1 VF 方式的 SDK 安装

1.1 HOST 测的方法

使用厂商提供的最新版本：DX_SDK_v2.3.0L_EXAR_20181223.tar.gz

DX_SDK_v2.3.0L_PUBLIC_20181223.tar.gz

1）安装 gcc 及 kernel-devel

yum install -y gcc kernel-devel

2）在源码目录执行编译命令编译模块：

make DRE_92XX_SUPPORT=1 DRE_SRIOV_PF_SDK=1

3）执行 sh Load 加载驱动

4）运行测试程序测试：

./sdemo sdemo.encode.cfg.xml

./sdemo sdemo.decode.cfg.xml

```
[root@pci-host dx_dev]# ./sdemo sdemo.encode.cfg.xml
DX SDK sdemo application - Built with SDK v2.3.0L
000: src 016426 => dst 003258, total 003258 bytes
Operation type: COMP
Direction: Encode
Comp algorithm: DEFLATE (OPTIMAL) stateless
Block size: 0
CRC Configuration: No CRC
Cmd Target Chip: Load Balance
Source file:       README.public (16426)
Destination file:  README.public.encode (3258)
Compression ratio: 5.04:1
sdemo passed
```

1.2 虚机测的方法

除了 2.1 流程的第二部按照如下方式，其他步骤相同

make DRE_92XX_SUPPORT=1 DRE_SRIOV_VF_SDK=1

2 安装 openssl 及 openssl engine：

安装 openssl 1.0.2i 版本

wget https://www.openssl.org/source/openssl-1.0.2i.tar.gz

1. 在对应版本源码目录执行

   ./config shared --prefix=/usr/local/openssl-1.0.2i/

   make clean

   make

   make test

   make install

2. cd 至厂商 eng_dx 目录，修改 Makefile 文件

   EXAR_DX_SDK_PATH := $(PWD)/../dx_sdk/（SDK build 目录）

   OPENSSL_INCLUDE_PATH := /usr/local/openssl-1.0.2i/include（openssl 安装目录）

   OPENSSL_LIB_PATH := /usr/local/openssl-1.0.2i/lib

   OPENSSL_DYN_ENGINE_PATH := $(OPENSSL_LIB_PATH)/engines

   执行编译安装：

   make clean

   make ~~HASH_ENABLE=1~~ ~~RNG_ENABLE=1~~（注按照厂商文档打开 ECC_ENABLE=1 时，会报错，怀疑是操作系统版本问题，7.5 不在其支持列表之内）

   当 VF 到虚机时：

   虚机编译 eng_dx 时，不需要把随机数 RNG 编译进去。

   make ~~HASH_ENABLE=1~~ ~~RNG_ENABLE=1~~

   执行下面的命令检查编译进去了哪些算法：

   /usr/local/ssl/bin/openssl engine eng_dx –c

   make install

3. 简单验证：

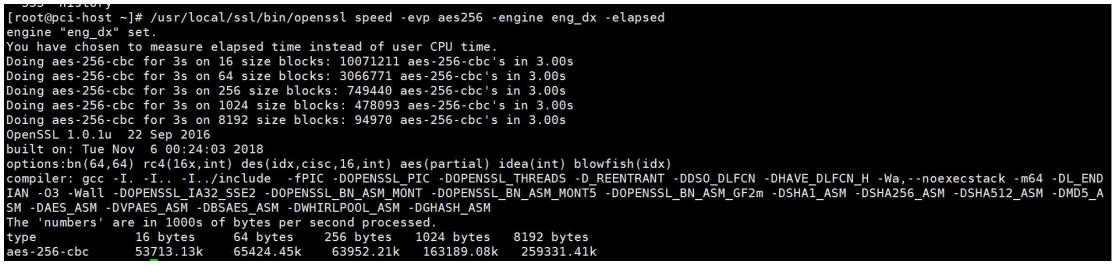
   /usr/local/openssl-1.0.2i/bin/openssl speed -evp aes256 -engine eng_dx -elapsed

   /usr/local/openssl-1.0.2i/bin/openssl speed -evp aes256 -engine eng_dx -elapsed -multi 20

/usr/local/openssl-1.0.2i/bin/openssl speed rsa2048 -engine eng_dx -elapsed

watch cat /proc/exar/dx_cmd_statistics

出现类似如图所示的输出：



3 编译安装 nginx（所选版本问厂商文档示例版本 nginx-1.7.3）：

wget http://ftp.pcre.org/pub/pcre/pcre-8.40.tar.gz

tar xzf pcre-8.40.tar.gz

5.2 get the nginx 1.15.5

wget http://nginx.org/download/nginx-1.15.5.tar.gz

tar xzf nginx-*.tar.gz

cd nginx-*

需要修改 Nginx 源代码 31-35 行编译配置 vim auto/lib/openssl/conf：

CORE_INCS="$CORE_INCS $OPENSSL/include"

CORE_DEPS="$CORE_DEPS $OPENSSL/include/openssl/ssl.h"

CORE_LIBS="$CORE_LIBS $OPENSSL/lib/libssl.a"

CORE_LIBS="$CORE_LIBS $OPENSSL/lib/libcrypto.a"

CORE_LIBS="$CORE_LIBS $NGX_LIBDL"

./configure --prefix=/usr/local/nginx-1.15.5 --with-http_ssl_module --with-pcre=../pcre-8.40

--with-ld-opt=-L/usr/local/openssl-1.0.2i/lib

--with-cc-opt=-I/usr/local/openssl-1.0.2i/include/

tengine 编译：

./configure　--with-http_ssl_module

--add-module=/opt/gnosek-nginx-upstream-fair-a18b409

--with-cc-opt=-I/usr/local/openssl-1.0.2i/include --with-ld-opt=-L/usr/local/openssl-1.0.2i/lib

--with-http_v2_module

make

make install

完成安装后可直接替换 nginx 配置文件/usr/local/nginx/conf/nginx.conf，在厂商提供的文件夹中，或参考 APN-0021-A01_Nginx_OpenSSL_Engine_2.0.0_AppNote.pdf 2.3.1 章节修改

3.1 启动 nginx：：

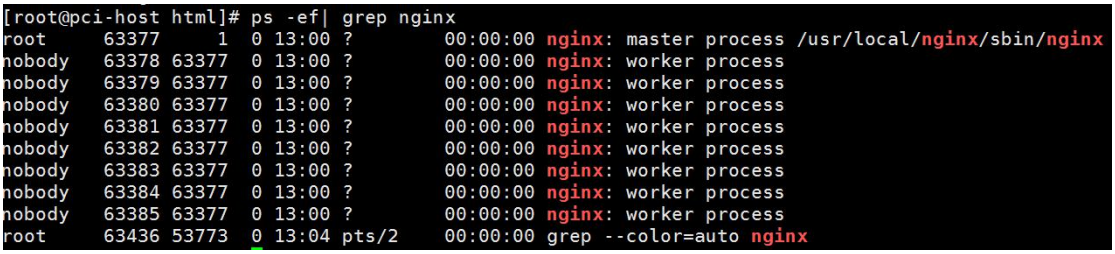
1. 在 nginx 配置目录/usr/local/nginx/conf，生成证书：

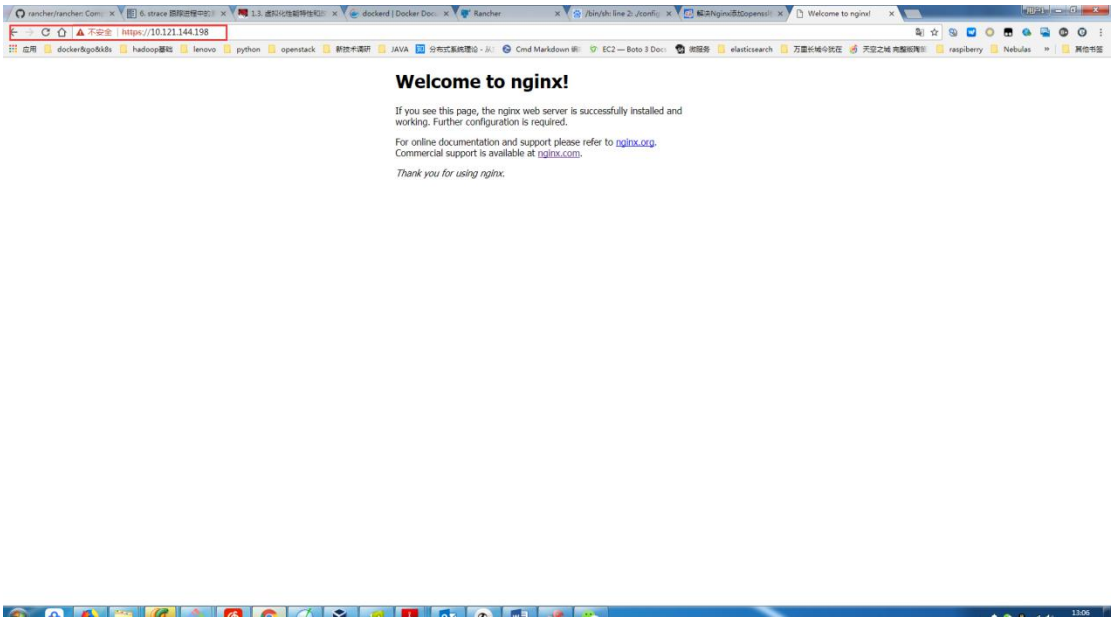
   /usr/local/openssl-1.0.2i/bin/openssl　req -x509 -nodes -days 365 -newkey rsa:1024

   -keyout cert.key -out cert.pem -subj '/C=XX/ST=XX/L=XX/CN=www.xxxx.com'
2. 在/usr/local/nginx/html 目录生成测试文件

   touch zero.html
3. 导入 openssl 环境变量

   export LD_LIBRARY_PATH=/usr/local/openssl-1.0.2i/lib/
4. 启动 nginx，/usr/local/nginx/sbin/nginx





4 宿主机上的配置

1. 配置计算节点 grub 参数

   vim /etc/default/grub（增加红色 intel_iommu=on）

   GRUB_CMDLINE_LINUX="console=ttyS0,9600 console=tty0 rootdelay=90 nomodeset

   crashkernel=auto rd.lvm.lv=os/root rd.md.uuid=acc67230:55b2fc46:bcfbfdf7:68a173d9

   rd.lvm.lv=os/swap biosdevname=0 rhgb quiet intel_iommu=on"

   执行以下命令更新 grub 参数：

   grub2-mkconfig -o /boot/grub2/grub.cfg

2. config nova-compute

   1）在卸载卡所在节点的 nova 配置文件 DEFAULT section 添加

   pci_passthrough_whitelist={"vendor_id": "13a3","product_id": "9240"}

   2）在最后一行加入

   [pci]

   alias = {"vendor_id": "13a3","product_id": "9200", "device_type":"type-VF", "name":"a1" }

   重启 nova-compute 服务

3. config nova-api

   在控制节点 nova 配置文件 DEFAULT section 添加

   pci_alias = {"vendor_id": "13a3","product_id": "9240", "device_type":"type-PF",

   "name":"a1" }

在 controller 节点 nova.conf 配置文件中，找到 scheduler_default_filters 字段，最后面添加 PciPassthroughFilter

重启 nova-api 服务

4. 选择一个合适的 flavor 并修改其 property

   openstack flavor set <flavor-name> --property "pci_passthrough:alias"="a1:2"

5. 以此 flavor 创建虚拟机（本实验中使用 CentOS7.5 镜像）
6. 创建完成后在虚拟机中执行 lspci -nnn | grep -v Intel 可见该设备

```
[root@pci-host ~]# lspci -nnn | grep -v Intel
00:02.0 VGA compatible controller [0300]: Cirrus Logic GD 5446 [1013:00b8]
00:03.0 Ethernet controller [0200]: Red Hat, Inc. Virtio network device [1af4:1000]
00:04.0 SCSI storage controller [0100]: Red Hat, Inc. Virtio block device [1af4:1001]
00:05.0 Processing accelerators [1200]: Hifn Inc. Device [13a3:9200]
00:06.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon [1af4:1002]
```

5 注意事项

1. Host 与 Vm 编译参数不同

   Host： make DRE_92XX_SUPPORT=1 DRE_SRIOV_PF_SDK=1

   VM： make DRE_92XX_SUPPORT=1 DRE_SRIOV_VF_SDK=1

2. 导入 openssl 环境变量

   export LD_LIBRARY_PATH=/usr/local/ssl/lib/

3. 编译 openssl engine 时，注意去掉 RAND 算法的选项 make ~~HASH_ENABLE=1~~ ~~RNG_ENABLE=1~~

   执行下面的命令检查编译进去了哪些算法：

   /usr/local/ssl/bin/openssl engine eng_dx –c

4. Host nova 相关配置：

   https://docs.openstack.org/nova/pike/admin/pci-passthrough.html

   https://wiki.openstack.org/wiki/Nova/pci_hotplug

   https://www.jianshu.com/p/95a0a407fceb?tdsourcetag=s_pctim_aiomsg

5. 更改了启动脚本 Load Unload

   Load 加入：

   rm -rf /dev/pk_drv || exit 1

   mknod /dev/pk_drv c `cat /proc/devices | grep pk_drv | awk '{print $1}'` 0 || exit 1

   chmod 777 /dev/pk_drv || exit 1