Toward autonomic service operation for cloud applications: the AESOP project

Raul Barbosa[1], João Tomás[1], André Bento[1], João Soares[2], Luís Ribeiro[2], António Ferreira[2], Rita Carreira[3], Filipe Araújo[1]

[1]University of Coimbra, CISUC, Department of Informatics Engineering
[2]Fiercely, Rua Pedro Nunes, C-2.28, Coimbra, Portugal
[3]Virtual Power Solutions, Coimbra, Portugal

Introduction

A software solution that is deployed to the cloud must guarantee availability and performance standards that are crucial for mission- and business-critical applications.
Although there are numerous approaches for monitoring technical parameters, it is not possible to infer the quality of services in operation and whether the service-level objectives (SLOs) are fulfilled.As a result, there is a gap between the availability and performance needs of software solutions and what you can see infrastructurally with today's tools.
To overcome this gap, the AESOP project (Autonomic Service Operation) aims to develop a platform that allows the specification of service parameters, per software application, and the creation of an adaptive feedback loop that acts on these applications.
With this platform we will be prepared to answer essential questions such as:

- The parameters imposed by the business are being guaranteed by the software in operation?
- Has a certain new deployment increased or decreased my resource consumption?
- What is the root cause localization of a given failure and what autonomic actuations have been attempted?

This feedback loop takes advantage of virtualized infrastructure approaches, based on infrastructure as code principles, expanded and strengthened to address the applicational detail and not just the infrastructure. With this, the characterization of the availability and business performance parameters to be monitored becomes explicit, both in the development phase and in the operational phase.

Low MTTR is Achievable in Microservice Applications

The collected results consist of MTTD (the time required to detect and diagnose a failure in the managed application) and MTTR (the time required to actuate in order to recover the application). It should be highlighted that some failures may not be fully recoverable, requiring operator intervention. Figure 1 depicts these metrics.

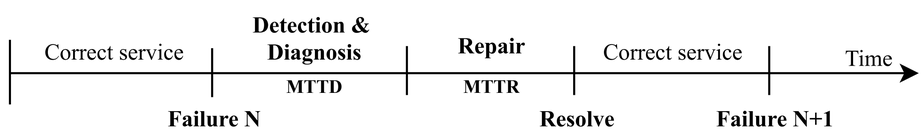

Figure 1: Availability components.

The experimental testbed is a virtual private cloud deployed on OpenStack with a total of 5 virtual machines. We deployed Kubernetes on 4 of them with a single master and four worker nodes. The remaining machine hosts the ElasticSearch database, where the Jaeger Ingester stores spans. Each machine in the Kubernetes cluster has 8 vCPUs, 16GB of RAM, 100GB of storage and runs Debian 10 operating system, with the exception of the database, which has 4 vCPUs and 8GB of RAM.

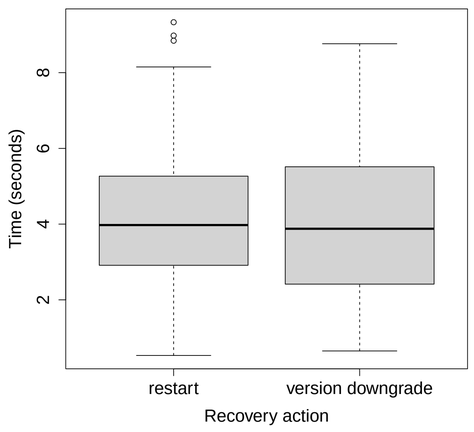

Figure 2: Mean-time to detect and diagnose.

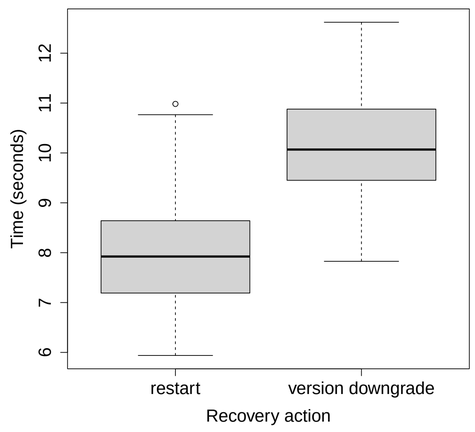

Figure 3: Mean-time to recover.

Figures 2 and 3 show the experimental results. The mean-time to diagnose is on average 4 seconds, for both recovery actions. Restart takes an average of 8 seconds and version downgrade takes an average of 10 seconds. Low MTTR can therefore be achieved.

Autonomic Service Operations Architecture

The proposed architecture aims to detect, diagnose and repair failures (self-healing) in microservice applications. **The ability to autonomically recover from failures improves availability by reducing the mean-time to recover from outages.** The architecture is inspired by IBM's MAPE-K feedback loop and uses a pub-sub paradigm.

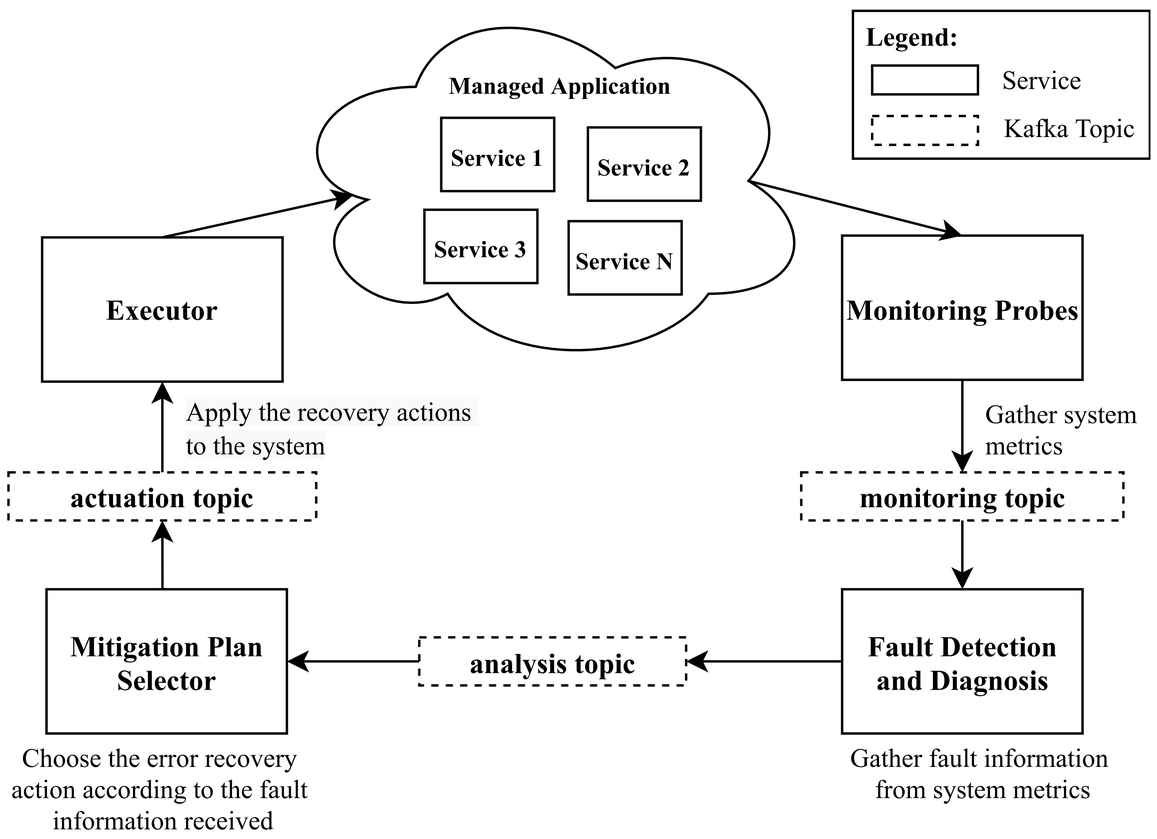

Figure 4: Autonomic architecture overview.

Figure 4 provides an overview of the proposed architecture, consisting of the four components in the MAPE-K loop, using Apache Kafka as their middleware to stream messages among each other, allowing the exchange of high volume data in a fast and reliable manner.

Challenges of Autonomic Service Operation

Monitoring is conceptually simple: sensors or probes measure/collect information about the system and feed it into the control loop. However, there are several open challenges:

- *Unavailability measurement is indirect and inaccurate.* Even the most advanced tools presently used focus on counting HTTP errors.
- *Measuring recovery times is disregarded in favor of best-effort.* Once a failure or anomaly occurs and recovery actions are taken, it is necessary to monitor the time to resume operations.
- *There's a gap between low-level metrics and high-level SLOs.* There aren't off-the-shelf sensors available for software, so applying control theory becomes difficult.
- *There's a lack of architectural knowledge.* In spite of decades of software architecture research, the scope of its application is very narrow.

Analysis is the second stage of the MAPE loop, aiming to identify discrepancies between the *current* state and the *desired* state.

- *There's a lack of clear contracts among services.* However, unless one uses a design by contract approach, it is unknown if the fault was on the caller side or on the callee side.
- *Reliability of diagnosis influences the correctness of actuation.* Unless root-cause localization is reliable, actuation shall take place on the basis of wrong information.
- *The actuation risk varies according to the foreseen downtime cost.* Being roughly correct is better than precisely wrong and not all components are equally valuable.

Planning is the stage that determines the actuations that must take place to lead the system back to the desired state once it has deviated from it.

- *There's a need to dynamically adjust the risk of actuation.* Not all downtime has the same cost.
- *It is necessary to predict the downtime generated by actuation.* Repairing and actuating over a system in operation may, inadvertently, lead to profit loss and/or incurring costs.
- *Cloud applications should be designed for recovery.* Components should be restartable and relocatable, with the application state being transactionally stored in a database.

Institutions

CENTRO 2020 · PORTUGAL 2020 · UNIÃO EUROPEIA Fundos Europeus Estruturais e de Investimento · Fiercely Developing Software · 1290 Universidade de Coimbra · VPS Virtual Power Solutions