白开水加糖
巧者劳，智者忧，唯无能者无所求。

hadoop参数配置

Hadoop参数汇总

@(hadoop)[配置]

linux参数

以下参数最好优化一下：

1. 文件描述符ulimit -n
2. 用户最大进程 nproc （hbase需要 hbse book）
3. 关闭swap分区
4. 设置合理的预读取缓冲区
5. Linux的内核的IO调度器

JVM参数

JVM方面的优化项Hadoop Performance Tuning Guide

Hadoop参数大全

适用版本：4.3.0

主要配置文件：

- core
- hdfs
- yarn
- mapred

重要性表示如下：

- 重要
- 一般
- 不重要

core-default.xml

- hadoop.common.configuration.version

  配置文件的版本。

- **hadoop.tmp.dir=/tmp/hadoop-${user.name}**

  Hadoop的临时目录，其它目录会基于此路径。本地目录。

  只可以设置一个值；建议设置到一个足够空间的地方，而不是默认的**/tmp**下
  服务端参数，修改需重启

- **hadoop.security.authorization=false**

  是否开启安全服务验证。

  建议不开启。认证操作比较复杂，在公司内部网络下，重要性没那么高

- **io.file.buffer.size=4096**

  在读写文件时使用的缓存大小。这个大小应该是内存Page的倍数。

  建议**1M**

- **io.compression.codecs=null**

  压缩和解压缩编码类列表，用逗号分隔。这些类是使用Java ServiceLoader加载。

- **fs.defaultFS=file:///**

默认文件系统的名称。URI形式。uri's的scheme需要由(fs.SCHEME.impl)指定文件系统实现类。 uri's的authority部分用来指定host, port等。默认是本地文件系统。

HA方式，这里设置服务名，例如：**hdfs://mycluster1**
HDFS的客户端访问HDFS需要此参数。

- **fs.trash.interval=0**

以分钟为单位的垃圾回收时间，垃圾站中数据超过此时间，会被删除。如果是0，垃圾回收机制关闭。可以配置在服务器端和客户端。如果在服务器端配置trash无效，会检查客户端配置。如果服务器端配置有效，客户端配置会忽略。

建议开启，建议**4320**（**3**天）
垃圾回收站，如有同名文件被删除，会给文件顺序编号，例如：a.txt,a.txt(1)

- **fs.trash.checkpoint.interval=0**

以分钟为单位的垃圾回收检查间隔。应该小于或等于fs.trash.interval。如果是0，值等同于fs.trash.interval。每次检查器运行，会创建新的检查点。

建议设置为**60**（**1**小时）

- **dfs.ha.fencing.methods=null**

HDFS的HA功能的防脑裂方法。可以是内建的方法(例如shell和sshfence)或者用户定义的方法。建议使用*sshfence(hadoop:9922)*，括号内的是用户名和端口，注意，这需要*NN*的*2*台机器之间能够免密码登陆

fences是防止脑裂的方法，保证NN中仅一个是Active的，如果2者都是Active的，新的会把旧的强制Kill。

- **dfs.ha.fencing.ssh.private-key-files=null**

使用sshfence时，SSH的私钥文件。 使用了**sshfence**，这个必须指定

- **ha.zookeeper.quorum=null**

Ha功能，需要一组zk地址，用逗号分隔。被ZKFailoverController使用于自动失效备援failover。

- **ha.zookeeper.session-timeout.ms=5000**

ZK连接超时。ZKFC连接ZK时用。设置一个小值可以更快的探测到服务器崩溃（crash),但也会更频繁的触发失效备援，

在传输错误或者网络不畅时。建议**10s-30s**

- **hadoop.http.staticuser.user=dr.who**

  在网页界面访问数据使用的用户名。默认值是一个不真实存在的用户，此用户权限很小，不能访问不同用户的数据。这保证了数据安全。也可以设置为*hdfs*和*hadoop*等具有较高权限的用户，但会导致能够登陆网页界面的人能看到其它用户数据。实际设置请综合考虑。如无特殊需求。使用默认值就好

- **fs.permissions.umask-mode=22**

  在创建文件和目录时使用此umask值（用户掩码）。类linux上的文件权限掩码。可以使用8进制数字也可以使用符号，例如："022" (8进制，等同于以符号表示的u=rwx,g=r-x,o=r-x)，或者"u=rwx,g=rwx,o="(符号法，等同于8进制的007)。注意，*8*进制的掩码，和实际权限设置值正好相反，建议使用符号表示法，描述更清晰

- io.native.lib.available=true

  是否启动Hadoop的本地库，默认启用。本地库可以加快基本操作，例如IO，压缩等。

- hadoop.http.filter.initializers=org.apache.hadoop.http.lib.StaticUserWebFilter

  Hadoop的Http服务中，用逗号分隔的一组过滤器类名，每个类必须扩展自org.apache.hadoop.http.FilterInitializer。 这些组件被初始化，应用于全部用户的JSP和Servlet页面。 列表中定义的顺序就是过滤器被调用的顺序。

- hadoop.security.authentication

  安全验证规则，可以是simple和kerberos。simple意味着不验证。

- hadoop.security.group.mapping=org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback

  user到group的映射类。ACL用它以给定user获取group。默认实现是 org.apache.hadoop.security.JniBasedUnixGroups MappingWithFallback, 如果JNI有效，它将发挥作用，使用Hadoop的API去获取user的groups列表。如果JNI无效，会使用另一个基于shell的实现, ShellBasedUnixGroupsMapping。这个实现是基于Linux、Unix的shell的环境。

- *hadoop.security.groups.cache.secs=300*

  user到gourp映射缓存的有效时间。如果超时，会再次调用去获取新的映射关系然后缓存起来。

- *hadoop.security.service.user.name.key=null*

  如果相同的RPC协议被多个Server实现，这个配置是用来指定在客户端进行RPC调用时，使用哪个principal name去联系服务器。不建议使用

- *hadoop.security.uid.cache.secs=14400*

安全选项。不建议使用

- *hadoop.rpc.protection=authentication*

  rpc连接保护。可取的值有authentication（认证）, integrity（完整） and privacy（隐私）。不建议使用

- hadoop.work.around.non.threadsafe.getpwuid=false

  一些系统已知在调用getpwuid_r和getpwgid_r有问题，这些调用是非线程安全的。这个问题的主要表现特征是JVM崩溃。如果你的系统有这些问题，开启这个选项。默认是关闭的。

- *hadoop.kerberos.kinit.command=kinit*

  用来定期的向Hadoop提供新的Kerberos证书。所提供命令需要能够在运行Hadoop客户端的用户路径中查找到，否则，请指定绝对路径。不建议使用

- *hadoop.security.auth_to_local=null*

  映射kerberos principals（代理人）到本地用户名

- io.bytes.per.checksum=512

  每次进行校验和检查的字节数。一定不能大于io.file.buffer.size.

- io.skip.checksum.errors=FALSE

  是否跳过校验和错误，默认是否，校验和异常时会抛出错误。

- io.serializations=org.apache.hadoop.io.serializer.WritableSerialization,org.apache.hadoop.io.serializer.avro.AvroSpecificSerialization,org.apache.hadoop.io.serializer.avro.AvroReflectSerialization

  序列化类列表，可以被用来获取序列化器和反序列化器（serializers and deserializers）。

- io.seqfile.local.dir=${hadoop.tmp.dir}/io/local

  本地文件目录。sequence file在merge过程中存储内部数据的地方。可以是逗号分隔的一组目录。最好在不同磁盘以分散IO。实际不存在的目录会被忽略。

- io.map.index.skip=0

  跳过的索引实体数量在entry之间。默认是0。设置大于0的值可以用更少的内存打开大MapFiles。注意：**MpaFile**是一组**Sequence**文件，是排序后的，带内部索引的文件

- io.map.index.interval=128

  MapFile包含两个文件，数据文件和索引文件。每io.map.index.interval个记录写入数据文件，一条记录(行key，数据文件位置)写入索引文件。

- fs.default.name=file:///

  过时。使用(fs.defaultFS)代替

- *fs.AbstractFileSystem.file.impl=org.apache.hadoop.fs.local.LocalFs*

  文件系统实现类：file

- fs.AbstractFileSystem.hdfs.impl=org.apache.hadoop.fs.Hdfs

  文件系统实现类：hdfs

- fs.AbstractFileSystem.viewfs.impl=org.apache.hadoop.fs.viewfs.ViewFs

  文件系统实现类：viewfs (例如客户端挂载表)。

  在实现**federation**特性时，客户端可以部署此系统，方便同时访问多个**nameservice**

- *fs.ftp.host=0.0.0.0*

  非Hdfs文件系统设置。暂不关注

- *fs.ftp.host.port=21*

  非Hdfs文件系统设置。暂不关注

- fs.df.interval=60000

  磁盘使用统计刷新间隔，以毫秒为单位

- *fs.s3.block.size=67108864*

  非Hdfs文件系统设置。暂不关注

- *fs.s3.buffer.dir=${hadoop.tmp.dir}/s3*

非Hdfs文件系统设置。暂不关注

- *fs.s3.maxRetries=4*

  非Hdfs文件系统设置。暂不关注

- *fs.s3.sleepTimeSeconds=10*

  非Hdfs文件系统设置。暂不关注

- fs.automatic.close=true

  默认的，文件系统实例在程序退出时自动关闭，通过JVM shutdown hook方式。可以把此属性设置为false取消这种操作。这是一个高级选项，需要使用者特别关注关闭顺序。不要关闭

- *fs.s3n.block.size=67108864*

  非Hdfs文件系统设置。暂不关注

- io.seqfile.compress.blocksize=1000000

  SequenceFiles以块压缩方式压缩时，块大小大于此值时才启动压缩。

- io.seqfile.lazydecompress=TRUE

  懒惰解压，仅在必要时解压，仅对块压缩的SequenceFiles有效。

- io.seqfile.sorter.recordlimit=1000000

  在SequenceFiles.Sorter spill过程中，保存在内存中的记录数

- io.mapfile.bloom.size=1048576

  在BloomMapFile使用的布隆过滤器内存大小。

- io.mapfile.bloom.error.rate=0.005

  BloomMapFile中使用布隆过滤器失败比率. 如果减少这个值，使用的内存会成指数增长。

- hadoop.util.hash.type=murmur

默认Hash算法实现. 'murmur':MurmurHash, 'jenkins':JenkinsHash.

- ipc.client.idlethreshold=4000

  连接数阀值，超过此值，需要进行空闲连接检查

- ipc.client.kill.max=10

  定义客户端最大数量，超过会被断开连接

- ipc.client.connection.maxidletime=10000

  毫秒，最大时间，超过后客户端会断开和服务器的连接。

- ipc.client.connect.max.retries=10

  客户端连接重试次数。

- ipc.client.connect.max.retries.on.timeouts=45

  在连接超时后，客户端连接重试次数

- ipc.server.listen.queue.size=128

  定义服务器端接收客户端连接的监听队列长度

- ipc.server.tcpnodelay=false

  在服务器端开启/关闭Nagle's算法，此算法可以延迟小数据包发送，从而达到网络流量更有效利用。但是这对小数据包是不利的。默认关闭。建议*false*，即开启*Nagle*算法

- ipc.client.tcpnodelay=false

  参考ipc.server.tcpnodelay，客户端参数。或许可以考虑关闭*Nagle*算法，增加客户端响应速度

- hadoop.rpc.socket.factory.class.default=org.apache.hadoop.net.StandardSocketFactory

  高级选项，暂不考虑

- hadoop.rpc.socket.factory.class.ClientProtocol=null

  高级选项，暂不考虑

- hadoop.socks.server=null

  高级选项，暂不考虑

- net.topology.node.switch.mapping.impl=org.apache.hadoop.net.ScriptBasedMapping

  机架感知实现类。

- net.topology.script.file.name=null

  配合ScriptBasedMapping使用。脚本文件。此脚本文件，输入是ip地址，输出是机架路径。

- net.topology.script.number.args=100

  机架感知脚本文件的参数最大数量。脚本每次运行被传递的参数，每个参数是一个ip地址

- net.topology.table.file.name=null

  在net.topology.script.file.name被设置为 org.apache.hadoop.net.TableMapping时，可以使用此配置。文件格式是一个有两个列的文本文件，使用空白字符分隔。第一列是DNS或IP地址，第二列是机架路径。如无指定，使用默认机架（/default-rack）

- *file.stream-buffer-size=4096*

  非hdfs文件系统，暂不关注

- *s3.stream-buffer-size=4096*

  非hdfs文件系统，暂不关注

- *kfs.stream-buffer-size=4096*

  非hdfs文件系统，暂不关注

- *ftp.stream-buffer-size=4096*

  非hdfs文件系统，暂不关注

- *tfile.io.chunk.size=1048576*

  非hdfs文件系统，暂不关注

- hadoop.http.authentication.type=simple

  Oozie Http终端安全验证。可选值：simple | kerberos |#AUTHENTICATION_HANDLER_CLASSNAME#

  建议**simple**，关闭验证

- *hadoop.http.authentication.token.validity=36000*

  安全选项。暂不关注

- *hadoop.http.authentication.signature.secret.file=${user.home}/hadoop-http-auth-signature-secret*

  安全选项。暂不关注

- *hadoop.http.authentication.cookie.domain=null*

  安全选项。暂不关注

- *hadoop.http.authentication.simple.anonymous.allowed=TRUE*

  安全选项。暂不关注

- *hadoop.http.authentication.kerberos.principal=HTTP/_HOST@LOCALHOST*

  安全选项。暂不关注

- *hadoop.http.authentication.kerberos.keytab=${user.home}/hadoop.keytab*

  安全选项。暂不关注

- dfs.ha.fencing.ssh.connect-timeout=30000

  SSH连接超时，毫秒，仅适用于内建的sshfence fencer。

- ha.zookeeper.parent-znode=/hadoop-ha

  ZK失效备援功能，需要在ZK上创建节点，这里是根节点的名称。ZKFC会在这下面工作。注意，NameService ID会 被写到此节点下，所以即便是开启federation功能，也仅需要指定一个值。

- ha.zookeeper.acl=world:anyone:rwcda

ZKFC创建的ZK节点的访问控制权限设置。可以多个，逗号分隔。此设置和ZK的CLI使用相同的格式。

- ha.zookeeper.auth=null

  ZK操作时的权限验证。

- *hadoop.ssl.keystores.factory.class=org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory*

  安全选项。暂不关注

- *hadoop.ssl.require.client.cert=FALSE*

  安全选项。暂不关注

- *hadoop.ssl.hostname.verifier=DEFAULT*

  安全选项。暂不关注

- *hadoop.ssl.server.conf=ssl-server.xml*

  安全选项。暂不关注

- *hadoop.ssl.client.conf=ssl-client.xml*

  安全选项。暂不关注

- *hadoop.ssl.enabled=FALSE*

  安全选项。暂不关注

- *hadoop.jetty.logs.serve.aliases=TRUE*

  是否允许在Jetty中使用别名服务。

- ha.health-monitor.connect-retry-interval.ms=1000

  HA功能的健康监控连接重试间隔

- ha.health-monitor.check-interval.ms=1000

  HA功能的健康监控连接间隔

- ha.health-monitor.sleep-after-disconnect.ms=1000

  HA功能的健康监控，在因网络问题失去连接后休眠多久。用于避免立即重试，此时网络问题仍在，没有意义

- ha.health-monitor.rpc-timeout.ms=45000

  HA功能健康监控的超时时间，毫秒

- ha.failover-controller.new-active.rpc-timeout.ms=60000

  FC等待新的NN变成active状态的超时时间。

- ha.failover-controller.graceful-fence.rpc-timeout.ms=5000

  FC等待旧的active变成standby的超时时间。

- ha.failover-controller.graceful-fence.connection.retries=1

  FC在做完美隔离是的连接重试次数（graceful fencing）

- ha.failover-controller.cli-check.rpc-timeout.ms=20000

  手动运行的FC功能（从CLI）等待健康检查、服务状态的超时时间。

hdfs-default.xml

- hadoop.hdfs.configuration.version=1

  配置文件的版本

- **dfs.datanode.address=0.0.0.0:50010**

  DN服务地址和端口，用于数据传输。0表示任意空闲端口。

  xferPort dfs.datanode.address 50010 数据流地址　数据传输 infoPort　dfs.datanode.http.address 50075 ipcPort　dfs.datanode.ipc.address 50020 命令

- **dfs.datanode.http.address=0.0.0.0:50075**

  DN的HTTP服务地址和端口。0表示任意空闲端口。

- **dfs.datanode.ipc.address=0.0.0.0:50020**

  DN的IPC地址和端口。0表示任意空闲端口。

- **dfs.namenode.rpc-address=0.0.0.0:50090**

  NN的RPC地址和端口

- **dfs.namenode.http-address=0.0.0.0:50070**

  NN的HTTP地址和端口。0表示任意空闲端口。

- **dfs.datanode.du.reserved=0**

  每个磁盘（volume）的保留空间，字节。要注意留足够的空间给非HDFS文件使用。建议保留磁盘容量的*5%*或者*50G*以上

- **dfs.namenode.name.dir.restore=FALSE**

  设置为true，允许NN尝试恢复之前失败的dfs.namenode.name.dir目录。在创建checkpoint是做此尝试。如果设置多个磁盘，建议允许

- **dfs.namenode.edits.dir=${dfs.namenode.name.dir}**

  本地文件，NN存放edits文件的目录。可以是逗号分隔的目录列表。edits文件会存储在每个目录，冗余安全。

- **dfs.namenode.shared.edits.dir=null**

  在多个NN中共享存储目录，用于存放edits文件。这个目录，由active写，由standby读，以保持命名空间数据一致。此目录不需要是dfs.namenode.edits.dir中列出的。在非HA集群中，它不会使用。建议使用*qj*方式，可以不关注这个选项

- **dfs.namenode.edits.journal-plugin.qjournal=org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager**

  qj方式共享edits。建议使用此方式

- **dfs.permissions.enabled=true**

  是否在HDFS中开启权限检查。

- **dfs.permissions.superusergroup=supergroup**

  超级用户组。仅能设置一个。

- **dfs.datanode.data.dir=file://${hadoop.tmp.dir}/dfs/data**

  本地磁盘目录，HDFS数据应该存储Block的地方。可以是逗号分隔的目录列表（典型的，每个目录在不同的磁盘）。这

些目录被轮流使用，一个块存储在这个目录，下一个块存储在下一个目录，依次循环。每个块在同一个机器上仅存储一份。不存在的目录被忽略。必须创建文件夹，否则被视为不存在。

- **dfs.replication=3**

  数据块副本数。此值可以在创建文件是设定，客户端可以只有设定，也可以在命令行修改。不同文件可以有不同的副本数。默认值用于未指定时。

- **dfs.replication.max=512**

  最大块副本数，不要大于节点总数。

- **dfs.namenode.replication.min=1**

  最小块副本数。在上传文件时，达到最小副本数，就认为上传是成功的

- **dfs.blocksize=67108864**

  块大小，字节。可以使用后缀: k(kilo), m(mega), g(giga), t(tera), p(peta), e(exa)指定大小 (就像128k, 512m, 1g, 等待)。

- **dfs.client.block.write.retries=3**

  客户端写数据到DN时，最大重试次数。超过重试次数就会报出错误。

- **dfs.client.block.write.replace-datanode-on-failure.enable=true**

  在进行pipeline写数据（上传数据的方式）时，如果DN或者磁盘故障，客户端将尝试移除失败的DN，然后写到剩下的磁盘。一个结果是，pipeline中的DN减少了。这个特性是添加新的DN到pipeline。这是一个站点范围的选项。当集群规模非常小时，例如3个或者更小，集群管理者可能想要禁止掉此特性。

- **dfs.client.block.write.replace-datanode-on-failure.policy=DEFAULT**

  此属性仅在dfs.client.block.write.replace-datanode-on-failure.enable设置为true时有效。

  - ALWAYS: 总是添加新的DN
    - NEVER: 从不添加新的DN
    - DEFAULT: 设r是副本数，n是要写的DN数。在r>=3并且floor(r/2)>=n或者r>n(前提是文件是hflushed/appended)时添加新的DN。

- **dfs.heartbeat.interval=3**

  DN的心跳间隔，秒

- **dfs.namenode.handler.count=10**

  NN的服务线程数。用于处理RPC请求。

- **dfs.namenode.safemode.threshold-pct=0.999f**

  数据进入安全模式阀值，百分比，float形式，数据块达到最小副本数（dfs.namenode.replication.min）的百分比。值小于等于0意味着在退出安全模式前不等待数据修复。大于1的值将导致无法离开安全模式。

- **dfs.namenode.safemode.extension=30000**

  安全模式扩展存在时间，在需要的阀值达到后，毫秒。可以设置为*0*，或者比较短的一个时间，例如*3*秒

- **dfs.datanode.balance.bandwidthPerSec=1048576**

  在做数据平衡时，每个DN最大带宽占用，每秒字节。默认值是1M。建议可以到*10M*

- **dfs.hosts=null**

  文件名，包含了一个host列表，允许列表内机器连到NN。必须指定完整路径。如果值为空，全部hosts都允许连入。

- **dfs.hosts.exclude=null**

  文件名，包含了一个hosts列表，不允许列表内机器连到NN。必须指定完整路径。如果值为空。没有host被禁止。如果上述*2*个都设置并且有重合，*dfs.hosts*中优先级高。

- **dfs.stream-buffer-size=4096**

  文件流缓存大小。需要是硬件page大小的整数倍。在读写操作时，数据缓存大小。注意和*core-default.xml*中指定文件类型的缓存是不同的，这个是*dfs*共用的

- **dfs.namenode.num.extra.edits.retained=1000000**

  除最小的必须的editlog之外，额外保留的editlog文件数量。这是有用的，可以用于审核目的，或者HA设置一个远程Standby节点并且有时可能离线时，都需要保留一个较长的backlog。

  典型的，每个edit大约几百字节，默认的1百万editlog大约有百兆到1G。注意：早先的extra edits文件可能操作这里设置的值，因为还有其它选项，例如dfs.namenode.max.extra.edits.segments.retained

  建议值：**2200**，约**3**天的

- **dfs.datanode.handler.count=10**

DN的服务线程数。这些线程仅用于接收请求，处理业务命令

- **dfs.datanode.failed.volumes.tolerated=0**

  可以接受的卷的失败数量。默认值0表示，任一个卷失败都会导致DN关闭。

  建议设置此值，避免个别磁盘问题。如果此值超过真实磁盘数，将会报错，启动失败

- **dfs.namenode.support.allow.format=true**

  NN是否允许被格式化？在生产系统，把它设置为false，阻止任何格式化操作在一个运行的DFS上。

  建议初次格式化后，修改配置禁止

- **dfs.client.failover.max.attempts=15**

  专家设置。客户端失败重试次数。

- **dfs.client.failover.connection.retries=0**

  专家设置。IPC客户端失败重试次数。在网络不稳定时建议加大此值

- **dfs.client.failover.connection.retries.on.timeouts=0**

  专家设置。IPC客户端失败重试次数，此失败仅指超时失败。在网络不稳定时建议加大此值

- **dfs.nameservices=null**

  nameservices列表。逗号分隔。

  我们常用的仅配置一个，启动**federation**功能需要配置多个

- **dfs.nameservice.id=null**

  nameservice id，如果没有配置或者配置多个，由匹配到的本地节点地址配置的IP地址决定。我们进配置一个*NS*的情况下，建议这里不配置

- **dfs.ha.namenodes.EXAMPLENAMESERVICE=null**

包含一个NN列表。EXAMPLENAMESERVICE是指具体的nameservice名称，通常就是dfs.nameservices中配置的。值是预备配置的NN的ID。

ID是自己取的，不重复就可以，例如**nn1,nn2**

- **dfs.ha.namenode.id=null**

NN的ID，如果没有配置，由系统决定。通过匹配本地节点地址和配置的地址。

这里设置的是本机的**NN**的**ID**（此配置仅对**NN**生效），由于要配置**2**个**NN**，建议没有特殊需要，这里不进行配置

- **dfs.ha.automatic-failover.enabled=FALSE**

是否开启自动故障转移。建议开启，*true*

- **dfs.namenode.avoid.write.stale.datanode=FALSE**

决定是否避开在脏DN上写数据。写操作将会避开脏DN，除非超过一个配置的比率 (dfs.namenode.write.stale.datanode.ratio)。

尝试开启

- **dfs.journalnode.rpc-address=0.0.0.0:8485**

JournalNode RPC服务地址和端口

- **dfs.journalnode.http-address=0.0.0.0:8480**

JournalNode的HTTP地址和端口。端口设置为0表示随机选择。

- **dfs.namenode.audit.loggers=default**

审查日志的实现类列表，能够接收audit事件。它们需要实现 org.apache.hadoop.hdfs.server.namenode.AuditLogger接口。默认值"default"可以用于引用默认的audit logger， 它使用配置的日志系统。安装客户自己的audit loggers可能影响NN的稳定性和性能。

建议**default**，开启

- dfs.client.socket-timeout=60*1000

- dfs.datanode.socket.write.timeout=8*60*1000
- dfs.datanode.socket.reuse.keepalive=1000
- dfs.namenode.logging.level=info

  DFS的NN的日志等级。值可以是：info，dir(跟踪命名空间变动)，"block" (跟踪块的创建删除，replication变动)，或者"all".

- dfs.namenode.secondary.http-address=0.0.0.0:50090

  SNN的http服务地址。如果是0，服务将随机选择一个空闲端口。使用了*HA*后，就不再使用*SNN*了

- dfs.https.enable=FALSE

  允许HDFS支持HTTPS(SSL)。建议不支持

- dfs.client.https.need-auth=FALSE

  安全选项，暂不关注

- dfs.https.server.keystore.resource=ssl-server.xml

  安全选项，暂不关注

- dfs.client.https.keystore.resource=ssl-client.xml

  安全选项，暂不关注

- dfs.datanode.https.address=0.0.0.0:50475

  安全选项，暂不关注

- dfs.namenode.https-address=0.0.0.0:50470

  安全选项，暂不关注

- dfs.datanode.dns.interface=default

  DN汇报它的IP地址的网卡。我们给*DN*指定了*0.0.0.0*之类的地址，这个地址需要被解析成对外地址，这里指定的是网卡名，即那个网卡上绑定的*IP*是可以对外的*IP*，一般的，默认值就足够了

- dfs.datanode.dns.nameserver=default

DNS的域名或者IP地址。DN用它来确定自己的域名，在对外联系和显示时调用。一般的，默认值就足够了

- dfs.namenode.backup.address=0.0.0.0:50100

  NN的BK节点地址和端口，0表示随机选用。使用*HA*，就不需要关注此选项了。建议不使用*BK*节点

- dfs.namenode.backup.http-address=0.0.0.0:50105

  使用*HA*，就不需要关注此选项了。建议不使用*BK*节点

- dfs.namenode.replication.considerLoad=true

  设定在选择存放目标时是否考虑负载。需要

- dfs.default.chunk.view.size=32768

  在浏览器中查看一个文件时，可以看到的字节数。

- dfs.namenode.name.dir=file://${hadoop.tmp.dir}/dfs/name

  本地磁盘目录，NN存储fsimage文件的地方。可以是按逗号分隔的目录列表，fsimage文件会存储在全部目录，冗余安全。这里多个目录设定，最好在多个磁盘，另外，如果其中一个磁盘故障，不会导致系统故障，会跳过坏磁盘。由于使用了*HA*，建议仅设置一个。如果特别在意安全，可以设置*2*个

- dfs.namenode.fs-limits.max-component-length=0

  路径中每个部分的最大字节长度（目录名，文件名的长度）。0表示不检查长度。长文件名影响性能

- dfs.namenode.fs-limits.max-directory-items=0

  设置每个目录最多拥有多少个子目录或者文件。0表示无限制。同一目录下子文件和目录多影响性能

- dfs.namenode.fs-limits.min-block-size=1048576

  最小的Block大小，字节。在NN创建时强制验证。避免用户设定过小的Block Size，导致过多的Block，这非常影响性能。

- dfs.namenode.fs-limits.max-blocks-per-file=1048576

  每个文件最大的Block数。在NN写时强制检查。用于防止创建超大文件。

dfs.block.access.token.enable=FALSE

访问DN时是否验证访问令牌。建议*false*，不检查

- dfs.block.access.key.update.interval=600

安全选项，暂不关注

- dfs.block.access.token.lifetime=600

安全选项，暂不关注

- dfs.datanode.data.dir.perm=700

本地数据目录权限设定。8进制或者符号方式都可以。

- dfs.blockreport.intervalMsec=21600000

数据块汇报间隔，毫秒，默认是6小时。

- dfs.blockreport.initialDelay=0

第一次数据块汇报时延迟，秒。目的是减轻*NN*压力？

- dfs.datanode.directoryscan.interval=21600

DN的数据块扫描间隔，秒。磁盘上数据和内存中数据调整一致。

- dfs.datanode.directoryscan.threads=1

线程池要有多少线程用来并发的压缩磁盘的汇报数据。

- dfs.namenode.safemode.min.datanodes=0

NN收到回报的DN的数量的最小值，达不到此值，NN不退出安全模式。（在系统启动时发生作用）。<=0的值表示不关心DN数量，在启动时。大于DN实际数量的值会导致无法离开安全模式。建议不设置此值

- dfs.namenode.max.objects=0

DFS支持的最大文件、目录、数据块数量。0无限制。

- dfs.namenode.decommission.interval=30

NN周期性检查退役是否完成的间隔，秒。

- dfs.namenode.decommission.nodes.per.interval=5

  NN检查退役是否完成，每dfs.namenode.decommission.interval秒检查的节点数量。

- dfs.namenode.replication.interval=3

  NN周期性计算DN的副本情况的频率，秒。

- dfs.namenode.accesstime.precision=3600000

  HDFS文件的访问时间精确到此值，默认是1小时。0表示禁用访问时间。

- dfs.datanode.plugins=null

  DN上的插件列表，逗号分隔。

- dfs.namenode.plugins=null

  NN上的插件列表，逗号分隔。

- dfs.bytes-per-checksum=512

  每次计算校验和的字节数。一定不能大于dfs.stream-buffer-size。

- dfs.client-write-packet-size=65536

  客户端写数据时的包的大小。包是块中的更小单位数据集合

- dfs.client.write.exclude.nodes.cache.expiry.interval.millis=600000

  最大周期去让DN保持在例外节点队列中。毫秒。操过此周期，先前被排除的DN将被移除缓存并被尝试再次申请Block。默认为10分钟。

- dfs.namenode.checkpoint.dir=file://${hadoop.tmp.dir}/dfs/namesecondary

  本地文件系统中，DFS SNN应该在哪里存放临时[用于合并|合并后]（to merge）的Image。如果是逗号分隔的目录列表，Image文件存放多份。冗余备份。建议不使用*SNN*功能，忽略此配置

- dfs.namenode.checkpoint.edits.dir=${dfs.namenode.checkpoint.dir}

建议不使用*SNN*功能，忽略此配置

- dfs.namenode.checkpoint.period=3600

  建议不使用*SNN*功能，忽略此配置

- dfs.namenode.checkpoint.txns=1000000

  建议不使用*SNN*功能，忽略此配置

- dfs.namenode.checkpoint.check.period=60

  建议不使用*SNN*功能，忽略此配置

- dfs.namenode.checkpoint.max-retries=3

  建议不使用*SNN*功能，忽略此配置

- dfs.namenode.num.checkpoints.retained=2
  建议不使用*SNN*功能，忽略此配置

- dfs.namenode.num.extra.edits.retained=1000000

  数量限制，额外的edits事务数。

- dfs.namenode.max.extra.edits.segments.retained=10000

  extra edit日志文件segments的最大数量。除了用于NN重启时的最小edits文件之外。一个*segments*包含多个日志文件

- dfs.namenode.delegation.key.update-interval=86400000

  NN中更新主代理令牌的时间间隔，毫秒。安全选项，不关注

- dfs.namenode.delegation.token.max-lifetime=604800000

  NN中更新主代理令牌的时间间隔，毫秒。安全选项，不关注

- dfs.namenode.delegation.token.renew-interval=86400000

  NN中更新主代理令牌的时间间隔，毫秒。安全选项，不关注

dfs.image.compress=FALSE

Image文件要压缩吗？

- dfs.image.compression.codec=org.apache.hadoop.io.compress.DefaultCodec

Image文件压缩编码。必须是在io.compression.codecs中定义的编码。

- dfs.image.transfer.timeout=600000

Image文件传输时超时。*HA*方式使用不到，可不关注

- dfs.image.transfer.bandwidthPerSec=0

Image文件传输时可以使用的最大带宽，秒字节。0表示没有限制。*HA*方式使用不到，可不关注

- dfs.datanode.max.transfer.threads=4096

= 旧参数 dfs.datanode.max.xcievers
DN上传送数据出入的最大线程数。

- dfs.datanode.readahead.bytes=4193404

预读磁盘数据。如果Hadoop本地库生效，DN可以调用posix_fadvise系统获取页面数据到操作系统的缓存中。这个配置指定读取当前读取位置之前的字节数。设置为0，取消此功能。无本地库，此功能也无效。？

- dfs.datanode.drop.cache.behind.reads=FALSE

在有些场景下，特别是对一些大的，并且不可能重用的数据，缓存在操作系统的缓存区是无用的。此时，DN可以配置自动清理缓存区数据，在已经发生向客户端之后。此功能自动失效，在读取小数据片时。(例如HBase的随机读写场景）。通过释放缓存，这在某些场景下可以提高性能。Hadoop本地库无效，此功能无效。看起来是一个可以尝试的特性

- dfs.datanode.drop.cache.behind.writes=FALSE

同dfs.datanode.drop.cache.behind.reads相似。

- dfs.datanode.sync.behind.writes=FALSE

如果是true，在写之后，DN将指示操作系统把队列中数据全部立即写磁盘。和常用的OS策略不同，它们可能在触发写磁盘之前等待30秒。Hadoop本地库无效，此功能无效。

- dfs.client.failover.sleep.base.millis=500

专家设置。失败重试间的等待时间，毫秒。这里的值是个基本值，实际值会根据失败/成功次数递增/递减50%。第一次失败会立即重试。第二次将延迟至少dfs.client.failover.sleep.base.millis毫秒。依次类推。

- dfs.client.failover.sleep.max.millis=15000

  专家设置。失败重试见的等待时间最大值，毫秒。

- dfs.ha.log-roll.period=120

  StandbyNode要求Active滚动EditLog，由于StandBy只能从已经完成的Log Segments中读，所以Standby上的数据新鲜程度依赖于以如何的频率滚动日志。秒。另外，故障转移也会触发一次日志滚动，所以StandbyNode在Active之前，数据也会更新成最新的。秒，默认是2分钟。

- dfs.ha.tail-edits.period=60

  StandbyNode以此频率检测共享目录中最新的日志，秒。

- dfs.ha.zkfc.port=8019

  zkfc的rpc端口

- dfs.support.append=TRUE

  是否允许append。

- dfs.client.use.datanode.hostname=FALSE

  是否客户端应该使用DN的HostName，在连接DN时，默认是使用IP。

- dfs.datanode.use.datanode.hostname=FALSE

  是否DN应该使用HostName连接其它DN，在数据传输时。默认是是IP。

- dfs.client.local.interfaces=null

  逗号分隔的网卡列表，用于在客户端和DN之间传输数据时。当创建连接时，客户端随机选择一个并绑定它的socket到这个网卡的IP上。名字可以以网卡名(例如 "eth0"), 子网卡名 (eg "eth0:0"), 或者IP地址(which may be specified using CIDR notation to match a range of IPs)。

- dfs.namenode.kerberos.internal.spnego.principal=${dfs.web.authentication.kerberos.principal}

  安全选项，暂不关注

- dfs.secondary.namenode.kerberos.internal.spnego.principal=${dfs.web.authentication.kerberos.principal}

  安全选项，暂不关注

- dfs.namenode.avoid.read.stale.datanode=FALSE

  决定是否避开从脏DN上读数据。脏DN指在一个指定的时间间隔内没有收到心跳信息。脏DN将被移到可以读取节点列表的尾端。尝试开启

- dfs.namenode.stale.datanode.interval=30000

  标记一个DN是脏的时间间隔。例如，如果NN在此设定的时间内没有接收到来自某一个节点的心跳信息，此DN将被标记为脏的。此间隔不能太小，否则容易导致被频繁的标记为脏DN。

  我们建议是**1**分钟

- dfs.namenode.write.stale.datanode.ratio=0.5f

  当全部DN被标记为脏DN的比率高于此阀值，停止不写数据到脏DN的策略，以免造成热点问题（有效的，可写的DN太少，压力太大）。

- dfs.namenode.invalidate.work.pct.per.iteration=0.32f

  高级属性。改变需小心。

- dfs.namenode.replication.work.multiplier.per.iteration=2

  高级属性。改变需小心。

- dfs.webhdfs.enabled=FALSE

  在NN和DN上开启WebHDFS (REST API)功能。

  可以开启尝试

- hadoop.fuse.connection.timeout=300

  秒，在fuse_dfs中缓存libhdfs连接对象的超时时间。 小值使用内存小。大值可以加快访问，通过避开创建新的连接对象。

- hadoop.fuse.timer.period=5

秒

- dfs.metrics.percentiles.intervals=null

Comma-delimited set of integers denoting the desired rollover intervals (in seconds) for percentile latency metrics on the Namenode and Datanode. By default, percentile latency metrics are disabled.

- dfs.encrypt.data.transfer=FALSE

是否加密传输数据？仅需要配置在NN和DN。客户端可以自行判断。

- dfs.encrypt.data.transfer.algorithm=null

可以设置为"3des"或"rc4"。否则使用默认的，通常是usually 3DES。3DES更安全，RC4更快。

- dfs.datanode.hdfs-blocks-metadata.enabled=TRUE

布尔值，设定后台DN端是否支持DistributedFileSystem#getFileVBlockStorageLocations API。

- dfs.client.file-block-storage-locations.num-threads=10

在调用DistributedFileSystem#getFileBlockStorageLocations()的并发RPC的线程数

- dfs.client.file-block-storage-locations.timeout=60

Timeout (in seconds) for the parallel RPCs made in DistributedFileSystem#getFileBlockStorageLocations().

- dfs.domain.socket.path=/var/run/hadoop-hdfs/dn._PORT

可选选项。socket文件路径，unix下。用来在DN和本地的HDFS客户端加快网络连接。如果字符串"_PORT"出现在路径中，它将被DN的TCP端口替换。

yarn-default.xml

- yarn.app.mapreduce.am.env=null

用户为MR AM添加环境变量。例如：

  - A=foo 设置环境变量A为foo
  - B=$B:c 继承并设置TT内的B变量

- **yarn.app.mapreduce.am.command-opts=-Xmx1024m**

MR AM的Java opts。如下符号会被替换：

- @taskid@ 被替换成当前的TaskID。其它出现的'@'不会改变。例如，为了让gc日志能够按task打印存储在/tmp目录，可以设置'value'为：-Xmx1024m -verbose:gc -Xloggc:/tmp/@taskid@.gc
- 如果hadoop本地库可以使用，使用-Djava.library.path参数可能造成程序的此功能无效。这个值应该被替换，设置在MR的JVM环境中LD_LIBRARY_PATH变量中，使用 mapreduce.map.env和mapreduce.reduce.env配置项。

- **yarn.app.mapreduce.am.resource.mb=1536**

  AM申请的内存

- **yarn.resourcemanager.address=0.0.0.0:8032**

  RM地址:端口

- **yarn.resourcemanager.scheduler.address=0.0.0.0:8030**

  调度器地址：端口

- **yarn.admin.acl=***

  ACL中谁可以管理YARN集群

- **yarn.resourcemanager.admin.address=0.0.0.0:8033**

  RM管理接口地址：端口

- **yarn.resourcemanager.am.max-retries=1**

  AM重试最大次数。服务端参数。重启生效。

  建议**4**

- **yarn.resourcemanager.nodes.include-path=null**

  存储有效节点列表的文件

- **yarn.resourcemanager.nodes.exclude-path=null**

  存储拒绝节点列表的文件。如和包含文件冲突，包含文件优先级高

- **yarn.resourcemanager.scheduler.class=org.apache.hadoop.yarn.server.resourcemanager.scheduler.fifo.FifoScheduler**

  调度器实现类。

  建议使用公平调度器

- **yarn.scheduler.minimum-allocation-mb=1024**

  每个container想RM申请内存的最小大小。兆字节。内存请求小于此值，实际申请到的是此值大小。默认值偏大

- **yarn.scheduler.maximum-allocation-mb=8192**

  每个container向RM申请内存的最大大小，兆字节。申请值大于此值，将最多得到此值内存。

- **yarn.resourcemanager.recovery.enabled=FALSE**

  是否启动RM的状态恢复功能。如果true，必须指定yarn.resourcemanager.store.class。尝试启用

- **yarn.resourcemanager.store.class=null**

  用于持久存储的类。尝试开启

- **yarn.resourcemanager.max-completed-applications=10000**

  RM中保存的最大完成的app数量。内存中存储。

- **yarn.nodemanager.address=0.0.0.0:0**

  NM中的container管理器的地址：端口

- **yarn.nodemanager.env-whitelist=JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,YARN_HOME**

  container应该覆盖而不是使用NM的环境变量名单。允许container自己配置的环境变量

- **yarn.nodemanager.delete.debug-delay-sec=0**

  秒，一个app完成后，NM删除服务将删除app的本地文件目录和日志目录。为了诊断问题，把这个选项设置成足够大的值（例如，设置为10分钟），可以继续访问这些目录。设置此选项，需要重启NM。Yarn应用的工作目录根路径是yarn.nodemanager.local-dirs，Yarn应用日志目录的根路径是yarn.nodemanager.log-dirs。

调试问题时可用

- **yarn.nodemanager.local-dirs=${hadoop.tmp.dir}/nm-local-dir**

本地文件存储目录，列表。一个应用的本地文件目录定位方式：${yarn.nodemanager.local-dirs}/usercache/${user}/appcache/application_${appid}。每个container的工作目录，是此目录的子目录，目录名是container_${contid}。

非常重要，建议配置多个磁盘，平衡IO。

- **yarn.nodemanager.log-dirs=${yarn.log.dir}/userlogs**

存储container日志的地方。一个应用的本地日志目录定位是：${yarn.nodemanager.log-dirs}/application_${appid}。每个container的日志目录在此目录下，名字是container_{$contid}。每个container目录中包含stderr, stdin, and syslog等container产生的文件

非常重要，建议配置多个磁盘

- **yarn.log-aggregation-enable=FALSE**

是否允许日志汇聚功能。建议开启

- **yarn.log-aggregation.retain-seconds=-1**

保存汇聚日志时间，秒，超过会删除，-1表示不删除。 注意，设置的过小，将导致NN垃圾碎片。建议*3-7*天 *= 7 * 86400 = 604800*

- **yarn.nodemanager.log.retain-seconds=10800**

保留用户日志的时间，秒。在日志汇聚功能关闭时生效。

建议**7**天

- **yarn.nodemanager.remote-app-log-dir=/tmp/logs**

汇聚日志的地方，目录路径，HDFS系统。

对于开了权限检查的系统，注意权限问题。**HDFS**上。

- **yarn.nodemanager.remote-app-log-dir-suffix=logs**

  汇聚日志目录路径后缀。汇聚目录创建在{yarn.nodemanager.remote-app-log-dir}/${user}/{thisParam}

- **yarn.nodemanager.resource.memory-mb=8192**

  NM上可以用于container申请的物理内存大小，MB。

- **yarn.nodemanager.vmem-pmem-ratio=2.1**

  在设置container的内存限制时，虚拟内存到物理内存的比率。Container申请的内存如果超过此物理内存，可以以此比率获取虚拟内存用于满足需求。虚拟地址的是物理地址的倍数上限。建议设置的大点，例如：*4.1*，*8.1*，此虚拟内存并非内存，而是占用的虚拟地址。

- **yarn.nodemanager.webapp.address=0.0.0.0:8042**

  NM的网页界面地址和端口。

- **yarn.nodemanager.log-aggregation.compression-type=none**

  汇聚日志的压缩类型。汇聚日志是TFile格式文件。Hadoop-3315。可以使用的值有none,lzo,gz等。

  可以尝试

- **yarn.nodemanager.aux-services=null**

  请配置为：*mapreduce.shuffle*，在*Yarn*上开启*MR*的必须项

- **yarn.nodemanager.aux-services.mapreduce.shuffle.class=org.apache.hadoop.mapred.ShuffleHandler**

  对应参考*yarn.nodemanager.aux-services*

- **mapreduce.job.jar=null**

  Job客户端参数。提交的job的jar文件。

- **mapreduce.job.hdfs-servers=${fs.defaultFS}**

  Job客户端参数。

- **yarn.application.classpath=$HADOOP_CONF_DIR,$HADOOP_COMMON_HOME/share/hadoop/common/*,$HADOOP_COMMON_HOME/share/hadoop/common/lib/*,$HADOOP_HDFS_HOME/share/hadoop/hdfs/*,$HADOOP_HD**

FS_HOME/share/hadoop/hdfs/lib/*,$YARN_HOME/share/hadoop/yarn/*,$YARN_HOME/share/hadoop/yarn/lib/

YARN应用的CLASSPATH，逗号分隔列表。

- yarn.app.mapreduce.am.job.task.listener.thread-count=30

  MR AM处理RPC调用的线程数。

- yarn.app.mapreduce.am.job.client.port-range=null

  MR AM能够绑定使用的端口范围。例如：50000-50050,50100-50200。 如果你先要全部的有用端口，可以留空（默认值null）。

- yarn.app.mapreduce.am.job.committer.cancel-timeout=60000

  毫秒，如果job被kill了，等待output committer取消操作的时间。

- yarn.app.mapreduce.am.scheduler.heartbeat.interval-ms=1000

  MR AM发送心跳到RM的时间间隔，毫秒

- yarn.app.mapreduce.client-am.ipc.max-retries=3

  在重新连接RM获取Application状态前，客户端重试连接AM的次数。

- yarn.app.mapreduce.client.max-retries=3

  客户端重连RM/HS/AM的次数。这是基于ipc接口上的规则

- yarn.ipc.client.factory.class=null

  创建客户端IPC类的工厂类

- yarn.ipc.serializer.type=protocolbuffers

  使用哪种序列化类

- yarn.ipc.server.factory.class=null

  创建IPC服务类的工厂类

- yarn.ipc.exception.factory.class=null

创建IPC异常的工厂类

- yarn.ipc.record.factory.class=null

  创建序列化记录的工厂类

- yarn.ipc.rpc.class=org.apache.hadoop.yarn.ipc.HadoopYarnProtoRPC

  RPC类实现类

- yarn.resourcemanager.client.thread-count=50

  RM用来处理客户端请求的线程数

- yarn.am.liveness-monitor.expiry-interval-ms=600000

  AM报告间隔，毫秒。?

- yarn.resourcemanager.principal=null

  安全选项

- yarn.resourcemanager.scheduler.client.thread-count=50

  调度器用于处理请求的线程数

- yarn.resourcemanager.webapp.address=0.0.0.0:8088

  RM的网页接口地址：端口

- yarn.resourcemanager.resource-tracker.address=0.0.0.0:8031

  ?

- yarn.acl.enable=TRUE

  开启访问控制

- yarn.resourcemanager.admin.client.thread-count=1

RM管理端口处理事务的线程数

- yarn.resourcemanager.amliveliness-monitor.interval-ms=1000

  RM检查AM存活的间隔

- yarn.resourcemanager.container.liveness-monitor.interval-ms=600000

  检查container存活的时间间隔，毫秒。建议短一些，例如*3*分钟

- yarn.resourcemanager.keytab=/etc/krb5.keytab

  安全选项

- yarn.nm.liveness-monitor.expiry-interval-ms=600000

  RM判断NM死亡的时间间隔。
  非主动检查，被动等待，不连接时间超过此值
  *10*分钟无检查到活动，判定*NM*死亡

- yarn.resourcemanager.nm.liveness-monitor.interval-ms=1000

  RM检查NM存活的时间间隔。

- yarn.resourcemanager.resource-tracker.client.thread-count=50

  处理资源跟踪调用的线程数。？

- yarn.resourcemanager.delayed.delegation-token.removal-interval-ms=30000

  安全选项

- yarn.resourcemanager.application-tokens.master-key-rolling-interval-secs=86400

  安全选项

- yarn.resourcemanager.container-tokens.master-key-rolling-interval-secs=86400

  安全选项

- yarn.nodemanager.admin-env=MALLOC_ARENA_MAX=$MALLOC_ARENA_MAX

应该从NM传送到container的环境变量

- yarn.nodemanager.container-executor.class=org.apache.hadoop.yarn.server.nodemanager.DefaultContainerExecutor

  启动containers的类。

- yarn.nodemanager.container-manager.thread-count=20

  用于container管理的线程数

- yarn.nodemanager.delete.thread-count=4

posted @ 2016-05-29 23:40 白开水加糖 阅读(...) 评论(...) 编辑 收藏

刷新评论 刷新页面 返回顶部