Managing a large scale SolrCloud cluster via APIs

Anshum Gupta
Apache Lucene/Solr Committer & PMC member
About me

• Anshum Gupta, Apache Lucene/Solr committer and PMC member, IBM Watson Search team.
• Interested in search and related stuff.
• Apache Lucene since 2006 and Solr since 2010.
Apache Solr is the most widely-used search solution on the planet.

You use Solr everyday. Solr has tens of thousands of applications in production.

Bloomberg

Solr is both established and growing.

8,000,000+
Total downloads

250,000+
Monthly downloads

Etsy

2,500+
Open Solr jobs and the largest community of developers.

Instagram
tumblr.

box
A little bit of history

• SolrCloud
  • Introduced in 4.0
  • Not Solr in the Cloud but set of distributed features for Solr
• No APIs to manage distributed clusters
• We’ve come a long way since then
SolrCloud - Physical Architecture

ZooKeeper

Node 1

Node 2

Load Balancer

Lots Of Interaction

Client

Client

Client

Client

Client

Client

Client

Client

Client

Coins by Creative Stall from the Noun Project
SolrCloud Terminology

- **Node**: JVM process that hosts the Solr web application
- **Collection**: Logic index separated across multiple nodes
- **Shard**: Logical slice of a collection. Each shard has a name, hash range (optional), leader, and replicationFactor.
- **ReplicationFactor**: Number of copies of each shard.
- **Replica**: Physical Solr index that can accept updates or queries. Implemented as a Solr core.
- **Leader**: One replica per shard which assumes special duties like versioning. They are elected using ZooKeeper.
- **Overseer**: Special node that manages cluster level operations. Also elected using Zookeeper.
Exploring the APIs - Getting Started
Creating a collection

- **HTTP API**

  ```
  http://localhost:8983/solr/admin/collections?
  action=CREATE&name=foo&numShards=2&replicationFactor=1
  ```

- **Response**

  ```
  {
  "responseHeader":{
  "status":0,
  "QTime":1848},
  "success":{
  "":{"responseHeader":{
  "status":0,
  "QTime":1741},
  "core":"foo_shard1_replica1"}}
  ```
Add a replica

• HTTP API

http://localhost:8983/solr/admin/collections?
action=ADDREPLICA&collection=foo&shard=shard1

• Response

{"responseHeader":{
  "status":0,
  "QTime":1156},
"success":{
  "":{"responseHeader":{
    "status":0,
    "QTime":1147},
  "core":"foo_shard1_replica4"}}}
We added more than what we need - Delete replica

**HTTP API**

```
http://localhost:8983/solr/admin/collections?
action=DELETEREPLICA&collection=foo&shard=shard1&replica=core_node3
```

**Response**

```
{"responseHeader":{
    "status":0,
    "QTime":111},
"success":{
  "":{
    "responseHeader":{
      "status":0,
      "QTime":2}}}}
```
Exploring the APIs - Rule based scaling out
Rule based Replica Assignment

• Use it during creation of:
  • Collection
  • Shard
  • Replica
• Rules based on:
  • Hostname
  • Free disk space
  • Number of cores
  • Node name
  • System property
  • Other Metrics
Rule based Replica Assignment

• How is a rule defined?
  \( f(\text{shard}, \text{replica}, \text{tag}) \)

• When creating a new replica, apply to each node.
• Supported Operators
  \( \text{default}, \gt, \lt, ! \)
Rule based Replica Assignment - Examples

• Example #1 - for any shard, limit number of replicas to 2 on any node
  
  \[ \text{shard:*,replica:<2,node:*} \]

• Example #2 - For any replica, limit number of cores to 5 on a node
  
  \[ \text{replica:*,cores:<5} \]

• Example #3 - Do not create any cores on host 198.21.2.19
  
  \[ \text{host:!198.21.2.19} \]
Adjusting to growth - Moving data
Divide and Rule? - Let’s split a shard

- HTTP API


- Response

  TooVerbose to Share!
Another way to move data - Migrate

- **HTTP API**
  
  ```
  http://localhost:8983/solr/admin/collections?
  action=MIGRATE&collection=foo&split.key=a!&target.collection=bar
  ```

- **Response**
  
  Too Verbose to fit in here!

- **What happens?**
  
  - Move documents to target collection
  - Handle incoming updates and have temporary routing for original collection to send updates to target
Monitoring your Cluster
LIST Collections - Let’s see what we have

• HTTP API


• Response

```json
{"responseHeader":{
    "status":0,
    "QTime":0},
    "collections":["a",
    "bar",
    "foo"]}
```
CLUSTERSTATUS: Don’t we like details?

• HTTP API


{"cluster": {
    "collections": {
        "foo": {
            "replicationFactor": "1",
            "shards": {
                "shard1": {
                    "range": "80000000-7fffffff",
                    "state": "active",
                    "replicas": {
                        "core_node1": {
                            "core": "foo_shard1_replica1",
                            "node_name": "192.168.1.125:8983_solr",
                            "state": "active",
                            "leader": "true"},
                        "core_node2": {
                            "core": "foo_shard1_replica2",
                            "node_name": "192.168.1.125:8983_solr",
                            "state": "active"}}},
            "router": {
                "name": "compositeId",
                "maxShardsPerNode": "100",
                "autoAddReplicas": "false",
                "znodeVersion": 14,
                "configName": "a"},
            "live_nodes": ["192.168.1.125:8983_solr"]}}}
OVERSEERSTATUS - More than a sneak peek

• HTTP API


• Response - Verbose. Actually, more than that!
• Stats for:
  • Errors
  • Time taken
  • Much more
Managing your Cluster
Set a specific node to be the Overseer

- ADD Role

```
http://localhost:8983/solr/admin/collections?
action=ADDRULE&role=overseer&node=192.168.1.125:8983_solr
```

- Response - Nothing substantial but a 200 on success
- Supported role:
  - Only Overseer for now
- Remember - It’s only a preference
Setting a preferred leader

- Using ADDREPLICAPROP

```
http://localhost:8983/solr/admin/collections?
action=ADDREPLICAPROP&shard=shard1&collection=foo&replica=core_node2&property=
preferredLeader&property.value=true
```

- Sets ‘preferredLeader’ to ‘true’. The only pre-defined property.
Setting preferred leaders for the entire collection

• Using BALANCESHARDUNIQUE

```
http://localhost:8983/solr/admin/collections?
action=BALANCESHARDUNIQUE&collection=foo&property=preferredLeader
```

• Sets ‘preferredLeader’ to ‘true’ evenly across the collection
Rebalance the leaders

• Using REBALANCELEADERS

http://localhost:8983/solr/admin/collections?
action=REBALANCELEADERS&collection=foo

• This allows preferredLeaders to take up leadership for shards.
I lost my leader! Where do I go now?

• Force a leader election using FORCELEADER API

  http://localhost:8983/solr/admin/collections?
  action=FORCELEADER&collection=foo&shard=shard1

• This can ONLY be used when the shard has no leader.
Recipe #1 - Shard Splitting
Splitting a shard - It’s more than just the single API call

- Monitor disk-size
  - CLUSTERSTATUS & Core STATUS
- SPLITSHARD
  - Make sure there’s enough spare disk space
- Add one more replica
- Force leader election
- Delete old INACTIVE shard
Recipe #2 - Ensuring high availability
High availability through a monitoring application

• Auto-add replica
• Monitor cluster status - CLUSTERSTATUS API
• ADD Replica using replica placement strategy
• Remove replicas when there are too many
Recipe #3 - Migrating Cluster Infrastructure
High availability through a monitoring application

- CLUSTERSTATUS
  - Overseer, Leaders, etc.
- Move the Overseer first
- Rebalance leaders
- Force leader elections if required
What’s more?
There’s a lot more, a single session is too less!

- Version 2 APIs
- SolrJ support for all the APIs - Easy to tool in Java
- Security
- Cross Data Center Support
Coming Soon - SolrCloud Backup and Restore!

• Committed just a few days ago
• Back up and restore of
  • Collections
  • ZK metadata
  • Configs
• Release with 6.1 (and may be 5.6)
Connect @

http://www.twitter.com/anshumgupta
http://www.linkedin.com/in/anshumgupta/
anshum@apache.org