Linux開発動向と ファイルI/Oの最新技術

OSDN セミナー
Linux Kernel Conference

2002年10月10日

VA Linux Systems Japan

高橋　浩和

taka@valinux.co.jp

目次

- Linuxカーネルサミット報告
  - エンタープライズ用途へ向けて
- V2.5における新しいファイルI/Oの仕組み
  - ページI/O
  - ダイレクトI/O
  - 非同期I/O (AIO)
  - VMとの統合
  - マルチプロセッサ対応強化
  - ブロックI/O改善
    - HighMemの扱い

VA LINUX SYSTEMS JAPAN

カーネルサミット報告

Linuxカーネルサミット報告

- Linux kernel developers summit
- OTTAWA Linux symposium
- 2002/6E カナダのオタワにて開催
- エンタープライズ向け拡張
- 企業の役割の増大
- V2.5 feature fixはハロウィーン(10月末)
  - Linus氏の都合で10/20に前倒し

VA LINUX SYSTEMS JAPAN

カーネル開発者

- Linus氏を頂点とする緩いピラミッド構造は健在
- 殆どのLinuxコア開発者は、企業の中で仕事として開発を行っている。
- カーネルサミット、カンファレンスを通してIBMの勢力が非常に伸びているのが見て取れる
  - 企業からの発表のうち半分はIBM。北米、欧州、豪州、インドの各拠点で活動
  - 日本の姿は殆ど見られない

VA LINUX SYSTEMS JAPAN

カーネルパラメータ

- 現在、カーネルにパラメータを与える方法は複数存在し、各々連携していない。カーネル内で統一的に扱えるべき。
  - /proc
  - モジュールパラメータ
  - bootパラメータ
- driverfsの導入。各デバイスの無秩序な/procを一元的なdriverfsにまとめられないか？
  - バス情報、ドライバクラス情報、デバイスドライバ情報などを元に階層構造を構成
  - ソフトウェアドライバも含められないか？

VA LINUX SYSTEMS JAPAN

空間管理の改良

- Rik van Riel氏とAndrea Arcangeli氏のＶＭ機能双方がv2.5に取り込まれることでなった
- メモリZONE間でのバランシング
- Kmap領域の多重仮想化
- ページテーブルの多重仮想空間へのマップ
- ページのカラーリング
- ＮＵＭＡ対応
- ファイルシステムとの連携強化
- I/O時のクラスタリング化、ブロック遅延確保
- Slab,dentry,inode解放とページ解放の連携

VA LINUX SYSTEMS JAPAN

ブロックI/Oの改良

ブロックI/Oは、現在のLinuxのスケーラビリティのネック

- I/Oオーダ制御。非同期であってもI/O順序制御ができる仕組み
- 複数ページへのI/O
- Dead Lineスケジュール
  - Deadline 1秒で3%の性能低下、100msecで8%の低下
  - I/O優先度の導入は？

VA LINUX SYSTEMS JAPAN

セキュリティモジュールLSM

UID/GID制御を超えた細かいアクセス制御

- カーネル内に細かいHook
- 利用用途によっては有効

VA LINUX SYSTEMS JAPAN

非同期I/O　（AIO）

I/O要求発行とI/O完了待ち合わせ処理の分離

- スレッド以外で、並列I/Oを行うための仕組み
- File操作テーブルにaio_read/ aio_write/ aio_fsync操作
- 現在のwaitキューにcallback関数を登録できるようにする。

VA LINUX SYSTEMS JAPAN

SCSIレイヤ

全面書き直しを検討中

- 現状3階層のSCSIレイヤを２階層構造にし、状態を単純化する。現ミッドレイヤ機能はライブラリ化
  - IDE-SCSIを捨てることも可能
- Write barrier機能の実装。ジャーナリングFS, DB用途に有効。
  - エラー、再実行、キャンセル時の扱いが困難
- SCSIデバイス名の命名手段のユーザへの開放
  - HotPlug式にユーザプロセスから検出させる？
- マルチパスドライバとの連携

VA LINUX SYSTEMS JAPAN

Read-Copy-Update高速排他

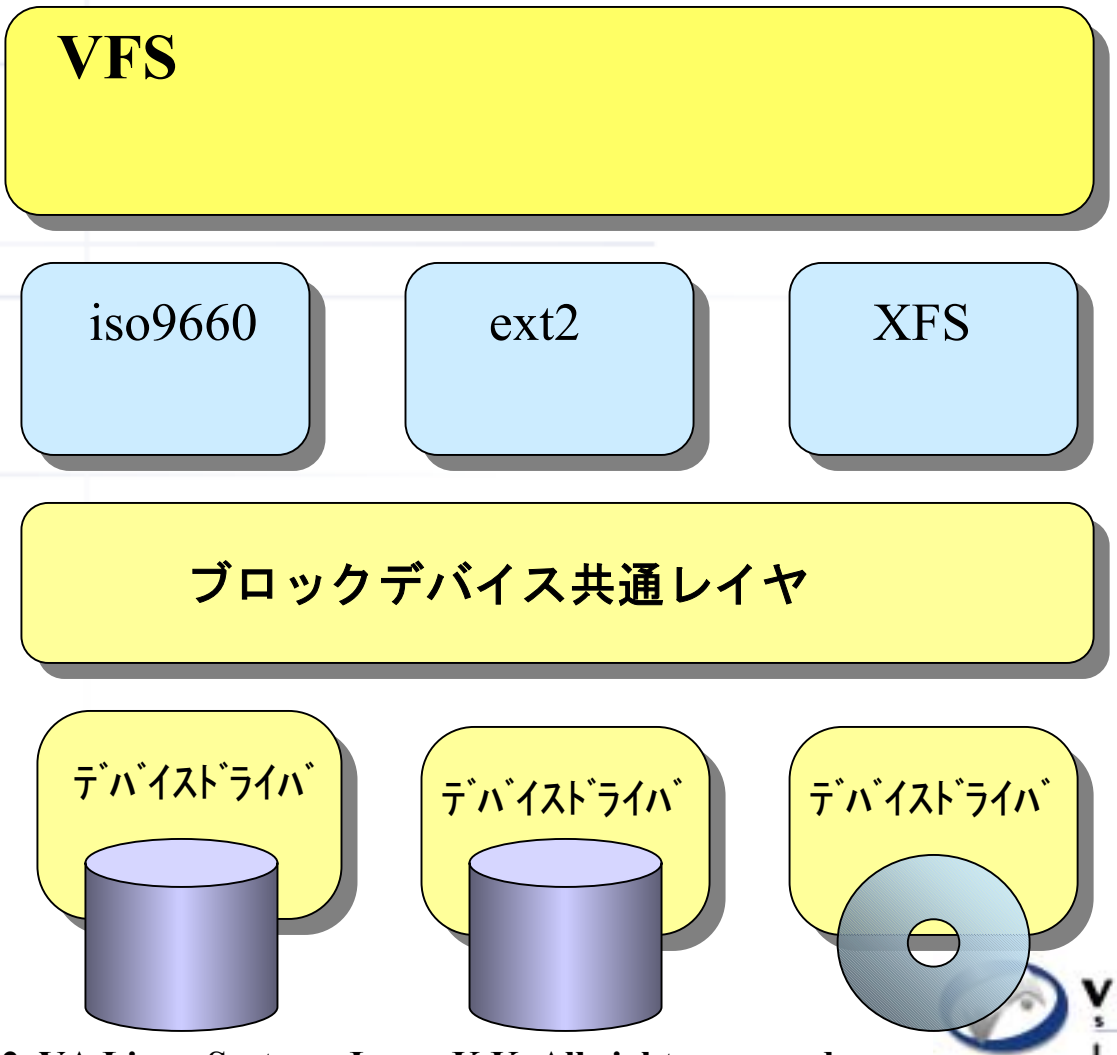
スピンロックに代わる新しい排他方式

- 単純なread/writeロックでは、CPU数が増えてくるとキャッシュライン共有により性能劣化
- Read側はロックせず、write側の処理を遅延させることにより、キャッシュライン競合問題を回避
- 問題点は状態が非常に複雑になること
- Dentryの排他処理への適用で大きな効果

VA LINUX SYSTEMS JAPAN

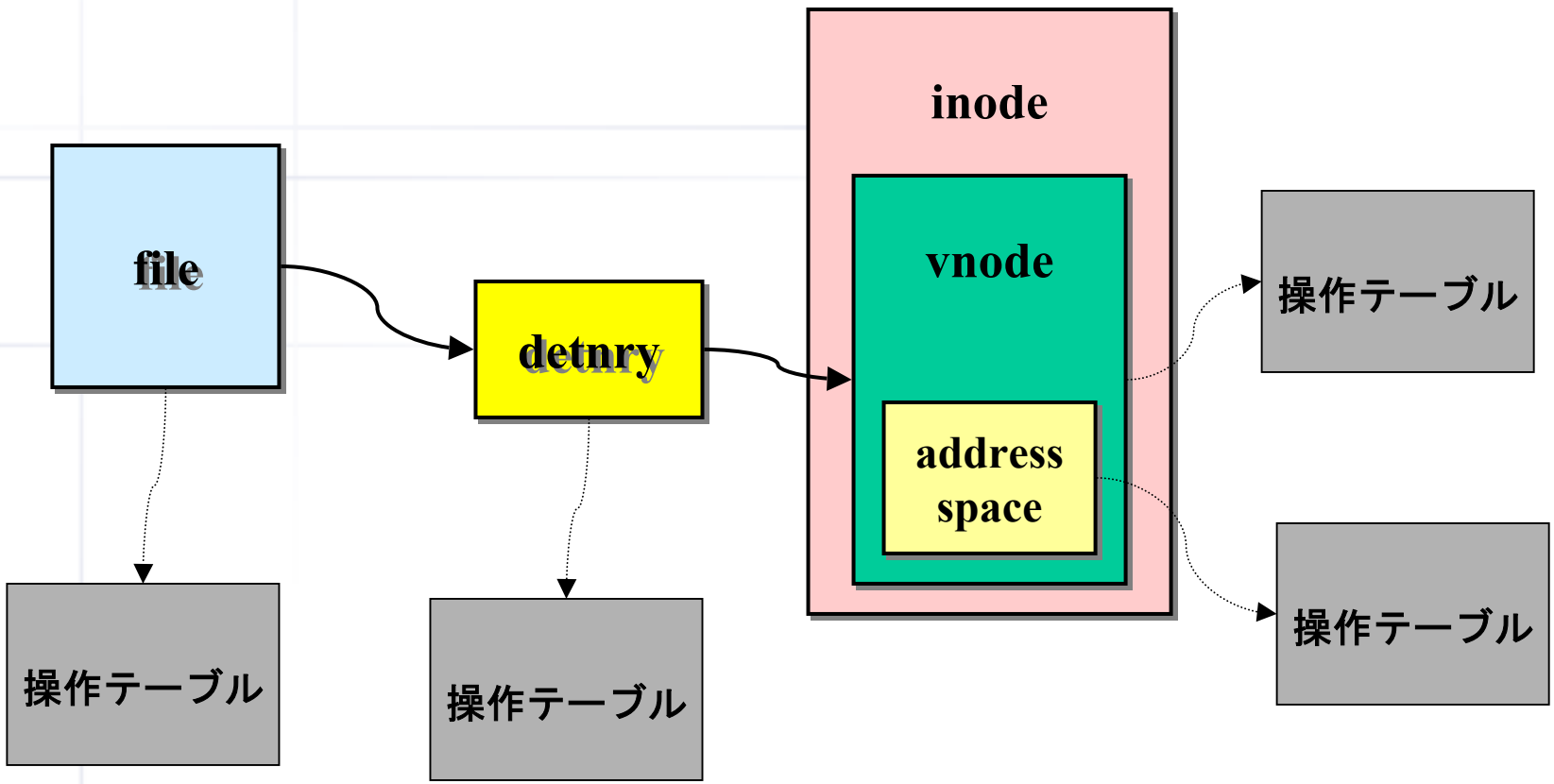
ファイルI/O最新技術

ＶＦＳ基本構造は同じ



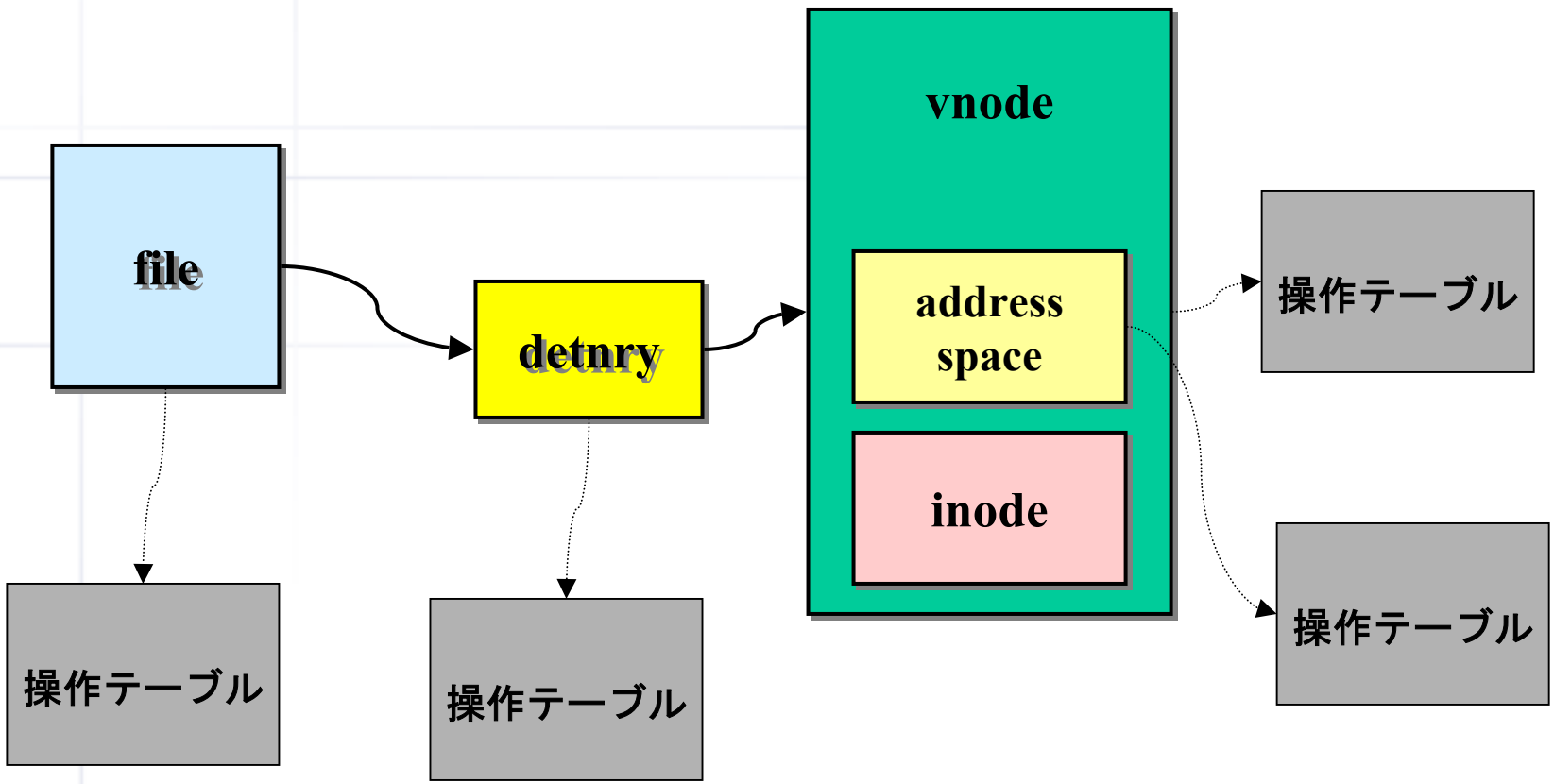
VA LINUX SYSTEMS JAPAN

主要なデータ構造(V2.5)



VA LINUX SYSTEMS JAPAN

主要なデータ構造(V2.4)



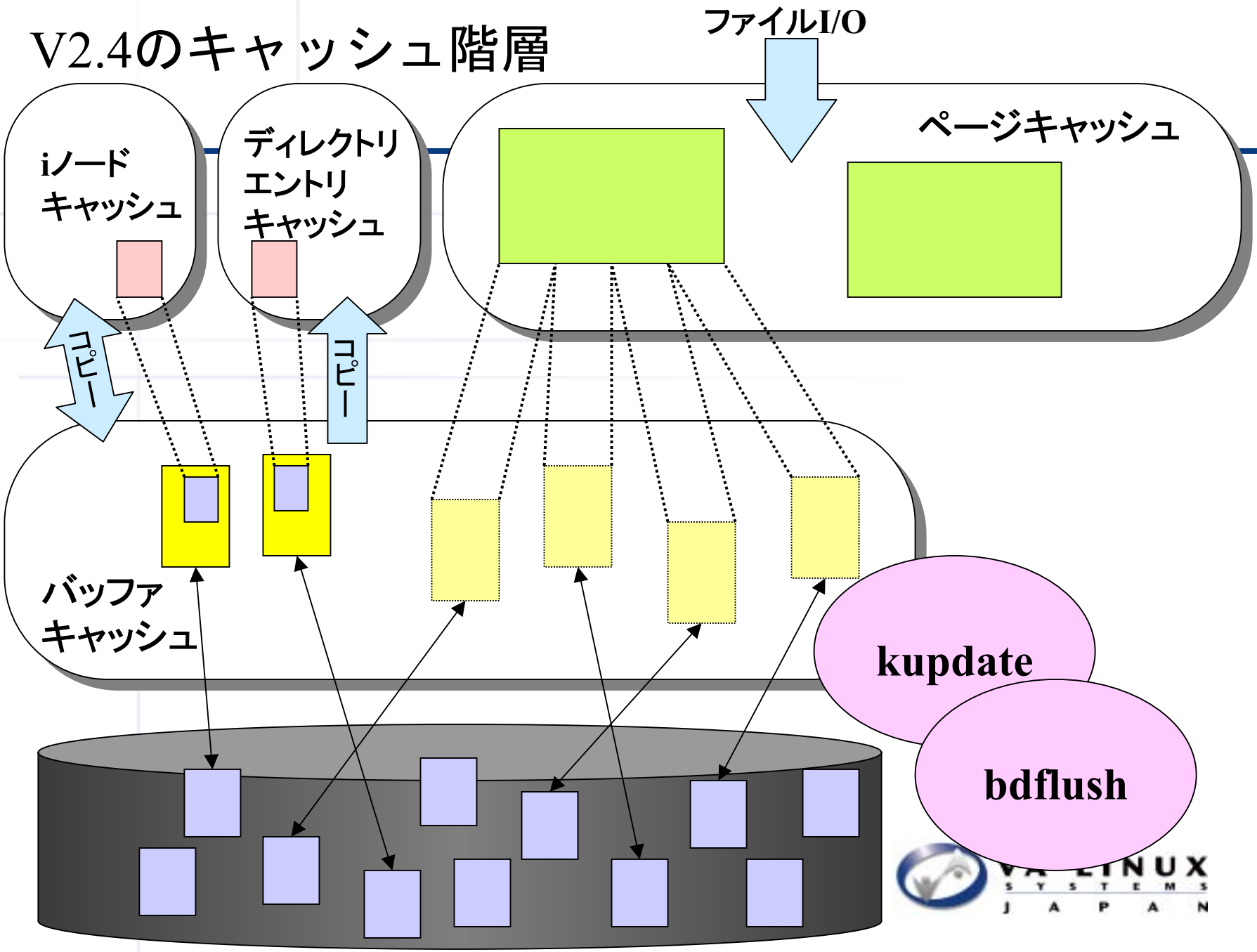
VA LINUX SYSTEMS JAPAN

ページI/O

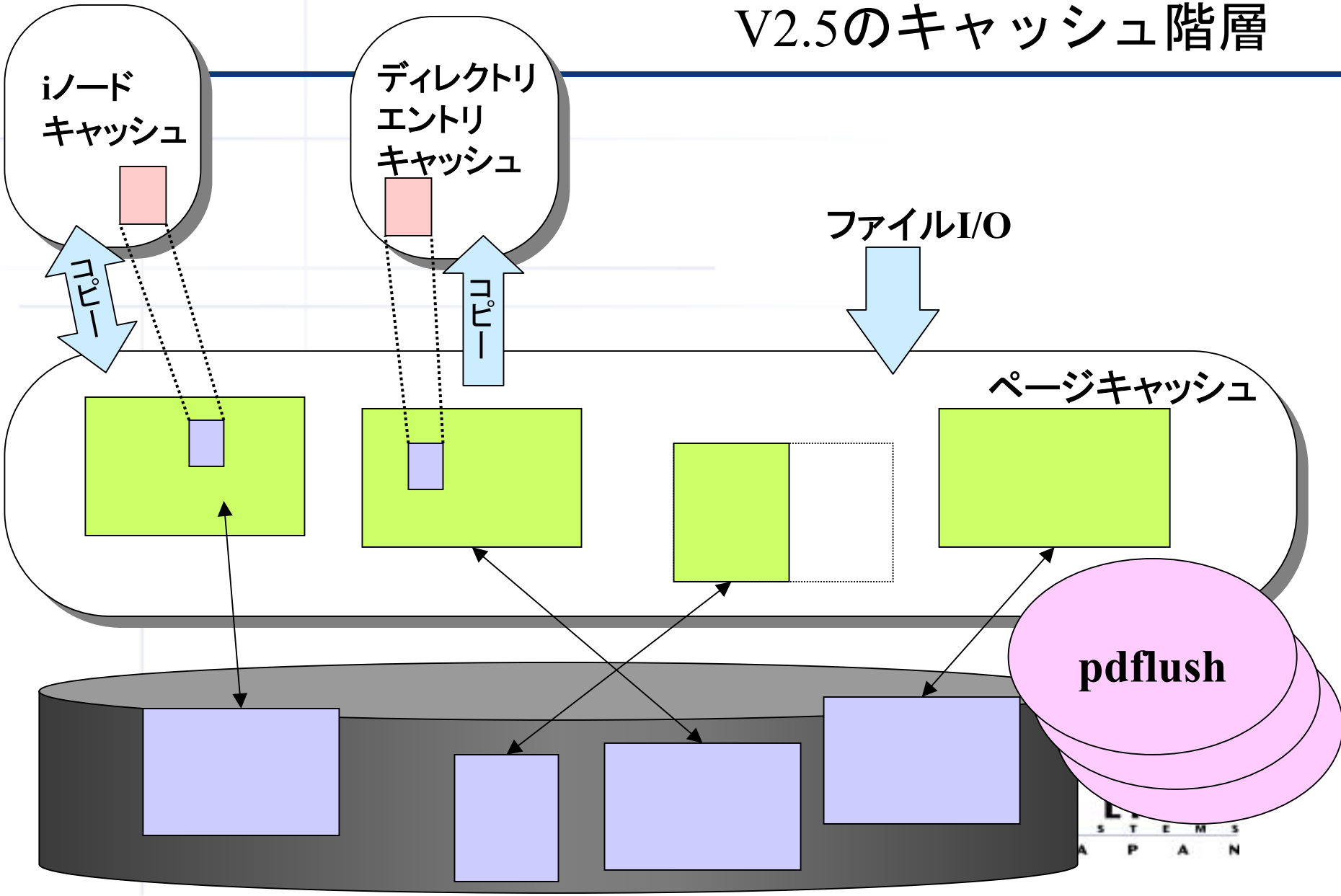
- バッファを中心とした仕組みから、ページを中心とした仕組みへの完全移行
- クラスタ化
  - ブロック的に連続する領域へのI/Oを一つのI/O要求として生成。ブロックが連続しやすいようにVFS側でも先行read・遅延write処理を再実装
- pdflushデーモン
  - 従来のkupdate, bdflushに代わるデーモン
- ブロック遅延確保機能の提供
  - ファイル拡張時ではなく、実際にディスクへ書き込む時点でデータブロックを割り当てる。I/O対象ブロックを連続させるための施策

VA LINUX SYSTEMS JAPAN

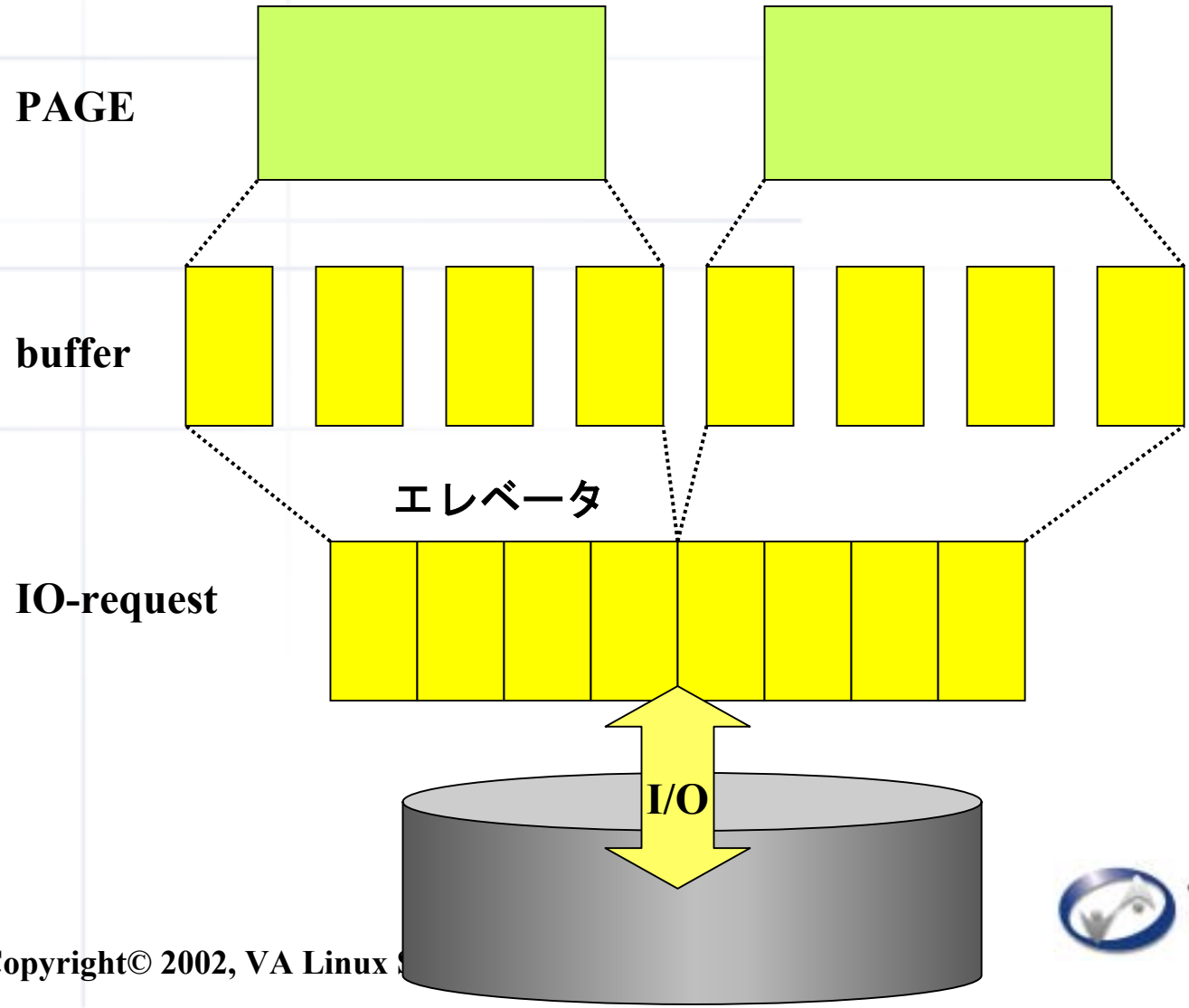
V2.4のキャッシュ階層



V2.5のキャッシュ階層

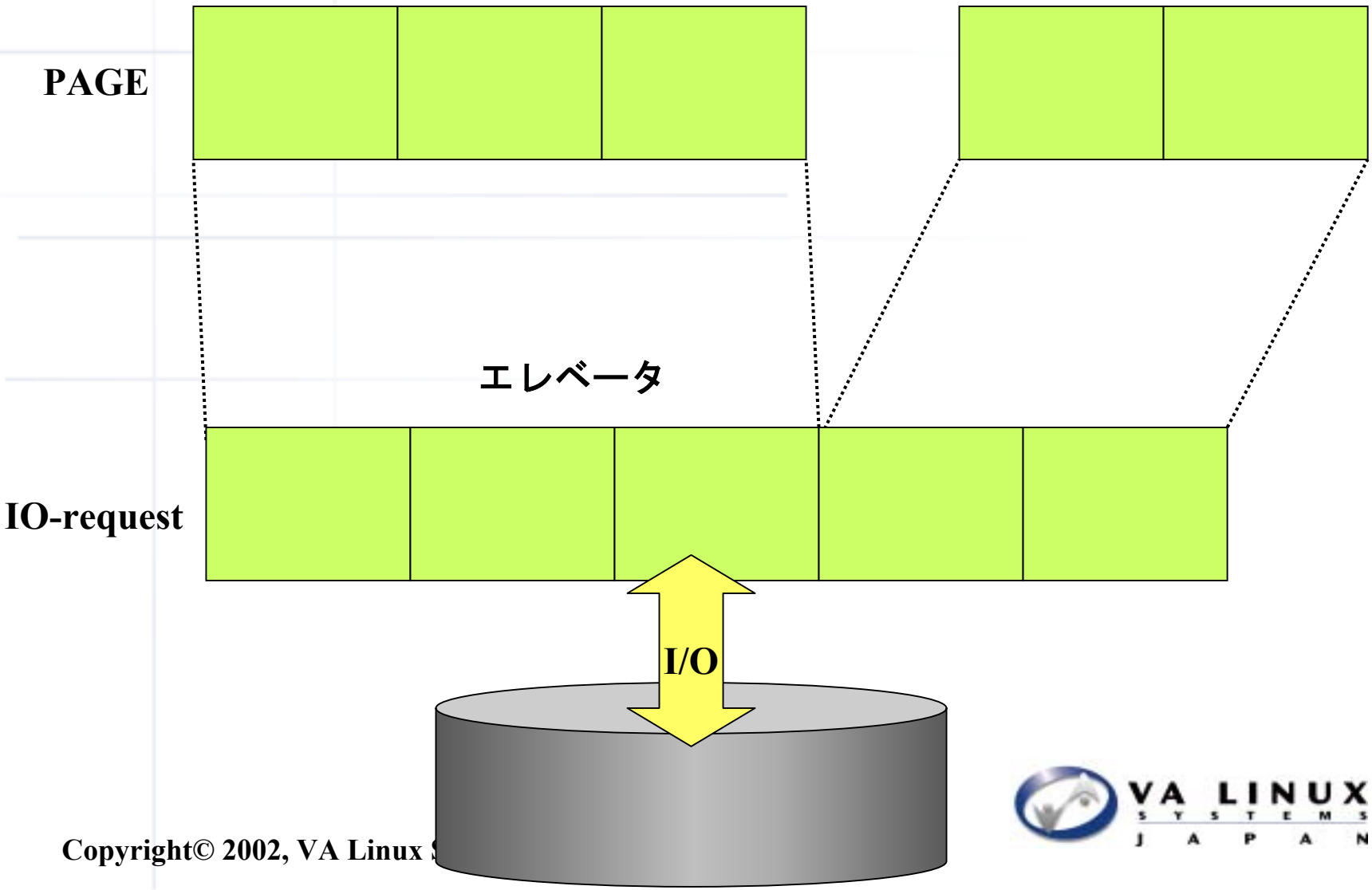


V2.4　キャッシュに対するI/O



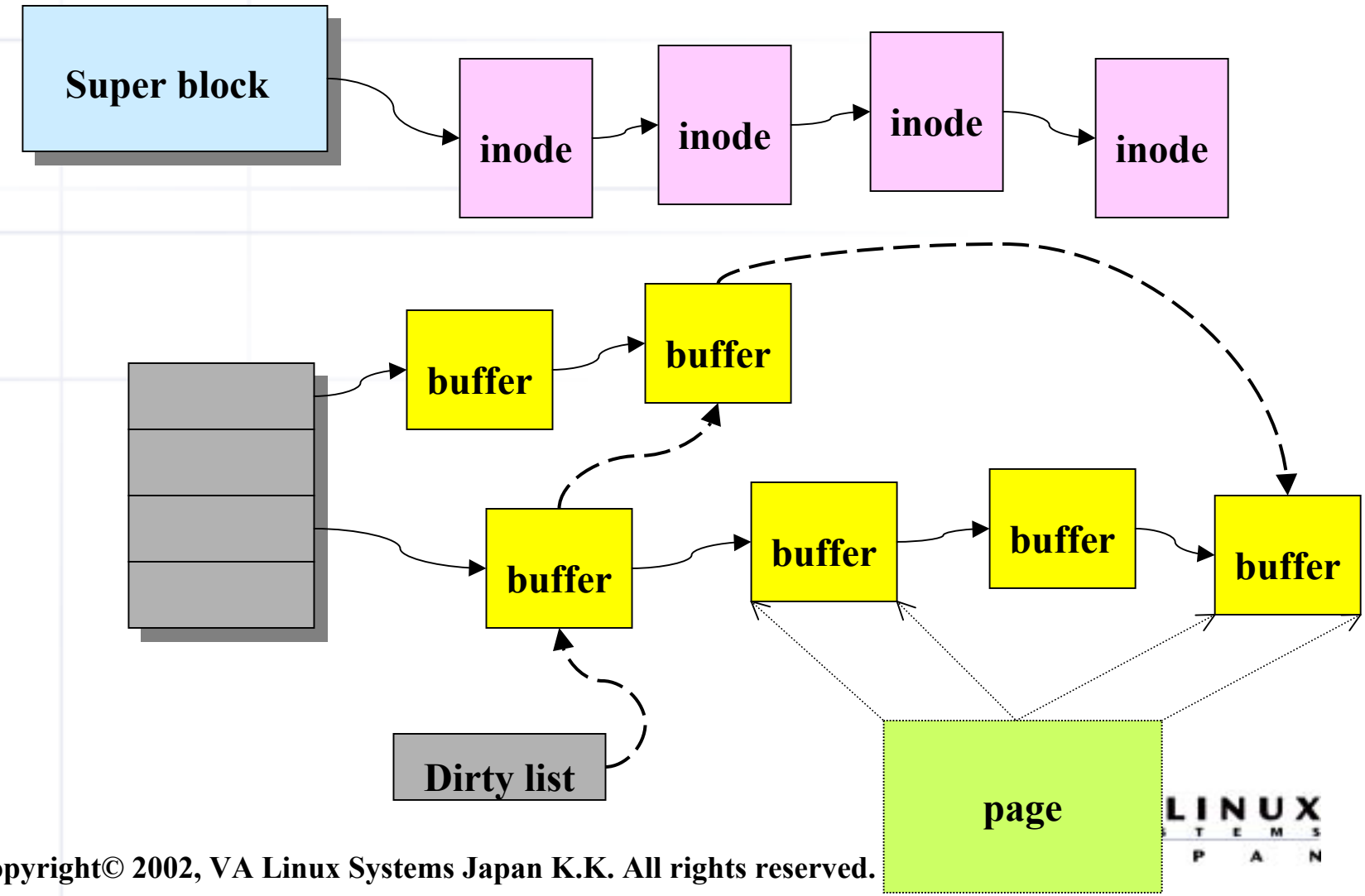
VA LINUX SYSTEMS JAPAN

V2.5　キャッシュに対するI/O

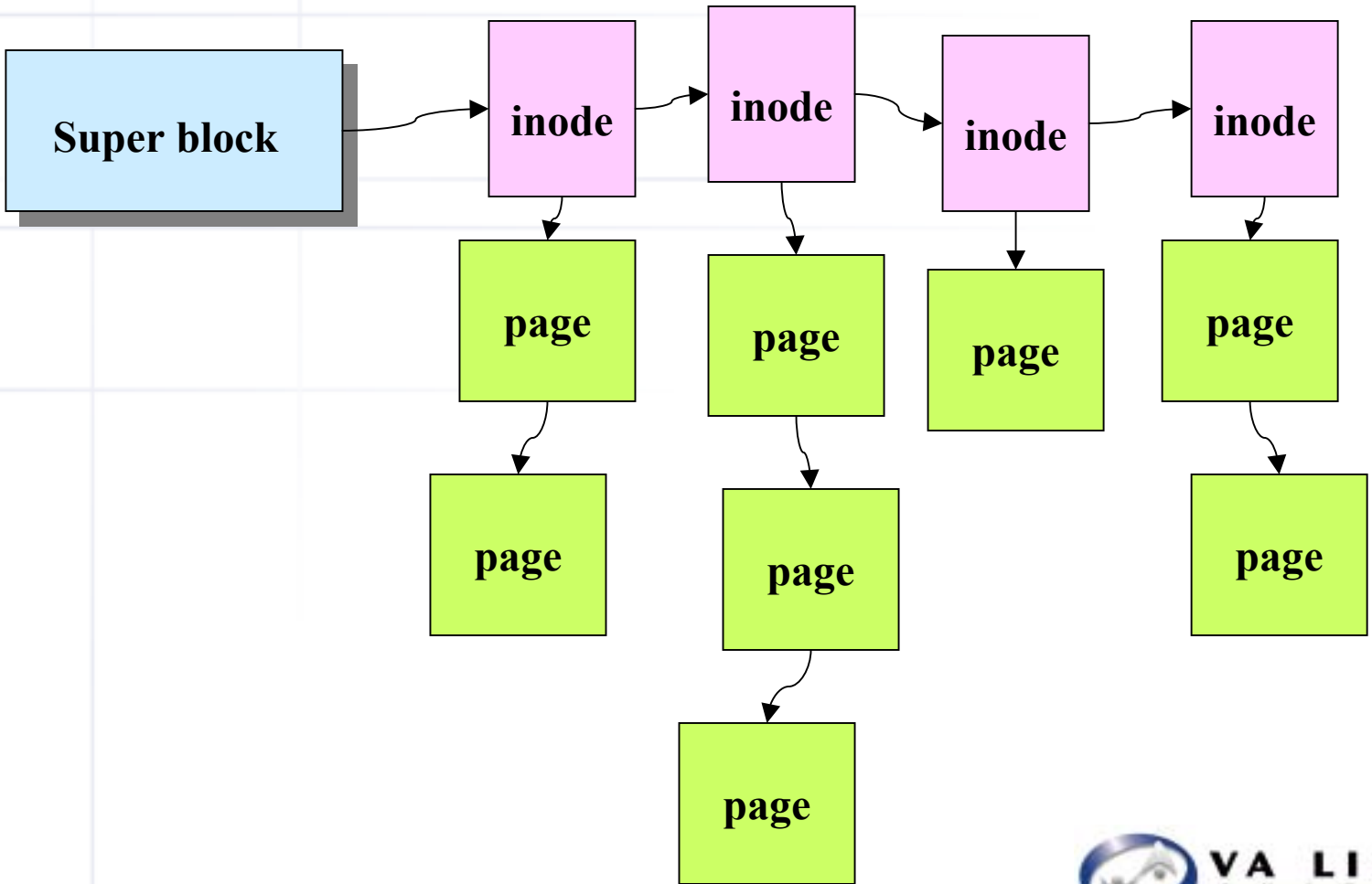


VA LINUX SYSTEMS JAPAN

V2.4　汚れたキャッシュの管理



V2.5　汚れたキャッシュの管理

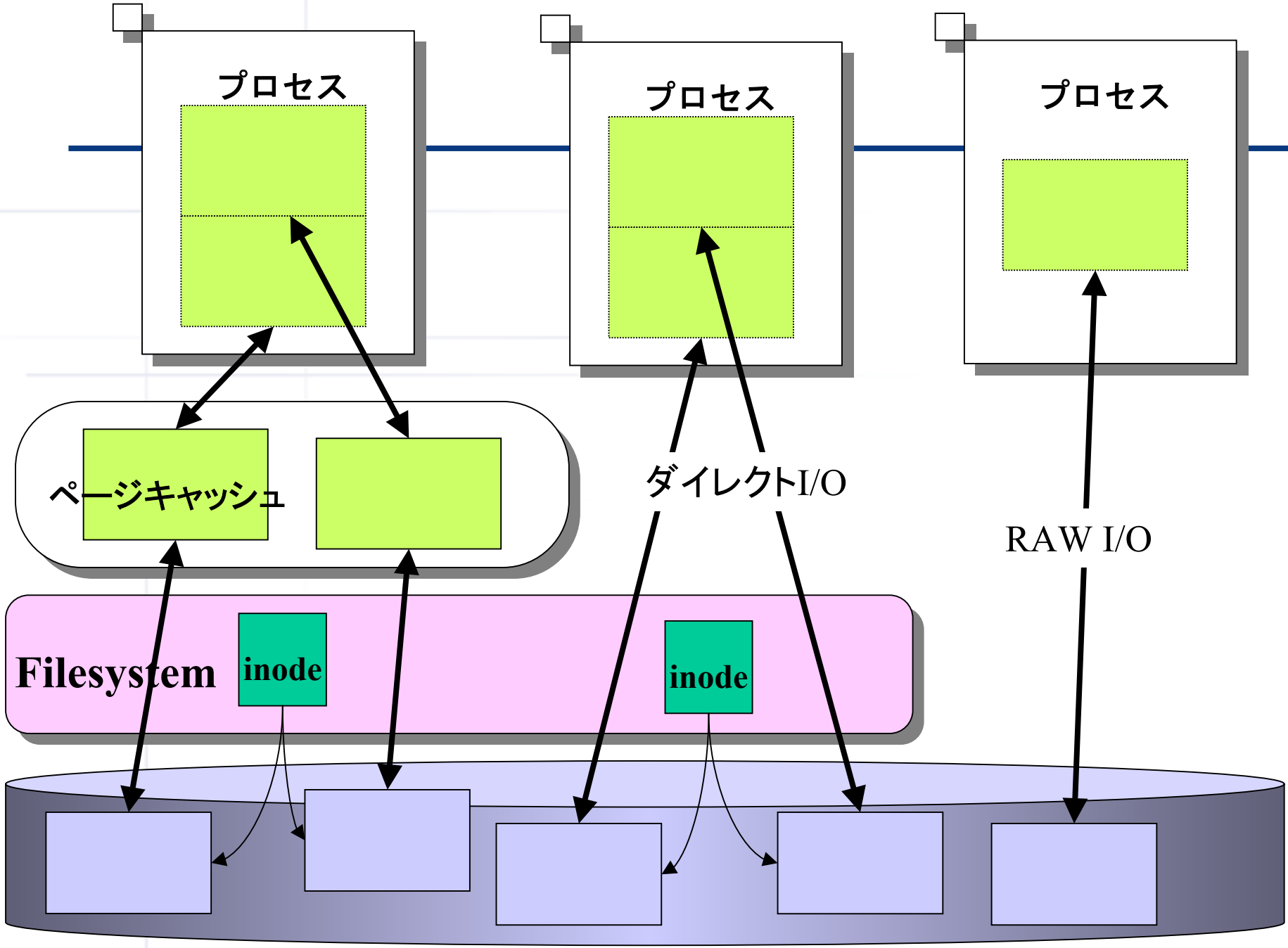


VA LINUX SYSTEMS JAPAN

ダイレクトI/O

- 従来のファイルI/OとRaw I/Oを組み合わせたようなI/Oが可能
  - ページキャッシュを通さずに、ユーザ空間とファイルの間でデータ転送
  - ユーザ空間のページロックと、DMAによるそのページとデバイスの間でのデータ転送
  - readv/writevの効率的な実装

VA LINUX SYSTEMS JAPAN

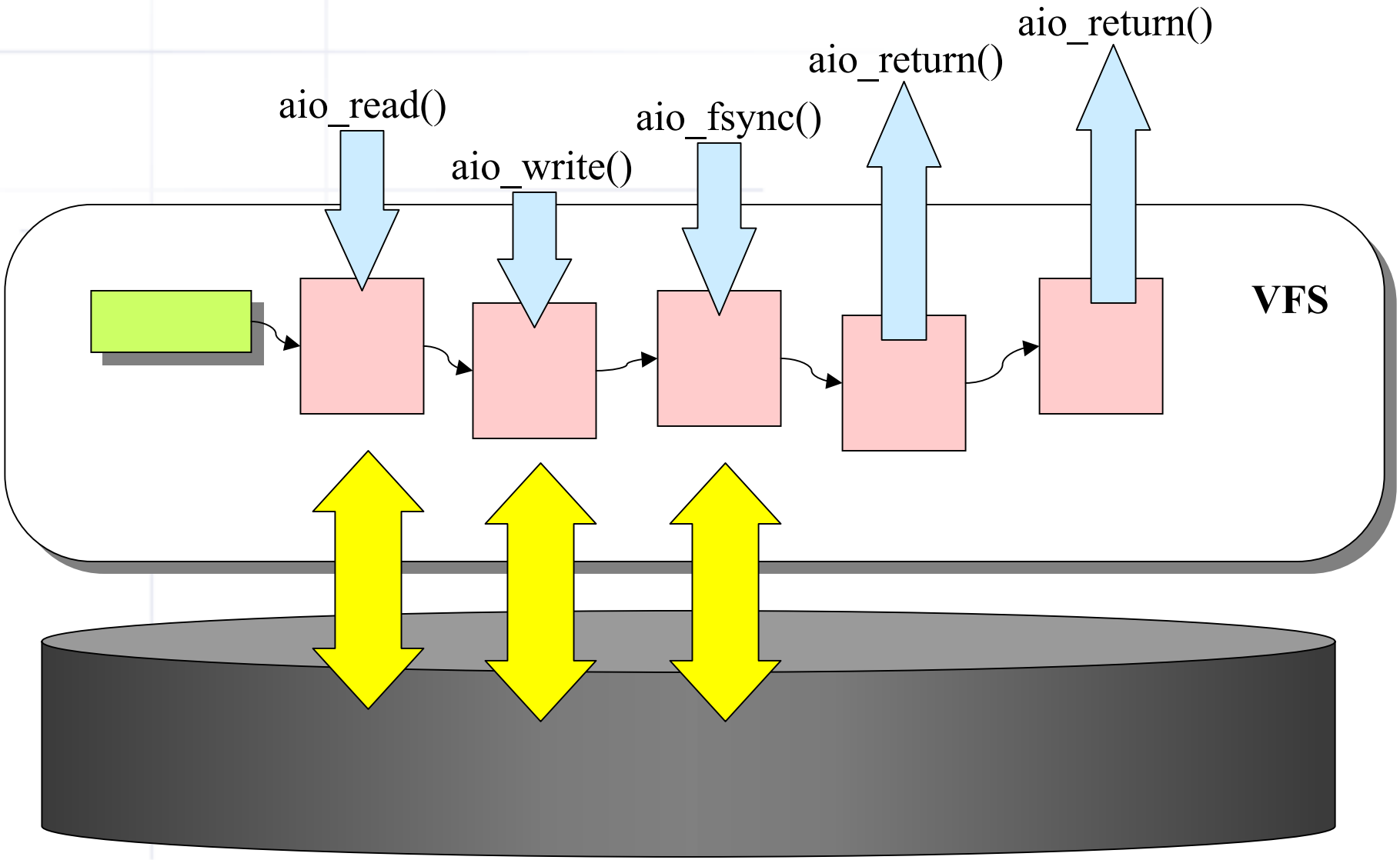

プロセス
プロセス
プロセス
ページキャッシュ
ダイレクトI/O
RAW I/O
Filesystem
inode
inode

非同期I/O（AIO） –実装中-

- I/O要求の発行と、I/O完了の待ち合わせを別に行うための仕組み (現在、実装中)。スレッドによるI/O並列化とは別の手法
  - io_submit、io_cancel、io_geteventsシステムコール（glibc的には、aio_read(), aio_write(), aio_listio(), aio_cansel(), aio_return()となると思われる）
  - 各ファイルシステムは、aio_read, aio_write, aio_fsyncメソッドを追加
- ＤＢ操作などにメリット

VA LINUX SYSTEMS JAPAN

非同期I/O（AIO）

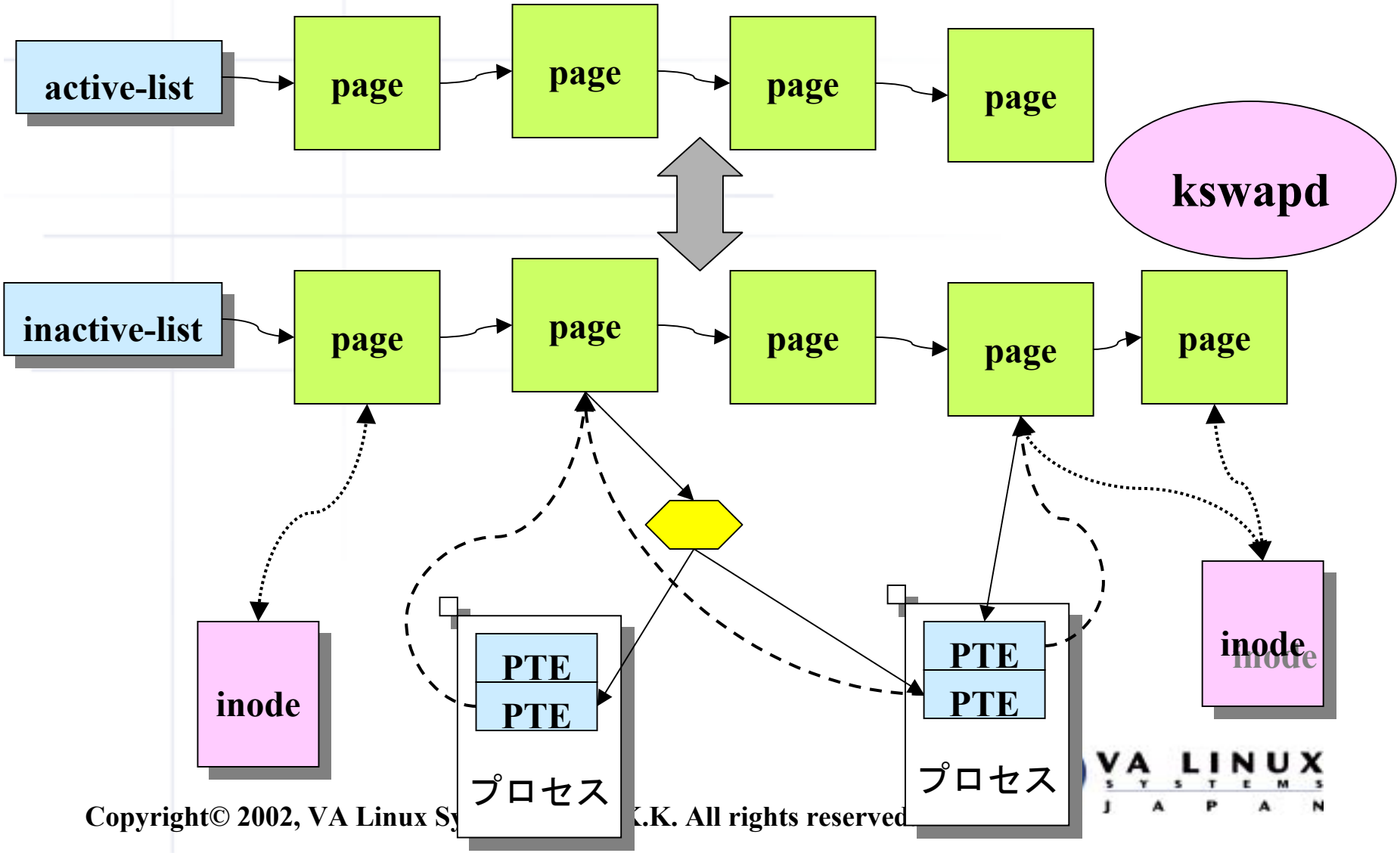


VMとの統合

- バランスの良いページ解放アルゴリズム
  - ページキャッシュとプロセス空間用ページの管理の一元化による、解放対象ページ選択の精度向上
  - 逆マッピングによる正確なアクセス頻度計算
- mmapされたファイルの更新と、writeシステムコールによるファイルの更新

VA LINUX SYSTEMS JAPAN

ページのAGING



VA LINUX SYSTEMS JAPAN

ブロックI/Oの改善

- HighMemの扱い
- Dead line I/Oスケジューリング

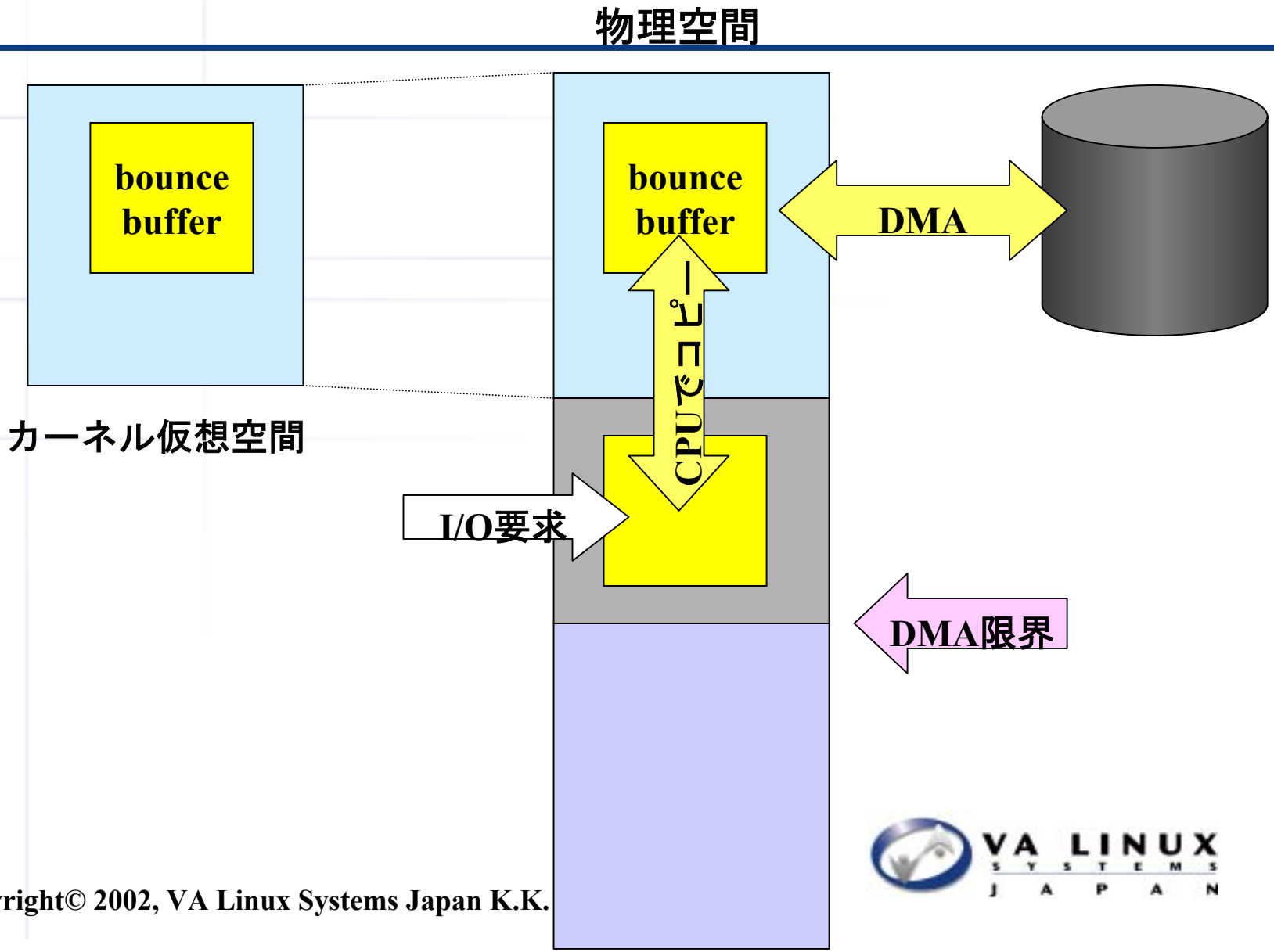
VA LINUX SYSTEMS JAPAN

HighMemの扱い

- HighMemであっても、ＤＭＡ可能な物理空間にはbounceバッファを経由せずにI/Oを行なうことが可能
  - カーネル空間内アドレスと物理アドレス(PCI空間内アドレス)の一対一対応の仮定の排除

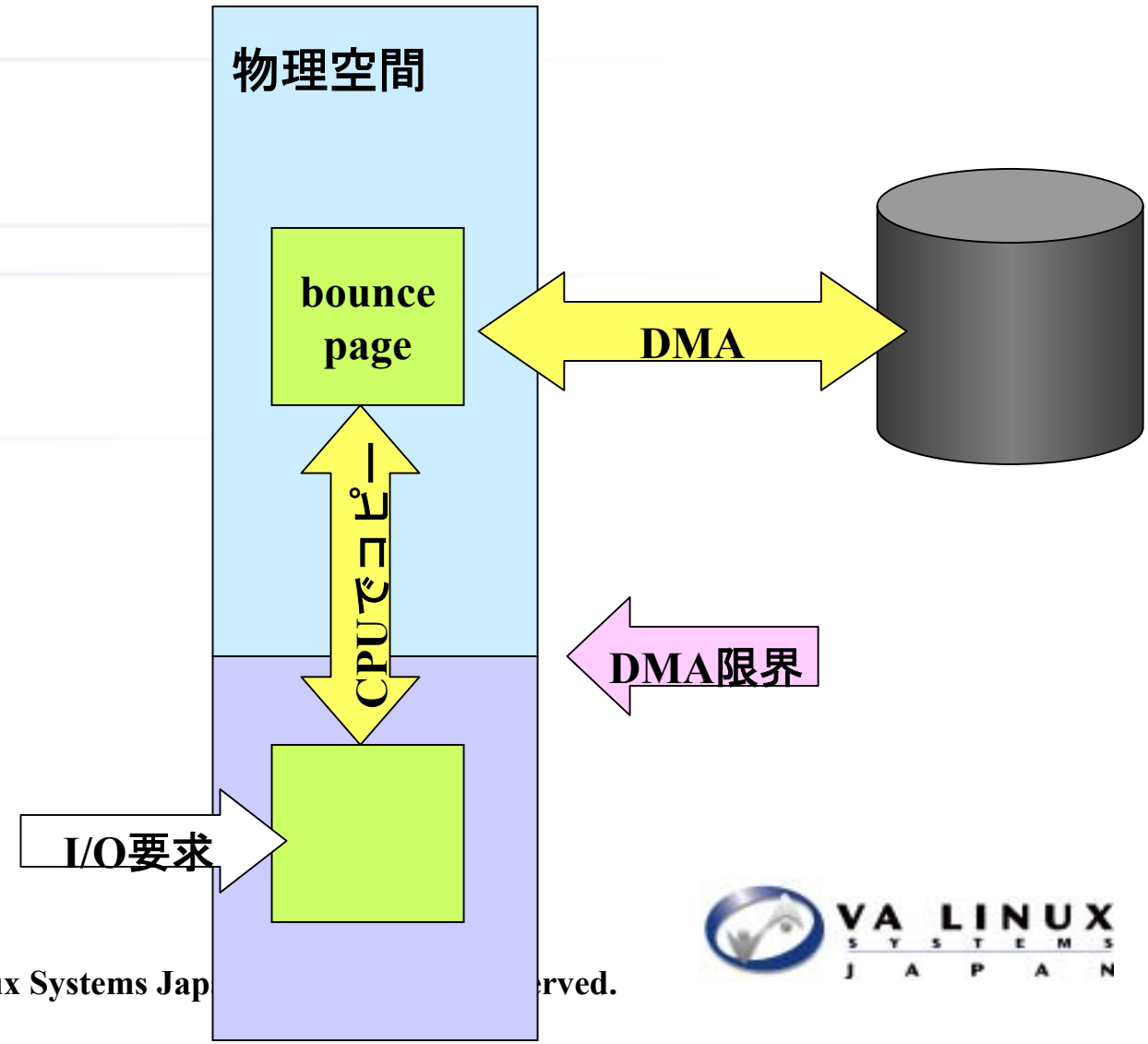
VA LINUX SYSTEMS JAPAN

2.4 Bounce Buffer



VA LINUX SYSTEMS JAPAN

2.5 Bounce Page



VA LINUX SYSTEMS JAPAN

Dead Line I/Oスケジューリング

- I/O時間の保証
  - とりあえず、read:500msec、write:2sec保証
- 新型elevatorとして実装

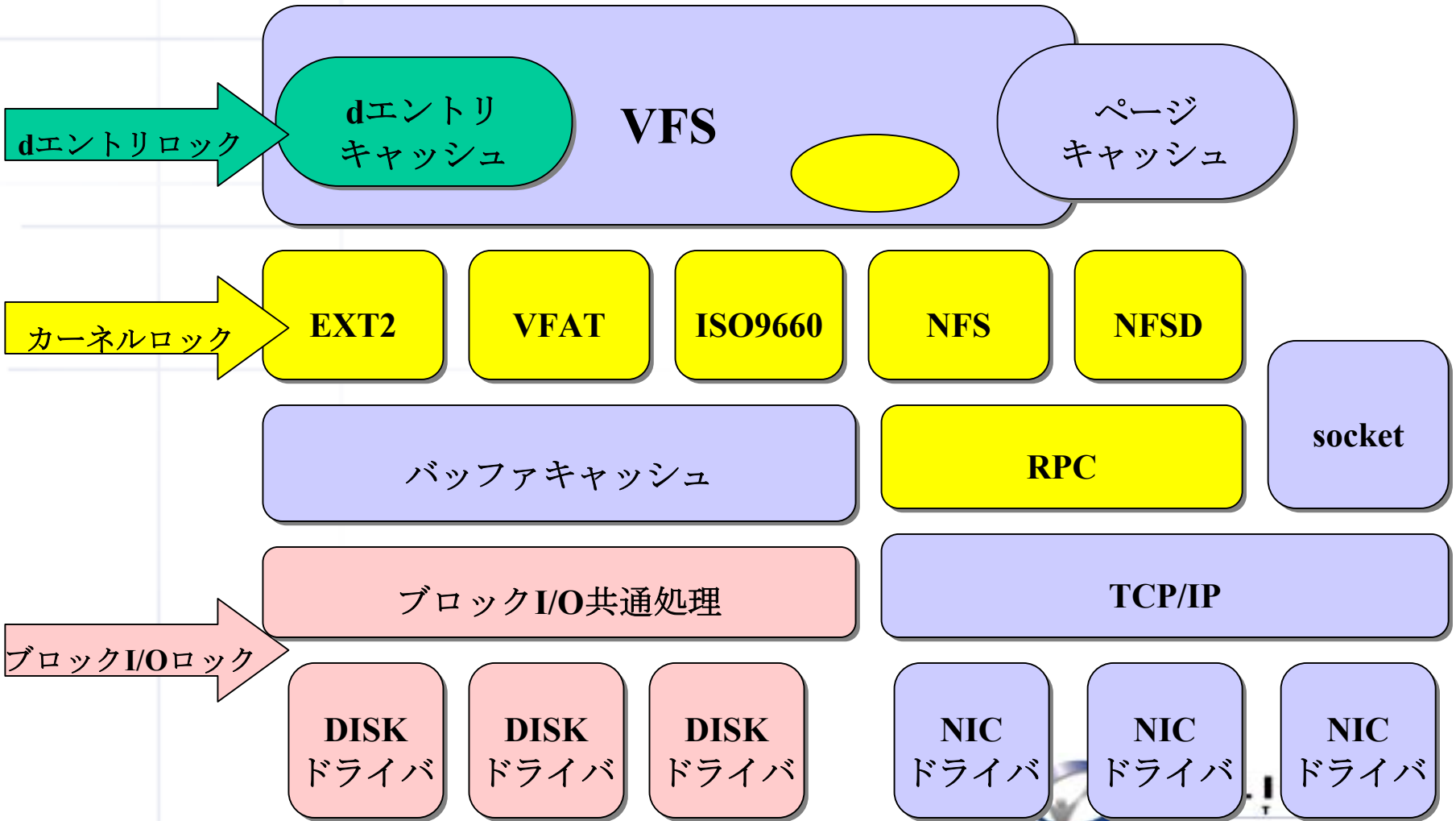
VA LINUX SYSTEMS JAPAN

マルチプロセッサ対応強化

- カーネルロック
  - ローカルファイルシステム呼び出し時のカーネルロックを解除。各ローカルファイルシステムの責任で排他
- ブロックデバイスのロック
  - ブロックデバイス全体のスピンロックから各デバイス毎のロックへ
  - SCSIではHBA毎のロック
- Dentryキャッシュのロック
  - RCU(Read-Copy-Update排他方式）が採用されるか？

VA LINUX SYSTEMS JAPAN

V2.4のスピンロック



V2.6(V3.0 ？) リリースをお楽しみに！