モーバイルコンピューティング 12－6
（2000．2．3）

モバイルコンピューティング環境に適した共通メモリ管理方式について

横山繁盛† 奥田隆弘‡ 水野忠則‡ 渡辺 尚‡

モバイル端末とサーバとで構成されるモバイルコンピューティングシステムは，無線通信の帯域幅の問題や接続の継続性，モバイル端末のバッテリーの持続時間等の課題が多い．さらにモバイル端末とサーバとが連携したアプリケーションは通信処理があるため，その構築が複雑となる．モバイル端末の一部のメモリ領域とサーバのメモリの一部の領域が共通メモリとなるように構成し，この共通メモリの内容の一貫性管理を行うことにより，モバイル端末やサーバから自由に共通メモリにアクセス可能な，モバイルコンピューティング環境向きメモリ管理方式(MMM)を提案する．これによりモバイル端末やサーバでのアプリケーションは通信を意識する必要がなくなるためプログラムの構築が容易となり，また通信の効率を上げることができる．アプリケーションのモデルによるシミュレーションを行い，一括伝送方式に比べ通信時間が減少し，実行時間が短縮されることを示す．

A Memory Management Architecture for Mobile Computing Environment

Shigemori Yokoyama,† Takahiro Okuda,‡ Tadanori Mizuno‡ and Takashi Watanabe‡

Mobile Computing Systems that consist of mobile terminals and servers have several issues these are narrow bandwidth of wireless communications, the limited battery duration of mobile terminals, and others. It is hard to develop application programs that operate in mobile computing environments because of the complicated procedure of wireless communications. We propose a common memory management system for mobile terminals and servers called "A Memory Management Architecture for Mobile Computing Environment (MMM)". MMM controls a part of memory areas of a mobile terminal and a part of memory areas of a server to be common, and maintains the consistency of the common memory areas. MMM differs from normal distributed shared memory systems in that its memory address space is not necessarily unique and communications between CPUs are normally wireless. MMM makes application programs on the mobile terminal and on the server to be developed more easily and increases the efficiency of communications. MMM was evaluated by the simulation using sample application program models. The results show that the communication time decreases and the execution speed increases compared to no common memory systems.

1. はじめに

無線通信技術の発展による携帯電話やPHS等の急速な普及，携帯情報端末の高性能化，小型化に伴い，モバイルコンピューティング環境が実現されつつある．しかし現状は，無線通信の伝送帯域幅が狭く、接続の継続性に難点があり，またモバイル端末においてはバッテリーの持続時間等の資源に制約があり課題が多い．さらにモバイル端末とサーバとで，データを共有する場合には内容の同期が課題である．モバイル端末とサーバとが連携したアプリケーションでは，モバイル端末とサーバとの間で通信が必要であり，その構築が複雑となる。

モバイルコンピューティング環境下でのこれらの課題解決のため各種の方式が提案されている．小型軽量化や消費電力の観点より，モバイル端末の機能を通信と表示の機能に特化し，他をサーバ側に持たせるアーキテクチャとした研究[1]，通信が切断される可能性がある環境下でのアプリケーションの構築法[2]やファイル方式[3,4]等のソフトウェア方式による研究がある．

本稿ではよりハードウェアに近い方式である，モバイル端末とサーバのメモリ領域の一部を共通メモリとして制御することを特徴とする，モバイルコンピューティング環境向きメモリ管理方式，MMM(A Memory Management Architecture for Mobile Computing Environment)を提案し，その評価を行う．

まず第2章でMMMのアーキテクチャについて述べ，特に2.6では一般の分散共有メモリとの差異について述べる．続いて第3章でMMMのモバイルコンピューティングへの適用について説明し，第4章ではシミュレーションによる評価結果について述べ，最後の第5章でまとめを行う．

2．MMMのアーキテクチャ

2.1 概要

MMMはモバイル端末とサーバのメモリ空間の一部を共有する方式であり，一種の分散共有メモリと考えることができる．

一般の分散共有メモリでは，ネットワーク上に分散したコンピュータのメモリを共有する方式であり，通信の

†三菱電機(株)情報システム製作所
Information Systems Engineering Center, Mitsubishi Electric Corp.
‡静岡大学情報学部
Faculty of Information, Shizuoka Univ.

帯域幅は比較的広く，任意のコンピュータ間で必ず通信ができることを前提としている．単一のCPUでは得られない大容量高速のデータ処理性能を得るために，多数のCPUを連携して動作させることを主目的とし，メモリを単一メモリ空間をとすることにより，プログラミングの容易化を図った方式と考えることができる[5,6]．

これに対して，MMMでは携帯電話等の無線通信を基本とし，帯域幅が狭く切断の可能性がある通信路を前提として，モバイル端末とサーバ間とでのメモリの共有をおこなう方式である．モバイル端末のアプリケーションが通信を意識することなく，共通メモリをアクセスすることにより，サーバ側のCPUやメモリ，入出力装置等のリソースが容易に利用可能とすることを目的とし，低速で切断の可能性のある通信路，通信非接続での利用が可能なモバイルコンピューティング環境向きの分散共有メモリ方式である．さらに標準的なアーキテクチャを持つコンピュータに対し，比較的簡単なハードウェアの追加により，性能にほとんど影響を与えることなく実現可能である．

MMMは，モバイル端末のメモリ空間の一部とサーバのメモリ空間の一部とを共通メモリ空間とし，ラインと呼ぶ一定の単位で内容の一貫性の管理をおこなう．図1にその概念図を示す．これによりモバイル端末から，サーバのメモリ空間の一部が自メモリ空間としてアクセス可能となり，またサーバからも、端末のメモリ空間の一部が自メモリ空間としてアクセス可能となる．ラインのサイズは通信路の幅，代表的アプリケーションに対応し，最適に選択可能である．

ソフトウェアによる分散共有メモリ方式では，共有メモリの制御のため，仮想メモリ方式の機構を利用している例があるが，MMM方式では，共有メモリの制御には専用の付加ハードウェアとソフトウェアにより実現し，それと共存させ，仮想メモリ機構はサーバ側の共通メモリ領域を二次記憶とすることでメモリ空間の拡大を実現している．

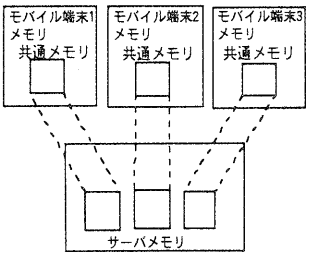

図1 MMMのメモリ管理方式
Fig.1 Concept of MMM

2.2 メモリの一貫性制御

一般の分散共有メモリ方式ではメモリ内容の一貫性制御が最大の課題である．共有メモリ方式のマルチプロセッサでは，ほぼシーケンシャル計算機としてのメモリ内容の一貫性制御が可能であが，分散共有メモリ方式では同様の一貫性制御を行うと，オーバーヘッドのため性能に大きな影響が出る．このため分散共有メモリ方式では，プログラミングに制約を与え，メモリの一貫性を緩和することで性能の向上を図っており，各種一貫性制御方式が提案されている[7-10]．MMMでは，モバイル端末とサーバ間の対向するプロセッサ間でのメモリ共有方式のため，一貫性制御を比較的単純化することができる．MMMでは次節以降で述べる3種のメモリ管理方式があり，基本メモリ管理方式と拡張メモリ管理方式1では，シーケンシャル型，拡張メモリ管理方式2では遅延リリース型[10]の一貫性制御に分類することができる．

2.3 基本メモリ管理方式

2.3.1 メモリラインとメモリ制御ステータスフィールド

サーバとモバイル端末のメモリの共通領域を，ラインと呼ぶ固定長の領域に分割し，各ラインに対応したメモリ制御ステータスフィールドをエントリを持つメモリ制御ステータスメモリを設ける．モバイル端末とサーバの同一番号のラインを1対1に対応させ，そのステータスをメモリ制御ステータスフィールドにより管理する．図2にその対応を示す．ラインをさらにブロックに分割し，メモリの書き込みや書き戻しをブロックの単位で管理する．ラインの大きさは16-4096バイト

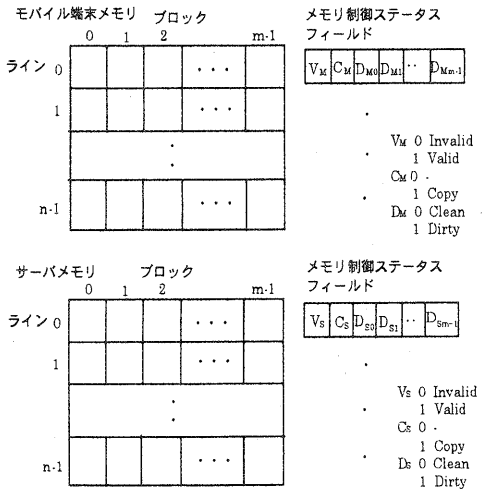

図2 メモリのマッピングとメモリ制御ステータスフィールド
Fig.2 Memory mapping and memory control status field

表1 メモリラインのステータス
Table 1 Status of memory line

VCD	メモリラインステータス
100	最新，内容は変更していない(相手側と一致)
101	最新，内容を変更した(相手側と不一致)
11-	最新でない可能性がある，相手側に書き写されている
0--	無効，初期状態

程度，ブロックの大きさは8-1024バイト程度を想定する．ブロックは一貫性制御のための書き戻し時のデータ量を減らすことが目的である．

2.3.2 メモリ制御ステータスフィールドとラインのステータス

メモリ制御ステータスフィールドはVビット，Cビット，および複数のDビットにより構成され，ラインのステータスを示す．Vは有効ビットでラインの有効無効を示し，Cはコピービットでラインの内容が相手側に書き写されているいるかどうかを示し，Dは変更ビットでブロック単位にデータの書込がおこなわれたかどうかを示す．メモリラインのステータスは，V，C，Dの組み合わせによって示され，それを表1に示す．メモリ制御ステータスフィールドはモバイル端末とサーバとで同一の意味を持ち，全く対称である．

メモリ制御ステータスの遷移等の詳細については文献[12]に記載する．

2.3.3 メモリアクセスとラインの転送制御

モバイル端末でのメモリアクセス時のメモリラインの転送制御を次に示す．モバイル端末でメモリアクセスが発生するとまずラインのステータスが調べられる．ラインのステータスが"最新" の場合には，メモリの内容は有効であり正常にリードまたはライトがおこなわれ，さらにライトの場合には変更ビットが1にセットされる．ラインのステータスが" 無効"，または" 最新でない可能性がある" の場合にはメモリ例外の割り込みを発生させる．以上はすべてハードウェア制御によって行われる．メモリ例外割込処理の中では，サーバと通信を行ってサーバ側にリモート割込を発生させ，モバイル端末のラインのステータスが" 無効" の場合には，サーバのラインを転送し，" 最新でない可能性がある" の場合にはサーバのステータスの変更ビットを調べ，1がセットされているブロックを転送する．同時にサーバのステータスを"最新でない可能性がある"にセットし，モバイル端末のステータスを" 最新" にセットして割込から戻る．メモリ例外割込処理が終了すると，割り込まれた命令が再実行される．以上はモバイル端末でのメモリアクセスであるがサーバでも同様である．

2.3.4 通信が接続不可の場合の制御

メモリ例外の割込が発生し，割込処理の中で相手側との通信が不可となった場合には，接続不可をアプリケーションに対して通知するとともに，メモリ制御ステータスフィールドを調べ現在のメモリステータスの内容を通知する．ステータスが" 最新でない可能性がある" の場合はラインの内容が相手側に書き写されており，その一部が変更されている可能性があることを示し，アプリケーションによっては，条件付きでデータを利用することが可能である．

2.4 拡張メモリ管理方式

2.3で述べた基本メモリ管理方式では，ラインはモバイル端末とサーバとで排他的に使用されアクセス権は一方のみに与えられる．リードオンリーの利用であってもラインの利用権を得てから使用する．これに対して，リードオンリーの場合には，モバイル端末とサーバとで並列にメモリを使用することを許せば，並列度を上げることができる．さらに書込が発生したときの制御方式において，一貫性制御を緩和することにより，さらに並列度が増す．ただしアプリケーションによっては制約が出るため，注意が必要である．MMMでは次の3種類のメモリ管理方式が選択可能である．なお説明で，アクセス権は読み出しと書き込みが可能，リード権は読み出しのみが可能の意味とする．

(1) 基本メモリ管理方式　2.3で述べた方式であり，共有ラインの一方にアクセス権が排他的に与えられる．

(2) 拡張メモリ管理方式1　一方のラインにアクセス権を与えられているとき，他方はリード権が与えられる．アクセス権が与えられている側で書込が発生すると，自身と他方のラインの変更ビットをセットする．リード権が与えられてる側で読み出しが発生し，かつ変更ビットがセットされている場合には割込を発生させ，変更のあったブロックを転送しラインを最新化してから読み出しをおこない，書込が発生した場合には，割込を発生させ，ラインの最新化とアクセス権を得てから書込をおこなう．

(3) 拡張メモリ管理方式2　一方のラインにアクセス権を与えられているときに，他方はリード権が与えられる．アクセス権が与えられている側で書込が発生しても自身のラインの変更ビットをセットするだけで，他方のラインには通知しない．リード権が与えられてる側で読み出しが発生しても，最新化せずに読む．メモリ内容の最新化は同期命令の発行により行う．書込が発生した場合には，割込を発生させ，ラインの最新化とアクセス権を得てから書込を行う．

以上のメモリ管理方式には，それぞれ長所と短所があり，特徴を表2にまとめる．モバイル端末の利用形態

表2 MMMのメモリ管理方式の特徴
Table.2 Characteristics of memory control system in MMM

	アクセス権，リード権	内容の一貫性	通信接続	データ転送量
基本管理方式	アクセス権は一方のラインのみ，アクセス権のないラインにアクセスを行った場合には，通信をおこない他方よりアクセス権を取得	常に最新	使用する領域のアクセス権をすべて取得すれば，その後は非接続の利用可	3方式の中では最も多い
拡張管理方式1	一方にアクセス権があるとき，他方はリード権，リード権のあるラインにライトを行った場合には通信を行い他方よりアクセス権を取得		常時接続して利用，ただしリードオンリーでの利用であれば，使用する領域のアクセス権またはリード権をすべて取得することで，その後は非接続で利用可	3方式の中で中間
拡張管理方式2		リード権のあるラインは必ずしも最新でない．最新化のためには同期処理が必要	使用する領域のアクセス権またはリード権をすべて取得すれば，その後は非接続での利用可，ただしアクセス権の場合にはリード及びライト，リード権の場合にはリードのみでの利用	3方式の中では最も少ない

による使い分けについては3.3で述べる．

2.5 メモリ空間の拡大

端末の共通メモリ領域を仮想メモリ方式で管理することにより、端末側のメモリ空間の拡大が可能となる。端末の共通メモリ領域は、仮想メモリであり、実際のメモリ内容は、サーバ側の共通領域に置かれる。すなわちサーバ側の共通メモリは，従来の仮想メモリの二次記憶に対応する．MMMのメモリ制御ステータスメモリは、共通メモリ領域に対応した実メモリに対して設けられる。

さらにMMMでは標準的な仮想アドレス方式のアーキテクチャを変更することなく利用可能である．

2.6 一般の分散共有メモリとの相違点

主要な相違点を次に示す．

(1) 通信路が低速なモバイル環境に適したラインサイズの選択が可能であり，さらラインを分割したブロック単位の書き戻し制御により通信量の最小化，通信コストの低減化が可能．

(2) 通信中の切断への対応及び通信非接続での使用が可能．

(3) モバイル端末の利用形態により一貫性制御方式の選択が可能．

(4) モバイルコンピューティング環境下での利用を前提に，モバイル端末とサーバとの対向するプロセッサ間でのメモリ共有に限定することで一貫性制御方式を比較的単純化．

(5) 他方のメモリを仮想記憶の二次記憶とすることでメモリ空間の拡大が可能．

3. MMMのモバイルコンピューティングシステムへの適用

3.1 MMMの適用

コンピュータネットワークの中でMMM方式を適用したモバイルコンピューティングシステムの例を図3に示す．図中，点線の部分がMMM方式の適用範囲例を示し，一つのサーバと複数のモバイル端末で構成されるモバイルコンピューティングシステムに対して適用される．図4にMMM方式によるモバイル端末とサーバのメモリ空間の共有方式を示す．一つの共有メモリ空間は同一時期には一つのサーバと一つのモバイル端末との間で共有される．モバイル端末は具体的には携帯情報端末(PDA)やモバイルPCであり，サーバは事務所や家庭のデスクトップPCを想定する．

3.2 通信媒体

通信媒体はモバイル環境下では一般的には携帯電話やPHSであるが，赤外線通信や無線LAN等の無線通信の他に，公衆電話網やLAN等の有線通信であってもよい．本稿では最も通信速度が遅い携帯電話での利用を前提に検討する．

携帯電話等の無線通信は，通信路が狭くまた通信コストがかかるため，通信は必要最小限にすることが求められる．MMM方式はアプリケーションが共通メモリにアクセスしたときに，通信によりサーバまたはモバイル端末より，ライン単位またはブロック単位でデータを書き移す，必要時要求方式（オンデマンド）のメモリ管理方式であり，さらにラインサイズを通信路に合わせ最適に選択可能であり，データ転送を必要最小限とすることができる．

3.3 モバイル端末の利用形態とMMM方式

モバイル端末のでの共用データの利用形態を次のように分類する．

(1) 必ず通信を行い，同期を取って利用する方式

(2) 移動前にモバイル端末に共用データをサーバよりダウンロードし，移動中はリードオンリーであれば，単独で使用し，内容を変更する場合にはサーバと通信し同期を取って利用する方式

(3) 移動前にモバイル端末に共用データをサーバよりダウンロードし，移動中も単独で読み出しまたは書込を行い，移動から戻った時にサーバと同期を取る方式

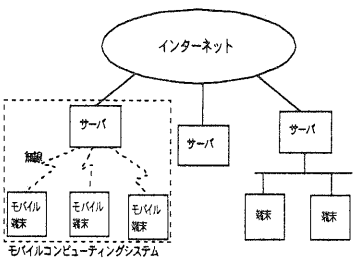

図3 コンピュータネットワークの中でのモバイルコンピューティングシステム
Fig.3 Mobile computing system in computer network

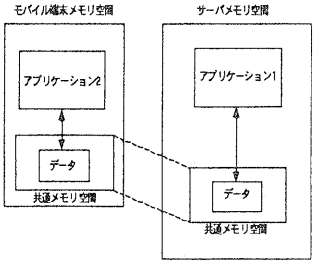

図4 MMM方式によるメモリ空間の共有
Fig.4 Memory sharing by MMM

(1)は，通信と表示機能に特化し，使用する場合にはサーバと連携して利用する端末向きであり，基本メモリ管理方式または拡張メモリ管理方式1が適している．(2)は通信非接続での使用を基本とした端末向きであり，拡張メモリ管理方式1が適している．(3)も通信非接続での使用が基本であるが，モバイル端末とサーバとで共用データを排他的に使用する場合は基本メモリ管理方式，並行して使用する場合には拡張メモリ管理方式2が適している．

3.4 ラインサイズの選択

MMM方式は一種の分散共有メモリであるが，一方ではキャッシュメモリの側面も持つ[11]．すなわちラインは使う必要が発生した時点で，自CPUに書き写される．書き移されたラインがプログラムの後続のメモリアクセスで利用されれば，通信の効率はよくなる．したがってラインサイズはある程度の大きい方がよいが，あまり大きくすると使われない部分が増えかえって効率が下がり，適切なラインサイズがあると考えられる．ラインサイズの最適値については4章でシミュレーションにより検証する．

3.5 ラインのプリフェッチ

プリフェッチ目的で単純にラインサイズを大きくすることは前記の問題があるが，要求のあったラインの次のライン，またはなんらかのアルゴリズムにより次に使用する可能性のあるデータを含んでいるラインのプリフェッチがプログラムの実行と並行可能であれば，実行速度の向上が可能できる．一方モバイル端末でのアプリケーションはインタラクティブなものがが多いと考えられ，モバイル端末のユーザーからの応答待ち時間をプリフェッチに利用することで性能の向上を図ることができる．MMMでは，プリフェッチなし，接続中ではあるがデータ転送をしていない時間を利用し1ラインプリフェッチ，可能な限りプリフェッチの3種の方式を選択可能とし，4章で評価する．

3.6 ブロックサイズの選択

MMMにおけるブロックは，データの書き戻し時のデータ伝送量を減らす目的でラインをさらに分割したものである．メモリの書込が発生した場合には対応するブロックの変更ビットをセットし，ラインの書き戻しが発生した時には変更ビットがセットされているブロックを書き戻す．変更されていないデータ転送の割合を減らすためには，ブロックのサイズは小さいほうが良いが，逆に書き戻し時のオーバーヘッドが増大することと，変更ビットのビット数が増大し，ハードウェアコストが増加するため，これも適性値があると考えられる．

4. MMM方式の評価

4.1 シミュレーションによる評価

モバイル端末上，またはモバイル端末とサーバとの双方でサーバ上に存在するデータを使用し，各種の処理を行うアプリケーションの例によりMMM方式の評価をおこなう．初期状態においてはアプリケーションのプログラム部は端末，またはモバイル端末とサーバとに格納されており，データ部はサーバに格納されているものとし，メモリの割り付けは，プログラム部は非共通領域に，データ部は共通領域に割り付けるものとする．

通信速度は9600bps，通信プロトコルのデータ以外の部分(コマンド，アドレス，データカウント，チェックコード)を10バイトとした．

作成したアプリケーションのモデルを次に示す

スケジュール管理

スケジュール管理のデータは，1項目のデータ量を48バイト，1日あたり1152バイトで50日分，総データ量は60Kバイトである．モバイル端末上での表示は1画面で1日分のスケジュールデータの表示とし，日の指定の順序，参照，変更を組み合わせた操作パターンの例を作成してシミュレーションを行った．

住所録管理

住所録のデータは，1レコード160バイト，500人分，総データ量86Kバイトである．モバイル端末上の画面は，初期画面，選択画面，及び表示変更画面がある．初期画面には名前を選択するための先頭の読みの文字の一覧表があり，最初に検索したい名前の先頭文字を一覧表から選択し，検索を指示すると選択画面になる．選択画面では選択された先頭文字を持つ名前が一覧表で表示されるので，これらの中から検索したい名前を選択すると，次に表示変更画面となり検索結果の住所録データが表示される．住所録データの変更及び新規登録は表示変更画面からおこなう．検索のみの操作と，変更を行う操作を組み合わせた操作パターンの例を作成してシミュレーションを行った．

4.1.1 ラインサイズの評価

まずラインサイズとデータ伝送時間との関係がどのように変化するかをアプリケーションを変えて調べる．

図5はスケジュール管理について，連続した日を順番に参照する操作パターン1，1週間あたり3日参照する操作パターン2，7日飛びに参照する操作パターン3，及び住所録管理については10人分のデータを検索した場合のラインサイズとデータ伝送時間の関係を示す．ラインサイズが減少するとデータ伝送時間が増大するのは，通信プロトコルのデータ部分の割合が減少し通

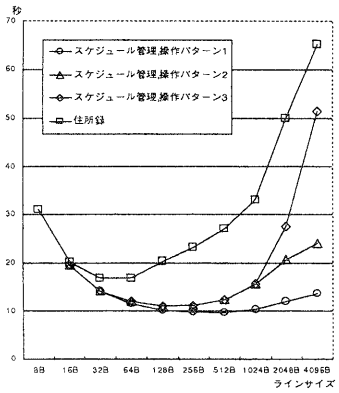

図5　ラインサイズとデータ伝送時間
－スケジュール管理，住所録管理－
Fig.5 Line size and Data transfer time
-Schedule book and Address bÇΣÇÈ

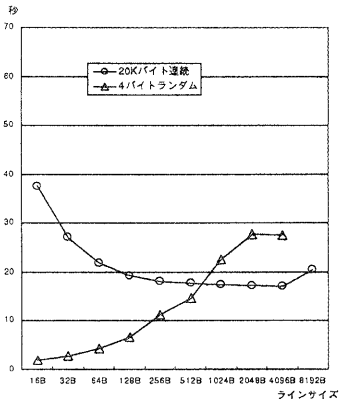

図6　ラインサイズとデータ伝送時間
-20Kバイト連続アクセス，
4バイトランダムアクセス-
Fig.6 Line size and Data transfer time
-Continuous accsess of 20K bytes and
Random access of 4 bytes-

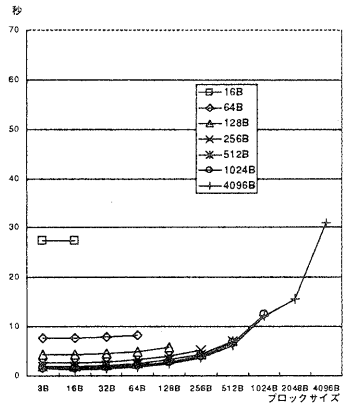

図7 ブロックサイズとデータ伝送時間
－スケジュール管理－
Fig.7 Block size and Data transfer time
-Schedule book-

信のオーバーヘッドの割合が増大するためであり，ラインサイズが増加したときにデータ伝送時間が増大するのは再利用しないデータを転送する割合が増加するためである．スケジュール管理の操作パターン1はラインサイズが大きくなってもに有効に再利用されるデータの割合が大きく、操作パターン3はラインサイズが大きくなると再利用されるデータの割合が少なくなることがわかる。スケジュール管理は1Kバイト程度単位のデータを，連続または非連続にアクセスする例であり比較的局所性のあるケースであり，住所録管理はインデックスでソートされているデータをアクセスするため，4バイトのインデックスを複数回アクセスしたあとに100バイト～200バイトのレコードをアクセスするという比較的局所性が少ない例である．

図6は20Kバイトのデータを連続したアドレスでアクセスした場合と，32Kバイトのデータの中より4バイト単位のデータをランダムのアドレスでアクセスする場合のラインサイズとデータ伝送時間の関係を示す．20Kバイトの連続アクセスは最も局所性のある例，4Kバイトのランダムアクセスは局所性が全くない例である．

以上を総合すると32～256バイト近辺が最適であることがわかる．ただしこれは通信プロトコルのオーバーヘッドが10バイトの場合であり，その大小により最適値は変動する．

4.1.2 ブロックサイズの評価

図7にモバイル端末とサーバとの相互動作時のブロックサイズとデータ伝送時間の関係を示す．データ伝送時間はラインサイズの影響を大きく受け，ラインサイズは大きい程よく，ブロックサイズは小さい程よいことがわかる．ただし8バイトのブロックサイズの場合にはブロック指定の情報量が増加するため若干時間が増加する．またブロックサイズを小さくすると制御のためのビット数が増加するため，ハードウェアコストとの兼ね合いとなる．

4.1.3 MMM方式と従来方式との比較評価

次に初期状態の時にサーバ側にあるデータを，モバイル端末とサーバ上のアプリケーションが相互に使用する場合を，スケジュール管理を使用し，MMM方式と従来方式とで比較評価する．ブロックサイズは16バイトの固定とした．

従来方式でも実際に使用するデータのみを伝送する方式が可能であるが，モバイル端末のアプリケーションは通信によるデータ伝送を意識する必要があり，またサーバ側でもデータハンドリング用のプログラムが必要になる等，アプリケーションの構築が複雑になる．このため従来方式としては，実行の始めに全データを読み込み，実行の最後に全データを書き戻す，一括伝送方式を比較対象とした．

モバイル端末とサーバの動作のシナリオは，最初にモバイル端末上で10日分のデータが参照され，その内5日分の中のデータを更新，その後サーバでモバイル端末上で参照・更新された10日分のデータが参照される．次にモバイル端末上で10日分のデータが参照され，その内5日分の中のデータを更新，最後にサーバ側でモバイル端末上で参照・更新された10日分のデータが参照される．1日分のデータが画面に表示された後，参照だけの場合は平均3秒の時間、データに変更のある場合には前記の参照時間に加えて，24文字分で約12秒の書込時間を要するものとした。総実行時間には、プログラムの実行時間の他に、データの参照時間、書込時間、および通信の呼設定時間を含む。通信の呼設定時間は10秒

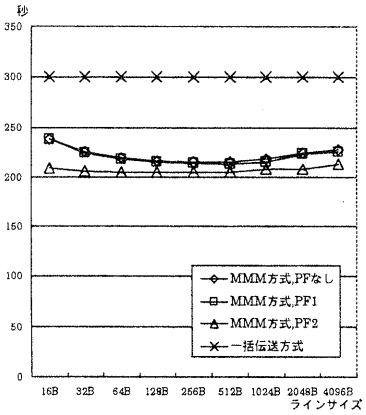

図8 ラインサイズと総実行時間
ーモバイル端末上のスケジュール管理ー
Fig.8 Line size and Total execution time
- Schedule book on the mobile terminal-

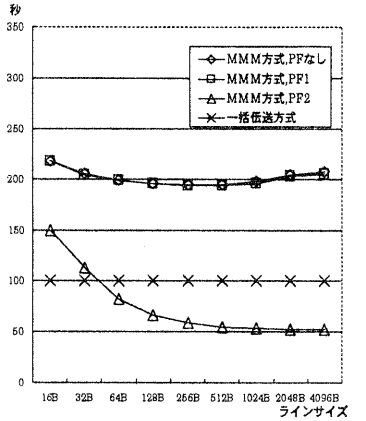

図9 ラインサイズと通信接続時間
ーモバイル端末上のスケジュール管理ー
Fig.9 Line size and Connection time
- Schedule book on the mobile terminal-

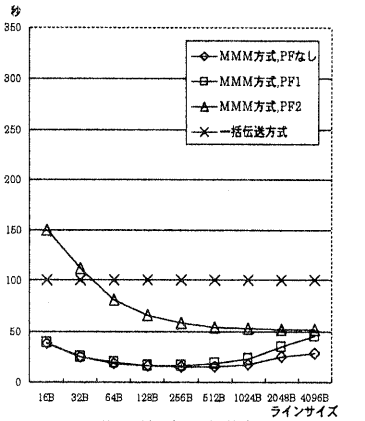

図10 ラインサイズとデータ伝送時間
ーモバイル端末上のスケジュール管理ー
Fig.10 Line size and Data transfer time
-Schedule book on the mobile terminal-

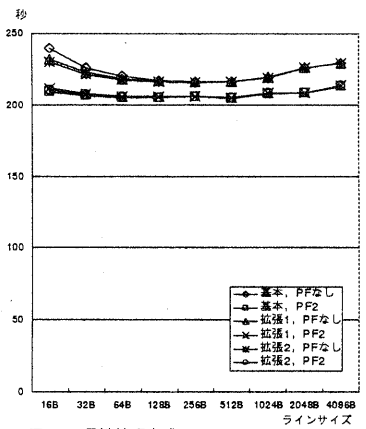

図11 一貫性管理方式についての
ラインサイズと総実行時間
ーモバイル端末上のスケジュール管理ー
Fig.11 Line size and total execution time
on memory consistency models
-Schedule book on the mobile terminal-

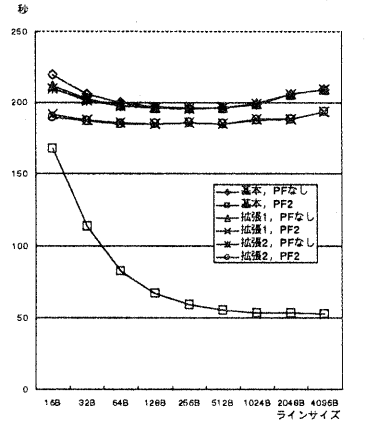

図12 一貫性管理方式についての
ラインサイズと通信接続時間
ーモバイル端末上のスケジュール管理ー
Fig.12 Line size and communication connection
time on memory consistency models
-Schedule book on the mobile terminal-

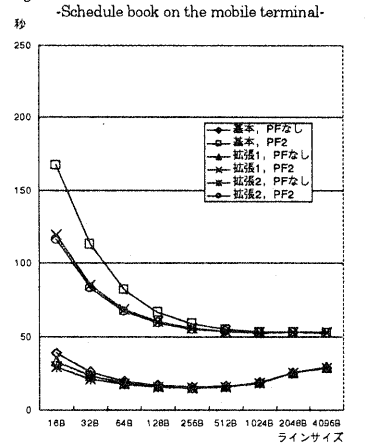

図13 一貫性管理方式についての
ラインサイズとデータ転送時間
ーモバイル端末上のスケジュール管理ー
Fig.13 Line size and data transfer time
on memory consistency models
-Schedule book on the mobile terminal-

とした。さらにMMM方式の場合にはプリフェッチする場合とプリフェッチをしない場合を示す.プリフェッチは直前に転送がおこなわれたラインに続くラインを転送する方式とし,1ラインプリフェッチする場合(図中PF1で示す，図8～図10)と，通信の空き時間のあるかぎりプリフェッチをおこなう場合(図中でPF2で示す，図8～図13)の，2とおりの場合について示す.通信接続は，一括伝送方式の場合は最初のデータの書き移しの終了後一旦切断し,最後の書き戻し時に再度接続するものとした.MMM方式の場合は，実行中に全データが書き移されればその時点で通信を切断し,最後の書き戻し時に再接続し，実行中に全データの書き移しが終了しなければ,最後になるまで通信の切断はおこなわないものとした.

図8はモバイル端末上でのラインサイズと総実行時間の関係を示す。MMM方式の時間が一括伝送方式に比較し大幅に少なくなっているが,その差はほぼ一括伝送方式の全データの書き移しと書き戻しの時間分に相当する。総実行時間は，できるだけプリフェッチをおこなうPF2の場合が最も少なくなることがわかる.MMM方式の場合ラインサイズによる差の割合が少なく見えるのは、画面を参照している時間や書込の時間の占める割合が大きいためである。

図9はモバイル端末側のラインサイズと通信接続時間の関係を示す。MMM方式のプリフェッチなしと1ラインプリフェッチするPF1の時間が一括伝送に比較し大きくなっている.これは全ライン転送が終了しないため最後まで通信接続が切断できないことによる.通信の空き時間を利用してできる限りプリフェッチするPF2の場合,ラインサイズが大きくなるに従い時間が少なく

なるのは、実行の途中で全データの書き移しが終了し通信接続を切断するためである。通信が接続時間による従量課金の場合にはPF2の方式が有利であることがわかる。

図10はモバイル端末側のラインサイズとデータ伝送時間の関係を示す。実際に必要とするデータのみを伝送するプリフェッチをおこなわない場合が最もデータ伝送時間が少なくなり、さらにラインサイズが128バイトから512バイトの近辺で最小になることがわかる．通信量で課金される場合にはこの方式を選択するのが有利であることがわかる．

4.1.4 一貫性管理方式の評価

図11，12，13は一貫性管理方式について，ラインサイズと総実行時間，通信接続時間およびデータ伝送時間の関係を示す.使用するアプリケーションは4.1.3と同じスケジュール管理で，モバイル端末とサーバとで動作させ，サーバ上でもリードライトを行う．図11の総実行時間では，プリフェッチしないグループとプリフェッチするグループに分かれており，プリフェッチすることにより総実行時間が少なくなることがわかる．ラインサイズの小さいところでは基本管理方式の時間が大きくなっている．図12の通信接続時間では，PF2のプリフェッチケース場合，基本管理方式のみがプリフェッチが完了し途中で通信の切断を行っている．拡張管理方式1と2が途中で切断できないのは，使用アプリケーションのシナリオがモバイル端末とサーバとでリードライトをおこなうため常時接続を必要とするためである．図13のデータ伝送時間では，プリフェッチを行うことにより，ラインサイズが小さくなるに従いオーバーヘッドが急速に大きくなっている．これはアクセス権またはリード権を端末側に戻すための時間である．プリフェッチを行わないケースで比較すると，オーバーヘッドが小さいのは拡張管理方式2，拡張管理方式1，基本管理方式の順であることがわかる．

5．おわりに

以上モバイルコンピューティングシステム向きのメモリ管理方式MMMの提案とその評価について述べた．MMMは，分散共有メモリの側面を持つが，低速度の通信路に合わせたラインサイズの選択が可能，通信量を減らすためのブロックによる書き戻し制御，通信路の切断への対応や非接続での利用，モバイル端末の利用形態に応じた一貫性制御の選択に関し，従来のものとは異なっている．本方式により，モバイル端末とサーバとの間で共通データの同期が可能となり，アプリケーションの構築では複雑な通信手順が不要となり，アプリケーションの実行時間の短縮，通信の効率化を図ることができ，さらにモバイル端末からサーバのリソースの利用が容易となることを示した．通信の効率化は，必要とするデータのみを伝送することの他に，モバイル端末の利用形態を利用したプリフェッチ方式の導入により，通信と他の動作を並行動作させることでさらに大きな効果が出せることを示した．

今後の課題は同一メモリ領域を共有する複数モバイル端末環境への拡張である．今回提案のMMM方式はモバイル端末とサーバとで同一時間では1対1に対応させる方式であり，同一データを他のモバイル端末が使用する場合にはシリアル化する必要がある．同時に複数端末で使用可能とするためにはMMM方式の拡張が必要であり，次の課題である．

参考文献

1) Truman, T. E., Pering, T. P., Doering R.and Brodersen, R. W.: The InfoPad Multimedia Terminal: A Portable Device for Wireless Information Access, IEEE Trans. on Computers, Vol.47, No.10, Oct. (1998).
2) Joseph, A. D., Tauber, J. A. and Kaashoek, M. F.: Mobile Computing with the Rover Toolkit, IEEE Trans. on Computers, Vol.46, No.3, March (1997).
3) Kistler, J. J. and Satyanarayanan, M.: Disconnected Operation in the Coda File System, ACM Trans. on Computer Systems, Vol.10, No.1, February (1992)
4) Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J. and Hauser, C. H.: Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System, Proc. 15th Symp. Operating Systems Principles, Dec. (1995).
5) Lenoski, D., Laudon, J. Gharachorloo, K., Gupta, A. and Hennessy, J.: The Directory-Based Cache Coherence Protocol for DASH Multiprocessor, 17th ISCA, (1990).
6) Kuskin, J. et al: The Stanford FLASH Multiprocessor, Proc. 21st ISCA, (1994).
7) タネンバウム，A. S.，水野他訳:分散オペレーティングシステム，プレンティスホール，(1996).
8) 平木敬，丹羽純平，松本尚:分散共有メモリに基づく計算機クラスタ，情報処理，Vol.39, No.11, Nov. (1998)
9) Adve, S. V. and Gharachorloo, K.: Shared Memory Consistency Models: A Tutorial, IEEE Computer, Vol.29, No.12, (1996)
10) Keleher, P., Cox, A. L. and Zwaenepoel, W.: Lazy Release Consistency for Software Distributed Shared Memory, Proc. of the 19th ISCA, (1992)
11) Alan Jay Smith: Cache Memories, Computing Surveys, Vol. 14, No. 13, Sept. (1982).
12) 清水正貴，横山繁盛，渡辺尚，水野忠則:モバイル環境に適したメモリ管理方式の評価， DICOMO, (1999)