Cloud Publications
International Journal of Advanced Cloud Computing and Applied Research
2013, Volume 1, Issue 1, pp. 1-12, Article ID Tech-187

Research Article **Open Access**

Location-Based Cloud Resource Allocation Based on Information of the Social Web

Anna Schwanengel[1], Alexander Schönl[2], and Claudia Linnhoff-Popien[2]

[1]Siemens AG, Systems Integration, Otto-Hahn-Ring 6, Munich, Germany
[2]Ludwig-Maximilians-University Munich, Institute of Informatics, Oettingenstr, Munich, Germany

Correspondence should be addressed to Anna Schwanengel, anna.schwanengel.ext@siemens.com

Publication Date: 15 November 2013

Article Link: : http://technical.cloud-journals.com/index.php/IJACCAR/article/view/Tech-187

Abstract The development of mechanisms as virtualization, load distribution, and data sharing has benefited the IT-evolution towards to cloud computing with its seemingly infinite resource capacities. This leads to novel approaches in resource provisioning and releasing. However, managing the cloud resource pool still needs manual configuration to react on changes in system load behaviour. This paper presents simulation results of the already described idea, which procures user-data based on social Web applications and uses this information for load forecasting. The solution includes the data of geocaching- or geotagging-services to determine how many people are located at a particular place with a specific interest focus. By inferring the pure load volume at a specific location and by the presented formulas, our system is able to provide an appropriately amount of resources. Thereby, we can optimize the trade-off between low costs with less resources and high user satisfaction with use of many machines.
Keywords *Cloud Computing; Cost Factors; Resource Management*

1. Introduction

In the past decade the progress of cloud computing is facilitated by novel technologies like virtualization. So, it is easier to map the actual workload on the correct amount of resources [1], while having more flexibility over the entire resources [2]. As cloud computing shows great opportunities to serve several individual clients, it has high market potential and load on cloud servers is rising enormously [3].

Chen et al., therefore, generate and use load patterns to predict the upcoming load at future time [4]. However, forecasting the exact instance amount needed by a company is difficult whilst essential to provide quality of services for the clients in a cloud [5]. Furthermore, present cloud solutions at most offer resource management on the basis of rule-based configurations [6]. The cloud user needs to define instructions on different load parameters and to set thresholds in order to counteract an overload situation [7]. Additionally, cloud users are faced with the problem that booting instances still requires a time up to 13 minutes [8] – depending on the complexity of the application and the underlying cloud platform [9].

Within this work, we present a solution to predict upcoming loads in order to avoid under- and over-provisioning of SaaS-offerings at specific locations. First of all, we identify a significant probability for load increase. For that reason, we use the data of the social Web application *Twitter* to achieve a forecast of the future amount of load on a particular service. We can assume that a measured increase of tweets about a specific topic infers to an increase of load on a service related to the corresponding tweet [10].

Secondly, by including data about the customers' whereabouts, the main resource demand at a location can be mapped to an optimal data center location. Data centers are offered locally distributed by the providers in different regions [11]. So, the decision where to instantiate the determined amount of resources is based on the location of the load. The required list of locations is filled with the actual places of the mobile users based on geo-service extractions via their API. Consequently, required resources for service provisioning can be offered in a timely and location-based manner even in bursting situations.

Additionally, we provide a formula which calculates the exact amount of required resources and leads to an ideal covering of resource demands. The further developed cost functions for users and providers enable the finding of an optimized trade-off between low emerging costs and a high user satisfaction. Thereby, we offer a suitable technique for public cloud applications to react on unpredictable load variations and to infer a dynamic resource allocation. Following, we will give an overview about related work in Section 2 and describe our approach in Section 3. Section 4 provides implementation details and the simulation results. Furthermore, the solution is evaluated theoretically in Section 5, while presenting functions to calculate the exact and optimal amount of needed resources at a specific time point as well as the emerging costs for users and providers. Finally, Section 6 summarizes the paper and concludes with future work.

2. Related Work

Beside the above already mentioned work, we find further approaches which use recurring load patterns to identify periodic load increases. Gmach et al., e.g., achieve load reaction in a self-organizing way [12]. In their 'AutoGlobe' system load is distributed and tasks are transferred to less loaded servers on overload detections. They use near term predictions of load for periodic occurrence and for abnormal happenings; they formulate hints based on instance utilization [13]. However, they accept potential delays for the user caused by long-distance task routing, which we aim to avoid.

When relying on cloud infrastructures and Internet accessibility based on best-effort load approaches [14], research areas are able to put the user-side in the focus. The question is how customers use the Internet concerning information search because this eventually implies load for the underlying infrastructure. Therefore, Phillips and Young presented interesting results about humans searching the Internet for acquisition of information [15]. With moving communication to the Internet, also new forms of ordering information emerged. Thereby, the importance of social networks like *Facebook* and communication ways via *Twitter* is increasing more and more, as flows of information are enormously fast within these. This holds especially for emergencies and crises [16]. These events occur normally by pure chance, which complicates forecasting of correlated load. However, by their studies about information flows in the Internet, J. Yang and the American professor Leskovec found out that after relevant events the further load progression is predictable, which eases load forecasting based on real-time data [10, 17].

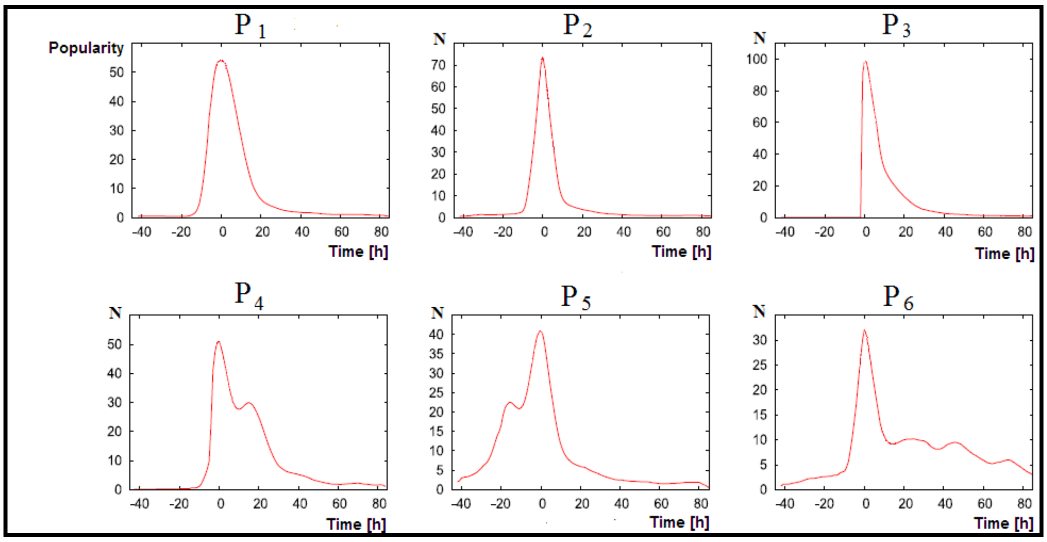


Figure 1: *Patterns of Popularity Behaviour in Social Web [17]*

For that reason, the patterns representing the main spread of information within the social Web were extracted (see Figure 1). All patterns P_1 to P_6 display an enormous increase of popularity in the first 20 hours. Within this time interval the curves feature a slope with an approximately exponential growth. The proposed model is able to forecast the upcoming popularity and load based on in-time data extraction [18].

In summary, with the growing interest in cloud systems, technical aspects concerning load variation have to be increasingly considered. However, optimal distribution of load emergence concerning low response times is still not regarded in total. That shows the need of further research for an efficient and automated resource management while also taking the main demand of users and their actual location into account.

3. Resource Management Based on Data of the Social Web

As already described in our previous work “Resource Allocation for Cloud SaaS Offerings based on Social Web Applications” the main goal of the designed system, is the automation of resource management in load peak situations of cloud computing environments [19]. Current related work (see Section 2) presents a verifiable correlation between user interests and the actual usage of a respective application. This way, it is possible to deduct a relation between the capacity utilization within a data center and the popularity of application information. That forms the basis for inferring the upcoming load progress of a cloud application in order to estimate the probability of an actual load increase. Consequently, cloud bursting situations are manageable through early resource allocation before the actual load increase. Thereby, a trade-off must be found between minimal cost caused by more reserved resource allocation and high efficient service provisioning and user satisfaction by more generous booting strategies for additional capacities.

As depicted in Figure 2, we use data of the social Web in order to infer exactly such a prognosis of load changes. Within, Twitter acts as a first indicator of increasing load parameters because according to today’s preliminary findings it transports the user reaction very prompt [20]. Furthermore, the recorded amount of tweets can be easily mapped onto the corresponding load curve. This data is analyzed in order to check to which extent the tweet-curve correlates with the load progress. Concluding the examination of correlation between load and tweets, a value can be stated expressing the probability of the expected load increase. Based on the history, a probable load volume and its arising can be forecast. In a second step, the location of the main demand is deducted in order to provide the needed resources in the near clients' proximity. Using data of geo-caching services, it is possible to get the users' surroundings and forward them to the optimized hosting location. This algorithm is described in detail in our last work [19].

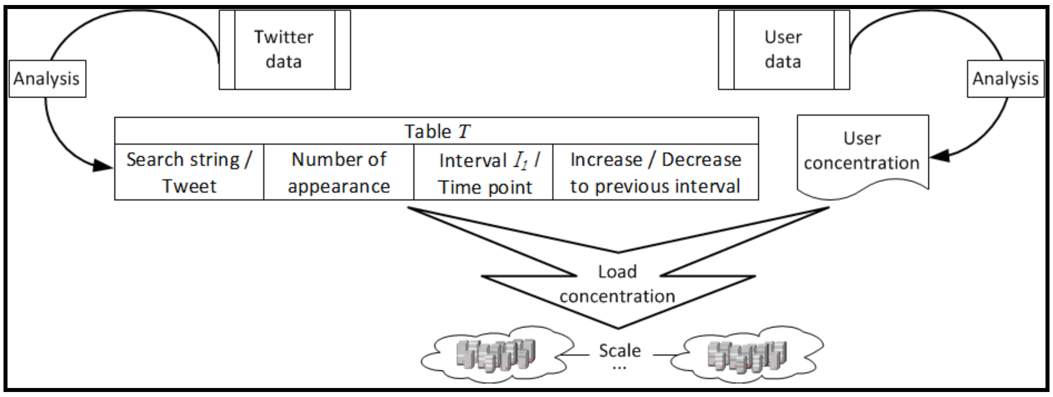


Figure 2: *Procedure of Resource Provisioning*

Based on the research of Yang and Leskovec, it is known that identifiable load patterns show, especially in the first 20 hours, an enormous increase of the popularity in load behaviour [17]. While dealing with information flows in the social Web, we can assume an exponential increase in the beginning [18]. Consequently, we define the exponential growth function (1). The Euler's number e represents the function's basis, the growth factor of the function is named α and β corresponds to the shift on the y-axis. This is the input for the simulation of the implemented tweet-server:

$$f(t) = e^{\alpha t} + \beta \tag{1}$$

Now, we have to relate the number of tweets with the recorded load-curve on the cloud resources. We trace the tweet mentions and the load variations for the same interval in tables and compare both curves. Finally, we use the chi-square-test in order to extract the correlation between the load- and tweet-progress.

If there is no slope in the curves of tweets and load, showing the identical behaviour of constant progress, the chi-square-test would result in a high probability of correlation. However, this conclusion is not desired because in case of correlation detection, the scaling process is initiated, which is not correct on constant load progression without any in- or decrease. Consequently, flat and steady load- and tweet-curves need to be filtered out by load pattern analysis.

The load pattern analysis compares the recorded curves with already identified patterns and checks according to their similarity whether the curves are rising, declining or constant. This way, negligible little increases in the curve progression cause no unnecessary resource allocation and only significant load changes are considered in the chi-square-test. As shown in Figure 3, on a positive match to a load increase, new resources are booted and on load decreases, resources are shut down. A further step is to decide what the optimal resource amount is like and how many instances can be booted or shut down. The solution of this problem is presented in Section 5 after having defined to following data basis.

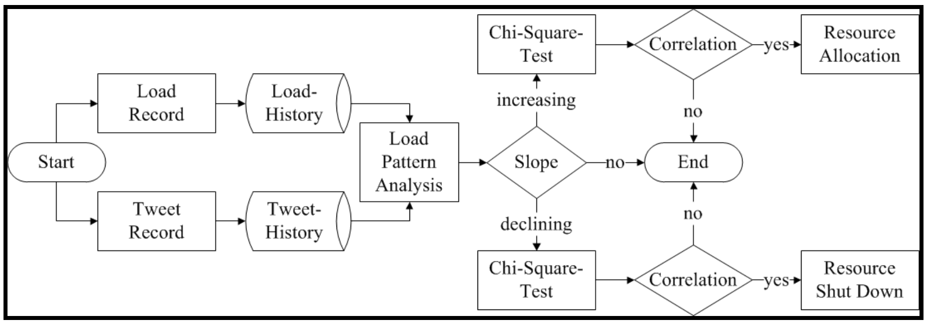


Figure 3: *Case Differentiation on Resource Allocation*

4. Implementation and Results of the Cloud Response Time Simulation

We wanted to decide whether the location-based resource allocation with social Web data works well. So, we evaluated the feasibility to offer a better response time on the services, and how much influence the server processing time and the distance to the clients have on the response time. For that reason, we simulated the Round Trip Delay (RTD) between sending a request to several servers and receiving a response. These Proxy- and DNS-servers are located at different data centers according to the Microsoft Windows Azure regions. Using the *Geobytes*-service, it is possible to detect the server location by extracting the sending location of the IP-address [21] and by measuring via the ICMP-protocol [22]. We sent 250 ICMP-packages each to the four regions *West Europe*, *North Europe*, *East US* and *East Asia* using *ping*-calls out of Munich. Table 1 (a) depicts the rounded measured values of the RTDs in milliseconds for each region. Following this, we could specify a factor comprising the ratios of RTDs compared to our home region West Europe, while regarding the signal transmission time and processing time – see Table 1 (b).

Table 1: *Round Trip Delays to Data Centers (a) and RTD-Ratios to Region West Europe (b)*

(a) *Round Trip Delays*

Region	Avg. RTD [ms]
North	44
Europe	47
West Europe	141
East Us	362
East Asia	

(b) *Round Trip Delay Ratios*

West Europe to	Ratio
North Europe	0.94
East Us	3.2
East Asia	8.23

In general, the response time T of a cloud application is affected by two times. It depends on the duration of data transfer T_D between clients and servers and on the processing time T_S of the claimed servers. In this simulation, we measured the total response time as the sum of these two values:
$T = T_D + T_S$.

Furthermore, we defined three categories *Small*, *Middle* = 10 * *Small* and *High* = 10 * *Middle* = 100 * *Small* in order to achieve comparisons by variations of these dimensions. The simulation uses the ratios of the different data transfer times to the four regions shown above. As starting point, for the category *Small*, a transfer time to East Asia of $T_{D\ \mathrm{Small}} = 10000\ ms$ is set and scaled up for all other regions and categories. The processing times T_S are determined constantly and equally for all regions: $T_{S\ \mathrm{Small}} = 100\ ms$, $T_{S\ \mathrm{Middle}} = 10 * T_{S\ \mathrm{Small}} = 1000\ ms$ and $T_{S\ \mathrm{High}} = 10 * T_{S\ \mathrm{Middle}} = 10000\ ms$.

The developed Service Load Manager performs *n* threads and the transfer time T_D is implemented via a sleep-instruction. Afterwards the reawakened thread stores a message in a queue. This states the end of transmission and launches a service thread which simulates the processing time T_S by a sleep-instruction. Finally, the service informs the client about the end of transfer. This total time is stored as response time.

The simulation was realized with 20 threads for the services and clients and 50 transactions per each client with requests out of the region West Europe. The averaged measured values of West Europe are listed in Table 2 (a) and range from 1.382 seconds to around 2.3 minutes. It is remarkable that the measurements to North Europe from around 1.319 seconds to 2.2 minutes feature, in fact, shorter response times compared to our times in West Europe (see Table 2 (b)). This shows that geographical proximity itself does not lead to better response times. With better expansion of infrastructure in North Europe, even higher distances may cause shorter RTDs compared to the closer servers in West Europe. Consequently, the provided underlying infrastructure must be taken into account when considering where to boot new instances.

Table 2: *Response Times to Region West Europe (a) and Region North Europe (b)*

(a) *Response Times to Region West Europe*

		Processing Time T_S [ms]		
		Small	*Middle*	*High*
Transfer Time T_D	*Small*	1382	2282	11285
	Middle	12917	13820	22827
	High	128259	129186	138210

(b) *Response Times to Region North Europe*

	Processing Time T_S [ms]		
	Small	*Middle*	*High*
Small	1319	2218	11219
Middle	12285	13188	22176
High	121865	122757	131795

Table 3: *Response Times to Region East US (a) and Region* East Asia *(b)*

(a) *Response Times to Region East US*

		Processing Time T_S [ms]		
		Small	*Middle*	*High*
Transfer Time T_D	*Small*	3947	4844	13847
	Middle	38575	39465	48465
	High	384639	385474	394548

(b) *Response Times to Region East Asia*

	Processing Time T_S [ms]		
	Small	*Middle*	*High*
Small	10103	10999	19998
Middle	100098	100933	110104
High	1000285	1000979	1010852

However, measurements for the regions East US and East Asia promise a crucial response time improvement after a resource allocation in West Europe. The response times to East US – presented in Table 3 (a) – range from four seconds in the smallest rubric to the high value of 6.58 minutes in the *High*-category. The response times to East Asia are actually even higher with 10 seconds to nearly 17 minutes – see Table 3 (b).

We aim to determine the improvement of the response time when moving resources to the near proximity of the requesting client in load situations. This improvement factor is called *q*. Therefore, we relate these response times with the times of West Europe in Table 2 (a) through dividing the West Europe measurements by each response time of the other regions. Table 4 (a) depicts the improvement factors of all categories, which arise out of shifting the resource allocation from East US to West Europe. The transfer category *High* and the processing category *Small* show the highest improvement factor of 3.0, which decreases diagonally to a factor of 1.23 for the *Small* transfer category and *High* processing category.

With providing the required resources in West Europe instead of in East Asia, we can observe the improvement factors shown in Table 4 (b). Again, the tendency declines from a high improvement factor of 7.8 with a transfer category *High* and the processing category *Small* diagonally to a minimum of the improvement factor of 1.77 with the transfer category *High* and the processing category *Small*. This is the evidence by specific values of the logical consideration that with decreasing processing times of a resource, also the whole response time shrinks after a resource allocation within a desired region.

Table 4: *Improvement Factors After Resource Shift to West Europe from East US (a) and from East Asia (b)*

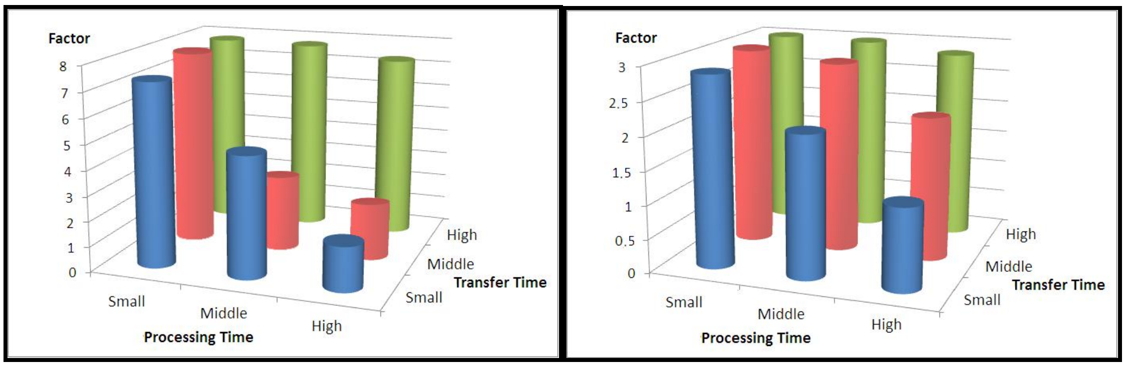
(a) *Factor East US -- West Europe*

ΔT	*Small*	*Middle*	*High*
Small	2.86	2.12	1.23
Middle	2.99	2.86	2.12
High	3.00	2.98	2.85

(b) *Factor East Asia -- West Europe*

ΔT	*Small*	*Middle*	*High*
Small	7.31	4.82	1.77
Middle	7.75	2.86	2.12
High	7.80	7.75	7.31

Figures 4 (a) and (b), additionally, point out that with an increasing amount of transferred data, also the improvement factor *q* of the response time rises. This makes clear: the best results can be achieved with a combination of the category *High* in the transfer time and the category *Small* in the processing time.



(a) *Improvement Factor East US -- West Europe* ***(b)*** *Improvement Factor East Asia -- West Europe*

Figure 4: *Effect of Processing and Transfer Time on Shift to West Europe form East US (a) and East Asia (b)*

To sum it up, our results show an enormous decrease of delays after the allocation of resources in the same region as the origin of the load. These shortened response times will facilitate the popularity of the service. Moreover, by enlarging the user satisfaction, the revenue of the cloud provider will rise.

5. Theoretical Evaluation and Development of a Cost Function

Having a determined data basis through the simulation, we are able to theoretically evaluate the results, and define functions to calculate the required amount of resources as well as emerging costs for users and providers. The resultant formula offers the possibility to compute the optimum of resource amount regarding costs for the provider and a high user satisfaction. The provider can, thereby, specify a probability value for scaling processes by the usage of the chi-square-test [23]. In doing so, one should consider which scaling strategy fits best: If the provider generously allocates instances by a low chi-square-probability threshold, additional resources are booted in many cases even on negligible load increases. They may stay underutilized while causing unnecessary costs. With cost-saving high limits, resources are only allocated on an enormous load increase, which leads to may request drops and high latencies. In order to solve this dilemma, we offer a possibility to determine the required amount of resources at a specific time point t.

In the following, we can use the exponential function f for the tweet-curve to match the popularity of a cloud application in the social Web – see Section 3. Additionally, we assume that the chi-square-test detects a correlation to the load-curve L, which is (compared to the tweet-curve) shifted in time about t_{shift} and a γ-multiple of the tweet-curve. Consequently, the origin of the load lies also in the region of the correlating tweet-curve. The provider can influence to which extent he wants to boot resources in proportion to the emerging load by the factor g. So, we have the following parameters:

- R = denotes the amount of available resources
- $f(t) = e^{\alpha t} + \beta$ is the exponential function of tweets [18]
- $L(t)$ specifies the amount of jobs in a server queue at time point t
- g is the allocation ratio
- t_{VM} stands for the booting time until new resources are ready to use
- t_{shift} describes the time shift between load- and tweet-curve

According to [18], the correlating load will increase exponentially during the booting time t_{VM} and the duration to the next chi-square-test-period $t_{\chi 2}$. So, the provider would like to previously forecast the load out of the tweet-curve including the machine booting time up to the next chi-square-test. In doing so, he would be able to provide the future required resources in an anticipatory way. However, the problematic condition prevails that the exact amount of tweets for this time point is unknown, yet. Hence, we compute with the already known values at the time point $t + t_{VM}$ by approximation. Thus, we extracted the subsequent formulas which specify the required amount of resources $R(t)$ at the time point t and γ as a multiple of the load-curve referred to the tweet-curve:

$$R(t) = g \cdot L(t + t_{VM}) = \gamma \cdot f(t + t_{VM} - t_{shift}) \tag{2}$$

$$\gamma = \frac{L(t)}{f(t - t_{shift})} \tag{3}$$

Furthermore, the costs for one resource instance per time unit is denoted with c_P. Using Formula (2) of the resource amount, it is possible to calculate the emerging costs C_P for the provider as follows.

$$\begin{aligned} C_P &= R(t) \cdot (t_{\chi^2} - t_{VM}) \cdot c_P \\ &\overset{2}{=} g \cdot \gamma \cdot f(t + t_{VM} - t_{shift}) \cdot (t_{\chi^2} - t_{VM}) \cdot c_P \\ &\overset{3}{=} g \cdot \frac{L(t)}{f(t - t_{shift})} \cdot f(t + t_{VM} - t_{shift}) \cdot (t_{\chi^2} - t_{VM}) \cdot c_P \end{aligned} \tag{4}$$

Looking at the user part, we identified c_U as the cost factor for the user, e.g., costs per working hours in a company when using the cloud service for business jobs. In this case, the load corresponds to the amount of users in the system at time point $t + t_{VM}$, because the sum of all user requests result in total costs C_U.

$$C_U = T_S \cdot c_U \cdot L(t + t_{VM}) \tag{5}$$

The server processing time T_S is determined by the distribution of workload and the provided resource amount. Assuming that load is modulated equally to all available resources, the processing time T_S can be represented as load $L(t + t_{VM})$ divided by the resource amount $R(t)$ multiplied with the job computation time T_J. Furthermore, we express the load function by $L\ (t + t_{VM}) = \gamma \cdot f\,(t + t_{VM} - t_{shift})$ and obtain this:

$$\begin{aligned}
C_U &\overset{5}{=} \frac{L(t+t_{VM})}{R(t)} \cdot T_J \cdot c_U \cdot L(t+t_{VM}) \\
&= \frac{L(t+t_VM)}{R(t)} \cdot T_J \cdot c_U \cdot \gamma \cdot f(t+t_{VM}-t_{shift}) \\
&= \frac{\gamma \cdot f(t+t_{VM}-t_{shift})}{R(t)} \cdot T_J \cdot c_U \cdot \frac{L(t)}{f(t-t_{shift})} \cdot f(t+t_{VM}-t_{shift}) \\
&\overset{2}{=} \frac{\gamma \cdot f(t+t_{VM}-t_{shift})}{g \cdot \gamma \cdot f(t+t_{VM}-t_{shift})} \cdot T_J \cdot c_U \cdot \frac{L(t)}{f(t-t_{shift})} \cdot f(t+t_{VM}-t_{shift}) \\
&= \frac{1}{g} \cdot T_J \cdot c_U \cdot \gamma \cdot f(t+t_{VM}-t_{shift}) \\
&\overset{3}{=} \frac{1}{g} \cdot T_J \cdot c_U \cdot \frac{L(t)}{f(t-t_{shift})} \cdot f(t+t_{VM}-t_{shift})
\end{aligned} \tag{6}$$

As a result, we created two opposing cost functions for the provider and the user. Both depend on the factor g and intersect in exact this point, as plotted in Figure 5. This point can be construed as optimal compromise between a generous way of resource allocation for high user satisfaction and a more reserved strategy for low costs at the provider. It is computable by the equation of the two cost formulas (4) and (6) $C_U = C_P$:

$$\frac{1}{g} T_J \cdot c_U \cdot \frac{L(t)}{f(t-t_{shift})} \cdot f(t+t_{VM}-t_{shift}) = g \cdot \frac{L(t)}{f(t-t_{shift})} \cdot f(t+t_{VM}-t_{shift}) \cdot (t_{\chi^2} - t_{VM}) \cdot c_P$$

Solving this equation for g, results in the following expression:

$$g = \sqrt{\frac{c_U}{c_P} \cdot \frac{T_J}{t_{\chi^2} - t_{VM}}} \tag{7}$$

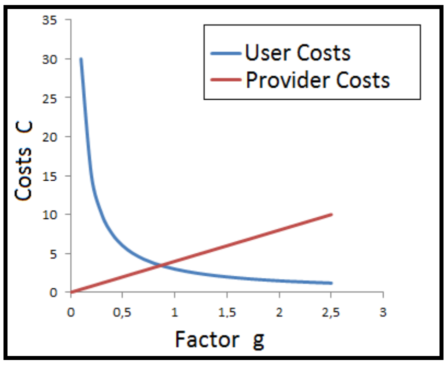


Figure 5: *Behaviour of the Cost Functions*

This way, a well-balanced proportion of required resources can be calculated dynamically. Thereby, we consider the cost factors of providers as well as users and the complexity of the application via the job computation time T_J. Additionally, the duration $t_{\chi 2}$ to the next chi-square-test, the booting time t_{VM} and the time shift between load- and tweet-curve are included. The current stored load value $L(t)$ and the tweet-curves $f\,(t - t_{shift})$ and $f\,(t + t_{VM} - t_{shift})$ can be extracted out of the data base. So, the optimal resource amount at time t can be determined:

$$R(t) = \sqrt{\frac{c_U}{c_P} \cdot \frac{T_J}{t_{\chi^2}}} \cdot \frac{L(t)}{f\,(t - t_{shift})} \cdot f\,(t + t_{VM} - t_{shift}) \tag{8}$$

Section 3 already states that scaling is only processed in case of a positive load pattern match on the test of slope changes. Based on this differentiation between rising and downward curves, we can decide whether to boot or shout down instances of the provided resource pool R regarding the above defined required resource amount $R(t)$. In case of increasing curves, the provider should boot additionally $\Delta R = R(t) - R$ resources and on decreasing load $\Delta R = R - R(t)$ can be shut down.

For clarification, imagine the following example scenario: the user's cost factor run up to c_U = 80 €/*h* and provider's cost factor is c_P = 4 €/*h*. The job computation time amount to T_J = 0.05 *s* and chi-square-test-period is set on $t_{\chi 2}$ = 1800 *s*. The data base stores a load of $L(t)$ = 300 and a tweet amount of $f\,(t - t_{shift})$ = 100 and $f\,(t + t_{VM} - t_{shift})$ = 150. Consequently, the required resource amount $R(t)$ and the processing time T_S can be calculated as below.

$$R(t) = \sqrt{\frac{80\text{ €}/h}{4\text{ €}/h} \cdot \frac{0.05s}{1800s}} \cdot \frac{300}{100} \cdot 150 = 0.02357 \cdot 450 = 10.607 \approx 11$$

$$T_S = \frac{L(t + t_{VM})}{R(t)} \cdot T_J = \frac{\gamma \cdot f(t + t_{VM} - t_{shift})}{R(t)} = \frac{\frac{300}{100} \cdot 150}{10.607} \cdot 0.05s \approx 2.121s$$

That means, at time point t the provider needs eleven resources in the treated region in order to handle the correlating workload to the tweet-rate. With eleven running instances in the corresponding region, we achieve a processing time T_S of a little bit more than two seconds.

6. Conclusion and Future Work

With cloud computing changing actual IT-environments, existing load management solutions are not suitable anymore because the infinity of resource pools claims for new defined handling. Constant availability of resources should be guaranteed for each client using the corresponding service. This is challenging because all clients have access to the same physical servers. Besides, instance scaling causes efforts regarding costs and time. This is mainly because cloud resource management, even today, claims for manual regulation [6, 7] and booting of additional machines requires up to 13 minutes [8]. Therefore, efficient, automated resource planning and rapid instance provisioning is exceedingly required.

This paper presents a solution for predicting future load while avoiding under- and over-provisioning of cloud offerings at a defined region. First of all, we identify a significant probability for load increases. For that reason, we use the data of the social Web application *Twitter* to achieve a load forecast on a particular service. This is possible as we can assume a correlation between tweet- and load-curves [10], identified via the chi-square-test with probability-thresholds freely selectable by the provider.

Secondly, by including user data, the main resource demand at a location can be mapped to an optimal hosting region. So, resources are instantiated at the main region of load and unnecessary latencies are omitted because of shorter routing distances. The simulation proves that response times can, indeed, be reduced if resources are located in the client's proximity by the improvement factor of *q*. Furthermore, we provide functions to calculate the exact amount of required resources as well as the costs for users and providers. This way, an optimized trade-off between low emerging costs and high user satisfaction can be determined. That leads to both, an optimal covering of resource demands and a reduction of the latency during service provisioning.

As future work, we will include the designed control entity for adding and releasing instances in a definite cloud computing environment and prove the functionality of the algorithm and the formulas via real data at an optimal compromise of low costs and simultaneous quality of service improvements.

References

[1] Armbrust M., et al., 2009: *Above the Clouds– A Berkeley View of Cloud Computing*. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

[2] Assunção M., Di Costanzo A., and Buyya R., 2009: *Evaluating the Cost-Benefit of using Cloud Computing to Extend the Capacity of Clusters*. 18th Int'l Symp. on High Performance Distributed Computing, NY, USA; 141-150.

[3] Pring B., et al., 2010: Access 2013: *Garntner Forecast: Public Cloud Services, Worldwide and Regions, Industry Sectors, 2009-2014*. ID Number: G00200833. http://www.g artner.com /id=1378 513.

[4] Chen J., Li W., Lau A., Cao J., and Wang K. *Automated Load Curve Data Cleansing in Power Systems*. IEEE Transactions on Smart Grid. 2010. 1; 213-221.

[5] Lucas J. Carrión C. and Caminero B., 2011: *Flexible Advance-Reservation (FAR) for Clouds*. 1st Int'l Conf. on Cloud Computing and Services Science, Noordwijkerhout, Netherlands; 610-615.

[6] Mao M. and Humphrey M., 2011: *Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows*. 24th Int'l Conf. for High Performance Computing, Networking, Storage and Analysis, NY, USA; 1-12.

[7] Alam K., Keresteci E., Nene B., and Swanson T., 2011: Access 2013: *Multi-Tenant Applications on Windows Azure– Documentation*. http://cloudninja.codeplex.com/releases/view/65798.

[8] Mao M. and Humphrey M., 2012: *A Performance Study on the VM Startup Time in the Cloud*. 5th Int'l Conf. on Cloud Computing, Honolulu; 423-430.

[9] Stackoverflow, 2011: Access 2013: *Why does Azure Deployment take so long?* http://stackoverflow.com/questions/5080445/why-does-azure-deployment-take-so-long.

[10] Wilder B., 2012: *Cloud Architecture Patterns: Using Microsoft Azure*. Sebastopol: O'Reilly Media Inc., 182.

[11] Gmach D., et al., 2006: *Dynamic Load Balancing of Virtualized Database Services Using Hints and Load Forecasting*. 17th Conf. on Advanced Information Systems Engineering, Atlanta, Georgia, USA.

[12] Seltzsam S., Gmach D., Krompass S., and Kemper A., 2006: *AutoGlobe– An Auto. Admin. Concept for Service-Oriented DB Applications*. 22nd Int'l Conf. on Data Engineering, Atlanta, Georgia, USA.

[13] Meddeb A. *Internet QoS– Pieces of the Puzzle*. IEEE Communications Magazine. 2010. 48; 86–94.

[14] Phillips D. and Young P., 2009: *Online Public Relations– A Practical Guide to Developing an Online Strategy in the World of Social Media*. PR In Practice, Kogan Page, 288.

[15] Wei K., Wen-wu D., and Lin W., 2011: *Research on Emergency Information Management based on the Social Network Analysis*. Int'l Conf. on Management Science and Engineering, Rome; 1302-1310.

[16] Leskovec J., 2011. Access 2013: *Rhythms of Information Flow through Networks*. http://vi deolectures.net/ eswc2011heraklion.

[17] Yang J. and Leskovec J., 2011: *Patterns of Temporal Variation in Online Media*. 4th ACM Int'l Conf. on Web Search and Data Mining, NY, USA; 177-186.

[18] Yang J. and Leskovec J., 2010: *Modeling Information Discussion in Implicit Networks*. 10th Int'l Conf. on Data Mining, Washington, DC, USA; 599–608.

[19] Schwanengel A., Hohenstein U., and Jaeger M.C., 2012: *Resource Allocation for Cloud SaaS Offerings based on Social Web Applications*. IEEE Int'l Conf. on Cloud Networking, Paris, France; 599-608.

[20] Kwak H., Lee C., Park H., and Moon S., 2010: *What is Twitter, a Social Network or a News Media?* 19th Int'l. Conf. on World Wide Web, NY, USA; 591-600.

[21] Geobytes, Inc., 2006. Access 2013: http://www.geobytes.com/iplocator.htm.

[22] Tanenbaum A., 2003: *Computernetzwerke*. Peearson Studium, 888.

[23] Andreß, H., 2003: *Chi-Quadrat-Verteilung*. Uni Köln, Lehrstuhl für Sozial- und Wirts chaf tsfor sc hung.