Compute and Services (CAS) optimized strategy for multi hybrid cloud deployment

Pijush Biswas[#1], Kailash Verma[*2]

Cloud Architect, CAMS, IBM India Pvt Limited, DLF Silokhera, Gurgaon, HR, INDIA

Abstract: *While doing application migration from a source environment, there are infrastructure & application services parts that should move to a destination environment. Based on the affinity, each part of infrastructure and application services need to go either in target environment, or they remain in source environment. The target and source mapping are not always 1-1 mapping in real use cases. The source component may be an infra component and, but the target component may be PaaS component in a cloud environment. So, there is a need to put a system and methods to map components and services to build the ecosystem and develop a machine learning model from experience.*

.

Keywords — *Cloud Migration, Multi-Cloud Deployment, Self-Contained model, Public Cloud*

I. INTRODUCTION

Model of multi-dimension decision tracing with tree maps are based on portioning the problem into multiple dimension and represent them in a tree view so that each micro component is assured to be taken care of by migration engineer when they make a move plan. When one application is chosen to migrate, the application in split into infra services and application services. Each service is either mapped to target services or they are left at source with a valid decision framework. The infra component, application component, location component, vendor component becomes part of the model.

II. THE PROBLEM DEFINITION

While executing a large migrations project, the validation of target models is unclear unless there is a signed off roadmap which is validated against each micro component in the source environment against target destination platform. It is difficult to foresee complete picture whether the model would work. If a governing body wants to validate the roadmap, affinity or wave planning provide very little or no options. So, sponsor wants to ensure that target platform would really fit to the requirement.

To do an efficient cloud migration here are some challenges for today's world. To list few of them:

1. Heterogenous data sources (TADDM, ADDI, ADDM, BMC etc) for Enterprise discovery
2. Multiple proprietary toolsets(SCOPE, Transformation Manager, Cloud Decision Matrix etc.) to consume discovery data and generate insights for target cloud roadmap design
3. Datasets become non compatible DB schema and cannot be ported across applications &DevOps tools
4. Cloud decision matrix are complex, and they are heterogeneous services
5. No alternatives disposition are recommended for a typical deployment model 6. No way to redesign the whole deployment system again

III. THE APPROACH

When each of the micro element of source application are scoped to a disposition and signed by migration engineer, we would get a complete picture that every element would be migrated or omitted with a given valid reason. If one element is not migrated what is the alternative and corresponding services in multi-cloud target. If we want to sign off a target design for a cloud move, we need a snapshot of self-contained problem definition to ensure the migration happens with sure success. So far, we have been doing affinity modelling or have been creating move group, the fact is that they are one dimensional model in nature. With tree view depiction, we can drill down the problems into multiple dimension and each micro component is supported by an architectural decision. MOGAS framework is built to provide full life cycle of the given problem.

Step 1: MAP: Depict source environment in a tree view with hierarchical breakups. The component are hardware and software entities.

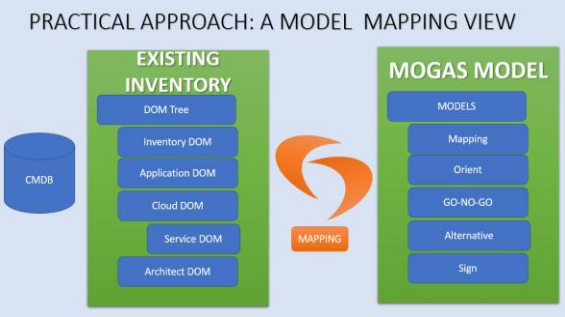
Step 2: ORIENT: Take every element of hierarchy and provide disposition of service equivalent in multi-cloud for target vendor. The vendors are azure, AWS, GCP, RedHat etc. If one component is not required for migration, we can tag it "NO-GO" against that component. If the source component is infra services and the mapping component is PaaS, the tree view will be updated with architectural decision to make cloud native services. Target cloud dimensions are added at this point to enable multi-cloud deployment model.

Step 3: GO-NO-GO: If the source component is mapped correctly either with infrastructure or platform service, they would be migrated to target cloud. Else they are tagged as "NO-GO" by design architect with an alternative solution.

Step 4: ALTERNATIVE: Alternative disposition are provided by architect by appropriate architectural decision and they are decommissioned or retired based on written recommendation by architects.

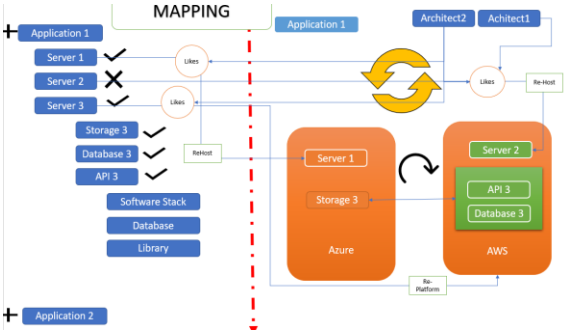
Step 5: SIGN: Each action is created, validated, reviewed by architects. So, the action items are signed off by an architect. There after the MOGAS model goes for DevOps consumption.

MAPPING VIEW: The source is broken down in multiple services and component to form a DOM (Document Object Model). Here is the view of the model:

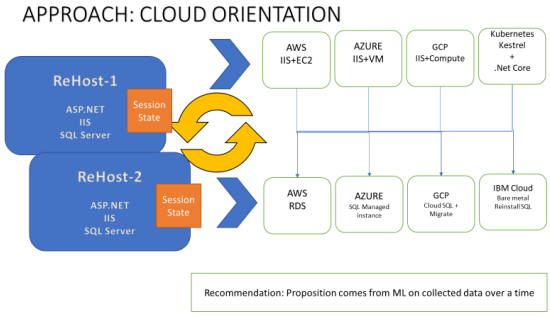


IV. THE MAPPING COMPONENT

Once the maps are established, the dispositions are made to each of the mapping by multiple cloud architects from multiple vendors including the customer. Then the scoring happens based on their ratings for each disposition on the service component and cloud services.



And orientation happens on following architects and multiple cloud vendors. Then top scoring mapping dispositions are signed off by the application owners and MOGAS model is generated for further consumption by toolchain. The model still contains the inventory information, architectural decision and scoring information which can be used for machine learning algorithm. Here is provided a view of each human and machine component of MOGAS model and how they are associated with multiple public cloud of current times.



Orientation is followed by GO or NOGO voting, where NOGO is loop breaker. Here is the rule that applies on this process before signing off happens.

1. GO OR NOGO are driven by Likes Count and Like Rates of design architects
2. GO is the happy use case and signed off by technical design authority
3. NOGO is conflicting use case, will go for further decision framework
4. NOGO is loop breaker and followed by disposition like RETIRE or OUT OF SCOPE

V. WHERE ARE THE BENEFITS

The system and methods applied for rapidly scaling a migration process needs to understand tool assisted delivery and adapt to a rapidly growing multi-vendor tool set. Every tool thus becomes a player of the MOGAS model, and they can play it on their canvas. The scripted DevOps become a feasible for Terraform, Ansible or other toolset and they all can run side by side for an engagement. Here are some listed benefits available from this model.

A. *The models are used in heterogenous systems and still a self-contained model what help in designing the target architecture and stockholder associated with multi-cloud deployment process.*

B. *Having standardization of inventory DOM models chosen by vendors, clients and partners*

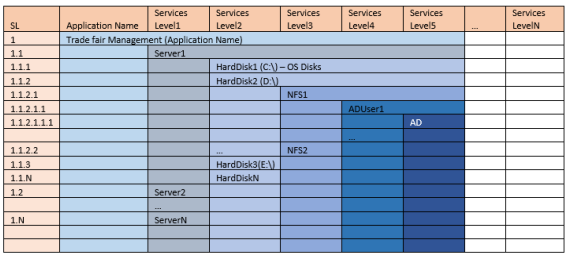
C. *Iterative framework to build data model for mapping source inventories into a finite set of target services*

D. *Data Model independent of proprietary toolset and contains every information to transform inventory to actional scripts for deployment*

E. *If correct units are allocated to cloud services, Total cost can also be obtained from this model. Deviation of disposition of service can trigger the chain to find total cost.*

VI. USE CASEI

The use cases are taken form service delivery where infrastructure data is taken from a CMDB. They are mapped into a target service. So, they are presented in a tree view and each of the services are depicted as tree node. At any point of time, it is very much clear if the migration roadmap has a gap or they need a further work. The DOM (Document Object Model) ensures the completeness of the whole program. This document is then converted into a MOGAS model for onward travel from one toolset to another toolset which acts of the MOGAS model.



The use case 1: Application is hosted and distributed over in 2 servers. Application shares the information with NFS which is configured with HardDisk2 on both the servers. When NFS is configured with accounts who has access to these file systems. Accounts are coming from Active Directory.

The problem is depicted in tree. If we do image cloning for lift and shift purposes, the MAP is just one to one. The orientation becomes null and void with rehost program. In case, we need to put the application into a cloud, the NFS would not migrate as AS-IS and at the same time AD is a PaaS (Platform as A Service). So, here MAP is done with heterogeneous entities. The Disk become storage account and NFS becomes Blob shares. So, next dimension "cloud vendor" comes into play. With various options with private cloud and public cloud, the MAP becomes complex decision metrics and should then go with ORIENTATION with target cloud.

With so many disks, few of them are redundant when we go with storage account to store objects. Disk1 is required as this is mapped one to one. But Disk2 is required to orient to new server with an approval from migration architect. When Disk3 does not contains any valid data, they would not make to target platform, so here this resource is NO-GO and other valid configuration are candidate to GO to target cloud. With these arrangements, the design architect must sign the plan to complete the picture and build a generic MOGAS model.

MOGAS model is a generic json model which can be consumed by any DevOps tool chain to create services in target platform. There are many DevOps tools like Terraforms, Ansible, Azure Templates, AWS CloudFormation which can consume these MOGAS model as Infrastructure as A Code. The deployment time reduced to 60%-70% with high accuracy.

F. References

The model is drawn from experience and knowledge from multiple service delivery in order to overcome the complexity of heterogeneous systems and multiple player like man & machines. The references are taken from public cloud vendors who continuously adding to inventory with newer services. Azure, GCP, OpenShift, AWS clouds are most talked about platform and they continuously hosting their manuals and product pages. These are all public website; the services are collected from them.

VII. CONCLUSIONS

The migrations of application happen various stages and various teams are involved into effective migration of the same applications. In a multi-cloud deployment, the towers are placed across platform, middles-wares and infrastructure. Then there are other segments like discovery, insight, design and build. Each of these towers are cross cutting and they need mode of communication which are descriptive and self-contained. With MOGAS model ensure that each model is complete and assured to execute complete migrations, the communication mode remain case for innovation and all totally complete by itself.

REFERENCES

[1] Raju Chiluvuri "*System and method of application development using replaceable self-contained components (RSCCS)*",US, US Patent 2010

[2] Koren Lev Mehernosh Naval Vadiwala Catherine Ruth Gulsvig Wood Derek Evan "*Multi-tiered cloud application topology modeling tool*", Cisco Technology Inc, US Patent 2014

[3] Luc BOUTIER Gauvin GIRAULT, "*Methods for managing the life cycle of a cloud application using a plurality of cloud infrastructures*," US Patent, 2014 FR 2015

[4] William L. Scheidel Robert M. FriesSrivatsan Parthasarathy Alan C. ShiJames P. Finnigan "*Automated deployment and servicing of distributed applications*," in US Patent, 2010

[5] Yan OrJohan Casier Krishna Garimella Umesh Bellur John Koper Shashank JoshiVinu Sundaresan, "*Deployment of applications in a multitier compute infrastructure* International Business Machines Corp, US Patent 2003.

[6] Caballer MBlanquer IMoltó Gde Alfonso C "Dynamic management of virtual infrastructures" Caballer et al., 2015

[7] Konstantinou AEilam TKalantar MTotok AArnold WSnible E "*An architecture for virtual solution composition and deployment in infrastructure clouds*" Konstantinou et al., 2009

[8] Azure: https://azure.microsoft.com/en-in/

[9] AWS: https://aws.amazon.com/

[10] GCP: https://cloud.google.com/

[11] OpenShift: https://www.openshift.com/
IBM Cloud: https://www.ibm.com/in-en/cloud