Containers/cgroups BoF: サーバから組込みまで

“Container actually is all around”

日本電気株式会社
システムプラットフォーム研究所
高橋 雅彦

2008/09/12

Japan Linux Conference 2008

発表の流れ

- 背景 ～仮想化の必要性～
- 仮想化手法
- OS レベル仮想化（コンテナ）とは
- コンテナの各機能の説明
  - 名前空間（と、その実装例）
  - 資源管理（と、その実装例）
  - checkpoint/restart

※ 発表の途中でも自由に割り込んで、発言をどうぞ

背景

- IT 化によってサーバがメインフレームからスケールアウトしたら、台数が多くなりすぎてその管理コストが無視できなくなってきた
- 一方、ハードウェア（特に CPU ）も急激に進歩し、余力ができた

→ サーバの統合が現実的な解。ただし、既存システムをなるべく変更なしでそのまま移行したい＆システムの安定性も必要

→ 仮想化技術の活用

… しかし、Linux には仮想化及びその周辺技術が充分に揃っているとはまだまだ言えない

仮想化環境実現に必要な機能

- 仮想化環境として最低限必要な機能
  - ハードウェア資源の共有
  - “ コンテキストスイッチ”
- 仮想化環境の“ isolation”に必要な機能
  - メモリなどのリソースの保護・セキュリティ（名前空間の独立）
- 仮想化環境をきちんと運用するために必要な機能
  - 資源の使用量管理

仮想化の手法

主にサーバで使われている仮想化技術はその実現レイヤによって様々

- エミュレータ (CPU 命令セットレベル)
  - bochs, qemu
- HW 仮想化 (full-virtualization)
  - VMWare, VirtualPC
- システムコール・特権命令エミュレーション
  - User-mode Linux
- ハイパバイザ型 (full- or para-virtualization)
  - Xen, VMWare, KVM, lguest, ...

} ゲスト OS on ホスト OS

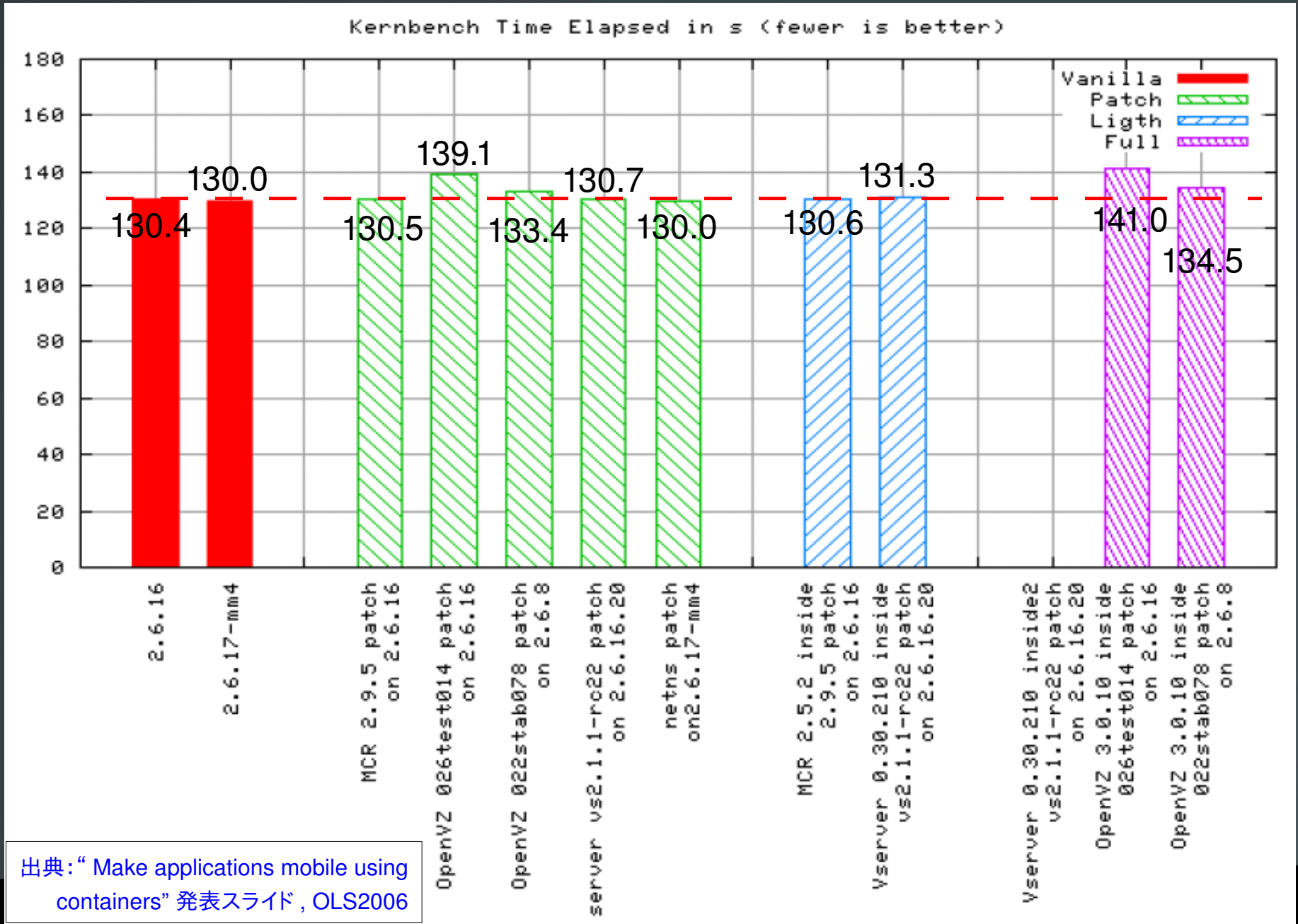
- OS レベル仮想化
  - コンテナ (OpenVZ, MCR など)

→ ホスト OS のみ

コンテナ（OSレベル仮想化）の特徴

- 一つの（ホスト）OS のみが動作する
  … 異なる OS のプログラムを実行することはできない
- 仮想化部分が微小なので、仮想化オーバヘッドが小さい
- 名前空間や資源制御などの仮想化に必要な機能が提供される（予定）
  - c.f. プロセスは、「アドレス空間が独立した実行環境」
- Linux 以外の実装例としては、Solaris10 の「 Solaris ゾーン（名前空間）と Solaris リソースマネージャ（資源管理）」がある

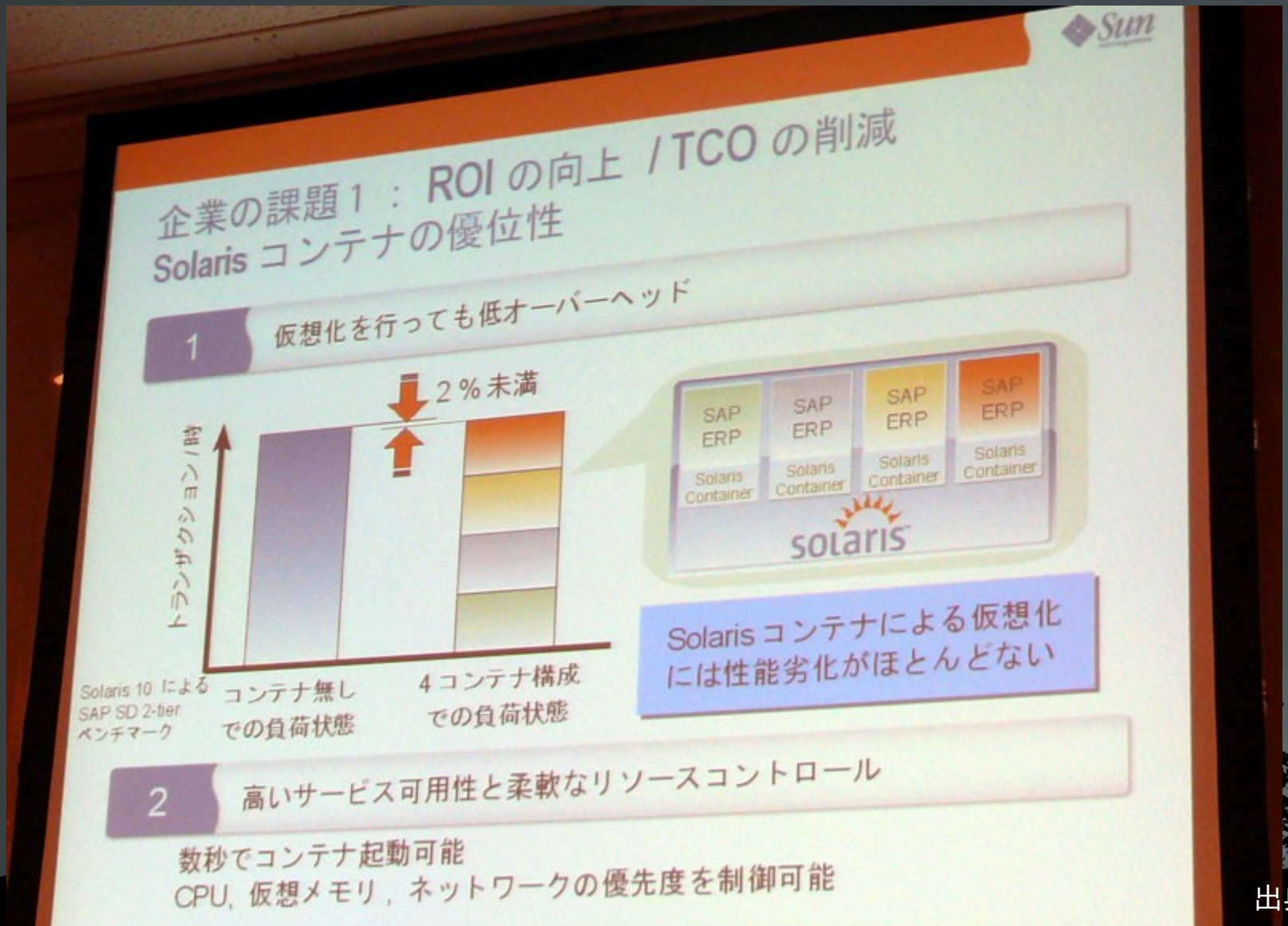
コンテナのオーバヘッド～ Linux



出典:“ Make applications mobile using containers” 発表スライド , OLS2006

コンテナのオーバヘッド～ Solaris

Sun(Solaris) の実装では、オーバヘッドはわずか約 1.5% (SAP ベンチ)



出典: Enterprise Watch

コンテナを実装するために

コンテナの実装は、つまり、OS の機能をネストすることを意味する。
　OS の機能はハードウェア資源を抽象化・独立空間化・管理すること

- 抽象化: HW を使いやすくするインターフェースを提供
  - ファイルシステム:ファイル (open/read,write) → セクタ読み書き
  - ネットワーク:ソケット (connect/send/receive) → パケット通信
- 独立空間化
  - ファイルディスクリプタ、(メモリアドレス空間 - MMU)
- 使用量管理
  - CPU 時間、メモリ制限、ディスク容量制限

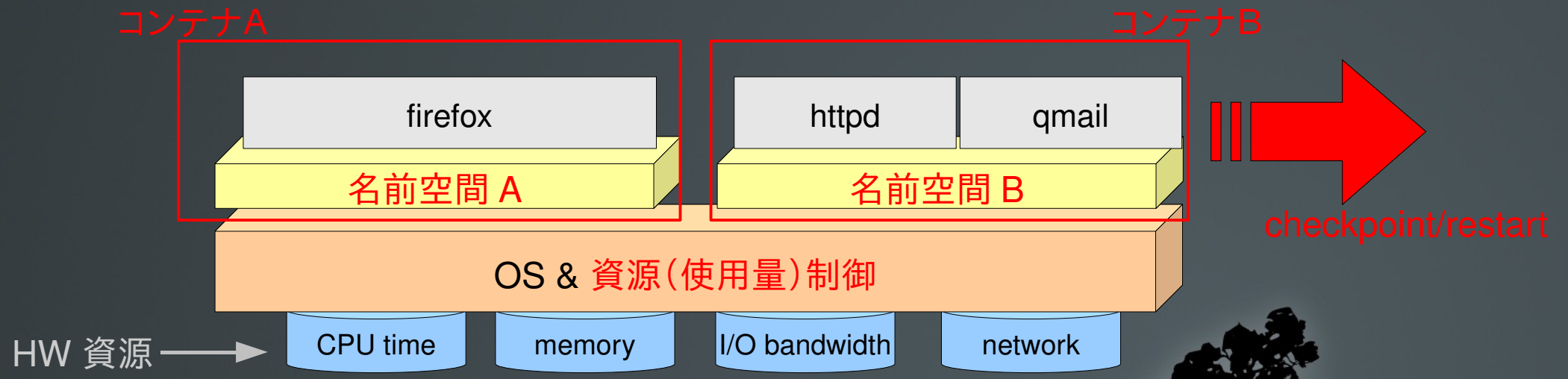
更に実装を増やす必要あり

通常は独立空間化されていないものの例

- IPC, IP/MAC アドレス , PID, UID, ディレクトリ構造 , /dev, /proc

Containers/Control Groups とは

- 「アプリの振舞いをコンテナ（入れ物）内だけで完結させる」仕組み。転じて、独立性の高い実行環境を OS が提供する機能
  - 主な機能：名前空間、資源制御、checkpoint/restart
- IBM, OpenVZ, コロンビア大などがそれぞれ Linux コンテナの独自実装を持っているので、ML を通じてメインライン化を推進中



コンテナでできること、できないこと

- できること
  - コンテナ毎に資源のアクセス制限や帯域制限ができる
  - コンテナ（プロセス）を移動させることができる
  - 仮想化オーバヘッドが小さい実行環境を提供できる
- できないこと
  - 異なる OS を混在させることができない
  - カーネル機能の開発には向いていない

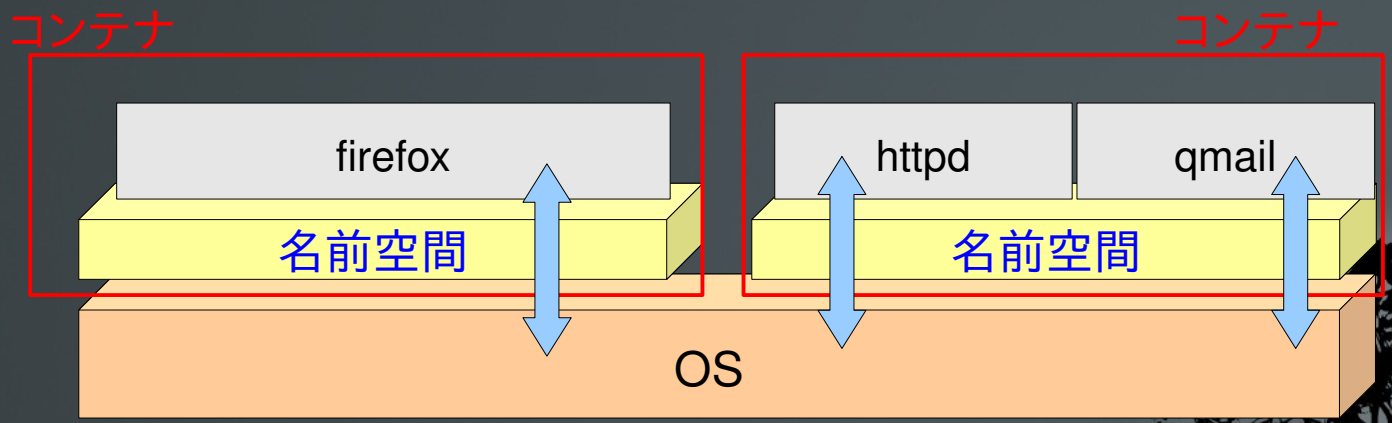
コンテナが活用できそうな例

サーバ		高信頼サーバ	ロードバランス リカバリ
デスクトップ		マルチメディア アプリケーション	
組込み	ネイティブアプリの 安全実行	ストリーミングの 帯域確保	ユビキタス モビリティ

名前空間 (namespace)

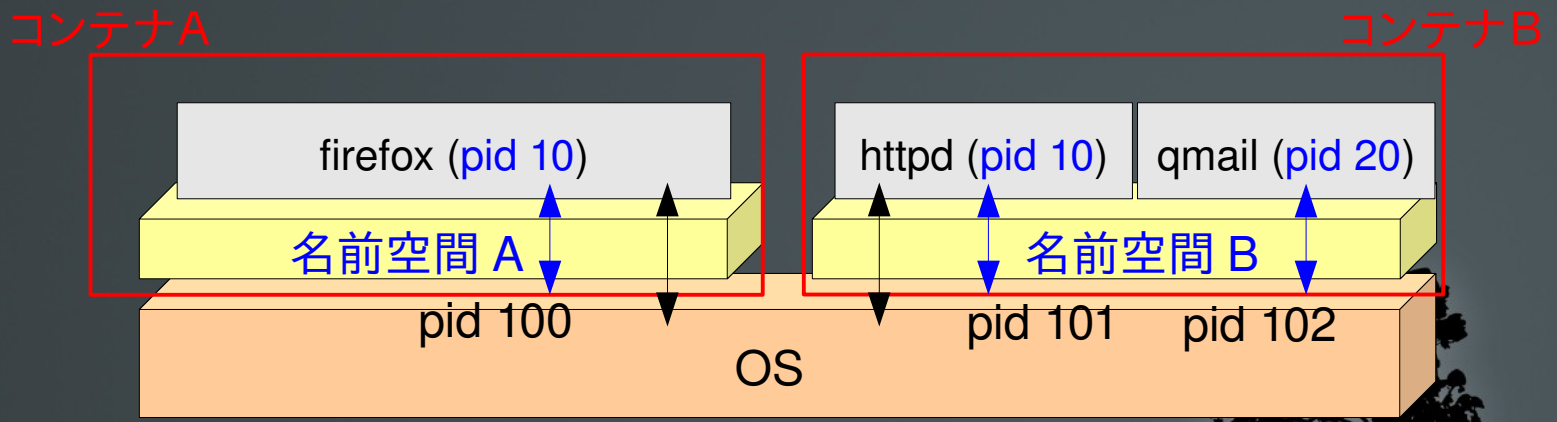
- アプリケーションとカーネルの間で名前 (ID) を相互変換する層
- カーネル内部では一意に決まる ID をつけていることもある
- 仮想化が必要な“名前”の一例

  各種 ID (プロセス、IPC 、ユーザ)、ファイル名(ツリー構造、/proc 、/dev 、/dev/pts)、デバイス、時刻、hostname 、IP/MAC アドレス、カーネルバージョン



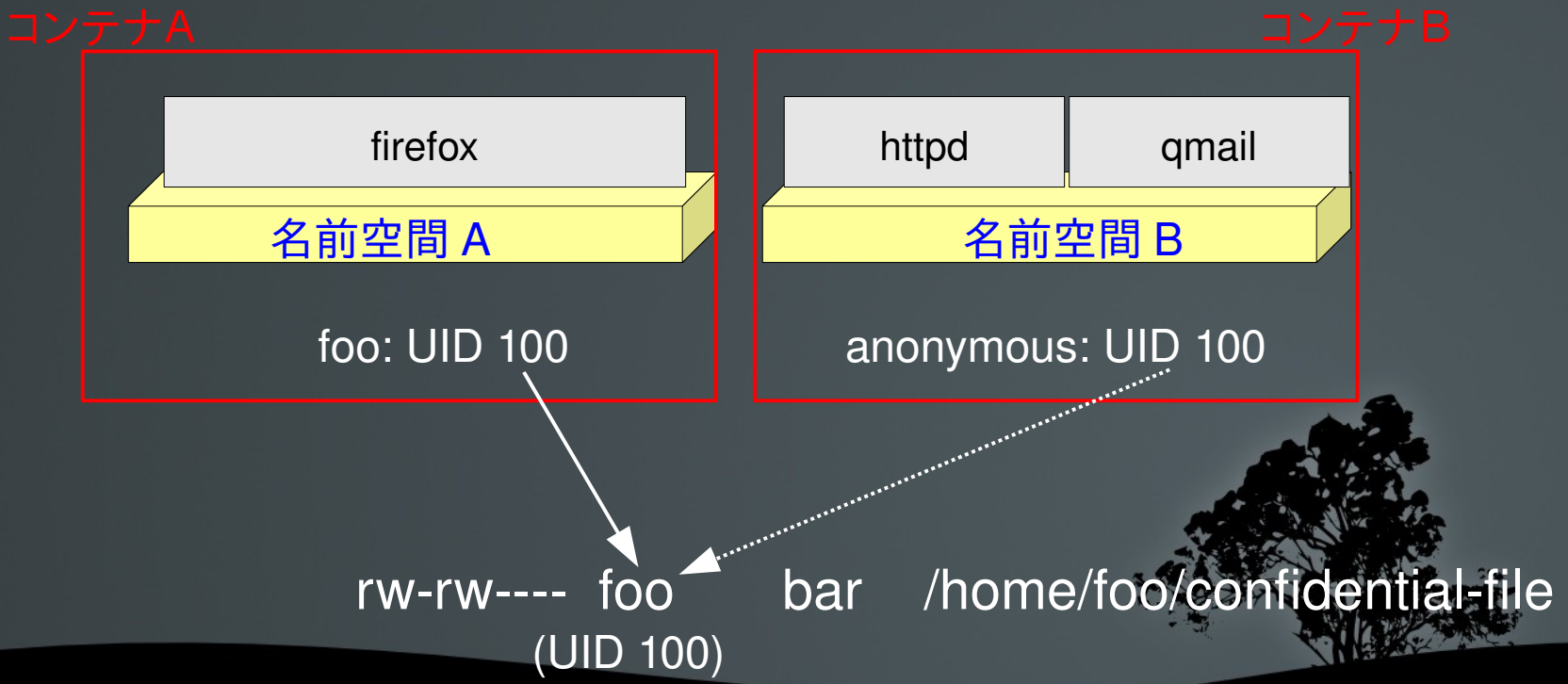
プロセス ID 名前空間

- アプリが使っている PID とカーネル内の PID を変換している
- 例えば checkpoint/restart を使ってプロセス（コンテナ）が移動してきたときに、同じ PID 番号が衝突しても問題にならない
- 当然、コンテナ外から PID を指定した処理は難しくなる
  - シグナル送信
  - プロセス間通信



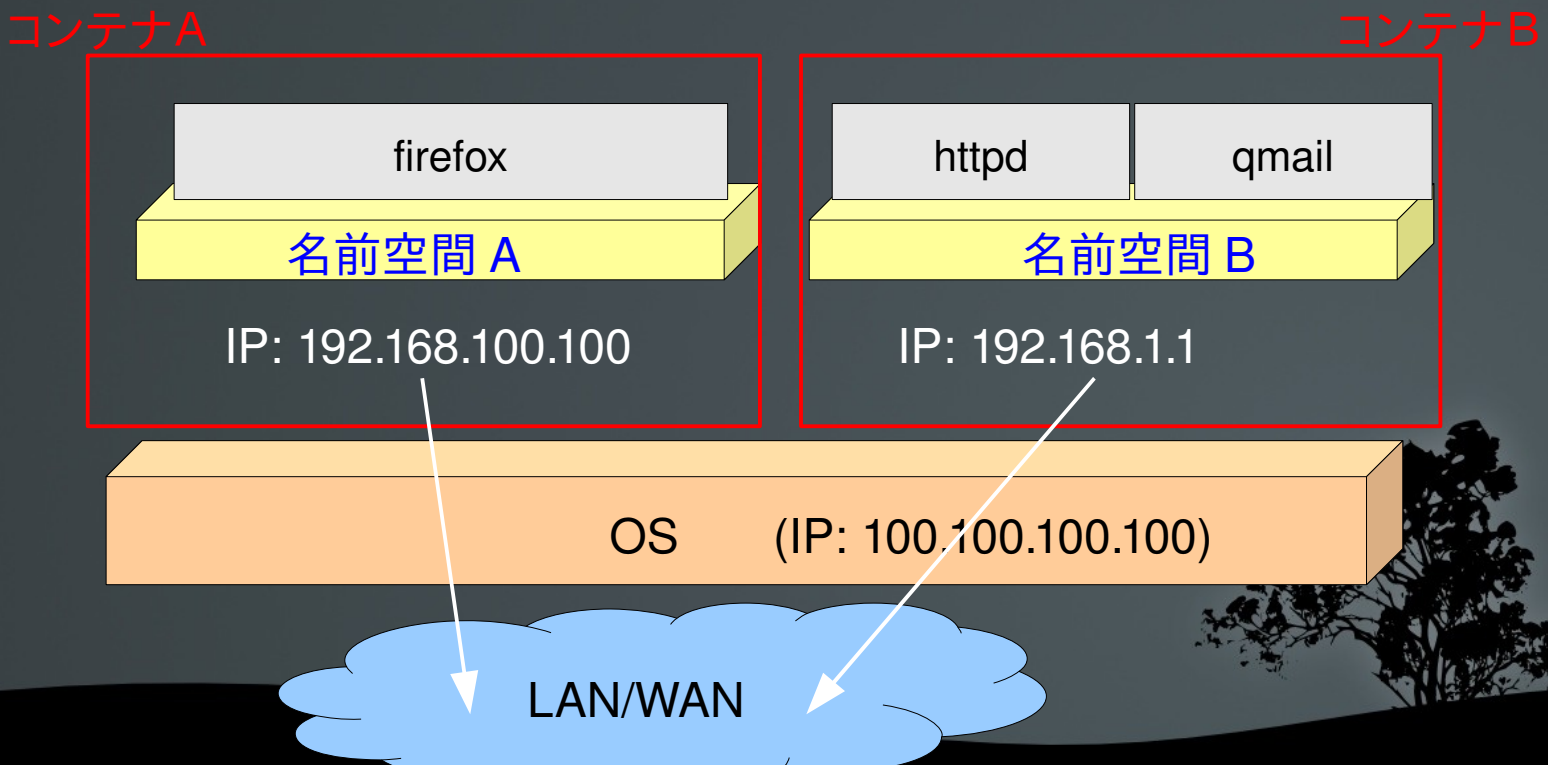
ユーザ ID 名前空間とファイル

- ユーザ ID はコンテナ依存なため、コンテナ毎にディレクトリ構造を変えなかった場合、ファイルの所有権などでセキュリティホールになりうる



ネットワーク名前空間

- 異なる IP アドレスをコンテナ毎に割り当てる
  - Web サーバと AP サーバは異なる IP アドレスだが、コンテナを独立させて同一マシン上で実行するなど



名前空間のメインライン状況

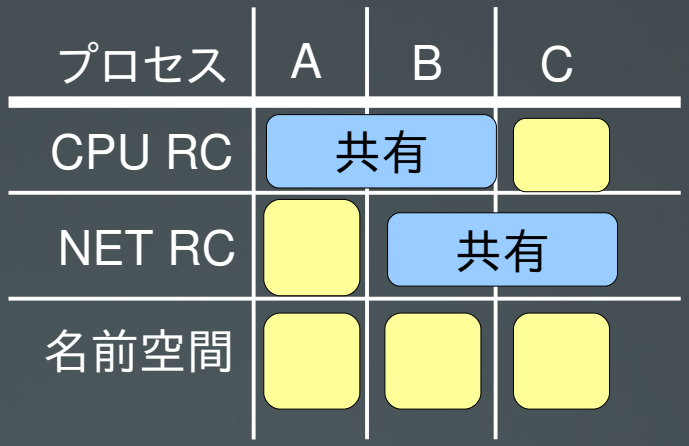
- 2.6.19 – 2006/11/29
  - UTSNAME/hostname
  - IPC
- 2.6.23 – 2007/10/09
  - USER
- 2.6.24 – 2008/01/24
  - PID
- 2.6.26 – 2008/07/13
  - NETWORK
  - /proc (/sysfs ?)
  - ro bind mount ?

[出典： http://lxc.sourceforge.net/]

資源制御 (resource control/management)

- 共有資源の使用量や帯域を適切に配分する機能
- 現在の API では、各資源管理を独立して設定することも可能

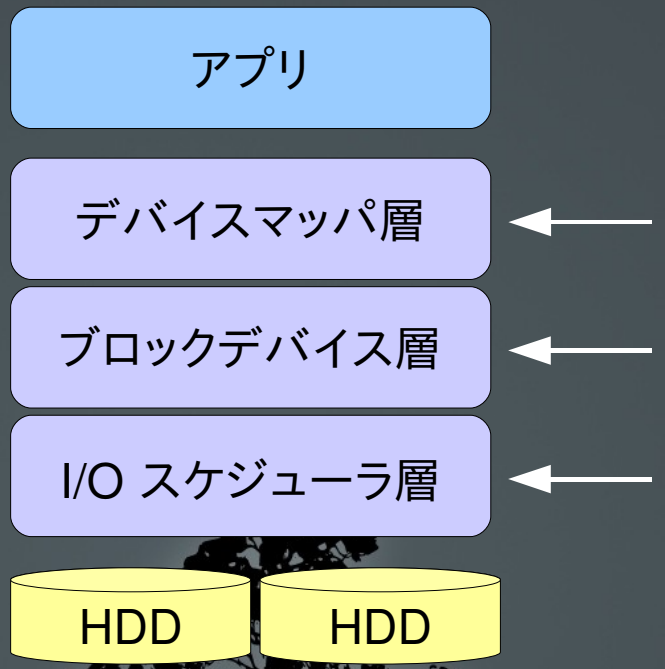
```
mount -t cgroup -o cpuset /cont/containerA
mount -t cgroup -o diskio /cont/containerB
```



- メインライン済(サンプル実装含む): CPU 時間、メモリ使用量
- 議論中:ディスク帯域、ネットワーク帯域、スワップ使用量

ディスク帯域制御

- 今年のホットトピック。現在、複数の実装が提案され群雄割拠状態
  - 物理デバイスとユーザアプリとの間に多くの層が入り込んでいるために直感的にならない
    - 配分の設定値は相対値？絶対値？
    - どのレイヤで実装するべきか
    - どれを使うかはシステム構成依存？



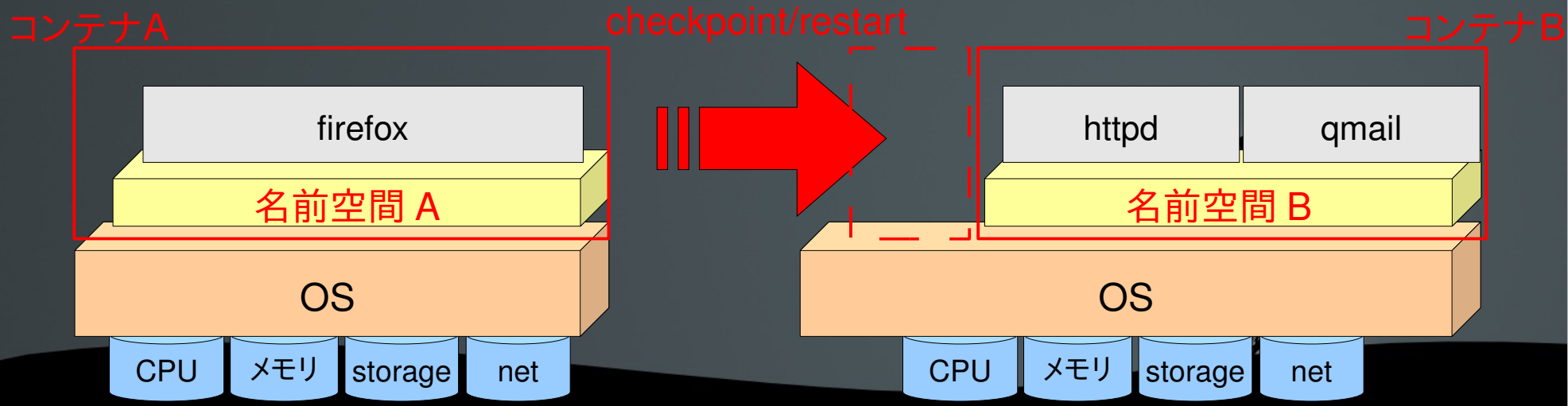
ディスク帯域制御 (cont.)

- その他の論点
  - オーバコミットは許可？使い切れなかった場合は？
  - 欲しいのは保証？制限？(HDD の場合、帯域保証は難しい)
  - CFQ のバグ？
  - ページキャッシュに対応させないとダイレクト I/O しか対応できず
- 前回のミニサミットでは、ほとんどの担当者が不在で議論できず
- OLS での発表のあとに、その場に居た人たちで議論
  - 「スケジューラ層の実現部分で共通部分を切り出す」という妥協案
- OLS 後に RFC が出され、とりあえずの方向性が決まる
  - まずはページキャッシュをコンテナと紐づけできるようにする
  - 次にスケジューラ層の実装がページキャッシュに対応する
  - dm 層の対応？

Checkpoint/restart

プロセスも含めてコンテナの実行状態を保存し、後になってそれを再開する機能。再開場所は必ずしも別のマシンである必要はない

- 主な用途
  - “Live Migration” → 信頼性向上（ディペンダビリティ）、負荷分散
  - 人・物の近くに移動 →モビリティ、ユビキタスコンピューティング
  - 情報の近くで計算する →高性能化、通信量削減、省電力
  - 障害発生時の実行再計算 →リカバリ



Checkpoint/restart (cont.)

- restart 時の ID 衝突回避の為、名前空間の実装が望ましい
- 実行環境であるコンテナを丸ごと保存すべし
  - 情報がすべて揃っているはず

some Checkpoint/restart stuff

まだ、コンテナ ML でのサンプル実装や議論が始まったばかり…

- Zap-based patch not using NS by Oren Laadan (Columbia University)
  [RFC v4][PATCH 0/9] kernel based checkpoint/restart (08/09/09)
- OpenVZ-based patch using NS by OpenVZ team (Andrey Mirkin)
  [PATCH 0/9] OpenVZ kernel based checkpointing/restart (08/09/03)
- Freezer: patch by Matt Helsley and Cedric
  [PATCH 0/5] Container Freezer v6: Reuse Suspend Freeze (08/08/11)
- Memory checkpoint patch using swapfile per container by Dave Hansen
  [RFC][PATCH 2/2] memory checkpoint with swapfiles (07/06/13)

important URIs: http://lxc.cvs.sourceforge.net/lxc/

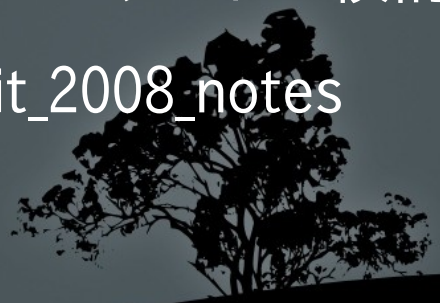
git://git.kernel.org/pub/scm/linux/kernel/git/daveh/linux-2.6-lxc.git

ミニサミット @ ケンブリッジ (2007/09)

- 主題:カーネルサミットで発表する今後の方針を議論
- 参加者は 7 人 (IBM, OpenVZ, Google, Columbia Univ., NTT, NEC)
- LinuxConf Europe の一室で開催
- API に関する議論
  - clone(2) の引数 flag を 64bit または可変長に拡張する案
  - C/R 用の API (セキュリティ面も考慮する必要有り)
- デバイスの仮想化
  - USB に追加したときにそれを見せるかをコンテナ毎に制御
  - 時刻の仮想化
- スワップ領域をコンテナ毎に割り当てるか否か。Dave のパッチがこれを前提として作られていたが…。
- http://lxc.sourceforge.net/doc/mini-summit2007/

ミニサミット@オタワ (2008/07)

- 主題:現在提案されているパッチのまとめと今後の方向性の議論
- 参加者は 30 人強で、特に IBM, OpenVZ, HP が目立った
- Ottawa Linux Symposium の前日に開催
- 現在提案中の名前空間や資源管理機能のサマリ。何も言わないとスルーされた
  - メモリ資源制御は？ユーザレベルへの通知が必要？ softlimit?
  - ユーザ ID とファイルシステムの仮想化の関係が重要。セキュリティ上の問題も発生する
- 今後、C/R をメインライン化するためのスモールスタートの検討
- http://wiki.openvz.org/Containers/Mini-summit_2008_notes



Thank you !!!