Solaris 10 Developer Meeting for x86 Solution

Solaris セッション 2
Solaris 10 インサイド

大曽根 明

東京ソフトウエア本部

Sun microsystems

本日の内容

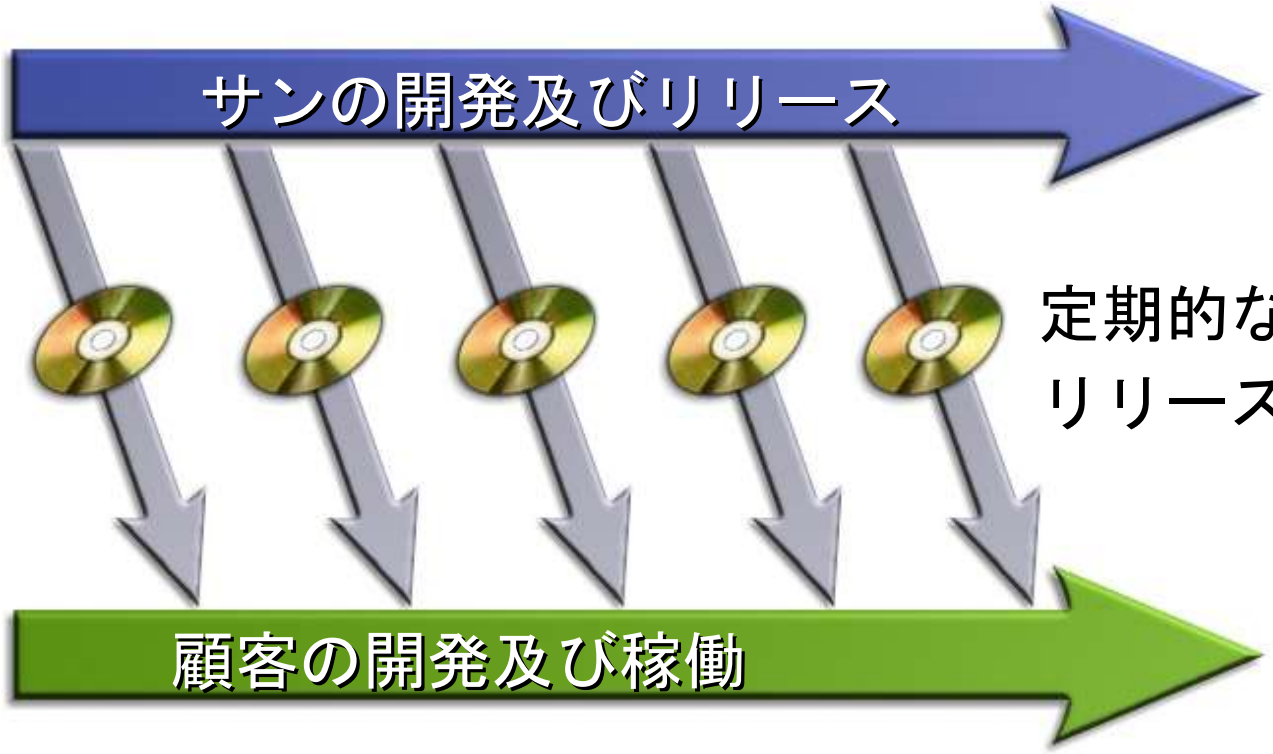
- Software Express for Solaris™
- Dtrace
- Container (zones)
- SMF

solaris™

Software Express

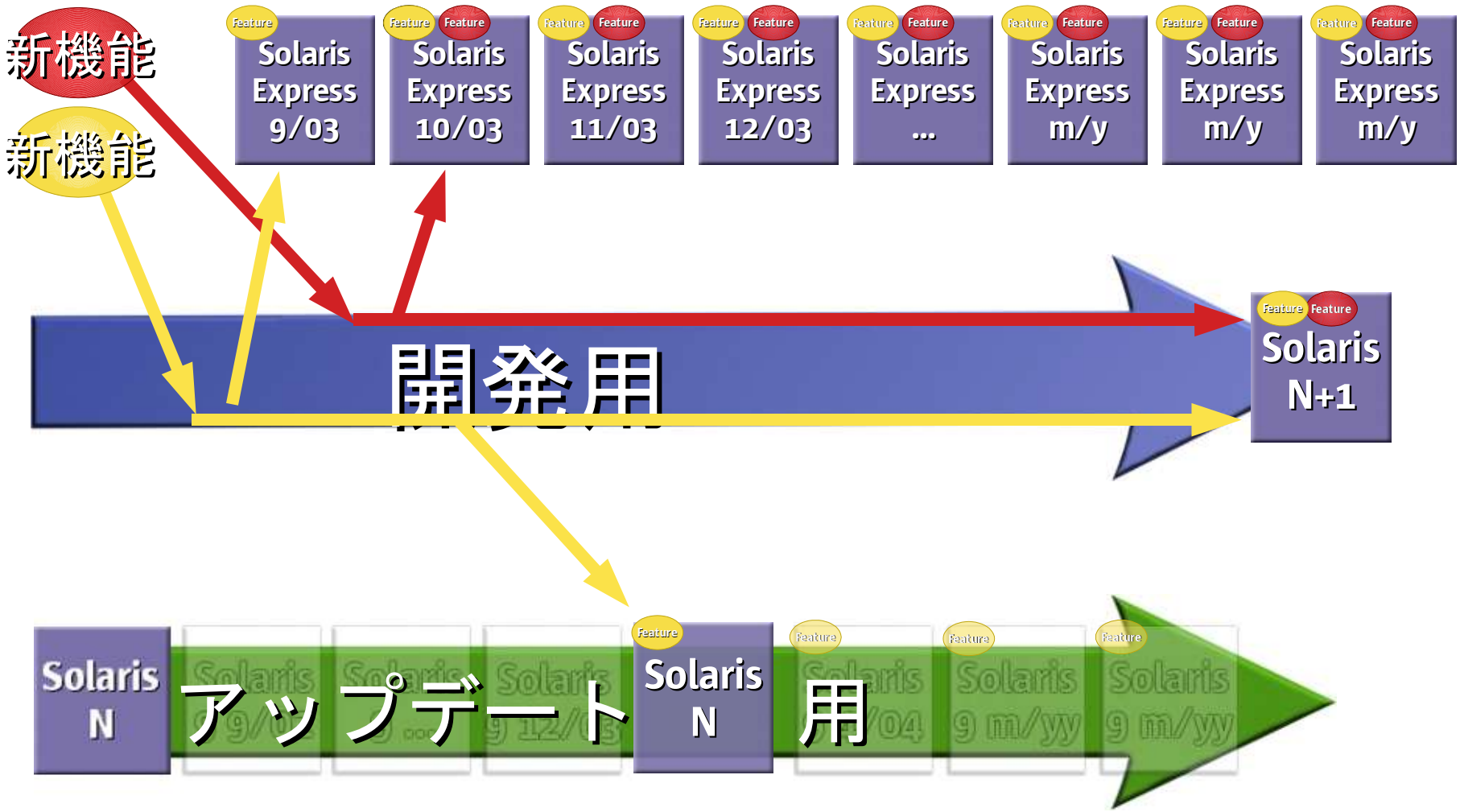
サンの
開発者

サンの
顧客

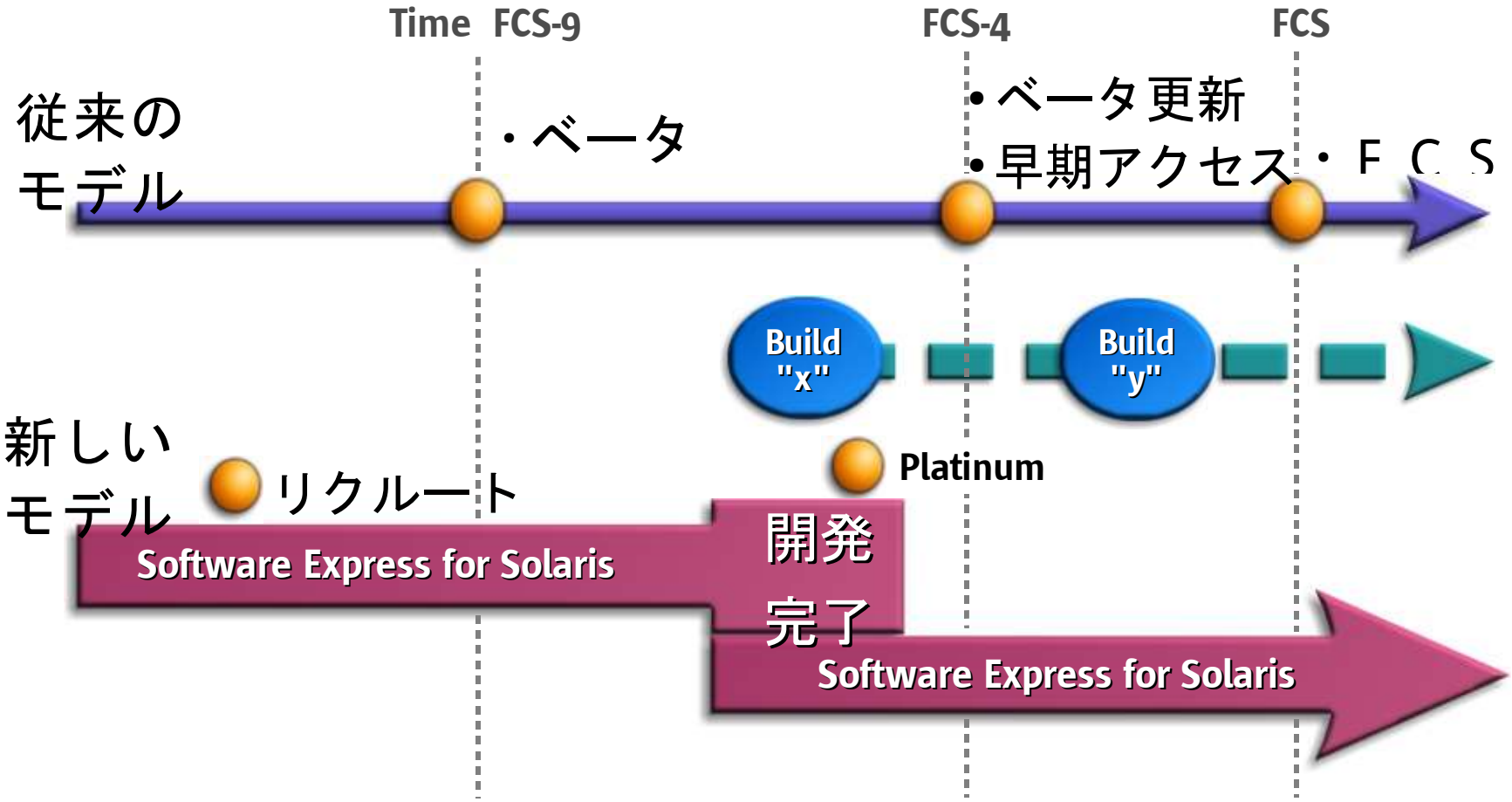


- 開発中のコード
  - 完動し、試験もされている
    - 定期的（毎月）なリリース

Software Express for Solaris™



Solaris リリースモデル



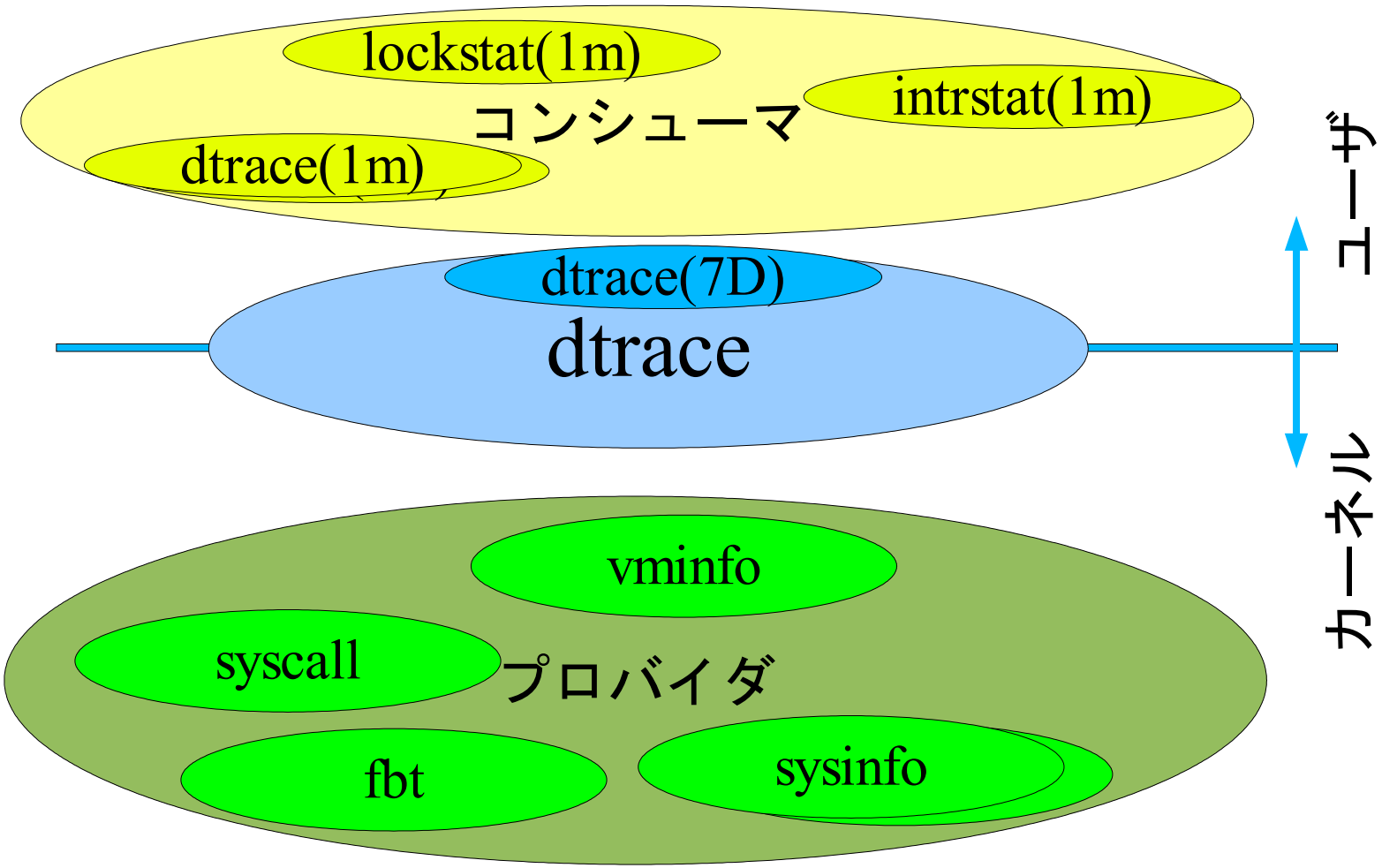
Dtrace - 概要

- 動的にシステム（カーネル及びユーザ）の動作の調査
- 本番稼働中のシステムであっても最小限の影響で使用可能
- 単発的に起こる問題の解析にも適している
- 統計データも簡単に集計できるのでパフォーマンスの解析にも適している
- 多くのデータの収集もとの提供とプログラマバブルなツールの提供できわめて柔軟性が高い

Dtrace - 構造

- プロバイダ：プローブを提供、関数の入り口、出口、 **VM** 関連情報、システムコール、その他
- プローブ：3 万以上のプローブが実装されているがオフの時はほとんどオーバーヘッドなし
  - 「プロバイダ：モジュール：関数：名前」で指定
- コンシューマ： データを利用するプログラム、代表例は **dtrace(1m)** コマンドであるが自由に作成可能（ 例： **lockstat(1m)** ）。同時にいくつものコンシューマを使うことができる

Dtrace - 構造（続き）


lockstat(1m)
intrstat(1m)
コンシューマ
dtrace(1m)
dtrace(7D)
dtrace
ユーザ
カーネル
vminfo
syscall
プロバイダ
fbt
sysinfo

Dtrace - プロバイダ

- 主なプロバイダは：
  - Syscall: システムコールの入り口、出口
  - Fbt: 各カーネル内関数の入り口、出口
  - Lockstat: カーネル同期関連
  - Profile: 統計データを取るための定期的な呼び出し
  - Pid: 動的にユーザコードのトレース

Dtrace - コマンド

- Dtrace のコンシューマの一つ
- D 言語で記述
  - Awk/perl に似ている
  - Kernel 内のすべての変数にアクセス可能
  - 特別なインタプリタであり安全な操作のみを（選択的に）実行可能
- Awk/Perl のようにコマンドスクリプトにできる

Dscript の例

```
#!/usr/sbin/dtrace -s
#pragma D option quiet
syscall::open*:entry
/copyinstr(arg0) == $$1/
{
        printf("%6d/%6d/%-15.15s\n",
                curthread->t_procp->p_cred->cr_uid,
                pid, execname)
}
```

Dtrace デモ -1

- 実際に動作するところをみましょう！

Dtrace - Dscript における統計

- パフォーマンスの問題を解析するときにはここの細かいデータをみるとのではなく、統計データとすると扱いやすい
- Dscript には様々な関数が用意されている
  - Count(), sum(), max(), min(), avg()
- 複数の統計データを簡単に扱えるように名前を個々にもうけないでも収集できるようになっている

Dscript - 例 2

```
syscall::write:entry
{
        @counts[execname] = count();
}
```

Dscript - デモ 3

```
syscall::write:entry
{
    self->ts = timestamp;
}

syscall::write:return
/selt->ts != 0/
{
    @time[execname] = avg(timestamp - self->ts)
    self->ts = 0
}
```

Dscript - デモ 3

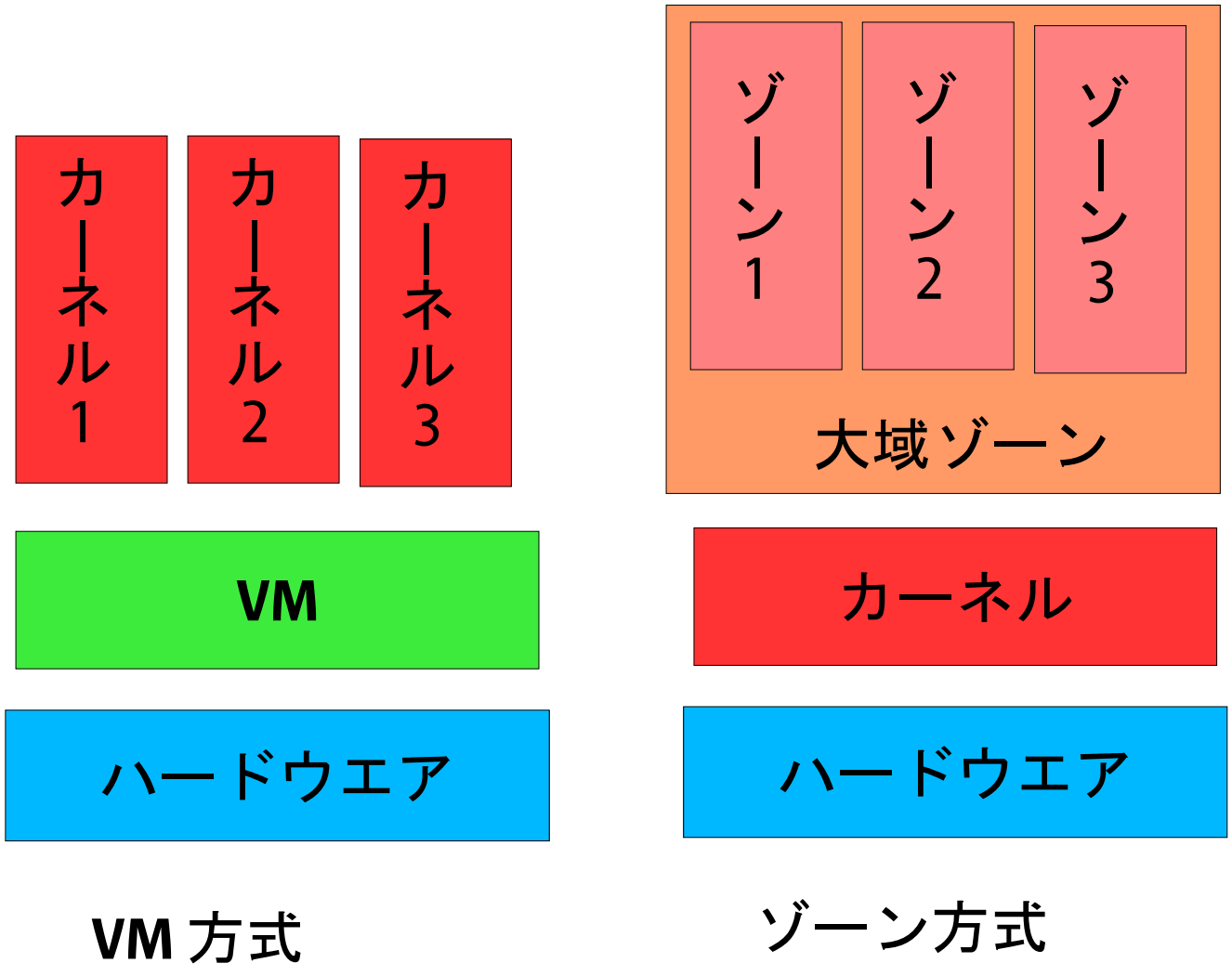
```
syscall::write:entry
{
    self->ts = timestamp;
}

syscall::write:return
/selt->ts != 0/
{
    @time[execname]
        = quantize(timestamp - self->ts)
    self->ts = 0
}
```

コンテナ (ZONES)

- 別々の Solaris かのように仮想 OS の環境を提供
- システムのセキュリティを向上可能
- 各アプリケーションを隔離可能
- 任意の細かさで資源管理や隔離が可能
- アプリケーション環境は既存のものと互換がある
- プラットホームの詳細を隠すことができる
- VM 方式のオーバーヘッドなしに実現

コンテナの構造


カーネル1
カーネル2
カーネル3
VM
ハードウエア
VM 方式
ゾーン1
ゾーン2
ゾーン3
大域ゾーン
カーネル
ハードウエア
ゾーン方式

コンテナ - セキュリティ

- 各ゾーンは他のゾーンから隔離されている
  - 他のゾーン（特に大域ゾーン）に対して影響できない
- ゾーンは制限された権限の元に動作
  - **Memcntl(2), mknod(2),stime(2)** などは使えない
  - **kernel** のモジュールの導入やネットワークを変更できない
- 名前空間は隔離されている

コンテナ - セキュリティ

- 大域ゾーンはすべてのゾーンのプロセスやファイルをアクセスできる
- 仮想化されたファイルシステムの空間が提供される
  - ディスクレス・クライアントに似ている
  - 個別にアクセスを追加できる
- ネットワークのトラフィックや **System V IPC** などはそのゾーンに向けられたものだけが見える
  - グローバルゾーンはすべて見える

コンテナ - 制限事項

- **/usr** など大域ゾーンと共有するファイルシステムにインストールできない
  - デバイスドライバを含む
- カーネルのデータ構造に依存できない
  - **Libkvm** 経由であれば可能
- 特定のデバイスに依存する
  - カスタマイズで可能であるが推奨しない
- 特定の **/etc/system** 設定に依存できない ***1**
- 特定のファイルシステムに依存できない ***1**

*1: 大域ゾーンのレベルでしかできない

コンテナ - デモ

- 実際の例をみてみましょう

SMF

- Service Management Facility
- /etc/init.d/*, /etc/rc* や /etc/inetd.conf などを入れ替える
- 各種サービスの起動、（必要に応じた）再起動、停止、利用中止などを司る
- 各種サービスの依存関係（入れ子構造）に基づいた管理
- Xml によるサービスの記述 (manifest)
- fm(fault manager) との連携

SMF - コマンド

- svcs(1)
  - 状態の表示
- svcadm(1m)
  - 状態の変更など管理
- svccfg(1m)
  - 構成の変更、管理

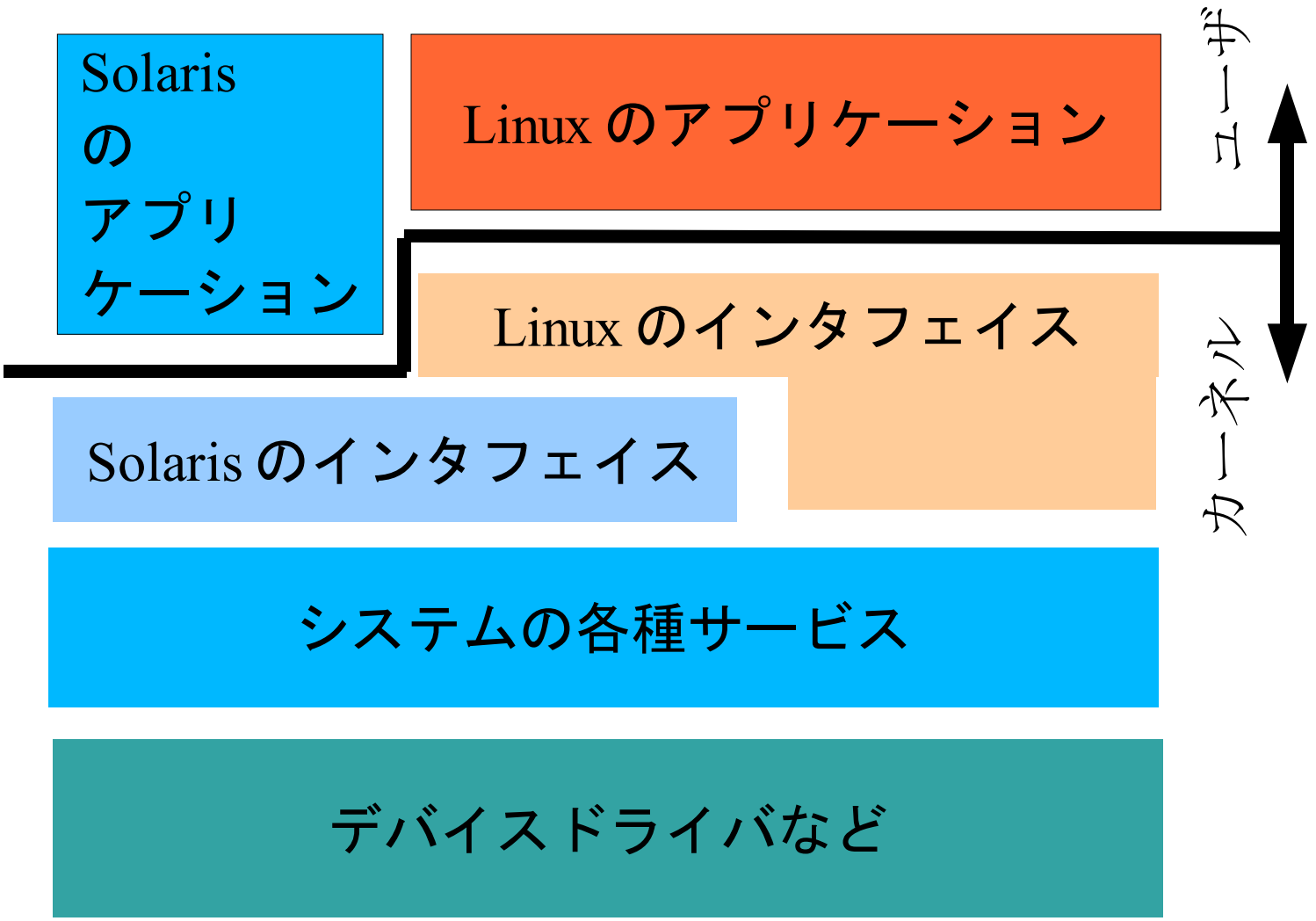
SMF - デモ

- 実際に使ってみましょう

Janus – Linux の顔：開発中

- ユーザモードのアプリケーションからみたときに Linux の様に振る舞う
- Lxrun と違いカーネル内で走るので高速
- 0x80 のインターラプトでも判断
- カーネル以外は別途インストールが必要
  - たとえば RedHat EL3 から必要なものをインストールする
- アプリケーションによってはより高速に Solaris で実行される例がある

Janus - 構造


Solaris
の
アプリ
ケーション
Linux のアプリケーション
ユーザ
Linux のインタフェイス
カーネル
Solaris のインタフェイス
システムの各種サービス
デバイスドライバなど

質疑応答

ありがとうございました

akira.ohsone@sun.com

大曽根 明

東京ソフトウエア本部

Sun microsystems