Optimizing resource topologies of workload in the cloud by minimizing consumption and maximizing utilization while still meeting service level agreements

Cataldo Mega,
ECM Cloud Services Development
IBM Germany Research & Development,
Schönaicherstraße 220,
71032 Boeblingen, Germany
(cataldo_mega@de.ibm.com)

Christoph Lange
ECM Cloud Services Development
IBM Germany Research & Development,
Schönaicherstraße 220,
71032 Böblingen, Germany
(chrisla@de.ibm.com).

Abstract: Reducing the total cost of ownership (TCO) and meeting the service level agreements (SLA) of a computing service or application in the Cloud remains a challenge. The traditional method of meeting SLA is to size production systems based on expected peak workloads, thus this leads to very low compute resource utilization. Our experience shows typical average utilization rates of 35 - 40%, topped only by IBM Mainframe systems with utilization rates of up to 80 - 85%. In the 'Cloud' with an inherent multi-tenant environment resource utilization rates of below 40% translates into a prohibitive cost factor and therefore into a non-competitive business model.

In this paper we discuss the dynamics of a policy based automated resource orchestration approach for servicing multi-tenant workloads in the Cloud. Our aim is to adapt system topologies dynamically based on changes in workload at runtime. The condition for being able to achieve these goals are: 1) a composable, self-tuned infrastructure (smart IaaS), 2) automatic coordination and provisioning of managed middleware (smart PaaS) and last but not least 3) elastic, self-driven orchestration of software defined workloads and system topologies (smart SaaS).

Introduction

Companies are starting to embrace and adopt the "cloud" delivery model for managed enterprise services i.e. in the Business-to-Business (B2B) market. Companies moving their on premise applications off-premise into the cloud are trying to reduce their total cost of ownership (TCO) while maintaining service level agreements (SLAs) the same or even improve. Cloud services like webmail and cloud storage originate from the consumer market and typically utilize a pricing model different from what is common in

B2B. Pay per use pricing models from business to consumer (B2C) scenarios are becoming a favorite model of every Chief Financial Officer (CFO) seeking to reduce capital expenditure (CAPEX). However, the new business model comes at the cost of integration rework and optimization of the solution for adjusting to the changed ecosystem when moving existing solutions into the cloud. In general, moving complex enterprise applications from an on premise to an off-premise production context is not a trivial task. The Cloud enforces a paradigm shift that affects the way software solutions are developed, delivered and operated. Providers of enterprise services must consequently cope with this new challenge as software services must be able to communicate with the underlying platform and infrastructure to dynamically request or release components as required, including the orchestration of resource topology changes[1] at varying workloads.

Characteristics of Software Defined Environments

Traditionally software is run on proprietary hardware using custom-built resource topologies i.e. a physical and logical layout of the components and their configuration. In software defined environments such as [8], [9] it is possible to use modeling languages such as the "Topology and Orchestration Specification for Cloud Applications" [10] allow the standards based definition of topologies and services as well as their dynamic management. This is in contrast to purpose built topology adaption methods based on the integration with proprietary Infrastructure-as-a-Service (IaaS) models. Enabling a dynamic automated system topology adaption requires all topology components and resources to be programmatically reconfigurable during runtime using new open standard cloud technology. This includes common infrastructure components such as network, compute, memory and storage (IaaS) elements, also platform (PaaS) components such as database schemas, application server, full text indexing and search engines, all the way up to software solution (SaaS) components like Content and Process Engine, Content Navigation Web-Clients, eDiscovery, Retention and Records Managers.

At the IaaS level scale is achieved by adding infrastructure elements but if the application component itself is saturated this does not yield the desired effect as adding new compute resources would not eliminate the bottleneck. With software defined environment application resources are considered and dynamically added or removed allowing a solution specific fine granular resource allocation. This approach lends itself more effectively to adapt the topology for a given SaaS workload.

2.1 Characteristics of ECM workloads

In order to validate our method we use a typical Enterprise Content Management (ECM) solution. ECM solution components include Business Support Services (BSS) and their respective administrative clients most of which are Java Enterprise Edition (Java EE) applications running on a Java EE application server complemented by a database and a filesystem among other software based topology components.

A workload is described by the actions of an end user population using one or more applications to perform a typical series of operations against a system. The set of operations and the sequence in which they are performed is called a workload pattern. The overall system workload consists of the aggregated individual workloads of all actors using the system. Service Level Agreements (SLA) specifies the Quality of Service (QoS) upon which the service provider and consumer [2] do contractually agree. SLA are then split into system component specific service-level objectives (SLO). SLO define system performance characteristics that must holistically be met by all components to sustain the workload at hand.

A typical ECM workload consists of a mix of CRUDS operations i.e. Create; Retrieve; Update; Delete; and Search presented in three different production contexts, interactive, batch and the administrative. Within a given context the assumption is that individual operations must process a request within a given time thus the KPI is 'response time' or 'throughput' or both. Examples of other type of KPIs are: document index time latency, archive capacity, storage size, system availability (HA), recovery time objectives (RTO), recovery point objective (RPO)… The cloud infrastructure is therefore forced to serve the different tenant-specific workload by translating each workload into a resource topology that defines the actual system layout. Under the above conditions and in order to save costs, as many resources as possible must be shared, which means that we have a resource optimization problem to solve.

Similar to a database query optimizer that is used to produce an optimized access plan in our case we need to develop a resource topology optimizer that produces an optimized topology change plan. The problem to solve can be defined as:

"What is the resource topology that best satisfies the load predicted and what are the topology changes required to get there."

The answer we seek consist of an execution plan specifying the necessary steps to chance the current resource topology considering the elasticity [1]of the cloud environment and that is executed by an orchestrator during runtime such for being able to define the system topology that best satisfies the given load, optimize utilization while minimizing consumption.

We examined three current approaches to solve this problem [3]: 1) utility based optimization techniques, 2) machine learning adaptive techniques and a 3) feedback loop based on control theory.

Utility-based optimization techniques are based on a performance model of a system. Creating such a model is a very complex and inflexible procedure. It implies longer times of the system to react to topology changes thus resulting in a reduced efficiency. As an example, a model based on queuing theory is used e.g. in [4].

The disadvantage of machine learning adaptive techniques is that they usually require several training hours. Furthermore, examples using this approach often assume a

[1] Note: In this paper, we define elasticity as the 'time to execute a topology change'.

maximum set of applications. Thus, an unlimited scalability of a service would not be possible. Examples for machine learning adaptive approaches are found in [5] and [6].

We have chosen to implement procedure that utilizes a feedback loop based on control theory employing a Monitor-Analyze-Plan-Execute (MAPE) concept and have adapted it to use heuristics derived from measured system performance base lines[7]. MAPE is a generic control-loop concept that uses 4 phases to control a target system. In the first monitor phase certain metrics of the system are measured and afterwards in the analyze phase compared to set baseline values. The plan phase then uses the result of this analysis to compile a plan of execution steps that adjust the system topology to meet the target values. Finally, this plan is executed in the final phase by an orchestrator that drives provisioning and de-provisioning workflows.

Gathering heuristics from performance measurements

As a first step a reference system and a normalized workload are defined. In our tests we used a typical enterprise content management solution i.e. compliance archive applications. The resource topology is defined in terms of domain specific resources (nodes), the relation among them (edges) and the hierarchy that describes a production system. The resource nodes cover infrastructure (IaaS), platform (PaaS) and solution entities (SaaS). In our case the common infrastructure components such as network, compute, memory and storage elements, then platform elements such as database schemas, application server, full text indexing and search engines, and finally software solution elements like Content Engine, Process Engine, Archive Web-Client, eDiscovery, Retention and Records Managers complemented by common Business Support Services (BSS). For that reason, we create an ECM specific vocabulary and a Domain Specific Language used as an abstraction suited for the domain considered. The ECM DSL was then complemented by a numerical normalization based on resource type and amount of capacity that the respective resource provides. This allows us to transfer measured performance data to other, even different systems and cloud environments.

Consider the following use case: A user logs on to the ECM application, navigates to her workplace and opens the workbasket. In the next step, she picks a work item e.g. an insurance claim, retrieves the document and starts reading the claim. A few minutes later, she opens the search user interface and performs a search against the ECM repository in order to find collateral documents relevant to the claim. The search yields a hit list with 20 documents, out of which the user selects to retrieve three, one after the other and annotates the 3rd, updating the document. After many iterations of the same mix of operations she eventually logs out of the system.

By using this and other representative usage scenarios a set of automated performance tests are developed that simulated end user behavior. During the performance test runs KPIs are measured and performance baselines created for all primitive operations. Then a final analysis is conducted to determine upper and lower threshold, saturation points and relationships among resource types utilizing best practices and experience. The outcome of the analysis is knowledge characterizing the system from which provisioning policies are derived that help the topology optimizer find a satisfactory resource

topology in reasonably short time. Figure 1 shows an overview of system performance graphs, the performance baselines measured and a few of the characteristic points used to derive heuristics and rules. It also outlines the topology optimization process.

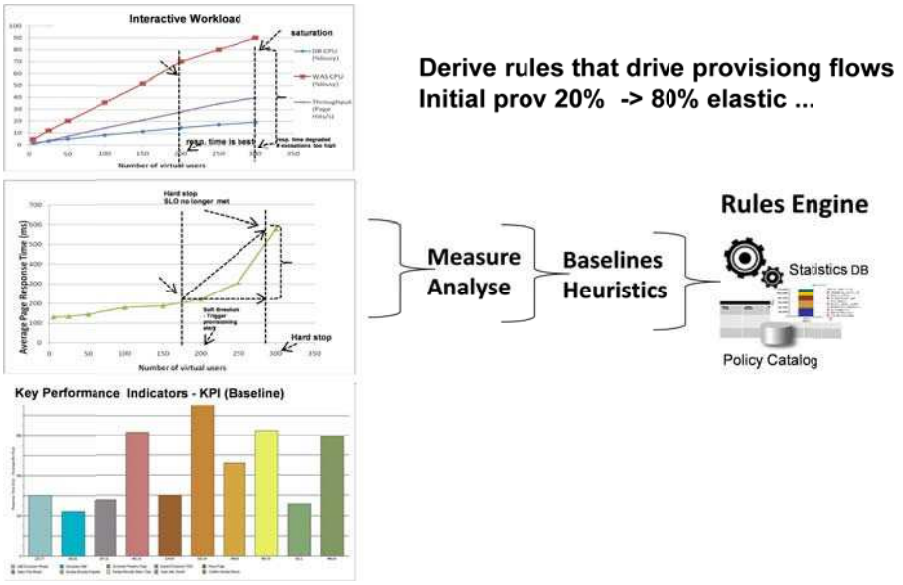

Figure 1 Heuristics are translated in provisioning policies.

Performing topology changes using heuristics

The optimizer uses the heuristics and rules to determine the optimal topology and to create a plan of the steps required to perform the changes. The orchestrator executes the plan and drives the creation of the new production system, as seen in Figure 2.

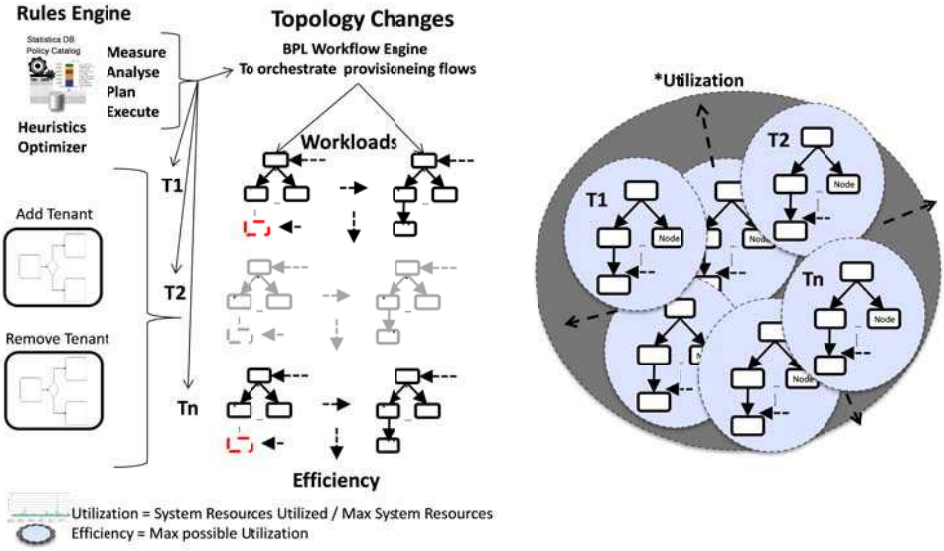

Figure 2 Reference System Topology and plans to apply elasticity for multiple tenants

In summary the MAPE-loop is continuously executed. The information collected in the monitor phase is used to predict how much time the service provider has to provision new resources since he is able to derive the slope of the workload from the last samplings until the most recent one.

Evaluation and Findings

In order to evaluate our approach we have set up a reference system running the Enterprise Content Management (ECM) cloud application and performed KPI measurements in order to derive performance baselines. Our ECM-based reference cloud-application provides a compliance archive as a Service for documents and emails in the cloud. Its three-tier Database, Application, Web architecture makes it well suited for our dynamic provisioning approach since the individual components can be scaled independently of each other.

The relevant components for our evaluation are a Web2.0 Client Application, the Batch Loader and an service administration client. The Web 2.0 Client Application serves as a graphical user interface for interactive access to the content in the cloud allowing customers to store and retrieve documents. The Batch Loader is used to ingest large amounts of documents according to batch-job definitions while the administrative client

application allows customers to configure and manage the whole system from a single user interface. The test system was loaded with several millions of documents of varying types and with sizes. The following performance baselines resulted from our initial interactive Web-Client tests.

The key baseline we measured by running our tests is shown in Figure 3.

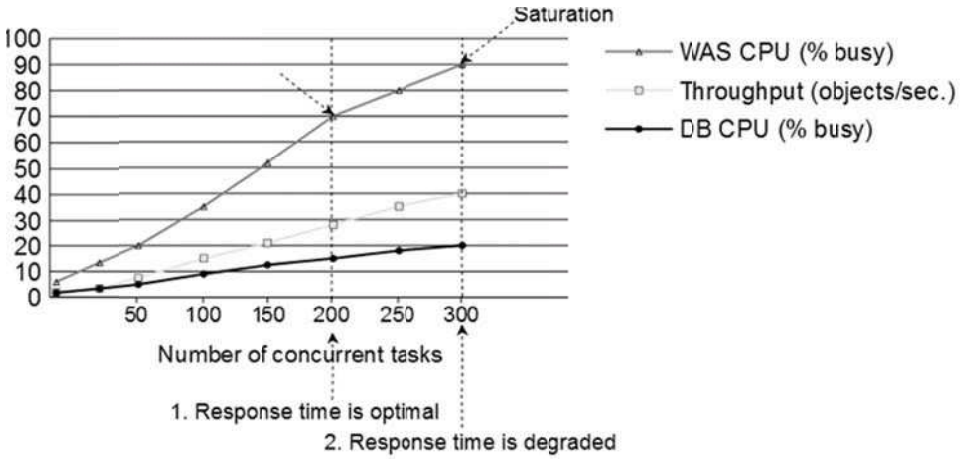

Figure 3. Batch workload performance. (WAS: WebSphere Application Server; DB: Database)

The term task on the x-axis represent a "virtual users". Every "virtual user" performs a certain amount of operations following a specific profile. The curve shows that at 300 simulated users, the WebSphere application server (WAS) on the reference system reaches saturation. Response times suffer and the service level objectives can no longer be met for this component. We therefore conclude that when the system hits 300 concurrent virtual users another application server needs to be deployed. From the picture above, one can also observe that at 200 concurrent virtual users the system load on the database system is below 20% CPU, therefore the database to application server CPU-ratio is ~ 1:3, leaving a CPU reserve to compensate load spikes.

From this we learn that a 2 CPU database node can serve up to 3 application sever nodes running the interactive workloads. Response time measurements also show that at 200 concurrent users the response time is optimal in relation to one application server, smaller load oscillations can be compensated and application server CPU reserves are sufficient for the Operating System (OS) to run smoothly.

Conclusion

The novelty of our approach is to: a) measure performance baselines on a reference test system using representative workloads, b) derive rules that describe the relationships (edges) among the resources (nodes) in the topology graph and c) use the heuristics learned to compute the optimized topology and the execution plan perform the topology changes. Finally an orchestrator is introduced for orchestrating resource topologies

changes and to programmatically create the system environment capable to sustain a given workload that fulfills service level agreements (SLA).

The benefit of our approach is to enable the sharing of resources between tenants on systems with different vertical layers of abstraction (virtual machine, application server, application component, etc.). Resources capacity is computed for tenant specific workloads at a fine granular level allowing SaaS providers to minimize the resources necessary to provide an SLA-compliant service offering and to reduce cost by eliminating unnecessary resources.

References

[MWLSB01] Mega, C. ; Waizenegger, T. ; Lebutsch, D. ; Schleipen, S. ; Barney, J.M. "Dynamic cloud service topology adaption for minimizing resources while meeting performance goals" [in IBM Journal of Research and Development Vol. 58 NO. 2/3 Paper 8 March/MAY 2014

1. F. Leymann, "Cloud Computing: The Next Revolution in IT",[in Proc. Photogramm. Week, 2009, vol. 52, pp. 3–12.
2. A. Paschke and E. Schnappinger-Gerull,"A categorization scheme for SLA metrics, [in Proc. Multi-Conf. Inf. Syst., 2006, vol. 80,pp. 25–40.
3. D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-aware autonomic resource allocation in multi-tier virtualized environments",[IEEE Trans. Serv. Comput., vol. 5, no. 1, pp. 2–19, Jan.-Mar. 2012.
4. B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Agile,dynamic provisioning of multi-tier Internet applications",[ACM Trans. Auton. Adap. Syst. (TAAS), vol. 3, no. 1, pp. 1:1–1:39,2008.
5. S. Islam, J. Keung, K. Lee, and A. Liu, "An empirical study into adaptive resource provisioning in the cloud",[in Proc. IEEE Int. Conf. UCC, 2010, p. 8.
6. J. Parekh, G. Jung, G. Swint, C. Pu, and A. Sahai, "Comparison of performance analysis approaches for bottleneck detection in multi-tier enterprise applications",[in Proc. IEEE Int. Workshop Quality Service, 2006, pp. 293-1–293-9.
7. A. Boerner, "Orchestration and provisioning of dynamic system topologies", [M.S. thesis, University of Stuttgart, Stuttgart, Germany, 2011.
8. The Open Networking Foundation standard. [Online]. Available: http://www.opennetworking.org
9. The OpenStack Project. [Online]. Available: http://www.openstack.org
10. The OASIS-TOSCA standard. [Online]. Available: https://www.oasis-open.org/committees/tosca/
11. IBM Corporation, "An architectural blueprint for autonomic computing", [in Autonomic Computing White Paper,2006. [Online]. Available: http://www-03.ibm.com/autonomic/pdfs/ AC%20Blueprint%20White%20Paper%20V7.pdf
12. A. Arefin, V. Singh, G. Jiang, Y. Zhang, and C. Lumezanu, "Diagnosing data center behavior flow by flow,[in Proc. 33rd IEEE Int. Conf. Distrib. Comput. Syst., 2013, pp. 11–20.
13. C. Mega, F. Wagner, and B. Mitschang, " From content management to enterprise content management, [in Proc. Datenbanksysteme für Business, Technologie und Web (BTW), 2005, pp. 596–613. 14. C. Mega, K. Krebs, F. Wagner, N. Ritter, and B. Mitschang, B Content-anagement-systeme der na¨chsten generation,[in Proc Wissens- und Informationsmanagement; Strategien,Organisation und Prozesse, 2008, pp. 539–567.

15. T. Ritter, B. Mitschang, and C. Mega," Dynamic provisioning of system topologies in the cloud,[in Proc. I-ESA, 2012,pp. 391–401.
16. H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent workload factoring for a hybrid cloud computing model,[in Proc. Congr. Serv., 2009, pp. 701–708.
17. T. Waizenegger, O. Schiller, and C. Mega, "Datensicherheit in mandantenfaehigen Cloud Umgebungen",[in Proc.Datenbanksysteme fuer Business, Technologie und Web (BTW), 2013, pp. 477–489.
18. F. Wagner, K. Krebs, C. Mega, B. Mitschang, and N. Ritter, „Towards the design of a scalable email archiving and discovery solution",[in Proc. 12th East Eur. Conf. Adv. Databases Inf. Syst., 2008, pp. 305–320.
19. F. Wagner, K. Krebs, C. Mega, B. Mitschang, and N. Ritter, "Email archiving and discovery as a service",[in Proc. Intell. Distrib. Comput., Syst. Appl., 2008, pp. 197–206