FluidCloud: An Open Framework for Relocation of Cloud Services

Andy Edmonds
ZHAW

Thijs Metsch
Intel Ireland Limited

Dana Petcu
Institute e-Austria Timisoara

Erik Elmroth
Department of Computing Science, UmeåUniversity

Jamie Marshall
Prologue

Plamen Ganchosov
CloudSigma

Abstract

TODO: more arguing about why it is important to be able to move things around

Cloud computing is about bringing the users new levels of being connected, instead of the once disconnected PC type systems. The proposal in this paper extends that level of connectedness in the cloud so that cloud service instances 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 service instances between each at the command of t he service owners. In the cloud world of competing cloud standards and software solutions, each partially complete, the central research question which this paper intends to answer:

How to intrinsically enable and fully automate relocation of service instances between clouds? TODOs:

- Page 5: The formulation "noted within" used as several places seems a bit odd to me.
- Abstract should highlight

1 Introduction

Today, cloud computing [1] service instances have little means to easily move from one cloud service provider to another. Cloud standards are seen to be the panacea, yet have little adoption by the market, especially by the market's dominant players.

Currently, cloud standards exist but most are not widely adopted, especially *de jure* style. *De facto* style standards such as Amazon EC2 has somewhat more adoption. Software libraries and frameworks that abstract cloud computing services to common interfaces are more widely adopted (see Related Work). However, even the most relevant standards or software libraries have little or no service instance relocation functionality. Ultimately, those cloud service instances remain locked under the control of the service provider, unless significant manual and/or ad-hoc efforts are spent by the service instance owners.

The proposed solution is the FluidCloud framework which aims to make relocating services instances easier with more automation. From this work a number of research and engineering challenges arise including data optimisation, runtime architecture adaptation, goal-oriented service instance relocation.

2 A Problem in the Cloud?

Cloud service instances remain locked under the control of the service provider. FluidCloud will liberate these instances. Having the ability for a cloud service instance to easily and seamlessly move from one provider to another will bring advantages to any cloud service owner. It would bring more freedom to cloud service owners, instances, including their application and data. Essentially, it will bring service instance "movement rights" to the cloud. However there is no encompassing means to accomplish this.

FluidCloud is something that fits within the soon future cloud. A reasonable view of this future cloud is the InterCloud. The InterCloud is described in [2], [3], [4] as well in a notable online conversation[1] where James Urquhart, Cisco's 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.

[1] I'm a little footnote short and stout!

2.1 The FluidCloud Concept

Within the highly populated ecosystem of cloud service providers combined with the InterCloud concept, it becomes crucial, both technically and financially, that service instances can be relocated and consequentially adapted to their new service provider. In the vision of the FluidCloud framework, the hosting cloud provider of the service instance is not a concern anymore as the service can be fluidly (easily, on-demand and dynamically) relocated between providers. The FluidCloud concept addresses multiple benefits – including:

- **Enhancing 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 service providers and based on economical differentiators and should the service owner want it; relocate their service instance to the suggested target provider.
- **Regulatory**: If a running cloud service instance 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 instance 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 instances and the efficient means to enable them.
- **Positive Market Disruption**. By having the ability to relocate a service instance from one provider to another, the market place is opened further, enabling greater competition based on service provider differentiation and not on technical lock-in or limitation.

2.2 FluidCloud Scenarios

To understand further how FluidCloud can be applied in a more practical sense, consider some examples which demonstrate the need for the FluidCloud framework:

- **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 [5]. 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). A cloud service provider would like to relocate (or "on board") new customers from other cloud service providers. What is needed is to relocate new end-user service instances 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.

The cloud service provider operates FluidCloud framework and offers it as a service.

- **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 a cloud service instance on the behalf of the service owner. Once that target service instance is provisioned the end-user interacts and uses the target service instance, either through interfaces provided by the cloud broker or directly using the interfaces of the provisioned target service instance. The cloud broker is aware of the end-users service instances and can continually watch for compatible service types that can be offered to the end-user as a replacement, based on economic/geo-location/cost bases.

If the service owner was interested the cloud broker can, using FluidCloud, relocate the service on the owner's behalf.

2.3 FluidCloud Contributions

For such scenarios, as described above, to be technically realised there is a set of missing technologies. FluidCloud fills these gaps by providing the following key contributions:

- **Service Instance 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 Instance 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 may need to be converted for running on a different hypervisor and the resumed VM may need to be re-contextualized as its environment changes [6]. 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[2] provide a service for easy transferring data between Grid sites using the proven GridFTP protocol [7], which allows for fast and reliable data transfers. Other solutions like the Zeta File System (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 to providers.

3 Architecture

Core to realising FluidCloud are the following components: The CloudConduit handles and coordinates the overall relocation (possibly partial) of the service instance. 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:

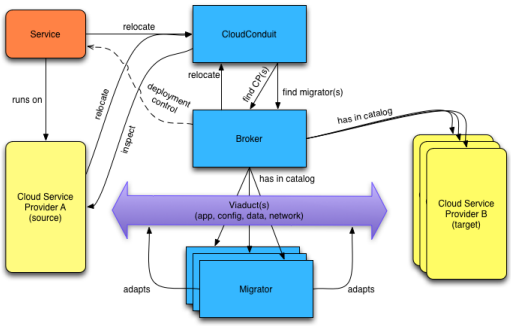

Figure 1: Conceptual Overview

The key components are the following:

- **Service Instance** - a logical container that comprises the application and the data. Executing the applications will form the workload. Can contain sub-service instances.

- **CloudConduit** - orchestrates the process, introspects the service instances (incl. 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 service instances. 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 service instances. 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

[2] I'm a little footnote short and stout!

accomplish the task of relocating a service instance 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 service types.

With these concepts, FluidCloud will provide the architecture and tooling to enable the InterCloud by enabling the three key contributions introduced earlier: Service Relocation, Service Adaptation and Data Relocation.

Service instances 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 Instances. FluidCloud will show the relocation of a service instance (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 instance will automatically be adapted to the new environment.

2. Relocation of PaaS-based Service Instances. The relocation of a PaaS based service instance and its data between providers. Key to this demonstration is the ability to adapt the service instance (on source code level) and convert it to the new environment. The motivation should be to bring PaaS services from closed environments (e.g. Google App Engine) into more open ones (e.g. CloudFoundry).

3. Service Instance Adaptation: IaaS to PaaS. A software developer has developer an 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.

4 Implementation

The first proof of concept of the conceptual architecture for IaaS based relocation 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.

The CloudConduit has capabilities to processes requests for service instances to be migrated. When such an relocation is triggered it inspects the service instances

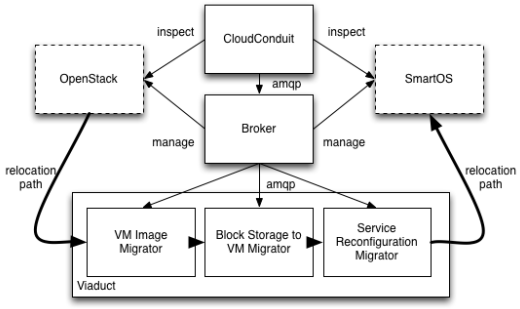

Figure 2: Architectural Overview

for sub-components (sub-services) 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 does now have the information to instantiate the appropriate Migrators that make up the Viaduct. The Migrators now take care of the actual relocation and topology change within the service instance.

For this PoC the setup has been done within in one Data center but with two different platforms. The source service provider is realized using OpenStack while the target side is SmartOS. The trigger for the relocation is done based on an evaluation of both platforms. It shows that an virtual machine on the SmartOS platform has a significantly higher IO throughput:

TODO: insert dd command output here. - @Andy do we want it here?

Based on this evaluation, a simple node.js application is deployed with an virtual machine within OpenStack is to be relocated to SmartOS. 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.

After relocation the virtual machine 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 (could be the use case of using an external provider such as Amazon's S3). This will demonstrate that the service topology of the service instance 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:

The decision for this service topology after relocation is made by the CloudConduit and should be guided by

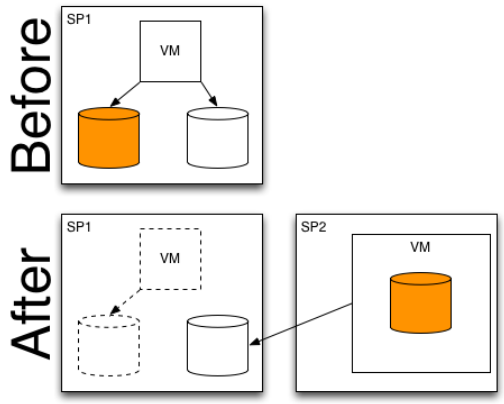

Figure 3: Service before and after relocation

service owner policies. Overall, to relocate this simple node.js application the following Migrators need to be placed on the Viaduct:

- The virtual machine image Migrator will pause the virtual machine on the OpenStack side and create a new virtual machine on the SmartOS side. Both machines are than booted using an prepared ubuntu iso image[3]. After booting the data is copied over using *netcat* and the *dd* command. After copying the virtual machine on the SmartOS platform can be booted.

- The Relocator for the OpenStack block storage copies the data from the block storage device in OpenStack cinder onto the Filesystem of the virtual machine running on KVM in a SmartOS zone. This is done with the help of the *ssh* protocol.

- Reconfiguration of the node.js application's configuration file is based on regular expressions and a simple Python script. It changes the paths for node.js interpreter and the path to the data.

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 instance

3. each of those VMs may have associated resources - CloudConduit must determine these

[3]In this case the virtual machine within OpenStack was initially booted from a volume instead of an image.

5 Evaluation

TODOs: * add in TCP/IP stack overhead and processing time of image, volume and service reconfig * 2GB volume, 512 VM size, over 1G ethernet

dd takes about ∼5mins for 5.4Gb disk size for reloc

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 technically feasible. More over the important thing is that the concepts noted in this paper describe a way of enable fluidity of services between clouds.

6 Related Work

Standards organisation defined interfaces such as OCCI[4], CIMI[5] or CDMI[6] might realise interoperability from a user perspective but they do not solve the issue of relocation. The paper [8] 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 discussed in [9]. Here it is noted that cloud systems utilising different hypervisors will not interoperate, in part because they do not use the same data formats.

Adapter libraries enabled the means to manage multiple cloud offerings, The most prominent of these are libcloud[7], fog.io[8] and jClouds[9]. There are commercial products available currently that aim to easily use multiple clouds at runtime. Such examples include RightScale[10] and Enstratus[11]. 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 service instances 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 CloudFoundry and Google App Engine. These platforms can be found to support Java, Python, Erlang, Node.js and

[4]I'm a little footnote short and stout!
[5]I'm a little footnote short and stout!
[6]I'm a little footnote short and stout!
[7]I'm a little footnote short and stout!
[8]I'm a little footnote short and stout!
[9]I'm a little footnote short and stout!
[10]I'm a little footnote short and stout!
[11]I'm a little footnote short and stout!

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 released a report [10] on long distance service instance 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 [11] 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[12], Xeround[13], MongoLab[14] or Amazon S3[15]. But currently they lack the ability to convert data between the service instances, 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 [12] 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 [13] offering aggregation, integration and customisation, the three primary roles expected of such a cloud service broker. This is further refined in the NIST Definition of Cloud Computing [1].

7 Conclusions and Further Work

The need for service relocation will become ever needed the more cloud services are used and the more that service owners move their service to the public cloud. FluidCloud will present a means for this to be supported and there is a prototype framework available. This framework will be released and supported under an Open Source license. There are further research and engineering areas to be investigated including live service instance relocation between data centres, data payload minimisation, service decomposition over multiple target service providers and the leverage of software-defined networking technologies.

[12] I'm a little footnote short and stout!
[13] I'm a little footnote short and stout!
[14] I'm a little footnote short and stout!
[15] I'm a little footnote short and stout!

References

[1] Grance, P. M. (2011). The NIST Definition of Cloud Computing. NIST special publication.

[2] Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., & Morrow, M. (2009). Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability. Internet and Web Applications and Services, 2009.

[3] D. Bernstein, D. V. (2010). Intercloud Exchanges and Roots Topology and Trust Blueprint.

[4] Y. Demchenko, M. X. (2012). Intercloud Architecture for Interoperability and Integration.

[5] Elson, J. (2008). Handling flash crowds from your garage. USENIX 2008 Annual Technical Conference.

[6] D. Armstrong, D. Espling, J. Tordsson, K. Djemame, and E. Elmroth. Runtime Virtual Machine Recontextualization for Clouds. Euro-Par 2012 Workshops, Lecture Notes of Compouting Science, Vol. 7640, Springer-Verlag, pp. 567 - 576, 2012.

[7] John Bresnahan, Michael Link, Gaurav Khanna, Zulfikar Imani, Rajkumar Kettimuthu and Ian Foster. Globus GridFTP. Proceedings of the First International Conference on Networks for Grid Applications (GridNets 2007), Oct, 2007

[8] Petcu, D. (2011). Portability and interoperability between clouds: challenges and case study. Towards a Service-Based Internet , 62–74.

[9] Jr, S. O. (2011). The problem with cloud-computing standardization. Computer magazine , 13–16.

[10] Alliance, O. D. (2012). Long Distance Workload Migration. ODCA.

[11] Mizgier, H. K. (2010). System to system migration for improving interoperability. Information Technology .

[12] C. Ward, N. A. (2010). Workload Migration into Clouds Challenges, Experiences, Opportunities. 2010 IEEE 3rd International Conference on Cloud Computing. IEEE.

[13] Daryl C. Plummer, B. J. (2011). Cloud Services Brokerage Is Dominated by Three Primary Roles. Gartner.