netaxis
SOLUTIONS

Performance Analysis Tools

Technical Note

Table of Contents

1 Introduction

This document describes several tools that are available on SRE to analyze the performance of individual service logic nodes or complete service logics.

These tools are to be used by advanced users as performing load tests on a live system may impact its performance. Executing logic with wrong input data may corrupt provisioned data.

2 Individual Node Testing

The tool sre-admin allows the user to test a single node execution by providing the node type, its parameters and the input call descriptor. The tool will then initialize a new node with these parameters and instruct it to process the input call descriptor.

A common issue on RHEL-derived Linux distributions is that the tool sre-admin fails to start because the locales are not correcly configured. You should fix the configuration first or you can temporarily set the locale to a fallback value:

```
export LC_ALL=en_US.utf8
```

The tool is available by running the command:

```
# /opt/sre/bin/sre-admin service-logic test-node
```

The usage help is available by running:

```
# /opt/sre/bin/sre-admin service-logic test-node --help
Usage: sre-admin service-logic test-node [OPTIONS] NODE_TYPE VALUES
                                         CALL_DESCRIPTOR

  Test single service logic node performance by supplying configuration values
  and input call descriptor

Options:
  --iterations INTEGER  Number of iterations to perform
  --interval FLOAT      Pause between runs
  --disable-cache       Disable caching entirely by re-initializing the node
                        before each run
  --disable-cd-print    Disable printing of output CD for each run
  --yes                 Confirm the action without prompting.
  --help                Show this message and exit.
```

Getting the node parameters and node type may prove difficult. One of the easiest method to acquire them is to build an SL and export it. The resulting file is in JSON format. The `nodes` object contains all the nodes. From this list of nodes, the `name` can help pinpoint the desired node. The type value provides the `node` type to be used as `NODE_TYPE` parameter and the `values` object holds the input parameters to be used as `VALUES`.

2.1 Sample Execution

The sample execution hereafter shows a database query node being executed with an empty call descriptor (as this node does not use any variable from the CD). In this node, caching is configured on 5 secs. This example will be re-used in the following chapters to run the tool in other scenarios.

```
# /opt/sre/bin/sre-admin service-logic test-node query.queryDatabaseGeneric '{"
    ↪ tables": ["inventory.instance"], "fields": [{"field": "instance.address1
    ↪ ", "storeInto": "address"}], "joins": [], "conditions": [], "logic": "and
    ↪ ", "orderBy": [], "offset": "", "joinType": "JOIN", "fetch": "all", "
    ↪ storeIntoRecordsList": "instances", "ifRecordFound": "", "ifnoRecordFound
    ↪ ": "", "caching": 5}' '{}'
Runs (msecs)
  Run #    Duration   Result type            Call descriptor
-------  ----------  ---------------------
    ↪ --------------------------------------------------------------------------------------
    ↪
      0    4.41289   jump (ifRecordFound:)  {'instances': [{"address": "
    ↪ 172.16.0.189"}, {"address": "172.16.0.39"}, {"address": "172.18.0.175"},
    ↪ {"address": "172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "
    ↪ 172.16.0.142"}, {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {
    ↪ "address": "10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "
    ↪ 172.18.4.105"}]}
...
    999    0.102043  jump (ifRecordFound:)  {'instances': [{"address": "
    ↪ 172.16.0.189"}, {"address": "172.16.0.39"}, {"address": "172.18.0.175"},
    ↪ {"address": "172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "
    ↪ 172.16.0.142"}, {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {
    ↪ "address": "10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "
    ↪ 172.18.4.105"}]}
Statistics (msecs)
    Max        Min       Mean     Median    Variance     Total
-------  ---------  ---------  ---------  ----------  -------
5.21827  0.080824  0.162545  0.102162    0.000147  162.545
```

By default, the tool will run 1000 executions spaced 100 msecs apart. At the end of the test, the max, min, mean, median and variance execution durations are printed out.

2.2 Validating Caching

Since this sample test use a 5 secs cache, the effect of the caching is visible by performing 10 executions spaced 1 sec apart:

```
# /opt/sre/bin/sre-admin service-logic test-node query.queryDatabaseGeneric '{"
    ↪ tables": ["inventory.instance"], "fields": [{"field": "instance.address1
    ↪ ", "storeInto": "address"}], "joins": [], "conditions": [], "logic": "and
    ↪ ", "orderBy": [], "offset": "", "joinType": "JOIN", "fetch": "all", "
    ↪ storeIntoRecordsList": "instances", "ifRecordFound": "", "ifnoRecordFound
    ↪ ": "", "caching": 5}' '{}' --iterations 10 --interval 1 --disable-cd-
    ↪ print
Runs (msecs)
  Run #    Duration   Result type
-------  ----------  ---------------------
      0    3.05128   jump (ifRecordFound:)
      1    0.096083  jump (ifRecordFound:)
      2    0.169039  jump (ifRecordFound:)
      3    0.105143  jump (ifRecordFound:)
      4    0.134706  jump (ifRecordFound:)
      5    1.64461   jump (ifRecordFound:)
      6    0.145674  jump (ifRecordFound:)
      7    0.096083  jump (ifRecordFound:)
      8    0.122786  jump (ifRecordFound:)
      9    0.094652  jump (ifRecordFound:)
Statistics (msecs)
    Max        Min       Mean     Median    Variance     Total
```

```
-------  ---------  ---------  ---------  ----------  --------
3.05128  0.094652  0.566006  0.128746    0.000993  5.66006
```

In this scenario, it can be observed that the initial setup requires some time (most probably, for the database connection setup) but afterwards, the result is cached and so, the executions are faster. After 5 secs, the caching is expired and the DB is effectively queried again. This execution is slower. Subsequent executions are faster again.

2.3 Initial Node Setup Delay

By using the option `--disable-cache`, the node is re-initialised before each execution. This is the same effect as a development service logic which is reloaded at regular intervals, except here, it occurs before each execution.

```
# /opt/sre/bin/sre-admin service-logic test-node query.queryDatabaseGeneric '{"tables": ["inventory.instance"], "fields": [{"field": "instance.address1", "storeInto": "address"}], "joins": [], "conditions": [], "logic": "and", "orderBy": [], "offset": "", "joinType": "JOIN", "fetch": "all", "storeIntoRecordsList": "instances", "ifRecordFound": "", "ifnoRecordFound": "", "caching": 5}' '{}' --iterations 10 --interval 1 --disable-cd-print --disable-cache
Runs (msecs)
  Run #    Duration   Result type
-------  ----------  ---------------------
      0    3.35264   jump (ifRecordFound:)
      1    1.55807   jump (ifRecordFound:)
      2    1.7097    jump (ifRecordFound:)
      3    0.870943  jump (ifRecordFound:)
      4    1.26076   jump (ifRecordFound:)
      5    1.1363    jump (ifRecordFound:)
      6    1.29771   jump (ifRecordFound:)
      7    2.63095   jump (ifRecordFound:)
      8    2.20418   jump (ifRecordFound:)
      9    1.369     jump (ifRecordFound:)
Statistics (msecs)
    Max        Min      Mean     Median     Variance     Total
-------  ---------  -------  ---------  ----------  --------
3.35264  0.870943  1.73903   1.46353    0.000592  17.3903
```

In this scenario, it can be observed that caching never kicks in because the node is re-initialized each time.

3 Service Logic Testing

The tool sre-admin allows the user to test a complete service logic execution by providing the service logic id and the input call descriptor. The tool will then initialize the service logic and process the input call descriptor.

The tool is available by running the command:

```
# /opt/sre/bin/sre-admin service-logic  test-service-logic
```

The usage help is available by running:

```
# /opt/sre/bin/sre-admin service-logic test-service-logic --help
Usage: sre-admin service-logic test-service-logic [OPTIONS] SERVICE_LOGIC_ID
                                                  CALL_DESCRIPTOR

  Run several times a service logic by providing an input call descriptor and
  aggregate performance statistics. SERVICE-LOGIC-ID may either be an
  interface id (e.g. sip, enum, ...) to use the service logic currently active
  for that interface or a numerical id.

Options:
  --iterations INTEGER  Number of iterations to perform
  --interval FLOAT      Pause between runs
  --disable-cache       Disable caching entirely by re-initializing the
                        service logic before each run
  --disable-cd-print    Disable printing of output CD for each run
  --enable-trace        Enable tracing of call
  --yes                 Confirm the action without prompting.
  --help                Show this message and exit.
```

It is possible to retrieve the service logic id by opening it with any browser. The last number in the URL is the service logic id (e.g. in the URL http://10.0.161.180:8080/serviceLogic/edit/1, the SL id is 1).

3.1 Sample Execution

The sample execution hereafter shows a SL execution repeated several times, with an empty call descriptor.

```
# /opt/sre/bin/sre-admin service-logic test-service-logic 1 '{}'
Runs (msecs)
  Run #    Duration  Result type    Call descriptor
-------  ----------  -------------
    ↪ ---------------------------------------------------------------------------------------
    ↪
  0    8.28671                  {}
  1    1.27482                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  2    1.21617                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  3    0.993252                 {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  4    1.30034                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  5    7.69997                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  6    1.15013                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  7    2.69771                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  8    1.00636                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
  9    2.11883                  {'instances': [{"address": "172.16.0.189"},
↪   {"address": "172.16.0.39"}, {"address": "172.18.0.175"}, {"address": "
↪  172.18.0.237"}, {"address": "172.16.0.54"}, {"address": "172.16.0.142"},
↪  {"address": "172.18.0.59"}, {"address": "172.18.0.34"}, {"address": "
↪  10.50.0.20"}, {"address": "172.16.0.2"}, {"address": "172.18.4.105"}]}
```

```
Statistics (msecs)
    Max       Min     Mean    Median    Variance    Total
-------  --------  -------  --------  ----------  -------
8.28671  0.993252  2.77443    1.28758    0.007874  27.7443
```

By default, the tool will run 10 executions spaced 100 msecs apart. At the end of the test, the max, min, mean, median and variance execution durations are printed out.

The same options as available as for the individual node test tool. So it is possible to:

- adjust the number of executions
- adjust the interval between runs
- disable caching by re-loading the service logic before each execution

Moreover, it is also possible to enable tracing for these execution by using the option `--enable-trace`. In that case, tracing log is dumped to sre.log.

4 Broker Benchmark tool

> Releases 4.0.0+

The broker benchmark tool allows the user to simulate a SIP message being sent to the broker (process sre-broker) and getting the result from it. The broker dispatchs the message to a call processor (process sre-call-processor). The tool allows running a single execution as well as multiple executions in parallel and sequence.

The tool is available by running the command:

```
# /opt/sre/bin/sre-admin benchmark broker
```

The usage help is available by running:

```
# /opt/sre/bin/sre-admin benchmark broker --help
Usage: sre-admin benchmark broker [OPTIONS] TEMPLATE_PARAMS

  Performs a benchmark of the broker by sending a number of message processing
  requests. The SIP message to be sent out is built by selecting a built-in
  template or a file-based template and a set of template parameters, in JSON
  format. The template message may contain placeholders in the format
  [placeholder]. These template parameters, if not provided, are defaulted:
  branch, callId, counter, destinationAddress, destinationPort, fromTag,
  sourceAddress, sourcePort.

Options:
  -r INTEGER                    Limit requests/sec (0=unlimited)  [default: 0]
  -c INTEGER                    Number of client threads  [default: 1]
  -m INTEGER                    Number of requests  [default: 1]
  --template TEXT               SIP message template name (use --templates-
                                list to list the built-in templates) or path
                                to file  [default: basic]
  --templates-list              List the built-in templates
  --extra-call-descriptor TEXT  Extra call descriptor variables in JSON
                                format, which will be added to the set of
                                variables, after the standard ones have been
                                extracted from the generated SIP message
                                [default: {}]
  --yes                         Confirm the action without prompting.
  --help                        Show this message and exit.
```

The tool is based around the use of message templates and the use of template parameters. These message templates must be valid SIP messages with the variable parts replaced by the SRE `[placeholder]` syntax. To run the tool, the name of a built-in template or a template file must be provided, along with the template parameters that will be used to resolve the placeholders.

The built-in templates can be displayed by running the comand:

```
# /opt/sre/bin/sre-admin benchmark broker '{}' --templates-list

basic
INVITE sip:[called]@ruri.netaxis.cloud SIP/2.0
Via: SIP/2.0/UDP 10.0.0.44:5096;branch=z9hG4bK-[branch]
From: "Alice" <sip:[calling]@from.netaxis.cloud>;tag=[fromTag]
To: "Bob" <sip:[called]@to.netaxis.cloud>
Call-ID: [callId]
CSeq: 102 INVITE
Contact: <sip:[calling]@contact.netaxis.cloud>
Expires: 10
Content-Type: application/sdp
Content-Length: 190
Accept: application/sdp

v=0
o=Sigma 27106 3566 IN IP4 1.2.3.4
s=SIP Call
c=IN IP4 1.2.3.4
t=0 0
m=audio 10000 RTP/AVP 8 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

In this example, there is a built-in template, named basic, that contains placeholders for calling, called and a few other fields.

Some template parameters, if not provided, are defaulted: branch, callId, counter, destinationAddress, destinationPort, fromTag, sourceAddress, sourcePort.

4.1 Sample SIP Processing

To send a sample SIP message with calling set to 123 and called set to 456, this command can be used:

```
# /opt/sre/bin/sre-admin benchmark broker '{"calling": "123", "called": "456"}'
...
Sample SIP payload
INVITE sip:456@ruri.netaxis.cloud SIP/2.0
Via: SIP/2.0/UDP 10.0.0.44:5096;branch=z9hG4bK-c413bd4d-fba7-42ef-a1bc-dca36adaa791
From: "Alice" <sip:123@from.netaxis.cloud>;tag=5fb2a9d0-4a7e-4598-9ced-1b521b9f01fe
To: "Bob" <sip:456@to.netaxis.cloud>
Call-ID: 61d885f7-4d1e-4008-8bef-6a120410a874
CSeq: 102 INVITE
Contact: <sip:123@contact.netaxis.cloud>
Expires: 10
Content-Type: application/sdp
Content-Length: 190
Accept: application/sdp

v=0
o=Sigma 27106 3566 IN IP4 1.2.3.4
s=SIP Call
c=IN IP4 1.2.3.4
t=0 0
m=audio 10000 RTP/AVP 8 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15


Clients
Thread 0 started
Benchmark started
  [#####################################]  100%
All requests sent, collecting results
Thread BC-00 exiting
```

```
Runs (msecs)
  Run #  Client id      Duration  Result
-------  -----------  ----------
    ↪  ---------------------------------------------------------------------------------------
    ↪
      0  BC-00           136.292  {"_lastRequestHash": "3
    ↪  b8cae1580cb2de010e2ccfb2879c9e4", "actions": [], "callId": "2be8719c
    ↪  -1916-422e-a35d-6600da116784", "destinationAddress": "127.0.0.1", "
    ↪  destinationHost": "", "destinationPort": "5060", "nit": "relay", "
    ↪  persistCallDescriptor": false, "recurse": true, "requestURI": "456-0@sre-
    ↪  sipp-called:5060"}
```

By running the tool, the “resolved” template is first displayed. The result of the execution is dislayed for each run.

4.2 Load Testing

By sending the message several times and without any rate limit, it is possible to measure the peak processing rate and the evolution of processing duration time over time (that may vary depending on caching mechanisms). Example:

```
# /opt/sre/bin/sre-admin benchmark broker '{"calling": "123", "called": "456"}'
    ↪   -m 100
...
Runs (msecs)
  Run #  Client id      Duration  Result
-------  -----------  ----------
    ↪  ---------------------------------------------------------------------------------------
    ↪
      0  BC-00          86.0064   {"_lastRequestHash": "
    ↪  ff44f16019ae2fee38786b6c5f6df393", "actions": [], "callId": "ca88978c-63
    ↪  b8-49a4-834a-0c3e6624ad11", "destinationAddress": "127.0.0.1", "
    ↪  destinationHost": "", "destinationPort": "5060", "nit": "relay", "
    ↪  persistCallDescriptor": false, "recurse": true, "requestURI": "456-0@sre-
    ↪  sipp-called:5060"}
...
     99  BC-00           6.89769  {"_lastRequestHash": "
    ↪  ea141d0a0669c236a98808c147276529", "actions": [], "callId": "321f8a64-1
    ↪  e8e-429e-9261-8158043817f6", "destinationAddress": "127.0.0.1", "
    ↪  destinationHost": "", "destinationPort": "5060", "nit": "relay", "
    ↪  persistCallDescriptor": false, "recurse": true, "requestURI": "456-0@sre-
    ↪  sipp-called:5060"}
Statistics (msecs)
    Max     Min     Mean    Median    Variance    Total    Benchmark duration
    ↪  (~)    Average CAPS
```

```
-------  ------  -------  ---------  ----------  -------
      ↪ ------------------------  --------------
86.0064  4.5917  9.16212   7.21431    0.079751  916.212
      ↪ 1382.93          72.3105
```

When performing load testing, the parameter `-r` can be used to control the rate and the parameter `-c` can be used to control the number of clients to perform parallel executions.

4.3 Custom Call Descriptor

By supplying the parameter `--extra-call-descriptor`, it is possible to provide extra CD variables to the execution. Some of the standard CD variables can not be derrived from the SIP message alone (e.g. counter, sourceAddress, …). In this case, they must be provided ad-hoc. Example:

```
# /opt/sre/bin/sre-admin benchmark broker '{"calling": "123", "called": "456"}'
      ↪  --extra-call-descriptor '{"destinationAddress": "10.20.30.40"}'
...
Runs (msecs)
  Run #  Client id      Duration  Result
-------  -----------  ----------
      ↪ ------------------------------------------------------------------------------------------------------------------------------------
      ↪
      0  BC-00            55.4602  {"_lastRequestHash": "
      ↪ b50513a8fc1af7945b2fc8ff9542e03b", "actions": [], "callId": "bde7478a-25
      ↪ d8-482a-b490-f7242dacfb8d", "destinationAddress": "10.20.30.40", "
      ↪ destinationHost": "", "destinationPort": "5060", "nit": "relay", "
      ↪ persistCallDescriptor": false, "recurse": true, "requestURI": "456-0@sre-
      ↪ sipp-called:5060"}
```