Project Ruby Acorn - Autoscaling a gaming service

Ruby Acorn is a new gaming company with many exciting and popular online games.

They are using a cloud-based infrastructure to host their game servers, which is working well enough provided they have enough server instances per game. If the number of servers instances is too low, the performance suffers and players have to wait in lobbies, which detracts from their overall gaming experience. Having many server instances per game solves this issue, but will drive the cost up, as every server instance costs money based on the time it is up.

The most optimal solution would be to adjust the number of instances relative to the number of players. Two variations have already been considered, and subsequently dismissed:

1. Dedicate engineers to manually supervise and adjust the number of instances. As this would very like provide a high correspondence between the needed and actual number of instances, it would more likely than not drive the cost up even more. Engineers are expensive, and their time could be put to better use.

1. Adjusting the number of instances based on the time of day. This does not work because of unforseen changes in the number of players based on events outside of the control. For example, when a YouTuber starts playing an older game it always brings a wave of players back to it for a short time.

One remaining option has not been investigated yet: Automaticall scaling the number of instances based on the current number of players. This would be a more reactive behavior compared to the two other alternatives. It would mean that there is a piece of software wich investigates the number of players for a game at regular intervals and makes a decision on wether or not to scale up or down the number of instances.

The CIO at Ruby Acorns is unsure how well this will work, but as agreed to let a pilot project investigate it further in order to help answer a few questions:

1. How much money could be saved compared to a fixed number of instances?
2. How well would such a solution react to any changes in player numbers?
3. What would be a good way to measure a "successful 24 hours" using automated scaling?

There are several technical milestones that this pilot needs to achieve:

First of all, there is currently no dashboard or other surveillance system which can be used to proberly monitor the number of players for the games. This needs to be in place, so that one can observe the pilot system in real time. The head of engineering

suggests trying Prometheus and Grafana. The data is already available, so one only needs to import it.

Secondly, the decision engine, meaning the software that evaluates the number of players and adjusts the instances accordingly needs to be developed. It is suggested to use Python here and the software should reside in a Git repository. The actual algorithm is not decided and will probably need to be adjusted and fine-tuned. Let's start with something simple, like dividing the number of current players on the capacity each instance has and use that to determine the number of instances needed. More advanced variations can be tried as well, but first after this one has been shown to work. More advanced examples are not required, but a plus.

The decision engine should get its data from somewhere too, either from Prometheus, or directly from the source. Getting the data from Prometheus has the benefit of running more advanced queries, which could be useful for more advanced scaling decisions.

Thirdly, there is no way that this will be tested in production even though it uses data from the production systems. Not just because of whatever damage any bugs in the algorithm might cause, but also becuase they do not want to pay the cost of spawning houndreds of unused server instances just for testing. Instead, the head of engineering suggests scaling the number of Docker instances on a couple of servers. The number of Docker containers per game can be monitored as well and compared to the number of players and cost.

Finally, allthough Ruby Acorns has many online games in production at the moment, the pilot does not have to be tested on all of them. However, it would be nice to see it being used on different *types* of games which have different usage profiles. Some games have a more volatile number of players than others.

A report should detail the pilot project and the CIO makes it clear that whoever is in charge of the pilot should not assume to be an advocate for the idea. The decision to ultimately use something like this or similar rests on the CIO and head of engineering. What they require is an unbiased assessment containing critical reflections which helps them in their deliberation. One is free to make reccommendations, but every *opinion* in the report should be grounded in the material and discussed. The report should also detail the architecture of the entire solution and showcase it's workings. If more variations of the decision algorithms have not been technically explored, then they should at least be discussed in writing, as it is already assumed that the most basic starting algorithm will have a few flaws.

Technical notes

It is assumed that one uses the ALTO cloud for this project. The game server instances do not have to run any particular software at all, they just need to run

The metrics data target can be found here:

acit-game-metrics.cs.oslomet.no/metrics

This data is "Live" in that it is changeing every five minutes. It represents the top 1000 most popular games in 2017 and the low 1000 least popular games (in terms of the number og players) in 2017.

The data source contains many games and it is not assumed that one needs to test in on all. You are free to pick out the game(s) which the decision engine will be considering.