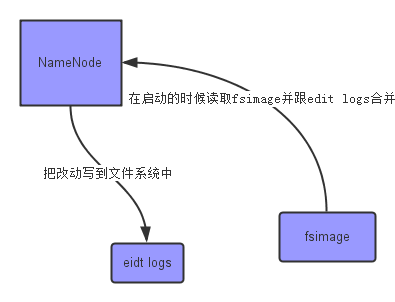
# HDFS NameNode and SecondaryNameNode

参考：<https://www.jianshu.com/p/d16a26dc5794>

## Namenode

NameNode主要是用来保存HDFS的元数据信息，比如命名空间信息，块信息等。当它运行的时候，这些信息是存在内存中的，但是这些信息也可以持久化到磁盘上。



上面的这张图片展示了NameNode怎么把元数据保存到磁盘上的。这里有两个不同的文件：

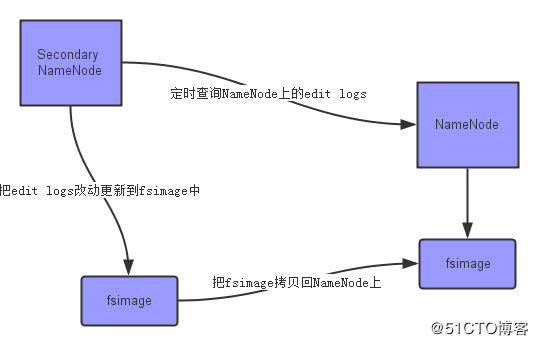
1. fsimage - 它是在NameNode启动时对整个文件系统的快照
2. edit logs - 它是在NameNode启动后，对文件系统的改动序列

只有在NameNode重启时，edit logs才会合并到fsimage文件中，从而得到一个文件系统的最新快照。但是在产品集群中NameNode是很少重启的，这也意味着当NameNode运行了很长时间后，edit logs文件会变得很大。在这种情况下就会出现下面一些问题：

1. edit logs文件会变的很大，怎么去管理这个文件是一个挑战。
2. NameNode的重启会花费很长时间，因为有很多改动[笔者注:在edit logs中]要合并到fsimage文件上。
3. 如果NameNode挂掉了，那我们就丢失了很多改动因为此时的fsimage文件非常旧。[笔者注: 笔者认为在这个情况下丢失的改动不会很多, 因为丢失的改动应该是还在内存中但是没有写到edit logs的这部分。]

为了克服这个问题，我们需要一个易于管理的机制来帮助我们减小edit logs文件的大小和得到一个最新的fsimage文件，这样也会减小在NameNode上的压力。这跟Windows的恢复点是非常像的，Windows的恢复点机制允许我们对OS进行快照，这样当系统发生问题时，我们能够回滚到最新的一次恢复点上。

SecondaryNameNode就是来帮助解决上述问题的，它的职责是合并NameNode的edit logs到fsimage文件中。



上面的这张图片展示了Secondary NameNode是怎么工作的：

1. 首先，它定时到NameNode去获取edit logs，并更新到fsimage上[笔者注：Secondary NameNode自己的fsimage]。
2. 一旦它有了新的fsimage文件，它将其拷贝回NameNode中。
3. NameNode在下次重启时会使用这个新的fsimage文件，从而减少重启的时间。

Secondary NameNode的整个目的是在HDFS中提供一个检查点。它只是NameNode的一个助手节点，这也是它在社区内被认为是检查点节点的原因。

现在，我们明白了Secondary NameNode所做的不过是在文件系统中设置一个检查点来帮助NameNode更好的工作。它不是要取代掉NameNode也不是NameNode的备份。所以从现在起，让我们养成一个习惯，称呼它为检查点节点吧。

最后补充一点细节，是关于NameNode是什么时候将改动写到edit logs中的？这个操作实际上是由DataNode的写操作触发的，当我们往DataNode写文件时，DataNode会跟NameNode通信，告诉NameNode什么文件的第几个block放在它那里，NameNode这个时候会将这些元数据信息写到edit logs文件中。