2U－1

スケーラブルプログラミングシステム 背景と目的

藤田昭平
fujita@cs.titech.ac.jp
東京工大 大学院情報理工学研究科

1 はじめに

1980年代に大学および企業の“研究室”レベルで研究され、試験的に使用されてきた並列マシンは、1990年代に入って商用化され、“実社会”での問題に適用されつつある。しかし、“雄大な挑戦(GC)”[1]問題[3]の解決に際しては、

- 大規模科学技術計算
- リアルタイムシミュレーション
- 大規模分散データベース

を統合しなければならない。このためには“能力的に”充分であるとはいえない。さらに“高性能”なコンピューティングシステム、すなわち、“3T”コンピューティングシステムを目指して、ハードウェア、ソフトウェア、および応用分野の研究を三位一体に推進すべきである[4]。

現在、並列処理の分野で最も必要なことは並列プログラミング環境を完備し、従来の単一プロセッサシステムと同様なソフトウェア開発環境を確立することである。ユーザにとって最も望ましいのは、“スケーラブル”なソフトウェア/マシンであり、ネットワーク環境が優れたシステムである。

2 MPP ソフトウェア技術の現状と研究課題

2.1 応用分野とアルゴリズム

エンドユーザは高度な計算アルゴリズムには詳しいが、(一般に)MPP システムそれ自体にはあまり慣れていない。MPP システムの厄介なプログラミングを容易にするソフトウェア開発ツールが、当面差し迫った問題である。MPP の市場および GC 応用分野は、従来のコンピューティングシステムに比べれば、現在はそんなに大きくない。今後の発展の“誘因”となるような役立つ事例を追求すべきである。

[1]Kenneth Wilson により提唱された新造語

“スケーラブル”な応用分野が充分に解明されていない。並列プログラミングのパラダイムに関する理解が不十分である。実行性能の最適化とポータビリティ/プログラミングの容易性に関するトレードオフをどうするか？

アルゴリズムとコードのポーティングを強力に支援すべきである。“かん詰め”パッケージでは、実行性能の面からは一般的すぎるし、マシンアーキテクチャの観点からはあまりにも特殊化されている。“型枠”的なアプローチを推進すべきである。この型枠は汎用的な並列コードとし、特別な要求あるいは特殊なシステムのために、エンドユーザがこれを自由に変更し最適化できるようにすべきである。

並列計算用の非同期アルゴリズムは、高性能コンピューティングには極めて重要である[1]。高位言語でアルゴリズムを記述することにより、ソースコードへの変換は容易になるが、マシンに依存した最適化の問題は残る。このような問題は、分散メモリアーキテクチャのマシンでは、特に重大である。共有アドレス空間を有している並列マシンでは、それほど問題にならない。このように、マシンアーキテクチャが種々あるために、それぞれに適したアルゴリズムを開発しなければならない。

数値計算用ソフトウェアの開発者は、応用分野の要求とコンパイラー、実行時支援システムを充分に理解する必要がある。コンパイラーは、コード生成過程で、浮動小数点表現に関する詳細な指示を与えるべきである。

2.2 可視化技術とデータベース

リアルタイムシミュレーションから得られる多量のデータの迅速で高度な解析手段である科学的可視化は、MPP マシンを導入する誘因の一つである。また、応用ソフトウェア開発に際しても、非常に有効な手段である。

しかし、ディスク、I/O の帯域幅の制限により、現

用されている可視化システムの性能は満足できるものではない。これを解決する方法は、分散可視化環境を構築することである。すなわち、MPP、グラフィックワークステーション、データベースサーバを "超" 高帯域の光ファイバー **ATM - LAN** でネットワーク (分散メタコンピュータ) 化すべきである。

2.3 ソフトウェア開発ツール

MPP 応用ソフトウェアの開発に際しては、並列プログラムのデバッグのためのツールが当面必要である。並列システムでは、誤りを含んでいる動作を解析する仕事は複雑になる。並列処理では、容易に再現できないような過渡的なエラーが潜んでいることがある。共有メモリシステムでは競合問題が、メッセージ交換システムではタイミングが原因となる。

並列応用コードが正しく実行されていても、その実行性能は利用可能なプロセッサの数に大きく依存する。性能評価ツールは、次のような問題に答えねばならない。

- 与えられた並列マシンでは、最高の実効性能はどのくらいか ?
- プログラムの修正により、性能がどれくらい改善されるか ?
- 適用問題およびコンピュータシステムのスケールを変えることにより、性能はどのように変化するか ?

以上の問題は、応用ソフトウェア開発者のみならず、システムソフトウェアおよびハードウエアの設計者にとっても、次世代の高性能コンピューティングシステムを開発するために必要不可欠である。

2.4 プログラミング言語とコンパイラ技術

言語/コンパイラは、エンドユーザとマシンの実行環境との論理的なインタフェースとなるものである。これらは、コンピューティングシステム全体の性能、使いやすさ、応用コードのポータビリティに大きな影響を及ぼす。唯一のプログラミングパラダイムだけでは、エンドユーザの種々の問題に対処できない。データ並列、タスク並列、オブジェクト指向、関数型等幾つかのプログラミングパラダイムを統合する必要がある。

言語/コンパイラは、それぞれ別個に開発されたプログラムモジュールの "再利用" を促進する必要がある。それぞれ別個に開発されたプログラムモジュールから一つのプログラムを構成する方法は、MPP のソフトウェア開発に大きく貢献する。

2.5 オペレーティングシステムと実行環境

ハードウェアおよびファームウェア化された計算資源との抽象的なユーザインタフェースをもたらすのが、オペレーティングシステムの役目である。MPP システムに代表されるような新しいコンピュータシステムの出現により、オペレーティングシステムに対する要求も変化している。TeraFLOP へとスケーラブルな新しいシステムソフトウェアの開発は最大の研究課題である [2]。

3 おわりに

プログラミングパラダイムと応用分野の関係をさらに研究すべきである。応用ソフトウェアの開発者とコンパイラ、実行時支援システム、オペレーティングシステム等のシステムソフトウェアの開発者が協力すべきである。

参考文献

[1] S. Fujita, "Distributed MIMD Multiprocessor System with MicroAda/SuperMicro(TM) for Asynchronous Concurrent Newton Algorithms," **Proc. 5th ACM - SIGSMALL Symp.**, 49 - 59, (1982).

[2] S. Fujita, "Self-Organization in Distributed Operating System," **Neural Networks**, Vol.1, Suppl. 1, (1988).

[3] Office of Technology Assessment, "**High Performance Computing & Networking for Science**," (Sep. 1989).

[4] **Pasadena Workshop : System Software and Tools for HPC Environments**, (Apr. 14 - 16, 1992).

Scalable Programming Systems : Motivation & Theme

Shohei FUJITA
TITech, Graduate School of Computer Science
Meguro-ku, Tokyo 152, Japan.