

Parasoft+BCLOUD+iPipe 接入详细文档

目录

- 一、我们会用到什么？
 - 1.1 Parasoft 代码扫描工具
 - 1.2 BCloud 编译
 - 1.3 iPipe 和自动化脚本
- 二、本地 Parasoft + BCloud 进行代码扫描
 - 2.1 生成 Makefile
 - 2.2 编译并生成.bdf
 - 2.3 代码扫描：
- 三、接入 iPipe 流水线
 - 3.1 建立流水线
 - 3.2 修改 ci.yml 文件
 - 3.3 修改 BCLOUD 文件
 - 3.4 触发扫描

本文具体描述

1. （本地）如何配合着使用 parasoft 代码扫描和 Bcloud 编译
2. （上云）如何将上述功能接入 iPipe

一、我们会用到什么？

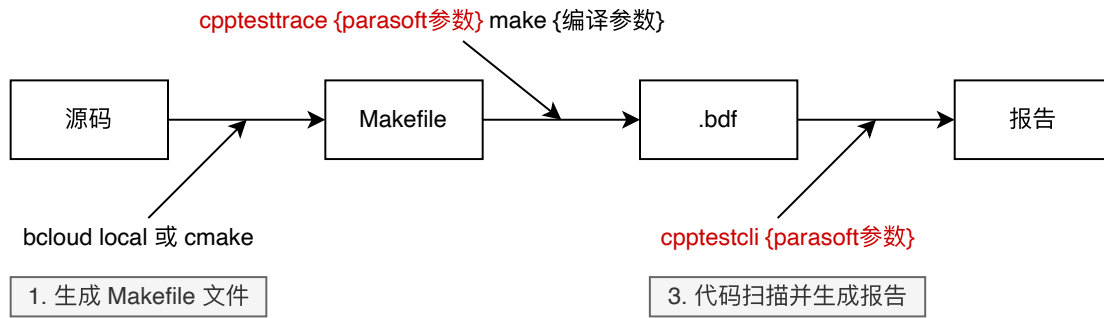
在这一部分，我们不会实际的进行扫描，只是让我们对整个流程熟悉起来，并熟悉一下要用到的工具

1.1 Parasoft 代码扫描工具

parasoft 代码扫描工具可以按照【某种车规规则】进行代码扫描。想要扫描代码，需要按照如下流程操作

- 1) 生成 Makefile
- 2) 编译过程中插入检测点，生成 .bdf 文件：使用 cpptesttrace 命令
- 3) 代码扫描，生成报告：使用 cpptestcli 命令

2. 这一步将 `cpptesttrace` 命令加在编译命令之前，在编译的过程中插入一些检测相关的东西，最后生成一个 `.bdf` 文件



这里主要用到了两个命令：1) `cpptesttrace`；2) `cpptestcli`；这里先简单介绍，具体的参数会在下文进行详细讲解

- 1) `cpptesttrace`：拼接在编译命令（`make`）之前，参与编译过程，最终生成 `bdf` 文件
- 2) `cpptestcli`：指定 `bdf`、扫描规则后，产出一份扫描报告

1.2 BCloud 编译

`bcloud` 包含两个组成部分

- 1) 配置文件：在代码库的根目录，通过 `BCLOUD` 文件来配置编译的依赖
- 2) 编译命令：
 - `bcloud local`：生成 `Makefile`
 - `bcloud build`：直接编译

在本教程中，我们仅会使用【`BCLOUD`】配置文件和【`bcloud local`】命令


1.3 iPipe 和自动化脚本

- 1) 流水线的触发：通过配置【分支】和【事件】作为触发条件
 - 比如：【`master`】分支出现【`MERGE`】事件时，触发 `iPipe`
- 2) 流水线实际要做的事情：通过在代码库中配置 `ci.yml`
- 3) 自动化脚本所在代码库 [文件页](#)：[baidu/adu-3rd/cptest *master](#)

二、本地 Parasoft + BCloud 进行代码扫描

在这一步中，我们将在本地跑通 `parasoft` 代码扫描

- ✅ 事先准备
 - 安装 `git`
 - 安装 `bcloud`：<http://buildcloud.baidu.com/bcloud/2-install>

- 代码库权限：  文件页： baidu/adu-3rd/cpptest *master，并下载
 - 使用 **v2023-10-26** 分支
- 被扫描的【模块的代码库】权限，并下载

2.1 生成 Makefile

使用 bcloud local 命令生成 makefile，相关生成参数来源【该模块的研发人员】


主要关注：

- **编译器类型**
- 目标平台（是不是跨平台编译——**交叉编译**）

进入【模块的代码库】，执行：

```
</> Bash  
  
1 # 比如  
2 bcloud local --target-arch='aarch64' --compiler="gcc930-aarch64-glibc-2.31-  
  ubuntu18-gnu" --profile-name="aarch64" --disable-cc-cxx
```

完成后我们将在当前目录下看到 Makefile 文件

-  如果提示找不到某种编译器，可以
- 联系吕敏
 - 或在http://buildcloud.baidu.com/bcloud/11-arm_build中寻找对应编译器的安装命令

常用编译器下载

```
</> gcc930-aarch64-glibc-2.31-ubuntu18-gnu Plain Text  
  
1 mkdir -p /home/opt/compiler && cd /home/opt/compiler && wget http://bcloud-  
  buildkit.bj.bcebos.com/build_env/gcc/custom-gcc/gcc930-aarch64-glibc-2.31-  
  ubuntu18-gnu.tar.gz -O gcc930-aarch64-glibc-2.31-ubuntu18-gnu.tar.gz && tar -zxf  
  gcc930-aarch64-glibc-2.31-ubuntu18-gnu.tar.gz && rm -f gcc930-aarch64-glibc-2.31-  
  ubuntu18-gnu.tar.gz && ln -s /home/opt/compiler/gcc930-aarch64-glibc-2.31-  
  ubuntu18-gnu /opt/compiler/gcc930-aarch64-glibc-2.31-ubuntu18-gnu
```

2.2 编译并生成.bdf

该部分命令由：cpptesttrace {parasoft参数} make {编译参数}；两部分组成

首先进入 `/baidu/adu-3rd/cpptest` 代码库的 `src/cpptest/bin`，可以看到 `cpptesttrace` 命令的二进制文件，这个命令主要有三个参数：

- `--cpptesttraceProjectName`：即模块名称，如 `gaia-service`
- `--cpptesttraceOutputFile`：即输出bdf文件的地址，如 `/baidu/anp3/gaia-service/gaia-service.bdf`
- `--cpptesttraceTraceCommand`：实际编译器的名字，需要用正则写法，编译器名称以\开头、以\$结束，不同编译器之间用|相连。
 - 如：`\gcc930-aarch64-glibc-2.31-ubuntu18$ | \gcc930-aarch64-glibc-2.31-ubuntu18-g\+\+$`
 - 其中 `\+` 是为了匹配 `+` 这个符号

现在我们完成了命令的第一部分，接下来是 `make` 编译命令。

由于我们使用 `bcloud` 生成的 `Makefile`，因此编译命令比较统一

- `make -j{cores} no-release.bcloud`
 - 其中 `cores` 为编译使用的CPU核数

完成上述步骤后我们得到了最终的命令：

```
</> Bash  
  
1 ~/baidu/adu-3rd/cpptest/src/cpptest/bin/cpptesttrace \  
2   --cpptesttraceProjectName=gaia-service \  
3   --cpptesttraceOutputFile=~/.baidu/anp3/gaia-service/gaia-service.bdf \  
4   --cpptesttraceTraceCommand=\gcc930-aarch64-glibc-2.31-ubuntu18$ | \gcc930-  
   aarch64-glibc-2.31-ubuntu18-g\+\+$ \  
5   make -j4 no-release.bcloud
```

如果顺利的话，我们将在 `~/baidu/anp3/gaia-service/gaia-service.bdf` 地址看到结果。

2.3 代码扫描：

代码扫描的命令为 `/baidu/adu-3rd/cpptest` 代码库的 `src/cpptest/bin/cli` 文件夹下的 `cpptestcli` 命令，该命令的主要参数如下：

- `-compiler`：编译器，可选的值为 `/baidu/adu-3rd/cpptest/src/cpptest/bin/engine/etc/compilers` 下的文件
夹名字
- `-input`：上一步生成的 bdf 的名字
- `-settings`：软件许可文件，该文件为 `~/baidu/adu-3rd/cpptest/src/cpptest/settings_network.txt`
- `-report`：扫描结果输出文件夹地址
- `-config`：扫描规则文件：若使用内置好的规则，则为 `builtin://MISRA C 2012` 等，更多选择详见
`/baidu/adu-3rd/cpptest/run_code_scan.sh` 的80行左右

- 也可以自定义规则，自定义规则详见 /baidu/adu-3rd/cpptest/src/cpptest/customize_rules
- -include (Optional): 扫描范围白名单，指定一个 .lst 文件，里面的元素会被扫描
- -exclude (Optional): 扫描范围黑名单，指定一个 .lst 文件，里面的元素不会被扫描

于是我们的扫描命令为：

</>

Bash

```
1 ~/baidu/adu-3rd/cpptest/src/cpptest/bin/cli/cpptestcli \  
2   -compiler gcc_9-aarch64_0  
3   -input ~/baidu/anp3/gaia-service/gaia-service.bdf  
4   -settings ~/baidu/adu-3rd/cpptest/src/cpptest/settings_network.txt  
5   -report ~/baidu/anp3/gaia-service/scan_report  
6   -config ~/baidu/adu-  
    3rd/cpptest/src/cpptest/customize_rules/misrac2012_misracpp2008.properties
```

该部分较为耗时，以 gaia-service 为例子，需要 18 分钟左右的时间

三、接入 iPipe 流水线

接入过程主要分为3个步骤：

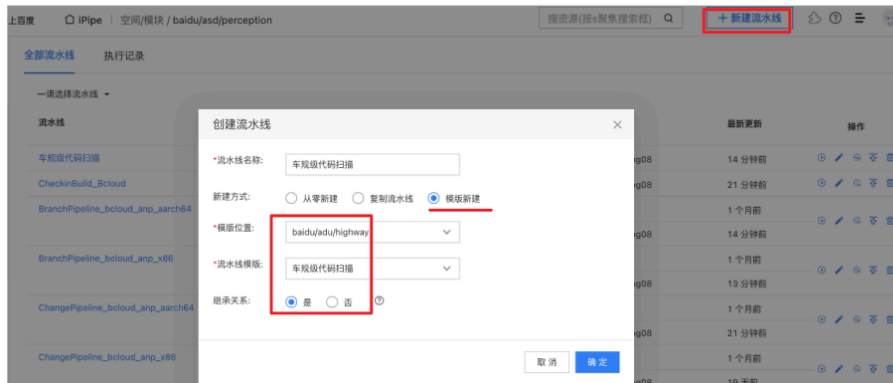
1. 在 iPipe 中建立流水线
2. 在 被扫描的模块的代码库中 修改 ci.yml
3. 在 被扫描的模块的代码库中 修改 BCLOUD

3.1 建立流水线

如图所示，安装模板建立流水线，之后按需配置触发条件

按模板的方式创建流水线，操作如下：

点击确定后，直接保存不用做任何修改！



⚠ 注意

当触发条件设置为 CHANGE 时，会阻塞每一次 CHANGE 的代码合入，影响代码合入效率。

可以加将触发条件设置为 MERGE，这样配置不会阻塞 merge 评审

详见资料：[☰ 分支命名规范和Pipeline整合](#)

3.2 修改 ci.yml 文件

在 被扫描的模块的代码库 的根目录下有 ci.yml 文件

- 在 Profiles: 下新增 profile
- **第8行，改为你的项目的编译器名称**
- **第9行，改为你的项目的 bcloud local 命令**
- 其余不变

```
</> Bash
1  - profile:
2    name : bcloud_parasoft
3    mode : AGENT
4    environment:
5      image: iregistry.baidu-int.com/idg-public/ubuntu-18_04:bcloud_dev_20220920
6    build:
7      command : export PATH=/root/.BCloud/bin:$PATH &&
8                export compiler_name=gcc930-aarch64-glibc-2.31-ubuntu18-gnu &&
9                bcloud local --target-arch='aarch64' --compiler="gcc930-aarch64-
glibc-2.31-ubuntu18-gnu" --profile-name="aarch64" --disable-cc-cxx &&
10               bash ../../adu-3rd/cpptest/run_code_scan.sh
11    artifacts:
12      release : true
```

3.3 修改 BCLLOUD 文件

在x86的逻辑中增加一行：CONFIGS("baidu/adu-3rd/cpptest@master@git_branch", NeedOutput())

如果不是x86的，根据业务需要 在对应位置添加即可

```
</> Bash
1 CONFIGS("baidu/adu-3rd/cpptest@v2023-10-26@git_branch", NeedOutput())
```

i 注意分支选择

上面的 v2023-10-26 为最新的 parasoft 软件，此外还有 master 分支，如何做选择：

- 如果不是交叉编译：使用 master
- 如果是交叉编译：使用 v2023-10-26 分支

- 有需要时联系 parasoft 工作人员，说 cpptestcli -compiler 参数需要自定义，我们需要提供【编译器名称】，parasoft 员工会帮忙修改编译器配置

3.4 触发扫描

流水线被触发后不会立刻执行代码扫描，需要手动确认

- 进入被触发的 iPipe，并点击开始按钮
- url和rule可以留空，点击确认，开始代码扫描
- 扫描完成后，会在第二个 stage 生成报告的链接，点击即可查看报告结果