Bachelor Thesis / Master Thesis

Autoscaling Real-Time 3D Applications on Elastic Cloud

This work is planned to be carried out in cooperation with the company LynxStep GmbH[1]. LynxStep works in the area of interactive web 3D media, Cloud infrastructure, and Service Oriented Architectures (SOA). They are experts in providing high-quality interactive 3D web applications that run on modern Cloud infrastructures.

In our recent work[2] we have studied high-performance Real-Time Online Interactive Applications (ROIA), with use cases like product configurators in the Configure-Price-Quote market, E-learning, and multiplayer online gaming. While core components of ROIA, such as interactive real-time 3D rendering, still widely run on local devices, it is very desirable to run them on cloud resources to benefit from the advantages of cloud computing, e.g., better quality provided by high-performance compute resources and accessibility[3]. Therefore, we designed and implemented a novel Cloud service deployment architecture for ROIA called Scalable Real-time Service Deployment (SRSD), which addresses three major challenges: meeting the high Quality of Service (QoS) requirements (especially scalability and responsiveness), auto-scalability, and resource usage optimization.

This proposed thesis should study, implement and evaluate different prediction algorithms for autoscaling 3D applications on elastic Cloud. There are currently two main approaches to predict the workload. The first one is using the time series approach to analyze the monitoring data. The second one is using the machine learning approach to predict the workload[4]. In this thesis only the time series approaches will be studied. An appropriate prediction algorithm will significantly increase the efficiency of resource usage. We expect this to be especially valuable for high-performance ROIA since infrastructure costs are a fundamental aspect of ROIA, which employ high-performance computing components like real-time 3D rendering.

Container orchestration systems, e.g., Kubernetes, provide service deployment architectures with integrated load balancing as well as autoscaling services. However, such solutions are based on classical metrics, e.g., CPU, GPU, and memory usage and therefore are not primarily designed to deal with the challenges of ROIA[2]. In contrast, our approach is based not on classical metrics but rather on the session slots concept that combines a high level of QoS with the economic use of resources like CPU, GPU, and memory. The concept of session slot describes a hard limit on the number of concurrent

[1] https://lynxstep.com

[2] Jarrous-Holtrup S., Schamel F., Hofer K., Gorlatch S. (2021) A Scalable Cloud Deployment Architecture for High-Performance Real-Time Online Applications. In: Jagode H., Anzt H., Ltaief H., Luszczek P. (eds) High Performance Computing. ISC High Performance 2021. Lecture Notes in Computer Science, vol 12761. Springer, Cham. https://doi.org/10.1007/978-3-030-90539-2_26

[3] https://www.w3.org/standards/webdesign/accessibility

[4] Yazhou Hu, Bo Deng, Fuyang Peng and Dongxia Wang, "Workload prediction for cloud computing elasticity mechanism," 2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), 2016, pp. 244-249, doi: 10.1109/ICCCBDA.2016.7529565.

user sessions running on a service instance without loss of QoS[5,6]. Therefore, if we are able to predict the number of concurrent users of the system at any time, i.e., predicting the number of required free session slots, then we can launch just enough ROIA-instances to handle the user requests without loss of QoS while keeping the resource usage as low as possible.

Fig. 1 shows the core components of our deployment architecture: The ROIA are implemented as a service, packaged as a Docker container image and instantiated inside pods on a Kubernetes cluster. A scalable Session Slots Database (SSD) and a scalable stateless Session Slots Orchestrator (SSO) replace traditional load balancers. SSD and SSO manage the session slots, sessions, and they also collect session slots usage data. A scalable stateless Session Slot Autoscaler (SSA) permanently evaluates the history of session slot usage, collected by the SSO and stored in the SSD, and launches and terminates nodes and pods (service instances) in the cluster accordingly.

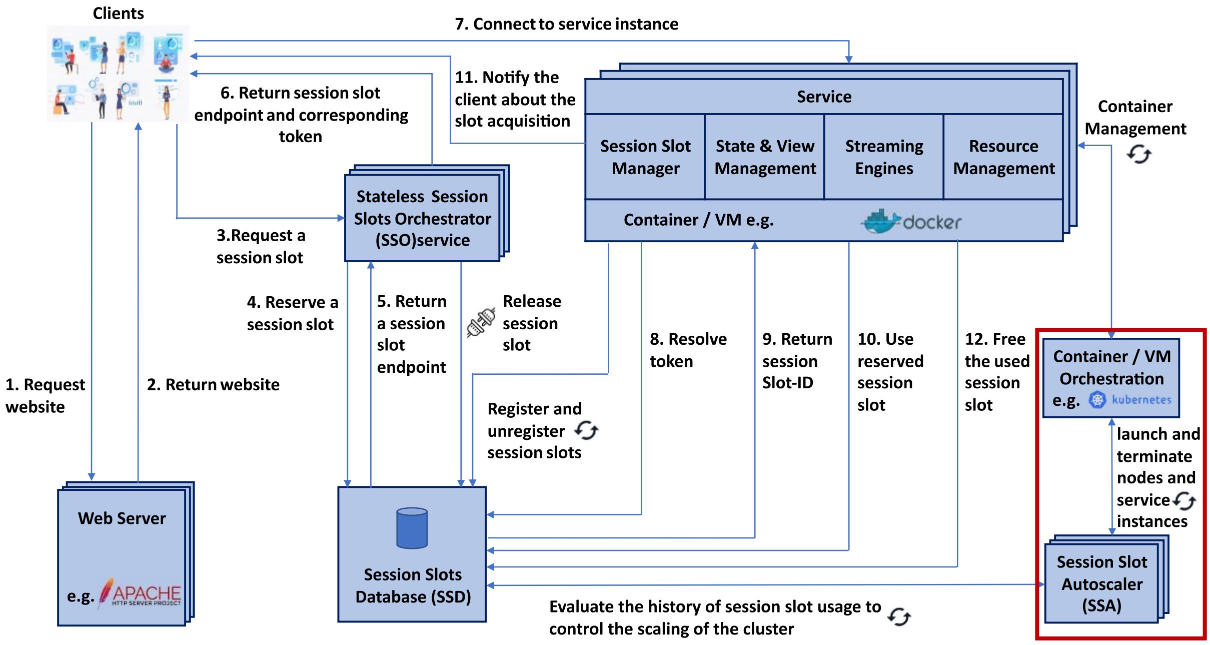
Clients
7. Connect to service instance
6. Return session slot endpoint and corresponding token
11. Notify the client about the slot acquisition
Service
Session Slot Manager
State & View Management
Streaming Engines
Resource Management
Container / VM e.g. docker
Container Management
Stateless Session Slots Orchestrator (SSO)service
3.Request a session slot
4. Reserve a session slot
5. Return a session slot endpoint
Release session slot
8. Resolve token
9. Return session Slot-ID
10. Use reserved session slot
12. Free the used session slot
Container / VM Orchestration e.g. kubernetes
1. Request website
2. Return website
Register and unregister session slots
launch and terminate nodes and service instances
Web Server
e.g. APACHE HTTP SERVER PROJECT
Session Slots Database (SSD)
Evaluate the history of session slot usage to control the scaling of the cluster
Session Slot Autoscaler (SSA)

Fig. 1. The overall architecture of SRSD.

The SSA component uses the Kubernetes REST API to query and manipulate the state of API objects in the Kubernetes cluster, specifically launching and terminating pods and nodes. In particular, it predicts the future session slot requirements by regression analysis of the recent usage patterns, taking into account the deployment and startup time of new service instances. Within the scope of this work the SSA component should be improved and optimized. Thereby, different prediction algorithms for autoscaling 3D applications on elastic cloud should be investigated, implemented and evaluated.

[5] Griffin, D., Rio, M., Simoens, P., Smet, P., Vandeputte, F., Vermoesen, L.,Bursztynowski, D., Schamel, F.: Service oriented networking. In: 2014 European Conference on Networks and Communications (EuCNC). pp. 1–5 (2014). https://doi.org/10.1109/EuCNC.2014.6882684

[6] F. Vandeputte et al., "Evaluator services for optimised service placement in distributed heterogeneous cloud infrastructures," 2015 European Conference on Networks and Communications (EuCNC), 2015, pp. 439-444, doi: 10.1109/EuCNC.2015.7194114.

In the course of this work, the candidate will gain knowledge and experience in the following areas: Cloud infrastructure, container technologies (Docker), container orchestration systems (Kubernetes), autoscaling in Cloud environment, and Service Oriented Architectures (REST).

Necessary skills:

- The necessary skills can also be acquired in the course of work
- Basic knowledge in Python
- Basic knowledge in cloud computing

Desirable skills:

- Experience with distributed systems
- Experience with Kubernetes and Docker
- Basic knowledge in SOA