可能なメディアには様々なものがある．フロッピーディスクなどのリムーバブルメディアやメイルに限定されたリンク，ポイントトゥポイントリンクからLANとしてマルチプルアクセスできるものまである．これらの間で制約の度合を変えながらファイルアクセス機能を提供することが望ましい．

この問題を解決するためにはCoda自体と連係する必要がある．

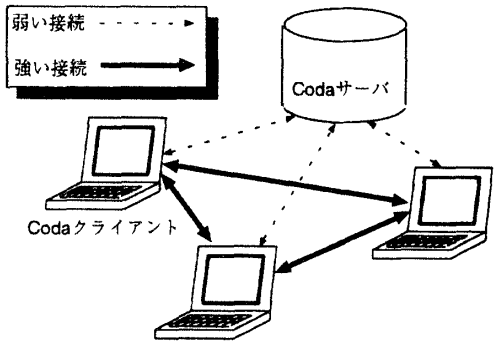

図1: 複数のクライアントがサーバに接続できない状況

例えばフロッピーでファイルをやり取りしてCodaの空間に書き込んでしまう場合を考えよう．共用のフォントファイルのように読み出しのみに設定されているファイルをキャッシュミスした場合，普通のコマンドでは同じ名前のファイルとして書きこむことはできない．書き込める場合でもタイムスタンプ等は変わってしまうので再統合の対象になり無駄である．その場だけNFSなどの他のファイルシステムを使ったとしても，Codaと連係しない限りクライアントキャッシュなどのメリットは得られない．

3 クライアント間キャッシュミス補償機能の設計

モーバイル環境で利用可能なメディアの多様性に対応したクライアント間キャッシュミス補償のために，Import/Exportコマンドとセッションサーバの2つのメカニズムをCodaの拡張として提案する．まず利用可能なメディアの特徴から導かれる2つの要求を以下にまとめる．

1. フロッピーディスクやメモリカードのようなリムーバブルメディアを使ってデータを移動させる場合，インタラクションの多いプロトコルを使うのは難しい．メディアの入れ換え回数が多くなり，時間と手間がかかるからである．
2. 無線LANなどのマルチプルアクセスが可能なメディアが利用できる場合には，要求されたファイルが一つのノードに見つからなくても他のノードにも問い合わせることができる．

以上を踏まえて，1の場合を扱うためのバッチ式のメカニズムとしてImport/Exportコマンドを，2の場合を扱うために，キャッシュディレクトリを備えた，そのセッションのみ有効なサーバであるセッションサーバを用いるメカニズムを考える．

3.1 Import/Exportコマンド

あるクライアントのキャッシュの内容のサブセットを書き出し(export)，他のクライアントで読み込む(import)コマンドを提供する．これらの操作を行うためにCodaのクライアント側のキャッシュマネージャであるvenusを拡張する．図2に示すようにVenusではキャッシュされたファイルのデータおよび状態はFSDB, VDB,

---

Mobile Computing Environment in the MKng Project:
A Client Side Cache-miss Compensation Mechanism in the Coda File System

Hiroshi INAMURA[1], Akira NAMBU[1] and Hideyuki TOKUDA[2]
[1]NTT Information Systems Laboratories
1-1, Hikarinooka Yokosuka, Kanagawa 239, Japan
E-Mail: <{inamura,nambu}@isl.ntt.co.jp>
[2]Faculty of Environmental Information, Keio University

†この研究は，情報処理振興事業協会（IPA）が実施している創造的ソフトウェア育成事業「次世代マイクロカーネル研究プロジェクト」のもとに行われた．

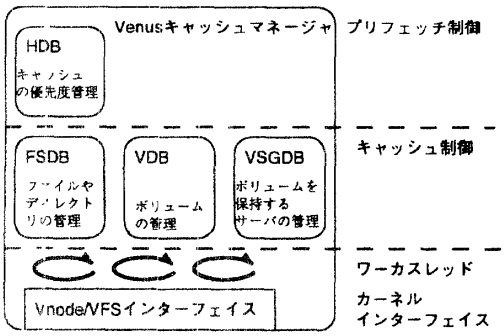

図 2: Venus の内部構造と主要なデータ

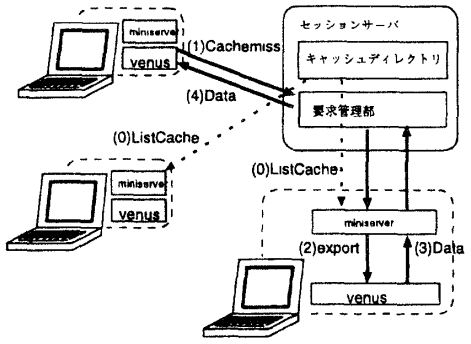

図 3: セッションサーバの構成とキャッシュ補償の流れ

VSGDB の 3 つのデータベースに格納されているデータで記述される.

FSDB ファイルやディレクトリのサイズやタイムスタンプなどのアトリビュート情報や, ローカルディスクのキャッシュファイルと Coda ファイルの関係などを記述するデータが管理されている.

VDB Coda ではファイルの集合はボリュームを単位に管理されている. これは通常の UNIX のパーティションに相当する. ボリューム自体もファイルと同じくアトリビュート情報を持っている. これらを記述するデータが管理されている.

VSGDB あるボリュームを保持しているサーバの情報が記述されている. そのボリュームに関する要求があると, VSGDB から得られたホストに接続を試みる. Coda ではサーバ複製をサポートしているので, そのための情報も含まれる.

バッチ式のメカニズムを指向し, 一度のインタラクションで完結するためには export コマンドは状態の回復に必要なデータを重複を厭わずに全て書き出す. すなわち, 上で説明した FSDB, VDB, VSGDB で関係する全てのデータの書き出しと回復を実現する.

Import/Export コマンドは, コマンドラインから起動する単純なインターフェイスにする. 特定のメディアに合わせて使うためにはシェルスクリプトなどで加工して使う. こうすることで例えば MIME エンコードするなどしてメイルでもデータのやりとりを行うこともできるようにする

3.2 セッションサーバ

複数のノードの持つキャッシュを有効に利用するために, キャッシュのディレクトリを管理するセッションサーバを置くことにする. Import/Export メカニズムでのキャッシュミス補償はほとんど手動である. しかし venus に既にあるキャッシュミス検知機構を応用することで自動化が計れる. このサーバ自体は永続的なデータを持たず, 必要になった時点でいずれかのノードで立ち上げる. さらに, セッションに参加する各ノードでは, セッションサーバとのインターフェイスのために miniserver を動かしておく. このセッションサーバを用いたキャッシュミス補償の流れを図 3 に示す. サーバを用いたキャッシュミス補償は, venus に拡張された Import/Export インターフェイスを基本にする. セッションサーバを用いたキャッシュ補償の流れを以下に説明する.

1. サーバにクライアントが接続した時点で可能ならサーバは List-Cache 要求を各 miniserver に出し, キャッシュディレクトリを作成する.
2. 各ノードの venus がキャッシュミスを検出すると, venus はその処理スレッドを待機させる. そのイベントからセッションサーバはキャッシュディレクトリを検索し, キャッシュしている可能性の高いクライアントを選択する. もし適切なノードが選択できず, キャッシュディレクトリが不完全な場合には List-Cache 要求によってキャッシュディレクトリを再構成する.
3. サーバは選択したクライアントに対して miniserver 経由で export 要求を出す.
4. データをキャッシュしている venus から要求元の venus にデータを転送する.

4 現状

3 節で述べたクライアント間キャッシュミス補償機能について実現を行っている. 現在 Import/Export 機構のコーディングとデバッグを行っている. Coda の利用者レベル管理コマンドである "cfs" からアクセスでき, 例えば

```
cfs export /coda/usr/inamura/doc /tmp/exported
```

とすることで venus のキャッシュファイルのデータと状態のダンプが/tmp/exported ファイルの形で得られる.

5 今後の課題

今回の検討ではキャッシュミス補償の問題を扱い, セッションサーバを導入する設計を行った. 次のステップとしてキャッシュミスの補償だけでなく, セッション中に行われた変更点を全ての参加クライアントに伝播させることを考えたい. キャッシュミス補償はキャッシュミスが起こって初めて起動されるメカニズムである. 更新の確実な伝播には, いずれかのノードで更新が起こったことによって起動するメカニズムが必要である. オリジナルの Coda ではサーバからのコールバックによって実現されているが, 同様のメカニズムが必要になるだろう.

6 おわりに

本稿では, モーバイルコンピューティング環境向けの分散ファイルシステムである Coda ファイルシステムに加える, クライアント間キャッシュミス補償機能の設計について述べた.

参考文献

[1] MUMMERT, L., EBLING, M., AND SATYANARAYANAN, M.: Exploiting Weak Connectivity for Mobile File Access, in 15th ACM Symposium on Operating Systems Principles(1995).

[2] 稲村浩: Coda プロジェクトの概要, 情報処理学会研究会報告 OS 情報処理学会 (11 1995).

[3] 徳田他: MKng: 次世代マイクロカーネル研究プロジェクトの概要, 第 55 回情処全大論文集 (1997), 1Z-2.