計算機アーキテクチャ　128－16
ハイパフォーマンス　70－16
コンピューティング
（１９９８．３．６）

# マルチグレインDSMをサポートするWSクラスタ JUMP-1/3

中條拓伯[†]　小野　航[††]　市川明弘[†]
安生健一朗[††]　天野英晴[††]　金田悠紀夫[†]

ワークステーション(WS)クラスタにおいて分散共有メモリ(DSM)の実現がさまざまな研究が行なわれている．しかしながら，I/Oバス上にキャッシュメモリをおいた場合，WS本体のMPUの一次キャッシュまで有効とはならずヒットした場合でも，高速メモリアクセスは困難であった．

一方，ソフトウエア制御によるDSMでは，ページ単位でコヒーレンス制御を行なわねばならず，フォールスシェアリングや，転送時間の増大などさまざまな問題を生ずる．そこで我々は，従来のI/Oバス外のハードウエア制御のDSMに加え，ソフトウエア制御のDSMを組み込むことで，ラインおよびページサイズの共有メモリをサポートすることにより，効率の良いDSMシステムの実現を試みた．本稿では，その実現例であるJUMP-1/3-SLについて述べ，マルチグレインの有効性ついて考察する．

# Multi-grained DSM systen on WS Clusters: JUMP-1/3

Hironori Nakajo,[†] Wataru Ono,[††] Akihiro Ichikawa,[†]
Ken-ichiro Anjo,[††] Hideharu Amano[††] and Yukio Kaneda[†]

There are a number of researches on implementations of distributed shared-memory systems on workstation clusters. However, in the case of being equipped with cache memory on an I/O bus, an executed program cannot utilize primary cache in the MPU of WS, so it is difficult to achieve high performance even if the access hits cache. On the other hand, in software controlled cache, coherence must be controlled in the unit of page, which causes problems of false sharing and growth of transfer delay in the network. Therefore we couple software controlled DSM into hardware controlled one, thus the system is able to support multigrained unit of coherence: a cache line and a page. In this paper, we describe an example of multigrained DSM system, and current performance of hardware controlled DSM in a workstation cluster system JUMP-1/3-SL.

## 1. はじめに

ワークステーションクラスタ(WSクラスタ)は，複数のWSをLANや高速結合網によって接続し，ノード間においてプロセス間通信を行なうことにより並列処理を行なうシステムである[1)]．従来の並列計算機と異なり，価格に応じたスケールアップが容易で手軽に導入することができることからいくつかのシステムが商用化されている．しかしながら，現状のWSクラスタの多くは共有メモリを持たず，並列プログラムの開発，移植性の点で問題があり，そのためWSクラスタ上で共有メモリを実現するためのさまざまな試みがなされている．

† 神戸大学工学部情報知能工学科
Department of Computer and Systems Engineering, Faculty of Engineering, Kobe University
†† 慶應義塾大学理工学部情報工学科
Department of Computer Science, Keio University

現在注目されているDSMシステムとして，以下のものが挙げられる．

- **Shasta**
  このシステムはAlpha workstationを用い，Alphaのプロセッサ能力を最大限に引き出すことを念頭に設計されている[2)]．特徴としてコンパイラによりTag checkの高速化を図り，様々なサイズでのコヒーレンス制御を行うことが出来る．高速なタグチェック機構の一つとしてTag tableとキャッシュメモリのアドレスを工夫することによってシフトによってタグテーブルのアドレスを計算できるようにすることがあげられる．また，コンパイル時に扱うデータサイズから最適なコヒーレンスサイズを決定することによって不要な通信を可能な限り排除する．しかし，このシステムにおいてはコヒーレンスサイズはハードウエアに依存しており，ソフトウエアによって完全に自由に選択できるものではない．

- **CRL**
  このシステムは，アプリケーションが必要とするサイズを自由に選択することのできる通信ライブラリを提案している．CRL(C Region Library)は，プログラム中に共有空間へのアクセスであることを明記することによって，必要な領域だけを共有するシステムを提供している[4]．領域を明記することにより，同期を取る際にも利点がある．通常のシステムにおいては同期は共有メモリ全体に対して行わないといけないが，領域を定義することにより，必要な範囲を関係のあるプロセッサだけで同期を取れば良いために通信量，通信時間ともに低減する効果がある．このライブラリはC 言語のもので，プラットフォーム依存がなく，ポータビリティに優れている．現在の商用並列計算機が message passing で占められているのは，PVM(Parallel Virtual Machine),MPI(Message Passing Interface) 等の通信ライブラリが充実していたことにある．そのため，DSM 用の通信ライブラリの整備は必須である．
- **Multigrain Shared Memory System(MGS)**
  このシステムでは，今までのものと異なったアプローチを取っている．近年，小規模の SMP は目覚ましい進歩をとげ，様々な商用機が登場している．SMP 内部ではプロセッサは密に結合されており共有メモリアクセスハードウエアのサポートもあり，は高速に行うことができる．そこで，これらの SMP を複数台結合し，メモリアクセスの局所性を活用することによって，より高速なシステムの構築を提案している．

DSM において，キャッシュミスを検知するために，UNIX のセグメンテーションフォールトをトラップする関係上，メモリの管理はページ単位で行われることが多かった．しかしながら，ページのような大きな単位での管理を行うとフォールスシェアリングが発生しやすく，小量のデータにアクセスする際にもレイテンシが大きくなるという欠点を持っていた．一方，キャッシュのライン単位で管理する場合，フォールスシェアリングは発生しにくくなるが，大量のデータ転送の際に通信オーバヘッドが大きくなり，また管理機構も複雑となってしまう．

この解消のために様々な粒度でのコンシステンシを維持する方法が検討されるようになった．その中にはハードウエアのメモリアクセスを念頭において開発されるものと，純粋にソフトウェアの指定したサイズでコンシステンシを保つものがある．

本稿では，我々がこれまでに開発してきた JUMP-1/3-SL において実装を進めているハードウエア制御の DSM とソフト制御の DSM を統合することにより，複数の粒度を持つ DSM の実現方式について述べる．

続く章において，これまで我々が取ってきたアプローチと，それらの問題点をまとめ，次に，その問題点の解決策としてマルチグレインの DSM をサポートする JUMP-1/3-SL のアーキテクチャについて述べる．さらに，JUMP-1/3-SL における現状の性能予測について説明し，最後に今後の方向性について述べる．

## 2. これまでのアプローチ

### 2.1 ソフトウェア制御 DSM

これまでに，我々はWS クラスタ上にソフトウエア制御による DSM を実装し，評価を行なってきた[5]．そこでは，既存のハードウエアに新たなハードウエアを付加することなく，また，システムを構成するソフトウエアをユーザレベルで記述することによって，高い移植性を確保している．

また，キャッシュミスは UNIX のセグメンテーションフォルト時に発生するシグナル (SIGSEGV) をトラップすることで自動的に検知でき，キャッシュにヒットした場合は DSM システムの処理が介在しないことからローカルメモリへのアクセスと変わらないアクセス性能が得られている．

しかしながら，通信を行なうために必要な OS に対するシステムコールによるオーバヘッドや，ノード間通信の媒体に用いている Ethernet の通信バンド幅の限界，バス型のネットワークトポロジに起因するメッセージの衝突などの要因によりキャッシュミス時のミスペナルティが大きく，十分な性能を発揮することが困難であった．そのため，DSM システムの性能を向上させるにはノード間の通信の高速化をはかり，キャッシュミス時のミスペナルティを削減することが必要となる．

### 2.2 ネットワークインタフェースの高速化

我々は前述の通信の問題点を打破するために，ノード間通信にネットワークルータを搭載した高速シリアルリンク STAFF-Link(Serial Transparent Asynchronous First-in First-out Link)[6] を用い，ノード間を point-to-point で接続することで通信の衝突を回避し，ミスペナルティの削減を図るシステムを実現した[7]．

このシステムでは，ルータにプログラマブルプロセッサを搭載することで DSM の制御の一部をこのプロセッサに処理させ，要素プロセッサにかかるシステムの処理の負荷を軽減させることができる．しかしながら，実験結果からヒット時の高速化は図れたものの，ミスペナルティが依然として大きく，十分な性能向上を得ることができなかった[7]．

その原因としては，管理の単位がページ単位であり，外部ノードからのページ要求が生じる度に 4KB のサイズの転送が生じ，また I/O バスのアクセス遅延に起因する性能低化が生じたものと考えられる．

### 2.3 DSM WS クラスタ JUMP-1/3 と JUMP-1/3-SL

我々は，簡単な付加ハードウエアにより，SUN WS クラスタ上に分散共有メモリを実現するシステム JUMP-1/3(SUN) を開発してきた[8)9)]．

JUMP-1/3 では，SUN WS の SBus に DPM (DSM Protocol Management) ボードを接続することにより，キャッシュ機能を持つ分散共有メモリ環境を実現する．DPM ボードは，一般の SVM と同様にボード上のメモリをページ単位で管理し，その一部を他のノードの DPM ボード上のメモリのキャッシュ領域として利用する．その一方で，キャッシュライン単位に用意されたタグを用いてアクセスされたラインが有効であるかを高速に判別する機構を持ち，WS 間のデータ転送はライン単位で行なう．

WS 間は，JUMP-1 用に開発された RDT ルータチップ[10)]を搭載した RDT ボードを DPM ボードの後部に接続し，高速大容量転送を行なうことができる．しかし，RDT ボードはオーバスペックであり，Sequential Consistency による方法はライトミスのペナルティで性能を大きく低下させるという問題点が判明した．

そこで，この結果に基づき，今まで開発してきた JUMP-1/3 において，RDT ボードを用いず，DPM ボード上に搭載した STAFF-Link により結合網を構成する．このことにより，容易かつ低コストで WS クラスタを構成することができる．このシステムを JUMP-1/3-SL と呼ぶ[11)]．

しかしながら，従来の JUMP-1/3-SL では，ハードウェアタグにより，粒度の小さな共有データの転送には有効であるが，DSM として各ノードごとに 1MB の領域しか確保できず，大規模なアプリケーションへの対応が困難であった．

## 3. JUMP-1/3-SL におけるマルチグレインのサポート

### 3.1 概　　要

これまでに得られた知見をもとに，問題点を考慮した結果，我々は前述のソフトウエア制御 DSM を JUMP-1/3-SL に組み込み，共有領域を拡大化を図ることにした．これより，配列などの粒度の大きい共有データはページサイズでソフトウエアにより制御し，同期データなどの粒度の小さなデータに対してはハードウェアタグにより制御してラインサイズで転送を行なうマルチグレインをサポートすることとした．このハードウェアタグにより共有される DSM を HardDSM と呼び，ソフトウエアにより制御される SoftDSM と区別する．また，HardDSM，SoftDSM に対するキャッシュをそれぞれ HardCache，SoftCache と呼ぶ．

### 3.2 JUMP-1/3-SL のハードウェア構成

JUMP-1/3-SL は，SBus カードサイズの DPM ボード単体に簡単な STP(シールド付きツイストペア) ケーブルによるシリアルリンクを付加するだけでシステムの構成が可能である．DPM ボードは 3 チャンネルの STAFF-Link を持ち，WS クラスタの接続はリング等の簡単な直接網となる．このため，机上で用いられている WS を接続して小規模な WS クラスタを構成することが可能である．JUMP-1/3-SL の DPM ボードのブロック図を図 1に示す．

DPM ボードは DSM 制御用プロセッサ，HardDSM-HardCache 部，SBus インタフェース部，STAFF-Link 制御部から構成される．以下にその構造と機能を述べる．

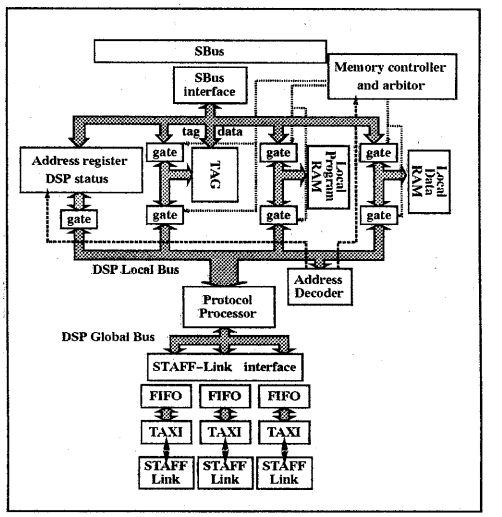


図 1　DPM ボードのブロック図

#### 3.2.1 DSM 制御用プロセッサ (Protocol Processor) ：

DSM 制御用プロセッサとして，汎用 DSP (Digital Signal Processor)(TI TMS320C40) を採用した．TMS320C40 は以下の特長を持つ．

- 2 組のバスを持ち，独立にアクセス可能である．この構成は SBus 側とネットワーク側で頻繁なデータ転送を行なう DSM プロトコル制御に適している．
- 内蔵の DMA 機能により高速転送が可能である．
- 内蔵のタイマにより，ネットワークの転送性能などのパフォーマンスのモニタリングが可能である．

TMS320C40 は，バスサイクル 25MHz，内部周波数 50MHz で動作する．通信制御およびアドレス変換テーブルを格納するため，LPR (Local Program RAM) が 2M バイト設けられている．LPR はアクセス 20nSec の高速 SRAM で実装され，内容はシステムの立ち上

げ時に，WS から SBus を経由して書き込まれる．

#### 3.2.2 Local Data RAM (HardDSM 部および HardCache 部) ：

DPM ボードはデータ領域として LDR(Local Data RAM) を 2M バイト持つ．このうち 1M バイトずつが HardDSM と HardCache 領域として利用される．LDR は，DSM 管理用プロセッサからアクセスされると共に，SBus を介して WS からもアクセスされる．

HardDSM の各ラインに対しては，別に 4 ビットずつタグメモリを設ける．JUMP-1/3-SL では，DSM 制御プロセッサのソフトウェアにより様々なプロトコルが実現可能であり，タグメモリの利用法も様々であるが，3 ビットタグ (shared/exclusive，owner/not owner，valid/not valid) を用いる方法[9]が最も標準的である．

WS が HardDSM にアクセスを行なうとタグは自動的に参照され，Read Hit の場合，アクセスされたデータは DSM 制御プロセッサを介することなしに，WS に送られる．その他の場合は，DSM 制御プロセッサに対して処理を要求する割り込みがかかる．

#### 3.2.3 メモリコントローラ およびアービタ ：

メモリコントローラおよびアービタは，SBus を介して WS 側から LPR，LDR，タグメモリをアクセスするための制御を行なうと共に，タグ参照と Hit 時のデータ転送，DSM 制御プロセッサのアクセスとの調停を行なう．

#### 3.2.4 STAFF-Link ：

STAFF-Link は，並列システムの入出力インタフェースや，狭域，広域ネットワーク用に開発された高速シリアルリンクである．STAFF-Link は，パラレル／シリアル変換と高速データ転送を行なう機能を持つ LSI(TAXI チップ) に，送信用／受信用の 2 つの FIFO，さらに FIFO が溢れないようにハンドシェーク操作を行なう通信コントローラから構成される．

このような構成により，リンクの両端には仮想的に双方向の FIFO が形成され，透過な通信路を構成することができる．ハードウェアの性能を最大限に用いた場合，140Mbps の転送レートを実現することができる．

DPM ボードの外観を図 2に示す．上辺の 3 つの金属性のコネクタが STAFF-Link 用で，WS の背面から外に突き出される．ここに STP ケーブルを接続する．

## 4. マルチグレイン DSM の JUMP-1/3-SL のシステム構成

図 3 にマルチグレインをサポートした JUMP-1/3-SL のシステム構成を示し，図 4 に JUMP-1/3-SL を構成する各ノードの構成を示す．次節でそれぞれのオブジェクトについて説明する

### 4.1 システムのソフトウェア構成

本システムは，DSM-Library，SoftCache 制御プロセス (CCP: Cache Control Process)，DSM-Manager の 3 つのオブジェクトから成り立っている．それぞれの機能を以下に示す．

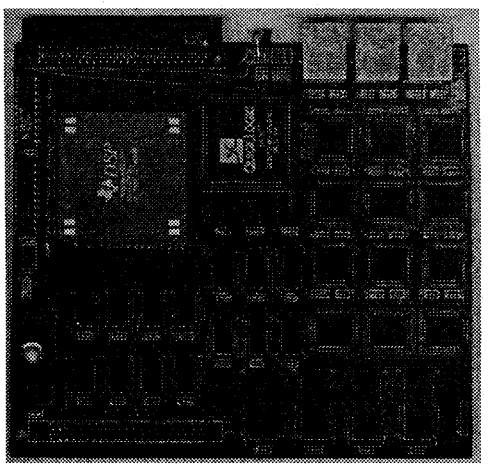

図 2　DPM ボード

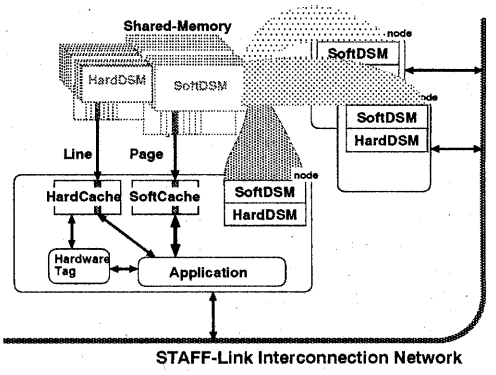


図 3　マルチグレインをサポートした JUMP-1/3-SL のシステム構成

- DSM-Library
  アプリケーションに静的にリンクされ，アプリケーションからの共有メモリの確保や，ロック，アンロック，バリア同期などの要求を処理し，必要に応じて他ノードもしくは自ノードの DSM-Manager に処理を依頼する．
  アプリケーションのキャッシュミスを UNIX のセグメンテーションフォルト発生時のシグナル (SIGSEGV) をトラップすることで検知し，CCP に対してページデータや他ノードの SoftCache の無効化などの要求を発行する．
- SoftCache 制御プロセス (CCP: Cache Control Process)
  DSM-Manager からの SoftCache に対する処理

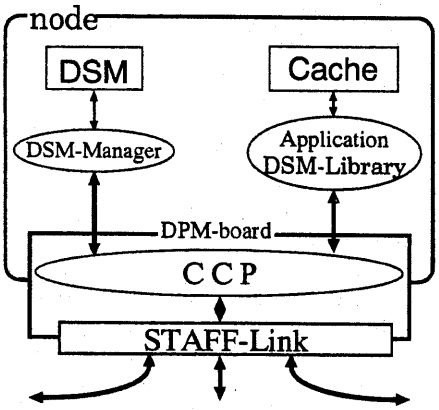


図4 JUMP-1/3-SL のノード構成

を行なうために起動されるスレッドで，SoftCache のページの無効化やライトバック時のページの書き戻しやさらには SoftCache 間のページ転送を行なう．

また，DSM-Library からの SoftDSM に対するページデータや無効化などの要求に基づいて，対象となるノードの DSM-Manager に対して処理要求を発行する．

このとき，DSM-Library や DSM-Manager から受けとった処理要求やデータなどに対してパケットの構築などの前処理も行なう．そのほか，アプリケーションからのバリア同期の要求を処理し，またパケットのルーティング制御も行なう．

- DSM-Manager
  CCP を通してアプリケーションから送られたページデータや他ノードのキャッシュのコピーの無効化要求，さらには管理対象となる SoftDSM のディレクトリの内容を基に，ページデータの返送や無効化要求の発行などを行なう．

### 4.2 巡回型マルチキャスト (Chained Multicast)

バス型のネットワークでシステムを構成している場合，ブロードキャストを行なうことで該当するページデータを保持しているノードに対して無効化要求を発行する．しかしながら，本システムではノード間の接続にシリアルリンクを用いて point-to-point 型のネットワークを用いているためにブロードキャストには膨大な負荷が生じる．そこで，本システムでは無効化要求の発行に巡回型マルチキャスト (Chained Multicast) 方式を採用している[5]．

巡回型マルチキャスト方式とは，無効化要求を受けとるべきノードの一つ一つにパケットを巡回させながらメッセージを伝達していき，最後のメッセージを受けとったノードが要求を出したノードの CCP に対してアクノリッジを返す方式である．

### 4.3 アップデートプロトコルのサポート

これまで，我々はインバリデートプロトコルをベースに開発を進めてきたが，ここでさらに書き込み時に複数のコピーに対して更新を行なうアップデートプロトコルの実装を試みることとした．

アップデートプロトコルでは，リード時は従来と同様のアクセスを行なうが，書き込み時には以下の関数を用いてアクセスを行なう．

```
==============================================
void write_us(void *dest, void *src, int len);
```

- dest: データが書き込まれる共有領域のキャッシュへのポインタ．
- src: 書き込まれるデータが格納されているバッファへのポインタ．
- len: 共有領域へ書き込むサイズ．バイト単位．

アップデートライトにおける共有領域の確保には，ug_malloc を用い，以下のような方法でアクセスを行なう．

```
==============================================
int *update; /* write-update 用の共有領域のキャッシュへのポインタ */
int local;   /* アプリケーションにローカルな変数 */

update = ug_malloc(sizeof(int), "update");
/* int の大きさの共有領域を確保．*/

write_us(update, &local, sizeof(int));
/* ライトアクセス．書き込むデータは変数の中など，アドレスを持つ必要があり，即値で与えることはできない．*/
==============================================
```

アップデート時に用いるメッセージのフォーマットを図5に示す．

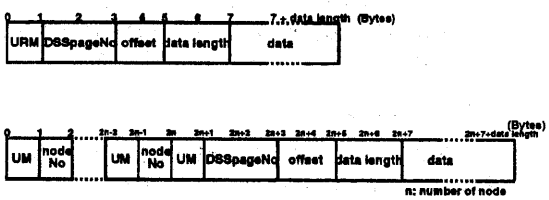


図5 アップデート・パケットフォーマット

ここで，URM，UM はメッセージの識別子を示す．data のフィールドには，memcpy(data, dest, len) で書き込みデータを格納することが可能である．

write_us がコールされた後 CCP は URM を発行し，それを受けとった DSM-Manager は以下に示す処理を行なう．

(1) DSSpageNO からディレクトリを検索し，対象ページのキャッシュを持っているノード番号 (node1, node2, ...) を取得．

(2) 巡回型マルチキャストメッセージとして，

UM(UpdateMessage) を node1 … に示された CCP に送る.

そして，アップデートされるノード上の CCP が UM を受けとった時の処理は以下のようになる.

(1) DSSpageNO, off, len, data を用いて，該当するキャッシュのデータを更新.

(2) メッセージの先頭を切って，次のノードに転送する.

## 5. HardDSM のアクセス性能

SoftDSM の性能に関しては，アップデート以外については，従来のものと変わらない[7]. ここではハードウェアタグを用いた HardDSM の現状でのアクセス性能について述べる.

### 5.1 HardCache へのヒット時

Sparc が DPM ボードにアクセスし，データをアクセスできるまでの時間は，リード／ライトともに SBus のアクセスにおいて 1 クロック分のウエイトが入るため，1 回のアクセスに 350nS かかる.

### 5.2 HardCache へのミス時

ミスした場合は以下のような処理時間を含む.

- DPM の cache が invalid(or Shared) であり，TAG を比較し，DSP に割り込みをかけるまでの時間 :170 nS
- DSP が Home に対して，データの要求をするためのパケットを生成し，FIFO に 4 ワード (16 バイト) 書き込むための時間 :900 nS
- DSP が受信 FIFO をポーリングし，到着を確認してから，そのパケットを解析し，データの入ったパケットの生成，TAG の書き換え，送信用 FIFO に書き込むまでの時間 :1720nS
- remote が受信用 FIFO をポーリングし，到着を確認してから，そのラインを LDR に書き戻し，TAG を valid にするまでの時間 :1800nS

## 6. おわりに

本論文では，我々が開発を進めてきたソフトウェア制御 DSM を JUMP-1/3-SL に実装し，それまでのハードウェアタグとともに，マルチグレインをサポートしたシステムを提案し，現状の HardDSM の性能を示した.

現在 SoftDSM におけるアップデートの実装を進めており，完了次第評価に移る予定である.

今後は JUMP-1/3-SL のデバッグを完了し，マルチグレインによる共有メモリへのアクセスをベンチマークプログラムにより，その性能を評価する予定である.

**謝辞** ルータボードの設計・制作にあたり多大なご支援をいただいた京都大学工学部富田眞治教授に感謝いたします. ならびに，PCB 設計にあたり全面的にご協力いただいた東京大学理学部平木敬教授，松本尚助手に感謝いたします. なお，本研究の一部は並列・分散処理研究推進機構により援助されたものです.

## 参考文献

1) C. C. Douglas, T. G. Mattson and M. H. Shultz: "Parallel Programming Systems for Workstation Clusters", Tech. Rep. TR-975, Yale University Depertment of Computer Science Research, August 1993.

2) D. J. Scales, K. Gharachorloo, C. A. Thekkath: "Shasta:A Low Overhead, Software-Only Approach for Supproting Fine-Grain Shared Memory", Proc. of the 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, pp.174–185 (1996)

3) D. Yeung, J. Kubiatowicz and A. Agrwal: "MGS: A Multigrain Shared Memory System", Proc. of the 23rd Annual Int. Sym. on Computer Architecture, pp.45–56 (1996)

4) L. L. Johnson, M. F. Kaashoek and D. A. Wallach: "CLR: High-Performance All-Software Distributed Shared Memory", Pro. of the 15th Symp. on Operating System Principles, pp.213–228 (1995)

5) 中條, 藏前, 金田, 前川: "ソフトウェア DSM におけるコヒーレント キャッシュシステムの実装と評価," 情報処理学会論文誌, 36 巻 7 号, pp.1719-1728, (1995).

6) 中條, 中野, 松本, 小畑, 松田, 平木, 金田: "分散共有メモリ型超並列計算機 JUMP-1 におけるスケーラブル I/O サブシステムの構成," 情報処理学会論文誌, 37 巻, 7 号, pp. 1429-1439, (1996).

7) 市川, 薬師神, 中條, 金田: "高速シリアルリンクを DSM システムの実装とその評価", 情処研報 *ARC125-9*, pp.49–54 (1997)

8) 安生, 西, 薑, 天野, 吉山, 中條, 工藤: "超並列計算機用結合網 RDT のルーティング制御評価用システム: JUMP-1/3", 信学技報 *CPSY96-52*, pp.39–46 (1996)

9) 安生, 中條, 小野, 工藤, 西, 天野: "分散共有メモリを持つ WS クラスタ: JUMP-1/3", 並列処理シンポジウム *JSPP'97* 論文集, pp.321–328 (1997)

10) H. Nishi, K. Nishimura, K. Anjo, H. Amano and T. Kudoh: "The JUMP-1 Router Chip: Versatile Router for Supporting a Distributed Shared Memory", *Proc. of 15th IPCCC*, pp.158–164 (1996)

11) 小野, 安生, 中條, 工藤, 山本, 西, 天野: *WS クラスタ JUMP-1/3* の実装と評価, 信学技報 CPSY97-36, pp.1–8 (1997).