ARM 架构适配介绍

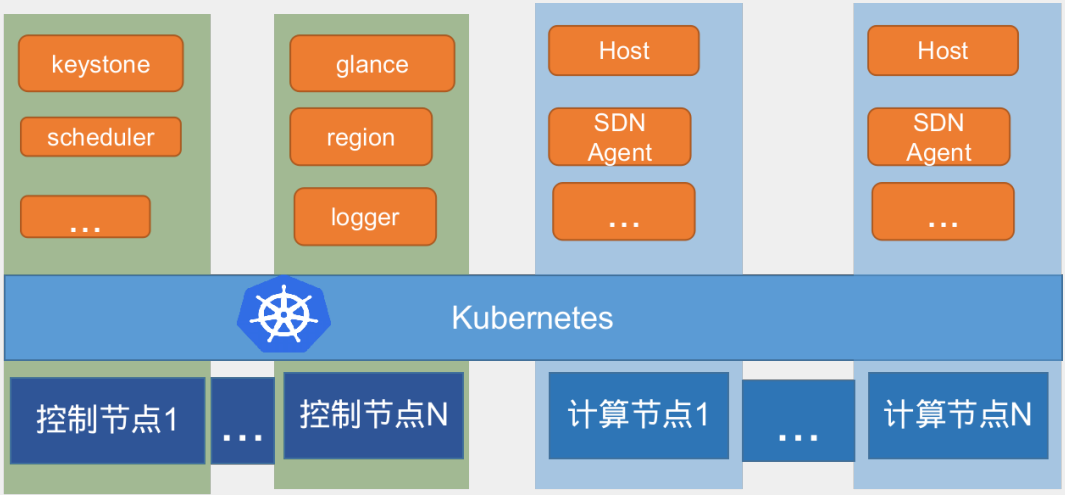
李泽玺

2021 年 4 月 28 日

云联壹云

1. 背景介绍

平台服务运行架构



- 服务容器化运行在 Kubernetes 之上
- 计算节点运行私有云虚拟机

CPU 架构简介

架构	特点	代表性使用者	64 位别名
X86	性能高、软件兼容性好	Intel、AMD 等	x86_64 、amd64
ARM	成本低、功耗低	苹果、华为等	arm64、aarch64

为什么适配 ARM

- 顺应大环境的发展趋势
- 响应国产化：鲲鹏、飞腾 CPU
- 提升产品竞争力：支持 ARM64 和 X86_64 虚拟化混合部署

2. 如何适配 ARM64？

需要解决关键问题

服务能够运行在 ARM64 架构上

- 部署异构 CPU 架构的 kubernetes 集群
  - 选择支持 ARM64 的 Linux 发行版
  - 安装依赖的软件（比如: docker-ce、kubelet 等）
- 统一容器镜像（x86_64 & arm64）

上层私有云业务支持 ARM64

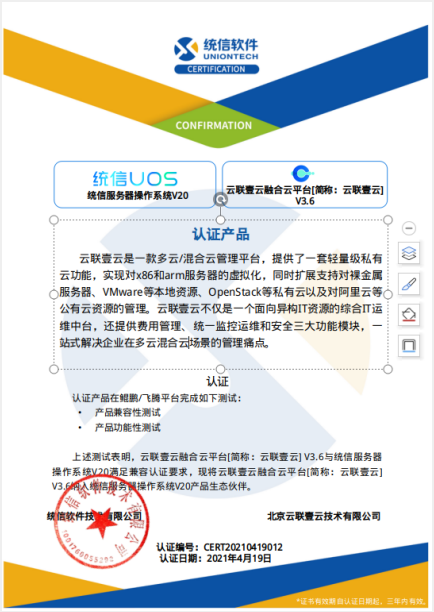
- 支持 arm64 虚拟机镜像、宿主机等
- 运行 arm64 虚拟化软件（qemu/kvm、openvswitch）

Linux 发行版选择

Debian 10(开源) / 统信 UOS(国产系统)

- 客户业务上要求在 ARM64 服务器上运行“统信 UOS”
- Debian 系列对 ARM64 支持良好
- 支持运行 docker 和 kubernetes
- CentOS 7 官方即将停止维护

统信 UOS 适配认证


统信软件
UNIONTECH
CERTIFICATION
CONFIRMATION
统信UOS
统信服务器操作系统V20
云联壹云融合云平台[简称：云联壹云]
V3.6
认证产品
云联壹云是一款多云/混合云管理平台，提供了一套轻量级私有云功能，实现对x86和arm服务器的虚拟化，同时扩展支持对裸金属服务器、VMware等本地资源、OpenStack等私有云以及对阿里云等公有云资源的管理。云联壹云不仅是一个面向异构IT资源的综合IT运维中台，还提供费用管理、统一监控运维和安全三大功能模块，一站式解决企业在多云混合云场景的管理痛点。
认证
认证产品在鲲鹏/飞腾平台完成如下测试：
• 产品兼容性测试
• 产品功能性测试
上述测试表明，云联壹云融合云平台[简称：云联壹云] V3.6与统信服务器操作系统V20满足兼容认证要求，现将云联壹云融合云平台[简称：云联壹云] V3.6纳入统信服务器操作系统V20产品生态伙伴。
统信软件技术有限公司
北京云联壹云技术有限公司
认证编号：CERT20210419012
认证日期：2021年4月19日
*证书有效期自认证日期起，三年内有效。

统一软件依赖 - 包管理工具

能够使用不同发行版的包管理工具安装相同的软件

举例：

```
# CentOS
$ yum install docker-ce kubeadm kubelet

# Debian
$ apt install docker-ce kubeadm kubelet
```

部署 kubernetes 集群

- ocboot 部署工具
  - https://github.com/yunionio/ocboot
  - 调用 ansible 实现部署 ARM64 + X86_64 的多节点 Kubernetes 集群

```
$ kubectl get nodes -o wide
NAME          STATUS VERSION         OS-IMAGE                KERNEL-VERSION                      CONTAINER-RUNTIME  LABELS
centos-x86-64 Ready  v1.15.9-beta.0  CentOS Linux 7 (Core)   3.10.0-1160.6.1.el7.yn20201125.x86_64 docker://19.3.9    kubernetes.io/arch=amd64,...
uos-arm64     Ready  v1.15.9-beta.0  UnionTech OS Server 20  4.19.0-arm64-server                 docker://20.10.5   kubernetes.io/arch=arm64,...
```

- Kubernetes 集群的好处
  - 提供不同 CPU 架构节点的统一管理
  - 支持容器服务运行

服务统一镜像运行

```
$ kubectl edit daemonset -n onecloud default-host
```

```
    containers:
    - command:
      - /opt/yunion/bin/host
      - --common-config-file
      - /etc/yunion/common/common.conf
      env:
      - name: HOST_OVN_ENCAP_IP_DETECTION_METHOD
      - name: HOST_OVN_SOUTH_DATABASE
        value: tcp:default-ovn-north:32242
      - name: HOST_SYSTEM_SERVICES_OFF
        value: host-deployer,host_sdnagent,telegraf
      - name: OVN_CONTAINER_IMAGE_TAG
        value: 2.10.5-1
      image: registry.cn-beijing.aliyuncs.com/yunionio/host:v3.6.10
      imagePullPolicy: IfNotPresent
      name: host
      resources: {}
      securityContext:
        privileged: true
      terminationMessagePath: /dev/termination-log
      terminationMessagePolicy: File
      volumeMounts:
      - mountPath: /etc/yunion
        name: etc-yunion
      - mountPath: /etc/yunion/pki
```

```
$ kubectl get pods -n onecloud -o wide | grep host
default-host-97hgc      3/3     Running            1          3d5h    10.127.40.252   centos-x86-64
default-host-p4gcw      3/3     Running            0          5d22h   10.127.100.9    uos-arm64
```

宿主机列表

宿主机

启用 | 禁用 | 批量操作 | 标签

添加筛选项

名称	状态	启用状态	服务	IP	CPU架构	物理CPU
debian-10-127-1... -	运行中	启用	在线	10.127.100.8（管理）	aarch64	8/13%
lzx-opensource-... -	运行中	启用	在线	10.127.40.252（管理）	x86_64	4/25%

私有云业务支持 ARM64

镜像列表

虚拟机创建


主机
主机
虚拟机
裸金属
反亲和组
主机模板
弹性伸缩组
镜像
系统镜像
主机镜像
存储
硬盘
硬盘快照
主机快照
自动快照策略
平台：
云联壹云
根据选择的区域不同，平台的可用类型不同且目前只有云联壹云支持GPU云服务器、云硬盘
CPU架构：
aarch64
x86_64
是否配置GPU：
关
目前只有云联壹云支持GPU云服务器
CPU核数：
1核
2核
4核
8核
12核
16核
24核
32核
内存：
2 GB
4 GB
8 GB
12 GB
16 GB
套餐：
全部
通用型
计算优化型
内存优化型
类型
区域
规格
CPU(核)
通用型
Default
ecs.g1.c2m2
2

虚拟机列表

虚拟机 全部 本地IDC

新建 开机 关机 重启 批量操作 标签

添加筛选项

名称	状态	IP	CPU架构	密码	安全组	平台	宿主机
lzx-x86-vm	运行中	192.168.21.253(私有)	x86_64		Default		lzx-opensource-v3…
lzx-arm-vm	运行中	192.168.22.253(私有)	aarch64		Default		debian-10-127-10…

3. 技术细节

概要

- 如何制作统一容器镜像（支持 X86_64 和 ARM64）？
  - docker buildx
  - 交叉编译
- 私有云相关技术细节

统一容器镜像

docker buildx 方案

- docker 原生支持的多架构镜像制作方案
- 官方介绍: https://docs.docker.com/buildx/working-with-buildx/

使用交叉编译然后打包

- 分别编译打包出 x86_64 和 arm64 的容器镜像，然后捆绑到一起

编写适用 buildx 的 Dockerfile

```
FROM --platform=$BUILDPLATFORM golang:alpine AS build
ARG TARGETPLATFORM
ARG BUILDPLATFORM
RUN echo "I am running on $BUILDPLATFORM, \
    building for $TARGETPLATFORM" > /log

FROM alpine
COPY --from=build /log /log
```

统一容器镜像 - docker buildx

使用 buildx 同时制作 x86_64 和 arm64 架构的镜像

```
$ docker buildx build \
    -t zexi/mul-arch:test \
    --push \
    --platform linux/amd64,linux/arm64 .
```

使用 buildx 同时制作 x86_64 和 arm64 架构的镜像

```
$ docker buildx build \
    -t zexi/mul-arch:test \
    --push \
    --platform linux/amd64,linux/arm64 .
```

```
[+] Building 12.7s (16/16) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [linux/arm64 internal] load metadata for docker.io/library/alpine:latest
 => [linux/amd64 internal] load metadata for docker.io/library/golang:alpine
 => [linux/amd64 internal] load metadata for docker.io/library/alpine:latest
 => [auth] library/alpine:pull token for registry-1.docker.io
 => [auth] library/golang:pull token for registry-1.docker.io
 => [linux/amd64 build 1/2] FROM docker.io/library/golang:alpine@sha256:49c07aa83790aca732250c2258b5912659df3
 => => resolve docker.io/library/golang:alpine@sha256:49c07aa83790aca732250c2258b5912659df31b6bfa2ab428661bc6
 => [linux/arm64 stage-1 1/2] FROM docker.io/library/alpine@sha256:69e70a79f2d41ab5d637de98c1e0b055206ba40a81
 => => resolve docker.io/library/alpine@sha256:69e70a79f2d41ab5d637de98c1e0b055206ba40a8145e7bddb55ccc04e13cf
 => [linux/amd64 stage-1 1/2] FROM docker.io/library/alpine@sha256:69e70a79f2d41ab5d637de98c1e0b055206ba40a81
 => => resolve docker.io/library/alpine@sha256:69e70a79f2d41ab5d637de98c1e0b055206ba40a8145e7bddb55ccc04e13cf
 => CACHED [linux/amd64 build 2/2] RUN echo "I am running on linux/amd64,  building for linux/arm64" > /log
 => CACHED [linux/arm64 stage-1 2/2] COPY --from=build /log /log
 => CACHED [linux/amd64 build 2/2] RUN echo "I am running on linux/amd64,  building for linux/amd64" > /log
```

查看镜像的信息

```
$ docker manifest inspect zexi/mul-arch:test
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 734,
         "digest": "sha256:95ee1d0e59f27882bd66d398e77d05a3b26acbacfcc7478c93a28de0d09daa37",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 734,
         "digest": "sha256:6412061faacc36569de5cf9f6ad76d936db17a7bcf5069b82c063d83516c9918",
         "platform": {
            "architecture": "arm64",
            "os": "linux"
         }
      }
   ]
}
```

统一容器镜像 - docker buildx

buildx 是如何在 x86_64 的机器上制作 arm64 的镜像的？

buildx 是如何在 x86_64 的机器上制作 arm64 的镜像的？

- 通过 binfmt_misc 模拟 arm64 硬件的用户空间
- 调用 qemu 的用户态模式编译程序

```
    PID USER      PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
 128239 root       20   0 1111M  109M 13072 R 236.  0.3  0:05.73 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128274 root       20   0 1111M 97132 13224 S 190.  0.3  0:04.62 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128388 root       20   0 1048M 85612 12936 S 133.  0.3  0:03.06 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128418 root       20   0  983M 79640 12384 R 58.0  0.2  0:01.73 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128541 root       20   0  787M 40876 11112 R 40.9  0.1  0:00.62 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128543 root       20   0  788M 41552 11284 S 40.9  0.1  0:00.62 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128552 root       20   0  595M 42116 12336 R 33.0  0.1  0:00.50 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128561 root       20   0  725M 37100 10716 S 27.0  0.1  0:00.41 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 128578 root       20   0  595M 29452 10304 R 16.5  0.1  0:00.25 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
 127971 root       20   0 2010M 80448 11612 S 15.2  0.2  0:08.30 qemu-aarch64-static /usr/bin/go build -mod vendor -
 128575 root       20   0  595M 29332 10256 R 15.2  0.1  0:00.23 qemu-aarch64-static /usr/lib/go/pkg/tool/linux_arm6
   1102 lzx        20   0 1911M  154M  121M R 12.5  0.5 12:23.52 sway
```

交叉编译: 直接在 x86_64 开发机上编译 arm64 二进制

```
$ cat /tmp/t.go
package main

func main() {}

# x86_64 机器上直接编译
$ go build -o t_x86_64 /tmp/t.go
$ file ./t_x86_64
./t_x86_64: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked

# 指定 GOARCH=arm64 环境变量, 交叉编译
$ GOARCH=arm64 go build -o t_arm64 /tmp/t.go
$ file ./t_arm64
./t_arm64: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), statically linked
```

统一容器镜像 - 不同架构镜像打包

将交叉编译后的 x86_64 和 arm64 容器镜像组合到一起

- x86_64 镜像：service:v1_x86_64
- arm64 镜像：service:v1_arm64

组合：service:v1 = service:v1_x86_64 + service:v1_arm64

统一容器镜像 - 不同架构镜像打包

将交叉编译后的 x86_64 和 arm64 容器镜像组合到一起

- x86_64 镜像：service:v1_x86_64
- arm64 镜像：service:v1_arm64

组合：service:v1 = service:v1_x86_64 + service:v1_arm64

```
# 创建 service:v1 的 manifest 镜像
$ docker manifest create service:v1 service:v1_x86_64 service:v1_arm64

# 标记镜像 service:v1_arm64 的架构为 arm64
$ docker manifest annotate service:v1 service:v1_arm64 --arch arm64

# 标记镜像 service:v1_x86_64 的架构为 amd64
$ docker manifest annotate service:v1 service:v1_x86_64 --arch amd64

# 将 service:v1 镜像上传
$ docker push service:v1
```

统一容器镜像 - buildx 和交叉编译打包对比

方式	速度	环境依赖	复杂度
buildx + binfmt_misc	慢	本地 x86 机器	低
buildx + ssh 远程节点	快	本地 x86 + 远端 arm64 机器	中
交叉编译 + manifest 打包	快	本地 x86 机器	高

结论

- 前端不需要编译的服务：使用 buildx + binfmt_misc
- 后端编译型的服务：使用交叉编译然后打包

私有云管理 - 虚拟化软件

- 交叉编译 qemu 虚拟化软件: qemu-aarch64

```
$ ./configure --target-list=aarch64-softmmu
```

- KVM 虚拟化加速: debian 10 4.19.0 aarch64 内核原生支持
- openvswitch 网络虚拟交换机

```
$ apt install openvswitch-switch
```

私有云管理 - 业务支持

支持 arm64 宿主机

https://github.com/yunionio/yunioncloud/pull/3594

支持 arm64 虚拟机镜像和调度

https://github.com/yunionio/yunioncloud/pull/7620

Thanks

Q&A