A SYSTEM FOR AUTOMATED LOAD ADAPTATION IN CLOUD COMPUTING ENVIRONMENTS

Anna Schwanengel, Michael C. Jaeger and Uwe Hohenstein
Siemens AG, Otto-Hahn-Ring 6, Munich 81739, Germany

Keywords: Monitoring of Services, Quality of Service, Service Level Agreements, Load Balancing, Cloud Architecture.

Abstract: Nowadays, cloud computing promises to supply a theoretically infinite resource amount, while enabling instance elasticity. However, using extra capacity requires organizational activities and leads to costs. To keep this overhead minimal, adding and releasing resources need to be well-scheduled. Therefore, it is inevitable to prepare an appropriate allocation automatism that considers differences in dynamics and price models of cloud computing. We present the idea for a system that effectively manages varying loads with regard to emerging costs, provisioning time, and customer service level agreements (SLAs). Contrary to existing, threshold-based solutions, our approach considers system observations oft he past, domain-specific load behaviours as well as external knowledge. That way, the system detects load patterns and adapts accordingly.

1 INTRODUCTION

With the increasing of service variety and available resource amount, distributed systems are expected to handle much load. In cloud systems, almost infinite pools of virtualized resource capacities enable appropriate covering of demands. When load raises, additional instances can be added in provider clouds by paying extra fees. Vendors predetermine different cost models and allow paying, e.g., per hours, data transfers or average resource quantity [1].

At the same time, customers prefer to keep accessory costs to a minimum. Besides, provisioning time is i.a. affected by dealing with queues, tables and compute instances. It must not take more time to allocate new instances than actual load peaks last. That raises the question when to add how many instances to meet load appropriately. Similarly, the inverse situation requires a solution: When should unused instances be released to save costs?

Challenges of load monitoring were also researched in the field of grid computing since years. However, with cloud computing some additional and even novel aspects appear. E.g., the reaction to an unpredictable increase of load is different with cloud applications. Though, consequent resource allocation can counteract a subsequent overload, it also leads delays in provider cloud. With cloud applications, instances are deployed at a certain cost and generate revenue, which requires deployment decisions in advance. Herein, established solutions for distributed system management do not offer optimal results.

We want to design a system that enables efficient load adaptation and management, while considering costs, provisioning time, and SLAs. With this solution, correlations through load patterns are identified, the design of representative load curves based on variance and slope is possible, and suitable algorithms to observe and describe load changes are offered.

This paper is organized as follows: Section 2 gives a short research overview of load behaviour. Section 3 motivates main challenges in cloud environments. Important load influencing aspects are illustrated in Section 4. Within Section 5, we describe a basic approach to achieve automated load adaptation. Section 6 discusses this solution and demonstrates further work. Finally, Section 7 concludes the paper.

2 RELATED WORK

Virtualization technology facilitates “cloud computing’s ability to add or remove resources” (Armbrust et al., 2009). E.g., if the current resource amount is not sufficient to handle all requests in a home cluster, additional VMs are instantiated at remote providers (Assunção et al., 2009). Thereby, customers can offer

[1] http://www.microsoft.com/windowsazure/offers
http://aws.amazon.com/de/ec2/#pricing

Schwanengel A., Jaeger M. and Hohenstein U..
A SYSTEM FOR AUTOMATED LOAD ADAPTATION IN CLOUD COMPUTING ENVIRONMENTS.
DOI: 10.5220/0003904605620567
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 562-567
ISBN: 978-989-8565-05-1

cloud services without wasting expensive resources by over-provisioning nor missing potential profits by under-provisioning (Armbrust et al., 2010). However, as studied in (Lucas et al., 2011), it is difficult to forecast exact resource amount a company needs to provide QoS to its clients, and 'end-to-end fees' are often not considered – we want to anchor this in our system.

The varying number of data and instances affect various load patterns and models. Therefore, (Katsaros et al., 2011) identify characteristics of cloud monitoring infrastructures and present an architectural approach. (Paton et al., 2009) define kinds of load in consideration of load properties, SLAs, and competition on the shared resources. Within workload execution, possible system states are mapped to a common scale, which can i.a. embody response times, QoS goals or throughput for requests. However, in contrast to our approach this work concentrates on limited instances, whereas we assume hypothetically infinite resource capacity in cloud environments.

To meet the lack of resources in case of load increasing, (Moran et al., 2011) propose a rule-based measure and control mechanism to allocate new instances. Though this is a reasonable starting point, they limit themselves on lower-level rule-mapping and do not address scalability issues entirely.

Also (Gmach et al., 2006) present approaches to react on load self-organizingly. Their 'AutoGlobe' system reacts on detected overload by distributing load and transferring tasks to less loaded servers. For services with periodic patterns, they formulate short-term load predictions and for abnormal happenings, they deduct hints on basis of, e.g., resource utilization (Seltzsam et al., 2006). However, they only address a short part of IT-business fields (ERP), whereas we want to examine a larger area of applications.

(Chen et al., 2010) propose forecasting that uses patterns to predict load at future time. That means trusting historical data to forecast future data values. Though trying to detect patterns, they concentrate on cleaning abnormal data, which is only a part of automated load adaptation in our point of view and the presented process of load cleansing is not automatic.

Similar to our approach, (Ferrer et al., 2012) investigate challenges for adaptive service provisioning in clouds – like cost-dependences, self-preservation and legislative issues. Unfortunately, their scenarios only concentrate on infrastructure and service providers, while ignoring the platform interlayering.

In summary, there are no clearly defined patterns to categorize load behaviour in different domains and parameters are not identified entirely. Aspects concerning load changes are to regard and trade-off between cost savings and SLA compliance is to solve.

3 PROBLEM STATEMENT

The problem, we want to tackle, is to handle various factors causing load changes in order to react on extraneous circumstances. In addition, we want to infer dynamic resource allocation in consideration of arising costs and negotiated customer SLAs. Unfortunately, established load treatments are not able to offer an optimal solution because of some facts, which are explained in more detail in the following. First of all, grid or high-performance computing systems and classical scheduling procedures can assume queues for the overall system. These queues buffer existing tasks and enable such processes to come to a decision based on existing facts. With missing such mechanisms in cloud offerings – which aim for public use – we need novel procedures on basis of forecasts.

Furthermore, existing algorithms for scheduling or load control in the Internet are based on best-effort approaches (Meddeb, 2010) or aspire guaranteeing service level agreements. In cloud environments, SLAs can be handled more flexible, and in specific situations they can even be expected to get violated because of beforehand negotiated compensation strategies (Vaquero et al., 2008). Therefore, SLAs must be adapted dynamically during actual service provisioning or violated SLAs have to be compensable. So it sometimes may be more expedient to not react on load changes, which means neglecting customer satisfaction for achieving cost savings.

An additional problem is the demand for an efficient and automated algorithm to detect whether new capacities are needed. Today, cloud providers only offer APIs (e.g., REST or SOAP) by which customers have to implement individual load management but no automated solution exists. We want to understand load characteristics, identify affecting parameters and consider cost aspects to create an automatism for load management and the corresponding adaptation.

4 LOAD INFLUENCERS

With load patterns varying by different application areas and even within one single application, it is difficult to define a generalized load behaviour based on collected observations in advance. But, as shown by an IBM analysis about the Olympic Games of 1998, pattern characterisation based on stored information is possible (Iyengar et al., 1999). We demonstrate a variety of load models by some examples and identify several load patterns and characteristics.

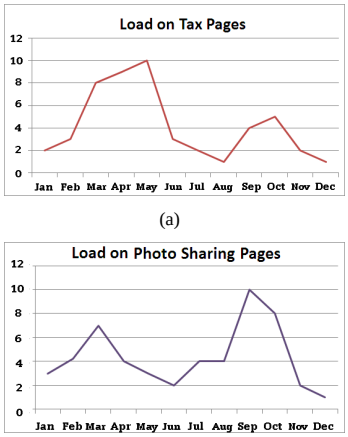

Figure 1: Load patterns on tax transaction (a) and photo sharing applications (b).

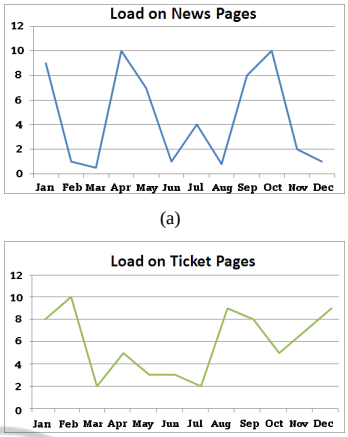

Figure 2: Random load models on news (a) and ticket applications (b).

4.1 Variety of Load Patterns

The first and most common load behaviour are growth situations, where load on required instances continuously increases over time. On the other hand, we find on-off situations (e.g. in batch job execution), where either a lot of instances are used to cover enormous load or only few resources are required because less load exists in the overall system.

As outlined in Figure 1 and 2, several load structures occur in everyday life with recurrent routine. Thereby, load is mostly low in average with high peaks in demand. Based on real life data, we assume that load peaks on tax servers arise around May following an annual cycle, because of tax return deadlines (Figure 1a). On photo sharing platforms most traffic is detected in March/April and September due to Alexa the Web Information Company [2] (Figure 1b). That fits to the surmise that people upload pictures after holidays to share vacation memories with friends. To conclude, these examples describe recurring yearly patterns, and histories can be generated.

We, further, identify processes following a cyclical predictable daily pattern, e.g., in use cases about electro mobility. During rush hours, an enormous amount of data will come up to management servers. When people drive to work, their cars produce numerous data, which are automatically transferred to a central management – such as telemetry data about car's technical conditions. Then, we can specify history-based absolute load patterns, and our forecasting system enables proactive actions to expected load increases before the actual occurrence.

On the other hand, there are cases, in which it is difficult to extract obvious patterns – e.g., predicting load behaviour on news applications – because events happen rarely and by pure chance. This type of data follows a random unpredictable scheme, as illustrated in Figure 2a. Based on Leskovec's statement at ESWC'11 that "peak intensity from blogs typically comes about 2.5 hours after peak intensity from news" (Leskovec, 2011), we assume that after relevant events awareness and further progression often behave in predictable manner with different intensity. Therefore, relative models with regarding variance and slope in load curves enable short-termed prediction of further progress after relevant events.

Furthermore, there exist events which – though, not following a cyclical annual model – can be precisely planned, as in the case shown in Figure 2b on servers selling tickets for concerts, football matches or other big and rare events. To detect precise patterns, information about external events are required.

We use these inputs for a prediction algorithm based on slope and variance in load curves. Within this model of unpredictable behaviour, the system has to define load characteristics in order to identify less important elements which can be ignored to give preference to more promising ones.

4.2 Load Curve Characteristics

Regardless of the domain, we determine characteristic load curves by their amplitudes and their climb. When such characteristics are observed in a non-recurring or at least in a seldom manner, it may be sensible not to react on load increases at all. If the additional machines need more time to come up than actual peaks remain, wasted over-provisioning is created with no use and unnecessary costs. Otherwise, if

[2] http://www.alexa.com/siteinfo/flickr.com

these short high amplitudes appear in a periodic manner with high frequency, additional instances have to be booted and should not be released in order to handle the sum of queued requests. In highly distributed environments often multi-dimensional SLAs exist as requirements – e.g., availability and reliability, low costs, high throughput and bandwidth, etc. So, not reacting must not violate predefined constraints!

Following these motivating thoughts, the problem turns in a multi-criteria optimization mission. This is challenging, because the parameters act in a contrarian manner: when requiring constant availability, one has to pay for permanent resource support. Furthermore, with a minimum of costs the customer calls for a maximum of resources with abidance of different service level agreements. Of course it is impossible to achieve these claims in entirety, but we want to get close to an implementation.

5 SYSTEM DESIGN

In the following of this section we describe how our system (see Figure 3) acts in:

- system observations
- identification of specific application domains
- integration of external knowledge
- load pattern detection and adaptation
- classification of load characteristics & parameters
- consideration about costs and SLAs
- reaction to load variations by different ways

Our approach intends to determine load based on domain specific aspects. The basic statement here is 'design for operation', which indicates developing applications based on different domains.

5.1 Interaction Models

As an attempt, our system takes account of the following specific domains. First of all, we differentiate between applications with machine-to-machine (M2M) communication and human machine interaction (HMI). M2M communication means less effort to handle protocols for consecutive operations than in case of user involvement. Consequently, future utilization can be predefined and pre-calculated with statements such as 'In x minutes the application y will require z more resources...' This performs worse with humans being involved, because system users often behave in a more unpredictable manner.

Another possibility is to specify categories of communications as transactional, bi-directional or uni-directional. This means identifying different communication ways to build up an assumption about load trends based on the amount of communications. Within transactional communication there are also differences regarding the complexity of execution. If the application is primarily designed to perform complicated business transactions, scalability and consistency are more difficult to handle than in systems based only on atomic transactions. These different interaction models will end up in clear load variations.

In this context, Alam et al. classify two groups of performance counters: one that show processing may go wrong in near future, and one that show the system status is already dissatisfying. The former are for example 'Requests Queued', 'Request Execution Time', 'CPU Usage', and 'Requests per Second'. "Having high numbers in these may indicate either the system is utilized at the most optimum level, or is really getting close to deteriorate" (Alam et al., 2011). 'Application Restarts', 'Worker Process Restarts', and 'Errors Total' are illustrations for the latter set of counters, which may result in the fact that the system is suffering from a resource scarcity.

Our assumption is that in the majority of cases the amount of waiting requests are sufficient to make a point about the load of the system. But we can also imagine cases, where applications exist with several request queues, so that one has to further observe, e.g., 'Request Execution Time', 'CPU Usage' or 'Application Restarts' in order to detect load variations.

5.2 Data Input for the System

Our system design covers the possibility to integrate external knowledge via stored observations. Out of a storage filled with information of beforehand announced events, the system can extract further important knowledge about proceeding load behaviour.

Besides external statistics, the system also considers cost parameters. Analysing the pricing models of relevant cloud computing providers, reveals that the monthly bill depends on various factors. Important are the numbers of reserved CPUs and time a virtual machine is occupied. Furthermore, one has to pay for the size of the database, respectively the amount of the used storage space. Besides, bandwidth and the sum of transactions are considered in billing. To conclude, there exist a lot of factors causing costs for the customer, which are dependent on load aspects produced by their applications in the cloud. Hence, we consider all influencing load parameters and match them to an entire assumption of load in order to precise statements about upcoming load and costs. Thereby, it is easier to detect a reasonable amount 'z' of resources.

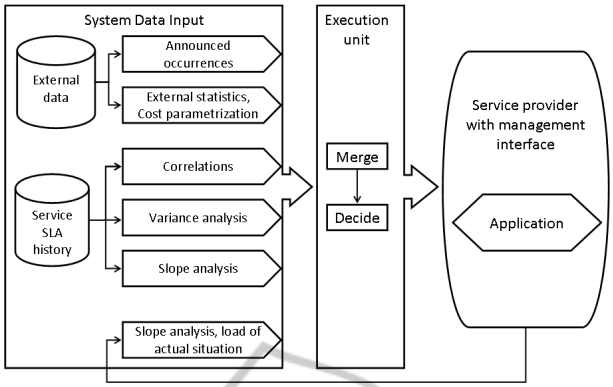

Figure 3: Overview of the system components.

Further, the system considers service level agreements and implies also modelling for the optimization problem with diametrically opposing attributes. Low costs and high reliability, strong consistency and permanent availability, high distribution and less administrative effort are only some of the contrary elements in this setting and one has to trade off which attribute to prefer (Das and Panigrahi, 2008).

The system identifies correlations through load patterns and differentiates between load curves that feature continuous increases, short high peaks in either a frequently recurring, a one-time / seldom manner or constant sinus amplitudes.

We, additionally, integrate a solution to design a representative curve based on variance and slope. With drawing out a pattern, which enfolds aspects as peak intensity, time-shifting, slope of load after a detected increase, etc., the systems succeeds in using suitable algorithms to observe and describe load changes. If a change of the situation is detected – e.g., the storage is filled with new external knowledge or the application does not behave as predicted – the pattern is re-defined in order to fit to load changes.

Beyond, the system defines load characteristics in order to identify less important elements, which can be ignored to give the preference to the more promising ones. Within these considerations, parameters influencing load curves can be identified and anchored inside our system design.

5.3 Execution Unit

As already stated before, it is essential to define how to react on load changes based on overall system information. If short load amplitudes are measured, they might be ignored, because costs and administrative effort bear no relation to increased value or higher profit. On the other hand, provided no one-time appearance of load increasing, but cyclical load escalations it is inevitable to act in order to preserve customer demands. These considerations are important for the execution unit, which merges all collected information and decides how to (re-)act.

Handling load variations, there exist two potential ways: foresighted acting in an absolute way based on measured and recorded data and reacting to load changes only afterwards. The former implies the need for a predefined model on which a proactive act is performed already before a load increase is measured. Presuming load to find a solution for unpredictable behaviour, turns in the online problem. Coming up from online-algorithms, one has to take an upfront decision according to the present unprecised knowledge, on condition of never paying more than if one would have known the future.

We differentiate between a model based on mathematical analyses or simulations, and a model which deals systems as black boxes and tries to deduct behaviours based in statistics or data mining (Machiraju et al., 2004). When reacting to load changes only in hindsight, one has to rely on measurements and ad-hoc reactions in a quick way.

Further, we want to enter cost and SLA information into the system. For this purpose, the system extracts the stored SLA history and takes different cost models into account. For example, if the agreement relating to costs provides to pay by hours, the system will not release the booted instance too early.

Combining all these considerations, we will be able to implement the system concept, which provides an automated load adaptation regarding costs, provisioning time and SLAs.

6 FUTURE WORK

Load modelling based on domain analysis is a complex approach. Existing various applications, the possibilities for precise load pattern detection are not re-

searched exhaustively. This raises questions concerning measurement and influencing parameters on load behaviour, on which we concentrate on in future.

We want to decide how to measure and model load and which load characteristics should preponderate. Furthermore, inferring from a model, how single services influence the total load in a multi-tenant cloud environment, implies a lot of previously collected and detailed knowledge. Thereby, we identify as an important task to minimize the amount of resources in order to save money without violating SLA aspects as availability, reliability or throughput.

As future work, we want to extract precise load patterns from cloud simulation environments, relevant literature and other observations and to develop appropriate algorithms to react adequately and automatically to load changes. The research fields of dynamic scalability and general load monitoring and load management will also be taken into account.

7 CONCLUSIONS

Load management is a long-standing issue in several computing areas but cloud computing generates new aspects. In contrast to existing grid management solutions, one has to deal with infinite resources, and flexibility increases because of elasticity. Furthermore, SLAs may now be treated less strictly and a violation can be condone in cloud environments for cost savings. The basis for solutions of 'computing problems' has changed and is to be redefined.

Within this paper we differentiate several contributors in the large world of load management in cloud computing. The appropriate reaction on load variances will be essential in competition about a dominate position on customer market. Handling SLAs in a flexible and reliable manner while satisfying customer expectations, optimizing provisioning times, and reducing costs by responsible resource decrease are important factors in this setting.

Proposing this, we make a step towards identifying different load patterns, which will be proven by pattern mining, and categorize their load behaviours in a domain-specific manner.

In future, we will invent a control entity for adding and releasing instances at an optimal compromise of low cost and simultaneous SLA compliance with including an algorithm for load pattern detection. Additionally, we will implement our system design and prove its functionality via real data concerning minimal costs, high availability, and scalability.

REFERENCES

Alam, K., Keresteci, E., Nene, B., and Swanson, T. (2011). Dokumentation. http://cloudninja.codeplex.com/releases/view/65798.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., and Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. *Berkeley Uni California*.

Armbrust, M., Stoica, I., and et al. (2010). A view of cloud computing. *Communication ACM*, 53.

Assunção, M., Di Costanzo, A., and Buyya, R. (2009). Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. *18th HPDC*.

Chen, J., Li, W., Lau, A., Cao, J., and Wang, K. (2010). Automated load curve data cleansing in power systems. *IEEE Computer*, 1.

Das, S. and Panigrahi, B. (2008). Multi-objective evolutionary algorithms, encyclopedia of artificial intelligence. *Idea Group Publishing*.

Ferrer, A. J., Hernndez, F., Tordsson, J., and et al. (2012). Optimis: A holistic approach to cloud service provisioning. *FGCS*.

Gmach, D., Krompass, S., Seltzsam, S., Wimmer, M., and Kemper, A. (2006). Dynamic load balancing of virtualized database services using hints and load forecasting. *ICDE`06*.

Iyengar, A. K., Squillante, M. S., and Zhang, L. (1999). Analysis and characterization of large-scale web server access patterns. *World Wide Web*.

Katsaros, G., Gallizo, G., Kübert, R., Wang, T., Fitó, J. O., and Henriksson, D. (2011). A multi-level architecture for collecting and managing monitoring information in cloud environments. *1st CLOSER*.

Leskovec, J. (2011). Rhythms of information flow through networks. http://videolectures.net/eswc2011_heraklion/.

Lucas, J. L., Carrin, C., and Caminero, B. (2011). Flexible advance-reservation (FAR) for clouds. *1st CLOSER*.

Machiraju, V., Bartolini, C., and Casati, F. (2004). Technologies for business-driven IT management. *Extending Web Services Technologies*.

Meddeb, A. (2010). Internet QoS: Pieces of the puzzle. *Communications Magazine*, 48(1).

Moran, D., Vaquero, L., and Galan, F. (2011). Elastically ruling the cloud: Specifying application's behavior in federated clouds. *4th CLOUD*.

Paton, N., De Aragão, M., Lee, K., Fern, A., and Sakellariou, R. (2009). Optimizing utility in cloud computing through autonomic workload execution. *IEEE*.

Seltzsam, S., Gmach, D., Krompass, S., and Kemper, A. (2006). AutoGlobe: An auto. admin. concept for service-oriented DB applications. *17th CAiSE*.

Vaquero, L., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: Towards a cloud definition. *SIGCOMM*.