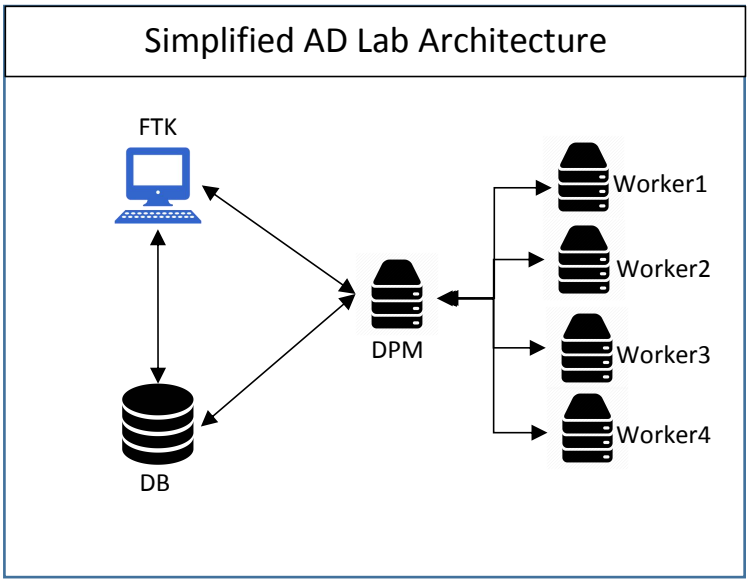
Quin-C's Architecture

Currently the most common request I get is to explain Quin-C's architecture. I have been asked it enough times that I figured it merited a blog entry of its own, so here goes. When designing Quin-C our goal was to deliver as scallable, modular, and simple a solution as possible that could run as well on a single machine as it can on a hundred machines. To achieve this we looked at two different approaches. The first was the one AD Lab uses, in which one centralize system hands out work to distributed workers. In this approach all parts of the system work together to deliver optimal utilization and speed. The approach is very attractive but it has a few major drawbacks. The first is that it is relatively complex to install and administer. To be clear it still only takes a day for a trained person to install but you do need to be trained. I personally can't do it so I consider it limits on scallability. Because a single system is dividing up the work and managing all the workers involved there is a single bottleneck Of course the work for that single point is small enough that the bottleneck is rarely if ever seen by customers but it definitely exists. Finally it really can't be installed on a single machine, which was a requirement for Quin-C. Given these drawbacks we decided this approach wouldn't work for us.



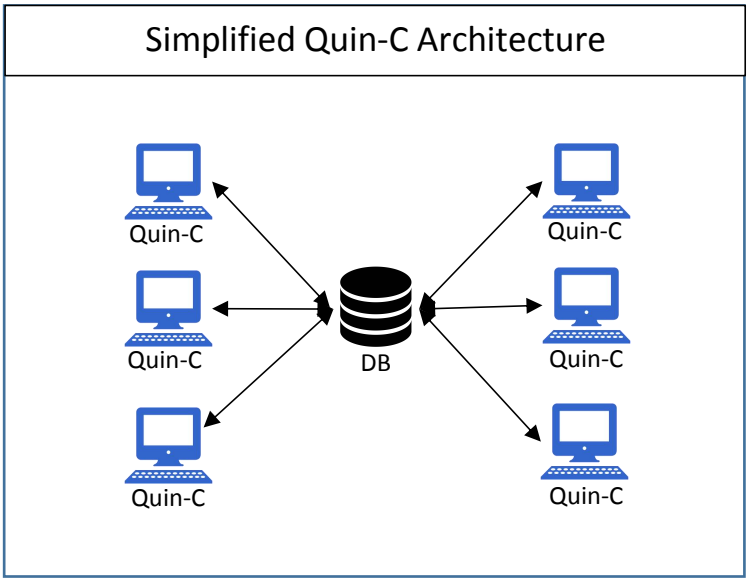
The other approach we explored and ultimately choose was a siloed base approach in which each unit of the system operates entirely autonomously. Since this is the approach we choose lets look at it in depth.

The approach has four basic key attributes.

1. The first is that each unit of the system is actually a full install of Quin-C. If you wanted to you could unplug each machine from the larger system and it would continue to work.
2. The second key attributed is that each unit of the system is entirely unaware of the other units. There is no one machine coordinating the activities of the other machines. Each machine operates entirely autonomously.
3. The third key attributed of the system is that every processing job and task is assigned to one machine and only one machine. There is no logic which machine gets the job other than first come first server.

4. The fourth and final attribute of the system is that all units point to the same database. While this goes almost without saying I am pointing it out explicitly just so there is no confusion.

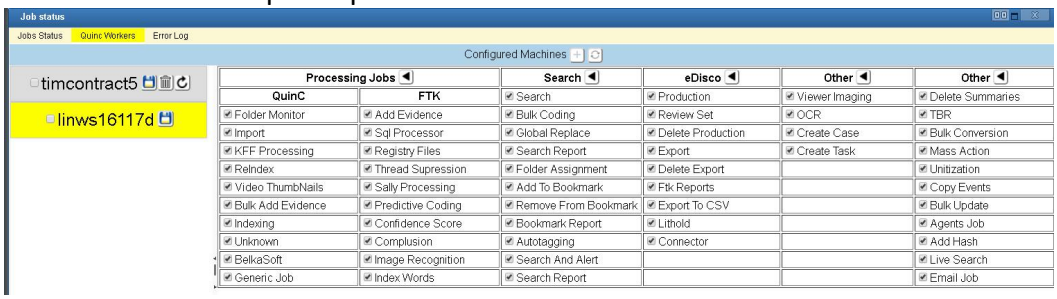
Lets discuss the implications of each of these key attributes because they are both positives and negative and definitely worth understanding. The first and second key attributes taken together mean the system is highly scallable. Frankly it scales well beyond what could be achieved with a more unified system like AD Lab. In addition, it is extremely easy to install. All that is required is to install Quin-C on each machine you want to use and then to point that Quin-C at the common DB. In addition, you can scale more than just the processing power, you can scale the review power by load balancing any number of Quin-C's in order to service as large a user population as needed. The system is highly flexible as well, in that machines can be added an removed whenever needed with no configuration or adjustments required other than installing the new machine and pointing it at the DB. However, there is a downside to this approach, which becomes apparent when we add the third key attribute. Since there is no entity that coordinates activity between the systems we needed to ensure that there were no collisions between them. We do this by implementing a first come first server processing system in which each job is viewed as single indivisible unit that is given to the first computer that asks for it. Once the job is given to a machine it cannot be given to any other machine.



So what does that all mean from a practical standpoint? Let me explain using an example. Lets image you have 50 images that you need to process and that each image is 100Gig. Using Quin-C you could quickly spin up 25 machines and set them to work. Each machine would likely end up processing 2 images and the job would finish reasonable quickly. However, lets now change that example and say you have only one image but this time that one image is 5Terabytes. In this case lets say you again spin up 25 machines. The problem here is that since you only have one image and one job to run only one machine would pick it up. The other 24 would sit idle and the work wouldn't complete until the one machine was finished. Since I think most organization have large numbers of smaller cases and processing jobs, this was a trade we thought made sense. However, if you have a few huge images, you will see far better speeds using AD Lab to process the data.

From an architectural standpoint that is the end of the blog. Hopefully how Quin-C works is now at least a little bit more clear. However, we didn't really love the issue associated with the reduced performance on small numbers of huge jobs so we have implemented a three key features that allow you to mitigate or completely remove this limitation.

1. The first key feature is that ability to inter-operate with AD Lab. Because of the way Quin-C works it can be utilize on the same system that supports AD Lab. If you already have a distributed processing system setup there is no need to abandon that. In fact you can simply add quin-c capabilities to you existing configuration and get the best of both worlds, though you do need to be careful in setup and configuration.
2. The second key feature is the ability to select jobs each Quin-C install will run. As the screenshot shows Quin-C allows you to configure by machine what jobs can run. This allows you to ensure that the best and most powerful hardware focuses on the longer running tasks while the other hardware handles the large number of small simple requests.



3. The final feature we implement was the ability to configure the time of day when a given instance of Quin-C will run and accept jobs. While this may sound unrelated to what we have been discussing it is actually very important. It means you can us Quin-C to take advantage of unused processing cycle available on machines that are normally only used during the daytime.

With these three features added to the Quin-C solution I believe we have delivered a system that can more than compensate for the one potential negative associated with its architecture. My hope is that leaves us with a system that is not only the most scalible and flexibility on the market but also one without am clear obvious drawbacks. That said I will leave you to be the ultimate judge of that. As always thanks for taking the time to read my blog and post any questions or comments you have on the message board.