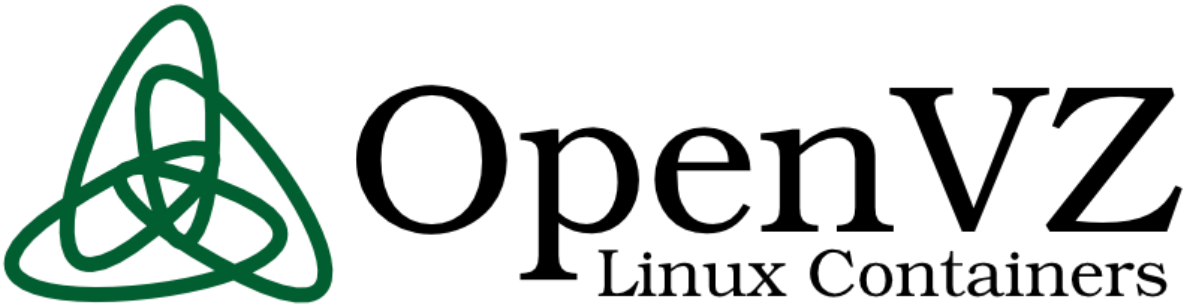


海老澤健太郎 @ebiken
パラレルス株式会社

本日の Topics

- 「コンテナ＝OS仮想化」とは何か？
  - 仮想化方式の比較
  - OS仮想化の特徴
  - OS仮想化の利用例
- OS仮想化の実現方法
  - リソース空間の分離
  - リソース管理
  - namespace / cgroups
- OpenVZのネットワーク
  - ネットワークタイプ
  - eth/veth/venet性能比較
- OpenVZ設定方法
- APPENDIX

コンテナ型仮想化のいろいろ

- OpenVZ
- Virtuozzo
- LXC（Linux Container）
- Linux-Vserver
- Solaris Zones
- FreeBSD Jails

- Google Container（独自）
- Facebook??

「コンテナ＝OS仮想化」
とは何か？

仮想マシン（VM）
ハードウェア仮想化と何が違うのか？

ハードウェア仮想化	OS仮想化
仮想マシン Virtual Machine（VM）	コンテナ Container（CT） Virtual Environment（VE）
仮想化のレベル	
ハードウェア環境を仮想化	OS環境を仮想化
物理サーバー毎の混在環境	
異なるOSが共存可能	異なるディストリビューションが共存可能（カーネル共通）
仮想環境の起動	
OSを起動	プロセスを起動
ネットワークアクセス	
仮想ハードウェア経由	仮想インターフェース経由

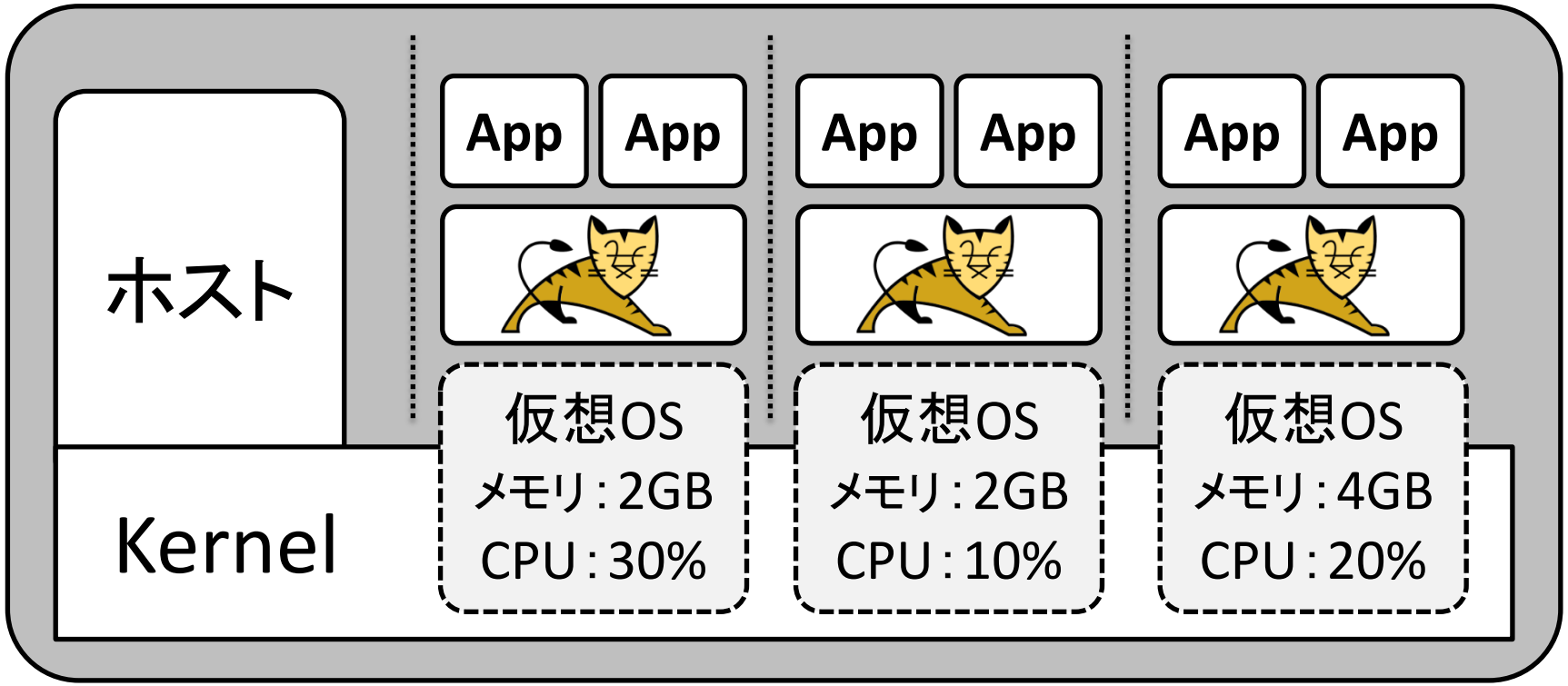
OS仮想化＝コンテナの特徴

起動が早い（OSはブート済み）
高密度（カーネル・メモリを共有）
ハードウェア仮想化のオーバーヘッド無し

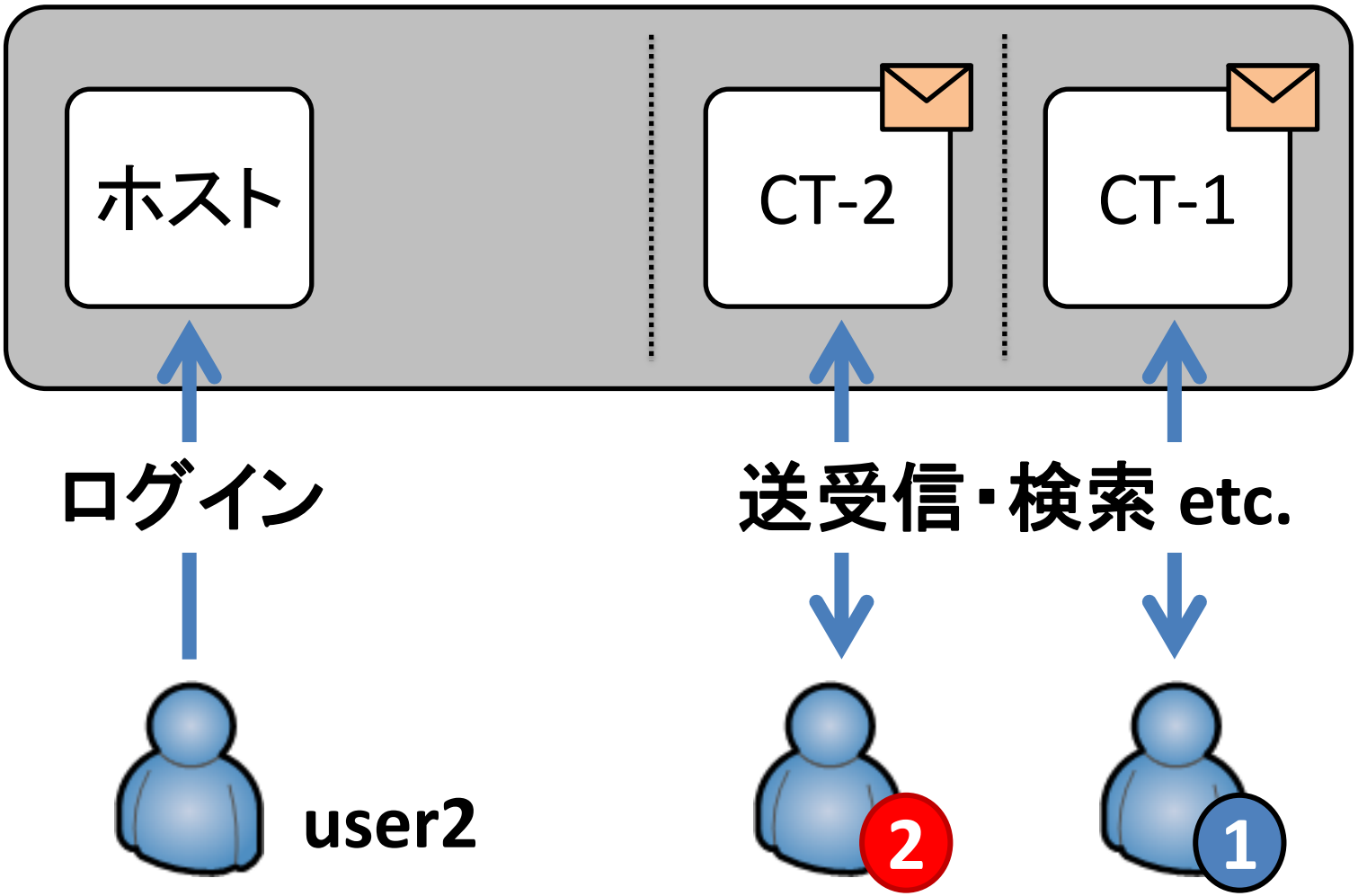
コンテナ毎のリソース

root / users / groups	IPアドレス
メモリ	プロセス
設定ファイル	ライブラリ

PaaS プラットフォーム



GoodMail（仮）- 某Mailサービス



「コンテナ＝OS仮想化」

どうやって実現するの？

Isolation & Resource Control

"Adding Generic Process Containers to the Linux Kernel"
Paul B. Menage @ Google / Balbir Singh and Srivatsa Vaddagiri @ IBM

- (Namespace) Isolation
  - Mechanism which adds an additional indirection or translation layer to the naming/visibility of some unix resource space (such as process ids, or network interfaces) for a specific set of processes.
  - Typically the existence of isolation itself is invisible to the processes being isolated.

- Resource control
  - Mechanism which can do either or both of:

  A. Tracking how much of a resource is being consumed by a set of processes

  B. imposing quantitative limits on that consumption, either absolutely, or just in times of contention.

リソース空間の分離 & リソースの管理

- リソース空間の分離
  - プロセスの集合を識別するための名前付けを実施、
  - リソース空間を分割し、参照可能な範囲を規定。
  - 各プロセスは分離されている事を認識しない

- リソース管理
  - あるプロセスの集合が使用しているリソースを追跡
  - リソース使用量の制限（絶対量 or 相対量）

※ リソース空間＝プロセスID、ネットワークインターフェース、メモリ、CPU 等

リソース空間の分離 & リソースの管理

- リソース空間の分離
  - プロセスの集合を識別するための名前付けを実施、
  - リソース空間 **namespace** 範囲を規定。
  - 各プロセスは分離されている事を認識しない

- リソース管理
  - あるプロセス **cgroups** るリソースを追跡
  - リソース使用 r 相対量）

※ リソース空間＝プロセスID、ネットワークインターフェース、メモリ、CPU 等

namespace & cgroups

namespace
File system
IPC
Networking
/proc
/sys
User
UTS

cgroups
cpuset
cpu
cpuacct
devices
freezer
blkio

cgroups = Control Groups

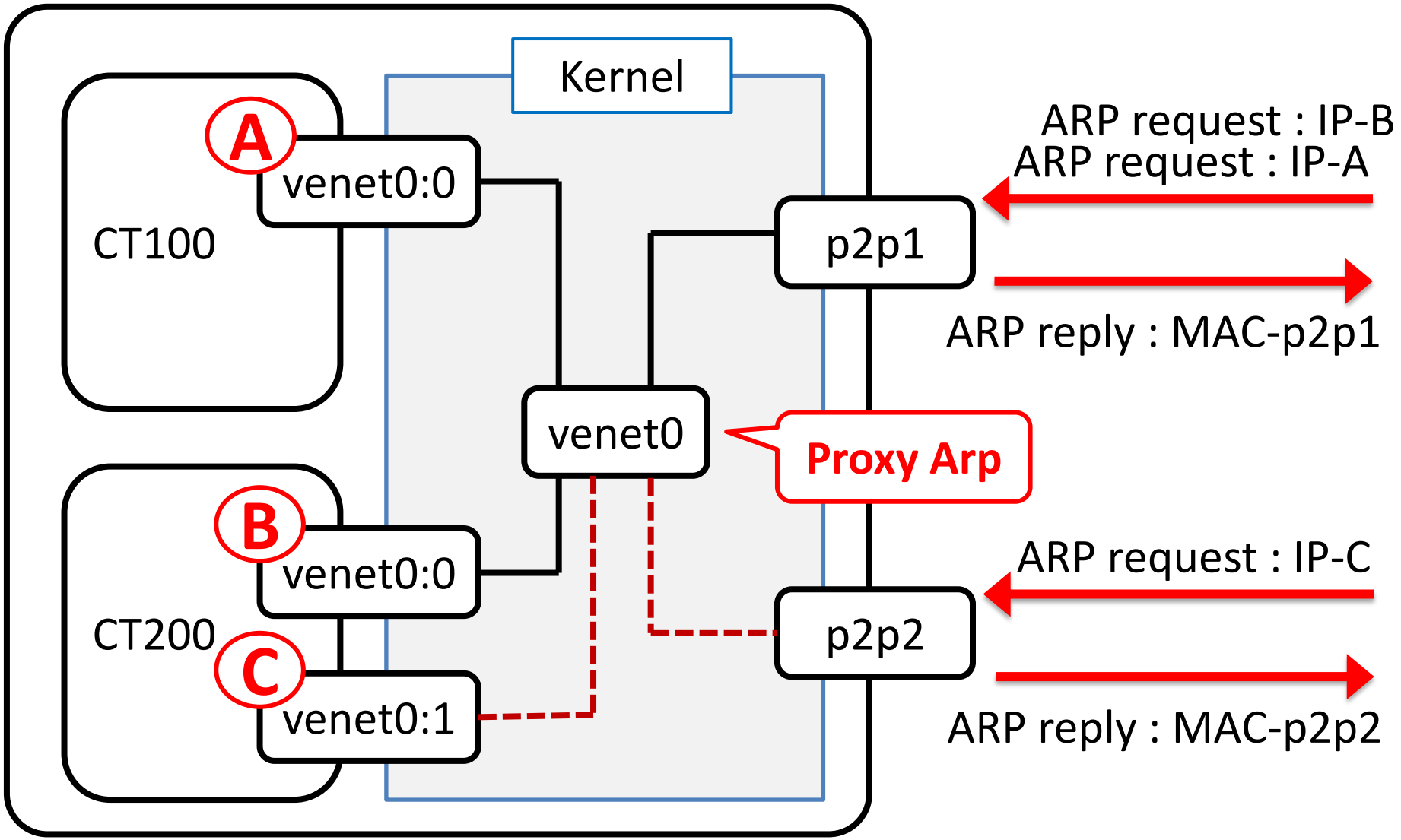
OPENVZのネットワーク

OpenVZのネットワークタイプ

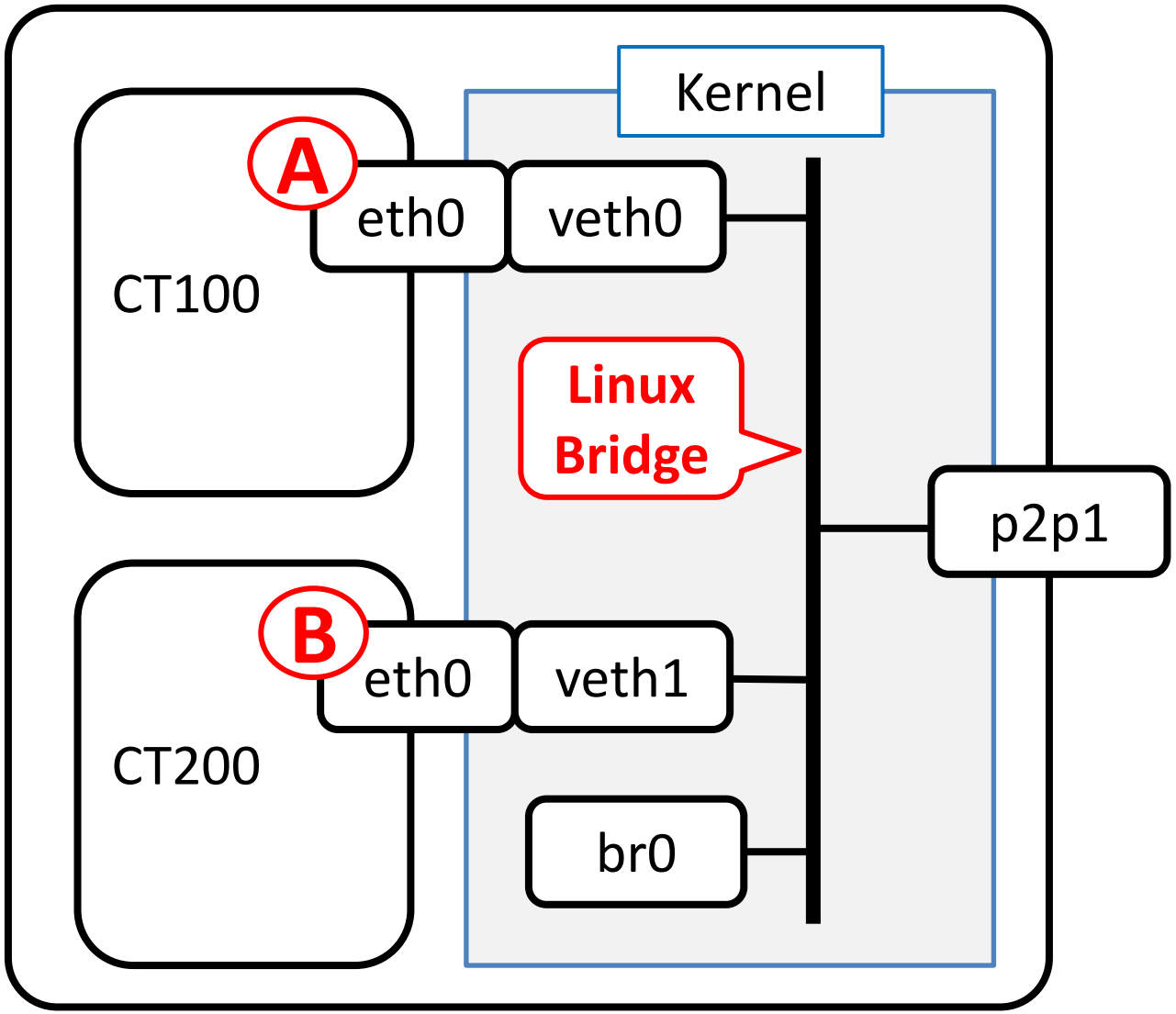
	venet	veth
Layer	Layer 3 (IP)	Layer 2 (Ethernet)
受信	venet0 経由	Linux Bridge経由
L2アドレス（ARP返信）	無し（*1）	veth毎（*2）

（*1）Proxy ARPにより宛先インタフェースのMACアドレスを返信
（*2）veth毎に異なるMACアドレスを返信

Logical Topology: venet



Logical Topology: veth



ARP request : IP-A

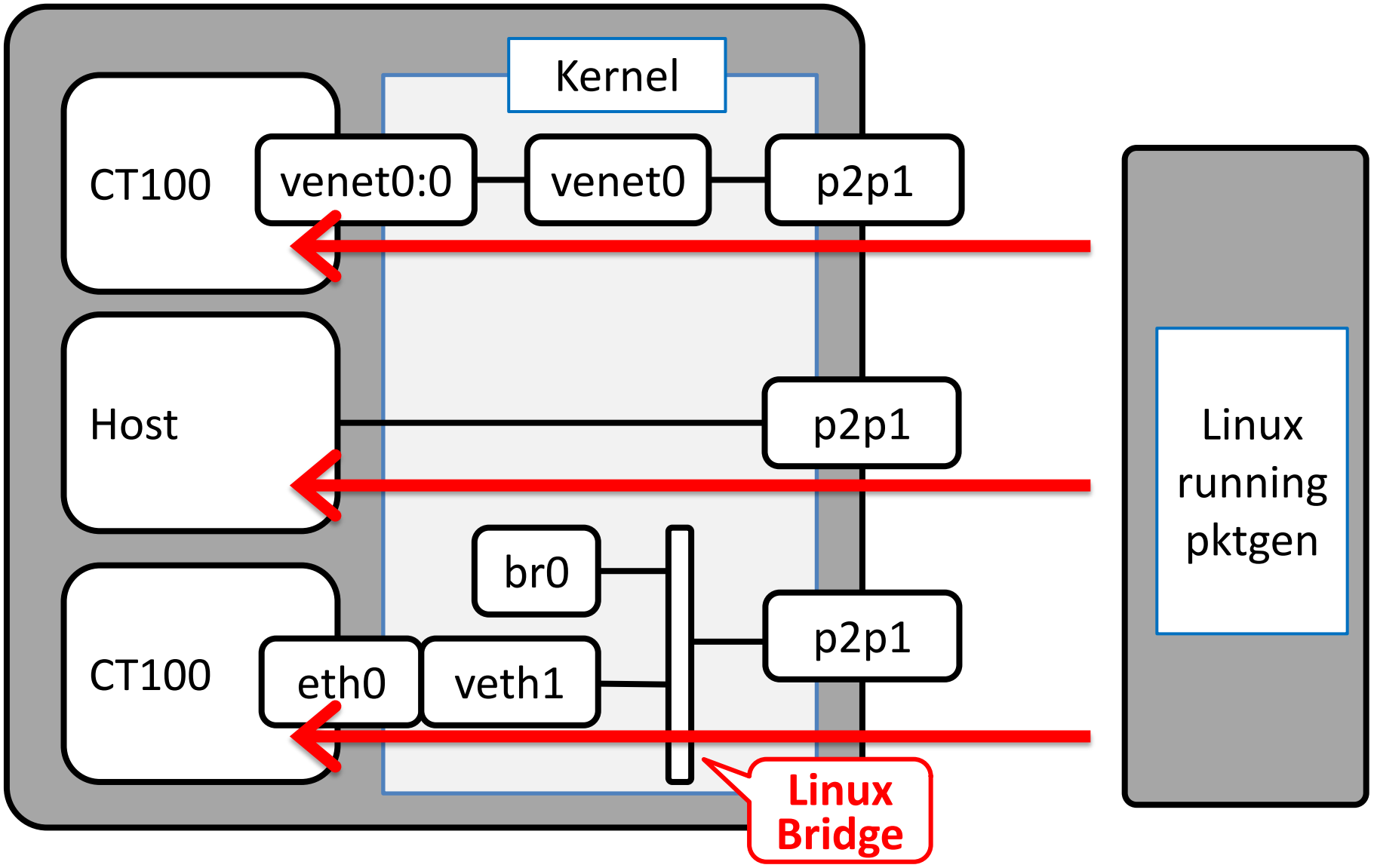
ARP reply : MAC-A

ARP request : IP-B

ARP reply : MAC-B

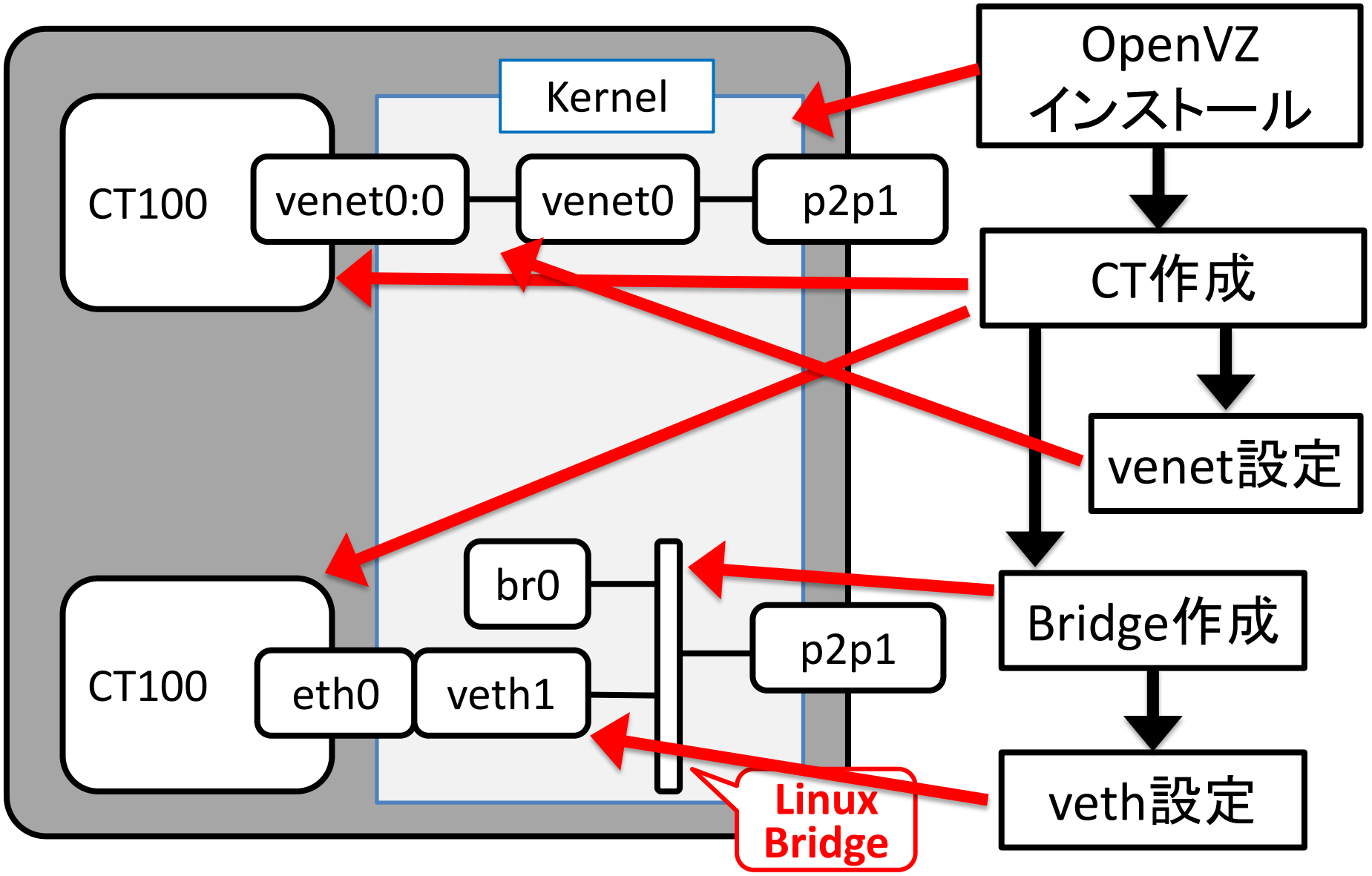
ETH/VETH/VENET性能比較

eth/veth/venet性能比較：測定環境



OPENVZ, VETH/VENET 設定方法

OpenVZ, veth/venet 設定方法



OpenVZ, veth/venet 設定方法

http://wiki.openvz.org/Quick_installation

1. OpenVZ Kernel のインストール

```
# wget -P /etc/yum.repos.d/ http://download.openvz.org/openvz.repo
# rpm --import  http://download.openvz.org/RPM-GPG-Key-OpenVZ

# yum install vzkernel
```

2. /etc/sysctl.conf の設定（追加＋変更）

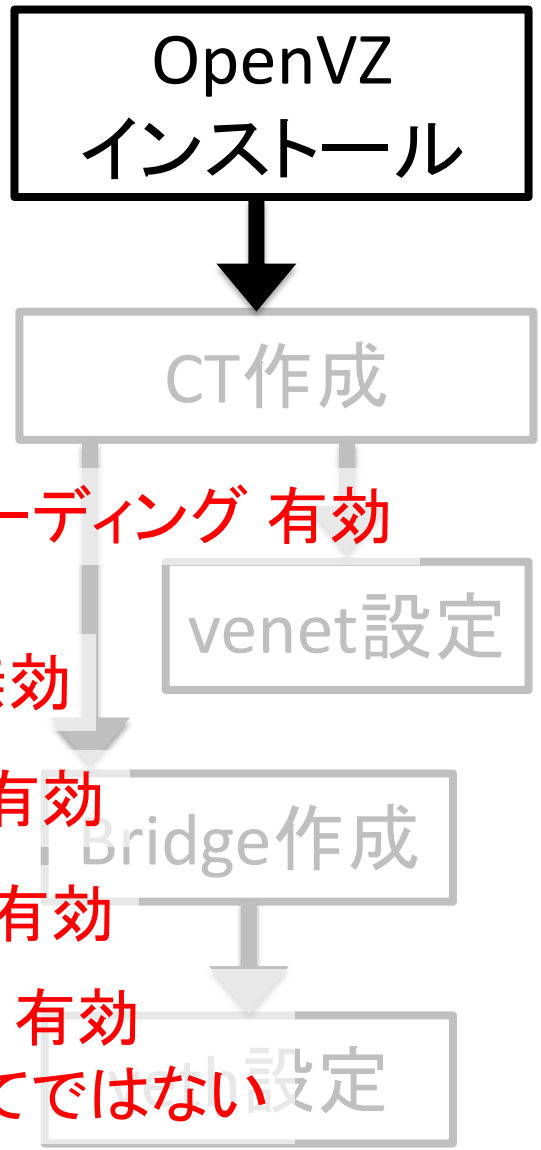
```
net.ipv4.ip_forward = 1
net.ipv6.conf.default.forwarding = 1
net.ipv6.conf.all.forwarding = 1
```
パケットフォワーディング 有効

```
net.ipv4.conf.default.proxy_arp = 0
```
プロキシARP 無効

```
net.ipv4.conf.all.rp_filter = 1
```
送信元IP検証 有効

```
kernel.sysrq = 1
```
マジック SysRq 有効

```
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
```
リダイレクト 有効
但し、すべてではない



http://wiki.openvz.org/Quick_installation

OpenVZ, veth/venet 設定方法

3. SELinux 無効化

```
# vi /etc/sysconfig/selinux
SELINUX=disabled
```

4. リブートし、OpenVZで起動された事を確認

```
#  uname -s -r -p -o
Linux 2.6.32-042stab063.2 x86_64 GNU/Linux
```

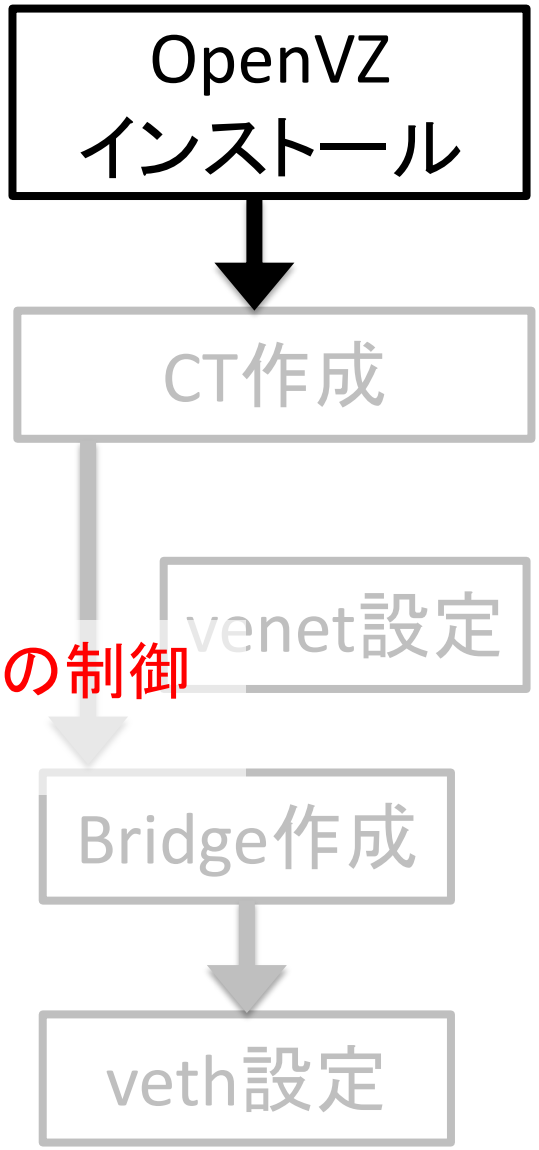
5. ツールをインストール

```
# yum install vzctl vzquota
```

vzctl … OpenVZコンテナの制御
vzquota … クォータ管理

6. OpenVZ カーネルモジュールの開始
（リブート時は自動）

```
# /sbin/service vz start
```



OpenVZ, veth/venet 設定方法

http://wiki.openvz.org/Quick_installation

1. コンテナのテンプレートを取得

```
# pwd
/vz/template/cache
# wget
http://download.openvz.org/template/precreated/centos-6-x86-devel.tar.gz
```

2. コンテナ作成＆メモリ設定

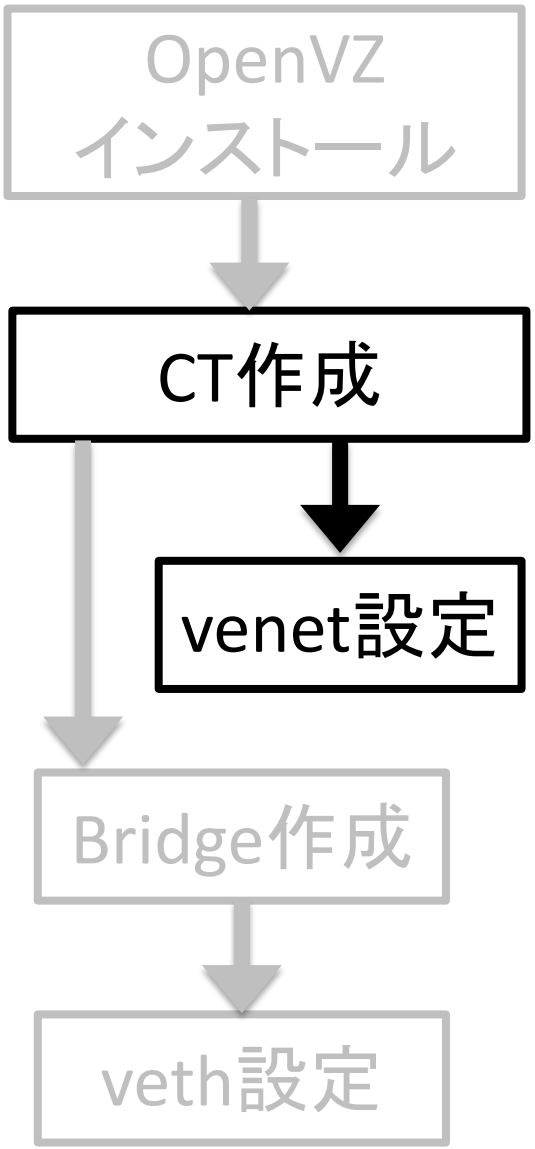
```
# vzctl create 100 --hostname ct100
                   --ostemplate centos-6-x86-devel
# vzctl set 100 --ram 1024M --swap 1024M --save
```

3. IP アドレス、nameserver設定（CT内venet自動生成）

```
# vzctl set 100 --ipadd 10.10.0.1
                --nameserver 8.8.8.8 --save
```

4. コンテナ開始

```
# vzctl start 100
```



OpenVZ, veth/venet 設定方法

http://wiki.openvz.org/Quick_installation

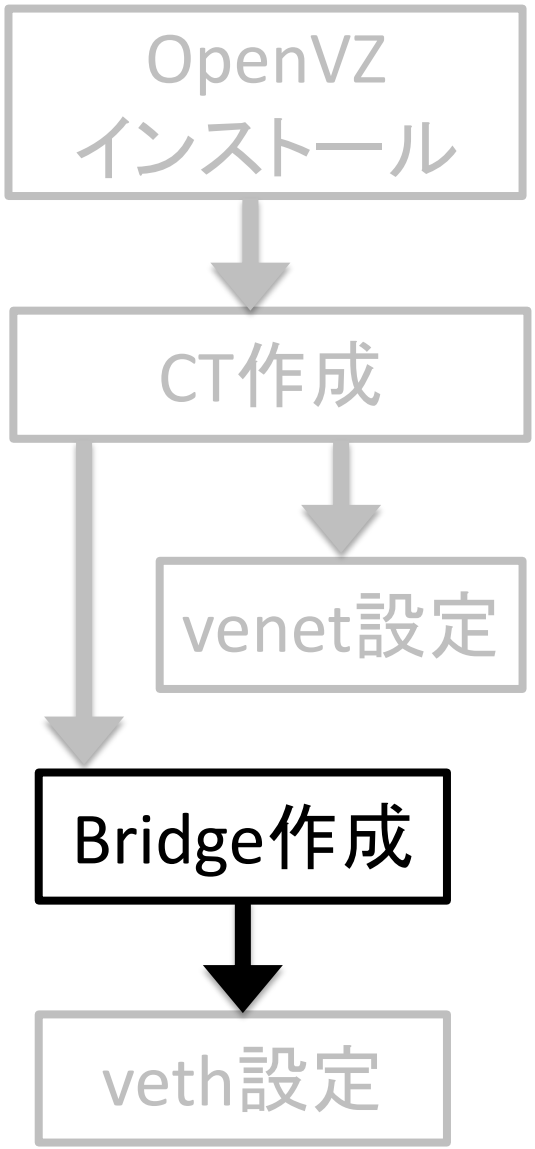
1. bridge-utilsのインストール
[host]# yum install bridge-utils

2. Bridge (vzbr0) の作成
[host]# cd /etc/sysconfig/network-scripts/
[host]# vi ifcfg-vzbr0
DEVICE="vzbr0"
ONBOOT="yes"
TYPE="Bridge"
BOOTPROTO="static"
NETMASK=255.255.255.0
IPADDR=10.10.0.10

3. eth0 (*) の Bridge への参加＋IPコメントアウト
[host]# vi ifcfg-eth0
#IPADDR=10.10.0.1
#NETMASK=255.255.255.0
BRIDGE="vzbr0"

(*) ホスト側物理NIC

4. [host]# service network restart



OpenVZ, veth/venet 設定方法

http://wiki.openvz.org/Quick_installation

5. コンテナへの veth 追加

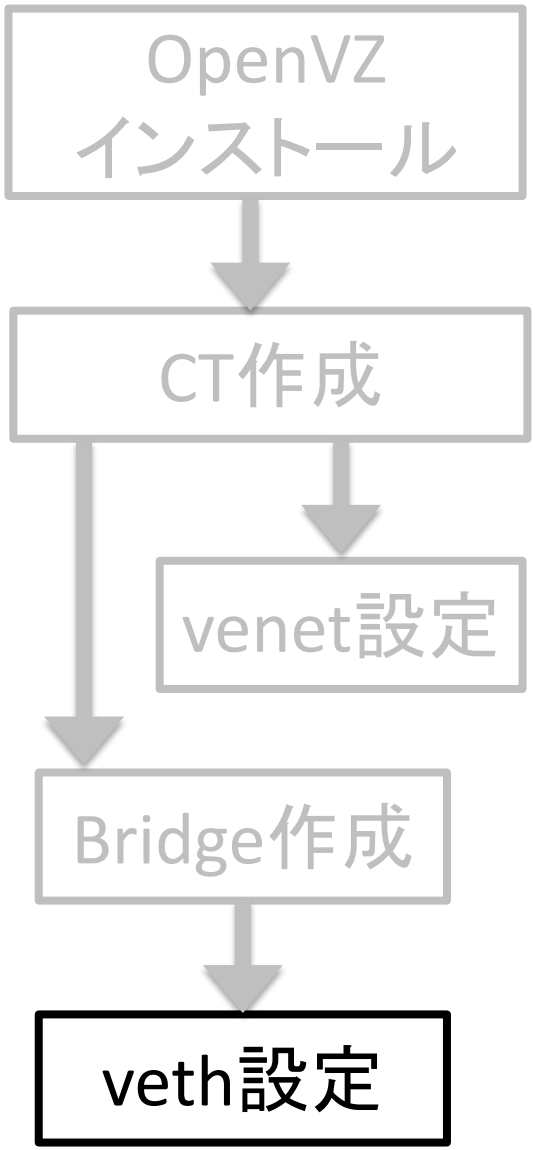
```
[host]# vzctl set 100 --netif_add eth0,,,,vzbr0 --save
Configure veth devices: veth100.0
CT configuration saved to /etc/vz/conf/100.conf
```

6. コンテナへ入り eth0（ホストのveth100.0）設定

```
[host]# vzctl enter 100
[CT]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="none"
NM_CONTROLLED="no"
ONBOOT="yes"
TYPE="Ethernet"
IPADDR=10.10.0.110
NETMASK=255.255.240.0
[CT]# service network restart
```

7. Bridge (vzbr0) へ veth (veth100.0) を追加

```
[host]# brctl addif vzbr0 veth100.0
```



ホストからの見え方


```
[host]# brctl show
bridge name     bridge id               STP enabled     interfaces
vzbr0           8000.0018511d38b4       no              eth0
                                                        veth100.0
[host]# ifconfig
... snip ...
veth100.0 Link encap:Ethernet  HWaddr 00:18:51:1D:38:B4
          inet6 addr: fe80::218:51ff:fe1d:38b4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:33 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51 errors:0 dropped:5 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2148 (2.0 KiB)  TX bytes:3316 (3.2 KiB)

vzbr0     Link encap:Ethernet  HWaddr 00:18:51:1D:38:B4
          inet addr:10.10.0.10  Bcast:10.10.0.255  Mask:255.255.255.0
          inet6 addr: fe80::218:51ff:fe1d:38b4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:248 errors:0 dropped:0 overruns:0 frame:0
          TX packets:37 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:12624 (12.3 KiB)  TX bytes:3046 (2.9 KiB)
```

コンテナからの見え方

OpenVZ, veth/venet 設定方法

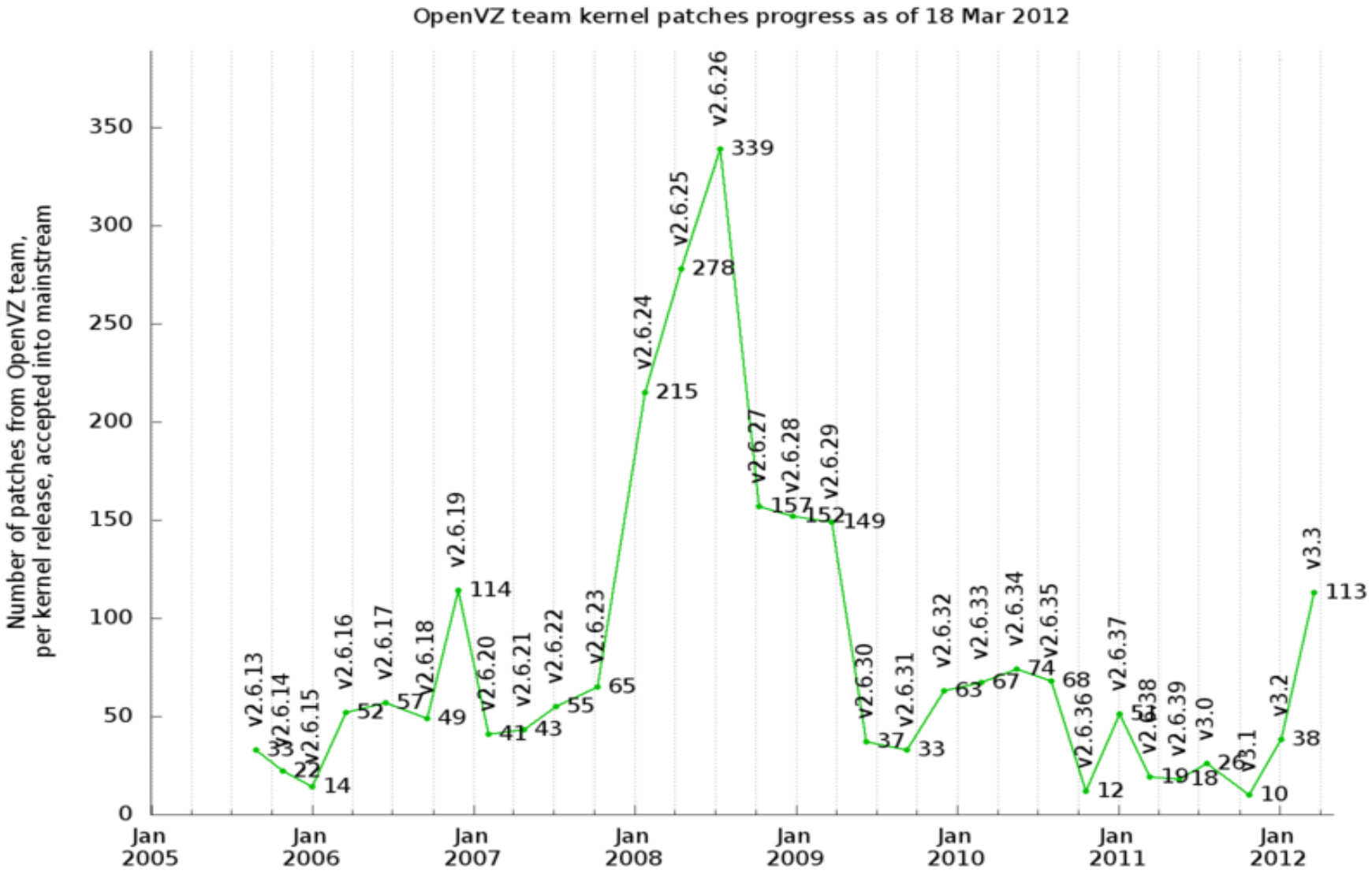
```
[CT]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:18:51:C2:FD:7B
          inet addr:10.10.0.110  Bcast:10.10.0.255  Mask:255.255.255.0
          inet6 addr: fe80::218:51ff:fec2:fd7b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:55 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3524 (3.4 KiB)  TX bytes:2148 (2.0 KiB)
... snip ...

venet0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
       inet addr:127.0.0.1  P-t-P:127.0.0.1  Bcast:0.0.0.0 Mask:255.255.255.255
          UP BROADCAST POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:168 (168.0 b)  TX bytes:5064 (4.9 KiB)

venet0:0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
         inet addr:10.10.0.100 P-t-P:10.10.0.100 Bcast:10.10.0.255  Mask:255.255.25
          UP BROADCAST POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
```

APPENDIX

Kernel updates by OpenVZ Team



http://openvz.livejournal.com/22369.html

仮想化プラットフォームによる性能比較 @ OpenVZ Wiki

http://wiki.openvz.org/Performance

Benchmark	Description
Response Time	Microbenchmark demonstrating latency issues of interactive applications in virtualized and loaded systems (netperf RR in various conditions).
Network Throughput	10Gbit simple network throughput comparison using netperf test.
LAMP	Linux Apache+MySql+PHP (LAMP) stack benchmark in multiple simultaneously running virtualization instances.
vConsolidate-UP	UP configuration of Intel vConsolidate server consolidation benchmark (Java+Apache+MySQL workloads).
vConsolidate-SMP	SMP configuration of Intel vConsolidate server consolidation benchmark (Java+Apache+MySQL workloads).
Microbenchmarks	Various microbenchmarks like context switch, system call, etc. Plus Unixbench resuls.

参考資料一覧

- OpenVZ Wiki - http://wiki.openvz.org/
- CRIU - http://www.criu.org/
- Linux Kernel Newbies / Kernel Changes - http://kernelnewbies.org/LinuxChanges

- O'Reilly, Linux Device Drivers (3rd Edition)
- O'Reilly, Understanding Linux Network Internals
- O'Reilly, Understanding the Linux Kernel (3rd Edition)

- "Adding Generic Process Containers to the Linux Kernel", Paul B. Menage @ Google / Balbir Singh and Srivatsa Vaddagiri @ IBM

- 00