解　説

IPS

並列処理のためのシステムソフトウェア

1. 並列処理のためのシステムソフトウェア†

笠　原　博　徳††

1. はじめに

現在，並列処理技術はワークステーションからスーパコンピュータに至るほとんどのコンピュータの基本構築技術となっている．これらのコンピュータでよく使用される並列処理方式としては，マイクロプロセッサ内で命令レベルの細粒度並列処理を行うスーパスカラ方式及び VLIW 方式，従来よりスーパコンピュータで使用されている演算パイプライン（ベクトルプロセッサ）方式，ミニスーパコンピュータ，スーパコンピュータなどで使用されているマルチプロセッサ方式などがあげられる[1)~3)]．これらの処理方式で共通している特徴は，ハードウェアのもつ高い処理性能を有効に引き出すためには，ソフトウェアによる強力なサポートが必須であるという点である．

特に，本特集で中心的に議論されるマルチプロセッサシステムにおいては，従来より，ハードウェアのもつピーク性能と実際のプログラムを実行したときの実質的な性能である実効性能との間のギャップがきわめて大きいという問題点があった．たとえば，比較的並列性の高いアプリケーションプログラムを用いた性能評価で，16台規模の共有メモリ型マルチベクトルプロセッサ・スーパコンピュータ上での実測性能がピーク性能の16%から50%，128台規模の分散メモリ型マルチプロセッサシステムで3%から20%，16384台規模の超並列システムで3%から12%という結果が報告されている[4)]．この実測性能がピーク性能の数%から50%という結果は一見すると低い値にみえるが，これらの値は並列処理の専門家が並列性の高いプログラムを手で最適化したときの最高性能であり，一般のユーザはこのような値を得ることすら困難である．このような問題が生じる原因としては，

①並列処理ハードウェア技術に比較し並列処理のためのシステムソフトウェア技術が大幅に遅れていること

②アーキテクチャ設計においてピーク性能の向上にとらわれ過ぎソフトウェアを実行したときの性能向上に対して十分な考慮が行われていない，すなわちシステムソフトウェアの能力を考慮し最大の実効性能を得るというハード・ソフト協調型の設計が行われていないこと

などがあげられる．

したがって，今後ピーク性能と実効性能との間のギャップを埋め，並列処理に関する専門知識のないユーザでも並列処理システムを容易に使用できるようにするためには，並列処理のためのシステムソフトウェア[5)~9)]を高度化させるとともに，それらのソフトウェアの性能を最大限に発揮させるハードウェアの開発が重要となる．

本稿では，このような並列処理システムソフトウェアの概要，及び課題について簡単に解説する．

2. 並列処理におけるシステムソフトウェア

今後の並列処理システムの使いやすさ及び性能を向上させるために重要なシステムソフトウェアとその代表的な機能を**図-1** に示す．以下図-1 を使用しながら，各ソフトウェア相互の関連及び各ソフトウェアの課題について述べる．

2.1 並列処理のためのプログラミング言語

ユーザが並列処理システムを使用するとき，最初に行うのは実行すべきプログラムの用意である．この際一般ユーザの多くは，Fortran, C などの逐次型言語を用いてプログラムを作成するか，

† System Software for Parallel Processing by Hironori KASAHARA (School of Science and Engineering, Waseda University).

†† 早稲田大学理工学部情報学科

従来より使い続けている逐次型言語で記述されたプログラム（ダスティデック）の無修正実行を試みる．一方，並列処理に興味があるユーザあるいは並列処理の専門家は，実効性能の向上を目指して，プログラム中の並列性，並列実行可能タスク間での同期（条件同期，排他制御など），データの分割・配置，あるいはプロセッサ間での通信などをユーザ自ら記述するために，Cray Fortran, Cedar Fortran[13], CM Fortran, Fortran D[14], HPF[5),15)], VPP Fortran のような逐次型言語を拡張した言語，あるいは Occam[10], GHC などのような並列型言語，DFC[12], Id, Valid のようなデータフロー言語[11),12)] を用いてプログラムを作成するというアプローチをとる．

ここで，プログラム中の並列性の記述とは，Fork-Join あるいは Cobegin-Coend などによる並列実行可能タスクの生成・終了の記述，Doall (Forall), Doacross などのループ並列化の記述，ベクトル化可能ループのベクトル命令による記述を意味する[3),13)]．

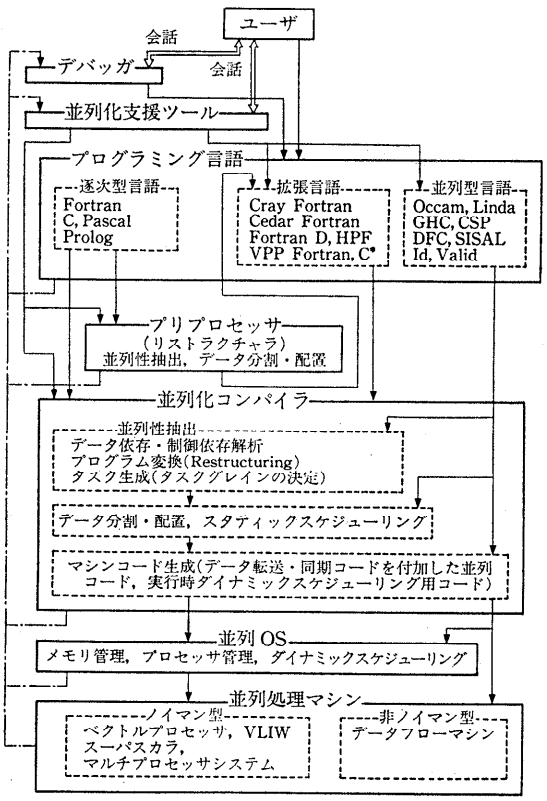

図-1 並列処理のためのシステムソフトウェアとその代表的な機能

また現在話題となっているデータの分割・配置[5),14),15)]は，各プロセッサ上のローカルメモリを有効に使用しプロセッサ間通信オーバヘッドを最小化するために行われるものであり，コンパイラによる自動分割・配置が実現されていない現状ではユーザがプログラム中で指定することが必要となる．このデータの分割・配置は，分散メモリ型マルチプロセッサシステムでは各プロセッサ上のローカルメモリへの分割・配置，各プロセッサ（あるいはプロセッサクラスタ）がローカルメモリをもちさらにプロセッサ間グローバル（集中型）共有メモリをもつシステムではローカルメモリとグローバル共有メモリへの分割・配置[13)]，また各プロセッサ上にローカルメモリと分散共有メモリさらにグローバル（集中型）共有メモリをももつシステムではそれら3種のメモリ上への分割・配置の記述が必要となる．

以上のような拡張言語及び並列型言語は，各研究機関，企業によって多くの種類が提案されており，同じ Fortran をベースに拡張した言語であっても各社ごとに記述法が異なり，ユーザは異なるメーカの並列システムを使用するたびにプログラムを書き換えなければならないという問題点があった．このような状況はユーザにとって好ましくなく，どの企業のマシン上でもそのまま実行可能な標準的な言語の開発が望まれる．このような観点からライス大学の Fortran D[14] 及び Fortran 90 をベースに標準化を目指す HPF[5),15)] は，同一プログラムの各社マシン上での実行・評価を可能にするという点で注目される．この HPF はデータ分割・配置のために，ALIGN（データの相互配置），DISTRIBUTE（プロセッサ上へのデータ分散の方法），PROCESSOR（割当対象となるプロセッサの構成）などのディレクティブを用意しているところに大きな特徴がある．しかしこのデータ分割・配置のように現在多くの研究がなされている分野での標準化は，分割・配置の技術が日々進歩していくためその進歩に応じて標準を変えていく必要が生じる場合があり，時として困難をともなう．

2.2 並列化コンパイラ

逐次型言語で記述されたプログラムは，自動並列化コンパイラ[3),6),20)~22)]により自動的に並列化マシンコードに翻訳され，並列 OS の管理下で実行される．また，拡張言語，並列型言語で記述さ

れたプログラムも並列化コンパイラにより翻訳されるが，拡張言語及び並列型言語用のコンパイラは並列性の抽出，データの分割・配置，スケジューリングなどをユーザに頼ることが多いため，逐次型言語用自動並列化コンパイラより一般的には低機能なものでよい．そこで，ここでは並列処理の専門知識のないユーザでも並列処理システムを簡単に使用できしかもハードウェアの最大性能を得られるようにするための，自動並列化コンパイラについて簡単に説明する．

現在までに実用化されている自動並列化コンパイラは主にループ内データ依存解析[19),20)]に基づく自動ベクトル化，ループ並列化 (Doall, Doacross)[3),20)] が中心であった．自動ベクトル化，ループ並列化に関しては，シンボリックデータ依存解析，セマンティックデータ依存解析，イタレーションスペースデータ依存解析，依存方向ベクタ解析，GCD テスト，Benerjee の exact test, Benerjee-Wolfe の inexact test, OMEGA テストなどのデータ依存解析技術[3),19),24)]，ノードスプリッティング，スカラエクスパンション，ストリップマイニング，ループインタチェンジ，ループフュージョン，マルチバージョンループ生成など各種のプログラムリストラクチャリング手法[3),20)]も出尽くしかなり成熟期に入っている．

そこで今後マルチプロセッサシステムの実効性能を向上させるためには，

（a） 従来並列処理が行なえなかったシーケンシャルループあるいはループ外ベーシックブロックを，プロセッサ間で，複数命令レベルあるいはステートメントレベルの粒度で並列処理を行う近細粒度並列処理[3),25)]

（b） プロセッサあるいはプロセッサクラスタ間で，シーケンシャルループ，サブルーティン，ベーシックブロック間の自動並列処理を可能とする粗粒度並列処理（マクロデータフロー処理）[4),6),27),28)]

（c） マクロデータフロー処理とも関連し，分散メモリ型マルチプロセッサシステム，あるいは，ローカルメモリ＋グローバル（集中）共有メモリ，ローカルメモリ＋分散共有メモリ，ローカルメモリ＋分散共有メモリ＋グローバル（集中）共有メモリなど各種共有メモリ型マルチプロセッサシステム，においてローカルメモリを有効利用しプロセッサ間データ転送を最小化するためのデータ自動分割・配置[26)]

（d） データプレローディング（プリフェッチ）・データポストストアなどの技術を導入しタスク処理とデータ転送のオーバラップを図りデータ転送オーバヘッドの最小化を行うデータ転送タイミングの自動最適化[29)]

などの新技術を実装したコンパイラ[30),31)]の実用化が望まれる．

また，コンパイラと同様に逐次型プログラムのリストラクチャリングを行うが，出力をユーザに理解しやすい拡張プログラミング言語で記述されたプログラムコードとするリストラクチャラ（たとえばイリノイ大 Parafrase, KAI 社 KAP など）と呼ばれるプリプロセッサもある．このようなリストラクチャラは次に述べる並列化支援ツールと複合的に使用されることにより，その適用効果が高まる[8),18)]．

2.3 並列化支援ツール

並列処理に興味をもつユーザあるいは並列処理の専門家が並列処理マシンを使用する際，コンパイラの性能が十分でないなどの理由でマシンの最高性能を引き出すためのチューニングを行いたい場合がある．このようなチューニングを行う際は，

（a） ユーザにコンパイラあるいはプリプロセッサが自動並列化をできない理由（たとえばデータ依存グラフなどを初めとした各種データ依存情報）を提示する

（b） 各プログラム部分に適用可能なリストラクチャリング手法を提示しユーザの指示によりリストラクチャリングを行った後のプログラムコード，及びその実行コストの推定値を提示する

（c） 実行プロファイルなどからプログラム各部の実行コスト（たとえば各ループ処理時間），条件分岐の分岐確率など各種情報をユーザに提示し，重点的にチューニングすべき部分をユーザに示す

などの機能をもつユーザフレンドリな会話型並列化支援ツール[8),18)]が重要となる．このような会話型並列化支援ツールの実現及び利用のためには，支援ツールとプリプロセッサ，コンパイラ，OS，並列処理マシンとの間で密接な情報授受が必要となる．

2.4 並列OS

並列OS[7),16),17)]は，メモリ管理（仮想メモリ，仮想分散共有メモリなどの管理），プロセッサ管理（仮想プロセッサと物理プロセッサのマッピング），ジョブ管理（ジョブスケジューリングなど），タスク管理（並列タスクの生成・駆動・終了，タスクスケジューリング），プロセッサ間通信，など多くの重要な役割を果たす．しかし従来のマルチプロセッサ・システムでは，ハードウェアの設計が優先され，OS，コンパイラなどはそのハードウェアを有効に使用するために後から設計するというアプローチがとられていた上，コンパイラが未熟のため静的な管理ができない処理はすべてOSに任せるというケースが多く見られた．このため，必然的にOSは重くなり，プロセッサ間通信，並列タスクの生成・駆動，タスク間同期，仮想分散共有メモリへのアクセスのためにOSコールを行うと数百から数千クロックを要するという場合が生じ，単一ジョブの応答性向上を阻害するという弊害がみられるようになった．この数百クロックという時間は，インタコネクションネットワークの高速化，コンパイラによる最適化などによる速度向上と比較しきわめて大きい時間であり，OSコールを頻繁に行うように設計されたシステムでは，インタコネクションネットワークの高速化，コンパイラの最適化レベルの向上がほとんど処理速度の向上につながらないという極端な場合すらあり得る．

このような状況を打破するためには，マイクロカーネル，ナノカーネル化といったOS自身の軽量化とともに，ハードウェア，並列OS，並列化コンパイラの適切な役割分担を考え直すことが必要となる．たとえば，

（a） コンパイラができるだけ静的にスケジューリングなどを行う，あるいは実行時に動的スケジューリングを行うコードをコンパイラが生成するなどの方式をとり，従来OSに頼っていた作業のうちコンパイラに移行できる機能は，コンパイラに任せる

（b） 分散メモリ型マルチプロセッサシステム上での仮想分散共有メモリの実現などOSに頼っていた機能を，もしプログラミングの容易化のために分散共有メモリが必要なら従来の分散メモリ型アーキテクチャに固執せず，実際に共有メモリを各プロセッサ上に分散する分散共有メモリ型アーキテクチャとしハードウェアで実現する

などによりOSの負荷軽減を図ることが考えられる．

特に今後の超並列マシンのようにプロセッサ台数が増えてくると，各ジョブの応答性向上とシステム全体でのスループットの向上という相異なる要求を満たすOSの開発がますます強く望まれる．究極的にはジョブ間の各種制御はOSが責任をもち，単一ジョブ内の並列処理，特にスケジューリング，同期，通信にはOSは関与しないですむコンパイラの開発及びアーキテクチャサポート技術の開発が重要になると考えられる．

2.5 並列処理用デバッガ

並列処理におけるデバッグには，

（a） デバッグ用プローブの挿入など実行タイミングの変化を生じる原因が多数存在するため，同期などに関連したバグ発生時の実行状況の再現及び原因の特定が困難である

（b） マルチプロセッサシステムにおいては全プロセッサの動作を同時にストップさせるのが困難でありバグ発生箇所の特定が難しい

（c） ハードウェアバグかソフトウェアバグ(ユーザプログラム，システムソフトウェアを含め）かの見極めが逐次型マシンの場合に比べさらに難しい

など各種の難しさがある．このため，現在まだ定番と呼べるデバッグ技術[9)]は確立されておらず，今後の研究が期待される．

このデバッガの開発においては，デバッガ内へのソフトウェアシミュレータの組込みによる並列プログラム中の同期などの徹底検証，ハードウェアによるデバッグサポート機能の充実，デバッグ作業におけるコンパイラ及びプリプロセッサから手にはいるデータ依存／制御依存情報などの利用，OS及びハードウェアから手にはいるプログラム実行情報の利用，ユーザとの会話機能の充実などが今後重要となる．

3. まとめ

本稿では，並列処理記述用プログラム言語，自動並列化コンパイラ，並列化支援ツール，並列OS，並列デバッガなど並列処理のためのシステムソフトウェアの概要，それらの相互関係，及びお

のおのの問題点と動向について簡単に述べた．並列処理のためのソフトウェアの研究・開発は，従来，並列ハードウェアの開発に比較し大幅に遅れていた．今後使いやすく，実効性能の高い並列処理システムを開発していくためには，システムソフトウェアに関する一層の研究努力が必要と考えられる．

参 考 文 献

1) 富田眞治：並列計算機構成論，昭晃堂 (1986).
2) 村岡洋一：並列処理，昭晃堂 (1986).
3) 笠原博徳：並列処理技術，コロナ社.
4) Cybenco, G., Kuck, D. J.: Teraflops Galore: Supercomputing/Reinventing the Machine—Revolution or Evolution?, IEEE Spectrum, pp. 39-41 (Sep. 1992).
5) 郷田：High Performance Fortran，情報処理，Vol. 34, No. 9, pp. 1179-1186 (Sep. 1993).
6) 本多：自動並列化コンパイラ，情報処理，Vol. 34, No. 9, pp. 1150-1157 (Sep. 1993).
7) 福田：並列オペレーティング・システム，情報処理，Vol. 34, No. 9, pp. 1139-1149 (Sep. 1993).
8) 菊池：並列化支援システム，情報処理，Vol. 34, No. 9, pp. 1158-1169 (Sep. 1993).
9) 山田：並列処理システムにおけるプログラムデバッギング，情報処理，Vol. 34, No. 9, pp. 1170-1178 (Sep. 1993).
10) 尾内：Occam とトランスピュータ，共立出版 (1986).
11) 曽和：データフローマシンと言語，昭晃堂 (1986).
12) 弓場，山口：データ駆動型並列計算機，オーム社 (1993).
13) Guzzi, M., Padua, D., Hoeflinger, J., Lawrie, D.: Cedar Fortran and Other Vector and Parallel Fortran Dialects, Proc. of Supercomputing '88, pp. 114-121 (Mar. 1988).
14) Hiranandani, S., Kennedy, K., Koelbel, C., Kremer, U. and Tseng, C.: An Over-View of the Fortran D Programming System, Proc. Workshop on Languages and Compilers for Parallel Computing, pp. 18-34 (Aug. 1991).
15) High Performance Fortran Forum: High Performance Fortran Language Specification DRAFT Ver. 1.0 (Jan. 1993).
16) Rashid, R.: Threads of a New System, UNIX Review, pp. 37-49 (Aug. 1986).
17) Emrath, P.: Xylem: An Operating System for the Cedar Multiprocessor, IEEE Software, Vol. 22, No. 4, pp. 30-37 (July 1985).
18) Kennedy, K., McKinley, K. S. and Tseng, C. W.: Interactive Parallel Programming Using the ParaScope Editor, IEEE Trans. on Parallel & Distributed Systems, Vol. 2, No. 3, pp. 329-341 (July 1991).
19) Banerjee, U.: Dependence Analysis for Supercomputing, Kluwer Academic (1988).
20) Padua, D. J. and Wolfe, M. J.: Advanced Compiler Optimizations for Supercomputers, C. ACM, Vol. 29, No. 12, pp. 1184-1201 (Dec. 1986).
21) Allen, J. and Kennedy, K.: PFC: A Program to Convert Fortran to Parallel Form, Proc. IBM Conf. on Parallel Computers and Scientific Computations (1982).
22) Allen, F. et al.: A Framework for Determining Useful Parallelism, Proc. 2nd ACM Int'l. Conf. on Supercomputing (1988).
23) Kasahara, H. and Narita, S.: Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, IEEE Trans. Comput., Vol. C-33, No. 11 (Nov. 1984).
24) Wolfe, M.: Optimizing Supercompilers for Supercomputers, MIT Press (1989).
25) Kasahara, H., Honda, H. and Narita, S.: Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR, Proc. IEEE ACM Supercomputing '90 (Nov. 1990).
26) Li, J. and Chen, M.: Compiling Communication Efficient Programs for Massively Parallel Machines, IEEE Trans. Parallel & Distributed Systems, Vol. 2, No. 3, pp. 361-376 (July 1991).
27) Kasahara, H., Honda, H., Iwata, M. and Hirota, M.: A Macro-Dataflow Compilation Scheme for Hierarchical Multiprocessor Systems, Proc. Int. Conf. on Parallel Processing (Aug. 1990).
28) 笠原，吉田，合田，岡本，本多：Fortran マクロデータフロー処理のマクロタスク生成手法，信学誌，Vol. J 75-D 1, No. 8, pp. 511-525 (1992).
29) 藤原，鈴木，白鳥，笠原：データプレロードポストストアを考慮したマルチプロセッサスケジューリングアルゴリズム，信学論，Vol. J 75-D 1, No. 8, pp. 495-503 (1992).
30) Kasahara, H., Honda, H. and Narita, S.: A Multi-Grain Parallelizing Compilation Scheme for OSCAR, Proc. 4th Workshop on Languages and Compilers for Parallel Computing (1991).
31) Kasahara, H., Honda, H. and Narita, S.: A Fortran Parallelizing Compilation Scheme for OSCAR Using Dependence Graph Analysis, IEICE Trans. Vol. E 74, No. 10, pp. 3105-3114 (1991).

(平成5年7月28日受付)

笠原 博徳（正会員）

昭和32年生．昭和55年早稲田大学理工学部電気工学科卒業．昭和60年同大学院博士課程修了．工学博士．昭和58年～60年早稲田大学理工学部助手．昭和61年早稲田大学理工学部電気工学科専任講師．昭和63年早稲田大学理工学部電気工学科助教授．平成3年情報学科助教授，現在に至る．平成元年～2年イリノイ大学 Center for Supercomputing Research & Development 客員研究員．昭和62年 IFAC World Congress 第1回 Young Author Prize 受賞．著書「並列処理技術」(コロナ社)．電子情報通信学会，電気学会，シミュレーション学会，ロボット学会，IEEE，ACM 各会員．