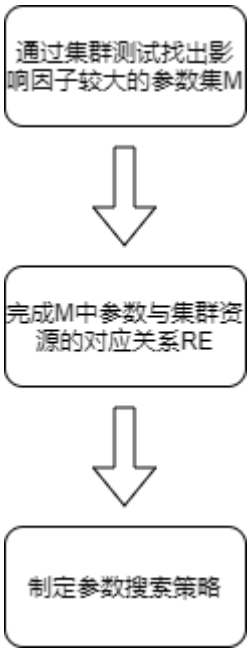
自动化调优模型详解

本文将对本科毕业论文《分布式文件系统Alluxio的参数智能调优研究与实现》中设计的参数自动化调优模型进行详细讲解，接下来将从：定制调优模型和运行任务调优两个部分对模型进行介绍。

定制调优模型

从下图可以看出，在定制调优模型部分可以进行细分。



1. **针对不同数据类型的参数选择不同的参数搜索方案。**

   对枚举型的参数而言，需要对每一个候选项进行遍历，如：

```
for item in 'ALWAYS' 'NEVER' 'ONCE'; do
    param='Alluxio.user.file.metadata.load.type'
    runJob $param $item
    if [ $( expr $cpu_param \> $memCompare) == 1 ]; then
        echo " Successfully change the $param's value to $item"
       exit
    fi
done
```

对连续型或范围较大的离散型参数而言，需要手动对定义域进行分割，选择几个候选项进行遍历，如：

```
    for item in '0' '1' '2' '4' '8'; do
        param='alluxio.user.network.netty.worker.threads'
        runJob $param $item
        if [ $( expr $cpu_param \> $cpuCompare) == 1 ]; then
            echo " Successfully change the $param's value to $item"
           exit
        fi
    done
```

2. **根据经验找出影响因子较大的参数集，并完成参数集与监测资源的关联。**

   参数集与CPU资源相关联

参数名	默认值
Alluxio.user.network.netty.worker.threads	0
Alluxio.worker.network.block.reader.threads.max	2048
Alluxio.user.file.writetype.default	ASYNC_THROUGH
Alluxio.user.file.readtype.default	CACHE_PROMOTE

   参数集与Memory资源相关联

参数名	默认值
Alluxio.user.block.write.location.policy.class	LocalFirstPolicy
Alluxio.user.file.metadata.load.type	ONCE
Alluxio.worker.evictor.class	LRUEvictor
Alluxio.worker.file.buffer.size	1MB

   参数集与Network资源相关联

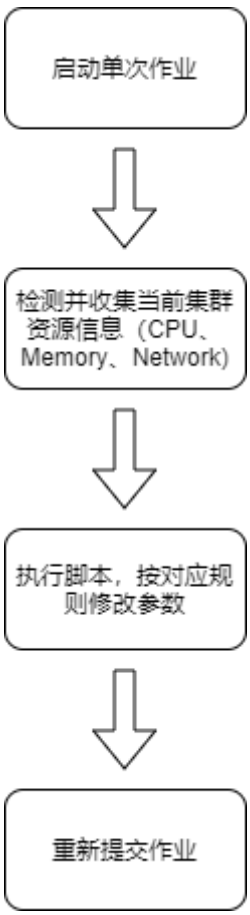
参数名	默认值
Alluxio.user.file.readtype.default	CACHE_PROMOTE
Alluxio.user.file.writetype.default	ASYNC_THROUGH
Alluxio.user.rpc.retry.base.sleep	50ms
Alluxio.user.rpc.retry.max.duration	2min
Alluxio.user.network.data.timeout	30sec

   在代码中实现了上述参数集与监测资源的关联，所以当监测到系统资源利用率不足时，可以调整相对应的参数以提升资源利用率。

```
# 当某项资源被确认为利用率不足时，将会对相应参数进行调整
if [ Network_chosen == true ]; then
……
elif [ CPU_chosen == true ]; then
……
else  # Memory_chosen is true
```

运行任务调优

从下图可以看出，运行任务调优可以分成：检测并收集资源信息、按规则修改参数、重新提交作业三个部分。



1. 检测并收集资源信息

该部分的功能由两个脚本完成，***getSample.sh***（负责采样）***getResource.sh***（负责集群资源的监测）

1.1 *getResource.sh*功能介绍

脚本 ***getResource.sh*** 的功能为实时获取当前集群资源的使用情况。该脚本主要分成两个部分：获取CPU和Memory资源、获取Network资源信息，分别由两个函数实现。

- 函数 ***getSysInfo*** 部分代码

```
psResult=`nohup ps -aux`
echo "$psResult" >> $cpu_snapshot
while read line
do
   if [ ${#line} > 0 ]; then
       cpu_num=`echo $line |grep -v "CPU"| awk -F ' ' '{print $3}'`
       mem_num=`echo $line |grep -v "MEM"| awk -F ' ' '{print $4}'`
       if [[ ${#cpu_num} > 0 && ${#mem_num} > 0 ]];then
           cpu_total=$(echo $cpu_total $cpu_num | awk '{printf "%.3f\n",$1 + $2}')
           mem_total=$(echo $mem_total $mem_num | awk '{printf "%.3f\n",$1 + $2}')
        fi
   fi
 done < $cpu_snapshot
```

由上述代码可以看出，首先通过*ps*指令将当前实时资源情况保存在文件中，便于后续操作。接着，从该文件中读取每一行，并获取该行的CPU、Memory利用率，最后进行求和。其中，*grep*指令可以过滤不符合条件的行，*awk*指令负责对每一行数据进行处理，以获取相应的信息。

- 函数 ***getNetInfo*** 部分代码

```
receivePre=$(cat /proc/net/dev | grep $eth| tr : " "| awk '{print $2}')
putPre=$(cat /proc/net/dev | grep $eth| tr : " "| awk '{print $10}')
.....
net_receive=$(echo $receive | awk '{print $1/1048576}')
net_put=$(echo $put | awk '{print $1/1048576}')
```

该函数执行的思路：首先通过访问*/proc/net/dev*文件来获取当前主机端口接收和发送的字节数，接着1s后再次获取该值，即可根据定义计算出这1s内的网络速率。

1.2 *getSample.sh*功能介绍

通过设置*SampleDots*（采样数，本模型设置为10次）来统计一定时间间隔内的集群资源使用情况。

```
./getResource.sh # To get Resource Rate
arr=($(cat ${sourceFileName} | head -n 1))
cpuRate=$(echo $cpuRate ${arr[0]} | awk '{printf "%.2f\n",$1 + $2}')
memRate=$(echo $memRate ${arr[1]} | awk '{printf "%.2f\n",$1 + $2}')
networkRate=$(echo $networkRate ${arr[2]} | awk '{printf "%.2f\n",$1 + $2}')
.....
cpuRate=$(echo $cpuRate $SampleDots | awk '{printf "%.2f\n", $1 / $2}')
memRate=$(echo $memRate $SampleDots | awk '{printf "%.2f\n", $1 / $2}')
networkRate=$(echo $networkRate $SampleDots | awk '{printf "%.2f\n", $1 / $2}')
```

首先，针对每一个采样点，执行脚本***getResource.sh***来获取某一时刻的资源使用情况，之后再进行累加求和、求平均值操作，最终获得采样时间段内各项资源的平均使用情况。在这个过程中，脚本***getResource.sh***和脚本***getSample.sh***之间的通信是通过文件***resource.txt***来实现的，前者将单次采样的资源信息存放在文本文件中，由后者在使用时获取。

通过上述两个脚本的操作，可以完成单次作业执行时资源监测的功能。

2. 按规则修改参数

在进行参数优化前，需要获取作业在默认参数配置情况下的执行情况，以便在后续参数优化时评判当前参数的效果。在参数优化过程中，对每一项参数的所有候选值进行遍历，直到找出能提升资源利用率的参数值。

2.1 默认参数下的资源消耗

为了实现在作业执行的同时进行资源监测的功能，代码中实现了并行执行命令行的操作。

```
timeout $runningTime $jobcommand &
./getSample.sh &
wait
line=($(cat ${sourceFileName} | head -n 1)) #sourceFileName中存储了资源使用情况
cpu_default=${line[0]}
mem_default=${line[1]}
network_default=${line[2]}
```

*runningTime*为每次作业的执行时间（本模型中设置的是90s，稍大于采样时间），*jobcommand*为作业执行命令（为全局变量，在脚本中设置）；通过执行*getSample.sh* 脚本可以在运行作业的同时对资源进行检测；*wait*命令可以保证后续命令只有在上述命令执行完成后才能执行。在分配的时间运行完成后，保存采样统计的资源使用情况。

本文通过比较三项资源：CPU、Memory、Network的资源利用率，找出三者中的最小值，然后调整与之相关联的参数，具体实现由函数***judgeResource***完成。

```
judgeResource() {
    Net_percent=$(echo $3 $netSpeed | awk '{printf "%.2f\n", $1 / $2}')
    # 获取network的资源利用率 netSpeed网络带宽
    if [ $(expr $Net_percent \< $1) == 1 ]
    then
        if [ $(expr $Net_percent \< $2) == 1 ]
        then
            Network_chosen=true
        else
            Memory_chosen=true
        fi
    else
    ..............
}
judgeResource $cpu_default $mem_default $network_default #比较三者中资源利用率的最小值
```

由于shell中的“>”、“<"无法直接比较浮点数，因此采用 *"$(expr $a \< $b) == 1"*的方法来比较浮点数a、b的大小。

2.2 参数遍历

参数搜索时执行的操作与默认参数情况下的操作相似，均包含作业执行与资源监测两个部分。主要功能在函数***runJob***中实现。

```
runJob() {
    ……
    sed -i "s/$1.*/$1=$2/" $properties_name
    scp $properties_name gurong1228@slave205:${properties_name}
    scp $properties_name gurong1228@slave206:${properties_name}
    # distribute the properties file to all slaves
    ……
    timeout $runningTime $jobcommand &
    ./getSample.sh &
    wait # instructions above finished
    ……
}
```

1. 需要修改配置文件*alluxio-site.properties*中相应参数的值为*item*，通过使用*sed*命令搭配正则表达式对相应内容进行替换来达到修改的目的。
2. 再将配置文件分发到所有的slave节点上，进行同步。
3. 对作业执行和资源监测并行处理，在给定的作业执行时间内完成集群资源使用情况的监测。
4. 将获取的资源使用率和默认参数配置情况下的进行对比，判断是否能提升资源利用率。若利用率能得到提升，则退出循环，并把当前参数配置当作推荐值。

3. 重新提交作业

在完成上述的参数优化后，再次执行提交的作业，输出作业的运行时间，完成参数调优流程。