A NETWORK SCIENCE BASED APPROACH FOR OPTIMAL MICROSERVICE GOVERNANCE

2020-021

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

Supervisor
Dr. Dharshana Kasthurirathna

Introduction

- Why Micro-Services ?
- Current trends and practices in Micro-Services
- Why Kubernetes ?
- Problem definition

Problems

Constant monitoring of metrics through APM tools. E.g.-Resource utilization process[1-2]

Difficult to understand why a problem occurs even though there is knowledge that a problem has occurred[5]

Have to make use of multiple monitoring sources to make effective decisions[3-5]

In short the research problem that our research aims to fulfil can simply be described as follows

"In 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. Furthermore, developers are not able to optimize their deployment such that they can make the optimal use of their deployed microservices in the cluster."

Research Questions

Can current orchestration tools successfully monitor relationships among microservices ?

How effective is the proposed optimization model compared to current optimization models ?

What is the impact of pod load prediction in determining an optimal deployment strategy ?

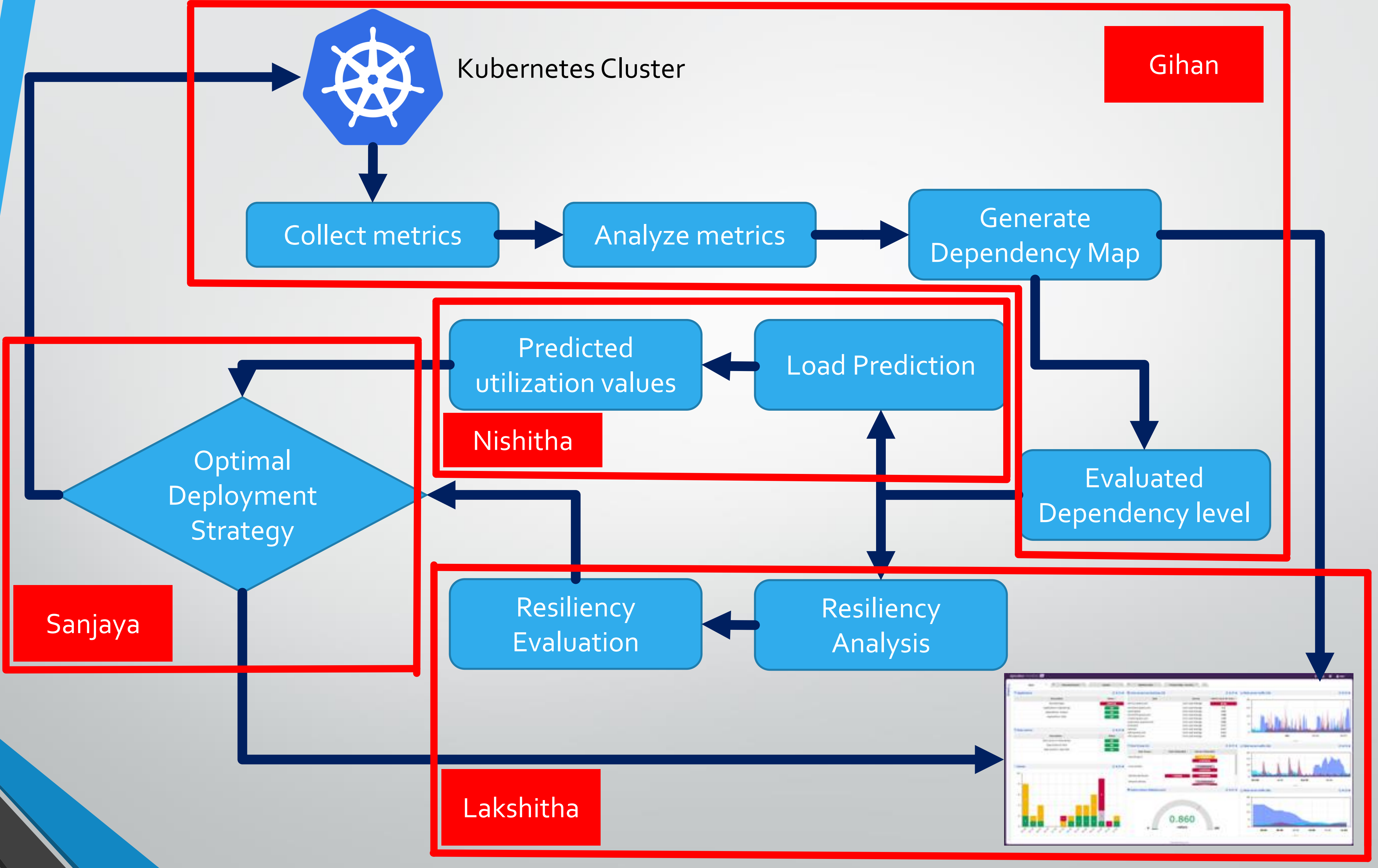
What is the importance and the impact of identifying the resiliency of microservices in deployments ?

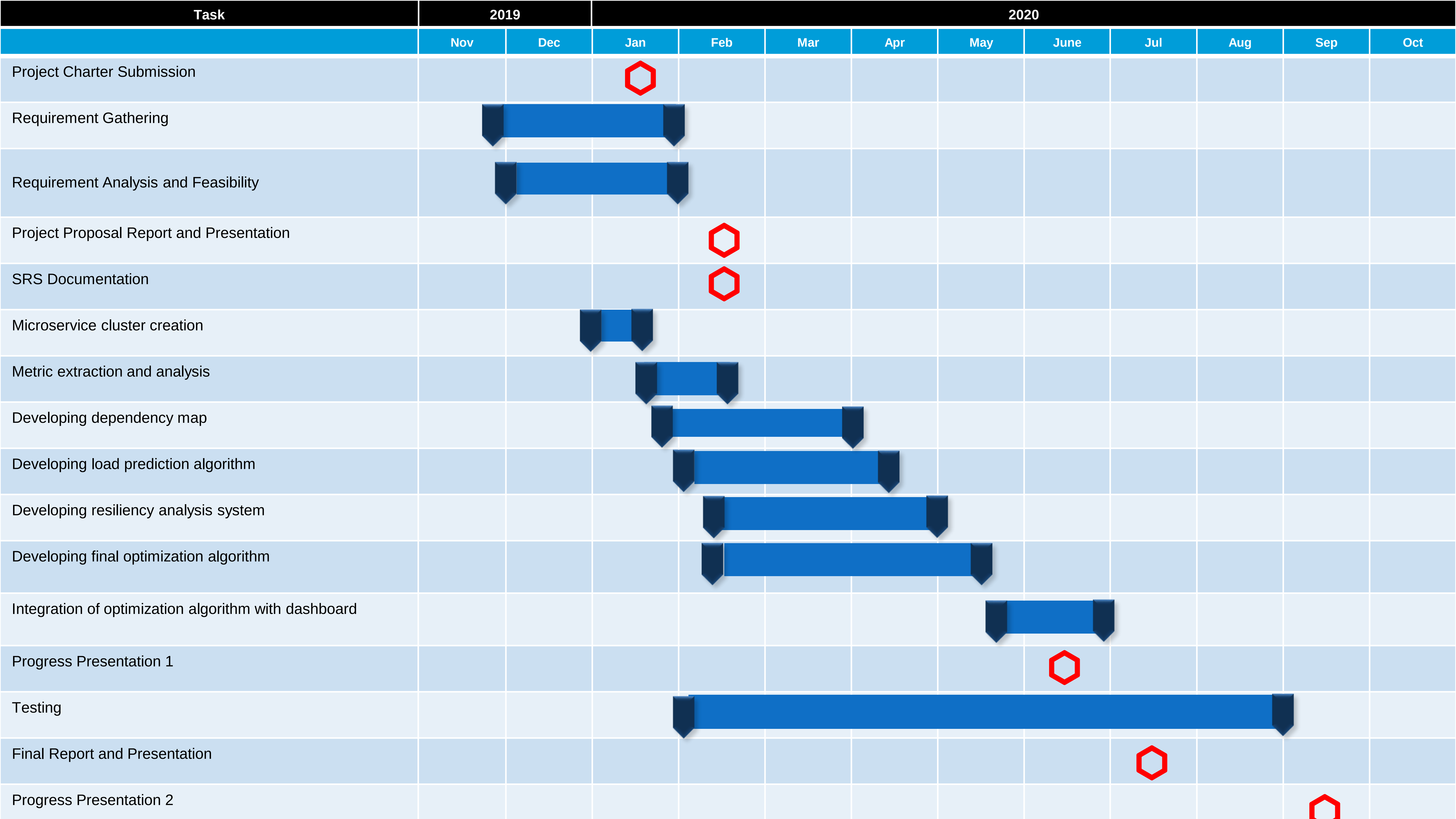
Main Objective

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.

Sub Objectives

- To increase the efficiency of microservices deployments by applying the metrics used in network analysis, such as centrality and resilience measures and link predictions on identified dependency measurements.
- To develop an improved auto-scaling policy for a deployment, based on load prediction
- To development of a business intelligence dashboard to evaluate performance and monitor microservice deployments.
- To identify key factors which lead to performance reduction in microservice deployments and come up with an optimal deployment strategy.


Kubernetes Cluster
Gihan
Collect metrics
Analyze metrics
Generate Dependency Map
Predicted utilization values
Load Prediction
Nishitha
Optimal Deployment Strategy
Evaluated Dependency level
Sanjaya
Resiliency Evaluation
Resiliency Analysis
Lakshitha


Task
2019
2020
Nov
Dec
Jan
Feb
Mar
Apr
May
June
Jul
Aug
Sep
Oct
Project Charter Submission
Requirement Gathering
Requirement Analysis and Feasibility
Project Proposal Report and Presentation
SRS Documentation
Microservice cluster creation
Metric extraction and analysis
Developing dependency map
Developing load prediction algorithm
Developing resiliency analysis system
Developing final optimization algorithm
Integration of optimization algorithm with dashboard
Progress Presentation 1
Testing
Final Report and Presentation
Progress Presentation 2

Requirements

Functional Requirements

- User should be able to view optimal deployment strategy for a given cluster
- User should be able to view possible deployment suggestions in order to optimize the deployment
- System should analyze the resiliency of the identified microservices in the cluster
- Load prediction should be performed on the identified microservices
- Load predictions should be made based for a maximum of 24hrs ahead
- Auto scaler should be configured to proactively auto scale based on predicted load
- Optimal deployment strategy for a particular deployment should be determined using load prediction, dependency measurements and resiliency evaluation

Requirements

Non Functional Requirements

- Usability
- Reliability
- Availability
- Interoperability
- Performance

Expected Outcomes

- To create a weight-based dependency network using metric analysis which highlights dependencies between the interconnected microservices.
- To create an improved auto-scaling policy integrating load-prediction analysis.
- To improve the performance of microservice deployments through resiliency analysis.
- To increase the performance in microservice deployments through the combination of the above-mentioned approaches.

Commercialization

Initial plan is develop model into an open source tool.

In the future, with the addition of newer features, we plan on making this tool a freemium tool.

A network science based approach for the generation of dependency map and a service mesh based on the network traffic and user behavior on a Kubernetes cluster

Saranga S.A.G
IT17016230

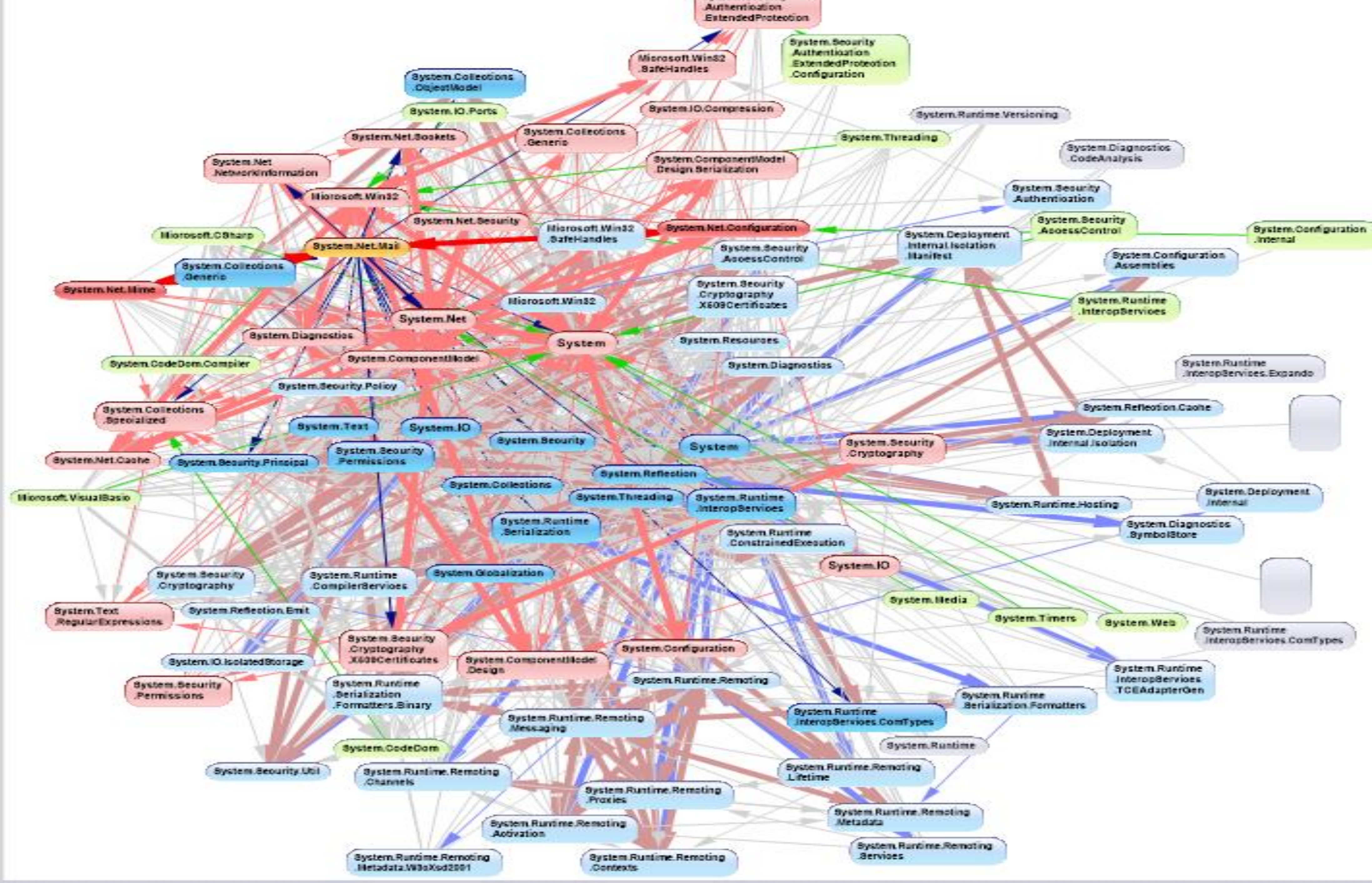
Introduction

- How Kubernetes works ?
- What is a service mesh ?
- Service Gateway
- Problem definition

Grafana
enterprise

JAEGER

kiali



Authentication .ExtendedProtection
System.Security .Authentication .ExtendedProtection .Configuration
Microsoft.Win32 .SafeHandles
System.Collections .ObjectModel
System.IO.Ports
System.IO.Compression
System.Runtime.Versioning
System.Net.Sockets
System.Collections .Generic
System.Threading
System.Diagnostics .CodeAnalysis
System.Net .NetworkInformation
System.ComponentModel .Design.Serialization
System.Security .Authentication
Microsoft.Win32
System.Net.Security
Microsoft.Win32 .SafeHandles
System.Net.Configuration
System.Security .AccessControl
System.Configuration .Internal
Microsoft.CSharp
System.Net.Mail
System.Security .AccessControl
System.Deployment .Internal.Isolation .Manifest
System.Collections .Generic
System.Configuration .Assemblies
System.Net.Mime
Microsoft.Win32
System.Security .Cryptography .X509Certificates
System.Runtime .InteropServices
System.Net
System.Diagnostics
System
System.Resources
System.CodeDom.Compiler
System.ComponentModel
System.Diagnostics
System.Runtime .InteropServices.Expando
System.Security.Policy
System.Collections .Specialized
System.Reflection.Cache
System.Text
System.IO
System.Security
System
System.Deployment .Internal.Isolation
System.Security .Cryptography
System.Net.Cache
System.Security.Principal
System.Security .Permissions
System.Reflection
Microsoft.VisualBasic
System.Collections
System.Threading
System.Runtime .InteropServices
System.Runtime.Hosting
System.Deployment .Internal
System.Runtime .Serialization
System.Diagnostics .SymbolStore
System.Runtime .ConstrainedExecution
System.IO
System.Security .Cryptography
System.Runtime .CompilerServices
System.Globalization
System.Media
System.Timers
System.Web
System.Runtime .InteropServices.ComTypes
System.Text .RegularExpressions
System.Reflection.Emit
System.Security .Cryptography .X509Certificates
System.Configuration
System.IO.IsolatedStorage
System.ComponentModel .Design
System.Runtime.Remoting
System.Runtime .InteropServices .TCEAdapterGen
System.Security .Permissions
System.Runtime .Serialization .Formatters.Binary
System.Runtime .Serialization.Formatters
System.Runtime.Remoting .Messaging
System.Runtime .InteropServices.ComTypes
System.Runtime
System.CodeDom
System.Security.Util
System.Runtime.Remoting .Channels
System.Runtime.Remoting .Lifetime
System.Runtime.Remoting .Proxies
System.Runtime.Remoting .Metadata
System.Runtime.Remoting .Activation
System.Runtime.Remoting .Services
System.Runtime.Remoting .Metadata.W3cXsd2001
System.Runtime.Remoting .Contexts

Related Research and Research Gap

- [1]Istio: Modernize digital applications with microservices management using the istio service mesh
- [4 Service Dependency Based Dynamic Load Balancing Algorithm for Container Clusters
- Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned

A data science based approach for an improved auto-scaling policy in Kubernetes based on load prediction

De Silva N.
IT17006880

Introduction

- What is autoscaling ?
- Current Autoscaling tools by Kubernetes
- What is Kubernetes HPA and how it works ?
- Problems in the current implementation of HPA
- Why we focused about issues in HPA ?

Problems faced in the current implementation of HPA

I. Response delay caused by the time required for Pod initialization[1,7]

II. Response delay causes increases the user's request response time, resulting drop in service quality [1,7]

III. Ineffective scaling caused due to the under allocation of resources in the configuration process[1,7]

Related Research

- [1] Uses a combination of ARIMA model and Empirical Mode Decomposition (EMD) in order to predict resource usage
  - Highlights current issues such as the response delay in Kubernetes
  - However, the use of EMD model has some drawbacks explained in [2]
- [3] Proposes a resource prediction algorithm called CRUPA
  - The CRUPA algorithm is based on a time series analysis model (ARIMA) combined with docker container techniques.
  - However this approach does integrate with existing auto-scaling tools provided by Kubernetes as well as does not make use of machine learning tools in order to perform the prediction process.
- [4-6] compares various models for short term load forecasting and their prediction accuracy

Objective

- To develop an improved auto-scaling policy for a deployment, based on load prediction

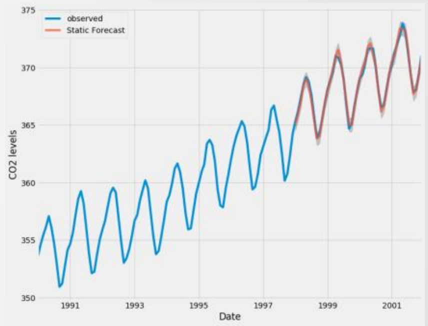
Research Questions

- What is the impact of load prediction in coming up with an optimal deployment strategy ?
- How accurately can we predict utilization values ?
- What is the impact of load prediction on the identified microservices in the overall performance of the cluster ?

Steps in the development of load prediction algorithm

1. Obtain the list microservices that show high level of dependency based on the proposed dependency map
2. Identify the relevant pods in which the identified microservices are deployed
3. Obtain the CPU utilization metrics of the deployed pods
4. Convert the metrics obtained into a time series

Steps in the development of load prediction algorithm cont.



1. Plot time series to identify the relevant time series components
2. Convert time series to a stationary dataset
3. Develop algorithm using prediction model to forecast future CPU utilization values for given pods
4. Forecasted CPU utilization values will be forwarded to optimization algorithm and used to configure auto-scaling of the cluster

Expected Outcome

To create an improved auto-scaling policy integrating load-prediction analysis which solves the current issues found in the HPA in Kubernetes

A network science based approach for evaluation of resiliency among microservices through the use of Chaos Engineering

Lakshitha M.V
IT17410250

Introduction

- What is resilience?
- What are the ways to analyze resilience?
- What is Chaos Engineering?
- What are the tools used in Chaos Engineering?
- Importance of analyzing resilience

Currently the dependency network is not considered when measuring the importance of a service in the deployment

Related Work

- [1]Gremlin: Systematic Resilience Testing of Microservices
  - how to use Gremlin to express common failure scenarios and how developers of an enterprise application were able to discover previously unknown bugs in their failure-handling code without modifying the application.
- [4]Chaos Monkey: Increasing SDN Reliability through Systematic Network Destruction

Objectives

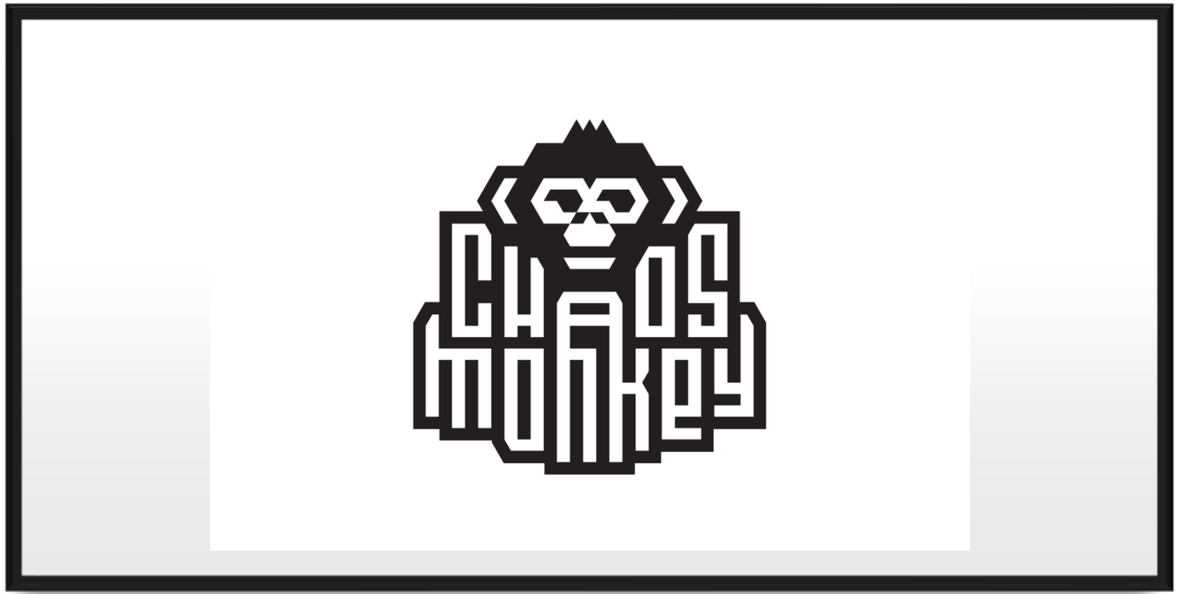
- Evaluate the resilience of microservices using Chaos Engineering to provide data to generate an optimal deployment strategy

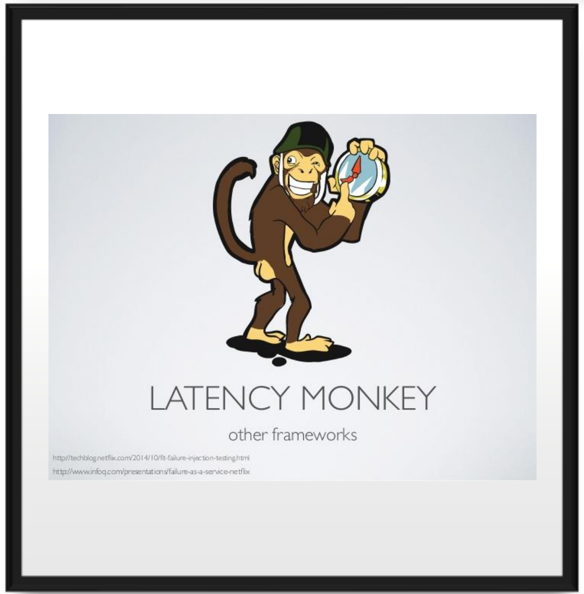
Research questions

- What are the impact of microservices on each other?
- How does the dependency network affect on the deployment of microservices?
- How to evaluate resiliency of microservices without interrupting customers?

Chaos Engineering

- Chaos Monkey - randomly kills a microservice
- Chaos Gorilla - kills entire availability zone
- Chaos Kong - kill whole reigon
- Latency Monkey - Introduces communication delays to simulate degradation or outages in a network.
- Doctor Monkey - Performs health checks, by monitoring performance metrics such as CPU load to detect unhealthy instances
- Failure Injection Testing - designed to give developers a "blast radius" rather than unmanaged chaos


CHAOS
monkey


LATENCY MONKEY
other frameworks
http://techblog.netflix.com/2014/10/fit-failure-injection-testing.html
http://www.infoq.com/presentations/failure-as-a-service-netflix

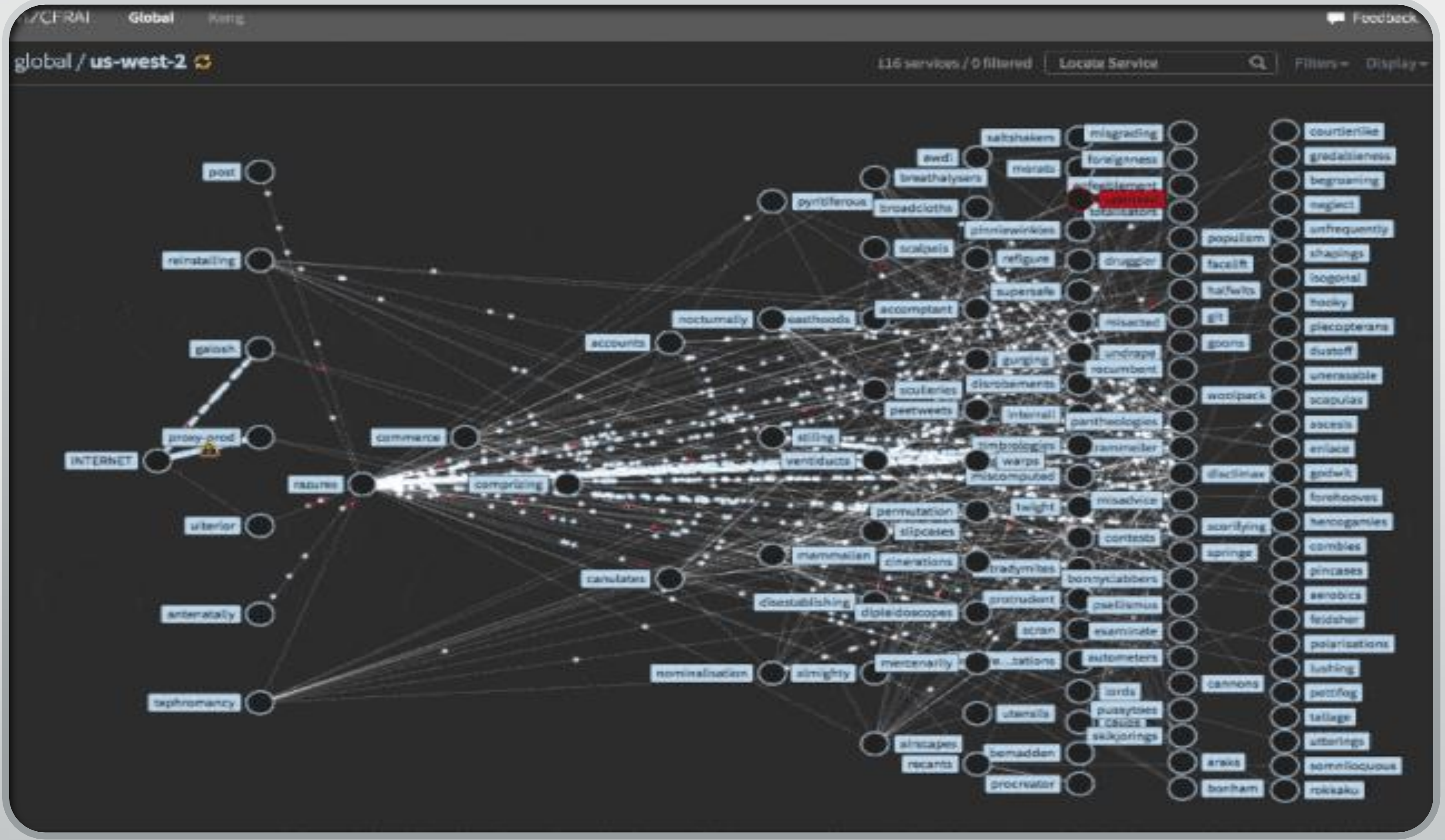

Chaos Gorilla
NETFLIX
@atseitlin


CHAOS KONG
because regions fail
http://techblog.netflix.com/2015/09/chaos-engineering-upgraded.html

Steps in designing a Chaos Experiment

1. Pick a hypothesis
2. Choose the scope of the experiment
3. Identify the metrics you're going to watch
4. Notify the organization
5. Run the experiment
6. Analyze the results
7. Increase the scope
8. Automate

Stimulating failovers to determine the highly dependent nodes



[7]How Netflix does failovers in 7 minutes flat

Expected Outcome

- To generate a detailed evaluation report on the resiliency of the microservices

A network science based approach for generating optimal deployment strategy by using dependency, load predictions and resiliency measures among microservices in Kubernetes.

L.S Jayasinghe
IT17016230

Availability and Network latency

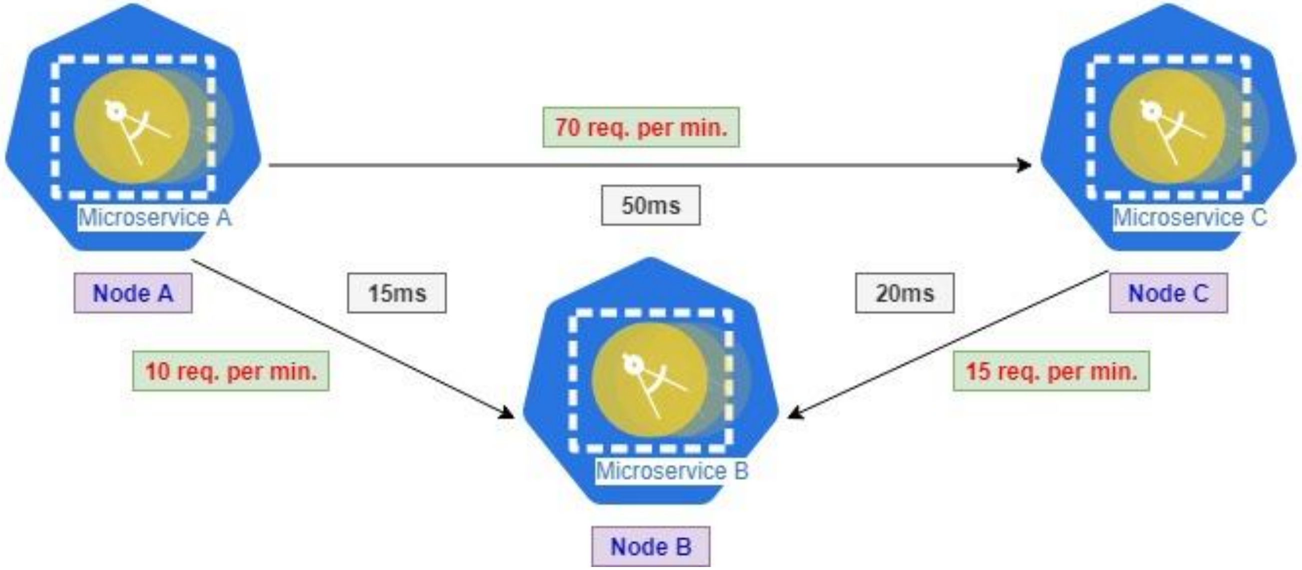
- When we deploy microservices, we deploy them in many nodes. because we need to increase availability.

- In the Kubernetes cluster, we have master node and salve nodes, these nodes can be located in the multiple places, may be another country.

- This problem causes delays between nodes. This is called network latency[2,4].

- **That is affected to overall response time[1]**

- E.g.: if network latency is high. Overall response time will be high.



Network Latency between different regions (Azure Cloud)

- Lowest latency in December: Australia central1 region – Australia central2 region(1ms)[5]
- Highest latency in December: Australia central-South Africa(392ms)[5]

	Australia Central	Australia Central2	Australia East	AustraliaSouthEast	Brazil South	Canada Central	Canada East	Central India	Central US	East Asia	East US	East US2	France Central	FranceSouth	Germany North	Germany West Central	Japan East	Japan West	Korea Central	Korea South	North Central US	North Europe	Norway East	Norway West	South Central US	South East Asia	South India	Southafrica West	SouthAfricaNorth	Switzerland North	Switzerland W
ralia Central		1	6	18	312	204	214	148	184	121	205	200	290	304	302	296	131	136	150	146	192	276	312	304	176	94	126	392	388	302	30
ralia Central2	1		6	18	312	206	214	146	186	120	206	202	291	304	302	296	130	134	148	146	192	278	312	306	176	92	124	392	388	302	30
ralia East	6	6		12	306	200	210	142	182	116	200	196	286	300	296	292	126	130	144	142	188	272	308	300	170	88	120	388	382	297	30
raliaSouthEast	18	18	12		318	210	220	140	192	118	210	206	296	310	308	302	132	136	146	140	198	282	318	310	182	86	118	386	382	308	31
il South	312	312	306	318		132	140	302	144	322	118	114	186	200	196	192	260	268	301	306	138	172	208	200	138	326	316	328	342	198	20
ada Central	204	206	200	210	132		12	208	20	198	25	30	92	106	104	98	152	160	178	182	15	80	114	108	44	216	222	238	248	104	10
ada East	214	214	210	220	140	12		218	30	208	35	38	102	116	114	108	162	170	186	192	24	88	124	116	52	226	232	246	256	114	11
tral India	148	146	142	140	302	208	218		222	88	196	198	116	104	128	120	122	122	116	110	216	132	146	140	224	54	24	264	262	114	11
tral US	184	186	182	192	144	20	30	222		176	27	34	106	112	118	112	132	140	157	162	12	94	128	120	22	196	228	246	260	120	11
Asia	121	120	116	118	322	198	208	88	176		200	208	184	172	196	188	50	50	52	56	184	198	214	206	182	34	66	334	328	182	17
US	205	206	200	210	118	25	35	196	27	200		6	80	90	90	86	168	168	181	186	18	66	102	94	32	220	209	220	234	92	8
US2	200	202	196	206	114	30	38	198	34	208	6		80	90	94	90	164	164	176	182	24	70	106	98	28	226	212	222	238	93	8
ce Central	290	291	286	296	186	92	102	116	106	184	80	80		12	20	10	218	216	212	206	98	16	32	24	106	151	129	152	166	14	1
ceSouth	304	304	300	310	200	106	116	104	112	172	90	90	12		26	17	204	202	200	194	106	26	44	36	116	138	116	164	158	10	8
many North	302	302	296	308	196	104	114	128	118	196	90	94	20	26		10	229	228	224	218	110	26	20	26	121	162	142	164	178	16	1
many West Central	296	296	292	302	192	98	108	120	112	188	86	90	10	17	10		220	218	216	210	106	20	28	20	116	154	132	160	174	6	1
n East	131	130	126	132	260	152	162	122	132	50	168	164	218	204	229	220		8	28	34	139	232	247	240	138	68	100	374	362	214	21
n West	136	134	130	136	268	160	170	122	140	50	168	164	216	202	228	218	8		36	42	147	232	246	238	138	76	100	366	360	214	21
ea Central	150	148	144	146	301	178	186	116	157	52	181	176	212	200	224	216	28	36		6	164	228	242	234	150	62	94	362	358	210	20
ea South	146	146	142	140	306	182	192	110	162	56	186	182	206	194	218	210	34	42	6		170	221	236	228	156	56	88	356	352	204	20
th Central US	192	192	188	198	138	15	24	216	12	184	18	24	98	106	110	106	139	147	164	170		84	122	114	34	202	228	244	256	110	10
th Europe	276	278	272	282	172	80	88	132	94	198	66	70	16	26	26	20	232	232	228	221	84		36	30	96	166	145	150	172	26	2
way East	312	312	308	318	208	114	124	146	128	214	102	106	32	44	20	28	247	246	242	236	122	36		8	132	180	158	174	186	32	3
way West	304	306	300	310	200	108	116	140	120	206	94	98	24	36	26	20	240	238	234	228	114	30	8		124	174	152	168	180	26	3
h Central US	176	176	170	182	138	44	52	224	22	182	32	28	106	116	121	116	138	138	150	156	34	96	132	124		200	222	252	264	120	11
h East Asia	94	92	88	86	326	216	226	54	196	34	220	226	151	138	162	154	68	76	62	56	202	166	180	174	200		34	300	296	148	14
h India	126	124	120	118	316	222	232	24	228	66	209	212	129	116	142	132	100	100	94	88	228	145	158	152	222	34		279	274	126	12
hafrica West	392	392	388	386	328	238	246	264	246	334	220	222	152	164	164	160	374	366	362	356	244	150	174	168	252	300	279		18	164	16
hAfricaNorth	388	388	382	382	342	248	256	262	260	328	234	238	166	158	178	174	362	360	358	352	256	172	186	180	264	296	274	18		170	16
zerland North	302	302	297	308	198	104	114	114	120	182	92	93	14	10	16	6	214	214	210	204	110	26	32	26	120	148	126	164	170		4
zerland West	306	306	300	312	200	108	118	110	114	178	88	88	10	8	19	10	211	210	206	200	106	24	36	30	116	144	124	160	166	4	
Central	186	186	182	178	294	202	210	30	216	126	188	190	108	96	120	112	161	160	156	150	209	124	138	130	218	94	48	258	254	106	10
North	186	186	180	178	300	206	214	29	220	126	192	194	112	100	124	116	160	160	154	148	212	130	142	135	220	94	48	262	258	110	10
outh	285	286	280	290	180	86	96	124	100	191	74	78	8	17	20	14	224	224	220	212	92	10	29	22	104	158	136	148	162	20	1
Vest	288	288	282	294	184	90	98	126	104	194	76	80	8	18	22	18	228	226	222	215	96	12	30	24	106	162	140	150	164	22	1
t Central US	170	172	166	178	159	36	44	236	14	162	47	48	120	132	132	126	118	126	142	148	26	108	142	136	22	182	214	264	276	134	13
t Europe	292	292	288	297	188	94	104	126	110	194	82	85	10	20	10	8	226	225	222	216	101	16	22	15	112	160	138	151	170	12	1
t India	146	144	140	138	302	208	216	4	222	86	194	196	116	102	126	118	120	120	114	108	214	130	144	138	222	52	22	265	260	114	11
t US	142	144	140	150	170	64	72	222	42	149	64	59	138	148	160	154	104	106	129	134	54	128	170	164	34	168	200	292	304	150	14
t US2	162	164	158	170	180	56	66	214	36	142	66	71	142	152	154	148	96	104	122	126	46	130	164	156	44	160	192	284	297	156	15

How to avoid from high network latency

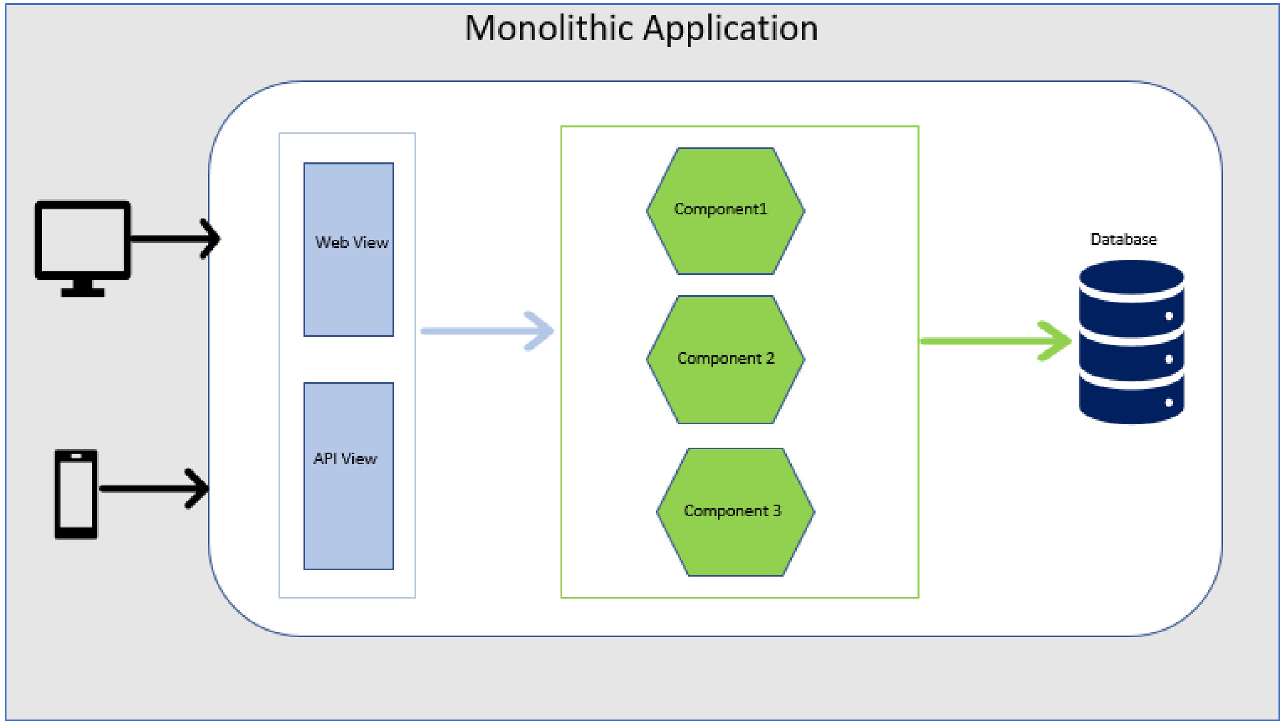
All the Microservices can be deployed in one Big Node.

Micro-services with higher communication rate can be deployed in nearby nodes.

One Big Node

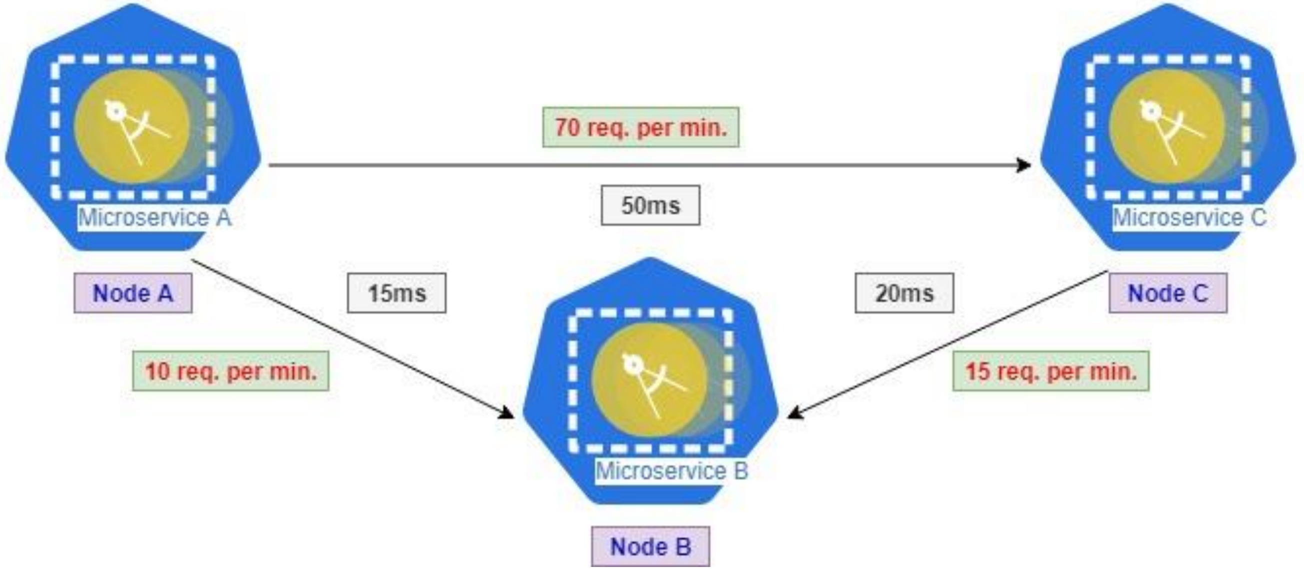
- If all the microservices are deployed in one big Node, we can decrease network latency.

- But then the application will become monolithic. Therefore availability can be reduced and it is one of the main features of Micro-Service architecture



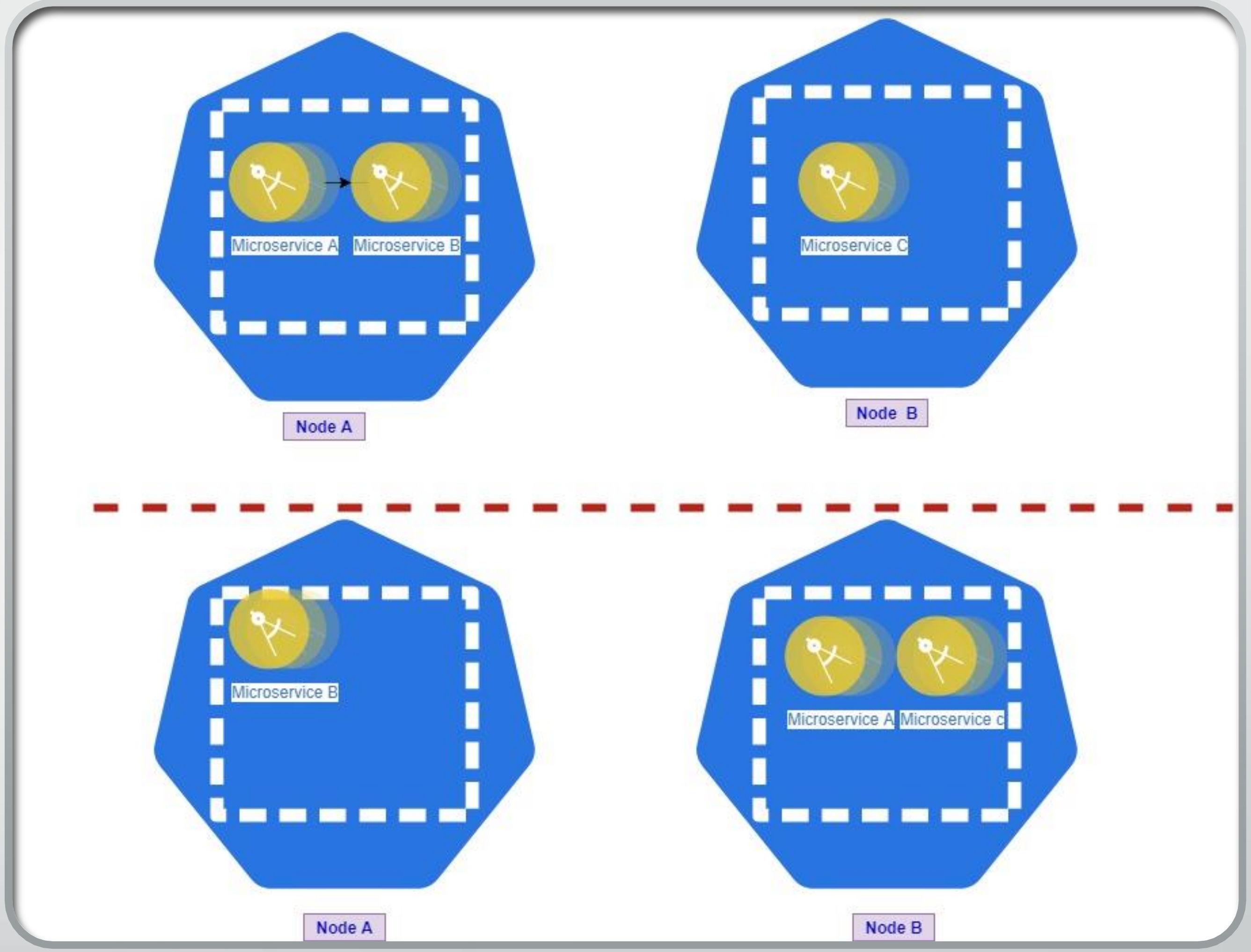
Decrease Overall network latency (scenario 1)

- If the couple of nodes are located in very far, the communication delay time is very low, also if couple of node are located near, communication is very fast.
- Best for multi-region problem



Scenario 2

- Microservice A and C communicate more than microservice A and B.



Related work

There is one research found on optimal deployment strategy[8], it is only concerned about ,

- resource power
- replication
- dependency map

Our research,

- Load prediction
- Resiliency

Objective

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

Research Questions

HOW SAME REGION NETWORK LATENCY AFFECT ON OVERALL PERFORMANCE ?

HOW DOES ALL THE METRICS AFFECT ON THE OPTIMAL DEPLOYMENT ALGORITHM ?

HOW DOES THE NODE NETWORK COMMUNICATE WITH THE POD NETWORK ?

HOW DOES DIFFERENT SERVICES AFFECT ON ALGORITHM?

Proposed Solution

To solved these previous scenario we develop an optimal deployment algorithm. It ensures the system has optimal availability and low network latency. It controls replication of the microservices and increases overall system performance.

To make this optimization algorithm we should be concerned about,

- Resource power
- Dependency map
- Future loads
- resiliency
- Network latency
- Node resource power
- Replication of microservices

Methodology

GET THE METRIC VALUES FROM DEPENDENCY MAP, LOAD PREDICTIONS, RESILIENCY

CREATE A NODE MAP(NETWORK LATENCY MAP:SOCKPERF (LINUX))

ANALYSIS THE INFORMATION USING OPTIMAL DEPLOYMENT ALGORITHM[6]

SHOW NEXT 24 HOURS DEPLOYMENT MAP AND FUTURE SUGGESTION IN THE UI

Path for the creation of optimal deployment strategy…

- Multi object optimization algorithm
- Identify shortest path between node(Dijkstra algorithm)[7]
- Rank all the metrics
- Identify correct deployment place using the algorithm

Expected Outcomes

- To generate an optimal deployment strategy for a given cluster based on load prediction, dependency analysis and resilience evaluation.
- Next 24 hours deployment map

Automated Deployment

YAML

Discussion and Conclusion

- Tools and frameworks such as Kubernetes provide developer a way in which to deploy and run their microservice applications easily with minimal configuration.
- However, currently, there are no such ways for a developer to evaluate the effectiveness of a particular deployment and identify potential performance issues.
- Furthermore, since identification of these issues require extensive analysis of metrics, it can be difficult for developer to evaluate if a particular microservice meets the expected performance criteria.
- Therefore, a solution should be developed which performs the above-mentioned metric analysis and come up with a model which identifies the potential performance issues such that developers can get the optimal use of their deployed microservices.

Questions ?

Thank you!

References - main

- [1]"DataDog/the-monitor", GitHub, 2020. [Online]. Available: https://github.com/DataDog/the-monitor/blob/master/kubernetes/monitoring-kubernetes-performance-metrics.md
- [2]"Resource Usage Monitoring in Kubernetes", Kubernetes.io, 2020. [Online]. Available: https://kubernetes.io/blog/2015/05/resource-usage-monitoring-kubernetes/
- [3]"Monitoring Kubernetes (Part 1): The Challenges and Data Sources - DZone Performance", dzone.com, 2020. [Online]. Available: https://dzone.com/articles/monitoring-kubernetes-part-1-the-challenges-and-da.
- [4]"Challenges of Monitoring and Troubleshooting in Kubernetes Environments | Sumo Logic", Sumo Logic, 2020. [Online]. Available: https://www.sumologic.com/blog/troubleshooting-kubernetes/.
- [5]D. Snyder, "How to Overcome Kubernetes Monitoring Challenges", OverOps Blog, 2020. [Online]. Available: https://blog.overops.com/how-to-overcome-monitoring-challenges-with-kubernetes/.
- [6]M. Vizard, "Running Kubernetes at Scale Top 2020 Challenge - Container Journal", Container Journal, 2020. [Online]. Available: https://containerjournal.com/topics/container-management/running-kubernetes-at-scale-top-2020-challenge.

References - main

- [7]"Kubernetes: The Challenge of Deploying & Maintaining", Techolution, 2020. [Online]. Available: https://techolution.com/kubernetes-challenges/.
- [8]"Kubernetes Challenges for Enterprises", Network Computing, 2020. [Online]. Available: https://www.networkcomputing.com/data-centers/kubernetes-challenges-enterprises.
- [9]"Monitoring Kubernetes (Part 1): The Challenges and Data Sources - DZone Performance", dzone.com, 2020. [Online]. Available: https://dzone.com/articles/monitoring-kubernetes-part-1-the-challenges-and-da.
- [10]Heinrich, Robert & van Hoorn, André & Knoche, Holger & Li, Fei & Lwakatare, Lucy Ellen & Pahl, Claus & Schulte, Stefan & Wettinger, Johannes. (2017). Performance Engineering for Microservices: Research Challenges and Directions.
- [11]Fazio, Maria & Celesti, Antonio & Ranjan, R. & Liu, Chang & Chen, Lydia & Villari, Massimo. (2016). Open Issues in Scheduling Microservices in the Cloud. IEEE Cloud Computing.

References-Gihan

- [1] J. Zhang, R. Ren, C. Huang, X. Fei, W. Qun and H. Cai, "Service Dependency Based Dynamic Load Balancing Algorithm for Container Clusters," *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, Xi'an, 2018, pp. 70-77. doi: 10.1109/ICEBE.2018.00021 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8592632&isnumber=8592611.
- [2] M. Gawel and K. Zielinski, "Analysis and Evaluation of Kubernetes Based NFV Management and Orchestration," *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, Milan, Italy, 2019, pp. 511-513. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8814580&isnumber=8814488
- [3] B. Carter, "Growing Software Applications by Incremental Development of Heterogeneous Micro-Applications Using Cellular Regeneration Concepts," *2015 Annual Global Online Conference on Information and Computer Technology (GOCICT)*, Louisville, KY, 2015, pp. 1-5. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7545087&isnumber=7545077
- [4] Preeth E N, F. J. P. Mulerickal, B. Paul and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," *2015 International Conference on Control Communication & Computing India (ICCC)*, Trivandrum, 2015, pp. 697-700. L: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7432984&isnumber=7432856.

References-Nishitha

- [1] Zhao, A., Huang, Q., Huang, Y., Zou, L., Chen, Z., & Song, J. "*Research on Resource Prediction Model Based on Kubernetes Container Auto-scaling Technology." IOP Conference Series: Materials Science and Engineering,2019.*
- [2]Yamin Wang, L. Wu and Shouxiang Wang, "Challenges in applying the empirical mode decomposition based hybrid algorithm for forecasting renewable wind/solar in practical cases," *2016 IEEE Power and Energy Society General Meeting (PESGM)*, Boston, MA, 2016, pp. 1-5
- [3]Y. Meng, R. Rao, X. Zhang and P. Hong, "CRUPA: A container resource utilization prediction algorithm for auto-scaling based on time series analysis," *2016 International Conference on Progress in Informatics and Computing (PIC)*, Shanghai, 2016, pp. 468-472
- [4] Nguyen, H., & Hansen, C. K. "*Short-term electricity load forecasting with Time Series Analysis." 2017 IEEE International Conference on Prognostics and Health Management (ICPHM).*
- [5] Bozkurt, Ö. Ö., Biricik, G., & Tayşi, Z. C. "*Artificial neural network and SARIMA based models for power load forecasting in Turkish electricity market".* 2017.
- [6] Vagropoulos, S. I., Chouliaras, G. I., Kardakos, E. G., Simoglou, C. K., & Bakirtzis, A. G. " *Comparison of SARIMAX, SARIMA, modified SARIMA and ANN-based models for short-term PV generation forecasting." 2016 IEEE International Energy Conference (ENERGYCON).*
- [7]Kim, Won-Yong & Lee, Jin-Seop & Huh, Eui-Nam. (2017). Study on proactive auto scaling for instance through the prediction of network traffic on the container environment.

References-Lakshitha

- [1]"Systematic Resilience Testing of Microservices", 2016 IEEE 36th International Conference on Distributed Computing Systems. Available: https://ieeexplore.ieee.org/abstract/document/7536505[Accessed: 27- Jan- 2020]
- [2]"Chaos Engineering" Basiri, A., Behnam, N., de Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., & Rosenthal, C. (2016). . IEEE Software, 33(3), 35-Available: https://ieeexplore.ieee.org/abstract/document/7436642[Accessed: 28- Jan- 2020]
- [3]"Optimal and Automated Deployment for Microservices",FASE 2019: Fundamental Approaches to Software Engineering pp 351-368 Available:https://link.springer.com/chapter/10.1007/978-3-030-16722-6_21
- [4]"Chaos Monkey: Increasing SDN Reliability through Systematic Network Destruction" Available : https://dl.acm.org/doi/pdf/10.1145/2785956.2790038 [Accessed: 25- Jan- 2020]
- [5]"Improving Kubernetes Resiliency with Chaos Engineering" Available:https://medium.com/faun/failures-are-inevitable-even-a-strongest-platform-with-concrete-operations-infrastructure-can-7d0c01643oc6 [Accessed: 20-Jan-2020]
- [6]"Chaos Engineering by Casey Rosenthal, Lorin Hochstein, Aaron Blohowiak, Nora Jones, and Ali Basiri Available: http://channyblog.s3-ap-northeast-2.amazonaws.com/data/channy/2018/01/18023151/chaos-engineering [Accessed: 23-Jan-2020]
- [7]"How Netflix does failovers in 7 minutes flat" Available:https://opensource.com/article/18/4/how-netflix-does-failovers-7-minutes-flat [Accessed: 23-Jan-2020]

References - Sanjaya

- [1]"Analysis of HTTP Performance Problems", W3.org, 2020. [Online]. Available: https://www.w3.org/Protocols/HTTP-NG/http-prob. [Accessed: 27- Jan- 2020].
- [2]"What is cloud service latency? - Definition from WhatIs.com", WhatIs.com, 2020. [Online]. Available: https://whatis.techtarget.com/definition/cloud-service-latency. [Accessed: 27- Jan- 2020]
- [3]"What Is Latency and How to Reduce It - KeyCDN Support", KeyCDN, 2020. [Online]. Available: https://www.keycdn.com/support/what-is-latency. [Accessed: 27- Jan- 2020].
- [4]"Test Azure virtual machine network latency in an Azure virtual network", Docs.microsoft.com, 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-test-latency. [Accessed: 27- Jan- 2020].
- [5]2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/networking/azure-network-latency https://searchcloudcomputing.techtarget.com/feature/Reducing-network-latency-means-focusing-on-location-location-location. [Accessed: 27- Jan- 2020].
- [6]"Multi-objective optimization", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Multi-objective_optimization. [Accessed: 27- Jan- 2020].
- [7]"Dijkstra's algorithm", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm. [Accessed: 27- Jan- 2020].
- [8]M. Bravetti, S. Giallorenzo, J. Mauro, I. Talevi and G. Zavattaro, "Optimal and Automated Deployment for Microservices", Fundamental Approaches to Software Engineering, pp. 351-368, 2019. Available: 10.1007/978-3-030-16722-6_21 [Accessed 27 January 2020].