MyDBaaS: A Framework for Database-as-a-Service Monitoring[*]

David A. Abreu[1], **Flávio R. C. Sousa**[1]
José Antônio F. Macêdo[1], **Francisco J. L. Magalhães**[1]

[1]Department of Computer Science
Federal University of Ceará (UFC)
Pici - Fortaleza - CE - Brazil

{araujodavid, flavio, jose.macedo, franzejr}@lia.ufc.br

Abstract. *Environments of database-as-a-service (DBaaS) promise high performance, high availability, and elastic scalability. These characteristics created the need for innovative monitoring approaches. There are many tools/solutions available for monitoring cloud services. However, there is no standard model for monitoring of databases-as-a-service. In this paper we demonstrate a powerful and easy-to-use framework that provides monitoring for environments of DBaaS, called MyDBaaS Framework. We also present MyDBaaSMonitor application that implements the framework and a demonstration of how to consume the collected metrics by the monitoring environment created.*

1. Introduction

Cloud computing is an extremely successful paradigm of service-oriented computing and has revolutionized the way computing infrastructure is abstracted and used. Scalability, elasticity, pay-per-use pricing, and economies of scale are the major reasons for the successful and widespread adoption of cloud infrastructures. Since the majority of cloud applications are data-driven, database management systems (DBMSs) powering these applications are critical components in the cloud software stack [Elmore et al. 2011].

With the advent of hosted cloud computing and storage, the opportunity to offer a DBMS as an outsourced service is gaining momentum, as witnessed by Amazon's RDS and Microsoft's SQL Azure. Such a DBaaS is attractive for two reasons. First, due to economies of scale, the hardware and energy costs incurred by users are likely to be much lower when they are paying for a share of a service rather than running everything themselves. Second, the costs incurred in a well-designed DBaaS will be proportional to actual usage ("pay-per-use") - this applies to both software licensing and administrative costs [Curino et al. 2011b].

When using DBaaS it is necessary to monitoring the services to verify the Service-Level Agreements (SLA) and costs. According to [Aceto et al. 2012], cloud computing involves many activities for which monitoring is an essential task. The most important ones are: i) Capacity and Resource Planning, ii) Capacity and Resource Management, iii) Data Center Management, iv) SLA Management, v) Billing, vi) Troubleshooting and vii) Security Management.

[*]A screencast of the tool is available at http://sbbd2013.cin.ufpe.br/screencasts.

There is a large number of works proposing solutions to monitor the cloud [Aceto et al. 2012] [NewRelic 2013]. Nevertheless, these works focus on resources on low-level computing resources (e.g., CPU speed, CPU utilization, disk speed). Furthermore, these works do not use specific metrics for monitoring DBMSs and they are not extensible. According to our research, there are no solutions that address cloud database monitoring, since previous works have focused only on some of its aspects.

In order to solve this problem, this paper proposes MyDBaaS[1], a Java-based framework that provides a comprehensive programming and configuration model for developing monitoring strategies for environments of DBaaS. The MyDBaaS Framework offers set of hook points like classes and services already implemented that make its use easy and transparent. It also comprises hotspots which make the structure flexible and extendable as required, as well as a set of metrics already defined. According to the known literature, this is the first framework that exploits the monitoring for environments of DBaaS, representing a significant contribution to the area. The paper also presents MyDBaaSMonitor, a web application implemented with MyDBaaS Framework and a demonstration of the use of MyDBaaS API.

This paper is organized as follows. Section 2 explains the MyDBaaS Framework. Section 3 illustrates the applications that we will demonstrate. Section 4 details related work and, finally, Section 5 presents the conclusions.

2. MyDBaaS Framework

The framework focuses on the creation of the monitoring of the resources allocated for the operation of DBaaS environments. Monitoring can be performed in a scenario of up to four levels: physical infrastructure (hosts), virtual machines (VM), DBMSs and database instances. The MyDBaaS enables the monitoring of metrics at all levels according to the need. It has a complete structure that provides from creating a metric until storage in the knowledge base. The MyDBaaS provides a pattern programming through nomenclatures, hook points and hotspots that make its use simple and intuitive. The communication between the server and monitoring agents is completely transparent to the user, as well as the creation and maintenance of knowledge base. The knowledge base is important because it creates a profile of the use of all the metrics monitored according to the time.

2.1. Architecture

The MyDBaaS Framework architecture is divided into four modules: *Core*, *Agent*, *API* and *Common*. Each module is composed by a set of components, as illustrated in Figure 1. The *Core* is the central module responsible for coordinating the resources of the environment, coordinate the monitoring agents and maintain the knowledge base. The **Controller** component provides the structure for the management of requests sent by the API and monitoring agents. The **MonitoringCoordinator** is the component whose function is to access the resources and set up the monitoring agents automatically. **Repository** is a major component, providing transparent and automatic features for creating and maintaining the knowledge base and for the storage of collected metrics and resources. The **Knowledge Base** has the function of storing the information generated by the framework creating a behavioral profile of resources as a function of time. The *Agent* is the module present in the resources of the environment and is responsible for to monitor,

[1]http://www.mydbaasmonitor.com

collect and interact with the *Core* module, and consists of four components. The **Monitor** provides functions to automatically load the agent settings, as well as methods that give the agent the ability to self-configure. The **Collector** is the component that provides the infrastructure to collect metrics. It enables the metrics to be collected independently and on different cycles. It also provides dynamic and automatic methods for sending the metrics to the *Core* module. The **MetricManager** is responsible for loading the settings of the metrics that the agent will use. The **ConnectionManager** is responsible to provide features of access to databases and DBMS automatically when a collector needs to measure a metric. The *API* is the module that allows external access to the monitoring environment. It provides a set of routines and patterns for the use of *Core* services. The **Client** is the component responsible for communication between API and Core, supplying the features of connection management to the other components. The **PoolManager** provides access to registered resources in the monitoring environment, allowing retrieve, manage and add new resources. **QueryMetric** is the main component of this module. It enables the consumption of collected metrics about resources through diverse methods. The **AccessCoordinator** allows the management of monitoring agents. The last module is the *Common*, which it has two components. **ResourceBucket** and **MetricBucket** are responsible for providing the pattern for creating the representation of new resources and metrics respectively.

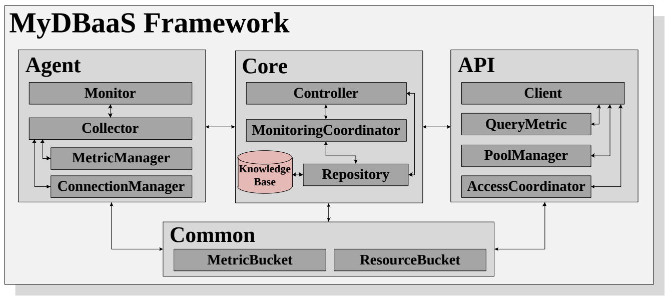


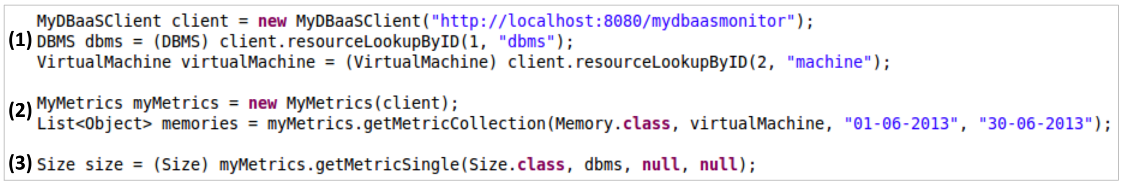
Figure 1. Overview of the MyDBaaS Framework architecture.

MyDBaaS was fully implemented in Java based on VRaptor Framework and Apache HttpComponents. Agent-side and API-side actions are processed by VRaptor on the Core-side, and the communication is done via HTTP requests. MyDBaaS's architecture is designed to easily include new metrics, as well as it was developed to be independent of the infrastructure and DBMS. The HTTP protocol was chosen because it makes easier the scalability of the architecture. The knowledge base stores a large volume of monitored data into a relational DBMS, but can be easily extended to a NoSQL (e.g., MongoDB[2] or Cassandra[3]). The framework already has some metrics implemented in the four levels. In the level of VMs: CPU, memory, network, disk, partitions and information about the VM settings. In the level of hosts: HostInfo that allows to collect the physical configuration information and the hypervisor used, HostDomains that allows to collect the amount of active and inactive VMs hosted, DomainStatus that allows to collect the use of CPU and memory of the VMs on the host side and all of the previous level. In the level of DBMS: CPU and memory used, active connections and base size. In the level of database instances are the same of the DBMS except CPU and memory.

[2]http://www.mongodb.org
[3]http://cassandra.apache.org

2.2. API

The MyDBaaS application programming interface (API) enables developers to build applications that interact directly with the monitoring environment created by MyDBaaS Framework. With these applications, users can more efficiently and creatively manage their resources and services. This API can be used to develop additional tools and applications to help the manage the service level objectives (SLOs) of the SLAs, integrate the collected data with systems of data analysis to help making decisions or applications that generate events and alerts about monitored resources. The MyDBaaS API gives access to all of the metrics data that are being monitored in real time, as well as the historical of all that has been collected. It also allows to access and manage the resources registered in the environment.



```
    MyDBaaSClient client = new MyDBaaSClient("http://localhost:8080/mydbaasmonitor");
(1) DBMS dbms = (DBMS) client.resourceLookupByID(1, "dbms");
    VirtualMachine virtualMachine = (VirtualMachine) client.resourceLookupByID(2, "machine");

(2) MyMetrics myMetrics = new MyMetrics(client);
    List<Object> memories = myMetrics.getMetricCollection(Memory.class, virtualMachine, "01-06-2013", "30-06-2013");

(3) Size size = (Size) myMetrics.getMetricSingle(Size.class, dbms, null, null);
```

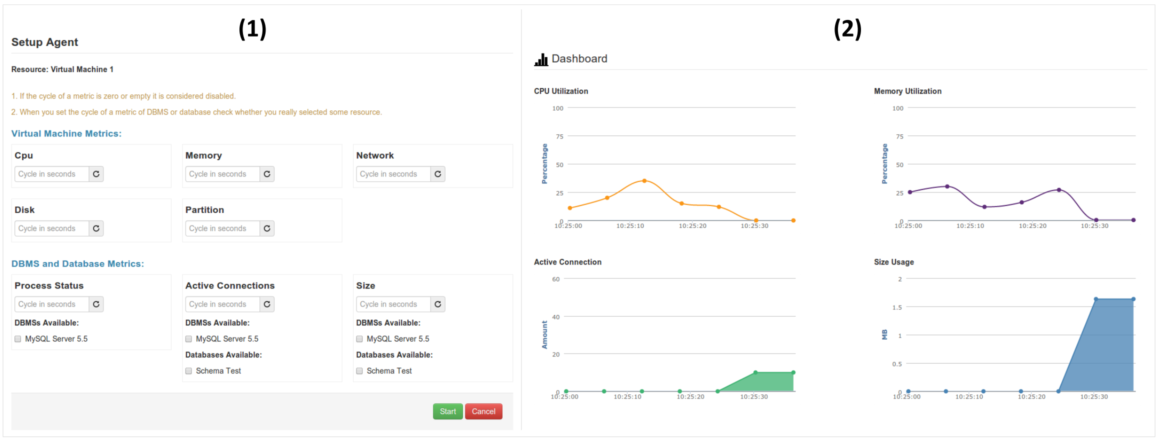
Figure 2. Example of using MyDBaaS API.

Figure 2 illustrates some available functionalities by the API. The example (1) shows the creation of the connection to the server that contains the *Core* module. Also presents how it can be done to recover the record of different resources. The (2) demonstrates one way to consume the historical of the metric of Memory in a given time range about a particular virtual machine. The last example (3) presents how to retrieve the latest collection of the metric of Size about a particular DBMS.

3. Demonstration

To showcase the benefits of the described framework we propose two demonstrations of the main features provided by the MyDBaaS Framework. As a running and implementing example we show MyDBaaSMonitor, a web application for managing the monitoring of a given scenario. Finally, we demonstrate one way of using the MyDBaaS API to consume the metrics monitored in this application.

MyDBaaSMonitor is a web application developed using MyDBaaS Framework as backend and Twitter Bootstrap with Highcharts to design the graphical user interface. This application enables the monitoring of the following scenario: a cloud managed by OpenNebula, using the KVM as virtualization solution. This cloud is composed of a set of servers hosting multiple virtual machines, and each one can have multiple DBMSs. And each DBMS can have multiple instances of databases. Finally, this cloud can contain several environments of DBaaS. The application has a web interface for the user to manage the environments, allowing to configure and register the resources composing each DBaaS. Once the resources have been registered, the user can configure monitoring for each resource of the environments as needed. The user chooses which metrics will be collected, as well as the monitoring cycle for each metric. Finally, the application accesses the resource automatically and sends the monitoring agent and its configuration file, installs and starts the agent automatically and transparently. The user can track the real-time monitoring through the graphics that are displayed when accessing the resource record. The user can also stop monitoring or reconfigure the agent as needed.


Setup Agent

Resource: Virtual Machine 1

1. If the cycle of a metric is zero or empty it is considered disabled.
2. When you set the cycle of a metric of DBMS or database check whether you really selected some resource.

Virtual Machine Metrics:

Cpu: Cycle in seconds
Memory: Cycle in seconds
Network: Cycle in seconds
Disk: Cycle in seconds
Partition: Cycle in seconds

DBMS and Database Metrics:

Process Status: Cycle in seconds
DBMSs Available: ☐ MySQL Server 5.5

Active Connections: Cycle in seconds
DBMSs Available: ☐ MySQL Server 5.5
Databases Available: ☐ Schema Test

Size: Cycle in seconds
DBMSs Available: ☐ MySQL Server 5.5
Databases Available: ☐ Schema Test

Figure 3. Overview of two application screens.

Figure 3 shows two screens available in the application. The screen (1) is where the user sets the monitoring that will run on a given virtual machine. If there are DBMSs and database instances within the VM it is also possible to configure the monitoring. On the screen (2) is where the user can view the metrics that are being monitored on a particular DBMS, and manage their information.

How to use MyDBaaS API - during this demonstration we will present how the API can be used to build applications that need to use monitoring. In a first scenario we show how to consume the metrics in different ways and at all levels of resources. To this end we employ our local cloud at Federal University of Ceará, which will have a running instance of the MyDBaaSMonitor application. And we will use the metrics collected in this scenario. In a second scenario we show an application to create alerts based on monitoring performed in the first scenario.

4. Related Work

In the last years, several systems have been developed in order to monitor cloud services. However, studies in the literature point to works and products that focus on specific points of monitoring. In [Curino et al. 2011a] the authors present Kairos, a system for database consolidation in VM-based environments. It uses monitoring on OS and DBMS about CPU, memory, disk, buffer pool utilization and log flushes. Using these metrics to estimate resource needs during the consolidation process. [Klems et al. 2012] shows a run-time quality measurement framework for cloud database service systems. This framework uses a monitoring on quality metrics for this service (eg, average throughput, average and upper-percentile response time, elastic scalability, and others). These metrics are composed of smaller granularity metrics. The authors of [Zhao et al. 2012] present a solution for database replication in virtualized environment of cloud. They also needed to monitor metrics on two levels, virtual machine (CPU, disk and network) and DBMS (throughput). Therefore as previous works showed, monitoring is essential for strategies used to provide an environment of DBaaS. Unlike these works which do not have the monitoring as the main problem, our work focuses on creating monitoring solutions according to the need for this type of environment. Our proposed framework can be used by the previous works, supplying the need for monitoring of them at all levels. Some existing solutions in the market, such as [NewRelic 2013] and [Amazon 2013], provide monitoring services ready for the end user, different from our work that enables the user to create their own monitoring environment/service in several layers. The focus of our work was to design a framework, so we opted not to extend the current solutions. Based on our review of the literature, there is none framework for monitoring database-as-a-service that enables the creation of a range of metrics ranging from physical infrastructure until the instances of databases. The MyDBaaS Framework differs by the fact to allow extent and be flexible to adapt to the real needs of the user's monitoring. Importantly, our solution is fully open source.

5. Conclusion and Future Works

This paper presented the MyDBaaS Framework that enables the creation of monitoring solutions for environments of DBaaS. Based on an architecture divided into modules, the framework can be extended by adding new features and metrics in a practical way. The paper also presented the MyDBaaSMonitor, a web application developed using the framework, having presented interesting initial results about its use. More metrics for databases (e.g., throughput, response time, latency, buffer pool size, cache hit rate, SQL commands, transactions committed and rolled back, freeable memory, network traffic, number of reads and writes, number of bytes read and written, and others) are being developed and will be added to the framework. As future work two modules will be created. The **Driver** that allows to create and collect metrics at the level of workloads sent to the DBMSs and database instances, and **ExporterMetric** that allows the extraction of metrics in CSV file in a personalized way.

References

Aceto, G., Botta, A., de Donato, W., and Pescape, A. (2012). Cloud Monitoring: Definitions, Issues and Future directions. In *CLOUDNET '12*, pages 63–67.

Amazon (2013). Amazon Cloudwatch. `http://aws.amazon.com/cloudwatch`.

Curino, C., Jones, E. P., Madden, S., and Balakrishnan, H. (2011a). Workload-aware database monitoring and consolidation. SIGMOD '11, pages 313–324.

Curino, C., Jones, E. P. C., Popa, R. A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011b). Relational Cloud: a Database Service for the Cloud. CIDR '11, pages 235–240.

Elmore, A. J., Das, S., Agrawal, D., and El Abbadi, A. (2011). Zephyr: live migration in shared nothing databases for elastic cloud platforms. In *SIGMOD '11*, pages 301–312.

Klems, M., Bermbach, D., and Weinert, R. (2012). A Runtime Quality Measurement Framework for Cloud Database Service Systems. QUATIC '11, pages 38–46. IEEE Computer Society.

NewRelic (2013). New Relic Application. `http://newrelic.com`.

Zhao, L., Sakr, S., Fekete, K., Wada, H., and Liu, A. (2012). Application-Managed Database Replication on Virtualized Cloud Environments. ICDEW '12, pages 127–134.