PRESENTED BY MASAFUMI OHTA @masafumiohta

GPU ON OPENSTACK - GPUインターナルクラウドのベストプラクティス

openstack®

CLOUD SOFTWARE

MASAFUMI OHTA

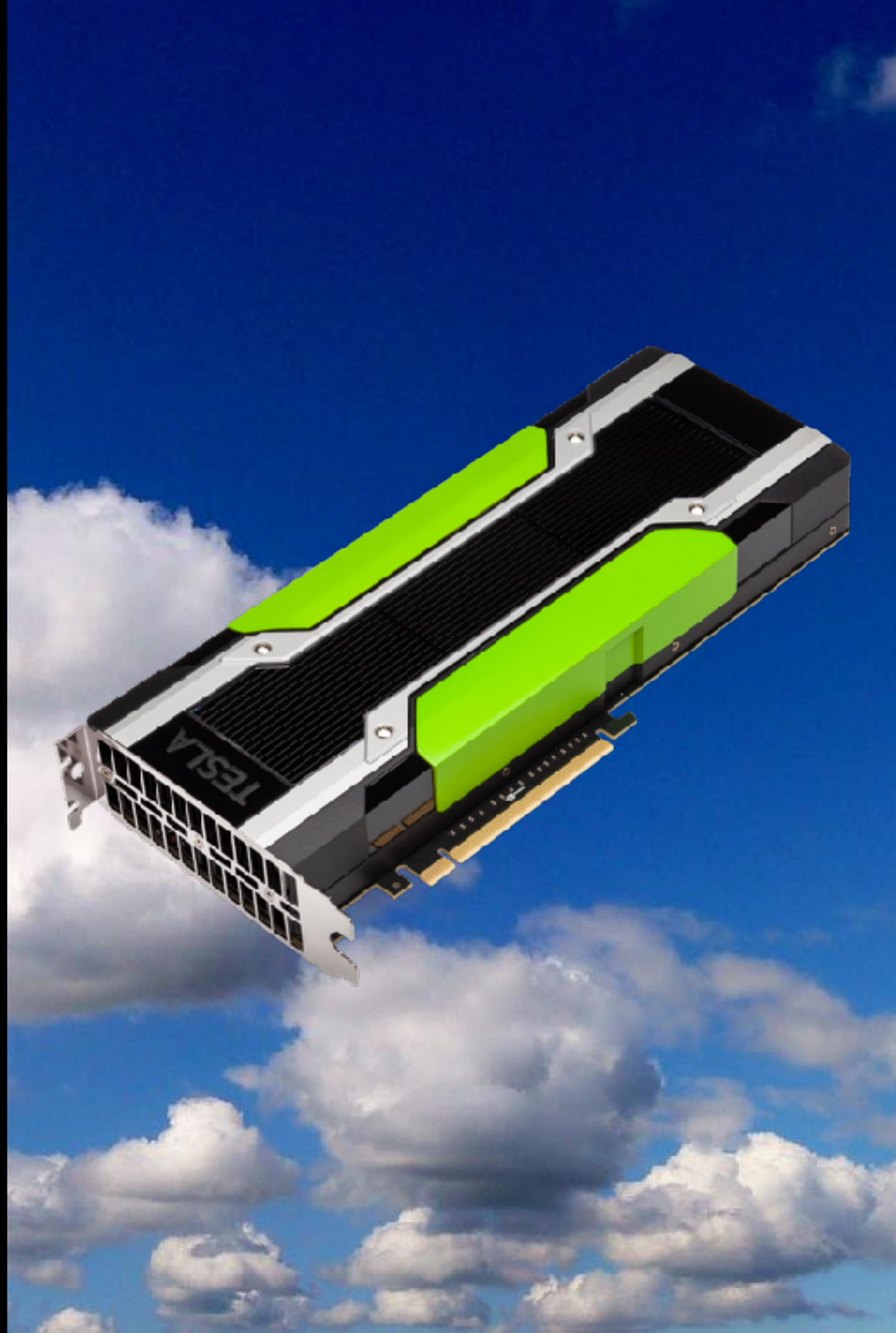
A ‘STACKER’ LOOKING INTO GPGPU USE. VOLUNTEER FOR THE RASPBERRY PI FOUNDATION :)

PRESENTATION RENEWED

THIS PRESENTATION IS RENEWED FOR OPENSTACK DAYS TOKYO 2017.
IT IS INCLUDED SOME FEEDBACKS FROM THE SESSION AT #LC3 CHINA IN BEIJING,2017

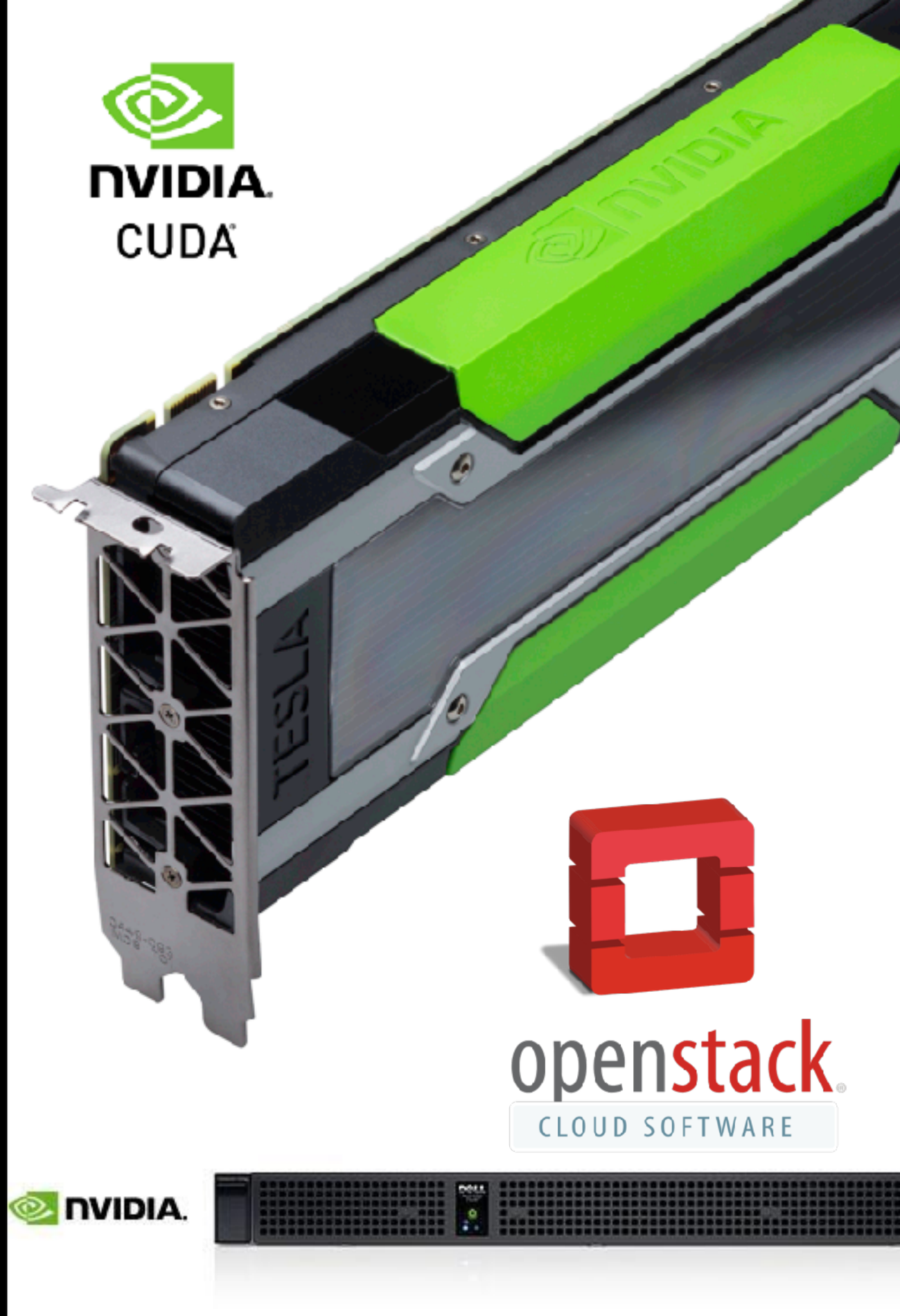
THIS PROJECT IS...

ORGIZNED BY VTJ AND GATHER
ME,NVIDIA(ASK),DELL EMC, NESIC AND SOME COMPANIES WHO INTERESTED GPU USE ON OPENSTACK ENVIRONMENT FOR OUR CUSTOMERS



NOW EVALUATING..

TESLA M60+DELLEMC POWEREDGE C4130+OPENSTACK TO EVALUATE GPU ON OPENSTACK ENVIRONMENT THANKS TWO HELPING OUT AND PROVIDING THOSE SERVERS+CARDS



日本仮想化技術、OpenStack環境でのGPU技術の活用「GPU on OpenStack」を推進

2017 年 7 月 18 日　ntamaoki

報道関係者各位
プレスリリース
2017年7月18日
日本仮想化技術株式会社

オープンソースのクラウド技術である"OpenStack"の導入支援やコンサルティングを手掛ける日本仮想化技術株式会社（本社：東京都渋谷区、代表取締役社長：宮原 徹）は、OpenStack環境でのGPU技術の活用「GPU on OpenStack」を推進いたします。

「GPU on OpenStack」は、OpenStack環境の仮想マシン／コンテナに対してGPUの計算リソースの割り当てを行い、ハイパフォーマンスコンピューティングやデスクトップ仮想化などの様々な用途で利用できるインフラ技術です。
「GPU on OpenStack」を使用したOpenStack環境を用意することで、GPUを業務で活用する利用者に対して利便性の向上を実現し、インフラ管理者に対して一元管理による運用の効率化を支援いたします。「GPU on OpenStack」の環境構築や利用方法についての情報は十分でなく、日本仮想化技術ではエヌビディア合同会社（以下、NVIDIA）と株式会社アスク（以下、アスク）の協力により環境構築と技術評価を行い、利用者やインフラ管理者にとって役立つ情報を発信していきます。日本仮想化技術およびこの取り組みに賛同いただける企業と共に「GPU on OpenStack」の導入支援やコンサルティングサービスを推進していきます。

国内最大規模のオープンクラウドイベントである「OpenStack Days Tokyo」（７月２０日～７月２１日、虎ノ門ヒルズフォーラム）ではNVIDIA／アスクの展示ブースでデモンストレーションを用意しております。弊社主催イベントである「OpenStack最新情報セミナー」（７月２６日、渋谷）ではOpenStack環境でのGPU技術の活用をテーマとした技術セミナーを企画いたします。

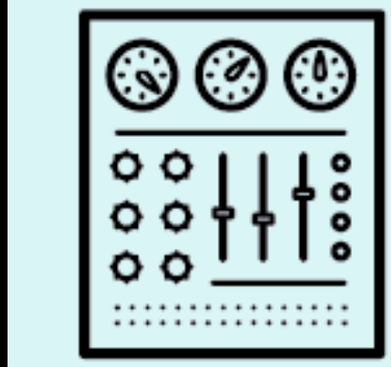
OpenStack最新情報セミナー
https://openstack-update.connpass.com

はじめに

なぜ GPU ON OPENSTACKなのか？

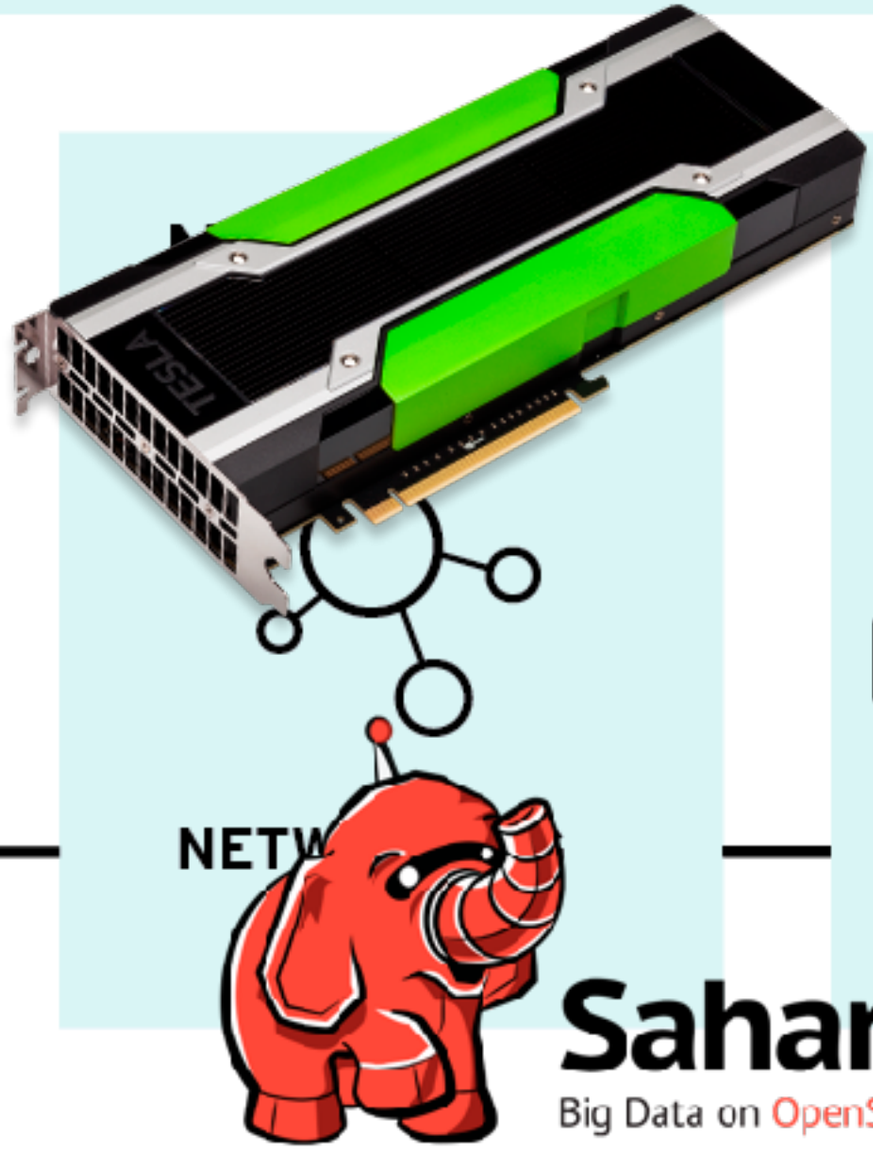
OPENSTACK 特殊利用の需要

- OpenStackの特殊な利用の需要が増えてきている
  - Hadoop(Sahara),HPC,などなど
- ほとんどのものはまとめたのがなく、ググって調べるしかない...
  - 曰く「ドキュメントロスト状態」
- これらのものはOpenStackのDocsにまとめられるべき



HORIZON

DASHBOARD



CI

VO
SE

NETW

Sahara

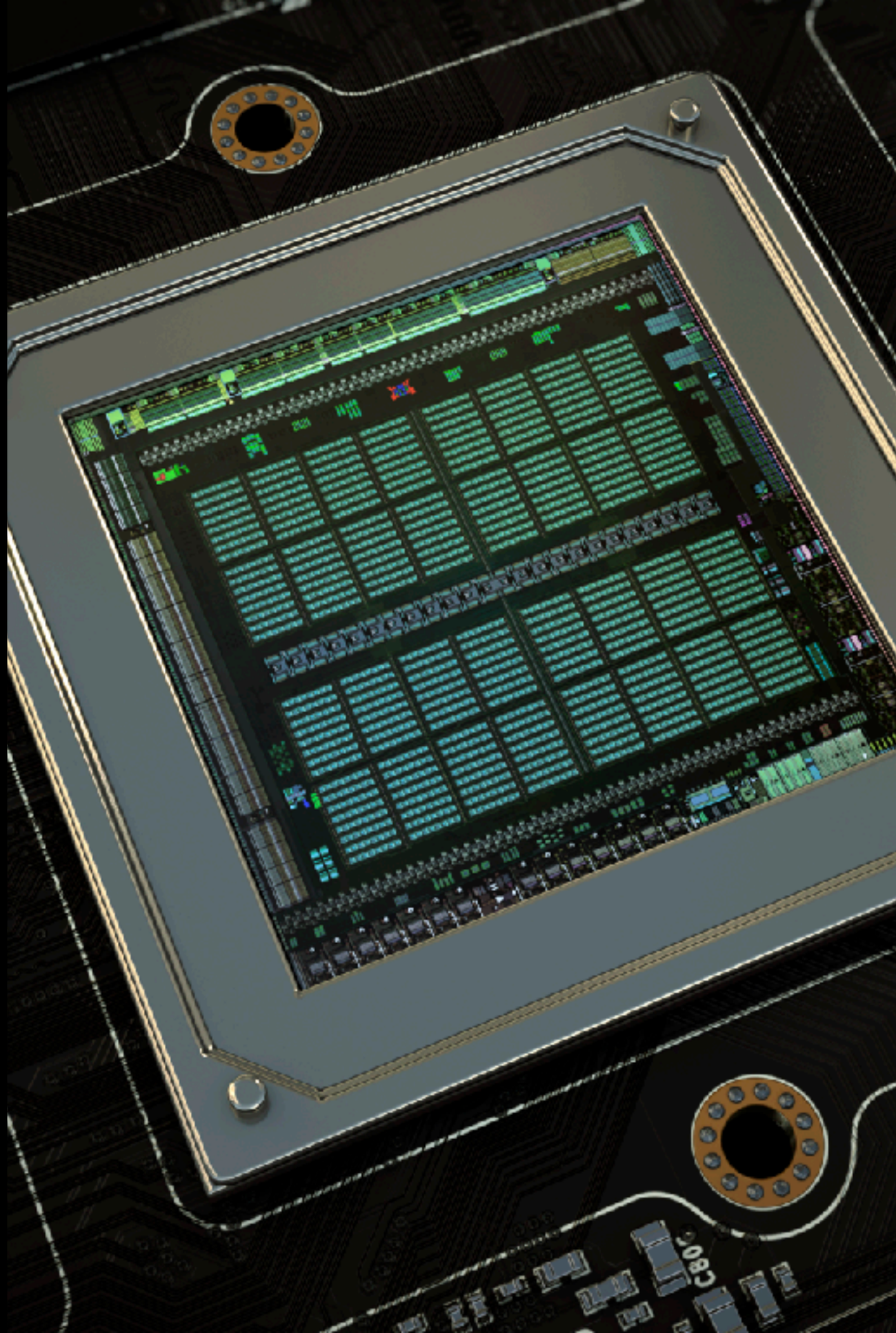
Big Data on OpenStack

‘GPU ON OPENSTACK’とは何か?

どうやってGPUはOPENSTACK環境で動作するのか？

‘GPU’ トレンドのおさらい

- 多くのGPUコアを利用する
  - 一部の計算は単一MPUユニットが小さくスピードが遅かろうとも多くのMPUコアを使うことはいくつかの計算処理には有効である
  - サーバ各々は高パフォーマンスでコンパクトにできる
- 低消費電力化はHPCエンドユーザにとって大事なこと
  - 多くの省電力・ハイパフォーマンスサーバを所有しやすくなる



どうやってGPUは動くのか？

- 現在のところPCIパススルーあるいはNVIDIA(GPU) dockerでの利用となる
- 'PCIパススルー' は土台となるベアメタル仮想環境に依存する
  - VSphereとXenは各々VMにGPUコアを任意の単位で分割して割り当てすることが可能
  - OpenStackの標準的なベアメタル仮想環境として使われるKVMではコア分割ができず、GPUユニットごとをVMに割り当てることとなる
- 昨今使われるコンテナ仮想環境でのデプロイとなるNVIDIA(GPU) DockerはDockerコンテナ同士で一つのGPUユニットを共有しているが、明示的な分割はしない
  - ウインドウズは'docker vm'としては動作しない

GPU OPENSTACKとは

- インスタントなHPC利用
  - いくつかの計算をしおえたらVMそのものを壊す。
  - ちょっとしたお試しでいくつかのVMを使ってHPCグリッド試してみる。終わったらすぐ壊す。
- GPUインターナルクラウドとしてのGPU利用
  - 主に製造業において、いくつかのシステムは情報管理上、パブリッククラウドに外だしができない。

SETUP:GPU ON OPENSTACK

どんなGPUのメカニズムがOPENSTACK環境でうごいているのか？

OpenStackでGPUを動かす方法（現在）

- PCIパススルー
  - PCIデバイスをダイレクトに接続する
  - ComputeNodeのハイパーバイザー依存、OpenStack依存ではない
  - Xen利用でのGPUコア分割、KVMはNVIDIA/AMDともコア分割不可
  - Intel GVT-g(Xen)/GVT-d(KVM)によるIntel GPUのコア分割
- コンテナ
  - NVIDIA Dockerの利用
  - 複数のコンテナによってGPUを利用できるが、明示的なGPUコア分割はできない
    タスク次第でのGPU利用
  - Kubernetes/Mesos/Docker Swarmなどとの組み合わせ管理

PCIパススルーはVM上でどう動作するのか？

- PCIデバイスをダイレクトにLinuxホストを通じて接続する
  - 物理ホストよりデバイスを切り離す必要あり
    - (NVIDIA) Dockerと違いホスト切り離しとなるため、監視管理等は各VMに必要
    - OpenStack依存のことではなく、Linuxのベアメタル環境に依存
- KVM上では一つのGPUデバイスにひとつのVM
  - Docker/Xen/VSphereと違いGPUは分割もできなければ共有もできない
  - これはKVMの制限であってOpenStackによるものではない

PCI Passthrough on OpenStack

- Redhatは公式サポート
  - ただ、あまり推しでない気が..
- Ubuntuはドキュメントすらまともにない...
- ググって探すしかない（さすがUbuntu..orz）
- 目下小職/NVIDIA JAPAN/DELLEMC/VTJで再度細く検証とOpenStackコミュニティのためにドキュメントをまとめ上げる予定（趣旨に同意いただき参画いただける企業様歓迎いたします)

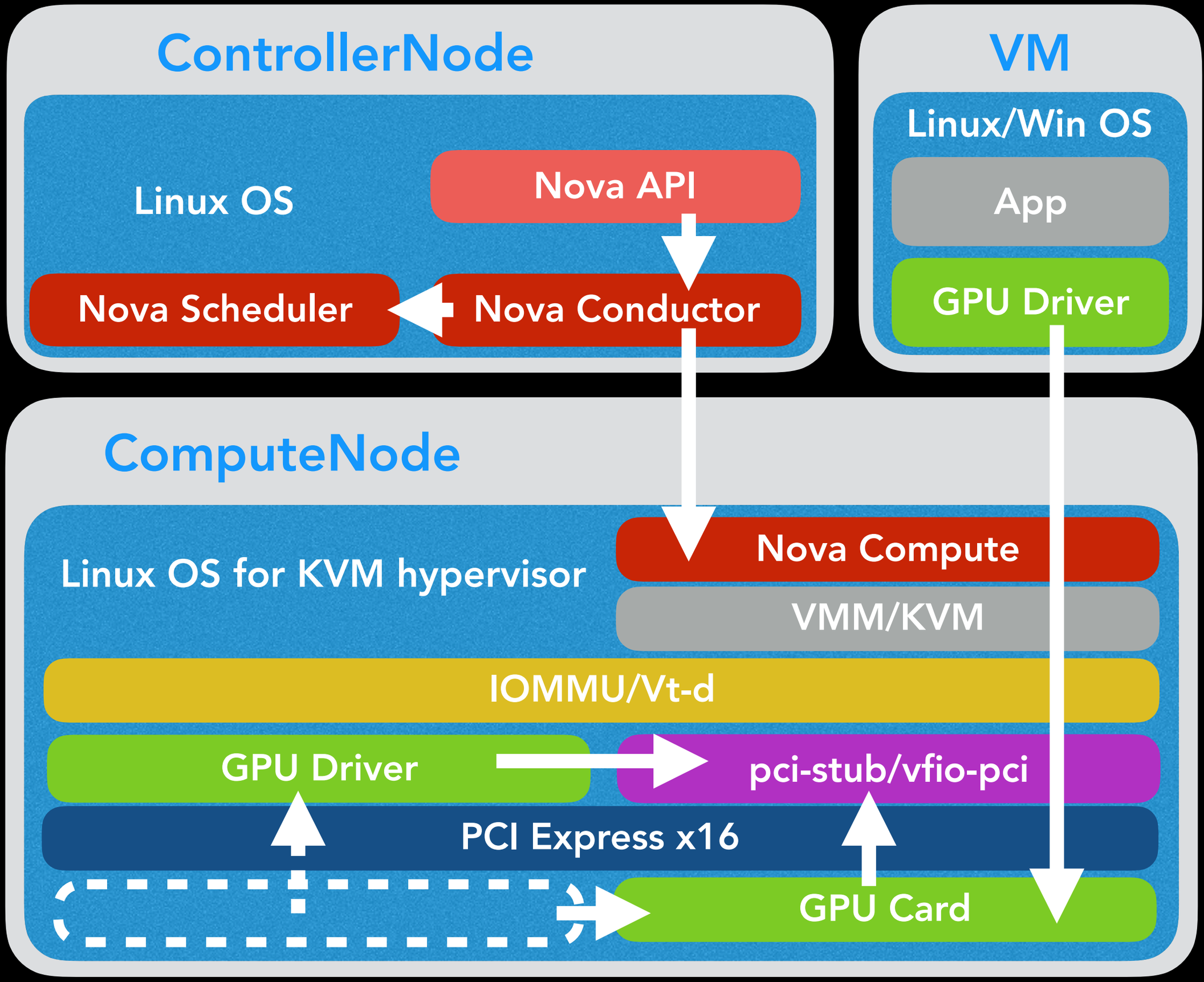


図1:OpenStackどうやってGPUパススルーを実現しているのか？(KVMの場合)

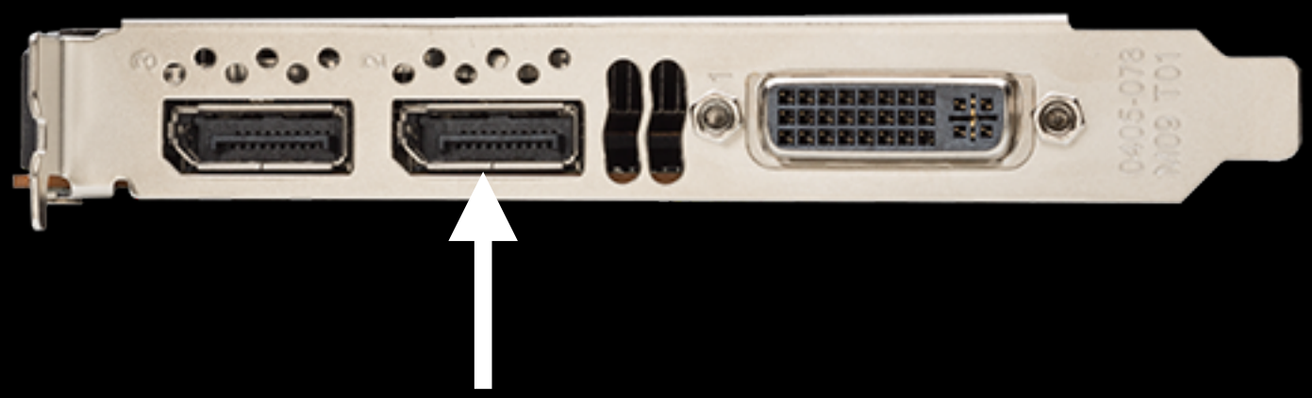
Ｓｔｅｐ1:ホストのＧＰＵの状態をまずは調べる

- lspci -nn | grep -i nvidia でGPUの状態をまずはチェック

```
lspci -nn | grep -i nvidia
88:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device [10de:11b4] (rev a1)
88:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller [10de:0e0a] (rev a1)
```

- 全てのGPUユニットがパススルーされている必要がある
  - GPUだけでなく、ユニットとして認識されるGPUデバイス自体もパススルーする必要あり
  - でないとVMは動いてくれない..（完全にパススルーにならない)

GPUポートがパススルーされているかチェック



- QuadroなどHDMIはビデオポートだけでなくオーディオポートももっているため注意
  - lspciでチェックされたものについては全てパススルーさせる必要がある。

STEP2:IOMMUセットアップ

- IOMMU(Input/Output Memory Management Unit) は物理デバイスを仮想化システムで使う上で必要なもの
- もちろんvt-dはオンにする必要あり (EFI/BIOSののデフォルトはON)
- intel_iommuとvfio_iommu_type1.allow_unsafe_interruptsの設定を/etc/default/grubに行う必要あり

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash intel_iommu=on
vfio_iommu_type1.allow_unsafe_interrupts=1”
```

STEP3:pci-stubとVFIO

- pci-stubは物理デバイスをLinuxホスト側が利用できないようにする
- VFIO(Virtual Function IO) は pci-stubと同様の働きをする。kernel 4.1以降のサポート
  - 未使用時はデバイスをD3ステート(低消費電力モード)に変更する

- It is not used by default デフォルトでは使われないため /etc/module を編集して、これらと関連するコンポーネンツを追記する (kvm,kvm_intel)

```
pci_stub
vfio
vfio_iommu_type1
vfio_pci
kvm
kvm_intel
```

STEP4-1:ブラックリスト(1)

- ramfsでGPUデバイスを認識できないようにする
  - /etc/initramfs-tools/modules to initramfs (ubuntuの場合)

```
echo ‘pci_stub ids=10de:11b4,10de:0e0a’ >>  /etc/initramfs-tools/modules
sudo update-initramfs -u && sudo reboot
```

STEP4-2:Blacklist(2)

- ブート時にGPUデバイスを認識できないようにする。

- /etc/modprobe.d/blacklist.conf に次を追加:

```
blacklist nvidia
blacklist nvidia-uvm
```

- ドライバについてもブラックリストに入れる必要あり

```
blacklist nouveau
```

STEP5:物理からのアンバインド

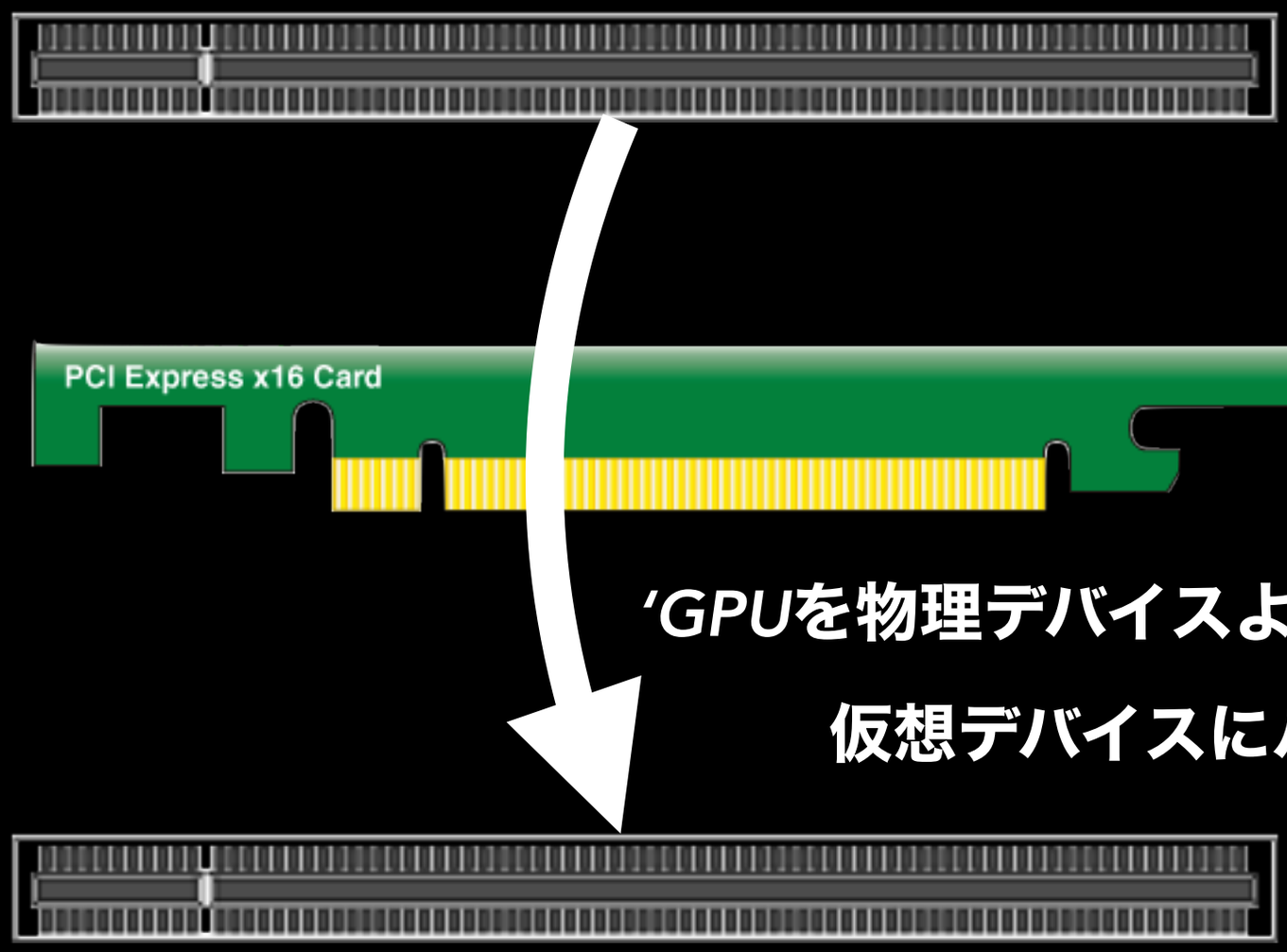
- 『物理ホストから切り離しVMへのバインドするため』にpci-stubをチェックする
  1. pci-stub/new_idにパススルーするドライバのPCI IDをエントリする。
  2. 関連するPCI識別子を物理ホストから切り離す
  3. pci-stubにこのPCI識別子をバインドする。

```
echo 11de 11b4 > /sys/bus/pci/drivers/pci-stub/new_id
echo 11de 0e0a > /sys/bus/pci/drivers/pci-stub/new_id
echo 0000:88:00.0 > /sys/bus/pci/devices/0000:88:00.0/driver/unbind
echo 0000:88:00.1 > /sys/bus/pci/devices/0000:88:00.1/driver/unbind
echo 0000:88:00.0 > /sys/bus/pci/drivers/pci-stub/bind
echo 0000:88:00.1 > /sys/bus/pci/drivers/pci-stub/bind
```

- 物理ホストのdmesgに以下の’claimed’がブートプロセスにあるかどうか確認する。

```
pci-stub 0000:88:00.1: claimed by stub
```

echo 0000:88:00.0 > /sys/bus/pci/devices/0000:88:00.0/driver/unbind
echo 0000:88:00.1 > /sys/bus/pci/devices/0000:88:00.1/driver/unbind



物理デバイス

GPUユニット
(全てのデバイス)

‘GPUを物理デバイスよりアンバインドして
仮想デバイスにバインドする’

pci-stub
(仮想で使う)

echo 11de 11b4 > /sys/bus/pci/drivers/pci-stub/new_id
echo 11de 0e0a > /sys/bus/pci/drivers/pci-stub/new_id
echo 0000:88:00.0 > /sys/bus/pci/drivers/pci-stub/bind
echo 0000:88:00.1 > /sys/bus/pci/drivers/pci-stub/bind

図2:物理にあるGPUを切り離し仮想につなぐ

UEFI/BIOS
Vt-d

IOMMU

GRUB
/etc/default/grub

IOMMU

ramfs
/etc/initramfs-tools/modules

BLACK LIST

modules
/etc/modules

BLACK LIST

IOMMU

modprobe
/etc/modprobe.d/blacklist.conf

BLACK LIST

pci-stub
/sys/bus/pci/drivers/pci-stub/
/sys/bus/pci/devices/$(Identifier)/driver/unbind

BLACK LIST

図*3:*ブート時の*GPU*ブラックリストのプロセス*(Ubuntu*の場合*)*

GPUを追加する(1)

- lspci の結果を確認- 2つの関連するPCI識別子が確認できるはず(識別子番号は使用するシステムに依存)

```
lspci -nn | grep -i nvidia
88:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device [10de:11b4] (rev a1)
88:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller [10de:0e0a] (rev a1)
84:00.0 VGA compatible controller [0300]: NVIDIA Corporation Device [10de:11b4] (rev a1)
84:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller [10de:0e0a] (rev a1)
```

- pci-stubにさらにパススルーするGPUの追記をする。

```
echo 0000:84:00.0 > /sys/bus/pci/devices/0000:84:00.0/driver/unbind
echo 0000:84:00.1 > /sys/bus/pci/devices/0000:84:00.1/driver/unbind
echo 0000:84:00.0 > /sys/bus/pci/drivers/pci-stub/bind
echo 0000:84:00.1 > /sys/bus/pci/drivers/pci-stub/bind
```

GPUを追加する(2)

- 追加のGPUがうまく動いたかVMで確認

```
ubuntu@guestos$ lspci -nn | grep -i nvidia
00:07.0 VGA compatible controller [0300]: NVIDIA Corporation GK104GL [Quadro K4200] [10de:11b4]
(rev a1)
00:08.0 VGA compatible controller [0300]: NVIDIA Corporation GK104GL [Quadro K4200] [10de:11b4]
(rev a1)
```

- アプリによっては両方同時に利用する際など、同一のGPUである必要がある可能性がある。

```
/nbody -benchmark -numdevices=2 -num bodies=65536
```

Openstack:nova-apiの設定

- ControllerNodeの/etc/nova/nova.confを編集しnova-apiを再起動する
  - pci_aliasにPCIデバイスの情報、エイリアス名を記述

```
pci_alias={“name”:”K4200”,"vendor_id":"10de","product_id":"11b4"}
```

OpenStack:nova-computeの設定

- ComputeNodeにある/etc/nova/nova.confを編集し、nova-computeを再起動する。
  - pci_passthrough_whitelistにPCIデバイスの情報、エイリアス名を記述

```
pci_passthrough_whitelist={“name”:”K4200","vendor_id":"10de","product_id":"11b4"}
```

*このケースの場合、ベンダーIDとプロダクトIDが適合したデバイスは全てVMにパススルーする。

- pci_aliasににPCIデバイスの情報、エイリアス名を同様に追記
  *Neuton以降

```
pci_alias={“name”:”K4200”,"vendor_id":"10de","product_id":"11b4"}
```

OpenStack:nova-schedulerの設定

- ControllerNodeの/etc/nova/nova.confを設定しnova-schedulerを再起動する。
  - PciPassthroughFilterを利用できるようにするためにPciPassthroughFilterをscheduler_default_filtersに追記する
  - 同様にPciPassthroughFilterをscheduler_available_filtersに記述する

```
scheduler_available_filters=nova.scheduler.filters.all_filters
scheduler_available_filters=nova.scheduler.filters.pci_passthrough_filter.PciPassthroughFilter
scheduler_default_filters=DifferentHostFilter,RetryFilter,AvailabilityZoneFilter,RamFilter,CoreFilter,DiskFilter,ComputeFilter,ComputeCapabilitiesFilter,ImagePropertiesFilter,ServerGroupAntiAffinityFilter,ServerGroupAffinityFilter,AggregateInstanceExtraSpecsFilter,PciPassthroughFilter
```

ControllerNode

Linux OS

pci_alias

Nova API

PCIデバイスを利用可にしてPCI利用のリクエストを送れるようにする

scheduler_available_filters

scheduler_default_filters

Nova Scheduler

pcipassthroughfilterを利用してGPUパススルーされたComputeNodeを選ぶ

Nova Conductor

VM

Linux/W

App

GPU D

ComputeNode

Linux OS for KVM hypervisor

pci_passthrough_whitelist

pci_alias

Nova Compute

GPUパススルーしたインスタンスをpci_aliasとpci_passthrough_whitelistによって発生させる

VMM/KVM

図4:NovaがGPU(PCI)パススルーされているComputeNodeで動作するプロセス

OpenStack:flavor-keyの設定

- flavor-keyを設定しGPUインスタンスで利用できるようにPCIパススルーの設定をflavorに追記する
  - pci_passthrough:alias=$(pci_alias_name):$(the number of GPUs we would like to use)

```
nova flavor-key $flavor_name set “pci_passthrough:alias”=“K4200:$(the number_of_gpus)”
```

既知の問題

GPU ON OPENSTACKを使うにあたって注意すべき問題点

Cloudイメージの問題

- CloudイメージはGPUを使う上ではとても小さくqemu-imgでリサイズする必要がある
- CUDAドライバはperl-packages(dev packages)がインストール時に必要
    - それが.debあるいは.rpmパッケージであろうとインストールが必要になる。なぜならCUDAパッケージ自体がバイナリパッケージでなく、ソースコードよりバイナリをビルドしていて、makeをパッケージインストールの際に実行している
- NVIDIA曰くCUDAドライバの次期リリースでFIX予定とのこと
    - CUDA 7.6以降でfixの予定だったのに..まだfixされてない..orz

VDIとしてのWindows利用

- CUDA on Windows は思ったより早いけどカクカクしてしまう
- 多分DISKのスピード、ネットワークなどなどいろいろなものが予想される。多分エフェメラルモードやその他爆速系のSSD/NVMeの利用やら10g以上のネットワーク環境が必要となるだろう
    - まだ改善対応をやってみたことはないが、なぜ発生するのかを調べるべく近々検証予定。
    - VMは基本メモリ/NW転送などコンテキストスイッチでの動作をするため、CUDAの重いワークロードはカクカクする可能性があるかもしれない。
    - 詳しくはデモビデオにて...
- GPU on OpenStack上でのWindowsの動作はもっと調査が必要..時間がほしい...

VDIとしてのWindows利用(feedback)

- Thanks giving some feedbacks to my session at LC3 China!
- Should checked and investigate the Windows-related issues below, will update later.
  - Windows 10 on KVM issue
    - http://bart.vanhauwaert.org/hints/installing-win10-on-KVM.html
      - Windows 10 deployment is succeeded from my KVM but I still have failed the same deployment from my OpenStack.I should investigate more what happened in detail.
  - Nvidia Driver issue (version 337.88 or later)
    - https://wiki.archlinux.org/index.php/PCI_passthrough_via_OVMF#.22Error_43:_Driver_failed_to_load.22_on_Nvidia_GPUs_passed_to_Windows_VMs
      - 'Nvidia drivers on Windows check if an hypervisor is running and fail if it detects one, which results in an Error 43 in the Windows device manager. ' I haven't found this issue on my Windows 7 VMs so I should check more in detail
      - Related links libvirt for adding the driver, should be checked.
        - https://github.com/openstack/nova/blob/master/nova/virt/libvirt/config.py#L2025-L2036
        - https://github.com/openstack/nova/blob/master/nova/virt/libvirt/driver.py#L4262-L4264

ライブマイグレーションの問題

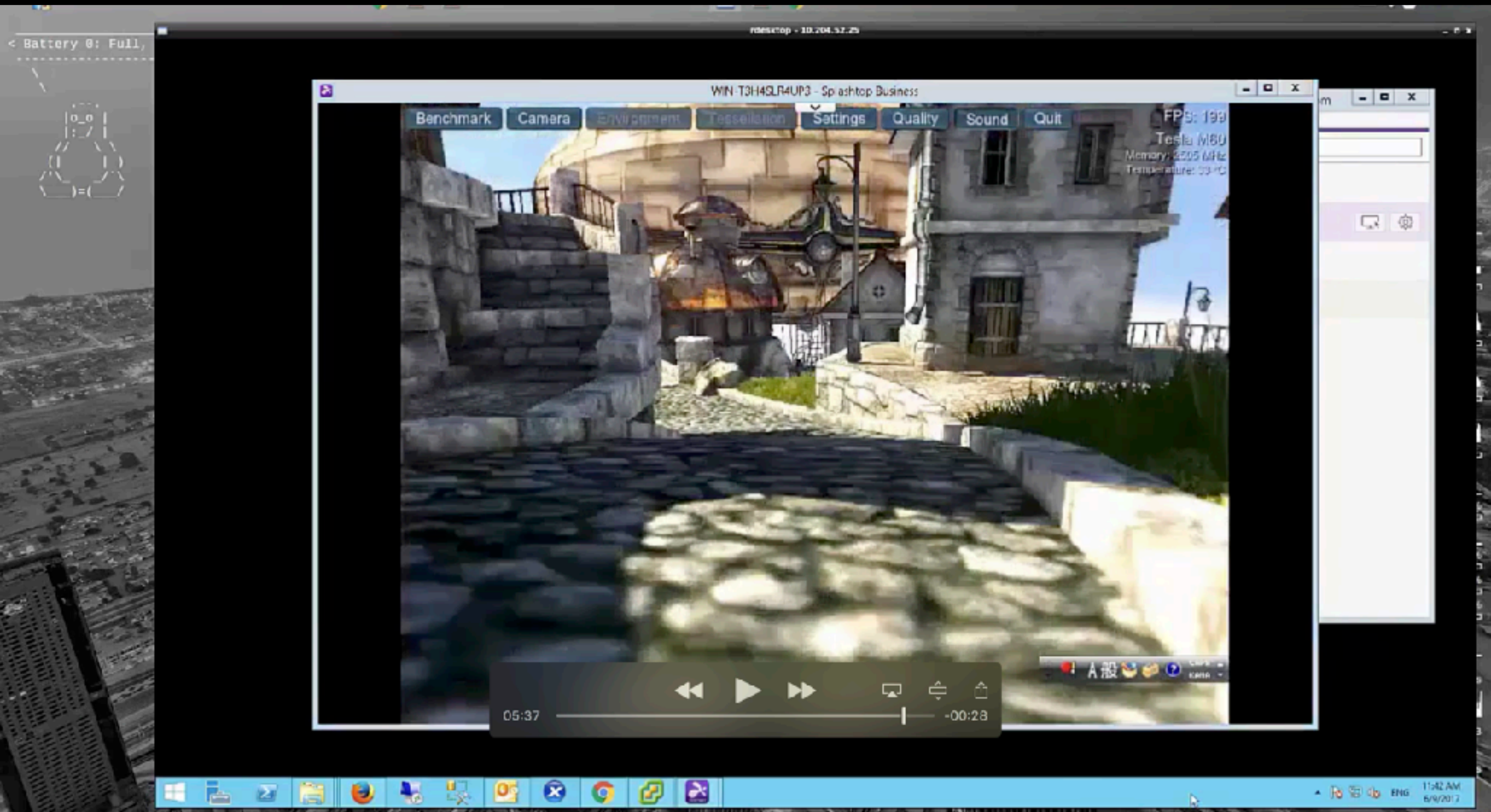
- PCIパススルーを使ったVMはライブマイグレーションができない。マイグレーション前のホストのPCIコネクションが切れないままになってしまう。
- ワークアラウンド:古いコネクションを下記にあるようなmysql DBのnova.pci_devicesから削除し古いホストを再起動する。
  - 再起動は他のホストに影響するため、ありえない。他にmysql DBなど関連プロセスの再起動で乗り切る形もあるが、これも他のホストに影響するためありえないことになる。

```
| 2016-08-11 00:54:45 | 2016-08-19 04:58:01 | NULL    |       0 | 45 |          21 | 0000:84:00.0 | 11b4   | 10de   | type-PCI | pci_0000_84_00_0 |
label_10de_11b4 | available | {}          | NULL                    |          NULL |        1 | <<-- old-host
| 2016-08-11 00:54:45 | 2016-08-19 04:58:01 | NULL    |       0 | 48 |          21 | 0000:88:00.0 | 11b4   | 10de   | type-PCI | pci_0000_88_00_0 |
label_10de_11b4 | available | {}          | NULL                    |          NULL |        1 | <<-- old-host
```

デモ(ビデオ)

GPUがOPENSTACK環境でどう動くか

チェック！



インスタンスを立ち上げGPUの動作を確認

- RHEL OpenStackを利用
- パススルーしてあるインスタンスを立ち上げる
- Ubuntuインスタンスを立ち上げlspciとdevicequeryを実行、GPUの動作を確認
- Windows上でのGPUの動きをGraphicsベンチで確認、RDSでのリモートサイトにあるものの接続のため、カクカクしてしまうがベンチの結果だけ？良いことを確認する。
  - デモビデオではSlashtopを利用

関連リンク

- Attaching physical PCI devices to guests: https://docs.openstack.org/admin-guide/compute-pci-passthrough.html
- Container as a Service on GPU Cloud- Our Decision Among K8s, Mesos, Docker Swarm, and OpenStack Zun: https://www.slideshare.net/secret/AiUdO4dLxNTkfI
- OVMF_による_PCI_パススルー https://goo.gl/icq9mV

Special Thanks to:

- GPU on OpenStack project members
  VirtualTech Japan
  NVIDIA
  DellEMC
  NEC Networks & System Integration

- @zgock999 at Tokaido-LUG, Nagoya, Japan
  Teach me some hints how to use GPGPU on VM!

- Matthew Treinish of IBM attended my session at LC3 china and figure out and feedback some point!

- Our customers! give the chance to evaluate!

THANKS VERY MUCH FOR COMING MY SESSION!

PRESENT BY MASAFUMI OHTA
TWEET @masafumiohta mailto:masafumi@pid0.org