ZooKeeperIntegration

<!> Solr4.0

THIS PAGE MAY BE OBSOLETE! Should we delete it?

Introduction

Integrating Solr and ZooKeeper allow us a lot more flexibility for dynamic, distributed configuration. Additionally, it does not require a breakage of back-compatibility and it can use the existing Solr infrastructure.

See

- http://hadoop.apache.org/zookeeper
- https://issues.apache.org/jira/browse/SOLR-1277
- https://issues.apache.org/jira/browse/SOLR-1431
- https://issues.apache.org/jira/browse/SOLR-1585
- SolrCloud (other design notes)
- DeploymentofSolrCoreswithZookeeper - https://issues.apache.org/jira/browse/SOLR-1724

ZooKeeper Component

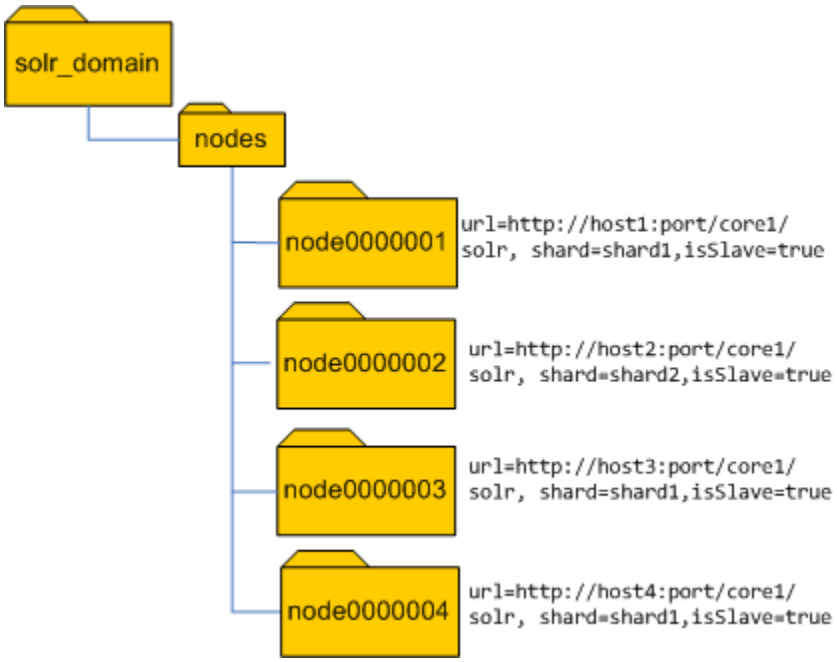
There will be a ZooKeeperComponent to be configured through solrconfig.xml. The ZookeeperComponent may expose the ZooKeeper client instance which could be used by any plugin for purposes such as adding/removing key/values or performing master election etc

Configuration

An example configuration for the ZooKeeperComponent in solrconfig may look like the following:

```
<zookeeper>
    <!-- See the ZooKeeper docs -->
    <str name="zkhostPorts">host1:2181,host2:2181,host3:2181</str>
    <!-- TODO: figure out how to do this programmatically -->
    <str name="me">http://host:port/solr/core1</str>
    <!-- Timeout for the ZooKeeper.  Optional.  Default 10000 -->
    <!-- Timeout in ms -->
    <str name="timeout">5000</str>
    <!-- this is the directory in which this zk node will be added. The name of the node is a sequential number
automatically assigned by zookeeper. The value is a Namedlist which may contain as many values as other
components wish to add. This component only adds the key-> value me=localhost:8983/solr/core1. For instance ,
the Shardhandler may add a key value shard=shard1. ReplicationHandler may add something like version=124544
etc. -->
    <str name="nodesDir">/solr_domain/nodes</str>
 </zookeeper>
```

After the nodes are started it may look as follows



ZooKeeper Aware Distributed Search

For distributed search, a new ShardHandler plugin will be created that moves the shard calculation code from QueryComponent and handles both the current approach and the ZooKeeper approach. There will be a new ShardHandler called ZooKeeperAwareShardHandler which will use the ZooKeeper component to figure out the available shards and their nodes.

ZooKeeperAwareShardHandler's configuration will contain the name of the shard to which this node belongs. On startup, it will get the core's ZooKeeperComponent and add a key/value (the shard name) in ZooKeeper to the current core's node.

Configuration

```
<requestHandler name="standard" class="solr.SearchHandler" default="true">
    <!-- other params go here -->

    <shardHandler class="ZooKeeperAwareShardHandler">
       <str name="shard">shard1</str>
    </shardHandler>
</requestHandler>
```

With the above configuration, on initialization, the ZooKeeperAwareShardHandler will get the ZKClient from the SolrCore and register itself as a sequential node under the path "/solr_domain/nodes" and value url=localhost:8983/solr/core1

TODO: Figure out where does "me" live - zk configuration or shard handler configuration.

Shards are ephemeral and sequential nodes in ZK speak and thus go away if the node dies.

ZooKeeperAwareShardHandler always maintains a list of shard names and a list of nodes that belong to that shard. If the setup has all the slaves sitting behind a loadbalancer, the value of 'me' points to the loadbalancer instead of the node's host:port. The Shardhandler would automatically load balance if there are multiple nodes serving up a shard.

Master/Slave

master/slave setup is only valid for ReplicationHandler. So the configuration of master node can be delegated to ReplicationHandler.

The design of replication in this section is only valid if you use the standard solr.ReplicationHandler . The configuration of ReplicationHandler would look as follows.

```
<requestHandler name="/replication" class="solr.ReplicationHandler">
    <lst name="master">
      <str name="replicateAfter">commit</str>
      <str name="confFiles">schema.xml,stopwords.txt,synonyms.txt</str>
    </lst>
    <lst name="slave">
      <!-- Note that the 'masterUrl' attribute is missing. It has to be fetched from Zookeeper-->
      <str name="pollInterval">00:00:60</str>
    </lst>
    <lst name="zkSetup">
       <str name="masterZnode">/solr_domain/shard1_master</str>
       <bool name="masterCandidate">true</bool>
    </lst>
  </requestHandler>
```

In this design the Zookeeper is used for leader election. When Solr starts up, the ReplicationHandler , if mastercandidate=true ,gets the ZookeeperComponent and gets hold of the Zookeeper client. It tries to become a master by creating a znode (SEQUENTIAL|EPHEMERAL) under the 'masterZnode'. The instance which succeeds in becoming the leader will be the master. All other nodes will treat this as the master. if this node dies another leader is elected and that becomes the master.

The node created will store all the details : eg: master000000001=url=http://host:port/solr/cor1/replication,version=13254424,generation=5

Other Replication Strategies

NOTE COMPLETELY IMPLEMENTED YET.

Nodes can register themselves as Masters by adding their entry to a Master Group and marking themselves as a master. The ReplicationHandler can then be configured to subscribe to that Master Group, getting the first one out of the list of children of the group (this is dependent on ZooKeeper supporting getFirstChild() which it currently does not) Masters are ephemeral. If that is not implemented, then we need some other way of selecting the master. For now, it could just be configured so that there is only one master.

Thus, if there are two groups of Masters, then it would look like this: master_group_1/

- 192.168.0.1_8080_solr [192.168.0.1:8080/solr] 192.168.0.2_8080_solr [192.168.0.2:8080/solr]

master_group_2/

- 192.168.0.3_8080_solr [192.168.0.3:8080/solr] 192.168.0.4_8080_solr [192.168.0.4:8080/solr]

The trick here is how to keep all the masters in a group in sync. Ideas: 1. Servlet filter that replicates out indexing commands to other masters in a master group 2. backup masters replicate from the master 3. Others?, as neither of these is 100% fault tolerant

Rebalancing

Through the ZK req handler, slaves can be moved around, at which point they will pull the index from the master in their group and thus you can have rebalancing. Additionally, new nodes that come online w/o an index will go to their master and get the index. The replication handler already handles replicating configuration files, so this is just a config issue.

The ShardHandler is automatically setup.

1. Setup ZooKeeper according to ZooKeeper docs, including a ZK config file.
2. Startup the ZooKeeper server with your configuration file.
3. Startup your Solr nodes, all properly configured

TODO: Show real example.