Journal of Theoretical and Applied Information Technology
31st August 2016. Vol.90. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

METADATA AS A SERVICE (METAAS) MODEL FOR CLOUD COMPUTING

[1]AMIR MOHAMED TALIB, [2]FAHAD OMAR ALOMARY, [3]RUSLI ABDULLAH

[1, 2]Asstt Prof., Department of Information Technology, College of Computer and Information Sciences, Al-Imam Muhammad Ibn Saud Islamic University (IMSIU), SAUDI ARABIA

[3]Professor., Department of Information System, Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM), MALAYSIA

E-mail: [1]ganawa53@yahoo.com, [2]fahd.alomary@gmail.com, [3]rusli@fsktm.upm.edu.my

ABSTRACT

Cloud computing has become the most attractive field in industry and research. Metadata as a Service (MetaaS) is an emerging technique that could help the cloud users, and cloud service providers (CSPs) according their needs. The increasing of the speed of searching and acquiring against the number of the data services in cloud computing that has leads the researchers to think about implementing a new technique. MetaaS model uses to serve as a backbone for providing and searching for data storage in cloud computing. MetaaS model consists of three main layers as Metadata component, cloud users and CSPs. The Metadata components consists of six main components as Metadata Entity (ME), Metadata File Information (MFI), Metadata Catalog Service (MCS), Metadata Management Engine (MME), Metadata Capturing (MC) and Metadata Analysis (MA). In this paper, an approach for enabling searching, storing, accessing, retrieving, and capturing the data from Cloud Data Storage (CDS) based on MetaaS model is presented. Taking the production of CDS service as example, this paper gives formal analysis of system running and compares with other related work. The results show that the model presents good reference on the construction of cloud computing applications and services according to the cloud services functionalities and MetaaS components.

Keywords: *Cloud Computing, Cloud Data Storage, Cloud Service Provider, Metadata and Metadata as a Service*

1. INTRODUCTION

Cloud computing has become the most attractive field in industry and research. The concept of cloud computing includes the Web 2.0, web infrastructure, virtualization technologies, and other emerging technologies. With the cloud computing technology, users use a variety of devices with laptops, smart phones, PCs, PDAs to access storage, and application-services offered by cloud computing providers. Some advantages of the cloud computing technology include cost savings, high availability, and scalability [1].

The architecture of cloud computing are broadly divided into six categories, which are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), Communication-as-a-Service (CaaS), Data Storage-as-a-Service (DaaS), and Hardware-as-a-Service (HaaS). Currently there are five types of cloud computing namely: public cloud, private cloud, hybrid cloud, community cloud, and combination cloud. Public cloud describes cloud computing in the traditional main stream sense, where resources are based on self-service basis over the Internet, via web applications or web services. In public cloud, numerous cloud users can share the computing resources provided by a single service provider. Cloud users can quickly access these resources and only pay for the operating resources [2, 3]. Private cloud is a term to describe computing architecture that provides hosted services to a limited number of people behind a firewall. In the private cloud, computing resources are used and controlled by a private enterprise [4, 5]. A third type can be hybrid cloud that is typical combination of public and private cloud. It enables the enterprise to running state-steady workload in the private cloud, and asking the public cloud for intensive computing resources when peak workload occurs, then return if no longer needed. Two clouds that have been joined

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

together are more correctly called a "combined cloud". A combined cloud environment consisting of multiple internal and/or external providers "will be typical for most enterprises". By integrating multiple cloud services users may be able to ease the transition to public cloud services while avoiding issues.

The primary uses of cloud computing is for CDS. CDS is composed of thousands of storage devices clustered by network, distributed file systems and other storage middleware to provide cloud storage service for users [6, 7]. CDS is a model of networked computer data storage where data is stored on multiple virtual servers, generally hosted by third parties, rather than being hosted on dedicated servers. In CDS, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various block-level operations of searching, acquiring, storing, accessing, and capturing the data [3].

The main benefits of cloud computing are: (1) Cost of Hardware: In the world of technology everything costs money –servers, workstations, software, databases, etc so, cloud computing allows you to spend less money on these items, especially hardware. (2) Easy Expansion: Today when you add an employee, you have to purchase a new workstation for them, purchase additional software licenses, etc. (3) Fewer IT Staff: In the cloud, the hosting provider maintains the applications it is responsible for.

Metadata is descriptive information about the data. It is semantics on the basic concepts, basic relationships and basic constraints of data model. The metadata can solve problems that model layer cannot resolve, such as fuzzy semantic of data model, model integration, and sharing of information. By using metadata functional strong coupling relationships into data type weak coupling relationships is translated. Metadata research is widely used in data-driving system such as file system, information system, and so on [8].

There are various types of metadata that are specialized for particular types of metadata cataloguing and discovery. In this paper, design of a Metadata as a Service Model (MetaaS) that provides a mechanism for storing and accessing descriptive metadata and allows users to query for data items based on desired attributes is presented. CSPs have [the metadata], it's just a decision about whether or not they're going to invest in sharing it with their cloud users – and that's not a small investment," she says. "It's easy to share a CSP that's far too large and far too old for anybody to care about. For performance data to be useful, you need to get it in almost real time, and make accurate decisions based on it almost immediately.

There are two main examples of metadata: (1) File Information Metadata: Every file on your computer contains metadata that tracks things like where it was saved, when it was last accessed or modified, etc. Right click on any file in Windows then look under Properties to see this metadata. In most cases, this information is not dangerous. (2) Document Metadata: Metadata allows programs like Microsoft Word to provide collaboration and revision tools. When you use track changes, commenting or any review tool in Word, these features are stored in metadata. If you don't turn them off before you send the document out, someone else may be able to see this metadata.

The benefits of metadata are: (1) Metadata is not necessarily a bad thing. (2) The ability to perform revision and collaboration in Microsoft Word is a substantial benefit. (3) Microsoft did not develop these features to put their customers in danger, and (4) Without metadata, some of Word's most powerful features would not exist.

The reason of the danger of metadata because is that metadata often contains information you don't want others to see. For example, when you make a revision, do you really want someone to know what you changed if from? When viewed by the wrong person, metadata can have consequences ranging from embarrassing to career threatening.

2. LITERATURE REVIEW

Recently a large amount of work is being pursued in data analytics in cloud data storage (CDS) [9, 10]. Verma et al, [11] proposed metadata scheme using Ring File System (RFS). In this paper, the scheme metadata for a ring file is stored based on hashing in parent location. Replica is stored in its successor metadata server Baidu, etc. The GFS architecture comprises of a single GFS master server which stores the metadata of the file system and multiple slaves known as chunk servers which store the data. Files are divided into chunks (usually 64 MB in size) and the GFS master manages the placement and data-layout among the various chunk servers. The GFS master also stores the metadata like filenames, size, directory structure and information about the location and placement of data in memory. One of the direct implications

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

of this design is that the size of metadata is limited by the memory available at the GFS master. Hua et al, [12] proposed a scalable and adaptive metadata management in ultra large scale file systems. Cammert et al. [9] divided their scheme into two types: static and dynamic metadata. The author has suggested publish-subscribe architecture, enabled a SSPS to provide metadata on demand and handled metadata dependencies successfully.

Leung et.al. [13] describes the file metadata search system (Spyglass) that achieves scalability by exploiting storage system properties. An index that exploit storage properties is designed, Spyglass is able to achieve significantly better query performance than existing solutions, while using less disk space.

Gammert et.al. [9] produced and analyzed petabytes and terabytes of data intensive applications that are distributed in millions of files or objects. To manage the distribution of the large data sets, metadata needs to be managed.

Anitha et al. [14] has concluded that the data retrieval using metadata in cloud environment is less time consuming when compared to retrieving a data directly from the cloud data server. Ravi Kumar et al. [15] shows that third party auditor is used periodically to verify the data integrity stored at cloud service provider without retrieving original data. In this model, the user sends a request to the cloud service provider and receives the original data. If data is in encrypted form then it can be decrypted using his secret key. However, the data stored in cloud is vulnerable to malicious attacks and it would bring irretrievable losses to the users, since their data is stored at an untrusted storage servers.

Shizuka Kaneko et al. [16] has proposed a query based hiding schema Information using a Bloom filter. The query given is processed and the attributes of the query is used for key generation. The key generated is used to hide confidential information from the data administrator. As the query gets changes every time the key generation process becomes more complex. Marcos K. Aguilera et al. [17] has proposed a practical and efficient method for adding security to network-attached disks (NADs). The design specifies a protocol for providing access to the remote block-based devices. Chirag Modi et al. [18] discussed a survey paper where they discussed about the factors affecting cloud computing storage adoption, vulnerabilities and attacks, and identify relevant solution directives to strengthen security and privacy in the cloud environment. They discuss about the various threats like abusive use of cloud computing, insecure interfaces, data loss and leakage, identity theft and metadata spoofing attack.

The RepMec (Replica Metadata) catalog developed by the European DataGrid's Reptor project [19] is similar in its design and function to MCS. The RepMec catalog is built upon the Spitfire database service. The RepMec catalog stores logical and physical metadata. Among other functions, this catalog is used within the EDG project to map from user-provided logical names for data items to unique identifiers called GUIDs. RepMec is used in the Reptor system in cooperation with a replica location service.

3. METHODOLOGY

This research shall be carried out in six steps in order to formulate the MetaaS model that could enabling the cloud users and cloud providers to search, store, access, retrieve, publish, and capture the data from CDS. These steps summarized as follow:

3.1 Performing A Review of the Related Literature

This step is involving the reviewing the MetaaS and cloud computing to ensure that the proposed model could achieve its mission statement.

3.2 Conducting the Preliminary Survey

The step of conducting the preliminary survey has been defined to: Firstly, to assist in formulating the model. Secondly, to adopt the Metadata components to be fit in cloud storage. For these purposes a survey has been done using a questionnaire to those who are involve in the project cloud computing integrated Metadata such as a researchers, cloud providers, cloud users and Metadata experts. So they were asked to verify the model components and inputs.

3.3 Model Formulation

This is the process involving the composition of attributes and its elements based on the previous steps into a specific format or manner. This step is the process of formulating the model into its component-based system with regards on MetaaS functionality together with cloud storage.

3.4 Model Implementation

MetaaS model is implemented using code to specify how CSPs should publish metadata.

Journal of Theoretical and Applied Information Technology
31st August 2016. Vol.90. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.5 Model Evaluation

This step is the process of evaluation that involved another round of questionnaire called post-survey in order not only to verify and validate the model, but also a part to enhancing of a comprehensive model specification.

3.6 Conclusion

This step is the process of summarizing the finding of the pre and post survey that has been done on in producing the MetaaS model for enabling search, store, access, retrieve and capture the data from cloud storage.

4. PROPOSED METADATA AS A SERVICE MODEL

The overall model of MetaaS for enabling searching, storing, accessing, publishing, and capturing the data from CDS is proposed as shown in Fig. 1. The model is consists of three layers as metadata components, cloud users, and CSPs.

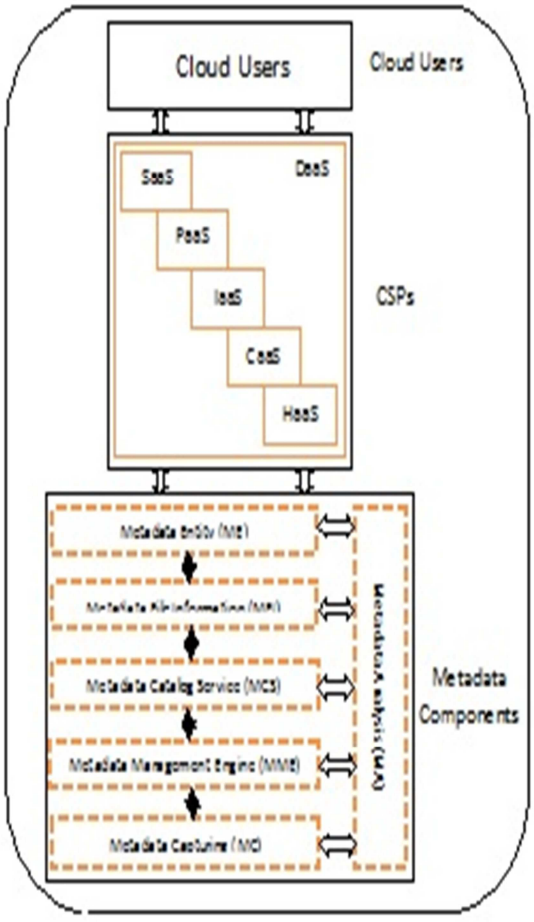

Figure 3: Metadata as a Service (MetaaS) Model

The layers of our model summarized as follow:

4.1 Metadata Components:

4.1.1 Metadata entity (ME)

The metadata entity in cloud is the core part of the model basis. The metadata entity is divided into two categories: descriptive metadata, and administrative metadata. Descriptive metadata includes resource description metadata, capability description metadata and service description metadata. Resource description metadata is a summary list of capability and service of the model, describing available resources of the entire model. Administrative metadata includes the capability management metadata, the service management metadata and the control metadata.

4.1.2 Metadata file information (MFI)

Every file in cloud storage contains metadata that tracks things like where it was saved, when it was last accessed or modified, etc.

4.1.3 Metadata catalog service (MCS)

Metadata Catalog Service in the cloud is a mechanism for storing and accessing descriptive metadata and allows cloud users to query for data items based on desired attributes. MCS may be used for storing and accessing metadata about cloud logical files.

MCS provides a synchronous java client application programming interface. The client API provides the following operations:

- Querying the catalog for logical objects based on object attributes
- Querying the static attributes of a logical object
- Querying the user defined attributes of a logical object
- Querying the contents of a logical view or a logical collection
- Creating a logical file, collection or a view
- Modifying the attributes of a logical object
- Deleting a logical file, view or a collection
- Annotating a logical object
- Adding logical objects to a view

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Metadata Service is more than simply a database service that stores metadata attributes. Rather, it is a specialized service that includes the following components:

- ✓ A data model that includes mechanisms for aggregation of metadata mappings
- ✓ A standard schema for domain-independent metadata attributes with extensibility for
- ✓ additional user-defined attributes
- ✓ A set of standard service behaviors
- ✓ Query mechanisms for accessing the database
- ✓ A set of standard interfaces and APIs for storing and accessing metadata
- ✓ A set of policies for consistency, access control and authorization, and auditing

4.1.4 Metadata management engine (MME)

Metadata management engine is institution of control and scheduling of model, completes unified monitoring, management and coordination work of capability and service and other resources. Metadata management engine controls the capability and service through the administrative metadata.

4.1.5 Metadata capturing (MC)

When cloud data is modified by the provider, catalog service needed to be changed in such a way that the data and catalog service remain consistent with each other. If catalog service is on the cloud, they are collected in the knowledge base using cloud tools. In order to speed up the whole change propagation process, only the catalog service is that depend on the change are gathered. This dependency data is obtained from the catalog service of the cloud that represents the performed change. Moreover, the output of this step is one list that makes references between Metadata catalog service and CDS.

4.1.6 Metadata Analysis (MA)

In this step, automatic translation of the catalog service is performed according to the changes in the data in the cloud. In order to avoid overhead of the cloud storage, which may heavily increase if the changes are performed every time the cloud storage has to be modified.

4.1.7 Cloud user

Cloud users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations. Cloud user that uses the cloud computing services. A cloud consumer represents a person or organization that maintains a business relationship with, and uses the service for a CSP. Also, the end user that actually uses the services, whether it is Software, Hardware, Platform or Infrastructure as a Service.

4.1.8 Cloud service provider (CSP)

CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live cloud computing system. A CSP also defined as a service provider that offers customers storage or software services available via a private (private cloud) or public network (cloud). Usually, it means the storage and software is available for access via the Internet.

5. METAAS IMPLEMENTATION

Publishing Metadata allows cloud users to search, store, access, retrieve, and capture the data from CDS using a WS-Transfer GET request or an HTTP/GET request using the ?wsdl query string. To insure that the code is working, a basic cloud service is created. Our proposed MetaaS model is implemented using a code. Cloud provides a rich infrastructure for exporting, publishing, retrieving, and importing service metadata. Cloud services use metadata to describe how to interact with the service's endpoints so that tools, such as svcutil.exe, can automatically generate cloud user code for accessing the service. Fig. 2 illustrated the a self-hosted cloud service code using MetaaS.

Cloud services do not search, store, access, retrieve, capture, and publish metadata by default. To publish metadata for cloud service you must explicitly enable metadata publishing by adding metadata endpoints to your cloud service. Leaving metadata publishing disabled reduces the attack surface for the cloud service and lowers the risk of unintentional information disclosure. Not all cloud services must publish metadata. In case of metadata is not published, consider leaving it turned off. Note that you can still generate metadata and client code directly from your cloud service assemblies.

Cloud services publish metadata by exposing one or more metadata endpoints. Publishing cloud service metadata makes cloud service metadata available using standardized protocols, such as

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MEX and HTTP/GET requests. Metadata endpoints are similar to other service endpoints in that they have an address, a binding, and a contract. Metadata endpoints added to a service host in code as shown in Figure 2.

To search, store, access, retrieve, capture, and publish metadata endpoints for cloud service, you must first add an instance of the Cloud Service Metadata Behavior service behavior to the service as shown in Figure 2. Adding a System Service Model Description Cloud Service Metadata Behavior instance to your service augments your service with the ability to publish metadata by exposing one or more metadata endpoints. Once you add the System Service Model Description Cloud Service Metadata Behavior service behavior you can then expose metadata endpoints that support the MEX protocol or metadata endpoints that respond to HTTP/GET requests.

```
using Cloud System;
using Cloud System.Runtime.Serialization;
using Cloud System.ServiceModel;
using Cloud System.ServiceModel.Description;

namespace Metadata.Samples
{
   [CloudServiceContract]
   public interface ISimpleCloudService
   {
      [OperationContract]
      string SimpleMethod(string msg);
   }

   class SimpleCloudService : ISimpleCloudService
   {
      public string SimpleCloudMethod(string msg)
      {
         Console.WriteLine("The caller passed in " + msg);
         return "Hello " + msg;
      }
   }
```

Figure 2: Self-Hosted Cloud Service Code using MetaaS

The following code example shows the implementation of a basic cloud service that publishes metadata for the CSP in code as shown in Fig. 3.

The cloud users and CSPs be able to acquire, store, search, retrieve, publish and capture the metadata from the CDS using the code in Fig 3. The detail of this code in Fig 3 can be summarized in following steps:

- Within the main method of a console application, instantiate a CloudService Host object by passing in the service type and the base address.
(
CloudServiceHost svcHost = new CloudServiceHost(typeof(SimpleCloudService),
newUri("http://localhost:8001/MetadataSample"));
- Create a try block immediately below the code for step above, this catches any exceptions that get thrown while the service is running.
(
try
}
)
- Check to see whether the service host already contains a Cloud Service Metadata Behavior, if not, create a new Cloud Service Metadata Behavior instance
(
// Check to see if the service host already has a CloudServiceMetadataBehavior
CloudServiceMetadataBehavior smb = svcHost.Description.Behaviors.Find<CloudServiceMetadataBehavior>();
// If not, add one
if (smb == null)
smb = new CloudServiceMetadataBehavior();
)
- Set the HttpGetEnabled property to true.
(
smb.HttpGetEnabled = true;
)
- The Cloud Service Metadata Behavior contains a Metadata Exporter property. The Metadata Exporter contains a Policy Version property. Set the value of the Policy Version property to Policy15. The Policy Version property can also be set to Policy12. When set to Policy15 the metadata exporter generates policy information with the metadata that" conforms to WS-Policy 1.5. When set to Policy12 the metadata exporter generates policy information that conforms to WS-Policy 1.2.
(
smb.MetadataExporter.PolicyVersion = PolicyVersion.Policy15;
)
- Add
the CloudServiceMetadataBehavior instan

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ce to the service host's behaviors collection.
(
svcHost.Description.Behaviors.Add(smb);
)

- Add the metadata exchange endpoint to the service host.
(
// Add MEX endpoint
svcHost.AddCloudServiceEndpoint(
CloudServiceMetadataBehavior.MexContractName,
MetadataExchangeBindings.CreateMexHttpBinding(),
"mex"
);
)
- Add an application cloud endpoint to the service host.
(
// Add application endpoint
svcHost.AddCloudServiceEndpoint(typeof(ISimpleCloudService), new WSHttpBinding(), "");
)
- Open the service host and wait for incoming calls. When the cloud user presses ENTER, close the service host.
(
// Open the service host to accept incoming calls
svcHost.Open();
// The service can now be accessed.
Console.WriteLine("The cloud service is ready.");
Console.WriteLine("Press <ENTER> to terminate cloud service.");
Console.WriteLine();
Console.ReadLine();
// Close the CloudServiceHostBase to shutdown the service.
svcHost.Close();
)
- Build and run the console application.
- Use Internet Explorer to browse to the base address of the service (http://localhost:8001/MetadataSample in this sample) and verify that the metadata publishing is turned on. You should see a Web page displayed that says "Simple Service" at the top and immediately below "You have created a service." If not, a message at the top of the resulting page displays: "Metadata publishing for this service is currently disabled."

```
using Cloud System;
using Cloud System.Runtime.Serialization;
using Cloud System.ServiceModel;
using Cloud System.ServiceModel.Description;

namespace Metadata.Samples
{
   [CloudServiceContract]
   public interface ISimpleCloudService
   {
      [OperationContract]
      string SimpleMethod(string msg);
   }

   class SimpleCloudService : ISimpleCloudService
   {
      public string SimpleCloudMethod(string msg)
      {
         Console.WriteLine("The caller passed in " + msg);
         return "Hello " + msg;
      }
   }
   class Program
   {
      static void Main(string[] args)
      {
         CloudServiceHost svcHost = new
CloudServiceHost(typeof(SimpleCloudService), new
Uri("http://localhost:8001/MetadataSample"));
         try
         {
            // Check to see if the service host already has a
CloudServiceMetadataBehavior
            CloudServiceMetadataBehavior smb =
svcHost.Description.Behaviors.Find<ServiceMetadataBehavior>(
);
            // If not, add one
            if (smb == null)
               smb = new CloudServiceMetadataBehavior();
            smb.HttpGetEnabled = true;
            smb.MetadataExporter.PolicyVersion =
PolicyVersion.Policy15;
            svcHost.Description.Behaviors.Add(smb);
            // Add MEX endpoint
            svcHost.AddServiceEndpoint(
              CloudServiceMetadataBehavior.MexContractName,
              MetadataExchangeBindings.CreateMexHttpBinding(),
              "mex"
            );
            // Add application endpoint

svcHost.AddCloudServiceEndpoint(typeof(ISimpleCloudService)
, new WSHttpBinding(), "");
            // Open the service host to accept incoming calls
            svcHost.Open();

            // The service can now be accessed.
            Console.WriteLine("The cloud service is ready.");
            Console.WriteLine("Press <ENTER> to terminate cloud
service.");
            Console.WriteLine();
            Console.ReadLine();

            // Close the CloudServiceHostBase to shutdown the
service.
            svcHost.Close();
         }
         catch (CommunicationException commProblem)
         {
            Console.WriteLine("There was a communication
problem. " + commProblem.Message);
            Console.Read();
         }
      }
   }
}
```

Figure 3: MetaaS Code

Journal of Theoretical and Applied Information Technology
31st August 2016. Vol.90. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Our proposed MetaaS model supports the metadata formats as shown in the Table-1.

Table-1. Metadata Formats

Protocol	Specification and usage
WSDL	Web Services Description Language (WSDL) MetaaS model uses Web Services Description Language (WSDL) to describe services.
XML Schema	XML Schema MetaaS model uses the XML Schema to describe data types used in messages.
WS Policy	Web Services Policy MetaaS model uses the WS-Policy 1.2 or 1.5 specifications with domain-specific assertions to describe service requirements and capabilities.
WS Policy Attachments	Web Services Policy Attachment MetaaS model implements WS-Policy Attachments to attach policy expressions at various scopes in WSDL.
WS Metadata Exchange	Web Services Metadata Exchange MetaaS model implements WS-MetadataExchange to retrieve XML Schema, WSDL, and WS-Policy.
WS Addressing Binding for WSDL	Web Services Addressing MetaaS model implements WS-Addressing Binding for WSDL to attach addressing information in WSDL.

6. RESULTS AND DISCUSSION

The MetaaS model in cloud computing has been gone though the steps that has been specified in the introduction as well as in the model sections. Based on this, there is a significant result shown that the MetaaS model should accommodated the following features or components, in order to become relevant to serve the cloud user and CSP in cloud computing environment.

Metadata in cloud computing can be associated with one representation of cloud entities and cloud component. The most common is to use is to map between metadata components and CDS. Examples of cloud frameworks which use this kind of mapping are Gaelyk [20] which makes use of metadata that associated to CDS, metadata components and cloud services and inject various cloud services to a cloud user. Objectify which implement a metadata directly to a CDS without identifying the default of cloud behavior. The framework Spring Data also propose the mapping between interface methods of metadata to database quires using code conventions. It is important to state that this solution is not exclusive for mapping to CDS. For instance, when mapping to a cloud services, a metadata components could be mapped to a cloud services and entities.

6.1 MetaaS Based-Cloud Services Functionality

Based on the cloud services functionality component in terms of level of type of its requirement at the first place, respondents are agreed that the MetaaS of DaaS is occupied 100% due to huge amount of data in cloud storage, PaaS is occupied 70%, IaaS is occupied 40%, SaaS is occupied 50%, HaaS is occupied 10%, and CaaS is occupied 30% as shown in Fig. 4.

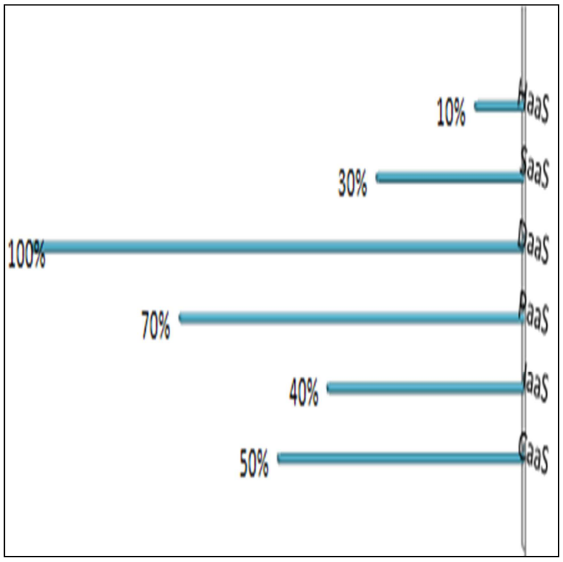

Figure 4: The Agreement Level of Cloud Services Functionality

6.2 MetaaS Components

Based on the MetaaS component in terms of level of type of its requirement at the first place. Respondents are agreed that the Metadata Entity (ME) is occupied 80%, Metadata File Information (MFI) is occupied 60%, Metadata Catalog Service (MCS) is occupied 100%, Metadata Management Engine (MME) is occupied 40%, Metadata Capturing (MC) is occupied 70% as shown in Fig. 5.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

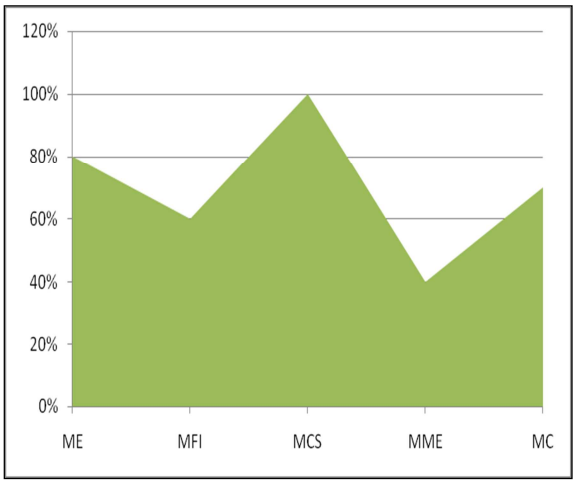

Figure 5: The Agreement Level of MetaaS Components

7. CONCLUSION

As a conclusion, this paper proposes a metadata as a service (MetaaS) model, based on the discussion and analysis of real needs which exist in cloud computing currently. The paper has shown that the MetaaS model is very important features to assist searching, storing, accessing, publishing, and capturing the data from cloud storage of PaaS, laaS, HaaS, CaaS, and SaaS, as indicated as DaaS in a cloud environment. In this context, the MetaaS model can be implemented by using three components which are: MetaaS components, cloud users, and CSPs. The finding is also shown that the cloud users can getting the collection of service of cloud project called MetaaS that has contributed a significant effect to those who are acquiring, storing, searching, retrieving, publishing, and capturing of the cloud data for a future purpose, which will help to minimize the search time of cloud user and fast access of the data. This will guarantee the minimization of the search time of cloud user, as well as guarantee the fast published data time of CSPs.

REFERENCES:

[1] S. Anjanadevi and D. Vijayakumar, "An Efficient Dynamic Indexing and Metadata Model for Storage in Cloud Environment," *Networking and Communication Engineering*, vol. 6, pp. 124-129, 2014.

[2] R. Abdullah, Z. D. Eri, and A. M. Talib, "A model of knowledge management system for facilitating knowledge as a service (KaaS) in cloud computing environment," *Proceedings of 2011 International Conference on Research and Innovation in Information Systems (ICRIIS 2011)*, pp. 1-4.

[3] A. M. Talib, R. Atan, R. Abdullah, and M. Azrifah, "CloudZone: Towards an integrity layer of cloud data storage based on multi agent system architecture," *Proceedings of IEEE Conference on Open Systems (ICOS 2011)*, pp. 127-132.

[4] A. M. Talib, R. Atan, R. Abdullah, and M. A. A. Murad, "Multi agent system architecture oriented prometheus methodology design to facilitate security of cloud data storage," *Journal of Software Engineering,* vol. 5, pp. 78-90, 2011.

[5] J. Yang and Z. Chen, "Cloud computing research and security issues," *Proceedings of international conference in Computational intelligence and software engineering (CiSE 2010)*, pp. 1-3.

[6] A. M. Talib, R. Atan, R. Abdullah, and M. A. Azmi Murad, "Security framework of cloud data storage based on Multi Agent system architecture-A pilot study," *Proceedings of International Conference on in Information Retrieval & Knowledge Management (CAMP 2012)*, pp. 54-59.

[7] A. M. Talib, R. Atan, R. Abdullah, and M. A. A. Murad, "Security framework of cloud data storage based on multi agent system architecture: Semantic literature review," *Computer and Information Science*, vol. 3, p. p175, 2010.

[8] Y. Xiao, G. Xu, Y. Liu, and B. Wang, "A Metadata-driven Cloud Computing Application Virtualization Model," *Journal of Computers*, vol. 8, pp. 1571-1579, 2013.

[9] M. Cammert, J. r. Kr¡¤mer, and B. Seeger, "Dynamic metadata management for scalable stream processing systems," *Proceedings of IEEE 23rd International Conference on in Data Engineering Workshop*, 2007, pp. 644-653.

[10] J.-J. Wu, P. Liu, and Y.-C. Chung, "Metadata partitioning for large-scale distributed storage systems," *Proceedings of IEEE 3rd International Conference on Cloud Computing (CLOUD)*, 2010, pp. 212-219.

[11] A. Verma and S. Venkataraman, "Efficient metadata management for cloud computing applications," 2010.

[12] Y. Hua, Y. Zhu, H. Jiang, D. Feng, and L. Tian, "Scalable and adaptive metadata management in ultra large-scale file systems," *Proceedings of 28th International*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Conference on Distributed Computing Systems, ICDCS'08, 2008, pp. 403-410.

[13] A. W. Leung, M. Shao, T. Bisson, S. Pasupathy, and E. L. Miller, "High-performance metadata indexing and search in petascale data storage systems," *Journal of Physics: Conference Series*, 2008, p. 012069.

[14] R. Anitha and S. Mukherjee, "A dynamic semantic metadata model in cloud computing," *Global Trends in Information Systems and Software Applications: Springer*, 2012, pp. 13-21.

[15] K. Ravi, and Revati, M., "Efficient Data Storage and Security In Cloud," *International Journal of Emerging trends In Engineering And Development*, vol. 6, 2012.

[16] K. Shizuka, Toshiyuki, A., and Chiemi W., "Performance Improvement of a Privacy-Preserving Query Method for a DaaS Model Using a Bloom filter," *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, Nevada, United States of America, 2011.

[17] M. K. Aguilera, M. Ji, M. Lillibridge, J. MacCormick, E. Oertli, D. G. Andersen, M. Burrows, T. Mann, and C. A. Thekkath, "Block-level security for network-attached disks," 2003.

[18] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of Cloud computing," *Journal of Supercomputing,* vol. 63, pp. 561-592, 2013.

[19] L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger, "Replica management in data grids," *Global Grid Forum*, 2002, pp. 278-280.

[20] E. M. Guerra and E. Oliveira, "Metadata-based frameworks in the context of cloud computing," *Computer Communications and Networks. Cloud Computing*: *Method and Practical Approach.* Springer, 2013, pp. 3-24.