CloudStack to a loosely-coupled component oriented distributed architecture

Conceptually, current CloudStack works very much like an JaveEE application server container, given the scale of all our current cloud deployments with CloudStack, I think CloudStack has done a great job.

Running as an application server container, it brings us a lot of benefits like easy to deploy, efficient resource utilization, etc, but in the mean time, as modules inside a container are too easy to be involved tightly, over time, as we add more and more features, the complexity of overall system increases dramatically. Moving forward, it makes a lot of sense to reshape CloudStack from a module container to be a component oriented distributed platform.

Component

Component is an independent deployment unit in a distributed environment(i.e., CloudStack), it usually runs as a separated process, can be independently scaled and has independent communication endpoint. It should have simple bindings to the distributed environment instead of requiring a complex container, to communicate with other components in the system, it should use bi-directional principal for loose-coupling and service-oriented, which is, using light-weight messaging event notification in one direction and service-oriented interaction at another direction.

Module

Module is a logic software unit that encapsulates software functions with well defined programming interface, a component contains one or more software modules

Component Service

Software service that is provided at component level, usually in shape of web service or REST-ful service

Messaging

A process for components to notify each other by exchanging messaging events

Messaging event

A package of information that is uni-casted or broadcasted to the destination component(s)

Service discovery

A process for component to discover services and endpoint bindings within the environment, it could be either through a directory service provided at infrastructure level or through auto-discovery

Under component oriented principal, at high level CloudStack core can be fine-tuned to a number of (not limited to) components

API component

Implements CloudStack API, it also initiates logic orchestration flows (i.e., deploy a VM) which will be notified and picked up in Orchestration engine component

Orchestration Engine component

This components is the heart to drive all async-flows currently active in the system, it orchestrates high-level flow through messaging among networking /storage/compute components.

Data service component

This components provides the data service for objects within the Cloud.

Networking component

A component to implement networking provisioning

Storage Component

A component to implement storage provisioning

Compute Component

A component to implement hypervisor VM provisioning

Fabric Monitoring Component

A component dedicated for monitoring on fabric resources, for example,

monitoring of hypervisor host status

Fabric Admin Component

A component that implements fabric resource administration.

Service Framework Component (module?)

A component(or a module inside Orchestration Engine?) to provide service to launch component service in a VM and auto-scale the component service.

At middleware level, we will need a messaging event delivery platform and middleware level synchronization system, possible candidates could be Redis and Apache ZooKeeper.