Ideas

1 Questions and Answers

– in that paper for which you reproduced the results, the idea is that executing tests in cloud environments is trivial and there are infinite resources available. However, doing it on embedded devices needs careful infrastructure pre-planning to optimize investments and costs and that's what the model is useful for.

– However, in reality many companies end up paying quite some money for various SaaS and cloud bills. There are start-ups that try to help keeping track of such expenses: https://www.cleanshelf.com So the question is: can we use the model to predict cloud infrastructure costs for various developer team sizes and test execution times? For instance consider on-demand instance pricing: https://aws.amazon.com/ec2/pricing/on-demand/ What are the costs on a regular month, what about when one has a deadline? Infrastructure cost vs human time cost tradeoff.

– Even broader - how many optimization dimensions do we have along which we could analyze: team size, test execution time, [Would you be able to make a list with them?]? The discussion section below (Sec 7) tries to do a bit of that but it's not strong enough.

1.1 Q1: What are the infrastructure costs in normal working periods vs periods with deadlines?

Something like: assuming a developer team of 20 people, average test execution time of 5 mins, average test wait time 5 mins how many units of cost U would a company spend per month to ensure various GoS 90, 95, 98, 100 %? (let's first talk about units of costs in an abstract way, then map to various on-demand instance costs).

Answer: First I calculated a number of servers that are required to achieve the desired grade of service. I calculated them for two different cases: in case there is a deadline and in case there is no deadline. In the case where there is no deadline (Figure 1), I assumed that each developer would make commits 3 times a day. In case there is a deadline (Figure 2), I assumed that each developer commits 9 times per day. I calculated the average number of hours per day that the servers are used. In our model it is about 10 hours per day. I don't think that this is very relevant, because we simulate the whole process with the same parameters, which in reality only apply to a peak hour.

	1 deadline/week	2 deadlines/week	3 deadlines/week
GoS 90%	680	760	840
GoS 95%	680	760	840
GoS 98%	880	960	1040
GoS 100%	3280	3560	3840

Table 1: Let the cost of servers be 1 U/hour. In the table we have monthly costs in U for 20 working days

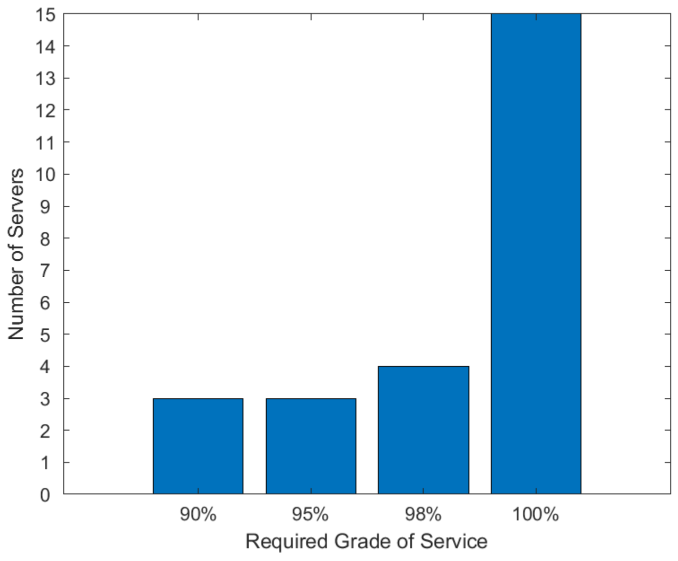

Figure 1: Number of required servers in order to achieve desired grade of service.

1.2 Q2: What are the infrastructure costs for various teams of various sizes?

Along the lines we discussed that for 300 devs you need N servers, but for 10x30 devs you likely need less than N servers all other params being the same.
Answer: Parameters used in Figure 3:

- number of developers $k = 300$
- average testing time $h = 3$ minutes
- acceptable waiting time $w = 10$ minutes
- number of commits per day per developer $r = 3$
- percent of commits in a peak hour $p = 0.1$

For $N = 6$, the grade of service is 99.72%. If we devide the developers into 10 teams of 30 developers each, we have several different options:

1. All teams share the servers: In this case we can easily see that the situation is exactly the same as before, because there are 300 individual developers using N servers, just as before. The teams play no role here.

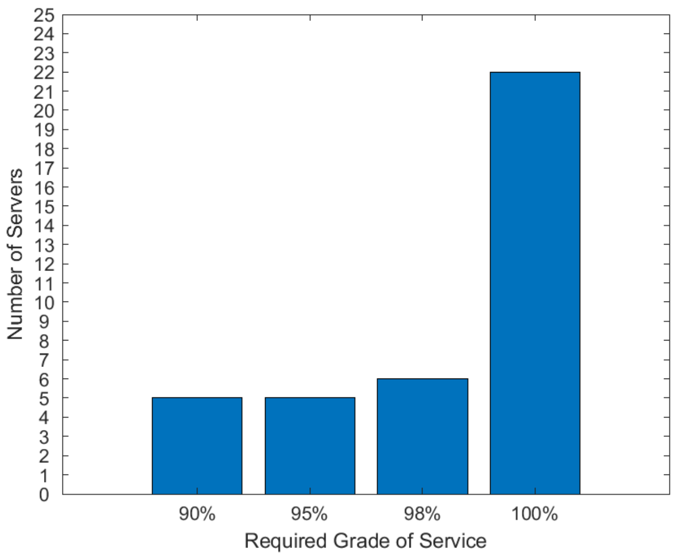

Figure 2: Number of required servers in order to achieve desired grade of service.

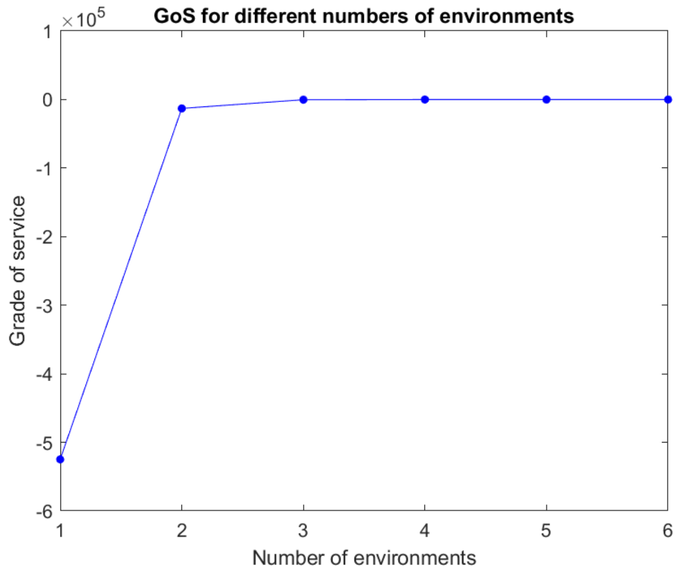

Figure 3: Grades of service for different N and 300 developers in one team

2. Each team has its own servers: In Figure 4 we see that for grade of service greater than 99%, a team of 30 developers needs 2 servers. If each team has its own servers and doesn't share them with another team, then we need 20 servers, which is much more than in the first case.

3. It is also possible for a team to have a few servers to itself and share some servers with others. If we devide 300 developers into 10 teams of

30 developers each, we can immediately see that we are not getting the optimal solution. Each team needs at least 1 server for itself, so we already need 10 servers and don't even have the required grade of service.

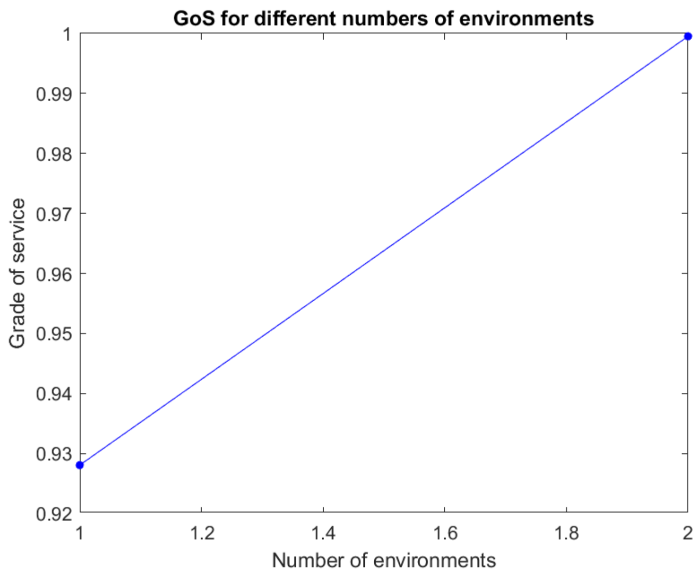

Figure 4: Grades of service for different N and 30 developers in one team

These results are actually quite to be expected. If each team has servers for itself and doesn't have to share them with another team, it can happen that in some teams some servers are free and in other teams there is a long queue of commits waiting to be tested.
There is another related interesting question: if one of the development teams has much longer average testing time, is it better for this team to have its own servers, or is it better to share them with other developers? I have simulated this situation. There are 200 developers who share the servers, and it takes an average of 3 minutes to run tests for them. For the required grade of service of 95% we need 4 servers.
We also have another team of 100 developers, whose tests take 10 minutes on average. For the required grade of service of 95% we need 7 servers. The acceptable waiting time is 10 minutes for both teams.

I simulated this situation and got some results. However, I don't quite know how to visualise them. I got similar results as before: for required GoS we (on average) need less servers when everybody is sharing them.

1.3 Q3: How does the way the tests are engineered affect the development process?

How does the way the tests are engineered affect the development process in terms of waiting times, in terms of cost units for infrastructure, in terms of cost units for people waiting?

1.4 Optimization dimensions:

- team size
- execution time
- number of environments (required GoS)
- deadlines/no deadlines