Enabling Decision Support for the Delivery of Real-Time Services

David McKee, David Webster, Jie Xu
School of Computing,
University of Leeds
Leeds, UK
{scdwm, D.E.Webster, J.Xu}@leeds.ac.uk

Abstract—**The domain of high assurance distributed systems has focused greatly on the areas of fault tolerance and dependability. As a result the paradigm of service orientated architectures (SOA) has been commonly applied to realize the significant benefits of loose coupling and dynamic binding. However, there has been limited research addressing the issues of managing real-time constraints in SOAs that are by their very nature dynamic. Although the paradigm itself is derived from fundamental principles of dependability, these same principles appear to not be applied when considering the timed dimension of quality of service. As a result the current state-of-the-art in SOA research only addresses soft real-time and does not seek to provide concrete guarantees about a systems performance. When a distributed system is deployed we do not understand enough the emerging behavior that will occur. This paper therefore proposes an approach that probabilistically monitors system state within a given workflow's execution window. Utilizing a real distributed system we experiment with services from the computer vision domain, with clear real-time constraints, evaluating the performance of each system component. Our approach successfully models the likelihood of the service meeting providing various levels of QoS, providing the basis for a more dynamic and intelligent approach to real-time service orientation.**

Keywords-real-time; SOA; QoS; workload patterns; workload characterization; resource usage patterns

I. INTRODUCTION

In service orientated architectures (SOA) the provision of services under real-time Quality of Service (QoS) constraints has remained an elusive challenge. However with the increasing usage of SOA in domains such as banking and manufacturing [1] increasingly there have been attempts to tailor SOA to industry specific domains with significant hard real-time constraints with critical deadlines where failure to meet them can result in issues of either safety, such as in-vehicle SOA [2], or more often cause financial repercussions [3].

In [4] we identified that one of the most significant challenges in providing real-time capability through SOA is the ability to guarantee that capability given variations that occur in the host execution environment. It is therefore necessary in real-time SOAs to provide the ability to manage and guarantee the adherence to hard real-time deadlines through management of processor and resource utilization [4]. Existing approaches [5], [6] provide limited real-time capability due to the static nature of defining QoS in service definitions which are unable to provide online adaptation to changing resource availability.

This paper expands a motivating case-study presented in our previous work [7] which identified the following crucial real-time constraints:

- The entire workflow has a critical deadline of 200ms.
- Individual services have deadlines of less than 200ms.
- A redundancy level of 3 or more is required in order to provide "good-enough" accuracy of results.

This paper extends that work by analyzing the system level performance in light of those temporal resource constraints. Specifically through the characterization of servers and tasks we explore the relationship between system resources and the realistic deliverable QoS. Previously [4] we identified that other tasks executing on the same servers as our system will affect the worst-case response time (WCRT) of the services through shared cpu and memory utilization. We adapt the M-VCR framework used in [7] to provide a dynamic model of QoS which models the likelihood of a given server having the capability to provide the required WCRT for a given service instance. We utilize both performance data from a real Google datacenter tracelog [8] as well as from our own lab cluster in combination with a dedicated SOA simulator to generate a probabilistic model of the likely performance of a given service which can then be used to adapt the system-wide workload distribution.

This rest of this paper is organized as follows: Section II provides a problem definition with key insights into the effect of CPU and memory interference on the workflow and services of interest. Section III proposes a scheme for probabilistically modelling the utilization of CPU and memory as well as the relevant requirements for a given task. Section IV outlines the system model that we have adopted as well as key assumptions that underpin the presented research. Sections V and VI present the implementation and experimental results respectively. The paper concludes comparing the proposed approach against related state-of-the-art followed by conclusions and our planned future work.

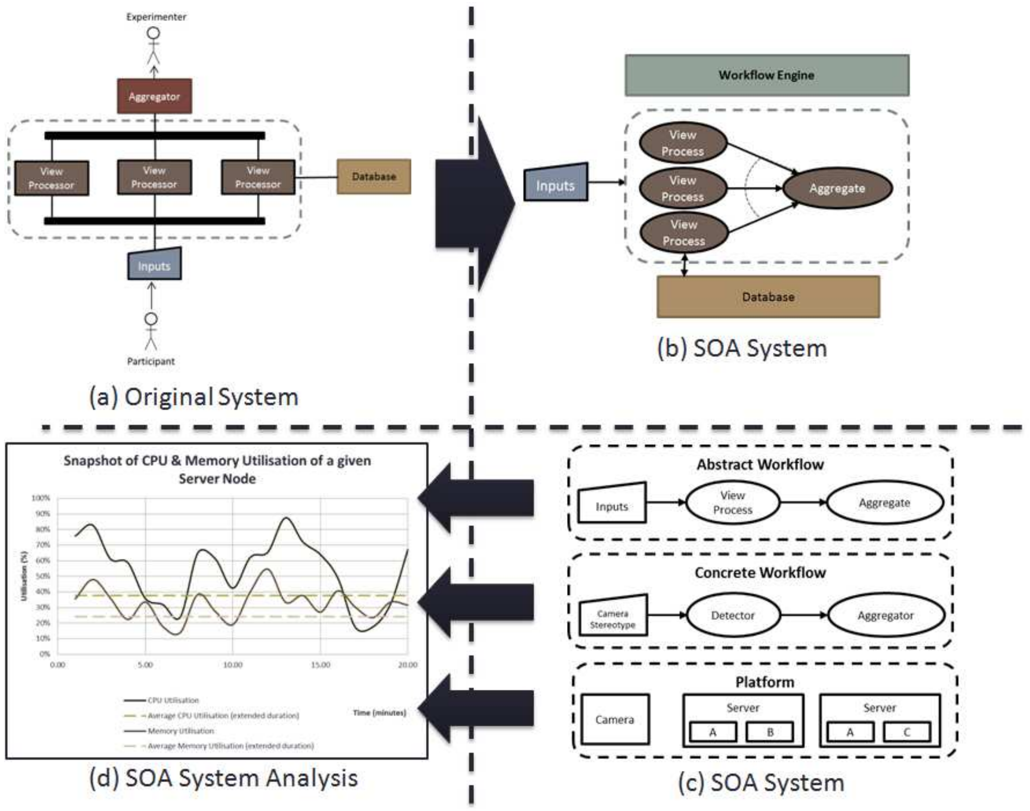


Figure 1: SOA representation of the case study and analysis of the system performance

II. THE CHALLENGE OF DELIVERING REAL-TIME SERVICES

In this section we briefly introduce the concept of service orientation, identifying the key benefits and reasons for adopting it as the architectural paradigm. We then consider the limitations of statically defined QoS with reference to experimental evidence.

A. Service Oriented Architectures

Service Oriented Architectures (SOA) (along with the more general concept of Software-as-a-service) have emerged in recent years as the de jure set of standards for building cross-organizational distributed systems. The motivation for SOA is to enable the cross-organizational integration of loosely coupled networked systems using clearly defined interfaces which support the concepts of discovery and reusability [9]. Service interfaces are, therefore, defined in terms of their abstract capability and independently of the concrete platforms that provide them.

As a result the SOA paradigm allows us to more effectively manage the levels of redundancy required at runtime for any given service in light of system performance. Figure 1(a) depicts the system presented in [7], (b) and (c) present the corresponding service orientated model.

B. QoS in Service Provision

Given the workflow defined in Figure 1(c) service level agreements defining the level of QoS are defined at each service step with an overall workflow critical response time of 200ms.

As can be seen in Figure 1(d) the underlying system has significant variation in terms of both cpu and memory utilization. The graph depicts a snapshot of a given server's utilization. Intuitively it can be seen that if a service was requested either between time 2minutes or 13minutes there is a significantly increased probability of the task being blocked, resulting in delays that are unacceptable for the overall workflow.

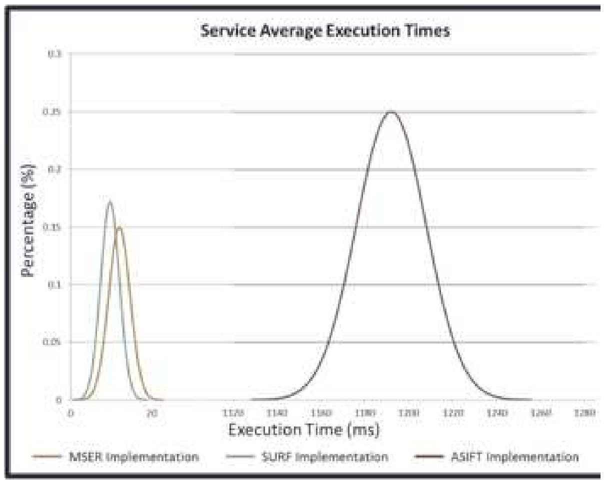


Figure 2: Analysis of isolated performance of individual services

In Figure 2 we also see that three different implementations of a service definition (coming from our previous work in gesture recognition [7]) provide significantly differing execution times. This is due to algorithmic differences, variations in the input data, variations in inputs, and in inter-service communication latencies even when running in isolated environments. As with the work presented by [6], both data-centric levels of QoS as well platform oriented QoS are modelled. However, it can also be seen that the QoS cannot be statically defined and there is clearly a relationship between the two types.

III. Proposed Scheme

This section outlines a formalized abstract model of SOA and is described with relation to the simulation environment. It then proposes a scheme for modelling QoS in a non-static fashion. And finally we describe how the QoS is specified within that experimental environment.

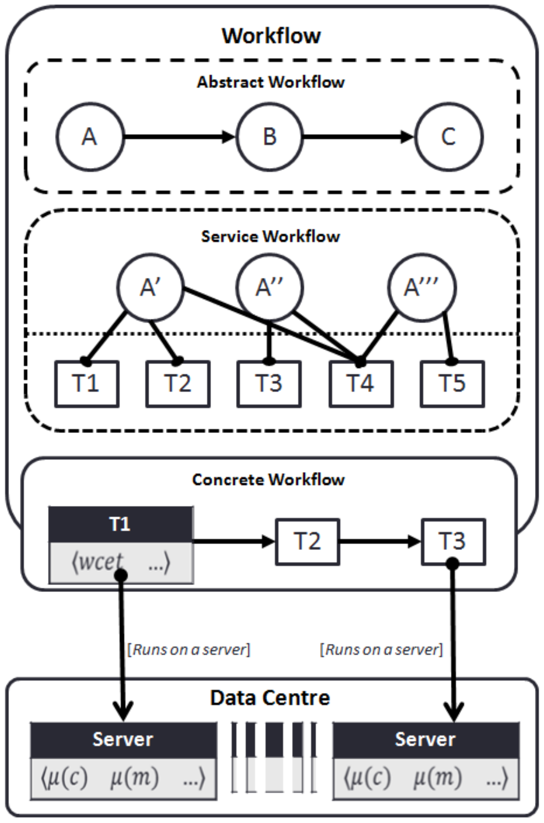


Figure 3: Extended SOA layers of abstraction considering their link to the physical infrastructure

A. *System Model*

As previously seen in Figure 1(c) the SOA paradigm consists of multiple layers of abstraction. Garcia-valls et al. [6], [10], [11] identify the need for an additional intermediate layer between the abstract and concrete workflows. Figure 3 depicts this as the service workflow layer considering all the possible workflows that adhere to the defined abstract workflow. Given workflow input 'X' and output 'Y' the workflow can be formally expressed as the functions {A, B, C} operating on the data communicated data:

$$W(in{:}\,x,\ out{:}\,y) = \bar{y}\left(\bar{c}\left(\bar{b}(a(x).A)\right).B\right).C$$

Figure 3 also introduces the concept that a service itself may be composed of multiple tasks that work together to provide a capability. According to the diagram we have three possible formalized implementations of service 'A'[1]:

$$A' \in A,\ A'(in{:}\,x,\ out{:}\,b) = \bar{b}((a(x).T1).T2).T3$$

$$A'' \in A,\ A'(in{:}\,x,\ out{:}\,b) = \bar{b}(a(x).T3).T4$$

$$A''' \in A,\ A'(in{:}\,x,\ out{:}\,b) = \bar{b}(a(x).T5).T4$$

Choosing a given implementation we result in a concrete workflow (with only the implementation of service A shown here) which must be mapped onto various servers which should each be modelled.

$$A'(in{:}\,x,\ out{:}\,b) = \bar{b}(\bar{t_3}(\bar{t_2}(a(x)\{t_1/a\}.T1)).T2).T3$$

B. *Simulation Environment*

The simulator developed separately as part of this research allows for a datacenter to be modelled in terms of the servers, network connections, and tasks that execute within it. It consequently allows collections of tasks to be mapped to services and therefore also workflows as depicted in Figure 3. This development was in response to the lack of simulation tools in the real-time service orientated domain as there are in the domain of cloud computing such as CloudSim [12]. Simulation forms a necessary component in analyzing large system performance due to the lack of access to truly large-scale systems and the inadequacy of using only formalization techniques to predict performance characteristics such as long-tailing [13].

Typically in web based SOA services are merely described using a name, their inputs and outputs, ignoring the importance of QoS. Various approaches have been considered such as the work by [14] which considers the impact of collaboration on task performance. Similar to our own approach the work by [15] uses a probability distribution to estimate the performance of web services (WS) and can be combined with the review of real-world WS-QoS by [16]. [17] proposes an approach that considers both the underlying infrastructure as well as the service layer to estimate response time and success rate. However, each of these approaches is not evaluated in light of real-time constraints. The work in [18] however considers non-safety-critical soft real-time in the context of media systems and allows for varying degrees of QoS to be delivered. That work, however, does not consider the impact of the underlying platform performance on capability provision.

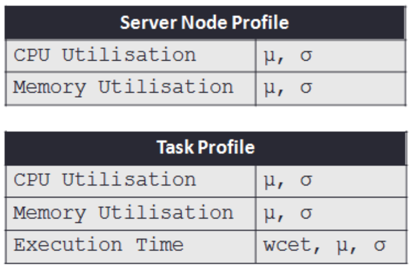
To analyze the relationship between tasks and resource as well as our ability to closely monitor the simulator we propose capturing, for each individual server within the specified datacenter and each specified task, the:

- Server cpu utilization

[1] Inter-function channels not shown

- Server memory utilization
- Task cpu utilization
- Task memory utilization
- Task execution time

As can be seen in the model depicted in Figure 4, for each of these values the mean and the standard deviation are considered providing a probabilistic model of performance and for tasks the worst case execution time is considered.



Server Node Profile	
CPU Utilisation	μ, σ
Memory Utilisation	μ, σ

Task Profile	
CPU Utilisation	μ, σ
Memory Utilisation	μ, σ
Execution Time	wcet, μ, σ

Figure 4: Probabilistic resource and performance model

IV. Methodology

Given the models described previously, this section outlines the methodology by which QoS is monitored as well as the probabilistic model which extends our prior work on M-VCR [7]. It highlights the underlying assumptions which exist in these models and consequently results of these.

A. *Assumptions*

This research is underpinned by a set of key assumptions:

- Service failures are restricted to execution and publishing faults.
- The servers and services are crash free.
- Network latencies are out of scope.
- Service discovery is not an option.

According to the SOA fault taxonomy presented by [19] there are at least 30 distinct types of service fault that can occur. We are concerned with faults relating to the specification of QoS at the publishing stage as well as the effect of performance degradation resulting in late timing.

Server failure falls outside of scope which would not result in a performance degradation of a service but rather causes an instantaneous service crash. For the same reason we assume that the implementations of the services are correct as this would invalidate the performance metrics.

Given that the network latencies within a datacenter are near negligible we assume that modelling of network latencies is out of scope at this stage of the research. Additionally there is a lack of data surrounding the actual observed latencies that exist in large scale datacenters.

Due to the limitations of the physical infrastructure and the critical real-time constraints under which we are operating we consider that mechanisms for service discovery are not relevant and would introduce additional challenges to providing a guaranteed response time. Finally an additional assumption is that the data publicly provided by Google is representative of other datacenters [20].

B. *System Model*

Figure 5 depicts the both the system model as well as the prior steps that are necessary to create a workflow:

1. Given task implementations (that already exist) need to be wrapped as services with service definitions. In our approach we use isolated performance data from these tasks to specify the resource utilization. If the services are compositions of multiple functions these must be captured prior to service wrapping.
2. Given a set of services a workflow is defined, as described earlier in this paper, and can then be passed to the system for execution.

A concrete workflow the system performs two parallel sequences of operation: the first monitoring the system state with regards to resource utilization and provides this as an update to the workflow engine.

The second thread of execution takes the system state information along with stored information about prior states and predicts the likelihood of a given next state. Through the use of M-VCR we allow for uncertainty by providing a ranked set of next states. Given the likelihood of an upcoming state the system should adjust levels of service redundancy and if possible overall system workload distribution, although we do not address this.

C. *Formalization*

In previous work [7] we mathematically modelled the likelihood of a prior result based on the accuracy of various observations we modelled the likelihood of a given result. For the purposes of performance monitoring we capture the probability of any given server's resource utilization.

We first model the prior probability of cpu and memory utilization, including the most recent observations:

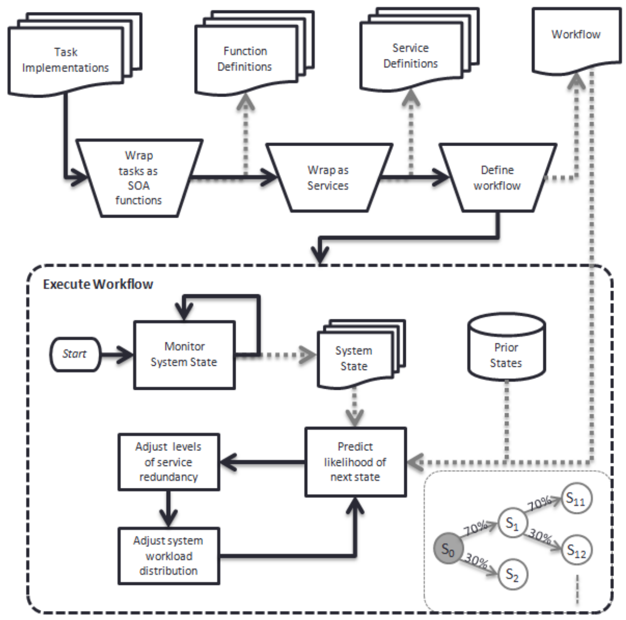


Figure 5: Proposed system model

$$P_s = f\begin{pmatrix}\mu(cpu)\\ \sigma(cpu)\\ \mu(mem)\\ \sigma(mem)\end{pmatrix}$$

This is used to provide the likelihood of the resources required by the specified task being available:

$$P_S = \left(\begin{matrix}|cpu\rangle\\ |mem\rangle\end{matrix}\middle| P_s\right)$$

From historical data we model the probability of a task meeting its deadline:

$$P_t = f\begin{pmatrix}\mu(cpu)\\ \vdots\\ wcet\end{pmatrix}$$

And the likelihood of the given task meeting its deadline given a specific hosting server is therefore:

$$P_T = P_t | P_S$$

This can be expressed completely as:

$$\therefore P_T = \left(f\begin{pmatrix}\mu(cpu)\\ \vdots\\ wcet\end{pmatrix} \middle| \left(\begin{matrix}|cpu\rangle\\ |mem\rangle\end{matrix} \middle| f\begin{pmatrix}\mu(cpu)\\ \sigma(cpu)\\ \mu(mem)\\ \sigma(mem)\end{pmatrix}\right)\right)$$

The following sections provide detail on the implementation and evaluation of the approach.

V. Implementation

This section presents our implementation of the service oriented system and the document format that is used for logging the performance of servers and tasks.

A. *Implementing the use case*

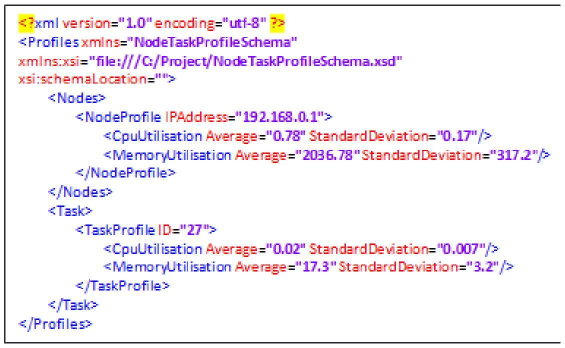
The case study touched upon earlier in this paper, formed a gesture recognition system utilizing multiple cameras to increase the reliability of results under strict real-time constraints, and consists of the following components:

- Cameras providing input data
- Various feature detectors implemented using either SURF, MSER, or ASIFT algorithms
- An aggregator that collates the results using M-VCR
- And a database that is used by the detectors for feature matching

B. *Experimental set-up*

As can be seen in Figure 6, in addition to the case study components we simulate 6 servers on which the system runs using performance profiles from real-world data. As the simulator models a physical infrastructure, we also capture the network links between servers and the network switch.

Each low level component within the simulator can produce log data in the format presented in Figure 7. For the purposes of this paper only server data, represented as system nodes, and task data is collected in the format that



```xml
<?xml version="1.0" encoding="utf-8" ?>
<Profiles xmlns="NodeTaskProfileSchema"
xmlns:xsi="file:///C:/Project/NodeTaskProfileSchema.xsd"
xsi:schemaLocation="">
    <Nodes>
        <NodeProfile IPAddress="192.168.0.1">
            <CpuUtilisation Average="0.78" StandardDeviation="0.17"/>
            <MemoryUtilisation Average="2036.78" StandardDeviation="317.2"/>
        </NodeProfile>
    </Nodes>
    <Task>
        <TaskProfile ID="27">
            <CpuUtilisation Average="0.02" StandardDeviation="0.007"/>
            <MemoryUtilisation Average="17.3" StandardDeviation="3.2"/>
        </TaskProfile>
    </Task>
</Profiles>
```

Figure 7: XML representation of the probabilistic model

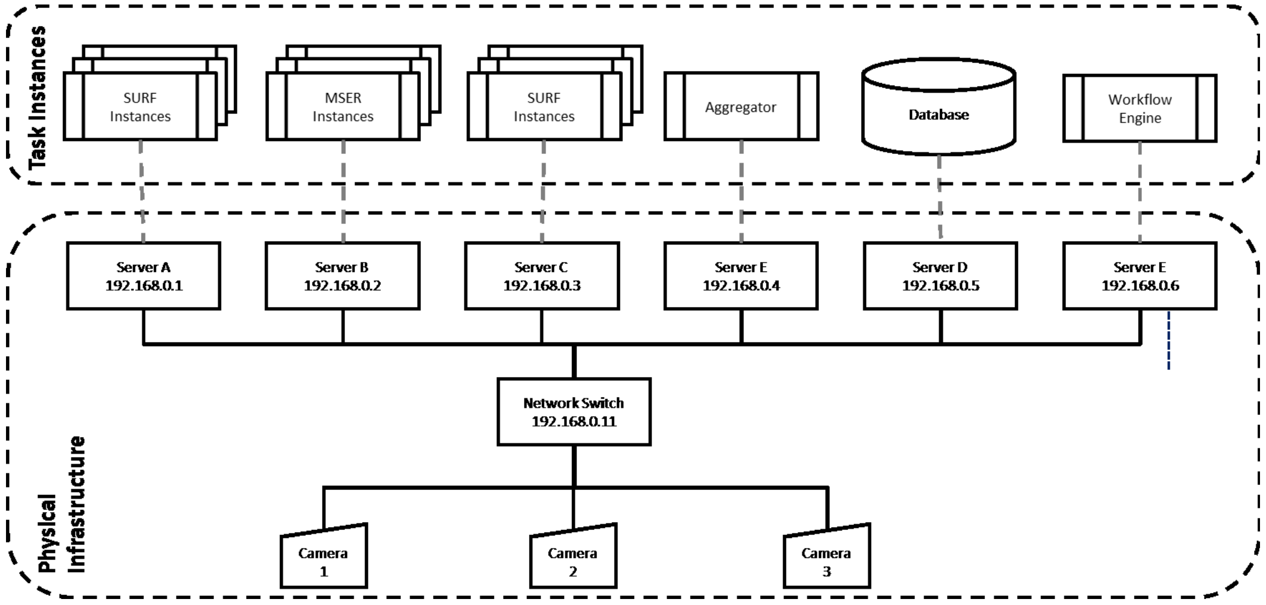
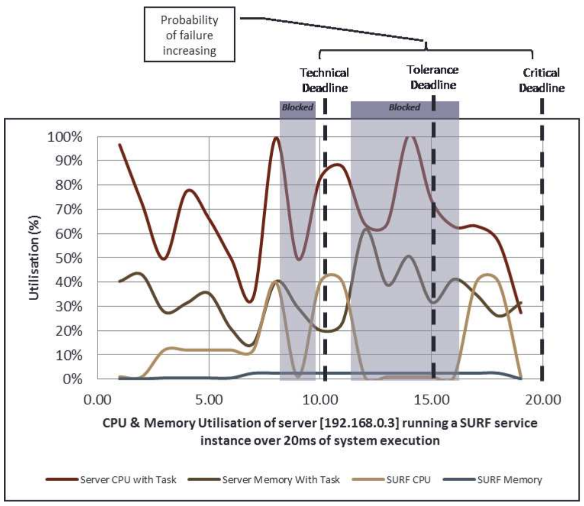


Figure 6: Experimental System Model



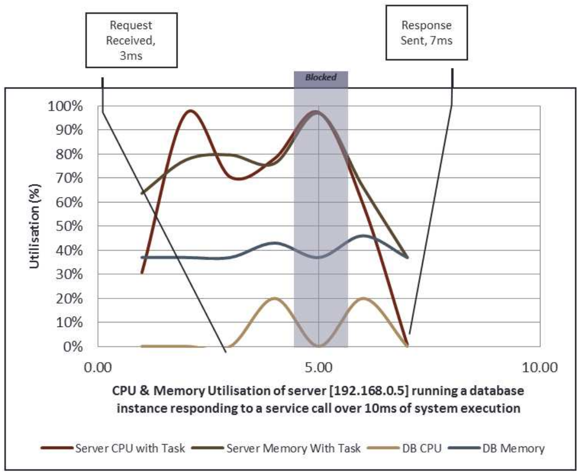


Figure 8: Task interference due to server behavior and task communication

corresponds to the earlier presented model (see Figure 4).

VI. Results

Using the developed simulator and the generated logs from monitoring both the physical infrastructure and the tasks associated with the previously specified workflow in this section we present performance analysis and an example probabilistic model of a service's behavior.

A. *Performance Interference*

Figure 8 depicts the execution time of a specific instance of a service running on server [192.168.0.3] using the SURF implementation communicating with the database executing on server [192.168.0.5]. The graphs begin at the instance the service was requested and depict:

- The server's utilization of cpu and memory with the task.
- The cpu and memory utilized by the task itself.

The particular instances shown follow the behavior demonstrated in Figure 10. It is observed that the database task is itself delayed for nearly 2ms (1ms being the fidelity of our monitoring tools) which will have a cascading effect on the execution time of the SURF service. Consequently due to interference by cohosted tasks on the server the SURF service is interrupted on two occasions resulting in a further delay in its response time of nearly 7ms.

As a result the response time observed by this service instance is just under 20ms for a single iteration of the execution loop.

B. *Average Performance*

Over approximately 15000 service calls that were made to a SURF service the average execution times for each iteration within the loop are depicted in Figure 9. Notably the average execution time is 17ms with worst case times of 30ms. [21] outlines a real-time deadline model which allows for tolerated degrees of response time:

- A technical deadline, i.e. the soft deadline
- Tolerance deadline, i.e. the firm deadline
- Critical hard deadline

Figure 8 shows these deadlines with respect to the execution of the service instance whilst Figure 9 shows the likelihood of each deadline being met:

- Only 3% of instances meet the technical deadline which is traditionally used when defining performance metrics and QoS.
- A subsequent 23% meet the tolerance deadline.
- A majority of 46% miss the above deadlines but complete within the critical time.
- The remaining 28% miss the specified deadlines.

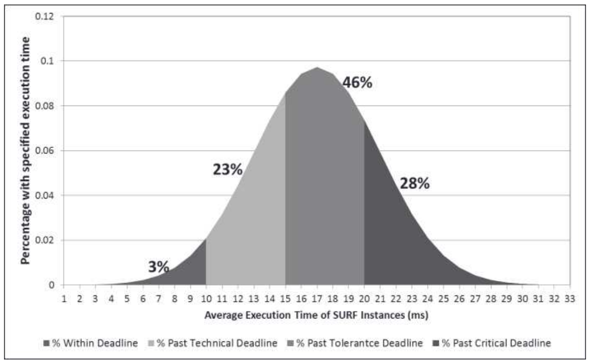


Figure 9: M-VCR generated likelihood of service performance

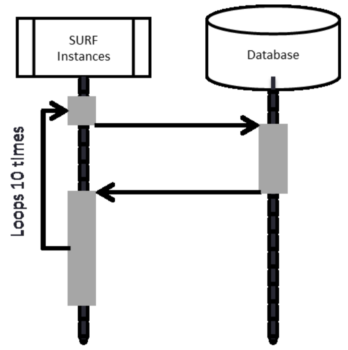


Figure 10: SURF service instance

Due to the cascading complexity of modelling each communicating service in this manner the results depicted in this paper assume that the modelled QoS for other services, excluding the SURF instances, use traditional static methods. As a result the probability of a SURF service responding within the tolerance deadline is:

$$P_T = \left(f \begin{pmatrix} 0.26 \\ 15ms \end{pmatrix} \middle| \left(\begin{matrix} |cpu: 0.43\rangle \\ |mem: 0.32\rangle \end{matrix} \middle| f \begin{pmatrix} cpu: 0.39 \\ mem: 0.22 \end{pmatrix} \right) \right)$$

$$P_T = \left(f \begin{pmatrix} 0.26 \\ 15ms \end{pmatrix} \middle| \begin{matrix} cpu: 0.45 \\ mem: 0.91 \end{matrix} \right) \leq 0.12$$

When compared with the original performance metric that specifies a WCET of 20ms we see the likelihood of actually achieving that deadline is at best 12%.

Therefore with these performances statistics it is clearly essential that a metric that considers the real-world performance of the service, given interference, needs to be defined. It is also necessary to develop mechanisms that utilize these statistics to adapt the system workload in an attempt to increase the likelihood of responding in-time.

VII. Related Work

The table depicted in Figure 11shows a comparison of our work against related state-of-the-art. Work by the likes of [22] focusses on estimating the WCET however in order to do so it requires knowledge about the internal method calls of the service which are typically not known by those responsible for deployment of the service.

The work referenced previously by [6] on the iLand project identifies several additional QoS parameters that have not been considered in our work. However their approach is limited due to the fact that those QoS values are statically defined prior to deployment. [23] proposes another approach that consider the history of execution as well as the environment in which the services must execute. Previous work does not consider that these technical defined values are not representative of real-world execution.

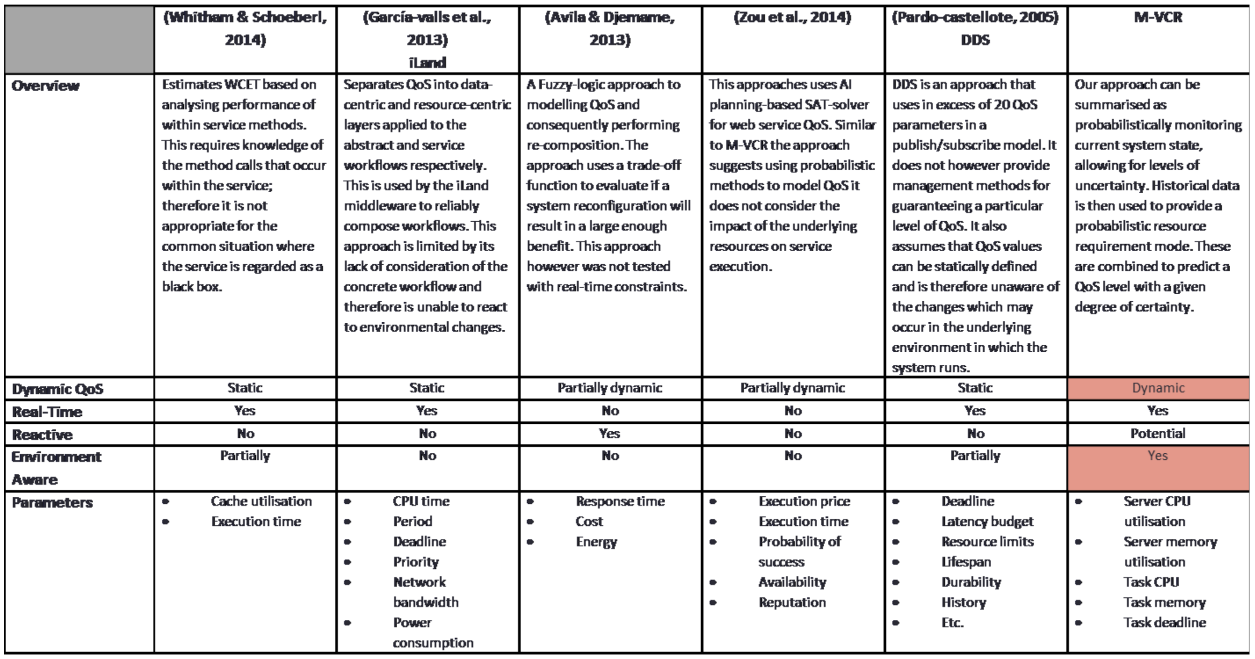
[5] and [24] both propose approaches which are partially dynamic. In the former case the a single measure of success likelihood is computed, however it does not take into account environmental conditions or therefore react to them. [24] however does provide a reactive system that uses fuzzy-logic at a high level of abstraction to dynamically define, to an extent, the level of QoS that can be delivered. This does not, however, lend itself to real-time environments where changes are required within milliseconds.

This paper has outlined a methodology for dynamically modelling QoS in a probabilistic fashion in the context of a changing environment. This work will in future empirically compare the approach against related work.

VIII. Conclusion

This paper has presented a probabilistic modelling approach to capture the relationship between service performance and the underlying environment. Furthermore, in experimental validation of the work real server utilization data has been used from both Google Cloud and also our own local server cluster. Through the use of a dedicated simulator for real-time service orientated architectures we find that services can fail to meet their advertised worst-case response times on nearly 90% of occasions. The proposed approach builds on our prior work on the M-VCR framework adapting it take into consider the perspectives:

- Individual server utilization of:
  - CPU and Memory
- Individual task utilization of:
  - CPU and Memory



	(Whitham & Schoeberl, 2014)	**(García-valls et al., 2013) iLand**	**(Avila & Djemame, 2013)**	**(Zou et al., 2014)**	**(Pardo-castellote, 2005) DDS**	**M-VCR**
Overview	Estimates WCET based on analysing performance of within service methods. This requires knowledge of the method calls that occur within the service; therefore it is not appropriate for the common situation where the service is regarded as a black box.	Separates QoS into data-centric and resource-centric layers applied to the abstract and service workflows respectively. This is used by the iLand middleware to reliably compose workflows. This approach is limited by its lack of consideration of the concrete workflow and therefore is unable to react to environmental changes.	A Fuzzy-logic approach to modelling QoS and consequently performing re-composition. The approach uses a trade-off function to evaluate if a system reconfiguration will result in a large enough benefit. This approach however was not tested with real-time constraints.	This approaches uses AI planning-based SAT-solver for web service QoS. Similar to M-VCR the approach suggests using probabilistic methods to model QoS it does not consider the impact of the underlying resources on service execution.	DDS is an approach that uses in excess of 20 QoS parameters in a publish/subscribe model. It does not however provide management methods for guaranteeing a particular level of QoS. It also assumes that QoS values can be statically defined and is therefore unaware of the changes which may occur in the underlying environment in which the system runs.	Our approach can be summarised as probabilistically monitoring current system state, allowing for levels of uncertainty. Historical data is then used to provide a probabilistic resource requirement mode. These are combined to predict a QoS level with a given degree of certainty.
Dynamic QoS	Static	Static	Partially dynamic	Partially dynamic	Static	Dynamic
Real-Time	Yes	Yes	No	No	Yes	Yes
Reactive	No	No	Yes	No	No	Potential
Environment Aware	Partially	No	No	No	Partially	Yes
Parameters	• Cache utilisation • Execution time	• CPU time • Period • Deadline • Priority • Network bandwidth • Power consumption	• Response time • Cost • Energy	• Execution price • Execution time • Probability of success • Availability • Reputation	• Deadline • Latency budget • Resource limits • Lifespan • Durability • History • Etc.	• Server CPU utilisation • Server memory utilisation • Task CPU • Task memory • Task deadline

Figure 11: Related Work Comparison

With respect to the technical soft, tolerated firm, and critical hard deadlines.

Finally this paper has demonstrated the importance of modelling service QoS with respect to its real performance on real systems as they change during the execution window.

A. Future Work

Currently the probabilistic model has only been applied to a single service in the workflow and the results have not been utilized to allow for online system adaptation. As future work, we will apply the methodology to the whole set of services in the workflow as well as consider workflows from other domains. We will empirically evaluate our approach against the state-of-the-art research using the developed simulator which is agnostic to any particular real-time SOA solution. Further work may include development of the simulator itself to provide greatly accuracy in modelling the execution times of services as well as allowing for additional environmental parameters to be considered, including but not limited to: network bandwidth and latency, inter-service communication models, and power consumption.

Acknowledgment

The work is supported in part by the National Basic Research Program of China (973) (No.2011CB302602), the U.K. EPSRC WRG platform project (No. EP/F057644/1), and other EPSRC and RC grants.

References

[1] I. Scheeren and C. E. Pereira, "Combining Model-Based Systems Engineering, Simulation and Domain Engineering in the Development of Industrial Automation Systems: Industrial Case Study," *2014 IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, pp. 40–47, Jun. 2014.

[2] M. Wagner, D. Zobel, and A. Meroth, "SODA: Service-Oriented Architecture for Runtime Adaptive Driver Assistance Systems," *2014 IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, pp. 150–157, Jun. 2014.

[3] B. Mueller, G. Viering, F. Ahlemann, and G. Riempp, "Towards Unverstanding the Sources of the Economic Potential of Service-Orientated Architecture: Findings from the Automotive and Banking Industry," in *ECIS*, 2007, pp. 1608–1619.

[4] D. McKee, D. Webster, P. Townend, and D. Battersby, "Towards a Virtual Integration Design and Analysis Enviroment for Automotive Engineering," in *Workshop on Real-Time CyberPhysical Systems*, 2014.

[5] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, and Y. Xiang, "QoS-Aware Dynamic Composition of Web Services Using Numerical Temporal Planning," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 18–31, Jan. 2014.

[6] M. García-valls, P. Basanta-val, M. Marcos, and E. Estévez, "A bi-dimensional QoS model for SOA and real-time middleware," *Int. J. Comput. Sci. Eng.*, 2013.

[7] D. McKee, P. Townend, D. Webster, and J. Xu, "M-VCR : Multi-View Consensus Recognition for Real-Time Experimentation," in *2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, 2014.

[8] Google, "Google Cluster Data V2." [Online]. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1.

[9] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," *IEEE Comput. Soc.*, vol. 40, no. 11, pp. 38–45, Nov. 2007.

[10] M. García-valls, P. Basanta-val, and I. Estévez-Ayres, "Supporting Service Composition and Real-Time Execution through Characterisation of QoS Properties," in *SEAMS '11 Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2011, pp. 110–117.

[11] M. Garcia Valls, I. R. Lopez, and L. F. Villar, "iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service Oriented Distributed Real-Time Systems," *IEEE Trans. Ind. Informatics*, vol. 9, no. 1, pp. 228–236, Feb. 2013.

[12] R. N. Calheiros, R. Ranjan, A. Beloglazov, and A. F. De Rose, "CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," no. August 2010, pp. 23–50, 2011.

[13] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, p. 74, Feb. 2013.

[14] D. Kuc, "Confucius: A Tool Supporting Collaborative Scientific Workflow Composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 2–17, Jan. 2014.

[15] H. Zheng, J. Yang, and W. Zhao, "QoS probability distribution estimation for web services and service compositions," *2010 IEEE Int. Conf. Serv. Comput. Appl.*, pp. 1–8, Dec. 2010.

[16] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 32–39, Jan. 2014.

[17] P. Mcburney, D. Efstathiou, S. Zschaler, and J. Bourcier, "Flexible QoS-Aware Service Composition in Highly Heterogeneous and Dynamic Service-Based Systems," in *IEEE Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2013, vol. 2013.

[18] P. Xue, I. Yen, and K. M. Cooper, "QoS-driven dynamic adaptation in media intensive systems," in *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2011, pp. 1–8.

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

[20] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models," *2013 IEEE Seventh Int. Symp. Serv. Syst. Eng.*, pp. 49–60, Mar. 2013.

[21] R. Kirner, "A Uniform Model for Tolerance-Based Real-Time Computing," *2014 IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, pp. 9–16, Jun. 2014.

[22] J. Whitham and M. Schoeberl, "WCET-Based Comparison of an Instruction Scratchpad and a Method Cache," *2014 IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, pp. 301–308, Jun. 2014.

[23] G. Pardo-castellote, "OMG Data-Distribution Service (DDS): Architectural Overview," 2005.

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