CDA-5106 Project Phase 2

Installation:

Dependencies installation:

sudo apt-get install -y build-essential m4 x11proto-xext-dev libglu1-mesa-dev libxi-dev libxmu-dev libtbb-dev

Next, download the Parsec 3.0 from the website using the command:

wget http://parsec.cs.princeton.edu/download/3.0/parsec-3.0.tar.gz

This link will download everything include the test inputs of all sizes except for the precompiled binaries.

Next, unzip the tarbal using:

tar xvf parsec-3.0.tar.gz

Next we need to add "*parsecmgmt*" to the environment variable in Linux. This can be done using the following command:

source env.sh

The "*env.sh*" has been provided along with the rest of the Parsec files. Now we can use the executable "*parsecmgmt*" for everything from building to running and even removing built binaries for Parsec.

Build parsec using:

time parsecmgmt -a build -p all

The time commands outputs the time it took for the command to execute

Errors during build:
"POD document had syntax errors at /usr/bin/pod2man line 68."
The error occured since Parsec was originally designed to work with older distributions of Linux. There are two ways around this problem, one is to install an older version of Perl. The second, and the one I used, is to update the files to the newer version of Perl. This was done by changing a number of files in the following directory: "*/home/meisam/Downloads/parsec-3.0/pkgs/libs/ssl/src/doc/ssl*" by updating lines according to the following regex *"=item [0-9]the =item C<[0-9]>"*.

I cleaned the working directory by running "*parsecmgmt -a fulluninstall*" and ran the build command again. This time I got another error for unbuilt *freebsd_kern* objects not being built. On investigation, I found that this was because the Parsec definition for "*__mbstate_t*" was conflicting with the system definition. I commented out the definition in the Parsec file "*pkgs/libs/uptcpip/src/include/sys/bsd__types.h*" (the path is relative to the Parsec root directory). and tried to build again. This time the build completed successfully and took a time of 5m36.089s. Although it must be noted that this was because most of the objects had already been built in previous runs and were not removed before running the last build.

Next, I ran the test cases using the command "*parsecmgmt -a run -p all*". The command's output is somewhat misleading as it does not stop on errors. For instance, the test case "*ext/splash2x/apps/ocean_ncp/inst/amd64-linux.gcc/bin/ocean_ncp -n258 -p1 -e1e-07 -r20000 -t28800*" met with a segmentation fault but the commands execution did not stop. With numerous tests

being run, it can tedious to find out if a particular test failed or not. One can use the string matching command from Linux, *grep* to make the job easier. The majority of the test cases ran fine however (see Figure 1 below).

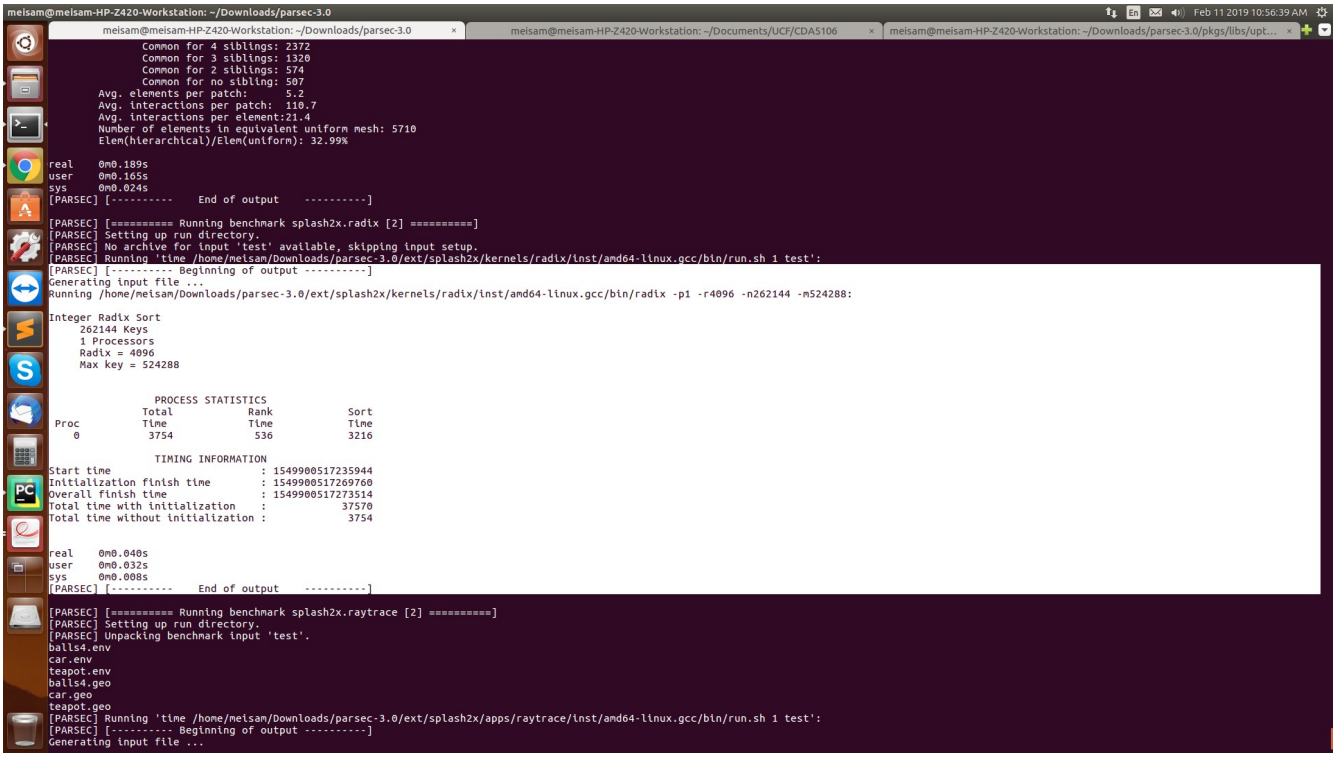

Figure 1: Sample output from benchmarks test runs

Finally, I ran the x264 test cases since my group was assigned that benchmark. All of the benchmarks kept crashing. After extensive debugging I found the error to be in "*pkgs/apps/x264/src/common/set.c*" in the function "*x264_cqm_delete*". The compiler on my machine "*gcc-5.**" was the root cause so I reverted my compiler to "*gcc-4.7*" after which the problem was solved and the benchmarks ran fine (Refer to Figure 2 below).

The benchmark was run using the following command:

parsecmgmt -a run -p x264 -i simdev

We can change the “-i” option to change the size of the input to the benchmark. We have the option between native, simdev, simsmall, simmedium and simlarge. A more detailed discussion will follow in the next phase of the project.

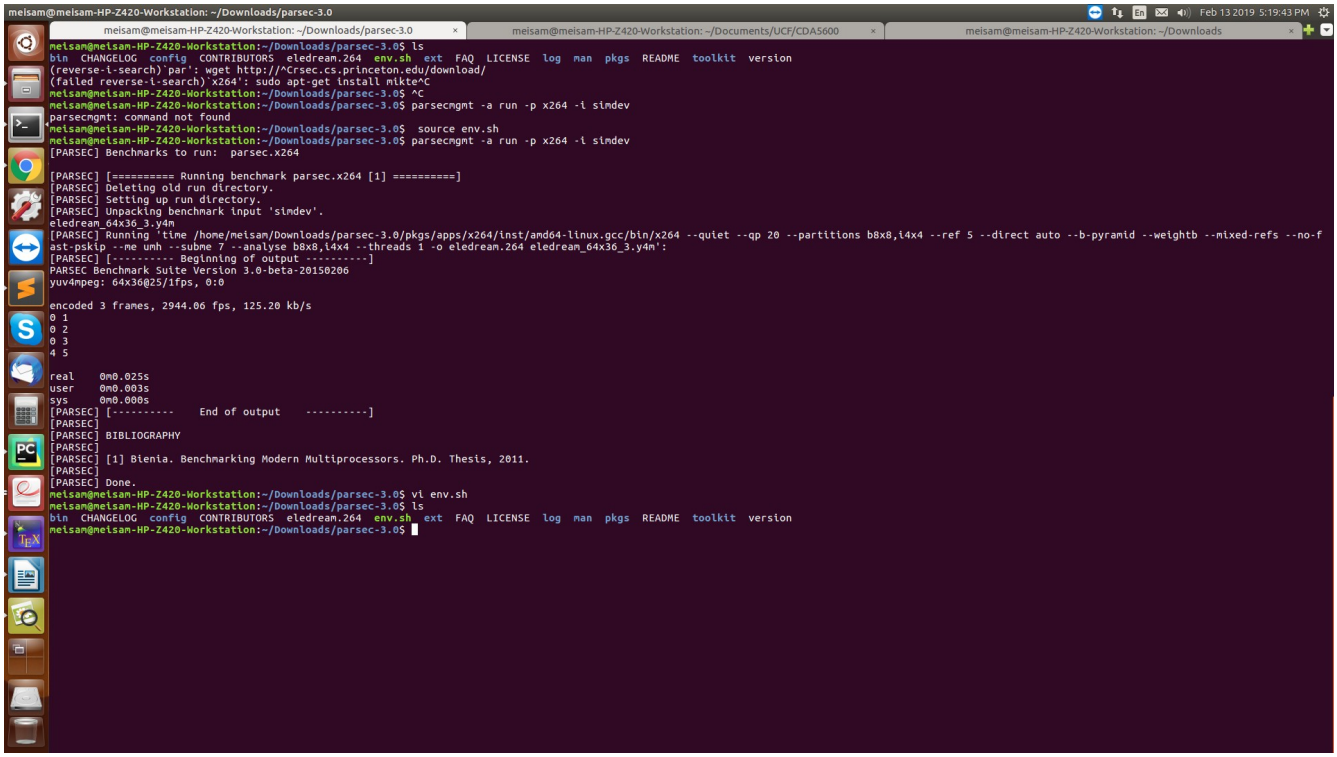

Figure 2: H264 example run