Linux NFSの高速化

ゼロコピーNFSの実装

VA Linux Business Forum

2003年6月20日

VA Linux Systems Japan

高橋　浩和

taka@valinux.co.jp

目次

- NFSとは、いかなるプロトコルか？
- Linux 2.4におけるNFSサーバの実装
- Linux 2.6におけるNFSサーバの実装
- 次の課題と目標

VA LINUX SYSTEMS JAPAN

NFSとはいかなるプロトコルか？

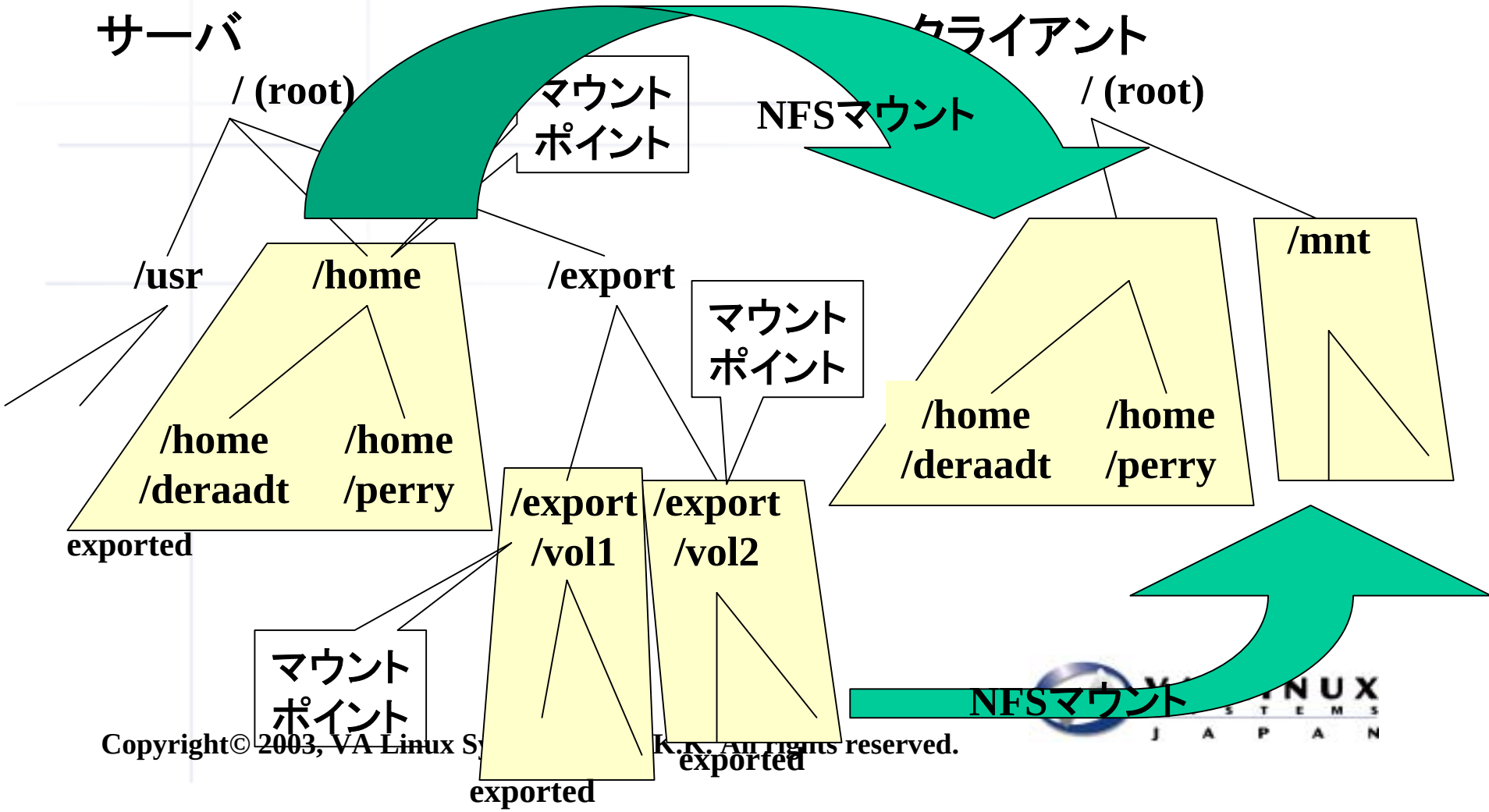
NFSの特徴と注意点

- 簡易にデータ共有を実現できる
- システムのボトルネックになりやすい
  - バランスの取れたハードウェア設計と、アプリケーション設計が必要
- UNIXのファイルシステム要件を完全には満たしていない
  - クライアントとサーバのキャッシュの一貫性問題、互いに状態反映を遅延
  - オープン中のファイルを削除可能

VA LINUX SYSTEMS JAPAN

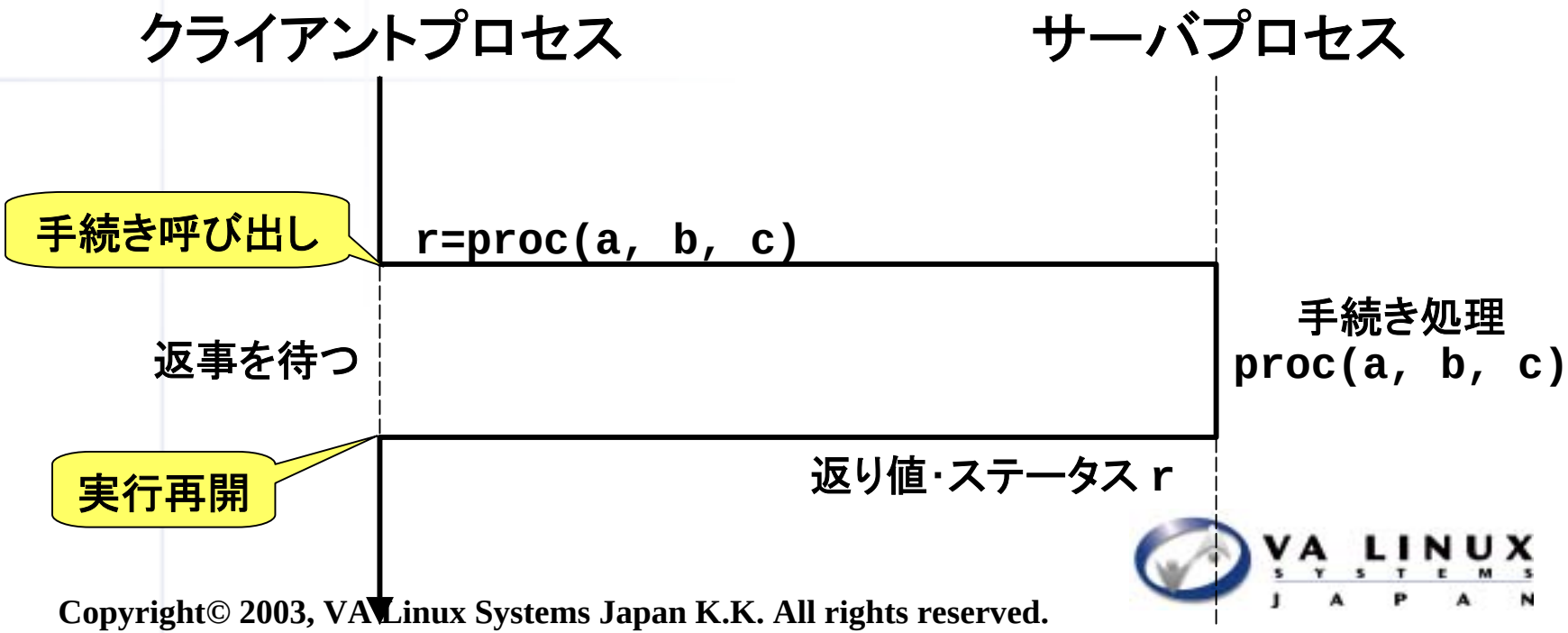
ファイルシステムモデル

Unixのファイルシステムモデル



ONC RPC (Sun RPC)

- Remote Procedure Call – リモートホストの手続きを呼び出す
- マーシャリング: XDR表現



ファイルハンドル

- NFSサーバ内のファイルを一意に特定する識別子
  - NFSのすべてのオペレーションの基本
  - サーバの再起動などでも不変
  - クライアントからは透過、サーバがすべて解釈

VA LINUX SYSTEMS JAPAN

リモートファイルへのアクセス

NFSクライアント上での下記コードの実行

```
fd = open(“home/esr/zen.txt”, O_RDONLY);
read(fd, buffer, size);
```


VA LINUX SYSTEMS JAPAN

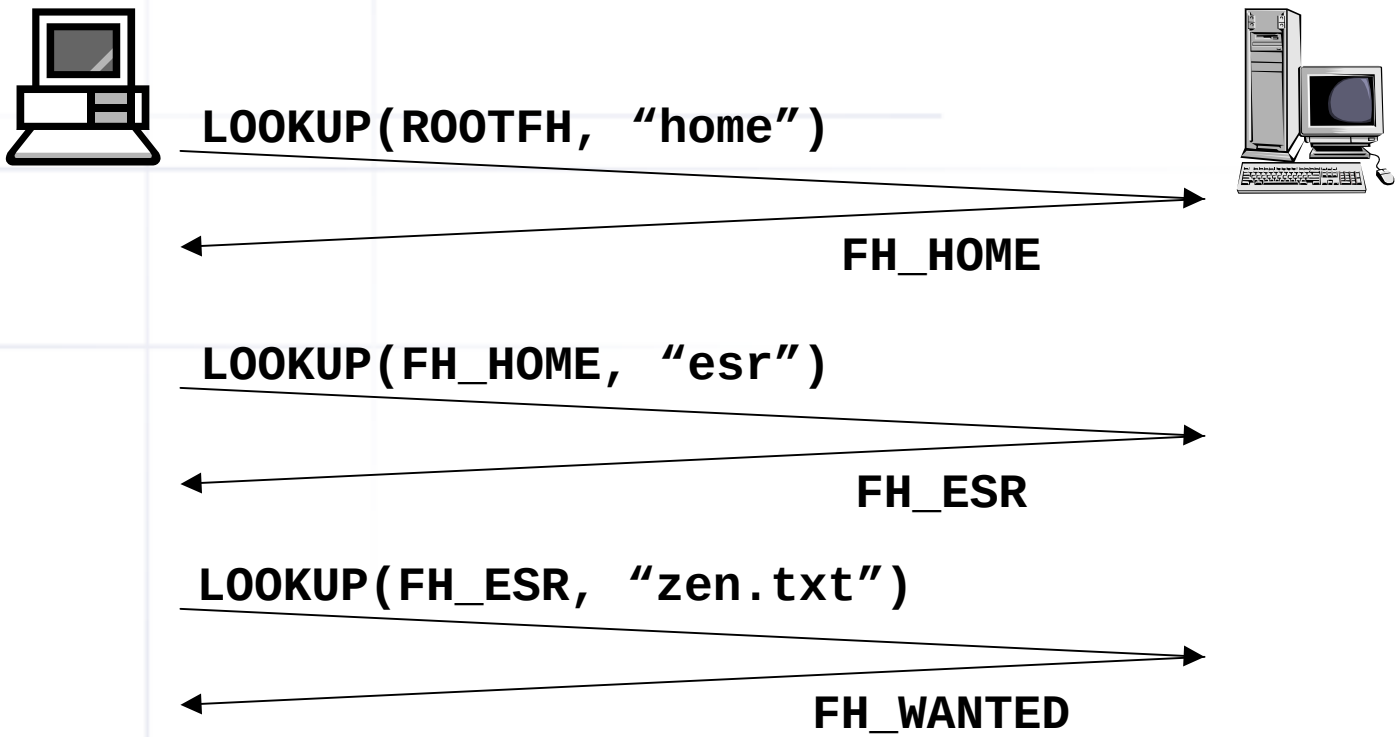
NFSのRPC手続き LOOKUP

- LOOKUP(FH, NAME) ⇒ RFH
- ファイルハンドルFHで示されるディレクトリにあるNAMEというファイルのファイルハンドルRFHを返す
- パス名のコンポーネントごとに呼び出し
  - シンボリックリンクの解釈はクライアントの仕事 ⇒ READLINK手続き

VA LINUX SYSTEMS JAPAN

LOOKUPの処理

例: home/esr/zen.txtのLOOKUP



ファイルハンドルのキャッシュが重要

NFSのRPC手続き　READ

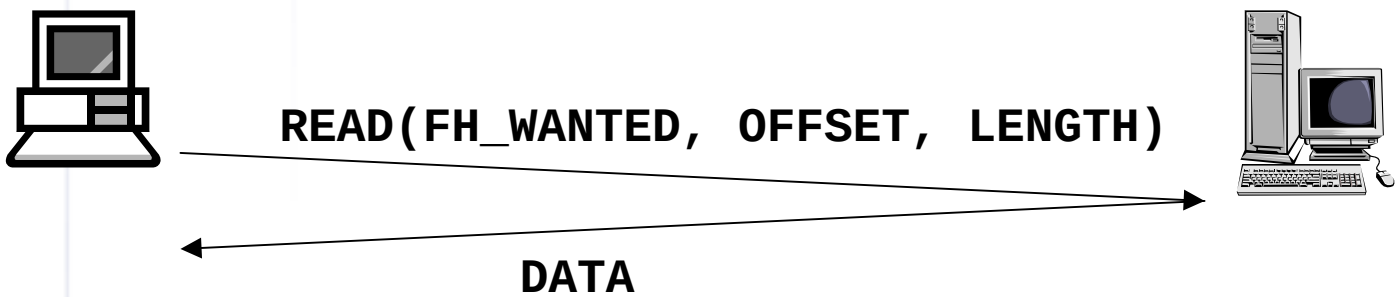
- READ(FH, OFFSET, LENGTH) ⇒ DATA

- FHで示されるファイルオブジェクトの、OFFSETバイト目からLENGTHバイトを読む
- ファイルポインタのような概念はない ⇒ クライアント側の実装で用意
- オープン状態を持たない。オープン操作は不要。（NFS v2, v3）

VA LINUX SYSTEMS JAPAN

READの処理

- 毎回、対象ファイル(ファイルハンドル)と、ファイルオフセットを指定してアクセス



VA LINUX SYSTEMS JAPAN

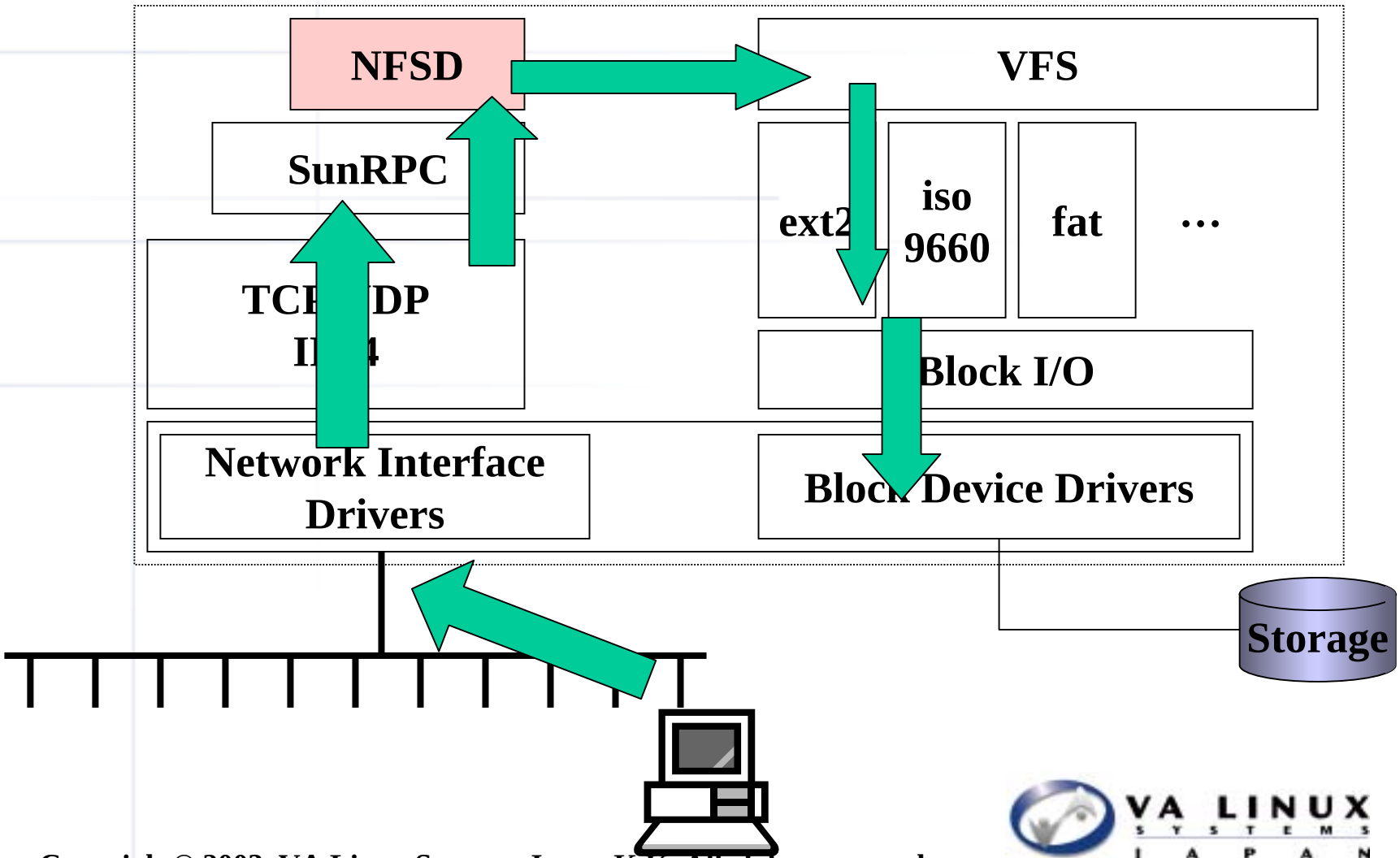
ステートレス性

- RPC要求と応答で全ての処理が完結 - 状態を保存する必要がない
- (特にクライアントの)実装は状態(ステート)を持つ
  - ファイルポインタ、モード、...
  - キャッシュ

VA LINUX SYSTEMS JAPAN

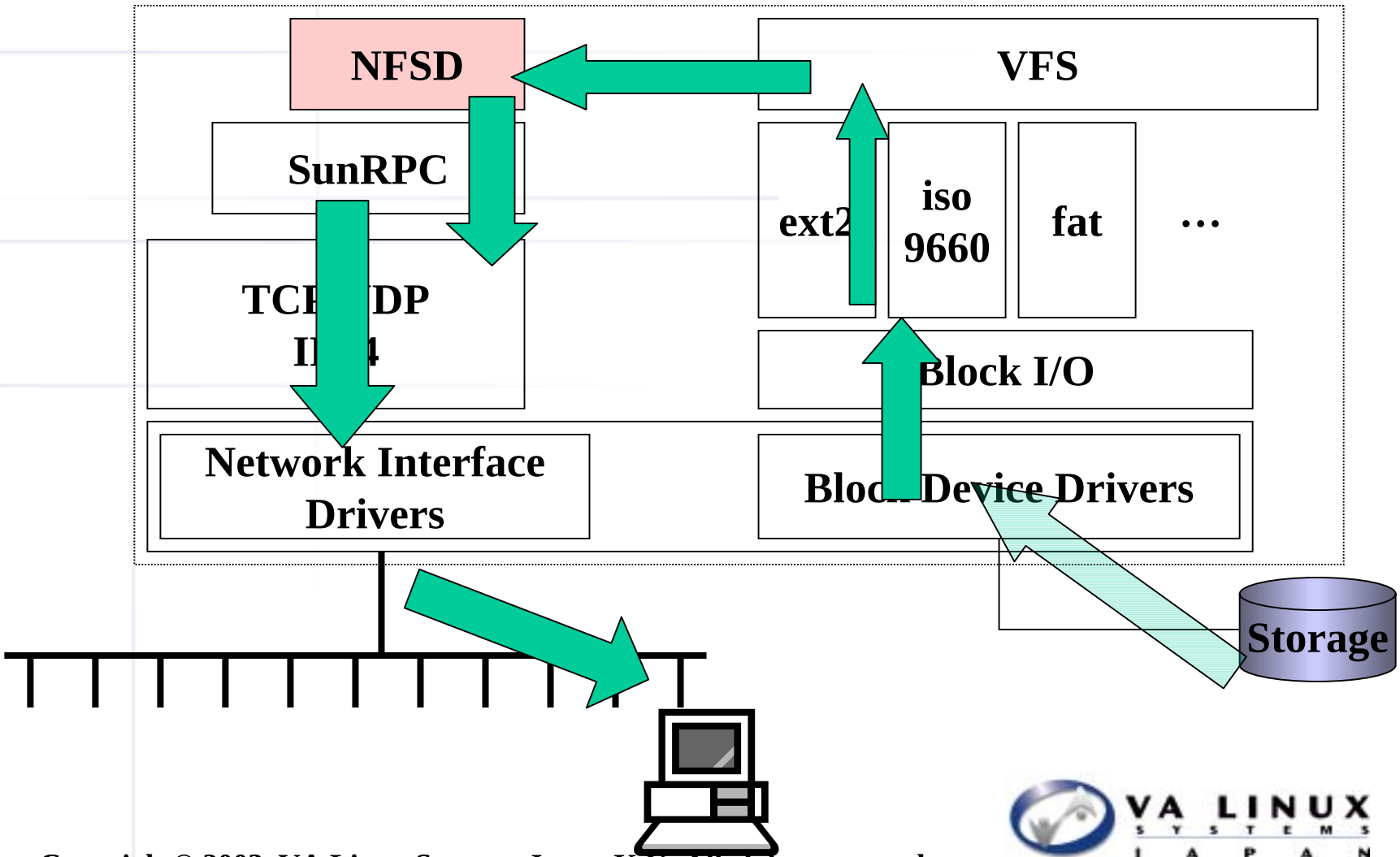
Linux 2.4における NFSサーバの実装

カーネルNFSD (1/2)



VA LINUX SYSTEMS JAPAN

カーネルNFSD (2/2)



VA LINUX SYSTEMS JAPAN

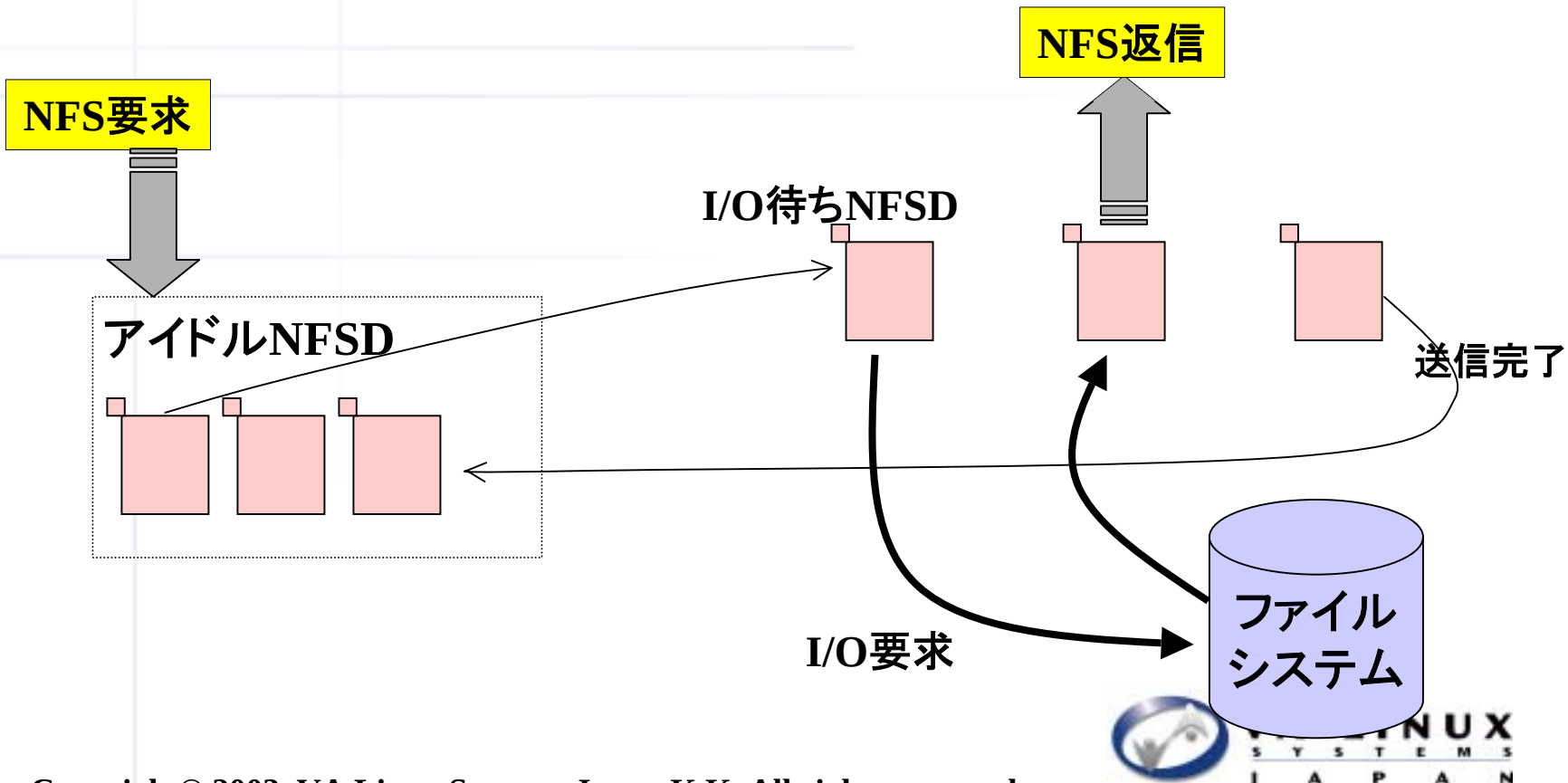
Linux 2.4のNFSサーバの動作

1. NFS要求の受け付けとカーネルスレッドの起動
2. NFS要求のデコード
3. NFS要求に応じた、VFSファイルオブジェクトのメソッドの呼び出し
   - read要求の場合は、readメソッドを呼び出し、ファイルオフセットに対応するデータを、NFSバッファに読み込む
4. NFS応答の生成とエンコード
   - NFSバッファ上にRPC/NFSヘッダを生成
5. NFS応答の送信
   - sock_sendmsgにより、NFSバッファ上のデータを送信

VA LINUX SYSTEMS JAPAN

NFSDカーネルスレッド

NFS要求の受信、ファイルI/O、結果の送信を並列実行

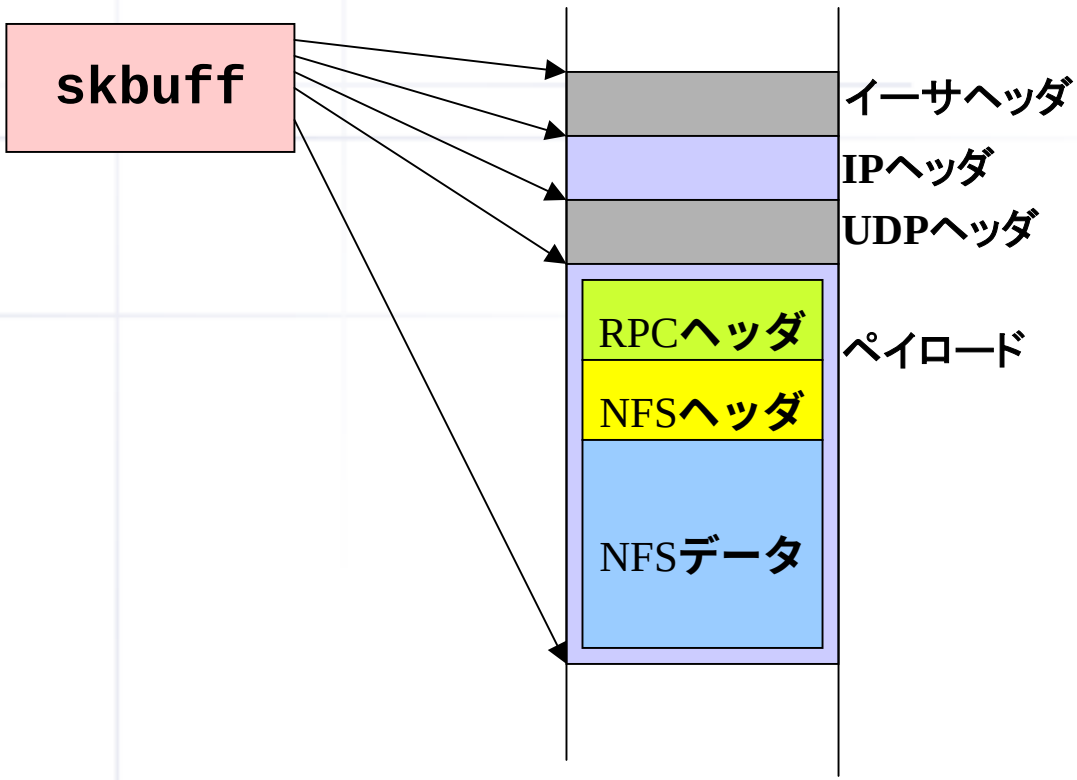


RPCパケットの受信

- ソケットに皮を被せた、RPC専用ソケット
- 事象発生時に呼び出される各種コールバック関数をRPC処理用のものに変更
- NFS over UDPの場合、一つのソケットを全てのスレッドで共有
- NFS over TCPの場合、各コネクションごとにソケットを用意

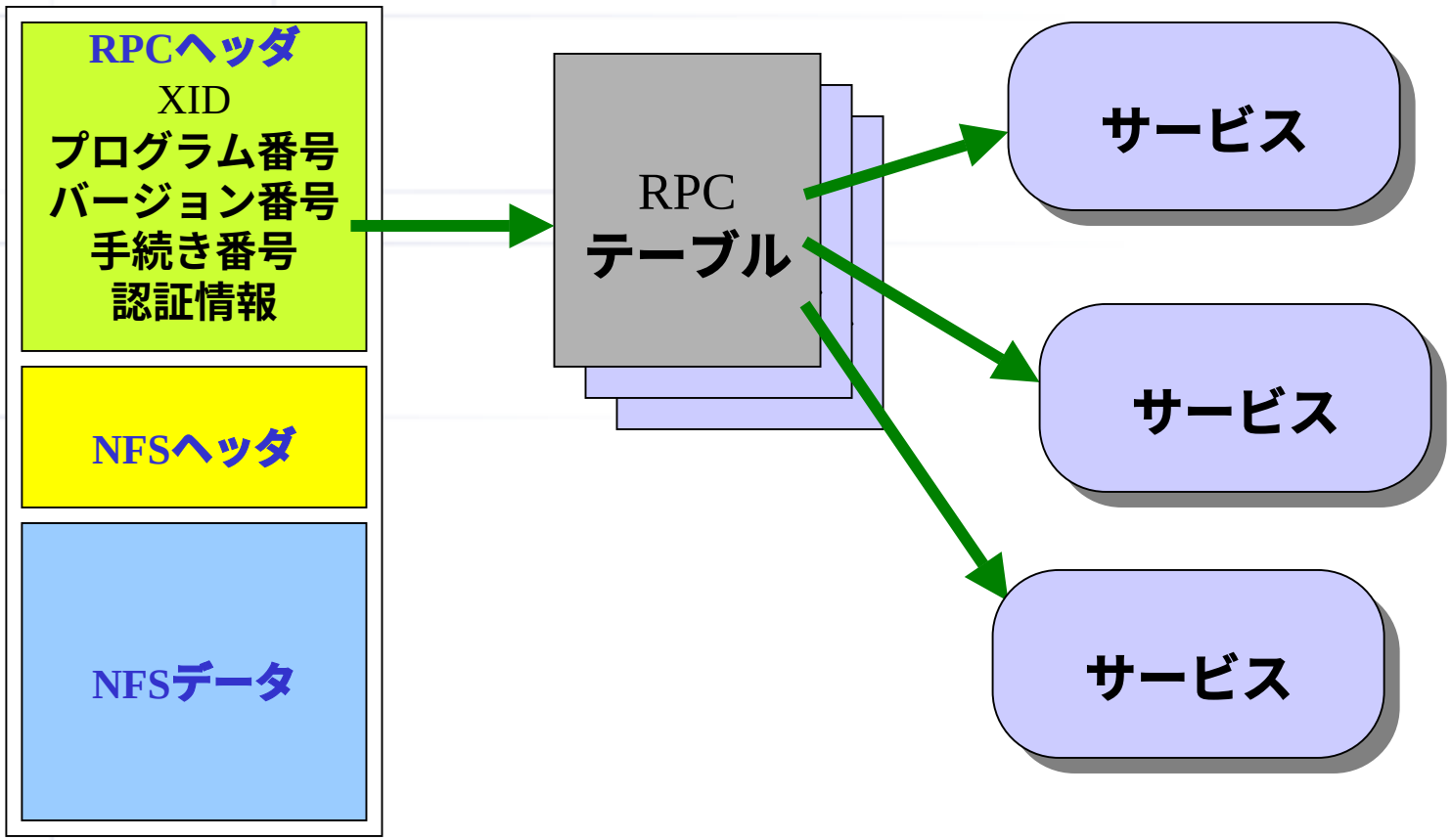
VA LINUX SYSTEMS JAPAN

UDPフレームとNFSバッファ



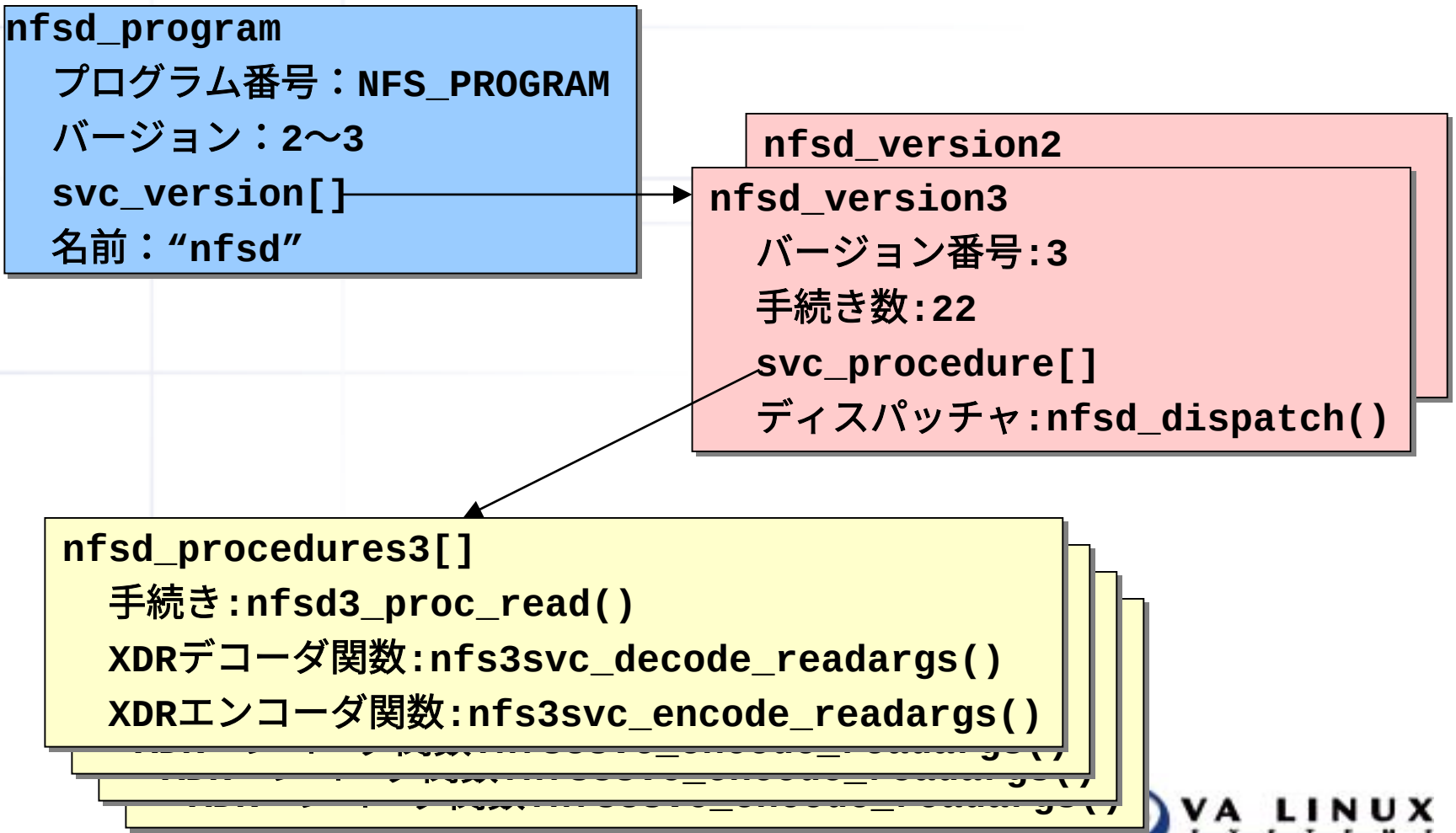
VA LINUX SYSTEMS JAPAN

RPCパケット



VA LINUX SYSTEMS JAPAN

RPC要求とNFS要求



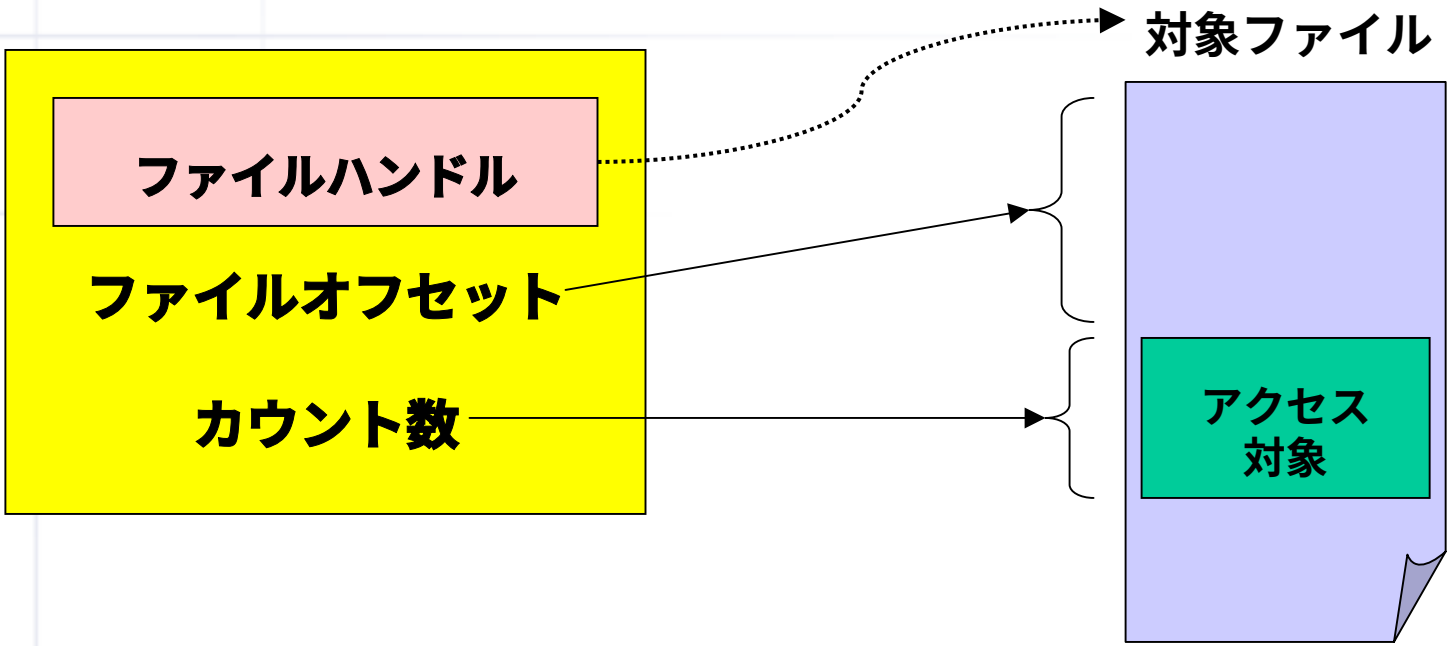
VA LINUX SYSTEMS JAPAN

XDR表現

- 相互にやりとりするデータの共通のエンコーディング方式
  - 各マシンでのエンディアン、型の大きさ、アライメントに依存しない
- カーネル内で利用する前に、ローカルマシンでのデータ形式に変換する（エンコード）

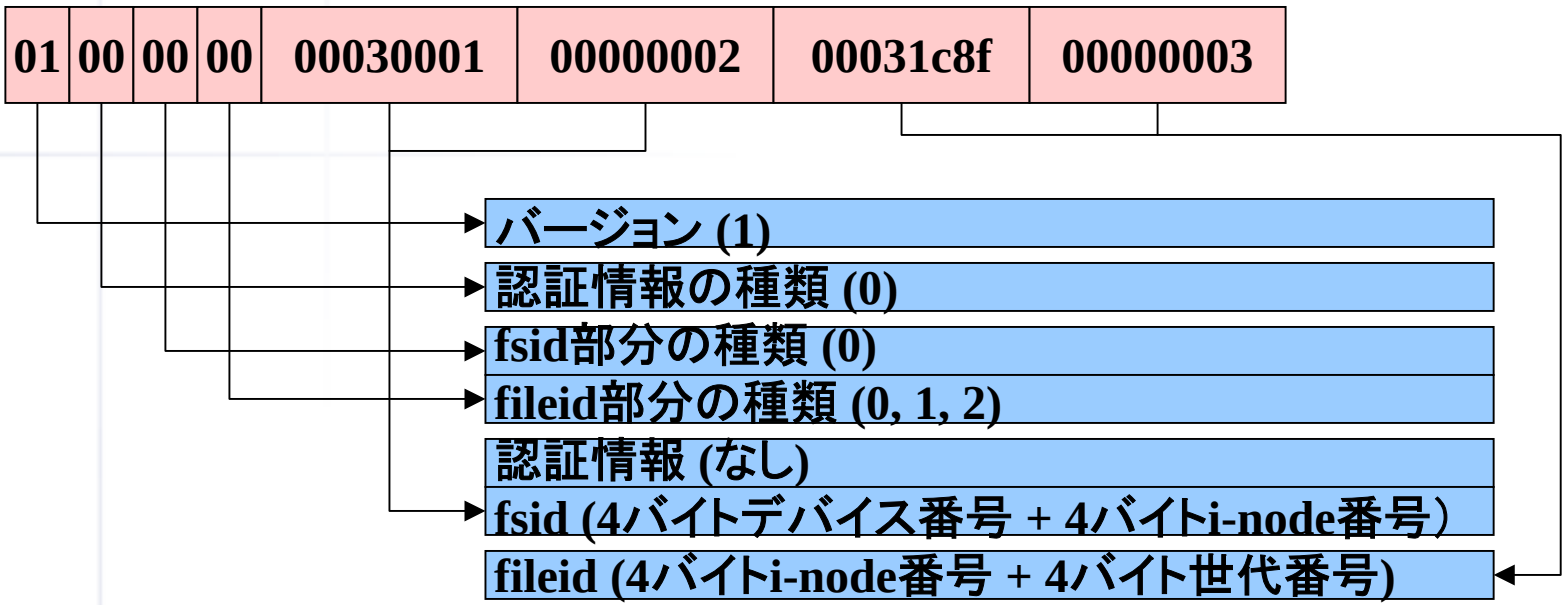
NFS要求(READ)

- NFS v3 のREAD要求形式(XDR表現)



ファイルハンドルの実装

- 対象ファイル(iノード)が一意に定まる
  - 形式はサーバ依存、クライアントからは見えない
  - サーバ再起動をまたいで永続的

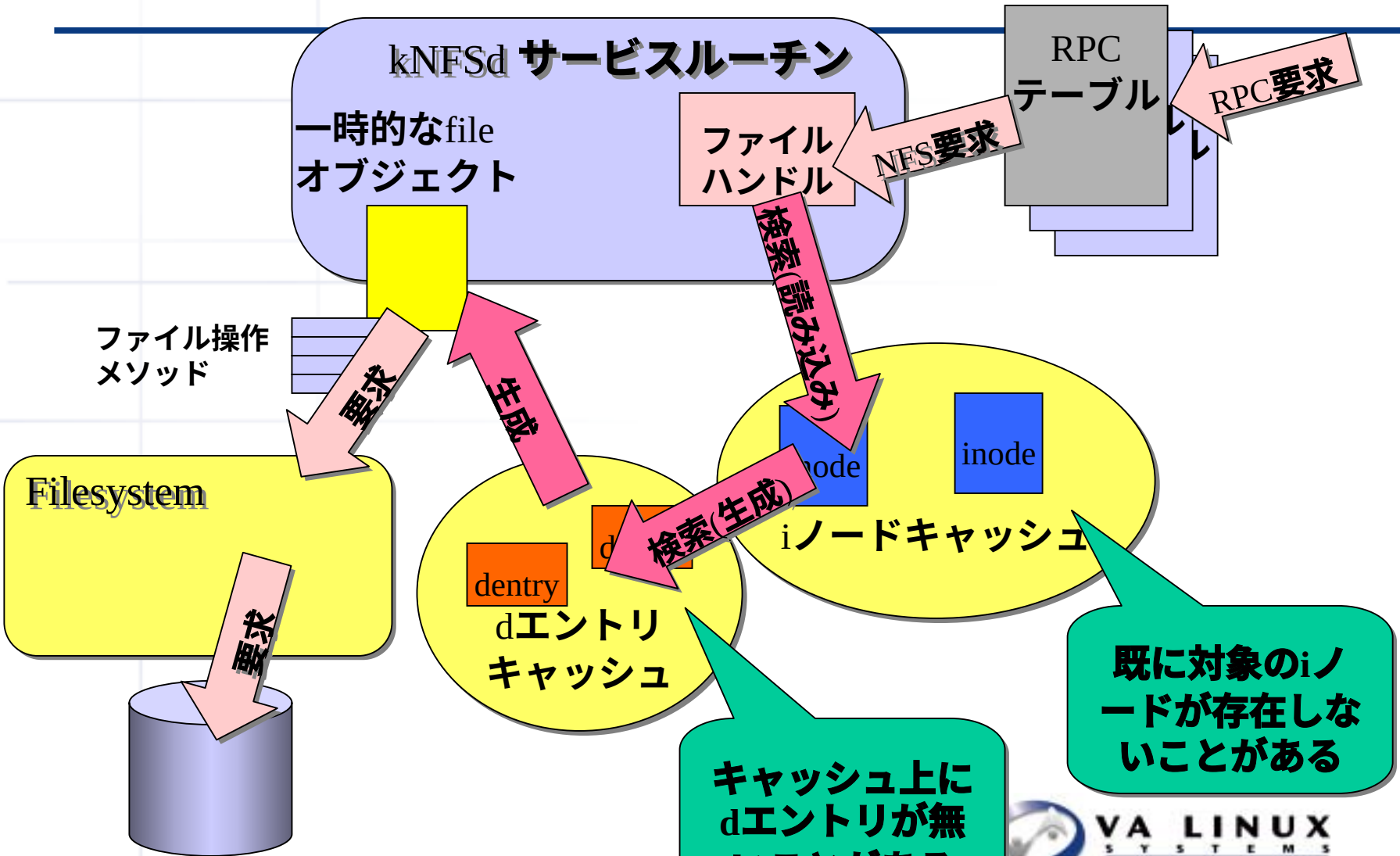


ファイルハンドルからファイル実体へ

- ファイルハンドルから、iノード検索
  - キャッシュから破棄されていることがある
  - 既に対象のiノードが存在しないことがある
  - 対象ファイルが、まさに削除処理中のことがある
- iノードからdエントリを検索
  - iノードに対応するdエントリがキャッシュ上に存在しないことがある
  - dエントリを生成し、既存dエントリツリーに登録
- iノードとdエントリをもとに、一時的なfileオブジェクトを用意

VA LINUX SYSTEMS JAPAN

ファイルハンドルから ファイル実体へ



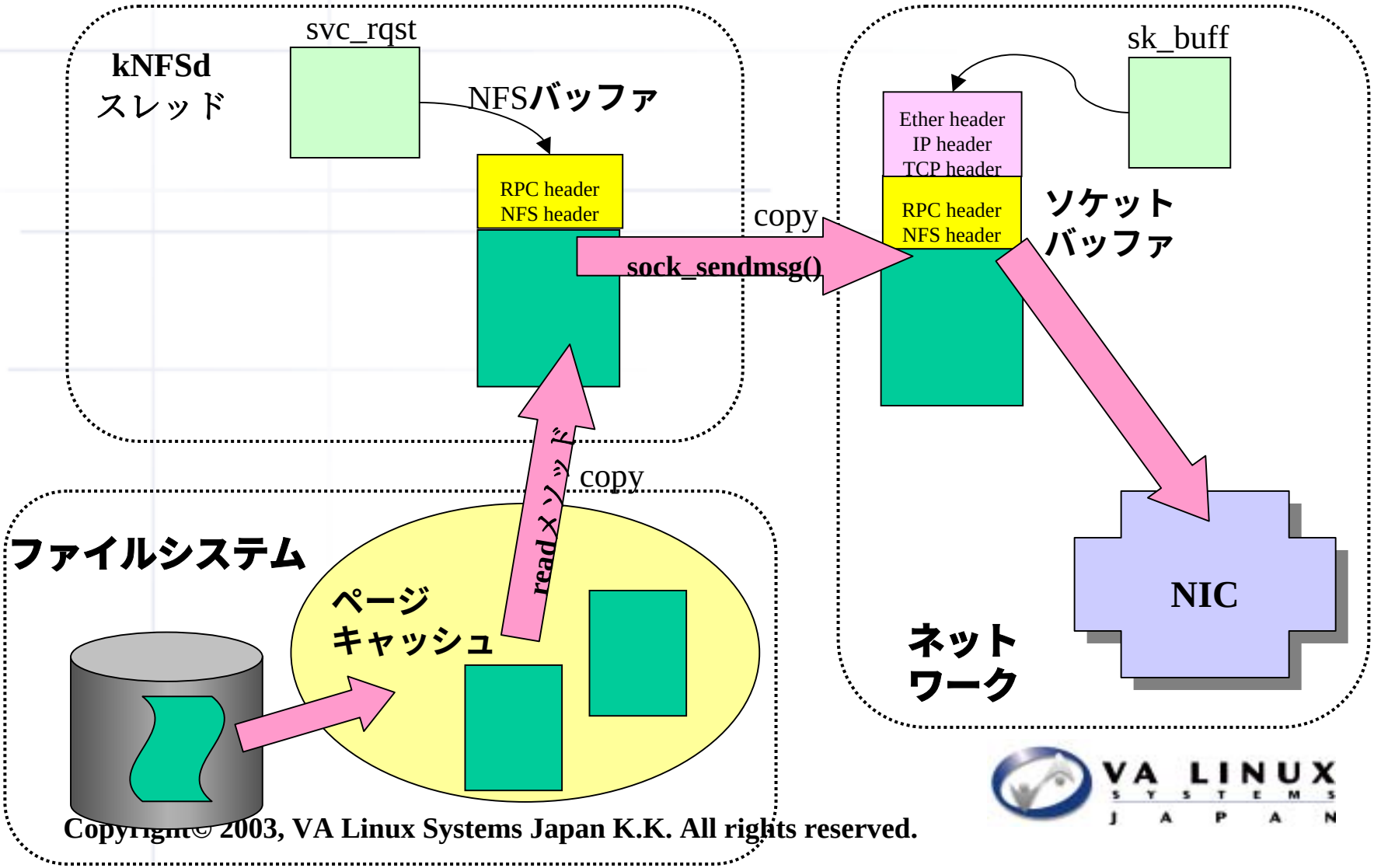
VA LINUX SYSTEMS JAPAN

ファイルシステムの呼び出し

- ファイルシステムを呼び出すためには、fileオブジェクト、dエントリ、iノードが揃っている必要がある。これらオブジェクトを利用して、ファイルシステム機能を呼び出す。
- READ要求、READDIR要求の場合、一度NFSバッファ内にデータを読み込む
- 先読み情報キャッシュ
  - ファイルオブジェクトが持つ属性。情報を失っても再構築できるようキャッシュしておく

VA LINUX SYSTEMS JAPAN

NFS read時のデータの動き



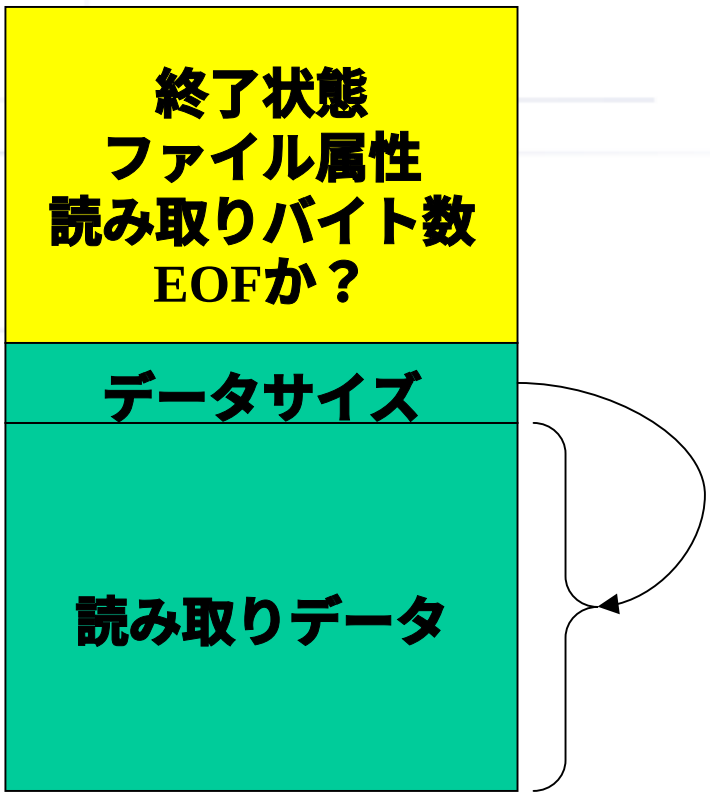
RPCパケットの送信

- NFSサービス処理の結果をXDR形式の応答パケットにまとめる
- 受信したソケットから、送り返す

VA LINUX SYSTEMS JAPAN

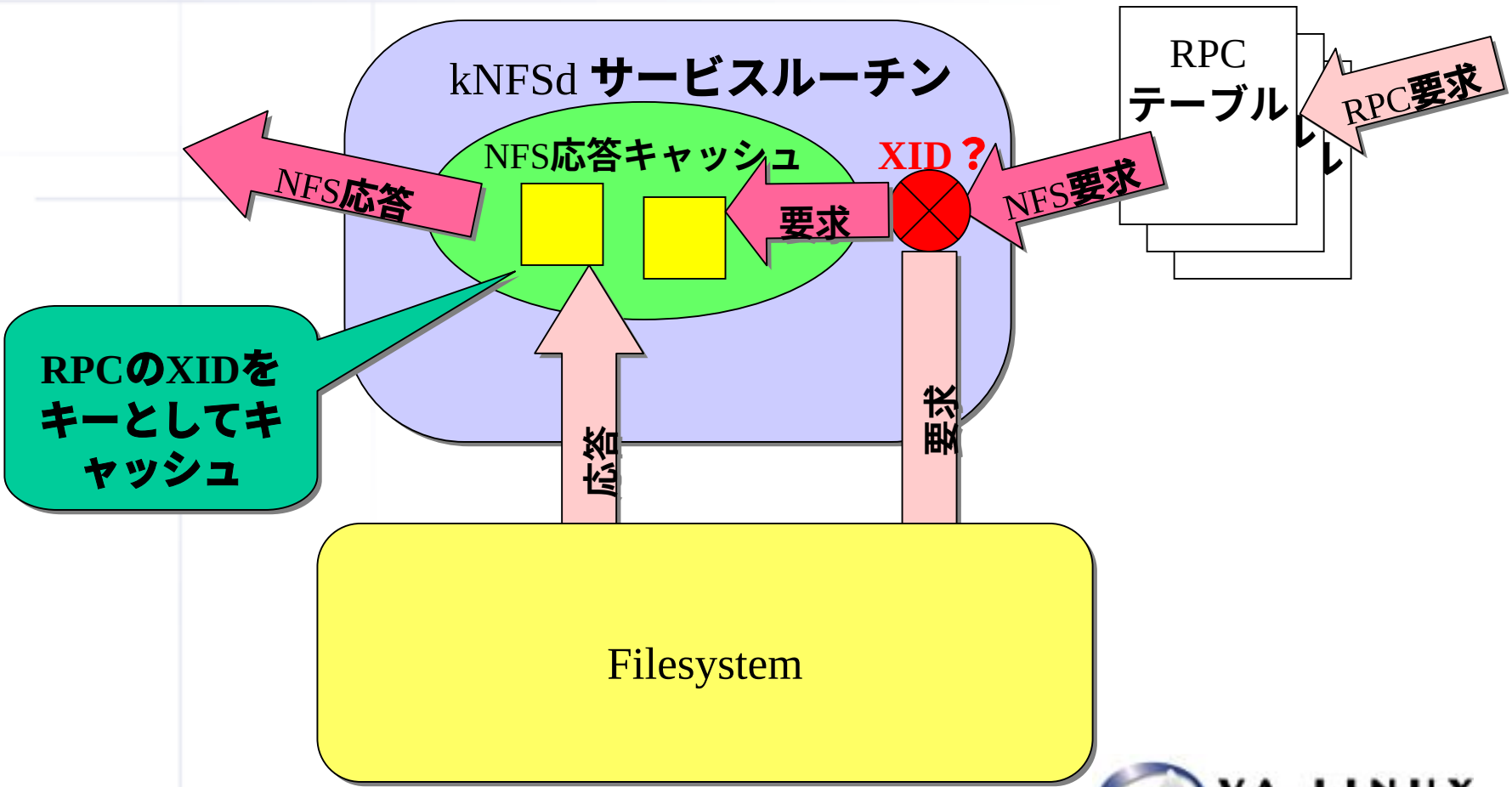
NFS応答(READ)

- NFS v3 のREAD応答形式(XDR表現)

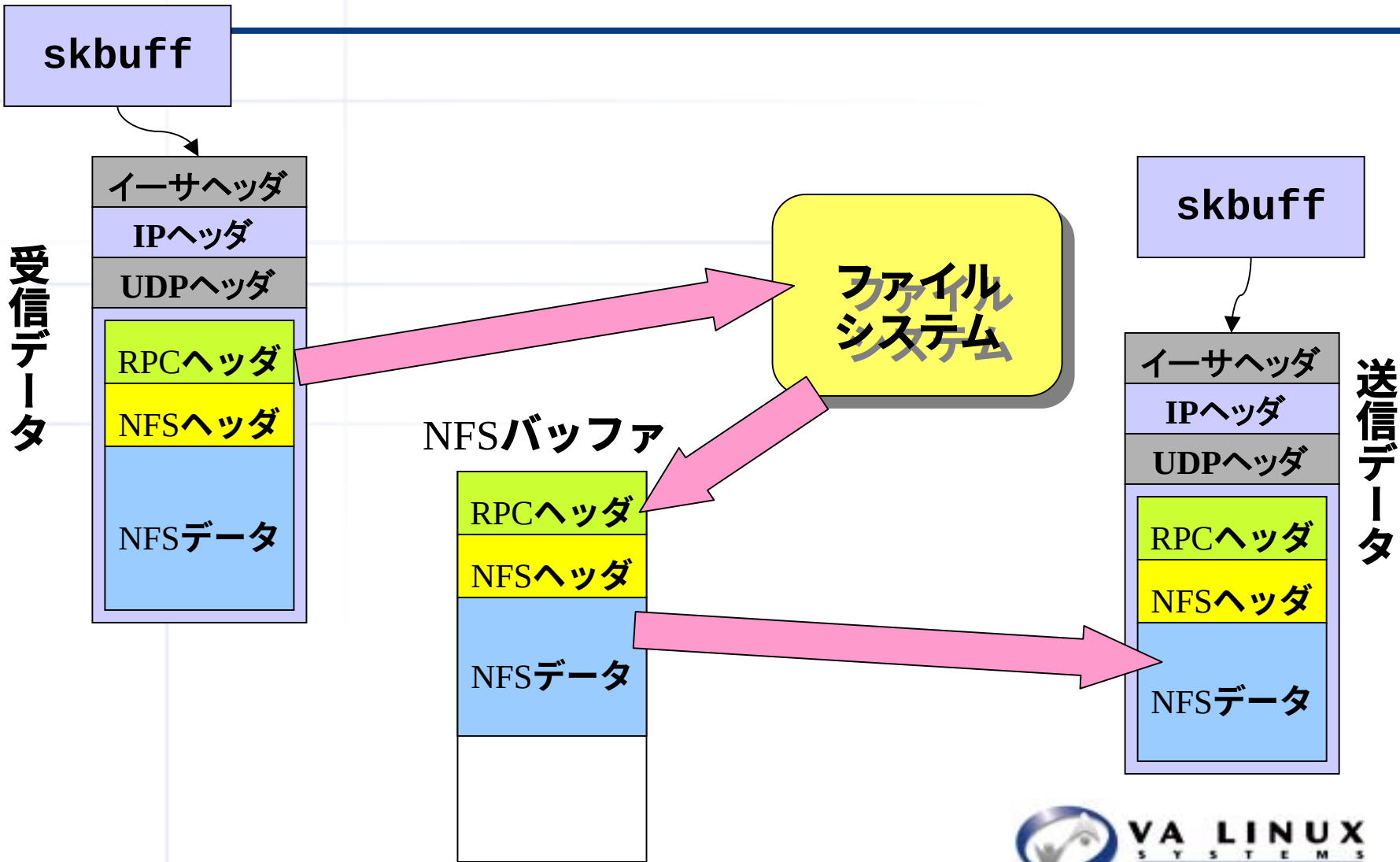


VA LINUX SYSTEMS JAPAN

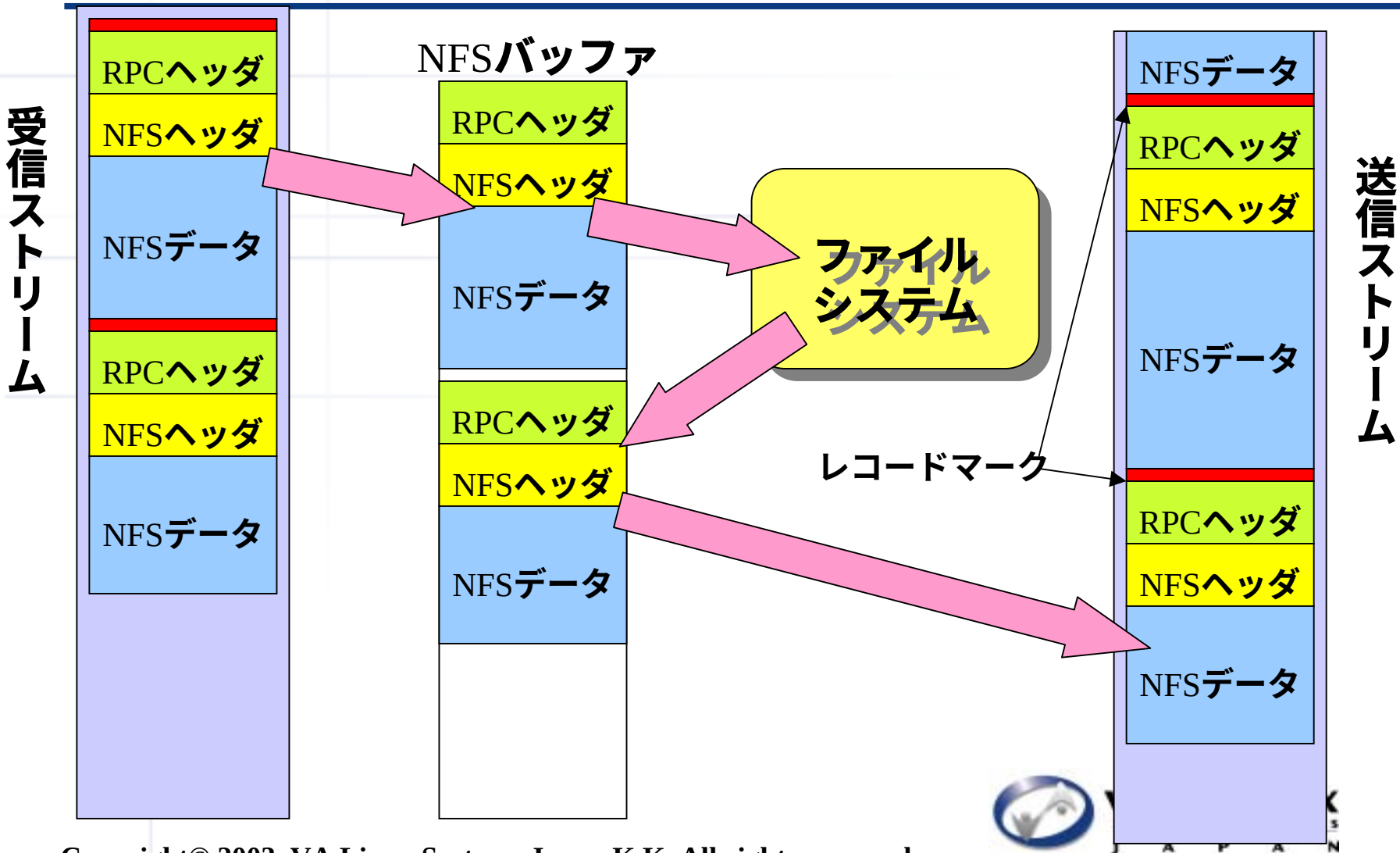
再送キャッシュ



UDPフレームとNFSバッファ



TCPストリームとNFSバッファ



Linux 2.6における NFSサーバの実装

Linux 2.6のNFS

- NFS over TCPの正式採用
- NFS v4対応(開発途上)
- 性能改善
  - SMP対応強化
  - ゼロコピー化
  - 一度に操作可能なデータサイズ拡大
  - 輻輳制御(NFS over UDPのクライアント)

VA LINUX SYSTEMS JAPAN

ゼロコピーNFSの実装

- データコピーの削減
  - ファイルシステムとkNFSdの間
  - kNFSdとネットワーク層との間
- 効果
  - データコピー時間そのものの削減
  - ハードウェアキャッシュ利用効率向上
  - メモリバスの解放
  - ソケットバッファ確保・解放のオーバヘッド削減

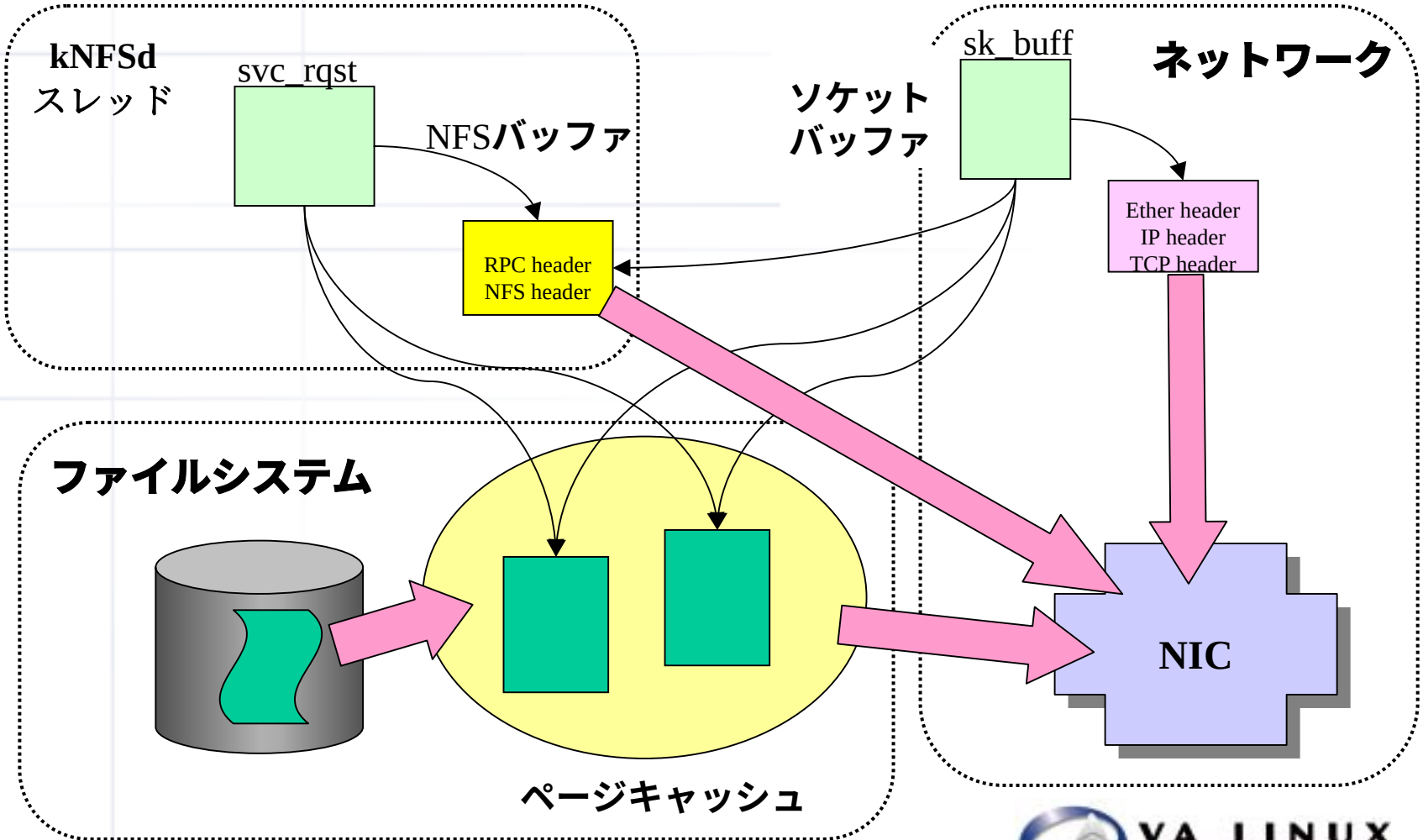
VA LINUX SYSTEMS JAPAN

Linux 2.6のNFSサーバの動作(READ)

1. NFS要求の受け付けとカーネルスレッドの起動
2. NFS要求のデコード
3. NFS要求に応じた、VFSファイルオブジェクトのメソッドの呼び出し
   - read要求の場合は、sendfileメソッドを呼び出し、ファイルオフセットに対応するページを取得
4. NFS応答の生成とエンコード
   - ページを確保、ページ上にRPC/NFSヘッダを生成
5. NFS応答の送信
   - inet層のsendpageメソッドを呼び出し、3と4のページを送信

VA LINUX SYSTEMS JAPAN

NFS read処理時のデータ構造



主な改造項目

1. VFSファイルオブジェクトのsendfileメソッドの拡張
   - コールバック関数を登録できるようにする。

2. INETレイヤのUDP送信処理の拡張
   - UDPフレームに対するsendpageメソッドの実装
   - UDP送信処理におけるCORK属性（MSG_MOREとUDP_CORK）のサポート
   - UDPフレームに対するハードウェアチェックサム機能実装

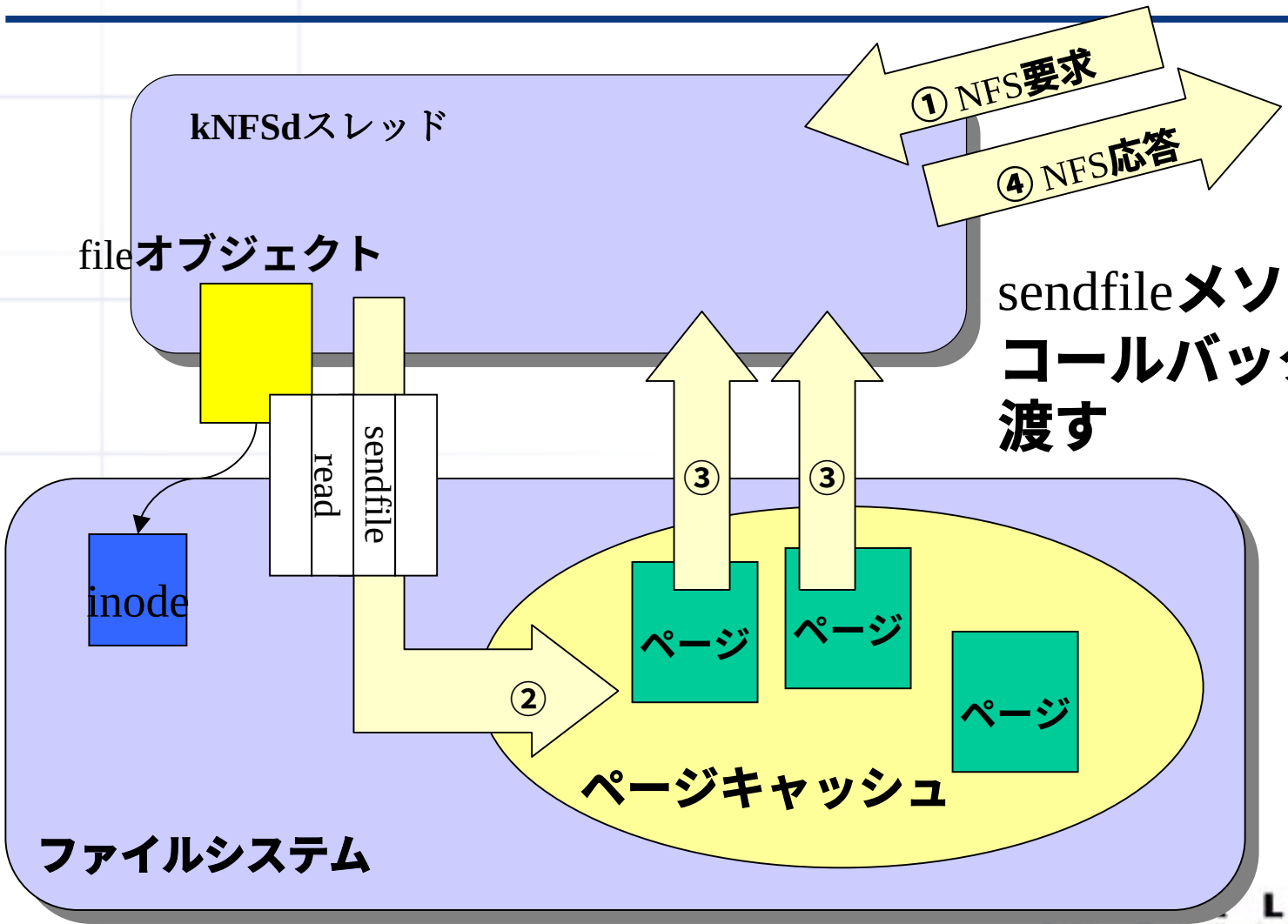
VA LINUX SYSTEMS JAPAN

主な改造項目

3. kNFSdが上記メソッドを利用するように変更
   - sendfileメソッドを利用し、ファイルに対応したページ獲得
   - RPCヘッダ、NFSヘッダも動的に確保したページ上に生成する。
   - 操作対象のページの管理機能
   - sendpageメソッドを利用し、ページ上のデータを送信。MSG_MORE属性を利用。

VA LINUX SYSTEMS JAPAN

kNFSdとsendfileメソッド

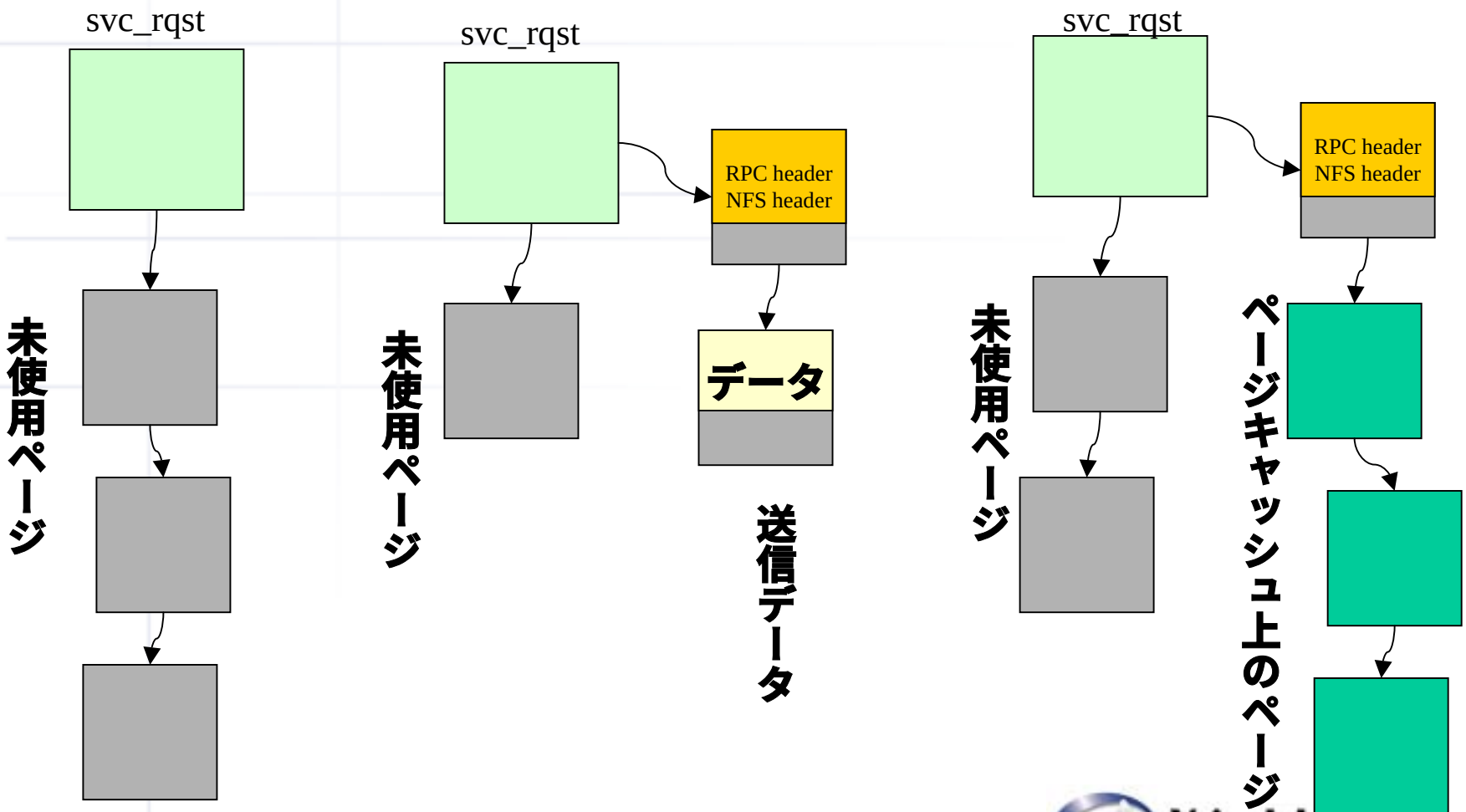


kNFSdとsendfileメソッド

```
nfsd_read(    )
{
                        :
    if (file.f_op->sendfile) {
            svc_pushback_unused_pages(rqstp);
            file.f_op->sendfile(&file, &offset, *count,
                                    nfsd_read_actor, rqstp);
    } else {
            oldfs = get_fs();
            set_fs(KERNEL_DS);
            vfs_readv(&file, vec, vlen, &offset);
            set_fs(oldfs);
     }
```


VA LINUX SYSTEMS JAPAN

NFSバッファの構造(送信データ)



kNFSdとsendpageメソッド

```
svc_sendto(  )
{
    if (rqstp->rq_prot == IPPROTO_UDP) {
                        :
            sock_sendmsg(sock, &msg, 0);
    }
    len = sock->ops->sendpage(.....);
                        :
    while (pglen > 0) {
            result = sock->ops->sendpage(.....);
                                :
```

VA LINUX SYSTEMS JAPAN

UDP用sendpageメソッド

```
struct proto_ops inet_dgram_ops = {
        .family =       PF_INET,
        .bind =         inet_bind,
        .sendmsg =      inet_sendmsg,
        .recvmsg =      inet_recvmsg,
        .sendpage =     inet_sendpage,
};
struct proto udp_prot = {
        .name =          "UDP",
        .sendmsg =      udp_sendmsg,
        .recvmsg =      udp_recvmsg,
        .sendpage =     udp_sendpage,
};
```

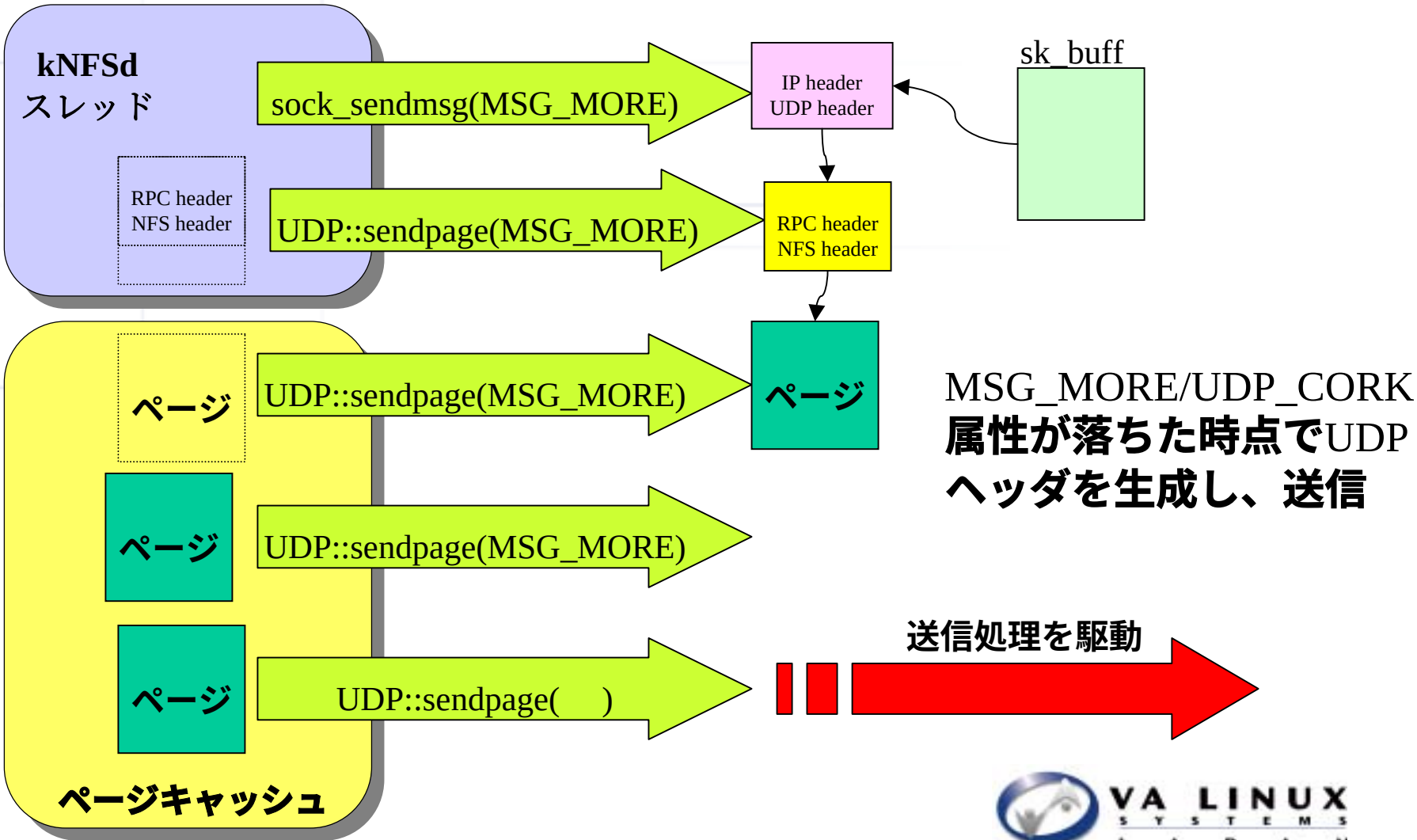

VA LINUX SYSTEMS JAPAN

UDP用sendpageメソッド

- TCP用sendpageと同じインターフェイス
- CORK属性との併用が前提
  - 単体での利用も可能だが、あまり有用ではない

VA LINUX SYSTEMS JAPAN

UDPソケットのCORK属性



MSG_MORE/UDP_CORK
属性が落ちた時点でUDP
ヘッダを生成し、送信

VA LINUX SYSTEMS JAPAN

UDPソケットのCORK属性

- CORK機能はIPレイヤで実現
  - TCP用CORK属性インターフェイスを踏襲
  - IPフラグメントを生成しつつ、ソケットにデータを蓄える。データはsendmsgで送られたものであっても、sendpageで送られたページであっても良い。
  - CORK属性が外れたところで、UDPヘッダを生成。必要があればソフトウェア的にチェックサム計算を行う。UDPヘッダ生成後、IP送信処理をキックする。
- UDP以外のプロトコルにも適用可能
- NFS以外のサービス(ストリーミングなど)でも有効

VA LINUX SYSTEMS JAPAN

本家Linuxでの採用(1/2)

- Linux 2.5.7用のパッチを公開
  - ファイルシステムとkNFSdの間でのゼロコピー化
  - TCP利用時の送信処理のゼロコピー化
- Linux 2.5.33用パッチを公開
  - UDP送信処理のsendpage対応
  - readdir応答送信処理のコピー削減
- Linux 2.5.43
  - NFSサーバ用パッチ取り込み開始
- Linux 2.5.47
  - NFSサーバ用パッチの取り込み完了

VA LINUX SYSTEMS JAPAN

本家Linuxでの採用(2/2)

- Linux 2.5.70用
  - NFSクライアント側のゼロコピー化パッチ取り込み
- その他、バグフィックスパッチを多数コミット

VA LINUX SYSTEMS JAPAN

効率的動作の条件

- ファイルシステム
  - ページキャッシュベースのファイルシステム
  - sendfileメソッドを定義していること
- ネットワークカード
  - スキャタギャザI/Oのサポート
  - ハードウェアチェックサムのサポート
- 豪華なハードウェア
  - 大量のディスクと高速なSCSIバス
  - Gbitイーサ、ジャンボフレーム
  - 高速なバス
  - 大容量メモリ

VA LINUX SYSTEMS JAPAN

実装における注意点

実装上で注意した点

- Ⓠゼロコピー機能において、ネットワーク送信キューが参照しているページが書き換えられることがあるが大丈夫か？NFSのread処理中にデータが更新されても問題ないのか？
- Ⓐローカル環境であっても、readシステムコールの実行中にwriteシステムコールで書き換えられることがある。ユーザプロセスから見るとどちらも同じ。

VA LINUX SYSTEMS JAPAN

実装上で注意した点

Ⓠ送信時のハードウェアチェックサム計算時にページ内容が書き換えられると、チェックサム計算を間違わないか？

Ⓐ間違わない。イーサコントローラ内のバッファに読み込んだ後に計算される。

VA LINUX SYSTEMS JAPAN

実装上で注意した点

- Ⓠ RPCパケット、IPデータフレームは1バイト単位での送信ができない。大丈夫か？
- Ⓐ 奇数バイトのread要求時は、利用していないページの一部をゼロクリアし、パディング領域と利用することとした。

VA LINUX SYSTEMS JAPAN

実装上で注意した点

- Ⓠ NFS応答の送信処理中に、送信対象としているファイルのページキャッシュが解放されることはないか？また、ファイルそのものが削除されても問題ないか？
- Ⓐ 送信処理中のページは参照数をあげているので、送信完了まで解放されることは無い。ファイル本体が削除されたときも同様である。

VA LINUX SYSTEMS JAPAN

次の課題と目標

性能改善へ向けての課題

- UDP送信処理の改善
  - IPフラグメントが必要な大きなUDPフレームに対するハードウェアチェックサムの利用
  - NICの種類(NETIF_F_FRAGLIST属性)によっては実現可能となるはず。現実には、NETIF_F_FRAGLIST属性を持つNICドライバは、まだ作成されていない。ハードウェア仕様的には、acenicなら可能と思われる。

VA LINUX SYSTEMS JAPAN

性能改善へ向けての課題

- NFS write要求の高速化
  - ファイルオブジェクトのwritevメソッドの有効利用。フラグメント化されたUDPデータグラムから、直接ページキャッシュにデータを転送
  - crypto処理との整合性を考えてから実装
- IPv6対応
  - 検討は始めた人がいる。IPv6のUDPスタックはsendpageメソッドをサポートしていない
- NFS v4対応
  - チューニングの余地

VA LINUX SYSTEMS JAPAN

性能改善へ向けての課題

- NFSクライアントの性能改善
  - Cacheファイルシステム
    - クライアント側でのキャッシュ機能を強化、NFSサーバの負荷軽減

VA LINUX SYSTEMS JAPAN

おまけ

詳解Linuxカーネル第二版出版
(Oreillyジャパン社)
- Linux 2.4の解析文献
- Linux 2.6の実装にも軽く触れる

会場の入り口にて先行発売
- １０％引き、消費税サービス

VA LINUX SYSTEMS JAPAN

Linux V2.6リリース目前！

V2.7カーネル開発参加者募集

VA LINUX SYSTEMS JAPAN

NFSv3

- スケーラビリティの向上
  - 64bitファイルポインタ、ファイルシステムサイズ
  - ファイルハンドルの拡大
- ロックファイルの作成 (creat(2)の**O_EXCL**)
- 性能強化
  - 遅延書き込み(COMMIT操作の採用)
  - READDIRPLUS操作
  - I/Oサイズの拡大(ネゴシエート方式)
- クライアント・サーバ間のネゴにより、両者のサポートしている最も高いバージョンのものを利用

VA LINUX SYSTEMS JAPAN

NFSv4

- 現行NFS (v2、v3) の課題を解決
  - 広域利用の考慮、性能改善
    - COMPOUNDメッセージ
    - Delegationによるキャッシュ
  - Windowsとの相互運用性
    - Windows ACLモデルをサポート
  - 強力なセキュリティ機能
    - GSS-APIによる認証、同一性保証、情報秘匿 (暗号化)
  - そのほか
    - MOUNTプロトコル廃止
    - ロックデーモンの廃止
    - オープン状態を持つ(ステートレス性の放棄)

VA LINUX SYSTEMS JAPAN