システムソフトウェアと
オペレーティング・システム 61－8
（１９９３．８．１８）

超並列システムカーネル *SCore* の構想

堀 敦史, 石川 裕, 小中 裕喜, 前田 宗則, 友清 孝志
技術研究組合 新情報処理開発機構（RWC） つくば研究センタ
e-mail:{hori,ishikawa,konaka,m-maeda,tomokiyo}@trc.rwcp.or.jp

RWC プロジェクトでは現在 1,000 台規模の超並列計算機 RWC-1 の設計が進められている．本稿は RWC-1 上での並列プログラミングシステムを構築するために必要なオペレーティングシステムカーネル *SCore* の基本構想と，柔らかい情報処理を実現するために必要とされるリアルタイム処理および快適なプログラミング環境を提供するタイムシェアリング機能を実現するために必要なハードウェア特性について述べたものである．*SCore* は，マルチユーザ，マルチタスク，タイムシェアリング，リアルタイム，並列オブジェクト，オブジェクトストレージなどを実現するための基本機能を提供する．

Overview of Massively Parallel Operating System Kernel *SCore*

Atsushi HORI, Yutaka ISHIKAWA, Hiroki KONAKA,
Munenori MAEDA and Takashi TOMOKIYO

Real World Computing Partnership, Tsukuba Research Center

In RWC (Real World Computing) project, the design of a massively parallel computer RWC-1 scalable up to thousand processing elements is proceeding. This paper describes the basic proposal of operaing system kernel *SCore*, and the hardware facilities to develop realtime processing that realizes soft information processing and time-sharing function to provide adequate programming environment on the massively parallel machine. *SCore* provides basic functions to realize the environment of multi-user, multi-task, time-sharing, realtime, parallel object, object storage and so on.

1 はじめに

RWC プロジェクトでは, 新たな超並列計算機 RWC-1 を開発し, それをベースに OS, 並列オブジェクト指向言語およびさまざまな応用ソフトウェアを開発する予定である. RWC-1[14] は 1,000 台規模の本格的な実用を目指した超並列計算機プロトタイプである.

現在, 商用に開発された並列計算機の多くが UNIX あるいは OSF/1 をベースとした OS を載せている. 逐次計算機で培われたソフトウェア資産を継承するには, UNIX をベースとすることはある意味で妥当である. しかしながら, 超並列 OS の研究は始まったばかりであり, 単に逐次計算機の OS をそのまま移植／拡張していいものかどうかは, 必ずしも明確にはなっているとはいえない.

一方, 超並列アーキテクチャの設計においても, 超並列 OS を支援するための機能が必ずしも十分に実現されているとは言い切れない. これは超並列 OS のあるべき姿が未だ明確になっていないことも理由として考えられるであろう.

我々は逐次 OS を見直し, 新たなカーネルを設計しようとしている. 貴重な研究成果である逐次 OS を全面的に否定しようとするものではなく, 個々の技術が超並列に適用可能かどうかの再検討を加えながら, 超並列 OS の姿を捉えようとするものである.

本論文ではこうした視点に立ち, 最初に RWC プロジェクトにおける超並列システム研究の位置付けに触れ, 次に RWC-1 上の超並列オペレーティングシステムカーネルである *SCore* の概要について述べる. 論文中では, 超並列 OS においてタイムシェアリングやリアルタイムプロセスを実現するための要件を, ハードウェア／ソフトウェアの両面から検討する. 最後に, RWC-1 ハードウェアにおいて実現される OS に関連した機能についても触れることにする.

2 Real World Computing

RWC プロジェクトでは, 柔らかな論理に基づいた新しい情報処理方式の構築を目指している [15]. アプリケーションとしてはリアルタイム画像処理, リアルタイム音声認識, リアルタイムロボット制御, ニューラルネットワークシミュレータなどが考えられている. RWC つくば研究センタで開発しようとしている超並列計算機 RWC-1 は, RWC プロジェクトにおける研究の大きな柱のひとつであり, 柔らかい情報処理の基盤技術でもある.

2.1 RWC-1

RWC-1 は柔らかい情報処理アプリケーションの実行環境提供する. それと同時に, 汎用の超並列計算機としての可能性を追求するつもりであり, 大規模な数値シミュレーションもターゲットのひとつとして捉えられている.

RWC-1 は疎結合型の MIMD 型並列マシンで, 細粒度並列処理を指向したアーキテクチャである. 高性能なネットワークを持ち, 通信と演算処理が高度に融合されている. 本格的な OS を載せることを前提に設計されており, OS 機能実現のための支援機能も備えている [14, 11, 13, 19].

2.2 プログラミングシステム

図 1に現在想定されているプログラミングシステムの階層図を示す.

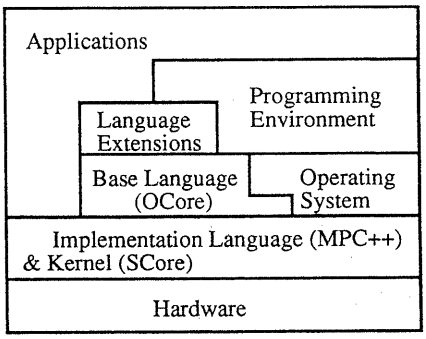


図 1: プログラミングシステム階層図 [16]

- *SCore*
  オペレーティングシステムカーネル.
- MPC++[17]
  C++ をベースとした並列実装言語. *SCore* は MPC++ で記述される.
- OCore[18]
  並列オブジェクト指向言語.

3 OS に対する要求

我々は RWC-1 の上に本格的なオペレーティングシステムカーネルを載せ, タイムシェアリングなど現在のワークステーションに近いプログラミング環境を実現する予定である. また, 柔らかい情報処理アプリケーションを想定し, リアルタイム処理にも対応可能なシステムを提供しようとしている.

超並列計算機が近い将来の汎用計算機のひとつの形態であるならば, 現在のワークステーションと同じ程度に快適なプログラミング環境が提供できなければならない. このため, マルチユーザ／マルチタスク環境も実現される必要がある. これは RWC-1 の利用可能性を高めることにつながる.

超並列計算機においてタイムシェアリングを実現するのは, 応用側からのニーズというだけでなく, かつてスーパーコンピュータが歩んだように「使いやすさ」を具体化した結果と考える.

リアルタイム処理に対し, 超並列計算機が提供する膨大な計算パワーを応用することは, その汎用性を広げることを意味する.

RWC-1 で実現される OS には, 想定されるアプリケーションの性格から, 1) **マルチタスク／リアルタイムの実現**, また, 並列プログラミング環境の研究という立場から, 2) **並列 OS 研究のためのプラットフォームの提供**, availability を高めるための 3) **マルチユーザ／タイムシェアリング機能の実現**が望まれる.

当然のことであるが, RWC-1 本来の性能を損なわないよう, OS オーバヘッドは最小限に抑える必要がある.

4 タイムシェアリングとリアルタイムの実現

超並列計算機においてタイムシェアリングとリアルタイムを実現することを検討した結果, 主にネットワークに OS 支援の観点が抜けていると考えられる. 以下に, タイムシェアリングとリアルタイムを実現するための, ハードウェアおよびソフトウェアレベルでどのような機能が必要かという点について述べることにする.

4.1 プロセス切替時間の保証

リアルタイム処理を実現するには, プロセス切替時間をできるだけ早くすると同時に, 切替に要する時間を保証しなければならない. このことは実用的なタイムシェアリングを実現する上においても重要である.

疎結合型並列計算機においては, ネットワーク上を流れるメッセージをどのように処理するかが, プロセス切替を実現する際に問題となる. CM-5 ではパーティション単位でタイムシェアリングが実現されている [8]. AFD (All Fall Down) と呼ばれるネットワーク操作により, ネットワーク上のメッセージを直ちに最寄りのプロセッサに退避することができる. これによりプロセス切替時間を保証することが可能になる.

しかしながら CM-5 ではメッセージの送出がアトミックな操作でないため[1], 送信途中に AFD モードに入るとそのメッセージ送信が失敗してしまう. このため, 全てのメッセージ送信の終了後にはメッセージ送信の正常終了を確認し, 失敗していた場合には再度同じメッセージを送信してやらなければならない [9]. また, 理論上はプロセス切替の時間間隔をメッセージ送信が完了する最長時間よりも長くしなくてはならない. このことは, CM-5 のようにタイムシェアリングのタイムスロット間隔でのみプロセス切替を起こす場合は良いが, リアルタイムのように事象によりプロセス切替をする必要がある場合は問題となる可能性がある.

より効率的にプロセス切替を実現するには, 任意のタイミングで, ユーザプログラムに意識させずに, プロセスが使っているネットワークを解放／復帰できなければならない. RWC-1 では, 任意のタイミングでネットワーク上のメッセージをプロセッサに退避する操作を **"Drain"** と呼び, AFD と区別することにした [19].

より一般的に言えば, 疎結合型並列計算機におけるネットワークは, プロセッサと同じ程度に重要な計算のための資源であり, プロセス切替時にはネットワークの**横取り (Network Preemption)** 機能が必要となる.

[1] 正確には, アトミックでなくてもネットワークインターフェイスおよびルータの状態を退避／復帰可能であればよい. しかし, CM-5 は間接網であるため, 全てのルータの状態を退避／復帰することは難しい.

4.2 プロセス相互干渉の排除

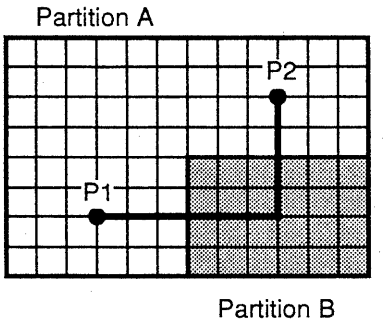
リアルタイム処理ではプロセス間の相互干渉をできるだけ排除する必要がある．逐次計算機においては，割込に優先度を設けることでプロセッサ資源の確保を保証するが，並列計算機においては，ネットワークも計算に必要な資源とみなし，ネットワークに関連したプロセス間の相互干渉に注意しなければならない．

プロセス間の相互干渉を排除するには，以下の事項についてハードウェア／ソフトウェアの両面から検討される必要があると考えられる．

パーティション分割の特性

ネットワークをパーティションで分割した場合，ネットワークトポロジー，パーティションの形状，ルーティング方式に依存して，ネットワークのパーティション分割における相互干渉性に関して，以下のように分類／定義する（直接網の場合）．

- 開パーティション (Open Partition)
  パーティション内のプロセッサ間通信のメッセージが，パーティションを越えて他のパーティションを横切るようなもの（図2）．相互干渉あり．



2次元メッシュトポロジーネットワークにおける「開パーティション」の例．パーティション A のノード P1 から P2 へメッセージを送った場合，パーティション B を経由する（X-Y ルーティング）．

図 2: 開パーティションの例

- 閉パーティション (Closed Partition)
  可能な全てのパーティション分割において，パーティション内のいかなるプロセッサ間通信メッセージもパーティションの外に出ないようなもの．相互干渉なし．

開パーティションにおいては，あるプロセスの通信が他のプロセスを実行中のネットワークを閉塞させる可能性があり，結果的に他のプロセスの計算を阻害することが考えられる．このため，並列計算に必要なネットワークの確保が保証できない．リアルタイム処理を実現するには閉パーティションを実現する必要がある．

パーティションの閉鎖性

プログラムが誤ってパーティション外のプロセッサに対しメッセージを大量に送信するような場合では，不正なパケットを受信したプロセッサではその対応に追われることになる．このため，パーティションの外に不正に出るようなメッセージは送信時にチェックされ，パーティションの外にメッセージが出ないようにする必要がある．

IO 通信の分離

IO のように大量のデータがプロセス（パーティション）にまたがって往来する場合は，IO と関係のないパーティションを経由させない工夫が必要である．これを実現するには，以下の3通りの方法が考えられる．

- 全てのノードに IO デバイスを接続する方式
- ひとつのネットワークにおいて IO メッセージのパスと計算メッセージのパスと別にする方式
- 計算のためのネットワークと IO ネットワークを別にする方式

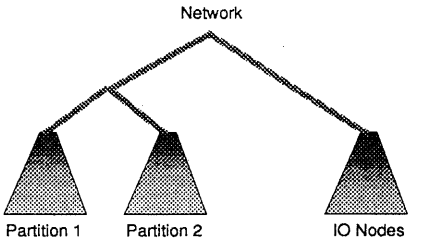


図 3: CM-5 の IO 通信

CM-5 では，樹状 (Fat Tree) の間接網の一方の枝に IO ノード，反対の枝に計算ノードの

パーティションを置く方式となっている（図3）．RWC-1 では IO 専用ネットワークを別に設ける方式とする予定である [12].

5 *SCore* カーネル

以下に, RWC-1 のオペレーティングシステムカーネルである *SCore* の概略について述べる.

5.1 設計方針

マイクロカーネル

超並列 OS の研究は始まったばかりであり, 研究的な要素も多い. このため, OS はマイクロカーネル構造とし, モジュラリティを高め, モジュール交換により様々な実験を容易とする構造とする. カーネルは全てのプロセッサに常駐し, 基本的にはそのプロセッサに関する資源管理を主に担当するが, 場合によっては他のプロセッサのカーネルと通信し, 協調してプロセスあるいはシステム全体の管理をおこなう.

言語独立

カーネルはプログラミング言語に独立した設計とする.

リアルタイム／タイムシェアリング

リアルタイム／タイムシェアリングを可能とする.

5.2 カーネルアブストラクション

プロセス

資源管理の単位であると同時に, パーティション内での大域的なアドレス空間でもある. プロセスはパーティションで指定された複数のプロセッサで並列動作する.

スレッド

RWC-1 は EM-4 と同様にハードウェアレベルでスレッドの実行が制御される [1, 5]. このためスレッドの生成／制御にカーネルの介在は必要がない. しかしながら, 例外あるいはカーネルコールが発生したスレッドに関してはカーネルが管理する.

パーティション

プロセスにプロセッサ資源を割り当てる単位であり, プロセッサの集合である. 全てのパーティションは閉パーティションでなければならない. ひとつのパーティションには複数のプロセスが並列動作可能であり, カーネルや OS はユーザプロセスと同じパーティションを共有するプロセスとして実現される. 複数のユーザプロセスが同じパーティション内で同時に実行されるような場合も考えられる.

プロセスのスケジューリングはパーティション毎に設定されたスケジューラにより管理される. パーティションは入れ子関係にある階層構造とすることも可能とする. プロセス間通信を必要とする複数のプロセスは兄弟パーティションとし, プロセス間通信の影響が他の関係のないプロセスに及ぶことを防ぐようにする. また, 兄弟関係にあるパーティションを親パーティションから同時にスケジューリングすることで, co-scheduling が可能となる [3].

パーティションの大きさは時分割タイムスロット毎に変えることも可能である（図4）. パーティションに設定できるスケジューラはカーネル外部に設けることも考える.

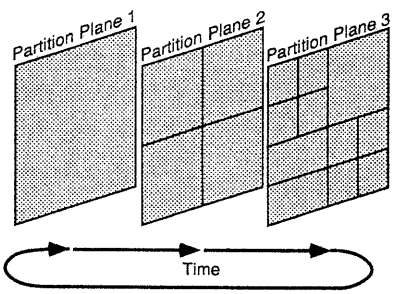


図 4: タイムスロット毎のパーティション

パーティションの管理方式としては Processor Pool Model[7, pp. 530–533] をベースに, パーティションの集合からアプリケーションに必要な大きさの比較的暇と思われるパーティションを割り当てるといった方式が考えられる.

仮想記憶

仮想記憶管理は 2 階層ある [13]. ひとつはパーティションの管理であり, パーティションという仮想的な並列計算機環境（プロセッサ群

とサブネットワーク）を提供し，パーティション外への不正なメッセージ送信を阻止するものである．もうひとつは，従来の計算機同様，プロセッサ内のページ単位での仮想記憶管理である．Mach[4] のようにカーネル外部にページャを設定することも可能とする予定である．

Mach 同様，DSM (Distributed Shared Memory) [4] あるいは SVM (Shared Virtual Memory) [2] を実装することにより，ページ単位でのプロセッサ間共有メモリモデルを提供することも可能である．しかしながら，RWC-1 では高速なネットワークによる低レーテンシな通信が可能なため，一貫性維持のためのオーバヘッドは相対的にコスト高となることが予想される．このため，DSM はプログラムテキスト（コード）のように Read Only (Write Once) なデータに関してのみ有効であると考えられる．

プロセス間通信

中大容量のプロセス間通信を効率良く実現するため，プロセス間の共有メモリを提供することを考える．これは Mach などに見られる copy-on-write のテクニック [10] と基本的に同じである．当然のことであるが，パーティションが異なるプロセスにおいては，データのコピーは避けられない．しかし，IO デバイスプロセスを同じパーティションに配置すれば効率的な IO が実現できることや，統一的なインターフェイスという意味で，プロセス間の共有メモリを IPC サービスのひとつとして提供する予定である．

プロセスの排他制御

リアルタイム処理を考慮して，優先度継承 [6] のあるセマフォあるいは MUTEX 程度をサポートする予定である．プロセス内のスレッド間の排他制御は言語処理系がサービスするものとする．

5.3 2次記憶ページング

UNIX を始め，現在の OS の多くに2次記憶への要求時ページングがサポートされている．しかしながら，超並列計算機において2次記憶へのページングは逐次計算機の場合ほど簡単ではない．

並列計算機におけるページイン／ページアウトの時間を隠蔽する場合，ページフォールトが発生した場合に何をサスペンドさせるかで、以下に示す3つの方式が考えられる．

- プロセス全体をサスペンド
  ワーキングセットの大きさが PE 台数に比例すると仮定すると，ページフォールトの発生頻度は，単体 PE の場合の発生頻度に PE 台数を掛けたものになり，プロセス全体でページフォールトが頻発することになる．

- プロセッサをサスペンド
  ページイン／ページアウト時間（数十ms）を考えると，そのプロセッサが停止することでメッセージの処理が滞り，その影響がパーティション全体に及ぶ恐れがある．

- スレッドのみサスペンド
  SPMD 型プログラミングでは性能が著しく低下する恐れがある．

いずれにせよ，並列計算機における要求時ページング逐次計算機程の効果は期待できそうにない．要求時ページングの有効性が認められそうなのは MIMD 型プログラムにおけるスレッドサスペンド方式である．この点に関してはカーネル実装後の研究課題としたい．

5.4 言語処理系まかせの部分

以下に示す機能に関してはカーネルあるいは OS のサービスとはせず，言語処理系が提供するサービスとする．

- プロセス内のスレッド間の排他制御
- バリア同期
- オブジェクトマイグレーション
- ガーベジコレクション
- オペランドセグメント管理 [5]

6 RWC-1 の OS 支援機能

OS あるいはソフトウェアという立場から，RWC-1 のハードウェア的特徴をまとめると以下のようになる [11, 13, 19, 12].

大域仮想記憶

RWC-1 は基本的に疎結合並列計算機であるが, 他の PE のメモリをアドレッシング可能な大域仮想記憶がサポートされている. また, プロセッサ空間であるパーティションは大域仮想記憶のマッピングで実現/保護される.

大域仮想記憶により, 1) アドレスがそのまま大域的な名前となる, 2) プロセッサ間通信においてポインタの輸出入が可能, という利点が生じる.

スレッド

RWC-1 ではスレッドの実行/制御が全てハードウェアで実現されている. このため, スレッド切替に OS が介在する必要がなく, 高速な切替が可能である.

スレッド間の同期

スレッド間の同期に関するハードウェアサポートがある.

優先度

ハードウェアはスレッド実行に関して 4 つの優先度をサポートする予定である. このうち高い方のふたつはカーネルおよび OS が使用し, ユーザには残りのふたつの優先度を開放する. カーネルおよび OS はその性格上, ユーザプロセスより高い優先度で実行されなければならない.

プロセッサ間通信

プロセッサ間通信は全てハードウェアがおこなうため, 通信のたびにカーネルコールを発生させる必要がない. また, 通信のためのパケット生成はパイプライン化されると同時に, 演算パイプラインと融合されている. このため遅延の少ない高速なプロセッサ間通信が可能である.

ネットワーク

ネットワークはソフトウェアの指定によりパーティション分割可能であり, 閉パーティションが実現される予定である. また, パーティション単位でルーティング中のメッセージを最寄りのプロセッサのメモリに退避すること (drain) が可能である.

IO ネットワーク

IO 専用ネットワークがあり, IO デバイスは IO ネットワークに接続される. このため, 計算に使われるプロセッサ間通信とは独立に IO 通信が可能である.

7 おわりに

本論文では, 超並列計算機 RWC-1 におけるオペレーティングシステムカーネル *SCore* の概要について述べた. 同時に, タイムシェアリングやリアルタイムを実現するような本格的な OS 機能を, 超並列計算機上で構築するために必要なハードウェア機能についても検討した. その結果, ネットワークに関して, 閉パーティション, パーティションの閉鎖性, IO 通信の分離などの特性が必要であることが判明した.

SCore カーネルは実用性を重視し, RWC アプリケーションを実行するための環境を整えることを目的とする. その意味で, ワークステーション並の使い心地を実現するタイムシェアリングやその性能を十分に発揮するためのリアルタイム機能を提供する予定である. 細部に関しては, その詳細が決まり次第, 機会を見て発表する.

謝辞

RWC 超並列ソフトウェアワークショップならびに RWC 超並列アーキテクチャWG に参加の各位からは数々の貴重なアドバイスを頂いた. ここに感謝の意を表す.

参考文献

[1] Yuetsu KODAMA, Shuichi SAKAI, and Yoshinori YAMAGUCHI. A prototype of a highly parallel dataflow machine EM-4 and its preliminary evaluation. *Future Generation Computer Systems, NORTH-HOLLAND*, pp. 199–209, 1991/92.

[2] Kai Li. *Shared virtual memory on loosely coupled multiprocessor*. PhD thesis, Yale University, 1986.

[3] John K. Ousterhout. Scheduling Techniques for Concurrent Systems. In *Proceedings of Third International Conference on Distributed Computing Systems*, pp. 22–30, 1982.

[4] Richard Rashid, et al. Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures. In *Second International Conference on Architectual Support for Programming Languages and Operating Systems (ASPLOS II)*, pp. 31–39, October 1987.

[5] Mitsuhisa Sato, Yuetsu Kodama, Shuichi Sakai, Yoshinori Yamaguchi, and Yasuhito Koumura. Thread-based Programming for the EM-4 Hybrid Dataflow Machine. In *Proceedings of 19th Annual International Symposium on Computer Architecture*, pp. 146–155, 1992.

[6] Lui Sha, Ragunathan Rajkumar, and John P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. *IEEE Transactions on Computers*, Vol. 39, No. 9, pp. 1175–1185, September 1990.

[7] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice-Hall, 1992.

[8] Thinking Machines Corporation. *Connection Machine CM-5 Technical Summary*, November 1992.

[9] Thinking Machines Corporation. *NI Systems Programming*, October 1992. Version 7.1.

[10] Michael Young, et al. The Duality of Memory and Communication in the Implementation of a Multiprocessor Operating System. *Operating System Review*, Vol. 21, No. 5, pp. 63–76, 1987.

[11] 岡本一晃, 松岡浩司, 廣野英雄, 横田隆史, 堀敦史, 児玉祐悦, 佐藤三久, 坂井修一. 超並列計算機 RWC-1 における同期構成. 情報処理学会計算機アーキテクチャ研究会, August 1993.

[12] 廣野英雄, 松岡浩司, 岡本一晃, 横田隆史, 堀敦史, 児玉祐悦, 佐藤三久, 坂井修一. 超並列計算機 RWC-1 における入出力機構. 情報処理学会計算機アーキテクチャ研究会, August 1993.

[13] 松岡浩司, 岡本一晃, 廣野英雄, 横田隆史, 堀敦史, 児玉祐悦, 佐藤三久, 坂井修一. 超並列計算機 RWC-1 における記憶構成. 情報処理学会計算機アーキテクチャ研究会, August 1993.

[14] 坂井修一, 岡本一晃, 松岡浩司, 広野英雄, 児玉祐悦, 佐藤三久, 横田隆史. 超並列計算機 RWC-1 の基本構想. In *JSPP'93*, pp. 87–94, 1993.

[15] 通商産業省機械情報産業局（編）. 4 次元コンピュータ -リアルワールドコンピューティング (RWC)-. 財団法人日本情報処理開発協会, 第 1 版, 1993.

[16] 石川裕, 堀敦史, 小中裕喜, 前田宗則, 友清孝史. RWC 超並列基本ソフトウェアの概要. 日本ソフトウェア科学会第 10 回大会論文集, pp. 13–16, 1993.

[17] 石川裕, 小中裕喜, 前田宗則, 友清孝史, 堀敦史. 超並列プログラミング言語 MPC++ の概要. 情報処理学会プログラミング - 言語・基礎・実践 - 研究会, August 1993.

[18] 小中裕喜, 石川裕, 前田宗則, 友清孝史, 堀敦史. 並列オブジェクトベース言語 OCore の概要. 情報処理学会プログラミング - 言語・基礎・実践 - 研究会, August 1993.

[19] 横田隆史, 松岡浩司, 岡本一晃, 廣野英雄, 堀敦史, 児玉祐悦, 佐藤三久, 坂井修一. 超並列計算機 RWC-1 の相互結合網. 情報処理学会計算機アーキテクチャ研究会, August 1993.