GRIDに対応した自動並列分散化コンパイラとその視覚化支援環境の開発

- 対話型コンパイラのための視覚的インタフェースの開発 -

１．背景

これまで、プログラムの最適化や並列化といったプログラムをより効率よく実行するためのプログラム変換は、コンパイラによって自動的に行われ、ユーザは一旦プログラムを書いたらそのプログラムの性質について知る必要はありませんでした。しかし、プログラムが大規模かつ複雑になってくるにつれ、プログラムのデバッグやメンテナンスのために、すでに書かれたプログラムの性質をユーザが理解することが重要となってきています。また、プログラムの最適化や並列化に関しても、完全に自動的に行うよりもユーザの知識を利用する方がより効率の良いプログラムに変換することができるため、プログラムの性質をユーザが理解することが重要です。分散環境におけるプログラムの分散実行での最も重要な問題の一つであるデータ分割においても、HPF(High Performance Fortran)のようにユーザに分割の仕方を指定させる場合が多く見受けられます。

このように、プログラムの性質をユーザが理解していることは重要でありますが、ユーザがプログラムの性質をソースコードから直接読み取るのは一般に非常に困難です。確かに、プログラムの性質はソースコードの中に表現されているものではありますが、明示的な形でソースコードに現れているわけではありません。そこで、ユーザのプログラムの理解を支援するシステムが必要になります。本プロジェクトで採用する並列化支援視覚化システムNaraViewは、このようなシステムの一つです。

また、自身での並列プログラムのコーディングが困難である圧倒的多数のユーザのために、自動並列化コンパイラの研究が長年続けられています。共有メモリ型並列計算機に対する自動並列化は、Banerjee博士のループ変換理論等により、既に成熟した研究となり、プロダクトも近年多く出回ってきています。イリノイ大学で開発されたPROMISもそのようなコンパイラの一つです。この自動並列化コンパイラを、分散メモリ環境で適用しようとすると、データ分割・再配置・通信遅延問題が発生し、生成される並列プログラムの実行効率が極端に悪くなってしまう場合があります。そこで分散メモリ型対応のコンパイラの研究がなされてきました。我々がPROMISを拡張して開発しているPROMIS-NWUもそのようなコンパイラの一つです。

しかし、そのような自動並列化コンパイラも多数の大規模数値計算のアプリケーションユーザには敷居が高いと言われており、その理由は、自動並列化コンパイラの操作方法が複雑だからなのではなく、コンパイラに与える指示

が複雑なためです。PROMIS や Parafrase-2 では、効率的な実行コードを生成するためには、コンパイラや並列化のことをよく理解し、それをパスとして正しくコンパイラに与えなければならず、それは一般的なアプリケーションユーザにはほとんど実現不可能な要求となります。
このような問題があることから、並列化支援視覚化システム NaraView の概念を自動並列分散化コンパイラ PROMIS-NWU に取り入れることで、ユーザが直感的にプログラム情報の把握とそれに基づく並列/最適化操作を行えるようにすることが本プロジェクトの目的になります。

２．目的

自動並列分散化コンパイラ PROMIS-NWU に並列化支援視覚化システム NaraView の概念を取り入れることで、ユーザが直感的にプログラム情報の把握とそれに基づく並列/最適化操作を行えるようにします。
これによって、人間の視覚的認識能力を並列最適化手法の選択に生かせるようになり、また、そのような操作がインタラクティブに行えることで、より簡単に効果的な最適化手法を選択することができるようになります。

３．開発の内容

自動並列分散化コンパイラ PROMIS-NWU に、並列化支援視覚化システム NaraView の概念に基づいた視覚化インタフェースを実装することで、ユーザが直感的にプログラム情報の把握ができるようにし、また、その NaraView インタフェースから直接、並列/最適化操作を行えるようにすることで、インタラクティブな最適化を実現します。
システム構成の概要は、以下の図１のようになります。

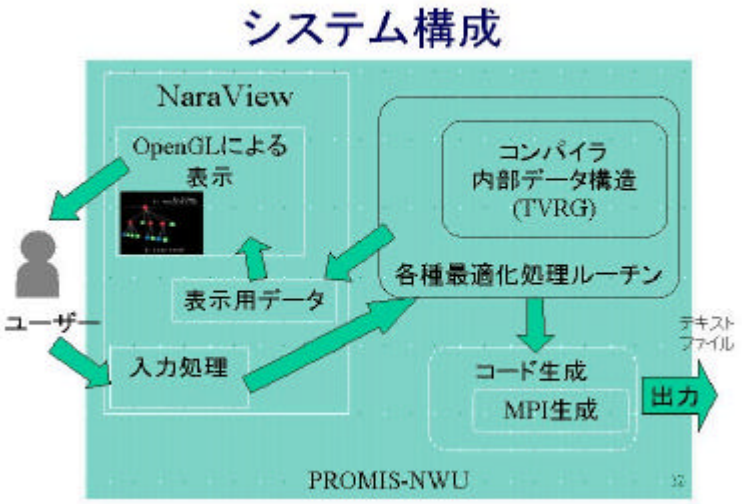

図１:システム構成

図１のように､ユーザは NaraView によって表示されたコンパイラ内部のプログラムに関する情報を元に、最適化手法を決定し、入力を行います。それらはコンパイラ内部の最適化処理ルーチンで内部データに反映され、その結果は即座に NaraView によって表示されます。これらを繰り返すことにより、より効果的な最適化手法が選択できます。

NaraView では、複数のビューでプログラム情報を様々な観点から見ることができます。一つは、PSV(Program Structure View)で、プログラムフローを基準に視覚化されているので、プログラム全体の構造を把握でき、ループの数及び位置、すでに並列化されているかどうかなどの情報を知ることができます。(図２)
また、PSV では、個々のループノードに対して直接最適化手法を選択することが可能です。(図３)
もう一つの DDV(Data Dependence View)では、データ依存情報を基準にした視覚化を行っており、従来把握しにくかった配列のデータ依存などが直感的に把握できるようになっています。(図４)

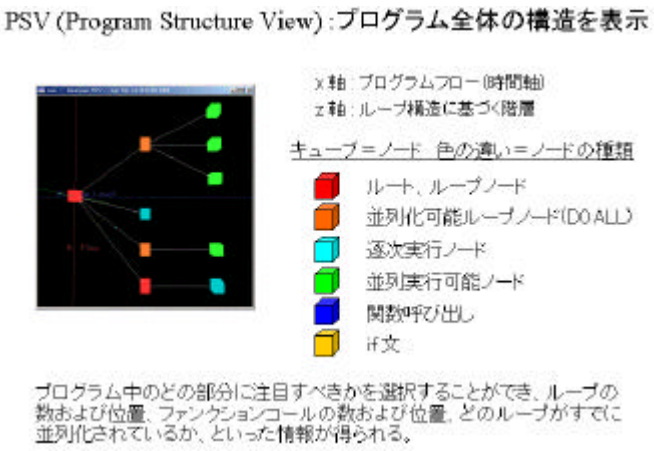

図２:PSV(Program Structure View)

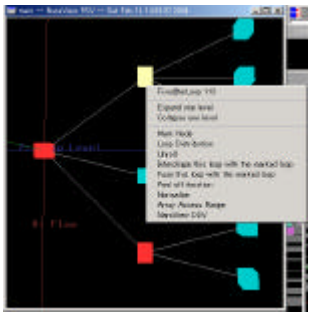

図３

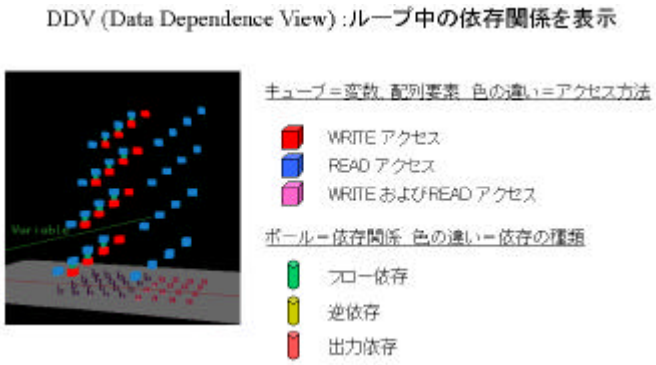

図４:DDV(Data Dependence View)

４．従来の技術(または機能)との相違

ユーザの知識を利用してプログラムをより効果的に並列/最適化しようとするシステムでは、プログラムの性質をユーザが理解するのを助けるために、コンパイラがプログラムを解析した結果をユーザに提示しています。
ところが従来のシステムでは、

- コンパイラ内部で使用することを目的として、プログラムを解析した結果をそのままユーザに提示している。
- 解析の表示がテキスト中心である。

という理由から、ユーザにとってわかりやすく情報を提示しているとは言い難い状況でした。
我々のシステムではこの問題を解決するため、ユーザに必要な情報という観点からデータのモデリングを行い、それに基づいた視覚化を行っています。
データ依存を変数中心にとらえて視覚化した例は、あまり類を見ません。

５．期待される効果

本システムを活用すれば、分散プログラミングへの負荷が軽減するので、クラスタなどの分散メモリ型並列計算機導入の際のコスト削減につながり、それらの購入・販売を促進し、市場を活性化する可能性があります。

６．普及(または活用)の見通し

大規模数値計算を行っている、比較的中・小規模な企業または大学の研究室などでは、計算機の購入に莫大な資金をかけられないため、比較的安くて性能の高いクラスタなどの分散メモリ型計算機の需要があります。ところが、分散メモリ型計算機では、並列化の知識がなければ逆にプログラミングのコストが高くなってしまうことがあり得るので、本システムを用いてそのコストを小さくおさえることができます。

７．開発者名

* 山口智美(奈良女子大学大学院 人間文化研究科 複合現象科学専攻 複合情報科学講座 tomomi@ics.nara-wu.ac.jp)