n-Dimensional QoS Framework for Real-Time Service-Oriented Architectures

D. W. McKee, S. J. Clement, Jie Xu
School of Computing
University of Leeds
Leeds, UK
{D.W.McKee, S.J.Clement, J.Xu}@leeds.ac.uk

D. Battersby
Jaguar Land Rover
Gaydon
UK
dbatters@jaguarlandrover.com

***Abstract*—Service-Orientation has long provided an effective mechanism to integrate heterogeneous systems in a loosely coupled fashion as services. However, with the emergence of Internet of Things (IoT) there is a growing need to facilitate the integration of real-time services executing in non-controlled, non-real-time, environments such as the Cloud. With the need to integrate both cyberphysical systems as hardware-in-the-loop (HIL) components and also with Simulation as a Service (SIMaaS) the execution performance and response-times of the services must be managed. This paper presents a mathematical framework that captures the relationship between the host execution environment and service performance allowing the estimation of Quality of Service (QoS) under dynamic Cloud workloads. A formal mathematical definition is provided and this is evaluated against existing techniques from both the Cloud and Real-Time Service Oriented Architecture (RT-SOA) domains. The proposed approach is evaluated against the existing techniques through simulation and demonstrates a reduction of QoS violation percentage by 22% with respect to response-times as well as reducing the number of Micro-Service (μS) instances with QoS violations by 27%.**

***Index Terms*—Real-Time, QoS, Services, SOA, Micro-Services, Schedulability, Resource Modelling, IoT, IoS, SIMaaS, Cloud**

I. INTRODUCTION

Predicting and guaranteeing service performance has long been a topic of research from domains of real-time scheduling through to Cloud Software as a Service (SaaS). With the emergence of Internet of Things (IoT), providing Quality of Service (QoS) guarantees is evermore vital to ensuring correct system performance. This introduces a particular challenge with ensuring that the services advertised QoS is actually met when deployed in the real-world where there are interfering workloads. This paper proposes a technique that captures the relationship between service performance and the execution environment in which it operates.

The importance of managing service performance becomes more significant with both the integration of cyberphysical systems and also the emergence of the Internet of Simulation (IoS) whereby simulations are deployed as services (SIMaaS) [1]. In the former, with hardware-in-the-loop (HIL) systems if the timing properties are not guaranteed the resulting action, or data, may be incorrect or unsafe [2]. When integrating simulations, either as a co-simulation or with other systems such as driver-in-the-loop (DIL) systems, the accuracy of the timing integration is critical to obtaining accurate results. The challenge to providing the necessary guaranteed response-times in this multi-domain, cross-organisation context is the lack of a fully controlled host environment for the services themselves. Alternatively there may not be a real-time operating system meaning that the approaches from the real-time systems community are not necessarily appropriate.

In the context of Cloud-based services where there are real-time constraints, the expected QoS may not be upheld under the influence of interference due to resource contention of CPU, memory, or even networks [3]–[6]. Currently most approaches for defining QoS and Service Level Agreements (SLAs) use static methods [7], [8] which assume the isolated execution of services. This paper evaluates those approaches which dynamically redefine QoS definitions in order to account for performance degradation due to task interference or degradation of the physical host machines.

In this paper we present both a review of the existing approaches from the Real-Time Service Oriented Architecture (SOA) (RT-SOA) and Cloud domains for predicting QoS and also a mathematically rigorous approach to capturing the QoS of services hosted in uncontrolled environments. The n-dimensional model captures the relationship between environmental resources, such as CPU and memory, and response-time. We also present a detailed analysis of the proposed technique against the existing approaches and we demonstrate an improved accuracy of 4% against the best Cloud technique and reduced QoS violation of 13% against real-time QoS techniques.

The remainder of this paper presents a review of state-of-the-art in QoS for RT-SOA. This is followed by proposed mathematical framework fo Real-Time QoS (RT-QoS) in Section III. Section IV presents an evaluation of both the proposed against the existing techniques followed by conclusions in Section V.

II. BACKGROUND: QOS FOR RT-SOA

Service Oriented Architectures (SOAs) have emerged as the premier set of standards for building cross-organisational dependable distributed systems [9]. In our previous work we utilised service-orientation in the development of a virtual engineering environment for the integration of vehicular simulations [5], [10]. In that context, models & simulations were provided as services through a paradigm known as Simulation as a Service (SIMaaS) and integrated into workflows which themselves could be provided as services (Workflow as a Service (WFaaS)) facilitating the simulation of an entire vehicle [1].

The SIMaaS and WFaaS paradigms were introduced as part of the wider Internet of Simulation (IoS), which extends the Internet of Things (IoT) with simulation for the purposes of decision support and in the context of engineering and manufacturing for rapid prototyping and product analysis [1]. These evolving domains, are hosted on infrastructure such as Cloud which mostly do not provide performance guarantees for individual services [5]. These environments are also shared with potentially millions of other processes and systems and the total workload experienced will vary [11], [12].

Hardware-in-the-loop cyberphysical systems require performance guarantees in order to maintain safe and correct operation. Mechanisms should consequently be in place to accurately predict and manage the execution performance of services which have real-time deadlines within the context of uncontrolled environments.

It is therefore necessary to consider how SOAs can be adapted to support real-time services. SOA research has focussed primarily on the challenges of service discovery, refactoring of legacy systems as services throughout system evolution [13], and online re-composition of services into workflows based on QoS violation. There has however been little work on providing a Real-Time QoS (RT-QoS) mechanism for RT-SOA that is capable of working in non-traditional real-time environments.

The remainder of this section therefore introduces the central concepts of real-time systems schedulability and then considers the state-of-the-art in RT-QoS techniques.

A. *Real-Time Systems Scheduling*

In order to guarantee the timely delivery of service, the appropriate formalisms from real-time systems research in real-time systems worst-case response times can be computed as a function of resource utilization and resource availability. Typically this is in terms of the number of processors available for execution and the time available:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq N \tag{1}$$

Where the sum of the utilisation factors U of each process i is less than the available processing defined by N. The utilisation factors are normally defined as the Worst-Case Execution Time (WCET) divided by the available time for computations C_i/T_i. The bounding condition N varies with the adopted scheduling model, for example in Fixed Priority Scheduling $\lim_{n\to\infty} N = n(2^{\frac{1}{n}} - 1) = 69.3\%$ or $N = 1$ in the case of Earliest Deadline First (EDF) [14]–[16]. Using equation 1 and a complete knowledge of all processes that will be executed it can be proved whether those processes will meet or miss their respective deadlines.

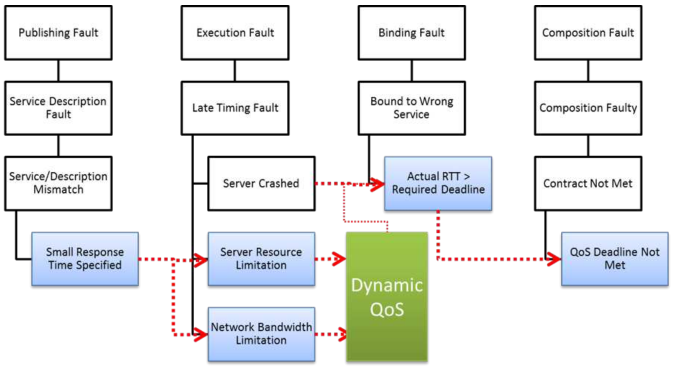


Fig. 1: SOA fault tree showing fault propagation mitigated by dynamic QoS, adapted from [18]

In more general distributed systems scheduling theory, the WCET can be adjusted with a *speedup* $\varsigma(i) = \frac{p(1)}{p(i)}$ where $p(1)$ defines the WCET of the process with no interference and full complete access to a processor and $p(i)$ the WCET across i processors [17].

B. *Managing Fault Propagation in SOAs*

In the context of SOAs there have been several approaches that aim to provide support for real-time systems and therefore provide guarantees about the QoS that any individual service can provided.

Specifically these approaches aim to mitigate the fault propagation depicted in Figure 1. There are a subset of SOA faults [18] that are directly related to the published execution time of a service followed by the actual observed response-time. If the service description specifies a response-time that could only achieved under certain circumstances a late timing fault can occur due to resource limitations. Therefore techniques are required which dynamically update the QoS definitions to mitigate the fault propagation through to causing a contract violation. Unless static approaches consistently overallocate time for execution they cannot provide guarantees in real-world execution environments with dynamic interfering workloads and physical equipment that will degrade over time.

*1) **Data Distribution Service (DDS)**:* DDS is a real-time middleware that utilises a publish/subscribe methodology to facilitate real-time communication between systems [19]. The iLand project [20], [21] utilises DDS to build a RT-SOA which considers the schedulability of the services with respect to the CPU utilisation and observed WCET. Subsequently the QoS is updated to predict a response time of $C \times \frac{1-U_{prev}}{U_{worst}}$. This approach has two primary limitations: the need for full-system control for a real-time middleware, and the need for full knowledge of the WCET of services.

*2) **Containers**:* The RT-Llama project utilised containers to provide bounding boxes to control the resource utilisation of services [22]–[24]. Services are dealt with as either *real-time* or *best-effort* where the former requires managed real-time CPUs with EDF scheduling. Similar to the previous technique this requires known WCETs and utilisation patterns for the services. Real-time processes are accepted on either an immediate or reserved basis. Immediate execution will only be admitted if the estimated response-time is less than the deadline given the currently available resources. These approaches require a fully controlled system with real-time CPUs.

*3) **Fuzzy-Logic**:* Is a technique that refers to the use of *fuzzy* terms such as "good" and "bad" along with probabilistic models. This has been applied to predicting QoS by defining "good" and "bad" response-time or memory consumption [25], [26]. Analysing the probability of a *fuzzy* term occurring provides the likelihood of observing a given level of performance. The likelihood is calculated from the number if appearances of the fuzzy term in the events of the service, divided by the number of executions of that service. This technique, as well as all the remaining techniques, is not explicitly a real-time approach and does not provide the levels of guarantee that are required by real-time systems.

*4) **Correlation**:* The use of Pearson's Correlation Coefficient (PCC) by Zheng et al. [27]–[29] in analysing and then predicting the QoS of web services has been adopted and integrated into many other approaches. This approach utilises the historical data regarding response-times around clusters of similar services. It correlates similar users and services using PCC and applies a *significance weighting* representing the density of invocations. They then select the *Top-K* similar neighbours, either users or services, and are able to recommend a service to a user.

*5) **Optimisation**:* Is another technique with a wide range of approaches, the majority focussing on the optimisation of the selection of services for workflows rather than the optimisation of QoS itself. Canfora et al. [30] do however optimise the specification of QoS using a genetic algorithm, accounting for cost, response-time, availability, and reliability. However since a minimum of 100 generations are required, the computation time to find a suitable QoS is not feasible for a system which must re-compute QoS and expected completion time at runtime.

*6) **Historical & Probabilistic**:* The simplest technique is the use of raw historical data to predict the service response-times. This could either use the mean observed response-time for previous execution instances of the service, or the worst observed response-time, or some other value based on a probabilistic model.

The NECTISE project took a different slant calculating the probability that the service would miss the desired deadline [31]. This approach was then used to inform the selection of services, specifically the level of service redundancy that would be required to provide a satisfactory likelihood of providing the services within the required timeframe: $1 - p^r$. This approach can also be applied to sequential workflows by using the sum of products.

*7) **Cost-Aware**:* Another approach models only the cost of execution, and therefore isn't directly useful for predicting service performance. However the modelling cost facilitates a trade-off between performance, power or energy, resource utilisation, and infrastructure pricing which in most circumstances must be considered and therefore features as a parameter in many of the other approaches [32]

Each of the approaches, except cost-awareness, that has been introduced here will be evaluated in Section IV. The next section presents a new approach for modelling QoS in dynamic and changing environments which are not supported by the real-time techniques.

III. Mathematical Framework for RT-QoS

This section outlines the mathematical framework for predicting the response-time and time-to-finish of executing services. The framework is defined by considering first the services and their host environments, followed by the models for resource utilisation, and finally the predictive model itself.

First however, the schedulability test in Equation 1 can be redefined in terms of the resource availability A_r such that the total availability of resources from the request time until the deadline be greater than or equal to the resources required by the service itself:

$$\forall r, \left(A_r \equiv 1 - \sum_{t}^{D} I_{i,r} \right) \geq C_{i,r} \tag{2}$$

Where D is the deadline, t is the current time, and I is the interference experienced by service i with regards to resource r. The interference is equivalent to the utilisation factor calculated in Equation 1 without the service of interest.

A. Service & Environment Model

Services can often be decomposed into μSs as those *"functional elements for which it is not practical to decompose into smaller components"* [33] and the interactions between them. A given μS ($\mathbf{s}$) may be invoked one or more times, each instance referred to as $\mathbf{s}_n$, and execution progress p is monitored with a frequency f to provide k observations.

Each execution instance of a μS will exist on a host $\mathbf{h}$ which has a set of available resources A at time t. At each observation during μS execution the resource availability can be recorded:

$$\forall r \in \mathcal{R}, \forall t, \alpha(h, \mathbf{s}_n) = 1 - (U(h(\mathbf{s}_n))_{r,t} - U(\mathbf{s}_n)_{r,t}) \tag{3}$$

Where U provides either the sum of resource utilisation by all processes hosted by h (the host of the μS), or the resources utilised by the μS at time t. $\mathcal{R}$ is the set of resource types being monitored, e.g. CPU and memory. Alternatively, in terms of traditional real-time systems notation, availability can be defined with respect to interference as $1 - I_r$ (Eq. 2). The observed availability can then be collated to provide a total A_r^{Σ} (Eq. 4) and average A_r (Eq. 5) observed availability over the duration of an execution instance $\mathbf{s}_n$:

$$A_r^{\Sigma}(h, \mathbf{s}_n) = \sum_{p=0}^{k} \alpha(h, \mathbf{s}_n)_{r,t}[t \equiv p] \tag{4}$$

$$A_r(h, \mathbf{s}_n) = \frac{A_r^{\Sigma}(h, \mathbf{s}_n)}{k} \tag{5}$$

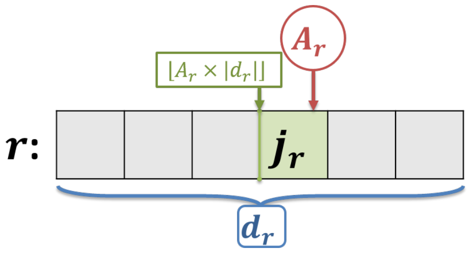


Fig. 2: Framework multi-dimensional coordinate system indexed by j, calculated from the availability A_r and the discrete space d_r

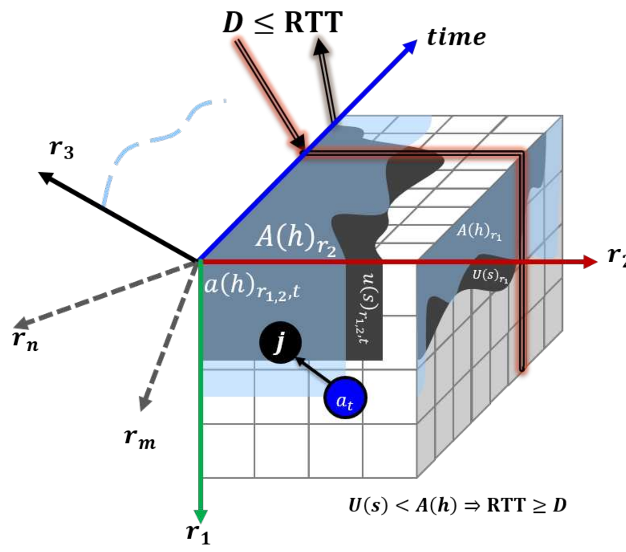


Fig. 3: Visual representation of proposed QoS approach

Finally these availability values can be converted into coordinate values j in the discrete space d_r of possible values for A_r as shown in Figure 2:

$$j = \{r \in \mathcal{R} : \lfloor A_r(h, \mathbf{s}_n) \times |d_r| \rfloor\} \tag{6}$$

Where $|d_r|$ can be configured based on experimental evidence or based on system constraints monitoring constraints.

B. Resource Utilisation

Given an executing μS its resource utilisation can be observed, recorded, and normalised into a resource utilisation model U indexed by the constraint $j$and the execution progress p. U is a 4-tuple comprising of the mean μ, minimum $\wedge$, maximum $\vee$, and variance σ^2 for each observation point. This model can then be used to provide a forecast F of the remaining resource required for execution to complete from a given point p:

$$F(s)^{x\in\{\mu,\wedge,\vee\}}_{j,r,p} = \sum_{i=p}^{k} U(s)^{x}_{j,r,k-i} \tag{7}$$

$$F(s)^{\sigma^2}_{j,r,p} = \frac{\sum_{i=p}^{k} U(s)^{\sigma^2}_{j,r,k-i}}{k-p} \tag{8}$$

Where the summations are performed in reverse from $i = k$ to $i = 0$ allowing the calculation of F to be performed in $\mathcal{O}(n)$ time.

C. Predictive Model

These models are used to populate the predictive model $\mathcal{M}$ which is an $|\mathcal{R}| + 1$ dimensional model representing the resource types and time. As shown in Figure 2 and in Equation 6, the finite set of possible coordinates of $j \in \mathcal{J}$ acts as a coordinate map applied to index the model:

$$m : \mathcal{J} \to \mathcal{M}, j \mapsto m_j \tag{9}$$

Which allows the observed resource availability to act as a constraint on the model. Each model element m_j is a 6-tuple $\langle t^{\vee}, t^{\mu}, T_j, U_j, F_j, I_j \rangle$. Where T stores the historical response-times and $I_j = |\mathcal{M}^T_j|$ is the indicator density function, as used by Zheng et al. [27] as a significance weighting.

The model is then used to estimate the response-time of μSs as they are deployed:

$$\mathbf{RTT}[A(h(\mathbf{s}_n)](\mathbf{s}_n) = \left\langle \frac{\sum m^T_j}{M^I_j}, m^{t^{\vee}}_j \right\rangle \tag{10}$$

Where the currently observed resource availability acts as a constraint (shown using Iverson brackets [34]) on the model to provide a pair of values representing the average and worst observed response-times. In the same fashion, once execution has started, the time-to-finish $\mathcal{TTF}$ for the μS can be estimated:

$$\mathcal{TTF}[A(h(\mathbf{s}_n)](\mathbf{s}_n) = \left(1 - \frac{p}{k}\right) \mathcal{M}(\mathbf{s}_n)_j \tag{11}$$

Where the execution is progress is estimated by compared the observed and expected resource utilisation in a pessimistic but non-decreasing manner:

$$p(\mathbf{s}_n)_t = \max\left\{p(\mathbf{s}_n)_{t-1}, \min\left\{\forall r \in \mathcal{R} : \tfrac{1}{k}\left\lfloor \frac{k\cdot\sum_{x=0}^{t} [\![U(\mathbf{s}_n)_{j,r,x}]\!]_{0..1}}{F(\mathbf{s})_{j,r}} \right\rfloor\right\}\right\} \tag{12}$$

D. The Sparse & Initial Cases

The model so far facilitates the estimation of response-times and remaining computation time of μSs using previous μS execution data under the currently observed environmental resource availabilities. There are however two situations where the current model is not sufficient:

1) **Initial-case** where the model is empty with no execution information about the μS.
2) **Sparse-case** where there is information about execution performance under a subset of resource availability configurations.

In the first instance for a prediction to be made one of the following must be provided:

1) Response-time purely as a nominal value where the resource utilisation must be assumed to be 100% of the best available host $\mathbf{h}$.
2) WCET with a utilisation model or resource requirement.
3) Response-time with or without resource utilisation information but alongside a host specification.

Algorithm 1: Core Algorithm

```
begin Online Monitoring
    Invoke s on $h$
    Start Timer
    $p$= 0
    while $s$ running do
        if Timer.Elapsed $\geq \omega$ then
            /* Utilisation and availability
               are observed in parallel */
            $u$ = OBSERVE_UTILISATION(s,$h$)
            $a$ = OBSERVE_AVAILABILITY($h$)
            ($p$,$\mathcal{TTF}$)=UPDATE_PREDICTION(s,$u$,$a$)
        end
        RTT= Stop Timer
    end
end
begin Model Updating
    $T$ = UPDATE_DATA_SETS(RTT, $A$)
    $U, A$ = BUILD_UTIL_AVAIL_MODEL($u, a$)
    $n++$
    $\mathcal{M}$= BUILD_TIME_MODEL
end
```

Allowing for a traditional real-time systems estimation of WCET.

In the second instance of a sparsely populated model, a neighbourhood approach can be used to predict the missing value:

$$\mathbf{RTT}[A(h(\mathbf{s}_n))](\mathbf{s}_n) = \frac{\sum_{i \neq j} \left(I(\mathbf{s})_j \cdot \mathcal{M}(\mathbf{s})_j \cdot \mathcal{D}^{-1}(i,j)\right)}{\sum_{i \neq j} \left(I(\mathbf{s})_j \cdot \mathcal{D}^{-1}(i,j)\right)} \tag{13}$$

Where $\mathcal{D}^{-1}$ is the inverse distance measure between the point j of interest and all other points i in the model at the current point in time: $\mathcal{D}^{-1} = \frac{1}{dist(i,j)}$.

Therefore overall the framework can be summarised as a set of three cases, Initial (A), Sparse (B), and Full (C):

$$\mathbf{RTT}[A(h(\mathbf{s}_n))] = \begin{cases} A & \mathcal{M}^T \equiv \emptyset \\ \beta_1 B + \beta_2 A & \mathcal{M}_j^T \equiv \emptyset, \exists i : \mathcal{M}_i^T \neq \emptyset \\ \gamma_1 C + \gamma_2 B + \gamma_3 A & \mathcal{M}_j^T \neq \emptyset \end{cases} \tag{14}$$

This is shown visually in Figure 3 as a multidimensional space recording utilisation and availability over time. The algorithmic representation is also shown in Algorithm 1.

The next section takes the proposed framework and evaluates it against the techniques outlined in Section II.

IV. Evaluation & Comparison of Approaches

In this section the metrics used to evaluate the proposed approach against the approaches discussed in Section II are discussed. Then the simulation infrastructure is described before the evaluation is presented focussing on QoS violation and wasted resource allocation.

Length	Id	RTT (s)	Historical (s)	Zheng (s)	iLand (s)	Benbernou (s)	McKee (s)
Short	T01	14.3	19.4	20.9	355.3	15.2	18.5
	T02	14.7	17.6	20.9	405.5	15.0	16.7
	T03	14.7	20.4	21.1	307.7	15.3	20.0
	T04	14.8	20.8	20.9	415.8	15.0	19.9
Average		**14.6**	**19.6**	**21.0**	**371.1**	**15.2**	**18.8**
Medium	T05	33.9	48.3	51.8	4680.4	36.9	47.5
	T06	34.5	51.9	51.8	4758.2	37.9	51.1
	T07	33.5	47.7	52.2	4771.1	36.6	46.9
	T08	34.0	48.9	51.7	4576.6	37.1	48.3
Average		**34.0**	**49.2**	**51.9**	**4696.5**	**37.1**	**48.4**

TABLE I: μS response-times and QoS allocation

A. Measures & Metrics

There are several metrics that have been commonly used to evaluate QoS techniques with respect to:

- **Prediction Accuracy** using Mean Absolute Error (MAE) as used by Zhu et al. [35] and Mean Percentage Waste (MPW). The former compares the measured Q_i against the predicted QoS value $\hat{Q}_i$ whilst the later takes the average overallocation of QoS:

$$MAE = \frac{\sum_i^N |\hat{Q}_i - Q_i|}{N} \tag{15}$$

$$MPW = \frac{\sum_i^N (\hat{Q}_i - R_i)[\hat{Q}_i > Q_i]}{N} \tag{16}$$

- **QoS Violation** measuring the Absolute Violation Count (AVC) as well as the Mean Absolute Violation (MAV) and Mean Percentage Violation (MPV), specified below using the square Iverson Brackets for summation conditions [34]:

$$AVC = \sum_i [(\hat{Q}_i < Q_i) = 1] \tag{17}$$

$$MAV = \frac{\sum_i (\hat{Q}_i - Q_i)[\hat{Q}_i < Q_i]}{AVC} \tag{18}$$

$$MPV = \frac{MAV}{\sum_i \hat{Q}_i / N} \tag{19}$$

B. Cloud Simulation of RT-QoS

For the purposes of analysing the QoS approaches simulations of services executing in a Cloud environment will be used [36]. The existing workload on the servers is represented as a periodic workload pattern with an average CPU and memory load of 80-95% [11]. The μSs themselves are modelled as Cloud tasks with resource utilisation patterns [37]. The elements of the cloud, including the server models, virtual machines, and tasks themselves are those used in other work studying the behaviour of Cloud through simulation [3], [6], [38] and only the QoS approaches themselves are introduced.

8 Cloud task types were used based on the short and medium Cloud task types [37], eager and lazy resource acquisition, and also active and non-releasing resource release paradigms [39]. Each task type defines a μS which was then executed 100 times. An initial QoS estimate was calculated by each approach based only on the Cloud task length parameter (7 and 16 million instructions respectively).

Length	Id	Historical	Zheng	iLand	Benbernou	McKee
Short	T01	38%	33%	28%	11%	13%
	T02	9%	56%	27%	12%	9%
	T03	8%	27%	24%	13%	11%
	T04	15%	19%	29%	14%	14%
Average		18%	34%	27%	12%	12%
Medium	T05	15%	40%	54%	16%	9%
	T06	18%	20%	45%	17%	13%
	T07	0%	56%	56%	15%	0%
	T08	10%	33%	50%	19%	8%
Average		11%	37%	51%	17%	7%
Average		14%	35%	39%	15%	10%

TABLE II: QoS MPV across μSs and approaches

Length	Id	Historical	Zheng	iLand	Benbernou	McKee
Short	T01	31%	38%	3503%	12%	27%
	T02	18%	37%	2702%	10%	14%
	T03	30%	38%	1853%	10%	32%
	T04	34%	37%	3124%	9%	32%
Average		28%	37%	2795%	10%	26%
Medium	T05	49%	59%	14258%	33%	50%
	T06	57%	59%	13959%	35%	60%
	T07	47%	62%	14755%	30%	46%
	T08	48%	57%	13640%	31%	52%
Average		50%	60%	14153%	32%	52%
Average		39%	48%	8474%	21%	39%

TABLE III: QoS MPW across μSs and approaches

		Historical	Corr	iLand	Ben	Average	Average'
Waste	*Count*	657	718	604	-502	295.4	291.0
	Mean	3.98%	13.92%	9667.19%	-32.10%	1930.60%	-4.73%
	Std.Dev	0.055	0.100	70.310	0.142	▨	▨
	Var	0.003	0.010	4943.445	0.020	▨	▨
Violation	*Count*	-16	-1	97	232	219.4	71.7
	Mean	-0.51%	18.15%	27.97%	13.40%	21.92%	10.34%
	Std.Dev	0.128	0.310	0.171	0.105	▨	▨
	Var	0.017	0.096	0.029	0.011	▨	▨
MP Trade-off	*Hard-RT*	-0.5%	18.1%	28.0%	13.4%	21.92%	10.34%
	Firm-RT	1.0%	16.7%	3241.0%	-1.8%	658.14%	5.32%
	Soft-RT	2.5%	15.3%	6454.1%	-16.9%	1294.37%	0.29%
	Not RT	4.0%	13.9%	9667.2%	-32.1%	1930.60%	-4.73%
Count Trade-off	*Hard-RT*	-16	-1	97	232	219.4	71.7
	Firm-RT	208	239	266	-13	244.7	144.8
	Soft-RT	433	478	435	-257	270.1	217.9
	Not RT	657	718	604	-502	295.4	291.0

TABLE IV: Difference in QoS violation and wasted execution time between existing approaches and proposed method: $Approach - Proposed$ (Average' ignores the iLand method)

Table I outlines the response-times of the 8 μSs execution instances and the predicted QoS by the historical, correlation by Zheng et al. [27], fuzzy-logic by Benbernou et al. [25], and the proposed approach. The average QoS allocation by the real-time approach by [21] is 2553.4, more than $40\times$ greater than largest prediction by any of the other approaches and in the remainder of this section can not normally be reasonably shown on the graphs.

The remainder of this section will look in detail at the results of the QoS approaches with regards to violation and wastage.

C. QoS Violation Analysis

Table II details the MPV by each of the QoS approaches, including the proposed method, across all of the μS execution instances. The proposed method reduces the MPV in comparison to each of the existing methods with 10% compared to MPVs between 14% and 39%. Figure 4b also depicts the difference between QoS violations between the different approaches. For each μS execution instance where the MPV is above the benchmark line, defined by the proposed method, the observed violation is improved using the proposed method. Alternatively for each MPV below the benchmark the respective existing method performed better than the proposed method.

Looking, as an example, specifically at the 100 execution instances of the first class of μS (a small a Cloud task with lazy resource acquisition and eager resource release), the Absolute Violation Count (AVC) can be clearly seen in Figure 4a. The correlated and historical based approaches have only 2 violations and the proposed approach only 3, whilst the real-time approaches using fuzzy-logic and DDS have a total of 23 and 20 violations respectively for this μS.

D. QoS Waste Analysis

In terms of MPW, Table III details the overallocation by each of the QoS approaches. The real-time iLand approach overallocates by an average of 8474% whilst the historical, correlation-based, and fuzzy-logic approaches overallocate on average by 39%, 48%, and 21% respectively. The proposed also returns an MPW of 39% which as can be seen in Figure 5b is a slight improvement on the historical approach.

Again looking specifically at the first μS, the correlation approach by Zheng et al. followed by the pure historical approach waste the most time whilst the fuzzy-logic approach wasted the least total time over the 100 execution instances (see Figure 5a).

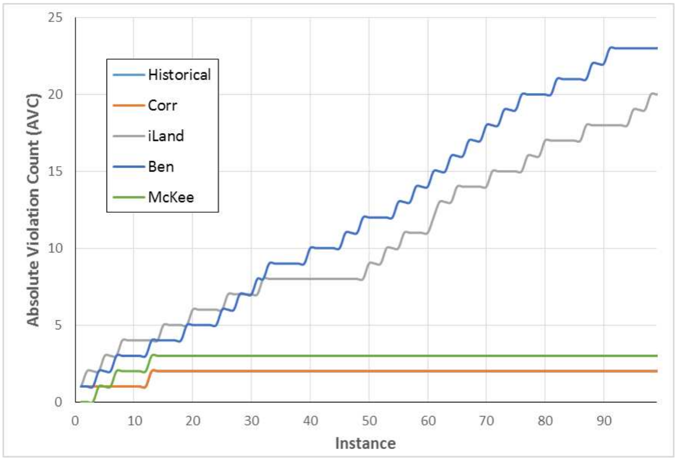
E. Evaluation of Violation vs. Waste

With regards to QoS violation the historical and correlated approaches appear to show a slight improvement on the proposed method in Figure 4a. However, as shown in Table IV the proposed approach across all 8 μSs was significantly more accurate than the correlated approach, wasting 14% less and reducing violation by 18%.

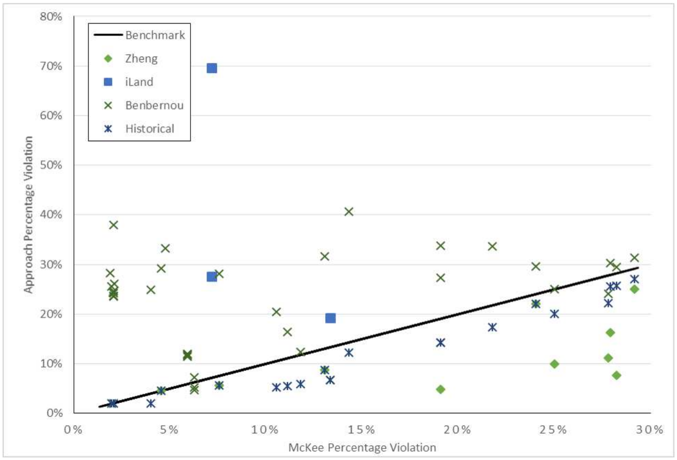
The proposed approach demonstrates a significant improvement over the real-time DDS and fuzzy-logic approaches with an accuracy over 9000% better than DDS and with an average violation percentage improved by 28% and 13% respectively.

The proposed approach resulted in 38 QoS violations compared to 18 for the correlation technique, 21 for the historical approach, 97 for DDS, and 241 for the fuzzy-logic approach. Although the fuzzy-logic approach demonstrated the most accurate predictions compared to each of the approaches the level of QoS violation is significantly worse.

The improvement against the historical technique is the least, 4% reduced MPW. This improvement is represented across 82% of μS execution instances. The historical approach bears the greatest It is anticipated that performance will improve with further μS executions and with a wider range of workload interference.

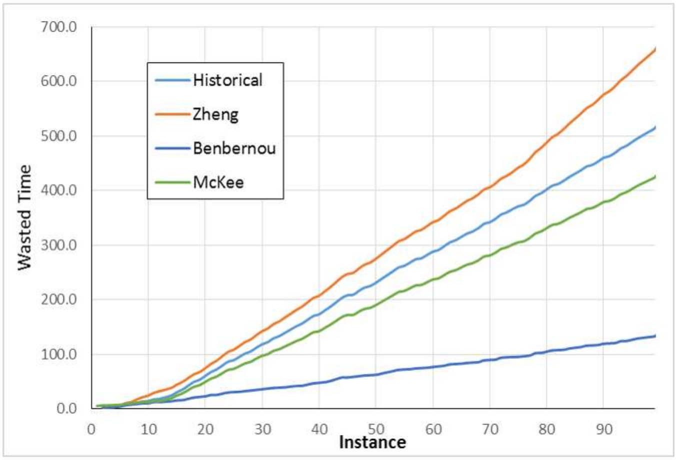


(a) Absolute Violation Count (AVC) of 1^{st} 100 μSs

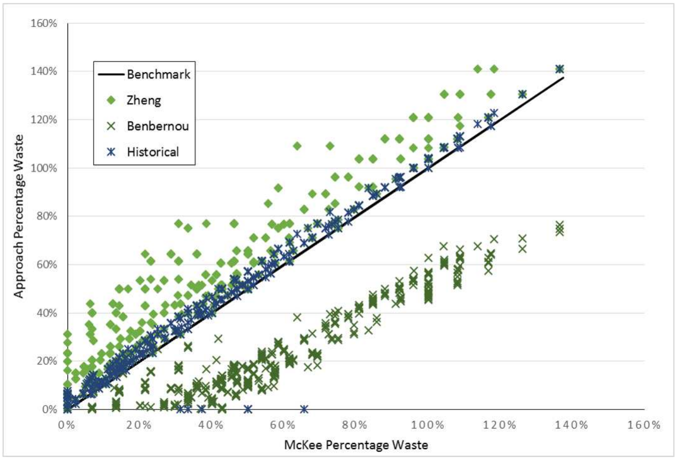


(b) Difference of Percentage Violation of all μS instances between approaches

Fig. 4: μS QoS violation



(a) Waste/Overallocated time of 1^{st} 100 μSs



(b) Difference of Percentage Waste/Over-allocated between approaches

Fig. 5: μS QoS wasted overallocation

V. Conclusion and Further Work

This paper has presented the need for a robust mathematical framework for guaranteeing response-times in Service Oriented Architectures (SOAs) with real-time constraints. A discussion of the existing techniques including the use of Data Distribution Service (DDS), fuzzy-logic, PCC, optimisation, and pure historical data has been presented. These techniques have then been evaluated against the proposed n-dimensional framework which captures the relationship between computational resources, such as CPU and memory, and service performance.

The proposed approach has been demonstrated using Cloud workload simulation to improve the Mean Percentage Violation (MPV) by over 13% for real-time QoS techniques. It has also increased the accuracy and therefore reduced the overallocation of time by general Cloud QoS approaches by between 4% and 14%.

There is further work to evaluate properties of the proposed approach, including the impact of the dimensionality and the impact of the granularity of each of the dimensions on prediction accuracy. Additional interfering workload patterns can be analysed with a greater degree of variation. Furthermore any dynamic QoS approach requires a training set of data before being deployed, deciding on the size of the training set for a given service type remains an open question.

Acknowledgment

This work was supported by Jaguar Land Rover, UK-EPSRC grant EP/K014226/1 as part of the Programme for Simulation Innovation.

References

[1] D. Mckee, S. Clement, X. Ouyang *et al.*, "The internet of simulation, a specialisation of the internet of things with simulation and workflow as a service (sim/wfaas)," in *11th IEEE International Symposium on*

Service-Oriented System Engineering (SOSE 2017). IEEE, January 2017.

[2] M. Dooner, J. Wang, and A. Mouzakitis, "Development of a Simulation Model of a Windshield Wiper System for Hardware in the Loop Simulation," in *Autom. Comput. (ICAC), 2013 19th Int. Conf.*, 2013.

[3] X. Ouyang, P. Garraghan, D. McKee *et al.*, "Straggler Detection in Parallel Computing Systems through Dynamic Threshold Calculation," in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, mar 2016, pp. 414–421.

[4] D. W. McKee, D. Webster, and J. Xu, "Enabling Decision Support for the Delivery of Real-Time Services," in *2015 IEEE 15th Int. Symp. High-Assurance Syst. Eng.* IEEE, jan 2015.

[5] D. McKee, D. Webster, P. Townend *et al.*, "Towards a virtual integration design and analysis enviroment for automotive engineering," in *2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*. IEEE, 2014.

[6] P. Garraghan, X. Ouyang, R. Yang *et al.*, "Straggler Root-Cause and Impact Analysis for Massive-scale Virtualized Cloud Datacenters," *IEEE Transactions on Services Computing*, vol. 1374, no. c, pp. 1–1, 2016.

[7] L. Liu, D. Russell, D. Webster *et al.*, "Delivering Sustainable Capability on Evolutionary Service-oriented Architecture," *2009 IEEE Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, pp. 12–19, mar 2009.

[8] N. Gold, A. Mohan, C. Knight *et al.*, "Understanding service-orientated software," *IEEE Softw.*, vol. 44, no. October, pp. 71–77, oct 2008.

[9] M. P. Papazoglou, P. Traverso, S. Dustdar *et al.*, "Service-Oriented Computing: State of the Art and Research Challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[10] D. W. McKee, D. Webster, J. Xu *et al.*, "DIVIDER: Modelling and Evaluating Real-Time Service-Oriented Cyberphysical Co-Simulations," in *2015 IEEE 18th International Symposium on Real-Time Distributed Computing*. IEEE, 2015, pp. 272–275.

[11] C. Fehling, F. Leymann, R. Retter *et al.*, *Cloud Computing Patterns*. Vienna: Springer Vienna, 2014.

[12] S. Singh and I. Chana, "Resource provisioning and scheduling in clouds: QoS perspective," *J. Supercomput.*, vol. 72, no. 3, pp. 926–960, mar 2016.

[13] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, "On the Evolution of Services," *IEEE Trans. Softw. Eng.*, vol. 38, no. 3, pp. 609–628, may 2012.

[14] A. Burns and A. Wellings, "Scheduling," in *Real-Time Syst. Program. Lang.*, 2001, pp. 465–522.

[15] H. Kopetz, "Real-Time Scheduling," in *Real-Time Syst. Des. Princ. Distrib. Embed. Appl.*, 2nd ed. Springer, 2011, pp. 239–258.

[16] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment Scheduling Algorithms for Multiprogramming," *J. Assoc. Comput. Mach.*, vol. 20, no. 1, pp. 46–61, 1973.

[17] M. Drozdowski, *Scheduling for Parallel Processing*, ser. Computer Communications and Networks. Springer London, 2009, vol. 1.

[18] S. Bruning, S. Weissleder, and M. Malek, "A Fault Taxonomy for Service-Oriented Architecture," in *10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. IEEE, 2007, pp. 367–368.

[19] R. Serrano-Torres, M. García-Valls, and P. Basanta-Val, "Virtualizing DDS middleware: Performance challenges and measurements," *IEEE Int. Conf. Ind. Informatics*, pp. 71–76, 2013.

[20] M. García-valls, P. Basanta-val, M. Marcos *et al.*, "A bi-dimensional QoS model for SOA and real-time middleware," *International Journal of Computer Science and Engineering*, 2013.

[21] I. Estévez-Ayres, P. Basanta-Val, and M. García-Valls, "Composing and scheduling service-oriented applications in time-triggered distributed real-time Java environments," *Concurr. Comput. Pract. Exp.*, vol. 26, no. 1, pp. 152–193, jan 2014.

[22] K.-J. Lin, M. Panahi, Y. Zhang *et al.*, "Building Accountability Middleware to Support Dependable SOA," *IEEE Internet Computing*, vol. 13, no. 2, pp. 16–25, mar 2009.

[23] T. Cucinotta, G. Anastasi, and L. Abeni, "Respecting Temporal Constraints in Virtualised Services," in *2009 33rd Annual IEEE International Computer Software and Applications Conference*. IEEE, 2009, pp. 73–78.

[24] T. Cucinotta, A. Mancina, G. Anastasi *et al.*, "A Real-Time Service-Oriented Architecture for Industrial Automation," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 267–277, aug 2009.

[25] S. Benbernou, A. Hadjali, N. Karam *et al.*, "Managing QoS Acceptability for Service Selection : A Probabilistic Description Logics Based Approach," in *Proceedings of the 28th International Workshop on Description Logics*, vol. 1350, 2015.

[26] S. de Gyves Avila and K. Djemame, "Fuzzy Logic Based QoS Optimization Mechanism for Service Composition," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE, mar 2013, pp. 182–191.

[27] Zibin Zheng, Hao Ma, M. R. Lyu *et al.*, "QoS-Aware Web Service Recommendation by Collaborative Filtering," *IEEE Transactions on Services Computing*, vol. 4, no. 2, pp. 140–152, apr 2011.

[28] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, jan 2014.

[29] Z. Zheng, H. Ma, M. R. Lyu *et al.*, "Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, jul 2013.

[30] G. Canfora, M. Di Penta, R. Esposito *et al.*, "An approach for QoS-aware service composition based on genetic algorithms," in *Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05*. New York, New York, USA: ACM Press, 2005, pp. 1069–1075.

[31] D. Russell, L. Liu, Z. Luo *et al.*, "Realizing Network Enabled Capability Through Dependable Dynamic Systems Integration," in *2010 10th IEEE International Conference on Computer and Information Technology*, no. Cit. IEEE, jun 2010, pp. 1269–1274.

[32] T. Kaur, D. Kaur, and A. Aggarwal, "Cost model for software as a service," in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*. IEEE, sep 2014, pp. 736–741.

[33] M. Fowler and J. Lewis, "Microservices a definition of this new architectural term." [Online]. Available: http://martinfowler.com/articles/microservices.html

[34] D. E. Knuth, "Two notes on notation," *Am. Math. Mon.*, vol. 99, no. 5, pp. 403–422, 1992.

[35] J. Zhu, P. He, Z. Zheng *et al.*, "Towards Online, Accurate, and Scalable QoS Prediction for Runtime Service Adaptation," in *2014 IEEE 34th Int. Conf. Distrib. Comput. Syst.* IEEE, jun 2014, pp. 318–327.

[36] P. Garraghan, D. McKee, X. Ouyang *et al.*, "SEED: A Scalable Approach for Cyber-Physical System Simulation," *IEEE Transactions on Services Computing*, vol. 9, no. 2, pp. 199–212, mar 2016.

[37] I. S. Moreno, P. Garraghan, P. Townend *et al.*, "Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 208–221, apr 2014.

[38] A. Albatli, D. McKee, P. Townend *et al.*, "Prov-te: A provenance-driven diagnostic framework for task eviction in data centers," in *The Third IEEE International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, February 2017.

[39] M. Kircher and P. Jain, "Patterns for resource management," *Pattern-oriented software architecture*, vol. 3, 2004.