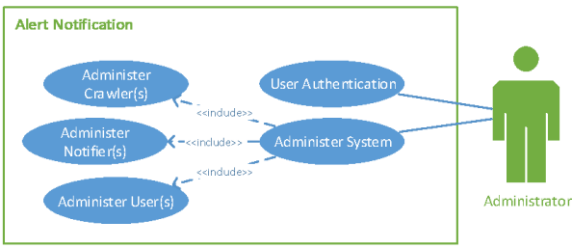

Figure 2. Use Case diagram for administrator functionalities of ANS

Privileges given to the administrator refer to administration and maintenance of specific modules, such as the crawler(s) and notifier(s). These functionalities include creating, adding, editing and dropping crawler(s) and notifier(s) through a user-friendly interface. Additionally, the administrator manages the user administration and has an overview of the complete system functionalities, including authorisation, and providing technical help to public users.

A specific function can be added to support accounting, especially interesting for those cloud providers, that would like to integrate the push notification functionality to their existing conventional web site solutions. This will enable the "pay-by-use" cloud related concept of the realised software as a service.

3.3. The Back-End Functionalities

The system's core functionality is crawling the inserted web page for the required keyword(s) with a specific frequency, which can have predefined values, such as, each minute, 10 minutes, hour, 2 hours, day, etc.). Naturally, the users themselves are provided with the opportunity to select this frequency from a set of predefined time intervals in accordance to their own personal preferences. The system crawls the keywords entered by the users to find their desired information. Note that selecting a high frequency, such as each minute, can be a costly function demanding high data throughput.

The advantage of the ANS system over the other similar systems lies in the opportunity for the user to select the delivery channels and means to receive the notification by the server (E-mail, SMS, Google Cloud Messaging, Skype, Twitter or Facebook messaging, etc.) [11]. Also, the system can be realised as a web page where one can see all notifications, or as an add-on tool for the web browser. An example of such an occurrence is the alert bell on Google Chrome.

3.4. System architecture

The general system architecture, based on the web based notification system presented is presented in Fig. 3 [2].

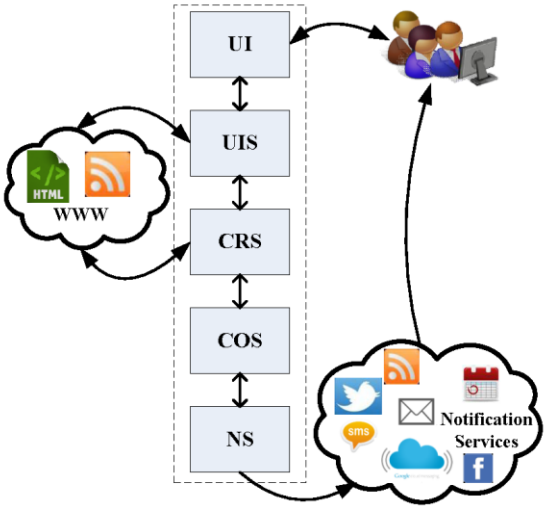

Figure 3. ANS System Architecture

It consists of four services that provide the main functionalities of the system.

The main User Interface Service (UIS) is realised by a web browser. UIS is used to define the alerts, including the macro recorder of a sequence of actions and simple web interface components.

The Crawler Service (CRS) scans the web sites and sends the appropriate content to the Controller Service (COS). Then COS performs the triggering functionality. It also activates the CRS periodically in order to scan the web page if there are some changes in the content.

Another tasks for COS are to analyse the provided content, to filter the relevant information from the junk and to store the relevant data. If the user defined condition is fulfilled, then COS triggers the Notifier Service (NS) to deliver the notification to the user. NS then activates the corresponding notification delivery channels that send push information.

3.5. System integration

The system should be able to do what a user does when he or she visits a web page. This means that apart of visiting a web page and finding an occurrence of a text within a paragraph, the system also simulates the clicking on certain buttons or entering a value in certain fields. It will simulate a human entering data on an interactive web page, with possibility of selecting a certain radio button, or an item in a drop down list, or entering a value in a certain text field, or clicking activation button. Then the system will be able to trace the file and find a predefined keyword phrase, by detecting if there is a change in the context or in the occurrence of the keyword phrase.

Those web sites that have push notification, alerting system or API can be used in the system. This system will actually build push notification for any web page.

The alerting system can be realised conventionally by e-mail, Skype, Twitter or Facebook messaging, or can be realised as a web page where one can see notifications, or as an add-on tool for the web browser. An example of occurrence is the alert bell on Google Chrome.

It can be thought of as a system for users who would like to be notified for a certain change, or for cloud service providers who would like to build push functionality for classical web sites that do not support this technology.

4. Cloud-based System Architecture & Design

In this section we specify how the ANS system is redesigned to be scalable and elastic when hosted on a cloud environment.

4.1. Methodology

Since the system is designed as a service, it is obvious that a relatively large number of users will be discrepant in different time. For example, the peaks for sports games will appear during the weekend, rather than during a week. Another example is for the scientists, that is, they will require the crawling more frequently after finishing the conference or accepting a paper. These issues lead us to redesign the system in order to migrate in the cloud and to utilise its "unlimited" flexible resources.

As a baseline for migration, we use the scalable and elastic cloud architecture for e-Assessment, proposed by Ristov et al. [3]. The idea is to determine (if possible) a dynamic part of the system and host it in the cloud, while the static part can be hosted either on cloud or on-premise.

The static part should be active always, while the dynamic part will utilise certain amount of cloud resources organised in VM instances, which can be instantiated by demand.

Fig. 4 presents the new cloud based system architecture concept. The next section presents the analysis to determine which services of the system are static and dynamic.

4.2. Static and Dynamic Services

Fig. 3 presented the main services of the system. The following analysis addresses the service that mostly depends on the number of the subscribers and thus when to be activated or deactivated.

Table 1 presents the service splitting to the static and dynamic services. UIS and COS are static part of the system, while CRS and NS are dynamic.

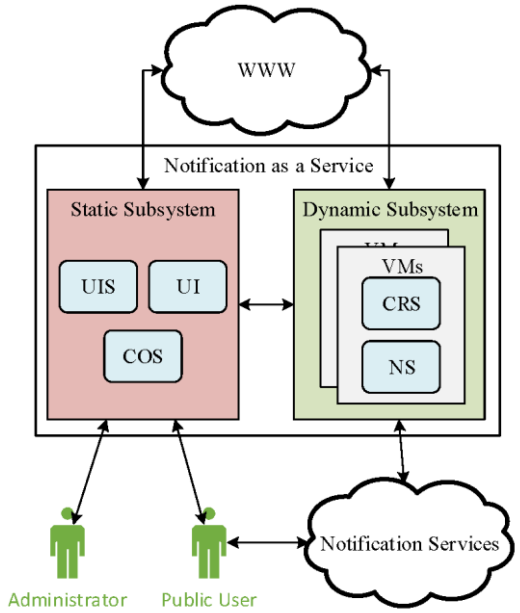

Figure 4. Cloud based system architecture

Table 1. Static and dynamic services

Static services	Dynamic services
UIS	CRS
COS	NS

Let us explain in more detail the information presented in Table 1. UIS should be active all the time since a user can access the system arbitrary. This service does not require huge amount of resources because it is only a presentation layer of the system.

COS is also a static service because it only activates CRS periodically. The other two services are dynamic because they are not used all the time, but only when they are scheduled. Also both CRS and NS directly depend on the number of active users and their scheduling policy for crawling.

4.3. Deployment on the cloud

After determining the static and dynamic parts, they should be deployed on a cloud in a certain resource pool. The dynamic parts should be deployed on the cloud VMs, which will be instantiated or deactivated when necessary, while the static parts should be active always.

Since scalability and flexibility are important properties of each software system, we distributed the proposed solution in four parts as depicted in Fig. 5. The Notifier Pool and the Crawlers Pool are resources that will be provisioned from the cloud provider whereas the Controller and the Public Server may be setup on premises or in the cloud.

Public server, *Controller*, *Notifiers Pool* and *Crawlers Pool* are discussed separately to define their characteristics and behaviour. The improved cloud architecture of the system is depicted in Fig. 5

where the defined services are positioned on corresponding software modules in the cloud architecture.

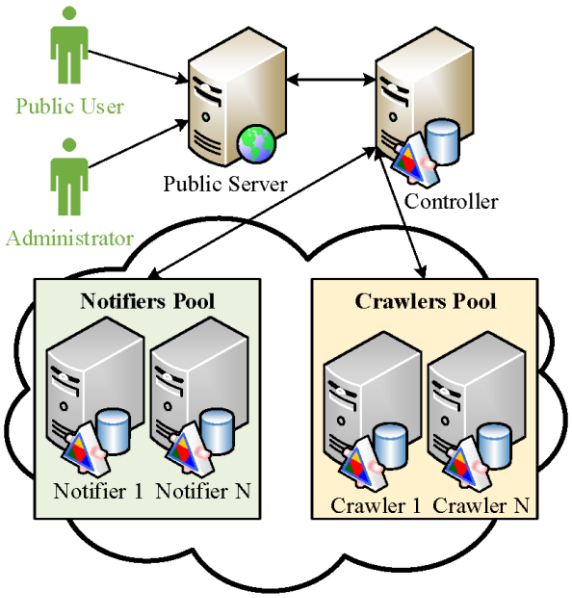

Figure. 5 New ANS cloud-based model

Each service will be deployed as a separate pool. That is, UIS will be hosted on Public server pool, COS on the controller pool, NS on Notifiers pool and CRS on the Crawlers pool.

4.3.1. Public Server. The public server is the publicly accessible part of the system. This is the system entity on which the web application is hosted. Also, it serves as a point where both public users and administrators can use the service through a user friendly interface. UIS module is hosted on this server.

The web application, realising the UIS module, hosted on the Public Server communicates with the Controller in order to insert and pull information that is required by the users.

4.3.2. Controller. The controller realises the COS service and is in fact the "mind" of the system, providing a mechanism that makes all features and functionalities available, by coordinating the tasks. In certain ways the Controller is both the heart and brain of this system, by activating various components and synchronising tasks.

In the scenario depicted in Fig. 5, the Controller is separated from the public server, but they also may work on the same server. Since the system is modular, both the public server and the controller may also be hosted on multiple servers in order to evade bottleneck in these segments.

The Controller communicates with the public server in order to collect user data needed for the system. Additionally, this module orchestrates the crawlers and notifiers available in the crawler pool and notifier pool respectively.

The controller tasks are composed by the user input. Later on, it distributes the work to each of the crawlers based on this. The controller contacts the crawlers and collects the results from their crawling in regular time intervals. Based on these results, tasks for the notifiers are created. Each of the tasks is sent to the notifiers based on their availability.

From a technical point of view, the controller consists of two services and a database. The first service is used to access the results of the system work. The second service is used for the pool control. The database is a collection of all results available in the system. In a distributed scenario, the database of each controller consists of data collected from the corresponding pools (Crawlers Pool and Notifiers Pool).

4.3.3. Crawlers Pool. This part of the system consists of a collection of crawlers. In this pool multiple servers are available. Cloud computing is especially valuable in this scenario since the system will be more flexible and not all of the servers would have to be active all the time. Additionally this cluster does not need to be homogeneous, so servers with different hardware architectures can be included.

The crawlers, as the name indicates, are crawling the web sites specified in campaigns that the users require based on their query. In order to maximise the performance, the crawlers are also designed to finalise the predefined tasks on a scanned web page first and then continue with a scan on the next web page.

Each crawler has a private database that is filled with data when the controller distributes tasks. All results from the crawling are also kept in this database. When the crawlers have finished their task, they asynchronously provide the controller with information that they are ready for data synchronisation. When both the controller and the crawler are ready for data synchronisation, they invoke it.

In order to provide easy deployment, VMs are constructed for different hypervisors and operating systems. Since cloud computing is a natural habitat for this kind of systems, we also provide VMs configured for Windows Azure, Amazon EC2 or Google Compute, and also for the cloud computing open source frameworks, such as OpenStack or Eucalyptus.

4.3.4. Notifiers Pool. This module is very similar to the Crawler pool. Their deployments and requirements are practically the same. Both pools differ only at the application level. The notifier is required to prepare and send the notifications to the users. In order to do so, each notifier is equipped with several adapters aiming to make the notifications available through several services such as:

- Facebook message;

- e-Mail;
- Google Cloud Messaging;
- Twitter;
- LinkedIn; etc.

Another difference is that the notifiers only get data from the controller. They don't synchronise data with the controller in the opposite direction, but only send notifications to the controller if an error occurred while sending notification to the users.

5. Discussion

This section presents the approach based on intelligent agent. We evaluate our solution by comparing it with the conventional push notification services.

5.1. Intelligent Agent Approach

In our earlier paper, we discuss how this solution fits in an Internet-based communication realised by an intelligent agent [12]. The idea is that instead of a human, the intelligent agent scans the web page and finds certain occurrences of keywords. According to the Franklin's definition of intelligent agents, as shown in Fig.6, it acts as autonomous agent where the web is the environment for sensing, the act of sensing is realised by scanning or crawling, and the algorithm that produces state change is realised by comparison [13]. The actuators are identified by the notification delivery channels. The agent characteristics include at least the following properties: reactive, autonomous, continuous and trustworthiness.

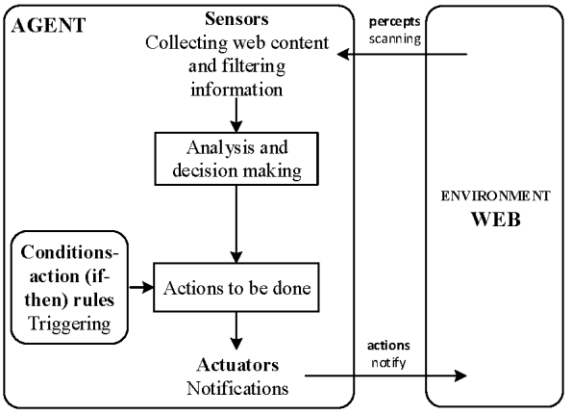

Figure. 6 Agent Functionalities

5.2. Evaluation

We have developed a prototype of a cloud-based alert notification based on the specification elaborated in this paper. Our solution is realised with the following technologies .Net 4.5 and MS SQL. The prototype is installed on a Windows Azure cloud. We have tested the solution by specifying several campaigns to search a predefined keyword or a change of information associated to a given keyword.

The first tested campaign was based on the access on a personalized Google scholar page. This site includes also push notification, enabling a comparison to evaluate the functionality. The idea was to test whenever there was a change in the number of citations for the personalized page. The functionality test was successful, whenever there was a change in the Google scholar, this was notified by e-mails from both the Google scholar alert (push service) and our ANS service. In 85% cases, the push notifications form google scholar came a little bit earlier, which happened due to the scanning interval set to 6 hours. However, in 15% cases, the notifications from our web services came earlier. It seems that push notifications are scheduled and delayed due to spam or other.

We have already tried several other campaigns, which include sport results and publishing certain names on web sites and all function correctly. Our latest ANS version connects to the Windows Notification Center and uses it as a delivery channel. It is realized via the Toast Notification service which allows sending the push notifications on all windows devices.

We have met several challenges in development of the solution. The following problems were identified and solved:

- *Time demanding crawling service*. Probably this was the highest problem in realization of the overall system. It initiated realization of a cloud-based solution having the property of a highly scalable service.
- *A variety of presentation forms prevent easy retrieval of keywords and their associations*. The existing web solutions build a lot of different techniques, such as cells in adjacent columns or rows, tables, frames, div and other objects. It was a great challenge to cover most of the known presentation patterns.
- *Realization of macro procedures*. This was a highly challengeable task to develop a service that will provide a sequence of keystrokes and mouse clicks instead of a human when accessing a web page and navigating on the links.

6. Comparison to related work

There are several products that enable notification services, but their functionality is limited to push notifications or similar alerts. They do not integrate keyword search and change of information associated to the content in a web page. Li et al. survey four different types of Push notification services (Google's Cloud Messaging, Blackberry Push

service, Microsoft Push Notification service and Apple Push Notification service) [14].

6.1. OS notifications

A lot of new approaches have been implemented by various software producers to enable better notification service. Examples include the new versions of the Mac Lion, Maveriks and Yosemite OS, iOS and Windows 9. The Notification Center in Mac OS and iOS is a built-in application that provides an overview of alerts from applications without interrupting the users. Instead of requiring instant resolution of a notification, the main idea is to display selected notifications until the user completes an associated action. The application enables the users to choose what applications appear in Notification Center, and configure how they are handled. However, they only include those system applications installed on the operating system, and does not support of third party apps.

Although the Notification Center application found in Mac OS and iOS looks sophisticated, still it does not offer the functionalities proposed in this article. It looks more like an advanced delivery channel, rather than supporting the alert notifications for certain web pages or occurrence of keywords and changes in web pages. The realised solution actually looks like a twitter for build-in applications in the operating system without a possibility to upgrade it to a real alert notification.

The versions newer than Windows 8 implement notification systems that communicates with the installed applications. The latest Windows version 10 uses more sophisticated options, and actually transfer all windows's phone notification center options on the desktop. However, all the realized features are implement only push notifications from other applications and does not cover the functionalities specified in this paper.

Growl (http://growl.info/) is an open source application that realises more advanced features than the Notification Center. Using Growl, applications can display small notifications for pre-selected events. Similar to the Notification Center, users can customise and control their notifications. In addition, Growl can send notifications to the Notification Center and use it as a delivery channel.

6.2. Web browsers and intelligent assistants

Web browsers recently include possibilities for notification and integration with OS or with other desktop notification channels. Google Chrome is one more example that builds a possibility to realise push notification. This system works only with web sites that offer push notifications. However, our system has a different goal, it includes the push notification to those web sites that use only push as a communication. Instead of rebuilding the site to enable the push notification, we realise a specialised web service that introduces this feature to the existing site.

Google Alerts (http://www.google.com/alerts) is another application with a similar approach allowing monitoring of interesting new content on the Web. It does not support the functionalities to notify a change in a keyword occurrence as our system does. It enables email updates based on user queries. The queries might search: news, blogs, discussions, books, videos or everything. The main difference between our suggested system and Google's is in the produced results, that is, alerts will fetch all data on the Web and even duplicate same information whereas our system will follow only user specified pages, and therefore duplicate information will not be pushed to the end user.

There are also intelligent personal assistants that integrate notification services. An example is Google Now (https://www.google.com/landing/now/), which works as a natural language user interface to answer questions, make recommendations, and perform actions by delegating requests to a set of web services. It delivers information proactively to the user that it predicts they will want, based on their search habits. It allows a kind of notification, but does not integrate the idea to notify if there is a change in a keyword or information related to a corresponding keyword. SIRI (https://www.apple.com/ios/siri/) is the Apple product that works as a intelligent digital personal assistant, but does not deliver the functionalities that we have developed for our notification service.

6.3. Other notification systems

There are many examples where push or pull notifications are used in various areas.

Lan and Nguyen presented a low delay push-pull based application layer multicast for video streaming on P2P networks [15].

Gore et al. designed a real-time notification service based on CORBA in order to obtain a scalable event-driven communication [16].

Hermes is a notification service for digital libraries, which is created to simplify the process of searching and browsing scholarly materials, proposed by Feansen et al. [17].

Gu et al. proposed and Enterprise Cloud Bus (ECB) framework in order to combine the WS-Notification (needed for ESB) with the cloud [18].

Quiroz and Parashar presented the design of the distributed content based notification broker for WS-Notification, which is based on publisher subscriber (push) notification [19].

Since push notifications are consistent, but inefficient, and pull notifications are inconsistent, but efficient, Huang and Wang proposed Push\&Pull model for user-oriented resource monitoring for cloud computing [20].

7. Conclusion and future work

In this paper we have presented the Alert Notification System realised as a scalable cloud solution that delivers software as a service. It is a newly proposed push service implementation aiming at alerting Internet users in case of a change in an existing text or appearance of a certain keyword phrase at web sites that are of the users' interest.

The difference to similar products and notification support of modern OS is that the triggering can be specified only on operating system events, such as application generated events, receipt of e-mails or push notifications. These systems can not be configured to scan certain web pages and trigger an event based on a occurrence of a specific information or change of an information on the analysed web page.

The sole purpose of our system is to save users valuable time and effort by providing them with the much needed user specific information at the utmost quickest and efficient manner. Therefore, crawling lies at the heart of this system, enabling us to collect the user specific information, which is to be later delivered to the users themselves, by means which they have selected on their own (E-mail, SMS, Google Cloud Messaging, Skype, Twitter or Facebook messaging, etc.)

This system is prone to peak loads by the arbitrary number of the subscribers and their activities, that is, their configurations for crawling and notifications. It retains the performance for minimal cost.

The main contribution of this paper is in developing a scalable and elastic cloud architecture, based on identifying static and dynamic software modules, and their organisation in various VMs that can be instantiated or closed on demand. We have realised the system and provided a proof of concept for such design.

As future work, we intend to realise a comprehensive functional and performance test of the new proposed scalable and elastic architecture, providing experimental proof of increased key success parameters that show scalability and elasticity.

8. References

[1] M. A. Fox, P. F. King, and S. Ramasubramani, "Method and apparatus for maintaining security in a push server," Jul. 16 2002, US Patent 6,421,781.

[2] M. Gusev, S. Ristov, G. Velkoski, and P. Gushev, "Alert notification as a service," in *MIPRO, 2014 Proceedings of the 37th International Convention*, IEEE Conference Publications, Opatija, Croatia, 2014, pp. 334–339.

[3] S. Ristov, M. Gusev, G. Armenski, K. Bozinoski, and G. Velkoski, "Architecture and organization of e-assessment cloud solution," in *IEEE Global Engineering Education Conference (EDUCON), 2013 IEEE*. Berlin, Germany: IEEE, Mar. 2013, pp. 736–743, best Paper Award.

[4] B. Medjahed, "Dissemination protocols for event-based service-oriented architectures," *Services Computing, IEEE Transactions on*, vol. 1, no. 3, pp. 155–168, July 2008.

[5] The Internet Society, "Simple mail transfer protocol (SMTP)," 2001. [Online]. Available: http://www.ietf.org/rfc/rfc2821.txt

[6] Wisegeek. [Online]. Available: http://www.wisegeek.org/ what-is-push-email.htm#

[7] M. Hauswirth and M. Jazayeri, "A component and communication model for push systems," in *Software Engineering?ESEC/FSE?99*. Springer, 1999, pp. 20–38.

[8] E. Bozdag, A. Mesbah, and A. Van Deursen, "A comparison of push and pull techniques for ajax," in *Web Site Evolution, 2007*. WSE 2007. *9th IEEE International Workshop on*. IEEE, 2007, pp. 15–22.

[9] M. Kostoska, G. Velkoski, K. Bozinoski, S. Ristov, and M. Gusev, "Service agents for calendar exchange," in *Local Proc. of the Fifth Balkan Conference in Informatics, ser. BCI '12*. Novi Sad, Serbia: CEUR-WS.org, 2012, pp. 142–146. [Online]. Available: http://ceur-ws.org/Vol-920/

[10] M. Gusev, S. Ristov, G. Velkoski, A. Guseva, and P. Gushev, "Scalable architecture of alert notification as a service," in i-Society, 2014 Proc. of the Int. Conf. on Information Society, Nov. 2014, pp. 82–87.

[11] Google. [Online]. Available: http://developer.android.com/google/gcm/index.html

[12] M. Gusev, S. Ristov, P. Gushev, and G. Velkoski, "Alert notification as a new model of internet-based transactions," in *Telecommunications Forum (TELFOR), 2014 22nd*, Belgrade, Serbia, Nov 2014, pp. 967–970.

[13] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," Intelligent Agents III Agent Theories, Architectures, and Languages, pp. 21–35, 1997.

[14] N. Li, Y. Du, and G. Chen, "Survey of cloud messaging push notification service," in *Information Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on*, Dec 2013, pp. 273–279.

[15] H. B. T. Lan and H. S. Nguyen, "A low-delay push-pull based application layer multicast for p2p live video streaming," in *Knowledge and Systems Engineering (KSE), 2011 Third International Conference on*. IEEE, 2011, pp. 104–111.

[16] P. Gore, I. Pyarali, C. D. Gill, and D. C. Schmidt, "The design and performance of a real-time notification service," in Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE. IEEE, 2004, pp. 112–120.

[17] D. Faensen, L. Faultstich, H. Schweppe, A. Hinze, and A. Steidinger, "Hermes: a notification service for digital libraries," in Proceedings of the *1st ACM/IEEE-CS*

joint conference on Digital libraries. ACM, 2001, pp. 373–380.

[18] P. Gu, Y. Shang, J. Chen, M. Deng, B. Lin, and C. Li, "Ecb: Enterprise cloud bus based on WS-notification and cloud queue model," in *Services (SERVICES), 2011 IEEE World Congress on*. IEEE, 2011, pp. 240–246.

[19] A. Quiroz and M. Parashar, "Design and implementation of a distributed content-based notification broker for WS-notification," in *Grid Computing, 7th IEEE/ACM International Conference on*. IEEE, 2006, pp.207–214.

[20] H. Huang and L. Wang, "P&P: A combined push-pull model for resource monitoring in cloud computing environment," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, July 2010, pp. 260–267.

