CHAPTER 3

Application Optimisation: Workload Prediction and Autonomous Autoscaling of Distributed Cloud Applications

Per-Olov Östberg, Thang Le Duc, Paolo Casari, Rafael García Leiva, Antonio Fernández Anta, and Jörg Domaschka

Abstract Optimisation of (the configuration and deployment of) distributed cloud applications is a complex problem that requires understanding factors such as infrastructure and application topologies, workload arrival and propagation patterns, and the predictability and variations of user

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

T. Le Duc
Tieto Product Development Services, Umeå, Sweden
e-mail: thang.leduc@tieto.com

P. Casari • R. García Leiva • A. Fernández Anta
IMDEA Networks Institute, Madrid, Spain
e-mail: paolo.casari@imdea.org; rafael.garcia@imdea.org; antonio.fernandez@imdea.org

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_3

behaviour. This chapter outlines the RECAP approach to application optimisation and presents its framework for joint modelling of applications, workloads, and the propagation of workloads in applications and networks. The interaction of the models and algorithms developed is described and presented along with the tools that build on them. Contributions in modelling, characterisation, and autoscaling of applications, as well as prediction and generation of workloads, are presented and discussed in the context of optimisation of distributed cloud applications operating in complex heterogeneous resource environments.

Keywords Resource provisioning • Workload modelling • Workload prediction • Workload propagation modelling • Application optimisation • Autoscaling • Distributed cloud

3.1 Introduction

Key to the RECAP approach for application optimisation is *application autoscaling*, the dynamic adjustment of the amount and type of resource capacity allocated to software components at run-time (Le Duc and Östberg 2018). In principle, this type of scaling can be done *reactively*—by dynamically adjusting the amount of capacity to match observed changes in load patterns, or *proactively*—by operating on predicted future load values. Naturally, proactive autoscaling requires the ability to predict or forecast future values of the workloads of applications, systems, and components.

In this chapter, we summarise the RECAP application optimisation system. Following the problem formulation, we discuss the RECAP approach to application modelling, workload modelling, and the models used for application optimisation (application and workload, including how models are constructed and trained), the optimisation approach, and the implementation and evaluation of the optimisation models. The application optimisation approach outlined in this chapter exploits the

J. Domaschka
Institute of Information Resource Management, Ulm University, Ulm, Germany
e-mail: joerg.domaschka@uni-ulm.de

advanced techniques for characterising, predicting, and classifying workloads presented in Chap. 2 to construct proactive autoscaling systems.

3.2 Problem Formulation

The problem of optimising the deployment and configuration of applications hosted in geo-distributed resource environments can conceptually be viewed as a graph-to-graph mapping problem. As discussed in previous chapters, RECAP models distribute applications as graphs of components where graph nodes denote application components and the edges of the graph represent the communication paths and dependencies among components. Similarly, infrastructure systems can also be represented as graphs where the nodes correspond to resource sites and the edges model the interconnecting network links of site-connecting networks. The optimisation problem then is to find the optimal mapping of application nodes to infrastructure nodes. This mapping is subject to constraints that reflect requirements on the application level (e.g. minimal acceptable Quality of Service (QoS) for applications, or co-hosting restrictions of I/O-intensive processes).

A graph-based formulation of the mapping problem facilitates reasoning on the scaling of both application and infrastructure systems. Application scaling can on the one hand regulate the optimal number of instances to deploy for specific component-associated services (horizontal autoscaling) and on the other hand define how much resource capacity to allocate to a particular application on a specific site (vertical autoscaling). Furthermore, application scaling can be global, when the entire application is scaled, or local, when only individual components are scaled independently. Hybrid approaches are also possible where individual parts of applications or infrastructures are treated differently. In that respect, the RECAP optimisation approach includes the concept of application and infrastructure resource zones—subsets of application and infrastructure graphs that need to be treated as a group.

Based on studies of the technical trade-offs that influence optimality in scaling and placement, e.g, power-performance trade-offs and sensors and actuators that can used in optimisation of systems (Krzywda et al. 2018), we define four types of constraints on application and infrastructure placement and scaling:

1. Affinity constraints—these specify co-hosting or pinning of components;
2. Anti-affinity constraints—these prohibit co-hosting or pinning;
3. Minimal number of instance constraints—these specify lower bounds of the number of instances to scale or the amount of capacity to allocate to application nodes; and,
4. Maximal number of instance constraints—these specify upper bounds of the number of instances to scale or the amount of capacity to allocate to application nodes.

3.3 Optimisation Framework

As well as infrastructure optimisation, RECAP provides an optimisation framework that enables the development and execution of optimisation tasks at application level. The core of the framework is an optimisation engine that consists of multiple modellers and optimisers. More specifically, the modellers produce dedicated models for each supported application, including workload models, load transition models, user models, and application models. These are used to provide a complete view of the application. In addition, they are used as input for optimisers to solve optimisation problems related to autoscaling. Depending on the type of optimiser, it can deal with a wide range of optimisation problems related to the placement, deployment, autoscaling, and remediation of applications.

For creating the respective modellers and optimisers, RECAP uses methodological framework that entails three optimisation levels for the deployment and management of applications in heterogeneous edge-cloud environments, see Fig. 3.1. The figure illustrates the three-level process that constitutes the optimisation methodology: the first level of optimisation is the simplest and aims at the placement of applications throughout the edge-cloud environments under fixed network, application, and quality-of-service requirements/constraints. Optimisation solutions created by this level of optimisation can be used for long-term resource planning as well as initialisations for further optimisation levels.

In the next level of optimisation, the variations of workload and user behaviours are taken into account for dynamic application placement and autoscaling. The workload model and user models are used to estimate the demand of resources of individual application components over time.

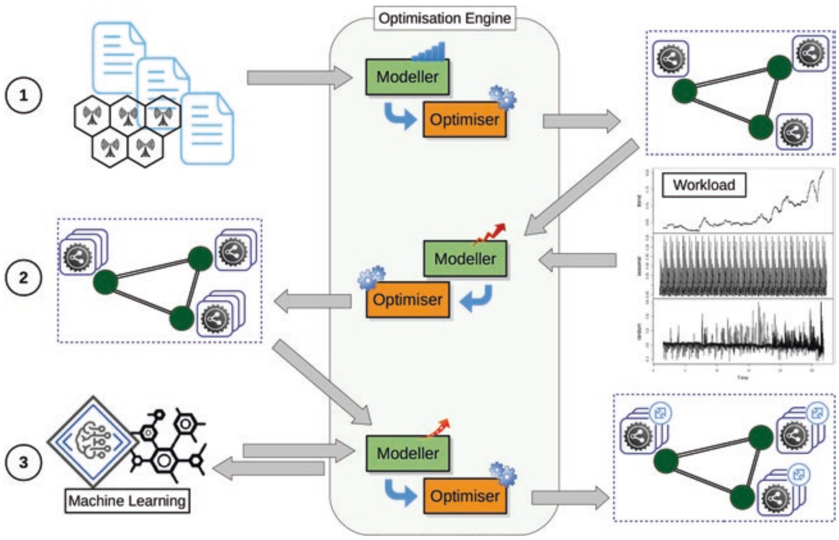

Fig. 3.1 A stratified approach to application optimisation iteratively building on three optimisation building blocks—(1) classic optimisation on static data, (2) application adaptation to variations in workloads and resource availability, (3) joint autoscaling and optimisation in multi-tenancy scenarios using machine learning (adapted from Le Duc et al. (2019))

With the estimation results, resources are allocated for each application component. Furthermore, based on predictions, workload can be redirected or migrated in order to maintain the load balance within applications.

The most advanced level of optimisation aims at proactive resource provisioning for applications. For that purpose, the RECAP Application Optimiser applies workload predictors that make use of workload models discussed in previous chapters. Machine learning is adopted to improve the understanding of both workload and application behaviours. This means more fine-grained models are derived and models can be refined and improved over time. Using these models, predictions can be performed more accurately to support load balancing, autoscaling, and remediation in a proactive manner.

3.4 Application Modelling

This section addresses the application modelling from an optimisation perspective. It first introduces the basic requirements and fundamental assumptions behind the RECAP application optimisation framework. Then, Sect. 3.4.2 introduces the modelling framework.

3.4.1 Application Characteristics and Modelling Requirements

A key enabling technology of cloud and edge computing is virtualisation. Abstraction of physical resources through containers and virtual machines (VMs) enables consolidation of compute capacity and resources (e.g. processors, storage, and networks) into software-defined infrastructures (SDIs). Such abstraction provides means for automation, scaling, and optimisation of resource allocations and resilience to variations in workload intensity.

The services that are running in an edge/cloud environment differ from the standard centralised cloud deployments. The difference is dictated in the way applications are used, by the infrastructure topology, and the infrastructure availability at the network edges. Geo-distributed infrastructures enable services (applications) to be brought closer to users, which increases the data exchange speeds and results in faster content delivery to consumers. However, the distributed nature of edge infrastructure comes with the limitation of physical space and associated limited hardware deployment capabilities. Different types of applications require different hardware profiles to process user requests. Varying hardware properties across distributed infrastructure stacks also require a distributed application architecture that is modular enough to adapt to the available edge/fog/cloud infrastructure horizontally and vertically.

As part of the application mode, the application topology depicts application components that can be deployed as separate entities (containers or VMs), and network link connections between them. A simple example of such a topology would be a deployment of an application that has a front-end web server and a database as two components. The web server can be deployed as a separate component on a separate VM or node, or even on a different datacentre than the database component, but both of them should have a bidirectional data flow connection for data exchange.

Most web applications serve different types of user requests, and to do so, different amounts of resources are needed depending on the requests made. For example, a request of streaming video from a content distribution network (CDN) would differ from a request to upload video to the service. The first request can assume a data download; however, the second request requires data upload, resampling, encoding, and other types of content analysis and optimisations.

Elasticity is one of the major benefits of virtualised resources. Elasticity makes it possible to maintain application QoS by dynamically scaling application components based on workload intensity (provided that the application architecture caters for it). To take advantage of elastic properties, application components can be managed by a load balancer that can spawn extra parallel application components and redirect user requests to evenly distribute the load.

Another vital characteristic in a distributed system is the geolocation and content variability within the same type of application. For example, a database or a CDN can be distributed across the edge infrastructure, where some instances will contain the same type of data, but some will not. Depending on data needed to serve a user request, the request needs to be routed to an application instance that has these data. Such scenario requires additional intelligence in the workload orchestration. From a RECAP perspective, it also requires that the application model have a notion of data content available within the application component.

All the aforementioned characteristics should be captured and reflected in the constructed application models. Specifically, the models should provide the means to estimate computation, memory, and storage capacity requirements of each components, as well as to present and calculate the mapping of the applications and application components on the underlying infrastructure. Moreover, they should help identify the type of traffic or content delivered to the users at different locations, and estimate the service delay for user requests.

3.4.2 Application Modelling Framework

The characteristics of the edge-cloud infrastructure and applications result in high complexity when it comes to application modelling. In particular, this modelling always has to be done in an application-specific manner. As such, it is necessary to have a comprehensive understanding of typical systems, models, and modelling tools from theoretical and practical perspective. This section outlines the strategy adopted in the RECAP methodology to perform application modelling for each system in its scope.

Firstly, a literature survey generated a universe of general and common architectures of distributed applications including client-server architecture, cloudlet, service-oriented architecture, and micro-services. Secondly, desk research allowed a comparison of the previously identified architectures with those of real large-scale distributed systems/applications and related industrial technology standards used for the realisation of operational systems. These systems include, for instance, the multi-tier caching system of Akamai's CDN, the operational architecture of Nextflix's CDN, the architectural framework of ETSI NFV, and Service Function Chaining (SFC) (Halpern and Pignataro 2015) amongst others.

Thirdly, following the literature survey and desk research, we explored mathematical models and tools widely adopted in computing science, such as queuing theory, graph theory, and control theory. These have been used to model components and the interaction among components and are used together to model applications. For instance, a model may be a combination of queuing theory, graph theory, and control theory as follows:

- Queuing theory is used to model the processing logic of a component, e.g. a VNF, a vCache, or a function node (database, data aggregator, balancer).
- Graph theory is used to model the network topology, service function chain, communications between components.
- Control theory is used to model the control logic of dispatchers, balancers, or orchestrators.

Fourthly, we decomposed the application into isolated components and analysed the components in detail in order to understand the nature of each component as well as how they communicate to each other in the whole application. In addition to simplifying the modelling task, this helps to identify how components impact each other and to identify the bottlenecks within the entire topology. Once all components have been modelled, these sub-models can be integrated to form a complete application model.

In order to keep the modelling effort manageable, it is further necessary to identify in advance the factors or metrics that should be captured in the models in accordance with the requirements of the application. That is, an application model always needs to capture business-specific constraints and goals and cannot be constructed on a technical level alone.

3.5 Workload Modelling

Besides an application model, an application optimisation engine further requires predictions of the amount of work the application shall be able to process overall or per zone. This enables proactive optimisation approaches, where readjustments do not happen on a best-effort basis upon changing workload conditions, but rather anticipate future workload levels and scale the applications accordingly.

As discussed in previous chapters, workload analysis and modelling focuses on techniques efficiently applied to time series data collected from both real production systems and emulating systems. Before being analysed, original time-series data have gone through a pre-processing step (gap filling, smoothing, resampling, etc.). Next, the data flows through the two-stage process of workload analysis and workload modelling.

Workload analysis is composed of two main tasks: workload decomposition (which splits a time-series into the trend, the seasonality, and the random factors of the workload data) and workload characterisation (which aims to extract workload features as an input and driver of workload models). Such workload characterisation is performed by an Exploratory Data Analysis (EDA), which is typically a manual step.

With this understanding of key features (metrics) of the workload, it is possible to derive which aspects should be considered in modelling, and how to effectively construct and evaluate appropriate workload models. To perform the modelling task, we adopt different categories of techniques in the RECAP methodology:

- Autoregressive integrated moving average (ARIMA) and seasonal ARIMA (SARIMA) models are chosen based on the analysis of the autocorrelation functions, partial autocorrelation functions, and tests of stationarity (for example, Dickey-Fuller tests).
- The family of autoregressive conditional heteroskedasticity models (ARCH, GARCH, NGARCH, etc.) is suited when the assumption is fulfilled that the variance of the time series is not constant but still a function of previous variances.
- Recursive neural networks and deep neural networks can be used to find more complex interactions between past and future requests.
- Long short-term memory neural networks (LSTM) shall be used when the aim is to detect if long past requests have a predictive value for future requests.

The models obtained by workload analyses are expected to provide forecasting capability. That is, based on them, it shall be possible to generate future workload predictions at different periods of time and different intervals, as required by the application under analysis.

The predictions can be validated through various metrics so as to identify the best one for each case, i.e. the one with the smallest errors. Depending on the application, different models are required for different prediction purposes (online and offline predictions). More specifically, models used for offline prediction are expected to provide high accuracy so that the results can be used for long-term planning.

In contrast, models used for online prediction additionally focus on low execution time besides the accuracy in order to offer short-term predictions in a timely manner. For the latter, it is necessary to evaluate the execution time in forecasting of each model and to balance between the error level and the execution time. Once selected, a model becomes the core of the online predictor component for that application within RECAP.

3.6 Model-based Application Optimisation

The key challenges of efficiently deploying distributed applications in edge and fog computing environments involve determining the optimal locations and allocations of resource capacity. This is complicated by the inherently varying load conditions of distributed infrastructures. Due to the complexity and distinct mathematical formulations of the problems for different applications, they are typically treated separately, often with the solution of one as input to the other.

3.6.1 Application Autoscaling

In this section, we focus on application load balancing and distribution, which can be seen as a special case of holistic application autoscaling (i.e. self-scaling of application capacity allocations) given specific workload arrival patterns and component placements. To avoid confusion, we use different terms for different types of load balancing in distributed, multi-tier applications. We denote (1) load balancing between multiple instances of one single application component as load *balancing*, and (2) load balancing within the entire application to balance the loads distributed to different application components as load *distribution*. This is addressed by load transition models that capture how workloads flow through applications in workload models. Principally, RECAP assumes that a RECAP-enabled application is designed such that it is capable of making use of more instances (scale out) or more resources (scale up).

The RECAP application models describe applications as networks/graphs of components with interdependencies and constraints in the form of network links, quality-of-service requirements, and communication patterns. Application components are split into front-end and back-end layers (modelling load balancing within components and management of component functionality respectively) that can be autoscaled independently. Using RECAP application and transition models, application workload arrival patterns can be used to derive how load propagates through distributed applications, and how the resulting component workloads impact the resources (including networks) where component services are deployed. Using prediction algorithms allows to provide improved performance and proactivity in autoscaling without changing the autoscaling algorithms themselves.

Depending on the application, arrival patterns can be measured from instrumented applications or infrastructure, or derived from simulation models build on the user (including mobility) models. The value of such models lies in the increased understanding of user and system behaviour, but also in their potential use for prediction of workload fluctuations in predictive scaling algorithms. Overall, RECAP applies the following types of autoscaling algorithms:

- Local reactive scaling algorithms (similar to the autoscaling algorithms used in Kubernetes) are used to individually scale component front-ends and back-ends. They apply varied degrees of downscaling inertia in order to reduce the amount of false positives.
- Global reactive scaling algorithms that predictively evaluate the performance of individual component autoscalers, and selectively apply those that maximise application objective functions, and used to control back-ends.
- Global proactive algorithms that use short time-frame simulation techniques to evaluate application performance for heuristically selected subsets for autoscaling actions. This class of algorithms shows the greatest autoscaling performance but is also significantly slower and resource demanding. This limits its applicability in large-scale systems.

3.6.2 Migration Techniques and Infrastructure Planning and Provisioning

In addition to the algorithms described above, which target application autoscaling in static deployment scenarios i.e. for deployments that do not change during or from autoscaling, RECAP also provides tools and techniques for evaluating and performing migrations of service components at run-time. In order to incorporate this functionality in the autoscaling-oriented perspective of application optimisation, such migrations can be formulated to be part of the autoscaling problem by scaling the amount of service instances for a specific component at a specific location in the space of *[0; n]* (when there is no mobility of services and *n* is a maximum instance count). Alternatively, it can be treated as a separate placement problem that does not include autoscaling (beyond considering autoscaling limits in the placement process) that is solved independently (either before or after the autoscaling).

RECAP explores and exploits both approaches to develop a flexible framework for application migration optimisation. The basic building blocks of this framework are application autoscaling algorithms, a set of heuristic functions that evaluate alternative deployment scenarios (under specified autoscaling settings), and an in-situ simulation framework that models application communication at message level within the application models. The simulation framework essentially uses application and (predicted) workload models to simulate how workloads will be processed under current application deployment and autoscaling settings. Heuristic functions are used to identify components and/or resource sites that underperform or for some other reason are candidates for reconfiguration, and alternative configuration settings are speculatively evaluated by the simulator to identify the reconfiguration actions most likely to improve application performance the most (according to application heuristics and KPIs).

Currently three types of heuristic functions are used for deployment cost evaluations:

- Local evaluation functions that build linear combinations of QoS or KPI metrics for individual component services or resources,
- Aggregate local evaluation functions that define statistical aggregates of local evaluation functions for sets of components (i.e. subsets of application components) or resources (e.g. regions of resource sites), and

- Global evaluation functions that operate on application and infrastructure models to aggregate QoS or KPI evaluation functions for all components of an application or large sets of resources.

Evaluation functions are defined as mathematical constructs and can be composed to develop utility functions that combine evaluation of both applications and infrastructure resources. Using the simulation techniques, recommendations for how to change application deployments (in single- and multi-tenancy scenarios) and autoscaling constraints can be derived from nominal size estimations of component placements and infrastructure capacities, or conversely component nominal sizes can be included in the decisions on the admission of scaled or migrated service instances from autoscaling constraints.

3.6.3 Workload Propagation Model

Workload propagation models describe how workload is propagating through an application (i.e. between application components), and how a fluctuation of workload at a certain component impacts the other ones. Such a model can be constructed using workload data collected from all the network nodes/locations in every system. Unfortunately, due to the size and complexity of large-scale systems, exhaustively collecting such data is extremely challenging (Le Duc et al. 2019). Therefore, the mechanisms for workload generation and/or propagation are needed that enable the production of workload data for all network nodes using data traces collected only from a subset of nodes.

The five workload diffusion algorithms to address this problem are classified into non-hierarchical and hierarchical diffusion as follows:

1. Non-hierarchical diffusion—these algorithms perform load propagation within networks according to a discrete spatial model of how heat is diffused in materials in physics or chemistry. They are applicable for controlling data exchange and the workload of synchronisation tasks that are carried out by neighbouring network nodes. This also can be extended to cover some general cases of unstructured peer-to-peer overlays or ad-hoc mobile networks.

2. Hierarchical diffusion—these algorithms rely on a hierarchical network model which slices the network into different layers. In this case, the workload propagation is directed through network layers from end users down to the core network layers or using predetermined routing paths destined to the dedicated service nodes. This diffusion technique is applicable for core broadband networks and CDNs.

3.6.4 *Approach and Realisation*

Non-hierarchical diffusion algorithms include population-based, location-based, and bandwidth-based algorithms, while hierarchical diffusion ones include hierarchy-based and network-routing-based algorithms. This section briefly describes the five algorithms including the assumptions, key inputs, the main flow, and properties (see Table 3.1). Further details about the calculations or formulae used in each task/step of the algorithms can be found in RECAP Deliverable 6.2 which can be downloaded at the RECAP website. The algorithms were tested with workload data traces collected as time series at three inner-core nodes of BT's CDN; the representative metric of the workload in this use case is the traffic generated at caches when serving user requests.

3.7 The RECAP Application Optimisation Platform

Elasticity is a key function for addressing the problem of reliable resource provisioning for edge-cloud applications as it ensures the reliability and robustness of the applications regardless of the non-linear fluctuation of the workload over time (Östberg et al. 2017; Le Duc et al. 2019). One of the key techniques adopted in RECAP to address elasticity and remediation is autoscaling. By flexibly adjusting the amount of resources allocated for applications and/or the number of application instances or components, autoscaling enables applications to adapt to workload fluctuations, which helps prevent the applications from becoming unresponsive or terminating.

Table 3.1 Summary of diffusion algorithms

Diffusion algorithm	*Assumptions*	*Key inputs*	*Description*
Population-based Location-based Bandwidth-based	• Non-hierarchical network/application topologies (e.g. telecom networks, P2P applications) • Homo-geneous user behaviour	User distribution in the network Geographical node locations Bandwidth capacity of network links	• Iterative refinement algorithms (similar to heat diffusion and spring relaxation equations) • Repeatedly solve state equations to distribute workload to neighbours until the overall load distribution approaches equilibrium • Algorithms highly parallelisable
Hierarchy-based	• Hierarchical network/ application topologies (e.g. broadband networks, CDN application) • Full mesh network of the inner-core nodes • Multiple shortest path routing	• Network hierarchy • Bandwidth capacity of network links • User distribution in the network	• Hierarchy-based user aggregation to identify the aggregated number of users at every node/location based on bandwidth capacity of neighbouring links • Backward workload extrapolation to collect the workload measurements from every node to the inner-code nodes • Inner-core workload extrapolation to extrapolate workload at every inner-core node (if needed) • Workload propagation to distribute the workload from inner-code nodes to every node in the network
Network-routing-based	• Homo-geneous user behaviour	• Network hierarchy • Bandwidth capacity of network links • User distribution in the network • A set of service (inner-core) nodes	• Routing path discovery to identify (shortest) routing paths from client-clusters to the service nodes • Network-routing-based user aggregation (using routing paths) • Backward workload extrapolation • Workload propagation

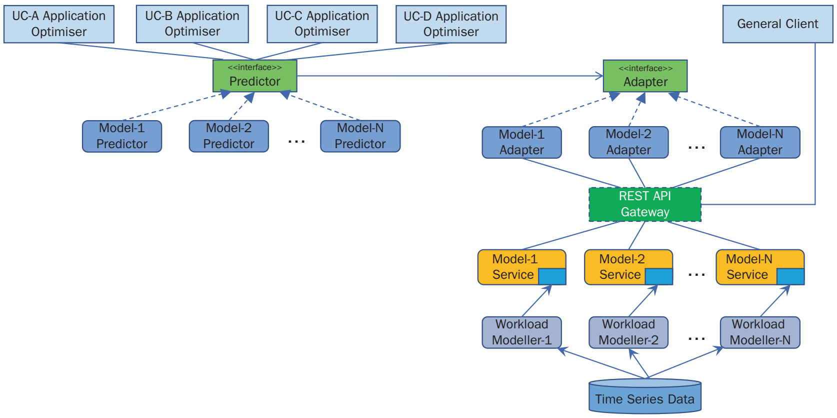

Fig. 3.2 A platform for the integration of predictors and modellers

The RECAP system model assumes that applications are dynamically distributed, and their behaviours are considerably difficult to predict and model. Moreover, each application component may be subject to different workloads, and an understanding of workload characteristics is required in order to autoscale efficiently. Therefore, workload analysis, modelling, and prediction based on time series analysis become the vital factor for the efficiency of our solutions, and especially so for the proactive schemes.

For application optimisation, RECAP has created a platform for the integration of application optimisers, workload modellers, and workload predictors, as shown in Fig. 3.2. The figure also shows how the optimisers (autoscalers) utilise the predictors when needed. Predictors, in turn, are fed by workload modellers.

By adopting different techniques, for example regression or machine learning, a workload modeller can construct multiple workload models. For flexibility in system integration, the modellers can be implemented using various technologies. The workload models are then wrapped and exported as a microservice using a REST API Gateway. On top of model services, a set of adapters are built to provide a unification layer. While workload models are constructed using historical workload data, they can be updated continuously at run-time by the workload modeller. Predictors in the platform make use of the adapters in order to access the available models and to make their predictions.

On top of this platform, robust and efficient approaches for autoscaling are constructed based on the results of workload modelling and prediction. Optimisers encapsulating optimisation algorithms and application-specific constraints make use of prediction for proactive optimisation. Note that an optimiser can call multiple predictors that access different models constructed using different techniques. This also implies that predictors are developed for every constructed model. Such implementation enables the capability of extension when a new optimiser (of a new application) or new model (using new techniques) is added to the system.

3.8 Conclusion

This chapter introduced the RECAP Application Optimisation approach and framework and outlined its the constituent building blocks. The interaction of the RECAP models and algorithms developed was further discussed. The RECAP Application Optimisation Framework addresses application placement and autoscaling, and provides models and tools for prediction, optimisation, and evaluation of the performance of distributed cloud applications deployed in heterogeneous resource environments.

References

Le Duc, Thang, and Per-Olov Östberg. 2018. *Application, Workload, and Infrastructure Models for Virtualized Content Delivery Networks Deployed in Edge Computing Environments*. Proceedings of the IEEE International Conference on Computer Communication and Networks (ICCCN), 1–7.

Le Duc, Thang, Rafael García Leiva, Paolo Casari, and Per-Olov Östberg. 2019. Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey. *ACM Computing Surveys (ACM)* 52 (5): 39. https://doi.org/10.1145/3341145.

Halpern, J., and C. Pignataro. 2015. *Service Function Chaining (SFC) Architecture*. RFC 7665.

Krzywda, Jakub, Ahmed Ali-Eldin, Trevor E. Carlson, Per-Olov Östberg, and Erik Elmroth. 2018. Power-Performance Tradeoffs in Data Center Servers: DVFS, CPU Pinning, Horizontal, and Vertical Scaling. *Future Generation Computer Systems* 81: 114–128.

Östberg, Per-Olov, James Byrne, Paolo Casari, Philip Eardley, Antonio Fernández Anta, Johan Forsman, et al. 2017. *Reliable Capacity Provisioning for Distributed Cloud/Edge/Fog Computing Applications*. European Conference on Networks and Communications (EuCNC).