Solr Cloud简介

原文地址：http://www.chepoo.com/solrcloud-introduction.html

简介

SolrCloud是Solr4.0版本以后基于Solr和Zookeeper的分布式搜索方案。SolrCloud是Solr的基于Zookeeper一种部署方式。Solr可以以多种方式部署，例如单机方式，多机Master-Slaver方式。

概念

Collection：在SolrCloud集群中逻辑意义上的完整的索引。它常常被划分为一个或多个 Shard，它们使用相同的Config Set。如果Shard数超过一个，它就是分布式索引，SolrCloud让你通过Collection名称引用它，而不需要关心分布式检索时需要使用的和Shard相关参数。

ConfigSet: Solr Core提供服务必须的一组配置文件。每个config set有一个名字。最小需要包括solrconfig.xml(SolrConfigXml)和schema.xml (SchemaXml)，除此之外，依据这两个文件的配置内容，可能还需要包含其它文件。它存储在Zookeeper中。Config sets可以重新上传或者使用upconfig命令更新，使用Solr的启动参数bootstrap_confdir指定可以初始化或更新它。

Core: 也就是Solr Core，一个Solr中包含一个或者多个Solr Core，每个Solr Core可以独立提供索引和查询功能，每个Solr Core对应一个索引或者Collection的Shard，Solr Core的提出是为了增加管理灵活性和共用资源。在SolrCloud中有个不同点是它使用的配置是在Zookeeper中的，传统的Solr core的配置文件是在磁盘上的配置目录中。

Leader: 赢得选举的Shard replicas。每个Shard有多个Replicas，这几个Replicas需要选举来确定一个Leader。选举可以发生在任何时间，但是通常他 们仅在某个Solr实例发生故障时才会触发。当索引documents时，SolrCloud会传递它们到此Shard对应的leader，leader再分发它们到全部Shard的replicas。

Replica: Shard的一个拷贝。每个Replica存在于Solr的一个Core中。一个命名为“test”的collection以numShards=1创 建，并且指定replicationFactor设置为2，这会产生2个replicas，也就是对应会有2个Core，每个在不同的机器或者Solr实 例。一个会被命名为test_shard1_replica1，另一个命名为test_shard1_replica2。它们中的一个会被选举为 Leader。

Shard: Collection的逻辑分片。每个Shard被化成一个或者多个replicas，通过选举确定哪个是Leader。

Zookeeper: Zookeeper提供分布式锁功能，对SolrCloud是必须的。它处理Leader选举。Solr可以以内嵌的Zookeeper运行，但是建议用独立的，并且最好有3个以上的主机。

特色功能

SolrCloud有几个特色功能：

集中式的配置信息：使用ZK进行集中配置。启动时可以指定把Solr的相关配置文件上传 Zookeeper，多机器共用。这些ZK中的配置不会再拿到本地缓存，Solr直接读取ZK中的配置信息。配置文件的变动，所有机器都可以感知到。另外，Solr的一些任务也是通过ZK作为媒介发布的。目的是为了容错。接收到任务，但在执行任务时崩溃的机器，在重启后，或者集群选出候选者时，可以再次执行这个未完成的任务。

自动容错：SolrCloud对索引分片，并对每个分片创建多个Replication。每个 Replication都可以对外提供服务。一个Replication挂掉不会影响索引服务。更强大的是，它还能自动的在其它机器上帮你把失败机器上的索引Replication重建并投入使用。

近实时搜索：立即推送式的replication（也支持慢推送）。可以在秒内检索到新加入索引。

查询时自动负载均衡：SolrCloud索引的多个Replication可以分布在多台机器上，均衡查询压力。如果查询压力大，可以通过扩展机器，增加Replication来减缓。

自动分发的索引和索引分片：发送文档到任何节点，它都会转发到正确节点。

事务日志：事务日志确保更新无丢失，即使文档没有索引到磁盘。

其它值得一提的功能有：

索引存储在HDFS(Hadoop分布式文件系统)上：索引的大小通常在G和几十G，上百G的很少，这样的功能或许很难实用。但是，如果你有上亿数据来建索引的话，也是可以考虑一下的。我觉得这个功能最大的好处或许就是和下面这个“通过MR批量创建索引”联合实用。

通过Morphline 批量创建索引：有了这个功能，你还担心创建索引慢吗？（相关的文章：hbase 数据同步到solr https://blog.csdn.net/tom_fans/article/details/79103457 ，

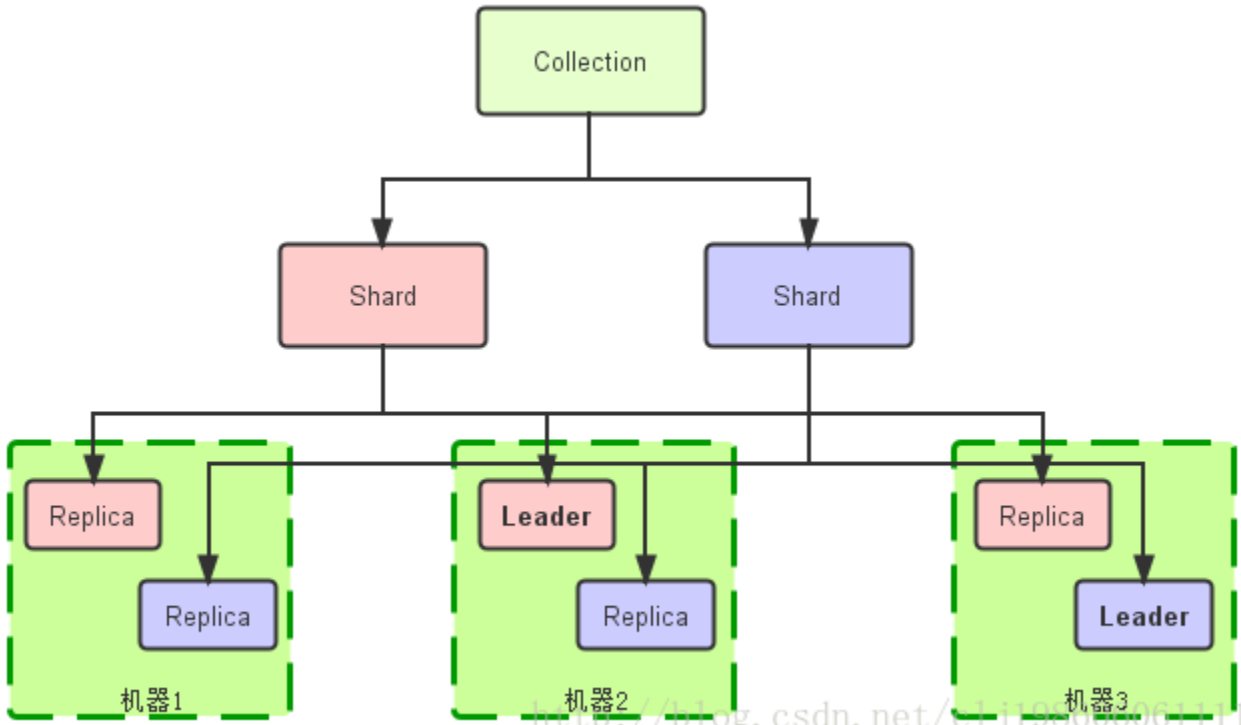
https://blog.csdn.net/u014091123/article/details/73322563)

强大的RESTful API：通常你能想到的管理功能，都可以通过此API方式调用。这样写一些维护和管理脚本就方便多了。

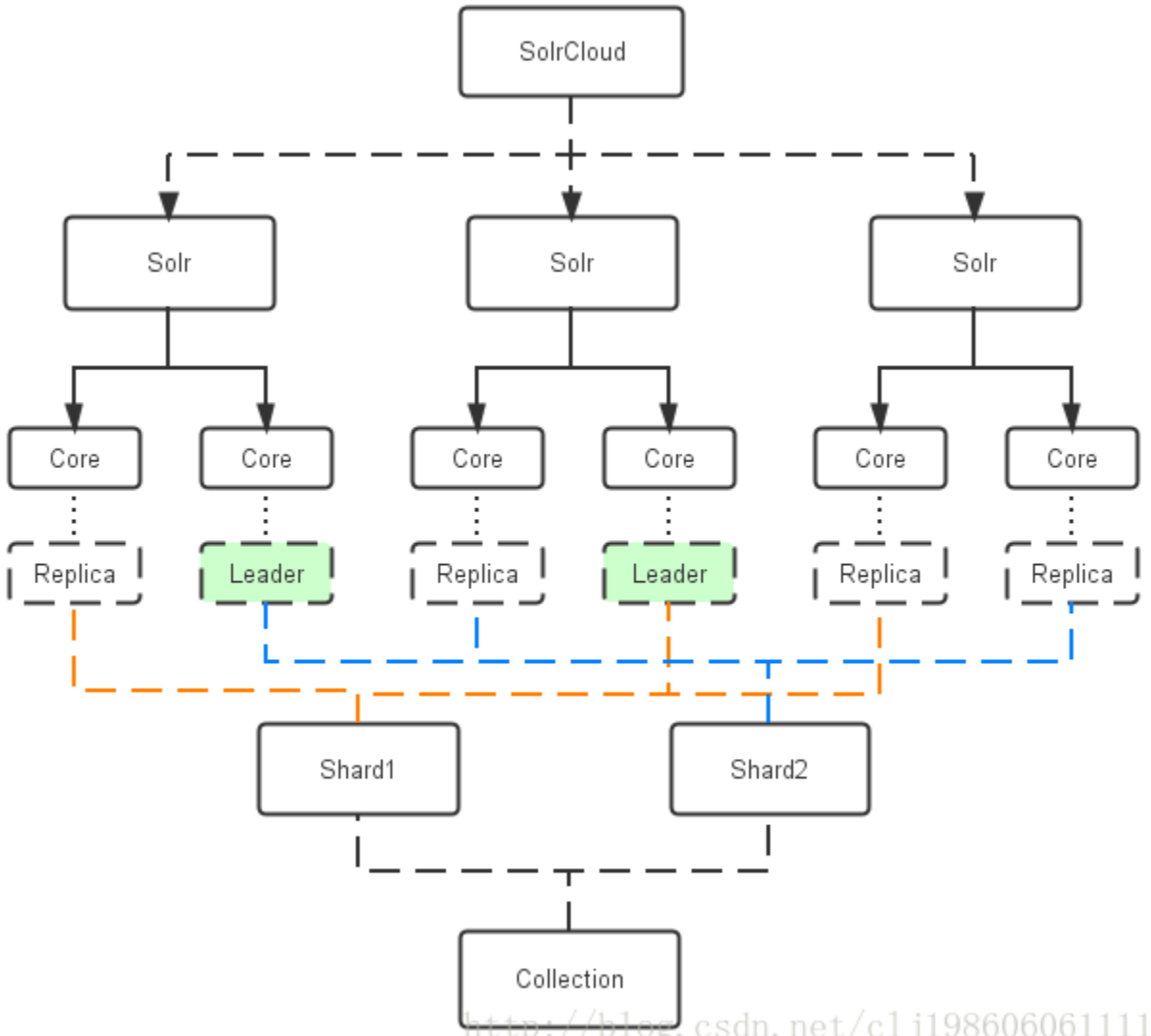
优秀的管理界面：主要信息一目了然；可以清晰的以图形化方式看到SolrCloud的部署分布；当然还有不可或缺的Debug功能。

架构图

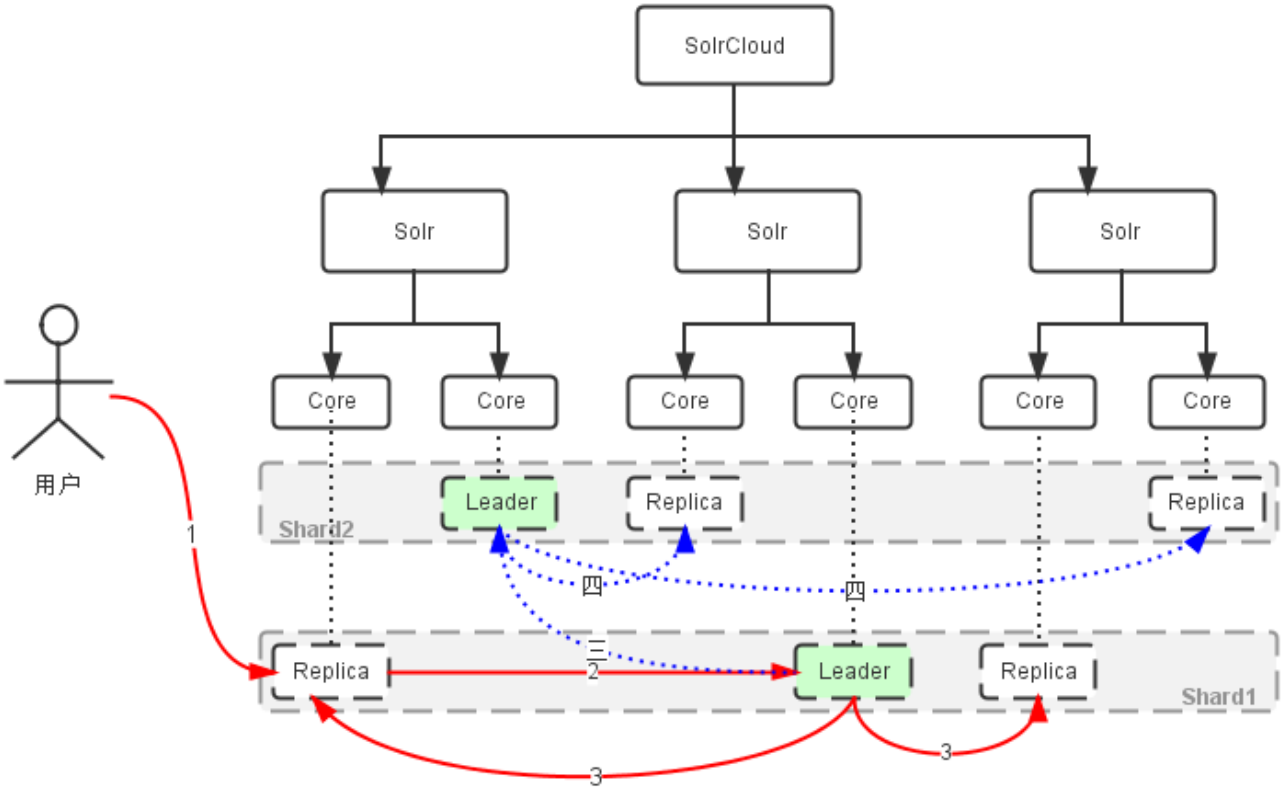
索引（collection）的逻辑图



Solr和索引对照图



创建索引过程



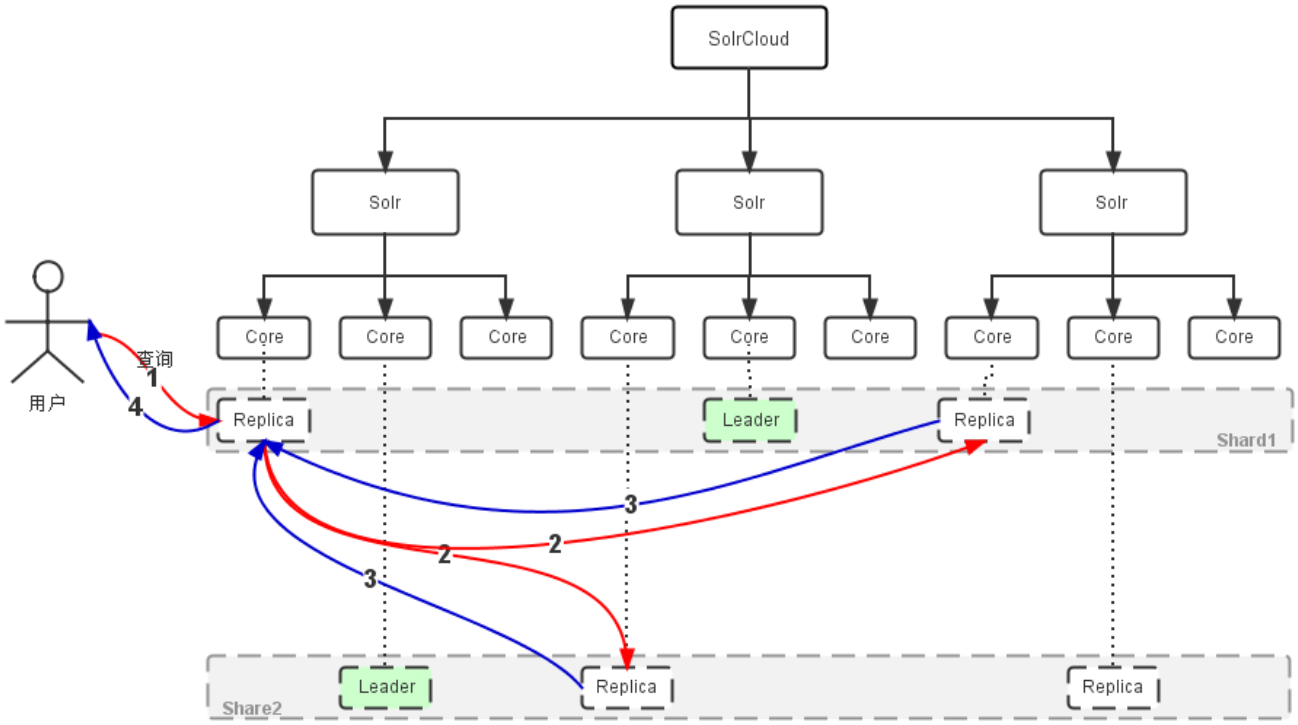
过程描述：

1. 用户可以把文档提交给任一Replica
2. 如果它不是Leader，它会把请求转交给和自己同Shard的Leader
3. Leader把文档路由给本Shard的每个Replica

三. 如果文档基于路由规则并不属于本Shard，leader会把它转交给对应Shard的Leader

四. 对应Leader会把文档路由给本Shard的每个Replica

分布式查询



说明：

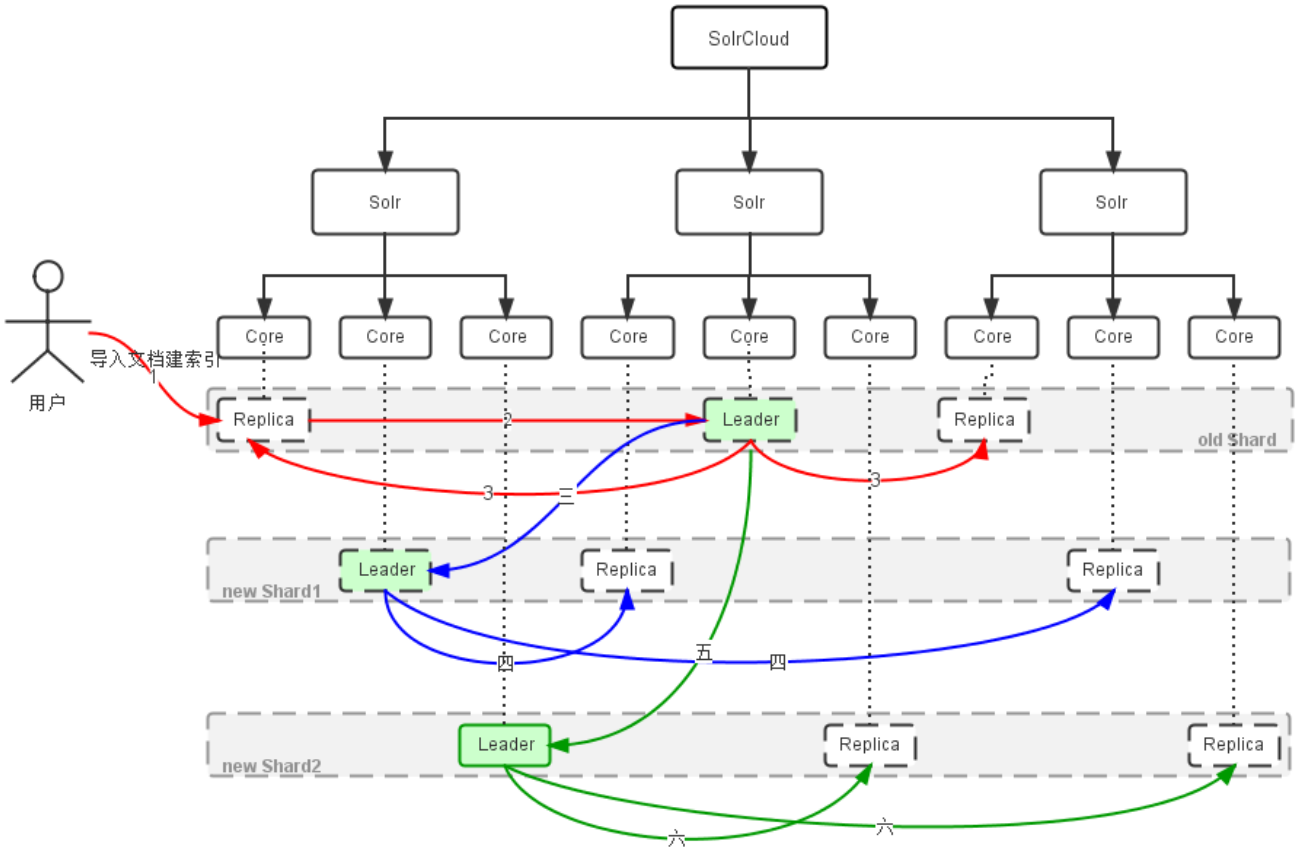
1. 用户的一个查询，可以发送到含有该Collection的任意机器，Solr内部处理的逻辑会转到一个Replica

2. 此Replica会基于查询索引的方式，启动分布是查询，基于索引的Shard个数，把查询转为多个子查询，并把每个子查询定位到对应Shard的任意一个的Replica

3. 每个子查询返回查询结果

4. 最初的Replica合并子查询，并把最终结果返回给用户

ShardSplitting



过程描述：
a. 在一个Shard的文档达到阀值，或者接收到用户的API命令，可以启动分裂过程
b. 此时，旧的Shard仍然提供服务，旧Shard的文档，再次提取并按路由规则，转到新的Shard做索引。
同时，新加入的文档：
1. 2. 用户可以把文档提交给任一Replica，转交给Leader
3. Leader把文档路由给旧Shard的每个Replica，各自做索引
三. 五. 同时，会把文档路由给新Shard的leader
四. 六. 新Shard的Leader会路由文档到自己的Replica，各自做索引
在旧文档重新索引完成，系统会把分发文档路由切到对应的新的Leader上，旧Shard关闭

其他

RT 近实时搜索Solr的建索引数据是要在提交时写入磁盘的，这是硬提交，确保即便是停电也不会丢失数据；为了提供更实时的检索能力，Solr设定了一种软提交方式。软提交（soft commit）：仅把数据提交到内存，index可见，此时没有写入到磁盘索引文件中。

（软提交是针对近实时搜索提供的功能。如果不需要近实时搜索，可以不使用，因为软提交，会导致部分索引需要重新创建且newSearcher，从而影响查询效率。

参考文章：https://blog.csdn.net/limengliang4007/article/details/78092252 ）

一个通常的用法是：每1-10分钟自动触发硬提交，每秒钟自动触发软提交。

RealTime Get 实时获取允许通过唯一键查找任何文档的最新版本数据，并且不需要重新打开searcher。这个主要用于把Solr作为NoSQL数据存储服务，而不仅仅 是搜索引擎。Realtime Get当前依赖事务日志，默认是开启的。另外，即便是Soft Commit或者commitwithin，get也能得到真实数据。 注：commitwithin是一种数据提交特性，不是立刻，而是要求在一定时间内提交数据.