The DOM pizza checker is meant to give better quality pizzas to customers. Stakeholders in this case would be:

- The customers: They ask for the pizza of their liking and get photo confirmations to assure the quality of the pizza is the same in real life as it is advertised. Some technical questions can rise, like, if at the start of implementation the pizzas will take longer to make thus perceiving a decrease in service quality. Another more important issue for this business is whether the model will identify certain characteristics as "important" through user feedback. It would be interesting to see that the model gets trained to please the majority of customers, while some other few prefer their pizza "unoptimized" by the model. Worse still, would be to see that the model gets trained and reinforced NOT by user feedback but by images or the company's belief of what the clients will like.
- The workers at the pizza places: They have to make the pizzas, the system will tell them if the pizza has to be remade, and this time it is not by customer request but by an automated system that, in the worst case, could be difficult to interpret (imagine having a pizza you made is rejected by a machine, and even though you think you have corrected the error the system still rejects it). This would lead to slower service times, and more importantly uncomfortable workers, stress, and could create a sense of "the machine's instructions are more important than my effort".
- The owners of franchises of these pizza places: What if there is an error in a DOM system in a franchise in New Zealand. First of all, this means less competitiveness with other pizza shops, second, it would mean more costs to cover for maintenance and repair. Even if there is no error, it is important to think who will be responsible for these costs, if the company is making this implementation "not voluntary" it would definitely mean imposing a possible (even if it is a small possibility) downgrade in the service for the franchise, in addition to incrementing their costs **without a certain evidence for more revenue to compensate for it**. Lastly, even if the decision of implementing this system in a franchise is voluntary, let's assume that 3 franchises do implement it and it is perceived as an upgrade in the service, and 1 franchise does not implement it. This franchise could be deemed of less quality, hurting the economy of the franchise even though maybe the service of that particular franchise is good enough..
- The company itself: in an extreme but possible case, the implementation strategy is such that the customers do not feel like they are given a better service but rather that they are being told what the best pizza is, regardless of their opinion, which would turn customers unhappy with the product.
- Finally, other pizza shops: Because if this becomes a huge success, it would implicitly mean that other pizzas are not of good quality, they do not have the right distribution/quantity of ingredients. They could receive a serious blow by being considered less good, and rendering less competitive. It is easy to see that, on the other hand, they get benefits for every point in the 3rd bullet point. (Since they are competing).

Julio Ponte Hernandez

I would start by asking the scale of the analytics and of the workloads. Firstly the data that we work with should be well organized: not scattered among a lot of databases. Also I would think how is the data collected, is it locally stored or is there a cloud to which it is being uploaded.
Very important is to know if these programs/analytics/computations require a lot of time running on one core. If the work that is being done in one core is made in a sufficiently reasonable amount of time (and there is reason to believe that this will continue in the short- to mid- future) then maybe IT could find the most benefit at the best cost by, before thinking about parallelizing, trying to optimize code, and use technology that makes the code run as fast as possible in the existing setup.For example by using a compiled language, or plain-old algorithm design. In this case we could also think about using cloud machines with a few cores, and in that case there is not so much availability concerns on this single core machine.

If the one core machine is insufficient even with code optimizations, then Amdahl's law would be useful to estimate how much of the workload can be parallelized and what the benefit can be. In order for there to be motivation to utilize parallel implementations, there has to be an estimated benefit to invest money to reduce analysis time or idle times and to take into account the programming time, and maintenance requirements. If the proportion of code that can benefit from parallelization is not enough, it is not the best option to go for investing into parallel programming implementations, many cores or an expensive cloud solution. Better to invest in better storage solutions with faster I/O, try to buy a commodity PC with better CPU in addition to the code optimizations.

If there is reason to believe that the current performance is totally under-performant and the estimations is that there will be a lot of new projects where performance will be critical, we have to think more about the type of workloads that there are / will be. Are there a lot of different workloads? For example: statistics and indicators tracking, or mathematical models and simulations, or are the workloads oriented towards machine learning: behaviour identification, classifiers, training models, and so on.

In the case of statistics and tracking of indicators (etc.), I would suggest using acyclic, or one pass solutions such as MapReduce programs. If there are math models that require a lot of arithmetic, calculus and the like, then I would suggest to implement a multiprocessing model with a distributed system, either with an interconnection network to shared storage or machines with local disks and interconnection network. If there are a lot of machine learning oriented operations, then Spark is a convenient and efficient approach, with the addition of an architecture with a lot of main-memory capacity, these AI models could be trained efficiently.

Julio Ponte Hernandez