AWS Approach for Testing a Distributed System

Ethan J. Nephew

Scenario

The scenario that will be broadly addressed is how to test a massively multiplayer online game (wikipedia, 2021). The MMO will need to be accessible over different geographical locations. Due to the decentralization of the user-base a distributed system is fitting for this scenario (Gibb, 2019). Considering the international nature of the example entity, it must comply with regional laws and regulations. Customer service would be email oriented. It is highly probable that the customer base will be diverse in spoken language, so using text translation services will be beneficial in breaking down language barriers. From a user standpoint, the specifications of the MMO/game will remain lenient.

The users will connect to the game through geographical servers. Geographical servers will be strategically located based on the anticipated size of the user-base. The anticipated size would likely be discovered through market research. The user portals would connect to their region-specific server in order to send and receive necessary system specific information. Each region server will connect to what could be described as a centralized server system that will facilitate the management, transfer, and storage of global variables.

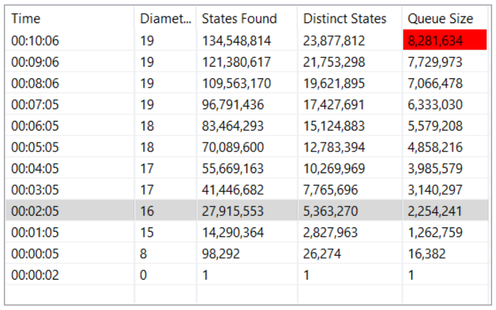
In this context, what is a server? In the wider sense a server could mean a common locality of devices that host services. Each server is not a singular machine. For example, the Eastern South American Server could consist of dozens of different computers, and each will be running a single service that is providing data to multiple users and/or to other services on other machines. To the user, it will appear as though all services are operating as a singular entity. An approach to determining what type of service should be run by a specific service machine and where the location of that service machine needs to be is by latency requirements. For a service that requires faster results, it will make sense for that service to operate on a machine that is geographically closer to its targeted user cluster(s).

Emphasis on Testing

At the beginning of development, testing and the development of test infrastructure should be taking place. Regular industry testing practices will take place, such as development testing and quality assurance testing. This isn't to say that the development process needs to be strictly test-driven, but rather testing needs to be part of the process from the beginning. Developers should be writing thorough unit tests and integration tests. Additionally, it is important to have assertion statements embedded inside the code. Assertion statements can be enforced by running code with the assertions on (stackoverflow, 2019).

Specification Abstraction as a Concept Test

During design, a misstep is a lack of specifications for a distributed system. While testing should take place at the code level, it should also take place on the design level itself. Some problems that distributed systems face are obscured by details that can vary greatly in relevance regarding solution development. During the engineering process, jumping into the minefield of details can be a strong temptation, but the minefield can distract or confuse us of the overall design of a system (Lamport, Teaching Concurrency, 2009).



Time	Diamet...	States Found	Distinct States	Queue Size
00:10:06	19	134,548,814	23,877,812	8,281,634
00:09:06	19	121,380,617	21,753,298	7,729,973
00:08:06	19	109,563,170	19,621,895	7,066,478
00:07:05	19	96,791,436	17,427,691	6,333,030
00:06:05	18	83,464,293	15,124,883	5,579,208
00:05:05	18	70,089,600	12,783,394	4,858,216
00:04:05	17	55,669,163	10,269,969	3,985,579
00:03:05	17	41,446,682	7,765,696	3,140,297
00:02:05	16	27,915,553	5,363,270	2,254,241
00:01:05	15	14,290,364	2,827,963	1,262,759
00:00:05	8	98,292	26,274	16,382
00:00:02	0	1	1	1

Figure 1 A simple model produced in TLA+ that shows the high degree of distinct states that can occur during a simple username authentication process during concurrent operation. A minor bug can have a major manifestation. Note: The model was ended before completion.

It is also easy to feel that if the implementation is not being directly addressed, then time is being wasted. However, we will always struggle to implement systems that we find difficult to understand at a higher level. Understanding the problem is addressing the problem. Learning how to approach gaining comprehension of a system is a difficult task, because it can appear anti-thetically to our learned development processes. At face value, it should be easy to understand how a detail-oriented approach to grand problems can result in the inclusion of unnecessary details. One tool that can be used to achieve a greater understanding of systems is TLA+ (Lamport, Industrial Use of TLA+, 2018).

In Figure 1, it is worth pondering what it would take for every state that was reached using the TLA+ Toolbox for 10 minutes to instead be reached during development, a test scenario, or product release. In traditional development, there are a potentially infinite number of unique states that might never be anticipated or reached. Formal methods are about getting developers to begin thinking about those unique and more difficult to reach states. TLA+ can be used to convert a programming language algorithm into a mathematical expression (wikipedia, 2021). In the TLA+ Toolbox, invariants can be specified in the Model Overview and the model can be run using TLC. If an invariant is violated, then the distinct states that lead to that violation will be displayed. Testing all execution paths is something that the TLA+ Toolbox can do.

Invariant Identification and Assertion

In software systems we usually test specific things. Unit tests will generally test that method(s) produce the correct outputs, given the specified inputs. In distributed systems, something that can be difficult to identify without the use of TLA+ is discovering different types of invariants (Hochstein, 2018). Invariants, generally, are things that should be true at a specific state or all states. A state can be understood as the facts surrounding a particular area after an execution. When we introduce concurrency to a program, even simple things can easily become a seemingly bottomless pit of unique states. The identified invariants should have their properties be asserted as true in key locations in the non-critical program execution path.

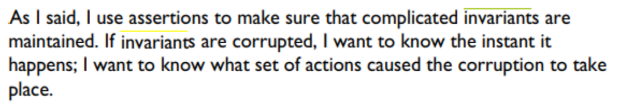

As I said, I use assertions to make sure that complicated invariants are maintained. If invariants are corrupted, I want to know the instant it happens; I want to know what set of actions caused the corruption to take place.

Figure 2 Quote from Joshua Bloch in Coders at Work (Seibel, 2009) from page 190. Invariants tend to have a negative connotation, but this could be a misplaced sentiment. Invariants are bad if they remain unidentified.

Test Case Generation

While engaging in property testing, the focus should move away from specific test cases and towards the testing of properties. Properties can be tested thoroughly by using generative testing (Normington, 2019). Generative testing involves the creation of a test case generator. The generator needs to have parameters possible parameters specified. The generator then creates the tests. A useful aspect of generative testing is that implementing random test cases during generation is feasible.

Integration Testing of Distributed Algorithms

There must be a way to discover bugs that are revealed during stressful or semi-stressful circumstances, such as multiple processes being operated simultaneously. Using what Tim Rath refers to as ‘in process clusters’ (Rath, 2015), can be an effective method for discovering the described bug. IBM has a cluster testing tool that has a useful overview that can help gain a broad idea of what cluster testing looks like.

The goal of in process clusters is to have the ability to isolate certain clusters of a process and test those specific clusters. This allows for the creation of test scenarios that would otherwise be very difficult to create.

Downsides

Some of the significant downsides to this approach is finding qualified individuals. TLA+ is not a popular technology, so education will likely be required. TLA+ is not particularly friendly towards new users. However, there are some useful tutorials made by Leslie Lamport (Lamport, Learning TLA+, 2019).

References

Gibb, R. (2019, July 26). *What is a Distributed System?* Retrieved from stackpath.com: https://blog.stackpath.com/distributed-system/

Hochstein, L. (2018, December 27). *https://surfingcomplexity.blog/2018/12/27/inductive-invariants/*. Retrieved from surfingcomplexity.blog: https://surfingcomplexity.blog/2018/12/27/inductive-invariants/

Lamport, L. (2009, November 30). *Teaching Concurrency*. Retrieved from microsoft.com: https://www.microsoft.com/en-us/research/uploads/prod/2016/12/Teaching-Concurrency.pdf

Lamport, L. (2018, December 4). *Industrial Use of TLA+*. Retrieved from lamport.azurewebsites.net: https://lamport.azurewebsites.net/tla/industrial-use.html

Lamport, L. (2019, August 19). *Learning TLA+*. Retrieved from lamport.azurewebsites.net: https://lamport.azurewebsites.net/tla/learning.html

Normington, J. (2019, January 18). *A smarter way to QA: introducing generative testing*. Retrieved from medium.com: https://medium.com/geckoboard-under-the-hood/how-generative-testing-changed-the-way-we-qa-geckoboard-b4a48a193449

Rath, T. (2015, February 21). *The Evolution of Testing Methodology at AWS: From Status Quo to Formal Methods with TLA+*. Retrieved from infoq.com: https://www.infoq.com/presentations/aws-testing-tla/

Seibel, P. (2009). Coders at Work: Reflections on the Craft of Programming. In P. Seibel. Apress. Retrieved from https://github.com/ndina/acm/blob/master/coders-at-work.pdf

stackoverflow. (2019). *How to enable the Java keyword assert in Eclipse program-wise?* Retrieved from stackoverflow.com: https://stackoverflow.com/questions/11415160/how-to-enable-the-java-keyword-assert-in-eclipse-program-wise

wikipedia. (2021, November 7). *Massively multiplayer online game*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/Massively_multiplayer_online_game

wikipedia. (2021, September 16). *TLA+*. Retrieved from wikipedia.org: https://en.wikipedia.org/wiki/TLA%2B#:~:text=TLA%2B%20is%20a%20formal%20specification,concurrent%20systems%20and%20distributed%20systems.&text=TLA%2B%20is%20also%20used%20to,for%20algorithms%20and%20mathematical%20theorems.