批量计算

快速入门

快速入门

如何开通批量计算

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，点左上角“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API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。

使用控制台：适合初级用户，使用网页可视化界面管理作业和集群。控制台快速开始例子

使用命令行工具：适合初级用户，快速提交程序上云，可以管理作业集群等。命令行工具快速开始例子

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。Java快速开始例子，Python快速开始例子

使用云渲染管理系统：专门为渲染场景定制的web管理系统。

如果您还没开通批量计算服务，请先开通。

步骤预览

- 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

- 假设您的bucket创建在 cn-shenzhen 区域, 假设命名为: your-bucket。
- 登录后配置 —osspath=oss://your-bucket/cli/ 。

2. 作业准备

本作业的目的是排序，将 input.txt 中的单词排序后写入 output.txt。

(1) 上传数据文件到OSS

先自行创建 input.txt。

input.txt的内容(确保一行一个单词):

```
Ken
Alice
Jene
Bob
Peter
```

将 input.txt 上传到:

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

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

> 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OSS上传文档。

3. 提交作业

我们将提交一个批量计算作业，该作业包含一个任务，该任务启动一个VM(虚拟机), 运行ubuntu镜像，然后执行一个命令（您可以执行任何命令，只要您的镜像支持）：

/bin/sort /home/input/input.txt > /home/output/output.txt

批量计算支持将OSS目录oss://your-bucket/cli/sum/inputs/挂载为VM里的本地目录/home/input/,VM中可以直接访问到: /home/input/input.txt。

提交作业命令：

```
bcs sub "/bin/sort /home/input/input.txt > /home/output/output.txt" sort-job -r oss://your-bucket/cli/sum/inputs/:/home/inputs/ -w oss://your-bucket/cli/sum/outputs/:/home/outputs/
```

返回作业ID表示提交成功.

> 这里使用默认镜像和默认实例类型
>
> sort-job 是作业名称
>
> -r 表示只读挂载，将OSS目录只读挂载为VM本地目录。（注意挂载OSS时是只读挂载, /home/inputs/目录下的文件是不能修改的。查看OSS挂载详解）
>
> -w 表示输出结果目录映射，写入VM本地目录/home/outputs/的文件，会在程序运行完后，被自动

上传到OSS目录oss://your-bucket/cli/sum/outputs/下。

4. 查看作业运行状态及运行结果

作业提交成功后，可以通过下面命令查看状态和日志：

```
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 job-idxxxxxxxx # 查看作业详情

# or 你可以使用序号查询
bcs j 1 # 查看缓存中第一个作业的详情 (注意，使用序号前需要先 `bcs j` 获取列表缓存)

bcs log 1 # 查看缓存中第一个作业日志
```

作业运行结束，可以使用以下命令查看结果：

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

结果应该是这样的:

```
Alice
Bob
Ken
Jene
Peter
```

如果您还没开通批量计算服务，请先开通。

步骤预览

- 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.
>
> 更多关于如何上传到OSS，请参考OSS上传文档。

(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
```

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目录只读挂载为VM本地目录

-w 表示可写挂载，将OSS目录挂载为VM本地目录，写入VM本地目录的文件，会在程序运行完后，被自动上传到OSS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
```

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。

如果您还没开通批量计算服务，请先开通。

步骤预览

- 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
```

(2) 准备任务程序

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

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。

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 # 查看缓存中第一个作业的状态
bcs log 1 # 查看缓存中第一个作业日志
```

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}
```

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。

如果您还没开通批量计算服务，请先开通。

步骤预览

作业准备。

- 上传数据文件到OSS。
- 上传任务程序到OSS。

使用控制台提交作业。

查看作业状态。

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，本例子假设region为: cn-shenzhen.
- 如何上传到OSS，请参考OSS上传文档。

上传任务程序到OSS

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

本例子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:

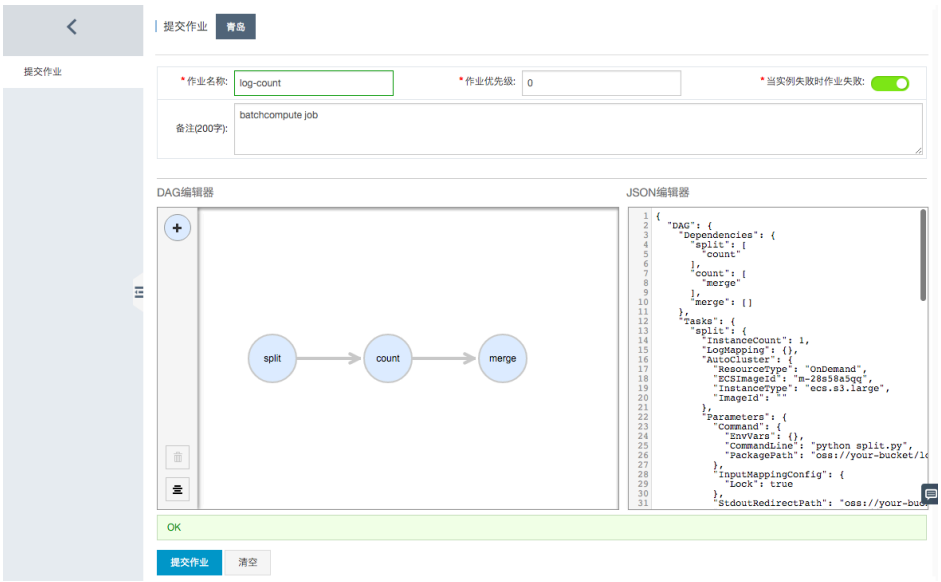
```
conf.py
count.py
merge.py
```

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bucket的region一致)。



这里使用了AutoCluster提交作业, 匿名集群需要至少配置 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。

如果需要运行本例子，还需把 PackagePath （作业打包上传的OSS路径，本例子中为 oss://your-bucket/log-count/log-count.tar.gz）、

StdoutRedirectPath、StderrRedirectPath（任务结果和错误的输出地址），修改成与上文对应的您的OSS路径(本例子中为：oss://your-bucket/log-count/logs/)。

作业JSON模板如下，具体参数含义请参照这里。

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

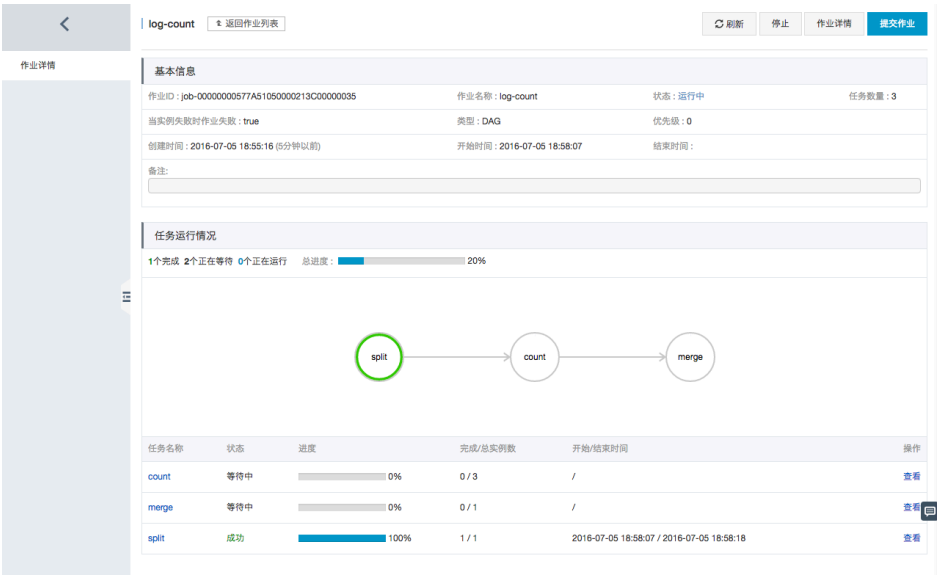
```
"count": [
"merge"
],
"merge": []
},
"Tasks": {
"split": {
"InstanceCount": 1,
"LogMapping": {},
"AutoCluster": {
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1.medium",
"ImageId": "img-ubuntu"
},
"Parameters": {
"Command": {
"EnvVars": {},
"CommandLine": "python split.py",
"PackagePath": "oss://your-bucket/log-count/log-count.tar.gz"
},
"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": {
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1.medium",
"ImageId": "img-ubuntu"
},
"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": {
"ResourceType": "OnDemand",
"InstanceType": "ecs.sn1.medium",
"ImageId": "img-ubuntu"
},
"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"
}
```

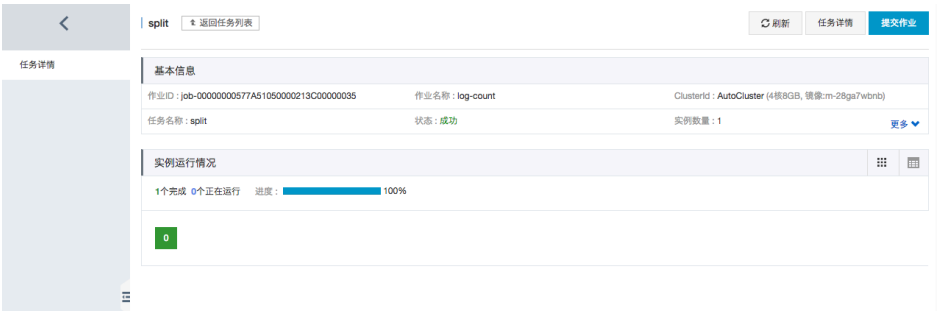

确定各个参数及路径填写正确后，点击左下角的**提交作业**，并确认。

3. 查看作业状态

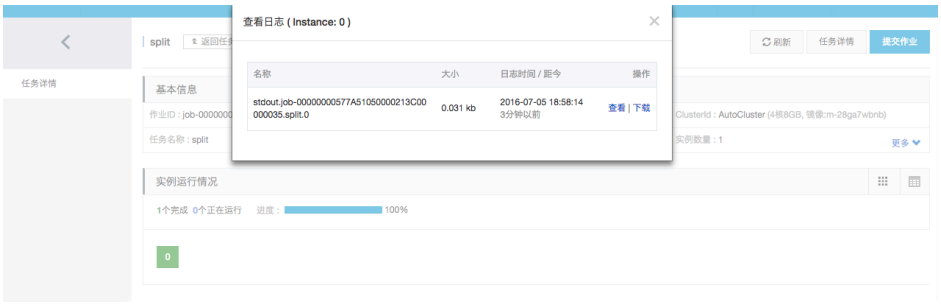
单击作业列表中最新提交的log-count作业，可以查看详情：



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}
```

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。

如果您还没开通批量计算服务，请先开通。

步骤预览

- 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) 使用示例代码

这里我们将采用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的路径下。

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>3.1.0</version>
</dependency>

<dependency>
<groupId>com.aliyun</groupId>
<artifactId>aliyun-java-sdk-core</artifactId>
<version>3.0.3</version>
</dependency>
</dependencies>
```

(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/
```

- 如需运行本例子，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：区域为青岛/杭州，目前只有青岛开通，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");

//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。

如果您还没开通批量计算服务，请先开通。

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

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

本例子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/

- 如需运行本例子，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
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 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。

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暂时不支持用户登录到计算节点。

BatchCompute的计算节点VM没有公网IP，不支持公网地址访问。