Distributed Operating Systems & Algorishms (by Chow, Johnson)

久野　靖*

2000.9.11

はじめに

□ 今学期のゼミ — Randy Chow, Theodore Johnson: Distributed Operating Systems & Algorithms, Addison-Wesley, 1997. (ISBN 0-201-49838-3)

□ 内容 — 分散システム/分散 OS とそこに出て来るさまざまな手法やアルゴリズムについて

- 原理も実際も等しくカバー
- 前半 (〜8 章) → OS のテキスト、後半 (9 章〜) →分散 OS のテキスト

□ 本書の構成

- 1〜2 章→基本概念
- 3〜5 章→分散プロセス (同期、通信、スケジューリング)
- 6〜8 章→分散資源 (ファイルとメモリ)
- 9〜13 章→さまざまな分散アルゴリズム

1 Operating System Fundamentals

□ 現代の計算機システム→高度化→ネットワーク、複雑なアプリ、分散計算

□ 本書ではそのようなシステムの基本技術 (特に分散 OS/分散アルゴリズム) を学ぶ

□ 「分散システム」と「分散 OS」は違う意味。さらに様々なハードウェアアーキテクチャの違い→本章ではこれらを整理

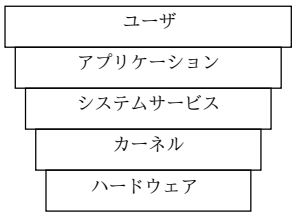
1.1 Evolution of Modern Operating Systems

□ OS →アプリケーションとハードウェアのインタフェースをとり、アプリケーションに「使いやすい環境」を提供

- プロセスの多重化、資源の管理、アクセス制御、エラー回復、ユザインタフェース

□ OS の構造→カーネルとシステムサービスに分けられる

- カーネル→ハードウェアと接する部分
- システムサービス→アプリケーションにサービスを提供する
- ユーザから見ると OS は「仮想マシン」を提供する←主たる目的
- 管理者から見ると OS は「資源管理機構」に見える←目的のために必要

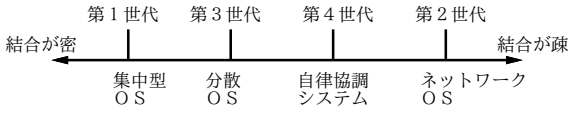


□ 集中型 (centtralized)OS →単一システム (1CPU/マルチ CPU) 上で稼働→その設計方法などはよく知られている

□ WS、LAN の普及→

- ネットワーク OS、分散 OS の出現
- 自律協調システム (cooperative autonomous system) の概念←分散アプリ、オープンシステム

□ 結合の疎/密さに注目した比較



*筑波大学大学院経営システム科学専攻

- 結合が密/疎…プロセッサ間通信時間とプロセッサ内通信時間の比が小/大
- 1 つの OS で全体を管理→密、複数 OS の協調→疎
- 集中型 OS →ハード、ソフトとも密結合
- 分散 OS(名前としてはよくない) →疎結合のハード上 1 つの OS に見える
- ネットワーク OS →別々のシステムがネット経由で通信
- 自律協調→分散 OS から「単一制御」を取り払ったもの

□ 何を主要な機能/目標としているかという面から見ると分かりやすい。

分類	特性	目標
集中型 OS	プロセス管理、メモリ管理、IO 管理、ファイル管理	資源管理、仮想マシンの提供 (仮想化)
ネットワーク OS	遠隔アクセス、情報交換、ネットワークブラウジング	資源共有 (相互運用性)
分散 OS	(ファイルシステム、ネームスペース、時刻、計算資源の) 単一ビュー	単一システムとして見える (透明性)
自律協調システム	オープンかつ協調的な分散アプリケーション	協調作業 (自律性)

□ 旧世代の OS →仮想化。最近の OS →相互運用、透明、自律性を追加

□ ネットワーク OS →ネットワークでの通信→相互運用が目標

□ 透明…仮想化とよく似ている

- しかし違う! 何が違うか分かりますか?

□ 透明と仮想化の違い:

- 仮想化→利用者は「見たいものを見る」(例: 広いメモリ空間)
- 透明→利用者は「見たくないものを見ない」(例: マシンの境界)
- 両方が備わっているのがベスト「見たいものだけを見れば済む」

□ 透明性はあればあるだけ望ましいか?

- ユーザは「自分がやりたいことができるれば」満足→透明性は必須ではない
- ユーザはシステムが複数のマシンから成ることも、他のユーザがいることも知っている。

□ 透明性に代って開放性 (openness) があればよい→無名性 (autonomy)

- システムは多数のマシンから成る→自律性
- システムにとってもユーザは沢山いる→自律性

□ 現代の OS →これらすべての特性を統合

1.2 Centralized Operating System

□ 集中型 OS の主要な機能

要求	管理機能
シングルユーザ	UI、I/O 制御、割り込み、デバイスドライバ
効率的 I/O	仮想 I/O、スプーリング
大規模アプリ	仮想記憶、ページング、セグメンテーション
マルチユーザ	マルチプログラミング、TSS、スケジューリング、アクセス制御/保護、ファイル共有、並行制御

1.2.1 Operating System Structure

□ OS は大きなソフト→どのように構造化するかが重要

- 伝統的な分割方法→垂直分割、水平分割

□ 垂直分割→階層化→各レイヤはその 1 つ上/下とだけやりとりすればよい→システム構造が分かりやすくなる

□ 水平分割→モジュール化→機能ごとの分割など。現在ではオブジェクトを単位とするのがよい

□ さらに「ハードウェア依存部分の分離」という構造化も重要

□ 依存部分は「最小カーネル」として設計しシステムサービスから分離

- 最小カーネルだけ移植すればそのまま他のマシンへ移せる

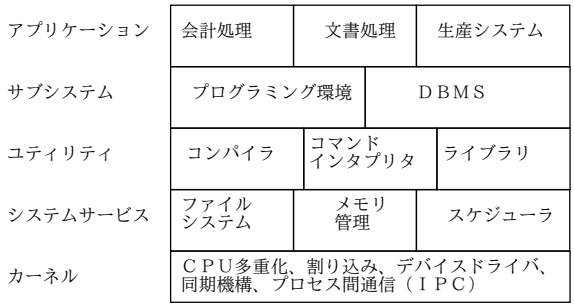
- 最小カーネルの典型的な機能→マルチプロセス、割り込み、デバイスドライバ、同期、IPC(Interprocess Communication)

□ カーネルの構造自体は単一 (Monolithic) カーネルが多い←効率がよい、カーネルの機能自体はなるべく小さくしてある (ホント?)

□ さらに C/S(client/server) 構造にするというのも良い方法

- システムサービスを呼ぶのがサーバプロセスへの通信と同等 (メッセージパッシング)
- システムの機能をサーバとして外に出しやすい
- 分散 OS のモデルと整合する

□ どこまでがシステムでどこからがアプリケーションかは曖昧

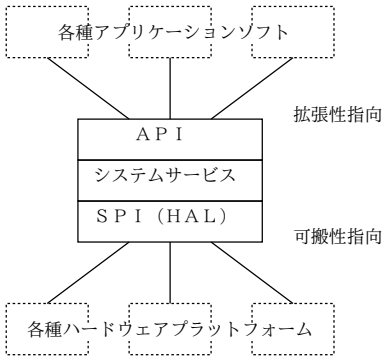


- コンパイラはシステム? アプリ?
- DBMS はシステム? アプリ?
- ユーザにとっては自分が使うサブシステム以下は全部システム

1.2.2 Subsystem and Microkernel

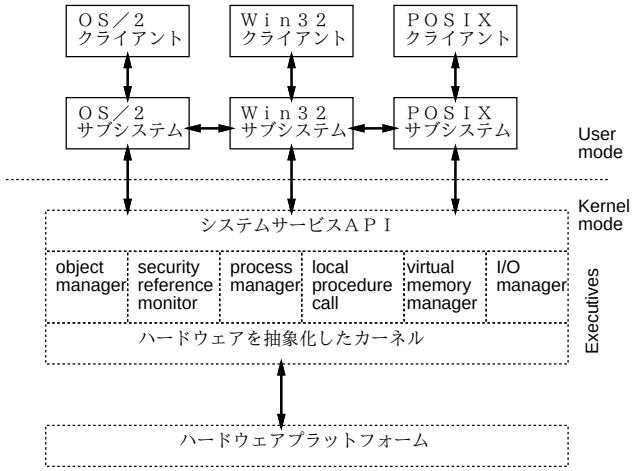
□ OS のサービスが構築可能な最小限の汎用カーネル→マイクロカーネル

- API(application programming interface) → カーネルが提供するサービスのインタフェース
- HAL(hardware abstraciton layer) ないし SPI(service provider interface) → プラットフォーム依存性を隠蔽する
- システムサービス→各種の OS 機能を実現するのに十分な機能群



□ 代表的なマイクロカーネル…Mach、OS/2、WindowsNT

□ マイクロカーネルの上で動くサブシステム→各種の OS 環境を同時に提供可能



1.2.3 Management Functions

□ OS は資源管理機構であると見ることもできる。資源→プロセサ/プロセス、メモリ、I/O、データ/ファイル

- プロセス→複数のプログラム実行を (時間) 多重化+同期、IPC
- メモリ→複数プロセスの記憶領域を (空間) 多重化

□ 同期機構…さまざま。セマフォ、CCR(conditional critical region)、モニタ、CSP、ランデブ、パス式、… 3 章でも出て来る

□ プロセス間通信→共有メモリと割り込み/メッセージパッシング

- 同期とプロセス間通信は密接に関連。メッセージを同期に使える

□ プロセススケジューリング→ターンアラウンド、スループットに影響

- 2 種類のスケジューリング: 静的スケジューリング (プライオリティベース)、動的スケジューリング (負荷分散等)

□ I/O デバイスの管理← OS の重要な仕事の 1 つ

- 難しさ←本質的にハードウェア依存←デバイスドライバで吸収
- 抽象化した I/O →「記憶装置」(読む/書く機能のみを提供)
- スプーリング、バッファリングなども提供 (spool → simaltaneous peripheral operation on-line)

□ 代表的な/重要なデバイス→ハードディスク、ビットマップ端末

□ デバイスの獲得/開放→デッドロックの心配

□ メモリ管理→主記憶の管理、仮想記憶機能の提供

- ページ方式 (固定長の単位)、セグメント方式 (論理的な単位)、両方併用
- 仮想記憶→遅いディスクと速い主記憶を組み合わせて「速くて大きな記憶領域」を実現←空間的/時間的局所性の利用

□ 共有メモリは古くから使われている共有手段。分散システム上でこれを利用可能にするもの→分散共有メモリ

□ ファイルシステム→「論理的な情報の単位」を提供→ディスク、メモリ、その他の I/O の上に構築可能

- ファイルの機能→読み書き、共有、アクセス制御、並行制御
- 分散システムではしばしばファイルも分散化されている→複数の複製の維持等、複雑な制御が必要

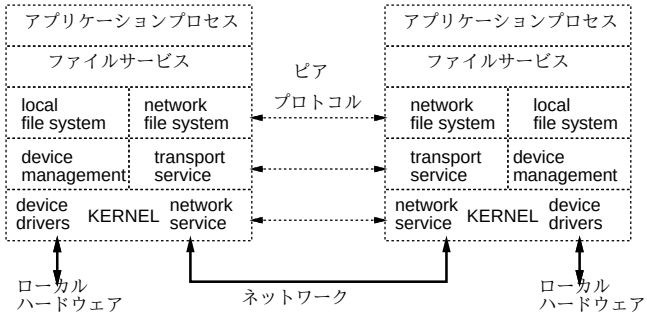
□ OS の 4 つの機能 (プロセス、I/O、メモリ、ファイル) のうちで、分散システムで問題になるのはプロセスとファイル。あとの 2 つはローカルな管理なので以下では扱わない。

1.3 Network Operating Systems

□ LAN で接続された WS →資源 (データやプログラム) を共有したい→ネットワーク OS。相互運用性が目標

□ ネットワーク機能→階層構造に構成

- 通信サブシステム→トランスポートサービス→ピア間 (アプリケーションレベル) 通信



□ OS の機能の一部はトランスポートサービスの上に実装 (例:NFS)

□ アプリケーションからのネットワークの利用形態→ソケット等の API

□ 各種のネットワークアプリケーションを含む→遠隔ログイン (仮想端末)、ファイル転送、電子メール/EDI、WWW、遠隔実行

- 遠隔実行は強力だが危険性もある。遠隔実行で相手先ホストが同じ機種でない場合→スクリプトを解釈実行すれば大丈夫→ Java VM など

1.4 Distributed Operating Systems

□ 多数の WS が高速な LAN でつながっている→それらの間での資源共有をもっと行いたい→個々のシステムを区別せず自律システムとして扱う

□ 各種の資源が分散→それらを管理する機能も分散→分散アルゴリズムの必要性

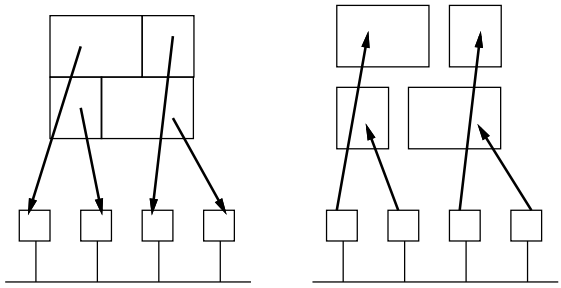
□ 分散 OS の重要な目標→透明性 (位置透明、並行透明)

- これらを実現するのにも分散アルゴリズムが必要

1.5 Cooperative Autonomous Systems

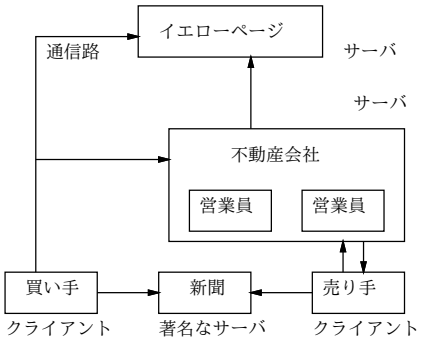
□ マルチユーザシステムを一人のユーザの視点ではなく全体としての視点から見る→各プロセスは自律的に動作し、互いに情報を交換し合いながら計算を進めて行く

□ 分散システムでは「1 つの OS」を複数のノードに分けて実現。自律協調システムでは複数ノードの協調動作が全体としてシステムを構成



分散システムへの分解　　協調システムへの統合

□ 例:　不動産の取り引き



- クライアントは各種のサーバに以来して目的を達成する
- 必要な情報を発見するには「著名なサーバ」を介する
- サーバのなかにはブローカとして働くものもある

□ ネットワークの巨大化→全体を 1 つとして動かすのではなく、個々の要素が自律的に協調し合いながら作業を進めて行く方があっている

□ CORBA などのミドルウェア、ORB(ブローカ、トレーダ)がサービスを仲介

1.6 Distributed Algorithms

□ 分散システムの実現→分散アルゴリズムが必要→何が難しいか?

□ 広域的に共有された情報がない、通信には必ず遅延

- 特に広域クロックがない→近似的な時刻の共有、イベントの順序を保つのにも手間が必要
- これらの時刻や順序づけの上に立ってさまざまな機能を実現

□ 故障からの回復/信頼できない通信への対処も重要な課題

- このため、集中型システムのアルゴリズムそのままでは駄目
- 2 つの方向→分散アルゴリズム、集中アルゴリズム+故障回復

□ ネットワークトポロジの影響、データの複製の管理なども

□ 分散アルゴリズムの特徴

- メッセージパシング→共有メモリがないのですべてメッセージで情報交換しなければならない
- 広域情報がない→合意プロトコルによって合意を形成する必要
- 重複化→データは複製されているかも→その整合性を管理
- 故障とその回復→冗長な情報を保持しておき、故障が起きた場合でも必要な情報を回復できるようにする

□ OS の進化…集中→ネットワーク→分散→自律協調

- 目標…仮想化→相互運用→透明→自律性

□ OS の各種技術

□ OS の教科書

□ 分散 OS の教科書

□ 分散アルゴリズムの教科書

□ その他…各種 OS、CORBA、WWW、Java など