Team: Josh Derrow, Brennan Krutis, & Onyx Bennett

Github Repository: Project Chrono - https://github.com/projectchrono/chrono

Advising Questions:

1. For our contribution, we want to create improved documentation for the installation and functionality of Chrono, in order to help new and experienced users alike.
2. Our project is going OK so far, so we would give our current project state a score of 3.
3. We urgently need feedback on this deliverable. Even though our entire group has working builds, we are still unsure how to extrapolate some of the data generated by the demos and progress in general.
4. Our group is effectively finding meeting times, getting work done, and there have been no issues so far; so it's going great in this aspect.
5. We are unsure whether our expected contribution suffices for the project expectations, or if a more technical contribution is required.

Build Instructions:

1. git clone https://github.com/projectchrono/chrono.git
2. Eigen3_ROOT="/shared/common/eigen-3.4.0/"
3. DTHRUST_DIR=/usr/local/cuda/targets/x86_64-linux/include/thrust/
4. module load mpi
5. cd chrono
6. mkdir build && cd build
7. srun --gres=gpu cmake .. -DTHRUST_INCLUDE_DIR=/usr/local/cuda/targets/x86_64-linux/include/ -DEIGEN3_INCLUDE_DIR="/shared/common/eigen-3.4.0/" -DCH_ENABLE_MODULE_GPU=ON -DCH_ENABLE_MODULE_VEHICLE=ON -DENABLE_MODULE_MULTICORE=ON
8. srun make -j

Demo Instructions:

1. After building the project, cd into build/bin
2. Run the command "srun --gres=gpu ./demo_[module]_[name_of_test] to run a given demo
3. An example demo that we found useful to analyze was ./demo_GPU_repose. First, we discovered that if the srun command does not specify a gpu node to use, the program will not run (showing that the GPU is utilized in the demo). This demonstration is meant to show the effect of granular dynamics on particles in a cone (visual demonstration we found on YouTube: https://www.youtube.com/watch?v=Fsy0Is3_7vU). While this demo is running, the terminal periodically updates the kinetic energy of the particles on a frame-by-frame basis. We tested that the accumulation of kinetic energy was different when modifying the JSON file of the particles (from build/data/gpu/repose.json, modifying values such as particle radius, density, etc.) we observed that when these

values are altered, the GPU will take an increased/decreased time to output a frame of data.

Progress:

So far, we have been able to compile and build Project Chrono on the cluster successfully, and we have run many of the provided demos. On top of the basic build, we have also installed various modules that Chrono has to offer, including Chrono::GPU, Chrono::SYNCHRONO (Uses MPI), Chrono::Vehicle, and Chrono::Multicore, each of which provides multiple demonstrations of their use cases. While many of these demos require HPC Visualization software, others show their computations through the CLI. It can be observed that the hardware is being utilized on the GPUs properly, as when srun is run without the GPU node requested, demos that require CUDA will not run, but work when “--gres=gpu” is included.

We are also currently in the process of verifying the availability of the optional dependencies on the cluster. Some of them may already be installed, like Eigen3 and Thrust, while other dependencies may not already be present/able to be installed on the cluster (Ex. Irrlicht). We have not done any performance experiments yet, but that will be one of our next steps after we run more demos and get fully settled with the configuration of our project build.

Obstacles:

We have run into a few obstacles so far with Project Chrono, but nothing major that can’t be overcome. First, we could not build Chrono on the cluster because the Eigen3 library was not visible, and we didn’t know it was already on the cluster. It was already on the cluster however, so we fixed this issue by specifying the environment variable Eigen3_ROOT with the correct path before the build command (Eigen3_ROOT="/shared/common/eigen-3.4.0/").

Once the correct path was specified, we were then able to build Chrono on the cluster and run a portion of the provided demos, but not all of them. When running the available demos, we noticed that they were producing the same output regardless of what cluster node we were running the demos on (login, compute, & GPU). This led us to our next issue: the demos were not utilizing the nodes properly, especially the GPU nodes. We later found that the cause of this was that the demos we were running were not compute-heavy operations; we ended up finding that in order to utilize the GPU components, the GPU module had to be loaded by changing a CMake option (-DCH_ENABLE_MODULE_GPU=ON).

Another roadblock that we encountered was when trying to enable the Chrono::OpenGL module. This module requires Blaze, a high-performance C++ math library, and we weren’t able to find a directory on the cluster that contained this library.

After that, we were experiencing many persistent build errors because it was having trouble locating thrust/reduce.h, thrust/pair.h, and thrust/sequence.h (even though it is on the cluster, located in /usr/local/cuda/targets/x86_64-linux/include/). We fixed this error by specifying the environment variable DTHRUST_DIR with the correct path (DTHRUST_DIR=/usr/local/cuda/targets/x86_64-linux/include/thrust/) before the build command, just like we did for Eigen3_ROOT. We also include the path to the Thrust include folder in the cmake command. Chrono will not build correctly without taking both of these steps to ensure it can find Thrust.

We also accidentally brought down the cluster while trying to build our project, which was quite unfortunate but it didn't happen at a bad time and it also served as a valuable learning opportunity. After the cluster came back online, we learned that the cause of this issue was inexplicably high memory consumption from VS Code processes that were spawned by two of our accounts, likely due to a VS Code bug.

Moving forward, we are still unable to graphically render any of these models without visualization software (not currently set up on the cluster). However, we can still run the demos and perform all of the necessary computations. The demos can be run on the command line, but it does not seem that they were made to do so. The data being output is not an accurate representation of the simulation because of time increments. For example, a demo of balls falling out of a funnel spends so long reporting the buildup of kinetic energy that we were not able to see the output from the balls actually falling. Here is a link to a playlist from one of the project developers of the various GPU simulations visualized with ParaView ($10,000 per installation):
https://www.youtube.com/watch?v=iSxYApFgqD0&list=PL1uQAy4kmxZBE7SJPRMakG_js3FPnDgFd

To potentially work around this, we are looking into extracting the computed information from the cluster and running the data on one of our local machines, which can support a wider range of HPC visualization software.

Modifications:

We had initially planned on finding a subtle bug listed on the Project Chrono forums (https://groups.google.com/g/projectchrono) and fixing it, however, most of the entries on the forum list specific user-errors or general questions on how to use various parts of the software. Any reports of bugs that we have found are either already fixed or too convoluted for us to tackle. There are also no reported bugs of unknown origin, so we cannot contribute by finding the source of any problems either.

Since, we are having trouble finding a bug that we are confident in fixing, we would like to modify our contribution to creating an improved version of the installation and user documentation. There is already official documentation on how to do this, however, the directions are not very clear and are not beginner-friendly. If we could create an improved version of this documentation, it could save future and current users alike the time and headaches that we have experienced while attempting this.

Finalization Plan:

Assuming our contribution plan is satisfactory to the project requirements, our plan is to identify the pot holes in the current official documentation and fill them. We have already found several of these through our own experiences as well as the community forum for Chrono. We plan to enhance current instructions that we think could use clarification and create a new section of common issues and their solutions. Our goal would be to dedicate a week to identifying common build issues. If we come across reported issues without already discovered solutions, we will try to solve them ourselves during this time. At least another week will be spent revamping the official docs and creating an easily searchable 'common build problems' page.

We have been, and will continue to, meet after class and work individually. We are compiling our proposed changes and additions to the documentation in a shared doc (Command Doc), so it is fairly easy for us to make progress on our own time.