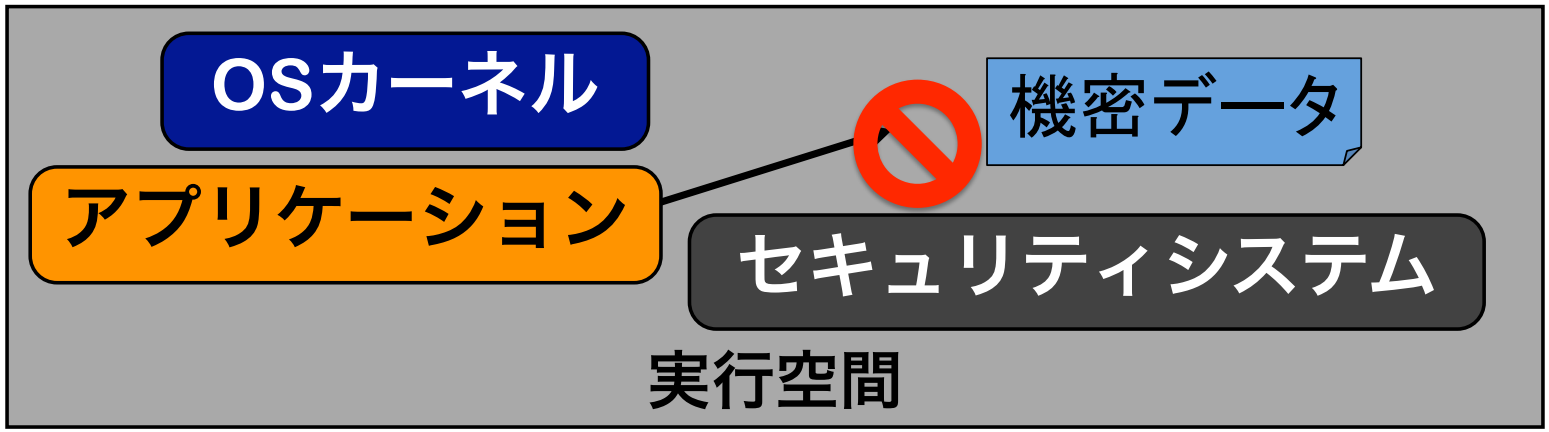
仮想マシンの外からの
アプリケーションに関連する
メモリ・ディスク上の
データの保護

尾上　浩一

アプリケーションの保護

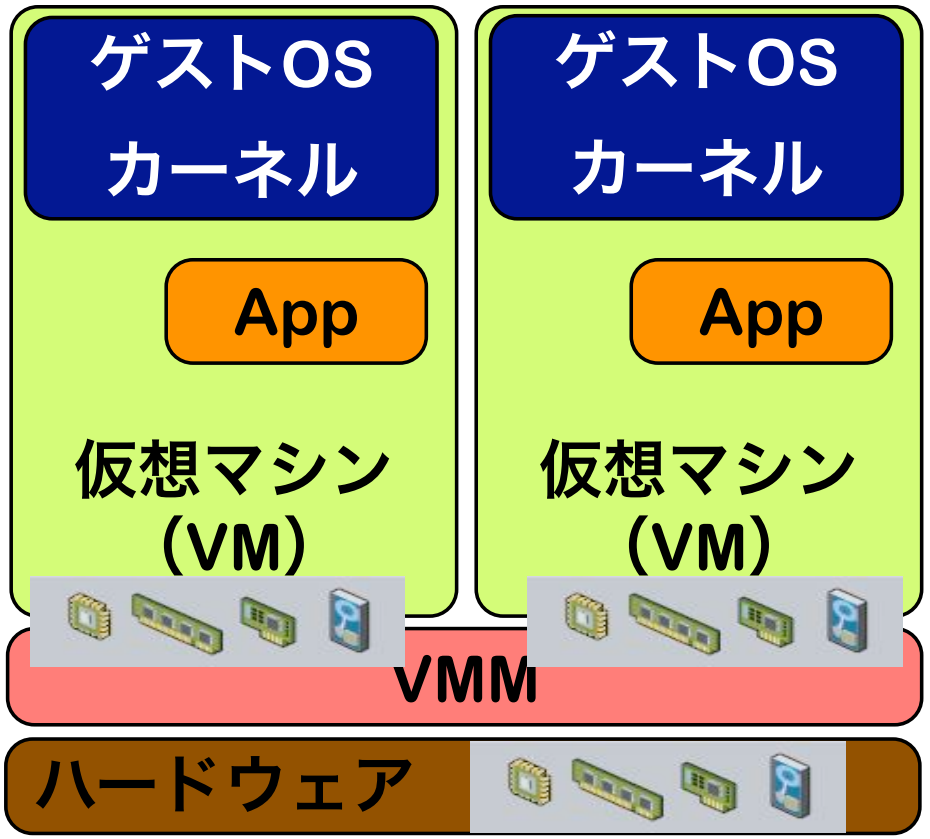
- 様々なセキュリティシステムが
  提案・開発されてきている
  - サンドボックスシステム
  - 侵入検知・防止システム（IDS•IPS）
  - アンチウィルスツール



セキュリティシステムも攻撃され得る！

仮想マシンモニタ（VMM）を利用することが有効

- VM単位の隔離
  - たとえあるVMが
    奪取されても、
    VMMや他のVMを
    奪取することが困難
- VMのメモリ・ディスク等の
  物理計算資源への操作を制御
  - VMMはVMよりも
    高い特権レベルで稼動



App:アプリケーション

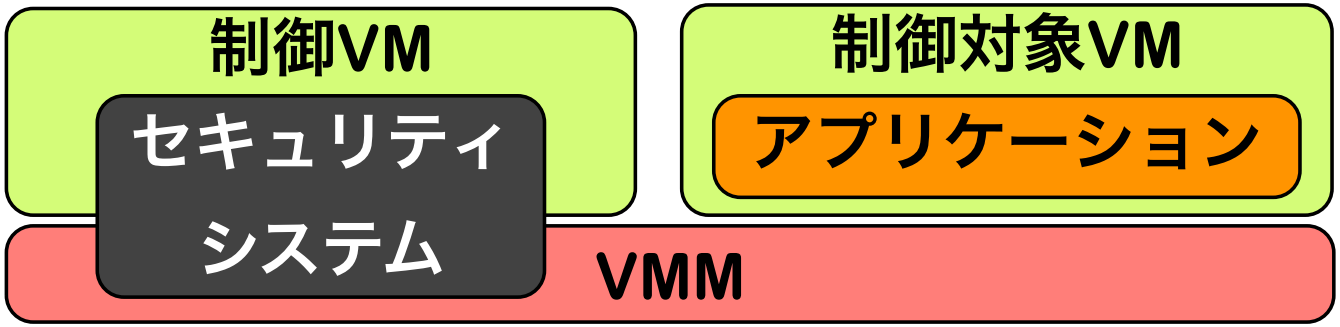
本研究の目標

- VMの外側からアプリケーションの安全性を向上させたい
  - アプリケーションの振る舞い制御
  - アプリケーションに関連するメモリ上・ファイル内のデータの漏洩・改竄を防止

本研究のアプローチ

- VMMとセキュリティシステムの協調
  - VMの外側からアプリケーションが
    発行したシステムコールの実行を
    プロセス単位で制御
  - VMMと制御VMが、アプリケーションに
    関連するメモリ・ファイル操作を制御

本日発表する内容



アプリケーションに関連するメモリ・ファイル操作の制御

既存のメモリ・ディスク上のデータを保護するセキュリティシステムは十分ではない

- OSカーネルや管理者権限で稼動するプログラムが信頼できるという仮定がある
- OSカーネルを含むプログラムは乗っ取られ得る
  - プログラム自体やそれらが利用するライブラリには脆弱性が存在する

メモリ・ディスク上のデータを操作できるのであれば．．．

- プログラムの脆弱性の有無に関わらず、プログラムの乗っ取りやそれらに関連するデータの漏洩させたり、改竄できる
  - Control data, Non-control data
- アプリケーションが正常に稼動しているように偽装するために悪用できる
  - cf.) Mimicry attacks

メモリ上・ファイル内のデータを保護するには？

- VM単位の隔離を利用する手法が考えられる
  - VMの外側で保護したいアプリケーションを稼動させる (cf. Terra)
    - ✓VM単位の計算資源を消費してしまう
  - 安全に実行したいプログラムコンポーネントのみVMの外側で実行する (cf. Nizza)
    - ✓プログラムのソースコードを書き換えることが必要になってしまう

メモリ上・ファイル内のデータの保護に関する目標

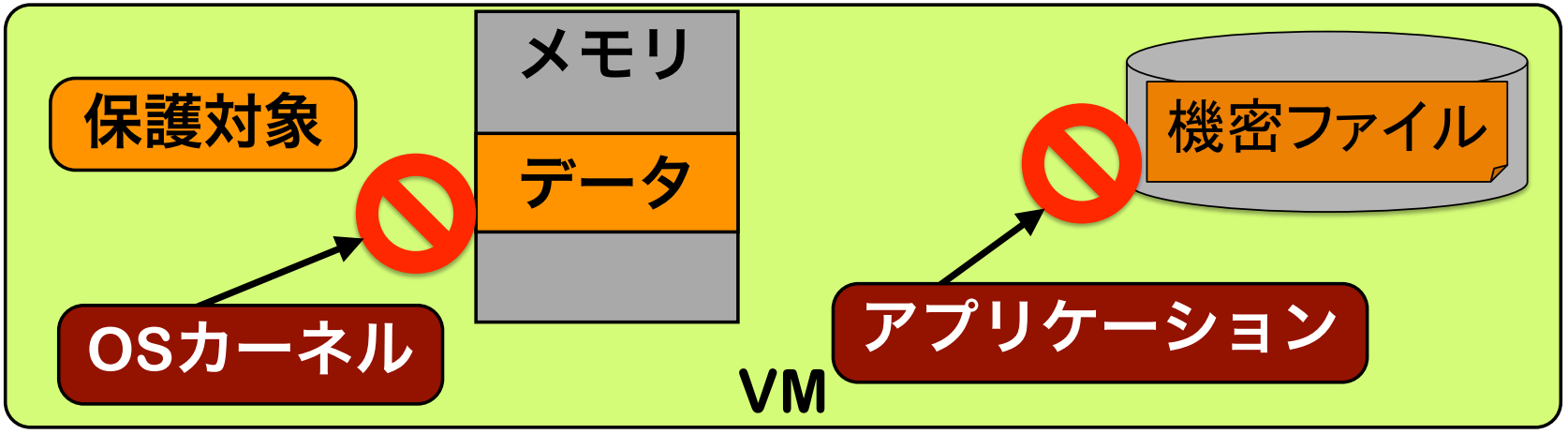
- 保護したいアプリケーション（保護対象）に関連するメモリ・ディスク上のデータの漏洩・改竄を防止する
- 信頼できないOSカーネルを含むプログラムと同じVM内で保護対象を稼動させる
- 保護対象のソースコードを修正することなく適用できる

どのような攻撃を防ごうとしているのか？

- メモリ上・ファイル内のデータの保護に関して
  - 本研究の対象である攻撃
  - 本研究の対象ではない攻撃

本研究の対象である攻撃

- 保護対象のユーザ空間領域と保護対象に関連するファイル内のデータを漏洩させたり、改竄する攻撃
  - デバッガやカーネルモジュール等を利用した攻撃
  - シンボリックリンクを利用した攻撃
  - レースコンディションを利用した攻撃



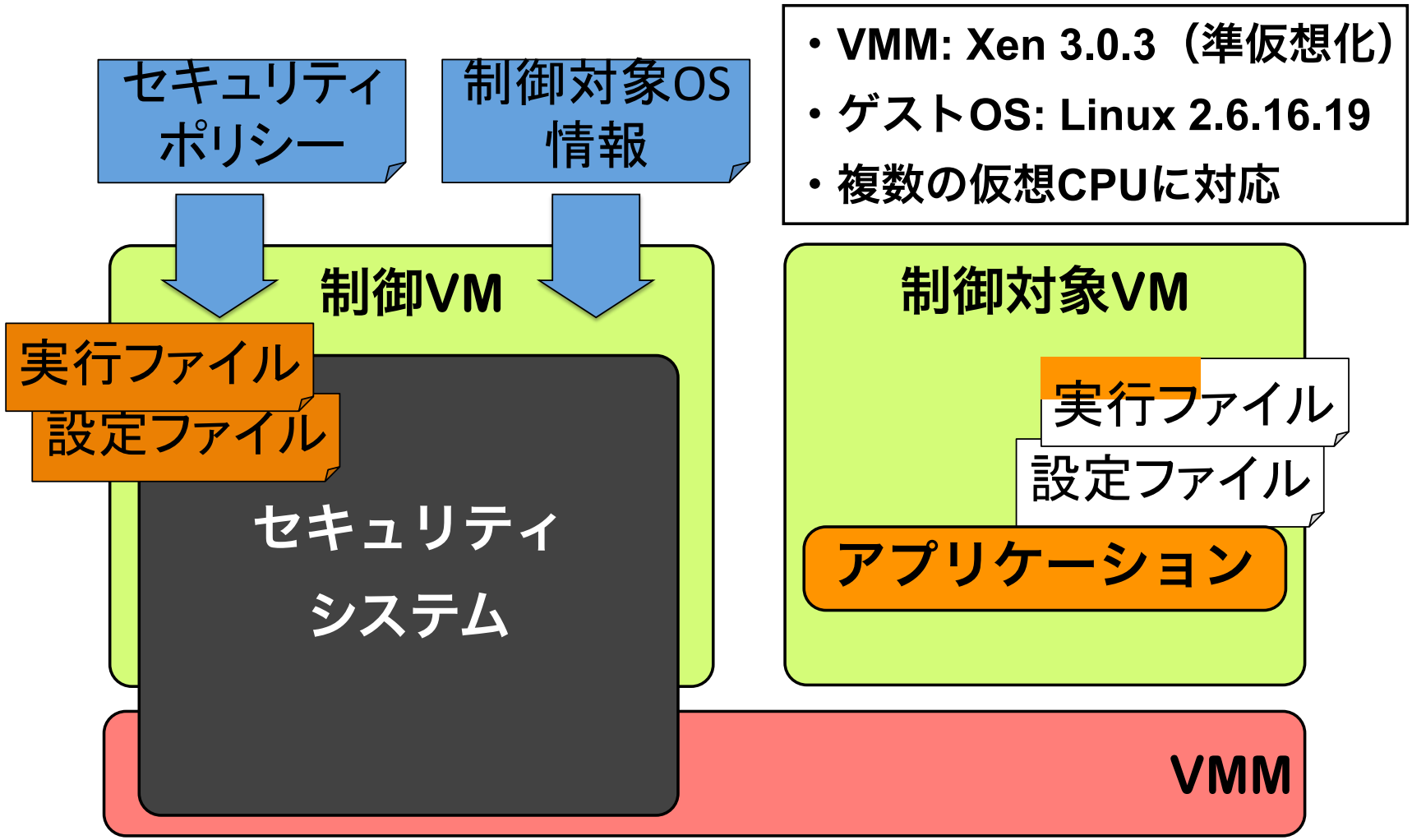
本研究の対象ではない攻撃

- 保護対象自体の脆弱性を利用した攻撃
  - バッファオーバーフローを利用した攻撃
- カーネル空間領域のデータを改竄する攻撃
  - 保護対象をスケジューリング対象から外すような攻撃

メモリ上・ファイル内のデータを保護するためのアプローチ

- 保護対象のメモリ上・ファイル内のデータの実体を制御対象外から隠蔽
  - メモリ上のデータ（コード、データ領域等）
    - 動作レベル（カーネル・ユーザ）ごとに物理メモリ領域を多重化
  - ファイル内のデータ（実行ファイル・設定ファイル等）
    - 異なるVMで管理

提案システム


セキュリティ
ポリシー
制御対象OS
情報
・VMM: Xen 3.0.3（準仮想化）
・ゲストOS: Linux 2.6.16.19
・複数の仮想CPUに対応
制御VM
実行ファイル
設定ファイル
セキュリティ
システム
制御対象VM
実行ファイル
設定ファイル
アプリケーション
VMM

提案システムの運用

- 制御対象VMの利用者が指定した
  アプリケーション（保護対象）のみ保護する
  - 保護対象に関するデータの保護の設定は
    制御VMから実行する
  - 保護対象の実行を開始するためには、
    保護対象VM内の実行ファイルを用いる
  - どのように保護対象に関するデータを
    保護するかはセキュリティポリシーで指定する

セキュリティポリシー

- メモリ上のどの部分のデータを保護するか
  - すべてのデータか、一部のデータか
- どのファイル（保護対象ファイル）を保護するか
  - 制御VM内と制御対象VM内の対応関係
- ポリシー文法（別紙参照）

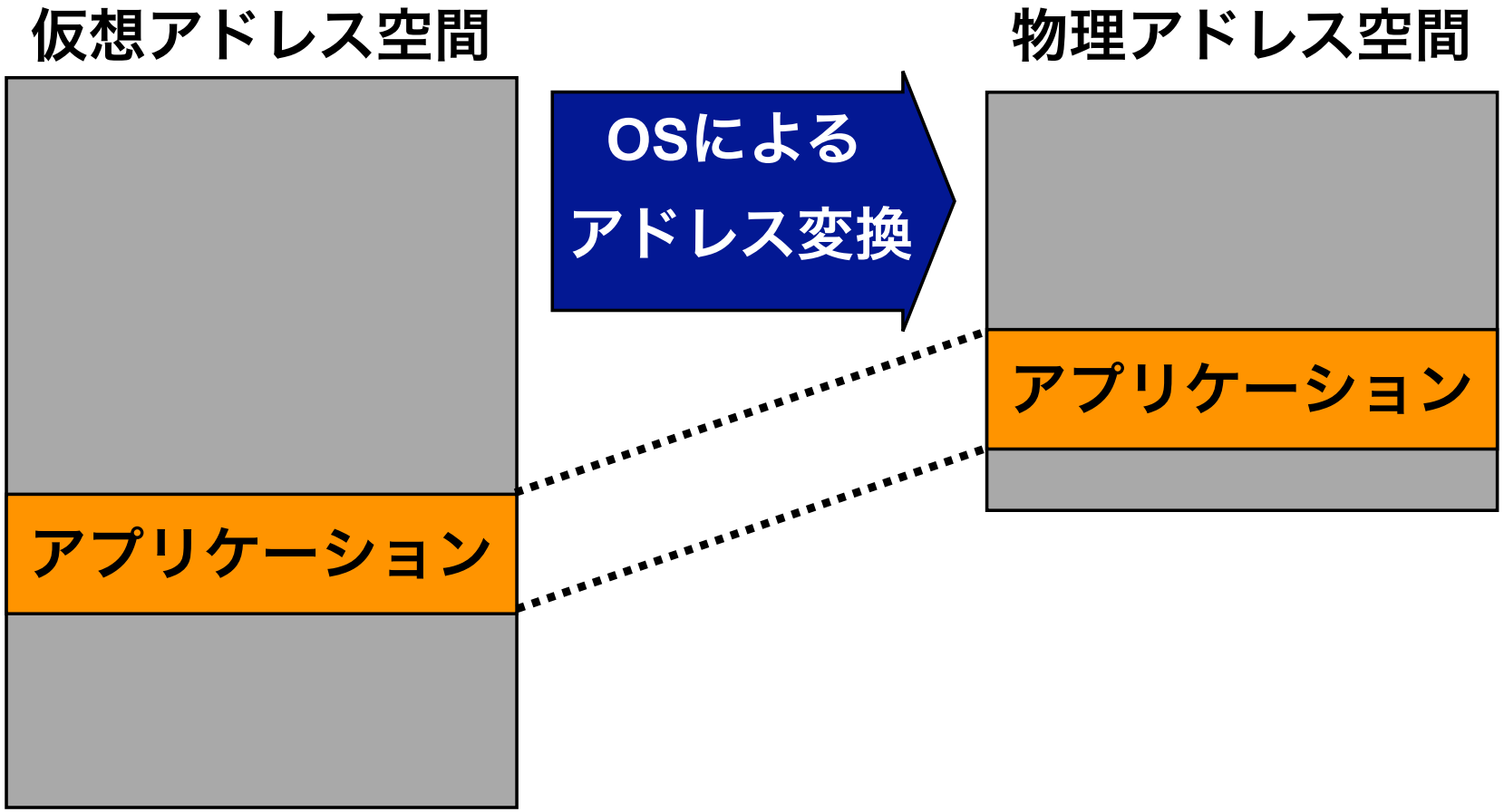
セキュリティポリシーの記述例 (1/2)

executable : *./test1_outsidevm*
shadowMemoryFile : *all*
shadowFile :
    pathPrefix(*/home/A_in/*, *./A_out/*) <*read*>

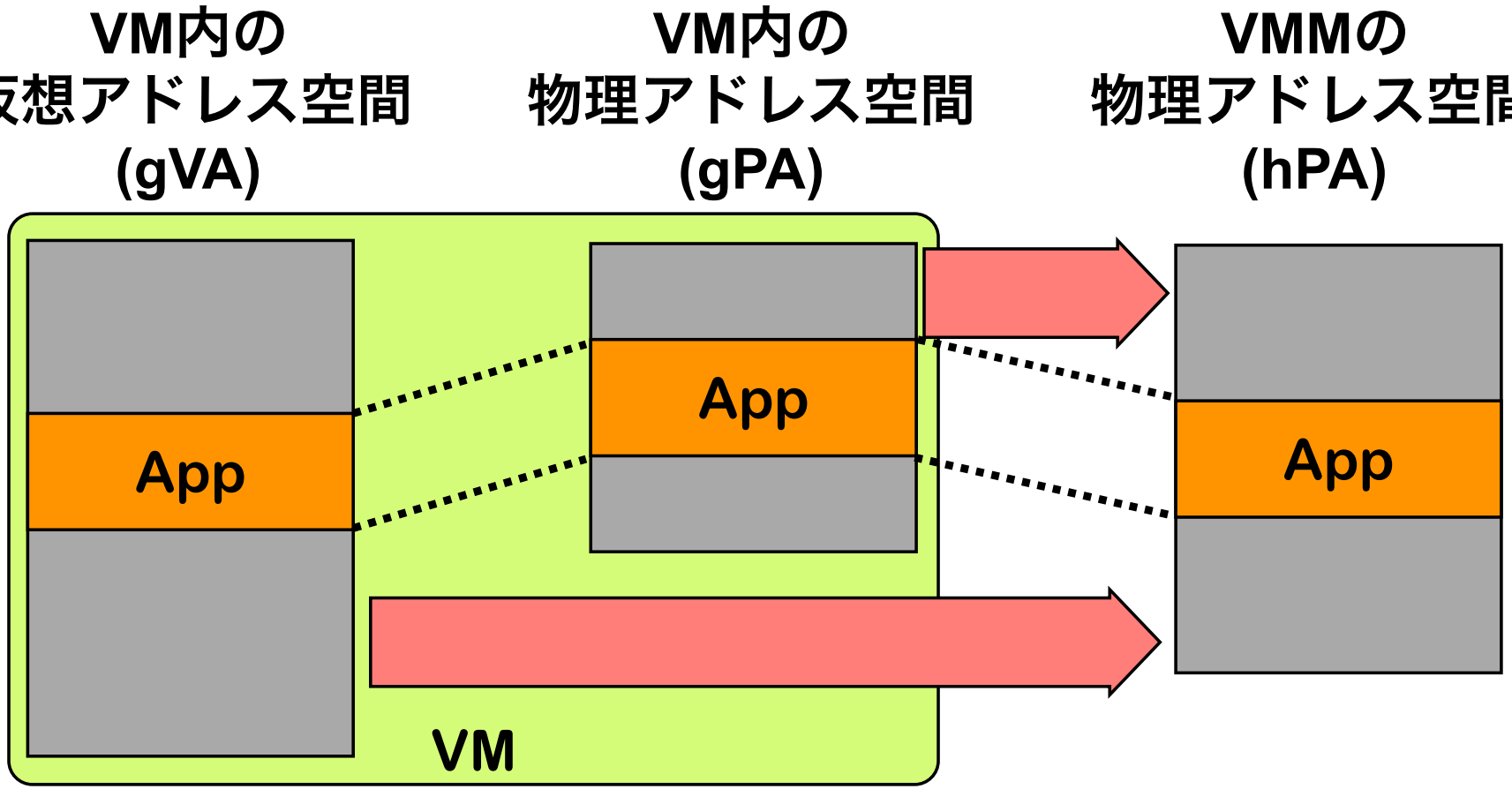
セキュリティポリシーの記述例 (2/2)

executable : *./test2_outsidevm*
shadowMemoryFile : *partially*
    kernelSeg : *data, brk, stack(DOWN, 1GB), env, arg, mmapRegion*
    readOnlyRegions : *text*
shadowFile :
    pathName(*/home/B/in.cfg*, *./B/out.cfg* <*read*>

OSによるメモリ管理



VMMによるメモリ管理



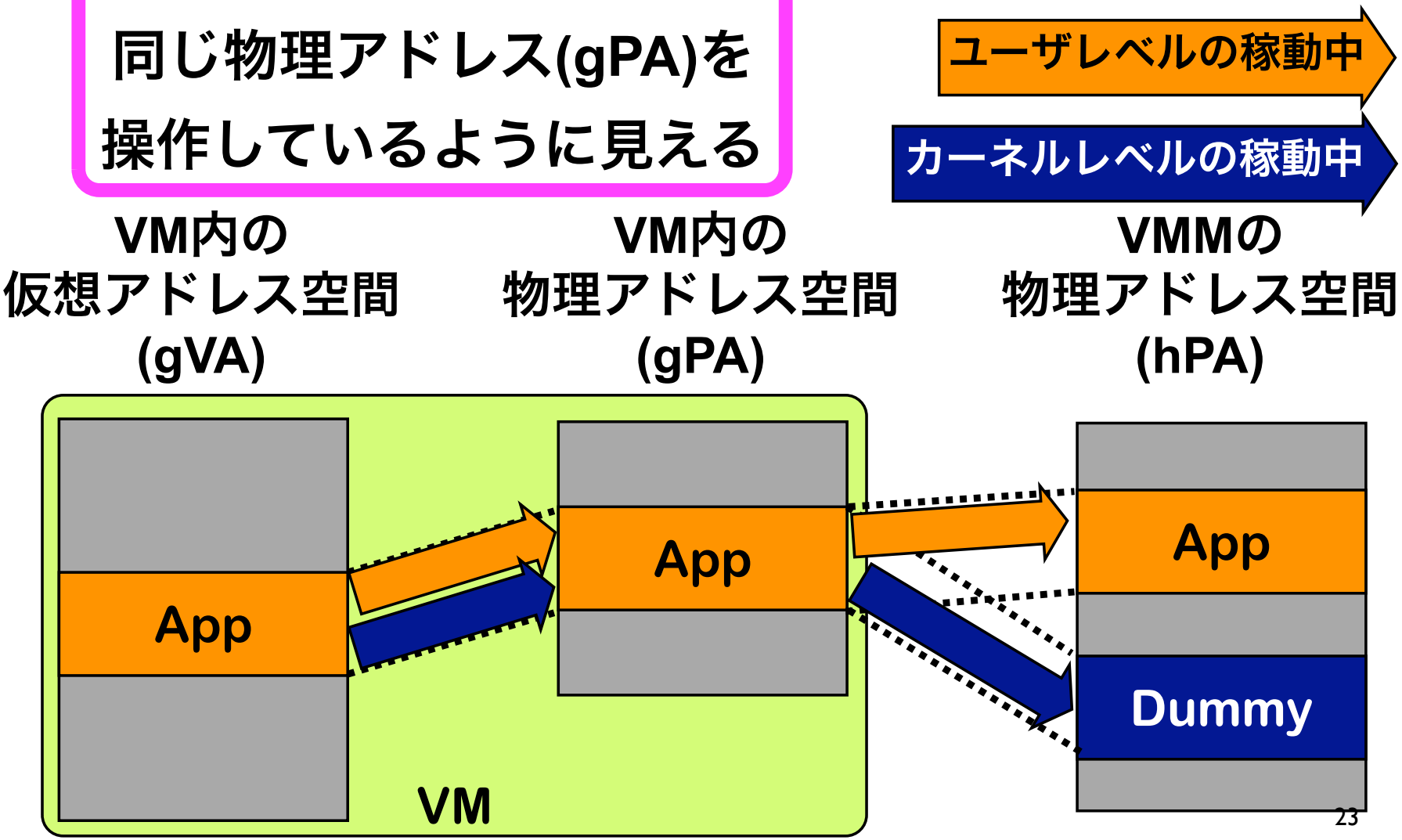
App:アプリケーション

メモリ上のデータの多重化

VM内のOSからは
同じ物理アドレス(gPA)を
操作しているように見える

ユーザレベルの稼動中

カーネルレベルの稼動中



VMMによるメモリの多重化処理

- 多重化するメモリ領域情報の管理
- 多重化処理の初期化
  - 保護対象の実行開始時に行う
- 多重化するメモリ領域の更新

多重化するメモリ領域の更新 (1/2)

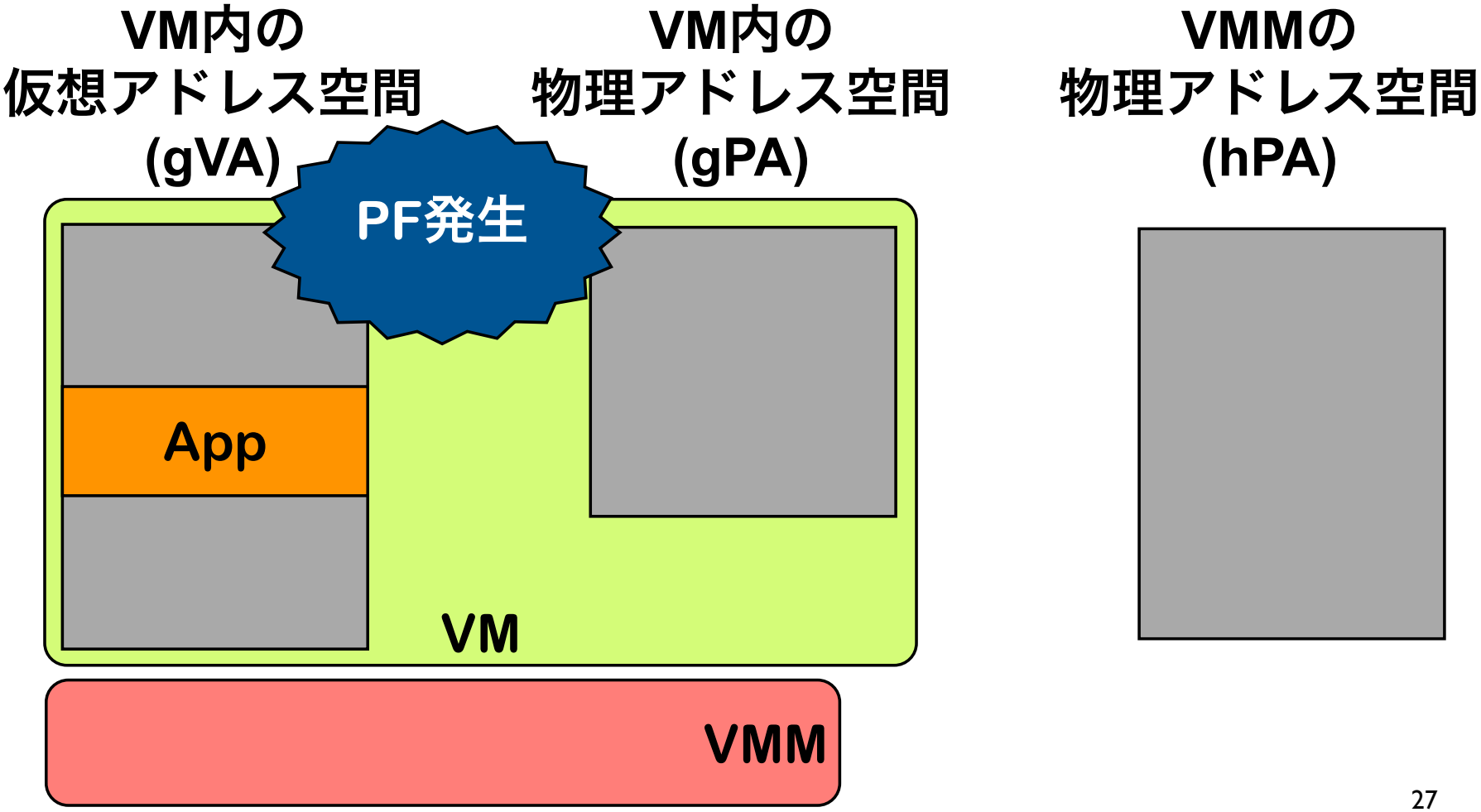
- いつ更新処理を行うか？
  - ページフォルト(PF)が発生したとき
  - ゲストOSカーネルによるVMM呼び出し（Hypercall）経由でページテーブルが更新されるとき

多重化するメモリ領域の更新 (2/2)

- 更新時にはどのような処理が必要か？
  - ページフォルトやHypercallの実行前後で、ページテーブルエントリの中身がどのように遷移するかにより決定する
    - 新たなページの確保、Copy-on-Write、ページイン・ページアウト等

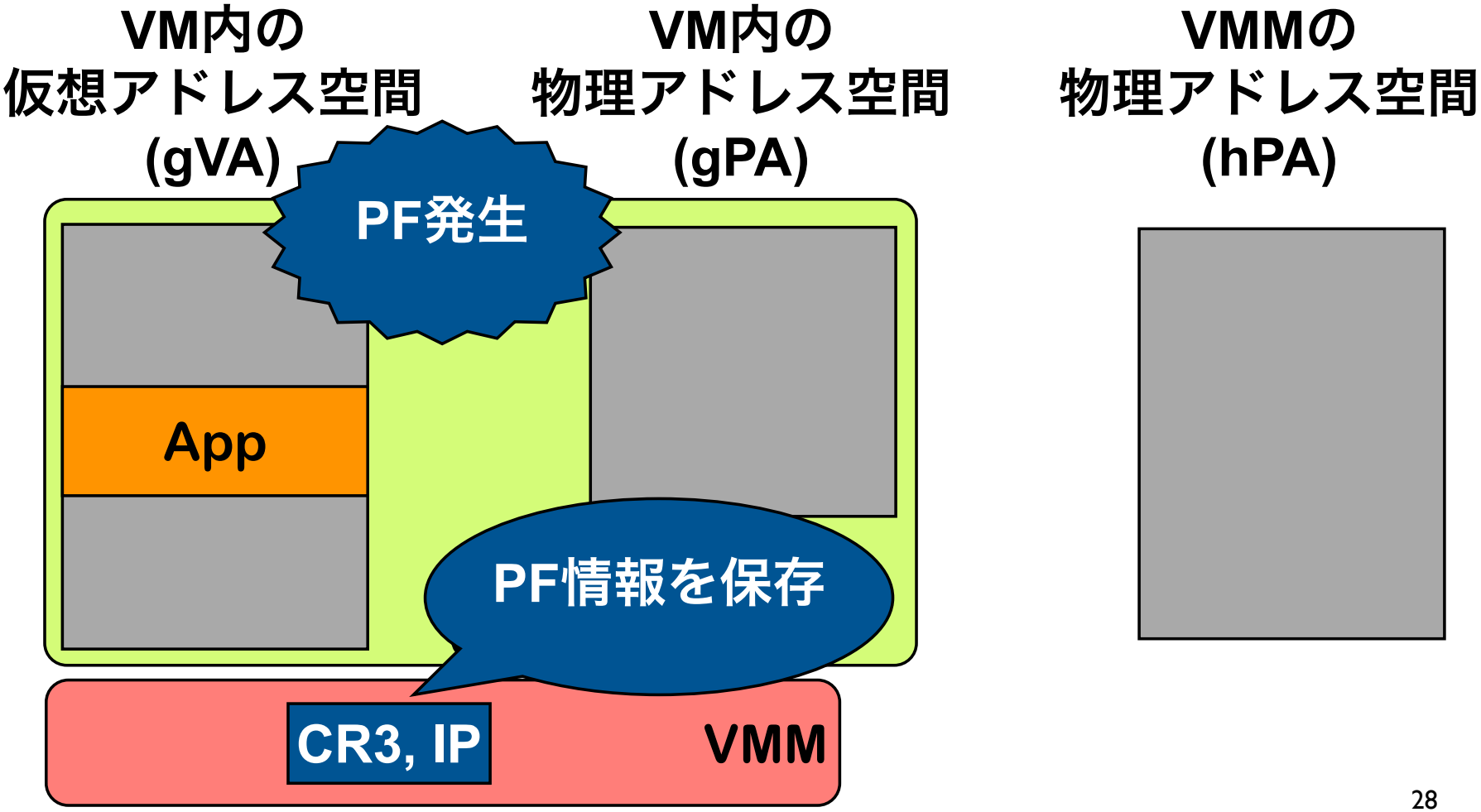
多重化するメモリ領域の更新：

新たなページを確保する場合



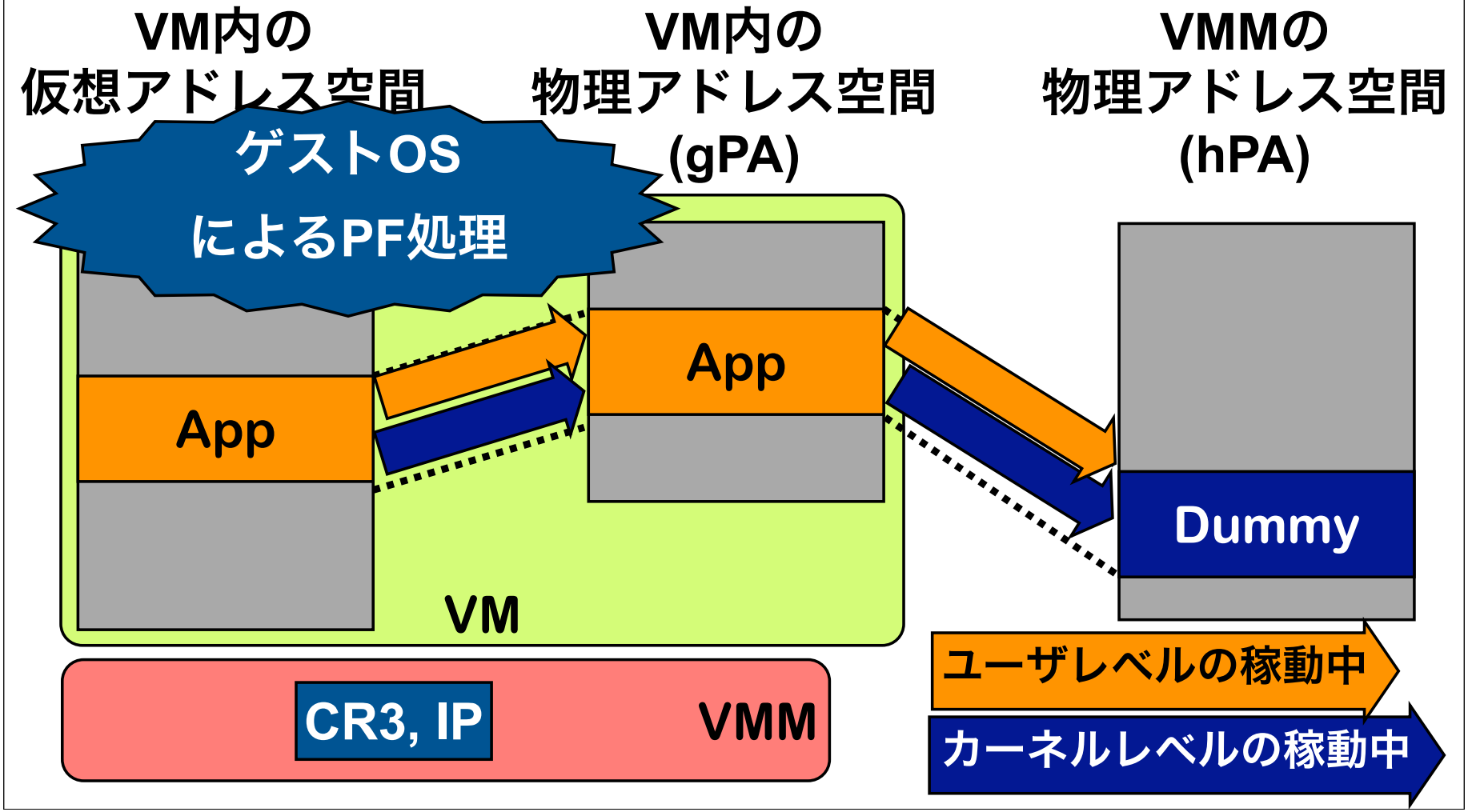
多重化するメモリ領域の更新：

新たなページを確保する場合



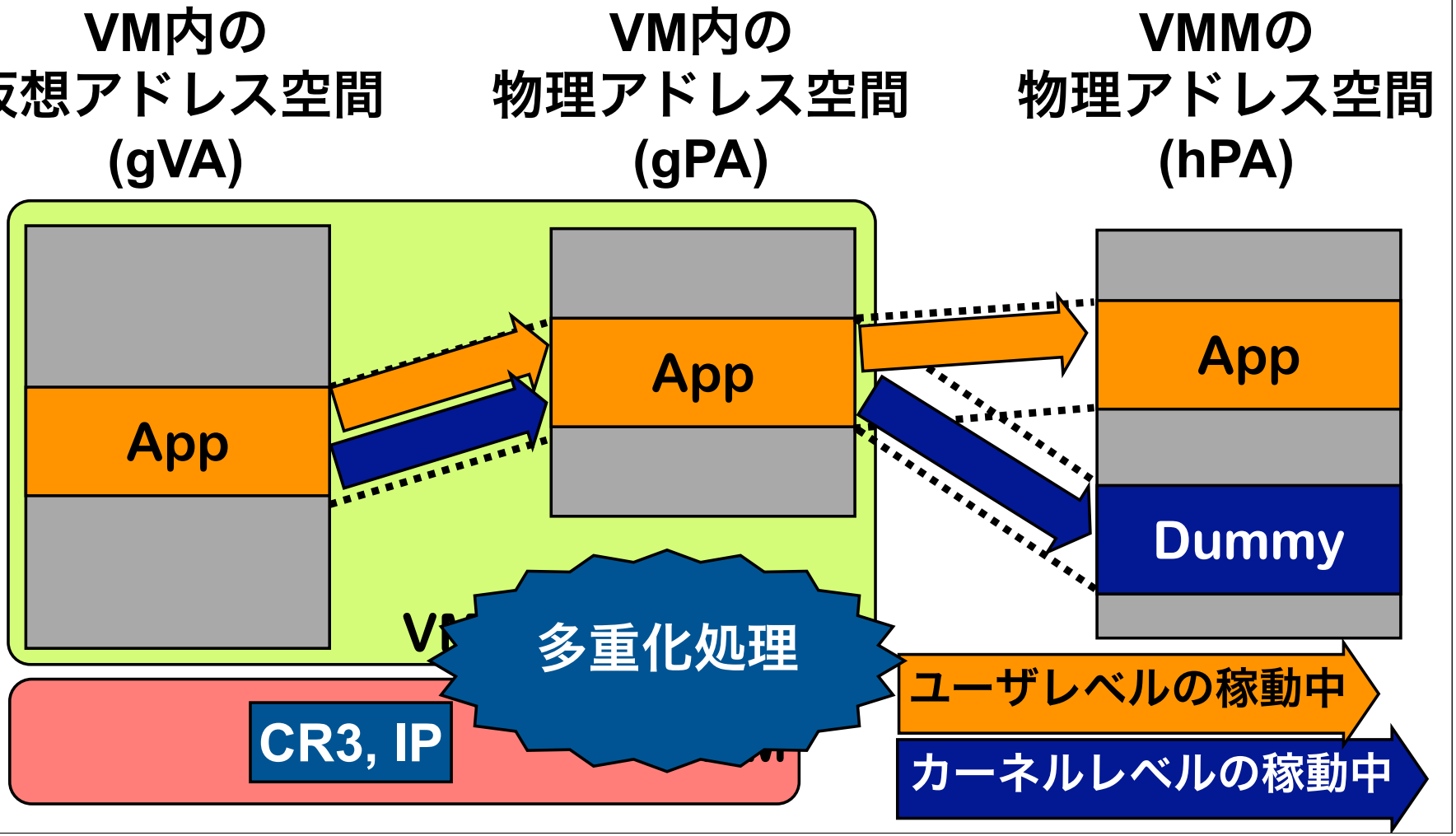
多重化するメモリ領域の更新：

新たなページを確保する場合



多重化するメモリ領域の更新：

新たなページを確保する場合



いつ、動作レベルの切り換えが発生するか？

- 割り込み・例外処理
- システムコール処理
- 動作レベルの切り換えを捕捉し、
  動作レベルごとに
  操作する物理メモリ領域を変更する

動作レベルの切換の捕捉

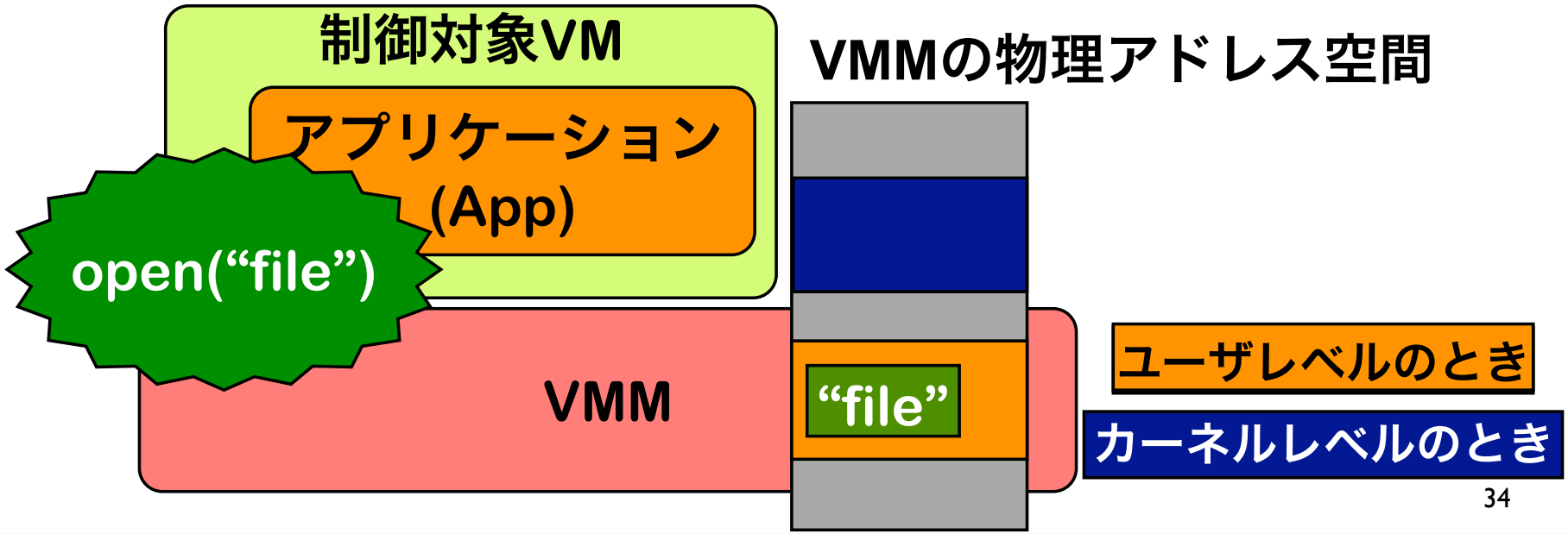
- 必ずしも、適当なときにVMMで動作レベルの切り換えが捕捉できるわけではない
  - ex.) システムコールの終了時
- 提案システムでは
  バイナリ書き換えの仕組みを導入
  - 任意の命令をVMMで捕捉できるようになる

システムコール処理

- ゲストOSカーネル内で行われる処理と
  整合性を保ち、多重化することが必要となる
  - ユーザ空間へのポインタ引数
  - 保護対象の実行の開始・終了
  - プロセス生成
  - ヒープ領域・メモリマップ領域
  - シグナル処理

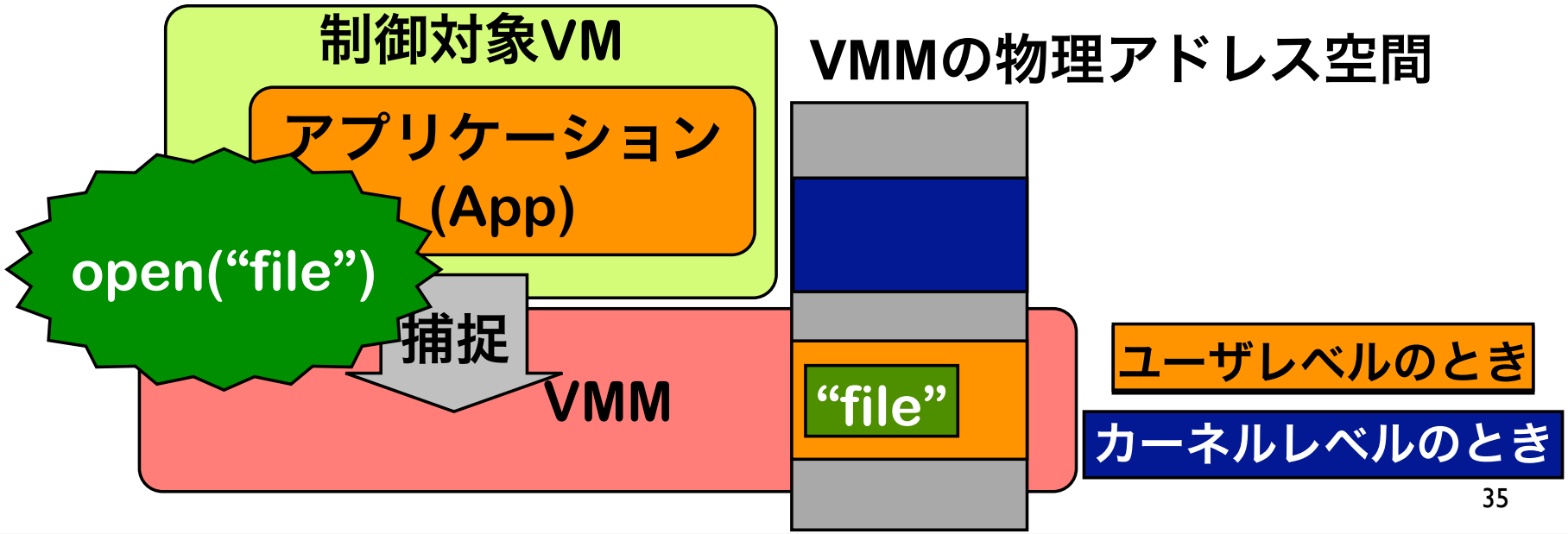
ユーザ空間へのポインタ引数に関する処理 (1/2)

- 一時的にゲストOSカーネルから正しいデータが操作できるようにする
  - システムコールの開始から終了時までの間



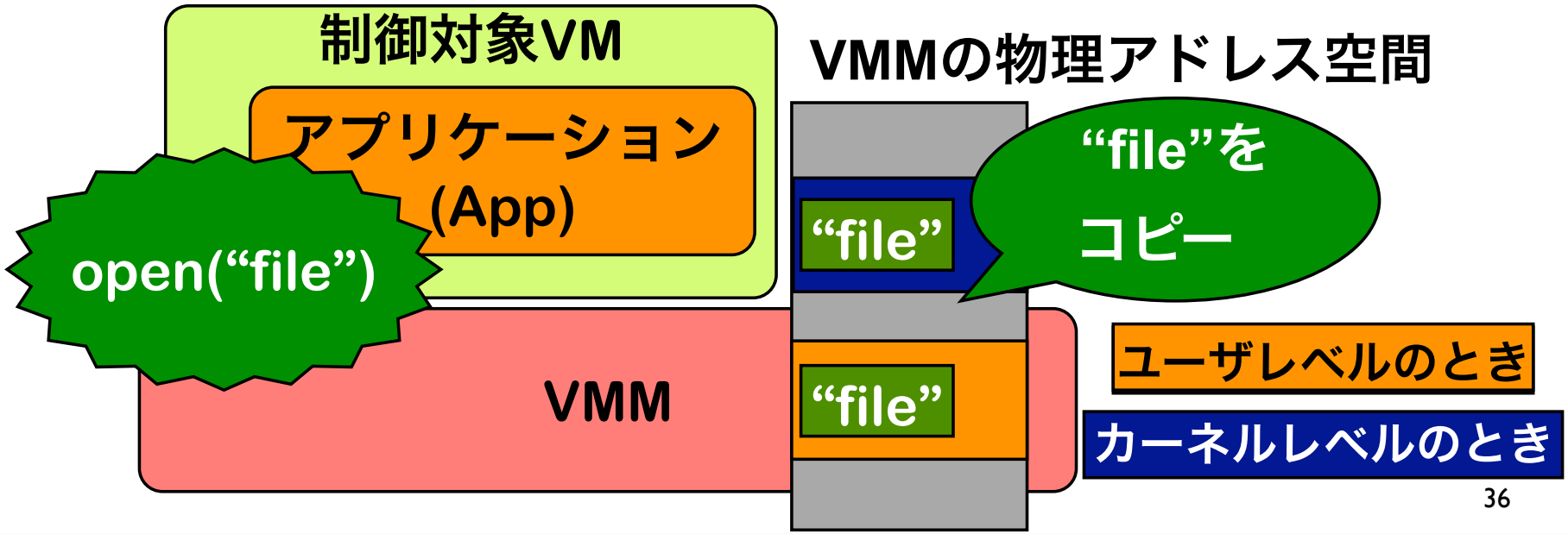
ユーザ空間へのポインタ引数に関する処理 (1/2)

- 一時的にゲストOSカーネルから正しいデータが操作できるようにする
  - システムコールの開始から終了時までの間



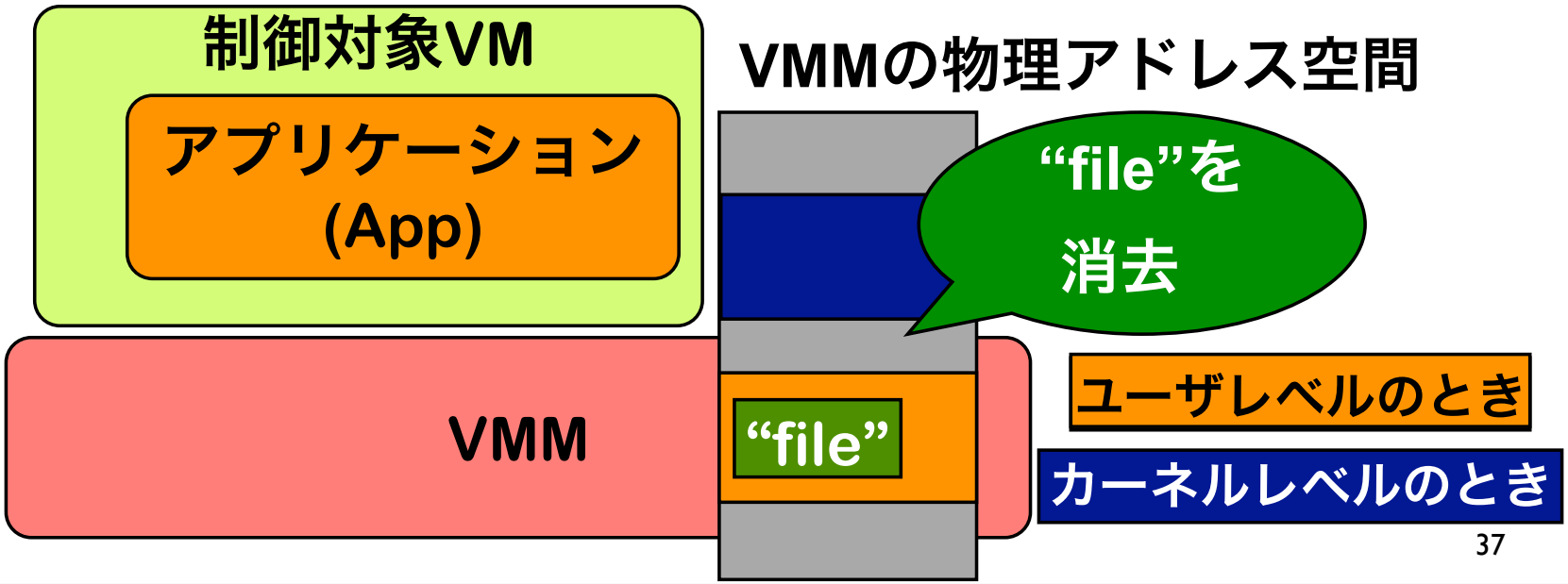
ユーザ空間へのポインタ引数に関する処理 (1/2)

- 一時的にゲストOSカーネルから正しいデータが操作できるようにする
  - システムコールの開始から終了時までの間



ユーザ空間へのポインタ引数に関する処理 (1/2)

- 一時的にゲストOSカーネルから正しいデータが操作できるようにする
  - システムコールの開始から終了時までの間

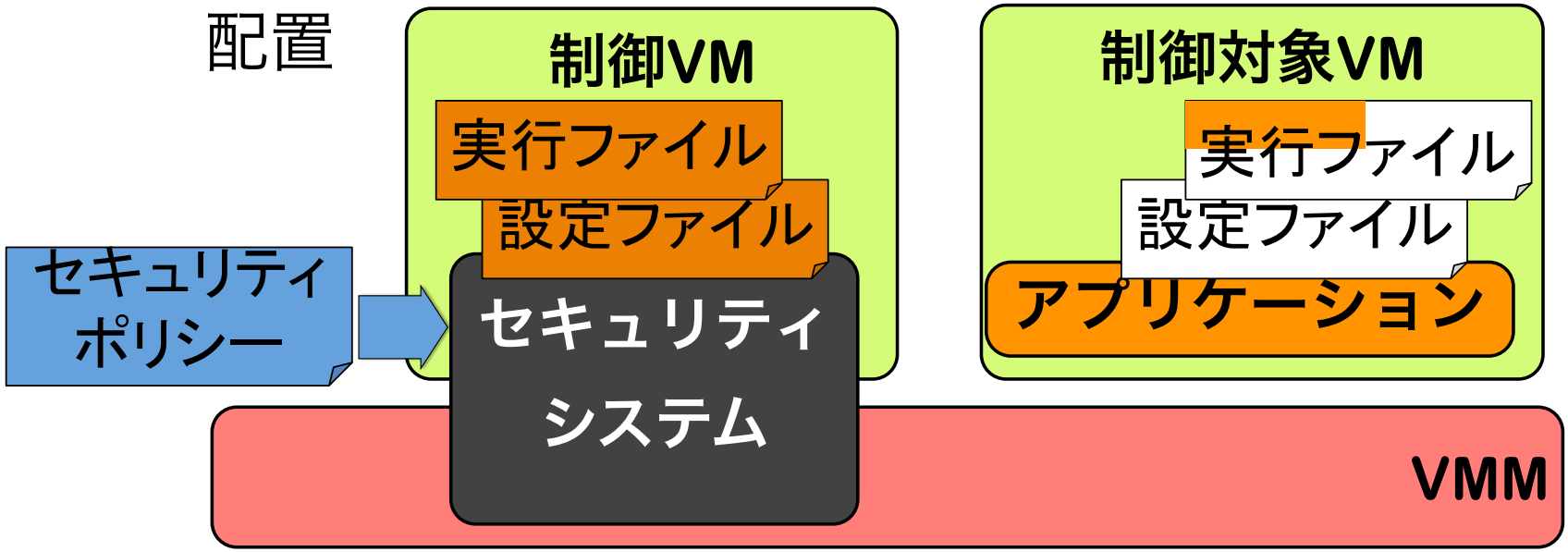


ユーザ空間へのポインタ引数に関する処理 (2/2)

- VMMが各システムコールごとに
  必要なデータ操作を行う
  - 操作に必要な各システムコールに関する情報が制御対象OS情報に含まれる
    - システムコール番号、ポインタ引数番号、ポインタ引数の大きさ

異なるVMでのファイルの管理

- 制御VMで保護対象ファイルの実体を管理
  - 保護対象ファイルはセキュリティポリシーで指定
  - 制御対象VM内には保護対象ファイルのdummyを配置



保護対象ファイルのDummy

- 保護対象固有のファイル
  - 実行ファイルの中身
    - 一部を残して、残りの部分をゼロクリア
  - その他（設定ファイル等）の中身
    - すべての部分をゼロクリア
- 他のプログラムと共有するファイル（libc等）
  - そのまま

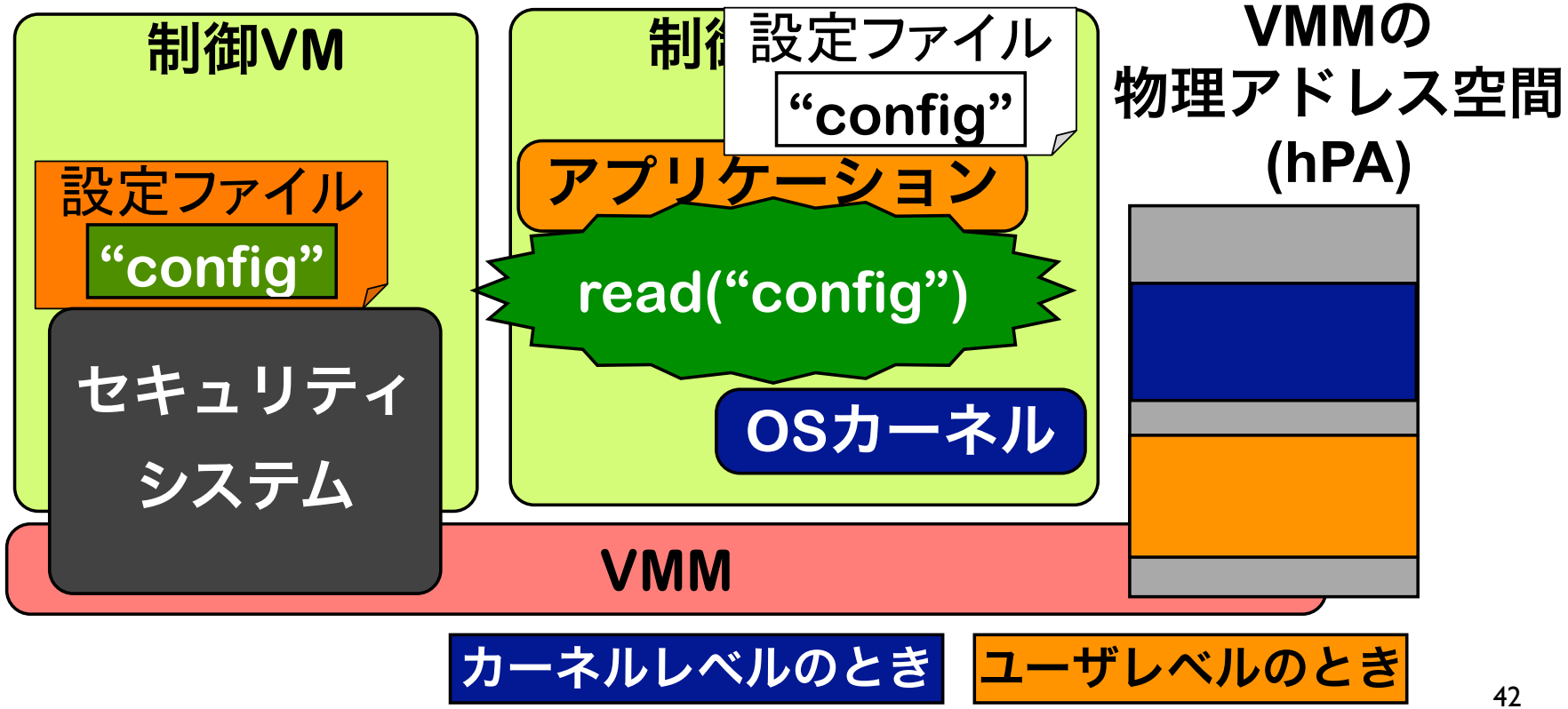
実行ファイルのDummy

- 対象の実行ファイルはELF形式の実行ファイル
- 保護対象の実行開始のプログラムのロード時に一部の実行ファイルの情報が必要になる
  - ELFヘッダ、プログラムヘッダ、”.interp”セクション
- 提案システムでは、上記以外をゼロクリアするプログラムを提供している

ファイル操作のエミュレート

VMMが直接、ユーザレベルの稼働時に操作する

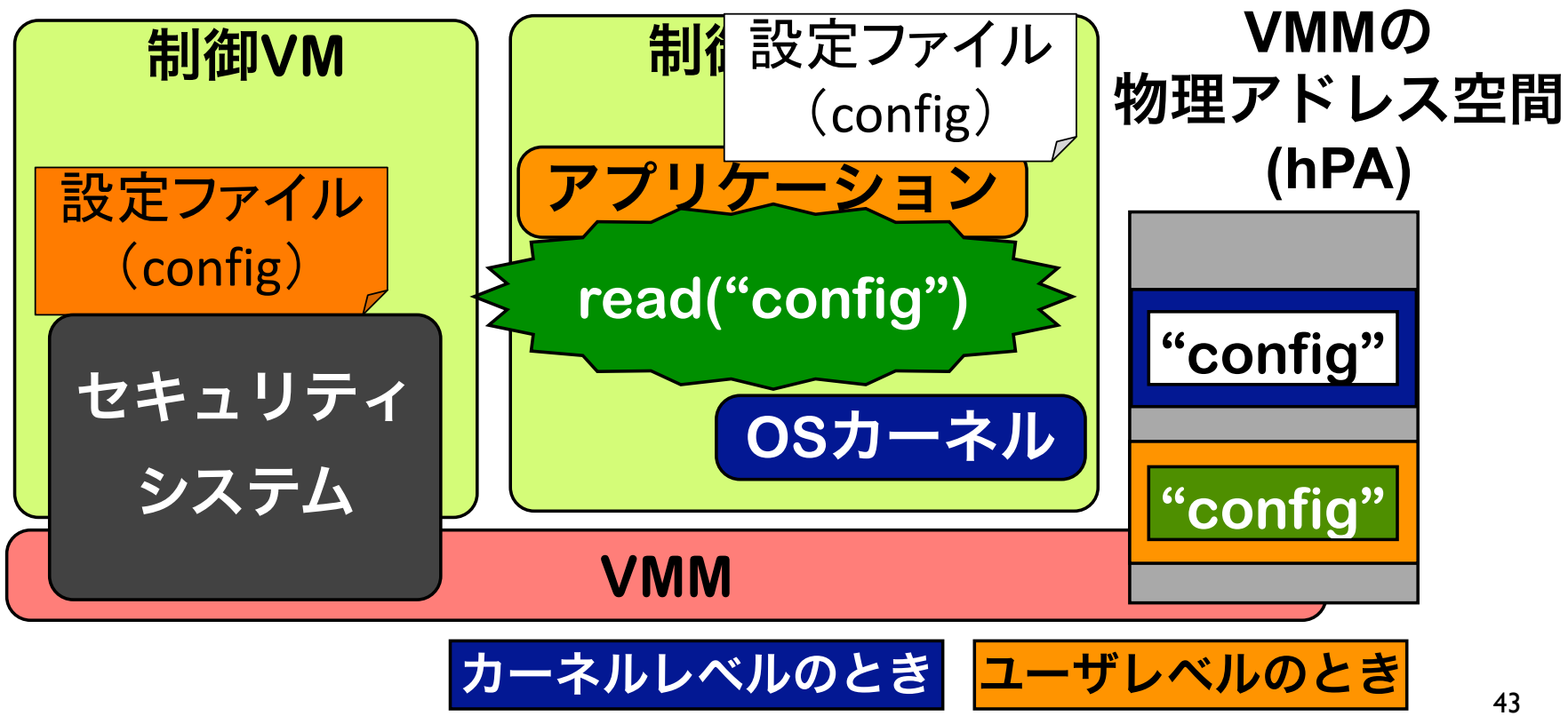
hPAに“config”を配置する



ファイル操作のエミュレート

VMMが直接、ユーザレベルの稼働時に操作する

hPAに“config”を配置する



ファイルの保護は
VMM+NFSでは十分でない

- 制御対象OSカーネルがデータを不正に操作できる
  - 制御対象VM内のNFSクライアントによる
    データ転送が制御対象OSカーネル内の処理を
    経由してしまう
  - プログラム単位の制御ではなく、
    ユーザID単位の制御になってしまう

提案システム評価

- メモリ上・ファイルのデータの保護の確認
  - Webサーバ thttpd
  - アンチウィルスツール ClamAV
- 実行時のオーバーヘッドの計測
  - マイクロベンチマーク
  - アプリケーションベンチマーク
    - thttpd, ClamAV

実験環境

- CPU: Dual-Core AMD Opteron 2.8GHz 2つ
- 8GBメモリ、1Gbps NIC
- 制御VM・制御対象VM
  - 仮想CPU: 4つ
  - 1GBメモリ

データ保護の確認: thttpd

- thttpdでchroot jail機能を有効にして稼動
  - /var/wwwをルートディレクトリに設定
    - /usr/bin/perlへのアクセスが制限され、CGIプログラムが実行できない
- 攻撃の模擬
  - chroot jail機能を一時的に無効にし、CGIプログラムを実行できるようにする

chroot jail機能の一時的な無効化

- thttpdを再起動
- GDBを用いて、変数*do_chroot*の改竄
  - *do_chroot*の検査前にthttpdを一時停止
  - *do_chroot*をクリアし、実行の再開

✓この時点で、CGIの実行ができるようになる

- CGIを実行後、thttpdを再起動
  - /var/log/syslogからthttpdの再起動と chroot jailの無効化に関するログを消去

chroot jail機能の一時的に無効化する対する攻撃からの保護(1/2)

- thttpdのメモリ上のデータを多重化
- thttpdの設定ファイルを制御VMで管理
- /var/log/syslogを制御VMで管理
  - syslogdのメモリ上のデータも多重化

chroot jail機能の一時的に無効化する対する攻撃からの保護 (2/2)

- 再度、同じ攻撃を模擬
  - GDBで一時停止のブレークポイントを設定できない
  - thttpdの設定ファイル、/var/log/syslogを改竄できない

マイクロベンチマーク

- システムコールを1000回実行し、
  1回あたりの実行時間を掲載（別紙表1参照）
- 物理計算機上(Linux)、Xen上と比較
- システムコール捕捉処理やプロセスの
  識別処理が追加された場合(non-target)も計測

アプリケーションベンチマーク

- thttpdに対し、httperfを用いて、同一LAN内から接続数128で、サイズが各々1KB・100KBの場合のファイル参照に伴うスループットを計測
- clamscanで15のファイル（ウィルスファイル5つ）のウィルス検査に要する時間を計測
  - ウィルスデータベースが保護対象内（VDB-inVM）と保護対象外（VDB-inVM）の場合を計測

関連研究（1/2）

- Overshadow [Chen et al., 2008]
  - メモリ上・ファイル内のデータを
    暗号化によりデータを保護
  - 複数の仮想CPUを利用する場合に、
    複合化されたデータが見えてしまう
  - メモリ上・ファイル内のデータは改竄できる
  - 専用のユーザプログラムが必要である

関連研究（2/2）

- [Rosenblum et al., 2008]
  - 命令参照かデータ参照かに基づき、

    メモリ上のデータを多重化する
  - 提案システムは動作レベルに基づいて保護する
  - ファイル内のデータは保護できない
- Proxos [Ta-Min et al., 2006]
  - 保護対象を異なるVM（制御VM）で稼動させ、

    保護したい操作のみ制御VMで実行する
  - 提案システムは保護対象を保護対象VMで実行する