システムソフトウェアと
オペレーティング・システム 73－2
（1996．8．26）

PPRAM ベース・システム向け分散共有メモリ・システムの提案

村上和彰[†,††] 吉井　卓[†]
岩下茂信[†] 宮嶋浩志[†]

本稿では，メモリ–マルチプロセッサ一体型 ASSP である *PPRAM* を構成要素とするコンピュータ・システム上での分散共有メモリ (DSM) システムの実装法を提案している．*PPRAM* ベース・システムは単一かつグローバルな物理アドレス空間を提供するものの，キャッシュ・コヒーレンスはハードウエア・レベルでは保証しない NCC-NUMA である．提案しているソフトウエア DSM「*PPRAM-DSM*」は，マッピングはページ単位で行なう一方，データ複製およびコヒーレンスはメモリ・ブロック単位で行なうことで false sharing の問題を解決している．また，MRMW および LRC の採用で，シンプルなハードウエアながら CC-NUMA に匹敵する性能を上げることが期待される．

Distributed Shared Memory System on *PPRAM*-Based System

KAZUAKI MURAKAMI ,[†,††] TAKASHI YOSHII ,[†] SHIGENOBU IWASHITA [†]
and HIROSHI MIYAJIMA[†]

PPRAM-based system is a NCC-NUMA system that consists of standard merged DRAM/logic LSI, or *PPRAM* , chips and provides a global physical address space without cache coherence. This paper proposes a software coherence protocol, on *PPRAM*-based NCC-NUMA systems, which supports MRMW and LRC.

1. はじめに

本稿では，メモリ–マルチプロセッサ一体型 ASSP (Application-Specific Standard Product) である *PPRAM* を構成要素とするコンピュータ・システム上での分散共有メモリ・システムの実装法を検討する．

PPRAM (*Parallel Processing Random Access Memory*) とは，DRAM– プロセッサ混載 LSI ならびに並列処理時代における新しいコンピュータ・システム構成法のための構成要素であり，

- 大容量メモリ：DRAM を始めとして，SRAM, Flash EEPROM, 等を単独あるいは組合せて構成
- マルチプロセッサ：汎用プロセッサ，特殊用途向けプロセッサ，FPGA，等の 0 個以上のプロセッサ/ロジックから構成されるホモジニアスまたはヘテロジニアスなマルチプロセッサ
- 通信：*PPRAM* チップ内のプロセッサ間通信，および，*PPRAM* チップ間通信を標準通信インタフェース/プロトコル『*PPRAM-Link*[1)]』に準拠して制御

を 1 チップに集積した ASSP である．

PPRAM ベース・システムとは，*PPRAM* チップをシステム構成要素とし，それらを 1 個以上 *PPRAM-Link* で相互結合したものである．*PPRAM* ベース・システムは，システム全体として 64 ビット・アドレス長の単一かつグローバルな物理アドレス空間を提供する[1)]．ノード☆識別子 (nodeId) は 64 ビット・アドレスの上位 32 ビット，ノード内オフセットは下位 32 ビットである．これにより，*PPRAM* ベース・システムは最大で 4G 個のノードを含み得，また，各ノードの最大メモリ容量は 4G バイトとなる．

PPRAM ベース・システムは上述の通り，メモリは各 *PPRAM* ノード毎に分散配置されているが，システム全体としてはグローバルな物理アドレス空間を提供する．よって，*PPRAM* ノードを相互結合する *PPRAM-Link* は，リモート・メモリへの read/write トランザクションを提供する．しかしながら，*PPRAM* ノードの中にはキャッシュを備えるものも存在するものの，*PPRAM-Link* はキャッシュ・コヒーレンス・プロトコルは提供しない．すなわち，*PPRAM* ベー

† 九州大学 大学院システム情報科学研究科 情報工学専攻
Department of Computer Science, Kyushu University
ppram@c.csce.kyushu-u.ac.jp
http://kasuga.csce.kyushu-u.ac.jp/~ppram
†† *PPRAM* コンソーシアム設立準備会
PPRAM Consortium: Launch Working Group

☆ *PPRAM* のノードは，0 バイト以上のローカル・メモリ，0 個以上のプロセッサ/ロジック，および，1 個以上のネットワーク・インタフェースから構成される．1 個の *PPRAM* チップは，1 個以上の *PPRAM* ノードから成る．

表1 検出方法

検出者	検出手段 SW Table	PT/TLB	Cache Dir.	Memory Dir.
SW	√			
HW		√	√	√

ス・システムは，そのグローバル物理アドレス空間内でのリモート・メモリ・アクセスは可能なものの，キャッシュ・コヒーレンスがハードウエア・レベルでは保証されない**NCC-NUMA** (*Non Cache Coherent, NonUniform Memory Access*)[2]である．

したがって，プログラミング・インタフェースとしてコヒーレントな共有メモリ・モデルを提供するには，ソフトウエアによる何らかの支援が必要である．

以下，まず2章で分散共有メモリの実現に際しての課題とその解決策の選択肢を整理する．次に，3章で*PPRAM* ベース・システムに適した分散共有メモリ実現方法を提案する．

2. 分散共有メモリ

分散共有メモリ (*DSM: Distributed Shared Memory*) とは，「物理的にメモリが分散配置されたシステム」上で「論理的に実現された共有メモリ」のことである[5]．

DSMを実現する上での主要課題は，次の3つである．

(1) **マッピング**：共有メモリ・アクセスに関して，そのアクセス側 (accesser) と 被アクセス側 (accessee) との間のマッピング (対応関係) を「どの時点で」,「誰が」,「どの単位で」,「どのように」確立/解消するのか?

(2) **データ供給**：マッピングの確立しているアクセス側に対して，データを「どの時点で」,「誰が」,「どの単位で」,「どのように」供給するのか?

(3) **コヒーレンス**：供給されているデータのコヒーレンスを「どの時点で」,「誰が」,「どの単位で」,「どのように」保証するのか?

2.1 どの時点で?

まず「どの時点で?」に関しては，マッピング，データ供給，および，コヒーレンスの3つに共通して以下の選択肢がある．

- 明示的手段：マッピング，データ供給，および，コヒーレンスそれぞれの処理を起動する専用のプリミティブ (命令やシステム・コール) の実行時
- 暗黙的手段：通常のメモリ・アクセス (命令フェッチやLOAD/STORE命令実行) 時

後者の暗黙的手段の場合は，マッピング，データ供給，および，コヒーレンスのいずれかの処理が必要か否かを通常のメモリ・アクセスの中で検出 (lookup) する必要がある．この検出方法としては，

- 誰が検出するのか?
- その検出手段は?

の違いにより，表1に示す選択肢が存在する．

表2 分岐方法

分岐先	分岐手段 分岐命令	割込み	HW状態遷移
SW	√	√	
HW			√

表3 検出-分岐方法

検出方法	分岐方法 分岐命令 (→ SW)	割込み (→ SW)	HW状態遷移 (→ HW)
SW Table(SW)	√		
PT/TLB(HW)		√	
Cache Dir(HW)		√	√
Memory Dir(HW)		√	√

2.2 誰 が?

「誰が?」に関しては，マッピング，データ供給，および，コヒーレンスの3つに共通して以下の選択肢がある．

- ソフトウエア (SW)
- ハードウエア (HW)

前節で暗黙的手段を採った場合，検出結果次第で上記ソフトウエアないしハードウエアに分岐して，所望の処理 (マッピング，データ供給，および，コヒーレンス) 行なうことになる．その分岐方法としては，

- 誰に分岐するのか?
- その分岐手段は?

の違いにより，表2に示す選択肢が存在する．

表3に，「どの時点で?」(表1) と「誰が?」(表2) との間の意味のある組合せを示す．

2.3 どの単位で?

「どの単位で?」に関しては，マッピング，データ供給，および，コヒーレンスの3つに共通して以下の選択肢がある．

- キャッシュ・ライン：最も粒度が小さい．
- メモリ・ブロック：キャッシュ・ラインと仮想記憶ページの間の中間的な粒度．サブページとも呼ぶ．
- 仮想記憶ページ：最も粒度が大きい．

表4に，「どの時点で?」および「誰が?」(表3) と「どの単位で?」との間の意味のある組合せを示す．

2.4 どのように?

2.4.1 マッピング

アクセス側と被アクセス側との間にどういうマッピング (対応関係) が存在するかという情報は，ディレクトリ (directory) として管理される．このディレクトリの構成法に関しては，以下の選択肢が存在する．

(1) 集中配置&管理：ディレクトリをある決まった1個のノードに集中配置し，マッピングもそこで集中管理する．たとえば，Li and Hudak[3]の集中マネージャアルゴリズム，等．

表4　検出-分岐方法と粒度

検出方法	分岐方法	粒度 Cache Line	Memory Block	Virtual Page
SW Table(SW)	分岐命令 (→SW)	任意サイズ可		
PT/TLB(HW)	割込み (→SW)			√
Cache Dir(HW)	割込み (→SW)	√		
	HW 状態遷移 (→HW)	√		
Memory Dir(HW)	割込み (→SW)		√	
	HW 状態遷移 (→HW)		√	

表5　データ供給方法

読出し時	書込み時 移動も複製もしない	移動	複製
移動も複製もしない	CS	—	—
移動	—	SRSW	—
複製	—	MRSW	MRMW

表6　データの供給元と供給先

供給元	供給先 キャッシュ	メモリ
キャッシュ		—
メモリ	CC-NUMA NCC-NUMA	RMS COMA SVM

(2)　分散配置&管理：ディレクトリを各ノードに分散配置し，マッピングもそこで分散管理する．これには，さらに以下の選択肢が存在する．

- 静的分散：ディレクトリの分散配置先が静的に定まっており，かつ，固定的である．たとえば，ディレクトリ法における各種ディレクトリ，あるいは，Li and Hudak[3]の固定分散マネージャ・アルゴリズム，等．
- 動的分散：ディレクトリの分散配置先が動的に決まり，かつ，流動的である．たとえば，Li and Hudak[3]のブロードキャスト分散マネージャ・アルゴリズム，動的分散マネージャ・アルゴリズム，等．

2.4.2　データ供給

データの供給方法としては，次の2方法がある．

- 移動 (migration)
- 複製 (replication)

データへの読出し/書込みの際にどのようなデータ供給を行なうかに関して，次の選択肢が存在する (表5参照)[5]．

(1)　CS (Central Server) 法
(2)　SRSW (Single-Reader/Single-Writer) 法 (migration アルゴリズム)
(3)　MRSW (Multiple-Readers/Single-Writer) 法 (read-replication アルゴリズム)
(4)　MRMW (Multiple-Readers/Multiple-Writers) 法 (full-replication アルゴリズム)

また，データの供給元と供給先に関しては，表6に示す選択肢が存在する．表中の略称は，以下の DSM 実現方式を表す．

- CC-NUMA (Cache Coherent, NonUniform Memory Access)
- NCC-NUMA (Non Cache Coherent, NonUniform Memory Access)
- RMS (Reflective Memory System)
- COMA (Cache Only Memory Architecture)
- SVM (Shared Virtual Memory)

2.4.3　コヒーレンス

ホーム☆という概念が存在する場合 (前出の CC-NUMA および NCC-NUMA，等) は，あるデータのコピーに対する書込みをどの時点でホーム上のデータに反映するか (ライト・ポリシ) について，次の2つの選択肢がある．

- ライト・スルー (write through)
- ライト・バック (write back)

さらに，MRSW 法および MRMW 法では，同一データのコピーが複数存在し得る．いずれかのコピーに対して書込みが行なわれた場合，他のコピーとの間で内容の不一致が生じる．この不一致を解消する方法 (コヒーレンス・ポリシ) として，次の2つがある．

- 書込み無効化 (write-invalidation)
- 書込み更新 (write-update)

表7に，データ供給方法 (表5) とコヒーレンス・プロトコルとの間の意味のある組合せを示す．

ここで，上記の「不一致解消」をいつの時点までに行なわなければならないかに関して，システムが採用しているメモリ・コンシステンシ・モデル (memory consistency model) と連動して，以下に示すいくつかの選択肢が存在する[5]．

(1)　SC (Sequential Consistensy)：あるプロセッサからの書込み無効化/更新は，当該プロセッサのそれ以前のすべての「メモリ・アクセス」が完了☆☆ するまでは完了することが出来ないと

☆ いわば，あるデータのコピーすなわち「現住所」が複数存在したとしても，そのデータにとっては必ず唯1個しか存在しない「本籍地」．

☆☆ プロセッサ P_1 がロケーション L に対して行ったメモリ・アクセスは，以下の条件を満足した時点で「プロセッサ P_2 に関して完

表7　データ供給方法とコヒーレンス・プロトコル

データ供給方法	コヒーレンス・プロトコル	
	書込み無効化	書込み更新
CS	不要	
SRSW	不要	
MRSW	√	—
MRMW	√	√

同時に，当該プロセッサのそれ以降のいずれの「メモリ・アクセス」も当該書込み無効化/更新が完了するまで完了することは出来ない．

(2) WC (Weak Consistensy)：あるプロセッサからの書込み無効化/更新は，当該プロセッサのそれ以前のすべての「同期アクセス」が完了するまでは完了することが出来ないと同時に，当該プロセッサのそれ以降のいずれの「同期アクセス」も当該書込み無効化/更新が完了するまで完了することは出来ない．

(3) RC (Release Consistensy)：あるプロセッサからの書込み無効化/更新は，当該プロセッサのそれ以前のすべての「ロック獲得アクセス (acquire)」が完了するまでは完了することが出来ないと同時に，当該プロセッサのそれ以降のいずれの「ロック解放アクセス (release)」も当該書込み無効化/更新が完了するまで完了することは出来ない (Eager RC)．あるいは，同一ロック変数に対するそれ以降のいずれの「ロック獲得アクセス (acquire)」も当該書込み無効化/更新が完了するまで完了することは出来ない (Lazy RC)．

3. *PPRAM-DSM* の提案

1章で述べた通り，*PPRAM* ベース・システムは *PPRAM-Link* 仕様に基づいて単一かつグローバルな物理アドレス空間を提供するが，キャッシュ・コヒーレンスはハードウエア・レベルでは保証しない．これは，NCC-NUMA と呼ばれるアーキテクチャに属する．同類のシステムとして，BBC TC2000 や Cray Research T3D，Princeton Shrimp，等がある．

NCC-NUMA 上でコヒーレントな共有メモリをソフトウエアにより提供する方法としては，Petersen and Li[4]，および，その拡張である Kontothanassis and Scott[2]の方法がある．本稿で検討する *PPRAM* ベース・システム向きのソフトウエア DSM システム (以降，*PPRAM-DSM* と表記する) は，これらの方法を基に変更を施したものである．*PPRAM-DSM* はまた，Wisconsin Wind Tunnel プロジェクトにおける Fine-Grain Access Control を用いた DSM 実現法の Blizzard[6]の特長も採り入れている．

前章で整理した項目に関して，*PPRAM-DSM* および関連する2手法とを比較した結果を表8～10に示す．

以下，*PPRAM-DSM* の概略を示す．

3.1 前提とするハードウエア

PPRAM-DSM は，メモリ・アクセスを行なうことの出来る *PPRAM* ノードが下記のハードウエアを備えていることを前提とする．

- PT/TLB：仮想記憶機構は必須で，各ノードはそれ自身のページ・テーブル (PT)，および，ソフトウエア制御可能なアドレス変換バッファ (TLB) を有する．各ページ・テーブル・エントリ (PTE) は少なくとも Invalid/ Local/ Remote の3属性を保持する．Invalid あるいは Remote な仮想ページに対する通常のメモリ・アクセス (命令フェッチおよび LOAD/STORE 命令) は，ページ・フォルトを起こす．
- メモリ・ディレクトリ：メモリ・ブロック (サイズはキャッシュ・ライン・サイズ以上，ページ・サイズ以下の任意の大きさ) 毎に，少なくとも Invalid/ Read-Only/ Read-Write の3属性を保持するディレクトリを設ける．Invalid なメモリ・ブロックに対する通常のメモリ・アクセス，および，Read-Only のメモリ・ブロックに対する通常の STORE 命令によるストア・アクセスは，メモリ・ブロック・フォルトを起こす．さらに，ワード単位に dirty ビットを設ける．
- リモート・メモリに対する読出し/書込み操作，および，不可分操作 (fetch& add，等)

さらに，各ノードはキャッシュを備えていてもよい．この場合，キャッシュ・ディレクトリの各エントリは，そのキャッシュ・ラインが属するメモリ・ブロックの Read-Only/ Read-Write の2属性を保持する必要がある．また，ワード単位に dirty ビットを備えるとともに，ソフトウエア制御で任意のラインをローカル・メモリに書き戻せる必要がある．

3.2 アルゴリズム

3.2.1 初期設定

(1) 共有仮想アドレス空間が生成される時，*PPRAM-DSM* は各ノードにページ・テーブル (PT) を1個づつ作成する．そのエントリ (PTE) の初期値は，ある1個の仮想ページに着目した場合，同一の内容 (すなわち，同じ物理ページ番号および同じページ属性) となる．ただし，Local/Remote 属性に関しては，次のように異なる．

- Local：ある仮想ページに対応する物理ページが最初に置かれたノード☆上に作成され

了した」と言う．すなわち，当該アクセスがロードの場合，プロセッサ P_2 が L に対してストア・アクセスを行っても P_1 に返る値に影響しない．また，当該アクセスがストアの場合，プロセッサ P_2 が L に対してロード・アクセスを行ったら，当該ストア値を P_2 に返す．さらに，メモリ・アクセスは，それが全プロセッサに関して完了した時点で「完了した」と言う．

☆ 当該ノードは，当該仮想ページのホーム・ノードとなる．

表8 マッピング

DSM 実現法	検出方法	分岐方法	粒度	配置& 管理
全者とも	PT/TLB(HW)	割込み (→SW)	ページ	静的分散

表9 データ供給

DSM 実現法	検出方法	分岐方法	粒度	供給方法	供給元→供給先
PPRAM-DSM	Memory Dir(HW)	割込み (→SW)	メモリ・ブロック	MRMW	メモリ→メモリ
Petersen& Li	Cache Dir(HW)	HW 状態遷移 (→HW)	キャッシュ・ライン	MRMW	メモリ→キャッシュ
Blizzard-ES	Memory Dir(HW)	割込み (→SW)	メモリ・ブロック	MRSW	メモリ→メモリ
	SW Table(SW)	分岐命令 (→SW)			

表10 コヒーレンス

DSM 実現法	検出方法	分岐方法	粒度	ライト・ポリシ	コヒーレンス・ポリシ	MCM†
PPRAM-DSM	acquire/release‡	分岐命令 (→SW)	メモリ・ブロック	ライト・バック	書込み無効化	LRC
	Memory Dir(HW)	割込み (→SW)				
Petersen& Li	acquire/release‡	分岐命令 (→SW)	ページ	ライト・スルー	書込み無効化	LRC
	PT/TLB(HW)	割込み (→SW)				
Blizzard-ES	Memory Dir(HW)	割込み (→SW)	メモリ・ブロック	—	書込み無効化	SC
	SW Table(SW)	分岐命令 (→SW)				

†: メモリ・コンシステンシ・モデル
‡: 明示的手段 (2.1節参照)

た当該仮想ページに対応する PTE
- Remote：それ以外のノード上に作成された当該仮想ページに対応する PTE

(2) また，物理ページが割り当てられたメモリ・ブロックの属性をすべて Read-Write に設定する．

3.2.2 マッピング (その 1)

(1) あるノードがある Remote な仮想ページに対して通常のメモリ・アクセスを行なうと，ページ・フォルトが生じ *PPRAM-DSM* が起動される．

(2) *PPRAM-DSM* は，ローカル・メモリ上に新しい物理ページを 1 個確保し，それを当該仮想ページにマッピングする．すなわち，当該ノードの PT 内の当該仮想ページに対応する PTE は，次のように内容が変更される．
- 物理ページ番号：ローカル・メモリ上に新たに確保した物理ページのページ番号
- Local/Remote 属性：Local

なお，元々の，すなわち，ホーム・ノード上の物理ページ番号☆は，当該ノード上でローカルに管理する「Home リスト」に登録される．

(3) また，新たに物理ページが割り当てられたローカル・メモリ・ブロックの属性をすべて Invalid に設定する．

(4) ページ・フォルトを起こしたプログラムの実行を再開する．

3.2.3 データ供給，および，マッピング (その 2)

(1) あるノードがある Local な仮想ページ中のある Invalid なメモリ・ブロックに対して通常のメモリ・アクセスを行なうと，メモリ・ブロック・フォルトが生じ *PPRAM-DSM* が起動される．

(2) *PPRAM-DSM* は，当該ノード上の Home リストに登録されているホーム物理ページ番号を用いて，当該ホーム物理ページからローカル物理ページへ所望のメモリ・ブロックを複製 (replication) して持ってくる．

(3) 同時に，ホーム・ノード上で管理されている当該メモリ・ブロックのディレクトリ情報を以下のように更新する．
- (a) ノード識別子および Read/Write の区別をコピー・セットに登録する．
- (b) Read-Replication の場合：Read 登録の結果，コピー・セットの登録数が 2 以上となり，そのうち少なくとも 1 つの Write 登録が存在し，かつ，当該メモリ・ブロックがまだ Weak 状態でなければ，当該メモリ・ブロックを Weak 状態に遷移させる．つまり，コピー・セットに登録されているノードそれぞれの「Weak リスト」に，当該メモリ・ブロックを登録する．
- (c) Write-Replication の場合：Write 登録の結果，コピー・セットの登録数が 2 以上となり，かつ，当該メモリ・ブロックがまだ Weak 状態でなければ，これを Weak 状態に遷移させる．

(4) 当該ローカル・メモリ・ブロックの属性を Read-Replication の場合 Read-Only に，Write-Replication の場合 Read-Write に設定する．

(5) メモリ・ブロック・フォルトを起こしたプログラムの実行を再開する．

☆ 以下，ホーム物理ページ番号と呼ぶ．

3.2.4 コヒーレンス

3.2.4.1 ストア・アクセス

(1) あるノードがあるLocalな仮想ページ中のあるRead-Onlyのメモリ・ブロックに対して通常のストア・アクセスを行なうと，メモリ・ブロック・フォルトが生じ*PPRAM-DSM*が起動される．

(2) *PPRAM-DSM*は，当該ノード上でローカルに管理している「Storeリスト」に当該メモリ・ブロックを登録する．

(3) 当該ローカル・メモリ・ブロックの属性をRead-OnlyからRead-Writeへと変更する．

(4) メモリ・ブロック・フォルトを起こしたプログラムの実行を再開する．

3.2.4.2 acquire

(1) あるノードでaquireを実行すると*PPRAM-DSM*が起動される．

(2) *PPRAM-DSM*は，当該ノード上でローカルに管理しているWeakリストに登録されているメモリ・ブロックそれぞれに対して，以下の操作を施す．

(a) キャッシュが存在し，当該メモリ・ブロックがキャッシングされている場合，当該メモリ・ブロック中のdirtyビットがセットされているワードをローカル・メモリに書き戻す．

(b) 当該メモリ・ブロック中のdirtyビットがセットされているワードをホーム・ノードに書き戻す(ライト・バック・ポリシ)．

(c) そのホーム・ノード上の当該メモリ・ブロックのディレクトリ情報を次のように更新する．すなわち，そのコピー・セットから当該ノードを削除する．

(d) 当該ノードのローカル・メモリの当該メモリ・ブロックの属性をInvalidに変更する．

(3) aquireを実行したプログラムの実行を再開する．

3.2.4.3 release

(1) あるノードでreleaseを実行すると*PPRAM-DSM*が起動される．

(2) *PPRAM-DSM*は，当該ノード上でローカルに管理しているStoreリストに登録されているメモリ・ブロックそれぞれに対して，以下の操作を施す．

(a) 当該メモリ・ブロックのホーム・ノード上で管理されているディレクトリ情報を次のように更新する．すなわち，それまでのRead登録からWrite登録へと変更する．

(b) この時，コピー・セットの登録数が2以上であり，かつ，当該メモリ・ブロックがまだWeak状態でなければ，これをWeak状態に遷移させる．

(3) releaseを実行したプログラムの実行を再開する．

4. おわりに

以上，*PPRAM*ベース・システム上での分散共有メモリ(DSM)システムの実現方法について検討した．提案したソフトウエアDSM「*PPRAM-DSM*」は，マッピングはページ単位で行なう一方，データ複製およびコヒーレンスはメモリ・ブロック単位で行なうことでfalse sharingの問題を解決している．また，MRMWおよびLRCの採用で，シンプルなハードウエアながらCC-NUMAに匹敵する性能を上げることが期待される．

今後は，*PPRAM-DSM*の$PPRAM^{\pi}_{m,f}$シミュレータ上での実装を行ない，その機能ならびに性能を評価する予定である．

謝辞

日頃から御討論頂く，九州大学 大学院システム情報科学研究科 安浦寛人教授，岩井原瑞穂 助教授，*PPRAM*プロジェクト・メンバ，安浦・村上・岩井原研究室の諸氏，ならびに，*PPRAM*コンソーシアム設立準備会の会員諸氏に感謝致します．

参考文献

1) 村上和彰, 岩下茂信, 宮嶋浩志, "メモリ-マルチプロセッサ一体型ASSP「*PPRAM*」用標準通信インタフェース『*PPRAM-Link* Standard』Draft 0.0の概要," 情処研報, ARC-119-27, 1996年8月.

2) Kontothanassis, L. I. and Scott, M. L., "High Performance Software Coherence for Current and Future Architectures," *J. Parallel & Distributed Comput.*, vol.29, no.2, pp.179–195, Sept. 1995.

3) Li, K. and Hudak, P., "Memory Coherence in Shared Virtual Memory Systems," *ACM Trans. Comput. Sys.*, vol.7, no.4, pp.321–359, Nov. 1989.

4) Petersen, K. and Li, K., "Multiprocessor Cache Coherence Based on Virtual Memory Support," *J. Parallel & Distributed Comput.*, vol.29, no.2, pp.158–178, Sept. 1995.

5) Protić, J., Tomašević, M., and Milutinović, V., "Distributed Shared Memory: Concepts and Systems," *IEEE Parallel & Distributed Tech.*, vol.4, no.2, pp.63–79, Summer 1996.

6) Schoinas, I., Falsafi, B., Lebeck, A. R., Reinhardt,S.K., Larus,J.R., and Wood,D.A., "Fine-Grain Access Control for Distributed Shared Memory," *Proc. ACM 6th ASPLOS*, pp.297–306, Oct. 1994.