Management-based License Discovery for the Cloud

Minkyong Kim, Han Chen, Jonathan Munson, Hui Lei

IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532
{minkyong, chenhan, jpmunson, hlei}@us.ibm.com

Abstract. Enterprise software is typically licensed through contracts that require organizations to monitor their own usage of the software and purchase the number or amount of licenses required by the vendor's terms and conditions for that software. Vendors reserve the right to audit an organization's use of their software, and if an organization is under-licensed, costly back-payments may be required. For this reason, organizations go to great expense to maintain a complete and accurate inventory of their software so that they know their license obligations. The cloud, as an environment offering both greater flexibility in, and a higher degree of control over, an enterprise's computing infrastructure, presents both new challenges for license compliance as well as new opportunities. In this paper, we introduce a new approach to producing accurate software inventories based on capturing the knowledge that is present in cloud management systems at the time of software provisioning and installation. We also demonstrate new capabilities for rule-based alerting and enforcement that are made possible by our approach.

1 Introduction

Enterprise software is typically licensed through contracts that require organizations to monitor their own usage of the software and purchase the number of licenses required by the vendor's terms and conditions for that software. Vendors reserve the right to audit an organization's use of their software, and if an organization is under-licensed, costly back-payments may be required, in addition to civil fines and legal fees. Recently, one software vendor demanded $9 million in back payments [6]. In addition to the vendors themselves, organizations such as the Business Software Alliance police this space, with many lawsuits resulting in $11 and $13 million settlements [10].

To protect themselves from such incidents, organizations may employ a variety of tools, including license management tools that enable them to keep track of the licenses they have purchased, as well as scanning tools that scan the computers on their network for software and generate reports describing what software is installed. These reports are then used to determine what licenses are required. This technology has inherent limitations, which we describe below, and as a result, much manual effort is required to ensure the organization is compliant.

The cloud, as an environment offering both greater flexibility in, and a higher degree of control over, an enterprise's computing infrastructure, presents both new challenges for license compliance as well as new opportunities. The level of automation that clouds provide enables application deployers to provision and deprovision virtual machines (VMs) quickly, and stop them and start them even more quickly. These dynamics are

not well-matched with current scanning technology, which relies on machines being idle and available on a regular schedule.

However, the same automated management of infrastructure that presents challenges also provides an opportunity to dramatically change the way software discovery is performed. Software instances are created through machine provisioning and software installation, which, in a modern cloud environment, can be performed under automated control. During such operations, the information needed to determine required licensing—precise software identifiers, characteristics of the VMs, on which the software will run, even the purposes to which the software is being put—are knowable.

In this paper, we describe a new approach to license management for cloud environments, based on our experience with both public clouds and enterprise license management. We present a Management-based License Discovery (MLD) system that taps into a cloud's infrastructure management processes to automatically discover license usage and support the license compliance process. We also demonstrate new capabilities for rule-based alerting and enforcement that are made possible by our approach. We present the architecture of MLD and its interaction with other components in the cloud. We describe the implementation of the prototype of the system briefly. We will then discuss the user interface (at the web font-end) in detail; this provides visualization of our system's capabilities.

2 Background

The problem of determining what an enterprise's license obligations are is surprisingly complex. A large enterprise may have hundreds of thousands of software instances to account for, and the license required for any given instance may be determined by many factors, including not only the characteristics of the machine on which the software is installed, but also what other software it may be a component of, what contract this particular instance was purchased under, and what purpose this instance is used for (e.g., production or test).

The first part of this license compliance process is to generate a complete and accurate software inventory, annotated with the various factors above. This is called *software discovery*. In conventional infrastructures this is normally performed by agent-based computer scanning tools that are guided by a catalog of software "signatures"—fingerprints, as it were, that indicate if a particular software is installed. Signatures may be based on what files are present, or may be based on what processes are running.

The discovery process reports the precise software version installed as well as the characteristics of the machine it is installed on (number of cores, clock speed, RAM, etc.). It may also attempt to make certain judgments that are designed to avoid overcounting. One of these is that, when multiple instances of a single software "family" (e.g. Adobe Acrobat; Reader and Pro) are identified, only one will be reported. The definition of a family is determined by the software catalog guiding the discovery.

The discovery technology may also attempt to determine (if the discovery catalog is sufficiently rich) whether a software instance is part of a larger software installation. Many IBM products contain within them middleware products such as an application server and a database server. Normally, these products are licensed separately, but when

bundled as components, they do not require separate licenses. Because bundle relationships cannot be known simply based on knowledge of disk or process-table contents, scanning tools will only suggest them.

In the last phase of the discovery step, the software inventory is checked and additional information is added to it (e.g., whether an instance is used for production or test). This phase is typically labor-intensive and depends on knowledge and data held by the IT staff responsible for software asset management. It typically consists of: identifying any software instances that were reported but for which there is no corresponding machine (software orphans); identifying machine instances for which there is no reported software (machine orphans); removing software instances that are the result of known identification problems; confirming probable bundles identified by the scanning tools; identifying known bundles that the scanning tools cannot detect; applying the results of software-specific tools or scripts that identify instances that were missed or correct identifications that were incomplete.

3 Motivation

Scanning-based discovery has several drawbacks, especially when it is used in a self-service, rapid provisioning Infrastructure-as-a-Service (IaaS) cloud environment. One is that it puts a high burden on the host computer. The time it takes to finish the scanning process is directly proportional to the storage volume size, unless we make the assumption about standard software installation locations, but this assumption increases the risk of missed discovery. Although CPU resource consumption may be limited in modern multicore computers, the impact on disk I/O bandwidth can be significant, which could lead to potentially adverse performance impact on the actual workload. This is especially true in a cloud computing environment, where disks are virtualized and multiple virtual machines share a limited number of physical disk spindles on the hypervisor.

Due to the high overhead mentioned above, the scanning can typically only be scheduled at a relatively low frequency, for example, once a week or even once a month. The rapid provisioning nature of cloud means that many workloads can be relatively short lived and thus may be missed entirely by the periodic scans.

Scanning technology is only as accurate as the discovery catalog is complete and up-to-date. New software products or versions can be missed if they are not first cataloged. Complex software products such as databases have many editions and optional features, and it is difficult for scanning technology to identify which editions are installed and which features are in use. These, however, may strongly affect licensing.

Once the list of software products has been discovered, the inventory needs to go through a manual process to be ready for the next step, *reconciliation*. License reconciliation is the process of matching each software instance that requires a license with a license owned by the enterprise, and ensuring that there are a sufficient number of licenses, or license units, owned. Although this manual step is necessitated in part by the accuracy problems inherent in scanning technology, it is also used to add licensing-relevant information beyond what scanning can provide, such as what contract software was purchased under; whether it is used in production, or test, or development; whether it is used as on a primary node or only in backup capacity; or whether it is used in a client or a server capacity.

4 Management-based License Discovery

As discussed in Section 2, in the last phase of software discovery, the enterprise applies, in a labor-intensive process, its knowledge of the context of the software in its inventory, such as bundle relationships, contracts, and the kind of workloads served by the software. In this section, we describe Management-based License Discovery (MLD) wherein this kind of knowledge is encoded in structures that are accessed in the management system workflows that are used to provision VM images and install software bundles. In this way, an accurate software inventory may be maintained at all times. MLD also maintains data on the licenses an enterprise owns, and is thus able to offer services such as warning when an under-licensing situation may exist. Because the system is integrated with the cloud management system, it has access to not only the image and bundle catalogs, telling it what software is included in each image and bundle, but also has the knowledge of the characteristics of each virtual machine, and so can generate the software inventory records completely and accurately using this information. It therefore avoids all the issues that scanning technology has in a cloud deployment.

4.1 Design

To tack the license usage, users need to provide the information on the licenses that they own. This information typically includes software name, software version, license types, license dates, etc. Users also need to specify whether the licenses can be applied at the general software level or at the specific version level.

Through the license management web front-end, users can specify a set of rules. There are pre-action rules that are applied before the service action is executed and the post-action rules that are applied after the action has been taken. One of the important pre-action rules is whether to *enforce* the license compliance or to *inform* the status of the license usage. If a user chooses the enforcement option, the user must own the required licenses in order for the action (such as provisioning or installation) to succeed.

The post-action rules include those related to alerts. Users can specify under what condition they want to receive an alert message. For example, a rule can be if the current usage reaches 90% or above of the licenses that a user owns, send an alert message to a specific e-mail account.

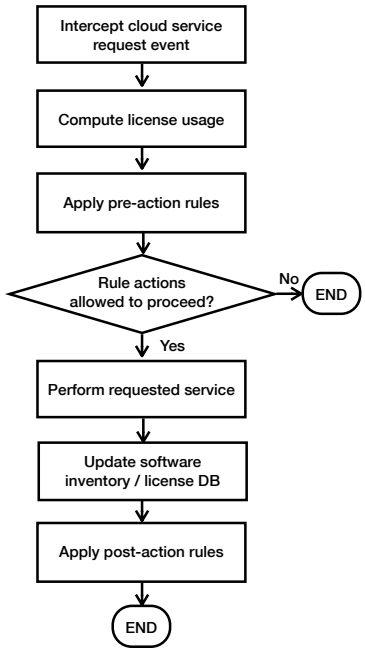


Fig. 1. MLD main flow

In addition to pre-action and post-action rules, there are other rules associated with the events that are not triggered by individual service request. For example, report generation is triggered periodically by a timer. Users can specify the period at which the reports should be generated and the level of details that reports should include.

Once the rules are defined, the license manager is ready to update license usage upon the arrival of cloud service requests. As the cloud service requests arrive through the web-based front-end, our license management engine taps into these events and computes the license usage. Our license manager can either enforce the license compliance or only detect the potential problems and generate warnings. In case of enforcement, the license manager will see whether the user owns appropriate licenses to proceed with the action. If the user indeed has the required license, the cloud management layer will execute the action. Once the action completes successfully, our license manager updates the software inventory and the license DB. The last step is to check whether there are post-action rules such as sending the alerts based on the current license usage. Figure 1 shows the main flow of the license manager as the individual cloud service request arrives at the system.

4.2 Persisted data

There are three main sets of data that need to be persisted for license management: information on available licenses, rules and license usage. In our prototype system, we use IBM DB2 to persist our data. Here are the three data sets: First, MLD keeps track of the licenses that users own. The information on each license includes the details on the software, effective dates, and license type. Second, MLD persists the rules that users enter through the web front-end. A rule is typically associated with an account (user) and has the start and end dates, during which it is affective. The rules include those describing the condition for alerts and the period for generating reports. Third, MLD persists license usage information by tracking the software instances that are running. Information for each software instance includes the start and end time of the software instance, ID of VM instance that contains this software instance and license type. Whenever a cloud service event changes the number of running software instances, MLD updates the number of required licenses and generates alerts based on the alert rules. Beside these three main sets of data, MLD also persists alert messages in database and generated reports in the file system, with the link to the file system location in database.

4.3 Interaction with the other components in Cloud

MLD assumes that certain information can be retrieved from the existing cloud components. When a provisioning request arrives at the cloud, it typically contains VM image ID, but does not contain the list of software components that come with the VM image. We assume that the cloud has the meta-data that describes the list of software components in VM images. Our current prototype has been tested for IBM SmartCloud Enterprise (SCE) [5] environment, in which a meta-data file (called *topology* file) in the XML format lists the software components in VM images.

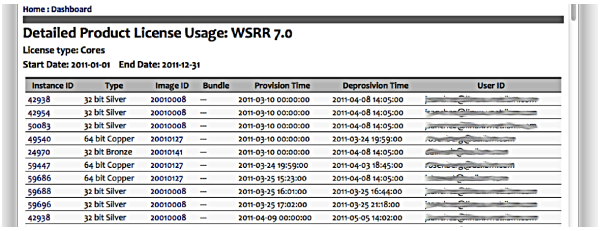
After a VM instance has been started, users can install additional software. We assume that users go through the management process to do so. When software components are installed in the form of software *bundle*, which consists of multiple software components, the list of software components again can be retrieved from the exiting cloud components. We assume that given the software bundle ID, MLD can easily retrieve this information.

5 Prototype

We have implemented a prototype system of MLD on IBM's SCE [5], an IaaS cloud offering. MLD receives the events from the cloud management layer that are relevant to software license management, including VM provisioning, VM deprovisioning, software bundle installation and uninstallation. In addition to license discovery, the system also implements additional value-added services, such as visualization, usage alerts and reports. The account administrator accesses the system via a Web-based user interface. Once logged in, the administrator has access to three main sections of the application: Dashboard, Limits and Reports.

Dashboard The dashboard page provides a high level overview of software license usage. After the administrator enters a query date range in the query section and clicks the "Go" button, the section below is refreshed to show an *overview table*. Each row represents a single software product, the type (e.g., PVU, Cores, MIPs) of license that is applied to it, the entitlement value and the peak usage value during the query range.

From the *overview table*, the administrator can click on a particular product and obtain a *detailed view* of all VM instances that are alive during the query range (Figure 2), which explains the license usage value that is presented in the *overview table*. At the bottom of this *detailed view* page, we also present a graphic visualization of the license consumption, with time as x-axis and licenses as y-axis (Figure 3). The administrator also has the ability to save the query result as a report, which will be described later.


Home : Dashboard

Detailed Product License Usage: WSRR 7.0

License type: Cores

Start Date: 2011-01-01 End Date: 2011-12-31

Instance ID	Type	Image ID	Bundle	Provision Time	Deprovision Time	User ID
42938	32 bit Silver	20010008	--	2011-03-10 00:00:00	2011-04-08 14:05:00	
42954	32 bit Silver	20010008	--	2011-03-10 00:00:00	2011-04-08 14:05:00	
50083	32 bit Silver	20010008	--	2011-03-10 00:00:00	2011-04-08 14:05:00	
49540	64 bit Copper	20010127	--	2011-03-10 00:00:00	2011-03-24 19:59:00	
24970	32 bit Bronze	20010141	--	2011-03-10 00:00:00	2011-04-08 14:05:00	
59447	64 bit Copper	20010127	--	2011-03-24 19:59:00	2011-04-03 18:45:00	
59686	64 bit Copper	20010127	--	2011-03-25 15:23:00	2011-04-08 14:05:00	
59688	32 bit Silver	20010008	--	2011-03-25 16:01:00	2011-03-25 16:44:00	
59696	32 bit Silver	20010008	--	2011-03-25 17:02:00	2011-03-25 21:18:00	
42938	32 bit Silver	20010008	--	2011-04-09 00:00:00	2011-05-05 14:02:00	

Fig. 2. Prototype: Per-software Detailed View

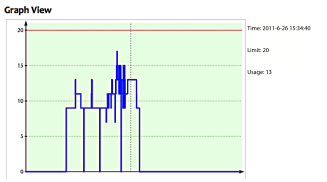


Fig. 3. License Usage

Limits The second main section of the application is the Limits and Alerts page. Each row in the table represents a single software product. By clicking on the hot links in the license type and entitlement columns, an administrator can configure the license type and entitlement value for the selected software product. The entitlement values are used by an alert rule engine that can generate notification e-mails based on current license consumption situation with respect to its configured limit.

By clicking on the "Edit alert rules" link from the page, the administrator can manage the alert rules for his account. An alert rule consists of condition (above, below, equal), threshold (expressed as a percentage of the entitlement value), message severity, notification e-mail address, and the set of software products to which it is applied. After an alert rule is applied to a software product, its license consumption value is monitored by the system. When the test condition is met, for example, usage is over 95% of the entitlement value, a notification e-mail is sent to the pre-configured e-mail address in the alert rule, so that the responsible stakeholder can take appropriate actions.

Reports The third main section of the application contains a list of all archived reports for the account. There are two ways to generate reports. First, an administrator can manually create a report based on the query result obtained from the dashboard page. Second, reports can be generated automatically according to rules. The administrator can manage the report generation rules in a separate page accessible from the main report page. A report generation rule specifies the frequency of the generation (daily, monthly, quarterly), start date, and time of date at which the report is to be created.

Users can choose the level of details that the report should include. A typical report contains the maximum number of software instances that were running for each software product during the period that the report includes and the number of licenses that the user owns for each product. It may also include alert messages that are generated during the corresponding reporting period.

Once a report is created, it is listed in the main report list. The administrator can download the report as an XML format for further downstream process or as a PDF format for human viewing. After reviewing the report content, the administrator has the option to sign the report with additional comments for audit and compliance purposes.

6 Related Work

There has been a limited amount of work on license management within the research community. To best of our knowledge, our work is the first to address how automation of license discovery can be achieved in a managed cloud environment.

One approach for automating license management is to develop a model to capture the business process and license requirements. Giblin *et al.* [3] developed a meta-model that can capture free-form passages describing the license requirements and regulations. Liu *et al.* [8] presented a modeling method for both business process and compliance rules. Once both are presented as models, process models can be verified against compliance rules by means of model-checking technology, which is automated. This thread of work is orthogonal to our work in management-based discovery but can complement it by providing information in a semantic-rich, machine-friendly manner. However, our approach does not require a model, as we implicitly collect semantically-rich information by tapping into the management process.

There have been some efforts on managing license compliance for free and open source software development. Gangadharan *et al.* [2] presents a way to implement clauses of open source software license in a machine interpretable way and describes a novel algorithm that analyzes compatibility between multiple free and open source licenses. Another example that shows developers' interest in license terms of open source software is Google's Advanced Search [4], which provides an option of filtering out search according to the different degrees of license obligations (e.g., "free to use or share," "free to use, share, or modify").

There are some less relevant efforts, but still consider licenses. LASS [1] presents a service selection problem with the license specification as one of the factors during the selection process. There have been other papers [7,9] on service selection, but these do not consider the license specification. While these papers focus on the service selection, the focus of our paper is license management.

7 Conclusion and Future Work

License management is an important problem in the enterprise world, as the costs of non-compliance are severely high. Avoiding these penalties has been a costly process for enterprises. The scanning-based approach has well known drawbacks, and the labor necessary to prepare software inventories for license reconciliation is extensive.

In this paper, we presented management-based license discovery (MLD), which takes advantage of a cloud's managed environment. MLD taps into a cloud's infrastructure management processes to automatically discover license usage and support the license compliance process. We also demonstrated new capabilities for rule-based alerting and enforcement that are made possible by our approach. We presented our initial prototype that has been intended for IBM's SCE. We believe that our automated system can significantly reduce the cost of license management.

While we believe that our prototype system demonstrates the value of our management-based discovery approach, we are aware that it does not fully address all the requirements an enterprise may have for preparing an inventory for license reconciliation. There is a need to capture the usage context of software, such as whether an instance is used in development or production. However, the precise context that matters may differ from enterprise to enterprise. We need to be able to allow the enterprise to indicate which context attributes should be captured when software is installed, and to extend the standard software inventory with those attributes. We are now extending our system in order to do that.

References

1. G. Gangadharan, M. Comerio, H.-L. Truong, V. D'Andrea, F. De Paoli, and S. Dustdar. Lass - license-aware service selection: Methodology and framework. In *Service-Oriented Computing - ICSOC*, pages 607–613, July 2008.
2. G. Gangadharan, V. D'Andrea, S. Paoli, and M. Weiss. Managing license compliance in free and open source software development. *Information Systems Frontiers*, 14:143–154, 2012.
3. C. Giblin, S. Muller, and B. Pfitzmann. From regulatory policies to event monitoring rules: Towards model driven compliance automation. Technical report.
4. Google. Advanced Search. See http://www.google.com/advanced_search?hl=en.
5. IBM. IBM Smart Cloud. See http://www.ibm.com/cloud-computing/us/en/.
6. Kanaracus, Chris. SAP, Rent-a-Center in Battle Over Millions in Fees. *CIO*, 2011. See http://www.cio.com/article/686832/SAP_Rent_a_Center_in_Battle_Over_Millions_in_Fees.
7. S. Lamparter, A. Ankolekar, and R. Studer. Preference-based selection of highly configurable web services. In *Proc. of the 16th Int. World Wide Web Conference*, pages 1013–1022. ACM Press, 2007.
8. Y. Liu, S. Muller, and K. Xu. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46(2):335 –361, 2007.
9. S. Reiff-marganiec, H. Q. Yu, and M. Tilly. Service selection based on non-functional properties. In *Proceedings of Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop*. Springer, 2007.
10. Rosenberg, Scott D. Software License Compliance: Myth vs. Reality. *E-Commerce Times*, 2008. See http://www.ecommercetimes.com/story/64465.html.