# 分散共有メモリアーキテクチャにおけるキャッシュコンシステンシプロトコル

2W−4

大溝 孝　岩佐 繁明　林 宏雄
(株) 東芝 総合研究所

## 1 はじめに

現在分散共有メモリアーキテクチャは共有メモリ、分散メモリアーキテクチャの利点を合わせ持つ方式として注目されている。Stanford 大の DASH[Lenoski92] では複数のメモリとキャッシュデータの一貫性を保証する方法としてメモリにディレクトリを設けて管理する方法が提案されている。今回我々はより一般的に用いられているスヌープ方式を拡張して分散共有メモリでのキャッシュの一貫性を保証するプロトコルについて提案する。

## 2 分散共有プロトコルの目的

現在さまざまな並列計算機アーキテクチャが提案されている。これらは一つのメモリを共有する共有メモリアーキテクチャ、プロセッサが独自のローカルメモリを持つ分散メモリアーキテクチャ、ローカルメモリを他のプロセッサから共有することの出来る分散共有メモリアーキテクチャの3つに大別できる。

共有メモリアーキテクチャの利点は実装が容易であることと、アプリケーションからメモリアーキテクチャを意識する必要がないことである。

しかしプロセッサ台数が増えるとメモリ、バスのアクセス競合によるオーバヘッドが大きくなり、性能が頭打ちになってしまうことが知られている。

一方分散メモリアーキテクチャではより多くのプロセッサを接続することが可能であるが、共有データにより頻繁に通信を必要とする場合は通信のレーテンシによるオーバヘッドが問題となり、応用分野が限られてしまう。

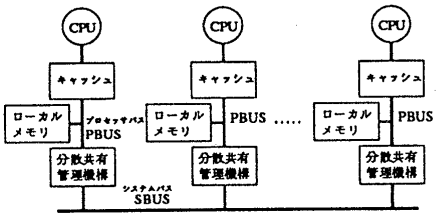


図 1: 分散共有メモリアーキテクチャ

そこで我々は図1に示す、それぞれの欠点を補うことが可能な分散共有メモリアーキテクチャを提案する。

分散共有メモリではそれぞれのプロセッサは自分のローカルメモリを用いて処理を行なうが、互いのローカルメモリをシステムバス (以下SBUS) を通してアクセスすることも可能である。

共有しないデータについてはプロセッサバス (以下PBUS) を用いたローカルメモリへのアクセスのみで済むため、SBUS へのトラフィックを発生しない。よってバスのアクセス競合によるオーバヘッドは小さくなる。

従ってメモリアクセス全体の 90% が共有されていない自分のローカルメモリへのアクセスであったならば、90% のバストラフィックは DMF によってブロックされ、SBUS の負荷を軽減することが可能であると考えられる。

また、共有が必要なデータについても SBUS を通してアクセスできるため、通信によるオーバヘッドが少なくなる。

ここで問題になるのはプロセッサ間でのキャッシュデータの一貫性の保証である。現在の高性能プロセッサはキャッシュの使用が不可欠であり、プロセッサ間でのキャッシュコンシステンシを保証することが必要である。これは分散共有管理機構 (以下 DMF) において分散共有プロトコルに基づき PBUS, SBUS をスヌープすることにより保証される。

今回この分散共有メモリアーキテクチャにおいて、キャッシュコンシステンシを保証する分散共有プロトコルについて報告する。

## 3 分散共有プロトコル

通常の共有メモリアーキテクチャにおけるキャッシュコンシステンシプロトコルではメモリに対するすべてのアクセスに関して一貫性保持操作を行わねばならないためシステムバスへのトラフィックを発生する。しかしその中にはプロセッサ間で共有されていないデータ (ローカルデータ) に関するアクセスも含まれており、それによって SBUS は無駄な占有時間を費やすことになる。

この占有時間はプロセッサの台数が増えるにつれて大きくなり、そのために生じるバス待ち時間の間プロセッサは命令を実行することが出来ない。このため大きなスケーラビリティを得ることは困難である。

もしプロセッサ間のデータ共有は多くないと考えると、大半のアクセスに関してはバスへ不要なアクセスを発行していることになる。

従ってこれらの無駄なアクセスを抑制すれば SBUS 上のトラフィックは減少し、より大きなスケーラビリティを得ることが可能である。

分散共有プロトコルは分散共有メモリアーキテクチャにおいて、上記の無駄なアクセスを抑制し、かつプロセッサ間で共有されているデータ (グローバルデータ) の一貫性を保証するキャッシュコンシステンシプロトコルである。

分散共有プロトコルの概念を図2に示す。

ここでサーバントとはアクセスの対象となるアドレスのメモリを持つプロセッサモジュールのことであり、それ以外のプロセッサモジュールをゲストと呼ぶ。

プロセッサが自分のローカルメモリにアクセスを行なう時、それが自分固有のデータ (ローカルデータ) であれば DMF によりブロックされ SBUS へはアクセスを発生しない。(図 2-a)

それが他のプロセッサにより共有されているデータ (グローバル) であれば、分散共有機構は正しいデータを得るために SBUS に対してアクセスを発生する。(図 2-b)

他の DMF はこのアクセスを監視し、正しいデータが返されるように動作するが、自分のキャッシュに対して何もする必要がない場合は PBUS に対してアクセスを発生することはないが、もし自分のキャッシュがシステム中でもっとも新しい値を有する場合、すなわちダーティな値を持つ時はサーバントのローカルメモリに代わってアクセスを処理する必要がある。(図 2-b)

---

A cache consistency protocol for a distributed memory architecture.
Takashi Omizo, Shigeaki Iwasa, Hiroo Hayashi
Toshiba Corporation R & D Center.

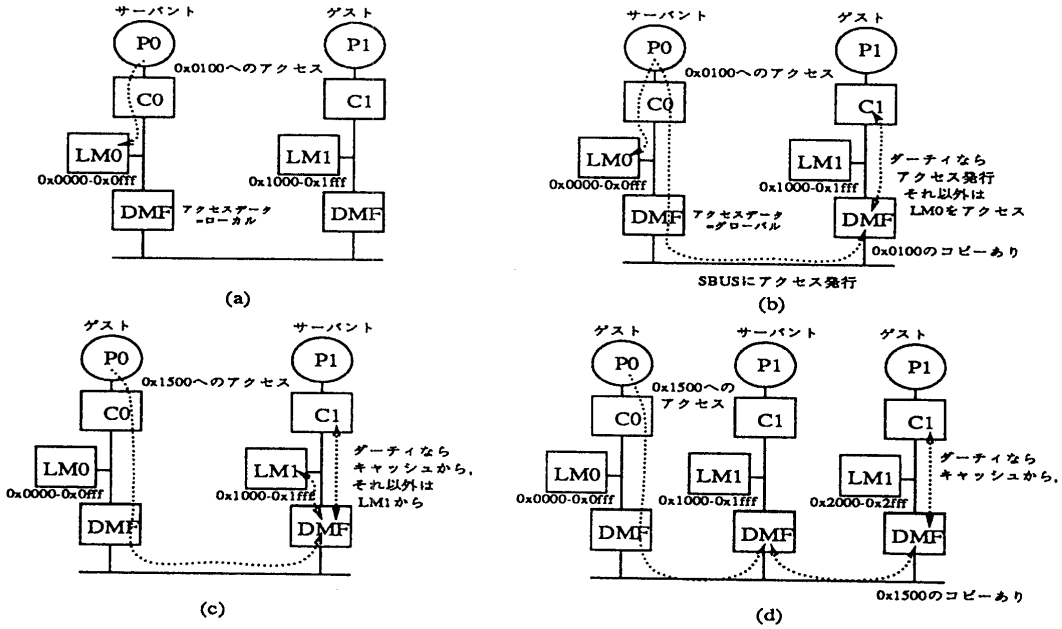


図 2: 分散共有プロトコルの概念

他のプロセッサのローカルメモリにアクセスする場合も、 DMF は SBUS に対してアクセスを発生する。(図 2-c)

この場合も同様に他の DMF はこのアクセスを監視し、 自分のキャッシュがダーティなデータを持っている場合はアクセスを処理する必要がある。 (図 2-d)

前述のように、 本プロトコルではローカルメモリ中のデータについて、 キャッシュラインサイズ単位でローカルデータであるかグローバルデータであるかの区別を行なっている。

ローカルデータ そのローカルメモリを所有するプロセッサのキャッシュ以外にはコピーが存在しないデータ。

グローバルデータ そのローカルメモリを所有するプロセッサのキャッシュ以外にコピーが存在する可能性のあるデータ。

このためローカルデータとグローバルデータを区別するために、 ローカルメモリのキャッシュラインサイズ毎にローカルかグローバルかを示すタグが必要である。このタグは メモリタグ (MTag) と呼ばれ、 1 ビットでそのデータがローカルであるかどうかを示す。

また自分のキャッシュデータが最新であるかどうかを知るために、 DMF はプロセッサキャッシュのタグ状態のコピーを持っている。このタグはキャッシュタグ (CTag) と呼ばれ、キャッシュラインの状態およびキャッシュにヒットしたか否かの判定のためのアドレス上位ビットを含んでいる。

DFM は MTag, CTag の情報を参照し PBUS からのアクセスを SBUS へ、 SBUS からのアクセスを PBUS へそれぞれ発行する必要があるかどうか判定することができる。

以上分散共有プロトコルの概念について述べた。本稿では紙面の都合により詳細な動作を示すことは出来ないが、 MOESI[Sweazey86] の 5 状態を有するコピーバック・インバリデート型キャッシュに対する本プロトコルが実現されている。

例えばこの MOESI の 5 状態を持つコピーバック・インバリデート型キャッシュについて本プロトコルを適用した場合その基本動作は以下のようになる。

- 自分のローカルメモリ中のローカルデータへのアクセスは SBUS へのアクセスを必要としない。
- SBUS 上に自分が最新のコピーを有するキャッシュライン (状態 M, O) への読み出しをスヌープした場合、 PBUS へ読み出しを発行し、 SBUS へデータを供給しなければならない。
- SBUS 上に自分がコピーを有するキャッシュライン (状態 M, O, E, S) への無効化をスヌープした場合、 PBUS へ無効化を発行し、そのラインを無効化しなければならない。

## 4 おわりに

分散共有メモリアーキテクチャにおけるキャッシュコンシステンシプロトコルである分散共有プロトコルの概念について述べた。

今後はシミュレーション等による性能評価により本プロトコルの有効性を確認する予定である。

## 参考文献

[Lenoski92] Daniel E. Lenoski. *The Design and Analysis of DASH: A Scalable Directory-based Multiprocessor*, Stanford Univ. Tech. report No. CSL-TR-92-507,Feb 1992.

[Sweazey86] Paul Sweazey, Alan Jay Smith. *A Class of Compatible Cache Consistency Protocols and their Support by The IEEE Futurebus*, 13th ISCA,June 1986.