8-6-2011

An Agent based Proactive System Management in the Cloud

Nandini Taneja
American Water, Nandini.taneja@gmail.com

Aakash Taneja
Richard Stockton College of NJ, aakash.taneja@stockton.edu

An Agent based Proactive System Management in the Cloud

Nandini Taneja
American Water
Nandini.taneja@gmail.com

Aakash Taneja
Richard Stockton College of NJ
Aakash.taneja@stockton.edu

ABSTRACT

Cloud computing is the newest approach that has quickly evolved in last few years. It provides consumers with services such as software, storage, network or hardware from a location-independent cloud. These services are on-demand with a "pay only for service used" model that reduces overall capital costs. Although cloud computing has many benefits, a lack of confidence in the technology is restricting many organizations to move their data and software to the cloud. The concerns include security risks and lack of clear system level monitoring from the cloud providers that leads to a need of control and visibility of effective fault tolerance in the cloud. In this paper, we focus on agent based proactive management system where configurable agents are spawned in the cloud on-demand to monitor system and application resources, collect data to identify possible failures, construct a pattern knowledge base and take decisions for failure mitigation.

Keywords

Agents, cloud computing, proactive system management.

INTRODUCTION

A cloud is a part of the Internet running a bundle of services that can be accessed by users. Cloud computing refers to both the applications delivered as "services" over the Internet and the "hardware and systems software" in the data centers that provide those services (Armbrust Fox Griffith Joseph Katz Konwinski Lee Patterson Rabkin and Stoica 2010). The users do not know the location of the services such as data, storage and servers. There is no need for organizations to set up their own infrastructures when they use cloud services that can be accessed by users after proper identification/authentication over the network.

Even though cloud computing is quickly evolving and IT organizations are realizing its benefits, security analysts argue that cloud computing is still at an early phase and security is one of the major concerns for various organizations. Compliance in terms of logging, auditing, monitoring the system and service level agreements about services, responsibilities, guarantees are not well defined or designed by the cloud providers. In the last few years, there have been incidents of outages at cloud computing providers that cause a lack of confidence among users and organization to use cloud. Mark Williams, a cloud computing consultant's blog (Williams 2011) reported that Microsoft had four outages, Saleforce.com had two, Google had twelve outages and Amazon had five. CRN Technology News (CRN.Technology.News 2011) reported top 10 cloud outages. The issues range from connectivity loss because of failure in software, being unable to restore data from backup, a lack of solid disaster recovery plan, loss of access to the applications possibly because of system or network issues. Any major (or minor) outage causes a downtime and influences the normal functionality of the operations.

What distinguish a cloud from any other software application or system is that a cloud is normally a set of servers and software systems configured to work together -either centralized or distributed and whose location is transparent to users, thereby forming a mesh of servers on the web. Therefore, it is important to have a system so that each cloud provider can manage their servers as well as can work with other providers who manage their own servers to perform preventive or proactive management without any interface and software platform hassles. Standards and best-practices of hosting, maintaining and health monitoring are still evolving(Dikaiakos Katsaros Mehra Pallis and Vakali 2009) and cloud providers need to have robust system monitoring and software management architecture. In this paper, we discuss how agents can be used to provide proactive system management to achieve long-term management of cloud services. We call it Agent based proactive system management – AgPSM

This paper is organized as follow: The next section presents an overview of cloud, followed by a discussion of agent approach. Next we discuss our agent based proactive system management (AgPSM) approach and an example of AgPSM in the cloud. The final sections discuss agent communication within and outside the cloud, and the conclusion

CLOUD – AN OVERVIEW

A cloud is a collection of services offered over the internet by cloud providers at an on-demand basis (Figure 1). The end-user or consumer does not have to know the physical location as well as any configuration details of the systems and services they are using. The end-user only pays for the services used and cloud-computing claims to reduce the total cost since capital expenditure is converted to operational expenditure. The various benefits of cloud computing includes: i) A faster time to setup a new environment without hassles of buying new hardware, storage, operating systems and servers. ii) A faster time to market. iii) the possibility to easily switch between releases and projects. iv) the possibility to expand thereby providing scalability when load exceeds a threshold. v) the pay-as-you-go or on-demand model where a user only pays for service used and does not have to pay for time, storage, bandwidth and system resources if they are not being used. vi) the ability to access the cloud environment for virtual or mobile company people such that they do not have to always use the company network but can access the external network to access the cloud.

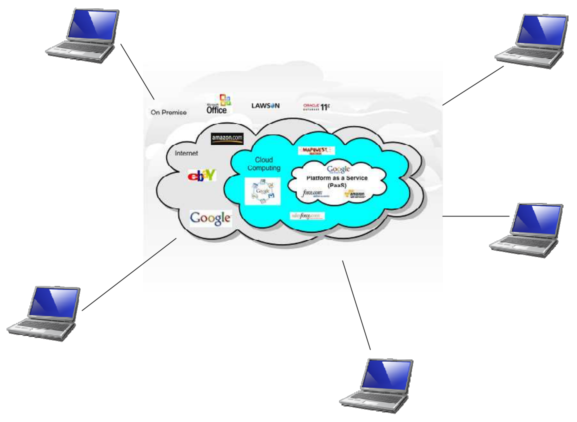


Figure 1. The Cloud (Adopted from cloudtimes.org)

There are mainly three levels of services provided by cloud computing: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In SaaS, an application is provided to users over the web and is hosted on the SaaS provider's data center. Some of the SaaS providers are Salesforce.com, Gmail, and Netsuite. In PaaS, the application's development platform is provided as a service and consists of application server software, database, middleware such as messaging queues and development tools for developers to develop their own apps. Some examples of PaaS providers are GoogleApp Engine, Engine Yard Salesforce.com's force.com. IaaS is the delivery of hardware as well as software as a service. So everything from server, storage, virtual server, network to operating systems, file system and disks can be offered to the users. Examples of IaaS providers are Amazon Web Service Elastic Compute Cloud (EC2), Microsoft's Azure, Akamai and Secure Storage Service (S3). The users can choose the services to fit their organization needs.

A cloud can be public, private or hybrid. IT organizations that have less concern in moving their data to cloud or choose only the non-sensitive data such as simulated data largely prefer public cloud. Public cloud is accessible over the internet and its location is transparent to the user. In contrast, a private cloud is set within the network of an organization in its own data center. A hybrid cloud is a mix of two options. Whatever be the model of cloud – public, private or hybrid, the proactive system management is still required and the monitoring, data collection and pattern making does not change.

AGENT APPROACH

An agent is a piece of software that can be programmed to run in an automated manner or invoked from another software/process(Nwana and Ndumu 1999). Agents can also be mobile agents(Lange and Oshima 1999) or expert systems knowledge-based agents(Chandrasekaran 1986). A mobile agent can roam around in the network, while a knowledge-based agent (a.k.a. Intelligent agent) uses "learn-and-act" model that comes with some preconfigured knowledge and algorithms and add up to its knowledge base to perform tasks based on what it learns. An agent needs a platform to run. For example: Java based agents need a java virtual machine (JVM) to run on, Python or Perl agents need a compatible operating system where they can run. Many agents can work cohesively and in a loosely couple manner to perform jobs elastically and with more flexibility. It is beyond the scope of this paper to discuss the technical architecture of agent's run time environment and their interaction with the operating system and with each other. In this paper we limit the scope to discuss how agents can be categorized based on their profiles and how they can be programmed to work cohesively within an AgPSM based system in the cloud.

Agents provide elasticity when there is a need for scalability or on-demand computing. Agents can be adaptive, knowledge based agents that can learn from user actions and profile. A large agent pool can be configured initially and a new agent can be spawned when demanded, and the agent returns back to its pool when the demand is no more present (Taneja and Taneja 2003) which is similar to cloud on-demand, scalable and pay-as-you-go service model. Introduction of agent approach on the cloud is relatively new. A mobile agent based service for cloud computing uses agents that can roam in the cloud between different platforms, therefore, rather than using RPC/RMI service, use mobile agent to implement the software and data service for cloud user in Internet environment, and make the cloud computing system adaptable to work in Internet environment(Chen Lu Huang and Wu 2010). Mobile agents can span across multiple platforms and provide portability and interoperability in the cloud where an application can migrate from one cloud provider to another (Zhang and Zhang 2009). However, mobile agents in the cloud between different cloud providers and application migration can cause potential security concerns to many organizations. Explicit security measures need to be taken to mitigate security risks because of mobile code(Dadhich and Dutta 2010). As each cloud service provider has its own API and implementation, it is almost impossible for them to allow mobile agents to move from one provider's network to another. We discuss the multi-agent based system where agents perform tasks within a cloud or outside cloud but without moving between platforms. The agents in our system are more focused to collect relevant data and construct patterns that form a knowledge base for the system to identify any failures, analyze and act on it at current time as well as future risks and issues can be proactively identified, thereby, making resources in the cloud more fault-tolerant.

OUR AGENT BASED PROACTIVE SYSTEM MANAGEMENT (AgPSM) APPROACH

An AgPSM based system involves actively monitoring system resources, application and software to identify failure, provide fast resolutions by following a sequence of events or activities (as programmed in the system) and construct a pattern to warn ahead of time of any potential outage, thereby managing effectively in the long-run. We designed AgPSM with an objective to effectively monitor the resources and software, collect data, and construct patterns. These patterns can be used for analysis and reporting, and agents' tasks can be configured or updated based on these patterns. In our AgPSM system, each agent has a profile and based on its profile, agent can perform various tasks. An agent's profile composes of abstract and concrete portion, where abstract profile cannot be configured but concrete profile is configurable during install time. Each agent has an associated configuration file that holds its concrete profile portion so the agent will load its conf. file and load its tasks from there during startup. In addition to a conf. file, the agent can also have a rule file. For example, the knowledge based agents have a rule file. Once the agent is up and running, it learns from the system about neighboring agents and resources, then updates its profile dynamically. An AgPSM also has a Naming and Registration Service (NRS) that acts as a gateway for the agents in the AgPSM to communicate within and outside the cloud. Figure 2 describes the structure of an agent in an AgPSM.

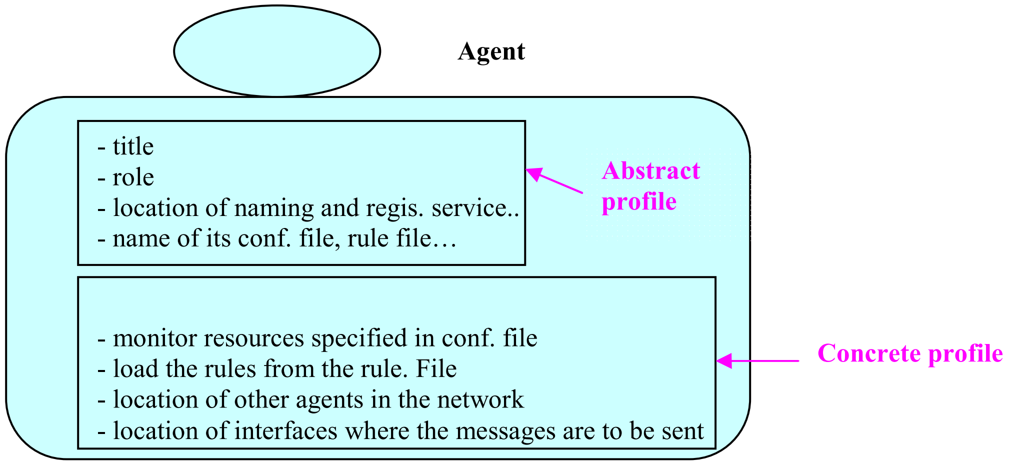

Agent
- title
- role
- location of naming and regis. service..
- name of its conf. file, rule file…
Abstract profile
- monitor resources specified in conf. file
- load the rules from the rule. File
- location of other agents in the network
- location of interfaces where the messages are to be sent
Concrete profile

Figure 2. A sample agent with its abstract and concrete profile structure

An AgPSM is located within a single cloud provider's network (Figure 3), and two AgPSM can communicate with each other as well. The number of agents as well as numbers of each type of agent is configurable for each AgPSM. There can be one or more AgPSMs in a cloud. Some of the agents are listed in Table 1

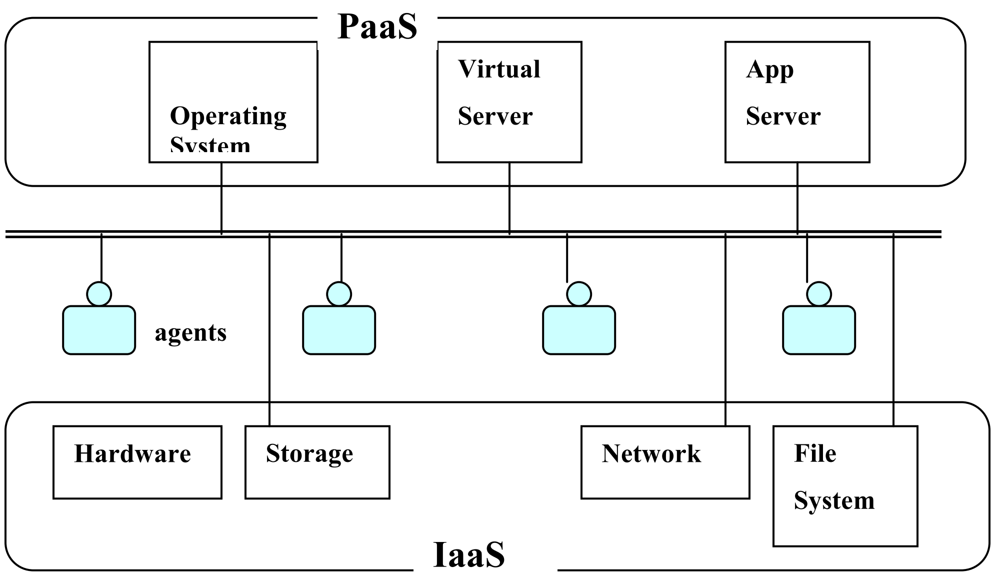


Figure 3. An Agent based proactive system management AgPSM in a IaaS and PaaS Cloud Service

.

Agent	**Profiles**	**Input(s)**	**Output(s)**
Watchman Agents (WA)	Agent that monitors the applications and how much of memory, CPU, disk and other system resources the application runs on. It sends periodic updates to the KA.	Configuration file.	Success/failure; send message to other internal agents
Knowledge-based Agents (KA)	These agents hold the knowledge about the system, its resources and applications. The knowledge base is updated by receiving messages from WA and is used by PA to make analysis and decisions. KA constructs the patterns based on the data collected.	Receive message from other agents, configuration and rule file.	Success/failure; send message to other internal agents
Performance Monitoring Agents (MA)	These agents monitor all system resources, network ports and traffic. They take system snapshot at any point in time about CPU, memory usage, monitoring number of incoming and outgoing calls etc.	Configuration file	Success/failure; send message to other internal agents
Proactive Agents (PA)	The core of AgPSM system is the proactive agents. These agents are capable of data and pattern analysis, taking an action or a decision based on system status, events and former pattern knowledge about the system.	Receive message from other agents, configuration and rule file.	Takes action, generates reports, sends message or warnings to external interfaces.

Table 1. List of Agents in a Proactive System Management System

NAMING AND REGISTRATION SERVICE (NRS)

All agents use naming and registration service to register when they come up so that they can know about one another (location IP address and port). There are a few options of communication interface between agents such as using extensible markup language (XML), CORBA IDL or any other messaging service. C++, Java are some of the languages that can be used to design the agents. The format of the configuration and rule files is extensible markup language (XML) because XML files are easy to create, read, modify and maintain (Harold and Means 2002). NRS also keeps track of all its agents within an AgPSM, so when an agent goes down or crashes, NRS can restart it, and if not possible, it can spawn another similar agent such that the concrete profile of the previous dead agent can be reused and users or administrators do not have to intervene. This ability of AgPSM enables it to provide stable, scalable, robust and reliable proactive system management.

WATCHMAN AGENT (WA)

The watchman agent monitors application(s) and loads the events to monitor from its configuration file. The application to be monitored forms a part of software- either from the consumer or from the cloud provider. The configuration file holds the details of the application location (IP address & port), application type (such as a web server or an application server) and details on monitoring such as monitor memory that the application is consuming, Central Processing Unit (CPU) of the system, disk or other operating system resources it is using, any internal application monitoring (such as monitor the number of threads in a Java based JVM or sub-processes it runs) and collects related data. WA performs application and software monitoring, therefore, programs or scripts to monitor will be based on programming languages such as a set of scripts for Java programming language or scripts for any Java based application server can be based on Java. The data is periodically forwarded to knowledge bases agent (KA) for storage or analysis. In a cloud, WA can be configured to watch multiple servers and devices.

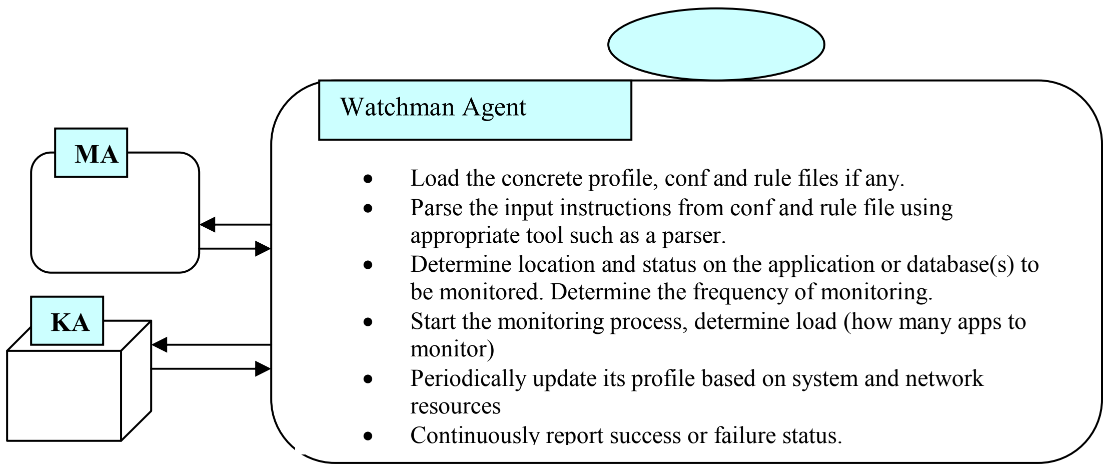

MA
KA
Watchman Agent
Load the concrete profile, conf and rule files if any.
Parse the input instructions from conf and rule file using appropriate tool such as a parser.
Determine location and status on the application or database(s) to be monitored. Determine the frequency of monitoring.
Start the monitoring process, determine load (how many apps to monitor)
Periodically update its profile based on system and network resources
Continuously report success or failure status.

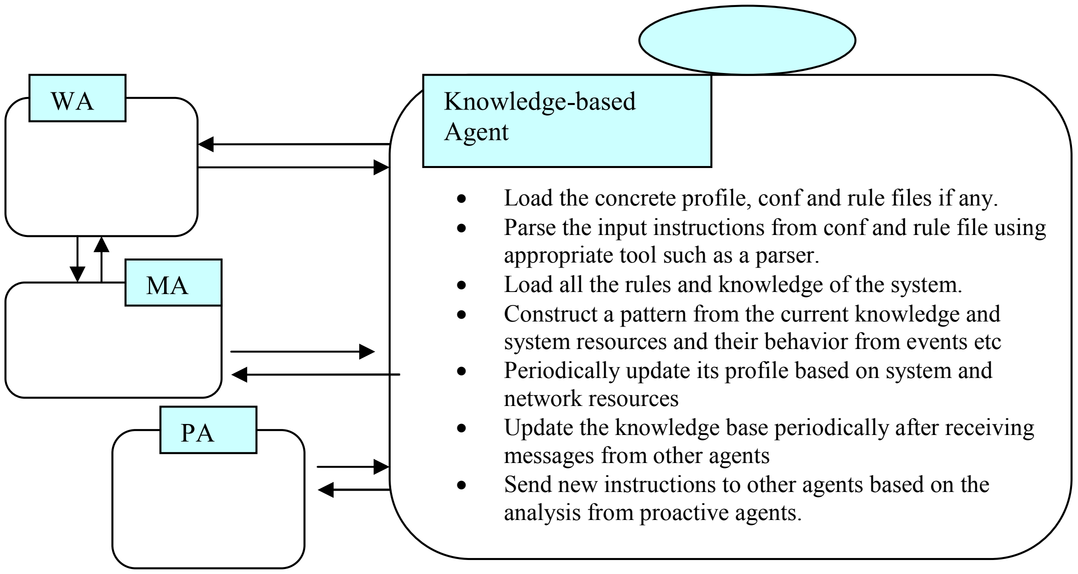
Figure 4. Watchment Agent and its profile specific jobs

PERFORMANCE MONITORING AGENTS (MA)

Monitoring of system resources such as memory, CPU, network and disk usage forms a very important part of a monitoring system. It is quite possible that the cloud infrastructure could cause performance issues due to varying number of applications, thereby affecting the CPU, network and disk I/O sharing, amount of read writes on the storage network etc. Performance Monitoring Agents monitor the system resources, network traffic and any other activity related to system. These agents differ from the Watchman Agents because WA monitors applications, software, database and their impact on system while MA monitors all system resources and their performance (i.e. everything other than the software or application). The idea to use two types of agents to perform monitoring is due to the huge amount of resources to be monitored and to segregate software monitoring from system monitoring. Moreover, system monitoring is mostly based on using tools or scripts that work on an operating system, so a set of scripts written for an operating system can be used to develop the MA.

KNOWLEDGE-BASED AGENTS (KA)

A knowledge base is a collection of knowledge or information that can be stored, updated, used to make decisions. The knowledge either can be stored in a database or in a file based system. The knowledge-based systems are categorized mainly into two major types: Machine-readable knowledge bases store the knowledge in the computer readable format and a set of rules and algorithms perform the analysis on the data; Human-readable knowledge bases store data that can be read by humans in order to perform reports and analysis. In AgPSM, we have a hybrid system where the machine-readable knowledge base uses a set of rules and algorithms to construct patterns on the data and converted into human-readable form for certain scenarios such as reporting. The patterns form the knowledge base of the system that is stored in a database.

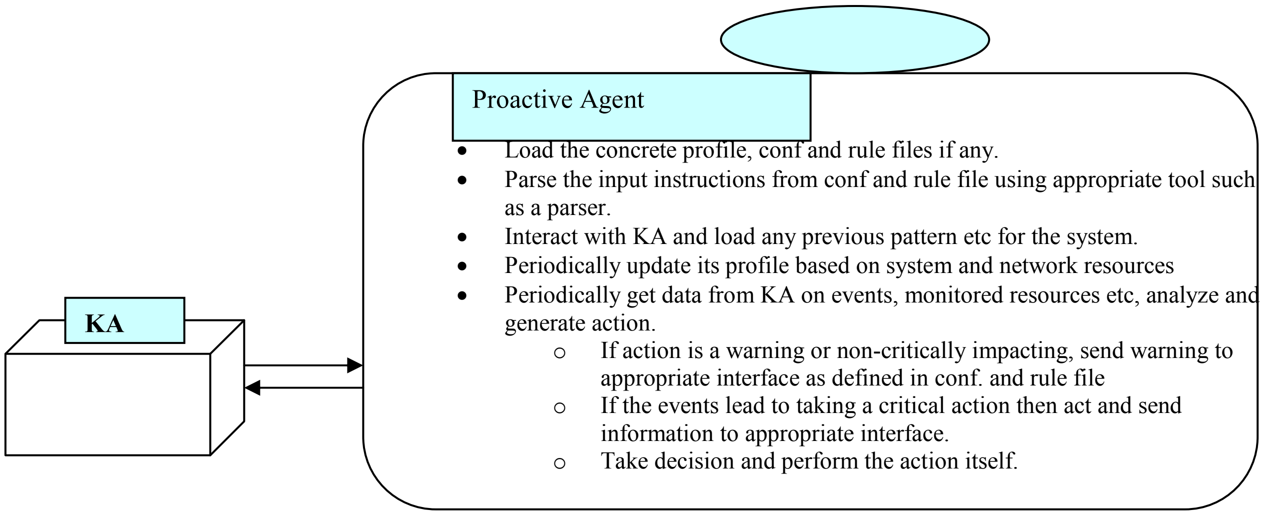


- Load the concrete profile, conf and rule files if any.
- Parse the input instructions from conf and rule file using appropriate tool such as a parser.
- Load all the rules and knowledge of the system.
- Construct a pattern from the current knowledge and system resources and their behavior from events etc
- Periodically update its profile based on system and network resources
- Update the knowledge base periodically after receiving messages from other agents
- Send new instructions to other agents based on the analysis from proactive agents.

Figure 5. Knowledge based Agent and its profile specific jobs

PROACTIVE **A**GENTS **(PA)**

Proactive system level management deals with identifying software and system issues early on in production based on events and rules collect and analyze the related data and result in intelligent diagnostic and warnings about the system status at point in time. In AgPSM proactive agents form the core of system, perform analysis on the data and patterns constructed by KA and act as the decision makers to take action. These agents can be configured for decision-making, reporting the pattern analysis or both based on any cloud requirements.



- Load the concrete profile, conf and rule files if any.
- Parse the input instructions from conf and rule file using appropriate tool such as a parser.
- Interact with KA and load any previous pattern etc for the system.
- Periodically update its profile based on system and network resources
- Periodically get data from KA on events, monitored resources etc, analyze and generate action.
  - If action is a warning or non-critically impacting, send warning to appropriate interface as defined in conf. and rule file
  - If the events lead to taking a critical action then act and send information to appropriate interface.
  - Take decision and perform the action itself.

Figure 6. Proactive Agent and its profile specific jobs

AN EXAMPLE OF AgPSM IN THE CLOUD

Let us consider a hypothetical scenario where there are multiple servers in the cloud and AgPSM is installed and configured. Figure 7 shows the system configuration in the cloud. There is an NRS and four type of agents running – WA, PA, KA and MA. We discuss two cases here – during archiving of logs where AgPSM agents can detect uneven activity; and resolving a resource deadlock situation where AgPSM can take decision based on the patterns.

Detecting uneven archiving of logs pattern - All applications, databases and resources create log files on the system. Logs can be overwritten, updated or (and) archived. Archiving of logs can be done by stating threshold parameters such as the maximum log file size or date before the log can be archived. It is possible that on some days there is too much activity on the system and logs are written more frequently resulting in large disk I/O and impact on CPU of the system. It can cause log file size to reach the threshold much earlier than other cases and can result in system performance due to too much I/O. In AgPSM system, agents collect data on the system performance. So while watchman agent has lists of tasks an application or software is performing on the system (log archival being one of them), performance monitoring agent is collecting data about performance of the system, CPU usage and disk I/O; and this data is fed to the knowledge base by knowledge based agent. Knowledge based agent constructs a pattern based on activity, and when the usage changes. Before an application failure happens because of too much disk I/O, CPU usage or shortage of space or virtual memory, PA feeds on the knowledge base and detects the uneven pattern in the system (due to CPU-intensive archival task) and constructs a system view for the consumer as well as cloud provider (whoever appropriate) so that required changes can be made to the software.

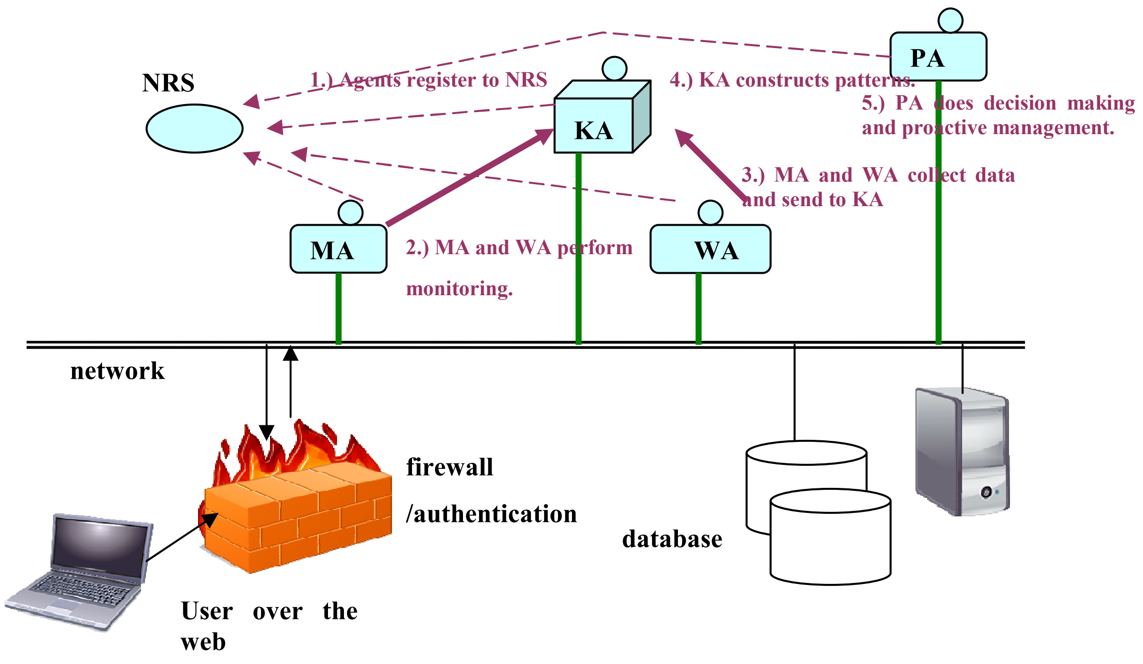

NRS
1.) Agents register to NRS
KA
4.) KA constructs patterns.
PA
5.) PA does decision making and proactive management.
3.) MA and WA collect data and send to KA
MA
2.) MA and WA perform monitoring.
WA
network
firewall /authentication
database
User over the web

Figure 7. Multiple agents running in an AgPSM system in the cloud.

Mitigation failure because of a resource deadlock – We consider a scenario where too many client requests can hit a server at some point and cause the server to become inactive and not respond to any further requests. The server does not crash but is in an inconsistent state. In such a case, the process stops responding to requests and needs to be explicitly restarted. The root cause analysis is extremely difficult and could be anything such as bad code resulting in incorrect memory usage or a deadlock. A deadlock is a situation where in two or more competing actions wait for the other to finish, and thus neither ever does. We consider a deadlock situation in this example where Java process is running in the server. Java is one of the most popular object-oriented languages used in application and web servers today. In Java to perform tasks, new threads can be started from a common thread pool and the maximum number of threads can be configured in the pool. It is possible that during a deadlock all threads are blocked as they try to access the resources and causing a deadlock. It is generally hard to detect such scenario by monitoring tools because the server or the process remains up but inactive. Various tools allow a Java based thread dump of the Java Virtual Memory to analyze the thread resource contention. However, it is important to proactively detect such a scenario before it happens, by active monitoring, data collection and pattern construction. In AgPSM system, agents monitor the resources actively and collect relevant data. Therefore, while WA reports the status of the process as still active, MA detects that the number of incoming client requests on the network keep piling up along with possible some spike in CPU usage. This data collection is sent to KA that can construct an uneven pattern on the system. This pattern causes the proactive agent to perform any of the actions based on its decision-making configuration – alert users, automatically restart the inactive server, or temporarily allocate more resource for a short time while a long-term removal of deadlock can be developed.

These two examples demonstrate the flexible and configurable nature of the AgPSM system. It is different from simple monitoring and diagnosis because proactive agents can take the necessary actions in addition to reporting and analysis.

AGENT COMMUNICATION WITHIN THE CLOUD AND OUTSIDE THE CLOUD

The advantage of using agent technology on the cloud software management is that agents can be configured to scale and shrink in number as well as functionality. In addition if two cloud providers come to an agreement related to working together on a larger software management system (and design a security and service level related policy for communication, allowing messages, data and knowledge transfer), then the AgPSM can scale up to more than one cloud. The latest trend in cloud providers built-in communication or interoperability is *Intercloud.*

Each AgPSM has an NRS that holds information about agents the management system manages. During startup, each agent loads its profile, sends its location information to the NRS and receives message from NRS related to location of other agents such as KA. When there is a need for AgPSM(1) to talk to AgPSM(2), within or outside the cloud, related information on the network address and location of AgPSM(1) and AgPSM(2)'s NRS appropriately.

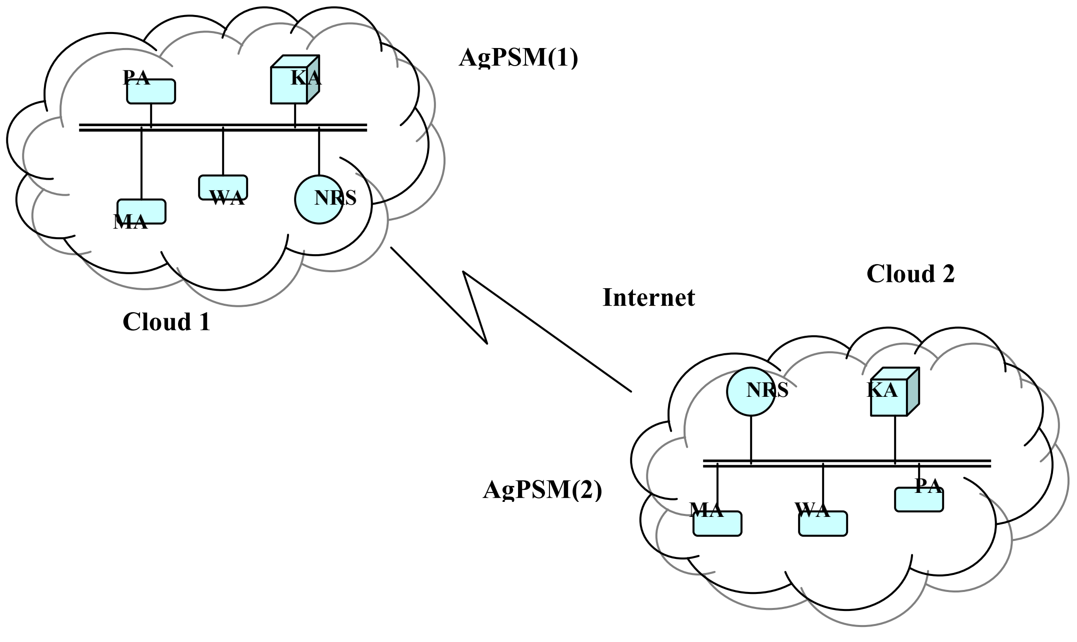


Figure 8. Interaction of two AgPSM systems between clouds

CONCLUSION

Cloud computing is the emerging technology that promises IT organization cost effective, faster time to market solutions by providing an off-the shelf service list the IT organizations can choose from in a pay-as-you-go basis. It is important for cloud vendors to provide a robust architecture of flexibility, scalability and secure access to continue its move to cloud. In addition, it is equally important for cloud service providers to have proactive management tools and servers installed to minimize any potential failures in running the applications in the cloud. This paper discuses some of the later techniques using agents in a cloud. An agent based proactive system management approach in the cloud provides greater flexibility because of their ability to load balance; platform independence due to programming languages such as Java and frameworks such as Spring; scalability as the agents can reside in a pool and based on the load new agents can be quickly spawned on central or distributed architected systems; and resilient fail safe model as the agent architecture allows to leverage the design where agents can be spawned on demand, they are knowledge based to detect patterns, system or resource failures resulting in a fault tolerant cloud. An agent based architecture allows new agent allocation when there is a need for the same, while it also allows for the agents/resources to be released when the demand is less. In addition, the agents dynamically construct patterns and are capable to detect potential system issues and failures. A robust architecture such as this along with proactive management in the cloud tends to increase customer confidence and is a key to more successful deployments to cloud.

REFERENCES

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., and Stoica, I. "A view of cloud computing," *Communications of the ACM* (53:4) 2010, pp 50-58.
2. Chandrasekaran, B. "Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design," *IEEE expert* (1:3) 1986, pp 23-30.
3. Chen, G., Lu, J., Huang, J., and Wu, Z. "SaaAS-The mobile agent based service for cloud computing in internet environment,"*Proceedings of the Sixth International Conference on Natural Computation (ICNC)*, IEEE, Yantai, Shandong 2010, pp. 2935-2939.
4. CRN.Technology.News 2011, http://www.crn.com/slide-shows/applications-os/225701829/10-biggest-cloud-outages-of-2010-so-far.htm?pgno=1
5. Dadhich, P., and Dutta, K. "Security Issues in Mobile Agents," *International Journal of Computer Applications IJCA* (11:4) 2010, pp 24-28.
6. Dikaiakos, M.D., Katsaros, D., Mehra, P., Pallis, G., and Vakali, A. "Cloud computing: Distributed internet computing for it and scientific research," *Internet Computing, IEEE* (13:5) 2009, pp 10-13.
7. Harold, E.R., and Means, W.S. *XML in a Nutshell* O'Reilly & Associates, Inc. Sebastopol, CA, USA, 2002.
8. Lange, D.B., and Oshima, M. "Seven good reasons for mobile agents," *Communications of the ACM* (42:3) 1999, pp 88-89.
9. Nwana, H.S., and Ndumu, D.T. "A perspective on software agents research," *The Knowledge Engineering Review* (14:2) 1999, pp 125-142.
10. Taneja, N., and Taneja, A. "Pool-Based Heterogeneous Mobile Agents for Network Management,"*Proceedings of the Ninth Americas Conference on Information Systems*, Tampa, Florida, 2003, p. 240.
11. Williams, M. 2011, February, http://www.readwriteweb.com/cloud/2010/02/top-5-cloud-outages-of-the-pas.php#
12. Zhang, Z., and Zhang, X. "Realization of open cloud computing federation based on mobile agent,"*Proceedings of the*, IEEE, 2009, pp. 642-646.