THOMSON REUTERS
ENDNOTE

[Gnanamani., 3(4): April, 2016] ISSN 2349-4506
Impact Factor: 2.265

Global Journal of Engineering Science and Research Management

CONFIGURATION FILE RECOMMENDATIONS USING METADATA AND FUZZY TECHNIQUE

Gnanamani.H*, Mr. C.V. Shanmuka Swamy
*PG Student Department of Computer Science Shridevi Institute Of Engineering and Technology Tumakuru, Karnataka, India
Associate Professor Department of Computer Science Shridevi Institute Of Engineering and Technology Tumakuru, Karnataka, India

KEYWORDS: cloud Migration; configuration file discovery; file metadata; fuzzy string matching.

ABSTRACT
In the real world installation of the software's or upload software's to the cloud requires a lot of understanding of the configuration files through which software's understands how to run , where to run and what to run. Basically we call configuration files are the heart of the software. Discovery of configuration files is one of the prerequisite activities for a successful workload migration to the cloud. The complicated and super-sized file systems, the considerable variance of configuration files, and the multiple presences of configuration items make configuration file discovery very difficult. The Traditional approaches usually highly rely on experts to compose software specific scripts or rules to discover configuration files, which is very expensive and labor-intensive. Our proposed approach is a novel learning based approach named MetaConf to convert configuration file discovery to a supervised file classification task using the file metadata as learning features such that it can be conducted automatically, efficiently, and independently of domain expertise. I report our evaluation with extensive and real-world case studies, and the experimental results validate that our approach is effective and it outperforms our baseline method.

INTRODUCTION
Cloud computing is dramatically changing the modern enterprise IT. More and more workloads have been, are being, and are going to be moved to the cloud. According to Cisco Global Cloud Index, global cloud traffic will account for nearly two-thirds of total data center traffic by 2016[1]. In 2012, International Data Corporation (IDC) reported that more than 50% of larger European companies stated that they have a strategy for refactoring IT infrastructure in order to benefit from cloud economics, of which the cloud migration investment is forecast to grow at 30.6% compound annual growth rate (CAGR) in the next 5 years[2] . For a success of workload migration, discovering the configuration files of existing enterprise workloads is the foundation [3].

The difficulty of configuration files is due to the following issues:
1) The huge amount of configuration files: Enterprise applications are usually deployed in a distributed and multiple-layer software environment. Each software usually requires a large number of configuration files.

2) The variance of configuration files: The locations of configuration files usually vary between different deployments environments.

3) The multiple presence of configurations: A configuration is usually defined in one configuration file but referred to by multiple other files including other configuration files, temporary files, and log files. This will cause simple configuration keyword search to fail if used for configuration file discovery.

Existing configuration file discovery approaches can be approximately categorized into three types:
1) Script-based approaches, which run pre-developed domain specific scripts to discover configuration files on the target server.

2) Rule-based approaches, which apply pre-defined rules written by application experts to annotate configuration files on the target server.

3) Human interactive approaches, which engage end users to interact with the system to define patterns to match configuration files or specify them directly. All of these approaches require rich domain expertise and high

familiarity with the software and system configurations. The main advantage of these approaches is that they are capable of achieving relatively high accuracy of identifying configuration files once the expertise requirement is met. The disadvantage is also obvious: the involvement of human expertise and labor makes these approaches very expensive and not scalable.

REALTED WORK

Discovering configuration files is the foundation of cloud migration. It attracts much research attention as well as commercial vendors to solve this problem. Existing approaches can be mainly categorized into three types: script-based approaches, rule-based approaches, and human interactive approaches. The script-based approaches are the most popular techniques that dominate the configuration file discovery market. Many commercial enterprise application configuration file discovery techniques such as IBM Tivoli® Application Dependency Discovery Manager (TADDM) [6], HP Discovery and Dependency Mapping (DDM) [5], EMC Application Discovery Manager (ADM) [9], and Evolven [4] utilize the product specific scripts or sensors by sending them to the running systems to capture application configuration files and configurations.

The academic efforts also apply these script-based approaches in discovering configuration files and configurations. Galapagos [7][8] used a model-based approach and product specific scripts to discover the configuration files and configurations from data centers. These approaches highly depend on the embedded domain specific discovery techniques developed by the domain experts. It limits the scalability of the products or tools based on these approaches. The rule-based approaches apply the rules to annotate configuration files and configurations from the target servers. The human interactive approaches allow end users to interact with the system to define patterns to match configuration files and configurations or specify them directly.

All of these approaches require rich domain expertise and high familiarity with the software and system configuration, while MetaConf can discover these configuration files with much less human interaction based on the nature of self-similarity in the metadata of configuration files. MetaConf solves the configuration file discovery problem by converting it to a learning based file classification problem.

PROPOSED SYSTEM

The main steps included in workflow of MetaConf are

(1) Token Base Construction: The file path and the file extension are two string features that should be transformed into numerical values prior to the training process. Our idea is to match a given file instance to a pre-built knowledge base that represents the target file class, and quantify this matching as the learning features. Therefore, as a first step we constructed two knowledge bases called Path Token Base and Extension Token Base respectively by capturing the representative tokens from the positive instances of a file class which contains a certain configuration item.

(2) Feature Extraction: I selected nine items from the file metadata and converted them into the learning features. They were file path, file extension, size, user access, group access, other access, change time, access time, and modify time. For the file path, we designed a fuzzy matching algorithm to compare a path to the Path Token Base, and calculate the similarity score as the feature. For the file extension, we searched the Extension Token Base for a match of an extension, and used the matched token's numerical weight as the feature. For the three time based items, I replaced them with *diff* (change time, modify time), *diff*(change time, access time), and *diff*(modify time, access time), to remove the environment dependency.

(3) Data Imbalance Handling: A system component to balance the class distribution in the training data to improve the classification performance. I implemented three different techniques: oversampling the minority (positive) class, under sampling the majority (negative) class, and assigning unequal error penalties for the two classes.

(4) Classifier Training: With balanced datasets and extracted features, We trained a Support Vector Machine (SVM) [6] based configuration file classifier with cross-validation for each file class.

ISSN 2349-4506
Impact Factor: 2.265

Global Journal of Engineering Science and Research Management

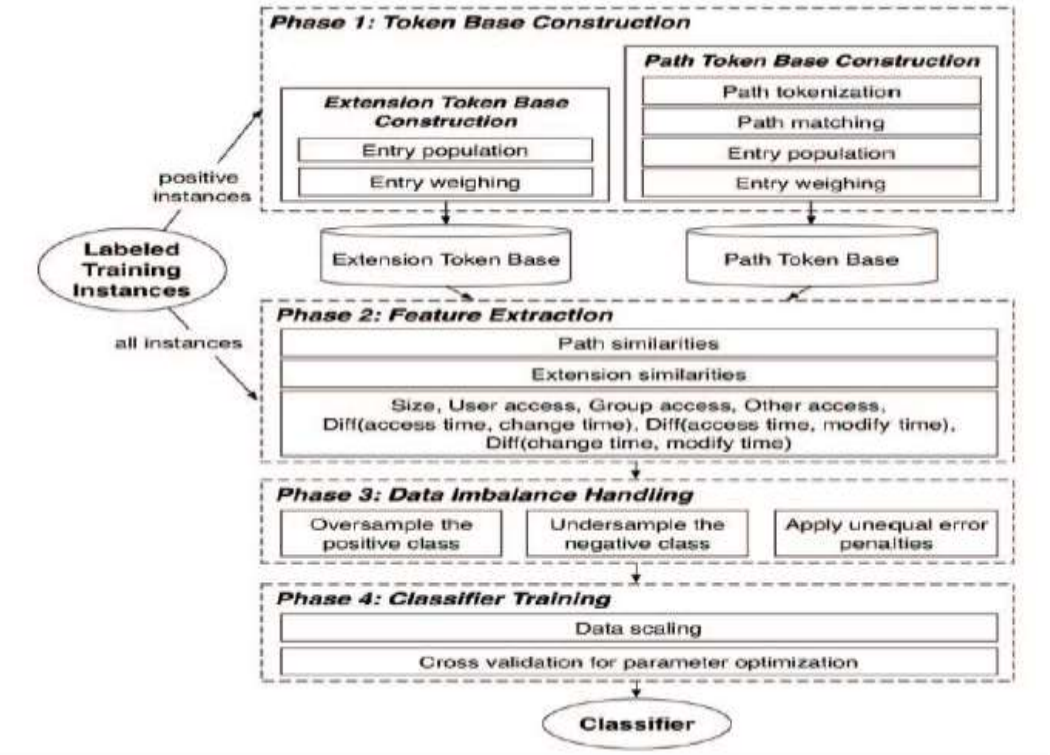

Fig. 1: Workflow of MetaConf.

SYSTEM DESIGN

I discuss the key components of our system including the representing file paths, file extensions and handling of data imbalance.

Representation of File Paths:

The file path carries a great amount of information about a configuration file in file metadata. In order to extract the repetitive patterns out of the file paths, we construct a Path Token Base for each class of configuration files. The path token represents a part of the most frequent locations of this file class and is assigned with a numerical weight that measures its significance based on its frequency, length, and position. A new file can then be compared with this Path Token Base by quantifying the matching between the path of the new file and the Path Token Base. If the matching score is high enough, i will conclude that this new file is similar to the file class represented by this Path Token Base in the path dimension.

Path token is a class of configuration files tends to have a number of common string patterns in their paths. The first goal is to tokenize the paths and obtain tokens, the smallest units constituting these patterns.

Having tokenized paths of a class of configuration files, we then deposit the most common path tokens into the Path Token Base. The commonality is measured by the similarity between the path tokens. If a set of tokens has higher pair-wise similarity than others do, these tokens should be playing a more important part in representing the locations of this class of files. When comparing two paths and their tokens, I adopt a fuzzy matching method.

With the fuzzy matching method, we are able to select abundant representative path tokens to fill in the Path Token Base. However, a path pattern usually not only includes tokens but also sequences of tokens. Moreover, the order of the tokens in the sequences is often strongly indicative of the locations of configuration files.

Using this Path Token Base, it is possible to match the path of a new file to it and quantify the matching result as a measurement of how similar the new file is to the file class represented by this Path Token Base in terms of the file path.

The matching process is as follows: the new file's path will first be tokenized into a set of tokens. Then a fuzzy matching algorithm shown in Fig. 2 is used to explore every possible token sequence combination of this path in order to find the best match in the Path Token Base that yields the highest similarity score. This similarity score serves to represent this file in the path dimension and is used for classification later.

Algorithm: findPathTokenSequenceMatching(p*tb*, *T*, *c*)
Input : *ptb*: a Path Token Base
T: an ordered path token set
c: a matching result cache (a hash map)
Output *r*: the matching result of *T* in *ptb*
1 *n*,*l* = length(T)
2 while *l* > 0
3 for *i* = 0 to *n*–(*l*–1)
4 *ltl* = *i*
5 *rtl* = *n*-(*i*+l)
6 divide *T* into three ordered token sets:
7 *lr* = findPathTokenSequenceMatching(*ptb*, *lT*, *c*)
8 *rr* = findPathTokenSequenceMatching(*ptb*, *rT*, *c*)
9 if *mT* in *c*
10 *mr* = *mT*'s value in *c*
11 else
12 for every path token sequence *seq* in *ptb*
13 *s* = computePathTokenSimilarity(*seq*, *mT*)
14 *s* = *s* * *ptb*.getWeight(*seq*)
15 *mr* = maximum value of *s*
16 put *mr* and corresponding *seq* into *c*
17 *ss* = *lr* + *rr* + *mr*
18 *r* = maximum value of *ss*
19 *l*--
20 return *r*

Fig. 2: The fuzzy matching algorithm for matching a path to a Path Token Base.

Representation of File Extensions
Extension Token Base to represent the information carried in the file extensions for a class of configuration files. The Extension Token Base is comprised of file extensions as the entries, with the frequencies of appearance as entries' weights. When comparing an extension to the Extension Token Base, if a match is found, the weight of the matched entry will be returned as the similarity score to represent the file in the extension dimension, otherwise the matching result is zero.

CONCLUSION

This paper proposed a supervised learning based approach, MetaConf, to automatically perform configuration file discovery, which has been currently dominated by manual or semi-automatic methods that heavily depend on human expertise and labor. We designed a fuzzy token matching based feature extraction schema to numerically represent the paths and the extensions of configuration files, and used them together with other file metadata as the learning features.

ISSN 2349-4506
Impact Factor: 2.265

REFERENCES

1. Cisco Analysis, "Cisco global cloud index: forecast and methodology, 2012-2017," White Paper, http://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/global-cloud-index gci/Cloud_Index_White_Paper.html.
2. M. Ahorlu, "European Cloud Professional Services, Cloud Management Services, and Hosted Private Cloud 2012–2016 Forecast", IDC Market Analysis , Nov 2012.
3. F.J. Meng, X.J. Zhuo, B. Yang, J.M. Xu, P. Jin, A. Apte, J. Wigglesworth. "A Generic Framework for Application Configuration Discovery with Pluggable Knowledge", Proceedings of Intl. Conf. on Cloud Computing, Jul. 2013.
4. Evolven, http://www.evolven.com/, 2014.
5. Hewlett-Pakard Development Company, HP Discovery and Dependency Mapping Software, http://www.hp.com/hpinfo/newsroom/press_kits/2007/softwareuniverseb arcelona/ds_inventory.pdf.
6. IBM Corporation, Tivoli Application Dependency Discovery Manager, http://www-306.ibm.com/software/tivoli/products/taddm.
7. K. Magoutis, M. V. Devarakonda, and K. M . Reddy, "Galapagos: automatically discovering application-data relationships in networked systems", Proceedings of the 10th IFIP/IEEE Internatinal Symposium on Integrated Network Management, 2007.
8. K. Magoutis, M. V. Devarakonda, N. Joukov, and N. Vogl, "Galapagos: model-driven discovery of end-to-end application-storage relationships in distributed systems", IBM J. Research and Development, 52(4-5), pp. 367-378, 2008.
9. M. Bowker, B. Garrett, and B. Laliberte, "EMC smarts application discovery manager", ESG Lab Validation Report, Technical report, 2007.