AP1000互換通信ライブラリ: WSクラスタ向けの新しい計算環境 －基本コンセプト－*

2L－1

林 憲一　小林 健一　堀江 健志†
富士通株式会社 HPC本部‡

1 はじめに

ネットワークで繋がれたワークステーション(Networks of Workstation, NOW)は，並列処理を行なうプラットフォームとして，近年その重要度を増している．これは，最近のワークステーションがMPPのノードと比べてコストパフォーマンスに優れていることに加え，ATM等のネットワークスイッチが安価で提供されるようになり，低レイテンシ・高スループット通信が専用並列計算機だけの特権ではなくなりつつあるからである．また共有メモリをベースにしたSMP(Symmetrical Multiprocessors)も容易にスループット性能を向上させることができるため，多くの支持を得ている．現在，多くの大学，企業の計算機環境はこれらのマシンが混在してネットワークで接続され，ユーザはこれらのマシンを用途や可用性に応じて利用している．

これらのマシンを単体ではなく，並列計算機として利用する試みはこれまでMPI[1]やPVM[2]等で行なわれている．しかし，これまでのMPI[1]の実装では1プロセッサのワークステーション間はTCP/IP，SMPでは共有メモリによる通信というように各々のアーキテクチャ毎に特化した構成を採っており，SMPのクラスタのような場合は考慮されていない．またSMPの場合，MPI[1]，PVM[2]のこれまでの実装では，プロセスを処理の基本にし，共有メモリで通信しているため，SMPの性能を充分引き出すことができなかった[3, 5]．

AP1000互換通信ライブラリ(以下 APlib)はマルチスレッドを処理の基盤に採用し，SMPのクラスタ等の様々なアーキテクチャが混在する環境において，各アーキテクチャで最も高速な通信手段を利用して，効率的に並列処理を行なうための計算環境である．

1.1 APlib開発の目的

APlibは様々なアーキテクチャを統合して利用することで，SMPのクラスタ等のアーキテクチャが混在する環境にも対応し，且つユーザプログラムのオブジェクト互換を実現することで，ユーザプログラムの利便性を高めることを目的に開発された．

APlibではこの目的を実現するために，実行モデルとしてマルチスレッドとLight Weight Process(LWP)を採用した．これには以下のような利点がある．

1　スレッド間のスケジュールはユーザが定義できる
2　プリエンプティブなスケジューリングはLWPで対応可能
3　コンテクストスイッチが高速
4　SMPマシンでは各スレッド，LWPが別々のプロセッサで実行され，効率が良い

処理の単位として，スレッドとLWPを利用すると，ノンプリエンプティブなスケジューリングにはスレッド，プリエンプティブなスケジューリングにはLWPを割り当てることで様々なスケジューリングモデルに対応できる．しかもコンテクストスイッチは高速に行なえる．さらに複数のスレッドを起動し，それを仮想ノードとして利用することができる．SMPマシンではそれぞれのLWPが別々のプロセッサで実行され，効率が良い．

この仮想ノードの機能を応用して，性能の異なるワークステーションクラスタで，プロセッサの性能に比例して仮想ノードを割り当てることで，バランス良く並列処理を行なうことも可能となる[4]．

さらに3.2節で述べるようなモジュール化によって様々なアーキテクチャ上でオブジェクト互換を実現することができる．

このようにスレッドとLWPを基盤にした実行モデルを採用することで，様々なアーキテクチャの統合とオブジェクト互換という所期の開発目的を実現することが可能となる．

2 APlibのターゲットアーキテクチャ

APlibで書かれたプログラムは単体のワークステーションからNOW，SMP，SMPクラスタまで様々なアーキテクチャ上で実行できる．図1にAPlibにおける通信のメカニズムの概念図を示す．

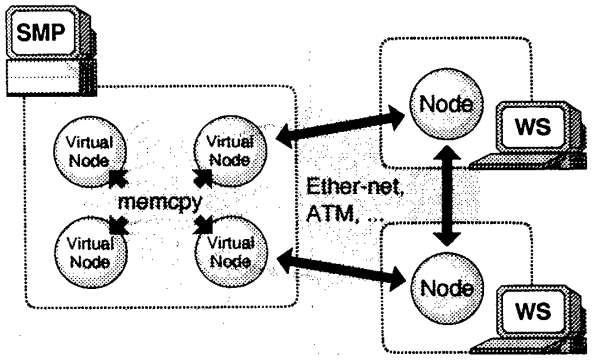

図 1: APlibの通信構造

2つのマシンに跨るノード間ではTCP/IP等の通信あるいは，ATM等の高速通信デバイスがある場合はそれを用い，同一マシン内に生成された仮想ノード間ではメモリ空間を共有していることを利用してメモリコピーで

*AP1000 compatible communication library: A new computing environment for workstation clusters(1)
†K. Hayashi, K. Kobayashi, and T. Horie
‡High Performance Computing Group, Fujitsu Ltd.

通信する.
以下に各ターゲットにおけるAPlibの特徴を述べる.

NOW ネットワークで繋がれたワークステーションではTCP/IP等の通信を用いて通信する. また高速通信デバイスを持っている場合にはそれを利用することも可能である.

SMP SMPでは1つのプロセス上に複数のLWPを起動して, これをノードに対応させ, 仮想ノードとする. 各ノード間の通信はメモリ空間を共有していることを利用してメモリコピーで行なうことが可能で, 通信オーバーヘッドが大幅に削減される.

またユーザプログラムはメッセージ・パッシングモデルに従っているので, SMPにおける通常の並列プログラムで発生するキャッシュコンフリクトやロック衝突などの性能阻害要因の影響が少なく, 台数効果が期待できる.

SMPクラスタ SMPのクラスタでは同一マシン上ではメモリ空間を共有していることを利用して, メモリコピーで通信を行ない, 他のマシンとの間ではTCP/IP等を用いて通信を行なう. 同一マシンかどうかの判断はAPlibが行なうので, ユーザはこれを区別する必要はない.

3 APlib: CellOSの実行モデルの実現

APlibが採用しているマルチスレッド実行モデルでは, ユーザがスケジューリングを制御可能なユーザスレッドとOSによってプリエンプティブに制御されるLWPを組み合わせて用いることにより, 様々な実行モデルを実現することができる. ここではその実現例として, AP1000の基本ソフトウェアであるCellOSの実行モデルを実現する方法を示す.

CellOSにおけるプログラムの実行は, AP1000が接続されているホストWS上で動作するホストプログラムと, セルと呼ばれるAP1000のノード上で動作するセルプログラムから成る. ホストプログラムは各ノードに対して, タスクの生成とデータの分散や収集などを行なう役割を持つ. セルプログラムはホストから各セルに1つあるいは複数ロードされ, タスクと呼ばれる独立の実行体となり, CellOSのスケジューラによってノンプリエンプティブにスケジューリングされる. タスクスイッチは受信メッセージ待ち, 及び同期成立待ちの時に起きる. CellOSのスケジューラは各タスクがスイッチすべきタイミングになると, 他のタスクの状態を調べ, 次に実行すべきタスクへ処理を移す.

3.1 APlibでのマルチタスク実行モデル

APlibでは, LWPをセル, スレッドをタスクに対応させることで, 仮想セルの実現とCellOSのマルチタスク実行モデルを実現している. これによって見かけのセル数を増やすことが出来ると共に, 各タスクは1つのプロセス内でCellOSの実行モデルに合わせてコントロールすることができる. 図2は仮想セルを2つ生成し, それぞれに3つのタスクを起動した様子を示している.

図2でAPkernelはタスク制御機構を持ったスレッドである. このAPkernelが各タスクのスケジューリングを制御し, ある瞬間に実行すべきタスクを選択する. タスクはLWPと結び付くことで高々1個が実行される.

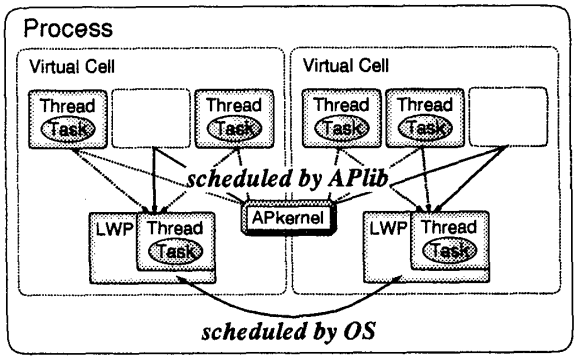

図 2: APlibの仮想セル

3.2 APlibのモジュラリティ

APlibのもう一つの目的であるオブジェクト互換を実現するために, 通信処理のうち, ハードウェアに依存する部分を独立なモジュールとしている. このモジュールはAPruntimeと呼ばれ, ダイナミックリンク可能ライブラリとして提供される. このため, APlibを利用して作られたプログラムは様々なプラットフォームでオブジェクト互換となり, 異なるシステム構成, 通信アーキテクチャにおいても, プログラムの変更や再コンパイルの必要はない.

4 まとめ

APlibによってこれまでAP1000のために作られたソフトウェア資産をNOWやSMP, SMPクラスタ上で有効に活用することができると共に, これら様々なプラットフォーム上で, 同じユーザビューで並列処理を行なうことが可能になる. またこれらの間でプログラムのオブジェクト互換を実現している.

APlibの詳しい実装と性能評価については[4]で述べる.

またAPlibが採用しているマルチスレッド実行モデルは様々な実行モデルに対応できる可能性を持っている. この可能性について今後検討を進めていく予定である.

謝辞

日頃御指導, 御助言頂く, HPC本部 石井本部長代理, 河内統括部長, 白石部長, 佐藤課長, 池坂課長, 高橋課長, 石畑課長ならびに同僚諸氏に感謝致します.

参考文献

[1] MPICH: Portable MPI implementation (ANL/MSU). WWW: http://www.mcs.anl.gov/home/lusk/mpich.

[2] PVM. WWW: http://www.epm.ornl.gov/pvm.

[3] A. Ferrari and V. S. Sunderam. TPVM: Distributed concurrent computing with lightwight processes. WWW: http://www.cs.virginia.edu/~ ajf2j, 1995.

[4] 小林 健一, 林 憲一, 堀江 健志. AP1000互換通信ライブラリ: WSクラスタ向けの新しい計算 環境 – 実装と評価 –. 情報処理学会第52回全国大会. 情報処理学会, 1996.

[5] H. Zhou and A. Geist. LPVM: A step towards multithread PVM. WWW: http://www.epm.ornl.gov/~ zhou, 1995.