オペレーティングシステム

2023/1/10

海谷 治彦

目次

- 性能
  - 1 性能の要素
  - 2 システムの性能
  - 3 資源の利用率とスループット
  - 4 スケジューリング，応答時間
  - 5 スループット
  - 6 オーバーヘッド
- OSの仕様
- 仮想化技術

- 若干のエピローグ

基本的な考え方

- システムの性能を決める要因
  - どんな処理(ジョブ)を行うのか？
  - どのくらいの量や頻度で行うのか？
  - どんなハードウェアで行うのか？
  - どのように処理群をスケジュールするのか？
- 処理の特徴や量
  - 単純な計算を逐次的に行う，いわゆるバッチ処理
  - 対話的な処理が複数同時に動作する.

誰が性能を計測する？

- システム全体の性能を横目に見ながらOSは資源割り当てやスケジュールを変える必要がある.
- その意味からは，性能計測はOSの役割ともいえる.
- しかし，今日では，専用の性能計測ツール(ベンチマークツール)で計測するのが一般的.

ハードウェアの能力 1/2

- プロセッサの性能
  - 単位時間辺りの命令実行数が一般的
    - MIPS Mega Instruction per second 秒あたり何メガ命令を実行できるかという指標
  - とはいえ，プロセッサによって個々の命令の能力が違うので，異なるシリーズ間では比較が困難.
    - 単純な四則演算しか命令として持たないCPUもあれば，命令一発で配列の計算とかできるCPUもある.
  - CPUのクロック数も用いられる.
  - レジスタの数と大きさ
  - バス幅 32bit 64bit
    - データとアドレスは別概念

ハードウェアの能力 2/2

- メモリ量
  - GB 等.
- 入出力性能
  - ディスクやネットワークの速度.
- ディスクの容量
  - GB, TB等

処理の特性

- 実行命令数
  - システムコール，数値計算，グラフィクスに特化した数え方もある
- 利用メモリ量
- 入出力回数
- 入出力量

- ベンチマーク (詳細は後述)
  - システムの性能を測定するための指標.

システムの性能

- 教科書での論調は処理(ジョブ)は明確な終わりがあることを仮定している.
  - 昼間の銀行窓口業務を夜間処理する.
  - 共通一次試験の結果を期日までに出す.
- しかし，昨今は対話型処理や常時稼働のものが多い.
  - ツイッターやライン等のクライアント通信ソフト
  - ウエブサーバーやプリンタサーバー等
  - オンラインゲーム
- とはいえ，前者にフォーカスした性能の議論をちょっとみていきます.
- リアルタイムシステムでは，非常に短い時間(10^{-5}s 等)の間に終わるという〆切厳守的な性能が重要.

スループット

- 単位時間あたりの処理数
- 最も直観的な「システムの速さ」
- 処理の種類によって当然スループットは変わる.
  - 具体的な処理例はベンチマークのところで.
  - 処理に明確な終わりがないとスループットは出しにくい.
- 処理によっては資源を100%まで使い切る場合あり，その資源の空き具合がスループットを左右する.
  - このような資源をボトルネック資源と呼ぶ.
  - 通信回線のバンド幅，データバスの転送速度等が現実的な例.

資源の利用率

- 順次使用資源
  - プロセッサ，入出力装置等
  - 時間帯を区切り，利用している時間の割合を計算する.
- 空間資源
  - メモリ，ディスク等
  - 全容量に対して，利用している部分の割合を計算する.

処理の時間

- 処理時間の分類1
  - CPU実行時間
  - 入出力時間
  - 待ち時間 (通常はマルチタスクなので)
- 処理時間の分類2
  - システム時間
    - カーネルモードでの実行時間，要はシステムコール実行時間.
  - ユーザー時間
    - ユーザーモードでの実行時間.
  - 待ち時間

timeコマンド

```
[kaiya@flute03 ACM]$ /usr/bin/time -v platex sig-alternate
This is e-pTeX, Version 3.1415926-p3.2-110825-2.3 (utf8.euc) (TeX Live 2012/dev)
 restricted \write18 enabled.
```

中略

```
(see the transcript file for additional information)
Output written on sig-alternate.dvi (5 pages, 29960 bytes).
Transcript written on sig-alternate.log.
        Command being timed: "platex sig-alternate"
        User time (seconds): 0.09
        System time (seconds): 0.01
        Percent of CPU this job got: 47%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.22
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 96992
        Average resident set size (kbytes): 0
        Major (requiring I/O) page faults: 0
        Minor (reclaiming a frame) page faults: 4158
        Voluntary context switches: 48
        Involuntary context switches: 47
        Swaps: 0
        File system inputs: 16
        File system outputs: 104
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
```

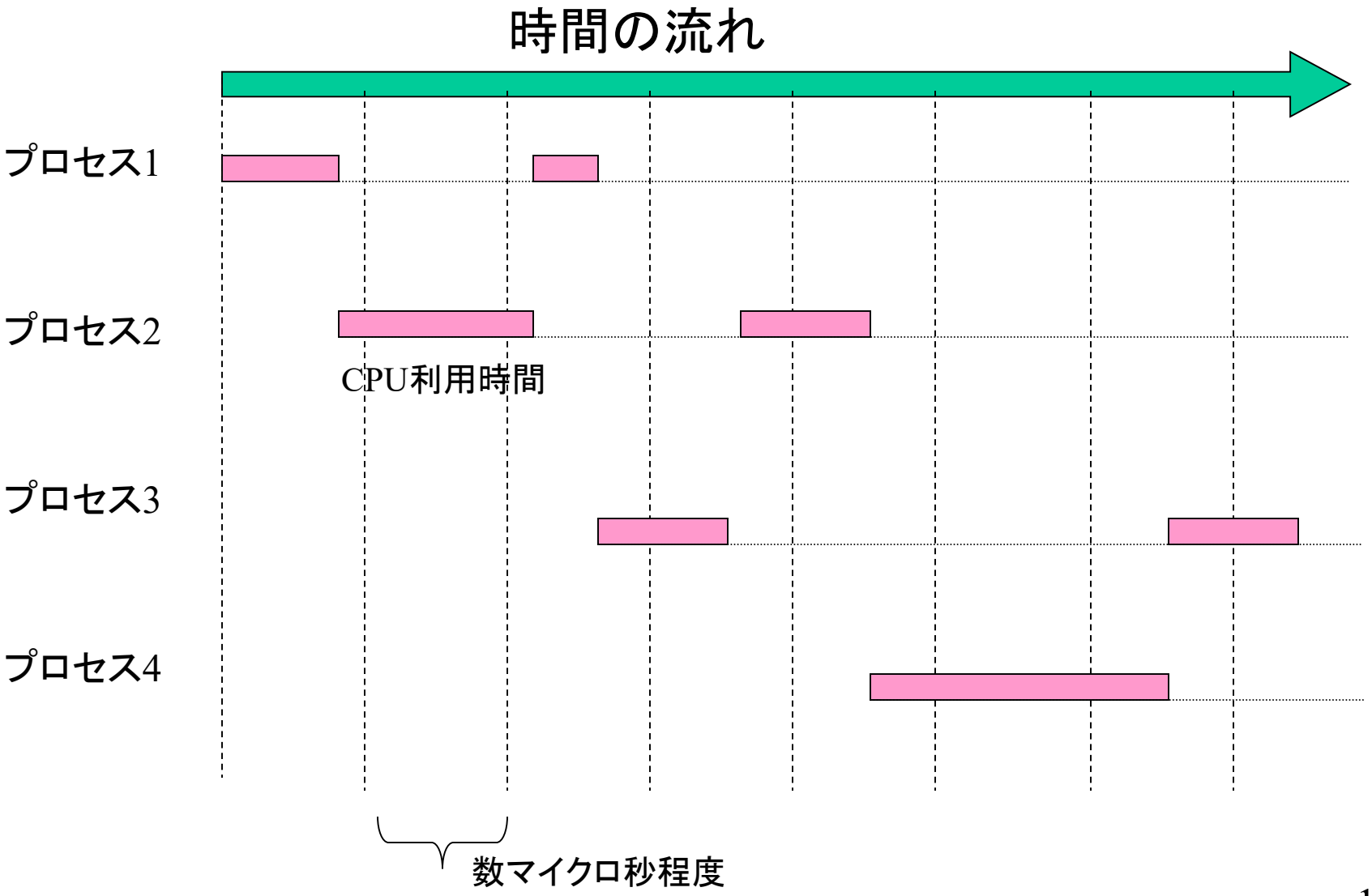
合計時間
0.1秒

経過時間
0.22秒

48+47回
プロセスは
停止している

復習

時分割の考え方 (図7.4改)

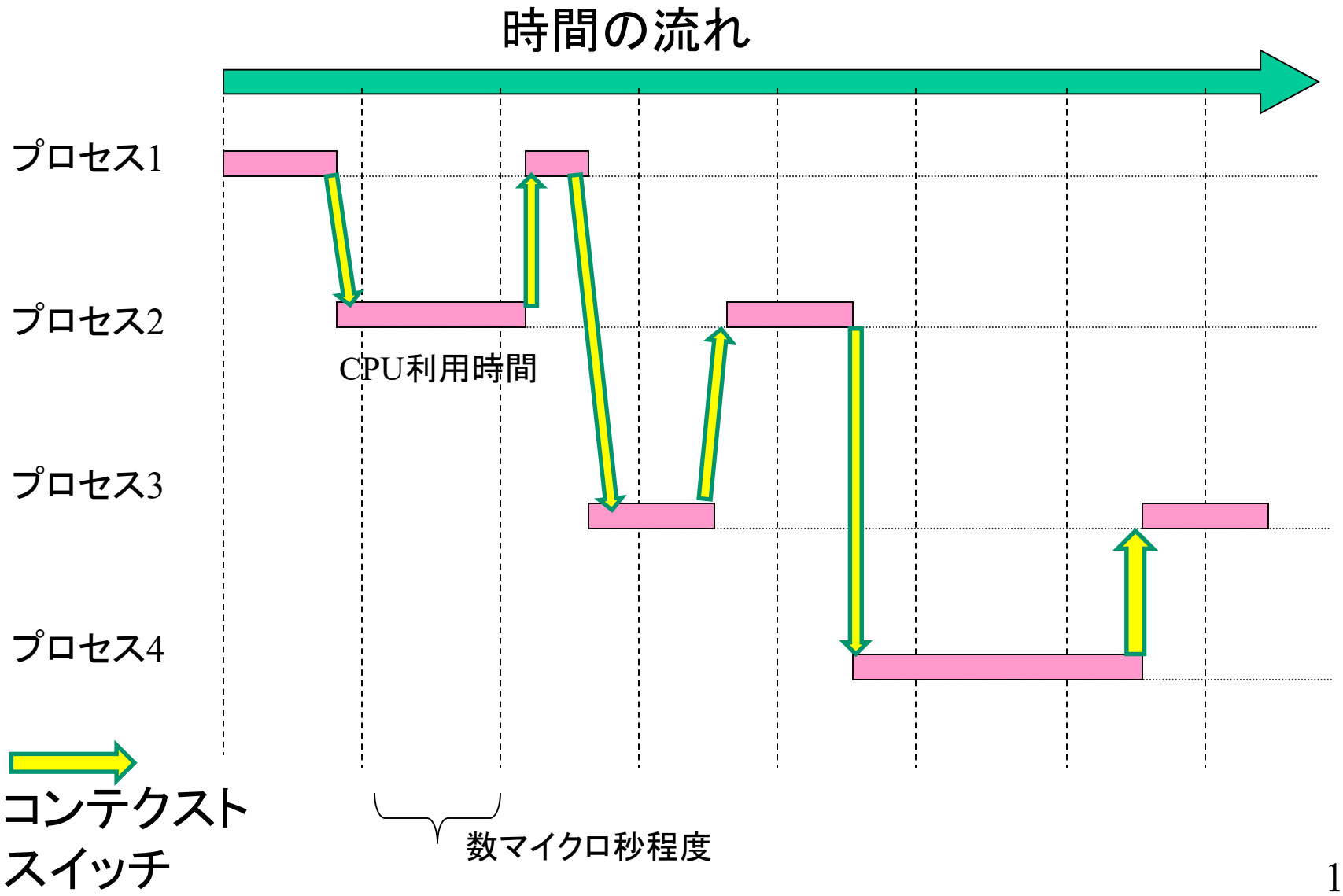


コンテクスト スイッチ

- Context Switch
- TSS等で，あるプロセスから他のプロセスへCPU利用権が移動する際，レジスタの内容等を更新すること.
- Context
  - あるプロセスが計算を行うためのCPUが持つ状態変数群.
  - 主にレジスタの内容.
- 本来，プロセスの章で話すべきでした

復習

時分割の考え方 (図7.4改)



端末の応答時間

- 一般人が直接ふれるプログラムのほとんどは対話型プログラムである.
- その意味で応答時間は性能を示す重要な指標である.
- とはいえ,「定番」といえる性能調査法やツールは無いようだ.

応答時間の簡易モデル

- 平均応答時間を予測するための計算式
- 入力としては以下の情報を使う
  - N: 対話するプロセス(人)の数
  - W: 対話中，対話者が考える時間の平均
  - S: 一回の入力を処理するための時間
  - R: サーバー利用率 (0～1) Nが増えれば1に近づく.

$$\text{平均応答時間} = \frac{S \times N}{R} - W$$

単一の対話者のモデル

- 以下のように，考えて，処理してを繰り返すと仮定する.

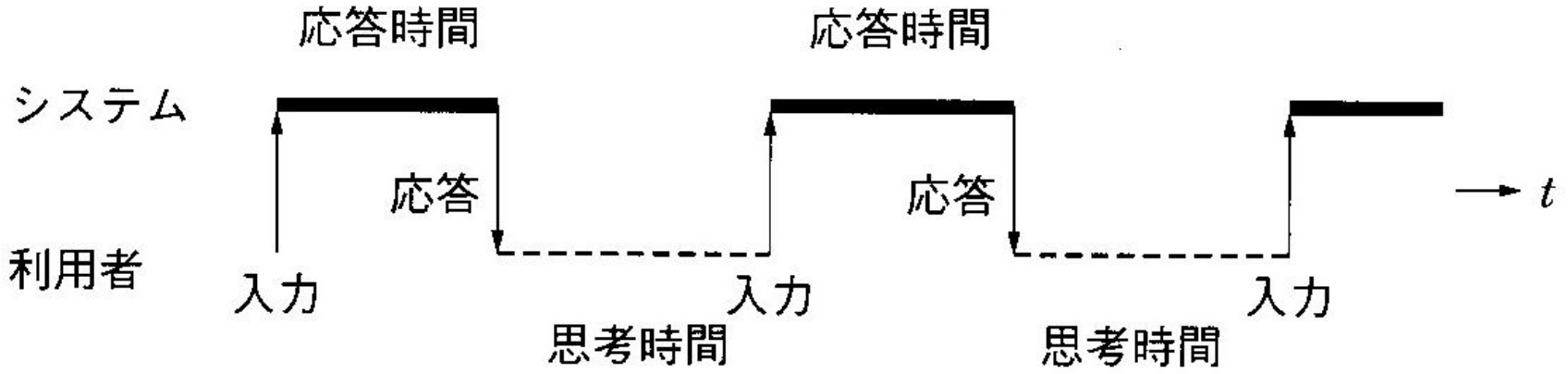


図 14.5　思考時間と応答時間

複数対話者の振る舞いモデル

- 以下のように思考が終わったらマシンに処理をしてもらう「待ち行列」に並ぶことを想定する.

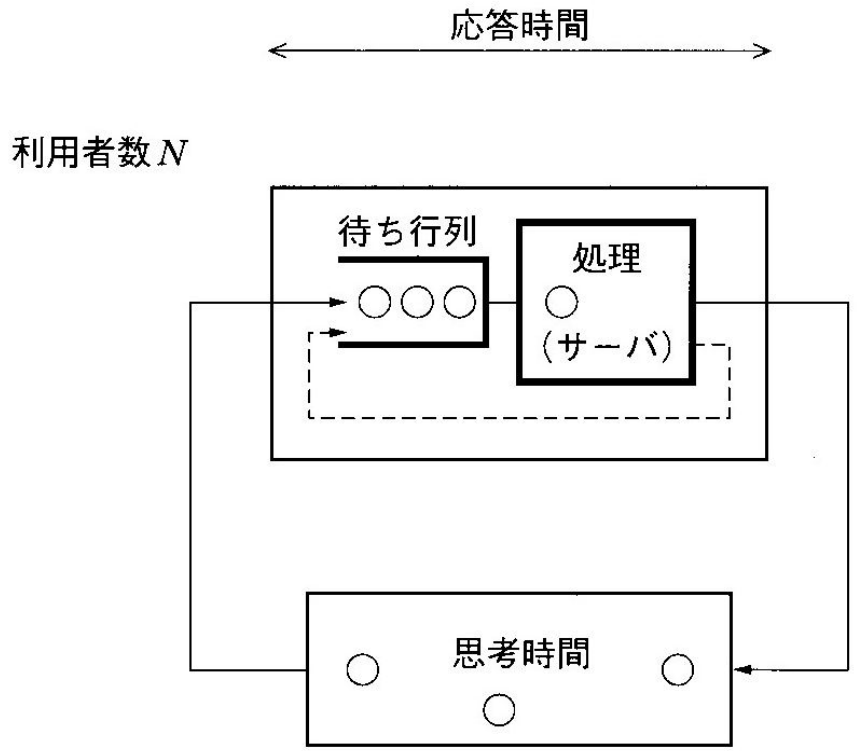


図 14.6　対話処理の有限待ち行列モデル

処理数と応答時間の関係

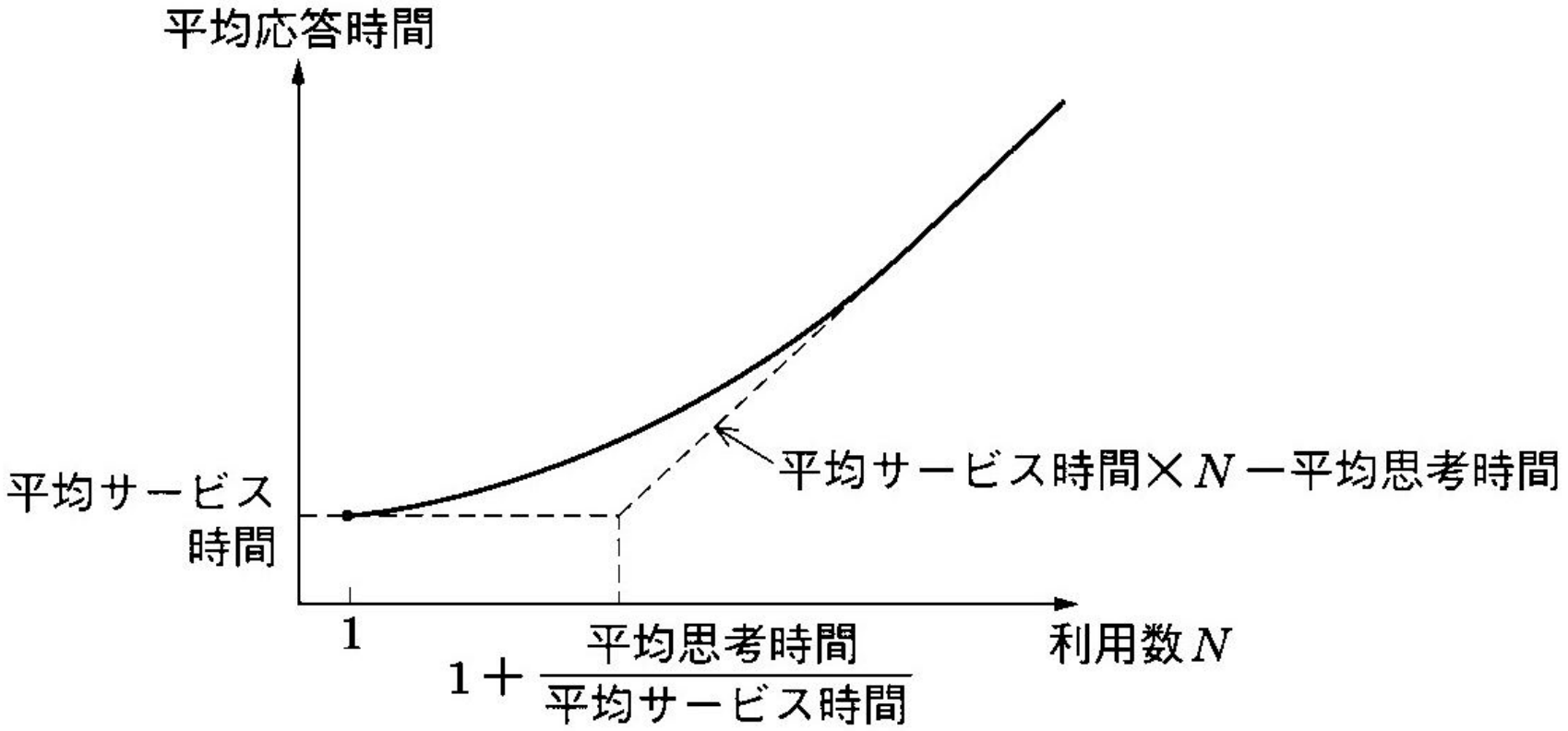


図 14.7　利用者数と平均応答時間の関係

復習

プロセススケジュール

- プロセスはそれぞれの事情で休止したり停止したりしているが，いつかは条件がそろい実行可能となる．
- 複数の実行可能プロセスがあった場合，どれから実行するかを決めるのはOSである．
- この決め方のアルゴリズムがいくつかあり，OSの授業では定番のネタとなっている．

復習

アルゴリズム紹介 1/2

- First Come First Served (FCFS)
  - 生成された順番にCPUを割り当てる.
- Shortest Processing Time First
  - 短いものから片づける
- Priority Scheduling
  - 予めプロセスに優先度をつける
  - UNIXでも一部採用されている
- Round Robin
  - 短いタイムスパンで公平に切り替える，TSS的.
  - 今時のOSの定番.

復習

アルゴリズム紹介 2/2

- PriorityとRound Robinの組み合わせ
- Dynamic Dispatching
  - I/Oの利用頻度に基づき，優先度を変更する.
  - 今時のPC等には不向きだが，大型汎用コンピュータでは有効なのかもしれない.
- Multilevel Feedback Queue
  - 優先度の異なる複数の実行待ちの行列を準備する.

復習

アルゴリズム全体の雰囲気

- 多くのプロセスをトータルで短時間で終わらせることに注視している.
  - 銀行の夜間一括処理的なものをスコープとしている.
  - 教科書の図7.7, 7.8等は如実にソレ示している.
- 今時の対話的な処理の場合「処理の終わり」が明確でないため，実は Round Robin 以外，あまり役に立たない.
- プロセスの応答性能を考慮したアルゴリズムが今後は重要となるでしょう.

応答時間と経過時間の関係

- 個々の応答時間の長さによって，応答時間がどうかわってくるかを，スケジュールアルゴリズムによって比較する.
- 正直，コンテクストスイッチのタイミングと，応答・思考の区切りが一致するとは限らないので，ちょっとこの比較に意味があるか疑問.
- 応答時間が長い(コンテクストスイッチが長い)と，どのアルゴリズムも不利.

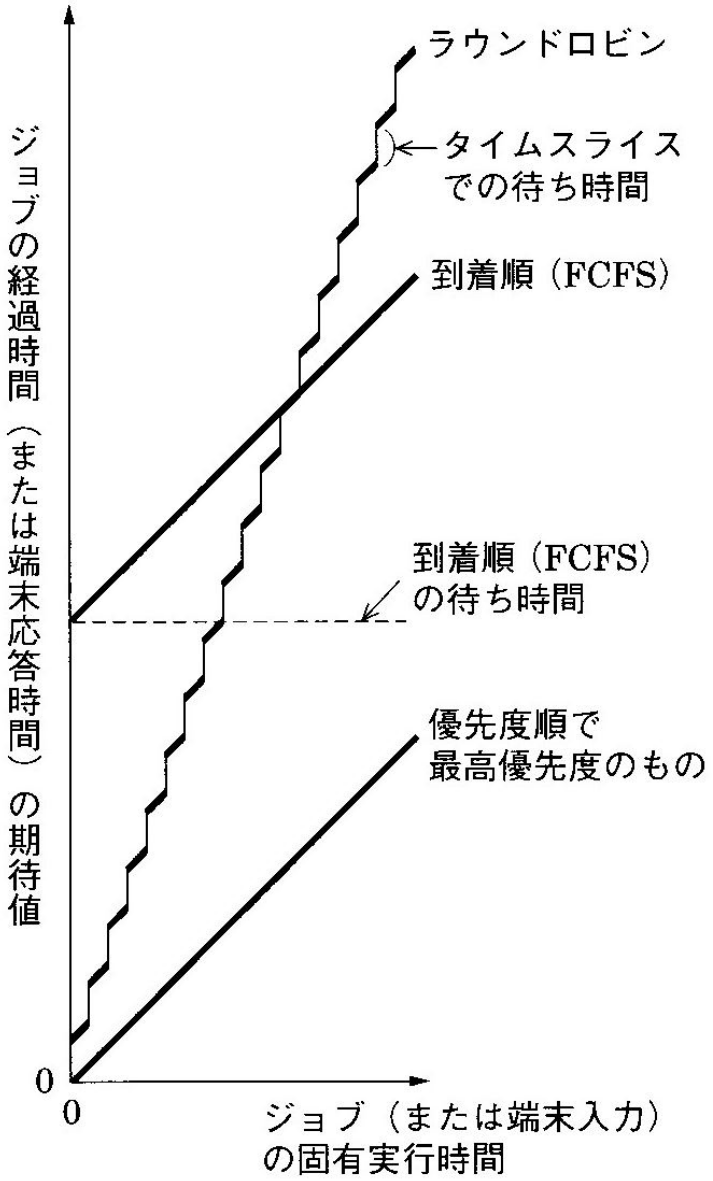


図 14.8　スケジューリングアルゴリズムとジョブの経過時間
（または端末応答時間）

スケジューリングとスループット

- 前述のように，処理に明確な終わりが無い場合，スループットの意味はあまりない．
- 一般利用者の手元で行われる処理では，スループットを議論しても仕方ない．
  - ラウンドロビン以外の選択肢はほぼ無い．
- しかし，ビッグデータの解析等では，ラウンドロビンよりもFCFS等のアルゴリズムのほうが良いのかもしれない．
  - 大規模データを長時間，分析するため．

オーバーヘッド

- 本来の処理を行う補助のため，OSが行わなければいけない処理をこう呼ぶ.
- 具体的には,
  - コンテクストスイッチ
  - 仮想記憶からのページの復帰
- 無論，オーバーヘッドは小さくする必要がある.
  - 前述のtimeの例をみても，経過時間0.22に対して，実際の処理時間は0.1である.

再掲載

timeコマンド

```
[kaiya@flute03 ACM]$ /usr/bin/time -v platex sig-alternate
This is e-pTeX, Version 3.1415926-p3.2-110825-2.3 (utf8.euc) (TeX Live 2012/dev)
 restricted \write18 enabled.
```

中略

```
(see the transcript file for additional information)
Output written on sig-alternate.dvi (5 pages, 29960 bytes).
Transcript written on sig-alternate.log.
        Command being timed: "platex sig-alternate"
        User time (seconds): 0.09
        System time (seconds): 0.01
        Percent of CPU this job got: 47%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.22
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 96992
        Average resident set size (kbytes): 0
        Major (requiring I/O) page faults: 0
        Minor (reclaiming a frame) page faults: 4158
        Voluntary context switches: 48
        Involuntary context switches: 47
        Swaps: 0
        File system inputs: 16
        File system outputs: 104
        Socket messages sent: 0
        Socket messages received: 0
        Signals delivered: 0
        Page size (bytes): 4096
        Exit status: 0
```

合計時間
0.1秒

経過時間
0.22秒

48+47回
プロセスは
停止している

ベンチマーク Benchmark

- システムの性能を測定するための指標.
- 数多くの指標が存在する. (後述)
- 実際に指標の値を測定するためのデータとツールが存在する. (後述)
- 同じベンチマークを異なるシステムで利用することで, システムの優劣を比較できる.

実例 UnixBench

- 名前の通りUNIX系OS(含むLinux)のベンチマークをするためのツール.
- 複数の指標を提供し，システムの性能を数値化する，具体的には，
  - 整数演算の性能
  - 浮動小数点演算の性能 FPUの性能
  - 新規プロセス生成に関する性能
  - ファイルコピーの性能
  - パイプによる通信の性能
  - システムコール呼び出しに関する性能
  - グラフィックス表示に関する性能

OSの外部仕様

- OSが提供するAPIやサービスがある基準に準拠していれば，OSの変更や更新がしやすい．
  - 例えば，APIの名前と引数が統一されていなければ，OS毎にアプリケーションプログラムを作り直す必要がある．
- そういった意味で，OSの仕様を標準化することが重要となる．

そもそも仕様とは何か？

- Specification
- 仕様を書類にしたものが仕様書と呼ばれる.
- 「あるシステムやプログラムが，どのような環境下で，何をするのかの記述」
- 機能仕様
  - 狭い意味の仕様.
  - 大まかにいうと，何を入れると何が出てくるか定めたもの.
  - 例:「read関数はファイル名を与えると，ファイルの中身のデータを得られる」
- 非機能仕様
  - 機能仕様に加えて，応答の速さや，信頼性を定めたもの.
  - 例:「read関数は1ms以内に必ず応答し，99.99%の精度で正しいデータを返す」(実際のreadはこんな仕様はありません)

OSの仕様の例

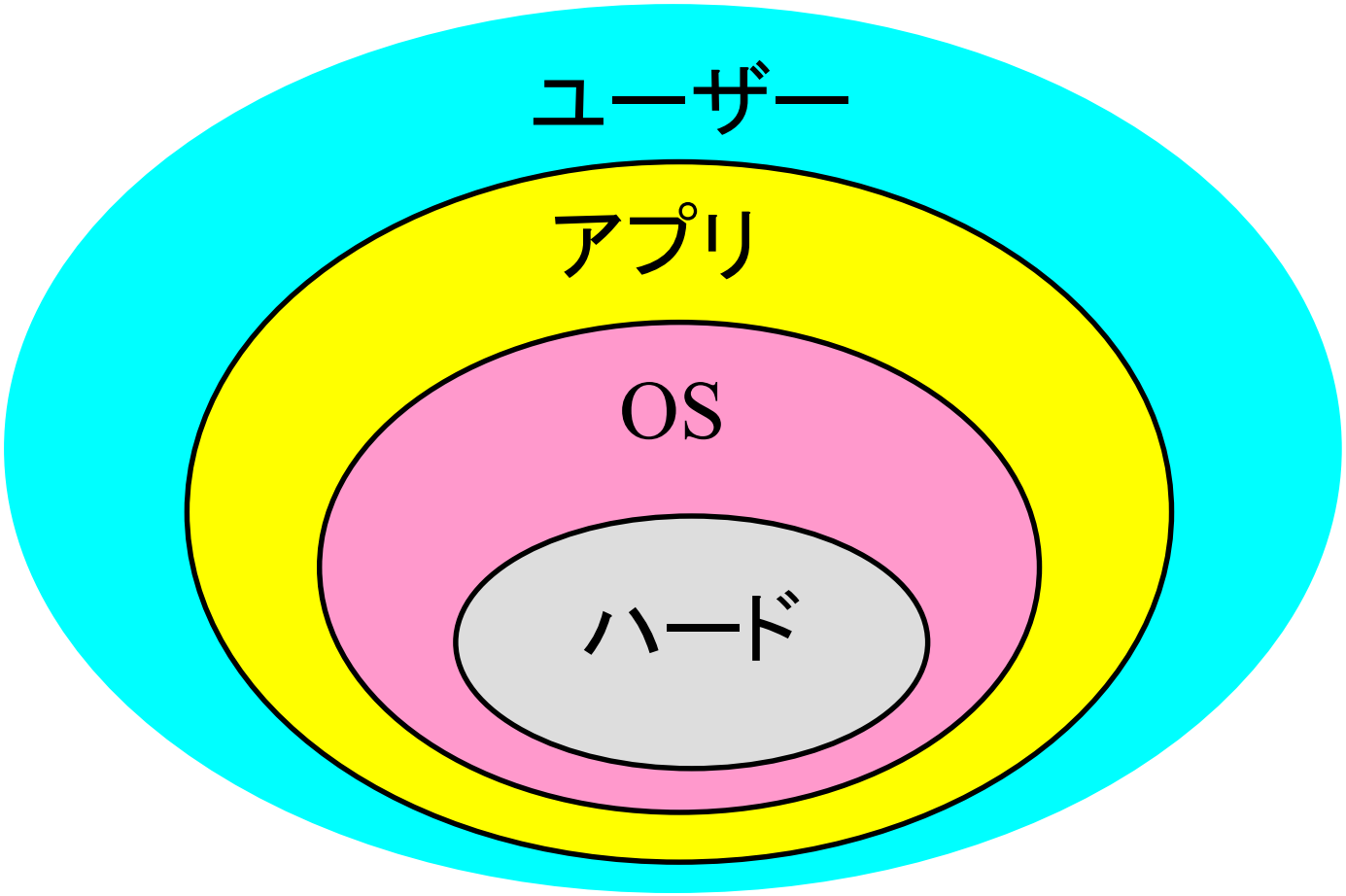
- POSIX
  - UNIX系OSをベースとしたAPIの仕様.
  - Windows NTも POSIX 1.0 に準拠している.
- X/OpenのUNIX
  - Single UNIX Specification (SUS)
  - UNIX98, UNIX03等の呼称で呼ばれる.
  - 商用の仕様なので正式登録にお金がかかると思われる.
    - 登録量が無料としても，登録にかかる作業には費用も時間もかかる.
  - 結果，フリーのもの(Linux, Free BSD)はUNIX仕様にはオフィシャルには準拠していない.

OSの仮想化技術

- OSの仮想化技術は，「成りすまし」の技術である．
- 本来，あるOSやあるハードであるはずなのに，OSに関わる何かを騙して，振舞う仕組み．

OS周辺を階層的にモデル化

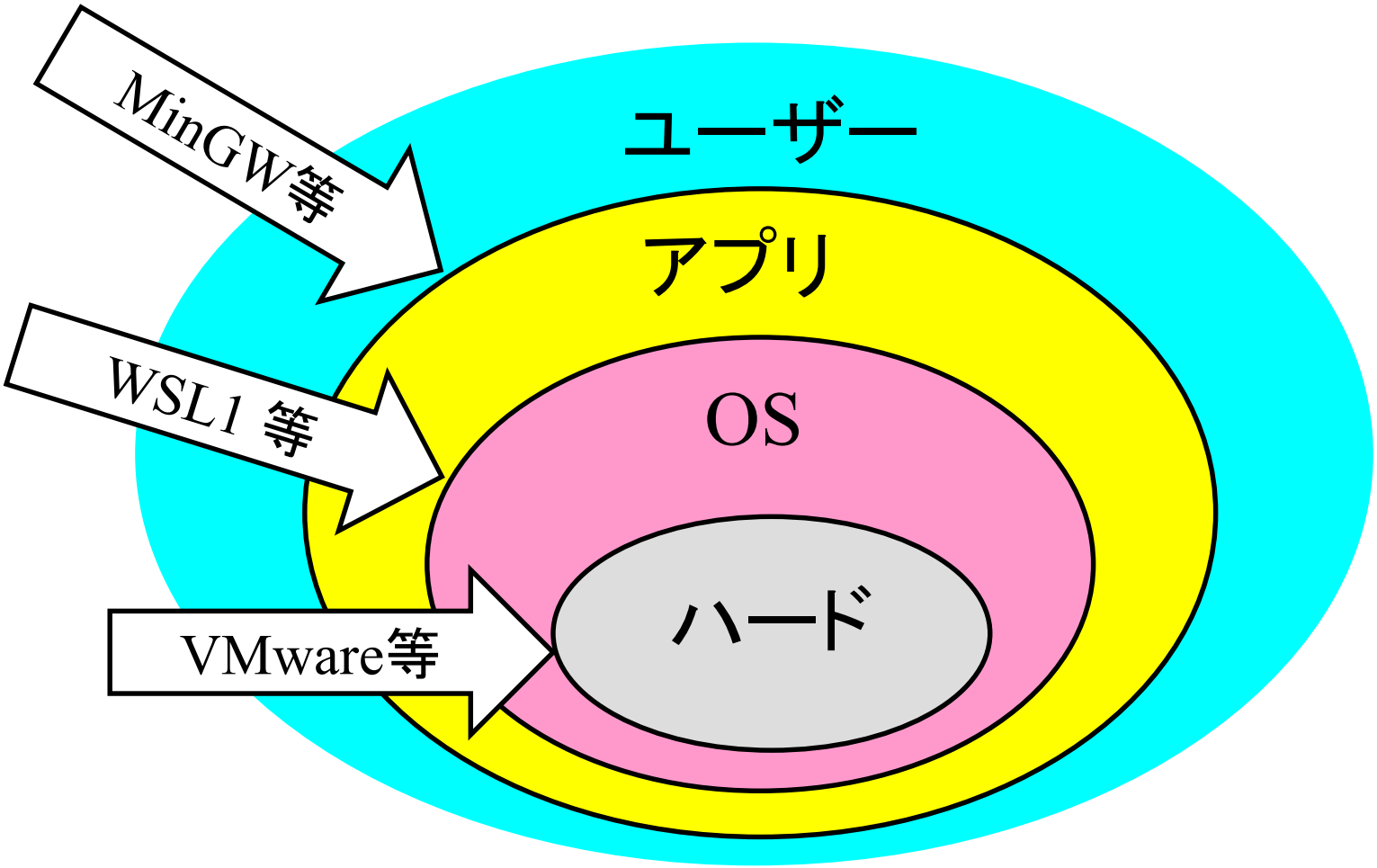
- 下図のように，より内側の存在を隠蔽するような役割がOSやアプリにはある.



騙しの分類

- ユーザーを騙す
  - 見た目が他のOSのように振舞う技術.
  - 例えば，本当は単なるWindowsアプリなのに，Linuxのような操作感がある.
  - 例 Cygwin, MinGW
- アプリを騙す
  - 他のOSの実行ファイルを動かす技術，所謂，エミュレータ.
  - 例えば，Linuxの実行ファイルがWindowsで動く.
  - 例 WSL1 (Windows Subsystem for Linux)
- OS全体を騙す
  - 本当はPC(ハードウェア)じゃないのに，アプリがハードウェアのように振舞う技術.
  - 所謂，仮想マシン，VMware, VirtualBox 等.

階層モデルでの騙される範囲



簡単なエピローグ

本授業で理解してほしいこと

- プロセス管理
- メモリ管理
- ファイルシステム管理
- デバイス管理

ハードウェアの上で，
複数のプロセスが同時に動き，
時には入出力をしつつ
処理を進めるイメージを頭に描けるようにしてください．

- その他，細かいことは，まぁ関心があったら覚えておいてください．

半年間，ご苦労さま

アンケートのほう，
よろしくご提出ください