Research Project Exhibition 2023

M.Sc in I.T. Architecture

M.Sc in DevOps

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Technology Ireland ICT Skillnet

FOREWORD

Today's Research Project Exhibition is a day presenting great achievement by our taught Master's students in our Software Architecture and Computing with DevOps Programmes. The students now presenting their research projects started their Masters programme journeys in January 2021. While it seems like only yesterday, yet so much has happened since then. Obviously these students started under COVID lockdown conditions and today, as they finish the programme, will be one of their first changes to have an in-person meet.

In other changes, those graduating today, started as students of the Department of Computing which was then part of the School of Science and Computing. You will graduate later this year as some of the first graduates to complete a programme in the new School of Enterprise Computing and Digital Transformation. This new school is part of a new Faculty of Computing, Digital and Data and represents a major investment by TU Dublin which sees Computing, Data Science and Digital Transformation as major activity areas for now and the future

As a School of Enterprise Computing and Digital Transformation we are using this opportunity to say that we are very proud to have you as graduates. Your dissertation work, which is at the cutting edge of Technology and Computing as it is practice in industry, represents some of the best work produced by students in this School. As a new School we are, engaged on an extensive consultation process around our mission, vision and activities and we welcome any comments, feedback and input on how our activities as they are anow and as they should or could be.

These programmes are being taught here in TU Dublin but they are collaborative programmes where we have joined with industry in defining and scoping these programmes. Industry needs in these programmes were refined and proposed by Technology Ireland Skillnet and their ICT employer partners identified the market need for these programmes, TU Dublin, Computing responded, co-designing an industry list of requirements into an academic Masters programme.

For the Software Architecture programme we are proud to partner with the International Association of Software Architects (IASA) and their Irish partners the Irish Computer Society (ICS) in offering this programme. It should be noted that this programme is recognised by the IASA for membership of the association.

Today we have a new cohort starting their journey as well as a cohort moving to the final part of their journey. They have plenty to learn but they too will soon reach this point. Creating and passing on knowledge is the duty of a University and doing this in fast moving Technological Fields relevant to industry is the particular mission of a Technological University.

We hope you enjoy the project symposium.

Finbarr Feeney PhD / Head of School
School of Enterprise Computing and Digital Transhformation
OT Bhaile Átha Cliath / TU Dublin - Tallaght Campus
D24 FKT9
Ireland

MSc in I.T. Architecture
MSc in DevOps

RESEARCH PROJECT SYMPOSIUM AGENDA

20th January 2023

14:00 - 14:15 Opening Addresses
Gary Clynch, TU Dublin, Tallaght Campus

14:15 - 14:45 DevOps Keynote Address
Eric Strong, Senior DevOps engineer at Payroc

14:45 - 15:15 IT Archictecture Keynote Address
Rory Cawley, Global Operations Team Manager, Qualtrics

15:15 - 17:00 Poster Presentations

Technology Ireland ICT Skillnet

TU DUBLIN
OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

Online MSc in DevOps

With most technology organisations moving their delivery platforms to a DevOps approach the shortage of people with cross sectional skills in DevOps is now acute. Developed by industry as a direct response to this need this first-ever Master's degree in DevOps aims to fill these important talent gaps and give credit, recognition and credibility to technologists working in this field.

The advantages of Development Teams and Operations Teams collaborating to improve the delivery of technology solutions has meant a rapid adoption of DevOps approaches to the Software Development Lifecycle. Closely associated with Lean and Agile concepts in enhancing the delivery of technology solutions, the DevOps approach has impacted very rapidly on the Technology industry.

Most existing DevOps 'specialists' grow or develop into their role with no formal standards or certification, and a modicum of training in the actual practice of cross functional DevOps practices. They may already be experienced, highly skilled, competent and high performers in their own field of Software Development, Computing, IT Management, or Quality Assurance but they can lack the knowledge and understanding of the other cross functional disciplines they now find themselves working with daily. Understanding not only the technical, but also the business and human factors at play during the high pressure demands of modern software delivery processes, is essential in the modern discipline of DevOps.

Award Level

There are two phases to the award. Candidates are registered for the full Masters of Science in DevOps Level 9 degree (90 credits) however candidates may opt to exit the programme on successful completion of the first three semesters with 60 credits and receive a Level 9 Postgraduate Diploma in DevOps (60 credits). Please note exit awards are at the discretion of the college and no refund of fees will be due.
The award structure will place greater emphasis on continuous assessment, practical and project work rather than on formal examinations. In fact there are only 2 modules that carry an actual exam.
The aim is that participants will gain a deep understanding of the topics and content covered, and be able to demonstrate this acquired knowledge as proven competence in tests and exercises drawn from practical "real life" DevOps scenarios.

Programme Delivery

The programme will start with a 3 day workshop which will involve all participants being physically present. This is seen as important to facilitate networking, experience sharing and group learning.

It is expected that lectures will be delivered one evening per week in term time and every 3-4 weeks there may be a requirement to hold lectures twice in that week. There will also be a requirement to attend one on-campus day at the end of each semester.

Lectures will be streamed live from TU Dublin (Tallaght Campus) i and will be available for download and offline viewing.

Semester 1: Introduction to DevOps

Human and Organisational Issues	Software Development Methodologies
• Lean and Agile movements and methods • Assess and evaluate organisational design and culture to facilitate DevOps style development, deployment and support • Develop and manage global multi-disciplinary teams including an understanding of the cultural and practical issues which arise • Be able to form, lead and develop teams • Assess competence, accountability, responsibility, norms and operational management • Collaboration, negotiation and partnering • Managing the Future - Creating a readiness for organisational change, organisational development and change management	• Technical implications of DevOps – the philosophy, the history, the SDLC, Lean, Agile Manifesto, continuous feedback and learning • Change, Source, Defect Control Systems, Examination of major industry implementations (e.g. Atlassian, VSTS) • Code Promotion • Code Synchronization • System Debugging • Software QA • Automated Testing • Software Security Vulnerability Management • Software Telemetry and Monitoring • Feedback and Learning

Semester 2: DevOps Fundamentals

Business Technology Strategy	IT Infrastructure Fundamentals for DevOps
• The Business Case for Agility and DevOps • Lean/Agile management/methods/frameworks (SAFE) • Product road maps, pipelines, backlogs, valuing new features and technical debt • Business case development and risk assessment • Creation/management of multi-annual business plans • Financial Management of Product and Technology life-cycles • Project Management and Methodologies • The end of the monolithic project • Designing for agility and value • Challenges for DevOps • Regulated Software • Impact for Customers of DevOps approach	• Automation of Infrastructure • Task and Process automation languages • Advanced System Administration • Software Security • System Hardening • Policies and implementation • Virtualisation • Containerisation • IT Network and Infrastructure Protocols • IT Network Monitoring • Continuous Deployment • Cloud Computing Concepts • Infrastructure as Code

"Understanding not only the technical, but also the business and human factors at play during the high pressure demands of modern software delivery processes, is essential in the modern discipline of DevOps."

Semester 3: Advanced DevOps

Advanced IT Infrastructure for DevOps	DevOps in Practice
• Architectural Design to support DevOps • The DevOps supply-chain and PLM relationship • DevOps in the Public Cloud • Comparative Analysis of Cloud Offerings • Cloud Scalability and Elasticity3 • Load Balancing • Virtualisation Automation • Provisioning and Orchestration • Software Configuration Management • Software Provisioning Management • Security in the Public Cloud • Degrading systems gracefully • Chaos Monkey • Server-less Compute in the Cloud	• The DevOps paradigm/pipeline in practice requirements • Develop Continuous Integration/Test/Deployment Release management • Monitor and Learn • Feedback and Iteration • Detailed DevOps Case Study of the technical and human experiences of typical practitioners, e.g. - Google SRE (Site Reliability Engineering) - Intercom (Customer Messaging Platform)

Semester 4: DevOps Research

Research Methods	Research Project
• Academic Writing • Qualitative and Quantitative research • Surveys • Statistics	• Applied piece of Research in DevOps area • Encompasses a Proof of Concept/Prototype • Supplements DevOps Theory knowledge *This is an opportunity for students to carry out a piece of work which is at the cutting edge of the field and explores in depth a feature or element of that field. It is perfectly feasible, and there are many examples of this, for students to carry out their research project on a piece of work of direct relevance to their company or organisation. The academic team in TU Dublin (Tallaght Campus) have deep industry experience and have supervised and developed MSc. projects which explore business values, infrastructure automation and DevOps projects with real industry relevance.*

Technology Ireland ICT Skillnet

M. Sc. Applied IT Architecture (online)

In conjunction with Irish Computer Society and accredited by International Association of Software Architects. A Technology Ireland Skillnet funded programme. This programme is 80% online with two days per semester attendance required.

M. Sc. Computing with DevOps (online)

This programme was designed in conjunction with leading ICT companies such as Microsoft, Fidelity, IBM, Ericsson who form the Technology Ireland Skillnet. This programme is 80% online with two days per semester attendance required.

Non-Standard Applicants

Note for interested applicants: Next intakes for these Skillnet programmes set for January 2019. Standard admissions requirements include a relevant bachelor's degree at honours level. It is recognised that there are experienced and skilled potential participants for the programme who may not fit the standard entry profile. A non-standard admission process is available here which can be based on prior experiential learning and/or qualifier modules. These qualifier modules can be taken from September 2019 for admission in January 2019. Contact bfeeney@it-tallaght.ie or mhendrick@it-tallaght.ie for more information.

Accreditation of Master of Science in Applied IT Architecture by IASA

Technology Ireland ICT Skillnet

The TUDublin (Tallaght Campus) M.Sc. in Applied IT Architecture is the first of its kind in the world to be developed based on the IASA Five Pillars.

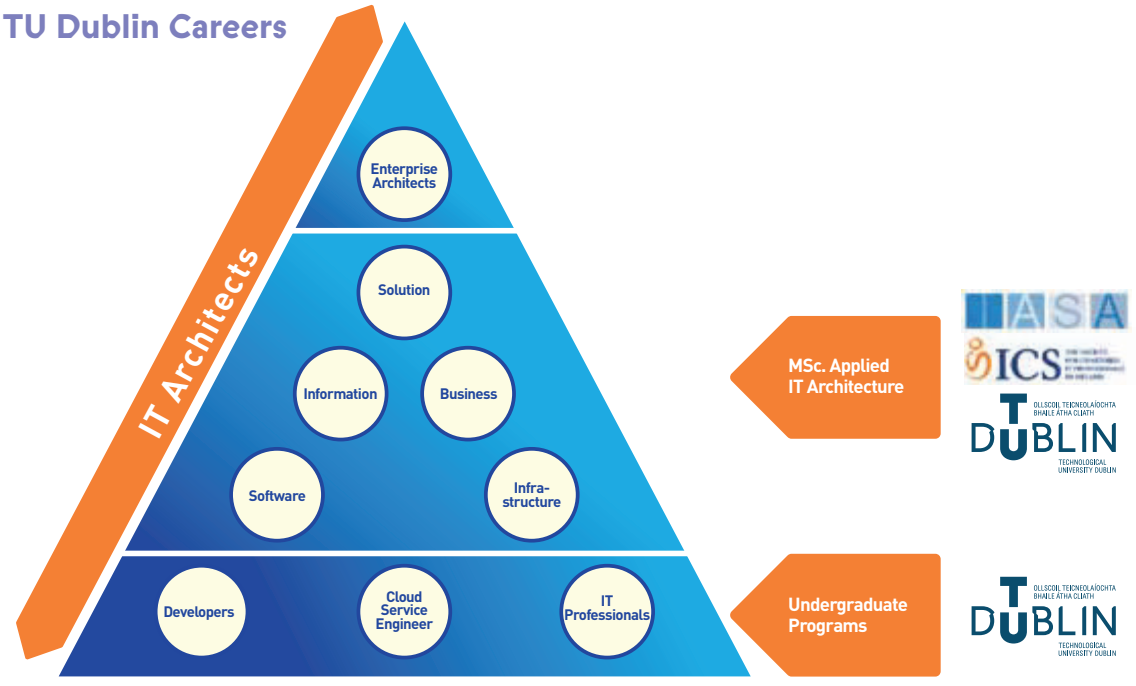
For the first time, candidates can gain a full Masters of Science degree in this specialist area through a mixed learning process with an emphasis on practical application in the workplace.

What is IT Architecture? (From IASA Global)

Architecture at IASA is the practice of business, organization or client gain through the application of technology strategy. It is the art and science of designing and delivering valuable technology strategy. At its core, the ITABoK describes how to create a professional person or group of professionals who can consistently find new applications of technology to generate positive outcomes for their client or employer. IT Architects:

- Retain depth in technical skill as well as business skill
- Able to successfully work with both business and technical staff
- Develop their own or others business cases based on technology driven innovation
- Retain the ability to deliver projects on those business cases
- Deliver business projects more successfully based on outcomes than others

TU Dublin Careers



PROJECTS

A study of VXLAN Peformance

Ndifor Augustine
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180718@myTUDublin.ie

Introduction

Virtual Extensible Local Area Network (VXLAN) is a new virtualization technology that is designed for use in large data centres to address the scaling challenges of traditional data centres. Given the correct hardware, VXLAN can easily scale to exceed the 4094 address limitation of the 802.1Q standard used in VLAN networks by up to 16 million addresses.
For ease of use, VXLAN works without the need to retrofit existing network infrastructures. To ensure interoperability, VXLAN has support from most major players like VMware, Intel, Broadcom, Arista, Open vSwitch, and others to ensure interoperability and no vendor lock-in.

VXLAN Architecture

Compared to a traditional VLAN 3-tier architecture, a VXLAN design is a 2-tier architecture comprised of a spine and leaf layer.

In this design, each leaf is directly connected to every spine, not another leaf. Traffic exchange between endpoints are encapsulated and decapsulated as VXLAN frames at the leaf layer using the virtual tunnel endpoint before being routed across the network through the spine layer.

BGP-EVPN Control-Plane

There are different ways to set up VXLAN, and Border Gateway Protocol Ethernet VPN (BGP-EVPN) is frequently used.

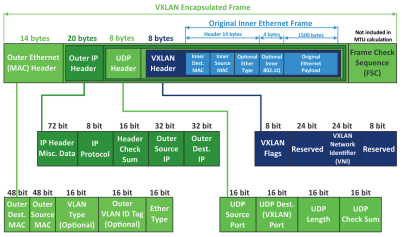
BGP is an open standard technology that lets autonomous systems on the network share routing information with each other. EVPN is a control plane that uses BGP to exchange information between VXLAN tunnel endpoints (VTEPs).

EVPN has helped to cut down on network flooding, solve problems with scalability, and make it easier to move virtual machines.

Major Components of VXLAN

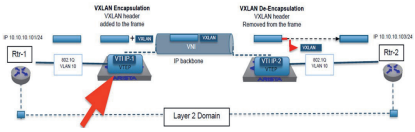
1. VXLAN Frame:

Unlike VLAN networks, VXLAN only transmits VXLAN frames across its network. To accomplish this, each incoming Ethernet frame is wrapped with an extra 50 bytes of VXLAN-specific data. Outer Ethernet, Outer IP, UDP Header, and a VXLAN Header, which consists of a 24-bit VXLAN Network Identifier (VNI) used to define the VXLAN address range, which provides up to 16 million addresses.



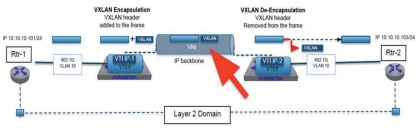
2. Virtual Tunnel Endpoint (VTEP):

This is the gateway to the VXLAN network, and it is used for encapsulation and decapsulation between the right VXLAN segments.

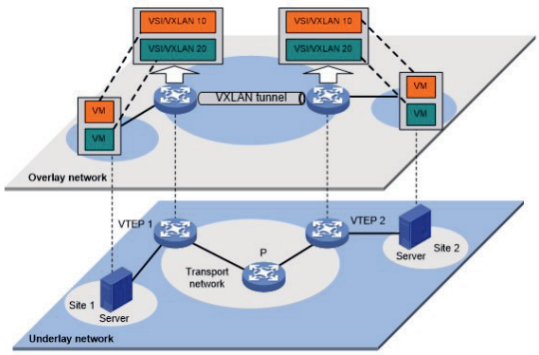


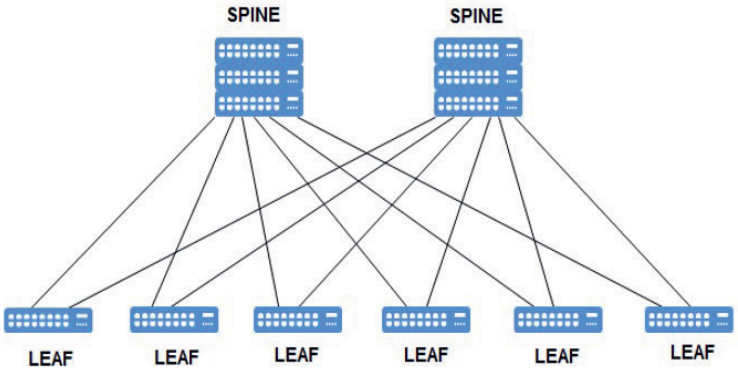
3. VXLAN Network Identifier (VNI):

This is used to identify specific Layer 2 Network Segment of an Ethernet frame that has been encapsulated and being routed across the VXLAN network.



VXLAN Design Layouts





Conclusions and Future Work

Today, using a VXLAN network has become very important for larger companies that need to grow beyond the limits of a traditional data centre, which still implements the 802.1Q standard.

While VXLAN can provide the much-needed addresses required for scaling modern network infrastructures, its effectiveness as a scaling solution may soon encounter a challenge of its own as the network continues to grow and expand. This is partly because it will become harder to manage manually. As such, there's a need for automation in deploying VXLAN networks to manage such complexity. This is where there is future work for next-generation programmable fabric solutions to automate VXLAN BGP EVPN deployment using an application programming interface (API).

QR Code for Recording

GKE Evaluation

David Foley
Department of Computing, TU Dublin, Tallaght, Ireland
x00180868@mytudublin.ie

Introduction

In recent years, Kubernetes has grown rapidly. As a result, 5.6 million engineers are said to be utilizing Kubernetes, with many big and medium-sized businesses adopting it.
Cloud service providers like Azure, Google Cloud, and Amazon Web Service are selling their flavor of Kubernetes, allowing small organizations to swiftly adopt the technology. With the growth and popularity of Kubernetes, Cloud providers such as Google Cloud are now offering Kubernetes managed service whereby the provider is responsible for the underlying Kubernetes control plane (Auto-healing, Autoscaling, Network and Node Configuration), allowing the developer to focus on the code.
Recently, VMware has released their own version of Kubernetes called Tanzu allowing on-premises enterprises to embrace Kubernetes eco-system. Researchers have been studying numerous facets of Kubernetes in recent years due to the popularity and expansion of the system.
The performance of Kubernetes within a Cloud Service Provider will be assessed in this research study. Load, endurance, and stress test scenarios are examined and evaluated in this paper. The evaluation's findings can be used to pinpoint use cases for a particular Kubernetes solution on the Google Cloud Platform (GCP).

Experiments

1. Load Testing:

Evaluate, the performance between a fully managed and a managed GKE cluster, by measuring how effectively it manages response times. Utilizing k6 which was developed by Grafana Labs, is an open source SaaS application load testing solution, offering users the ability to test their web applications, K6 provides consumers with a comprehensive breakdown of the number of requests issued and the responsiveness.

2. Endurance Testing:

Evaluate, the performance between a fully managed and a managed GKE cluster, under moderate load for extended periods of time. Utilizing Locust is a Python-based testing tool for load testing and simulating user behavior. Locust is a software application that generates a collection of testing routines that imitate a large number of consumers. This will define the critical performance, security, and application load management breaking point.

3. Stress Testing:

Evaluate, the performance between a fully managed and a managed GKE cluster, by exposing the clusters under intense stress. Utilizing Stress-ng will stress the cluster in a variety of methods. Stress-ng created to test several physical computer components as well as the numerous operating system kernel interfaces.

Results

1. Load
   In the context of the studies, no cluster scaling or pod scaling occurred during the trails on either cluster.
   During the k6 performance testing, Managed GKE Cluster has a shorter response time then a Fully Managed Google GKE Cluster.
   On average K6 performance testing had shown that a managed cluster was 17.98% quicker in response time.
2. Endurance
   During the endurance testing, the managed Kubernetes Cluster had an average failure rate of 91.93% compared to the fully managed cluster's average failure rate of 94.51%. This represents a 2.58% increase in failure rate compared to a managed Kubernetes Cluster.
3. Stress
   During the analysis, it was observed that the fully managed GKE cluster appeared to have no scaling issues or performance difficulties. However, performance testing appeared to demonstrate that the managed cluster had scalability difficulties with two containers, which failed to deploy properly.

Conclusions and Future Work

managed GKE cluster appeared to suffer in the stress test for scenarios involving CPU workloads of 80% and 90%. It could be argued based on the results of all three evaluating testing, that a fully managed Kubernetes cluster would be best suited for operations with high resource requirements and/or weekly/daily recurring jobs *("cronjobs")*, whereas managed Kubernetes Cluster might be utilized for Web or API applications because to the decreased latency for those with no mesh network requirements.
Future Work may involve the evaluation process for both Kubernetes clusters utilized a simple microservice application, the proposal for future performance evaluation, for load testing could make use of a consolidate database running within a microservice application, evaluate how both Kubernetes types perform for more complex microservice architecture. It may be useful to run the latency performance evaluation on a Managed Kubernetes cluster with a service mesh deployed, to get an indication of latency. At the time of written this research project, Google Cloud (GCP) has introduced a preview of Vertical Pod Scaling.

Considerations for selecting the correct automated testing tool for end-to-end web application testing

John Gunter
Department of Computing, TU Dublin, Tallaght, Ireland
X00180702@myTUDublin.ie

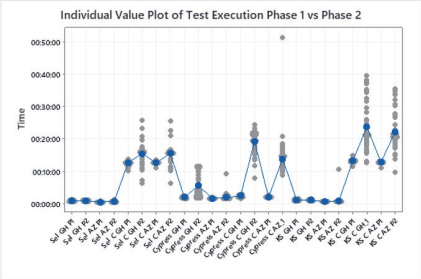
Introduction

With the vast majority of people interacting and using software daily, there is a pressure on organisations to delivery high quality software as fast and as frequently as possible. To help with this demand, organisations are incorporating agile and DevOps principals to allow for more frequent releases of their software. For this reason, testing has become a critical step in the software development lifecycle. This thesis investigates the considerations for selecting the correct automated testing tool. Three automated testing tools were selected, Selenium WebDriver, Katalon Studio and Cypress.io. Areas such as level of documentation, ease of installation and ease of use were noted before a testing framework and 64 automated tests were created. These tests were then integrated into a CI/CD pipeline and run across two applications, a monolithic application, and a containerised application. Variables such as pipeline execution time, test execution time and failure rate were collected, and results from each testing tool are compared against each other. Finally, Artificial Intelligence was integrated with the testing tools to verify if the use of AI added noticeable improvements to the testing tools.

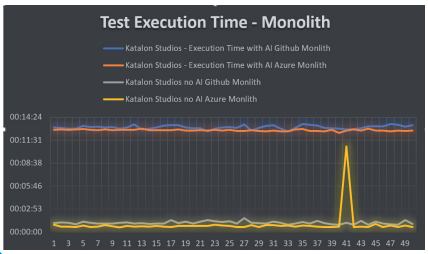
Comparison Table

	Documentation	Installation	Additional Software Requirements	Programming Skills	Framework Creation	Test Case Creation
Selenium WebDriver	3	3	2	3	3	2
Cypress.io	3	2	1	2	2	2
Katalon Studio	3	1	1	1	1	1

Test Results Overview



Execution Time AI vs no AI



Research Questions

RQ1. How do the tools compare when it comes to ease of use, documentation, implementation, and execution speed?

All three tools were installed on a virtual machine and test framework created. Once this was completed 64 test cases were created. Katalon Studio was the easiest tool to set up, using an installer that installed the application UI and all required packages and libraries. Cypress required some knowledge of the command line to be installed which made it a little more difficult. Selenium was the hardest with each component needing to be installed and configured. The documentation for all three testing tools was at a good standard.

RQ 2. What level of maintenance is required for each testing framework and how easy is it to integration the tools into a Continuous Integration / Continuous Delivery (CI/CD) pipeline?

Initial integration into a CI/CD pipeline for Katalon and Cypress is straightforward thanks to the GitHub actions and Azure plugin for Katalon, there is no need for further steps for reporting. However, this does come at a cost. The pipeline execution times were slower for both Cypress and Katalon with Cypress having a significant pipeline execution time. In contrast, Selenium is more difficult to set up, requiring command line commands, with additional requirements if reporting is to be added but it makes the execution faster. Framework maintenance for both Katalon Cypress is quite minimal as the products are managed by the organisation and there are updates are released frequently. An issue occurs if a release breaks a custom configuration that a user might have. The maintenance for Selenium is all a manual task and requires a user to ensure that all libraries and dependencies are kept updated and continued to work

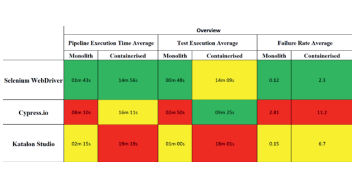
RQ3. How can incorporating Artificial Intelligence (AI) enhance the automated testing process, if at all?

While various tools were researched for the Selenium and Cypress, it was decided that using these tools would not offer any benefit to the research project as they required a large amount of manual intervention. Instead the self-healing AI and the Visual-AI tools for Katalon were used. The AI integration for Katalon was quite beneficial as it was able to catch the failures of the change application front-end code. However, this came at a cost to the speed of the test execution which would not be beneficial for large scale operations. The number of failures increased with AI integrated also and this was a surprising finding, as the assumption at the start of this project would be that AI would help decrease the number of failures

Topic Overview

For RQ1, the tools were compared using qualitative and quantitative methods of analysis. For the qualitative analysis each point of comparison was assigned a resulting score out of three. Katalon came out as the easiest tool to install and setup with very little prior knowledge or programming skill required. Selenium proved to me the most difficult and time consuming.

For the quantitative analysis, results from running the test suites were analysed any compared. As can be seem in the final overview of results (on the right) Selenium had the fastest pipeline and test execution time and the lowest amount of test failures. Cypress had the worst results overall because even with the containerised pipeline and test execution times being better, the failure rate was much higher, indicating the results are not as stable as the other testing tools.



The results from integrating with AI were interesting also. Overall, the three variables considered for this project: pipeline execution time, test execution time and failure rate were all considerable worse when AI was integrated compared to when it was not integrated.

Conclusions and Future Work

Overall Selenium was the tool with the best results. It was faster then the other two testing tools and had less failures across the different rounds of testing. Katalon Studio had some very nice, practical features, however it is worth nothing that these come at a cost. Cypress was nice and simple to work with however the test results overall were poor and would not inspire confidence if using the tool. The introduction of AI was interesting as overall it had a negative affect on results. AI however is around to stay, so future research would be interesting into this area and ways to improve it.

Azure Functions Comparison: Serverless vs. Kubernetes with KEDA

Cristian Liberona Riquelme
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180707@myTUDublin.ie

Introduction

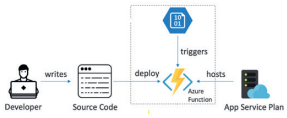
This research thesis aims to compare the performance and cost of Azure Functions when they are deployed using the Azure Functions serverless service, versus when they are deployed using Kubernetes and the Kubernetes-based Event-Driven Autoscaling (KEDA) component. The main problem is the lack of evidence about the performance and costs of Azure functions as a serverless solution provided by Microsoft versus the exact implementation on containers. The research question is: What is the difference in performance and cost between Azure functions Serverless and its deployment on Kubernetes using KEDA and in which scenarios it is suitable the use one over the other? The research hypothesis is that there will be a difference in performance and cost between Azure functions Serverless and its deployment on Kubernetes using KEDA, and that the suitable use of one over the other will depend on the specific requirements and priorities of the application.

Research Outline

The experiments for this research will include a backoff test, a concurrency test, and performance measures. The backoff test will analyze cold start times and expiration behaviors of Azure function instances and how KEDA emulates these behaviors. The concurrency test will measure the ability of the serverless and KEDA platforms to invoke a function at scale. The performance measures will evaluate the efficiency and effectiveness of Function B in consuming queue messages and identify potential performance bottlenecks. The costs of each implementation of Azure functions will also be compared.

Design & Infrastructure

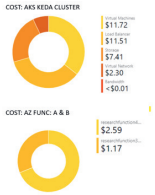
This research focuses on the infrastructure and platform for running Azure Functions, using Microsoft Azure and KEDA on a Kubernetes cluster. The functions implemented in C uses queue and time triggers, running on both an AKS cluster with KEDA and Azure Functions on Azure. They will be tested using predefined scenarios and performance measures in the West Europe region.



Research Result Analysis

1. Introduction:

In this analysis, the cost and performance of using KEDA on an AKS cluster versus Azure Functions on Azure are evaluated. The cost of each solution is affected by factors such as pricing models, resource allocation, deployment and management, and scale. In terms of performance, both Azure Functions and KEDA on an AKS cluster show good results in warm start scenarios, but Function B performs significantly slower on the AKS cluster with KEDA in a cold start scenario



2. Performance Analysis:

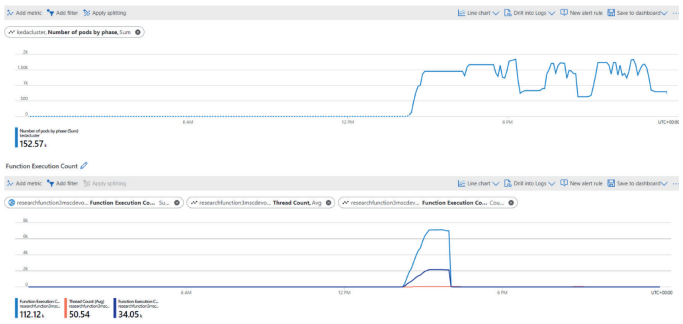
Function B was found to perform better on Azure than on an AKS cluster with KEDA in warm start scenarios, taking 48 minutes to consume 13,472 queue items on Azure and 246 minutes on the AKS cluster. In cold start scenarios, Function B took 109 minutes to consume 12,984 queue items on Azure and 680 minutes to consume 112,984 queue items on the AKS cluster, indicating poorer performance on the latter. Further investigation is needed to determine the causes of these differences in performance and to explore potential solutions and optimization strategies.

3. Cost Analysis:

The use of KEDA on an AKS cluster or Azure Functions on Azure can have different implications on cost. Azure Functions offers a pay-per-use pricing model and is a managed service, which can be beneficial for applications with variable workloads. However, KEDA on an AKS cluster may require the provisioning and allocation of additional resources and specialized expertise, which can increase cost. Additionally, Azure Functions offers automatic scaling, while KEDA on an AKS cluster may require manual scaling and management, which can also affect cost. Factors such as workload, scaling requirements, and deployment and management approach should be considered when determining the most cost-effective approach.

Total Cost & Overall Performance Overview

Scenario	Cost (USD)
AKS cluster implementation	32.94
AKS cluster + Function A (container)	2.88
AKS cluster + Function B (container)	6.11
AKS cluster (total)	8.99
Function A (Azure Function Service)	1.17
Function B (Azure Function Service)	2.59
Total (Azure Function Service)	3.76



Conclusions & Future Work

The main reason for using Kubernetes to run Azure Functions is for cost control, such as for consuming third-party APIs or handling seasonal demand. Other benefits of using Kubernetes include control over code and runtime, flexibility in deployment and hosting options, and consistent deployment model. However, the researchers recommend using native serverless Azure Functions due to their ease of use and good performance, even on the lowest paid tier, as well as the reduced development time and support required. Potential areas for future work include conducting a more in-depth analysis of the performance and cost implications of using KEDA on a Kubernetes cluster versus Azure Functions on Azure, exploring the benefits and drawbacks of using KEDA in different runtime environments and programming languages, and investigating the potential benefits and drawbacks of using KEDA in hybrid and multi-cloud environments.

QR Code for Recording

WebAssembly runtime performance in a resource constrained environment

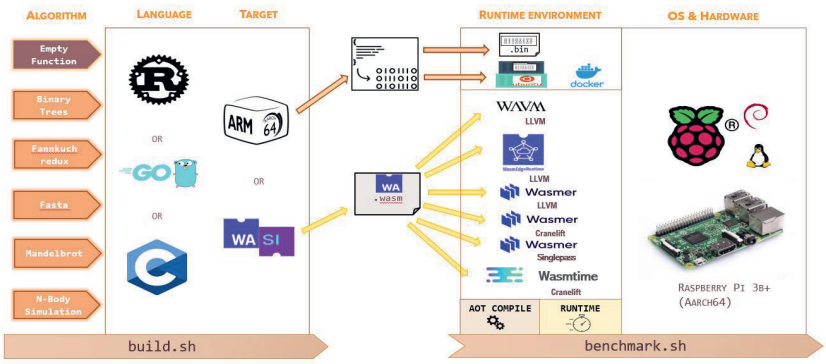
David Mahon
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180710@myTUDublin.ie

Introduction

Compute models such as serverless at the edge are gaining in popularity as a method of managing low latency bursty traffic requirements. However, given the hetrogeneous nature of devices on the edge, there are many underlying variables which can affect compute performance. WebAssembly is increasingly cited as a viable technology to provide near native performance in edge scenarios. Modern WebAssembly runtime projects such as WasmEdge claim to be lightweight, high-performing and extensible. However, studies to date tend to concentrate mainly on resource plentiful and x86 based cloud infrastructure, and less so on resourced constrained ARM scenarios which make up a significant portion of devices at the network edge. In this study an experimental project is run to examine the performance characteristics of WebAssembly runtimes in an example ARM based resource constrained environment. Using different combinations of WebAssembly runtimes, compilers, high level programming languages, and algorithms, this study performs an in-depth comparison of startup and runtime compute performance, comparing natively compiled algorithms binaries with cross-compilation to WASM bytecode and subsequently ahead-of-time complied into compatible machine code, for various WASM runtime environments.

Research Questions & Experimental Setup

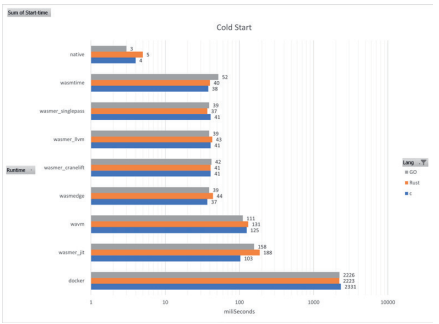
- **RQ-1:** What commercial or open-source webassembly runtimes are currently available that support ARM and aarch64 hardware and instruction sets?
- **RQ-2:** Given a resourced constrained IOT device using ARM hardware, comparatively how will a software function written in C, RUST or GO perform when either run as a native binary, or cross-compiled to WASM bytecode and Ahead of time (AOT) complied to native machine code running in a webassembly runtime environment
- **RQ-3:** ...how will a software function perform when either run as a native binary in a docker container, or cross-compiled to WASM bytecode.
- **RQ-4:** ...how will cold start times for an empty function perform when AOT complied to WASM code and run in a webassembly runtime environment, when compare with running natively in a docker container.



Raspberry Pi Model 3B

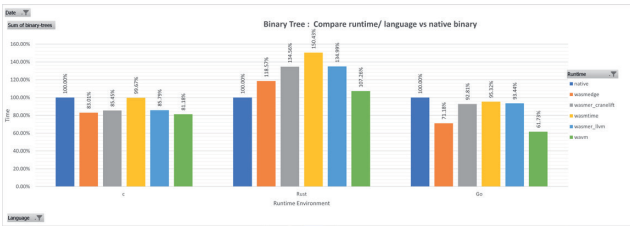
Device	Pi 3B plus Rev 1.3
Processor	1.4 GHz Quad-Core ARM Cortex-A53
FPU	VFPv4 + NEON
CPU	64 Bit
RAM	1GB
Swap	1024M
OS	Raspberry PI OS (64-bi) - lite
OS Version	R5.15.61-v8+ #1579
OS Implementation	Linux
Machine Hardware	aarch64

Cold Start Times



The data presented in logarithmic scale graph format shows that the ahead of time (AOT) compiled wasm bytecode produced comparable results in all WASM runtime environments with the exception being WAVM. It can be seen that it only takes the WASM bytecode version of these these algorithms approximately 2% of the time needed to start a native binary in docker.

Binary Tree : Advantages of AOT compiled WASM byte code



For Binary Tree algorithm the results presented suggest that for C and GO it is more performant to cross compile to WASM and AOT compile to machine code using a WASM runtime environment. WAVM and TinyGo produced the best result for this scenario taking only 61.7% of the time to run relative to running the same function natively. Long et al.(2020) observed this phenomena also when using WasmEdge and attribute this to *"AOT compilation ...allows the compiler to optimize specifically for the machine it is currently working on as opposed to making generic optimizations for the entire class of CPU architecture"*.

Conclusions and Future Work

This study finds that running AOT WASM in different runtimes in a resource constrained environment can in some cases produce better performance and faster startup times when compared to running natively in a docker container. Ultimately this study presents a strong case for WASM as a medium for FaaS in resource constrained environments on the edge, while also suggesting that further enhancement and optimization of runtimes, language compilation targets and features is needed before it can become prevalent. Examples of further work in this space includes a deeper analysis of SIMD / NEON and other comparable technology in WASM and ARM devices to fully assess the advantage they would bring in resource constrained environments. further investigation should also be conducted into new WASM and WASI feature specifications such as multi-threading to determine their impact on runtime performance. Similarly, further work could be carried out to measure other aspects of WASM runtimes in this environment, such as memory and file I/O analysis.

QR Code for Recording

Azure Function App Native vs. Azure Function App Container Investigation

Ahmed Salah Mohamed
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180732@myTUDublin.ie

Introduction

Serverless function container image support in AWS or Microsoft Azure is substantial as it shows the potential of controlling the application environment by the developers and bypassing the programming language support by the cloud providers. It also helps resolve the vendor lock-in problem for companies considering multi-cloud solutions. Moreover, this approach will be highly appreciated in many industries, such as Machine Learning, IoT, Telecommunication, Data Science, and many more, as it will easily solve the use cases of integrating with other cloud services. The cloud provider of choice for this research is Microsoft Azure, with Java as a programming language. An experiment has been conducted to address the differences between Azure Function App Native and Azure Function App Container in terms of cost and performance. At the same time, a scoring system has been utilized to state the differences in terms of tooling. The results show cost variance between the two configurations as well as a difference in performance based on the used pricing plans and the different load profiles. Also, the scoring system stated the differences between the two configurations using defined metrics.

Motivation

Using containers solves many problems related to portability and repository management. Moreover, working with serverless functions as small services to solve business problems has become popular nowadays. However, for a software company with many serverless functions deployed on the cloud, it will be challenging to maintain the repository, and versioning of the code changes as they grow. Therefore, it seems appealing to manage containers that can be deployed once and used everywhere and use them inside those serverless functions. Therefore, the motivation for this research is based on getting the best of both worlds.

Research Questions

Q1: *What are the differences between Azure Function App Native and Azure function App Container in terms of performance, cost, and tooling?*

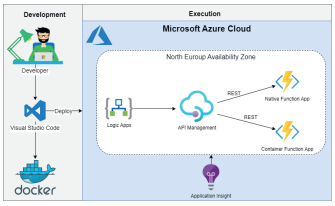
Q2: *Are there situations to favor one over the other?*

Q3: *Do synchronous and asynchronous approaches impact this comparison?*

Methodology

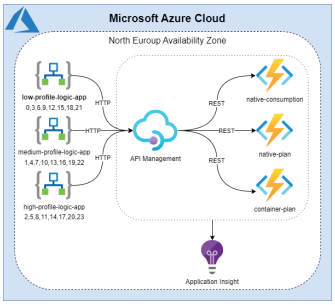
Objectives

Address the cost, performance, and tooling differences and understand when to use one over the other.



1. Experiment:

2 experiments for the different pricing plans "App Service Plan" & "Function Premium Plan".
3 Types of processing loads "Low, Medium, & High" on an hourly basis.
20 Runs with 10 seconds between each run per hour.
Same Algorithm programing language in all cases.



2. Scoring System:

A simple scoring system from 1 to 5, while 5 is the highest, to compare the personal experience of the tooling for the native and container configurations. The metrics to compare against are Ease of use, Portability, Maintainability, Manageability, Integration, and Vendor Lock-in.

3. Considerations:

The same code and load profiles, the same deployment method, the same availability zone & region, the same data inputs, exact executions count, timing, and different pricing plan instances with the same configuration.

Results

Metrics	Azure Function App Native			Azure Function App Container		
	High	Medium	Low	High	Medium	Low
ASP Duration	✓	✓	✓			
FPP Duration		✓	✓	✓		
ASP Response Time	✓	✓	✓			
FPP Response Time		✓		✓		✓
ASP CPU - Process				✓	✓	✓
FFP CPU - Process	✓	✓	✓			
ASP CPU - Resource	✓	✓	✓			
FPP CPU - Resource				✓	✓	✓
ASP Memory - Process				✓	✓	✓
FPP Memory - Process	✓	✓	✓			
ASP Memory - Resource				✓	✓	✓
FPP Memory - Resource				✓	✓	✓

Metrics	Azure Function App Native	Azure Function App Container
Ease of use	4	3
Portability	2	4
Maintainability	4	3
Manageability	2	4
Integration	4	4
Vendor Lock-in	4	4
Total	**20**	**22**

For the cost, *Azure Function App Native* has the edge as it can use the Consumption plan, which is less costly than the other two plans.

Conclusions and Future Work

When the cost is a concern, then the *Native* version is the smart choice. However, when codebase manageability and portability are the concern, the *Container* version is better. And when choosing the *Container* version, it is better to use the *Function Premium Plan* for better overall performance. In addition, there is no impact of using synchronous or asynchronous triggers as Microsoft uses a schematic implementation for the functions.

Microsoft Azure offers several options for running serverless containers, including Azure Container Instance, Azure Container App, and Azure Kubernetes Service. Comparing these options in terms of cost and performance and considering industry-specific use cases would be beneficial.

QR Code for Recording

Evaluation of Amazon Kinesis as a Real-Time Data Ingestion Tool

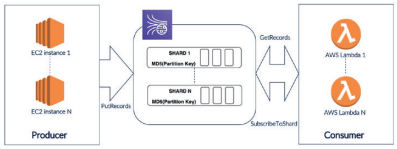
Cyriac Paulose
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180736@myTUDublin.ie

Introduction

Globally, data generation is increasing exponentially in the form of advertising, gaming, security monitoring, machine learning (ML), operating logs, social media feeds, website click-streams, financial transactions, IoT telemetry data, analytics, and other applications. Businesses need to access these data and turn them into meaningful analytics. The growing volume and rate of streaming data make it challenging for businesses to keep up with demand for near-real-time processing and analysis. This research shows why AWS Kinesis Data Streams (KDS) is a good contender for real-time data streaming. The research utilizes two sets of experiments spread across two main capacity modes - Provisioned & On-Demand - to assess Kinesis Data Stream's dependability, performance, and cost implications of its adoption.

Kinesis Data Stream

Kinesis Data Stream uses shards to store data which also determines the streaming capacity of Kinesis stream. The research makes use of producers hosted on AWS EC2 instances to simulate the data volume ingested into the Kinesis stream. One or more AWS Lambda consumers are used to consume the data.



Methodology

Experiment #1 (Provisoned Mode)			
Scenario	**# Shard**	**# Producer**	**# Consumer**
Performance under normal load	1	1	1
Performance under peak load	1	5	1
Improve performance under peak load with EFO	10	5	4

Experiment #2 (On-Demand Mode)			
Scenario	**# Shard**	**# Producer**	**# Consumer**
Performance under normal load	4	1	1
Performance under peak load	16	5	1
Improve performance under peak load with EFO	20	5	4

Research Questions

1. How reliable and performant is Amazon Kinesis Data Stream, as a cloud-native data ingestion solution?

Amazon Kinesis Data Stream does present a highly reliable and performant cloud based solution for real-time data ingestion requirements in terms of latency and throughput. The Kinesis shard provides write speeds of 1 MB/second and 1,000 records per second, and read speeds of 2 MB/second. The shard constraints guarantee consistent performance, which simplifies the development and maintenance of a highly dependable data streaming process.

The amount of data Kinesis data streams can consume dictates the number of shards to employ. Experiments using provisioned mode needed insight of data demand before setting up shard capacity. In On-Demand mode, shards are scaled up/down based on data load.

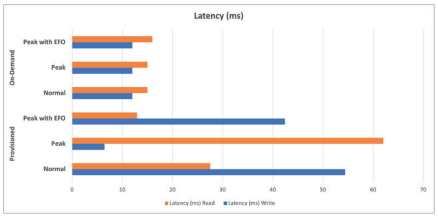
The testing also confirmed the value of EFO consumers. They would guarantee customers used the resource well, with maximum allocated throughput for each. Higher throughput and decreased read latency are worth the extra cost for each EFO consumer. Standard consumers are advised when there are few consumers (3 or fewer) and latency is not a concern. There are financial implications to using EFO, such as an additional hourly fee per EFO consumer and a cost for each gigabyte of data retrieved.

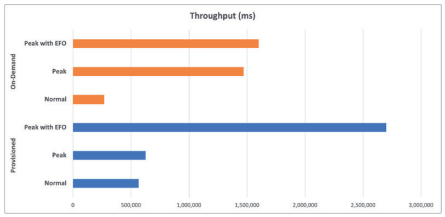
2. What are the cost implications for choosing a range of configurations that would meet performance targets?

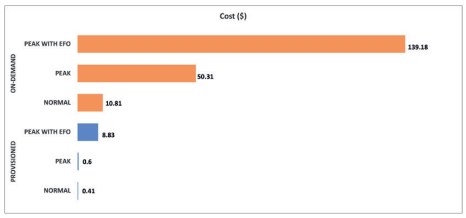
All results considered, the price of KDS is reasonable. Costs at AWS KDS are based on either the volume of data transported, or the number of shards provisioned, or data retention. There are only 2 capacity settings to worry about in this setup. When the incoming data streaming rate is predictable, provisioned capacity mode is an ideal option which would keep the cost lower. In situations where the rate at which data streaming in is unpredictable, on-demand capacity mode is the most appropriate choice. However this would drive the cost upwards in-turn providing a reliable and solid solution for businesses.

The improved performance with the provisioned mode test (Experiment 1), involving the use of an EFO consumer under peak load conditions, is $12.67/day when compared to a similar on-demand test (Experiment 2) which is $139.18/day. The main difference here is the scalability feature that on-demand mode offers, along with the much improved read throughput of the consumers, which is critical in handling data streams in real-time.

Testing Results







Conclusions and Future Work

The experiments in this research show that AWS Kinesis can operate effectively with GetRecords latency of 16 ms and PutRecords latency of 12 ms when the right number of shards are provisioned. Kinesis reached this level of performance by using dynamic shard provisioning in on-demand capacity mode. This feature allows Kinesis to scale up or down based on the amount of data being ingested. Although provisioned capacity mode has similar performance, it required additional computation to determine how many shards were needed to process the data. This is a more cost-effective option for businesses with steady data input and no traffic spikes. Future research in real-time data streaming using AWS Kinesis must be compared to AWS Managed Streaming for Apache Kafka (MSK) to identify the frameworks' ability to dynamically scale to provide guaranteed performance.

QR Code for Recording

Smart Contracts and Group Disability

Derek Potter
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180739@myTUDublin.ie

Introduction

Blockchain and smart contracts are seen as a disruptor in the insurance industry. Some studies have implemented some straightforward examples of smart contracts being used for claims processing interactions between a claimant, medical provider and insurance company.

This paper describes another use case where the policy is not an agreement between a patient and the insurance company, but the policy is between the insurer and an employer who is providing benefits for their employees. In this case, eligibility and liability calculations may require additional information such as whether pre-existing conditions are present; or if there is an offset associated with the condition which might reduce our liability.

This paper presents this more complex use case with these considerations in mind and suggests an architecture using Hyperledger Fabric. The challenges around implementing such a solution are discussed. Suitability has been determined based on selecting the system quality attributes that most contribute to the company's digital transformation goals.

Business Background

Group Disability insurance is where an employer purchases an insurance policy and their employees are covered by the policy.

There may be complicated rules determinations required to figure out if a potential claimant is eligible for the benefits they are claiming under any given policy.

There may even be manual overrides or considerations that need to be taken into account.

Smart Contracts could be used to execute these rules based determinations.

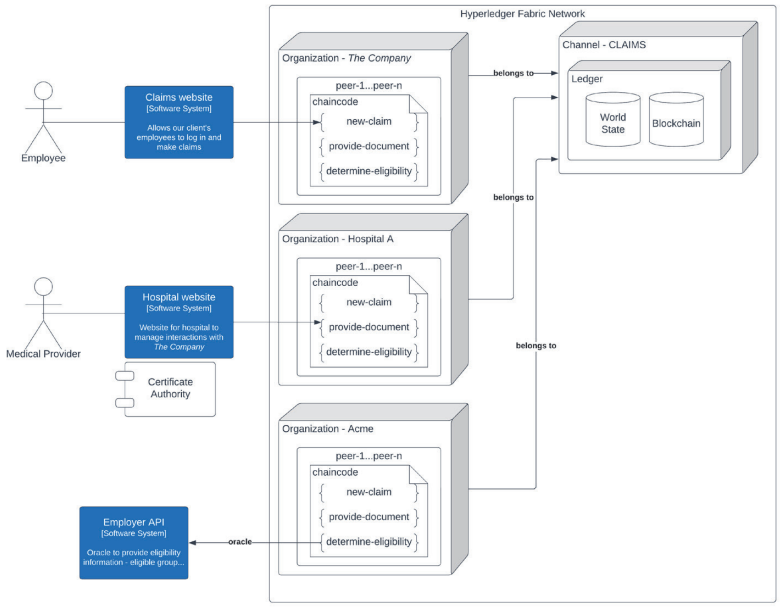
Smart Contract Frameworks

Ethereum and Hyperledger Fabric are popular blockchain frameworks which can execute Smart Contracts. They are compared below:

Ethereum	**Hyperledger Fabric**
Public permissionless	Private permissioned
No security	Certificate Authority
No privacy	PDCs
Solidity; Vyper	Java; Typescript etc...
Proof of Stake	Defined by policy
30–150	2,000–20,000
Business to consumer	Business to business

Potential Future-state High Level Architecture

We extracted possible use cases and came up with this potential end-state high level architecture.



Primary Research Findings

Channel granularity: At what level of granularity should we define our channels? A channel is a ledger and a group of organizations cooperating together submitting transactions to that ledger. It would be possible to create a straightforward smart contract that provides basic functionality with little customization. But what do we do with those employers that require additional customization.

Private Data Collections: Hyperledger Fabric abstracts identity management and validation into the concept of a membership services provider. A policy identifies the data an identity can access and the actions it can take. Eg: who can deploy chaincode and who can perform actions on a ledger. By default, any organization that belongs to a channel can create transactions and interact with the ledger within that channel. In cases where only some of the organizations should have access to certain data within the channel or its transactions, it is necessary to use a *private data collection*.

Fabric chaincode lifecycle: This governs the sequence of events that must occur in order for smart contracts to be deployed to the ledger. The code must be packaged into an archive, installed on each peer that will execute the contained smart contracts, approved by identities belonging to the organizations that own those peers and finally, it can be committed to the ledger.

Quality Attribute	Traditional Architectures	Hyperledger Fabric
Affordability	High	Low
Scalability	High	Medium
Transparency	Low	High
Immutability	Low	High
Auditability	Medium	High
Testability	High	Low
Compatibility	Medium	Low
Usability	High	Low
Supportability	High	Medium
Manageability	High	Low
Modifiability	High	Low
Reusabliity	High	Medium
Interoperability	Medium	Low

Conclusions and Future Work

Final Conclusion: Blockchain and smart contracts implemented with Hyperledger Fabric are not suitable in general for group disability claims processing at this time. This was determined by comparing quality attributes of existing, mature architectures with Hyperledger Fabric.

Future Work: Simulate adding organizations to the network on demand; Customized smart contracts between actors; Examine the implications of utilizing Private Data Collections to restrict access to data; Compare Private Data Channels vs multi-channel approach.

QR Code for Recording

Latency and Throughput Evaluation of Managed Databases on Public Cloud Service Providers: Azure Cosmos DB and AWS DynamoDB

Felipe Rodrigues
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00169675@myTUDublin.ie

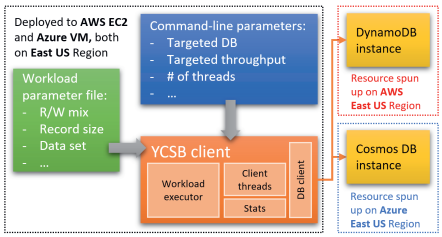
Introduction

By 2025, there is an expectation that the worldwide population will generate 463 exabytes of data daily. The industry increasingly heads towards applications that manage unstructured data. For the last two decades, Not Only SQL (NoSQL) databases have gained momentum as it addresses the shortcomings of Relational Databases Management Systems (RDBMS) scalability capabilities. Global-scale applications rely on scalable databases and cloud resources to ensure customers have the best user experience regardless of the incoming traffic. Cosmos DB and DynamoDB are scalable and highly available managed databases provided by Microsoft and AWS, respectively. Understanding how these managed databases work under different workloads and configurations is critical for software architects to ensure that proposed solutions meet business needs. This research aimed to investigate DynamoDB and Cosmos DB capacity mode and consistency level configurations to understand these databases and their configurations in more detail when handling different workloads, providing recommendations to software engineering teams on which database configuration is more suitable for specific workloads.

Research Goals

1) To run multiple workloads with different data distribution and operation types to recommend which database and configuration should be applied to specific workload types. 2) Assess whether Service Level Agreement (SLA) statements meet test results. 3) Provide recommendations rubric for engineering teams by comparing on-demand and provisioned capacity modes and consistency levels of DynamoDB and Cosmos DB (compared to other managed databases popular in the industry, research are scarce).

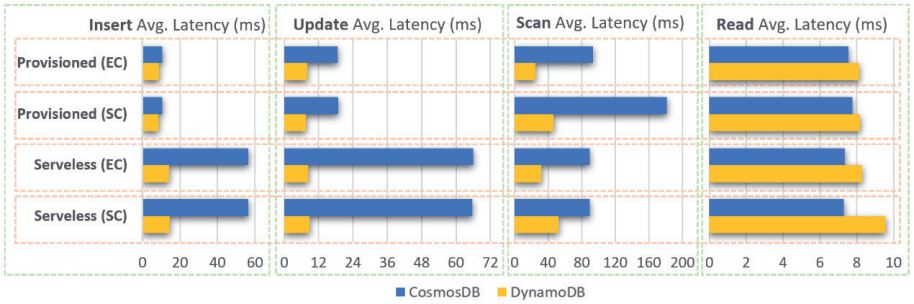
Tool and Test Environment



Yahoo Cloud Service Benchmark (YCSB) is a commonly used tool to benchmark databases. YCSB clients and managed databases were spun up on the same region to reduce latency.

Most Relevant Results

Reading, Insert, Update and Scan Operations:



Cosmos DB maintained the throughput and average latency with increased reading operations. Cosmos DB was more efficient than DynamoDB only when handling reading operations. Unexpectedly, serverless Cosmos DB databases handled reading operations more efficiently than the provisioned ones. Both databases on this type of workload met the latency claims stated in the Service Level Agreement (SLA). DynamoDB outperformed CosmosDB on all other operations. For DynamoDB, consistency levels impacted performance only when handling scan workloads. Both provisioned databases handled insert operations very similarly; however, serverless databases presented a higher latency average, and Cosmos DB handled insert operations at least three times slower on average compared to provisioned databases and serverless DynamoDB. Contradicting the SLA latency claims that should be under 10 ms. Likewise, Cosmos DB underperformed DynamoDB when handling update operations. Workload results that contained update operations revealed that Cosmos DB throttled about two per cent of requests due to high traffic volume and resource conflict, while DynamoDB showed no failures. Although Cosmos DB has more features and a more comprehensive SLA, DynamoDB has shown to be a more predictable product as results aligned with expectations.

Workload Definitions

YCSB Workload	Title	Description	Capacity Mode	Read/Write Consistency Type	# of Threads
Workload A	Insert Only	This workload is **100% write** operations.	Serverless & Provisioned	**Dynamo DB and Cosmos DB:** Eventual Consistency & Strong Consistency **Cosmos DB only:** Bounded Staleness, Session Consistency, Consistent Prefix	30 threads running 100k operations
Workload B	Update workload	This workload has a mix of **50/50 read** and **write** operations.			
Workload C	Read only	This workload is **100% read** operations.			
Workload D	Read latest workload	**95% read** operations whereas the **most recently inserted** records are the **most popular**. **5% insert** operations.			
Workload E	Short ranges	Complex paging **"scan"** operation. Requests up to 100 records that could potentially be **stored in different partitions**.			
Workload F	Read-Modify-Write	Workload composed of **100% read** operations and **modify and write back 50%** of the reading operations.			

Conclusions and Future Work

DynamoDB is recommended for systems that handle a high throughput of critical update operations, such as trading platform applications since Cosmos DB is not as efficient in handling the load while scaling up. Cosmos DB is recommended for systems that handle a high throughput of reading operations in large-volume databases, such as reporting applications, as it showed to keep the latency and throughput performance as the data grows. In general, avoid using serverless databases as provisioned databases workload results demonstrated to perform more efficiently. Consistency level does not impact throughput and latency as much as a capacity mode on a local scale. Cosmos DB partially met SLA claims, while DynamoDB fully met them. Evaluating these database configurations on a global scale is part of future work to gain more clarity.

QR Code for Recording

Mathematical Modelling of Firebase Stability and Consistency

Stanislaw STAWOWY
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180719@myTUDublin.ie

Introduction

This work is, to the extent to our knowledge, the first attempt at creating mathematical model of Firestore database persistence performance. As the amount of performance data in the available literature is very small, and cloud based NoSQL database providers don't offer any performance promises, the statistically meaningful results of the experiments performed, along with further analysis, allow for creation of mathematical Firestore write performance model and provide specific architecture and implementation recommendations for prospective Firestore users. In particular, we observed stable write sizes for data packets up to approximately 32 kilobytes, with write time rising sharply and predictably for larger ones. We also evaluated the difference in write times for inter-regional and intra-regional operations, relevance of geographical location of the database to write times, impact of multi-regional configuration on write times and provided set of possible explanations for observed results. The mathematical model of Firestore write performance presented in this work is the first time this subject was researched, as can be seen from available literature on this subject.

Firestore Day Variance

Firestore write performance does not seem to depend of time of the day in any of the six tested locations. It suggests that the underlying systems performance is not the reason for time required to write documents. We also found that shortest write time for documents for the case when both client and database are in the same Google Cloud region can be lower than one millisecond, especially for documents smaller than 32 kilobytes.

Methodology

We used Golang application, placed on GCP Compute Engine instances. This allowed us to place them in the locations providing minimal possible latency. We also used firestore.ServerTimestamp() Firestore specific field in our documents - this allowed us to measure only write time, as this field was filled by Firestore during write operation. We performed 1000 write operations for each document size and location combination, doing this on different hours to minimize possible influence of time-dependent data load on our results.

Research Questions Answered

1. Analysis how Firebase latency varies by region under message size variation:

We analysed the Firestore write performance from experiments run in four separate and geographically well dispersed Google Cloud regions into six Firestore locations (`asia-east1`, `eu-central2`, `eu-west3`, `us-east1`, `eur3` multiregion and `nam5` multiregion), and gathered statistically meaningful amount of data points. Additionally, we found that using multiregion locations (`eur3` and `nam5`) results in slower write times than using regional, geographically close locations.

2. Mathematical model of Firebase performance under message size variation:

We found that for document sizes larger than approximately 32 kilobytes, the write time increases linearly (with very good correlation between linear regression model and actual data), This allowed us to create mathematical model of Firestore performance:

$$T = (58.68 \times t - 0.3330) \times 10^{-6} \times n - y$$

where **t** is time required to save document smaller than 8-16 kilobytes (in seconds), and **y** is variable coefficient (also expressed in seconds), specific for particular source and target region combination.

3. Blueprint and recommendation rubric for engineering teams developing on Firebase

Write speed of documents smaller than 32 kilobytes stays constant, and write time for bigger ones can be estimated using above equation, rising proportionally to document size. This suggests that there is no reason to make documents smaller than 32 kilobytes, and that to achieve better performance, larger documents should be split if possible. The results show also that, to achieve best performance, geographically close Firestore locations should be used, and that using multi-region Firestore locations brings some write speed penalty.

Sample Figures - Persist Time and Day Variance

Figure 6: Time to persist document - `eu-west3`

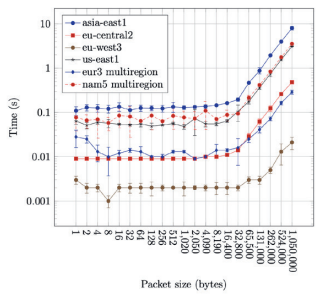


Figure 7: Time to persist document - `us-east1`

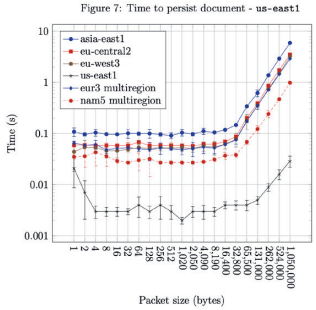
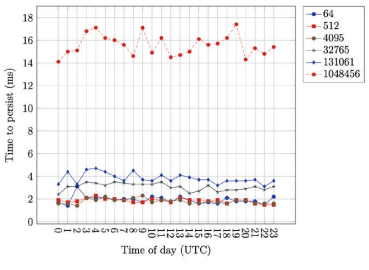


Figure 9: Day variance `eu-west3`



Conclusions and Future Work

The amount of in-depth research into cloud NoSQL offerings is very small, and finding any performance related data for Firestore is very hard. This work attempts to fill some of the missing information with up-to-date results. The analysis of the data gathered revealed some unusual behaviours of the Firestore, but additional research is needed. There are many possible directions in which further work on Firestore can be undertaken - one of them would be estimating y values for all region - region combinations, using even larger number of samples greater than 16 kilobytes. Another one would be work on write times for very small documents - again requiring statistically meaningful number of samples, which, due to small range of variations observed, would need to be very large.

QR Code for Recording

A study on Azure Service Bus queue performance

Author Raghunath Thekkemadathil
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180742@myTUDublin.ie

Introduction

Traditional single-application systems are not enough to meet all modern business requirements efficiently. Message-oriented middle-ware is a message queue pattern that can be used to build such complex distributed systems.The main participants in a distributed system that uses MOM are the producers and consumers.A sudden increase in the number of producers could result in a heavy load on the queue and the queue may be flooded with a large number of messages. In this scenario, a single receiver may not be adequate to process all the requests from the producers. Adding multiple instances of the same receiver could help to process the messages faster and thus increase the overall throughput and decrease the latency of process time.A competing consumer architectural pattern is very much useful in scenarios where each of the receiving tasks can be run asynchronously and in parallel without any dependency between them.

Motivation

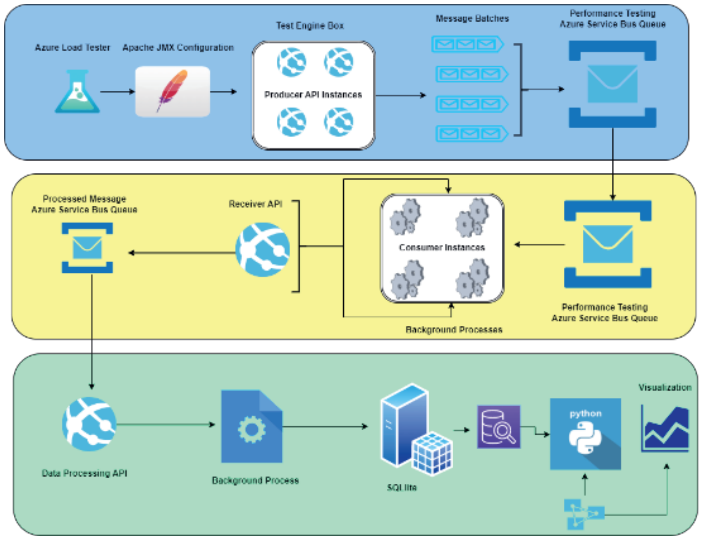
Increasing number of producers could produce larger number of messages injected into the queue and demands more receivers to process message packets. In the case of the Azure Service Bus queue, if predictable throughput and latency are required a premium tier service needs to be consumed. Although the standard tier of the Azure Service Bus queue doesn't guarantee a predictable throughput and latency, the motivation of this research work was to understand the level of variance in throughput and latency when the number of producers and consumers varies with a standard tier service.

Research

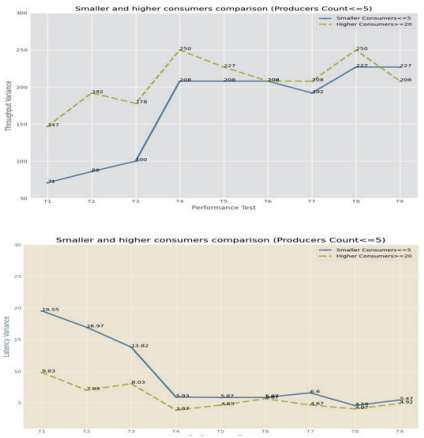
What is the variance of throughput and latency in the event of a varying number of producers and consumers for Azure Service Bus queue in standard tier?

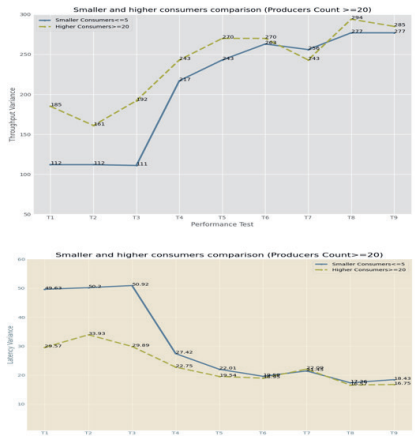
Test Cases: All the test cases were conducted with 2kb message packets in unreliable mode delivery with varying numbers of producers and consumers with prefetching and multithreading.

Test Architectural Framework



Performance Overview





Maximum pressure on queue

CaseId	Producers	Consumers	Latency	Throughput
cs-03.04	100	5	138	197

Conclusions and Future Work

Standard tier Azure Service Bus queue recorded an average throughput between 80 to 300 messages/second with a latency of 3 seconds to 50 seconds in the event of varying number of producers , consumers with multi threading and prefetching. The maximum pressure queue could withstand was 100 producers and 5 consumers for unordered, unrelaible message packets of size 2kb. To get a better understanding of the performance the future study also should take into consideration important factors like the message size, message ordering, message delivery guarantee and auto-scaling of receivers in addition to varying number of producers and consumers

QR Code for Recording

Retrofitting IIoT to a Manufacturing Plant

Seán Wallace
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180743@myTUDublin.ie

Introduction

The Industrial Internet of Things (IIoT) refers to the application of Internet of Things (IoT) technology in industrial settings. The term IoT can encompass a broad range of topics, including smart devices, (such as smart speakers, cameras, and sensors), edge/fog computing, supporting cloud infrastructure (including hubs, gateways, datastores), and related communication protocols. There has been some reluctance from industry to adopt these technologies, due to the large investment made in traditional automation systems to date. While research has been carried out into the benefits of installing IIoT technologies in a green-field site, this study investigates the possibility of retro-fitting IIoT to an existing manufacturing plant with the intention of then performing a phased migration to IIoT-native devices, using the Microsoft Azure cloud platform. A proposed architecture is suggested, and the cost-effectiveness of such an approach is analysed.

Method

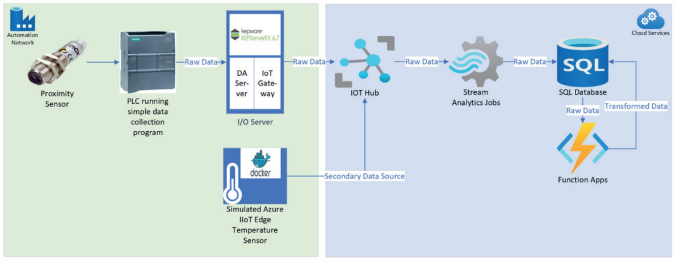
A proximity switch was connected to a Siemens S7-1200 Programmable Logic Controller (PLC) running a data acquisition program, which then communicated with a Kepware KepServerEx Data Acquisition server. An IoT Gateway plugin was installed and configured to publish data to an Azure IoT Hub over MQTT. This data was then passed to an Azure SQL Database via a Stream Analytics job. Data from a simulated IIoT-native Azure IoT Edge temperature sensor was published to the same IoT Hub simultaneously.

In order to compare carrying out calculations on the edge against doing them in the cloud, an Overall Equipment Effectiveness (OEE) calculation was implemented, both in the PLC program to replicate an ETL architecture, and using Azure Functions to represent an ELT architecture.

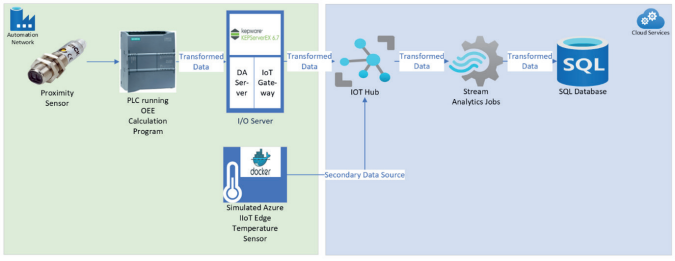
The cost and resource use of both of these architectures was analysed, and a theoretical exercise was carried out to calculate the cost implications of running either architecture over a longer period of time.

Given the number of variables that exist in the cost calculation, and the number of variables that exist from business to business, a generic high-level calculation was devised in order to help determine whether it is cost-effective to retrofit IIoT to a manufacturing plant.

Extract-Load-Transform (ELT) Architecture



Extract-Transform-Load (ETL) Architecture



Research Questions

Q1. Is it possible to gather sensor data from traditional PLCs and publish the data to the cloud using updated IoT platforms offered by traditional IT cloud providers such as Microsoft Azure?

Q2. Is it possible to simultaneously gather sensor data from both traditional PLCs and IIoT-native devices and log the data to the cloud?

Q3. What are the cost considerations of logging raw sensor data to the cloud and performing analysis in the cloud in comparison to performing analysis at the edge (i.e., locally at the manufacturing plant, such as on a pre-existing I/O server) and logging the resultant data (ELT vs ETL)?

Q4. Based on the data gathered, is it cost-effective to retrofit an IIoT solution to a traditional industrial automation system?

Cost of Logging OEE Data for an Enterprise

Item	Unit Cost	ELT Cost / 5 years	ETL Cost / 5 years
KepServerEx License	755/year	3775	3775
IoT Gateway Plugin	1735 perpetual	1735	1735
IoT Hub	9.618/month/unit	1154	577
Stream Analytics Jobs	0.12/hour * 730 hours/month	4985	4985
Database	14000/3 years	23000	23000
Azure Functions		1440	0
Total		36089	34072

*Calculations based on OEE data for 65 machines

Cost Effectiveness of Retrofitting IIoT

Due to the large number of variables that exists between businesses and industries, it should be left up to each individual enterprise/business unit to carry out a cost benefit analysis, using the following calculation:

$$\begin{aligned} Benefit = & ReductionInCapitalITCosts + \\ & ReductionInLocalOperationalITCosts - \\ & CostOfImplementation - \\ & IncreaseInCloudOperationalITCosts + \\ & IncreaseInProfitRealisedOnAdditionalGoodProduction \end{aligned} \quad (1)$$

Conclusions and Future Work

It is possible to log data from a traditional automation system to a Microsoft Azure SQL Database using IIoT technologies and protocols. It is also possible to log data from smart IIoT-native sensors to the same database in parallel. Logging raw data to the cloud will result in increased cloud compute costs when compared to logging transformed data. The cost effectiveness of introducing IIoT varies dramatically on the specific use case. In either architecture implemented for this study, it is likely that an Azure SQL database is not the most cost-effective datastore and an alternative such as blob storage should be considered.

Future Work: (1) Alternative cloud platforms; (2) Alternative Data Stores; (3) Implement ML for predictive maintenance; (4) Implement Azure IoT Edge; (5) Scale volume & frequency of data; (6) Implement cloud-to-device messaging; (7) Security implications;

QR Code for Recording

Geographic workload scheduling

Patrick Walsh
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
pwalsh31@hotmail.com

Introduction

Multiple approaches have been developed to optimise compute resource power consumption within data centres, but not *across* data centres. In a world where energy availability, cost, and attendant CO2 emissions are global problems, we can broaden our horizons, leverage the hyper-mobility of compute workloads, combine it with location awareness and grid condition or pricing data, and literally move the work to the optimal location and time for scheduling it: *geographic* workload scheduling has the potential to deliver lower power bills for data centre compute workload execution, and may enable specific optimisations around renewable power consumption or grid congestion events.

Electricity demand management

Demand management strategies at a consumer level currently focus on differential time of use (TOU) tarrifs, but take up rates are low. At an industrial level, and in particular in the case of data centre electricity demand management, the main approach to date has been to require data centres to provide on-site dispatchable power generation capacity, and disconnect from the grid supply at times of peak electricity demand.

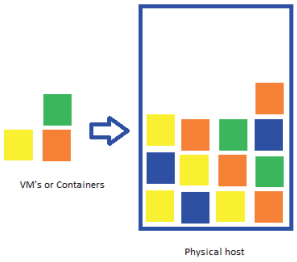
Data centre power optimisation

To date, a lot of the effort and attention in optimising data centre power consumption has focused on the portion of total power devoted to compute workloads, and minimising all other power consumption such as cooling, etc. The last decade has seen quite a bit of research into workload placement strategies aimed at minimising overall power consumption. But very little on architectural approaches to date, and highly focused within the data centre.

Conventional and alternative approaches

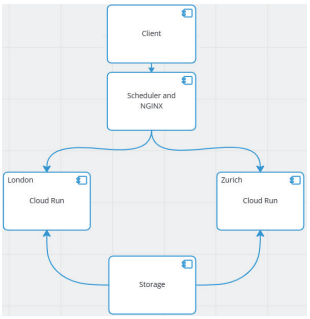
1. Bin packing approaches:

This class of approaches generally approach the problem as a bin-packing problem, and optimise power consumption by consolidating workloads on as few active machines as economically possible. Some examples are 'Minimum Migrations' (MM) and the FireFly Optimisation (FFO) scheme.



2. Geographic optimisation:

Research into workload characteristics suggested that at least a portion of data centre workloads were both time and latency flexible workloads. This gives rise to the possibility of building a time and latency flexible workload scheduler / placement scheme. Such a scheme allows us to leverage the mobility of compute workloads to take advantage of opportunities like lower power pricing in different locations, or greater availability of intermittent renewable sources.



Summary

- **Climate change is a global problem**
- **Grid management is a global problem**
- **Compute workloads are mobile**
- **Compute workloads are time flexible**
- **Compute workloads are latency flexible**
- **Practical, global solutions are possible**

Conclusions and Future Work

Future work should include construction of a prototype, extension to include statefull workloads, and further quantification of benefits. Stretch goals for future work include development of a generic work measurement system for a proof of work algorythm (Proof Of useful Work) so a distributed, autonomous, implementation could be exposed as a 'distributed data-centre' and managed as a blockchain DAO.

QR Code for Recording

Testing Ethereum Smart Contracts in Continuous Integration Pipelines

Stuart Woods
School of Cloud Computing and Digital Transformation, TU Dublin, Tallaght, Ireland
X00180720@myTUDublin.ie

Introduction

Blockchain has become more prevalent in the technological space over the last number of years. Predominately the utilisation of these is for Cryptocurrency, where an attack on the network can return a large financial reward. Attacks can occur by developing smart contracts with vulnerabilities, smart contracts are used to automate the execution of a transaction on the blockchain.
The focus of this research project was to investigate the integration of static and dynamic analysis and formal verification smart contract testing tools, into continuous integration pipelines to detect known issues in smart contracts before they are deployed to the Ethereum blockchain.
Four tools for testing smart contracts written in Solidity language were selected, Securify, Mythril, MAIAN and Manticore. These tools were integrated into pipelines which were developed utilising Microsoft Azure DevOps.
Once all the tools were integrated into pipelines, an analysis was performed, by running a selection of contracts that contain vulnerabilities from the Decentralized Application Security Project's top 10 vulnerabilities in Solidity code.
We developed five proof-of-concept continuous integration pipelines, one umbrella pipeline and one pipeline for each testing tool. These pipelines provided successful answers to our research questions; however, they did not come without challenges, including installation times and Solidity compiler version issues.

Ease of Installation

Tool	Analysis	Ease of Installation – (Vivar, et al., 2021)	Ease of Installation – Woods
Securify	Static	5/5	1.5/5
Mythril	Static	5/5	5/5
MAIAN	Dynamic	1/5	3/5
Manticore	Static / Formal Verification	5/5	4/5

Installation / Execution Times

Tool	Initialisation + Finalisation	Setup Time (Seconds)	Tool Running Time (Seconds)	Total Pipeline Time (Seconds)
Securify	7	228.4	1.6	237
Mythril	6	104.09	62.24	172.33
MAIAN	8	67.53	2.4	77.93
Manticore	5	36.1	59.11	100.21

Tool Maintenance

Tool	# of Commits	Last Commit	Open Bugs	Fixed Bugs	Percentage Fixed
Securify	16	5/9/2021	31	4	11.43
Mythril	4759	3/11/2022	86	696	89.00
MAIAN	15	4/11/2021	27	8	22.86
Manticore	1064	26/9/2022	229	562	71.05

Research Questions

1. Can static and/or dynamic analysis tools be integrated into continuous integration pipelines for testing Ethereum smart contracts?

The successful development of the proof-of-concept pipelines directly answers this question. We implemented five Microsoft Azure DevOps pipelines, utilising four different testing tools, two static analysis tools Securify and Mythril, one dynamic analysis tool MAIAN and one static analysis and formal verification tool Manticore.

2. How do the testing tools compare when it comes to ease of installation / integration, tool maintenance, execution times and error coverage?

We performed an analysis on the four testing tools comparing the tool requirements, the ease of installation and integration, the length of time to install and execute each of the tools and the maintenance of the tools. The findings are presented in the analysis section.

3. Do smart contract testing tools detect known vulnerabilities in a smart contract and can testing tools be combined in continuous integration pipelines to reduce the likelihood of missing them?

The results of the research project show that the testing tools are not detecting vulnerabilities in smart contracts that have been deployed to the Ethereum blockchain that we know are in good order, however it is imperative to combine testing tools due to the testing tools not detecting some known vulnerabilities in Ethereum smart contracts. This can be seen in the testing tools vulnerability analysis

Testing Tools Vulnerability Analysis

Category	Number of Contracts Analysed	Securify		Mythril		MAIAN		Manticore		Combined	
		Expected	Actual	Expected	Actual	Expected	Actual	Expected	Actual	Expected	Actual
Access Control	5	✓	5/5	✓	4/5	✓	1/5	✓	4/5	✓	5/5
Arithmetic	5	×	2/5	✓	4/5	×	1/5	✓	3/5	✓	5/5
Bad Randomness	5	×	0/5	×	2/5	×	3/5	×	2/5	×	5/5
Denial of service	5	×	3/5	×	1/5	×	0/5	×	0/5	×	4/5
Front running	4	✓	0/4	✓	1/4	×	3/4	×	2/4	✓	4/4
Reentrancy	5	✓	0/5	✓	3/5	×	2/5	✓	3/5	✓	5/5
Short addresses	1	×	0/1	×	0/1	×	1/1	×	0/1	×	1/1
Time manipulation	4	×	1/4	×	3/4	×	0/4	✓	1/4	✓	4/4
Unchecked low level calls	4	✓	0/4	✓	4/4	×	0/4	✓	2/4	✓	4/4

Conclusions and Future Work

To conclude, this research project was a success, shown by the development of CI pipelines which integrated testing tools for Ethereum smart contracts. The answers to our research can be seen above, however we also noted some weaknesses in the project, namely relatively long installation times and issues with the version of Solidity compiler used.
During our research we noted some potential future work ideas.
1) Managing self-hosted Microsoft Azure agents to reduce the costs of hosting the pipelines, and reducing the installation times.
2) Modify the testing tools to utilise the same dependencies so that the four tools could use the same pipelines.
3) We implemented Manticore utilising its command line tool, however the tool is more flexible. It has a Python API for custom analyses and application specific optimisations. Future research could investigate using available APIs in the testing tools for a more robust and customisable CI pipeline.

Investigate the effect of Istio and Linkerd service mesh installation on scaling performance

James Salisbury
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00180716@myTUDublin.ie

Introduction

As microservice architecture is adopted by more and more applications, service meshes have become the go-to tool to simplify traffic management, error tracing and to secure inter-service communication. While service meshes advertise themselves as plug-in solutions to these problems, they use resources that otherwise would be available by the applications themselves. While previous research showed that service meshes do use compute resources and that choice of service mesh can have a significant impact on resource usage, there is little, if any, on the effect this has on scaling and the associated costs.
The results of this project showed that service meshes do increase node scaling under high loads but this can be mitigated by careful selection of the scaling parameters (CPU and memory). It was also discovered that the choice of service mesh has a significant impact on scaling, with Istio scaling to 2-3 times more virtual machines than when no service mesh is used and up to double that provisioned by Linkerd. However, Istio showed the best performance under high loads, with latency and dropped calls being the lowest of the three tested setups under very high loads. This is based on the services tested in this project and under these particular resource scaling conditions.

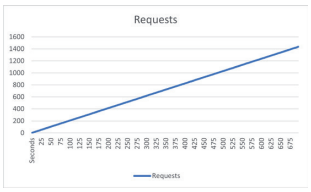
HPA Scaling Thresholds

The following table shows the horizontal pod autoscaler thresholds used in this experiment.

CPU	Memory	CPU & Mem
80%	80%	80%
50%	50%	50%

Load Testing

For load testing, the number of requests was increased by 2 every second, until a total of 1 million requests were reached. This load pattern was chosen as it guaranteed node scaling would occur under all scaling thresholds. The GHZ load testing tool was used for this and was installed in a different VM in the same region and zone as the K8 cluster.



Test Subjects and Research Questions

Microservices

Two microservices were chosen from the "Online Boutique" example microservice application from Google. These were:

- Ad Service - This service provides text-based advertisements for the front end based on context words or randomly if no words are provided. It uses the highest amount of resources.
- Currency Service - This service provides the available currencies available to the user and also their currency exchange rates. It uses the lowest memory and CPU resources

These services were installed on a single node Kubernetes cluster in Google Cloud Platform.

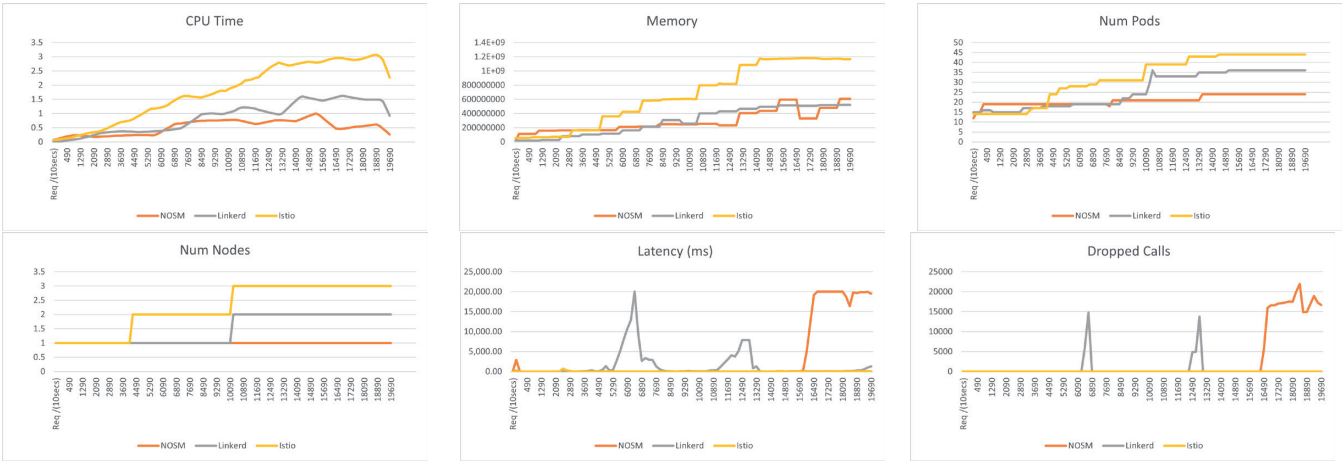
RQ1 - To what extent do the service meshes impact scaling when compared to when not used?:

It was discovered that Service meshes can increase resource usage and therefore scaling.
However this is not a given. It is dependent on the scaling thresholds set for each of the two resources examined, under these particular testing conditions.

RQ2 - To what extent does the choice of service mesh impact scaling?

The choice of service mesh has an effect on node scaling. Linkerd finishes with same number of nodes as no service mesh in 3/6 scenarios for adservice and 5/6 for currency service.
Istio finishes with more nodes than NOSM in 6/6 of the ad service tests and 6/6 for currency service.
The comparative difference between the two service meshes was less in the currency service, likely due to its lower resource usage.

Example Results - Currency Service Scale at CPU & Memory 80%



Conclusions and Future Work

It was found that service mesh choice does affect resource usage and therefore scaling. While Istio used more resources and scaled to more nodes that the No Service mesh and Linkerd setups, it performed far better on latency and dropped calls.
Future work on this topic could involve using different scaling metrics such as requests numbers. The experiments could also be performed on other microservices in the application, or on the application as a whole.

QR Code for Recording

NOTES