システムソフトウェアと
オペレーティング・システム 76－10
（1997．8．21）

並列アプリケーションを指向した分散システム コンピュータ・コロニーの構想

山添博史† 田中慎司†† 伊達新哉††
五島正裕† 森眞一郎† 富田眞治†

本研究では、並列アプリケーションの実行を指向した分散システムとして“コンピュータ・コロニー”を提案する。コンピュータ・コロニーは、ネットワークで結合された要素マシン群と、分散環境における並列処理を支援するオペレーティングシステム“Colonia”で構成され、共有メモリをベースとした低コストな通信機構を提供する。

高速なメモリアクセスのためには、キャッシュの利用が不可欠である。リモートメモリの内容をローカルメモリあるいはプロセッサに附属するキャッシュに置くことで、高速なメモリアクセスが期待できる。よって、コンピュータ・コロニーが高速かつ低遅延な共有メモリアクセスを実現するには、キャッシュコヒーレンシプロトコルが制御可能な通信ハードウェアが要求される。

Computer Colony: A Distributed System for Efficient Execution of Parallel Applications

HIROSHI YAMAZOE,† SHINJI TANAKA,†† SHIN-YA DATE,††
MASAHIRO GOSHIMA,† SHIN-ICHIRO MORI† and SHINJI TOMITA†

We propose “computer colony”, which is a distributed system oriented for the execution of parallel applications. Computer colony consists of machines connected with networks, and operating system “Colonia”, it provides us a low-cost communication mechanism based on shared memory.

Effective use of cache is indispensable for fast memory access. Memory access is expected to become fast by bringing contents of remote memory on local memory or cache belong to processors. Therefore, to achieve fast and low-latency shared memory access, computer colony requires a communication hardware which can operate cache-coherency protocol.

1. はじめに

比較的低コストで導入や拡張が可能な分散システムが、さまざまな場面で活用されている。ディスクやプリンタなどの資源がネットワーク結合されていて共有利用でき、通常の業務を行うには充分な環境が整っている。しかし、ときに大規模なジョブを投入する必要があると、分散システムは1台の要素マシンの性能以上のものを出すことができず、実行に非常に長い時間を要する。また、その1台に非常に重い負荷がかかることになり、他のユーザはそのマシンでの業務を行うことがほとんどできない。

† 京都大学大学院工学研究科情報工学専攻
Division of Information Science, Graduate School of Engineering, Kyoto University
†† 京都大学工学部情報工学科
Department of Information Science, Faculty of Engineering, Kyoto University

本研究では、分散システムには大きな魅力があるが、その単なる延長のみではこれらの問題点を克服できないと考え、新しい計算機システムとして“コンピュータ・コロニー”を提案し、ハードウェアとソフトウェアを再構成する。次に、コンピュータ・コロニーの目標を挙げる。

低価格性 実際の問題として無視できないのが価格の問題である。一般のユーザに手が届かない価格の大型並列計算機などではなく、もっと身近に導入や拡張ができるシステムが必要である。

計算機資源の有効活用 複数ある計算機のプロセッサ資源を有効に利用するには、並列処理が有効である。SMP(対称型マルチプロセッサ)機の普及に伴い、将来は並列アプリケーションの増加が期待され、並列処理を行う意義が大きくなると考えられる。

マルチユーザ環境 ユーザそれぞれが要求するQoS(Quality of Service)を満足しつつ、特定のユーザのみが

計算機資源を占有することのないよう、システム全体に亘るスケジューリングが必要である。

以下、2章で現在の計算機システムの背景について、3章でコンピュータ・コロニーの概要について、4章で本研究室での実験環境の実装について述べる。

2. 背　　景

2.1 集中システムと分散システム

複数のプロセッサを含む計算機環境としては、主に集中システムと分散システムの2つが挙げられる。集中システムは1つの筐体に多数のプロセッサを搭載して並列計算機を構成し、ユーザは端末を通して利用するというものであり(図1(a))、分散システムは端末としても使える、独立した計算機をネットワーク結合するというものである(図1(b))。以下、いくつかの観点から両者を比較する。

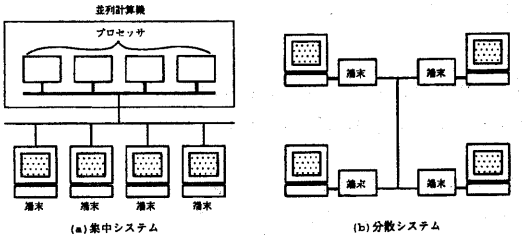

図1　集中システムと分散システム

価格性能比

計算機を導入するにあたって、その価格は重要な判断材料である。いかに高性能なシステムであっても、予算の限界を越えてしまっては現実的に導入は不可能である。

図2に、同種のプロセッサで構成される、ある並列計算機とワークステーションのプロセッサ台数と価格の関係を示す。これだけで並列計算機とワークステーションの価格には倍以上の開きがあることが分かる。

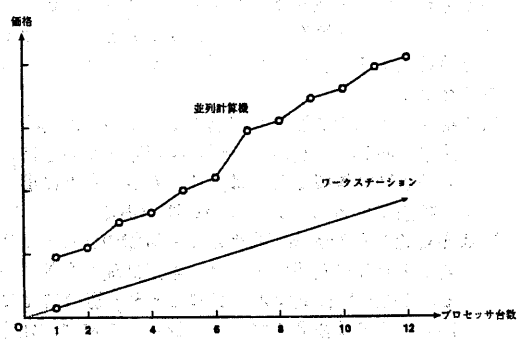

図2　並列計算機とワークステーションの価格

これは、並列計算機は販売台数が少なく大量生産ができないが、ワークステーションなどでは大量販売、大量生産によって低価格化が起こっているためと推測される。この状況は技術革新などによっては変わらず、並列計算機の低価格化は今後も起こらないと考えられる。

拡張性

集中システムではプロセッサが近傍に配置されるため、プロセッサ間の物理的な通信速度の向上は容易であるが、これは拡張性を犠牲にした上でのことである。確かにプロセッサを1つの筐体内に固定して配置すれば内部での高速化はしやすいが、反面プロセッサの交換や追加は分散システムと比べて制限が多い。

現時点で最も高速な計算機を購入したとしても、それはすぐに「遅い」部類の計算機になってしまう。そのようなときに拡張性に乏しい並列計算機は、その資源を再利用することが困難であり、いずれは完全に捨て切って新しい計算機に入れ換える必要がある。

これに対し分散システムの大きな特徴としては、柔軟な拡張性が挙げられる。分散システムを構成する要素マシンは基本的に独立しており、アップグレードや要素マシン台数の追加の制限は比較的緩やかである。このことによって、既存の計算機資源を継続して有効に活用できる。

マルチメディアへの対応

分散システムは集中システムに比べてマルチメディア処理に向いているという特徴がある。動画の再生などには高速な画像処理が必要であるが、それにはプロセッサとグラフィックス・アクセラレータがより近接して結合されている必要がある。分散システムを構成する要素マシンがこのような構成をとることは容易だが、集中システムにおいてはディスプレイと並列計算機本体が離れているためこのような構成をとることは困難である。

このように分散システムは、主に処理の高速化に重点を置く集中システムと異なり、価格性能比やそれを補う拡張性などから、一般ユーザにとってより身近である。このため、高価な並列計算機を中心とする集中システムよりも普及しており、利用しやすい環境となっている。

2.2 従来の分散システムの問題点

分散システムは、低価格で利用できるため一般のユーザにとって身近であり、また拡張性やマルチメディアへの適性を考慮すると、将来的に可能性の高いシステムである。しかし、従来の分散システムでは、ネットワーク結合された計算機資源の性能を最大限に発揮させることはできない。これは、次に挙げる事柄が理由

として考えられる。

- 通信コストが高い
- 有効な負荷分散が図られていない
- マルチユーザへの対応が不十分

並列化されたアプリケーションを実行させようとしても、通信コストが高ければ到底並列化の効果は期待できない。またもし、通信コストを充分低くできたとしても、負荷分散やマルチユーザへの対応など、さまざまな意味での透過性に欠けている。以下、上の理由のそれぞれについて述べる。

通信コストが高い

分散システムにおけるネットワークを経由した通信は、集中システムにおける並列計算機内のプロセッサ間の通信に比べ、従来の方法ではコストが高い。よく見られる通信方法は、システムコールを介してI/Oにアクセスすることで通信を確立し、ライブラリを用いてメッセージ通信や共有メモリを実現するというものである。このような方式ではいくらネットワークが高速でも、そこに至るまでの過程においてライブラリやシステムコールなど、ソフトウェアによるオーバーヘッドが大きい。これでは細粒度の並列処理を行うと、かえって処理性能が落ちる可能性さえある。

有効な負荷分散が図られていない

並列処理を有効に実行するには、高速な通信を実現するとともに、最適な負荷分散を行うことが重要である。負荷状況は刻々と変化するので、システム全体が常に現時点での負荷状況を把握し、適切な負荷の配置を行わなければならない。

従来の分散システムにおいては、計算機の処理能力や負荷状況をユーザが把握し、明示的に負荷の配置を指定しなければならない。これでは負荷状況の変化には対応できず、最適な負荷分散は行われない。

ユーザは分散システムを単一の高性能計算機としてとらえて処理を投入し、システムが自動的に負荷を管理して均質化するということが必要である。

マルチユーザへの対応が不十分

動画や音声の再生など、リアルタイム性の高いアプリケーションの実行には、一定時間ごとに一定の処理能力の割り当てが必要である。このように、ユーザが実行するアプリケーションによって要求するQoSは異なり、それに応じたスケジューリングが必要となるが、マルチユーザに計算機資源を提供する分散システムとしては、単一のユーザのみに集中した処理能力を割り当ててはならない。あるマシンにおいて優先順位が高くなっているユーザに対しては、他のマシンでは優先順位を下げるなどして、システム全体としての観点から計算機資源の占有を避ける必要がある。

従来の分散システムではこのような、ユーザの要求に応じかつユーザ間のバランスを考慮したスケジューリングが実現されていない。

各要素マシンは単一プロセッサを基本とするハードウェアと、単独で稼働するオペレーティングシステムがもとになっており、I/O機能の1つとして通信ハードウェアとソフトウェアを付加したものが多い。これでは、通信は追加機能であり、各計算機は独立して稼働するというしくみが根本から抜けきっておらず、並列処理を高速化するための足枷となっている。

3. コンピュータ・コロニー

3.1 設計目標

2章で見てきたように、分散システムは集中システムより低価格で導入、維持が可能であり、通常の業務に関しては充分な性能を示すが、計算機資源の有効利用が図られていない。この点を克服するには、単なる追加機能ではなく、不可欠な基本機能としてネットワークを構成するためのハードウェア及びソフトウェアを再構成する必要がある。

本研究では、分散システムの利点を採り、さらにシステム全体を意識してその計算機資源を最大限に活用できる計算機システムとして、“コンピュータ・コロニー”を開発している。コロニーとは生物でいう“群体”であり、複数の個体が密着して共同生活を行い、全体として1つの個体であるかのように振舞うものをいう。コンピュータ・コロニーは、群体のように複数の要素マシン(以下、ノードと呼ぶ)が互いに密に連絡して並列処理や負荷分散、システム全体に亘るスケジューリングなどを行い、1つの高機能な計算機であるかのように振舞うことを目標としたシステムである。

コンピュータ・コロニーの設計目標は、次のようなものである。

- 低価格性
- 計算機資源の有効活用
- マルチユーザ環境

以下にこれらの設計目標について述べる。

低価格性

従来の分散システムと同じく、各ノードは低価格の計算機であり、システム全体も低価格で導入、維持できなければならない。一般のユーザにとって予算は切実な問題であり、その範囲内で最大限の性能を得ることが必要である。これには、各ノードのモジュラリティが不可欠である。つまり、古くなったプロセッサを交換したり、新しくプロセッサ台数を追加したりということが容易にできることで、それまでの計算機資源を継続して利用し、低価格で大幅な性能向上を図ることができる。

計算機資源の有効活用

複数のプロセッサ資源を有効に活用し、並列処理を実行させることで、システムの性能を大きく引き出すことができる。従来の分散システムでは細粒度の並列処理は不可能なので、これを集中システムに近づけるような通信機構を提供するハードウェアが必要となる。我々は単にバンド幅の高いネットワークデバイスを用いるだけではなく、ソフトウェアによるオーバーヘッドを極力省くことで大幅な速度向上を図る。

マルチユーザ環境

リアルタイム性が要求される動画や音声の再生などを実行するためには、単に均等にユーザに処理能力を提供するのではなく、QoS に応じた提供が必要である。しかし、これを満足すると同時に、特定のユーザのみが計算機資源を占有することはあってはならない。このためコンピュータ・コロニーはシステム全体を見渡し、重いジョブは負荷の少ないノードにおいて実行するなどして、他のユーザに提供する処理能力を損なわうことのないようスケジューリングを行う必要がある。

コンピュータ・コロニーにおいては、並列処理を行うための高速かつ低遅延な通信機構を提供するハードウェアと、マルチユーザに資源全体を 1 つの高性能計算機に見せるオペレーティングシステム "Colonia" が、互いに緊密して稼働することが必要となる。

3.2 共有メモリベースの通信

並列処理を高速化するにあたって、最優先で解決すべき問題は、通信のコストである。通信を高速化するには、物理的にバンド幅の高いハードウェアを用いることも重要だが、それを生かすには、遅延が小さい通信機構を設計しなければならない。

遅延の主な原因として、ソフトウェアによるオーバヘッドが挙げられる。通常、ユーザが I/O を操作するには、システムコールを発行する必要がある。よって、通信の遅延を削減するには、できるだけシステムコールを避ける、つまりユーザに解放できる通信方式を採用することが必要である。

コンピュータ・コロニーでは、通信プリミティブとして共有メモリを用いることで、高速かつ低遅延な通信機構をユーザに解放する。共有メモリ方式においては、ユーザはシステムから与えられる仮想メモリ空間に対してアクセスすることによって通信を行う。この時点ですでに他のユーザのアクセスからは保護されており、オペレーティングシステムによるメモリ保護機構は必要ない。通信機構はユーザに解放され、システムコールによるオーバーヘッドは削減できる。

3.3 通信ハードウェアの要件

前述したような共有メモリを実現するには、通常のメモリアクセスに対応してリモートメモリアクセスを行うハードウェアが必要になる。しかし、単にシステムコールを省いただけでは、やはり主記憶へのアクセス速度に近づくことはできない。各ノードにおいてあたかも主記憶が増設されたかのようにリモートアクセスするためには、通信ハードウェアは次のような要件を満たさなくてはならない。

共有データのキャッシング リモートアクセスにおいても、参照の局所性を利用し、プロセッサからできるだけ高速にアクセスできる記憶装置にデータを置くことで、大幅な高速化が期待できる。具体的には、ローカルなキャッシュや主記憶にリモートメモリの内容をキャッシングする。

主記憶を共有メモリとして利用 新たに共有データ用メモリを増設するのではなく、現在ある主記憶を共有データを置く場所として使用することで、計算機資源の有効活用、ならびに大容量の共有メモリが可能である。

現在多く提案されている共有メモリ用の通信ハードウェア (Memory Channel や SHRIMP など) は、高速ネットワークを I/O バスに接続するというものであり、仮想メモリ空間にマッピングすることにより、ユーザがオペレーティングシステムの介在なしにアクセスすることはできる。

しかし、いくらネットワークデバイスや I/O バスのバンド幅が高く、仮に 1 回のリモートアクセスにかかる時間が主記憶へのアクセスと同じだとしても、この接続方法では主記憶へのアクセスに比べてリモートアクセスの速度は総合的には遠く及ばない。

これは、主記憶へのアクセス時にはプロセッサは近傍にある高速なキャッシュメモリにデータをキャッシングできるが、I/O バスを通して接続されるリモートメモリについてはそれは実際上不可能であることによる。プロセッサに比べ主記憶アクセスの遅さは益々目立ってきており、キャッシングを抜きにした高速なメモリアクセスは考えられない。

キャッシュの利用が事実上不可能なのは、I/O バスに接続された通信ハードウェアからは、システムバス側のプロセッサやメモリに対して、キャッシュコヒーレンシ制御に関する操作ができないからである。一般にキャッシュコヒーレンシ制御はシステムバスに接続される複数のプロセッサとメモリ間で行われ、I/O はそれに関わらない。よって、システムバスと I/O バスのインターフェイスは、キャッシュコヒーレンシ制御に関する機能を I/O バス側に提供しておらず、I/O バスに通信ハードウェアを接続しても主記憶やキャッシュは利用できないのである。

もし I/O バス側の通信ハードウェアがノード内の

記憶装置にキャッシングを行うのであれば、共有データやそのキャッシュは、コヒーレンシ制御が可能な範囲になければならないため、通信ハードウェアが保持することになる。

しかしこの場合は、主記憶を共有メモリとして利用できず、計算機資源を有効活用しているとは言えない。これでは計算機を結合して共有メモリを実現するというよりも、共有メモリにプロセッサを接続するというような形態になってしまう。また、共有データをキャッシングしてもプロセッサがアクセスするにはI/Oを通さねばならないため、プロセッサの近傍に配置されるキャッシュへのアクセス速度には到底及ばない。

いずれにしても、I/Oバスに接続する形態の、従来の共有メモリモジュールでは、すでに存在する大容量の主記憶を共有メモリとして扱い、それをキャッシングすることが不可能なため、高速な共有メモリアクセスは実現できないのである。

3.4 通信プロトコル

コンピュータ・コロニーが必要とするネットワークは、共有メモリに特化した専用ネットワークであるため、TCP/IPのような繁雑な汎用プロトコルは必要ない。よって単純なプロトコルを採用し、プロトコル処理にかかるオーバヘッドを削減する。

ネットワークの具体的なデバイスは、高いバンド幅を持つものである必要があるが、プロトコル等に関してはコンピュータ・コロニーとしての要求は特にない。よってコンピュータ・コロニーを構成するネットワークデバイスはほぼ任意であり、より高速なものに交換しても構わない。

下位プロトコルはネットワークデバイスに依存するが、上位プロトコルはネットワーク透過な共有メモリを実現するためにコンピュータ・コロニーにおいて定義される。

コンピュータ・コロニーの通信ハードウェアは、特定の物理アドレスに対するアクセスを検出してリモートアクセスを行う。このとき、物理的なノード番号や物理アドレスでリモートメモリを指定すると、柔軟性に欠ける。

メモリの物理的なアドレスは、スワップアウトによって常に変化する。よってユーザが用いる仮想アドレスを物理的な位置に変換する機構が必要になるが、リモートメモリアクセス要求を送信するノードがこの変換を行うことにすると、物理的な位置の変更のたびに全ノードの変換テーブルの変更が必要になる。

そこで、リモートメモリアクセスの際には、ネットワーク仮想アドレスを用いてアクセス先を指定する。アクセス要求を受信したノードは、変換テーブルを用いてノード内の物理アドレスに変換し、実際のアクセスを行う。物理アドレスの変更は、受信ノードの変換テーブルを変換するだけで行える。

1つの実現例として、ネットワークで唯一であるアクティビティの識別子と、そのアクティビティの持つ仮想アドレスで、ネットワーク仮想アドレスを表現するという方法が考えられる。よって、物理アドレスをもとに通信を開始する送信側ノードでは逆引きページテーブルを参照することで仮想アドレスを取得し、受信側ノードではページテーブルを参照することでその仮想アドレスに対応する物理アドレスを取得することになる。このようすを図3に示す。

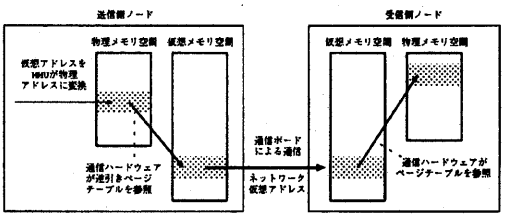

図3 ネットワーク仮想アドレスを用いたリモートアクセス

4. 実験環境

4.1 通信ボードの実装

前述したように、通信ハードウェアをI/Oバス側に接続するのはさまざまな制限が多く、効率よい並列処理を実現するための共有メモリ通信機構を提供できない。コンピュータ・コロニーでは、キャッシュに関して完全な操作ができる(マルチプロセッサを実装できる)システムバスに、専用通信ボードを接続するという方針を採る。

マルチプロセッシング可能なシステムバスにボードを接続すると、それはキャッシュを持つプロセッサと同様に振舞うことができる。プロセッサのキャッシュは物理アドレスをスヌープすることで、自身の持つキャッシュの状態を変更したり、メモリの反応を停止して自身の持つデータを送信したりする。この代わりに通信ボードを接続することで、物理アドレスをスヌープしてそのアクセスに対してアクセスの停止や自分が持つデータの無効化などが行えるのである。つまり、システムバスに接続された通信ボードは、あたかもリモートメモリすべてを扱うキャッシュメモリであるかのように振舞う。

本研究では、コンピュータ・コロニーの実現例として、現在研究室にあるワークステーションSPARCstation 20を結合してコンピュータ・コロニーを構成する。目的の高速、低遅延な通信を実現するために、我々は専用通信ボードをシステムバスであるMBusに接続する。ネットワークデバイスは前述のように特に要求は厳しくなく、本研究室における実験環境ではFibre Channelを用いる。このときのハードウェア構成を図4に示す。

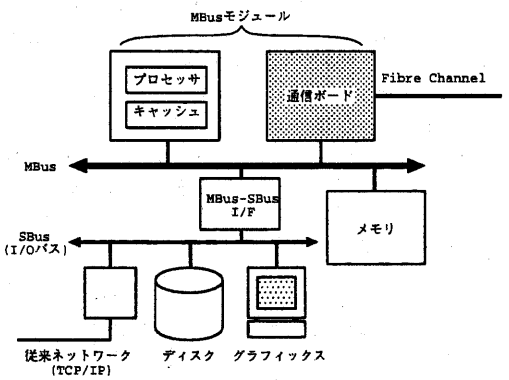

図4　MBus に接続した通信ボード

4.2 MBus の仕様

SPARCstation 20 はマザーボード上に複数の MBus スロットを持つ。ここに標準でプロセッサを搭載した MBus モジュールが装着されている。MBus モジュールを交換することで古いプロセッサを新しいプロセッサに交換したり、あるいは外部キャッシュ付きのプロセッサに交換することができる。また、他の MBus スロットに別のプロセッサを持つモジュールを追加することで、SMP(対称型マルチプロセッサ) に拡張することができる。

SPARCstation 20 の MBus は、キャッシュコヒーレンシ制御のための信号をサポートしており、複数のキャッシュがスヌープすることで一貫性を保つことができる。この信号には、次の 2 つがある。

MIH(Inhibit)　無効なメモリに対するアクセスがあった場合、有効なデータを持つキャッシュがアサートする。

MSH(Shared)　有効なメモリに対する読み込みがあった場合、そのデータを共有するキャッシュがアサートする。

MIH がアサートされると、メモリは要求されているアドレスへのアクセスを停止する。これによって、無効なデータへのアクセスを防ぐことができる。例えば、あるキャッシュに変更されたデータがあり、メモリ上のデータが無効になっているときは、他のプロセッサによるそのアドレスへのアクセスには、メモリではなく有効なデータを持つキャッシュが対応しなければならない。よって、そのキャッシュが MIH をアサートしてメモリがデータを転送するのを停止し、自分の持つ有効なデータを要求元のプロセッサに転送する。

また、あるプロセッサが有効なメモリを読み込む要求を出すと、そのときにすでにそのデータを共有しているキャッシュが MSH をアサートし、このプロセッサが読み込んだデータは共有状態となる。よって、このプロセッサは自分のキャッシュが他に共有されていることを知り、書き込みがあったときに他のキャッシュに対して無効化の信号を送る。

5. ま と め

本稿では、並列処理を実行させるための新しいシステムとしてのコンピュータ・コロニーとそのハードウェア構成について述べた。プロセッサ資源を最大限に利用し並列処理を行うマルチユーザ環境を低価格で実現することは、並列アプリケーションの増加に伴って、将来の計算機科学が避けて通れない道であると考える。キャッシュコヒーレンシ制御信号を持つマルチプロセッサ対応のシステムバスに通信ハードウェアを接続することで、完全なキャッシングを実現し、これまでにない高速な共有メモリアクセスを行うことができる。

今後は、コンピュータ・コロニーにおける並列処理の実行単位の仕様や、Fibre Channel を用いた通信プロトコルの設計および実際のハードウェア開発、さらに Colonia の実装などを進めていく。

謝　辞

メンター・グラフィックス・ジャパン（株）には、Higher Education Program の一環として製品とサービスをご提供いただきました。ここに感謝いたします。

なお本研究の一部は、文部省科学研究費補助金（基盤研究 (C) 課題番号 09680334 ならびに 奨励研究 (A) 課題番号 09780268）による。

参 考 文 献

1) A. S. タネンバウム (引地信行, 引地美恵子 訳): OS の基礎と応用 — 設計から実装, DOS から分散 OS Amoeba まで, トッパン (1995)
2) J. ベーコン (藤田昭平, 篠田陽一, 今泉貴史 訳): 並行分散システム — オペレーティングシステム, データベース, 分散マルチメディアシステムへの統合的アプローチ, トッパン (1996)
3) 前川 守, 所 真理雄, 清水 謙太朗: 分散オペレーティングシステム — UNIX の次にくるもの, 共立出版 (1991)
4) 萩原 宏, 津田 孝夫, 大久保 英嗣: 現代オペレーティングシステムの基礎, オーム社 (1988)
5) 前川 守: オペレーティングシステム, 岩波講座ソフトウェア科学 6, 岩波書店 (1988)
6) SPARC International, INC. (相越克久, 田中長光 訳 / 多田好克監訳): SPARC アーキテクチャ・マニュアル バージョン 8, トッパン (1992)
7) 日本サン・マイクロシステムズ株式会社: SPARCstation20 システム・アーキテクチャ, Sun Expert, No. 16, Sun Expert 16(1994)
8) SPARC Technology Business: SuperSPARC & MultiCache Controller User's Manual, Sun Microsystems(1996)