Automating the cloud VM estimation and scaling as a control system

Project report submitted to Indian Institute of Information Technology, Nagpur in partial fulfilment of the requirements for the Award of Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Parth Wazurkar
BT16CSE002

Under the guidance of

Dr. Jitendra Tembhurne

Department of Computer Science and Engineering

Indian Institute of Information Technology, Nagpur - 440006

2016-2020

Automating the cloud VM estimation and scaling as a control system

Project report submitted to Indian Institute of Information Technology, Nagpur in partial fulfilment of the requirements for the Award of Degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Parth Wazurkar

BT16CSE002

Under the guidance of

Dr. Jitendra Tembhurne

Department of Computer Science and Engineering

Indian Institute of Information Technology, Nagpur - 440006

2016-2020

Department of Computer Science and Engineering
Indian Institute of Information Technology, Nagpur

<u>Declaration</u>

I, **Parth Wazurkar (BT16CSE002)** hereby declare that my project work titled "**Automating the cloud VM estimation and scaling as a control system**" is carried out by me in the Department of Computer Science and Engineering at Indian Institute of Information Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

Parth

Date: 21st June, 2020 **Parth Wazurkar**

BT116CSE002

Declaration

I, **Parth Wazurkar, (BT16CSE002)**, understand that plagiarism is defined as any one or the combination of the following:

1. Uncredited verbatim copying of individual sentences, paragraphs or illustrations (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Uncredited improper paraphrasing of pages or paragraphs (changing a few words or phrases, or rearranging the original sentence order).
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did or wrote what. (Source: IEEE, the institute, Dec.2004) I have made sure that all the ideas, expressions, graphs, diagrams, etc. that are not a result of my own work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

I affirm that no portion of my work can be considered plagiarism and I take full responsibility if such a complaint occurs. I understand fully well the guide of the thesis may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Parth

Date: 21st June, 2020

Parth Wazurkar

BT116CSE002

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR

"An Institution of National Importance by an Act of Parliament"

RTTC, BSNL Near TV Tower, Beside Balaji Temple, Seminary Hills, Nagpur - 440 006

Website: www.iiitn.ac.in, Email: director@iiitn.ac.in, registrar@iiitn.ac.in Phone: 0712 - 2985010

Thesis Approval Certificate

This is to certify that the project titled "**Automating the cloud VM estimation and scaling as a control system**", is submitted by **Parth Wazurkar** with enrollment number **BT16CSE002** in the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**. The work is found to be fit for final evaluation.

[illegible]

Dr. Jitendra Tembhurne
(Supervisor)

[illegible]

Dr. Pooja Jain
(Head, Computer Science and Engineering)

Date: 21/ 06/2020

Place: Nagpur

Acknowledgements

I would like to thank the one who made this dissertation possible, my sincere appreciation and gratitude to my academic supervisor, **Dr. Jitendra Tembhurne**. I thank him for believing in me and guiding with patience. His invaluable advice, unwavering trust, and unconditional support helped immensely in the timely and successful completion of the project.

I am grateful to **Dr. O.G. Kakde**, Director, all the faculties, Computer Science and Engineering Department, Indian Institute of Information Technology, Nagpur for extending the departmental facilities for my research work.

I would also like to thank Saurav Dubey, Aman Kumar, Braj Kishor and all those who have helped me directly or indirectly during the completion of this project.

Lastly, I would like to thank my **family members** for helping me realize my own potential and their continuous moral support.

Abstract

In cloud computing, the feature which has contributed most to its development is the pay-per use model. The amount of resources to be provisioned are determined at runtime and the enterprise pays for only the provisioned resources. This automatic scaling is called as autoscaling in cloud terminology. Before deploying the resources, the enterprise needs to select the type and amount of VMs from the provider. This problem of VM selection and scaling has always been addressed seperately in the literature. In this work we formulate this problem as a holistic model considering both the aspects together. We also propose optimization models and exact algorithm to solve the formulated problem, further we model the system as a control system and evaluate our approach using varying workload data. The experimental results show that the resource cost decreases and efficiency increases as we consider the complete time horizon for computing of VMs, also the control system further decreases the resource cost, compared to the optimization model results.

Keywords: Resource allocation, cloud computing, VM scaling, VM estimation, Autoscaling, Autonomic computing

Contents

List of Figures

List of Tables

1 Introduction

Today the reach of cloud computing has increased manifold. Web application providers have either already migrated their services on cloud or are planning to do so. The ease of availability and deployment of cloud services has made it easier for enterprises and startups to host and deploy software services without much hassle of setting up infrastructure. One of the appealing features of the cloud which has contributed to the development of cloud is pay-per-use pricing model of cloud, which enables the enterprises to pay only for the amount of resources which have been utilised. Elasticity, i.e. adjusting resources according to varying demand and paying for only utilised resources (2) and in turn increasing the enterprise's profit has also been a major encouraging factor for the enterprises to migrate to cloud.

Traditional computational way demands to Set up your own physical servers and securely connect them. Cloud computing promises an easier way to access the Web safely. Not only resource and web access, but also allow dynamic allocation of resources are as lean as possible. Given the high variability a lot of work is currently focused on making clouds actively revise resource allocations in the workload of internet applications, to applications which change their workload. Such revisions permit resource sharing between cloud applications, a must for overall resource optimisation, particularly in respect of cost reduction.

Today there are many different cloud service providers in the market such as Amazon AWS, Microsoft Azure, and Google Cloud, etc. All these providers offer different configurations of virtual machines(VM) with different compute capacities (3), these are either memory optimized, compute optimized or storage optimized.

Some service providers also come up with custom VMs specific for certain type of computations like high performance computing applications. etc. (3).

1.1 Motivation

In cloud computing environments a major challenge is effective management of shared resources (for example: memory, CPU, disk). This is a difficult issue, because the system is highly dynamic, with locators using on-demand resources as a response to requests from clients. Thus the workload variation is widespread over the day. Static configurations aren't appropriate in this scenario, as it would lead to either inefficiency or over provisioning. So the systems should be autonomous, With cloud computing resources automatically administered as a reply to changes in workload.

Enterprises have different software requirements and budget constraints. Thus their resource and performance requirements are different. So, to select appropriate and optimized VM configuration (type and number of VMs) so as to minimize cost together with fulfilling resource constraints becomes an important decision problem (4).

VM selection is a challenging problem as there are multiple criteria that need to be considered and predicting the resource requirement accurately for different time epochs becomes important as the resource requirement is not static and rather dynamic (5) and stochastic (varies with time) in nature.

Enterprises currently use performance testing and some optimization techniques to select right instance type(s) for their application (6). These leads to over or under estimation of resources thus wasting a lot of money. After the deployment of these instances, based on varying workload and demand these instances are scaled accordingly (2). These systems can be either reactive or proactive(predicitve). Reactive systems, react on the basis of incoming load and scale the resources accordingly whereas predictive or proactive systems predicts the resources prior to time and scales VMs accordingly, thus improving on the accuracy of VMs and saving on costs (2). It has already been proved that proactive systems are better than reactive systems (7) because of several reasons.

This problem opens door for developing a self reliant, adaptive and autonomic infrastructure for cloud computing.

1.2 Objective

The VM scaling is a challenging problem, as there are multiple criterion which needs to be considered like minimizing the cost and latency and also satisfying constraints of Cpu, Ram, Disk, Network bandwidth. This makes it a good continuous decision problem to be solved.

Various techniques have been used previously by researchers from simple 'if-then' rules to complex machine learning, reinforcement learning based algorithms to address the problem of autoscaling. Optimization techniques, control theory have also been used previously to address the problem. The main point to note here is that all these methods have been used from the cloud service provider's(CSP) perspective, i.e to optimize the VM deployment on underlying infrastructure, or adjusting the VMs to minimize the resource usage. This has not been addressed from a enterprise's perspective before.

Also prior articles, to the best of our knowledge, have addressed the VM selection and autoscaling problem separately. None has considered these problems together from enterprise's perspective. We consider both these problem together and provide a holistic approach which optimally selects VM configuration by also considering the previous state of VMs and future estimate of resources from a enterprise's perspective.

The aim of this paper is to address this research gap and propose a system for the solution of the proposed problem. Two simple approaches are presented: Linear optimization and Control theory. Control theory provides a systematic approach to design feedback controllers to adjust the VM estimate and scaling. Initially start by implementing the problem as a simple optimization problem for one time instance and gradually bring in the dependency of previous state for determining VM estimate. We

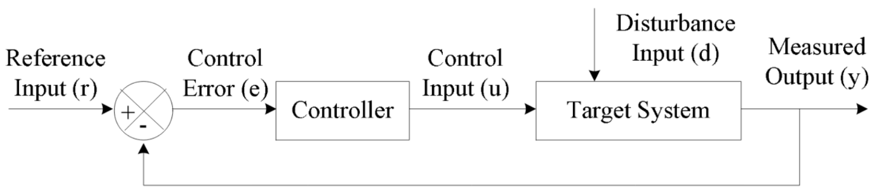

Figure 1.1: Standard feedback control system

further find that calculating the estimate together for a complete time horizon gives more optimized results.

1.2.1 Control system architecture

Control engineering ideas can be used to conceive and represent the problem, given the dynamic nature of the system. We seek to apply the principles of control theory to the Autonomous Cloud computing systems design.

In this work we present feedback control theory VM based architecture for virtualised adaptive management of cloud computing resources. Feedback control provides a conceptual instrument to address resource dynamics administration. Unforeseeable workload changes and configuration disruptions are emerging as a main viable approach to self-managed design and autonomous computers.

In Figure 1.1 an illustration of the essential standard architecture Control feedback system is shown. The target system is calculation Controlled system with controller. To get a desired objective of target system(referred to as reference input), the controller adjusts the control input value dynamically based on the error of control which is the difference between the The measured output and the reference input. The Error in Control Changes frequently due to disruptions such as workloads. The measured output represents a measurable target metric Installation.

Cloud Elasticity Implementation using a control theoretical approach generally employs a feedback loop model, in which the controller keeps the output system around certain desired value by input monitoring and system outputs. Such a control system generally can be used to meet a constraint or to guarantee in-variance on the System outputs. Fig. 1 outlines the general feedback mechanism of such a model where it notes the output of the system to correct any deviation from wanted value. The basic control elements can be seen from Fig. 1. Details of those items describe aspects of implementing control systems. Those elements therefore become the attributes and will help us analyze the perspective of the solution proposed for implementation. These attributes do not, however, tell anything about the elasticity properties of the proposed system.

Our contributions:

- Consider the VM selection and autoscaling problems together and propose a combined solution approach.
- Propose a two tier comprehensive and holistic model for VM auto scaling.
- Provide an optimization model and an exact algorithm to solve the VM selection problem.
- Propose a control system model to provide a more adaptive and dynamic methodology.

Further the thesis is organised as follows: In section 2, we address the literature review. Section 3 provides insights into the necessary background about cloud elasticity and control theory. In section 4, we describe the proposed solution. In section 5 we discuss the experimental results, and in section 6 we conclude the paper and present the future work.

2 Literature Review

This section summarizes the related state-of-the-art works in the field of cloud resource allocation to achieve VM selection and scaling.

2.1 VM selection and autoscaling based

Cloud service providers offer the web application providers (enterprises) different scaling mechanisms to manage their application compute resources, Amazon AWS Autoscaling (2) and Rightscale (8) are some examples. These mechanisms requires to specify the scaling rules manually, thus unable to get full automation. Even amazon's recently announced elasticache (9) cannot scale automatically without manual intervention.

Even though autoscaling is available, the need of selection of VMs from a vast amount of available options is important. Different approaches have been proposed for VM selection. (10) proposes a machine learning based approach where the proposed framework **Daleel** which matches the customer demands and available cloud offerings to get the optimal VM instance type. (11) proposes a performance aware data-driven framework **PARIS** which predicts workload performance for differnt VMs for some benchmark and then based on the user defined parameters profile the VMs and accordingly choose appropriate VM type. Some articles also use optimization techniques to efficiently search for the best performing VM type (12). While selection of VM type amongst different options gives a good estimate but combination of differnt VM types can give a more cost optimized and better estimate. Also, these papers only consider static resource constraints which are given by user. While an intelligent prediction based resource estimator would give a better resource estimate and in turn a better VM selection.

The problems of VM estimation and autoscaling has always been addressed seperately (5), but these problems have lot of inter dependencies (4). They have been addressed in the literature using differnt techniques like rule based techniques (2), machine learning (10) and a recent trend has been to use reinforcement learning[(13) (14) (15)]. All these paper address only the problem of autoscaling and leave apart the VM selection problem.

2.2 Control theory based

A review article, (16) sets out a detailed taxonomy, includes relevant attributes that define the following two perspectives, i.e. the control theory as an implementation technique as well as the elasticity of the cloud as target application domain. It provides a thorough review of the literature by classification of existing elasticity solutions using the theoretical perspective of control attributes. It summaries findings and presents by clustering them in terms of the type of control solutions, thus helping in comparison with the related solutions Surveillance solutions. Finally , it provides a discussion which summarizes the pros and cons of each type of control solution. This is what it is like a detailed description of the various open research challenges in the field is followed by a discussion.

Vast literature is available which addresses the problem from CSP's perspective, (17), proposes a management Automatic Resource Allocation Framework in Virtualized Applications, and identifies from experiments sources of Controlled Systems instability. They analyze two kinds of policies: threshold-based and reinforcement learning based techniques to dynamically scale resources.

In (18) addresses why control engineering approaches in the cloud elasticity context are appropriate, and illustrate the argument by mapping the problem to the problem of dynamic memory partitioning caches in cloud. Basic control system approaches are presented in such a direction as a starting point for further research in the field.

In (19), using the theory of feedback control, present a VM-based adaptive management architecture for virtualised cloud computing resources, and adaptive modeling controller that adjusts multiple virtualised resources dynamically and makes use of Service Level Objectives (SLO) to achieve application SLOs. It basically proposes a adaptive VM management solution over underlying hardware and proposes a adaptive control method.

The study presented in (20) highlights all the strengths and the restrictions on the use of control theory in researching systems. The paper starts by stating that data centers and cloud enterprises computing environments are designed with a computer utility. Consider this paradigm, which faces many challenges when trying to achieve Service Level Objectives (SLOs) in attendance. Dynamic sharing of resources and unpredictable interactions across a wide range of applications, including: complex SLOs, time varying demands on workload, distributed allocation of resources and its dependencies on resources.

The paper (21) proposes a feedback-based control approach in the AWS EC2 public cloud for VM management. they evaluate their proposed model using Gain Scheduling policy against various workloads. They also give results on the robustness of the proposed Gain Scheduling policy in the presence of failures. Their proposed control strategy tries to guarantee high performance with regard to both constant and time-varying workloads, without the need for prior information on system dynamics or complex estimates of operating conditions.

In (22), they propose a combination of resource controllers on both application and container levels. On the application level, a decision maker (i.e., an adaptive feedback controller) determines the total budget of the resources that are required for the application to meet SLA requirements as the workload varies. On the container level, a second controller divides the total resource budget among the application components to optimize the performance of the application (i.e. to minimize the round trip).

In, (23), tackles the challenge of building an efficient controller as a customer addon outside the cloud utility in the paper. It proposes such external controllers that function within the utility service APIs constraints. It is important to consider techniques for effective feedback control using cloud APIs, as well as how to design those APIs to enable more effective control.

So, a holistic model which together addresses the problem of VM selection and autoscaling in an intelligent way is needed. Thus, we propose a method which gives an intelligent estimate of VM configuration based on the predicted resource configurations together for a complete time horizon and also addresses the problem using control theory.

3 Cloud elasticity and control theory

3.1 Cloud Computing

The provision of resources and services in the cloud is on request, over the network. There are three markets related to it. It defines Infrastructure as a Service (IaaS) providing IT and network resources , for example processing, storage, bandwidth and intermediate management. Platform as a service (PaaS) means programming cloud providers supported environments and tools that consumers/enterprises can use this to build and deploy applications on to infrastructure in the cloud. SaaS (Software-as-a-Service) Designates applications to the host vendor, IaaS, and PaaS. All of them include self-service (APIs) and pay-as-you-go accounting model.

Cloud computing services include storage services including Dropbox, productivity suites such as Office 360, and computer services such as those offered by Amazon Web Services (AWS), Microsoft Azure, Rackspace and Google Cloud services, among others. Those services are available on demand, and tenants only pay for the amount and the length of use of resource. The same flexible concepts of provisioning apply also to private clouds which are internal data centers where they are virtual on-demand machines (VMs) are supplied with different units. In both public and private clouds, then there are lots of resources shared between tenants. Each has variable and distinct demands. Optimal value for money in these environments allocation must take into account techniques that are responsive, automatic, cost-aware and predictive. Take utility computing offers, for example, which include providers Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS), as well as the latest serverless computing development which allows tenant defines the functions to perform as a response to a client request . We have computers in utility computing offerings units such as virtual machines, containers, functions, or Lambdas. These computer units, which belong to different tenants and applications are packaged into physical shared machines; So, the memory, CPU, storage and networking of the machine is shared amongst them. A natural question which comes up is: How to partition or allocate resources to various tenants and Applications, so to maximize some utility function? How to select amount of resources from the available pool of VMs? These problems are basically address basic design aspects of the Iaas,PaaS or enterprise's perspectives. Thus it becomes important to address them.

3.1.1 Control Theory concepts

A control system's core ingredients are:

1. Control objectives, or inputs
2. system components
3. outputs or results part of closed-loop control

The inputs are state variables that provide information about the Controlled system that the control-system can use. Components for determining the correction value to obtain the desired output apply.

Feedback can help to stabilize a different uncontrolled system but it can also be harmful if not used appropriately. In broad terms, stability is characteristic of pace of variation in closed-loop control output installation. When the variation is too high, the instability arises. It could have adverse effects on the control system. In the systems, unstable allocation of resources can lead to a form of thrashing, where more time is spent on the system resource allocation and distribution than execution of applications, therefore its overall performance is impaired.

Automatic checking shows us there are three main sources of Control System Instability:

- Latency to obtain observable variable stationary values after a control action has been applied.

- Control power, if the minimum magnitude of the control has too great an impact making of the outputs immediately out of the goals for control;
- Input fluctuations, when they are too large to be smoothed out by too weak control, it appears nearly as in outputs.

The brief explanations of all the essential elements of a control system are provided below.

1. *Control objective* refers to the intended main objective aim for which a control system is designed, For example, to maintain an average overall response time of minor seconds.
2. *Reference input* the desired value of the system output required for controller continue. For example , the overall average CPU utilisation of all acquired VMs (cluster) must be Sixty percent.
3. *control error* refers to the difference between reference input desired and the measured value of output system.
4. *Control input* refers to the dynamic parameter calculated by the controller affecting the behavior objective system for obtaining the required reference input, for instance number of VMs.
5. *Actuator* is an element which executes the decision made from a controller.
6. *Sensor* measures the values of metrics needed by the controller for next decision to scal. Because for example, to measure the use of VMs by CPU.
7. *Controller* is the mechanism for calculating the control input values required for achieving the objective value desired, e.g., Input Reference by bearing in mind different measurements. The systematic feedback controller design consists of two steps below: (1) a formal construction system model development and (2) implementation of a supervisory control mechanism.
8. *Control system architecture* refers to pattern how to implement a particular control methodology. The most common patterns observed in research into cloud elasticity include centralized and decentralized (also termed Distributed). There are also few instances, however, where cascade and hierarchical patterns are also used.

3.2 Cloud elasticity and control theory

Cloud elasticity implementation using a control theoretical approach generally employs a feedback loop model, in which the controller keeps the output system around certain desired value by input monitoring and system outputs. Such a control system generally can be used to meet a constraint or to guarantee invariance on the system outputs. Fig. 1 outlines the general feedback mechanism of such a model where it notes the output of the system to correct any deviation from wanted value. The basic control elements you can see systems from Fig. 1. Details of those items describe aspects of implementing control systems.

Many web applications face significant, fluctuating loads. In predictable (new campaign or seasonal) situations, resources can be supplied in advance via proven capacity planning. But allocating for unplanned spiky loads becomes challenging. Scaling is a means of automatically adjusting allocated resources to an application at any given time, based on its needs. The core features are, (i) a pool of resources available that may be on demand be pulled or released, and (ii) a control loop to monitor the system and decide whether it should be in real time wanting to grow or shrink. This estimation and scaling of resources.

3.2.1 Controller from IaaS perspective

Application controller allows automatic adjustment of the capacity of virtualized applications, the performance given Goals and actual workload. Its design is open, and it enables to link custom online scaling policies to one or more potential bottlenecks (front end, layer of application, ...) an application, using the concept of scaling point. Each scaling point is a sizeable group of virtual machines controlled by a three-submodulated control loop: Monitoring, policy decision making and scaler. The monitor follows suit current (workload, performance and resource use) status, The decision-making policy comes into play at specific times Action and decides whether to take action: to grow or to shrink the resources in the scaling point, or do nothing. If so, the action chosen modifies the associated scaling VM count operations (VM allocations and destroyments) are carried out on the cloud controller via its interface. The scaler is finally in charge of the initialization Before registering all virtual machines in a scaling point. In the event of a removal, it makes sure the server is correctly unregistered before release from application assets.

The IaaS platforms are abstracted by the Cloud controller interface. It publishes a list of descriptions of virtual machines supported by The underlying platform, and offers three non-generic generics Blocking services (allocation of VM, re-dimensioning and destruction) And two asynchronous (resource delivery, and preemption) services to get resources back when necessary. Lastly, Its Design allows the use of policies for sharing resources, such as Arbitration or optimization by green.

4 Proposed Approach

4.1 Problem formulation

We try to give a new perspective to the VM scaling problem. The problem can be thought of as a two tier problem, one tier being estimating the VM configuration prior to time for different time epochs and giving a best optimized configuration for the predicted demand. The next tier being where we scale the resources if need be either horizontally or vertically. In this way our model first predicts the VM configuration for time epochs and if need be (like there comes a unprecedented spike which requires more resources than predicted) our other tier will scale accordingly.

So, briefly our model can be considered as a two layer scaling of resources done for the complete selected time horizon where the first layer predicts the VM configuration for each time epoch based on the predicted demand which in turn learns from the previous data where the second layer provides a double check of the deployed configuration thus making the scaling module more efficient and robust.

4.2 Proposed Algorithm

4.2.1 Modelling as an optimization model

Considering the problem from a optimization perspective, it can be considered as a combination of two sub problems, one being independently estimating VM configuration required at each time epoch independently to satisfy resource constraints and other to also consider the previous configuration along with resource constraints to estimate the VM configuration. This can again be done for each time epoch subsequently with time or to consider complete time horizon window and calculate the VM configurations at all time epochs together to get configurations at all time epochs. Our findings show that the later gives better results than the prior. In Figure 4.1 we implement the broker as a optimization model, which based on various available VM options and constraints of cpu, ram and storage selects the most optimized VM configurations.

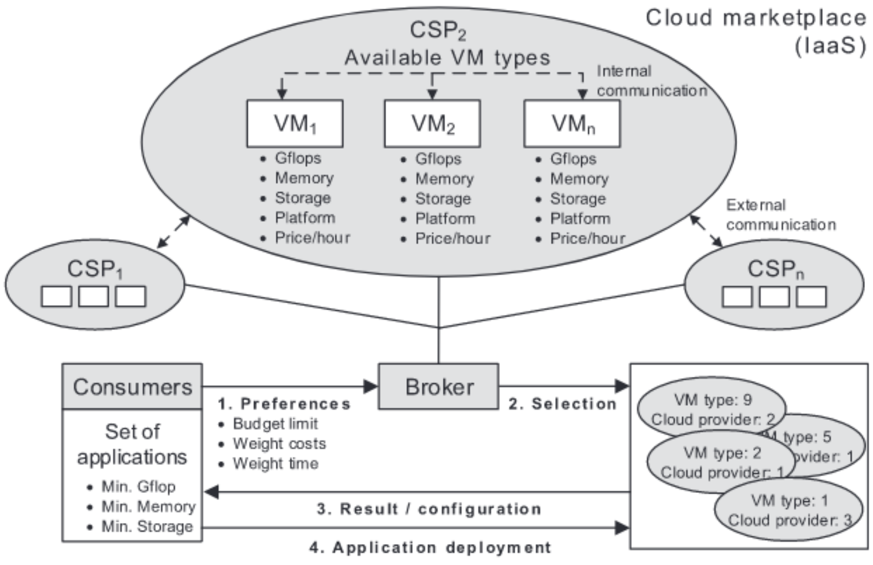
Cloud marketplace (IaaS)
CSP₂
Available VM types
Internal communication
VM₁
VM₂
VMₙ
• Gflops • Memory • Storage • Platform • Price/hour
External communication
CSP₁
CSPₙ
Consumers
Set of applications
• Min. Gflop • Min. Memory • Min. Storage
1. Preferences
• Budget limit • Weight costs • Weight time
Broker
2. Selection
VM type: 9 Cloud provider: 2
VM type: 5 Cloud provider: 1
VM type: 2 Cloud provider: 1
VM type: 1 Cloud provider: 3
3. Result / configuration
4. Application deployment

Figure 4.1: *VM selection architecture (1)*

At each time epoch we have,

- R_t is the required ram space.
- D_t is the required disk space.
- C_t is the required cpu space.
- p_i is the running cost of VM type i.
- s_i is the setup cost of VM type i.
- x_i is the number of VM type i to be selected.
- r_i is the ram size provided by instance i.
- c_i is the cpu size provided by instance i.
- d_i is the disk size provided by instance i.

Method 1 – Independently solving for each time epochs

We solve the optimization problem separately at each time epoch by minimizing the running cost + setup cost without considering previous configuration.

Mathematical model

At each time epoch t,

$$\text{minimize} \sum_{i=1}^{n}(p_i x_i + s_i x_i)$$

subject to,

$$\sum_{i=1}^{n} r_i x_i \geq R_t, \quad \sum_{i=1}^{n} d_i x_i \geq D_t, \quad \sum_{i=1}^{n} c_i x_i \geq C_t,$$

$$x_i \in Integer$$

Method 2 – Considering the previous VM state:

We solve the optimization problem separately at each time epoch by minimizing the running cost + setup cost also **considering previous configuration**.

Mathematical model

At each time epoch t,

$$\text{minimize} \sum_{i=1}^{n}(p_i x_i + s_i \max(x_i - x_i^*, 0))$$

subject to,

$$\sum_{i=1}^{n} r_i x_i \geq R_t, \quad \sum_{i=1}^{n} d_i x_i \geq D_t, \quad \sum_{i=1}^{n} c_i x_i \geq C_t,$$

$x_i \in Integer$

$x_i^*$is the number of ith VM already setup at previous instance.

Method 3 – Solving for complete time horizon

We solve the optimization problem in the **complete time horizon** by minimizing the running cost + setup cost.

Mathematical Model

$$\text{minimize} \quad \sum_{i=1}^{n}\sum_{t=1}^{T}(p_i x_{it}) + \sum_{i=1}^{n}\sum_{t=2}^{T} s_i \max(x_{it} - x_{it-1}, 0) + \sum_{i=1}^{n} s_i \max(x_{i1} - x_i^*, 0))$$

subject to,

$$\sum_{i=1}^{n} r_i x_{it} \geq R_t, \quad \forall t \in [1, T],$$

$$\sum_{i=1}^{n} d_i x_{it} \geq D_t, \quad \forall t \in [1, T],$$

$$\sum_{i=1}^{n} c_i x_{it} \geq C_t, \quad \forall t \in [1, T],$$

$$x_{it} \geq 0, \ \forall i \in [1, n] \quad \forall t \in [1, T]$$

$$x_{it} \in Integer$$

4.2.2 Modelling as a control system

Cloud computing resources are managed in a continuous process where the system has to be reset in reply to changes in workload. When we describe the problem, it is clear that the ideas of classical control engineering can be used to this purpose.

A formal definition of a feedback control loop, demonstrated at Figure 1.1, is one in which the system is controlled. The target system has a set of interest metrics (commonly referred to as measured output) and control knobs set (referred to as the controlling Input). Regularly the controller adjusts the value of the control input so that the measured output can correspond to a Value desired (referred to as reference input) specified by the input system engineer. That is, it aims at keeping the difference between the two (called a control error) at zero, despite the system disturbance (i.e. something that affects the system). Measured output non-controlled.

Traditional principles of control engineering apply to different aspects of this questionable. Analyzing and designing strategies are carried out in time or frequency domains and consideration Single-single-output (SISO) or multiple-input Settings to Output (MIMO). The first natural step is that of finding a mathematical model, which relies on differential equations, capable of representing and reproducing target behaviour. Analysis of stability is very important, since other things, that will lead to the amount determined of controller manipulation that the system is capable of Tolerance. Multiple alternatives can be used in controller design pursued, from simple Proportional Integral Derivatives Controllers (PID) to more advanced strategies as model Controllers Predictive Control (MPC), Fuzzy or Adaptive.

The target system in relation to this paper is the cloud managed component, disturbances are changes in the workload, and the output measured is optimized specific metrics (e.g., latency or throughput). The desired target metric value or service is the reference input level objective (e.g., 0.1 seconds latency) and control input is an adjustment of the resource (e.g., amount of resource). Allocation or distribution of memories).

4.2.3 Architecture

In Figure 4.2 we present handling of virtual resources, a system virtualisation architecture and feedback control. The adaptive manager is comprised of three controllers: CPU controller, memory controller, and disk controller. These controllers send commands to corresponding controllers powered by actuators like CPU capacity estimator, memory estimator and I/O estimator to estimate the corresponding resources based on current demand and controller command. Those actuators are controlled by the optimizer per VM CPU, memory and disk allocation to meet desired requirements. Application performance within VM's. VM_i has got a sensor(Sr_i) module which measures resource usage periodically. Application metric i and Sensor module (Sp_i) measure the application i's performance metric periodically, namely, CPU usage. The information which was collected by the sensors are transmitted periodically to the adaptive manager. The adaptive manager compares the performance of the Sp_i with that of the performance desired for application i calculates the required performance. Resources are computed using feedback based on Sr_i using appropriate control algorithms and send to respective controllers. So, the selection of all VMs can be automatically adjusted in real time to satisfy the resource demand of the hosted applications.

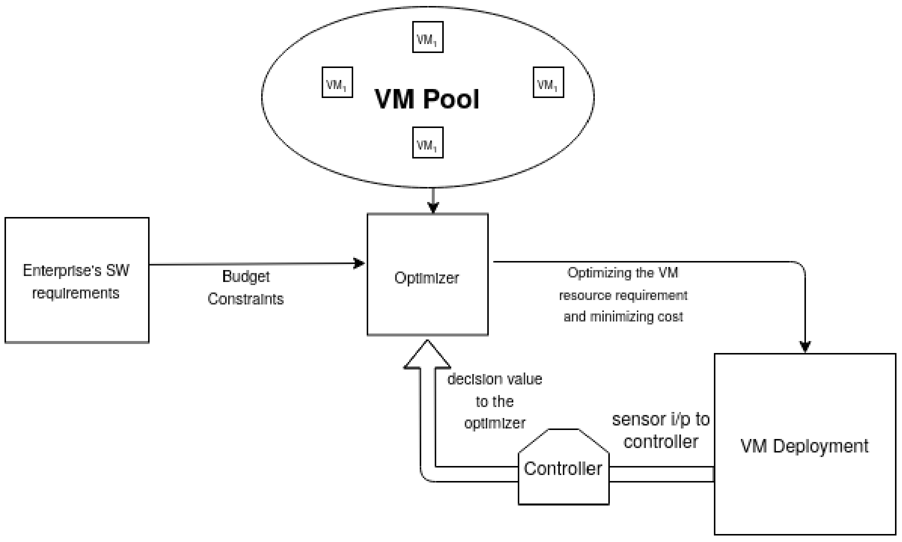
VM Pool
VM1
VM1
VM1
VM1
Enterprise's SW requirements
Budget Constraints
Optimizer
Optimizing the VM resource requirement and minimizing cost
decision value to the optimizer
sensor i/p to controller
Controller
VM Deployment

Figure 4.2: VM selecion architecture as a control system

An important contribution to dynamic development of the architecture is the ability to reallocate resources to VMs because, they are able to migrate live across networks. The live migration of VMs transport capability from one host to another without interrupting execution of the applications adds to more optimised VM selection.

Adaptive Manager

The Adaptive manager is a Multi-input, multi-output (MIMO) resource controller for which multiple regulations are intended. Whose objective being use of the virtualized resources to achieve more reduced cost of VM resources by using the CPU , Memory and I/O control inputs per VM Allotment. We use dynamic state-space feedback control for Adaptive Manager modelling.

Reference input to the controller system is

$$r = \{r_1, ..., r_n\}^T$$

, desired resources within the allocated budget. Inside VMs where r_i denotes performance desired of application i, namely the CPU need of application i. The State-space model representing the relation between input and output in the virtualised controlled system is,

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$$

where,

$$\mathbf{x}(k) = [x_1(k), ..., x_n(k)]^T$$

and

$$x_i(k) = S_i(k)S_i$$

is application offset value of cpu need during the kth time interval. S_i is the point of operation of application i's CPU need. The control input is,

$$u(k) = [u_1(k), ..., u_n(k)]^T$$

, where

$$u_i(k) = R_i(k)R_i$$

is offset value of application i's resource allocation during the kth time interval. R_i is point of operation of Application allocated resource i.

The control vector error

$$e(k) = r - y(k) = r - Cx(k)$$

. The Integrated error,

$$x_i(k+1) = x_i(k) + e(k)$$

, where $x_i(k)$ describes an error in the accumulated control. A Disruption Input d(k) affecting the control input denotes unforeseeable Changes and Workload Disruptions and Configuration at Time interval k.

The system uses the control law,

$$\mathbf{u}(k) = K[\mathbf{x}(k), \mathrm{x}_\mathrm{i}(k)]^\mathrm{T} = [K_\mathrm{p}, K_\mathrm{i}][\mathbf{x}(k), x_\mathrm{i}(k)]^\mathrm{T}$$

Where K_p denotes feedback gain for **x**(k) and K_i denotes the feedback gain of $x_i(k)$.

The adaptive manager model allows for the allocation of resources reasonably in the face of dynamic change Resource requirements of various applications implementing to achieve achieve on different VMs in the virtual resource pool to achieve resource need.

5 Results

5.1 Experimental setup

We use the VM type, as mentioned in table 5.1 described in (13) for experimental evaluation. Also we generate synthetic workload data, as in table 5.2 for four time epochs to test our models. The synthetic data has been generated so as to provide a spike and steep in demand together to better analyse the resource allocation.

Optimization models discussed in the previous section were implemented using open source PuLP (24). The controller was implemented using open source python-control library.

Table 5.1: *VM instance configurations*

VM type	*Extralarge*	*Large*	*Medium*	*Small*
Core	8	4	2	1
CPU (MIPS)	4000	2000	1000	500
RAM (GB)	15	7.50	3.75	1.7
Disk (GB)	1690	850	410	160
VM price ($/h)	2.560	1.280	0.640	0.320
VM initiation price ($/h)	3.200	1.550	0.720	0.450
VM initiation time (min)	9	7	5	2

Table 5.2: *Resource demands*

Time	*Disk*	*CPU*	*RAM*
09:00:00 AM	100000	6000	50
10:00:00 AM	200000	10000	50
11:00:00 AM	800000	200000	50
12:00:00 PM	75000	5000	50

5.2 Experimental results

We evaluated our models on the above data, using the mentioned VM types. We initially start with independently determining the VM resources on requested demand. Gradually we bring in previous state consideration, then complete time horizon and conclude our experiments by evaluating the controller.

Results of method 1(Optimizer 1) We observe from table 5.3, the 'XL' VMs are not used at all and a huge number of 'M' VMs are used for demand, which increases the total running as well as complete cost. Thus, we consider bringing in the previous state for determining VMs.

Abbreviations:

- **RC**: Running cost.
- **SC**: Setup cost.
- **TC**: Total cost.

Table 5.3: *Optimizer 1 - Results*

Time	*XL*	*L*	*M*	*S*	*RC*	*SC*	*TC*
09:00 AM	0	0	244	0	156.16	14.63	170.79
10:00 AM	0	11	465	0	311.68	15.18	326.89
11:00 AM	0	3	1945	0	1248.64	88.76	1337.40
12:00 PM	0	0	183	0	117.12	0.00	117.12

Results of method 2 (Optimizer 2) We observe from table 5.4, the VMs are distributed among ‘L’ and ‘M’ VMs. Although initially the cost remains same, later a decrease in cost is obtained. Thus reducing the complete total cost over complete time horizon. Further, we try to compute this for the complete time horizon in method 3.

Table 5.4: *Optimizer 2 - Results*

Time	*XL*	*L*	*M*	*S*	*RC*	*SC*	*TC*
09:00 AM	0	0	244	0	156.16	14.63	170.79
10:00 AM	0	11	465	0	311.68	15.18	326.86
11:00 AM	0	17	1916	0	1248.00	88.08	1336.08
12:00 PM	0	17	148	0	116.48	0.00	116.48

Results of method 3 (Optimizer 3) At the start of application deployment, we have the predicted workload demand for the coming time. Using this information we try to compute the resource allocation at the start of deployment itself. We observe from table 5.5, that although in the first epoch the cost increases, a substantial decrease in future cost is observed.

Table 5.5: *Optimizer 3 - Results*

Time	*XL*	*L*	*M*	*S*	*RC*	*SC*	*TC*
09:00 AM	0	117	1	1	150.72	20.56	171.28
10:00 AM	0	235	1	0	301.44	20.66	322.1
11:00 AM	0	236	1462	0	1237.76	87.8	1325.56
12:00 PM	0	88	1	0	113.28	0	113.28

As the application is deployed on VMs, we analyze the VM state and based on the state values we develop a controller module which will feed the live VM stats to the optimzer which will further improve the cost of VM deployment. Thus we deploy the developed controller with the Optimizer 3 and compute the results.

Results with controller and optimizer3 We observe from table 5.6 that, after the controller is brought in the system, total cost in all the time epochs after the first reduces considerably. This is possible because load is distributed amongst the 'S' VMs also. and as the VMs are deployed in the previous instance the setup cost is also reduced. Thus deployment of controller improves the enterprise's profit.

Table 5.6: *With controller and optimizer - Results*

Time	*XL*	*L*	*M*	*S*	*RC*	*SC*	*TC*
09:00 AM	0	117	1	1	150.72	20.56	171.28
10:00 AM	0	199	38	54	305.12	16.23	321.35
11:00 AM	0	204	1458	72	1136	25.32	1161.32
12:00 PM	0	76	8	32	108.29	0	108.29

Thus from table 5.5 and 5.6 we observe that when a controller is brought into the system more optimized results are obtained, with more resources spread over different VM types which decreases the subsequent setup cost, which in turn reduces the total cost.

6 Conclusion and Future work

In this work we investigate the problem of VM selection and scaling from an enterprise's perspective and we present the problem as a optimization model, to reduce VM cost together with satisfying the resource demand. We further propose a feedback controller which, based on the changes in VM state inputs the parameters to the optimizer, which further reduces the resource cost. We compute results by implementing the models and running them for different work loads. The results show that the proposed approach satisfies the resource usage and decreases the total cost while satisfying the demand. In future work, we plan to explore: the possibility of using application specific parameters in the controller to make the system application specific, try to bring in the idea of deploying multiple applications of the same enterprise together on the system to reduce cost.

6 Bibliography

[1] L. Heilig, E. Lalla-Ruiz, and S. Voß, "A cloud brokerage approach for solving the resource management problem in multi-cloud environments," *Computers & Industrial Engineering*, vol. 95, pp. 16–26, 2016.

[2] Amazon, "Amazon aws autoscaling," 2019.

[3] Amazon, "Amazon instance types," 2019.

[4] Y. Shoaib and O. Das, "Cloud vm provisioning using analytical performance models," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pp. 68–72, IEEE, 2019.

[5] J. Jiang, J. Lu, G. Zhang, and G. Long, "Optimal cloud resource auto-scaling for web applications," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 58–65, IEEE, 2013.

[6] Amazon, "Amazon choosing right instance type for an application," 2019.

[7] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 73, 2018.

[8] Rightscale, "Rightsclae autoscaling," 2019.

[9] A. elasticache, "Amazon elasticache," 2019.

[10] F. Samreen, Y. Elkhatib, M. Rowe, and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 557–563, IEEE, 2016.

[11] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz, “Selecting the best vm across multiple public clouds: A data-driven performance modeling approach,” in *Proceedings of the 2017 Symposium on Cloud Computing*, pp. 452–465, ACM, 2017.

[12] O. Alipourfard, H. H. Liu, J. Chen, S. Venkataraman, M. Yu, and M. Zhang, “Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics,” in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pp. 469–482, 2017.

[13] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, “An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach,” *Future Generation Computer Systems*, vol. 78, pp. 191–210, 2018.

[14] Y. Wei, D. Kudenko, S. Liu, L. Pan, L. Wu, and X. Meng, “A reinforcement learning based auto-scaling approach for saas providers in dynamic cloud environment,” *Mathematical Problems in Engineering*, vol. 2019, 2019.

[15] J. B. Benifa and D. Dejey, “Rlpas: Reinforcement learning-based proactive autoscaler for resource provisioning in cloud environment,” *Mobile Networks and Applications*, pp. 1–16, 2018.

[16] A. Ullah, J. Li, Y. Shen, and A. Hussain, “A control theoretical view of cloud elasticity: taxonomy, survey and challenges,” *Cluster Computing*, vol. 21, no. 4, pp. 1735–1764, 2018.

[17] X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck, “From data center resource allocation to control theory and back,” in *2010 IEEE 3rd international conference on cloud computing*, pp. 410–417, IEEE, 2010.

[18] M. Mendieta, C. A. Martín, and C. L. Abad, “A control theory approach for managing cloud computing resources: A proof-of-concept on memory partitioning,” in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, pp. 1–6, IEEE, 2017.

[19] Q. Li, Q. Hao, L. Xiao, and Z. Li, "Adaptive management of virtualized resources in cloud computing using feedback control," in *2009 First International Conference on Information Science and Engineering*, pp. 99–102, IEEE, 2009.

[20] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, "What does control theory bring to systems research?," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, pp. 62–69, 2009.

[21] D. Grimaldi, V. Persico, A. Pescapé, A. Salvi, and S. Santini, "A feedback-control approach for resource management in public clouds," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2015.

[22] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach," in *2011 31st International Conference on Distributed Computing Systems*, pp. 571–580, IEEE, 2011.

[23] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "Automated control in cloud computing: challenges and opportunities," in *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pp. 13–18, 2009.

[24] S. Mitchell, M. OSullivan, and I. Dunning, "Pulp: a linear programming toolkit for python," *The University of Auckland, Auckland, New Zealand*, 2011.