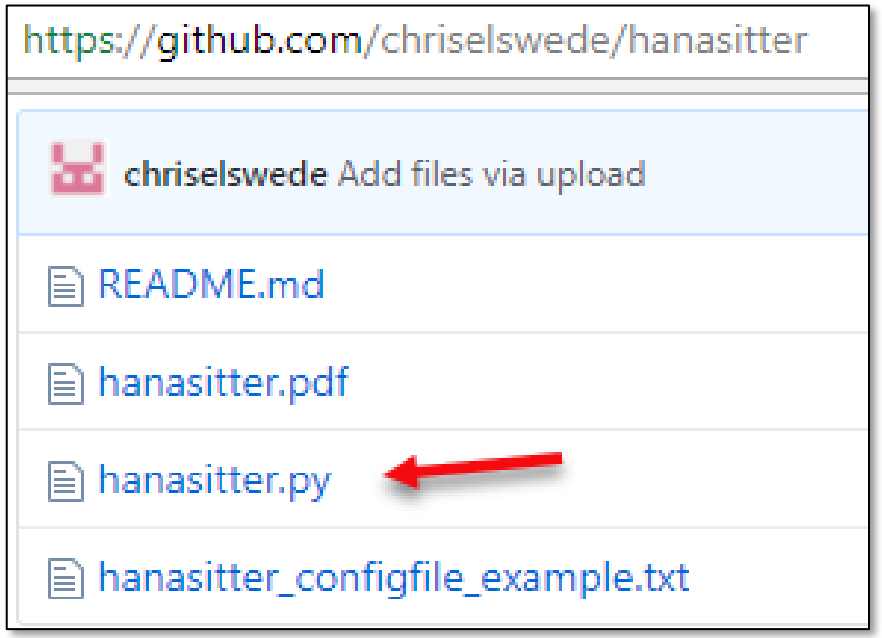
HANASitter – SAP Note 2399979

SAP Note 2399979 presents a tool that can help with monitoring tasks

2399979 - How-To: Configuring automatic SAP HANA Data Collection with SAP HANASitter

- It is a python script to be downloaded from https://github.com/chriselswede/hanasitter
- It is intended to be executed as <sid>adm on your SAP HANA Server
  (since then the proper python version is already in your path,
  installed together with SAP HANA)
- It connects via host, port and DB user, provided in hdbuserstore



Monitoring

HANASitter – using hdbuserstore

Host, port and DB user needs to be provided in the hdbuserstore:

```
mo-fc8d991e0:~> hdbuserstore SET HANASITTER1KEY mo-fc8d991e0:30015 HANASITTER1 PassWord1
mo-fc8d991e0:~> hdbuserstore LIST
DATA FILE       : /usr/sap/CH0/home/.hdb/mo-fc8d991e0/SSFS_HDB.DAT
KEY FILE        : /usr/sap/CH0/home/.hdb/mo-fc8d991e0/SSFS_HDB.KEY

KEY HANASITTER1KEY
  ENV : mo-fc8d991e0:30015
  USER: HANASITTER1
```

Then the hanasitter can connect using the info stored in hdbuserstore:

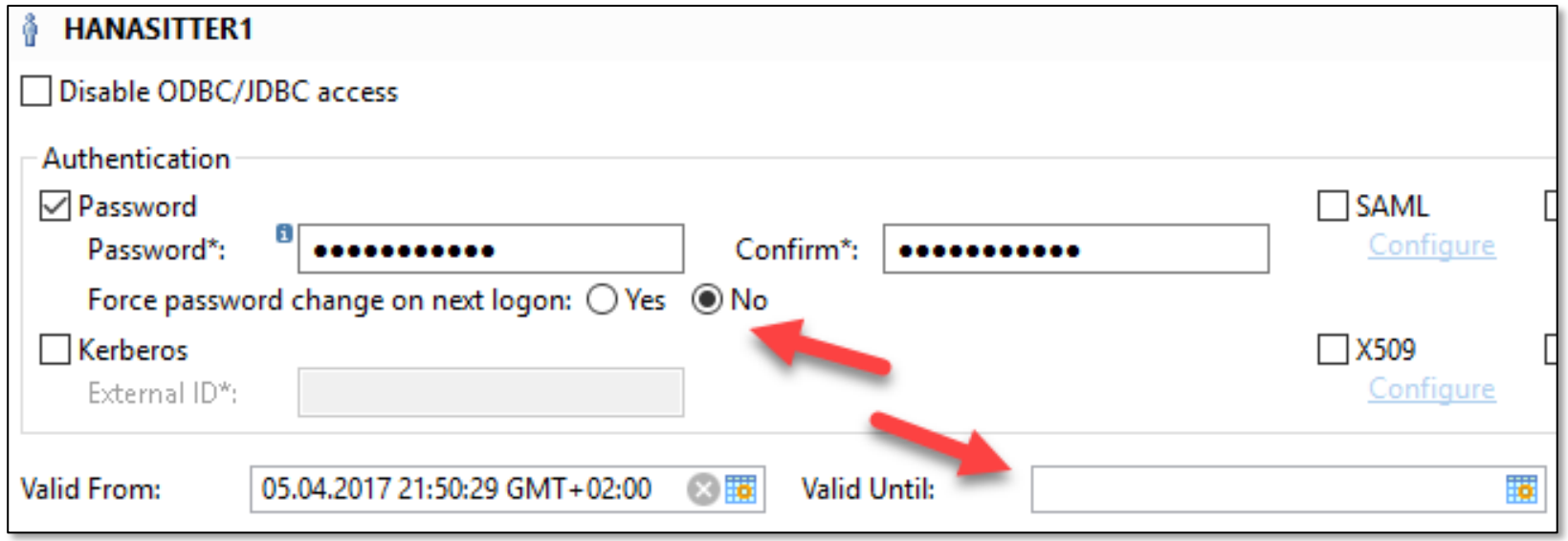
```
mo-fc8d991e0:/tmp/HANASitter> whoami
ch0adm
mo-fc8d991e0:/tmp/HANASitter> python hanasitter.py -k HANASITTER1KEY -nc 1
DB Address = , mo-fc8d991e0 , DB Instance = , 00
Online, Primary and Not-secondary Check: , Every 3600 seconds
Ping Check:    , Every 60 seconds, Ping Timeout = , 60 seconds
Thread Checks: , Every 60 seconds, Thread Checker Timeout = , 60 seconds
```

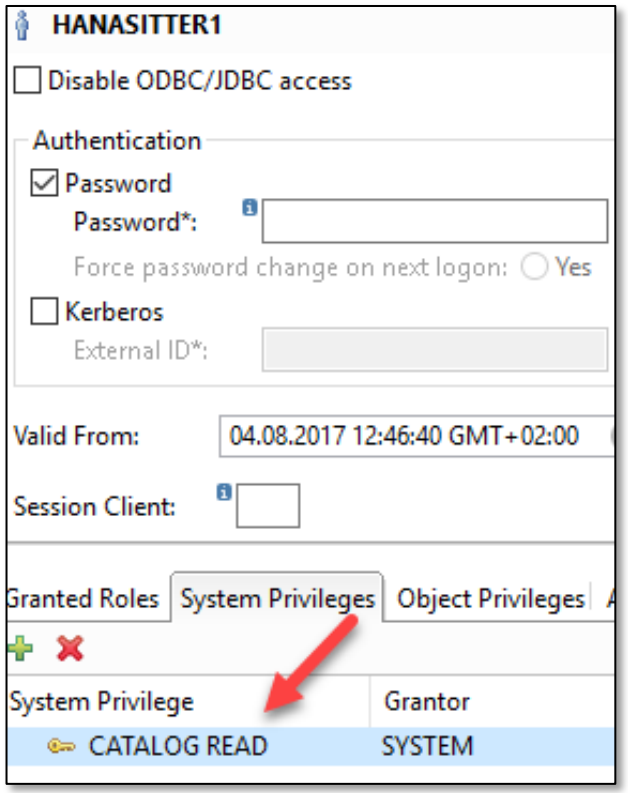
HANASitter – needs a user

The user that hanasitter uses to connect can be treated as a technical user

The user needs CATALOG READ and it must be a standard user

The user could be treated as a technical user, i.e. that it password should not expire





HANASitter – Online Check

First check tests if HANA is online, i.e. that

- **all services are running**
- **it is the primary instance (in case of a system replication setup)**
- **it is a worker node (in case of a scale-out scenario)**

If not online, it sleeps, by default, 1 hour and then tests if it is online again

The online test interval can be controlled by the `-oi` flag

Flag	Unit	Details	Explanation	Default
-oi	sec	online test interval	time it waits before it checks again if DB is online, primary (in SysRep scenario), and not-secondary (in ScaleOut scenario)	3600

Example:
The online check finds that this HANA is secondary, therefore HANASitter will not do anything for a while

```
ha1adm@dewdfglp00765:/tmp/HANASitter> python hanasitter.py -nc 1
Host = dewdfglp00765, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Action          , Timestamp           , Duration        , Successful    , Result      , Comment
Online Check    , 2017-06-09 09:53:43 ,     -           , True          , True        , Number running services: 7 out of 7
Primary Check   , 2017-06-09 09:53:46 ,     -           , True          , False       ,

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 3600 seconds break.
```

HANASitter – Tracking

If HANA is online the hanasitter starts “tracking” using three types of checks

1. **CPU Check**
2. **Ping Check**
3. **Critical Feature Checks**

If any of these checks finds a critical situation hanasitter starts to “record”, using four possible types of recording

1. **GStacks**
2. **Kernel Profiler Trace**
3. **Call Stacks**
4. **RTE Dumps**

If no recording was done, the tracking checks will restart after `-ci` seconds

If recording was done, hanasitter will exit (`-ar` < 0) or break `-ar` seconds before restarting the online check and tracking

Flag	Unit	Details	Explanation	Default
-ci	sec	check interval	time it waits before tracking checks restart (if no recording was done)	60
-ar	sec	after recording	time it waits before online check and tracking after recording	-1 (exit)

HANASitter – CPU Check

First tracking check (which is non-compulsory) tests if HANA is currently using too much CPU

The CPU check can consist of a number of CPU readings with a time interval between each readings; then the CPU check is done over a period

The CPU test can be controlled by the `-cpu` flag which has 4 items

Flag	Unit	Details	Explanation	Default
-cpu	, #, sec, %	CPU test	this flag should be followed by 4 items separated by only a comma; <1st item>,<2nd item>,<3rd item>,<4th item> • 1st item defines CPU type, 0=not used, 1=user cpu, 2=system cpu • 2nd item defines number of CPU readings • 3rd item defines the interval between the CPU readings [sec] • 4th item sets a limit of average used CPU for all readings [%]	0,0,0,100 (not used)

Example:

System CPU is checked with the average CPU over 5 readings with 5 seconds intervals with the limit 1 % The result turns out to be almost 4 %, so hanasitter starts to record

```
ha1adm@dewdfglp00766:/tmp/HANASitter> python hanasitter.py -cpu 2,5,5,1 -nc 1
Host = dewdfglp00766, DB Instance = 00, Single DB System
Action            , Timestamp           , Duration        , Successful   , Result     , Comment
Online Check      , 2017-06-09 10:11:42 ,     -           , True         , True       , Number running services: 7 out of 7
Primary Check     , 2017-06-09 10:11:44 ,     -           , True         , True       ,
Non-standby Check , 2017-06-09 10:11:44 ,     -           , True         , True       ,
System CPU Check  , 2017-06-09 10:12:09 , 0:00:25.009656  , True         , False      , Av. CPU = 3.67 % (Allowed = 1 %)
Call Stack Record , 2017-06-09 10:12:10 , 0:00:00.301395  ,   -          ,   -        , /tmp/hanasitter_output/callstack_2017
```

HANASitter – Ping Timeout Check

Second tracking check tries to connect to the database with a simple ping statement:
`select * from dummy`

If there is no response after `-pt` seconds, HANA is considered unresponsive, i.e. we have a "hanging" situation

Flag	Unit	Details	Explanation	Default
-pt	sec	ping timeout	time it waits before the DB is considered unresponsive during a ping test (`select * from dummy`)	60

Example:
Here the ping timeout was defined to only 1 second and there was no response from HANA within this time
HANA is considered unresponsive and recording starts

```
DEWDFGLP00765:/tmp/HANASitter> python hanasitter.py -pt 1 -nc 1
DB Address = , localhost , DB Instance = , 00
Ping Check         , 2017-04-10 01:04:12    , 0:00:01.000700   ,   -          , False       , No response from DB within 1 seconds.
Call Stack Record , 2017-04-10 01:04:13    , 0:00:01.281263   ,   -          ,   -         , /tmp/hanasitter_output/callstack_2017-0
```

HANASitter – Critical Feature Checks (1/6)

Third tracking check searches for, user defined, critical features - The flag `-cf` has two different modes:

1. One Column; a column in an `M_` view, a value and maximum number counts of that "feature", or
2. Where Clause; an `M_` view, a where clause and maximum number counts of that where clause

Flag	Unit	Details	Explanation	Default
-cf	-	list of critical features	a list, surrounded by ", of multiples of 4 items, separated by a comma only; "<1st item>, …,<4th item>, …,<1st item>, …,<4th item>" 1. One column mode: • 1st item defines a monitoring view, i.e. a SYS.M_* view • 2nd item defines a column in the view • 3rd item defines a possible value of column specified by 2nd item - use * before and/or after the value, to declare "wildcards", or - use ">" followed by an integer, to look for more repeats of that value than that integer specifies • 4th item sets a limit of number of counts allowed for that feature (default, 〈i〉, and <〈i〉: maximum number, >〈i〉: minimum number, where 〈i〉 is an integer) 2. Where clause mode: • 1st item defines a monitoring view, i.e. a SYS.M_* view • 2nd item is the keyword WHERE • 3rd item defines a complete sql where clause • 4th item sets a limit of number of counts allowed for that feature (default, 〈i〉, and <〈i〉: maximum number, >〈i〉: minimum number, where 〈i〉 is an integer)	'' (not used)
-tf	sec	feature check timeout	time it waits before the DB is considered unresponsive during a feature check (see above)	60
-lf	true/ false	log critical features	true → all info of the critical feature states defined by -cf will be logged, in the log directory in a criticalFeatures log file	false

HANASitter – Critical Feature Checks (2/6)

Example:
Here 2 critical feature checks are defined by only allowing

- 1 unload from table VARINUM
- 10 threads with the state IS_ACTIVE = TRUE

After the ping check the first feature check finds 0 unloads from table VARINUM, then the second feature check finds 11 threads that are active, this is more than allowed, so recording starts

```
mo-fc8d991e0:/tmp/HANASitter> python hanasitter.py -cf "M_CS_UNLOADS,TABLE_NAME,VARINUM,1,M_SERVICE_THREADS,IS_ACTIVE,TRUE,10" -nc 1
Host = mo-fc8d991e0, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features in the state, TABLE_NAME = VARINUM, in the view, M_CS_UNLOADS
Feature Check 2, allows maximum 10 features in the state, IS_ACTIVE = TRUE, in the view, M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 1                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-06-11 16:26:22   ,     -           , True         , True       , Number running services: 11 out of 11
Primary Check     , 2017-06-11 16:26:28   ,     -           , True         , True       ,
Non-standby Check , 2017-06-11 16:26:28   ,     -           , True         , True       ,
Ping Check        , 2017-06-11 16:26:28   , 0:00:00.164583  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-11 16:26:30   , 0:00:01.668655  , True         , True       , # Critical Features = 0 (allowed = 1), C
Feature Check 2   , 2017-06-11 16:26:30   , 0:00:00.264757  , True         , False       , # Critical Features = 11 (allowed = 10)
Call Stack Record , 2017-06-11 16:26:30   , 0:00:00.101899  ,   -          ,   -        , /tmp/hanasitter_output/callstack_2017-06
```

HANASitter – Critical Feature Checks (3/6)

Example:

Here 1 critical feature check is defined by only allowing 1 indexserver thread to be active

The feature check finds 3 indexserver threads that are active, this is more than allowed, so recording starts

```
mo-fc8d991e0:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,WHERE,IS_ACTIVE='TRUE' and SERVICE_NAME='indexserver',1" -nc 1
Host = mo-fc8d991e0, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features from the where clause = IS_ACTIVE='TRUE' and SERVICE_NAME='indexserver', in the view, M_SERVICE_THREADS
Recording Mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 1                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-06-11 16:32:51   ,     -           , True         , True       , Number running services: 11 out of 11
Primary Check     , 2017-06-11 16:32:58   ,     -           , True         , True       ,
Non-standby Check , 2017-06-11 16:32:58   ,     -           , True         , True       ,
Ping Check        , 2017-06-11 16:32:58   , 0:00:00.164220  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-11 16:32:58   , 0:00:00.164219  , True         , False      , # Critical Features = 3 (allowed = 1), Check: WHERE =
Call Stack Record , 2017-06-11 16:32:58   , 0:00:00.105039  ,   -          ,   -        , /tmp/hanasitter_output/callstack_2017-06-11_16:32:58.t
```

HANASitter – Critical Feature Checks (4/6)

Example:
Here 1 critical feature check is defined by requiring at least 10 indexserver threads to be active
The feature check finds 3 indexserver threads that are active, this is not enough, so recording starts

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,WHERE,IS_ACTIVE='TRUE' and SERVICE_NAME='indexserver',>10" -nc 1

Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 requires at least 10 times that IS_ACTIVE='TRUE' and SERVICE_NAME='indexserver' in M_SERVICE_THREADS
Recording mode: 1
After Recording: Exit
Action           , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check     , 2018-04-16 14:47:59   ,     -           , True         , True       , Number running services: 9 out of 9
Primary Check    , 2018-04-16 14:48:01   ,     -           , True         , True       ,
Ping Check       , 2018-04-16 14:48:01   , 0:00:00.164004  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1  , 2018-04-16 14:48:01   , 0:00:00.163813  , True         , False      , # Critical Features = 3 (minimum required = 10)
Call Stack Record , 2018-04-16 14:48:01  , 0:00:00.130066  ,   -          ,   -        , /tmp/hanasitter_output/callstack_ls80010_OQL_201
```

HANASitter – Critical Feature Checks (5/6)

Example:
Here 2 critical features are defined

- THREAD_STATE = Semaphore Wait in M_SERVICE_THREADS (Single Column Mode)
- IS_ACTIVE = 'TRUE' in M_SERVICE_THREADS (Where Clause Mode)

Since the log feature flag, -lf, is set to true, all features found with one of these states will be logged

```
mo-fc8d991e0:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Semaphore Wait,1,M_SERVICE_THREADS,WHERE,IS_ACTIVE = 'TRUE',2" -lf true
Host = mo-fc8d991e0, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features in the state, THREAD_STATE = Semaphore Wait, in the view, M_SERVICE_THREADS
Feature Check 2, allows maximum 2 features from the where clause = IS_ACTIVE = 'TRUE', in the view, M_SERVICE_THREADS
All information for all features that are in one of the above critical feature states is recorded in the /tmp/hanasitter_output/criticalFeatures log
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 0                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action              , Timestamp           , Duration        , Successful   , Result     , Comment
Online Check        , 2017-06-11 19:25:10 ,     -           , True         , True       , Number running services: 11 out of 11
Primary Check       , 2017-06-11 19:25:16 ,     -           , True         , True       ,
Non-standby Check   , 2017-06-11 19:25:16 ,     -           , True         , True       ,
Ping Check          , 2017-06-11 19:25:16 , 0:00:00.164571  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1     , 2017-06-11 19:25:17 , 0:00:00.264591  , True         , False       , # Critical Features = 4 (allowed = 1), Check: THREAD_STATE = Semap
```

NOTE: This log flag, -lf, could be very costly and is normally not to be used with any of the other recording types

HANASitter – Critical Feature Checks (6/6)

Example:
Here the critical feature is to find an active SQL statement that contains the string “invoice_ix_cs” for more than one time.

Once the long running SQL statement
`create column table invoice_ix_cs_copy like invoice_ix_cs with data;`
is executed, HANASitter finds it and executes the recording of one call stack

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -cf "M_ACTIVE_STATEMENTS,STATEMENT_STRING,invoice_ix_cs>1,0" -nc 1
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 allows only 0 times that column STATEMENT_STRING in M_ACTIVE_STATEMENTS contains the string invoice_ix_cs more than 1 times
Recording mode: 1
Recording Type      , Number Recordings    ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                    ,   60                  ,
Kernel Profiler     , 0                    ,   60                  ,   60                      ,    0
Call Stack          , 1                    ,   60                  ,
RTE Dumps (normal)  , 0                    ,   60                  ,
Recording Priority: RTE   Call Stacks   G-Stacks   Kernel Profiler
After Recording: Exit
Action              , Timestamp            , Duration        , Successful    , Result      , Comment
Online Check        , 2018-07-08 15:26:13  ,     -           , True          , True        , Number running services: 9 out of 9
Primary Check       , 2018-07-08 15:26:14  ,     -           , True          , True        ,
Ping Check          , 2018-07-08 15:26:14  , 0:00:00.164113  ,   -           , True        , DB responded faster than 60 seconds
Feature Check 1     , 2018-07-08 15:26:15  , 0:00:00.213936  , True          , True        , # Critical Features = 0 (maximum allowed = 0), C
heck if column STATEMENT_STRING in M_ACTIVE_STATEMENTS contains the string invoice_ix_cs more than 1 times
Ping Check          , 2018-07-08 15:27:15  , 0:00:00.163859  ,   -           , True        , DB responded faster than 60 seconds
Feature Check 1     , 2018-07-08 15:27:15  , 0:00:00.213954  , True          , False       , # Critical Features = 1 (maximum allowed = 0),
Check if column STATEMENT_STRING in M_ACTIVE_STATEMENTS contains the string invoice_ix_cs more than 1 times
Call Stack Record , 2018-07-08 15:27:15    , 0:00:00.141062  ,   -           ,   -         , /tmp/hanasitter_output/callstack_ls80010_OQL_201
8-07-08_15-27-15.txt
```

HANASitter – Critical Feature Iteration

HANASitter can do the critical feature checks multiple times and compare the average from the results to the threshold

Flag	Details	Explanation	Default
-if	number checks and intervals	<# checks 1>,<interval [s] 1>,…,<# checks N>,<interval [s] N>	

Example:

if, on average from 3 checks with 5s interval, > 30 THREAD_STATE=Running, or if any column from table VARINUM was unloaded → record

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Running,30,M_CS_UNLOADS,TABLE_NAME,VARINUM,1" -if 3,5,1,0 -nc 2
Will make a CF with  M_CS_UNLOADS TABLE_NAME VARINUM 1
Host = ls80010, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 allows only 30 times that THREAD_STATE = 'Running' in M_SERVICE_THREADS as an average from 3 checks with 5 seconds intervals
Feature Check 2 allows only 1 times that TABLE_NAME = 'VARINUM' in M_CS_UNLOADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                     ,    0
Call Stack          , 2                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action           , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check     , 2017-07-14 17:58:11   ,     -           , True         , True       , Number running services: 9 out of 9
Primary Check    , 2017-07-14 17:58:12   ,     -           , True         , True       ,
Non-standby Check , 2017-07-14 17:58:12  ,     -           , True         , True       ,
Ping Check       , 2017-07-14 17:58:12   , 0:00:00.163895  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1  , 2017-07-14 17:58:28   , 0:00:15.387558  , True         , True       , # Critical Features = 8 (allowed = 30), Check: THREAD_STATE =
Feature Check 2  , 2017-07-14 17:58:34   , 0:00:06.923696  , True         , False       , # Critical Features = 4 (allowed = 1), Check: TABLE_NAME = '
Call Stack Record , 2017-07-14 17:58:35  , 0:00:00.145709  ,   -          ,   -        , /tmp/hanasitter_output/callstack_ls80010_OQL_2017-07-14_17:58
```

HANASitter – Recording Mode (1/2)

HANASitter can record with the following recording types

1. **GStacks**
2. **Kernel Profiler Trace**
3. **Call Stacks**
4. **RTE Dumps**

If hanasitter is supposed to record using more than one of the recording types then there are 3 different "recording modes", defined with `-rm`

Flag	Unit	Details	Explanation	Default
-rm	-	recording mode	1 = each requested recording types are done one after each other, e.g. GStack1, GStack2, …, GStackN, RTE1, …, RTEN 2 = recordings are done after each other, e.g. GStack1, RTE1, GStack2, RTE2, … 3 = different recording types are recorded in parallel threads, e.g. if 2 GStacks and 1 RTE are requested then GStack1 and RTE1 are first done in parallel, when both are done GStack2 starts	1

HANASitter – Recording Mode (2/2)

Example:

Here hanasitter is requested to find the situation that more then 5 threads in the state IS_ACTIVE = TRUE
When this situation is found hanasitter records using 3 Call Stacks, 2 RTE Dumps, and 1 GStack
Since Recording Mode 3 is requested they are recorded in parallel in the following order:

1. RTE Dump1, Call Stack 1, and GStack 1
2. Call Stack 2, and RTE Dump 2
3. Call Stack 3

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,5" -nc 3 -nr 2 -ng 1 -rm 3
Host = ls80010, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 allows only 5 times that IS_ACTIVE = 'TRUE' in M_SERVICE_THREADS
Recording mode: 3
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 1                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 3                   ,   60                  ,
RTE Dumps           , 2                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-10-01 17:18:45   ,     -           , True         , True       , Number running services: 9
Primary Check     , 2017-10-01 17:18:46   ,     -           , True         , True       ,
Ping Check        , 2017-10-01 17:18:46   , 0:00:00.164017  ,   -          , True       , DB responded faster than 60
Feature Check 1   , 2017-10-01 17:18:47   , 0:00:00.214006  , True         , False      , # Critical Features = 13 (
RTE Dump Record   , 2017-10-01 17:21:25   , 0:02:37.905730  , True         ,   -        , /tmp/hanasitter_output/rted
Call Stack Record , 2017-10-01 17:21:24   , 0:02:37.756701  ,   -          ,   -        , /tmp/hanasitter_output/call
GStack Record     , 2017-10-01 17:21:25   , 0:02:37.926312  ,   -          ,   -        , /tmp/hanasitter_output/gsta
RTE Dump Record   , 2017-10-01 17:22:26   , 0:00:00.992576  , True         ,   -        , /tmp/hanasitter_output/rted
Call Stack Record , 2017-10-01 17:22:25   , 0:00:00.225087  ,   -          ,   -        , /tmp/hanasitter_output/call
Call Stack Record , 2017-10-01 17:23:26   , 0:00:00.146175  ,   -          ,   -        , /tmp/hanasitter_output/call
```

HANASitter – Recording Priority

With the -rp flag one can define the order of the recording types

Flag	Unit	Details	Explanation	Default
-rp	List with 4 integers between 1 and 4	recording priority	This list of 4 integers defines the order of the recording types 1 = RTE, 2 = Call Stacks, 3 = G-Stacks, 4 = Kernel Profiler	1,2,3,4

Example:
Here the recording order is requested to be G-Stacks, Call Stacks, Kernel Profiler and then RTE dumps. Since 1 RTE dump, 2 Call Stacks, and 1 G-Stack was required with recording mode 2, first the G-Stack, then a Call Stack and then the RTE dump is recorded before the last Call Stack

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Running,1" -nr 1 -nc 2 -ng 1 -rm 2 -rp 3,2,4,1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 1                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                     ,    0
Call Stack          , 2                   ,   60                  ,
RTE Dumps           , 1                   ,   60                  ,
Recording Priority: G-Stacks   Call Stacks   Kernel Profiler   RTE
After Recording: Exit
Action              , Timestamp           , Duration            , Successful    , Result     , Comment
Online Check        , 2017-11-15 14:07:13 ,     -               , True          , True       , Number running services: 9 out of 9
Primary Check       , 2017-11-15 14:07:14 ,     -               , True          , True       ,
Ping Check          , 2017-11-15 14:07:14 , 0:00:00.164236      ,   -           , True       , DB responded faster than 60 seconds
Feature Check 1     , 2017-11-15 14:07:14 , 0:00:00.214050      , True          , False      , # Critical Features = 8 (allowed = 1), Chec
GStack Record       , 2017-11-15 14:10:14 , 0:03:00.055366      ,   -           ,   -        , /tmp/hanasitter_output/gstack_11417_2017-11-
Call Stack Record   , 2017-11-15 14:11:15 , 0:00:00.153899      ,   -           ,   -        , /tmp/hanasitter_output/callstack_ls80010_OQL
RTE Dump Record     , 2017-11-15 14:12:16 , 0:00:01.354217      , True          ,   -        , /tmp/hanasitter_output/rtedump_ls80010_OQL_2
Call Stack Record   , 2017-11-15 14:13:16 , 0:00:00.116943      ,   -           ,   -        , /tmp/hanasitter_output/callstack_ls80010_OQL
```

HANASitter – GStack

One of the possible recording options is to do gstack of the indexserver, i.e. an execution stack trace of the indexserver from OS-level

This recording option is controlled by the `-ng` and `-ig` flags

Flag	Unit	Details	Explanation	Default
-ng	-	number gstacks	Number indexserver gstacks created if the DB is considered unresponsive	0
-ig	sec	gstacks interval	-rm = 1: time it waits between each gstack -rm = 2: time it waits after a gstack -rm = 3: time the thread waits after a gstack	60

Note: This recording option will only be done on current host

Example:
Here 2 GStacks with 30 seconds delay are requested when there is more than 1 active thread
Hanasitter slept for 30 seconds after the 1st GStack finished, at 13:35:35, and the 2nd GStack started to record at 13:36:05

```
mo-fc8d991e0:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,1" -ng 2 -ig 30
Host = mo-fc8d991e0, DB Instance = 00, Single DB System
Ping Check        , 2017-06-16 13:33:41    , 0:00:00.214811   ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-16 13:33:41    , 0:00:00.214887   , True         , False      , # Critical Features = 10 (allowed = 1), Check: IS_ACTIVE = TRUE
GStack Record     , 2017-06-16 13:35:35    , 0:01:54.029790   ,   -          ,   -        , /tmp/hanasitter_output/gstack_6090_2017-06-16_13:33:41.txt
GStack Record     , 2017-06-16 13:37:56    , 0:01:51.052227   ,   -          ,   -        , /tmp/hanasitter_output/gstack_6090_2017-06-16_13:36:05.txt
```

HANASitter – Kernel Profiler

Another possible recording option is to do Kernel Profiler traces of the indexserver – mainly for performance analysis, this is controlled by the `-np`, `-dp`, `-wp`, and `-ip` flags

Flag	Unit	Details	Explanation	Default
-np	-	number kernel profiler traces	Number indexserver kernel profiler traces created if the DB is considered unresponsive	0
-dp	sec	profiler duration	How long time it is tracing	60
-wp	millise conds	profiler wait time	wait time after callstacks of all running threads have been taken	0
-ip	sec	kernel interval	-rm = 1: time it waits between each profiler trace -rm = 2: time it waits after a profiler trace -rm = 3: time the thread waits after a profiler trace	60

Example:

Here 2 Kernel Profiler traces with a duration of 30 seconds and a delay of 30 seconds are recorded:

```
mo-fc8d991e0:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,1" -np 2 -dp 30 -ig 30
Host = mo-fc8d991e0, DB Instance = 00, Single DB System
Feature Check 1    , 2017-06-16 13:23:19    , 0:00:00.214674   , True         , False       , # Critical Features = 10 (allowed = 1), Check: IS_ACTIVE = TRUE, in
 view M_SERVICE_THREADS
Kernel Profiler    , 2017-06-16 13:23:50    , 0:00:30.684400   ,   -          ,   -         , /tmp/hanasitter_output/kernel_profiler_cpu_2017-06-16_13:23:19.dot a
nd /tmp/hanasitter_output/kernel_profiler_wait_2017-06-16_13:23:19.dot
Kernel Profiler    , 2017-06-16 13:25:21    , 0:00:30.648090   ,   -          ,   -         , /tmp/hanasitter_output/kernel_profiler_cpu_2017-06-16_13:24:50.dot a
nd /tmp/hanasitter_output/kernel_profiler_wait_2017-06-16_13:24:50.dot
```

HANASitter – Kernel Profiler at Scale Out

Kernel Profiler traces are done for each host in a scale out scenario
(due to limitations of hdbcons the kernel profiler traces will not have be separated in cpu and wait files)

This is controlled by the `-np`, `-dp`, `-wp`, and `-ip` flags

```
hsiadm@dewdfglp00835:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,1" -np 1
Host = dewdfglp00835, DB Instance = 00, Scale Out DB System with hosts: dewdfglp00835, dewdfglp00837, dewdfglp00836
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features in the state, IS_ACTIVE = TRUE, in the view, M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 1                   ,   60                  ,   60                      ,    0
Call Stack          , 0                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-06-16 15:10:09   ,     -           , True         , True       , Number running services: 5 out of 5
Primary Check     , 2017-06-16 15:10:12   ,     -           , True         , True       ,
Non-standby Check , 2017-06-16 15:10:12   ,     -           , True         , True       ,
Ping Check        , 2017-06-16 15:10:12   , 0:00:00.164554  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-16 15:10:12   , 0:00:00.264813  , True         , False      , # Critical Features = 30 (allowed = 1), Check: IS_ACTIVE = TRUE, in view M_SERVICE_THREADS
Kernel Profiler   , 2017-06-16 15:11:14   , 0:01:01.595220  ,   -          ,   -        , /tmp/hanasitter_output/kernel_profiler_cpu_wait_dewdfglp00835_HSI_2017-06-16_15:10:12.dot
Kernel Profiler   , 2017-06-16 15:12:15   , 0:01:00.851089  ,   -          ,   -        , /tmp/hanasitter_output/kernel_profiler_cpu_wait_dewdfglp00837_HSI_2017-06-16_15:11:14.dot
Kernel Profiler   , 2017-06-16 15:13:17   , 0:01:01.835549  ,   -          ,   -        , /tmp/hanasitter_output/kernel_profiler_cpu_wait_dewdfglp00836_HSI_2017-06-16_15:12:15.dot
```

HANASitter – Call Stacks

Another recording option is to do Call Stacks

This is controlled by the `-nc`, and `-ic` flags

Flag	Unit	Details	Explanation	Default
-nc	-	number call stacks	Number call stacks created if the DB is considered unresponsive	0
-ic	sec	call stacks interval	-rm = 1: time it waits between each call stack -rm = 2: time it waits after a call stack -rm = 3: time the thread waits after a call stack	60

Example:
Here, when there are more then 5 threads with the state IS_ACTIVE=TRUE, 2 call stacks, with 30 seconds between them, are recorded:

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py  -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,5" -nc 2 -ic 30
Host = ls80010, DB Instance = 00, Single DB System
Ping Check         , 2017-10-01 17:33:24    , 0:00:00.164094   ,   -          , True       , DB responded fas
Feature Check 1    , 2017-10-01 17:33:24    , 0:00:00.213978   , True         , False      , # Critical Feat
Call Stack Record  , 2017-10-01 17:33:24    , 0:00:00.127802   ,   -          ,   -        , /tmp/hanasitter_
Call Stack Record  , 2017-10-01 17:33:54    , 0:00:00.122938   ,   -          ,   -        , /tmp/hanasitter
```

Monitoring

HANASitter – Call Stacks at Scale Out

Call Stacks will be done for all hosts in a scale out automatically

```
hsiadm@dewdfglp00835:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,1" -nc 1
Host = dewdfglp00835, DB Instance = 00, Scale Out DB System with hosts: dewdfglp00835, dewdfglp00837, dewdfglp00836
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features in the state, IS_ACTIVE = TRUE, in the view, M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 1                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration         , Successful   , Result     , Comment
Online Check      , 2017-06-16 12:55:43   ,     -            , True         , True       , Number running services: 5 out of 5
Primary Check     , 2017-06-16 12:55:46   ,     -            , True         , True       ,
Non-standby Check , 2017-06-16 12:55:46   ,     -            , True         , True       ,
Ping Check        , 2017-06-16 12:55:46   , 0:00:00.164382   ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-16 12:55:46   , 0:00:00.314905   , True         , False      , # Critical Features = 28 (allowed = 1), Check: IS_ACTIVE = TRUE, in view M_SERVICE_THREADS
Call Stack Record , 2017-06-16 12:55:47   , 0:00:00.232236   ,   -          ,   -        , /tmp/hanasitter_output/callstack_dewdfglp00835_HSI_2017-06-16_12:55:46.txt
Call Stack Record , 2017-06-16 12:55:47   , 0:00:00.269409   ,   -          ,   -        , /tmp/hanasitter_output/callstack_dewdfglp00837_HSI_2017-06-16_12:55:47.txt
Call Stack Record , 2017-06-16 12:55:47   , 0:00:00.248774   ,   -          ,   -        , /tmp/hanasitter_output/callstack_dewdfglp00836_HSI_2017-06-16_12:55:47.txt
```

HANASitter – RTE Dumps

Another recording option is to do RunTime Environment Dumps (RTE Dumps)

This is controlled by the `-nr`, `-ir`, and `-mr` flags

Flag	Unit	Details	Explanation	Default
-nr	-	number RTE dumps	Number RTE Dumps created if the DB is considered unresponsive	0
-ir	sec	RTE dumps interval	-rm = 1: time it waits between each RTE dump -rm = 2: time it waits after an RTE dump -rm = 3: time the thread waits after an RTE dump	60
-mr	0,1	RTE	-mr = 0: normal RTE dump -mr = 1: light RTE dump mode, only RTE dump with STACK_SHORT and THREADS sections, and the views M_JOBEXECUTORS_, M_DEV_JOBEX_THREADGROUPS, M_DEV_JOBEXWAITING, M_DEV_CONTEXTS, M_CONNECTIONS, M_DEV_SESSION_PARTITIONS	0

Example:
Here, when there are more then 5 threads with the state IS_ACTIVE=TRUE, 2 RTE dumps, with 30 seconds between them, are recorded:

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py  -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,5" -nr 2 -ir 30
Host = ls80010, DB Instance = 00, Single DB System
Ping Check        , 2017-10-01 17:38:00    , 0:00:00.163893    ,   -          , True       , DB responded fa
Feature Check 1   , 2017-10-01 17:38:00    , 0:00:00.213988    , True         , False      , # Critical Fea
RTE Dump Record   , 2017-10-01 17:38:01    , 0:00:00.897289    , True         ,   -        , /tmp/hanasitter
RTE Dump Record   , 2017-10-01 17:38:32    , 0:00:00.898280    , True         ,   -        , /tmp/hanasitter
```

Monitoring

HANASitter – Light RTE Dumps

The light RTE dump mode is here used to make 3 small RTE dumps with an interval of 10 seconds:

```
ha2adm@atgvmls7050:/tmp/HANASitter> python hanasitter.py -cf M_SERVICE_THREADS,THREAD_STATE,Running,3 -nr 3 -ir 10 -mr 1 -k SYSTEMDBKEY

HANASitter executed 2018-05-03 15:45:52 with
hanasitter.py -cf M_SERVICE_THREADS,THREAD_STATE,Running,3 -nr 3 -ir 10 -mr 1 -k SYSTEMDBKEY
as SYSTEMDBKEY: KEY SYSTEMDBKEY
  ENV : atgvmls7050:30013
  USER: system
  DATABASE: systemdb

Host = atgvmls7050, SID = HA2, DB Instance = 00, MDC SystemDB, Nameserver Port = 30001
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 allows only 3 times that THREAD_STATE = 'Running' in M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 0                   ,   60                  ,
RTE Dumps (light)   , 3                   ,   10                  ,
Recording Priority: RTE   Call Stacks   G-Stacks   Kernel Profiler
After Recording: Exit
Action              , Timestamp             , Duration         , Successful   , Result      , Comment
Online Check        , 2018-05-03 15:45:52   ,     -            , True         , True        , Number running services: 11 out of 11
Primary Check       , 2018-05-03 15:45:54   ,     -            , True         , True        ,
Ping Check          , 2018-05-03 15:45:54   , 0:00:00.164322   ,   -          , True        , DB responded faster than 60 seconds
Feature Check 1     , 2018-05-03 15:45:54   , 0:00:00.265588   , True         , False       , # Critical Features = 4 (maximum allowed = 3), Check: THREAD_STATE = 'Running'
RTE Dump Record     , 2018-05-03 15:45:55   , 0:00:00.800020   , True         ,   -         , /tmp/hanasitter_output/rtedump_light_atgvmls7050_HA2_2018-05-03_15-45-54.trc
RTE Dump Record     , 2018-05-03 15:46:06   , 0:00:00.666470   , True         ,   -         , /tmp/hanasitter_output/rtedump_light_atgvmls7050_HA2_2018-05-03_15-46-05.trc
RTE Dump Record     , 2018-05-03 15:46:16   , 0:00:00.679907   , True         ,   -         , /tmp/hanasitter_output/rtedump_light_atgvmls7050_HA2_2018-05-03_15-46-16.trc
```

The light RTE dump mode, -mr = 1, might be useful to continuously create small dumps during certain situations…

Monitoring

HANASitter – RTE Dumps at Scale Out

RTE Dumps will automatically be done for all hosts in a scale out scenario

```
hsiadm@dewdfglp00835:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Running,1" -nr 1
Host = dewdfglp00835, DB Instance = 00, Scale Out DB System with hosts: dewdfglp00835, dewdfglp00836, dewdfglp00837
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 allows only 1 times that THREAD_STATE = 'Running' in M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 0                   ,   60                  ,
RTE Dumps           , 1                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-09-07 15:45:22   ,     -           , True         , True       , Number running services: 7 out of 7
Primary Check     , 2017-09-07 15:45:26   ,     -           , True         , True       ,
Non-standby Check , 2017-09-07 15:45:26   ,     -           , True         , True       ,
Ping Check        , 2017-09-07 15:45:26   , 0:00:00.164829  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-09-07 15:45:26   , 0:00:00.316117  , True         , False       , # Critical Features = 14 (allowed = 1), Chec
E_THREADS
RTE Dump Record   , 2017-09-07 15:45:27   , 0:00:00.802223  , True         ,   -        , /tmp/hanasitter_output/rtedump_dewdfglp00835_
RTE Dump Record   , 2017-09-07 15:45:28   , 0:00:00.991286  , True         ,   -        , /tmp/hanasitter_output/rtedump_dewdfglp00836_
RTE Dump Record   , 2017-09-07 15:45:28   , 0:00:00.354565  , True         ,   -        , /tmp/hanasitter_output/rtedump_dewdfglp00837_
```

HANASitter – Customized SQL Dumps

Another recording option is to dump the output of any self defined SELECT SQL statement

This is controlled by the `-ns`, `-is`, and `-cs` flags

Flag	Unit	Details	Explanation	Default
-ns	-	number customized SQL dumps	Number customized SQL dumps created if the DB is considered unresponsive	0
-ir	sec	customized SQL dumps interval	-rm = 1: time it waits between each customized SQL dump -rm = 2: time it waits after an customized SQL dump -rm = 3: time the thread waits after a customized SQL dump	60
-cs	-	customer SQL	provide the SELECT statement that defines the output	'' (not used)

Example:
Here, when there are more than 5 active threads, first 1 dump from the view M_DEV_TRANSACTIONS_HISTORY_, then 1 Kernel Profiler dump, then 1 RTE dump and then 1 Call Stack, is recorded

```
xscadm@atgvmls866:/tmp/HANASitter> python hanasitter.py -k T1KEY -pt 10
-cf "M_SERVICE_THREADS,WHERE,IS_ACTIVE='TRUE',5" -ns 1 -cs "SELECT CONNE
CTION_ID, PRIMARY_CONNECTION_ID, TRANSACTION_ID from SYS.M_DEV_TRANSACTI
ONS_HISTORY_ WHERE START_TIME >= ADD_SECONDS (CURRENT_TIMESTAMP, -100)"
-nr 1 -np 1 -nc 1 -rp 5,4,1,2,3
```

HANASitter – Kill Sessions (1/2)

Instead (or additionally) to recording if a critical feature is found, HANASitter can also try to kill the session of that critical feature

Flag	Unit	Details	Explanation	Default
-ks	list of true/false	kill session	list of booleans (length must be the same as number of features defined by -cf) that defines if -cf's features could indicate that the sessions (connections) are tried to be disconnected or not	[] (not used)

HANASitter – Kill Sessions (2/2)

Example:
If there is an active statement with the string “invoice_ix_cs” repeated for more than 1 time, and if that statement is active on average based on 3 checks with 5s interval, then the connection (session) that runs that statement is disconnected:

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -cf "M_ACTIVE_STATEMENTS,STATEMENT_STRING,invoice_ix_cs>1,0" -ks true -if 3,5

Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1 allows only 0 times that column STATEMENT_STRING in M_ACTIVE_STATEMENTS contains the string invoice_ix_cs more than
 1 times as an average from 3 checks with 5 seconds intervals
Recording mode: 1
Recording Type      , Number Recordings    ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                    ,   60                  ,
Kernel Profiler     , 0                    ,   60                  ,   60                      ,    0
Call Stack          , 0                    ,   60                  ,
RTE Dumps (normal)  , 0                    ,   60                  ,
Recording Priority: RTE   Call Stacks   G-Stacks   Kernel Profiler
After Recording: Exit
Action              , Timestamp            , Duration        , Successful    , Result      , Comment
Online Check        , 2018-07-09 00:07:26  ,     -           , True          , True        , Number running services: 9 out of 9
Primary Check       , 2018-07-09 00:07:28  ,     -           , True          , True        ,
Ping Check          , 2018-07-09 00:07:28  , 0:00:00.164119  ,   -           , True        , DB responded faster than 60 seconds
Feature Check 1     , 2018-07-09 00:07:43  , 0:00:15.298323  , True          , False        , # Critical Features = 1 (maximum allow
ed = 0), Check if column STATEMENT_STRING in M_ACTIVE_STATEMENTS contains the string invoice_ix_cs more than 1 times
Will disconnect session 233821 due to the check: column STATEMENT_STRING in M_ACTIVE_STATEMENTS contains the string invoice_ix_cs m
ore than 1 times
oqladm@ls80010:/tmp/HANASitter>
```

```
create column table invoice_ix_cs_copy like invoice_ix_cs with data;

Could not execute 'create column table invoice_ix_cs_copy like invoice_ix_cs with data' in 51.033 seconds
Data receive failed [Connection reset].
```

HANASitter – Key With All Hosts

It is possible to provide a user key in hdbuserstore that contains all hosts:

```
KEY HANASITTERKEY
  ENV : dewdfglp00835:30015,dewdfglp00836:30015,dewdfglp00837:30015
  USER: HANASITTER1
```

HANASitter will then automatically use the local host:

```
hsiadm@dewdfglp00836:/tmp/HANASitter> python hanasitter.py -k HANASITTERKEY -cf "M_SERVICE_TH
Host = dewdfglp00836, DB Instance = 00, Scale Out DB System with hosts: dewdfglp00835, dewdfg
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
```

HANASitter – Host Mode Recording (1/2)

If HANASitter runs in "host mode" it will only record on those hosts where the critical feature happen

Flag	Unit	Details	Explanation	Default
-hm	true/false	host mode	if true then all critical features are considered per host and the recording is done only for those hosts the critical feature is above allowed limit per host, default: false Note: -hm is not supported for gstack (-ng)	false

Example:
Here, 2 calls stacks and 1 RTE dump are recorded only for those hosts, or that host, for which there are more then 5 threads running (on average of 3 checks, with 5 seconds intervals):

```
hsiadm@dewdfglp00835:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Running,5" -if 3,5 -nc 2 -nr 1 -hm true
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp            , Duration        , Successful   , Result     , Comment
Online Check      , 2018-11-09 12:53:56  ,     -           , True         , True       , Number running services: 7 out of 7
Primary Check     , 2018-11-09 12:53:59  ,     -           , True         , True       ,
Ping Check        , 2018-11-09 12:53:59  , 0:00:00.164877   ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2018-11-09 12:54:15  , 0:00:15.902442   , True         , True       , # CFs = 4 for dewdfglp00837, max allowed = 5, check: THREAD_STATE = 'Running'
Feature Check 1   , 2018-11-09 12:54:15  , 0:00:15.902442   , True         , True       , # CFs = 4 for dewdfglp00836, max allowed = 5, check: THREAD_STATE = 'Running'
Feature Check 1   , 2018-11-09 12:54:15  , 0:00:15.902442   , True         , False      , # CFs = 6 for dewdfglp00835, max allowed = 5, check: THREAD_STATE = 'Running'
RTE Dump Record   , 2018-11-09 12:54:21  , 0:00:06.071313   , True         ,   -        , /tmp/hanasitter_output/rtedump_normal_dewdfglp00835_HSI_2018-11-09_12-54-15.trc
Call Stack Record , 2018-11-09 12:55:21  , 0:00:00.239098   ,   -          ,   -        , /tmp/hanasitter_output/callstack_dewdfglp00835_HSI_2018-11-09_12-55-21.txt
Call Stack Record , 2018-11-09 12:56:24  , 0:00:02.612926   ,   -          ,   -        , /tmp/hanasitter_output/callstack_dewdfglp00835_HSI_2018-11-09_12-56-21.txt
hsiadm@dewdfglp00835:/tmp/HANASitter>
```

We see that only host dewdfglp00835 crossing this allowed limit, so only host dewdfglp00835 is recording 2 callstacks and 1 RTE dump

Monitoring

HANASitter – Host Mode Recording (2/2)

Another Example:
Here, 1 kernel profile dump is recorded only for those hosts, or that host, for which there are less than 6 threads running (on average of 3 checks, with 5 seconds intervals):

```
hsiadm@dewdfglp00835:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Running,>6" -if 3,5 -np 1 -hm true
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp             , Duration        , Successful    , Result      , Comment
Online Check      , 2018-11-11 16:28:17   ,     -           , True          , True        , Number running services: 7 out of 7
Primary Check     , 2018-11-11 16:28:20   ,     -           , True          , True        ,
Ping Check        , 2018-11-11 16:28:20   , 0:00:00.164794  ,   -           , True        , DB responded faster than 60 seconds
Feature Check 1   , 2018-11-11 16:28:36   , 0:00:15.863664  , True          , False       , # CFs = 4 for dewdfglp00837, min required = 6, check: THREAD_STATE = 'Running'
Feature Check 1   , 2018-11-11 16:28:36   , 0:00:15.863664  , True          , False       , # CFs = 4 for dewdfglp00836, min required = 6, check: THREAD_STATE = 'Running'
Feature Check 1   , 2018-11-11 16:28:36   , 0:00:15.863664  , True          , True        , # CFs = 6 for dewdfglp00835, min required = 6, check: THREAD_STATE = 'Running'
Kernel Profiler   , 2018-11-11 16:29:38   , 0:01:01.599605  , True          ,   -         , /tmp/hanasitter_output/kernel_profiler_cpu_dewdfglp00836_HSI_2018-11-11_16-28-36.dot
and /tmp/hanasitter_output/kernel_profiler_wait_dewdfglp00836_HSI_2018-11-11_16-28-36.dot
Kernel Profiler   , 2018-11-11 16:30:38   , 0:01:00.962716  , True          ,   -         , /tmp/hanasitter_output/kernel_profiler_cpu_dewdfglp00837_HSI_2018-11-11_16-29-38.dot
and /tmp/hanasitter_output/kernel_profiler_wait_dewdfglp00837_HSI_2018-11-11_16-29-38.dot
hsiadm@dewdfglp00835:/tmp/HANASitter>
```

We see that the hosts dewdfglp00836 and dewdfglp00837 are not reaching this required limit, so the hosts dewdfglp00836 and dewdfglp00837 record 1 kernel profiler dump each

HANASitter – and MDC (1/2)

HANASitter detects an MDC scenario and if it is the System or a Tenant

Example:
Here the key of a DB user in the System DB in an MDC setup is used to run hanasitter:

```
ls2999:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,1" -nr 1 -k SYSKEY
Host = ls2999, DB Instance = 01, MDC system
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features in the state, IS_ACTIVE = TRUE, in the view, M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                       ,    0
Call Stack          , 0                   ,   60                  ,
RTE Dumps           , 1                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-06-16 16:37:52   ,     -           , True         , True       , Number running services: 7 out of 7
Primary Check     , 2017-06-16 16:37:55   ,     -           , True         , True       ,
Non-standby Check , 2017-06-16 16:37:55   ,     -           , True         , True       ,
Ping Check        , 2017-06-16 16:37:55   , 0:00:00.164097  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-16 16:37:55   , 0:00:00.163950  , True         , False       , # Critical Features = 3 (allowed = 1), Check:
RTE Dump Record   , 2017-06-16 16:37:56   , 0:00:00.974544  , True         ,   -        , /tmp/hanasitter_output/rtedump_ls2999_H00_2017
ls2999:/tmp/HANASitter>
```

HANASitter – and MDC (2/2)

HANASitter detects an MDC scenario and if it is the System or a Tenant

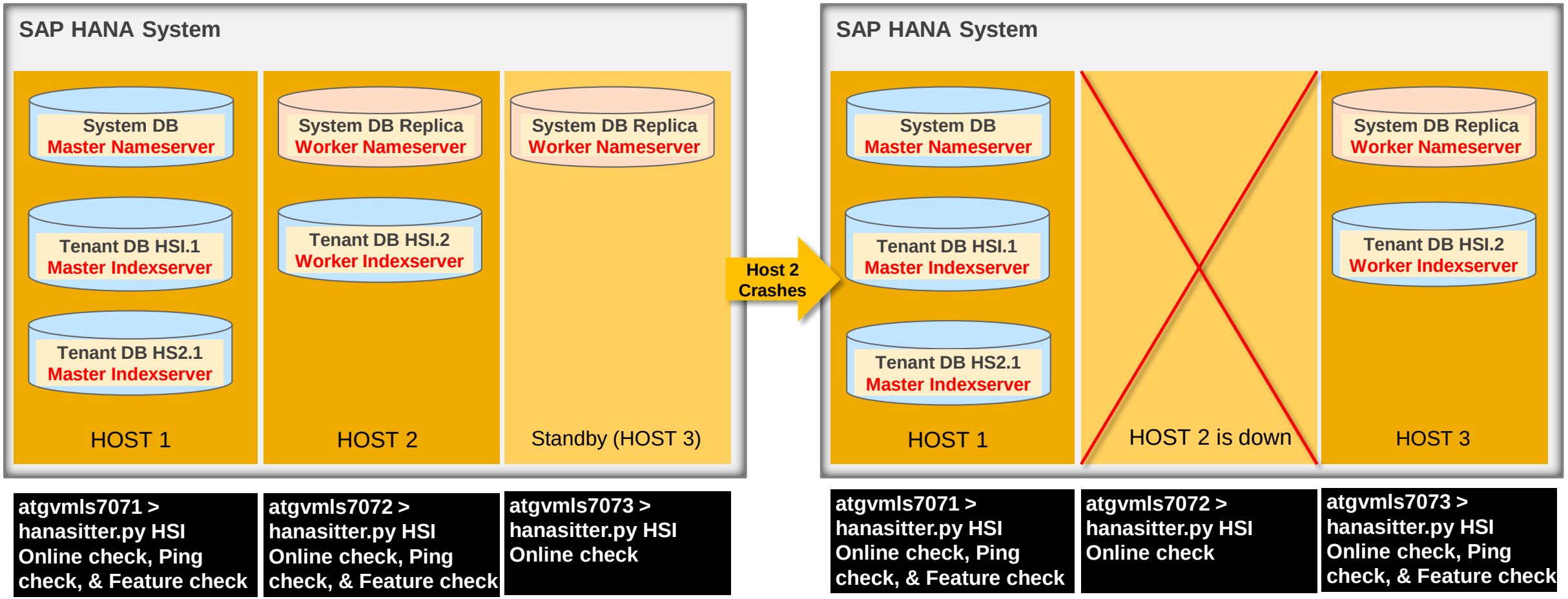
Example:
Here the key of a DB user in one of the Tenant DBs in an MDC setup is used to run hanasitter:

```
ls2999:/tmp/HANASitter> python hanasitter.py -cf "M_SERVICE_THREADS,IS_ACTIVE,TRUE,1" -nc 1 -nr 1 -k TEN1KEY
Host = ls2999, DB Instance = 01, MDC tenant = H01, Indexserver Port = 30140
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 1 features in the state, IS_ACTIVE = TRUE, in the view, M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings    ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                    ,   60                  ,
Kernel Profiler     , 0                    ,   60                  ,   60                      ,    0
Call Stack          , 1                    ,   60                  ,
RTE Dumps           , 1                    ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2017-06-16 16:48:11   ,     -           , True         , True       , Number running services: 7 out of 7
Primary Check     , 2017-06-16 16:48:14   ,     -           , True         , True       ,
Non-standby Check , 2017-06-16 16:48:14   ,     -           , True         , True       ,
Ping Check        , 2017-06-16 16:48:14   , 0:00:00.164167  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-16 16:48:14   , 0:00:00.163977  , True         , False       , # Critical Features = 3 (allowed = 1), Check: IS_ACTI
Call Stack Record , 2017-06-16 16:48:14   , 0:00:00.154666  ,   -          ,   -        , /tmp/hanasitter_output/callstack_ls2999_H01_2017-06-16
RTE Dump Record   , 2017-06-16 16:49:15   , 0:00:00.902135  , True         ,   -        , /tmp/hanasitter_output/rtedump_ls2999_H01_2017-06-16_1
ls2999:/tmp/HANASitter>
```

HANASitter – Scale Out Example (1/9)

Scenario for this example:

The HANASitter monitoring of HSI will automatically follow the indexserver after a fail-over:



HANASitter – Scale Out Example (2/9)

On the first host, atgvmls7071, HANASitter can check the first tenant, HSI

Example:
Here HANASitter is executed with the Critical Feature (-cf) check that allows max 3 running threads. After the Online check, HANASitter will do a Ping check and then the Feature check. In this example the first host, atgvmls7071, broke the Feature check, so one Call Stack (-nc 1) was written by this host only (-hm true):

```
hsiadm@atgvmls7071:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,3' -nc 1 -hm true -oi 60 -k T1KEY

HANASitter executed 2021-07-20 18:37:20 with
hanasitter.py -cf M_SERVICE_THREADS,THREAD_STATE,Running,3 -nc 1 -hm true -oi 60 -k T1KEY
as T1KEY: KEY T1KEY
  ENV : atgvmls7071:30015
  USER: SYSTEM
  DATABASE: HSI
```

```
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2021-07-20 18:37:22   ,     -           , True         , True       , # index services: 2, # running services: 9 ou
t of 9
Ping Check        , 2021-07-20 18:38:24   , 0:00:00.164630  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2021-07-20 18:38:25   , 0:00:00.665678  , True         , False       , # CFs = 4 atgvmls7071, max allowed = 3, chec
k: THREAD_STATE = 'Running'
Feature Check 1   , 2021-07-20 18:38:25   , 0:00:00.665678  , True         , True       , # CFs = 3 atgvmls7072, max allowed = 3, check
: THREAD_STATE = 'Running'
Call Stack Record , 2021-07-20 18:38:25   , 0:00:00.183103  ,   -          ,   -        , /tmp/hanasitter_output/callstack_atgvmls7071_
HSI_30003__HSI_2021-07-20_18-38-25.txt
```

Note: This could of course been executed via a CRON job instead (see last slide)

HANASitter – Scale Out Example (3/9)

On the first host, atgvmls7071, HANASitter can check the second tenant, HS2

Example:
Here HANASitter is executed with the Critical Feature (-cf) check that allows max 0 running threads. After the Online check, HANASitter will do a Ping check and then the Feature check. In this example the first host, atgvmls7071 (the only host where there was an indexserver of HS2), broke the Feature check, so one Call Stack (-nc 1) was written by this host only (-hm true):

```
hsiadm@atgvmls7071:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,0' -nc 1 -hm true -oi 60 -k T2KEY

HANASitter executed 2021-07-20 18:52:10 with
hanasitter.py -cf M_SERVICE_THREADS,THREAD_STATE,Running,0 -nc 1 -hm true -oi 60 -k T2KEY
as T2KEY: KEY T2KEY
  ENV : atgvmls7071:30044
  USER: SYSTEM
  DATABASE: HS2
```

```
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2021-07-20 18:52:12   ,     -           , True         , True       , # index services: 2, # running services: 9 ou
t of 9
Primary Check     , 2021-07-20 18:52:13   ,     -           , True         , True       ,
Ping Check        , 2021-07-20 18:52:13   , 0:00:00.164622  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2021-07-20 18:52:14   , 0:00:00.415285  , True         , False       , # CFs = 1 atgvmls7071, max allowed = 0, chec
k: THREAD_STATE = 'Running'
Call Stack Record , 2021-07-20 18:52:14   , 0:00:00.144097  ,   -          ,   -        , /tmp/hanasitter_output/callstack_atgvmls7071_
HSI_30043__HS2_2021-07-20_18-52-14.txt
```

Note: This could of course been executed via a CRON job instead (see last slide)

HANASitter – Scale Out Example (4/9)

On the second host, atgvmls7072, HANASitter can check the first tenant, HSI

Example:

Here HANASitter is executed with the Critical Feature (-cf) check that allows max 2 running threads. After the Online check, HANASitter will do a Ping check and then the Feature check. In this example both hosts, atgvmls7071 and atgvml7072, broke the Feature check, so one Call Stack (-nc 1) was written by both hosts (-hm true)

```
hsiadm@atgvmls7072:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,2' -nc 1 -oi 60 -hm true -k T1KEY

HANASitter executed 2021-07-20 19:04:11 with
hanasitter.py -cf M_SERVICE_THREADS,THREAD_STATE,Running,2 -nc 1 -oi 60 -hm true -k T1KEY
as T1KEY: KEY T1KEY
  ENV : atgvmls7071:30015
  USER: SYSTEM
  DATABASE: HSI
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2021-07-20 19:04:12   ,     -           , True         , True       , # index services: 1, # running services: 8 ou
t of 8
Primary Check     , 2021-07-20 19:04:13   ,     -           , True         , True       ,
Ping Check        , 2021-07-20 19:04:13   , 0:00:00.164055  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2021-07-20 19:04:14   , 0:00:00.614764  , True         , False       , # CFs = 3 atgvmls7071, max allowed = 2, chec
k: THREAD_STATE = 'Running'
Feature Check 1   , 2021-07-20 19:04:14   , 0:00:00.614764  , True         , False       , # CFs = 3 atgvmls7072, max allowed = 2, chec
k: THREAD_STATE = 'Running'
Call Stack Record , 2021-07-20 19:04:14   , 0:00:00.136588  ,   -          ,   -        , /tmp/hanasitter_output/callstack_atgvmls7072_
HSI_30003__HSI_2021-07-20_19-04-14.txt
Call Stack Record , 2021-07-20 19:04:14   , 0:00:00.144659  ,   -          ,   -        , /tmp/hanasitter_output/callstack_atgvmls7071_
HSI_30003__HSI_2021-07-20_19-04-14.txt
```

Note: This could of course been executed via a CRON job instead (see last slide)

HANASitter – Scale Out Example (5/9)

On the second host, atgvmls7072, HANASitter can check the first tenant, HSI

Example:

HANASitter is executed with the Critical Feature (-cf) check that allows max 2 running threads. After the Online check, HANASitter does a Ping check and then a Feature check. It is defined that HANASitter will continue the checks after recording and after sleeping 60 seconds (-ar 60). In this example both hosts break the Feature check and write a Call Stack (-hm true). HANASitter sleeps 60 seconds, and then keeps checking …

```
hsiadm@atgvmls7072:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,2' -nc 1 -hm true -ar 60 -oi 60 -k T
1KEY
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp             , Duration        , Successful   , Result     , Comment
Online Check      , 2021-07-20 20:01:30   ,     -           , True         , True       , # index services: 1, # running services: 8 ou
t of 8
Primary Check     , 2021-07-20 20:01:31   ,     -           , True         , True       ,
Ping Check        , 2021-07-20 20:01:31   , 0:00:00.164188  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2021-07-20 20:01:32   , 0:00:00.715000  , True         , False       , # CFs = 4 atgvmls7071, max allowed = 2, chec
k: THREAD_STATE = 'Running'
Feature Check 1   , 2021-07-20 20:01:32   , 0:00:00.715000  , True         , False       , # CFs = 3 atgvmls7072, max allowed = 2, chec
k: THREAD_STATE = 'Running'
Call Stack Record , 2021-07-20 20:01:32   , 0:00:00.136699  ,   -          ,   -         , /tmp/hanasitter_output/callstack_atgvmls7072_
HSI_30003__HSI_2021-07-20_20-01-32.txt
Call Stack Record , 2021-07-20 20:01:33   , 0:00:00.213937  ,   -          ,   -         , /tmp/hanasitter_output/callstack_atgvmls7071_
HSI_30003__HSI_2021-07-20_20-01-32.txt
Online Check      , 2021-07-20 20:03:33   ,     -           , True         , True       , # index services: 1, # running services: 8 ou
t of 8
Primary Check     , 2021-07-20 20:03:34   ,     -           , True         , True       ,
Ping Check        , 2021-07-20 20:03:34   , 0:00:00.164020  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2021-07-20 20:03:35   , 0:00:00.565222  , True         , False       , # CFs = 4 atgvmls7071, max allowed = 2, chec
```

Note: This could of course been executed via a CRON job instead (see last slide)

HANASitter – Scale Out Example (6/9)

On the third host, atgvmls7073, HANASitter can wait for the first tenant, HSI

Example:
Here HANASitter is executed with the Critical Feature (-cf) check that allows max 3 running threads. The Online check finds out that this host is a Stand By (there are 0 indexservers) and therefore decides to sleep for 60 seconds (-oi 60):

```
hsiadm@atgvmls7073:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,3' -nc 1 -oi 60 -hm true -k T1KEY

HANASitter executed 2021-07-20 19:23:09 with
hanasitter.py -cf M_SERVICE_THREADS,THREAD_STATE,Running,3 -nc 1 -oi 60 -hm true -k T1KEY
as T1KEY: KEY T1KEY
  ENV : atgvmls7071:30015
  USER: SYSTEM
  DATABASE: HSI

Online Check      , 2021-07-20 19:23:09    ,     -           , True         , False       , # index services: 0, # running services: 5 o
ut of 5

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 60 seconds break.

Online Check      , 2021-07-20 19:24:09    ,     -           , True         , False       , # index services: 0, # running services: 5 o
ut of 5

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 60 seconds break.
```

Note: This could of course been executed via a CRON job instead (see last slide)

HANASitter – Scale Out Example (7/9)

If second host, atgvmls7072, crashes, a failover to third host, atgvmls7073, will happen

Example:
Here second host, atgvmls7072, is stopped (HDB stop) to simulate a crash. Third host, atgvmls7073, takes over as the slave node

SYSTEMDB@HSI (SYSTEM) [Production System] atgvmls7071.wdf.sap.corp 00 Last Update: 20.07.2021 16:54:30 Inte

Overview | Landscape | Alerts | Performance | Volumes | Configuration | System Information | Diagnosis Files | Trace Configuration

Services | Hosts | Redistribution | System Replication

Host	Active	Host Status	Name Server Role (Configured)	Name Server Role (Actual)	Index Server Role (Configured)	Index Server Role (Actual)
atgvmls7071	YES	OK	MASTER 1	MASTER	WORKER	MASTER
atgvmls7072	YES	OK	MASTER 3	SLAVE	WORKER	SLAVE
atgvmls7073	YES	IGNORE	MASTER 2	SLAVE	STANDBY	STANDBY

```
hsiadm@atgvmls7072:/usr/sap/HSI> HDB stop
```

SYSTEMDB@HSI (SYSTEM) [Production System] atgvmls7071.wdf.sap.corp 00 Last Update: 20.07.2021 19:31:24 Interval: 60

Overview | Landscape | Alerts | Performance | Volumes | Configuration | System Information | Diagnosis Files | Trace Configuration

Services | Hosts | Redistribution | System Replication

Host	Active	Host Status	Failover Status	Name Server Role (Configured)	Name Server Role (Actual)	Index Server Role (Configured)	Index Server Role (Actual)
atgvmls7071	YES	OK		MASTER 1	MASTER	WORKER	MASTER
atgvmls7072	NO	INFO		MASTER 3	SLAVE	WORKER	STANDBY
atgvmls7073	YES	INFO		MASTER 2	SLAVE	STANDBY	SLAVE

HANASitter – Scale Out Example (8/9)

On the third host, atgvmls7073, HANASitter continues automatically with further checks

Example:
After the failover, the Online Check goes through. HANASitter continues with the Ping check and the Feature check. The first node, atgvmls7071, breaks the Feature Check and records a Call Stack:

```
hsiadm@atgvmls7073:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,3' -nc 1 -oi 60 -hm true -k T1KEY
```

```
Online Check      , 2021-07-20 19:23:09    ,     -           , True         , False       , # index services: 0, # running services: 5 o
ut of 5

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 60 seconds break.
```

```
 - - - - - Start HANASitter - - - - - -
Action            , Timestamp             , Duration         , Successful   , Result     , Comment
Online Check      , 2021-07-20 19:31:11    ,     -            , True         , True       , # index services: 1
, # running services: 8 out of 8
Ping Check        , 2021-07-20 19:33:14    , 0:00:00.164149   ,   -          , True       , DB responded faster
 than 60 seconds
Feature Check 1   , 2021-07-20 19:33:15    , 0:00:00.565024   , True         , False       , # CFs = 4 atgvmls7
071, max allowed = 3, check: THREAD_STATE = 'Running'
Feature Check 1   , 2021-07-20 19:33:15    , 0:00:00.565024   , True         , True       , # CFs = 3 atgvmls70
73, max allowed = 3, check: THREAD_STATE = 'Running'
Call Stack Record , 2021-07-20 19:33:15    , 0:00:00.251892   ,   -          ,   -        , /tmp/hanasitter_out
put/callstack_atgvmls7071_HSI_30003__HSI_2021-07-20_19-33-15.txt
```

Note: This could of course been executed via a CRON job instead (see last slide)

HANASitter – Scale Out Example (9/9)

On the second host, atgvmls7072, HANASitter waits

Example:

When the HANA crash (HDB stop) happened on the second host, atgvmls7072, HANASitter started to wait for the Online Check to go through:

```
hsiadm@atgvmls7072:/tmp/HANASitter> python hanasitter.py -cf 'M_SERVICE_THREADS,THREAD_STATE,Running,2' -nc 1 -hm true -ar 60 -oi 60 -k T
1KEY
Online Check      , 2021-07-20 20:03:33    ,     -           , True         , True       , # index services: 1, # running services: 8 ou
t of 8
Primary Check     , 2021-07-20 20:03:34    ,     -           , True         , True       ,
Ping Check        , 2021-07-20 20:03:34    , 0:00:00.164020  ,   -          , True       , DB responded faster than 60 seconds
Feature Check 1   , 2021-07-20 20:03:35    , 0:00:00.565222  , True         , False       , # CFs = 4 atgvmls7071, max allowed = 2, chec
k: THREAD_STATE = 'Running'
Feature Check 1   , 2021-07-20 20:03:35    , 0:00:00.565222  , True         , False       , # CFs = 3 atgvmls7072, max allowed = 2, chec
k: THREAD_STATE = 'Running'
Call Stack Record , 2021-07-20 20:03:35    , 0:00:00.101387  ,   -          ,   -         , /tmp/hanasitter_output/callstack_atgvmls7072_
HSI_30003__HSI_2021-07-20_20-03-35.txt
Call Stack Record , 2021-07-20 20:03:35    , 0:00:00.156070  ,   -          ,   -         , /tmp/hanasitter_output/callstack_atgvmls7071_
HSI_30003__HSI_2021-07-20_20-03-35.txt
Online Check      , 2021-07-20 20:05:35    ,     -           , True         , False       , # index services: 0, # running services: 0 o
ut of 1

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 60 seconds break.

Online Check      , 2021-07-20 20:06:35    ,     -           , True         , False       , # index services: 0, # running services: 5 o
ut of 5

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 60 seconds break.
```

Note: The Ping Check will never raise any issue if only one host is down (the implementation of making the Ping Check host specific would have been to use the hint ROUT_TO(<volume_id>,…), but to get the volume_id a SELECT on M_VOLUMES would be needed before the Ping Check, which would destroy the purpose of the Ping Check)

HANASitter – If HANA Goes Offline

HANASitter has a Online check so that it will not start tracking if DB is offline
Additionally, if HANA goes offline during tracking, it will exit tracking and start with the online check

Example: Here HANA is turned off during tracking so the 4th Ping Check finds that the DB is offline:

```
HANASitter executed 2017-11-24 17:32:48 with
hanasitter.py -ci 30 -pt 30 -nc 1

Host = mo-fc8d991e0, SID = CH0, DB Instance = 00
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 30 seconds, Timeout = 30 seconds
Feature Checks: Interval 30 seconds, Timeout = 60 seconds
Feature Check 1 allows only 30 times that IS_ACTIVE = 'TRUE' in M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [milliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                      ,    0
Call Stack          , 1                   ,   60                  ,
RTE Dumps           , 0                   ,   60                  ,
Recording Priority: RTE   Call Stacks   G-Stacks   Kernel Profiler
After Recording: Exit
Action            , Timestamp             , Duration          , Successful   , Result     , Comment
Online Check      , 2017-11-24 17:32:48   ,     -             , True         , True       , Number running services: 11 out of 11
Primary Check     , 2017-11-24 17:32:54   ,     -             , True         , True       ,
Ping Check        , 2017-11-24 17:32:54   , 0:00:00.164599    ,   -          , True       , DB responded faster than 30 seconds
Feature Check 1   , 2017-11-24 17:32:54   , 0:00:00.264682    , True         , True       , # Critical Features = 11 (allowed = 30)
Ping Check        , 2017-11-24 17:33:24   , 0:00:00.164389    ,   -          , True       , DB responded faster than 30 seconds
Feature Check 1   , 2017-11-24 17:33:25   , 0:00:00.214380    , True         , True       , # Critical Features = 10 (allowed = 30)
Ping Check        , 2017-11-24 17:33:55   , 0:00:00.164413    ,   -          , True       , DB responded faster than 30 seconds
Feature Check 1   , 2017-11-24 17:33:55   , 0:00:00.214246    , True         , True       , # Critical Features = 10 (allowed = 30)
Ping Check        , 2017-11-24 17:35:00   , 0:00:34.455389    ,   -          , False      , DB is offline, will exit the tracker
Online Check      , 2017-11-24 17:35:00   ,     -             , True         , False      , Number running services: 3 out of 11

One of the online checks found out that this HANA instance is not online. HANASitter will now have a 3600 seconds break.
```

HANASitter – Configuration File

HANASitter can be controlled with a configuration file
(additional flags given will overwrite flags in the configuration file)

Flag	Unit	Details	Explanation	Default
-ff		flag file	full path to the configuration file	

Example:

```
ha1adm@dewdfglp00766:/tmp/HANASitter> more hanasitter_configfile.txt
MY HANASITTER CONFIGURATION FILE
If more than 20 threads have been in state TREAD_STATE=Running for more than 10 seconds
-cf M_SERVICE_THREADS,THREAD_STATE,Running,20,10
then 2 call stacks
-nc 2
with 30 seconds between them
-ic 30
are recorded. This is the key in hdbuserstore that is used:
-k SYSTEMKEY
ha1adm@dewdfglp00766:/tmp/HANASitter> python hanasitter.py -ff hanasitter_configfile.txt
Host = dewdfglp00766, DB Instance = 00, Single DB System
Online, Primary and Not-Secondary Check: Interval = 3600 seconds
Ping Check: Interval = 60 seconds, Timeout = 60 seconds
Feature Checks: Interval 60 seconds, Timeout = 60 seconds
Feature Check 1, allows maximum 20 features in the state, THREAD_STATE = Running, for > 10 seconds, in the view, M_SERVICE_THREADS
Recording mode: 1
Recording Type      , Number Recordings   ,   Intervals [seconds] ,   Durations [seconds]      ,    Wait [miliseconds]
GStack              , 0                   ,   60                  ,
Kernel Profiler     , 0                   ,   60                  ,   60                       ,    0
Call Stack          , 2                   ,   30                  ,
RTE Dumps           , 0                   ,   60                  ,
After Recording: Exit
Action            , Timestamp             , Duration         , Successful   , Result      , Comment
Online Check      , 2017-06-09 11:54:21   ,     -            , True         , True        , Number running services: 7 out of 7
Primary Check     , 2017-06-09 11:54:23   ,     -            , True         , True        ,
Non-standby Check , 2017-06-09 11:54:23   ,     -            , True         , True        ,
Ping Check        , 2017-06-09 11:54:24   , 0:00:00.164766   ,   -          , True        , DB responded faster than 60 seconds
Feature Check 1   , 2017-06-09 11:54:24   , 0:00:00.314682   , True         , True        , # Critical Features = 0 (allowed = 20),
```

Monitoring

HANASitter – output

To control the output of the hanasitter there are these flags

Flag	Unit	Details	Explanation	Default
-od		output directory	full path of the folder where the ouput files will end up (if the folder does not exist it will be created)	/tmp/hanasitter_output
-ol		log output directory	full path of the folder where the HANASitter log filles will end up (if not exist, it will be created)	/tmp/hanasitter_output
-so		standard out switch	true: write to std out, false: do not write to std out	true

Example:
Here output folders are deleted and then automatically created again by hanasitter:

```
rm -r ../hanasitter_output/
rm -r ../hanasitter_logs/
python hanasitter.py -cf "M_SERVICE_THREADS,THREAD_STATE,Running,3" -nc 2 -ol /tmp/hanasitter_logs
```

```
Feature Check 1    , 2019-12-03 21:43:39    , 0:00:00.365156    , True          , False         , # CFs = 5 for , max allowed = 3, check: THREAD_STATE = 'Ru
nning'
Call Stack Record , 2019-12-03 21:43:39    , 0:00:00.104933    ,   -           ,   -           , /tmp/hanasitter_output/callstack_atgvmls866_XSC_2019-12-03_
21-43-39.txt
Call Stack Record , 2019-12-03 21:44:39    , 0:00:00.143667    ,   -           ,   -           , /tmp/hanasitter_output/callstack_atgvmls866_XSC_2019-12-03_
21-44-39.txt
```

```
xscadm@atgvmls866:/tmp/HANASitter> ls ../hanasitter_output/
callstack_atgvmls866_XSC_2019-12-03_21-43-39.txt  callstack_atgvmls866_XSC_2019-12-03_21-44-39.txt
xscadm@atgvmls866:/tmp/HANASitter> ls ../hanasitter_logs/
hanasitterlog_2019-12-03.txt
```

HANASitter – output

Automatic house keeping of the hanasitter logs with -or flag

Flag	Unit	Details	Explanation	Default
-or		log retention days	hanasitterlogs in the path specified with -od are only saved for this number of days	-1 (not used)

Example:
Here there are two old hanasitterlog files and after a run of hanasitter with -or they are removed and there is now only the new hanasitterlog file left

```
oqladm@ls80010:/tmp/HANASitter> ll ../hanasitter_output/
total 224
-rw-r----- 1 oqladm sapsys  25092 Dec  6 13:18 hanasitterlog_2017-12-06.txt
-rw-r----- 1 oqladm sapsys  19134 Dec 26 17:31 hanasitterlog_2017-12-26.txt
-rw-r--r-- 1 oqladm sapsys 102842 Dec  6 11:29 kernel_profiler_cpu_ls80010_OQL_2017-12-06_11-28-36.dot
-rw-r--r-- 1 oqladm sapsys  65260 Dec  6 11:29 kernel_profiler_wait_ls80010_OQL_2017-12-06_11-28-36.dot
oqladm@ls80010:/tmp/HANASitter>
```

```
oqladm@ls80010:/tmp/HANASitter> python hanasitter.py -nc 1 -or 2
2 hanasitter daily log files were removed
Ping Check         , 2018-01-05 12:09:59     , 0:00:00.163895    ,    -          , True
Feature Check 1    , 2018-01-05 12:10:00     , 0:00:00.163768    , True          , True
```

```
oqladm@ls80010:/tmp/HANASitter> ll ../hanasitter_output/
total 180
-rw-r----- 1 oqladm sapsys   1968 Jan  5 12:10 hanasitterlog_2018-01-05.txt
-rw-r--r-- 1 oqladm sapsys 102842 Dec  6 11:29 kernel_profiler_cpu_ls80010_OQL_2017-12-06_11-28-36.dot
-rw-r--r-- 1 oqladm sapsys  65260 Dec  6 11:29 kernel_profiler_wait_ls80010_OQL_2017-12-06_11-28-36.dot
oqladm@ls80010:/tmp/HANASitter>
```

HANASitter & CRON (1/2)

HANASitter can be scheduled with CRON to run in background and so that the terminal can be closed

Note: hanasitter expects the environment of <sid>adm → source /bin/bash (or equivalent)

Note: cron will try to start hanasitter every minute, but thanks to a lock, it will not start until the first hanasitter process stopped

This shell script, hanasitter.sh, provides the <sid>adm environment, with **`source $HOME/.bashrc`** and then executes hanasitter:

```
xscadm@atgvmls866:/tmp/HANASitter> more hanasitter.sh
#! /bin/bash
source $HOME/.bashrc
python /tmp/HANASitter/hanasitter.py -ff /tmp/HANASitter/hanasitter_configfile.txt -od /tmp/hanasitter_output
```

Then a new crontab can be created, calling this shell script, e.g. once every minute (`* * * * *` = once every minute), but only if the previous lock is gone:

```
xscadm@atgvmls866:/tmp/HANASitter> crontab -e
crontab: installing new crontab
xscadm@atgvmls866:/tmp/HANASitter> crontab -l
* * * * * /usr/bin/flock -xn /tmp/HANASitter/hanasitter.lck -c "/tmp/HANASitter/hanasitter.sh"
```

One can check with ps that hanasitter is running in background:

```
xscadm@atgvmls866:/tmp/HANASitter> ps aux | grep sitter
xscadm    4429  0.0  0.0   8752  1456 ?        Ss   21:03   0:00 /usr/bin/flock -xn /tmp/HANASitter/hanasitter.
xscadm    4432  0.0  0.0  12132  3120 ?        S    21:03   0:00 /bin/bash /tmp/HANASitter/hanasitter.sh
xscadm    4459  0.0  0.0  97252 10996 ?        S    21:03   0:00 python /tmp/HANASitter/hanasitter.py -ff /tmp/
```

HANASitter & CRON (2/2)

If HANASitter was started with cron as in previous slide, it will continue until the crontab is changed

Edit the crontab:

```
xscadm@atgvmls866:/tmp/HANASitter> crontab -e
```

```
* * * * * /usr/bin/flock -xn /tmp/HANASitter/hanasitter.lck -c "/tmp/HANASitter/hanasitter.sh"
```

E.g. comment out the line that is restarting HANASitter:

```
#* * * * * /usr/bin/flock -xn /tmp/HANASitter/hanasitter.lck -c "/tmp/HANASitter/hanasitter.sh"
```

Then exit vim by saving this change (esc : x):

```
crontab: installing new crontab
```

Now we can kill the current running HANASitter by first checking the PIDs:

```
xscadm@atgvmls866:/tmp/HANASitter> ps aux | grep sitter
xscadm    6021  0.0  0.0   8752  1504 ?        Ss   21:09   0:00 /usr/bin/flock -xn /tmp/HANASitter/hanasitter
xscadm    6023  0.0  0.0  12132  2996 ?        S    21:09   0:00 /bin/bash /tmp/HANASitter/hanasitter.sh
xscadm    6050  0.0  0.0  97252 10912 ?        S    21:09   0:00 python /tmp/HANASitter/hanasitter.py -ff /tmp/
```

Then stopping hanasitter by using kill -9:

```
xscadm@atgvmls866:/tmp/HANASitter> kill -9 6021 6023 6050
```

Double check that no HANASitter restarted:

```
xscadm@atgvmls866:/tmp/HANASitter> ps aux | grep sitter
xscadm    9847  0.0  0.0  10548  1620 pts/0    S+   21:24   0:00 grep --color=auto sitter
```