[SCA] 多进程框架设计文档

文档历史

版本	修订日期	修订者	修订内容
v1.0	2021.05.07	lanjing.zeng@shopee.com	初次实现多进程框架

1、需求概况

1.1 需求背景

简要说明需求的背景

背景知识：安卓系统是基于进程来分配内存空间的，同一个进程共享同一份内存，所有功能模块都运行在同一个进程中；apk在运行过程若没有及时释放内存，内存将不断上升，apk会变得愈加卡顿；如果占用的内存超过了系统分配的上限，就会导致OOM。

目前WMS等项目都是采用单进程模式开发，拥有无需考虑进程间通信、代码简洁等优点，但存在以下弊端：

- 随着业务不断迭代，功能变得愈加复杂，对内存需求越来越大，而系统分配给进程的可用内存大小是有上限的；
- apk中有些功能之间并无直接联系，例如业务功能模块和网络监控模块是两个完全独立的模块，在单进程模式下，如果网络监控不可用，将导致业务模块也无法使用，这在现实是应该尽量避免的。

于是考虑将项目从单进程转换为多进程模式，转换为多进程后不仅可以申请更多的内存，减少OOM崩溃；且可以将独立的功能模块运行在不同的进程中，即使某个功能模块异常，其他的模块仍然能正常运行。

多进程模式下虽然有以上好处，但也存在以下问题：

- 因为进程间的内存是不共享的，所以多进程模式必须要考虑进程间的通信问题；
- 进程间通信时不适合传输大数据量；
- 静态成员和单例模式失效；
- Application将被多次创建，Application的onCreate()将被执行多次；

1.2 需求目的

简要说明需求要解决的问题及期望达到的效果

该框架主要解决进程间的通信问题。

2、技术方案设计

2.1 整体方案设计

可包含方案选型、整体架构、数据流、模块间关系等，如果是部分模块变动，需突出变动及影响部分，需对核心模块及关系进行文字描述，需具备良好的架构特性如通用性、扩展性等

多进程之间通信原理的选择：

目前多进程通信主要包括但不限于以下几种方式：Intent，AIDL，Messager，ContentProvider，Broadcast。

Intent：主要用于启动其他进程的Acitivity，例如在应用中调用系统的拨打电话服务，拍照服务等；

AIDL：与Service配合使用可实现跨进程通信，但是使用过程中需要客户端和服务端都定义相同的aidl文件，如果业务功能复杂，代码会显得很臃肿；且如果客户端和服务端端开连接后需要再次连接，极端情况下会出现不断重连的情况；

Messager：AIDL的简化版本，底层仍是AIDL实现。

ContentProvider：有两种用法，一种是与Sqlite配合实现数据库数据共享；一种是通过系统提供的API直接调用到另一个进程的方法；

Broadcase：一个进程发送广播后，另一个进程只能被动的接收广播，是一种被动的通信方式；发送者和接收者都不知道互相是谁，且比较消耗系统资源，会导致系统性能下降；

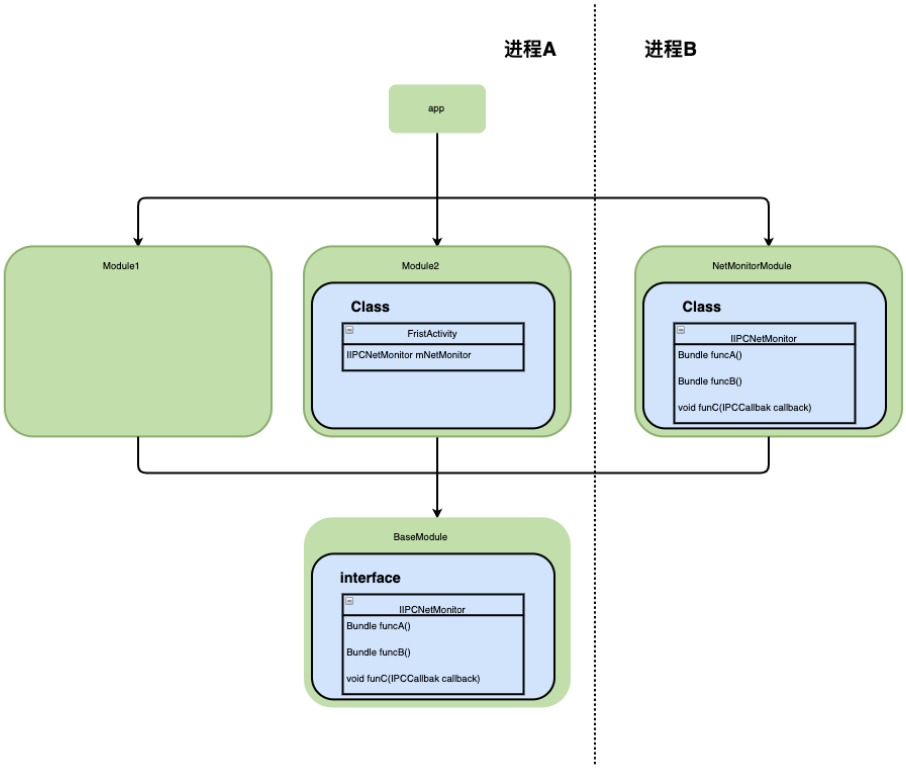
本框架选择使用ContentProvider来实现跨进程通信，主要是通过call()来调用到另一个进程的服务。考虑到call()方法是同步方法，框架在实现异步时需要借助Aidl来实现。

2.2 详细方案设计

可包含核心模块功能和交互实现方案详细设计，如：路由、布局、组件、交互、数据流、状态管理、兼容性、性能优化、异常处理、安全防护、扩展性及注意事项等等

考虑到项目目前架构和使用三方库的情况：项目是组件化架构，单个业务对应一个Module，Module间使用阿里的Arouter库来实现页面跳转和进程内通信。

为了使项目代码更加清晰易懂，这里实现的跨进程通信框架在使用方式上尽量与Arouter保持一致，以"网络监控服务"为例，框架的结构大致如下：



即：

- 在BaseModule中定义相关的接口及方法
- 在提供跨进程服务的Module中去写这个接口的实现
- 在需要使用跨进程服务的Module中去通过接口调用服务

2.2.1 关键注解及其使用方式

使用IIPCProvider.java作为IPC的预置接口类，实现该接口的类表示对外提供跨进程服务;

@IPC是该框架提供的关键注解之一；

2.2.1.1 在BaseModule中定义服务接口

```
public interface IPCNetMonitor extends IIPCProvider{

    Bundle funcA();

    Bundle funcB();

    void funC(IPCCallbak callback);

}
```

2.2.1.2 在网络监控Module即NetMonitorModule中写服务的实现

```
@IPC(“process”=””,authorities=””)
public class NetMonitor implement IPCNetMonitor {
    Bundle funcA(){
    }
    Bundle funcB(){
    }
    void funC(IPCCallbak callback){
    }
}
```

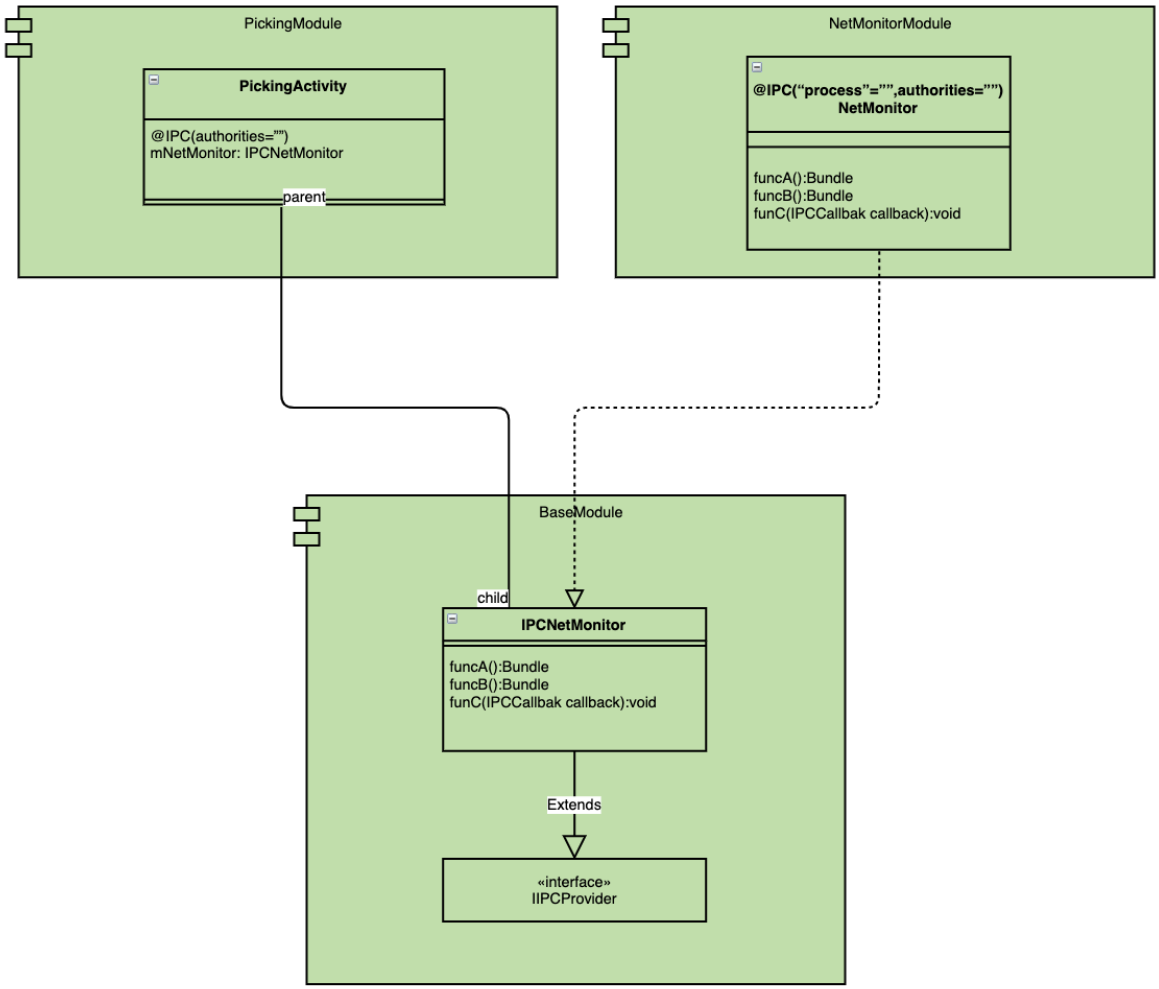
authorities：用来指定这个实现类归属于哪个ContentProvider,一个authorities对应一个ContentProvider，可能会存在多个跨进程服务对应一个ContentProvider的情况。

process：用来指定ContentProvider运行在什么进程。

2.2.1.3 现在假设在PickingModule中调用跨进程服务

```
public class PickingActivity{
     @IPC(authorities=””)
     IPCNetMonitor  mNetMonitor;
}
```

关键注解及部分类对应的UML类图如下：



2.2.2 注解处理器

注解编译器里面需要实现的内容主要包括以下两部分：

2.2.2.1 生成ContentProvider类

自动生成ContentProvider文件，并且将实现类作为ContentProvider的成员变量，在ContentProvider中的call()中调用实现类的对应方法。

如果多个实现类对应同一个ContentProvider，怎么区分是调用哪个实现类(成员变量)的什么方法呢？

NetMonitor mNetMonitor;

LogUpload mLogUpload;

解决方式：Bundle中将类的全限定名作为参数传递，用于具体什么实体类的区分。参数传递在下个部分的动态代理中处理。

2.2.2.2 注入

需要对PickingActivity里面的mNetMonitor做注入，默认使用Proxy.newProxyInstance()动态代理的方式来做，这样可以方便拦截相应的方法，以便在调用ContentProvider之前在Bundle里面传递一些我们需要用到的参数。

mNetMonitor = Proxy.newProxyInstance(classload ,interface ,invocationHandler);

说明：

具体调用到哪个ContentProvider就通过读取注解上面的authorities来区分；

注入的入口考虑通过调用IPC.inject(Object obj)的方式进行注入；

2.2.2.3 注解处理器的具体实现

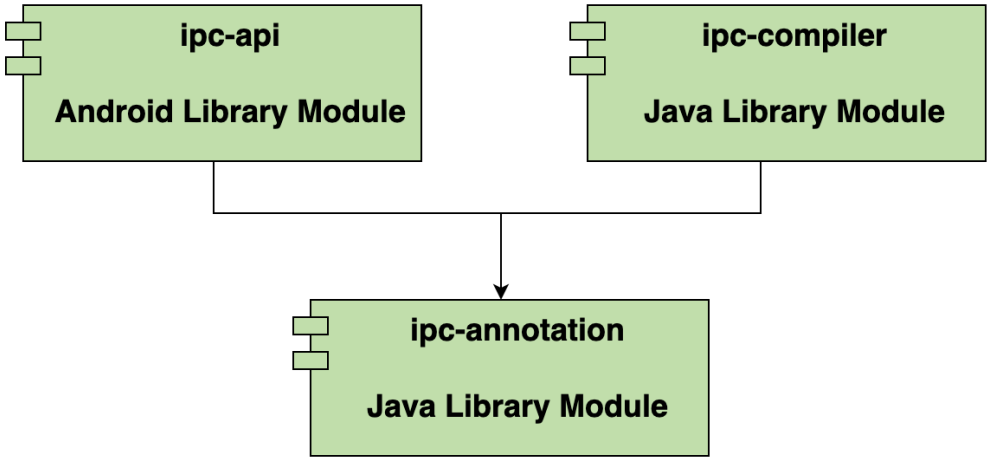
主要通过AnnotationProcessor+Javapoet来实现注解处理器相关的功能，具体分为三个子模块：

ipc-annotaion：该模块主要用于定义相关注解；

ipc-api：该模块主要用于实现框架内部逻辑、提供对外调用的api，对外的api主要包括：异步回调接口、实现注入的入口api;

ipc-compiler：该模块主要用于处理注解，自动生成相关代码，主要包括：生成ContentProvider类、对需要使用跨进程服务的成员变量完成注入；

module之间的依赖关系如下：



2.2.3 异步调用实现

ContentProvider.call()本身是同步的，因此同步调用比较好实现，现在需要解决异步调用的问题。针对异步，需要回调，但是Bundle中是不允许传递接口类型的，因此需要将回调定义为aidl文件。定义的回调接口如下：

```
public interface IPCCallback{
   void callback(Bundle bundle)
}
```

异步的实现方式：

在动态代理里面拦截的时候对方法参数进行判断，如果有IPCCallback这个参数则将IPCCallback.Stub通过bundle传递。

```
使用BundleCompat.putBinder(key,new IPCCallback.Stub() {
        @Override
        public void callback(Bundle remoteResult) {
            Ipccallback.callback(remoteResult);
})
```

2.2.4 注意事项

相同的authoties对应同一个contentProvider，process必须相同的。

不同的authorities对应不同的contentProvider，process可以相同，也可以不同。

生成的ContentProvider需要在清单文件中注册。

2.3 接口设计

对外提供的接口相关信息，如接口名称、协议、参数，及注意事项等（如未涉及则不需填写）

针对异步调用，定义的接口（aidl文件）如下：

```
public interface IPCCallback{
   void callback(Bundle bundle)
}
```

2.4 数据存储设计

数据库、表、文件等存储方案设计（如未涉及则不需填写）

2.5 监控方案设计

可包含业务监控如PV、点击、曝光、用户行为等，及质量监控如页面性能、错误、API质量等部分

2.6 部署方案设计

可包括部署相关准备工作如机器、资源申请等，部署顺序、目标节点，灰度和回滚方案等

3、风险评估

列出可能存在的风险及应对策略，如外部依赖、技术成熟度、关联影响、不可控因素、注意事项等

1.ContentProvider生成后怎么在清单文件中注册??

因为前期涉及到的跨进程服务比较少，对应的ContentProvider也比较少，暂时采用手动注册的方式；后期可以考虑通过gradle的方式主动注册。

2.同步调用时的方法返回参数只能是Bundle？

因为ContentProvider进行跨进程通信时，数据传输只支持Bundle，针对同步调用的方法返回参数，若不想使用Bundle，想定义为具体的类型，考虑通过强转的方式来实现。

其他

以上未包含的其他补充项说明