特別研究報告

題目

フォトニックグリッド環境における
分散計算のためのメモリアクセス手法の提案

指導教官

村田 正幸 教授

報告者

中本博久

平成 15 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 14 年度 特別研究報告

フォトニックグリッド環境における分散計算のためのメモリアクセス手法の提案

中本博久

内容梗概

WDM 技術を基盤としてインターネットの高速化を図る、いわゆる IP over WDM ネットワークの研究開発が、現在さかんに進められている。また、それを一歩進めて WDM 技術以外のさまざまなフォトニック技術を下位レイヤの通信技術とした GMPLS と呼ばれるインターネットのルーティング技術の標準化も IETF で進められている。さらに、フォトニックネットワークの真の IP 化を狙って、フォトニック技術に基づいたフォトニックパケットスイッチに関する研究も始められつつある。しかしながら、これらの諸技術は現在のインターネット技術を是としている。すなわち、情報を扱う細粒度として IP パケットを扱い、ネットワーク上でそれをいかに高速に運ぶかを研究開発の目標としている。そのため、パケット交換技術に基づいたアーキテクチャをとる限り、個々のコネクションに対する高品質通信の実現は非常に難しい。

Storage Area Network やグリッド計算など新しい応用技術では、高速かつ、高信頼な通信パイプをエンドユーザに提供する必要があり、そのためには、エンドユーザ間に大容量波長パスを設定し、ユーザに提供することが考えられる。すなわち、既設のファイバを利用し、あるいは必要に応じて、ファイバを新たに敷設し、ファイバおよびファイバ内に多重化された波長を最小粒度として情報の交換を行うフォトニックネットワークを構築することによって、超高速かつ高品質な通信パイプをエンドユーザに提供することが可能である。そこで、ネットワークノードや計算機群を光ファイバで接続したフォトニックグリッド上に仮想チャネルをメッシュ状に張ることにより、高速チャネル上での分散計算が可能になる。さらには、フォトニックグリッド上に仮想リングを構成し、リング上にデータを載せることによって、波長を仮想的な共有メモリとすることも可能である。その結果、広域分散システムにおける

共有メモリと通信チャネルの区別の必要がなくなり、コンピュータ間の高速なデータ交換が可能になると考えられる。

本報告では、フォトニックグリッド上に仮想リングを構成した際の共有メモリアクセス方式を提案した。共有メモリとして光リングを構成する場合、従来の共有メモリシステム同様に、共有メモリに対する競合とキャッシュの整合性の問題が生じる。また、長距離ファイバ上に展開する仮想光リングへはアクセスのタイミングや頻度に制約がある。そのため、そのことを十分に考慮したアクセス方式を考え、シミュレーションを用いてその性能を評価した。その結果、広域に分散した計算機群で仮想光リングを用いてデータの共有をはかる場合、メモリアクセスの遅延の影響が大きいことから、制御メッセージの交換の少ない方式が特に有効であることがわかった。

主な用語

フォトニックグリッド、光リング、共有メモリ、メモリアクセスの競合、キャッシュの整合性、スヌープ手法

目 次

図目次

1　はじめに

近年のインターネットをはじめとするネットワークの利用者の増大により、ネットワークを流れるトラヒック量は増大する一方である。特に、映像などを利用したさまざまなアプリケーションが利用されるようになり、ネットワークにおける高速かつ大容量な伝送を可能とする技術への要求はますます高まっている。

これらの要求を満たすために、現在、光伝送技術を用いた研究が活発に進められている。特に、光の波長を多重化して利用する WDM (Wavelength Division Multiplexing) 技術が開発の中心であり、1000 波を利用できる新たな WDM 技術の研究も進められている [1]。さらに、WDM 技術を基盤としてインターネットの高速化を図る、いわゆる IP over WDM ネットワークの研究開発が、現在さかんに進められている。また、それを一歩進めて WDM 技術以外のさまざまなフォトニック技術を下位レイヤの通信技術とした、GMPLS (Generalized Multi-Protocol Label Switching) と呼ばれるインターネットのルーティング技術の標準化も IETF (the Internet Engineering Task Force) で進められている [2]。さらに、フォトニックネットワークの真の IP 化を狙って、フォトニック技術に基づいたフォトニックパケットスイッチに関する研究も始められつつある [3, 4]。

しかし、これらの諸技術は現在のインターネット技術を是としている。すなわち、情報を扱う細粒度として IP パケットを扱い、ネットワーク上でそれをいかに高速に運ぶかを研究開発の目標としている。そのため、このようなパケット交換技術に基づいたアーキテクチャをとる限り、個々のコネクションに対する高品質通信の実現は非常に困難である。例えば、SAN (Strage Area Network) やグリッド計算など新しい応用技術では、高速かつ、高信頼な通信パイプをエンドユーザに提供する必要があり、そのためには、エンドユーザ間に大容量波長パスを設定し、ユーザに提供することが考えられる。すなわち、既設のファイバを利用し、あるいは必要に応じて、ファイバを新たに敷設し、ファイバおよびファイバ内に多重化された波長を最小粒度として情報の交換を行うフォトニックネットワークを構築することによって、超高速かつ高品質な通信パイプをエンドユーザに提供することは可能である。

そこで、本報告では、ネットワークノードや計算機群を光ファイバで接続したグリッド環境において、グリッド上での通信を波長パスを利用して行う新たなアーキテクチャとして

フォトニックグリッド環境を考える。従来のグリッド環境においては、メッセージパッシング (MPI; Message Passing Interface) を TCP/IP を利用して実現していたが、フォトニックグリッド環境においては、グリッド上の通信を従来の TCP/IP を用いて実現するのではなく、あらかじめ設定した波長パスを利用することにより高速かつ高信頼な通信を可能とする。すなわち、フォトニックグリッド上にデータ通信用の仮想チャネルをメッシュ状に張ることにより、高速チャネル上での分散計算が可能となる。また、フォトニックグリッドを構成するネットワークノードおよび計算機群を結ぶ仮想リングを想定し、仮想リング上の波長を高速な共有メモリとして利用することも可能である。その結果、広域分散システムにおける共有メモリと通信チャネルの区別の必要がなくなり、コンピュータ間の高速なデータ交換が可能になる。

本章では、フォトニックグリッド環境を実現するための、波長パスを用いてデータ交換を行う新たなフォトニックネットワークアーキテクチャを対象に、計算機群が効率よくフォトニックネットワークへアクセスする手法を提案する。ネットワークノードや計算機群を光ファイバで接続したフォトニックグリッド上に仮想チャネルをメッシュ状にはることにより、高速チャネル上で共有すべきデータの受け渡しを行う。さらに進め、フォトニックグリッド上に仮想リングを構成し、リング上にデータを載せることによって、波長パスを伝送路ではなく、仮想的な共有メモリとすることも可能である。その結果、広域分散システムにおける共有メモリと通信チャネルの区別の必要がなくなり、計算機間の高速なデータ交換が可能になると考えられる。

以上より、本報告では、フォトニックグリッド環境を実現する波長パスを用いたデータ交換を行う新たなフォトニックネットワークアーキテクチャを対象に、それらのフォトニックネットワークへアクセスする手法を提案する。まず、フォトニックグリッド上の仮想リングを共有メモリとして利用する場合のアクセス方式を提案する。すなわち、分散配置された計算機群に対する共有メモリをネットワーク上に展開するためのアクセス方式を実現し、フォトニックグリッドの活用技術に対する API を提供する。具体的には、仮想光リングを共有メモリとして利用し、各計算機群のローカルメモリや CPU におけるキャッシュを共有メモリに対するキャッシュとして利用することを考える。仮想光リング上の伝送路として多重化さ

れた波長パスを用いることとし、その光リングへのアクセスは、各波長へアクセス可能なインターフェースを介して行う。仮想光リングを共有メモリとして利用する場合、同一計算機内の共有メモリのバス結合とは異なり、長距離の光ファイバ上に展開しているためアクセスのタイミングや頻度に制約を受ける。従って、通常の共有メモリシステム以上に、仮想光リングにおける共有メモリと各計算機群のキャッシュのコヒーレンシを十分考慮する必要がある。以上、述べたような特徴を考慮し、フォトニックグリッド環境を想定した光共有メモリのアクセス手法を提案し、シミュレーションを用いてその性能を明らかにする。

以下、2 章では、マルチプロセッサ技術、共有メモリシステム技術、キャッシュコヒーレンシ技術などの従来の共有メモリに関するアクセス手法について述べる。3 章では、対象とするフォトニックグリッド環境と提案する共有メモリアクセス手法について述べる。4 章では、提案手法の評価を行う。5 章で、まとめと今後の課題について述べる。

2　共有メモリシステムの概要

本章では、従来の共有メモリシステムについて述べる。まず、複数のプロセッサを持つマルチプロセッサシステムについて述べ、次にそれらの複数のプロセッサでメモリを共有する共有メモリシステム、さらには、共有メモリシステムにおけるアクセス制御方式とキャッシュの整合性について述べる。

2.1　マルチプロセッサシステムの概要

複数のプロセッサを持つマルチプロセッサシステムは、MIMD (Multiple Instruction / Multi Data) 型の並列計算機として、UMA（Uniform Memory Access: 均一アクセス) モデル、NUMA（Non-Uniform Memory Access: 不均一アクセス) モデル、NORA（NO Remote Memory Access: 無遠隔アクセス) モデルがあげられる。

UMA モデルは、すべてのプロセッサがアドレス空間を共有し、同一時間でアクセス可能な共有メモリのモデル、または、そのようなメモリを持つ計算機のことである。NUMA モデルは、すべてのプロセッサが、アドレス空間を共有するメモリを持つが、あるプロセッサから見た時のアクセス速度は、メモリの番地によって異なるモデル、または、そのようなメモリを持つ計算機のことである。NORA モデルは、各プロセッサは互いに独立したアドレス空間のメモリを持ち、メッセージのやりとりによって計算を進めていくモデルである。つまり、NORA モデルは共有メモリを持たないモデルである。

一般にマルチプロセッサシステムとは、並列計算機全体を指す場合もあるが、共有メモリを持つ MIMD 型の計算機、つまり UMA モデルと NUMA モデルを指す場合が多い。場合によっては、1) 共有メモリを有する、2) 各プロセッサの能力が均等である、3) I/O を共有する、4) 分散 OS が動作する、の 4 つの条件を満足するシステムのみを指す。本報告では、共有メモリを持つマルチプロセッサシステムを対象とする。

2.2　共有メモリシステム

本節では、共有メモリシステムに着目し、複数のプロセッサを持つシステムで共有メモリを持つ UMA モデル，NUMA モデルについて述べる。

UMA モデルでは、均一なアクセスを前提としている（図 1 参照)。すなわち、共有メモリはすべてのプロセッサから等距離にあり、プロセッサ能力にも差がないシステムでなければならない。基本的に共有メモリへのアクセスは、共有バスを介して行われる。また、ひとつの OS で共有メモリ空間を管理しているため、比較的制御が容易である。しかしながら、均一なアクセスを要求するため、スケーラビリティに欠けるという問題がある。

一方、NUMA モデルでは、不均一なアクセスを許しており、スケーラビリティが高い（図 2 参照)。共有メモリは、プロセッサの能力や距離に依存せず、配置される。プロセッサ、あるいは複数のプロセッサでひとつのサブシステムをなし、OS は、それぞれ独立に稼働し、多くの場合、それぞれのサブシステムが持つメモリを分散共有メモリとして利用する。そのため、それぞれのサブシステムは共有メモリに対する固有のアドレス空間を持ち、他のサブシステムからは、アドレス変換機構を参照する。従って、他のサブシステムにある共有メモリ部分にアクセスする際は、そのアドレス変換等を行うためオーバヘッドが生じ、十分な性能が得られない場合もある。

2.3　共有メモリの競合回避とキャッシュの整合性

共有メモリシステムにおける重要な問題は、複数のプロセッサから共有メモリにアクセスする際に発生する競合を回避することである。すなわち、各プロセッサから同一の共有メモリにアクセスする場合、あるいは同時にプロセッサと共有メモリを結ぶバスなどの伝送路を利用する場合に競合が発生するため、ロックなどを利用して回避する必要がある [5-7]。

また、それらの競合回避のためのオーバヘッド処理によりメモリアクセス時に制約を受け、プロセッサの性能を十分に引き出せなくなる可能性がある。そこで、高速なキャッシュを用いて共有メモリへのアクセスを減らし、性能向上を図る。その際、キャッシュとメモリの整合性を保つ機構が重要になる。本節では、これらの手法について詳しく述べる。

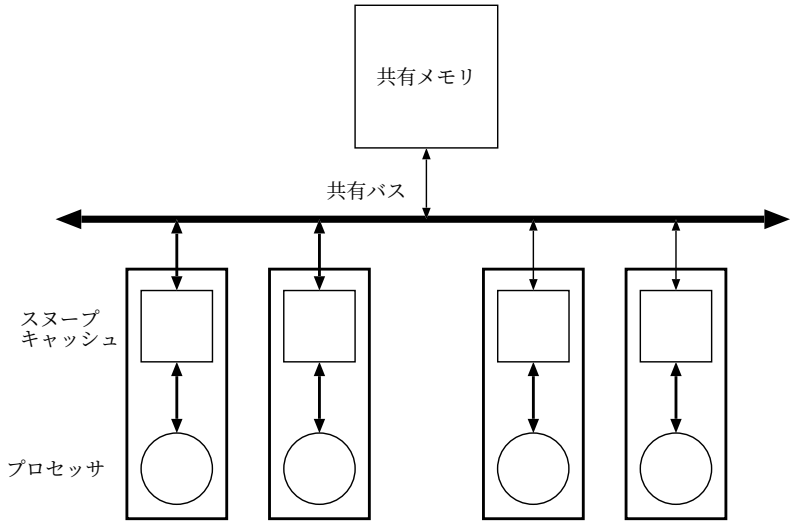

図 1: バス結合型 UMA モデル

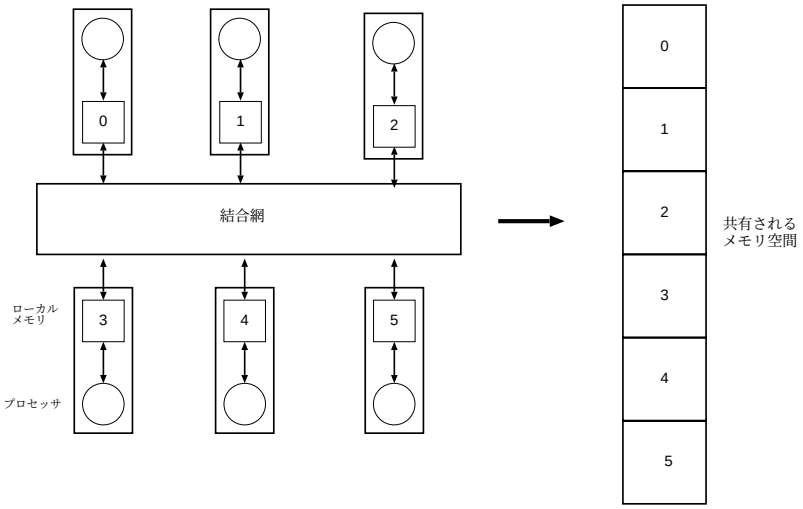

図 2: NUMA モデル

2.3.1　共有メモリにおける競合回避

まず、共有メモリにおける競合について述べるため、各プロセッサがキャッシュを持たない場合を想定する。各プロセッサが必要なデータを取得する基本的な手順は、各プロセッサ間で同じデータを参照しても差し支えないため、同一の共有メモリに対する競合は考慮しなくてもよい。しかしながら、共有バス上では競合が生じるため、ロック機構を用いて競合回避を図る。すなわち、取得の制御メッセージを制御バスに転送し、データを取得する。

各プロセッサが処理したデータを書き出す、あるいは書き換える場合の基本的な手順も、共有メモリ上、および共有バス上で競合が発生するため、ロック機構を用いて競合を回避する。すなわち、データを書き込みの制御メッセージを制御バスに転送し、共有バス、共有メモリとも保護し、データを書き込む。

2.3.2　キャッシュの整合性

各プロセッサから共有メモリへのアクセスは、共有バスにおける競合回避のため、アクセスに制約を受ける。そこで、共有メモリへのアクセスを減らすためにキャッシュを用いた高速化が図られるのが一般的である。通常、プロセッサごとにキャッシュをもち、1 次キャッシュ、2 次キャッシュ、3 次キャッシュと多段に構成される。1 次キャッシュは、プロセッサ内に配置され、容量は小さいが高速に動作する。2 次、3 次キャッシュはプロセッサ外に配置され、1 次キャッシュより容量は少し大きくなるが、プロセッサへの転送速度は 1 次キャッシュに比べ遅くなる。

各プロセッサがキャッシュを持つ場合は、キャッシュ上のデータと共有メモリ上のデータの整合性を十分に考慮する必要がある。キャッシュの整合性を解決する手法に次の 2 つの方式がある。

- ディレクトリ方式

共有メモリがキャッシュブロックと同じ大きさのブロックに分割されている。そして、共有メモリのブロックに対応して、ディレクトリが存在する。ディレクトリのエントリの数は共有メモリのブロック数と同じであり、ディレクトリのエントリには、対応

するブロックが存在するキャッシュの場所を記憶しておく。キャッシュの無矛盾性を保つためにキャッシュ内のブロックを無効にするときは、ディレクトリのエントリをみてそこに書いてあるキャッシュで真の値を持っているところ以外にメッセージを送り無効にする。こうすれば、ブロードキャストする必要がなくなり、無駄なトラヒックは生じない。

- スヌープキャッシュ方式

  キャッシュ間のデータの一致制御の目的で、共有バス上のメモリアクセスを監視し、必要に応じて自キャッシュブロックに対する制御を分散的手法で行う。このようなキャッシュコントローラをスヌープキャッシュコントローラと呼ぶ。共有バス上に観測されたそれぞれのメモリアクセスごとに、スヌープキャッシコントローラは、対応するコピーが自キャッシュ中に存在しているのかどうかを、自キャッシュディレクトリのタグと共有バス上のアドレスを比較することによって決定する。その結果、対応するコピーが自キャッシュ中に存在する場合には、自キャッシュブロックのステートの変更、要求元に対するなんらかの返答など、その実装上で定義されているなんらかのデータの一致性制御の動作を実行する。一方、要求元のスヌープキャッシュコントローラも他キャッシュからの返答をみて、自キャッシュブロックのステートを決定するなど、実装上において定義された一致性制御の動作を実行する。この実装上に定義された一致性制御の動作は、スヌープキャッシュプロトコルと呼ばれる。

次に、プロセッサとメモリ間でデータ交換する際の手順について考える。各プロセッサがキャッシュを持つ場合のデータへのアクセス手法はかなり複雑になる。各プロセッサがデータを取得する基本的な手順は、まず、1 次キャッシュから順に探し、キャッシュにデータが存在しない場合にはじめて共有メモリにアクセスする。ただし、取得するデータが他のプロセッサの持つキャッシュになければ、共有メモリから取得すればよいが、他のプロセッサのキャッシュに存在する場合には、対応の仕方によっていくつかの手法が考えられる。

各プロセッサが処理したデータを書き出す、あるいは書き換える場合は、さらに複雑になり、キャッシュに書き出す、共有メモリにも書き出す、他のプロセッサのキャッシュに対象

となるデータが存在する場合の対応、などによって、いくつかの手法がある。

ここでは、スヌープキャッシュを利用する場合の手法について説明する。スヌープキャッシュプロトコルとして、キャッシュ一致プロトコルは、一致させるタイミング（ライトスルー、ライトバック）と方法（無効化、更新）によって 4 つに分類される。以下に、その 4 つのプロトコルの動作を示す。

1. 無効化型ライトスルー (Invalidate write through)

   各データの状態は、有効 (V: Valid)、無効 (I: Invalid) の 2 状態しか持たない。プロセッサ A、 B、 C のキャッシュが同一データを共有している時、プロセッサ A がそのキャッシュ上のデータに書き込みを行ったとする。この書き込みデータは、共有バスを通じて直接共有メモリに送られ、対応するデータを書き換える。この時、プロセッサ B、 C のキャッシュコントローラは、共有バス中のアドレス、コマンドから、このアドレスに書き込みが起こったことを知り、書くキャッシュのデータの状態を I にして無効にする。すなわち、書きこみが行われる度に、他のキャッシュ上のデータを無効にすることでキャッシュの内容の一致性を保証する。プロセッサ A のキャッシュ上に存在しない、データに書き込んだ場合も、データは直接共有メモリに送られるとともに、これを共有しているキャッシュB、 C を無効化する。この書き込みデータは、共有バスを通じて直接主記憶に送られるとともに、このコピーを共有しているキャッシュB、 C を無効化する。

2. 更新型ライトスルー

   無効化型同様、書き込みデータは、共有バスを通じて共有メモリに送られるが、同時にこのデータを共有している他のプロセッサのキャッシュ上のデータの内容も書き換える。この方法は、全てのデータが常に最新のデータを持つことで、キャッシュの内容の一致性を保証する。

3. 無効化型ライトバック

   各データの状態は、無効 (I: Invalid)、共有メモリと一致 (C: Clean)、共有メモリと不一致 (D: Dirty) の 3 状態を持つ。複数あるいはひとつのプロセッサがあるアドレスを参

照すると、データが共有メモリからコピーされ、複数のプロセッサのキャッシュ上でC 状態になる。C 状態のデータは、共有メモリと内容が一致しているので、このラインに対する読み出しはバス操作を伴わない。ここで、C 状態のデータに対して書き込みを行うと、そのデータはD 状態になる。この時、バス上には無効化を示す信号と、ブロックに対応するアドレスが送出される。他のプロセッサのキャッシュコントローラは、バスを監視し、対応するデータを保持していればそれを無効化する。以後、D 状態のデータに対する読み書きは、バスを介さずに行える。D 状態のデータのアドレスに対して他のプロセッサが読み出し要求を出した場合、D 状態のデータを共有メモリに対して書き戻して一致をとる。次に、要求を出したプロセッサに対して転送が行われ、両方のキャッシュはC 状態になる。一方、D 状態のラインのアドレスに対して他のプロセッサが書きこみ要求を出した場合、読み込み同様、D 状態のデータの書き戻しが起こり、次に要求を出したキャッシュに対して転送が行われる。最後に要求をだしたプロセッサは自分のキャッシュ上のデータに書きこみを行い、自分も状態はD 状態になる。もともとデータを保持していたキャッシュは無効化される。

4. 更新型ライトバック

無効化型が、書き込みに際して他のデータを無効にしてしまうのに対して、更新型は、書き込んだデータで相手のキャッシュを更新することにより一致をとる。

3　仮想光リングを用いた共有メモリアクセス手法の提案

広域に分散した計算機間やシステム間でデータを共有する環境のひとつに、自律分散環境がある [8-12]。自律分散環境では、共有するデータをデータフィールド (DF; Data Field) と呼ばれるネットワークを通じて転送し、必要な計算機やシステムがデータを取得し、処理した後、再びネットワークに送出する、という手順を繰り返すことにより、各計算機やシステムが自律的に稼働する。具体的には、送信側は、データフィールドにコンテントコード (CC; Content Code) と呼ばれるデータの内容を示すヘッダをつけたメッセージを DF に送出し、受信側では、あらかじめ記録しておいた取得すべき CC と受け取ったデータの CC が一致するとシステムに取り込み処理を行う。このデータ受渡し手法により同期操作が必要なくなり、データ共有が容易になる。

本章では、フォトニックグリッド環境を実現する波長パスを用いたデータ交換を行う新たなフォトニックネットワークアーキテクチャを対象に、それらのフォトニックネットワークへアクセスする手法を提案する。ネットワークノードや計算機群を光ファイバで接続したフォトニックグリッド上に仮想チャネルをメッシュ状にはることにより、高速チャネル上で共有すべきデータの受け渡しを行う。さらに進め、フォトニックグリッド上に仮想リングを構成し、リング上にデータを載せることによって、波長パスを伝送路ではなく、仮想的な共有メモリとすることも可能である。その結果、広域分散システムにおける共有メモリと通信チャネルの区別の必要がなくなり、計算機間の高速なデータ交換が可能になると考えられる。

まず、本報告においては、仮想光リングを共有メモリとして利用し、各計算機群のローカルメモリや CPU におけるキャッシュを共有メモリに対するキャッシュとして利用することを考える。次に、仮想リングを共有メモリとして利用するのではなく、共有データを交換する高速チャネルとして利用する場合について考える。すなわち、仮想リングの距離が比較的短い場合や、仮想リングの容量が小さくなる場合を想定する。具体的には、他ノードのローカルメモリも自分のメモリと同一の空間でアクセスできる機構を持つモデルを考える。

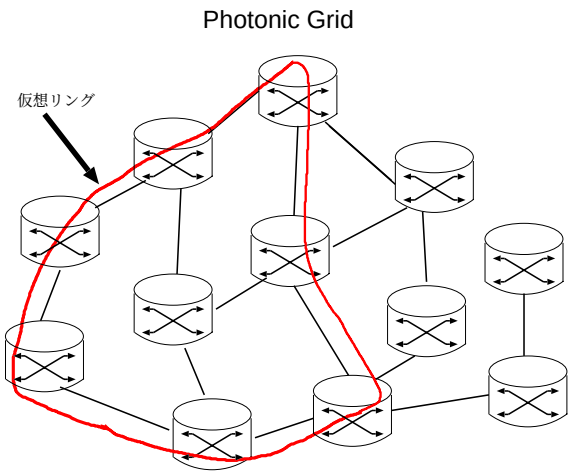

図 3: フォトニックグリッドのアーキテクチャ

3.1 共有メモリ型モデルにおけるアクセス手法

仮想光リングを共有メモリとして用いるモデルを図 5 に示す。仮想光リングを共有メモリとして利用し、各計算機群のローカルメモリや CPU におけるキャッシュを共有メモリに対するキャッシュとして利用することを考える。仮想光リング上の伝送路として多重化された波長パスを用いることとし、その光リングへのアクセスは、各波長へアクセス可能なインターフェースを介して行う。仮想光リングを共有メモリとして利用する場合、同一計算機内の共有メモリのバス結合とは異なり、長距離の光ファイバ上に展開しているためアクセスのタイミングや頻度に制約を受ける。従って、通常の共有メモリシステム以上に、仮想光リングにおける共有メモリと各計算機群のキャッシュの整合性を十分考慮する必要がある。以上、述べたような特徴を考慮し、フォトニックグリッド環境を想定した光共有メモリのアクセス手法を提案する。

図 5 に示すように、フォトニックグリッド上の計算機群を仮想光リングの計算機ノードと呼ぶ。仮想光リングによる共有メモリシステムは、各計算機ノードあるいは各プロセッサがひとつの共有メモリにアクセスするシステムであるため、従来の共有メモリシステムに近い。すなわち、共有メモリへのアクセスに関しては、NUMA のような分散共有メモリに対するアクセスではなく、それぞれの計算機ノードは独立であるものの UMA の共有メモリに対

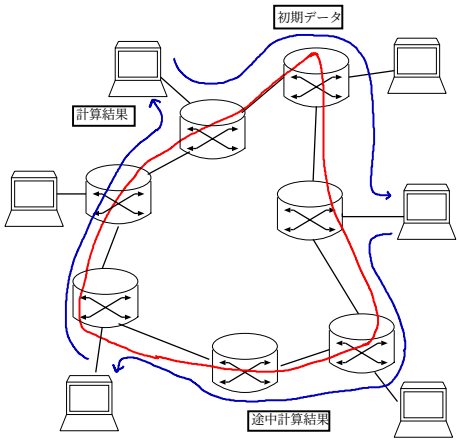

図 4: 仮想リングの構成

するアクセスに近いといえる。しかしながら、従来の共有メモリシステムでは、共有バスにより共有メモリにアクセスするが、光リング共有メモリシステムの場合は共有バスはなく、各計算機ノードが直接共有メモリにアクセスする。また、従来の共有メモリシステムが、ランダムアクセス可能であるのに対して、光リング共有メモリシステムは、リング上をメモリ空間が展開しているため、ランダムアクセスができない。すなわち、あるメモリ空間にアクセスする場合、共有バスにおける競合は存在しないが、当該のメモリ空間にアクセスできるまで待つ必要がある。さらに、光リングが広域に展開しているため、従来の共有メモリシステムに比べ、アクセス時間が非常に大きくなる可能性がある。

仮想光リングを共有メモリとして用いる場合、波長パスのうち、計算機ノード間での通信用に光リングの数波を割り当て、残りの波長を共有メモリとして用いる。各ノードが共有メモリのコピーをローカルキャッシュに持つため、前章で述べた従来の共有メモリシステムと同様、キャッシュの整合性の問題が生じる。解決法としてディレクトリ方式とスヌープの手法があるが、本報告では、スヌープの手法を応用して解決する。また、全てのプロセッサがひとつの共有メモリにアクセスするため、書き込みの競合の問題も生じる。前章で述べた従来の共有メモリに対する競合回避の手法を光リング共有メモリに応用する。

仮想光リングによる共有メモリへのアクセス手法は、メモリ競合とキャッシュの整合性を考慮し、無効化型のライトスルーおよびライトバックを用いて実現する。制御メッセージは、

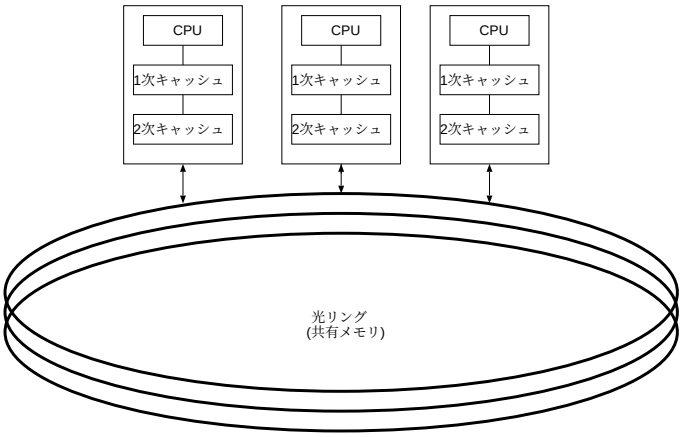

図 5: 共有記憶型モデル

制御用の波長を使って送ることとする。また、同一ブロックに対するロックメッセージを複数受け取った場合は、時間の古いメッセージを有効とする。以下では、それぞれの手法について詳しく述べる。

3.1.1 ライトスルーを用いた場合

図 6、図 7 にライトスルーを用いたモデルを示す。キャッシュ無効化型のライトスルーは、前章に示したように、書きこみが行われる度に他の計算機ノードがローカルキャッシュに持つ該当ブロックのコピーを無効にすることによりキャッシュの内容の一致性を保証する手法である。

まず、プロセッサにデータを取得する際のアクセス手法について述べる。データがローカルキャッシュに存在する場合、リードヒットとなり、該当データをローカルキャッシュから読み込むため、光リング共有メモリへのアクセスは伴わない。データがローカルキャッシュに存在しない場合、リードミスとなり、光リング共有メモリからデータを読み込む必要がある。光リングによる共有メモリの場合、共有バスを持たないため、共有バスにおける競合は発生しないが、共有メモリへのランダムアクセスができず、当該メモリ空間にアクセスできるまで待つ必要がある。すなわち、データ取得の際は、光リング上の該当データにアクセス

するのに平均半周の待ち時間が必要となる。他のノードによるロック要求メッセージを受け取った場合には、もう 1 周待たなければならない。

次に、プロセッサからデータを書き出す際のアクセス手法について述べる。プロセッサで処理したデータがキャッシュに存在する場合、ライトヒットとなり、まず該当ブロックへのロック要求メッセージを光リングの制御用の波長に流す。ロックメッセージは情報として、要求ノード識別子、要求アドレス、要求時間を含むものとする。書き込みの競合が起きなければ、ロック要求をしたプロセッサは 1 周待てば書き込む権利を得る。1 周待っている間に同一アドレスに対するロックメッセージが他のノードから届いた場合は、要求した時間が古い方のロック要求が優先される。書き込む権利を得たノードは、ローカルキャッシュと共有メモリを新しいデータで更新し、該当ブロックへのロック無効化メッセージとキャッシュ無効化要求メッセージを光リングの制御用の波長に流す。このとき、該当アドレスのコピーを持つ他のノードは該当ブロックに書きこみが起こったことを知り、自分がローカルキャッシュに持つコピーを無効にする。

プロセッサで処理したデータがキャッシュに存在しない場合、ライトミスとなり、まず該当ブロックへのロック要求メッセージを光リングの制御用の波長に流す。書き込みの競合が起きなければ、ロック要求をしたプロセッサはロック要求メッセージを送出して 1 周待てば書き込む権利を得る。書き込む権利を得たノードは、光リング上の該当ブロックを新しいデータで更新し、該当ブロックへのロックの無効化メッセージと無効化要求メッセージを光リングの制御用の波長に流す。このとき、該当アドレスのコピーを持つ他のノードは該当ブロックに書きこみが起こったことを知り、自分がローカルキャッシュに持つコピーを無効にする。

3.1.2　ライトバックを用いた場合

図 8 から図 11 にライトバックを用いたモデルを示す。キャッシュ無効化型のライトバックは、前章に示したように、書き込みが行われる度に共有メモリを更新するのではなく、キャッシュの状態に従って共有メモリへの書き込みを行う。他の計算機ノードがローカルキャッシュに該当ブロックのコピーを持つ場合は、それを無効にすることによりキャッシュの内容の一

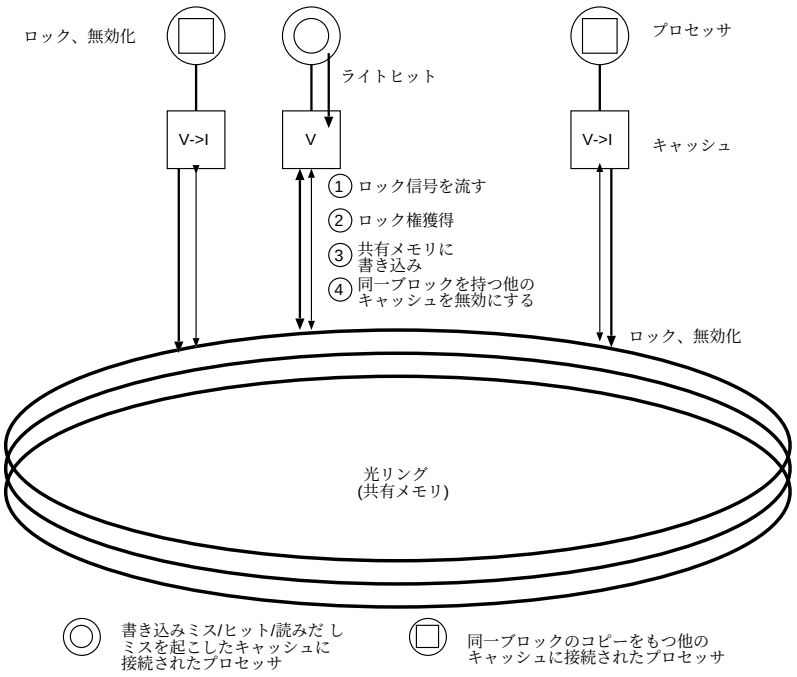

図 6: 無効化型ライトスルーのライトヒット処理

致性を保証する手法である。

各ノードがローカルキャッシュに持つ共有メモリのコピーの状態は、I（無効)、C（共有メモリと一致)、D（共有メモリと不一致）の 3 つである。

これらの状態によって各ノードの動作が大きく異なるため、状態に分けて説明する。

まず、データがキャッシュに存在する場合を考える。データが C 状態でキャッシュに存在し、プロセッサがそのデータを取得する場合、キャッシュ上のデータは主記憶と内容が一致していることを表すため、このデータの取得に関しては光リングへのアクセスを伴わない。

データが C 状態でキャッシュに存在し、プロセッサがそのデータを書き出す場合、キャッシュに対してデータを書き出すとそのデータは C 状態から D 状態になる。続いて該当ブロックへの無効化要求メッセージを光リングの制御用の波長に送出する。このとき、該当アドレスのデータを持つ他のノードは該当ブロックに書き込みが起きたことを知り、そのノードのローカルキャッシュに持つデータを I（無効）状態にする。

次に、データが D 状態でキャッシュに存在する場合を考える。プロセッサがそのデータを

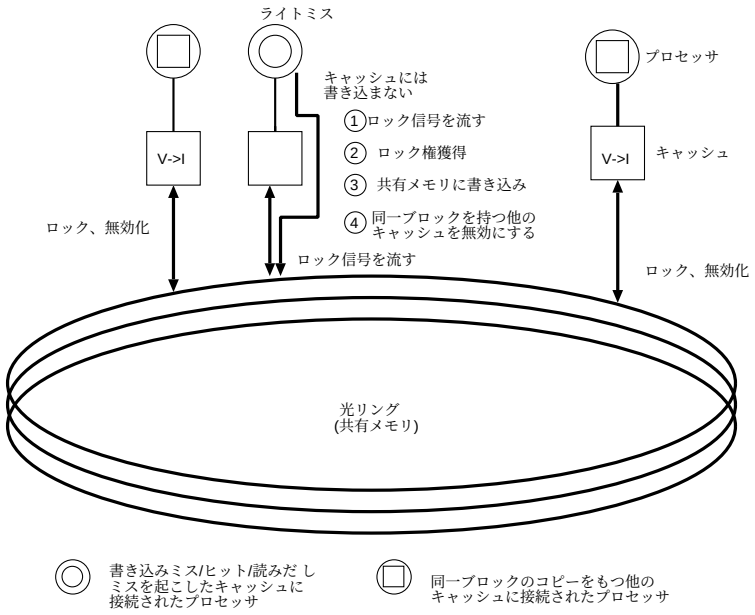

図 7: 無効化型ライトスルーのライトミス処理

取得する場合も書き込む場合も、そのデータは共有メモリとは一致していないが、他のプロセッサにも影響しないため、共有メモリへのアクセスは行わない。

次に、データがキャッシュに存在しない場合を考える。プロセッサにデータを取得する場合、取得したいデータが他のノードのキャッシュの状態によって大きく動作が異なる。取得したいデータが他のノードのキャッシュの C 状態である場合には、共有メモリから取得すればよい。取得したいデータが D 状態の場合、プロセッサは、制御用の波長に他のプロセッサのキャッシュにあるデータの書き戻しを要求するメッセージを送出する。キャッシュに D 状態のコピーを持つノードは共有メモリに対して該当ブロックを書き戻して一致をとる。共有メモリに対する書き込みに関しては、ライトスルーと同様の処理が必要である。書き戻しが完了したら、取得要求を出したプロセッサが、共有メモリから該当ブロックをローカルキャッシュに読み込んで両方のノードのキャッシュのデータは C 状態になる。

プロセッサのデータを書き出す場合、他のノードのキャッシュのデータが D 状態であると、読み込みの場合と同様、D 状態のコピーを持つノードの書き戻しを要求し、次に書き込

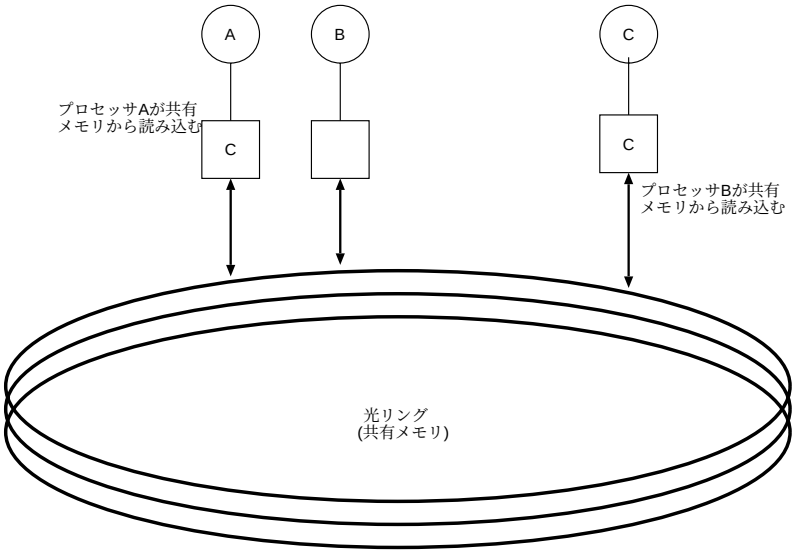

図 8: 無効化型ライトバックのリードミス処理

み要求を出したノードが共有メモリから該当ブロックをローカルキャッシュに読み込む。最後に、書き込み要求を出したノードは該当ブロックを更新し、自分のコピーブロックの状態は D になる。最初に D 状態のデータを保持していたノードのコピーは I 状態になる。

3.2 分散メモリ型モデルにおけるアクセス手法

光リングの分散メモリ型モデルを図 12 に示す。このモデルは、既に CC-NUMA (Cache Coherency NUMA) システムなどで実現されているが、CC-NUMA の場合は各ノード間が結合網で結ばれているため、他ノードのメモリアクセスをしることができない。よって、キャッシュの一貫性を維持するためにディレクトリ方式を採用している。ディレクトリ方式の場合、スヌープ方式と比べるとディレクトリでの一貫性の維持のための処理がオーバーヘッドになる。しかしながら、仮想リングを各ノード間の結合網として置き換えることにより、スヌープ方式を採用できる。よって、各ノードのローカルメモリを統合して共有メモリとして用い、ノード間でのデータの転送、キャッシュコヒーレンシ、ロックのための制御用のデータ転送に仮想リングを用いる。

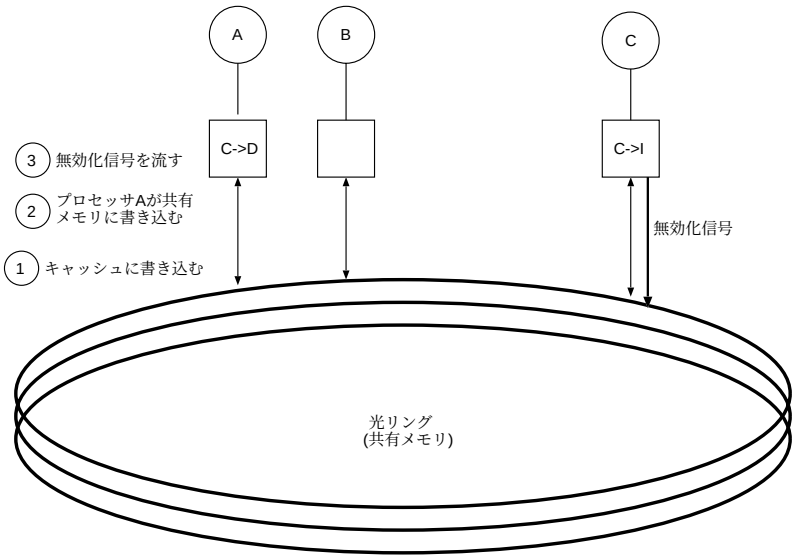

図 9: 無効化型ライトバックのライトヒット処理

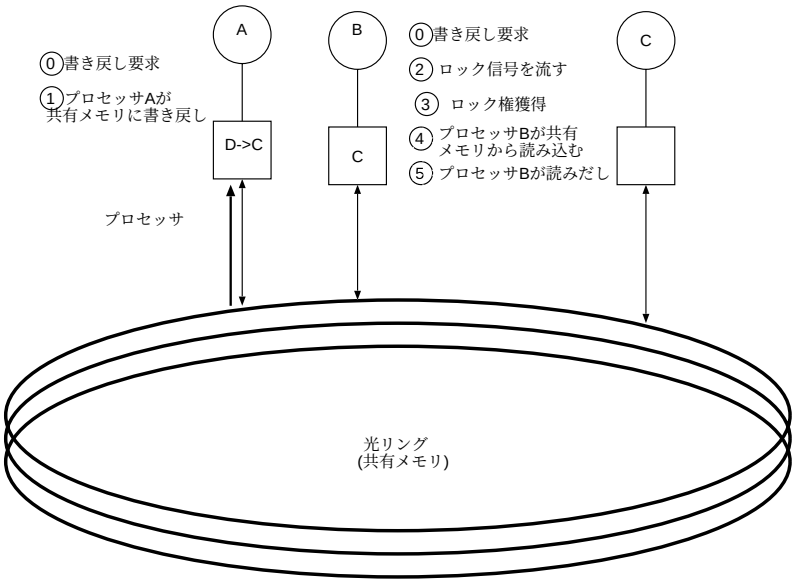

図 10: 無効化型ライトバックの D 状態のブロックへのリードミス

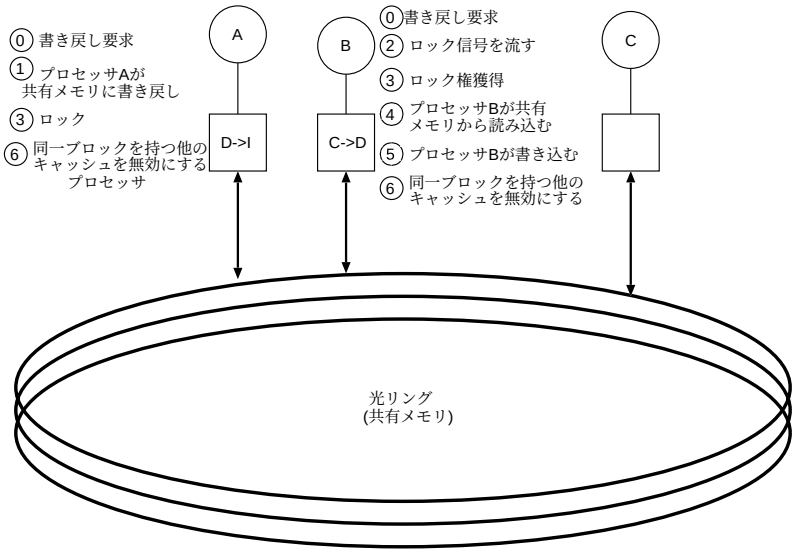

図 11: 無効化型ライトバックの D 状態のブロックへのライトミス

共有メモリ型モデルと異なる点は、書き込み要求をする際に、書き込み要求ノードはロックメッセージを制御用の波長に流し、該当ブロックを所有するホームメモリから書き込み権利を獲得しなければならない点である。ホームメモリはロック要求をしたノードを管理し、順番に書き込み権利をわたす。書き込み権利を獲得したら、ノードは書き込める。

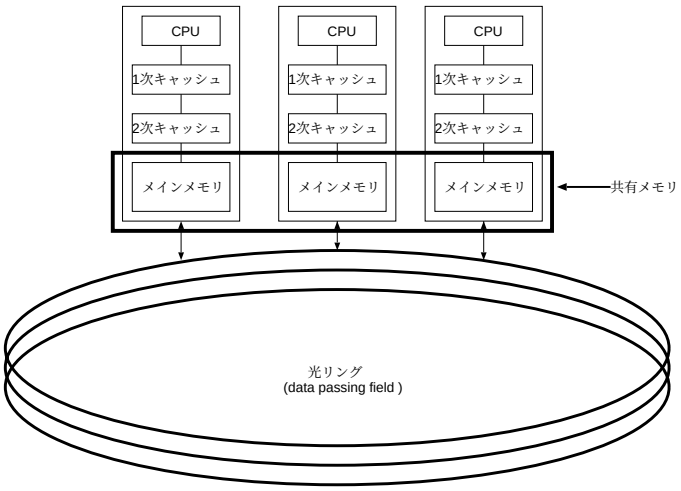

図 12: 分散記憶型モデル

4　提案手法の評価

本章では、3 章で提案した仮想光リングを用いた共有メモリアクセス手法をシミュレーションを用いて評価する。

4.1　シミュレーションモデル

提案したメモリアクセス手法の性能評価に用いたシミュレーションモデルについて説明する。本報告では、節の共有メモリ型モデルを用いる。

4.1.1　プロセッサおよび光リングの仕様

本報告では、光リングを共有メモリとして利用する場合の基本性能をはかるため、各ノードは、プロセッサ 1 つと、キャッシュメモリ 1 つを持つと想定する。プロセッサは、Intel Pentium4 3.06GHz (Northwood processor) を想定する。プロセッサとキャッシュメモリ、光リングの性能に関してはそれぞれ表 1, 表 2 に表す。際のプログラムを実行する際は、命令コードもメモリからデータとして取り込み命令コードをプロセッサに渡して処理を行うが、本報告では、光リング共有メモリの振る舞いに着目しているため、問題を簡単化し、共有メモリにはデータ領域のみを格納するものとして扱う。さらに、通常、想定する Intel Pentium4 プロセッサでは、L1 キャッシュには実行トレースキャッシュとデータキャッシュの 2 種類から構成される。実行トレースキャッシュには、プロセッサが効率よく実行できるマイクロオペレーションが格納され、データキャッシュには、処理すべきデータが格納される。しかしながら、ここでは、実際のプログラムをトレースしたモデル化までは行わないので、L1 キャッシュにはプロセッサ命令を格納し、L2 キャッシュに共有メモリからのデータを格納すると仮定する。また、光リングの距離はパラメータとし、例えば、100km とすると、仮想光リングによる共有メモリの総容量は 640MB、1 周に要する時間は 0.5ms となる。

表 1: プロセッサの性能

コアクロック	3.06GHz
L1 キャッシュサイズ	8KB
L1 キャッシュ帯域	96GB/s
L1 キャッシュクロック	コアクロック
L2 キャッシュサイズ	512KB
L2 キャッシュブロックサイズ	64Byte
L2 キャッシュクロック	コアクロック
L2 キャッシュ帯域	96GB/s
システムバスクロック	533MHz

表 2: 光リングの仕様

光リングの距離	パラメータ
多重波長数	1000 波長
波長あたりの転送速度	10Gbps
総転送速度	10Tbps
光リング上の伝送速度	5μs/km

表 3: 命令と実行ステップ

命令の実行ステップ	RR 形式演算命令	RX 形式演算命令	SI 形式演算命令	ロード命令	ストア命令	分岐命令
1	IF	IF	IF	IF	IF	
2	ID	ID	ID	ID	ID	ID
3	EX	AC	AC	AC	AC	AC
4		AT	AT	AT	AT	
5		OF	OF	OF	MW	
6		EX	即値との論理演算	MDR から GR へ転送		
7			MW			

4.1.2 命令の種類

次に、各ノードが処理する命令として、RR 形式（レジスタ、レジスタ間）演算命令、RX（レジスタ、メモリ間）形式演算命令、SI（メモリ、即値間）形式演算命令、LOAD 命令、STORE 命令、分岐命令の 6 つの命令を用意した。各命令の実行ステップは、以下の表 3 に示す。本報告では、RR 形式演算命令は 3 クロック、RX 形式演算命令は 5 クロックとデータの読みだし時間を合わせた時間、SI 形式演算命令は 5 クロックとデータの読みだし時間、データの書き込み時間を合わせた時間、ロード命令は 5 クロックと読みだし時間を合わせた時間、ストア命令は 4 クロックと書き込み時間を合わせた時間、分岐命令は 3 クロックかかるものとする。各命令の割合は、RR が 30 パーセント、 RX が 20 パーセント、SI が 10 パーセント、ロードが 15 パーセント、ストアが 5 パーセント、分岐命令が 20 パーセントとする。œ. 暦

4.1.3　メモリアクセスパターン

各計算機ノードがアクセスする共有メモリ領域を 40KByte と想定する。40KByte を 1 ブロックを 64Byte として、625 ブロックに分ける。それらのブロックへのアクセスパターンを次の 4 種類としてシミュレーションを行う。

1. ランダムアクセスパターン

   各ノードは、共有メモリ領域からランダムに 1 ブロックを選びアクセスする。

2. 連続アクセスパターン

   各ノードは、共有メモリ領域からランダムに 1 ブロックを選び、そのブロック以降の 25 ブロックに連続アクセスする。25 ブロックに連続アクセスし終わったら、また共有メモリ領域からランダムに 1 ブロックを選び、そのブロック以降の 25 ブロックに連続アクセスすることを繰り返す。

3. 書き込み連続アクセスパターン

   各ノードは、共有メモリ領域からランダムに 1 ブロックを選びアクセスする。命令がストア命令の場合はもう一度同じブロックにアクセスする。

4. 分割領域アクセスパターン

   各ノードごとに共有メモリ領域を分割し、各ノードは分割された共有メモリ領域からランダムに 1 ブロックを選びアクセスする。

4.2　提案手法のシミュレーションによる評価

本節では、4.1 節に示したモデルを用いた提案メモリアクセス手法の評価結果を示す。特に断らない限り、光リングの距離を 100km としている。

まず、図 13 に 1 ノードあたり 10000 命令を実行する場合の 1 命令あたりの実行時間を示す。ここでは、メモリアクセスパターンに連続アクセスパターンを用いている。この図より、ノード数が増加すると、お互いの相互作用により、実行時間が増加していることがわかる。

ライトスルーとライトバックでは、ライトスルーの方がよい性能を示している。ライトバックの長所は、同じブロックに上書きする際に、メモリアクセスを伴わないことである。これはライト命令の割合が少ないのでライトバックの長所が活かせていないためである。

次に、全ノードで実行する命令数を同数にして、各ノードで分散して命令を実行する場合の評価を行う。すなわち、大きな命令群を分割し並列処理を行う場合を想定した。図 19 に、アクセスパターンに連続アクセスを用いたときの全命令が終了するまでの時間を示す。この図より、複数のノードを利用することにより全体の実行時間を減らすことができていることがわかる。図 14 から図 17 に、アクセスパターンを、ランダムアクセス、連続アクセス、書き込み連続アクセス、分割領域アクセスを用いた場合に、総命令数が 1000, 10000 の場合の 1 命令あたりの実行時間を示す。図 17 より、書き込み命令が多いパターンである書き込み連続アクセスパターンの場合のみノード数が低いうちはライトバックが勝つが、ノード数が増大するとライトスルーの方が性能が良くなることがわかる。

次に、光リングの距離を変えた場合の評価を行う。図 18 に、ノード数を 20 に限定し、アクセスパターンに連続アクセスパターンを用いた場合の 1 命令あたりの実行時間を示す。その結果、実行時間は、光リングにおける遅延時間が直接、影響することがわかる。ノード数を増やすと、実行時間はノード間の競合のためにさらに長くなる。

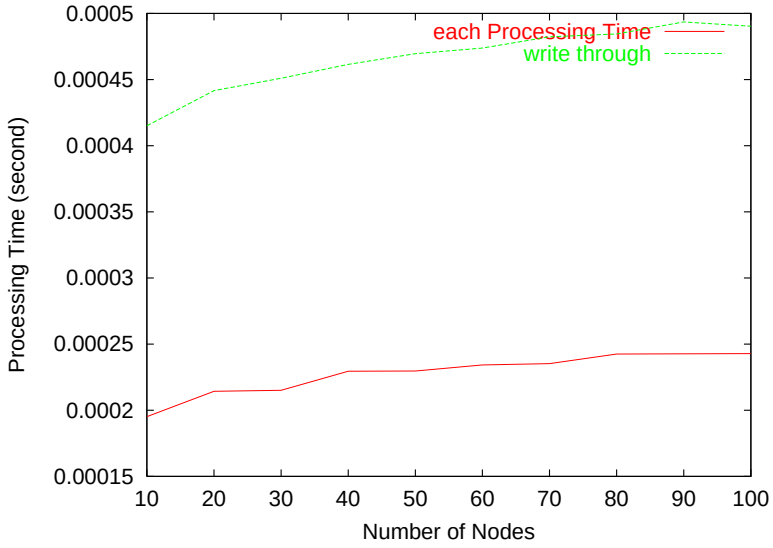

図 13: 1 ノード辺りの命令数固定

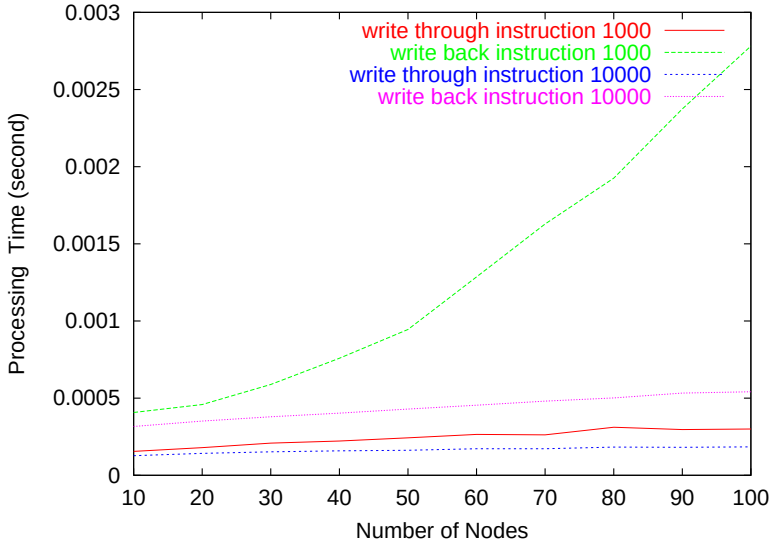

図 14: ランダムアクセスパターン

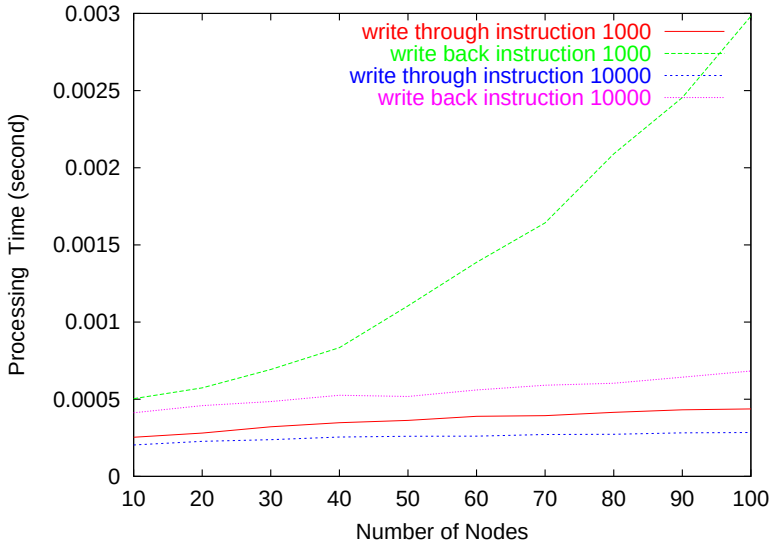

図 15: 連続アクセスパターン

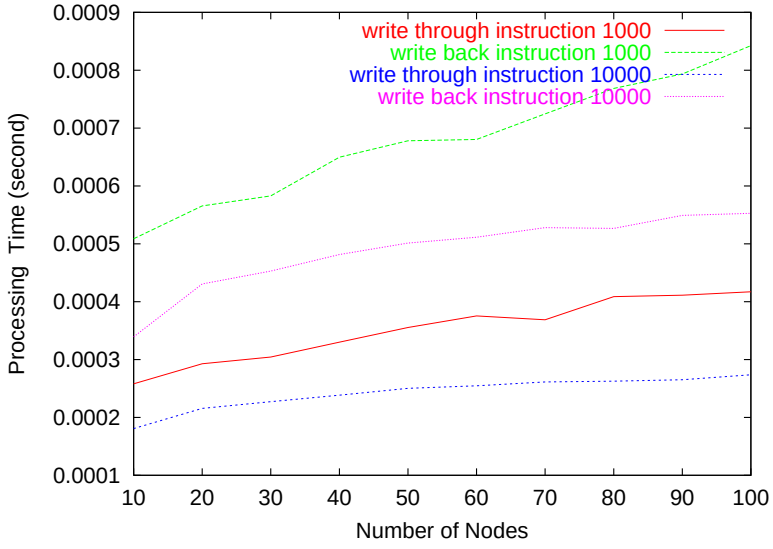

図 16: 分割領域アクセスパターン

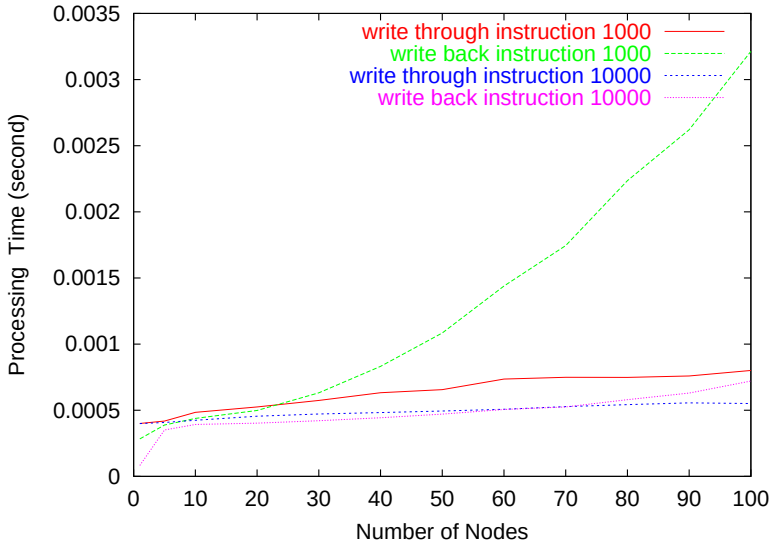

図 17: 書き込み連続アクセスパターン

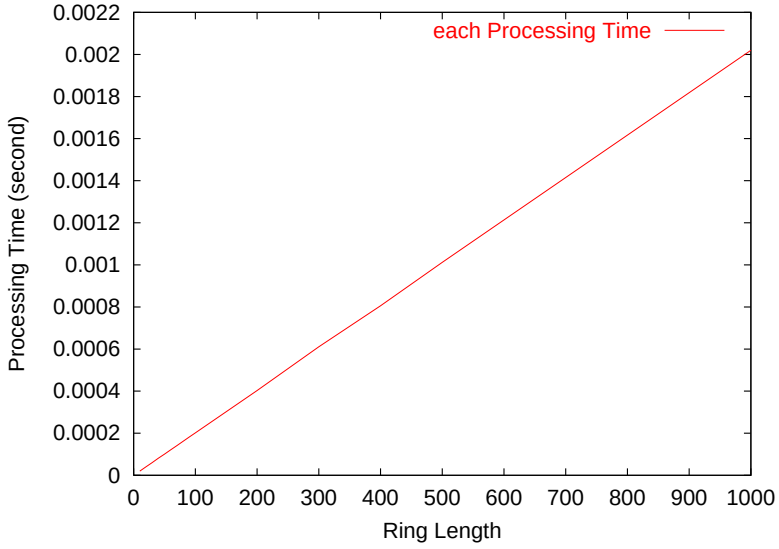

図 18: 光リングの距離を変えた場合の 1 命令の実行時間

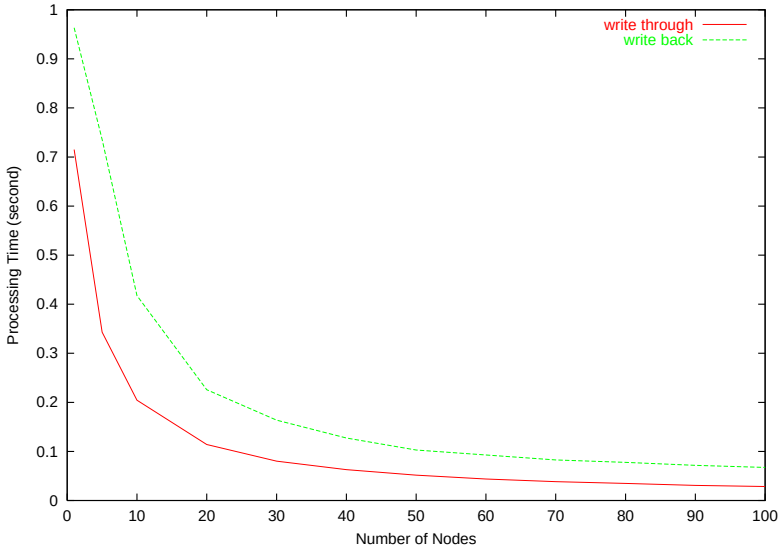

図 19: ライトスルー、ライトバックの計算終了時間 トータル命令 10000

5　終わりに

本稿では、フォトニックグリッド上に仮想リングを構成した際の共有メモリアクセス方式を提案した。共有メモリとして光リングを構成する場合、従来の共有メモリシステム同様に、共有メモリに対する競合とキャッシュの整合性の問題が生じる。また、長距離ファイバ上に展開する仮想光リングへはアクセスのタイミングや頻度に制約がある。そのため、そのことを十分に考慮したアクセス方式を考え、シミュレーションを用いてその性能を評価した。その結果、広域に分散した計算機群で仮想光リングを用いてデータの共有をはかる場合、メモリアクセスの遅延の影響が大きいことから、制御メッセージの交換の少ない方式が特にに有効であることがわかった。

本稿では、ローカルメモリを活用していないが、ローカルメモリを用いることで分散環境を考慮した分散共有メモリも考えられる。また、本稿では各ノードが１つのプロセッサしか持たないことを想定しているが、各ノードをクラスタ化したモデルも考えられる。さらに、本稿では、光リングの全波長にアクセスできると仮定しているが、全波長にアクセスできない場合の性能も測らなければならない。

謝辞

本報告を終えるにあたり、御指導、御教授を頂いた村田正幸教授に深く感謝致します。また、本報告において、終始御指導頂いた大阪大学サイバーメディアセンターの馬場健一助教授に深く感謝致します。並びに適切な助言を頂いた宮原秀夫教授、今瀬研究室の今瀬真教授、大阪府立看護大学医療技術短期大学の菅野正嗣助教授、長谷川剛助教授、宮原研究室の若宮直助教授、今瀬研究室の大崎博之助教授、大阪市立大学の阿多信吾助手、大阪大学大学院経済学研究科の荒川伸一助手、今瀬研究室の多田知正助手、宮原研究室の牧一之進助手に心から感謝致します。最後に、御協力を頂いた村田研究室および宮原研究室の皆様に心からお礼申し上げます。

参考文献

[1] 村田 正幸, 北山 研一, 宮原 秀夫, “1000 波長 WDM によるインターネットのボトルネック解消の 可能性,” **電子情報通信学会技術研究報告**, May 2000.

[2] E. L. Berger, “Generalized multi-protocol label switching (gmpls) signaling functional description,” in *IETF RFC3471*, Jan. 2003.

[3] T. Yamaguchi, K. Baba, M. Murata, and K. Kitayama, “Packet scheduling for WDM fiber delay line buffers in photonic packet switches,” in *Proceedings of OptiComm 2002*, pp. 262–273, July 2002.

[4] K. Baba, R. Takemori, M. Murata, and K. Kitayama, “A packet scheduling algorithm for the 2x2 photonic packet switch with FDL buffers,” in *Proceedings of 28th Europian Conference on Optical Communication 2002 (ECOC2002)*, Sept. 2002.

[5] 天. 英晴, **並列コンピュータ**. 昭晃堂, June 1996.

[6] 鈴木 則久, 清水 茂則, 山内 長承, **共有記憶型並列システムの実際**. コロナ社, Mar. 1993.

[7] 富. 眞治, **並列コンピュータ工学**. 昭晃堂, Aug. 1996.

[8] I. Kaji, Y. Tan, and K. Mori, “Autonomous data synchronization in heterogeneous systems to assure the transaction,” in *Proceedings of 4th IEEE International Symposium on High-Assurance Systems Engineering*, 1999.

[9] S. Sameshima, K. Kawano, J. Kumayama, T. Ito, K. Inoue, and S. Fujishiro, “An autonomous decetralized system architecture and techniques for on-line development and maintenance,” in *Proceedings of ISADS ’97*, Apr. 1997.

[10] H. F. Ahmad and K. Mori, “Autonomous information service system: Basic concepts for evaluation,” in *Proceedings of IEICE TRANS ’2000*, Nov. 2000.

[11] I. Kaji and K. Mori, “Autonomous data consistency technique through fair evaluation among heterogeneous systems,” in *Proceedings of ISDAS ’01*, Mar. 20001.

[12] I. Kaji and K. Mori, “Atomic actions of transaction processing and design technique in heterogeneous autonomous decentralized systems,” in *Proceedings of IEICE TRANS ’2000*, May 2000.