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

大阪大学 村田研究室
博士前期過程 2年 井本 舞

発表内容

- 研究の背景
- λコンピューティング環境の提案
  - AWG-STAR システム
  - 並列計算言語 OpenMP
- 実装したデータ共有機構
- 性能評価
- まとめと今後の課題

研究の背景

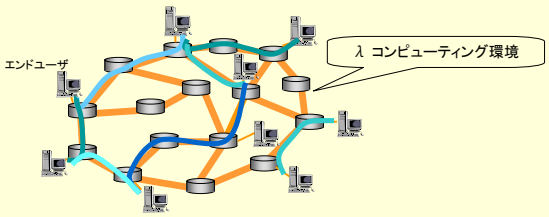
- グリッドコンピューティング
  - ネットワークを介して複数の計算機を接続し，計算資源，ストレージを共有
    - 広域で大規模な計算
    - 大容量データの転送
- 通信オーバーヘッドが問題
  - TCP/IP が通信に使われる
    - パケット処理によるオーバーヘッド
    - パケットロスによる再送遅延
    - 輻輳制御による転送レート低下

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新たな技術が必要

λコンピューティング環境の提案

- 計算機，ルータを光ファイバで接続する
- 波長パスを張り，波長パスを通信の最小粒度とする

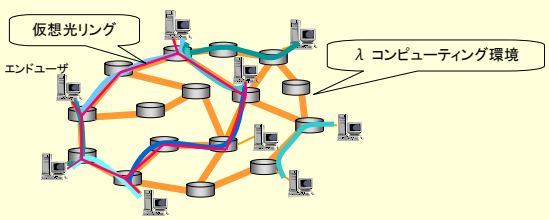
波長パスを専用線として用いる



λコンピューティング環境の提案

- 計算機，ルータを光ファイバで接続する
- 波長パスを張り，波長パスを通信の最小粒度とする

波長パスをつないで仮想光リングを構成

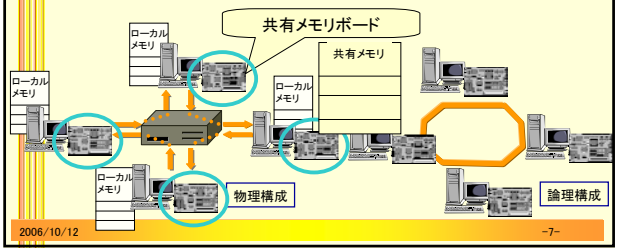


研究の目的

- λコンピューティング環境において高速な並列計算環境を実現する
  - NTT フォトニクス研究所が開発した AWG-STAR システムを利用
- AWG-STAR システムの共有メモリの特性を活用し，並列計算言語 OpenMP ライブラリを実装し，評価する
  - OpenMP コンパイラ・ライブラリの実装
  - OpenMP ライブラリが必要とするデータ共有機構の実装

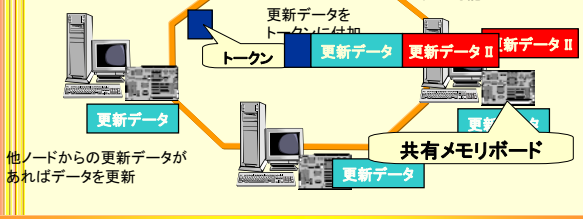
AWG-STAR システム

- 各ノードは波長ルータ（AWG）に接続し光リングネットワークを構成
- 各ノードは共有メモリボードを搭載
  - 各共有メモリボードは同じデータを保持



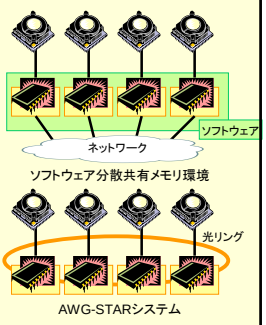
AWG-STARにおけるデータ共有手法

- 書き込みの際、自ノードのメモリボードへアクセスする時間と、トークンが光リングを一周する時間がかかることを考慮
  - 自ノードの共有メモリボードから読み込むだけで共有データの取得ができる



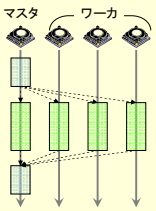
分散並列計算環境でのデータ共有

- 分散並列計算環境でどのように計算データを共有するか
  - 従来手法
    - ソフトウェア分散共有メモリ
      - ソフトウェアで仮想的に共有メモリを実現
      - データ共有のための通信を隠蔽，自動化
      - 性能はあまり良くない
  - λコンピューティング環境
    - AWG-STARシステムが提供する共有メモリを利用
      - 光リングを用いた高速な共有メモリ



並列計算言語 OpenMP

- 複数プロセス間で共有メモリをもつことを前提とした並列計算用プログラミング言語
  - 1つのマスタプロセスと複数のワーカプロセスで実行
  - 通常のプログラム中の並列処理可能な部分をディレクティブで指定する



OpenMP ソースコード

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


OpenMP の実現

- OpenMP コンパイラが実行環境に適した中間コード生成
  - AWG-STAR を共有メモリとして活用
  - 1ノード上で1プロセスを動作させることとする
- OpenMP ライブラリのために必要なデータ共有機構
  - 動的なメモリ割り当て
  - ロック機能
  - バリア同期機能

動的メモリ割り当て機能

- OpenMP プログラム内共有変数のための領域をAWG-STAR 共有メモリ上に動的に割り当てる機能
- OpenMP プログラムとコンパイラの性質利用
  - メモリを割り当てた順序と逆順でメモリを開放する
  - ⇒共有メモリ内に動的割り当て用スタックを作成
  - マスタプロセスのみがメモリ割り当て/開放要求を出す
  - ⇒スタックの最後尾の値はマスタプロセスのローカルメモリに保持

ロック制御機能

- プログラマが明示的にロック位置を指定
- 共有メモリプログラミングでロック制御が必要な理由
  - OpenMPでは共有メモリに対して緩やかな一貫性制御を規定
    - 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い
    - 明示的に flush 関数を呼んだ際と，並列計算終了時にデータの一貫性が保たれればよい
  - AWG-STAR でデータの一貫性がとれるまでトークンが光リング一周する時間がかかる

ロック制御機能の設計 (1/2)

- それぞれのクリティカルセクションに対して LockID をつける
- 共有メモリ上のインデックスでロック/アンロックを管理
  - マスタのみがインデックスを更新

ロックのインデックス

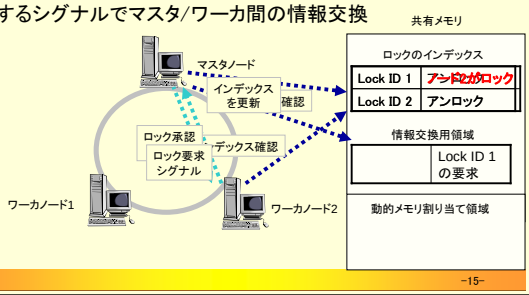
Lock ID	ロック/アンロック
1	ノード1がロック
2	ノード2がロック
3	アンロック

OpenMP コード

```
非クリティカル
セクション
_omp_lock (LockID: 1)
クリティカル
セクション
_omp_unlock
非クリティカル
セクション
_omp_lock (LockID: 2)
クリティカル
セクション
_omp_unlock
```

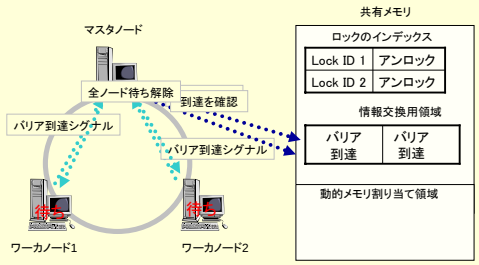

ロック制御機能の設計 (2/2)

- ロックするためにワーカはマスタにロックを要求
  - マスタに要求届いた順にロックする
- 共有メモリに確保した情報交換領域と AWG-STAR が提供するシグナルでマスタ/ワーカ間の情報交換

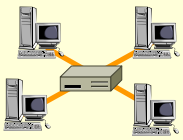


バリア同期機能

- コード上のある定められたポイント（バリア）に全てのプロセスが到達するまで，先に到達したプロセスを待たせる機能

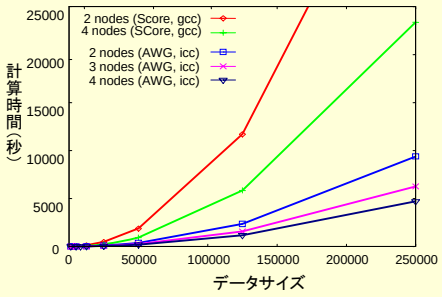


性能評価



- 実験環境
  - CPU: Intel Xeon (3.06GHz)×1
  - ノード数: 4台
  - AWG-STAR 共有メモリボード
    - 光インターフェース速度: 2.152Gbps
    - 共有メモリアクセス：書き込み64MB/s，読み込み80MB/s
- 性能比較対象：SCore
  - Linux用クラスタミドルウェア
  - 専用のOpenMP実装 Omni/SCASH を備える
    - Gigabit Ethernet + ソフトウェア分散共有メモリ
      - 無償配布されていることから OpenMP 関連の研究で性能比較実験によく利用される
- 簡単なベンチマークプログラムによる評価
  - マンデルブロー集合計算
    - 共有メモリアクセス，同期処理の発生頻度：低
    - 並列処理によって性能が向上しやすい

マンデルブロー集合計算



- 計算時間はプロセス数に反比例
- AWG-STARはSCore に比べて高速

まとめと今後の課題

- AWG-STAR を用いてλコンピューティング環境を構築した
- OpenMP アプリケーションが動作するために必要な AWG-STAR のデータ共有機構の設計と実装を行った
- ベンチマークで性能を評価をした
  - 既存の代表的な分散並列処理向け OpenMP 実装よりも高い性能を達成
- 今後の課題
  - 性質の違う OpenMP プログラムで性能を評価する