Indexer Clustering Fixups

Cluster recovery process

Da Xu | Engineering | Splunk
During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2017 Splunk Inc. All rights reserved.
Clustering Fixup Cycle
The Cluster Master (CM) remembers a list of buckets it needs to fix.

- Every time a bucket is modified, the CM will add the bucket into any to_fix lists that are needed.

- There are multiple to_fix lists, each corresponding to a different fixup:
  - Generation, Replication Factor, Search Factor, Excess Buckets, Summary, etc.

- These lists are an internal list of all the buckets that POTENTIALLY need fixing:
  - Very possible that nothing needs to be done - in which case they’ll be removed on the service() call (part 2).
var/log/splunk/metrics.log

- to_fix_rep_factor=104
  - buckets to check if we need/can schedule replications
- to_fix_added=116k to_fix_removed=116k
  - likely a re-add operation – a peer resynced its buckets (restart) and most buckets were added/removed from the to_fix list with little jobs scheduled
Fixup Lifecycle – service

- Periodically, the CM will run through its service() loop
  - Defaults to every second, see the setting of “service_interval”
- During the service() loop, the CM will iterate through ALL the bucket on all the to_fix lists, and schedule any jobs that it needs to for repairs.

```plaintext
function service()
{
  for bucket in to_fix_lists:
    schedule_jobs_to_fix(bucket) // maybe schedule a job?
}
```

- (not actually taken from splunk/clustering.cpp)
- If the bucket doesn’t need any jobs OR we’ve scheduled all jobs to fix a bucket, its removed from the to_fix list
Fixup Lifecycle – jobs

Once service() schedules jobs, the CM has a threadpool running and picking up all the scheduled jobs.

These jobs are generally REST POSTs to the cluster Indexers, telling them to run some command

<table>
<thead>
<tr>
<th>Job name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMRepJob</td>
<td>Replicate a bucket (includes recipient info)</td>
</tr>
<tr>
<td>CMChangeBucketJob_build</td>
<td>Make bucket searchable (RF copy -&gt; SF copy)</td>
</tr>
<tr>
<td>CMChangeMasks</td>
<td>Change bucket primaries (batch job, can change maybe buckets)</td>
</tr>
<tr>
<td>CMRollHotBucket</td>
<td>Roll a bucket from hot-&gt;warm</td>
</tr>
<tr>
<td>CMTruncJob</td>
<td>Truncate a bucket’s size (remove bucket will use this with size=0)</td>
</tr>
</tbody>
</table>
CMChangeMasksJob=57
- The CM ran 57 jobs to change bucket primaries amongst the indexers

CMRepJob=40
- The CM ran 40 jobs to start bucket replication jobs amongst the indexers (fixup RF)
Fixup Lifecycle – jobs

▶ All the jobs are ran asynchronously on separate threads
  • "executor_workers" specifies how many separate threads run these jobs (defaults to 10)
▶ These jobs generally POST to a specific Peer endpoint
  • ex: indexer01:8089/services/cluster/slave/buckets/main~1~/replicate
▶ If the job runs successfully
  • All is well! The bucket replicated, or got its primary changed, or etc.
▶ If the job encounters an error
  • The bucket is placed BACK into the to_fix list, so that we can fix it up again on the next service loop!
Clustering Fixup Cycle - examples
to_fix lists metrics

index=*_internal source=*_metrics.log name=cmmaster_service | timechart max(to_fix_gen) max(to_fix_rep_factor) max(to_fix_search_factor) span=30m

18 events (9/5/17 1:41:00.000 AM to 9/5/17 1:45:30.000 AM) No Event Sampling

Searchable

RF+SF Met
Thanks - Q&A

Don't forget to rate this session in the .conf2017 mobile app
APPENDIX Searches

1. `index=_internal host=MASTER source=*splunkd.log* CMRepJob running job | timechart count by job`  
   • Master jobs ran

2. `index=_internal source=*metrics.log* name=cmmaster group=jobs | timechart max(CM*)`  
   • Master jobs metrics

3. `index=_internal source=*metrics.log* *fix* host=MASTER | timechart max(to_fix_*)`  
   • to_fix list sizes

4. `index=_internal source=*metrics.log* group=subtask_seconds name=cmmaster | timechart max(service)`  
   • Master time spent calling service() in between previous log to metrics.log (every 30s)