Indexer clustering basics, internals & general debugging

Dhruva Kumar Bhagi

dbhagi@splunk.com

Sr. software engineer

Splunk Inc.

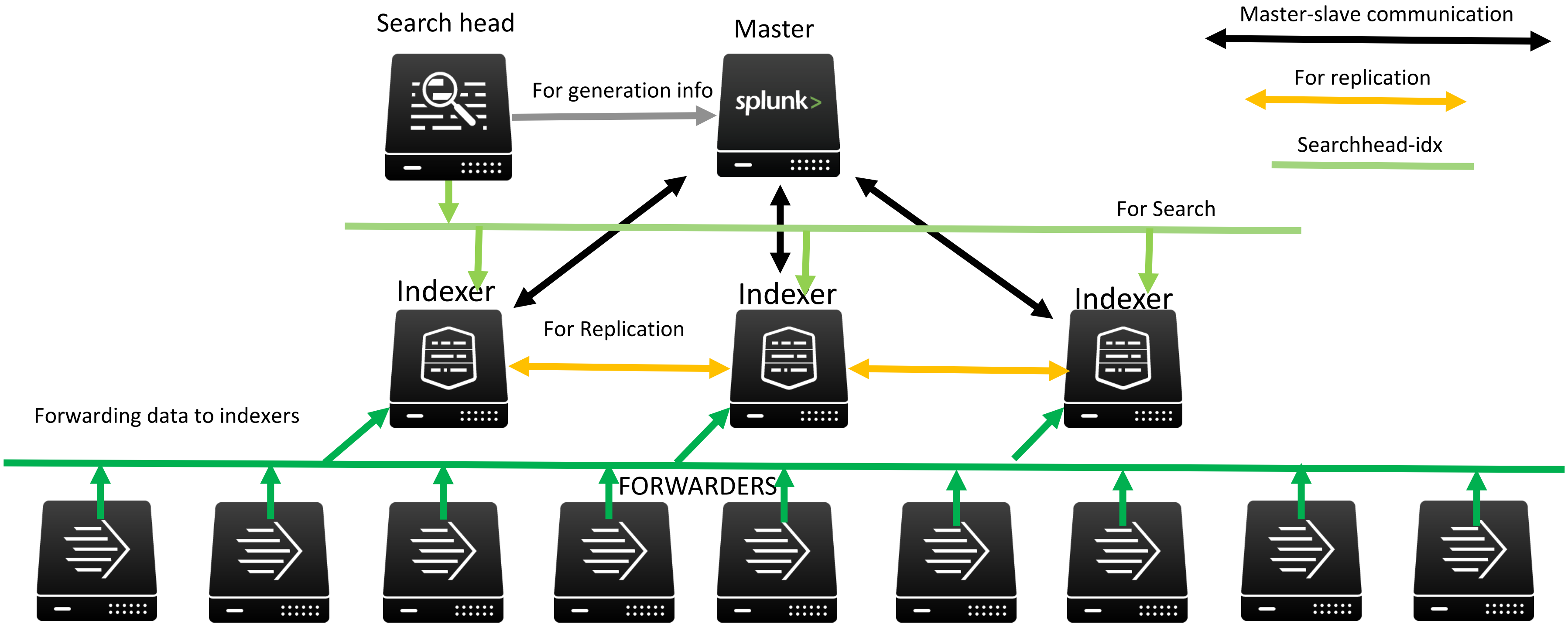
.conf2016

splunk>

Disclaimer

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 the 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.

Indexer cluster topology



Why indexer clustering

- **Data availability**: Your system can tolerate downed indexers without loosing data or access to the data
- **Disaster recovery**: With multisite clustering, your system can tolerate the failure of an entire data center
- **Search affinity**: With multisite clustering, Search heads can access the data through their local sites thereby improving search performance by lowering network latency
- **Other advantages**: uniform configuration across indexers, ease of management & monitoring of the indexers

Parts of the cluster

- **Cluster Master**
  - Manages the cluster activities
  - Maintains an in-memory state of all the peers & their corresponding buckets, configs
  - Orchestrates remedial activities during peer failures
  - Tells search heads where to search
- **Cluster Peer (Indexer)**
  - Receive and index incoming data (typically from forwarders)
  - Replicate data to other peers for data availability
  - Respond to the incoming searches by providing search results
  - Update cluster master on any state change (peer, buckets, configs etc.)
- **Search head**
  - Runs & coordinates searches & aggregates the search results coming from indexers
  - Periodically interacts with cluster master for generation updates

Communication amongst members

Cluster master & peers communicate over REST endpoints.

Few Examples:

- **Peers->Master:**
  - /services/cluster/master/peers
    - Add peer to cluster
    - Heartbeat to master
  - /services/cluster/master/buckets
    - Notify master on bucket creation & removal
    - Notify master on bucket state changes
- **Master->Peers:**
  - /services/cluster/slave/buckets
    - Change primaries
    - Become searchable/unsearchable
- **Search head->Master**:
  - cluster/master/generation - To get the latest generation information

event=addPeer

- Peer joins the cluster by executing an event called ‘**addPeer**’ which is a REST call to CM (services/cluster/master/peers)
- This happens on peer startup.
- On AddPeer request, peer reports its entire state to cluster master.
  - reports all its buckets and corresponding states
  - active_bundle_id, latest_bundle_id, mgmt_port, GUID, replication_port
  - add_type = Initial-Add |ReAdd
- Master stores entire peer’s state in its memory

event=addPeer

- Slave logs: 08-02-2016 15:54:06.098 -0700 INFO CMSlave - **event=addPeer** status=success request: AddPeerRequest: { }
- Up on successful addPeer, master also logs to its splunkd.log
  - 08-02-2016 15:54:06.094 -0700 INFO CMMaster - **event=addPeer** guid=F1B6E8F0-002A-4947-83CA-0A5BC56E0A53 peer_name=slave1 AddPeerRequest: {} bucket_count=4
- On addPeer success, master commits a new generation.
  - CMMaster - **committing gen**=1 numpeers=1 requesterReason=addPeerSuccess guid=F1B6E8F0-002A-4947-83CA-0A5BC56E0A53 lastCompleteGenId=0
- When enough replication_factor # of peers join the cluster, cluster transitions into **indexing ready state**.

Heartbeats

- Heart beating is a way cluster master & peer tell each other that they are up and running
- Heartbeat happens over REST endpoint (cluster/master/peers)
- Once peer registers to master, it sends out heartbeat request to master once in every **heartbeat_period** seconds (defaults to 1)
- Master responds back to the heartbeat request indicating its up
- Master and peer exchange some basic information (like bundleId’s, peer states etc.) over the heartbeats.

Heartbeats

- More the # of peers, more the heartbeat requests master receives and respond to
- For relatively large clusters (with >50 peers or 200k+ buckets), its recommended to adjust **heartbeat_period** value to 5-30.
- Master marks a peer as “**Down**” if it hasn’t received heartbeat for **heartbeat_timeout** period (defaults to 60 seconds)
- For relatively large clusters, its recommended to adjust this value to 20x-60x of **heartbeat_period**
- **FYI:** Its recommended to also adjust **restart_timeout** as the peer load (like bucket/summary/job count) goes up

Config management in the cluster

.conf2016

splunk>

Cluster bundles

- **Bundle** is basically a set of updated configuration files (mostly indexes.conf, props.conf, transforms.conf etc) spread over different apps distributed to cluster peers from cluster master
- Its just the content under $SPLUNK_HOME/ etc /**master_apps**
- In order to push a new bundle, update your master_apps content & run '**splunk apply cluster-bundle [--skip-validation]**'

Cluster bundles

Bundle push is a multi step process

- **Creation**
  - Happens at cluster master
  - Involves creating the bundle tar ball & calculating the checksum
  - Master does minimal config validation while creating the bundle
  - Master updates its **latest_bundle_id** to the new bundle checksum
- **Validation**
  - Happens at the cluster peers
  - Peers detect new **latest_bundle_id** from master & performs validation
  - Validation involves downloading the bundle & actually validating the configs
  - Peer reports the outcome of the validation to cluster master
  - Master reverts its **latest_bundle_id** to old bundle if any peer reports error

Cluster bundles

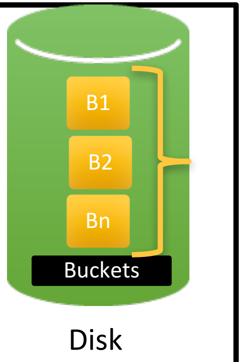
- **Reload (or) Restart**
  - Depending on the contents of the bundle, cluster peers determine if they can accept the new bundle without a restart (by just reloading)
  - Peer reports that bundle needs restart, CM then issues rolling-restart of cluster peers for the new bundle to take into effect.

FYI: Its not recommended to change cluster peer configurations (like indexes, props, transforms etc.) locally at the peers. All the configs should come from cluster master. This guarantees uniformity of the configuration among cluster members.

BUCKETS

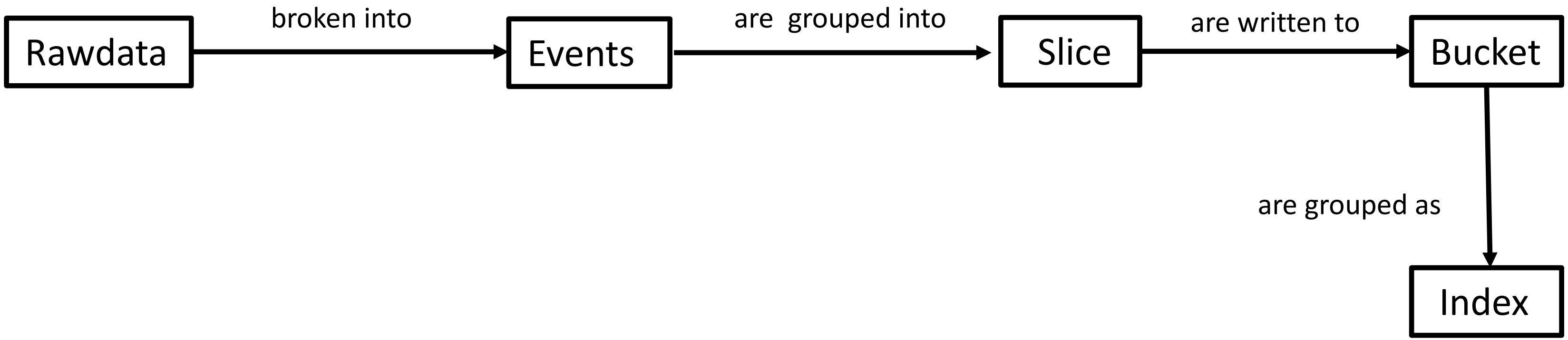
Buckets

Buckets are created on the indexer (cluster peer).



Flow of bucket creation:

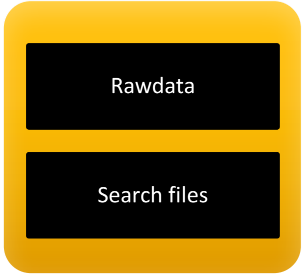
- Indexer receives raw-data and transforms them into events
- Groups the events into a bucket & generates index for each keyword
- Groups buckets into a logical/physical partition called index
- Typical data flow hierarchy:



Rawdata → broken into → Events → are grouped into → Slice → are written to → Bucket → are grouped as → Index

Buckets

- **Bucket** is usually a unit of data the cluster is aware of
- For data availability, each indexer replicates its buckets
- Replication is of two types:
  - Streaming replication (for hot buckets)
  - Non-streaming replication (for warm|cold buckets)
- Buckets can be **searchable** or **unsearchable**
- Among multiple searchable copies, master picks one copy as "**primary**"
- Peers only serve data from **primary** buckets to the search
- Cluster peer notifies cluster master upon every state change of its bucket(s) so that master stays up to date



Bucket

Buckets

- **More buckets means more work**
  - Since bucket is the unit of the data that cluster handles, Most of the work/communication in the cluster is related to buckets
  - Some examples of bucket related work:
    - Bucket creation
    - **Bucket state changes**
      - Hot -> warm, Warm -> cold, Cold -> frozen
      - Searchable -> unsearchable, Unsearchable -> searchable
      - Changing primary mask (needs generation commit)
    - Bucket truncation
    - Bucket deletion
    - Handling replications
    - Handling success|failures|errors of various bucket transitions & transactions

Reduced disk space for aged buckets

- **Searchable** buckets occupy more disk space due the substantial storage requirements of **tsidx/index** files
- Infrequently searched old/aged searchable buckets size can be greatly reduced with tsidx reduction at the cost of significant search performance
- Reduced tsidx files are one-third to two-third smaller than the original ones
- Each indexer reduces its searchable copies on its own
- By default tsidx reduction is **disabled** (enableTsidxReduction=false)
- **NOTE**: tstats & typeahead commands won’t work on reduced buckets

Master service & fixups

CM service

- Cluster master executes its **service()** call once in every few seconds.
- Master schedules all its pending work in this service call.
  - Work involves:
    - Responding to node failures (or) state transitions
    - Running **fixup** jobs (to move primaries & meet factors)
- More the # of peers & # of buckets, more the work to do in the service call
- Spike in the **service()** duration during node failure if peer has lot of buckets
- The interval between two successive service calls can be configured using config “**service_interval**”
- The new default value of **service_interval = 0**, which means auto mode

CM service

- In auto mode, next **service** call is scheduled based on duration of the current service call (interval is capped by **max_auto_service_interval**)

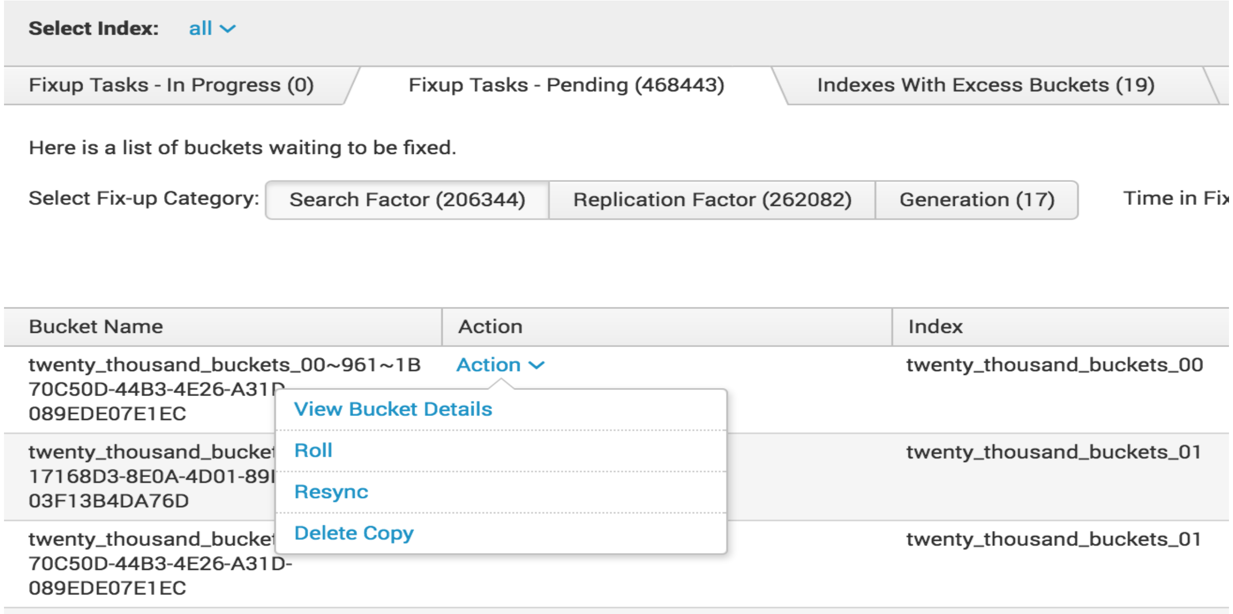
- Alternatively, you can manually tune **service_interval** as the cluster grows in size (along with heartbeat & restart timeouts)

Fixups

- CM iterates through list of buckets in its **fixup** list attempting to fix them
- It involves re-assigning primaries, creating replication copies, making buckets searchable, rolling buckets, freezing buckets etc.
- Assuming sf > 1, primary fixups are expected to finish faster without delay
- **cluster/master/fixup** end point displays buckets in the **fixup** list by ’**level**’ (level=replication_factor, search_factor etc.)
- Its expected for the master to take sometime to fix rf/sf if there are lot of buckets in fixup & this can be carefully controlled by tuning **max_peer_rep_load**(5) & **max_peer_build_load**(2)
- Fixup supports a **’filter**’ option which allows filtering buckets based on some condition
  - For example /services/cluster/master/fixup?level=replication_factor&filter=minutes_in_fixup>100 lists buckets stuck in fixup for more than 100 minutes – Something wrong with this bucket?

FYI: CM does not perform rep & search fixups in **maintenance mode**, this can be helpful to avoid unnecessary replications during planned downtime of peer(s)

UI actions on buckets stuck in fixup



Select Index: all

Fixup Tasks - In Progress (0)	Fixup Tasks - Pending (468443)	Indexes With Excess Buckets (19)

Here is a list of buckets waiting to be fixed.

Select Fix-up Category: Search Factor (206344) | Replication Factor (262082) | Generation (17) | Time in Fix

Bucket Name	Action	Index
twenty_thousand_buckets_00~961~1B 70C50D-44B3-4E26-A31D 089EDE07E1EC	Action	twenty_thousand_buckets_00
twenty_thousand_bucke 17168D3-8E0A-4D01-89I 03F13B4DA76D		twenty_thousand_buckets_01
twenty_thousand_bucke 70C50D-44B3-4E26-A31D- 089EDE07E1EC		twenty_thousand_buckets_01

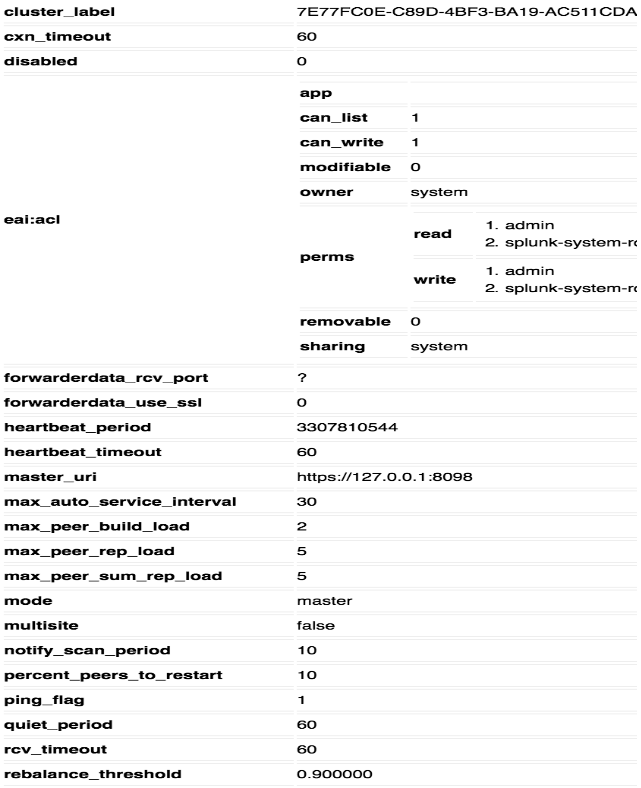
- **Note**: Be careful with ’Delete copy’ especially if there is only one copy

Cluster config/info

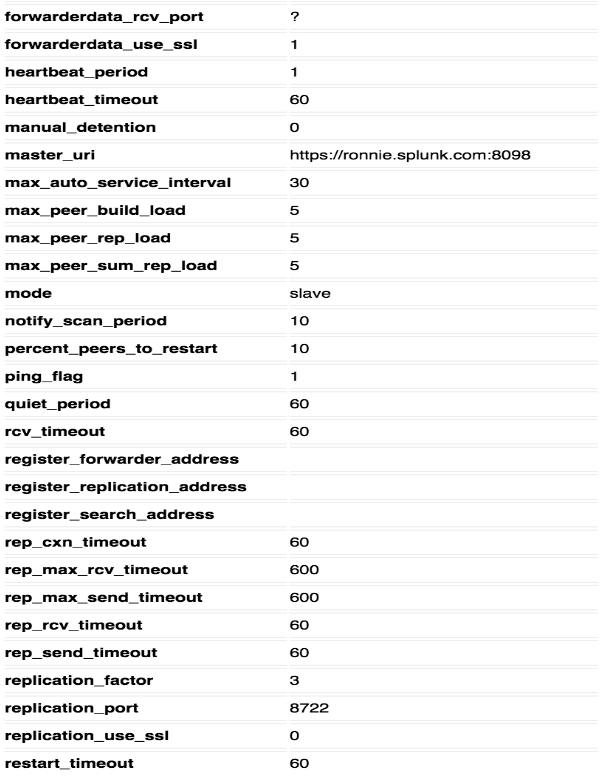
.conf2016

splunk>

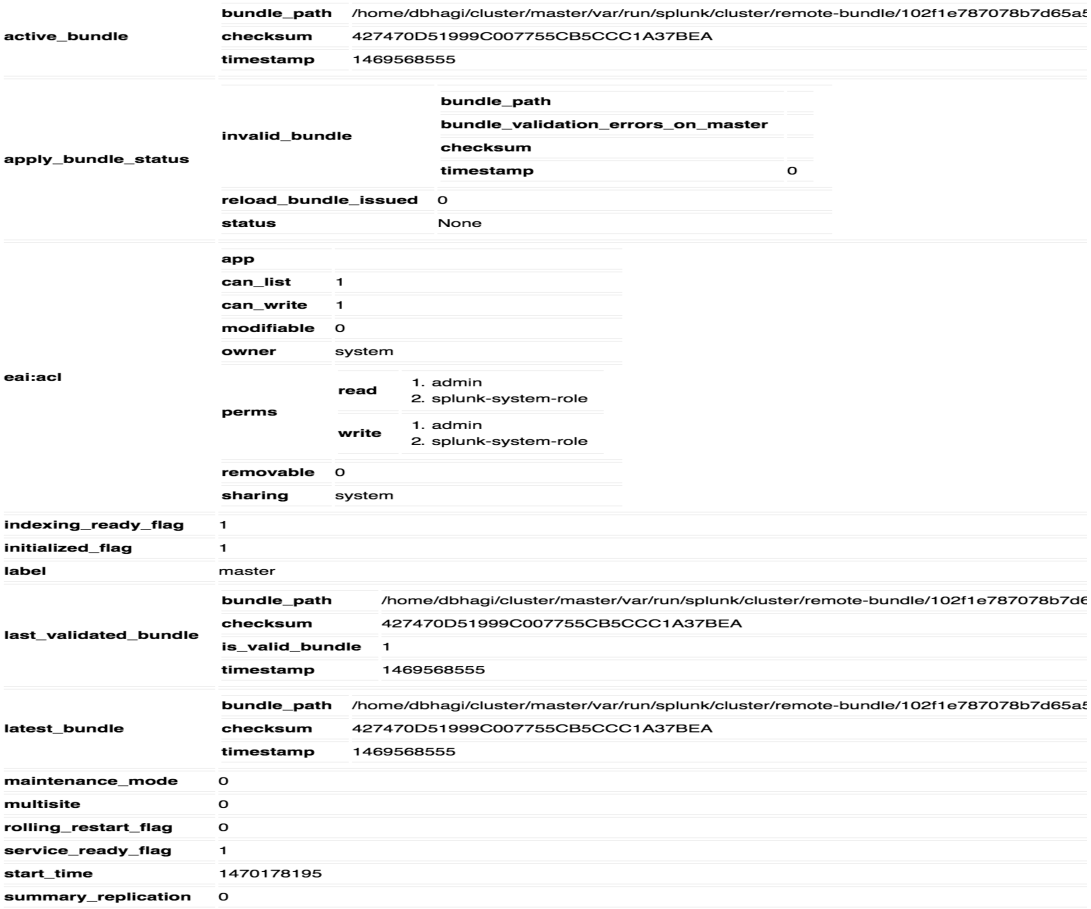
- **services/cluster/config** on master & peers lists clustering configuration



cluster_label	7E77FC0E-C89D-4BF3-BA19-AC511CDA		
cxn_timeout	60		
disabled	0		
eai:acl	app		
	can_list	1	
	can_write	1	
	modifiable	0	
	owner	system	
	perms	read	1. admin 2. splunk-system-r
		write	1. admin 2. splunk-system-r
	removable	0	
	sharing	system	
forwarderdata_rcv_port	?		
forwarderdata_use_ssl	0		
heartbeat_period	3307810544		
heartbeat_timeout	60		
master_uri	https://127.0.0.1:8098		
max_auto_service_interval	30		
max_peer_build_load	2		
max_peer_rep_load	5		
max_peer_sum_rep_load	5		
mode	master		
multisite	false		
notify_scan_period	10		
percent_peers_to_restart	10		
ping_flag	1		
quiet_period	60		
rcv_timeout	60		
rebalance_threshold	0.900000		



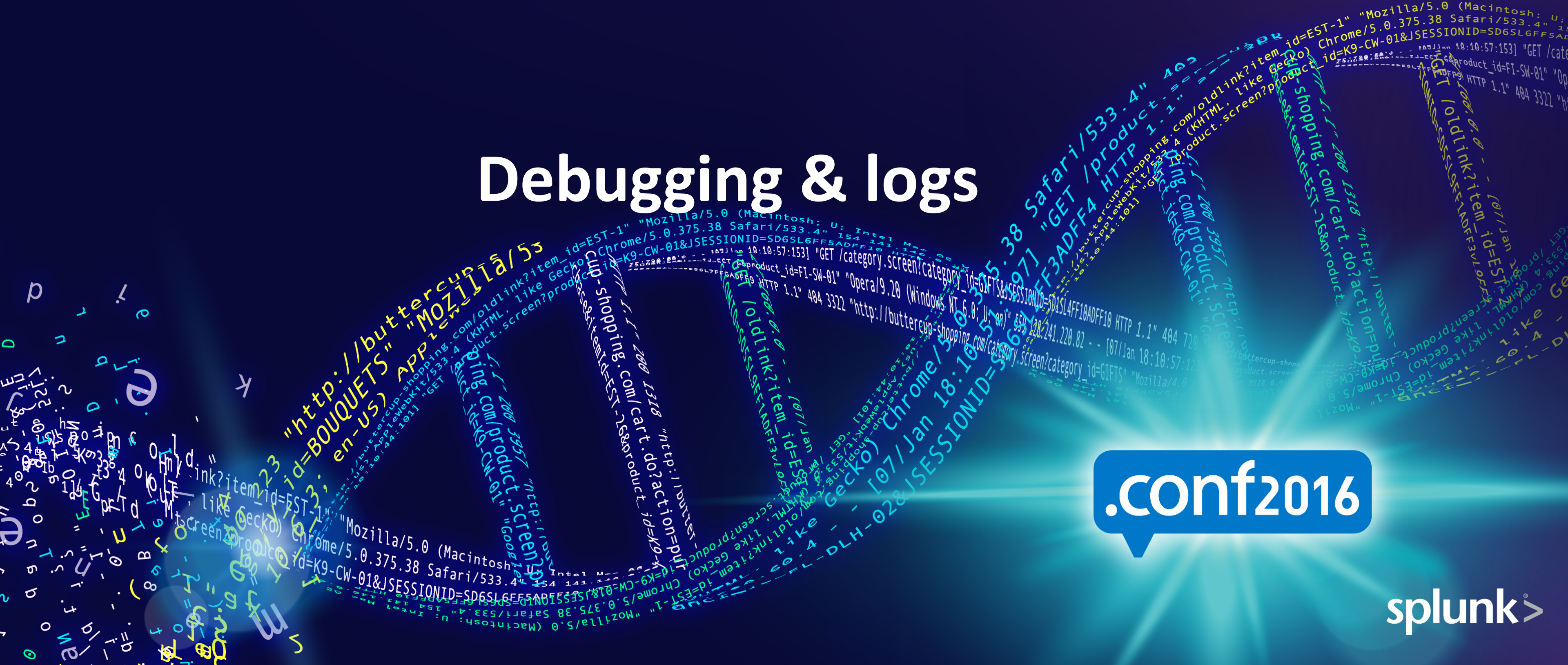
forwarderdata_rcv_port	?
forwarderdata_use_ssl	1
heartbeat_period	1
heartbeat_timeout	60
manual_detention	0
master_uri	https://ronnie.splunk.com:8098
max_auto_service_interval	30
max_peer_build_load	5
max_peer_rep_load	5
max_peer_sum_rep_load	5
mode	slave
notify_scan_period	10
percent_peers_to_restart	10
ping_flag	1
quiet_period	60
rcv_timeout	60
register_forwarder_address	
register_replication_address	
register_search_address	
rep_cxn_timeout	60
rep_max_rcv_timeout	600
rep_max_send_timeout	600
rep_rcv_timeout	60
rep_send_timeout	60
replication_factor	3
replication_port	8722
replication_use_ssl	0
restart_timeout	60



active_bundle	bundle_path	/home/dbhagi/cluster/master/var/run/splunk/cluster/remote-bundle/102f1e787078b7d65a5	
	checksum	427470D51999C007755CB5CCC1A37BEA	
	timestamp	1469568555	
apply_bundle_status	invalid_bundle	bundle_path	
		bundle_validation_errors_on_master	
		checksum	
		timestamp	0
	reload_bundle_issued	0	
	status	None	
eai:acl	app		
	can_list	1	
	can_write	1	
	modifiable	0	
	owner	system	
	perms	read	1. admin 2. splunk-system-role
		write	1. admin 2. splunk-system-role
	removable	0	
	sharing	system	
indexing_ready_flag	1		
initialized_flag	1		
label	master		
last_validated_bundle	bundle_path	/home/dbhagi/cluster/master/var/run/splunk/cluster/remote-bundle/102f1e787078b7d6	
	checksum	427470D51999C007755CB5CCC1A37BEA	
	is_valid_bundle	1	
	timestamp	1469568555	
latest_bundle	bundle_path	/home/dbhagi/cluster/master/var/run/splunk/cluster/remote-bundle/102f1e787078b7d65a5	
	checksum	427470D51999C007755CB5CCC1A37BEA	
	timestamp	1469568555	
maintenance_mode	0		
multisite	0		
rolling_restart_flag	0		
service_ready_flag	1		
start_time	1470178195		
summary_replication	0		

services/cluster/{master|slave}/info
Displays node configuration

Debugging & logs



.conf2016

splunk>

Index=_internal

- **_internal** index is the source for all the activity of splunkd
- Few log files to look at (or) correlate
  - source=***splunkd.log*** : to get an overview of what splunkd is doing
  - source=***splunkd_access.log****: to see all incoming REST calls & response codes
  - Source=***metrics.log****: to see metrics about how splunk is performing (different throughputs, queue sizes, response times, jobs count etc.)

Clustering related logs

- Look for **WARN/ERROR's** in following clustering components to get an overview of what went wrong when things go unexpected
- **Few components at cluster master:**
  - CMMaster – handles general cluster master functionality
  - CMPeer – handles a particular slave/peer specific work
  - CMBundleMgr – handles cluster bundle related functionality
  - CMRepJob – handles any replication related jobs/functionality
  - CMBucket – represents a bucket
- **Few Components at cluster peer:**
  - CMSlave – handles all the general slave/peer functionality
  - CMBundleMgr – handles slave bundle related functionality
  - BucketReplicator (send side), S2SFileReceiver (receive side) – Replicating buckets

Logs related to buckets

- Search by bid (index~0~1108~10BBFD2B-BDF8-411B-B574-FEAF37D6F486) helps understand/trace more about what went wrong with a particular bucket
- Most of the internal logs usually gets rotated fast in the production clusters so ‘**splunk diag**’ might not have any/all the information related to a particular bad bucket
- Exporting search results on a bucket id (like index=_internal {source=*splunkd.log*} BUCKET_ID) helps us understand more about what went wrong with a particular bucket

Recent enhancements

Recent enhancements

- **Scaling master & peers to be able to handle larger bucket volumes**
  - Batching jobs, reducing restarts, optimize/eliminate expensive operations, reducing disk scans
- **Better failure recovery when things go wrong**
  - Auto recover from state inconsistencies b/w master & peers, Provide options to take actions on any anomalous bucket states
- **Data Rebalancing** for balanced data & search load distribution
- **Summary replication** to reduce io & cpu spikes due to summary regeneration on node failures
- **Tsidx reduction** for reduce storage costs

THANK YOU

.conf2016

splunk>