Introduction

IEC Type3 mainly focus on Android Application running on edge ARM Cloud architecture with GPU/ vGPU Management. Also, ARM cloud games need to have the basic features of "cloud", such as flexibility, availability everywhere. Based on cloud infrastructure optimized for android application, providing ARM application services.

How to use this document

This document describes the construction, compilation and use of robox Android container environment. Introduce environment configuration, code download, build and compile, and how to use it.

This document is mainly suitable for users who build and compile robox container Android emulation.

Deployment Architecture
Figure 1: Deployment Main Framework

Figure 2: Robox Framework
Pre-Installation Requirements

- **Hardware Requirements**

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>2*arm64 server:</td>
<td></td>
</tr>
<tr>
<td>Arch</td>
<td>AARCH64</td>
</tr>
<tr>
<td>Processor model</td>
<td>1*Aarch64 processor( A72 inside)</td>
</tr>
<tr>
<td>RAM</td>
<td>16*DDR4-2933</td>
</tr>
<tr>
<td>Storage</td>
<td>10<em>2.5 inch SAS/SATA/SSD or 8</em>2.5 inch NVMe SSD</td>
</tr>
<tr>
<td>Network</td>
<td>1 onboard network card, each card supports 4<em>GE port or 4</em>10GE port or 4*25GE port</td>
</tr>
<tr>
<td>Power Supply</td>
<td>Power 100~240V AC240V DC</td>
</tr>
<tr>
<td>Scale</td>
<td>447 mm x 490 mm x 86.1 mm</td>
</tr>
</tbody>
</table>

ARM Server satisfies the Arm Server Ready certified.

- **Software Perequisites**

<table>
<thead>
<tr>
<th>Item</th>
<th>Comments</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>os</td>
<td>ubuntu 18.04.3/6 (<strong>key</strong>)</td>
<td></td>
</tr>
<tr>
<td>robox</td>
<td>robox is an Android container.</td>
<td>git clone [<a href="https://github.com/kunpengcompute/robox.git">https://github.com/kunpengcompute/robox.git</a> -b release-phase2.3](<a href="https://github.com/kunpengcompute/robox.git">https://github.com/kunpengcompute/robox.git</a> -b release-phase2.3)</td>
</tr>
<tr>
<td>robox Compile and run dependent packages</td>
<td>support for robox compilation and operation.</td>
<td>apt-get install build-essential cmake cmake-data debhelper dbus google-mock libboost-dev libboost-filesystem-dev libboost-log-dev libboost-iostreams-dev libboost-program-options-dev libboost-system-dev libboost-test-dev libboost-thread-dev libcap-dev libdbus-1-dev libdbus-cpp-dev libegl1-mesa-dev libgles2-mesa-dev libglib2.0-dev libgimpl-dev libgtest-dev liblxc1 libproperties-cpp-dev libprotobuf-dev libssl2-dev libstdc++-image-dev lxc-dev pkg-config protobuf-compiler</td>
</tr>
<tr>
<td>docker</td>
<td>needed by K8S/Robox</td>
<td>apt-get install docker.io</td>
</tr>
</tbody>
</table>

- **Components Version**

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anbox</td>
<td>Run Android applications on any GNU/Linux operating system.</td>
<td>8.4.3</td>
</tr>
<tr>
<td>Grafana</td>
<td>Compose and scale observability with one or all pieces of the stack</td>
<td>2.34.0</td>
</tr>
<tr>
<td>Prometheus</td>
<td>Cloud native system performance monitoring</td>
<td>2.34.0</td>
</tr>
<tr>
<td>K8s</td>
<td>container orchestration engine for automating deployment, scaling, and management of containerized applications</td>
<td>k8s: v1.23.5; kube-apiserver:v1.21.11; kube-scheduler:v1.21.11; kube-proxy:v1.21.11; etcd:3.4.13-0; etcd:3.4.13-0; etcd:3.4.13-0; coredns:v1.8.0</td>
</tr>
</tbody>
</table>

- **Database Perequisites**

<table>
<thead>
<tr>
<th>Item</th>
<th>Comments</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

schema scripts: N/A
Other Installation Requirements

- Jump Host Requirements: N/A
- Network Requirements: The network connection is normal and can communicate with the external network.
- Bare Metal Node Requirements: N/A
- Execution Requirements (Bare Metal Only): N/A

Installation High-Level Overview

Bare Metal Deployment Guide

- Install Bare Metal Jump Host: N/A
- Creating a Node Inventory File: N/A
- Creating the Settings Files: N/A
- Running: N/A

Upstream Deployment Guide

- Upstream Deployment Key Features: N/A
- Special Requirements for Upstream Deployments: N/A
- Scenarios and Deploy Settings for Upstream Deployments: N/A
- Including Upstream Patches with Deployment: N/A
- Running: N/A
- Interacting with Containerized Overcloud: N/A

Verifying the Setup as defined in the Akraino validation feature project plus any additional testing specific to the blueprint

Install Main Components

Since the components and images required by the project are relatively large, and the process of compilation takes time, we store the compiled images on the Github repository.

The link is: https://github.com/ysemi-computing/RoboxWidget.git

```bash
  git clone https://github.com/ysemi-computing/RoboxWidget.git
```

After this step, the code structure is as follows:

```
  RoboxWidget/
  android.img
  components
  grafana-server
  node_exporter
  perf_exporter
  prometheus
  README.md
```

Run Anbox On Ubuntu18.04

Build Android image for anbox
Code cloning and compilation is very time-consuming. If you just want to deploy and experience the Android system by iec, you can skip this section, because the project you just cloned already exists.

Firstly, you should install a ubuntu 18.04 system on Aarch64 processor, and switch to root user, then run the script “build_android_image.sh”

After about two hours, the Android image was compiled successfully, and the results are as follows:

```
ls out/target/product/arm64/
android-info.txt obj previous_build_config.mk recovery symbols system.img build_fingerprint.txt cache.img data gen module-info.json ramdisk.img root system userdata.img
```

### Required packages installation

Before running anbox, we need to install some necessary packages, you can execute the fol script “arm64_env_setup.sh”

After this, the image of anbox has been completed, which can be viewed through docker related commands. Type the command below:

```
sudo docker images | grep robox
```

### Run robox android system

After the above two scripts are executed, the robox operating environment is ready, and the entire directory structure of robox can be seen as follows

```
android
binaryFiles
cmake
cmake_uninstall.cmake.in
COPYING.GPL
cross-compile-chroot.sh
data
docs
external
kernel
patch
products
scripts
src
```

Step1: You can copy the binaryFiles in the robox code to the some place, then run robox

start the first container instance

```
./robox -v start 1
```

1 is the id, used by session manager and docker container
Step2: Check the docker instance process and session manager process.

```bash
sudo docker ps | grep instance
ps -aux | grep session
```

step3: Log in the container and confirm whether the robbox instance started successfully.

container name format: instance + id

```bash
docker exec -it instance1 sh
```

step4: get android property sys.boot_completed

```bash
getprop | grep sys.boot.completed
[sys.boot_completed]: [1] //1 which means start successfully
```

Cluster deployment and access by K8S

It is troublesome to start robbox through commands to deploy in a real environment. Here, the well-known K8S system is used to complete the deployment of robbox instances. On this basis, Robbox can be much more effectively deployed, run, monitored, and analyzed for multiple host nodes. Here is k8s cluster setup and container orchestration.

1 Basic Architecture
2 K8s Cluster construction

The k8s cluster construction can be completed through the warehouse code, you can do as follows

```
git clone https://gerrit.akraino.org/r/iec.git
cd iec/deploy/compass && bash deployIEC.sh
```

current configuration:

- Ubuntu Version 18.04
- Docker: 20+
- k8s: 1.21.3

Host Network:

- master: 192.168.10.66
- work: 192.168.10.62

3 Check cluster running status

```
kubectl apply -f https://addons.kuboard.cn/kuboard/kuboard-v3-swr.yaml
kubectl get cs
```
At the same time, we can present the status of the cluster through the k8s visual component. We use kuboard, it is very convenient to install and use. Then start kuboard with web browser

url master-ip-address:30080
user admin
password Kuboard123

Cloud platform monitoring & Analyze

1 Basic Architecture
2 Bootup basic components

Prometheus is only used for monitoring data. The real data needs an exporter. Different exporters export different data, these data are finally presented by Prometheus.

Only node exporter and perf exporter are used in this project, these two can be directly downloaded and installed with binary files. Prometheus is used for data monitoring, and then a better visual interface is Grafana, the data it needs can just be obtained through Prometheus.

This project only needs to run the script below:


After the script is executed, the required Prometheus and Grafana are downloaded and run in the background.

3 Analyze data by web browser

Let us explore data that Prometheus has collected about itself. To use Prometheus's built-in expression browser, navigate to http://localhost:9090/graph and choose the "Table" view within the "Graph" tab. Enter the below into the expression console and then click "Execute":

prometheus_target_interval_length_seconds
In addition, we can log in to grafana through a web browser and see the effect as shown below

url: http://localhost:3000

user: admin

password: admin
Troubleshooting

When the server restarts, the binder_linux module needs to be loaded and installed as follows:

```
sudo modprobe ashmem_linux
sudo rmmod binder_linux || true; sudo modprobe binder_linux num_devices=254
sudo chmod 777 /dev/ashmem /dev/binder*
```

If there is an external graphics card, you need to manually disable the built-in graphics card after each server restart. For example:

```
lspci | grep -in VGA
```

```
6:0007:41:00.0 VGA compatible controller: Huawei Technologies Co., Ltd. Hi1710 [iBMC Intelligent Management system chip w/VGA support] (rev 01)
```

```
echo 1 > /sys/bus/pci/devices/0007:41:00.0/remove
```

Maintenance

- Blue Print Package Maintenance
  - Software maintenance N/A
  - Hardware maintenance N/A
- Blue Print Deployment Maintenance (N/A)

Frequently Asked Questions

N/A

License

N/A

References

N/A
When porting an Android app to Anbox Cloud (usually in the form of an APK), there are a few issues that might cause your app to not function properly: