Journal of Universal Computer Science, vol. 17, no. 8 (2011), 1135-1160
submitted: 15/3/10, accepted: 19/10/10, appeared: 28/4/11 © J.UCS

An Ontology based Agent Generation for Information Retrieval on Cloud Environment

Yue-Shan Chang
(Dept. of Comp. Sci. & Info. Eng., National Taipei University, Taipei County, Taiwan
ysc@mail.ntpu.edu.tw)

Chao-Tung Yang
(Department of Computer Science, Tunghai University, Taichung City, Taiwan
ctyang@thu.edu.tw)

Yu-Cheng Luo
(Institute of Communication Engineering, National Taipei University, Taipei County, Taiwan
humor718@gmail.com)

Abstract: Retrieving information or discovering knowledge from a well organized data center in general is requested to be familiar with its schema, structure, and architecture, which against the inherent concept and characteristics of cloud environment. An effective approach to retrieve desired information or to extract useful knowledge is an important issue in the emerging information/knowledge cloud. In this paper, we propose an ontology-based agent generation framework for information retrieval in a flexible, transparent, and easy way on cloud environment. While user submitting a flat-text based request for retrieving information on a cloud environment, the request will be automatically deduced by a Reasoning Agent (RA) based on predefined ontology and reasoning rule, and then be translated to a Mobile Information Retrieving Agent Description File (MIRADF) that is formatted in a proposed Mobile Agent Description Language (MADF). A generating agent, named MIRA-GA, is also implemented to generate a MIRA according to the MIRADF. We also design and implement a prototype to integrate these agents and show an interesting example to demonstrate the feasibility of the architecture.

Keywords: Ontology, Agent Generation, Information Retrieval, Cloud Computing
Categories: H.3.3, H.3.4, H.4.1, M.8, D.2.2

1 Introduction

With the advance of networked devices and more and more applications on the Internet, the cloud computing obtains increasingly attention and becomes one of most important research topic recently [Armbrust et al. 09, Geelan 08, Vouk 08, Koehler et al. 10]. What is the cloud computing? There is a great diversity of definitions regarding to the Cloud Computing [Geelan 08]. Cloud concept is still changing and these definitions show how the Cloud is conceived today: Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization [Vaquero et al. 09]. Besides, McFedries [McFedries 08] described data center (conceived as a

huge collection of clusters) as the basic unit of Cloud offering huge amounts of computing power and storage by using spare resources. This is related to the concept of massive data scalability proposed by Hand [Hand 07].

However, most of ongoing works or researches are aiming at developing the techniques and constructing cloud platforms, such as Amazon, Google AppEngine, Microsoft Azure, and Manjrasoft Aneka [Vecchiola et al. 09]. Massive data in data center of cloud platform can provide benefits to the cloud provider and consumer in retrieving or seeking information among business, medical information, and cooperative information retrieval platform [Golovchinsky et al. 09, Chow et al. 09]. In such environment, constructing an intelligent framework for fast analyzing user request and compositing the operations of retrieving is necessary. It does obtain increasingly attention and become an important research issue in supporting information retrieval and knowledge extraction in cloud platform [Chow et al. 09, Jaeger et al. 08, Li et al. 10, Wöhrera et al. 05, Cerri 08] to building Information and Knowledge Cloud. Such concept is similar to the knowledge grid [Cannataro and Talia 03]. Nevertheless, Delic et al. [Delic and Riley 09] from Hewlett-Packard Co. concluded that: "We believe that enterprise intelligence will draw its capacities from the Enterprise Knowledge Clouds (EKC) embedded in the global, dependable fabrics consisting of subjects, objects and devices. This may yet evolve into a 'social computing' paradigm as the likely advanced form of future society. Massive collaboration (on content tagging, for example) followed by the emergence of ontologies based on the Semantic Web, and adjusted by the folksonomies developed as user-oriented Web 2.0 applications, will embody 'collective intelligence' as the new source of knowledge."

In addition, they also regarded the Enterprise Knowledge Management as the business applications of data center. It is expected to exploit the technology of knowledge grid to such applications to benefit the Business Grid [Delic and Riley 09] come into being. Once the technology maturity, the EKC will be the super-structure of business grids connecting multiple data centers to support diverse applications and functions. Therefore, an effective approach for retrieving desired information or extracting useful knowledge is an important issue in the knowledge cloud.

Many knowledge discovery projects in the grid community [Cannataro et al. 04, Cannataro et al. 02, Krishnaswamy et al. 04, Luo et al. 07, Stankovski et al. 08, Wang et al. 06b, Shih et al. 10] had been proposed. From the observation of these researches and others [Qu et al. 08, Glitho et al. 02], it showed that information retrieval and knowledge discovery exploiting mobile agent technology [Manvi et al. 09, Waluyo et al. 05] is a popular approach in recent years. Besides, retrieving information or knowledge from such environment is requested to be familiar with its schema, structure, and architecture of cloud node, which against the concept and characteristics of cloud platforms. Nevertheless, composing a mobile agent always is a tedious work, especially on cloud environment with probably heterogeneous and hidden characteristics. Therefore, existing knowledge grid platforms are not entirely suitable for the EKC platform. Agent generation is an important technique in a variety of domains, such as information retrieval, distributed intrusion detection system, and network management etc. Some well-developed researches have been presented [Wang et al. 06a, Zhang et al. 07, Lage et al. 04, Chang et al. 02]. Existing researches

are necessitating somehow intervention of programmers or users in agent generation process.

In addition, as well-known, ontology [Shih et al. 09, Flahive et al. 05, Flahive et al. 06, Flahive et al. 09, ZadJabbari et al. 10] is one of most widely applied approach in enabling system intelligence and improving the automation ability of system by obtaining system semantics [Bhatt et al. 04a, Bhatt et al. 04b, Martino 09, Bhatt et al. 06, Lanzenberger et al. 10]. In order to solve the problems mentioned above, in this paper, we propose an ontology-based Agent Generation framework to dynamically and fully automatically generate mobile agent for retrieving desired information on cloud platforms. The framework can assist users to automatically generate mobile agents for retrieving desired information spread over cloud platforms without user's intervention. The framework is integrated into the AIR [Chang et al. 08], which is our previous work. The framework can be smoothly integrated, by referred to [Aversa et al. 09], with various cloud platforms, such as Google AppEngine, by means of slightly modifying the AIR and the framework.

While user submitting a flat-text based request for retrieving information on a cloud environment, the request will be automatically deduced by a Reasoning Agent (RA) based on predefined ontology and reasoning rule, and then be translated into a Mobile Information Retrieving Agent Description File (MIRADF) that is formatted in a proposed Mobile Agent Description Language (MADF) that defines the behavior of mobile IR agent. A generating agent, named MIRA-GA, is also implemented to generate a MIRA according to the MIRADF. In addition, the Knowledge Synthesis Agent in the architecture is designed to collect and synthesize information from clouds. We also design and implement a prototype to integrate these agents and show an interesting example to demonstrate the feasibility of the architecture.

The remainder of the paper is organized as follows. Section 2 surveys related works about agent generation, distributed data mining and knowledge grid. Section 3 describes the architecture used in the paper and presents the internal components and operation flows. Section 4 depicts and the formal model of MIRA. Section 5 depicts the implementation issues and MIRA generating process. Section 6 shows an example to demonstrate the approach and gives some discussions about approach. And finally we give a conclusion in section 7.

2 Related Works

2.1 Mobile agent generation

Although mobile agent is a well-known technology and has been applied to various domain [Manvi et al. 09, Luo and Ni 09, Camponogara and Shima 10], implementing a mobile agent still is not a trivial work. The section presents some well known research result and their approach.

Wang et al. [Wang et al. 06a] presented a tool that automatically translated CPNs (Colored Petri Nets) that specify IDS (Intrusion Detection System) design into software intrusion detection agents in MAIDS (Mobile Agent Intrusion Detection System). Together with the translator, authors have developed to convert SFTs (Software Fault Trees) that model intrusions into the CPN for IDS design, this tool can automatically generate intrusion detection software agents from a high level

description of intrusions. Authors used formal methods to reduce design and implementation errors of IDSs. The MAIDS is a mobile agent based distributed IDS developed at the Information Security Lab in Iowa State University.

Zhang et al. [Zhang et al. 07] proposed an approach to automate the agent generation process, while utilizing the existing tools and mechanisms as much as possible, by means of using a drag-and-drop mechanism where the user can select components to plug in the system depending on the application. The authors adapted a quantitative model to perform a quantitative reasoning on how important a role instance is given its preference, its utility function and its current achievement.

Lage et al. [Lage et al. 04] described a method for automatically generating agents to collect hidden Web pages. This method used a pre-existing data repository for identifying the contents of these pages and took the advantage of some patterns that can be found among Web sites to identify the navigation paths to follow.

The similar approach applied to information retrieval is wrapper generation. Many approaches also have been proposed to deal with the problem. In our previous work [Chang et al. 02], we proposed a framework for automatically generating the wrappers that supports a unified interface meta-search system based on an XML data model and CORBA standard. In the work, we only focused on extracting fields of interest from returned documents and not on analyzing their content.

To retrieve information or to discover knowledge from grid and cloud platform need to specific algorithms. Therefore, how to generate mobile agents involving the algorithms needs specific approach. In the work, we will not only extend our approach [Chang et al. 02] to the work, but also utilize ontology and reasoning rule to automatically generate a MIRA, and apply the approach for information retrieval on cloud environment.

2.2 Data mining projects on grid computing

Alexander W¨ohrer [Wöhrera et al. 05] described a data integration system based on the wrapper-mediator approach, namely the Grid Data Mediation Service (GDMS), of the Grid-Miner project conducted in Vienna. The technology developed has been validated and tested on an advanced medical application addressing the treatment of traumatic brain injury (TBI) victims. The developed concepts have been implemented (but are not limited to) by reusing the OGSA-DAI Grid Data Service (GDS) as a framework to show their feasibility.

Wang et al. [Wang et al. 06b] proposed a mobile agent-based architecture for service-oriented knowledge grid to implement KG-services by utilizing mobile agent technology. The proposed architecture integrates MA technology and knowledge grid technology and improves the performance of knowledge grid such as reducing network traffic, enhancing reliability and intelligence of KG-Services.

Cannataro et al. [Cannataro and Talia 03] described a software platform for geographically distributed high-performance knowledge discovery applications called Knowledge Grid, which is designed on top of computational grid mechanisms, provided by grid environments such as Globus. The Knowledge Grid uses the basic grid services such as communication, authentication, information, and resource management to build more specific parallel and distributed knowledge discovery tools and services.

Luo et al. [Luo et al. 07] systematically analyzed the issues of Agent Grid and implemented an Agent Grid platform AGrIP which provides infrastructure for agent based DDM (Distributed Data Mining) on Grid environment. Furthermore, a user-friendly and extensible development toolkit VAStudio is presented to help users to develop real DDM applications.

3 Mobile Agent based information retrieval on Cloud

This section briefly depicts the system architecture used in the work, which is modified from the AIR [Chang et al. 08]. Detailed please refer to [Chang et al. 08]. The system is a three layers architecture involving Application Layer (AL), Context Layer (CL), and Physical Layer (PL) respectively, as shown in Figure 1. The AL contains mainly a User Agent (UA). Most of jobs in the system are leaving to the CL in order to diminish the loading of UA. The CL can be divided into three modules; which are Inference Module, Retrieving Algorithm Selection Module, and Knowledge Synthesis Module respectively. Each module consists of multiple agents in order to achieve specific goals. Following will describe the internal components and depict their operations.

Here we assume that all information retrieving agents know the structure of clouds' data center, such as the schema, location etc., and have the access right of the data center. The structure information of data center will be kept in the domain knowledge repository. These can be easily achieved if the data centers tied to the platform. Following briefly present the components of these modules.

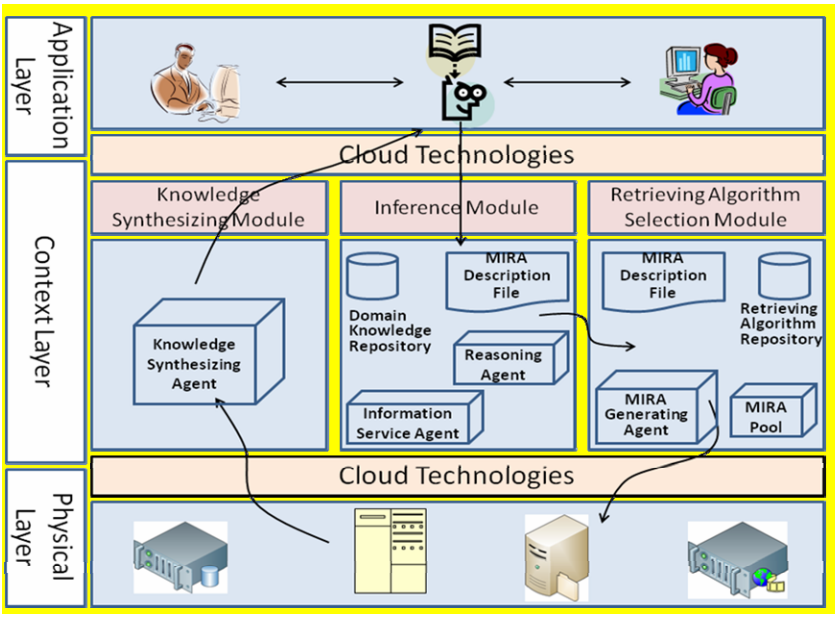

Figure 1: System architecture

3.1 Inference Module (IM)

The module consists of Reasoning Agent, Information Service Agent, and Domain Knowledge Repository.

– **Reasoning Agent (RA):**

The RA is responsible for reasoning the request from UA utilizing domain ontology and rules, which are kept in the Domain Knowledge Repository. While the RA receiving a request from the UA, the RA tokenizes the request and parses the tokenized request. If the tokenized request accommodates to the ontology, the RA will apply the associated rule to reason the request. If the parsing ontology and the application of associated rules are correct, the RA sends Information Service Agent (ISA) a query to retrieve information of clouds node for understanding where a MIRA can be dispatched. Finally, the RA will generate a MADF file according to the reasoning result and reply the MADF to the UA. The operations can refer to the Figure 2.

– **Information Service Agent (ISA):**

The ISA keeps the metadata of information and the resource of all cloud nodes. When a request is received by the ISA, it will reply the node ID and its associated information required by RA.

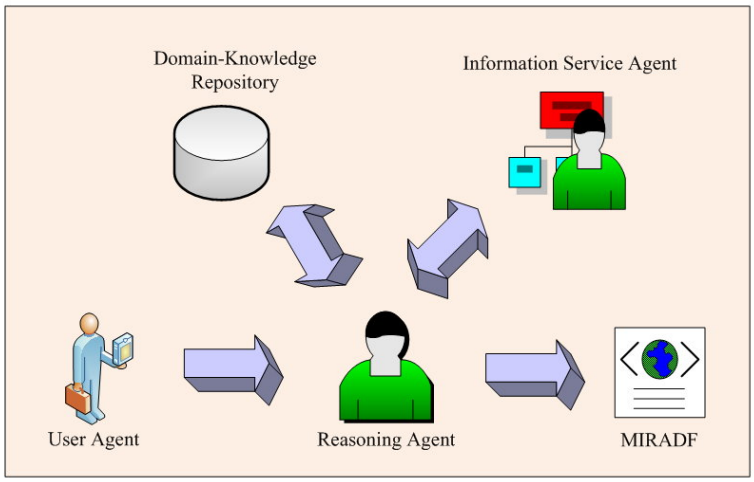

Figure 2: The components' relationship of IM

– **Domain-Knowledge Repository**

The repository is used to keep the ontology and associated rule. The ontology can be represented as Figure 3. In this paper, we construct a music-related ontology for demonstrating the feasibility of the system. In this prototype we adapt the Protégé

platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies to edit and construct the simple music ontology that can be exported into a variety of formats including RDF(S), OWL, and XML schema.

A reasoning rule involved in the RA is used to reason the correctness of request and to generate the MADL file that is used for generating MIRA. The rules can be constructed using the Jena reasoning engine that is a Semantic Web Framework for Java, which support semantics for defining the rule. In the Jena reasoning engine, the RA can use SPARQL query language to inquire the result that is reasoned from an OWL data model. We first construct a Jena reasoning engine, and then put the rule to the reasoning engine for future reasoning.

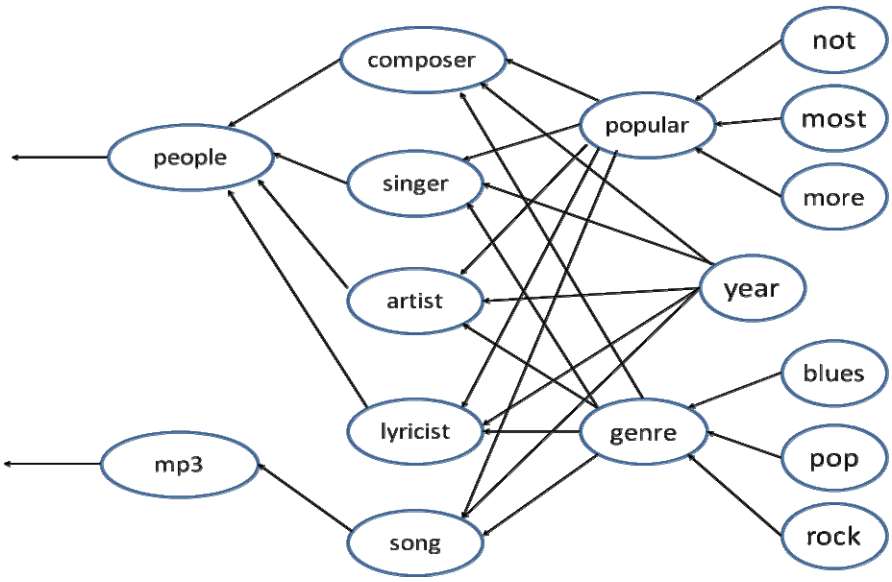

Figure 3: Partial Ontology of example

3.2 Retrieving Algorithm Selection Module

In the Retrieving Algorithm Selection Module (RASM) the main task is to select a proper retrieving algorithm according to the specification of MADL file to generate a mobile IR agent.

The RASM, as shown in Figure 4, consists of Mobile Information Retrieving Agent –Generating Agent (MIRA-GA) and Retrieving Algorithm Repository. The MIRA-GA is used for generating MIRA according to the MIDL from UA. The task of Retrieving Algorithm Repository is to store various algorithms that are to retrieve a variety of information.

– **Mobile Information Retrieving Agent – Generating Agent**

When the MIRA-GA receives a request from the UA, it will first get the MADF and parse the MADF to generate the MIRA. During the parsing phase, the MIRA-GA queries the metadata of Retrieving Algorithm Repository to select a suitable algorithm. The query can be a simple matchmaking process. Finally the MIRA-GA generates a MIRA by composing the retrieving algorithm and mobile agent stub.

– **Retrieving Algorithms Repository**

The management of Retrieving Algorithm Repository is designed and implemented as a hierarchy. The advantage of the design is easy to search and access the component stored in the repository. While the MIRA-GA parse MADL file line by line, the MIRA-GA can access those components stored in the repository by searching and matching the name of component in the hierarchy. The components are predesigned in the prototype. Of course it only can process and retrieve specific information.

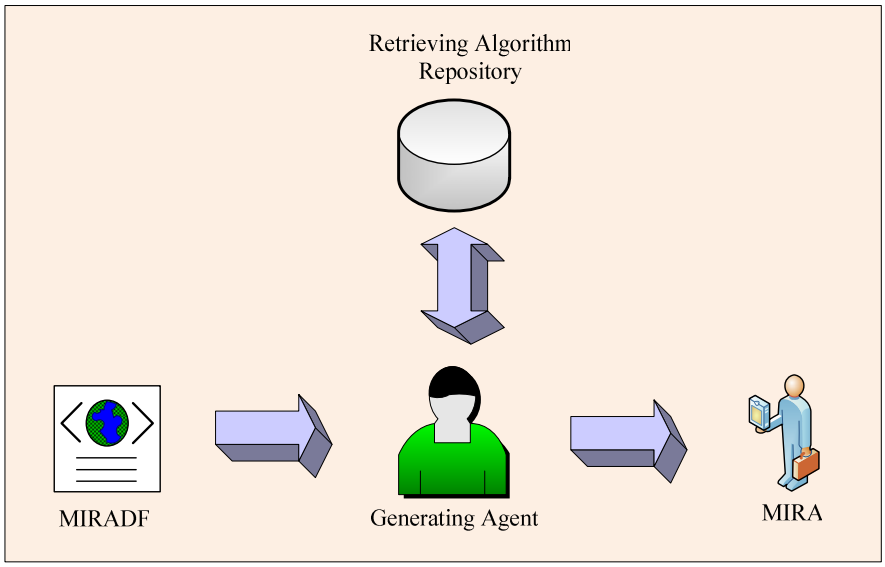

Figure 4: The components' relationship of RASM

3.3 Knowledge Synthesis Module

The module involves a Knowledge Synthesizing Agent (KSA) that is responsible for synthesizing and deducing global view knowledge collected from cloud platforms.

– **Knowledge-Synthesizing Agent (KSA)**

The KSA will get the reasoning rule that is constructed by Jena according to the User_ID, and deduce all of results from clouds. The reasoning process is similar to the RA. It is divided into four major steps. The first is to compose the rules. The rules can be constructed in the Jena reasoning engine. The second is to use SPARQL query language to inquire the result that is reasoned from an OWL data model. And then, we construct a Jena reasoning engine, and put the rule to the reasoning engine for future reasoning. Finally, we inquire the result and reply the result to UA.

4 MIRA model

Before presenting practical design and generation of Mobile Information Retrieving Agent, we first refer the navigation patterns [Lage et al. 04] to define the MIRA model to formulate the mobile information retrieving agent.

Definition 1 (Ontology): An ontology O is represented as O={C, ξ, R}, where C is a set of classes (concept), ξ is a set of relations, and R is a set of inference rules, respectively.

For example, as shown in Figure 3, the concept “pop” has the relation with the concept “genre” and the concept “genre” has the relation with the concept “singer”. Therefore we can deduce that the concept “pop” has the relation with “singer”.

Definition 2 (MIRA): A mobile information retrieving agent is denoted by a quintuple MIRA = {*I, Be, Ls, Ss, O*}, where *I* is an input pattern, *Be* is a set of behaviors on the target data of cloud nodes to obtain the information, *Ls* denotes a set of cloud nodes to be accessed for the request, which can be obtained by inquiring the ISA. *Ss* is a set of schema of cloud nodes and *O* is the output pattern for the request for user.

In addition, we also define a few of operations used in generating MIRA, as follows:

- **Tokenizing** *Tk(I, Ts):* To tokenize the input pattern *I* to a set of token *Ts*.
- **Querying** *Q(Ts, X, Ts’):* Using the token set *Ts* to inquire the information source *X* and obtain the new token set *Ts’*.
- **Deducing** *D(Ts’, Ro, L):* Deducing the rule of a specific ontology, *Ro*, for the token set *Ts’* to generate a set of elements in a language *L*.
- **Translating** *T(L, L’):* Translating a set of elements in a language to another set of elements in a new language.
- **Comparing** *C(Be, B):* Comparing the behaviors *Bs* with the universal behavior set *B*. The *B* is predefined and stored in a repository.
- **Merging** *M(<Bei> | Bei B):* Merging all behaviors into a archive.

The *Be* can be a combination of a sequence of primitives or algorithms stored in the Retrieving Algorithms Selection Repository. The *Be* can be obtained by deducing the input pattern using the ontology. The deducing can be found as follow:

$$\mathrm{I} \underset{\mathrm{O}}{\Rightarrow} \mathrm{B}e \tag{1}$$

where $Be = \{bi \mid bi \in BS\}$, *BS* is the universal set of retrieving algorithms.
and O is the ontology used to deduce the behaviors.

It is difficult to deduce the behaviors simply from the input pattern. The deducing needs to be extended using the operations defined above, as formula (2).

$$\mathrm{I} \overset{\mathrm{T}k}{\Rightarrow} Ts \underset{Ro}{\overset{D}{\Rightarrow}} Be \underset{RSM}{\overset{C}{\Rightarrow}} Be \tag{2}$$

Formula 2 shows the sequence of operations to derive to a set of behaviors. The input pattern first needs to be tokenized. Therefore, the RA can deduce the tokens to possible behaviors using rules of ontology *Ro*. And then utilize the Comparing operation to check the operation is valid.

Here gives a simple example. We can define the concept “most popular singer” as following: “A singer’s songs are downloaded that the number of times is maximum, we say that the singer is the one of the most popular singer.” or “A singer’s songs are downloaded is over 80% singers' songs, we call the singer is the one of the most popular singer.” Obviously, the former only count the maximum number of downloading, while the latter not only counting the number of downloading, sorting it, and then retrieving preceding 20% singers. In the case of former, the behaviors are counting, sorting, and then selecting maximum one. The behaviors *Be* is equal to {*counting*, *sorting*, *selecting maximum*}. In the case of latter, the behaviors are counting, sorting, and then selecting the preceding 20% singers. There is a little bit difference between them. Therefore, according to the definitions of the concept, we can get the behaviors *Be* based on the inference of the ontology.

Similarly, the *Ls* and the *Ss* also can be deduced from the input pattern.

$$\mathrm{I} \overset{\mathrm{T}k}{\Rightarrow} Ts \underset{Ro}{\overset{D}{\Rightarrow}} F \underset{ISA}{\overset{Q}{\Rightarrow}} Ls \underset{ISA}{\overset{Q}{\Rightarrow}} Ss \tag{3}$$

where the F is the file attribute that is a metadata of the file.

As shown in Figure 3, the reasoning agent deduces the file attribute utilizing the rules of ontology and knows that the information is stored in MP3-related files. Therefore, it will inquire the ISA in order to obtain cloud nodes' location that store the desired information. Of course, reasoning agent also can obtain cloud node's schema by means of inquiring the ISA.

Finally, the MIRA can be derived from a triple by the Translating and Merging operations, as formula 4.

$$\{Be, Ls, Ss\} \underset{RSM}{\overset{T}{\Rightarrow}} Be' \overset{M}{\Rightarrow} MIRA \tag{4}$$

5 Ontology-based Agent Generation

5.1 MADL

Automatic generating a mobile agent for information retrieval needs a flexible and scalable description language to describe the context of the mobile agent. A similar scheme and language to generate a mobile agent can refer to MASL [Ahmed 05]. Due to some lacks on the issue, such as standardization of language and extensibility for information retrieval, we design the MADL for automatic generation of a mobile information retrieval agent. The MADL is a XML-based description language designed as a simple and flexible language to describe the behavior of MIRA. The RA can understand what the user wants after completing the reasoning process and then generate a MIRADF formatted in the MADL to present the operations of MIRA. MIRA-GA will use the MIRADF to generate a MIRA. An example of MADL is shown in Fig. 5 and Fig. 6.

The structure of MIRADF includes five parts, as shown in Figure 5. There are *Agent_Rules*, *Agent_Name*, *Top_Stub_Code*, *Code_Snips*, and *Bottom_Stub_Code*, respectively. The *Agent_Rules* is derived by the RA from Inference Rule (R) defined in the Ontology. It is mainly to depict the main action of the MIRA. The *Agent_Name* shows the name of the MIRA. The *Top_Stub_Code* is aiming at the top level stub code. The *Code_Snips* is to describe the detailed behaviors in order to involve the retrieving algorithms. And the final is *Bottom_Stub_Code*, it contains the bottom level stub code. Each part involves individual description, such as shown in Figure 6. Here we utilize Figure 6 to simply explain the MADL. The details will be depicted in the next subsection.

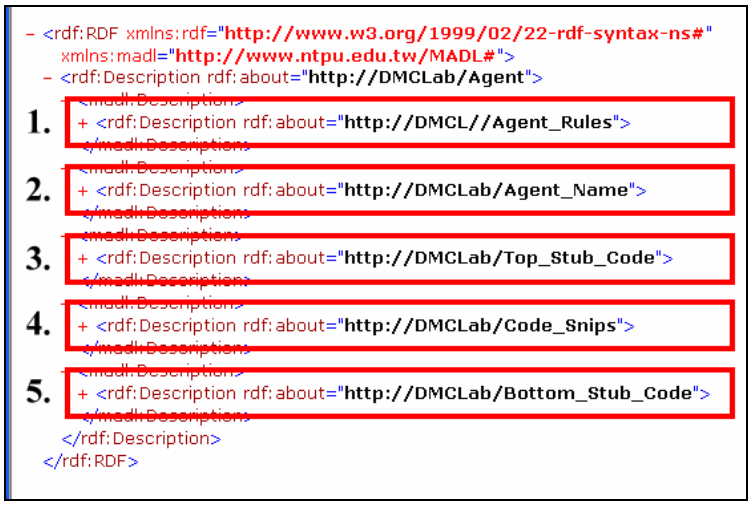

Figure 5: MIRADF structure

5.2 Ontology reasoning

Firstly, we assume that the Domain-Knowledge Repository will store all keywords that user may submit. And each deducible[1] keyword can be represented a concept by a class of ontology. These classes are correlated, for which is according to a set of predefined relations. The MIRA-GA can simply deduce the relationship using the rules, as mentioned in above section and shown in Figure 3. In order to demonstrate clearly the generation process of MIRA, we utilize a simple example to show the operations. While user submitting a flat-text based request for retrieving information on a cloud environment, for instance "Who is the most popular pop singer in 2008?", the request will be automatically deduced by a RA based on the predefined ontology and reasoning rules.

```
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:madl="http://www.ntpu.edu.tw/MADL#">
  - <rdf:Description rdf:about="http://DMCLab/Agent">
    - <madl:Description>
      - <rdf:Description rdf:about="http://DMCL//Agent_Rules">
          <madl:Who>people</madl:Who>
          <madl:People>Artist</madl:People>
        - <madl:Description>
          - <rdf:Description rdf:about="http://DMCLab/Properties">
              <madl:hasGenre>pop</madl:hasGenre>
              <madl:hasYear>2008</madl:hasYear>
              <madl:hasSingerCountValue>Value</madl:hasSingerCountValue>
              <madl:greaterThan>80</madl:greaterThan>
            </rdf:Description>
          </madl:Description>
          <madl:Popular>most</madl:Popular>
        </rdf:Description>
      </madl:Description>
    - <madl:Description>
      - <rdf:Description rdf:about="http://DMCLab/Agent_Name">
          <madl:Agent_ID>MIRA</madl:Agent_ID>
          <madl:User_ID>U01</madl:User_ID>
          <madl:Session_ID>S01</madl:Session_ID>
        </rdf:Description>
                                    :
                                    :
```

Figure 6: Partial example of MIRADF

The RA does two operations for generating the MIRADF. One is to tokenize the input request and to hash the tokenized keyword for quickly seeking the concept stored in the ontology, as shown in Figure 7. The other is to generate the MIRADF based on the reasoning using reasoning rules that are constructed using the OWL [Lin and Chen 07] language, as shown in Figure 8. In this step, the Jena[2] API is exploited to retrieve the ontology stored in the Domain-Knowledge Repository. The MIRA-GA

[1] There are existing keywords that are not deductively, such as "the", "a", "an", "is", etc.
[2] Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net/

also retrieves related entitles and OWL-based rules to deduce and to generate the MIRADF for the MIRA.

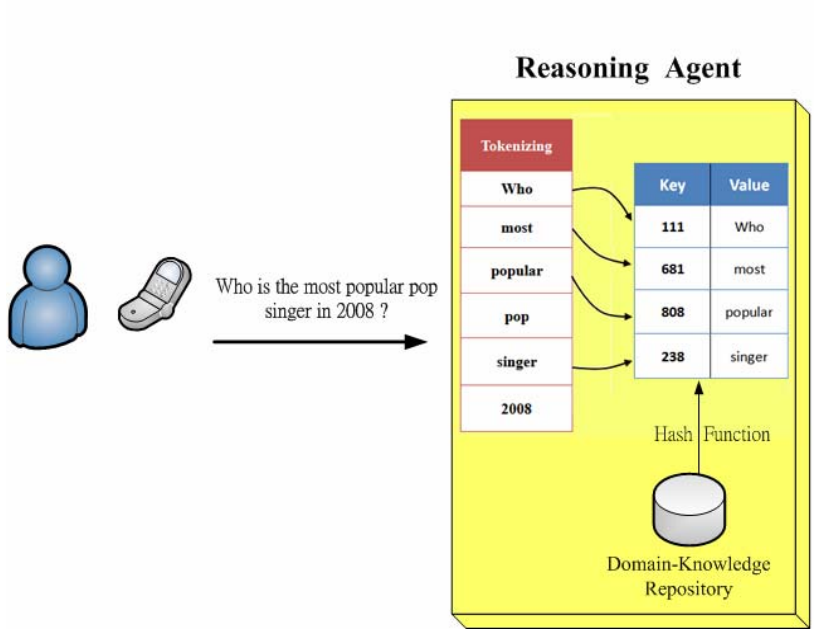

Figure 7: Tokenized user request

Following shows the reasoning algorithm for the MIRA in details.

1. *RA receives user query from UA.*
2. *RA carries out tokenization of user query.*
3. *After tokenization, RA uses Hash Function to map respective tokens to the ontology entities stored in HashMap that is prior constructed.*
4. *If all the mappings are correct, RA sends out an ACL INFORM message to UA. If not, RA replies an ACL NOT-UNDERSTOOD message instead.*
5. *RA sends out a request to ISA for getting resource metadata.*
6. *RA receives resource metadata from ISA.*
7. *RA acquires relevant ontology entities and inference rules based on user query.*
8. *RA obtains User_ID, Session_ID, Agent_ID, Behaviour and AgentAction.*
9. *RA combines all the elements listed in Steps 9 and 10 to generate a MIRA Description File.*
10. *RA sends out the MIRA Description File to UA, and the reasoning algorithm is complete.*

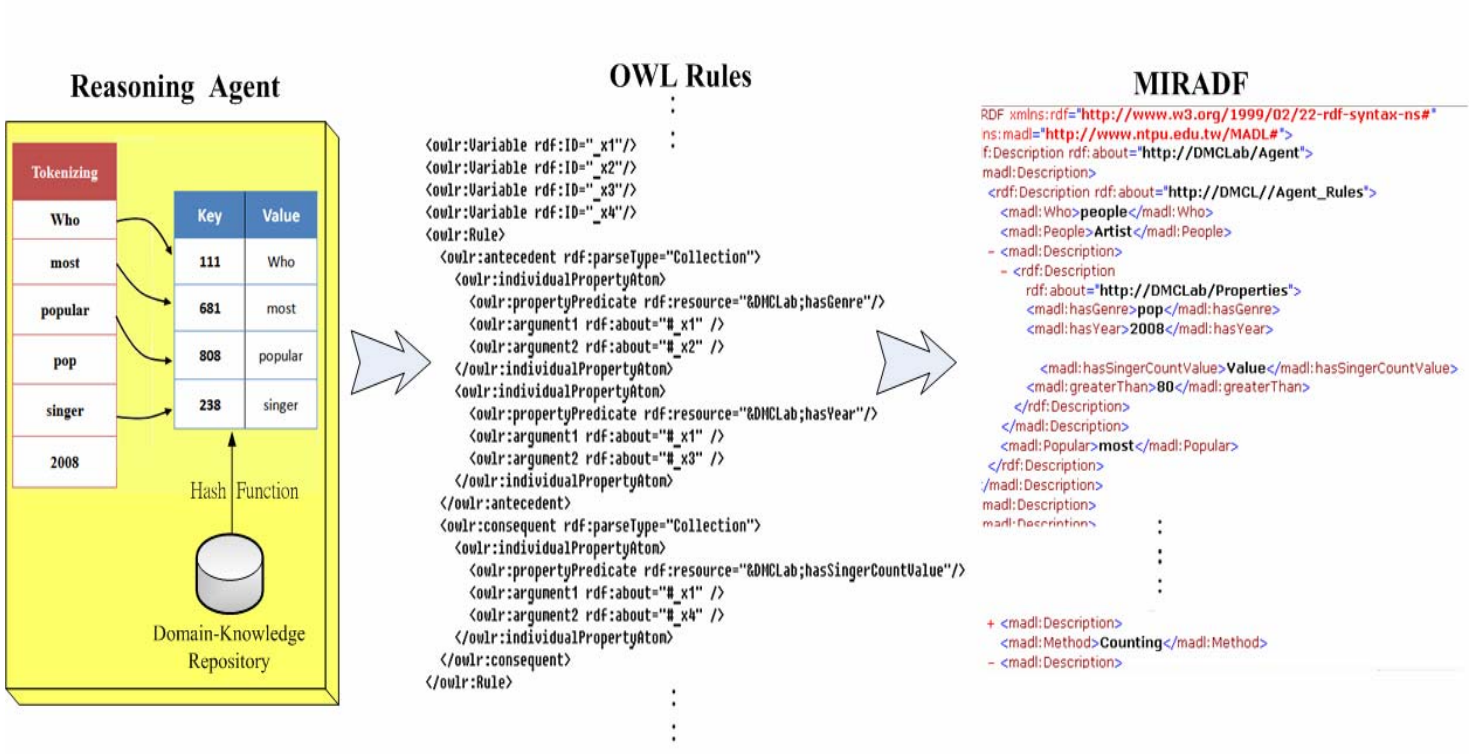

Figure 8: The process of generating MIRADF

5.3 Agent generation process

The next is to present the process of agent generation. Figure 9 shows the macro view of agent generation. The processes that will be dealt with by MIRA-Generating Agent to generate a MIRA consist of three steps; there are MIRADF parsing, Retrieving Algorithm selection, and MIRA generation, respectively. Next subsection will depict the processes in detail.

5.3.1 MIRADF Parsing

The first step of MIRA-Generating Agent is to parse and verify the validation of the MIRADF. If a part of MIRADF is verified by the parser, the parser will then invoke the process Retrieving Algorithm Selection and MIRA-Generation, as described in next, to compose a MIRA; otherwise, it will respond to the user agent the error. Here shows an example to demonstrate the parsing and the generating of the top stub and the bottom stub of a MIRA.

Figure 10 shows the *Top_Stub_Code* part in MIRADF. The *Top_Stub_Code* includes three parts; the *startup*, the *resource*, and the *behavior* between agents. The *startup* part shows the activities of initialization phase. The *resource* part indicates the nodes and the path of data. In the example, we assume the MIRA will be migrated to the node "DMCL-NTPU" and the data is stored in the "D:\Music\Music.csv". The information of resource can be inquired from ISA. And the *behavior* is the interaction scheme between agents. Because the most of code segments of a MIRA, such as top stub and bottom stub, are frequently used. When the parser parsed the part of *Top_Stub_Code*, it directly generated the source code that is compliant with JADE, as shown in Figure 11. The similar approach can be applied to other parts of MIRADF. The *Bottom_Stub_Code* part is simple than the *Top_Stub_Code*. Figure 12 shows the example of *Bottom_Stub_Code* part. The operation of the part is to release resources and close the agent. Here we neglect listing the generated source code of this part.

5.3.2 Retrieving Algorithm Selection

The next step is to properly select algorithms and code snips, which are regarding to the MIRA, after parsing the MIRADF. As the example shown in Figure 13, the MIRADF specifies what operations the MIRA do. Here we neglect to list the generated source code. Due to the possible diversity of algorithms for information retrieval, the algorithms have to be well organized for being properly selected. We refer the hierarchy approach proposed in the [Chang et al. 07] and construct a repository to easily, flexibly and effectively manage the algorithms.

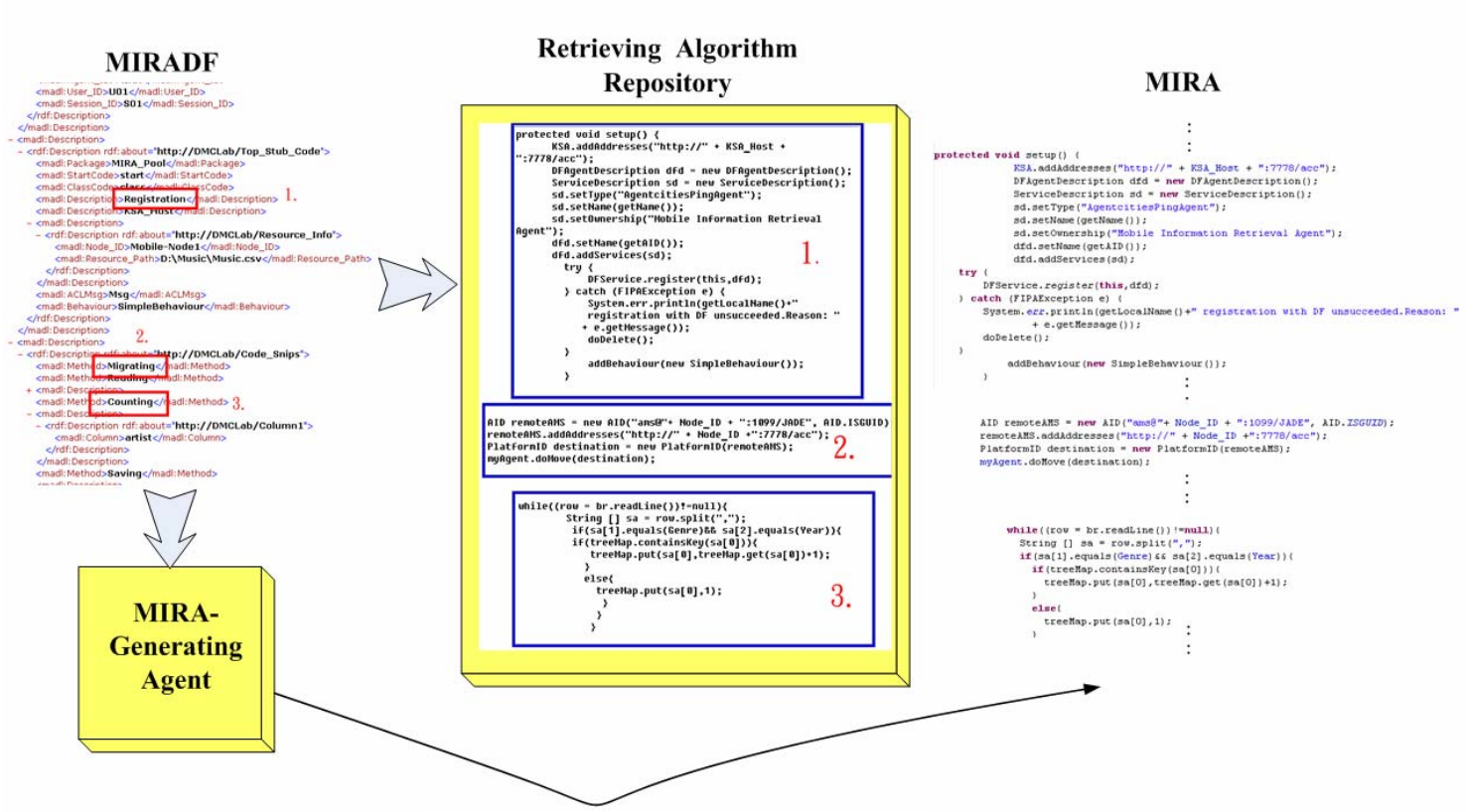

Figure 9: A macro view of agent generation

Figure 10: The Top-Stub in the MIRADF

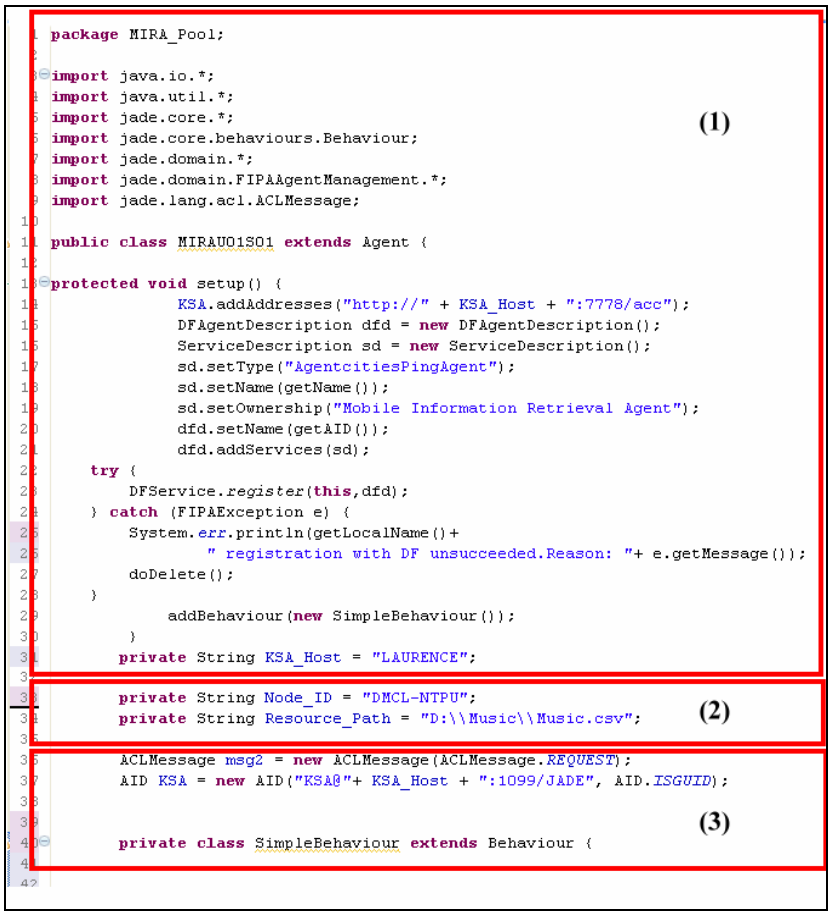

Figure 11: The generated source code for Top_Stub

```
- <madl:Description>
  - <rdf:Description rdf:about="http://DMCLab/Bottom_Stub_Code">
      <madl:SendMsg>Inform</madl:SendMsg>
      <madl:End_Code>End</madl:End_Code>
    </rdf:Description>
  </madl:Description>
```

Figure 12: The Bottom_Stub in the MIRADF

```
- <madl:Description>
  - <rdf:Description rdf:about="http://DMCLab/Code_Snips">
      <madl:Method>Migrating</madl:Method>
      <madl:Method>Reading</madl:Method>
    - <madl:Description>                                              (1)
      - <rdf:Description rdf:about="http://DMCLab/Column">
          <madl:Column>Genre</madl:Column>
          <madl:Column>Year</madl:Column>
        </rdf:Description>
      </madl:Description>
      <madl:Method>Counting</madl:Method>
    - <madl:Description>
      - <rdf:Description rdf:about="http://DMCLab/Column1">           (2)
          <madl:Column>artist</madl:Column>
        </rdf:Description>
      </madl:Description>
      <madl:Method>Saving</madl:Method>
    - <madl:Description>
      - <rdf:Description rdf:about="http://DMCLab/Column2">           (3)
          <madl:Column>artist</madl:Column>
          <madl:Column>Value</madl:Column>
        </rdf:Description>
      </madl:Description>
      <madl:Code>ReaderClose</madl:Code>
      <madl:Code>Output</madl:Code>                                   (4)
    </rdf:Description>
  </madl:Description>
```

Figure 13: The code description for retrieving information in MIRADF

5.3.3 MIRA Generation

This section summarizes the MIRA generation flows spread on above sections. Once the MIRADF is generated by the RA, all of operations are executed by the MIRA-GA. The MIRA-GA will execute following operations to generate a MIRA.

1. *Parsing the MIRADF line by line.*
2. *Extracting all elements described in tags.*
3. *Extracting the content between <madl:N> and </madl:N>. (Note: N can be "Who", "People", etc.)*
4. *According to the content from Step 3, MIRA-GA retrieves the corresponding code snips from Retrieving Algorithms Repository.*
5. *Combining all code snips to form MIRA.*
6. *Extracting the Agent_ID, User_ID, and Session_ID.*
7. *Combining all contents obtained from the previous step to generate a unique name for MIRA.*
8. *Generating the MIRA.*
9. *Dispatching the MIRA to relevant cloud node.*

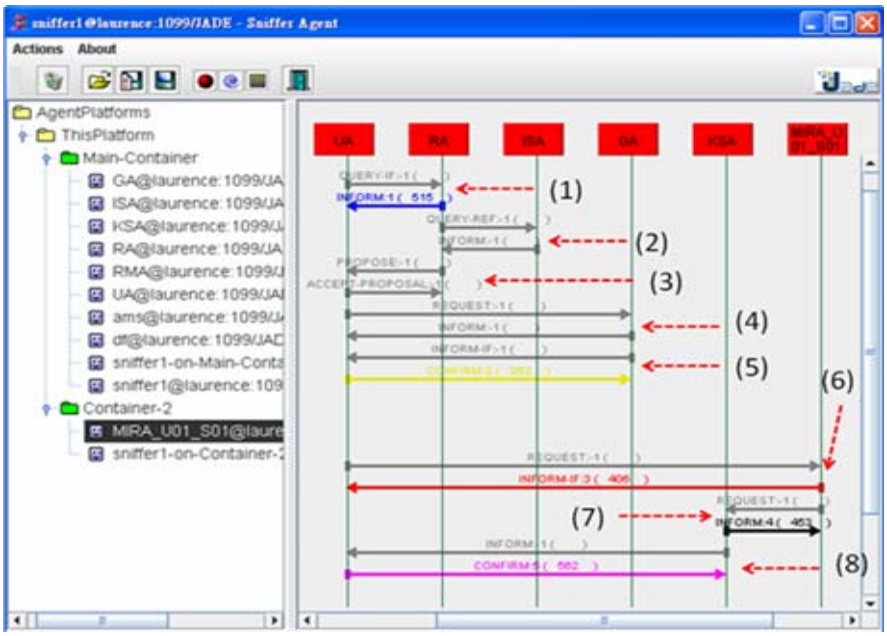

Figure 14: Scenario of the example

6 Experiments and Discussion

We utilize a well-known mobile agent development toolkit-Jade, to develop prototype of agent-based service network [Genco et al. 06] used in platform. All of agents are run on the Jade platform, as shown in Figure 14. In the prototype, we utilize a desktop with an Intel Quad 2.39GHz CPU and 2GB RAM and a Intel Pentium Dual CPU, 1.86GHz, 2.00GB RAM, Windows XP to emulate two cloud nodes, run the framework on a laptop with Intel Pentium Dual CPU, 1.86GHz, 2.00GB RAM,

Windows XP, and run the UA on a UMPC (Utra Mobile Personal Computer) with Intel processor 800MHz, 1024MB RAM, Windows VISTA. All are connected by a 100Mbps ethernet. In addition, an interesting example is shown to demonstrate the feasibility of the approach.

6.1 Experiments

In this example, we assume each cloud node is with some dataset, in which at least one of the data set store music related data and its metadata, such as *File Path, File Name, File Size, Artist, Year, Genre, Play Time*, etc. If someone wants to know that who is the most popular pop singer in 2008? And then he sends a request to the cloud environment. The environment can automatically generate and send a mobile agent to those nodes, which store the accumulated information of downloaded MP3 file, and retrieve and count the desired information. And then respond the result to KSA. KSA will collect responses from all cloud nodes and synthesize all responses to form a global view. Finally KSA send the global view of result to UA.

In the example, we first implement a UA, as shown in Figure 15(a). User can input a flat-based text and submit the request to the platform, such as "Who is the most popular pop singer in 2008?" In the scenario, as shown in Figure 12, the UA first sends a request to the RA (1) and the RA replies an acknowledgement to the UA. The RA sends a request to inquiry the ISA to get the metadata of available cloud nodes (2) and then the ISA replies the result. When the RA completes the reasoning of request and generates a MIRADF file, it will send the file to the UA (3) to show that the request is correct and can be inquired and retrieve information from cloud node.

After the UA gets a MIRADF file, it can send the file to MIRA-GA (short as GA) to generate a MIRA (4) and then the GA replies an acknowledgement to UA. In the (5) the GA sends the generated MIRA to the UA. In step (6), the UA dispatches the MIRA to all candidate cloud nodes to retrieve desired information. These MIRAs in the cloud nodes will retrieve and send the result to the KSA for synthesizing the result and form a global view of information (7). Finally, the KSA will collect all of responses from cloud nodes, construct a global view of the information, and send the global view to the UA (8). The final result is shown in Figure 15(b).

(a) (b)

Figure 15: The user interface and UA

In this example, we measure the latency consumed by all agents, as shown in Table 1. The total latency is less than 3 (2.844 s) second. We think that the latency is reasonable because most of the latency are consumed by generating a MIRA (the time

of RA plus the time of MIRA-GA). The MIRA only consume a bit of latency. It is reasonable because the RA and MIRA-GA need to deduce some of rules and combine some of code segments to generate the mobile agent. From the measurement, the overhead seems that is a little bit heavy. But when the number of cloud node increasing, the overhead will decrease greatly.

Table 1 shows the RA will consume most of total time in the process of MIRA generation. And the most of time consumed in the RA will be to deduce the rules of ontology. Figure 16 shows the latency on the rule deducing. Obviously, the latency of rule deducing will slightly increase with the increasing of number of rule. It also is reasonable.

RA	ISA	MIRA-GA	KSA	MIRA
1500	1	828	469	46

Table 1: Time consuming of the example.

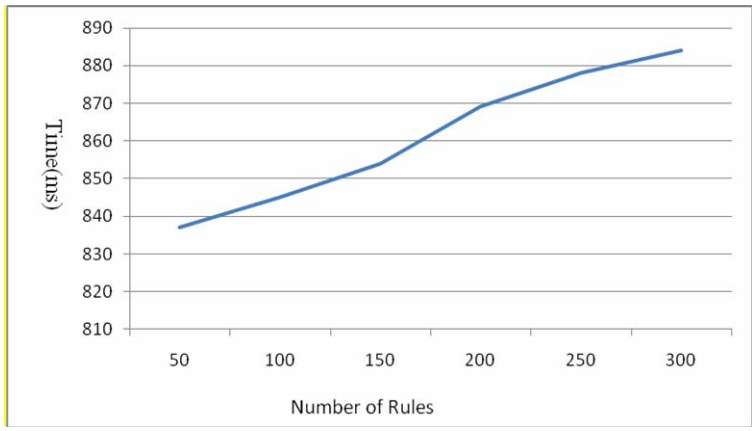

Figure 16: Latency of rule deducing in the RA

	Manual writing	**Our approach**
Coding/generating time	Several hours	3.8 s
Execute time	4.3 s	4.9 s

Table 2: Comparison between manual writing and our approach

Table 2 shows the comparison between manual writing and our approach. Although the execution time is slower than manual writing one, the coding time is superior to the manual writing. I think that is reasonable because the generated code needs more redundant statements than manual writing approach. In addition, as mentioned above, several previous approaches, each with its own characteristics, have

focused on automatic or semi-automatic code/agent generation. We summarize those of [Zhang et al. 07, Lage et al. 04, Chang et al. 02] and our own approach in Table 3. As can be seen there, we use ontology/reasoning rule to deduce the behavior of agent and full automatically merge all codes into a mobile agent without user/developer intervention. This constitutes the major merit of our design.

	Zhang et al. [Zhang et al. 07**]**	**Lage et al. [**Lage et al. 04**]**	**Chang et al. [**Chang et al. 02**]**	**Our approach**
Need Developer assistant	Yes	Yes	Yes	No
Reasoning approach	Quantitative model	Web page's Pattern recognition	Web page's Pattern recognition	Ontology/ reasoning rule
automation	Semi-	Fully	Semi-	Fully

Table 3: Comparison among some famous approaches

6.2 Discussion

The section discusses the advantages and disadvantages of using the methodology to generate agent for information retrieval on cloud environment. First, we examine the advantages, as follows:

- Security: The issue is the most important on the Internet-based computing. The approach is easy to prevent the malicious processes to retrieve information from data center of cloud node. Although the data centers are shared among all authorized users, some malicious code segments might be incidentally embedded into the retrieving program. Using the approach, all mobile agents are combined and generated dynamically, there may be secure than the manually written programs.
- Context-awareness: The approach can automatically generate the mobile agent without any user's intervention. User does not need to be familiar with the schema and structure of data center while he would like to retrieve information from the data centers. Obviously, it is a context-awareness approach and fit the characteristic of cloud computing.
- Scalability: The framework is based on a famous agent platform. We can extend the framework and add other agents easily to the framework for future, only with a little bit modification.
- Effectiveness: A simple example has demonstrated to the approach can apply a simple information retrieval. Obviously, the approach can be applied to other applications by extending new ontologies and rules in the Domain-Knowledge Repository, and adding new algorithms into the Retrieving Algorithm Repository.
- Transparency: Utilizing the approach, user is fully transparent to the structure and location of data centers of cloud nodes.

- Efficiency: Based on the latency measurement, the approach is shown that is an efficient approach to generate a mobile agent for information retrieval on the internet-based environment.

Next, we explain the possible drawbacks if the approach is exploited, as follows:

- Flexibility: When the schema of information source changed, all code segments in the Retrieving Algorithm Repository also need to updated.
- Fault tolerant issue: Now the platform and the framework do not consider the fault occurring. If the platform crashed, the system will be forced to stop. If a cloud node crashed, a mobile agent cannot run properly. In these cases, the system is not able to execute continuously.

7 Conclusion and Future work

In this paper, we have proposed and designed an ontology based Agent Generation for Information Retrieval on cloud environment. The framework can assist users to automatically generate mobile agents for retrieving desired information spread over cloud platforms without user's intervention. A user's request will be automatically deduced by a Reasoning Agent (RA) based on predefined ontology and reasoning rule, and then be translated to a Mobile Information Retrieving Agent Description File (MIRADF) that is formatted in a proposed Mobile Agent Description Language (MADF) that defines the behavior of mobile IR agent. A generating agent, named MIRA-GA, is also implemented to generate a MIRA according to the MIRADF. We also design and implement a prototype to integrate these agents and show an interesting example to demonstrate the feasibility of the architecture.

The system is only a prototype for retrieving petty information. But we believe that the methodology can be applied to other applications. The future work we first, would like to extend he framework to other applications. In addition, we will take the drawbacks mentioned in Section 6.2 into account to strengthen the robust of system.

Acknowledgements

We are grateful for the many excellent comments and suggestions made by the anonymous referees. This work was supported by the Nation Science Council of Republic of China under Grant No. 98-2221-E-305-006, 98-2218-E-029-004 and No. 99-2628-E-305-001.

References

[Ahmed 05] Ahmed T. M.: Generating Mobile Agent Securely by Using MASL, In Proc.: 25th IEEE International Conference Distributed Computing Systems, 6-10 June 2005, pp. 291-296.

[Armbrust et al. 09] Armbrust M., Fox A., Griffith R., Joseph A. D., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., Stoica I., Zaharia M.: Above the Clouds: A Berkeley View of Cloud Computing, is available at http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[Aversa et al. 09] Aversa R., Martino B. D., Venticinque S.: Integration of Mobile Agents Technology and Globus for Assisted Design and Automated Development of Grid Services, 2009 International Conference on Computational Science and Engineering, CSE '09, pp 118 – 125.

[Bhatt et al. 04a] Bhatt M., Flahive A., Wouters C., Rahayu J. W., Taniar D. Dillon T.S.: A Distributed Approach to Sub-Ontology Extraction. AINA (1) 2004: 636-641.

[Bhatt et al. 04b] Bhatt M., Wouters C., Flahive A., Rahayu J. W., Taniar D.: Semantic Completeness in Sub-ontology Extraction Using Distributed Methods. ICCSA (3) 2004: 508-517

[Bhatt et al. 06] Bhatt M., Flahive A., Wouters C., Rahayu J. W., Taniar D.: MOVE: A Distributed Framework for Materialized Ontology View Extraction. Algorithmica 45(3) 2006: 457-481.

[Camponogara and Shima 10] Camponogara E., Shima R. B.: Mobile Agent Routing with Time Constraints: A Resource Constrained Longest-Path Approach, Journal of Universal Computer Science 16(3) 2010: 372-401.

[Cannataro et al. 04] Cannataro M., Congiusta A., Pugliese A., Talia D., Trunfio P.: Distributed data mining on grids: Services, tools, and applications, IEEE Transactions on Systems, Man, and Cybernetics 34(6) 2004: 2451-2465.

[Cannataro et al. 02] Cannataro M., Talia D., Paolo T.: Distributed data mining on the grid, Future Generation Computer Systems 18(8) 2002 : 1101-1112.

[Cannataro and Talia 03] Cannataro M., Talia D.: The knowledge grid, Communication of ACM 46(1) 2003:89-93.

[Cerri 08] Cerri D.: Towards Knowledge in the Cloud, Lecture Notes in Computer Science, Vol. 5333, 2008: 986-995.

[Chang et al. 02] Chang Y.-S., Ho M.-H, Sun W.-C., Yuan S.-M.: Supporting Unified Interface for Wrapper Generator in Integrated Information Retrieval, Computer Standards and Interfaces, 24(4) 2002: 291-308.

[Chang et al. 07] Chang Y.-S., Juang T.-Y., Cheng C.-M.: Efficient Computational Grid Middleware for Mobile Device, Journal of Information Technology and Applications 1(4) 2007: 275~283.

[Chang et al. 08] Chang Y.-S., Luo Y.-C., Shih P.-C.: AIR: Agent and ontology -based Information Retrieval architecture for Mobile Grid, In Proc.: 2008 IEEE Asia-Pacific Services Computing Conference, Yilan Taiwan, December 9-12, 2008, pp. 650-655.

[Chow et al. 09] Chow T.Y., Tsai W.-T., Balasooriya J., Bai X.: Ontology-based Information Sharing in Service-Oriented Database Systems, In Proc.: 2009 IEEE International Conference on Services Computing, 2009, pp.276-283.

[Delic and Riley 09] Delic K. A., Riley J. A.: Enterprise Knowledge Clouds: Next Generation KM Systems, In Proc.: International Conference on Information, Process, and Knowledge Management, 2009, pp. 49 – 53.

[Flahive et al. 06] Flahive A., Apduhan B. O., Rahayu J. W., Taniar D.: Large scale ontology tailoring and simulation in the Semantic Grid Environment. International Journal of Metadata, Semantics and Ontologies 1(4) 2006: 265-281.

[Flahive et al. 09] Flahive A., Taniar D., Rahayu J. W., Apduhan B. O.: Ontology tailoring in the Semantic Grid. Computer Standards & Interfaces 31(5) 2009: 870-885.

[Geelan 08] Geelan J.: Twenty one experts define cloud computing, Virtualization, August 2008. Electronic Magazine, article available at http://cloudcomputing.sys-con.com/node/612375?page=0,1

[Genco et al. 06] Genco A., Sorce S., Reina G., Santoro G.: An Agent-Based Service Network for Personal Mobile Devices, IEEE Pervasive Computing 5(2) 2006: 54-61.

[Glitho et al. 02] Glitho, R.H., Olougouna, E., Pierre, S.: Mobile agents and their use for information retrieval: a brief overview and an elaborate case study, IEEE Network 16(1) 2002: 34 – 41.

[Golovchinsky et al. 09] Golovchinsky G., Qvarfordt P., Pickens J.: Collaborative Information Seeking, IEEE Computer 42(3) 2009: 47-51.

[Hand 07] Hand E.: Head in the clouds. Nature 449, 2007:963.

[Jaeger et al. 08] Jaeger P.T., Lin J., Grimes J. M.: Cloud Computing and Information Policy: Computing in a Policy Cloud? Journal of Information Technology & Politics 5(3) 2008: 269 – 283.

[Koehler et al. 10] Koehler M., Ruckenbauer M., Janciak I., Benkner S., Lischka H., Gansterer W. N.: A grid services cloud for molecular modelling workflows, International Journal of Web and Grid Services 6(2) 2010: 176-195.

[Krishnaswamy et al. 04] Krishnaswamy S., Loke S. W., Zaslasvky A.: A hybrid model for improving response time in distributed data mining, IEEE Transactions on Systems, Man and Cybernetics Part B 34(6) 2004: 2466 -2479.

[Lage et al. 04] Lage J. P, Silva A. S. d. , Golgher P. B., Laender A. H.F.: Automatic generation of agents for collecting hidden Web pages for data extraction, Data & Knowledge Engineering 49, 2004: 177–196.

[Lanzenberger et al. 10] Lanzenberger M., Sampson J., Rester M.: Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data, Journal of Universal Computer Science 16(7) 2010: 1036-1054.

[Li et al. 10] Li J., Wang Q., Wang C., Cao N., Ren K., Lou W.: Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing, access in 2010, http://bx.businessweek.com/cloud-computing-/enabling-efficient-fuzzy-keyword-search-over-encrypted-data-in-cloud/1743390911738279326O-9a74d50f93c8576792e104d2ca365192/

[Lin and Chen 07] Lin, Y.-F., Chen, J.-Y.: OWL-Based Description for Agent Interaction, In Proc. 31st Annual International Computer Software and Applications Conf., (COMPSAC 2007) Vol. 2 : 147-152.

[Luo et al. 07] Luo J, Wang M, Hu J, Shi Z.: Distributed data mining on Agent Grid: Issues, platform and development toolkit, Future Generation Computer Systems 23 (1) 2007: 61-68.

[Luo and Ni 09] Luo J., Ni X.: A clustering analysis and agent-based trust model in a grid environment supporting virtual organisations. International Journal of Web and Grid Services 5(1) 2009: 3-16.

[Manvi et al. 09] Manvi S. S., Kakkasageri M. S., Pitt J.: Multiagent based information dissemination in vehicular ad hoc networks. Mobile Information Systems 5(4) 2009: 363-389.

[Martino 09] Martino B. D.: Semantic web services discovery based on structural ontology matching. International Journal of Web and Grid Services 5(1) 2009: 46-65.

[McFedries 08] McFedries P.: The cloud is the computer. IEEE Spectrum Online, August 2008. Electronic Magazine, available at http://www.spectrum.ieee.org/aug08/6490.

[Qu et al. 08] Qu W., Li K., Zhang C.: Efficient Information Retrieval by Dispatching Mobile Agents in Parallel, International Conference on Multimedia and Ubiquitous Engineering, MUE 2008, 24-26 April 2008, pp. 73 – 76.

[Shih et al. 09] Shih W.-C., Yang C.-T., and Tseng S.-S.: Ontology-based content organization and retrieval for SCORM-compliant teaching materials in data grids, Future Generation Computing Systems 25(6) 2009: 687-694.

[Shih et al. 10] Shih W.-C., Yang C.-T., and Tseng S.-S.: Performance-based Data Distribution for Data Mining Applications on Grid Computing Environments, Journal of Supercomputing 52(2) 2010:. 171-198.

[Stankovski et al. 08] Stankovski V et al.: Grid-enabling data mining applications with DataMiningGrid: An architectural perspective, Future Generation Computer Systems 24 (4) 2008: 259-279.

[Vaquero et al. 09] Vaquero L. M., Rodero-Merino L., Caceres J., Lindner M..: A Break in the Clouds: Towards a Cloud Definition, SIGCOMM Comput. Commun. Rev. 39(1) 2009: 50-55.

[Vecchiola et al. 09] Vecchiola, C., Chu X., Buyya R., Aneka: A Software Platform for .NET-based Cloud Computing, available at http://www.gridbus.org/reports/AnekaCloudPlatform2009.pdf

[Vouk 08] Vouk M. A.: Cloud Computing – Issues, Research and Implementations, In Proc. 30th International Conference on Information Technology Interfaces, June 2008, pp. 235–246.

[Waluyo et al. 05] Waluyo A. B., Srinivasan B., Taniar D.: Research on location-dependent queries in mobile databases. Comput. Syst. Sci. Eng. 20(2) 2005: 79-95

[Wang et al. 06a] Wang Y., Behera S. R., Wong J., Helmer G., Honavar V., Miller L., Lutz R., Slagell M.: Towards the automatic generation of mobile agents for distributed intrusion detection system, Journal of Systems and Software 79 (2006) 1–14.

[Wang et al. 06b] Wang G., Wen T., Guo Q., Ma X.: A Knowledge Grid Architecture Based on Mobile Agent, In Proc.: Second International Conference on Semantics, Knowledge, and Grid, 2006, pp. 48 – 51.

[Wöhrera et al. 05] Wöhrera A., Brezanya P., Tjoa A. M.: Novel mediator architectures for Grid information systems, Future Generation Computer Systems 21(1) 2005: 107-114

[ZadJabbari et al. 10] ZadJabbari B., Wongthongtham P., Hussain F.K.: Ontology based Approach in Knowledge Sharing Measurement, Journal of Universal Computer Science 16(6) 2010: 956-982.

[Zhang et al. 07] Zhang X., Xu H., Shrestha B.: Developing Multi-Agent Systems with Automatic Agent Generation and Dynamic Task Allocation Mechanisms, The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems, May 14–18 2007, Honolulu, Hawaii, USA, pp. 1254-1256.