(2)分散共有メモリシステム

- Distributed Shared Memory: Concepts and Systems
  Jelica Protic, Milo Tomasevic, and Veljko Milutinovic
  IEEE PARALLEL & DISTRIBUTED TECHNOLOGY,
  Vol. 4, No. 2; SUMMMER 1996, pp. 63-79

- Distributed Shared Memory Home Pages
  (TreadMarks開発者の P.Keleherが管理)
  - http://www.cs.umd.edu/~keleher/dsm.html

分散共有メモリ環境の実現

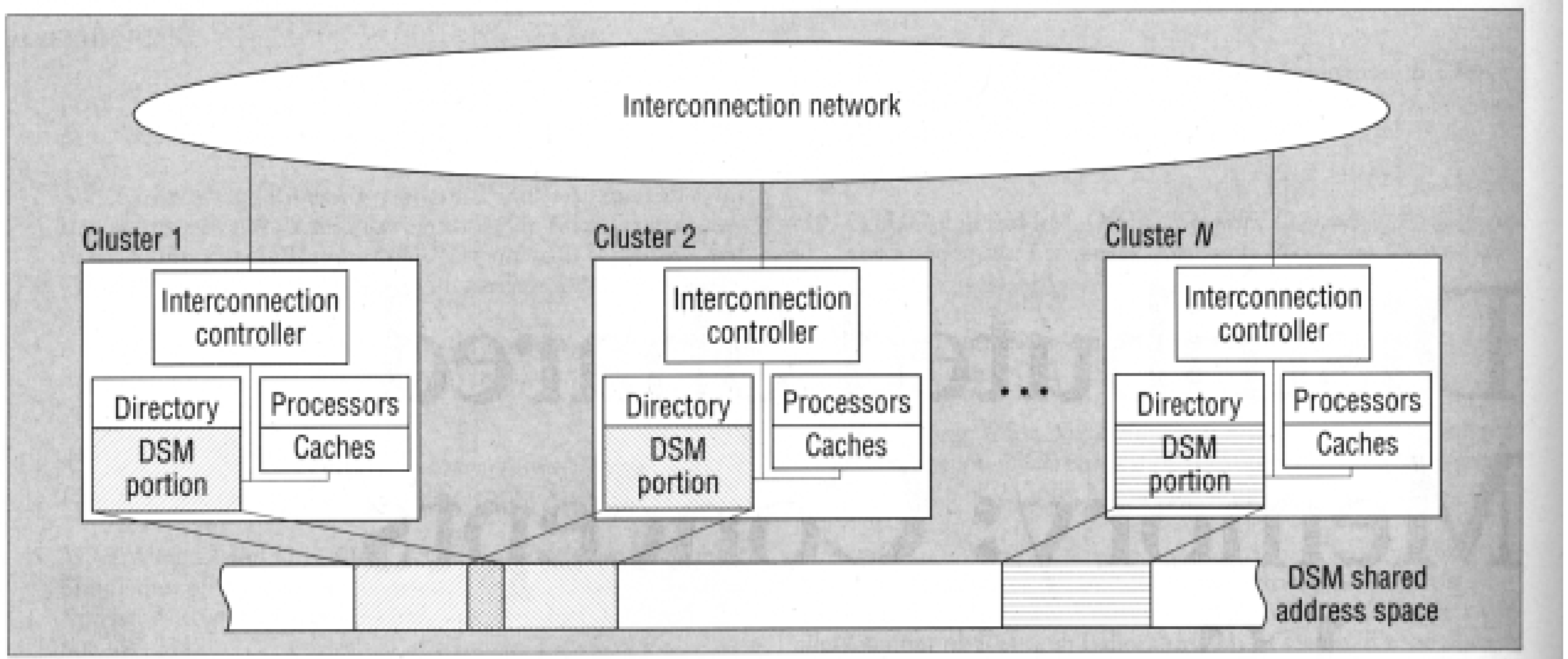
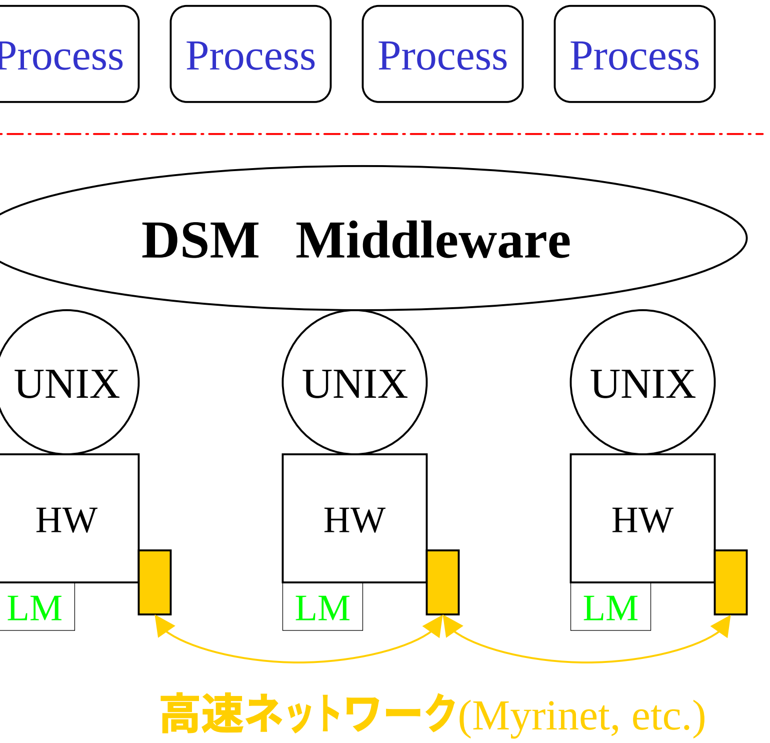


Figure 1. Structure and organization of a DSM system.

分散共有メモリ環境の実現



(Software) Distributed shared memor

provides the programmer with the illusion of a single virtual address space, which is shared among a network of processors that do not share physical memory.
As local memory is updated, the modifications are propagated to the other processors, so that all maintain a consistent view.

[極端な例]

Read Only Data **の** replication **だけを提供**

Classifications of DSM systems

- How the access actually executes?
- Where the access is implemented?
- What the precise meaning of the word consistent is ?

Latency, Granularity, Availability

DSM Algorithms

- Single Reader/Single Writer algorithms
  ⇒機能限定DSM
- Multiple Reader/Single Writer algorithms
  ⇒多くのHW　DSM，無効化型の一貫性制御
- Multiple Reader/Multiple Writer algorithms
  ⇒Page-based SW DSM におけるFalse Sharing対策
  更新型の一貫性制御

Avenues for performance improvement

Directory 構成法と一貫性制御プロトコル
SWオーバヘッドの軽減

DSM mechanismの実装レベル

- Software DSM implementation
- Hardware DSM implementation
- Hybrid DSM implementation

Software DSM

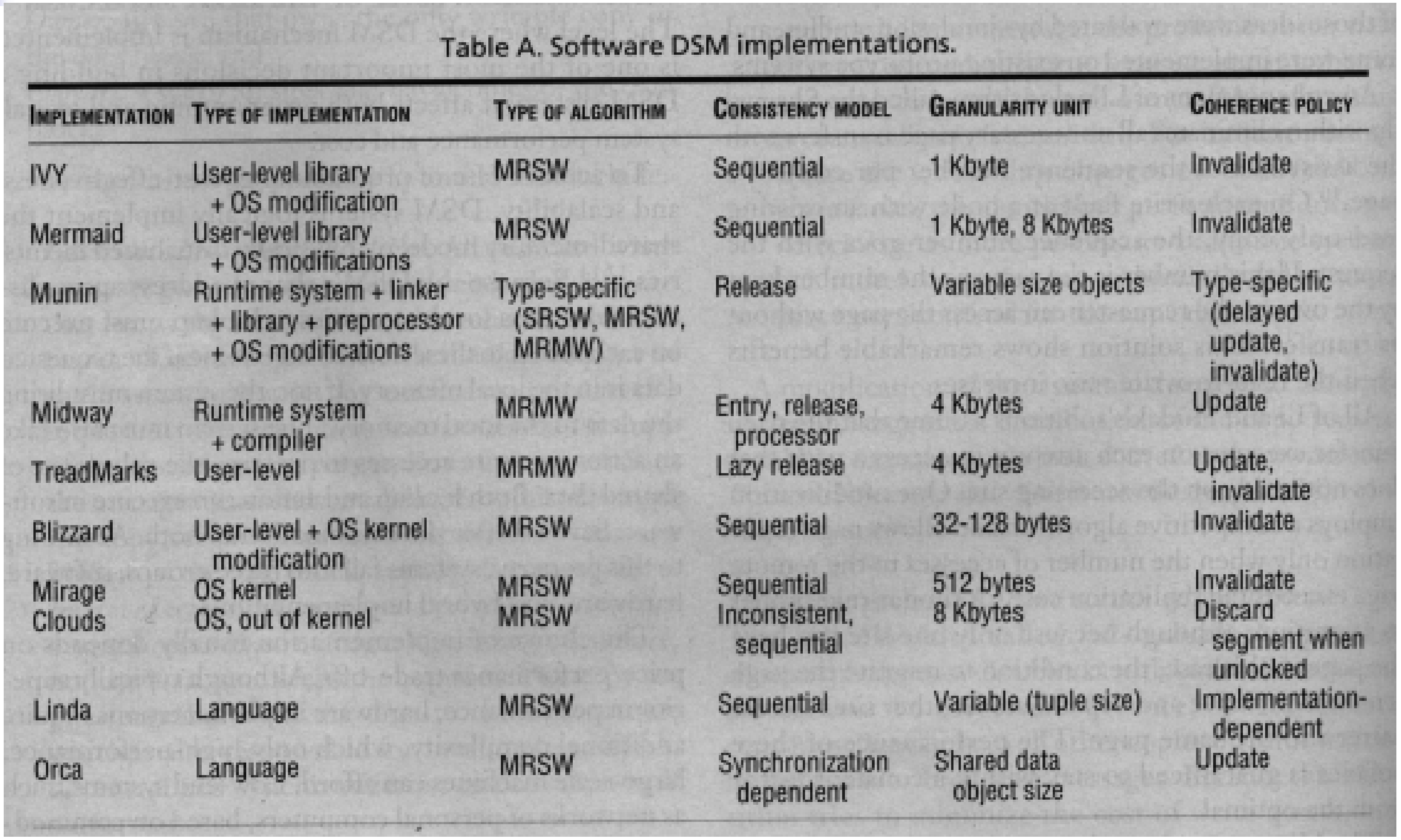


Table A. Software DSM implementations.

Implementation	Type of implementation	Type of algorithm	Consistency model	Granularity unit	Coherence policy
IVY	User-level library + OS modification	MRSW	Sequential	1 Kbyte	Invalidate
Mermaid	User-level library + OS modifications	MRSW	Sequential	1 Kbyte, 8 Kbytes	Invalidate
Munin	Runtime system + linker + library + preprocessor + OS modifications	Type-specific (SRSW, MRSW, MRMW)	Release	Variable size objects	Type-specific (delayed update, invalidate)
Midway	Runtime system + compiler	MRMW	Entry, release, processor	4 Kbytes	Update
TreadMarks	User-level	MRMW	Lazy release	4 Kbytes	Update, invalidate
Blizzard	User-level + OS kernel modification	MRSW	Sequential	32-128 bytes	Invalidate
Mirage	OS kernel	MRSW	Sequential	512 bytes	Invalidate
Clouds	OS, out of kernel	MRSW	Inconsistent, sequential	8 Kbytes	Discard segment when unlocked
Linda	Language	MRSW	Sequential	Variable (tuple size)	Implementation-dependent
Orca	Language	MRSW	Synchronization dependent	Shared data object size	Update

Software DSM implementation

- Write Detection
  - Page-Based vs Object-Based
- Coherence Enforcement

IVY	OS level -----> Shared Virtual Memory
TreadMarks	User level -----> Diff//LRC
Midway	Compiler level-----> Entry Consistency
Shasta	Compiler level-----> Any
Linda	Language level-----> content addressable “Tuple” space

Princeton大学 “IVY”

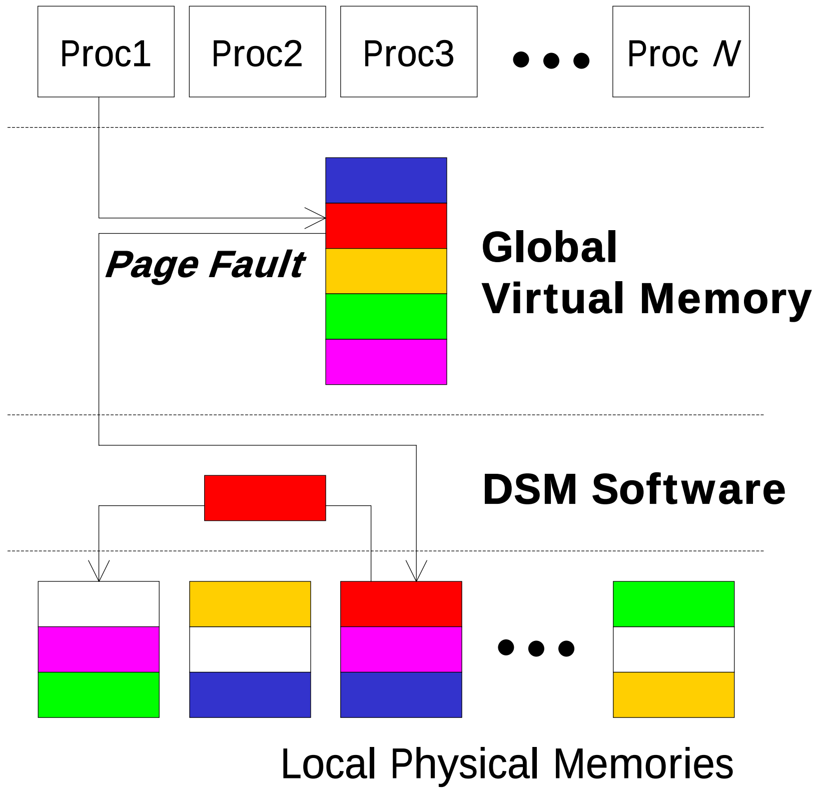
1KB Page based DSM

OS Modification+
User-level Library

MRSW

Sequential

Invalidate

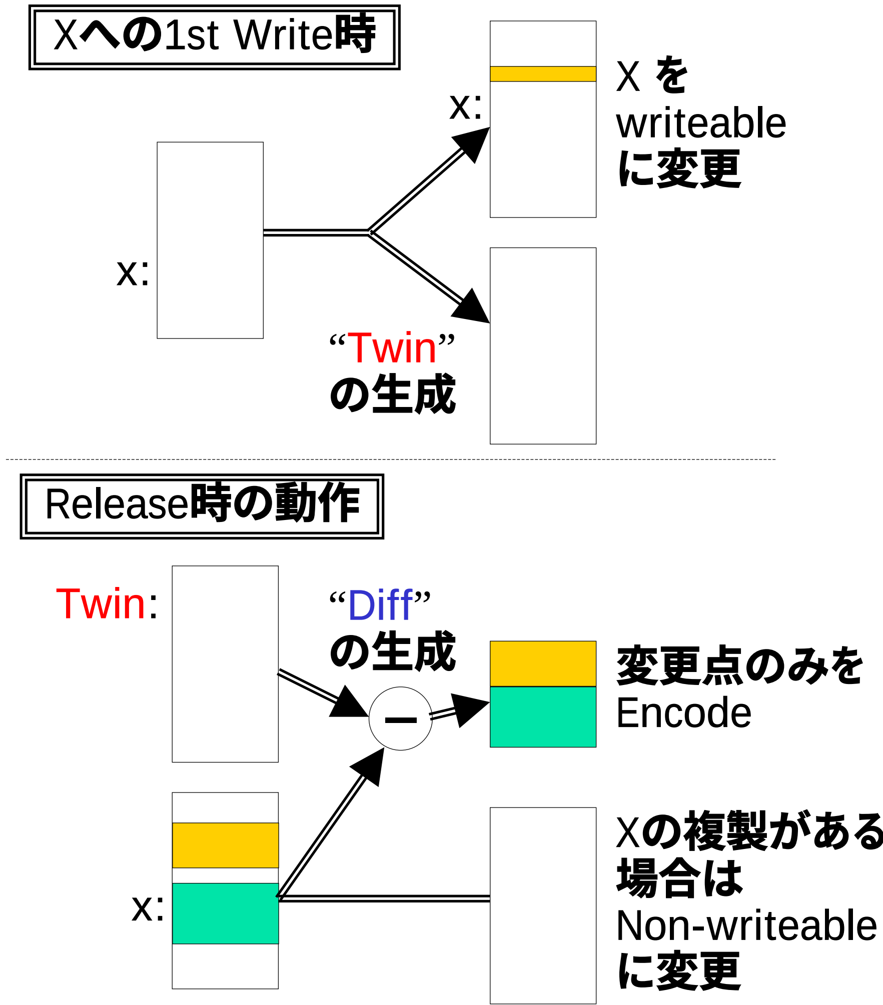


Rice大学 “TreadMarks”

4KB Page based DSM
User-level Library
- TLBでWriteを検出　Twin作成
- Release時に Diff を作成
- Acquire時に Patch をあてる

MRMW
Lazy Release
Update/Invalidate

False Sharing対策
無駄な一貫性制御の排除



Midway

- Entry Consistency

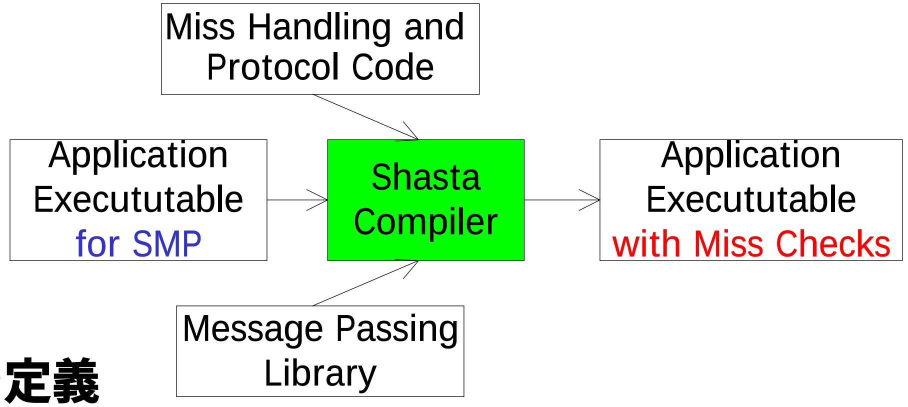
  In Midway, there is an explicit binding of locks to the data that is logically guarded by each lock.

  - As the application acquires a lock for its own synchronization, Midway piggybacks the memory updates on the lock acquisition message. Thus Midway sends no extra messages.
  - Furthermore, the updates are sent only to the acquiring processor and only for the data explicitly guarded by the acquired lock. This serves to batch together updates and minimize the total amount of data transmitted.
- Midway detects updates to shared memory via compiler and runtime support.
- To provide high performance communication, Midway has its own application oriented protocols which reduce message counts, and it utilizes Mach's low-overhead network interfaces to reduce message latency.

DEC　WRL　“Shasta”

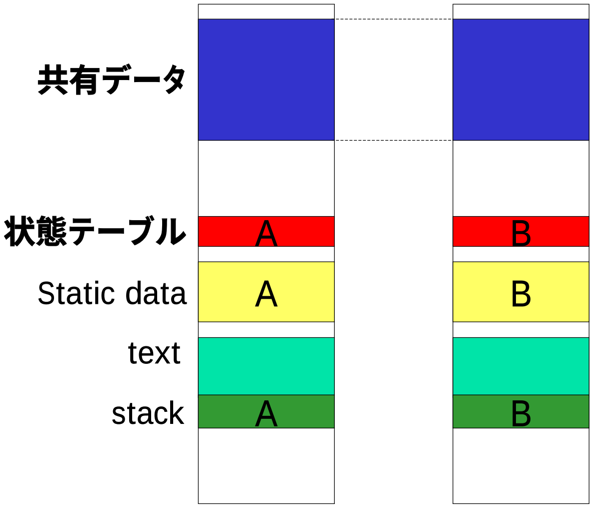
Fine Grain Software DSM

Compiler　Level


Miss Handling and Protocol Code
Application Exectutable for SMP
Shasta Compiler
Application Exectutable with Miss Checks
Message Passing Library

1仮想アドレス空間内に共有データ領域を定義
2キャッシュ上の共有データの状態をSWで管理
3共有データへのアクセスは全てmiss check
　必要に応じてメッセージ通信ライブラリを利用して通信

MRSW
Any Consistency Model
Any Protocol



Shasta のメモリレイアウトの例

Hybrid DSM implementation

- SHRIMP@Princeton Univ.
  - Virtual Memory Mapped I/O
  - Automatic Update Release Consistency(AURC)

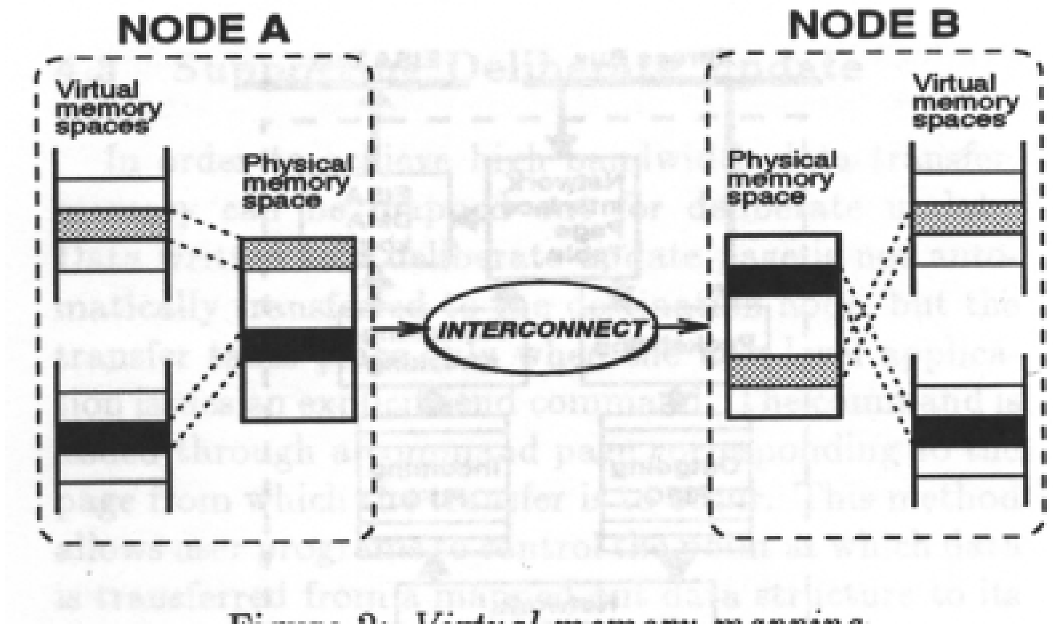


Figure 2: *Virtual memory mapping*

Hybrid DSM implementation

- SHRIMP@Princeton Univ.
  - Virtual Memory Mapped I/O
  - Automatic Update Release Consistency(AURC)

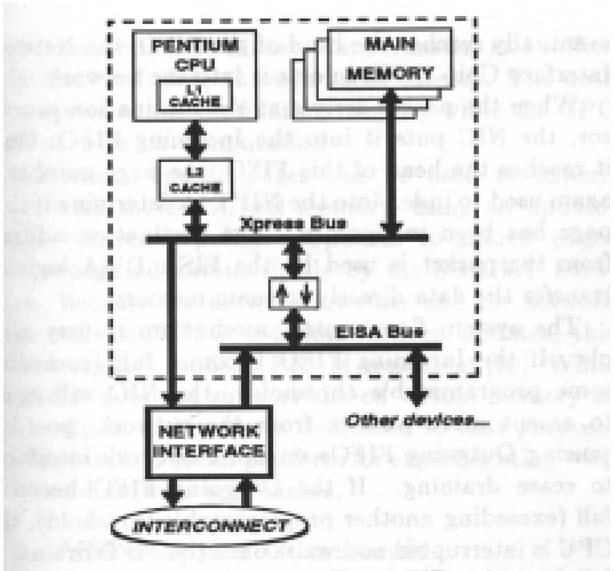


Figure 1: *A SHRIMP node with network interface*

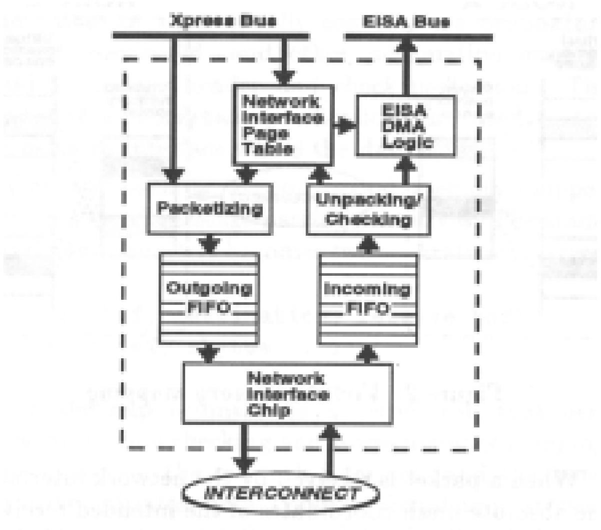


Figure 3: *SHRIMP Network Interface Data Path*

Hardware DSM 実装

- CC-NUMA
  - Directory-based

    → JUMP-1, Cenju-4, Origin2000, AsamA, **その他多数**
  - Broadcast-based ……. Reflective Memory

    → Memory Channel
- COMA Family

Hardware DSM 実装

- COMA Family

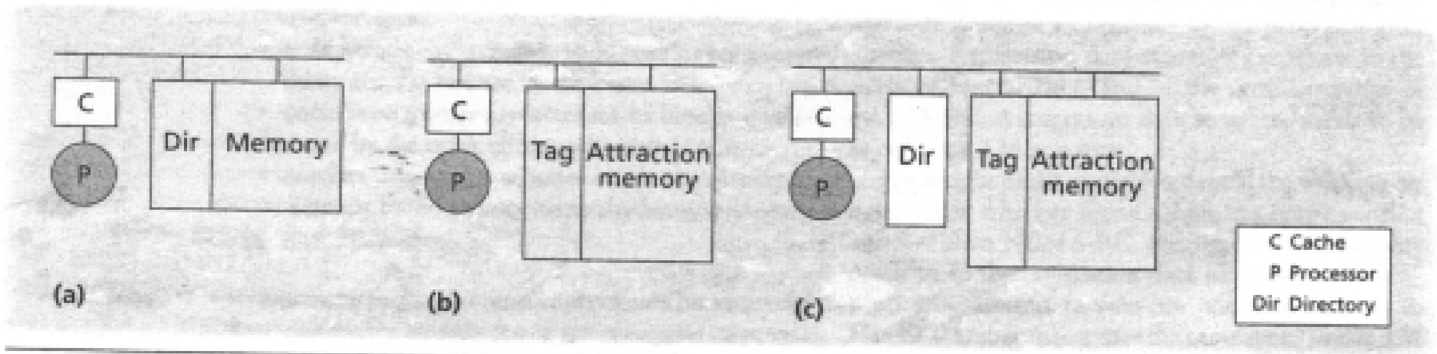
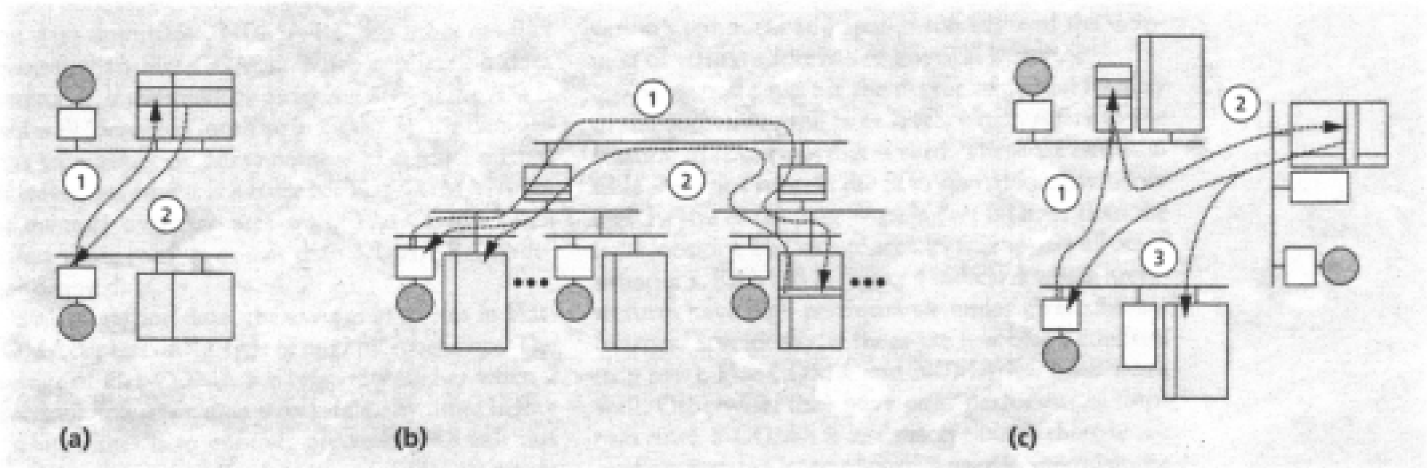


Figure 1. Node organization in (a) NUMA, (b) Hierarchical-COMA, and (c) Flat-COMA.



Memory Consistency Models

仮定１

　データを共有する場合には、生産者と消費者が定義でき、プログラムの不確定性をなくすためには生産者と消費者の間には必ず何らかの同期が存在する

仮定２

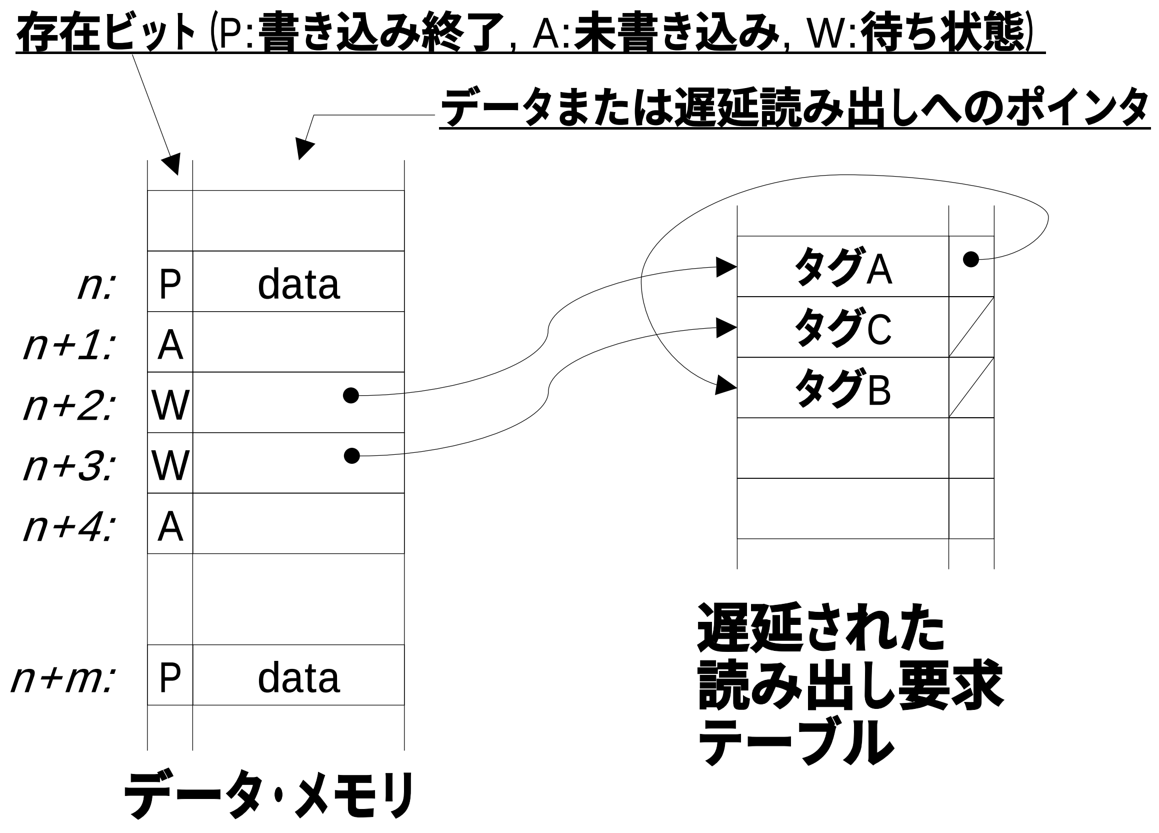
　同期のためのメモリ操作　と　それ以外のメモリ操作が区別可能である。

Memory Consistency Models

「特殊ケース」　同期と通信の融合

I-structure

- プログラミングモデルで 単一代入則を保証
- 専用HWによる同期機構 (マッチングメモリ)



存在ビット (P:書き込み終了, A:未書き込み, W:待ち状態)
データまたは遅延読み出しへのポインタ
n:
n+1:
n+2:
n+3:
n+4:
n+m:
P
A
W
W
A
P
data
data
タグA
タグC
タグB
遅延された読み出し要求テーブル
データ・メモリ

Memory Consistency Models

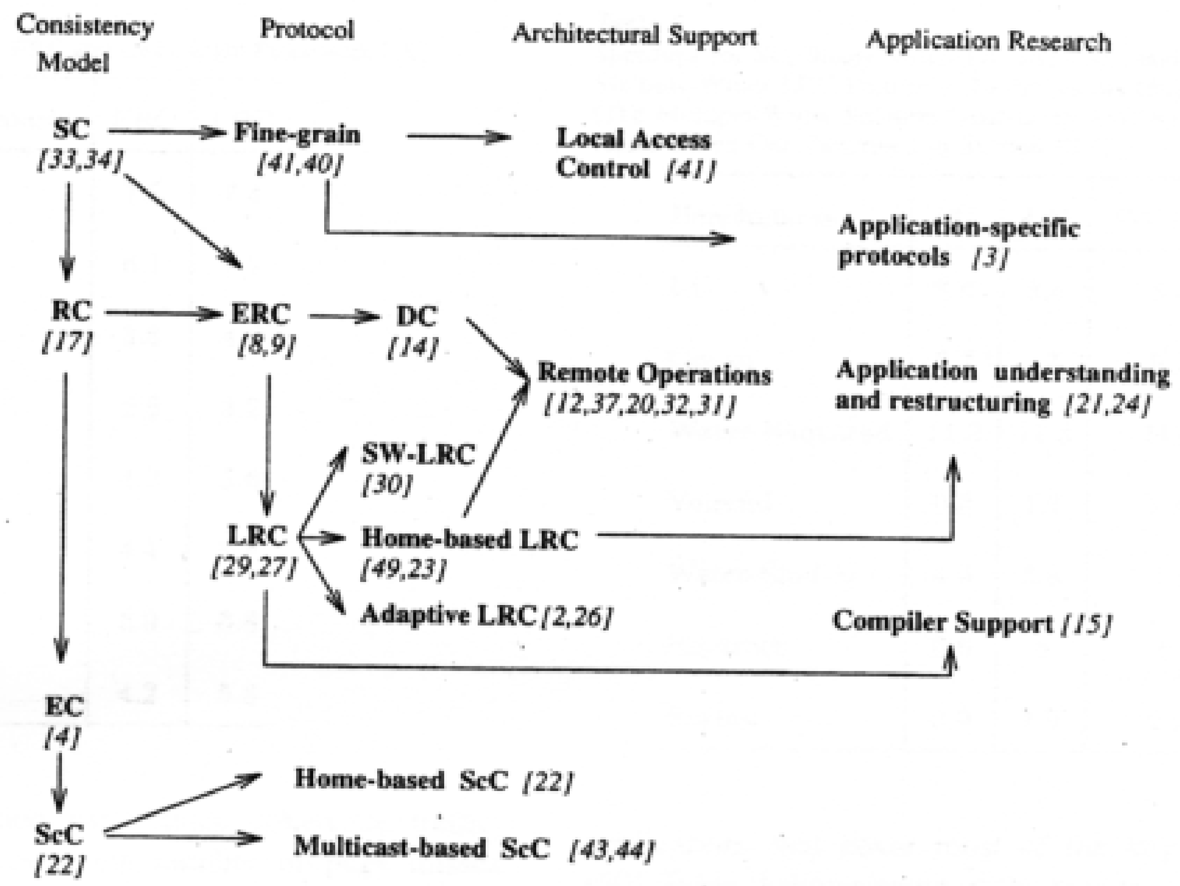


Fig. 1. Research in SVM. The figure treats lazy release consistency (LRC) and eager release consistency (ERC) as different protocols implementing the RC consistency model, though they are in fact slightly different consistency models. SW-LRC is single-writer LRC protocol. SC, RC, DC,

Important design choices in building DSM systems

- Cluster configuration
- Interconnection networks
- Shared data structure
- Coherence unit granularity
- DSM management responsibility
- Coherence policy

レポート課題

- 計算機クラスタ (森)
  - 課題：以下のいずれか一つの課題についてA4 1~2枚にまとめて報告せよ。
    - 並列処理システムを構築する必要が生じた場合を想定する (要求仕様は各自考えること)。 このとき、自分であればどのようなシステムを構築するか説明せよ。 (構築したシステムの特徴を要求仕様の特徴と関連付けて説明すること。)
    - 高速シリアル転送を行う際の8B10B符号化について調べよ
    - Wave Pipeline技術について調べよ
    - SMT(or HT by Intel)環境で同期プリミティブを実装する場合の注意点を考察せよ。
    - Memory Consistency Model の一つであるLRC(Lazy Release Consistency)モデルについて調べよ。
  - 〆切 1月29日 森 教官室へ提出 (不在の場合は事務室横の教官ポストへ提出しメールで提出を連絡すること)
    (今年修了予定の人は 1/22 までに提出のこと)

本資料のPDF版　と　講義に関する参考文献リスト　を
以下からアクセスできるようにしています。
ttp://www.lab3.kuis.kyoto-u.ac.jp/members/moris/lecture/PDS/

今は自由にアクセスできますが近日中にアクセス制限をかける予定です。）