CHAPTER 1

Towards an Architecture for Reliable Capacity Provisioning for Distributed Clouds

Jörg Domaschka, Frank Griesinger, Mark Leznik, Per-Olov Östberg, Keith A. Ellis, Paolo Casari, Frank Fowley, and Theo Lynn

Abstract The complexity of computing along the cloud-to-edge continuum presents significant challenges to ICT operations and in particular reliable capacity planning and resource provisioning to meet unpredictable, fluctuating, and mobile demand. This chapter presents a high-level

J. Domaschka (✉) • F. Griesinger • M. Leznik
Institute of Information Resource Management, Ulm University, Ulm, Germany
e-mail: joerg.domaschka@uni-ulm.de; frank.griesinger@uni-ulm.de; mark.leznik@uni-ulm.de

P.-O. Östberg
Umeå University, Umeå, Sweden
e-mail: p-o@cs.umu.se

K. A. Ellis
Intel Labs Europe, Dublin, Ireland
e-mail: keith.ellis5@mail.dcu.ie

T. Lynn et al. (eds.), *Managing Distributed Cloud Applications and Infrastructure*, Palgrave Studies in Digital Business & Enabling Technologies, https://doi.org/10.1007/978-3-030-39863-7_1

conceptual overview of RECAP—an architectural innovation to support reliable capacity provisioning for distributed clouds—and its operational modes and functional building blocks. In addition, the major design concepts informing its design—namely separation of concerns, model-centricism, modular design, and machine learning and artificial intelligence for IT operations—are also discussed.

Keywords Capacity provisioning • Distributed cloud computing • Edge computing • Infrastructure optimisation • Application optimisation

1.1 Introduction

The objective of this book is to introduce readers to RECAP, an architectural innovation in cloud, fog, and edge computing based on the concepts of separation of concerns, model-centricism, modular design, and machine learning and artificial intelligence (AI) for IT operations to support reliable capacity provisioning for distributed clouds. The remainder of this chapter provides a brief overview of computing across the cloud-to-edge (C2E) continuum and the challenges of distributing and managing applications across geo-distributed infrastructure. This chapter also introduces some of the major design concepts informing the RECAP architectural design and provides an overview of the RECAP architecture and components.

P. Casari
IMDEA Networks Institute, Madrid, Spain
e-mail: paolo.casari@imdea.org

F. Fowley
Irish Institute of Digital Business, Dublin City University, Dublin, Ireland
e-mail: frank.fowley@dcu.ie

T. Lynn
Irish Institute of Digital Business, DCU Business School, Dublin, Ireland
e-mail: theo.lynn@dcu.ie

1.2 From the Cloud to the Edge and Back Again

The convergence and increasing ubiquity of wireless internet access, cloud computing, Big Data analytics, social and mobile technologies presage the possibilities of billions of people and things connected through mobile devices and smart objects in the cloud. This phenomenon is heralded as the coming of the fourth industrial revolution, the networked society, the Internet of Things (IoT), indeed the Internet of Everything. Connecting but a fraction of the 1.4 trillion "things" worldwide today is predicted to create US$14.4 trillion and US$4.6 trillion in private and public sector value, respectively, through accelerated innovation and improved asset utilisation, employee productivity, supply chain, logistics, and customer experience (Cisco 2013a, b).

Today, while we are moving towards a society whose social structures and activities, to a greater or lesser extent, are organised around digital information networks that connect people, processes, things, data, and social networks, the reality is still some distance away (Lynn et al. 2018). The dawn, if not the day, of the Internet of Things is here. Haller et al. (2009) define IoT as:

> A world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. Services are available to interact with these "smart objects" over the Internet, query their state and any information associated with them, taking into account security and privacy issues. (Haller et al. 2009, p. 15)

This definition largely assumes that smart objects (end-devices), ranging from the simple to the complex in terms of compute, storage, and networking capabilities, will interact with each other and the cloud to provide and consume services and data, but not necessarily at all times. Furthermore, these smart end-devices, e.g. smart phones or transport sensors, may move to different geographic areas where, for economic, geographic, or technological reasons, they cannot always be connected, yet will be expected to carry on functioning regardless. IoT embodies many of the drivers that see an increased move from cloud-centric deployments to distributed application deployments in the cloud or on the edge infrastructure.

Within the traditional cloud computing paradigm, processing and storage typically take place within the boundaries of a cloud and its underlying infrastructure, and are often optimised for specific types of applications and workloads with predictable patterns. Neither the cloud nor the networks connecting these objects to the cloud were designed to cater for the flood of geographically dispersed, heterogeneous end points in the IoT and the volume, variety, and velocity of data that they generate.

Fog computing and edge computing are two relatively new paradigms of computing that have been proposed to address these challenges. Fog computing is a horizontal, physical, or virtual resource paradigm that resides between smart end-devices and traditional cloud data centres. It is designed to support vertically isolated, latency-sensitive applications by providing ubiquitous, scalable, layered, federated, and distributed computing, storage, and network connectivity (Iorga et al. 2018). In contrast, edge computing is local computing at the edge of the network layer encompassing the smart end-devices and their users (Iorga et al. 2018). If one imagines a cloud-to-edge (C2E) continuum, data processing and storage may be local to an end-device at the edge of a network, located in the cloud, or somewhere in between, in "the fog".

As discussed, while fog computing and edge computing offer solutions for delivering IoT to industry and the masses, they introduce new and significant challenges to cloud service providers, network operators and enterprises using this infrastructure. These environments face a high degree of dynamism as an immediate consequence of user behaviour. Overall, this setting creates a set of challenges regarding how to distribute and run applications in such unpredictable geo-distributed environments. Similar demands are seen at the network edge given the growth of relatively nascent services, e.g. Content Delivery Networks. Spreading infrastructure out over large geographic areas increases the complexity and cost of planning, managing, and operating that physical infrastructure. Firstly, it raises the question of how much infrastructure of what type to place where in the network—a decision that must be made in advance of any service being offered. Secondly, applications deployed over large geographically distributed areas require a detailed understanding of the technical requirements of each application and the impact on the application when communication between an application's components suffers due to increased latency and/or reduced bandwidth. Thirdly, for a service provider along the C2E continuum, the question arises about which (parts) of the various applications in a multi-tenant setting should be operated at the edge and which should not be. This is of critical importance due to the potentially limited compute resources available at each edge location. To add to the complexity, some of these questions must be answered in advance with incomplete data on user demand while others require near real-time decision making to meet unpredictable and fluctuating user demands.

Incorrect placement decisions may result in inflexible, unreliable, expensive networks and services. This is more likely as the decision space becomes so complex; it is no longer realistic for IT teams to cost-effectively foresee and manually manage all possible configurations, component interactions, and end-user operations on a detailed level. As such, mechanisms are needed for the automated and intelligent placement and scaling of dynamic applications and for the management of the physical resources that underpin such applications. RECAP—an architectural innovation in cloud and edge computing to support reliable capacity provisioning for distributed clouds—is posited as such a mechanism.

1.3 Design Principles

This section outlines some of the major design concepts informing the RECAP architectural design, namely separation of concerns, model-centricism, modular design, and machine learning and AI for IT operations.

1.3.1 Separation of Concerns

Separation of concerns is a concept that implements a "what-how" approach to cloud architectures separating application lifecycle management and resource management where the end user or enterprise customer focuses its efforts on what needs to be done and the cloud service provider or cloud carrier focuses on how it should be done (Lynn 2018). At its core, the end user or enterprise customer focuses on specifying the business functionality, constraints, quality of service (QoS), and quality of experience (QoE) (together KPIs) they require, with minimal interference with the underlying infrastructure (Papazoglou 2012). To support a separation of concerns, a detailed understanding of the KPIs but also the relationship between the performance of the applications and underlying infrastructure, and the achievement of these APIs is required.

In multi-tenant environments, for example clouds and networks, the separation of concerns is complicated because the actors will, most likely, belong to different organisations (including competitors), have very different KPIs, different load patterns, different network topologies, and more critically, different priorities. Any architecture for reliable capacity provisioning, whether from an application or infrastructure perspective, across the C2E continuum must have mechanisms to support separation of concerns in an agile way.

1.3.2 Model-Centricism

Due to the complexity, heterogeneity, and dynamic nature of (i) the business domains in which enterprises, cloud service providers, and cloud carriers operate; (ii) the application landscape (including legacy and next generation applications); and (iii) the infrastructure in and upon which these applications operate and are consumed, a flexible software architecture is required that can evolve in line with business, application, and infrastructure requirements. Model-centricism is a design principle that uses machine-readable, highly abstract models developed independently of the implementation technology and stored in standardised repositories (Kleppe et al. 2003). This provides a separation of concerns by design, and thus supporting greater flexibility when architecting and evolving enterprise-scale and hyperscale systems. Brown (2004, pp. 319–320) enumerates the advantages of using models including:

- Models help people understand and communicate complex ideas.
- Many different kinds of elements can be modelled depending on the context offering different views of the world.
- There is commonality at all levels of these models in both the problems being analysed, and in the proposed solutions.
- Applying the ideas of different kinds of models and transforming them between representations provide a well-defined style of development, enabling the identification and reuse of common approaches.
- Existing model-driven and model-centric conceptual frameworks exist to express models, model relationships, and model-to-model transformations.
- Tools and technologies can help to realise this approach, and make it practical and efficient to apply.

To meet the needs of infrastructure providers as well as application operators, an understanding is needed on how the impact of load and load changes on the application layer influences the application's resource demands at the infrastructure layer and further, how competing resource demands from multiple applications, and indeed multiple application providers, impact the infrastructure layer.

From a high-level perspective, users impose a certain load on the applications; that load will change over time. At the same time, users have performance requirements for a given application. For instance, a lack of responsiveness from a website may make them switch while otherwise they would have stayed. The operators of that application want to ensure that some level of performance is guaranteed in order to keep their customers. Hence, it is their task to adapt the performance of the application to the amount of workload imposed by the users. How and whether this can be done depends on the architecture and implementation of the application. For distributed applications (that constitute a huge portion of today's applications), horizontal scaling increases the computational capacity. This, in turn, reduces queuing and keeps latency constant despite increasing workload. Moreover, for applications composed of multiple different components, it is important to understand how load imposed at the customer-facing components ripples through the application graph and impacts the loads on each and every component. Finally, to understand how much performance a component running on a dedicated hardware unit (e.g. processor type, RAM type, and disk type) can deliver under a specific configuration (e.g. available RAM and available cores), a mapping needs to be available that translates load metrics on the application level such as arrival rate of requests of a specific type to load metrics on hardware such as CPU used, RAM used, disk usage, as well as the performance achieved from it. In multi-tenant environments such as virtualised cloud and cloud/edge systems, the mutual impact of multiple, concurrently running components from different owners on the same physical hardware is critical.

A model-centric approach for capacity provisioning for distributed clouds requires at least six models—(1) user models, (2) workload models, (3) application models, (4) infrastructure models, (5) load translation models, and (6) Quality-of-Service (QoS) models (Fig. 1.1).

User models describe the behaviour of users with respect to the usage of individual network-based services. That is, they capture different types of users and their usage patterns over time. What is more, they also describe their movement over geographical regions such that it becomes possible to understand which edge parts of the network will have dedicated

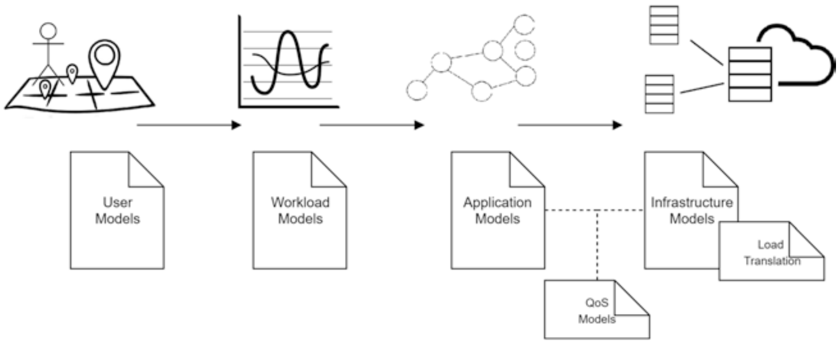

Fig. 1.1 Interdependencies between models

demands for specific services. This is of special interest to edge computing systems as user mobility impacts network load and application access patterns.

Workload models describe the workload issued on a system from users and external systems. While the user model captures the location and type of users, the workload model describes what actions these users execute and how this translates into interaction with which parts of an application.

Application models fulfil multiple purposes. First and foremost, they describe which components compose a distributed application and how these components are linked with each other (static application topology). This part of the application model also captures how to technically install the application in the infrastructure and how to update a running deployment. Deploying an application creates a run-time application topology that describes how many instances of each application component are currently available at which location and how they communicate with each other on a per-instance basis. The **(work)load transition models** as a sub-model of the application model describe for the application how incoming workload propagates through the applications' components and the impact this has on the outgoing links of the component.

As application models are not capable of determining whether or not a given application topology (or scaling factor) is capable of servicing a certain amount of load, as they neither have an understanding of the available hardware and its capabilities nor about how the application load translates on load on the physical layers.

Infrastructure models capture the layout of the physical and virtual infrastructure and represent key components such as compute, storage, and network capabilities, as well as their grouping in racks, data centres, and similar. Furthermore, they describe capabilities of the hardware including hardware architecture, virtualisation platform (e.g. type of hypervisor), and virtual machines (containers) running on the host.

Load translation models enhance the infrastructure models and provide a mapping from workload on application components to resource demands on the physical infrastructure. They are crucial for understanding whether enough physical resources are available to handle workload on application level. In addition, they describe the impact of congestion caused by components with similar hardware demands concurrently running on the same hardware.

Finally, **Quality-of-Service (QoS) models** provide a means to express QoS demands towards an application and monitor the fulfilment of these QoS requirements. In addition, they are able to represent the interdependencies between QoS aspects on different levels, e.g. what QoS requirements at the infrastructure level follow from QoS requirements on the application level. QoS models may be taken as constraints for the optimisation problems solved for rearranging application and infrastructures.

1.3.3 Modular Design

A modular architecture is an architecture where at least some components are optional and there exists the ability to add or remove modules or components according to the needs of a given use case (Aissaouii et al. 2013). The benefits of modular design are well known, not least it supports separation of concerns and provides greater implementation flexibility thus reducing costs and risk. A discrete module or component can be implemented without having to implement the entire system. Enterprises, cloud service providers, and cloud carriers (to a lesser extent) come in all sizes and with their own constraints. A modular design provides these firms with greater choice and flexibility.

1.3.4 Machine Learning and AI for IT Operations

As discussed above, the complexity and scale of distributed cloud infrastructure increasingly require an automated approach. As the deluge of data generated by IoT continues to increase, and as demands from new use cases increasingly require edge deployments, e.g. vCDN, the ability of cloud service providers and cloud carriers to respond quickly to demands on infrastructure, service incidents, and improve on key metrics decreases (Masood and Hashmi 2019). Increasingly, enterprises are looking to AI for IT Operations (or AIOps).

AI for IT Operations (AIOps) seeks to use algorithms and machine learning to dramatically improve the monitoring, operation, and maintenance of distributed systems (Cardoso 2019). Although at a nascent stage of development, AIOps has the potential of ensuring QoS and customer satisfaction, boosting engineering productivity, and reducing operational costs (Prasad and Rich 2018; Dang et al. 2019). This is achieved by:

1. automating and enhancing routine IT operations so that expensive and scarce IT staff have more time to focus on high value tasks,
2. predicting and recognising anomalies, serious issues, and outages more quickly and with greater accuracy than humanly possible thereby reducing mean time to detect (MTTD) and increasing mean time to failure (MTTF), and
3. suggesting intelligent remediation that reduces mean time to repair (MTTR) (IBM 2019; Masood and Hashmi 2019).

Predictions suggest that by 2024, 60% of enterprises will have adopted AIOps suggesting that novel solutions to capacity provisioning must accommodate this shift in enterprise IT operations (Gillen et al. 2018).

1.4 Operational Modes

A model-centric approach assumes cloud-edge applications, and the environments that they run in, can be described by a set of models and that, based on these models, it is possible to optimise both cloud-edge infrastructures and their applications at run-time. As such, an optimisation (control) system and mechanism for creating, validating, and extrapolating these models to large-scale environments are required. This requires a variety of interoperating components, which we refer to here as modes.

Data Analytics Mode: The creation of high-quality models requires an in-depth understanding of many aspects ranging from users to application to infrastructure. For deriving this understanding, a sufficient amount of data needs to be available that can either come from a live system or be derived from a simulation environment. The Data Analytics Mode provides the necessary tooling and guidelines to process those data and generate models from it. The analytics itself is a manual or semi-automated process that applies approaches from statistics and machine learning in order to create the models. It consists of pre-processing and data analysis (or model training respectively). When complete, there is a newly generated insight in the form of a mathematical formula, a statistical relationship, some other model, or a trained neural network. These insights form the baseline of the models that are used by other modes and underlying components.

Run-time Operation Mode: The Run-time Operation Mode uses online optimisation to continuously update geo-distributed infrastructure based on the models and the current deployment scenario (deployed applications, available infrastructure, and user behaviour). Data on the actual usage of the hardware and software requirements are collected during run-time. These data are used by optimisers in the system to weight the current placement and usage against other options and come up with new and better configurations. These are output in the form of an optimisation plan that can then be enacted. This changes the configuration of the actual system. The decisions made in order to improve the system are based on mathematical, stochastic, or programmatic models of the system itself, e.g. the capabilities of the hardware, the needs of the application, current and predicted workload in the system, and the movement of users in the real world.

Simulation and Planning Mode: The Simulation and Planning Mode is capable of performing the same steps as the run-time in what-if scenarios and, hence, evaluates the use and acquisition of new, updated, or re-allocated hardware. This mode supports scenario (what-if) analyses such as “what if I bought more or different hardware at existing sites”, “what if I added a new network site in the topology”, and “how much longer can the available hardware handle my workload, if it keeps growing as predicted”. Hence, simulation helps operators to take strategic decisions about their infrastructure. What is more, using simulation, different placement scenarios are explored and weighed against each other to serve as calibration and constraints for optimisation algorithms.

1.5 RECAP Conceptual Reference Model

Figure 1.2 presents an overview of the RECAP conceptual reference model which identifies the main components in RECAP and how they interoperate. The diagram depicts a generic high-level architecture and is intended to facilitate the understanding of how RECAP operates.

The diagram below outlines the components in the RECAP architecture and shows the process flow loops in the optimisation framework. The **Landscaper Component (1)** acquires information on the state and configuration of the physical and virtual infrastructure resources from disparate sources and presents same as a graph. The **Monitoring Component (2)** uses probes to collect telemetry metrics needed for the modelling and optimisation tasks, including CPU consumption, disk I/O, memory loads, network loads, and packet statistics—both from virtual and physical resources. These are input to the optimisers and the output is used to orchestrate and enact resource changes in the cloud network.

The **Application Optimiser (3)** is used to optimally autoscale the applications and resources. Application scaling refers to horizontal scaling, namely adding additional application components into the system dynamically, while infrastructure scaling relates to vertical scaling, whereby virtual resources are increased for a component. Applications can be scaled locally or globally and may be in response to run-time traffic limits or

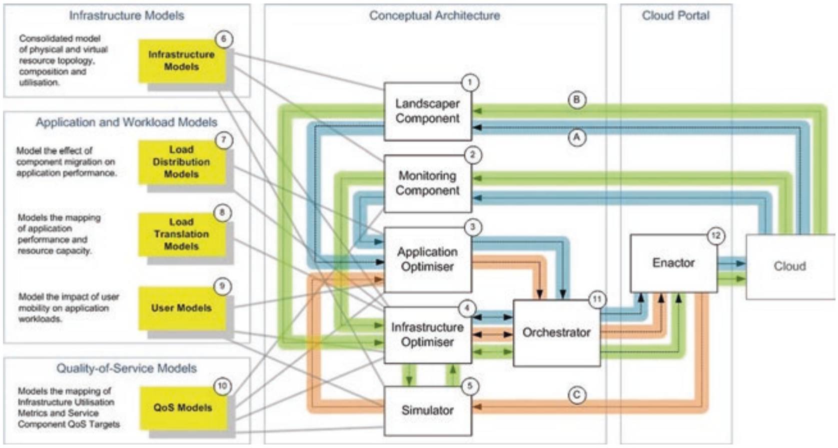
Infrastructure Models
Consolidated model of physical and virtual resource topology, composition and utilisation.
Infrastructure Models
6
Application and Workload Models
Model the effect of component migration on application performance.
Load Distribution Models
7
Models the mapping of application performance and resource capacity.
Load Translation Models
8
Model the impact of user mobility on application workloads.
User Models
9
Quality-of-Service Models
Models the mapping of Infrastructure Utilisation Metrics and Service Component QoS Targets
QoS Models
10
Conceptual Architecture
Landscaper Component
1
Monitoring Component
2
Application Optimiser
3
Infrastructure Optimiser
4
Simulator
5
Orchestrator
11
B
A
C
Cloud Portal
Enactor
12
Cloud

Fig. 1.2 RECAP conceptual reference model

resource levels being reached or may be controlled by data analytic workload predictive systems. The application to be deployed is composed of multiple connected service components in the form of service function chains (SFC), which need to be placed together. In order to achieve better than a very sub-optimal application deployment onto a distributed virtual cloud infrastructure, it is necessary to introduce sufficient functional granularity into the application structure to allow separate components to be provisioned and scaled independently. Application optimisation is essentially a mapping of a graph of application components and dependencies to the network of computing resources that delivers an optimal overall KPI target such as maximum latency or minimum throughput or maximum usage cost. The mapping is done subject to application-specific rules or constraints relating the individual resource requirements for components (Minimum/Maximum instance constraints) and their mutual co-hosting needs (Affinity/Anti-Affinity constraints).

The outputs of the application optimiser are treated as requests or recommendations for application scaling and placement, to be subsequently evaluated by the **Infrastructure Optimiser (4)** which augments the initial placement decision by taking into account the additional knowledge of the available physical infrastructures, the infrastructure policies of the infrastructure provider and specific committed Service Level Agreement (SLA) targets. This allows the infrastructure optimiser to retain full control of the infrastructure resources and to ultimately decide what application requests are enacted and how applications are orchestrated. The **Infrastructure Optimiser (4)** includes (1) Application Placement which optimally maps application components to virtual infrastructure resources to deliver an optimal overall target such as maximum power consumption, maximum operational cost, or specific committed Service Level Agreement (SLA) targets; (2) Infrastructure Optimisation to optimally utilise the physical infrastructure; and (3) Capacity Planning to perform what-if scenarios for additional physical infrastructure.

The Infrastructure Optimiser and Simulator use **Infrastructure Models (landscapes) (6)**. These models/landscapes present the physical and virtual structure, configuration, and topology of the known resources. The telemetry utilisation and performance statistics and the application KPI information are also needed for the Infrastructure Optimiser. Together these inputs form a consolidated infrastructure model that has the appropriate granularity tailored for the given use case thus making optimisation practicably achievable.

Application and Workload Models (7 and 9) describe the application components and their behaviours and dependencies and map the application components with their virtual resource requirements. The **Workload Models** describe the traffic flows through the application components. Both models are used by the workload predictor and application optimiser to forecast workloads and application components and recommend how these components should be placed on the network topology based on optimising the overall application KPIs. The application models describe applications as graphs of components with interdependencies and constraints in the form of graph links. The workload models describe the relationships between control and data plane traffic, between end-to-end latency and traffic, and between traffic and resource usage. They have been built based on the data analysis of historical trace and synthetic workload data using statistical and machine learning techniques.

In the **Application Optimiser (3)**, the traffic workloads are mapped to the application sub-components, and the propagation of workloads is modelled to account for the migratory capability of the components and the mobile nature of users. The Optimisers use **Load Distribution Models (6)** to account for this mobility of application components and the impact of component migration on application performance. They effectively model the traffic flows in the system and can predict the effect on workloads if application components are changed. They are based on the results of load balancing after a component migrates and on user models which drive component migration. These models are used by the optimisers to calculate the cost of component migration when selecting an optimisation option.

Load Translation Models (7) are used by the **Infrastructure Optimiser (4)** to map application configuration to physical infrastructure capacity. The optimiser correlates the virtual resources (VMs/Containers) to physical resources, and the physical resource utilisation with the application component KPIs (throughput, response time, availability, speed of service creation, and speed of service remediation). The translation provides a mapping of actual (specific in time) telemetry metrics of physical resource consumption (utilisation metrics) to application components workloads (i.e. the utilisation of resources by the components that are running on those physical machines). Effectively, this maps the application placement with the performance of components so placed.

The **User Models (9)** are based on an agent-based modelling of users, e.g. citizens navigating through a city and utilising mobile services.

It is possible to create models based on historical trace data and simulated synthetic data. In this case, **Simulators (5)** are a valuable tool for generating the user mobile behaviour and demand for application services as well as the corresponding traffic from the related cloud services.

1.5.1 Optimisation Process Flows

Process A: The **Application Optimiser (3)** is fed with appropriate output from the **Landscaper Component (1)** and **Monitoring Component (2)**, which represents the current resource capacity and utilisation, as well as the **Application Models**, which represent the application workload and performance targets. The **Application Optimiser's (3)** prediction engine produces a recommended deployment of components and outputs this to the **Infrastructure Optimiser (4)** for evaluation, and then to the **Orchestrator (11)** for orchestration. The **Application Optimiser (3)** can be subsequently triggered dynamically to handle variations in application workloads and user behaviours so that placement and autoscaling can take place. In its most proactive mode, the optimiser can create virtual resources, placing and autoscaling based on machine-learning models that are run against workload and user metrics in real-time.

Process B: The **Infrastructure Optimiser (4)** uses the output of the **Landscaper Component (1)** and **Monitoring Component (2)**, which represents the current resource capacity and utilisation, as well as the **Workload and Infrastructure Models** to optimise the utilisation of the physical hardware resources based on required Service Level targets and policies. The **Infrastructure Optimiser (4)** optimises the use of the physical resources taking energy, equipment, and operational costs into account as well as the plans and policies around physical resource utilisation. This is based on a logical model of the infrastructure, virtual and physical resources, and their utilisation mappings. The **Infrastructure Optimiser (4)** also needs to represent the mobile nature of workloads and the ability of application component migration to properly optimise the deployment. The Infrastructure Optimiser uses the **Simulator (5)** in a Human-in-the-Loop fashion, using the simulator to formulate deployment mapping selections and calibrating the optimiser's algorithmic process. The **Simulator (5)** validates the results of the optimisation and provides "what-if" scenario planning.

1.6 RECAP Building Blocks

While the previous section presents RECAP as a loosely integrated conceptual architecture, this section focuses on four high-level functional building blocks (subsystems) that encapsulate RECAP logic and provide the necessary functionality to realise the three operational modes discussed in Sect. 1.4. The respective building blocks are loosely coupled and are a frame for the RECAP architecture. The building blocks are themselves distributed so that the entire RECAP system represents a distributed architecture. The major functional building blocks (subsystems) are Infrastructure Modelling and Monitoring, Optimisation, Simulation and Planning, and Data Analytics and Machine Learning. Each of the blocks is discussed in-depth in the remaining chapters of the book.

1.6.1 Infrastructure Modelling and Monitoring

The old adage “garbage in, garbage out” particularly applies to making valued optimisation decisions. Thus, within RECAP’s Run-time Operation Mode, having an accurate understanding of the current state of applications and the underpinning infrastructure is of paramount importance. Furthermore, the long-term collection of accurate data is a key requirement for being able to apply meaningful data analytics and machine learning strategies (see Data Analytics Mode). Hereby the current state of application and infrastructure is represented by two complementary data sets, the infrastructure landscape and the infrastructure monitoring (telemetry) provided through the Landscaper Component and the Monitoring Component respectively. As discussed earlier, the Landscaper Component is tasked with providing physical and virtual infrastructure data as “a landscape” consisting of nodes and edges. In that landscape, nodes represent for instance physical servers, virtual machines, or application instances. In contrast, edges either represent mappings from applications to virtual resources and further to physical resources, or (network) connections between instances on the same abstraction layer. In short, the Landscaper Component identifies what type of infrastructure is available and where, while the Monitoring Component provides live data from that infrastructure. Both are essential for modelling and optimisation and are encompassed in a requisite distributed design.

As discussed in Sect. 1.5, the RECAP Monitoring Component collects telemetry-like data from physical infrastructure, virtual infrastructure, and applications; stores this data in a unified format; and ultimately provides the data in a consumer-specific format to other components in the wider RECAP system. Both the Landscaper Component and the Monitoring Component have been designed to operate on a per-location (data centre) basis. This helps in respecting administrative domains and, in the case of monitoring, reduces overall network traffic.

1.6.2 Optimisation

Optimisation goals in a multi-tenant distributed cloud-edge environment vary depending on the respective perspective. On the one hand, infrastructure optimisation has the goal to enforce a scheduling strategy that best reflects the intention of the infrastructure provider, e.g. to improve the utilisation of the available hardware or to save energy. On the other hand, application optimisation strategies try to find the best-possible configuration for an application deployment. Hence, the latter will increase the available compute capacity when high workload is expected. This, however, will only lead to satisfaction when the scheduling at the infrastructure level does not apply strategies that counteract these goals. Consequently, RECAP's optimisation subsystem realises a cooperative two-level optimisation framework, in which the optimisers at the two levels (application and infrastructure) interact in order to avoid conflicting scheduling decisions. Besides infrastructure-level and application-level optimisers, the subsystem further contains an optimisation orchestrator that mediates between the two levels. All entities in that subsystem consume monitoring data, application load data, and infrastructure data. The outputs of the optimisation algorithms in turn are optimisation steps that are then processed by the Enactor.

Figure 1.3 illustrates the dependencies between the major components of the optimisation subsystem. While there is just one Infrastructure Optimiser in a given installation, there may be multiple Application Optimisers, one per deployed application. Each of these is equipped with its own application-specific optimisation strategy and optimisation rules. The Infrastructure Optimiser in turn is equipped with provider-specific optimisation policies.

The Application Optimisers constantly receive the current status information from the Infrastructure and Modelling subsystems and, based on this information, estimate the future coming workload. Based on the current and predicted workload, each Application Optimiser suggests

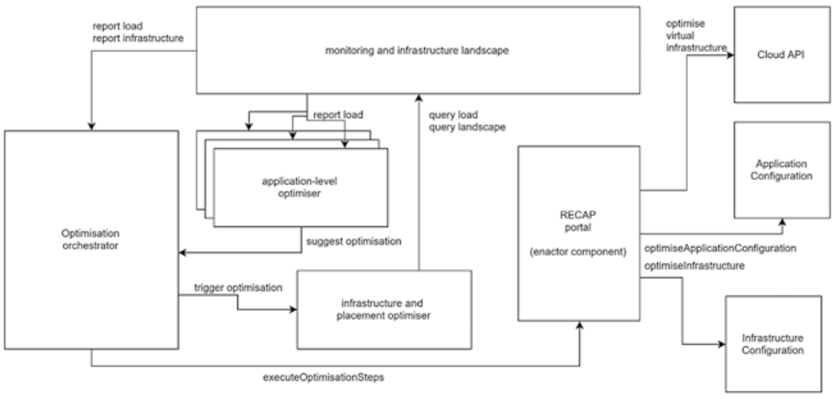
report load
report infrastructure
monitoring and infrastructure landscape
optimise virtual infrastructure
Cloud API
report load
query load
query landscape
application-level optimiser
Application Configuration
RECAP portal
(enactor component)
Optimisation orchestrator
suggest optimisation
optimiseApplicationConfiguration
optimiseInfrastructure
trigger optimisation
infrastructure and placement optimiser
Infrastructure Configuration
executeOptimisationSteps

Fig. 1.3 Component-oriented overview of the RECAP optimisation subsystem

optimisation steps for its particular application. These suggestions are fed to the optimisation orchestrator, which, based on the input received, triggers the infrastructure optimiser that then decides on whether these operations are feasible and also the mapping between application components (bundled in virtual machines or containers) and physical resources. Application Optimisation and Infrastructure Optimisation are presented in detail in Chaps. 3 and 4 respectively.

1.6.3 Simulation and Planning

Figure 1.4 illustrates the core architecture of the RECAP Simulation Framework. It consists of an API Component, a Simulation Manager, and Simulation Engines. The API component serves as an entry point for users, be they human or other RECAP components, or external parties. The API Component offers an interface for controlling simulation runs. In particular, it is used for submitting experiments and retrieving simulation results from these runs. From the API Component, the experiment data is forwarded to the Simulation Manager, which, in turn, checks model validity and submits models to an appropriate Simulation Engine. The RECAP Simulation Framework currently supports two simulation engines that address different use case requirements. First, the discrete event simulator (DES), based on CloudSim, is targeted towards the simulation of

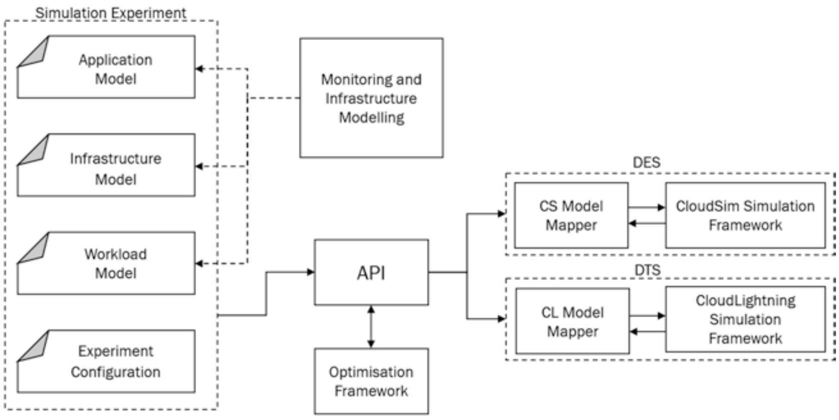

Fig. 1.4 High-level overview on RECAP simulation framework

large-scale cloud-computing infrastructures, data centres, virtual machines, and virtual machine components. It is tailored for fine-grained and detailed simulations. On the other hand, the discrete time simulator (DTS), based on the CloudLightning Simulator, is well suited for large-scale simulations that need to run at speed and whose execution time is bounded.

The primary input to a RECAP simulation is a simulation experiment comprising instances of the application model, the infrastructure model, the workload model, and in addition, an experiment configuration. All of these models are represented in the very same way for both simulation engines. Once the input has been validated by the Simulation Manager, it has to be transformed to the simulation engine-specific format. This is done by the Model Mapper components shown in Fig. 1.4.

1.6.4 Data Analytics and Machine Learning

The Data Analytics and Machine Learning subsystems make use of the data collected by Landscaper Component and the Monitoring Component. The primary goal of this functional block is to distil statistical properties and patterns from load traces. Previously, this activity would be undertaken within an engineering team; however, due to the massive volume of data involved, this can no longer be easily undertaken by humans. As such, the Data Analytics and Machine Learning subsystem operates in a separate processing pipeline that is decoupled from the Optimisation and the

Simulation and Planning subsystems. The steps for analytics cannot be fully automated and require the involvement of a data analyst. Despite this decoupled processing, the results of the analysis do flow back into the RECAP optimisation cycles, either through insights gained by the data analyst performing the analytics (generally in the case of descriptive and/or visual statistical analysis) or through codified models integrated into other RECAP components as libraries or micro-services (more applicable in the machine learning case).

The overall approach of the Data Analytics and Machine Learning subsystem is shown in Fig. 1.5. First, a data scientist retrieves data collected from the Monitoring Component. Then, they perform pre-processing followed by the actual analysis and/or training on the pre-processed data set. Both steps take place in iterations so that the analyst may go back and perform different types of analysis, but they may also go back and perform different types of pre-processing. Finally, as a last step, the results are exported as mathematical models, as codified models, as a library, or as an instantiable service. Due to the decoupled nature of the offline processing, requirements towards the API of the actual data analytics components are less strict than for other RECAP components. The only exception to that rule is the format of the data retrieved from the Monitoring Component. After the data has been fetched, pre-processing and all other steps performed by the data analyst are open and not fixed by APIs. Also, the integration of results into, for example, the optimisation algorithm needs to be defined on a case-to-case basis.

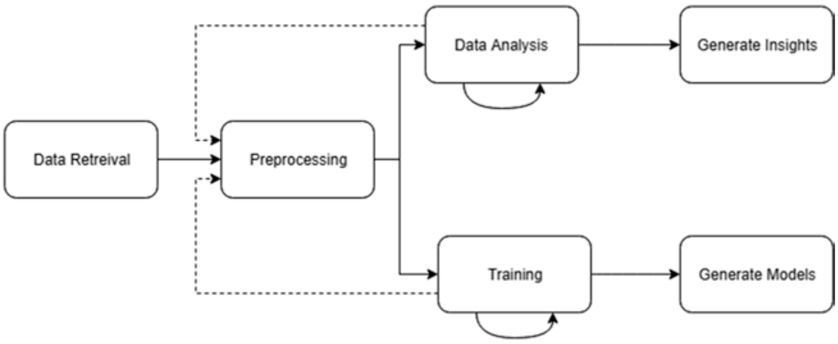

Fig. 1.5 The RECAP approach to retrieve data, analyse it, and export the resulting models to other RECAP components

1.7 Mapping Functional Blocks to Operational Modes

This section describes how the functional building blocks introduced in the previous section interact to deliver the operational modes introduced earlier.

1.7.1 Run-time Operation Mode

The Run-time Operation Mode (see Fig. 1.6) manages a set of applications spread out over a distributed physical and virtual infrastructure such as an IaaS infrastructure with different geo-distributed locations. Based on the user behaviour, and the current and predicted load in the system, the run-time cycle identifies improvements to the current live system on both infrastructure and application level and enacts them by executing optimisation steps. For that purpose, the Run-time Operation Mode makes use of the infrastructure modelling and monitoring subsystem and the optimisation subsystem. Depending on the type of system to optimise, the optimiser may be configured with or without the Infrastructure Optimiser. Not using it yields classical infrastructure unaware application-level optimisation. Internally, the optimisers may make use of additional components generated by the Data Analytics and Machine Learning subsystem. The optimisation plans produced by the optimisers are consumed by the Enactor that interacts with application, physical infrastructure, and virtual infrastructure to enact the optimisations.

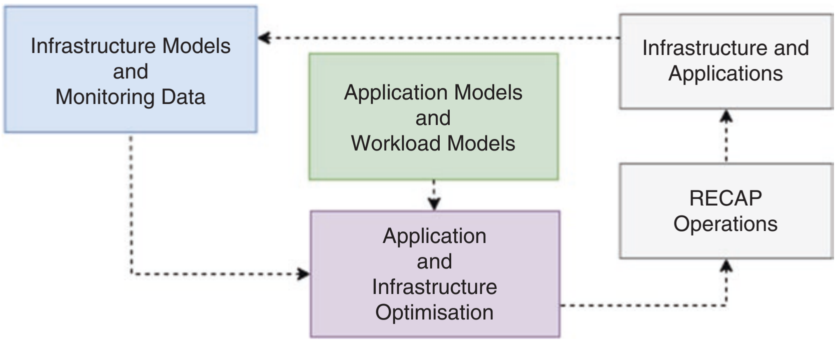
Infrastructure Models and Monitoring Data
Application Models and Workload Models
Infrastructure and Applications
RECAP Operations
Application and Infrastructure Optimisation

Fig. 1.6 Run-time loop of RECAP

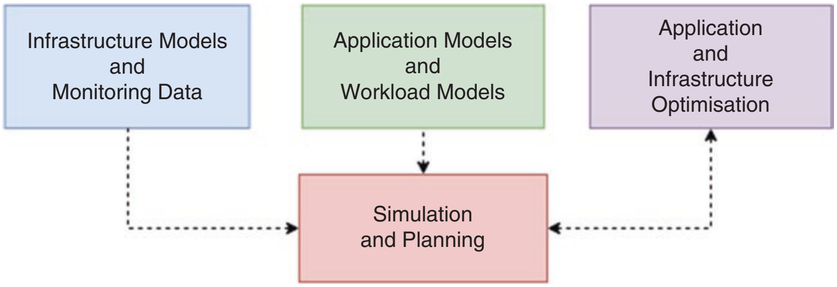
Infrastructure Models and Monitoring Data
Application Models and Workload Models
Application and Infrastructure Optimisation
Simulation and Planning

Fig. 1.7 High-level overview on simulation interaction

1.7.2 Simulation and Planning Mode

As discussed in Sect. 1.4, the purpose of the Simulation Mode is to perform two kinds of tasks. Firstly, it helps users and operators conducting experiments about the performance of their infrastructure and applications running therein. This includes the interplay of different types of applications but also the choice of configuration patterns for the Run-time Operation Mode. Secondly, it can be used as a tool for operators to estimate future needs with respect to the amount and type of hardware. Both of these tasks require interaction with the Infrastructure Optimiser.

Figure 1.7 shows how the Simulation Mode is embedded in the wider RECAP architecture. It supports (but does not mandate) importing real-world telemetry and infrastructure landscape data that serve as input to the simulation. These data are combined with the user models, workload models, and load translation models to define a simulation (experiment). Alternatively, parts of the input, or even all of the input, to a simulation can be manually constructed by the user. For helping operators improve their hardware choice, the Simulation Component supports an optimisation-oriented approach that iterates over different simulation configurations and picks the best-possible one for a given application mix and usage scenario.

1.7.3 Data Analytics Mode

The Data Analytics Mode enables statistical evaluation and analysis, as well as applying state-of-the-art machine learning techniques to the data collected by the Monitoring Component. This mode envisions a data

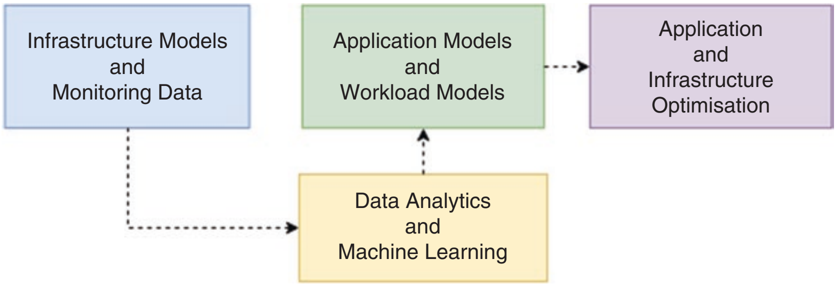
Infrastructure Models and Monitoring Data
Application Models and Workload Models
Application and Infrastructure Optimisation
Data Analytics and Machine Learning

Fig. 1.8 High-level overview on data analytics subsystems

scientist performing many of the steps. Hence, while a certain degree of automation can be achieved in the process, it still requires human interaction, guidance, and input. Figure 1.8 summarises the interaction of the Data Analytics and Machine Learning subsystem with the other RECAP subsystems. It relies on the monitoring subsystems to export metrics as bulk in a normalised manner. This data is then analysed, and the resulting insights and models provided to other RECAP components. In particular, the optimisation components are users of these models, for instance, for the purpose of workload prediction.

1.8 Conclusion

The chapter introduces the challenges of reliable capacity provisioning across the cloud-to-edge continuum. The scale and complexity across this continuum is so complex; it is no longer realistic for IT teams to cost-effectively foresee and manage manually cloud and network operations on a detailed level due to high levels of dynamism and dependencies in the system. This chapter, and the book as a whole, presents a high-level conceptual overview of RECAP—an architectural innovation to support reliable capacity provisioning for distributed clouds— and some of the major design concepts informing its design, namely separation of concerns, model-centricism, modular design, and machine learning and artificial intelligence for IT operations.

The remainder of this book is organised around the four functional building blocks outlined in Sect. 1.6 above. Chapter 2 describes the Data Analytics and Machine Learning subsystem, followed by Application

Optimisation (Chap. 3), Infrastructure Optimisation (Chap. 4), and Simulation and Planning (Chap. 5). The book ends in Chap. 6 with four case studies each illustrating an implementation of one or more RECAP subsystems. The first case study presents a case study on infrastructure optimisation for a 5G network use case. The second case study explores application optimisation for virtual content distribution networks (vCDN) on a large Tier 1 network operator. The third case study presents how data analytics and simulation components, within RECAP, can be used by a small-to-medium-sized enterprise (SME) for cloud capacity planning. The final case study looks at how RECAP components can be embedded in an IoT platform to reduce costs and increase quality of service.

References

Aissaoui, Nabila, Mohammed Aissaoui, and Youssef Jabri. 2013. For a Cloud Computing based Open Source E-Health Solution for Emerging Countries. *International Journal of Computer Applications* 84 (11): 1–6.

Brown, A.W. 2004. Model Driven Architecture: Principles and Practice. *Software and Systems Modeling* 3 (4): 314–327.

Cisco. 2013a. Embracing the Internet of Everything To Capture Your Share of $14.4 Trillion. https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-economy-insights.pdf.

———. 2013b. Internet of Everything: A $4.6 Trillion Public-Sector Opportunity. https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-public-sector-vas-white-paper.pdf.

Cardoso, J. 2019. *The Application of Deep Learning to Intelligent Cloud Operation*. Paper presented at Huawei Planet-scale Intelligent Cloud Operations Summit, Dublin, Ireland.

Dang, Y., Q. Lin, and P. Huang. 2019. *AIOps: Real-World Challenges and Research Innovations*. Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, 4–5. IEEE Press.

Gillen, A., C. Arend, M. Ballou, L. Carvalho, A. Dayaratna, S. Elliot, M. Fleming, M. Iriya, P. Marston, J. Mercer, G. Mironescu, J. Thomson, and C. Zhang. 2018. *IDC FutureScape: Worldwide Developer and DevOps 2019 Predictions*. IDC.

Haller, S., S. Karnouskos, and C. Schroth. 2009. The Internet of Things in an Enterprise Context. In *Future Internet Symposium*, 14–28. Berlin, Heidelberg: Springer.

IBM. AIOps, IBM Cloud Education. 2019. https://www.ibm.com/cloud/learn/aiops.

Iorga, M., L. Feldman, R. Barton, M.J. Martin, N.S. Goren, and C. Mahmoudi. 2018. Fog Computing Conceptual Model. Special Publication No. 500-325. NIST.

Kleppe, A.G., J. Warmer, J.B. Warmer, and W. Bast. 2003. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Professional.

Lynn, T. 2018. Addressing the Complexity of HPC in the Cloud: Emergence, Self-Organisation, Self-Management, and the Separation of Concerns. In *Heterogeneity, High Performance Computing, Self-Organization and the Cloud*, 1–30. Cham: Palgrave Macmillan.

Lynn, T., P. Rosati, and P. Endo. 2018. *Towards the Intelligent Internet of Everything: Observations on Multi-disciplinary Challenges in Intelligent Systems.* Proceedings of the Research Coloquio Doctorados: Tecnología, Ciencia y Cultura: una visión global.

Masood, A., and A. Hashmi. 2019. AIOps: Predictive Analytics & Machine Learning in Operations. In *Cognitive Computing Recipes*, 359–382. Berkeley, CA: Apress.

Papazoglou, M.P. 2012. Cloud Blueprints for Integrating and Managing Cloud Federations. In *Software Service and Application Engineering*, 102–119. Berlin: Springer.

Prasad, P., and C. Rich. 2018. *Market Guide for AIOps Platforms.* Gartner.