# *Integrated and Autonomic Cloud Resource Scaling*

Masum Z. Hasan, Edgar Magana, Alexander Clemm, Lew Tucker
Cisco Systems
San Jose, CA USA
masum, eperdomo, alex, letucker{@cisco.com}

Sree Lakshmi D. Gudreddi
Dept. of Computer Engineering
Santa Clara University
sgudreddi@scu.edu

*Abstract*—**A Cloud is a very dynamic environment where resources offered by a Cloud Service Provider (CSP), out of one or more Cloud Data Centers (DCs) are acquired or released (by an enterprise (tenant) on-demand and at any scale. Typically a tenant will use Cloud service interfaces to acquire or release resources directly. This process can be automated by a CSP by providing auto-scaling capability where a tenant sets *policies* indicating under what condition resources should be auto-scaled. This is specially needed in a Cloud environment because of the huge scale at which a Cloud operates. Typical solutions are naïve causing spurious auto-scaling decisions. For example, they are based on only thresholding triggers and the thresholding mechanisms themselves are not *Cloud-ready*. In a Cloud, resources from three separate domains, compute, storage and network, are acquired or released on-demand. But in typical solutions resources from these three domains are not auto-scaled in an integrated fashion. Integrated auto-scaling prevents further spurious scaling and reduces the number of auto-scaling systems to be supported in a Cloud management system. In addition, network resources typically are not auto-scaled. In this paper we describe a Cloud resource auto-scaling system that addresses and overcomes above limitations.**

***Keywords: cloud computing; Cloud resource scaling; performance metrics, virtualized resources, autonomic scaling, integrated compute, storage and network domain scaling***

## I. Introduction

One of the major features of a Cloud is to acquire/release resources at any scale. Typically a Cloud Service Provider (CSP) offers interfaces so that a Cloud Service Consumer (CSC or tenant) can acquire/release resources anytime and on-demand. While tenants can acquire/release resources via automated scripts or programs by utilizing existing interfaces (or API: Application Programming Interfaces), it can be cumbersome and not scalable itself. Automation is a major trait of Cloud where a CSP Cloud management system automates and orchestrates acquisition or release of resources when a tenant invokes the relevant interfaces. The automation can be extended to other functions of Cloud, such as monitoring and Cloud resource scaling. In this paper we focus on the latter.

Typical Cloud resource scaling solutions possess following limitations:

1. Thresholding mechanisms used to trigger scaling decision process are not Cloud-ready. That is, they are not suitable for huge scale and dynamic Cloud environment. They are naïve or rudimentary causing spurious scaling of resources.
2. In a Cloud, resources from three different domains, compute, storage and network, are acquired/released (scaled up or down) on demand (typically in virtualized form, including application or software resources). But in current solutions resources from these three domains are scaled in isolation rather than in combination resulting in spurious auto-scaling.
3. The performance metrics that trigger scaling are considered in isolation. For example, compute resources are scaled based on CPU/compute resource metric only, but not in correlation with, for example, metrics from the network domain.
4. While existing solutions may scale compute or storage resources automatically when specified condition or policy is satisfied, network resources are scaled statically.
5. Network resources are not scaled in relation to compute and storage domains and vice versa. For example, when virtual compute resources are scaled, virtual load-balancer (LB) or firewalls (FW) may also need auto-scaling.

In this paper we describe a Cloud resource auto-scaling system called Integrated and Autonomic Cloud Resource Scaler (IACRS) that addresses and overcomes the limitations mentioned above. The IACRS supports Cloud-ready advanced thresholding mechanism, integrates performance metrics from multiple domains in making scaling decisions and scales network resources in combination with resources from the other two domains and vice versa. In this paper we describe in details the algorithms and heuristics that embody the IACRS.

The paper is organized as follows. In Section II we provide an overview of terminologies and major concepts used in IACRS, which includes resources, performance metrics, groups, metrics thresholding, metrics correlation, and network resource auto-scaling. The main algorithms are presented in Section III, which includes tenant interface input, thresholding heuristics, metric correlation heuristics, and group heuristics. A number of use cases are provided in Section IV. Comparisons with related works are discussed in Section V. We conclude in Section VI.

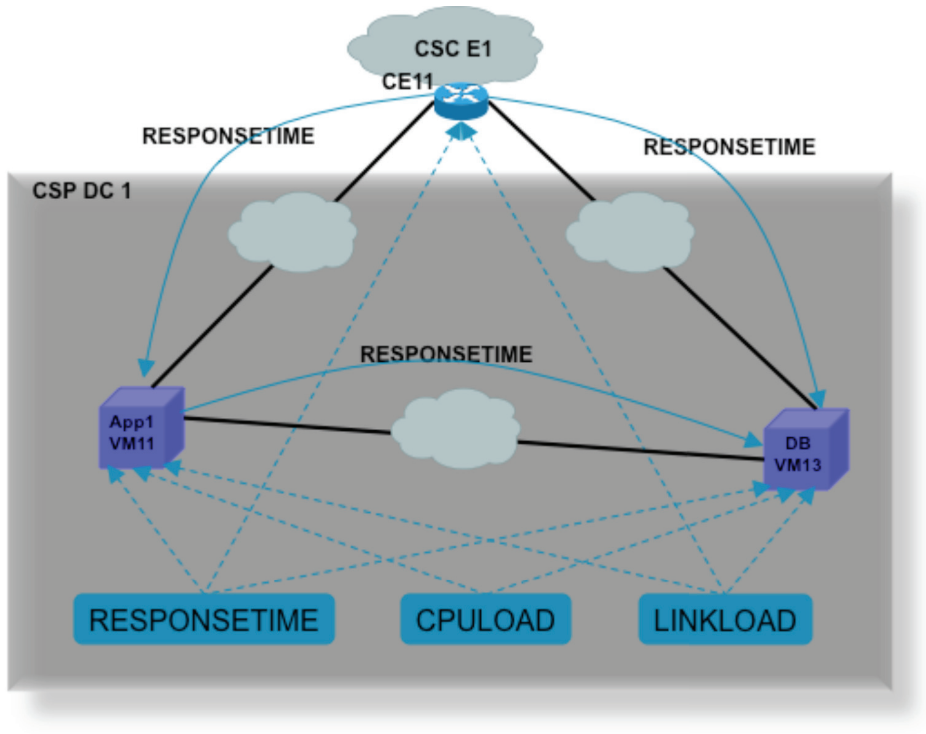

Figure 1. Multi-domain, multi-metric Correlation Graph

## II. IACRS Background

Before we proceed to describe the IACRS algorithms, we provide brief introduction to a number of concepts or elements used in the IACRS, viz., resources, performance metrics (metrics in short), groups, metrics thresholding, metrics correlation, and network resource auto-scaling.

### A. *Resources*

Resources are any Cloud or virtualized compute, storage and network resources. While virtualized resources are prevalent in a Cloud environment, physical resources can also be auto-scaled. Examples of resources are a VM, storage unit, link bandwidth, virtual load-balancer (LB), virtual firewall (FW), and virtual switch. A resource may have associated sub-resources, such as VM1.CPU1 or CE1.link1. Note that Cloud resource modeling is beyond the scope of this paper. Any management resource model can be used.

### B. *Performance Metrics*

A resource may have multiple performance metrics associated with it. A few examples of metrics are CPULOAD (CL), network link load (LL), response time (RT) from one end-point to another, delay (DL) and jitter (JT) (such as those supported in Ganglia [1], NetFlow [2], Cisco IPSLA [3] and NAM [4]). Metrics specified in auto-scaling policies are monitored to decide when to scale resources up or down.

### C. *Groups*

Resources and metrics can be grouped together so that scaling policies can be applied on the group in aggregate to control the scaling process. In IACRS following groups can be defined:

- Multiple metrics of a resource. For example, CL and LL of a VM or LL of a router link and RT to it.
- Instances of a particular resource type. For example, a set of compute resources, such as VMs.
- A set of compute resources under the control of a load-balancer.
- A set of network, compute and storage resources under the control of a firewall.

It is expected that the user create groups initially. A hierarchical group can be formed. For example, a load-balancer can be a parent to a set of compute resources. In the current solution we assume only one parent and one level in the hierarchy. The process of group creation is beyond the scope of this paper. Any management resource aggregation model can be used.

### D. *Thresholding*

Typically resource scaling is based on simple threshold policies, including hysteresis mechanism [5], which we believe is a naïve approach to be applied in Cloud resource auto-scaling. The huge scale (with many millions of resources) at which a Cloud operates dictates tighter control on how resources are scaled up or down. In this work we propose a Cloud-ready and advanced thresholding heuristics that facilitates such tighter control so that resources are auto-scaled more accurately.

### E. *Correlation of Metrics*

Scaling based on a single metric threshold crossing is a naïve approach as is typical in existing solutions. The IACRS supports a more sophisticated thresholding mechanism. In addition, a scaler can be augmented with a mechanism where multiple metrics from multiple domains (compute, storage and network) are integrated or correlated before making a scaling decision. Consider the multi-domain performance metric correlation graph shown in Figure 1. Following are two use cases highlighting scaling based on the correlation graph:

- Acquire (scale up) new compute resource(s) (VM) when both CL of VM11 AND RT to it from a customer edge router, CE11 are high. The assumption is that the high CL may have caused high RT, hence acquire more VM and split load.
- Increase the bandwidth (scale up) of the network link resource of CE11 when both RT (from CE11 to VM11) and LL are high. The assumption is that scaling up the BW (to 10% more, for example) may lower the RT.

### F. *Network Resource Auto-Scaling*

A network resource, such as a virtual load-balancer (LB) may be acquired on-demand and a set of compute resources (VMs) associated with it for load balancing. But typically network resources are not scaled automatically (via a resource scaling system) and in correlation or integration with compute and storage domains.

Consider following situations with respect to an LB, which balances load between a set of compute resources or servers under its control (as shown in Figure 2):

- During peak time or over course of time all the servers under the LB can get overloaded or under-loaded requiring scaling up or down. But an LB is not capable of automatically scaling compute resources under its control. The IACRS will auto-scale compute resources and place them under the control of the LB.
- The LB itself can get overloaded (or under-loaded), in which case the LB itself has to be scaled. The same scaling capabilities of the IACRS applied to compute and storage resources can be applied to scale LB or other network resources. This is shown in Figure 2, where compute and network resources are scaled in combination.

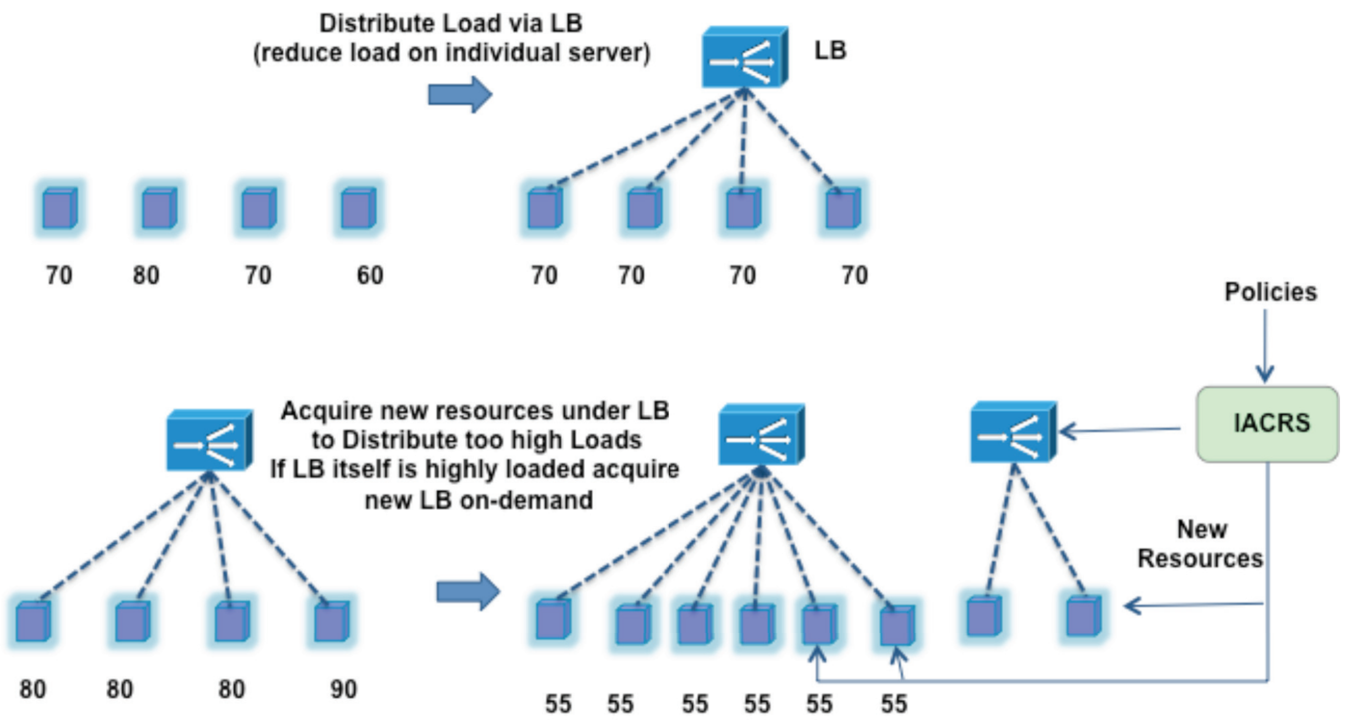
Distribute Load via LB
(reduce load on individual server)
LB
70
80
70
60
70
70
70
70
Policies
IACRS
Acquire new resources under LB
to Distribute too high Loads
If LB itself is highly loaded acquire
new LB on-demand
New
Resources
80
80
80
90
55
55
55
55
55
55

Figure 2. Integrated Compute and Network Resource Auto-scaling

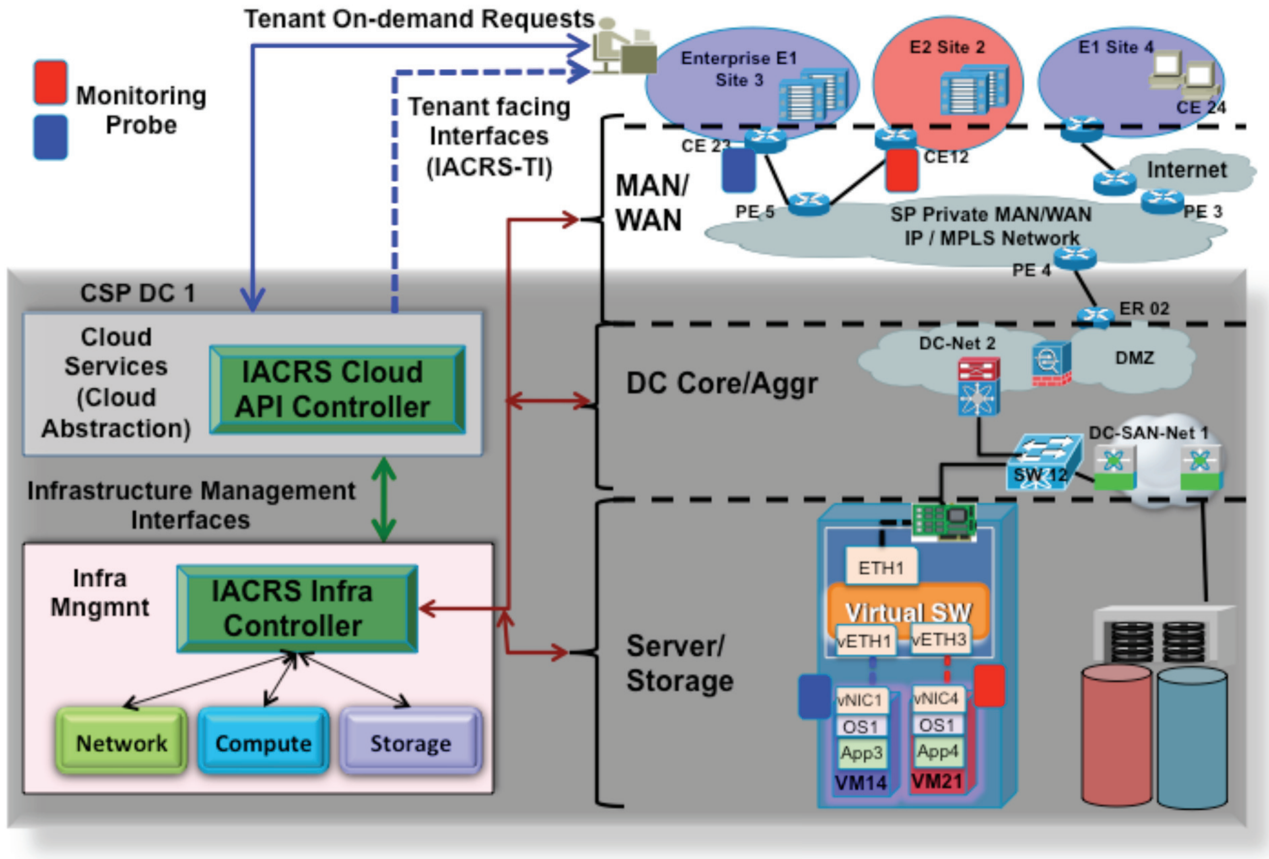

Figure 3. IACRS Cloud Management Architecture

## III. IACRS Algorithm

In this section we describe the IACRS algorithms, which consist of following:

- Advanced Cloud-ready thresholding heuristics.
- Multi-domain and multi-metric integrated correlation heuristics.
- Integrated resource and metric group based heuristics.

Before we proceed to describe the algorithms, we provide a brief overview of Cloud management architecture as it relates to the IACRS. The detail of the management architecture, including interfaces and process of realization (provisioning, configuration and monitoring) mentioned below are outside the scope of this paper. We provide a summarized overview of the IACRS management architecture for completeness, especially to highlight how tenants interact with the IACRS and how IACRS interacts with the Cloud and network infrastructure to realize tenant requests. We refer to "realization" to indicate management functions, such as configuration/provisioning and monitoring.

A high-level conceptual Cloud management architecture is shown on the left hand side and a Cloud and network infrastructure is shown on the right in Figure 3. The management architecture consists of two components:

- Cloud services management: This component or layer is tenant facing providing Cloud abstraction and service interfaces to tenants. As is typical in a Cloud, a tenant requests resources on-demand via published interfaces. In the same vain, the IACRS also provides tenant-facing interfaces (IACRS-TI) so that tenants can request auto-scaling of resources via those interfaces. A tenant does not request resources directly, rather sets up *policies* for auto-scaling.
- Infrastructure management: This component realizes requests from tenants via relevant configuration, provisioning and monitoring. The input (described below) provided in IACRS-TI is used by the IACRS Controller to set up the infrastructure, which includes placement (configuration) of performance metric monitoring probes on relevant resources and subsequent monitoring of those probes. The results from the probes (monitored metric values or threshold events) are evaluated by the IACRS controller, which implements the auto-scaling algorithms described below.

### A. *IACRS Tenant Interface Input*

The interfaces and relevant parameters provided in IACRS-TI are as follows (we call the input IACRS Policy or IACRS-Pol):

- Resource R to be monitored and auto-scaled (typically this will be an existing resource URI).
- Performance metric to be monitored (PMT).
- Reference to a resource (end-point 2: EP2), if the metric is binary (such as response time, delay or jitter from one resource to another).
- Group ID of the group R belongs to and optionally an indication whether R is a parent of the group.
- Threshold Policy (TP) for R and PMT:
    - ThrU = #%, ThrbU = #%, ThrL = #%, ThroL = #%, Duration = # sec (see Section B below for definition).
- Auto-scaling policies (SP) for a resource and metric:
    - Resource acquisition (SP-RA) policy: *<increment in #%> (INCR), <max resources in #> (MAX),* where *increment* is the number of resources to be acquired, which is calculated as *<original # of resources in a group>* * *INCR* rounded to the next higher number and *MAX* is the maximum number of resources that can be acquired when no more resources will be acquired irrespective of load on all the resources in a group.
    - Resource release (SP-RR) policy: *<decrement in #%> (DECR), <min resources in #> (MIN),* where *decrement* is the number of resources to be released, which is calculated as *< # of resources in a group>* * *DECR* rounded to the next higher number and *MIN* is the minimum number of resources that should be retained when no more resources will be released irrespective of load on all the resources in a group. If SP-RR is not specified, then SP-RA will be used with INCR replaced with DECR and MIN defaulting to 1.

Not all resources and metrics (in a group) will have SP-RA/RR associated with. Resources are acquired or released for the ones that have SP-RA/RR associated with. We now proceed to describe the IACRS algorithms.

### B. *IACRS Thresholding Heuristics*

We extend the standard hysteresis mechanism (IACRS-Hyst) to add two levels of threshold parameters: *ThrbU,* which is slightly below the higher threshold *ThrU,* and *ThroL*, slightly above the lower threshold *ThrL* as shown in Figure 4. In addition, we add a *duration* parameter (specified as seconds), which is used for checking persistence of metric value above/below ThrU/ThrL and Thrbu/ThroL. The IACRS-Hyst will provide more accurate and controlled scaling of resources.

A description of the heuristics evaluation method follows. Let, performance metric value is *PMT_value* and the variable to decide when to auto-scale is *Scale*, which is initially set to *00*.

- **Metric value trending up with persistence:**

A *PMT_Value* behavior for this case is shown in Figure 4.

- When the *PMT_Value* crosses above the *ThrU* for the first time the duration clock starts to tick and we set a variable *Tick_Up_Start* to *1*:

- *If Tick_Up_Start == 0 && first_time (PMT_Value > ThrU), Tick_Up_Start = 1.*

- If *PMT_Value* remains above the *ThrU* or oscillates around it, but remains above the *ThrbU* for the specified duration (such as 5 minutes), then stop (reset) clock tick and set *Tick_Up_End*:
  - *If Tick_Up_Start == 1 && (PMT_Value > ThrU or PMT_Value > ThrbU) && duration == <specified duration>, Tick_Up_End = 1.*

- If *PMT_Value* persisted for duration since crossing *ThrU* and remains above *ThrbU* since last time Scale value was *00* or *01* (this latter condition is to prevent repeated pattern such as the one shown in Figure 4), then set the variable *Scale* to *11* (which may potentially trigger scaling; see below):
  - *If (Scale == 00 or Scale == 01) && Tick_Up_Start == 1 && Tick_Up_End == 1, Scale = 11.*

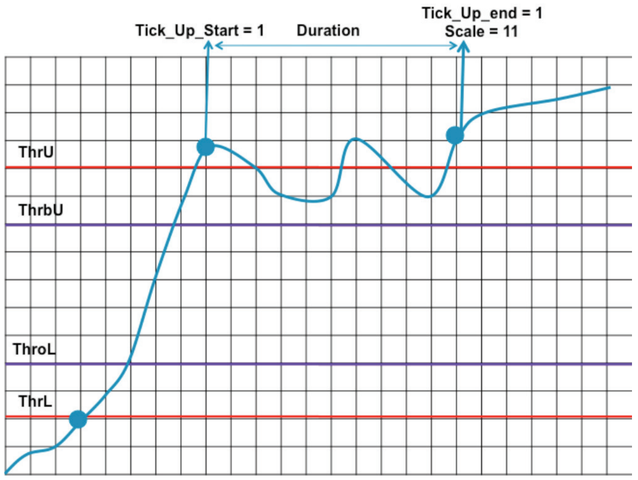

Figure 4. Metric value trending up with persistence

- **Metric Value Trending Down from above:**
  - A *PMT_Value* behavior for this case is shown in Figure 5. If metric value remains below the *ThrbU* for the specified *duration*, then variable *Scale* will be set to *01* (see below group/correlation heuristics on how Scale value *01* is utilized). The clock for this duration starts to tick when the *PMT_Value* crosses below the *ThrbU* for the first time, since it was above *ThrU*.
    - *If (Tick_Up_Start == 1 or Tick_Up_End == 1) && first_time (PMT_Value < ThrbU) && duration == <specified duration>, Scale = 01; Tick_Up_Start = 0; Tick_Up_end = 0.*

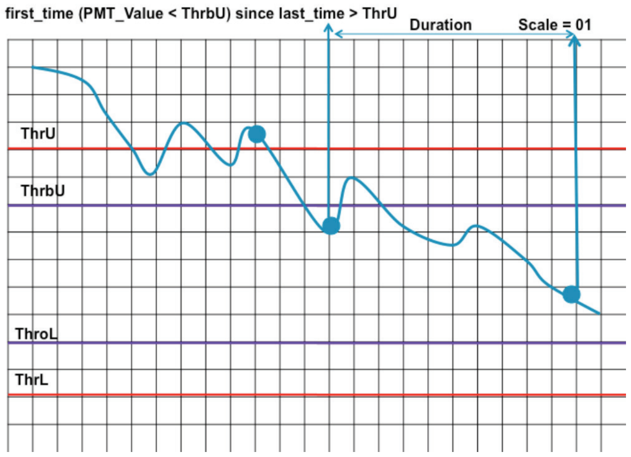

Figure 5. Metric Value Trending Down from above

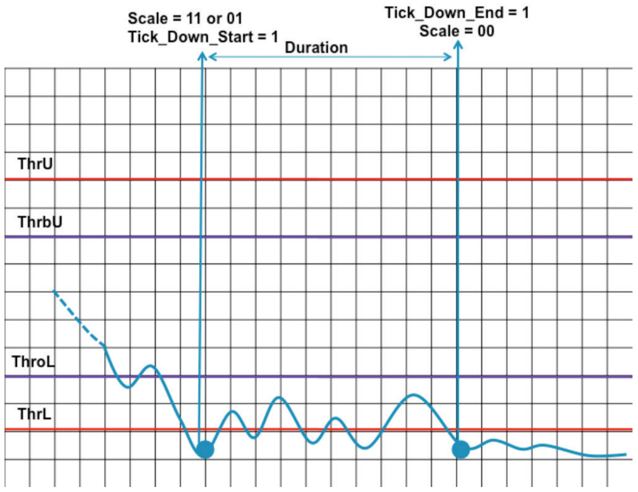

Figure 6. Metric value trending down with persistence

- **Metric value trending down with persistence:**
  - The heuristics algorithm is similar as *PMT_value* trending up with persistence except that the algorithm is applied around *ThrL* and *ThroL* and *Scale* value is set to *00* when *PMT_Value* crosses below *ThrL* for the first time since Scale value was *11* or *01* and remain around *ThroL* for the specified *duration*. A *PMT_Value* behavior is shown in Figure 6.

- **Non-scaling Metric Value behavior:**
  - Two of the *PMT_Value* non-scaling behavior graphs are shown in Figure 7 and Figure 8, in which cases the *Scale* value will retain its last value.

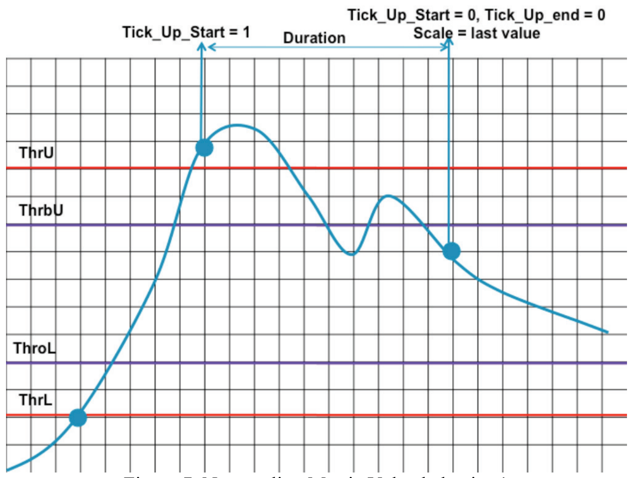

Figure 7. Non-scaling Metric Value behavior 1

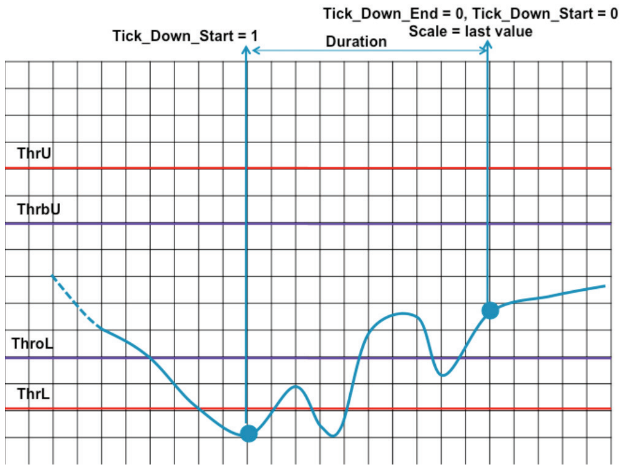

Figure 8. Non-scaling Metric Value behavior 2

### C. IACRS Metric Correlation Heuristics

This heuristics (IACRS-MCH) is based on the concept that spurious auto-scaling can be prevented by grouping together multiple metrics and resources related to each other. If a tenant has specified in IACRS-Pol multiple metrics for a specific resource R, then the inputs are grouped together for evaluation and application of associated scaling policy. A tenant can explicitly group in an IACRS-Pol. If not, the IACRS controller will group the IACRS-Pol together.

### D. IACRS Group Heuristics

The algorithm is as follows:

- *If Scale == 11 for ALL in a group, apply all SP-RA in the group.*
- *If Scale == 00 for ALL in a group, apply all SP-RR in the group.*
- *If Scale == 01 for 50% (this value is configurable) of a group, apply 50% SP-RR.*

## IV. Use Cases

In this section we provide a number of detail use cases to further clarify functioning of IACRS (not all TP and SP for each use case are shown).

- Figure 9 shows a single resource (type) IACRS-Pol. If CPU1 resource of a VM (VM1) is loaded as specified in TP, then more VM will be acquired (for distributing application load running on VM1). Every time Scale is 11, then one more resource (round (1*.5) → 1) will be acquired with maximum of 5 VMs when no more VMs will be acquired irrespective of loads on all the 5 VMs.

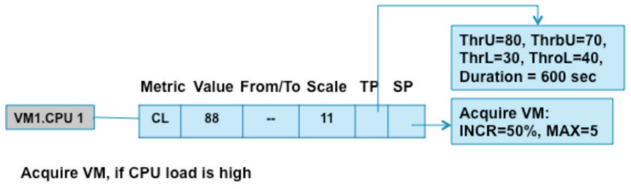

Figure 9. Single resource and metric auto-scaling

- Figure 10 shows an example of (single resource) metric correlation based auto-scaling. This example shows that more resources are acquired under the condition that both CPU load (on VM1) is high and the response time (RT) from a tenant site (measured from the tenant edge router CE11) is high. Note that there is no SP-RA/RR associated with RT.

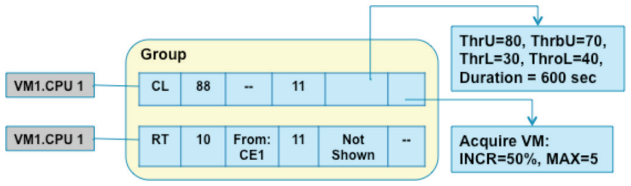

Figure 10. Single resource and multi-metric correlated auto-scaling

- Figure 11 shows a use case corresponding to Figure 2, which shows integrated network (load-balancer) and compute resource auto-scaling. This use case also shows grouping heuristics. The SP-RR/RA policies are associated with both an LB and a compute resource. Note that, we show SP-RR/RA associated with VM1, but in reality it should be associated (in the input specification) with the compute resource group or type.

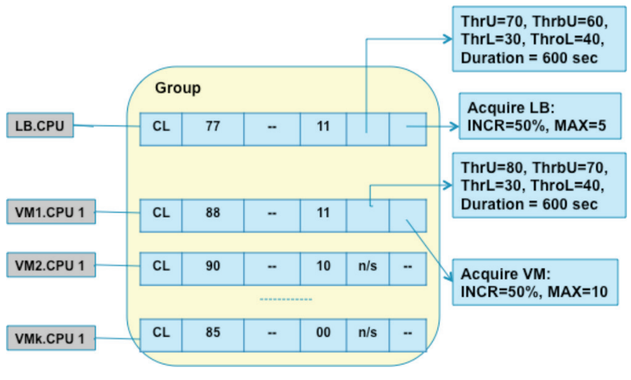
ThrU=70, ThrbU=60, ThrL=30, ThroL=40, Duration = 600 sec
Group
LB.CPU
CL 77 -- 11
Acquire LB: INCR=50%, MAX=5
ThrU=80, ThrbU=70, ThrL=30, ThroL=40, Duration = 600 sec
VM1.CPU 1
CL 88 -- 11
VM2.CPU 1
CL 90 -- 10 n/s --
Acquire VM: INCR=50%, MAX=10
VMk.CPU 1
CL 85 -- 00 n/s --

Figure 11. Multi-resource, Multi-metric and Integrated auto-scaling

- Figure 12 shows network resource auto-scaling. It shows that any type of network resource (in this case a router link) can be auto-scaled. It also shows a case of metric correlation based auto-scaling. It shows that when response time from an edge router (CE11) is high AND link load (LL) on CE11 is high, scale up bandwidth on the link. For example, if bandwidth was originally configured to take 10% of full link bandwidth (that is 100 mbps on an 1 gbps link), then increase to 15% (10% + 50% of 10=5%; with max allowed is 25% of link bandwidth: 3*(50% of 10=5%)+original 10%).

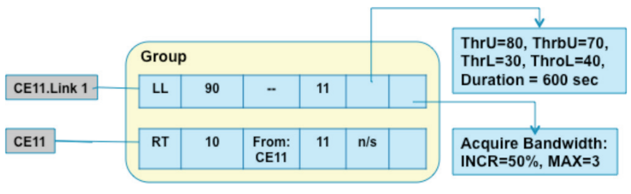
Group
CE11.Link 1
LL 90 -- 11
ThrU=80, ThrbU=70, ThrL=30, ThroL=40, Duration = 600 sec
CE11
RT 10 From: CE11 11 n/s
Acquire Bandwidth: INCR=50%, MAX=3

Figure 12. Single Network resource and multi-metric auto-scaling

## V. Related Work

The Amazon EC2 Cloud service [6] supports a service called the EC2 Auto Scaling (EAS) [7]. The EAS is limited in its support of Cloud resource scaling. The domain of Cloud resource support is limited and specific to Amazon web services (AWS). For example, scaling is supported on EC2 instances, EBS (Elastic Block Storage) and RDS (Relational DB Service). Performance metrics are used to trigger scaling in the EAS. But metrics supported are limited to what is supported in EC2 CloudWatch service [8]. Integrated and correlated scaling as described above is not supported. Network resources as a whole are not considered in the EAS.

OpenStack [9] does not provide auto-scaling mechanisms and just provides a limited resource scheduling system, that considers memory and cpu available on compute nodes but does not consider resources consumption behavior by tenants as we have addressed in this paper.

## VI. Conclusions

One of the major features of a Cloud is on-demand acquisition or release of resources in the Cloud, known as scaling of Cloud resources. The scaling process can be automated benefiting both the Cloud service provider and its tenants (customers). The automation involves *policy* triggered auto-scaling of resources. In a Cloud resources from three domains, compute, storage and network, are acquired/released on-demand. Hence an advanced auto-scaling system should integrate all the three domains for effective scaling of resources. In this paper we described such an advanced Cloud resource scaling system, called the integrated and autonomic Cloud resource scaler or IACRS. The IACRS correlate metrics from the three domains in making scaling decisions. For example, CPU load is correlated with response time over network or network device link load. In addition, the IACRS scale resources from one domain in relation to or integrated with other domains. For example, a load-balancer is scaled together with compute resources or network link bandwidth resources scaled together with compute resources. The IACRS facilitates optimized scaling of resources and avoidance of spurious resource scaling.

We are pursuing further extension of this work in a number of areas, including 1) integration of IACRS with the open source Cloud stack called the OpenStack [9], specially the OpenStack scheduler for advanced auto-scaling of OpenStack controlled and managed Cloud resources, 2) integration with a telco-grade hybrid Cloud [10] framework called the Seamless Cloud [11], [12].

## References

[1] Ganglia Monitoring System: http://ganglia.sourceforge.net/

[2] Cisco Systems NetFlow Services Export Version 9, RFC 3954.

[3] Cisco IOS IP Service Level Agreements (IPSLA): http://www.cisco.com/en/US/products/ps6602/products_ios_protocol_group_home.html

[4] Network Analysis Module (NAM): http://www.cisco.com/en/US/products/ps5740/Products_Sub_Category_Home.html

[5] Remote Network Monitoring Management Information Base, RFC 2819.

[6] Amazon Elastic Compute Cloud (EC2) Service: http://aws.amazon.com/ec2/

[7] Amazon EC2 Auto Scaling: http://aws.amazon.com/autoscaling/

[8] Amazon CloudWatch Service: http://aws.amazon.com/cloudwatch/

[9] OpenStack: openstack.org

[10] NIST Definition of Cloud: www.nist.gov/itl/cloud/upload/cloud-def-v15.pdf

[11] Masum Z. Hasan, et. Al. Seamless Cloud Abstraction, Models and Interfaces, Proc. Of ITU/IEEE Kaleidoscope Conference, Cape Town, South Africa, 12-14 December 2011. http://www.itu.int/dms_pub/itu-t/oth/0B/0C/T0B0C0000383301PDFE.pdf

[12] Masum Z. Hasan, et. Al. Network Abstraction for Enterprise and SP class Cloud: Seamless Cloud Abstraction and Interfaces, IETF Draft, http://trac.tools.ietf.org/area/app/trac/attachment/wiki/Clouds/draft-rfc-seamless-Cloud-masum-01.txt