OpenStackとPythonと私

日本仮想化技術株式会社
野津 新

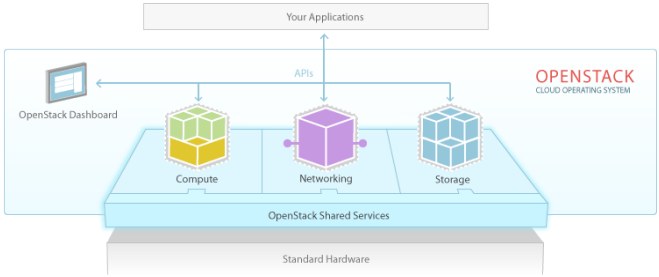
VirtualTech Japan
VirtualTech Japan
VirtualTech Japan

アジェンダ

- OpenStack
  - 概要
- Python
  - メッセージ処理を一部紹介
- 私
  - Bare-Metal Provisioning in OpenStack

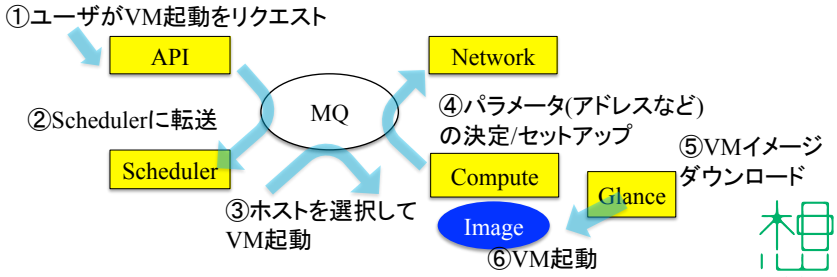
OpenStack概要

- クラウド構築ソフトウェア
  - クラウド=AWSみたいなもの
  - VM（仮想マシン）を利用者に提供
  - 主にPythonで記述
  - 複数の要素で構成
- Nova: VM管理(IaaS)
  - AWSのEC2相当
  - この話のメイン
- 他の構成要素
  - Quantum: ネットワーク
    - ユーザ(テナント)ごとにVMのネットワーク構成を定義
    - OpenFlowなどで実現
  - Swift: オブジェクトストレージ
    - AWSのS3相当
    - 単体での使用も可能
  - Horizon(Dashboard): Web UI



Novaの構成要素(プロセス)

- Compute: ハイパーバイザ制御
  - KVM,Xen,VMwareなど
  - ハイパーバイザに対応するクラスを定義
- Glance: イメージ管理
  - Swiftをバックエンドにできる
- Scheduler: ComputeへのVM割当
  - 特定のcomputeに負荷が集中することを防ぐ
- Volume/Cinder: ブロックデバイス管理
  - AWSのEBS相当
- Network:ネットワーク管理
  - Quantumと連携する場合もここで
- API: ユーザリクエストの処理
  - HTTPサーバ
- メッセージキュー
  - 各構成要素間のリクエストをキューイング



メッセージングの実装

- 新しくインスタンスを作成する場合を例に
- computeの nova.compute.manager.ComputeManager.run_instance() を実行すればいい
- ただし・・・
  - 呼び出し元はScheduler
  - Scheduler内で直接上記メソッドを呼び出しても無意味
  - →schedulerプロセスからcomputeプロセスへのRPC
- Schedulerで nova.compute.rpcapi.ComputeAPI.run_instance()を実行
- メッセージキューを経由して ComputeManager.run_instance()が実行される

リクエスト送信

- 呼び出し側はメッセージキューを（あまり）意識する必要なし
- メソッド名(run_instance)と引数をメッセージにしてメッセージキューに送信(cast)

```
def run_instance(self, ctxt, instance, host, request_spec,
                 filter_properties, requested_networks,
                 injected_files, admin_password,
                 is_first_time):
    instance_p = jsonutils.to_primitive(instance)
    self.cast(ctxt, self.make_msg('run_instance', instance=instance_p,
            request_spec=request_spec, filter_properties=filter_properties,
            requested_networks=requested_networks,
            injected_files=injected_files, admin_password=admin_password,
            is_first_time=is_first_time),
            topic=_compute_topic(self.topic, ctxt, host, None))

@staticmethod
def make_msg(method, **kwargs):
    return {'method': method, 'args': kwargs}
```

リクエスト受信

- 受信されたメッセージはcompute側のディスパッチャへ
- 委譲先オブジェクト(ComputeManager)のrun_instanceメソッドを呼び出す
  - 文字列→メソッドに変換(getattr)

```
def dispatch(self, ctxt, version, method, **kwargs):
                    （省略）
          if not hasattr(proxyobj, method):
            continue
        if is_compatible:
          return getattr(proxyobj, method)(ctxt, **kwargs)

    if had_compatible:
      raise AttributeError("No such RPC function '%s'" % method)
    else:
      raise rpc_common.UnsupportedRpcVersion(version=version)
```

run_instance本体

- ここではメッセージキューを意識する必要なし
- ハイパーバイザに応じた処理に続く・・・

```
nova.compute.manager.ComputeManager:
 @exception.wrap_exception(notifier=notifier, publisher_id=publisher_id())
 @reverts_task_state
 @wrap_instance_fault
 def run_instance(self, context, instance, request_spec=None,
                  filter_properties=None, requested_networks=None,
                  injected_files=None, admin_password=None,
                  is_first_time=False):

     （省略）
    @utils.synchronized(instance['uuid'])
    def do_run_instance():
      self._run_instance(context, request_spec,
              filter_properties, requested_networks, injected_files,
              admin_password, is_first_time, instance)
    do_run_instance()
```

Bare-Metal OpenStack

- OpenStackのインスタンスを物理マシン上で直接実行（Bare-Metal）
  - 通常OpenStackはVMインスタンスのみサポート
  - 仮想化オーバーヘッドが無く高速実行可能
- OpenStackを拡張した独自実装
  - インスタンス作成時にBare-Metalを選択して起動
  - OpenStack本体にマージされるよう調整中

B-M OpenStackの想定利用方法

- 物理・仮想のハイブリッド構成システム全体をOpenStackで一元管理
  - x86 VMと同じイメージを使用可能
- 高速処理が必要となったシステム、アプリケーションを物理マシン（Bare-Metal）上にプロビジョニング
- DR（災害復旧）サイトを柔軟に構成
- マイクロサーバー群をOpenStackで管理

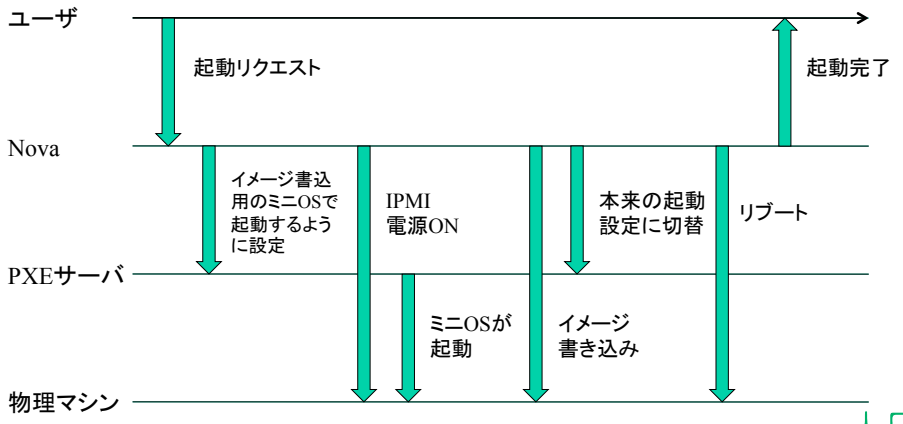
課題

- 物理マシンにはハイパーバイザがない
- ハイパーバイザが行う処理の代替が必要
- VMの起動は以下のフロー
  1. カーネル/起動RAMディスクの指定
  2. ディスクイメージ
  3. 電源ON
  4. 他ユーザネットワークとの干渉防止

対策

実装上の課題	対策/利用テクノロジ
カーネル/RAMディスクの切り替え	ネットワークブート（PXE）を利用
ローカルディスクへのイメージ書き込み	書き込み用のミニOSをPXE起動、完了後、書き込んだイメージで再起動
外部からの強制電源ON/OFF	IPMIを利用
ネットワークの分離	OpenFlowを利用

起動シーケンス



参考

- OpenStack
  - http://www.openstack.org/
  - https://launchpad.net/nova/ (BTSなど)
  - https://github.com/openstack/
- Bare-Metal OpenStack
  - https://github.com/NTTdocomo-openstack/