principes architecturaux du "cloud computing"

IAAS

OpenStack

En 2010 la Nasa et Rackspace scellent un partenariat afin de fonder le projet OpenStack, qui vise à fournir un gestionnaire IaaS sous une licence de logiciel libre. L'objectif est de fournir une collection de services qui peuvent être utilisés indépendamment, mais qui ensemble permettent la mise en place d'infrastructures de Cloud Computing. La première version d'OpenStack contenait deux services : le service Nova en charge de la gestion des machines virtuelles, dont le code venait de la plate-forme Nebula opérée par la Nasa, et le service Swift dont le code était issu du projet Cloud File Platform utilisé par Rackspace.

OpenStack

OpenStack est une collection de services, où chaque service s'occupe d'un aspect précis d'une infrastructure IaaS. Ces services sont relativement faiblement couplés entre eux,

permettant des déploiements modulaires où seuls les services basiques sont indispensables

(Nova, Keystone, Glance, Cinder).

OpenStack

OpenStack ?

Écrit en Python

Communication entre les composants

API Rest et RPC

AMQP (RabbitMQ ou QPid)

KVM comme premier choix d'hyperviseur

LXC pour des tests rapides

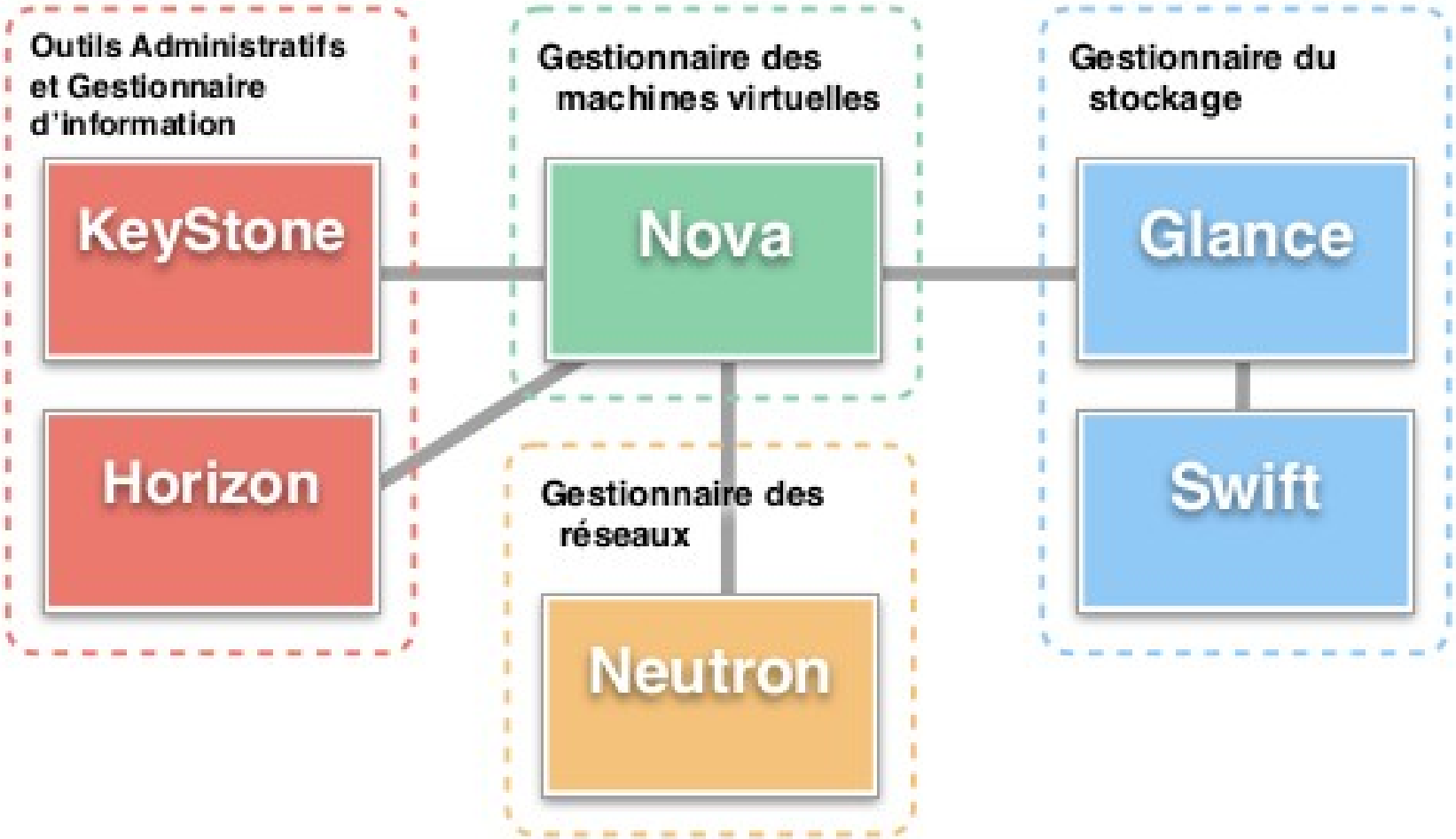
Xen/XenServer (dont XCP)

VMware/ESX

Docker

Baremetal

OpenStack



Architecture du système OpenStack.

OpenStack(Les principaux composants)

le rôle des six principaux services

les plus fréquemment déployés au sein d'une infrastructure gérée

par OpenStack :

• Nova : ce service s'occupe de la gestion des machines virtuelles

et de leur cycle de vie. Il sert aussi de liant avec les autres

services pour ajouter, aux machines virtuelles qu'il gère, de la

connectivité réseau et le support des images.

OpenStack(Les principaux composants)

Gestion des traitements

Responsable de la gestion des instances (lien fort avec la

Virtualisation) La gestion des ressources (par des quotas)

Possède la notion de zones de disponibilité

Personnalisables.Pour une réduction des risques

Nova regroupe de nombreux "projets internes" pour assurer ces

tâches comme novacompute

Glance : ce service s’occupe de la gestion des images des machines virtuelles.

La gestion des images

Stockage et manipulation d'un ensemble d'images (personnalisées) utilisables

-Lancement rapide

-Déjà configurées

-Intégrité vérifiée

-Nova supporte plusieurs systèmes de virtualisation => plusieurs formats d'images supportés par Glance

-Stockage multiple

-Disques locaux

NAS/iSCSI

Swift

Ceph

Cinder : ce service s'occupe de la gestion du stockage en mode bloc (Block DeviceStorage). Cette technique permet à une machine virtuelle d'avoir accès à un disque dur virtuel distant, afin d'y stocker des données persistantes.

-La gestion des volumes

Nouveau paradigme pour les données :

-Ephémère si non précisé

-Persistant à la demande, on parle de volume

C'est le role de Cinder

-Cinder est l'équivalent de EBS pour AWS Permet de stocker sur

-Systèmes de fichiers (sur plusieurs nodes)

SAN / NAS

Ceph

Swift : le stockage dans le cloud

Permet de gérer un stockage de plusieurs pétaoctets en 1 seul système

Assure la réplication et l'intégrité des données

Stockage de blobs bruts accompagnés d'attributs (nécéssité de xattr)

Maintenance d'un mapping entre nom et localisation (the ring)

Notion de zones (similaires à nova)

Personnalisables

Réplication (par défaut) dans 3 zones différentes

Subit la concurence de Ceph (projet non OpenStack mais très utilisé)

Neutron : ce service est chargé d'assurer la connectivité entre machines virtuelles à travers la création de réseaux virtuels. Ce service utilise les techniques des réseaux définis de manière logicielle (Software Defined Network [26] - SDN). Il est possible de se passer de Neutron en utilisant le sous-service Nova-network, cependant ce dernier tombe en désuétude.

- La gestion avancée du réseau

-Neutron constitue la nouvelle génération de gestion du réseau sur OpenStack

-Propose une API unifiée qui est capable de piloter plusieurs architectures réseaux

Linux Bridge

Open vSwitch

Cisco (matériel Nexus)

NSX (VMWare) ...

-Permet la création de réseaux complets et autonomes par tenant

Base de fonctionnalités avancées

- LBaaS
- DNSaaS
- FWaaS

Keystone : ce service est responsable de la sécurité et de l'authentification au sein d'un déploiement OpenStack.

Keystone est le point d'entrée dans son cloud centralise l'authentification

liste des services et endpoints gère les différents rôles

Horizon : ce service offre une interface graphique aux utilisateurs, qui leur permet de manipuler les ressources que ceux-ci hébergent sur une infrastructure OpenS-tack. Il est possible de se passer d'Horizon en utilisant l'API Rest d'openStack, ou au moyen d'une interface en ligne de commande.

Pourquoi choisir OpenStack ?

- Création de cloud privé
- Proche d'une solution de virtualisation sur mesure (avec tous les composants et possibilités offertes)
- Multi hyperviseurs dans un seul environnement
- Gestion fine des lieux d'éxecution (ou pas)
- Haute disponibilité
- Scalable par conception
-

Pourquoi choisir OpenStack ?(suite)

- Load balancing natif
- Souplesse/facilité des migrations pour permettre une bonne gestion des hotes
- Ajout/intégration simplifié de nouveaux hosts
- Décomposition native en projets simplifiant les permissions
- Possibilité d'avoir une gestion avancée et fine du réseauPourquoi choisir OpenStack ?
- Le futur avec le cloud hybride
- Passage simple vers le cloud public (API communes)

Eucalyptus

En 2009 des chercheurs de l'Université de Californie publient "The Eucalyptus Open-source Cloud-computing System" qui s'intéresse aux problèmes inhérents au modèle des grilles de calcul : les auteurs analysent que pour les très gros besoins en ressources de calcul, il est nécessaire d'agréger les ressources de plusieurs fournisseurs, ce qui se traduit bien souvent par une hétérogénéité matérielle. Cette hétérogénéité doit être prise en compte par les concepteurs de service, ce qui nécessite des compétences techniques avancées.

Eucalyptus

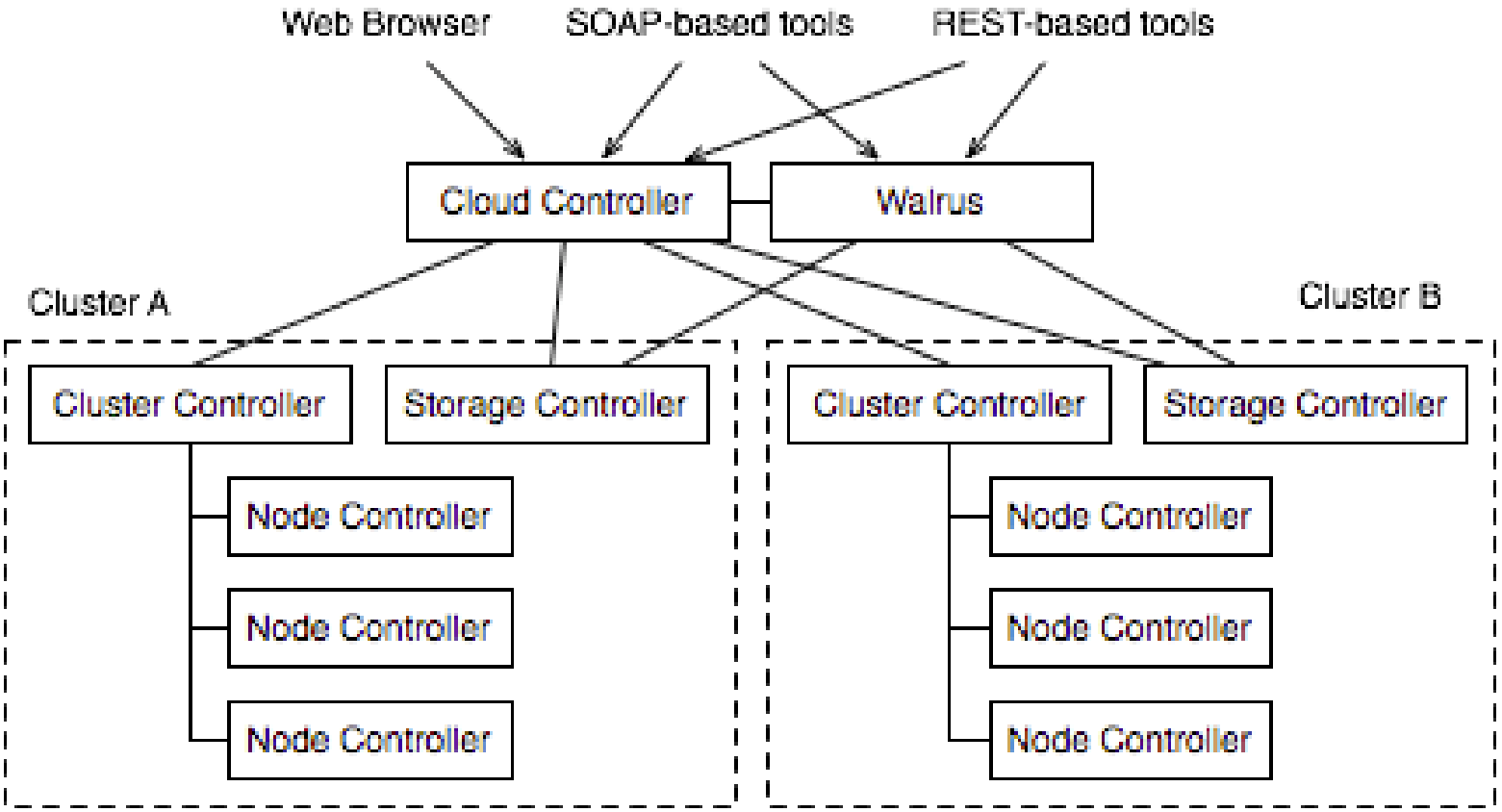
Les auteurs proposent alors de mettre en place un accès uniforme à cette collection de ressources, en les fournissant aux utilisateurs sous la forme de machines virtuelles.

L'infrastructure sur laquelle sont déployées ces machines virtuelles est gérée par le système Eucalyptus qui est défini par ses auteurs comme un cadriciel (framework) logiciel permettant de transformer des ressources de calcul et de stockage d'une organisation en ressources de Cloud Computing.

Eucalyptus

Le système Eucalyptus est un gestionnaire IaaS conçu autour d'une architecture mo-
dulaire, où chaque composant est responsable d'une fonction précise de l'infrastructure IaaS, tout en restant indépendant des autres composants. Chacun des composants d'Eucalyptus est rendu accessible aux autres composants sous la forme d'un service web. Une
infrastructure IaaS, utilisant Eucalyptus, est organisée de façon hiérarchique : une infra- structure de grande taille sera divisée en plusieurs groupes de nœuds (clusters) de tailles
inférieures et où chaque serveur de l'infrastructure (c.-à-d. nœud) est spécialisé comme la figure suivante :

Eucalyptus



architecture d'Eucalyptus

Eucalyptus

• **Cloud Controller :** sert de point d'entrée pour communiquer avec l'infrastructure
IaaS. Il expose une API web qui peut être utilisée pour manipuler l'infrastructure
Sous-jacente.

Cluster Controller : est déployé dans chacun des clusters de l'infrastructure. Ce type de nœud surveille l'exécution des machines virtuelles au sein du cluster dont
il est responsable. Il reçoit des ordres de création de machines virtuelles du Cloud Controller et s'occupe de l'ordonnancement en choisissant le Node Controller qui
les hébergera.

Eucalyptus

- **Node Controller :** est responsable du contrôle et de la surveillance des machines virtuelles qu'il héberge. Il reçoit des ordres de création de machines virtuelles du Cluster Controller.

Walrus (Storage Controller) : est en charge de la fourniture d'un service de stockage utilisant les mêmes interfaces logicielles que le service Amazon S3. Walrus est principalement utilisé pour le stockage des images des machines virtuelles ainsi que les métadonnées des utilisateurs.

Eucalyptus

Chaque composant de haut-niveau du système a sa propre interface web et est implémenté comme un service web indépendant. Cela a deux principaux avantages :
Chaque service web expose une API bien définie et indépendante du langage, sous la forme d'un document WSDL contenant à la fois les opérations que le service peut effectuer et les structures de donnée d'entrée/sortie.

Eucalyptus, utilise avec avantage les fonctionnalités existantes dans les services web telles que les polices de sécurité (WSS) pour la communication sécurisée entre les composants, et se base sur les logiciels de services web se conformant aux standards industriels.

Eucalyptus

Concernant les communications entre machines virtuelles, Eucalyptus propose de connecter ces dernières à travers un système de réseau virtuel, défini de manière logicielle en se basant sur les techniques de réseau d'overlay (Overlay Network). Cela permet à des machines virtuelles qui n'appartiennent pas à la même organisation d'être isolées. Cette technique de virtualisation du réseau entrainant un léger surcoût (un saut réseau supplémentaire via l'ordinateur hébergeant la machine virtuelle.), le choix est donc laissé à l'utilisateur entre privilégier l'isolation de ses ressources de calculs ou de reposer sur un réseau plus conventionnel.

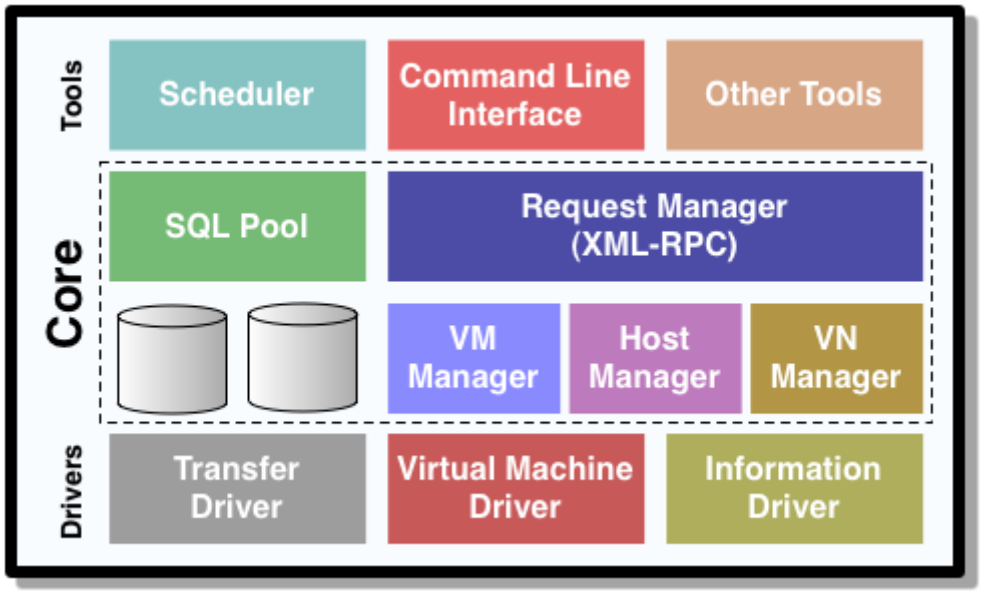
OpenNebula

Parallèlement à l'apparition du projet Eucalyptus, le projet OpenNebula émerge en 2009
et est le fruit de la collaboration de chercheurs issus de l'université de Chicago, de l'université de Madrid ainsi que du laboratoire d'Argonne proposent le projet OpenNebula

l’équipe du projet OpenNebula estime qu’il faut rendre possible et même faciliter la mise en place d’infrastructures IaaS qui soient exploitées par plusieurs fournisseurs. Dans cette optique, le projet OpenNebula propose un gestionnaire IaaS qui se place à un niveau d’abstraction plus élevé que les projets d’alors tels Nimbus et Eucalyptus, afin de faciliter la fédération d’infrastructures appartenant à plusieurs fournisseurs.

L’absence de standards ouverts pose le problème de la fédération d’infrastructures qui ne possèdent pas les mêmes APIs, voire qui ne partagent pas toutes les mêmes fonctionnalités. OpenNebula a été conçu pour être nativement ouvert à l’utilisation de ressources issues de fournisseurs externes, grâce à un système de pilotes logiciels externes (Cloud

Drivers). Cette approche permet à OpenNebula de pouvoir piloter les ressources de calculs, qu'elles soient hébergées localement ou dans des infrastructures distantes, du moment que ces dernières sont rendues disponibles via un Cloud Driver. Initialement, les infrastructures externes supportées étaient Amazon EC2, Eucalyptus et ElasticHosts.



F IGURE 2.4 : Architecture du projet OpenNebula

OpenNebula internal architecture can be divided into three layers:

Tools, management tools developed using the interface provided by the OpenNebula Core.

Core, the main virtual machine and host management components.

Drivers, to plug-in different virtualization technologies into the core.

Tools

This layer contains tools shipped with OpenNebula, such as the CLI or the scheduler, but also 3rd party tools that can be easily created using the XML-RPC interface or the ONE client API.

Command Line Interface

A command line interface is provided with OpenNebula to manually manipulate the virtual infrastructure under ONE.

Scheduler

The Scheduler is an independent entity in the OpenNebula architecture, so it can be easily tailored or changed since it is decoupled from the rest of the components. It uses the XML-RPC interface provided by ONE to invoke actions on virtual machines. The scheduler distributed in the Technology Preview provides the following functionality: User-driven consolidation, the scheduler assigns VMs to those hosts that meets the capacity requirements of the VM (that can be adjusted to consolidate VMs).

Matchmaking policy, to schedule a VM those resources that do not meet the requirements (boolean expression) are filtered out. Then a ranking process is made to select the best Host. This is done by applying the ranking expression defined in the VM template. The scheduler is built on a policy template, this template can be used to further tailor the Scheduler behavior.

OpenNebula Core

The core consists of a set of components to control and monitor virtual machines and hosts. The core performs its actions (e.g. monitor a host, or cancel a VM) by invoking a suitable driver. The main functional components of OpenNebula core are:

Request Manager, to handle client requests

Virtual Machine Manager, to manage and monitor of VMs

Host Manager, to manage and monitor physical resources

Database, persistent storage for ONE data structures

Request Manager

The Request Manager exposes a XML-RPC Interface, and then depending on the invoked method a given component is called internally. The XML-RPC decouples most of the functionality in the OpenNebula core, from external components i.e. the Scheduler.

Virtual Machine Manager

This component is responsible for the management and monitoring of VMs. The operations of the VM Manager are abstracted from the underlying hypervisor the use of plugable drivers.

Host Manager

This component manages and monitors the physical hosts. Monitor and management actions are performed also through a suitable driver. The host monitoring infrastructure is flexible and can be extended to include any attribute

Database

A persistent generic pool based on a SQLite3 backend is the core component of the OpenNebula internal data structures. This component provides ONE with the scalability and reliability (in case of failure the state of OpenNebula is automatically recovered) needed in the management VMs.

Drivers

OpenNebula has a set of pluggable modules to interact with specific middleware (e.g. virtualization hypervisor, file transfer mechanisms or information services), these modules are called Middleware Access Drivers (MAD). The communication between the core and the drivers is performed using a simple ASCII Protocol, which simplifies the development of new drivers.