ハイパフォーマンス
コンピューティング 77－14
（１９９９．８．３）

異機種並列分散システム向けプログラミング環境

末広 謙二，松浦 健一郎，菊地 賢太郎，村井 均，妹尾 義樹

新情報処理開発機構　並列分散システム NEC 研究室

異機種並列分散システムを簡単に効率よく利用するための、プログラム開発環境について述べる。本環境は並列分散ライブラリ、並列分散化コンパイラ、並列分散化支援ツールからなり、これらが連携してプログラミング上の障害となる種々のシームを取り除く。ツールは SX-4 と Cenju-4 の結合システムをターゲットに、現在までに、静的プログラム情報の可視化機能、自動データレイアウト機能、静的性能予測機能が実装され、さらにプログラム実行時情報の採取と利用のためのフレームワークが実装されている。採取された実行時情報はツールやコンパイラにフィードバックされ、プログラムのさらなる最適化や性能予測の精度向上のほか、動的再コンパイルによる実行時最適化に用いる予定である。

A Programming Environment for Heterogeneous Parallel and Distributed Systems

Kenji SUEHIRO, Ken-ichiro MATSUURA, Kentaro KIKUCHI, Hitoshi MURAI, Yoshiki SEO

Parallel and Distributed Systems NEC Laboratory, Real World Computing Partnership

This paper describes a programming environment for heterogeneous parallel and distributed systems. The environment consists of three parts; a parallel library, a parallelizing compiler and a parallelizing support tool. The tool currently has the features of: (1) visualization of program information, (2) automatic data layout, (3) static performance prediction, and (4) gathering and utilizing run-time program information. The run-time information is fed back to the tool and the compiler for static program optimization and performance prediction, and it will be also used for run-time program optimization by dynamic recompilation.

1. はじめに

大規模化・複雑化の一途をたどる数値計算アプリケーションのニーズに応えるため、ベクトル機とスカラ並列機、あるいは共有メモリ型並列機と分散メモリ型並列機といった、異なる特徴を持つ計算機を高速ネットワークで接続した、異機種並列分散システムに対する期待が高まっている。異機種並列分散システムでは、たとえばベクトル機

上で規則的な計算を、スカラ並列機上で不規則な計算をというように、各ノード計算機の特徴を活かしながら、複合的巨大計算を高性能に実行することができると期待されている。

しかし、異機種並列分散システム向けのプログラムを開発することは容易ではない。各ノード計算機のアーキテクチャの違いや性能差、ノード内とノード間にわたる複数レベルの並列性、刻々と変化するシステムの状態などをうまく扱わないと、システムの規模に見合った高い性能を得ることができないからである。異機種並列分散システムに存在するこれらの「継ぎ目（シーム）」とも呼ぶべきものは、プログラマにとっては大きな障害である。そこで、これらの「継ぎ目」を意識せず、シームレスにプログラム開発が行なえるプログラミング環境が求められている。

我々は、共有メモリ型ベクトルスーパーコンピュータ SX-4 と分散メモリ型スカラ並列計算機 Cenju-4 の結合システムをターゲットに、言語処理系技術をベースとするシームレスな異機種並列分散システム向けプログラミング環境の構築に取り組んでいる[1][2]。本稿では、2 章でこのプログラミング環境全体の概要について述べ、3 章ではそのうちツール部分の主要な機能の一つである実行時情報の採取と利用の枠組について説明する。4 章ではツールの実装について説明し、5 章ではまとめと今後の予定について述べる。

2. プログラミング環境の概要

我々が目指すプログラミング環境の概要を図 1 に示す。本プログラミング環境は、シームレス並列分散ライブラリ、シームレス並列分散化コンパイラ、シームレス並列分散化支援ツールからなり、これらが連携してプログラミング上のシームを取り除く。

2.1 シームレス並列分散ライブラリ

シームレス並列分散ライブラリは、異機種複数ノードとノード内並列プロセッサから構成される階層性のあるシステム上で、ノード間／ノード内のシームレスな通信を実現する。API としては MPI (Message Passing Interface) を採用している。ノードごとのアーキテクチャの違いによる数値データのバイトオーダーや浮動小数点形式の違いなどもここで吸収される。

次節で述べるシームレス並列分散化コンパイラは、通信プリミティブとしてこの異機種 MPI を使用する。

2.2 シームレス並列分散化コンパイラ

シームレス並列分散化コンパイラは、HPF 言語をベースに、異機種並列分散処理をプログラム文面上に記述することができるよう拡張した、独自言語のコンパイラである。システムの異機種性を積極的に活用したい利用者は、拡張された文法を利用して、異機種並列分散システムの任意のノードにデータや処理を割り当てることができる。異機種並列分散システムをシームレスに扱いたい利用者は、

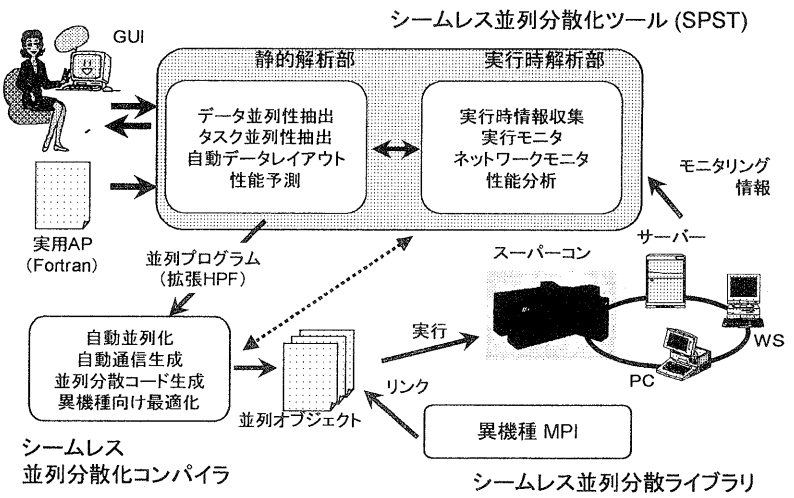

図 1. 異機種並列分散環境向けプログラミング環境

通常のHPF言語によりプログラムを記述すれば、コンパイラが適切なノードにデータや処理を割り当てる。本コンパイラの出力は、異機種並列分散システム上で動作するMPMDコードである。本コンパイラは以下のような処理を入力プログラムに対して行なう。

・プログラムに記述された論理プロセッサと、異機種並列分散システムの物理プロセッサとのマッピング
・プログラムに明示的に指示されなかったデータや処理の、プロセッサへの割り付け
・ループやタスクの並列実行可能性の判定とスケジューリング
・並列実行に必要な通信やデータ再配置コードの挿入

また、後述するシームレス並列分散化ツールと連携して、

・プログラム実行時情報を採取するためのコードの挿入

を行なう。

2.3 シームレス並列分散化支援ツール

シームレス並列化支援ツール（Seamless Parallelization and Distribution Support Tool、以下SPST）は、GUIベースのプログラミング支援ツールであり、前述のシームレス並列化コンパイラ、シームレス並列ライブラリと連携して、逐次Fortranで記述されたアプリケーション・プログラムの並列分散化を支援する。

SPSTは、逐次FortranまたはHPFで記述されたアプリケーション・プログラムを入力として受け取り、GUIにより利用者と対話しながらプログラムの並列分散化を行なう。SPSTの出力する並列Fortranコード（拡張HPFコード）をシームレス並列分散化コンパイラによりコンパイルし、シームレス並列分散ライブラリとリンクすることにより、異機種並列分散システム上で動作する目的コードを得ることができる。

SPSTには現在のところ、以下の機能が実装されている。

(1) プログラム情報のビジュアル表示と、並列化指示のビジュアル操作

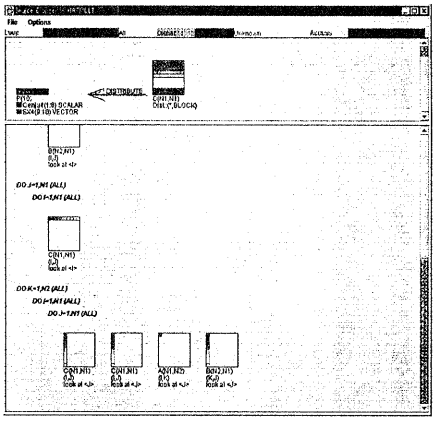

図2. プログラム情報のビジュアル表示

プログラムの論理構造、配列の分散、ループの並列化状態、通信の発生個所などのプログラム情報をビジュアルに表示する。また、利用者はドラッグ＆ドロップにより配列の分散が指示できるなど、直感的な操作によりプログラムの並列分散化を進めることができる。

図2にプログラム情報のビジュアル表示の例を示す。

(2) 自動データレイアウト

プログラム中のすべてのデータについて、適切なプロセッサへの割り当てを自動的に決定することができる。また、利用者がプログラムに関する知識を利用して、主要なデータの分散を適切に指示すれば、それに合わせる形でより最適なデータレイアウトを決定することができる[3]。

(3) 静的性能予測

プログラムの並列化チューニング作業の途中で、いつでもおおよその性能見積もりを表示することができるので、利用者は作業の方向性の正しさを確認することができる。また現在は実装されていないが、並列分散化コンパイラ内のプログラム最適化処理において、最適化方式に複数の候補がある場合に、どの方式を採用すべきかの判断のため、この性能予測機能を用いることを計画している。

(4) 実行時情報の採取と利用

プログラムの最適化に利用できる情報のうち、プログラムの文面の解析だけでは得ることのできない（実行時にしかわからない）情報を収集するためのフレームワークが実装されている。実行時情報は、プログラムの実行後または実行中に SPST のランタイムマネージャにフィードバックされ、プログラムの最適化等に利用される。

本機能の詳細については次章で詳述する。

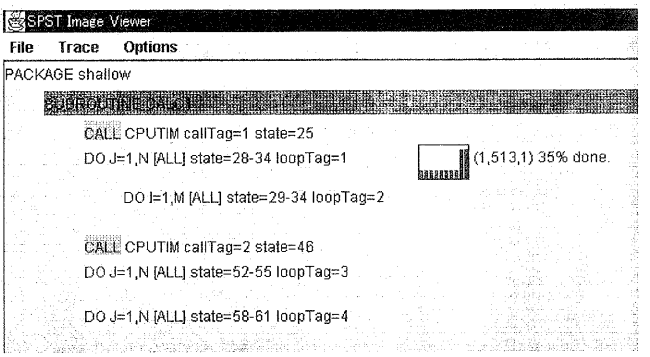

図 3. ループ進捗表示

3. 実行時情報の採取と利用

3.1 実行時情報の採取

SPST には、異機種分散環境下で、プログラム実行時の情報を採取するためのフレームワークが実装されている。現在、Cenju-4 上での実行モニタリングが可能となっており、引き続き SX-4 上でのモニタリング機能を実装中である。

SPST では、対象プログラムに実行時情報採取用の手続きを挿入した試行プログラムを自動生成し、実行することで実行時情報を採取する方式を採っている。

利用者は GUI を用いて、元のソースコード上のどのポイントで、どのような情報を採取したいのかを指示することができる。また、SPST 内の他の機能モジュールが、必要な実行時情報の採取を実行時情報採取モジュールに対して依頼する。実行時情報採取モジュールはこれらの要求に従い、ターゲットマシン用の試行プログラムを自動的に生成した後、これを実行し、実行時情報を出力させる。並列プログラムの場合には、プログラムを並列実行する各ノードから実行時情報が同時に出力されるが、これら複数ノードから発生する実行時情報の加工・集計も実行時情報採取モジュールが行なう。

採取された実行時情報は、自動データレイアウトモジュール、性能予測モジュールで利用されるとともに、GUI により利用者に提示される。また、この実行時情報採取フレームワークは、実行中にプログラムを再コンパイルしながら常に最適な形へとプログラムを適応的に変化させる、動的再コンパイル最適化方式実現のための基礎技術として用いる予定である。

SPST では、現在のところ以下の 6 種類の実行時情報を採取することができる。

- 文実行回数
- 変数の値の平均値
- 実行中の変数値
- 手続き実行時間
- ループ実行時間
- ループ処理進捗状況

プログラムの実行中にループの進捗状況をリアルタイム表示させた例を図 3 に示す。プログラム構造表示の右側で縦に伸びるプログレスバーの列がプロセッサごとの進捗状況を、その右のパーセント表示が全体としての進捗状況を示している。

3.2 実行時情報の利用例

SPST の自動データレイアウト機能では、プログラム中で頻繁に実行される部分を優先してデータの分散配置を試みる[3]。このとき、プログラムのどの部分の実行回数が多いかを知るために、実行時情報を利用する。

図 4 のプログラムでは、IF 文によって呼び出されるサブルーチンが切り換えられている。ここで、サブルーチン subx と suby で引数の配列 a, b の最適な分散が異なる場合、呼び出し元での a, b の最適な分散は、IF 文が成立する確率によって変わる。

```
if (f(t).eq.1) then
   call subx(a,b)
else
   call suby(a,b)
endif
```

図 4. サンプルプログラム

SPST の自動データレイアウト機能は、実行時情報（文の実行回数）を利用して、subx, suby のうち多く実行される方に a, b の分散を合わせることにより、最適な分散を選択する。

4. 実装

SPST の内部構成を図 5 に示す。SPST の各機能モジュールは、独立したサーバプログラムとして実装されており、各機能サーバが通信サーバを介して「SPST プロトコル」と呼ぶ手順に従い通信を行なうことにより、協調動作する構成になっている。このような構成をとることにより、

・機能ごとの独立性が高まることで、プログラマによる直接利用ばかりでなく、ツール内外他の機能モジュールからの呼び出しなど、柔軟な利用が可能になる。
・新規機能の追加や既存機能の拡張が容易になる。
・異機種並列分散環境下において、SPST を容易に機能分散させることができる。

といった利点が得られる。

(1) 通信サーバ

すべての機能サーバ間の通信を仲介する、SPST の根幹部分である。

(2) ユーザインタフェース部

利用者の指示を SPST に伝え、また SPST からの情報を利用者に提示する窓口となる。

(3) 自動データレイアウトサーバ

ユーザインタフェース部からの指示を受け、入力プログラム中で用いられているデータの分散を自動的に決定する。入力プログラムの静的な解析情報に関してはプログラム情報マネージャ、実行時の情報に関しては動的情報マネージャからそれぞれ提供を受け、結果となるデータのレイアウトをプログラム情報マネージャに対して登録する。

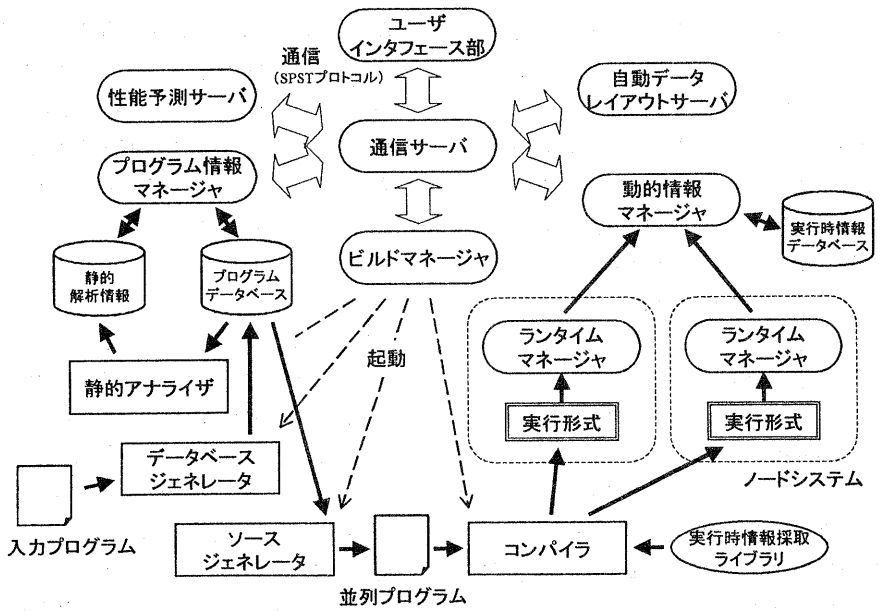

図 5. SPST の構成

(4) 性能予測サーバ

ユーザインタフェース部からの指示を受け、入力プログラムの実行性能を予測する。自動データレイアウトサーバと同様に、プログラム情報マネージャ、動的情報マネージャからそれぞれ静的解析情報、実行時情報の提供を受ける。結果は現在のところ、直接ユーザインタフェース部に返され表示される（将来は他の機能モジュールからの利用も見込んでいる）。

(5) プログラム情報マネージャ

入力プログラムの三つ組レベルの中間表現であるプログラム・データベースと、コントロールフロー・グラフや依存解析情報などの基本的な静的解析情報を管理し、他の機能サーバから要求によりこれらのデータの参照や更新を行なう。要求されたデータがデータベース中に存在しない場合、またはプログラム最適化変形などに伴ってデータベースが陳腐化した場合には、ビルドマネージャに対してデータベース再生成のための外部コマンド起動を要求する。

(6) 動的情報マネージャ

ユーザーインターフェース部や自動データレイアウトサーバ、性能予測サーバに対し、要求された実行時情報の提供を行なう。実行時情報採取のための試行プログラムの生成・起動をビルドマネージャに対して要求するとともに、試行プログラムの実行時に各ノードシステム上に配置されたランタイム・マネージャから時々刻々通知される実行時情報を、実行時情報データベースに蓄積する。

(7) ビルドマネージャ

動的情報マネージャからの指示に従い、実行時情報採取用の試行プログラムの生成処理（ソースコード生成、コンパイル、リンク）を制御する。またプログラム情報マネージャからの指示により、静的解析情報の生成処理（静的アナライザの起動）を制御する。

(8) ランタイムマネージャ

異機種並列分散システムの各ノード上に配置され、実行時情報を収集し、複数ノード間にまたがる実行時情報の集計・加工を行ない、生成した情報は動的情報マネージャに送信する。

(9) 周辺コマンド群、ライブラリ

対象となる Fortran プログラムを読み込んでプログラム・データベースを作成するデータベース・ジェネレータ、プログラム・データベースから静的解析情報を作成する静的アナライザ、プログラム・データベースから HPF プログラムを生成するソース・ジェネレータ等は、外部コマンドとして実装されており、ビルドマネージャの制御下で実行される。試行プログラム作成のための HPF コンパイラやリンカも同様である。

また、実行時情報採取のための試行プログラムにリンクされる、実行時情報採取ランタイム・ライブラリが各ノードシステム用に用意されている。

5. まとめ

異機種並列分散システム向けプログラミング環境の概要と、ツール部分の機能および実装について述べた。今後は、実装済みの実行時情報採取・利用の枠組を利用して、実行時情報を利用した種々のプログラム最適化方法を考案・評価するとともに、動的再コンパイル最適化方式の実現へ向けて研究を進める予定である。

参考文献

[1] 草野ほか：異機種並列分散処理による高性能計算機システムの構想, SWoPP97, 信学技報 CPSY97-63, pp.91–96, 1997.

[2] 片山ほか：高性能並列分散システムの設計, 1998 RWC シンポジウム, RWC Technical Report (TR-98001), pp.41–46, 1998.

[3] 松浦ほか：データ並列プログラムに対する高速な自動データ分割手法, JSPP'99, pp.87–94, 1999.