Vol. 36 No. 6　　情　報　処　理　　June 1995

解　説

重点領域研究：超並列原理に基づく情報処理基本体系

4. 超並列処理用オペレーティングシステムCOSとその設計†

齊　藤　信　男††

1. はじめに

「超並列」の重点領域の研究では，ハードウェアの設計から始まり，それを稼働するための基本ソフトウェアとしての新しいオペレーティングシステムの研究開発が課題の一つとして行われた．オペレーティングシステムは，計算機資源の管理をすることがその役割であるといえるが，さらに詳細を考えれば，与えられた計算機資源の効率良い利用環境を提供すること，計算機資源を抽象化してより使いやすいオブジェクトとして提供すること，計算機システムの安全性，堅牢性のような全体的な特質を支援すること等がその役割や目的であるといえる．これは，一般的にどんな計算機システムにも適用できることであるが，本研究では特に超並列処理を実現する計算機システムを対象としており，その特性を考慮しながらオペレーティングシステムの設計や開発を進めてきた．

超並列計算機システムは，商用のものもいくつか開発されているが，そのオペレーティングシステムはUNIXを基本としたものが多い．これは，超並列システムが主に研究機関で利用されていたことと無関係ではないであろう．既存のオペレーティングシステムを使わず，新しいものを研究開発するプロジェクトは，あまり見られない．しかし，オペレーティングシステムを応用ソフトウェアのプラットフォームと見なしてその応用プログラムインタフェースAPI（Application Program Interface）を抽象化した並列マシンとして設定する試みは行われており，PVM（Parallel Virtual Machine）[1]やMPI（Message Passing Interface）[2]はその代表例である．このようなインタフェースが共通化してくれば，超並列の応用ソフトウェアの移植性も向上する．

新しい超並列用オペレーティングシステムは，UNIXのような既存のオペレーティングシステムでは提供できないような使いやすいユーザサービスの追究，効率の良い超並列用資源の利用，超並列用プログラム言語の共通インタフェース等を実現できる可能性がある．新しいオペレーティングシステムの設計を柔軟に行えば，このような利用環境の実証実験をいくつか行うこともできる．このような見地から，超並列用オペレーティングシステムの新しい開発を重点領域の一つの課題として設定し取り組んできた．ここで開発したオペレーティングシステムは，COS（Comprehensive Operating System）という[3]．

2. 超並列オペレーティングシステムCOSの設計方針

COSの設計方針は以下の通りである．

（1） 多様なエンドユーザの要求に対応

（2） マルチユーザの実行環境を提供

（3） 独立かつ完全に機能するオペレーティングシステムであること

（4） ハードウェアに依存しないオペレーティングシステムであること

（5） 保護機能と実行性能との程良い均衡をとること

（6） 柔軟な実現法たとえばマイクロカーネル方式を採用

（7） ポリシー/メカニズムの分離

（8） 分散共有メモリの実現

（9） 強力に保護した資源の分割機能

以上の設計方針の中で，(1) から (5) はユーザ要求から，また，(6) から (9) はオペレーティングシステムの技術的な興味から要請されるも

† Comprehensive Operating System for Highly Parallel Machine by Nobuo SAITO (School of Environmental Information, Keio University).

†† 慶應義塾大学環境情報学部

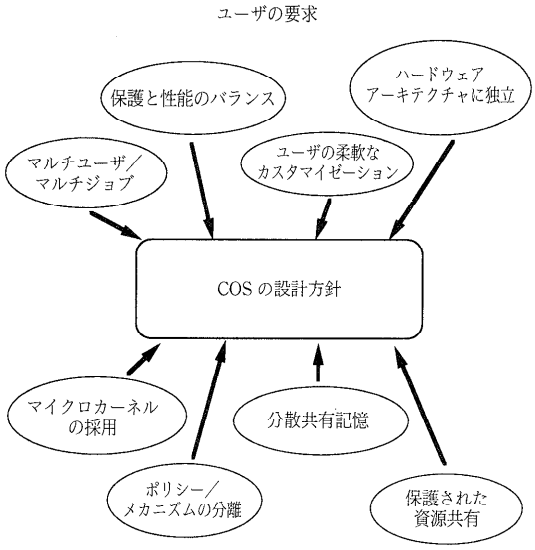


図-1　COS の設計方針

のである．この関係を図-1 に示す[4]．

3. 超並列用オペレーティングシステム COS のアーキテクチャ

3.1 ユーザに対するサービスインタフェースの構造

設計方針にも述べたように，超並列システムは様々な異った要求を持ったエンドユーザの応用ソフトウェアを処理しなければならない．これらは，ハードウェア資源あるいはソフトウェア資源に対して，異った利用要求を持っている．オペレーティングシステムをユーザの立場で見たときには，どんなサービスをシステムが提供するかが最も重要な問題になる．このサービスインタフェースを使って，応用プログラムを記述するので，これは応用プログラムインタフェース API（Application Program Interface）となる．

COS では，サービスを提供する対象ユーザを以下の 3 種類に分類し，それぞれのサービスクラスが提供しやすいように基本アーキテクチャを設計している[5]．

● COS におけるユーザのクラス

（1）　お任せ型ユーザ

（2）　協調型ユーザ

（3）　独立自尊型ユーザ

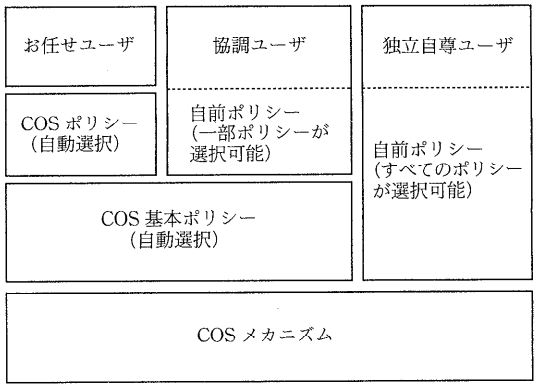


図-2　COS のユーザクラスとポリシー

従来，オペレーティングシステムの設計方法として，ポリシーとメカニズムの分離という原理があった．ここでは，上記のユーザのクラスの動作に関して，この原理で説明する．COS の基底には，いわゆるメカニズム のモジュールが共通にある．これに対して，ポリシーのモジュールは，ユーザのクラスによって，以下のように選択する．この関係を，図-2 に示す．

（1）　お任せ型ユーザ

すべてのポリシー モジュールは，既定のものを使う．したがって，ユーザに対するサービスも既定のものになる．

（2）　協調型ユーザ

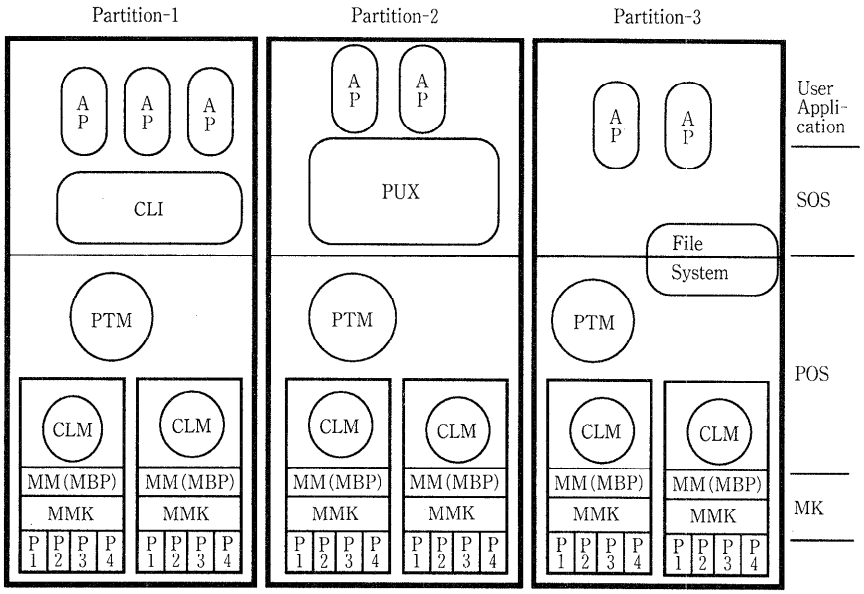
COS の基本的なポリシー モジュールは，既定のものを使い，それ以外のポリシーモジュールは自分自身で提供するものを使う．したがって，ある程度ユーザの要求にカスタマイズしたサービスを受けることができるが，全体的な協調性の中にユーザは存在している．

（3）　独立自尊型ユーザ

COS の基本的なポリシーモジュールも含めて，すべてのポリシーモジュールをユーザが提供する．したがって，全体的な協調性から見れば，他のユーザに割り当てられた資源を破壊しないという最低の条件以外には制約条件はない．完全にユーザの要求にカスタマイズしたサービスを受けることができる．

3.2 COS のソフトウェアアーキテクチャ

前節に示したようなサービスを各クラスのユーザに提供するのが，COS の役割である．これを能率良く実現するようにシステム設計したのが，



P 1 : Processing Element 1, MMK : Meta Micro Kernel, MK : Micro Kernel MM(MBP) : MBP Based Memory Management, CLM : Clustcr Manager, PTM : Partition Manager, PUX : Parallel Unix, CLI : Common Language Interface, POS : Partition OS, SOS : Service OS

COS Architecture

図-3 COSのソフトウェアアーキテクチャ

以下に述べるCOSソフトウェアアーキテクチャである[6]．これは，超並列システムの資源管理を実施する部分，およびユーザの応用ソフトウェアに有益なサービスを与える部分とに分けられる．

COSのソフトウェアアーキテクチャとそれを構成する要素は，以下のとおりである．

（1） POS（Partition OS）

COSにおける資源管理を直接実施する制御システムの部分で，ハードウェア資源の集合の基本単位であるパーティションの管理を行うことからその機能を決定できる．パーティションの定義，割当て，パーティション外からの不正なアクセスへの保護，パーティション間の情報交換の制御，補助記憶装置上のボリュームの管理，入出力操作の管理等を行う．

実際には，POSはさらに以下の構成要素からなる．

- MK（Micro Kernel）とMMK（Meta Micro Kernel）
- CLM（Cluster Manager）
- PTM（Partition Manager）

（2） SOS（Service OS）

COSにおける具体的なシステム機能のサービスを実現する部分，パーティション割当ての要求，プログラムの起動や停止，高度な通信，入出力操作起動，ファイルシステムの実現，ユーザインタフェース等を提供する．COS上では，複数種類のSOSが稼働している．

SOSとして提供するものには，以下のようなものがある．

- PUX（Parallel Unix）
- ファイルシステム
- CLI（Common Language Interface）

このソフトウェアアーキテクチャを，**図-3**に示す．

4. COSの構成要素とその設計

ここでは，COSの構成要素に関して，その設計と実装について述べる．

4.1 MK（Micro Kernel）とMMK（Meta Micro Kernel）

これは，COSの基本となる最小のカーネルで，

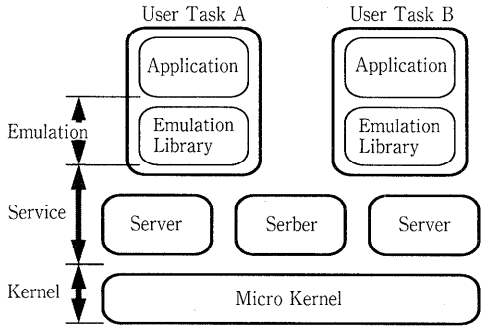


図-4 マイクロカーネルとサーバ

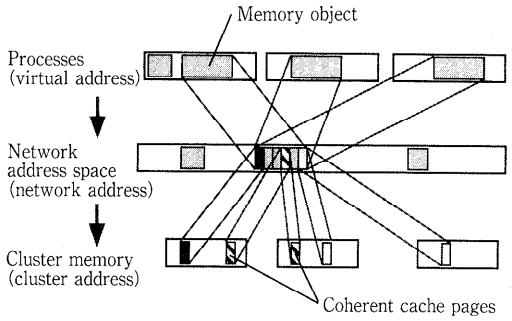


図-5 COS記憶管理のアドレス

CMUで開発したMach 3.0[7]を基礎としてそれを超並列用に拡張した.

COSは,いくつかの階層から構成されるがMMKはその最下層に位置し,ハードウェアの基本機能の制御を行う.MKは,その上の層を構成するサーバ群の実行環境を提供するものであり,必要最小限の機能を持つものである.基本的にはMMKに記憶管理を付加して拡張したものがMKである.各ユーザ環境で実現すべき資源管理に関するポリシーは,上位層のサーバによって与えられるべきで,ポリシーモジュールを実現するためのメカニズムのみを機能として提供している.

MKは各々のPE上で動作し,(1)タスク管理/スケジューリング,(2)プロセス間通信(IPC),(3)仮想記憶管理(VM),(4)デバイス制御/割込み制御,(5)カーネルデバッガといった機能を提供するためのメカニズムを持っている.

ここで基礎としたMach 3.0マイクロカーネルは,ソースコードが公開されているソフトウェアであるとともに,各種のワークステーションや並列マシンに移植され,商用にも利用されている.また,UNIXサーバ,VMSサーバ,MS-DOSサーバ,Macintoshサーバ等のオペレーティングシステムの機能をユーザレベルサーバとして提供する実験がすでに行われており,ピュアカーネルとしての実績が多くあり,この上に,オペレーティングシステムのサービス機能を提供するユーザレベルサーバを動かすことができる(**図-4**参照).

Mach 3.0をCOSのMKとして使用するためには,少なくとも(1)ファーストクラスユーザレベルスレッドのサポート,(2)高速なIPCのサポート,(3)クラスタ間通信,(4)MBPの機能のシミュレーション,の拡張をする必要がある.

4.2 記憶管理

JUMP-1は,並列処理の粒度を柔軟に制御しかつ効率良く実行ができるような機構として,分散共有記憶を支援するMBP(Memory Based Processor)と呼ぶ専用の機構を備えている.これを使って,より効率の良い超並列システムの記憶管理を実現することはCOSの一つの重要な機能である.この記憶管理は,MKの一部としてPE上で作動するプログラムとMBP上で作動するMBPプログラムの両者から構成される.

COSは,ユーザに対して抽象化したメモリオブジェクトを提供する.これは,連続する仮想記憶ページからなる.メモリオブジェクトにアクセスするには,プロセスの仮想アドレス空間にその一部または全部をマップする.通常のメモリアクセスによりメモリオブジェクトがアクセスできる[8].

COSの記憶管理では,**図-5**に示すようにプロセスの仮想アドレス空間を各クラスタごとの物理メモリのアドレスであるクラスタアドレスに高速ネットワークを使って対応付けする.JUMP-1では,この対応付けのためにシステム全体で一意なネットアドレスと呼ぶ中間形式を導入している.

また,MBPはハードウェア支援機能として通信,同期機構を提供している.すなわち,COSのメモリオブジェクトは,任意の位置に同期用アドレスを持つことができ,barrier, semaphore, FIFO, signal等の抽象化した同期機構を実現できる.

このようなシステムに一意なネットアドレスと同期機構を持った分散共有メモリを使えば，ユーザの活動全体の保護を記憶保護のみによって統一的に行うことができる．COSはマルチユーザ，マルチタスクのオペレーティングシステムであり，このような一様で扱いの容易な保護機構は重要な役割を果たす．

4.3 クラスタおよびパーティション管理

クラスタは，ハードウェアの分割単位であり，その管理システムであるCLM（Cluster Manager）は，クラスタに一つずつ常駐し，クラスタ内の資源を管理する．その機能としては，クラスタ内のアドレス管理，クラスタ内のPE（Processor Element）の管理，クラスタ間通信，スケジューラの管理，入出力制御等がある．

パーティションはCOSでの論理的空間分割の単位であり，その管理システムであるPTM（Partition Manager）は，各パーティションに一つ存在し，パーティション内の資源の管理をする．具体的には，パーティション間通信，パーティションの生成，消滅，システム内で一意なネットアドレスとクラスタアドレスとの対応の管理等の機能を提供する．パーティションの生成/消滅に関しては，確定的なアルゴリズムはないが，最上位のスーパPTMを設置する方式，いくつかのグループに分散しPTMを設置する方式，ネットワークのブロードキャスト機能を使って分散的にフリーのクラスタを回収する方式等が考えられる．

4.4 ファイル管理

JUMP-1のファイル装置は，Taxiチップを基本にしたSTAFFリンクと呼ぶコミュニケーションラインを介して接続するディスクアレイを含んだディスク制御装置からなる．1台のディスク制御装置に対して，いくつかのクラスタからSTAFFリンクが接続されている．

このような入出力システムに対するファイル管理は，ドライバ層，仮想ハードウェア層，ディスクキャッシュとページングの管理層，メモリオブジェクト層およびファイルシステム層からなる．仮想ハードウェア層では，1台の超並列ディスクアレイが超並列マシーンに接続されていると見なし，各計算ノードは超並列ディスクアレイと直接的に入出力リンクで結合されるようにする．JUMP-1では，すべてのクラスタが入出力リンクで接続されているわけではないので，COSでMBPを介してその接続を管理する．

ファイルシステム層は，永続的なメモリオブジェクトと名前空間からなる．この永続的なメモリオブジェクトは，プロセスのアドレス空間にマップして利用する．ファイルの領域は，32 KBのディスクブロックを単位に管理する．

4.5 ユーザサービス機能

ユーザとのインタフェースをとり，超並列の応用プログラムを起動したりその後始末をしたりするために，一般的なオペレーティングシステムのユーザインタフェースの最小版が必要になる．COSでは，そのようなインタフェースのサーバとしてPUXを提供する．これは，通常のUNIXサーバを基本とし，超並列マシーンに対するマルチタスク，マルチユーザの実行環境，複数スレッドの利用，並列パイプ，超並列入出力等の拡張機能を提供している．

また，超並列の応用を記述するプログラム言語は，今回のプロジェクトでも開発されているが，その実行環境を実現するための共通インタフェースとしてCLIを提供している．このプロジェクトで考える様々な並列プログラム言語や，それによって記述された実装のターゲットとされる並列言語は，以下のような分類ができると考えられる．

（1） FORTRANやC等の従来の逐次型プログラム言語上にMPI等の標準並列ライブラリを付加したもの

（2） 従来のFORTRAN等の逐次型プログラム言語に対する自動並列化コンパイラ

（3） High Performance Fortran（HPF）やNCX等のSIMD/SPMDのデータパラレル型言語

（4） Valid/V，ABCL/f，KLIC等のMIMD型あるいはオブジェクトパラレル型言語

COSは，異なるハードウェアプラットフォーム上に実現されるが，ハードウェアの抽象化・隠蔽化を促進して可搬性を高めるため，細粒度並列言語の実装を行うためのランタイムのメカニズムやライブラリ群をCLIは提供する．

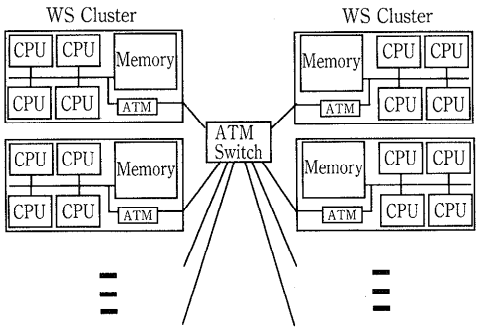


図-6 開発用テストベッド

5. COS開発の支援機能

COSは，マイクロカーネルから始めて統合的なオペレーティングシステムを目指しており，その開発のためには，いくつかの開発環境やツールが必要である．ここでは，今回の開発で利用したツール等の概略を述べる．

5.1 エミュレータの開発

COSのマイクロカーネルを既存のOS上で実行するために，エミュレータを開発し，マイクロカーネルを直接SUNOS上でユーザプロセスとして実行できるようにした[9]．また，MBP機能の開発を行うためにそれが持っている記憶制御のいくつかの機能を直接エミュレートするエミュレータを作成した．

また，JUMP-1のハードウェアをエミュレートさせ，その上に簡単なCOSマイクロカーネルの機能も利用できるようなシステムを作成し，記憶管理等のデバッギングに利用した．

また，パーティションの生成/消滅アルゴリズムの実験や検証を行うために，簡単なJUMP-1のエミュレータを作成した．これは，PEはスレッドに，通信用ネットワークはソケットに，クラスタはプロセスに対応させてエミュレートしている．

5.2 開発用テストベッドの整備

COSの開発用のテストベッドとして，図-6に示すような構成のシステムを考慮した．これは，JUMP-1のPEであるSparcチップからなるSparc 20のマルチプロセッサを高速ネットワークであるATM-LANで接続したワークステーションクラスタである．これにより，MMK, CLM, PTM, PUX等の開発ができる．MBP関連の開発は，エミュレータを使用しなければならないが，そのエミュレータの一部モジュールを利用してこのようなワークステーションクラスタに対する超分散環境での分散共有記憶の実装も可能と考えられる．

6. おわりに

JUMP-1に対する汎用超並列用オペレーティングシステムとしてのCOSの設計と開発について紹介した．これは，アーキテクチャに独立になるように設計してあるが，将来大規模なワークステーションクラスタ等に対しても適用できる可能性を持っている．

謝辞 本稿を記述するにあたり，科研費重点領域超並列のC班「超並列処理体系・制御系に関する研究班」のメンバである徳田英幸，多田好克，柴山悦哉，米澤明憲，益田隆司，萩野達也，松岡聡，砂原秀樹，篠田陽一，猪原茂和，斎藤靖，追川修一，石井秀治，村山正之，荒川和進，斎藤鉄也，斎藤正伸，中村嘉志，曽和将容の諸氏に感謝の意を捧げる．

参考文献

1) PVM Reference Manual.
2) MPI Reference Manual.
3) Saito, N. et al.: Comprehensive Operating System for Highly Parallel Machine, Proceedings of ISPAN '94, IEEE Computer Society, pp. 435-442 (1994).
4) 文部省重点領域研究：超並列原理に基づく情報処理基本体系，第一回シンポジウム予稿集 (1992).
5) 文部省重点領域研究：超並列原理に基づく情報処理基本体系，第五回シンポジウム予稿集 (1994).
6) 萩野達也他：超並列コンピュータのためのオペレーティングシステムの構想，情報処理学会オペレーティングシステム研究会 (1993).
7) Accetta, M. et al.: Mach: A New Kernel Foundation for Unix Development, USENIX Summer Conference (1986).
8) 猪原茂和他：分散共有記憶型超並列オペレーティングシステムCOSマイクロカーネルの保護機構，並列処理シンポジウム(JSPP '94) (1994).
9) 村山他：協調的オペレーティングシステム，COS開発環境，並列処理シンポジウム (JSPP '94) (1994).

(平成7年4月5日受付)

齊藤 信男 (正会員)

1940年生．1964年東京大学工学部計数工学科卒業．1966年同大学院応用物理学専攻修士課程修了．1966年電気試験所(現電子技術総合研究所)入所．1974年筑波大学電子情報工学系専任講師．1978年慶應義塾大学工学部数理工学科助教授．1990年同大学環境情報学部教授．その間，スタンフォード大学，カーネギメロン大学客員研究員．オペレーティングシステム，並列/分散処理，ソフトウェア工学，ネットワーク，マルチメディアなどの研究に従事．著書「ユーザーズUNIX」など．電子情報通信学会，計測自動制御学会各会員．