プログラミング－言語・基礎・実践－ 13－7
（1993. 8. 20）

ワークステーション複合体による並列処理システム

— 中・小粒度オブジェクト指向並列処理の実現 —

瀧 和男* 小倉 毅** 小西 健三**

*神戸大学工学部情報知能工学科
**神戸大学大学院工学研究科システム工学専攻

高性能ワークステーション複数台をLANとは別系統の高速メッセージ通信路で接続し，応答性のよいメッセージ通信を行なう並列処理システムを実現した．言語として，粒度が小さめの多数のオブジェクトを効率良く扱える並列オブジェクト指向言語と処理系を試作中である．非データ並列の大規模計算を対象と考えており，大規模並列計算機へも適合し易い応用プログラムが開発できる開発・実行環境を目指している．言語およびシステムの評価に使える本格的な並列応用として，論理シミュレータとLSIレイアウトプログラムを開発中である．プロトタイプが一部稼働を始めており，初期の性能測定結果についても報告する．

Parallel Processing System on the Multi-Workstations

Kazuo Taki* Tsuyoshi Ogura** Kenzo Konishi**

*Department of Computer and Systems Engneering,
Faculty of Engineering, Kobe University
**Graduate School of System Engineering, Kobe University

A parallel processing system has been developed in which high speed workstations are connected with low-latency communication path. A concurrent object-oriented language and a processing system have been designed and implemented on the multi-workstations, which can handle a lot of non-large objects efficiently. A target application domain is non data-parallel processing. The system focuses a development capability of such applications that can scale up for large parallel computers. A logic simulator and a LSI layout system are under development as practical applications. Early measurements results for prototypes will be also reported.

1 はじめに

数値計算を中心としたデータ並列計算に分類される並列処理は，並列化コンパイラの技術を軸として近年急速な進歩を遂げている．それとは対照的に非データ並列計算，すなわち動的な計算や，均質でないデータに対する均質でない計算の並列処理に関しては，まだ十分に研究が進んでいないのが現状である．けれどもこの領域は，大規模並列計算機の応用を拡大する上ではなくてはならないと考えられる．

本研究では，動的で均質さの低い大規模計算を対象問題領域ととらえて，それに適した並列言語系を提案するとともに，身近で性能の良い並列実行環境としてそれを実現し，合わせて大規模並列計算機への移行性も考慮することを目指している．

本研究の動機は次の3点である．

KL1の長所を生かした並列オブジェクト指向言語 第一は，第五世代コンピュータプロジェクトで研究されたKL1言語によるプログラム開発が，非データ並列問題の並列処理に関してまとまった成果をあげていることであり[1]，それに関わるKL1言語の長所・短所も明らかになりつつある．著者らは，KL1言語でよく利用される機能がある程度限られていること，KL1で記述されたプログラムの多くが並列オブジェクトモデルに基づくことに着目し，KL1の長所を取り込んだ並列オブジェクト指向言語が，当該領域の並列処理に有効であるとの仮説を立てた．

KL1の弱点の補強とC言語の取り込み 第二点として，これまでのKL1実装では，通信・同期を実現するためのオーバヘッドが，単純なプログラム(本来通信・同期を必要としないようなプログラムなど)の場合ほど大きく目立つ傾向があり，その改善が望まれていたこと，また，Cなどの普及した言語に習熟したプログラマにとって移行しやすい本格的な並列言語系も望まれていたことがあげられる．このことから，通信・同期をメッセージ送受部分に限定し、C言語をメソッド記述に使うことのできる並列オブジェクト指向言語系を検討することにした。

大規模並列計算機への移行性 第三点は，大規模並列計算機が非常に高価であり誰でも利用できるわけではなく，さらに，利用しやすい小規模並列計算機で開発した並列プログラムは，かならずしも大規模並列計算機に移行し易いものではないことがあげられる．このことは，大規模並列計算機で動く応用を拡大するうえで障害となる．このことから，利用し易い並列処理環境としてワークステーションを複数接続したものを考え，その上に大規模並列計算機へ移行し易い並列言語系を実装することにした．移行の容易さを確保するため，粒度が小さめの多数のオブジェクトを効率良く扱える実装を指向した．

以上のことから本研究の目的をまとめると次のようになる．

1. KL1言語の長所を取り込み，メソッド記述にはC言語が使える並列オブジェクト指向言語「mosaic」の提案
2. 提案した言語の非データ並列処理領域における有効性の検証
3. 提案した言語に関して，粒度が小さめの多数のオブジェクトを効率良く扱える言語処理系の実現(大規模並列計算機への移行性を考慮した実装)
4. 提案した言語を効率良く実行する身近な実行環境としてのマルチワークステーションシステム「並列くん」の提案と性能評価

本稿では，このような研究・試作の概要および速報として次の事項を報告する．2節では，提案する並列オブジェクト指向言語の設計思想と言語仕様の概要を述べる．3節では，ワークステーションを接続したハードウェアの概要とその上での言語システムの実現方針を述べる．4節では，試験実装における言語実装方式を紹介する．5節では試験実装の性能について，6節では応用プログラムの開発について速報としての報告を行なう．

2 並列オブジェクト指向言語「mosaic」の設計思想

mosaic初版の設計では，KL1言語[2]で並列オブジェクトモデルに基づくプログラムを記述する場合に比べ，同等の記述力となるような並列オブジェクト指向言語を目指している．合わせて，言語の実行モデルと記法の両方とも分り易く簡単なものであること，実用プログラムの記述に十分な機能を備えることを目指す．

基本となる並列オブジェクトのモデルはActor[3]に近い単純なもので，メッセージ通信は[3]の分類によれば，非同期メッセージ送信(送信側はメッセージを送ると，受信側の受理を待たずに次の動作を行なう)のみを許す．クラスを継承する機能は初版には入れない．

KL1の長所を取り入れ以下の機能を持たせる．

- 未生成オブジェクトへのメッセージ送信：オブジェクトの生成完了を待たずにメッセージを送り付けられる機能．オブジェクトを表すことになっている変数に対しては，オブジェクトが代入されていなくても，先にメッセージの送りつけを許す機能である．逐次性を減少させる効果が期待される．

- 優先度付きメッセージ：メッセージに対して処理優先度を指定できる機能．優先度レベル数は数千以上を用意する．優先度指定にはKL1のpragmaに類する記法を用いる．スケジューリング制御に利用する．特に見込み計算を許して並列性を高めるアルゴリズムの実現に効果が期待される．受信メッセージの優先度はメソッドのプログラム中で知ることができる．

- メッセージの順序性の保存：あるオブジェクトから別の1個のオブジェクトへ送出された複数のメッセージについて，それらが同一優先度であれば送出と同じ順序で受信側オブジェクトに到着する(実行される)．

- オブジェクト生成時のプロセッサ指定：KL1の負荷分散pragmaに類する記法により，どのプロセッサにオブジェクト生成するかを指定可能とする．負荷分散の調整・変更が容易であり，プログラム完成後の性能調整に強い．負荷分散はプログラムコントロールを基本とし，自動化には負荷分散ライブラリを提供していく．

C言語のプログラマが移行し易い言語となるよう以下の方針をとる．

- メソッド記述にはC言語がそのまま使えるようにする．したがってメソッド内は逐次実行である．メッセージ送出やオブジェクト生成機能等はライブラリ関数の形で与える．

- C言語にもともとあるデータ型とオブジェクト指向システムとして新たに管理すべきデータ型の区別が問題となり検討中である．暫定仕様として，メッセージの引数にはCのアトミックなデータとオブジェクトのみを許している．

C言語を活用することで言語処理系の実現を容易にすることも狙っている．言語実装にあたっては，プロセッサ間メッセージ通信での応答時間短縮に留意する．このことは，粒度の小さめのオブジェクトを非常に多く扱うプログラムが効率良く並列処理できるために必要不可欠で，大規模並列計算機への移行性を確保する上で重要である．

3 システムの概要

3.1 システムの実現方針

ハードウェアおよび言語処理系の基本的な実現方針をOSとの関係も含めて述べる．

対象マシンと処理系実装方針 言語処理系は，小規模な並列計算機へも分散メモリ型の大規模並列計算機へも容易に実装できる方式とする．このために，言語処理系から見たプロセッサ間通信はメッセージ通信インタフェースとする．OSは一般的なUNIXを仮定する．

初版のハードウェア 初期の対象ハードウェアとしては，身近に確保できるものとして，高性能ワークステーションを接続したものを考える．

プロセッサ間通信応答時間の短縮 異なるプロセッサ上のプロセス間通信応答性を確保する方法として，徹底してOSを呼ばない通信方式をとる．このために，非特権プロセス(処理系を実現するプロセス)から直接操作できる通信機構を用意する．受信時には割り込みの代わりにポーリングを用いる．

オブジェクトの実現 一つのプロセス(処理系を実現するプロセス)中に複数のオブジェクトを実現する．オブジェクトのスケジューラも同時に実現する．別プロセッサからのメッセージ到着確認のためのポーリングは，オブジェクトのメソッド実行からスケジューラへ処理が戻るたびに行なう．

オブジェクト間通信 ユーザプログラムのレベルでは，オブジェクト間のメッセージ通信は，プロセッサ内，プロセッサ間を問わず同じインタフェースとする．すなわち，プロセッサ間通信は暗黙のうちに行なわれる．

接続機構　ワークステーション間接続機構は，LANとは別に，処理系プロセスから直接操作できてかつ実現が容易な方法をとる．初版では，ワークステーションの外にVMEバスと通信バッファ用の共有メモリを用意し，バスアダプタによりワークステーションのI/OバスとVMEバスを接続する．

処理系の立ち上げ　言語処理系(実行系)の立ち上げは，始めにLAN経由で各プロセッサに処理系プロセスを立ち上げ，初期化が終了するとVMEバス経由の通信が始まり，各処理系プロセスは一体となって一つの並列オブジェクト指向実行系を構成する．

3.2　マルチワークステーション「並列くん」のハードウェア構成

「並列くん」のハードウェア構成について述べる．

ワークステーション(SPARCstation-2互換機)8台を2系統で接続している．一方はイーサネットであり，もう一方はVMEバスを用いた接続機構である．ワークステーションの外部にVMEバスのシャーシを用意し，各WSのS-busとのあいだを市販のバス変換アダプタ(BiT3 model 466)で接続している．このアダプタボードは，VMEバスのアドレス空間をS-bus上にマップする．さらに言語処理系プロセスは，初期化時にS-busアドレス空間の必要部分をベクターにマップして参照する．

VMEバス側のアダプタボード(8枚)には，それぞれdaughterボードの形で，通信バッファに使用する共有メモリ(128KB)が実装されている．これらはデュアルポートメモリになっており，VMEのバス負荷軽減に役立っている．WSからVMEバス側の共有メモリへのケーブルを介したアクセス時間は2μsec(32ビット転送)である．

通信バッファ上で領域確保するときには排他制御が必要となるが，このためのセマフォを効率化するためにハードウェア(ボードを1枚)を用意している．

他のマシンにメッセージを送出する際には，そのマシンの通信バッファに領域を確保し書き込みに行く．各マシン上の処理系プロセスは，自身の通信バッファをポーリングしており，メッセージが到着していればそれを読み込む．

4　言語実装の概要

現在，シングルユーザ用実行系の試験実装が終了している．以下では最終的な実行系のイメージと，

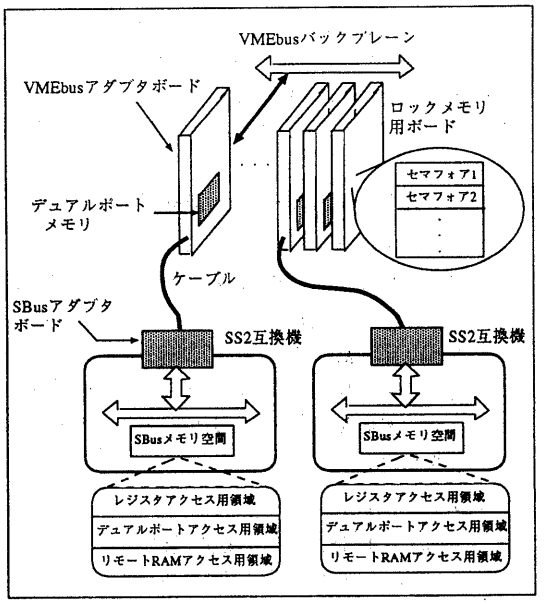

図1:「並列くん」のハードウェア構成

現段階の試験実装での動作概要を述べる．

4.1　実行系

言語実行系のモジュール構成は図2のようになる．これらのモジュール群は，一つのUNIXプロセスとして実現する．以下，主なものについて説明する．

- スケジューラ: プロセッサ内のオブジェクトおよびメッセージキューを管理するモジュール．キューからメッセージを取りだし，宛先オブジェクトのメソッドを起動する．
- メッセージキュー：スケジューラが処理すべきメッセージの列．メッセージのプライオリティ毎に存在するFIFOキュー．
- メッセージバッファ: 他のプロセッサからのメッセージの受け口．実体は各マシンのデュアルポートメモリ．
- 通信ライブラリ: プロセッサ間通信用モジュール群
- アドレス変換表: プロセッサの外にオブジェクトを見せるためのアドレス変換表．ローカル

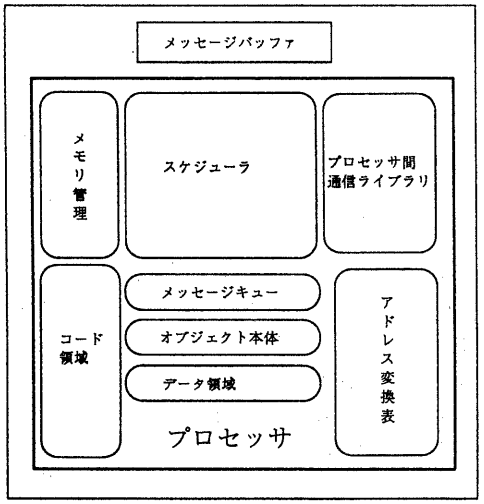

図 2: 言語実行系の構成要素 (1 プロセッサ内)

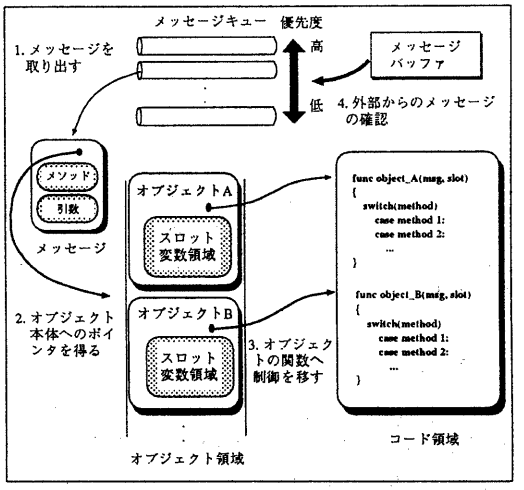

図 3: スケジューラの動作

GC でオブジェクトのアドレスが変化することを許す目的や，外部参照数の管理，ローカル GC のマーキングルートとして用いる．現在は GC を実装していないので未使用．

4.2 試験実装の概要

オブジェクトの GC, 未生成オブジェクトへのメッセージ送出機能を除いた基本的な機能の試験実装が完了している．以下にその動作の概要を説明する．

4.2.1 プログラムと翻訳系

現在, 表層言語は設計中であり，スケジューラ, プロセッサ間通信用モジュール等がライブラリの形で提供されている．ユーザは各オブジェクトを C 言語の関数としてプログラムし, これらのライブラリをリンクする形でシステムを使用できる．ライブラリの中には main 関数も含まれており, そこからシステムの起動と初期化が行なわれる．最初に実行されるオブジェクトが root_object であり，ユーザはこれをを必ず記述する必要がある．

翻訳系は, 表層言語のソースプログラムを C 言語に変換するプリプロセッサとして実現する．

4.2.2 システムの起動

ログインマシンのユーザプログラムから p4[4]（RPC コール）を用いて各プロセッサの処理系プロセスを起動している．

各プロセッサのプロセスは初期化の際，system_object と呼ばれる特別のオブジェクトを生成する．これは処理系自体を抽象化したオブジェクトであり，将来リフレクション機能の実現に用いる．現在はオブジェクトの生成時とシステムの終了時に呼び出される．

4.2.3 スケジューラの動作

プロセッサ内のスケジューラは, 以下の動作を繰り返しキューの中のメッセージを処理する（図 3）．

1. 優先度最大の空ではないキューからメッセージを一つ取り出し, 宛先オブジェクトへのポインタ（オブジェクト本体用領域内のオフセット）を得る．
2. オブジェクト本体からそのオブジェクト自身の動作を記述している関数へのポインタを得, その関数に制御を移す．その際メッセージ本体へのポインタ, オブジェクト本体のスロット変数領域へのポインタを引数に渡す．
3. 関数内ではメッセージから所望のメソッドを識別し, 対応する動作へ分岐する．
4. 関数から制御が戻されたスケジューラは通信バッファをポーリングし, 他のプロセッサから

のメッセージがあればキューに格納する．その後1へ戻る．

このような動作によって，同一優先度のメッセージについては必ず送出順に処理されることになる．オブジェクト自身にはローカルなメッセージキューは持たせておらず，スケジューラから見るとキューから取り出されオブジェクトに渡されたメッセージは直ちに処理される．未生成オブジェクトへのメッセージ送出機能は，プログラマからは見えない「メッセージバッファオブジェクト」を用いて実現する．

5 試験実装の性能

試験実装の性能を測定した．プログラムの最適化はこれからである．全ての場合の測定条件として，メッセージ長は3word (12byte) である．宛先オブジェクトのID，メッセージ名，メッセージ長を含む．

スケジューラの処理時間 4.2.3節のスケジューラのサイクルのうち，メソッドの実行部分以外の処理時間を計測した．オブジェクトにreturn命令だけの何もしないメソッドを持たせ，これを起動するメッセージの処理を繰り返して，所要時間を測定した．スケジューラ1サイクルあたりの処理時間は11.8μsecであった．一つのメッセージの処理が終了し，他のプロセッサからのメッセージ到着確認のため通信バッファにアクセスする部分に8μsec（トータル4回のVMEバスアダプタボードへのアクセス）かかるため，ほとんどがこの部分の処理時間となっている．

メッセージ通信 プロセッサ内，プロセッサ渡り2つの場合について，メッセージ通信にかかる時間を計測した．2つのオブジェクトの間で，メッセージを受信するとすぐに新しいメッセージを組み立て相手に送出する処理を繰り返し，時間を計測した．片道の通信時間は，プロセッサ内通信で15.2μsec，プロセッサ渡りで92μsecであった(92μsecのうち52μsecは通信バッファのアクセス時間と見積もられているが，残りが大き過ぎるため解析中である)．

プロセッサ内時間は，前述のスケジューラの処理時間に，メッセージの組み立てとキューへの登録時間を加えたものに相当する．

プロセッサ間通信に関しては，参考のためp4を用いてLAN経由で12byte長メッセージを転送する時間を測定した．平均転送時間は794μsecで，本システムのメッセージ通信時間の約9倍であった．

オブジェクトの生成 プロセッサ内，プロセッサ外にオブジェクトを生成するのに要する時間を計測した．プロセッサ内，外の場合ともオブジェクトを生成してほしいプロセッサのsystem_objectにオブジェクト生成を依頼して，応答メッセージを受け取るまでの時間である．結果はプロセッサ内生成で36μsec，他のプロセッサへの生成で240μsecであった．現仕様ではsystem_objectからの応答メッセージ長が16wordであるため，転送時間が長めになっている．

通信性能について VMEバス経由のデータ転送能力は，最大スループットで2M byte/sec(16Mbps)，送信，受信が直列化される場合(レスポンス性能)では最大1M byte/sec(8Mbps)である．これはハードウェアの性能としてはイーサネットと同程度である．

したがって本システムの通信応答性がLAN経由の場合に比べ約1桁高いのは，ほとんどがOSオーバヘッドをなくしたことに依っているといえる．なお，通信バッファアクセス以外の部分のCPU時間はまだ短縮の余地がある．

6 応用

システムの評価と改良に役立てるため，初期の応用プログラムとしてつぎのものを試作中である．LSI CAD関係のプログラムは，実用プログラムのプロトタイピングを指向している．

6.1 最短経路問題

点の数1万から100万のグラフ上の最短経路を分散アルゴリズムで解くもので，点の数と同数のオブジェクトが生成されそれらが非同期的にメッセージ交換して経路探索を行なう[5]．粒度が小さめのオブジェクトを多数扱う例として性能評価に用いる．見込み計算の制御にメッセージの優先度を利用する．

6.2 タイムワープ法による並列論理シミュレータ

離散系イベントシミュレーションの一手法であるタイムワープ法を論理シミュレーションに適用したものである．各ゲートがオブジェクトとしてモデル化され，信号値変化がイベント(メッセージ)として伝達される．ゲート毎に分散時刻管理を行なうため，並列性を引出し易い特徴を持つ．KL1言語用の専用マシンであるPIM/マルチPSI上での試作・評価が報告されている[6][5]が，ワークステーションや汎用計算機での実装の報告はまだない．ゲートをプロセッサに割り付けるときに，通信の局所性を高

める割り付け方式をとることが並列処理性能を高める上で重要である.

現在は基本部分のコーディングを終えてデバッグ中であり, プロセッサ1台当たりのシミュレーション性能として, 約 30K[event/sec] を得ている.

将来は, 動作記述レベルのシミュレーションを混在させたミックスレベル・シミュレーションへ発展させる予定である.

6.3 LSI のレイアウト設計

LSI のレイアウト設計問題は計算量の多いことで知られており, 並列処理の応用として好適であるが, ここではとくに, 設計精度を高め設計結果の LSI が高い性能を持つことを目指した「性能指向 CAD」を実現しようとしている [7]. レイアウト設計は配置設計と配線設計からなる.

6.3.1 温度並列 SA 法によるブロック配置

セルベース IC(マクロブロックおよびブロックの組合せで一つのチップを構成する IC) を対象としたブロック配置プログラムを試作中である.

ブロック配置は, 短い配線長と小さなチップ面積の実現を目指した組合せ最適化問題であるが, ここでは最近考案され最適化能力が高いと考えられている「温度並列シミュレーテッド・アニーリング (SA) 法」をブロック配置にはじめて適用した [7].

温度と呼ばれるパラメータの時間方向のスケジューリングを自動化する方式である. プロセッサ毎に温度一定の逐次 SA を独立に走らせるが, それぞれには高温から低温に至る異なる温度を割り振っておく. 隣接温度間では周期的に解の (確率的) 交換を行なう. 高温で良い解の種が生成されると, それは解の改善を経ながら解交換によって次第に低温部に移動してゆき, 最終的には最低温度の逐次 SA 処理の中に (準) 最適解が出現する.

必要な温度の数として, 小規模のテスト用問題で 16 温度, 実用問題で 32 から 128 温度程度を考えている. 通信と同期は, 温度間の解交換の時点で発生する. この頻度が高い場合, 通信応答性の高いシステム実装が有効に働く.

図 4は, MCNC ベンチマークの例題の配置結果である. [7]

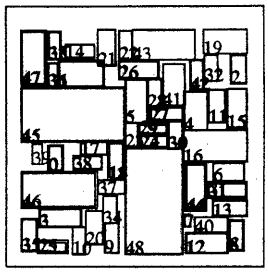

図 4: MCNC ami49 の配置結果

6.3.2 概略配線と詳細配線

概略配線には, オブジェクト指向の並列配線 (予測線分探索法に基づく)[8][5] を改良し, 配線チャネルをオブジェクトとしてオブジェクト間でメッセージ交換しながら並列に配線割り当て処理を行なう方式を検討している.

詳細配線には, 複数のチャネルにおけるチャネル配線の並列実行と, チャネル内での並列配線を検討している. 詳細配線を並列に進めるため, オブジェクト指向機能を活用した協調型問題解決方式を必要に応じて検討の予定である.

7 おわりに

本稿では, 非データ並列計算を対象と考え, 小規模から大規模並列計算機までの移行性に優れた並列オブジェクト指向言語系の提案, その言語の身近な実行環境としてのマルチワークステーションシステムと言語処理系の試作報告, その上での並列応用プログラムの開発概要について述べた.

粒度が小さめの多数のオブジェクトを扱えるようにするため, 通信応答時間の短縮に留意した. 異なるワークステーション上のオブジェクト間通信時間は約 90μsec であり, まだ改善の余地がある.

今後は, GC 等未実装機能の追加と言語処理系の評価・改良, 表層言語の設計完結, LSI CAD 等の応用プログラムの稼働, マルチワークステーションの性能評価, 大規模並列計算機へのソフトウェアの移植, ソフトウェア公開などを計画している.

謝辞

本学 4 年の屋鋪正史君, 笹木歩君には試験実装で協力頂いた. また, 実験設備の面で御協力頂いた (株) マイクロ ヘリオス殿に感謝する.

参考文献

[1] Nitta, K., Taki, K. and Ichiyoshi, N.: Experimental Parallel Inference Software, in *Proc. FGCS'92*, pp. 166–190 (1992).

[2] Ueda, K. and Chikayama, T.: Design of the kernel language for the parallel inference machine, *The Computer Journal*, Vol. 33, No. 6, pp. 494–500 (1990).

[3] 石川裕, 所真理雄：オブジェクト指向並行プログラミング言語, 情報処理, Vol. 29, No. 4 (1988).

[4] Butler, R. and Lusk, E.: *User's Guide to the p4 Programming System*, Argonne national laboratory Mathematics and Computer Science Divition (1992).

[5] 瀧和男, 市吉伸行：マルチ PSI における並列処理とその評価 — 小粒度高並列オブジェクトモデルに基づくパラダイムについて—, 電子情報通信学会論文誌, Vol. J75-D-I, No. 8, pp. 723–739 (1992).

[6] 松本幸則, 瀧和男：バーチャルタイムによる並列論理シミュレーション, 情報処理学会論文誌, Vol. 33, No. 3, pp. 387–396 (1992).

[7] 瀧和男, 小西健三：マルチワークステーション上の並列 DA システム — 大規模並列マシンへの移行性を考慮した実行環境と DA システム —, 情報処理学会 DA シンポジウム'93 論文集 (1993).

[8] 伊達博, 大嶽能久, 瀧和男：並列オブジェクトモデルに基づく LSI 配線プログラム, 情報処理学会論文誌, pp. 378–386 (1992).

[9] 田浦健次朗, 松岡聡, 米澤明憲：マルチコンピュータ上の並列オブジェクト指向言語の高性能実装, 並列処理シンポジウム JSPP'92, pp. 163–170 (1992).

[10] Hirata, K., et al.: Parallel and Distributed Implementation of Concurrent Logic Programming Language KL1, in *Proc. FGCS'92*, pp. 436–459 (1992).

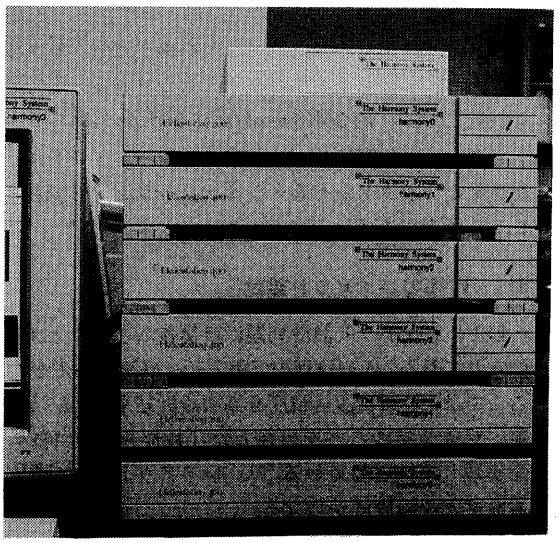

図 5: マルチワークステーション「並列くん」(本体)

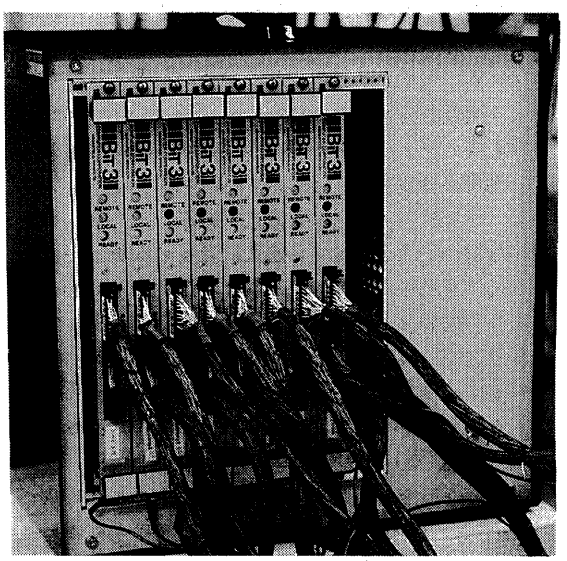

図 6: 「並列くん」の VME バス実装