Towards a Cloud Robotics Platform for Distributed Visual SLAM

Peng Yun[1], Jianhao Jiao[2], and Ming Liu[2(✉)]

[1] Department of Computer Science Engineering, HKUST, Kowloon, Hong Kong
pyun@ust.hk
[2] Department of Electronic and Computer Engineering, HKUST, Kowloon, Hong Kong
{jjiao,eelium}@ust.hk

Abstract. Cloud computing allows robots to offload computation and share information as well as skills. Visual SLAM is one of the intensively computational tasks for mobile robots. It can benefit from the cloud. In this paper, we propose a novel cloud robotics platform named RSE-PF for distributed visual SLAM with close attention to the infrastructure of the cloud. We implement it with Amazon Web Services and OpenResty. We demonstrate the feasibility, robustness, and elasticity of the proposed platform with a use case of perspective-n-point solution. In this use case, the average round-trip delay is 153 ms, which meets the near real-time requirement of mobile robots.

Keywords: Cloud robotics · Mobile robots · Visual SLAM · Perspective-n-point

1 Introduction

1.1 Motivation

Visual SLAM is one of the intensively computational tasks for mobile robots [1,2]. It requires powerful processors to estimate the pose from each incoming frame in real time [3]. Cloud computing is a promising method to make up for the deficiency of local computational ability. Traditional robotics, which combined with cloud computing, evolves into cloud robotics in recent years. The cloud allows robots to offload intensively computational tasks and large-scale storage, so that both cost and power consumption of robots get reduced [4]. Besides,

This work was sponsored by the Research Grant Council of Hong Kong SAR Government, China, under project No. 16212815, 21202816 and National Natural Science Foundation of China No. 6140021318 and 61640305; Shenzhen Science, Technology and Innovation Comission (SZSTI) JCYJ20160428154842603 and JCYJ20160401100022706; partially supported by the HKUST Project IGN16EG12. All rewarded to Prof. Ming Liu.

M. Liu et al. (Eds.): ICVS 2017, LNCS 10528, pp. 3–15, 2017.
DOI: 10.1007/978-3-319-68345-4_1

robots have an access to a shared repository on the cloud [5–7]. As a result, data sharing and skills reusing can be achieved. A robot is eventually not only a machine but also a bridge between its user and heterogeneous services.

Although the cloud benefits robots, some drawbacks and challenges still exist and must be further addressed. Existing cloud robotics platforms enable data transmission and executing computation parallelly [4,8] and optimizedly [9,10]. However, few of them pay attention to the infrastructure of the cloud. Similar to other cloud platforms, the platform for distributed visual SLAM is open to the public and required to run stably in a long time. Under a long-term operation, some modules of the cloud tend to crash occasionally. In this paper, we pay close attention to the robustness, security, and elasticity of the cloud. We propose a novel cloud robotics platform named RSE-PF for distributed visual SLAM.

Besides, most of the previous work constructed their cloud platforms with ROS[1]. ROS provides researchers a middleware to design their robot systems in a modular way. The loosely coupled modules can be re-used, which helps researchers implement their robot systems conveniently. However, there are still some shortcomings with ROS, like its huge overhead and platform dependencies. Therefore, different with [5], RSE-PF does not treat ROS as a necessary part.

1.2 Contribution

We stress the following contributions in this paper.

- We clarify the tasks of each cloud service type for distributed visual SLAM and propose a novel cloud robotics platform named RSE-PF.
- We demonstrate the feasibility, robustness, and elasticity of RSE-PF with implementing the perspective-n-point solution for autonomous robots creating point cloud maps in a distributed way.

The features of RSE-PF are as follows.

Robust: The broken servers are detected and replaced automatically.
Secure: Only authorized users are allowed to send requests to the cloud.
Elastic: The cloud scales outward when it is under huge pressure. It scales inward when it keeps idle for a while.
Multiple programming languages supported: Visual SLAM services could be implemented with C/CPP and Lua scripts. Researchers are allowed to implement their application with/without ROS.

1.3 Organization

In Sect. 2, a brief introduction to cloud computing and previous work is described. In Sect. 3, we clarify the tasks of three cloud service types for distributed visual SLAM. The RSE-PF model is presented in Sect. 4 along with the implementation in Sect. 5. In Sect. 6, we demonstrate the feasibility and features of RSE-PF via implementing the perspective-n-point solution based on proposed cloud platform. Finally, we conclude with a discussion of our future work.

[1] http://www.ros.org/.

2 Related Work

2.1 Cloud Computing

NIST defined cloud computing and three service types of it [11], i.e. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). In brief, SaaS applications are exposed to users. The users have an access to the functional applications via various devices (e.g. mobile phones, laptops, tablets) and do not manage or control hardware resources or runtime environments directly. The users of PaaS are provided a platform which includes programming languages, dependencies, and management tools. They are allowed to deploy their applications on the platform without consideration on the infrastructure of the cloud. As for IaaS, the users are able to manage or control the cloud infrastructure in the way provided by IaaS providers. IaaS, PaaS, and SaaS offer increasing abstraction. The relationship of these three cloud service types is considered as Fig. 1.

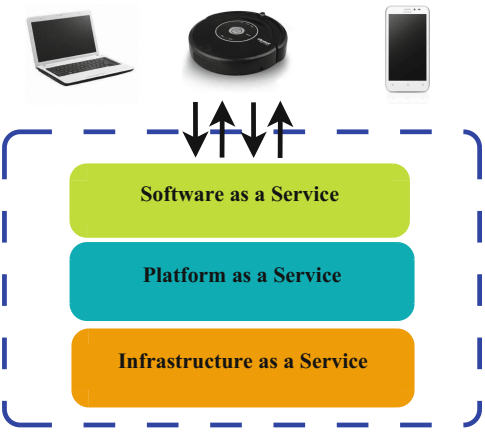

Fig. 1. The relationship of IaaS, PaaS, and SaaS: IaaS provides basic computing, storage as well as network components to the platform. PaaS provides necessary runtime environment to applications. SaaS is the functional applications which are exposed to the users.

2.2 Cloud-Based Visual SLAM

As for visual SLAM, robots are required to compute the camera pose from each incoming frame. Limited by on-board computational ability, robots are hard to locate themselves and map the scene in real time. Enti et al. [4] proposed DAvinCi framework which offloaded intensively computational workload from onboard resources to a backend cluster system. They implemented such a software framework around Hadoop cluster with ROS message. They proved the feasibility of their system and performance gains in execution times through

implementing the FastSLAM based on DAvinCi. RoboEarth [6] aims to develop a giant repository and achieve a World Wide Web for robots so that robots are able to share data and reuse skills. As a part of RoboEarth project, Repyuta [5] provided an access to RoboEarth knowledge repository which enables robots to share data and skills. Based on Rapyuta, [7] implemented a dense visual odometry algorithm on a smartphone-class ARM multi-core CPU.

A common feature of previous work is that they built a bridge between clients and servers with ROS message or WebSocket. Clients send data to the cloud, and servers finally send results back. In this way, offloading can be achieved. However, the infrastructure of the cloud does not get a full use. In this paper, we propose a cloud robotics platform for distributed visual SLAM with consideration on robustness, security, and elasticity. Besides, in order to show the feasibility and features of RSE-PF, we demonstrate the use case that multiple robots create point cloud maps simultaneously and distributedly with the PnP solution service based on RSE-PF.

3 Cloud Services for Distributed Visual SLAM

In general, cloud services can be grouped into three categories: IaaS, PaaS, and SaaS [11]. Similar to general cloud services, cloud services for distributed visual SLAM includes these three types as follows.

3.1 IaaS

IaaS provides basic computation, storage, and network resources. Since the cloud is open to the public, the security of the cloud should get attention. Besides, the cloud may be under high concurrent connection at some point, and it may also keep idle for a while. Therefore, it should be adjusted elastically according to some metrics like CPU utilization or robot counts. In addition, the infrastructure must be fault-tolerant. Since there are hundreds of servers in the cloud, some crashes or errors may happen occasionally. There should be a robustness mechanism to handle such situations, like replacing the crashed server with a new one.

3.2 PaaS

PaaS provides prerequisites to SaaS applications, like dependencies, programming language supports, etc. It also provides each robot client with secure and isolated runtime environments. Besides, the communication protocol between robots and the cloud should be well-designed to save bandwidth resources and reduce network delay. In addition, the platform should be with low overhead so that the cloud are able to handle high concurrent connections.

3.3 SaaS

SaaS provides visual SLAM services in a stateless way. The stateless way means the received requests contain all necessary information for SaaS applications, and the cloud does not need to retain information for computation. In Sect. 6, the PnP solution is one of the stateless applications.

As mentioned in IaaS, one robustness mechanism is to replace the crashed server with a new one. With this mechanism, the new server can hardly provide services to the past users if the SaaS application is stateful. It is because the state information has gone with the crashed server. Take ORB-SLAM [3] for an example. ORB-SLAM is classified as a stateful algorithm in this paper, for the keyframes of ORB-SLAM could be considered as the state of this algorithm. When the server running ORB-SLAM crashes, another one can hardly replace it and continue to provide services because the keyframes have gone with that broken server.

Besides, it is important to consider the influence of network delay in some visual SLAM applications which are sensitive to network delay. At $t + \Delta t_1$, the cloud receives the message which was sent at t. The result of this message will be received by the robot at $t + \Delta t_1 + \Delta t_2$. Δt_1 and Δt_2 are the network delay of uploading and downloading respectively.

4 RSE-PF Model

Based on three cloud service types, we propose RSE-PF for distributed visual SLAM. As depicted in Fig. 2, it consists of a firewall, a load balancer, a monitor, a scale modifier, multiple backend servers, and multiple data servers. The main functions of each part are as follows.

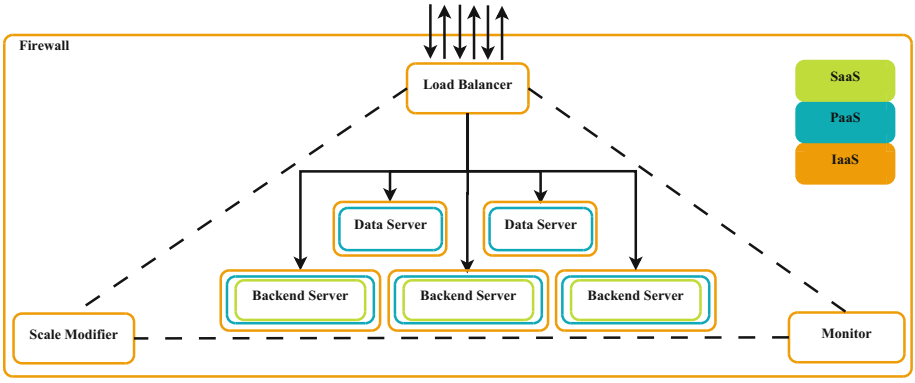

Fig. 2. The architecture of RSE-PF model (Orange: infrastructure, blue: platform, green: software) (Color figure online)

firewall: Since the cloud provides services to the public, a firewall is a necessary part of the cloud. Developers design the security rules, open ports, and access control lists to protect the cloud from unauthorized visits.
load balancer: It distributes requests to backend and data servers. It also executes health detection towards backend and data servers periodically.
monitor: It monitors the states of the cloud, including the result of the health detection, CPU and bandwidth usage of backend server, etc. Once some events get detected, the monitor will inform the scale modifier.
scale modifier: It modifies the scale of the cloud after getting informed by the monitor. It launches servers when the cloud is under big pressure. It also kills servers when the cloud keeps idle for a while. Besides, it replaces the broken server with a new one according to the result of health detection.
backend server: Developers design the transmission method between robots and the cloud. They also implement their visual SLAM applications like PnP, place recognition in a stateless way.
data server: It stores data which needs long-term storage or to be shared among multiple robots. Images, key frames, and maps can be stored in visual SLAM applications.

In Fig. 2, IaaS, PaaS, and SaaS are labeled accordingly. The mechanism of IaaS originates from Amazon Web Services[2], which is as follows. After requests reach the load balancer through a firewall, the load balancer distributes the requests to backend and data servers. Backend servers make computation and exchange data with data servers. In addition, when the monitor detects events like unhealthy servers or some metrics exceeding warning lines, it will inform the scale modifier. Subsequently, the scale modifier will adjust the cloud scale.

5 RSE-PF Implementation

In this part, we will detail the implementation of RSE-PF model from IaaS, PaaS, and SaaS.

5.1 IaaS

As proposed above, RSE-PF consists of a load balancer, a scale modifier, a monitor, data servers and backend servers. Multiple companies provide their IaaS productions, such as Amazon Web Services(AWS) and Google Compute Engine[3]. AWS provides some cloud services infrastructures, covering computing, storing and network. We implement the IaaS of RSE-PF based on AWS as follows.

Computing: We implement backend servers with Amazon EC2 instances. Amazon EC2 allows users to exploit cloud resources on demand. Via Amazon Machine Images or executing pre-prepared scripts, EC2 instances are armed with qualified environments immediately after they get created.

[2] https://aws.amazon.com/.
[3] https://cloud.google.com/compute/.

Storage: AWS provides multiple types of storage, including S3, EBS, and Glacier, etc. S3 allows long-time storing, just like network disks. A little difference to network disks is that S3 naturally serves machines instead of human beings. API is provided to help developers manage and develop the cloud automatically. EBS provides high IOPS, just like hard disk, while Glacier provides cheap and permanent storing service, which suits for storing log files and raw data.
Network: Amazon Virtual Private Cloud can be used to construct the cloud architecture and improve the security of the cloud. Besides, main parts can be doubled with VPC to add redundancy and achieve high availability.
Security: Besides VPC, each EC2 instance is protected by a firewall. Users can limit unauthorized visits with secure strategies of this firewall.
Management: Elastic Load Balancer of AWS can be used as a load balancer. It performs request distribution and health detection. In particular, HTTP, HTTPS, and WebSocket protocols are all supported in AWS Elastic Load Balancer. CloudWatch of AWS helps monitor the whole cloud. It alarms when some metrics of the cloud exceeding their thresholds. In addition, AWS Auto Scaling helps modify the scale of cloud system elastically after being alarmed by CloudWatch.

5.2 PaaS

The PaaS of RSE-PF is implemented based on *OpenResty*^{TM}[4]. It is a powerful web platform which integrates Nginx core, Lua libraries and LuaJIT. Researchers are allowed to implement web services on it with Lua scripts, Nginx C modules, and C/CPP programming languages. Compared to huge overhead of ROS, *OpenResty*^{TM} is capable of handling 10 K to 1000 K connections in a single server.

The flowchart RSE-PF PaaS is shown in Fig. 3. Subsequently, we will define the communication protocol and explain how to build an isolated computing environment for each robot.

Communication Protocol: In general, messages of visual SLAM transmitted between robots and the cloud could be classified as three types: command, data, and result messages. In most applications of visual SLAM, both command and result messages are small-scale, like place recognition and pose estimation. Therefore, we encode command and result messages into JavaScript Object Notation (JSON). JSON is a lightweight data exchange format which is easy for machines to parse and generate. The JSON format string can be easily decoded in the cloud via *OpenResty*^{TM} cjson Lua model. Typical messages are shown in Fig. 3.

To maintain a bi-directional full duplex connection between robots and the cloud, WebSocket is used to transmit messages, especially command and result messages. WebSocket is built on the top of HTTP. It creates open communication channels between clients and servers. As a consequence, the cloud is able to publish messages actively without robots asking results periodically.

[4] https://OpenResty.org/.

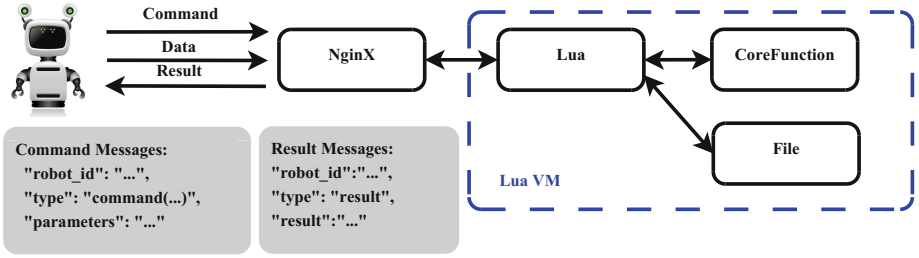

Fig. 3. The flowchart of RSE-PF PaaS (Robots send command and data messages to the cloud, while the cloud publishes results back to robots. In the cloud, when the connection between the robot and the cloud is established, a Lua VM is created. Following computation work will be completely processed in this Lua VM.)

Besides, compared with HTTP, WebSocket-based transmission helps save bandwidth resources hundreds of times under high concurrent connection. It also helps reduce 70% network delay compared with HTTP long polling [12].

Payload length of messages is one of the key metrics because large scale messages consume plenty of bandwidth resources. Some results demonstrated that the delay and packet loss will increase if payload length of messages increases [5]. Therefore, we compress JSON format string with Zip before transmission.

In applications of visual SLAM, robots might need to offload some data like point clouds, maps, and RGB images to the cloud for storage or computation. These data might be larger than 100 Kbytes. Converting images or point clouds to JSON format string would result in an even larger message size. When the payload length goes up, the performance goes down. As a result, we adopt HTTP to post files to data servers instead of encoding them to JSON string so that larger message size caused by converting is avoided.

Isolated Computing Environment: When the connection between the robot and the cloud is established, a Lua VM is created. Further computational tasks will totally be processed in this Lua VM. After loading necessary Lua models or C dynamic link libraries, the computational environment specific for this robot

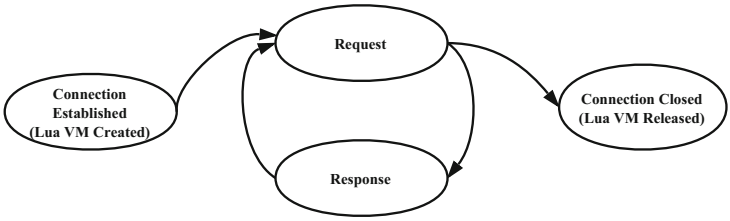

Fig. 4. The state diagram of the cloud (When the connection is established, the Lua VM is created. The robotic services will be provided in a request-response way. Once the connection is closed, the Lua VM will be released)

is prepared. Besides, the memory of each Lua VM is also isolated from others. After the Lua VM having been created, the robotic services start. The robot sends command messages to the cloud, and the cloud will publish result messages actively towards the robot. If the robot does not need services any more, this connection will be closed, and the Lua VM will be released simultaneously. The life length of Lua VM is as the same to the connection, which is shown in Fig. 4.

5.3 SaaS

In RSE-PF, we develop visual SLAM services with Lua and C/CPP. Lua is a lightweight script language, which could help implement some basic processing, like encoding and decoding of JSON format string, recording logs, etc. C/CPP could help implement complicated logics in visual SLAM algorithms. In RSE-PF, developers are allowed to compile their C/CPP codes of robotic services into dynamic link libraries, which will be called by LuaJIT FFI.

As shown in Fig. 3, The message received by Nginx are processed by Lua basically, like file operations, encoding and decoding. It implements complicated logics of visual SLAM by calling core functions which is dynamic link libraries coded by C/CPP. Finally, the returned value from core functions will be published actively to the robot. Furthermore, in order to detail the method to implement SaaS of RSE-PF, we take perspective-n-point as an example in next section.

6 RSE-PF Use Case

In this section, we implement perspective-n-point (PnP) solution [13] based on RSE-PF to demonstrate its feasibility and features. With PnP robotic service, robots create point cloud maps from the NYU Depth V2 dataset [14].

The perspective-n-point is the problem of estimating the position and orientation of the camera given n correspondent points. It originates from camera calibration and is widely used in many fields, especially location and navigation of autonomous robots.

6.1 Implementation

Since both matched key points and transformation matrix are in small scale in PnP solution service, all necessary information are sent by command or result messages via WebSocket. Besides, we compress JSON-format string with libz to reduce bandwidth costs.

The division of computation between robots and the cloud is shown in Fig. 5. In the cloud, Lua modules process requests basically, including encoding, decoding, compressing and decompressing. We implement PnP solution based on the solvePnPRansac function of OpenCV. It makes the final solution more robust to outliers with RANSAC. We encapsulate the PnP solution into a core function. The process is shown in Fig. 6. The robot computes descriptors and matches

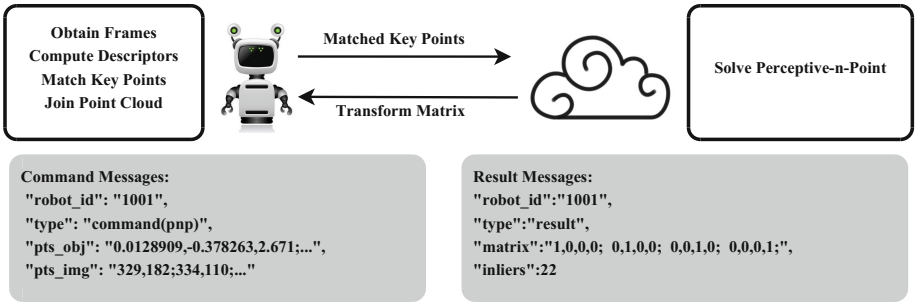

Fig. 5. Computation division between robots and the cloud (PnP)

Lua Scripts

```
server_initialization()
while true
{
   data =  receive()
   data = uncompress(data)
   if the data contains close command
      close the connection
   else
   {
      result = ffi.pnp_solution(data)
      result = compress(result)
      send(result)
   }
}
```

C/CPP core function

```
char* pnp_solution(char* data)
{
   parameters = decode_json(data)
   result = solvePnPRansac(parameters)
   result = encode_json(result)
   return result
}
```

Fig. 6. The pseudocode of PnP robotic service.

keypoints locally. After received transform matrix, it will join current frame into the point cloud. Finally, the point cloud of the whole scene will be created.

As proposed in Sect. 5, we implement the infrastructure of RSE-PF with AWS. Data servers are not used in this case because necessary information is in small scale (less than 10 KB) and can be carried in command messages. We choose AWS in Seoul because the delay from our lab in Shenzhen to Seoul EC2 server is the lowest among all AWS regions. AWS ELB, Cloud Watch, and Auto Scaling are used to construct the IaaS of RSE-PF. A hundred Docker containers act as robots and request PnP services. Docker containers run on a Core i7 processor with 16 GB RAM (Fig. 7).

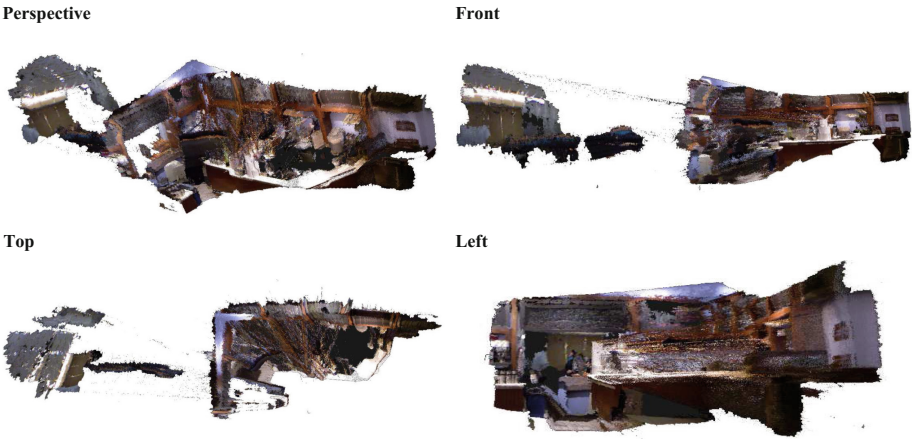

Fig. 7. The point cloud map created by one of Docker containers

6.2 Result

We record the real-time performance of all these robots. Robots perform near 5 fps in this use case, and the average time from sending a command message to receiving a result message is 156.36 ms. Specifically, the average time for server

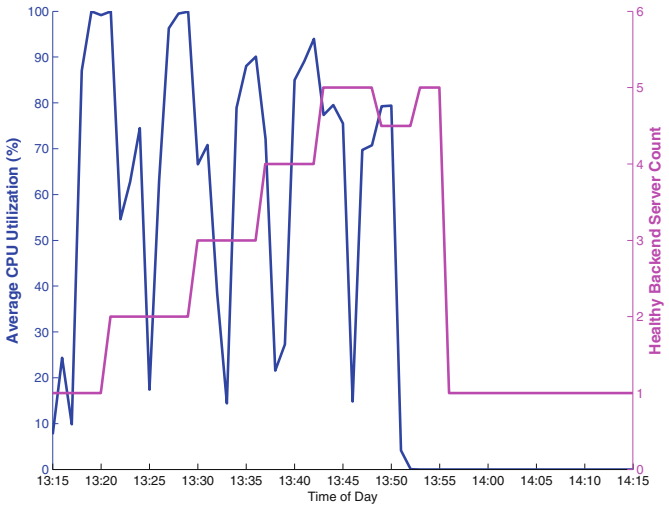

Fig. 8. The average CPU utilization and the healthy server count of the cloud when the cloud faces an increasing number of requests. (Every 20 s, a docker container is launched, so that a hundred docker containers sent requests to the cloud at the same time. Finally, we kill all docker containers after 13:52.)

computing is 29.82 ms. In other words, the transmission delay is near 130 ms which is in the normal range for cloud computing.

From the results, the transmission delay is a bottleneck for real-time distributed visual SLAM services. If better real-time performances are expected, cloud resources close to robots are preferred. Besides, developers are recommended to assign the computation tasks carefully. The tasks which are not sensitive to delay are recommended to be assigned to the cloud, like map merging and global optimization. In addition, the saved time should be more than transmission delay so that real-time performance gains from cloud computing.

Besides, Fig. 8 demonstrates the elasticity and robustness of the cloud. The cloud scales up when the average CPU utilization exceeds the threshold (70% in this case). It scales down when keeping idle for a while (5 min in this case). At 13:50, one backend server crashed. Another backend server took its place and continued to serve robots soon after that.

7 Conclusion and Future Work

In this paper, we firstly defined the tasks of three cloud service types for distributed visual SLAM. IaaS provides basic computation, storage, and network infrastructures. It is also equipped with mechanisms of security, elasticity, and robustness. PaaS defines the transmission protocols between robots and the cloud. It also provides isolated computational environments to each robot. SaaS provides visual SLAM applications for robots in a stateless way.

Besides, we proposed RSE-PF based on three cloud service types and described the implementation method of RSE-PF based on AWS and $OpenResty^{TM}$. With the firewall around EC2 instances, the cloud is under secure protection. With the load balancer, monitor, and scale modifier, the cloud runs robustly and elastically. The Lua VM created in Nginx provides each robot with an isolated runtime environment. In addition, the visual SLAM applications should be stateless in order to help the cloud replace the broken server without shooting troubles.

Moreover, we demonstrated perspective-n-point as a use case to prove the feasibility, elasticity, and robustness of our proposed framework. In this use case, the average round trip delay is 153 ms, which meets the near real-time requirement of autonomous robots. According to the round-trip delay, we found one bottleneck of the cloud-based algorithms was the transmission delay and proposed two recommendations for developers to improve the real-time performance of their algorithms.

From this work, we find the stateless requirement of SaaS for cloud robotics is a little harsh. For visual SLAM, functions like solutions of perspective-n-point, iterative closest point, and place recognition could be processed in a stateless way, but some algorithms like ORB-SLAM are complicated to implement statelessly. In the future, we plan to implement the PaaS part of RSE-PF with Docker to handle such complex robotic services. Besides vision, robots are capable of locating themselves based on WiFi signal strength [15]. It can be implemented as a SaaS application and helps robots locate themselves in indoor scenes.

References

1. Liu, M.: Robotic online path planning on point cloud. IEEE Trans. Cybern. **46**(5), 1217–1228 (2016)
2. Gianni, M., Papadakis, P., Pirri, F., Liu, M., Pomerleau, F., Colas, F., Zimmermann, K., Svoboda, T., Petricek, T., Kruijff, G.J.M.: A unified framework for planning and execution-monitoring of mobile robots. In: AAAI Conference on Automated Action Planning for Autonomous Mobile Robots, pp. 39–44 (2011)
3. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
4. Arumugam, R., Enti, V.R., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F.F., Kumar, A.S., Meng, K.D., Kit, G.W.: DAvinCi: a cloud computing framework for service robots. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 3084–3089. IEEE (2010)
5. Hunziker, D., Gajamohan, M., Waibel, M., D'Andrea, R.: Rapyuta: the roboearth cloud engine. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 438–444. IEEE (2013)
6. Waibel, M., Beetz, M., Civera, J., d'Andrea, R., Elfring, J., Galvez-Lopez, D., Häussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., et al.: Roboearth. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011)
7. Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., Waibel, M.: Cloud-based collaborative 3D mapping in real-time with low-cost robots. IEEE Trans. Autom. Sci. Eng. **12**(2), 423–431 (2015)
8. Beksi, W.J., Spruth, J., Papanikolopoulos, N.: CORE: a cloud-based object recognition engine for robotics. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4512–4517. IEEE (2015)
9. Wang, L., Liu, M., Meng, M.Q.-H.: A hierarchical auction-based mechanism for real-time resource allocation in cloud robotic systems. IEEE Trans. Cybern. **47**(2), 473–484 (2017)
10. Rahman, A., Jin, J., Cricenti, A., Rahman, A., Yuan, D.: A cloud robotics framework of optimal task offloading for smart city applications. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2016)
11. Mell, P., Grance, T., et al.: The NIST Definition of Cloud Computing (2011)
12. Lubbers, P., Albers, B., Salim, F., Pye, T.: Pro HTML5 Programming. Springer, Heidelberg (2011). doi:10.1007/978-1-4302-3865-2
13. Gao, X.-S., Hou, X.-R., Tang, J., Cheng, H.-F.: Complete solution classification for the perspective-three-point problem. IEEE Trans. Pattern Anal. Mach. Intell. **25**(8), 930–943 (2003)
14. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33715-4_54
15. Sun, Y., Liu, M., Meng, Q.H.: WiFi signal strength-based robot indoor localization. In: IEEE International Conference on Information and Automation, pp. 250–256 (2014)