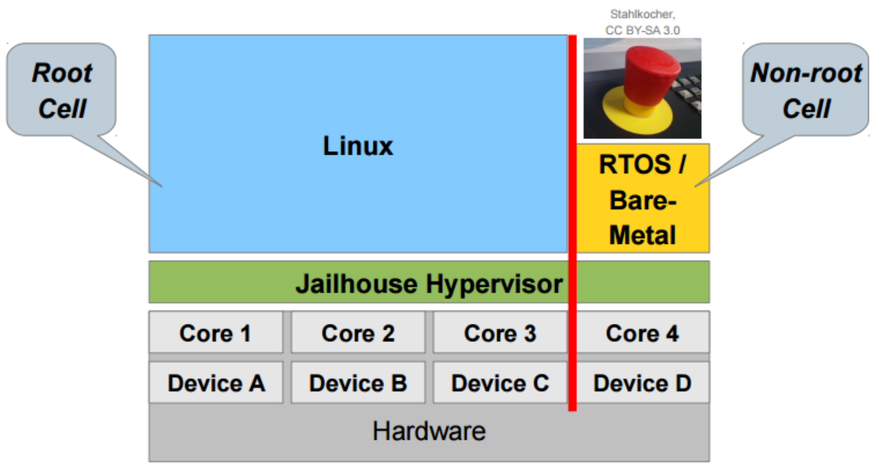
Reconnecting...

Jailhouse hypervisor for x86_64

Jailhouse Hypervisor

Jailhouse was born in Siemens and is developed as Free Software project (GPLv2) since November 2013.
On May 2015, Jailhouse 0.5 was released to general public.

- 2015 / Hard Partitioning for Linux The Jailhouse Hypervisor / siemens [PDF]



A tool to run
... real-time and/or safety tasks ... on multicore platforms (AMP)
... aside Linux

It provides
• strong & clean isolation
• bare-metal-like performance & latencies
• no reason to modify Linux (well, almost)

linux早有一堆virtualization：KVM, VirtualBox, Xen, VMware, lguest, hobbyst Xvisor
而Jailhouse的特點是lightweight，safety
lightweight (for real-time camp),small and simple (for security

folks) open-source Linux-friendly hypervisor for real-time an
certifiable workloads. (this being said, safety, not security is
primary focus at this point).

Reconnecting...

Jailhouse is static partitioning hypervisor，可以run bare-metal 但是要跟linux一起合作。

用 QEMU 驗證 Jailhouse x86_64 + VMX

- 詳細說明請見 jailhouse 原始程式碼內建文件: Documentation/articles/LJ-article-04-2015.txt

- Host 端

下載 Debian installer：debian-stretch-DI-alpha4-amd64-netinst.iso
安裝說明：Debian QEMU image

qemu-img 指令工具可以把 Xen 或 KVM 所使用的多種檔案系統格式化（Guest 端的映像檔、額外的儲存裝置與網路儲存裝置）。這裡我們使用 qemu-img create 來建立 Debian guest 端的虛擬硬碟映像檔。注意，若使用 Desktop 版本要記得將容量調大一點，不然安裝到最後會失敗。

```
$ qemu-img create debian.img 2G
```

或者:

```
$ qemu-img create -f qcow2 debian.qcow2 2G
```

莊彥宣 *若是使用ubuntu14.04 LTS Desktop version, 安裝最少需要6.8G，所以在加上額外的程式碼以及其他哩哩扣扣的東西，至少8G比較保險*

Intel CPU:

確認 kvm-intel 核心模組正確載入，而且設定了 nested=1

```
$  sudo rmmod kvm_intel
$ sudo sh -c "echo 'options kvm_intel nested=y' >>
/etc/modprobe.d/dist.conf"
$ sudo modprobe kvm_intel nested=1
```

AMD CPU:

確認 kvm-amd 核心模組正確載入，而且設定了 nested=1

```
$ sudo rmmod kvm_amd
$ sudo sh -c "echo 'options kvm_amd nested=1' >>
/etc/modprobe.d/dist.conf"
$ sudo modprobe kvm_amd nested=1
```

檢查方式

```
$ cat /sys/module/kvm_intel/parameters/nested
```

預期會看到 `Y´╯

Reconnecting...

準備執行 QEMU 來安裝 Debian GNU/Linux

```
$ qemu-system-x86_64 -machine q35 \
    -m 1G -enable-kvm -smp 4 \
     -cpu kvm64,-kvm_pv_eoi,-kvm_steal_time,-kvm_asyncpf,-kv
mclock,+vmx,+x2apic \
     -drive file=debian.img,id=disk,if=none \
     -device ide-hd,drive=disk -serial stdio -serial vc \
     -device intel-hda,addr=1b.0 -device hda-duplex \
     -cdrom debian-stretch-DI-alpha4-amd64-netinst.iso -boot d
```

CHUNYEN... *不知道是不是只有我遇到，我用Mint 17.1 在執行這個步驟的時候，qemu內會顯示No bootable device*
用virt-manager可以解決

楊鈞皓 *~~qemu-system-x86_64 -hda ./ubuntu.img -cdrom ubuntu-14.04.4-desktop-amd64.iso -m 4G -boot d~~*
~~我自己是這樣就可以了，不過上面那個問題我也有遇到，好像是qemu-system-x86_64再讀到ROM的時候沒有進去然後就iPXE什麼的一直下去，變成網路開機，然後就No bootable device。~~
~~不過我這樣寫好像就沒有用到kvm了耶~~

NEIL H *我用apt-get 的qemu 有遇到這樣的問題(把 -machine q35 拿掉可以執行)，後來直接從qemu git server clone source code下來安裝，就沒有遇到這樣的問題*

藍挺瑋 *只記得 q35 還算是新加入的東西，也許用新版本會比較好。*

一旦安裝完畢後，用以下指令重新啟動 Debian: (只有 cdrom 那行拿掉)

```
$ qemu-system-x86_64 -machine q35 \
    -m 1G -enable-kvm -smp 4 \
     -cpu kvm64,-kvm_pv_eoi,-kvm_steal_time,-kvm_asyncpf,-kv
mclock,+vmx,+x2apic \
     -drive file=debian.img,id=disk,if=none \
     -device ide-hd,drive=disk -serial stdio -serial vc \
     -device intel-hda,addr=1b.0 -device hda-duplex
```

CHUNYEN... *推薦沒有灌Desktop envir.的同學可以在qemu-system-x86_64 指令後面加上-redir參數，這樣就可以使用ssh進去guest的方式執行，最重要的好處是複製網址和指令很快。*

藍挺瑋 *看起來這指令是和 configs/qemu-vm.c 配合好的，只要稍*

微修改參數，就可能造成 /proc/iomem 內容改變，導 jailhouse enable 的時候當機。

Reconnecting...

TED J *安裝後, 會無法連上外部網路, 需修改 /etc/network/interfaces , 將 ifconfig -a 看到的dev name 與此檔案內的dev name 需改成一樣, 如下*
allow-hotplug enp0s2
iface enp0s2 inet dhcp
另外要 ssh 連線, 除了 qemu-system-x86_64 後加上 -redir tcp:2222::22 , debian image 也需裝上 apt-get install dropbear ,
host 端再下 ssh -p 2222 localhost 才能連線進入

- Debian Guest 端

  # **apt-get install build-essential**

  # **apt-get install linux-headers-4.2.0-1-amd64**

松錡 李 *似乎找不到4.2的headers package，可以參照這篇把kernel升上4.3，然後指令改為 apt-get install linux-headers-$(uname -r)*

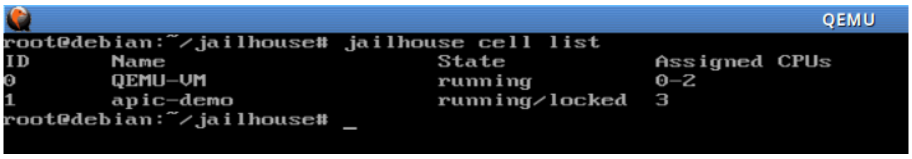
  # **apt-get install python-mako** # for `jailhouse config create`

  # **apt-get install vim**

編輯 /etc/default/grub
修改以下，並存檔:

    GRUB_CMDLINE_LINUX="memmap=66M\\\$0x3b000000"

之後執行 update-grub2 && reboot



```
root@debian:~/jailhouse# jailhouse cell list
ID      Name                State           Assigned CPUs
0       QEMU-VM             running         0-2
1       apic-demo           running/locked  3
root@debian:~/jailhouse# _
```

如果是使用 Ubuntu desktop（14.04），內建 make 是3.81 版本，編譯 jailhouse 需要更新到 3.82 以上的版本，不然無法會出現以下訊息。

    Makefile:16: *** Too old make version 3.81, at least 3.82 required. Stop.

下載 make 3.82+：http://ftp.gnu.org/gnu/make/

  $ **tar -zxvf make-4.1.tar.gz**

  $ **cd make-4.1**

  $ **./configure --prefix=/usr**

  # **make install**

重開 terminal

Reconnecting...

- **編譯 & 安裝:**

  $ **cd jailhouse**

  $ **make**

  $ **sudo make modules_install firmware_install**

楊鈞皓	*還要make install*
藍挺瑋	*不想 make install 應該可以 ./tools/jailhouse*
CHING L	*It seems that "make install" covers both module_install and firmware_install.* *Oh and these should be run with sudo.*

  $ **sudo depmod**

- **測試 cells:**

  $ **sudo -s**

  $ **modprobe jailhouse**

  $ **jailhouse enable configs/qemu-vm.cell** # 啟用jailhouse hyperviso r

CHIH-AN L	*我在這個步驟會顯示Input/Output: error ，請問也有其他人遇到這問題嘛？目前不知道怎麼解決*
志威 葉	*大概是Kernel版本(>3.18)？我不確定是不是因為這個*
CHIH-AN L	*kernel版本應該要大於3.18嘛？我的原本是3.13，現在正在編譯4.1的版本，等等編譯完成試試看。*
CHING L	*I'm using 4.3.0, same error.*
PENG L	*Same Problem here with kernel 4.3.0,，請問這步該怎麼解決Input/Output: error的問題?*
藍挺瑋	*這裡 QEMU 2.4.1 + Linux 4.4.4 可以跑，只要執行 QEMU 使用的參數和文件上完全相同 ......* *如果沒有當機的話，dmesg 可能可以看到一些訊息？*
CHING L	*Every time I try to enable jailhouse, dmesg shows:* *jailhouse: firmware: direct-loading firmware jailhouse-intel.bin* *And the terminal on my host shows:* *Initializing Jailhouse hypervisor v0.5 (211-g5298ecc) on CPU 1 (or some other number)* *They don't seem to be error messages, but the IO error still occurs.*
藍挺瑋	*serial console 上不知道有沒有印什麼東西？*
CHING L	*Which one?*
藍挺瑋	*應該只有一個？那個在 Linux 被叫做 /dev/ttyS0 的那個？*
CHING L	*Oh I get what you mean. It says:* *Initializing Jailhouse hypervisor v0.5 (211-g5298ecc) on CPU 1 (or some other number)*

Code location: 0xfffffffff0000030
Using x2APIC
And it stops here.
It seems like my CPU is missing unrestricted guest mode support. Farewell, hw2.

Reconnecting...

CHUNYEN... *成功Enable後，Qemu會卡住*

CHIH-AN L *there is no more "I/O error" after i upgrade my kernel to 4.2*

CLIFF T *qemu2.4.1+Linux4.4.4, same error....*

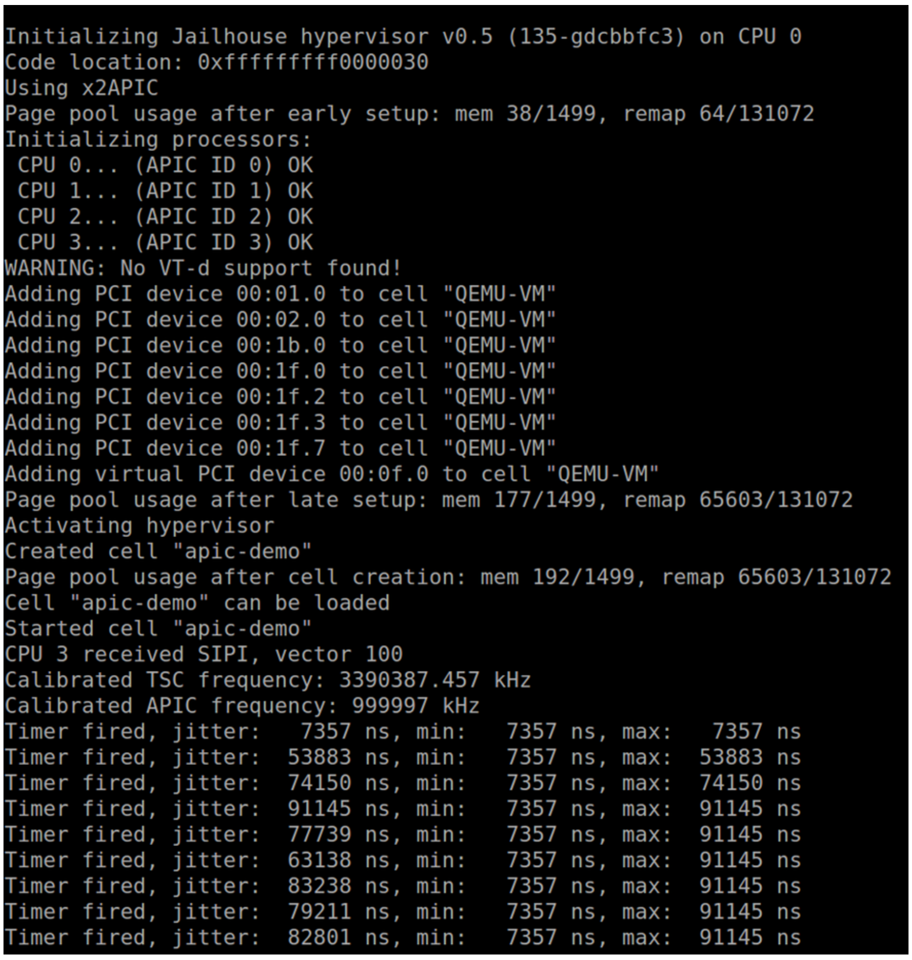
志威 葉 *已經確認是因為CPU硬體不支援x2APIC才會導致Input/Output Error問題，無解，只能升級硬體CPU。 可以用 `grep "x2apic" /proc/cpuinfo` 來確認是否有此flag。*

CHING-HU... *我的有支援x2APIC一樣會Error欸，kernel是4.2，vt-d和vmx也有*

TED J *Host site kernel version also need 4.x otherwise it will fail*

```
$ jailhouse cell create configs/apic-demo.cell
$ jailhouse cell load apic-demo inmates/demos/x86/apic-demo.bin -a 0xf0000
$ jailhouse cell start apic-demo
```

預期輸出


```
Initializing Jailhouse hypervisor v0.5 (135-gdcbbfc3) on CPU 0
Code location: 0xfffffffff0000030
Using x2APIC
Page pool usage after early setup: mem 38/1499, remap 64/131072
Initializing processors:
 CPU 0... (APIC ID 0) OK
 CPU 1... (APIC ID 1) OK
 CPU 2... (APIC ID 2) OK
 CPU 3... (APIC ID 3) OK
WARNING: No VT-d support found!
Adding PCI device 00:01.0 to cell "QEMU-VM"
Adding PCI device 00:02.0 to cell "QEMU-VM"
Adding PCI device 00:1b.0 to cell "QEMU-VM"
Adding PCI device 00:1f.0 to cell "QEMU-VM"
Adding PCI device 00:1f.2 to cell "QEMU-VM"
Adding PCI device 00:1f.3 to cell "QEMU-VM"
Adding PCI device 00:1f.7 to cell "QEMU-VM"
Adding virtual PCI device 00:0f.0 to cell "QEMU-VM"
Page pool usage after late setup: mem 177/1499, remap 65603/131072
Activating hypervisor
Created cell "apic-demo"
Page pool usage after cell creation: mem 192/1499, remap 65603/131072
Cell "apic-demo" can be loaded
Started cell "apic-demo"
CPU 3 received SIPI, vector 100
Calibrated TSC frequency: 3390387.457 kHz
Calibrated APIC frequency: 999997 kHz
Timer fired, jitter:   7357 ns, min:   7357 ns, max:   7357 ns
Timer fired, jitter:  53883 ns, min:   7357 ns, max:  53883 ns
Timer fired, jitter:  74150 ns, min:   7357 ns, max:  74150 ns
Timer fired, jitter:  91145 ns, min:   7357 ns, max:  91145 ns
Timer fired, jitter:  77739 ns, min:   7357 ns, max:  91145 ns
Timer fired, jitter:  63138 ns, min:   7357 ns, max:  91145 ns
Timer fired, jitter:  83238 ns, min:   7357 ns, max:  91145 ns
Timer fired, jitter:  79211 ns, min:   7357 ns, max:  91145 ns
Timer fired, jitter:  82801 ns, min:   7357 ns, max:  91145 ns
```

Reconnecting...

接著要關閉cell，可用以下操作

```
$ jailhouse cell shutdown apic-demo
$ jailhouse cell shutdown apic-demo
```

由於apic-demo在被載入的時候，是鎖定模式(locked)，所以shutdown要執行兩次才能關閉

```
$ jailhouse cell destroy apic-demo
$ jailhouse disable # 關閉 jailhouse hypervisor
```

- **測試 inter-cell communication**

Jailhouse 的 inter-cell communication 實現方法是利用 ivshmem (inter-vm shared memory)，是利用共享記憶體的方法來完成，首先先來談談 ivshmem 機制

IVSHMEM

- 虛擬機 (VM) 之間共享 PCI device
- 支援三個 PCI BAR (Base Address Register)
  - BAR0 -> 1 k byte MMIO region
  - BAR1 -> MSI-X
  - BAR2 -> Mapping memory from host
- IVSHMEM Registers
  - interrupt mask
  - interrupt status
  - interrupt vector position (Positive integer for guest ID)
  - doorbell (knock knock! send interrupt!) -> Generally support 256 interrupts, but jailhouse only support 1 interrupt

到這裡我們知道，要啟用 ivshmem ，必須要

1. 與 host 共用一段記憶體
2. 共用一個虛擬 PCI 裝置

那麼在 Jailhouse 要如何實現? 首先

- 與 host 共用一段記憶體，必須先在 cell config 的時候加入一段記憶體區間，如下

```
.mem_regions = {
  {
    /* mem region 1 */ ...
  },
  {
    /* mem region 2 */ ...
  },
```

Reconnecting...

```
    {  /* ivshmem region, region 3 */
        .phys_start = 0x3f1ff000,
        .virt_start = 0x3f1ff000,
        .size = 0x1000,
        .flags = JAILHOUSE_MEM_READ | JAILHOUSE_MEM_WRITE
|
            JAILHOUSE_MEM_ROOTSHARED,
    }
}
```

要注意的是，必須要指定此段記憶體是與 host 共享的，加入 `JAILHOUSE_MEM_ROOTSHARED` 來標記此段記憶體，並且兩個 cell 共享記憶體的起始位置與大小要相同

- 若要共用虛擬 PCI ，要在 cell config 裡面加上虛擬 PCI ，如下

```
.pci_devices = {
    .type = JAILHOUSE_PCI_TYPE_IVSHMEM,
    .domain = 0x0,
    .bdf = (0x0f<<2),
    .bar_mask = {
        0xffffff00, 0xffffffff, 0x00000000,
        0x00000000, 0xffffffe0, 0xffffffff,
    }
    .shmem_region = 2,
    /* not always 2, can be different due to different memory layout */
    .num_msix_vectors = 1,
}
```

要注意的是，兩個 PCI 之間的 B/D/F 必須要相同，並且 type 指定 `JAILHOUSE_PCI_TYPE_IVSHMEM`，還要告知 host 分享的記憶體是哪一塊，寫在 `shmem_region` 裡面，而數值是該記憶體的 index，例如假設我之前所設定的 ivshmem memory 是在第三塊，則我在此處的 index = 2，如此設定就可以完成 cell config，接下來就是如何達到互相傳送 interrupt 的使用方法

要能夠互相傳送 interrupt ，只要完成以下步驟

- 當然是先 init，呼叫 `int_init();`
- 建立 pci ， `bdf = pci_find_device(VENDORID, DEVICEID, bdf);`
- map memory， `map_shmem_and_bars(d);`
  - 細節請看 ivshmem-demo
- 接下來對 `doorbell` 暫存器寫入數值 `mmio_write32(d->registers + 3,`

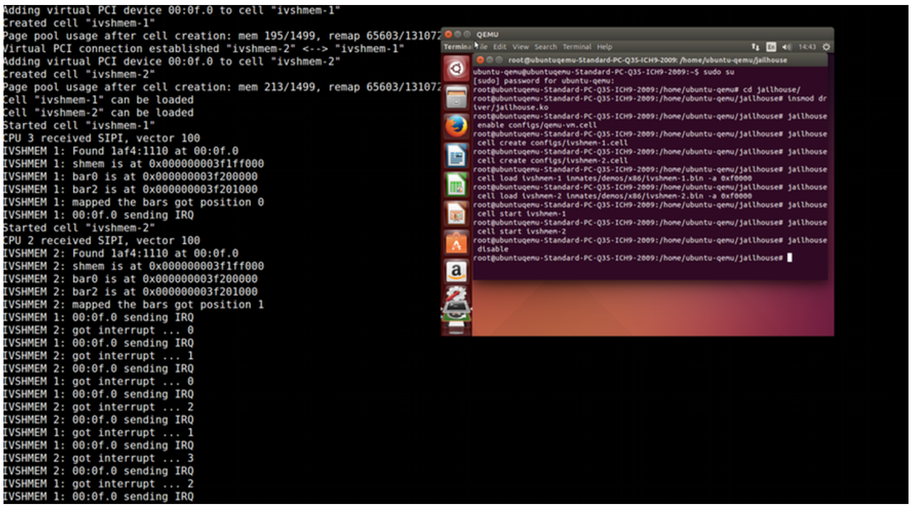
Reconnecting...

`1);`

  - `d->registers + 3` 因為是第四個暫存器，
  - `1` 是因為目前只支援一個中斷，只能寫 1 進去

如此一來就能送中斷給另外一個搭配的 cell 了!
接收中斷很簡單，只要注冊好 `irq_handler()` 即可!
ex. `int_set_handler(IRQ_VECTOR + ndevices - 1, irq_handler);`

/* 最後，最重要的是兩個 cell 必須要用不同的資源 (ex. cpu, memory region)，所以在 cell config 裡面要改變記憶體以及CPU的設定，兩個 cell 要用不同的記憶體區段以及佔用不同的CPU才可以載入 */



```
Adding virtual PCI device 00:0f.0 to cell "ivshmem-1"
Created cell "ivshmem-1"
Page pool usage after cell creation: mem 195/1499, remap 65603/131072
Virtual PCI connection established "ivshmem-2" <--> "ivshmem-1"
Adding virtual PCI device 00:0f.0 to cell "ivshmem-2"
Created cell "ivshmem-2"
Page pool usage after cell creation: mem 213/1499, remap 65603/131072
Cell "ivshmem-1" can be loaded
Cell "ivshmem-2" can be loaded
Started cell "ivshmem-1"
CPU 3 received SIPI, vector 100
IVSHMEM 1: Found 1af4:1110 at 00:0f.0
IVSHMEM 1: shmem is at 0x000000003f1ff000
IVSHMEM 1: bar0 is at 0x000000003f200000
IVSHMEM 1: bar2 is at 0x000000003f201000
IVSHMEM 1: mapped the bars got position 0
IVSHMEM 1: 00:0f.0 sending IRQ
Started cell "ivshmem-2"
CPU 2 received SIPI, vector 100
IVSHMEM 2: Found 1af4:1110 at 00:0f.0
IVSHMEM 2: shmem is at 0x000000003f1ff000
IVSHMEM 2: bar0 is at 0x000000003f200000
IVSHMEM 2: bar2 is at 0x000000003f201000
IVSHMEM 2: mapped the bars got position 1
IVSHMEM 1: 00:0f.0 sending IRQ
IVSHMEM 2: got interrupt ... 0
IVSHMEM 1: 00:0f.0 sending IRQ
IVSHMEM 2: got interrupt ... 1
IVSHMEM 2: 00:0f.0 sending IRQ
IVSHMEM 1: got interrupt ... 0
IVSHMEM 1: 00:0f.0 sending IRQ
IVSHMEM 2: got interrupt ... 2
IVSHMEM 2: 00:0f.0 sending IRQ
IVSHMEM 1: got interrupt ... 1
IVSHMEM 1: 00:0f.0 sending IRQ
IVSHMEM 2: got interrupt ... 3
IVSHMEM 2: 00:0f.0 sending IRQ
IVSHMEM 1: got interrupt ... 2
IVSHMEM 1: 00:0f.0 sending IRQ
```

```
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009: /home/ubuntu-qemu/jailhouse
ubuntu-qemu@ubuntuqemu-Standard-PC-Q35-ICH9-2009:~$ sudo su
[sudo] password for ubuntu-qemu:
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu# cd jailhouse/
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# insmod dr
iver/jailhouse.ko
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 enable configs/qemu-vm.cell
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 cell create configs/ivshmem-1.cell
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 cell create configs/ivshmem-2.cell
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 cell load ivshmem-1 inmates/demos/x86/ivshmem-1.bin -a 0xf0000
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 cell load ivshmem-2 inmates/demos/x86/ivshmem-2.bin -a 0xf0000
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 cell start ivshmem-1
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 cell start ivshmem-2
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# jailhouse
 disable
root@ubuntuqemu-Standard-PC-Q35-ICH9-2009:/home/ubuntu-qemu/jailhouse# 
```

未來展望：https://github.com/hw-claudio/virtio-peer/wiki

FreeRTOS for Jailhouse Cells
FreeRTOS-cell 專案目標是在 Jailhouse Hypervisor 上同時執行 General Purpose Linux 以及 hard real time 的 FreeRTOS ，作為一個強調低延遲的 Hypervisor，或許因此可以有更多的應用空間。
$ git clone https://github.com/siemens/freertos-cell.git

核心程式碼：Source

- tasks.c：主要掌管 task 的檔案
- queue.c：管理 task 間 communication (message queue 的概念)
- list.c：提供系統與應用實作會用到的 list 資料結構

- 與硬體相關的檔案在：Source/portable/GCC/ARM_A7jailhouse/

freertos-demo

```
$ insmod driver/jailhouse.ko
$ jailhouse enable configs/bananapi.cell
    Initializing Jailhouse hypervisor v0.5 (150-g3ac0aa5) on CPU 1
    Code location: 0xf0000020
    Page pool usage after early setup: mem 17/16368, remap 32/32768
    Initializing processors:
     CPU 1... OK
     CPU 0... OK
    Page pool usage after late setup: mem 23/16368, remap 32/32768
    Activating hypervisor
$ jailhouse cell create configs/bananapi-freertos-demo.cell
    Created cell "FreeRTOS"
    Page pool usage after cell creation: mem 31/16368, remap 32/32768
$ jailhouse cell load 1 ../freertos-cell/freertos-demo.bin
    Cell "FreeRTOS" can be loaded
$ jailhouse cell start FreeRTOS
    ===== MMU/Cache status at entry =====
            Icache 0
            Flow   1
            Dcache 0
            MMU    0
    Initializing the HW...gicc_base=1c82000 gicd_base=1c81000
    MMU page table: 0x00008000
    hardware_mmu_ptable_setup: [0]=0x1c00000
    hardware_mmu_ptable_setup: [1]=0x1c00000
    UART gicd=0x01c81800 CPUID=2
            Orig GICD_ITARGETSR[52]=2
            New  GICD_ITARGETSR[52]=2
    IRQ52 prio original: 0xa0
    IRQ52 prio readback after 0xff: 0xf0
    IRQ52 prio modified: 0xe0
    FreeRTOS inmate cpu-mode=13
    ===== MMU/Cache status at runtime =====
            Icache 1
            Flow   1
            Dcache 1
            MMU    1
    Create task 0 with prio 1
```

Reconnecting...

```
Create task 1 with prio 2
Create task 2 with prio 3
Create task 3 with prio 4
Create task 4 with prio 5
Create task 5 with prio 6
Create task 6 with prio 1
Create task 7 with prio 2
Create task 8 with prio 3
Create task 9 with prio 4
Create task 10 with prio 5
Create task 11 with prio 6
Create task 12 with prio 1
Create task 13 with prio 2
Create task 14 with prio 3
Create task 15 with prio 4
Create task 16 with prio 5
Create task 17 with prio 6
Create task 18 with prio 1
Create task 19 with prio 2
```

vTaskStartScheduler goes active

```
IRQ27 prio original: 0xa0
IRQ27 prio readback after 0xff: 0xf0
IRQ27 prio modified: 0xe0
T06     period:  600;   loop:    0;     tick:     0
T12     period: 1200;   loop:    0;     tick:     1
T18     period: 1800;   loop:    0;     tick:     1
Sending ...T05  period:  500;   loop:    0;     tick:    12
T11     period: 1100;   loop:    0;     tick:    13
T17     period: 1700;   loop:    0;     tick:    13
Value received: 1
T04     period:  400;   loop:    0;     tick:    26
T10     period: 1000;   loop:    0;     tick:    27
T16     period: 1600;   loop:    0;     tick:    27
T03     period:  300;   loop:    0;     tick:    37
T09     period:  900;   loop:    0;     tick:    38
T15     period: 1500;   loop:    0;     tick:    38
T02     period:  200;   loop:    0;     tick:    49
T08     period:  800;   loop:    0;     tick:    50
T14     period: 1400;   loop:    0;     tick:    50
T20     period: 2000;   loop:    0;     tick:    50
T07     period:  700;   loop:    0;     tick:    65
```

Reconnecting...

```
    T13     period: 1300;   loop:    0;     tick:    66
    T19     period: 1900;   loop:    0;     tick:    66
    FT0: 1.11^0=   1.000000
    FT1: 1.11^0=   1.000000
    T01     period:  100;   loop:    0;     tick:    64
    T01     period:  100;   loop:    1;     tick:   164
    T02     period:  200;   loop:    1;     tick:   249
    TUA     1
    T01     period:  100;   loop:    2;     tick:   264
    T03     period:  300;   loop:    1;     tick:   337
    T01     period:  100;   loop:    3;     tick:   364
    T04     period:  400;   loop:    1;     tick:   426
    T02     period:  200;   loop:    2;     tick:   449
    T01     period:  100;   loop:    4;     tick:   464
    T05     period:  500;   loop:    1;     tick:   512
    T01     period:  100;   loop:    5;     tick:   564
    T06     period:  600;   loop:    1;     tick:   600
    (以下略...)
$ jailhouse cell shutdown FreeRTOS
    Cell "FreeRTOS" can be loaded
$ jailhouse cell destroy FreeRTOS
    Closing cell "FreeRTOS"
    Page pool usage after cell destruction: mem 23/16368, remap 32/327
    68
```