Nuno Morais

Master of Science

Monitoring and deploying services on edge devices

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador:	João Leitão, Assistant Professor, NOVA University of Lisbon

Júri

Presidente:	Name of the committee chairperson
Arguente:	Name of a raporteur
Vogal:	Yet another member of the committee

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

December, 2019

Monitoring and deploying services on edge devices

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Resumo

Lorem ipsum em Português.

Palavras-chave: Palavras-chave (em Português) ...

Abstract

Lorem ipsum in english.

Keywords: Keywords (in English) . . .

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Capítulo 1

Introduction

1.1 Context

Nowadays, the Cloud Computing paradigm is the standard for development, deployment and management of services, it has proven to have massive economic benefits that make it very likely to remain permanent in future of the computing landscape. It provides the illusion of unlimited resources available to services, and has changed the way developers, users and businesses rationalize about applications [1].

Currently, most software present in our everyday life such as Google Apps, Amazon, Twitter, among many others is deployed on some form of cloud service. However, currently, the rise in popularity of mobile applications and IoT applications differs from the centralized model proposed by the Cloud Computing paradigm. With recent advances in the IoT industry, it is safe to assume that in the future almost all consumer electronics will play a role in producing as well as consuming data. As number of devices at the edge and the data they produce increases rapidly, transporting the data to be processed in the Cloud will become unfeasible.

Systems that require real-time processing of data may not even be feasible with Cloud Computing. When the volume of data increases, transporting the data in real time to a Data Center is impossible, for example, a Boeing 787 will create around 5 gigabytes of data per second [2], and Google's self-driving car generates 1 Gigabyte every second [4], which is infeasible to transport to the DC for processing and responding in real-time.

When all computations reside in the data center (DC), far from the source of the data, problems arise: from the physical space needed to contain all the infrastructure, the increasing amount of bandwidth needed to support the information exchange from the DC to the client, the latency in communication from the client to the DC as well as the security aspects that arise from offloading data storage and computation, have directed us into a post-cloud era where a new computing paradigm emerged, Edge Computing.

Edge computing takes into consideration all the computing and network resources that act as an "edge"along the path between the data source and the DC and addresses the increasing need for supporting interaction between cloud computing systems and mobile or IoT applications [5]. However, when accounting for all the devices that are external to the DC, we are met by a huge increase in heterogeneity of devices: from Data Centers to private servers, desktops and mobile devices to 5G towers and ISP servers, among others.

1.2 Motivation

The aforementioned heterogeneity implies that there is a broad spectrum of computational, storage and networking capabilities along the edge of the network that can be leveraged upon to perform computations that rely on the individual characteristics of the devices performing the tasks, which can vary from from generic computations to aggregation, summarization, and filtering of data. [3]

There have been efforts to move computation towards the Edge of the network, Fog Computing [6], which is an extension of cloud computing from the core of the network to the edge of the network, has shown to benefit web application performance [7], additionally, Content Distribution Networks [] and Cloudlets [] are an extension of this paradigm and are extensibly used nowadays.

To fully materialize the Edge Computing paradigm, applications need to be split into small services that cooperate to fulfill applicational needs. Additionally, tools must be developed that allow service and resource discovery as well as service and device monitoring. All of this performed over the inherent scale considered by the infrastructure of the Edge Computing paradigm.

paragrafo com sobre para quem e que isto e (e.g. Google, Amazon, Smart City, Smart Country ?? also porque e que isto e diferente do que ja existe

These tools must be based on protocols that are able to federate all the devices and aggregate massive amounts of device data in order to perform efficient service deployment and management. Additionally, they must handle the churn and network instability that arises when relying on devices that do not have the same infrastructure as those in Data Centers.

1.3 Expected Contribution

To achieve this, we propose to create a new novel algorithm which employs a hierarchical topology that resembles the device distribution of the Edge Infrastructure. This topology is created by assigning a level to each device and leveraging on gossip mechanisms to build a structure resembling a FAT-tree [].

The levels of the tree will be determined by ...undecided... and will the tree be used to employ efficient aggregation and search algorithms. Each level of the tree will be composed by many devices that form groups among themselves, the topology of the groups ...undecided...

The purpose of this algorithm is to allow:

1. Efficient resource monitoring to deploy services on.
2. Offloading computation from the cloud to the Edge and vice-versa through elastic management of deployed services.
3. Service discovery enabled by efficiently searching over large amount of devices
4. Federate large amount of heterogeneous devices and use heterogeneity as an advantage for building the topology.

We plan to research existing protocols (both for topology management and aggregation) and enumerate their trade-offs along with how they behave across different environments. Then, employ a combination of different techniques according to their strengths in a unique way that is tailored for this topology.

1.4 Document Structure

The document is structured in the following manner:

Chapter 2 focuses on the related work, first section covers the different types of topology management protocols, with an emphasis on random and self-adapting overlays, second section studies the different types of aggregation and popular implementations for each aggregation type. Third section addresses resource discovery and how to perform efficient searches over networks composed by a large number of devices. Finally,fourth section discusses recent approaches towards enabling Edge Computing along with discussion about Fog, Mist and Osmotic Computing.

Chapter 3 further explains the proposed contribution along with the work plan for the remainder of the thesis.

Capítulo 2

Related Work

2.1 Peer to Peer

The Peer to Peer (P2P) paradigm has been extensibly used to implement distributed, scalable, fault-tolerant services, it overcomes limitations such as scalability and fault-tolerance that arise from the client-server model.

Collaboration is the foundation of P2P systems, it is the foundation of the scalability provided by P2P. participants of the system (peers) perform tasks that contribute towards a common goal which is beneficial for the overall functioning of the system. Collaboration enables P2P systems that accomplish tasks would otherwise be impossible by an individual server.

Peers contribute with a portion of their resources (such as computing, memory or network bandwidth, e.t.c) to other participants as well as consume resources from other peers in the system. This contrasts with the traditional client-server paradigm where the consumption and supply of resources is decoupled. Finally, fault tolerance is achieved by the absence of a centralized point of failure.

There are many types of services that are based on P2P systems, however, most popular types of services build on this paradigm are: resource location, content delivery, crypto-currency or blockchain, e.t.c.. Popular services built on this paradigm are file sharing applications, (e.g. Napster, BitTorrent, Emule and Gnutella), cryptocurrencies (bitcoin), streaming (Skype), anonymity (TOR) among others.

One factor that imposes a crucial difference in how these systems behave is the membership information, a participant in the system that knows every other participant in the system is said to have full membership information, the alternative is called partial membership, where a peer is only aware of a partial number of elements in the system.

Full membership systems are usually employed in small to medium sized storage solutions like on One-Hop DHTs e.g. Kademlia, Amazon's Dynamo and DynamoDB. However, full membership solutions tend to have scalability problems and behave poorly in the face of churn (participants leaving and entering the system), which increases the workload necessary to maintain the membership information up-to-date.

Partial membership systems rely on some membership mechanism that restricts each peer to only have a few neighboring relations that are used to perform communication (usually through message exchanges) between each other. Similarly to full membership, the number of connections a peer has dictates the scalability of the system, where systems with larger views will have a harder time maintaining them. The accumulation of the partial views of all peers in the system dictates the topology of the overlay network.

2.2 Topology Management

Topology management consists in the creation and management of an **overlay network**, which consists in a logical network built on top of another network (usually the internet). Elements of overlays are connected through virtual links that are a combination of one or more underlying physical links.

The way the aforementioned links are organized dictate the type of overlay: if the links are logically organized by some metric, we call it a **structured overlay**. On contrary, when there are no restrictions form the links, we call it an **unstructured overlay**. Services then leverage on the topologies that are tailored as closely as possible towards application requirements to build services.

2.2.1 Structured overlays

Structured overlays impose restrictions over the neighboring relations that can be established among peers, often based on the identifier of each participant of the system. Usually these identifiers are unique and are independently generated by each peer, then peers employ a global coordination mechanism to tightly control the topology. Examples of common resulting topologies of structured overlays are rings, meshes, hypercubes, among others.

As previously mentioned, a widely used type of structured overlays are distributed hash tables (DHT). Often, DHT's provide application-level routing by employing topologies that can be transversed in logarithmic time. Then, using hash functions to map objects (files, multimedia, messages, e.t.c) to the peer identifier space, and assigning a key-space interval to each peer, peers can find the peer responsible for any key in a logarithmic number of steps.

DHT's have been extensibly used to support many large-scale different types of services (publish-subscribe, resource location, monitoring, e.t.c) and are especially used in

Cloud-based environments. Their popularity derives from providing the previously mentioned functionality while maintaining very little membership information (typically 1% of the peers).

However, a DHT's behavior depends on the correctness of the topology, and when in the presence of high churn or network partitions / failures, the correctness of the topology is harder to maintain, nodes must exchange more messages to ensure that the topology is correct. Additionally, whenever a node fails, DHT's rely on the correctness of the routing infrastructure to replace the failed node, this means that if there is a failure in the routing infrastructure, the DHT may become unrepairable.

Following we present some popular implementations of structured overlays:

- Chord is a distributed lookup protocol that addresses the need to locate the node that stores a particular data item, it specifies how to find the locations of keys, how nodes recover from failures and how nodes join the system.

  Chord's behavior is as follows: it employs consistent hashing to assign each node and key an m-bit identifier, the higher m is, the higher probability of achieving better load distribution among peers (peers receive roughly the same number of keys). Peers are ordered by identifier in a clockwise circle, then, any key k is assigned to the first peer whose identifier is equal or follows k in the identifier space.

  Queries for a given identifier can be passed around the circle via these successor pointers until they find a peer that succeeds the identifier, this is the node that the query maps to. However, this approach by itself is inefficient because it may require transversing the whole circle to find the target.

  To evade this limitation, Chord implements a system of "shortcuts" called the **finger table**. The finger table contains at most m entries, each ith entry of this table corresponds to the first peer that succeeds a certain peer n by 2^{ith} in the circle. This means that whenever the finger table is up-to-date, lookups only take logarithmic time to finish.

- Pastry

- Kelips

2.2.2 Random overlays

2.2.3 Self-adapting overlays

2.3 Aggregation

2.3.1 Types of aggregation

2.3.2 Relevant aggregation protocols

2.4 Resource Discovery

2.5 Offloading computation to the edge

Capítulo 3

Proposed Solution

Capítulo 4

Planning

4.0.1 Proposed solution

4.0.2 Scheduling

Bibliografia

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica e et al. “A View of Cloud Computing”. Em: *Commun. ACM* 53.4 (abr. de 2010), 50–58. ISSN: 0001-0782. DOI: 10.1145/1721654.1721672. URL: https://doi.org/10.1145/1721654.1721672.

[2] M. Finnegan. *Boeing 787s to create half a terabyte of data per flight, says Virgin Atlantic*. 2013. URL: https://www.computerworld.com/article/3417915/boeing-787s-to-create-half-a-terabyte-of-data-per-flight--says-virgin-atlantic.html.

[3] J. Leitão, P. Á. Costa, M. C. Gomes e N. M. Preguiça. “Towards Enabling Novel Edge-Enabled Applications”. Em: *CoRR* abs/1805.06989 (2018). arXiv: 1805.06989. URL: http://arxiv.org/abs/1805.06989.

[4] *Self-driving Cars Will Create 2 Petabytes Of Data, What Are The Big Data Opportunities For The Car Industry?* URL: https://datafloq.com/read/self-driving-cars-create-2-petabytes-data-annually/172.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li e L. Xu. “Edge Computing: Vision and Challenges”. Em: *IEEE Internet of Things Journal* 3 (out. de 2016), pp. 1–1. DOI: 10.1109/JIOT.2016.2579198.

[6] S. Yi, Z. Hao, Z. Qin e Q. Li. “Fog computing: Platform and applications”. Em: *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE. 2015, pp. 73–78.

[7] J. Zhu, D. Chan, M. Prabhu, P. Natarajan e H. Hu. “Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture”. Em: mar. de 2013, pp. 320–323. ISBN: 978-1-4673-5659-6. DOI: 10.1109/SOSE.2013.73.

Apêndice A

Appendix 2 Lorem Ipsum

Anexo I

Annex 1 Lorem Ipsum