Data collection for debugging Splunk scheduler issues

A scheduler issue may be described as:
- reduced number of completed scheduled searches running during certain periods
- scheduler locks up and doesn't run any scheduled searches for a period of time
- high number of skipped/deferred scheduled searches

1.) Set **SavedSplunker** on the search head cluster captain to DEBUG while the issue is occurring:

a.) This can be enabled via the UI on the captain node (or if standalone SH , on the single SH) and will take effect immediately but will not persist a restart of Splunk

```
>settings>server settings>server logging>SavedSplunker > DEBUG
```

b.) This can also be set in log.cfg however it will require a restart of Splunk to take effect and **will** persist future restarts. If Splunk is restarted , make sure the captain node comes back up with SavedSplunker in DEBUG as captaincy may shift to another node after a restart.

$SPLUNK_HOME/etc/log.cfg

```
#splunkd search scheduler log#:

#appender.scheduler.maxBackupIndex=5

appender.scheduler.maxBackupIndex=25

#category.SavedSplunker=INFO,scheduler

category.SavedSplunker=DEBUG,scheduler
```

c.) To determine which member is the shc captain you can perform one of the following:
- From the monitoring console (within the splunk_monitoring_console app context):

```
|rest splunk_server_group=dmc_group_search_head splunk_server_group="
dmc_searchheadclustergroup_<shcluster_label>"  /services/shcluster
/status|head 1|fields captain.label|rename captain.label AS
captain_label
```

- or from any SHC member cli:

```
./splunk show shcluster-status
```

```
[root@rplinux08 bin]# ./splunk show shcluster-status

 Captain:
                                dynamic_captain : 1
                                elected_captain : Tue Mar  8 16:01:43
2022
                                             id : A4DC6068-1149-459E-
87CD-34A834C275AC
                              initialized_flag : 1
                                         label : rplinux-SH1
                                      mgmt_uri : https://rplinux08.sv.
splunk.com:8089
                            min_peers_joined_flag : 1
                             rolling_restart_flag : 0
                               service_ready_flag : 1
```

2.) collect pstacks on the shc captain main splunkd pid while observing the issue.
The script below will default to stack collection every .5 sec and 1000 samples which is ok.

https://github.com/rephillips/debugging_splunk

^ requires root access to the server

note: if the issue is too intermittent to catch or doesn't persist long enough to collect pstacks manually we can use watchdog

use watchdog to generate pstacks when any thread goes goes unresponsive for (responseTimeout) ms
pro: stacks collected automatically
con: stacks may be collected when threads other than schedulerThread are unresponsive, therefore we may have stacks collected that do not lineup with the scheduler outage

on search head cluster captain:

server.conf

```
[watchdog]
actions = pstacks
responseTimeout = 100
actionsInterval = 1

[watchdogaction:pstacks]
maxStacksPerBlock = 500

restart splunk
```

3.) once pstack collection is finished generate a diag on the captain (or standalone SH if not a SHC) and set SavedSplunker back to INFO

a.) If the behavior is such that the scheduler is locked and no searches complete successfully you may want to consider running an alert from the Monitoring Console instance so the admin is notified and can manually kickoff the pstack collection script:

run a scheduled search with email alert from the MC every 1m:

```
index=_internal source=*scheduler.log  host IN(*shc*)
status="completed" OR status="success" | stats count by status | search
count=0
```

when condition is met where completed searches =0 (indicating scheduler may be hung) , run a script `collect-stacks.sh` https://github.com/rephillips/debugging_splunk on the captain to collect pstacks.

b.) also generate a diag on all SHC members and 1 indexer from each unique index cluster if there are more than 1 index clusters.

note:

If there are too many SHs in the SHC where generating a diag on every SHC member is unreasonable you can export the logs from the members by running the following searches and exporting to csv and attaching to the support ticket:

log export from all SHC members where host IN(*shc*) includes all shc members
determine scheduler outage time and set earliest / latest to be 1 hour before / 1 hour after outage:

```
index=_internal sourcetype=splunkd source=*splunkd.log host IN(*shc*)
earliest=07/12/2022:12:00:00 latest=07/12/2022:15:00:00  | table _time
host sourcetype component _raw

index=_internal sourcetype=splunkd source=*metrics.log host IN(*shc*)
earliest=07/12/2022:12:00:00 latest=07/12/2022:15:00:00  | table _time
host sourcetype component _raw

index=_internal sourcetype=scheduler source=*scheduler.log host IN
(*shc*) earliest=07/12/2022:12:00:00 latest=07/12/2022:15:00:00 | table
_time host sourcetype component _raw

index=_audit sourcetype=audittrail  host IN(*shc*) earliest=07/12/2022:
12:00:00 latest=07/12/2022:15:00:00 | table _time host sourcetype _raw

index=_internal sourcetype=splunkd_access source=*splunkd_access.log
host IN(*shc*) earliest=07/12/2022:12:00:00 latest=07/12/2022:15:00:00
| table _time host sourcetype  clientip method status user spent bytes
uri*   _raw

index=_introspection sourcetype=splunk_resource_usage
source=*resource_usage.log host IN(*shc*) earliest=07/12/2022:12:00:00
latest=07/12/2022:15:00:00 | table _time host sourcetype component data.
* _raw

index=_internal sourcetype=splunkd source=*watchdog.log host IN(*shc*)
earliest=07/12/2022:12:00:00 latest=07/12/2022:15:00:00 | table _time
host sourcetype component *thread* _raw
```

This will help Splunk Support have the best visibility into the issue.

4.) Upload files to Support case.