RESEARCH ARTICLE OPEN ACCESS

Framework for Application Service Behavior Prognostication with Cost-Effective Provisioning In a Utility Cloud

Monika Sainger[#1], Dr. K. P. Yadav[*2], Dr. H. S. Sharma[#3]
Mewar University, Rajesthan, Pro VC Mats University Raipur, Mewar University, Rajesthan
Correponding auther: Monika Sainger

ABSTRACT— In A Utility Cloud Resources Are Operating On A Utility Basis That Allows Customers To Pay As They Go And Only For The Resources They Will Use At Multiple Granularities For A Specified Level Of Quality Of Service. Cloud Infrastructure Services Or Iaas (Infrastructure–As-A-Service) Delivers The Computer Infrastructure Or We Can Say A Platform Virtualization Environment, As A Service And The Service Is Typically Billed On A Utility Computing Basis And Amount Of Resources Consumed. Generally, This Infrastructure Allocation Is Static And Resources Are Allocated During VM Instantiation. Any Change In Workload Leading To Significant Increase Or Decrease In Resources Is Handled By VM Migration. Hence, Cloud Users Tend To Characterize Their Workloads At A Coarse Grained Level Which Potentially Leads To Over-Allocated VM Resources Or Under-Performing Application. That Means Either User Is Paying More In Case Of Over-Allocation Of Resources Or The Application Is Under-Performed In Case Of Under-Allocation Of Resources. In This Paper, We Present A Framework For Iaas Utility Cloud Where A Prognostication Component Predicts At Run-Time The Expected Demand By Application Which Is Then Used To Modulate Resource Allocation Based On The Predicted Demand. We Derive This Component Based On A Cost Model To Make It More Cost-Effective For Both The Service User And The Service Provider. Confidence Interval For Predicted Workload Is Used To Minimize This Effective Cost.

Keywords— Utility Cloud, Application Service Prognostication, Cost-Effective Model, Confidence Interval, Resource Allocation.

Date of Submission: 27-02-2018 Date of acceptance 14-03-2018

I. INTRODUCTION

Cloud Computing Is A Pay-Per-Use Model Where In Cloud Users Pay For The Resources That Are Allocated To Them. The Pay-Per-User Characteristic Of Cloud Can Be Leveraged By Users By Opting For Cloud Services Especially In The Case Of Variable Workloads And Time-Critical Workloads. For Variable Workloads, Users Might Request For Variable Resources At Different Times And Pay Accordingly, Instead Of Buying Their Own Resources To Satisfy The Workload At Peak Times. Infrastructure-As-A Service (Iaas) Provides The Physical Machines (Pms) Or Virtual Machines (Vms) And Other Hardware Resources As A Service To The Users. VM Is An Abstraction Of The Underlying Hardware, And Hence Is A More Flexible Way Of Providing The Service To Customers. VM Provides An Isolated Environment Which Is In Full Control Of The Customer. Iaas Allows The Cloud Users To Install Their Own Stack Of Software In The Isolated VM Environment.

II. IAAS ARCHITECTURE

Generally, Large Data-Centers Have Series Of Many Distributed Systems That Form The Underlying Physical Infrastructure. In Order To Provide Infrastructural Resources As A Utility, There Must Exist Some Software Object Which Controls And Manages The Infrastructure, And Provide The Virtual Resources. This Software Object Is Called Infrastructure Manager Is Generally Referred To As Cloud OS. Cloud OS Manages The Deployment Of Vms On To Physical Machines (Pms). It Also Needs To Cater To Dynamic Resource Allocation, That Is, Increasing Or Decreasing Demand Of Vms.

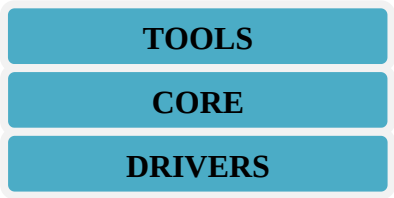

Fig. 1: Cloud OS Architecture

There Are A Number Of Iaas Architectures That Are Currently Being Used To Build Iaas Cloud. In This Work, We Explain Iaas Architecture With The Help Of An Example Of Opennebula, Which Is Widely Used And Similar To Other Iaas Architectures. Figure 1 Shows The Cloud OS [1] Architecture For Opennebula. Cloud OS Architecture Is Divided Into Three Main Layers : Tools, Core And Drivers. Tools Are The Components To Interact With The External World And To Take Inputs From Administrators Regarding Different Policies. Core

Components Are The Backbone Of The Cloud OS, And Drivers Are The Components Used To Communicate With Local Infrastructure Or External Clouds.

Fig. 2: Core Layer Of Opennebula Iaas Architecture

The Main Components Of The Core Layer Of Opennebula Iaas Architecture (Fig. 2) Are The Following:

- Scheduler: One Of The Challenging Task In Iaas Cloud Is The Optimal Placement Of Vms Onto The Physical Hosts. Scheduler Takes Decision To Place A VM Either On One Of The Physical Server, Or On To The External Clouds. Scheduler Can Also Take Dynamic Reallocation Decisions For The Variable Workloads To Optimize Some Criteria Based On Different Policies.
- Information Manager: Information Manager Is A Monitoring System, Which Checks The State Of Vms, Server Resource Utilization, Network Usage, Etc..
- VM Manager: A VM Is The Basic Allocation Unit In A Cloud OS. VM Manager Is Responsible For Allocating Vms And Managing A VM's Life Cycle.
- Accounting And Auditing: It Keeps Track Of The Usage Information Of The Deployed Services. It Is Essential To Produce Billing Information And Protect It From Threats Like Unauthorized Access, Abusive Use Of Resources And Other Forms Of Intrusion.
- Authorization And Authentication: It Incorporates Mechanisms To Verify The Identity Of Users, And Ensure Their Permissions To Access Different Cloud Resources.
- Image Manager: It Manages The Varied Of VM Images, And Support Creation, Deletion, Cloning And Sharing Of VM Images.
- Network Manager: Network Manager Manages Private Virtual Networks Among Vms In Multi-Tier Applications And Assigns Public Ips To Vms. It Also Ensures Traffic Isolation Between Different Virtual Networks.
- Storage Manager: It Provides Storage Services As A Commodity, Ensuring That Storage System Is Scalable, Highly Available, And Delivers High-Performance For Data-Intensive Workloads. It Relies On External Storage Drivers To Meet These Goals By Creating A Storage Resource Pool.
- Federation Manager: It Enables Access To Remote Cloud Infrastructures.

III. LIMITATIONS OF CURRENT IAAS ARCHITECTURE

In The Present Iaas Utility Cloud Systems, The Resources Are Allocated Statically In The Form Of Virtual Machines (Vms) That Means Cannot Be Changed Over Time. However, Real Time Time-Varying Workloads Require Time-Varying Resources To Keep Their Usage Low And Without Leaving Resources Idle. In Order To Acquire More Resources A Cloud User Needs To Request For More Number Of Vms With Same Configuration Or Migrate Its Application To Another VM With More Resources. For Example, Amazon Provides Few Standard Types Of Vms [2] With Some Pre-Defined Configuration Like Small, Medium, And Big Instance, And Some Job Specific Instances Like High Memory, High CPU, High I/O Instances. Resource Provisioning Can Be Done As Per The Workload Requirements And Pricing By Static Allocation.

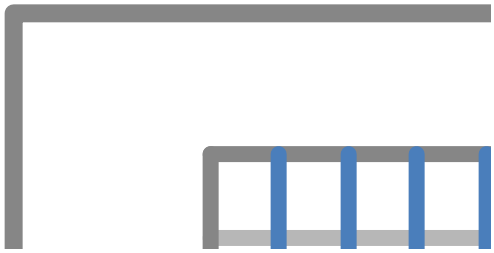

FIG 3: Typical Workload Characteristics Of Web Server

Figure 3 Shows The Actual Workload Of A Web Server Hosted In Our Institute. In The Figure, X-Axis Represents Time In Hours And The Y-Axis Indicates The Web Server Workload, Measured As The Number Of Http Requests Per Hour, Received By The Server. There Are Higher Requests Received By The Server During The Day Time As Compared To Night Times. With The Existing Resource Provisioning Model, For This Workload, A User Would Demand Two Types Of Vms, As Indicated By The Peak And Trough Of The Workload Corresponding To Allocated Resources Curve. However, As Is Shown, The Resources Are Mostly Over-Allocated. Assuming Linear Relationship Between Workload And Resources, Calculations Show That The Effective Utilization Of Resources As Per This Allocation Is Just About 30%, Which Is Ratio Of Area Under Curve Of The Actual Workload To The Workload Corresponding To Allocated Resources. In This Type Of Cases, Users Of A Utility Cloud End Up Paying More Than What They Actually Used.

Cloud Providers Also Cannot Achieve High Server Utilization With Present Provisioning Models. Over-Allocated Resources Are Idle And Available But Provider Cannot Release Them For Better Usage. Inefficient Models Thus Defeat The Whole Idea Of Achieving High Server Utilization Using Cloud Computing.

One More Challenge With The Existing Iaas Architecture Is To Deal With The Virtualization Overhead. Virtualization Overhead Plays A Major Role In Impacting The Performance Of The Application Service, Especially For I/O Workloads As The Vms Run On Virtualized Platforms.

IV. PROPOSED FRAMEWORK

For The Varying Workloads, The Present Provisioning Models Ensure Good Performance Of Application, But Result In Significant Wastage In The Form Of Idle Resources. To Overcome This Gap, In This Paper, We Propose A Modified Iaas Architecture In A Utility Cloud As Shown In Figure 2. The Picture Shows A Modified Opennebula Iaas Architecture. To Enable Flexibility In Provisioning Of Resources, We Propose A Prognostication Engine Based On Cost Model (Shown In Black Color) In The Core Layer Of Iaas Architecture. Addition Of This Component May Be Affecting Some Other Components Of The System. These Components Are Information Manager, VM Manager And Scheduler. In The Existing Iaas Architecture, VM Manager Allocates The Virtual Machines To The Applications. Scheduler Interacts With VM Manager To Reallocate The VM On The Selected Server When Workload Increases. The Component Information Manager Monitors The Resource Utilization In The Vms, We Propose A Component Prognostication Engine Based On Cost Model Into The Existing Architecture, Which Basically Predicts The Resource Requirement Based On The History Of The Application Workload That The VM Hosts. It Provides Exquisite Resource Requirements To The VM Manager. VM Manager Then Tries To Fulfill The Demand On The Same Server. If It Is Not Possible To Fulfill The Demand Locally It Interacts Back With Scheduler To Allocate Resources On Other Server. The Component Information Manager, In The New Framework, Also Collects And Interprets The Workload History Saved In VM And Presents It To The Prognostication Engine [3]. The Prognostication Engine Then Predict The Application Workload In Next Cycle Based On Past Resource Usage Pattern Of The Application That The VM Hosts And Resource Manager Further Translates This Predicted Workload Into The Resource Requirements.

The Prediction Made By The Prognostication Engine May Not Be Always Correct, Therefore Leading To Under-Allocation Or Over-Allocation Of Resources. Over-Allocation Leads To Under Utilization Of Resources And Under-Allocation Leads To Poor Performance Of The Application. To Assess The Same, We Derive An Excess Cost Model And Formulate The Problem Into A Cost Optimization Problem. The Excess Cost Comprises Of Two Components, Namely, Cost Due To Over-Allocation Of Resources ($C_{over\ Allocation}$) And The Penalty Cost ($C_{slapenalty}$) Corresponding To Poor Application Performance Resulting From Under-Allocation Of Resources. The Goal Of Optimization Problem Is To Reduce The Excess Cost. The Basic Intuition Of This Cost Model Is To Reduce The Resource Cost Of The User By Enabling Resource Allocation Closer To What Is Actually Used, Without Compromising On The Application Performance.

V. PROGNOSTICATION MODEL

Prognostication Or Forecasting Means Estimating A Future Event. Forecasting Can Be A) Causal Forecast [4] [5] Where In Several Factors Are Identified Which Influence The Forecast Variable And A Complete Cause And Effect Model Is Developed. B) Non-Causal Forecast [6] [7] [8] [9] Where In Prediction Is Made Based On The History (Past Patterns In The Data) And Are Usually Known As "Time-Series" Methods. In Non-Causal Methods, Forecast Is Usually Not Influenced Or Minimally Influenced By Other Factors. In This Work, We Have Used A Non-Causal Forecasting Model To Forecast Cloud Workloads Where In The Past Workload Is Used To Predict The Workload For Next Scheduling Cycle For Cost-Effective Allocation Of Resources. In This Work Forecasting Strategy Used Is Gaussian Process Model. Gaussian Process Models Are Prediction Models Which Are Non-Parametric In Nature That Means We Do Not Have To Worry About Whether It Is Possible For The Model To Fit The Data That Can Be Linear Or Nonlinear.

A Gaussian Process Is A Collection Of Any Finite Number Of Random Variables Which Have Joint Gaussian Distribution. Hence The Problem Can Be Reduced To Finite Dimensions By Taking Only The Points Containing Training Data And Test Data. The Interesting Characteristic Of Gaussian Distribution Is That It Can Be Characterized By Only Two Parameters, Mean And Covariance. Hence, Its Mean Function M(X) And The Covariance Function K(X, X'), Given By

$M(X) = E[F(X)]$

$K(X, X') = E[(F(X) - M(X))(F(X') - M(X'))]$

And The Gaussian Process Can Be Written As

$F(X) \sim GP(M(X), K(X, X'))$

Gaussian Processes Assume Gaussian Distribution As Prior Over The Function Values. Each Value Of The Function Is Assumed To Be A Random Variable With Gaussian Distribution. And The Set Of Values Of The Function (Set Of Random Variables) Also Form A Joint Gaussian Distribution.

Gaussian Processes Also Represent A Powerful Way To Perform Bayesian Inference [10] About Functions. In Bayesian Inference, A Prior Probability Distribution Is Assumed Initially To Describe The Underlying Process Generating Data, And Then After Gaining Knowledge About The Observed Values A Posterior Probability Distribution Is Obtained. A Prior Denotes The Initial Assumptions About The Data, Generally, The Kind Of Relation That The Data Points Can Exhibit. Then, A Posterior Improvises The Knowledge Of The Observed Over The Prior.

That Means, Given The Training Data Set D = {(X(I) , Y(I))|I = 1, 2, ..., N} And A Test Point X(N+1), The Goal Is To Compute The Distribution P(Y(N+1) |D, X(N+1)), Which Can Be Further Used For Prediction Purposes. In This Work, For The Prediction Of Resource Requirements, We Use Gaussian Model With Bayesian Inference

VI. PROGNOSTICATION ENGINE

The Key Component In The Proposed Framework Is The Prognostication Engine Which Uses An Excess Cost Model To Arrive At An Optimal Resource Requirement Based On Predicted Workload For The Application. The Prognostication Engine Uses A Confidence Interval For The Predicted Workload To Optimize The Resource Requirements. Following Figure 4 Details The Component Prognostication Engine.

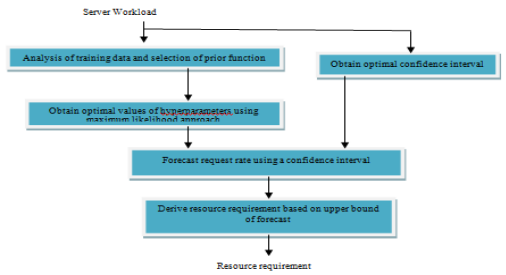

Fig.4: Prognostication Engine based on Gaussian Process

The Input To The Engine Is The Application Workload And It Outputs The Optimal Resource Requirements. The Prognostication Engine Uses Past History Of The Application Workload And A Prior Covariance And Mean Function Is Selected Offline Based On Some Initial Assumptions About Data. Then After Gaining Knowledge About Observed Data A Posterior Probability Distribution Is Obtained.

VII. FINDING OPTIMAL CONFIDENCE INTERVAL

The Optimal Confidence Interval Is Based On The Minimization Of The Over-Allocation Cost $C_{over\ Allocation}$ And The Optimization Problem Can Be Formulated As Follows:

$$\text{M Inimize} \quad C_{effective} = C_{over\ Allocation}(A)$$

$$\text{Subject To} \quad 0 < A < 100$$

Where, A Is The Confidence Interval. Cost $C_{over\ Allocation}$ Is A Function Of Confidence Interval. The Solution To This Optimization Problem Would Provide Us The Optimal Value Of Confidence Interval, Which Would Result Into Minimized Effective Cost.

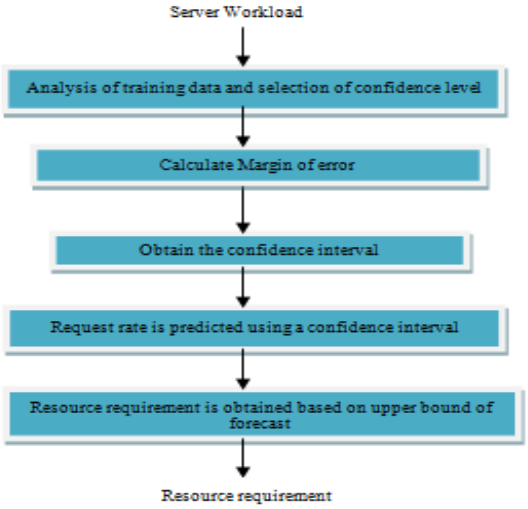

Fig.5: Finding Confidence Interval

Figure 5 Further Describes The Component Find Optimal Confidence Interval Discussed In The Figure 4. First Of All, We Split The Data Into Two Parts: Training Data And Test Data. Using The Training Data, A Prior And A Confidence Level Is Selected. An Optimal Confidence Level Is Obtained Using A Critical Value Of 1.96 And Standard Deviation Error. Then A Model Is Built And We Derive The Resources Based On Upper Bound Of Confidence Interval. Then, We Compare The Predicted Resources With The Resources Actually Required, And Calculate The Over-Allocation Cost. Effective Cost Is The Minimum Over-Allocation Cost For All Of The Test Points. For Each Point, We Need To Update The Model To Include The Next Test Point. Note That Update Does Not Mean Recalculation Of Model Parameters Every Time. Then, We Update The Confidence Interval And Repeat The Procedure Again To Find Effective Cost, And The Objective Is To Reach To A Point Near To Optimal Confidence Interval, Where Effective Cost Is Minimized.

VIII. EXEMPLIFICATION

As An Example, We Analyze The Workload Of The Web Server Hosted In Our Academic Institute. Web Servers Are One Of The Most Popular I/O Workloads

That Are Getting Hosted On Cloud Systems [3]. These Workloads Can Particularly Benefit From Inherently Elastic Resource Provisioning.

A. Analysis Of Training Data And Selection Of Prior Function

For The Gaussian Process, The Prior Is Expressed As An Initial Estimate Of Meanand Covariance Of The Function. If There Is No Reason To Prioritize One Mean Function Over The Other, It Is Initially Assumed To Be Zero. Covariance Function Encodes The Assumptions About The Similarity Of The Various Data Points In Function. Based On Prior Knowledge About The Data An Appropriate Covariance Function Is Chosen. Both Mean And Covariance Functions Are Specified By A Set Of Hyperparameters That Are Collectively Represented By Θ Here. These Are The Hyperparameters For Which Optimal Values Need To Be Calculated. In Our Work We Analyzed The Workload Of Our Institute's Web Server And Mail Server. It Has Been Observed That The Workload On Servers Starts Getting Heavily Loaded Afternoon Onwards Then In Evening And Night As Compared To In Morning. Following This Observation We Considered And Collected The Sample Data In Two Different Time Slots; One From 12 Pm To 12 Am And Another From 12 Am To 12 Pm. After Analyzing The Data, The Basic Assumption About Both The Webserver And Mailserver Workloads Is That They Are Locally Periodic I.E. They Are Periodic, But They Can Slowly Vary Over Time. In Other Words, They Don't Repeat Themselves Exactly. Hence, Locally Periodic Covariance Function Class Is Taken To Be As The Prior For The Given Data. Mathematically, A Locally Periodic Covariance Function Class Is Of The Following Form:

$K_{\mathbf{localper}}(X,X') = K_{per}(X,X')\ K_{se}(X,X') = \Theta_1^2 \exp(-\frac{2\sin^2(\Pi|X-X'|/\Theta_2)}{\Theta_3^2}) \exp(-\frac{(X-X')^2}{2\theta_3^2})$

GP Prior $= (0, \Theta_1^2 \exp(-\frac{2\sin^2(\Pi|X-X'|/\Theta_2)}{\Theta_3^2}) \exp(-\frac{(X-X')^2}{2\theta_3^2}))$ (1)

Here Θ_1 Captures The Magnitude Of Similarity In Nearby Data Points, Θ_2 Length Scale Of The Function After Which The Value Of The Function Can Change Significantly I.E. Period Of The Function. And Θ_3 Controls The Consistency In Data In Periodic Interval. Also A Confidence Level Of 0.95 Is Selected. In The Beginning, An Initial Value Of The Confidence Interval Is Specified As The Input. After That, This Value Is Provided As A Feedback To Minimize The Effective Cost Of The System.

B. Obtaining Optimal Values Of Hyperparameters

(1) Likelihood Function:

After Selecting The Prior Function Next Step Is To Determine The Optimal Values Of Hyperparameters. This Step Is Very Important To Develop Useful And Effective Gaussian Process Models. In Bayesian Analyses When The Prior Is Selected And Observed Distribution Is Collected Then The Likelihood Of Observed Distribution As A Function Of Parameter Values Is Obtained. Likelihood Is The Hypothetical Probability That An Event Has Already Occurred Would Yield A Specific Outcome. A Likelihood Function Takes The Data Set As A Given And Represents The Likeliness Of Different Parameters For Collected Distribution. Let X1, X2, ..., Xn Have A Joint Density Function F(X1, X2, ..., Xn|Θ). Given X1 = X1, X2 = X2, ..., Xn = Xn Is Observed, The Function Of Θ Defined By: P(Θ) = P(Θ|X1, X2, ..., Xn) = F(X1, X2, ..., Xn|Θ) Is The Likelihood Function. Likelihood Function Measures The Support Provided By The Data For Each Possible Value Of Parameter. If We Compare Likelihood Function At Two Parameter Points And Find That $P(\Theta_1|X) > P(\Theta_2|X)$

Then The Sample We Actually Observed Is More Likely To Have Occurred If $\Theta = \Theta_1$ Than If $\Theta = \Theta_2$. This Can Be Interpreted As Θ_1 Is More Plausible Value For Θ Than Θ_2.[Mon1]. According To Likelihood Principle If X And Y Are Two Sample Points Such That $P(\Theta|X) \propto P(\Theta|Y)\ \forall\ \Theta$ Then The Conclusions Drawn From X And Y Should Be Identical. Thus The Likelihood Principle Implies That Likelihood Function Can Be Used To Compare The Plausibility Of Various Parameter Values. For Example, If $P(\Theta_2|X) = 2P(\Theta_1|X)$ And $P(\Theta|X) \propto P(\Theta|Y)\ \forall\ \Theta$, Then $P(\Theta_2|Y) = 2P(\Theta_1|Y)$. Therefore, Whether We Observed X Or Y We Would Come To The Conclusion That Θ_2 Is Twice As Plausible As Θ_1. For Our Case, Using Experimental Server Workload, The Model Parameters (In Eq (1)) That Best Describe The Model Are $\Theta_1 = 1200.58$, Θ_2=12 Hours, $\Theta_3 = 1$.

(2) Posterior Function:

: In A Bayesian Framework, The Posterior Of Huperparameters Can Be Defined As Follows:

$P(\Theta|Y,X) = P(Y|X,\Theta)\ P(\Theta)$

Where $P(\underline{\Theta}|Y,X)$ Is The Posterior Of The Hyperparameters,

$P(\Theta)$ Is The Prior Of The Hyperparameters,

$P(Y|X,\Theta)$ Is Likelihood Function (The Probability Density Of Observations Given The Parameters),

Here We Have Used An Approach Which Uses The Observation That Likelihood Function Is Directly Proportional To The Posterior Distribution. It Selects The Θ Corresponding To Maximum Value Of The Likelihood Function And Is Called Maximum Likelihood Type II (ML-II) [11] Approximation. Maximum Likelihood Approach Is A Common Approach Used In Selecting Values Of Hyperparameters For Gaussian Processes [12]. A Systematic Analysis Of Hyperparameters And

Posterior Probabilities Is Executed Offline To Build A Prediction Model.

(3) Forecasting:

We Use The Above Model To Predict The Resource Requirement Within A Confidence Interval. A Probabilistic Bound Of The Forecast, Within An Obtained Confidence Interval, Is Generated Using The Prediction Model. For Instance, Upper And Lower Bounds Of The Forecast For 95% Confidence Interval Implies That The Probability That The Forecast Would Be Within The Upper And Lower Bounds Is 0.95. We Use The Upper Bound Of The Forecast To Provision The Resources To Keep The Performance Of Application Intact. We Used 120 Observations To Construct The Model And It Is Forecasted And Tested For The Next 120 Observations. 0% Confidence Interval Implies The Actual Forecast With No Bounds. As The Confidence Interval Is Increased, The Upper Bound Of The Forecast Moves Upwards. That Means By Increasing Confidence Interval, Over-Allocation Cost Will Increase And Penalty Cost Will Decrease. Hence, Confidence Interval Is The Key To Control The Effective Cost Function Associated With The Prediction.

(4) Obtaining Resource Requirements:

Fig. 6 Shows The Resource Requirements For Experimental Data On Web Server. We Derive Resource Requirements For The Observed Data By Using An Experimental Cloud Set Up. In The Experimental Setup, Web Server Is Hosted On A VM And We Used The Web Server Logs To Synthesize Workload. In Order To Generate The Http Requests, We Use Httperf [13] As A Client Program. Along With Generating Varying Http Requests, It Also Has Other Capabilities Like Measuring Average Response Time And Throughput, Which Would Help Us Evaluating Our Model In The Later Steps. We Run Our Experiments On An AMD 2.4 Ghz System Having 12 Cores And 16 GB Memory. Figure 7 Shows The Variation Of VM CPU Usage With Request Rate. As Expected, CPU Requirement Of VM Increases With Increase In Request Rate [14]. We Use This To Derive The Resources Required To Support A Given Workload (Request Rate). The Forecasted Request Rate Can Be Used To Provision The Infrastructural Resources. . Based On The Variation Of VM CPU With Request Rate As Mentioned In , We Calculate The CPU Requirement Using Nearest-Neighbor Interpolation. We Round Off The Request Rate To The Nearest Multiple Of 100 Requests/S, And The Corresponding VM CPU To The Nearest Multiple Of 2%. For Example, If The VM CPU Requirement Has Been Derived To Be 14.9%, We Allocate 16% Of The CPU. Also, Using The Actual Request Rate Values, We Calculate The Actual VM CPU That Is Required, Such That The Response Time Of The System Stays Within The Defined Limits. Figure 7 Shows The Comparison Of The Actual And Predicted VM CPU Requirement, Based On The Earlier Forecast Of The Request Rates. Since It Can Be Shown That The VM CPU Requirements Is Almost Linear With Respect To Request Rate, Forecast Of CPU Requirement Is On The Similar Lines To That Of Forecast Of Request Rate, But Has Discrete Values Of CPU Because Of The Rounding Off.

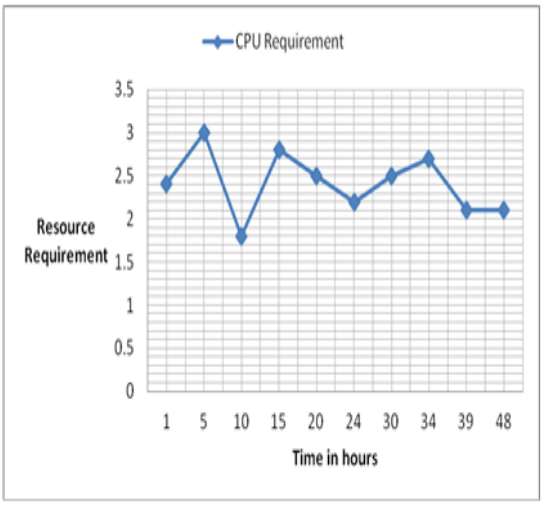

FIG 6: CPU Requirement For Workload

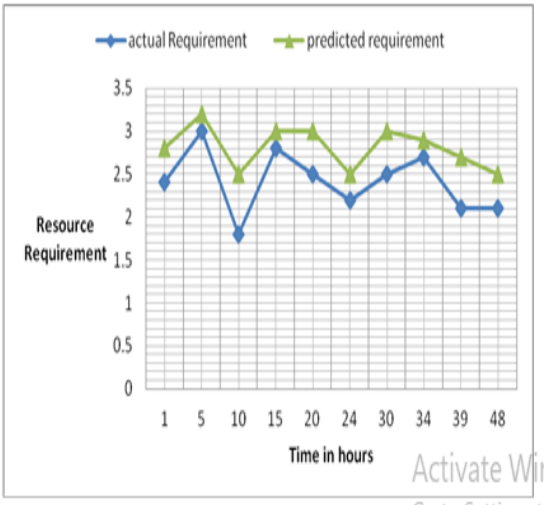

FIG 7: Actual And Predicted Requirement

(5) Validation Of Prediction Accuracy:

In This Work, We Predict The Request Rate And Use It To Derive The Resources For Allocation. Hence, Forecast Accuracy Is Measured In Terms Of Resource Required And Resource Allocated Based On Prediction. Several Performance Metrics Have Been Proposed In Literature To Measure Forecast Accuracy. In This Paper, We Use An Absolute Percentage Accuracy Measure, SMAPE (Symmetric Mean Absolute Percentage Error). SMAPE Is Defined As Follows:

$$SMAPE = \frac{\sum \left| X_t - \hat{X}_t \right|}{\sum \left| X_{t+} \hat{X}_t \right|}$$

Here, X_t Is The Actual CPU Requirement And $\hat{X}_T$ Is The Predicted CPU Requirement. For Our Case, The Value Of SMAPE For The Actual CPU Requirement Prediction (0% Confidence Interval) Has Come Out To Be 7.7051%, Which Is Quite Good And Shows The Credibility Of Our Approach.

(6) Deriving The Effective Cost:

As We Discussed Earlier In This Paper That The Effective Cost Is A Function Of Confidence Interval A, We Can Define The Same As Follows:

$C_{effective} = C_{over\ Allocation}(A)$

And $C_{over\ Allocation} = A * ER$

Where ER Denotes Excess Resources Which Is The Difference Between Resources Predicted And Resources Actually Required.

A Minimum +Ve (Positive) Effective Cost Indicates A Good Prediction That Means User Is Paying For What Has Been Used Actually Whereas A –Ve (Negative) Effective Cost Represents A Bad Prediction That Results In Under Performance Of Application.

IX. CONCLUSION

Adaptive Allocation Of Resources Based On Variations In Workload, Improves The Overall Resources Utilization Of The System. In This Paper We Have Described The Framework For A Prognostication Engine By Capturing The Changes In Workload Demand. This Approach Provides A Cost Effective Allocation Of Resources Which Is Beneficial For Both Cloud User As Well As Cloud Provider. Here The Option Of An Application Specific Prognostication Engine May Be Argued But It Is A Challenge For An Application Developer To Develop An Application Specific Prognostication Engine. However, If The Iaas Framework Provides Some Provision For A Standard Prognostication Engines Many Applications Will Benefit. Therefore A Prognostication Engine Generic Enough Like This Will Be Able To Cater Many Applications Needs.

REFERENCES

[1]. R. Moreno-Vozmediano, R. Montero, And I. Llorente, "Iaas Cloud Architecture: From Virtualized Datacenters To Federated Cloud Infrastructures," Computer, Vol. 45, No. 12, Pp. 65–72, 2012.

[2]. S. M. Metev And V. P. Veiko, *Laser Assisted Microtechnology*, 2nd Ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[3]. S. M. Metev And V. P. Veiko, *Laser Assisted Microtechnology*, 2nd Ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[4]. B. Jeong, H.-S. Jung, And N.-K. Park, "A Computerized Causal Forecasting System Using Genetic Algorithms In Supply Chain Management," J. Syst. Softw., Vol. 60, No. 3, Pp. 223–237, Feb. 2002. [Online]. Available: Http://Dx.Doi.Org/10.1016/S0164-1212(01)00094-2

[5]. W.-B. Yu, "Agent-Based Demand Forecasting For Supply Chain Management," Ph.D. Dissertation, Louisville, KY, USA, 2003, Aai3089526.

[6]. H. Song And G. Li, "Tourism Demand Modelling And Forecastinga Review Of Recent Research," Tourism Management, Vol. 29, No. 2, Pp. 203 – 220, 2008. [Online]. Available: Http://Www.Sciencedirect.Com/Science/Article/Pii/S0261517707001707.

[7]. H.-Y. Li, C. S. Xie, And Y. Liu, "A New Method Of Pefetching I/O Requests," In Networking, Architecture, And Storage, 2007. NAS 2007. International Conference On, 2007, Pp. 217–224.

[8]. F. Li And P. Luan, "Arma Model For Predicting The Number Of New Outbreaks Of New- Castle Disease During The Month," In Computer Science And Automation Engineering (CSAE), 2011 IEEE International Conference On, Vol. 4, 2011, Pp. 660–663.

[9]. S. Rajagopalan And S. Santoso, "Wind Power Forecasting And Error Analysis Using The Autoregressive Moving Average Modeling," In Power Energy Society General Meeting, 2009. PES '09. IEEE, 2009, Pp. 1–6.

[10]. M. Osborne, "Bayesian Gaussian Processes For Sequential Prediction, Optimisation And Quadrature," Ph.D. Dissertation, Phd Thesis, University Of Oxford, 2010.

[11]. C. E. Rasmussen And C. K. I. Williams, Gaussian Processes For Machine Learning (Adaptive Computation And Machine Learning). The MIT Press, 2005. [Online]. Available: Http://Www.Gaussianprocess.Org/Gpml/Chapters/.

[12]. S. Brahim-Belhouari And J. Vesin, "Bayesian Learning Using Gaussian Process For Time Series Prediction," In Statistical Signal Processing, 2001. Proceedings Of The11th IEEE Signal Processing Workshop On, 2001, Pp. 433–436.

[13]. D. Mosberger And T. Jin, "Httperf-A Tool For Measuring Web Server Performance," SIGMETRICS Perform. Eval. Rev., Vol. 26, No. 3, Pp. 31– 37, Dec 1998. [Online]. Available: Http://Doi.Acm.Org/10.1145/306225. 306235.

A. Anand, M. Dhingra, J. Lakshmi, And S. K. Nandy, "Resource Usage Monitoring For Kvm Based Virtual Machines," In Proceedings Of The 18th Annual International Conference On Advanced Computing And Communications (ADCOM 2012), To Be Published, Dec. 2012.

Monika Sainger"Framework for Application Service Behavior Prognostication with Cost-Effective Provisioning In a Utility Cloud"International Journal of Engineering Research and Applications (IJERA) , vol. 8, no. 03, 2018, pp. 15-22