Harness - Feature #6677

Performance test mode

08/16/2022 06:10 AM - Tomasz Domin

Status:	Review	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Tomasz Domin	**% Done:**	70%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

History

#1 - 08/16/2022 07:11 AM - Tomasz Domin

For some major changes introduced to the system there is a need to check how the change affects overall performance of the platform.
The check should be as similar to the real world application usage scenarios as possible.
The results should be repeatable in the given environment.
The results will be not comparable between different environments.

#2 - 08/17/2022 01:13 PM - Tomasz Domin

- Assignee set to Tomasz Domin

The basic assumption is to try to reuse existing regression test scenarios to emulate real-life application behavior when controlled by user or an external system.

- The most basic scenario is login and logout of customer application.
- The advanced test includes running full main regression scenario having failing tests excluded with a skip list

In order to avoid test scenario modification I've started to implement in harness so called performance test mode and have implemented some additional features.

performance test mode introduces following changes:

- hard limits for timeouts for tests that allow specifying a timeout (e.g. check-screen-buffer max timeout is 60s)
- for code that uses poll loops the poll frequency is increased to about 100 Hz (poll delay is 10ms) e.g. in JobQueue, so checks are done more frequently
- any delay introducing tasks like pause task are skipped completely (I had to remove some hardcoded delays from tests as well)

I also wanted to have Pre- and Post-conditions to be removed from

Additional functionality implemented (till now):

1. skip list - it is a list of names of tests to be skipped during test plan execution.
2. optimized check-screen-buffer to increase responsiveness - screen is checked for match every time terminal receives new data from application
3. in debug mode (-d)expected screen contents is included in test report in case test failed for check-screen-buffer

Additional functionality to be implemented:

- as test include some Java code to introduce random delays I need to extend pause element to allow specifying range for random delay to replace the code (5-6 places), now all tests need to have all java code for delays removed in performance test mode

The conversion of regression test scenario into performance test scenario includes two steps:

1. turning on performance test mode in Harness
2. executing test scenarios to find out failing tests to be able to exclude them in future runs (by placing on ignore list)
3. if there are no failing test in the scenario we can execute the scenario again to get execution times and compare it with execution times of changed application later.

#3 - 08/17/2022 01:13 PM - Tomasz Domin

- Status changed from New to WIP

#4 - 08/17/2022 02:12 PM - Greg Shah

What about tests which depend upon a database reset to occur before the tests run?

#5 - 08/18/2022 01:30 AM - Tomasz Domin

Greg Shah wrote:

> What about tests which depend upon a database reset to occur before the tests run?

I consider database reset as a pre-condition which needs to done before each run by a script.
It could be included as a task in test plan (as every other script) as a pre-condition.

#6 - 08/18/2022 05:45 AM - Greg Shah

Good.

#7 - 08/18/2022 11:19 AM - Tomasz Domin

Committed revision 23 with following changes:

- Implement test skip list feature
- Logging tweaks
- Correct value for PRE_FAILED result code
- Implement performance test mode
- Log expected screen in addition failing screen to in case test fails (only if debug mode is enabled)
- Allow to check detailed log levels
- Experimental removal of wait loop from screen polling implemented ONLY for performance test mode

The last item is probably the 6th or 7th version of the code, still have some issues with it, so I've put enabled it only if performance test mode is enabled by providing alternate version of crucial functions (to keep diff happy).

Working on getting of a list of tests to skip for main-regression for Java 8 - After 10 runs there are still failing tests.

#8 - 08/18/2022 05:01 PM - Tomasz Domin

Committed revision 24.

- Screen update processing optimization, time unit correction for waitForTimeout
- Fix for performance mode bug

For now the main-regression based performance tests scenario consists of 488 tests (359 are skipped) taking care of many different areas of customer app and system and execution takes about 11 minutes to complete.

#9 - 08/18/2022 05:01 PM - Tomasz Domin

- % Done changed from 0 to 70

#10 - 08/19/2022 10:53 AM - Greg Shah

Code Review Revisions 23 and 24

Overall, these are very good changes.

1. I wonder if the PM versions of methods could be implemented with less duplicated code. Perhaps the core processing could be passed in via lambda and the majority of the logic would be the same? At least some of these would work in that model.

2. LogHelper was envisioned as a generic set of logging helpers that could be used outside of the Harness project. We have a version of it in FWD too, and I would eventually like to move some common code out of these projects into a kind of utility project. Moving Harness.completionMsg() into LogHelper seems too specific to the Harness project. Perhaps it should be a public static in Harness?

3. Please add javadoc for the new data members in Harness.

#11 - 08/19/2022 12:05 PM - Tomasz Domin

Greg Shah wrote:

> Code Review Revisions 23 and 24
>
> Overall, these are very good changes.
>
> 1. I wonder if the PM versions of methods could be implemented with less duplicated code. Perhaps the core processing could be passed in via lambda and the majority of the logic would be the same? At least some of these would work in that model.

My goal is replacing normal methods with PM versions if possible. I dont know if it will be feasible as we need to keep testing results stable between versions and PM changes may change harness behavior, which must be avoided.
For now as PM methods are currently under heavy testing thus I wanted to have them completely separated to make sure I dont change regression testing results by introducing new bugs or changing the way harness work for normal regression tests.
Once PM methods are tested and work correctly they would replace old ones or would be merged with old ones to get less duplicated code. As a side effect the changes to PM mode are better visible in source code history.

> 2. LogHelper was envisioned as a generic set of logging helpers that could be used outside of the Harness project. We have a version of it in FWD too, and I would eventually like to move some common code out of these projects into a kind of utility project. Moving Harness.completionMsg() into LogHelper seems too specific to the Harness project. Perhaps it should be a public static in Harness?

Yes, this would be better, I will move it there.

> 3. Please add javadoc for the new data members in Harness.

Ok, Thank you Greg.

I am testing now PM version of send-special-key taking care of drain to minimize thread waits related with feed to keyboard input.
And I am running tens of tests all the time to find a working subset of regression tests that can be converted to (kind of) performance tests (in performance test mode)

#12 - 08/19/2022 12:19 PM - Greg Shah

> Once PM methods are tested and work correctly they would replace old ones or would be merged with old ones to get less duplicated code. As a side effect the changes to PM mode are better visible in source code history.

OK, it is a good plan.

#13 - 08/19/2022 12:23 PM - Eric Faulhaber

Sorry I'm late to the party, but I have a basic question: it seems one of the big challenges in re-purposing the existing regression tests for performance measurement is the user think or other wait time we hard-code into the tests. An extreme example would be the time clock tests which have to wait for hours to simulate time-to-lunch or shift changes before the next clock punch occurs. Is the idea to identify and bypass all such tests with the skip-list feature?

In various entries above, I see:

> hard limits for timeouts for tests that allow specifying a timeout (e.g. check-screen-buffer max timeout is 60s)

> for code that uses poll loops the poll frequency is increased to about 100 Hz (poll delay is 10ms) e.g. in JobQueue, so checks are done more frequently

> optimized check-screen-buffer to increase responsiveness - screen is checked for match every time terminal receives new data from application

> Experimental removal of wait loop from screen polling implemented ONLY for performance test mode

I guess these are all about eliminating the waiting we are doing to check for screen changes, since this waiting easily could mask any improvement or degradation in performance?

#14 - 08/19/2022 01:31 PM - Tomasz Domin

Eric Faulhaber wrote:

> Sorry I'm late to the party, but I have a basic question: it seems one of the big challenges in re-purposing the existing regression tests for performance measurement is the user think or other wait time we hard-code into the tests. An extreme example would be the time clock tests which have to wait for hours to simulate time-to-lunch or shift changes before the next clock punch occurs. Is the idea to identify and bypass all such tests with the skip-list feature?

Correct. Test that do not stress system or do not fit into time constraints (mainly because of waits) would be removed from test set by adding its name to a skip-list.
The assumption is there will be enough tests left to check overall system performance. For now there is about 400 test cases (about 50%) working correctly in performance mode, I guess this would be enough to cover most of system/platform functions and get overall view of performance changes over time.
skip-list may be also used to support working state of a system at given time - as some tests may be temporarily not working due to recent changes and this would be a way to avoid test set editing/modification.
And additionally skip-list would help having only single/global test plan for different purposes/customer, as sub-plans can be created only by editing the list.

> I guess these are all about eliminating the waiting we are doing to check for screen changes, since this waiting easily could mask any improvement or degradation in performance?

Exactly.
This is not a perfect way, but I hope it may save time and effort to build performance test scenarios manually (for now - later new tests may be introduced).
The first goal is to compare system performance for Java8 vs Java11.

#15 - 08/22/2022 11:45 AM - Tomasz Domin

I had a problem that all tests (all TestSet-s) were executed in parallel which generated a very high load and (which was even worse due to single tests still failing) failure of a single test made all results invalid.
I've modified harness in performance test mode to execute all TestSet-s in sequence instead of doing that in parallel (which is default).
So now working TestSet results cannot be broken by other failed TestSet if there is no logical level dependency between them (e.g. by database data or files)
This also allows controlling maximum number threads executed in parallel by a number of threads specified for a TestSet

#16 - 08/23/2022 12:22 PM - Tomasz Domin

Tomasz Domin wrote:

> I had a problem that all tests (all TestSet-s) were executed in parallel which generated a very high load and (which was even worse due to single tests still failing) failure of a single test made all results invalid.
> I've modified harness in performance test mode to execute all TestSet-s in sequence instead of doing that in parallel (which is default).
> So now working TestSet results cannot be broken by other failed TestSet if there is no logical level dependency between them (e.g. by database data or files)
> This also allows controlling maximum number threads executed in parallel by a number of threads specified for a TestSet

There was a problem that some tests duration was different depending on other tests running in parallel and database load. The same test sometimes was executing 30 seconds, and the other time over 100 seconds.
To manage such situation my idea is to use smaller timeout value to initially filter out tests by execution time, and bigger value when executing actual test plan, so tests that execution time is just below filtering timeout will not (should not) fail randomly during test plan execution.
I've allowed to configure cut-off timeout for "-m" option, by default its 60 seconds.

Committed revision 26.

#17 - 08/23/2022 01:49 PM - Greg Shah

Code Review Revisions 25, 26

The changes are good.

Minor code formatting/standards issues: Terminal.resetDrain() needs javadoc and both Terminal.waitForDrain() and Terminal.resetDrain() should have throws InterruptedException on a separate line.

#18 - 02/01/2024 06:48 AM - Tomasz Domin

To sum up all changes in order to support Performance Test/Benchmark mode in Harness:

1. support for test skip list - a list of tests from the test suite, in order to run in Benchmark mode unrelated tests can be masked
2. Harness.isBenchmarkMode() is public so test scenarios can check it in run-code sections to prevent introduction of delays when running in Benchmark mode
3. all waits and delays are minimized, that could cause higher CPU load
4. TestSet-s are executed in sequence
5. PAUSE scenario elements are ignored/skipped
6. Text/Key output in sendKey and send-text is done with a minimal fixed delay (10ms fixed)
7. all screen changes in wait-for-screen scenario elements are processes instantly

Test skip list is needed to:

- exclude test that have unneeded pausing inside
- exclude tests that execute too long (but it relative what that means).
- exclude implicit or explicit dependencies for above tests

In order to run Harness in Benchmark mode following command line parameters needs to be provided:

- -m - to enable Benchmark mode
- -s skipTestFileName.txt - optionally in case some tests need to be excluded from running

#19 - 02/01/2024 07:15 AM - Greg Shah

Please add this useful information to Running the Harness.

#20 - 02/02/2024 07:42 AM - Tomasz Domin

Greg Shah wrote:

> Please add this useful information to Running the Harness.

Updated by adding Benchmark-mode-Performance-test-support-mode.

In addition I've committed revision 31 which contains updates to Benchmark mode, please review.

#21 - 02/22/2024 02:25 AM - Tomasz Domin

- Status changed from WIP to Review