Cloud Robotics: A Software Architecture

For Heterogeneous Large-Scale Autonomous Robots

Seyed Ali Miratabzadeh, Nicolas Gallardo, Nicholas Gamez, Karthikpai Haradi, Abhijith R Puthussery, Paul Rad, Mo Jamshidi
Electrical and Computer Engineering Department, University of Texas at San Antonio
Open Cloud Institute, University of Texas at San Antonio
San Antonio, Texas, USA
Ali.Miraftab@utsa.edu, {hbq744, jyi358, dxq821, qaw164}@my.utsa.edu, Paul Rad@utsa.edu, moj@wacong.org

***Abstract*—The paper proposes a software architecture for cloud robotics which intends three subsystems in the cloud environment: Middleware Subsystem, Background Tasks Subsystem, and Control Subsystem. The architecture invokes cloud technologies such as cloud computing, cloud storage, and other networking platforms arranged on the assistances of congregated infrastructure and shared services for robotics, for instance Robot Operating System (ROS). Since the architecture is looking for reliable, scalable, and distributed system for the heterogeneous large-scale autonomous robots, Infrastructure as a Service (IaaS) is chosen among the cloud services. Three major tasks can be handled by the proposed software architecture Computing, Storage, and Networking. Hadoop–MapReduce provides the appropriate framework in the cloud environment to process and handle these tasks.**

Keywords—cloud robotics; heterogeneous system; large-scale autonomous; Hadoop-MapReduce, Robot Operating System (ROS).

I. INTRODUCTION

Robotic services are systems, devices, and robots with three functions: sensation, actuation, and control [1]. Providing robotic services to support intelligent artificial activities through socially conscious interactive behaviors is an emerging topic in robotics research for instance to support daily human activities [2] and IBM Smarter Cities [3]. To afford this important, robotic technologies can be integrated with advanced networking technologies [18] to foster the emergence of networked robotics. A networked robotic system indicates to a group of robotic devices that are connected via a wired and/or wireless communication network [4]. Networked robotics, especially the multi-robot system, distributes the workload of sensing, actuating, communication, control, and computation among a group of participating robots. It has achieved great success in industrial applications, intelligent transportation systems, and security applications. However, the advancement of networked robotics is restricted by resource, information, and communication constraints inherent in the existing framework [5]. These drawbacks are significant in the networks of mobile robotic, and may lead to severe performance degradation.

Networked robotics serves as a stepping stone [5] towards cloud robotics. For instance, cloud-enabled network robotics leverages emerging cloud computing technologies to enhance networked robotics removing analytical duties off the robot.

The design objective is to overcome the limitations of networked robotics using elastic resources provisioned by an ever-present cloud infrastructure. Cloud computing uses a sophisticated networked ecosystem but presents a clear, concise interface to extend the capabilities of networked robotics.

In 2010, Kuffiner J. J. penned the term "Cloud Robotics" and described a number of potential advantages over conventional robotics [6]. Cloud robotics is an emerging field of robotics embedded in cloud computing, cloud storage and cloud networking. While providing advantages of powerful computational, storage, and communications resources of modern data centers, it also allows robots to benefit from the platform which includes infrastructure and shared services. Robots connected with the cloud platform can access the services running on the remote servers. Furthermore, one byproduct of the cloud robotics is the allocation of the background tasks, services not related to core function of the robot, to the cloud platfrom, for instance Big Data management and implementation of advanced Machine Learning algorithms [7], [20] and heavy image enhancement algorithms [17], [21], [24] or software [19]. The definition of cloud computing by NIST [8] enables remarkable elasticity in designing and implementing new applications for networked robotics.

Several researchers have begun to study the architecture of cloud technologies in robotic applications. For instance, a research group at Singapore's ASORO laboratory announced a software framework that merges the scalability and parallelism advantages of cloud computing to large-scale environments of robots. The system is implemented through Hadoop clusters and ROS communication networks [9]. The dustbot project [10] is an example of such a sophisticated network within a robotic system. In this project, two kind of robots perform the cooperative tasks while benefitting from the use of external sensory systems. Two tasks have been implemented within the system: garbage collection and street cleaning and sweeping. Z. Du et. al. [11] introduced the concept of Robot as a Service and the initial system of a Robot Cloud Center. To the best of our knowledge, substantial works have few distinct design directions and implementation, especially in the area of cloud robotics utilizing emerging advanced technologies in the cloud research area such as OpenStack [12] and Docker [13].

This work was supported by Grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD, and by Open Cloud Institute at University of Texas at San Antonio, Texas, US.

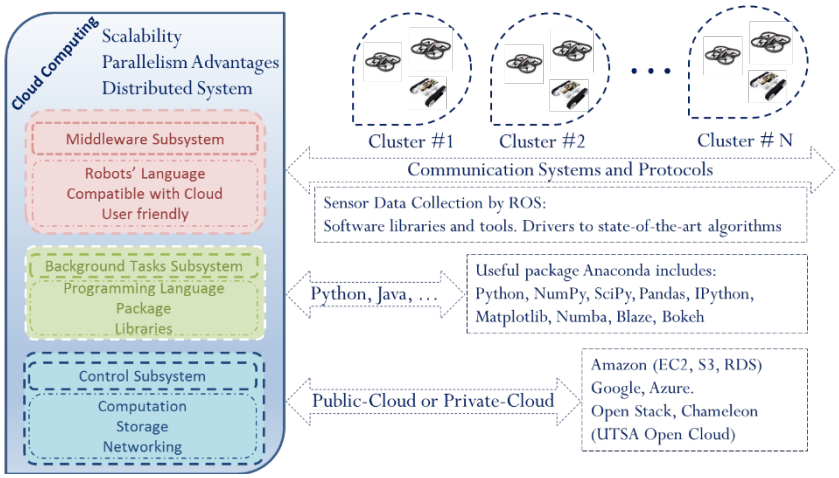

Fig. 1. *Architecture of the proposed Cloud computing Platform.*

The major aspect of our research expounds on the creation of a software architecture which will enable large-scale systems of autonomous agents in a heterogeneous environment with robots publishing sensor data throughout the cloud and also capturing data for on demand processing and post processing for the computationally intense algorithms. The information flows from robots to instances within the cloud servers and back to the robots after analytical work is performed on the cloud. This architecture includes the advantage of ROS handling the modular communication mechanism among the nodes. More specifically, ROS provides the distributed communication for not only the multi-types of robots but also for the cloud servers. Elasticity properties of the cloud has the potential to provide a scalable system. IaaS provides the governor of the software and hardware which are needed for this scalable system. OpenStack accomplishes the management of the provisioning for the resources in the cloud environment. In the proposed architecture Hadoop is used as the powerful cloud based data management tools which serves to store the sensor data.

The rest of the article is organized as follows. First, we outline various applications of the concepts required for cloud robotics and how cloud computing can overcome critical challenges in large-scale robotics. We will also categorize the robots' tasks in order to get benefit from the cloud. Next, we describe the proposed cloud robotics architecture, and elaborate on three key enabling subsystems. Following that, we present the implementation of the system, including two types of robots, the Kobuki turtlebot 2 as a ground robot and the Parrot Bebop as an aerial robot. We address technical challenges in designing and operating the cloud robotics architecture. In chapter five the advantages of the proposed software architecture for cooperative robotics are described in details focusing on the OpenStack. Finally, we conclude and summarize this article.

II. CLOUD FUNCTIONALITY AND THE ROBOTS' TASKS

Networked robotics, as same as standalone robots, express intrinsic native constraints. For instance, robots have smaller computer architecture for mobile purposes leading to inadequate computing capability, since all computations must be conducted onboard the robots. Data access is also constrained to the discrete storage of the network. Collaboration from brisk progression of wireless communications and recent cloud computing technologies can be utilized to overcome some of these restrictions through the concept of cloud robotics. This creates a computationally more intelligent, well-organized and less expensive robotic network.

To carry this out, important robot's tasks are categorized based on the cloud resources types. Some of robots' tasks can be summarized as obstacle avoidance, vision processing, localization, path planning and environment mapping. In a practical point of view, handling all tasks by multi agents or swarms of robots imposes significant down sides such: dedicated power supplies, good shock protection for HDD, cost prohibitions, duplication and redundancy efforts, and solving networking problems. Cloud environments overcome these unnecessary and inefficient concerns by shifting the paradigm of computing resources off the physical robots. In this cloud architecture we propose an infrastructure which includes computing, storage and network in the datacenter. To break down the objects, we categorized these tasks as follow:

- The computing resources (VM or vDC) allocated for each agent
- Storage resources for Data Analytic research activities (vSLAM, world-maps, etc.)
- Network resources for swarm performing cooperative missions (shared information)

With this order, each node of the system is treated with respect to all three aspects: computation, storage, and networking. In the next session the proposed architecture is described to handle all the aforementioned kinds of tasks simultaneously.

III. PROPSED SOFTWARE ARCHITECTURE

A cloud based architecture for large-scale autonomous robots has been proposed in Fig. 1. The architecture consists of three subsystems: (1) Middleware Subsystem which can be counted as the main carrier for the platform. (2) Background

Tasks Subsystem, the framework for batch processing including software packages. (3) Control Subsystem, which is the brain of the platform. However, the subsystems are ordered horizontally, and the computation, storage, and networking tasks are processed vertically. Serialization of the three latter functions is processed in the following order: Networking, Storage, and Computation. The order of the storage and computations tasks could be arguable, however we suggest the storage as the prior.

Since provisioning the control resources is more valuable for intense computational tasks. Diving into the details of the subsystems follow with the next three sections:

A. Middleware Subsystem

The first subsystem provides the compatibility between robots and protocols of the communication networks. This subsystem is implemented at the lowest level of the system, connecting the robots and the cloud platform with bidirectional communication. In addition, the middleware subsystem allows all nodes of the platform with the ability to access all types of sensor data. To do so, ROS plays a large role in our systems first layer. Fig. 2. shows the systems data communication network. Our proposed system requires a reliable communication network that allows many agents to pass information back and forth to each other quickly and easily. ROS uses a data passing model based on the publisher-subscriber relationship. This relationship makes sending and receiving data intuitive for the software developers writing programs in ROS. For example, if program A requires information from program B, program B would publish this data to the ROS master node, while program A would subscribe to the data and retrieve it when available from the ROS master node. This is a direct reason for ROS's modularity and immunity to dynamic network infrastructure where communication reliability is poor or intermittent. ROS is able to handle these situations because it is made up of many functional nodes, or set of processes that are publishing and subscribing to the ROS master node. Each node is responsible only for a certain task, which correlates to the developer's choice of responsibilities for each node. Consequently, if each node's functionality is a small enough portion of the whole, the loss of one or two nodes only effects a small portion of the systems functional processes and will not degrade the entire system to failure increasing robustness. However, ROS is a common language among multi robots, effectively use of critical resources among threads [22], [23] which can control the transent behaviors of the systems.

- ROS Multimaster FKIE: Furthering communication reliability is a ROS package named multimaster fkie. This package allows more that one ROS Master Node to be ran within the same ROS network. It works by syncing each ROS Master to a heartbeat. The heartbeat allows the system to know when a ROS Master has dropped from the network, and waits for the heartbeat to show up again. This idea is similar to separating the system out into nodes; the more you break your system up, the more resilient it becomes. Using ROS multimaster fkie gives software developers an easy way to add robustness to the system.

B. Background Tasks Subsystem

This subsystem provides the framework for doing the batch processing from a software point of view including programming language, packages and libraries. The goal of this subsystem is to feed the control subsystem with required data. Handling the information is carried on in this stage. Hence, powerful programming enriches the system with less bugs and faster processing time. The useful programming languages in this area are Python, C++ and Java. And among them Python is more common now a days because of its power and free libraries and package such as Anaconda. Anaconda includes Python, SciPy, NumPy, Pandas, IPython, Matplotlib, Numba, Blaze and Boken all useful for computation and mathematical programming. Furthermore, Hadoop Map/Reduce framework for doing the batch processing of sensor data is efficient for data storage and processing in this step. We describe the Hadoop–MapReduce later in this paper.

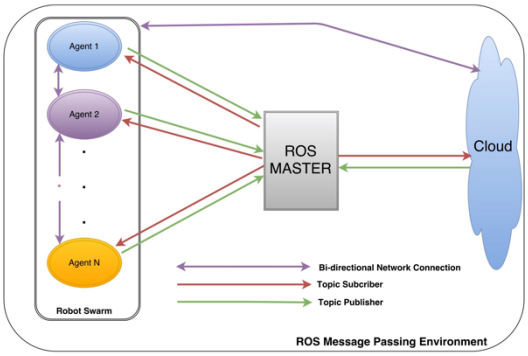

Fig. 2. Illustration of the ROS messaging in the platform

C. Control Subsystem

The final subsystem is for controlling the three aforementioned tasks: Computation, Storage and Networking. IaaS is chosen for the cloud service since we want to utilize the advantages of scalable hardware and software for managing our hardware based on the required computational needs. Hadoop is installed on top of the platform to capture the data and podcast it for the appropriate agent. Hadoop Distributed File System (HDFS) provides us the powerful capability to handle the huge amount of information that has been acquired and store it in the proposed platform. HDFS also allows the use of Google's map-reduce algorithm to search and find the specific data we need. The architecture gets the advantage of the Hadoop framework which allows for the distributed processing of large data sets across clusters of agents using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of agents is able to be directly monitored in the third subsystem.

IV. IMPLEMENTATION

A. Connecting Multiple agents (Kobuki and Bebop)

ROS is an open source software developing platform for any robotic system. ROS includes many powerful packages, functionality wise, which are available for free and legal to use as the developer chooses. With that, the Kobuki ROS package

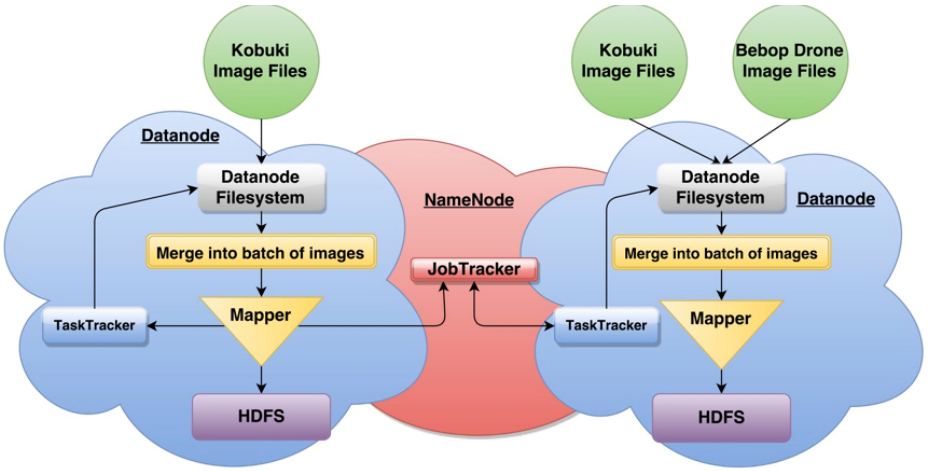

Fig. 3. Image storage based on the proposed architecture

and Bebop ROS driver were used in order to operate each agent in ROS. These packages output data topics pertaining to the agents angular velocities, linear velocities, camera data, control topics, and many other state topics. This data is then used to perform various co-operative tasks. The Kobuki agents along with the Bebop drones use wifi communication and a nework router to achieve bi-directional passing of data. Utilizing ROS multimaster fkie, each agent is made a ROS Master, allow for each robot to pass in and out of the wifi communication network without fatal communication or system errors.

B. Connecting each agent to the cloud using ROS

The final piece of the network layer for system is the communication between each agent and the cloud. Each Kobuki relies on an ODroid microprocessor that runs Samsung Exynos5422 Cortex™-A15 2Ghz and Cortex™-A7 Octa core CPUs with 2 Gigs of RAM. This microprocessor runs an Android version of Linux and is essentially a credit card sized Linux computer. Despite its impressive specifications, the ODroid falls short with computational heavy functionality such as Simultaneous Localization and Mapping (SLAM). To solve this issue a cluster of servers running the OpenStack cloud computing software is used in order to perform the necessary resource demanding functions such as image processing with SLAM. OpenStack cloud computing software adds dynamic resource scheduling functionality to the servers. This means processing resources such as hard disk space and RAM can be allocated in real time according to the demands on the server cluster. By doing this, resources become elastic and power is saved by sending unused servers into low power mode while they are not being used. OpenStack allows the user to create an instance, which is a virtual machine that has a certain allocation of RAM and hard disk space, and allocate a floating IP address to the instance. Each agent is wirelessly connected to the cloud through a wireless router, allowing bi directional communication. With all of the agents on the same network, ROS is able to pass agent data to the cloud by using the cloud instances floating IP.

C. Hadoop ROS interaction

The robotic agents each publish their own image topics consisting of the video feed. The images are collected and published as a topic in the ROS network. The Hadoop cluster and the robots, land and air, are connected through this ROS network allowing the transfer of our topics from robots to the cloud Hadoop cluster. A collection of images are aggregated to create one file to be sent into the HDFS. The library HIPI, Hadoop image processing interface, allows for images to be concatenated into one file for the Hadoop distributed filesystem, HDFS, and the created image bundles can be manipulated by OpenCV a popular image processing library. OpenCV is first used to convert the ROS images into a format which can be modified by a script. Next, the OpenCV software can perform modifications to the image such as drawing on the image given pixel location. This can be used to identify obstacles or other irregularities the robots may see. The master node of the Hadoop cluster, or the Namenode, has a process called the JobTracker which handles the jobs asked by the user. This job tracker assigns the map and/or reduce tasks to other nodes, mainly data nodes, for execution. Another process, the TaskTracker, is running on the corresponding data nodes of the cluster. The TaskTracker executes the map or reduce algorithm. Once finished, the TaskTracker sends the status back to the JobTracker. The cluster namenode then sends the job of mapping the input files into key:value pairs which is then output into the HDFS from the datanode.

D. Storing images

Since we want the majority of data processing off the robots, we add the HIPI [14] software to the cloud instances. Therefore, the datanodes are running their own small ROS program saving images onto a folder located within the datanode. Once a request is received to map the images from the JobTracker, an HIPI program will create the bundled images file and run the mapper as well. The results are added to the HDFS creating a set of images which can be manipulated and analyzed with openCV.

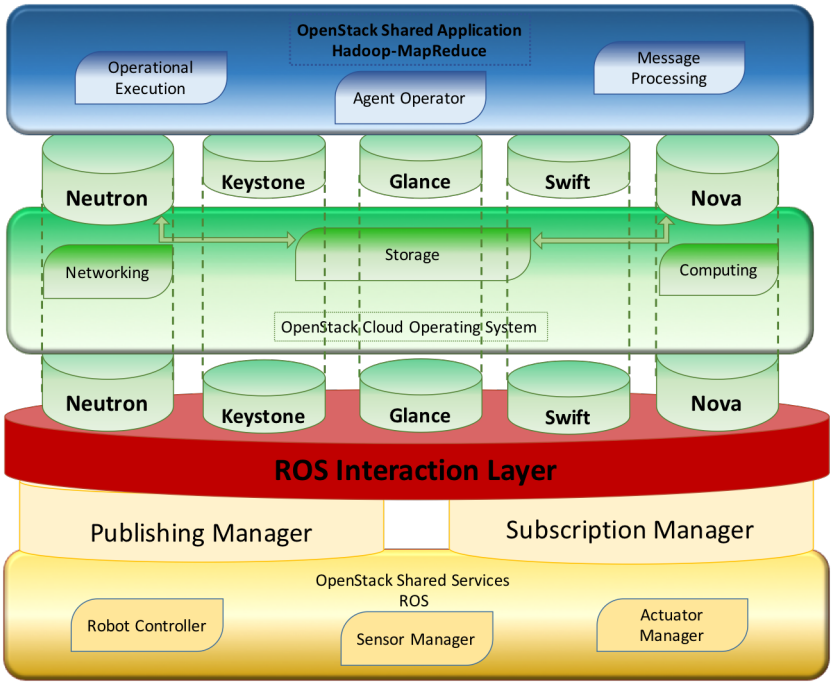

Fig. 4. Illustration of the Software Architecture for implementing the Cooperative Robots on OpenStack

V. Software Architecture and Cooperative Robotics

Two hot issues in cooperative robotics are reliable networked robotic systems and formal models and methods for cooperational tasks. Cooperative robotics requires data sharing, cooperative perception, and collective intelligence. While networked robotics suffers from the resource, information, learning, and communication constraints, the proposed architecture gets the advantages of OpenStack-Nova having a strong networked ecosystem for provisioning elastic computational and storage recourses. OpenStack-Neutron serves as a heavy-duty networking tool for handling the communication among the system, and finally to overcome the security challenges by controlling the virtual machines (VM) persistently.

Referring to the proposed software architecture in Fig. 1, making a coupled system comprises of the OpenStack and Hadoop platforms. OpenStack function as the shared data center which can be controlled through the dashboard as a user interface. In addition, the decision to offload a specific task requires a unified framework that can handle a list of complex issues. This important task is handled by Hadoop-MapReduce system. First, the offloading strategy should consider various factors, including the amount of data exchanged, and the delay deadline to complete the task. Second, the decision should also consider whether it is more advantageous to execute the task within the group of networked robots hardware, given the presence of cloud resources and computation requirements. Finally, given a pool of cloud resources spread across different data centers, it is a challeng to allocate virtual machines optimally to execute the offloaded task and to manage live VM migrations. The other services by OpenStack facilitate solving the problem. For instance, Glance holds the created disk images allowing for quick provisioning of new instances with all the requred software in the saved images. Keystone serves as an access point for the other OpenStack services with authentication and authorization between the endpoints of each service. Swift implements a RESTful API to send commands for storing and receiving data from the OpenStack through HTTP requests. The illustration of this software architecture for the cooperative robots is depicted in Fig. 4.

To overcome the communicational challenge, the choice of standalone or cloud execution depends on the sensitivity to delay of the task. The communication delay introduced in sending the computation request to the cloud has to be factored into each decision. Packet delivery failures and communication outages are inherent in any wireless communication systems. The additional increase in failure rate depends on the network topology. The use of the IaaS cloud as a super node in the network effectively controls this rate.

Trust and security issues are major considerations in cloud robotics. The VM environment must be secure from outside interference. A malicious VM can subtly sabotage an important task without the robot being aware of the damage. A robot needs to trust the VM is not compromised to launch task delegation on a public cloud, especially when the computation and network traffic have monetary costs. The computing environments in the cloud should be verified by a user or a trusted party. Confidential data may be stored in the public cloud storage, while logically private to cloned devices. Therefore, strong integrity and confidentiality protection are needed to secure application data.

All the aformentioned capabilities make a system, consisting of a multitude of networked robots and other devices, which, as a whole, is capable of interacting with the environment through the use of perception and action for the performance of tasks. Forthermore, two current technologies for cooperation of robots, localization and learning, have shown to ability to be implemented in the software architecture.

Localization for mobile robots is a basic requirement, when it comes to cooperative spatial perception [16]. The localization problem is meanwhile predominantly addressed together with mapping, which leads to the paradigm of Simultaneous Localization and Mapping (SLAM) [15]. Most current approaches to SLAM are based on probabilistic (Bayesian) approaches, typically employing Kalman filter methods, particle filters, or expectation maximization techniques. While SLAM is well established for 2D mapping by single land robots, multi-robot mapping is considered to be a very important but also still largely open problem. In a cooperative robotics context, "Petri nets" have been used for modeling multi-robot plans, and a multi-robot coordination algorithm for environment exploration.

Learning can be successfully applied in behavior-based systems to adapt task assignment in heterogeneous robot teams, dealing with individual robot capabilities that change over time. Unsupervised learning methods, such as evolutionary algorithms, are popular for multi-robot task optimization.

Having the capabity to implement both localization and learning technologies enriches the software architecture to implement formal models and methods for cooperative robots. These methods include, but are not limited to, logic-based knowledge representation and planning, decentralized decision-making, graph-based control methods, game theory, Bayesian networks, discrete and hybrid system models, or models of natural systems applicable to robotics. Therewith, the venue is provided for hosting the cooperative planning and execution, cooperative perception, and cooperative learning and evaluation.

VI. Conclusion

A software architecture in cloud environment consisting of three subsystems was proposed. It has been implemented on an IaaS platform using OpenStack to add scalability to the system. By using ROS Multimaster FKIE the distributed platform correctly serves the heterogeneous large-scale autonomous robots with serious problems when one robot loses communication. Introducing any type of robot just entails installing the related package onto the robot and network. Finally with the advantage of Hadoop, data storage is carried out in a series of packages, allowing for large scale processing of the massive amount of data received from the robotic agents. The proposed architecture is reliable, scalable, and powerful in the three functions discussed: Computing, Storage, and Networking. Hadoop–MapReduce provides the appropriate framework in cloud to process and handle these tasks. At the end, the advantages of the proposed software architecture are described for cooperative robotics with an authentic shared data center for communication and a trustworthy platform provided.

References

[1] K. Kamei, S. Nishio, N. Hagita, and M. Sato, "Cloud networked robotics," *Network, IEEE,* vol. 26, pp. 28-34, 2012

[2] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and autonomous systems,* vol. 42, pp. 143-166, 2003.

[3] http://www.ibm.com/smarterplanet/us/en/smarter_cities/overview/

[4] IEEE Society of Robotics and Automation's Technical Committee on Networked Robots, available: http://www-users.cs.umn.edu/~isler/tc/

[5] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *Network, IEEE,* vol. 26, pp. 21-28, 2012.

[6] J. J. Kuffner, "Cloud-enabled robots," *IEEE-RAS International Conference on Humanoid Robotics, Nashville, TN*, 2010.

[7] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *Automation Science and Engineering, IEEE Transactions on,* vol. 12, pp. 398-409, 2015.

[8] P. Mell and T. Grance, "The Nist Definition of Cloud Computing," NIST Special Publication 800-145, Sept. 2011, available: http://csrc.nist.gov/publi- cations/nistpubs/800-145/SP800-145.pdf.

[9] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, *et al.*, "DAvinCi: A cloud computing framework for service robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 3084-3089.

[10] P. Salvini, C. Laschi, and P. Dario, "Do Service Robots Need a Driving License?," *IEEE Robotics and Automation Mag*., vol. 18, no. 2, 2011, pp. 12–13.

[11] Z. Du et al., "Design of A Robot Cloud Center," *Proc. IEEE ISDAS 2011*, 2011, pp. 269–75.

[12] *OpenStack*. http://www.openstack.org/

[13] *Docker*. https://www.docker.com/

[14] *HIPI-Hadoop*. http://hipi.cs.virginia.edu/

[15] Benavidez, P., Muppidi, M., Rad, P., Prevost, J. J., Jamshidi, M., & Brown, L. (2015, April). Cloud-based realtime robotic Visual SLAM. In Systems Conference (SysCon), 2015 9th Annual IEEE International (pp. 773-777). IEEE

[16] H. L. Akin, A. Birk, A. Bonarini, G. Kraetzschmar, P. Lima, D. Nardi, *et al.*, "Two hot issues in cooperative robotics: Network robot systems, and formal models and methods for cooperation," *A white paper, EURON Special Interest Group on Cooperative Robotics,* 2008.

[17] Mehdi Roopaei, Sos Agaian, Mehdi Shadaram, and M .K. Eghbal, "Noise-free rule-based fuzzy image enhancement", Electronic Imaging 2016, San Francisco, CA, Feb 14-18, 2016

[18] M. K. Eghbal, M. Shadaram, "Tandem-Modulator Generated W-Band OCDMA Radio-Over-Fiber System" International Conference on Transparent Optical Networks (ICTON 2016), Trento, Italy, July 10-16, 2016.

[19] M. Ghanat Bari, N. Ramirez, Z. Wang, and J. M. Zhang, "MZDASoft: a software architecture that enables large- scale comparison of protein expression levels over multiple samples based on liquid chromatography/tandem mass spectrometry," *Rapid Communications in Mass Spectrometry,* vol. 29, no. 19, pp. 1841-1848, 2015.

[20] M. G. Bari, X. Ma, and J. Zhang, "PeakLink: a new peptide peak linking method in LC-MS/MS using wavelet and SVM," Bioinformatics, pp. btu299, 2014.

[21] M. Ghanat Bari, F. Ghanat Bari, and J. Zhang, "The positive and random impulse noise reduction using ann and Gaussian recursive filter." pp. 763-767.

[22] A. Sahba, Y. Zhang, M. Hays, and W.-M. Lin, "A Real-Time Per-Thread IQ-Capping Technique for Simultaneous Multi-threading (SMT) Processors." pp. 413-418.

[23] A. Sahba, R. Sahba, and W.-M. Lin, "Improving IPC in simultaneous multi-threading (SMT) processors by capping IQ utilization according to dispatched memory instructions." pp. 893-899.

[24] M. Bagheri, M. Madani, R. Sahba, and A. Sahba, "Real time object detection using a novel adaptive color thresholding method", International ACM workshop on Ubiquitous meta user interfaces (Ubi-MUI'11), Scottsdale, AZ, November 2011.