Gaea Pipeline V2.0 说明文档 (V1.3)

大数据计算组　　黄志博

2017 年 5 月 1 日

目录

1 概述

1.1 新版本说明

随着 Gaea 平台的应用越来越多，原流程（Gaea_Pipeline_standard_output_multi_beta.pl）变得越来越复杂，为了便于日后的流程维护和应用扩展，开发了此流程（python），定名为 GaeaPipelineV2.0.0 beta。新版本有如下特征：

- 采用模块化设计，便于 APP 的扩展，若增加新的分析模块只需按约定写好 APP 并存放到相应目录即可。
- hadoop 任务与单机任务之间能更好的衔接，hadoop 任务完成后单机任务会分样本投递任务，并分别考虑单个样本内任务依赖关系。
- 优化了用户参数配置文件，采用 cfg 文件格式，并简化参数配置，使参数配置更加简单。
- 优化了 Hadoop 任务运行错误检测方法，根据任务脚本标准错误输出信息的更多指标对任务执行状态进行判断。
- 使用新的任务重投方式，单样本分析时能更灵活的选择重跑步骤。

1.2 设计原则

1. **可重复**
   每个步骤中断后均可重复执行，不影响结果的一致性。某个步骤的执行不修改其他步骤的结果。
2. **依赖独立性**
   原则上应不依赖于其他步骤的目录结构和执行结果，单个步骤可独立运行。
3. **参数独立性**
   在 user.cfg 中有一个以 APP 名命名的参数列表，除通用参数外不应调用其他 APP 的参数。
4. **模块化**
   一个 APP 就是一个模块，即一个 Python 脚本，添加新的 APP 不需要改动流程。
5. **可扩展**
   - APP 可由用户自定义，将其放置到相应目录即可像其他 APP 一样使用。
   - sample.list 的解析方式也可由用户自定义，只需按固定的数据结构返回信息。

2 流程框架

2.1 流程框架图

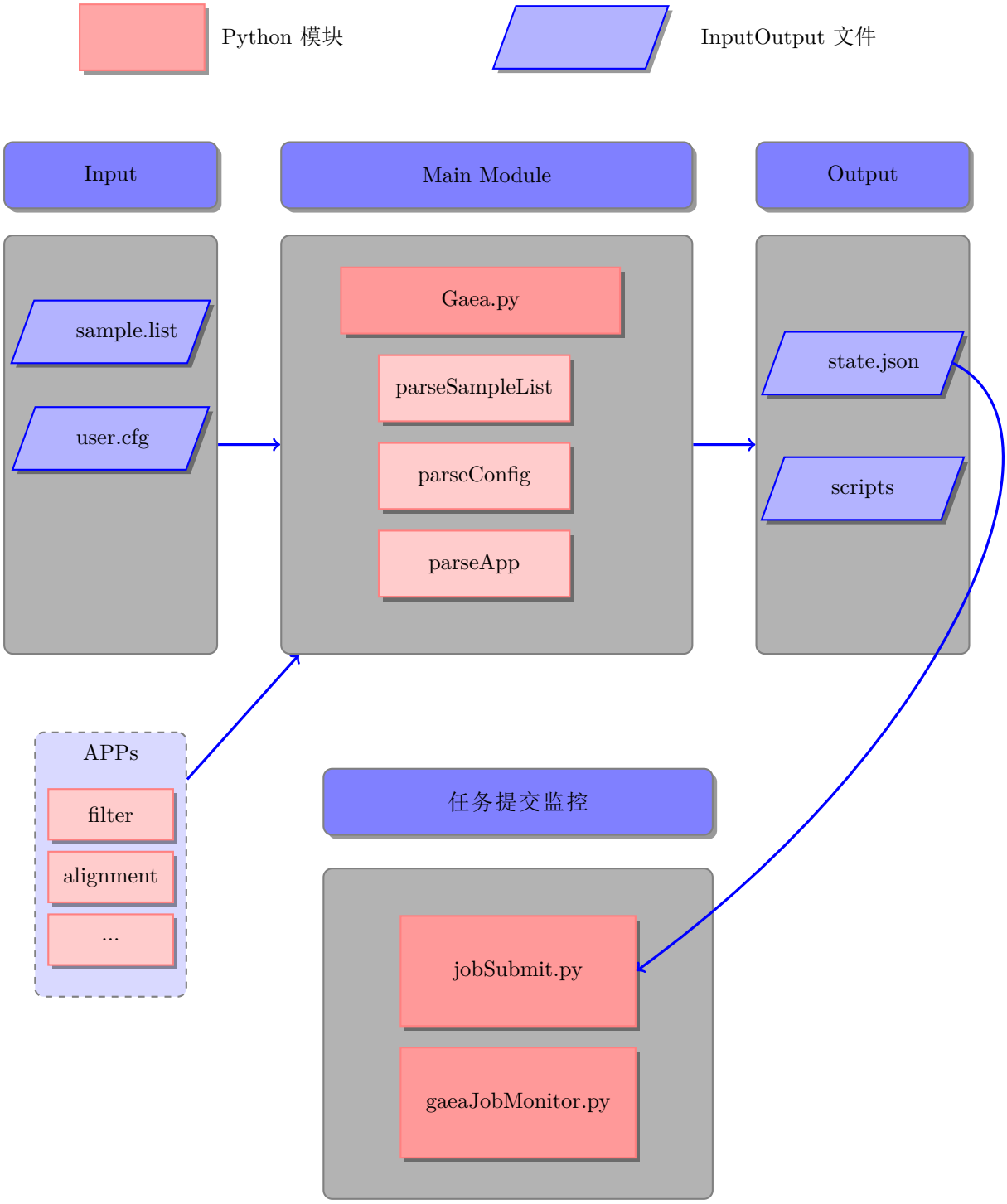
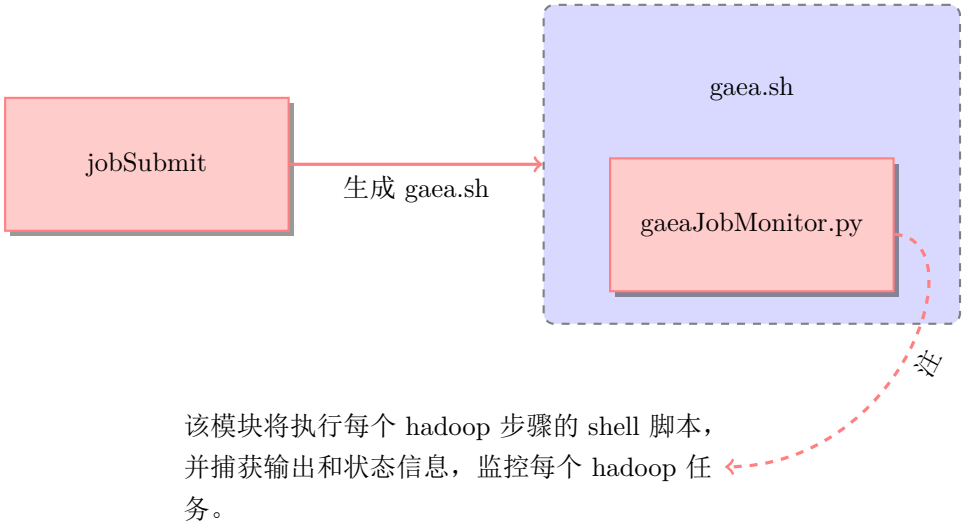


图 1: GaeaPipeline 流程框架图

2.2 任务提交与监控

jobSubmit.py 是根据 state.json 信息提交任务的主程序，对每个样本的 hadoop 任务会重新生成 gaea.sh 脚本，然后和 Standalone 任务的 shell 脚本统一提交调度。其中 gaea.sh 中通过调用 gaeaJobMonitor.py 来执行和监控每个 hadoop 任务。



3 自定义 APP 模块

3.1 简述

GaeaPipeline 的一个 APP 模块就是一个按约定书写的 python 脚本,将写好的 APP 放到 $GAEA_HOME/workflow 目录下或在 user.cfg 中指定存放 APP 的路径即可使用。使用时只需在 user.cfg 中设置 “analysis_flow” 即可(如图)。

3.2 书写约定

3.2.1 App 的输入输出

APP 的输入输出信息会存储到一个的字典类型的数据结构中[1]，APP 解析时也需根据依赖关系，从该数据结构中查找所依赖步骤的输出信息，作为输入。

3.2.2 模块命名

以一个名为 HelloWorld 的 APP 为例。按其运行平台，应将该模块命名为 “S_HelloWorld.py” 或 “H_HelloWorld.py”。“S” 指单机程序，“H” 指 Hadoop 程序。

3.2.3 主要方法与数据结构

[1]即程序执行后生成的 state.json 文件

1. **必要引用和继承**

```
from gaeautils.bundle import bundle
from gaeautils.workflow import Workflow

class APPname(Workflow):
    ...
```

2. INIT **属性**

   INIT 是定义默认参数的属性，其值将被存放到 self 中，会被 user.cfg 参数覆盖，用户要在 user.cfg 文件中设置的参数，需在 INIT 中设置一个默认值或空值，一些不常更改的参数在此设置也比较方便，如单机 APP 在此设置 INIT.APP.mem(内存资源) 等。

3. **参数调用**

   user.cfg，INIT 默认参数和以及继承自 Workflow 的信息，均会以字典的形式存储到 self 对象中，APP 可对该对象中的数值进行修改，但除非必要不要在此修改其他 APP 的参数。程序完成后，会在 workdir/scripts/state 目录下生成一个 state.json 文件[2]，保存了所有 self 对象中的信息。

4. impl **类**

   impl 是一个传入的类，包含了一些开发 APP 的常用方法。

   - impl.expath() 是扩展程序路径的方法，如果传入的程序路径不是绝对路径，则会根据名称在相应目录中查找并返回绝对路径。该方法有两个参数，第一个参数固定为 self.Path.prgDir[3]，第二个参数为程序名，程序名一般存储形式为 self.APP.prgParamName。
   - impl.mkdir() 是创建目录的方法，可有多个参数，返回值为所创建目录的绝对路径。原则上一个 APP 不能依赖其他 APP 创建的路径，所以要使用该方法，而不使用 os.path.join().
   - impl.write_shell() 该方法用于创建脚本。

[2]提交任务后也会生成 results.json 文件，该文件主要信息是 state.json 的一个子集，但若提交了单机任务，此处会保存 jobId 信息

[3]由 user.cfg 设置的存放 APP 所用程序的目录，多个路径由":" 分开，按顺序查找。

3.3 代码示例

模块 S_HelloWorld.py 代码:

```
# encoding: utf-8
from gaeautils.bundle import bundle
from gaeautils.workflow import Workflow

class HelloWorld(Workflow):
    " An example for APP develop. (samtools index). "

    #INIT 设置默认参数。
    INIT = bundle(HelloWorld=bundle())
    INIT.HelloWorld.program = "/home/huangzhibo/bin/samtools"  #若user.conf中设置了HelloWorld.program, 则此值会被覆盖
    INIT.HelloWorld.parameter = ''
    INIT.HelloWorld.mem = '2G'    #standalone 需要设置所需计算资源 (用于向集群提交任务)

    def run(self, impl, dependList):
        '''
        dependList是该步骤的依赖步骤列表, 如['S','HelloWorld','bamSort'], 则dependList==['bamSort']
        self.results是一个包装了的字典类型 (bundle, 可通过'.'取值), 其中存储了各步骤的输出信息, 如下
        self.results = \
        {
            "bamSort": {
                "output": {
                    "sample1": "/path/sample1.bam",
                    "sample2": "/path/sample2.bam"
                },
                "script": {
                    "sample1": "/path/sample1/bamSort.sh",
                    "sample2": "/path/sample2/bamSort.sh"
                }
            },
            ...
        }
        从self.results中获取bamSort步骤的输出信息: inputInfo = self.results.bamSort.output
        '''

        impl.log.info("step: HelloWorld!")
        inputInfo = self.results[dependList[0]].output

        #result 定义返回值, 将被赋值给 self.results.HelloWorld, 其中script必须设置用以提交任务, output如果不设置则该APP不能被依赖
        result = bundle(output=bundle(),script=bundle())

        #extend program path
        self.HelloWorld.program = impl.expath(self.Path.prgDir, self.HelloWorld.program)

        #script template  生成脚本, cmd是个列表, 每个值生成shell脚本的一行,${XXX}将被ParamDict中的值替换
        cmd = []
        cmd.append('%s index ${PARAM} ${INPUT}' % self.HelloWorld.program)
        cmd.append('echo "Hello World!"')

        for sampleName in inputInfo:
            scriptsdir = impl.mkdir(self.scriptsDir,'standalone',sampleName)

            ParamDict = {
                    "INPUT": inputInfo[sampleName],
                    "PARAM":self.HelloWorld.parameter
                }

            #write script
            scriptPath = \
            impl.write_shell(
```

```
                    name = 'HelloWorld',
                    scriptsdir = scriptsdir,
                    commands=cmd,
                    paramDict=ParamDict)

            #result
            result.output[sampleName] = inputInfo[sampleName]
            result.script[sampleName] = scriptPath
        return result
```

4 样本信息

1. **华大标准下机数据**

   华大下机数据文件中有详细的建库等信息，所以只需提供部分信息，每列信息以制表符隔开，每列信息见表 8。Gaea 会自动生成 RG，并且在 fq1 和 fq2 所在目录寻找 adapter 序列文件。

表 1: 华大标准下机数据配置表

列号	1	2	3	4	5	6	7	8
名称	样品名	性别	家系	类型	建库 pool 号	数据基本路径	上机 pool 号	样品的子文库号（index）

2. **独立数据**

   一个样品往往包含多个 lane 信息，其中每个 lane 需要配置如下格式，其中需要配置的属性包括：样品名，fq1，fq2，rg（reads group）；可选择配置属性为：gender（性别），family（家系），pool，type（类型），libname（文库号），adp1，adp2。

```
>sample_1
属性=值
>
>sample_2
属性=值
>
……
```

3. BAM **数据**

   如果已做完比对相关分析并得到 BAM 文件，可以用 Gaea 继续变异检测和注释。此时，只需要提供以制表符分隔的两列信息，两列信息分别是：样品名称和 BAM 文件路径或 BAM 文件所在文件夹的路径。需要注意的是，BAM 文件所在文件夹的路径中只能有 BAM 文件。

4. VCF **数据**

   如果已经得到 VCF 文件，只需做后续分析，可以只配置 VCF 文件。此时，只需要提供以制表符分隔的两列信息，两列信息分别是：样品名称和 VCF 文件路径。

5 参数配置

5.1 用户参数配置文件格式

用户参数配置文件（user.cfg）使用 cfg 格式（json 格式不再使用）。cfg 文件一种是基于 INI 文件的配置文件，用 “#” 号作为注释标记（不能用分号 “;”），用中括号标记小节（section），可进行小节的嵌套，区分 key 值大小写，且支持多种数据类型，详见格式说明。示例如下

```
# The First Section
[section1] # first section
keyword1 = value1
    [[sub-section]]
        keyword2 = value2

# The Second Section
[section2] # second section
keyword2 = value2
keyword3 = """ A multi line value
on several
lines"""

[section3]
keyword1 = value1, value2, value3
keyword2 = value1,
```

5.2 extend 配置文件

extend 配置文件[4]中提供了扩展 user.cfg 与具体应用无关的通用参数 hadoop 和 ref 的完整信息，目的在于简化用户配置，用户不需要再指定详细的集群参数等。简化后用户只需如下指定所使用集群：

```
[hadoop]
    cluster = cluster35
```

user.cfg 有较高的优先级，用户也可以在 user.cfg 的 hadoop 小节下指定完整的信息，这样则会覆盖 extend 中的设置见附录 2。

5.3 主要参数项说明

用户需要提供的参数配置[5]，主要有以下几个要素：

- analysis_flow
  设置分析步骤和依赖关系，详见下一小节。
- file
  需要经常更换，或多个 APP 通用的文件（如 bed 文件），需在此设置。另外，此参数项下的参数可被其它 APP 参数以一定的格式引用，见附录 1中 genotype 等 APP 的参数。
- hadoop
  Hadoop 集群信息。该参数在流程的 extend 配置文件中有相应信息的映射，用户只需简单指定所用集群，文件系统等信息。

[4]该文件位于 $GAEA_HOME/config/目录下, 如深圳集群: /ifs4/ISDC_BD/GaeaProject/software/GaeaPipeline/config/extend.cfg
[5]用户配置文件相关 example 位于 $GAEA_HOME/config/目录下

- ref
  参考序列信息。其具体路径同样保存在 extend 配置文件中，如果用户设置的不是绝对路径，将通过关键字在 extend.cfg 文件查找。

- Path
  指定 APP 程序的搜索路径和 APP 模块的额外存放路径。

- APP 参数
  存放各个 APP 的程序和参数，参数项名称为各个 APP 名。其中有两个功能较特殊的 APP：

  – init
    init 用于从接收 sample 信息到启动正式分析流程的过度，必要时会对数据进行解压，需在此参数项中设置 qualitySystem 等。此步骤始终是用户设置的起始步骤的依赖步骤，即事实上的起始步骤。

  – self_defined
    用户可使用此参数项自定义 APP。这也是一个名为 self_defined 的 APP 模块，通过接收参数生成脚本，但对于较复杂的步骤建议直接实现 APP 模块。

  – 应用与参数要对应
    在已有的配置文件中修改时，若改变程序，要注意调整对应的参数。如: alignment 在使用 aln 和 mem 两个子工具时参数是不同的，不能只修改工具名而不改参数。

5.4 步骤依赖关系

流程中每个步骤的依赖关系是一个有向无环图，其示意图如下：

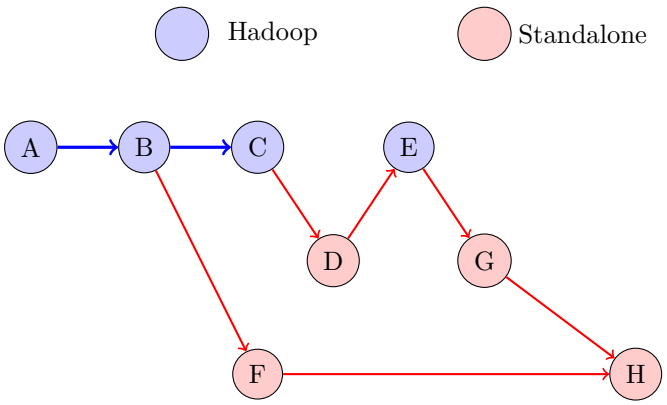


图 2: APP 依赖关系示意图

流程的分析步骤需要在用户配置文件中设置。下图是配置文件中 “analysis_flow” 参数的设置形式，其中只有第一行的起始步骤没有依赖。另外，对执行步骤只依赖于前一步骤，且属于同一平台的步骤列表可以合并, 步骤间用 “|” 分割，代表前一步骤的输出是后一步骤的输入。

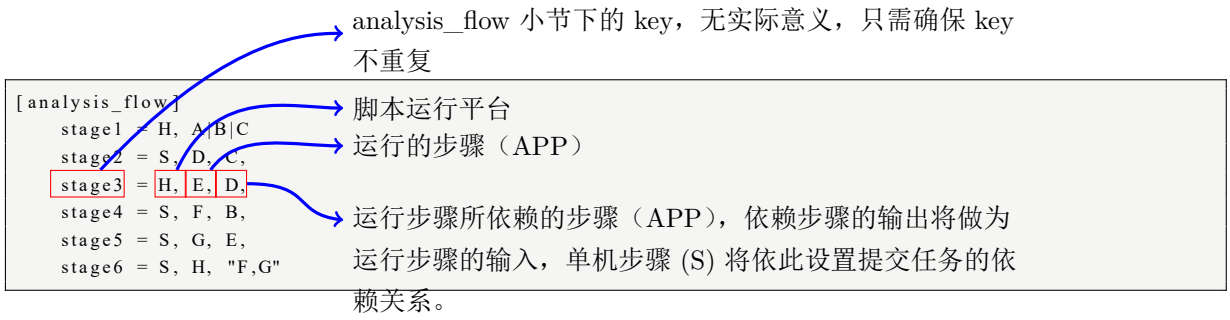


图 3: analysis_flow 参数

5.5 用参数自定义 APP

在用户配置文件中可以仅通过参数实现自定义 APP，参数 APP 的定义需在 self_defined 下，以嵌入的子小节名作为 APP 名，需定义的 key 有以下几项：

- mem
  设置所需计算资源，若是 hadoop 程序则不需要此项
- output
  定义输出路径，有 ${WORKDIR} 和 ${SAMPLE} 两个变量可用，${WORKDIR} 指工作路径，${SAMPLE} 指样本名。关于该变量的值可参考 APP 代码示例。
- command
  command 为生成脚本的命令模版，定义在两个三引号内。可用变量为 file 参数项中的所有参数，以及 ${INPUT} 和 ${OUTPUT}。另外，变量 ${checkstatus} 是检查上个命令执行状态信息的语句，需单独成一行。

参数定义 APP 示例：

```
[self_defined]
    [[bamdst]]
    mem='4G'
    output='${WORKDIR}/Coverage/${SAMPLE}'
    command = '''
        bamdst -p ${region} -o ${OUTPUT} ${INPUT}
        ${checkstatus}
        echo "done"
    '''
    [[bamindex]]
    mem = '2G'
    command = '''
        if [ -e ${INPUT}.bai ]\nthen
        \texit 0
        fi
        samtools index ${INPUT}
        ${checkstatus}
        echo "done"
    '''
```

6 使用说明

6.1 程序路径及示例

- 程序路径

  天津 /THL4/home/bgi_gaea/software/GaeaPipeline/RunGaea

  深圳 /ifs4/ISDC_BD/GaeaProject/software/GaeaPipeline/RunGaea

- 相关示例在深圳集群 /ifs4/ISDC_BD/GaeaProject/software/GaeaPipeline/example 目录下

6.2 生成并运行脚本

6.2.1 准备

6.2.2 Usage

RunGaea 是用 shell 对 Gaea.py 进行的包装，不用配置环境变量[6]，其功能是生成各分析步骤的 shell 脚本。其中，WORKDIR, SAMPLELIST 和 CONFIG 为必须指定的参数。help 信息如下：

```
usage: Gaea.py [-h] [-w WORKDIR] -d DIRHDFS -s SAMPLELIST [-m MODE]
               [-c CONFIG] [-j PROJECTID] [-n] [-q QUEUE] [-p PARTITION]
               [-t {write,local,submit}] [-V]

USAGE

optional arguments:
  -h, --help            show this help message and exit
  -w WORKDIR, --workdir WORKDIR
                        working directory of the workflow. [default: '.']
  -d DIRHDFS, --dirHDFS DIRHDFS
                        hdfs work directory(When you use it in TH-1A,you must
                        set it according lustre directory structure)
  -s SAMPLELIST, --sample SAMPLELIST
                        fq list or bam list or independent data list.
  -m MODE, --mode MODE  samplelist mode:1,standard BGI data;2,independent raw
                        data;3,bam data; 4,vcf data. [default: 1]
  -c CONFIG, --config CONFIG
                        user config file.
  -j PROJECTID, --projectId PROJECTID
                        project ID.
  -n, --multi           multiSample in one batch to run. [default: False]
  -q QUEUE, --queue QUEUE
                        the queue of the job. [default: None]
  -p PARTITION, --partition PARTITION
                        the job partition. [default: None]
  -t {write,local,submit}, --type {write,local,submit}
                        1.write: just write run scripts; 2.local: run tasks on
                        one local node; 3:submit: submit tasks to SGE
                        [default: write]
  -V, --version         show program's version number and exit
```

6.2.3 投递任务

用 RunGaea 生成脚本后，需要到工作目录的 scripts 目录下，执行 run.sh 提交任务。

- ./run.sh
  本地运行 hadoop 的任务调度程序，如在 hadoop 的 master 节点直接执行 run.sh。
- ./run.sh submit
  将 hadoop 任务提交到 gaea.q 队列[7]，单机任务提交到参数指定的队列。

[6]若使用 Gaea.py 需要设置环境变量，具体设置见 GaeaPipeline/config/envset.sh
[7]该队列为深圳 hadoop35(cluster35) 集群专用

6.2.4 项目目录结构

每个分析项目会形成较为固定的目录结构。scrpts 目录下是流程生成的脚本和 log 等其它信息，其中 gaea 目录下是 hadoop 脚本，standalone 目录下是单机脚本，每个脚本执行后产生的标准错误和标准输出也会存在于与脚本相同的位置。

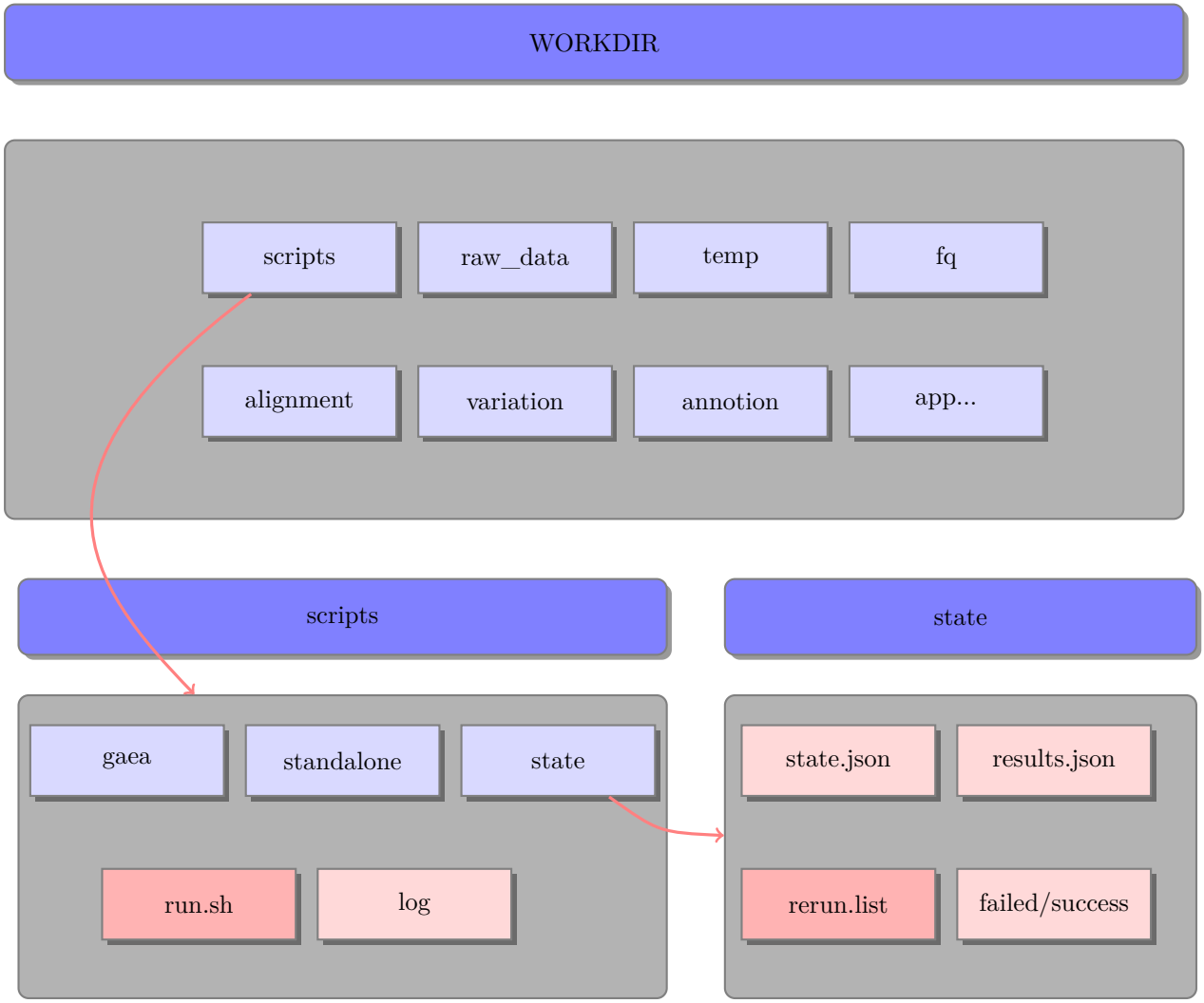


图 4: 项目目录结构图

6.3 查看错误信息并重跑

任务执行结束后，用户需要查看 scripts 目录下的 log 文件 (如有错误在 state 目录下会生成 failed 文件)，如果有失败任务需要通过查看相应步骤的标准错误输出文件找到原因，例如发现是参数配置错误，则需修正后重新生成脚本，然后编辑 scripts/state 目录下的 rerun.list，删掉不需执行的任务，然后提交任务即可。

- rerun.list 文件，是一个包含样本名和运行步骤两列信息的文本文件，单样分析时能区分不同样本的步骤。程序将根据此文件提交任务，若要重跑只需删除不需要跑的步骤，然后重新提交即可。其格式如下：

```
1 sample1    APP1,APP2,APP3
2 sample2    APP1,APP2,APP3
```

附录

I 用户参数配置文件

深圳集群：

```
[analysis_flow]
    stage1 = H, filter|alignment|rmdup|realignment|baserecal|genotype|mergeVariant
    stage2 = H, bamqc, baserecal
    stage3 = H, bamSort, baserecal
    stage4 = S, BGICGAnnotation, bamSort
    stage5 = S, bamindex,bamSort
    stage6 = S, bamdst,bamSort
[file]
    annoProtoclConfig = ""
    cnvRegions = ""
    region = ""
    cnvAnnoConfig = ""
    bamRelationList = ""
    regionVariation = ""
    newCnvConfig = ""
    dbsnp = ""
[ref]
    [[normal]] #如果不是绝对路径，则根据 extend.cfg 获取绝对路径
        ref = hg19
        bwaIndex = hg19
        gaeaIndex = hg19
        soap2Index = ""
        gaeaAlignIndex = ""
[hadoop]
    cluster = cluster35
[Path]
    prgDir = /ifs4/ISDC_BD/GaeaProject/software
    appDir = ""
    modeDir = ""
[init]
    gzUploader = GzUpload.jar
    multiUploader = multi_uploader.pl
    bgzip = bgzip
    samtools = samtools
    ## 如果不设置质量值则会自动检测
    #    qualitySystem = 0
[filter]
    #不需要设置 -Q 参数（qualitySystem）
    program = /ifs4/ISDC_BD/lishengkang/work/SoapNuke1.5.2_test/fastqQC/GaeaFastqQC.jar
    parameter = -lowQual 11 -qualRate 0.1
[alignment]
    # aln / mem
    bwaSubTool = aln
    parameter = -i 10 -q 10
    program = bwa-0.7.10-streaming
    streamingJar = Streaming_fq.jar
    indexer = /ifs4/ISDC_BD/zhangyong2/work/bwa/bwa-0.7.10/bwa
[rmdup]
    program = /ifs4/ISDC_BD/lishengkang/project/liangxinming/20151103/GaeaDuplicateMarker.jar
    parameter = ""
[realignment]
    program = GaeaRealigner.jar
    parameter = ""
[baserecal]
    #参数bqsr_param可以用 ${} 的格式调用 file section下的变量
    bqsr_param = "-knownSites file://${dbsnp}"
    printreads = GaeaPrintReads.jar
    bqsr = GaeaBqRecalibrator.jar
```

```
    printreads_param = ""
[bamSort]
    picard = /ifs4/ISDC_BD/GaeaProject/lib/picard.x.1.jar
    program = hadoop-bam.jar
[genotype]
    program = GaeaGenotyper.jar
    parameter = "-genotype_likelihoods_model BOTH  -stand_call_conf 30.0 -stand_emit_conf 10.0 -
        out_mode EMIT_ALL_CONFIDENT_SITES -dbsnp file://${dbsnp}"
[cgConversion]
    program = GaeaVoyagerConverter.jar
[mergeVariant]
    filter_param = '-snp "QD<2.0 || MQ<40.0 || FS>60.0 || HaplotypeScore>13.0 || MQRankSum<-12.5 ||
        ReadPosRankSum<-8.0" -indel "ReadPosRankSum<-20.0 ||InbreedingCoeff<-0.8 || FS>200.0"'
    merge = vcfmerge.pl
    filter = ""
    split = Medicine/vcf_sample_split.pl
    sort = vcf-sort
[BGICGAnnotation]
    departAnnos_param = ""
    excelReport = /ifs5/ST_TRANS_CARDIO/PUB/analysis_pipelines/BGICG_Annotation/bin/excel_report_v2.
        pl
    bgicgAnno_param = -n 5 -b 500 -q -t vcf
    departAnnos = /ifs5/ST_TRANS_CARDIO/PUB/analysis_pipelines/BGICG_Annotation/bin/depart_annos_v2.
        pl
    bgicgAnno = /ifs5/ST_TRANS_CARDIO/PUB/analysis_pipelines/BGICG_Annotation/bin/bgicg_anno.pl
[bamqc]
    program = GaeaBamQC.jar
    parameter = -M
    exonDepthSort = Medicine/exon_sort.pl
[cnv]
    program = Medicine/CNV-gaea.pl
    parameter = -run_type pool
[graph]
    totalCoverageDepth = Medicine/total_coverage_depth.pl
    gaeaInsertsize = Medicine/gaea-insertsize.R
    uncoverAnno = Medicine/uncover_anno_v2.0.pl
    exonGraph = Medicine/exon_graph.pl
[self_defined]
    [[bamdst]]
    mem='4G'
    output='${WORKDIR}/Coverage/${SAMPLE}'
    command = '''
        bamdst -p ${region} -o ${OUTPUT} ${INPUT}
        ${checkstatus}
        echo "done"
    '''
    [[bamindex]]
    mem = '2G'
    command = '''
        if [ -e ${INPUT}.bai ]\nthen
        \texit 0
        fi
        samtools index ${INPUT}
        ${checkstatus}
        echo "done"
    '''
```

II extend 配置文件

深圳集群:

```
[hadoop]
    [[cluster35]]
    bin = /ifs4/ISDC_BD/HadoopCluster/compute-298d3fd5-9d16-4138-bb50-fc56daae8aaa/hadoop-0.20.2-cdh3u6/bin/hadoop
    streamingJar = /ifs4/ISDC_BD/HadoopCluster/compute-298d3fd5-9d16-4138-bb50-fc56daae8aaa/hadoop-0.20.2-cdh3u6/contrib/streaming/hadoop-streaming-0.20.2-cdh3u6.jar
    mapper_num = 104
    reducer_num = 104
    ishadoop2 = false
    is_at_TH = false
    [[cluster50]]
    bin =/usr/bin/hadoop
    streamingJar = /opt/cloudera/parcels/CDH-5.6.0-1.cdh5.6.0.p0.45/jars/hadoop-streaming-2.6.0-cdh5.6.0.jar
    mapper_num = 190
    reducer_num = 190
    ishadoop2 = true
    is_at_TH = false
[ref]
    [[normal]]
    [[[hg19]]]
        soap2Index = ""
        ref = /ifs4/ISDC_BD/GaeaProject/reference/hg19/hg19.fasta
        gaeaAlignIndex = ""
        bwaIndex = /ifs4/ISDC_BD/GaeaProject/reference/hg19/hg19.fasta
        gaeaIndex = /ifs4/ISDC_BD/GaeaProject/reference/hg19/GaeaIndex/ref_bn.list
    [[male]]
    [[[hg19]]]
        soap2Index = ""
        ref = /ifs4/ISDC_BD/GaeaProject/reference/bgi_medicine/male/hg19_chM_male_mask.fa
        gaeaAlignIndex = ""
        bwaIndex = /ifs4/ISDC_BD/GaeaProject/reference/bgi_medicine/male/hg19_chM_male_mask.fa
        gaeaIndex = /ifs4/ISDC_BD/GaeaProject/reference/bgi_medicine/male/GaeaIndex/ref_bn.list
    [[female]]
    [[[hg19]]]
        soap2Index = ""
        ref = /ifs4/ISDC_BD/GaeaProject/reference/bgi_medicine/female/hg19_chM_female.fa
        gaeaAlignIndex = ""
        bwaIndex = ifs4/ISDC_BD/GaeaProject/reference/bgi_medicine/female/hg19_chM_female.fa
        gaeaIndex = /ifs4/ISDC_BD/GaeaProject/reference/bgi_medicine/female/GaeaIndex/ref_bn.list
```