Proactive Resource Orchestration Framework for Cloud/Fog Platform

Somnath Mazumdar[†], Thomas Dreibholz[‡]
[†]*Department of Digitalization, Copenhagen Business School, Solbjerg Plads 3, 2000 Frederiksberg, Denmark*
[‡]*Centre for Resilient Networks and Applications, Simula Metropolitan, Pilestredet 52, 0167 Oslo, Norway*
[†]sma.digi@cbs.dk, [‡]dreibh@simula.no

***Abstract*—Cloud computing makes complex processing an off-premise activity by offering software- and hardware-based services using standard security protocols over the Internet. It has been seen that the cloud is not *ideal* for latency-sensitive applications. Thanks to the current growth of network communication and infrastructure, fog adds a computing resource delegation model between the user and the cloud. Fog aims to improve latency-sensitive applications support. Here, we propose one unified, proactive resource orchestration framework from a cloud/fog service provider perspective. The framework consists of a predictor and a resource allocator module. Users subscribe to these resources to execute their applications. The framework is modular and does not require application-specific information. A service provider can customise each module. We have presented the framework prototype by showing each module's simulated performance results using the parameters of our cloud/fog research testbed.**

***Index Terms*—Cloud, Fog, Neural Network, Prediction, Wavelet**

I. Introduction

Cloud has become a popular computing platform, primarily due to the readily available resources at scale and a lower price than hosting on-premise. Fog[1] brings a new resource delegation-based business model between the user and the cloud [1], [2]. Fog can better support latency-sensitive applications while being resourcefully backed by the cloud. Fog devices have limited computing and storage capacities, while the cloud is still a viable solution to offload applications for more resources. Fog's resource capacity can be improved by employing upcoming hardware packed with smaller, faster, more energy-efficient transistors.

Offloading the compute-intensive portion or the complete application[2] to nearby fog can improve the applications' performance. The latency-tolerant part of an application can be hosted in the cloud and transferred back to fog when needed. Overall, fog can optimise network bandwidth and support heterogeneous hardware and services.

Fog still needs to be a mature distributed resource delegation model and suffers from resource management and data security issues [2]. With one unified resource orchestration framework, efficiently offloading tasks to fog may be manageable while managing users' requirements and limited capacity [3]. Allocation can be improved by exploiting historical resource usage patterns. Delegating resources in advance can improve the quality-of-service (QoS) [4] and also the quality-of-experience (QoE) of an application.

Server virtualisation using a hypervisor is the backbone of currently distributed computing platforms. Hypervisor-based virtualisation hosts virtual machines (VMs) on top of the host operating system. Container-based virtualisation is a lightweight way to host a virtual environment. Cloud uses both, while fog primarily supports containers. Resource orchestration is a classical problem in distributed computing and a very popular research topic [2], [3], [5]. Many scholars propose multiple computation offloading strategies to optimise latency, energy cost, and architectures to improve the QoS [6]. However, except [7], we have yet to find much work that proposes a unified resource allocation framework combining forecasting and resource allocation strategies.

To fill this gap, here, we propose a Proactive Resource Orchestration Framework, called PROF, for cloud/fog platforms. In particular, we are answering the following research question: *How to build a unified, proactive resource orchestration framework based on a fast prediction and a light resource allocation policy?* PROF aims to support partial and complete task offloading. Our contributions are:

- A unified resource orchestration framework that offers seamless resource management in a cloud/fog platform. A cloud/fog service provider should employ it, and users can subscribe to such resources for their applications.
- We explain the architecture and its core components. We also argue the applicability of such a unified framework.
- Next, we develop a hybrid prediction model and compare multiple lightweight task allocation policies by simulating using parameters from a cloud/fog research testbed.
- Finally, we show that such a framework can improve resource utilisation and task processing time. Overall, it can be deployed using best practices from software engineering.

II. Background

The proposed framework should be used to optimise the overall resource orchestration of a cloud/fog platform. Here, we will briefly discuss two primary modules of the framework.

[1]In this paper, we have followed the NIST definition of cloud and fog computing.

[2]We define a task as a single unit of work of an application, while an application may consist of multiple tasks.

A. Prediction

Time-series models are popular and efficient for predicting regular, historical time-series data. Machine learning (ML) models can outperform time-series models when data contains many features, but at an increased computational cost. Below, we briefly discuss the employed time series models and one simple ML model for prediction.

1) Time-Series-Based Models: The autoregressive (AR) model is a stochastic process for modelling univariate time series. It can be analysed with standard linear least squares and the moving average (MA), known as ARMA.

AutoRegressive Integrated Moving Average (ARIMA) model generalises the ARMA class that incorporates non-stationary univariate time series data. It includes the lag of stationary series, the forecast errors, and the integrated (I) component, which allows the series to be differentiated. In $\texttt{ARIMA}(p,d,q)$, p is the number of AR terms, d represents non-seasonal differences, and q is the number of MA terms. The differencing parameter d takes integer values and can be scaled up or down, provided the sum of AR coefficients and the sum of MA coefficients are close to one.

Seasonal ARIMA (SARIMA) can exploit the seasonality of time series. It can be written as $\texttt{ARIMA}(p,d,q)\ (P,D,Q)_h$, where $(P,D,Q)_h$ identifies the order of seasonality and h is the number of periods. The seasonal part consists of back-shift operators of the seasonal data sets. We have used wavelet decomposition to improve the prediction accuracy of the used models, where the wavelet output is used as input for the prediction models. We call these models hybrid models, and such models can outperform the standard time-series prediction models [8].

2) Wavelet: It converts time series into high- and low-frequency components using a finite length window or *wavelets*. The attractiveness of wavelet transformation lies in its decomposition properties for time-scale localisation. It has an optimal trade-off between time and frequency. It is free from any assumption related to stationary characteristics. Wavelet can analyse non-stationary systems. The input of a wavelet model is a signal with its temporal aspect. The mother wavelet function transforms the given input signal into its different components, which helps to understand the data. To decompose the given series, we have used a maximal overlap discrete wavelet transform with *Daubechies wavelets* [9] as the mother function, with a window length of eight. It is worth noting that Daubechies does not suffer from the slow convergence issue. Four levels of the original times series' additive decomposition are used for the ARIMA and SARIMA models.

3) ML Model: Neural network (NN) supports the nonlinear relationship between a variable and its predictors (input layers). The prediction is made via a nonlinear combination of input layers. The existence of a hidden layer makes it either linear or nonlinear. The selection of weights of the inputs is non-trivial. Weights can be selected using a learning algorithm. NNs are ordered by the number of hidden units and the amount of weight decay.

Employed NN (in this paper) is a single-hidden layer feed-forward network (SLFN) combined with a wavelet model. The primary motivation behind using the SLFN is that input weights may not be needed to adjust manually as the model can approximate any continuous function. It has also been seen that with n hidden neurons and randomly chosen input weights, SLFN can learn n distinct observations with small error [10]. No hidden layer is also possible, if there are skip-layer units. Setting zero weight can be beneficial when the data is very dynamic. We have developed a wavelet and SLFN-based hybrid predictor for our prediction scenario.

B. Allocator

We have used Reliable Server Pooling (RSerPool) for the allocation purpose [11]. RSerPool is an IETF standard for a server pooling approach. It provides a simple application-independent and open-source framework. It is lightweight and aims to serve latency-sensitive applications. Furthermore, it supports large distributed resources to achieve high resilience and is suitable for low-capacity devices.

RSerPool can manage combined cloud/fog resource pools using suitable policies and configurations [12]. It also supports simple resource selection policies such as random (RAND) and round-robin (RR) as baseline models. Resource pool-related information, such as the current workload, can be used to select resources with the lowest current workload. RAND, RR, and even least used (LU)-like policies are unsuitable, because they do not differentiate between the cloud and fog resource types. As a solution, fog resources can be prioritised using the priority least used (PLU) policy, trying to use least-loaded fog resources, otherwise, use cloud resources. Load states need updates, which may take time, particularly for remote cloud resources. Therefore, the priority least used with degradation (PLUD) combines PLU with a load degradation counter. Each time a resource is selected, its degradation counter is increased to compensate for the inaccurate load information. This counter is reset on load state update (i.e. the latest data from the resource). Finally, PLU and PLUD are combined with a delay penalty factor (DPF), which also considers the current network round-trip times between the components of PLU-DPF and PLUD-DPF. It may provide further performance benefits for small tasks.

III. Proposed Framework

Tasks can move back and forward based on available resources, and employing a PROF-like framework will ease such management. PROF is designed to be *proactive* and modular. It should be operated by a single service provider who manages a cloud and fog platform. PROF uses a lightweight hybrid prediction model and a resource allocation policy. The framework works in two sequential stages (refer to the blue-shaded region of Figure 1). The first stage is focused on *predicting* the number of upcoming resource requests, mainly VMs and containers[3]. Combining a cloud and fog platform in

[3]The framework is tested with VM request data while container-related data is unavailable.

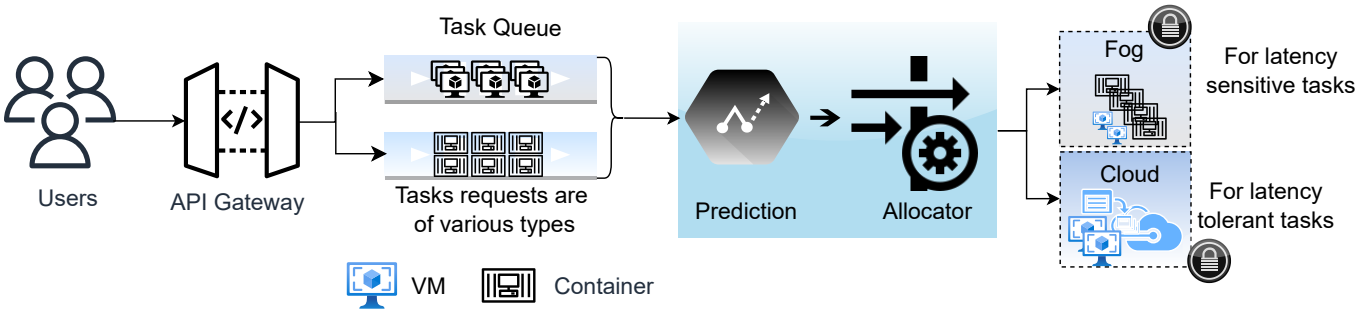

Figure 1: Representation of proactive resource orchestration framework.

a unified way reduces the information silos, leading to better platform resource management. Predicting upcoming applications' resource requirements brings the proactive feature. Using the forecasting data, the allocator allocates resources in advance to reduce task waiting and queuing time, which can further improve the application's QoE/QoS.

A. Design Consideration

We have considered four primary design goals while developing this framework. They are briefly described below:

- Low Overhead: PROF implementation overhead should be low to offer lower latency. Hence, we assume that a single service provider will manage both the cloud and fog platforms, and employ RSerPool, which is suitable for devices with minimal resources.
- High Throughput: PROF should handle many requests at high speed. We have used a lighter predictor and allocation framework to achieve that.
- Ease of Management: PROF is based on an IETF standard for a server pooling approach, state-of-the-art model wavelet and NN. Using such models eases the deployment process.
- Interoperability: RSerPool is an application-independent framework. Besides, open-source-based application development and RESTful APIs-based approach also help to achieve the goal.

B. Prediction

From Figure 1, we see that users submit their tasks using RESTful APIs. Applied resource filtering logic stores the application in a container- or a VM-specific queue. The queue length is dynamic and can be set to a fixed unit of time (valid for our case). Creating a dedicated task queue helps to parallelise the prediction process. Each prediction model can run in parallel inside containers. The allocator uses predicted results for proactive scheduling. Based on the resource selection policy's decision, tasks will be moved to fog or cloud.

It has been seen that the collected trace from a production environment is smoother than random data [13]. The prediction helps to identify resource requirements of tasks ahead of time. For prediction, the predictor uses three application attributes. They are *i)* application arrival time, *ii)* application required resource type, and *iii)* requested quantity. As an output, the predictor forecasts the total VM or container count based on each type. Generally, time-series-based prediction models are less complex and faster than ML models. Applying a simpler prediction model can reduce overheads significantly, but in some cases, efficiency can be reduced. Parallel prediction of task queues can speed up the overall allocation process, provided the allocation policy is also faster.

We have compared three prediction models: ARIMA, SARIMA and SLFN. We also built their hybrid version using wavelet decomposition. We found hybrid models are more accurate in prediction and reduce the total model execution time. There might be failed predictions due to the randomness (spikes) of the users' requests. PROF will allocate extra resources in such cases, increasing application waiting time. If the model fails to perform and forecasts to reserve more or fewer resources, the predictor should be tuned further or replaced. Here, we have focused on the latency-sensitive application; tasks are primarily assigned to the fog resources.

C. Allocator

The predicted result will be used as input to the allocation process. The service provider can customise the prediction and allocation modules based on their preference and platform insights. For VMs, RSerPool framework-based allocator (refer to Subsection II-B) expects the VM type and required quantity, while for containers, it expects CPU and memory requirements. It is important noting that the RSerPool framework allows the definition of more policies using application-specific features when necessary.

IV. Evaluation

We have reported the results of prediction and allocation modules, while simulating using the parameters collected from our cloud/fog research testbed. For analysis, we have used a trace of incoming VM requests over seven days with six minutes intervals, collected from one small private cloud service provider. For privacy reasons, the trace is a univariate time series containing a time stamp and the corresponding interval's total number of VM requests.

Table I: Error measures of all six models.

	1-Hour Prediction		2-Hours Prediction	
Model	**RMSE**	**MAPE**	**RMSE**	**MAPE**
ARIMA	0.4974432	7.987505	0.5367095	8.318853
SARIMA	0.5297760	8.274079	0.4678396	6.146552
SLFN	0.4930488	6.944248	0.4963941	6.280406
Wavelet_ARIMA	0.4095534	4.077404	0.5396644	8.131191
Wavelet_SARIMA	0.3916113	5.600092	0.339822	4.567944
Wavelet_SLFN	0.00784473	0.08739554	0.01041639	0.1215156

A. Prediction Result

We have performed an out-sample, multi-step prediction using two popular forecasting accuracy measures: root mean squared error (RMSE) and mean absolute percentage error (MAPE). RMSE is a scale-dependent forecasting error measure based on squared errors, while MAPE is a scale-independent error measure. We have six models, where ARIMA, SARIMA and SLFN are the standard models, and their hybrid forms use wavelet decomposition. Results[4] focusing on one-hour and two-hours prediction scenarios are presented in Table I.

1) Standard Models' Performance: We fitted an `ARIMA(1,1,2)` model for prediction on the seven-day data set. The ARIMA model offers better prediction accuracy than `SARIMA(2,1,2)(1,1,1)` model. However, ARIMA misses the trend for two-hour out-sample prediction and performs worse than the SARIMA model. Unlike ARIMA, the fitted `SARIMA(2,1,2)(1,1,1)` with $h = 1$ model performs better for two-hours prediction. SARIMA takes more time (≈6 seconds) to complete both cases' forecasts. Next, we have employed SLFN for forecasting. We fitted an average of 20 networks, each of which is a 32-16-1 network with 545 weights for both cases. Compared to other time series models, SLFN performs better but with a longer execution time (≈70 seconds).

2) Hybrid Models' Performance: We developed a hybrid model using wavelet decomposition to improve the prediction ability of the above models. Wavelets decompose the data into high- and low-frequency components; each part is forecasted separately and combined. We employ a Dabuchies least asymmetric wavelet of length eight and perform a four-level decomposition. It means four wavelet details and one wavelet smooth for each data set. Table I shows that the hybrid Wavelet_ARIMA model is better for one-hour. For short prediction, the Wavelet_ARIMA variant achieves ≈4% (MAPE error measure) improvement compared to the vanilla ARIMA model. Wavelet decomposition improves the SARIMA model for both scenarios. It can be seen that wavelet can improve the prediction by 1.6% (MAPE error measure).

Finally, compared to other models, the hybrid SLNF-based approach achieves more than 6% improvement in prediction accuracy for both cases. We found that four levels of wavelet decomposition and a wavelet length of eight offer the best result for SLFN. It is because a wavelet can decompose its input signal with temporal aspects into different components. Apart from that, weights in SLFN can approximate any continuous function. For both cases, the execution time of the Wavelet_SLFN model is ≈29 seconds, which is lower than the standard model. The wavelet part is fast, and execution time is around ≈22 milliseconds. Hence, building a hybrid model improves the model performance and reduces the total execution time for SARIMA and SLFN.

[4]We used a server with an Intel processor (model i7-8565U with 1.80 GHz clock speed) with eight cores, 16 GiB memory and running Ubuntu 22.04.2 LTS.

B. Allocation Results

Based on the service provider's trace, we have simulated using RSPSIM [11, Chapter 6] and CalcAppProtocol [11, Section 8.3], which is a generic application where servers provide a specific capacity (in work units/second). Requests consume this capacity, and a multi-tasking server processes these requests. Clients generate several work units with an inter-request time (in seconds). They are queued and sequentially distributed to and processed by servers.

The simulated environment consists of 14 cloud servers, providing a capacity of 18,000 work units/s, handling up to two simultaneous requests. That is, each task is processed with at least 9000 work units/s. The one-way delay between the user and cloud server is 30 milliseconds (based on intra-European communication) and 300 milliseconds (based on communication between Europe and China). The one-way delay between clients and fog units is between 5 milliseconds and 15 milliseconds. The result is based on simulations using our cloud/fog research testbed parameters. 50 clients issue requests of 9,720,000 work units on average. With an average inter-request time of 2719 seconds, server utilisation is around 70.9%. For the cloud/fog setup, four cloud servers are replaced by dedicated fog units, with a 50% increased capacity (i.e. 27,000 work units/second) and up to 3 simultaneous requests.

1) Performance of Policies without Prediction: In Table II, we have reported the resource utilisation (in % scale for both, fog and cloud) together with queuing time, startup time and task processing time in seconds. Task handling time is the sum of all three times. We compare all policies mentioned in Subsection II-B. We have used RAND and RR as baseline models for a fair comparison. RAND and RR *just* distribute the tasks, leading to a longer processing time. Sometimes, it results in rejection on a fully-loaded server with a following retry, leading to increased startup time. As a result, RAND and RR offer the highest task-handling time per request among all policies.

LU makes better choices based on load information. However, it does not distinguish between fog and cloud resources and selects mainly cloud servers instead fog servers. PLU is the best scheduling policy, and PLUD is the second-best one. Although, there is very little difference in this scenario. Both use mostly fog resources, leading to better task-handling times. DPF-based policies select the nearer servers, but there is a little benefit as the tasks are long-running.

2) Performance of Policies with Prediction: Table III and Table IV show the pros and cons (↑ denotes the deterioration

Table II: Average server utilisation in percentage and average request times in seconds.

Policy	Fog Util.	Cloud Util.	Queuing Time	Startup Time	Processing Time	Handling Time
LU	47.89	67.19	0.00	0.06	411.99	412.05
PLU	61.91	13.60	0.00	0.02	370.52	370.54
PLUD	62.00	14.02	0.00	0.02	371.39	371.41
PLUD-DPF	62.05	13.65	0.00	0.02	371.86	371.88
PLU-DPF	62.01	13.86	0.00	0.02	372.27	372.29
RAND	50.08	58.86	1.25	0.08	630.62	631.95
RR	47.09	69.60	0.15	0.06	434.48	434.69

Table III: Model comparison on one hour prediction relative to original data.

Policy	Fog Util. (%)	Cloud Util. (%)	Processing Time (seconds)	Handling Time (seconds)
LU	↓ 5.84	↓ 8.23	↓ 10.08	↓ 10.09
PLU	↓ 9.74	↓ 1.40	↓ 0.04	↓ 0.04
PLUD	↓ 6.70	↓ 2.24	↑ 0.66	↑ 0.66
PLUD-DPF	↓ 6.81	↓ 2.43	↑ 13.06	↑ 13.05
PLU-DPF	↓ 7.25	↓ 3.96	↓ 9.26	↓ 9.27
RAND	↓ 3.97	↓ 7.45	↓ 50.60	↓ 50.60
RR	↓ 7.28	↓ 6.76	↓ 7.60	↓ 7.61

Table IV: Model comparison on two hours prediction relative to original data.

Policy	Fog Util. (%)	Cloud Util. (%)	Processing Time (seconds)	Handling Time (seconds)
LU	↓ 0.08	↑ 0.29	↑ 1.44	↑ 1.44
PLU	↓ 0.65	↓ 1.12	↓ 5.71	↓ 5.72
PLUD	↑ 0.52	↑ 0.42	↑ 2.14	↑ 2.14
PLUD-DPF	↑ 0.12	↑ 1.11	↑ 4.32	↑ 4.32
PLU-DPF	↑ 0.26	↓ 0.56	↓ 0.87	↓ 0.87
RAND	↑ 0.17	↓ 0.61	↑ 3.09	↑ 3.09
RR	↓ 0.10	↑ 0.14	↓ 2.00	↓ 2.02

and ↓ shows the improvements) related to resource utilisation and task processing time of all scheduling policies while running on predicted data sets. To compare the models, we have used one-hour and two-hours trace data, together with the predicted output of the Wavelet_SLFN model.

Overall, the two-hours prediction results in Table IV fit better. The one-hour results in Table III are short, and the PLUD model performs better for task processing scenarios, while resource utilisation is not good. Moving to Table IV, we can see that PLUD improves resource utilisation again, and PLUD-DPF optimises the task processing times.

Determining what performance can be expected from and achieved by underlying public cloud service providers is also interesting. Here, we aim to showcase the feasibility of such a unified resource orchestration model. We do not want to claim that the PLUD policy is good. Service providers should tune/choose an applicable policy. Therefore, we briefly provide network performance results of public cloud service providers together with our cloud/fog platform.

C. Network Performance

In Table V, we have reported the cloud network connection test result. All tests present average download speed and access latency of VM, cloud storage and content delivery network. We also have reported the VM latency data of our cloud/fog research testbed. Our cloud platform's speed is the *minimum* Internet service provider (ISP) subscription speed, and the maximum speed is 1 Gbit/s. Our fog platform is based on a 4G network and does not have a dedicated storage and content delivery network. The VM access latency of our cloud platform is comparable with cloud service providers.

We also compare cloud platforms based on the media content, because it represents a content distribution over the Internet use case scenario (an ideal QoE-based use case). We had chosen three typical media sizes : 1 GiB for standard definition (SD), 3 GiB for high definition (HD), and 7 GiB for ultra-high definition (4K). Due to resource constraints, the download speed of our cloud/fog testbed is significantly lower. Such download speed can reduce the application's performance. Our platform data shows the possibility of implementing our proposed unified resource orchestration framework into a commercial platform. It is worth noting that we have used such platform-related parameters for all simulations.

D. Applicability

PROF aims to become vendor-neutral and portable to multi-cloud platforms. PROF will not store user resource usage data following the user privacy policy. Two upcoming QoE-based applications, such as content distribution over the Internet and mixed-reality applications, will benefit from such frameworks, where no significant amount of time will be consumed for task-resource mapping. Currently, large content distribution platforms do not have fog-based services. To offer better QoS/QoE, such platforms collaborate with ISPs. They use an ISP's existing infrastructure to store popular media content near the users. The Open Connect program from Netflix[5]

[5]urlhttps://openconnect.netflix.com/en_gb/

Table V: Access latency (in milliseconds) of cloud service providers and content downloading time (minutes:seconds) for three different media file formats.

Service	Region	Compute	Storage	Content Delivery Network	Download Speed Mbit/s	SD	HD	4K
Amazon Cloud	Stockholm	15.00	23.00	32.08	836.01	0:10	0:31	1:13
Microsoft Cloud	Oslo	38.00	42.33	22.34	476.59	0:18	0:55	2:09
Google Cloud	Helsinki	25.08	97.08	17.50	461.51	0:19	0:57	2:13
NorNet Fog	distributed over Norway	59.17	–	–	32.49	4:31	13:34	31:41
NorNet Cloud	distributed over Norway	27.00	–	–	$\geq$100.00	1:28	4:24	10:17

is an example of such a scenario. For mixed reality-based applications, the issue is primarily network bandwidth to offer optimum QoE. Moving the primary QoE-related services to fog will improve the individual user's perceived experiences.

V. Technical Challenges

In the following, we will discuss four primary technical challenges focusing on content/media distribution and mixed-reality applications. They are *i)* security, *ii)* resource capacity, *iii)* network support for better QoE, and *iv)* standardisation for better cloud/fog platform adoption.

A. Security

We should protect applications and data, while running in the public cloud and also while in transit. Data encryption is essential for collaborative content distribution platforms. In this direction, researchers are working on *trust-based computing*. While protecting the code and data, trust can be achieved from hardware and software. Homomorphic Encryption (HE) is a mechanism that preserves the message structure and supports certain operations without decrypting the data. However, the efficiency and practicality of HE are known challenges. Cryptographic approaches to trust-based computing, such as fully homomorphic encryption and secure multi-party computations, are progressing slowly. Cloud-based sandboxing can protect user code from malware at the data transport level by applying Internet-level security protocols such as transport layer security (TLS) [14].

Application-level security isolates the network, I/O and memory environment for safe execution, but increases overheads. Applications are running inside the VMs, and there are events related to VM-based attacks. Current virtualisation technologies also suffer from new security vulnerabilities, such as the denial-of-service of xenstored (Xen Project) via a malicious user's out-of-memory allocation [15]. Besides that, certified container images contain at least one high or critical vulnerability, while community container images are the most vulnerable. Such vulnerabilities come from popular scripting languages, such as JavaScript and Python [16]. Some countermeasures could be based on cryptographic models, custom security layers and sandboxing. We also need to improve container engines' security by ensuring that containers process their task inside themselves and third parties do not misuse them. One solution could be to use a secure container runtime such as Kata containers[6], but with extra overhead.

[6]Kata containers: https://katacontainers.io.

B. Resource Capacity

Fog devices are resource-constrained. Hardware *disaggregation* may help the fog units to scale [17]. Disaggregation creates several consolidated single-resource pools and allows the composition of logical compute platforms with flexible and scalable resources. For instance, providing more CPU/GPU-based compute resources than memory can better service the image-processing-related tasks (ideal for mixed reality-based use cases), while stacking more memory than compute units can support content streaming type applications. The system disaggregation approach can help to customise the fog based on the application requirements. Unfortunately, the disaggregated server approach has yet to find more attention from the community, but we should rethink improving user QoS/QoE by employing resource disaggregation. Networking infrastructure and software support is needed to use disaggregated resources at the fog and cloud levels.

C. Bandwidth Requirements

Internet users are geographically distributed and use different connections, such as Asymmetric Digital Subscriber Lines and optical fibre-based connections. Service providers can place caches next to an ISP's point-of-presence to improve the QoE (for a content distribution application) while generating extra network management overheads. A congested network will yield lower video quality, due to packet loss. Content providers invest significantly large resources to get more customers. Currently, the issue is not only about the subscriber listing, but also about offering them higher QoE.

In an experiment, it has been seen that public cloud platforms are compatible towards applications that can tolerate delays of up to 100 ms [18]. For better QoE, network congestion plays an important role, mainly for latency-sensitive use cases such as mixed-reality applications, which require a minimum frame rate of 60 Hz. Therefore, the maximum delay between the process and the rendered frame should be limited to 16.6 ms [19]. Such features can be achieved with innovative network technology and unifying cloud/fog infrastructure.

D. Standardisation

Standardising service-oriented abstract models is crucial. It allows heterogeneous devices to communicate via unified abstraction for seamless multiple service offerings. Standardising resource delegation models can counter scalability challenges. Generally, when the standardisation process completes, the system interoperability improves. For instance, codec standardisation is highly required for streaming media content.

There is a need to develop a fast and efficient image and video codec technologies similar to the AOMedia Video 1 coding format. Similarly, codec standardisation is also needed to stream immersive media files for mixed-reality applications. On a lower level, protocol standardisation is also needed.

VI. Related Work

We refer readers to the surveys that focus on orchestration in fog [2], in cloud [5], [6] for computation offloading processes and [3] for research status and existing challenges.

Resource Prediction. In [20], the authors propose an ML-based prediction model to forecast the future CPU and memory workload demands. [8] shows how adding Kalman filter and wavelet decomposition to the ARMA models' popular variants can increase the forecasting accuracy while predicting the CPU, memory and network demand, while [4] shows how workload prediction can improve cloud-based applications' QoS.

Resource Orchestration. [21] presents a resource management algorithm to place ML applications onto edge accelerators while respecting their latency constraints. [22] proposes another resource orchestrating framework for delegating resources to multiple cloud/edge platform tenants. Similarly, [23] proposes a resource orchestrator for serverless architectures to achieve the targeted time latency, and [24] discusses the needs for container orchestration in fog and their required capabilities for IoT applications. In [25], another workload orchestration framework is proposed to find a trade-off between the requirements of infrastructure owners and the applications. Furthermore, [26] presents an orchestration architecture for fog, focusing only on the functional aspects such as time and quality of the resource allocation process.

VII. Conclusion and Future Work

Thanks to the networking domain's progress, fog will soon become a subscription-based resource delegation model to users. A modular unified resource orchestration framework can effectively deploy different applications on the cloud/fog platform. PROF aims to be one such framework, consisting of two modules: prediction and allocator. In this paper, we have developed a hybrid prediction model based on wavelet decomposition and SLFN. We showed that the hybrid predictor outperforms (in terms of speed and accuracy measures) other state-of-the-art time series models. We also used one light resource allocation framework called RSerPool, which accepts the output of the prediction model as input and offers multiple resource selection policies. However, the cloud/fog service provider can customise further or replace prediction and allocation policies as per their platform insights. Our experiments showed the feasibility of implementing such a unified cloud/fog resource orchestration framework. We also advocate for a lighter and faster framework to support both QoS/QoE-based applications. In future work, we plan to add latency-tolerant application-specific features to allocation policies and look into data sets management issues.

References

[1] M. Iorga *et al.*, "The NIST Definition of Fog Computing," National Institute of Standards and Technology, Tech. Rep., 2017.

[2] B. Costa *et al.*, "Orchestration in Fog Computing: A Comprehensive Survey," *ACM Computing Surveys*, vol. 55, pp. 1–34, 2022.

[3] Q. Duan *et al.*, "Convergence of Networking and Cloud/Edge Computing: Status, Challenges, and Opportunities," *IEEE Network*, vol. 34, pp. 148–155, 2020.

[4] R. N. Calheiros *et al.*, "Workload Prediction using ARIMA Model and its Impact on Cloud Applications' QoS," *IEEE Transactions on Cloud Computing*, vol. 3, pp. 449–458, 2014.

[5] T. L. Duc *et al.*, "Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey," *ACM Computing Surveys*, vol. 52, pp. 1–39, 2019.

[6] H. Lin *et al.*, "A Survey on Computation Offloading Modeling for Edge Computing," *Journal of Network and Computer Applications*, vol. 169, pp. 1–25, 2020.

[7] S. Mazumdar *et al.*, *Adaptive Resource Allocation for Load Balancing in Cloud*. Springer, 2017.

[8] S. Mazumdar and A. S. Kumar, "Forecasting Data Center Resource Usage: An Experimental Comparison with Time-Series Methods," in *8th Int'l Conf. on Soft Computing and Pattern Recognition*. Springer, 2018, pp. 151–165.

[9] I. Daubechies, *Ten Lectures on Wavelets*. SIAM, 1992, vol. 61.

[10] G.-B. Huang, "Learning Capability and Storage Capacity of Two-Hidden-Layer Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 14, pp. 274–281, 2003.

[11] T. Dreibholz, "Reliable Server Pooling – Evaluation, Optimization and Extension of a Novel IETF Architecture," Ph.D. dissertation, University of Duisburg-Essen, Germany, Mar. 2007.

[12] T. Dreibholz *et al.*, "Towards a Lightweight Task Scheduling Framework for Cloud and Edge Platform," *Internet of Things*, vol. 21, 2023.

[13] M. Schroeder, *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*. Courier Corporation, 2009.

[14] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[15] NVD NIST, "CVE-2022-42311 Detail," 2022, accessed on 2022-11-07. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2022-42311

[16] K. Wist *et al.*, "Vulnerability Analysis of 2500 Docker Hub Images," in *Advances in Security, Networks, and Internet of Things*. Springer, 2021, pp. 307–327.

[17] S. Han *et al.*, "Network Support for Resource Disaggregation in Next-Generation Datacenters," in *Proc. of 12th ACM Workshop on Hot Topics in Networks*, 2013.

[18] I. Pelle *et al.*, "Towards Latency-Sensitive Cloud-Native Applications: A Performance Study on AWS," in *12th IEEE Int'l Conf. on Cloud Computing*. IEEE, 2019, pp. 272–280.

[19] ITU, "Quality of Experience (QoE) Requirements for Real-Time Multimedia Services over 5G Networks," Int'l Telecommunication Union, Tech. Rep. GSTR-5GQoE, 2022. [Online]. Available: https://www.itu.int/dms_pub/itu-t/opb/tut/T-TUT-QOS-2022-1-PDF-E.pdf

[20] A. Rossi *et al.*, "Bayesian Uncertainty Modelling for Cloud Workload Prediction," in *15th IEEE Int'l Conf. on Cloud Computing*. IEEE, 2022, pp. 19–29.

[21] Q. Liang *et al.*, "Model-Driven Cluster Resource Management for AI Workloads in Edge Clouds," *ACM Transactions on Autonomous and Adaptive Systems*, 2023.

[22] F. Tusa and S. Clayman, "End-to-End Slices to Orchestrate Resources and Services in the Cloud-to-Edge Continuum," *Future Generation Computer Systems*, vol. 141, pp. 473–488, 2023.

[23] I. Fakinos *et al.*, "Sequence Clock: A Dynamic Resource Orchestrator for Serverless Architectures," in *15th IEEE Int'l Conf. on Cloud Computing*. IEEE, 2022, pp. 81–90.

[24] S. Hoque *et al.*, "Towards Container Orchestration in Fog Computing Infrastructures," in *41st IEEE Annual Computer Software and Applications Conf.*, vol. 2. IEEE, 2017, pp. 294–299.

[25] D. Santoro *et al.*, "Foggy: A Platform for Workload Orchestration in a Fog Computing Environment," in *IEEE Int'l Conf. on Cloud Computing Technology and Science*. IEEE, 2017, pp. 231–234.

[26] M. S. D. Brito *et al.*, "A Service Orchestration Architecture for Fog-Enabled Infrastructures," in *Second Int'l Conf. on Fog and Mobile Edge Computing*. IEEE, 2017, pp. 127–132.