Master's Degree in Informatics Engineering

Thesis - Intermediate Report

Observing and Controlling Performance in Microservices

Author:
André Pascoal Bento

Supervisor:
Prof. Filipe João Boavista Mendonça Machado Araújo

Co-Supervisor:
Prof. Jorge Cardoso

UNIVERSIDADE D COIMBRA

January 2018

Abstract

In a world of increasingly decoupled microservices ...

Keywords

Microservices, Cloud Computing, Observing, Monitoring.

This page is intentionally left blank.

Resumo

Hoje em dia encontramos-nos num mundo em que a crescente evolução tecnológica, exige cada vez mais dos sistemas computacionais e das pessoas que os desenvolvem e mantêm. Com este crescimento, certas características como a complexidade e a distribuição dos sistemas aumentam a passos largos, de modo a que se torna bastante difícil de os gerir e de perceber o seu funcionamento em geral. É neste problema que este trabalho visa prestar soluções. As soluções apresentadas neste documento, têm como principal objetivo permitir ??? em sistemas com as referidas características.

Palavras-Chave

Micro-serviços, Computação na nuvem, Observação, Monitorização.

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

API Application Programming Interface. 9

CPU Central Processing Unit. 7, 8

DEI Department of Informatics Engineering. 1

GDB Graph Database. 6, 7

HTTP Hypertext Transfer Protocol. 5

QA Quality Attribute. 13, 14

RPC Remote Procedure Call. 5

TSDB Time Series Database. 6, 8

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

This document represents the *Master Thesis* in *Informatics Engineering*, of the student *André Pascoal Bento* during the school year of 2018/2019, taking place in the *Department of Informatics Engineering (DEI)* of the *University of Coimbra*.

1.1 Context

In today's world, IT(Information Technology) and DevOps(Development and Operations) teams have lots of problems when is needed to identify the root causes that creates issues in their cloud or distributed systems. Moreover, debugging this type of systems has not been an easy task, as it involves a hard and tedious work of diving through logs and system input and output registries and, in the most cases, it reveals like a big "find a needle in the barn" problem. Taking this into consideration, the work presented in this thesis, aims to perform an investigation around this kind of needs and present some solutions to the presented problem.

1.2 Motivation

- Explain and present the motivation about the theme behind this thesis.

1.3 Goals

- Present the main goals of this project.

1.4 Work Plan

In this section will be presented the work plan as well as the work performed, including the foreseen and real work plans for the whole year of work.

The time spent in each semester of the year by week are sixteen hours for the first semester, and forty hours for the second one. In the end, it was spent a total of **TODO:**

XXXX hours for the first semester, starting in 11.09.2018 and ending in XX.XX.XXXX.

In the beginning, there was a work plan for the two semesters presented in the the thesis proposition. For record it is presented in the figures 1.1 and 1.2.

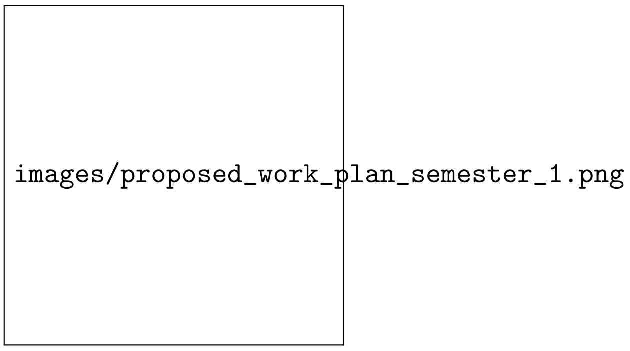

Figure 1.1: Proposed work plan for the first semester.

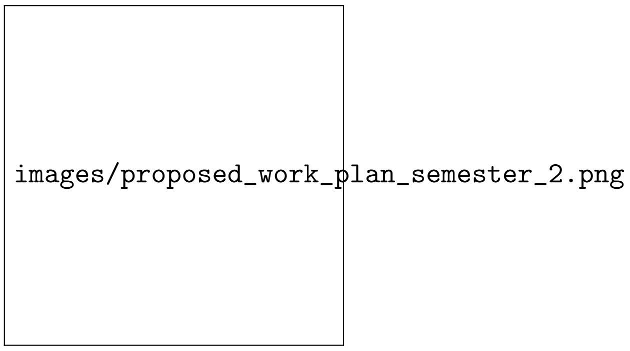

Figure 1.2: Proposed work plan for the second semester.

The "foreseen" work plan for the first semester has suffered some changes, when comparing it to the real work plan. For effects of analysis, the real work plan for the first semester is presented in the figure 1.3.

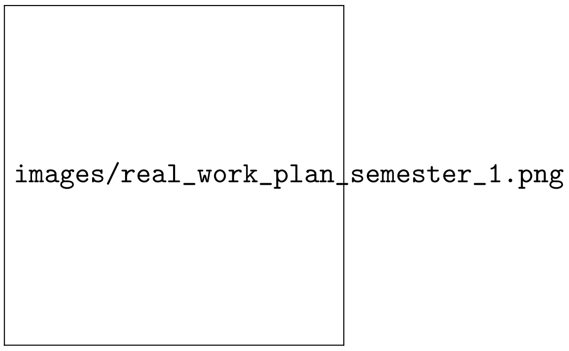

Figure 1.3: Real work plan for the first semester.

TODO: As we can see ...

All the figures have been created by an open-source tool called GanttProject[14] that produce Gantt charts, a kind of diagram used to illustrate the progress of the different stages of a project.

1.5 Document Structure

- Present the document structure of this report.
- Refer the next sections and explain what they contains.

Chapter 2

State of the Art

In the following chapter, we will be discussing the core concepts regarding the project subject and the most modern techniques and technology that's available for the purpose today. All the information that will be presented was the result of a work of investigation through published articles, knowledge exchange and web searching.

The main purpose of the section 2.1 is to introduce and provide a brief explanation about the core concepts. In the second section 2.2, all the important technologies are analysed and discussed using tables, diagrams and clear text.

2.1 Concepts

The following concepts represents the baseline to understand the work related to this project.

2.1.1 Microservices

Microservices is "an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities"[12].

2.1.2 Observing

Observing is "to be or become aware of, especially through careful and directed attention; to notice"[15].

The presented definition represents the meaning of the word Observing and it is reflected exactly as it is in the project context. For example, observe the interaction between some microservices to notice a fault.

When we want to understand the working and behaviour of a system, we need to watch it very closely and pay special attention to all details and data it provides. This kind of details and data may be in multiple structured text formats, and it can contain lots of information regarding the interaction between microservices and the corresponding access to them. Therefore, we may work with this information as a starting point to perceive the characteristics of the system and build a tool that is able to be aware of it.

2.1.3 Controlling Performance

- TODO: Explain what is Controlling and Performance with a definition for each, and how they correlate.

2.1.4 Traces and Spans

First things first, we can think in a trace as a group of spans. A trace is a representation of a data/execution path through the system and a span represents the logical unit of work in the system. The span has an operation name, the start time of the operation, and its duration. An example of a span can be an Hypertext Transfer Protocol (HTTP) call or an Remote Procedure Call (RPC) call. With a couple of spans, we might be able to model a graph of a portion of the system, because they represents causal relationships in the system.

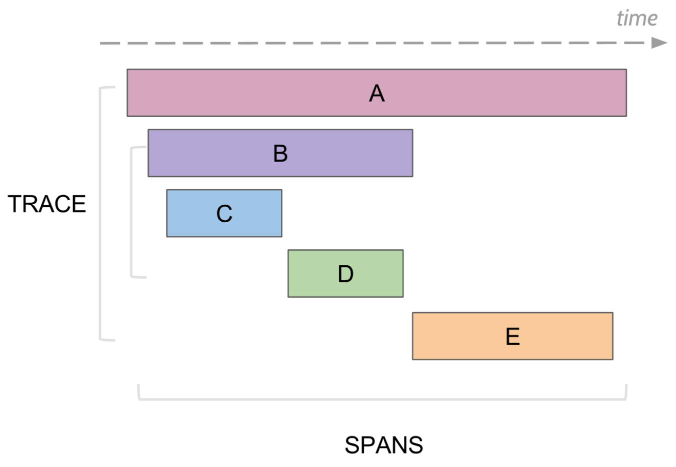

Figure 2.1: Traces and spans disposition over time.

As you can see in the figure 2.1, the spans spread over time, overlapping each other, since nothing prevents the occurrence of multiple calls in very close times.

This type of data is extracted and can be obtained, as trace files or data streams, from technologies like Kubernetes[4], OpenStack[10], and other cloud or distributed management system technologies that implements some kind of system or code instrumentation using, for example, OpenTracing[11] or OpenCensus[6].

In the end and as explained before, traces and spans contains some vital system details as they are the result of instrumentation of a part or the whole system and therefore, this kind of data can be used as a starting point resource information to analyse the system.

2.1.5 Graphs

As it was briefly explained before, we might be able to model a graph of the system using a couple of spans. Normally, in discrete mathematics and more specifically in graph theory, a graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related"[17].

Taking the very common sense of the term, a graph is an ordered pair G = (V, E), where G is the graph itself, V are the vertices/nodes and E are the edges.

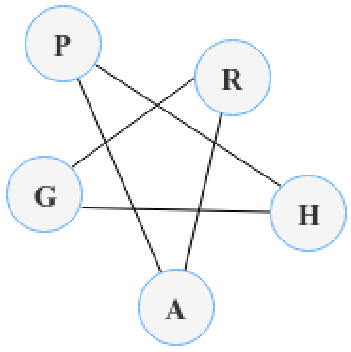

Figure 2.2: Graph visual representation.

The figure 2.2 gives us, a simple visual representation, of what a graph really is for a more clear understanding. There we can see a graph composed by a total of five nodes that contains some labels in it and, in this case, five relationships between them.

This graph is the representation of the following information:

V = {'G', 'R', 'A', 'P', 'H'}

E = {{'G', 'R'}, {'R', 'A'}, {'A', 'P'}, {'P', 'H'}, {'H', 'G'}}

There are multiple types of graphs. In this term they can be undirected, where the set of edges don't have any orientation between a pair of nodes like in this example, or be directional, where the set of edges have one and only one orientation between a pair of nodes, or be a multigraph, where in multiple edges are more than one connection between a pair of node that represents the same relationship, and so forth.

Graphs can have many use cases, has they can model the representation of a lot of real life practical problems in the fields like physics, biology, social and information systems and for the purpose of this thesis, they are considered first class citizens.

2.1.6 Graph Database

A Graph Database (GDB) is "a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data"[18].

The composition of a GDB is based on the mathematics graph theory, and therefore this databases uses three main components called nodes, edges, and properties. This main components are defined and explained in the following list:

- Node:
- Edge:
- Property:

2.1.7 Time Series Database

- TODO: Explain what is a Time Series Database (TSDB) with a definition and how they work.

2.2 Technologies

In this section are presented the technologies and tools that were researched, as well as the corresponding discussion considering the main objectives for this project. With the concepts presented in the section 2.1 in mind, the research were focused in a group of main topics regarding the problem we have in hands. The main topics were graph manipulation tools, graph database tools and time-series database tools.

2.2.1 Graph Manipulation and Processing Tools

Considering that we have data to be processed and manipulated, we have to be sure that we can handle it for analysis. For this purpose, and knowing that the data is a representation and an abstraction of a graph, we needed to study the frameworks available this task. The table 2.1 presents the main technologies available at the time for graph manipulation and processing.

With the information presented in the previous table, we can have a notion that this three frameworks don't work and perform in the same level in many ways.

One thing to consider when comparing them is the scalability and performance that each can provide, for instance, in this component the first one, Apache Giraph is the winner since it is implemented with the distributed systems paradigm in mind and can scale to multiple-machines to get the job done in no time, considering high amounts of data. In other way, we have the third framework, called NetworkX, that is different from the previous as it works in a single-machine and doesn't have the ability to scale to multiple-machines. This can be a very problematic feature if we are dealing with very high amounts of data and we have to process it in short amounts of time. The last framework, called Ligra, works in a single-machine environment like the previous one, but it can scale vertically as it has the benefit of use and exploit multi-core CPU's.

The second, and also most important thing to consider, is the support and quantity of implemented graph algorithms in the framework, and in this field, the tables turned, and the NetworkX has a lot of advantage as it have lots of implemented graph algorithms defined and studied in graph theory. The remaining frameworks don't have very support either because they don't have it documented or because the implementation and architecture that where considered don't allow it.

For a more clear insight of the position of the presented technologies in the previous table an explanation, we have the 2.3.

With the presented figure, our perception of what we might choose when considering this tools is more clear, but we have always some trade offs we cannot avoid. The best approach, and if it's possible, is to consider the usage of an hybrid environment where Giraph and NetworkX coexist one with another, as one fills the gaps of the other, but always taking into consideration that a bottleneck might occur between them [7].

2.2.2 Graph Database Tools

Manipulating and process graph data is not enough, we need to store this data somewhere, and to do this we need a GDB. The results of the research for the best graph databases tools available are presented in the table 2.2.

Table 2.1: Graph manipulation and processing tools comparison.

Name	Apache Giraph [1]	Ligra [13]	NetworkX [9]
Description	It's an iterative graph processing system built for high scalability. For example, it is currently used at Facebook to analyse the social graph formed by users and their connections.	A library collection for graph creation and manipulation, and for analysing networks.	A Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
Licence[19]	Free Apache 2	MIT	BSD - New License
Supported languages	Java, Scala	C, C++.	Python.
Pros	Distributed and very scalable. Excellent performance (Can process one trillion edges using 200 modest machines in 4 minutes).	Can handle very large graphs. Exploit large memory and multi-core CPU's (Vertically scalable).	Good support and very easy to install with Python. Lots of graph algorithms already implemented and tested. Mature project.
Cons	Uses the "Think-Like-a-Vertex" programming model that often forces into using sub-optimal algorithms and is quite limited and sacrifices performance for scaling out. Can't perform many complex graph analysis tasks because it primarily supports Bulk synchronous parallel.	Lack of documentation and therefore, very hard to use. Don't have many usage in the community.	Not scalable (single-machine). High learning curve due to the maturity of the project. Begins to slow down when there's a high amount of data (400.000 plus nodes).

As we can notice by the data provided by the presented table, the state of the art about graph databases is not very good. The offers are very limited and all of them lack something when we start to see them in detail. The interest in this databases is increasing as they can solve lots of problems in nowadays.

2.2.3 Time-Series Database Tools

- TODO: Explain what is a TSDB with a definition and how they work.

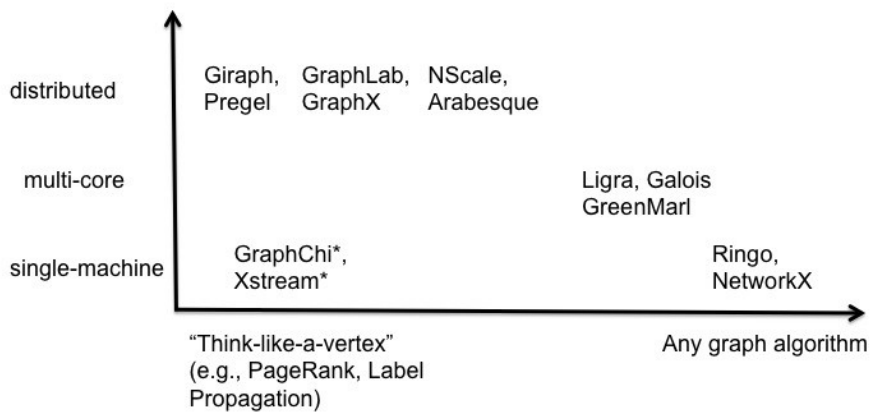

Figure 2.3: Graph manipulation and performance tools regarding scalability and graph supported algorithms[5].

Table 2.2: Graph databases comparison.

Name	ArangoDB [2]	Facebook TAO [8]	Neo4J [3]
Description	It's a NoSQL database developed by ArangoDB Inc. that uses a proper query language to access the database.	TAO, "The Associations and Objects", is a proprietary database, developed by Facebook, that stores all the data related to the users in the social network.	It's the most popular open source graph database. Has been developed by Neo4J Inc. and is completely open to the community.
Licence	Free Apache 2	Proprietary	GPLv3 Community Edition
Supported languages	.NET, C++, Go, Java, JavaScript, Node.js, PHP, Python, Ruby, Scala	—	.NET, Java, JavaScript, Python, Ruby, Scala
Pros	Multi data-type support (key/value, documents, graphs). Allows the combination of different data access patterns in a single query. Supports cluster deployment.	Supports ACID(Atomicity, Consistency, Isolation, Durability)[16]. High-availability. Has a visual node-link graph explorer. REST API interface. Most popular open source graph database.	Very fast(=100ms latency). Accepts millions of calls per second. Distributed.
Cons	Needed to learn a new query language called AQL(Arango Query Language). High leaning curve. Has paid version with high price tag.	Can't be distributed (It needs to be vertically scaled).	Not accessible to use.

Chapter 3

Research Objectives and Approach

- TODO: Consider to remove this chapter.
- If we keep this chapter, explain here how the work were performed (approach: meetings, week work, prototyping, usage and test of the different tools, etc) and expose the main objectives of this whole thesis (clash with other topics explained before :(?)

This page is intentionally left blank.

Chapter 4

Solution

In this chapter we will be presenting and discussing the solution to be implemented regarding the research objectives of this work. To present the solution and explain it, we will cover some aspects and topics that are considered when defining a software based solution. This topics are the well known functional requirements 4.1, the quality attributes or non-functional requirements 4.2, the business restrictions 4.3, the technical restrictions 4.4 and finally, the architecture 4.5 that is produced based on all the previous topics.

4.1 Functional Requirements

The functional requirements are in software and systems engineering, by definition, a declaration of the intended function of a system and its components. For the purpose of this specification, we decided to present a brief description of each functional requirement **TODO: composed by an id, the corresponding name, the level, and the XXXX**.

For this example the notation of level was based on XXXX. For this, we decided to use a level from Cloud to Clamp. The considered levels are the following:

- Cloud - High
- Sea - Medium
- Clamp - Low

Therefore, the functional requirements of the proposed solution are briefly specified in the table 4.1.

The functional requirements presented in the previous table, can be grouped in **TODO: three main groups**.

4.2 Quality Attributes

When designing a system, one of the most important things to consider is to specify and describe well all the quality attributes or non-functional requirements. These kind of requirements are usually Architecturally Significant Requirements and they are the ones that require more of the architects attention, as they reflect directly all the architecture

Table 4.1: Functional requirements specification.

ID	**Name**	**Level**
FR-01	The system must know what are the neighbours of a certain service, based on the service incoming requests.	Cloud
FR-02	The system must know what are the neighbours of a certain service, based on the service outgoing requests.	Cloud
FR-03	The system must be able to identify if there is any problem related to the response time.	Cloud
FR-04	The system must be able to identify if there is any problem related to its morphology or topology.	Cloud
FR-05	The system must be able to identify if there is any problem related to the occupation/load.	Cloud
FR-06	The system must be able to identify if there is any problem related to the number/profile of the client requests.	Cloud
FR-07	The system must be able to identify if there is any problem related to the entire work-flow of one or more requests.	Cloud
FR-08	The system must know how endpoints orders distributions are done when using a specific endpoint.	Cloud
FR-09	The system must know which endpoints are the most popular.	Cloud

decisions and aspects. Sometimes, they are also named the ”itilities” because most of them share this suffix in the word.

To specify them, we usually use a representation called a utility tree, in which we insert the Quality Attribute (QA) by a certain order of priority, for the architecture and for the business inherent to each one, and in order to consider the trade-offs and decide the weight of each in the produced architecture. The codification of the order of priority is the following:

- H - High
- M - Medium
- L - Low

To describe them, we must pay attention and try to define six important things, the **stimulus source**, the **stimulus**, the **environment**, the **artifact**, the **response** and the **measure of the response**.

The figure **??** contains all the raised QA for this project exposed in a utility tree format.

- TODO: Place the quality attributes table here and .

As we can see by the information presented in the table, we can notice that the most important QA is **TODO: explain and analyse them**

4.3 Business Restrictions

- TODO: Place the business restrictions table here and explain it.

4.4 Technical Restrictions

- TODO: Place the technical restrictions table here and explain it.
- At the time there's 'None'.

4.5 Architecture

In this section, the architecture will be presented based on all the previous topics with resource to a diagram. After presenting the diagram, we will cycle thought all the QA, in order to explain where it is reflected and the considerations taken to produce the current architecture.

The figure **??** represent the architecture diagram.

- TODO: Place the architecture image here and explain it based on the Quality attributes, Business restrictions and Technical restriction.

References

[1] Apache Software Foundation, *Apache Giraph.* [Online]. Available: http://giraph.apache.org/.

[2] ArangoDB Inc., *ArangoDB Documentation.* [Online]. Available: https://www.arangodb.com/documentation/.

[3] N. Bronson, Z. Amsden, G. Cabrera, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, S. Kulkarni, H. Li, M. Marchukov, D. Petrov, L. Puzar, Y. J. Song, and V. Venkataramani, "TAO: Facebook's Distributed Data Store for the Social Graph", [Online]. Available: https://cs.uwaterloo.ca/~brecht/courses/854-Emerging-2014/readings/data-store/tao-facebook-distributed-datastore-atc-2013.pdf.

[4] Cloud Native Computing Foundation, *What is Kubernetes?* [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

[5] A. Deshpande, *Surveying the Landscape of Graph Data Management Systems.* [Online]. Available: https://medium.com/@amolumd/graph-data-management-systems-f679b60dd9e0.

[6] Google LLC, *What is OpenCensus?* [Online]. Available: https://opencensus.io/.

[7] R. McColl, D. Ediger, J. Poovey, D. Campbell, and D. A. Bader, "A Performance Evaluation of Open Source Graph Frameworks", Georgia, [Online]. Available: http://www.stingergraph.com/data/uploads/papers/ppaa2014.pdf.

[8] Neo4J Inc., *No Title.* [Online]. Available: https://neo4j.com/docs/.

[9] NetworkX developers, *NetworkX.* [Online]. Available: https://networkx.github.io/.

[10] OpenStack, *What is OpenStack?* [Online]. Available: https://www.openstack.org/software/.

[11] OpenTracing.io, *What is OpenTracing?* [Online]. Available: https://opentracing.io/docs/overview/what-is-tracing/.

[12] C. Richardson, *Microservices Definition.* [Online]. Available: https://microservices.io/.

[13] J. Shun and G. E. Blelloch, "Ligra: A Lightweight Graph Processing Framework for Shared Memory", Pittsburgh, [Online]. Available: https://www.cs.cmu.edu/~jshun/ligra.pdf.

[14] The GanttProject Team, *GanttProject.* [Online]. Available: https://www.ganttproject.biz/.

[15] thefreedictionary.com, *Observing definition.* [Online]. Available: https://www.thefreedictionary.com/observing.

[16] Wikipedia, *ACID(Computer Science)*. [Online]. Available: https://en.wikipedia.org/wiki/ACID_(computer_science).

[17] ——, *Graph*. [Online]. Available: https://en.wikipedia.org/wiki/Graph_(discrete_mathematics).

[18] ——, *Graph database*. [Online]. Available: https://en.wikipedia.org/wiki/Graph_database.

[19] ——, *Software License*. [Online]. Available: https://en.wikipedia.org/wiki/Software_license.