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

横山繁盛[†] 渡辺尚[††] 水野忠則[††]

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 Environments

Shigemori Yokoyama,[†] Takashi Watanabe[††] and Tadanori Mizuno[††]

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 Environments (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 makes application programs on mobiles terminal and on servers to be developed more easily and increases the efficiency of communications. MMM is 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，小型化にともない，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動作時間が短く，CPU 性能，メモリ容量，入出力装置等に制約がある等の課題が多い．また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．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では，モバイル端末とサーバとの間で通信が必要であり，通信も各種の媒体や通信方式があるため，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)]，通信の切断の可能性がある環境下でのアプリケーションの構築法[2),3)]やファイル方式[4),5)]等のソフトウェア方式による研究がある．

本論文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 Environments）を提案し，その評価を行う．

† 三菱電機株式会社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 University

MMMにおいて，モバイル端末は，必要とするデータの存在するサーバ上のメモリ空間を自メモリ空間の一部として制御することにより，アプリケーションは通信を意識することなくサーバ上のデータを単なるメモリとしてアクセスすることができる．サーバでも同様である．これによりアプリケーションの構築の容易化，データの同期化を図ることができ，必要時，必要な量のデータ通信により，通信による遅延の最小化と，通信コストの低減が可能となる．

まず2章でMMMのアーキテクチャについて述べ，特に2.7節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3章でMMMのモバイルコンピューティングシステムへの適用について説明し，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[6)~9)]．

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行う方式である．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を容易に利用可能とし，低速で切断の可能性のある通信路および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としてメモリを共有する方式であり，**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．

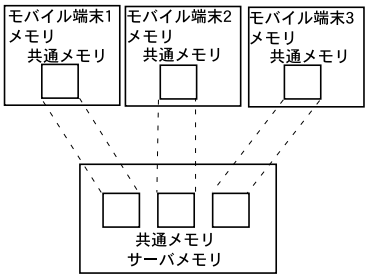


図1 MMMのメモリ管理方式
Fig. 1 Concept of MMM.

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[10)]．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，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に使用している．

2.2 メモリの一貫性制御

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るが，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[11)~16)]．MMMでは次節以降で述べる基本メモリ管理方式，拡張メモリ管理方式1および拡張メモリ管理方式2の3種のメモリ管理方式がある．最初の2つがシーケンシャル型，3番目が一種の遅延リリース型[15)]の一貫性制御方式に分類することができる．なお遅延リリース型一貫性制御方式は，緩和型メモリ一貫性制御方式の1つであり，1つのプロセッサの共有メモリの排他使用開始時に，まず共有メモリの内容のほかとの一致をとり，その後メモリをローカルに変更，排他使用の終了時には変更を他に通知しないという方式であり，必要になったときにメモリ内容の一致をとることで，一貫性制御のための通信の軽減を図った方式である．

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

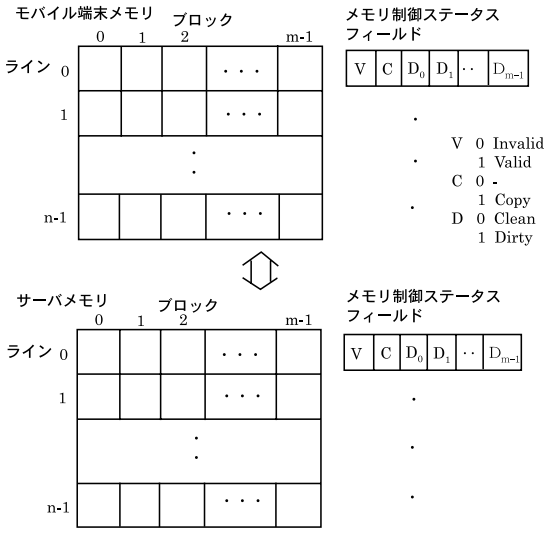


図 2 メモリのマッピングとメモリ制御ステータスフィールド
Fig. 2 Memory mapping and memory control status field.

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に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，メモリの変更をブロックの単位で管理可能にする．ラインの大きさは 16〜4096 バイト程度，ブロックの大きさはラインを 2 のべき乗を単位に分割し，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り，メモリステータスの管理の基本はライン単位である．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て最適に選択可能である．なおラインサイズ，ブロックサイズはモバイル端末とサーバとでそれぞれ同一でなければならず，その切替えを行う場合には共通メモリを使用するアプリケーションの実行開始前に管理プログラムにより行われなければならない．アプリケーション自身はそれらを意識する必要はない．

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る（図 2 参照）．メモリラインのステータスは，V，C，D の組合せによって示される．それを**表 1** に示す．V は有効ビットでラインの有効無効を示し，C はコピービットで有効ビット（V）が 1 の場合に有効であり，0 は最新のデータとアクセス権が自身にあることを示し，1 は最新のデータとアクセス権が相手側にあり，自身は “最新でない可能性がある” ことを示す．メモリアクセスが許されるのは “最新（VC=10）” の場合のみであり，他のステータスではメモリ例外の割込みが発生する．D は変更ビットでラインのステータスが “最新” の場合にのみ有効であり，ラインの中のブロック数分のビットがあり，ブロック単位にデータの変更が行われたかどうかを示す．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，まったく対称である．

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（相手側が最新である）
0--	無効，初期状態

メモリ制御ステータスの遷移等の詳細については文献 17) 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行われ，さらにライトの場合には対応するブロックの変更ビットが 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込みが発生する．以上はすべてハードウェア制御によって行われる．メモリ例外割込み処理の中では，サーバと通信を行ってサーバ側にリモート割込みを発生させる．サーバ側ではモバイル端末のラインのステータスが “無効” の場合には，まだライン単位の転送が行われていないためサーバのラインを転送し，“最新でない可能性がある” の場合にはすでにライン転送が行われているため，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のみを転送する．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，リモート割込みを終了させる．モバイル端末側ではステータスを “最新” にセットし，メモリ例外割込み処理を終了させ，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動作であるが，サーバでのメモリアクセス時のメモリライン転送制御も同様である．

2.3.4 ラインの競合制御

モバイル端末とサーバとが同一ラインを同時に使用した場合，タイミングによってはラインの取り合いとなり，デッドロックが発生する可能性がある．これはラインの使用権がない側でメモリアクセスによる割込みが発生し，ラインのアクセス権を得て割込み処理から戻った直後に，他方の側で同一ラインに対するメモリアクセスが発生し，ラインの使用権がなくなっているため割込みが発生するという，両方の側で割込みが繰り返し発生し命令が進まなくなるタイミングが発生した場合である．この防止のため，メモリ例外割込みが発生してから，割込み終了直後の最初の命令の実行完了までの間，リモート割込みを禁止する．これによりモバイル端末とサーバとが同一ラインを使用した場合でも，1 命令ずつ交互に実行可能となる．

2.3.5 通信が接続不可の場合の制御

メモリ例外の割込みが発生し，割込み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” の場合にはラインの内容が一度最新になり，その後アクセス権が相手側に移り，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 節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アクセス権を得てから使用しなければならない．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．さらに書き込みが発生したときのメモリ制御方式において，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 節で述べた方式であり，共通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き込みが発生すると，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い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，割込みを発生させ，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行い，書き込みが発生した場合には，割込みを発生させ，ラインの最新化とアクセス権を得てから書き込みを行う．

(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，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き込み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いる側で読み出しが発生しても，最新化せずに読み，書き込みが発生した場合には，割込みを発生させ，ラインの最新化とアクセス権を得てから書き込みを行う．メモリ内容の最新化は同期命令の発行により行う．

以上のメモリ管理方式には，それぞれ長所と短所があり，モバイル端末の利用形態による使い分けについては 3.3 節で述べる．

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領域は，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．仮想メモリのページサイズと MMM のラインサイズが異なる場合には一方が他方の整数倍の関係となるようにサイズを選択する．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 ハードウェア構成

図 3 に MMM のハードウェア構成のブロック図を示す．基本的には従来の標準的なアーキテクチャを持つ CPU のハードウェアを変更することなく，メモリ制御ステータスメモリ用の RAM とその制御回路を付加することで実現可能である．図 3 において点線で囲まれたメモリ制御ステータス制御部が，MMM を構成するための付加回路であり，ライン切替えは，ラインサイズを変更するときにメモリアドレスの中のラインアドレスとして使用する範囲を切り替えるためのライ

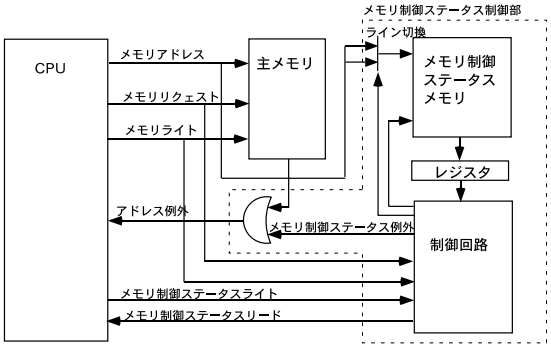


図 3 MMM のハードウェア構成
Fig. 3 Hardware architecture of MMM.

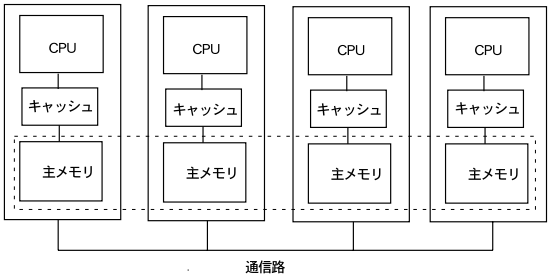


図 4 一般の分散キャッシュ型共有メモリ
Fig. 4 Distributed shared memory with cache.

ンアドレス制御を示す．ブロックの選択およびサイズの切替えは図示していないが，まずラインアドレスでメモリ制御ステータスフィールドを選択し，その中のDビットのフィールドをラインアドレス内のブロックアドレスで選択する．

前記のようにメモリ制御ステータスメモリのアクセスのため，メモリアクセス時にメモリアドレスがCPUの外部に出ていることが必要である．このためサーバ，モバイル端末とも内部にキャッシュメモリを持つことはできるが，その場合にはストア（ライト）スルー方式の制御方式とすることが必要である．

2.7 一般の分散共有メモリおよび分散キャッシュ型共有メモリとの相違点

主要な相違点を次に示す．

(1) 分散キャッシュ型共有メモリでは，**図 4** に示すように主メモリは全体として 1 つのアドレス空間を構成，各主メモリはアドレス空間の一部を構成し，各プロセッサは自身のキャッシュ内に必要とする主メモリの写しをキャッシングすることによりメモリを共有する．これに対してMMM では**図 5** に示すように主メモリの一部に共通メモリ領域を設けて，主メモリ間で同一内容となるように管理しこれをプロセッサ間で共有する方式であり，分散キャッシュ型共有メモリのキャッシュに対応するものは存在しない．

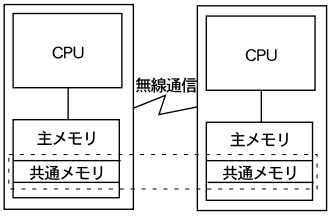
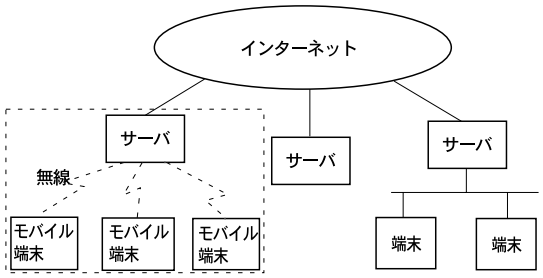


図 5 MMM の共通メモリ方式による共有メモリ
Fig. 5 Distributed shared memory by common memory in MMM.



モバイルコンピューティングシステム
図 6 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. 6 Mobile computing system in computer network.

(2) 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，さらに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．

(3) 通信中の切断への対応および通信非接続での使用が可能．

(4) MMM では図 5 に示すように，分散キャッシュ型共有メモリに必須なキャッシュにあたるものがなく，さらに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管理するステータスの数を減らし，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 6** に示す．図中，点線の部分が MMM 方式の適用範囲例を示し，1 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．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 等の無線通信のほかに，公衆電話網や LAN 等の有線通信であってもよい．本論文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写す，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，同期をとって利用する方式
(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とって利用する方式
(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き込みを行い，移動から戻ったときにサーバと同期をとる方式

(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[18]．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．書き写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，通信の効率は良くなる．一般的にはメモリのアクセスには局所性があり，アクセスのあった近傍のデータが次に使われる確率は高い．したがって，ラインサイズはある程度の大きい方が良いが，あまり大きくすると使われない部分が増加し，かえって効率が下がるため，適切なラインサイズがあると考えられる．

ラインのサイズを拡大することは，あらかじめ次に使う可能性のあるデータをプリフェッチすることになる．MMM 方式の場合は，ラインの伝送が終了するまでプログラムの実行が中断するため，速度を上げる効果は少なく，逆に待ち時間が増加するため操作感が悪くなる．操作感を優先しプログラムの中断による待ち時間を 1 秒程度以内とすると，通信速度が 9600 bps の場合，ラインサイズの上限は 1 K バイト程度となる．適切なラインサイズを考えるうえでのもう 1 つの制約は主たる通信媒体が無線のため，ラインサイズを大きくすると通信誤りが多くなった場合に通信効率が低下することである．

ラインサイズを L バイトとすると，N バイトのデータを参照するのに必要なデータ転送時間 T は次のように表すことができる．

$$T = 8N(1 + H/L)/(C * F(L))$$

H ： 通信オーバヘッド　バイト数
C ： 通信速度　bps
$F(L)$： ラインの使用率

$F(L)$ はラインの使用率でラインサイズの関数であり，アプリケーションにより異なるが，一般的にはラインサイズが小さければ 1 に近づき，ラインサイズが大きくなれば値は小さくなる．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ある．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多いと考えられ，モバイル端末のユーザ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，接続中の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および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

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

モリの書き込み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とき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方が良いが，逆に書き戻し時のオーバ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 方式の評価を行う．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の双方に格納されており，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 9600 bps，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．

(1) スケジュール管理

スケジュール管理のデータは，30 分単位で予約可能とし，用件を 24 文字以内に書くとして，1 項目あたりのデータ量を 48 バイト，1 日あたり 12 時間分として 48 × 24=1152 バイト，全部で 50 日分として総データ量は 60 K 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．

(2) 住所録管理

住所録のデータは，氏名，住所，電話番号，備考等を合わせて 1 レコードあたり 160 バイトとし，500 人分で総データ量は 86 K 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，および表示変更画面からなる．初期画面には名前を選択するため，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，検索を指示することにより選択画面になる．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および新規登録は表示変更画面から行う．検索のみの操作と，データの一部の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．

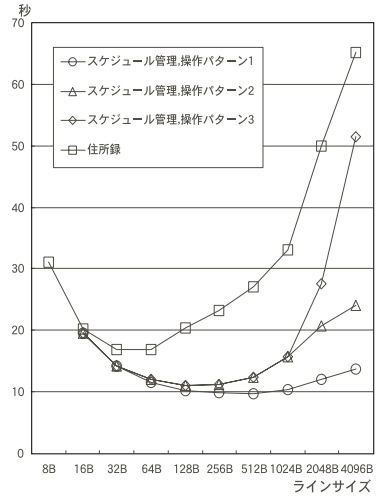


図 7 ラインサイズとデータ伝送時間—スケジュール管理，住所録
Fig. 7 Line size and data transfer time – schedule book and address book.

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スケジュール管理と住所録管理のアプリケーションを使用し以下の操作シナリオを想定してシミュレーションを行った．

スケジュール管理については，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 の 3 種の代表パターン，および住所録についてはランダムに 10 人分のデータを検索する場合についてラインサイズとデータ伝送時間の関係を**図 7** に示し，評価結果を次に示す．

(1) スケジュール管理についてはラインサイズが 64 バイトから 512 バイト近辺，住所録管理については 16 バイトから 128 バイト近辺が最もデータ伝送時間が短くなることが分かる．

(2) 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信のオーバ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．

(3) 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

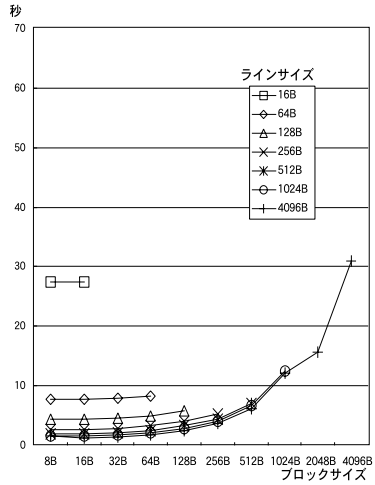


図 8 ブロックサイズとデータ伝送時間—スケジュール管理
Fig. 8 Block size and data transfer time – schedule book.

が少なくなることが分かる．

(4) スケジュール管理は 1 K 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 100〜200 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．

(5) 以上総合すると 32〜256 バイト近辺が最適であることが分かる．ただしこれは通信プロトコルのオーバヘッドが 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 8 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，ラインサイズをパラメータにして示す．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ほど良く，ブロックサイズは小さいほど良いことが分かる．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ため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

次に初期状態のとき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．シミュレーションの条件を次に示す．

(1) ブロックサイズは 16 バイトの固定とした．

(2) 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の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，アプリケーションの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．

(3) モバイル端末とサーバの動作のシナリオは，たとえば営業マンが客先での打合せでスケジュールの予約をモバイル端末で行い，事務所のサーバで秘書がスケジュールのチェックを行うというような使用方法を仮定する．まず最初にモバイル端末上で 10 日分のデータが参照され，そのうち 5 日分の一部のデータを更新，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，そのうち 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．

(4) 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 秒の書き込み時間を要するものとした．

(5) 総実行時間には，プログラムの実行時間のほかに，データの参照時間，書き込み時間，および通信の呼設定時間を含む．通信の呼設定時間は 10 秒とした．

(6) 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行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 9** 〜**図 12**）と，通信の空き時間のあるかぎりプリフェッチを行う場合（図中で PF2 で示す，図 9 〜図 12）の，2 通りの場合について示す．

(7) 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いったん切断し，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写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写しが終了しなければ，最後まで通信の切断は行わな

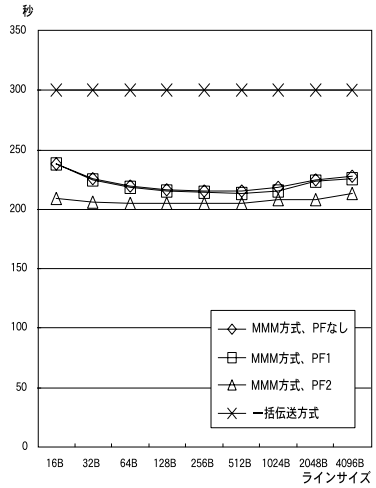


図 9 ラインサイズと総実行時間—モバイル端末上のスケジュール管理

Fig. 9 Line size and total execution time – schedule book on the mobile terminal.

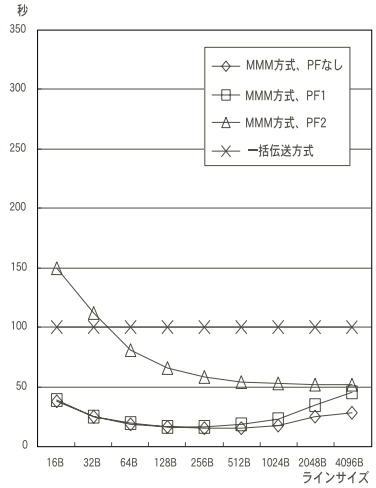


図 11 ラインサイズとデータ伝送時間—モバイル端末上のスケジュール管理

Fig. 11 Line size and data transfer time – schedule book on the mobile terminal.

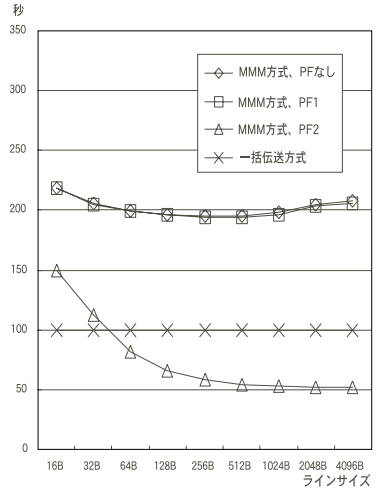


図 10 ラインサイズと通信接続時間—モバイル端末上のスケジュール管理

Fig. 10 Line size and connection time – schedule book on the mobile terminal.

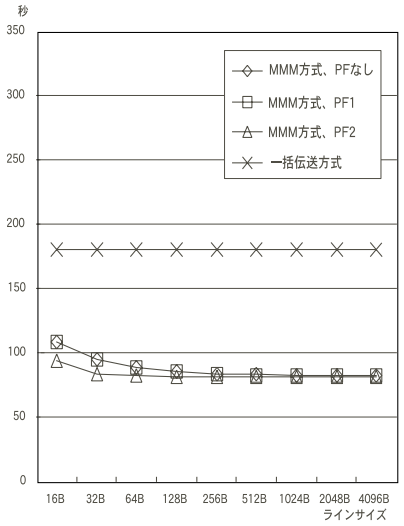


図 12 ラインサイズと総実行時間—サーバ上のスケジュール管理

Fig. 12 Line size and total execution time – schedule book on the server.

いものとした．

評価結果を以下に示す．

(1) 図 9 と図 12 はモバイル端末とサーバ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写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行う PF2 の場合が最も少なくなることが分かる．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き込みの時間の占める割合が大きいためである．

(2) 図 10 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写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分かる．

(3) 図 11 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行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分かる．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分かる．

一貫性管理方式に関する評価等については文献 19) で記述する．

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，共通メモリによる共有方式，低速度の通信路に合わせたラインサイズの選択が可能，データ伝送量を減らすためのブロック単位の書き戻し制御，通信路の切断への対応や非接続での利用等に関し，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，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ほかに，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．今回の提案の 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．

謝辞 本研究を進めるにあたり，各種シミュレーションに協力いただいた静岡大学大学院の清水正貴氏ならびに奥田隆弘氏に感謝いたします．

参考文献

1) Truman, T.E., Pering, T., Doering R. and Brodersen, R.W.: The InfoPad Multimedia Terminal: A Portable Device for Wireless Information Access, *IEEE Trans. Comput.*, Vol.47, No.10, pp.1073–1087 (1998).
2) Joseph, A.D. and Kaashoek, M.F.: Building Reliable Mobile-Aware Applications using the Rover Toolkit, *MOBICOM*, pp.117–129 (1996).
3) Joseph, A.D., Tauber, J.A. and Kaashoek, M.F.: Mobile Computing with the Rover Toolkit, *IEEE Trans. Comput.*, Vol.46, No.3, pp.337–352 (1997).
4) Kistler, J.J. and Satyanarayanan, M.: Disconnected Operation in the Coda File System, *ACM Trans. Computer Systems*, Vol.10, No.1, pp.3–25 (1992).
5) 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*, pp.172–183 (1995).
6) Lenoski, D., Laudon, J., Gharachorloo, K., Gupta, A. and Hennessy, J.: The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, *17th ISCA*, pp.148–251 (1990).
7) Kuskin, J., et al.: The Stanford FLASH Multiprocessor, *Proc. 21st ISCA*, pp.302–313 (1994).
8) Lovett, T. and Clapp, R.: STiNG: A CC-NUMA Computer System for the Commercial Marketplace, *Proc. 23rd ISCA*, pp.308–317 (1996).
9) Matsumoto, T., Nishimura, K., Kudoh, T., Hiraki, K., Amano, H. and Tanaka, H.: Distributed Shared Memory Architecture for JUMP-1 a general-purpose MPP prototype, *Proc. IEEE 1996 International Symposium on Parallel Architectures, Algorithms and Networks*, pp.131–137 (1996).
10) Li, K.: IVY: A Shared Virtual Memory System for Parallel Computing, *Proc. 1988 ICPP*, pp.94–101 (1988).
11) タネンバウム，A.S., 水野ほか（訳）：分散オペレーティングシステム，プレンティスホール出版 (1996).
12) 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, pp.1078–1083 (1998).
13) Adve, S.V. and Gharachorloo, K.: Shared Memory Consistency Models: A Tutorial, *IEEE Comput.*, Vol.29, No.12, pp.66–70 (1996).
14) Iftode, L., Dubnicki, C., Felten, E.W. and Li,

K.: Improving Release-Consistent Shared Virtual Memory using Automatic Update, *Proc. 2nd Inter. Symp. on HPCA*, pp.14–25 (1996).
15) Keleher, P., Cox, A.L. and Zwaenepoel, W.: Lazy Release Consistency for Software Distributed Shared Memory, *Proc. 19th ISCA*, pp.13–21 (1992).
16) Keleher, P., Cox, A.L., Dwarkadas, S. and Zwaenepoel, W.: TreadMarks: Disributed Shared Memory on Standard Workstations and Operating Systems, *Proc. 1994 Winter USENIX*, pp.115–131 (1994).
17) 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，マルチメディア，分散，強調とモーバイル（DICOMO'99）シンポジウム，Vol.99, No.7, pp.459–464 (1999).
18) Smith, A.J.: Cache Memories, *Computing Surveys*, Vol.14, No.3, pp.473–530 (1982).
19) 横山繁盛，奥田隆弘，水野忠則，渡辺 尚：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，情報処理学会研究報告，Vol.2000, No.14, pp.33–40 (2000).

(平成 11 年 12 月 22 日受付)
(平成 12 年 7 月 5 日採録)

横山 繁盛（正会員）

1945 年生．1968 年東京大学工学部電子工学科卒業．同年三菱電機（株）入社．汎用コンピュータ，エンジニアリングワークステーション等のコンピュータハードウェア開発に従事．1998 年静岡大学大学院理工学研究科博士後期課程に社会人学生として入学，在学中．2000 年 3 月より（財）日本規格協会に出向．電子情報通信学会会員．

渡辺 尚（正会員）

1959 年生．1982 年大阪大学工学部通信学科卒業．1984 年同大学院前期課程修了．1987 年同大学院博士後期課程修了．工学博士．同年，徳島大学工学部情報工学科助手．1990 年静岡大学工学部情報知識学科助教授．現在同大学情報学部情報科学科助教授．1995 年文部省在外研究員（カリフォルニア大学アーバイン校）．1997 年情報処理学会モーバイルコンピューティング研究会幹事．計算機ネットワーク，分散システム，マルチエージェントシステムに関する研究等に従事．訳書「計算機設計技法」等．電子情報通信学会，IEEE 各会員．

水野 忠則（正会員）

1945 年生．1968 年名古屋工業大学工学部経営工学科卒業．同年三菱電機（株）入社．1993 年静岡大学工学部教授，1995 年より同大学情報学部教授．工学博士．分散システム，コンピュータネットワーク，モーバイルコンピューティングに関する研究・開発に従事．著書としては，「分散システムコンセプトとデザイン」（電気書院，訳），「コンピュータネットワーク」（オーム社），「インターネットとコンピュータ」（プレンティスホール出版）等多数．情報処理学会標準化委員会委員，研究会幹事，理事等を歴任，フェロー，ベストオーサ賞受賞．電子情報通信学会，IEEE，ACM 各会員．