High Availability Services Using Containers In Edge Computing Embedded Platform

Gnanadeep Pallela, Manoj Ramesh Rao, Roop Kumar Sriramulu, Ruthra Vadivel Murugesan Department of Computer Engineering
San Jose State University
San Jose, USA
Email:{gnanadeep.pallela, manojrameshrao.rameshrao, roopkumar.sriramulu, ruthravadivel.murugesan}@sjsu.edu

Abstract—**The surge in devices connected to internet and data intensive applications has grown to an extent that network traffic management is essential. It is very important to ensure the bandwidth is high, hence processing of data is pushed to the edge of the network. This scenario is referred to as edge computing where millions of IoT devices are connected to edge computing nodes. It is essential to build failover edge nodes to ensure high availability of services to process real-time data. In todays scenario, high availability of service is ensured by load balancing swarm clusters. The applications running inside end nodes are containerized to eliminate dependencies so that it can be live migrated to another node during single point of failure. In this project, live migration across applications is achieved using checkpoint and restore method. The downtime during this method is contrasted against swarm cluster load balancing. This project is under research to achieve 100% availability by performing live migration across edge nodes.**

Keywords*—*Live migration, CRIU, ECEP, swarm clusters.

I. Introduction

The growth of IoT devices and technology has led to the generation of large amount of data that can be transformed in to actionable intelligence. Due to the generation of big data there is a considerable increase in the network traffic. Hence edge computing reduces network traffic by pushing computation to the edge of the network. The edge computing architecture consists of devices connected to edge nodes which are backed up by a datacenter. The availability of nodes providing service is critical to edge computing devices. To ensure high availability of these nodes, load balancing can be done and managed using dashboard. Currently Jelastic and Google cloud platform provide such an interface to manage and migrate applications across nodes. Swarm clusters can be created using containers and high availability of the service is ensured by balancing the load across clusters. Although swarm cluster provides an availability of 99.999%, a lot of research is being done to achieve 100% availability. In this project, we propose to build a tool for live migration across nodes in case of a single point of failure. By implementing live migration zero downtime can be achieved and node failure in a network can be eliminated.

II. Related Work and Background

The ultimate goal of any user is to get a service that is uninterrupted owing to any technical reasons. In the current scenario, high availability services are provided using solutions such as periodic data backup and load balancer. Though these solutions provide highly available services, they have their own shortcomings, which make the service unreliable and doesn't provide very high services at times.

Data backup is one of the oldest and existing solutions wherein the data from the server will be periodically stored as a back up. Whenever the master server goes down , then the data from the back up can be used to provide responses to the request. This solution is being in existence since 1980 using various devices such as tapes, disks or other storage devices. Data back up has evolved since its beginning with new ways and techniques added. Even though this services, tries to provide the user with best service as possible, there are many drawbacks of this method, which made to look for alternatives. The disadvantages are The Memory required to store the back up data is large, storage and maintenance of back up data is tough, if the data back up is damaged, then the entire system will collapse and data back up method poses many security threats when the data is stored on an external device.

Load balancing is another technique that is currently available to provide high availability services. This method has more advantages in terms of reduced downtime, security, reliability and scalability in comparison with data back up method. In this technique, to achieve high availability the requests to the server is shared among a pool of servers, which are managed by a load manager. In this way the overloading of a single server is eliminated. Although this method provides high available service, the shortcomings of this method are failure of the load manager will make the entire system to fail, hardware cost of building and maintaining pool of servers are higher, configuration of connection between the client and servers is cumbersome and when the number of requests received is higher than the the server to be assigned, then the response time will be higher.

Edge computing is one of the new concepts in the field of IoT, wherein the data being processed is at the edge of the network. This type of computing has more advantages in terms of reducing the traffic across the network.In the present architecture of clouldlets, the application will be managed by the platform on which it runs, using the web manager. This platform will be implemented on an environment that provides support for the application and this architecture is based on the edge computing tasks. Another solution that uses cloudlet architecture is based on OpenStack++ that uses edge nodes.These nodes are sensing nodes and the cloudlet provides managerial services. These edge nodes are configured by messaging protocols such as ”web application messaging protocol (WAMP) or web sockets”. The recent work by Abhishek Gurudutt et. al, is about creating a distributed edge computing platform, which can auto discover the edge nodes those are active in the environment. The discovered node are then used for the implementation of the multi-tenant applications.

III. Innovative Design

The project proposes to integrate live migration feature within the edge computing embedded platform []. The edge computing embedded platform helps in establishing end nodes and monitoring the same through a dashboard. Live migration has been achieved in the field of virtual machines (VM) using hypervisors. In order to perform live migration across nodes containers are used to eliminate the dependencies associated with an application. By performing automatic live migration 100% availability can be achieved in contrast to current 99.999% availability benchmark. The different approaches for performing live migration across end nodes were analyzed and tested on the model.

A. *End Nodes*

The end nodes chosen for the project was raspberry pi device which could be connected to sensors to fetch real time data. The containers can be created inside the end nodes using the platform. The status of the containers can be monitored and deleted using this platform. The end nodes communicate using web application messaging protocol (WAMP).

B. *Performing live migration across end nodes*

The project uses a live migration tool called ”Checkpoint and Restore In Userspace (CRIU)” to migrate the application lively from one host to another with zero or less downtime. To achieve this the application to be migrated is run in a container. The application needs to be containerized so that it doesnt have any dependencies upon migration to a new host. These dependencies could be a hindrance on the target host when trying to restore the application. The state of the application is frozen and stored as images in the path provided by the user. Once the process state has been captured into the directory it needs to be transferred to the target host to restore the process. In this project, we analysed both secure copy and rsyc to transfer files to the destination host. To reduce latency in file transfer and downtime, rsync file transfer utility was run as daemon across the shared filesystem. In this method, the state of the process running on source machine is saved on a periodic basis. By doing so, the process can be restored at any time on the target machine as the files would be in sync.

C. *Docker Swarm*

In this project we implemented this swarm cluster by deploying the virtual machines. The virtual machines are spun up by the use of a docker feature called the docker machine. It is a tool which will be connected to a type-2 virtual manager like VirtualBox or Vsphere. The resources of the nodes are all of default sizes and processing speeds. We can access these virtual nodes using the docker machine tool. We have deployed two virtual machines. The main host will act as the manager. By using the inbuilt docker feature, docker swarm, we create a docker swarm for making the host to act as a swarm manager. The swarm manager will give out a ssh key for the other nodes to be securely added to the swarm cluster. The virtual machines deployed will be added to the swarm cluster by secure logging in to the virtual machines and using the swarm join commands. Now we have successful swarm cluster.

Once the cluster is formed, we will deploy the Nginx web service onto the swarm manager. Now we check that the web service is working by sending a http request to the swarm manager. Next, we have to scale up the Nginx web service to the other two swarm nodes. Once this is scaled up, we can test whether the service will work on the other two nodes by giving separate http requests to the nodes. To check the real purpose of the clustering, the high availability service, we can make a virtual host manually down and do the http request. If everything works correct, we can find that the high availability service is provided.

IV. Technical Aspects

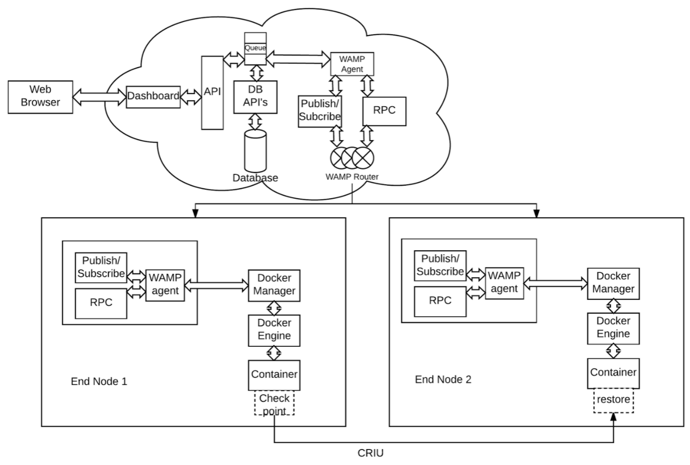

Fig. 1
ECEP Architecture with Live Migration

A. *CRIU*

The checkpoint and restore tool is used to perform live migration across the nodes. The state of the application to be migrated is saved in regular intervals so that the state of the application is available even if the node fails. The state of the application running on the source machine is in sync with the target machine using the rsync daemon running on target machine. The user can perform live migration when needed using the command line interface.

There are two stages to be performed on CRIU to migrate a process.

B. Checkpoint

Checkpointing of the application will be done on the primary host, where the process to be migrated will be frozen and converted into images. These images contain the necessary dependencies and the application files. These images are then transferred to the secondary host.

C. Restore

Restoring of the application takes place on the secondary host, wherein the images that are received from the primary host will be restored on the container that is running on this host. Using these images the application that is checkpointed on the primary host will be made to run here from the point where it was frozen.

D. Application

A simple game application is used which runs on nginx server. The application is a simple web interface that displays a game which can be played. Initially live migration was tested using this application within the same node and across two nodes. The dependencies of the application were containerized using Docker for live migration. The service is hosted on localhost of the machine running nginx server.

E. Live migration within the host

In this process, migration happens within the same host. The application migrated is a simple 2048 game. Initially the container is created on the host and the docker file is run on the container. The application is now accessible through a web browser. Using the CRIU checkpoint command the container is checkpointed. Once the process is checkpointed, the application in the web browser will be frozen and stopped. The images relevant to the application is formed using the commands and stored in the host. The process can now be again restored in the same host by running the images using CRIU restore commands.

F. Live migration across the host

This process is similar to the above process except the fact that there will be participation of two different hosts. For this process, we used a simple web application that displays a welcome page when run. The procedure remains the same until the creation of the images. Once the images are created, the images are transferred to the host system using CRIU commands that uses ssh. Here, the images are transferred to the host machine using the IP address of the secondary host. After the transfer of images, they can be restored.

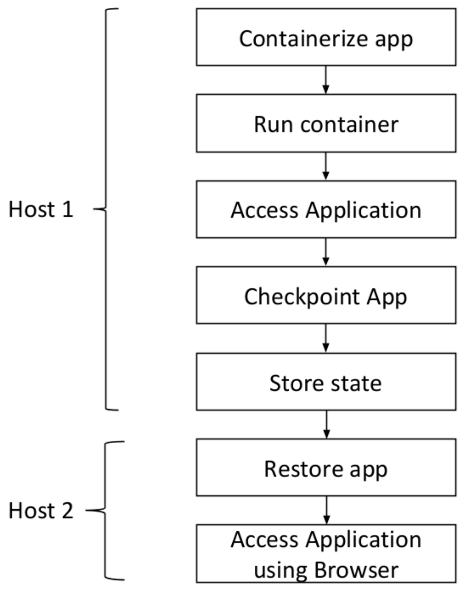

Fig. 2
Flowchart

G. Swarm Clusters

A swarm is a group of devices connected together. In Docker, a swarm cluster is group of docker engines networked between each other and as users, we can deploy services in the docker swarm. The swarm mode is a term for a group of docker engines connected together. When we initialize a docker swarm in a node, that node is called a swarm manager. When other docker enabled nodes join to the swarm, they are called swarm agents.

In our project, we have created a swarm cluster as an add-on feature to integrate with our Edge Computing Embedded Platform. The idea is to deploy a random web service in one of the docker nodes and scale it up to all the nodes. The docker swarm gives a load balancing feature pre-built with docker engine. The load balancing feature implemented is an ingress load balancing. The way this works is, the swarm manager will replicate the services which the user has scaled up to the number of nodes available in the cluster.

H. Future Scope

As part of further research, we are working on converting the command line interface into a web GUI for ease of use. During failure of node, automatic live migration will occur in the background that will prevent disruption of service to the user. The application can be containerized using other tools such as Kubernetes, OpenVZ which may have built-in features for live migration. The creation of swarm clusters can be integrated into the edge computing platform for ease of use. The platform can be deployed into cloud service thereby supporting cloudlet architecture.

V. Experimental Setup and Results

In this section, we discuss about the set up made for the experiment and the corresponding results obtained.

A. Experimental Setup for Live Migration

The end nodes used for testing the edge computing platform is raspberry pi. Since Docker is used for containerization, the application works in Raspbian Jessi operating system. For the system testing, two nods were configured for live migration. The Dockerfile is loaded into the end node using the edge computing platform. Once the container starts running the status is available in the dashboard. The files are transferred to target machine using rsync for restoring the process. The steps involved can be enumerated as follows:

1) Create Dockerfile for application (nginx-based game).
2) Build the Dockerfile and run the container.
3) Access the application on the localhost in the browser.
4) Checkpoint the container and store the state in filesystem.
5) Transfer the state to target host
6) Restore the application in the target machine.

The fig.3 shows the gaming application used as the application in the project.

B. Results

The gaming application was check pointed at the primary host and was able to restore on the secondary host. The application was able to run successfully on the migrated host from the point where it was check pointed in the primary host.

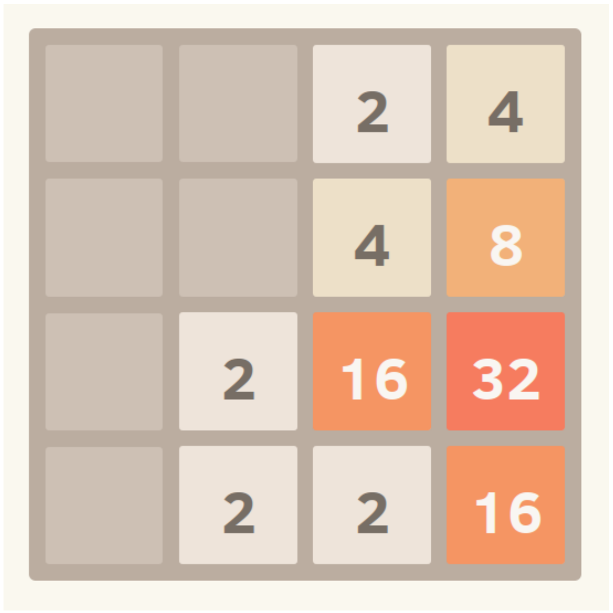

Fig. 3
Gaming Application

C. Experimental Setup for Swarm Clusters

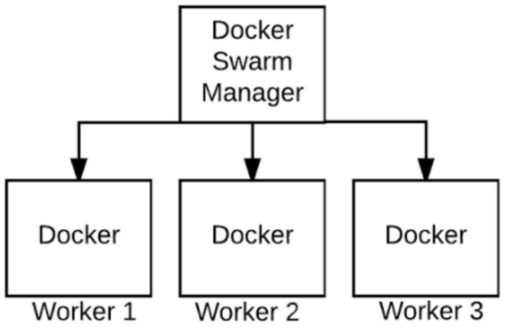

Fig. 4
Swarm cluster Setup

The fig. 4 shows the experimental set up used for the building of swarm clusters.

The set up consists of a docker swarm manager which manages the three worker docker nodes connected to it. In this every worker docker is automatically configured with separate IP addresses.

D. Results

The deployed web service has given a high availability and load balancing services in such a way that even if one of the node fails the user was provided with a continued service.

VI. Conclusion

The project proves that live migration can be used to achieve zero downtime by improving the filesystem and initiating automatic migration between hosts. Currently, the state of the process is saved periodically which can be further reduced to achieve live migration whenever desired. This can be implemented as part of the edge computing platform which will in turn make the end nodes highly available. The swarm clusters built using Docker provides a highly available service during node failure. By integrating creation of swarm clusters into the platform, the service can be deployed into different nodes.

References

[1] C.Divarakara, T.C. Kamal, A.Gurudutt and P.Prabhakaran. "Edge Computing Embedded Platform". San Jose State University, San Jose, CA, Tech. Rep.,Dec.2016.

[2] W. Li and A. Kanso, "Comparing Containers versus Virtual Machines for Achieving High Availability," *2015 IEEE International Conference on Cloud Engineering*, Tempe, AZ, 2015, pp. 353-358. doi: 10.1109/IC2E.2015.79

[3] R.Nelson.(2016, Dec. 14). "Docker Swarm Load Balancing with NGINX and NGINX Plus" [Online].Available at https://www.ieee.org/documents/ieeecitationref.pdf

[4] CRIU. (n.d). Live Migration. [Online]. Available at http://libguides.nps.edu/citation/ieee

[5] Docker Docs. (n.d). Deploy Services to a Swarm. [Online]. Available at https://docs.docker.com/engine/swarm/services/

Gnana Deep Pallela has completed Bachelors in Electronics and Communication Engineering in 2016. He is pursuing Masters in Networking specialization. He worked at IBM for six months as the Virtual Network administrator. He is interested in Network Security, IOT, and SDN.

Manoj Ramesh Rao is a computer engineering graduate student specializing in embedded systems at San Jose state University. He received his Bachelors degree in Electronica and Communication engineering from Anna University, Chennai. He has about two years of experience in software web development and automation at Verizon. His core interests include firmware development and software web development.

Roop Kumar Sriramulu completed his bachelors degree from Anna University in Chennai, India. Before pursuing his graduate studies, he worked as a Systems Engineer at Tata Consultancy Services, Chennai. He is currently interning at Cisco Systems, San Jose. He is proficient in python programming language. His specialization is in Computer Networks and Software Development.

Ruthra Vadivel Murugesan has completed his Bachelors in Electrical and Electronics Engineering in Anna University, India in 2014. After the completion of Bachelors, he was working for Infosys Limited for 2 years in India. Currently he is pursuing his Masters in San Jose State University with a specialization in Networking. He is interested in cloud based networking network security and Android app development.