批量计算

快速开始

快速开始

准备工作

1. 创建阿里云账号

如果您还没有阿里云账号，请登陆阿里云官网，点击右上角“免费注册”创建阿里云账号。

2. 开通BatchCompute

使用注册成功的阿里云账号登陆，点菜单中“产品”，在“弹性计算”中找到批量计算（BatchCompute）进入 产品主页，开通 BatchCompute 服务。

3. 获取 AccessKeyId 和 AccessKeySecret

进入 AK控制台 获取 AccessKeySecret:

找到启用状态的 AccessKeyId 点击显示，即可获取对应的 AccessKeySecret。

如果没有可用的 AccessKeyId 可点击右上角的”创建 Access Key”按钮，创建成功后,系统会生成一对 AccessKeyId 和 AccessKeySecret。

4. 理解基本概念

为更好的继续下面的操作，首先简单理解BatchCompute中的几个 核心概念。

5. 关于OSS

批量计算服务默认使用 OSS 作为持久化存储， 所以很有必要先花 10 分钟时间了解 OSS。

如果您已经对OSS有所了解，请忽略。

6. 选择合适的工具

批量计算服务提供基于 HTTP 的 API，在此之上我们还封装了一些工具，您可以使用这些工具提交作业，和管理作业等。

使用命令行工具：适合初级用户。命令行工具实现了自动化的程序打包上传和作业提交，用户无需了解批量计算的 JSON 格式即可管理作业和集群，强烈推荐新手从 命令行工具快速开始。

使用控制台：适合初、中级用户，使用网页可视化界面管理作业和集群，使用控制台需要对批量计算的程序包部署逻辑以及 JSON 格式有一定的了解，参考 控制台快速开始示例 。

使用SDK：适合高级用户，需要编程基础。官方提供 java 版和 python 版的 sdk，参考 Java 快速开始示例 和 Python 快速开始示例

使用云渲染管理系统：专门为渲染场景定制的 web 管理系统。

控制台快速开始

介绍如何使用控制台来提交一个作业，目的是统计一个日志文件中 **INFO**、**WARN**、**ERROR**、**DEBUG** 出现的次数。

步骤预览

1. 作业准备
    - 上传数据文件到 OSS
    - 上传任务程序到 OSS
2. 使用控制台提交作业
3. 查看作业状态
4. 查看结果

1. 作业准备

本作业是统计一个日志文件中 **INFO**、**WARN**、**ERROR**、**DEBUG** 出现的次数。

该作业包含3个任务: split、 count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中 **INFO**、**WARN**、**ERROR**、**DEBUG** 出现的次数(count 任务需要配置 InstanceCount 为3，表示同时启动 3 个 count 任务)。
- merge 任务会把 count 的结果统一合并起来。

DAG图例

上传数据文件到OSS

下载本例子所需的数据: log-count-data.txt

将 log-count-data.txt 上传到:

```
oss://your-bucket/log-count/log-count-data.txt
```

- your-bucket 表示您自己创建的 bucket，本例假设 region 为: cn-shenzhen。
- 更多关于如何上传到 OSS，请参考 OSS 文件上传 以及 常用 OSS 工具 。

上传任务程序到OSS

本例的作业程序是使用 python 编写的， 下载本例所需的程序: log-count.tar.gz

本例不需要改动示例代码。直接将 log-count.tar.gz 上传到 oss，如上传到：

```
oss://your-bucket/log-count/log-count.tar.gz。
```

如何上传前面已经讲过。

- BatchCompute 只支持以 tar.gz 为后缀的压缩包, 请注意务必用以上方式(gzip)打包, 否则将会无法解析。
- 如果你要修改代码，可以解压后修改，然后要用下面的方法打包：

命令如下:

```
> cd log-count  #进入目录
> tar -czf log-count.tar.gz * #打包，将所有这个目录下的文件打包到 log-count.tar.gz
```

可以运行这条命令查看压缩包内容：

```
$ tar -tvf log-count.tar.gz
```

可以看到以下列表:

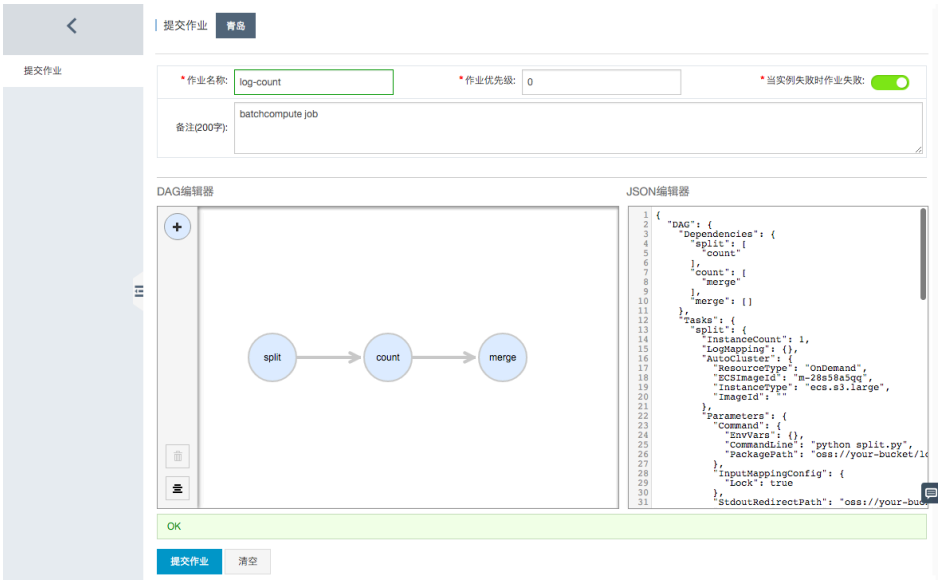
```
conf.py
```

```
count.py
merge.py
split.py
```

2. 使用控制台提交作业

登录 BatchCompute 控制台 。

单击 **作业列表** > **提交作业** 进行作业提交。请选择合适的 Region（该 region 需要和前面上传数据的OSS的 bucket 的 region 一致）。



如上图所示，首先填写作业名称、作业优先级等基本信息，接下来填写作业的详细描述，有两种方法：

- **DAG编辑器**：DAG编辑器可以用图形化的方式来描述作业 (Job) 包含的任务 (Task) 以及依赖关系。拖动编辑器左上角的“+”来添加 Task，拖动 Task上的箭头来描述依赖关系。单击 Task，可以为每个 Task 设置参数。

  JSON编辑器：也可以直接使用 JSON 的方式描述作业 (Job) 包含的任务 (Task) 以及依赖关系，关于作业的 JSON 描述及其参数说明，可参考作业 (Job) 相关的API使用文档。

  我们这里直接采用下面已经准备好的JSON描述粘贴到 **JSON编辑器**中就可以完成作业描述。

```
{
"DAG": {
"Dependencies": {
"split": [
"count"
],
"count": [
```

```
"merge"
],
"merge": []
},
"Tasks": {
"split": {
"InstanceCount": 1,
"LogMapping": {},
"AutoCluster": {
"Configs": {
"Networks": {
"VPC": {
"CidrBlock": "192.168.0.0/16"
}
}
},
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1ne.large",
"ImageId": "img-ubuntu-vpc"
},
"Parameters": {
"Command": {
"EnvVars": {},
"CommandLine": "python split.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"InputMappingConfig": {
"Lock": true
},
"StdoutRedirectPath": "oss://your-bucket/log-count/logs/",
"StderrRedirectPath": "oss://your-bucket/log-count/logs/"
},
"InputMapping": {
"oss://your-bucket/log-count/": "/home/input/"
},
"OutputMapping": {
"/home/output/": "oss://your-bucket/log-count/"
},
"MaxRetryCount": 0,
"Timeout": 21600,
"ClusterId": ""
},
"merge": {
"InstanceCount": 1,
"LogMapping": {},
"AutoCluster": {
"Configs": {
"Networks": {
"VPC": {
"CidrBlock": "192.168.0.0/16"
}
}
},
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1ne.large",
"ImageId": "img-ubuntu-vpc"
```

```
},
"Parameters": {
"Command": {
"EnvVars": {},
"CommandLine": "python merge.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"InputMappingConfig": {
"Lock": true
},
"StdoutRedirectPath": "oss://your-bucket/log-count/logs/",
"StderrRedirectPath": "oss://your-bucket/log-count/logs/"
},
"InputMapping": {
"oss://your-bucket/log-count/": "/home/input/"
},
"OutputMapping": {
"/home/output/": "oss://your-bucket/log-count/"
},
"MaxRetryCount": 0,
"Timeout": 21600,
"ClusterId": ""
},
"count": {
"InstanceCount": 3,
"LogMapping": {},
"AutoCluster": {
"Configs": {
"Networks": {
"VPC": {
"CidrBlock": "192.168.0.0/16"
}
}
},
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1ne.large",
"ImageId": "img-ubuntu-vpc"
},
"Parameters": {
"Command": {
"EnvVars": {},
"CommandLine": "python count.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"InputMappingConfig": {
"Lock": true
},
"StdoutRedirectPath": "oss://your-bucket/log-count/logs/",
"StderrRedirectPath": "oss://your-bucket/log-count/logs/"
},
"InputMapping": {
"oss://your-bucket/log-count/": "/home/input/"
},
"OutputMapping": {
"/home/output/": "oss://your-bucket/log-count/"
},
```

```
"MaxRetryCount": 0,
"Timeout": 21600,
"ClusterId": ""
}
}
},
"Description": "batchcompute job",
"Priority": 0,
"JobFailOnInstanceFail": true,
"Type": "DAG",
"Name": "log-count"
}
```

上述 JSON 描述的作业要正常运行，您需要根据自己的实际情况，对描述中一些配置项做一些适配修改，包括：

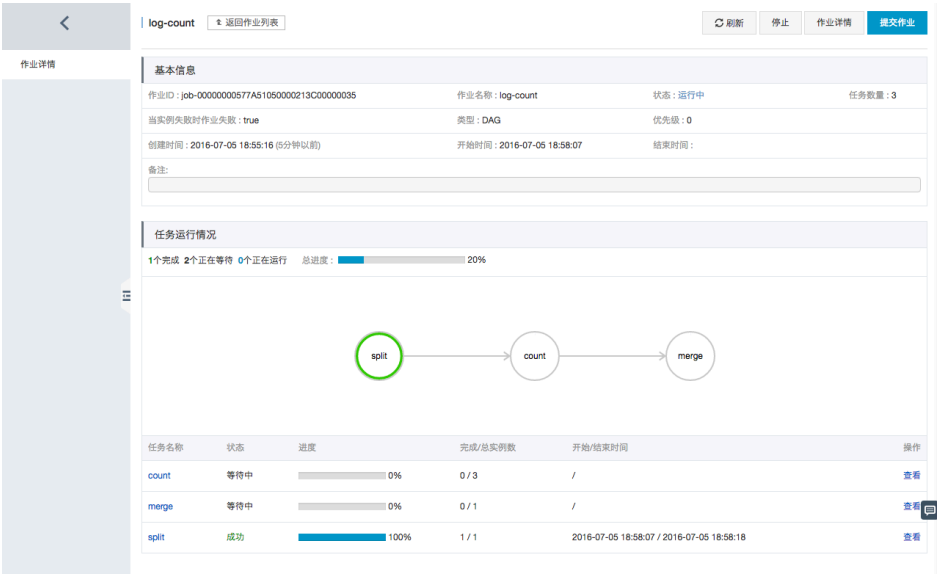
- 实例类型："InstanceType": "ecs.sn1ne.large"。用户实际可用的实例类型，可以单击**DAG编辑器**中的单个 Task，通过下拉选择框来选择。
- 程序包的OSS路径："PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"，填写为实际的程序包路径。
- Stdout日志的OSS路径：oss://your-bucket/log-count/logs/，填写为实际的日志路径（需要在OSS上提前创建好）。
- Stderr日志的OSS路径：oss://your-bucket/log-count/logs/，填写为实际的日志路径（需要在OSS上提前创建好）。
- 输入数据的映射路径："oss://your-bucket/log-count/": "/home/input/"，填写为输入数据的实际路径。

  输出数据的映射路径："/home/output/": "oss://your-bucket/log-count/"，填写为输出数据的实际路径。

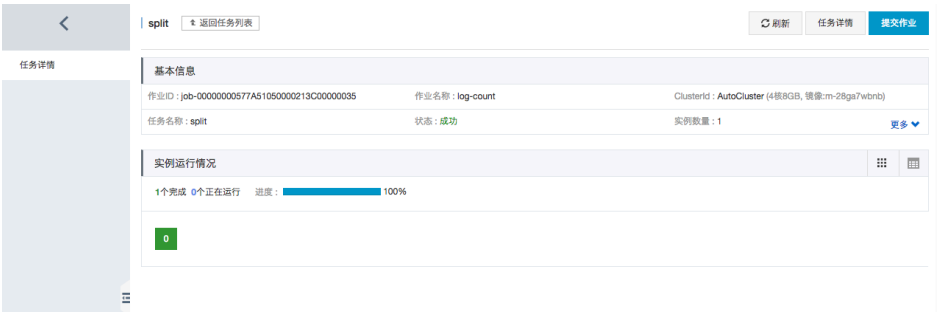
  确定各个参数及路径填写正确后，点击左下角的**提交作业**并确认，就完成了作业提交。

3. 查看作业状态

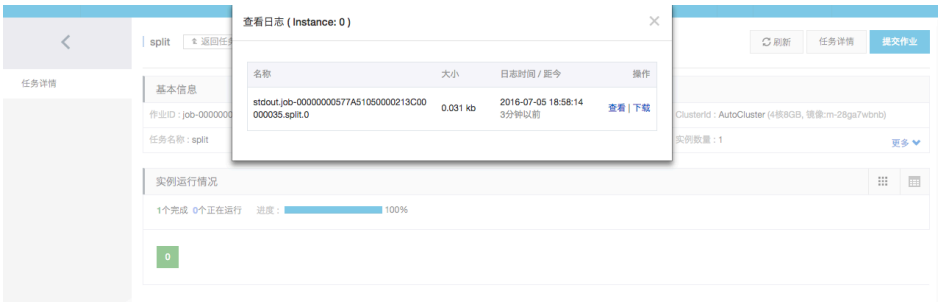
单击作业列表中最新提交的 log-count 作业，可以查看详情：



单击任务名称 split，可以查看任务详情：



单击绿色方块，可以查看实例的日志:



4. 查看结果

您可以登录 OSS 控制台 查看 your-bucket 这个 bucket 下面的这个文件：/log-count/merge_result.json。

内容应该如下：

```
{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}
```

限制说明

BatchCompute 的资源池在用户间共享，申请集群资源时可能可分配资源数会小于用户申请资源数，提交计算任务时也可能需要排队等待。

BatchCompute 的计算节点 VM 没有公网 IP，不支持直接访问公网地址。如果需要访问公网，可以在用户 VPC 内创建节点，并配置 NAT 网关来实现。

其他范例

命令行快速开始1

如果您还没开通批量计算服务，请先 开通。

步骤

- 命令行工具安装和配置
- 作业准备
    - 上传数据文件到 OSS
    - 准备任务程序
- 提交作业
- 查看作业运行状态及运行结果

1. 命令行工具安装和配置

命令行工具安装和配置

2. 作业准备

本作业的目的是求和，将 input.txt 中的数字全部加起来，求和后写入 output.txt。

由于计算比较简单本作业只需 1 个任务。

本例将 OSS 的目录挂载为 VM 本地目录，使用文件方式操作。

(1) 上传数据文件到OSS

先自行创建 input.txt。

input.txt的内容(确保一行一个数字):

```
2
40
51
```

将 input.txt 上传到:

```
bcs o up input.txt oss://your-bucket/sum/inputs/

# 上传完成后check
bcs o ls oss://your-bucket/sum/inputs/
```

- your-bucket 表示您自己创建的 bucket，本例子假设 region 为: cn-shenzhen。
- bcs o 命令提供几个 OSS 常用的功能，使用 bcs o -h 可以查看帮助，测试少量数据时使用很方便，但上传下载大量数据时不建议使用（没有实现多线程，上传下载慢。更多关于如何上传到 OSS，请参考 常用 OSS 工具。

(2) 准备任务程序

sum.sh 内容：

```
#!/bin/bash
t=0
while read LINE
do
t=$(($t+${LINE}))
done < /home/inputs/input.txt
echo $t
echo $t > /home/outputs/output.txt
```

注意在上一个步骤里我们把输入文件 input.txt 上传到了oss://your-bucket/sum/inputs/，在以上程序中 input.txt 是从虚拟机的/home/inputs/目录中读取，这是通过批量计算中对 OSS 的挂载功能实现的，具体配置将在下一步骤“提交作业”中解释。

在这个示例程序里，我们通过 bash 脚本完成了求和的功能。您也可以在脚本里执行任何其他应用程序，如何把应用程序部署到批量计算环境请参考 自定义镜像 和 使用 Docker 。

3. 提交作业

在 sum.sh 所在目录运行下面的命令来提交作业：

```
bcs sub "sh sum.sh" -p sum.sh -r oss://your-bucket/sum/inputs/:/home/inputs/ -w oss://your-bucket/sum/outputs/:/home/outputs/
```

> 这里使用 默认镜像和默认实例类型 。
>
> -r 表示只读挂载，将 OSS 目录oss://your-bucket/sum/inputs/只读挂载到 VM 本地目录 /home/inputs/，程序可以通过/home/inputs/路径来访问 input.txt 文件。
>
> -w 表示可写挂载，将 OSS 目录oss://your-bucket/sum/outputs/挂载为 VM 本地目录 /home/outputs/，写入本地目录/home/outputs/下的文件 output.txt，会在程序运行完后，被自动上传到 OSS 的对应目录。

4. 查看作业运行状态及运行结果

```
bcs j  # 获取作业列表, 每次获取作业列表后都会将列表缓存下来，一般第一个即是你刚才提交的作业
bcs j 1 # 查看缓存中第一个作业的详情
bcs log 1 # 查看缓存中第一个作业日志
```

可以使用以下命令查看结果：

```
bcs o cat oss://your-bucket/sum/outputs/output.txt
```

命令行快速开始2

本文档将介绍如何使用命令行工具来提交一个作业，目的是统计一个日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数。

步骤

- 命令行工具安装和配置
- 作业准备
    - 上传数据文件到 OSS
    - 准备任务程序
- 提交作业
- 查看作业运行状态
- 查看运行结果

1. 命令行工具安装和配置

命令行工具安装和配置

2. 作业准备

目的：统计一个日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数。

该作业包含3个任务: split, count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数 (count 任务需要配置 InstanceCount 为3，表示同时启动 3 个 count 任务)。
- merge 任务会把 count 的结果合并起来。

DAG图例:

(1) 上传数据文件到OSS

下载本例子所需的数据: log-count-data.txt

将 log-count-data.txt 上传到:

```
oss://your-bucket/log-count/log-count-data.txt
```

- your-bucket 如表示您自己创建的 bucket，本例子假设 region 为: cn-shenzhen.

```
bcs oss upload ./log-count-data.txt oss://your-bucket/log-count/log-count-data.txt
```

```
bcs oss cat oss://your-bucket/log-count/log-count-data.txt # 检查是否上传成功
```

- bcs o 命令提供几个 OSS 常用的功能，使用 bcs o -h 可以查看帮助，测试少量数据时使用很方便，但上传下载大量数据时不建议使用（没有实现多线程，上传下载慢。更多关于如何上传到 OSS，请参考 常用 OSS 工具。

(2) 准备任务程序

本例的作业程序是使用 python 编写的， 下载本例所需的程序: log-count.tar.gz

使用下面的目录解压：

```
mkdir log-count && tar -xvf log-count.tar.gz -C log-count
```

解压后的log-count/目录结构如下

```
log-count
|-- conf.py # 配置
|-- split.py # split 任务程序
|-- count.py # count 任务程序
|-- merge.py # merge 任务程序
```

- 注意：不需要改动程序

3. 提交作业

(1) 编写作业配置

在 log-count 的父目录下创建一个文件: job.cfg(此文件要与 log-count 目录平级), 内容如下：

```
[DEFAULT]
job_name=log-count
description=demo
pack=./log-count/
deps=split->count;count->merge

[split]
cmd=python split.py

[count]
cmd=python count.py
nodes=3

[merge]
cmd=python merge.py
```

这里描述了一个多任务的作业，任务的执行顺序是 split->count->merge。

- 关于 cfg 格式的描述，请看 多任务支持 。

(2) 提交命令

```
bcs sub --file job.cfg -r oss://your-bucket/log-count/:/home/input/ -w oss://your-bucket/log-count/:/home/output/
```

- -r 和 -w 表示只读挂载和可写映射，具体请看这里: OSS 挂载 。
- 同一个 oss 路径，可以挂载到不同的本地目录。但是不同的 oss 路径是不能挂载到同一个本地目录的，一定要注意。
- 这里需要注意的是，如果挂载的是目录，一定要以“/”结尾。

4. 查看作业运行状态

```
bcs j   # 获取作业列表, 每次获取作业列表后都会将列表缓存下来，一般第一个即是你刚才提交的作业
bcs ch 1 # 查看缓存中作业的状态，这里的1是bcs j命令查询出的刚刚提交的作业序号
bcs log 1 # 查看缓存中序号为1的作业日志
```

5. 查看结果

Job 结束后，可以使用以下命令查看存在 OSS 中的结果。

```
bcs oss cat oss://your-bucket/log-count/merge_result.json
```

内容应该如下：

```
{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}
```

Java快速开始

Java快速开始例子

本文档将介绍如何使用 Java 版 SDK 来提交一个作业，目的是统计一个日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数。

步骤

- 作业准备
  - 上传数据文件到 OSS
  - 使用示例代码
  - 编译打包
  - 上传到 OSS
- 使用 SDK 创建（提交）作业
- 查看结果

1. 作业准备

本作业是统计一个日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数。

该作业包含 3 个任务: split, count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数 (count 任务需要配置 InstanceCount 为 3，表示同时启动 3 台机器运行个 count 程序)。
- merge 任务会把 count 任务的结果统一合并起来。

DAG图例:

(1) 上传数据文件到OSS

下载本例所需的数据: log-count-data.txt

将 log-count-data.txt 上传到:

```
oss://your-bucket/log-count/log-count-data.txt
```

- your-bucket 表示您自己创建的 bucket，本例假设 region 为: cn-shenzhen.
- 如何上传到 OSS，请参考 OSS 上传文档 。

(2) 使用示例代码

本示例将采用 Java 来编写作业任务，使用 maven 来编译，推荐使用 IDEA：
http://www.jetbrains.com/idea/download/ 选择 Community 版本(免费).

示例程序下载：java-log-count.zip

这是一个 maven 工程。

- 注意：无需修改代码。

(3) 编译打包

运行命令编译打包:

```
mvn package
```

即可在 target 得到下面 3 个 jar 包:

```
batchcompute-job-log-count-1.0-SNAPSHOT-Split.jar
batchcompute-job-log-count-1.0-SNAPSHOT-Count.jar
batchcompute-job-log-count-1.0-SNAPSHOT-Merge.jar
```

再将 3 个 jar 包，打成一个 tar.gz 压缩包，命令如下:

```
> cd target  #进入 target 目录
> tar -czf worker.tar.gz *SNAPSHOT-*.jar #打包
```

运行以下命令，查看包的内容是否正确:

```
> tar -tvf worker.tar.gz
batchcompute-job-log-count-1.0-SNAPSHOT-Split.jar
batchcompute-job-log-count-1.0-SNAPSHOT-Count.jar
batchcompute-job-log-count-1.0-SNAPSHOT-Merge.jar
```

- 注意：BatchCompute 只支持以 tar.gz 为后缀的压缩包, 请注意务必用以上方式（gzip）打包, 否则将会无法解析。

(4) 上传到OSS

本例将 worke.tar.gz 上传到 OSS 的 your-bucket 中:

```
oss://your-bucket/log-count/worker.tar.gz
```

- 如要运行本例子，您需要创建自己的 bucket，并且把 worker.tar.gz 文件上传至您自己创建的 bucket 路径下。

2. 使用SDK创建(提交)作业

(1) 新建一个maven工程

在 pom.xml 中增加以下 dependencies：

```
   <dependencies>
<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-batchcompute</artifactId>
<version>5.2.0</version>
</dependency>

<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>3.2.3</version>
</dependency>
</dependencies>
```

- 请确定使用最新版本的 SDK: Java 版 SDK

(2) 新建一个java类： Demo.java

提交作业需要指定集群 ID 或者使用匿名集群参数。本例子使用匿名集群方式进行。匿名集群需要配置 2 个参数, 其中:

- 可用的镜像 ID, 可以使用系统提供的 Image，也可以自行制作镜像, 请看 使用镜像 。
- 实例规格（InstanceType,实例类型），请看 目前支持类型 。

在 OSS 中创建存储 StdoutRedirectPath（程序输出结果）和 StderrRedirectPath（错误日志）的文件路径，本例中创建的路径为

```
oss://your-bucket/log-count/logs/
```

- 如需运行本例，请按照上文所述的变量获取以及与上文对应的您的 OSS 路径对程序中注释中的变量进行修改。

Java SDK 提交程序模板如下，程序中具体参数含义请参照 SDK 接口说明。

Demo.java:

```
/*
* IMAGE_ID：ECS 镜像，由上文所述获取
* INSTANCE_TYPE: 实例类型，由上文所述获取
* REGION_ID：提交作业的地域，此项需与上文 OSS 存储 worker 的bucket 地域一致
* ACCESS_KEY_ID: AccessKeyId 可以由上文所述获取
* ACCESS_KEY_SECRET: AccessKeySecret 可以由上文所述获取
* WORKER_PATH：由上文所述打包上传的 worker 的 OSS 存储路径
* LOG_PATH：错误反馈和 task 输出的存储路径，logs 文件需事先自行创建
*/
import com.aliyuncs.batchcompute.main.v20151111.*;
import com.aliyuncs.batchcompute.model.v20151111.*;
import com.aliyuncs.batchcompute.pojo.v20151111.*;
```

```
import com.aliyuncs.exceptions.ClientException;

import java.util.ArrayList;
import java.util.List;

public class Demo {

static String IMAGE_ID = "img-ubuntu";; //这里填写您的 ECS 镜像 ID
static String INSTANCE_TYPE = "ecs.sn1.medium"; //根据 region 填写合适的 InstanceType

static String REGION_ID = "cn-shenzhen"; //这里填写 region
static String ACCESS_KEY_ID = ""; //"your-AccessKeyId"; 这里填写您的 AccessKeyId
static String ACCESS_KEY_SECRET = ""; //"your-AccessKeySecret"; 这里填写您的 AccessKeySecret
static String WORKER_PATH = ""; //"oss://your-bucket/log-count/worker.tar.gz"; // 这里填写您上传的 worker.tar.gz
的 OSS 存储路径
static String LOG_PATH = ""; // "oss://your-bucket/log-count/logs/"; // 这里填写您创建的错误反馈和 task 输出的 OSS
存储路径
static String MOUNT_PATH = ""; // "oss://your-bucket/log-count/";

public static void main(String[] args){

/** 构造 BatchCompute 客户端 */
BatchCompute client = new BatchComputeClient(REGION_ID, ACCESS_KEY_ID, ACCESS_KEY_SECRET);

try{

/** 构造 Job 对象 */
JobDescription jobDescription = genJobDescription();

//创建 Job
CreateJobResponse response = client.createJob(jobDescription);

//创建成功后，返回 jobId
String jobId = response.getJobId();

System.out.println("Job created success, got jobId: "+jobId);


//查询 job 状态
GetJobResponse getJobResponse = client.getJob(jobId);

Job job = getJobResponse.getJob();

System.out.println("Job state:"+job.getState());

} catch (ClientException e) {
e.printStackTrace();

System.out.println("Job created failed, errorCode:"+ e.getErrCode()+", errorMessage:"+e.getErrMsg());
}
}

private static JobDescription genJobDescription(){
```

```
JobDescription jobDescription = new JobDescription();

jobDescription.setName("java-log-count");
jobDescription.setPriority(0);
jobDescription.setDescription("log-count demo");
jobDescription.setJobFailOnInstanceFail(true);
jobDescription.setType("DAG");

DAG taskDag = new DAG();


/** 添加 split task */

TaskDescription splitTask = genTaskDescription();
splitTask.setTaskName("split");
splitTask.setInstanceCount(1);
splitTask.getParameters().getCommand().setCommandLine("java -jar batchcompute-job-log-count-1.0-SNAPSHOT-
Split.jar");
taskDag.addTask(splitTask);

/** 添加 count task */
TaskDescription countTask = genTaskDescription();
countTask.setTaskName("count");
countTask.setInstanceCount(3);
countTask.getParameters().getCommand().setCommandLine("java -jar batchcompute-job-log-count-1.0-
SNAPSHOT-Count.jar");
taskDag.addTask(countTask);

/** 添加 merge task */
TaskDescription mergeTask = genTaskDescription();
mergeTask.setTaskName("merge");
mergeTask.setInstanceCount(1);
mergeTask.getParameters().getCommand().setCommandLine("java -jar batchcompute-job-log-count-1.0-
SNAPSHOT-Merge.jar");
taskDag.addTask(mergeTask);



/** 添加 Task 依赖: split-->count-->merge */

List<String> taskNameTargets = new ArrayList();
taskNameTargets.add("merge");
taskDag.addDependencies("count", taskNameTargets);

List<String> taskNameTargets2 = new ArrayList();
taskNameTargets2.add("count");
taskDag.addDependencies("split", taskNameTargets2);

//dag
jobDescription.setDag(taskDag);

return jobDescription;
}

private static TaskDescription genTaskDescription(){
```

```
AutoCluster autoCluster = new AutoCluster();
autoCluster.setInstanceType(INSTANCE_TYPE);
autoCluster.setImageId(IMAGE_ID);
//autoCluster.setResourceType("OnDemand");

TaskDescription task = new TaskDescription();
//task.setTaskName("Find");

//如果使用 VPC，需要配置 cidrBlock, 请确保 IP 段不冲突
Configs configs = new Configs();
Networks networks = new Networks();
VPC vpc = new VPC();
vpc.setCidrBlock("192.168.0.0/16");
networks.setVpc(vpc);
configs.setNetworks(networks);
autoCluster.setConfigs(configs);

//打包上传的作业的 OSS 全路径
Parameters p = new Parameters();
Command cmd = new Command();
//cmd.setCommandLine("");
//打包上传的作业的 OSS 全路径
cmd.setPackagePath(WORKER_PATH);
p.setCommand(cmd);
//错误反馈存储路径
p.setStderrRedirectPath(LOG_PATH);
//最终结果输出存储路
p.setStdoutRedirectPath(LOG_PATH);

task.setParameters(p);
task.addInputMapping(MOUNT_PATH, "/home/input");
task.addOutputMapping("/home/output",MOUNT_PATH);

task.setAutoCluster(autoCluster);
//task.setClusterId(clusterId);
task.setTimeout(30000); /* 30000 秒*/
task.setInstanceCount(1); /** 使用 1 个实例来运行 */

return task;
}
}
```

正常输出样例：

```
Job created success, got jobId: job-01010100010192397211
Job state:Waiting
```

3. 查看作业状态

您可以用 SDK 中的 获取作业信息 方法获取作业状态：

```
//查询 job 状态
```

```
GetJobResponse getJobResponse = client.getJob(jobId);
Job job = getJobResponse.getJob();
System.out.println("Job state:"+job.getState());
```

Job 的 state 可能为：Waiting、Running、Finished、Failed、Stopped.

4. 查看结果

您可以登录 batchcompute 控制台 查看 job 状态。

Job 运行结束，您可以登录 OSS 控制台 查看your-bucket 这个 bucket 下面的这个文件：/log-count/merge_result.json。

内容应该如下：

```
{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}
```

您也可以使用 OSS 的 SDK 来获取结果。

Python快速入门示例

本文档将介绍如何使用 Python 版 SDK 来提交一个作业，目的是统计一个日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数。

步骤预览

- 作业准备
    - 上传数据文件到 OSS
    - 上传任务程序到 OSS
- 使用 SDK 创建(提交)作业
- 查看结果

1. 作业准备

本作业是统计一个日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数。

该作业包含3个任务: split, count 和 merge:

- split 任务会把日志文件分成 3 份。
- count 任务会统计每份日志文件中“INFO”,”WARN”,”ERROR”,”DEBUG”出现的次数 (count

任务需要配置 InstanceCount 为 3，表示同时启动3台机器运行个 count 程序)。
- merge 任务会把 count 任务的结果统一合并起来。

DAG图例:

(1) 上传数据文件到OSS

下载本例子所需的数据：log-count-data.txt

将 log-count-data.txt 上传到：

```
oss://your-bucket/log-count/log-count-data.txt
```

- your-bucket 表示您自己创建的 bucket，本例子假设 region 为: cn-shenzhen。
- 如何上传到 OSS，请参考 OSS 上传文档 。

(2) 上传任务程序到OSS

本例的作业程序是使用 python 编写的， 下载本例子所需的程序: log-count.tar.gz

本例不需要改动示例代码。直接将 log-count.tar.gz 上传到 oss，如上传到：

```
oss://your-bucket/log-count/log-count.tar.gz。
```

如何上传前面已经讲过。

- BatchCompute 只支持以 tar.gz 为后缀的压缩包, 请注意务必用以上方式（gzip）打包, 否则将会无法解析。

  如果您要修改代码，可以解压后修改，然后要用下面的方法打包：

  命令如下:

```
> cd log-count  #进入目录
> tar -czf log-count.tar.gz * #打包，将所有这个目录下的文件打包到 log-count.tar.gz
```

  可以运行这条命令查看压缩包内容：

```
```

```
$ tar -tvf log-count.tar.gz
```

可以看到以下列表:

```
conf.py
count.py
merge.py
split.py
```

2. 使用SDK创建(提交)作业

python SDK 的相关下载与安装请参阅 这里。

v20151111 版本，提交作业需要指定集群 ID 或者使用匿名集群参数。本例子使用匿名集群方式进行，匿名集群需要配置 2 个参数, 其中：

- 可用的镜像 ID, 可以使用系统提供的 Image，也可以自行制作镜像, 请参考 使用镜像。
- 实例规格（InstanceType,实例类型），请参考 目前支持类型。

在 OSS 中创建存储 StdoutRedirectPath（程序输出结果）和 StderrRedirectPath（错误日志）的文件路径，本例中创建的路径为

oss://your-bucket/log-count/logs/

- 如需运行本例，请按照上文所述的变量获取以及与上文对应的您的 OSS 路径对程序中注释中的变量进行修改。

Python SDK 提交程序模板如下，程序中具体参数含义请参照 这里。

```
#encoding=utf-8
import sys

from batchcompute import Client, ClientError
from batchcompute import CN_SHENZHEN as REGION #这里的region根据实际情况填写
from batchcompute.resources import (
JobDescription, TaskDescription, DAG, AutoCluster
)

ACCESS_KEY_ID='' # 填写您的 AK
ACCESS_KEY_SECRET='' # 填写您的 AK

IMAGE_ID = 'img-ubuntu' #这里填写您的镜像 ID
INSTANCE_TYPE = 'ecs.sn1.medium' # 根据实际 region 支持的 InstanceType 填写
WORKER_PATH = '' # 'oss://your-bucket/log-count/log-count.tar.gz' 这里填写您上传的 log-count.tar.gz 的 OSS 存储路径
LOG_PATH = '' # 'oss://your-bucket/log-count/logs/' 这里填写您创建的错误反馈和 task 输出的 OSS 存储路径
```

```
OSS_MOUNT= '' # 'oss://your-bucket/log-count/' 同时挂载到/home/inputs 和 /home/outputs

client = Client(REGION, ACCESS_KEY_ID, ACCESS_KEY_SECRET)

def main():
try:
job_desc = JobDescription()

# Create auto cluster.
cluster = AutoCluster()
cluster.InstanceType = INSTANCE_TYPE
cluster.ResourceType = "OnDemand"
cluster.ImageId = IMAGE_ID

# Create split task.
split_task = TaskDescription()
split_task.Parameters.Command.CommandLine = "python split.py"
split_task.Parameters.Command.PackagePath = WORKER_PATH
split_task.Parameters.StdoutRedirectPath = LOG_PATH
split_task.Parameters.StderrRedirectPath = LOG_PATH
split_task.InstanceCount = 1
split_task.AutoCluster = cluster
split_task.InputMapping[OSS_MOUNT]='/home/input'
split_task.OutputMapping['/home/output'] = OSS_MOUNT


# Create map task.
count_task = TaskDescription(split_task)
count_task.Parameters.Command.CommandLine = "python count.py"
count_task.InstanceCount = 3
count_task.InputMapping[OSS_MOUNT] = '/home/input'
count_task.OutputMapping['/home/output'] = OSS_MOUNT

# Create merge task
merge_task = TaskDescription(split_task)
merge_task.Parameters.Command.CommandLine = "python merge.py"
merge_task.InstanceCount = 1
merge_task.InputMapping[OSS_MOUNT] = '/home/input'
merge_task.OutputMapping['/home/output'] = OSS_MOUNT

# Create task dag.
task_dag = DAG()
task_dag.add_task(task_name="split", task=split_task)
task_dag.add_task(task_name="count", task=count_task)
task_dag.add_task(task_name="merge", task=merge_task)
task_dag.Dependencies = {
'split': ['count'],
'count': ['merge']
}

# Create job description.
job_desc.DAG = task_dag
job_desc.Priority = 99 # 0-1000
job_desc.Name = "log-count"
job_desc.Description = "PythonSDKDemo"
job_desc.JobFailOnInstanceFail = True
```

```
job_id = client.create_job(job_desc).Id
print('job created: %s' % job_id)

except ClientError, e:
print (e.get_status_code(), e.get_code(), e.get_requestid(), e.get_msg())

if __name__ == '__main__':
sys.exit(main())
```

3. 查看作业状态

您可以用 SDK 中的 获取作业信息 方法获取作业状态：

```
jobInfo = client.get_job(job_id)
print (jobInfo.State)
```

State 状态可能为：Waiting, Running, Finished, Failed, Stopped。

4. 查看结果

您可以登录 OSS 控制台 查看 your-bucket 下面的这个文件：/log-count/merge_result.json。

内容应该如下：

```
{"INFO": 2460, "WARN": 2448, "DEBUG": 2509, "ERROR": 2583}
```

- 您也可以使用 OSS 的 SDK 来获取结果。