**Summary of Project**

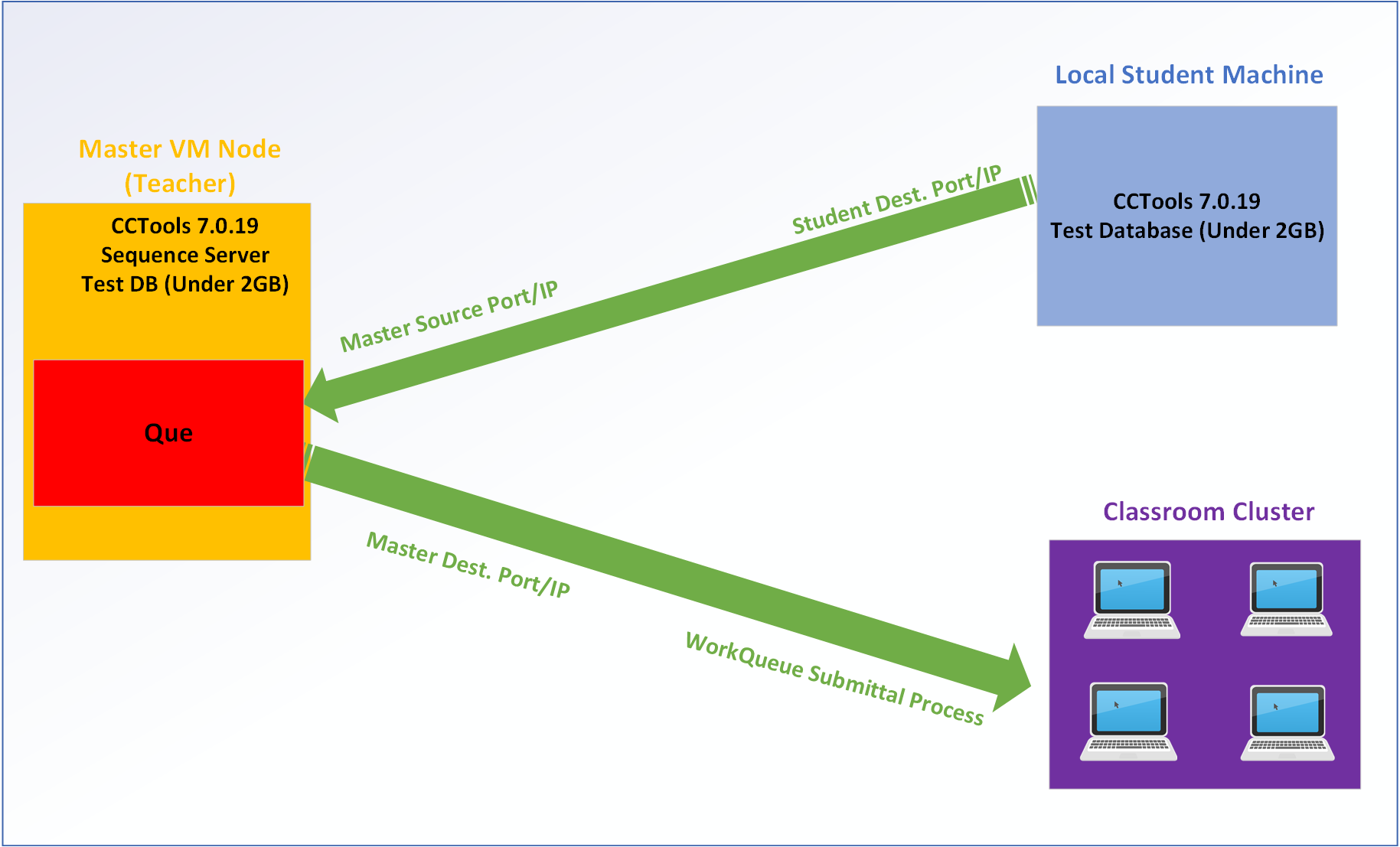
In the beginning of the semester, Wilson came to us with a problem: How can we create a scalable solution capable of supporting up to hundreds of people using SequenceServer to query NCBI Blast databases. Moreover, Wilson needed us to also include:

1. A web client which can be used by educators and students to run the NCBI BLAST Basic Local Alignment Search Tool algorithm.
2. The client would specifically like for us to provide the [wurmlab/sequenceserver](https://github.com/wurmlab/sequenceserver) local web front-end version of the BLAST tool, which delivers the user interface and analysis capabilities they require. In other words he needed us design this with the same BLAST interface the staff we’re already used to.
3. Finally the client needs the hosted blast implementation to support multiple classes of at least 100 students all using the BLAST curriculum concurrently (courtesy of [Wilson's](https://wiki.cyverse.org/wiki/download/attachments/45327818/GEA_BLAST_Project_Proposal.pptx?version=1&modificationDate=1568134287000&api=v2) slides), with an average run time of 2-3 minutes per job. As well as some assurance that the ~100 jobs will finish at approximately the same time.

Team Frankenstein aimed to create a ‘classroom cluster’ model, meaning that the teacher is the master node, and the class becomes a cluster of workers. Essentially the teacher will download a docker file containing sequenceserver onto a Cyverse master CCTools instance. Once the IP address is available and the teacher has selected a port number, this information will be shared with the students. The students download a docker work file on their local machine containing CCTools and the selected database, and then set their machine up to talk to the teachers master Cyverse VM. This process essentially creates the class as a set of workers, whom are able to submit jobs to the class.

Team Frankenstein’s solution sets the stage for a completely offline solution, assuming the students and master machine have Docker already installed. This is a route that we have not explored, but wanted to make the user aware that this is an option.

**Description of Project**



**Code Availability**

Github Repository - <https://github.com/GEABlast/frankenstein>

Full Report on Wiki - <https://wiki.cyverse.org/wiki/display/A2/Midterm+Deliverable>

**Installing and Running Instructions**

To begin this process, the master is going to have to create a Cyverse master instance, which can be a ‘tiny1’ size or bigger, although any ‘tiny1’ sized instance will suffice for this solution. Once the instance is up and running, the teacher needs to download the master docker file using the ‘wget’ command. Once the master docker file is downloaded, the professor can look at the IPv4 of the instance found on the Cyverse Launch page and share that with students via email, verbal communication, etc. The Master docker file will have downloaded SequenceServer, a test blast DB, and the CCTools 7.0.19 tools suite. It will operate over the SeqServer default port 4567, which must also be shared with the students via email, verbal communication, etc.

On the student side, they will go through a similar process that includes downloading the docker file which includes CCTools 7.0.19 and the test DB. This process will be done on the local machine, and the professor must help them set up their machine to talk on the correct port/IP address. Once the class has all been hooked up to the same Master instance, they are now able to submit BLAST queries through the master instance which has sequence server, and thus a que, that jobs will be filed through. Submitted jobs will then be distributed across the classroom, using the classroom’s local resources to accomplish jobs.

This essentially allows for a scalable solution that is as efficient as the resources on hand allow. Their are admittedly shortcomings to our solution that have potential for improvement. One aspect is that our benchmarking times in a live use case would be varied because the worker cluster efficiency relies on computational resources in the classroom. The more students we have as workers, the quicker the times and vice versa. Another shortcoming is that this solution is meant to use small databases; because we are using student local machines, synching large databases across the local wifi network may cause networking issues/failure. The last caveat is that we are assuming the students and professor can follow our directions to input the port/IP address; in other words, this is not a download and done solution, there is some user input required.

We have also included a tutorial for how to create and upload your own custom database as seen in Reetu & Friends solution, should the instructor trust the classroom bandwidth to handle it. It can be seen on the wiki [here](https://wiki.cyverse.org/wiki/display/A2/Creating+a+database+in+cyverse). This solution uses iRODS, which is what Cyverse uses to move data to/from their data storage. We have made a tutorial for how to download iRODS which can be followed here. (**INSERT LINK)**

**Team Members Contribution**

Josh Forstedt – Linked independent master and worker VMs running docker containers together, running BLASTEasy’s modified 1.0.12 Sequence Server with WorkQueue.

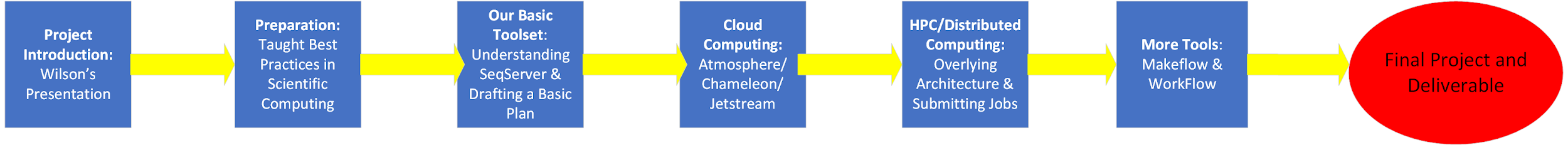
Derek Caldwell - documentation, final presentation, progress presentations in class, and attending class collaboration meetings.

Nasser Albalawi - contributed in creating VMs to test benchmarking and Creating” Master Machine”, created docker image for the master, having hands on makeflow and work queue attending class meeting, attending workshops, contributed with writing and presentation.

Asiedu Owusu-Kyereko - Contributed on making the docker container. Contributed with write-up and presentation

Special Thanks to John Xu, Sateesh Peri, and Team BLASTEasy.

**Project Timeline**



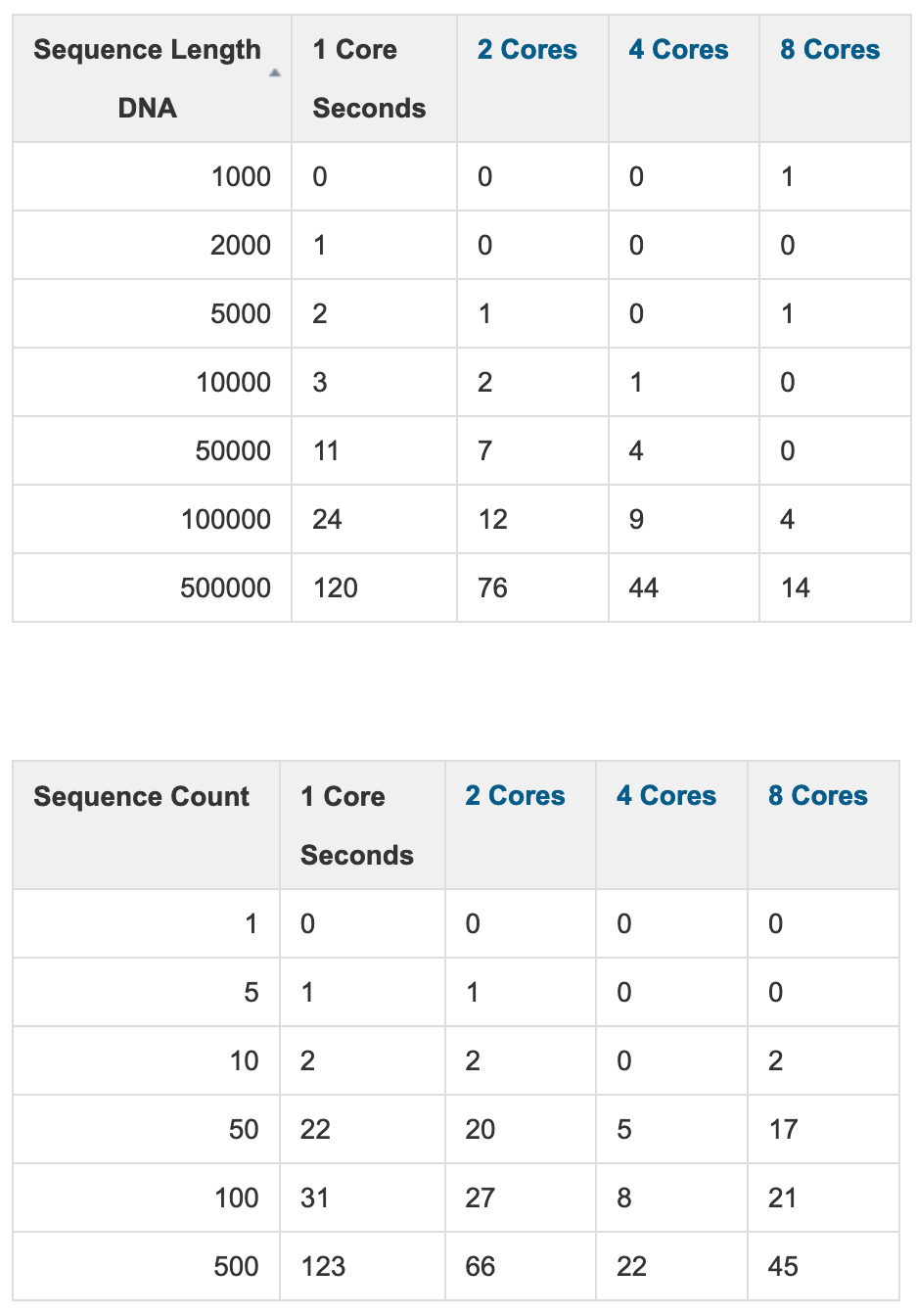
**Benchmarking**

So unfortunately we did not get an in-depth look at the benchmarking process of our project, however we anticipate it to be similar to BLASTeasy’s solution considering the overall architecture is similar. We recognize that since our solution relies on the classroom’s collective computing resources, in a live use case benchmarking times would be inconsistent; the more computers connected, the quicker benchmarking times. The other obvious difference between our solution and BLASTeasy’s is that ours is containerized.

The containerization of the classroom cluster implementation may or may not tack on time to the benchmarking process. The following results were run from our initial benchmarking using this solution with both the master and workers as VM’s. This provides a ballpark for how long these queries may take.

For benchmarking, we ran the test on four different virtual machines with four different types of cores (1, 2, 4, and 8 cores) and each machine has different size of memory and harddisk. The instance size of the four virtual machines is presented as following:

|  |  |
| --- | --- |
| Instance Name | Instance size |
| Ubuntu 18\_04 GUI XFCE Base | Tiny 2(CPU:1 Mem:8 GB, Disk:60 GB) |
| Ubuntu 18\_04 GUI XFCE Base | Small 2(CPU:2 Mem:16 GB, Disk:120 GB) |
| Ubuntu 18\_04 GUI XFCE Base | Medium 3(CPU:4 Mem:32 GB, Disk:240 GB) |
| Ubuntu 18\_04 GUI XFCE Base | Large 3(CPU:8 Mem:64 GB, Disk:480 GB) |



**Presentation**:

**Input Exported PowerPoint Presentation.** [**https://docs.google.com/presentation/d/1SOUsKjVtZrnL7GM0E0m\_KDS\_d3FSJZITMnHaoGh\_rFs/edit?pli=1#slide=id.g73ce69d719\_2\_5**](https://docs.google.com/presentation/d/1SOUsKjVtZrnL7GM0E0m_KDS_d3FSJZITMnHaoGh_rFs/edit?pli=1#slide=id.g73ce69d719_2_5)

**Post-Mortem Analysis**

The Good: As far as the things that went right throughout this project, I think the team learned about some interesting tools and how they can be theoretically applied. We tried our best to help each other out if there were questions, and the support from other teams/classmates was crucial to our success.

What Could Have Been Done Better: I think the teams collective lack of experience held us back; without Josh we really do not have any solid command line experience. Maybe adding one more grad student to our team would have increased our chances at getting this done in a more timely fashion. Maybe we could have met more, either online or in person, to do the homework so we could get more out of the class. We could also have communicated more frequently throughout the project instead of the week of the due date.