inteliScaler

Workload and Resource Aware, Proactive Auto Scaler for PaaS Cloud

Paper #10368
RS Shariffdeen, UKJU Bandara, DTSP Munasinghe, HS Bhathiya, and HMN Dilum Bandara

Dept. of Computer Science & Engineering, University of Moratuwa, Sri Lanka

Introduction

Autoscaling at PaaS Level

- Reactive Nature
  - Inability to adapt to complex workload patterns
  - Not considering the execution time (Spin Up/Down time)
- Rule-Based System
  - Event triggered system to manage on-demand resources
- User Involvement
  - User is required to have a deeper understanding of the domain to utilize the auto scaling mechanism, define rules, and threshold

Research Goal

Autoscale computing resources in a PaaS cloud environment based on current workload and resource usage, while predicting the workload to reduce cost and meet desired QoS/SLA goals.

Research Contributions

- Performance study of an existing auto scaler
- A workload forecasting technique
- A penalty-based proactive scaling method
- Evaluation of feasibility of smart killing on various IaaS providers
- Analysis of different combinations of scaling methods
- Implementation of proposed combination on Apache Stratos and AWS

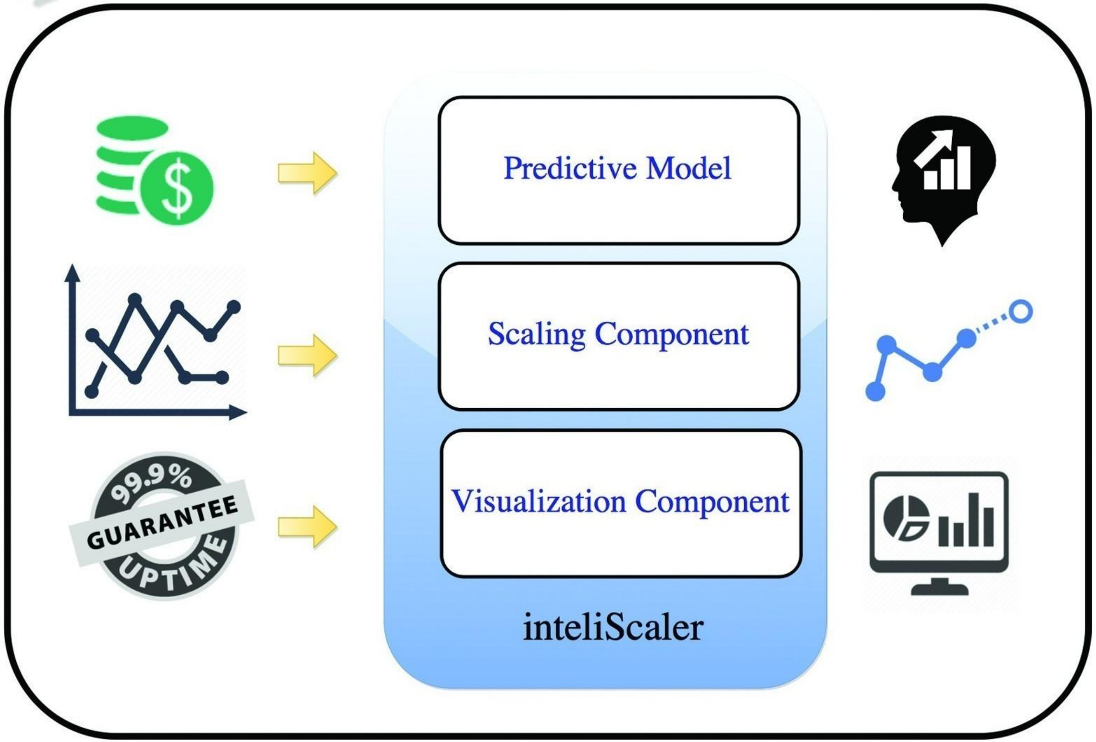
Our Solution

Key features

- Proactive
- Better predictive ability
- Better resource utilization
- Considering resource acquisition costs and service degradation penalties

Evaluation

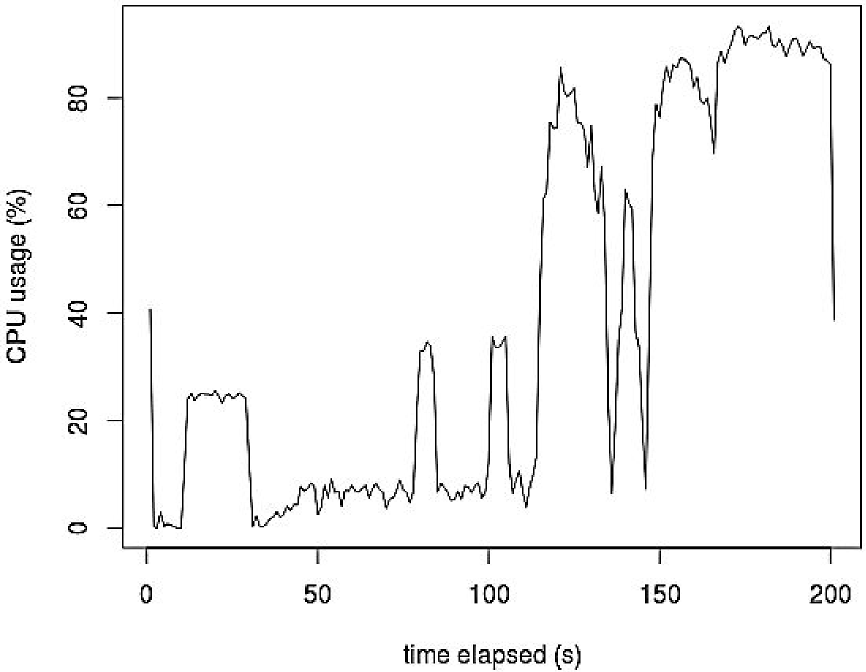
- Against Apache Stratos



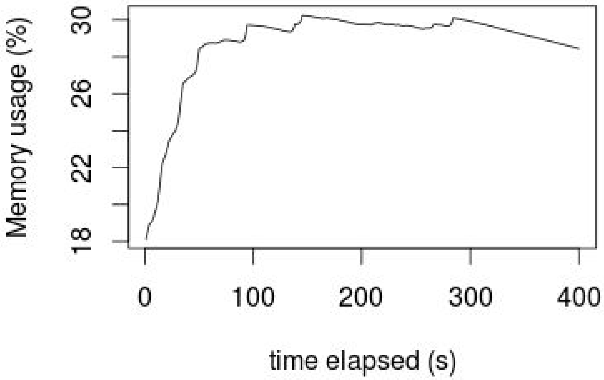
Analysis of Stratos
Auto Scaling Performance

Workload

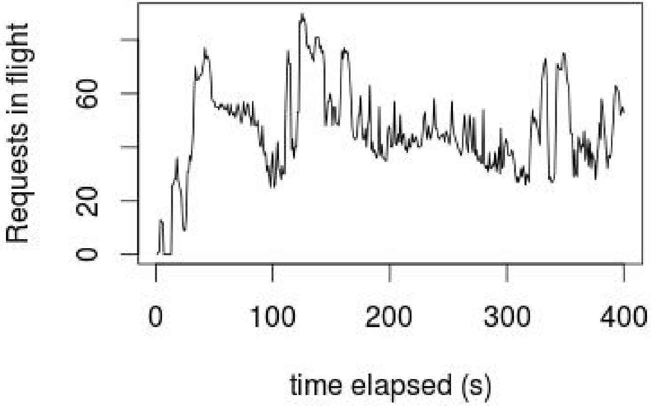
RUBiS Workload



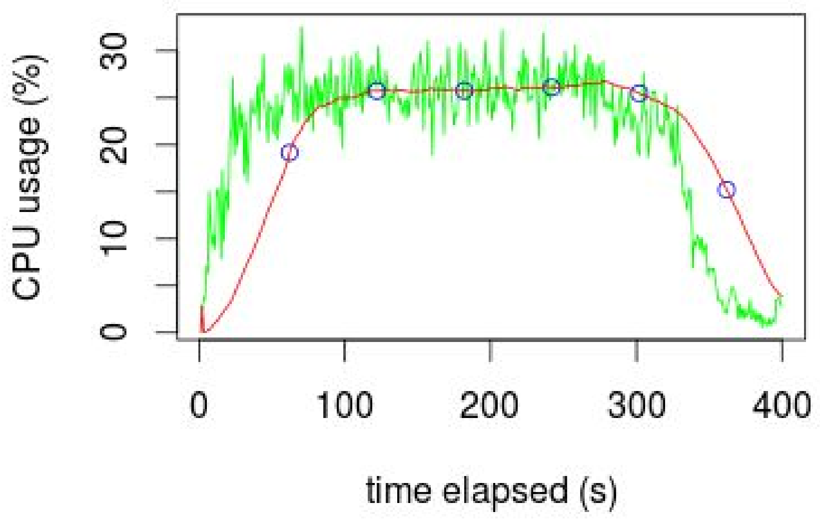
RUBiS Workload

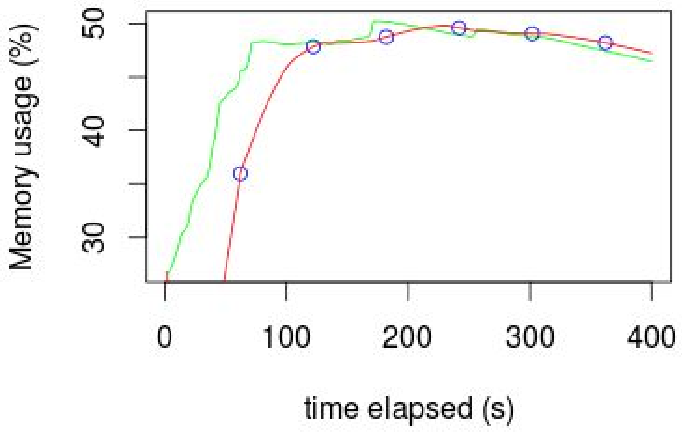


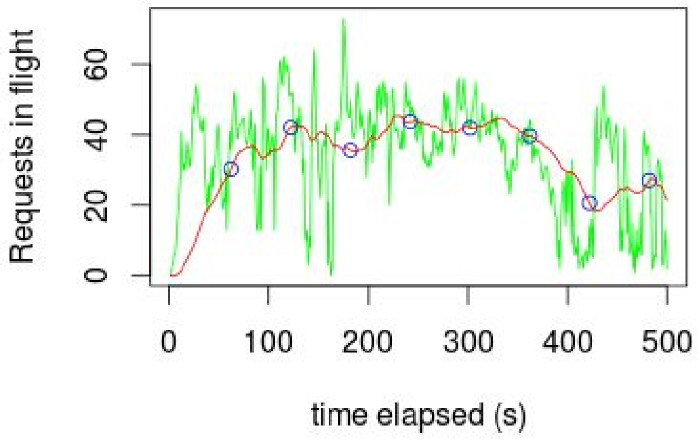
Moraspirit Workload



Prediction Evaluation





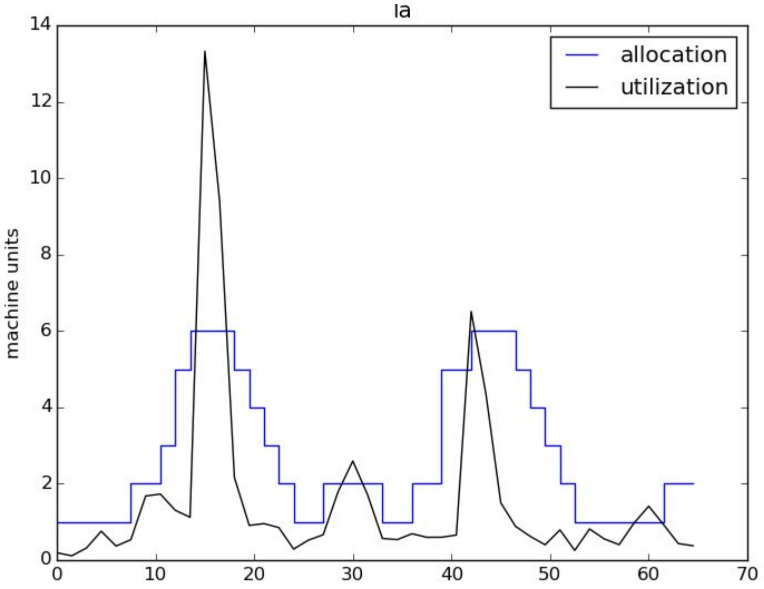


Actual workload

Apache Stratos prediction using $S = ut + 0.5at^2$

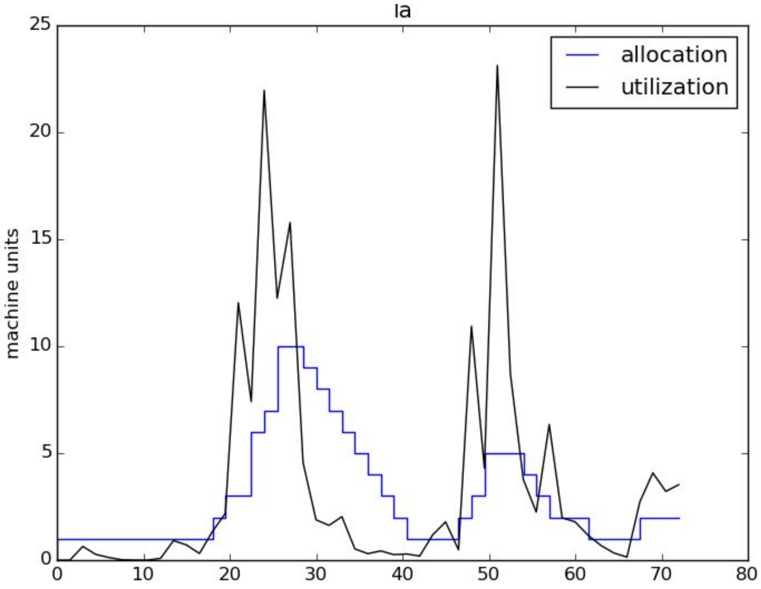
Resource Comparison

High Threshold



Total Instances Spawned = 12

Low Threshold



Total Instances Spawned = 14

Actual Workload   Resource Allocation

Workload Prediction

Base Model Selection

- **ARIMA** - Linear model with ability to capture seasonal behavior
- **Neural Network** - Nonlinear model which is data driven and adaptable
- **Exponential Model** - Non linear exponential models have no counterparts in ARIMA
- **Naive Prediction** - As an error correction step

Ensemble Technique

- Ensemble forecast for (t+h) is weighted average over the individual models
- Weights are calculated from the inverse error of the forecasts.

$$\hat{x}_{(t+h)} = \frac{\sum_{i=1}^{k} c_i \hat{x}^{(i)}_{(t+h)}}{\sum_{i=1}^{k} c_i} \qquad c_i = \frac{1}{e_{(i,t)}}$$

E.g.:

$$\hat{y}_t = \frac{\alpha y_{t,arima} + \beta y_{t,exp} + \gamma y_{t,nnet}}{\alpha + \beta + \gamma}$$

$$\alpha = \frac{1}{error_{T,arima}} \qquad \beta = \frac{1}{error_{T,exp}} \qquad \gamma = \frac{1}{error_{T,nnet}}$$

Determination of Weights

Effectiveness of C_i depends on how you use the past forecast error from the *i*-th model

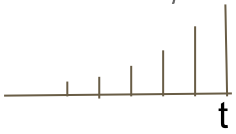
Averaged Error over past data

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{x}_t - x_t)^2}{n}}$$

Most recent error

$$SE_t = (\hat{x}_t - x_t)^2$$

We fitted the past forecasting error in *i*-th model using exponential smoothing model and calculated the contribution coefficients C_i based on the result



Comparison of Results

Model	Google Cluster		Memory		CPU	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ARIMA	12.963	0.051	7.238	0.136	2.976	0.036
Exponential	12.886	0.041	7.005	0.160	3.150	0.048
Neural net.	12.530	0.036	8.169	0.135	**2.792**	0.031
Stratos	19.757	0.116	9.928	0.172	5.692	0.024
ARMA-based model	12.549	0.069	7.185	0.180	3.477	**0.023**
Mean Ensemble	12.099	0.051	7.036	0.130	2.900	0.029
Median Ensemble	12.059	0.055	7.010	0.141	2.944	0.028
Proposed Ensemble	**11.934**	**0.027**	**6.972**	**0.129**	2.873	0.027

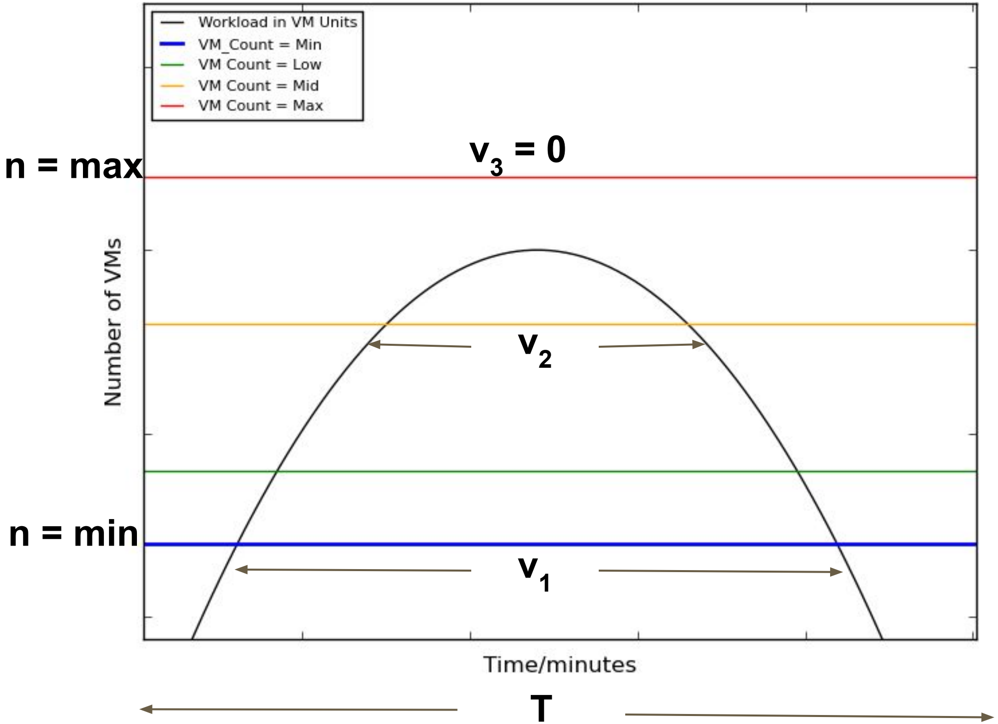
Resource Allocation

Scaling Algorithm

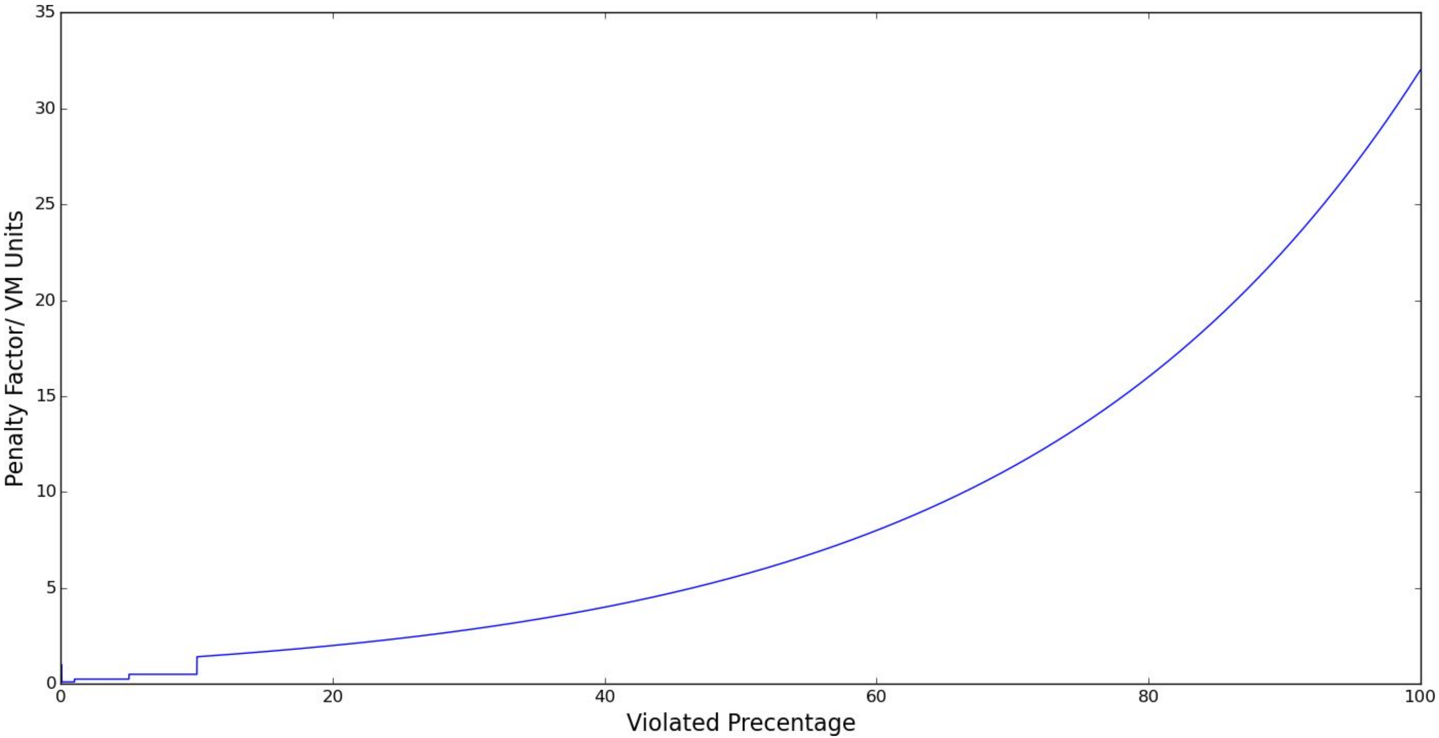
$$C_t(n) = C_{ins}.n + C_{ins}.n.\,f(\frac{v_i}{T}.100)$$

$$n_{opt} = argmin_{n \in N \wedge n \in [min,max]} C_t(n)$$

$$\begin{aligned}
T &= Total\ time\ for\ prediction \\
C_t(n) &= Total\ cost\ for\ time\ T \\
C_{ins} &= Cost\ for\ an\ instance \\
n &= Number\ of\ instances \\
v_i &= Violation\ time
\end{aligned}$$



Scaling Algorithm: Example Penalty Function



Awareness of Resource Pricing Model

Considerations

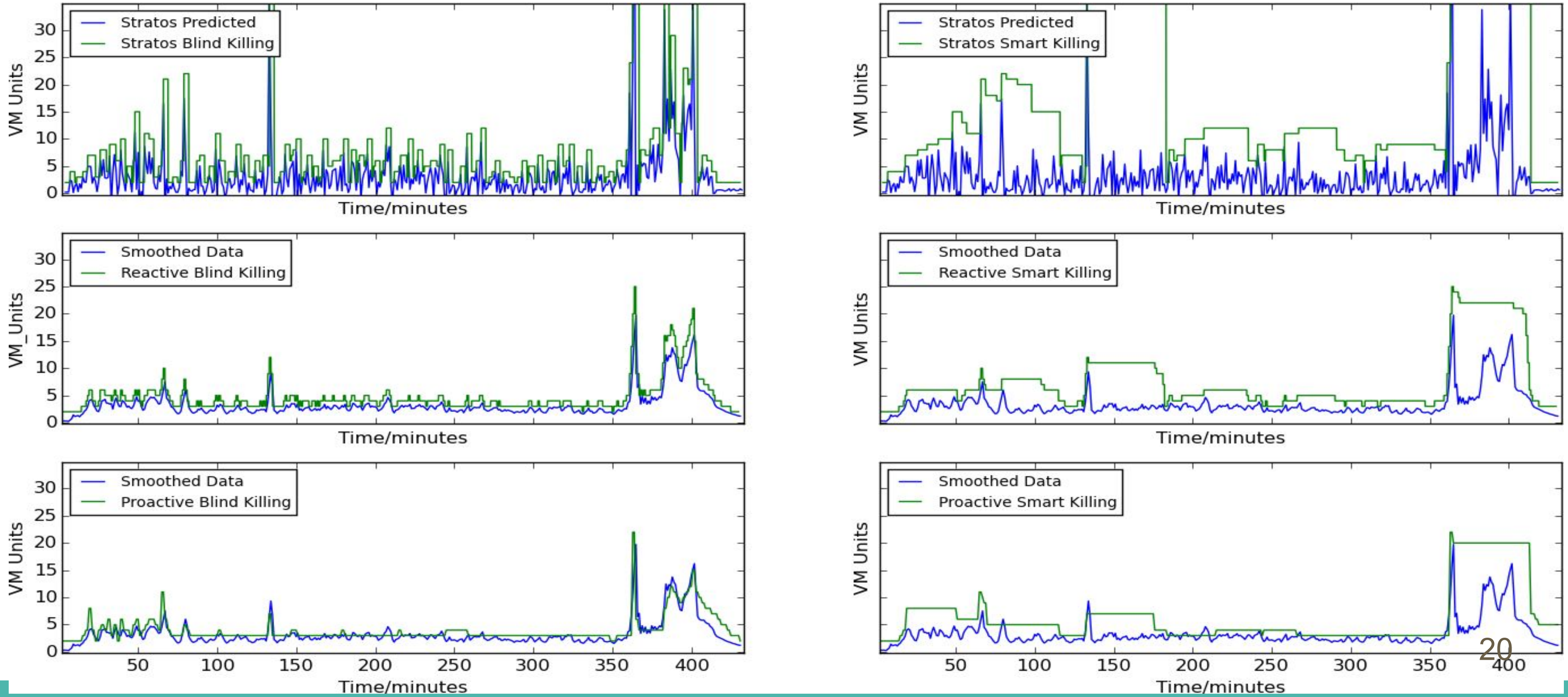
- Each IaaS provides many different instance types
- Non-uniform pricing policies
  - AWS hourly billing vs. GCE per-minute billing

Solution

- Cost optimize considering IaaS pricing model separately
  - Smart Killing for AWS

Resource Scaling Flow

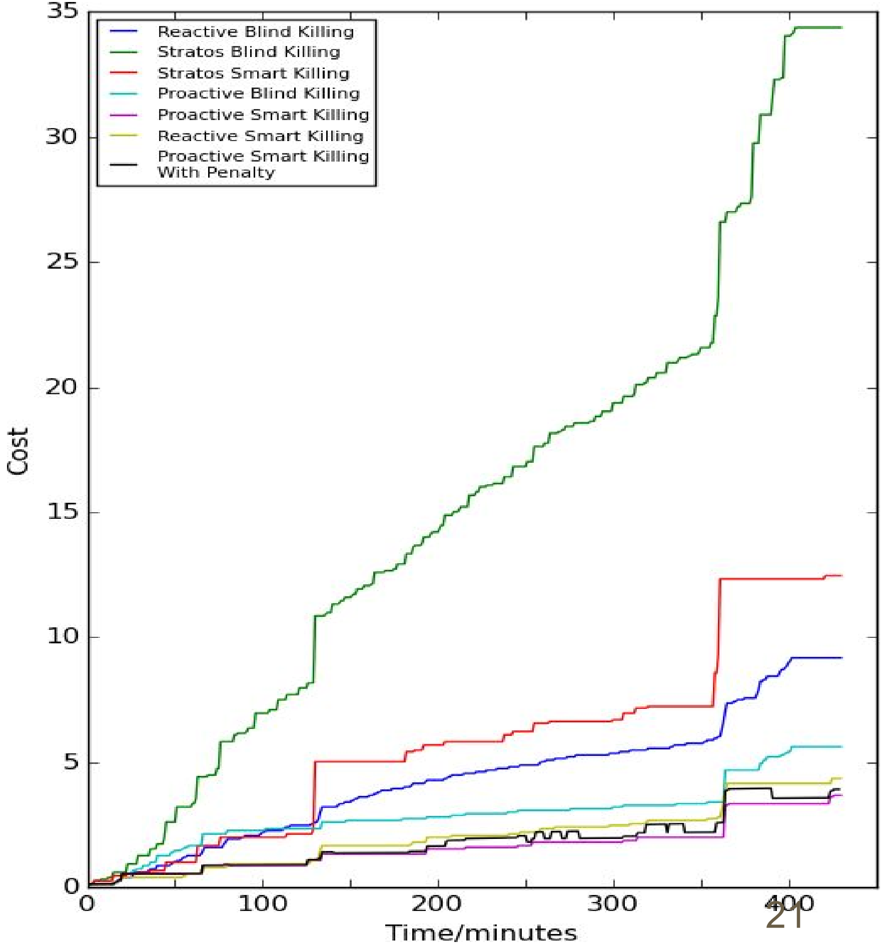
Resource Scaling - Approaches



Simulation - Cost

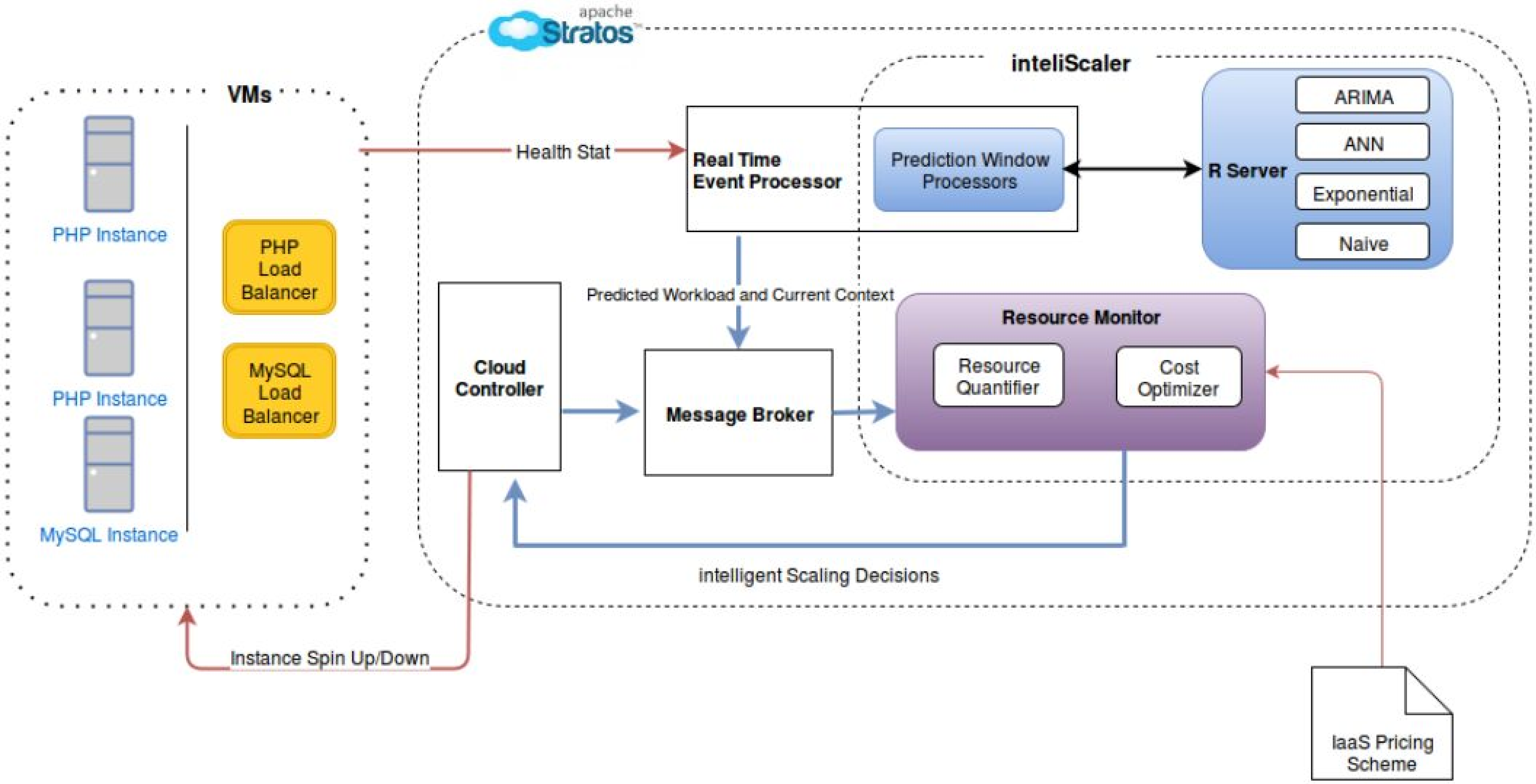
Model	Blind Killing	Smart Killing
Stratos Prediction	34.29	12.35
Reactive	9.14	4.13
Proactive	5.53	3.25

Proactive Smart Killing
Violation Percentage After 7 Hours : 9.26%
Violation Cost After 7 Hours: 0.235



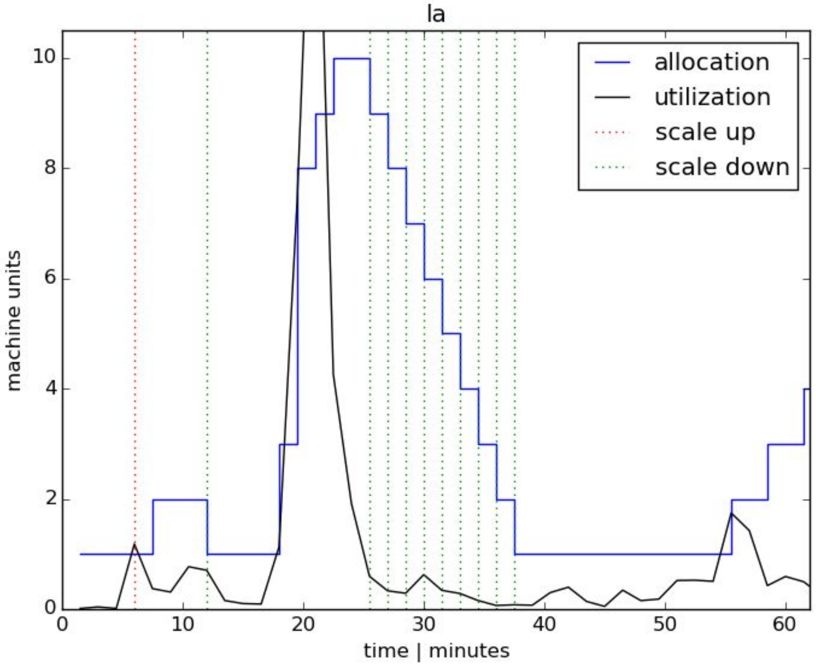
Overall Solution

Architectural Design

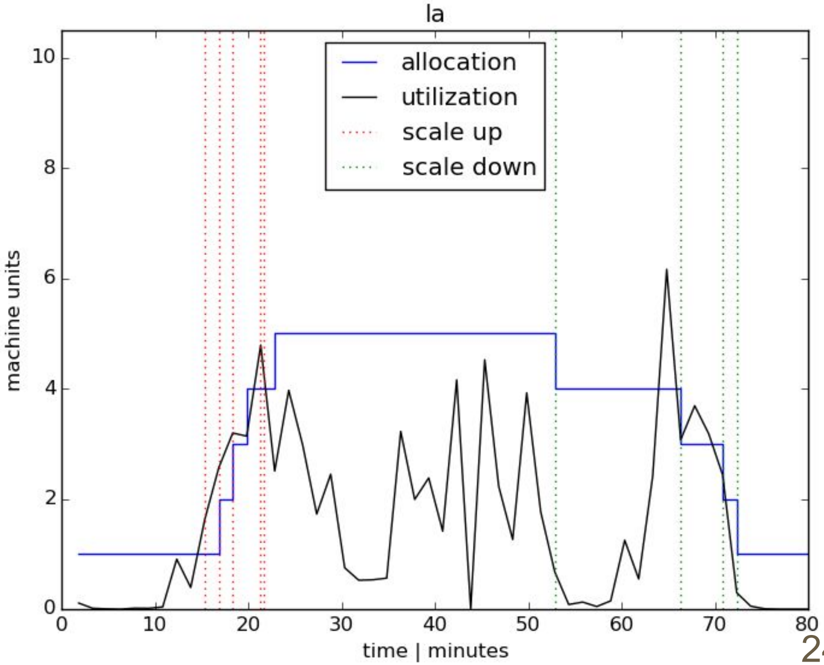


Tests on AWS EC2

Stratos



inteliScaler



Summary

Performance Evaluation

- Current auto scaler implementation in Apache Stratos

Workload Prediction

- Evaluating current workload prediction techniques
- Proposing a new ensemble workload prediction technique for PaaS

Resource Allocation

- Comparison of different combinations of resource allocation methods
- Proposing a penalty-based resource allocation technique

Overall Solution

- Implementation and testing on Apache Stratos

Further Improvement

- Defining different penalty functions based on required QoS
- Bidding for Spot Instances on AWS
- Inclusion of a Performance Model
- Support for Heterogeneity

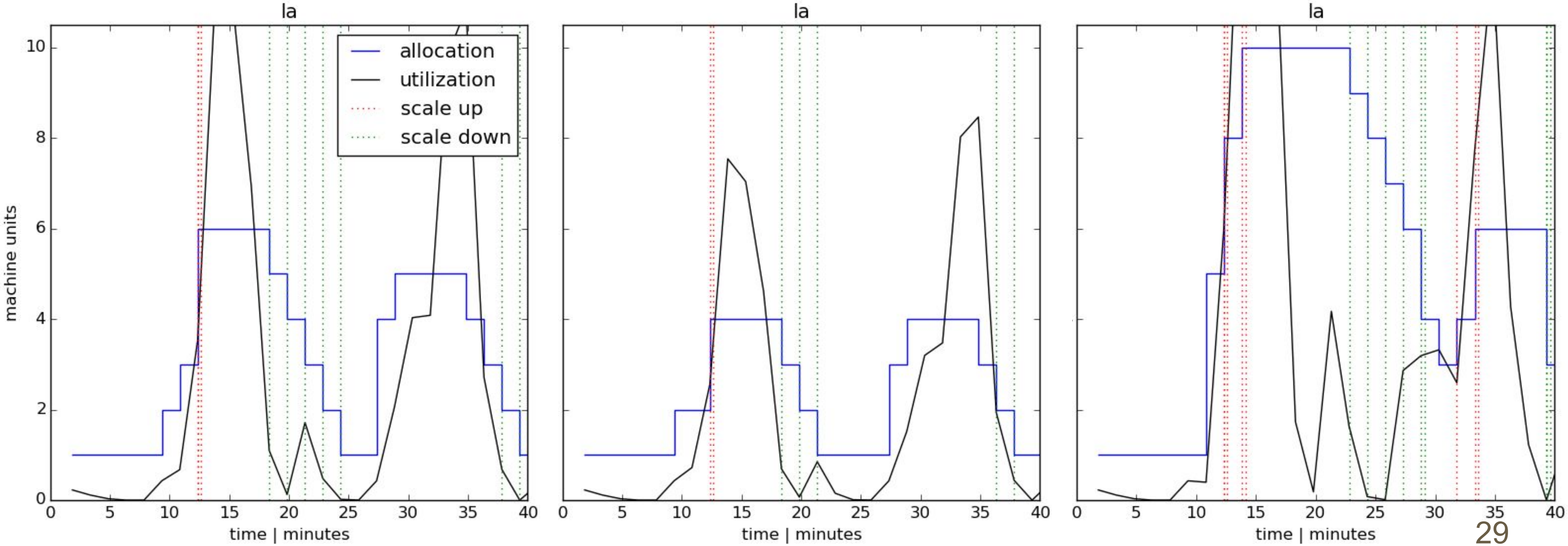
Q & A

ridwan.11@cse.mrt.ac.lk

Thank You

Tests on Mock IaaS

Stratos - Low Threshold **Stratos - High Threshold** **inteliScaler**



Upcoming Sections

Performance Evaluation

- Current auto scaler implementation in Apache Stratos

Workload Prediction

- Evaluating current workload prediction techniques
- Proposing a new ensemble workload prediction technique for PaaS

Resource Allocation

- Comparison of different combinations of resource allocation methods
- Proposing a penalty-based resource allocation technique

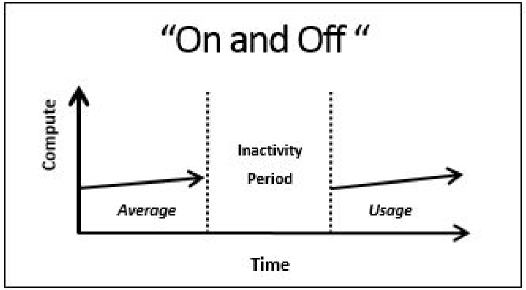
Overall Solution

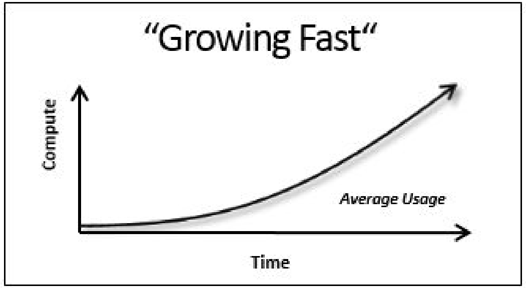
- Implementation and testing on Apache Stratos

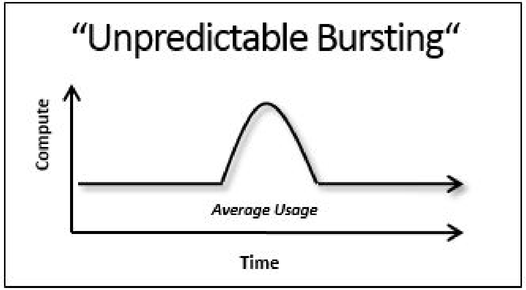
Project Summary

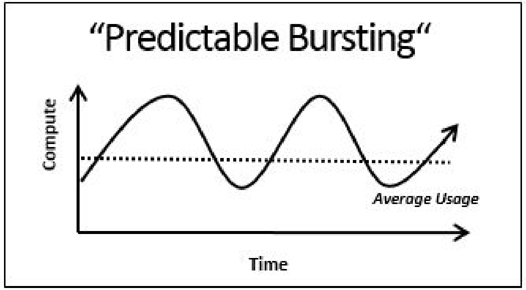
Workload Types

- Nature
  - Synthetic
  - Empirical
- Pattern
  - On-and-Off
  - Growing
  - Stable
  - Sudden Peak
  - Cyclic Bursting









Why Simulation?

- Clouds exhibit varying demands, supply patterns, system sizes and computing resources.
- Cloud users have heterogeneous, dynamic and competing QoS requirements.
- Cloud applications have varying performance, workload, and dynamic application scaling requirements.

Experimental Platforms

- CloudSim
- MDCSim
- GreenCloud

Experimental Workloads

- 98 World Cup
- Google Cluster Data
- ClarkNet

Synthetic Workload Generators

- Httperf
- Faban
- Rain

Application Benchmarks

- RUBiS
- RUBBoS
- TPC-W

Weight Calculation

$$b_{i,t} = \alpha e_{(i,t)} + (1-\alpha) b_{(i,t-1)} \qquad (10)$$

where $0 \leq \alpha \leq 1$

$$b_{i,t} = \alpha e_{(i,t)} + \alpha(1-\alpha) e_{(i,t-1)} + \alpha(1-\alpha)^2 e_{(i,t-2)} + \ldots$$

$$c_{i,t} = \frac{1}{b_{i,t}} \qquad (11)$$

Research Overview

Contributions

- Performance study of an existing auto scaler
- A workload forecasting technique
- A penalty based proactive scaling method
- Evaluation of feasibility of smart killing on various IaaS providers
- Evaluation of various combinations of scaling methods
- Implementation of proposed combination on Apache Stratos

Achievements

Research Papers

- Acceptance of *"Adaptive Workload Prediction for Proactive Auto Scaling in PaaS Systems"* for International Conference - Internet of Things and Cloud Computing Technologies, Singapore
- Working on another paper for International Cloud Computing Conference - IEEE Cloud 2016, USA

Research Grant

AWS Education Research Grant for 1 year

Apache Stratos

A PaaS framework

- Can be set up on existing IaaS
- AWS, GCE, Azure
- Private cloud support

Why Stratos?

- Open source
- Apache Community support
- Mock IaaS - eliminates need and cost of using actual IaaS resources

Architecture

- Services via cartridge instances (PHP, MySQL, Tomcat, etc.)
- Managed by modules (Cloud Controller, Auto Scaler, etc.)
- Pub-sub messaging

Resource Allocation

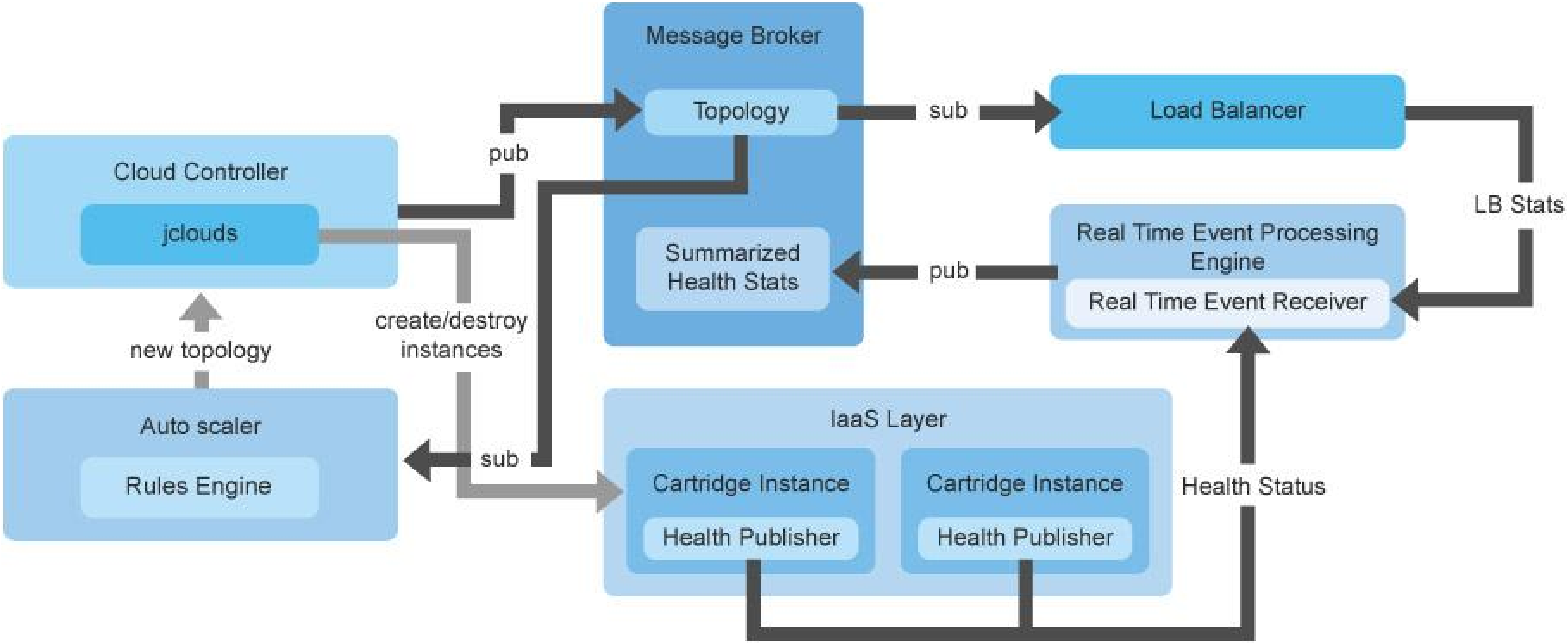
Current PaaS Auto Scalers

- User need to define threshold parameters
- Autoscaler unaware of the pricing models of IaaS

Inteliscacler

- Cost and QoS aware scaling algorithm to calculate resource count required
- Pricing model aware resource scaling

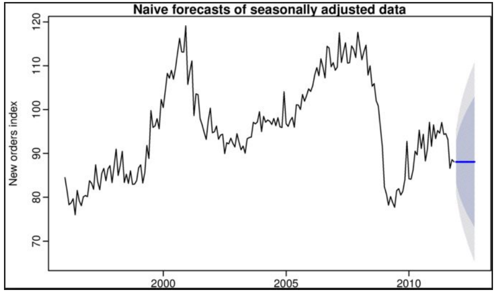
Stratos Auto Scaling



Literature - Prediction Techniques : Statistical

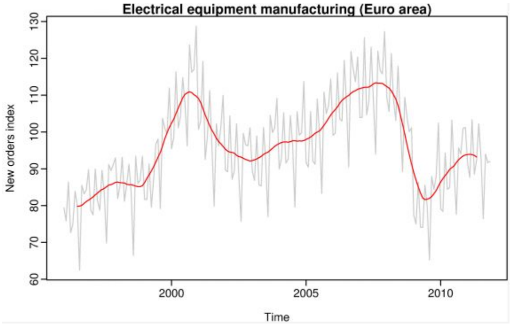
Naïve Prediction

$$\hat{y}_{T+h|T} = y_T$$



Moving Average

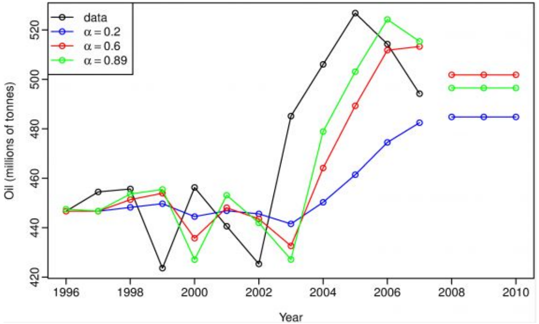
$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t$$



Exponential Smoothing

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}$$

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} + \cdots,$$



Literature - Prediction Techniques : Statistical

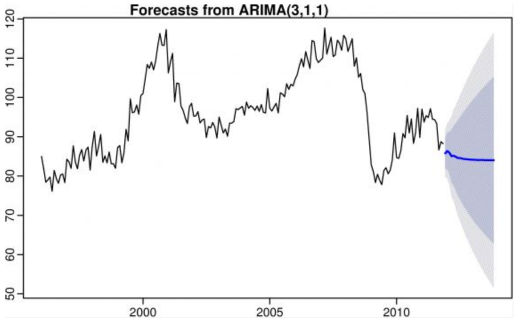
Autoregressive Integrated Moving Average Model(p,d,q)

A combination of ,

Differencing(d) $y'_t = y_t - y_{t-1}$

Autoregression(p) $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$

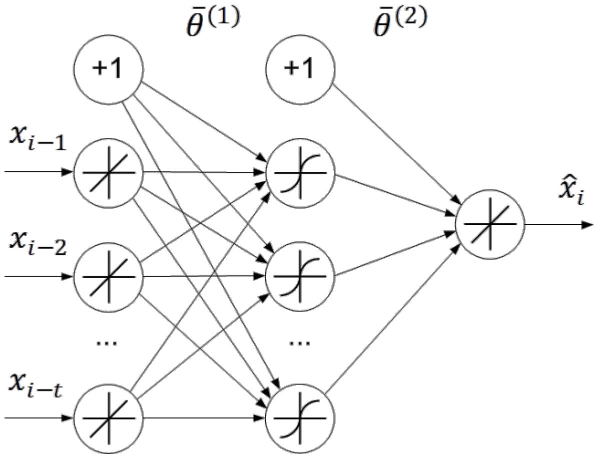
Moving Average(q) $y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$

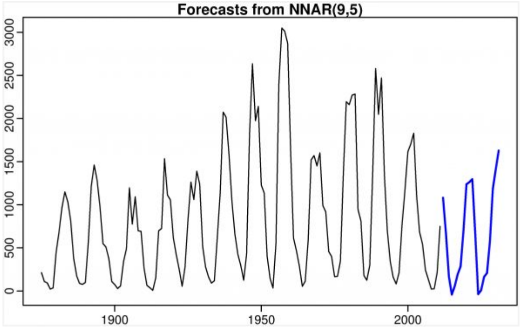


Literature - Prediction Techniques : Machine Learning

Autoregressive Neural Networks

Lagged values of the time series can be used as inputs to a neural network.

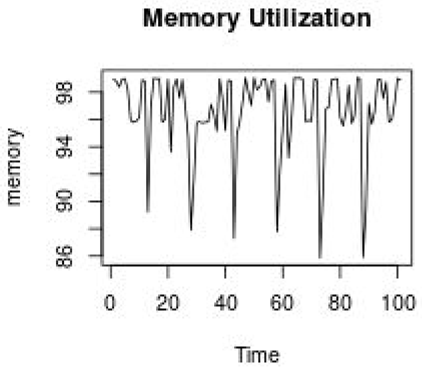


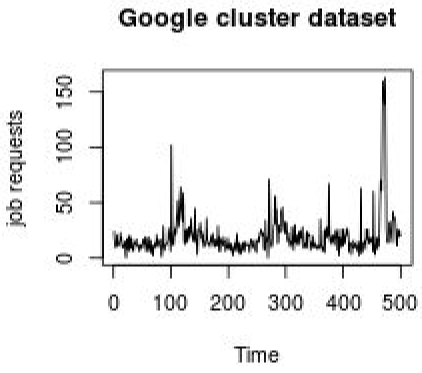


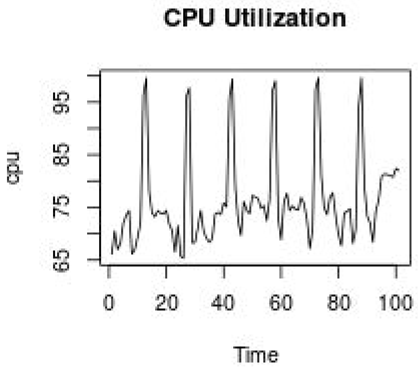
Challenges In PaaS Workload Prediction

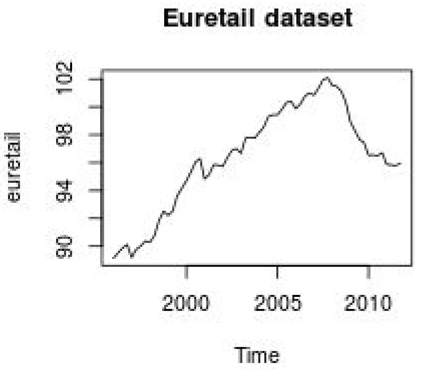
- A PaaS cloud may be used to build different applications with vastly different workload patterns.
- As the workload dataset grows with time, predictive model should evolve and continuously learn the latest workload characteristics.
- The workload predictor should produce results within a bounded time.
- The predictor should produce sufficiently accurate results over a sufficiently large time horizon.

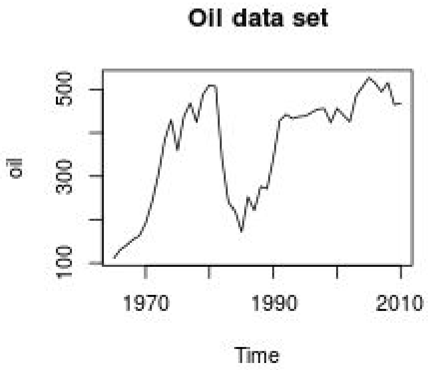
Evaluation of Existing Models - Datasets

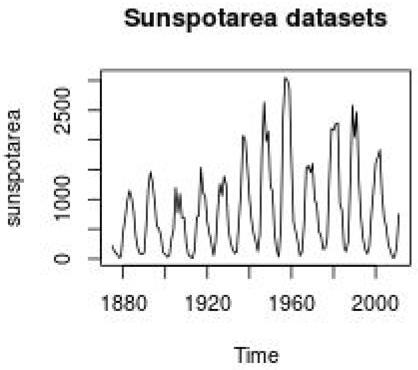












Evaluation of Existing Models - Results

Model/ Dataset	sunspotarea		euretail		oil		Memory		CPU	
	MSE	MARE	MSE	MARE	MSE	MARE	MSE	MARE	MSE	MAPE
Arima	**382.360**	1.359	**0.524**	**0.004**	55.313	0.251	7.599	0.075	2.976	0.036
Exponential	505.750	1.161	0.576	0.011	54.989	0.251	8.869	**0.024**	3.150	0.048
Neural Network	473.924	**0.465**	1.882	0.006	**51.616**	**0.160**	**7.136**	0.153	**2.792**	0.031
Stratos	546.938	0.965	0.650	**0.004**	61.807	0.585	11.736	0.046	5.692	**0.024**

- Each model performs well on some datasets while performing worse on other datasets
- A single model will not perform well in all online training scenarios

BUT,

- Different applications on PaaS have different workload characteristics
- Dynamic adjustment of models is required for robust results

Our Simulator

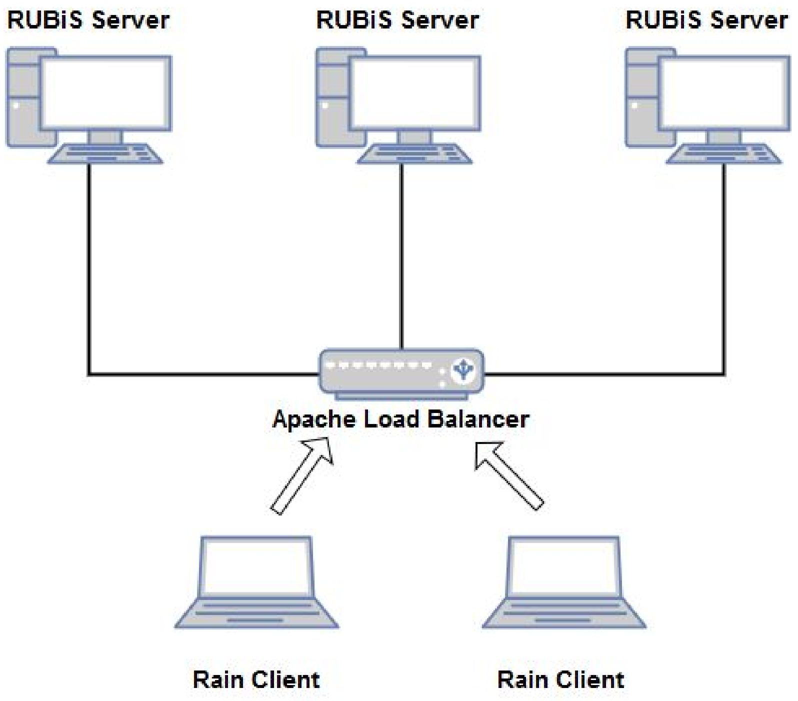
- Workloads
  - Synthetic
    - Rain-Toolkit
    - Httperf
  - Empirical
    - Sports website (Moraspirit Log)
    - 98 World Cup Trace
    - Rackspace Cloud Data (Hosting a HR System)
    - Google Cluster Data
- Platform
  - 4 Server Nodes
  - Apache Stratos 4.1 setup on top of AWS
- Benchmark
  - RUBiS

10	90000	981883546	1488529849	3	0.003125	8.38861e-02
11	90000	1263655469	1488529853	2	0.000000	0.00000e+00
12	90000	757745334	1488529860	0	0.000000	0.00000e+00
13	90000	1487094655	1488529866	0	0.003125	3.82931e-03
14	90000	1458411965	1488529868	0	0.000000	4.02513e-04
15	90000	1390006181	1488529872	1	0.000000	0.00000e+00
16	90000	1164728954	1488529877	0	0.003125	1.63840e-03
17	90000	1288997448	1488529880	0	0.012500	9.92832e-04
18	90000	1433362210	1488529881	0	0.000000	2.45760e-02
19	90000	1263655469	1488529882	2	0.015625	2.51658e-02
20	90000	1288997448	1488529889	0	0.003125	4.91520e-03
21	90000	1488529887	1488529894	2	0.000000	0.00000e+00
22	90000	1213243701	1488529902	0	0.000000	3.36337e-04
23	90000	1164728954	1488529904	0	0.000000	0.00000e+00
24	90000	1488529890	1488529897	0	0.000000	0.00000e+00
25	90000	1488529890	1488529900	0	0.003125	2.08077e-03
26	90000	975992247	1488529906	0	0.015625	1.63840e-03
27	90000	1488529907	1488529909	2	0.000000	8.15488e-04
28	90000	757745334	1488529915	0	0.000000	3.11296e-02

isplayed 1000 rows of 3,535,029 (3,534,029 omitted)

Google Cluster Data

Experimental Setup



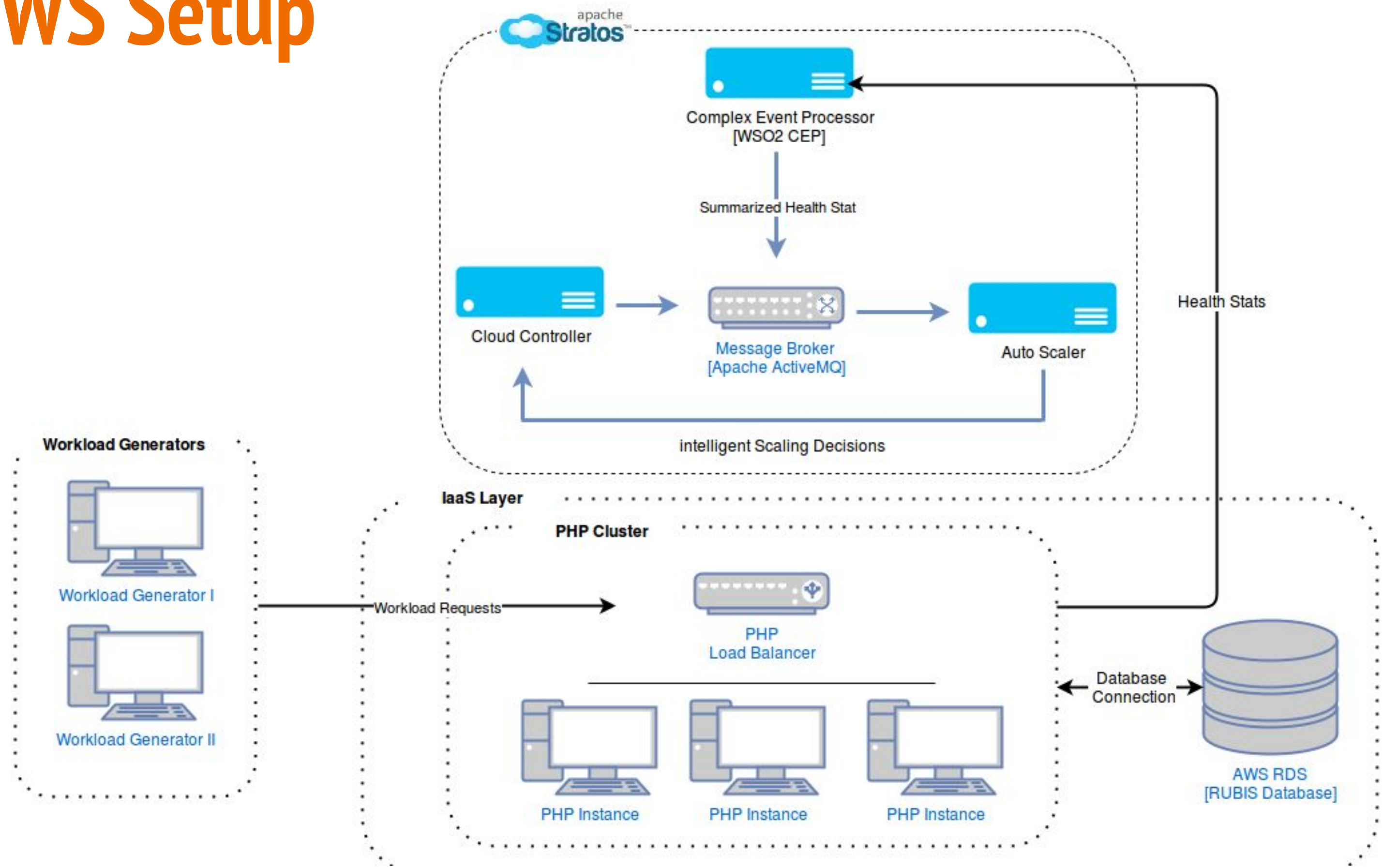
Server Node Specification

- Intel Core i3 1.6 GHz
- 4 GB DRR3 RAM
- 500 GB Hard Disk
- MySQL 5.5.
- Apache 2.4
- PHP 5.5

Client Specification

- Intel Core i5 2.6 GHz
- 4 GB DRR3 RAM
- Rain Tool Kit
- Java 1.7

AWS Setup


apache Stratos
Complex Event Processor
[WSO2 CEP]
Summarized Health Stat
Cloud Controller
Message Broker
[Apache ActiveMQ]
Auto Scaler
Health Stats
intelligent Scaling Decisions
Workload Generators
Workload Generator I
Workload Generator II
IaaS Layer
PHP Cluster
Workload Requests
PHP
Load Balancer
Database
Connection
PHP Instance
PHP Instance
PHP Instance
AWS RDS
[RUBIS Database]

Stratos Prediction

$$s = ut + 1/2\ at^2$$

- s = predicted load
- u = first derivative (eg. first derivative of current load average)
- t = time interval (eg. one minute)
- a = second derivative (eg. second derivative of current load average)

Stratos Scaling Process

- Policies
  - Application Deployment Policy
  - Auto-Scaling Policy
    - Load Average Threshold
    - Memory Consumption Threshold
    - Requests in Flight Threshold

Project Summary

Summary

Performance Evaluation

- Current auto scaler implementation in Apache Stratos

Workload Prediction

- Evaluating current workload prediction techniques
- Proposing a new ensemble workload prediction technique for PaaS

Resource Allocation

- Comparison of different combinations of resource allocation methods
- Proposing a penalty-based resource allocation technique

Overall Solution

- Implementation and testing on Apache Stratos

Challenges Faced

- Resource and Infrastructure Cost
- Insufficient sources for Workload Data
- Setting up Apache Stratos Paas Cloud on AWS EC2

Key References

N. Roy, A. Dubey, and A. Gokhale, "Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting", in proc. Intl. Conf. on Cloud Computing (CLOUD), Washington, DC, July 2011.

H. Alipour, Y. Liu, and A. Hamou-Lhadj, "Analyzing Auto-scaling Issues in Cloud Environments," in proc. 24th Annual Intl. Conf. on Computer Science and Software Engineering, Markham, Ontario, Canada, Nov. 2014.

C. Bunch, V. Arora, N. Chohan, C. Krintz, S. Hegde, and A. Srivastava, "A Pluggable Autoscaling Service for Open Cloud PaaS Systems," in proc. 5th IEEE Intl. Conf. on Utility and Cloud Computing (UCC), Chicago, IL, Nov. 2012.

J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload Predicting-Based Automatic Scaling in Service Clouds," in proc. 6th IEEE Intl. Conf. on Cloud Computing, Santa Clara, CA, Jul. 2013.

G.F.S. Garg, R. Buyya, and W. Li, "Revenue Maximization Using Adaptive Resource Provisioning in Cloud Computing Environments," in proc. ACM/IEEE 13th Intl. Conf. on Grid Computing (GRID), Beijing, China, 2012.

M. Mao and M. Humphrey, "Auto-scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," in proc. Intl. Conf. for High Performance Computing, Networking, Storage and Analysis, New York, NY, 2011.