SIC tests Protocol

{{Project/Mission name}}

Service Introduction Center

Version 1.0, 01-Jan-1970

Table of Content

Copy to	N/A
Contact	Service Introduction Center
Project Name & Version	N/A
Writer	{{John Doe}} <{{John.Doe}}@smals.be> ({{JOD}})

Version	Status	Date	Author	Nature of modifications	File location
0.1	WiP	01-Sep-2016	AVM	Initial version	https://git01.smals.be/sic/tools.protocols
{{0.x}}	{{status}}	{{DD-MMM-YYYY}}	{{AUTH}}	{{ Comments }}	
1.0	Final	01-Jan-1970	{{JOD}}	{{ Current version }}	

1. Scope

{{ Scope description... }}

2. Environment

Components	Environment	Tenant
test unit	{{ Where is the component deployed? }}	{{ SIC, socsec, eHealth... }}
database	{{ idem }}	{{ idem }}
ESB	{{ idem }}	{{ idem }}
base WS	{{ idem }}	{{ idem }}
specific WS	{{ idem }}	{{ idem }}

3. Scenarios

S1:main

- *Objective:* {{ Description of the objective(s) of this scenario }}
- *Configurations:* [C1:WebApp], [Cn:Other]
- *Load profiles:* [P1:average], [P2:peak], [P3:saturation]...
- *Sequence:* for each virtual user
  - {{ step 0, e.g. user login }}

    - in a loop:
        - {{ page 1 }}
        - {{ page 2 }}
        - ...
        - {{ page *n* }}
    - {{ final step, e.g. user logout / session destroy, when possible }}

- *Expected observations*
    - [C1:WebApp] & [P1:average]: {{ results we expect with this load profile and configuration, e.g. P98 response time < 500ms, actual rate attained, 0% of errors }}
    - [C1:WebApp] & [P2:peak]: {{ results we expect with this load profile and configuration, e.g. P98 response time < 2000ms, actual rate attained, 0% of errors }}
    - [C1:WebApp] & [P3:saturation]: {{ results we expect with this load profile and configuration, e.g. measure max rate, saturation rate, critical rate... }}
    - [Cn:Other] & [P1:average]: {{ ... }}
    - [Cn:Other] & [P1:peak]: {{ ... }}
    - ...

Maximum rate
: highest rate attained before apparition of errors

Saturation rate
: highest rate of successful requests attained, even in presence of errors

Critical rate
: highest rate attained while respecting [SLA]

Sn:other

4. Load profiles

(See also: https://fr.wikipedia.org/wiki/Test_de_performance#Types_de_Tests)

P1:average

- *duration:* {{ total duration }}
- *threads:* {{ rampup duration }} rampup from 0 to [AVG_USERS], then constant {{ not required for Gatling }}
- *virtual users:* {{ rampup duration }} rampup from 0 to [AVG_USERS], then constant (1 per thread)
- *sessions:* {{ rampup duration }} rampup from 0 to [AVG_USERS], then constant (1 per virtual user)

- *throughput:* {{ rampup duration }} rampup from 0 to [AVG_RATE], then constant, through auto adjusting delay between requests

P2:peak

- *duration:* {{ total duration }}
- *threads:* {{ rampup duration }} rampup from 0 to [PEAK_USERS], then constant {{ not required for Gatling }}
- *virtual users:* {{ rampup duration }} rampup from 0 to [PEAK_USERS], then constant (1 per thread)
- *sessions:* {{ rampup duration }} rampup from 0 to [PEAK_USERS], then constant (1 per virtual user)
- *throughput:* {{ rampup duration }} rampup from 0 to [PEAK_RATE], then constant, through auto adjusting delay between requests

P3:saturation

- *duration:* {{ total duration }}
- *threads:* {{ total duration }} rampup from 0 to [SAT_USERS] {{ not required for Gatling }}
- *virtual users:* {{ total duration }} rampup from 0 to [SAT_USERS] (1 per thread)
- *sessions:* {{ total duration }} rampup from 0 to [SAT_USERS] (1 per virtual user)
- *throughput:* {{ total duration }} rampup from 0 to [SAT_RATE], through auto adjusting delay between requests

5. Configurations

C1:WebApp

{{ description of the configuration }}

- {{ physical machines }}
- {{ VM's }}
- {{ containers }}
- {{ reverse proxies, and the corresponding VM's }}
- {{ admin server(s), and the corresponding VM(s) }}
- {{ clusters (e.g. WebLo sync, WebLo async…), and the corresponding VM's }}
- {{ databases and schemas }}
- {{ … }}

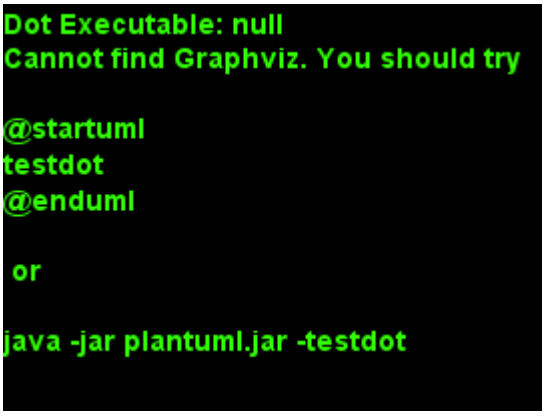
Dependencies:

- {{ base services running on ??? }}
- {{ … }}

Injection infrastructure:

- {{ physical machine, or VM ... }}

Architecture:



Cn:Other

6. Parameters

Some tests parameters should be given as part of the test requirements (typically in a mission sheet). Others need to be fine tuned as part of the tests preparation.

6.1. Given

According to mission sheet:

AVG_RATE

Average requests rate during business hours (# of requests per unit of time).
Defined as: the average requests rate during **one hour**, measured during the hour with the highest load (consider a meaningful period for the statistics : day / week / month / trimestre, depending on the periodicity of your business)

PEAK_RATE

Peak requests rate.
Same as above, but for a period of **5 minutes**

In some circumstances, the project team is not able to produce figures for [AVG_RATE] or [PEAK_RATE] (for example a new application for which there are no statistics from production).

In this case, these figures must be approximated based on asumptions from the projects team, like: max concurrent users, most loaded business hours, volume per month, etc.

SLA

performance objective should be given in the Mission Sheet, typical values are given in the below table

Client	Type	Percentage	Duration
ONSS Portal	WebApp	95%	4s
	WS	95%	4s
	SOA	95%	2s
OrliPro - Intramuros	WebApp	95%	4s
	WS	95%	4s
	SOA	95%	2s
eHealth Core	WebApp	95%	4s
	WS	98%	1s
VAS	WebApp	95%	4s
	WS	95%	1-4s (?)

— Luc Vandam, eMail sent on 2016-06-08 12:50

6.2. Calibration

Before final measures, we must determine:

AVG_USERS
: number of virtual users required to reach [AVG_RATE]

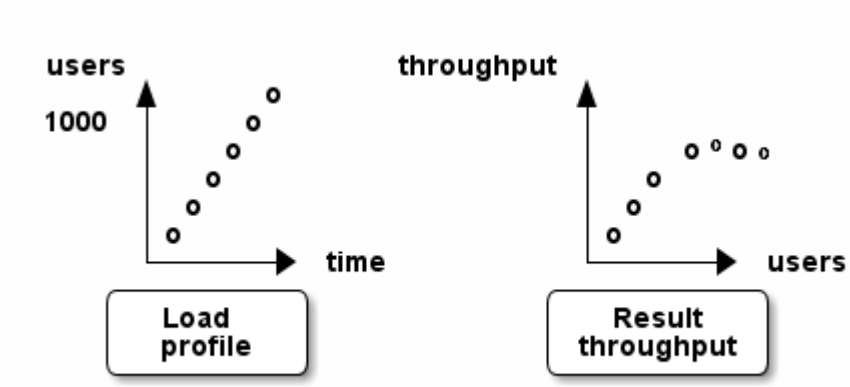
PEAK_USERS
: number of virtual users required to reach [PEAK_THROUGHPUT]

SAT_USERS
: number of virtual users required to saturate the server (i.e. such that the requests rate (a.k.a. throughput) cannot be higher)

Actual values of these parameters must be documented in the tests report.

These values may be determined by convention: use typical values and make sure the throughput setpoint is reached during the actual test. However, this gives no guarantee that the saturation point is reached for the saturation test.

Another method is to build a load profile consisting of a long rampup for the number of users, up to a very big count. This profile is injected in the system, and the resulting throughput is measured. It is then possible to determine the minimum number of virtual users required to reach a given throughput. Also there will be an inflexion point in the throughput graph, when the saturation point is reached.



7. Metrics

The following data will be recorded during each test run:

- on each node ("node1", "node2"): **vmstat** every 5 seconds, with timestamp
- for each weblogic server ("*bubble*Sync1", "*bubble*Sync2", "*bubble*Async1", "*bubble*Async2"): **GC log**
- on the injector(s): **http requests** (label, timestamp (ms), duration (ms), status code)

Based on these data, the following metrics will be calculated (for each test run):

vmstat

- run queue size over time
- CPU usage kind over time, by kind (user, kernel, io wait, stolen…)
- ratio kernel time / user time over time
- io activity over time
- context switches rate over time
- interrupts rate over time

GC log

- liveset over time (i.e. used heap size after each full (or old) GC)
- linear regression of the live set → live set slope

- ratio GC time / total time, over time
- memory allocation rate over time
- GC frequency over time, by kind (young, full/old)

http requests

for each label (and only for succesful requests)

- overall response times statistics (min, mean, median, pct90, pct95, pct98, max) during test window (i.e. rampup period excluded)
- linear regression of the response time → response time slope
- overall throughput during test window, by status (success, error)
- overall throughput statistics (mean, pct90, max)
- overall ratio error/success
- response times distribution
- dispersion ("cloud") graph, over time
- response time statistics, over time (pct90, mean, median, minimum)
- effective throughput over time
- requests concurrency level over time