UDC 681.3.016=111

D.E. Bortyakov, S.V. Mescheryakov, D.A. Shchemelinin

INTEGRATED MANAGEMENT OF BIG DATA TRAFFIC SYSTEMS IN DISTRIBUTED PRODUCTION ENVIRONMENTS

An effective management of a cloud-based distributed production infrastructure with multiple service equipment and big data traffic is impossible without a centralized automated control system. The purpose of integrated control includes on-line monitoring of the current state of the entire distributed infrastructure according to key performance indicators, automatic alerting in emergency cases or outages and, when possible, auto-restoring of production services. All these question described in this paper are based on the work of a particular company.

AUTOMATED CONTROL; DATABASE; BIG DATA; INTEGRATION; CLOUD COMPUTING; DISTRIBUTED ENVIRONMENT.

Д.Е. Бортяков, С.В. Мещеряков, Д.А. Щемелинин

ИНТЕГРИРОВАННОЕ УПРАВЛЕНИЕ БОЛЬШИМИ ДАННЫМИ ТРАНСПОРТНЫХ СИСТЕМ В РАЗЛИЧНЫХ ПРОИЗВОДСТВЕННЫХ УСЛОВИЯХ

Эффективное управление обширной производственной инфраструктурой с многочисленным сервисным оборудованием и большими потоками данных невозможно без централизованной системы автоматического контроля. В задачи интегрированного управления входит непрерывный мониторинг текущего состояния всей распределенной инфраструктуры по ключевым производственным критериям, автоматическое оповещение в случае критических ситуаций или отказов и, по возможности, автоматическое восстановление работоспособности сервисов. Все эти вопросы рассмотрены в данной статье на примере конкретного предприятия.

АВТОМАТИЗИРОВАННОЕ УПРАВЛЕНИЕ; БАЗА ДАННЫХ; БОЛЬШИЕ ДАННЫЕ; ИНТЕГРАЦИЯ; ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ; РАСПРЕДЕЛЕННОЕ ОБОРУДОВАНИЕ.

Overview of RingCentral Cloud Infrastructure

RingCentral (RC) is the international IT company with a central office in Silicon Valley (San Mateo, CA, USA) provisioning voice over Internet protocol (VoIP), mobile platforms, short message service (SMS), email, fax, conference and other IP communication services for more than 300,000 business customers in the USA, Canada and Europe [1].

RC production system is a cloud based multi-component infrastructure with big data traffic across all the distributed environments. Fig. 1 shows RC a scalable architecture located in 4 data centers on the West Coast and East Coast of the USA and in Western Europe. RC environments both production and stress test are rapidly growing along with customer demand amounting to 40 % annual rate. So now it consists of more than 3500 hosts including hardware (HW) but mainly they are the virtual machines (VM) for more efficient IT maintenance and saving computing resources.

All production servers are grouped into about 60 pools and RC components (Table 1), each provisioning a particular custom service and functionality or connecting with an external public switched telephone network (PSTN) and other third party providers. The example of workflow for the registration of RingCentral Mobile (RCM) devices in RC system is shown in Fig. 2.

Registration is the initial mandatory action to authenticate in RC system and grant access to the entire set of RC services for mobile users. RCM registration consists of the following steps:

1. RCM sends the authentication request via IP-network to RC system. HTTP request is

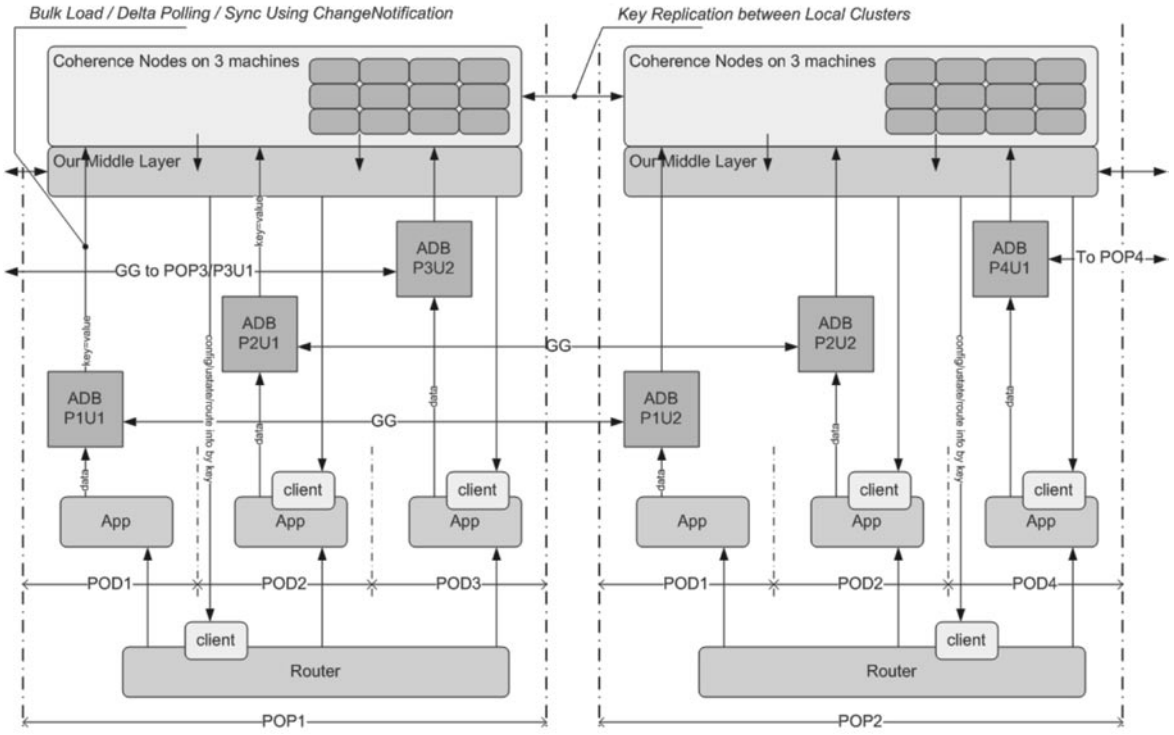

Fig. 1. RingCentral Scalable Architecture

Table 1

Definitions of Some RC Partitioning and Service Components

Abbreviation	Full Name	Definition
ADB	Account Database	Separate database to limit the clients count per POD for better performance and scalability
CDB	Common Database	DB storing common client's data for all PODs in POP
GUD	Global User Directory	Storing all clients global data and routing requests from Common Layer to a particular POD
ISR	Inbound Telco SIP Proxy Server	Routing inbound telecommunication requests to a particular POD based on GUD query results
JWR	JEDI Proxy Router	Routing inbound HTTP requests to JWS of a particular POD based on GUD query results
JWS	JEDI Server	Windows based server running RC Java applications provisioning RC web services such as signup, RCM registration to the clients
POD	Part of Data	The part of environment providing all RC services to clients of a particular ADB for better performance and scalability
POP	Point of Presence	RC environment, either active or standby, located physically in one data center
PWS	Platform Web Server	Windows based server running RC Java applications provisioning all RC services to mobile users
TAS	Telephony Access Server	Windows based server running RC applications provisioning telephony calls, inbound faxes and other VoIP services to RC clients

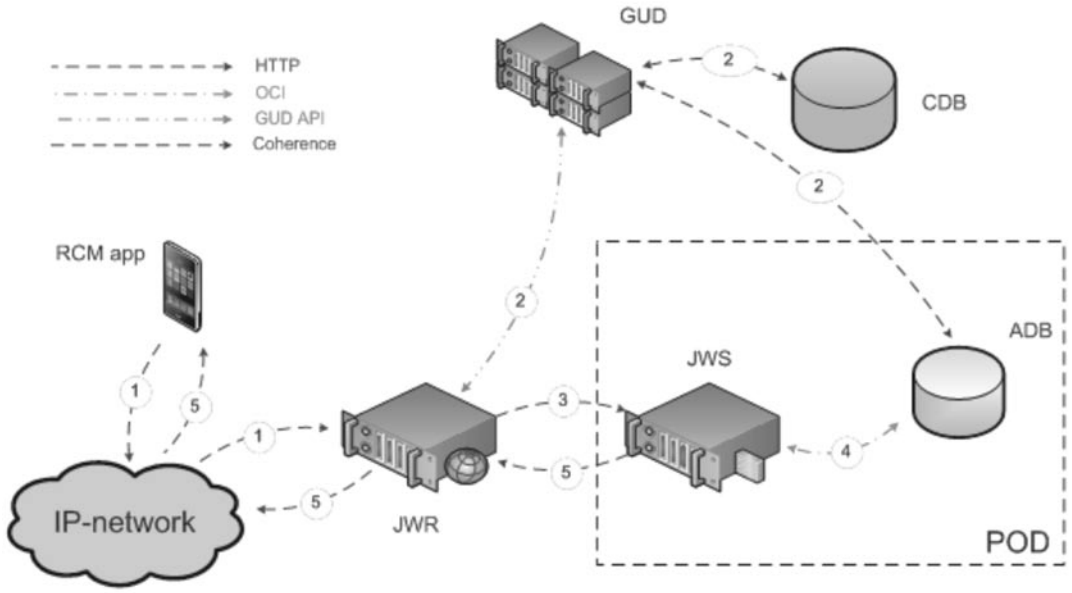

Fig. 2. Workflow for RCM Registration Service

transmitted and processed by JWR router.

2. JWR sends a query against GUD to determine the destination POD number of that particular RCM client. GUD global data is always up-to-date due to Golden Gate (GG) replication between CDB and all ADB databases via Oracle Coherence Interface (OCI).

3. HTTP request for authentication is routed via JWR through JWS server to a particular POD based on GUD query results.

4. JWS sends a query against local ADB in POD to grant or deny access to RC system for that particular request.

5. If the account exists in ADB and it is successfully registered, JWS server compiles corresponding XML configuration file and sends it back to RCM end device.

Zabbix Integrated Monitoring System

Zabbix enterprise-class monitoring system [2] is introduced as an open source integrated solution to manage such a huge multi-host distributed cloud infrastructure. Zabbix is effectively used for real-time monitoring, alerting, troubleshooting, an automated control, capacity analysis and other purposes.

A simple monitoring architecture consists of Zabbix Web Server, SQL DB server and a series of proxies to get counters from external RC hosts with a regular polling time interval and store performance data into DB (Fig. 3). There are different ways to send the results of measurement to Zabbix:

1. Zabbix agent installed on a host is automatically configured for predefined system metrics like CPU utilization, memory usage, disk free space, network packets loss, server ping, service availability, etc.

2. Zabbix trapper that is helpful for implementing any custom script on a host is sending the results to Zabbix via Simple Network Management Protocol (SNMP).

3. An external check which doesn't require Zabbix agent installation and allows executing any custom SQL queries or other code script against a host is returning the results back to Zabbix. Some examples of Perl scripts are given [3].

All the events, the latest metric values and historical trends are integrated in Zabbix DB. Any database of object-relational type (MySQL, PostgreSQL, Oracle, etc.) can be used as Zabbix DB.

Zabbix configuration includes:

1. The list of external hosts being under monitoring.

2. The descriptions of metrics for measuring availability and the performance of hosts, system services and business applications running on servers.

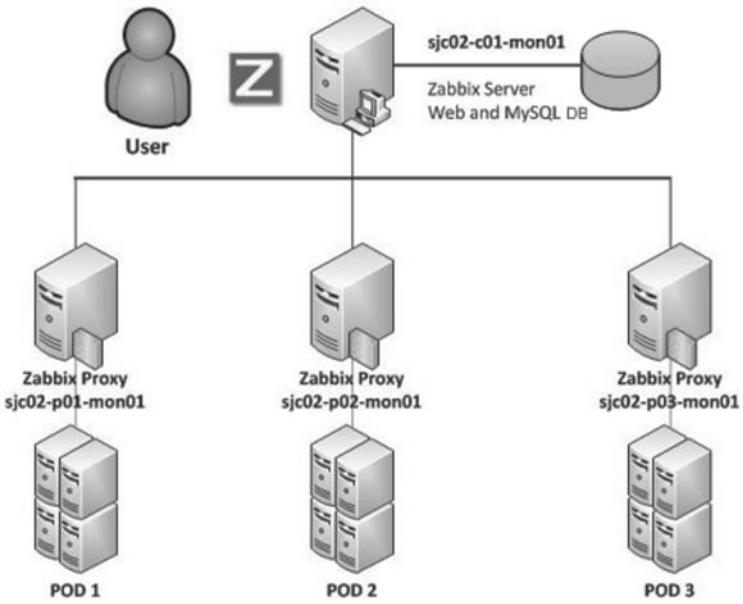

Fig. 3. Zabbix Monitoring System Architecture

Table 2

Example of Zabbix Items for RC Host

Name	Last check	Last value	Change	Interval	History
ICMP Availability (3 Items)					
ICMP ping	1 Dec 2013 12:22:30	1	-	90	180
ICMP packet loss	1 Dec 2013 12:22:34	0 %	-	90	180
ICMP ping sec	1 Dec 2013 12:22:32	0.28 msec	-0.04 msec	90	180
JAVA CMS Memory (5 Items)					
Java CMS Memory heap used	1 Dec 2013 12:22:37	233.5 Mbytes	+5.97 Mbytes	1800	365
Java CMS Perm Gen pool used	1 Dec 2013 12:22:47	145.01 Mbytes	+368 bytes	1800	365
Java CMS Thread count	1 Dec 2013 12:22:49	161	-1	1800	365
Java CMS Memory heap free	1 Dec 2013 12:22:55	293.42 Mbytes	+3.56 Mbytes	1800	365
Java CMS Perm Gen memory free	1 Dec 2013 12:22:58	261.71 MB	-	1800	365
JMX_App_ServiceWeb (6 Items)					
JMX_JEDI (7 Items)					
JMX: JEDI: JDBC pool connection wait time	1 Dec 2013 12:22:56	0	-	60	365
JMX: JEDI: JDBC statement execution time	1 Dec 2013 12:22:56	0 seconds	-	60	365
JMX: JEDI: JDBC statement prepare time	1 Dec 2013 12:22:56	0	-	60	365
JMX: JEDI: JDBC pool leased connections	1 Dec 2013 12:22:56	0	-	60	365
JMX: JEDI: HTTP requests processing time	1 Dec 2013 12:22:56	35	-429	60	365
JMX: JEDI: HTTP requests per sec	1 Dec 2013 12:22:56	0.04	-0.16	60	365
JMX: JEDI: Web processing active thread count	1 Dec 2013 12:22:56	0	-	60	365
- other - (69 Items)					

Table 3

Example of Zabbix Triggers for RC Host

Severity	Name	Expression
Information	POD_SWS_v3:JBoss needs to be restarted	{sjc01-p01-sws01:java.cms.memory.free.max(300)}<{$CMS_PERMGEN_MEM_FREE_CRITICAL}
Critical	Template_Java_CMS_Memory:Free CMS memory is too low	{sjc01-p01-sws01:java.cms.memory.free.max(300)}<{$CMS_MEM_FREE_CRITICAL}
Warning	Template_Java_CMS_Memory:Free CMS memory is too low	{sjc01-p01-sws01:java.cms.memory.free.max(600)}<{$CMS_MEM_FREE_WARNING}

3. The triggers that fire on the events when the predefined thresholds are exceeded. Triggers have different severities such as information, warning and critical.

4. The graphs to analyze performance data and historical trends.

The sample lists of Zabbix configuration items and triggers defined for RC hosts are shown in Tables 2 and 3 respectively.

In addition to standard system metrics, which are predefined in Zabbix agent and if they are installed on a host, custom items and business oriented metrics are implemented. To monitor user's activity and server's resources of Java Enterprise Development Implementation (JEDI) special metrics using Java Management Extensions (JMX) shown in Table 2 are introduced [4–6]. JMX technique provides remote access to internal objects, classes, services and other resources of a Java application that allows measuring actual workload in a server pool. The example of daily statistics in Fig. 4 shows that 20 database connections are permanently established with a host, but they are not actively used in threads (4 as maximum). It means that the capacity of a server pool for this particular RC component can be reduced.

Automatic Alerting on Critical Events

Zabbix triggers fire alarms automatically when a certain metric item exceeds the specified threshold value. Fig. 5 shows the example of Zabbix graph where the degradation of Java memory is observed in many instances, each time triggering an alert in Zabbix system when Java free memory is below critical threshold 5 MB.

Java memory leak is a well-known and it is a

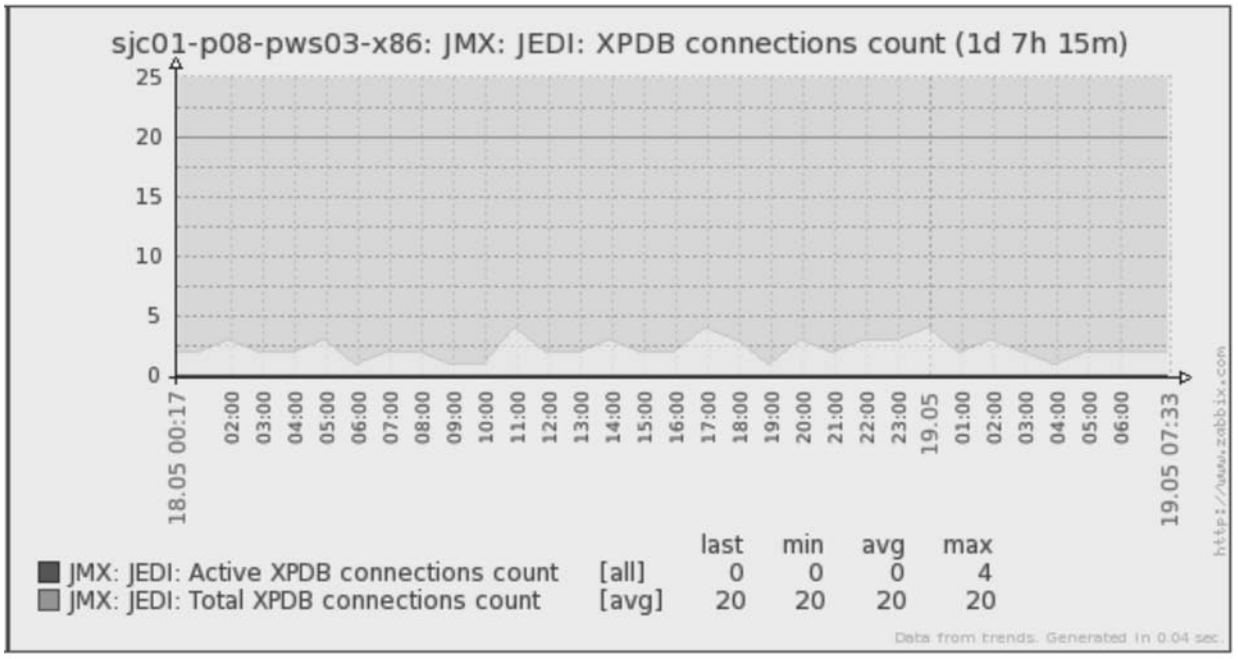

Fig. 4. Example of JMX Metrics for DB Connections

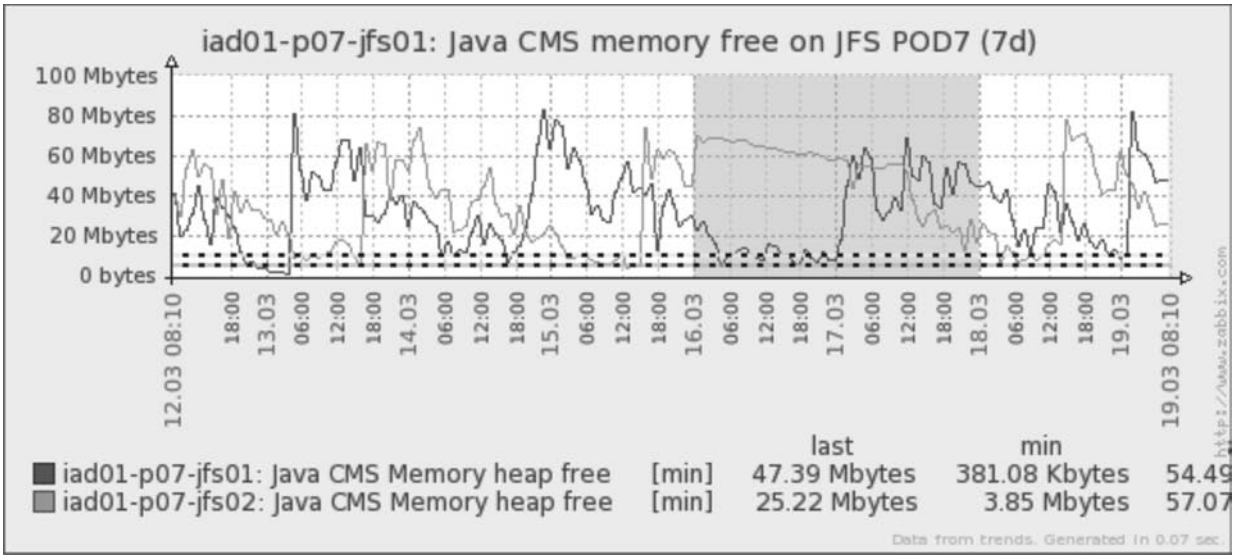

Fig. 5. Sample Graph with Multiple Triggering on Critical Java Free Memory

chronic problem all over the world. The objects, classes and their instances are dynamically created in Java applications that cause virtual memory degradation which depends on user's activity. Traditionally Garbage Collector (GC) is used to partially free up Java memory and reduces the impact of memory leak. The example in Fig. 5 figures out a capacity redundancy in a server pool because critical alarms take place too often while GC is not enough to restore Java memory.

All the alarms, detected in Zabbix monitoring system, are sent via email to an appropriate operation team and are also reflected on the specially designed integrated dashboard.

Integrated Operations Dashboard

An integrated dashboard can be created with a minimal development effort and is really valuable for business services management. The example of RC integrated dashboard or operations monitoring console is shown in Fig. 6. It displays all the events and alerts from different locations having 3500 hosts and over 200K items under Zabbix monitoring with 2 min refresh rate. Additional details for every alert such as JIRA link, time and a person to acknowledge, workaround to fix, etc., can be provided for more efficient troubleshooting.

There are a lot of examples of real world scripts for getting information from external data sources and creating integrated dashboards [7].

Automatic Restoring Anomalies

In addition to monitoring and automatic alerting, Zabbix monitoring system can be configured for automatic actions to resolve the anomalies detected on a host. A good example of such solution is the auto-remediation procedure implemented on JEDI hosts to restart JBoss service when Java virtual free memory is

	Last change	Host	Name	Acknowledged	Acknowledged by
Warning	06 Dec 2013 09:49:43	total.pod6	Free activated numbers in 866 NPA is less than 7%	12 m 15 s	Jayn Layson
Warning	06 Dec 2013 09:49:38	total.pod1	Free activated numbers in 866 NPA is less than 7%	12 m 20 s	Jayn Layson
Critical	05 Dec 2013 09:04:43	sjc01-p09-tas01	TAS has been blacklisted by MORE THAN one ISR during last 10 minutes	13 h 31 m 2 s	Kathlyn Mercado
Critical	05 Dec 2013 08:51:46	sjc01-p09-tas02	TAS has been blacklisted by MORE THAN one ISR during last 10 minutes	13 h 43 m 59 s	Kathlyn Mercado
Critical	05 Dec 2013 08:40:01	sjc01-p09-TAS	At least 2 TAS hosts blacklisted by sjc01-c01-isr05-06 during 10 mins	13 h 55 m 44 s	Kathlyn Mercado

Fig. 6. Example of RC Integrated Operations Dashboard [1]

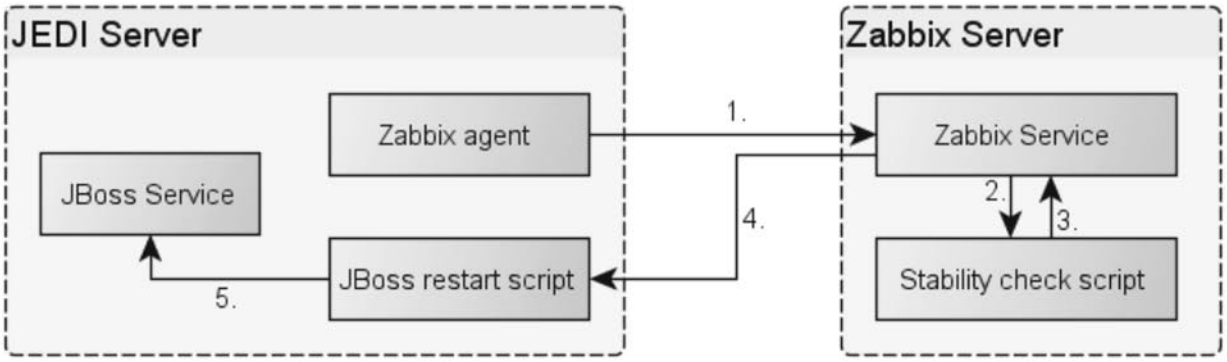

Fig. 7. Auto-remediation Procedure for JEDI Hosts

below critical threshold in spite of GC which are periodically issued on a host. JMX metrics and triggers like those listed in Tables 2 and 3 and described above are used to measure the allocated Java objects and remaining virtual memory resources more accurately than standard system items predefined in Zabbix. Prior to restart Java service, an additional check of the availability of the other VMs in a pool is needed to prevent business service outage. So the logic of auto-remediation procedure is rather complicated and consists of the following steps (Fig. 7):

1. Zabbix agent sends monitoring data from JEDI host to Zabbix server determining that Java free memory is too low and activating the trigger «JBoss which needs to be restarted».

2. Zabbix server executes the stability check script which is specially designed and located locally on Zabbix server to verify the availability of the other JEDI servers in the same pool as a problematic host.

3. If the capacity in a server pool is enough to provide a stable business service, the confirmation is returned back to Zabbix server stating that JEDI host can be safely restarted.

4. Zabbix server initiates the specially designed script which is located on JEDI host to restart JBoss service locally.

5. Finally, after JBoss is restarted, the availability and Java memory resources of JEDI host are monitored by Zabbix as usual.

Scalability and Capacity Analysis

Zabbix enterprise-class system allows monitoring up to 3000 hosts with 200,000 items total from HW and VM, network devices, databases and other cloud environment. High availability of Zabbix system strongly depends on Zabbix DB size, HW performance, the amount of monitoring hosts and items and their polling frequency. Data delay in monitoring, if it happens, may lead to missed service outage that is not acceptable on production environment.

In case of capacity growth and limitations caused by user's workload, the scalability of Zabbix monitoring system can be reached using one way or the combination of the following ways:

1. Reducing the number of monitoring items and/or extending the polling time interval is not a good idea. This might be applicable as a short term workaround if other solutions suggested below can't be implemented.

2. Installing more Zabbix servers, proxies, high performance storages or other HW devices shown in Fig. 3 can help to reduce the impact of monitoring data delay and prevent potential service outages. But this is too expensive solution and doesn't help a lot because the database productivity is the main bottleneck due to multiple read-write transactions executed against the same DB table in parallel.

3. Zabbix splits into 2 or more independent monitoring systems each working with a separate database seems like a permanent workaround. On the other hand, special reports and integrated dashboards should be created half manually to observe consolidated alarms, historical trends and the other data from all

locations on a single monitoring console. The risk of data delay still exists and depends on partitioning between the amount of real-time data and the history.

4. Alternative Zabbix architecture, named Octopus, which is based on MongoDB noSQL open-source document-oriented data warehouse [8], is proposed to reach scalability with a relatively lower cost [9]. The main advantage of the Octopus architecture is that a real-time data is stored in MySQL relational database on a local drive or even in memory due to small size, while all the other history with almost unlimited data retention and the forecasting trends are transferred to MongoDB noSQL data warehouse on network storage.

The main disadvantage of the Octopus distributed architecture is that human resources and web API programming skills are required to consolidate real-time and historical data in a single dashboard and Zabbix reports.

Octopus noSQL approach is currently being under construction at RingCentral Company [1] and has been presented at the annual Zabbix Conference [10].

Recap of Integrated Management Solutions

In multi-host virtualized cloud based distributed production infrastructure like RingCentral [1], the performance data and servers availability can be effectively monitored and managed by Zabbix enterprise-class integrated system.

On JEDI hosts, in particular, independently of HW or VM, JMX metrics [4–6] are introduced for measuring more accurately the actual user's workload and virtual memory resources, more effective monitoring Java applications and automatic alerting in case of the critical degradation.

It is highly recommended to create the integrated operations dashboard to display consolidated events and triggers on a centralized monitoring console, since it provides a lot of benefits for managing a distributed production system with big data traffic with minimal development efforts using SQL, Perl, Ruby or other code.

Besides monitoring and alerting, many critical anomalies concerning system resources in general and Java virtual memory in particular can be detected and automatically fixed if we configure Zabbix and custom scripts for that purpose.

In a cloud computing infrastructure monitoring the database is often a bottleneck which depends on the amount of hosts, metrics, polling time interval, the size of real-time and historical data. So for better DB performance and no reporting data delay, several approaches to distributed Zabbix monitoring architecture are proposed to reach acceptable scalability and capacity.

REFERENCES

1. RingCentral official web site. Available: http://www.ringcentral.com/

2. Zabbix official web site. Available: http://www.zabbix.com/product.php

3. **Hegedus J.C.** Perrobix + Zabbix DB Monitoring. *Zabbix International Conference*, 2013. Available: http://youtu.be/pJCV_ui0orQ

4. Java Management Extensions (JMX) – Best Practices. Oracle Technology Network. Available: http://www.oracle.com/technetwork/java/javase/tech/best-practices-jsp-136021.html

5. JMX Monitoring. *Zabbix 2.0 official documentation*. Available: http://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/jmx_monitoring?s[]=jmx

6. **Mescheryakov S., Shchemelinin D.** Capacity Management of Java-based Business Applications Running on Virtualized Environment. *Proc. of the 39th Internat. Conference for the Performance and Capacity by CMG*. La Jolla, USA, 2013. Available: http://www.cmg.org/conference/

7. **Lipski L.** Integrated Dashboard Design. *Zabbix Internat. Conference*, 2013. Available: http://youtu.be/_gy4qzyZf_o

8. MongoDB official web site. Available: http://www.mongodb.org/

9. **Yulenets L.** When It Comes to Scalability. *Zabbix Internat. Conference, 2013*. Available: http://youtu.be/1Eq-9q16AOU

10. Agenda and presentation abstracts. *Zabbix Internat. Conference 2013*. Riga, 2013. Available: http://www.zabbix.com/conf2013_agenda.php

СПИСОК ЛИТЕРАТУРЫ

1. Официальный Интернет-сайт RingCentral [электронный ресурс]/URL: http://www.ringcentral.com/

2. Официальный интернет-сайт Заббикс [электронный ресурс]/URL: http://www.zabbix.com/product.php

3. **Hegedus, J.C.** Перробикс + Заббикс мониторинг БД [электронный ресурс]//Доклад Междунар. конф. Заббикс. Рига, 2013. URL: http://youtu.be/pJCV_ui0orQ

4. Java Management Extensions (JMX) – Практические рекомендации [электронный ресурс]//Oracle Technology Network. URL: http://www.oracle.com/technetwork/java/javase/tech/best-practices-jsp-136021.html

5. Мониторинг с использованием JMX [электронный ресурс]//Официальная документация Zabbix 2.0. URL: http://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/jmx_monitoring?s[]=jmx

6. **Мещеряков С., Щемелинин Д.** Управление потенциалом бизнеса Java приложений на виртуальном оборудовании [электронный ресурс] // Тез. докл. 39 Междунар. конф. CMG по оценке производительности и потенциала вычислительных систем. Ла Хойя, США, 2013. URL: http://www.cmg.org/conference/

7. **Lipski L.** Проектирование интегрированных отчетов [электронный ресурс] // Доклад Междунар. конф. Заббикс. Рига, 2013. URL: http://youtu.be/_gy4qzyZf_o

8. Официальный интернет-сайт MongoDB [электронный ресурс] / URL: http://www.mongodb.org/

9. **Юленец, Л.** Когда приходит время масштабирования [электронный ресурс] // Доклад Междунар. конф. Заббикс. Рига, 2013. URL: http://youtu.be/1Eq-9q16AOU

10. Перечень докладов и аннотации [электронный ресурс] // Междунар. конф. Заббикс-2013. Рига, 2013. URL: http://www.zabbix.com/conf2013_agenda.php

---

BORTYAKOV, Danil E. *St. Petersburg State Polytechnical University.*
195251, Polytekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: bortyakov@mail.ru

БОРТЯКОВ Данил Евгеньевич – *доцент кафедры транспортных и технологических систем Санкт-Петербургского государственного политехнического университета, кандидат технических наук.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: bortyakov@mail.ru

MESCHERYAKOV, Sergey V. *St. Petersburg State Polytechnical University.*
195251, Polytekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: serg-phd@mail.ru

МЕЩЕРЯКОВ Сергей Владимирович – *профессор кафедры автоматов Санкт-Петербургского государственного политехнического университета, доктор технических наук.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: serg-phd@mail.ru

SHCHEMELININ, Dmitry A. *RingCentral Inc.*
1400 Fashion Island Blvd., San Mateo, CA, USA 94404.
E-mail: dshchmel@gmail.com

ЩЕМЕЛИНИН Дмитрий Александрович – *руководитель департамента развития и эксплуатации облачных IT платформ компании RingCentral, кандидат технических наук.*
1400 Fashion Island Blvd., San Mateo, CA, USA 94404.
E-mail: dshchmel@gmail.com