

λ コンピューティング環境における OpenMP ライブラリのためのデータ共有機構の設計

井本 舞[†] 合田 圭吾[†] 馬場 健一^{††} 村田 正幸[†]

[†] 大阪大学 大学院情報科学研究科 〒 565-0871 大阪府吹田市山田丘 1-5

^{††} 大阪大学 サイバーメディアセンター 〒 567-0047 大阪府茨木市美穂ヶ丘 5-1

E-mail: [†]{m-imoto,k-gouda,murata}@ist.osaka-u.ac.jp, ^{††}baba@cmc.osaka-u.ac.jp

あらまし 近年，グリッド計算に関する研究開発が盛んに行われている．現状の計算環境ではノード計算機間の通信に TCP/IP を用いているが，パケット単位のデータ交換ではオーバーヘッドの影響が大きいため大規模計算に必要な性能を得ることは難しい．そこで我々は，各ノード計算機に光ファイバを直結し，各ノード計算機上に存在する共有メモリを波長パスで結ぶことにより，高速計算を可能とする λ コンピューティング環境を提案している．本稿ではその実現形態として WDM 技術に基づく AWG-STAR システムを用いることとし，共有メモリを用いた並列計算プログラミング言語である OpenMP のライブラリのためのデータ共有機構を設計し，AWG-STAR システム上で実装した．また，OpenMP アプリケーションを実行させることでその性能を評価し，従来のクラスタ型による並列計算に対して優位性を示した．

キーワード λ コンピューティング環境，分散並列計算，AWG-STAR，OpenMP ライブラリ，分散共有メモリ

Design of Data Sharing Structure for OpenMP Library in λ Computing Environment

Mai IMOTO[†], Keigo GOUDA[†], Ken-ichi BABA^{††}, and Masayuki MURATA[†]

[†] Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

^{††} Cybermedia Center, Osaka University 5-1 Mihogaoka, Ibaraki, Osaka 567-0047 Japan

E-mail: [†]{m-imoto,k-gouda,murata}@ist.osaka-u.ac.jp, ^{††}baba@cmc.osaka-u.ac.jp

Abstract In recent years, the Grid technology has been studied and developed by many researchers. In the conventional Grid environment, data is changed by using TCP/IP. However, as long as the architecture is based on packet switching, realization of high performance computing is difficult. Thus we propose a new architecture, the λ computing environment. In the λ computing environment, network switches and computing nodes are connected each other with optical fibers, and by establishing optical wavelength path between shared memories on the computing nodes, we can offer high performance computing. In this paper, we establish the λ computing environment using the AWG-STAR system, and design data sharing structure for OpenMP library which is parallel computing programming language utilizing shared memory. Moreover, we evaluate the performance by executing an OpenMP application and show the advantage over the existing parallel computing in PC cluster environment.

Key words λ Computing Environment, distributed parallel computing, AWG-STAR, OpenMP library, distributed shared memory

1. はじめに

近年，グリッドコンピューティングに対する期待が高まり，活発に研究開発が行われている．グリッドコンピューティングと

は，広域に分散した計算機やストレージ，様々なデバイスなどの多くの資源を，ネットワークで接続することでひとつの大規模な仮想計算機として機能させる技術である．このグリッドコンピューティングによって構築された仮想計算機を用いること

により、単体の計算機では有効時間内に解くことが難しい大規模な科学計算を行う、高性能観測機器から発生する大量のデータを高速に処理する、大規模データをリアルタイムに計算しながら可視化を行う、などより高度な処理能力を得ることができると期待されている。我々はこれらの機能のうち、グリッドコンピューティング環境にある複数のノード計算機の CPU を用いて高速に分散並列計算を行うことを目指す。

現在のグリッドコンピューティング環境ではデータ通信に通常 TCP/IP が用いられている。しかし、TCP/IP のようなパケット単位のデータ交換ではパケット処理に要するオーバーヘッドが大きく、大規模な計算で行われる大量のデータ共有やデータ交換を行うには十分な性能を得ることは非常に難しい。さらに輻輳制御による通信帯域の抑制や、パケット損失によるデータの再送などで遅延が生じる。TCP/IP のこのような問題点を解決するため、光通信による高速高信頼な伝送が研究、開発されている。特に光の波長を用いて多重化を行う WDM (Wavelength Division Multiplexing) 技術が研究の中心であり、WDM を利用してインターネットの高速化を目指す IP over WDM ネットワークの開発が進んでいる。さらに、さまざまな光技術を下位レイヤの通信技術とする GMPLS (Generalized Multi-Protocol Label Switching) [1] を用いたルーティングの標準化も IETF によって進められている。しかしながら、そのような技術は現在のインターネットでの利用を前提としている。つまり IP パケットを情報の最小粒度として扱い、ネットワーク上でその IP パケット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困難である。

そこで我々の研究グループでは、各ノード計算機を接続している光ファイバを専用の通信路として利用し、WDM 技術を用いた高速な通信チャネルとして活用する λ コンピューティング環境を提案している [2], [3]。 λ コンピューティング環境上の通信は、TCP/IP ではなく波長パスを利用するため、高速高信頼な通信を実現することができる (図 1)。

本稿では λ コンピューティング環境を構築する 1 つの手段として、日本電信電話株式会社フォトニクス研究所が開発している「情報共有ネットワークシステム (AWG-STAR)」を用いる [4]~[6]。この AWG-STAR システムは、各ノード計算機が波長可変光源を通じて光ファイバにより AWG (Arrayed Waveguide Grating) と呼ばれるルータに接続され、物理的にはスタートポロジを、論理的にはリングトポロジを形成している (図 2)。また、各ノード計算機は共有メモリボードを搭載しており、共有メモリボード上のデータは全ノード計算機で同一のものになるよう設計されている。

本稿では AWG-STAR システムを利用した λ コンピューティング環境において分散並列計算を行うために OpenMP を使用することとし、そのための OpenMP ライブラリおよびデータ共有機構の設計を行った。OpenMP は共有メモリモデルによる並列プログラムの標準規格であり、既存の Fortran や C (C++) の文法の中にあるコメント文や pragma 文を利用して並列化を

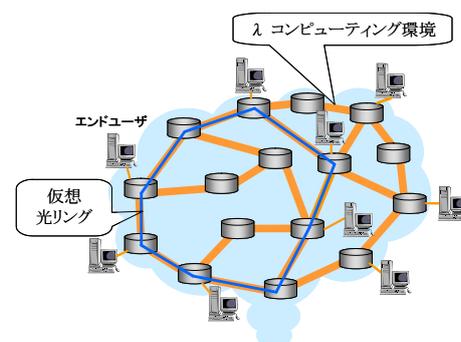


図 1 コンピューティング環境

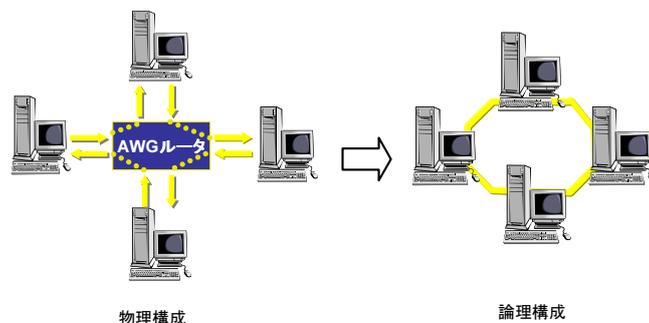


図 2 光リングネットワークの構成

行うプログラミング言語である。複数の CPU がメモリを共有するという点において AWG-STAR システムとの親和性が高いため、AWG-STAR システムの性能がより引き出せると考える。

以下、2 章では本稿で用いた AWG-STAR システムと OpenMP について説明し、3 章で OpenMP コンパイラの設計方法を説明する。4 章では AWG-STAR システムにおける並列計算に必要なライブラリの実装方法について述べ、5 章では構築した λ コンピューティング環境において分散計算を行った場合の性能を評価する。最後に 6 章で本稿についてのまとめと今後の課題について述べる。

2. AWG-STAR による分散並列計算環境の構築

2.1 AWG-STAR システムの概要

AWG-STAR システムは、日本電信電話株式会社フォトニクス研究所により開発されたシステムであり、WDM 技術によるデータ転送と AWG ルータによる波長ルーティング技術によって実現された情報共有ネットワークシステムである [4]~[6]。AWG ルータは波長に基づいたルーティングを行っており、電気信号に変換せず光信号をそのまま処理するため、高速なネットワークを構築することができる。また、AWG-STAR システム上の各ノード計算機は、共有メモリボードを搭載し、共有メモリボード上で同一のデータを保持することでメモリを共有しており、高速な光リングネットワークを利用してデータ交換をリアルタイムに行うことができる。従来のシステムでは、データ共有するためには何らかの明示的なデータ転送が必要であったが、AWG-STAR システムでは共有メモリに書き込まれ

たデータは光リングネットワークに送出され、全ノード計算機の共有メモリが自動的に更新される。従って AWG-STAR システムを用いることにより、共有メモリ上のデータ共有は、共有メモリに書き込む処理によりハードウェアがバックグラウンドで行うため、高速に実行される。他ノード計算機が更新したデータの取得は、AWG-STAR システムを通じて共有メモリに配信され、自動的に更新されるため、自ノード計算機上の共有メモリから読み込むことにより実現できる。

2.2 分散並列計算と OpenMP

分散並列計算のプログラミングモデルは共有メモリの存在によって 2 種類に大別できる。一方は分散メモリプログラミングであり、他方は共有メモリプログラミングである。代表的な分散メモリプログラミングは MPI (Message Passing Interface) であり、プロセス間はメッセージパッシングによってデータ交換をし、同期をとりながら並列計算を行うプログラミングモデルである。そのため、どのタイミングでどのデータのメッセージの交換を行うかということプログラマが定める必要がある。それに対し、共有メモリプログラミングの標準規格である OpenMP は、全ての CPU 間で共有のメモリを持つことを前提としている。共有メモリを用いるため、プログラマが明示的なメッセージ交換の指示をすることなくデータを共有 / 交換できる。我々の研究グループでは AWG-STAR システム上で動作する MPI ライブラリを既に作成しているため [7]、本研究では OpenMP を動作させることを目指した。

OpenMP は元となるプログラミング言語（本稿では C 言語を用いた）に並列化の指示文を加えることによって、コンパイラが並列化を行う。つまり、プログラム内で並列実行の指示文がない箇所は単一のプロセスによる逐次実行を行い、並列実行の指示文の箇所では全プロセスによって並列実行を行うよう、コンパイラが実行環境に適した中間コードを生成する。生成された中間コードを元となるプログラミング言語のコンパイラによって再度コンパイルすることによって、実行可能な機械語が生成される。以降、逐次実行で動作するプロセスをマスタープロセス、それ以外のプロセスをワーカープロセスと呼ぶ。OpenMP はコンパイラがメモリアクセスの順序入れ換えなど高度な最適化をかけられるように、共有メモリの一貫性に対して緩やかな制約をとっている。つまり、並列実行時に常に全プロセスが同一の共有変数に対して同じ値を保持している必要はなく、flush 関数を呼んだ際と並列実行が終了する際にデータの一貫性が保たれていればよい。そのためプログラマは適宜ロック関数による排他的制御によって共有変数のデータ不整合が起きないようにしなければならない。

2.3 OpenMP の AWG-STAR システムへの適用

共有メモリへの書き込みには 2 つの手法が考えられる。ひとつは、共有変数への書き込みがあるたびに AWG-STAR システムの共有メモリに直接書き込む手法である。この場合、毎回の書き込みごとに共有メモリへアクセスするため、共有メモリへのアクセス遅延が大きい状況では好ましくない。しかし、他ノード計算機との共有メモリ一貫性を考えなくてよい場合、共有メモリのデータが自動的に他ノード計算機と同一のデータを



図 3 OpenMP アプリケーション実行のプロトコルスタック

持つ AWG-STAR システムとは相性がよい。

もうひとつの書き込み手法は、書き込むべきデータをローカルメモリへバッファリングしておき、flush 関数を呼び出すタイミングでメモリに書き込むという手法である。この手法は共有メモリへのアクセス回数が少ないため、共有メモリへのアクセス遅延が大きい場合に有効である。ただしノード計算機間の一貫性を保持するための手法を考える必要がある。

本稿では AWG-STAR システムの特性を生かして OpenMP プログラムを動作させるため、前者の共有変数は直接共有メモリに書き込む手法を選んだ。さらに、AWG-STAR システム上で分散並列計算を行う際に必要となる OpenMP ライブラリを作成し、データ共有への設計と実装を行った。

3. OpenMP コンパイラ設計

OpenMP の実現は前述のようにコンパイラによってなされる。AWG-STAR システムに OpenMP を適用するには、AWG-STAR システムのための OpenMP コンパイラ、および生成された中間コードが呼び出すためのランタイムライブラリの作成が必要である。本研究では既存の OpenMP コンパイラ OMPi [8] に基づいて AWG-STAR システム向けコンパイラおよびランタイムライブラリの実装をした。なお、実装の詳細については後日報告する予定である。

4. データ共有機構の設計と実装

図 3 に OpenMP アプリケーションを実行するためのプロトコルスタックを示す。OpenMP コンパイラは AWG-STAR システムが提供する共有メモリを使用するが、それ以外に AWG-STAR システムが提供しないデータ共有機構も必要となる。それは、動的メモリ割り当て機能、ロック機能、バリア同期機能の 3 点である。本章ではその 3 点の設計と実装方法を説明する。

4.1 動的メモリ割り当て機能

OpenMP プログラムの共有変数およびプロセス間制御情報共有のために、共有メモリの動的メモリ割り当て / 解放機能が必要となる。そのため、メモリを割り当てた順番と逆順でメモリを解放するという OpenMP コンパイラの性質を利用し、共有メモリ上にスタックを用意することで動的メモリ割り当てを実装した。つまり、あらかじめ共有メモリ上に割り当てるための領域を確保しておき、その領域の先頭から順にメモリを割り当てていく。このとき領域のどのアドレスまで割り当て済みか

という値を保持しなければならないが、マスタプロセスのみが要求を出すという性質を用いて、マスタプロセスのローカルメモリ内にその値を保持することとした。

4.2 ロック機能

2章で述べたように、プログラマは共有変数の一貫性を保つためにロック機能（排他的制御機能）を用いなければならない。OpenMP ではプログラム内のクリティカルセクションに対して、指示文でクリティカルであるということを指定することによってその箇所が排他制御の対象となる。ただし、プログラム内には複数のクリティカルセクションが存在することもあるため、プログラマは各セクション毎に名前を付けることができる。今回実装したコンパイラはプログラマが付けたクリティカルセクションの名前を、順に 1, 2, ... のように正整数に変換して中間コードを生成する。以降、この正整数をロック ID と呼び、ロック ID によってそれぞれのクリティカルセクションは区別される。

ロック機能を実装するひとつの方法として、共有メモリ上の領域に各ロック ID のロック / アンロックを表すインデックスを設けるという方法がある。つまり、プロセス i がロックしたいときは、まずインデックスにあるロック ID の値が“アンロック”であることを確認し、その後インデックスの値を“プロセス i がロック”に更新することでロック獲得とする方法である。しかし、この方法はロックを獲得するまでのステップがアトミックアクションでないため複数のプロセスが同時にクリティカルセクションを実行する可能性がある。そこで本稿では共有メモリ上でインデックスを設けて、さらにマスタワーカ型によってインデックスを制御する方法をとった。つまり、ワーカプロセス i はまず共有メモリ上のインデックスを確認して対象とするロック ID n が“アンロック”であることを確認する。そしてマスタに対してロック ID n のロックを要求する。要求を受けたマスタは、再度ロック ID n がアンロック状態であることを確認してから、インデックスの値を“プロセス i がロック”に更新し、ワーカにロック獲得の通知をする。このようにマスタがワーカから要求を受け取った順にプロセスへロックを認めるため、複数プロセスが同時にクリティカルセクションに入ることを避けることができる。

この方法を採用するとき、マスタとワーカ間で要求を交換する手段が必要となる。そこで AWG-STAR システムが提供するシグナル機能と、共有メモリ上の新たな情報交換用領域を用意することでその交換手段を実装した。つまり、ワーカプロセス i がマスタにロック要求をだすときは、情報交換用領域の i 番目のブロックにその旨を書き込み、マスタプロセスにシグナルを送る。マスタプロセスはシグナルを受信するとそのシグナルの送信プロセスがわかるため、情報交換用領域の i 番目のブロックから要求を読み取る。また、マスタからワーカに対しての通知はその逆を行う。以上の手順を図 4 に示す。

クリティカルセクションを抜ける際には、アンロックしなければならない。これはロックと同様に、ワーカがマスタに対してアンロックの要求をし、マスタがインデックスを更新するという方法をとることで実現した。さらに、マスタプロセスの

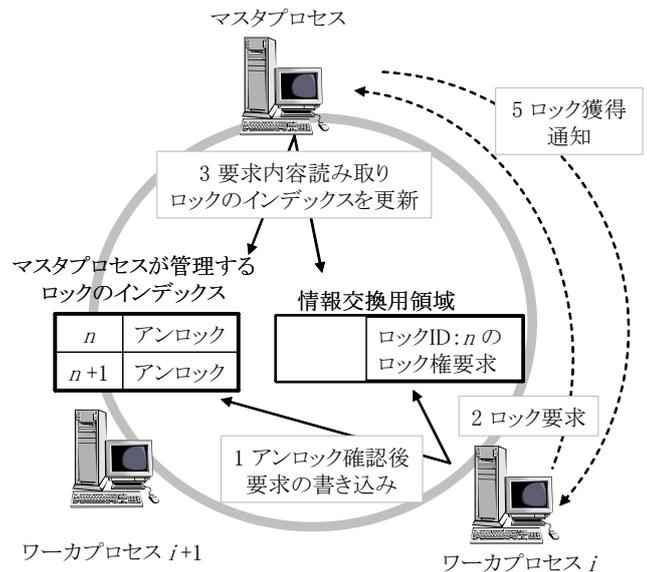


図 4 ロック機能実装方法

ローカルメモリ上でキューを実装することによって、ロックを獲得できずにブロック状態のプロセスの管理をする。この待ち状態のプロセスに対しては、前にロックしていたプロセスがアンロックした後にマスタプロセスがロックが獲得通知を送る。

4.3 バリア同期機能

バリア同期機能とはプログラム内のある箇所において、全プロセスが到達するまで他のプロセスを wait させる機能のことである。この機能はプログラマが明示的に指示することもできるが、並列実行から逐次実行に戻る際などコンパイラによって自動的に中間コードに挿入されることもある。

バリア同期機能はロック機能と同様にマスタワーカ型をとった。マスタに到達したプロセスは情報交換用領域にその旨を伝えてマスタにシグナルを送り、wait 状態に入る。マスタは全てのワーカからのバリア到達のシグナルを受信したあと、全ワーカに対してバリア解除のシグナルを送信する。シグナルを受信したプロセスはプログラム実行を再開させる。

5. OpenMP アプリケーションによる評価

本章では、マンデルブロー集合の計算を OpenMP アプリケーションとして並列化して動作させ、その性能を評価する。ただし、今回の実装ではそれぞれのノード計算機上では一つの OpenMP プロセスを動作させることを前提としている。これは、AWG-STAR システムでは共有メモリボードを複数のアプリケーションやプロセスで共有することを想定していないためである。また、Ethernet を用いた環境で OpenMP を動作させるため、ソフトウェア分散共有メモリ環境を構築し、比較対象とした。具体的には、クラスタ環境 SCORE および SCORE に含まれるソフトウェア分散共有メモリ SCASH を用いた。

5.1 実験システム環境

評価に用いた計算機の仕様を表 1 に、AWG-STAR システムの仕様を表 2 に示す。実験に使用したノード計算機の台数は 1 台から 4 台の範囲で行い、全て同じ性能のノード 計算機を用

表 1 実験に用いた計算機の仕様

CPU	Xeon 2.80 GHz
メインメモリ	SDRAM 512MB
1 次キャッシュ	512KB
2 次キャッシュ	512KB
NIC	Intel PRO/100MT
PCI バス	64 bit/66MHz
PCI 転送速度	533MBytes/sec
OS	Redhat Linux 7.3
コンパイラ	icc 9.1

表 2 AWG-STAR システムの仕様

光インターフェースの伝送速度	2.152Gbps
ノード計算機の 1 回あたりの転送データ量	1KByte
ノード計算機でのフレーム転送処理遅延	500ns
共有メモリへの書き込み	64MBytes/s
共有メモリから読みだし	80MBytes/s

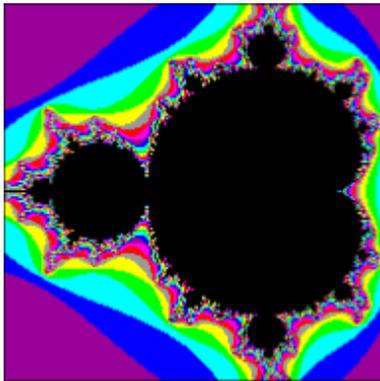


図 5 マンデルブロー集合の図形例

いた．今回の実験では，ノード計算機数に応じて光リングの長さを変えている．具体的には，ノード計算機数を N とすると，光リングネットワークの長さは $10Nm$ としている．SCore を利用する際はネットワークとして 1Gbps の Ethernet を用いている．

5.2 評価に用いるアプリケーションプログラム

マンデルブロー集合とは， $z_0 = 0, z_{n+1} = z_n^2 - c$ ($n = 0, 1, \dots$) で定義される複素数の数列 z_n が有界である ($n \rightarrow \infty$ で $|z_n|$ が発散しない) ような複素数 c の集合のことである．また，その集合を 2 次元平面上にプロットしたものがマンデルブロー集合図形であり (図 5)，一般的に黒い部分が有界である複素数 c の集合を表し，その他の色は無限大に発散する速さを表している．単純な静的分割による並列化処理方式では，指定サイズの複素数平面をプロセス数分の均等固定領域に分割し，それぞれの領域の計算を各プロセスが行う．計算は独立で，複素数平面の範囲を与えるパラメータが分かれば，複素数平面の初期値が必要ないため，最後に 1 回計算結果を回収すればよい．したがって，プログラムの始めと終わりにのみ共有変数を参照するため，共有メモリへのアクセス回数は少ない例として今回のアプリケーションプログラムを用いている．

5.3 プロセス数による評価

複素数集合のデータサイズを $x \times y$ とすると，常に $x = y$ の

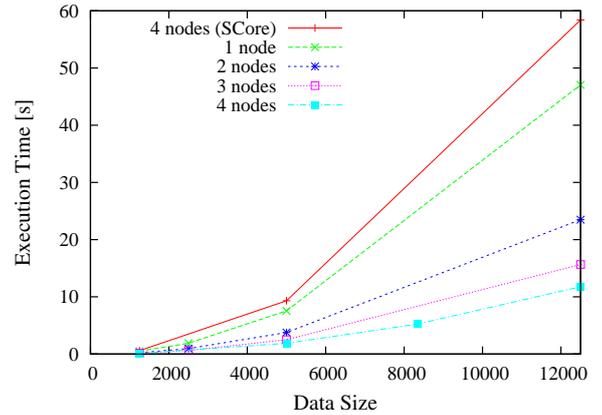


図 6 データサイズ (小) と実行速度

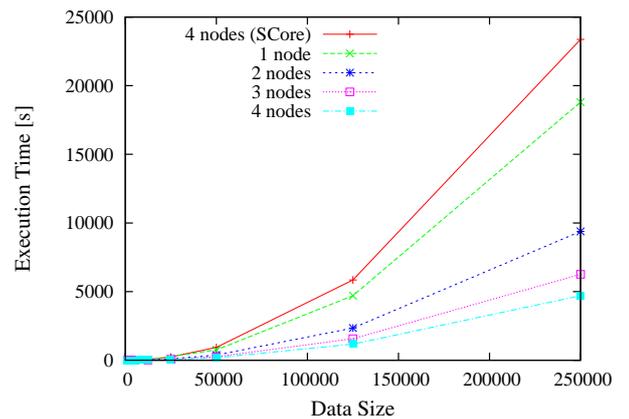


図 7 データサイズ (大) と実行速度

データサイズで計算を行った．図 6 および図 7 にマンデルブロー集合の計算にかかる実行時間を示す．横軸にデータサイズ x をとり，縦軸に実行時間をとっている．図 6，図 7 で示されているように，プロセス数が増えるに従って実行時間が短くなっている．さらに，2 プロセス，3 プロセス，4 プロセスの実行時間は 1 プロセスによる実行時間のほぼ $1/2, 1/3, 1/4$ の実行時間となっており，マンデルブロー集合の計算が OpenMP によって効率よく並列化されていることがわかる．

5.4 Ethernet との比較による評価

Ethernet による OpenMP 動作環境として SCore クラスタを採用した．SCore はクラスタ環境における並列計算のデファクト標準であり [9]，その上で動作するソフトウェア分散共有メモリ SCASH が提供されている．SCASH を共有メモリとして使用する OpenMP コンパイラに Omni/OpenMP [10] があり，本稿のイーサネットによる OpenMP アプリケーションを実行するコンパイラに採用した．

図 6，図 7 に示されているように，SCore 上で実行した OpenMP アプリケーションの実行時間に比べて，AWG-STAR システム上での実行時間は短くなっている．理由として考えられる AWG-STAR システムと SCASH の違いとして，共有メモリをハードウェアまたはソフトウェアのどちらで実現しているかがある．AWG-STAR システムでは共有メモリの一貫性維持をハードウェアレベルで行っており，高速なデータ共有が

行われるのに対し，SCASH はソフトウェアを通じてデータの
一貫性を保っているため遅くなっている．さらに AWG-STAR
システムは下位レイヤに光リングを用いており，その点におい
ても Ethernet を用いている SCASH より性能がよい．

6. おわりに

本稿では， λ コンピューティング環境の実現形態として WDM
技術に基づく AWG-STAR システムを用いることとし，共有
メモリを用いた並列計算プログラミング言語のデファクト標準
である OpenMP のためのデータ共有機構の設計を行った．さら
に，OpenMP アプリケーションを実行することにより，実装
したデータ共有機構が動作することを確認し，並列計算性能を
評価した．その結果，従来のクラスタ型による並列計算に対し
て優位性を示すことができた．

今回の実行した OpenMP アプリケーションは効率よく並列
化が可能であり，共有メモリへのアクセスが少ないという特徴
がある．そのため，頻繁に共有メモリへアクセスする OpenMP
アプリケーションによって，より詳しく性能を調査する必要が
ある．

謝 辞

本研究を進めるにあたり，日本電信電話株式会社フォトク
ス研究所の岡田顕氏，大阪大学大学院情報科学研究科の藤本典
幸助教授に多大なご支援を頂いた．深く謝意を示す．

文 献

- [1] E. L. Berger: “Generalized multi-protocol label switching (GMPLS) signaling functional description”, IETF RFC3471 (2003).
- [2] H. Nakamoto, K. Baba and M. Murata: “Shared memory access method for a λ computing environment”, Proceedings of IFIP Optical Networks and Technologies Conference (OpNeTec), pp. 210–217 (2004).
- [3] E. Taniguchi, K. Baba and M. Murata: “Implementation and Evaluation of Shared Memory System for Establishing λ Computing Environment”, in Proceedings of 10th Opto-Electronics and Communications Conference (OECC2005), 5A2-3, pp. 20–21 (2005).
- [4] Y. Sakai, K. Noguchi, R. Yoshimura, T. Sakamoto, A. Okada and M. Matsuoka: “Management system for full-mesh WDM AWG-STAR network”, 27th European Conference on Optical Communication, 2001, Vol. 3, pp. 264–265 (2001).
- [5] A. Okada, H. Tanobe and M. Matsuoka: “Dynamically reconfigurable real-time information-sharing network system based on a cyclic-frequency AWG and tunable-wavelength lasers”, in Proceedings of ECOC2003 (2003).
- [6] 岡田顕，田野辺博正，松岡茂登： “波長ルーティング技術を用いたダイナミックに再構成可能な情報共有ネットワーク”，電子情報通信学会技術研究報告 (IN2003-332)，第 103 巻，692 号，pp. 423–427 (2004).
- [7] M. Imoto, E. Taniguchi, K. Baba and M. Murata: “Implementation and Evaluation of MPI Library with Globus Toolkit for Establishing λ Computing Environment”, Proceedings of 6th Asia-Pacific Symposium on Information and Telecommunication Technologies, pp. 421–426 (2005).
- [8] V. Dimakopoulos, E. Leontiadis and G. Tzoumas: “A Portable C Compiler for OpenMP V. 2.0”, Proc. of the European Workshop on OpenMP (EWOMP '03), Aachen, Germany, September (2003).
- [9] “PC Cluster Consortium,” available at <http://www.pccluster.org/>.
- [10] “Omni OpenMP Compiler Project,” available at <http://phase.hpcc.jp/Omni/home.ja.html>.