Daniel Filipe Santos Pimenta

Mestrado Integrado em Engenharia Informática

Gestão Dinâmica de Micro-serviços na Cloud/Edge

Relatório intermédio para obtenção do Grau de Mestre em

Engenharia Informática

Orientadora: Maria Cecília Farias Lorga Gomes, Profª. Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-orientador: João Carlos Antunes Leitão, Prof. Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Fevereiro, 2019

Resumo

Observa-se, hoje em dia, um crescimento muito elevado da utilização de dispositivos no domínio da Internet of Things (IoT) e de dispositivos móveis, bem como do número de aplicações com consumo elevado de largura de banda (ex.: visualização de vídeos, a pedido). Tal implica que, num futuro próximo, não será viável suportar a quantidade de dados transferidos entre os dispositivos clientes ("*end devices*") e os centros de dados *cloud*, onde tipicamente são alojadas aplicações de acesso ubíquo. O problema do consequente aumento da latência percebido nas aplicações clientes, é ainda agravado no caso de aplicações "sensíveis à latência" *(latency sensitive)*, como sejam aplicações bastante interativas ou de tempo real/quase-real (ex.: carros autonómicos, jogos online, etc.). A localização deste tipo de aplicações na *cloud*, onde é grande a distância entre os clientes e a localização dos centros de dados, resulta em níveis inaceitáveis de Quality of Service (QoS) percebida pelos clientes, ou mesmo a impossibilidade de cumprir os requisitos funcionais das aplicações.

A computação na *edge* (*Edge computing*) surge como resposta aos problemas de latência referidos, ao usar recursos computacionais dos dispositivos na periferia da rede, que se situam mais próximo das aplicações cliente. É ainda possível realizar computações que filtrem os dados gerados na periferia, contribuindo para diminuir o volume de dados em trânsito, e que teriam de ser processados na *cloud*. Comparativamente com os recursos presentes nos centros de dados *cloud*, os recursos dos nós na *edge* são, no entanto, de capacidade computacional bastante limitada. Isto implica que utilizar aplicações monolíticas (tipicamente de grandes dimensões) não é uma opção eficaz na computação na *edge*, quer pelo custo da sua migração/replicação, quer pela impossibilidade de alojar as aplicações nesses nós. O uso da arquitetura de micro-serviços procura solucionar este problema. As aplicações são compostas por múltiplos micro-serviços, cada um com pequena dimensão, oferecendo uma funcionalidade única, com interfaces bem definidas e que comunicam entre si através de mensagens, tornando-os independentes entre si. Desta forma, é possível realizar uma gestão mais eficaz dos recursos disponíveis nos nós periféricos.

O trabalho proposto baseia-se num sistema, já existente, de gestão automática de migração/replicação de micro-serviços, dos centros de dados *cloud* para a periferia e entre componentes periféricos. O objetivo do trabalho é estender esse sistema tendo em conta

um conjunto mais alargado de métricas (ex.: taxa de ocupação do CPU e *throughput*, para além da latência) que permita utilizar os recursos limitados de modo mais eficiente, e permitir uma QoS adequada às aplicações. Será ainda re-avaliada a definição da arquitetura tendo em vista acomodar a existência de nós computacionais heterogéneos no *continuum* entre a *cloud* e a periferia, bem como as características das aplicações em termos do número de micro-serviços possíveis de replicar/migrar.

Palavras-chave: *Cloud*, *Edge*, Micro-serviço, Gestão de recursos, Virtualização

Abstract

There is nowadays a very high growth in the use of devices in the domain of IoT and mobile devices, as well as the number of applications with high bandwidth consumption (e.g. video on-demand). This implies that in the near future it will not be feasible to support the amount of data transferred between end devices and data centers, where ubiquitous access applications are typically hosted. The problem of the consequent increase in perceived latency in client applications is further aggravated in the case of latency-sensitive applications, such as very interactive or real-time applications (e.g. autonomic cars, online games, etc.). The location of this type of cloud applications, where the distance between customers and the location of data centers is large, results in unacceptable levels of QoS perceived by customers, or even the impossibility of meeting the functional requirements of such applications.

Edge computing emerges as a response to the latency problems referred to by using computing resources of the devices at the edge of the network, which are closer to the client applications. It is also possible to perform computations that filter the data generated in the edge, contributing to decrease the volume of data in transit, that would have to be otherwise processed in the cloud. Compared with the resources available in the cloud data centers, the resources of the nodes in the edge are, however, of very limited computational capacity. This implies that using monolithic (typically large) applications is not an efficient choice in edge computing, either because of the cost of its migration/replication or because it is impossible to host the applications on those nodes. The use of the micro-services architecture seeks to solve this problem. The applications are composed of multiple micro-services, each with a small dimension, offering a unique functionality, with well-defined interfaces that communicate with each other through messages, making them independent of each other. In this way, it is possible to perform a more efficient management of the available resources in the edge devices.

The proposed work is based on an existing system of automatic micro-service migration/replication management, from the cloud data centers to the edge and between edge components. The goal of the work is to extend this system by taking into account a broader set of metrics (e.g. occupation rate of CPU and throughput, in addition to latency) to allow the limited resources to be used more efficiently, and to allow adequate

QoS applications. We will also re-evaluate the architecture definition in order to accommodate the existence of heterogeneous computational nodes in the edge, as well as the characteristics of the applications, in terms of the number of possible micro-services to replicate/migrate.

Keywords: Cloud, Edge, Microservice, Resource Management, Virtualization

Índice

Lista de Figuras ix

Lista de Tabelas xi

Listagens xiii

Glossário xv

Siglas xvii

1 Introdução 1
- 1.1 Contexto e Motivação . . . 1
- 1.2 Problema . . . 2
- 1.3 Contribuições . . . 4
- 1.4 Organização do documento . . . 5

2 Estado da Arte 7
- 2.1 Computação em Cloud . . . 7
  - 2.1.1 Benefícios da Computação em Cloud . . . 10
  - 2.1.2 Limitações e Desafios da Computação em Cloud . . . 11
  - 2.1.3 Casos de estudo . . . 13
- 2.2 Computação na Edge . . . 14
  - 2.2.1 Motivações e Vantagens da Computação na Edge . . . 18
  - 2.2.2 Desafios e Limitações da Computação na Edge . . . 20
  - 2.2.3 Gestão de Recursos na Edge . . . 22
- 2.3 Micro-serviços . . . 23
  - 2.3.1 Vantagens e desafios da arquitetura de micro-serviços . . . 27
- 2.4 Computação na Cloud e Edge . . . 29
- 2.5 Virtualização . . . 29

3 Proposta de Solução 31
- 3.1 Trabalho prévio . . . 31
- 3.2 Extensão à arquitetura . . . 33
- 3.3 Plano de trabalho . . . 34

Bibliografia **37**

I Anexo 1 **41**

Lista de Figuras

1.1 Arquitetura atual [7]. 3

2.1 Os diferentes modelos presentes na computação em cloud [14]. 9
2.2 Representação da interação dos dispositivos na periferia com a computação na edge e cloud [6]. 14
2.3 Taxonomia da computação na edge/fog [25]. 17
2.4 Os vários domínios da computação: cloud, fog, edge, mobile cloud e mobile edge [25]. 17
2.5 O modelo de computação na edge/fog apresentando os recursos na periferia. Os micro-serviços poderão ser migrados para qualquer dispositivo edge/-fog [25]. 18
2.6 Problema da distância na computação em cloud [11]. 19
2.7 Os domínios da gestão de recursos na periferia da rede. [12]. 22
2.8 Comparação entre a gestão de dados em aplicações monolíticas e aplicações baseadas em micro-serviços. [19]. 25
2.9 Comparação entre o modo de operação de aplicações monolíticas e aplicações baseadas em micro-serviços. [19]. 26
2.10 Comparação entre a escalabilidade de aplicações monolíticas com a escalabilidade de micro-serviços [19]. 28

3.1 Arquitetura de um nó do sistema [7]. 32
3.2 Arquitetura da solução [7]. 33
3.3 Duração temporal das tarefas realizadas durante a dissertação. 35

I.1 As diferentes entidades envolvidas na computação em cloud [14]. 41
I.2 Sistema de computadores para disponibilizar o conteúdo mais perto do utilizador [6]. 42
I.3 Comparação entre o beneficio de usar um modelo monolítico ou micro-serviços, considerando a complexidade da aplicação. [18]. 43

Lista de Tabelas

Listagens

Glossário

aplicação monolítica	Uma aplicação monolítica não permite a execução independente dos seus módulos [23].
economia de escala	Economias de escala são os fatores que conduzem à redução do custo médio de produção de um determinado bem, à medida que a quantidade produzida aumenta.
latência	Período de latência é a diferença de tempo entre o início de um evento e o momento em que os seus efeitos se tornam percetíveis.
on-premises	*On-premises software*, também conhecido como *shrink wrap*, é um modelo de distribuição de software que é instalado e operado numa infraestrutura computacional presente no local do cliente.
osmose	Osmose é o processo de difusão de moléculas, de um maior para um menor ponto de concentração [21].

Siglas

Amazon EKS	Amazon Elastic Container Service for Kubernetes.
Amazon ECS	Amazon Elastic Container Service.
AMQP	Advanced Message Queuing Protocol.
APIs	Application programming interfaces.
AWS	Amazon Web Services.
CDN	Content Delivery Network.
CoAP	Constrained Application Protocol.
DDoS	Distributed Denial of Service.
DDS	Data Distribution Service.
EC2	Elastic Compute Cloud.
ESB	Enterprise Service Bus.
GCE	Google Compute Engine.
GCP	Google Cloud Platform.
HTTP	Hyper Text Transfer Protocol.
HTTPS	Hyper Text Transfer Protocol Secure.

IaaS	Infrastructure as a Service.
IEEE	Institute of Electrical and Electronics Engineers.
IoT	Internet of Things.
MCC	Mobile Cloud Computing.
MEC	Mobile Edge Computing.
MQTT	Message Queuing Telemetry Transport.
NIST	National Institute of Standards and Technology.
P2P	Peer-to-Peer.
PaaS	Platform as a Service.
QoE	Quality of Experience.
QoS	Quality of Service.
RPC	Remote Procedure Call.
SaaS	Software as a Service.
SLA	Service Level Agreement.
SLOs	Service Level Objectives.
VM	Virtual Machine.
VOD	Video on Demand.

Capítulo 1

Introdução

Neste capitulo é feita uma introdução ao trabalho, apresentando o contexto e a motivação na Seção 1.1. Na Seção 1.2 é explicado o problema encontrado, que irá justificar as contribuições apresentadas na Seção 1.3. Por último, a Seção 1.4 descreve a estrutura geral do documento.

1.1 Contexto e Motivação

O paradigma de computação centralizado, denominado por computação cloud (*cloud computing*), tem sido o mais predominante, com maior interesse e maior investimento nas últimas duas décadas. Implementado inicialmente na prática pela Amazon, e anos mais tarde seguido por gigantes do mundo tecnológico como a Google, Microsoft, IBM, entre outros, a computação cloud permite fornecer poder computacional, armazenamento e largura de banda como um serviço. Teve como motivação principal a eficiência do processamento da informação que é obtida, devido à economia de escala, em grandes centros de computação. E teve sucesso entre as entidades individuais e empresariais por poderem beneficiar de uma infinidade aparente de recursos computacionais a um preço utilitário.

Com a previsão do crescimento da utilização de dispositivos IoT (Ex.: sensores para *smart cities*, *smart homes*, dispositivos *wearables*) e o aumento da exploração de aplicações com consumo elevado de largura de banda (Ex.: Video on Demand (VOD), streaming de conteúdo vídeo, 4k TV), prevê-se que a rede fique demasiado congestionada devido à grande quantidade de dados transferidos entre os dispositivos *front-end* (Ex.: telemóveis, *tablets*, *desktops*, computadores portáteis, televisões, consolas de jogos) e os centros de dados cloud. Também a distância entre estes dispositivos e os centros de dados cloud impõe uma limitação na interatividade das aplicações (Ex.: realidade virtual e aumentada)

devido à latência elevada entre os mesmos.

Os avanços tecnológicos recentes nos dispositivos periféricos (Ex.: *routers*, *gateways*, *switchs* e estações base) indicam uma possível mudança gradual de paradigma de computação como tentativa de resolver esses problemas. O paradigma de computação distribuída usando os dispositivos periféricos, denominado por computação edge (*edge computing*), pretende solucionar os problemas relacionados com a latência e a transferência elevada de dados entre dispositivos *front-end* e centros de dados cloud. Isto é conseguido ao usar dispositivos na periferia para fazer a ligação entre os dispositivos *front-end* e os centros de dados cloud. Como consequência, os dispositivos *front-end* usados pelos utilizadores podem beneficiar de uma redução de latência devido à proximidade aos dispositivos periféricos. Permite também a filtragem e agregação dos dados nos dispositivos periféricos reduzindo a quantidade de dados transferidos para a cloud.

No entanto, os recursos reduzidos dos dispositivos periféricos complicam a migração/replicação de aplicações monolíticas, para esses dispositivos, por serem demasiado grandes e complexas. Como solução, pode ser usada a arquitetura de micro-serviços que permite desenvolver software com base em múltiplos micro-serviços, de pequena dimensão e *interfaces* bem definidas, que comunicam entre si através de mensagens. Cada micro-serviço pode implementar uma funcionalidade especifica e isolada permitindo, assim, o desacoplamento aos outros micro-serviços. Para além dos benefícios no desenvolvimento de aplicações baseadas em micro-serviços (ex.: módulos bem definidos, modularidade e reutilização, e desenvolvimento distribuído), a utilização desta arquitetura tem benefícios, quer na cloud, quer na edge (ex.: *deployment* independente).

A complexidade da gestão de micro-serviços nos centros de dados cloud e nos dispositivos na periferia da rede é o problema essencial que esta dissertação pretende abordar.

1.2 Problema

Considerando a heterogeneidade e dinamismo dos componentes periféricos, as interligações entre os micro-serviços que compõem as aplicações, e a necessidade de um mapeamento dinâmico e eficiente dos micro-serviços na infraestrutura, o resultado é um sistema complexo impossível de administrar manualmente. Em particular, o problema da migração/replicação automática de micro-serviços resulta num problema complexo ao nível da monitorização do sistema e sua gestão dinâmica.

O objetivo deste trabalho foca-se neste aspeto particular da gestão dinâmica de aplicações baseadas em micro-serviços no contexto de infraestruturas heterogéneas cloud/edge. Nomeadamente no problema da migração/replicação dinâmica de micro-serviços (individual ou de micro-serviços relacionados), dos centros de dados cloud para a periferia e entre componentes periféricos, tal que tenha em consideração as características e carga dos nós disponíveis na infraestrutura, num determinado momento, bem como o volume e local de acessos dos clientes. Este trabalho pretende completar um trabalho anterior, contribuindo para o objectivo de otimizar a gestão dos recursos limitados dos componentes

periféricos, permitindo níveis de latência adequados com a Qualidade de Serviço (QoS) / Qualidade de Experiência (QoE) de aplicações particulares.

O processo de gestão automática, com avaliação e decisão das ações de reconfiguração dinâmica, baseia-se num conjunto de requisitos e métricas que são disponibilizadas por um subsistema de monitorização. Alguns exemplos de métricas são a latência dos pedidos aos micro-serviços e volume de dados associados, a taxa de ocupação da rede, o nível de distribuição de carga pelos componentes periféricos e o custo associado à computação de informação e à comunicação entre dispositivos. Estes valores podem assim ser usados para se tomarem decisões dinâmicas quanto à migração/replicação de micro-serviços e dos nós da infraestrutura considerando, entre outros fatores, os recursos dos componentes periféricos, a sua distribuição geográfica, o volume de dados na rede e a dependência entre serviços.

A visão simplificada da arquitetura atual é apresentada na Figura 1.1.

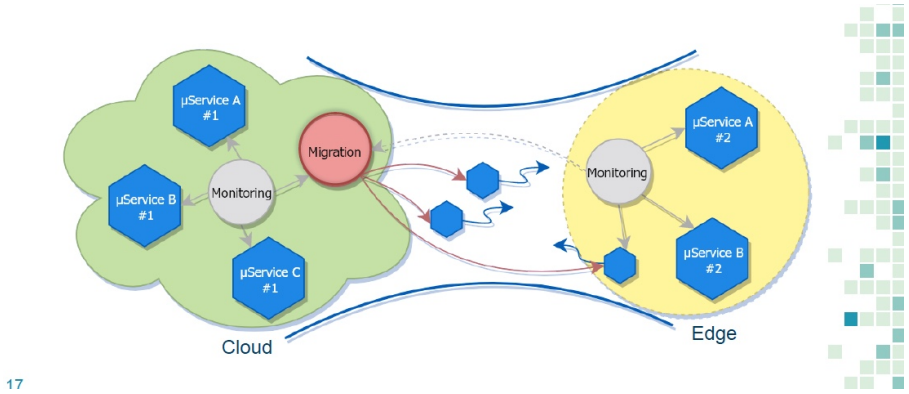

Figura 1.1: Arquitetura atual [7].

A figura realça a possibilidade do *deployment* de micro-serviços (identificados como *µService* A, B, e C) na cloud e em nós da edge. Cada um destes serviços pode estar replicado em ambos os nós (ex.: *µService*A #1 e #2). Existem componentes de monitorização nestes nós ("*Monitoring*"). O componente de gestão de micro-serviços (*"Migration"*) está centralizado na cloud.

Serão consideradas alguns aspetos desta arquitetura atual que podem constituir um problema para eficiência do sistema de gestão, e que são os pontos de partida na definição da extensão desse trabalho anterior.

- **Componente de gestão centralizado**. O componente de gestão, responsável pela decisão da migração/replicação de microserviços ("Migration"na figura), está centralizado num centro de dados cloud. Tal pode afetar negativamente a eficiência da solução atual, devido à distância entre os nós na periferia e o componente de gestão. Serão assim considerados mecanismos de gestão descentralizados do próprio componente de gestão.
- **Métricas de decisão**. As métricas para a decisão das ações de gestão sobre os nós e serviços são, atualmente, limitadas à percentagem de CPU, percentagem de RAM

e número de *bytes* transferidos. Será considerado que métricas adicionais deverão ser tidas em conta no processo de decisão, de modo a melhorar a eficiência dos recursos e garantir a QoS das aplicações. Por exemplo, a taxa de ocupação da rede, a localização dos dispositivos periféricos e sua carga).

- **Dependência entre micro-serviços**. Muitas vezes a decisão relativa a uma ação sobre um serviço afeta um conjunto de outros serviço que estão dependentes do primeiro (ex.: caso típico de um micro-serviço que implementa um *front-end* de uma aplicação web, tal como seja uma aplicação de venda de meias [1]). A gestão dos recursos na periferia deve ter em conta essas dependências, por forma a melhorar o desempenho do sistema. Com um conhecimento das dependências entre micro-serviços, irá ser possível existir uma gestão mais pro-ativa dos serviços.
- **Decisões pouco dinâmicas**. Atualmente, o sistema de gestão toma decisões de adaptação com base na carga de um único serviço de cada vez. Serão estudadas formas adicionais de adaptação, por exemplo, tendo em conta o pré-conhecimento de onde um serviço terá um elevado volume de acessos numa data determinada.

1.3 Contribuições

A contribuição principal deste trabalho é a extensão de um sistema simplificado, já existente, de gestão e coordenação automático para migração/replicação de micro-serviços, entre componentes na periferia e centros de dados cloud. O objetivo da extensão ao sistema é torná-lo mais dinâmico, adaptando-se melhor às condições da infraestrutura e da utilização dos serviços, e melhorar as decisões de gestão ao considerar os aspetos da arquitetura referidos na Seção 1.2 - Problema. O desenho da arquitetura e o desenvolvimento inicial do sistema foi efetuado numa dissertação anterior [7]. A extensão do sistema contempla a:

- **Extensão da arquitetura**. Estender o processo de tomada de decisão de reconfiguração dinâmica com métricas adicionais, para além da latência, como por exemplo:
  a) a carga de tarefas e o custo da computação nos componentes da periferia;
  b) contemplar a dependência entre micro-serviços tendo em vista melhorar o desempenho do sistema;
  c) definir um modelo de gestão descentralizado do próprio componente de gestão de micro-serviços.
- **Extensão do middleware**. As considerações adicionais na arquitetura permitem a extensão do middleware ao adicionar novas funcionalidades.

E para a avaliação das implementações realizadas no sistema:

- **Avaliação**. A extensão à aplicação de demonstração e testes (chamada *SockShop*) permite a verificação do desempenho e a correção das extensões realizadas ao sistema.

[1] https://microservices-demo.github.io

1.4 Organização do documento

Para além deste capitulo, o documento está dividido em mais dois capítulos:

2. **Estado da Arte**. Capítulo 2 descreve as dimensões do estado da arte, nomeadamente, computação em cloud, computação na edge, arquitetura de micro-serviços, e gestão e virtualização de recursos.

3. **Proposta de Solução**. O Capítulo 3 apresenta uma descrição do trabalho prévio, da solução proposta e do plano de trabalhos, incluindo a duração de cada uma das tarefa a realizar durante a dissertação.

Capítulo 2

Estado da Arte

O objetivo deste capitulo é apresentar os conceitos e definições necessários para compreender o Capítulo 3, incluindo a computação em cloud (*cloud computing*), a computação em edge (*edge computing*), aplicações com arquiteturas baseadas em micro-serviços, a gestão de recursos na cloud/edge e tecnologias de virtualização de recursos (ex.: máquinas virtuais e *containers* (docker e kubernets)).

2.1 Computação em Cloud

A National Institute of Standards and Technology (NIST) define a computação em cloud como sendo: *um modelo para permitir acesso ubíquo, conveniente e a pedido a um conjunto de recursos de computação partilhados (ex.: redes, servidores, armazenamento, aplicações e serviços) que possam ser rapidamente provisionados e libertados com um mínimo esforço de gestão ou interação do fornecedor do serviço [22].* A computação em cloud surgiu nos início do milénio, motivado pelo facto do processamento de informação em grandes sistemas de computação e armazenamento ser mais económico e facilmente acessível através da Internet. O acesso à computação pode ser visto como um serviço público semelhante à distribuição de água ou eletricidade [14]. Muitos fornecedores de serviços computacionais como a Google e Amazon (abordados na Subseção 2.1.3), IBM e Microsoft estão atualmente a promover este paradigma como uma utilidade [25].

É um dos resultados do *Grid movement* [14] que tinha como objetivo desenvolver uma computação em grelha (*Grid computing*) constituído num sistema distribuído com um grande número de sistemas heterogéneos e diferentes domínios administrativos. O facto de serem utilizados sistemas heterogéneos resultou num problema de gestão difícil, com agendamento, alocação de recursos, distribuição de carga e tolerância a falhas [14]. Embora fosse um projeto popular na comunidade científica não teve impacto na indústria

[14]. Aprendendo com as lições retiradas do movimento *Grid computing*, uma estrutura de suporte à computação em cloud é maioritariamente homogénea em termos de segurança, gestão de recursos e custos, tendo como alvo inicial a computação industrial [14]. Antes do aumento de popularidade da computação em cloud, os sistemas Peer-to-Peer (P2P) foram um dos grandes interesses da comunidade cientifica e industrial [14]. Os dois modelos têm diferenças relevantes, como o facto dos sistemas P2P serem auto-organizados e descentralizados, enquanto que os servidores na cloud têm um único domínio administrativo e uma gestão central.

De acordo com a entidade NIST, a computação em cloud tem cinco características essenciais, três diferentes modelos de serviço e quatro diferentes tipos [22]. As cinco características essenciais [22] são, nomeadamente, *self-service* a pedido, amplo acesso à rede, agrupamento de recursos, rápida elasticidade e medição de serviço:

a) ***Self-service* a pedido**. Esta característica vem do facto do consumidor conseguir adquirir automaticamente, e após necessidade, recursos computacionais como o tempo de computação, o armazenamento ou capacidade de rede sem ser necessária a interação humana com os fornecedores de serviço.
b) **Amplo acesso à rede**. As capacidades da cloud estão disponíveis através da rede e são acedidas por plataformas heterogéneas de clientes através de mecanismos padrão.
c) **Agrupamento de recursos**. Os diferentes recursos físicos e virtuais dos fornecedores cloud são agrupados para dinamicamente fornecer múltiplos consumidores dependendo da procura. Existe uma independência da localização porque normalmente o consumidor não tem controlo nem tem conhecimento da localização exata dos recursos a serem utilizados, mas pode ser possível a identificação da localização mais geral como o continente, pais ou centro de dados.
d) **Rápida elasticidade**. Os recursos devem ser elasticamente provisionados e libertados, em alguns casos automaticamente, de modo a ser possível escalar rapidamente dependendo da procura do consumidor. Desta forma, para o cliente é dada uma visão de que os recursos disponíveis são ilimitados e que podem ser consumidos em qualquer quantidade a qualquer momento.
e) **Medição de serviço**. Os sistemas cloud controlam e otimizam automaticamente a utilização dos recursos através da monitorização de capacidades métricas do sistema (e.g. armazenamento, processamento, utilização de rede, número de consumidores ativos).

Na computação em cloud é feita a distinção entre três modelos disponibilizados aos seus clientes, visualizados na figura 2.1. Estes modelos permitem isolar características para lidar com as diferentes necessidades empresariais, educacionais e sociais [25].

- **Software as a Service (SaaS)**. O modelo SaaS fornece as capacidades necessárias para a utilização de aplicações disponibilizadas pelos fornecedores. O acesso a essas aplicações, por parte dos clientes, é feito através de uma interface bem definida. Neste modelo, o cliente não gere nem controla qualquer componente da infraestrutura cloud, apenas acede aos serviços através da interface a este disponibilizada.

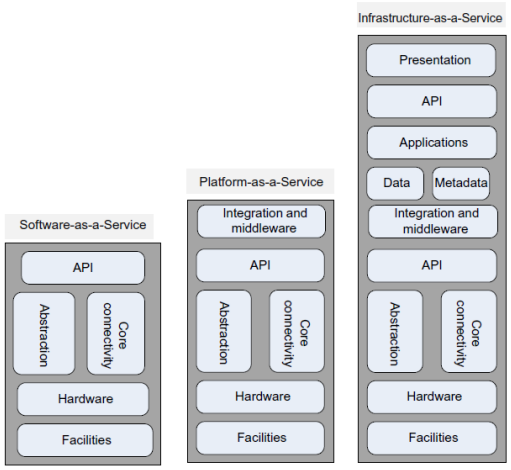

Figura 2.1: Os diferentes modelos presentes na computação em cloud [14].

No modelo SaaS é comum o armazenamento dos dados ser efetuado noutro local relativamente longe da aplicação, não sendo, portanto, um modelo ideal para aplicações que não permitam o armazenamento externo dos dados. As características deste modelo, enumeradas anteriormente, também não são ideais para aplicações que requerem respostas em tempo real.

- **Platform as a Service (PaaS)**. O modelo PaaS é caracterizado por ter a capacidade de alojar aplicações criadas ou adquiridas pelos consumidores utilizando as ferramentas e linguagens de programação dos fornecedores cloud. Tal como no modelo SaaS, o cliente não gere os componentes da infraestrutura cloud, mas neste modelo existe controlo sobre as aplicações que desenvolve. Este modelo é particularmente bom na área de desenvolvimento de software para permitir colaboração entre vários utilizadores e, eventualmente, automatização do processo de desenvolvimento, lançamento e manutenção de aplicações. Aplicações que necessitem de ter um considerável nível de portabilidade, que utilizem linguagens de programação proprietárias, ou que necessitem do controlo da infraestrutura cloud, não são adequadas ao modelo PaaS.
- **Infrastructure as a Service (IaaS)**. O modelo IaaS fornece ao cliente um conjunto de recursos presentes num sistema de computação, como, processamento, armazenamento e rede. Esses recursos a que o cliente tem acesso podem ser utilizados para executar o mais variado tipo de software. Tal como no modelo SaaS e PaaS, o cliente não gere os componentes da infraestrutura cloud, mas diferentemente dos outros dois modelos, o cliente tem controlo sobre o software que executa no sistema e que pode incluir sistemas operativos e/ou aplicações. Este modelo é utilizado principalmente para arrendamento de poder de computação utilizado para os mais diversos serviços.

2.1.1 Benefícios da Computação em Cloud

A computação em cloud tem várias vantagens por ser um modelo realista, utilizar tecnologias avançadas e recentes, ser conveniente para o utilizador e ser económico:

- **Modelo realista**. O modelo da computação em cloud é realista por ser homogéneo, tanto em hardware como em software, e ter um único domínio administrativo. Este foi um dos aspetos revistos e melhorados após o fracasso do, anteriormente referido, *Grid movement*. O facto de haver um sistema homogéneo e com um único domínio administrativo permite simplificar algumas soluções de problemas relacionados com sistemas de computação, como tolerância a falhas, qualidade de serviço, gestão de recursos ou segurança. Por exemplo, a criação de um conjunto de máquinas virtuais iguais e/ou similares, facilita o *deployment* das aplicações bem como as ações de gestão como atualizações, ou substituição em caso de falhas, nessas máquinas virtuais.
- **Tecnologias avançadas e recentes**. A computação em cloud é desenvolvida e promovida por grandes empresas e grupos comerciais, com grandes capacidades económicas, o que permite vastos investimentos em software, hardware, armazenamento, processadores e redes. Essa tecnologia é necessária para desenvolver novos e melhores sistemas de computação em cloud e competir por uma melhor posição de mercado [14]. O facto dessas grandes empresas terem a capacidade para alcançar uma distribuição global dos seus centros de dados permite também uma maior proximidade aos clientes nas diferentes regiões.
- **Escalabilidade**. A computação em cloud é altamente escalável e elástica por ser consistida por um conjunto homogéneo de sistemas de computação com uma política de pagamento baseada na sua utilização. As aplicações com elevada computação e utilização de dados que se possam particionar podem beneficiar de escalabilidade horizontal para melhorar os seus tempos de execução. E uma aplicação com crescente número de utilizadores pode tirar proveito da elasticidade da cloud de modo a suportar maior carga adicional com esforço mínimo dos *developers* da aplicação [14].
- **Paradigma cliente-servidor**. A computação em cloud baseia-se num paradigma cliente-servidor e a maioria das aplicações cloud atualmente disponíveis tiram proveito de uma comunicação sem estado entre clientes e servidores. Numa comunicação sem estado, cada pedido é independente entre si e não é guardada informação entre pedidos do mesmo cliente. A utilização deste tipo de servidores tem vários benefícios como a sua rapidez e facilidade no estabelecimento de comunicações, facilidade de recuperação de falhas, bem como uma maior simplicidade, escalabilidade e robustez comparada com servidores com estado [14].
- **Preço utilitário**. Como a cloud utiliza uma política de pagamento baseada na utilização dos seus recursos, evita que os utilizadores tenham que investir numa infraestrutura e efetuar a manutenção de uma sistema de larga escala [14, 26]. Nos grandes

centros de computação, devido à economia de escala, os fornecedores cloud conseguem um aproveitamento melhor dos recursos na cloud obtendo um baixo custo marginal de administração e operação [14, 26]. Devido ao baixo custo de operação é possível que os utilizadores utilizem os serviços cloud a preço reduzido comparativamente a outros centros de menor dimensão. O modelo de negócio praticado na cloud e o resultado da economia de escala permite tornar a computação em cloud cada vez mais popular. A popularidade é absolutamente crucial para que o negócio seja rentável visto que existe um enorme investimento em infraestruturas. Com um negócio rentável, é do interesse dos fornecedores investirem cada vez mais em centros de dados.

2.1.2 Limitações e Desafios da Computação em Cloud

Apesar das diversas vantagens e desenvolvimentos tecnológicos proporcionados pela computação em cloud, ainda há obstáculos neste modelo que devem ser superados, como a garantia da disponibilidade de serviço, dependência ao fornecedor cloud, dependência da velocidade de transferência de dados, previsibilidade do desempenho, licenciamento de software, erros na infraestrutura cloud, segurança, monopólio do mercado, entre outros [14, 15]:

- **Garantia da disponibilidade de serviço**. Um desses obstáculos é garantir disponibilidade de serviço por parte dos fornecedores de cloud. Como a cloud pode ser partilhada por múltiplos utilizadores, o facto de existirem muitos utilizadores a usar o serviço simultaneamente não pode afetar negativamente a disponibilidade de serviço. Uma das soluções, embora não perfeita do ponto de vista económico, é garantir que existam recursos suficientes para satisfazer a previsão do maior número de utilizadores simultaneamente a usar o sistema.
- **Dependência ao fornecedor cloud** Outro problema associado com a computação em cloud está relacionado com a dependência dos utilizadores ao fornecedores cloud. Um utilizador ao usar um dos fornecedores cloud fica dependente do mesmo, sendo difícil a sua mudança para outro fornecedor. Uma solução atualmente em curso, mas não completa, passa por padronizar as tecnologias de modo a existir uma menor dependência ao fornecedor cloud.
- **Velocidade da transferência de dados**. Certas aplicações na cloud utilizam muitos dados ficando bastante dependentes da velocidade de transferência de dados, o que pode causar um ponto de *bottleneck* no sistema. Uma das estratégias implementadas para aliviar o problema é armazenar a informação o mais perto possível do local onde é necessária. Quando a velocidade de transferência na rede é relativamente baixa, uma opção mais rápida e barata passa por armazenar os dados em memória não volátil e enviar os dados através de um método offline. À medida que a velocidade média de transferência dos dados aumenta ao longo dos anos, esta necessidade vai deixando de ser um problema.

- **Previsibilidade do desempenho**. Como dito anteriormente, a cloud é baseada na partilha de recursos para permitir uma política de pagamento utilitária. Por vezes, esta partilha de recursos pode ser problemática no aspeto da previsibilidade do desempenho esperado. Para permitir a elasticidade e escalabilidade da computação em cloud, são necessários novos algoritmos para o controlo de alocação de recursos e distribuição de carga capazes de escalar rapidamente.
- **Licenciamento de software** A atual tecnologia de gestão de licenciamento de software foi desenvolvida para ser utilizada em serviços não distribuídos. A computação em cloud fornece um serviço distribuído, não sendo a tecnologia de licenciamento de software atual adaptável às necessidades da infraestrutura que constituem a cloud.
- **Erros na infraestrutura cloud**. A infraestrutura cloud é complexa envolvendo múltiplos componentes o que torna a deteção de erros no sistema extremamente complexa e difícil. Outra razão para a dificuldade na deteção de erros deve-se ao facto do sistema poder envolver várias organizações com barreiras de segurança pouco definidas, um processo chamado *de-perimeterisation*. É também um problema para a determinação da responsabilidade de erros devido à cadeia complexa de eventos, em diferentes entidades, que muitas vezes é necessária para desencadear o erro. Com a responsabilidade do erro a ser partilhada por várias entidades, muitas vezes é difícil responsabilizar o conjunto de entidades pelo erro causado.
- **Segurança**. A computação em cloud reforça ainda mais alguns problemas relacionados com a ética computacional. Como o controlo da computação passa a ser delegado a um serviço de terceiros, existe maior risco potencial para situações de acesso não autorizado.

  A computação em cloud aumenta o armazenamento e a circulação de dados pessoais entre entidades o que agrava problemas relacionados com roubo de identidade devido ao acesso indevido dessa informação pessoal. Este facto pode colocar em causa o sucesso da computação em cloud devido ao aumento de desconfiança da sociedade perante a (in)segurança deste modelo. É do conhecimento da sociedade que os fornecedores cloud tenham armazenado uma enorme quantidade de sensíveis dados pessoais. A confiabilidade e auditabilidade dos dados é um importante problema a superar.

  Os recentes acontecimentos e acusações relacionados com a venda de dados pessoais por parte de empresas tecnológicas gigantes, caso do Facebook [9], impõe um grande obstáculo à aceitação da computação em cloud como um método viável e seguro para o armazenamento de dados pessoais. E, para complicar mais a situação, a privacidade é afetada por diferenças culturais. Há culturas que favorecem a partilha enquanto outras favorecem a privacidade. A computação em cloud pretende ser global e portanto devem ser discutidas soluções para este tipo de problemas culturais. É critico para o sucesso da computação em cloud que seja obtida a confiança da sociedade porque a cloud necessita de um grande número de utilização para

que seja viável o grande investimento em infraestruturas feito pelas companhias fornecedoras de cloud.

- **Monopólio do mercado**. Apenas grandes companhias conseguem ter o poder económico para investir em infraestruturas cloud. Um outro problema está relacionado com o facto de apenas existirem algumas companhias que dominam o mercado. Existem preocupações sérias relacionadas com a manipulação de preços e políticas.

Com ainda importantes e difíceis problemas para serem superados, o futuro sucesso da computação em cloud está dependente da capacidade de promoção da computação utilitária, por parte das companhias e centros de investigação, para convencer uma maior população das vantagens da computação centrada em rede e conteúdo. É necessário encontrar soluções para aspetos críticos de disponibilidade e qualidade de serviço, escalabilidade e elasticidade, segurança e confiabilidade.

2.1.3 Casos de estudo

A Amazon foi a empresa pioneira que colocou em prática o conceito de computação em cloud à escala global. Após o estudo do potencial impacto que este tipo de computação poderia vir a ter na forma como temos acesso à computação, a aposta na computação em cloud foi reforçada por outros grupos empresariais, como a Google.

- **Amazon**. Os Serviços Web da Amazon (Amazon Web Services (AWS)) disponibilizam um conjunto de produtos baseados na cloud, incluindo computação (ex.: Elastic Compute Cloud (EC2) [16]), armazenamento, bases de dados, analítica, capacidade de rede, móvel, IoT, ferramentas de desenvolvimento e gestão, e aplicações de segurança e empresarial. Com recurso a esses serviços, as organizações podem ser mais ágeis, diminuir os custos de tecnologia da informação e escalar mais facilmente [4]. Tem também desenvolvidas uma serie de soluções, baseadas nos seus próprios serviços AWS, para ajudar os *developers* a resolverem certos problemas comuns [3]. Dois dos serviços AWS, que se destacam por terem uma importância relevante no contexto deste trabalho, é o Amazon Elastic Container Service (Amazon ECS) [1] e o Amazon Elastic Container Service for Kubernetes (Amazon EKS) [2], que permitem a gestão de *containers* Docker e Kubernetes, respetivamente, facilitando a execução e escalabilidade de aplicações conteinerizadas na plataforma AWS.
- **Google**. A Google Cloud Platform (GCP) é a plataforma de cloud da Google que fornece diversos serviços cloud para computação (ex.: Google Compute Engine (GCE)), armazenamento, rede, segurança, análise de dados, media, inteligência artificial e aprendizagem automática, IoT, gestão de Application programming interfaces (APIs), ferramentas para desenvolvimento e gestão de aplicações e migração de dados, de aplicações e de estruturas. Disponibiliza também ferramentas e produtos como o G-Suite, Google Maps, Cloud Identity, Chrome Enterprise, Android, Apigee, Firebase, etc, que usam os seus serviços GCP [10]. Em relação à tecnologia de *containers*, a Google está diretamente associada à Kubernetes, por ter sido o *developer*

inicial do sistema [13].

2.2 Computação na Edge

A computação na edge é um novo paradigma promissor de computação em cloud descentralizado que coloca a aquisição de dados e funções de controlo, o armazenamento de conteúdo com grande utilização de rede, e as aplicações, mais perto do utilizador final. São utilizados dispositivos na periferia da rede (Internet ou rede privada) ligados a uma arquitetura de computação em cloud com maior capacidade de recursos [6]. É possível assim mover alguma carga computacional e aplicações dos centros de dados centralizados para a periferia da rede, de modo a aproveitar melhor os recursos computacionais atualmente pouco utilizados [5]. Deste modo, a computação na edge atua como camada intermédia entre o utilizador e os centros de dados cloud para oferecer um serviço com menor latência de rede [11]. Os dispositivos na periferia complementam as computações realizadas na cloud [5].

A origem da computação na edge pode ser associada ao aparecimento de Content Delivery Network (CDN) que utiliza nós na periferia para melhor o desempenho da web ao fazer *cache* e *pre-fetch* de, principalmente, conteúdo com grande utilização de rede como é o caso do conteúdo multimédia. A computação na edge generaliza esse conceito ao ser integrado numa infraestrutura de computação em cloud distribuída. E em vez de apenas fazer *cache* de conteúdo web, o objetivo é ser possível executar código arbitrário nos nós periféricos [26]. Uma das possibilidades para permitir a execução de código arbitrário em dispositivos que normalmente estão adaptados a uma certa carga de trabalho é a utilização de virtualização, discutida na Seção 2.5. O código a executar pode ser encapsulado em máquinas virtuais ou *containers* para existir isolação, segurança e gestão de recursos [26].

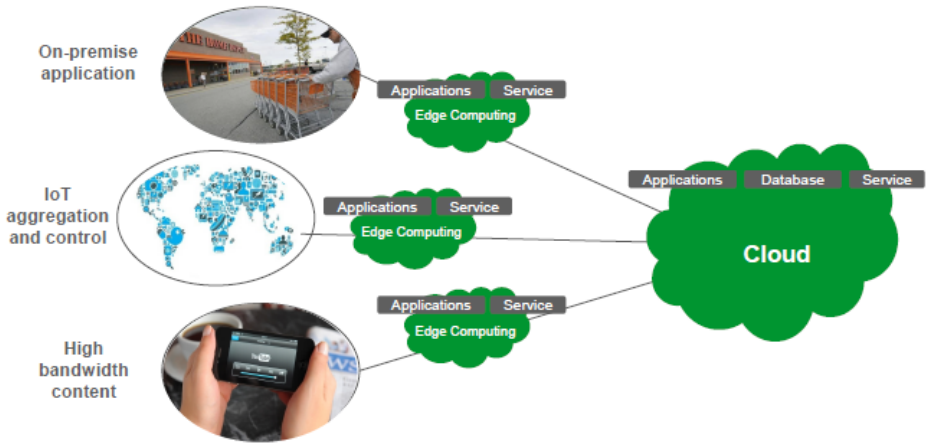

Figura 2.2: Representação da interação dos dispositivos na periferia com a computação na edge e cloud [6].

É possível distinguir na figura 2.2 três tipos de aplicações na periferia: aplicações on-premises, aplicações para agregação e controlo de dados IoT e distribuição de conteúdo com elevada utilização de rede:

- **Aplicações on-premises**. A garantia de disponibilidade é uma grande preocupação das aplicações on-premises (ex.: controlo industrial ou institucional). A utilização de componentes na periferia pode ajudar a superar desafios de disponibilidade no caso de, por exemplo, falha de energia prolongada ou um ataque Distributed Denial of Service (DDoS). As companhias que estão atualmente a utilizar a cloud no seu negócio, podem beneficiar da computação na edge para aumentar a disponibilidade do seu sistema ao introduzir redundância das suas aplicações criticas em componentes periféricos. E aumentar a segurança dos dados, visto que podem ser processados e/ou filtrados próximo onde são recolhidos [6].
- **Aplicações IoT**. As tecnologias associadas ao fenómeno de tornar tudo inteligente (ex.: cidades, agricultura, automóveis e saúde) requerem a utilização de grandes quantidades de sensores IoT [6]. As soluções atuais requerem a migração dos dados para centros de dados cloud para serem interpretados. O que implica uma latência grande, incomportável para aplicações de tempo real. E a grande quantidade de dados gerados por estes sensores IoT não é compatível com o paradigma da computação em cloud porque a quantidade é de tal maneira elevada, que se torna incomportável transportar todos esses dados pela rede. As aplicações para agregação e controlo de dados IoT podem beneficiar da estrutura que compõem a computação na edge para reduzir a latência devido à maior proximidade aos componentes na periferia. E ao processar e/ou filtrar os dados gerados pelos sensores IoT nesses componentes periféricos, é possível reduzir a quantidade de dados transmitidos para os centros de dados cloud.
- **Aplicações com elevada utilização de rede**. As aplicações que requerem utilização elevada de rede (ex.: VOD, 4k Tv, video streaming) são as mais prováveis para congestionar a rede. Por forma a aliviar essa congestão, por parte dessas aplicações, os fornecedores do serviço estão a interligar um sistema de computadores com capacidade para disponibilizar mais rapidamente o conteúdo ao utilizador com menor congestão da rede. Esse sistema de computadores, visualizado na I.2, no Anexo I, que fazem *cache* do conteúdo, são um exemplo de computação na edge [6].

Avanços tecnológicos nos dispositivos perto dos utilizadores (ex.: dispositivos inteligentes, móveis e *wearables*, sensores, nano-centro de dados) e a previsão do aumento da sua utilização [12], podem ajudar a implementar essa mudança de paradigma devolvendo o controlo das aplicações, dados e serviços à periferia da Internet [24]. Esses avanços tecnológicos estão a permitir visionar novos tipos de aplicações aplicáveis em, por exemplo, cidades inteligentes, cuidados de saúde pervasivos, realidade aumentada, multimédia interativa, IoT e assistência cognitiva [11]. Mas, essas novas aplicações têm todas um problema em comum, são extremamente sensíveis à latência [11]. Outro problema está relacionado com a enorme quantidade de dados produzidos e usados por esse tipo de

aplicações.

Um dos aspetos interessantes deste novo paradigma está relacionado com o facto da fronteira entre homem e máquina ficar menos definida. Ou seja, os humanos fazem parte da computação e das decisões relacionadas o que origina sistemas desenhados com foco nos humanos (*human-centered system design)*. Esta visão onde o aspeto principal é o papel fundamental dos humanos é referida por [24] como sendo a *Edge-centric Computing*. A computação P2P, conceito introduzido por volta dos anos 2000 com o aparecimento de sistemas de partilha de ficheiros, está bastante relacionado com a computação na edge. O paradigma *Edge-centric Computing* origina do P2P, mas em vez de tentar uma descentralização completa do sistema, expande o conceito de *peer* para todos os dispositivos na periferia da Internet e combina a computação P2P com a cloud.

Dois exemplos de *Edge-centric Computing* são a Mobile Edge Computing (MEC) e Mobile Cloud Computing (MCC) [25]. A MEC combina a operação de servidores periféricos com estações base para melhorar a eficiência da rede e a utilização de serviços considerando dispositivos móveis. A MCC pretende auxiliar na execução de aplicações nos dispositivos móveis usando nano-centros de dados (*cloudlets*) para deslocar a carga de tarefas dos dispositivos móveis para outros dispositivos com menos restrições de recursos (energia, armazenamento, computação). Os *cloudlets* fazem a intermediação entre os dispositivos móveis e os servidores na cloud permitindo melhorar a Quality of Experience (QoE) dos utilizadores [25].

Outro tipo de computação na periferia é a computação fog. Enquanto que a computação na edge foca-se apenas na rede periférica, a computação fog procura usar tanto a rede na periferia como o núcleo da rede (ex.: routers principais, servidores regionais, wan switches). Este tipo de computação é particularmente bom para ser usado por dispositivos IoT porque os componentes fog podem ser colocados ao alcance desses dispositivos resultando numa significativa redução da latência. Ao contrário da computação na edge, a computação fog tem as capacidades para estender os serviços base da cloud (IaaS, PaaS, SaaS) [25]. Na Figura 2.3, estão ilustrados todos os tópicos relevantes associados à computação na edge/fog. E na figura 2.4, estão visualizados os vários tipos de domínios de computação na edge anteriormente abordados.

Podem ser considerados vários dispositivos na periferia da Internet que podem ser usados para suportar a computação na edge como, por exemplo, *routers*, *gateways*, *switchs*, e estações base [11]. E componentes mais especializados como os dispositivos locais e centros de dados localizados e regionais [6]. Na figura 2.5 é possível a visualização da hierarquia que compõe a computação na edge/fog. Os dispositivos edge/fog permitem estabelecer um ponto intermédio entre os dispositivos dos utilizadores e os centros de dados cloud.

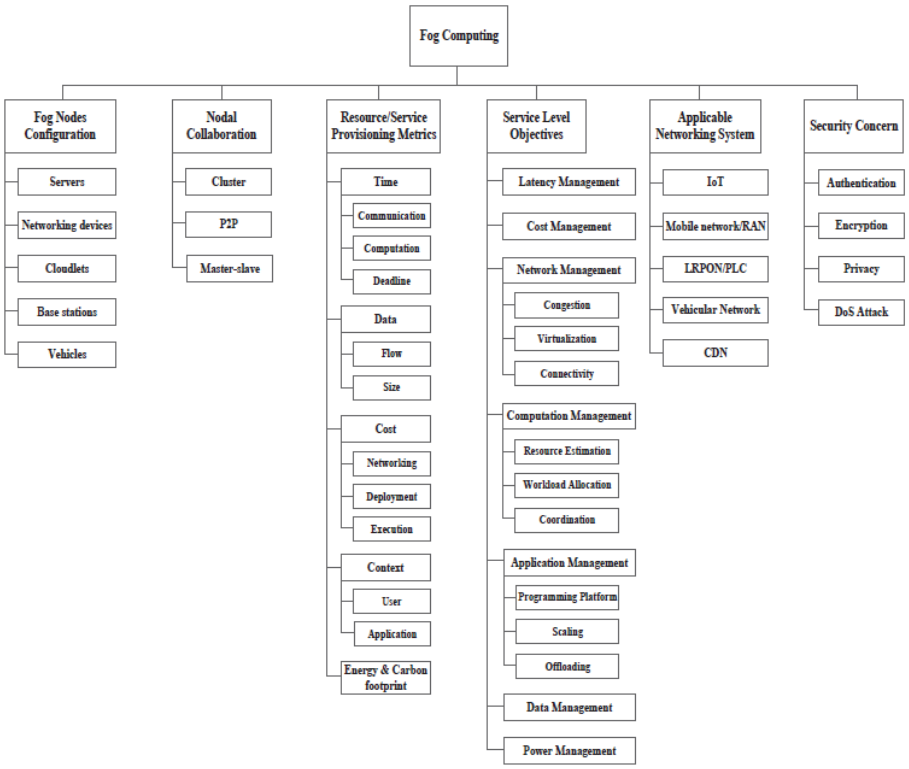

Figura 2.3: Taxonomia da computação na edge/fog [25].

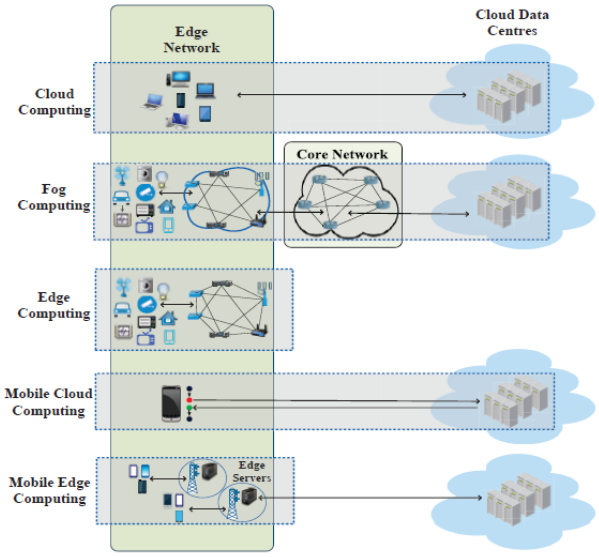

Figura 2.4: Os vários domínios da computação: cloud, fog, edge, mobile cloud e mobile edge [25].

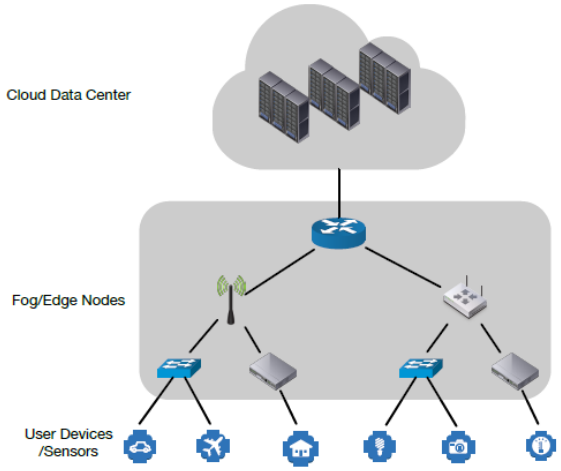

Figura 2.5: O modelo de computação na edge/fog apresentando os recursos na periferia. Os micro-serviços poderão ser migrados para qualquer dispositivo edge/fog [25].

2.2.1 Motivações e Vantagens da Computação na Edge

Podem ser identificados alguns problemas importantes na computação em cloud que a computação na edge pretende solucionar, ou pelo menos, diminuir o seu impacto [24]:

- **Recuperação do controlo das aplicações alojadas na cloud**. A computação na edge pode devolver o controlo das aplicações e dos sistemas aos dispositivos periféricos. Com um desenvolvimento suficiente da segurança na computação na edge, poderá deixar de ser necessário que os utilizadores confiem unilateralmente numa entidade, como acontece no caso da computação em cloud com o fornecedor cloud.
- **Maior privacidade de dados pessoais e privados**. Atualmente, os dados pessoais são partilhados a serviços centralizados como sites e-commerce, serviços de classificação, motores de pesquisa, redes sociais e serviços de localização. A computação na edge introduz a oportunidade de filtrar, nos dispositivos periféricos, os dados pessoais e privados. Isto permite uma maior segurança e maior controlo dos dados enviados para os centros de dados cloud.
- **Oportunidade de utilização de recursos**. Com a utilização de um paradigma centralizado, está se a perder a oportunidade de exploração do grande potencial de computação, comunicação e poder de armazenamento presente nos dispositivos da periferia da rede. A computação na edge pode fazer um melhor aproveitamento desses recursos ao migrar parte da computação para a periferia.

As principais motivações para uma mudança de paradigma centralizado (computação em cloud) para um paradigma periférico (computação na edge) são a diminuição do tempo de transporte de dados (latência), a diminuição de transferência de dados e o aumento da disponibilidade [6, 11]. Esta mudança é particularmente importante sabendo que a utilização da Internet está cada vez mais associada a conteúdo com elevada utilização de rede. Esse tipo de aplicações (ex.: VOD, 4k Tv, video streaming) contribuem para a congestão na rede porque recolhem um elevado volume de dados que é tipicamente

processado na cloud. A computação na edge coloca esse conteúdo e aplicações sensíveis à latência em dispositivos com poder computacional e com capacidades de armazenamento mais perto dos utilizadores [6].

A principal razão para o aparecimento da computação na edge é aproximar o nível de poder computacional atualmente presente em centros de dados centralizados aos utilizadores. Ao explorar novas aplicações na área da computação móvel e IoT, é óbvia a importância dessa aproximação porque afeta a latência e limita a largura de banda. Mesmo utilizando uma conexão direta com tecnologia fibra, a latência está limitada devido à velocidade da luz. E utilizar uma estratégia *multihop* para cobrir uma grande área usando muitos pontos de acesso limita também a latência e a largura de banda porque cada salto na rede aumenta o tempo de encaminhamento [26]. Este fenómeno pode ser observado na Figura 2.6.

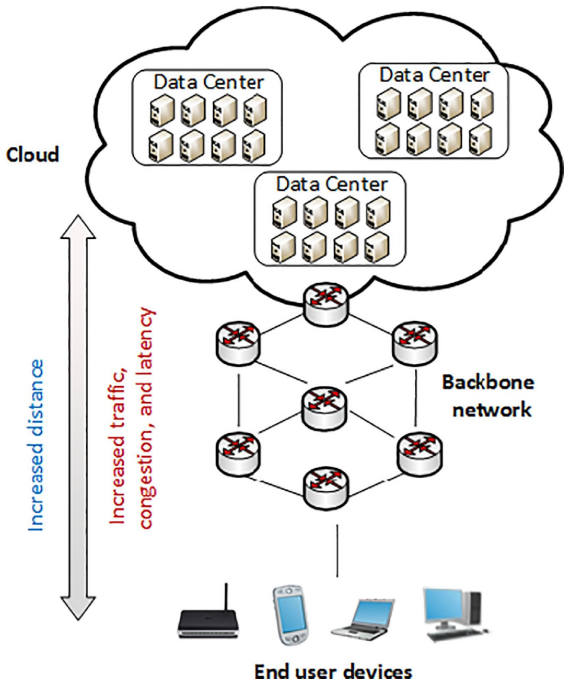

Figura 2.6: Problema da distância na computação em cloud [11].

A proximidade dos nós periféricos, referidos por [11, 25, 26] como cloudlets, permite a existência de serviços cloud mais responsivo. Nova tecnologia, como a realidade virtual e aumentada, pode beneficiar da computação na edge ao processar a informação em cloudlets próximos sem afetar a interatividade e a experiência do utilizador. A proximidade dos cloudlets permite existir baixa latência, maior largura de banda e baixa instabilidade entre o utilizador e o dispositivo que faz a computação [26].

Os dispositivos usados por utilizadores (ex.: computadores pessoais, telemóveis, tablets, dispositivos *wearables*) têm hardware relativamente limitado comparado com os servidores nos centros de dados. Os dados gerados por esses dispositivos normalmente têm que ser enviados para a cloud para serem processados. Mas, às vezes nem toda a informação enviada é necessária para construir o resultado enviado ao cliente. Existem

dados que podem ser filtrados ou analisados em nós na periferia. Apenas a informação e os metadados extraídos após a analise precisam de ser enviados para a cloud aliviando a comunicação entre os dispositivos *front-end* e os centros de dados [5, 26].

A expansão rápida da utilização de serviços cloud tem como consequência inevitável o consumo crescente de energia por parte dos centros de dados cloud. Existe uma grande necessidade de adotar estratégias eficientes de consumo energético, não só devido a preocupações monetárias, mas também ambientais. A computação na edge permite incorporar uma gestão sensível de consumo de energia ao executar parte, ou toda, da computação necessária às aplicações e utilizadores, mais perto onde os resultados são usados [5]. O facto de ser reduzido o tráfego na rede a longa distância entre os utilizadores e os centros de dados cloud permite a existência de uma maior eficácia energética [11].

Funcionando como primeiro ponto de contacto na infraestrutura de cloud distribuída, um cloudlet pode restringir os dados que são enviados para a cloud como forma de garantir a sua privacidade [26].

Se um serviço cloud ficar indisponível devido a falhas de rede, falhas técnicas na cloud ou ataques DDoS, a utilização de cloudlets permite a ação de serviços que escondam temporariamente essa falha [26].

2.2.2 Desafios e Limitações da Computação na Edge

A computação na edge ainda é muito recente e as *frameworks* atualmente públicas, que facilitem o processo de desenvolvimento, ainda são muito limitadas. Esse tipo de *framework* deve satisfazer requerimentos como, por exemplo, o desenvolvimento de aplicações que processem pedidos em tempo real nos nós periféricos. A implementação de processamento de dados em tempo real na periferia da rede ainda é um campo de estudo em aberto. Outro requerimento é a facilitação do *deployment* de aplicações em nós periféricos. É preciso saber onde colocar a carga da aplicação, estudar políticas de conexão e saber que nós utilizar de modo a otimizar a utilização da computação na edge. Para desenvolver tal *framework*, é necessário considerar alguns desafios a nível do *hardware*, *middleware* e *software* [5]:

- **Heterogeneidade de componentes.** Muitos nós computacionais localizados na periferia da rede (ex.: pontos de acesso, estações base, *gateways*, pontos de agregação de tráfico) têm características diferentes e funções especificas para a carga de trabalho que suportam. O objetivo da computação na edge é suportar qualquer tipo de computação e muitos nós na periferia estão adaptados para suportar apenas um certo tipo de computações [5, 25]. Um dos desafios passa por estudar como utilizar esses recursos na periferia de modo a suportar computação mais genérica. Usar técnicas de virtualização, como é o exemplo dos *containers* (ex.: docker e kubernetes, abordados na Seção 2.5) são sérios candidatos para superar este desafio.
- **Descoberta de nós periféricos**. De modo a explorar a periferia da rede são necessários novos mecanismos de descoberta para encontrar os nós que possam ser

utilizados numa arquitetura de cloud descentralizada. Os métodos terão que ser bastante rápidos na identificação da disponibilidade e capacidade de recursos na periferia sem aumentar a latência ou comprometer a experiência do utilizador [5, 25].

- **Particionamento de tarefas e carga**. Um outro desafio vem do facto das tarefas e da carga terem que ser particionadas pelos nós na periferia. O particionamento eficiente e automático das tarefas, sem necessidade de indicar explicitamente as capacidades individuais de cada nó, é um grande desafio da computação na edge. É necessário o desenvolvimento de novas ferramentas de agendamento que particionem as tarefas pelos nós disponíveis [5, 25].
- **Garantia de qualidade de serviço**. O quatro desafio está relacionado com a garantia da qualidade de serviço dos nós na periferia. Esses nós terão que assegurar uma execução confiável das cargas de trabalho inicialmente pretendidas. A carga adicional de trabalho proveniente de dispositivos *front-end*, de outros nós periféricos ou da cloud, não pode comprometer de forma alguma o desempenho desses nós na execução das suas próprias tarefas [5, 25]. Na computação fog, o Service Level Agreement (SLA)[1] é afetado por diversos fatores (ex.: custo de serviço, utilização de energia, características da aplicação, fluxo de dados, estado de rede). Portanto, num ambiente fog é bastante difícil especificar as medidas do serviço e os correspondentes Service Level Objectives (SLOs)[2] [25].
- **Segurança**. A utilização pública e segura de nós na periferia é outro desafio da computação na edge. É necessária uma definição clara dos riscos associados à utilização desses nós periféricos, tanto para os seus donos como para os seus utilizadores. A tecnologia relacionada com *containers* é um potencial candidato para uma utilização de nós na periferia com sucesso. Mas, ainda tem características de segurança pouco robustas que devem ser mais desenvolvidas [5, 25].
  Devido à enorme quantidade de tipos de dados, escala física, frequência de comunicação e variedade de utilizadores, é difícil definir as considerações relativas à segurança na computação na edge. Uma variável importante a ter em conta será a definição de segurança imposta pelos donos dos dispositivos periféricos que pode variar consoante as suas próprias necessidades e disposições [27].
- **Desenvolvimento de *frameworks***. Como os nós na periferia não têm todos os mesmos recursos, é necessário desenhar uma plataforma para desenvolvimento de aplicações distribuídas que tenha políticas de distribuição de tarefas pelos dispositivos periféricos e ajude na visualização de dados.
- **Novas tecnologias e definições padrão.** O nível de padronização nas várias camadas (ex.: infraestrutura, identificação, comunicação, descoberta, gestão, semântica e segurança) da computação na edge e dispositivos IoT ainda está pouco definido [27]. É necessário haver um esforço conjunto das entidades reguladoras para desenvolver

[1] Especificação clara dos serviços que o cliente pode esperar do fornecedor.
[2] Características mensuráveis específicas associadas a um SLA.

novos padrões robustos devido à heterogeneidade dos componentes periféricos. Um dos aspetos importantes a definir é a forma de comunicação entre os dispositivos *front-end* usados pelos utilizadores e os dispositivos IoT com os nós periféricos que fazem a ligação aos centros de dados centralizados. As tecnologias de comunicação têm tido bastante evolução recentemente procurando melhorar a velocidade e confiança dos vários métodos de transmissão. Um dos padrões que se destaca são as especificações da família 802.x definidos pelo Institute of Electrical and Electronics Engineers (IEEE) [27]. Os desenvolvimentos recentes pretendem abordar problemas relacionados com a comunicação na periferia da Internet. Na área de troca de mensagens, pode ser usado o habitual Hyper Text Transfer Protocol (HTTP)/Hyper Text Transfer Protocol Secure (HTTPS). Embora devem ser considerados outros métodos mais especializados, que têm as suas vantagens, como o Message Queuing Telemetry Transport (MQTT), o Constrained Application Protocol (CoAP), o Advanced Message Queuing Protocol (AMQP) e o Data Distribution Service (DDS) [27].

2.2.3 Gestão de Recursos na Edge

Por forma a possibilitar o sucesso da computação na edge, é necessário existir uma gestão eficiente dos recursos dos dispositivos edge, que têm uma capacidade computacional muito inferior e uma distribuição geográfica muito maior, comparativamente aos recursos nos centros de computação cloud. A Figura 2.7 mostra os vários domínios relacionados com a gestão de recursos na edge.

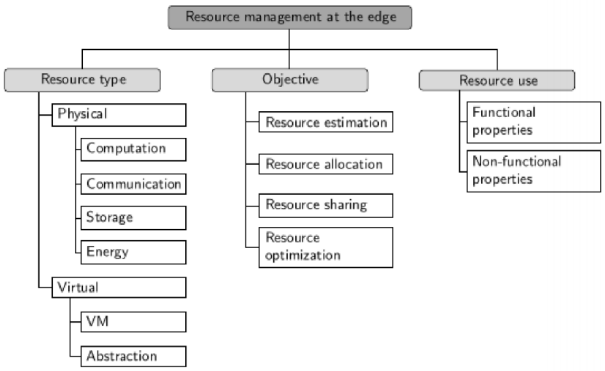

Figura 2.7: Os domínios da gestão de recursos na periferia da rede. [12].

A gestão de recursos considera duas categorias de recursos: físicos (*physical*) (ex.: capacidade computacional, como CPU, ou armazenamento) e virtuais (*virtual*) (ex.: máquinas virtuais, *containers*). E tem vários objetivos, como a capacidade de estimativa dos recursos (*resource estimation*) utilizados, a alocação dos recursos (*resource allocation*) para executar as tarefas, a partilha de recursos (*resource sharing*) como forma de otimização e/ou necessidade, e a otimização dos recursos (*resource optimization*), que está relacionado com os outros objetivos. O último aspeto da gestão de recursos aborda o propósito da utilização dos recursos, onde são consideradas as propriedades funcionais (*functional properties*), ou

seja, o uso da sua capacidade de computação/armazenamento/etc, e as propriedades não funcionais (*Non-functional properties*), como o custo de migração de máquinas virtuais entre dispositivos.

2.3 Micro-serviços

O estilo de arquitetura de micro-serviços, baseado na computação orientada a serviços, pode ser definido como sendo a estruturação de uma aplicação distribuída como um conjunto de serviços coesos e independentes [18–20, 23]. Coeso no sentido de apenas implementar funcionalidades fortemente relacionadas com aquelas que pretende modelar [23]. E independente porque cada micro-serviço executa no seu próprio processo usando mecanismos leves para comunicação (ex.: pedidos de serviço web, ou Remote Procedure Call (RPC)) [19]. O facto dos micro-serviços serem independentemente *deployables* e escaláveis permite existir uma fronteira clara entre cada módulo das aplicações constituídas por micro-serviços, para além de permitir que o desenvolvimento de cada micro-serviço seja feito por equipas independentes [19].

A arquitetura de micro-serviços pode ser considerada como sendo o oposto do estilo monolítico, em que uma aplicação monolítica é construida como uma unidade única. Isto torna as aplicações monolíticas difíceis de usar em sistemas distribuídos sem uso especifico de *frameworks* ou soluções *ad hoc* (ex.: Network Objects, RMI, CORBA) [23]. É difícil de indicar uma definição formal de micro-serviços, mas este tipo de arquitetura tem várias características que, embora não estejam presentes em todas as arquiteturas de micro-serviços, podem caracterizar bem o tipo de aplicação, baseada em micro-serviços, esperada:

- **Componentização de software via serviços** [19]. A Componentização de software pode ser descrita como sendo a construção de sistemas através da junção de vários componentes, sendo um componente uma unidade de software que é independentemente substituível e atualizável. Uma arquitetura de micro-serviços tem esta caraterística ao usar vários micro-serviços para compor um sistema.
  Outro tipo de componente são as bibliotecas, presentes em praticamente todas as linguagens de programação. Em comparação com as bibliotecas, que são usadas pelas aplicações monolíticas como chamadas de funções em memória, os micro-serviços têm a vantagem de serem independentemente "*deployable*". Ou seja, a mudança numa biblioteca da aplicação requerer um novo *deployment* de toda a aplicação. Enquanto que numa aplicação composta por múltiplos micro-serviços, a mudança num serviço apenas requer um novo *deployment* do próprio serviço, e potencialmente, de outros serviços dependentes [19].
  O estilo de arquitetura de micro-serviços disponibiliza apenas *guidelines* para a partição de componentes de uma aplicação distribuída em entidades independentes, não favorecendo qualquer tipo de paradigma ou linguagem de programação [23].

Com a utilização de serviços como componentes há também uma melhor definição da interface exposta ao público, usando mecanismos de RPC, independentemente da linguagem de programação usada, o que permite evitar problemas relacionados com mecanismos de definição explicita de interfaces que cada linguagem de programação oferece. Contudo, a utilização destes mecanismos tem como grande desvantagem o facto de chamadas a procedimentos remotos terem maior custo, principalmente em termos de tempo e coordenação, do que chamadas a funções dentro do mesmo processo [19].

- **Organização baseada nas capacidades do negócio** [19]. Normalmente a separação do desenvolvimento de uma aplicação grande implica a formação de várias equipas que se focam em partes diferentes da aplicação (ex.: equipa para interfaces ou equipa para bases de dados). Isto leva a que, a mais pequena mudança na aplicação possa ter que envolver a coordenação entre várias equipas.
  Usando uma arquitetura de micro-serviços, o desenvolvimento da aplicação pode ser separada em serviços tendo em conta as capacidades do negócio. Esta abordagem implica que as equipas sejam especializadas numa área, mas sim multidisciplinadas incluindo todas as áreas necessárias para a implementação e operação do serviço (ex.: experiência do utilizador, bases de dados, gestão de projetos). A arquitetura de micro-serviços tem a separação entre serviços explicita, tornando a fronteira entre equipas mais clara quando comparada com uma aplicação monolítica.
- **Produtos em vez de projetos** [19]. O modelo de desenvolvimento de software normalmente utilizado tem o objetivo de completar um conjunto de funcionalidades, e quando completo, é entregue a outra organização/equipa responsável pela manutenção do mesmo. A utilização da arquitetura de micro-serviços está associada a outra abordagem: a responsabilidade de todo o tempo de vida de um micro-serviço deve ser apenas de uma equipa, como se o serviço fosse um produto e não um projeto. Isto implica que os *developers* tenham que ter constantemente presente o modo de funcionamento dos seus serviços. Esta mentalidade também pode ser aplicada no desenvolvimento de aplicações monolíticas, mas a granularidade menor dos micro-serviços torna mais fácil a criação de relações pessoais entre os *developers* do serviço e os seus utilizadores.
- **Comunicação simples** [19]. Muitos produtos, abordagens e estruturas (ex.: Enterprise Service Bus (ESB) com funcionalidades para *routing* de mensagens, coreografia, transformação, aplicação de regras de negócio, etc) enfatizam a utilização de mecanismos de comunicação complexas. A arquitetura de micro-serviços favorece outra alternativa: serviços coesos e comunicação simples. As aplicações desenvolvidas com micro-serviços normalmente recebem um pedido, aplicam a sua lógica apropriada e produzem um resultado. As comunicações podem ser simplesmente protocolos REST em vez de protocolos complexos de comunicação. Os dois protocolos mais utilizados são pedido-resposta HTTP, amplamente usado na web e com

facilidade na realização de *cache* dos recursos, e a troca leve de mensagens, com implementações simples como o RabbitMQ e ZeroMQ.
A forma de comunicação dos componentes numa aplicação monolítica é completamente diferente. Os componentes efetuam a comunicação através da invocação de métodos/funções em memória. A grande dificuldade em converter uma aplicação monolítica relativamente grande numa aplicação baseada em micro-serviços é a mudança do padrão de comunicação devido às diferenças de comunicação dos dois modelos.

- **Gestão de dados descentralizada** [19]. O modelo conceitual do mundo é diferente entre sistemas. Por exemplo, a visão que um vendedor tem do cliente é diferente da visão de uma pessoa que dá suporte a clientes. Esta diferença está presente entre aplicações, mas também dentro da mesma aplicação, quando esta está dividida em vários componentes. Enquanto que as aplicações monolíticas preferem uma base de dados exclusiva para armazenar os dados persistentes, muitas empresas preferem a utilização de uma base de dados única para várias aplicações, para facilitarem a imposição das regras do negócio. Já na arquitetura de micro-serviços é preferível deixar cada serviço gerir a sua própria base de dados, por forma a manter a independência a outros serviços diferentes. A ilustração da diferença entre a gestão de dados pode ser vista na Figura 2.8.

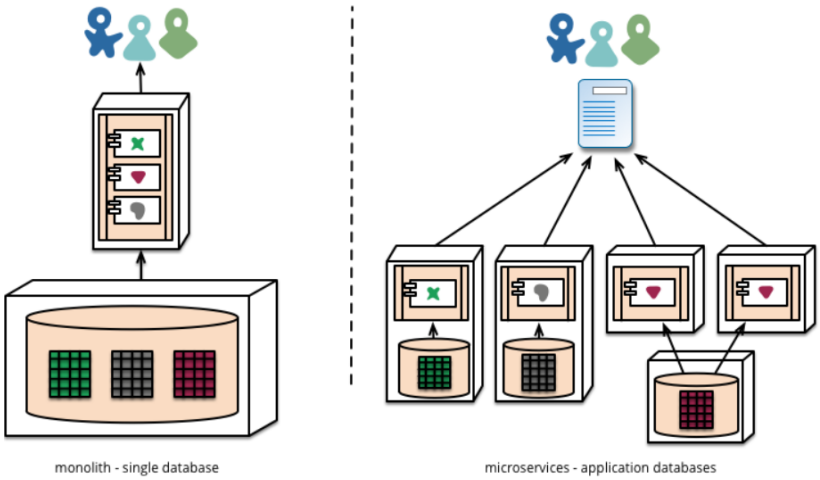

Figura 2.8: Comparação entre a gestão de dados em aplicações monolíticas e aplicações baseadas em micro-serviços. [19].

- **Automação da infraestrutura** [19]. Muitos produtos e sistemas construidos com micro-serviços fazem uso extensivo de técnicas de automação de infraestrutura, que também podem ser usadas na construção, teste e operação de aplicações monolíticas, mas a forma de operação entre estes dois modelos é bastante diferente, como mostra a Figura 2.9).
- **Desenhado para falhas** [19]. Uma das desvantagens da arquitetura de micro-serviços,

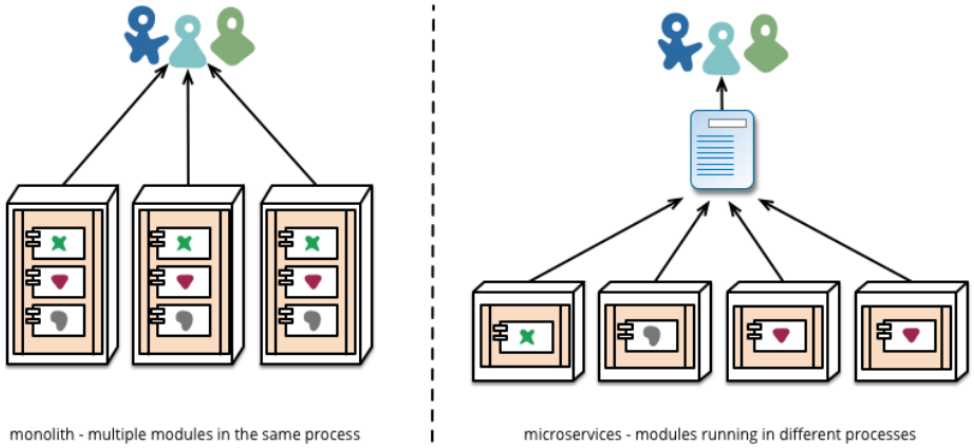

Figura 2.9: Comparação entre o modo de operação de aplicações monolíticas e aplicações baseadas em micro-serviços. [19].

comparativamente com o modelo monolítico, é a maior dificuldade da gestão de falhas. O desenho de aplicações baseadas em micro-serviços deve ter em consideração as falhas dos seus serviços. Como as falhas podem acontecer a qualquer momento, é importante existir uma capacidade de deteção rápida, e possivelmente, uma recuperação automática para garantir a disponibilidade do serviço. Consequentemente, o desenho de micro-serviços enfatiza muito a monitorização e o *logging*, em tempo real, do estado da aplicação, recolhendo dados cruciais (ex.: métricas de operação e negócio, estados, *throughput*, latência) para haja um aviso rápido de qualquer problema que possa existir no sistema. Este aspeto é particularmente importante na arquitetura de micro-serviços, que implica a coreografia e colaboração entre serviços a executar em processos, e eventualmente, máquinas diferentes.

- **Desenho evolutivo** [19]. O modo como são separados os componentes, como serviços, de uma aplicação é decisivo para o sucesso do desenho evolutivo numa aplicação baseada em micro-serviços. Os fatores principais para a formação de um componente são a substituição independente e capacidade de atualização, significando que um serviço deve ser substituído e/ou atualizado sem afetar demasiado os outros serviços da aplicação.

A utilização de micro-serviços pode ser vista como uma ferramenta que, através da agilidade da adição/remoção de serviços à aplicação, associando cada serviço a uma, ou várias funcionalidades, permite maior controlo das alterações na aplicação. Este facto é particularmente importante em aplicações que têm funcionalidades temporárias. Comparado com o modelo monolítico, a utilização de micro-serviços apenas é vantajosa se a aplicação for relativamente complexa, como mostra a Figura I.3, no Anexo I. A utilização de micro-serviços em aplicações de menor dimensão apenas complicam a sua implementação. Por essa razão, muitas pessoas favorecem a utilização do modelo monolítico no inicio do desenvolvimento, mesmo sabendo que o uso da arquitetura de micro-serviços irá ser

mais vantajosa no futuro. Mas essa abordagem tem outro tipo de complicações devido à dificuldade em construir uma aplicação monolítica, que seja facilmente transformada numa aplicação baseada em micro-serviços [18].

2.3.1 Vantagens e desafios da arquitetura de micro-serviços

Tendo enunciado as características principais de uma aplicação baseada em micro-serviços, agora é mais simples perceber as vantagens e desafios da utilização desta arquitetura. Como a arquitetura de micro-serviços tem como base a computação orientada a serviços, esta herda os benefícios inerentes ao modelo orientado a serviços [23]:

- **Fronteira entre módulos bem definida** [18]. A arquitetura de micro-serviços tem como base uma estrutura modular, que é um aspeto muito importante para o sucesso de equipas de desenvolvimento grandes.
- **Dinamismo** [23]. Um serviço, por ser independente e de relativamente pequena dimensão, pode ser facilmente replicado em novas instâncias para separar a carga no sistema.
- ***Deployment* independente** [18]. O facto dos serviços serem independentes permite, quando os componentes da aplicação são devidamente separados, que exista um processo ágil para a remoção, adição e/ou alteração de serviços sem grandes dificuldades.
- **Modularidade e reutilização** [23]. Sistemas complexos podem ser compostos por serviços mais simples. E o mesmo serviço pode ser utilizado em diferentes sistemas, partilhando o mesmo tipo de funcionalidade, promovendo reutilização.
- **Desenvolvimento distribuído** [23]. Como os serviços são independentes entre si, cada equipa pode, após acordar na interface externa do respetivo serviço, desenvolvê-lo em paralelo, em relação às restantes equipas.
- **Diversidade tecnológica** [18]. A tecnologia usada em cada serviço é também independente. Logo, podem ser utilizadas linguagens de programação, *frameworks* e sistemas de bases de dados diferentes em cada serviço.
- **Integração de sistemas heterogéneos e legados** [23]. Os serviços apenas necessitam de usar protocolos padrão para comunicar. Tudo o resto, desde o desenvolvimento, testes, ou manutenção, pode ser feito com qualquer tipo de linguagens/ferramentas/modelos, desde que adequadas para o efeito.
- **Características compatíveis com a computação na edge**. A arquitetura de micro-serviços tem características ideias para o seu uso em dispositivos com pouca capacidade computacional, como é o caso da computação na edge. A divisão de uma aplicação em componentes independentes tem, como consequência, uma maior eficiência da utilização dos recursos periféricos e uma rápida migração/replicação de serviços entre dispositivos e centros de dados cloud.

Mas a arquitetura de micro-serviços também tem alguns desafios:

- **Distribuição** [18]. A arquitetura de micro-serviços é baseada num sistema distribuído, sendo a comunicação feita por RPC. Esta abordagem é mais difícil de programar, comparativamente com aplicações monolíticas, porque as chamadas de procedimentos remotos são mais lentas do que execuções de funções em memória e existe risco de falhas na comunicação entre serviços.
- **Consistência eventual** [18]. O uso de consistência forte não é prática em sistemas distribuídos. Portanto, aplicações baseadas na arquitetura de micro-serviços têm de usar uma consistência mais fraca, como a consistência eventual.
- **Complexidade operacional** [18]. As características do modelo de desenvolvimento de micro-serviços requerem uma equipa competente capazes de gerir os vários serviços, com *deployments* regulares, que compõem as aplicações.
- **Complexidade da rede** [23]. A utilização de vários serviços independentes, com comunicação através de mensagens, pode resultar numa atividade de rede complexa. Esta complexidade dificulta a monitorização e deteção de erros da aplicação como um todo e pode expor o sistema a uma número maior de ameaças contra a aplicação.
- **Especificações do comportamento** [23]. A definição de interfaces usadas para a comunicação entre os serviços não é o suficiente para garantir o funcionamento correto da aplicação. É necessário descrever o comportamento dos serviços, através de tipos de comportamento (*Behavioural types*), para garantir que os serviços tenham ações compatíveis entre si.
- **Maior possibilidade de ataques** [23]. Enquanto que no modelo monolítico os ataques são restritos à máquina e ao sistema operativo, na arquitetura de micro-serviços, a interface externa dos serviços, que é exposta ao público através de APIs, é independente da máquina e até de linguagens de programação, ficando mais exposta a ataques.

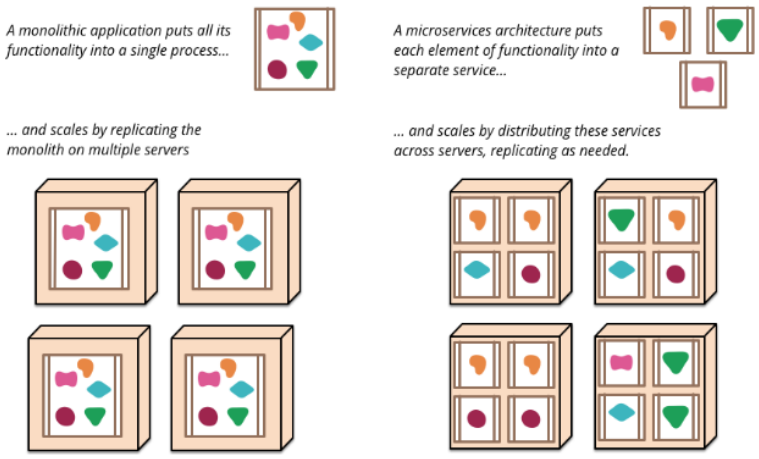

Figura 2.10: Comparação entre a escalabilidade de aplicações monolíticas com a escalabilidade de micro-serviços [19].

2.4 Computação na Cloud e Edge

Na última década tem existido uma centralização e consolidação de serviços e aplicações em centros de dados originando o conceito de computação em cloud [24], que mudou radicalmente quase todos os aspetos da vida humana [11]. Os benefícios económicos da computação em cloud Subseção 2.1.1, e o atual enorme investimento no seu desenvolvimento e recursos, realçam a sua importância na computação futura [26]. As aplicações baseadas na cloud utilizam porém servidores centralizados em centros de dados, que são acedidos por dispositivos na periferia. Devido aos recentes avanços tecnológicos e a novos tipos de aplicações, é insuportável continuar a enviar e receber toda a informação produzida e necessária para um centro de dados central. Com o aumento rápido do número desses dispositivos é colocada cada vez maior pressão nas estruturas computacionais cloud e na comunicação com os centros de dados. Isso pode ter efeitos adversos na qualidade do serviço e na experiência do utilizador [5]. Também a preocupação crescente com problemas relacionados com a confiança, privacidade e autonomia associados à computação em cloud sugere claramente a necessidade de uma mudança de paradigma.

A computação osmótica (*Osmotic computing*) [21] é um paradigma recente, motivado pelo aumento da capacidade dos recursos na periferia da rede, e pelo desenvolvimento na área dos protocolos de transferência de dados, que facilitam a utilização desses recursos. Tem como inspiração o processo de osmose e evidencia a migração/replicação dos serviços (onde se encontrem em maior "concentração") para os vários locais (na cloud e na edge) onde são necessários mas escassos (menor "concentração"de serviços). O objetivo da computação osmótica é permitir a existência um ambiente distribuído de *deployment* autónomo de micro-serviços, através da sua gestão dinâmica tanto em infraestrutura na cloud, como na edge, abordando problemas relacionados com o *deployment*, rede e segurança, e permitindo o suporte mais confiável de IoT.

2.5 Virtualização

A complexidade da gestão de recursos de um sistema, discutida na Subseção 2.2.3, é quão grande quanto maior for o tamanho do sistema, a sua quantidade de utilizadores e diversidade de aplicações que usam esse sistema. Para além desses fatores externos, a gestão de recursos é afetada por fatores internos, como a heterogeneidade do hardware e software, redistribuição de carga e falha de componentes [17]. Uma solução para simplificar algumas tarefas de gestão de recursos, é designada virtualização de recursos, que permite que o estado de aplicações possa migrar ou replicar para outro servidor/dispositivo. Simplifica ainda a utilização de recursos proporcionando aos utilizadores um ambiente familiar para operarem, com isolação sobre os outros utilizadores [17]. A virtualização tem um papel fundamental na cloud [17]: - **Segurança**. Permite a isolação dos serviços que executam no mesmo hardware. - **Desempenho e confiabilidade**. Permite que as aplicações migrem entre plataformas. - **Desenvolvimento e gestão de serviços**. Permite o desenvolvimento

e gestão de serviços por parte dos fornecedores cloud. - **Isolação de Desempenho**. Os fatores externos não afetam tanto o desempenho de aplicações a executar em máquinas virtuais, permitindo a obtenção de métricas de desempenho mais previsíveis. É uma condição critica para garantir a QoS de aplicações a executar em ambientes de computação partilhados.

Existem duas tecnologias que permitem a virtualização de recursos:

a) **Hipervisor**. Através de máquinas virtuais (*Virtual Machine (VM)*), a virtualização é feita usando um ambiente isolado que parece estar a executar no seu próprio hardware, mas na realidade só tem acesso a um conjunto dos recursos. Com múltiplas VMs existe a ilusão de existirem várias instâncias do mesmo hardware, mas na verdade são todas suportadas por apenas um sistema físico [17].

b) **Containerização**. Através de *containers*, a containerização fornece uma camada adicional de abstração e automação de virtualização ao nível do sistema de operação. Permite a virtualização dos recursos com menos *overhead* quando comparado com o hipervisor porque enquanto que as máquinas virtuais apenas podem executar em cima de um sistema operativo, os *containers* não requerem o uso de nenhum sistema operativo [28]. Por esta razão, a escolha de *containers* para a virtualização de recursos em dispositivos periféricos parece ser a mais viável.

Com o foco na containerização como tecnologia de virtualização, podemos destacar duas soluções muito utilizadas: docker e kubernetes.

Capítulo 3

Proposta de Solução

Neste capitulo é apresentada a solução proposta. Inclui, na Seção 3.1, a análise ao trabalho prévio, com uma descrição geral da arquitetura e da solução atualmente desenvolvida. Na Seção 3.2, é proposto um conjunto de alterações a efetuar sobre a arquitetura, com o objetivo de melhorar a eficiência das decisões de gestão, e de modo a tornar o sistema mais dinâmico. Por fim, a Seção 3.3 define o plano de trabalho, bem como a duração temporal de cada tarefa, ao longo da realização da dissertação.

3.1 Trabalho prévio

O trabalho prévio, efetuado numa dissertação concluída no ano 2017/18 [7], permitiu identificar o problema da complexidade de gestão (replicação/migração) de muitos micro-serviços num ambiente de execução cloud e edge. Como solução, foi proposta uma arquitetura (brevemente descrita na 3.2) cujos componentes principais são: conjunto de nós computacionais onde os micro-serviços podem executar, componente de gestão, e micro-serviços (replicados ou não) constituindo uma aplicação. De modo a que um nó computacional possa suportar o *deployment* dinâmico de micro-serviços e a sua gestão dinâmica, é necessário um conjunto de funcionalidades/componentes, cuja organização por níveis se apresenta na Figura 3.1.

Explicando um pouco cada camada de um nó:

1. **Componentes do sistema** (*System components*):
   a) Docker. Usado para a execução e gestão de *containers*, bem como para a obtenção dos valores de utilização de recursos dos mesmos (ex.: CPU, RAM, etc.).
   b) Node exporter. Usado para a obtenção dos valores dos recursos usados pelo nó (ex.: CPU, RAM). Este componente faz parte do sistema de monitorização, tal como apresentado na Figura 3.1.

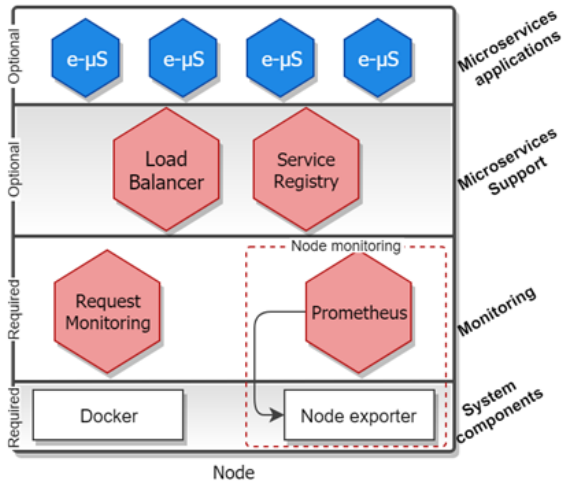

Figura 3.1: Arquitetura de um nó do sistema [7].

2. **Sistema de Monitorização** (*Monitoring*), componentes necessários à monitorização, que foram também integrados na solução:
   a) Monitorização dos pedidos. Cada *container*, contendo o contexto de execução de um único micro-serviço, contém um sub-componente responsável por realizar a agregação da monitorização desse micro-serviços (e.g. sobre a invocação deste e outros micro-serviços) e permitir que esses dados sejam recolhidos pelo Prometheus.
   b) Prometheus. Componente principal do sistema de monitorização, responsável por recolher (através do Node Exporter) as métricas dos nós (CPU, RAM, etc.), e os dados recolhidos em a).
3. **Suporte aos micro-serviços** (*Microservices support*). Estes componentes apenas estão presentes em alguns nós, servindo de suporte aos micro-serviços:
   a) *Load balancer*. Permite fazer uma distribuição da carga, ao redirecionar pedidos para outros nós, quando necessário.
   b) *Service registry*. É utilizado para guardar os endpoints dos serviços, de modo a possibilitar a comunicação entre os mesmos, tendo associada a informação sobre a localização e o estado (ativo, desativado) de cada serviço.
4. **Encapsulamento de micro-serviços** (*Microservices applications*). Este componente contém (é um *wrapper*) o micro-serviço em si, juntamente com um sub-componente necessário ao serviço de registo e descoberta de micro-serviços.

Foi ainda desenvolvido um sistema de gestão de nós e serviços com algumas propriedades de automação identificado por "*µServices Manag.*"na Figura 3.2. Esta Figura 3.2 apresenta um exemplo particular de como os componentes do sistema podem ficar dispostos num ambiente de execução, sendo visível que é possível haver nós quer na cloud, quer na edge. O componente *µServices Manag.*, fica sempre num centro de dados cloud, e é responsável pelas decisões de gestão (replicação, migração e remoção) tomadas sobre cada serviço e por adicionar ou remover nós, quando necessário. Os nós encontram-se

organizados num *cluster*, para que o componente de gestão (*µServices Manag.*) saiba quais os nós disponíveis para a execução de micro-serviços.

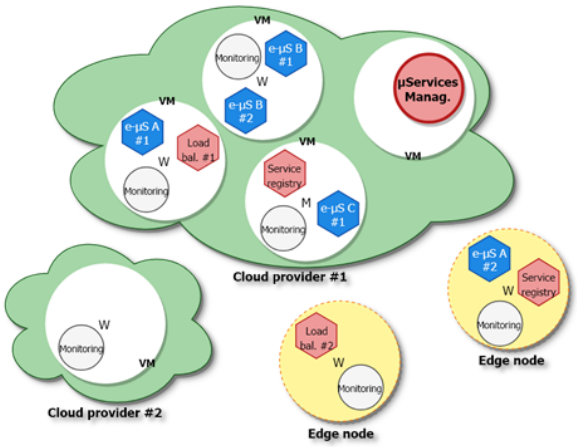

Figura 3.2: Arquitetura da solução [7].

Em relação à gestão dos micro-serviços e da infraestrutura, as decisões de gestão consideraram as seguintes métricas, obtidas pela camada de monitorização: 1. Percentagem de CPU usado. 2. Percentagem de RAM usado. 3. Número de bytes transferidos (apenas afeta os micro-serviços).

Quanto à aplicação realizada como forma de demonstração e teste do sistema, foi usada a aplicação *Sock Shop*[1] composta por um micro-serviço *front-end* e sete micro-serviços *back-end*, desenvolvidos com linguagens de programação diferentes (ex.: Javascript, Java, Go), e utilizados diversos sistemas de bases de dados (ex.: mongoDB, mySQL). A solução atual tem limitações quanto à eficiência das decisões de gestão tomadas e quanto ao dinamismo que é esperado num sistema de gestão autónomo. Com isso em conta, nesta dissertação espera-se fazer uma extensão à arquitetura e ao sistema, de modo a considerar mais fatores que afetem as decisões de gestão da infraestrutura e dos micro-serviços.

3.2 Extensão à arquitetura

A extensão à arquitetura deverá alterar o sistema atual para contemplar novas situações:

- Permitir que sejam replicadas várias instâncias do mesmo serviço, em locais distintos. Deste modo, é possível satisfazer as necessidades dos utilizadores em locais diferentes.
- Alterar a forma como é feita a migração dos micro-serviços, com o objetivo de otimizar os recursos utilizados.
- Aumentar a eficiência das decisões de gestão e da utilização dos recursos ao considerar mais métricas, como: a) A taxa de ocupação da rede; b) A distribuição de carga

[1] https://microservices-demo.github.io

sem afetar demasiado a latência; c) O custo da utilização do dispositivo físico (em termos de computação, energético e monetário).

- Aumentar o dinamismo do sistema ao considerar ações de gestão pro-ativas com base em regras de dependências entre micro-serviços. Deste modo, deverá ser possível melhorar a rapidez das decisões de gestão sobre os micro-serviços dependentes.
- Aumentar o dinamismo do sistema ao considerar ações de gestão adaptativas com base em antecipação de acontecimentos, considerando as características particulares de cada aplicação e micro-serviço. O objetivo é melhorar a rapidez das decisões de gestão sobre alguns micro-serviços.
- Permitir a migração/replicação do nó de gestão para o aproximar aos restantes nós na edge, criando uma gestão distribuída, num modelo hierárquico de nós ao longo do *continuum* cloud/edge.

3.3 Plano de trabalho

O plano de trabalho seguido para a realização da dissertação pode ser decomposto nas seguintes tarefas, cuja duração individual pode ser visualizada no diagrama de Gantt (Figura 3.3):

1. **Estudo aprofundado do trabalho prévio**. Realização de um estudo aprofundado, com análise da implementação e experimentação do sistema, do trabalho efetuado previamente;
2. **Estudo do trabalho relacionado mais recente**. Incluindo algoritmos para definição de regras de decisão, dependências entre micro-serviços, previsão de acontecimentos e distribuição de carga.
3. **Reavaliação da tecnologia de virtualização usada**. Estudo com o objetivo de comparar as vantagens e as desvantagens da utilização do *docker*, comparativamente com o *kubernetes*;
4. **Estudo e desenho da extensão à arquitetura**. Desenhar a extensão à arquitetura, considerando os pontos indicados na Seção 3.2.
5. **Extensão incremental do sistema**. Implementação das funcionalidades novas, incluindo:
   - 5.1. **Diversificação do *deployment* inicial de uma aplicação**. Funcionalidade para permitir a definição de uma configuração inicial que contemple nós na cloud e nós na edge, em vez da configuração inicial atualmente considerada, que só inclui nós da cloud;
   - 5.2. **Múltiplas instâncias de um serviço**. Funcionalidade para permitir a replicação de várias instâncias do mesmo serviço, em locais diferentes simultaneamente;
   - 5.3. **Migração de serviços**. Atualmente, a migração é simulada através da replicação, seguida da desativação da instância inicial do serviço. A nova funcionalidade pretende mudar a forma de migração dos serviços para tentar otimizar a

utilização dos recursos.

5.4. **Mais regras de decisão**. Implementação de mais regras de decisão ao considerar mais métricas, para além das já consideradas, nomeadamente, a taxa de ocupação da rede, a distribuição de carga, e o custo de utilização do dispositivo físico;

5.5. **Dependências entre serviços e ações de gestão pro-ativas**. Adaptar a interface de gestão para permitir a definição das dependências entre serviços e implementar as ações de gestão pro-ativas;

5.6. **Antecipação de acontecimentos e ações de gestão adaptativas**. Adaptação com base em eventos pré-definidos. Incorporar a possibilidade de reconfiguração dinâmica com base na definição de eventos pré-definidos que envolvam um elevado volume de acessos, tendo em atenção o local e a data onde ocorrem;

5.7. **Gestão distribuída**. Alterar o componente de gestão, de um modelo centralizado para um modelo distribuído (ex.: modelo hierárquico);

6. **Avaliação e melhorias**. Avaliação da solução encontrada e realização de eventuais melhorias à arquitetura e às funcionalidades, com recurso a uma extensão da aplicação *Sock Shop*.
7. **Escrita do documento de dissertação**. Elaboração do documento de dissertação, incluindo o trabalho desenvolvido, avaliações, testes, análises criticas e conclusões finais.

Nome da tarefa	Data de Início	Data de Conclusão	Dia de Início	Duração
Disertação de mestrado	04/mar	17/set	0	198
1. Estudo aprofundado do trabalho prévio	04/mar	10/mar	0	7
2. Estudo do trabalho relacionado mais recente	11/mar	17/mar	7	7
3. Reavaliação da tecnologia de virtualização usada	18/mar	24/mar	14	7
4. Estudo e desenho da extensão à arquitetura	25/mar	07/abr	21	14
5. Extensão incremental do sistema	08/abr	30/jun	35	84
5.1. Diversificação do deployment inicial	08/abr	14/abr	35	7
5.2. Múltiplas instâncias de um serviço	15/abr	28/abr	42	14
5.3. Migração de serviços	29/abr	05/mai	56	7
5.4. Mais regras de decisão	06/mai	19/mai	63	14
5.5. Ações de gestão pro-ativas	20/mai	02/jun	77	14
5.6. Ações de gestão adaptativas	03/jun	16/jun	91	14
5.7. Gestão distribuída	17/jun	30/jun	105	14
6. Avaliação e melhorias	01/jul	21/jul	119	21
7. Escrita do documento de dissertação	22/jul	17/set	140	58

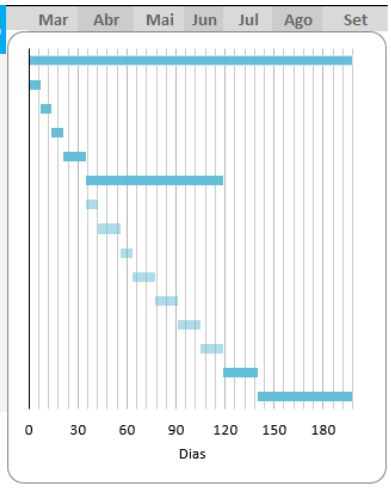

Figura 3.3: Duração temporal das tarefas realizadas durante a dissertação.

Bibliografia

[1] Amazon. *Amazon ECS*. URL: https://aws.amazon.com/ecs (acedido em 02/2019).

[2] Amazon. *Amazon EKS*. URL: https://aws.amazon.com/eks (acedido em 02/2019).

[3] Amazon. *AWS Solutions*. URL: https://aws.amazon.com/solutions (acedido em 02/2019).

[4] Amazon. *Cloud Products*. URL: https://aws.amazon.com/products (acedido em 02/2019).

[5] S. B.P.K.D.S. N. Blesson Varghese Nan Wang. "Challenges and Opportunities in Edge Computing". Em: *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. New York, NY, USA: IEEE, nov. de 2016. ISBN: 978-1-5090-5263-9. DOI: https://doi.org/10.1109/SmartCloud.2016.18. URL: https://ieeexplore.ieee.org/abstract/document/7796149/.

[6] S. Carlini. *The Drivers and Benefits of Edge Computing*. Rel. téc. 226. Schneider Electric's Data Center Science Center, fev. de 2019. URL: https://www.schneider-electric.com/en/download/document/APC_VAVR-A4M867_EN/.

[7] A. V. Carrusca. "Gestão de micro-serviços na Cloud e Edge". Tese de mestrado. Calçada de Alfazina 2, 2825-149 Caparica: Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, set. de 2018.

[9] N. C. Gabriel J.X. Dance Michael LaForgia. Dez. de 2018. URL: https://www.nytimes.com/2018/12/18/technology/facebook-privacy.html (acedido em 02/2019).

[10] Google. *Products and services*. URL: https://cloud.google.com/products (acedido em 02/2019).

[11] A. E.S.U. K. Kashif Bilal Osman Khalid. "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers". Em: *Computer Networks*. Vol. 130. Elsevier, jan. de 2018, pp. 94–120. DOI: https://doi.org/10.1016/j.comnet.2017.10.002. URL: https://www.sciencedirect.com/science/article/pii/S1389128617303778.

[12] S. N.-T. Klervie Toczé. "Where Resources Meet at the Edge". Em: *2017 IEEE International Conference on Computer and Information Technology (CIT)*. Helsinki, Finland: IEEE, set. de 2017. ISBN: 978-1-5386-0958-3. DOI: https://doi.org/10.1109/CIT.2017.60. URL: https://ieeexplore.ieee.org/document/8031490.

[13] Kubernetes. URL: https://kubernetes.io (acedido em 02/2019).

[14] D. C. Marinescu. "Cloud Computing: Theory and Practice". Em: *Introduction*. First. Morgan Kaufmann, 2013. Cap. 1, pp. 1–21. ISBN: 978-0-12404-627-6.

[15] D. C. Marinescu. "Cloud Computing: Theory and Practice". Em: *Cloud Computing: Applications and Paradigms*. First. Morgan Kaufmann, 2013. Cap. 4, pp. 99–131. ISBN: 978-0-12404-627-6.

[16] D. C. Marinescu. "Cloud Computing: Theory and Practice". Em: *Cloud Infrastructure*. First. Morgan Kaufmann, 2013. Cap. 3, pp. 67–99. ISBN: 978-0-12404-627-6.

[17] D. C. Marinescu. "Cloud Computing: Theory and Practice". Em: *Cloud Resource Virtualization*. First. Morgan Kaufmann, 2013. Cap. 5, pp. 131–163. ISBN: 978-0-12404-627-6.

[18] J. L. Martin Fowler. *Microservices*. URL: https://martinfowler.com/microservices (acedido em 02/2019).

[19] J. L. Martin Fowler. *Microservices*. 25 de mar. de 2014. URL: https://martinfowler.com/articles/microservices.html (acedido em 02/2019).

[20] S. D.O.R.R. R. Massimo Villari Maria Fazio. "Challenges in Delivering Software in the Cloud as Microservices". Em: *IEEE Cloud Computing*. Vol. 3. 5. IEEE, nov. de 2016, pp. 10–14. DOI: https://doi.org/10.1109/MCC.2016.105. URL: https://ieeexplore.ieee.org/document/7742281.

[21] S. D.O.R.R. R. Massimo Villari Maria Fazio. "Osmotic Computing: A New Paradigm for Edge/Cloud Integration". Em: *IEEE Cloud Computing*. Vol. 3. IEEE, dez. de 2016, pp. 76–83. DOI: https://doi.org/10.1109/MCC.2016.124. URL: https://ieeexplore.ieee.org/abstract/document/7802525.

[22] P. Mell e T. Grance. *The NIST Definition of Cloud Computing*. Rel. téc. 800-145. Computer Security Division, Information Technology Laboratory, National Institute of Standards e Technology, Gaithersburg, MD 20899-8930: National Institute of Standards e Technology, set. de 2011.

[23] A. L.L.M.M.F.M.R.M.L. S. Nicola Dragoni Saverio Giallorenzo. *Microservices: yesterday, today, and tomorrow*. Cornell University on Software Engineering (cs.SE), Ithaca, NY 14850, EUA. Set. de 2018. URL: https://arxiv.org/abs/1606.04036.

[24] D. E.A.D.T.H.A.I.M.B.P.F.E. R. Pedro Garcia Lopez Alberto Montresor. "Edge-centric Computing: Vision and Challenges". Em: *ACM SIGCOMM Computer Communication Review*. Vol. 45. 5. ACM, out. de 2015, pp. 37–42. DOI: https://doi.org/10.1145/2831347.2831354. URL: https://dl.acm.org/citation.cfm?id=2831354.

[25] R. B. Redowan Mahmud Ramamohanarao Kotagiri. "Fog Computing: A Taxonomy, Survey and Future Directions". Em: *Internet of Everything. Internet of Things (Technology, Communications and Computing)*. Out. de 2017, pp. 103–130. DOI: https://doi.org/10.1007/978-981-10-5861-5_5. URL: https://link.springer.com/chapter/10.1007%2F978-981-10-5861-5_5.

[26] M. Satyanarayanan. "The Emergence of Edge Computing". Em: *Computer*. Vol. 50. IEEE, jan. de 2017, pp. 30–39. DOI: https://doi.org/10.1109/MC.2017.9. URL: https://ieeexplore.ieee.org/document/7807196.

[27] A. Sill. "Standards at the Edge of the Cloud". Em: *IEEE Cloud Computing*. IEEE, abr. de 2017, pp. 63–67. DOI: https://doi.org/10.1109/MCC.2017.23. URL: https://ieeexplore.ieee.org/document/7912167.

[28] G. A. Vitor Goncalves da Silva Marite Kirikova. "Containers for Virtualization: An Overview". Em: *Applied Computer Systems*. Vol. 23. 1. sciendo, mai. de 2018, pp. 21–27. DOI: https://doi.org/10.2478/acss-2018-0003. URL: https://content.sciendo.com/view/journals/acss/23/1/article-p21.xml.

ANEXO I

ANEXO 1

Existem diferentes entidades envolvidas na computação em cloud [14], como mostra a figura I.1.

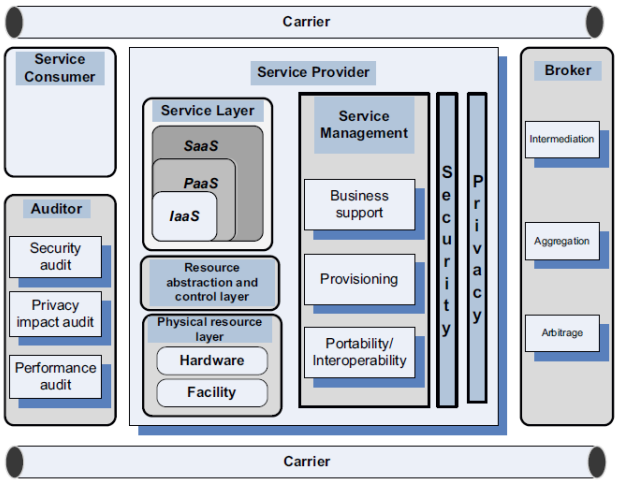

Figura I.1: As diferentes entidades envolvidas na computação em cloud [14].

O consumidor do serviço (***Service Consumer***) utiliza os serviços dos fornecedores cloud mediante uma relação comercial. O fornecedor cloud (***Service Provider***) é a entidade responsável por disponibilizar o serviço cloud. O transportador (***Carrier***) atua como intermediário entre o fornecedor e consumidor, e é responsável por disponibilizar a conetividade e o transporte dos serviços cloud entre ambas as entidades. O corretor (***Broker***) é a entidade que gere a utilização, desempenho e garantias de entrega dos serviços cloud negociando as relações entre o consumidor e o fornecedor. O auditor (***Auditor***) tem a responsabilidade de avaliar os serviços cloud, o desempenho, a segurança e a implementação da cloud através de audições feitas ao sistema que avaliam e medem esse sistema de acordo com certos critérios.

A computação em cloud pode ser divida em quatro tipos diferentes: as clouds privadas, as clouds comunitárias, as clouds públicas e as clouds híbridas [14]:

- **Cloud privada**. O tipo de computação em cloud privada está adaptado e otimizado para ser utilizado por organizações, sendo que a sua infraestrutura é operada exclusivamente para uma organização.
- **Cloud comunitária**. A infraestrutura de uma cloud comunitária está adaptada para ser partilhada por várias organizações com requerimentos específicos.
- **Cloud pública**. As clouds públicas são utilizadas por organizações para venda de serviços cloud ao público em geral ou a um grupo industrial.
- **Cloud híbrida**. A infraestrutura das clouds híbridas é composta por dois ou mais dos tipos de computação em cloud anteriormente referidos. Cada tipo envolvido é agregado através de tecnologia padrão ou proprietária para permitir portabilidade entre os diferentes tipos de cloud.

Figura I.2: Sistema de computadores para disponibilizar o conteúdo mais perto do utilizador [6].

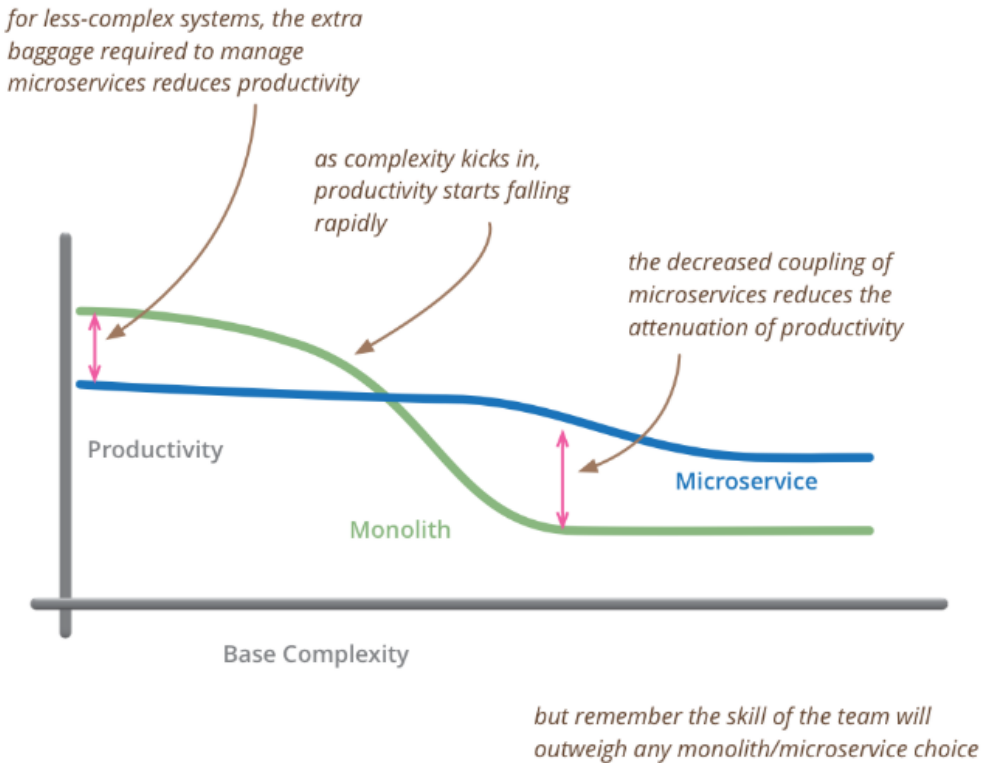

Figura I.3: Comparação entre o beneficio de usar um modelo monolítico ou micro-serviços, considerando a complexidade da aplicação. [18].