DGP's software development methodology is heavily dependent on frequent testing to verify that the API methods are able to meet all of the requirements that have been defined for a system. For this reason, all three of the major types of testing (unit testing, integration testing and end-to-end testing) must be designed and built into a system, and also need to be augmented by various types of client application UI testing as well. The testing tools must also provide the capability to create synthetic workloads for long periods of time in order to measure the efficiency of the deliverables under realistic workloads.

Requirements

1. Unit tests are permanently built into each API method: Depends on message-based APIs

What: *DGP's API methods are built to include unit test functionality, so they become end-to-end unit tests for every API method in a system every time a method is called.*

Why: *In order to improve the quality of software systems, testing must be built into its normal operation so that the system is constantly being tested for correct functionality and correct data as it is being used. These unit tests are also necessary for the functionality of the API Tester test harness application.*

Testing: *The API response message contains a collection of method result info for each method called in the API request message (batch) and each result returned by the API method called (batch). The method result info includes the results of the unit test of each API method called.*

2. API performance monitoring is built into each API method: Depends on message-based APIs

What: *DGP's API methods are built to include server-side performance measurements, so they monitor the end-to-end performance of every API method in a system as it is being used, and display those measurements to the user.*

Why: *The single best metric to collect is end-to-end performance in a software system. A useful subset of that measurement is the end-to-end performance of each API method on the server. Delivering very high performance requires simplicity in both the architecture and code of a system. Poor performance is an indication of many types of problems.*

Testing: *The API response message contains a collection of method result info for each method called in the API request message (batch) and each result returned by the API method called (batch). The method result info includes server-side performance data measuring the duration of each method's execution.*

3. API test harness for end-to-end full regression testing: Depends on message-based APIs

What: *A native client application that runs end-to-end tests of every API method in a system, and measures the end-to-end performance of each API method call. The API test harness can run tests of individual methods ad hoc or all tests for all API methods in a system (full regression).*

Why: *The API test harness runs end-to-end unit tests which verify the correct functionality of each API method in each environment, and also measures the end-to-end performance of each API method as well. These tests are run in the same way that client applications would call the methods during their actual use, with full network security, host server security, account authentication, authorization, etc.*

Testing: *The functionality of the API test harness can itself be tested and verified by stepping through its code and analyzing the results of the API method calls in the UI and the associated databases. Once methods are adequately tested and debugged, the immutable append-only conventions ensure that the methods and their corresponding tests require little or no work to maintain them going forward.*

4. Client application monitoring: Depends on message-based APIs

What: *Testing built into the client applications is another form of end-to-end testing that is run constantly in every environment, while the UI displays the test results to the user.*

Why: *In addition to displaying the results of each API call response in the application UI, DGP client applications also display the result info returned in each API response message such as the unit test results and the performance measurements. This helps users understand the cause of any issues more clearly, as they also become accustomed to the* *normal level of high performance for a system, which helps them notice any decrease in performance. A decrease in performance is often the only sign of gray failures in a distributed system.*

Testing: *Verifying the correctness of the unit tests and performance measurements are accomplished by stepping through the open-source code as it is being executed.*

5. API test harness for end-to-end load testing: Depends on message-based APIs

What: *A native client application that runs end-to-end tests of every API method in a system, and measures the end-to-end performance of each API method call. The API test harness can run tests of individual methods or all tests for all API methods in a system (full regression). The load test option allows the selected test files to be run continuously for a set number of iterations, which helps to create a synthetic workload for API load testing.*

Why: *Load testing is very important to help measure the efficiency of a system, which directly affects both performance and scalability, and can only truly be measured under realistic workloads. The objective of high efficiency is to ensure that memory usage does not increase over time (behaving like a memory leak), and performance does not slow down over time under prolonged high workloads.*

Testing: *The functionality of the API test harness can itself be tested and verified by stepping through its code and analyzing the results of the API method calls in the UI and the associated databases. Once methods are adequately tested and debugged, the immutable append-only conventions ensure that the methods and their corresponding tests require little or no work to maintain them going forward.*