



CIP 云服务器使用手册

产品名称：CIP 加速方案

文档版本: V1.3.4

目录

一 简介	3
二 创建实例.....	4
1.选择实例规格为 ecs.f1-c8f1.2xlarge , 步骤如下：	4
2.选择收到的共享的镜像 , CIP_JpegOpen_V2.1 :	4
3.云服务器其他配置请按需求自行修改。	4
4.完成创建。	4
三 运行 demo	5
1.在控制台查看刚才开启的实例的详情 , 记录实例 ID.....	5
2.登入实例.....	5
四 软件 SERVICE.....	8
五 软件开发	9
OpenCV	9
1.环境变量配置	9
2.头文件	9
3.基于 OpenCV 的 API 拓展 (C++ API)	9

一 简介

CIP 加速器在软件层支持 OpenCV , ImageMagick。同时可依照客户需求，提供指定版本及定制化修改的软件方案。整体的软件架构图如图 1-1 所示：

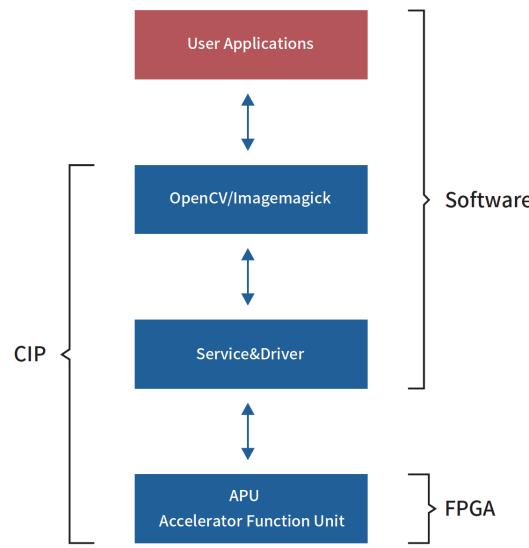


图 1-1.软件架构图

基于 OpenCV , CIP 加速器软件拓展了部分函数功能和接口，兼容标准版 OpenCV , 支持 C、C++、Python、Java 版 API 调用。

基于 ImageMagick , CIP 加速器软件提供一个包含拓展接口的动态链接库，兼容标准版 ImageMagick , 支持 C、Go 版 API 调用。

二 创建实例

1. 选择实例规格为 ecs.f1-c8f1.2xlarge , 步骤如下 :

基础配置 (必填)

计费方式: 包年包月 (已选)

地域: 华南 1 (深圳) (已选)

实例

- I/O 优化实例
- 实例规格族: ecs.f1-c8f1.2xlarge
- 实例配置: 8 vCPU, 60 GiB 内存, 1 * INTEL ARRIA 10 GX 1150 GPU
- 架构: 异构计算 GPU / FPGA (已选)
- 分类: GPU 计算型 (已选)

规格族	实例规格	vCPU	内存	GPU/FPGA	实例本地存储	处理器型号	处理器主频	内网带宽	内网收发包	支持 IPv6	规格参考价
FPGA计算型 f1	ecs.f1-c8f1.2xlarge	8 vCPU	60 GiB	1 * INTEL ARRIA 10 GX 1150	-	Intel Xeon E5-2682v4	2.5 GHz	2 Gbps	80 万 PPS	是	8.66 元/时
FPGA计算型 f1	ecs.f1-c8f1.4xlarge	16 vCPU	120 GiB	2 * INTEL ARRIA 10 GX 1150	-	Intel Xeon E5-2682v4	2.5 GHz	4 Gbps	200 万 PPS	是	17.33 元/时
FPGA计算型 f1	ecs.f1-c8f1.7xlarge	28 vCPU	112 GiB	1 * INTEL ARRIA 10 GX 1150	-	Intel Xeon E5-2682v4	2.5 GHz	5 Gbps	200 万 PPS	是	15.14 元/时
FPGA计算型 f1	ecs.f1-c8f1.14xlarge	56 vCPU	224 GiB	2 * INTEL ARRIA 10 GX 1150	-	Intel Xeon E5-2682v4	2.5 GHz	10 Gbps	450 万 PPS	是	30.28 元/时

2. 选择收到的共享的镜像 , CIP_JpegOpen_V2.1 :

公共镜像

自定义镜像

共享镜像

镜像市场

CIP_JpegOpen_V2.1

3. 云服务器其他配置请按需求自行修改。

4. 完成创建。

三 运行 demo

1. 在控制台查看刚才开启的实例的详情，记录实例 ID

The screenshot shows the Alibaba Cloud ECS instance list interface. A new instance has been created and is listed. The instance ID 'i-wz926pieyxhu6tunm42' is highlighted with a red box. Other details visible include the instance name 'i-wz926pieyxhu6tunm42', location '深圳 可用区D', IP address '47.112.128.173(公有)', status '运行中', network type '专有网络', configuration '28 vCPU 112 GB (I/O优化) FPGA: INTEL ARRIA 10 GX 1150 ecs.f1-c28f1.7xlarge 50Mbps (峰值)', and creation time '2019年4月12日 15:22 创建'. There are also links for '管理 | 远程连接', '更改实例规格 | 更多'.

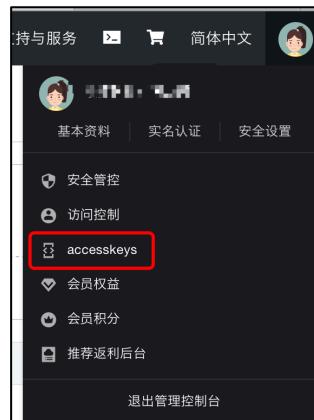
2. 登入实例

3. 设置大页

```
# sudo bash -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages"
```

4. 获取账号的 accesskey 信息

打开阿里云控制台页面，鼠标移到头像位置，选中“accesskey”。如果之前没有创建过 Accesskey 请创建一个，并记录 AccessKeyID 和 AccessKeySecret.



用户AccessKey					创建AccessKey
AccessKey ID	Access Key Secret	状态	创建时间	操作	
LTAloyfF6sGM11pj	显示	启用	2019-03-14 14:47:00	禁用 删除	
LTAISQjRYx4x2Yxa	显示	启用	2017-11-14 17:45:12	禁用 删除	

5. 配置 faascmd 工具

```
# faascmd config --id=<yourAccessKeyID> --  
key=<yourAccessKeySecret>
```

6. 关联 OSS 桶

如果阿里云账户中没有创建 OSS 存储桶，“hereIsYourBucket” 可填写任意值

```
# faascmd auth --bucket=hereIsYourBucket
```

7. 获取当前 FPGA 实例的 FPGAUUUID

```
# faascmd list_instances --instanceId=实例 ID
```

该命令输出可以看到 "FPGAUUUID": "0x198494329" 样式的结果，示例内容 "0x198494329" 为 FPGAUUUID



```
[root@iZwz9c9ah69vte09bz56wcZ test]# faascmd list_instances --instanceId=i-w...  
{  
    "Instances": {  
        "instance": [  
            {  
                "DeviceBDF": "05:00.0",  
                "FpgaStatus": "valid",  
                "FpgaType": "intel",  
                "FpgaUUID": "0x198494329",  
                "InstanceId": "i-w...  
                "ShellUUID": "V1.1"  
            }  
        ]  
    }  
}
```

8. 停止服务

```
# sudo service acceld stop
```

9. 下载 FPGA 镜像到 FPGA 板卡

```
# faascmd download_image --instanceId=实例 ID --  
fpgauuid=FPGAUUUID --fpgatype=intel --imageuuid=intelb062e3c0-  
7efc-11e9-acee-6f22b0376ea1 --imagetype=afu --shell=V0.11 --  
owneralias=market
```

10. 确认 FPGA 下载是否成功

```
# faascmd fpga_status --instanceId=实例 ID --fpgauuid=FpgaUUID
```

如果返回结果里出现 "TaskStatus": "valid" 时，且 FpgaImageUUID 和下载镜像时的 FpgaImageUUID 一致，说明下载成功。

11. 重启服务并加载环境变量

```
# service acceld restart
```

```
# source /usr/accel/bin/environment.sh
```

12. 进入测试目录

```
# cd example
```

13. 将您需要跑的图库放置在/dev/shm/下，并将文件夹重命名为 input

14. 运行测试脚本，并记录查看测试结果

```
# bash run.sh
```

PS. 您可以通过修改 commands.cfg 和 run.sh 配置自己的测试设置

四 软件 SERVICE

镜像启动成功后，加速器可由后台 CIP SERVICE 服务进行统一调度和管理。修改 SERVICE 配置后，重启 SERVICE 服务使配置生效。

SERVICE 服务配置

配置文件路径：/usr/accel/service/cfg/acceld.cfg

配置选项见表 3-1：

DEV_TYPE	设备类型：默认值 auto
NUM_THREADS	SERVICE 服务并发线程数：默认值 8
LOG_FILE	SERVICE 服务日志文件存储位置：默认值 /var/log/acceld-%Y-%m-%d.log
LOG_LEVEL	SERVICE 服务日志等级：默认值 info，支持 trace、debug 模式。 debug 模式可查看加速器资源利用率及温度信息，trace 模式可查看完整的 log 信息。 trace 模式下可能会出现性能下降的情况。

表 3-1.SERVICE 配置选项及说明

SERVICE 服务管理

根据安装加速器的系统类型选择相对应的 SERVICE 服务管理方式。

启动服务：[sudo service acceld start](#)

停止服务：[sudo service acceld stop](#)

重启服务：[sudo service acceld restart](#)

查看状态：[sudo service acceld status](#)

五 软件开发

OpenCV

CIP 加速器软件层在标准 Open CV 的基础上提供一个包含加速接口 API 的动态链接库，使用前需配置相关环境变量。

1. 环境变量配置

CTAccel 环境变量配置

```
$ source /usr/accel/bin/environment.sh
```

JAVA 环境变量配置

```
$ sudo ln -s /usr/accel/cv/share/OpenCV/java/libopencv_java343.so  
/usr/accel/cv/lib/libopencv_java343.so  
$ sudo ldconfig
```

2. 头文件

加速接口 API 函数声明头文件包含在软件 SDK 组件包中，位置如下：

/usr/accel/imagemagick/include/AccelMagick.h

3. 基于 OpenCV 的 API 拓展（C++ API）

在 imgcodecs.hpp 头文件中包含 IMACCEL_FPGA_SFT_ENABLE 的宏定义，该参数用于控制软件容错机制的启用（默认值为 1）。当值为 1 时，启用软件容错，CIP 加速器不支持或处理失败的任务会自动调用相应的 OpenCV API 转由 CPU 计算进行容错处理，当值不为 1 时，CIP 加速器不支持或处理失败的任务拓展 API 会抛出包含字符串 “FPGA accelerator processing failed.” 的异常信息。

3.1 获取图片尺寸

```
Size imAccelGetSize ( const String& filename,  
                      int flags = IMREAD_COLOR  
                    )
```

获取图片尺寸。

参数：

filename	读取的图片文件名
flags	请参考标准 OpenCV cv::ImreadModes

3.2 获取图片尺寸

```
Size imAccelGetSizeBuff ( const InputArray _buff,  
                          int flags  
                        )
```

获取图片尺寸。

参数：

buf	输入内存 buffer
flag	请参考标准 OpenCV cv::ImreadModes

3.3 从文件读取图片并缩放

```
Mat imAccelRead ( const String& filename,  
                   int resize_width,  
                   int resize_height,  
                   double fx=0,  
                   double fy=0,  
                   int interpolation = INTER_LINEAR,  
                   int flags = IMREAD_COLOR,  
                   int fpga_sft = IMACCEL_FPGA_SFT_ENABLE  
                 )
```

从文件读取图片并进行缩放操作。该拓展函数功能与标准 OpenCV 的 cv::imread() + cv::resize() 组合调用方式相同。如只进行图片解码，不需要缩放，缩放参数 resize_width、resize_height 请赋值 imAccelGetSize() 获取的原始宽高。该拓展函数执行的缩放操作使用 CIP 加速器内置算法，当加速器处理失败时，若 fpga_sft 为有效值（1），则缩放操作使用 interpolation 参数指定的缩放算法调用 cv::resize() 进行处理，参数说明同 cv::resize() 函数。

参数：

filename	文件名
resize_width	缩放后的目标尺寸宽值
resize_height	缩放后的目标尺寸高值
fx	请参考标准 OpenCV cv::resize() 函数说明
请参考标准 OpenCV cv::resize() 函数说明	
interpolation	请参考标准 OpenCV cv::InterpolationFlags
flag	请参考标准 OpenCV cv::ImreadModes
fpga_sft	软件容错标识符，默认启用软件容错

3.4 保存图片到文件

```
bool imAccelWrite ( const String& filename,  
                    InputArray img,  
                    const std::vector<int>& params = std::vector<int>(),  
                    int fpga_sft = IMACCEL_FPGA_SFT_ENABLE  
                )
```

保存图片到文件。

参数：

filename	文件名
img	被保存的图片
params	请参考标准 OpenCV cv::ImwriteFlags
fpga_sft	软件容错标识符，默认启用软件容错

3.5 从内存 buffer 读取图片并缩放

```
Mat imAccelDecode ( InputArray buf,  
                    int flags,  
                    int resize_width,  
                    int resize_height,  
                    double fx=0,  
                    double fy=0,  
                    int interpolation = INTER_LINEAR  
                    int fpga_sft = IMACCEL_FPGA_SFT_ENABLE  
                )
```

从内存 buffer 读取图片并进行缩放操作，该拓展函数功能与标准 OpenCV 的 cv::decode() + cv::resize() 组合调用方式相同。其余特性同 imAccelRead() 函数。

参数：

buf	输入内存 buffer
flag	请参考标准 OpenCV cv::ImreadModes
resize_width	缩放后的目标尺寸宽值
resize_height	缩放后的目标尺寸高值
fx	请参考标准 OpenCV cv::resize() 函数说明
请参考标准 OpenCV cv::resize() 函数说明	
interpolation	请参考标准 OpenCV cv::InterpolationFlags
fpga_sft	软件容错标识符，默认启用软件容错

3.6 从内存 buffer 读取图片并缩放

```
Mat imAccelDecode ( InputArray buf,  
                    int flags,  
                    Mat* dst,  
                    int resize_width,  
                    int resize_height,
```

```

        double fx=0,
        double fy=0,
        int interpolation = INTER_LINEAR
        int fpga_sft = IMACCEL_FPGA_SFT_ENABLE
    )

```

从内存 buffer 读取图片并进行缩放操作，该拓展函数功能与标准 OpenCV 的 cv::decode() + cv::resize() 组合调用方式相同。其余特性同 imAccelRead() 函数。

参数：

buf	输入内存 buffer
flag	请参考标准 OpenCV cv::ImreadModes
dst	图片解码后的可选输出占位符
resize_width	缩放后的目标尺寸宽值
resize_height	缩放后的目标尺寸高值
fx	请参考标准 OpenCV cv::resize() 函数说明
请参考标准 OpenCV cv::resize() 函数说明	
interpolation	请参考标准 OpenCV cv::InterpolationFlags
fpga_sft	软件容错标识符，默认启用软件容错

3.7 编码图片并保存到内存 buffer

```

imAccelEncode ( const String& ext,
                 InputArray img,
                 CV_OUT std::vector<uchar>& buf,
                 const std::vector<int>& params = std::vector<int>(),
                 int fpga_sft = IMACCEL_FPGA_SFT_ENABLE
)

```

保存图片到 buffer。

参数：

ext	文件后缀名，用来定义输出格式
-----	----------------

img	被保存的图片
buf	图片存储 buffer
params	请参考标准 OpenCV cv::ImwriteFlags
fpga_sft	软件容错标识符，默认启用软件容错

3.8 按文件的方式生产缩略图

```
bool imAccelReadResizeWrite ( const String &input_filename,  
                           const String &output_filename,  
                           int resize_width,  
                           int resize_height,  
                           const std::vector<int>& params = std::vector<int>(),  
                           double fx=0,  
                           double fy=0,  
                           int interpolation = INTER_LINEAR,  
                           int flags = IMREAD_COLOR,  
                           int fpga_sft = IMACCEL_FPGA_SFT_ENABLE  
)
```

从文件读取图片并进行处理。该拓展函数功能与标准 OpenCV 的 cv::imread() + cv::resize() + cv::write() 组合调用方式相同。如不需要缩放，缩放参数 resize_width、resize_height 请赋值 imAccelGetSize() 获取的原始宽高。该拓展函数执行的缩放操作使用 CIP 加速器内置算法，当加速器处理失败时，若 fpga_sft 为有效值（1），则缩放操作使用 interpolation 参数指定的缩放算法调用 cv::resize() 进行处理，参数说明同 cv::resize() 函数。

参数：

input_filename	输入文件名
output_filename	输出文件名
resize_width	缩放后的目标尺寸宽值
resize_height	缩放后的目标尺寸高值
params	请参考标准 OpenCV cv::ImwriteFlags

fx	请参考标准 OpenCV cv::resize() 函数说明
请参考标准 OpenCV cv::resize() 函数说明	
interpolation	请参考标准 OpenCV cv::InterpolationFlags
flag	请参考标准 OpenCV cv::ImreadModes
fpga_sft	软件容错标识符，默认启用软件容错

3.9 按内存 buffer 的方式生产缩略图

```
bool imAccelDecodeResizeEncode ( InputArray in_buf,
                                CV_OUT std::vector<uchar> &out_buf,
                                const String &ext,
                                int resize_width,
                                int resize_height,
                                int flags,
                                const std::vector<int>& params = std::vector<int>(),
                                double fx=0,
                                double fy=0,
                                int interpolation = INTER_LINEAR,
                                int fpga_sft = IMACCEL_FPGA_SFT_ENABLE
                            )
```

从内存读取图片并进行处理。该拓展函数功能与标准 OpenCV 的 cv::imdecode() + cv::resize() + cv::imencode() 组合调用方式相同。如不需要缩放，缩放参数 resize_width、resize_height 请赋值 imAccelGetSizeBuff() 获取的原始宽高。该拓展函数执行的缩放操作使用 CIP 加速器内置算法，当加速器处理失败时，若 fpga_sft 为有效值（1），则缩放操作使用 interpolation 参数指定的缩放算法调用 cv::resize() 进行处理，参数说明同 cv::resize() 函数。

参数：

in_buf	输入图片内存 buffer
out_buf	输出图片内存 buffer

ext	文件后缀名，用来定义输出格式
resize_width	缩放后的目标尺寸宽值
resize_height	缩放后的目标尺寸高值
flag	请参考标准 OpenCV cv::ImreadModes
params	请参考标准 OpenCV cv::ImwriteFlags
fx	请参考标准 OpenCV cv::resize() 函数说明
请参考标准 OpenCV cv::resize() 函数说明	
interpolation	请参考标准 OpenCV cv::InterpolationFlags
fpga_sft	软件容错标识符，默认启用软件容错