National College of Ireland

Robotic Process Automation using Container-related Methodologies

MSc Research Project
Cloud Computing

Jaison John
Student ID: 19104910

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquiredo

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Jaison John
Student ID:	19104910
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Manuel Tova-Izquiredo
Submission Due Date:	17/08/2020
Project Title:	Robotic Process Automation using Container-related Methodologies
Word Count:	5937
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	23rd September 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Robotic Process Automation using Container-related Methodologies

Jaison John
19104910

Abstract

The World is moving towards a new normal of "Work-From-Home" in this pandemic which has led to the adoption of Intelligent Automation more than ever before. And what better than Robotic Process Automation(RPA) which is an emerging technology based on software robotics to aid the automation of business processes. However, the use of RPA can be revolutionized by using the container-related methodologies. To realize this objective, the concept of Cloud Robotics have been analyzed in this paper with respect to both Hard and Soft Automation. Existing methodologies of different architectures have been reviewed, identifying its corresponding gaps and thus laying the groundwork for better practices in the field of RPA which comes under Soft Automation. Inspired from the Resource allocation techniques of the containers, a new Queue-based bot locking algorithm has been implemented using Blue Prism RPA tool with virtualized "Multi-Robot" architecture. This has helped in achieving better scalability and increased performance. It has been successfully implemented on an Insurance Web application to solve its pandemic-related surging problems. An evaluation using Multiple Regression is conducted to determine the performance difference of Multi-Robot RPA against the traditional RPA along with determining the mathematical functions for the same.

1 Introduction

While the world is slowly coming out of the pandemic slumber, Engineers throughout the world are constantly researching to develop and innovate new applications and processes that would help the mankind to tackle such calamitous scenarios. One of the primarily focused research is on the extension of Cloud Computing principles. This research focuses on how their applications can be done in the field of Robotic Process Automation(RPA).

1.1 Background

The world is shifting towards the paradigm of Containers. Gone are those days, when there were separate environments for development and deployment of various applications. Containerization has eliminated this problem by bundling the run dependencies along with the application code. While Containers is a relatively older term, RPA is on the horizon with its ability to automate complex applications efficiently in a non-intrusive manner. To explore further, Process Automation is generally classified into two parts, namely, Hard Automation and Soft Automation (Issac et al.; 2018). In Hard Automation, designing of a physical robot or machine is done to perform a set of specific repetitive tasks whereas in Soft Automation, a more developed version is provided where software bots can be programmed in the computer as per the requirements to perform a set of tasks. And Robotic Process Automation (RPA), rightly comes under Soft Automation (Lacity and Willcocks; 2015).

As per Forrester estimates by 2021 (Clair; 2018), there would be more than 4 million robots carrying out administrative and office tasks. Thus, Robotic Process Automation is a software that can be programmed to automate different forms of activities and processes that are otherwise carried out by humans thus reducing their burden of repetitive tasks. RPA is suitable for processes with high transactions volume, structured data, logic-driven procedures, low variance, etc. (Leshob et al.; 2018) The software robot is fed with a workflow of instructions to perform operations which are based on the application of Artificial Intelligence as it allows companies with legacy systems to automate their tasks. Different RPA tools in the market include Blue Prism, UiPath, Appium, Kofax, Workfusion, etc. out of which this research focuses implementation using Blue Prism since it is one of the pioneers in this industry.

1.2 Motivation

RPA often has a single robot assigned for carrying out a particular process. Though multiple robots can be used on multiple processes, the idea of using multiple robots for a single process can increase its scalability and also the performance of the task. This could be achieved by using a newer architecture with multiple bots installed using Queue-based algorithms.

Moreover, the world is plunging towards a global recession due to the chaos created by the Coronavirus pandemic (Leiva-Leon et al.; 2020). Global recession would eventually accelerate Intelligent Automation adoption (Ingalls; 2020). As the World economy goes further into the recession, most of the companies would look to intelligent automation to reduce unnecessary operation costs and thus survive themselves in a highly competitive market. This makes the need of hybrid RPA solutions even more relevant and proper research in this field could seed to brilliant start-up ideas for adventurous entrepreneurs.

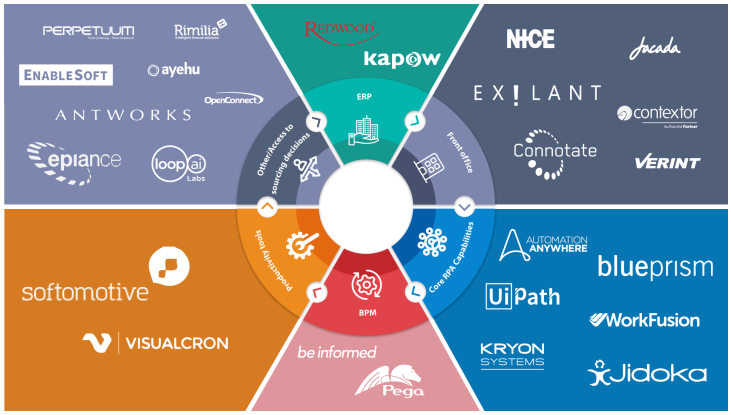

Figure 1: RPA in a nutshell

This paper discusses on how to exploit Cloud capabilities in the field of RPA thus analyzing the research work done on cloud robotic models and its resource allocation methodologies in various containers.

1.3 Research Question & Objectives

Can Robotic Process Automation(RPA) be improved by using Container-related methodologies?

How can orchestrating containers or its Resource allocation techniques be implemented in RPA to provide better performance and scalability?

The main objective is to improve RPA by combining it with virtualization techniques and container-related allocation techniques to offer higher scalability and better performance for the overall solution. The secondary objective is how RPA can be used to solve a real-world problem arising due to the present pandemic.

1.4 Limitations

Blue Prism being a licensed RPA tool hasn't been led to much explorations in contrast to the other open-source tools like UiPath, Robocorp, etc. Hence, the process of deploying multiple robots for a single process requires multiple licences thus increasing the cost of the proposed RPA architecture. Implementation of Blue Prism in this project was carried out under trial licenses which had some restrictions.

1.5 Structure of the report

The next section gives a literature review about the use of Robotic Models in both Hard and Soft Automation along with scheduling allocation techniques. Section 3 talks about the methodology used while Section 4 discusses the design specifications and architecture. Section 5 gives a description about the implementation aspects followed by Section 6 which gives a detailed evaluation of results. The final section gives the overall conclusion along with scope for future work.

2 Related Work

This section along with identifying a desired set of criteria to deploy RPA, mainly discusses about the different frameworks involved in Robotic automation. It also analyzes different container scheduling algorithms and its architectures which can be deployed in the frameworks to gain better scalability and performance.

2.1 RPA challenges

This section discusses the challenges involved in implementing RPA and encompasses a list of desired criteria which could help the Robotic Process Automation to work more efficiently.

RPA being the new horizon in the market, is ideal for applications(Leshob et al.; 2018) which require high level of i) process standardization, ii) transaction volume, iii) process maturity, and iv) business rules. However, it has some limitations which needs to be improved. Though front-end integration of RPA with applications can enhance its flexibility along with the speed, its back-end integration is not suitable for machine-to-machine communication. Aleksandre Asatiani Asatiani and Penttinen (2016) explains that RPA presents a temporal solution which can fill in the void for some manual processes based on legacy IT system and the ones running on fully automated systems. While outsourcing have its own disadvantages, its practice have a proved record with a variety of business needs and applications. RPA, however, is missing out on these needs thus posing quite a dilemma to its potential customers.

Hence, potential customers needs a proven business case to collaborate with the RPA providers. Combining the papers of (Fung; 2014) and (Slaby; 2012), a set of criteria has been identified where RPA can be deployed as displayed in Fig.2.

Criteria	Description
High volume of transactions	Task considered for RPA is performed frequently or includes high volume of sub-tasks.
Need to access multiple systems	Task involves accessing multiple systems. Example: copying data from a spreadsheet to a customer registry.
Stable environment	Task is executed within predefined set of IT systems that remain same every time a task is performed.
Low cognitive requirements	Task does not require creativity, subjective judgment or complex interpretation skills.
Easy decomposition into unambiguous rules	Task is easy to break down into simple, straightforward, rule-based steps, with no space for ambiguity or misinterpretation. Example: Allocate all incoming invoices from Company X with value 3000€ or more to category Y.
Proneness to human error	Task is prone to human specific error, not occurring to computers. Example: matching numbers across multiple columns.
Limited need for exception handling	Task is highly standardized. Little or no exceptions occur while completing a task.
Clear understanding of the current manual costs	Company understands current cost structure of a task and is able to estimate difference in cost and calculate return on investment (ROI) of RPA.

Figure 2: Criteria for RPA

It paved out a way for Cloud Robotics which is nothing else but a combination of cloud and robotics that made a breakthrough in the Robotics domain. Using the auto-scaling feature of the Cloud, RPA processes can effectively meet customer's varying demands and workloads with very less latency. The different strategies that are used and its details are discussed in the next section.

2.2 Smart Cloud Robotic System

This section identifies different strategies used in the field of Cloud and Robotics identifying the gaps and suggesting improvements in the same. It helps to identify the key applications along with some of the architecture practices followed in Hard Automation which can be realized in Soft Automation using RPA tools.

"Cloud Robotics" (Kehoe et al.; 2015) can be defined as any robot or automation system that depends on different data from a network to support its undertakings, i.e. where not all computation, sensing and memory is incorporated into a single standalone system. DAvinCi (Dyumin et al.; 2015) is one of the best examples of a cloud robotics software architecture built out of Robot Operating System(ROS) and Hadoop to provide scalability and parallelism for different service bots. Duan Yong Duan and Yu (2016) has discussed the architecture of multi robot system using cloud-robot framework. It consists of various layers such as physical resource layer, resource pool layer, intermediate manage layer, service-oriented architecture and robot client layer. Functions like parallel processing, load balancing, fault tolerance were realized by using the cloud computing model. By using multiple robots, the overall accuracy and real-time performance was significantly improved. However, the architecture is designed for Hard Automation thus failing to address process-oriented tools for Soft Automation.

Lujia Wang et al. Wang et al. (2013) have proposed a game theory based auction for allocation of resources in a multi-robot environment. Due to multiple robots being deployed, a large amount of data was considerably generated resulting into bottlenecks in bandwidth. To address the limited bandwidth and other challenges like CPU occupancy for parallel computation, limited number of proxy hosts, etc they used game theory with a joint-surveillance experiment scenario. Game theory has been one of the best state-of-the-art algorithms in decision making. It includes the mechanism of market-based negotiation which is applied in grid resource management and sequential bargaining mechanism. Although sequential bargaining can generate optimal solutions Sim and An (2009), it requires more communication and computation requirements in distributed paradigms. However, a well-defined Wang et al. (2013) set of Quality of Service(QoS) can greatly help in improving the quality of a resource allocation mechanism. It also included a dynamic priority scheduling method which were implemented by logic programming. The experimental results improved the CPU usage, thereby reducing the bottleneck and computational complexity.

Another strategy for maximizing bandwidth in multi-robots system was developed by Julio and Bastos Julio and Bastos (2015) with Dynamic Bandwidth Management Library(DBML). It was developed in ROS to segregate functionalities into independent modules for code reuse. The system prioritizes communication channels with respect to each environment event thus offering greater bandwidth for important channels. In ROS, multiple nodes are combined together into a graph using RPC and Parameter Server. The package architecture was divided into two libraries of Publisher and Subscriber along with a default optimizer node. They effectively carried out an experiment on Tele-operating

system using DBML and the proposed algorithm. However, bandwidth rate was assumed to be fixed while running the library. This can degrade the system performance in real-time environment where bandwidth of the wireless links depends upon the proximity of nodes location and number of obstructions present. However this problem was successfully managed by Wen et al. (2016) where they proposed a cloud platform, micROS-cloud, which supports the direct deployment of ROS software packages. With the container-based isolation and on-demand instantiation, the packages can be automatically converted into Internet-accessible services.

Multiple robots can thus access a service simultaneously even if the corresponding ROS package is designed initially to serve a single bot. This Hard Automation-based "Multi-Robot" inspires the architecture for our Soft Automation which would be carried out using RPA tools. Instead of deploying multiple robots for multiple processes, deploying multiple robots for a single process would help in achieving greater performance and flexibility.

"Cloudroid", another cloud robotic framework (Hu et al.; 2017) has a more transparent and QoS-aware software framework where quality of service(QoS) such as timeliness has been considered which is very critical to robot's behaviour. Along with the packages for direct deployment of existing software packages, it includes automatically generated service stubs which helps the robotic applications to outsource their computation without any modification in code. One of the famous "Monte Carlo sampling process" (Kehoe et al.; 2014) in robot grasp planning is parallelized to cope with uncertainty by the cloud computing clusters. However, its solution is specific task-related and it cannot be generalized to other tasks. Hence, Hu et al. (2017) have provided a general solution on the infrastructure level. They have also tried to address the gap between the ROS package model and the cloud service model which supports ubiquitous access and rapid elasticity with multiple clients.

To bridge the gap between them, Cloudroid have introduced four mechanisms:

1. Self-contained VM encapsulation
2. Cloud Bridging
3. On-demand Servant Instantiation and Multiplex
4. Service Stub Automatic Generation

Thus a comprehensive study of Cloud Robotics was conducted which includes most of the strategies for Hard Automation whose cloud methodologies could be realized with some more improvement in Soft Automation.

2.3 Scalability and Scheduling in Container Orchestrations

This section analyzes about the various scheduling policies in Container Orchestrations and different architectures that have been followed to gain maximum scalability and performance. It also identifies the research gaps and suggests future scope for its improvement.

Efficient resource utilization offers container scalability, for instance, response time of web requests created on PAS algorithm includes proportional-integral-derivative controller(PID). de Abranches and Solis (2016) have conducted these experiments using Docker container, HAproxy load balancer and Redis database with varying workloads. They emphasized on improving the PID parameters to scale containers. Though the implementation is built on the control theory scalability, they failed to prove better results on elasticity through existing methodologies. Another instance of evaluation of container

performance was conducted by Fr Jaison Paul & team (Preeth et al.; 2015) with Bonnie++ bench-marking tool and metrics like CPU utilization and memory usage with psutil. Comparison between Docker containers and host machine on disk usage, memory usage and network I/O was made to realize that containers outperformed host system on resource utilization.

Kandan et al. (2019) has argued that Default scheduling algorithm selects the resource based on the current trend of resource utilization. Their proposed scheduling algorithm differs from the default scheduling algorithm which is based on the identification of future utilization of resources. Once request manager validates the container request, it checks for the feasibility of deploying the requested containers. The proposed solution analyzes the future utilization on the targeted resource before it is deployed to reduce the migration of resource and ensure smooth functioning across all the containers. Here the solution is only for the experienced users who has the upfront information which fails in the case of new users. They are assumed to give correct initial placement of containers which poses a big risk on their proposed algorithm.

Container application deployment has helped in the advancement of microservice architecture. This architecture is a well defined Docker container management which offers insight on core and container scalability Inagaki et al. (2016). It can easily identify the bottlenecks of both the core and container scalability that could affect the performance of container orchestration. It also helps in the mitigation of operation issues by analysing different layers more efficiently. This would help in achieving higher scalability and availability. However, Containers lack stability for windows-based applications. They fail to provide features for terminal session which is very essential for their deployment. Other possibility would be the combination of VMs and containers which is again not a good option as Mavridis and Karatza (2019) explains that their combination can affect the performance of virtualization due to increased overheads.

Use of Virtual machines alone can achieve convincing results as far as windows applications are concerned. However, to keep all the machines in sync for a given task is bit of a challenge. Zhang et al. (2013) presents a queue-based lock algorithm for shared-memory multiprocessors whose clients accesses multiple containers. Their methodology is based on a non-blocking queue which acts like a scalable lock algorithm for multi-systems architecture. Instead of deploying separate locks for each resource, this queue acts as a centralized manager. It has achieved better results since FIFO nature implies fair acquisition of locks. Due to lower access overhead, it has better scalability thus increasing the performance of the task computation.

2.4 Conclusion

Thus, a collaborative study of various implementations of Cloud Robotics along with different scheduling policies and resource allocation methods in Container Orchestrations were reviewed and the gaps were identified. Though there have been many robotic frameworks designed for Hard Automation to gain better performance, the same is not true for Soft Automation(RPA). Lack of appropriate architectures in RPA has led this research to propose a new efficient architecture based on the Multi-Robot framework. Using virtualization and resource locking techniques, the scalability and performance of RPA can be improved significantly.

3 Methodology

This section details the various steps taken while approaching the research project. It also includes information regarding the equipment used and the preparation of data along with their measurements.

3.1 Steps taken

The first and foremost step taken was to select the appropriate RPA tool. Different RPA tools were considered like UiPath, Appium, Blue Prism, etc out of which Blue Prism was eventually selected because of its business agility.

Since this research was intended to solve a real-world problem arising due to pandemic, the domain of Health Insurance was carefully chosen. It was reported that many health insurance companies had a surge of applicants due to Covid-19. [1]

The architecture was inspired by the"Multi-Robot" (Wen et al.; 2016) framework as discussed in the literature review subsection 2.2

It was also planned to use Dockers as Orchestrating Containers, however Blue Prism being a Windows-based application needed a terminal session which was not supported by containers. Hence, it was more appropriate to use VMs (Mavridis and Karatza; 2019) for operating multiple bots through Blue prism clients.

Locking algorithms of containers Zhang et al. (2013) were used to formulate a new algorithm named "Queue-based bot locking algorithm" exclusively for RPA framework to handle multiple bots simultaneously.

Different external libraries were created as Blue Prism objects which were mapped to the corresponding processes accurately.

The Blue Prism control room was used for the execution of various processes by the different bots in a synchronized manner.

The Blue prism log manager provided the status of execution for each bot and the individual and total time taken to complete the task.

3.2 Equipment used & set-up techniques

Blue Prism was used as the RPA tool for deploying the proposed architecture. Though Blue Prism was a licensed tool, it was preferred over the other open-source RPA tools like UiPath because of its client-server architecture. Also it has tremendous speed and proven business agility. Multiple virtual machines were added as Blue prism clients which were included to provide scalability and increase performance. These were connected to the global database to maintain consistency.

Different user roles were identified and each role was configured with a set of operations. Only the admin had the sole right for all operations at once. Earlier security group was created along with the group membership for appropriate domain users as shown in the below figure. The input file which was passed to the process studio across different bots was shared through a common network drive.

[1] Health Insurance surge: `https://www.prnewswire.co.uk/news-releases/health-insurance-broker-experiences-surge-in-coronavirus-related-enquiries-858566783.html`

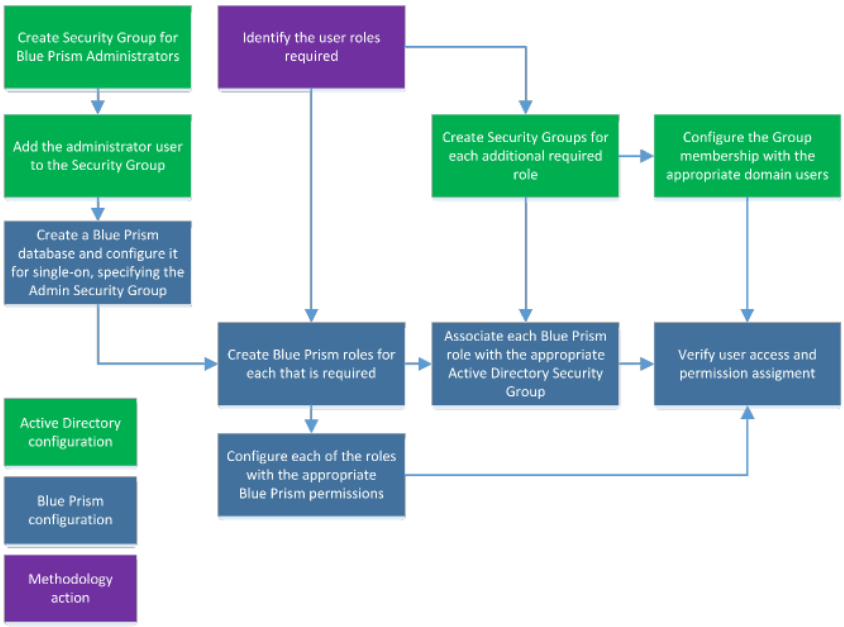

Figure 3: Configuration of Blue Prism User roles

3.3 Data prepared

Since the task involved software bots carrying out operations in the web application, data preparation included collection of input data needed for the bot to perform the specific operation. Laya Travel Insurance(part of AIG), one of the leading Health Insurance companies was taken into account for real-time application. Earlier the datasets were passed through the Blue Prism process itself. Eventually the data was passed in through an Excel file which included the travel dates and date of births of each individual. Travel dates were taken as per the companies' policies which cited that both the journey and return dates should be within a month. The ages were considered for all groups, children, youth and elders to check whether they give the insurance details correctly. Also the listing column was synchronized with the queue manager in the Blue Prism application.

3.4 Measurements made

Measurements were made on different basis from process execution time to the consistency of each bots performing the tasks. The control studio shows each different items from the work queue along with the resource bot executing it. It gives the total number of time taken by each resource and also gives information about the number of attempts made while executing a particular task.

The calculation of time execution for this Multi-robot architecture were also made against the traditional approach without any use of virtual machines to understand the performance change. Incorrect data and null values were also passed in the input file to see how the bots performed under Exception scenarios. The updation of the Excel file was also done using a different lock mechanism so that there is no inconsistency during its updation from different bots. Blue Prism also kept an interactive dashboard for graphical analysis of different workforce availability and their process execution.

4 Design Specification

4.1 RPA Architecture

Following diagram depicts the architecture for the research work which follows a "Multi-Robot" approach.

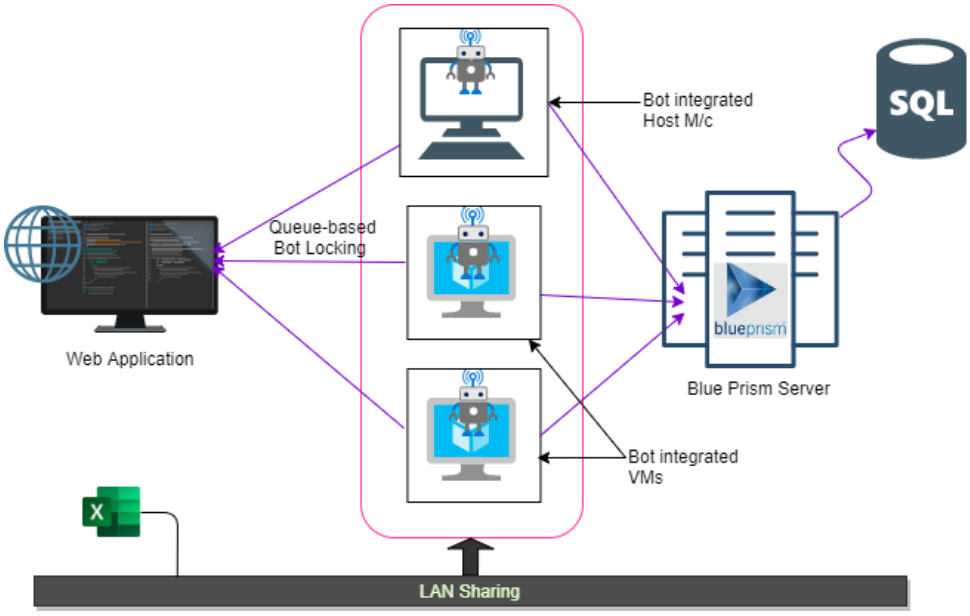

Figure 4: Blue Prism Multi-Robot Architecture

This architecture consists of Blue Prism as the main RPA tool. This "Multi-Robot" approach follows a Client-Server design with Blue Prism client application(RPA bot) being installed in both the host and the virtual machines. All these RPA bots are then connected to the Blue Prism server.

The Blue Prism Server is connected using secure AES-256 encryption technique. This server is also connected to the global database server. These Blue prism clients residing on the host and virtual machines act as multiple bots to perform various tasks that has been designed using Blue Prism process and object studio. All these bots simultaneously perform operations on the web application using a new Queue-based Bot locking algorithm which was inspired by the resource locking algorithm of the containers. These bots work synchronously with each other to get the data from the input Excel sheet shared to all the machines using drive mapping through LAN.

Initially one of the bots acquires a lock to access the Input sheet from the mapped drive and gathers the data into its collection dataset which is accessible by all the bots. When other bots find that the lock has already been acquired, they directly proceed to the task execution phase and retrieves data from the collection dataset. The common repetitive set of activities is performed by different bots instead of one single bot thus maximizing the performance of the task execution and increasing its speed considerably. These bots after performing each round of task, marks it complete and then proceeds to the next item in the work queue until the queue list is empty.

Once all the tasks are executed, the results obtained are updated in the input file in a timely fashion. The algorithm along with the flowchart is described in the subsection 4.3

4.2 Process and Object Studio interaction

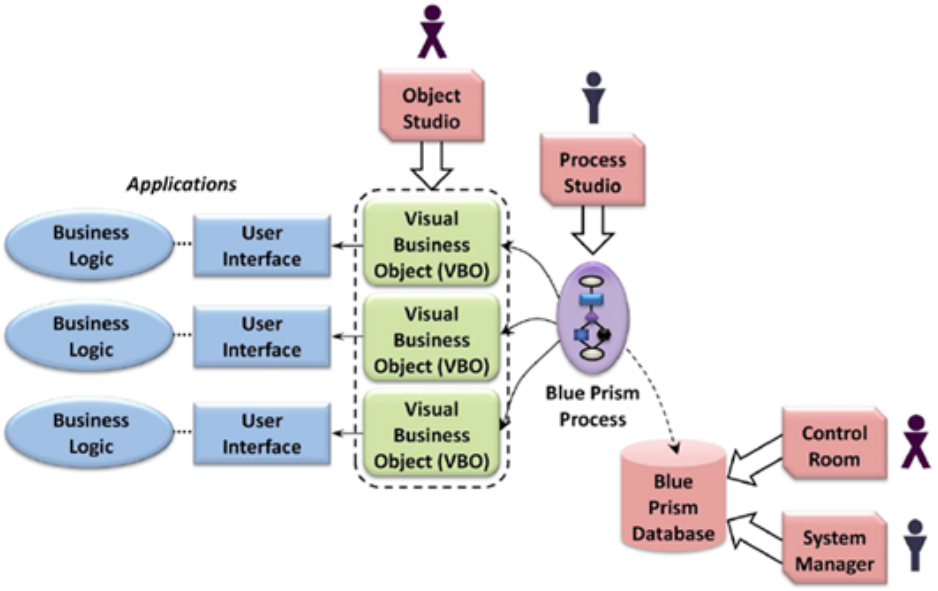

Figure 5: Inside Blue prism client

- Process Studio: Based on the .Net framework, it is the area where the process is developed. It includes all the actions, loops, exception handling, business logic, etc. The different objects created in the object studio and the imported libraries are called in process studio to define various functions.
- Object Studio: Object studio is ideally used to create the functionality of the task so that it can be deployed by various processes. It helps to create Visual Business Object(VBO) which helps in consistency and scalability. Bot maintenance becomes much more easier because the system changes could be updated directly in the main location instead for every process. Thus it helps in reducing bot development time for subsequent processes.
- Control Room: Control room is used to execute the processes through various resources(bots). It shows the account of the currently running sessions and gives the status, time required and the process details information. It includes the work queue which shows all the status of the items in queue along with their resource, attempt and execution time details. It also includes the scheduler and data gateways.
- System Manager: The System manager includes the exposure, management, history and environment variables of all the processes and the objects in the system. It entails configuration settings for SOAP Web Services and its APIs. The encryption scheme along with key details are stored in this manager. Various Users, user roles, credentials and licensing details are also included here. It essentially stores all the audit logs, process logs and object logs too.

4.3 New Queue-based locking algorithm

Algorithm 1 Queue-based Bot locking

```
init bots
init taskQueue, init maxLimit
check lockExists
if lockExists is empty then
    acquire lock
    create instance
    get data as collection
    add to queue
    release lock
    proceed taskQueue
else
    proceed taskQueue
    retrieve collection
end
while taskQueue ≠ 0 do
    while taskAttempt ≲ maxLimit do
        get next item
        execute all actions
        if action ← Exception then
            Mark action ← Exception
            get next item
        else
            add rows to collection
            Mark as completed
        end
    end
end
write collection(Excel updation)
```

Thus Multiple bots can execute tasks simultaneously in a non-blocking manner.

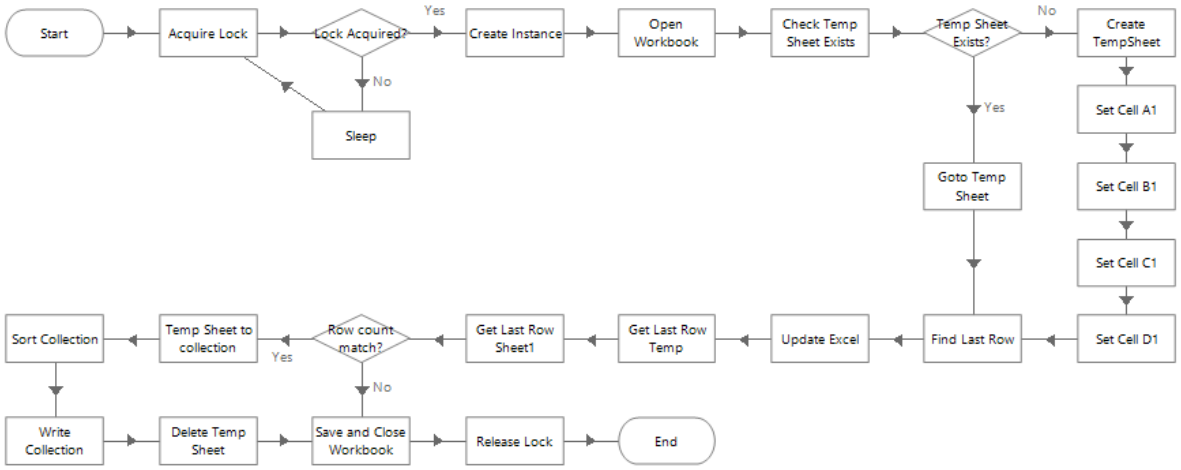

Figure 6: Excel updation flowchart

5 Implementation

Using the "Multi-Robot" architecture, the Blue Prism was installed and configured in the host machine as the tool comes with both server and client application respectively. Virtual machine instances were created on which only the Blue prism client applications were installed. All of these machines were connected with the Blue prism server using secure AES-256 encryption technique.

After connecting the server to the database, it was ensured that the input Excel file is shared through drive mapping for all the connected machines. The whole architecture was implemented using Queue-based Bot locking algorithm for multiple bots so that they can execute the task simultaneously without blocking any other bot. The final updation of the Excel file was done using the algorithm as per fig. 6

The RPA execution can be done in multiple ways, i.e. it can be either done from the process studio or from the control room. This research uses process studio execution because it allows us to monitor each and every action of the complete process(fig.7) and thus helping in the overall Testing of the solution to debug each and every exception.

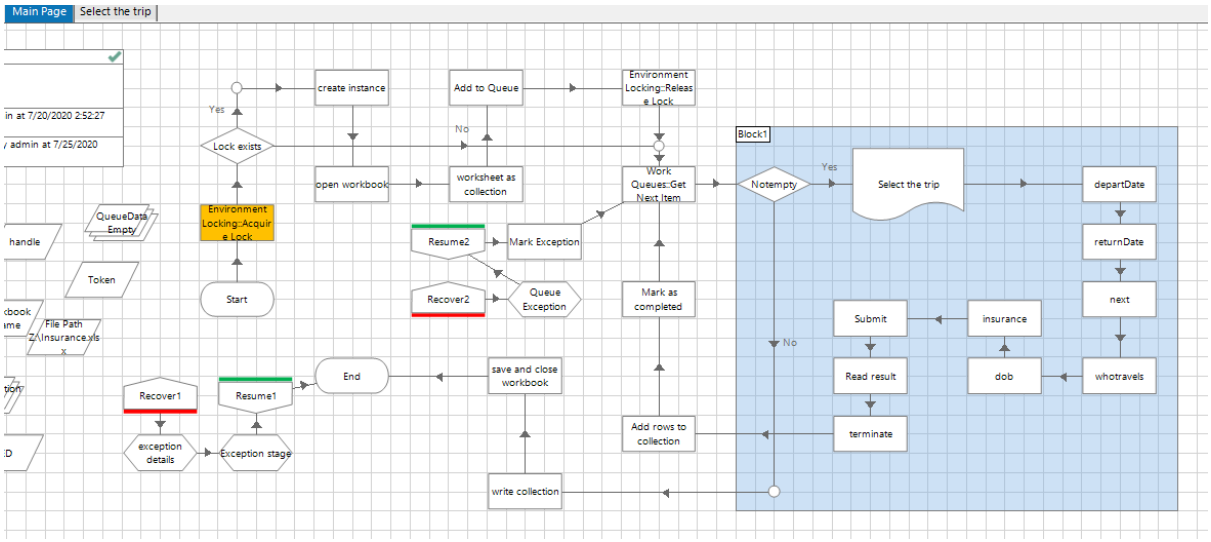

Figure 7: Process Studio Implementation

Since RPA is mainly used for a set of repetitive tasks, the "Multi-Robot" architecture was implemented on such a healthcare web application of Laya Travel Insurance wherein different sets of insurances have to be calculated based on the travelling dates and ages of the passengers. Instead of using one bot to carry out different sets of execution as in the traditional approach, this novel approach used multiple bots to carry out different sets simultaneously thus increasing the scalability of the RPA. Also the total execution time was reduced significantly thereby bettering the RPA performance.

Tools and Technologies used:

RPA Tool: Blue Prism v6.8(Licensed)
Framework: .Net framework 4.7
Languages: XML, C#, Java
Database: Microsoft SQL Server
Other tools: Oracle VM Virtualbox, IBM SPSS v26

Activities carried under Blue Prism include:

- Creation of processes in process studio
- Creating various objects in the object studio
- Spying elements across the webpage using dynamic attributes
- Importing the input file and other necessary objects
- Resume Recover operation for exception handling
- Task queues implementation using Bot locking algorithm
- Queue management using key synchronization and flagging options

The Blue Prism version control helps us to keep a check of all the activities that are carried out during the course of implementation.

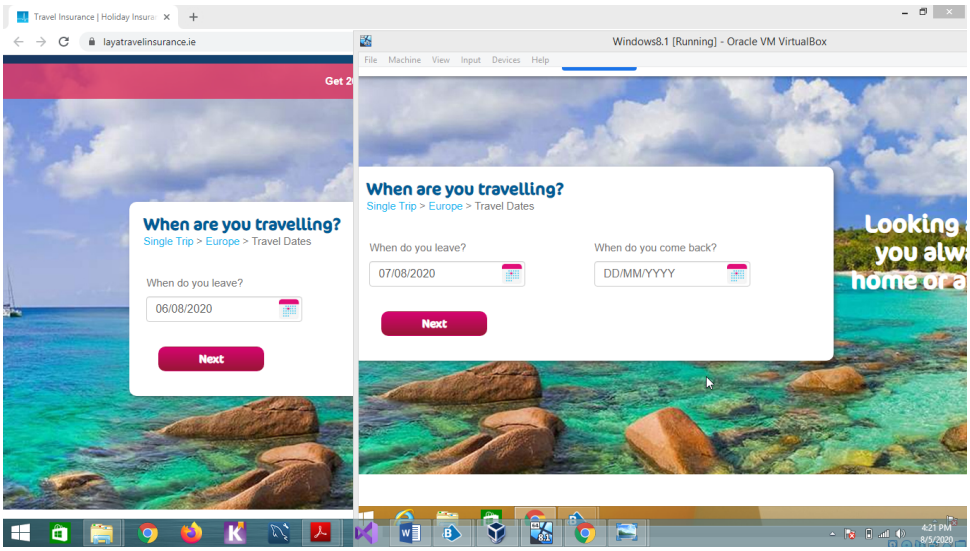

Figure 8: RPA Implementation on Insurance application

As shown in fig.11, multiple bots simultaneously perform operations on the Laya Insurance application with greater efficiency. The insurance values generated from the application are stored in the Excel sheet. If any of the bots fail in performing a particular task due to application inconsistency, it retries until it reaches the maximum attempts thereby improving scalability. As RPA itself is Robotic Automation, it is not prone to any errors caused by human intervention thus increasing the accuracy and productivity of the solution.

Thus the research question of whether RPA can be improved by using Container-related methodologies has been successfully answered but in an unlikely fashion. The resource allocation techniques of containers formed the basis on which Queue-based bot locking algorithm was designed with the help of virtualization thus offering higher scalability and better performance.

The secondary objective of solving a real-world problem arising due to the pandemic was also administered as this solution helps Insurance companies to solve the surging requests at a faster rate with increased productivity and less labour.

6 Evaluation

The new "Multi-Robot" architecture has improved the task execution speed by 40% as detailed in 6.1 and 6.2. Multiple bots have been distributed with different sets of a task thus increasing the performance immensely. The Blue Prism application provides us with an interactive dashboard which helps to understand the total automations and the ratio of objects and processes in the database. And essentially, it helps us to understand the status of queue volumes whether it is complete, deferred or still pending.

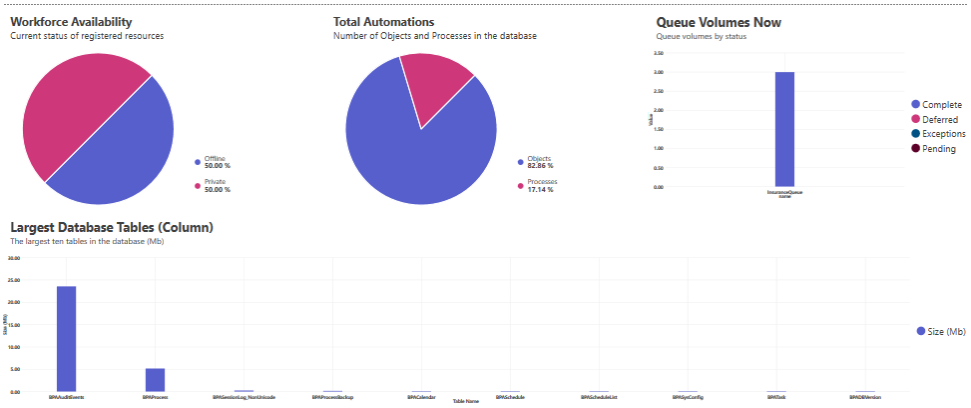

Figure 9: RPA Analytical Insights

6.1 Experiment 1

The task of repetitive sets of generating insurances from Laya Healthcare application were distributed among multiple resources as shown in fig. 10. 3 sets of the task were allocated among both the resources i.e. JAISON and HOSHEA. Each set of task is considered as 1 Queue item in the RPA terminology. As per the total work time displayed in the last column, average execution time taken for each item is around 30-35 sec.

Queue Contents Clear Filters Show Positions in Queue

Item Key	Priority	Status	Tags	Resource	Attempt	Created	Last Updated	Completed	Total Work Ti...
2	0			HOSHEA_debug	2	7/25/2020 11:26:14 P	7/25/2020 11:28:11 PM	7/25/2020 11:28:11 P	01:33
2	0		Exception: Autom	HOSHEA_debug	1	7/25/2020 11:26:14 P	7/25/2020 11:27:31 PM		00:58
3	0			JAISON_debug	1	7/25/2020 11:26:14 P	7/25/2020 11:27:23 PM	7/25/2020 11:27:23 P	00:34
1	0			JAISON_debug	1	7/25/2020 11:26:14 P	7/25/2020 11:26:48 PM	7/25/2020 11:26:48 P	00:32

Figure 10: Control Room Queue Results

It was also noted that due to application inconsistency when the second resource failed to execute the task in the first attempt, it automatically retried the queue item until it executes it successfully. Maximum such tries were configured to three times after which it would proceed with the next item in the queue.

Thus the overall execution time taken by both the resources(bots) to complete the task falls under 85 sec whereas the same task if executed by only one bot would have required 120 sec i.e. around 2 mins. It means here the efficiency is increased by 30% for odd combination of task sets and bots.

The overall execution time(T) for a single bot can be calculated as follows:

$$T = \sum_{n=0}^{N} Q_n * A$$

where Q: time reqd. for single Queue Item
A: no. of attempts
N: total no. of queue items

While for multiple bots, the execution time is given by T_{bots}

$$T_{bots} = \sum_{n=0}^{N} (Q_n * A)_{maxBot}$$

where maxBot: bot with the maximum time taken

As described in the second equation, here only the bot with the maximum time is taken into account, thus overlapping with all the time taken by the other bots thereby reducing execution time and increasing the performance. Hence, we can conclude that,

$$T_{bots} < T$$

6.2 Experiment 2

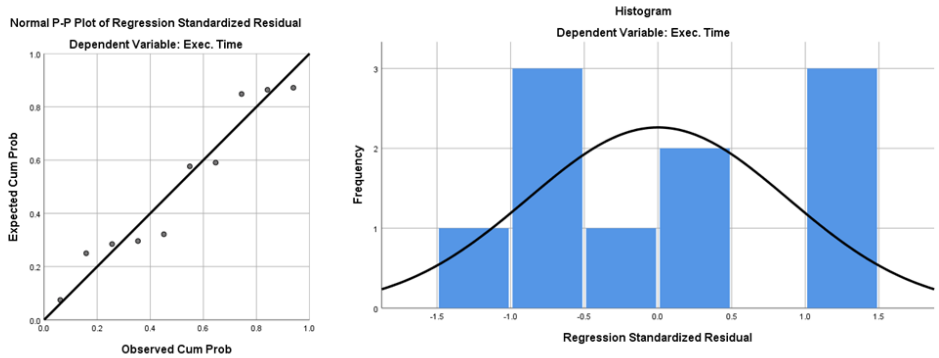

Figure 11: Multiple Regression - Histogram

Traditional RPA of single bot was evaluated against the Multi-Robot RPA using two and three bots respectively. A further analysis was conducted using Multiple regression which helped in understanding the normal distribution for execution time against both the number of sets and bots. The Normal P-P plot shows all the points are oscillating near the normal line and they are not much deviating from the line. It concludes that the histogram generated is fine and normal and no peculiar observations are reported.

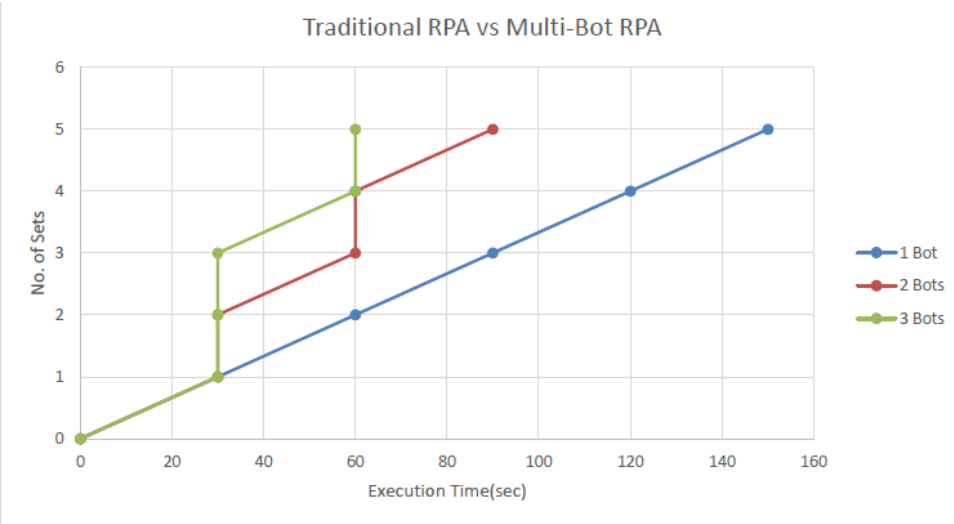

Figure 12: Traditional RPA vs Multi-Robot RPA

Here assumption is made that each bot takes an average of 30 sec to complete one set of task. The blue line in the fig.12 represents the single-bot RPA which takes the longest execution time as the number of sets increases. For instance, for 4 sets of execution, when the red and green lines(multi-bots) takes only 60 sec, the blue line takes 120 sec to complete the execution. Here the efficiency is increased by 50%.

Thus the higher the number of bots in the system, the lesser the total execution time and greater the scalability of the system. Because Multi-robot system helps in the provision of multiple bots thus distributing the task load amongst them thereby increasing scalability and if needed, more VMs(bots) can be added to handle excess load.

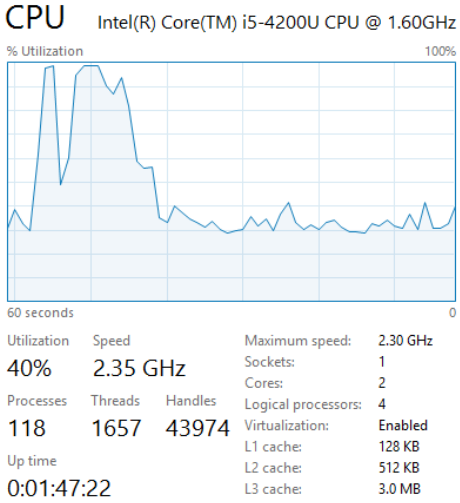

Figure 13: CPU Resource Utilization

. Also, the CPU Resource Utilization was closely monitored since Blue Prism is a Windows-based application. As the utilization stands at 40%, we can conclude that the RPA tool has good CPU utilization rate and it doesn't degrade the performance of the CPU for other applications as well.

6.3 Discussion

The various experiments conducted helps us to determine how the performance of the system is greatly improved by using "Multi-Robot" architecture. For odd combination of sets and bots, multiple bots helped in achieving around 30% efficiency while for the latter experiment which had even combinations, the efficiency achieved was around 50% i.e. giving a mean performance increase of 40%. Thus for larger tasks, the higher the number of bots in the system, the better is their overall performance.

However, more the number of bots means more licenses have to be acquired which increases the cost of the system since Blue Prism is a licensed tool. Thus though the performance of the system is improved greatly, their architecture deployment do comes with an increased cost.

7 Conclusion and Future Work

The research work has successfully implemented a new architecture in the field of Robotic Process Automation. The novel "Multi-Robot" architecture was realized in Soft Automation using Blue Prism as the RPA tool along with container-related methodologies. Though virtual machines were preferred over containers for virtualization, this research made use of containers' resource allocation techniques to devise a new algorithm called "Queue-based Bot Locking". This algorithm was a breakthrough in achieving better scalability and performance by a mean of whooping 40%. It was also successful in administering a real-time problem arising due to pandemic by helping Insurance companies to cope up with their surging requests with better performance at faster rate.

One of the hurdles faced in the research project was the acquisition of multiple licenses since Blue Prism is not an open-source tool. As a result, it was also noted that, more the number of licenses, more the number of bots in the system which could ultimately give a better performance but with an increased cost due to its licensing factor. However, if we consider the whole scenario, Robotic Process Automation is a much better choice because it eliminates the additional human workforce thus reducing the overall costs apart from its licensing costs.

Future work could include the combination of multiple RPA tools to realize the goal of better task execution with lesser cost. Blue Prism can be combined with open-source RPA tools like UiPath to execute a task where Blue Prism can be deployed for performing high-end processes while UiPath can be used to perform the lower ones. Thus it can reduce the number of bots require for Blue prism licensing thus reducing the overall cost of the system. Also, different mechanisms of locking algorithms could be explored in such a combined scenario.

References

Asatiani, A. and Penttinen, E. (2016). Turning robotic process automation into commercial success–case opuscapita, *Journal of Information Technology Teaching Cases* **6**(2): 67–74. JCR Impact Factor: 4.535.

Clair, C. L. (2018). The forrester wave™: Robotic process automation, q2 2018, *Technical report*. Number: 142662.

de Abranches, M. C. and Solis, P. (2016). An algorithm based on response time and traffic demands to scale containers on a cloud computing system, *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, IEEE, Cambridge, pp. 343–350. Core Ranking: A.

Duan, Y. and Yu, X. (2016). Multi-robot system based on cloud platform, *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, IEEE, Nanjing, pp. 614–617. Number: 7819651.

Dyumin, A., Puzikov, L., Rovnyagin, M., Urvanov, G. and Chugunkov, I. (2015). Cloud computing architectures for mobile robotics, *2015 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, IEEE, St. Petersburg, pp. 65–70. ISBN: 9781479973071.

Fung, H. P. (2014). Criteria, use cases and effects of information technology process automation (itpa), *Advances in Robotics & Automation* **3**. JCR Impact factor 3.573.

Hu, B., Wang, H., Zhang, P., Ding, B. and Che, H. (2017). Cloudroid: A cloud framework for transparent and qos-aware robotic computation outsourcing, *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, IEEE, Honolulu, pp. 114–121. CORE Ranking: B.

Inagaki, T., Ueda, Y. and Ohara, M. (2016). Container management as emerging workload for operating systems, *2016 IEEE International Symposium on Workload Characterization (IISWC)*, IEEE, pp. 1–10. JCR Impact Factor: 2.89.

Ingalls, A. (2020). Kofax reveals top 10 intelligent automation predictions for 2020, `https://www.businesswire.com/Kofax-Reveals-Top-10-Intelligent-Automation-Predictions/`. Accessed: 2020-03-25.

Issac, R., Muni, R. and Desai, K. (2018). Delineated analysis of robotic process automation tools, *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, IEEE, Bangalore, pp. 1–5.

Julio, R. E. and Bastos, G. S. (2015). Dynamic bandwidth management library for multi-robot systems, *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Hamburg, pp. 2585–2590. Core Ranking: A.

Kandan, R., Khalid, M. F., Ismail, B. I. and Hoe, O. H. (2019). Advanced resource allocation and service level monitoring for container orchestration platform, *2019 IEEE International Conference on Sensors and Nanotechnology*, IEEE, Penang, pp. 1–4. Core Ranking: B.

Kehoe, B., Patil, S., Abbeel, P. and Goldberg, K. (2015). A survey of research on cloud robotics and automation, *IEEE Transactions on automation science and engineering* **12**(2): 398–409. JCR Impact Factor: 3.667.

Kehoe, B., Warrier, D., Patil, S. and Goldberg, K. (2014). Cloud-based grasp analysis and planning for toleranced parts using parallelized monte carlo sampling, *IEEE Transactions on Automation Science and Engineering* **12**(2): 455–470. JCR Impact Factor: 3.667.

Lacity, M. and Willcocks, L. (2015). Paper 15/07 robotic process automation: The next transformation lever for shared services. URL: https://vdocuments.mx/paper-1507-robotic-process-automation-the-next-capabilities-derek-toone.html.

Leiva-Leon, D., Pérez-Quirós, G. and Rots, E. (2020). Real-time weakness of the global economy: A first assessment of the coronavirus crisis. ISBN: 978-92-899-4024-5.

Leshob, A., Bourgouin, A. and Renard, L. (2018). Towards a process analysis approach to adopt robotic process automation, *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, IEEE, Montreal, pp. 46–53. CORE Ranking: B.

Mavridis, I. and Karatza, H. (2019). Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing, *Future Generation Computer Systems* **94**: 674–696. JCR Impact Factor: 6.125.

Preeth, E., Mulerickal, F. J. P., Paul, B. and Sastri, Y. (2015). Evaluation of docker containers based on hardware utilization, *2015 International Conference on Control Communication & Computing India (ICCC)*, IEEE, Trivandrum, pp. 697–700. Core Ranking: C.

Sim, K. M. and An, B. (2009). Evolving best-response strategies for market-driven agents using aggregative fitness ga, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **39**(3): 284–298. JCR Impact Factor: 5.131.

Slaby, J. R. (2012). Robotic automation emerges as a threat to traditional low-cost outsourcing, *HfS Research Ltd* **1**(1): 3–3.

Wang, L., Liu, M. and Meng, M. Q.-H. (2013). An auction-based resource allocation strategy for joint-surveillance using networked multi-robot systems, *2013 IEEE International Conference on Information and Automation (ICIA)*, IEEE, Yinchuan, pp. 424–429. Core Ranking: C.

Wen, S., Ding, B., Wang, H., Hu, B., Liu, H. and Shi, P. (2016). Towards migrating resource-consuming robotic software packages to cloud, *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, IEEE, Angkor Wat, pp. 283–288. Core Ranking: B.

Zhang, D., Lynch, B. and Dechev, D. (2013). Fast and scalable queue-based resource allocation lock on shared-memory multiprocessors, *International Conference On Principles Of Distributed Systems*, OPODIS, Nice, pp. 266–280. Core Ranking: B.