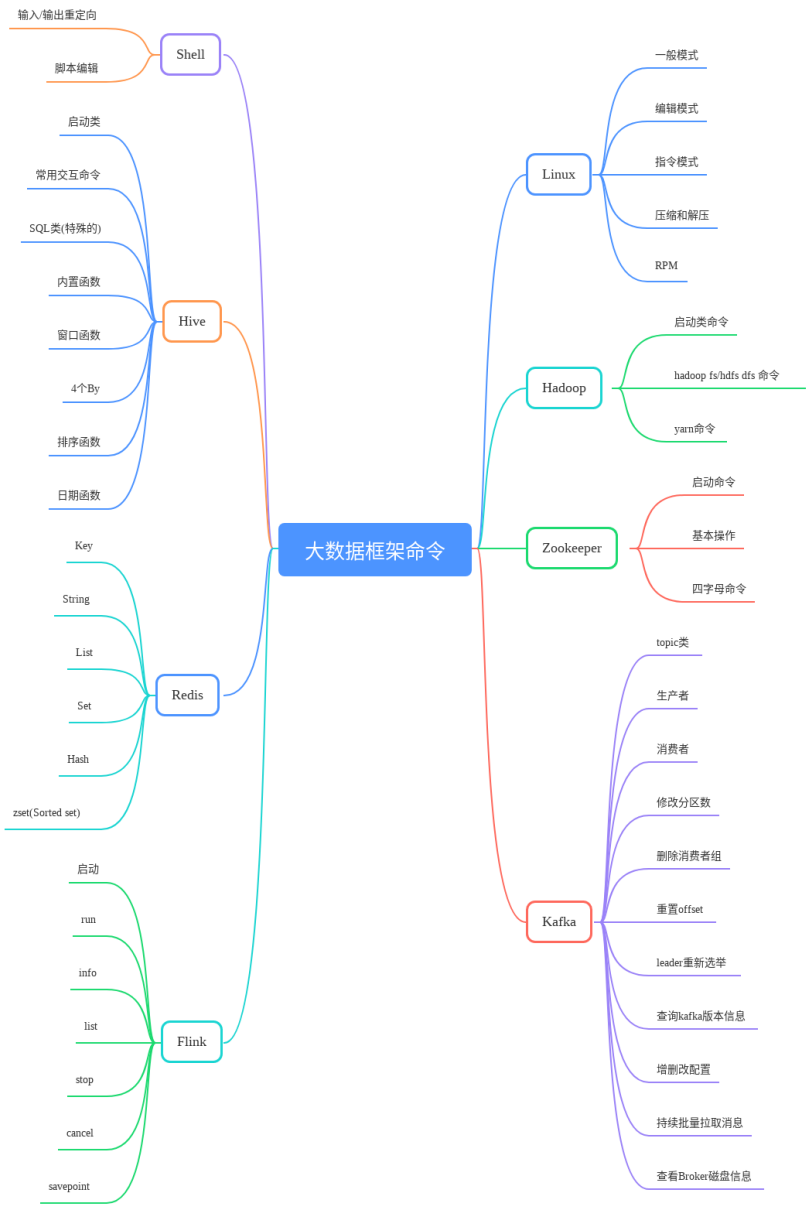
史上最全干货！大数据框架知识点总结

本文作者：左右
本文档来自公众号：大数据左右手

技术如同手中的水有了生命似的，汇聚在了一起。作为 **大数据左右手** 的号主，致力于大数据技术的学习，分享大数据原理、架构、实时、离线、面试与总结，分享生活思考与读书见解。我想总有适合你的那一篇。

本文档总结范围如下


大数据框架命令
Shell
输入/输出重定向
脚本编辑
Hive
启动类
常用交互命令
SQL类(特殊的)
内置函数
窗口函数
4个By
排序函数
日期函数
Redis
Key
String
List
Set
Hash
zset(Sorted set)
Flink
启动
run
info
list
stop
cancel
savepoint
Linux
一般模式
编辑模式
指令模式
压缩和解压
RPM
Hadoop
启动类命令
hadoop fs/hdfs dfs 命令
yarn命令
Zookeeper
启动命令
基本操作
四字母命令
Kafka
topic类
生产者
消费者
修改分区数
删除消费者组
重置offset
leader重新选举
查询kafka版本信息
增删改配置
持续批量拉取消息
查看Broker磁盘信息

Linux（vi/vim）

一般模式

语法	功能描述
yy	复制光标当前一行
y数字y	复制一段（从第几行到第几行）
p	箭头移动到目的行粘贴
u	撤销上一步
dd	删除光标当前行
d数字d	删除光标（含）后多少行
x	删除一个字母，相当于del
X	删除一个字母，相当于Backspace
yw	复制一个词
dw	删除一个词
shift+^	移动到行头
shift+$	移动到行尾
1+shift+g	移动到页头，数字
shift+g	移动到页尾
数字N+shift+g	移动到目标行

编辑模式

按键	功能
i	当前光标前
a	当前光标后
o	当前光标行的下一行
I	光标所在行最前
A	光标所在行最后
O	当前光标行的上一行

指令模式

命令	功能
:w	保存
:q	退出
:!	强制执行
/要查找的词	n 查找下一个，N 往上查找
? 要查找的词	n是查找上一个，shift+n是往下查找
:set nu	显示行号
:set nonu	关闭行号

压缩和解压

gzip/gunzip 压缩

（1）只能压缩文件不能压缩目录

（2）不保留原来的文件

gzip压缩：gzip hello.txt

gunzip解压缩文件：gunzip hello.txt.gz

zip/unzip 压缩

可以压缩目录且保留源文件

zip压缩（压缩 1.txt 和2.txt，压缩后的名称为mypackage.zip）：zip hello.zip hello.txt world.txt

unzip解压：unzip hello.zip

unzip解压到指定目录：unzip hello.zip -d /opt

tar 打包

tar压缩多个文件：tar -zcvf hello.txt world.txt

tar压缩目录：tar -zcvf hello.tar.gz opt/

tar解压到当前目录：tar -zxvf hello.tar.gz

tar解压到指定目录：tar -zxvf hello.tar.gz -C /opt

RPM

RPM查询命令：rpm -qa |grep firefox

RPM卸载命令：

rpm -e xxxxxx

rpm -e --nodeps xxxxxx（不检查依赖）

RPM安装命令：

rpm -ivh xxxxxx.rpm

rpm -ivh --nodeps fxxxxxx.rpm（--nodeps，不检测依赖进度）

选项	功能
-i	-i=install，安装
-v	-v=verbose，显示详细信息
-h	-h=hash，进度条
--nodeps	--nodeps，不检测依赖进度

Shell

输入/输出重定向

命令	功能说明
command > file	将输出重定向到 file
command < file	将输入重定向到 file
command >> file	将输出以追加的方式重定向到 file
n > file	将文件描述符为 n 的文件重定向到 file
n >> file	将文件描述符为 n 的文件以追加的方式重定向到 file
n >& m	将输出文件 m 和 n 合并
n <& m	将输入文件 m 和 n 合并
<< tag	将开始标记 tag 和结束标记 tag 之间的内容作为输入

脚本编辑

快捷方式	功能说明
shift	参数左移
$@	所有的参数
$#	参数的个数

Hadoop

启动类命令

功能说明	命令脚本
启动hdfs集群	sbin/start-dfs.sh
启动yarn	sbin/start-yarn.sh

hadoop fs/hdfs dfs 命令

功能说明	命令
创建目录	hdfs dfs -mkdir -p /data/flink
显示目录	hdfs dfs -ls /
从HDFS拷贝到本地	hdfs dfs -copyToLocal /data/data.txt ./
文件上传到集群(从本地)	hhdfs dfs -copyFromLocal data.txt /
文件下载	hdfs dfs -get /data/flink
删除集群的文件	hdfs dfs -rm /data/flink
删除文件夹	hdfs dfs -rm -r -skipTrash /data
从本地剪切粘贴到HDFS	hdfs dfs -moveFromLocal data.txt /data/
追加一个文件到已经存在的文件末尾hdfs dfs -appendToFile data1.txt /data/data.txt	
显示文件内容	hdfs dfs -cat data.txt
修改文件所属权限	hdfs dfs -chmod 777 xxx.sh
修改文件所属用户组	hdfs dfs -chown root:root data.txt
从HDFS的一个路径拷贝到HDFS的另一个路径	hdfs dfs -cp data.txt /data1.txt
在HDFS目录中移动文件	hdfs dfs -mv data.txt /opt/
合并下载多个文件	hdfs dfs -getmerge /data/* ./data_merge.txt
hadoop fs -put	等同于copyFromLocal
显示一个文件的末尾	hdfs dfs -tail data.txt
删除文件或文件夹	hdfs dfs -rm /data/data.txt
删除空目录	hdfs dfs -rmdir /data
统计文件夹的大小信息	hdfs dfs -s -h /data
统计文件夹下的文件大小信息	hdfs dfs -h /data
设置HDFS中文件的副本数量	hdfs dfs -setrep 3 /data/data.txt

yarn命令

功能说明	命令
查看正在运行的yarn任务列表	yarn application -list appID
kill掉指定id的yarn任务	yarn application -kill appID
查看任务日志信息	yarn logs -applicationId appID

Zookeeper

启动命令

功能说明	命令脚本
启动zookeeper服务	zkServer.sh start
查看zookeeper状态	zkServer.sh status
停止zookeeper服务	zkServer.sh stop
启动zookeeper客户端	zkCli.sh -server 127.0.0.1:2181
退出zookeeper客户端	quit

基本操作

功能说明	命令脚本
当前znode中所包含的内容	ls /
创建普通节点(前面是节点的路径，后面是值)	create /bigdata/flink "flink"
获取节点的值	get /bigdata
修改节点的值	set /bigdata/flink "flinksql"
删除节点	delete /bigdata/flink
递归删除节点	rmr /bigdata

四字母命令

命令	功能说明	例子
conf	zk服务配置的详细信息	echo conf | nc 127.0.0.1 2181
stat	客户端与zk连接的简要信息	参考上面
srvr	zk服务的详细信息	参考上面
cons	客户端与zk连接的详细信息	参考上面
mntr	zk服务目前的性能状况	参考上面
crst	重置当前的所有连接、会话	参考上面
dump	列出未经处理的会话和连接信息	参考上面
envi	列出zk的版本信息、主机名称、Java版本、服务器名称等等	参考上面
ruok	测试服务器是否正在运行，如果在运行返回imok，否则返回空	参考上面
srst	重置Zookeeper的所有统计信息	参考上面
wchs	列出watch的总数，连接数	参考上面
wchp	列出所有watch的路径及sessionID	参考上面
mntr	列出集群的关键性能数据，包括zk的版本、node数量、临时节点数等等	参考上面

Kafka

注: 这里机器我只写一个。命令你们也可使用 ./bin/xx.sh (如：./bin/kafka-topics.sh)

查看当前服务器中的所有topic

```
kafka-topics --zookeeper xxxxxx:2181 --list --exclude-internal

说明：

exclude-internal：排除kafka内部topic

比如： --exclude-internal  --topic "test_.*"
```

创建topic

```
kafka-topics --zookeeper xxxxxx:2181  --create
--replication-factor
--partitions 1
--topic topic_name

说明：

--topic 定义topic名

--replication-factor  定义副本数

--partitions  定义分区数
```

删除topic

注意: 需要server.properties中设置delete.topic.enable=true否则只是标记删除

```
kafka-topics --zookeeper xxxxxx:2181 --delete --topic topic_name
```

生产者

```
kafka-console-producer --broker-list xxxxxx:9092 --topic topic_name

可加：--property parse.key=true（有key消息）
```

消费者

```
kafka-console-consumer --bootstrap-server xxxxxx:9092 --topic topic_name

注：可选

--from-beginning：会把主题中以往所有的数据都读取出来

--whitelist '.*' ：消费所有的topic

--property print.key=true：显示key进行消费

--partition 0：指定分区消费

--offset：指定起始偏移量消费
```

查看某个Topic的详情

```
kafka-topics --zookeeper xxxxxx:2181 --describe --topic topic_name
```

修改分区数

```
kafka-topics --zookeeper xxxxxx:2181 --alter --topic topic_name --partitions 6
```

查看某个消费者组信息

```
kafka-consumer-groups --bootstrap-server  xxxxxx:9092  --describe --group
group_name
```

删除消费者组

```
kafka-consumer-groups --bootstrap-server  xxxxxx:9092  ---delete --group
group_name
```

重置offset

```
kafka-consumer-groups --bootstrap-server  xxxxxx:9092  --group group_name

--reset-offsets --all-topics --to-latest --execute
```

leader重新选举

指定Topic指定分区用重新PREFERRED：优先副本策略 进行Leader重选举

```
kafka-leader-election --bootstrap-server xxxxxx:9092
--topic topic_name --election-type PREFERRED --partition 0
```

所有Topic所有分区用重新PREFERRED：优先副本策略 进行Leader重选举

```
kafka-leader-election --bootstrap-server xxxxxx:9092
--election-type preferred  --all-topic-partitions
```

查询kafka版本信息

```
kafka-configs --bootstrap-server xxxxxx:9092
--describe --version
```

增删改配置

功能说明	参数
选择类型	--entity-type (topics/clients/users/brokers/broker- loggers)
类型名称	--entity-name
删除配置	--delete-config k1=v1,k2=v2
添加/修改配置	--add-config k1,k2

topic添加/修改动态配置

```
kafka-configs --bootstrap-server xxxxxx:9092
--alter --entity-type topics --entity-name topic_name
--add-config file.delete.delay.ms=222222,retention.ms=999999
```

topic删除动态配置

```
kafka-configs --bootstrap-server xxxxxx:9092
--alter --entity-type topics --entity-name topic_name
--delete-config file.delete.delay.ms,retention.ms
```

持续批量拉取消息

单次最大消费10条消息(不加参数意为持续消费)

```
kafka-verifiable-consumer --bootstrap-server xxxxxx:9092
--group group_name
--topic topic_name --max-messages 10
```

删除指定分区的消息

删除指定topic的某个分区的消息删除至offset为1024

json文件offset-json-file.json

```
{
    "partitions": [
        {
            "topic": "topic_name",
            "partition": 0,
            "offset": 1024
        }
    ],
    "version": 1
}
```

```
kafka-delete-records --bootstrap-server xxxxxx:9092
--offset-json-file offset-json-file.json
```

查看Broker磁盘信息

查询指定topic磁盘信息

```
kafka-log-dirs --bootstrap-server xxxxxx:9090
--describe --topic-list topic1,topic2
```

查询指定Broker磁盘信息

```
kafka-log-dirs --bootstrap-server xxxxxx:9090
--describe --topic-list topic1 --broker-list 0
```

Hive

启动类

功能说明	命令
启动hiveserver2服务	bin/hiveserver2
启动beeline	bin/beeline
连接hiveserver2	beeline> !connect jdbc:hive2://hadoop102:10000
metastroe服务	bin/hive --service metastore

hive 启动元数据服务（metastore和hiveserver2）和优雅关闭脚本

```
启动： hive.sh start
关闭： hive.sh stop
重启： hive.sh restart
状态： hive.sh status
```

脚本如下

```
#!/bin/bash
HIVE_LOG_DIR=$HIVE_HOME/logs

mkdir -p $HIVE_LOG_DIR

#检查进程是否运行正常，参数1为进程名，参数2为进程端口
function check_process()
{
    pid=$(ps -ef 2>/dev/null | grep -v grep | grep -i $1 | awk '{print $2}')
    ppid=$(netstat -nltp 2>/dev/null | grep $2 | awk '{print $7}' | cut -d '/' -f
1)
    echo $pid
    [[ "$pid" =~ "$ppid" ]] && [ "$ppid" ] && return 0 || return 1
}

function hive_start()
{
    metapid=$(check_process HiveMetastore 9083)
    cmd="nohup hive --service metastore >$HIVE_LOG_DIR/metastore.log 2>&1 &"
    cmd=$cmd" sleep4; hdfs dfsadmin -safemode wait >/dev/null 2>&1"
    [ -z "$metapid" ] && eval $cmd || echo "Metastroe服务已启动"
    server2pid=$(check_process HiveServer2 10000)
    cmd="nohup hive --service hiveserver2 >$HIVE_LOG_DIR/hiveServer2.log 2>&1 &"
    [ -z "$server2pid" ] && eval $cmd || echo "HiveServer2服务已启动"
}

function hive_stop()
{
    metapid=$(check_process HiveMetastore 9083)
    [ "$metapid" ] && kill $metapid || echo "Metastore服务未启动"
    server2pid=$(check_process HiveServer2 10000)
    [ "$server2pid" ] && kill $server2pid || echo "HiveServer2服务未启动"
}

case $1 in
```

```
"start")
    hive_start
    ;;
"stop")
    hive_stop
    ;;
"restart")
    hive_stop
    sleep 2
    hive_start
    ;;
"status")
    check_process HiveMetastore 9083 >/dev/null && echo "Metastore服务运行正常" ||
echo "Metastore服务运行异常"
    check_process HiveServer2 10000 >/dev/null && echo "HiveServer2服务运行正常" ||
echo "HiveServer2服务运行异常"
    ;;
*)
    echo Invalid Args!
    echo 'Usage: '$(basename $0)' start|stop|restart|status'
    ;;
esac
```

常用交互命令

功能说明	命令
不进入hive的交互窗口执行sql	bin/hive -e "sql语句"
执行脚本中sql语句	bin/hive -f hive.sql
退出hive窗口	exit 或 quit
命令窗口中查看hdfs文件系统	dfs -ls /
命令窗口中查看hdfs文件系统	! ls /data/h

SQL类(特殊的)

说明	语句
查看hive中的所有数据库	show databases
用default数据库	use default
查询表结构	desc table_name
查看数据库	show databases
重命名表名	alter table table1 rename to table2
修改表中字段	alter table table_name change name user_name String
修改字段类型	alter table table_name change salary salary Double
创建外部表	create external table ....
查询外部表信息	desc formatted outsidetable
创建视图	create view view_name as select * from table_name .....
添加数据	load data local inpath 'xxx' overwrite into table table_name partition(day='2021-12-01')

内置函数

（1） NVL

给值为NULL的数据赋值，它的格式是NVL(value，default_value)。它的功能是如果value为NULL，则NVL函数返回default_value的值，否则返回value的值，如果两个参数都为NULL ，则返回NULL

```
select nvl(column, 0) from xxx;
```

（2）行转列

函数	描述
CONCAT(string A/col, string B/col...)	返回输入字符串连接后的结果，支持任意个输入字符串
CONCAT_WS(separator, str1, str2,...)	第一个参数参数间的分隔符，如果分隔符是 NULL，返回值也将为 NULL。这个函数会跳过分隔符参数后的任何 NULL 和空字符串。分隔符将被加到被连接的字符串之间。
COLLECT_SET(col)	将某字段的值进行去重汇总，产生array类型字段
COLLECT_LIST(col)	函数只接受基本数据类型，它的主要作用是将某字段的值进行不去重汇总，产生array类型字段。

（3）列转行(一列转多行)

Split(str, separator)： 将字符串按照后面的分隔符切割，转换成字符array。

EXPLODE(col)：
将hive一列中复杂的array或者map结构拆分成多行。

LATERAL VIEW

```
用法：

LATERAL VIEW udtf(expression) tableAlias AS columnAlias
```

解释：lateral view用于和split, explode等UDTF一起使用，它能够将一行数据拆成多行数据，在此基础上可以对拆分后的数据进行聚合。

lateral view首先为原始表的每行调用UDTF，UDTF会把一行拆分成一或者多行，lateral view再把结果组合，产生一个支持别名表的虚拟表。

准备数据源测试

movie	category
《功勋》	记录,剧情
《战狼2》	战争,动作,灾难

SQL

```
SELECT movie,category_name
FROM movie_info
lateral VIEW
explode(split(category,",")) movie_info_tmp  AS category_name ;
```

测试结果

```
《功勋》        记录
《功勋》        剧情
《战狼2》       战争
《战狼2》       动作
《战狼2》       灾难
```

窗口函数

（1）OVER()

定分析函数工作的数据窗口大小，这个数据窗口大小可能会随着行的变而变化。

（2）CURRENT ROW（当前行）

```
n PRECEDING：往前n行数据

n FOLLOWING：往后n行数据
```

（3）UNBOUNDED（无边界）

```
UNBOUNDED PRECEDING 前无边界，表示从前面的起点

UNBOUNDED FOLLOWING后无边界，表示到后面的终点
```

SQL案例：由起点到当前行的聚合

```
select
    sum(money) over(partition by user_id order by pay_time rows between UNBOUNDED
PRECEDING and current row)
from or_order;
```

SQL案例：当前行和前面一行做聚合

```
select
    sum(money) over(partition by user_id order by pay_time rows between 1
PRECEDING and current row)
from or_order;
```

SQL案例：当前行和前面一行和后一行做聚合

```
select
    sum(money) over(partition by user_id order by pay_time rows between 1
PRECEDING AND 1 FOLLOWING )
from or_order;
```

SQL案例：当前行及后面所有行

```
select
    sum(money) over(partition by user_id order by pay_time rows between current
row and UNBOUNDED FOLLOWING  )
from or_order;
```

（4）LAG(col,n,default_val)

往前第n行数据，没有的话default_val

（5）LEAD(col,n, default_val)

往后第n行数据，没有的话default_val

SQL案例：查询用户购买明细以及上次的购买时间和下次购买时间

```
select
    user_id,,pay_time,money,

    lag(pay_time,1,'1970-01-01') over(PARTITION by name order by pay_time)
prev_time,

    lead(pay_time,1,'1970-01-01') over(PARTITION by name order by pay_time)
next_time
from or_order;
```

（6）FIRST_VALUE(col,true/false)

当前窗口下的第一个值，第二个参数为true，跳过空值。

（7）LAST_VALUE (col,true/false)

当前窗口下的最后一个值，第二个参数为true，跳过空值。

SQL案例：查询用户每个月第一次的购买时间 和 每个月的最后一次购买时间

```
select
    FIRST_VALUE(pay_time)
        over(
            partition by user_id,month(pay_time) order by pay_time
            rows between UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING
            ) first_time,

    LAST_VALUE(pay_time)
        over(partition by user_id,month(pay_time) order by pay_time rows between
UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING
        ) last_time
from or_order;
```

（8）NTILE(n)

把有序窗口的行分发到指定数据的组中，各个组有编号，编号从1开始，对于每一行，NTILE返回此行所属的组的编号。（用于将分组数据按照顺序切分成n片，返回当前切片值）

SQL案例：查询前25%时间的订单信息

```
select * from (
    select User_id,pay_time,money,

    ntile(4) over(order by pay_time) sorted

    from or_order
) t
where sorted = 1;
```

4个By

（1）Order By

全局排序，只有一个Reducer。

（2）Sort By

分区内有序。

（3）Distrbute By

类似MR中Partition，进行分区，结合sort by使用。

（4） Cluster By

当Distribute by和Sorts by字段相同时，可以使用Cluster by方式。Cluster by除了具有Distribute by的功能外还兼具Sort by的功能。但是排序只能是升序排序，不能指定排序规则为ASC或者DESC。

在生产环境中Order By用的比较少，容易导致OOM。

在生产环境中Sort By+ Distrbute By用的多。

排序函数

（1）RANK()

排序相同时会重复，总数不会变

```
1
1
3
3
5
```

（2）DENSE_RANK()

排序相同时会重复，总数会减少

```
1
1
2
2
3
```

（3）ROW_NUMBER()

会根据顺序计算

```
1
2
3
4
5
```

日期函数

datediff：返回结束日期减去开始日期的天数

```
datediff(string enddate, string startdate)

select datediff('2021-11-20','2021-11-22')
```

date_add：返回开始日期startdate增加days天后的日期

```
date_add(string startdate, int days)

select date_add('2021-11-20',3)
```

date_sub：返回开始日期startdate减少days天后的日期

```
date_sub (string startdate, int days)

select date_sub('2021-11-22',3)
```

Redis

启动类

key

命令	功能说明
keys *	查看当前库的所有键
exists	判断某个键是否存在
type	查看键的类型
del	删除某个键
expire	为键值设置过期时间，单位秒
ttl	查看还有多久过期,-1表示永不过期,-2表示已过期
dbsize	查看当前数据库中key的数量
flushdb	清空当前库
Flushall	通杀全部库

String

命令	功能说明
get	查询对应键值
set	添加键值对
append	将给定的追加到原值的末尾
strlen	获取值的长度
setnx	只有在key 不存在时设置key的值
incr	将key中存储的数字值增1只能对数字值操作，如果为空，新增值为1
decr	将key中存储的数字值减1只能对数字之操作，如果为空,新增值为-1
incrby /decrby 步长	将key中存储的数字值增减，自定义步长
mset	同时设置一个或多个key-value对
mget	同时获取一个或多个value
msetnx	同时设置一个或多个key-value对，当且仅当所有给定的key都不存在
getrange <起始位置> <结束位置>	获得值的范围,类似java中的substring
setrange <起始位置>	用覆盖所存储的字符串值，从<起始位置>开始
setex <过期时间>	设置键值的同时，设置过去时间，单位秒
getset	以新换旧,设置了新值的同时获取旧值

List

命令	功能说明
lpush/rpush	从左边/右边插入一个或多个值。
lpop/rpop	从左边/右边吐出一个值。值在键在，值光键亡。
rpoplpush	从列表右边吐出一个值，插到列表左边
lrange	按照索引下标获得元素(从左到右)
lindex	按照索引下标获得元素(从左到右)
llen	获得列表长度
linsert before	在的后面插入 插入值
lrem	从左边删除n个value(从左到右)

Set

命令	功能说明
sadd	将一个或多个 member 元素加入到集合 key 当中，已经存在于集合的 member 元素将被忽略。
smembers	取出该集合的所有值。
sismember	判断集合是否为含有该值，有返回1，没有返回0
scard	返回该集合的元素个数。
srem	删除集合中的某个元素。
spop	随机从该集合中吐出一个值。
srandmember	随机从该集合中取出n个值。不会从集合中删除
sinter	返回两个集合的交集元素。
sunion	返回两个集合的并集元素。
sdiff	返回两个集合的差集元素。

Hash

命令	功能说明
hset	给集合中的 键赋值
hget	从集合 取出 value
hmset ...	批量设置hash的值
hexists key	查看哈希表 key 中，给定域 field 是否存在。
hkeys	列出该hash集合的所有field
hvals	列出该hash集合的所有value
hincrby	为哈希表 key 中的域 field 的值加上增量 increment
hsetnx	将哈希表 key 中的域 field 的值设置为 value ， 当且仅当域 field 不存在

zset(Sorted set)

命令	功能说明
zadd ...	将一个或多个 member 元素及其 score 值加入到有序集 key 当中
zrange [WITHSCORES]	返回有序集 key 中，下标在 之间的元素带WITHSCORES，可以让分数一起和值返回到结果集。
zrangebyscore key min max [withscores] [limit offset count]	返回有序集 key 中，所有 score 值介于 min 和 max 之间(包括等于 min 或 max)的成员。有序集成员按 score 值递增(从小到大)次序排列。
zrevrangebyscore key max min [withscores] [limit offset count]	同上，改为从大到小排列。
zincrby	为元素的score加上增量
zrem	删除该集合下，指定值的元素
zcount	统计该集合，分数区间内的元素个数
zrank	返回该值在集合中的排名，从0开始。

Flink

启动

```
./start-cluster.sh
```

run

```
./bin/flink run [OPTIONS]

./bin/flink run -m yarn-cluster -c com.wang.flink.WordCount
/opt/app/WordCount.jar
```

OPTIONS	功能说明
-d	detached 是否使用分离模式
-m	jobmanager 指定提交的jobmanager
-yat	–yarnapplicationType 设置yarn应用的类型
-yD	使用给定属性的值
-yd	–yarndetached 使用yarn分离模式
-yh	–yarnhelp yarn session的帮助
-yid	–yarnapplicationId 挂到正在运行的yarnsession上
-yj	–yarnjar Flink jar文件的路径
-yjm	–yarnjobManagerMemory jobmanager的内存(单位M)
-ynl	–yarnnodeLabel 指定 YARN 应用程序 YARN 节点标签
-ynm	–yarnname 自定义yarn应用名称
-yq	–yarnquery 显示yarn的可用资源
-yqu	–yarnqueue 指定yarn队列
-ys	–yarnslots 指定每个taskmanager的slots数
-yt	yarnship 在指定目录中传输文件
-ytm	–yarntaskManagerMemory 每个taskmanager的内存
-yz	–yarnzookeeperNamespace 用来创建ha的zk子路径的命名空间
-z	–zookeeperNamespace 用来创建ha的zk子路径的命名空间
-p	并行度
-yn	需要分配的YARN容器个数(=任务管理器的数量)

info

```
./bin/flink info [OPTIONS]
```

OPTIONS	功能说明
-c	程序进入点，主类
-p	并行度

list

```
./bin/flink list [OPTIONS]
```

OPTIONS	功能说明
-a	–all 显示所有应用和对应的job id
-r	–running 显示正在运行的应用和job id
-s	–scheduled 显示调度的应用和job id
-m	–jobmanager 指定连接的jobmanager
-yid	–yarnapplicationId 挂到指定的yarn id对应的yarn session上
-z	–zookeeperNamespace 用来创建ha的zk子路径的命名空间

stop

```
./bin/flink stop  [OPTIONS] <Job ID>
```

OPTIONS	功能说明
-d	在采取保存点和停止管道之前，发送MAX_WATERMARK
-p	savepointPath 保存点的路径 'xxxxx'
-m	–jobmanager 指定连接的jobmanager
-yid	–yarnapplicationId 挂到指定的yarn id对应的yarn session上
-z	–zookeeperNamespace 用来创建ha的zk子路径的命名空间

cancel(弱化)

```
./bin/flink cancel  [OPTIONS] <Job ID>
```

OPTIONS	功能说明
-s	使用 "stop "代替
-D	允许指定多个通用配置选项
-m	要连接的JobManager的地址
-yid	–yarnapplicationId 挂到指定的yarn id对应的yarn session上
-z	–zookeeperNamespace 用来创建ha的zk子路径的命名空间

savepoint

```
./bin/flink savepoint  [OPTIONS] <Job ID>
```

OPTIONS	功能说明
-d	要处理的保存点的路径
-j	Flink程序的JAR文件
-m	要连接的JobManager的地址
-yid	–yarnapplicationId 挂到指定的yarn id对应的yarn session上
-z	–zookeeperNamespace 用来创建ha的zk子路径的命名空间