VA Linux Kernel Forum
Linuxチューニング

Linux World Expo/Tokyo 2002

VA Linux Systems Japan

高橋　浩和

taka@valinux.co.jp

目次

- チューニングの考え方
- 簡単なチューニング
- 本格的なチューニング

VA LINUX SYSTEMS JAPAN

チューニングとは？

- スループット向上
- 応答性確保
- スケーラビリティ向上
- 高負荷時の性能維持

VA LINUX SYSTEMS JAPAN

歴史的なしがらみ

- ユニプロセッサ用として実装されたことによる問題
- 非力なマシン用に設計されていたことによる問題
- プロトコル等の仕様自体の問題

VA LINUX SYSTEMS JAPAN

チューニングレベル

- メモリ増設、ハードウェア追加
- 運用レベル
- カーネルレベル
  - /proc、カーネルconfig、カーネルの変更

スループットとスケーラビリティ向上

- 目つくボトルネック
  - I/OネックでCPUが遊んでいる
  - CPU増設しても性能があがらない

限界までの性能を引き出す場合はカーネルチューンが必要

VA LINUX SYSTEMS JAPAN

前提知識

- カーネル内には競合資源が非常に多い
- CPU、キャッシュ、物理メモリ、I/Oの速度差
- ハードウェア(I/Oコントローラ、バス)能力の限界
- システムコール処理の重さ

準備

- 利用目的、用途に合わせたチューニング
- ボトルネックの検出
  - プロファイリング
  - 過去の経験の積み重ね

ポイント

- キャッシュ
  - 優先的にキャッシュすべきオブジェクトは？
- 遅延処理、Lazy処理
- 処理並列度向上
- ハードウェア機能の有効活用

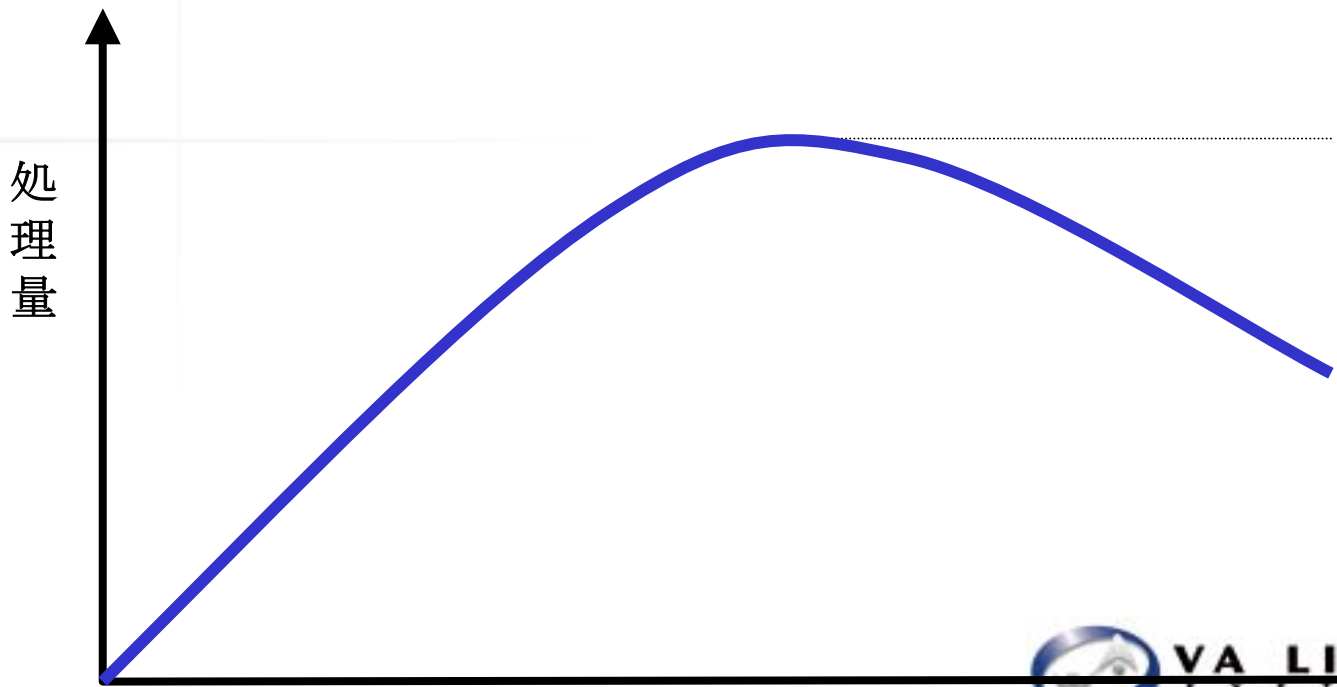
ロック粒度の変更例

- スケジューラの交換
  - V2.5 0(1)スケジューラなど
- SMP対応VM
- タイマ処理の交換
  - CPUローカルタイマーリスト
- デバイスごとのI/Oロック
  - V2.5 BIO機能

VA LINUX SYSTEMS JAPAN

高負荷時の性能維持

- 限界を過ぎると性能が悪化する
- Best Effort型実装による問題点
  - 全ての要求を受け入れ、最大限実行を試みる



高負荷対策

- 資源の枯渇への対応
  - 高負荷の検出と早めの対策
  - 高負荷要因の除去
- 突発的負荷への対応

その他のポイント

- 大容量メモリ
  - 動作効率と利用効率
  - HighMemの扱い
  - 超大容量に向けＶＭ
- デバイスのスケーラビリティ
  - 多くのデバイスの同時操作、I/Oスケジューリング

応答性向上

- 時間保証
  - 事象発生から対応する処理の完了までの時間の保証
- 応答性を追求するととスループットは落ちる

問題点

- 不確定なCPU割り当て
  - プロセス、割込み
  - プライオリティ逆転
- カーネル内ノンプリエンプティブ
  - 2.5ではExperimental実装
- 処理時間不定
- 粗いタイマー粒度
- I/OとCPUの速度差

チューニング対象（主なもの）

- プロセススケジューリング
- 割込みスケジューリング
- 資源割り当て方式

課題）

- 応答性とスループットの両立
- 従来システムとの互換性
- ハードウェア自体の問題

簡単なチューニング(例)

VA LINUX SYSTEMS JAPAN

簡単なチューニング　（例）

- 運用による解決
- /proc
- ブートパラメータ
- モジュール起動パラメータ
- カーネル内マジックナンバ

運用による解決例

- 不要なサービスは止める
- cronによる突発的高負荷
- ファイルシステムの使い方
  - 適切なファイルシステム選択
  - ファイルシステムの配置
  - ファイルシステム利用率
  - sync/asyncマウント
  - tmpfsとramdisk

VA LINUX SYSTEMS JAPAN

運用による解決

- 適切なハードウェア選択
  - I/Oバウンドな処理では、高速なディスクは重要
  - RAIDモードの選択
  - SCSIホストバスアダプタの選択
  - 高いNICは伊達じゃない
  - PCIバスの速度とバス幅
  - 最新ハードか？実績のあるハードか？

VA LINUX SYSTEMS JAPAN

殿様的解決

- とりあえず、お金で解決
  - メモリ増設
  - SMPマシンの導入、マシンの増設
  - 前述したハードを片っ端から増設

VA LINUX SYSTEMS JAPAN

/procによるチューンの例

実行例）

- echo 8192 > /proc/sys/kernel/threads-max
- sysctl -w kernel.threads-max=8192
- /proc/sys以下は、/etc/sysctl.confに書いておくことも可能。
  - cat /etc/sysctl.conf
    kernel.threads-max=8192

VA LINUX SYSTEMS JAPAN

/proc/sys/kernel

- threads-max
- shmmax
- shmall
- msgmax
- msgmnb
- sem

/proc/sys/vm

- bdflush
- pagetable_cache
- page-cluster
- max-readahead

/proc/sys/fs

- file-max

/proc/sys/net/core

- rmem_max
- wmem_max
- rmem_default
- wmem_default
- netdev_max_backlog
- hot_list_length

/proc/sys/net/ipv4

- ip_local_port_range
- その他、各種タイムアウト値、リトライ回数、最大資源量など様々な値を設定可能なproc達各種

VA LINUX SYSTEMS JAPAN

その他

- /proc/mtrr
- 各種デバイスドライバが用意する/procも利用可能

VA LINUX SYSTEMS JAPAN

bootパラメータ

- カーネル内で__setup()で指定されたものはカーネル起動時に変更可能

- /etc/lilo.confへの記述例

  append="mem=５１２M“

  （もしくは/boot/grub/grub.conf）

VA LINUX SYSTEMS JAPAN

bootパラメータ

- rootflags=xxxxx
- ramdisk=# # # #
- max_loop=# # # #
- idle=xxxx
- sg_def_reserved_size=# # # #

VA LINUX SYSTEMS JAPAN

モジュール起動パラメータ

- カーネル内にMODULE_PARM()で指定されたものは変更可能

例)

  - Insmod scsi.o max_scsi_luns=8

モジュール起動パラメータ

- def_reserved_size
- do_sync_supers
- nlm_timeout

VA LINUX SYSTEMS JAPAN

モジュール起動パラメータ

- 各ドライバは設定変更ができるよう、各種変更可能パラメータを用意している。

　bootパラメータとモジュール起動パラメータのどちらでも変更可能となっているものが多い

VA LINUX SYSTEMS JAPAN

カーネル内マジックナンバ（例）

- カーネル内のマクロや、変数初期化のじか書き
  /proc、ブートパラメータ、モジュールパラメータに出ていないものはカーネルコードを直接編集（危険）

- 変更例
  block I/Oのエレベータキューの長さ
  kmap領域の拡大(PTE確保アルゴリズムの方も変更が必要)
  TCP　ACK遅延時間変更
  カーネルスタックの拡張（結構面倒だ）

注意

- /proc、 bootパラメータ、カーネルマクロはしばしば変更になる。また変更すると危険なものもある。

しかし……

- アプリケーションの作りが悪すぎて、システム側では手の打ちようのないものも、この世に存在する
- コンピュータの性能に頼りきった設計のＡＰが多い

本格的なチューニング(例)

VA LINUX SYSTEMS JAPAN

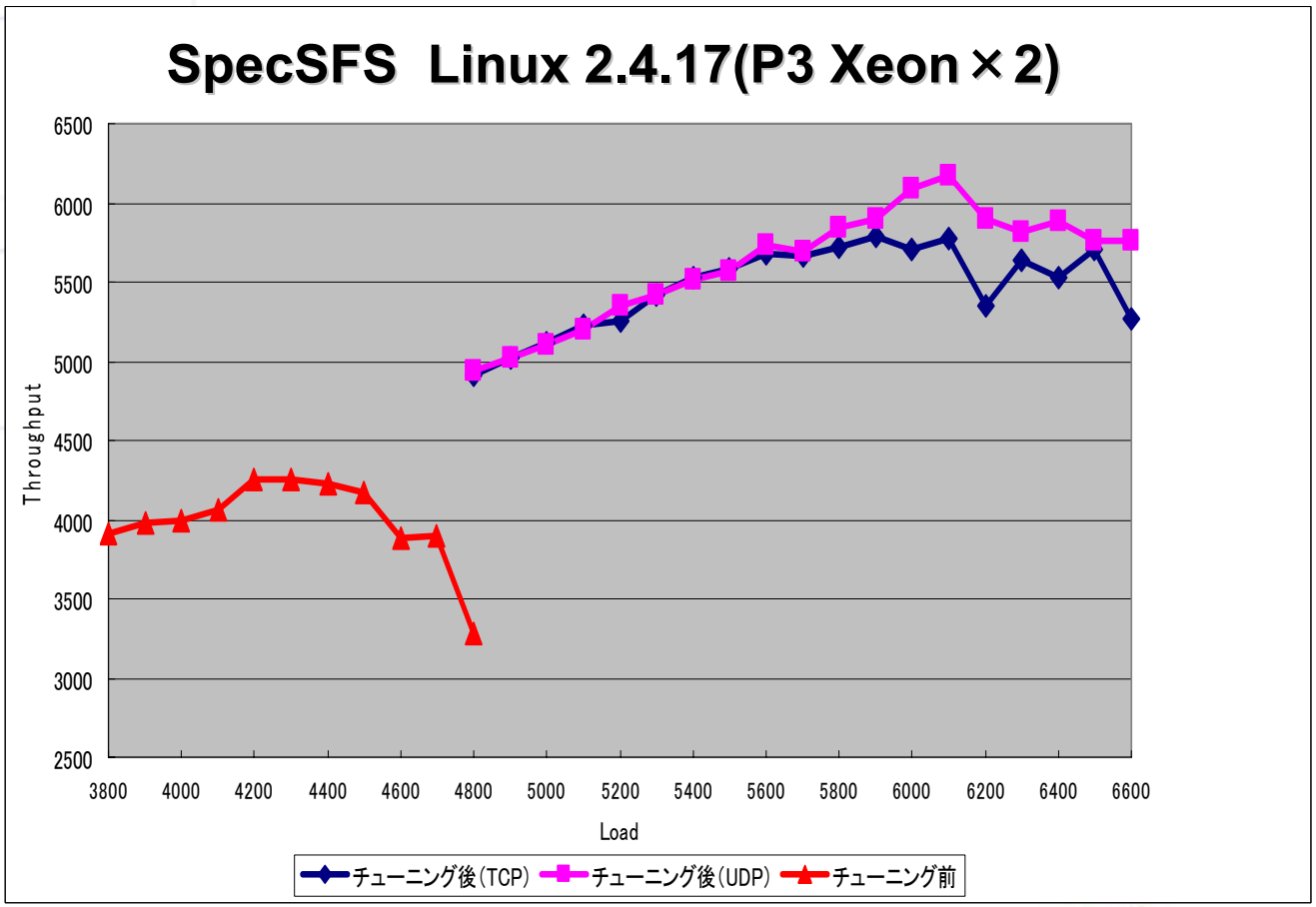
実例：NFSDハードチューン

- スループット向上: ネットワークからファイルシステム、デバイスドライバに至るまで全てが改善の対象（実装継続中）

- NFSが利用する資源を優先的に拡大
- SMPロック粒度の見直し
- 高負荷対策
- アルゴリズム自体の変更

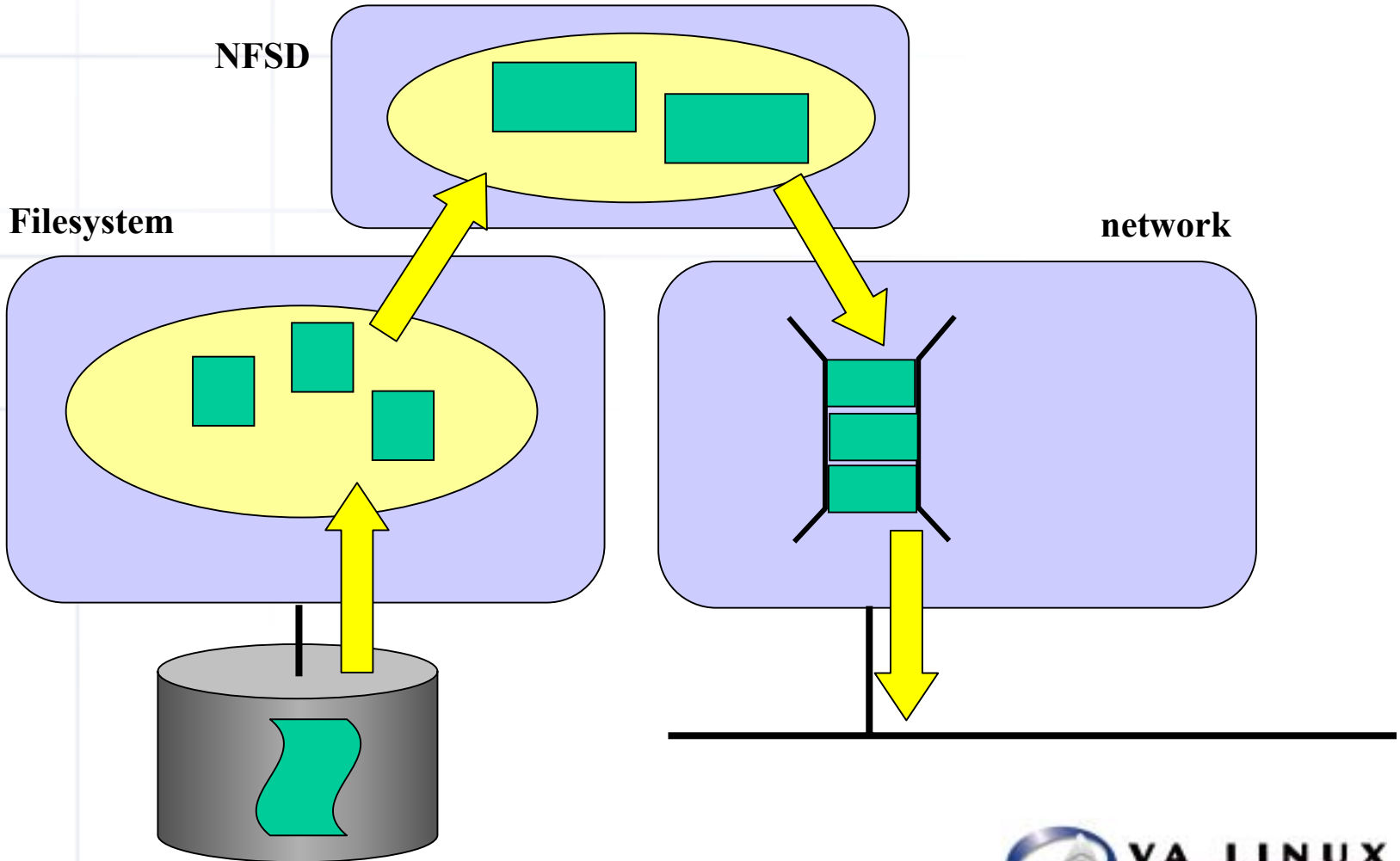
VA LINUX SYSTEMS JAPAN

チューニング例）NFSサーバ性能



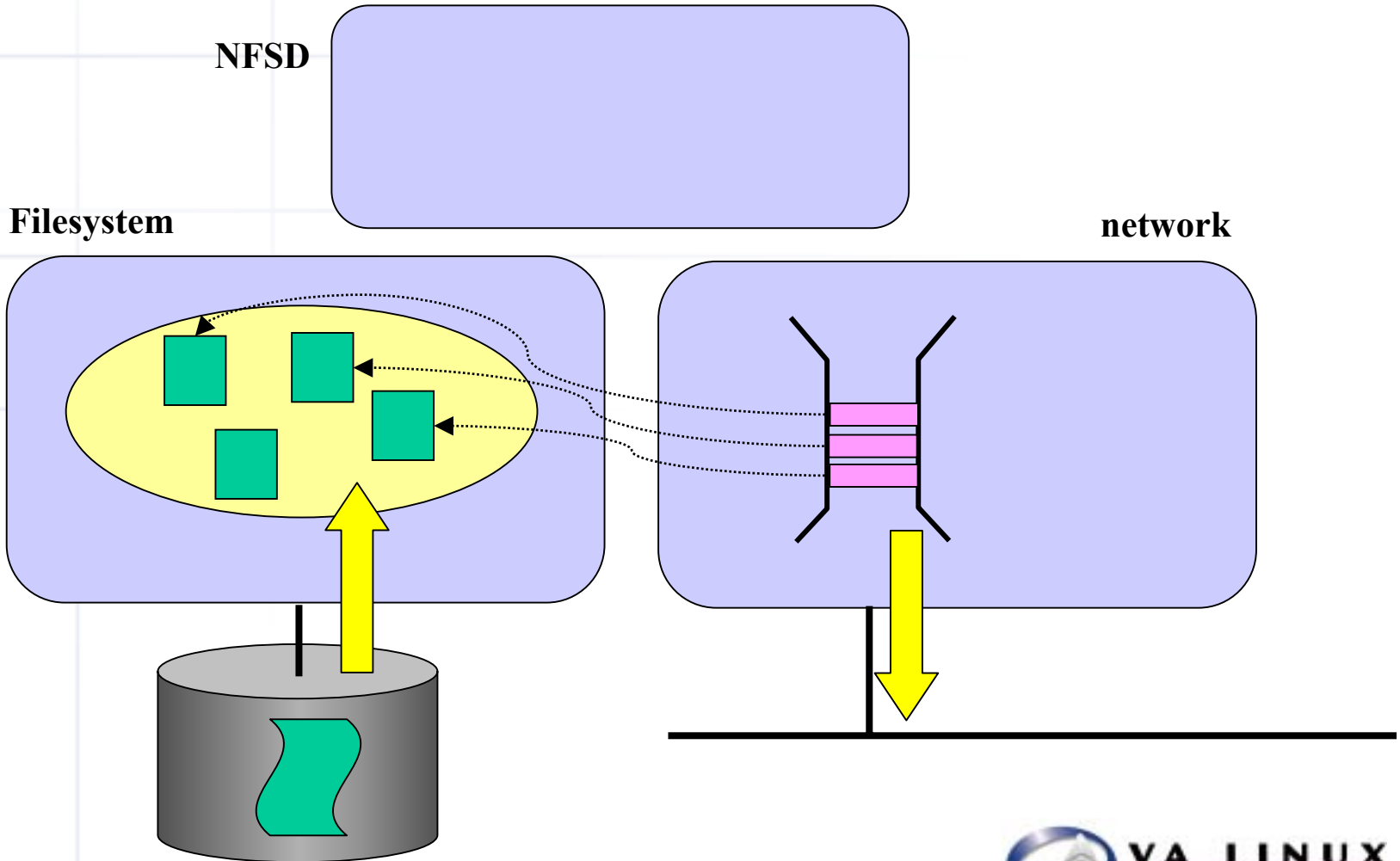
VA LINUX SYSTEMS JAPAN

ゼロコピー



VA LINUX SYSTEMS JAPAN

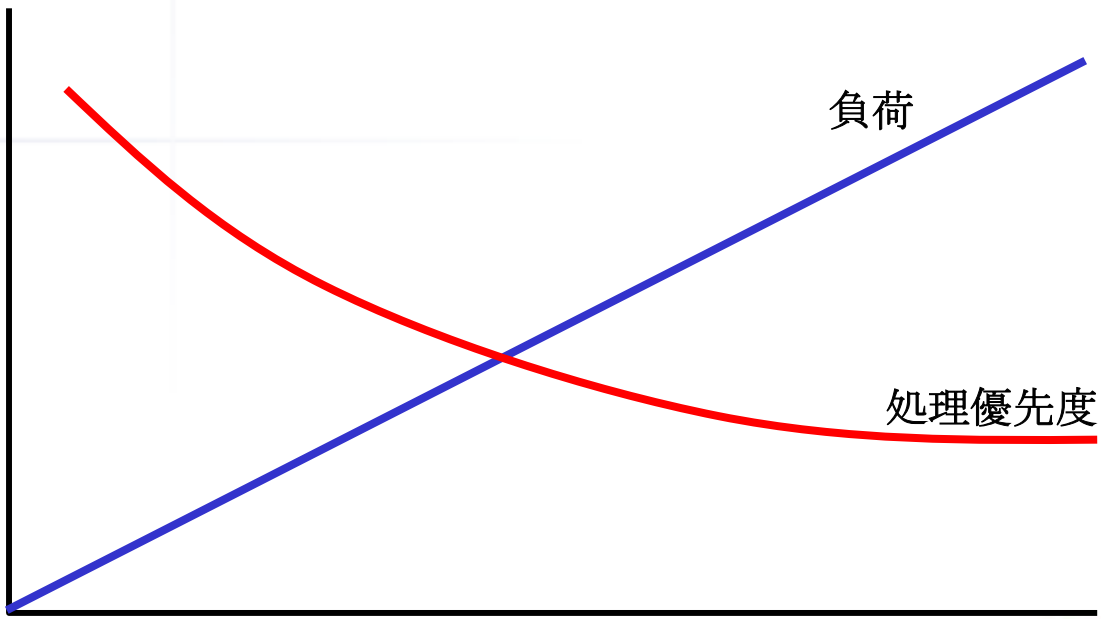
ゼロコピー



VA LINUX SYSTEMS JAPAN

スケジューリング

- カーネルスレッドのプライオリティ維持



VA LINUX SYSTEMS JAPAN

v2.5に向けて

- v2.5の開発目標の一つがスケーラビリティ向上。既に開発が始まっている。
  - プロセススケジューラのスケーラビリティ向上
  - ブロックI/Oレイヤの改善
  - SMPロック粒度の改善
- ハイエンド分野向けにはまだまだ改善の余地がある

VA LINUX SYSTEMS JAPAN