2011 Fourth IEEE International Conference on Utility and Cloud Computing

Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing

Prasad Saripalli, GVR Kiran, Ravi Shankar R, Harish Narware and Nitin Bindal
Runaware Inc.
Coral Springs, FL, USA
prasadsa@runaware.com

Abstract: Cloud computing is leading to transformational changes with stringent requirements on usability, performance and security over very heterogeneous workloads. Their run-time management requires realistic algorithms and techniques for sampling, measurement and characterization for load prediction. Due to the expectation of elasticity, large swings in their demand are common, which cannot be modeled accurately based on raw measures such as the number of session requests, which show very large variability and poor auto-correlation. We demonstrate the use of load prediction algorithms for cloud platforms, using a two-step approach of load trend tracking followed by load prediction, using cubic spline Interpolation, and hotspot detection algorithm for sudden spikes. Such algorithms integrated into the autonomic management framework of a cloud platform can be used to ensure that the SaaS sessions, virtual desktops or VM pools are autonomically provisioned on demand, in an elastic manner. Results indicate that the algorithms are able to match representative SaaS load trends accurately. This approach is suitable to support different load decision systems on cloud platforms with highly variable trends in demand, and is characterized by a moderate computational complexity compatible to run-time decisions.

Keywords: Cloud Computing; Elasticity; Load Prediction; Hotspot Detection; Slashdot; Autonomic Management

INTRODUCTION

Cloud computing is leading to fundamental, transformational changes in computing and IT. Internet workloads supporting cloud computing have stringent requirements on usability, performance and security [1,2,3], which must be met in a globally distributed, multi-tenant environment used by multiple Concurrent Users (CCU). Cloud workloads map to multiple tenants representing different organizations, co-existing on the same physical (data center) infrastructure, sometimes in co-located VMs. Such environments comprise of a heterogeneous mix of workloads, including remote virtual desktops, virtual application sessions, virtual servers with several connected virtual machines (VMs), graphically intense virtual applications, Rich Internet Applications and the usual Web-based applications. Since all such SaaS sessions are ultimately accessed by the end-users via a Web browser, requests for any such session would count as a Web session request [3]. Run-time management of such cloud platforms requires realistic techniques for sampling, measurement and characterization of the expected workloads (load prediction). History matching and modeling algorithms are necessary to iteratively evaluate and train the load prediction algorithms to improve the accuracy of prediction. Due to the expectation of elasticity inherently built into Software as a Service (SaaS), large swings in demand, including sudden spikes known as 'hot spots', are to be expected. To respond effectively, heavy replication of content, autonomic provisioning of VMs and virtual sessions may be applied just in time, across tenants.

Raw measures such as the number of requests for sessions over time show very large variability even at different time scales [4,5,6]. As such, it is difficult to make run-time decisions on expected loads and hotspots based on such raw data. We report on the adaptation and implementation of algorithms for predicting future resource loads under real-time constraints [6] to Web-based cloud platforms. This method is based on a two-step approach that first aims to get a representative view of the load trend from measured raw data, and then applies a load prediction algorithm to load trends. Given a set of timestamps and the corresponding loads at those timestamps, we predict the loads at any given timestamp (in the future), using cubic spline interpolation. This prediction can be applied separately to the different SaaS and PaaS session types as well as to a combined total session count, which appears as the number of Web session requests to a cloud platform. Other resource measures such as the number of VMs on demand at any time, amount of RAM, disk space, CPU or network bandwidth may also be modeled using this approach, for load prediction on clouds.

Using Runaware's cloud platform as an example, we demonstrate how the proposed models can be applied to a cloud platform with the ability to autonomically respond to the predicted loads. A separate algorithm is developed for the prediction of hot spots from the load prediction data. The algorithms are integrated into the autonomic management framework of the Runaware enterprise cloud platform for testing the methods, such that prepackaged VM pools can be created and deployed to ensure that the required SaaS sessions, virtual desktops or VM networks are provisioned on demand, in an elastic manner.

WEB WORKLOAD CHARACTERIZATION

It is expected that the Internet would be the global network to enable cloud computing and the Web browser its client interface on the end user devices to present the Software as a

978-0-7695-4592-9/11 $26.00 © 2011 IEEE
DOI 10.1109/UCC.2011.66

Service (SaaS). It is not realistic to assume that the cloud workloads can be predicted using the Web workload characterization models. The nature, economic value, size and duration of the cloud workloads (SaaS for example) are very different from those of typical Web applications. Multi-tenancy, elastic on-demand provisioning and 'pay-per-use' billing also are unique to cloud workloads. However, Web workload characterization can serve as a good early surrogate for developing methods for load prediction and hot spot detection of cloud (SaaS) workloads, which are evolving and yet to be characterized. Predicting the server loads have been a problem of high importance in past few years. Many prediction techniques have been developed to address the same, but all of them have their limitations. There is no global prediction algorithm developed yet that works well for all type of patterns of load data. So, one needs to make a choice for prediction algorithm depending upon the pattern their load data follows. A review of the state of Web workload characterization efforts is presented in this section, to throw light on the efforts necessary for similarly characterizing the cloud workloads and to identify a method suitable for adoption to clouds.

Lacort et al (2002) conducted a comprehensive Web workload characterization [7], collected over several weeks among a large (more than 50, 000 users and 122M requests) user base, confirming that graphic files are the most common files in the Web, comprising of more than 60% of the total Web requests. File sizes among Web transactions showed an enormous dispersion, shown by a large standard deviation (305.7KB). They also reported dramatic increase in the use of the Web in just one year, 49% more requests and 77% more bytes transferred, which is consistent with the global increase in the Internet use by more than 400% between 2000 and 2010. The noticeable increase of requests for dynamic pages (64.1% higher) against a decrease in the demand of static pages, (6.3% lower), showed that new generation of dynamic Web applications is gaining popularity. This study serves as a good model for similar characterization studies needed for cloud workloads, whose global size and reach are growing rapidly as well.

Arlitt and Williamson et al (1997, 2007) presented [8, 9] methods for Internet workload characterization, which focused on the document type and size distribution, document referencing behavior and geographical distribution of server requests. Using six different data sets, they showed that the mean size of the documents transferred was between 5 - 21 KB, among which HTML and image documents accounted for more than 90% of the total requests. Caching was shown to improve the server response, where caching to reduce the number of requests was more effective than caching to reduce number of bytes transferred. Cloud workloads are likely to involve much larger file sizes; a similar characterization and caching analysis would be useful to designing cloud performance.

Baryshnikov *et al* (2005) presented methods [10] for the prediction of Web server traffic congestion, targeting page request traffic in automobile and airline transportation networks and focusing on hot spot detection. A hot spot is a sudden spike in traffic volume, also known as a flash crowd, a storm and the Slashdot effect. This concept is very relevant to cloud workloads because satisfying the demand during hotspots is critical to satisfying the elasticity expectation. Using auto-correlation functions and linear least squares extrapolation as the basis, [10] presented a hotspot detection method applied to typical Web session request data from events such as the Olympics. Results indicate that even very simple prediction algorithms have a significant predictive power for hotspots, where accuracy is not critical, in the sense that false predictions are better than missing a hot-spot completely [10]. While this is true for Web apps, accurate prediction and response to hotspots is much more critical to cloud workloads serving business.

Jung et al (2002) presented methods [11] for characterizing load patterns leading to flash events and denial of service attacks on Content Distribution Networks (CDN). They proposed new ways of characterizing these events, such as rejecting the service request of the users under a DOS attack, and adaptive algorithms based on caching and dynamic delegation. This work is relevant to cloud load prediction in identifying load data generated by genuine (legitimate) users and not attackers.

Recently, Andreolini et al (2009) presented [12] an overview of dynamic management of virtualized app environments, for cloud applications. This work focused on supporting VM migration decisions in a cloud environment, to answer questions such as which VMs should be migrated and when, using an algorithm which does not depend on the instantaneous behavior or average trends but uses load trend behavior, a better method to avoid false (inaccurate) prediction or alarms. For example, if the CPU utilization more than 85% for only 2-3 seconds, no relacement measures are triggered. This work is not directly relevant to the present cloud workload characterization, which is a prerequisite input needed for such VM migration decisions.

From the literature review, it is clear that algorithms for load prediction, critical to the success of cloud computing, are not simple, deterministic functions of raw resource measures over time. Elastic and autonomic management of cloud workloads of any kind requires response to spikes effectively, using just-in-time (JIT) replication of content, pre-preparation, retirement and migration of VMs etc. Web workloads are characterized by large variability even at different time scales, poor auto correlation functions (ACF), heavy-tailed distributions and flash crowds [4,5,6,7]. Such trends lead to skewness of raw data, such that future resource measures appear as unrelated to the previous sample. Deducing a clear trend about the load behavior of a

resource (e. g. find out whether a resource is offloading, overloading or stabilizing) is almost impossible with this type of workloads [6,8]. Popular linear auto regressive models (ARMA and ARIMA) modeling such workload trends require frequent updates of parameters in oscillatory systems. Auto regressive models typically are created offline, applied to workloads characterized by smaller variability and high autocorrelation of load measures, an approach which is computationally costly and inadequate to support run-time decisions.

Keeping these challenges in mind, Andreolini et al (2006) investigated load prediction methods for Internet based systems [6], where the raw data represented as number of Web session requests over time varies drastically, making any attempts to predict loads based on methods such as linear autoregression very difficult, given the poor auto-correlation which characterizes Web workloads. To address this, they proposed a two-step approach that first aims to get a representative view of the load trend from measured raw data, and then applies a load prediction algorithm to load trends. In the present work, we have adopted the two-step method [6] to cloud workload prediction, using a cubic spline for load tracking, detailed in the following section.

METHOD DEVELOPMENT

The two step approach consists of load tracking followed by load prediction. The first step is focused on getting a representative view of load trend from raw data in the form of a load tracker, which *a priori* filters out noises in raw sequence of uncorrelated resource measures S_i sampled over time t_i. A Moving Average (MA) is a finite impulse response filter to analyze a data set by creating a series of averages of different subsets of the full data set. Given a series of numbers and a fixed subset size, MA is found by first taking the average of the first subset. The fixed subset size is then shifted forward, creating a new subset of numbers, which is averaged. This process is repeated over the entire data series. The plot line connecting all the (fixed) averages is the MA. Thus, MA is a set of numbers, each of which is the average of the corresponding subset of a larger set of data points. Calculation of MA may also use unequal weights for each data value in the subset. MA is commonly used with time series data to smooth out spikes and highlight trends [13].

Load tracker (LT) algorithm takes as its input $S_n(t_n)$ for a sample size of n, gives a representation of the resource load conditions l_i at t_i. Applying LT multiply to a sequence of load values helps yield a regular trend of load conditions. SMA evaluates a Simple Moving Average for each S_i over entire observation period. Equal weight is assigned to every resource measure, basing on short-term MA (working on a small set). As such SMA is more responsive to short variations, but causes increased oscillations. Long-term moving averages are able to smooth out all minor fluctuations, and tend to show only long-term trends. SMA models tend to introduce a significant delay in the trend representation as the set size increases. As such, an SMA based prediction cannot facilitate a JIT response to meet the elasticity requirements the cloud workloads demand.

Exponential Moving Average (EMA) is the weighted mean of the n resource measures of the set $S_n(t_i)$, where the weights decrease exponentially. For each time t_i where $i > n$, the EMA-based load tracker is equal to:

$$EMA\left(\vec{S}_n(t_i)\right) = \propto * s_i + (1-\propto) * EMA\left(\vec{S}_n(t_{i-1})\right) \quad (1)$$

constant $\alpha = 2/n+1$ is a smoothing factor. The EMA value is initialized to the arithmetical mean of the first n measures. As the last resource measures give a contribution higher than the first measures of the set $S_n(t_i)$, EMA load tracker is able to react rather quickly to changes in the resource load conditions similarly to a short-term SMA, with the advantage that EMA is subject to less oscillations than a short-term SMA [10]. However, even EMA load tracker introduces a delay in load trend prediction when the sample size $S_n(t_i)$ increases. ARMA and ARIMA trackers are not suitable for cloud as they are strongly dependent on the considered resource metrics and workload characteristics.

Andreolini [6] recommended the use of non-linear LT when the ACF of the data trend shows high fluctuations. Lower-order curves (degree < 3) do not react quickly to load changes, while higher-order curves (degree > 3) are unnecessarily complex, introduce undesired wiggles and are computationally too expensive for a run-time context. Based on [6], we use a cubic spline-based load tracker $LT(S_n(t_i))$, at time t_i is defined as the cubic spline $CS^j(t_j)$. It is obtained through a subset of J control points from the vector $S_n(t_i)$ of the load measures having dimension n. A cubic spline function $CS^j(t_j)$ based on J control points, is a set of $J - 1$ piecewise third-order polynomials $p_j(t)$, where $j \in [1, J - 1]$, satisfying the following properties [14]:

Property 1. The control points are connected through third-order polynomials:

$$\begin{cases} CS^j(t_j) = s_j & j = 1, \dots, J \\ CS^j(t) = p_j(t) & t_j < t < t_{j+1}\,,\ j = 1, \dots, J \end{cases} \quad (2)$$

Property 2. To guarantee $C2$ continuity, 1st and 2nd order derivatives of $p_j(t)$ and $p_j+1(t)$ are set equal at time t_j+1. Applying both properties leads to (where $h_j = t_j+1 - t_j$):

$$CS^j(t) = \frac{z_{j+1}(t-t_j)^3 + z_j(t_{j+1}-t)^3}{6h_i} + \left(\frac{s_{j+1}}{h_i} - \frac{h_j}{6}z_{j+1}\right)(t-t_j) + \left(\frac{s_j}{h_j} - \frac{h_j}{6}z_j\right)z_j(t_{j+1}-t)$$

The Z coefficients are solved by setting Z_0 and Z_n to 0, and:

$$h_{j-1}z_{j-1} + 2(h_{j-1} + h_j)h_j z_j + h_j z_{j+1} = 6\left(\frac{s_{j+1} - s_j}{h_j} - \frac{s_j - s_{j-1}}{h_{j-1}}\right)$$

Load trackers based on moving average models compute a LT value at each resource measure, while the proposed LT based on the cubic spline function returns a new value after *n* resource measures. Further, CS responds well to changes independent of resource metrics and workloads. Cubic spline based LT provides high auto-correlation among values, using which even simple linear predictors can forecast the future behavior of resource load. A load predictor $LP_k(L_q(t_i))$ takes as its input the set of q values and returns a real number that is the predicted value at time t_i+k, where $k > 0$. For cloud application, an LP takes as its input a set of LT values and returns a future LT value. Predicted window k is the size of the prediction interval, and the past time window q, where q is the size of LT vector $L_q(t_i)$, distance between the 1st and last LT value. Considering two load tracker values, the first l_i-q and the last l_i, load predictor is the line that intersects the two points $(t_i-q,\ l_i-q)$ and $(t_i,\ l_i)$ and returns a predicted value of the load tracker l_i+k at time t_i+k [6].

The above methods for load prediction are suitable for prediction of Web-based cloud workloads over a period of several weeks to months. In addition, it is also necessary to predict load patterns on a daily and hourly basis in production platforms. For example, the cloud IT Production engineers managing a SaaS platform across global data centers will need to know how many workloads of which specific session type are to be expected by the hour on each day of the week, and also by the week day over a typical work week. Load trends over such short periods typically do not exhibit wild fluctuations, as is the case with the seasonal trends [6]. A load prediction algorithm based on the generalized cubic equations such as $y = a + bx + cx^2 + dx$, where y is no. of session requests x is hour of the day, was implemented for this purpose. These algorithms also were implemented in C# .Net and integrated with the cloud platform's management server rule engine, for testing.

HOTSPOT DETECTION

Hotspot is an abnormally high demand on a cloud resource over a short period of time. Hotspot level H corresponds to the maximum capacity the platform can handle and meet the SLA [10, 15]. Cloud traffic is experiencing a hotspot at time t if the volume of traffic over the last W_d time slots satisfies the condition (obtain r_i from load prediction):

$$\sum_{i\in[t-w_d,t]} r \geq H \times W_d \qquad (3)$$

This means the average request rate over $[t\text{-}W_d, t]$ is at least H. Working with IT Production, we define ranges for a number of load-resources measures H_n. The platform is hotspot capable only when all of the set H_n are under control [10]. A period of 'Advance Notice' t is defined for each load measure based on IT's ability to respond the hotspot and absorb it following a protocol of best practices. An alarm is set if the traffic currently is in a hotspot, or if an extrapolation of the trend of advance notices in $[t\text{-}W_d,\ t]$ shows that a hotspot will occur sometime in $[t,\ t+t_{max}]$. Alarm is reset if neither of the conditions is currently satisfied [10]. Given that the diurnal, weekly and seasonal variations in resource demand exhibit very different trends, prediction of hotspots on the cloud platform is implemented at 3 granular scales (hour, week, season) separately, so the cloud IT staff can plan and respond at different time scales.

IMPLEMENTATION

Algorithms were implemented in C# .Net integrated with the Management Server Rule engine, which is a Microsoft Windows Work Flow (WF) based framework. In this algorithm, input data (the actual load values available) is first processed by a Non-linear Load Tracker module, which uses cubic spline functions to generate a graphical trend of the data. Using this graphical trend (which is a set of piece-wise cubic mathematical functions, represented by model parameters), the load values between any two actual load points are calculated. The result is a continuous curve representing the input data.

Load predictor takes as its input a set of load tracker values and returns a future load tracker value. Each predictor is characterized by a past window size. Each predictor gives one prediction for a given future point on time line. For example if past window is p, then predictor will take the tracked values at T(L[l]) and T(L[l-q]) (l being the index of last available load data),it draws a straight line joining these two points and extrapolate this line to predict the load value at the require d future point. Accuracy of this load predictor depends on the past window size chosen for prediction. This algorithm works well for near future predictions. In cloud applications, resources are consumed on demand. Services may go down when load suddenly increases because it is not always possible to allocate the demanded resources to an application. Even approximate future load estimates can address this problem enabling better prepared service.

Class Extractinput is implemented to generate the input files from the database, a SQL data store which is continually populated with the cloud session request data over a period of several weeks. The Input file contains data fields to represent Normalized Time Unit and the Actual Load Value. It should be noted that the 'Actual Load Value' is typically the number of SaaS session requests, but can also be any of the key cloud resource metrics such as RAM, CPU, disk usage and network bandwidth, for example. This flexibility is valuable for cloud capacity planning. A Model class calculates the model parameters of cubic splines defined over the input data [12]. A Predict class implements the

function for prediction over a given range of time, and is integrated into the cloud platform's autonomic deployment and management framework.

User (typically a cloud IT Admin) enters window width to define two points on time axis (*L* and *L-W*) where *L* is the last actual load available and W is window width provided by user). Then, the load prediction algorithm is applied using the tracked values within the window to predict the next immediate load at the required time. Additional curve fitting routines were implemented for the prediction of hourly and daily variation in load trends, as explained, using the cubic equation model.

Representative load input data were obtained from [5, 6 and 8], after verifying that they are representative of typical SaaS workloads on Runaware's cloud platform [3]. This platform enables SaaS-ifying Windows desktop applications using a system virtualization layer created on the physical (DC), a presentation virtualization layer to create remote application sessions using Citrix Xen App/Xen Desktop, an autonomic deployment and management middleware layer for orchestrating the cloud workflows, and a Web interface layer to present the SaaS apps to the end users via the Internet, via a variety of Web 2.0 technologies, including Adobe Flash, Java and AciveX clients. Datasets representing hourly, daily and seasonal trends in an example resource measure (number of session requests over time) were adapted as inputs to the load predictions and hotspot detection algorithms so implemented.

RESULTS AND DISCUSSION

Shown in Figure 1 is the seasonal variation in the # session requests predicted using the 2 step Cubic Spline Load Tracker Load Predictor Implementation. The input dataset is based on [6]. It can be seen that the method predicts raw

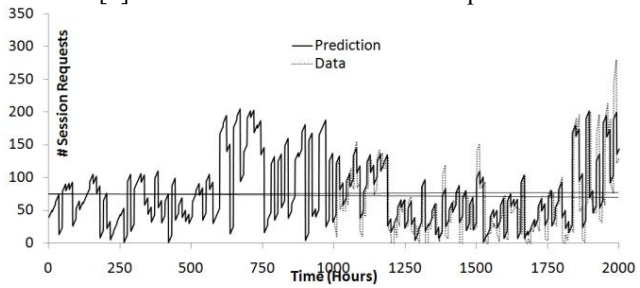

Figure 1. Load prediction of SaaS sessions using the 2-step CS method

SaaS session counts in the time (1000 – 2000) range well, based on the load data shown in the (0 – 1000) range. Application of this to two other data sets showed similarly reasonable agreement. Even if the method is able to predict future load within a 25 – 75% margin of error, it would still be a valuable tool for capacity planning on cloud platforms during run time, in production.

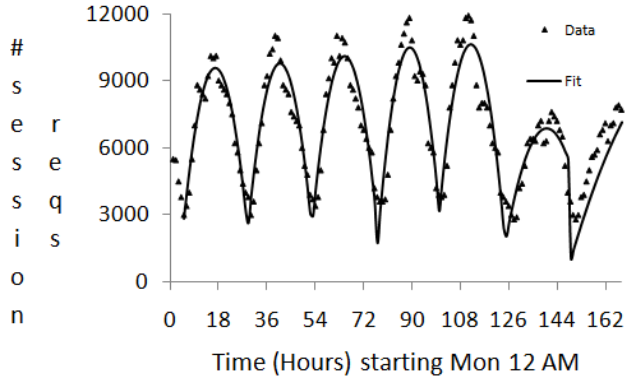

Figure 2. Load prediction of daily trends over a week

Shown in Figure 2 are the daily variations in the # session requests predicted using the cubic equation method, and input data reported in [5]. As expected, the number of Web session requests on each work day (Mon – Fri) starts as low at midnight and peaks around midday. The request count is smaller over the week-end. Such trends can be modeled with reasonable accuracy using the cubic equation method, as can be seen from Fig. 2. A useful way to apply this method is to model fit such available weekly data over several weeks, to obtain a range of values to be expected for the coefficients a, b and c in the cubic equation and then obtain their mean values using regression. A reconstruction of the cubic equation trend would then yield a representative expected value for the load trends by day.

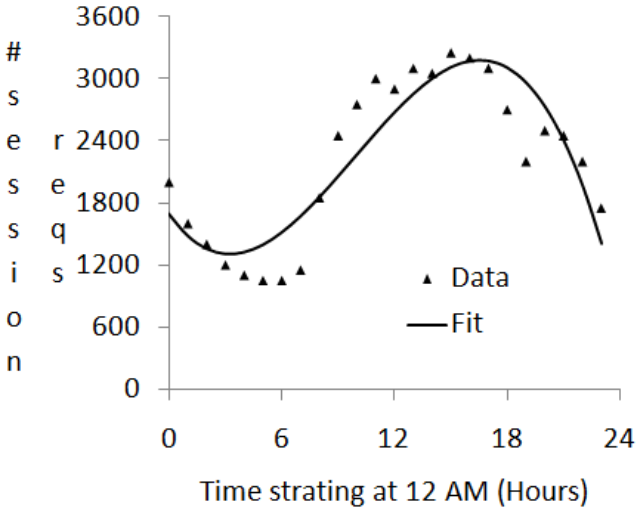

Figure 3. Load prediction of hourly trends over a day

Finally, shown in Figure 3 is daily variation in the # session requests predicted using the cubic equation method as well. Input data used in Figure 3 are from [5], collected over Web servers. As in the case of the daily variations, the hourly data trend over a given day can also be modeled using this method. Regression analysis of such hourly trends over data available from several days, for similar types of

workloads on the same (or similar) cloud platforms, would be useful for constructing expected daily trends as a guideline for capacity planning.

Application of these prediction methods to the heterogeneous session requests among multiple tenants on a cloud platform is implemented using a simple arithmetic summation approach. This is feasible because, mathematically, both the cubic spline based predictions of the seasonal loads and the cubic equation based prediction of hourly and daily loads are amenable to a simple arithmetic summation, when a raw data measure trend needs to be apportioned among the multiple measure types it is comprised of. For example, consider *M* different tenants (*i.e.,* cloud customer orgs) - each with a particular type of SaaS or PaaS workload (session type), all being served by the same single cloud vendor's platform such as the Runaware platform. All such session requests are received ultimately as a pool of session requests over time, which can be directly modeled as in Fig. 1 – 3. In addition, the cloud management server also keeps track of the number of session requests by tenant and session type, using cookies, tenant IDs or IP addresses. As such, prediction trends can also be developed as shown in Fig. 1 – 3, for each session type and tenant, which would sum to yield the global cloud platform load predictions at any given time. These trends are important inputs into dynamic capacity planning, which would be very different for each session type, however. For example, a SaaS session to IOPS and compute intensive workloads would need high capacity VMs, unlike SaaS sessions to email or CRM.

Limitations: These methods predict the load at points not very far from the latest available points. For example, if raw data measures up to time unit 2000 are available, one can only predict up to 2005-2010 with reasonable accuracy. Predicted value depends upon the window width chosen. For example, if the window width is say *W*, the algorithm will make use of (*L* to *L-W*) actual values to predict the load, where *L* is the last actual data point available.

CONCLUSIONS

Load prediction algorithms were adopted for use on a typical global cloud platform, by first implementing a load tracker function to get a representative view of the load trend from measured raw data, and then applying a load prediction algorithm to load trends, for predicting future resource loads under real-time constraints. Given a set of timestamps and the corresponding loads at those timestamps, we predict the loads at any given timestamp (in the future), using cubic spline interpolation. This prediction can be applied separately to the 8 different session types as well as to a combined session count (*i.e.,* number of Web session requests). It is applicable to a host of other load measures (*e. g.* system's CPU, disk and memory; Network and DB metrics etc.) A separate algorithm is developed for hot spot prediction from the load prediction data. Implementation is integrated into the cloud autonomic management framework, to aid runtime decisions for deploying prepackaged VM pools, virtual sessions and other resources on demand, in an elastic manner. Results indicate that the autonomic management framework is able to respond to both synthetic and real-world load trends accurately. This approach is suitable to support different decision systems on cloud platforms, even for highly variable workloads, and is characterized by a computational complexity that is compatible to run-time decisions.

ACKNOWLEDGEMENTS

We are thankful to Ben Walters, CTO of Runaware Inc, for his guidance and review of this work.

REFERENCES

[1] Reese, G. (2009) Cloud Application Architectures, O'Reilly, Inc.

[2] Saripalli, P. and Walters, B. (2010) "QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security," 2010 IEEE 3rd Intl. Conf. on Cloud Computing; Miami, FL, July, 2010.

[3] Saripalli, P., Oldenburg, C., Walters, B. and Nanduri, R. (2010) "Implementation and Usability Evaluation of A Cloud Platform for Scientific Computing as a Service (SCaaS)," Scientific Prog. Journal, Special Issue on Science-driven Cloud Comp (under review).

[4] A. K. Iyengar, M. S. Squillante and L. Zhang, "Analysis and characterization of large-scale Web server access patterns and performance," World Wide Web, Springer, 1999.

[5] Dilley, J. A. (1996) "Web Server Workload Characterization," HP Report http://www.hpl.hp.com/techreports/96/HPL-96-160.html

[6] M. Andreolini, S. Casolari, Load prediction models in Web-based systems", Proc. of First Intl. Conf. on Perf. Evaluation Methodologies and Tools, Pisa, Italy, October 2006.

[7] J. Lacort, A. Pont, J.A Gil, J. Sahuquillo, A Comprehensive Web Workload Characterization 2nd Intl. Working Conf. on Perf. Mod. Eval. Heterogeneous Networks (HETNETs-04) 2004.

[8] M. Arlitt and C. Williamson, ``Internet Web Servers: Workload Characterization and Performance Implications", IEEE/ACM Transactions on Networking , Vol. 5 (5), pp. 631-645, October 1997.

[9] A. Williams, M. Arlitt, C. Williamson and K. Barker, "Web Workload Characterization: Ten Years Later," in Web Content Delivery: Web Information Systems Engineering and Internet Technologies Book Series, 2005, Vol. 2, I, 3-21.

[10] Baryshnikov, Y., Coffman, E., Pierre, G., Rubenstein, D., Squillante, M. and Yimwadsana, T. "Predictability of Web-Server Traffic Congestion," 10th International Workshop on Web Content Caching and Distribution (WCW'05), Sophia Antipolis, France.

[11] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In Proc. Intl. WWW Conference, Hawaii, May 2002.

[12] M. Andreolini, S. Casolari, M. Colajanni, M. Messori, "Dynamic load management of virtual machines in a cloud architecture", Proc of First Int. Conference on Cloud Computing (ICST CLOUDCOMP2009), Munich, Germany, Oct. 2009.

[13] J. D. Hamilton. (1994) "Time Series Analysis," Princeton University Press.

[14] http://en.wikipedia.org/wiki/Spline_(mathematics) visited on May 8th, 2010.

[15] S. Adler. The slashdot effect: An analysis of three internetpublications. http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html.