「書くべきは手順書ではなく
スクリプトです。」

定型業務をスクリプトで自動化して楽をしよう。

OSC2019北海道
2019年6月1日
有限会社サンビットシステム
佐々木伸幸

Sunbit System

きょうお話すること

- やらなきゃいけないことをやる
  - LOOP
    - 課題はなにか
    - 効率的な解決手順を考える
    - 前の手順とつなげる
  - できた手順を実行する

- 具体的にどうやって考えればいいの？

Sunbit System

やらなきゃいけないこと

- XenServerベースのcloudstackで管理されている200台のVMを
- 新しくXenServerクラスタを作成してそこに移行する。

やることには条件がある

- VM1台について
  - ブートディスクは最大100GB
  - 追加ディスクが最大5台、最大容量は1TB/台
  - メモリ構成、CPU構成は変更しない
    - 1CPU+3GB, 3CPU+8GB, 10CPU+16GBの３種
  - DNS名、MAC、IPは変更しない
    - CloudStack環境のネットワーク構成を移行しないと、複数ホスト構成で動作していたものが動作しなくなる可能性大

やることには条件がある

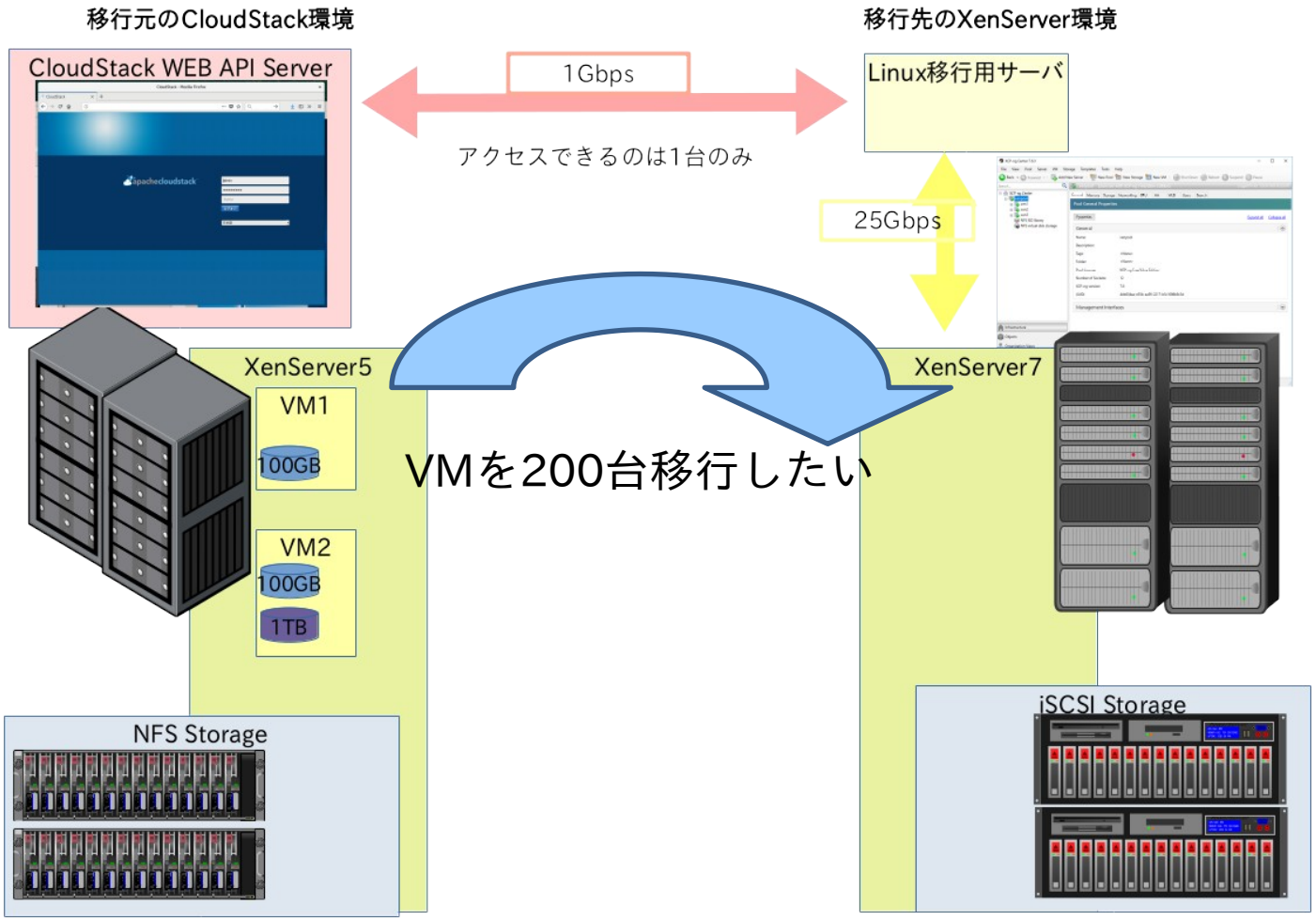
- CloudStack環境について
  - WEBアクセス、APIアクセスは新しい環境のLinuxから可能（但し許可されるのは1台）
  - ストレージはNFSで新しいXenServer環境から参照可能（但し許可されるのは1台）
  - VMのブートディスクをどうやって取り出すかは、この時点では謎
  - VMのディスクイメージはVHD形式という噂

やることには条件がある

- 新しいXenServer環境について
  - XenServer7.6でクラスタ構成
  - 200台のVMを十分収容可能なリソース
  - ストレージ回線帯域は25Gbps
  - CloudStackのホストへアクセスする回線帯域は1Gbps。アクセスできるのは移行用サーバ1台。

Sunbit System

つまりこういうこと

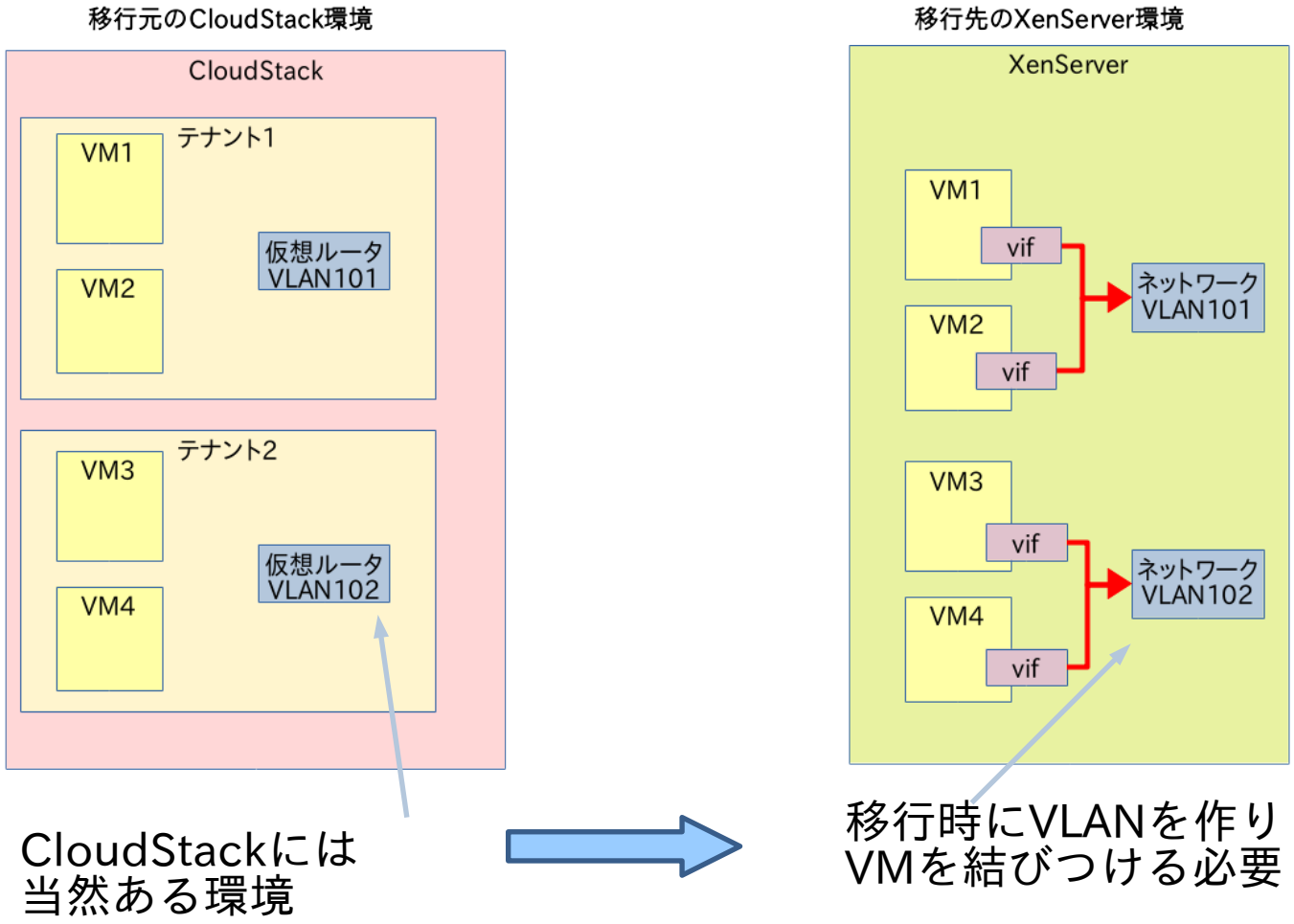


Sunbit System

CloudStack環境にあってXenServer環境にないこと

- テナント/仮想ルータ
  - グループに割り当てるリソース単位の概念
  - テナント毎に仮想ルータ
  - VLAN, DHCP, DNSなどは仮想ルータが管理
- XenServerにはテナントの概念がない
  - 移行する環境で何らかの形で作成が必要
    - G/W装置でVLAN,DHCPを事前に設定
      - MACを変えなければIPは問題なし
    - Xen環境のVLANは移行時に作成が必要

つまりこうしなきゃいけない



Sunbit System

最初の移行実験

- CloudStackからVMを引っ張り出す
- WEBインタフェースにあるもの
  - テンプレートの作成
  - テンプレートのエクスポート

- ん？Xenだとxvaにエクスポートできるんじゃないの？
  - CloudStackからだとできないらしい（げ）

Sunbit System

移行を試す

- GUIで操作して移行する手順を考える
  - WEB-UIでCloudStackのVMを停止する
  - VMをテンプレート化する
  - テンプレートをエクスポートする
  - 移行先のXenCenterでインポートする

移行を試す

- 手順に沿ってやってみる
  - VMを停止する
  - VMをテンプレート化する
    - 問題：テンプレートにするとVMでなくなる!
    - 理解：ファイルの形式はvhdらしい
  - テンプレートをエクスポートする
    - 問題：むちゃくちゃ遅い!!
  - XenCenterからインポートする
    - 問題：MACが変わる(あとから直せるけど)

Sunbit System

試行の検討

- VMはテンプレート化するとVMでなくなる
  - 「テンプレート変換」でコピーではない
- ダウンロードがむちゃくちゃ遅い
  - 100GBで2時間半。1TBだと26時間！！
    - 200台全部に1TBついてたら200日以上なの？
  - WEBサーバからダウンロードするのが遅い
    - 回線ではなく、WEBサーバが遅いらしい
- MACが変わる
  - 全部移行してから手で変えるのかぁ...

Sunbit System

もしこれを手順書にしていたら？

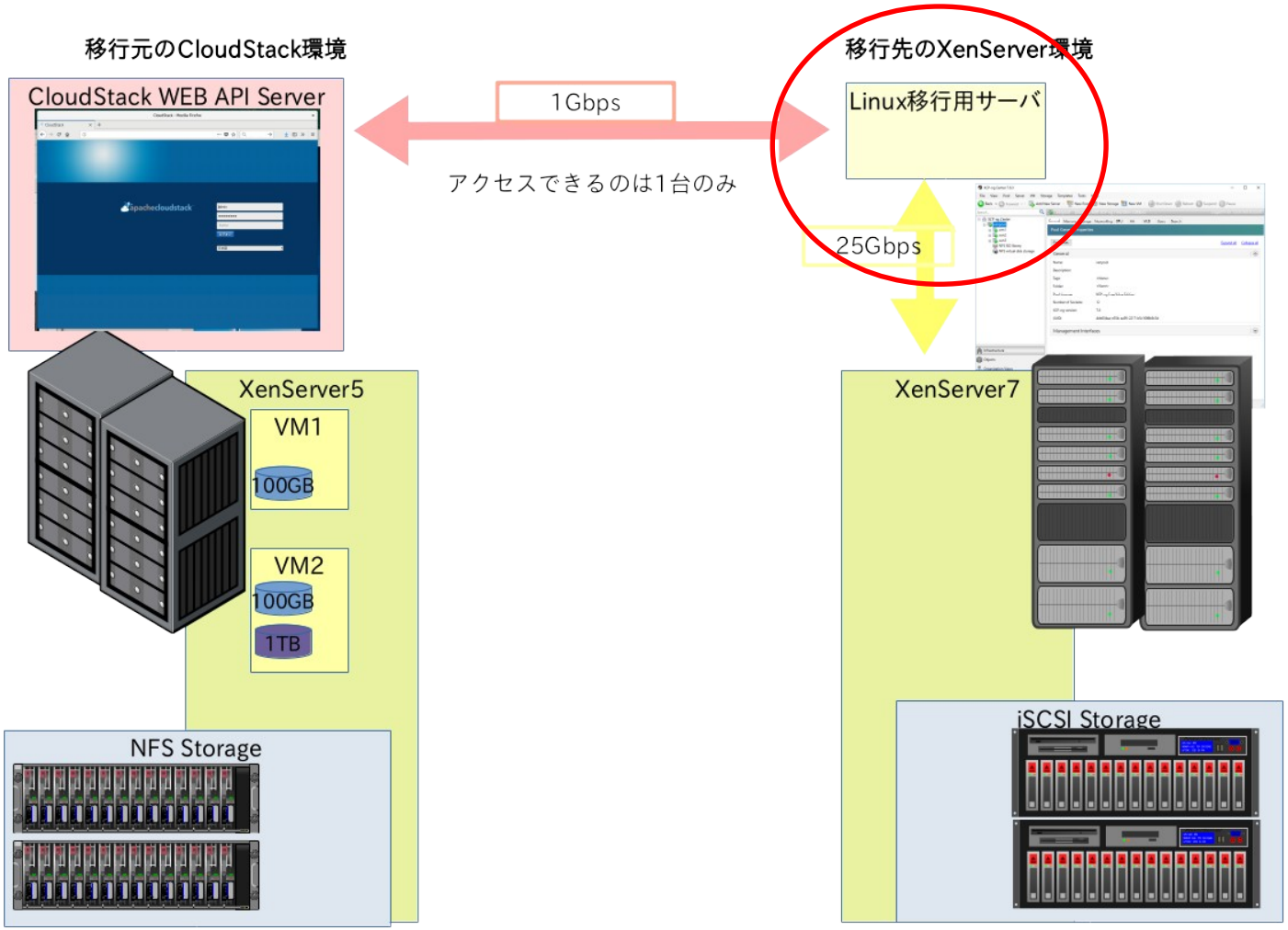
- やることを手順に書くことはできる
  - ログインとか、この画面でこれを押せも含む
- すべての手順を操作しなければならない
  - 間違う確率が高い、いや間違うでしょ

移行を試す(2)

- GUIを操作しない方法を考える
  - CloudStack API
    - テンプレート化以外にvhdをそのまま外部に出せる機能がある
    - コマンド発行して結果を取得できる
  - XenServer
    - XML-APIがある
    - xeコマンドでも全ての操作ができる
    - VMの作成とデバイスの作成を独立でできる
      - 後からVMとデバイスを結びつける
  - まずはコマンドラインで試す

Sunbit System

移行を試しているのはここ



Sunbit System

kick_api.sh

- CloudStack APIを実行するスクリプト
  - 例えばlistVirtualMachine

```
root # ./kick_api.sh command=listVirtualMachine zoneid=1 id=1000
<?xml version="1.0" encoding="ISO-8859-1"?>
<listvirtualmachinesresponse cloud-stack-
version="2.2.15.20121003223903">
  <count>425</count>
  <virtualmachine>
    <id>1000</id>
    <name>vm0014</name>
    <displayname>vm0014</displayname>
    <account>j21796</account>
    <domainid>1</domainid>
    <domain>ROOT</domain>
    <created>2016-01-29T13:55:34+0900</created>
    <state>Running</state>
...
```

Sunbit System

CloudStackAPI listVirtualMachineの情報

id	VMのID	
name	VMの名前	
state	VMの状態(Runnning, Halted)	
account	VMの所有者ID	
cpunumber	割当CPU数	
memory	割当メモリ数(MiB)	
ipaddress	IPアドレス	NICごとにある
netmask	ネットマスク	
gateway	ゲートウエイ	
networkid	ネットワークのID	
macaddress	NICのMACアドレス	

ディスクに関する情報がないですね
listVolumesに情報があるらしい

Sunbit System

CloudStackAPI listVolumesの情報

id	volumeのID	
name	volumeの名前	
type	ディスクの種別(ROOT,Data)	
deviceid	デバイス番号	≒　接続順序
virtualmachineid	結びついているVMのID	
vmstate	結びついているVMの状態	
storage	このvolumeがあるストレージ名	
size	サイズ(B)	
state	volumeの状態 (Allocated,Ready)	

ディスク(volume)に結びついているVMのIDが入っている
複数ディスクがあっても結びつけやすい構造

Sunbit System

APIから取得した有効な情報

- VMのCPU数、メモリ数
- VMのNIC情報(MAC,IP)
- VMに紐づくディスクファイル
- ディスクファイルのメタ情報
  - アロケートサイズ、種別、接続順序
  - ファイル名、格納場所パス名の一部
- これだけあればなんとかなる
- ディスクファイル取得行為が必要

Sunbit System

CloudStackAPI extractVolume

- 指定ディスクを外部に出せる形にする
  - 事前にVMは止めておきます。

```
root # ./kick_api.sh command=extractVolume zoneid=1 id=2019
mode="HTTP_DOWNLOAD"
<?xml version="1.0" encoding="ISO-8859-1"?>
<extractVolumeresponse cloud-stack-
version="2.2.15.20121003223903">
  <jobid>425</jobid>
  <jobstatus>0</jobstatus>
 …
</extractVolumeresponse>
```

- ん？すぐ終わる

Sunbit System

CloudStackAPI AsyncJobってなに？

- 時間がかかる処理はjobidを返しレスポンスまでブロックしない

```
root # ./kick_api.sh command=extractVolume zoneid=1 id=2019
<?xml version="1.0" encoding="ISO-8859-1"?>
<extractVolumeresponse cloud-stack-
version="2.2.15.20121003223903">
  <jobid>425</jobid>
  <jobstatus>0</jobstatus>
```

- queryAsyncJobResultコマンド

```
root # ./kick_api.sh command=queryAsyncJobResult jobid=425
<?xml version="1.0" encoding="ISO-8859-1"?>
<queryAsyncJobResultresponse cloud-stack-
version="2.2.15.20121003223903">
  <jobid>425</jobid>
  <jobstatus>1</jobstatus>    <<ここが1になったら終了している
```

Sunbit System

extractVolumeの終了結果

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<extractVolumeresponse cloud-stack-
version="2.2.15.20121003223903">
  <jobid>425</jobid>
  <jobstatus>1</jobstatus>
  <jobresult>
    <volume>
      <id>2019</id>
...
  <state>DOWNLOAD_URL_CREATED</state>
...
  <url>http:%2F%2F192.168.128.10%2Fuserdata%2F65ae79b7-c1d3-
49f7-8c85-2611e94ff1d1.vhd</url>
...
```

- ダウンロード用のURLが返される
  - でもHTTPは遅いことが判明している

Sunbit System

ファイルはどこにある？

- 移行前の管理用Linuxからストレージを NFSマウントしているらしい
  - 見せてもらうと、Primaryにあった！
  - /マウントポイント/volumes/ボリュームID/ uuid形式のファイル名.vhd
  - このファイルは24時間後に削除される設定
- このパスをXenServerが見れたら 直接インポートできるよね？
  - この時点では管理用Linuxから見てる

Sunbit System

追加ディスクはどこにある？

- 移行前の管理用Linuxからストレージを NFSマウントしているらしい
  - 見せてもらうと、Secondaryに見えている
  - /マウントポイント/ディスククラスタ名/uuid形式のファイル名.vhd
  - このファイルは実態がそのまま見えている
- このパスをXenServerが見れたら 直接インポートできるよね？
  - この時点では管理用Linuxから見てる

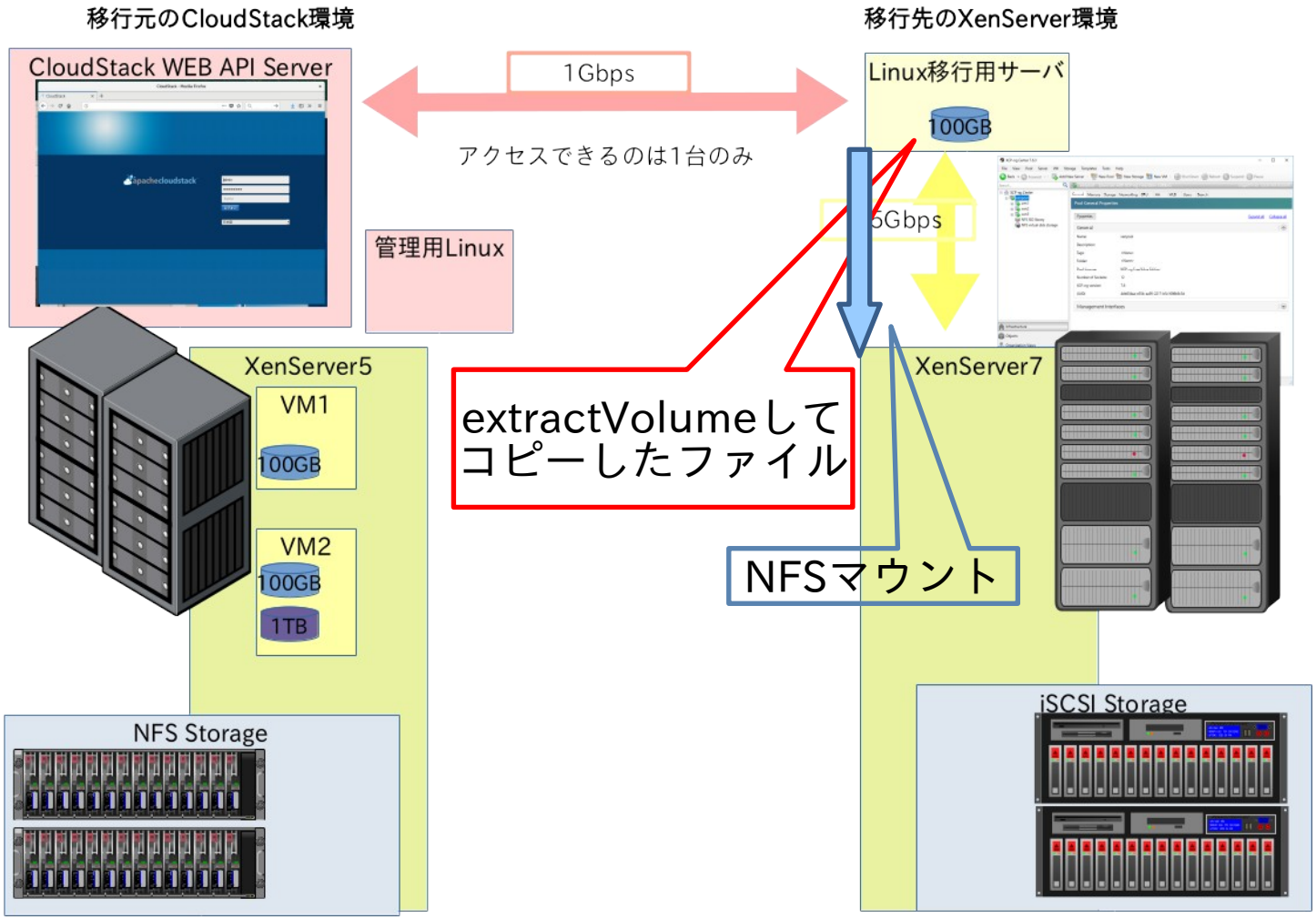
Sunbit System

クラスタ？

- ストレージがクラスタ運用されていて、どのクラスタにどのボリュームを格納するのか、をCloudStackが管理してるらしい
  - 見る方法はあるのか
    ＞mysqlに格納されている
  - mysqlの内容確認
    - volumeid,クラスタ名,ファイル名がある
      ＞ディスクファイルのパスを特定可能
- ディスク関係はこれで拾える
    - 管理用Linuxから見えるディスクファイルは移行用Linuxにコピーする必要あり(最大5TBあるけど...)

Sunbit System

ちなみにここまでの状態



Sunbit System

ネットワーク

- もともと移行先には仮想ルータがない
  - VMが所属するネットワークとVLANは新規に設計して実装するしかない
    - VMとVLANの関係表となるはず
  - 新たな設計情報を参照してXenServerにVLANを作成しておく
  - 移行するときそのVLANにVMを結びつける

XenServerの操作

- xapi
  - XenServerのapi
  - xeコマンドで操作可能
  - Nativeなapi操作ツールもあるようだが、それを習得するよりはxeコマンドのほうが資料も多い
- パスフレーズなしssh鍵認証で移行用サーバからxeコマンドをリモート操作する

Sunbit System

VMを作る

- xe vm-install
  - 返されるuuidがVMの識別子になる

```
[root #~] ssh root@192.168.10.20 xe vm-install \
      new-name-label=vm0001 \
      sr-uuid=e3532491-ecc2-2ae6-a59f-88eddedb5dff \
      memory=3221225472\
      template=552bce37-51b2-445d-84f2-5f33fa112d7e
3dc3e0dd-0309-85d9-84a8-88ffb76340bf

[root #~]
```

このあとの操作でVMを指定するために必要

Sunbit System

VMのリソースを設定する

- xe vm-param-set
  - CPU数など、必要なパラメータを設定する

```
[root #~] ssh root@192.168.10.20 xe vm-param-set \
      uuid=3dc3e0dd-0309-85d9-84a8-88ffb76340bf \
      VCPUs-max=1

[root #~] ssh root@192.168.10.20 xe vm-param-set \
      uuid=3dc3e0dd-0309-85d9-84a8-88ffb76340bf \
      VCPUs-at-startup=1
```

Sunbit System

VMのディスクを設定する

- xe vdi-create, vbd-create
  - ディスクイメージを作成し、VMと紐付ける

```
[root #~] ssh root@192.168.10.20 xe vdi-create \
      name-label=ROOT-8239 \
      sr-uuid=e3532491-ecc2-2ae6-a59f-88eddedb5dff \
      virtual-size=107374182400 \
      device=0
eb2a2624-506b-44ca-a753-509030ce8ea4

[root #~] ssh root@192.168.10.20 xe vbd-create \
      vm-uuid=3dc3e0dd-0309-85d9-84a8-88ffb76340bf \
      vdi-uuid=eb2a2624-506b-44ca-a753-509030ce8ea4 \
      device=0 \
      type=Disk
400c035b-b830-7d8c-d091-c0f839de40e4
```

イメージを指定するために必要

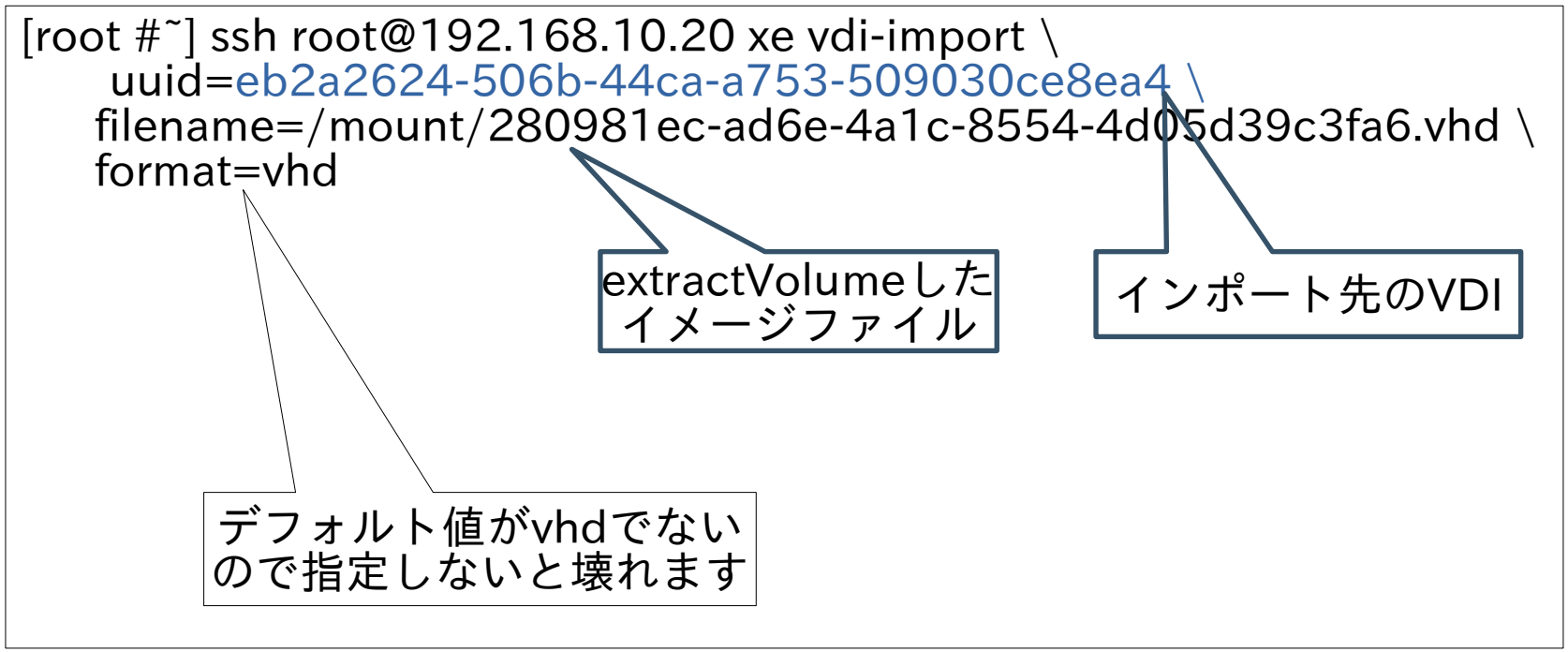
VBDはイメージとVMを紐付ける

Sunbit System

ディスクイメージを移行する

- xe vdi-import
  - vdiにディスクイメージをインポートする

```
[root #~] ssh root@192.168.10.20 xe vdi-import \
      uuid=eb2a2624-506b-44ca-a753-509030ce8ea4 \
     filename=/mount/280981ec-ad6e-4a1c-8554-4d05d39c3fa6.vhd \
     format=vhd
```



Sunbit System

VMが接続するネットワーク

- xe network-create, pool-vlan-create
  - ネットワークを作りVLANに結びつける

```
[root #~] ssh root@192.168.10.20 xe network-create \
      name-label=vlan1001
706b42d2-2834-11a3-54b6-c63aa4f257a0


[root #~] ssh root@192.168.10.20 xe pool-vlan-create \
      network-uuid=706b42d2-2834-11a3-54b6-c63aa4f257a0 \
      pif-uuid=69cdcbc0-7397-4b85-39a4-a97169057ba5 \
      vlan=vlan1001

6b4c91c9-0b95-53d7-2cf8-879fc6591899
635cf8e1-ce7d-824e-9de6-179c91876ec9
df883341-a4fc-f7a4-2f32-13e67c3ea5cb
```

networkを指定するために必要

networkを載せるI/Fのuuidは先に調べておく必要

poolのホスト毎に作られる

Sunbit System

VMが接続するネットワーク

- xe vif-create
  - VMとネットワークを紐付ける

```
[root #~] ssh root@192.168.10.20 xe vif-create \
      vm-uuid=3dc3e0dd-0309-85d9-84a8-88ffb76340bf \
      network-uuid=706b42d2-2834-11a3-54b6-c63aa4f257a0 \
      mac=02:00:0d:6e:00:01 \
      device=0
```

元のMAC

作成したnetwork

installしたVM

Sunbit System

手順のおさらい：事前準備

- CloudStack上のVMリストを得る
- CloudStack上のvolumeリストを得る
- CloudStack上のvolumeidとクラスタの関係表を得る
- 移行対象VMリストを得る
- 移行対象VMのVLAN情報を得る
- 移行対象VMとVLAN情報、MAC,IPを紐付ける

Sunbit System

手順のおさらい: CloudStack

- VMを停止する
- extractVolumeでイメージファイルを作成する
- 作成したイメージファイルをXenServerが参照できるところに配置する
- 追加ディスクがあればそのイメージファイルもXenServerが参照できるところに配置する

手順のおさらい: XenServer

- VMをinstallする
- VMにCPUなどのパラメータを設定する
- 元と同じサイズのVDIを作成する
- VDIとVMを結びつける
- VDIをインポートする

手順のおさらい: XenServer

- Networkを作成する
- VLANを作成する
- VLANをPIF（物理I/F）と結びつける
- VMにVIFを作成する
- VIFとNetworkを結びつける

もしこれを手順書にしていたら？

- 事前準備だけで大きめの物語になる。
  - このXMLのこのタグのこの値で、あのXMLのこのタグが同じところを見て...
- XenServerで作成したuuidは記録しながら進める必要がある
  - GUIより手順は複雑＆動的になった
- そして手順を操作しなければならない
  - もう、間違わずにはいられない！

Sunbit System

もしこれを手順書にしていたら？

- 並列作業が難しい
  - 200台あんのよ、人海戦術でやる？
- 時間がかかる処理の待ち時間が多い
  - extractVolumeとscpは
    100GBで2時間半くらい
  - 移行は17:00以降翌日12:00まで
    深夜作業かよ...
  - 5TBあるやつ無理ぢゃね？

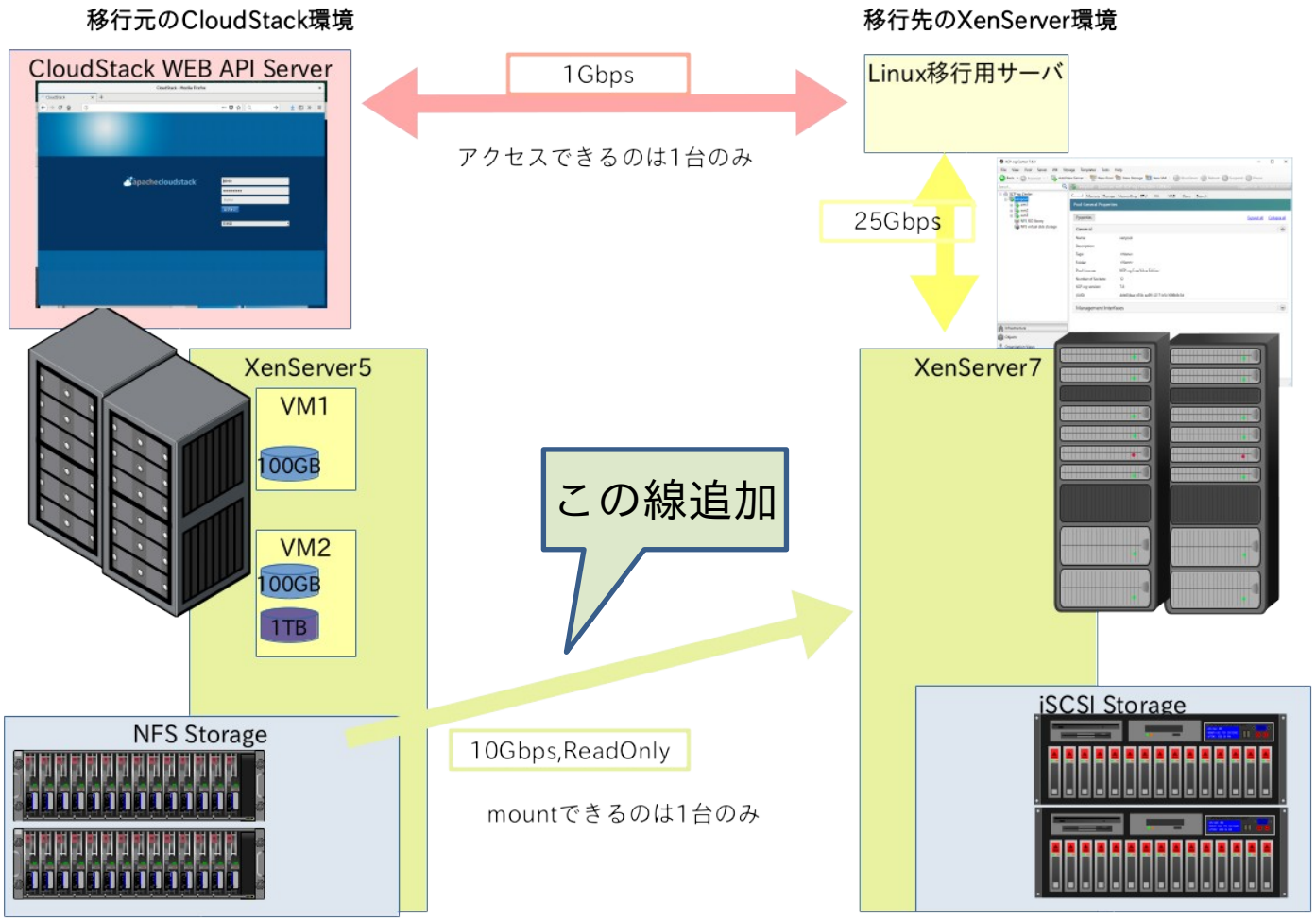
Sunbit System

ところで...

- CloudStack側のイメージファイルを移行用サーバにコピーしていましたね？
  - 割と大きな課題（時間がかかる,容量不足）
  - 交渉
    - CloudStackのストレージを直接見せてよ
    - HTTPやscpが遅いことは知ってるでしょ
    - ちょっと線引いて設定変更してもらえれば
    - 遅いでしょ？速いほうがいいでしょ？
- NFSマウント　ｷﾀｰｰ(ﾟ∀ﾟ)ｰｰ!!

Sunbit System

移行環境改善！



Sunbit System

ストレージが直接見えると...

- XenServerがvdi-importするときに直接ファイルを読み込める
  - volumeがブートディスクの場合
    primaryのvolumes/vmid/
    extractVolumeのファイル名.vhd
  - volumeが追加ディスクの場合
    volumeidとクラスタの関係表を参照し、
    secondaryの/クラスタ/uuid.vhd
- 事前準備でパスが作れる！
  - 但しブートディスクファイル名は不明

Sunbit System

そして速度比較

	1GB線 HTTP_DOWNLOAD 移行ホスト仲介	10GB線 NFS直接参照 XenServer直接マウント
ROOT 100GB extractVolume	５０分	50分
ROOT 100GB ファイル転送	約3時間半	不要
ROOT 100GB XenServerへのインポート	約2時間	約1時間半
DATA 1TB extractVolume	不要	不要
DATA 1TB ファイル転送	約26時間	不要
DATA 1TB XenServerへのインポート	未検証 (ファイル転送時間を見て萎える)	約10時間

Sunbit System

スクリプトを書こう！

- 要するに検証したコマンドを並べればいい
  - 事前準備する
  - CloudStackからVMを抜き出す
  - XenServerにVMを作る
  - VDIをインポートする

気をつけるポイント

- どうやったら効率的にできるか
  - 判っていることは先にデータ化する
    - listVirtualMachinesなどは一度取ればよい
      - ファイル化しておくとどこでもデバッグできる
    - Networkは新設計なので先にデータ化できる
  - データ化されたものを使いやすくまとめる
    - VMに紐づくディスクの順序とか
  - 何度も処理されるものは部品として書く
  - あとは手順通りに書く

Sunbit System

事前準備

- volumeidのリストを作る

```
my $volfile = 'listVolumes.xml';                # 先に取得したvolume情報
my %volumes;                                    # vol情報を格納するハッシュ

my $vols = XML::LibXML->load_xml(location => $volfile);
foreach my $v ($vols->findnodes('//volume')) {
        my $zoneid = $v->findvalue('./zoneid');
        my $id = $v->findvalue('./id');
        $volumes{$id}->{'id'} = $id;
        $volumes{$id}->{'name'} = $v->findvalue('./name');
        $volumes{$id}->{'vmid'} = $v->findvalue('./virtualmachineid');
        $volumes{$id}->{'vmname'} = $v->findvalue('./vmname');
        $volumes{$id}->{'storage'} = $v->findvalue('./storage');
        $volumes{$id}->{'size'} = $v->findvalue('./size');
        $volumes{$id}->{'deviceid'} = $v→findvalue('./deviceid');
}
```

事前準備

- VMのリストを作る

```
my $vmfile = 'listVirtualMachines.xml';   # 先に取得したVM情報
my %vms;                                  # vol情報を格納するハッシュ

foreach my $m ($vms->findnodes('//virtualmachine')) {
        my $zoneid = $m->findvalue('./zoneid');
        my $id = $m->findvalue('./id');
        $machines{$id}->{'id'} = $id;
        $machines{$id}->{'vmname'} = $m->findvalue('./name');
        $machines{$id}->{'cpunumber'} = $m->findvalue('./cpunumber');
        $machines{$id}->{'memory'} = int($m->findvalue('./memory')) *
1024 * 1024;
        $machines{$id}->{'nicid'} = $m->findvalue('./nic/id');
        $machines{$id}->{'mac'} = $m->findvalue('./nic/macaddress');

つづく
```

Sunbit System

事前準備

- VMのリストを作る

```
        # volumeidのリストからこのvmのvolumeidをつなげる(複数あり)
        my %va;
        while( my ($volid,$v) = each(%volumes) ){
                my $vmid = $v->{'vmid'};
                # 順序がわかるようにデバイスIDをキーにしてハッシュにする
                if ( $vmid eq $id ){
                        $va{$v->{'deviceid'}} = $v;
                }
        }
        # デバイスID≒ディスク順なので順序通りvolumeがつながる
        $machines{$id}->{'volumes'} = \%va;

        # このVMに割り当てるvlanを取得する
        # %hostvlanは設計したvlanとvmの対応表
        $machines{$id}->{'vlan'} = $hostvlans{$id}→{'vlan'};

} # VMのリストを作るforeachのおわり
```

Sunbit System

事前準備

- これで変数%machinesが完成
  - VMごとに情報がまとまっている
  - ディスクが順序通り結びつく
  - MACとVLANの情報も持っている
- この処理はinit()と名付けよう

よく使いそうな部品

- CloudStack非同期処理の終了を待つ

```
sub waitAsyncJob
{
        my ($jobid) = @_;
        my ($xml, $status);

        while(1){
                $xml = XML::LibXML->load_xml(string =>
                        kickapi("queryAsyncJobResult","jobid=$jobid"));
                my $node = $xml->findnodes('//jobstatus');
                $status = $xml->findvalue('//jobstatus');
                last if ($status != 0);
                sleep(10);
        }
        if ($status != 1){ logprint("err"); return(undef); }
        return($xml);    # SUCCESS
}
```

Sunbit System

CloudStackからVMを抜き出す

- VMを停止する

```
sub stopVM
{
	my ($zoneid, $vmid) = @_;
	# kick_apiを利用して stopVirtualMachine を投げる
	my $xml = XML::LibXML->load_xml(string =>
		kickapi("stopVirtualMachine", "zoneid=$zoneid", "id=$vmid"));
	my $jobid = $xml->findvalue('//jobid');

	# stopVirtualMachineは非同期処理なので処理終了まで待つ
	$xml = waitAsyncJob($jobid);
	my $state = $xml->findvalue('//state');
	if ($state =~ "Stopped"){
		return(1);	# 停止成功
	}
	return(0);	# 結果を待ったが止まらなかった
}
```

Sunbit System

CloudStackからVMを抜き出す

- VMをexportする

```
sub extractVolume
{
        my ($zoneid, $volid) = @_;
        my $xml = XML::LibXML->load_xml(string =>
                kickapi("extractVolume", "zoneid=$zoneid",
"mode=HTTP_DOWNLOAD", "id=$volid"));
        my $jobid = $xml->findvalue('//jobid');

        # extractVolumeは非同期処理なので待つ
        $xml = waitAsyncJob($jobid);
        my $state = $xml->findvalue('//state');
        if ($state =~ "CREATED"){
                return(1);              # 取り出し成功
        }
        return(0);      # 失敗している
}
```

Sunbit System

XenServerにVMを作る

```
sub vminstall
{
        my ($vmid) = @_;
        my $cmd;
        my @r;

        # VLAN定義がない＝対象でないか設計漏れ＝どっちにしろ作らない
        if (!defined($machines{$vmid}->{'vlan'})){ return(0);}
        # VMを作成
        my $newvm_uuid = undef;
        $cmd = sprintf("ssh root@%s ". "xe vm-install".
                " new-name-label=%s sr-uuid=%s memory=%s template=%s",
                $opt{'targetserver'},      # XenServerへsshでリモートコマンド
                $machines{$vmid}->{'vmname'}, $destsr_uuid,
                $machines{$vmid}->{'memory'}, $tpl_uuid);
        @r = execshell($cmd);
        if ($r[0] =~ /^[0-9a-f]{8}/){
                $newvm_uuid = $r[0];    # uuidが戻ってきたら覚える
        }
つづく
```

Sunbit System

XenServerにVMを作る

```
# メモリを割り当てる
$cmd = sprintf("ssh root@%s ".
        "xe vm-memory-set uuid=%s memory=%d",
        $opt{'targetserver'},
        $newvm_uuid,
        $machines{$vmid}->{'memory'});

# このコマンドは正常終了で何も返さないので文字が戻ると負け
@r = execshell($cmd);
if (defined($r[0])){
        return(0);     # 負け
}
```

つづく

Sunbit System

XenServerにVMを作る

```
        # CPU数を割り当てる
        $cmd = sprintf("ssh root@%s ".
                "xe vm-param-set uuid=%s VCPUs-max=%d",
                $opt{'targetserver'},
                $newvm_uuid, $machines{$vmid}->{'cpunumber'});

        # このコマンドも正常終了で何も返さないので文字が戻ると負け
        @r = execshell($cmd);
        if (defined($r[0])){
                return(0);     # 負け
        }
```

つづく

Sunbit System

XenServerにVMを作る

```
        # VLANが作成済かを調べ、なければ作る
        my $newnetwork = undef;
        my $vlanname = sprintf("vlan%d", $machines{$vmid}->{'vlan'});
        $cmd = sprintf("ssh root@%s xe network-list name-label=%s",
                $opt{'targetserver'}, $vlanname);
        @r = execshell($cmd);
        if ($r[0] =~ /^uuid.*: ([0-9a-f\-]+)$/){
                $newnetwork = $1;      # もし既にあればそのuuidを使う
        }
        else {
                # VLAN用のnetworkを作る
                $cmd = sprintf("ssh root@%s xe network-create name-label=
%s",
                        $opt{'targetserver'}, $vlanname);
                @r = execshell($cmd);
                if ($r[0] =~ /^[0-9a-f]{8}/){
                        $newnetwork = $r[0];    # 新しく作ったらそれを使う
                }
        }
つづく (このあとVLANとvifを結びつけますが、省略します)
```

Sunbit System

XenServerにVMを作る

```
        # vdiを作成してvbdを接続する
        my @keys = sort(keys($machines{$vmid}->{'volumes'}));
        foreach my $k (@keys){
                my $newvdi = undef;
                my $newvbd = undef;
                $cmd = sprintf("ssh root@%s ".
                        "xe vdi-create".
                        " name-label=%s sr-uuid=%s virtual-size=%s device=%s",
                                $opt{'targetserver'},
                                $machines{$vmid}->{'volumes'}->{$k}->{'name'},
                                $destsr_uuid,
                                $machines{$vmid}->{'volumes'}->{$k}->{'size'},
                                $machines{$vmid}->{'volumes'}->{$k}->{'deviceid'});
                @r = execshell($cmd);
                if ($r[0] =~ /^[0-9a-f]{8}/){
                        $newvdi = $r[0];    #  VDIのuuidが作られる
                }
```

つづく

XenServerにVMを作る

```
            # vbdを作成し、VMとVDIを結びつける
            $cmd = sprintf("ssh root@%s ".
                  "xe vbd-create".
                  " vm-uuid=%s vdi-uuid=%s device=%s type=Disk",
                        $opt{'targetserver'},
                        $newvm_uuid, $newvdi,
                        $machines{$vmid}->{'volumes'}->{$k}->{'deviceid'});
            @r = execshell($cmd);
            if ($r[0] =~ /^[0-9a-f]{8}/){
                  $newvbd = $r[0];          # VBDのuuidが作られる
            }
      }  # VMごとのディスク数分foreach

} # vminstall() の終了
```

Sunbit System

ROOTディスクのインポート

```
# ルートディスクのインポート
sub importroot
{
	my ($vmid,$dev) = @_;
	my $cmd;
	my @r;

	my $volid = $machines{$vmid}->{'volumes'}->{$dev}->{'id'};
	my $uuid = $machines{$vmid}->{'volumes'}->{$dev}->{'vdi-uuid'};
	my $size = $machines{$vmid}->{'volumes'}->{$dev}->{'size'};
	my $name = $machines{$vmid}->{'volumes'}->{$dev}->{'name'};
	my $zoneid = $machines{$vmid}->{'zoneid'};

	# 実はここでextractVolumeをする
	if (!extractVolume($zoneid,$volid)){
		return(0);
	}


つづく
```

Sunbit System

ROOTディスクのインポート

```
        # extractVolumeしたイメージをインポートする
        $cmd = sprintf("ssh root@%s ".
                "xe vdi-import ".
                "uuid=%s filename=%s format=vhd",
                $opt{'targetserver'},
                $uuid, $path);
        @r = execshell($cmd);
        if ($r[0] =~ /Error/){
                return(0);          # 実行後の文字列にErrorがあると失敗
        }
        return(1);

}   ### importroot()の終了
```

Sunbit System

DATAディスクのインポート

```
 # 追加ディスクのインポート
sub importdisk
{
        my ($vmid,$dev) = @_;
        my $cmd;
        my @r;
        # folderのパスをマウントポイントに合わせる
        $folder =~ /\/(\w+)$/;
        $folder = $1;
        # コピー元ファイル名をフルパスで作る
        my $path = "$datamount/$folder/$filename.vhd";
        $cmd = sprintf("ssh root@%s "."xe vdi-import ".
                "uuid=%s filename=%s format=vhd",
                $opt{'targetserver'},$uuid, $path);
        @r = execshell($cmd);
        if ($r[0] =~ /Error/){ return(0); }   # Errorが返ったら失敗
        return(1);   # インポート成功
}
```

Sunbit System

並列処理を考える

- 1VMに最大で6つのディスクがある
  - 1TBインポートは最大10時間程度かかる
  - 直列処理していたのでは間に合わない
  - 移行リソースは十分にある
  - ROOTディスクのextractは、追加ディスクインポートと平行するほうが時間効率高い

Sunbit System

組み合わせる

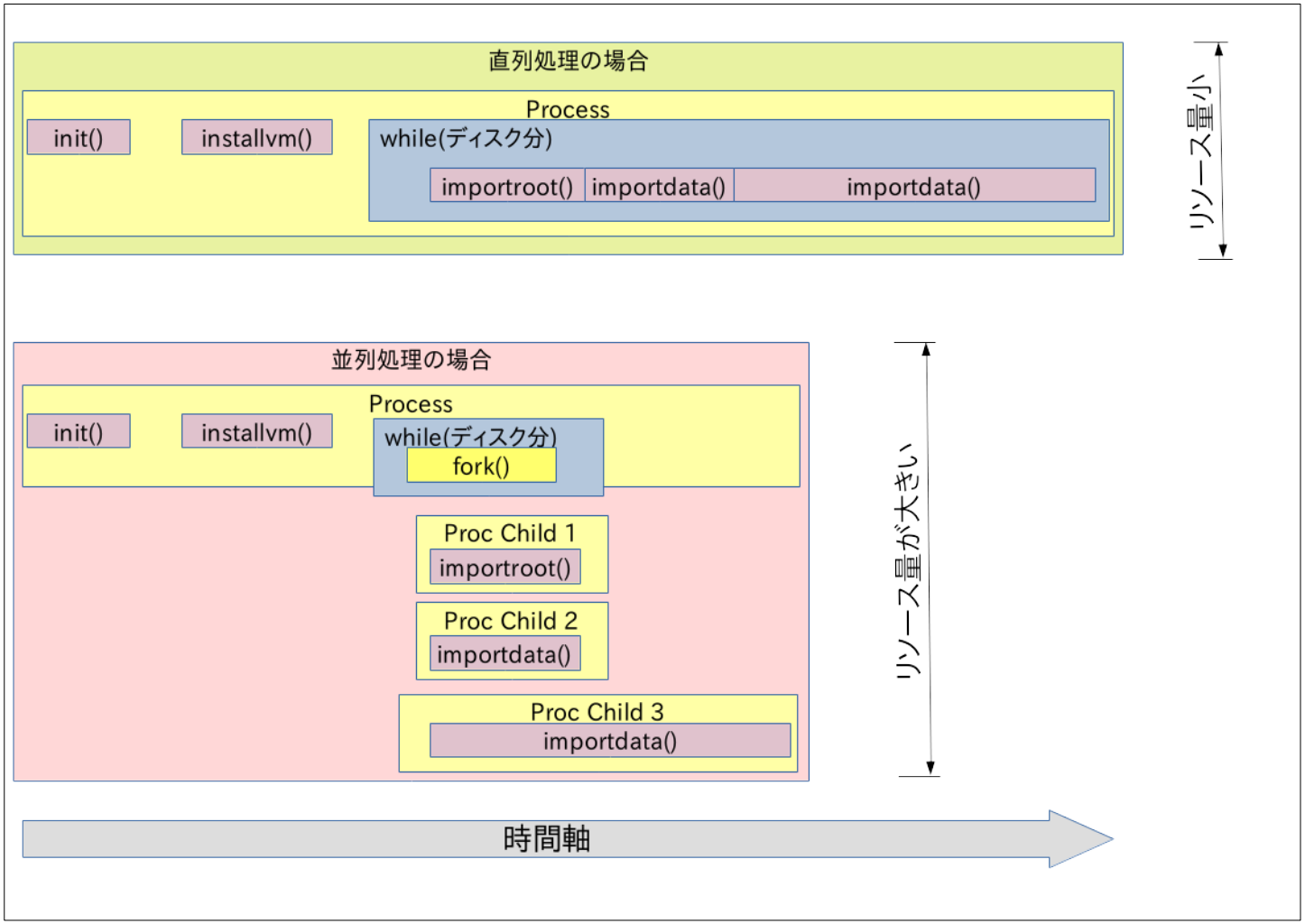
```
#!/usr/bin/perl

my $vmid = shift(@ARGV);    # 引数はVMID
init();                     # 初期化して
stopVM($vmid);      # 移行元のVM止めて
vminstall($vmid);   # XenServerにVMを作る。あとはインポート

# 対象VMに接続しているディスク分ループ
my $vols = $machines{$vmid}->{'volumes'};
while( my ($dev, $v) = each(%$vols) ){
        # forkして接続しているディスクを並列処理
        my $r;
        my $pid = fork();
        if ($pid){ next; }    # 親プロセスは次のディスクへ
        # インポート処理は子プロセスが行う
        if ( $v->{'type'} eq "ROOT" ){ $r = importroot($vmid,$dev); }
        else { $r = importdisk($vmid,$dev); }
        exit($r);
}
```

Sunbit System

直列処理 vs 並列処理



Sunbit System

スクリプト化の効果

- きちんとデバッグしておけば失敗しない
  - 手動だと間違いなく人為ミスが発生する
  - メタデータ不備の失敗があったがカバー範囲≒失敗への対処に余裕ができる
- 言葉で説明できるならスクリプトにできる
  - 時間が長いとか量が多いものほど効果が高い
  - 動的なデータが多いほど効果が高い
  - 書き方によってはすごい時間短縮
  - 言葉で説明するより正確

Sunbit System

書くべきは、手順書よりも、スクリプト

- 自分が使いやすいスクリプト処理系の使い方を身に着けておきましょう。
- 上手なデータ化もスクリプトを書く上で有効です。
- スクリプト化したほうが効率が良いのか判断できる能力は必要です。

- スクリプトを上手に使って、*サボれる*できるITエンジニアを目指しましょう。

Sunbit System 有限会社サンビットシステム

- 1998年8月設立
- 2009年より札幌市豊平区平岸
- 今年のフレーズ：オープンソースの活用で社会貢献
  - 港営業務システム「あまつみ®」の開発・運用
  - ITインフラ構築、ITシステム設計支援
  - ソフトウエア受託開発,パッケージ開発
  - IT教育など