I have a good shard key now what?

David Murphy , Mongo Master
Lead DBA, ObjectRocket
@dmurphy_data @objectrocket

Background

- 16 yrs in databases, development, & system engineering
- Lead DBA @ ObjectRocket
- Mongo Master with a focus on sharding, chunks, and scaling mongo beyond normal means.

Quick Sharding Introduction

Why you might need to shard:

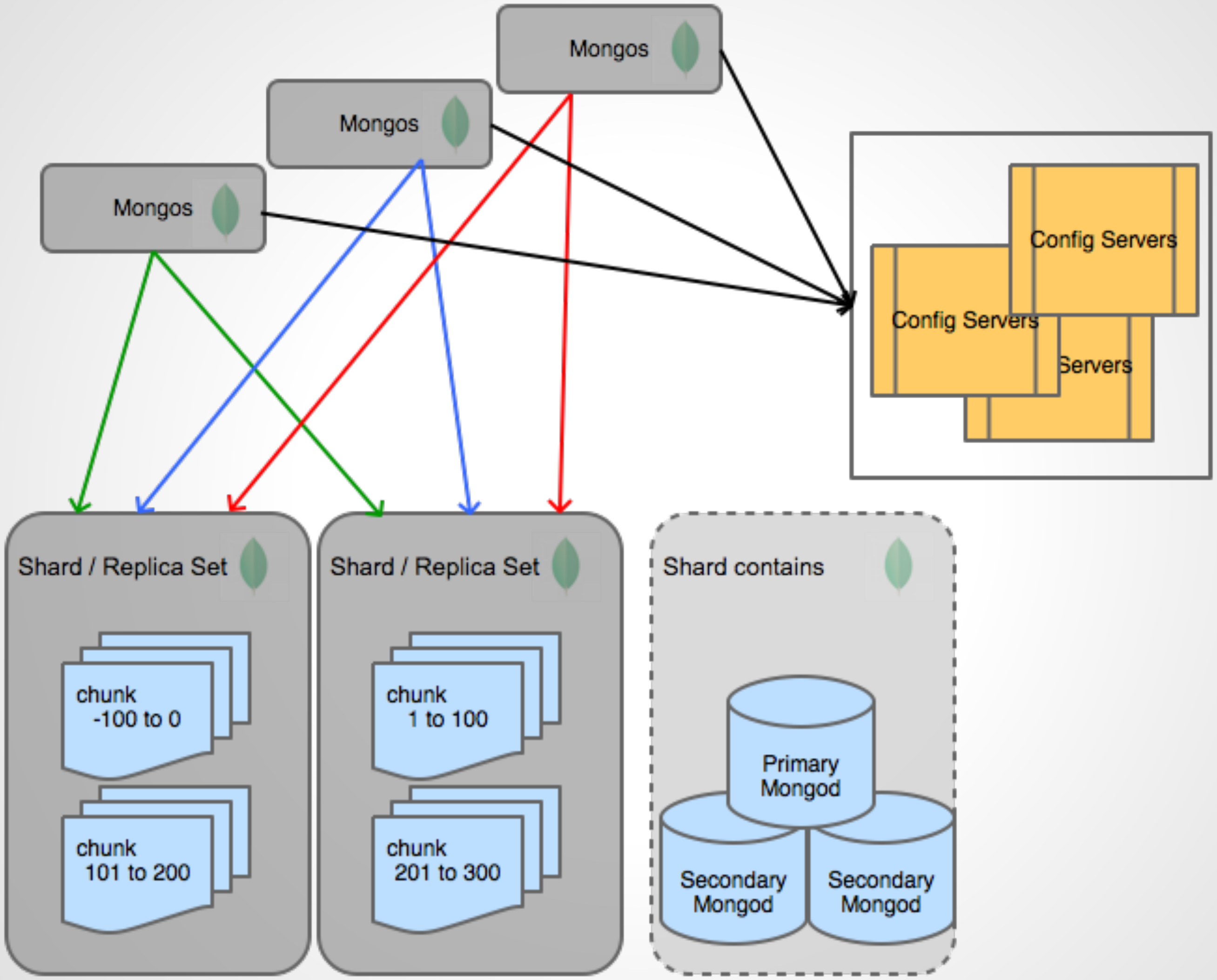
- Scaling Reads
- Fast Queries
- Scaling Writes
- Balancing nodes

What are things should you consider?

- Shard Key is Immutable
- Targeted vs Scatter-Gather
- Sharding Overhead
- Some commands no longer work

http://bit.ly/1oXYDfm - Kenny Gorman sharding talk
http://bit.ly/ZTtDI1 - Several other sharding talks

What does a shared cluster look like?



ObjectRocket

The balancer is great…

… but you can also perform some proactive checks to predict issues.

- ☐ **Understanding the reasons why not all chunks are equal**
- ☐ **Counting Chunks per shard without sh.status()**
- ☐ **Finding out how many chunkMove's or splits occur at a time**
- ☐ **Getting the size of all chunks for a collection**

Ways to Count Chunks

Many functions for this:

- sh.status()
- db.collection.stats()

Simple query for counting chunks:

- db.chunks.count({ns:"database.collection"});

Why I prefer using Aggregation:

- Faster as you only look at one type of data
- Provides chunk count per shard
- Can programmatically use the output

Tracking Chunks and Moves

The **config.changelog** tracks chunk changes.

The types of changes to chunks are

- **split**
- moveChunk
  - **moveChunk.start**
  - moveChunk.from
  - moveChunk.to
  - **moveChunk.commit**

Chunk Collection Examples

http://bit.ly/112tXEx

ChangeLog Examples

http://bit.ly/1oZlyqN

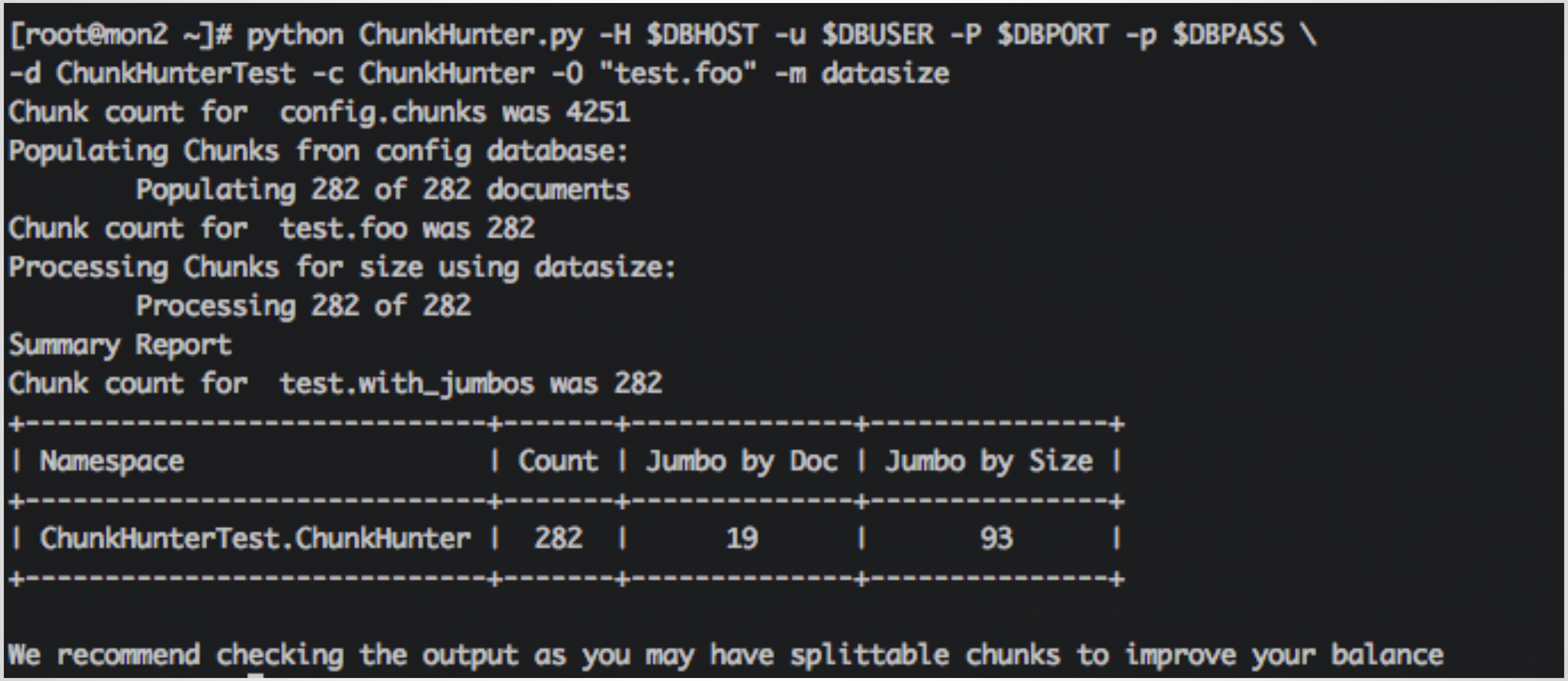
Sizing Chunks

- dataSize command - Very expensive
  Scans all documents to be 100% accurate

- find().count() * avgObjSize on chunk ranges
  Rough size estimate based on stats

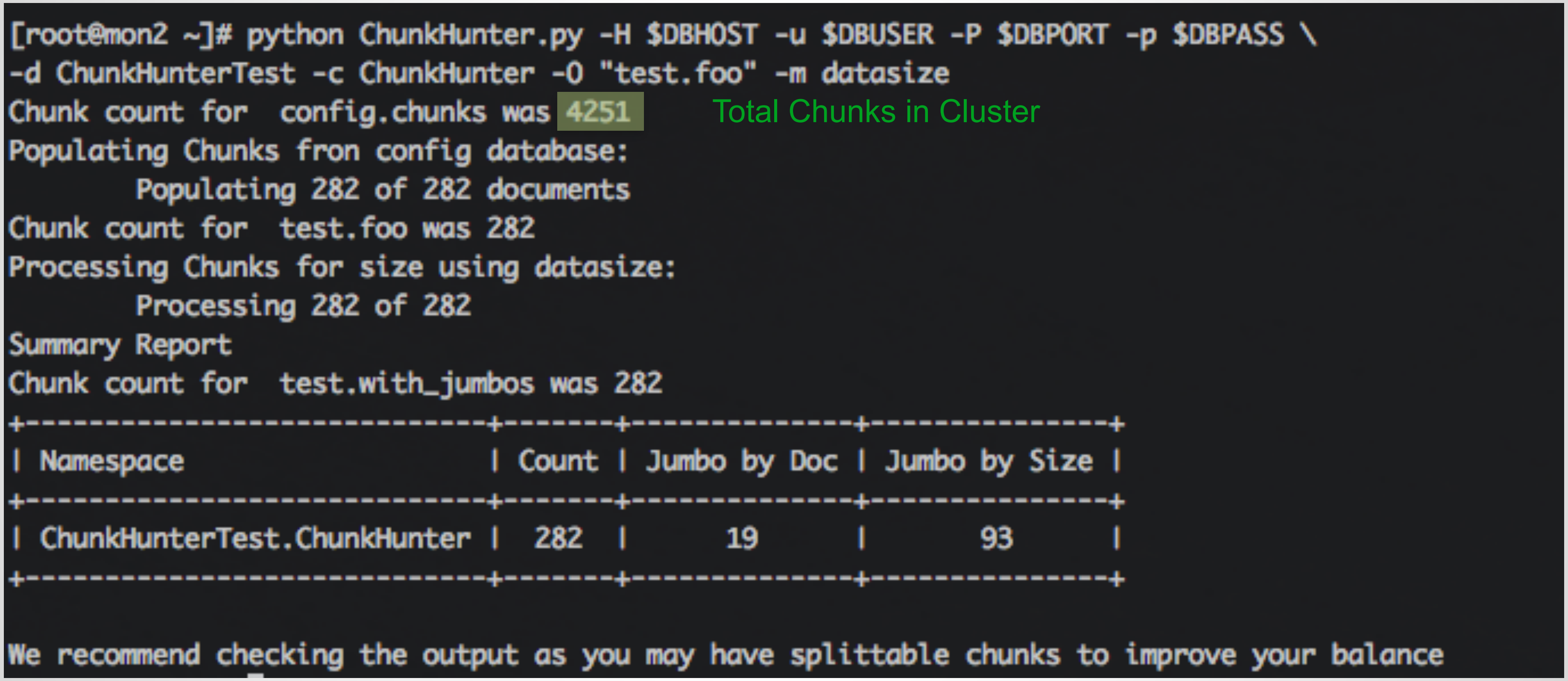
- ChunkHunter.py (On GitHub http://bit.ly/1yWQf9T)
  New community script from ObjectRocket to:
  a) copy chunks meta data to new collection
  b) scan each chunk there with dataSize or count
  c) output summary

Example ChunkHunter Output

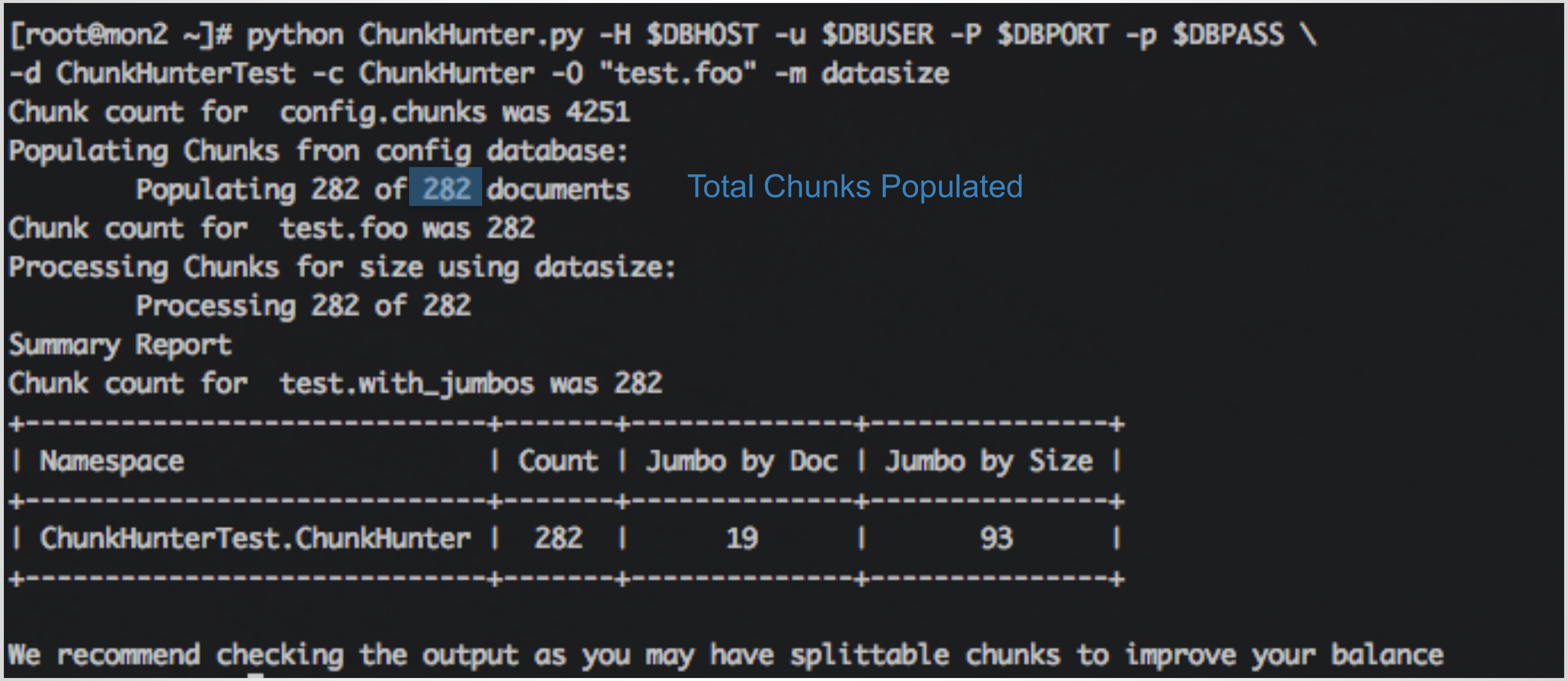


```
[root@mon2 ~]# python ChunkHunter.py -H $DBHOST -u $DBUSER -P $DBPORT -p $DBPASS \
-d ChunkHunterTest -c ChunkHunter -O "test.foo" -m datasize
Chunk count for  config.chunks was 4251
Populating Chunks fron config database:
        Populating 282 of 282 documents
Chunk count for  test.foo was 282
Processing Chunks for size using datasize:
        Processing 282 of 282
Summary Report
Chunk count for  test.with_jumbos was 282
+-----------------------------+-------+--------------+---------------+
| Namespace                   | Count | Jumbo by Doc | Jumbo by Size |
+-----------------------------+-------+--------------+---------------+
| ChunkHunterTest.ChunkHunter |  282  |      19      |       93      |
+-----------------------------+-------+--------------+---------------+

We recommend checking the output as you may have splittable chunks to improve your balance
```

Example ChunkHunter Output

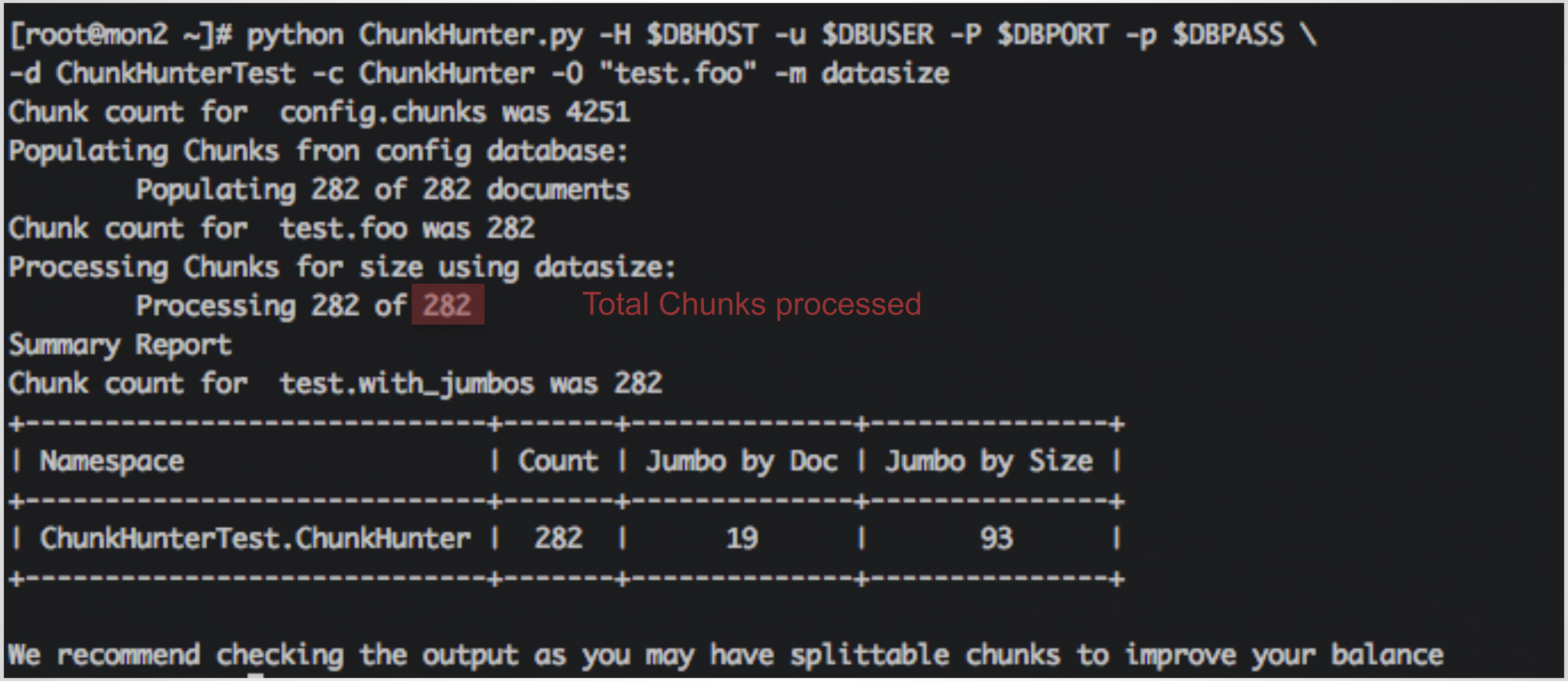

```
[root@mon2 ~]# python ChunkHunter.py -H $DBHOST -u $DBUSER -P $DBPORT -p $DBPASS \
-d ChunkHunterTest -c ChunkHunter -O "test.foo" -m datasize
Chunk count for  config.chunks was 4251
Populating Chunks fron config database:
        Populating 282 of 282 documents
Chunk count for  test.foo was 282
Processing Chunks for size using datasize:
        Processing 282 of 282
Summary Report
Chunk count for  test.with_jumbos was 282
+------------------------------+-------+-------------+--------------+
| Namespace                    | Count | Jumbo by Doc | Jumbo by Size |
+------------------------------+-------+-------------+--------------+
| ChunkHunterTest.ChunkHunter |  282  |      19     |      93      |
+------------------------------+-------+-------------+--------------+


We recommend checking the output as you may have splittable chunks to improve your balance
```

Example ChunkHunter Output



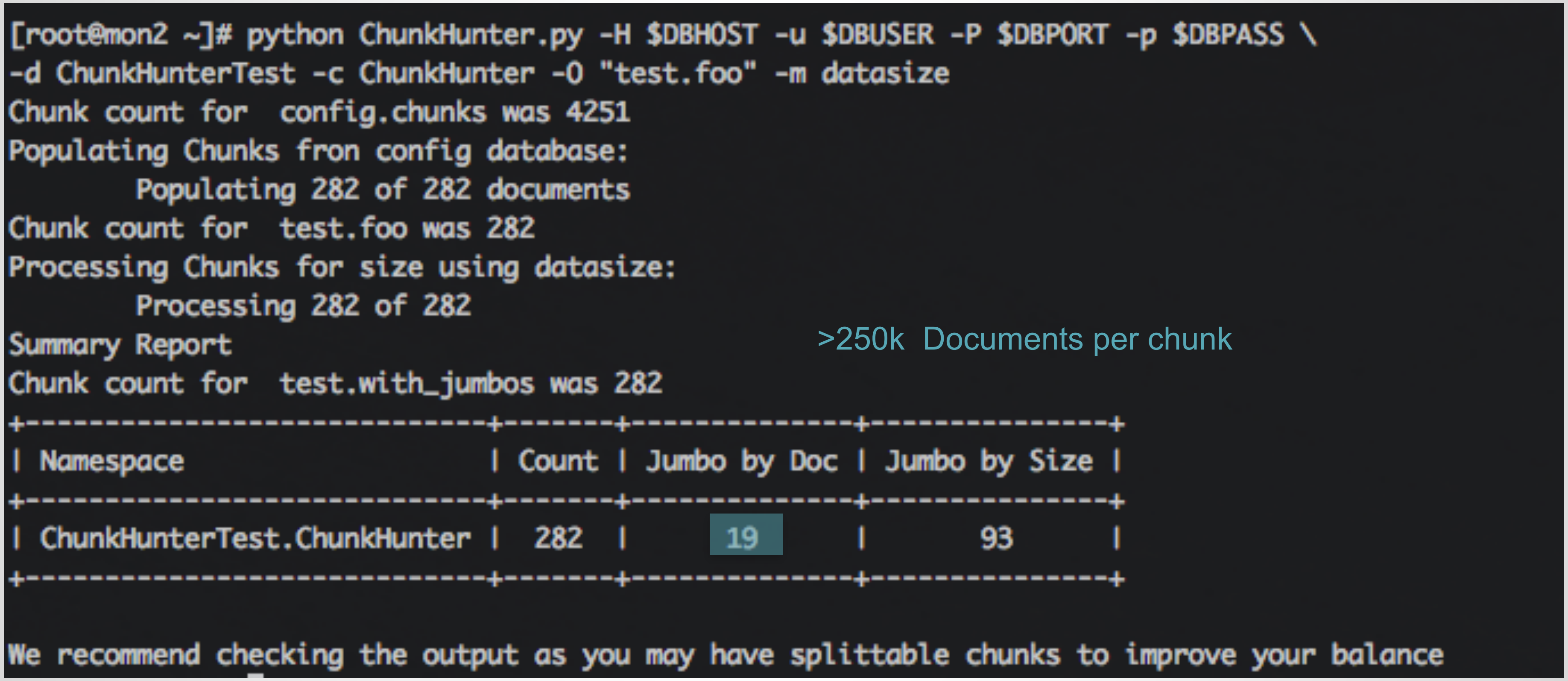
```
[root@mon2 ~]# python ChunkHunter.py -H $DBHOST -u $DBUSER -P $DBPORT -p $DBPASS \
-d ChunkHunterTest -c ChunkHunter -O "test.foo" -m datasize
Chunk count for  config.chunks was 4251
Populating Chunks fron config database:
        Populating 282 of 282 documents
Chunk count for  test.foo was 282
Processing Chunks for size using datasize:
        Processing 282 of 282
Summary Report
Chunk count for  test.with_jumbos was 282
+------------------------------+-------+-------------+---------------+
| Namespace                    | Count | Jumbo by Doc | Jumbo by Size |
+------------------------------+-------+-------------+---------------+
| ChunkHunterTest.ChunkHunter |  282  |      19     |       93      |
+------------------------------+-------+-------------+---------------+

We recommend checking the output as you may have splittable chunks to improve your balance
```

Example ChunkHunter Output



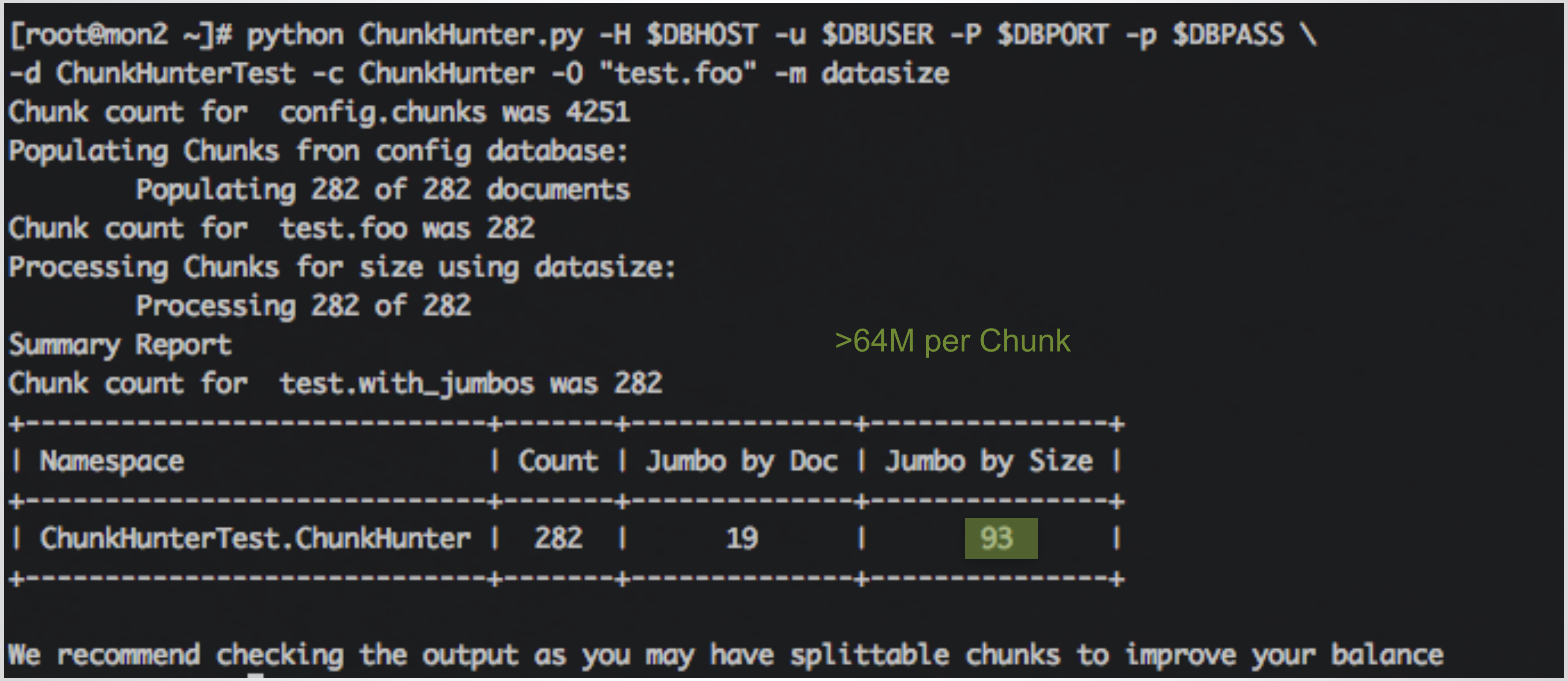
```
[root@mon2 ~]# python ChunkHunter.py -H $DBHOST -u $DBUSER -P $DBPORT -p $DBPASS \
-d ChunkHunterTest -c ChunkHunter -O "test.foo" -m datasize
Chunk count for  config.chunks was 4251
Populating Chunks fron config database:
        Populating 282 of 282 documents
Chunk count for  test.foo was 282
Processing Chunks for size using datasize:
        Processing 282 of 282
Summary Report
Chunk count for  test.with_jumbos was 282
+------------------------------+-------+-------------+---------------+
| Namespace                    | Count | Jumbo by Doc | Jumbo by Size |
+------------------------------+-------+-------------+---------------+
| ChunkHunterTest.ChunkHunter |  282  |      19      |       93      |
+------------------------------+-------+-------------+---------------+

We recommend checking the output as you may have splittable chunks to improve your balance
```

Example ChunkHunter Output



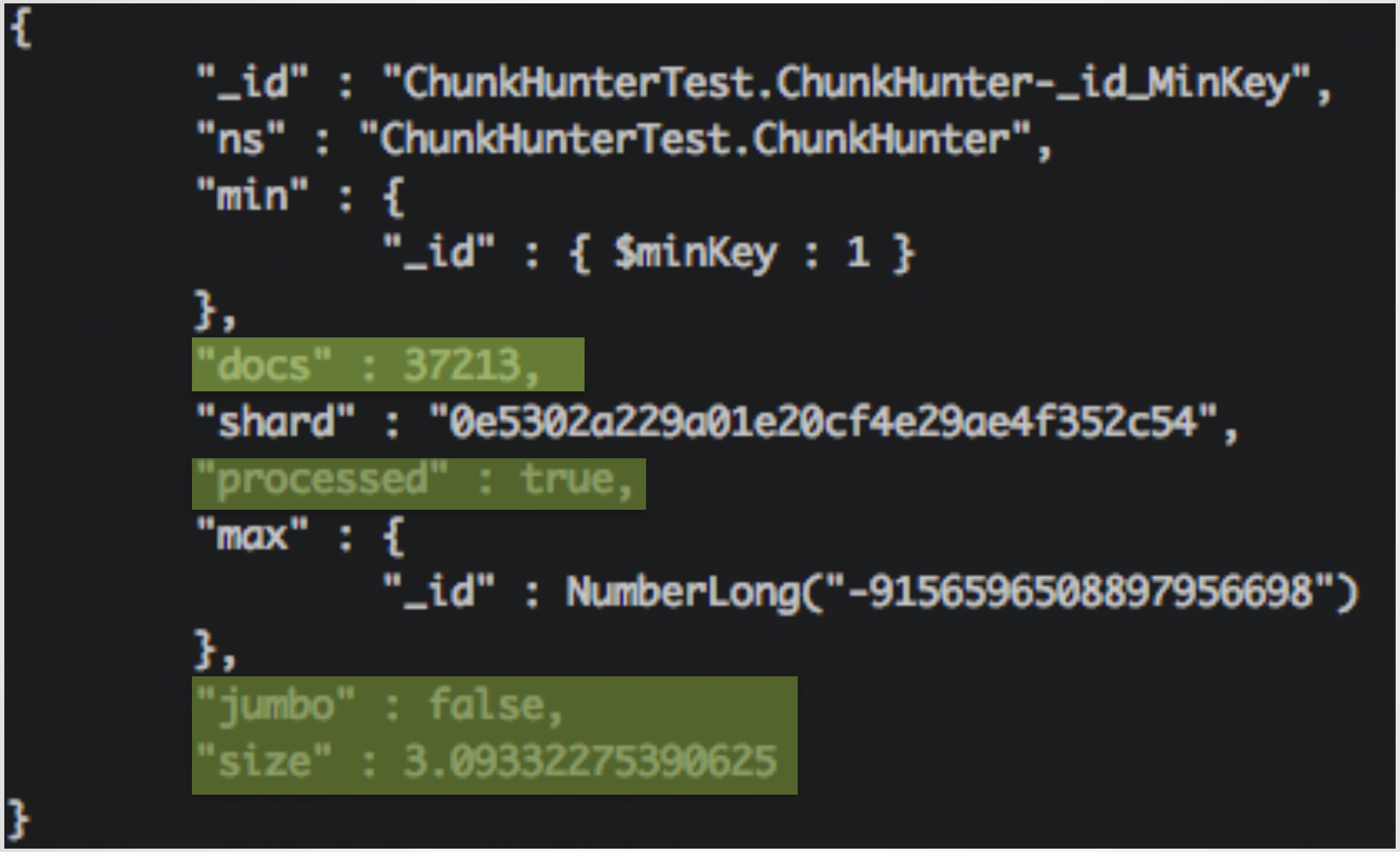
```
[root@mon2 ~]# python ChunkHunter.py -H $DBHOST -u $DBUSER -P $DBPORT -p $DBPASS \
-d ChunkHunterTest -c ChunkHunter -O "test.foo" -m datasize
Chunk count for  config.chunks was 4251
Populating Chunks fron config database:
        Populating 282 of 282 documents
Chunk count for  test.foo was 282
Processing Chunks for size using datasize:
        Processing 282 of 282
Summary Report
Chunk count for  test.with_jumbos was 282
+------------------------------+-------+-------------+---------------+
| Namespace                    | Count | Jumbo by Doc | Jumbo by Size |
+------------------------------+-------+-------------+---------------+
| ChunkHunterTest.ChunkHunter  |  282  |      19     |       93      |
+------------------------------+-------+-------------+---------------+


We recommend checking the output as you may have splittable chunks to improve your balance
```

Example ChunkHunter Output



```
[root@mon2 ~]# python ChunkHunter.py -H $DBHOST -u $DBUSER -P $DBPORT -p $DBPASS \
-d ChunkHunterTest -c ChunkHunter -O "test.foo" -m datasize
Chunk count for  config.chunks was 4251
Populating Chunks fron config database:
        Populating 282 of 282 documents
Chunk count for  test.foo was 282
Processing Chunks for size using datasize:
        Processing 282 of 282
Summary Report
Chunk count for  test.with_jumbos was 282
+------------------------------+-------+-------------+---------------+
| Namespace                    | Count | Jumbo by Doc | Jumbo by Size |
+------------------------------+-------+-------------+---------------+
| ChunkHunterTest.ChunkHunter |  282  |      19      |      93       |
+------------------------------+-------+-------------+---------------+

We recommend checking the output as you may have splittable chunks to improve your balance
```

What did we add to the doc?



```
{
        "_id" : "ChunkHunterTest.ChunkHunter-_id_MinKey",
        "ns" : "ChunkHunterTest.ChunkHunter",
        "min" : {
                "_id" : { $minKey : 1 }
        },
        "docs" : 37213,
        "shard" : "0e5302a229a01e20cf4e29ae4f352c54",
        "processed" : true,
        "max" : {
                "_id" : NumberLong("-9156596508897956698")
        },
        "jumbo" : false,
        "size" : 3.09332275390625
}
```

Some example queries might be

- Give me all Jumbo:true chunks
- Order Jumbo chunks by documents so I can split them
- Order Jumbo chunks by size so I can split them
- Aggregate Jumbo chunks to count them per shard
- Give me all Jumbo on Shard “X”

Find all these and more at:
http://bit.ly/1oYVGeJ

The Future…

Will be releasing tools to simplify:

- Manual Splits
- Moving of Chunks

These are not replacements for the balancer but ways to help it stay on track if things deviate. As opposed to waiting for an incident and fixing it in a panic.

Contact

@dmurphy_data
@objectrocket
david@objectrocket.com
https://www.objectrocket.com

WE ARE HIRING! (DBA,DEVOPS, and more)
https://www.objectrocket.com/careers

Resources:
ChunkHunter.py http://bit.ly/1yWQf9T
Chunk Hunter Example Scripts bit.ly/1oYVGeJ
Chunk Collection Queries http://bit.ly/112tXEx
Changelog Queries http://bit.ly/1oZlyqN

Presentations:
Kenny Gorman Sharding - bit.ly/1oXYDfm
Other Sharding Links - bit.ly/ZTtDI1