Time Series Forecasting of Database Workloads in Hybrid Cloud

MSc Research Project
Cloud Computing

Richard McDonald
Student ID: 17123437

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Richard McDonald
Student ID:	17123437
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Mr. Vikas Sahni
Submission Due Date:	16/8/2021
Project Title:	Time Series Forecasting of Database Workloads in Hybrid Cloud
Word Count:	7884
Page Count:	22

Signature:	[illegible]
Date:	16th August 2021

Time Series Forecasting of Database Workloads in Hybrid Cloud

Richard McDonald
17123437

Abstract

The two biggest factors for the Enterprise in moving to a Public Cloud offering over existing offerings are control of cost and ensuring availability. The challenge is to ensure that both factors are managed wisely, even when both compete for attention at the expense of the other. In this work well-trusted statistical techniques were applied to aid that decision-making process. System logs are collected and maintained in a non-invasive manner and ARIMA and ES models are then applied to these logs to build a profile of a workload that identifies the sweet spot for configuration and satisfies cost control. In this work large Database workloads were created in a Private Cloud lab and then a simple method was used to extract and maintain the information before transforming it into a Time-Series Data-set. This work extends research in this field by keying in on Relational Database workloads, and by using non-invasive methods to collect the necessary information to complete the project. The results of the work shows a 96.9 percent level of accuracy in forecasting CPU activity as well as profiling workload that ensures optimum Quality of Service.

1 Introduction

1.1 Research Motivation and Objectives

The heuristic approach to problem solving in the IT area has been prevalent for a considerable time. Go-To people will assess difficult situations and apply a solution that is based on experience, which very often will have positive results. These positive results often lead to a confirmation bias that relies upon this style of decision making. The experienced engineer will diagnose issues by reviewing logs and make quick decisions to resolve. In fact, some research works propose greedy algorithms to ensure optimum performance as the only focus.

In the Virtualisation field contention issues are met with calls for more compute. In a Private Cloud configuration the penalty for unused compute is rarely penalised. In the Public Cloud arena this sort of decision making process has a financial penalty that has its own obvious repercussions and that in the Public Cloud arena, contention issues can be managed simply with an effective horizontal scaling strategy. Horizontal scaling does not apply to all types of workload though. For example, with Relational Databases great attention is needed when deciding configuration. Neither a greedy or a cost aware approach is preferable.

Other works have looked at profiling workload to fulfil both criteria with simulation tools that mimic CPU/Disk and Memory activity to report results. Some papers have used extraordinary ML techniques to derive simple conclusions. Other works use ML techniques to tune Database configuration parameters to ensure peak performance. In this work the main objective is to build an active database environment, automate a workload that thoroughly interrogates it and, can easily identify contention issues if and when they occur. The non-invasive output from that interrogation will lead to modelling an accurate configuration for that exact environment using a well known statistical technique for forecasting future workload and by extension, profiling that environment. For comparison sake another popular technique for modelling is used. Accuracy is measured and is the primary result to base the conclusion of the work. The secondary objective is the profiling of each environment, running that configuration and monitoring QOS metrics for that database environment.

1.2 Time-Series Forecasting methods.

Auto-Regressive Integrated Moving Average (ARIMA) is a modelling Tool-set used to forecast future patterns in time-series datasets using a merged approach of two traditional statistical models. Auto-Regression(AR) is the number of lags required in a time series to help predict next value and is represented by the letter 'p'. Moving Average (MA) is the number of lagged forecast errors required and is represented by letter 'q'. Differencing (I) is the process that makes the data stationary. That is what Differencing value to we need to apply to our model to make it stationary and it is represented by the letter 'd'.

For a Data-set to fit an ARIMA model it must be stationary, and for a data-set to be considered stationary it must follow the following rules.1. have a constant mean 2. Be of constant variance 3. Have no seasonal trends or major events in Data. Applying a weighted value to D allows non-stationary data be forecast using ARIMA. To forecast any Time-Series data using ARIMA, a value must be found for all three variables (p,d,q). This is known as the Box Jenkins method of forecasting.

Exponential Smoothing is another Time-Series forecasting method that is quite similar to ARIMA. In this work it is used as a comparative forecasting method to the ARIMA forecasting technique. We will report the results from each model at the end of the work for conclusion.

The layout of the work is as follows. In Section 2 we will complete an up-to-date literature review of the current works. Section 3 outlines the methodology applied to the work presented here. In Section 4 The main design decisions made to achieve the work are discussed. Section 5 discusses the path to implementing the proposed framework. Finally, Section 6 Presents a number of experiments testing the success of the decisions made throughout the implementation.

2 Related Work

The first work reviewed for this paper identify key issues with the adoption of Public Cloud Implementation. Alkhater et al. (2018) identifies the main obstacles to a Public Cloud Service model for the enterprise. In this study the Technological, Organisational, Environmental and Social factors are considered as the main challenges for lack of adoption of Public Cloud Infrastructure. Given the tricky nature of identifying how complex environments adopt into a public cloud model one quote rings out.. *"The results reveal* *that quality of service has a statistically significant impact on the decision to adopt cloud technology"*. Trust in the Public Cloud paradigm, security, privacy concerns and most importantly, ensuring strong QOS are the other major issues inhibiting migration.

To that end Cloud Service Providers (CSP) offer scaling solutions to improve performance. CSPs offer Rapid Elasticity as an essential characteristic in the Public Cloud Computing Model by Mell and Grance (2011). An extensive study of the state of Rapid Elasticity is completed in this work. Hwang et al. (2016) complete research on the state of scaling features provided by Various CSPs. They lay out the differences and strengths of the different scaling strategies. Through experiment, horizontal scaling is a far more effective solution than scaling up as a method of reducing SLA breaches. Several different experiments are completed to show the benefits of horizontal scaling. However, with certain workloads horizontal scaling isn't possible. Relational Database workloads are one such example, and will be discussed in a later section. In the next paper by Dinda (2006) an earlier example of use logging to predict resource utilisation. A Research Prediction System(RPS) is created through statistical sampling of applications and operating system. The authors found that performance can be predicted using an Auto-Regressive model. This led to further seek out these approaches in later sections.

2.1 Modelling Cloud Services

Please refer to Table 1. with listed works, methodology and outcomes.

Author	Method	Measure
Rahmanian et al. (2018)	LA-Ensemble	CPU
Wang et al. (2021)	Ensemble	Application
Guo et al. (2021)	Gradient Boost	Functionality
Shaw et al. (2020)	Ensemble/LR	Power/Efficiency
Zacarias et al. (2021)	Multiple	Application
Nawrocki et al. (2021)	MLP	VM Scaling
Laskawiec et al. (2021)	Decision Tree	Application Configuration
Van Aken et al. (2017)	Lasso/Linear Regression	Lower Latency/Faster Delivery
Han et al. (2013)	ARIMA	VM Number
Calheiros et al. (2015)	SARIMA	Horizontal Scaling

Table 1: List of important papers and methods deployed in research

In Zacarias et al. (2021) the authors propose a machine learning approach that uses logging and performance counters to tune the ideal performance of a VM. Pinpointing degradation when load reaches critical limits. The aim is to reduce makespan of an application job cycle. This degradation is modelled using Random Forest techniques. An added bonus is this process identifies those applications work in tandem with others. Using different model types to plot workload, namely Elastic NET, Support Vector Machine (SVM) and MLP modelling. In what reads as a complex approach with many different elements at play it offers a good framework for discussion.

in Nawrocki et al. (2021) the authors are motivated to use models to help control cost of resources. In their work they collect metrics from lab machines to build an Adaptive Resource Planning system, which in turn uses ML to pinpoint ideal configuration. The use a proprietary Deep Learning module called 'Weka'. This module after identifying the best fit sends information to the host servers to ring-fence resource for the ideal configuration. As an interesting segue Alkhater et al. (2018) also identifies cost as one of the major organisational barriers to public cloud adoption and adds value to the approach of the work completed in this paper.

In "A learning based ensemble resource usage prediction algorithm for cloud computing environment", Rahmanian et al. (2018) research is applied to generating relevant machine statistics and applying weighted ensemble machine models to find best CPU configuration for services. Historical and real-time metrics are used to best effect in calculating requirements. The Learning Automata feeds back information to the hypervisor to adapt configuration where also needed. One of the ensemble methods attempted in this work is the use of Auto-regressive modelling which we will see more of later.

Shaw et al. (2020) use an ensemble method to profile VM workload in a Public Cloud Arena. The motivation for their model is to ensure SLA compliance and drive power efficiency at the host level. Their Predictive Interference and Energy Aware (PIEA) algorithm achieves this by profiling the workload of each VM in the environment. Paying attention to the resources available from physical servers, their ensemble algorithm maintains the balance between resource consumption and over-committing resource. Like Rahmanian et al. (2018) this work receives data from the host server and interacts with it when it comes to identifying workload and grouping the workload together. The approach is to use an ensemble of Logistic Regression, ANN and SVM models both criteria for success are met, with over 30 percent saving in energy consumption and over 70 percent decrease in SLA violations.

Another approach to applying an ensemble based algorithm comes from Wang et al. (2021). In this research the authors offer their solution to profiling Complex Systems Architectures. Using Forward Search Feature Selection and Accuracy Based Error Pruning algorithms to achieve this. All necessary information is stored in a central repository and extracted to model CPU utilisation. Decision Tree, K Nearest Neighbour and Support Vector Machine models are used. By building a framework, removing unnecessary information and predicting the configuration they are able to improve prediction accuracy over regression-based approaches.

Application profiling features in the next work, where the authors looks at an approach to measuring application workload using the PARIS framework Guo et al. (2021). By building a HPC environment and extracting metrics in the lab to run against Machine Learning Models. Implementing a Machine Learning Model is a more accurate and less time-consuming process than the traditional method of using fault injection to predict workload requirements which can run for an extended time period. The authors are quoted as saying that "Application resilience is naturally a regression problem". The aim of the work is to reduce the amount of Silent Data Corruption in HPC clusters. An ensemble method is chosen with Gradient boosting, MLPR and CV the prominent models to test accuracy of their prediction mechanism but found Gradient Boosting as the most accurate model for their approach.

In Łaskawiec, Choraś, Kozik and Varadarajan (2021), the authors introduce Intelligent Operator, which is an automated decision support system applied to the placement of Cloud Services. Their study observes containerised public cloud edge and fog network applications. Using a third party and lab information they generate a knowledge base of configuration and performance metrics. The knowledge base is then applied to a Decision Tree Model to build an ideal configuration and also use it to identify any mis-configuration in their environment. Using Decision Tree model in this instance is practical because the authors are trying to ensure container configuration and performance information therefore something a little more complex is required.

A similar approach to the automation of Database environments is also considered in a paper by Van Aken et al. (2017). In this work the authors look at the most impactful way of using Machine Learning to tune a Cloud Hosted Database. The motivation in doing so is to speed up delivery of configuration and reduce complexity of tuning. The authors build their own tuning service called "Ottertune" to extract configuration and the necessary components that make up the 'knobs' in a Database Service. That repository then feeds and interacts with the Machine Learning models to build the training information, and eventually identify critical components that require tuning. Lasso is used as the preferred model for configuration selection to provide more accurate results over a linear regression method.

Finally, the final two works address Regression analysis to model Public Cloud services. In Han et al. (2013) use ARIMA (Auto-Regressive Integrated Moving Average) forecasting to identify workload on a Virtual host. By estimating the number of VMs that go live on a host or number of hosts. ARIMA is chosen because its well suited to forecasting Time-Series models but also because its well tuned to modelling workload that is not consistent. The authors refer to the VM workload with the phrase 'burstiness'.i.e the Moving average. The authors claim by monitoring these three parameters they can identify requirements for those hosts up to 60 time periods into the future. This is backed up in the work Calheiros et al. (2015) who look at profiling auto-scaling events in the Public Cloud. They preempt scaling events to ensure QOS than wait for a reactive scaling. The key components of the solution are the application provisioned to ensure the correct number of VMs have been commissioned, a load predictor vm that calculates the required number of VMs, and analyser vm that predicts future workload.Their use Seasonal ARIMA modelling techniques model the service.

2.2 Log Error Detection and Service Related Benchmarking Tools

In the first work by Tong et al. (2016) the authors devise a collection method of errors from an Openstack and Hadoop platforms. Using two proprietary algorithms to parse and categorise normal logs and extract only the pertinent errors from lab systems . A list of known bugs from online knowledge bases are also stored and compared to the local logs generated. These two components form an alert system once a bug is triggered. This is a somewhat elegant approach that cuts down on workload. KNN, Stochastic Gradient Decent and Bernouli NB algorithms are used in the process. This paper identifies a good use of supervised machine learning to identify potential errors in live systems and has a cloud agnostic approach in design.

One of the challenges of greater consumption of Cloud resources is that it leads to greater complexity in the environment. In the next paper by Yuan et al. (2019) the authors attempt to preempt failures in a virtualised environment using logs. They propose a lightweight error-logging detection system to guide troubleshooting and aid resolution. Similarly by creating a data repository of all of the known errors they can pipe all logs through their ML algorithm using a Word2vec NN. The purpose of this is to educate the machine learning environment to detect errors. Through experiment they review their data by running through various classifiers to see how to provide the best results for their approach. Their approach correctly identify over 96 percent of the errors generated in the environment.

The TPC Benchmarking Tool is used to simulate Enterprise type Database workloads. In the work by Wang et al. (2020)the authors research looks at implementing performance improvements and strong security to Database Systems in the Public Cloud. The TPC-H data-set [1] is used to drive that workload. The authors provide a framework for improving security and process improvements for managing Database Environments. They cannot measure the impact of their process without the ability to interrogate Data properly, and TPC-H provides them with the necessary tool to ensure availability and performance. Being able to identify and run queries against these databases is a critical success factor that some other benchmarking tools cannot offer.

In the paper "A performance study on on different data load methods in relational databases" by Martins et al. (2019) the authors use the TPC-H tool to simulate different methods of data loading for Relational Databases in a Cloud Environment. The process of loading and running the Benchmark are discussed and provides an excellent resource for the uninitiated. This is another paper that solidifies the use of proper database benchmarking when it comes to measure success of an approach.

The final paper from Jose and Abraham (2020) highlights the place for the Relational Database and its place within the organisation. In their experimentation the use a publicly available data-set and run some rudimentary select statement queries between a Relational Database and a Big Data Data-store using the MongoDB platform.[2]. The work compares queries and tests against each Environment. The queries simply serve the purpose of calculating sorting and return response from the Environments. Whilst the aim is to benchmark MySql and MongoDB the use of proper Data-sets with results does allow for a clear presentation to the work.

2.3 Other approaches to improve VM performance

In this final section a number of papers that address performance improvements in virtualised platforms are reviewed. The first is from Mashhadi Moghaddam et al. (2020). The topic of power consumption in the Cloud Data Center is the focus of attention. Using an algorithm to identify over-utilised hosts, the VMs that cause the issue and the appropriate host to migrate to. Added to that some regression analysis, the work was able to reduce power consumption by over 26 percent and by using load balancing reduce SLA violations by half. This is another example of using regression analysis to solve practical computing problems.

In Kim et al. (2021), applies another approach to ensure that SLA in a cost controlled manner. In it they apply a similar approach to identify high CPU activity but factor disk operations as a factor that drives high utilisation. Whilst the authors don't elucidate this, there is a correlation between high IO and CPU usage. An algorithm is built to stabilise CPU consumption across all hosts. Their solution is based min/max evaluation. Returning to the issue of disk usage as a driver of CPU cycles, it is worth further inspection of the work to see how they achieved the goals in a shared storage environment. Its hard to see how migrating cpu and memory aids disk contention issues.

Witanto et al. (2018) also looks at the problem of consolidating VMs in hosts to reduce power consumption Using NN models to identify and classify VM workloads. Just like the previous works maintaining an acceptable QOS is of foremost concern to

[1]TPC-H Benchmarking Tool: `http://tpc.org/tpch/`

[2]MongoDB: `https://www.mongodb.com/`

the work. Using the Deeplearning4j library a training data-set is built and workload is classified. Workload is estimated and recommendations for load splitting is made to improve power supply and performance. Monitoring SLA violations with VM migrations the model eventually leads to the improvement of of SLA breaches with savings to power consumption.

In Balaji et al. (2014) the environment workload is profiled using Holt-Winter and ARIMA models. Using a public data-set the authors model the workload against the one another. The outcome of the modelling work being the ability to preempt auto-scaling events of the web environments. Using Holt Warner as a time-series model the authors identified this technique as having the greatest success. Results from the Data-set were able to identify the use of time series modelling as an acceptable approach but also identify the HW method as the most accurate.

Prasad et al. (2018) propose the 'first' attempt at profiling ideal configuration parameters for Cloud Services. In contrast to the greedy approach to building services they propose a solution to the 'Virtual Configuration Problem'. That being the correct configuration at best price. Using two queuing theory algorithms which calculates the optimal instance creation time based on the SLA and cost constraints. To experiment with their algorithms they create instances of Ec2 instances across all tiers of the AWS environment. these instances are deliberately given a once size fits all instance profile. Their first algorithm, it is able to successfully find the strongly performing configuration for that instance. Their second provides very strong results and is able to tune services for optimal performance. The authors are able to experiment and find significant improvements using their model over standard methods, showing again that use of available data can bring about positive performance changes.

In Qiu et al. (2016) work addresses the problem of Cloud capacity by establishing a baseline future configuration for all VM workload in the Cloud Data Center. The approach of this paper is to build a Deep Learning Algorithm to predict workload coming into the virtual machine with a Regression Model to predict the resources required for that workload. The Deep Learning Algorithm uses a Deep Belief Network model accurately predicts the future workload taking the Mean Average Percentage Error readings from model to identify accuracy. Again we see ARIMA being used to help identify resource configurations in this work, and do so with confidence.

The final paper uses three different models to predict ideal VM configuration. Logeswaran et al. (2016) focus on benchmarking performance problems with applications when physical resources (Memory,CPU and I/O) reach points of contention. By identifying the minimum conditions in which an application runs on the server the Application Performance model is aligned. A cost model is defined by service at acceptable QOS, but also calculates the cost of running the application cheaply (i.e what penalties would apply for running the service at a lower QOS) The aim of this model is to find the best possible configuration that meets SLA but at the cheapest configuration. Finally, a reconfiguration algorithm that decides if a horizontal scaling execution task needs to be completed is executed. By understanding if the SLA is still being achieved the algorithm holds off on an execution task. The authors understand and show that in order for a service to function properly, maintaining and managing platform integrity is key and offer a way to tune it, but do try to push configuration to its limit to gain acceptable results.

3 Methodology

To generate the data needed, a lab environment was built running four active Virtual machines running an active Relational Database Environment. This section discusses the end-to-end methodology that is relevant to this work.

3.1 Data Generation and Gathering

Each VM running in the lab environment will run different Database workloads and automated queries to mimic Enterprise level workload. The first step in this task is to provide a host server running a robust Hypervisor. the best choice of lab hypervisor for this project is VMWare Esxi.[3]. The host server can be run in standalone mode or as part of a cluster with the ability to store up to 30 days worth of metrics from the Virtual Machines running on it. For the collection of data standalone mode was chosen as the most convenient method to collect the data. the downside to this decision was that the metrics for each VM would only be stored in resident memory for up to 60 minutes and needed to be collected regularly, in this case by script. Using the native VMware PowerCli scripting tool to extract and write metrics to comma delimited sequential files.

The statistics generation process is described here by vendor SME Drummonds (Cited June 2021a). Effectively the host accumulates resource utilisation over a 20 second period and reports at the end of that time sequence. The most important counters from CPU and Memory are calculated and presented with maximum, minimum and average results. Five minute collection leads to 288 files per day. We see similar approaches to data extraction in the following papers in the literature review Nawrocki et al. (2021); Shaw et al. (2020). This process is managed by a dedicated task scheduled in the lab. The full list of counters extracted is available in the configuration manual. However, the counters that most concern the project are show in Table.2. This reference material from the vendor lists the counters available for collection and an explanation for each counter Drummonds (Cited June 2021b).

Measure	Counter	Measure	Comment
CPU	cpu.usagemhz.maximum	Megahertz	Max CPU consumed by VM
Memory	mem.usage.maximum	Percent	Max Memory consumed by VM

Table 2: Metrics used in experiment process.

3.2 VM Database Workload Generation

The TPC-H database benchmarking tool was created by the Transaction Processing Performance Council [4] was used on all 4 VMs to to simulate Enterprise level Database activity. The Database consists of eight different tables with distinct Primary and Foreign key relationships. Also included are= 22 standard queries that can be run to generate Database activity. As per the literature review this Benchmarking tool used to simulate

[3]VMware: `https://www.vmware.com/content/vmware/vmware-published-sites/us/products/vsphere-hypervisor.html.html`

[4]TPC-H Benchmark: `https://http://tpc.org/tpch/default5.asp/`

this specific type of workload and can be reviewed here. Wang et al. (2020); Martins et al. (2019).

Each VM running in the lab runs windows 2012 Server with MySql(Version 8.0) as the Database. Each Server has an Infrastructure and Database Monitoring tool by Solarwinds [5] and was used to ensure operational stability and monitor QOS to see if configuration changes were too severe.

3.3 Log Collection and Database Ingestion

Metrics taken from the host server are stored in individual comma delimited files in five minute sequences. Over time these log files are fed into our Database using a simple python script. It should be noted that the process of collecting data is somewhat complicated by the need to run one VM on a secondary host server. This leads to a second set of log files that need to be maintained. For the sake of completeness the Database is called 'VMT' and single table called 'VMTDATA'. Using this central repository a simple query can extract VM relevant data for analysis later in the project.

3.4 Time-Series Model Forecasting

The main research goal of this project is to identify and test a statistical model that can successfully predict VM configuration using the metrics collected over a period time. A secondary research goal for this project to to test the results of those models against the VMs in the lab focusing on QOS of those VMs. In Related Works several researchers have used ARIMA as a forecasting tool for profiling different types of cloud services Rahmanian et al. (2018); Calheiros et al. (2015); Han et al. (2013) with positive results. This has informed the decision to use this model.

Using the 'Forecast'[6] and 'TSeries' [7] packages in RStudio the data extracted by query is turned into a Time-Series Data-Frame for analysis and forecasting. Mindful that a number of the papers use several ensemble methods to predict workload and that the approaches in these papers were working with a number of different variables. Decision Trees, LSTM and NN were mentioned a number of times but in those works the approach to forecasting was using a multivariate approach. The approach here is to test resources in a univariate fashion. Two of the best approaches to Univariate are ARIMA and Exponential smoothing.

The overall approach to modelling is as follows. once the data has passed into the analysis phase a number of tests are to be conducted. 1. To ensure the worthiness of testing the data and 2. to find the most accurate models for this approach. Time is spent on a series of tests to find the best values for the ARIMA (p,d,q) Box Jenkins model, including plotting of data, running Augmented Dickey Fuller test, test the graphs in Decompose model, and reference the ACF and PACF charts to check relationships in time series and lags in Time-Series. Once this information has been processed two different tests are performed to identify the best models for forecasting. A number of models are manually checked for suitability using the Lowest BIC score. A secondary 'brute force' approach using the auto.arima command identifies its best fit model. From these tests the best three to four models are identified, with tests run against the residual data and

[5]Appoptics by Solarwinds: `https://www.appoptics.com/`

[6]Forecast in R: `https://cran.r-project.org/web/packages/forecast/forecast.pdf`

[7]TSeries in R: `https://https://cran.r-project.org/web/packages/tseries/index.html/`

a final decision on fit being made based on lowest RMSE score, but also considering the information derived previously. The Mean Average percentage score is then presented as the final result confirming the accuracy of that model. For comparative purposes, Each Time-Series Data-set will be tested against a Simple Seasonal Exponential Smoothing model. The RMSE and MAPE values for each model will be compared and presented in the results section.

4 Design Specification

In this section the overall architecture, configuration and design choices are discussed as well as some of the insights that make up the process of collection management and preparation for review.

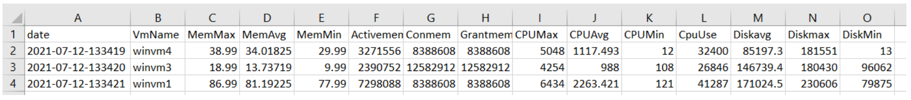
4.1 VM and MySql Environments

Each VM running in the lab environment has different Virtual hardware and database configurations. Database sizes range from 2GB to 10GB. For a fuller explanation and setup of each VM please consult the relevant section of the configuration manual.

Similarly, the TPC-H Data-set is compiled with two inbuilt binaries called 'DBGen' and 'QGen'. All data and key relationship's are configured using the Navicat for Mysql utility.[8]. Database and VM Monitoring is managed through Solarwinds Appoptics Monitoring. For a fuller description of steps please consult the configuration manual.

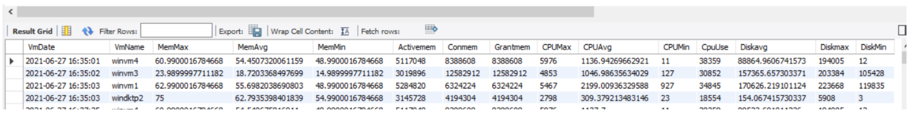
4.2 Collection and Database Configuration

Once all queries are loaded and functioning the process of collecting data begins. using the PowerCli scripting language. The extraction script is setup as a scheduled task to write time stamped files for later use. The script creates a comma delimited file and populates the file with the extracted information. An example of one of these files can be seen in Figure. 1. At the end of the collection period the accumulated files are then fed into a centralised MySql table 'VMTDATA' using a Python script. This script uses several libraries to convert the raw files into Database format. 'Psycopg','mysql.connector' and 'csv' libraries allow for the transfer the contents of a single directory into the Database. /each row is time-stamped using the RDBMS standard. YMD H:MS Figure. 2 contains a screenshot of the contents of the VMTDATA table.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	date	VmName	MemMax	MemAvg	MemMin	Activemen	Conmem	Grantmem	CPUMax	CPUAvg	CPUMin	CpuUse	Diskavg	Diskmax	DiskMin
2	2021-07-12-133419	winvm4	38.99	34.01825	29.99	3271556	8388608	8388608	5048	1117.493	12	32400	85197.3	181551	13
3	2021-07-12-133420	winvm3	18.99	13.73719	9.99	2390752	12582912	12582912	4254	988	108	26846	146739.4	180430	96062
4	2021-07-12-133421	winvm1	86.99	81.19225	77.99	7298088	8388608	8388608	6434	2263.421	121	41287	171024.5	230606	79875

Figure 1: .csv formatted output



	VmDate	VmName	MemMax	MemAvg	MemMin	Activemem	Conmem	Grantmem	CPUMax	CPUAvg	CPUMin	CpuUse	Diskavg	Diskmax	DiskMin
▶	2021-06-27 16:35:01	winvm4	60.9900016784668	54.4507320061159	48.9900016784668	5117048	8388608	8388608	5976	1136.94269662921	11	38359	88864.9606741573	194005	12
	2021-06-27 16:35:02	winvm3	23.9899997711182	18.7203368497699	14.9899997711182	3019896	12582912	12582912	4853	1046.98635634029	127	30852	157365.657303371	203384	105428
	2021-06-27 16:35:03	winvm1	62.9900016784668	55.6982038690803	48.9900016784668	5284820	6324224	6324224	5467	2199.00936329588	927	34845	170626.219101124	223668	119835
	2021-06-27 16:35:03	windktp2	75	62.7935398401839	54.9900016784668	3145728	4194304	4194304	2798	309.379213483146	23	18554	154.067415730337	5908	3

Figure 2: MySql Database view

[8]Navicat for MySql: https://navicat.com/en/products/navicat-for-mysql

Figure 3 presents a data-flow diagram of the processing of data extracted from the host server. In step 1. the activity is created by Database queries running on each DB. The hypervisor stores and calculates the utilisation information. That information is then extracted by script in Process 2. That data is stored in unstructured format in a data repository. Then, the Python script populates that data into a structured format in preparation for the data analysis phase of the work. A number of targeted queries are run to extract VM specific reports including Date-Time and individual metric information.

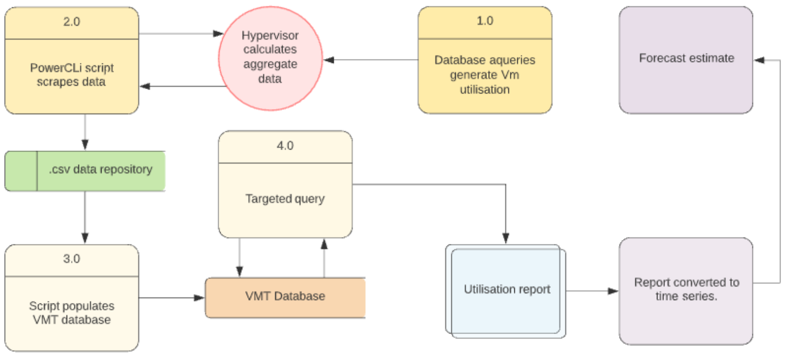

2.0
PowerCLi script scrapes data
Hypervisor calculates aggregate data
1.0
Database aqueries generate Vm utilisation
Forecast estimate
.csv data repository
4.0
Targeted query
3.0
Script populates VMT database
VMT Database
Utilisation report
Report converted to time series.

Figure 3: VMT Data flow

Another comma delimited file is presented to be transferred for analysis in the next phase of work.

4.3 Time series Forecasting Project Decisions

In the next section the main features of data analysis are discussed. There is however, one aspect of Data collection that has not been discussed. The original Data collection method allowed for the creation of 288 data points per day. This caused difficulty in the Time series creation process in RStudio. The 'TSeries' library allowed for the ingestion of data many formats, 'Month-Day' or 'Day-Hour' for example. But for the Data collected in this work there needed to be a 'YYYY-MM-DD HH:MM:SS' format. This proved to be a task that was beyond completion. Several attempts to convert data using the 'PosixCT' function to load the data as Date/Time format led to memory problems turning this Data during the conversion to Time-Series for analysis. Ultimately the decision was made to only plot the data using the first reading per hour, giving a total of 24 observations per day. This allowed the data-set to be easily converted to a Time-Series.

Through experiment, each forecast object will have a number of candidate ARIMA models to find best fit. The final section of the report will review each option with one final model that can be run against each object. Using the 'ses' function for comparisons sake each ARIMA model will be compared to Simple Seasonal Exponential Smoothing model. This comparison allows the researcher to assess the overall success of using ARIMA for this type of workload but also give an idea how a much easier model approach would work.

5 Implementation

The task of bringing the Decision Support process to fruition begins with the process of extracting the reports from the Database and feeding into the individual tool-sets. There will be 8 different models that apply to Memory and CPU utilisation on each VM. Each utilisation report is extracted from the Database returning the timestamps and utilisation information for each resource between dates 'x' and 'y'.

RMarkdown was the chosen ecosystem to generate the Forecast reports for each test. At thie point the Data was loaded for conversion to a Time-Series format. Each .csv file was loaded using into the RStudio environment following the 'VMName'+'Metric' naming convention. The column storing utilisation usage is formatted as an integer.

5.1 Converting Data to Time-Series and Initial Analysis.

The Data frame is now transferred into a Time-Series and given a memorable variable name.In this case we follow the naming convention 'VMName'+'Metric'+'ts'.This variable name will be used later to plot charts and predictive models. A summary statement of the data-set is then provided including mean and median points. The first plot of data is then performed. This initial plot will give a visual aid to see if the data-set is stationary.

5.2 Plot/Decompose and Check Stationarity

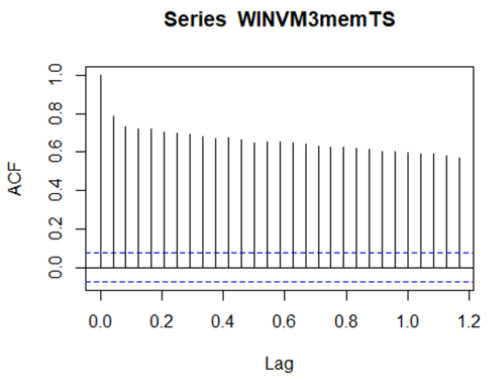
The process of finding the best fitting model for the Data-set begins. The first task is to check for stationarity of data. An initial plot of the data should give a visual aid to identify if data is or not stationary. Using the adf.test (Augmented Dickey Fuller test) function which is part of the tseries library a test is performed. The null hypothesis is that data is not stationary. A p value below 0.05 confirms the data is stationary. The next steps of the process is to show a decompose plot of the data. This returns a single graph with observed plot, trend in data, checks for seasonality and randomness.

5.3 Finding values for P and Q. The ACF and PACF graphs.

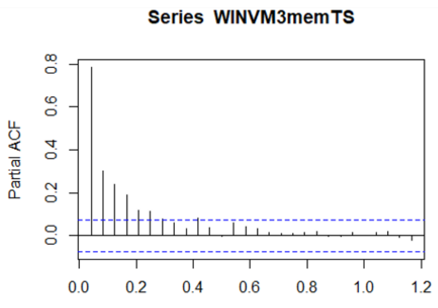
Auto-correlation checks the relationship between a value and its past values. These graphs return information regarding the relationship between the point-in-time data and its relationship to previous or lagged values. The purpose of the ACF and PACF graphs is to help identify the P(PACF) and Q(ACF) values for Box Jenkins method testing. Given that the model order is key to find the best fitting model for the forecast identifying trends and lag values in the ACF and PACF charts is an important step in the ARIMA process. Ruling out trends in the data to ensure there is no white noise or obvious trend is key. The PACF model helps identify the exact number of lags that best fits the ARIMA model, giving an estimate figure for 'p'. Examples of plots for ACF in Fig.4a and PACF in Fig.4b

5.4 Finding Best Model Fit.

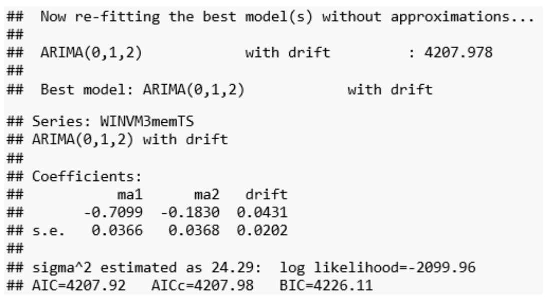
There are two methods of finding the best model for forecast. The first method is to completely create a number of Box Jenkins tests manually using the 'arima' function. By force coding a number of ARIMA models one can get an idea of the BIC scores to identify best



(a) acf plot example



(b) PACF plot example.


```
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,1,2)           with drift         : 4207.978
##
##  Best model: ARIMA(0,1,2)           with drift

## Series: WINVM3memTS
## ARIMA(0,1,2) with drift
##
## Coefficients:
##           ma1      ma2   drift
##       -0.7099  -0.1830  0.0431
## s.e.   0.0366   0.0368  0.0202
##
## sigma^2 estimated as 24.29:  log likelihood=-2099.96
## AIC=4207.92   AICc=4207.98   BIC=4226.11
```

(c) auto.arima example.

Figure 4: ACF/PACF and auto.arima output examples from experiment.

fitting models. The format of the code to run this is "arima(timeseries,order=c(p,d,q))" where p,d and q represent numbers to fit each value.

The second and much quicker option is to run the auto.arima function. This function force tests a number of arima models against the data and returns what it believes to be the best ARIMA model to complete the forecast. Once all tests are completed the three best candidates are sent for residual testing and forecast modelling. An example of the output is provided in Fig 4c.

5.5 Residual Testing and Forecast Modelling

This section of the report allows the researcher to test the models for best fit. Usually three candidate models are proposed including the 'auto.arima' model. The tests that are completed in this phase are the return a summary of the model, plotting the forecast for 4 days, complete a 'checkresiduals' test and return a BIC value for the model. Summary information allows to asses best candidate model by identifying the model with lowest RMSE. The 'checkresiduals' test returns a timeline of Data, a distribution curve including a plotted normal distribution, and an ACF plot with differencing enabled. The forecast plot signifies the required utilisation parameter and can be used as a guideline for ideal configuration.

5.6 Exponential Smoothing

The final model added to each object test is a Simple Exponential Smoothing forecast, with seasonality enabled. Each 'ses' plot will give a 48 observation forecast at a 95 percent confidence. The 'ses' function is included in the 'forecast' library.

6 Evaluation

6.1 Experiment / Case Study 1

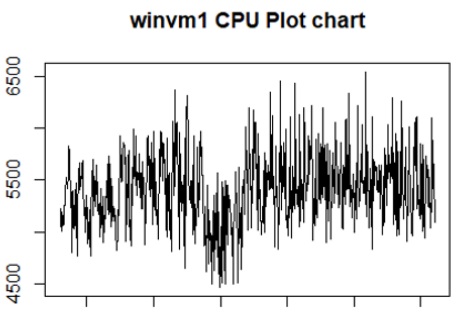
VM - Metric	WINVM11 - Max CPU	WINVM1 - Max Mem
Date Observed	June 25- July 23	June 25- July 23
Number of Events	674	674
Model chosen	(1,1,2)(0,0,2)	(1,2,2)
ARIMA RMSE	319.68	1.670533
ARIMA MAPE	4.669804	1.67213
SES RMSE	332.136	1.702341
SES MAPE	4.88634	1.656314

Table 3: Significant Results from Experiment 1 - winvm1

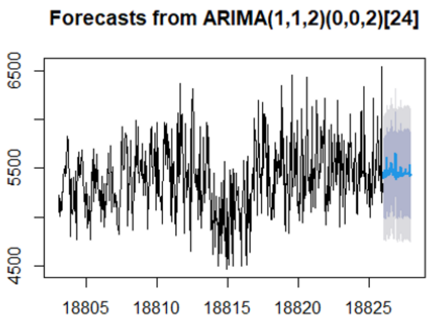
6.1.1 CPU Model Process WINVM1

The overall plot of of the CPU activity is presented in Figure 5a. The adf test returns a P value of 0.01 indicating stationarity of data. The ACF and PACF tests show non correlation. The final candidate models are ARIMA (1,1,2) and ARIMA (2,1,1)(0,0,2) based on lowest BIC score. Whilst the auto.arima function identifies ARIMA (1,1,2)(0,0,2) as the model of best fit.

Using the output for the residual testing and lowest RMSE the seasonal ARIMA model (1,1,2)(0,0,2) was chosen as best model. RMSE is calculated at 319.1408 with an MAPE of 4.643025. The forecast for this model is visually represented in Fig5b. Using this Seasonal Exponential Smoothing model, the RMSE(332.136) and MAPE (4.88634) values were added to the table. The exponential smoothing model is a little less of a fit for this workload. The results of all tests are listed in Table 3.



(a) winvm1 CPU Plot

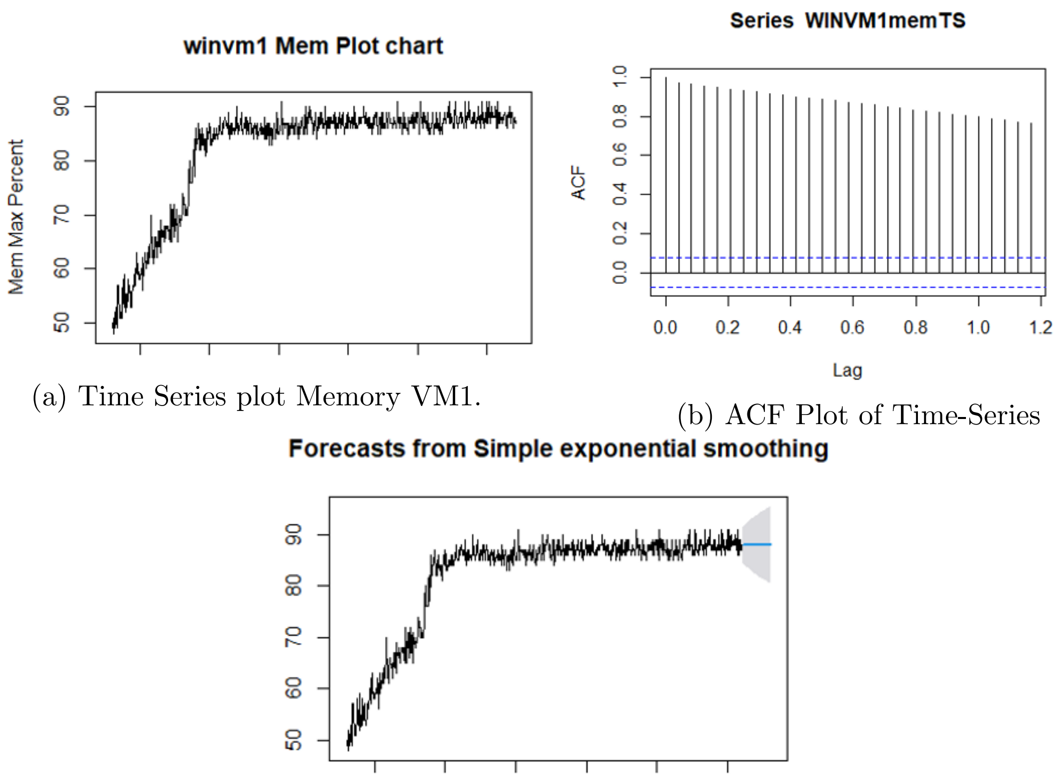


(b) winvm1 ARIMA Forecast Plot

Figure 5: Significant Graphs ARIMA model winvm1 CPU Forecasting

6.1.2 Memory Forecasting Process

From the Time-Series Plot in Figure 6a, memory utilisation grows at a steady pace and plateaus after a certain period of time. This graph shows us what we know to be true. That is that memory utilisation in stable Database environments tends to remain constant. There are other plots within the report we can see that can back this hypothesis up. The ACF plot shows a strong correlation between the lags in the series. This correlation could be considered 'white noise' as you can see in Figure 6b. The Ljung-Box test suggests data could be white-noise. All other results are listed in Table 3. For comparative purposes the Exponential Smoothing Model performed slightly better. It can see from the results that the model was extremely accurate but can also be explained by the predictability of usage. Further analysis of the auto.arima process shows a recommendation to run select ARIMA (0,1,1)(2,0,0) with drift had a lower Mean Average Percentage error but higher RMSE value. Residuals looked pretty similar.



(a) Time Series plot Memory VM1.

(b) ACF Plot of Time-Series

(c) Memory Forecast Exponential Smoothing

Figure 6: Significant Plots and Charts for Memory forecast on VM1.

6.1.3 Resource Savings

Based on the forecasts of both Memory and CPU no changes should be made to this configuration. This VM is currently running an optimum configuration.

6.2 Experiment / Case Study 2

6.2.1 CPU Experiment Process

In the second experiment attention is turned to the largest VM, which runs a 10GB Database and executes 150 queries per hour. Highlights are as follows. The Augmented Dickey Fuller test rejects the Null Hypothesis of non stationarity. Both ACF and PACF plots shows random effect in each plot. These plots are made available in Fig.7a. This

VM - metric	WINVM2 - Max CPU	WINVM2 - Max Mem
Date Observed	July 10-Aug 1.	July10-Aug 1.
Number of Events	540	540
Model chosen	(1,1,2) (0,0,2)	(1,2,2)
ARIMA RMSE	309.9947	1.910114
ARIMA MAPE	3.515186	3.575327
SES RMSE	329.7779	1.970669
SES MAPE	3.856783	3.60273

Table 4: Significant Results from Experiment 2 - WINVM2

should lead to a model with low 'p' and 'q' values. It was surprising to see ARIMA (4,1,4) return the lowest RMSE score. The residuals for this model also show random data as can be seen in Fig.7c.

ARIMA (1,1,2) (0,0,2) was the model selected by auto.arima and was the model selected as best fit for this model, even with slightly higher RMSE and MAPE, because the ACF/PACF plots are in disagreement with the selection of a (4,1,4) model. All results from this experiment are available in Table.4.

6.2.2 Memory Experiment Process

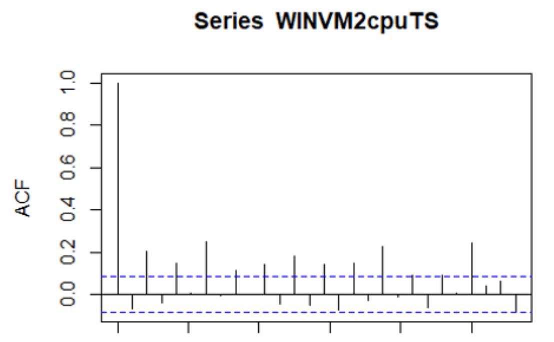
When reviewing the report for Memory Utilisation on this VM the first thing of note is the Memory Utilisation report. It is clear to see from the chart that the object is not stationary.8a. This analysis is confirmed by the results of the Augmented Dickey Fuller Test (p-value 0.174). The decomposition chart(see Fig.8b) which shows trend and seasonality, and the ACF plot showing high correlation between lag values. The three models chosen for further review were ARIMA (1,2,2), ARIMA (1,1,3) and ARIMA(0,1,2). All three models failed the Ljung-Box test for white noise, however Model (1,2,2) recorded the lowest RMSE value (1.910114). All results from this experiment are available in Table4.

6.2.3 Resource Savings

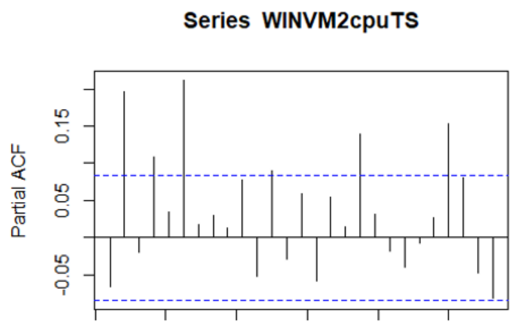
A review of forecast models presented. No changes should be made to CPU configuration as the forecast shows a trend of over 50 percent utilisation. we can see that Memory configuration can drop from 24GB to 16GB. Memory was reconfigured on VM on 9-Aug. Using the monitoring tool for MySql queries have not been affected by the change in memory. In Fig.9b the appoptics monitoring tool shows no reduction in slow queries as part of the hardware configuration change, Fig9c shows no change in mysql performance over a 4 week period, and Fig9a shows the change in machine configuration observed.

6.3 Experiment / Case Study 3

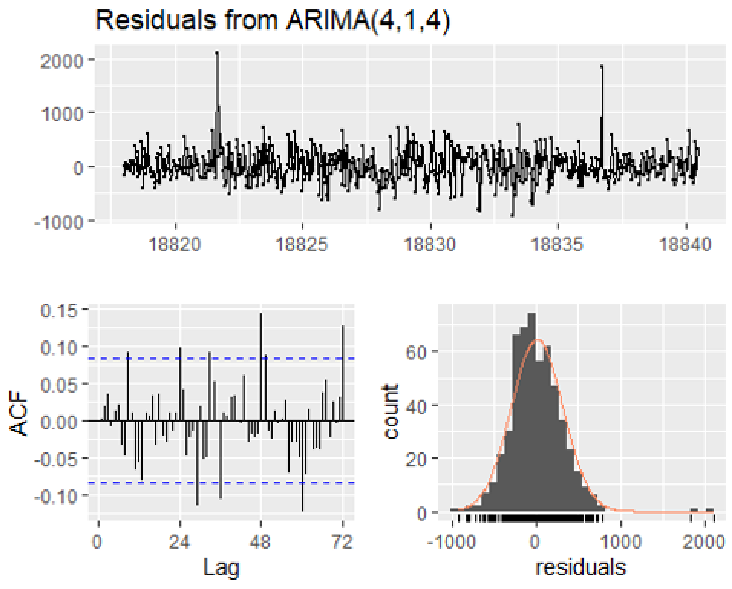
The final case study will concentrate on the reports of CPU activity on the remaining two VMs in the lab environment. All results pertaining to memory activity is available in Table.6



(a) ACF plot showing random collection of data



(b) PACF showing random lag in data.



(c) Plot of Residuals WINVM22 CPU ARIMA (4,1,4

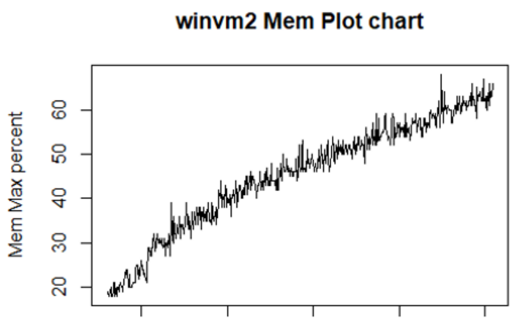
Figure 7: Significant graphs ARIMA Forecast WINVM2 CPU.

VM - metric	WINVM3 - Max CPU	WINVM4 - Max CPU
Date Observed	July 16- August 7	July 16 - August 7
Number of Events	553	553
Model chosen	(2,0,2)(2,0,0) non zero mean	(0,1,1)(0,0,2)
ARIMA RMSE	92.88816	234.6027
ARIMA MAPE	1.463682	2.750899
SES RMSE	93.87037	239.7934
SES MAPE	1.482439	2.813565

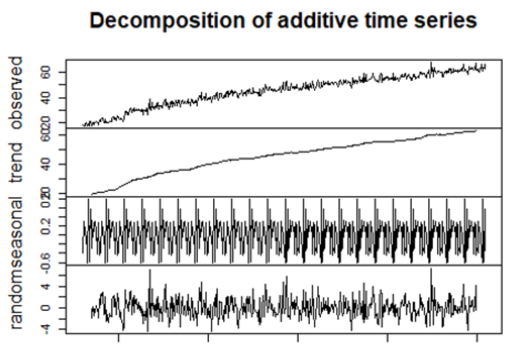
Table 5: Significant Results from Experiment 3 - CPU activity WINVM3 and WINVM4

6.3.1 WINVM3 and WINVM4 CPU Analysis

All results for this analysis are available on Table 5. Initial analysis of report shows the time-series data accepts the null hypothesis of non-stationarity with a p-value of 0.08111. The auto.arima process recommended a best model fit of ARIMA (2,0,2)(2,0,0) with a non zero mean (RMSE 92.88816). Three other candidate models were selected from

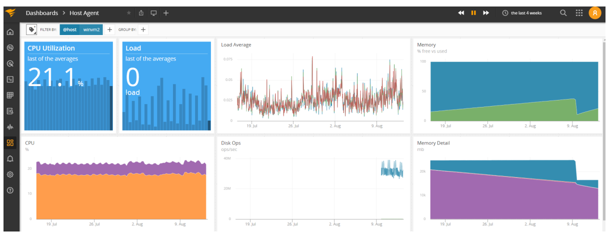


(a) Data plot winvm2 memory

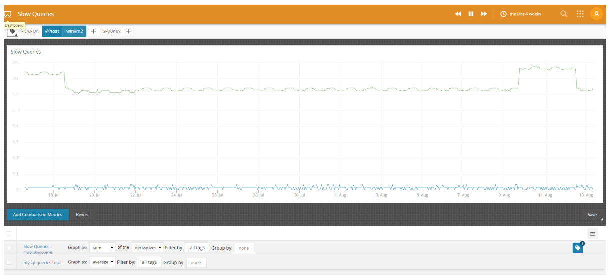


(b) Decomposition Chart winvm2 Memory

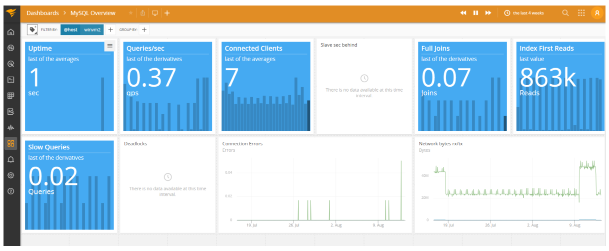
Figure 8: Significant charts WINVM2 Memory Report.



(a) VM Dashboard change in Memory Configuration.



(b) Chart showing no Degradation in Query Response



(c) Chart showing no Degradation in MySql Performance

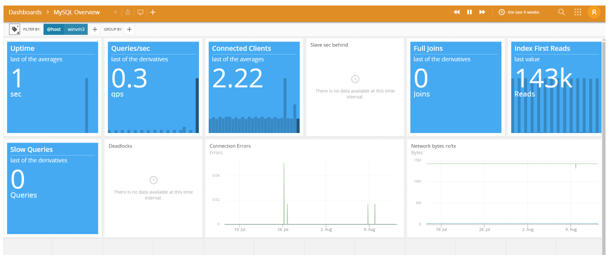
Figure 9: configuration changes WINVM2.

VM - metric	WINVM3 - Max Mem	WINVM4 - Max Mem
Date Observed	July 16- Aug 7	July 16- August 7
Number of Events	553	553
Model chosen	(1,2,2)	(0,1,1) with drift
ARIMA RMSE	1.870671	1.51407
ARIMA MAPE	2.979884	1.43974
SES RMSE	1.926707	1.520805
SES MAPE	3.104156	1.437622

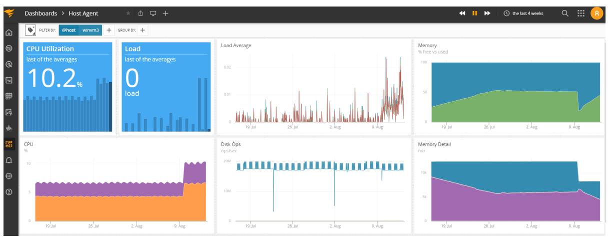
Table 6: Memory findings WINVM3 AND WINVM4

lowest BIC value. ARIMA(1,1,2), ARIMA(1,2,2) and ARIMA (0,1,2). The auto.arima recommended model had lowest RMSE of all candidate models. The normal distribution plot showed a concentration of data at the mean, and Ljung-Box Test showed a high p-value (0.4195). when reviewing the MAPE values, ARIMA and exponential smoothing do a good job of predicting utilisation.

Based on the modelling results, hardware configuration for winvm3 allowed for a reduction of 2 CPUs and 4GB of RAM with no effect on QOS. See Fig 10.



(a) Dashboard Mysql activity and errors.



(b) Configuration Changes

Figure 10: Configuration Changes WINVM3.

All results for analysis of winvm4 CPU cycles are available on Table 5. The critical information to take from the forecast report shows the time-series rejects the null hypothesis or non stationarity, The decomposition chart shows a level of seasonality in the data, and the ACF and PACF graphs show some level of differencing is required in the model. The three best fitting models based on BIC value and auto.arima are ARIMA(0,1,1), ARIMA(1,1,2)(1,0,0) and ARIMA(0,1,1)(0,0,2). The final candidate model has the lowest RMSE score, but accepts the null hypothesis of the Ljung-Box test. This suggests that there is a better fitting model for use.

6.4 Discussion

The results for each experiment are presented in Fig 11. The overall results show an overall prediction accuracy for CPU activity on the lab VMs of 96.90 percent for the ARIMA models. In comparison the results of the Exponential Smoothing show a 96.74 percent level of accuracy. The secondary objective of re-configuring environments to ensure correct QOS has also been met by re-configuring winvm2 and winvm3 without causing any QOS errors on the Database Environment.

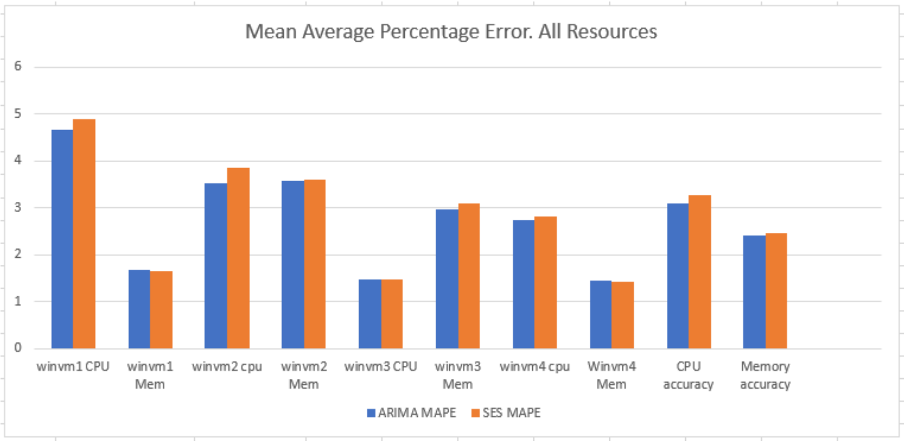


Figure 11: Overall Results

7 Conclusion and Future Work

The challenge brought forward from the work outlined in this paper was to find a way to accurately profile a Relational Database workload that ensures an optimal configuration that ensures operational excellence without an optimal configuration. This is achieved by building a framework that includes non invasive log gathering and using two trusted analytical models to forecast consumption patterns.

There are two future recommendations for future work based on this work. One is to find a way computationally to forecast more than once an hour as a lot of the logs gathered during this process were never used due to a failure to convert that much data into a time series. The second is to further integrate native Database tools (native dashboards and slow query logs) to see if it enables a more accurate prediction pattern for this type of workload.

References

Alkhater, N., Walters, R. and Wills, G. (2018). An empirical study of factors influencing cloud adoption among private sector organisations, *Telematics and Informatics* **35**(1): 38 – 54.
URL: *http://www.sciencedirect.com/science/article/pii/S0736585317303088*

Balaji, M., Rao, G. S. V. and Kumar, C. A. (2014). A comparitive study of predictive models for cloud infrastructure management, *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 923–926.
URL: *https://ieeexplore.ieee.org/abstract/document/6846547*

Calheiros, R. N., Masoumi, E., Ranjan, R. and Buyya, R. (2015). Workload prediction using arima model and its impact on cloud applications' qos, *IEEE Transactions on Cloud Computing* **3**(4): 449–458.
URL: *https://ieeexplore.ieee.org/document/6881647*

Dinda, P. (2006). Design, implementation, and performance of an extensible toolkit for resource prediction in distributed systems, *IEEE Transactions on Parallel and Distributed Systems* **17**(2): 160–173.

Drummonds, S. (Cited June 2021a). Understanding virtualcenter performance statistics.
URL: *https://communities.vmware.com/t5/Storage-Performance/Understanding-VirtualCenter-Performance-Statistics/ta-p/2780065*

Drummonds, S. (Cited June 2021b). Vcenter performance counters.
URL: *https://communities.vmware.com/t5/Storage-Performance/vCenter-Performance-Counters/ta-p/2790328*

Guo, L., Li, D. and Laguna, I. (2021). Paris: Predicting application resilience using machine learning, *Journal of Parallel and Distributed Computing* **152**: 111–124.
URL: *https://www.sciencedirect.com/science/article/pii/S0743731521000368*

Han, Y., Chan, J. and Leckie, C. (2013). Analysing virtual machine usage in cloud computing, *2013 IEEE Ninth World Congress on Services*, pp. 370–377.

Hwang, K., Bai, X., Shi, Y., Li, M., Chen, W. and Wu, Y. (2016). Cloud performance modeling with benchmark evaluation of elastic scaling strategies, *IEEE Transactions on Parallel and Distributed Systems* **27**(1): 130–143.

Jose, B. and Abraham, S. (2020). Performance analysis of nosql and relational databases with mongodb and mysql, *Materials Today: Proceedings* **24**: 2036–2043. International Multi-conference on Computing, Communication, Electrical and Nanotechnology, I2CN-2K19, 25th April 2019.
URL: *https://www.sciencedirect.com/science/article/pii/S2214785320324159*

Kim, M.-H., Lee, J.-Y., Raza Shah, S., Kim, T.-H. and Noh, S.-Y. (2021). Min-max exclusive virtual machine placement in cloud computing for scientific data environment, *Journal of Cloud Computing* **10**(1).
URL: *https://doi.org/10.1186/s13677-020-00221-7*

Logeswaran, L., Bandara, H. M. N. D. and Bhathiya, H. S. (2016). Performance, resource, and cost aware resource provisioning in the cloud, *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pp. 913–916.

Martins, P., Sá, F., Wanzeller, C. and Abbasi, M. (2019). A performance study on different data load methods in relational databases, *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–7.

Mashhadi Moghaddam, S., O'Sullivan, M., Walker, C., Fotuhi Piraghaj, S. and Unsworth, C. P. (2020). Embedding individualized machine learning prediction models for energy efficient vm consolidation within cloud data centers, *Future Generation Computer Systems* **106**: 221–233.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X19308969*

Mell, P. and Grance, T. (2011). The nist definition of cloud computing, *Technical Report 800-145*, National Institute of Standards and Technology (NIST), Gaithersburg, MD.
URL: *http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf*

Nawrocki, P., Grzywacz, M. and Sniezynski, B. (2021). Adaptive resource planning for cloud-based services using machine learning, *Journal of Parallel and Distributed Computing* **152**: 88–97.
URL: *https://www.sciencedirect.com/science/article/pii/S0743731521000393*

Prasad, A. S., Koll, D., Iglesias, J. O., Aroca, J. A., Hilt, V. and Fu, X. (2018). Rconf(pd): Automated resource configuration of complex services in the cloud, *Future Generation Computer Systems* **87**: 639–650.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X17314231*

Qiu, F., Zhang, B. and Guo, J. (2016). A deep learning approach for vm workload prediction in the cloud, *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 319–324.
URL: *https://ieeexplore.ieee.org/document/7515919*

Rahmanian, A. A., Ghobaei-Arani, M. and Tofighy, S. (2018). A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment, *Future*

Generation Computer Systems **79**: 54–71.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X17309378*

Shaw, R., Howley, E. and Barrett, E. (2020). An intelligent ensemble learning approach for energy efficient and interference aware dynamic virtual machine consolidation, *Simulation Modelling Practice and Theory* **102**: 101992. Special Issue on IoT, Cloud, Big Data and AI in Interdisciplinary Domains.
URL: *https://www.sciencedirect.com/science/article/pii/S1569190X1930125X*

Tong, J., Ying, L., Hongyan, T. and Zhonghai, W. (2016). An approach to pinpointing bug-induced failure in logs of open cloud platforms, *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pp. 294–302.
URL: *https://ieeexplore.ieee.org/document/7820284*

Van Aken, D., Pavlo, A., Gordon, G. J. and Zhang, B. (2017). Automatic database management system tuning through large-scale machine learning, *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, Association for Computing Machinery, New York, NY, USA, p. 1009–1024.
URL: *https://doi.org/10.1145/3035918.3064029*

Wang, L., Yang, Z. and Song, X. (2020). Shamc: A secure and highly available database system in multi-cloud environment, *Future Generation Computer Systems* **105**: 873–883.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X17314759*

Wang, S., Zhu, F., Yao, Y., Tang, W., Xiao, Y. and Xiong, S. (2021). A computing resources prediction approach based on ensemble learning for complex system simulation in cloud environment, *Simulation Modelling Practice and Theory* **107**: 102202.
URL: *https://www.sciencedirect.com/science/article/pii/S1569190X20301416*

Witanto, J. N., Lim, H. and Atiquzzaman, M. (2018). Adaptive selection of dynamic vm consolidation algorithm using neural network for cloud resource management, *Future Generation Computer Systems* **87**: 35–42.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X17323336*

Yuan, Y., Shi, W., Liang, B. and Qin, B. (2019). An approach to cloud execution failure diagnosis based on exception logs in openstack, *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pp. 124–131.
URL: *https://ieeexplore.ieee.org/abstract/document/8814553*

Zacarias, F. V., Petrucci, V., Nishtala, R., Carpenter, P. and Mossé, D. (2021). Intelligent colocation of hpc workloads, *Journal of Parallel and Distributed Computing* **151**: 125–137.
URL: *https://www.sciencedirect.com/science/article/pii/S0743731521000319*
ΩŁaskawiec et al.

Łaskawiec, S., Choraś, M., Kozik, R. and Varadarajan, V. (2021). Intelligent operator: Machine learning based decision support and explainer for human operators and service providers in the fog, cloud and edge networks, *Journal of Information Security and Applications* **56**: 102685.
URL: *https://www.sciencedirect.com/science/article/pii/S2214212620308371*