

FluidCloud: An Open Framework for relocation of Cloud services

Andy Edmonds
ZHAW

Thijs Metsch
Intel Corp.

Abstract

Cloud computing is about bringing the users new levels of being connected, instead of the once disconnected PC type systems. This proposal extends that level of connectedness in the cloud so that cloud services hosted by providers can relocate between clouds. This paper will demonstrate that one cloud service provider will be able to connect with another and relocate services between each at the command of the service owners. In this world of competing cloud standards and software solutions, each partially complete, this paper central research question is:

”How to intrinsically enable and fully automate relocation of services between clouds?”

1 Overall Comments

- We may need to be careful about the use of “service” - it should be qualified e.g. service instance or service type or service provider
- structure wise we need:
 - problem
 - vision/concept
 - scenario
 - solution
 - evaluation

2 Abstract

state problem, why it's interesting/relevant solution achieves, what follows - 4 sentences

3 Introduction

Summarise problem and vision?

(Problem) Today, cloud computing Grance,2011 services have little means to easily move from one cloud service provider to another. Standards are seen to be the panacea, yet have little adoption by the market, especially by the dominant forces.

(vision) Having the ability for a cloud service to easily and seamlessly move from one provider to another will bring a huge competitive advantage to any cloud computing service user. It would bring liberation to cloud services, both their application and data. Essentially, it will bring service “movement rights” to the cloud (AE: argument to make: however there is no means to accomplish this).

(Problem) So today's cloud standards exist but are not adopted. Also, even the most relevant standards are limited. Software libraries and frameworks that abstract cloud computing services to common interfaces are more widely adopted (examples). *Yet for cloud services those standards or software can manage, they do not contain any concepts to or mechanisms of relocating those services.* Ultimately, those cloud service instances remain locked under the control of the service provider.

4 Problem Statement

(we're in the solution space here already)

(scenarios) To address this let us consider some examples which demonstrate the need for a Framework such as FluidCloud:

- Cloud Service Developer (CSD, e.g. University startup). Take the example where a university startup implements a new service in the cloud. The type of service is one that is architected to handle bursty traffic as described in Elson,2008. After a period of time, the selected provider does not satisfy from technical (e.g. Amazon EC2 outages), economical (e.g. Google increasing prices for Google app engine) or regulatory purpose, due to service

offer changes. Now the benefits from a FluidCloud concept is that the startup can easily relocate their service to a new cloud service provider.

- Cloud Service Provider (CSP; e.g. CloudSigma, ProfitBricks). The cloud service provider operates FluidCloud software and offers it as a service. It is used to relocate new end-user services to their services from other, potentially, competing (through hardware differentiation - for example faster storage through SSDs) services. This would allow to CSP to let their customers seamlessly relocate to their service offering.
- Cloud Broker (e.g. Spotcloud or Zimory). Developers and providers of cloud brokerage services and software could consider adding cloud service relocation functionality to their cloud brokerage offers. Typically a cloud broker discovers and provisions the right cloud service on the behalf of the end-user. Once that target service is provisioned the end-user interacts and uses the target service, either through interfaces provided by the cloud broker or directly using the interfaces of the provisioned target cloud service. The cloud broker is aware of the end-users services and can continually watch for compatible services that can be offer to the end-user as a replacement, based on economic/geo-location/cost reasons. If the end-user was interested the cloud broker can relocate the service on the end-users behalf.

(*solution & key contributions*) For such scenarios, to be technically realised there is a set of missing technologies. FluidCloud fills these gaps such as:

- Service Relocation – *Ensuring the overall orchestration and process of moving a cloud service from the source to the target cloud service provider.* There are two types of cloud services that FluidCloud will support and enable relocation for: IaaS and PaaS based services. This entails that IaaS entities such as virtual machines and virtual storage devices (block and object storage) will need to be relocated. For PaaS, entities such as the service-components with packaged or compiled source code and possibly the data will need to be relocated. The decision to relocate will be something initiated by the owner of the service (e.g. through a user interface). The relocation may also be initiated by a Cloud Broker who has user-supplied policies to dictate under what conditions (such as costs) a service should be relocated.
- Service Adaptation - *The Conversion, transformation and movement of the service and its related data.* Related to relocating IaaS and PaaS

services are the potential service adaptations that need to take place. Services comprise of service-components (For example a web server, database), which are essentially applications. Those service-components might need to be changed when relocated. For example VM images need to be converted or if we turn our attention to the PaaS area is the adaptation of applications written for a certain target platform.

- Data Relocation - *Relocation, migration, transcoding, transformation and conversion of the data belonging to the service.* In FluidCloud a service is defined as a set of service-components and the data belonging to the service. Relocation of data fundamentally means moving bits and bytes. Currently tools such as GlobusOnline provide a service for easy transferring data between Grid sites using the proven GridFTP protocol, which allows for fast and reliable data transfers. Other solutions like the ZFS send/receive feature allow for snapshotting a dataset in constant time and allow easy relocation. Certainly upcoming technologies like Software Defined Networking can help when data path are needed on-demand to be established between providers.

5 Cloud Fluidity and FluidCloud (*vision and solution*)

(*vision*)

This might be better moved before the use cases? A future InterCloud is described in (Bernstein, Ludvigson, Sankar, Diamond, & Morrow, 2009), (D. Bernstein, 2010), (Y. Demchenko, 2012) as well in a notable online conversation where James Urquhart, Ciscos Cloud Programs and Communications Manager, clearly outlined the genesis and progression of it from singular and multi-cloud ecosystem we see today. The concept of the InterCloud is based on the proliferation and continued growth of public clouds ranging from IaaS, PaaS and up to SaaS. The ecosystem of these cloud service providers include the popular Amazon EC2, Google App Engine, CloudFoundry, OpenShift, Rackspace, Heroku and CloudSigma.

Within the high variety of service providers in this InterCloud concept, it becomes crucial that Service can be relocated/adapted. In the vision of the FluidCloud framework, the hosting cloud provider of your service is not a concern anymore given that the service can be fluidly (easily, on-demand and dynamically) relocated between providers.

The FluidCloud concept addresses multiple benefits – including:

- Supporting and Enabling the InterCloud: FluidCloud advances the definition, architecture and implementations of cloud computing, yet for stakeholders makes the transition easy through the an open source framework.
- Economical: FluidCloud can suggest new compatible services and based on economical differentiators and should the service owner want it; relocate their service to the suggested target provider.
- Regulatory: If a running cloud service is in an unrecognised or risky geographic region, a region where data privacy policies are incompatible or change to being incompatible then to comply with regulatory policy or law, the service provider can use FluidCloud to relocate that service at risk.
- Liberation: Cloud service developers and operators own their application and are responsible for end-user data. They should have right of movement for those services and the efficient means to enable them.
- Positive Market Disruption. By having the ability to relocate a service from one provider to another, the market place is opened further, enabling greater competition based on service differentiation and not on technical lock-in or limitation.

(solution)

Service in the Cloud are a wide field because their implementation can use IaaS or PaaS for example. Therefore FluidCloud addresses:

1. Relocation of IaaS-based service. FluidCloud will show the relocation of a service (and its data) running within VMs (on a local development machine, private end or public cloud). Triggers for the relocation can be scaling, costs, dependability, hardware differentiation or geo location. The service will automatically be adapted to the new environment.
2. Relocation of PaaS-based service. The relocation of a PaaS based service and its data between providers. Key to this demonstration is the ability to adapt the service (on source code level) and convert it to the new environment. Motivation should be to bring PaaS services from closed environments (e.g. Google App Engine) into more open ones (e.g. CloudFoundry).
3. Service adaptation: IaaS to PaaS. A software developer has developed a service on his own VM in an (Private) Cloud now he wants to roll-out this service on a available PaaS provider such as Google App Engine.

6 Architecture Implementation and Evaluation (*solution: more details*)

Core to realising FluidCloud are the following components: The CloudConduit handles and coordinates the overall (partial) relocation of the service. Viaducts form a path between cloud providers (if needed with underlying support of the network). The CloudConduit is responsible for setting up the Viaduct. Within Viaducts, Migrators are placed on these to adapt the Application, its environment and if necessary data as it relocates through the Viaduct. In order to relocate a service efficiently the CloudConduit can establish multiple Viaducts. The CloudConduit analyses the service to be relocated and based on that it uses the Broker to find suitable replacement providers. Based on the replacement providers the CloudConduit uses the Broker again to find suitable Migrators to aid the relocation process. The following diagram shows the proposed high-level proposal:

The key components are the following:

- **Service** - a logical container that comprises the application and the data. Executing the applications will form the workload.
- **CloudConduit** - orchestrates the process, introspects the services (application topology) to be relocated. It is also responsible for the lifecycle of Migrators. The relocation is triggered with this module.
- **Broker** - discovers and provides both cloud provider services and Migrator facilities. Eventually it monitors and inspects the workload of the services. Also orchestrates the deployment of the Migrators.
- **Migrator** - these are the libraries and services for adaptation and carry out one specific task related to (partial) relocation of services. Multiple Migrators might be needed to carry out the overall relocation.
- **Viaduct** - a logical path between two parties in which Migrators are organised. All together those accomplish the task of relocating a service from one cloud provider to another. Migrators as well as more network-oriented components (like proxies) are located on this path. All those components might be organised as workflows with multiple pipelines if needed.
- **Cloud Service Provider** - a provider of either IaaS or PaaS services.

With these concept in mind FluidCloud will provide the architecture and tooling to enable the InterCloud by enabling the three main topics introduced earlier: Service Relocation, Service Adaptation and Data Relocation.

7 Implementation (solution:lowest level of details)

As a first Proof of Concept (PoC) the conceptual architecture previously has been implemented using the Python programming language. Each of the components is a standalone process which eventually communicate with each other using a messaging queue. The prototype uses the Advanced Message Queuing Protocol (AMQP) implementation by RabbitMQ.

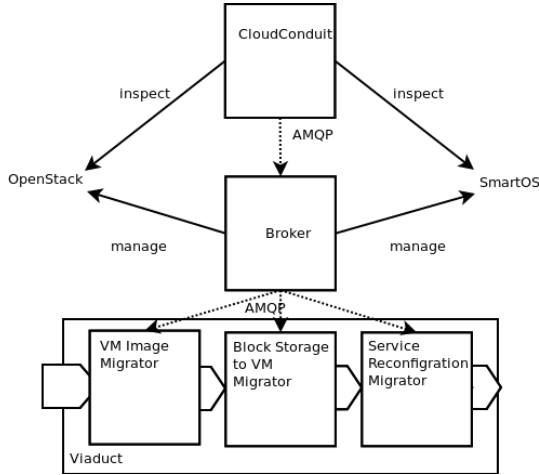


Figure 1: Architectural Overview

The CloudConduit has capabilities to processes requests for services to be migrated. When such a relocation is triggered it inspects the services for subcomponents and their dependencies. This is done an RESTful Cloud API which is implemented by both Cloud Providers (OpenStack and SmartOS based) in this PoC. Based on the inspection it creates a set of tasks which need to be executed. For this early setup the task are executed in sequential order. Later on the scheduling of these tasks may become more complex. The distribution of the tasks is handled by the Broker.

The Broker has not the information to instantiate the appropriate Migrators which make up the Viaduct (*need to note that the broker has a list of suitable replacement services*). The Migrators now take care of the actual relocation and topology change within the service.

To test this implementation the following scenario is considered: A simple node.js application is deployed with an Virtual Machine running within the Domain of OpenStack. This Virtual Machine has a block storage attached to it through an OpenStack Volume. The node.js also makes use of the Object Storage provided by OpenStack Swift.

Based on the performance of the Hypervisors the decision is made (*by the service owner*) to trigger a relo-

cation to a Cloud provider which uses SmartOS as an Hypervisor. After relocation the VM will be running on the SmartOS platform. The data within the block storage will be relocated whereas the data in the object storage will stay where it is. This will demonstrate that the Service topology of the service can change after the relocation depending on the new destination service provider. This change in topology although is done automatically. The service topology before and after relocation is shown in the following diagram:

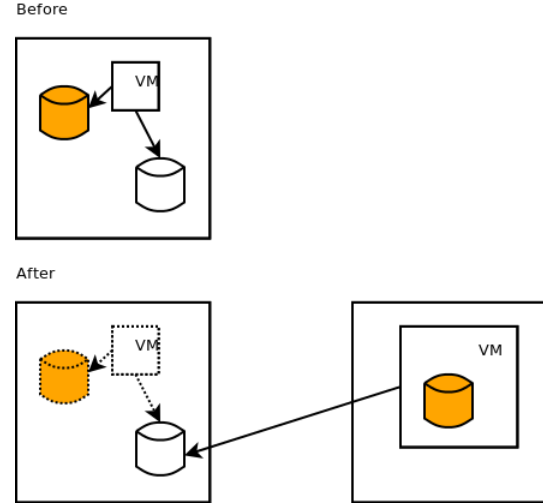


Figure 2: Service before and after relocation

The decision for this service topology after relocation is made by the CloudConduit and should be guided by service owner policies. Overall, to relocate this simple node.js application the following Migrators need to be placed on the Viaduct:

- OpenStack to SmartOS image converter (Hypervisor level) – Snapshots an KVM image within OpenStack Glance and copies it to the target platform. Essentially converts the image format in between.
- Relocator for the OpenStack block storage. Copies the data from the block storage device in OpenStack cinder onto the Filesystem of the VM running on KVM in SmartOS (which needs to be provisioned at this stage).
- Reconfiguration of the node.js application's configuration file based on regular expressions. Change paths for node.js interpreter, and network configuration needed (IP of OpenStack Swift instance).

(*should note that the object storage is not migrated - perhaps as it's not provided by openstack it's S3*)

When the relocation of the data parts (the VM image) is accomplished the Broker restarts the Virtual Machine on the target side.

This then overall demonstrates the earlier described process:

1. the CloudConduit gets a request to migrate an application
2. an application (*application versus service instance*) is a set of related VMs that make up the service
3. each of those VMs may have associated resources - CloudConduit must determine these

7.1 Evaluation (*solution: results*)

TODO

- It works :p
- Time to migrate depends on data payload sizes

The Architecture described in the last section should demonstrate that the concepts of FluidCloud are technical feasible. More over the important thing is that the concepts noted in this paper describe a way of enable fluidity of services between clouds.

Note: further work and idea of FluidCloud: SDN, Data migration etc.

8 Related Work

Standards organisation defined interfaces (e.g. OCCI or CDMI) might realise interoperability from a user perspective but they do not solve the issue of relocation. The paper (Petcu, 2011) reviews aspects related to portability and interoperability in clouds. It notes the lack of adoption of standards by vendors saying that “vendor[s] like[s] to put barriers to exit for their customers”. Related thoughts are discuss in (Jr, 2011). Here it is noted that cloud systems utilising different hypervisors wont interoperate, in part because they dont use the same data formats.

Adapter libraries enabled the means to manage multiple cloud offerings, The most prominent of these are libcloud , fog.io and jClouds. There are commercial products available currently that aim to easily use multiple clouds at runtime. Such examples include RightScale and Enstratus . However these solutions only manage the lifecycle on each various cloud service provider’s platform and they do not allow for the relocation of cloud services between providers.

There are quite a number of Platform as a Service (PaaS) offerings available today. Most PaaS offerings tend to support multiple languages and a number of supporting services. Examples of these include Heroku , Windows Azure, Red Hat OpenShift , VMware Cloud-Foudry and Google App Engine. These platforms can

be found to support Java, Python, Erlang, Node.js and other languages. What is important to consider with all of these offerings is the degree of lock-in that each offering brings to its users. The majority of the PaaS offerings leverage the existing interoperability work that each language (and its standard libraries - e.g. WSGI for Python) and supporting services (e.g. MySQL , RabbitMQ) already has. However, this is not uniform across all PaaS offerings.

The Open Data Centre Alliance (ODCA) released a report (Alliance, 2012) on long distance service relocation. Noted within is that relocation of workload possible but [] migrations occur between disparate data centres of the same cloud provider [] most of the time. Also noted within is the fact that the [] abstraction of a PaaS workload is not yet finalised and that more collaborative efforts on this issue need to be made. So here the most obvious issues become clear: relocation is possible but mostly within service providers domain and that inter-domain (InterCloud) relocation on IaaS and PaaS level needs more research.

Also in Mizgier, 2010 a model and means to evaluate the migration of one system to another. It notes that aspects related to the migration of data are on of the biggest challenge and, as particular to this work, where data model conversion must be carried out.

Currently available software solutions for data management exists such as: Cloudant[^somesamplefootnote], Xeround[^somesamplefootnote], Mongo-Lab[^somesamplefootnote] or Amazon S3[^somesamplefootnote]. But currently they lack the ability to convert data between the services, or relocate data. It is noted that services like Cloudant offer means of distributing data location based. Some means of dealing with the relocation of IaaS-based services (composed of Virtual Machines) is available today.

The paper Ward,2010 looks at InterCloud more from the federation aspect and the authors describe their architectural vision of that InterCloud. One important aspect that the authors do note is the importance of cloud brokers in their architecture. The concept of cloud brokerage is compliant with the definition provider by Gartner of Cloud Services Brokerage Plummer,2011 offering aggregation, integration and customisation, the 3 primary roles expected of such a cloud service broker. This is further refined in the NIST Definition of Cloud Computing Grance,2011.

9 Conclusions and Further Work

TODO: what follows from your solution

- investigate the area of live relocation etc.

- this is a large challenge - quote vmotion limitations
- investigate data payload minimisation
- inputs into stds bodies
- work will continue
- targets are

The Framework used for this PoC will be released and supported under an Open Source License (Apache 2.0) at: github.com/???

References

- [1] tbd Elson,2008: “Handling Flash Crowds from Your Garage” Mizgier,2010: tbd Ward,2010: C. tbd Plummer,2011: Daryl C. tbd