# 分散環境のための実行時間の見積もり

4X－11

浜中 征志郎　首藤 一幸　菅原 健一　村岡 洋一
早稲田大学　理工学部

## 1. はじめに

ワークステーションクラスタ等の分散環境を対象とした並列化コンパイラ及び実行時システムの実現を目指している。並列化コンパイラはプログラムをタスクに分割する際、そのタスクの計算時間と、タスクが別の計算機に割り当てられた時に必要となる通信にかかる時間について見積もる必要がある。しかし分散環境においては、ソースプログラムのみからの見積もりは困難である。

そこでコンパイラと実行環境を把握する実行時システムが協力することで上記の問題の解決を目指す。コンパイラは必要に応じて実行時システムからのベンチマークの結果、計算機の負荷、仮実行の結果を利用することで、実行環境を反映した実行時間の予測値を返す。本稿では見積もり手法の概要について述べ、Livermore Loop 24 に本手法を適用した場合の有効性について考察する。

## 2. 分散環境での問題点

従来、並列化コンパイラはタスク分割の際に必要となる計算時間、通信時間をソースプログラムから見積もってきた。ソースプログラムから見積もられたこれらの計算コスト、通信コストは単位を持たない値であるため、両者の値を比較する場合にはこれらを何らかの方法で時間に直す必要がある。専用の並列計算機を対象としたコンパイラでは時間との比を予め与えておく手法が採られている。

しかし分散環境では予め比を与えておく手法では精度のよい見積もりは行えない。それは主に以下の理由による。

**heterogeneity** 専用の並列計算機と異なり、必ずしも同性能の計算機が接続されている訳ではないため、計算機環境の把握がより困難となる。

**計算機の負荷** 専用の並列計算機では一つのプログラムが占有して利用できる状況が多いが、分散環境では多くの場合他のユーザも計算機を利用しているため、その分の負荷がかかる。

An Execution Time Estimation Method for Distributed Environment
Seishiro HAMANAKA, Kazuyuki SHUDOH, Kenichi SUGAHARA, Yoichi MURAOKA
School of Science and Engineering, WASEDA University

従って分散環境では実行環境を把握しておくことがより重要となる。従来の多くの並列化コンパイラはMPIを利用したコードを生成することにより分散環境に対応しているが、実行環境を把握しているとは言えない。

## 3. 見積もり手法

そこで本研究ではソースプログラムからの情報だけではなく、実行環境から得られる情報を利用して実行時間の見積もりを行う。以下の二つの手法を併用する。

### 3.1 ソースプログラムベース

ソースプログラムより静的に見積もった結果から計算機の性能と負荷を加味して実行時間を予測する。まず本コンパイラでの中間表現上での演算子の数を基本とした計算コストを求める。ループのコストはそのイタレーション数が求められればループボディのコストにイタレーション数を乗じた値とし、イタレーション数が求められなかった場合は予め設定しておいた値を乗じることにする。条件分岐のコストは全ての分岐先のコストの最大値に条件文のコストを加えた値とする。この結果得られたコストを $c$ とする。

またそのタスクを実行する計算機上で予め上記の手法によりコストの求められているベンチマークを実行し、実行時間を測定しておく。そして実行時間をコストで割ったコスト当りの実行時間を $r$、その時の計算機の負荷を $w_1$ とする。ここで計算機の負荷とは平均実行待ちプロセス数である。

そしてタスクを実行する時の計算機の負荷の予測値を $w_2$ とする。負荷の予測値は見積もり時の計算機の負荷とする。予測実行時間を、

$$c \times r \times \frac{w_2 + 1}{w_1 + 1}$$

で与える。

### 3.2 仮実行

実際にそのタスク(サブプログラム)を計算機上で実行して実行時間を計測し、予測実行時間とする。計測時間が一定になるように計算コストから予測実行時間が一定になるようにタスクのサイズの調整を行い、中間表現からコード生成し、実行、計測を行う。この結果から元のタスクの予測実行時間を得る。

## 4. 見積もりのためのシステム

これらの見積もりを行うために、コンパイラの他に実行時システムを用意する。システム構成を図1に示

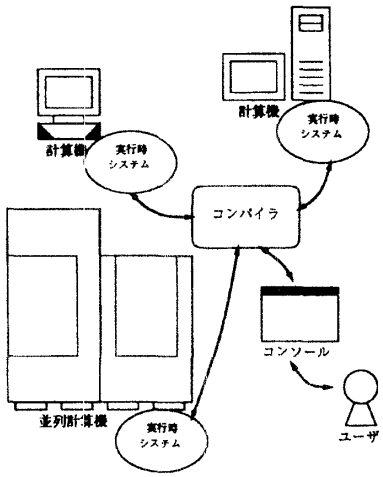


図 1: システム構成

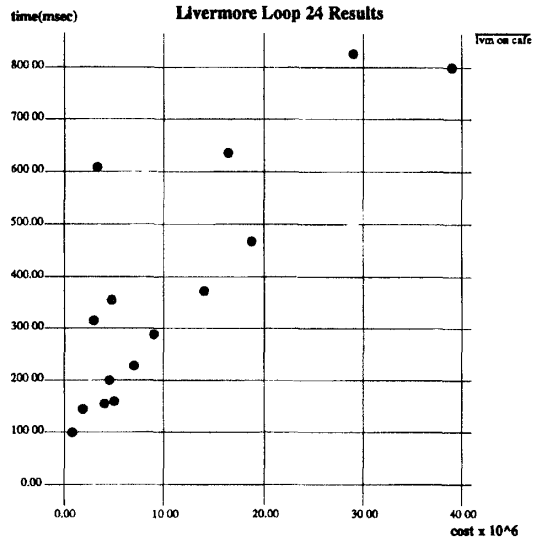


図 2: コストと実行時間

す。実行時システムは各計算機にデーモンとして起動され、実行環境に関する情報を取得、蓄積し、必要に応じてコンパイラに対してその情報を提供する役割を果たす。

そしてコンパイラとこの実行時システムの協力によるタスクの実行時間の見積もりは以下のように行われる。

1. ソースプログラムベースのタスクの計算コストを見積もる。

2. タスクが仮実行可能な条件を満たすかを調べる。ここでの条件とは、ループがある場合はそのイタレーション数を求めることができ、配列変数のアクセスも添字から範囲内に納まることが静的に判定できたものとする。

3. この条件を満たすものについては仮実行による実行時間計測を行う。コンパイラは実行時システムよりその計算性能を取得し、1. で求めた計算コストより実行時間を調整して実行時システムにそのタスクの中間表現を渡す。実行時システムはコード生成、実行、計測を行い、その結果をコンパイラに返す。

4. この条件を満たさなかった場合は、ソースプログラムベースの予測時間とする。

## 5. 考察

本手法の有効性を検証するため、Livermore Loop 24 に適用した場合について考察する。本コンパイラは構造化されていない IF 文や FORTRAN の組み込み関数をサポートしていないため、受け付けるプログラムが 24 個中 17 個であった。そのうちループ回数を見積もることができたものが 14 個であった。見積もったコストと実際の実行時間の関係を図 2に示す。この結果からコストからだけでもある程度見積もることができると考えられるが、幾つかのプログラムでは予測時間が大きくずれることが分かった。しかしこの 14 個のプログラムの内、13 個は配列変数のアクセス範囲の解析と定数伝搬により仮実行が可能なものであり、仮実行まで行えば 14 個のプログラムが比較的よい精度で見積もりが可能であると考えられる。

## 6. おわりに

本稿では分散環境における実行時間の見積もりについては実行環境を把握しておくことが重要であることを述べた。そして実行時システムがコンパイラと協力することでこの問題を解決する見積もり手法を提案した。また Livermore Loop 24 に本手法を適用した場合、本コンパイラが対応する 17 個中 14 個のプログラムについては精度のよい見積もりが可能であることが分かった。なお現在、実行時システムとコンパイラに必要とされる解析部の一部が未実装である。今後早急に解決し、評価を行っていく予定である。

## 参考文献

[1] Henri Casanova and Jack Dongarra: "NetSolve: A Network Server for Solving Computational Science Problems", the proceedings of Super Computing 96.

[2] Renzo Davoli, Luigi-Alberto Giachini, Ozalp Babaoglu, Alessandro Amoroso and Lorenzo Alvisi: "Parallel Computing in Networks of Workstations with Paralex", IEEE trans. of Parallel and Distributed Systems Vol. 7 No. 4, April 96.