仮想化な、
まあまあてにすな
ひどすぎる乱立

VM実現技術の本質

たけおか@AXE

2009/AUG/28

2009/SEP/04

乱立中

有名な仮想化ソフトウェア

- Xen (Citrix)
- KVM
- VirtualBox (Sun)
- VMWare (VMWare (EMC))
- HyperV (Microsoft)
- Virtual PC (Microsoft)
- Parallels (Parallels)

組込みに適用できる要件(最低限)

- マルチプラットフォーム
  - ホストOS、ゲストOS、CPU
  - Windowsが動く必要なんか無い(?)
- ソースコードが供給される
  - オープンソースであれば、いじり易い(現状)
  - 信頼性、品質の向上を独自におこなえる
- エンタープライズ・サーバと事情は近い
  - OS: Linux, UNIX(AIX, Solaris, HP-UX), 独自OS
  - CPU: PowerPC, Sparc, IA64, PA-RISC, Mシリーズ(390)

組込みに適用できる

- Linuxとμ iTRONが同時に動くのか???
  - 可能でしょう

仮想化とは?

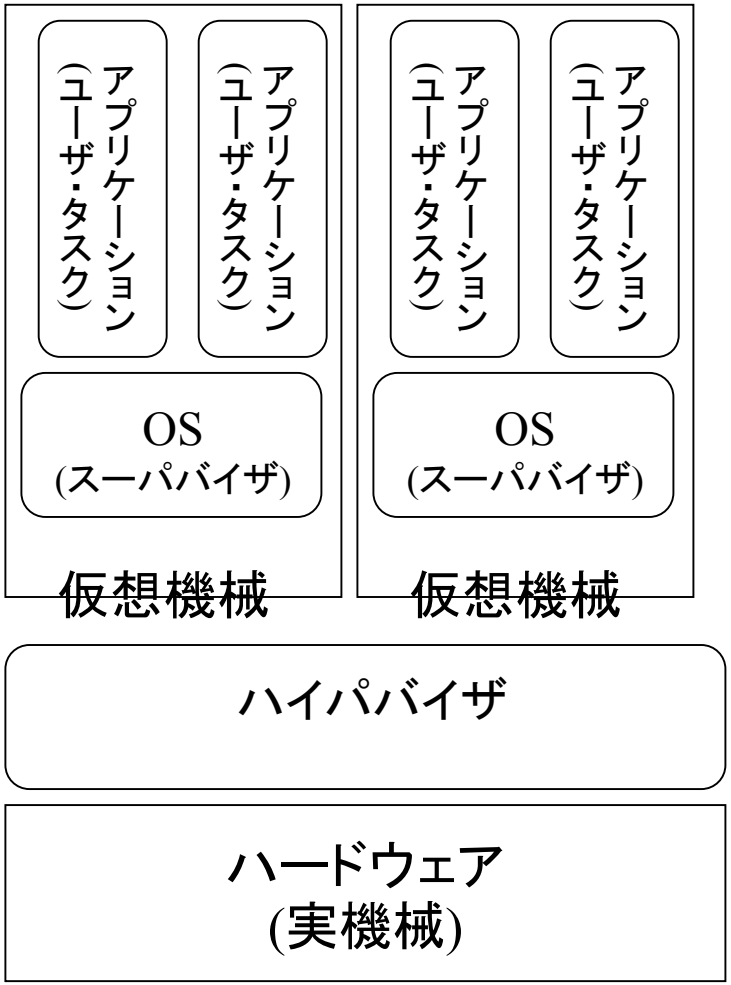
VM 対 VM !? 誤解なきよう

- OSのVMは、仮想マシン・インタープリタではない
  - 「仮想化技術(VM)」と、Java仮想マシン(JavaVM)とは、無関係
  - なんでも「仮想マシン」とも「仮想機械」ともいう…
    - 仮想化技術と区別がつかない
  - 英語でも両者とも「Virtual Machine」
    - 仮想化技術と区別がつかない
  - ま、言葉が区別付かないのは、今では仕方ないです。
  - UCSD-P Systemが悪いのです(多分)
    - 仮想マシン・インタープリタは「Pseudo Machine」とでもしてくれれば良かったのですが…

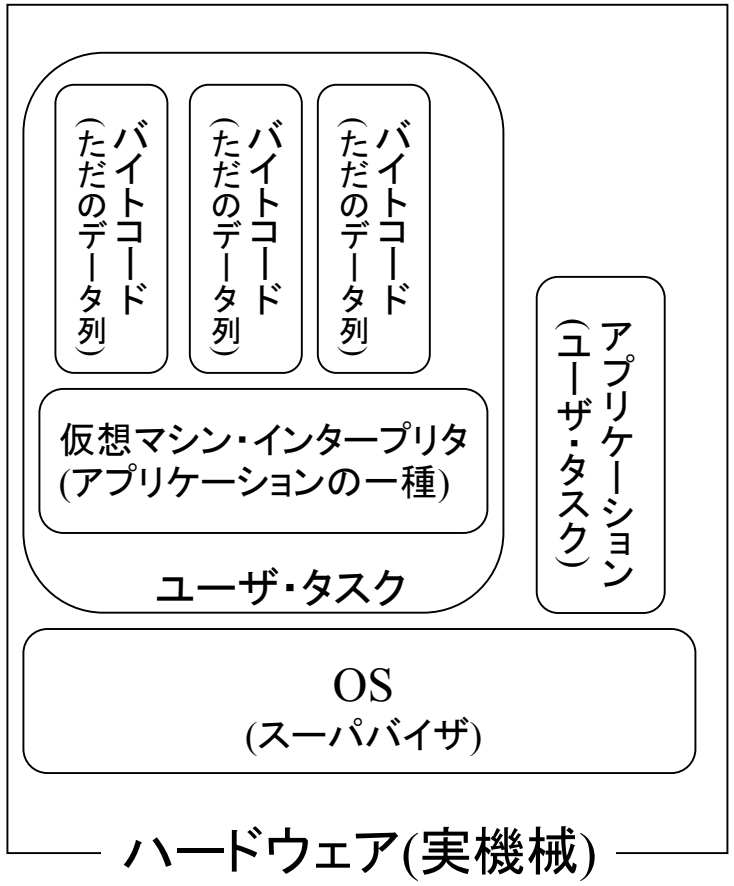
仮想マシン・インタープリタとは(寄り道)

- 仮想マシンの機械語命令を、実行時に解釈して実行する実行系
  - 機械語命令を「バイトコード」などと呼ぶことが多い
  - 実行系を「バイトコード・インタープリタ」と呼ぶことも多い
  - JIT(Just Intime Translator):実行時にネイティブ機械語に変換することもあり
- 有名な仮想マシン(インタープリタ)
  - Java仮想マシン(JavaVM) 1990年代末期～
  - UCSD-Pシステム (1970年代中期～80年代初期)
    - UCSD Pascalから始まったシステム
    - 核に、P-codeマシンという、仮想マシン・インタープリタがあった (http://en.wikipedia.org/wiki/UCSD_p-System)
  - Smalltalk80もバイトコードにコンパイルした (1980年代初頭～)
    - Smalltalk80のダンプイメージは、Smalltalk80が動いていればどこでも動いた (http://ja.wikipedia.org/wiki/Smalltalk)
  - Warren Abstract Machine(WAM)(1980年代末期)
    - Prologの振る舞いを抽象化した。WAMがあまりに効率的なので、WAM命令をバイトコードとする仮想機械実現が流行した (http://en.wikipedia.org/wiki/Warren_abstract_machine)

仮想マシン・インタープリタとは(寄り道)



仮想機械

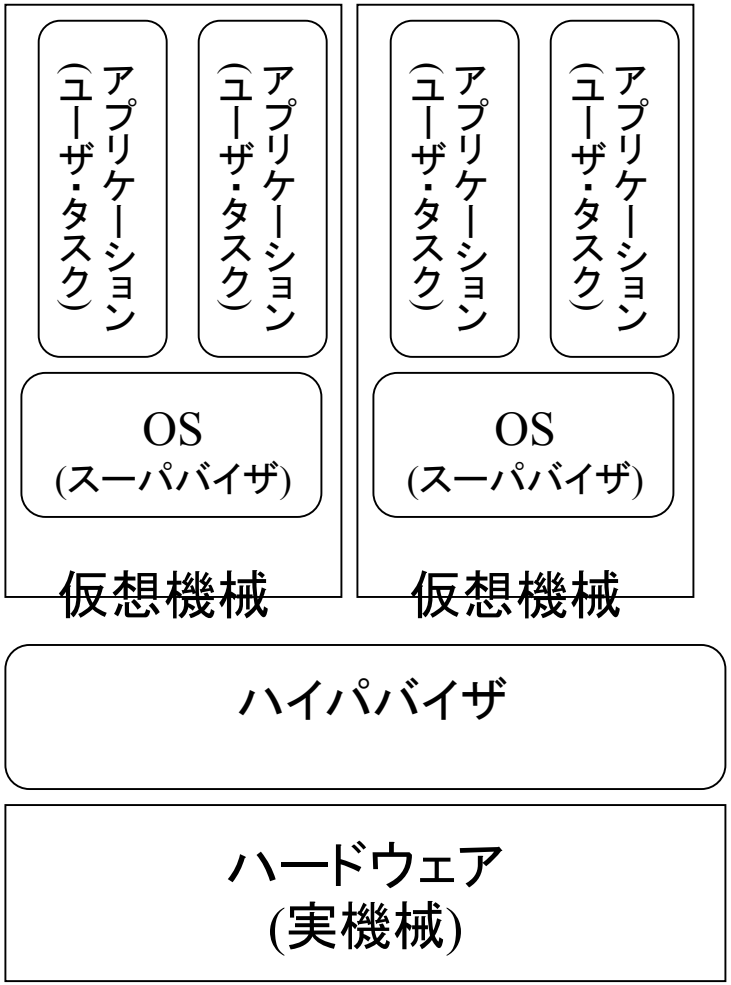


仮想マシン・インタープリタ

VMの年表もどき

- 1966
  - System/360
  - CP-40(ハイパバイザ), CMS(ゲストOS)
  - System/360 model40 上で 14個の仮想機械
- 1972
  - System/370-VS (仮想記憶システム)
  - VM/370 (ハイパバイザなど)
- 1982
  - 68010発表: 仮想機械と仮想記憶をサポートしたアーキテクチャ
- 2003
  - Xen
  - ケンブリッジ大学Computer Laboratory

VMとは(1)


アプリケーション
(ユーザ・タスク)
アプリケーション
(ユーザ・タスク)
OS
(スーパバイザ)
仮想機械
アプリケーション
(ユーザ・タスク)
アプリケーション
(ユーザ・タスク)
OS
(スーパバイザ)
仮想機械
ハイパバイザ
ハードウェア
(実機械)

VMとは(2)

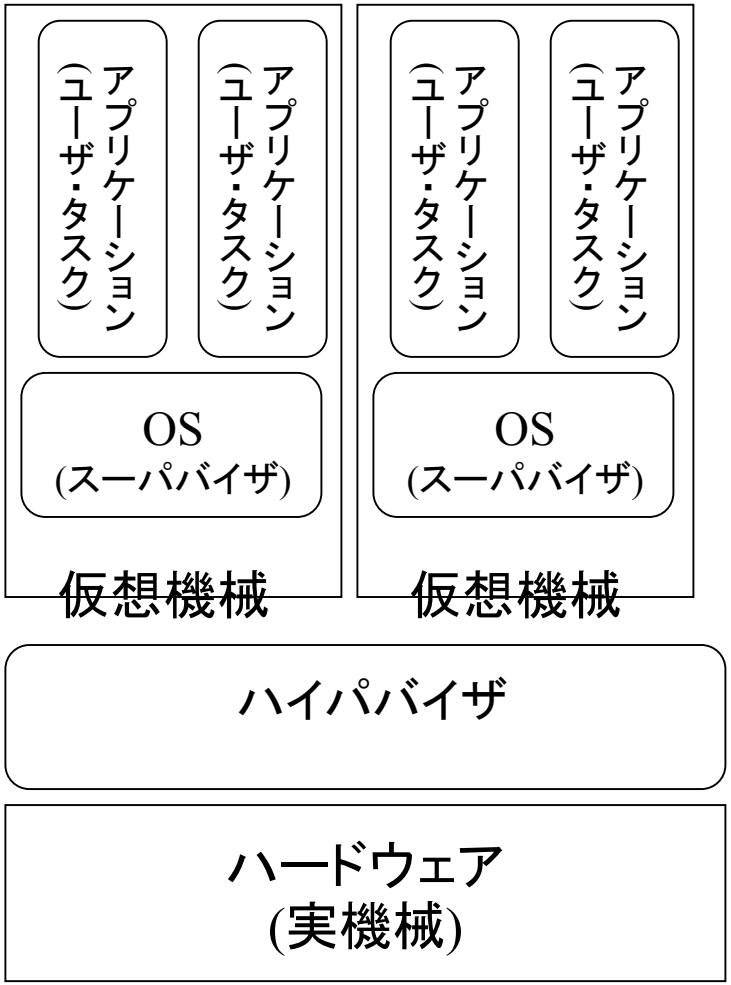
- 機械語命令は,そのまま実計算機で実行される
- しかし、機械が複数ある(ように見える)

- 完全なメモリの分離
  - それだけなら、マルチタスクOSでも可能
  - それ以外にも…

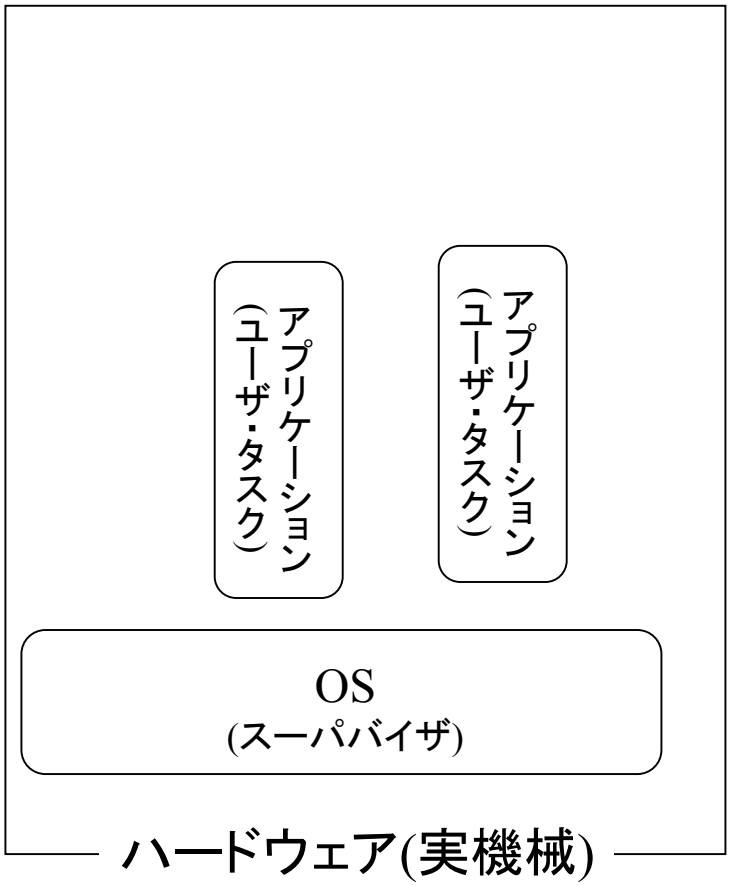
VMとは(3)

- マルチタスクとどう違うのか?
- OSが仮想機械ごとに動く
  - マルチタスクはOSは一つ
  - VMでは、仮想機械ごとにOSが必要(またはOSレス?)
- IOが独立
  - マルチタスクOSは、IOはシステム全体で共用
  - VMでは、仮想化された独立IOが与えられる
- 仮想機械で実行されていることに気づかない
  - アプリケーションもOSも
- 仮想機械ごとに完全に独立、不干渉
  - 安全。バグの影響が伝播しない
  - セキュリティ上も

マルチタスクとどう違うのか?



仮想機械



マルチタスク

VMの簡単な構造

- ハイパバイザが居る
  - 実計算機の資源の管理を行う
- 仮想機械
  - 仮想機械を「ゲスト機械(マシン)」と呼ぶことはない
    - 豆知識:「ゲスト・マシン」というのは、「ユニバーサル・ホスト」という概念の計算機アーキテクチャで実現されるハードウェア計算機のことをいう言葉
- 仮想機械内で動作するOSを「ゲストOS」と呼んだり
- 「ホストOS」は、実計算機の実資源を駆動する
  - ハイパバイザとの関係は後述

VMとは(4)

ハイパバイザの存在に気づかない(1)

- 仮想機械で実行されていることに気づかない
  - アプリケーションもOSも
  - 仮想化管理ソフトウェアが存在するのに
  - ハードウェア的にも…

- 仮想機械管理ソフトウェア
  - ハイパバイザ Hypervisor
    - OSはスーパバイザ Supervisor
  - IBM System370では「CP」

VMとは(5)

ハイパバイザの存在に気づかない(2)

- 仮想機械で実行されていることに、ソフトウェアは気づかない
  - ハードウェア的にも…
- ハイパバイザ用のレジスタが実機械にあったとしても、仮想機械は気がつかない
  - ハイパバイザ用レジスタが仮想機械に見えて(読み/書きできて)はいけない
    - (極論)仮想機械内のOS,アプリケーションが暴走したときに、仮想機械と実機械で暴走の仕方に差が無いのが理想
      - ハイパバイザ・レジスタの状態によって、仮想機械内のソフトウェアの振る舞いに差が出てはならない

ハイパバイザ用のレジスタが実機械にあったとしても、仮想機械は気がつかない

- ハイパバイザ用レジスタは、OS(特権レベル動作)にもアプリケーション(ユーザ権限動作)にも見えない
  - 空間が違って、アクセスできない
  - 特殊なレジスタ・アクセス命令になっていて命令を発行できない
    - ハイパバイザ特権命令とか
- あまつさえ、通常のステータス・レジスタさえ、ユーザ権限からはアクセスできなくする
  - IBM/360,370, 68010

VMとは(6)

ハイパバイザの存在に気づかない(3)

- ハイパバイザ用のレジスタが実機械にあったとしても、仮想機械は気がつかない
  - 実機械の定義が重要
  - ハイパバイザは、実機械のハイパバイザ・レジスタを見る、実は。
    - 実機械って仮想機械と別物になってるじゃん！？

VMとは(7)

ハイパバイザの存在に気づかない(4)

- 歴史的には、ハイパバイザ用の特殊な機能は無しに、仮想化を行っていた
- 実機械と仮想機械は、完全に同じレジスタセット(ハードウェア機能)を持っていた
- IBM S/360, S/370では
  - ハイパバイザだけが、巧妙にスーパバイザ・モードで走行し、
  - 仮想機械のスーパバイザは、だまされて走行していた
  - CPUステータス・レジスタを読み書きしようとすると、特権例外が発生し、スーパバイザに制御が移る
    - その後、ハイパバイザがよきに計らって、制御をOSなどに戻す

ハイパバイザのお仕事

VMとは

- OSが仮想機械ごとに動く
  - VMでは、仮想機械ごとにOSが必要(またはOSレス?)
- IOが独立
  - マルチタスクOSは、IOはシステム全体で共用
  - VMでは、仮想化された独立IOが与えられる

- 仮想機械で実行されていることに気づかない
  - アプリケーションもOSも

ハイパバイザのお仕事

- 資源管理
- VM間通信

- IOシミュレート

資源管理

- 資源とは?
  - 計算機一般の資源と同じ、ではあるが…
- 資源
  - メモリ
  - CPU時間
  - IO
    - HDD
    - プリンタ
    - 端末
    - ネットワーク
    - その他
  - システム・レジスタ
    - スーパバイザが管理すべき特殊レジスタ(アクセスに特権が必要)

資源管理 メモリ(1)

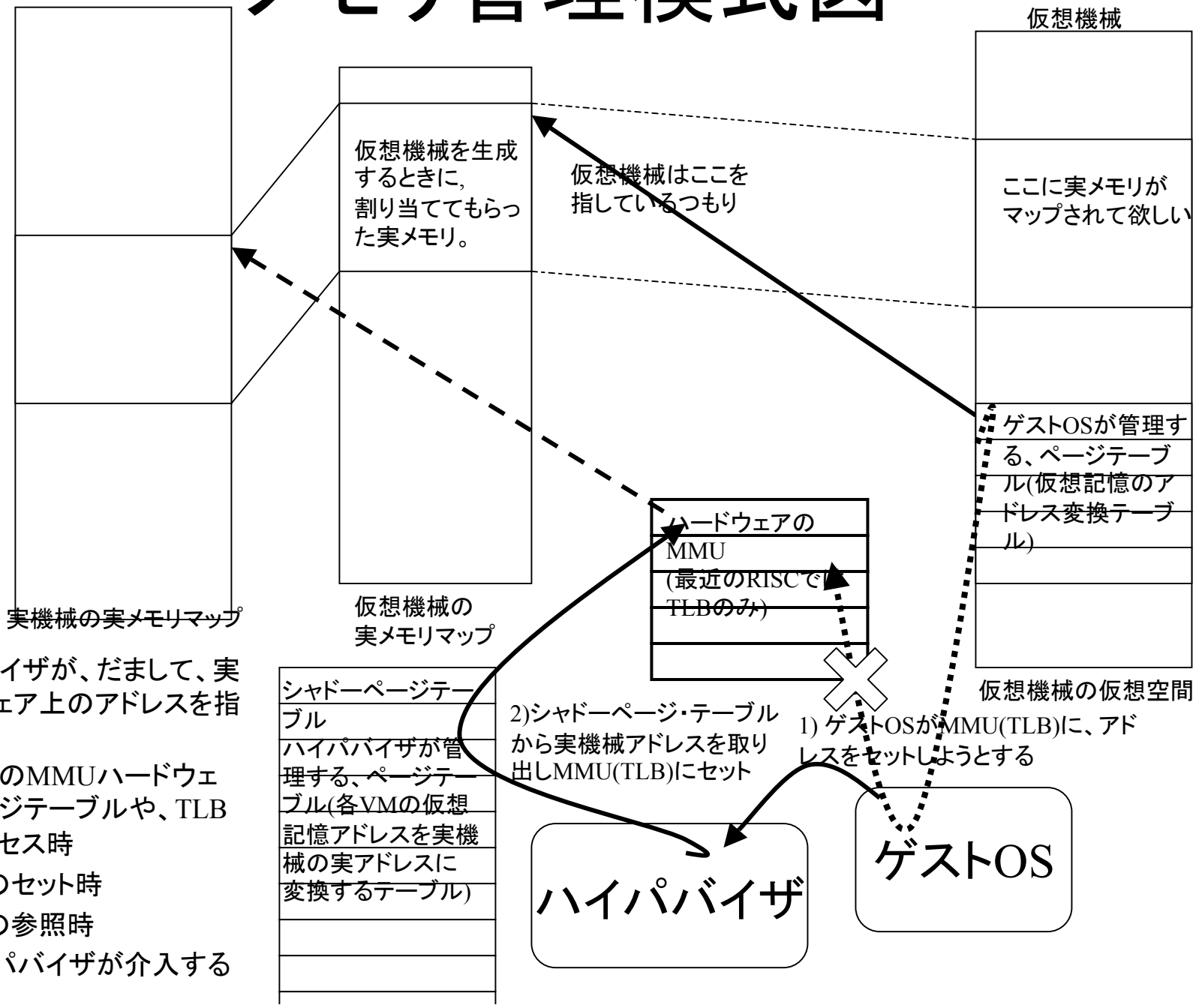
- 実計算機の実メモリを割り当てる
- 仮想空間も重要な資源
  - ページテーブル (仮想記憶用のアドレス変換テーブル)
  - ページテーブルは有限
    - 機械ごとにページテーブルを持つことも可能だが…
    - ページテーブルはかなり大きい
      - ページテーブルを仮想空間に置く事は可能だが、OS,ハイパバイザがかなり複雑になる
  - システムに一つの大きなページテーブルを持つ
    - シャドー・ページテーブル

豆知識:マイクロ・カーネル Machも、仮想空間を資源として管理してました

資源管理 メモリ(2)

- 仮想空間も重要な資源
  - ハイパバイザが隠したページテーブルを持つ
    - シャドー・ページテーブル
  - 仮想機械のゲストOSが、MMUのページテーブルに値をセットするとき、トラップが発生してハイパバイザに制御が移る
    - ハイパバイザは、実機械のメモリ番地を、MMUのページテーブルにセットする。その後、正常復帰。
    - MMUのページテーブルを読み出すときも同様にハイパバイザが介入し、実機械のアドレスではなく、仮想機械内のアドレスが読み出せたように振舞う
    - 通常、MMUアクセスは、特権命令や特権IOになっている
    - 近代的なTLBしか無い機械では、TLBにセットする値を調整するだけ。より簡単。
      - TLBミスによりハイパバイザを起動すると、オーバヘッドが少なくできたり

メモリ管理模式図



- ハイパバイザが、だまして、実ハードウェア上のアドレスを指させる
- ゲストOSのMMUハードウェアのページテーブルや、TLBへのアクセス時
  - 値のセット時
  - 値の参照時

  に、ハイパバイザが介入する

資源管理 メモリ(3)

- 仮想機械同士が干渉しないようにする
- 仮想機械間で情報が漏れないようにする
  - 仮想機械は完全に独立
  - 他の仮想機械のメモリを読み書きできてはいけない

- ページテーブル管理で実現
  - 無関係な機械のメモリ・ページは絶対に、仮想機械の論理空間にマップさせない

資源管理 メモリ(4)

- 仮想計算機の肝はページテーブル管理
  - 仮想機械のアイソレーションは、メモリを読み書きさせない
  - 無関係な機械のメモリ・ページは絶対に、仮想機械の論理空間にマップさせない
  - 仮想記憶のための「アドレス変換機構」を使用して、ページ単位でアクセスを制御
  - 仮想記憶用のページ・テーブルや、TLB(アドレス変換バッファ)へセットする値を、ハイパバイザが制御する

資源管理 CPU時間

- CPU時間が適切に割り当たるようにする
  - 無限封鎖がない
  - 公平がいいとは限らない
    - そもそも「公平」とは???
    - 組み込みは実時間性が必要 (いつもか?)
- 実時間性
  - 割り込みルーチンは、仮想機械内で動くの?
    - 実機械レベル=ハイパバイザ内で動くの???
    - ハイパバイザで消費する時間は誰に付けるのか?

資源管理 IO

- IO
  - HDD
  - プリンタ
  - 端末
  - ネットワーク
  - タイマ割り込み
  - その他
    - パラレルIOポート (GPIOポート)
    - シリアルIOポート
    - その他が、組込みで重要
- そもそも大型機(System370)のIOとは…

資源管理 IO

- そもそも大型機(System370など)のIOとは
  - IOチャンネルというものが普通
    - 「チャンネル・プロセッサ」というものが居る
      - 高速DMAでIO処理を行う
      - メインのプロセッサは、チャンネル・プロセッサを起動して、あとは終了割り込みを待つ
    - アプリケーションは、(出力の場合)

      1)DMAされる領域にデータを置き

      2)チャンネル・プロセッサの起動を呼び出す

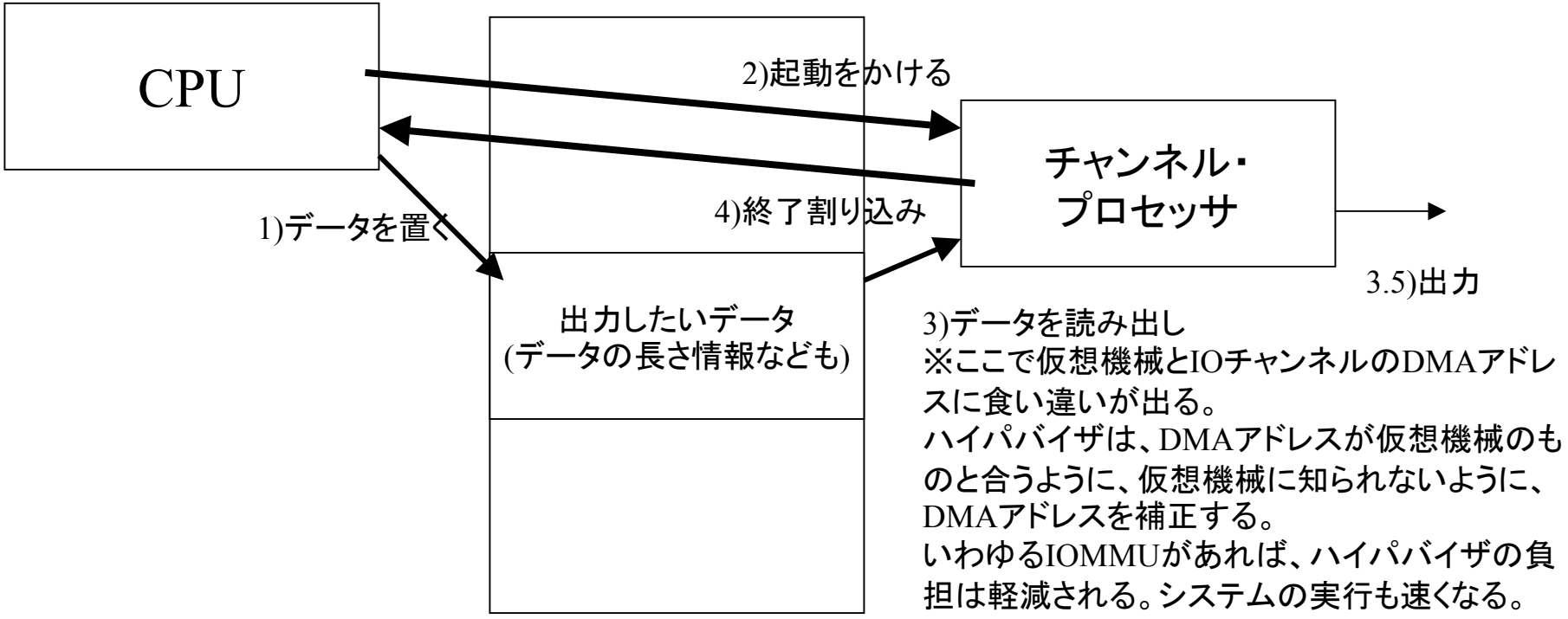
      3)チャンネル・プロセッサが動作終了すると、コールバック・ルーチンが呼び出される

    ※入力も同様。

      IOチャンネルから、終了割り込みが来たら、指定した領域にデータが入っている

資源管理 IO

- IOチャンネルというものが普通
  - (制御ブロックに終了結果を書くのが普通。だが略)



資源管理 IO

- IOチャンネルというものが普通
- なんでもIOチャンネルなので、仮想化が容易
  - IOポートを直接叩いたりしていない(重要)
  - 割り込みはシミュレートし易い
  - HDDのセクタ読み出しも、IOチャンネル
    - でも、だんだん高度なシミュレーションを
  - プリンタもIOチャンネル
    - スプーリングなどは、当然ソフトウェアだが
  - ネットワークもIOチャンネル
    - ネットワーク・プロセッサもどんどん賢くなっているが
      - プロトコル処理がどんどんネットワーク・プロセッサに…

資源管理 IO

- IOチャンネルの仮想化対応
  - DMAと終了割り込みが基本であるから
    - IOポートを直接叩いたりしていない(重要)
  - IOチャンネルにセットするDMAアドレスを

    仮想機械内の空間のアドレス->実アドレス

    と変換すればよい

  - 割り込みはハイパバイザが受け、ソフトウェア・割り込みで出しなおすなど
    - 直接、仮想機械に割り込むことも、ハードウェア支援があれば難しくない

資源管理 IO

- HDD、ファイルは仮想化の中心のひとつ
  - だんだん高度なシミュレーションを
  - ファイルを本当にハンドリングするのは誰か?
    - 特別なゲストOS

- ネットワーク
  - そもそも速くないので、オーバヘッドが目立たない
  - 仮想機械間の通信は、仮想的なネットワークI/Fにすることがよくある

資源管理 IO

- 組込みの世界にありがちな…
- IOポート
  - それって仮想化できるの?
  - 割り込み番号は?
- 極端な話、

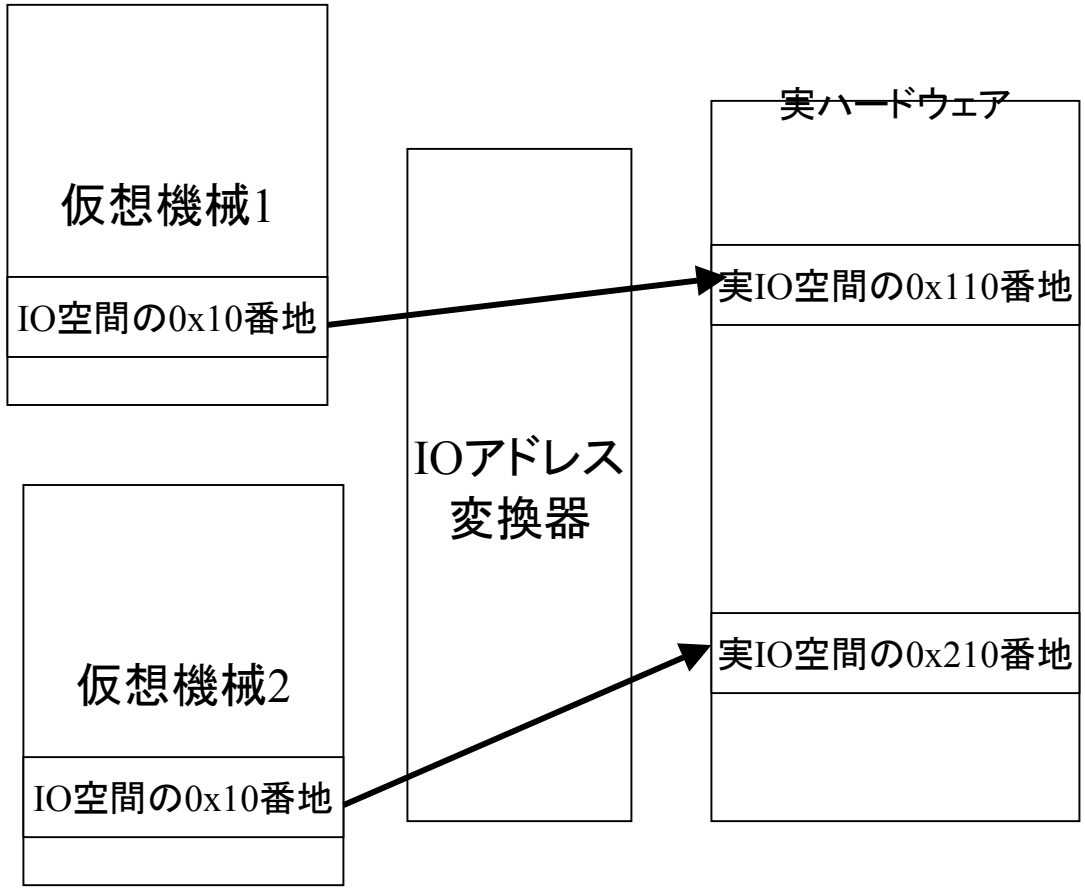
  まったく同じLinuxを、同時に複数の仮想機械で動かすとき、直接IOポートを叩いているデバイス・ドライバは、どうなるのか?

資源管理 IO

- IOポートを、仮想化するとした場合
  - IOポートの番地は通常固定だが…
    - ソフトウェアに気づかれずに、IOポートを見せるのは??
    - アドレス変換が必須
      - 領域単位ででもアドレス変換をするハードウェアは必要
  - 割り込み
    - 遅めであれば、ソフトウェア(ハイパバイザ)介在でなんとかできる場合も
    - 高速割り込みなら、実ハードウェアと仮想機械の割り込み番号の食い違いは?
    - 割り込みルーティングという技術を使って解決
      - 割り込み番号を仮想機械ごとに付け替えられるようにすべき

資源管理 IO

- IOポートのアドレス変換
  - 理想的にはこうしたい
  - IOポートは、バイト(ワード)単位でアドレスがついているから、変換するのが大変
  - アクセス禁止も同様に大変
  - 大きなIOのブロック単位(4kバイトごととかにするするか…(するとMMUと同じになる))
  - いわゆるIOMMUではない



資源管理 システム･レジスタ

- スーパバイザしかアクセスできない、特権の必要なシステムの情報の入ったレジスタのアクセスが発行された場合、
- すなわち、仮想機械上のゲストOSが、システム･レジスタにアクセスしたとき、
- トラップが発生して、ハイパバイザに制御が移る。
- ハイパバイザは、都合のいいように処理し、ゲストOSに制御を戻す

- 一般プロセスが、システム･レジスタにアクセスした場合も同様のシーケンスを行うが、最終的には、ゲストOSにより一般プロセスの例外がハンドリングされる

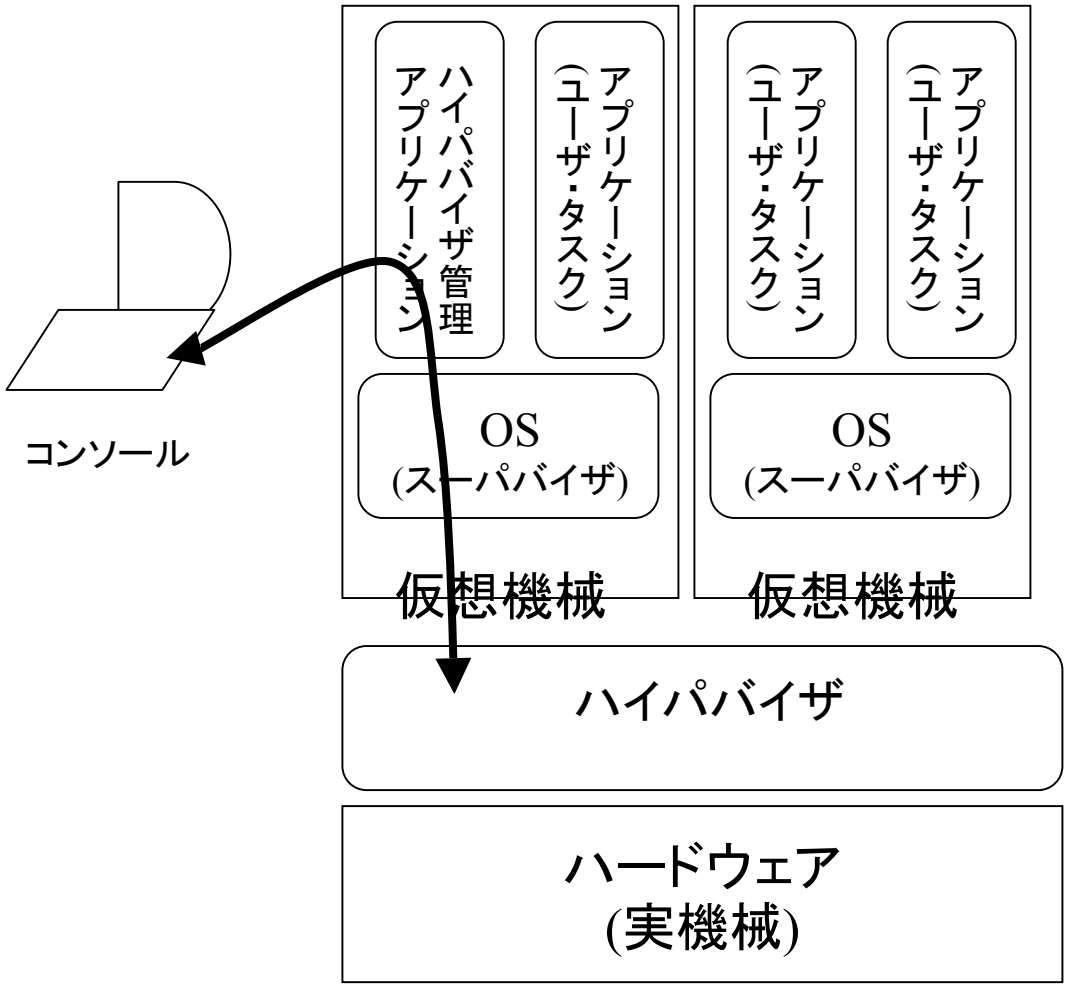
ハイパバイザは対話できない

- 通常のハイパバイザは、トップレベル(対話環境)が無い
- 仮想機械のゲストOSから、ハイパバイザ・サービスを起動するしかない
- システム管理者であっても、ハイパバイザと直接やりとりする方法はない

※KVMは、ハイパバイザとホストOSが一体なので、例外的

ハイパバイザは対話できない

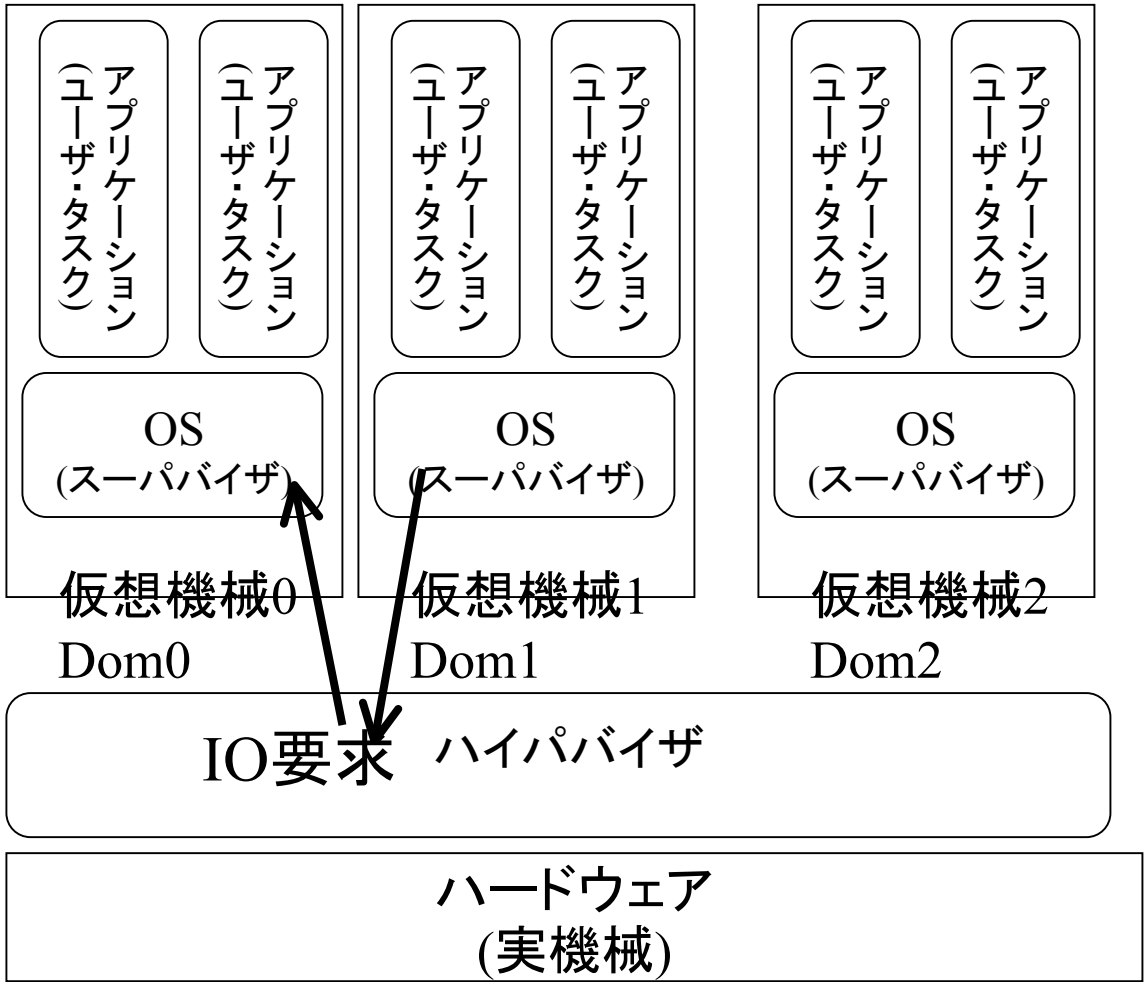
- 仮想機械でハイパバイザ管理アプリケーションを起動して管理
- ハイパバイザ単独では、ほぼ何もできない
  - ハイパバイザは資源管理しかしない



様々な仮想化方式

Xen

- ホストOSは仮想機械(Dom0)の一つ(のOS)が担当
  - ゲストOSからの実IO要求をハイパバイザが受け
  - ハイパバイザが、Dom0のOSに実際の要求を依頼



Xen

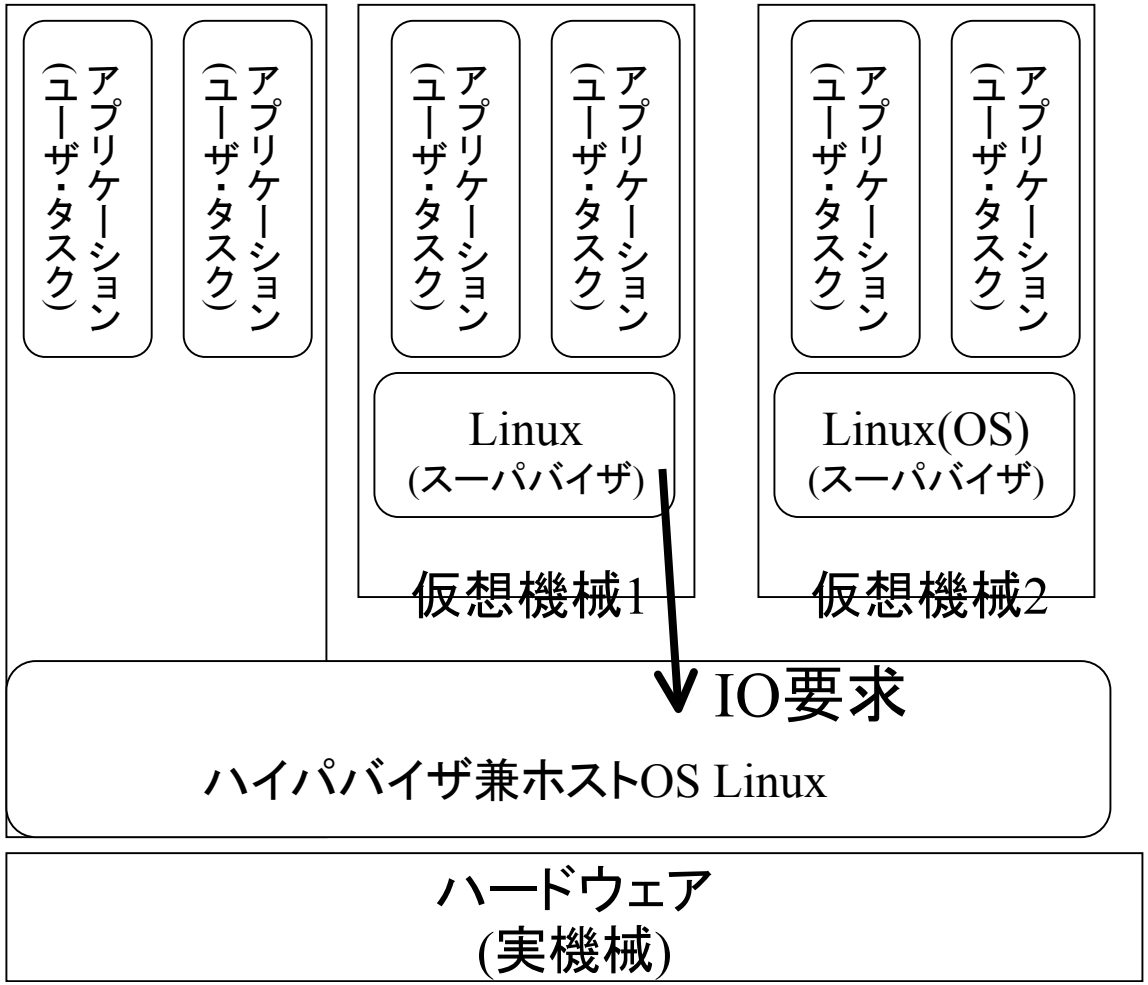
- http://www.xen.org/

- オープンソース
- ホストOS
  - Linux, NetBSD
- ゲストOS
  - Windows, Linux, Solaris, *BSD

- CPU
  - 86, x86_64, IA64, PowerPC

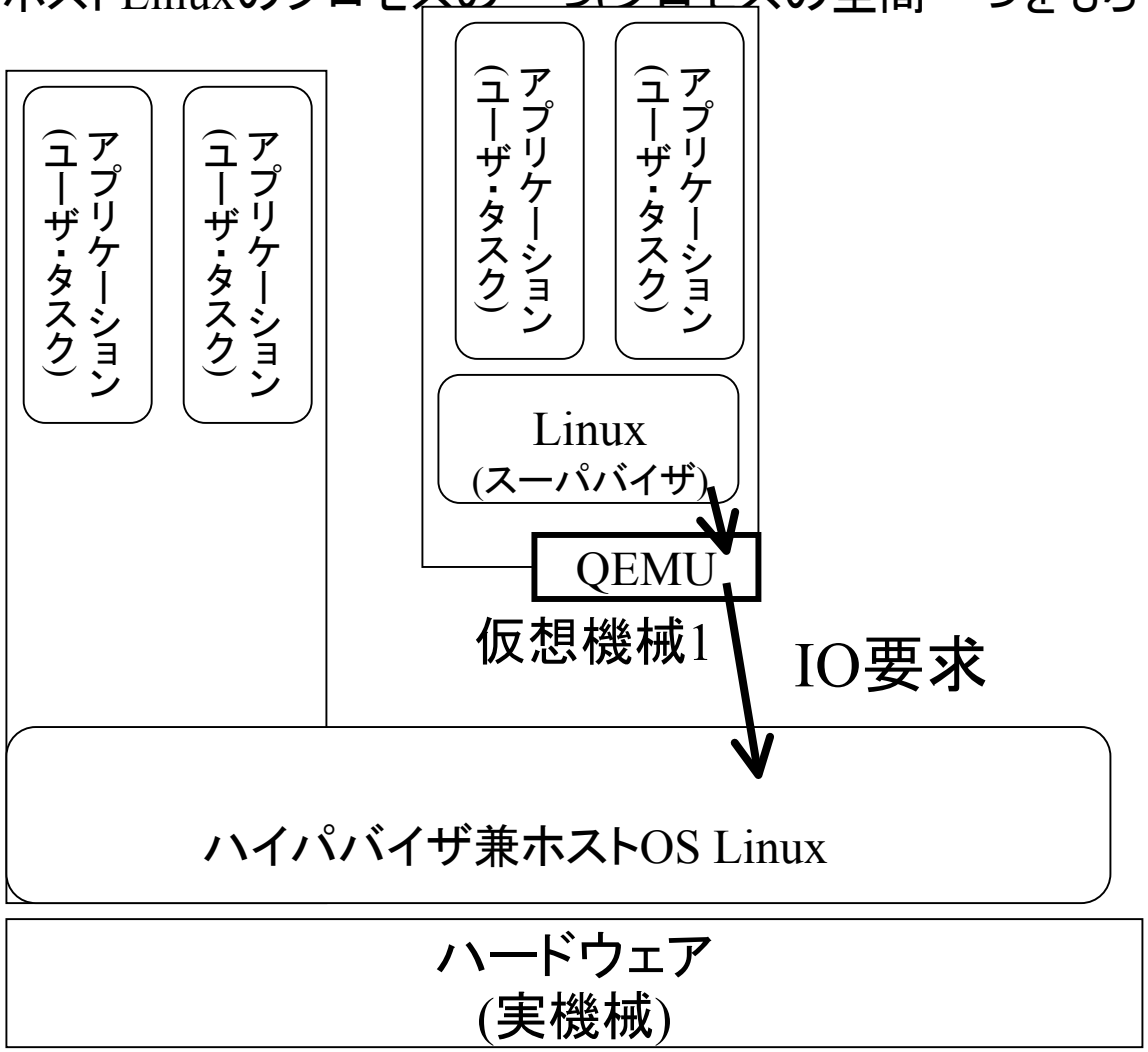
KVM

- ホストOS兼ハイパバイザのLinuxが全部やる
  - ゲストOSからの要求をホストOS Linuxが受けなんでも処理する
  - Linuxべったり



KVM

- 仮想機械のIO処理にQEMUを使ったのが特徴的。
- QEMUは仮想機械の空間で動く
- 仮想機械は、ホストLinuxのプロセスの一つ(プロセスの空間一つをもらって仮想機械が動く



KVM

- http://www.linux-kvm.org/page/Main_Page
- オープンソース
- Linuxカーネルを拡張し、ハイパバイザ機能を付加

- ホストOS
  - Linux
- ゲストOS
  - Linux,Windows, Solaris,FreeBSD, OpenBSD, DOS

- ハイパバイザが複雑で、エラーやセキュリティ・ホールがあったら、ダメなのでは???

仮想化用ハードウェア

ハードウェア・サポートのある有名なCPUアーキテクチャ

- VT-x
- AMD-V

- 68010

Intel VT-x

- Extended Page Tables (EPT)
  - ページテーブルに、属性を増やす
- Intel's Virtualization Technology for Directed I/O (VT-d)
  - いわゆるIOMMU
    - DMAするデバイスから出力されるアドレスを、アドレス変換する
    - IOチャンネルの仮想化が、非常に単純に！
- 64bitのゲストOSが動けるように配慮

68010

- 68010は仮想機械をサポート
  - 仮想記憶(VS)もサポート。SUN2は仮想記憶のあるBSDが動作。
- “MOVE from SR” (CPUのシステム・ステータス・レジスタ読み出し)を特権命令とした
  - 68000は、ユーザモードでも読み出せた
- SR読み出しでもトラップして、Hypervisorを起動した
- IBMのCP (IBMのHypervisor)は、ゲストOSもユーザ・モードで動いていた
- SR読み出しで、OSがなにかをしようとしたら、Hypervisorで手当てした
  - ユーザ・アプリケーションがSRを見ることは、通常は無い

- メモリ空間のプロテクションは、外付けのアドレス変換ハードウェア(MMU) 68451への設定で行える

デスクトップ/
エンタープライズ
仮想化の
方向

デスクトップOSの仮想化

- 典型的使用シーン
  - Windows7になってもWindowsXPが使える
  - Windowsメインだが、組込み開発にLinuxも必要
  - Windowsメインだが、同じx86でμiTRON開発ができれば素敵(夢)
  - IntelMacユーザだが、Windowsも使う

デスクトップOSの仮想化

- 求める性能
  - Windowsが欲しい
  - IO性能がすごく良い必要はない
  - ワープロ、Excel、ブラウザが普通に動けばいい
  - デバッガが普通に動けばいい
  - 各仮想機械(各ゲストOS)から、ファイルが便利にアクセスできればいい

- 結論: VMWareでも使えばよし
  - Windowsを欲しい人とは、話すことがあまりないよね

エンタープライズOSの仮想化

- 典型的使用シーン
  - 大型機ネイティブOS、各社UNIXとLinuxが動く
    - SolarisとLinux, AIXとLinux, HP-UXとLinux,MVSとLinuxとか
  - Linuxの動く仮想機械をカスタマごとに起動する
    - ハードウェアのパワーに対して、各カスタマの処理トランザクションが小さい
  - ひとつの案件を複数の仮想機械で実現
    - ある仮想機械のバックアップを、別の仮想機械で行う
    - Linuxのフォールトでゲスト OSが停止しても、同じ実ハードウェア上の、別仮想機械で即座にバックアップする
    - ある仮想機械が、セキュリティを破られて(停止させて)も、即座に別な仮想機械でバックアップする

エンタープライズのOS,CPU

- エンタープライズ・サーバには各社の事情がある
- OS
  - Linux, UNIX(AIX, Solaris, HP-UX), 独自OS
- CPU
  - PowerPC, Sparc, IA64, PA-RISC, Mシリーズ(390)
    - Windows動きません

- 組込みと近い状況にある
  - PCは家電と同じ民生品なので、PC&家電が特殊と考えるべき。本当のプロは。

エンタープライズOSの仮想化

- 求める性能
  - CPUパワーより、トランザクション性能
    - Web系だと、ネットワーク性能で抑えられる
      - CPUパワーが余りがちなので、仮想機械をたくさん作れる
  - データベース・アクセスは速い方がいい
  - 通常の仮想機械間通信はほぼ無い
  - 仮想機械のマイグレーションはできるべき

- 結論: ホストOSは、各社ネイティブかLinuxだね
  - Xen, VitualBox, KVMっていい感じ

組込み分野について

組込み分野の仮想化

- 想像できる典型的使用シーン
  - 情報系をLinuxかWindowsで、実時間系をRTOS(μiTRON)で動かしたい
  - 複数の仮想機械で、複数のRTOSを動作させる
    - 一つのRTOS(仮想機械)がフォールトしても、他の仮想機械は安全に動き続ける
      - 旧来のRTOSアプリケーションは、一つのメモリ・アクセス・エラーでシステムが落ちる可能性がある

組込みに適用できる

- Linuxとμ iTRONが同時に動くのか???
  - 可能

組込みに適用できる要件(最低限)

- マルチプラットフォーム
  - ホストOS、ゲストOS、CPU
  - Windowsが動く必要なんか無い(?)
- ソースコードが供給される
  - オープンソースであれば、いじり易い(現状)
  - 信頼性、品質の向上を独自におこなえる
- エンタープライズ・サーバと事情は近い
  - OS: Linux, UNIX(AIX, Solaris, HP-UX), 独自OS
  - CPU: PowerPC, Sparc, IA64, PA-RISC, Mシリーズ(390)

組込みに適用できる

- Xen (Citrix) http://www.xen.org/
  - ホストOS, ゲストOSとも複数種類(Linux, NetBSD)
- KVM http://www.linux-kvm.org/page/Main_Page
  - オープンソース
  - ホストOS:Linuxべったり。ゲストOS:Linuxべったり
- VirtualBox (Sun) http://www.virtualbox.org/
  - オープンソース
  - CPU:Sparc, X86, etc...
  - ホストOS:Solaris10,Windows,Linux,MacOS X
  - ゲストOS:事実上あらゆるx86ベースのOSをサポート

    Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7), DOS/Windows 3.x, Linux (2.4 and 2.6), Solaris and OpenSolaris, and OpenBSD.

本当の組込みの要件

- 実時間性
- コケてもコケない

本当の組込みの要件

- 実時間性
  - CPUパワーの配分
    - 適正な方式とは
    - 実時間性を高めすぎて、他の仮想機械が動かなくなる?!
    - スケジューリングの本質的な問題である
- Linux,Windowsが時間を取りすぎて、RTOSが動かないのは、ナンセンス
  - 公平なスケジューリングなんか糞食らえ！

本当の組込みの要件

- 実時間性
  - Linux,Windowsが時間を取りすぎて、RTOSが動かないのは、ナンセンス
    - 公平なスケジューリングなんか糞食らえ！

  とは言え…
  - LinuxやWindowsのデバイス・ドライバは仮想機械内で動いているので、それのサービスはさせないと…
    - 全部、抽象IOにしてしまうか?
    - 大型機, PCは、実時間性要請が低くていいなぁ(本当に)

本当の組込みの要件

- 実時間性
- 結局、情報系ゲストOSのユーザが、

  デバイス・ドライバをどこで動かしたいかに、大きく依存
- 情報系のデバイス・ドライバはコンソールと、低速なファイルIOデバイスだけにして頂きたい…

本当の組込みの要件

- コケてもコケない
  - ハイリライアビリティ(高信頼性)
  - ロバストネス(堅牢性)
  - ディペンダビリティ(高可用性)
  - ある仮想機械の影響は、絶対に他の仮想機械に影響してはいけない
- マイグレーション
  - 仮想機械をまるごと引越し
    - 他の実機械に
    - 仮想機械がフォールトしたら、少し前の仮想機械イメージで再開

本当の組込みの要件

- コケてもコケない対策
- マイグレーション
  - 仮想機械をまるごと引越し
    - 他の実機械に
    - 仮想機械がフォールトしたら、少し前の仮想機械イメージで再開
  - 適当なタイミングで、スナップショットを取る
    - コミット・ポイント
    - PCクラスタでは、かなり有効
      - HPC計算機は、ときどき落ちる

本当の組込みの要件

- コケてもコケない
- マイグレーション
  - 仮想機械アーキテクチャは、マイグレーションをやりやすい
  - 性質がいい。健全
  - ディペンダブルな(高可用性)組込みでは、仮想機械によるマイグレーションをするべきでしょう。

組込みのこれから

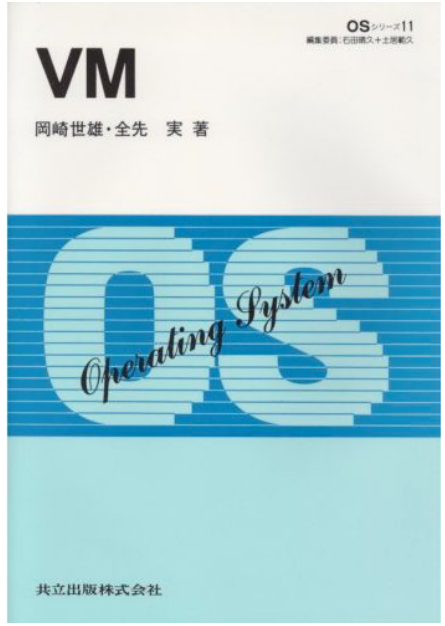
- 実時間用のスケジューリング指定APIの標準化
  - 実時間対応のパラメータとそれを設定するハイパバイザ・コール
- 割り込み処理方法の統一化(?)
  - 性能の低いCPUで、割り込み応答をする場合、どこでやるべきか?
    - 特殊なゲストOS?
    - ハイパバイザ?
    - 割り込みルーチン⇔ゲストOS ⇔アプリケーション のインターフェースはどうするか?
      - 特に、割り込み処理をハイパバイザの層でやった場合
- 今後、APIやフレームワークを決めて、プログラマが困らないようにしたい

参考文献など

- VM (OSシリーズ11)

  岡崎世雄/全先実著, 1981/AUG,共立出版,

  ISBN4-320-02405-2



- Xen

`http://www.xen.org/`

- KVM

`http://www.linux-kvm.org/page/Main_Page`

- **Kernel-based Virtual Machine (KVM) for Itanium Architecture**

`http://www.ice.gelato.org/apr07/pres_pdf/gelato_ICE07apr_kvm_yu_intel.pdf`