PARSEC SETUP GUIDE

Running Parsec in Gem5 FS/SE mode

1. Environment:
    a. I recommend using a Linux Installation
        i. Virtual machines are completely unusable with this - they'll be too slow
        ii. I dual-booted two of my computers to have Ubuntu and it works great
    b. I Used Ubuntu version 20.04 for my environment
        i. I created a FRESH installation before making this guide.
        ii. I cannot guarantee you'll get similar results with other environments
2. PARSEC Repositories
    a. ORIGINAL Parsec 3.0 files:
        i. https://parsec.cs.princeton.edu/parsec3-doc.htm
        ii. Only HALF of these files would successfully build for me.
        iii. This repository stopped being supported in 2012
    b. UPDATED Parsec 3-UCD Repository
        i. https://github.com/darchr/parsec-benchmark
        ii. I've had a large amount of success with this repository
        iii. It contains a series of patches made by UC Davis
        iv. Is still being updated
    c. ALTERNATIVE Parsec Repositories
        i. Many different organizations are working with PARSEC, and patching it on their own
        ii. If you are having issues with making certain benchmarks work, try going through a google search and finding other repositories
        iii. You can always mix different benchmarks between different repositories to get a fully working version of PARSEC.
        iv. EG: https://github.com/cirosantilli/parsec-benchmark is another current repository. I cannot verify who is working on it, but it is functional.
3. Building PARSEC
    a. Download a parsec repository, and untar if necessary
    b. Open the folder in terminal, then type "source env.sh"
        i. This loads all PARSEC commands into shell, enabling you to use parsecmgmt right in your terminal.
    c. PARSECMGMT
        i. Is a very user friendly interface
            1. type "parsecmgmt -h" for help
            2. type "parsecmgmt -a info" for a list of all benchmarks
        ii. To build, you can try to build all benchmarks
            1. "Parsecmgmt -a build -p all"
            2. I was never able to get this working.
        iii. I successfully built most benchmarks individually with:

1. “parsecmgmt -a run -p benchmarkname”
2. Worked for most benchmarks. Your results may vary.

d. SPLASH-2 with PARSEC
   i. SPLASH-2x is an upgrade on the original SPLASH-2 allowing for variable input sizes
   ii. You can build with:
      1. “Parsecmgmt -a build -p splash2x” OR
      2. “parsecmgmt -a build -p splash2x.benchmarkname”
      3. Once again, if the general build fails try building one at a time

e. COMPILE ERRORS:
   i. With The UCD repository, I was able to build every benchmark except for the SPLASH-2x suite and Raytrace.
   ii. Fortunately, Splash-2 AND Raytrace built successfully with the Parsec 3.0 repository.
      1. I created a working hybridized repository with MOST benchmark folders coming from the UCD repository, and with the SPLASH and RAYTRACE folders coming from Parsec3.0
         a. SPLASH2x is found in PARSEC/ext/
         b. Raytrace is found in pkgs/apps

f. The following table shows my compilation success for differing repositories:

Compilation			
Benchmark	PARSEC 3.0	PARSEC 3-UCD	Hybridized
Blackscholes	Compiles	Compiles	Compiles
Bodytrack	No Compile	Compiles	Compiles
Canneal	Compiles	Compiles	Compiles
Dedup	No Compile	Compiles	Compiles
Facesim	No Compile	Compiles	Compiles
Ferret	No Compile	Compiles	Compiles
Fluidanimate	Compiles	Compiles	Compiles
Freqmine	Compiles	Compiles	Compiles
Raytrace	Compiles	No Compile	(Old) Compiles
Streamcluster	Compiles	Compiles	Compiles
Swaptions	Compiles	Compiles	Compiles
Vips	Compiles	Compiles	Compiles
x264	No Compile	Compiles	Compiles
Netdedup	No Compile	Compiles	Compiles
Netferret	No Compile	Compiles	Compiles
Netstreamcluster	No Compile	Compiles	Compiles
SPLASH-2 Suite	Compiles	No Compile	(Old) Compiles

4. RUNNING benchmarks with PARSECMGMT
    a. Use “parsecmgmt -h” to see ALL flag options for running like numthreads
    b. Examples:
        i. “Parsecmgmt -a run -p benchmarkname”
        ii. “Parsecmgmt -a run -p all -n 50”
    c. Input Sizes:
        i. Check the links to the original repositories for installing different input sizes
        ii. Sizes vary :
            1. “test” & “simdev” are very small
            2. “simsmall” & “simmedium” are probably best for testing
            3. “simhard” may be best for getting data for papers
            4. “native” is ridiculously large
        iii. EG:
            1. “Parsecmgmt -a run -p blackscholes -i simdev”
    d. The following table shows which benchmarks I could run with parsecmgmt:

Benchmark	Running w/ parsecmgmt
SPLASH-2 Suite	Runs
Blackscholes	Runs
Fluidanimate	Runs
Swaptions	Runs
Raytrace	Runs
Streamcluster	Runs
Freqmine	Runs
Vips	Runs
Netdedup	Runs
Netferret	Runs
Netstreamcluster	Runs
Bodytrack	Runs
Ferret	Runs
Dedup	Runs
Facesim	Runs
Canneal	SegFault
x264	SegFault

5. Running benchmarks WITHOUT parsecmgmt
    a. Whenever you run a benchmark, it will print out the EXACT line that is used in the shell to run the benchmark
    b. EG:
        i. “Parsecmgmt -a run -p blackscholes -i test” gave me
        ii. “path/PARSEC/pkgs/apps/blackscholes/inst/amd-linux.gcc/bin/blackscholes 1 in_4.txt prices.txt”

```
nate@beany-buntu:~/NoC-Research/Parsec3.1$ parsecmgmt -a run -p blackscholes -i test
[PARSEC] Benchmarks to run:  parsec.blackscholes

[PARSEC] [========== Running benchmark parsec.blackscholes [1] ==========]
[PARSEC] Setting up run directory.
[PARSEC] Unpacking benchmark input 'test'.
in_4.txt
[PARSEC] Running 'time /home/nate/NoC-Research/Parsec3.1/pkgs/apps/blackscholes/inst/amd64-linux.gcc/bin/blackscholes 1 in_4.txt prices.txt':
```

    c. These commands start with the object file, then are followed with many different arguments such as input files and output files, numthreads, etc.
        i. ALL arguments have the potential to change based on input sizes. For example, bodytrack increases the number of frames to simulate as the input size increases.
            1. EG: #frames go from 1 to 2 (simmedium) to 4, to 261

```
Bodytrack Arguments by Input:
<input> <#cams> <#frames> <#parts> <#layers> <threadmod> <#threads>
TEST:             sequenceB_1 4   1    5 1 0 1
SIMDEV:           sequenceB_1 4   1  100 3 0 1
SIMSMALL:         sequenceB_1 4   1 1000 5 0 1
SIMMEDIUM:        sequenceB_2 4   2 2000 5 0 1
SIMLARGE:         sequenceB_4 4   4 4000 5 0 1
NATIVE:         sequenceB_261 4 261 4000 5 0 1

SIMSMALL w/ Changing threads:
1Threads:       sequenceB_1 4 1 1000 5 0 1
2Threads:       sequenceB_1 4 1 1000 5 0 2
4Threads:       sequenceB_1 4 1 1000 5 0 4
```

        ii. Differing input sizes cause the program to be run on different source data. Although the input filename inside of the shell command may be the same between input sizes, the actual file comes from a different place and has differing size and content.
        iii. The input files and output files can be found in the input folder for a given benchmark. They are stored in a zipped state.
            1. Parsecmgmt UNZIPS the input file into the “run” folder for a given benchmark. Then, the shell command is run as if the terminal is inside of that run folder. That’s why input/output files do not have a path.
    d. If you want to run PARSEC properly, be very careful to ensure that all of the input arguments fit the ones used by parsecmgmt for your target input size

i. If you want more information on the arguments passed, try running the object file for the given benchmark with -h. The object files are in varying locations, but their path is shown when using parsecmgmt.

6. Running a benchmark with GEM5 in Syscall emulation mode (SE)
    a. Once you can run a benchmark without parsecmgmt, you can easily run it in syscall emulation mode
    b. Use the following command layout:
        i. “path/gem5.opt path/se.py [args] -c path/objectFile -o “args for objectFile”
    c. EG:
        i. build/X86/gem5.opt configs/example/se.py --num-cpus=4 --cpu-type=TimingSimpleCPU --cpu-clock=2GHz --caches --l1d_size=16kB --l1i_size=16kB --mem-size=4GB -c benchmark/parsec/blackscholes/blackscholes -o "16 in_4.txt prices.txt"
        ii. You can always use a direct path to input/output files if that makes your setup easier to run
    d. Current Functionality: With my hybridized repository, I got 5 benchmarks running:

Current PARSEC functionality		
Benchmark	Running w/ parsecmgmt	Running w/ gem5 SE mode
SPLASH-2 Suite	Runs	Runs
Blackscholes	Runs	Runs
Fluidanimate	Runs	Runs
Swaptions	Runs	Runs
Raytrace	Runs	Runs
Streamcluster	Runs	Runs
Freqmine	Runs	Syscall error -either implement the syscall or use FS
Vips	Runs	Syscall error - either implement the syscall or use FS
Netdedup	Runs	2 executables - Requires FS Mode
Netferret	Runs	2 executables - Requires FS Mode
Netstreamcluster	Runs	2 executables - Requires in FS Mode
Bodytrack	Runs	Runs - but never finishes (no error code)
Ferret	Runs	Assertion (obj!=NULL) fails
Dedup	Runs	Assertion (s2!=NULL) fails
Facesim	Runs	Core Dump
Canneal	SegFault	SegFault
x264	SegFault	SegFault

    e. The errors fall under three categories:
        1. Syscall Error - this means that the program requires a system call that is NOT implemented in SE mode. You can either fix this error by manually implementing the missing system call, or by using FS mode
        2. 2 executables - this means that the benchmark requires two separate object files to be run. One for a client, and one for a

server. For multiple clients, you need even more executables. This is not doable in SE mode - FS mode is required
3. Miscellaneous - I was unable to isolate the error

7. Full-System Mode (FS)
    a. Benefits and costs of FS
        i. Heavily Recommended for PARSEC benchmarking from various sources
            1. Pthreads are not properly implemented in SE
            2. Pthreads are a MAJOR part of PARSEC
        ii. Allows for more realistic testing (includes operating system)
        iii. Implements all syscalls (would help w/ some broken benchmarks)
        iv. Allows running multiple processes
        v. Might allow for the use of parsecmgmt - which would make running the benchmarks astronomically easier, and could potentially trivialize getting all the benchmarks to run successfully
        vi. Much more difficult to set up
        vii. Simulations take much longer (FS runs like a VM)
    b. I was unable to get a disk for FS mode to work properly.
        i. In order to run FS mode, you create a disk image and run it similarly to a virtual machine
        ii. The disk image needs an operating system installed on it, along with some modifications to allow it to interface with gem5
    c. Official Guide - ways to create a disk image:
        i. https://www.gem5.org/documentation/general_docs/fullsystem/disks
        ii. Provides four options for getting a disk:
            1. Use gem5 utils to create a disk image
            2. Use gem5 utils and chroot to create a disk image
            3. Use QEMU to create a disk image
            4. Use Packer to create a disk image
        iii. I ran into issues with all four options. You may get different results. Try following the guides on your own before looking at my results:
            1. Is not a full guide. Following it led to a dead end with an empty disk image file.
            2. Created the disk image with the same version of ubuntu shown in the guide. Added all of the modifications. Did not run with gem5
            3. Was able to boot and run a VM using QEMU. I installed Ubuntu server version 20.10, but ran into great difficulty SSH'ing into it. There is promise with this route.
            4. Packer has had some large changes since the guide came out. They completely reworked how to create and modify templates. This is another promising route, if more Packer research is done.
    d. Potential workaround - download a working disk image
        i. https://www.gem5.org/documentation/general_docs/fullsystem/guest_binaries

ii. Tried working with an ARM image, was unable to compile gem5.opt for ARM.
    1. I believe a cross compiler might be necessary (unsure)
iii. Was unable to find an image for other binaries from this link