情報工学系学部学生に対する並列プログラミング演習教育

朝倉宏一† 渡邉豊英†

我々は，1995 年度から 3 年間にわたって本学情報工学科の 3 年生に対して，富士通社製の分散メモリ型並列計算機 AP1000 を用いて並列プログラミング演習を行った．本並列プログラミング演習は「プログラミング演習」という授業の一部として実施された．「プログラミング演習」は学生のプログラムの作成能力向上のために設けられ，卒業研究やその後の研究に必要となる問題に対処可能なプログラミング能力の習得を目的としている．並列プログラミング演習は，1997 年度には 4 回，1995 年度，1996 年度には 5 回の授業で行われ，基本的な並列プログラミング環境やツールの習得，数値計算問題によるプログラミング演習などが行われた．この演習を通じて概略以下のようなことが明らかとなった．まず，並列処理に特有かつ困難な対象問題の分割，プロセッサの割当て問題や，非決定性問題などがあるが，これでつまずく学生はそれほど多くなかった．しかし，多くの学生は応用的なプログラムを作成することに慣れておらず，並列処理以前のポインタの使用法，ライブラリの使用法，メモリ管理などでつまずくことがあった．すなわち，学部学生に対して並列処理の演習を行うことは，演習のレベルとしてはそれほど問題なく，新たな計算機を経験させるという意味でも有効であった．

Class-exercise of Parallel Programming for Undergraduate Students in Computer Science

Koichi Asakura† and Toyohide Watanabe†

We have applied the parallel programming exercise to 3rd-grade-undergraduate students on Fujitsu AP1000 from 1995 to 1997. This exercise was taken place as a part of the lecture named "programming exercise". In this paper, we describe the outline of our parallel programming exercise. For the basic exercise of parallel processing, we prepared 4 lecture units in 1997 and 5 lecture units in 1995 and 1996. In these classes, the basic introduction of parallel processing, the introduction of programming environment and tools for parallel processing, and the exercises for parallel processing such as numeric analysis problems were planned. In this parallel processing programming exercise, we can find out the following effects. First, many students can cope appropriately with the parallel-processing-specific topics such as decomposition of given problems, processor allocation, undeterministic problem and so on. Second, many students cannot deal with the programming issues which do not relate directly to parallel-processing strategies themselves: pointer variable utilization, library utilization for parallel processing, memory management and so on. Overall, we can conclude that the parallel programming exercise for the undergraduate students is very effective.

1. はじめに

近年，情報工学系のカリキュラムでは並列・分散処理に関する話題が盛り込まれ，並列プログラミング，並列処理の情報教育の必要性が叫ばれている．たとえば，ACM により 1984 年に提言された CS1[1]，CS2[2] カリキュラムには並列・分散処理が言及されていなかったが，1991 年に提言された Computing Curricula '91[3] では，アルゴリズムとデータ構造，オペレーティングシステム，プログラミング言語の各教授要目において，並列・分散処理の話題が盛り込まれている．また，最近のオペレーティングシステム[4] や計算機アーキテクチャ[5] に関する教科書には，並列・分散処理のまとまった記述が見られるようになった．さらには，近年のインターネットにはじまるネットワークの普及に対して情報システム，情報ネットワーク技術の必要性が叫ばれている．計算機ファシリティの面からも，情報工学系のほとんどすべての学部・学科には並列計算機やワークステーション・クラスタによる分散処理環境が整備され，それらを有効に活用するという点からも，並列・分散処理の専門教育が重要となっている．

並列・分散処理に関する教育は少しずつではあるが，様々な講義で行われている．しかし，実際のプログラ

† 名古屋大学大学院工学研究科情報工学専攻
Department of Information Engineering, Graduate School of Engineering, Nagoya University

ミングをともなった並列・分散処理の演習・実験はほとんど行われていないのが現状である．各講義において，並列・分散処理に関する基本的な知識，考え方は教授されている．しかし，実質的な能力を養うためにはプログラミング演習が必須である．並列処理のプログラミングでは，逐次処理のプログラミングとは異なり，実際に動作する主体が複数存在することを理解しなければならない．すなわち，複数のプロセスの並列実行がオペレーティングシステム上で仮想的に行われるのではなく，実際に別々のプロセッサ上で行われる．並列プロセスを矛盾なく動作させるためには，プロセス間の同期手法やプロセス間メッセージの到着順序の非決定性などを理解する必要がある．並列処理では逐次処理にはない新しい概念の学習が必須であり，しかもそれらの現象は実際にプログラムを作成し実行させることで表面化する．したがって，並列処理の教育ではプログラミング演習により理解を深めることが重要であり，効果的である．

我々は，並列処理教育の一貫として1995年度から1997年度の3年間にわたり，本学情報工学科の3年生に対して,「プログラミング演習」の一部として並列プログラミング演習を行った．本稿ではこの並列プログラミング演習における目的，内容，評価などについて報告する．演習には富士通社製の分散メモリ型並列計算機AP1000（16セル）が使用され，プログラミング言語Cを用いてコーディングを課し，各デバッギング・ツールの下でプログラムを作成させた．

本稿は以下のような構成になっている．まず，2章において並列プログラミング演習計画における基本方針について述べる．どのような視点により各種課題を設定したかについてまとめる．続く3章，4章では演習で実施した課題内容について具体的に述べる．先にも述べたように，並列プログラミング演習は1995年度から3年間にわたって実施されたが，ここでは1997年度の課題内容について詳しく述べる．5章では並列プログラミング演習の結果について述べる．最後に6章においてまとめを述べ，並列プログラミング演習を行う際に必要なプログラミング環境について言及する．

2. 並列プログラミング演習の基本方針

まず，並列プログラミング演習を計画・実施するうえでの基本方針について述べる．本学科のカリキュラムを**表1**に示す．プログラミング演習が開講されるまでのプログラミング教育は概略以下の流れで行われている．

表1 本学科のカリキュラム
Table 1 Curriculum of our department.

学年	講義
1年生前期	電気・電子・情報工学序論*，離散数学及び演習，図学*
後期	計算機リテラシ及びプログラミング，線形回路論及び演習，力学及び演習*
2年生前期	数学1及び演習，論理回路及び演習，確率・統計，計算機システム，プログラミング第2
後期	数学2及び演習*，数値解析第1及び演習，数理論理学及び演習，オートマトン理論及び演習，計算機ハードウェア，情報工学実験第1†
3年生前期	伝送と符号理論，アルゴリズムとデータ構造，パターン情報処理，オペレーティングシステム，コンパイラ，計算機アーキテクチャ*，信号・音声処理*，数値解析第2*，生体情報処理*，非手続き型言語*，情報工学実験第2††
後期	データベース*，画像処理*，情報システム*，計算機ネットワーク*，知識情報処理*，信頼性工学**，数理計画法**，情報工学実験3
4年生前期	計算機と社会*，電子回路*，自動制御**，機械工学通論**，卒業研究
後期	卒業研究

（注） * ...選択科目
** ...関連専門科目（選択科目）
† ...ハードウェアの基礎，ソフトウェアの基礎（テキスト・フォーマッタの作成）
†† ...マイクロコンピュータ，コンパイラの設計と製作

初歩 ... 計算機リテラシ及びプログラミング（1年生後期）
基礎 ... プログラミング第2（2年生前期），情報工学実験第1（2年生後期），情報工学実験第2（3年生前期）
応用 ... 情報工学実験第3（3年生後期）

これらのほか，各講義における演習，宿題でもプログラムの作成が課されている．

このようにプログラミングの初歩から応用までを教育していたが，学生のプログラミング能力は4年生での卒業研究に対し，十分であるとは必ずしもいえなかった．アルゴリズムが与えられた，ある特定の処理をするプログラムを作成する能力は育成していたが，それらを組み合わせる能力，特にモジュール間の実行制御や，メモリ・ファイル管理などの能力は不十分であった．すなわち，処理・機能に関したプログラミング能力は育成してきたが，管理・制御に関したプログラミング能力はあまり育成していなかった．したがって，卒業研究などにおいて，ある処理アルゴリズムに従った基本的なプログラムを作成することはできるが，システムを視野に入れた応用的なプログラムの作成は困難であった．

そこで，卒業研究やその後の研究に必要となる応用的な問題に対処可能な，システムを視野に入れたプログラミング能力の習得を目的とし，3年生後期にプログラミング演習が実施された．演習内容として，B木などを用いたファイル，インデックス管理，LISPを用いた数式処理システムの開発，遺伝的アルゴリズムなど，様々な課題が設定された．

並列プログラミング演習は，プログラミング演習の一部として計画され，1回3時間の演習が4回割り当てられた☆．したがって，ある程度の基礎的なプログラミング能力を持った学生に対し，応用能力として並列処理における基本的なものの考え方を習得させ，応用的プログラミング能力の向上を図ることを目指した．特にプログラミング演習の目的である管理・制御に関するプログラミングを行うために，複数のプロセッサを効果的に動作させ，プログラムの実行効率を向上させることを最終的な目標とした．単に課題が解けるだけでなく，その課題を解くために効果的な処理アルゴリズム，通信アルゴリズムを選択し，実行時間の短いプログラムの作成を目指した．つまり，プログラムの機能面だけでなく，性能面をも意識させてプログラミング演習を実施することを基本方針とし，卒業研究以降の研究に対処可能な応用的プログラミング能力を養うことを目標としている．この基本方針に対する並列プログラミング演習の課題内容について以下に述べる．

2.1 並列処理に関する導入説明

効果的に並列プログラミング演習を実施するためには，並列処理に関するハードウェア，ソフトウェア両面の知識が不可欠である．現在，並列処理それ自身に関する講義は開講されていないので，実際の演習に先立ち基本的な解説が必要となる．しかし，並行・並列処理に関する基礎的な話題は，オペレーティングシステムや計算機アーキテクチャ，プログラミング言語の講義において触れられている．たとえば，オペレーティングシステムであれば，プロセス管理やマルチ・プログラミングの章において並行・並列プロセスについて講義されている．さらに，最近の教科書では，並列・分散処理についての章が設けられている．したがって，並列処理に関する基礎的な知識は，断片的ではあるがすでに得られているので，本演習内での並列処理に関する解説は必要最小限にとどめることとした．

2.2 学習実験と体得的演習

文献6)に指摘されているように，情報処理教育における実験には，学習実験と体得的演習があり，両者をうまく共存させることが重要である．学習実験とは，機器の使用法や実験の手順を習得するための実験である．一方，体得的演習とは，課題のみを与え，具体的な課題の実現手法などは学生に委ねられる形式のもので，学生の創造性，独自性が期待される演習である．通常，プログラミング演習は体得的演習に分類される．しかし，すべての学生が並列プログラミング演習において初めて並列計算機を使用することになり，並列計算機を用いたプログラミングは通常の学生実験で行われているワークステーションを用いたそれとは異なる．したがって，実験機器として並列計算機を使用するための学習実験も必要となる．すなわち，並列プログラミング演習においては，プログラムの翻訳方法や並列計算機でのプログラムの実行方法，プログラムの実行状態の可視化ツールなどの使用法を習得する学習実験と，与えられた問題を解決する並列プログラムを作成する体得的演習の両方を，バランスよく組み合わせなければならない．

また，管理・制御に関するプログラミングのためには，プログラムの機能面だけでなく，性能面を意識させる体得的演習でなければならず，また，そのために必要なプログラミング環境の機能を学習可能な課題内容でなければならない．つまり，プログラムの修正がプログラムの性能に対しどのように影響するかを体得可能な課題内容であることが必要である．並列処理の場合，各プロセスでの処理アルゴリズムや，ネットワークを介したプロセス間の通信手法，プロセス間の同期手法の違いがプログラムの実行効率にどの程度影響するかを理解させることが重要である．そして，各種手法の違いやプログラムの実行効率の違いを容易に確認できる能力を持たせるために，学習実験においてプログラム解析ツールやデバッグツールなどの習得が必要である．

2.3 体得的演習の効果的実施

プログラミング演習のような体得的演習には，マンツーマン的な指導が効果的であることが知られている[6]．しかし，現状では不可能であり，多人数の学生に対し，体得的演習を実施するには様々な工夫が必要となる．特に，プログラムは容易に複製可能であるので，意義ある演習のためには学生のやる気を高める工夫が必要である．文献6)に指摘されているように，学生は他の学生の演習の到達点を自分の到達点と比較

☆ 実際には，1997年度の並列プログラミング演習には10回が割り当てられていた．後半の6回は並列処理応用演習としてX Window Systemと並列プログラミングの演習を行った．本稿では，1995年度，1996年度の演習との比較という観点から，前半の並列処理基礎演習のみについて説明する．1995年度，1996年度の演習回数は5回であった．

することで，自分の足りない点，気づかなかった点などを自分の力で発見し，体得することができる．この過程は体得的演習において非常に重要であり，効果的である．演習の過程において，各自の課題の到達点を互いに比較できる環境を用意することで効果的な演習が期待できる．並列処理の場合，プログラムの実行時間という，演習の到達点を計る絶対的で客観的な尺度があり，プログラムの性能面も評価可能である．したがって，プログラムの実行時間を比較する環境を用意し，効率良く体得的演習を進めることを目指した．

3. 並列プログラミングの学習実験

前章で述べた基本方針に従い，1995 年度から 1997 年度の 3 年間，並列プログラミング演習を実施した．3，4 章では，並列プログラミング演習における演習内容について詳しく述べる．本稿では，1997 年度に実施した課題内容を中心に説明する．1997 年度の課題内容を**表 2** に示す．まず 3 章では，演習の 1 回目と 2 回目の前半を用いて行われた並列プログラミングの学習実験について述べる．次の 4 章では，2 回目の後半と 3，4 回目で行われた並列プログラミングの体得的演習について述べる．

3.1 並列処理の解説

演習の 1 回目では，並列プログラミング演習において必要な並列処理の話題について解説した．解説した内容は以下のとおりである．

並列処理の目的 並列処理の目的は処理時間の短縮であり，単に複数の計算機を同時に動作させることではないことを解説した．

並列処理の方法 並列プログラミングでは，プログラムや問題をどのように分割し，並列実行可能な単位を生成するかが重要となる．ここでは主要な方法であるデータ並列処理と機能並列処理について解説した．このとき，はがきの宛名書きの例を用いて各処理について説明することで，概念を容易に理解できるよう工夫した．

並列処理に必要な機能 並列プログラミングにおける必要な機能として，プロセス生成機能，プロセス間通信機能，プロセス間同期機能をあげ，それぞれについて説明した．これらの機能を効率良く実現することが実行効率の良いプログラムを作成する鍵となるので，この時点でそれぞれの機能について印象づけることとした．

並列プログラミングの過程 課題を並列処理により解く場合に必要となる処理を，問題分析，プログラム設計，プログラム実装の各過程についてまとめて解説した．このように，問題の分析段階から並列処理について考えるプログラミングを指導することにより，効果的で効率の良いプログラミングが可能となる．

AP1000 の解説 並列計算機のアーキテクチャについて概説し，それぞれのアーキテクチャの特徴をまとめた．そして，演習で使用する並列計算機 AP1000 について解説した．分散メモリ型並列計算機であること，3 種類の高速ネットワークにより接続されたプロセッサ・エレメントであるセルが 16 台備わっていること，全体を制御するためのホスト計算機が存在することなどを説明した．最後に，AP1000 における並列プログラミングでは，ホスト・プログラムとセル・プログラムの 2 種類のプログラムを作成しなければならないことを説明し，それぞれのプログラムの役割について解説した．

日常的な例により，データ並列処理，機能並列処理を解説することで，並列処理の手法，およびその概念が容易に理解可能となった．また，並列プログラミングの過程において，問題の分析段階から並列化を意識させることにより，効果的な並列処理が期待できる．

表 2 課題内容
Table 2 Exercise subject.

回	1997 年度演習内容
1	並列処理の解説，AP1000 の使用法
2	プログラミングの方法，ライブラリの使用法，並列プログラミング
3	並列プログラミング
4	プログラムの性能解析

3.2 並列計算機使用の学習実験

本演習が並列計算機を使用する最初の機会であるので，並列計算機の使用法，すなわちプログラムの翻訳方法，プログラムの実行方法などに関する学習実験を採り入れなければならない．したがって，演習の 1 回目の後半と 2 回目の前半において，並列プログラミングの学習実験を行った．まず初めに，並列プログラムの実行環境として，ワークステーションで動作可能な並列計算機シミュレータと，AP1000 の 2 種類が提供されていることを説明し，それぞれの実行環境の使い分けについて注意した．本演習は多人数によるものであるので，1 台しか装備されていない AP1000 を効率良く公平に使用させるためには，並列計算機シミュレータによるワークステーション上の実行環境が必須である．そこで，それぞれの環境の違いと使用法を説明し，各環境における並列プログラムの翻訳方法，プ

ログラムの実行方法について解説した．そして，既存の例題プログラムを用いて，各環境におけるプログラム翻訳，プログラム実行を実習させた．また，学習実験後，並列計算機シミュレータの動作限界やAP1000使用上の注意点について述べた．実際に，両実行環境で並列プログラムを翻訳，実行することにより，プログラミング環境の使用法，その差異について学習でき，次に続く並列プログラミングの体得的演習に対する導入としての学習実験とした．

並列プログラミングにおける重要なツールとして，プログラム解析・可視化ツールがある．しかし，この導入時の学習実験では，このツールについては説明しなかった．これは，プログラム解析・可視化ツールは実際に使用する状況が発生したときに学習した方が学習効果が高いと判断したからである．

4. 並列プログラミングの体得的演習

並列処理の概略の説明や並列プログラミング環境の学習実験の後，演習の2回目の後半以降で実際のプログラミングをともなう体得的演習を実施した．本章では，この体得的演習について述べる．演習の効果を高めるため，体得的演習を前半と後半に分けて実施した．前半では例題プログラムの改造を通して，並列処理ライブラリの使用法を演習した．後半は課題として問題の仕様のみを与え，プログラムを作成させた．また，この演習に対する学生の意欲を高めるため，作成したプログラムごとに評価基準の1つである実行時間を比較させる実行時間コンテストを実施した．このとき，演習の4回目でプログラムの性能解析の方法について演習した．

4.1 前　　半

前半では，まず例題プログラムを示し，それを問題に従い各自で改良する過程において，並列プログラミング環境の学習実験を兼ねつつ，並列処理ライブラリの使用法，関数の機能の相違などを体得させることを目標とした．作成したプログラムを実行したときの挙動の差異と，使用した並列処理ライブラリとの関係を比較，検討させることで，並列処理ライブラリ中の各関数の機能，特徴などを理解させる．これにより，次に続く後半の演習への導入が容易となる．例題として，最小値，最大値を求める問題を採用し，様々な処理アルゴリズムに従ってプログラムを作成させた．具体的には以下のようである．

問　題　1

処理データをすべてのセルに同報通信する．その後，それぞれのセルが最小値，最大値を計算する配列の添字の範囲を同報通信する．各セルは担当範囲内のデータの最小値，最大値を計算し，結果をホストに送信する．ホストでは，各セルから受信した結果を集計する．□

この問題では，セル・プログラムの起動，ホストからセルへの同報通信，セルでのホストからの受信，セルからホストへの送信，ホストでのセルからの受信，の基本的な5種類のライブラリ関数を学習させ，AP1000における並列プログラムの作成に対してその基礎を学習させることを狙いとした．実際，典型的なマスタ・スレーブ型並列処理のプログラムであれば，上記5種類のライブラリ関数で作成可能である．

問　題　2

問題1では処理データをすべてのセルに同報通信していたが，それを各セルに処理データを分割して送信するようにプログラムを変更する．□

この問題では新たにホストからセルへの個別通信を学習させる．ホスト・セル間の同報通信と個別通信で使用するライブラリ関数の違い，使用法の差異などの学習が目的となっている．

問　題　3

問題1，2では各セルで計算した結果をホストへ送信し，ホストで集計していた．ここでは，セルで結果を集計し，ホストに集計結果を送信するように変更する．セルでの計算結果の集計方法は以下のようである．セル n はセル $n-1$ から受信した結果と自セルで計算した結果を集計し，集計結果をセル $n+1$ へ送信する．最後のセルは集計結果をホストへ送信する．□

この問題により，セル間通信の方法について学習させる．以上の問題により，並列プログラムにおけるプロセス間通信の基本形態，すなわち，ホスト・セル間通信，セル間通信，同報通信について，プログラミングを介してそれぞれの差異を学習させることができる．通信処理のライブラリ関数は多数存在するが，基本的には上記に分類できる．すなわち，他のライブラリ関数には必要な引数や実行時の挙動において微妙に違いが存在する☆のみであり，基本的な機能には差がない．したがって，この3問題により，基本的な並列プログ

☆ たとえば，データを送受信するセルを指定するとき，セルIDを二次元で表現するか一次元で表現するかの違いや，データ未受信のとき，データが受信されるまでライブラリ関数内で実行が中断するか否かの違いなどである．

ラミングが習得可能である．

これらの一連の演習において，セル・プログラムの非同期的な実行にともなう通信処理における非決定的な動作，すなわちメッセージの到着順序の非決定性について説明した．また，メッセージ送受信のオーバヘッドについても説明し，メッセージの送受信の方法によりプログラムの処理時間が異なることを学習させた．特に，メッセージの到着順序の非決定性については詳しく解説した．1995 年度，1996 年度の演習において，並列計算機シミュレータ上で動作するプログラムが並列計算機上で動作しない，あるいは処理結果が異なるという質問が学生から多く発せられた．その多くはメモリ管理が原因であったが，中にはメッセージの到着順序の非決定性によるものもあった．そこで，ある程度プログラムを作成した後で非決定性について解説することにより，並列プログラムの動作についてより理解が深まると考え，このような説明を施した．

4.2 後　半

前半の演習によって得られた基礎的な並列プログラミング能力を用いて，後半では問題の仕様のみを与え，プログラムを作成させた．ここでは，学生の問題理解力，プログラム設計能力，プログラム実装能力の向上を目指した．すなわち，与えられた問題をどのように並列化するかを検討し，問題に適切な並列化手法を分析・選択し，プログラムの実装において効率良く並列処理ライブラリを使用する能力，総合的なプログラミング能力の向上を目指した．与えた問題は文献 7) より引用した．

問　題　1

与えられたデータ（データ数 400 万）の分散，ねじれ度，尖度を計算するプログラムを作成する．ここで，それぞれの値は以下のように定義される．

$$\text{分散} = \frac{1}{n}\sum_{i=1}^{n}(a[i]-|a|)^2$$

$$\text{ねじれ度} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{a[i]-|a|}{std(a)}\right)^3$$

$$\text{尖度} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{a[i]-|a|}{std(a)}\right)^4 - 3$$

ただし，$|a|$ は a の平均値，$std(a)$ は a の標準偏差を表す．□

問　題　2

与えられたデータ（データ数 400 万）の中央値を求めるプログラムを作成する．ここでの中央値は

$$\sum_{i=1}^{j} a[i] = \sum_{i=j+1}^{n} a[i]$$

となる j，あるいはこれにできるだけ近い j とする．□

問　題　3

与えられた 2 つのデータ列（データ数各 400 万）の相関係数を求めるプログラムを作成する．ここで，相関係数は以下のように定義される．

$$\text{相関係数} = \sum_{i=1}^{n}\frac{(a[i]-|a|)\cdot(b[i]-|b|)}{\sqrt{std(a)\cdot std(b)}}$$

□

この演習では，数式のみを与え，どのように計算するかの方法は指示しなかった．学生が各自でアルゴリズムを考案し，セルへの仕事の分配法を考察することを期待した．問題 1 では，3 種類の互いに関連した数式を，どのように効率良く計算するかがポイントである．単に，データ並列処理方式で計算することもできるし，各セルで異なった計算をさせる機能並列処理も適用可能である．問題 2 では，セルへどのように仕事を分配させるかが鍵となる．また，問題 3 は問題 1 と同様に，数式内の各項をいかに効率良く計算するかが重要となる．

通常の演習では，問題が解ければ，すなわちプログラムを作成，実行できればそれで終了とし，作成したプログラムを解析し，実行効率の向上を求めようとする学生は少ない．したがって，上に述べたような問題作成者が考えたポイントまで学生が達成せずに演習が終了し，体得的演習の実施効果が少ない場合がある．しかし，次に述べる実行時間コンテストの開催により，学生に対してプログラムの解析と改良を動機付け，演習の効果的な実施を目指した．

4.3 体得的演習の効果的実施

プログラミング演習のような体得的演習の場合，単にプログラムが作成できればよいわけではない．学生がプログラムを解析・改良し，その結果がどのように処理時間などに反映されるかを試行錯誤を繰り返しながら理解することで，演習の目的が達成される．特に，本プログラミング演習が目標とする管理・制御に関するプログラミングでは，作成したプログラムの機能的

な完成度はもちろんであるが，性能的な完成度も求められる．単にプログラムを作成するにとどまらず，そのプログラムの挙動を解析し，必要であれば処理アルゴリズムを再設計することで，実行時間の短縮や，使用メモリ空間の最小化を目指さなければならない．そして，プログラム内の各モジュールの制御方法やプログラム内で使用される計算機資源の管理手法を効率化するためにプログラムの改良を繰り返すことにより，プログラミング演習の目標が達成される．そのためには，学生に対してプログラムの解析・改良の動機付けが必要となる．体得的演習の場合，特に他人の演習の到達点を知ることにより演習効果が高まることが知られている[6]．これは，他人の到達点を知り，自分の到達点と比較・検討することで，自分の考えの足りなかった点や，自分が気づかなかった点を自分で発見することができるからである．そして，他人の演習の到達点との比較・検討により，自分のプログラムを解析・改良することが動機付けられ，効果的に効率良く課題内容を体得可能となる．

並列処理の場合，プログラムの到達点を計る基準としてプログラムの実行時間がある．並列処理の目的の1つとして実行時間の短縮があり，実行時間の短いプログラムは演習の到達点として高いレベルにあると考えることができる．したがって，並列プログラミング演習の到達点を計る1つの基準であるプログラムの実行時間を各自が公表し，比較・検討することで，演習の効果が高まる．

並列プログラミング演習における体得的演習の効果を高めるため，学生が作成したプログラムの実行時間を，Web ページを介して登録させ，その登録結果を Web ページとしてリアルタイムに閲覧可能とするシステムを開発し，実行時間コンテストを開催した．この Web ページを参照することで，他人の演習の到達点を参照することができ，それを自分の到達点と比較することができる．

また，実行時間コンテストの開催と並行して，プログラム解析・可視化ツールの使用法について説明した．単に，プログラム解析・可視化ツールを導入するのではなく，実行時間コンテストを開催し，作成したプログラムの挙動を解析する要求が生じた時点でツールを説明することで，能率良く理解させることが可能であると考え，この時点でツールの使用法を解説した．

学生が各自のプログラムを解析・可視化し，改良を加え，プログラムの実行時間を競い合って演習を遂行することにより，体得的演習の効果が高まり，並列プログラミング演習の実施効果を高めることができる．そして，これにより本プログラミング演習が目標とした管理・制御に関するプログラミング能力の向上が期待できる．

5. 演習の結果

前章までで説明した方針，計画内容により実施した並列プログラミング演習の結果について報告する．

5.1 1995 年度，1996 年度の課題内容

演習結果について述べる前に，1995 年度，1996 年度に実施された演習の内容について，1997 年度との比較で簡単に説明する．特に，以下の議論において必要となる相違点について簡単に説明する．

1995 年度，1996 年度とも，実施した演習は 5 回である．1997 年度と比較すると 1 回多いが，演習内容はほぼ同じで，プログラム作成，すなわち体得的演習の部分に時間をかけた．実行時間コンテストを実施したのは 1997 年度のみであり，1995 年度，1996 年度には実施しなかった．また，1996 年度は 1997 年度と同様にプログラム解析・可視化ツールの使用法について説明し，各自で作成したプログラムを改良する問題を課した．1995 年度は，プログラム解析・可視化ツールを使用せずに，プログラムを手作業で解析，改良する問題を課した．

5.2 レポートの得点による解析結果

図 1 に 1995 年度から 1997 年度までの並列プログラミング演習のレポートの得点分布を示す．また，**表 3** にはレポートの平均点，標準偏差を示す．参考として，同学期に開講された情報工学実験第 3 の成績も示す☆．レポートの評価は 100 点満点である．55 点を合格の最低基準とし，各問題の得点（採点基準は後述），考察内容，レポート全体の構成などを総合して評価されている．表を見ると，レポートの平均点は情報工学実験とほぼ同じだが，標準偏差の値が大きく，得点が分散していることが分かる．これは並列処理に対して興味を持った学生とそうでない学生の差と考えられる．すなわち，並列処理に興味を持った学生は演習に対して多くの時間を割き，その結果レポートの得点も高くなった．反対に，並列処理にあまり興味を示さなかった学生は，演習，レポート作成にあまり時間を使わず，その結果，得点が低くなっていると考えられる．これにより，得点の分布が広範囲となり，その結果標準偏差の値が大きくなったと考えられる．この傾向は 1995

☆ 情報工学実験第 3 は必修科目であるが，プログラミング演習は必修科目ではない．また，情報工学実験第 3 では，画像認識，エキスパート・システム，自然言語インタフェースなどの 11 課題から 3 課題を選択して実験する．

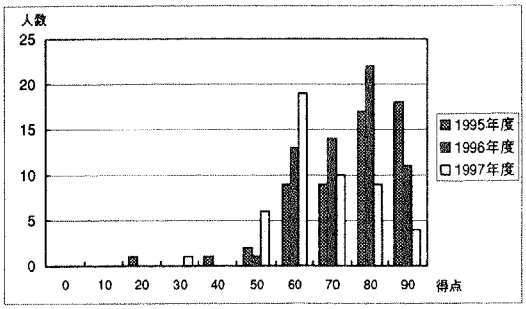


図 1 レポートの成績分布
Fig. 1 Histogram of report score.

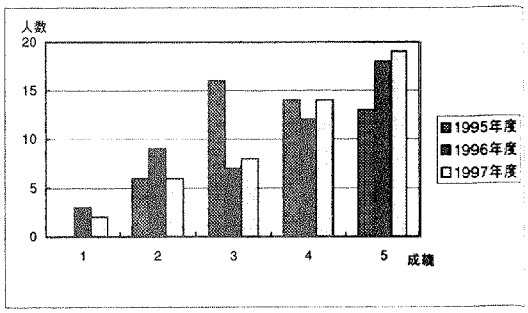


図 2 プログラム解析の成績分布
Fig. 2 Histogram for score of program analysis exercise.

表 3 レポートの平均点と標準偏差
Table 3 Statistics of report.

年度	並列プログラミング演習 人数	平均点	標準偏差	情報工学実験第 3（参考） 人数	平均点	標準偏差
1995 年度	57	81.8	14.1	68	79.5	8.13
1996 年度	61	82.4	10.4	65	78.1	8.18
1997 年度	49	76.3	12.7	67	76.8	8.24

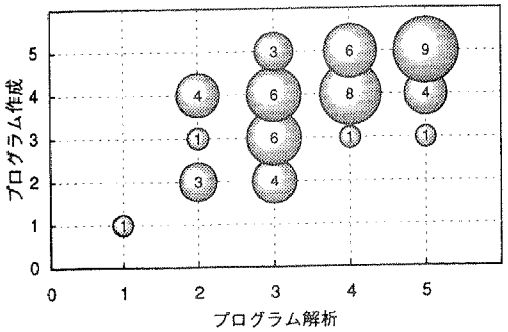
年度が最も高く，図 1 の 1995 年度のレポートの得点分布で，90 点以上の人数が一番多くなっている．1995 年度は並列処理に興味を持った学生が多く，その結果 90 点以上の分布が高くなったと考えられる．

次に，プログラム解析・可視化ツールの有効性について調べた．先に述べたように，1996 年度，1997 年度には実行時間コンテストによる体得的演習の効果的実施のために，プログラム解析・可視化ツールを演習した．このとき，プログラム解析・可視化ツールを用いて各自作成したプログラムを解析し，その解析結果に基づいてプログラムを改良する問題を課した．同様に，1995 年度にはプログラムを手作業で解析，改良する問題を課した．この問題を以下プログラム解析問題と呼ぶ．このプログラム解析問題の得点分布を図 **2** に示す．ここでは，5 段階で評価した得点を示した．評価の基準は以下のとおりである．ツールの基本的な使い方を理解し，プログラムの解析しているものは 3 とした．プログラムの解析結果に対し十分に考察を加えているものを 4 とした．また，解析結果に対して優れた考察を加えているものを 5 とした．逆に，ほとんど考察が加えられていないものを 2 とし，ツールの使い方が不十分であり，プログラムの解析がされていないものを 1 とした☆．図を見ると，得点の分布の傾向が 1995 年度と 1996 年度，1997 年度で違いがあることが分かる．5.1 節でも述べたように，1995 年度の演習ではプログラム解析・可視化ツールを使用しなかったが，その結果が得点の分布傾向に現れている．すなわち，1996 年度，1997 年度はプログラム解析・可視化ツールを用いることで効果的なプログラムの解析が可能となり，プログラム解析問題の得点が高くなったと考えられる．本結果により，並列プログラミング演習においては，プログラム解析・可視化ツールが必要であり，また有効であることが確認された．

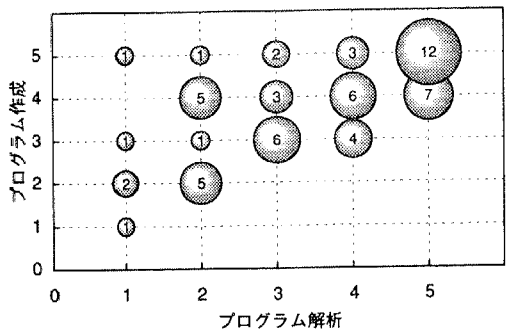
さらに，プログラム解析能力とプログラム作成能力の関係について調べた．我々は，プログラム解析能力とプログラム作成能力には強い相関があると考えた．これは，作成したプログラムを解析し，実行状態を調査し，それに基づいてプログラムに改良を加えることで，質の高いプログラムを作成できるからである．特に，並列プログラミング演習においては，作成したプログラムの実行状態を分析し，改良を加え，性能を向上させる過程が重要であり，プログラム解析能力とプログラム作成能力に強い関係が生じると考えられる．また，プログラム解析問題の得点が高い学生は，それだけ熱心に演習を行っており，その結果プログラム作成問題☆☆の得点が高いとも考えられる．横軸をプログラム解析問題の得点，縦軸をプログラム作成問題の得点とし，相関関係を表したものが図 **3** である．グラフ中の円の大きさが人数を表している．結果を見ると，データがほぼ左上三角の領域に集中していることが分かる．これは，プログラム解析問題の得点が高い学生はプログラム作成問題の得点も高いことを表している．この結果より，プログラムの解析能力の高い学生はプログラムの作成能力も高いことが実証できた．

☆ 1995 年度には，解析の方法，改良の方針などを 5 段階で評価した．

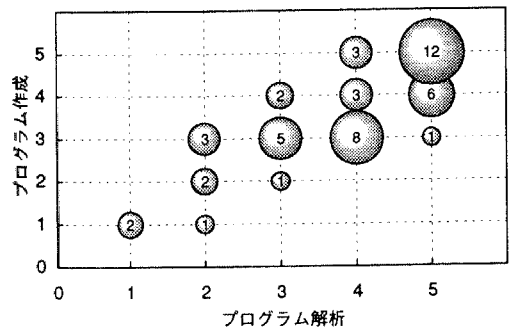
☆☆ プログラム作成問題は，与えられた問題に対するプログラムを実装する問題である．評価の基準はプログラム解析問題と同様に，平均的なプログラムを 3 として，設計に対して十分な考察を加えているものを 4，優れた考察を加えているものを 5 とした．逆に，ほとんど考察のないものを 2，仕様どおりのプログラムができていないものを 1 とした．



(a) 1995 年度



(b) 1996 年度



(c) 1997 年度

図 3 プログラムの解析能力と作成能力の関係
Fig. 3 Interrelationship between program analytic ability and program generative ability.

逆に，このグラフからはプログラムの作成問題の得点が高くても，必ずしもプログラム解析問題の得点が高くないことが分かる．解析ツールの使用が不十分であり，プログラム解析問題においてあまり考察が加えられていないレポートがそれにあたる．原因としては，時間不足や，プログラム改良の動機の低下など様々なものがあるが，これらは応用的なプログラミング能力の育成，性能を意識したプログラミング過程の習得など，本演習の目標が達成されなかった例である．これらに対処するための，演習内容や演習の進め方などの再検討は今度の課題である．

表 4 プログラム解析問題の得点
Table 4 Score for exercise of program analysis.

年度	平均点
1995 年度	3.58
1996 年度	3.51
1997 年度	3.86

最後に，体得的演習を効果的に実施するための工夫，すなわちプログラムの実行時間コンテストの有効性について調べた．ここでは，プログラム解析問題の得点により有効性を判断する．実行時間コンテストの実施により，学生がプログラムを改良するため，プログラム解析・可視化ツールを頻繁に使用するので，それにより，プログラム解析問題の得点が高くなると考えたためである．**表 4** にプログラム解析問題の平均点を示す．表より分かるように，1995 年度，1996 年度の平均点は 3.5 点であるのに対し，1997 年度は 3.9 点と平均点が上がっている．これは，実行時間コンテストによりプログラム解析・改良の能力が向上し，その結果平均点が上がったと考えられる．これにより，実行時間コンテストの有効性が確認された．

5.3 演習の感想からの解析結果

レポートでの演習の感想についての項目を分析する．レポートの最後に並列プログラミング演習の感想を書かせ，それを分析した☆．分析した結果を**表 5** に示す．

まず，並列プログラミング演習の有効性であるが，これは多くの学生が有効であると答えている．初めて並列計算機を使用して良い経験となったという記述が多かった．学部学生に対し，並列プログラミング演習を実施するのは有効であると考えられる．

次に，演習の困難さであるが，約 30% の学生が難しいと感じている．特に，1995 年度の演習において難しいと感じた学生が多かった．これは，プログラム解析・可視化ツールを導入しなかったことが原因であると考えられる．ツールを用いずにプログラムを改良しなければならず，プログラムをどのように解析・改良すればよいかが分からず，これにより困難さを感じたものと考えられる．しかし，難しいと答えた多くの学生は，難しいが有効であると答えており，演習のレベルとしては適切であった．

最後に，演習遂行時に問題となったことを問い，その回答をまとめたのが**表 6** である．「ワークステーション上で動作する並列計算機シミュレータの挙動が実際

☆ アンケート形式ではなく，自由記述形式であるので，すべてのレポートに次に述べる項目が記述されているわけではない．

表 5 演習の感想
Table 5 Impression of parallel programming exercise.

年度	人数	演習は有効	難しい	難しいが有効	難しくない
1995 年度	57	37 (64.9%)	30 (52.6%)	19 (33.3%)	2 (3.5%)
1996 年度	61	52 (85.2%)	10 (16.4%)	8 (13.1%)	8 (13.1%)
1997 年度	49	33 (67.3%)	15 (30.1%)	13 (26.5%)	3 (6.1%)
合計	167	122 (73.1%)	55 (32.9%)	40 (24.0%)	13 (7.8%)

表 6 演習の不満
Table 6 Dissatisfaction for exercise.

年度	人数	シミュレータ	AP1000 確保
1995 年度	57	9 (15.8%)	9 (15.8%)
1996 年度	61	23 (37.7%)	29 (47.5%)
1997 年度	49	19 (38.8%)	9 (18.4%)
合計	167	51 (30.5%)	47 (28.1%)

(注) シミュレータ ・・・並列計算機シミュレータと並列計算機の挙動が不一致
AP1000 確保 ・・・AP1000 を占有状態で使用しなければならない

の並列計算機と異なる」という意見と「並列計算機が占有状態で使用されるので，なかなか使用できない」という意見が多く見られた．AP1000 の場合，16 台のセルを分割して独立に運用する機能がなく，あるユーザが AP1000 を使用していると他ユーザは使用することができない．そのため，演習時間中に並列計算機を使用することができず，演習の遂行に困難を感じたようである．学部学生に対する演習のように，多人数で行う演習の場合，並列計算機にプロセッサの分割運用機能は必須である．

また，並列計算機シミュレータの挙動に対しての不満が多かった．多くは，プログラムを並列計算機シミュレータで実行するときと，並列計算機で実行するときでは，異なるコンパイラ・コマンドを使用しなければならず，面倒であるという意見であった．中には，あるライブラリ関数の挙動が並列計算機シミュレータと並列計算機では異なるという意見もあった．特に，上で述べたような理由により AP1000 が演習中に使用できず，並列計算機シミュレータを使用しなければならない状況が多く発生したため，並列計算機シミュレータと並列計算機の挙動の不一致がより気になったものと思われる．演習のように，試行錯誤によりプログラムを修正・改良していく場合，並列計算機シミュレータによる予備実行，少数のプロセッサによる試験実行，全プロセッサを使用した実行というステップを踏む必要がある．したがって，学生に対する教育目的の並列プログラミング演習の場合，精度の高い並列計算機シミュレータが必要であり，それが並列計算機との間でシームレスに使用可能であることが望ましい．

5.4 その他の解析結果

演習期間中，演習に関する質問は電子メールで受け付けた．その質問から，学生のプログラミング能力の低さが感じられた．特に，応用的なプログラムを作成する能力が備わっていないと感じられた．メモリ管理やポインタ変数の操作などに苦労する学生が多かった．したがって，並列プログラミング演習のようなシステムに関わる演習を効率良く行うためには，より高いプログラミング能力が求められる．そのため，より多くのプログラミング演習を行い，学生のプログラム能力向上に努めなければならないと感じた．

6. おわりに

本稿では，本学科 3 年生に対し，1995 年度から 1997 年度の 3 年間にわたって実施した並列プログラミング演習について述べた．本演習は，学生のプログラミング能力向上のために開講された「プログラミング演習」の一部として行われ，並列計算機 AP1000 上での並列プログラミングの基礎を習得させる目的を持っていた．本演習は，卒業研究以降の研究に対処可能な，応用的なプログラミング能力の育成，および機能だけでなく性能も意識したプログラミング過程の習得を目標として行われた．

本演習実施の結果，以下のようなことが分かった．まず，学部学生に対する並列プログラミング演習の適用の可否であるが，問題のレベルとしては十分対処可能であった．また，並列処理に関する解説，プログラミング環境に対する学習実験，並列プログラミングに対する体得的演習をバランス良く行うことにより，効果的な演習が実施できた．特に，体得的演習を効果的に実施するためには，学生のやる気を持続させることが必要であるが，そのためにプログラムの実行時間コンテストを開催した．この実行時間コンテストにより，体得的演習の効果が高まったことが分かった．実行時間短縮は並列処理の大きな目的の 1 つであるので，並列プログラミング演習において実行時間コンテストを開催するのは特に効果的であった．演習の問題点としては，学生のプログラミング能力の低さがあげられる．特に，応用的なプログラムを作成する能力が低く，メ

モリ管理やポインタ変数使用などの点でつまずく学生が多かった．まとめると，レポートの解析からは本演習の目標を達成することができたと考えられる．もちろん，5.2 節のプログラム解析問題とプログラム作成問題の得点評価の議論のように，必ずしもそうでない例もあるが，演習の感想などを総合すると，目標は十分達成できたと結論づけてもよい．

また，並列プログラミング演習において必要な計算機ファシリティとしては，以下のものがあげられる．

- パーティション分割機能があり，複数人で使用可能な機能を有する並列計算機．
- ワークステーション上で動作する並列計算機シミュレータ．ただし，そのシミュレータが高精度であることが重要である．
- プログラムの実行状態を可視化可能なプログラム解析ツール．

これらは現在利用可能な一般的な並列計算機システムの提供するプログラミング環境であれば備わっているものである．したがって，本稿で述べた並列プログラミング演習は AP1000 に限らず，多くの並列計算機にも適用可能で，同様の演習効果が期待できる．

学部学生に対して並列プログラミング演習を実施することは，演習のレベルとしてそれほど問題なく，新しい計算機を経験させるという点で非常に有意義である．しかし，上であげたような計算機ファシリティを用意して，体得的演習が効果的に実施できるような演習を設定することが重要である．

謝辞 日頃よりご指導，ご協力いただいている本学大学院工学研究科・稲垣康善教授，鳥脇純一郎教授をはじめ，本情報工学教室の諸先生，特に演習時に計算機システムに関していろいろお世話になった齊藤豊文助教授，大下弘技官，ならびに TA の牛尼剛聡君，大沼宏行君に感謝いたします．さらに，有益な助言をいただいた査読者の方々に感謝いたします．

参 考 文 献

1) Koffman, E., Miller, P. and Wardle, C.: Recommanded Curriculum for CS1, 1984 – A Report of the ACM Curriculum Committee Task Force for CS1, *Comm. ACM*, Vol.27, No.10, pp.998–1001 (1984).
2) Koffman, E., Stemple, D. and Wardle, C.: Recommanded Curriculum for CS2, 1984 – A Report of the ACM Curriculum Committee Task Force for CS2, *Comm. ACM*, Vol.28, No.8, pp.815–818 (1985).
3) Tucker, A., Barnes, B., Aiken, R., Barker, K., Bruce, K., Cain, J., Conry, S., Engel, G., Epstein, R., Lidtke, D., Mulder, M., Rogers, J., Spafford, E. and Turner, A.: *Computing Curricula 1991 – Report of the ACM/IEEE-CS Joint Curriculum Task Force*, ACM PRESS (1991).
4) Silberschatz, A. and Galvin, P.: *Operating System Concepts*, Addison-Wesley (1997).
5) Hennessy, J., Patterson, D., 成田光彰（訳）: コンピュータの構成と設計—ハードウェアとソフトウェアのインタフェース [上][下]，日経 BP 社 (1996).
6) 都倉信樹：情報処理教育における実験・演習，情報処理，Vol.32, No.10, pp.1101–1108 (1991).
7) Brawer, S., 大森健児（訳）: 並列プログラミングの基礎，丸善 (1990).

(平成 10 年 9 月 1 日受付)
(平成 11 年 2 月 8 日採録)

朝倉 宏一（正会員）

1970 年生．1992 年名古屋大学工学部情報工学科卒業．1994 年同大学院工学研究科情報工学専攻博士・前期課程修了．1995 年同大学院工学研究科情報工学専攻博士・後期課程中途退学．同年同大学院工学研究科情報工学専攻助手．ワークステーション・クラスタ環境，並列計算機環境におけるシステム・ソフトウェアに興味を持つ．

渡邊 豊英（正会員）

1948 年生．1972 年京都大学理学部卒業．1974 年同大学院工学研究科数理工学専攻修士課程修了．1975 年同博士課程中途退学．同年京都大学大型計算機センター助手．1987 年名古屋大学工学部情報工学科助教授．現在，同大学大学院工学研究科情報工学専攻教授．京都大学工学博士．統合化環境，分散協調環境，データベース環境，データベースの高度インタフェース，知的 CAI，文書理解，地図理解に興味を持つ．電子情報通信学会，日本ソフトウェア科学会，人工知能学会，ACM，IEEE Computer Society，AAAI 各会員．