メタプロセスモデルの共有メモリマシンへの適用

渡辺　義人　　片野　真吾　　緑川　博子　　飯塚　肇
成蹊大学　工学部

1　はじめに

筆者らはメタプロセスモデルとそれを実現する言語 MpC 言語を提案，開発してきた[1][2]．従来はクラスタなどを対象に並列実行単位をプロセスとしてきた．しかし，共有メモリ並列マシンを対象にする場合には実行単位をスレッドとする方が，メモリアクセスの効率化や実行速度の向上が期待できる．本報告では，メタプロセスモデルのスレッド実装方式を報告する．

2　メタプロセスモデルと MpC 言語

メタプロセスモデルとは，従来の共有メモリモデルに，NUMA マシンなどにも対応できるような共有データの階層構造を取り入れたプログラミングモデルである．メタプロセスとは，１つの応用を協力して並列処理する複数のプロセス全体を指し，ユーザにとっての実行単位である．図１はメタプロセスで扱うデータ階層を示し，図２はクラスタにおけるメタプロセスの実行の様子を示す．

MpC 言語は shared というデータ型を新しく導入した C 言語の拡張である．すでに MpC 言語で記述されたプログラムをソフトウェア DSM（SDSM）向けの C プログラムに変換するトランスレータが開発されており，メタプロセスを SDSM を用いたクラスタで実行できる．

3　スレッドによる実装方式

本報告では，SPMD 型の MpC プログラムを対象に pthread による実装を構築した．

3.1　実行単位の対応関係

表１で示すように，従来のメタプロセスモデルはメタプロセスをプロセス群で，並列実行単位を１プロセスで実装していた．今回の実装では共有メモリ並列マシン向けにメタプロセスを

Thread Implementaion of Meta Process Model for Shared Memory Machines
Yoshihito Watanabe, Shingo Katano, Hiroko Midorikawa, Hajime Iizuka
Faculty of Engineering, Seikei University

プロセスで，並列実行単位を１スレッドで実装する．

3.2　データ階層の対応関係

表２のように，SDSM 向けの実装ではメタプロセスモデルの shared は DSM の機能を利用したプロセス間で共有される変数に変換し，global と local はそれぞれ通常のグローバル変数，ローカル変数にしていた．それに対し，今回のスレッド実装では shared は通常のグローバル変数へ，global と local はローカル変数にする．しかし，global の変数についてはスコープがスレッド局所になるように実装する．

4　実装

MpC 言語で記述されたプログラムを C プログラムに変換するトランスレータには，以下の機能が必要になる．

①並列実行単位をスレッドにする．
②メタプロセスモデルの階層データを表２のようにスレッド向けの実装にあわせる．

①については，ユーザの main 関数をスレッド実行のためのサブルーチンに変換し，スレッド生成と同期変数の初期化を行う main 関数を新たに付け加える．②については，メタプロセスモデルで global に対応する変数をユーザプログラムから抽出し，スレッド局所変数に置きかえる．

さらに，今回の実装のために pthread による MpC 標準ライブラリ関数（表３）を新たに作成した．

5　実行結果

評価は表４の SMP 環境で行った．今回作成したライブラリの基本性能を表５に示す．

また ep と mm(行列積計算)の MpC ベンチマークプログラムをトランスレートし，実行時間を計測した．ベンチマークのパラメータを表６に示す．

表７，表８はそれぞれ ep と mm の結果である．表の左から，今回の実装による(1)１スレッド実行，(2)２スレッド実行した場合である．さらに右側は比較のために，SDSM(SMS)で(3)１ノード

１プロセス実行，(4) １ノード２プロセス実行，(5) ２ノード各１プロセス実行した場合である．それぞれ１０回ずつ測定し平均を取った時間である．

mm では，２スレッドで実行した結果は，SDSM で同一ノードに２プロセス作った場合より僅かながら速い．しかし２ノードに各１プロセスずつ作った場合と比べると遅くなっている．mm は計算に比べメモリアクセスが多いため，１ノードに２つの実行実体があるとバスネックが起きている可能性がある．

一方 ep では，メモリアクセスに比べ計算負荷が高いため，同一ノードに２プロセスもしくは２スレッド作成した場合と，２ノードに各１プロセス作成した場合とで，大きな違いはない．

６　おわりに

今回の結果により，１ノードに２プロセスを生成し SDSM 機構を用いて実行する場合と，１ノードに２スレッドを生成し，pthread によって実行する場合とでは，必ずしも２スレッド実行のほうが有利であるとはいえないということがわかった．メモリバスネックと計算量とが要因になると考えられる．今後，さらに様々なメモリアクセスパターンを持つベンチマークテストを評価していく予定である．

参考文献

[1]緑川,片野,飯塚:"メタプロセスモデル －明示的並列処理記述のための分散共有メモリプログラミングモデル－",情報科学技術フォーラム FIT2003，情報科学技術レターズ,LB_002, pp.33-35，(2003,9)

[2]片野,緑川,飯塚:"分散共有メモリに適した並列言語 MpC",情報化学技術フォーラム FIT2003，FIT 論文集,B－017，pp.175-176，(2003,9)

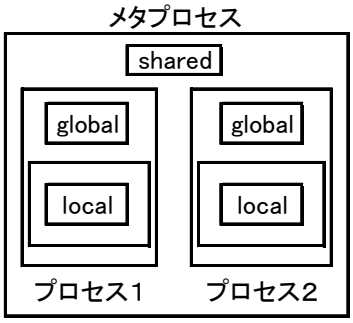

図1　メタプロセスモデル

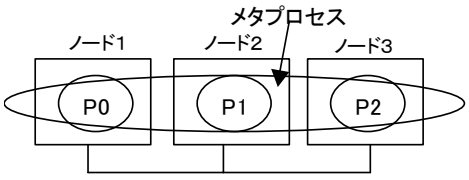

図2　クラスタにおけるメタプロセス

表1　メタプロセスモデルの実行単位実装系への対応

	SDSM クラスタでの実装	共有メモリ並列 マシンでの実装
一つの応用 （メタプロセス）	プロセス群	プロセス
並列実行単位	1プロセス	1スレッド

表2　メタプロセスモデルのデータの実装系への対応

メタプロセスモデル	SDSM クラスタでの実装	共有メモリ並列 マシンでの実装
shared	プロセス共有 （DSM機能利用）	スレッド共有 （グローバル変数）
global	プロセス局所 （グローバル変数）	スレッド局所 （ローカル変数）
local	ブロック内局所 （ローカル変数）	ブロック内局所 （ローカル変数）

表3　MpC標準関数

	MpC標準関数（一部）
初期化	mpc_init(int argc, char *argv)
終了	mpc_exit(int value)
エラー終了	mpc_err(int value, char *msg)
バリア	mpc_barrier(int value)
ロック	mpc_lock(int lockid)
アンロック	mpc_unlock(int lockid)
条件シグナル	mpc_cond_signal(int condid)
条件シグナル	mpc_cond_broadcast(int condid)
条件シグナル待	mpc_cond_wait(int condid, int lockid)
共有データ割付	void *mpc_alloc(size)
共有データ割付	void *mpc_alloc(char *declare)

表4 測定環境

CPU	INTEL PENTIUMⅢ-S 1.13GHz
CPU数	2
メモリ	1GB
OS	Redhat Linux9 kernel2.4

表5　基本APIの１回あたりの実行時間 （秒）

ロック	0.00000008
バリア	0.00003955
スレッド生成	0.00005477

表6　実行したベンチマーク

ベンチマーク	パラメータ
ep	NPB　クラスS
mm（行列積計算）	サイズ　double　1024×1024

表7　ep実行時間 （秒）

実装	pthread		SDSM(SMS)		
実行形式	1ノード 1スレッド	1ノード 2スレッド	1ノード 1プロセス	1ノード 2プロセス	2ノード 各1プロセス
計算時間	10.9789	5.6176	10.1988	5.1076	5.0968

表8　mm実行時間 （秒）

実装	pthread		SDSM(SMS)		
実行形式	1ノード 1スレッド	1ノード 2スレッド	1ノード 1プロセス	1ノード 2プロセス	2ノード 各1プロセス
計算時間	11.9282	9.8162	11.9575	10.3509	6.2769