## 优化方案

#### Log4j日志优化

##### 快照目录 dataDir 和 事务日志目录 dataLogDir 分离

写事务日志的目录，需要保证目录空间足够大，并挂载到单独的磁盘上

为了保证数据的一致性，Zookeeper 默认 forceSync 配置为 yes ，

即在返回客户端事务请求响应之前，要将此次请求对应的事务日志刷入到磁盘中，所以事务日志的写入速度，直接决定了 Zookeeper 的吞吐率

##### 1.调整为 DaliyRollingFileAppender，每天滚动创建新的日志文件

vim $ZOOKEEPER\_HOME/conf/log4j.properties

zookeeper.root.logger=INFO, CONSOLE zookeeper.console.threshold=INFO zookeeper.log.dir=. zookeeper.log.file=zookeeper.log zookeeper.log.threshold=DEBUG zookeeper.tracelog.dir=. zookeeper.tracelog.file=zookeeper\_trace.log log4j.appender.ROLLINGFILE=org.apache.log4j.RollingFileAppender log4j.appender.ROLLINGFILE.Threshold=${zookeeper.log.threshold} log4j.appender.ROLLINGFILE.File=${zookeeper.log.dir}/${zookeeper.log.file} log4j.appender.ROLLINGFILE.DatePattern='.'yyyy-MM-dd

##### 2.增加 ZOO\_LOG\_DIR 配置

vim bin/zkServer.sh

ZOO\_LOG\_DIR=$ZOOBINDIR/../log4j

CONSOLE改为ROLLINGFILE

vim bin/zkEnv.sh

# if [ "x${ZOO\_LOG4J\_PROP}" = "x" ]

# then

# ZOO\_LOG4J\_PROP="INFO,CONSOLE"

# fi

if [ "x${ZOO\_LOG4J\_PROP}" = "x" ]

then

ZOO\_LOG4J\_PROP="INFO,ROLLINGFILE"

fi

##### 日志文件自动清除

autopurge.snapRetainCount=3 # 要在dataDir中保留的快照数 autopurge.purgeInterval=24 # 设置日志清除时间间隔；设置为“0”以禁用自动清除功能

#### zooconf配置

###### 限制最大连接数，避免OOM：

globalOutstandingLimit=100000

###### 多少次写事务，生成一个快照如果快照生成频繁可适当调大该参数

preAllocSize=64M 日志文件预分配大小; snapCount=100,000 多少次写事务，生成一个快照如果快照生成频繁可适当调大该参数。

Note：一般zk适用于读写 10：1 如果写很多适用缓存解决

#### **谨慎修改：**

###### 忽略ACL验证，可以减少权限验证的相关操作，提升一点性能。

skipACL：yes

###### 写入磁盘优化

forceSync=no

在不追求强一致性的情况下可以把这个设为no，对写请求的性能提升很有帮助。

forceSync指每次写请求都强制从pagecache固化到磁盘上，才算是写成功返回。

当写请求数量到达一定程度的时候，后续写请求会等待前面写请求的forceSync操作，造成一定延时。

如果追求低延时的写请求，配置forceSync=no，数据写到pagecache后就返回。

但是机器断电的时候，pagecache中的数据有可能丢失。

###### 强一致性场景

* forceSync=yes；fsync.warningthresholdms=50

若为强一致性场景，可以再设置个fsync.warningthresholdms=50， 数据固化到磁盘的操作fsync超过50ms的时候，将会在zookeeper.out中输出一条warn日志（forceSync=yes有效）。