プロセス並列における OpenMP

5K−6

山中 栄次　岩下 英俊　堀田 耕一郎
APC研究体　富士通(株)ソフトウェア事業本部

1 はじめに

並列プログラムの移植性の問題を解決するために提案された OpenMP は、そのシンプルで実用的な仕様から、現在では共有メモリ型並列計算機においては、デファクトスタンダードとしての地位を確立している。

OpenMP の普及に伴い、OpenMP の適用範囲を共有メモリ型並列計算機から拡大することをねらった研究が行われている。

本報告も、OpenMP の適用範囲の拡大をねらったものである。我々は、現在の OpenMP が、スレッド並列より適用範囲の広いプロセス並列に対して、仕様として、どこまで適用可能か、という観点で調査を行った。

2 OpenMP の概要

OpenMP には、Fortran 向けと C/C++ 向けの仕様が存在する。本報告では、Fortran 向けの仕様を調査の対象とする。

OpenMP は、共有メモリ並列計算機向けのデータ並列型の並列拡張言語仕様であり、共有メモリ空間とスレッドをベースとした仕様である。

OpenMP の仕様は、以下の 4 種類に大別される。

・parallel region 構文
マルチスレッドで実行する範囲を指示する構文。
parallel と end parallel の間がマルチスレッドで実行される。parallel region は、ネストさせることが可能な仕様となっているが、ネストの実装はベンダに任されている。

・Work-sharing 構文
実際の処理の並列化を指示する構文。do ループを doall 的に並列化する do 構文、プログラムブロックを並列化する sections 構文、プログラムブロックの一スレッドでの実行を指示する single 構文、配列演算を並列化する workshare 構文で構成される。Work-sharing 構文は、外側に結合できる parallel region 構文が必要なため、Work-sharing 構文だけのネストは不可能となっている。

・同期構文
スレッド間の同期処理に関連する構文。マスタースレッドだけによる実行を指示する master 構文、クリティカルセクションを指示する critical 構文／ atomic 構文、バリア同期を指示する barrier 構文、メモリの一貫性を指示する flush 構文、do ループの逐次化を指示する ordered 構文で構成される。

・データ環境構文
データ環境を制御するための構文。共通ブロックを各スレッド毎に確保することを指示する threadprivate 構文、データスコープを指示する private ／ shared 節、private なデータの値の初期化を指示する firstprivate ／ copyin 節、private なデータの値の反映を指示する lastprivate 節、リダクション演算の並列化を指示する reduction 節で構成される。節は、他の構文の要素となるもので、他の構文の機能を修飾する。

3 OpenMP のプロセス並列での実現

3.1 目的

OpenMP は、共有メモリ空間とスレッドをベースとしていることから判るように、プロセス内のマルチスレッド機構を念頭においた仕様であると言える。このために、プロセス内マルチスレッド機構を使用して実現するのが、一般的な実現方法となっている。

しかし、スレッドを、制御の流れという抽象的な意味でのスレッドと捉えれば、プロセス間共有メモリ機能を用いることにより、プロセス並列をベースとしても、OpenMP をインプリメントできる可能性がある。

本報告の目的は、OpenMP が、プロセス並列をベースとした場合にインプリメント可能か、という点について、最も一般的なプロス間共有メモリ機構である、UNIX のプロセス間共有メモリ機構の仕様をベースとして、言語仕様的な面から調査を行うことである。

Implementation of OpenMP on Parallel Process Environment.
Eiji YAMANAKA Hidetoshi IWASHITA Kohichiro HOTTA
APC Project Software Grou p Fujitsu Ltd.

3.2 制御系の仕様

スレッドの制御や、処理の分割、同期処理、private なデータの値の初期化／反映、リダクション演算処理など、制御系の処理については、プロセス並列においても、プロセス内マルチスレッド機構と同様な実行論理で、実現できることを確認した。

OpenMP の制御系の仕様は、スレッド間の相互作業を処理するための機構が存在すれば、実現可能な仕様となっている。スレッド間の相互作業を処理するための機構としては、UNIX のプロセス間共有メモリ機構で十分である。

3.3 データスコープ系の仕様

shared 変数、private 変数、threadprivate 共通ブロックなどのデータスコープ系の仕様は、プロセス並列においても、一部を除いて UNIX のプロセス間共有メモリ機構を用いることで、実現できることを確認した。

private 変数、threadprivate 共通ブロックなどの private 系のデータスコープは、プロセス並列の場合、基本的にデータ空間がプロセスに private なため、プロセス内のマルチスレッド機構をベースとする場合よりも、容易に実現できる。

一方、shared 変数の実現は、プロセス内のマルチスレッド機構をベースとする場合よりも、特別な考慮が必要となる。

shared 変数に関するほとんどの仕様は、UNIX のプロセス間共有メモリ機構を用いることで、プロセス並列で実現可能であるが、parallel region がネストした場合、親 parallel region の private 変数を、子 parallel region で、shared 変数とする場合について、UNIX のプロセス間共有メモリ機構では、インプリメントが困難であることが判明した。図 1 に示すように、親側で private とした領域を、子側で、shared として扱わなければならないためである。

この場合のインプリメントが困難なのは、現在の UNIX のプロセス間共有メモリ機構の仕様では、プロセス内のデータ域として使用している通常の領域を、共有メモリ化することはできないことによる。

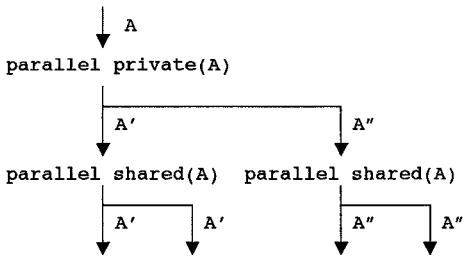

図 1 ネストした parallel region

4 おわりに

本報告では、OpenMP のプロセス並列をベースした実現について、その可能性と問題点について報告した。

OpenMP をプロセス並列で実現する場合、制御系の仕様については、スレッド並列での実現と、ほぼ同等のロジックで実現可能との結論を得た。

一方、宣言系の仕様については、メモリ空間の構造の違いにも関わらず、一部の仕様を除いて、ほとんどの仕様が、UNIX のプロセス間共有メモリ機構で、実現可能であるとの結論を得た。

今回の調査を通して、OpenMP がプロセス並列でも、ほぼ実現可能なことが確認できた。これは、OpenMP が適用できるアーキテクチャの拡大の可能性を示すものである。

今後は、問題となった仕様について、問題を解決する機構を検討していきたい。

参考文献

[1] OpenMP Architecture Review Board. OpenMP Fortran Application Program Interface Version 2.0, November 2000.

[2] H.Lu, Y.C.Hu and W.Zwaenepel. OpenMP on Network of Workstations, In Supercomputing'98, Nov. 1998.

[3] M.Sato, S.Satoh, K.Kusano and Y.Tanaka. Design of OpenMP Compiler for an SMP Cluster. In EWOMP'99, Sep. 1999.

[4] 佐藤 三久, 原田 浩, 長谷川 篤史, 石川 裕, Cluster-enabled OpenMP: ソフトウエア分散共有メモリシステム SCASH 上の OpenMP コンパイラ, 並列処理シンポジウム JSPP2001, Jun 2001.