|  |  |
| --- | --- |
| **问题** | **答案** |
| HDFS：hdfs数据块放置策略？  为什么默认是3？ | 块的副本策略需要平衡如下几点：  1. 写入性能：尽量就近写入数据  2. 可靠性：最大程度容忍机器故障  3. 数据均衡性：考虑到各个dn的负载  3个块的写入策略：  \* 第一个块：如果当前是在一个dn上，则选取当前dn为第一个；否则随机选取  \* 第二个块：与第一个块不同机架的节点  \* 第三个块：与第二个块同一机架的另一个节点  3个块的设定综合考虑了写入效率和数据的可靠性 |
| HDFS：调整块的大小会造成哪些影响？ | 数据块过大会影响并发执行效率，数据块过小会增加数据寻址时间及造成小文件问题 |
| HDFS：namenode 启动过程？ | 1. 开启安全模式：不能执行数据修改操作 2. 加载fsimage 3. 逐个执行所有Edits文件中的每一条操作将操作合并到fsimage，完成后生成一个空的edits文件 4. 接收datanode发送来的心跳消息和块信息 5. 根据以上信息确定文件系统状态 6. 退出安全模式 |
| HDFS：namenode HA方案？ | 1. 把name.dir指向NFS（Network File System）  2. QJM：基于zookeeper实现主namenode和备namenode的状态切换，journal node保证edit的写入成功  一般使用QJM方案 |
| HDFS：Secondary namenode 的作用？ | 1. 与namenode主备 2. 处理namenode edits到fsimage的合并，减少namenode自身做合并的性能损耗 |
| HDFS：如何扩展HDFS的存储容量？ | 1. 增加datanode节点 2. 扩充datanode 磁盘空间 |
| HDFS：小文件问题的影响及解决？ | 影响：需要内存存储文件元数据及块信息，消耗namenode内存资源；mr任务以块为单位处理数据，块过多会降低处理效率；  解决：   1. 从源头减少小文件（使用其他存储方式，hive不使用动态分区等） 2. 使用hadoop archive打包小文件 |
| HDFS：权限如何管理 | 每个文件和目录有一个所有者（owner）和一个组（group）。文件或目录对其所有者、同组的其他用户以及所有其他用户分别有着不同的权限。  hdfs 默认是开启文件访问权限的，按照rwx 来进行判断是否有读写执行权限。然后根据用户拥有者， 同组用户， 非同组用户 来进行权限管理  也可以设置ACL更加细粒度的管理权限 |
| HDFS：namenode 安全模式？ | 在NameNode启动时，HDFS先进入安全模式，文件系统只接受读数据请求，而不接受删除、修改等变更请求。满足如下条件后退出安全模式：  1. 可用的block占总数的比例：dfs.namenode.safemode.threshold-pct 参数确定，默认0.99  2. 可用的datanode节点数满足配置的数量要求：dfs.safemode.min.datanodes 参数确定，默认0  安全模式相关命令：  \* hadoop dfsadmin -safemode get：查看当前状态  \* hadoop dfsadmin -safemode enter：进入安全模式  \* hadoop dfsadmin -safemode leave：强制离开安全模式  \* hadoop dfsadmin -safemode wait：一直等待直到安全模式结束 |
| MapReduce：map任务数是如何确定的？与hdfs数据块的关系？ | MapReduce处理时按照split读取数据，split默认为block的大小，可以通过设置调整（但需大于block的大小），map任务数由如下逻辑确定：  file\_size / max(mapred.min.split.size, block\_size) |
| MapReduce：解释一下shuffle，partition与sort？ | \* partition：分区，即将数据切分到几个节点。分区数为reduce的任务数（reduce 任务数由 mapreduce.job.reduces 参数设定），map输出的数据根据【partition id + key】来确定由哪个reduce任务处理  \* sort：排序，在map任务写入本地文件时会进行一次排序，目的是为了提高shuffle的效率；在reduce接收到map的数据后，也会进行一次排序  \* shuffle：洗牌，是指将key值相同的数据发给同一个reduce任务，在把map最终结果发送给reduce任务前发生，涉及到数据在不同节点之间的传输 |
| MapReduce：map任务的结果是如何保存的？如何发送给reduce任务？ | map的结果先会保存到内存中的一块缓冲区中（默认是100m），在缓存区中的内容超过一定阈值时（默认80%），将由一个守护进程将内容写入到磁盘文件中（spill操作），map端的数据排序也是在这个阶段执行；在map任务执行完后，会将之前spill产生的文件合并成一个大的文件；  map任务执行完后会通知resourcemanager，reduce任务通过轮询resourcemanager获知map任务执行完毕 |
| YARN：yarn有哪些资源调度算法？ | \* FIFO：先进先出队列，可以设置任务的优先级  \* Fair：公平策略  \* DRF：主资源配额 |
| YARN：yarn如何保证不同container之间资源的隔离？ | \* 内存，两种方式：  \* 线程监控：如果发现任务长时间的超过限制使用内存，则将任务kill  \* cgroup：任务只要超限，就kill。不够灵活，因为任务可能某一时刻短时间超限  \* cpu：可以使用linux cgroup程序实现cpu资源的隔离 |
| Flume：flume与kafka的区别？ | flume是一个即插即用的数据收集组件，只需添加一些配置即可使用；更易使用  kafka是一个高性能的分布式消息队列，需要自己编写生产端和消费端代码，扩展性和稳定性更好 |
| Flume：如果数据没有收集到，可能是什么问题？ | 1. 检查日志，查看是否有报错  2. source收集的问题：不能正常收集数据  3. channel问题：缓冲区溢出  4. sink收集端问题：写入错误 |
| Flume：使用flume采集的一般架构？ | 一般分为三层结构：采集层、汇总层和存储层，而不是直接从采集端将数据发送到存储端，这样的好处有：   * 如果存储端如Hadoop集群、Kafka等需要停机维护或升级，对部署在应用服务器上的采集端没有影响，只需要汇总层做好数据的缓冲，在存储端恢复正常后继续写入数据。 * 采集层只负责数据的采集，由汇总层统一维护数据的路由逻辑（比如发送到hdfs还是kafka？），由于采集端所在的应用服务器一般数量较多，且会随着业务的扩展而不断增加，这种方式可以降低日志采集配置的维护成本，降低大数据应用对业务系统的影响 |