Auto Data Balancing

The `confluent-rebalancer` tool balances data so that the number of leaders and disk usage are even across brokers and racks on a per topic and cluster level while minimising data movement. It also integrates closely with the replication quotas feature in Apache Kafka to dynamically throttle data balancing traffic.

The tool is part of Confluent Enterprise and can also be installed on its own via the `confluent-rebalancer` package.

Requirements

In order to compute the rebalance plan, the tool relies on metrics collected from the Kafka cluster. This data is published by the Confluent Metrics Reporter (../metrics-reporter.html#metrics-reporter) to a configurable Kafka topic (`_confluent-metrics` by default) in a configurable Kafka cluster.

To enable it, please follow the Confluent Metrics Reporter (../metrics-reporter.html#metrics-reporter) installation instructions.

Quickstart

Start a Kafka cluster

First, startup a ZooKeeper server. In this guide, we are assuming services will run on `localhost`.

```
# Start ZooKeeper.  Run this command in its own terminal.
./bin/zookeeper-server-start ./etc/kafka/zookeeper.properties
```

Tip

These instructions assume you are installing Confluent Platform by using ZIP or TAR archives. For more information, see Install (../../installation/installing_cp/index.html#installation).

Copy the broker configuration file to a temporary location, enable the metrics reporter and duplicate the config for additional brokers.

```
# Copy the config files to /tmp
cp ./etc/kafka/server.properties /tmp/server0.properties

# Add metrics reporter configs (alternatively, we could uncomment the configs)
echo "" >> /tmp/server0.properties
echo "metric.reporters=io.confluent.metrics.reporter.ConfluentMetricsReporter" >> /tmp/server0.properties
echo "confluent.metrics.reporter.bootstrap.servers=localhost:9092" >> /tmp/server0.properties
echo "confluent.metrics.reporter.topic.replicas=1" >> /tmp/server0.properties

# properties for broker.id=1
cp /tmp/server0.properties /tmp/server1.properties
sed -i '' -e "s/broker.id=0/broker.id=1/g" /tmp/server1.properties
sed -i '' -e "s/9092/9082/g" /tmp/server1.properties
sed -i '' -e "s/#listen/listen/g" /tmp/server1.properties
sed -i '' -e "s/kafka-logs/kafka-logs-1/g" /tmp/server1.properties

# properties for broker.id=2
cp /tmp/server0.properties /tmp/server2.properties
sed -i '' -e "s/broker.id=0/broker.id=2/g" /tmp/server2.properties
sed -i '' -e "s/9092/9072/g" /tmp/server2.properties
sed -i '' -e "s/#listen/listen/g" /tmp/server2.properties
sed -i '' -e "s/kafka-logs/kafka-logs-2/g" /tmp/server2.properties

# properties for broker.id=3
cp /tmp/server0.properties /tmp/server3.properties
sed -i '' -e "s/broker.id=0/broker.id=3/g" /tmp/server3.properties
sed -i '' -e "s/9092/9062/g" /tmp/server3.properties
sed -i '' -e "s/#listen/listen/g" /tmp/server3.properties
sed -i '' -e "s/kafka-logs/kafka-logs-3/g" /tmp/server3.properties
```

Next, start the Kafka brokers.

```
# Start Kafka.  Run these commands in a separate terminal.
./bin/kafka-server-start /tmp/server0.properties &
./bin/kafka-server-start /tmp/server1.properties &
./bin/kafka-server-start /tmp/server2.properties &
./bin/kafka-server-start /tmp/server3.properties &
```

For complete details on getting these services up and running see the quickstart (../../quickstart/index.html#quickstart) instructions for Confluent Platform.

Create topics and produce data

First let's create a couple of topics, each with 4 partitions and replication factor of 2. We intentionally create an unbalanced assignment.

```
./bin/kafka-topics --create --topic topic-a --replica-assignment 0:1,0:1,0:1,0:1 --zookeeper localhost:2181
./bin/kafka-topics --create --topic topic-b --replica-assignment 1:0,2:1,1:2,2:1 --zookeeper localhost:2181
```

Let's see how they look.

```
./bin/kafka-topics --describe --topic topic-a --zookeeper localhost:2181

Topic:topic-a        PartitionCount:4        ReplicationFactor:2     Configs:
   Topic: topic-a    Partition: 0    Leader: 0       Replicas: 0,1   Isr: 0,1
   Topic: topic-a    Partition: 1    Leader: 0       Replicas: 0,1   Isr: 0,1
   Topic: topic-a    Partition: 2    Leader: 0       Replicas: 0,1   Isr: 0,1
   Topic: topic-a    Partition: 3    Leader: 0       Replicas: 0,1   Isr: 0,1

./bin/kafka-topics --describe --topic topic-b --zookeeper localhost:2181

Topic:topic-b        PartitionCount:4        ReplicationFactor:2     Configs:
  Topic: topic-b     Partition: 0    Leader: 1       Replicas: 1,0   Isr: 1,0
  Topic: topic-b     Partition: 1    Leader: 2       Replicas: 2,1   Isr: 2,1
  Topic: topic-b     Partition: 2    Leader: 1       Replicas: 1,2   Isr: 1,2
  Topic: topic-b     Partition: 3    Leader: 2       Replicas: 2,1   Isr: 2,1
```

We'll now produce some data.

```
./bin/kafka-producer-perf-test --topic topic-a --num-records 200000 --record-size 1000 --throughput 10000000 -
-producer-props bootstrap.servers=localhost:9092
./bin/kafka-producer-perf-test --topic topic-b --num-records 800000 --record-size 1000 --throughput 10000000 -
-producer-props bootstrap.servers=localhost:9092
```

And finally we'll force the creation of the offsets topic by running a consumer.

```
./bin/kafka-consumer-perf-test --topic topic-a --broker-list localhost:9092 --messages 10
```

Execute the rebalancer

Before we start, it's worth mentioning that if you run `./bin/confluent-rebalancer` with no arguments, it will output a list of supported commands along with a description.

Let's start with the `execute` command. We specify the connection string for the ZooKeeper ensemble (with optional chroot), the bootstrap servers for Kafka cluster containing the metrics topic and the maximum bandwidth (in bytes per second) allocated to moving replicas. The `verbose` flag includes per broker stats in the CLI output, which is useful unless the number of brokers is very large.

```
./bin/confluent-rebalancer execute --zookeeper localhost:2181 --metrics-bootstrap-server localhost:9092 --thro
ttle 10000000 --verbose
```

You will be presented with a rebalancing plan, which will be slightly different depending on whether the Kafka brokers are configured to use a single log directory (like in our example) or multiple log directories. In the former case, the rebalancer will ensure that the volume containing the log directory will have at at least the specified percentage of free space during and after the rebalance (with 20% as default). We intend to expand this feature to support multiple log directories in a future release.

```
Computing the rebalance plan (this may take a while) ...
You are about to move 17 replica(s) for 14 partitions to 4 broker(s) with total size 827.2 MB.
The preferred leader for 14 partition(s) will be changed.
In total, the assignment for 15 partitions will be changed.
The minimum free volume space is set to 20.0%.

The following brokers will have less than 40% of free volume space during the rebalance:
      Broker     Current Size (MB)  Size During Rebalance (MB)   Free % During Rebalance      Size After Rebal
ance (MB)    Free % After Rebalance
      0          413.6              620.4                        30.1                         519.6
             30.5
      2          620.4              723.8                        30.1                         520.8
             30.5
      3          0                  517                          30.1                         520.8
             30.5
      1          1,034              1,034                        30.1                         519.6
             30.5

Min/max stats for brokers (before -> after):
      Type  Leader Count                Replica Count                Size (MB)
      Min   12 (id: 3) -> 17 (id: 0)     37 (id: 3) -> 43 (id: 3)     0 (id: 3) -> 517 (id: 1)
      Max   21 (id: 0) -> 17 (id: 0)     51 (id: 1) -> 45 (id: 0)     1,034 (id: 1) -> 517 (id: 3)
No racks are defined.

Broker stats (before -> after):
  Broker     Leader Count    Replica Count   Size (MB)            Free Space (%)
      0          21 -> 17        48 -> 45        413.6 -> 517       30.5 -> 30.5
      1          20 -> 17        51 -> 44        1,034 -> 517       30.5 -> 30.5
      2          15 -> 17        40 -> 44        620.4 -> 517       30.5 -> 30.5
      3          12 -> 17        37 -> 43        0 -> 517           30.5 -> 30.5

Would you like to continue? (y/n):
```

Because we are running all brokers in the same volume, the free space numbers don't change after the rebalance. If multiple log directories are configured, it's worth paying close attention to the additional disk space requirements during and after the rebalance to ensure that it won't cause a broker to run out of disk space.

In addition, the `min/max` stats provide a quick summary of the data balance improvement after the rebalance completes. The goal should be for the `min` and `max` values to be closer to each other after the rebalance. In this case, we are able to achieve near optimal balance so the numbers are virtually identical.

If you proceed, you will then see the following (don't ignore the warning!):

```
Rebalance started, its status can be checked via the status command.

Warning: You must run the status or finish command periodically, until the rebalance completes, to ensure the
 throttle is removed. You can also alter the throttle by re-running the execute command passing a new value.
```

At this point, the tool will exit and the rebalance will happen in the background (driven by the Kafka Controller).

Check status and finish

Let's check the status of the rebalance.

```
./bin/confluent-rebalancer status --zookeeper localhost:2181

Partitions being rebalanced:
        Topic topic-a: 1,2
        Topic topic-b: 0
```

Eventually, the rebalance will complete (`finish` and `status` are similar, the latter has more concise output and returns a 0 exit status code when the rebalance is finished).

```
./bin/confluent-rebalancer finish --zookeeper localhost:2181                    ⏎

The rebalance has completed and throttling has been disabled
```

We can now verify that the replica assignment is now balanced.

```
./bin/kafka-topics --describe --topic topic-a --zookeeper localhost:2181

Topic:topic-a        PartitionCount:4        ReplicationFactor:2     Configs:
        Topic: topic-a       Partition: 0    Leader: 3       Replicas: 3,2   Isr: 2,3
        Topic: topic-a       Partition: 1    Leader: 2       Replicas: 2,3   Isr: 2,3
        Topic: topic-a       Partition: 2    Leader: 0       Replicas: 1,0   Isr: 0,1
        Topic: topic-a       Partition: 3    Leader: 0       Replicas: 0,1   Isr: 0,1

./bin/kafka-topics --describe --topic topic-b --zookeeper localhost:2181

Topic:topic-b        PartitionCount:4        ReplicationFactor:2     Configs:
        Topic: topic-b       Partition: 0    Leader: 3       Replicas: 3,0   Isr: 0,3
        Topic: topic-b       Partition: 1    Leader: 0       Replicas: 0,3   Isr: 0,3
        Topic: topic-b       Partition: 2    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: topic-b       Partition: 3    Leader: 2       Replicas: 2,1   Isr: 2,1
```

Note that the partition sizes are communicated asynchronously (every 15 seconds by default and configurable via the `confluent.metrics.reporter.publish.ms` config), so there may be a delay before the tool reports the correct information after a rebalance is finished (particularly noticeable when doing local tests such as this).

Leader balance

If `auto.leader.rebalance.enable` is disabled on your brokers, run the preferred leader election tool after the rebalance completes. This will ensure that the actual leaders are balanced (not just the preferred leaders).

Decommissioning brokers

The `confluent-rebalancer` tool automatically generates a rebalance plan for decommissioning brokers via the `--remove-broker-ids` option. Note that the cluster will also be balanced in the same execution.

```
./bin/confluent-rebalancer execute --zookeeper localhost:2181 --metrics-bootstrap-server localhost:9092 --thro
ttle 100000 --remove-broker-ids 1

Computing the rebalance plan (this may take a while) ...
You are about to move 48 replica(s) for 48 partitions to 3 broker(s) with total size 775.5 MB.
The preferred leader for 20 partition(s) will be changed.
In total, the assignment for 49 partitions will be changed.

You have requested all replicas to be moved out of 1 broker(s) with ID(s): 1.
After the rebalance, these broker(s) will have no replicas.

The following brokers will require more disk space during the rebalance and, in some cases, after the rebalanc
e:
     Broker     Current (MB)    During Rebalance (MB)  After Rebalance (MB)
     0          517             775.5                  568.7
     2          517             775.5                  723.8
     3          517             775.5                  775.5

Min/max stats for brokers (before -> after):
     Type  Leader Count                Replica Count               Size (MB)
     Min   17 (id: 0) -> 0 (id: 1)     43 (id: 3) -> 0 (id: 1)     517 (id: 2) -> 0 (id: 1)
     Max   17 (id: 0) -> 23 (id: 0)    45 (id: 0) -> 59 (id: 0)    517 (id: 1) -> 775.5 (id: 3)
No racks are defined.

Would you like to continue? (y/n):
```

Given that we have 8 partitions with roughly the same amount of data, it's not possible for the 3 remaining brokers to have the same amount of data. Broker `2` would have 2 partitions after the rebalance and hence less data.

Limiting Bandwidth Usage during Data Migration

We have used the `--throttle` option to limit the amount of bandwidth used for replication. It is possible to update the value while a rebalance is in progress by simply rerunning the tool.

```
./bin/confluent-rebalancer execute --zookeeper localhost:2181 --metrics-bootstrap-server localhost:9092 --thro
ttle 100000

The throttle rate was updated to 100000 bytes/sec.
A rebalance is currently in progress for:
        Topic topic-b: 0,1
```

See the Throttling (../post-deployment.html#throttling) documentation for more details.

Licensing

We didn’t specify a license in the quickstart. That means that it will trigger a 30 day trial period. If you have a Confluent license, you can specify it via the `confluent.license` config.

Configuration Options

» Configuration Options for the rebalancer tool (configuration_options.html)
