A NETWORK SCIENCE BASED APPROACH FOR OPTIMAL MICROSERVICE GOVERNANCE

2020-021

Supervisor: Dr. Dharshana Kasthurirathna

Student Name	Student ID
Leader: Saranga S.A. G	IT17016230
Member 2: De Silva N.	IT17006880
Member 3: L.S. Jayasinghe	IT17012966
Member 4: M.V. Lakshitha	IT17410250

SLIIT
FACULTY OF COMPUTING

Research Problem

The research problem that is addressed in this research can be simply summarized as follows.

- Current tools and services offered by platforms such as Kubernetes fail to obtain a holistic view of microservice deployments and thereby optimize cluster performance.
- Difficultly in successfully configuring and integrating these monitoring disjoint monitoring tools with the existing tools used by organizations.
- Difficult to understand why a problem occurs even though there is knowledge that a problem has occurred.
- Constant monitoring of metrics through APM tools.

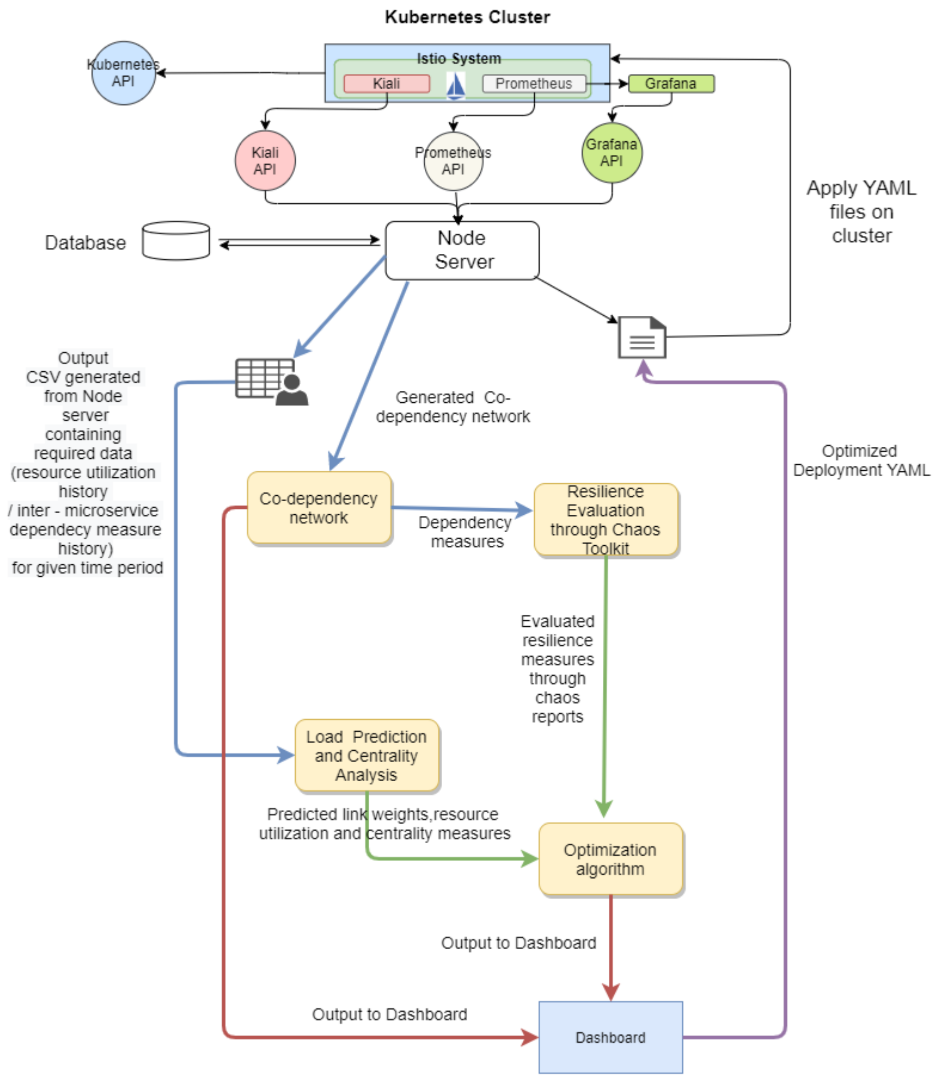
Research Problem

In short, when deploying microservices through Kubernetes, there is no efficient and effective way for developers to evaluate and monitor the effectiveness and viability of a microservice deployment and identify possible performance bottlenecks through the disjoint monitoring solutions that currently available.

Main Objective

To model a network science-based approach to govern microservice deployments through evaluation and analysis of metrics gathered, and ultimately produce a proposed model which aids to optimize microservice deployments.

Developed Solution



IT17016230 | SARANGA S.A.G

B.Sc. (Hons) Degree in Information Technology Specializing in Software Engineering

Introduction

- **Background / Research Gap** - No tools to capture the and quantify the dependencies in way to use in training prediction models.

- **Problem** - Current marketplace does not contain a way to monitor and query all the metrics regarding the network and the hardware utilization, get an idea about the whole microservice architecture and take the dependency between each and every microservice into consideration.

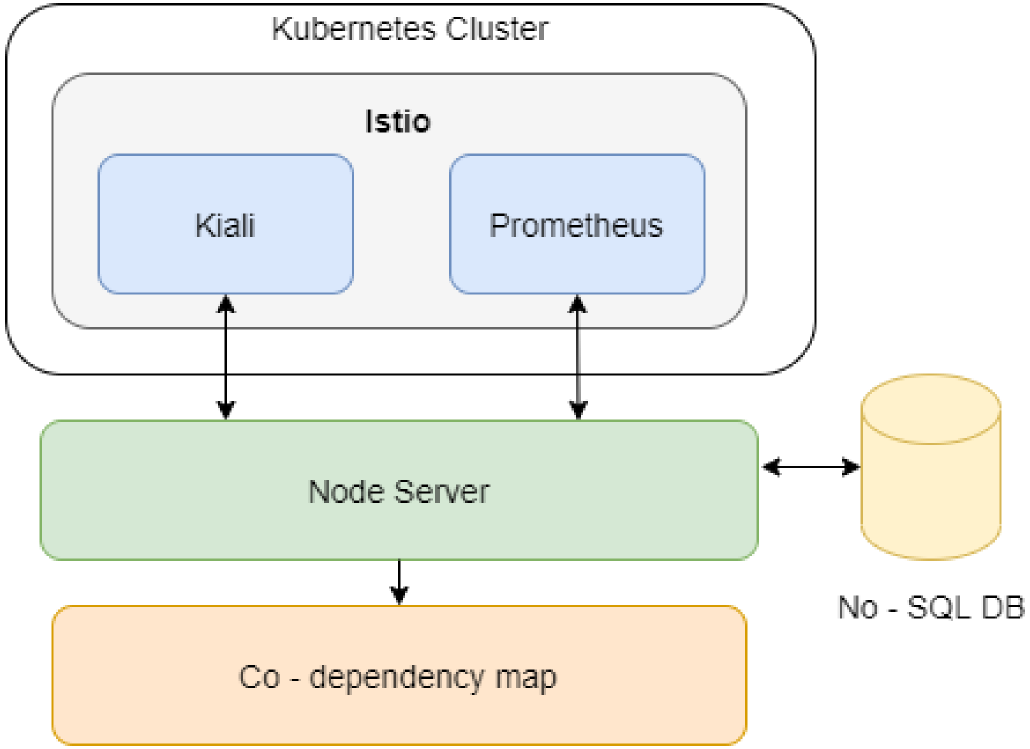
- **Primary Objective** - To generate a dependency network based on the metrics gathered from the Kubernetes cluster to provide a better understanding of the whole microservice architecture.

- **Solution** - To develop a system which can query necessary metrics in regard to network and hardware utilization of a cluster and to obtain an holistic idea about the quantified dependency between microservices.

Current Progress (90%)

- Creation and configuration of a Kubernetes cluster on AKS.
- Deployment of a sample microservice system
- Installation and configuration of Istio in the cluster and enable auto injection.
- Installation of Prometheus and Kiali and configuration to query metrics.
- Port forwarding of services to assign static ports.
- Implementation of Middle Tier Server
- Deployment of Middle Tier Server

Methodology



Results

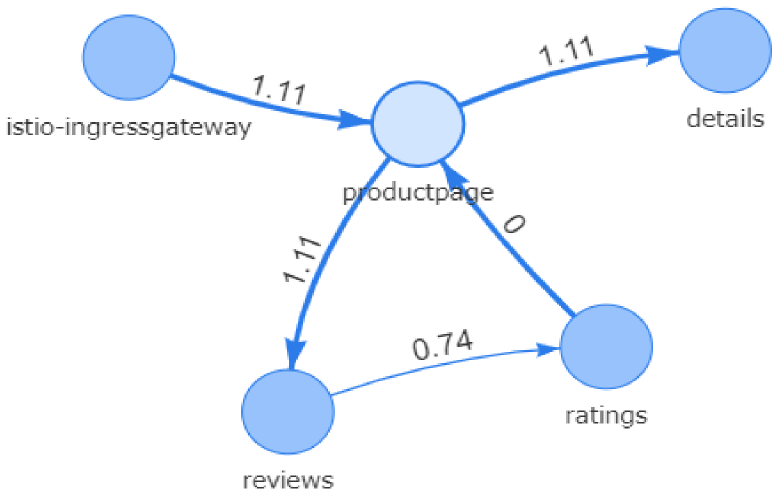
timestamp	istio-ingressgateway-produ	productpage-details	ratings-productpage	reviews-ratings	productpage-reviews
1597900834493	0.54	0.53	0.36	0	0.53
1597900894378	0.79	0.77	0.52	0	0.77
1597900954384	1.02	1.01	0.68	0	1.01
1597901015106	1.11	1.11	0.74	0	1.11
1597901074387	1.1	1.1	0.74	0	1.1
1597901134385	1.11	1.11	0.74	0	1.11
1597901194359	1.11	1.11	0.74	0	1.11
1597901254343	1.11	1.11	0.74	0	1.11
1597901314394	1.13	1.13	0.75	0	1.12
1597901374394	1.13	1.14	0.75	0	1.14
1597901434416	1.14	1.14	0.75	0	1.14
1597901494369	1.14	1.14	0.76	0	1.14
1597901554393	1.14	1.14	0.76	0	1.14
1597901614475	1.02	1.02	0.67	0	1.02
1597901674398	0.79	0.8	0.52	0	0.8
1597901734369	0.75	0.75	0.49	0	0.75
1597901794442	0.74	0.73	0.49	0	0.73
1597901854276	0.73	0.72	0.48	0	0.72
1597901914415	0.91	0.89	0.6	0	0.89
1597901974384	1.08	1.08	0.71	0	1.08
1597902034501	1.07	1.07	0.71	0	1.07
1597902094505	1.07	1.07	0.72	0	1.07
1597902154350	1.07	1.07	0.72	0	1.07
1597902214319	1.08	1.07	0.72	0	1.07

Generated quantified dependency CSV

timestamp	reviews-v2-6cb	istiod-587bbdc	coredns-869cb	prometheus-5d	dashboard-met	reviews-v1-7f6	omsagent-2hn
1597901673	0.01	0.01	0.04	0.04	0.02	0.01	0.01
1597901313	0.01	0.01	0.05	0.04	0.02	0.01	0.01
1597901193	0.01	0.01	0.05	0.04	0.02	0.01	0.01
1597901733	0.01	0.01	0.04	0.04	0.02	0.01	0.01
1597902753	0.01	0.01	0.04	0.05	0.02	0.01	0.01
1597900833	0.01	0.01	0.04	0.04	0.02	0.01	0.01
1597901553	0.01	0.01	0.05	0.04	0.02	0.01	0.01
1597901913	0.01	0.01	0.05	0.05	0.02	0.01	0.01
1597900953	0.01	0.01	0.05	0.04	0.02	0.01	0.01
1597901253	0.01	0.01	0.05	0.04	0.02	0.01	0.01
1597901793	0.01	0.01	0.04	0.05	0.02	0.01	0.01
1597902273	0.01	0.01	0.05	0.05	0.02	0.01	0.01
1597902813	0.01	0.01	0.03	0.05	0.02	0.01	0.01
1597902513	0.01	0.01	0.05	0.05	0.02	0.01	0.01

Generated CPU load CSV

RESULTS



Generated node-edge graph based on the dependency

IT17006880 | DE SILVA N.

B.Sc. (Hons) Degree in Information Technology Specializing in Software Engineering

Introduction

- **Background / Research Gap** - Presence of rule - based autoscaling strategies based on minimal infrastructure level metrics such as CPU utilization which in turn leads to the creation localized autoscaling policies.

- **Problem** - The use of localized rule-based autoscaling technologies used in microservice deployment which fail to capture a globalized perspective on the effect of autoscaling decisions.

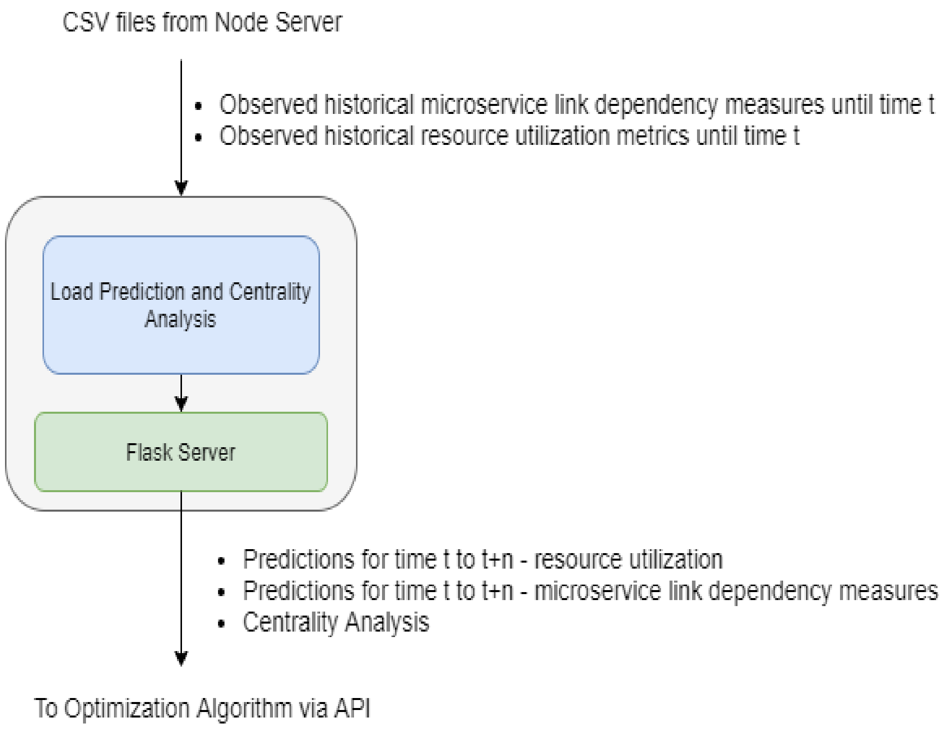
- **Primary Objective** - To facilitate the development of an improved auto-scaling policy for a Kubernetes - based microservice deployment, based on load prediction and centrality measures.

- **Solution** - Utilization of a combination of resource utilization prediction and prediction of load based metrics to facilitate an improved autoscaling policy which captures a holistic perspective on the effect of autoscaling.

Current Progress (90%)

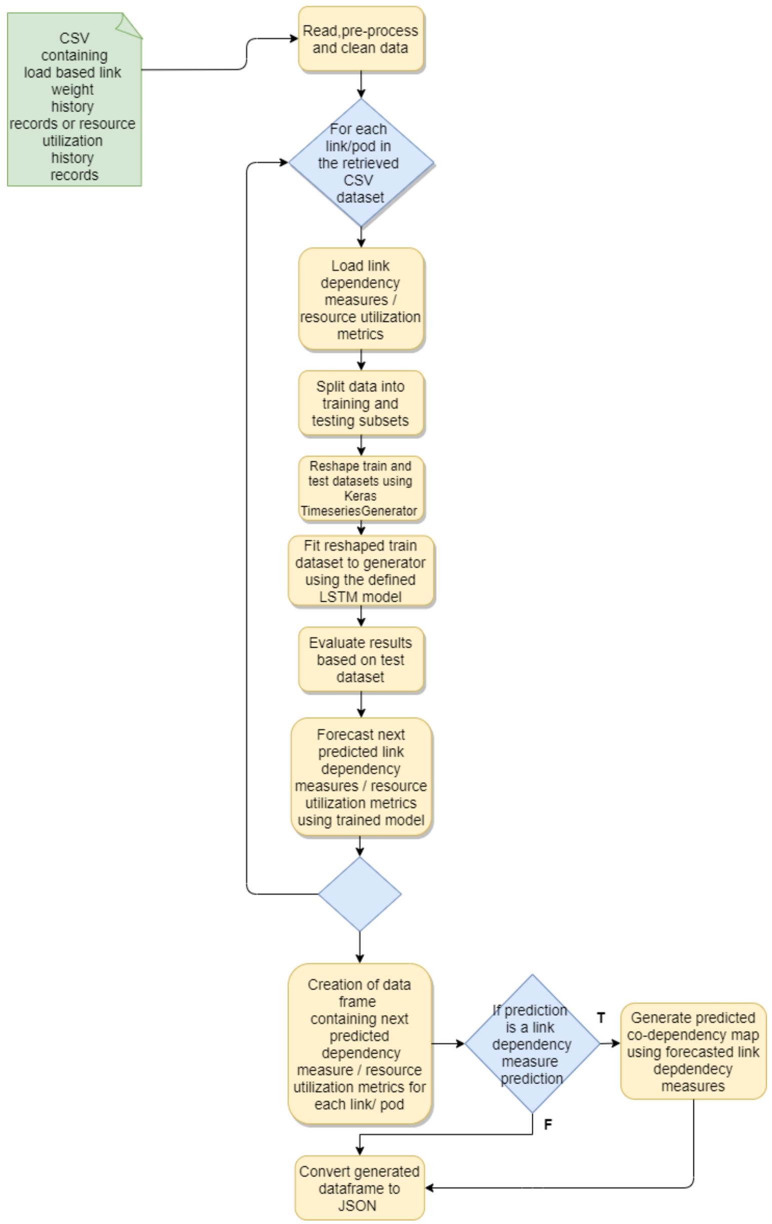
- Evaluation of various time series prediction models for resource utilization prediction and load-based inter-microservice link weight.
- Selection of the optimal prediction model from the evaluated prediction models.
- Implementation of code for resource utilization prediction and microservice link dependency measures prediction.
- Implementation of predicted co –dependency network from the predicted link dependency measures.
- Implemented methodology for calculation of centrality measures from co-dependency networks.
- Implementation of Flask server to facilitate the communication of output to other components.

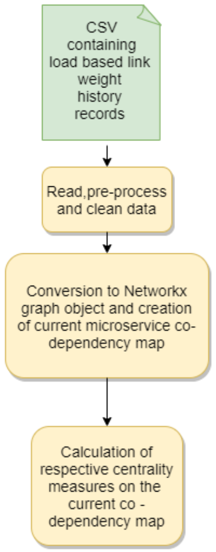
Methodology



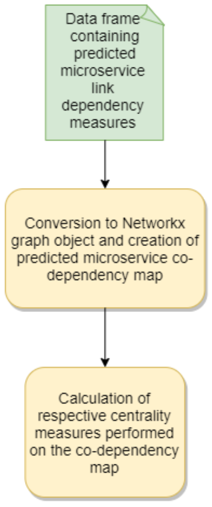
Load Prediction and Centrality Analysis Component

Methodology





Process utilized in the centrality evaluation of the co-dependency network with current microservice link dependency measures



Process utilized in the centrality evaluation of the co-dependency network of the predicted microservice link dependency measures

RESULTS – Link Dependency Measures

	A	B	C	D	E
1	link(from|to)	MAPE	RMSE	SMAPE	MASE
2	istio|ingressgateway-productpage	5.390398	0.070128	5.390398	0.377979
3	productpage|details	3.549873	0.036949	3.549873	0.257367
4	reviews|ratings	4.523298	0.034373	4.523298	0.312078
5	productpage|reviews	3.533761	0.050313	3.533761	0.271808

MAPE, RMSE, SMAPE and MASE values - resource utilization prediction

	A	B
1	link (from|to)	predictions
2	istio|ingressgateway-productpage	[0.76 1.03 1.12 1.11 1.09 1.01 0.84 0.9 0.8 0.76 0.75 0.82]
3	productpage|details	[0.56 0.97 1.09 1.11 1.11 1.04 0.88 1.07 0.74 0.71 0.68 0.79]
4	reviews|ratings	[0.54 0.7 0.74 0.77 0.77 0.69 0.54 0.55 0.41 0.34 0.49 0.67]
5	productpage|reviews	[0.91 0.99 1.1 1.11 1.08 1.03 0.77 0.71 0.38 0.39 1.23 0.98]

1-hour forecast of load-based inter-microservice link dependency measures

RESULTS – Resource Utilization Prediction

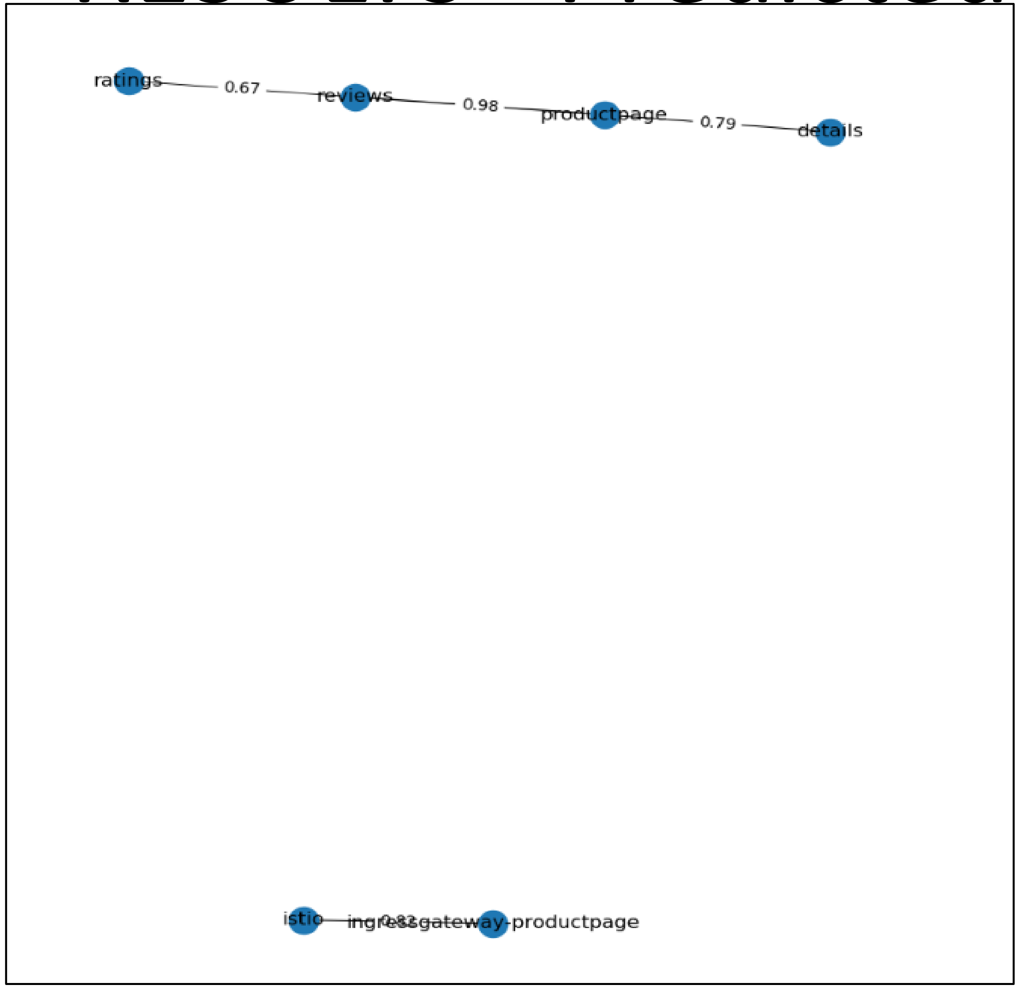
	A	B	C	D	E
1		MAPE	RMSE	SMAPE	MASE
2	kube-proxy-ld9rw	3.14644832	0.00013838	3.146448	1.312559
3	istiod-587bbdc654-w5lw4	7.13514631	0.00043847	7.135146	1.409351
4	coredns-869cb84759-wtv4b	0.78381415	9.72E-05	0.783814	0.823703
5	productpage-v1-64794f5db4-mv8lc	9.12748347	0.00424272	9.127483	1.177343
6	kube-proxy-rcnj6	2.90774165	0.00016373	2.907742	1.265455
7	coredns-autoscaler-5b867494f-mp4j7	5.55992745	2.79E-05	5.559927	1.369627
8	prometheus-5db67458fb-qmfdg	7.01766096	0.0036571	7.017661	2.848922
9	omsagent-rs-8569fb99f4-52fqk	4.67680256	0.0010736	4.676803	3.255847
10	ratings-v1-c6cdf8d98-7b9hc	6.32166424	0.0005966	6.321664	1.291949
11	details-v1-5974b67c8-qj7mx	8.51347963	0.00082289	8.51348	1.206416
12	tunnelfront-544444f4c7-z6jvl	1.12882351	0.00223926	1.128824	1.339517
13	grafana-5dc4b4676c-2mdb9	6.87129518	0.00032174	6.871295	3.217171
14	omsagent-2hn87	3.8461441	0.0007724	3.846144	1.431958
15	istio-ingressgateway-6bff4c6896-sc7d5	5.83544149	0.00071254	5.835441	0.839115
16	istio-egressgateway-6d98c888ff-wmmxn	1.45952086	8.86E-05	1.459521	1.447872
17	reviews-v3-cc56b578-6xhlm	9.14165889	0.00192517	9.141659	1.988966
18	istio-tracing-8584b4d7f9-8zdgl	111.815226	0.00807655	111.8152	6.027805
19	dashboard-metrics-scraper-566c858889-jq6jc	9.3964056	3.38E-05	9.396406	0.786595
20	kubernetes-dashboard-7f7d6bbd7f-vhc6r	12.7499743	0.00018396	12.74997	1.282703
21	omsagent-v55np	6.18248482	0.00151792	6.182485	1.02715
22	metrics-server-6cd7558856-mvnjn	3.45426686	6.31E-05	3.454267	1.328064
23	coredns-869cb84759-xwk2p	2.23875178	0.00022369	2.238752	0.761145
24	kiali-6f457f5964-hh7tv	9.37648456	0.00066369	9.376485	1.004612
25	reviews-v2-6cb6ccd848-6f757	14.6239628	0.00447454	14.62396	1.817528
26	reviews-v1-7f6558b974-fzb5k	13.7027242	0.00201316	13.70272	3.335447

MAPE, RMSE, SMAPE and MASE values - inter-microservice link dependency measures

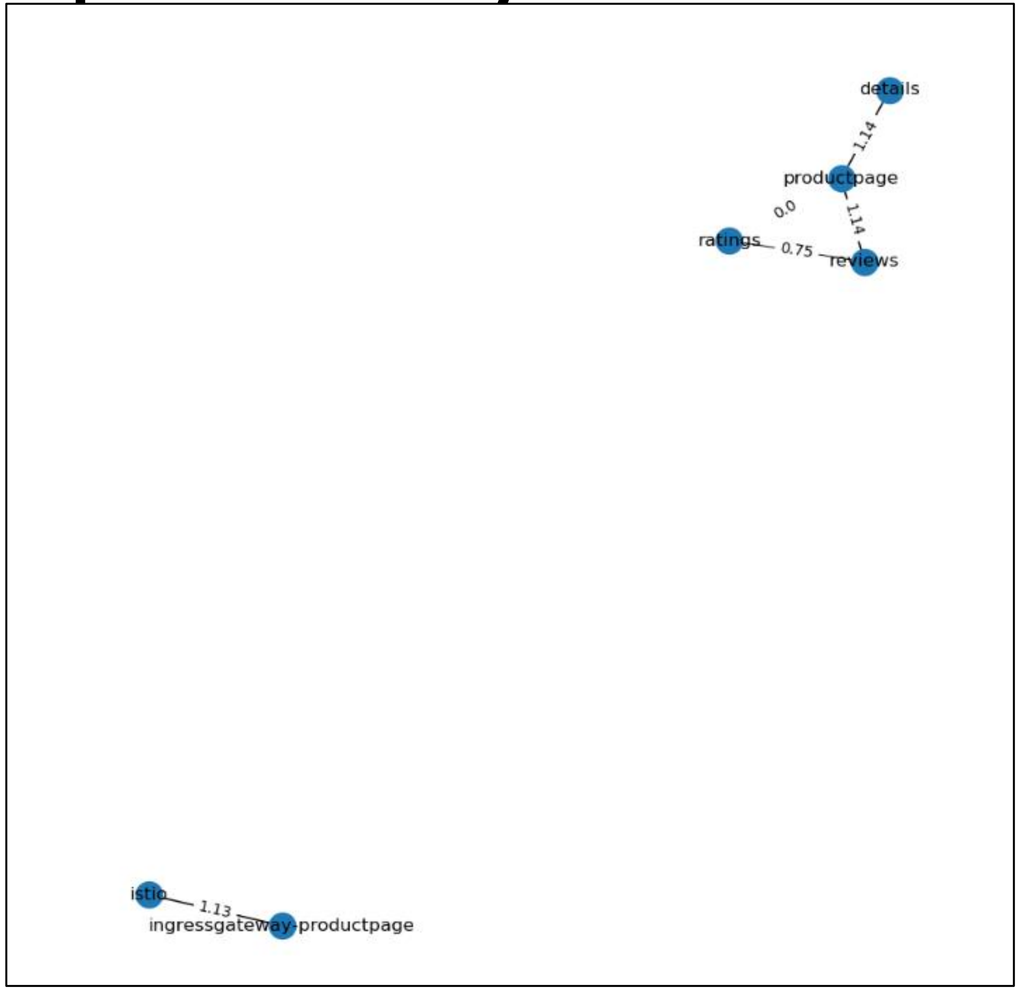
	A	B
1		predictions
2	kube-proxy-ld9rw	[0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042]
3	istiod-587bbdc654-w5lw4	[0.0066 0.0061 0.0062 0.0061 0.0062 0.0061 0.0061 0.0061 0.0061 0.0061 0.0061 0.0061]
4	coredns-869cb84759-wtv4b	[0.0093 0.0092 0.0091 0.0091 0.0091 0.009 0.0089 0.0089 0.0089 0.0089 0.009 0.0092]
5	productpage-v1-64794f5db4-mv8lc	[0.0342 0.0338 0.0342 0.0352 0.0368 0.0387 0.0409 0.0432 0.0457 0.0481 0.0509 0.0531]
6	kube-proxy-rcnj6	[0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042 0.0042]
7	coredns-autoscaler-5b867494f-mp4j7	[0.0004 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005 0.0005]
8	prometheus-5db67458fb-qmfdg	[0.0488 0.049 0.0506 0.0534 0.0597 0.0746 0.1103 0.1959 0.3807 0.788 1.7352 3.7833]
9	omsagent-rs-8569fb99f4-52fqk	[0.0206 0.0206 0.0205 0.0202 0.0202 0.02 0.02 0.0199 0.0199 0.0198 0.0198 0.0198]
10	ratings-v1-c6cdf8d98-7b9hc	[0.0085 0.0086 0.0086 0.0086 0.0086 0.0086 0.0086 0.0086 0.0086 0.0085 0.0085 0.0084]
11	details-v1-5974b67c8-qj7mx	[0.0073 0.0075 0.0078 0.0084 0.0093 0.0107 0.0128 0.016 0.0206 0.0265 0.0332 0.0354]
12	tunnelfront-544444f4c7-z6jvl	[0.1716 0.1714 0.1713 0.1712 0.1711 0.171 0.1709 0.1708 0.1707 0.1707 0.1706 0.1706]
13	grafana-5dc4b4676c-2mdb9	[0.0048 0.0047 0.0046 0.0046 0.0045 0.0045 0.0045 0.0045 0.0045 0.0045 0.0045 0.0044]
14	omsagent-2hn87	[0.0172 0.0168 0.0164 0.016 0.0157 0.0153 0.0152 0.0155 0.0171 0.0221 0.0353 0.0569]
15	istio-ingressgateway-6bff4c6896-sc7d5	[0.0088 0.0092 0.0097 0.0105 0.0113 0.0122 0.013 0.0138 0.0141 0.0143 0.0131 0.0108]
16	istio-egressgateway-6d98c888ff-wmmxn	[0.0055 0.0055 0.0055 0.0055 0.0055 0.0055 0.0055 0.0055 0.0055 0.0055 0.0055 0.0055]
17	reviews-v3-cc56b578-6xhlm	[0.0121 0.0112 0.01 0.0085 0.0064 0.0034 -0.0014 -0.0091 -0.0224 -0.0445 -0.0775 -0.1217]
18	istio-tracing-8584b4d7f9-8zdgl	[0.0164 0.0197 0.0233 0.0273 0.0325 0.0379 0.0431 0.0507 0.0604 0.0735 0.0892 0.1082]
19	dashboard-metrics-scraper-566c858889-jq6jc	[0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004 0.0004]
20	kubernetes-dashboard-7f7d6bbd7f-vhc6r	[0.0012 0.0012 0.0012 0.0012 0.0012 0.0012 0.0012 0.0012 0.0012 0.0012 0.0012 0.0012]
21	omsagent-v55np	[0.0179 0.0185 0.0189 0.0191 0.0192 0.0191 0.0191 0.019 0.0189 0.0189 0.0184 0.0181]
22	metrics-server-6cd7558856-mvnjn	[0.0016 0.0016 0.0016 0.0016 0.0016 0.0016 0.0016 0.0016 0.0016 0.0016 0.0016 0.0016]
23	coredns-869cb84759-xwk2p	[0.0094 0.0094 0.0093 0.0093 0.0092 0.0091 0.0091 0.0089 0.0089 0.0089 0.009 0.0091]
24	kiali-6f457f5964-hh7tv	[0.0069 0.007 0.007 0.0071 0.0071 0.0072 0.0072 0.0073 0.0073 0.0074 0.0075 0.0076]
25	reviews-v2-6cb6ccd848-6f757	[0.0116 0.0108 0.01 0.0091 0.0081 0.007 0.0058 0.0044 0.0027 0.0006 -0.0018 -0.005]
26	reviews-v1-7f6558b974-fzb5k	[0.0096 0.0093 0.009 0.0087 0.0083 0.0079 0.0074 0.0069 0.0062 0.0054 0.0042 0.0028]

1-hour forecast (t+12) of CPU utilization metrics

RESULTS –Predicted Co – dependency Network



Predicted microservice co-dependency network at time t+12



Microservice co-dependency network at time t

RESULTS – Centrality Measures | Predicted vs Current Co-dependency Network

	Deg.		Clos.		Bet.		Eigen.		Weighted Eigen.		Weighted Bet.	
istio	C	F	C	F	C	F	C	F	C	F	C	F
	0.2	0.2	0.2	0.2	0	0	3.2E-6	8.2E-6	1.1E-5	7.6e-6	0	0
ingress gateway-productpage	0.2	0.2	0.2	0.2	0	0	3.2E-6	8.2E-6	1.1E-5	7.6e-6	0	0
productpage	0.6	0.4	0.6	0.45	0.2	0.2	0.61	0.60	0.67	0.64	0.3	0.2
details	0.2	0.2	0.36	0.3	0	0	0.28	0.37	0.44	0.37	0	0
reviews	0.4	0.4	0.45	0.45	0	0.2	0.52	0.60	0.55	0.60	0	0.2
ratings	0.4	0.2	0.45	0.3	0	0	0.52	0.37	0.21	0.29	0.3	0

- **C** – Centrality measure for co – dependency network at time t
- **F** – Centrality measure for predicted co - dependency network at time t+12

Deg. – Degree Centrality
Clos. – Closeness Centrality
Bet. – Betweenness Centrality
Eigen. – Eigenvector Centrality

IT17012966 | L. S JAYASINGHE

B.Sc. (Hons) Degree in Information Technology Specializing in Software Engineering

Introduction

- **Background / Research Gap** - Current orchestration tools such as Kubernetes fail to create optimized deployment and autoscaling strategies, based on a holistic perspective of microservice deployments.

- **Problem** - Increase performance and availability in microservice application.

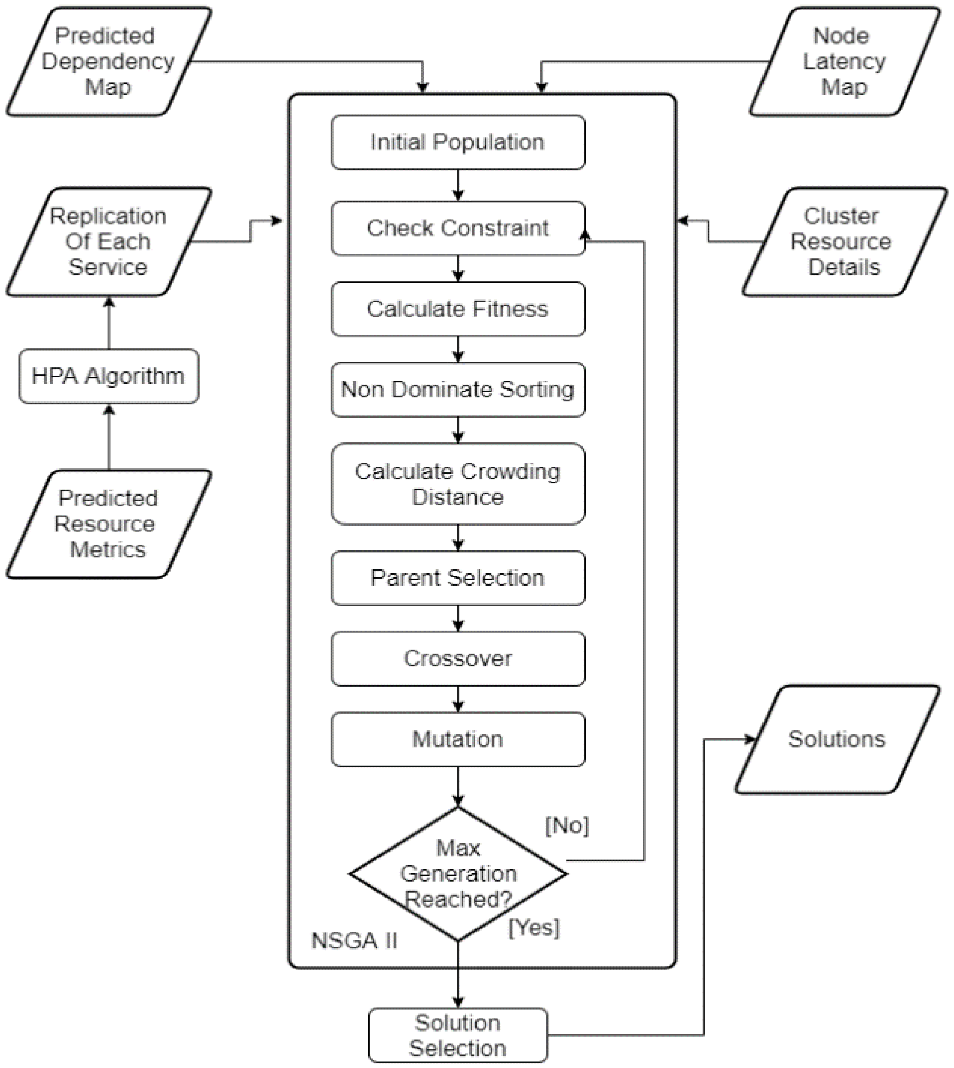
- **Primary Objective** - To identify key factors that lead to performance reduction in microservice deployments and come up with an optimal deployment strategy.

- **Solution** - Optimal placement for microservices.

Current Progress (90%)

- NSGA II
- Performance fitness
- Availability fitness
- Separate the solution
- API server
- Deployment view
- HPA Algorithm

Methodology



Methodology

TO	FROM	PREDICTED DEPENDENCY MEASURE
M0	M2	100
M1	M3	50
M4	M5	20

"pod_dependency_map": {"[0, 2]": 100, "[1, 3]": 50, "[4, 5]": 20}

RESULTS – Best Performance

Deployment Stratergy

N0	N1	N2
M1	M4	M0
M1	M4	M0
M1	M4	M0
M1	M4	M0
M1	M5	M0
M1	M5	M0
M1	M5	M0
M1	M5	M0
M3		M2
M3		M2
M3		M2
M3		M2
cpu:3800 milicores 15.79%free	cpu:1900 milicores 15.79%free	cpu:3800 milicores 15.79%free
memory:16000MB 85.00%free	memory:4000MB 60.00%free	memory:16000MB 85.00%free
storage:16000MB 77.50%free	storage:4000MB 40.00%free	storage:16000MB 77.50%free
ip:10.240.0.4	ip:10.240.0.5	ip:10.240.0.6
cost:0.166$	cost:0.0416$	cost:0.166$

Service Index	name	cpu	memory	solution instances	required instances	instances satisfaction
M0	order	300	200	8	4	True
M1	employee	300	200	8	4	True
M2	payment	200	200	4	2	True
M3	salary	200	200	4	2	True
M4	information	200	200	4	2	True
M5	supplier	200	200	4	2	True

Metrics	Exisiting	Solution	Improvement
Availability Measurement	0.000	3.144	increase by 2.000 x times
Average Link Network Latency	2.448	0.100	reduce by 24.480 x times
Ratio Difference	0	0	reduce by 0.000 difference
Scalability	not measured	not measured	not measured
Cost	not measured	not measured	not measured

RESULTS – Highest Availability

Deployment Stratergy

N0	N1	N2
M0	M0	M0
M1	M1	M0
M1	M1	M0
M1	M4	M0
M1	M4	M0
M1	M4	M0
M1	M5	M2
M1		M2
M3		M2
M3		M2
M3		M4
M3		M5
M3		M5
		M5
cpu:3800 milicores 10.53%free	cpu:1900 milicores 10.53%free	cpu:3800 milicores 10.53%free
memory:16000MB 83.75%free	memory:4000MB 65.00%free	memory:16000MB 82.50%free
storage:16000MB 75.63%free	storage:4000MB 47.50%free	storage:16000MB 73.75%free
ip:10.240.0.4	ip:10.240.0.5	ip:10.240.0.6
cost:0.166$	cost:0.0416$	cost:0.166$

Service Index	name	cpu	memory	solution instances	required instances	instances satisfaction
M0	order	300	200	8	4	True
M1	employee	300	200	9	4	True
M2	payment	200	200	4	2	True
M3	salary	200	200	5	2	True
M4	information	200	200	4	2	True
M5	supplier	200	200	4	2	True

Metrics	Exisiting	Solution	Improvement
Availability Measurement	0.000	3.144	increase by 2.125 x times
Average Link Network Latency	2.448	0.651	reduce by 3.760 x times
Ratio Difference	0	2.5	reduce by -2.500 difference
Scalability	not measured	not measured	not measured

RESULTS – Most Cost - Effective

Deployment Stratergy

N0	N1	N2
M0		M1
M0		M1
M0		M1
M0		M1
M0		M1
M0		M1
M2		M2
M2		M3
M3		M5
M3		M5
M4		M5
M4		
M4		
cpu:3800 milicores 15.79%free	cpu:1900 milicores 100.00%free	cpu:3800 milicores 26.32%free
memory:16000MB 83.75%free	memory:4000MB 100.00%free	memory:16000MB 86.25%free
storage:16000MB 75.63%free	storage:4000MB 100.00%free	storage:16000MB 79.38%free
ip:10.240.0.4	ip:10.240.0.5	ip:10.240.0.6
cost:0.166$	cost:0.0416$	cost:0.166$

Service Index	name	cpu	memory	solution instances	required instances	instances satisfaction
M0	order	300	200	6	4	True
M1	employee	300	200	6	4	True
M2	payment	200	200	3	2	True
M3	salary	200	200	3	2	True
M4	information	200	200	3	2	True
M5	supplier	200	200	3	2	True

Metrics	Exisiting	Solution	Improvement
Availability Measurement	0.000	3.084	increase by 1.500 x times
Average Link Network Latency	not measured	not measured	reduce by not measured x times
Ratio Difference	0	0	reduce by 0.000 difference
Scalability	not measured	not measured	not measured
Cost	0.374	0.332	reduce by 0.042 $

IT1741250 | LAKSHITHA M.V

B.Sc. (Hons) Degree in Information Technology Specializing in Software Engineering

Introduction

- **Background / Research Gap** - Lack of focus on resiliency measures when considering the microservice deployment and governance.

- **Problem –** How the resiliency evaluation can help in achieving a better microservice governance?

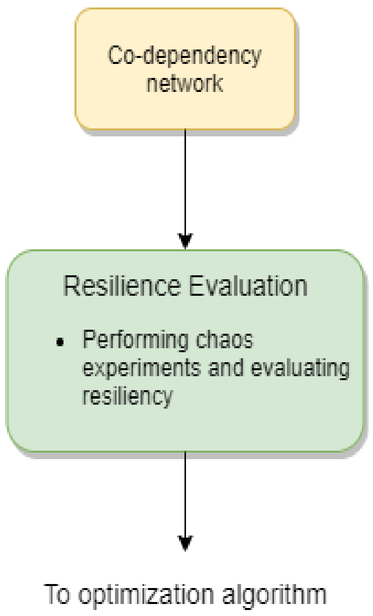
- **Primary Objective** - To perform a resiliency evaluation on Kubernetes deployments using the principles of chaos engineering.

- **Solution** - Perform resiliency evaluations and exploit the results on creating a deployment plan for an optimal microservice governance.

Current Progress (90%)

- Identifying the impact of resiliency measures on microservice governance.
- Studying about chaos engineering concepts and tools.
- Performing chaos experiments on Kubernetes cluster.
- Initial implementation of dashboard

Research Methodology



RESULTS

```yaml
version: 1.0.0
title: What happens if we terminate an instance of the application?
description: If an instance of the application is terminated, the applications as a whole should still be operational.
tags:
- k8s
- pod
- deployment
ingress_host:
    type: env
    key: INGRESS_HOST
steady-state-hypothesis:
    title: The app is healthy
    probes:
    - name: app-respnds-to-requests
      type: probe
      tolerance: 200
      provider:
        type: http
        timeout: 3
        verify_tls: false
        url: http://${ingress_host}/productpage
        headers:
            host: default
method:
- type: action
  name: terminate-pod
  provider:
    type: python
    module: chaosk8s.pod.actions
    func: terminate_pods
    arguments:
        name_pattern: ratings-v1
  pauses:
    after: 2
```

Sample Chaos Experiment

Experiment

What happens if we terminate an instance of the application?

If an instance of the application is terminated, a new instance should be created

Summary

Status	completed
Tagged	k8s, pod, deployment
Executed From	DESKTOP-29JARHE
Platform	Linux-4.4.0-18362-Microsoft-x86_64-with-glibc2.29
Started	Sun, 20 Sep 2020 21:54:48 GMT
Completed	Sun, 20 Sep 2020 21:55:06 GMT
Duration	18 seconds

Definition

The experiment was made of 1 actions, to vary conditions in your system, and 0 probes, to collect objective data from your system during the experiment.

Steady State Hypothesis

The steady state hypothesis this experiment tried was "**The app is healthy**".

Before Run

The steady state was verified

Probe	Tolerance	Verified
all-apps-are-healthy	True	True

After Run

The steady state was verified

Probe	Tolerance	Verified
all-apps-are-healthy	True	True

Method

The experiment method defines the sequence of activities that help gathering evidence towards, or against, the hypothesis.

Chaos Report

Commercialization - Market and Stats

63% of Enterprises Are Adopting Microservices

58% of companies are using Kubernetes in production

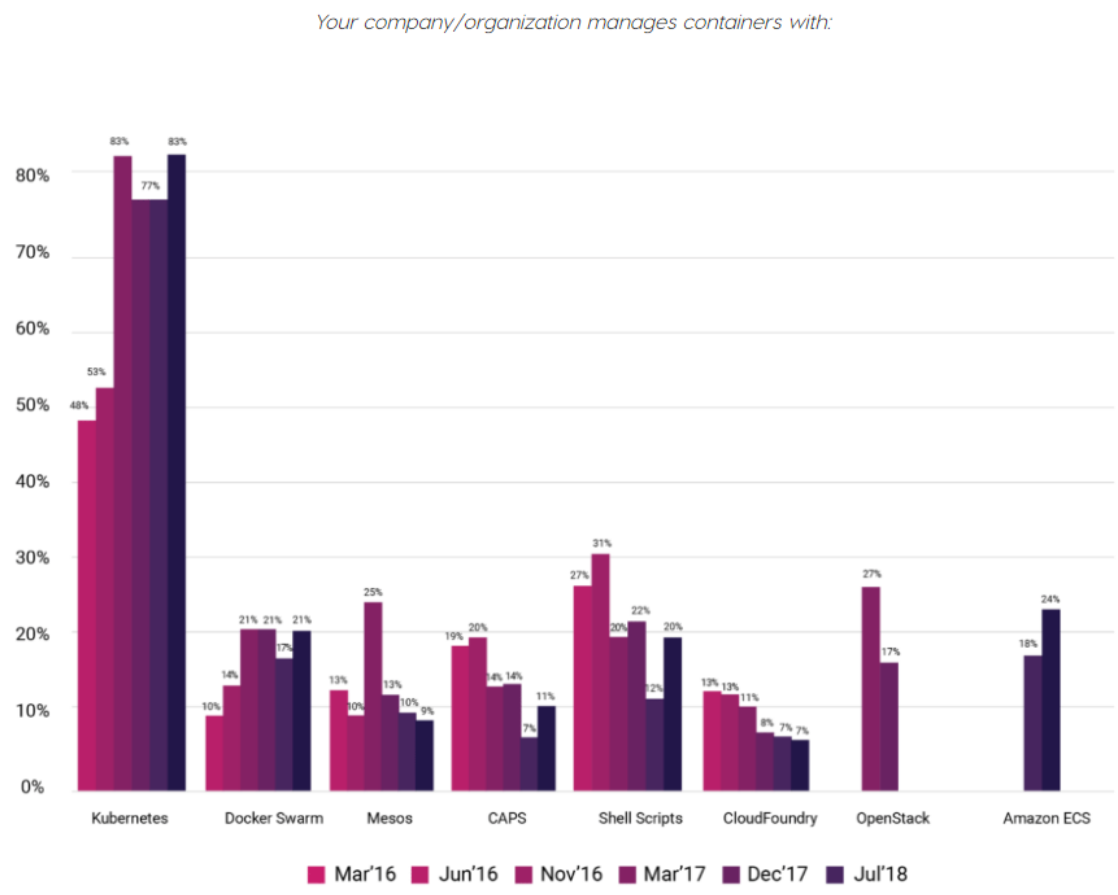
32% are evaluating Kubernetes for future use

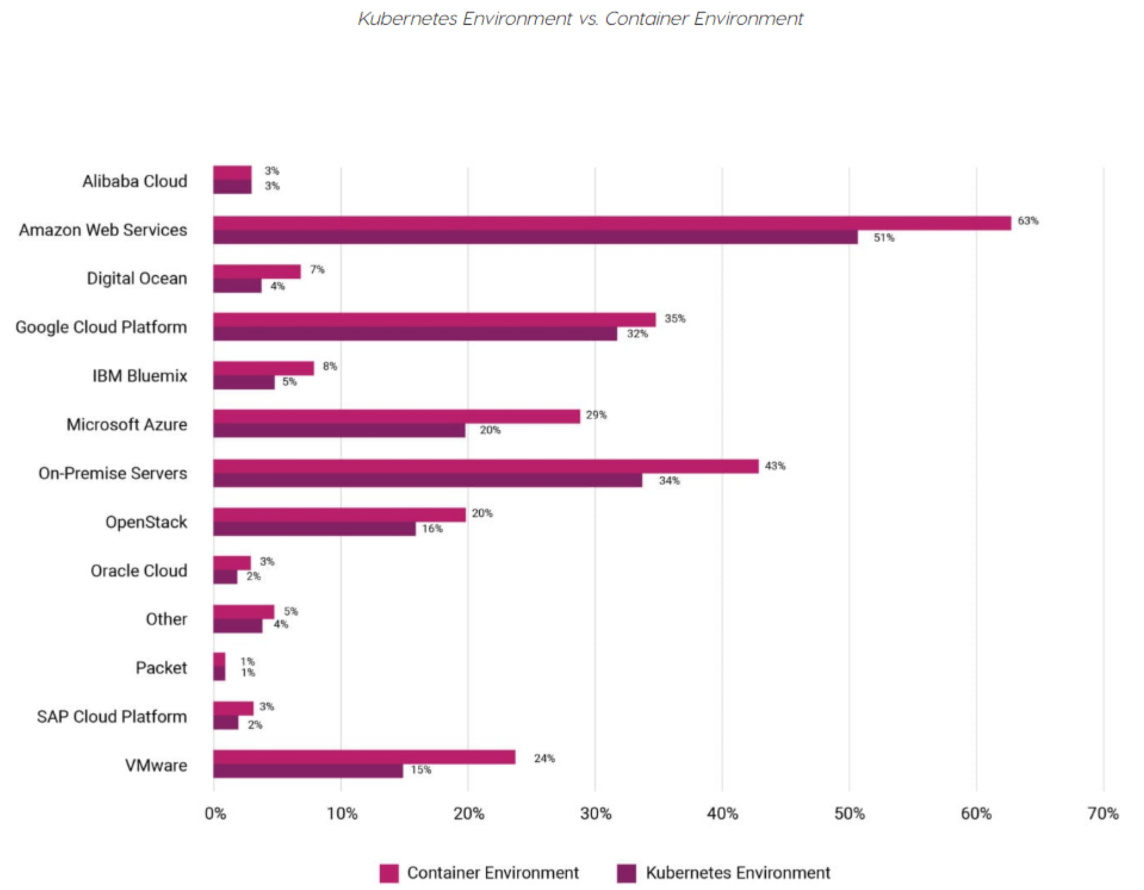
83% Of enterprise workloads will be in The Cloud by 2020

Sources :

www.cncf.io
www.forbes.com

Commercialization - Market and Stats





Q – Your Company manage containers with ?

Q – Kubernetes Environment vs Container Environment

Commercialization – Why K8ADVISOR ??

- Optimized deployments considering microservice dependency.
- Integrated dashboard, with a user-friendly, interactive interface.
- Proactively auto scale deployments.
- An easy tool to obtain holistic perspective on Kubernetes deployments.
- Privacy and security will be protected with in-house data storage.
- Machine learning with organizational specific data.
- Easy integration into existing Kubernetes deployments.

Commercialization – Business Plan

	BASIC FREE	MONTHLY $ 20	YEARLY $ 200
Dependency Visualization	✓	✓	✓
Load Prediction	✓	✓	✓
Centrality Analysis	✓	✓	✓
Optimization Suggestions	✓	✓	✓
Resiliency Evaluation	✗	✓	✓
Automatic Deployment	✗	✓	✓