ISSN (Online): 2349-7084
GLOBAL IMPACT FACTOR 0.238
ISRA JIF 0.351

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 1, ISSUE 6, DECEMBER 2014, PP 391-396

Revolution of Storms from Vendor lock-in to the Data storage

[1] **Y.ANITHA,** [2] **D.RAVIKIRAN**

[1] M.Tech Research Scholar, Priyadarshini Institute of Technology and Science for Women
[2] Professor, Priyadarshini Institute of Technology and Science for Women

Abstract: The open stack-based private cloud can help mitigate vendor lock-in and promises transparent use of cloud computing services. Most of the basic technologies necessary to realize the open stack-based private cloud already exist, yet lack integration. Thus, integrating these state-of-the-art tools promises a huge leap toward the open stack-based private cloud. To avoid vendor lock-in, the community must drive the ideas and create a truly open stack-based private cloud with added value for all customers and broad support for different providers and implementation technologies.

Keywords- Open stack-based private cloud, privacy preserving, Integrity, Cloud Storage

1. INTRODUCTION

To some point, we can understand the Meta cloud based on a grouping of existing tools and concept, part of which we just examine. Figure 1 depicts the Meta cloud's main components. We can arrange these components based on whether they're important generally for cloud software developers throughout expansion time or whether they execute tasks throughout runtime. We explain their interaction utilizing the games gambling portal for a simple example. The Meta cloud API gives a combined programming interface to summary from the difference among source implementations of API. For users, utilizing this Application Program Interface prevent their request from being typically-wired to a particular cloud service submission. The API of Meta cloud can develop on available source cloud provider abstraction APIs, as previously mentioned. Even these deals mostly with the key value stores and computer services, in standard, all services can be covered that are theoretical more than one service to offer and whose specific APIs don't differ too much, Theoretically. Resource template engineers explain the cloud services required to process an application utilizing resource templates. They can identify service categories with extra proper ties, and a model of graph explores the functional and interrelation dependency between services. Developers create the Meta cloud reserve templates utilizing a plain DSL (domain-specific language), hire them in a few words specify necessary resources. Store definitions are based on a kind of masterpiece model; thus engineers can develop reusable and configurable emplate components, which use them and their groups to reuse and share general resource templates in various projects. Utilizing the domain-specific language, engineers model components of their application and their necessary runtime needs, like as memory, I/O capacities, and CPU, as well as weighted and dependency communication between these components. The provision strategy uses the relations of subjective component to conclude the application's optimal deployment configuration.

Moreover, resource template allows engineers to describe restrictions depending upon expenditures, geographical distribution and component proximity.

2. CLOUD COMPUTING USE CASE

The Cloud Computing Use Case group took together cloud customers and cloud vendors to describe general use case scenarios for cloud computing. The use case scenarios demonstrate the performance and economic benefits of cloud computing and are based on the needs of the widest possible range of consumers.

The goal of this white paper is to highlight the potentiality and requirements that require to be standardized in a cloud environment to ensure interoperability, ease of integration and portability. It must be possible to apply all of the use cases described in this paper without using closed, proprietary technologies. Cloud computing must evolve as an open environment, minimizing vendor lock-in and increasing customer choice.

2.1 The use cases:

Provide a practical, customer-experience-based context for discussions on interoperability and standards. Make it clear where existing standards should be used. Focus the industry's attention on the importance of Open Cloud Computing. Make it clear where there is standards work to be done. If a particular use case can't be built today, or if it can only be built with proprietary APIs and products, the industry needs to define standards to make that use case possible.

A use case that clearly describes a common task and outlines the difficulties in accomplishing it is the best possible justification for any standards effort.Lately, the Meta cloud idea has received some attention, and several techniques try to tackle at least parts of the problem.

3. CURRENT WEATHER IN THE (META) CLOUD

First, standardized programming APIs must enable developers to make cloud-neutral requests that aren't hardwired to any single supplier or cloud service. Cloud supplier abstraction libraries such as libcloud (http:// libcloud.apache.org), fog (http://fog. io), and jclouds (www.jclouds.org) provide unified APIs for getting into different vendors' cloud products. Using these libraries, developers are pleased of technological vendor lock-in because they can switch cloud providers for their applications with relatively low overhead.

As a second ingredient, the meta cloud uses resource patterns to describe existing features that the application needs from the cloud. Some current tools and schemes — for example, Amazon's- Cloud Formation(http://aws.amazon.com/cloudformation/) or the upcoming TOSCA specification (www.oasis-open.org/committees/tosca) — are working near similar objectives and can be adapted to provide these required features for the meta cloud.

Most implicit metadata is practical in nature and is normally although not always of more use to those administering the collection rather than those using it. While most implicit metadata is derived from the file itself, a certain amount could also be derived from its context (e.g. its location within directories/folders or on servers). In developing a digital collection it may be useful to extract some implicit metadata and hold it separately within a database for the purposes of retrieval, quality control, or digital preservation. Typically though, much implicit metadata is left untouched within the file.

Although explicit metadata should be created by humans, it require not all be created by those building and classification a digital collection. It is very likely that there is some pre-existing *legacy* metadata that can be exploited (even if it is just a scrawled inscription on the back of a photograph, film can or audio cassette). Or it might be possible to get your collection users to add to the metadata in a semi-controlled way (via tags or annotations).

3.1 Inside the Meta Cloud

Metadata might take the form of controlled terminology, carefully constructed or chosen from formal lists and entered into pre-established categories. Or it might be simply a free text description or set of keywords used to explain or 'tag' an image. It might describe something objective and straightforward, such as the file size of the digital file; or something much more complex, such as the subject matter of the resource or legal rights associated with its use. Metadata is often held within databases, but it can take other forms - it can just as easily be found embedded within the digital file itself. In short, metadata provides the means for us to describe our digital resources in a structured way that enables us to share those resources with other people and machines.

3.2 Meta Cloud API

In providing an API that is responsive, fast, clean, and performs brilliantly for a core set of tasks that a developer couldn't live without. And that's just what we've set out to build.

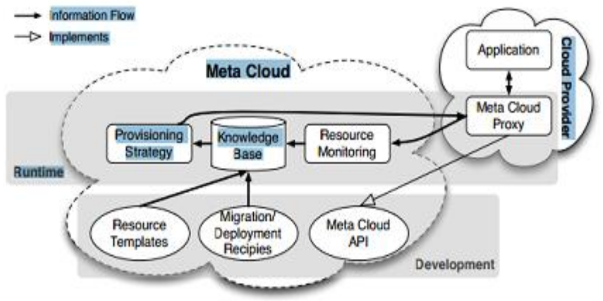

Figure 1. Conceptual meta cloud overview. Developers create cloud applications using meta cloud development components. The meta cloud runtime abstracts from provider specifics using proxy objects, and automates application life-cycle management.

3.3 Resource Templates

Developers explain the cloud ser-vices need to run an application using resource templates. They can specify service types with extra properties, and a graph model says the interrelation and functional dependencies between services. Developers create the meta cloud resource templates utilizing a simple domain-specific language (DSL), letting them in brief specify required resources. The provision-ing strategy uses the weighted com-ponent relations to determine the application's optimal deployment configuration. Moreover, resource templates allow developers to define constraints based on costs, component proximity, and geographical distribution.

3.4 Meta Cloud Proxy

The Meta cloud gives proxy objects, which are arranged with the request and run on the provisioned cloud resources. Proxies give a way to perform deployment and migration recipes triggered by the Meta cloud's provisioning strategy. Moreover, proxy objects send QoS statistics to the resource monitoring component running within the Meta cloud. The Meta cloud contains the data by capturing the application's calls to the underlying cloud services and measuring their processing time, or by executing short benchmark programs. Applications can also describe and monitor custom QoS metrics that the proxy objects send to the resource monitoring component to enable advanced, application-specific management strategies. Proxies don't run inside the Meta cloud, and normal service calls from the request to the proxy aren't routed through the Meta cloud, either.

3.5 Resource Monitoring

Meta cloud proxies are the resource monitoring component receives data collected on an application's request, about the resources they're using. The component sifts and processes these data and then stores them on the knowledge base for further processing.

3.6 Provisioning Strategy

The use of a company's cloud computing strategy, which typically first engages selecting which application and services will reside in the public cloud and which will remain on site behind the firewall or in the private cloud. Cloud provisioning also entails developing the processes for interacting with the cloud's applications and services as well as auditing and monitoring who accesses and utilizes the resources.

The most common reference to cloud provisioning is when a company seeks to transition some or all of its existing applications to the cloud without having to significantly re-architect or re-engineer the applications.

4. KNOWLEDGE BASE

To estimate migration costs based on their pricing and QoS, and information necessary to the knowledge base stores data about cloud provider services. Which cloud providers are eligible for a certain customer to the knowledge base indicates. Several information sources contribute to the knowledge base: Meta cloud proxies repeatedly send data about request behavior and cloud service QoS. Clients can add cloud service providers' pricing and capabilities manually or use swarming methods that can get this information automatically.

A Meta Cloud Use Case

Let's come back to the sports request use case. The Meta cloud API and doesn't directly talk to the cloud-provider-specific service APIs to the meta-cloud-compliant variant of this application accesses cloud services. For our particular case, this means the request doesn't depend on Amazon EC2, SQS, or RDS service APIs, but rather on the meta cloud's compute, message queue, and relational database service APIs.

Metadata relating to a digital resource can come from one of two sources: (a) it can be automatically derived from the digital resource itself, or (b) can be created and associated with a resource by human beings.

The first kind of metadata might be called *intrinsic* or *implicit metadata*. Examples of this include file formats, resolution, bit-depth, or frame-rate. File formats typically encode this sort of information within the header (the first section) of the digital file. If an image has been created by a digital camera, it is likely that the camera has also written a certain amount of information about the digital capture into the file header, such as the camera make and model, its settings, and the date the photograph was taken (this makes use of the EXIFstandard).Most implicit metadata is technical in nature and is generally - although not always - of more use to those administering the collection rather than those using it. While most implicit metadata is derived from the file itself, a certain amount could also be derived from its context (e.g. its location within directories/folders or on servers). In developing a digital collection it may be useful to extract some implicit metadata and hold it separately within a database for the purposes of retrieval, quality control, or digital preservation. Typically though, much implicit metadata is left untouched within the file.

The second kind of metadata might be called *extrinsic* or *explicit metadata*. Because this is created by humans, it is the most difficult and expensive metadata to create. But it is also usually the most important - especially to the end user. The advice documents in this series are mostly concerned with creating and managing explicit metadata.

Although explicit metadata must be created by humans, it need not all be created by those building and cataloguing a digital collection. It is very likely that there is some pre-existing *legacy* metadata that can be exploited (even if it is just a scrawled inscription on the back of a photograph, film can or audio cassette). Or it might be possible to get your collection users to add to the metadata in a semi-controlled way (via tags or annotations).

Those developing digital collections will need to make decisions about what implicit metadata should be extracted and what explicit metadata needs to be gathered or created to support the collection and its users. Usually resource limitations will play a role in these decisions. Some collections can afford to spend hours creating metadata for each resource; others less so. Some digital asset management systems are able to automatically extract implicit metadata from a digital file; many cannot and will rely on the cataloguer using other tools to discover this information manually.

5. CHALLENGES ON META CLOUD

Working on the Meta cloud, we face the following technical challenges. Resource monitoring must collect and process data describing different cloud providers' services such that the provisioning strategy can compare and rank their QoS properties in a normalized, provider-independent fashion. Although solutions for deployment in the cloud are relatively mature, application migration isn't as well supported. Finding the balance between migration facilities provided by the Meta cloud and the application is particularly important. Cloud-centric migration makes the Meta cloud infrastructure responsible for most migration aspects, leading to issues with application-specific intricacies, whereas in application-centric migration, the Meta cloud only triggers the migration process, leaving its execution mostly to the application. We argue that the Meta cloud should control the migration process but offer many interception points for applications to influence the process at all stages. The provisioning strategy, the most integrative component, which derives strategies mainly based on input from runtime monitoring and resource templates and effects them by executing migration and deployment recipe, requires further research into combining approaches from the information retrieval and autonomic computing fields.

6. CONCLUSSION

In this paper we addresses the open stack-based private cloud can help mitigate vendor lock-in and promises transparent use of cloud computing services. The majority of the basic technologies necessary to realize the open stack-based private cloud already exists yet lack integration. Thus, integrating these state-of-the-art tools promises a huge leap toward the open stack-based private cloud. To avoid vendor lock-in, the community must drive the ideas and create a truly open stack-based private cloud with added value for all customers and broad support for different providers and implementation technologies.

REFERENCES

[1] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm, "What's inside the Cloud? An

architectural map of the Cloud landscape," in Software Engineering Challenges of Cloud Computing, 2009. CLOUD'09. ICSE Workshop on. IEEE, 2009, pp. 23–31.

[2] C. Baun, M. Kunze, J. Nimis, and S. Tai,Cloud Computing: Web-basierte dynamische IT-Services, ser. Informatik im Fokus. Berlin:Springer, 2010.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman,A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo:amazon's highly available key value store," inProc. SOSP, 2007

[4] M. Welsh, D. Culler, and E. Brewer, "SEDA: architecture forwell-conditioned, scalable Internet services,"ACM SIGOPS OperatingSystems Review, vol. 35, no. 5, pp. 230–243, 2001.

[5] S. Garfinkel, "An Evaluation of Amazon's Grid Computing Services:EC2, S3, and SQS," inCenter for. Citeseer, 2007

[6] A. T. Velte, T. J. Velte, and R. Elsenpeter,Cloud Computing: A Practical Approach. Upper Saddle River, NJ: McGraw-Hill, 2010.

[7] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Yous-eff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE, 2009, pp.124–13

[8] N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman,and R. Wolski, "Appscale: Scalable and open appengine applicationdevelopment and deployment,"First International Conference on CloudComputing, 2009.

[9] A. Lakshman and P. Malik, "Cassandra: a decentralized structuredstorage system,"ACM SIGOPS Operating Systems Review, vol. 44, no. 2,pp. 35–40, 2010.

[10] R. Thomas, "A majority consensus approach to concurrency controlfor multiple copy databases," ACM Transactions on Database Systems(TODS), vol. 4, no. 2, pp. 180–209, 1979

[11] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, andD. Lewin, "Consistent hashing and random trees: Cloud cachingprotocols for relieving hot spots on the World Wide Web," in Proceedings of the twenty-ninth annual ACM symposium on Theory ofcomputing. ACM, 1997, pp. 654–663.

[12] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan,"Chord: A scalable peer-to-peer lookup service for internet applications, "in Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. ACM, 2001,pp. 149–160.

[13] J. Elson and J. Howell, "Handling flash crowds from your garage,"in USENIX 2008 Annual Technical Conference on Annual TechnicalConference. USENIX Association, 2008, pp. 171–184.

[14] S. Bourne, "A conversation with Bruce Lindsay,"Queue, vol. 2, no. 8,pp. 22–33, 2004.

[15] T. Chandra, R. Griesemer, and J. Redstone, "Paxos made live: anengineering perspective," inProceedings of the twenty-sixth annual ACMsymposium on Principles of Cloud computing. ACM, 2007, pp.398–407

[16] L. Lamport, "The part-time parliament,"ACM Transactions on ComputerSystems (TOCS), vol. 16, no. 2, pp. 133–169, 1998

[17] H. Weatherspoon, P. Eaton, B. Chun, and J. Kubiatowicz, "Antiquity:exploiting a secure log for wide-area Cloud storage,"ACM SIGOPSOperating Systems Review, vol. 41, no. 3, pp. 371–384, 2007.

[18] M. Burrows, "The Chubby lock service for loosely-coupled Cloudsystems," in Proceedings of the 7th symposium on Operating systemsdesign and implementation. USENIX Association, 2006, pp. 335–350.

[19] A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur,J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer, "FARSITE:Federated, available, and reliable storage for an incompletely trustedenvironment,"ACM SIGOPS Operating Systems Review, vol. 36, no. SI, pp. 1–14, 2002

[20] K. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability andintegrity layer for cloud storage," in Proceedings of the 16th ACMconference on Computer and communications security. ACM, 2009,pp. 187–198.

[21] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels,R. Gummadi, S. Rhea, H. Weatherspoon, C. Wellset al., "Oceanstore: An architecture for global-scale persistent storage,"ACM SIGARCH Computer Architecture News, vol. 28, no. 5, pp. 190–201, 2000

[22] Google, Datastore Performance Growing Pains. Google, Jun.2010, (accessed on December 4, 2010). [Online]. Available: http://googleappengine.blogspot.com/2010/06/datastore-perfor mance-growing-pains.html

[23] C. Bunch, N. Chohan, C. Krintz, J. Chohan, J. Kupferman, P. Lakhina,Y. Li, and Y. Nomura, "An evaluation of Cloud datastores using the appscale cloud platform," in Cloud Computing (CLOUD), 2010 IEEE3rd International Conference on, 2010, pp. 305 –312

[24] Broberg, Buyya, and Tari, "Creating a Cloud Storage Mashup forHigh Performance, Low Cost

Content Delivery," inService-OrientedComputing– ICSOC 2008 Workshops. Springer, 2009, pp. 178–183.
[25] J. Broberg, R. Buyya, and Z. Tari, "MetaCDN: Harnessing 'StorageClouds' for high performance content delivery,"Journal of Network andComputer Applications, vol. 32, no. 5, pp. 1012–1022, 2009
[26] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: a casefor cloud storage diversity," in Proceedings of the 1st ACM symposiumon Cloud computing. ACM, 2010, pp. 229–240.
[27] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication:A quantitative comparison,"Peer-to-Peer Systems, pp. 328–337, 2002.