Intelligent Micro-Service Auto Scalar

D. Kasthurirathne[1], D.P.Liyanage[2], S. Ganeshalingam[3], A.M.Firnas[4], W.J.J.Abayarathne[5]
Faculty of Computing,Sri Lanka Institute of Information Technology, New Kandy Road, Malabe
[1]dharshana.k@sliit.lk, [2]it13148478@my.sliit.lk, [3]it14056826@my.sliit.lk, [4]it14064432@my.sliit.lk,
[5]it10230848@my.sliit.lk

Abstract—**this research paper presents a new way of scaling Docker [1] micro service instances using a reactive, machine learning model based on the HTTP request load and the resource consumption of the application. The paper presents how micro service application behaves and key performance metrics which needs be consider to scale the instances and how the auto scaling is achieved using machine learning algorithms. The initial approach of the development of this project was to collect the necessary analytical data for developing the model which was used for the testing of the existing systems. It was also necessary to develop a rule based scaling system. Based on the results of the experiment it was decided that a hybrid based approach would be more efficient compared to the systems in existence. The outcome of the project is to manage the micro services with the least interaction from the human user.**

I. INTRODUCTION

With the advancement in technology the benefit that a person gets from day to day life has been exponentially increased. Out of all the possible products which has been developed, the main field which has gone beyond a certain extent is the information technology field. Out of all the products available there is some way in which the product has been managed to a certain level by making use of some mechanism available in the IT area. The development of products related to this area, requires a proper plan. This is where the architectures of applications come into play. That is the proper planning of technology based on the consumer's need. However, in terms of building such systems which would be able to complete complex tasks with the least number of defects is extremely difficult.

Cloud computing is the most promising fields of technology available today. This basically covers the concept of internet based computing. The area of computer technology has been enhanced to make use of networks where the capability of one node can be utilized in other nodes. This means it is possible to have multiple computers connected to provide the necessary processing to complete a particular tasks. Thus it opens many areas which has not been thought of at the earlier introduction of the internet. But along with the advantages there come some disadvantages basically the balancing of the load of a server.

There are many architectures on which a server level application can be built upon. The current mainstream ones are built upon the

Service oriented architecture. The SOA is the main architecture that is being used to achieve interoperability between systems. The main purpose is to reduce the amount of dependent code that will be present in the system. This is achieved by following a standard mechanism for the data transfer. This is mainly focused on data piping. There are many other architectures which have been built upon this architecture as the base. One such example is the micro service architecture [2]. There are many advantages that are present while making use of this architecture while there are also many disadvantages. As this architecture is built with the principle of scalability as the main aspect, it is highly adaptable [4]. But since it is adaptable it becomes difficult to keep track of the services individually. Out system provides a solution for such problems. The current systems that are deployed such as the Amazon Web Services make use of the elastic balancing mechanism which makes use of virtualization for the purpose of having multiple instances of different applications to be deployed within a single server [3].This introduces the possibility of having internal traffic within the servers thus putting more load to the internal systems[4], this includes the issues which will need to be faced when thus it becomes a necessity to introduce a load balancing mechanism to handle the internal load to the split up virtual machines[5]. The elastic load balancing used by AWS makes use of different mechanism for the sole purpose for scaling [6]. The existing system makes use of variations of the heartbeat mechanism for the purpose of identifying live and dead services in the network, the algorithm currently being used for this purpose is known as SWIM[7].

To overcome such issues, the micro services architecture was built by Martin Fowler [8]. The basic idea was to have a containerized system, i.e. multiple VM's running different functionalities with the least amount of coupling possible [9]. Micro service architecture can be used to create an application where the micro service instance level coupling is too low. So that the application can be scaled as per to the requirements. But in a normal micro service based application there can be number of independent micro service instances allocated to do specific task where the application managers could not be able to identify each individual micro service instance which should be scaled up or down. This is one of the main reasons where most of the companies do not go for micro service based architectures

But with intelligent auto scaling the application itself will identify the load patterns and resource consumption of each running micro service and based on the collected data and the past data the application will auto scale which will remove the burden of management of the application from the respective engineers.

Google has implemented a platform named Kubernetes which is one of the key tools available for container orchestration [10]. It implements an auto scaling module but it is based only on the cup consumption level of micro service instances. The existing system will be able to manage the micro

services individually via rules, but this again would require human intervention for the system to function with the least downtime in production [11]. Our main project is focused on the development of a mechanism or tool which would be able to bypass this limitation without giving any disadvantage to the user.

The existing system will be able to manage a system which is developed using the micro services, this would give an overall visibility of how the system which is deployed, will function in a live environment. The current drawback which has been identified would be that the system would not be able to scale if there are requests for which it would not be able to find the pattern for. Therefore in the future it would be necessary for implementing an architecture which would be able to bypass these limitations without any constraints in place.

II. METHODOLOGY

The existing system, as described in the need for auto scaling, does not have a way of identifying the required resource allocation based on the usage of the system. The

current ways of identifying the shortest path to the service [12] make use of the orchestration of the services to identify the routes. These routing mechanisms are not being used in the existing systems to get the best execution patterns. The proposed system will be able to bypass this deficiency by providing a plugin which can be attached to any system which has the Docker system [13] involved with the micro services. The system will need to have an adjustment period of a minimum of at least 2 weeks to get the required data. Until the data has been accumulated it will function as a rule based auto scalar. The system will be composed of the following modules

A. API Gateway

The API gateway will be implemented to associate the required URL called to the required service by introducing a tagging system to the Meta data. This tag will be made use for identifying the required tasks and putting them onto the queue. This will be made use of when the results are obtained and the reply is sent back to the API gateway.

B. Message Router

The message router is the main component for uniquely identifying the tasks and to assign it to the designated queue. The queue will be built by making use of apache Kafka. A common topic will be placed to which the message router will listen to. The messages from the API gateway will be placed on this queue and the router will read its content and based on the configuration which will be set for the router, the router will identify the queue to which the message will need to be passed. After the message router picks up on the result, the result will later be forwarded back to the API gateway.

C. Apache Kafka

Apache's Kafka queue implementation will be made use of for the purpose of managing the tasks. The message router will be responsible for putting and getting the tasks from the queue. These components will be responsible for the deployment of the micro services to our customized system. After the deployment is completed the following components will be responsible for the management of the services in an intelligent manner.

D. Data Collection Module

The data collection module is responsible for the aggregation of data from the API gateway along with the resource management data from the micro services analyzing module. This will later format the data and store them inside the database for future usage. The data collection module will decide on the stream it should take based on the type of data it accesses.

E. Micro service analyzing Module

This will be a service which will be deployed in the Docker that would be able to collect data from the micro services which are currently deployed on the Docker. To state a few, the resources used based on the container and the images deployed. The Docker will provide the required information by making use of command line interface. This data will be collected individually for all the deployed containers and send to the data collection module.

F. Data Analytics and load prediction

This module will be mainly responsible for the collection of data and analysis of the URL load which enters through the API gateway. The collection of this data will take place in the Kafka queue which will be able to provide the required data dump based on the queue tasks. This includes details such as the container which previously completed the tasks for a particular tag, the time for the completion, the containers sequence through which the call took place. This among others will be responsible for the prediction of the load for the next couple of days. The load will be based on the number of service calls which would be required to complete a particular tasks. The data analytics and load prediction module will be able to analyze it in a higher level and analyses just the URL load rather than what will tend to happen to the micro services on the call.

G. Micro services instance manager

The micro services instance manager will be the module which would be able to increment and decrement the instances based on the load and other prediction based data. The Micro services instance manager will have a mechanism, a rule based management system at the initial deployment of the system which will make use of a set of predefined values for the CPU and memory to decide whether the instance should be duplicated or whether it should be removed based on the real time data feed from the Docker. After the system has aggregated the required amount of data, the micro services instance manager will be able to automatically switch to the ML based prediction and have the rule based mechanisms running in the background in case the service fails at some instance.

H. System traffic analyzing and learning module

This module is mainly responsible for generating a base m module. Since the data which is collected from the other modules do not have a common metric, it becomes necessary to build a weight based model which would be able to give a weight for the container based on the load analysis and the resource allocation analysis. The system will make use of Recurrent Neural Networks for the purpose as the RNN's are capable of recursively learning from the collected data thereby increasing the accuracy of the output results.

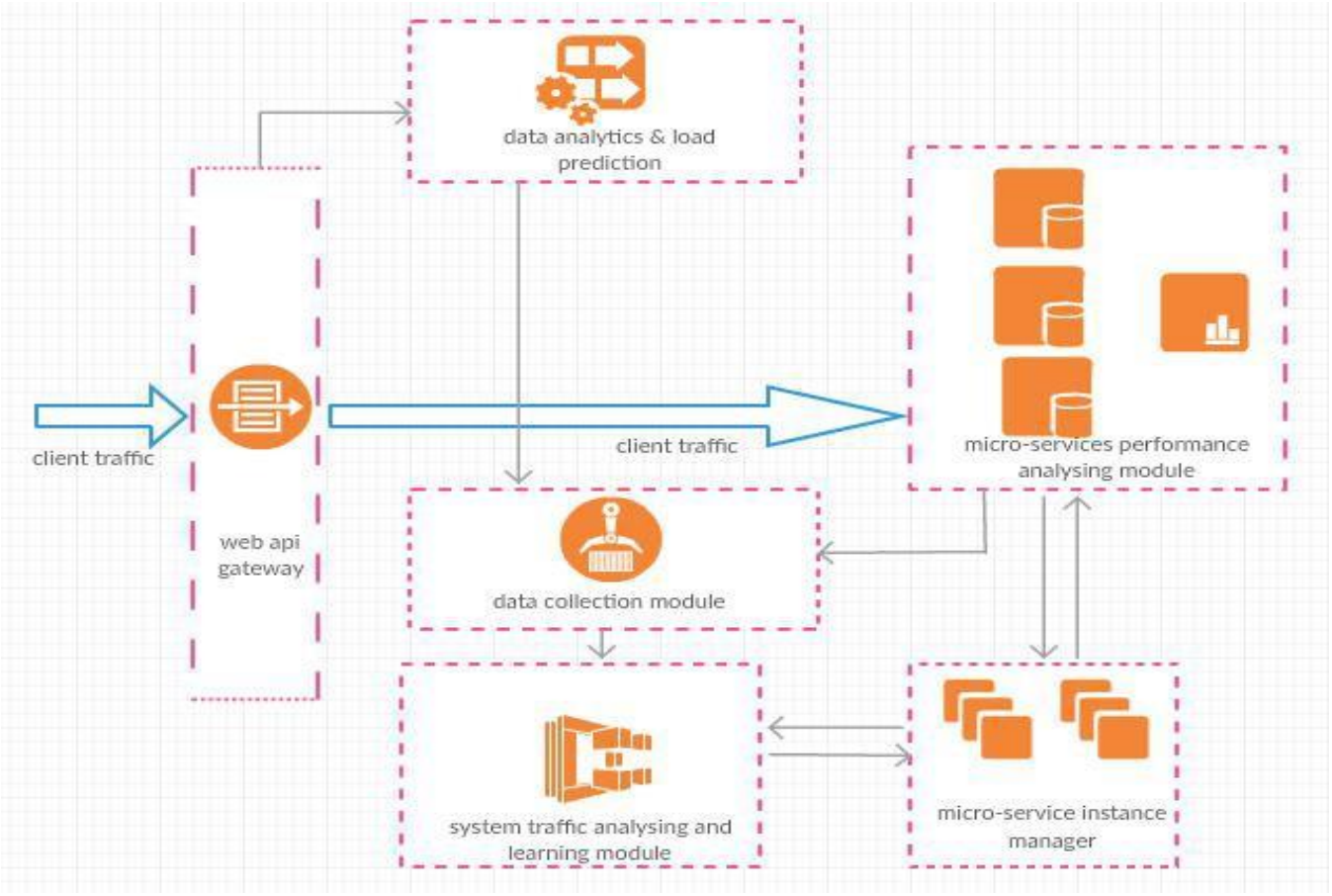

Fig. 1: High level architecture diagram

The LSTM RNN(Long Short Term Memory Recurrent Neural Networks) are able to take this a step further by allowing the model to remember the previous data, so if the normal RNN have a single layer to learn from, the LSTM RNN would have multiple layers instead with all its previous records included . This will increase the accuracy of the predictions to a higher level as they would recursively learn from the past data along with the current data. The tagging introduced to each micro services will be useful in identifying the services which are frequently called based on usage. This will also be considered while developing the weights for the containers.

I. Reporting module

The reporting module would be able to generate the necessary visuals to show the current status of the services, along with the overall architecture of how the micro services connect. It is also possible to view the sequence in which the container executed based on the URL along with finding out the services which are most latent and which are most used. The reporting module can generate the required

Overall the main system will be independent of application that will be deployed onto the Docker and the proposed system will be analyze any type of micro service that are deployed regardless of the language used as long as they are tagged.

I. Micro service framework

Below figure 2 is the high level architectural diagram which describes the message routing design of the implemented

A pseudo-code for the system where the rule based mechanism is given below.

```
Program RuleMechScaling
WHILE TRUE
        DO FOR 10 MINS
                AGGREGATE CONTAINER
                DATA; DESERIALIZE DATA;
                IF CPU_LOAD > THRESHOLD_CPU
                OR MEMORY_LOAD >
                THRESHOLD_MEM THEN
                        INCREASE CONTAINER
                        FOR IMAGE;
                END
```

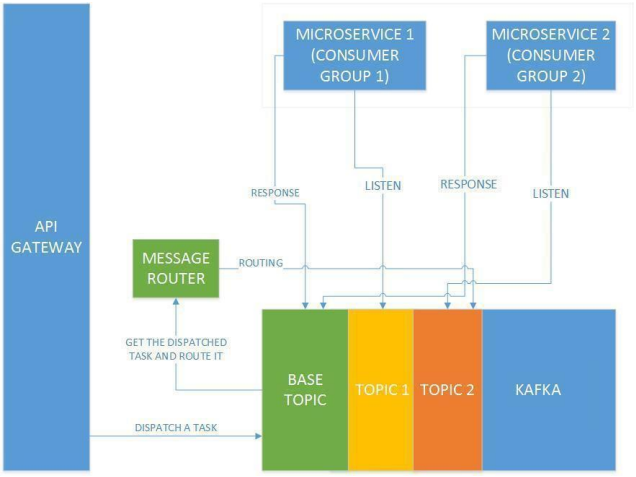

Fig. 2: Micro service framework

This is the simplified version of the algorithm that is being used in the system for the rule based mechanism. The idea behind and the machine learning model for the data to be used in the scaling of the containers. The machine learning models that are currently being used in the system are basically making use of the LSTM RNN. The short term long term recursive neural networks are basically RNN which have the ability to conditionally retain data based on the new data being fed into the model and make use of these conditions to determine the predictions for parameters passed onto the model.

The evaluations were fair, by which the deployed systems and the configurations used were the same, plus the experiment was carried out 3 times from which the average was taken. The system evaluation criteria was based on the properties which would need to be tested on a live system. This would basically include anything which is directly related to the scaling of the system

III. RESULTS AND DISCUSSION

It has been realized by the team that as the architecture which is currently in existence is not mature enough to include variations to a scale which would overall change the architecture a different approach was necessary. The existing frameworks which were used to develop the micro services were found to be full of bugs making it impossible to create a proper micro service based application. Therefore the system is required to be micro services. The services were built through a custom architecture which we build to ensure that we could the required amount of data for the development of the ML model. The data we collected proved that the existing mechanisms such as rule based. The advantage of using the system we developed would be to enable a dynamic environment which would be able to change according to the system being deployed and auto manage so there would not be less human interaction. The results we have found out is that the current system will be able to manage the micro services with the initial configuration for the orchestration deployed with the Dockerfile.

The system is being evaluated by making comparisons to existing systems which make use of the machine learning model and the rule based mechanisms individually. The system which is currently the product of this research makes use of the both of them to predict and respond on real time. The comparison chart for the system is given below.

TABLE I: Comparison chart

	Rule Based	Machine Based	Hybrid
Load to system(Analysis)	200 Requests	100 Requests	
Resource Usage	10 %	45%	55%
Scalability	600 sec	350 sec	200 sec

Based on the above comparison it is clear that the current project that has been developed yielded better results compared to the existing systems

The system was tested in varying scenarios giving out a wide range of results. When comparing the rule based and machine learning models, it is possible to see that overall the load analysis and the resource usage is a bit higher for the hybrid model, as it would do the combined tasks of both the rule based and machine learning based systems, but overall comparing the time required for the system to respond to immediate demand of the services, is significantly less for the Hybrid based model. This would basically mean at the initial stages of the system the hybrid model might take more time for processing the data but after it has properly aggregated and the predictions are made, it is possible to scale the system based on demand more efficiently.

Based on the above comparison it is clear that the current project that has been developed yielded better results compared to the existing systems.

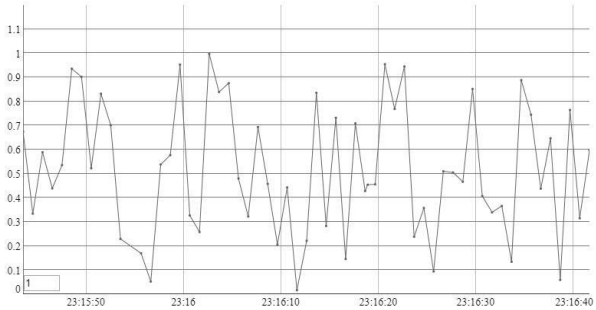

Fig. 3 cpu load-time graph (rule based load balancing)

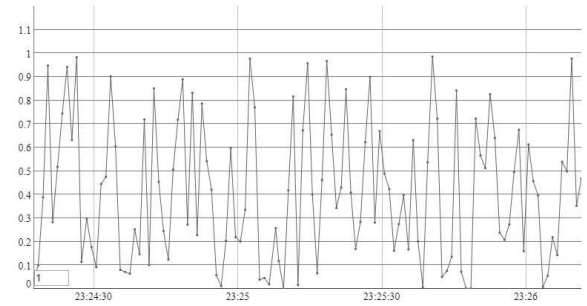

Fig. 4:cpu load-time graph (machine leaning based load balancing)

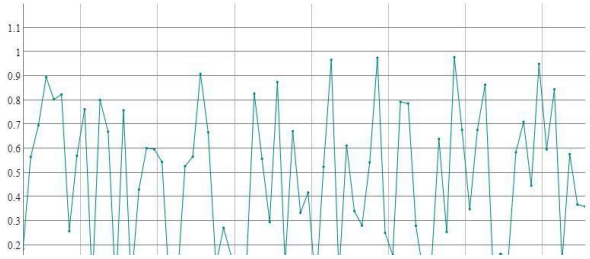

Fig. 5: cpu load-time graph (hybrid load balancing)

The above graphs are a comparison of the weightage of the average of CPU usage and the memory usage required by the system for the scaling of the services. The data has been collected from the rule based, ML based and hybrid based scaling systems for comparison purposes. This has been given a fair evaluation by providing the same specifications for all systems and running the same services with the same load for testing purposes. A client was prepared which would give out the required requests to all 3 systems in a similar manner. The graphs denote that the weightage for the ML based system happens to be more significant compared to the other 2. The hybrid which makes use of both mechanism has a faster response time compared to the others. Therefore in conclusion we can say that the hybrid based model is more efficient and less resource intensive in comparison to the ML based system. But the rule based system has the least usage of resources, but will not be nearly as accurate the hybrid which makes use of the prediction from the ML for scaling purposes.

III. CONCLUSION

The paper proposes an intelligent auto scalar for the Docker a containerized management system which is capable of the deployment of micro services. By having something like an auto scaling application it becomes possible to manage the applications built on top of the micro services architecture with minimal effort. It also becomes possible to analyze the changes in the application in real time while monitoring all the changes which take place in the application. Thus this takes out the disadvantages of having well trained teams to manage each individual micro service while also gives an insight as to how the micro services work within the system.

It also becomes possible to keep track of all the services which are being used, which are currently being tracked and make use of this information to free up server resources.

Apart from the above features it is also possible to track the redundant micro services by making use of the system. The system will be able to individually identify the services that are currently in use in real time along with the redundant service, allowing the developer to know the exact performance of the micro services and to decide whether certain micro services can be removed from the application to boost the overall performance.

REFERENCES

[1] "Docker Documentation", Docker Documentation, 2017. [Online]. Available: https://docs.docker.com/. [Accessed: 18- Aug- 2017].

[2] RICHARDSON, C. What are microservices? In-text: [6] Your Bibliography: [6]C. Richardson, "What are microservices?", *Microservices.io*, 2017. [Online]. Available: http://microservices.io/. [Accessed: 27- Mar- 2017].

[3] G. Wang and T. Ng, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center," in IEEE INFOCOM, San Diego, CA, March 2010, pp. 1–9.

[4] How to Think About AWS & Scalability," *A Comprehensive Guide to Building a Scalable Web App on Amazon Web Services - Part 1*. [Online]. Available: https://www.airpair.com/aws/posts/building-a-scalable- web-app-on-amazon-web-services-p1#2-how-to-think- about-aws-scalability. [Accessed: 27-Mar-2017].

[5] MOHAPATRA, S., SMRUTI REKHA, K. AND MOHANTY, S. A Comparison of Four Popular Heuristics for Load Balancing of Virtual Machines in Cloud ComputingIn-text: (Mohapatra, Smruti Rekha and Mohanty 33-38) Your Bibliography: Mohapatra, Subasish, K. Smruti Rekha, and Subhadarshini Mohanty. "A Comparison Of Four Popular Heuristics For Load Balancing Of Virtual Machines In Cloud Computing". International Journal of Computer Applications 68.6 (2013): 33-38. Web.

[6] "How elastic load balancing works," [Online]. Available: http://docs.aws.amazon.com/ElasticLoadBalancing/latest/ DeveloperGui de/how-elb-works.html

[7] DAAS, A., GUPTHA, I. AND MOTIVALA, A. SWIM: scalable weakly-consistent infection-style process group membership protocol - IEEE Xplore Document In-text: (Daas, Guptha and Motivala) Your Bibliography: Daas, Abinandhan, Indrnail Guptha, and Ashish Motivala. "SWIM: Scalable Weakly-Consistent Infection-Style Process Group Membership Protocol - IEEE Xplore Document". *Ieeexplore.ieee.org*. N.p., 2017. Web. 27 Mar. 2017.

[8] FOWLER, M. martinfowler.com In-text: [5] Your Bibliography: [5] M. Fowler, "martinfowler.com", *Martinfowler.com*, 2017. [Online]. Available: https://martinfowler.com. [Accessed: 27- Mar- 2017].

[9] Shridhar G.Domanal and G. Ram Mohana Reddy, "Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines" in the Proceeding of the IEEE International Conference on Communication Systems and Networks, Bangalore, Jan. 2014, pp. 1-4.

[10] MICROSERVICES IN PRACTICE - KEY

ARCHITECTURAL CONCEPTS OF AN MSA In-text: [7] Your Bibliography: [7]"Microservices in Practice - Key Architectural Concepts of an MSA", *Wso2.com*, 2017. [Online]. Available: http://wso2.com/whitepapers/microservices-in-practice-key-architectural-concepts-of-an-msa/. [Accessed: 27- Mar- 2017].

[11] Hemant S. Mahalle, Parag R. Kaveri and Vinay Chavan (2013, Jan.). Load Balancing On Cloud Data Centres. International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3(issue 1), pp. 1-4.

[12] C.E.Perkins and E. Royer. Ad hoc on-demand distance vector routing. In Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications, pages 90–100, 1999.

[13] JARMILLO, D., NGUYEN, D. V. AND SMART, R. Leveraging microservices architecture by using Docker technology - IEEE Xplore Document In-text: [4] Your Bibliography: [4]D. Jarmillo, D. Nguyen and R. Smart, "Leveraging microservices architecture by using Docker technology-IEEEXploreDocument", *Ieeexplore.ieee.org*, 2017. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=750 6647. [Accessed: 27- Mar- 2017].