



Solaris Innovation Forum (5-Jun-08)

C: 1440 - 1520

# アプリケーションの運用性能を 最大化させる Solaris10 の底力

サン・マイクロシステムズ  
ビジネス・アプリケーション技術部  
国谷俊夫



# セッション概要

- Solaris Container, ZFS, DTrace, セキュリティ機能は **Solaris 10** の存在価値を特徴づけ、OS 製品としての競争力をリードしてきました。
- しかし、そのメリットを知りながらも実運用で応用できなければ、ビジネス価値を生み出すことはできません。
- 本セッションでは、具体的なビジネス・アプリケーションやオープンソース・ソフトウェアとの組み合わせを例に、**Solaris 10** の真の価値を現場で発揮して頂くための技術解説を行います。



# Agenda



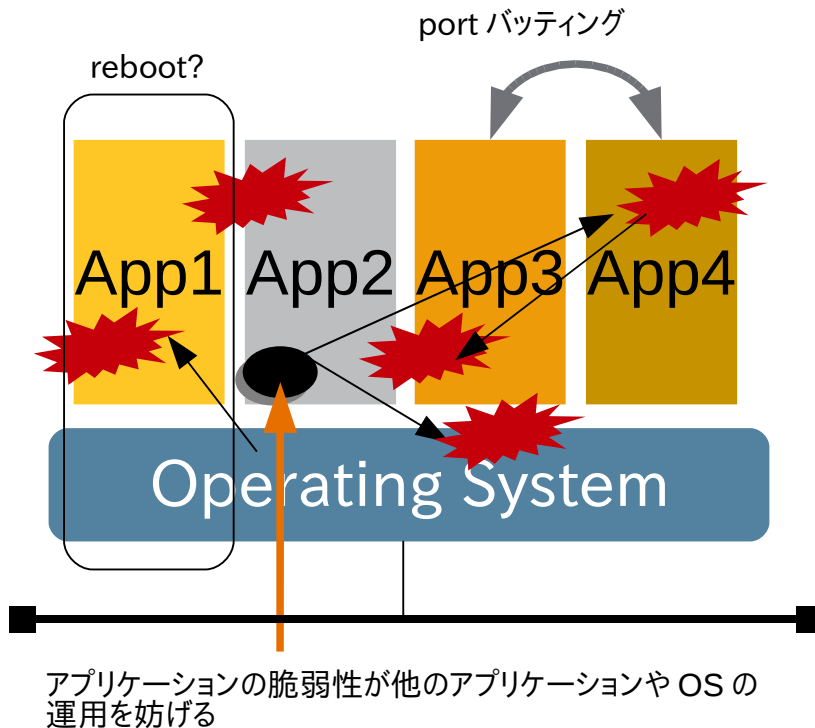
- Solaris 10 諸機能再レビュー
  - > Solaris Zone, Solaris Container
  - > ZFS
  - > DTrace
- ISV, OSS アプリケーションデモ
- まとめ
- Q&A

# Solaris 10 諸機能 再レビュー

# Solaris 10 諸機能再レビュー

Solaris Zone / Solaris Container

# サーバ統合とアプリケーションのギャップ

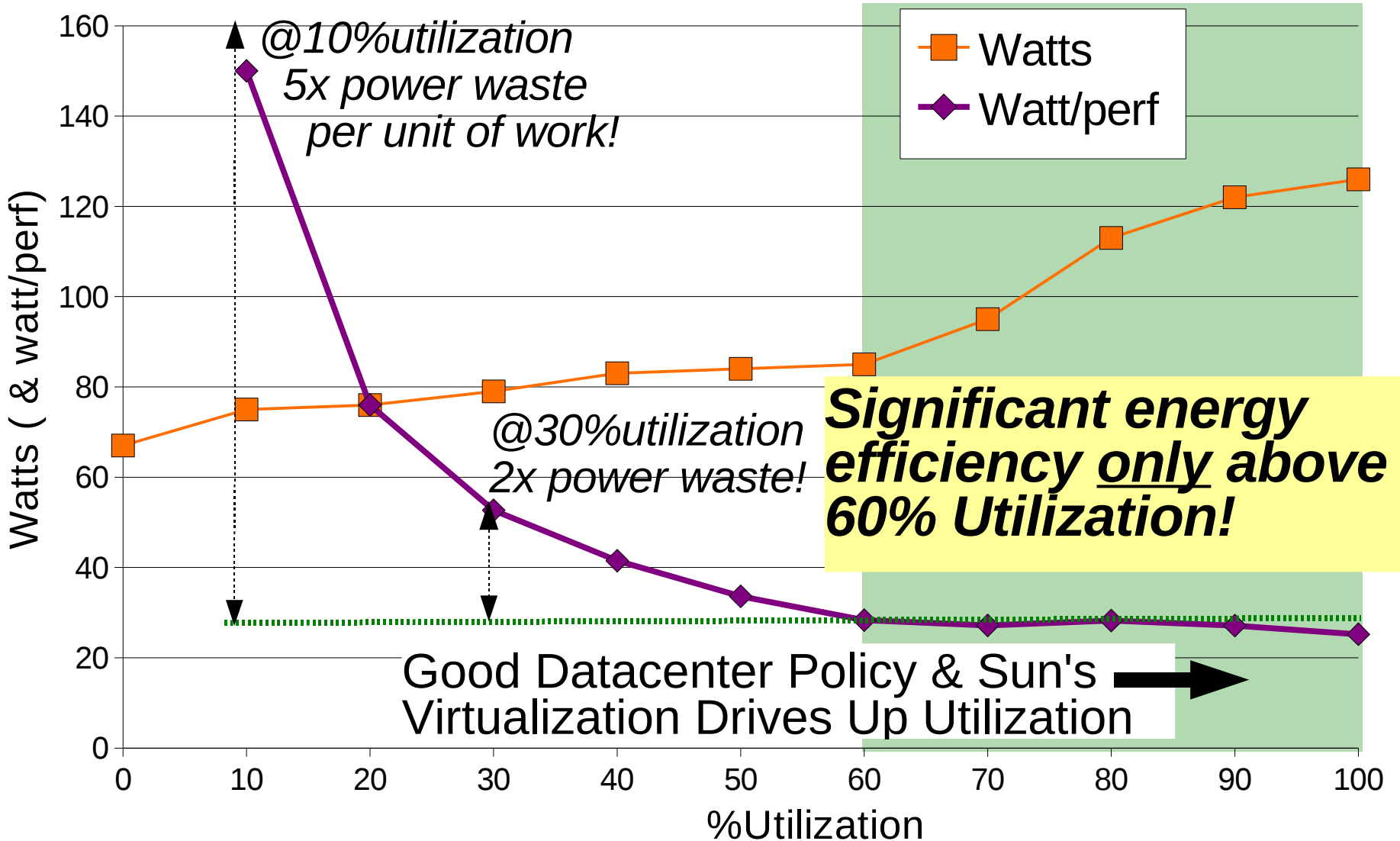


- 個別にサーバが増えてきた要因
  - > OS がアプリケーションの複数回インストールを認めない
  - > OS リソースの競合が発生する
  - > 一ヶ所の脆弱性が全体のセキュリティを低下させる
  - > 導入時期が異なる
  - > 再起動などの operational 副作用
  - > 業務アプリケーションが高度になり、個々のリスクと他への影響度が把握しにくい(見えないリスク)

一方で、ハードウェアの能力が高性能化するにつれてサーバの余剰能力は投資対効果へのネガティブインパクトとして目立ちはじめている

# 仮想化密度のゴール@ Sun

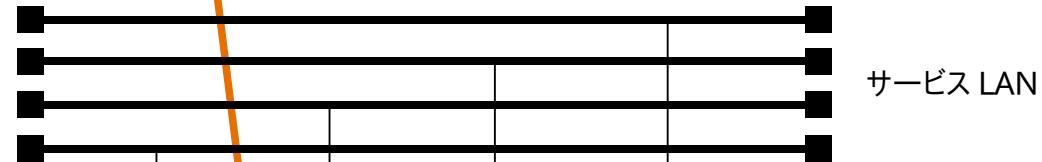
High %Utilization Saves More than Best Power Saving HW/SW



# Solaris Zone

アプリケーションの脆弱性が原因となる、  
OS や他の zone への影響がない

ネットワーク分離可能

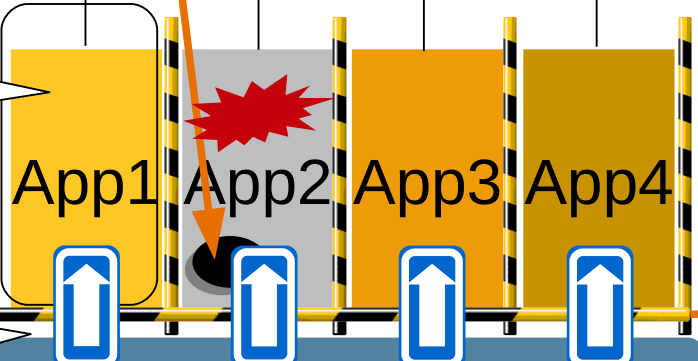


アプリケーションにとっては、  
自分専用 OS が与えられて  
いるかのように見える

reboot OK

TCP/IP バッティング無し

仮想化オーバーヘッド無し



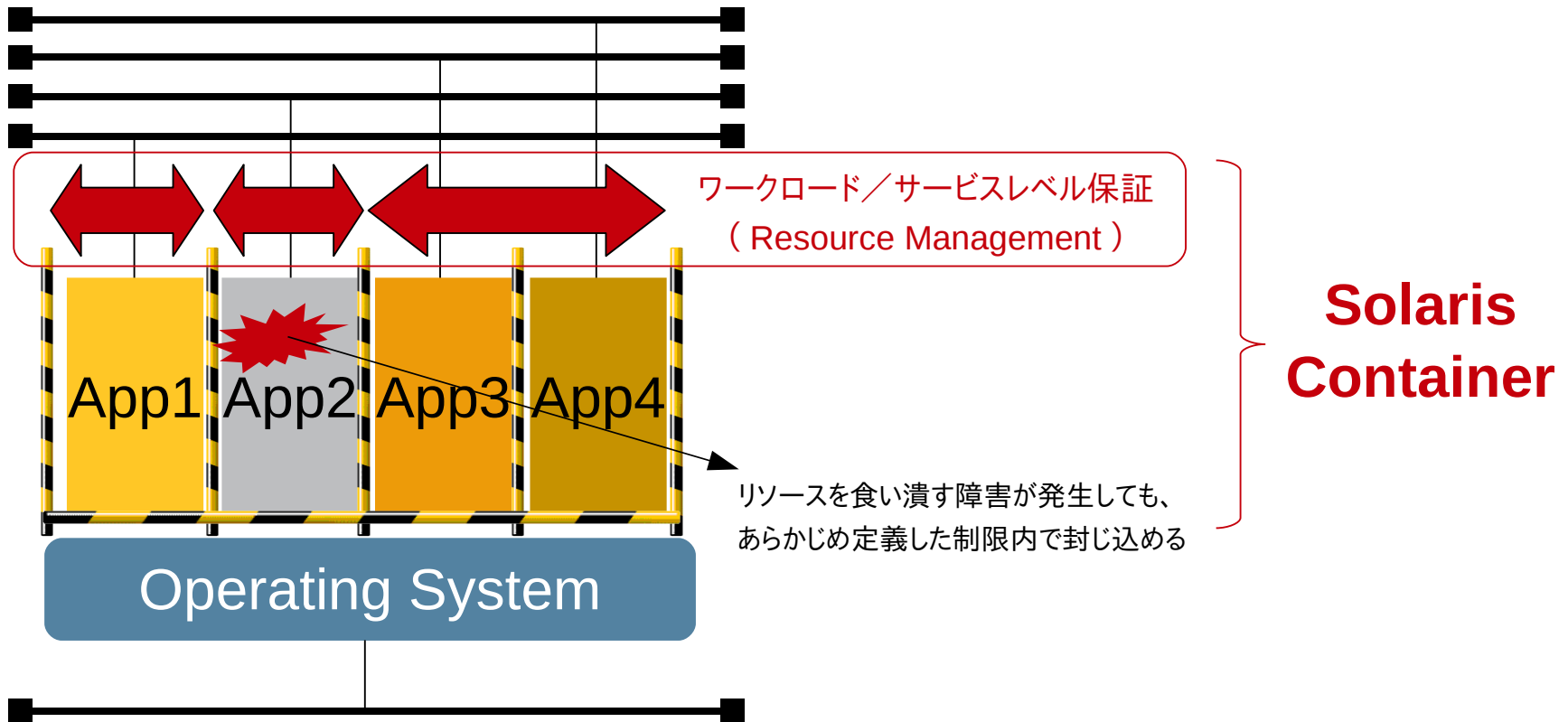
Global zone から local  
zone へアクセスできるが、  
local zone から global  
zone へアクセスできない

local zone

global zone



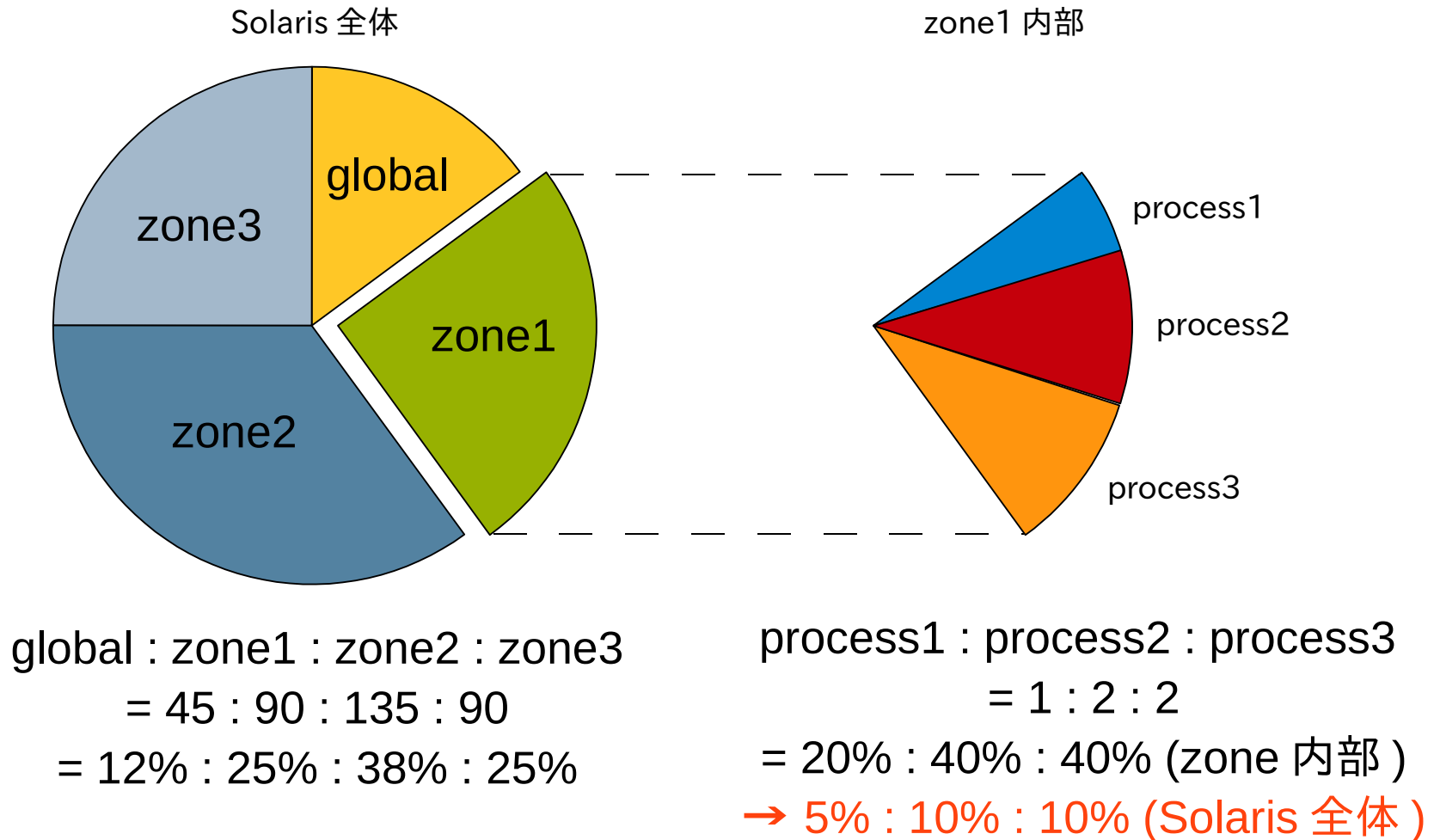
# Solaris Container



CPU, memory, swap, スケジューリングの調整により  
OS が自律的に、あらかじめ定義したサービスレベルを維持する

# Solaris Container リソース管理

## プロセススケジューリングの場合



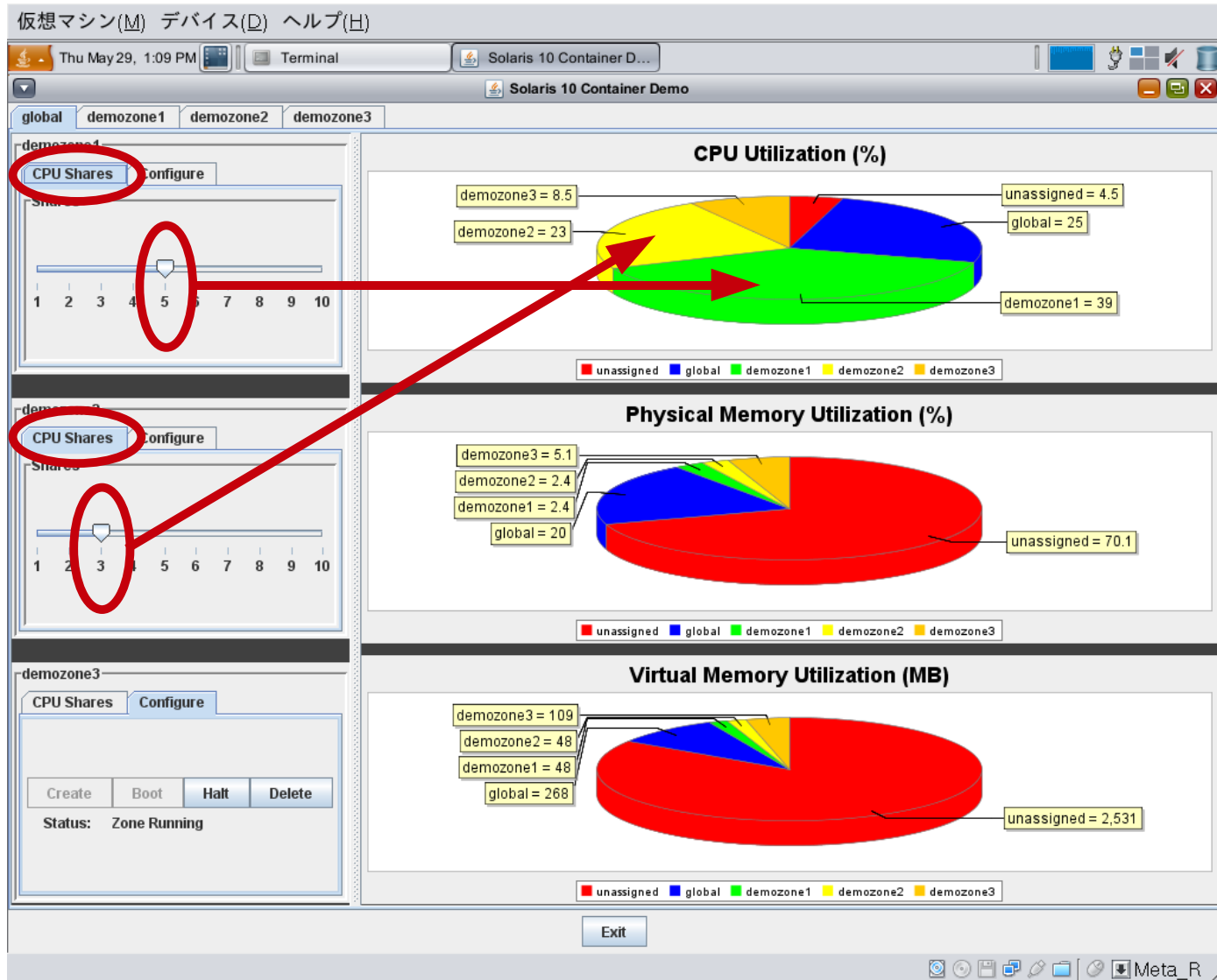
# Solaris Container デモンストレーション

## on xVM VirtualBox

zone1

zone2

zone3



# Solaris Container デモンストレーション

## on xVM VirtualBox

user0

user1

user2

user3

仮想マシン(M) デバイス(D) ヘルプ(H)

Thu May 29, 1:13 PM Terminal Solaris 10 Container D...

Solaris 10 Container Demo

global **demozone1** demozone2 demozone3

user0

Shares: 2 (relative priority)

Process Number: 0, 1, 2, 3, 4 (load)

user1

Shares: 4

Process Number: 0, 1, 2, 3, 4

user2

Shares: 7

Process Number: 0, 1, 2, 3, 4

user3

Shares: 1

Process Number: 0, 1, 2, 3, 4

**CPU Utilization (%)**

user.user3	3.3
user.user2	18
user.user1	12
user.user0	7.5
system	0.3
unassigned	58.9

**Physical Memory Utilization (%)**

user.user3	0.6
user.user2	1.1
user.user1	1.6
user.root	15
user.user0	3.6
system	12
unassigned	66.1

**Virtual Memory Utilization (MB)**

user.user3	20
user.user2	40
user.user1	60
user.root	164
user.user0	141
system	283
unassigned	2,296

Exit

# Solaris 10 諸機能再レビュー

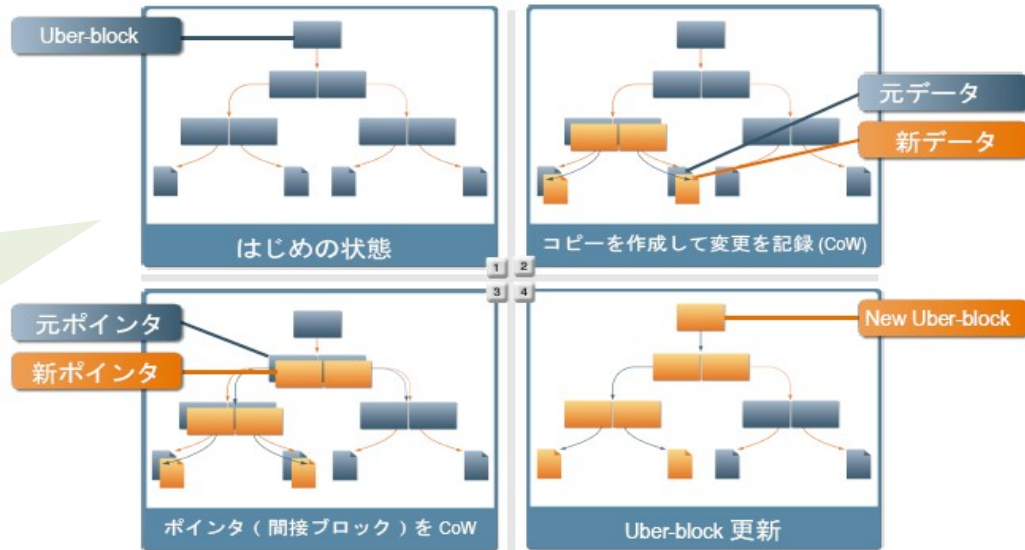
ZFS

# ZFS の特徴

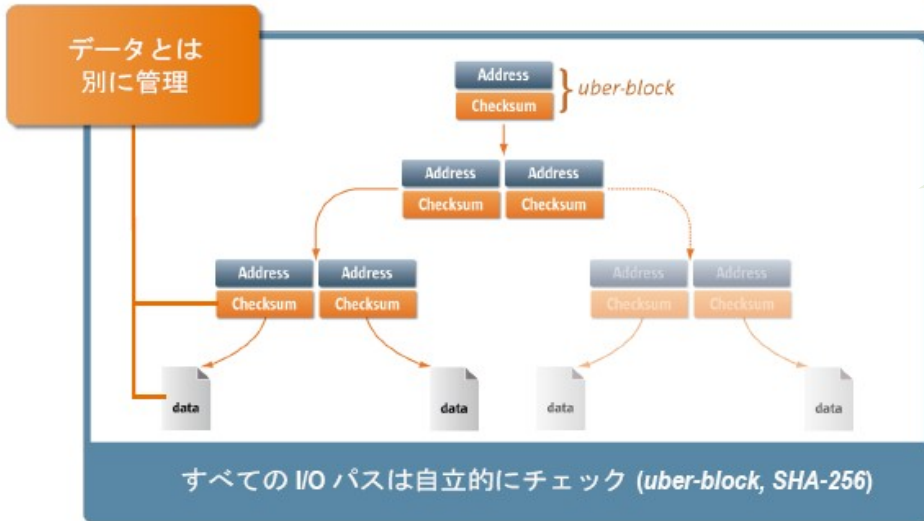
99.9999999999999999% の安全性

ファイル上書き中の障害でも古いファイルを保持可能。  
 (ハード RAID はブロックを直接上書きするので新旧ともに消失。  
 ロギングや fsck では復活しない。)

## Copy-on-Write and Transactional



## End-to-End Checksums



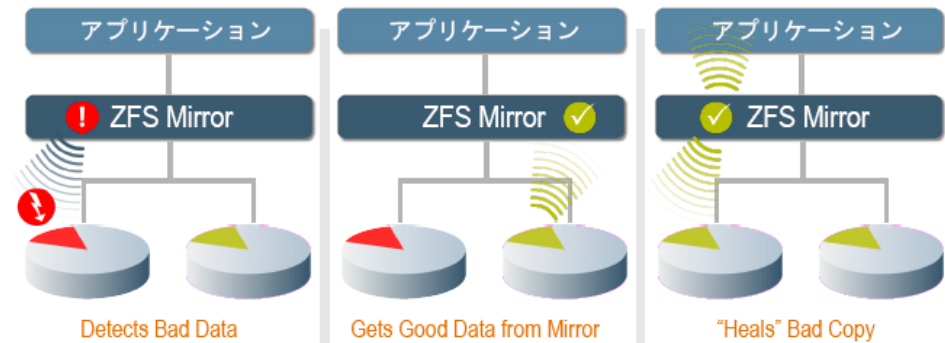
チェックサムが正しい限り、ファイルの不整合が発生していないことを保証する。  
 (UFS ではファイルの不整合を検知できない。)

# ZFS によるメリット

チェックサムにより、ミラーディスクのどちらに障害が発生しているか判断可能。ZFS が自律的に補正を行う。(これまでの技術ではどちらのミラーが壊れているか判断不能。)

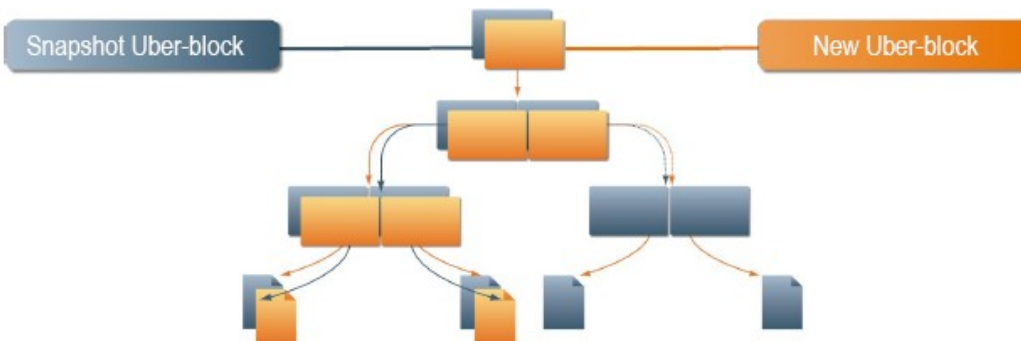
## Self-Healing Data

- checksums を用いて破損したミラーデータを検出し、自動的に補正がかかる



## ZFS Snapshots

- 書き込み不可な point-in-time コピーを作成
- 物理的なコピーではなく変更履歴のみをトラック
  - > 容量節約
- 瞬間的に終了
- Snapshot から clone させると書き込み可能なファイルシステムが作成できる

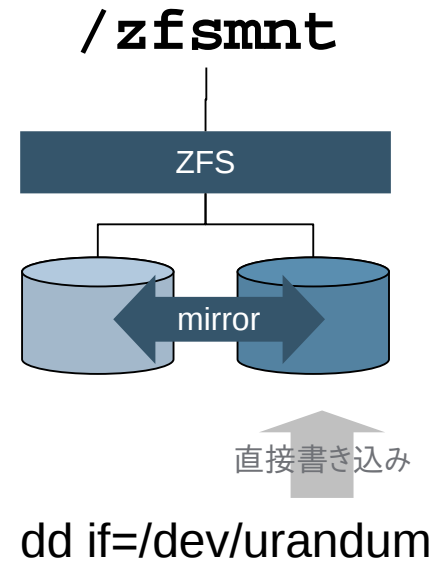
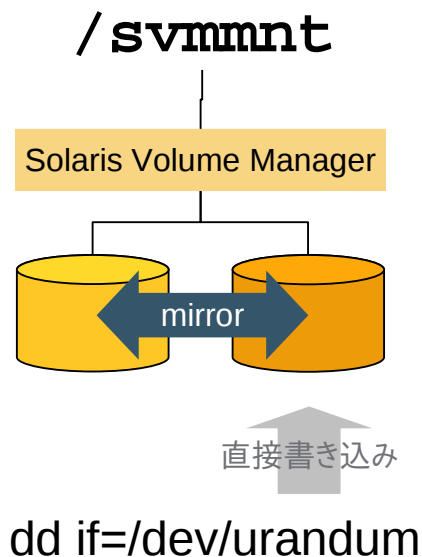


任意の時点でファイルシステム全体のイメージを追加容量なしで取得。

派生したファイルシステムの構築も瞬間的に可能。

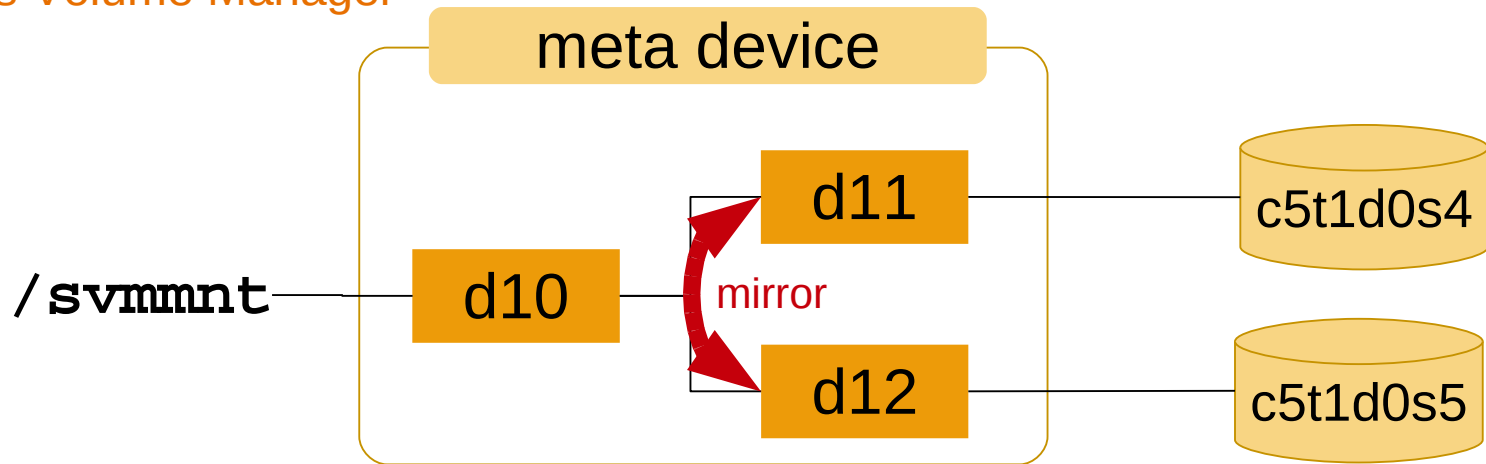
# ZFS デモンストレーション

- 2 種類のミラーされたファイルシステム
  - > Solaris Volume Manager
  - > ZFS
- dd コマンドを用いて擬似的に「 silent corruption 」を発生させる
- ファイルを読み出すことができるか？

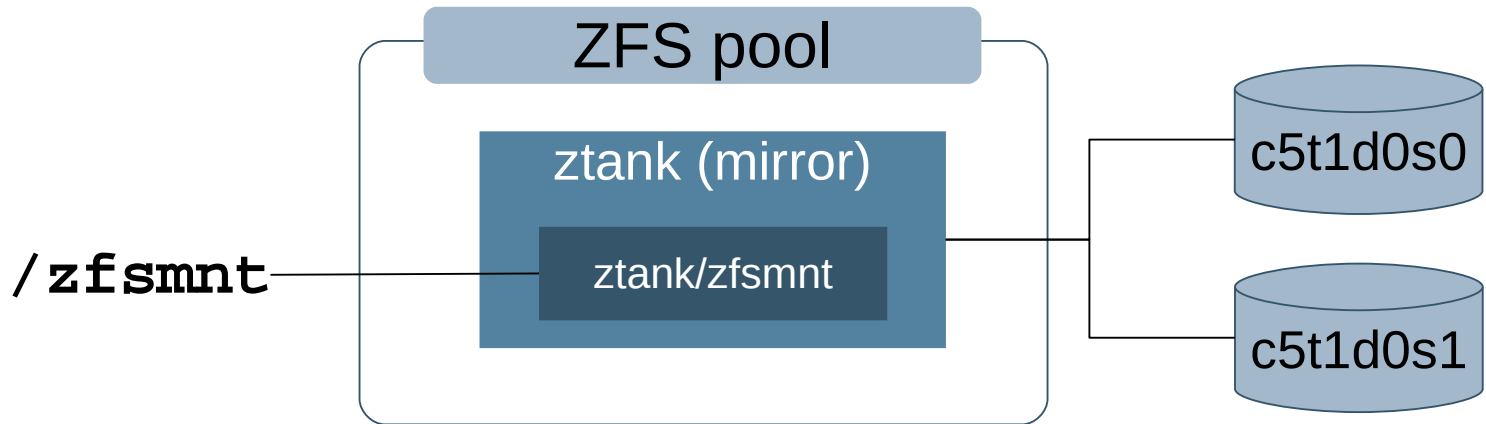


# ZFS デモンストレーション

## Solaris Volume Manager



## ZFS



# ZFS デモンストレーション

## ステータスの確認、状況の復帰 (SVM)

```

bash-3.2# metastat -i
d10: Mirror
  Submirror 0: d11
    State: Okay
  Submirror 1: d12
    State: Okay
  Pass: 1
  Read option: roundrobin (default)
  Write option: parallel (default)
  Size: 2654208 blocks (1.3 GB)

d11: Submirror of d10
  State: Okay
  Size: 2654208 blocks (1.3 GB)
  Stripe 0:
    Device      Start Block  Dbase      State Reloc Hot Spare
    c5t1d0s4      0          No         Okay   Yes

d12: Submirror of d10
  State: Okay
  Size: 2654208 blocks (1.3 GB)
  Stripe 0:
    Device      Start Block  Dbase      State Reloc Hot Spare
    c5t1d0s5      0          No         Okay   Yes

Device Relocation Information:
Device  Reloc Device ID
c5t1d0  Yes  id1,sd@AHITACHI_HDS7225SBSUN250G_0619NN9GKK=VDK41GT5EN9GKK

```

```

bash-3.2# pwd
/svmmnt
bash-3.2# ls
./                ../                C91BSML.tar.gz*

bash-3.2# gtar xvzf ./C91BSML.tar.gz | head -20
gtar: JDK: Cannot mkdir: I/O error
gtar: JDK/jre.pak: Cannot mkdir: No such file or directory

```

ステータスはOKにもかかわらず、  
ファイルシステムとしては機能を  
失っている

復旧させるためには、ボリュームの  
再構成とリストアが必要

# ZFS デモンストレーション

## ステータスの確認、状況の復帰 (ZFS)

```
bash-3.2# zpool scrub ztank
bash-3.2# zpool status ztank
pool: ztank
state: ONLINE
status: One or more devices could not be used because the label is missing or
invalid. Sufficient replicas exist for the pool to continue
functioning in a degraded state.
action: Replace the device using 'zpool replace'.
see: http://www.sun.com/msg/ZFS-8000-4J
scrub: scrub in progress for 0h0m, 62.40% done, 0h0m to go
config:

```

NAME	STATE	READ	WRITE	CKSUM	
ztank	ONLINE	0	0	0	
mirror	ONLINE	0	0	0	
c5t1d0s0	ONLINE	0	0	0	
c5t1d0s1	UNAVAIL	0	0	0	corrupted data

```
errors: No known data errors
```

```
bash-3.2# pwd
/zfsmnt
bash-3.2# ls
./                ../                C91BSML.tar.gz*

bash-3.2# gtar xvzf ./C91BSML.tar.gz
./
JDK/
JDK/jre.pak/
JDK/jre.pak/repository/
:
:
readme/readme_nd_de.html
readme/readme_nd_zh.html
readme/readme_nd_fr.html
readme/readme_nd_ko.html
readme/readme_nd_ja.html
readme/readme_nd_en.html
bash-3.2#
```

ディスクに障害があってもファイルシステムは機能し続ける

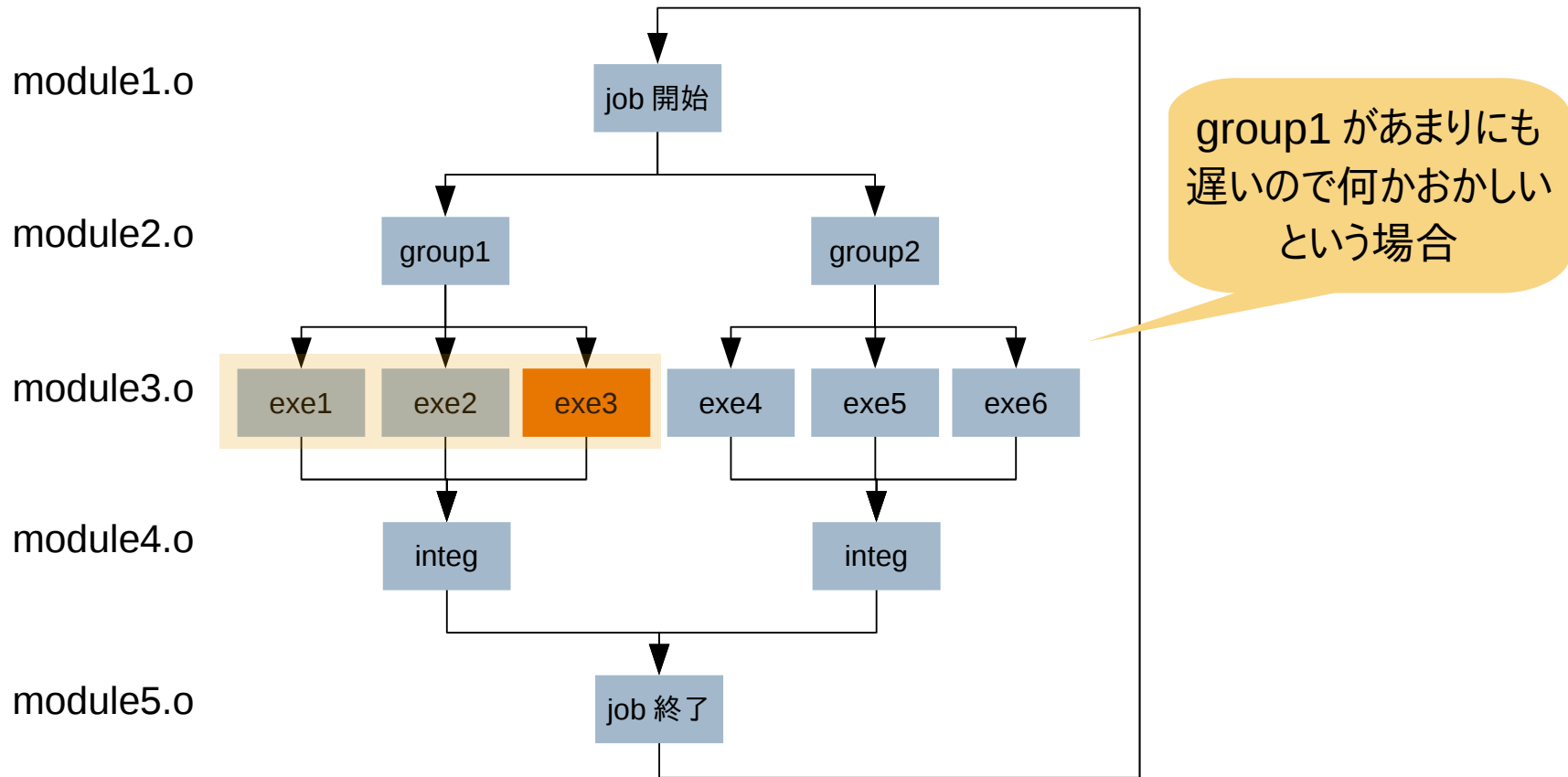
UNAVAIL(ABLE) なディスクを復旧させるために交換

```
zpool replace -f c5t1d0s1 c5t1d0s3
                        (new disk)
```

# Solaris 10 諸機能再レビュー

DTrace

# バッチジョブを debug していき詰まる例



- module3 でボトルネックが発生しているように見える
  - > プロセスやスレッドを特定する方法は？
  - > 原因は？(データに依存するのか、ハードウェアに起因するのか)

# ありがちなデバック手法 ....

module3.c

```
int execute(char *args) {  
    printf("In module3::execute\n");  
    -----  
    printf("Out module3::execute\n");  
}
```

- マルチプロセス、マルチスレッド環境下で printf 出力されるメッセージでは時系列に沿った因果関係を把握することが困難
  - > ますます printf の挿入して、判断に必要な情報を画面出力しようとする
- 一行 printf の挿入毎にオーバーヘッド
  - > 数ナノ秒～ミリ秒の世界で起こるリソースの競合を調査しようとしているところに、ミリ秒～秒レベルの分解能を持ったコードが差し込まれる
  - > スルーポイントが落ちるため、ボトルネックが再現しなくなる

# DTrace デモンストレーション

```
#include <stdio.h>

int dummy();
int i=0;
double x=0;

int main(void) {
    // for (i=0; i<100000000; i++) // 1億回
    // {
    //     dummy();
    // }
    // printf("result = %g at i = %d\n", x, i);

    dummy();

    return 0;
}

int dummy() {
    x = (double)i * 0.0000001;
    return 0;
}
```

blank.c

```
#include <stdio.h>

int dummy();
int i=0;
double x=0;

int main(void) {
    // for (i=0; i<100000000; i++) // 1億回
    // {
    //     dummy();
    // }
    // printf("result = %g at i = %d\n", x, i);

    dummy();

    return 0;
}

int dummy() {
    printf("In -dummy i= %d ....", i);
    x = (double)i * 0.0000001;
    printf("Out -dummy\n");
    return 0;
}
```

debug?

print.c

```
#!/usr/sbin/dtrace -s

pid$target::dummy:entry
{
    self->ts = vtimestamp;
}

pid$target::dummy:return
{
    timespent = vtimestamp - self->ts;
    printf("ThreadID %d spent %d nsecs in %s",
        tid, timespent, probefunc);
    self->ts = 0;
    timespent = 0;
}
```

cpucost.d

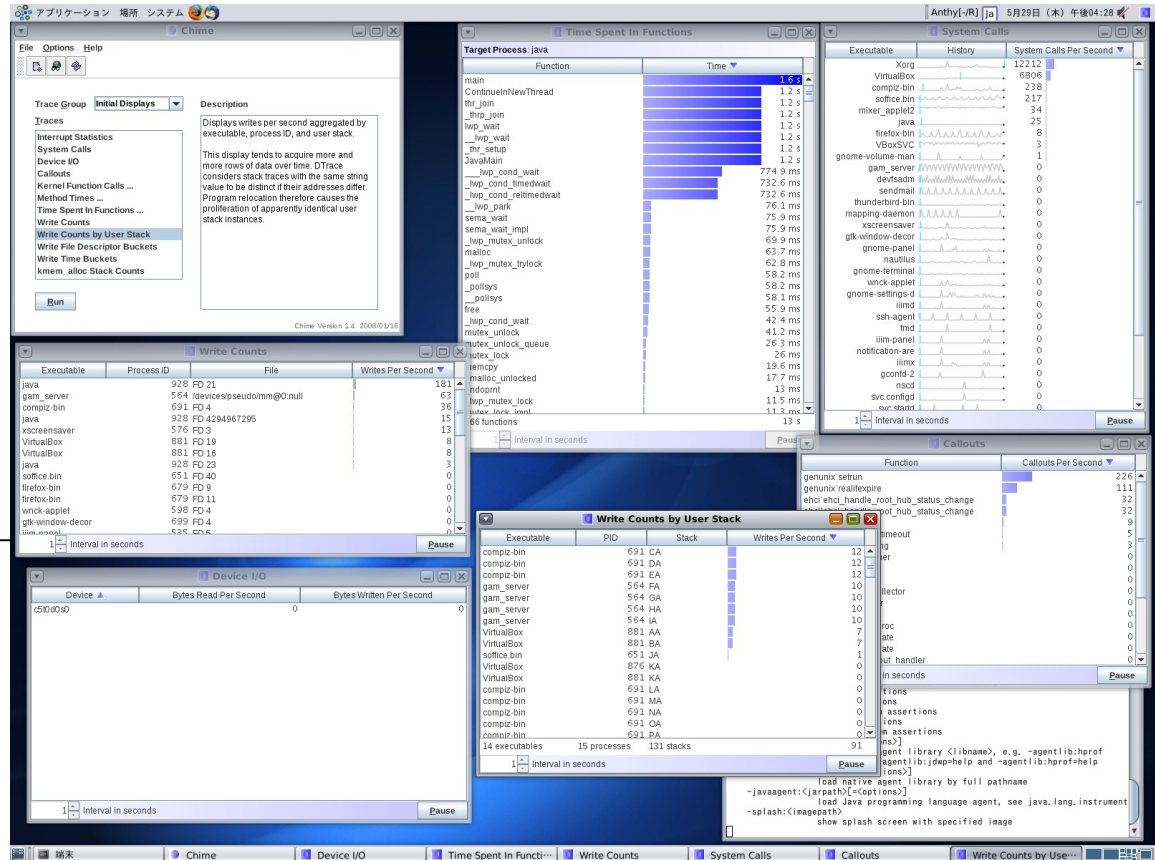
```
bash-3.2# ./cpucost.d -c ./blank
dtrace: script './cpucost.d' matched 2 probes
dtrace: pid 1850 has exited
CPU    ID          FUNCTION:NAME
  2    68143      dummy:return ThreadID 1 spent 2357 nsecs in dummy

bash-3.2# ./cpucost.d -c ./print
dtrace: script './cpucost.d' matched 2 probes
In -dummy i= 0 ....Out -dummy
dtrace: pid 1852 has exited
CPU    ID          FUNCTION:NAME
  0    68143      dummy:return ThreadID 1 spent 152253 nsecs in dummy
```

オーバーヘッドが増えてボトルネックが再現しなくなる可能性

# DTrace で状況を把握し原因を絞り込む

- オーバーヘッドが極小
- オンラインで実施



```
#!/usr/sbin/dtrace -s
```

```
syscall::open*:entry,  
syscall::close*:entry
```

```
{  
    self->ts = timestamp;  
}
```

```
syscall::open*:return,  
syscall::close*:return
```

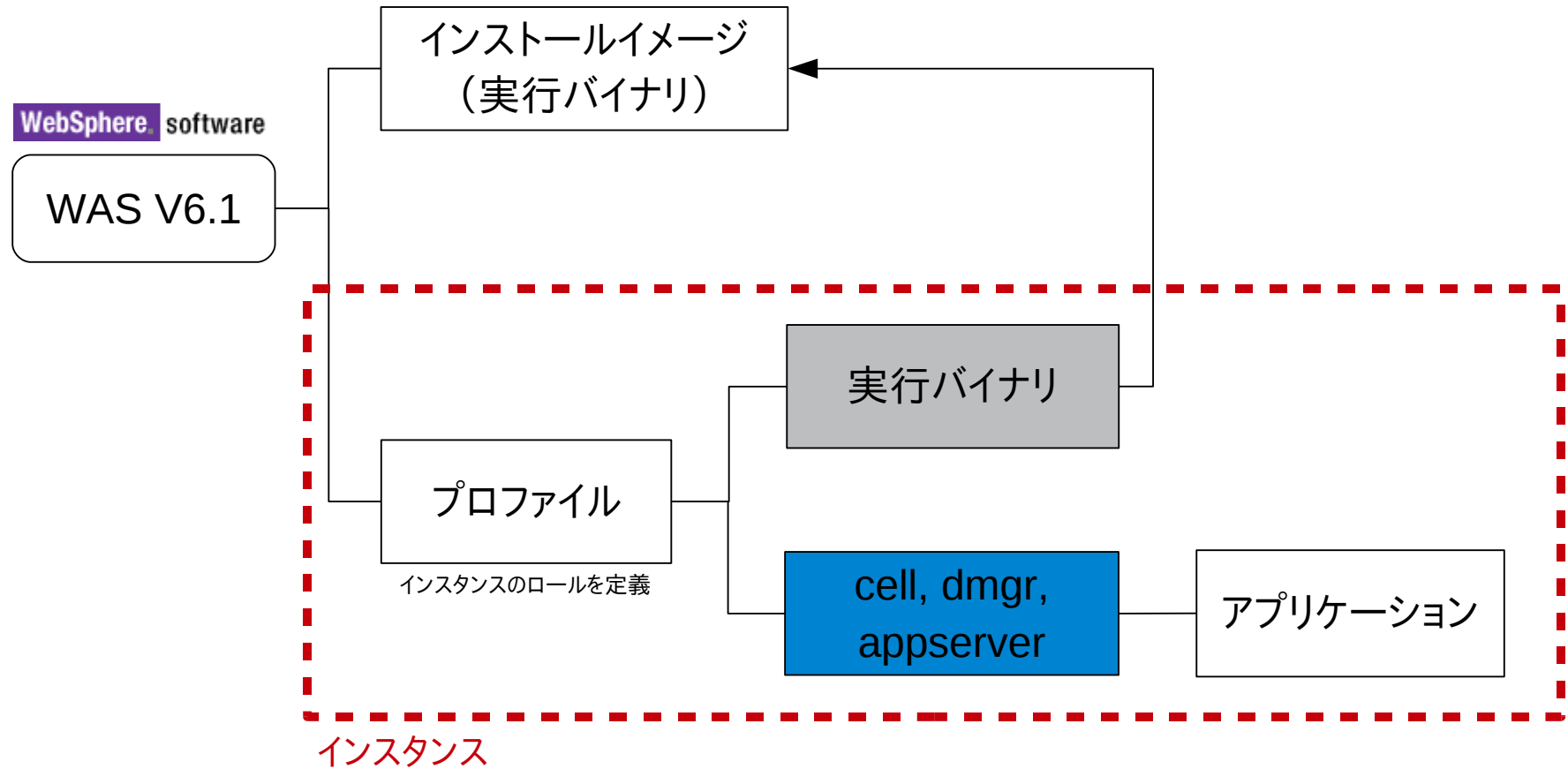
```
{  
    timespent = timestamp - self->ts;  
    printf("ThreadID %d spent %d nsecs in %s", tid, timespent, probefunc);  
    self->ts = 0;  
    timespent = 0;  
}
```



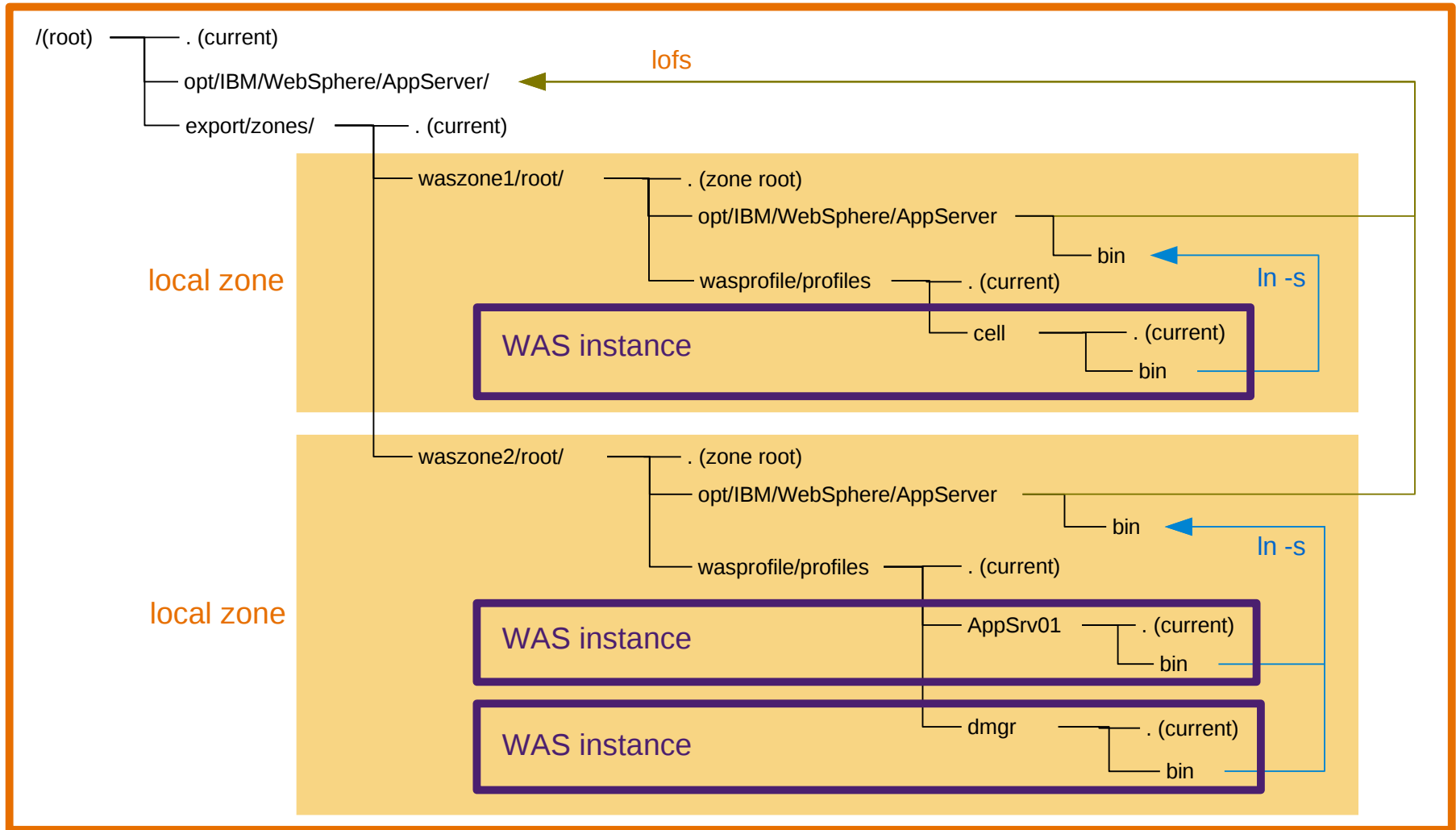
# ISV, OSS アプリケーションデモ

# IBM WebSphere Application Server V6.1

## 製品構造



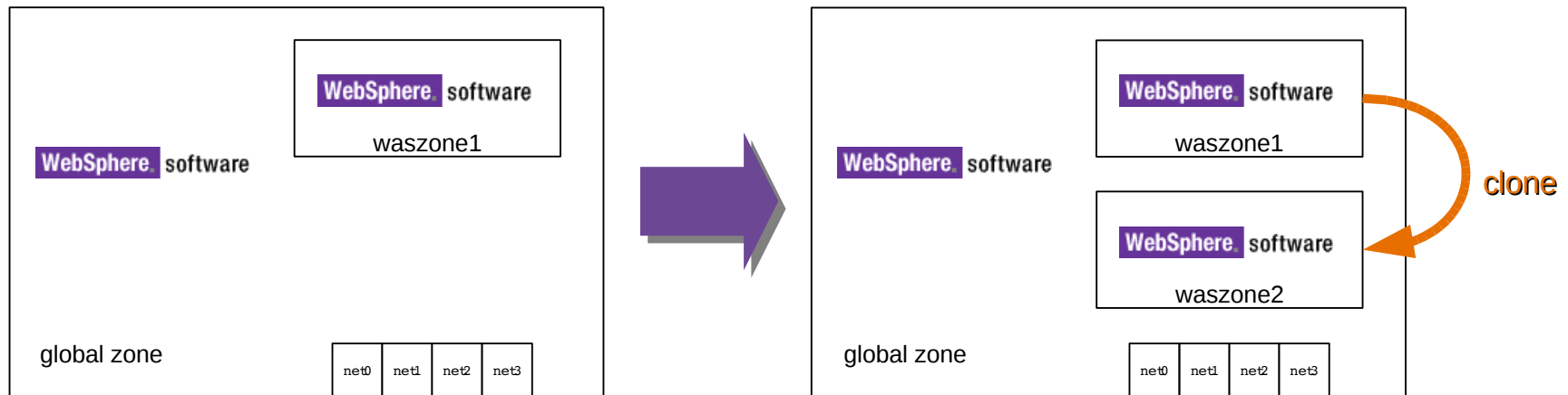
# IBM WebSphere Application Server V6.1



global zone

要点のみ記載: 実際のディレクトリ数や構造の深さについて異なる場合があります。

# 状況の設定



- パフォーマンスに不満があり問題点を把握していきたい
  - > 目標
    - システムリソースの確認をしたい
    - アプリケーションのプロファイルを取りたい
  - > 手法
    - OS コマンドを使って基本的な情報を取得する
    - JDK5 は native に DTrace 対応していないので dvm を利用
    - 本番 zone を clone させ、検証環境を構築する
    - ( dvm はオーバーヘッドがあるため切り分け環境を用意したほうがよい)

# OS コマンドによるボトルネック調査

Example vmstat output

kthr			memory		page				disk				faults		cpu			id		
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s0	s2	s4	--	in	sy		cs	us
46	0	0	23158808	8700152	0	5	0	10	10	0	0	5	0	0	9750	66324	22325	62	19	18
46	0	0	23154296	8682592	0	6	0	12	12	0	0	4	0	0	8855	66115	21963	63	19	18
28	0	0	23146520	8653920	0	13	0	3	3	0	0	2	0	0	8519	64506	20813	63	19	18
53	0	0	23143616	8636272	0	7	0	3	3	0	0	3	0	0	8987	65462	21436	62	19	19
47	0	0	23139520	8616152	0	12	0	3	3	0	0	3	0	0	7747	66572	20841	65	19	16
38	0	0	23123136	8583000	0	5	0	9	9	0	0	4	0	0	7436	65995	21025	64	19	17
40	0	0	23119856	8564560	0	9	0	3	3	0	0	3	0	0	8624	66608	22553	64	19	17

Example mpstat output

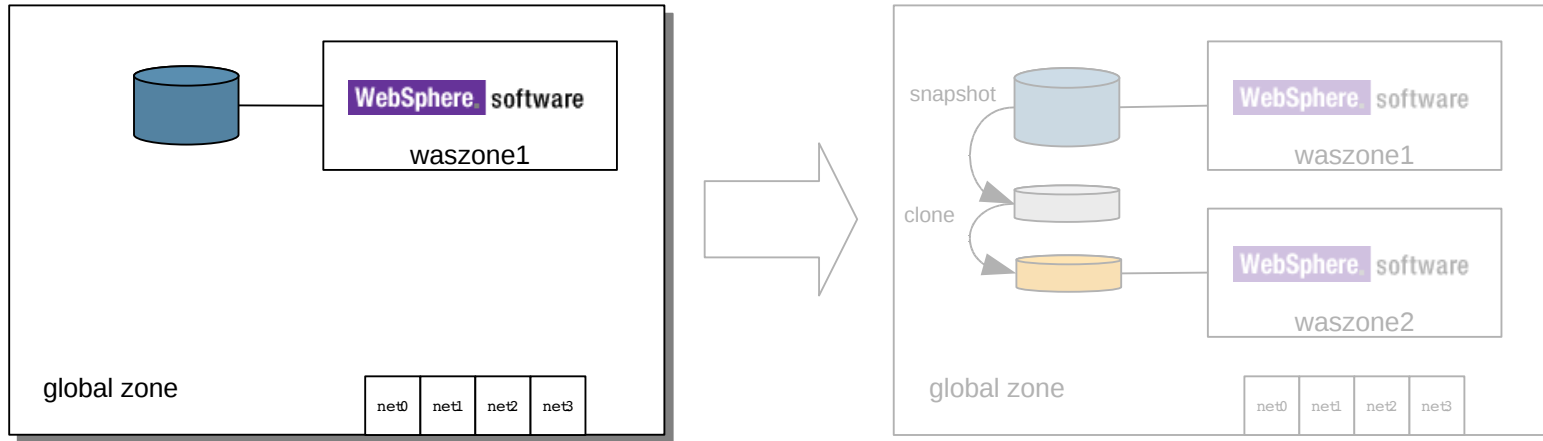
CPU	minf	mjf	xcal	intr	ithr	csw	icsw	migr	smtx	srw	syscl	usr	sys	wt	idl
0	1	0	944	458	180	1472	54	323	683	0	5062	69	17	0	14
1	1	0	82	216	0	1389	55	304	784	0	5172	67	17	0	16
2	1	0	78	223	0	1460	60	302	722	0	4864	68	15	0	17
3	0	0	65	182	0	1394	50	283	641	0	5089	68	14	0	18
8	1	0	60	239	2	1549	64	317	717	0	5319	69	16	0	16
9	1	0	68	183	2	1767	53	293	656	0	5256	69	14	0	17
10	1	0	54	200	0	1558	55	297	691	0	5345	69	15	0	17
11	1	0	74	172	0	1390	53	270	657	0	5314	67	14	0	18
16	1	0	57	232	1	1507	63	303	723	0	5549	69	15	0	16
17	1	0	67	181	0	1565	53	279	683	0	5109	68	15	0	17
18	1	0	85	190	0	1507	51	286	643	0	4847	69	14	0	17
19	1	0	57	171	0	1346	53	267	643	0	4799	67	14	0	19
24	1	0	89	189	0	1445	51	295	565	0	4934	71	12	0	17
25	4	0	58	164	0	1379	48	281	588	0	5066	70	12	0	18
26	1	0	71	145	0	1159	48	254	457	0	4372	70	14	0	16
27	0	0	1576	4874	4840	335	91	34	2260	0	385	7	93	0	0

# OS コマンドによるボトルネック調査

Example intrstat output

device	cpu16 %tim	cpu17 %tim	cpu18 %tim	cpu19 %tim
e1000g#0	0 0.0	0 0.0	0 0.0	0 0.0
e1000g#1	0 0.0	0 0.0	0 0.0	0 0.0
e1000g#2	0 0.0	0 0.0	0 0.0	0 0.0
mpt#0	0 0.0	0 0.0	0 0.0	0 0.0
device	cpu24 %tim	cpu25 %tim	cpu26 %tim	cpu27 %tim
e1000g#0	0 0.0	0 0.0	0 0.0	1726 88.7
e1000g#1	0 0.0	0 0.0	0 0.0	0 0.0
e1000g#2	0 0.0	0 0.0	0 0.0	0 0.0
mpt#0	0 0.0	0 0.0	0 0.0	0 0.0
device	cpu0 %tim	cpu1 %tim	cpu2 %tim	cpu3 %tim
e1000g#0	0 0.0	0 0.0	0 0.0	0 0.0
e1000g#1	0 0.0	0 0.0	0 0.0	0 0.0
e1000g#2	0 0.0	0 0.0	0 0.0	0 0.0
mpt#0	1 0.0	0 0.0	0 0.0	0 0.0

# zone cloning



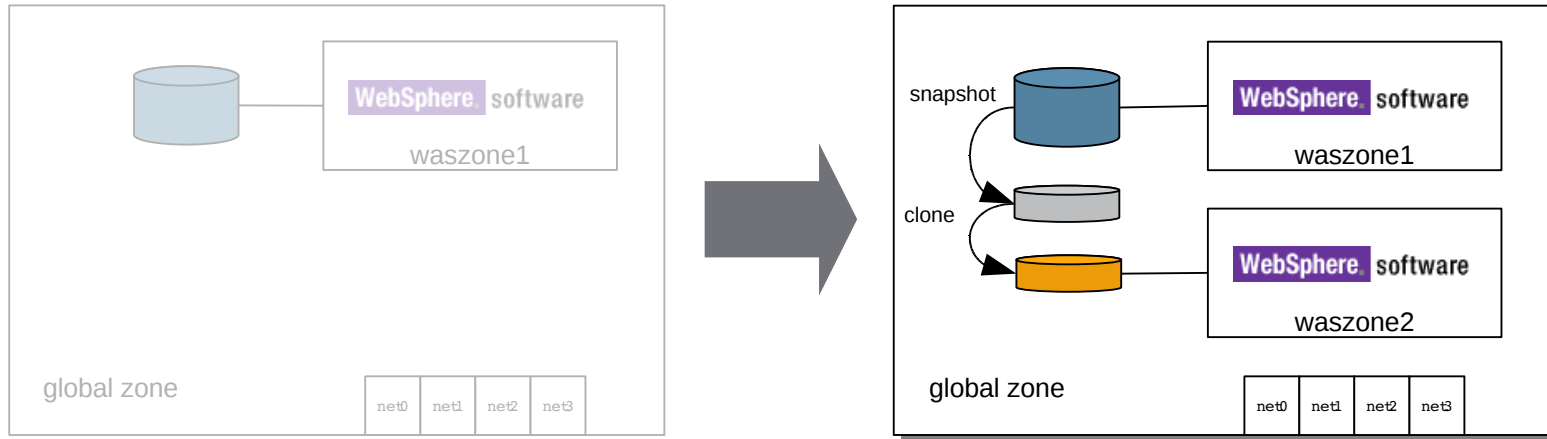
```
bash-3.2# zoneadm list -civ
```

ID	NAME	STATUS	PATH	BRAND	IP
0	global	running	/	native	shared
-	dbzone	installed	/export/zones/dbzone	ipkg	shared
-	waszone1	installed	/export/zones/waszone1	ipkg	shared

```
bash-3.2# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
(省略)				
rpool/export/zones/waszone1	242M	204G	242M	/export/zones/waszone1

# zone cloning



```
bash-3.2# zoneadm list -civ
```

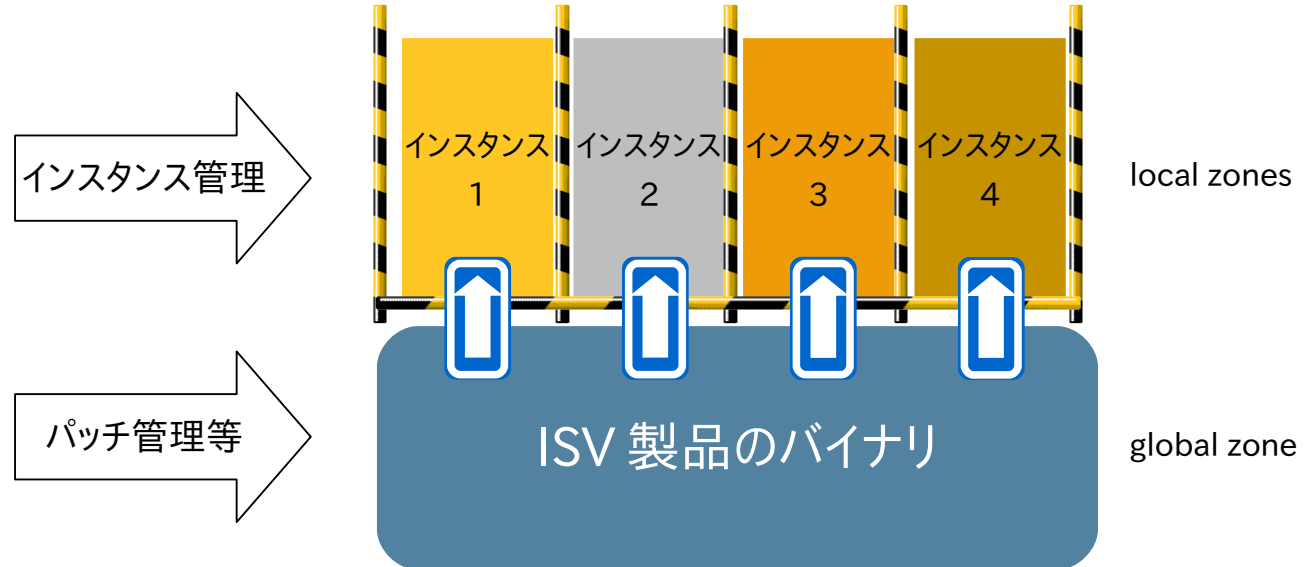
ID	NAME	STATUS	PATH	BRAND	IP
0	global	running	/	native	shared
39	waszone2	running	/export/zones/waszone2	ipkg	shared
-	dbzone	installed	/export/zones/dbzone	ipkg	shared
-	waszone1	installed	/export/zones/waszone1	ipkg	shared

```
bash-3.2# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
(省略)				
rpool/export/zones/waszone1	242M	204G	242M	/export/zones/waszone1
rpool/export/zones/waszone1@base_snap	0	-	242M	-
rpool/export/zones/waszone2	3.00M	204G	244M	/export/zones/waszone2

# 同様な構築方針が適用可能なアプリケーション

- IBM WebSphere MQ
- IBM DB2
- Apache
- PostgreSQL
- MySQL
- 等多数



# Viking Zone

## 仮想化「アプライアンス」

Made In Japan



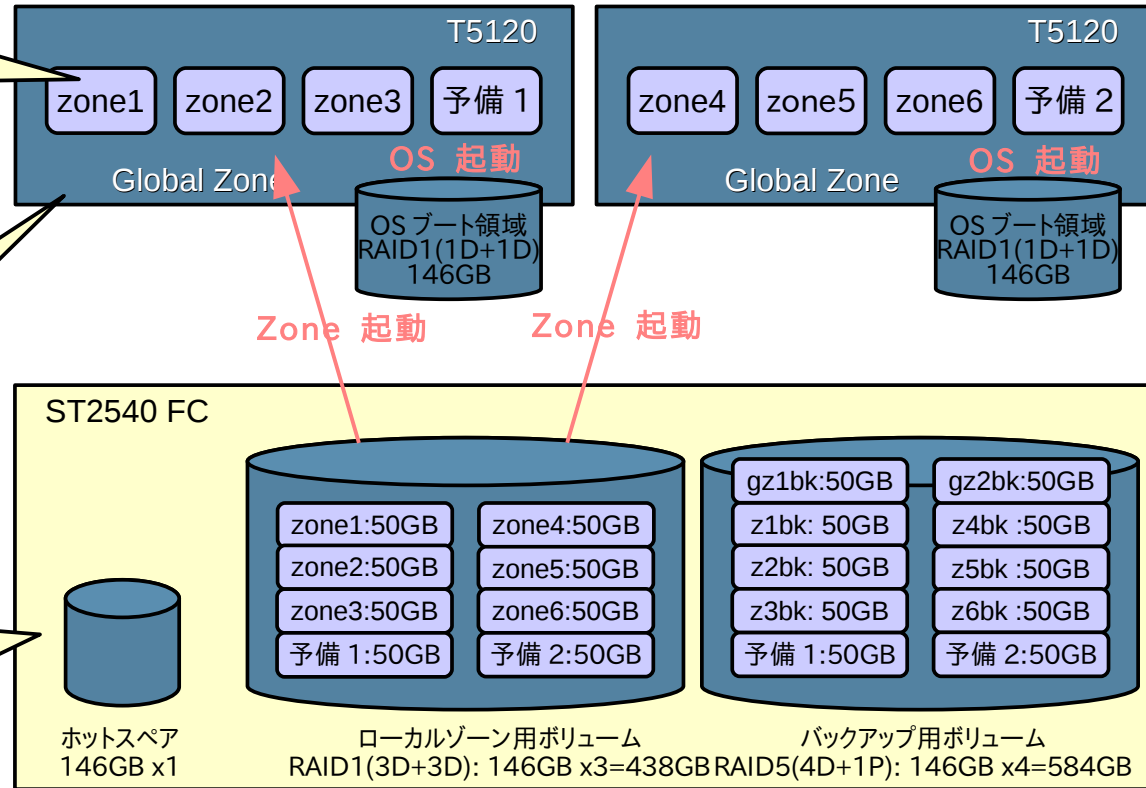
### ローカルゾーンのサービス例

Web/AP/DB/Mail/LDAP  
 ユーザアプリ / 開発環境  
 等々選択は自由!!

### グローバルゾーンのサービス例

DNS/NFS/NTP 等の  
 共有サービス  
 CAM 等の管理サービス  
 バックアップサービス

ゾーンのシステム領域は  
 ミラーリングで冗長化  
 バックアップ用領域は RAID5  
 で冗長化  
 ホットスペアディスクを用意



# まとめ

# まとめ

- **Solaris 10**
  - > サンの伝統を継承し、効果的に進化
  - > 無償
  - > 仮想化に対する最も正しい理解と実装
- **実運用での応用**
  - > アプリケーションの構造と特徴に応じた配備
  - > シンプルなコマンドによる安全確実な操作
  - > 複雑なシステムを開発・維持するための適切なツールの利用

経済性



生産性  
信頼性



時間稼働率  
性能稼働率

運用性能の最大化



# アプリケーションの運用性能を 最大化させる Solaris10 の底力

国谷俊夫  
Toshio.Kuniya@Sun.COM

# Redbook: IBM WebSphere on Solaris 10

- Sun と IBM の共同著書

> <http://www.redbooks.ibm.com/redpieces/abstracts/sg247584.html?Open&pdfbookmark>

- マニュアルに記載されていない WAS の Solaris10 における運用
- 英語のみ .... ( Solaris Zone 部分の日本語要約を準備中)

