社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

λコンピューティング環境におけるOpenMPライブラリの設計と実装

合田　圭吾† 　井本　舞† 　藤本　典幸† 　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: †{k-gouda,m-imoto,fujimoto,murata}@ist.osaka-u.ac.jp, ††baba@cmc.osaka-u.ac.jp

あらまし　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，WDM 技術に基づく AWG-STAR システムを用いてコンピューティング環境を構築し，共有メモリを用いた並列計算のためのプログラミング API である OpenMP を AWG-STAR システム上に設計・実装した．また，ベンチマークプログラムを利用して共有メモリの性能を評価した．

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，分散共有メモリ

Design and Implementation of OpenMP Library for λ Computing Environment

Keigo GOUDA†, Mai IMOTO†, Noriyuki FUJIMOTO†, 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: †{k-gouda,m-imoto,fujimoto,murata}@ist.osaka-u.ac.jp, ††baba@cmc.osaka-u.ac.jp

Abstract　Our research group has proposed a new high performance computing architecture, which we call the λ computing environment. In the λ computing environment, every computing node has a shared memory connected with the optical wavelength path. In this paper, we establish the λ computing environment using the AWG–STAR system which is based on WDM technology, and we present the design and implementation of OpenMP Parallel Programming API for AWG–STAR system. Furthermore, we evaluate the shared memory system of the AWG–STAR system by using benchmark programs.

Key words　λ Computing Environment, Distributed Parallel Computing, AWG–STAR, OpenMP, 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．

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のパケット処理に要するオーバヘッドが大きい．このため，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が非常に難しい。

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 [1], [2]．λコンピューティング環境上では，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．

本稿ではλコンピューティング環境を構築するひとつの手段として，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 [3]〜[5]．AWG–STAR システムでは，各ノード計算機は共有メモリボードを搭載しており，それらは波長パスにより接続され，共有メモリボード上のデータは波長パスを通じ，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並列プログラミング API である OpenMP [6] を使用することとし，AWG–STAR システム向けの OpenMP 実装を設計し，アプリケーションを実行することによって共有メモリの性能を明らかにする．

2. AWG–STAR システムと OpenMP 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，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 [3]〜[5] ．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，データを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．しかし，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を自動的に更新する．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 OpenMP の概要

並列プログラミング API である OpenMP によるプログラミングは，ソースコード中に OpenMP ディレクティブと呼ばれる指示文を記述することで行う．OpenMP ディレクティブは計算の並列化など，OpenMP の持つ機能の適用箇所をソースコード中で指定する命令である．OpenMP の機能は OpenMP コンパイラによって実現される．このため，プログラマはノード計算機への計算処理の割り当てやノード計算機間の通信など，実際の並列計算プログラムの詳細な動作を直接ソースコード中に記述することなく並列計算プログラムを作成できる．

図 1 に示したのは簡単な OpenMP プログラムのソースコードの例である．2 行目の “pragma”に続く記述が OpenMP ディレクティブであり，この例では 3 行目以降のループ処理部分が並列実行される．並列化指示のない部分は逐次実行される．

OpenMP コンパイラが並列計算プログラムの実行ファイルを生成するために行う処理はおよそ 2 段階に分かれる．まず第 1 段階として，OpenMP コンパイラはソースコード中の OpenMP ディレクティブを解釈し，その内容に基づいて中間コードを生成する．中間コードとは，与えられたソースコードに対して変換処理を適用し，並列処理に必要な通信ライブラリ呼び出しコード等を適切な位置に挿入したものである．続く第 2 段階で中間コードを機械語のプログラムに変換し，実行ファイルを生成する．

```
double pi = 0.0;
#pragma omp parallel reduction(+:pi)
for (i = 0; i < N; i++) {
    double x = (i + 0.5) * w;
    pi += 4.0 / (1.0 + x * x);
}
```

図 1　OpenMP ソースコードの例

このように，OpenMP コンパイラがプログラマによる並列化作業を肩代わりすることによって並列計算プログラムの開発が容易に可能となる．また，並列プログラムのソースコードが特定の計算環境に依存することもない．しかし，OpenMP コンパイラは対象となる並列計算環境のハードウェアや OS，通信ライブラリなどに応じた中間コードを生成する必要があるため，コンパイラは各計算環境に合わせて実装する必要がある．

3. OpenMP コンパイラの設計と実装

λ コンピューティング環境を実現する AWG–STAR システムにおいて OpenMP を実装する場合には，AWG–STAR システムに対応した中間コードを生成する OpenMP コンパイラを設計する必要がある．AWG–STAR システムにおいて OpenMP の実装に必要となる要素は主に以下の 3 項目である．

（1） 並列実行部の複数ノード計算機による実行の実現

（2） データ共有の実現

（3） ロック，およびバリア同期機構の実現

それぞれの要素について，AWG–STAR システム上での実現方法を次節以降に述べる．

3.1 並列実行部の実行方法の設計

AWG–STAR システムにおいて OpenMP を利用するためにはまず，複数のノード計算機を用いた並列処理を実現する必要がある．OpenMP ではプログラム中に逐次実行部と並列実行部が存在するが，分散並列計算環境での OpenMP の実装には逐次実行部の扱いに大別して 2 種類ある．ひとつは逐次実行部を 1 台のノード計算機で実行する方式，もうひとつは逐次実行部も全てのノード計算機で実行する方式である．前者は逐次実行部の計算を実行するノード計算機をマスタ，それ以外のノード計算機をワーカとして明確に区別する．ワーカはマスタからのリクエストを受けて並列実行部を実行する．したがって，常に計算を実行しているのはマスタのみである．対して後者は常時全てのノード計算機が計算を実行する．後者は前者の方式におけるネットワークを介した計算開始リクエストにかかるオーバヘッドを問題視して提案された方式 [7] である．しかし，AWG–STAR システムでは低遅延の波長パスを用いてノード計算機間を接続しているため，このオーバヘッドは大きな問題とはならない．このため，本研究では前者の方式を用いる．

本研究の手法では，マスタが逐次実行部を実行している間，

ワーカは待機している．マスタは逐次実行部の実行を終えるとワーカに並列実行部の実行をリクエストし，全てのノード計算機を用いて並列計算が行われる．並列実行部実行開始のリクエストは後述のバリア同期機能を利用してワーカに伝えられ，TCP/IP などを用いた従来方式のネットワークを別途必要とすることはない．

3.2 データ共有の設計

OpenMP におけるデータ共有とは，ソースコード中の変数の内容を並列計算に用いるノード計算機間で共有することである．例えば図 1 のコードでは変数 *pi* をノード計算機間で共有して計算を行う．

変数の共有は AWG–STAR システムが提供する共有メモリボードを用いて実現する．すなわち，変数のメモリ領域を共有メモリボード上の共有メモリに割り当てるように中間コードを生成する．これにより AWG–STAR システムの機能によって変数の内容が自動的に計算機間で共有される．ただしこの時，プログラム中の変数のうちで計算機間で共有される必要のないものは共有メモリを使わず，計算機のローカルメモリに割り当てるようにする．これは共有する必要のないデータを共有メモリに置く必要がなく，また共有メモリのアクセス速度の影響を受けないためである．変数が共有される必要があるかどうかはソースコード中の OpenMP ディレクティブのパラメータを読み取ることで判断できる．共有する必要のない変数は明示的に非共有指定されたローカル変数および並列実行部でアクセスされないグローバル変数全てである．

3.3 ロックおよびバリア同期機構の設計

並列計算プログラムにおいては，計算中に計算機間で共有されるデータの読み書きはデータの不整合を防ぐため，ロックによって排他制御されねばならない．また，依存関係のある処理の実行順序を保証するために計算機間でタイミングを同期するバリア同期機能も必要である．OpenMP を実現するにはこれらの機能を並列計算プログラムに対して提供する必要がある．

そこで我々は AWG–STAR システムが提供する共有メモリおよびシグナリング機能を用いて独自に同期ライブラリを作成した [8]．この同期ライブラリをコンパイラの生成する中間コードから呼び出すことで OpenMP プログラムにおける計算機間同期を実現する．

4. ベンチマークによる性能評価

OpenMP を用いて記述されたベンチマークプログラムを AWG–STAR システム上で実行し，実装した OpenMP の性能を評価する．AWG–STAR システム向けの OpenMP コンパイラは既存の共有メモリ環境向けコンパイラである OMPi [9] に基づいて実装した．

比較対象としては，Ethernet 環境で OpenMP による並列計算を実現できるクラスタミドルウェア SCore [10] を用いた．SCore は分散メモリ環境で仮想的に共有メモリを実現するソフトウェア分散共有メモリ SCASH を備えており，専用の OpenMP コンパイラである Omni/SCASH [11] が提供されている．

表 1　実験に用いた計算機の仕様

CPU	Xeon 3.06 GHz
メインメモリ	512MB
1 次キャッシュ	20KB
2 次キャッシュ	512KB
NIC	Intel PRO/1000
PCI バス	64 bit/66MHz
PCI 転送速度	533MBytes/s
OS	Redhat Linux 7.3
コンパイラ	GCC 2.96

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速度	64MBytes/s
共有メモリから読みだし速度	80MBytes/s

4.1 実験システム環境

実験システムの計算機仕様および AWG–STAR システムの仕様をそれぞれ表 1 および表 2 に示す．実験に使用した計算機は 1 台から 4 台で，全て同じ性能である．また，今回の実験では AWG–STAR システムの光リングネットワークの長さは計算機台数 N のとき $10N$m としている．

利用したベンチマークプログラムは姫野ベンチマーク [12] および NPB (NAS Parallel Benchmark) 2.3 [13] に含まれる BT, EP である．NPB 2.3 は RWCP によって C 言語に移植されたもの [14] を用いた．問題サイズは姫野ベンチマークにおいては XS ($128 \times 64 \times 64$)，NPB 2.3 では Class W（それぞれ BT : $24 \times 24 \times 24$，EP : 2^{25}）を用いた．姫野ベンチマークおよび NPB 2.3 BT は共有データへのアクセス頻度が比較的高く，NBP 2.3 EP はアクセス頻度が低い特徴がある．

4.2 実験結果と考察

姫野ベンチマークの結果を表 3，NPB 2.3 の BT および EP の結果を表 4 および図 2 に示す．結果を見ると，全体的に SCore の性能が AWG–STAR システムを上回っており，AWG–STAR システムではそれほど高い性能を得られてはいないが，ノード数の増加に伴って性能が向上する傾向は見られる．AWG–STAR システムにおいて全体的に性能が低調である理由としては，共有メモリに対するアクセス性能の差の問題が考えられる．

ソフトウェア分散共有メモリシステムである SCASH は，ローカルメモリの内容をノード計算機間でコピーし合うことで共有メモリを実現する．具体的には，あるノード計算機がメモリ上のデータにアクセスした際，そのデータが他のノード計算機により更新されていたならば最新のデータをノード間通信によって取得し，ローカルメモリにコピーする．このような動作により SCASH はノード計算機間のデータの一貫性を保つ．データの取得には時間がかかるが，取得済みのデータへのアクセスは高速である．

対して AWG–STAR システムの共有メモリにおいては共有メモリに書き込んだ内容はハードウェア的に同一に保たれる．し

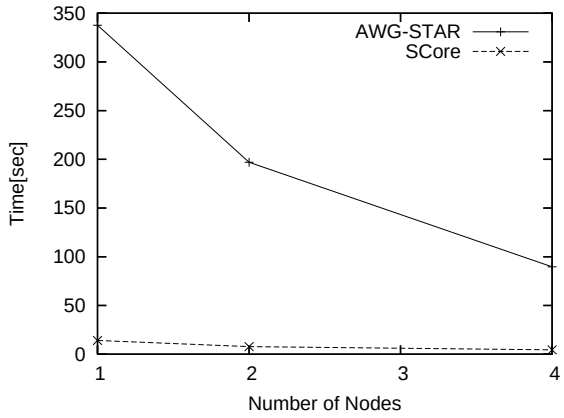

図 2　NPB 2.3 EP Class W の実行時間

かし，このメモリは CPU からは PCI バスを通してアクセスする必要があり，現状の AWG–STAR システムにおいては高速に読み書きできない．CPU からローカルメモリへのアクセスは約 2GB/s で可能であるが，共有メモリへのアクセスは約 80MB/s となる．共有データへの読み書きは全て共有メモリボード上のメモリに対して行われるため，全てのアクセスが低速となっていると考えられる．

従って，AWG–STAR システムにおいてはメモリ性能がボトルネックとなり，全体的な計算性能に影響を与えていると考えられる．実験結果を見ると，NPB 2.3 EP では SCore との性能差は 4 ノードの場合で約 20 倍であったが，NPB 2.3 BT では約 40 倍となっている．共有データへのアクセス頻度の高いベンチマークでは頻度の低いベンチマークと比べて SCore との性能差が大きくなっていることが分かる．姫野ベンチマークでは性能差が縮小するが，これは個々のノード計算機がアクセスするメモリ領域が広範囲に及ぶ，SCore にとって不利なベンチマークであるためと考えられる．

表 3　姫野ベンチマークの実行結果

環境	ノード数	MFLOPS	性能向上率
SCore	1	393.9016	1.000
	2	5.8346	0.015
	4	4.0228	0.010
AWG-STAR	1	0.1074	1.000
	2	0.2073	1.930
	4	0.3760	3.501

表 4　NPB 2.3 BT Class W の実行時間

環境	ノード数	実行時間（秒）	性能向上率
SCore	1	25	1.00
	2	357	0.07
	4	430	0.06
AWG–STAR	1	47930	1.00
	2	27322	1.75
	4	17172	2.79

5. 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 AWG–STAR システムを用いることとし，並列プログラミング API である OpenMP の AWG–STAR システム上への実装と，ベンチマークプログラムによる性能評価を行った．その結果，現状の AWG–STAR システムでは既存の計算環境と同等の性能を達成することはできなかった．しかし，現在開発中の次期 AWG–STAR システムではメモリアクセス性能が改善する見込みがあり，次期システム上で OpenMP を利用すれば十分な性能を達成できる可能性がある．

また，現状は OpenMP の仕様に含まれる全ての機能を AWG–STAR システム上に実装しているわけではないため，それらの機能を利用できるようにしていくことも課題である．

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ニクス研究所の岡田顕氏に多大なご支援を頂いた．深く謝意を示す．

文　献

[1] H. Nakamoto, K. Baba and M. Murata: “Shared memory access method for a λ computing environment”, in Proc. of IFIP Optical Networks and Technologies Conference (OpNeTec), pp. 210–217 (2004).

[2] E. Taniguchi, K. Baba and M. Murata: “Implementation and Evaluation of Shared Memory System for Establishing λ Computing Environment”, in Proc. of OECC2005, 5A2-3, pp. 20–21 (2005).

[3] Y. Sakai, K. Noguchi, R. Yoshimura, T. Sakamoto, A. Okada and M. Matsuoka: “Management system for full-mesh WDM AWG–STAR network”, in Proc. of ECOC2001, Vol. 3, pp. 264–265 (2001).

[4] 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 Proc. of ECOC2003 (2003).

[5] 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).

[6] OpenMP Architecture Review Board: “OpenMP Application Program Interface Version 2.5” (2005).

[7] S. J. Min, A. Basumallik and R. Eigenmann: “Optimizing OpenMP Programs on Software Distributed Shared Memory Systems”, International Journal of Parallel Programming, **31**, 3, pp. 225–249 (2003).

[8] 井本 舞, 合田 圭吾, 馬場 健一, 村田正幸 : “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”, 電子情報通信学会技術研究報告 (PN2006-28), 第 106 巻, pp. 19–24 (2006).

[9] V. Dimakopoulos, E. Leontiadis and G. Tzoumas: “A Portable C Compiler for OpenMP V. 2.0”, in Proc. of the European Workshop on OpenMP (EWOMP’ 03), September (2003).

[10] “PC Cluster Consortium,” available at `http://www.pccluster.org/`.

[11] M. Sato, H. Harada and A. Hasegawa: “Cluster-enabled OpenMP: An OpenMP compiler for the SCASH software distributed shared memory system”, Scientific Programming, **9**, 2, pp. 123–130 (2001).

[12] “Himeno Benchmark,” available at `http://accc.riken.jp/HPC/HimenoBMT/`.

[13] “NAS Parallel Benchmarks,” available at `http://www.nas.nasa.gov/Resources/Software/npb.html`.

[14] “OpenMP C versions of NPB2.3,” available at `http://phase.hpcc.jp/Omni/benchmarks/NPB/`.