



PHASE 2, 3

Parsec Benchmark Installation and Simulation with Gem5



Submitted by: Maliha Arif

PID: 4506817

Submitted to: Dr. Dan C. Marinescu

Submitted on: 11th October 2018

ADVANCED COMPUTER ARCHITECTURE

UNIVERSITY OF CENTRAL FLORIDA

Fall 2018

1.Phase 2: Parsec benchmark Installation

1.1. Parsec:

The Princeton Application Repository for Shared-Memory Computers (PARSEC) is a benchmark suite composed of multithreaded programs. It consists of 13 workloads, 10 applications and 3 kernels common to desktop and server programs. It has been developed at Princeton as a PhD dissertation of a researcher in 2008.

PARSEC differs from other benchmark suites in the following ways:

- **Multithreaded**: While serial programs are abundant, they are of limited use for evaluation of multiprocessor machines. PARSEC is one of few benchmark suites that are parallel.
- **Emerging Workloads**: The suite includes emerging workloads which are likely to become important applications in the near future but which are currently not commonly used. Our goal was to provide a collection of applications as might be typical in a few years.
- **Diverse**: PARSEC does not try to explore a single application domain in detail, as was done by several previous benchmark suites. The selection of included programs is wide and tries to be as representative as possible.
- **Not HPC-Focused**: Computationally intensive, parallel programs are very common in the domain of High-Performance Computing (HPC), but HPC programs are just a small subset of the whole application space. In the future, parallelization techniques will become increasingly popular in other areas as well. The PARSEC suite anticipates this development and tries to avoid a program selection which is skewed towards HPC. It focuses on programs from all domains, such as desktop and server applications.
- **Research**: The suite is primarily intended for research. It can also be used for performance measurements of real machines, but its original purpose is insight, not numbers. Benchmark suites intended for research usually go beyond pure scoring systems and provide infrastructure to instrument and manipulate included programs in an efficient manner.

1.2. Workloads:

The current version of the suite contains the following 13 programs from many different areas such as computer vision, video encoding, financial analytics, animation physics and image processing:

The 13 workloads are as follows:

- blackscholes - Option pricing with Black-Scholes Partial Differential Equation (PDE)
- bodytrack - Body tracking of a person
- canneal - Simulated cache-aware annealing to optimize routing cost of a chip design
- dedup - Next-generation compression with data deduplication
- facesim - Simulates the motions of a human face
- ferret - Content similarity search server
- fluidanimate - Fluid dynamics for animation purposes with Smoothed Particle Hydrodynamics (SPH) method
- frequentmine - Frequent itemset mining
- raytrace - Real-time raytracing
- streamcluster - Online clustering of an input stream
- swaptions - Pricing of a portfolio of swaptions
- vips - Image processing
- x264 - H.264 video encoding

Our group 5 basically has been assigned benchmark 5- facesim to simulate and work with in phase 3-simulation with gem5. But before we go to that phase, let's first discuss how the benchmark was installed and which operating system was used.

1.3. Architecture/Operating System Used:

Linux – Ubuntu 16.01

1.4. Parsec download and installation

Use the following sequence of commands to download and extract the Parsec-3.0 benchmark suite on the terminal. In our case, we are using a Ubuntu desktop and hence these commands have to be entered in the terminal.

Reference link: <http://parsec.cs.princeton.edu/parsec3-doc.htm>

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

This basically downloads the tar file in your home directory. Once extracted, do the following.

- **cd parsec-3.0**
- **source env.sh**

This takes you to the parsec directory where you set environment variable.

Here you can build different benchmarks using the following 2 commands. 'Parsecmgmt' command is specific to this benchmark suite and will run once you add the environment variable using the source env.sh command.

- **parsecmgmt -a build -p "benchmark"**
- **parsecmgmt -a run -p "benchmark"**

Here benchmark can be any of the following.

1. blackscholes
2. bodytrack
3. canneal
4. dedup
5. facesim
6. ferret
7. fluidanimate
8. freqmine
9. raytrace
10. streamcluster
11. swaptions
12. vips
13. x264

- Building gives us the following results

```
vision@maliha-precision-workstation-t3500:~/parsec-3.0$ parsecmgmt -a build -p streamcluster
[PARSEC] Packages to build: parsec.streamcluster

[PARSEC] [===== Building package parsec.streamcluster [1] =====]
[PARSEC] [----- Analyzing package parsec.streamcluster -----]
[PARSEC] parsec.streamcluster does not depend on any other packages.
[PARSEC] [----- Building package parsec.streamcluster -----]
[PARSEC] Copying source code of package parsec.streamcluster.
[PARSEC] Running 'env version=threads /usr/bin/make':
/usr/bin/g++ -O3 -g -funroll-loops -fprefetch-loop-arrays -fpermissive -fno-exceptions -static-libgcc -WL,--hash-style=both,--as-needed
ON=3.0-beta-20150206 -DENABLE_THREADS -pthread -c streamcluster.cpp
streamcluster.cpp: In function 'void outcenterIDs(Points*, long int*, char*)':
streamcluster.cpp:1825:39: warning: format '%u' expects argument of type 'unsigned int', but argument 3 has type 'long int' [-Wformat=]
    fprintf(fp, "%u\n", centerIDs[i]);
                                ^
streamcluster.cpp: In function 'void streamcluster(PStream*, long int, long int, int, long int, long int, char*)':
streamcluster.cpp:1890:46: warning: format '%d' expects argument of type 'int', but argument 3 has type 'size_t {aka long unsigned int}'
    fprintf(stderr, "read %d points\n", numRead);
                                ^
/usr/bin/g++ -O3 -g -funroll-loops -fprefetch-loop-arrays -fpermissive -fno-exceptions -static-libgcc -WL,--hash-style=both,--as-needed
ON=3.0-beta-20150206 -DENABLE_THREADS -pthread -c parsec_barrier.cpp
/usr/bin/g++ -O3 -g -funroll-loops -fprefetch-loop-arrays -fpermissive -fno-exceptions -static-libgcc -WL,--hash-style=both,--as-needed
ON=3.0-beta-20150206 -DENABLE_THREADS -pthread -L/usr/lib64 -L/usr/lib -static streamcluster.o parsec_barrier.o -o streamcluster
[PARSEC] Running 'env version=threads /usr/bin/make install':
mkdir -p /home/vision/parsec-3.0/pkgs/kernels/streamcluster/inst/amd64-linux.gcc/bin
cp -f streamcluster /home/vision/parsec-3.0/pkgs/kernels/streamcluster/inst/amd64-linux.gcc/bin/streamcluster
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Bienia. Benchmarking Modern Multiprocessors. Ph.D. Thesis, 2011.
[PARSEC]
[PARSEC] Done.
vision@maliha-precision-workstation-t3500:~/parsec-3.0$
```

- Testing with different benchmarks and then our benchmark '5' facesim gives us the following output at run time.

```
vision@maliha-precision-workstation-t3500:~/parsec-3.0$ parsecmgmt -a build -p facesim
[PARSEC] Packages to build: parsec.facesim

[PARSEC] [===== Building package parsec.facesim [1] =====]
[PARSEC] [----- Analyzing package parsec.facesim -----]
[PARSEC] Package parsec.facesim already exists, proceeding.
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Bienia. Benchmarking Modern Multiprocessors. Ph.D. Thesis, 2011.
[PARSEC]
[PARSEC] Done.
vision@maliha-precision-workstation-t3500:~/parsec-3.0$ parsecmgmt -a run -p facesim
[PARSEC] Benchmarks to run: parsec.facesim

[PARSEC] [===== Running benchmark parsec.facesim [1] =====]
[PARSEC] Deleting old run directory.
[PARSEC] Setting up run directory.
[PARSEC] No archive for input 'test' available, skipping input setup.
[PARSEC] Running 'time /home/vision/parsec-3.0/pkgs/apps/facesim/inst/amd64-linux.gcc/bin/facesim -h':
[PARSEC] [----- Beginning of output -----]
PARSEC Benchmark Suite Version 3.0-beta-20150206
Usage: /home/vision/parsec-3.0/pkgs/apps/facesim/inst/amd64-linux.gcc/bin/facesim [-restart <int>] [-l]
-restart    <no description available> (default 0)
-lastframe  <no description available> (default 300)
-threads    <no description available> (default 1)
-timing     <no description available>

real    0m0.057s
user    0m0.000s
sys     0m0.000s
[PARSEC] [----- End of output -----]
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Bienia. Benchmarking Modern Multiprocessors. Ph.D. Thesis, 2011.
[PARSEC]
[PARSEC] Done.
vision@maliha-precision-workstation-t3500:~/parsec-3.0$
```

1.5. Parsec benchmark with some test input

The above is when we test the benchmark without any input. If we get the above output, it shows parsec has been correctly installed but there is a way we can test the benchmark with some input also. It can be done in the following ways.

- **wget <http://parsec.cs.princeton.edu/download/3.0/parsec-3.0-input-sim.tar.gz>**

Once downloaded, we should move this to parsec-3.0 root directory and then extract it.

- **mv parsec-3.0-input-sim.tar.gz ~/**
- **cd ~/**
- **ls**
parsec-3.0 parsec-3.0-input-sim.tar.gz
- **tar -xzf parsec-3.0-input-sim.tar.gz**

The unpacking will show us a new folder in the parsec-3.0 directory with the name 'pkgs'. Inside which there will be different folders like 'apps', 'kernel' etc. Apps contain subfolders with different benchmarks.

We used the one named 'facesim'. The following command runs the benchmark with some input.

- **parsecmgmt -a run -p facesim -i simsmall**

Screenshot is attached below.

After running, it creates a folder run with 'FaceData' and 'storytelling' folder.

Benchmark output with some input >

```

Simulation                                     Reading simulation model : ./Face_Data/Eftychis_840k/Front_370k/face_simulation_1.tet
Total particles = 80598
Total tetrahedra = 372126
muscles = 32
attachments = 3

Frame 1
END Frame 1                                4.5547 s
SIMULATION                                0.0000
FRAME                                    4.5547
ADB                                    4.5544
  UPBS                                0.7911
    UPBS (FEM) - Initialize              0.0643 s
    UPBS (FEM) - Element Loop            0.7267 s
    UPBS (CPF)                          0.0000 s
  ADO                                    3.7556
    ADO - Update collision bodies         0.0000 s
    AOTSQ                                3.7555
      AOTSQ - Initialize                 0.0003 s
      AOTSQ - NR loop                   3.7551
        AOTSQ - NR loop - Initialize     0.0001 s
      UCPF                               0.0679 s
      NRS                               2.7376
        NRS - Initialize                0.0000 s
        NRS - Boundary conditions 1     0.0003 s
        UPBS                            0.8563
          UPBS (FEM) - Initialize         0.0000 s
          UPBS (FEM) - Element Loop       0.8563 s
          UPBS (CPF)                    0.0000 s
        AFD (FEM)                       0.0079 s
        AVIF                             0.0833
          AVIF (FEM)                    0.0831 s
          AVIF (CPF)                    0.0002 s
        NRS - Boundary conditions 2     0.0000 s
        NRS - Compute residual          0.0003 s
        NRS - Copy initial guess        0.0002 s
      UPBS                              0.8634
        UPBS (FEM) - Initialize          0.0000 s
        UPBS (FEM) - Element Loop       0.8634 s
        UPBS (CPF)                     0.0000 s
      AVIF                              0.0854
        AVIF (FEM)                     0.0852 s
        AVIF (CPF)                     0.0001 s
      AOTSQ - NR loop - Boundary conditions 0.0000 s
      AOTSQ - NR loop - Compute residual 0.0002 s

real    0m6.542s
user    0m6.236s
sys     0m0.108s
[PARSEC] [----- End of output -----]
[PARSEC]

```

Challenges/Problems: Parsec installation was very smooth and simple without any complications, running Parsec with Gem5 was challenging, the difficulties of which shall be discussed later in the report.

2. Running Parsec benchmark with Gem5

To run Parsec with gem5, we need to download a Research Starter Kit. This kit includes patches, 2 of which we will use to enable Gem5 and Parsec to work together.

This repository can be downloaded in the home folder using the following command.

- `git clone https://github.com/arm-university/arm-gem5-rsk.git`

To simulate the benchmarks on Gem5, we need to compile Parsec benchmarks with ARM architecture.

There are 2 ways of doing this

1) Cross-compiling on x86 machine

2) Compiling on QEMU

We used the first technique, i.e. cross compile with x86 and it worked successfully. 2 common steps to both techniques are as following.

1) Apply static-patch.diff from the Research Starter Kit to support generating static binaries (in the parsec-3.0 directory):

- `patch -p1 < ../arm-gem5-rsk/parsec_patches/static-patch.diff`

2) Update PARSEC to recognize the ARM architecture that we will use to build PARSEC (replacing the config.sub and config.guess with different versions) **(screenshot below)**

- `mkdir tmp; cd tmp # make a tmp dir outside the parsec dir`
- `wget -O config.guess 'http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD'`
- `wget -O config.sub 'http://git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD'`
- `cd /parsec-3.0`
- `find . -name "config.guess" -type f -print -execdir cp {} config.guess_old \;`
- `find . -name "config.guess" -type f -print -execdir cp /home/vision/tmp /config.guess {} \;`
- `find . -name "config.sub" -type f -print -execdir cp {} config.sub_old \;`
- `find . -name "config.sub" -type f -print -execdir cp /home/vision/tmp /config.sub {} \;`


```

vision@maliha-precision-workstation-t3500:~/parsec-3.0$ find . -name "config.sub" -type f -print -execdir cp {} config.sub_old \;
./pkgs/apps/x264/src/config.sub
./pkgs/apps/bodytrack/src/config.sub
./pkgs/apps/vips/src/config.sub
./pkgs/libs/mesa/src/bin/config.sub
./pkgs/libs/gsl/src/config.sub
./pkgs/libs/libjpeg/src/config.sub
./pkgs/libs/glib/src/config.sub
./pkgs/libs/libxml2/src/config.sub
./pkgs/tools/yasm/src/config/config.sub
./pkgs/tools/libtool/src/libltdl/config/config.sub
vision@maliha-precision-workstation-t3500:~/parsec-3.0$ find . -name "config.sub" -type f -print -execdir cp /home/vision/tmp/config.sub {} \;
./pkgs/apps/x264/src/config.sub
./pkgs/apps/bodytrack/src/config.sub
./pkgs/apps/vips/src/config.sub
./pkgs/libs/mesa/src/bin/config.sub
./pkgs/libs/gsl/src/config.sub
./pkgs/libs/libjpeg/src/config.sub
./pkgs/libs/glib/src/config.sub
./pkgs/libs/libxml2/src/config.sub
./pkgs/tools/yasm/src/config/config.sub
./pkgs/tools/libtool/src/libltdl/config/config.sub

```

2.1. Cross-compiling on x86 machine

After these 2 common steps, we compiled Parsec benchmark using x86. Following commands were used.

Download and extract the aarch-linux-gnu toolchain using

- `wget https://releases.linaro.org/components/toolchain/binaries/latest-5/aarch64-linux-gnu/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz`
- `tar xvfJ gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz`

Once it is extracted outside the main Parsec directory, we should do the following:

- Open “arm-gem5-rsk/parsec_patches/xcompile-patch.diff” with an editor and change the following lines

```

+ export CC_HOME=<gcc-linaro-directory>
+ export BINUTIL_HOME=<gcc-linaro-directory>/aarch64-linux-gnu

```

```

# CC_HOME is installation root of the C compiler
- export CC_HOME="/usr"
+ export CC_HOME="/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu"
# BINUTIL_HOME is installation root of the GNU binutils
- export BINUTIL_HOME="/usr"
+ export BINUTIL_HOME="/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/aarch64-linux-gnu"
# GNUTOOl_HOME is installation root of the GNU tools
export GNUTOOl_HOME="/usr"
# BINARY_PREFIX is the string which is used as prefix for the GNU binaries
export BINARY_PREFIX=""

```

Now apply the patch from within the parsec directory by typing:

- **patch -p1 < ../arm-gem5-rsk/parsec_patches/xcompile-patch.diff**

Further compile parsec by typing the following:

- **export PARSECPLAT="aarch64-linux"**
- **source ./env.sh**
- **parsecgmt -a build -p facesim**

We did the above and got no error, hence we were able to cross compile Facesim with x86 successfully. Few of my group members had some trouble, they hence used a different benchmark. The following shows the output when Facesim was compiled with x86.

```

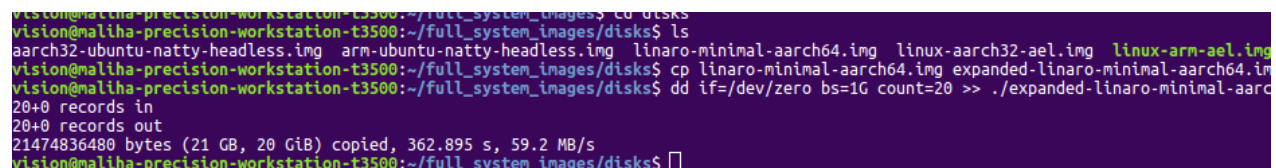
3.0/objs/EXAMPLE.o obj/objs/AND_UTILS.o obj/objs/SOLIBS_PARAMETERS.o obj/ThreadUtilities/THREAD_DIVISION_PARAMETERS.o obj/Thread
o obj/ThreadUtilities/THREAD_POOL_SINGLE.o obj/ThreadUtilities/THREAD_ARRAY_LOCK.o obj/Utilities/DEBUG_UTILITIES.o obj/Utilit
/Utilities/STRING_UTILITIES.o obj/Utilities/TIMER.o
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/aarch64-linux-gnu/bin/ar: creating /home/vision/parsec-3.0/pkgs/
make[2]: Leaving directory '/home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu/Public_Library'
cd /home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu/Public_Library && /usr/bin/make -f Makefile.PhysBAM
make[2]: Entering directory '/home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu/Public_Library'
make[2]: Nothing to be done for 'default'.
make[2]: Leaving directory '/home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu/Public_Library'
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++ -c -O3 -g -funroll-loops -fprefetch-lo
type=both,--as-needed -DPARSEC_VERSION=3.0-beta-20150206 -fexceptions -DENABLE_PTHREADS -I/home/vision/parsec-3.0/pkgs/apps/f
-DNDEBUG -o obj/main.o main.cpp
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++ -c -O3 -g -funroll-loops -fprefetch-lo
type=both,--as-needed -DPARSEC_VERSION=3.0-beta-20150206 -fexceptions -DENABLE_PTHREADS -I/home/vision/parsec-3.0/pkgs/apps/f
-DNDEBUG -o obj/FACE_DRIVER.o FACE_DRIVER.cpp
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++ -c -O3 -g -funroll-loops -fprefetch-lo
type=both,--as-needed -DPARSEC_VERSION=3.0-beta-20150206 -fexceptions -DENABLE_PTHREADS -I/home/vision/parsec-3.0/pkgs/apps/f
-DNDEBUG -o obj/FACE_LANDMARK_OPTIMIZATION_GOAL.o FACE_LANDMARK_OPTIMIZATION_GOAL.cpp
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++ -c -O3 -g -funroll-loops -fprefetch-lo
type=both,--as-needed -DPARSEC_VERSION=3.0-beta-20150206 -fexceptions -DENABLE_PTHREADS -I/home/vision/parsec-3.0/pkgs/apps/f
-DNDEBUG -o obj/FACE_OPTIMIZATION.o FACE_OPTIMIZATION.cpp
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++ -c -O3 -g -funroll-loops -fprefetch-lo
type=both,--as-needed -DPARSEC_VERSION=3.0-beta-20150206 -fexceptions -DENABLE_PTHREADS -I/home/vision/parsec-3.0/pkgs/apps/f
-DNDEBUG -o obj/LANDMARK_PROXIMITY_OPTIMIZATION.o LANDMARK_PROXIMITY_OPTIMIZATION.cpp
/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++ -WL,-o,facesim obj/main.o obj/FACE_DRI
o obj/LANDMARK_PROXIMITY_OPTIMIZATION.o -LPhysBAM /home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu/TaskQ/lib/ta
m/obj/aarch64-linux-gnu/lib -L/home/vision/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu/lib64 -L/home/vision/gcc-linaro-5.
make[1]: Leaving directory '/home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu/Benchmarks/facesim'
[PARSEC] Running 'env version=threads PHYSBAM=/home/vision/parsec-3.0/pkgs/apps/facesim/obj/aarch64-linux-gnu CXXFLAGS=-O3 -g
-static-libgcc -WL,--hash-style=both,--as-needed -DPARSEC_VERSION=3.0-beta-20150206 -fexceptions /usr/bin/make install':
mkdir -p /home/vision/parsec-3.0/pkgs/apps/facesim/inst/aarch64-linux-gnu/bin
cp -f Benchmarks/facesim/facesim /home/vision/parsec-3.0/pkgs/apps/facesim/inst/aarch64-linux-gnu/bin/facesim
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Bienia. Benchmarking Modern Multiprocessors. Ph.D. Thesis, 2011.
[PARSEC]
[PARSEC] Done.
vision@maliha-precision-workstation-t3500:~/parsec-3.0$

```

2.2. Expanding ARM disk image

The gem5 FS mode doesn't support shared directories with the host. Therefore, the compiled PARSEC benchmarks need to be copied to the ARM disk image. This will require expanding the disk image:

- `cd ../full_system_images/disk`
- `cp linaro-minimal-aarch64.img expanded-linaro-minimal-aarch64.img`
- `dd if=/dev/zero bs=1G count=20 >> ./expanded-linaro-minimal-aarch64.img`
- `sudo parted expanded-linaro-minimal-aarch64.img resizepart 1 100%`



```
vision@maliha-precision-workstation-t3500:~/full_system_images$ cd disks
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$ ls
aarch32-ubuntu-natty-headless.img  arm-ubuntu-natty-headless.img  linaro-minimal-aarch64.img  linux-aarch32-ael.img  linux-arm-ael.img
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$ cp linaro-minimal-aarch64.img expanded-linaro-minimal-aarch64.img
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$ dd if=/dev/zero bs=1G count=20 >> ./expanded-linaro-minimal-aarch64.img
20+0 records in
20+0 records out
21474836480 bytes (21 GB, 20 GiB) copied, 362.895 s, 59.2 MB/s
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$
```

After expanding the image disk, we need to mount the expanded disk image and add PARSEC benchmark to it:

- `mkdir disk_mnt`
- `name=$(sudo fdisk -l expanded-linaro-minimal-aarch64.img | tail -1 | awk -F: '{ print $1 }' | awk -F" " '{ print $1 }')`
- `start_sector=$(sudo fdisk -l expanded-linaro-minimal-aarch64.img | grep $name | awk -F" " '{ print $2 }')`
- `units=$(sudo fdisk -l expanded-linaro-minimal-aarch64.img | grep ^Units | awk -F" " '{ print $9 }')`
- `sudo mount -o loop,offset=$((start_sector*units)) expanded-linaro-minimal-aarch64.img disk_mnt`
- `df # find /dev/loopX for disk_mnt`
- `sudo resize2fs /dev/loopX # resize filesystem`
- `df # check that the Available space for disk_mnt is increased`
- `sudo cp -r /path_to_compiled_parsec-3.0_dir/ disk_mnt/home/root # copy the compiled parsec-3.0 to the image`
- `ls disk_mnt/home/root # check the parsec-3.0 contents`
- `sudo umount disk_mnt`

The above steps are easy to execute except for one major error we get while executing `sudo mount` command, this has been discussed under Challenges heading ahead.

We add the parsec benchmark, resize and then verify using the df command. Results are as follows:

df output

```
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
udev            8193284         0    8193284   0% /dev
tmpfs           1642680        9916    1632764   1% /run
/dev/sdb1       817710984 277360584 498789980  36% /
tmpfs           8213384       190216    8023168   3% /dev/shm
tmpfs           5120           4        5116   1% /run/lock
tmpfs           8213384         0    8213384   0% /sys/fs/cgroup
cgfs            100            0         100   0% /run/cgmanager/fs
tmpfs           1642680        76    1642604   1% /run/user/1001
/dev/sda3       300197884 196557760 103640124  66% /media/vision/OS
/dev/loop0      1031800        89484    889904  10% /home/vision/full_system_images/disks/disk_mnt
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$
```

Resize

```
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$ sudo resize2fs /dev/loop0
resize2fs 1.42.13 (17-May-2015)
Filesystem at /dev/loop0 is mounted on /home/vision/full_system_images/disks/disk_mnt; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 2
The filesystem on /dev/loop0 is now 5504952 (4k) blocks long.
vision@maliha-precision-workstation-t3500:~/full_system_images/disks$
```

2.3. Run gem5 FS with PARSEC benchmark

Now we need to go back to gem5 root folder and run gem5 using the expanded disk that contains Parsec benchmark.

The command that worked correctly, i.e. allowed gem5 to simulate properly with Parsec benchmark is as under, **other commands had some issue, discussed under challenges heading.**

- `./build/ARM/gem5.opt -d fs_results/facesim configs/example/fs.py --disk-image=/home/vision/full_system_images/disks/expanded-linaro-minimal-aarch64.img -machine-type=VEExpress_EMM64`

So we use fs.py as our config file, disk image is the new expanded disk and since the disk is 64 bit, we have to specify the machine type in terminal so that Gem5 simulates correctly.

After this, we should open another command terminal and type

➤ **telnet localhost 3456**

This opens a session and if simulation starts fine, we will see this.

```
vision@maliha-precision-workstation-t3500:~/gem5$ ./build/ARM/gem5.opt -d fs_results/facesim configs/example
=/home/vision/full_system_images/disks/expanded-linaro-minimal-aarch64.img
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 compiled Aug 31 2018 22:39:00
gem5 started Oct 10 2018 18:26:08
gem5 executing on maliha-precision-workstation-t3500, pid 9963
command line: ./build/ARM/gem5.opt -d fs_results/facesim configs/example/fs.py --mem-type=HBM_1000_4H_1x64 -
ks/expanded-linaro-minimal-aarch64.img

warn: You are trying to use Ruby on ARM, which is not working properly yet.
Global frequency set at 1000000000000 ticks per second
warn: DRAM device capacity (256 Mbytes) does not match the address range assigned (512 Mbytes)
info: kernel located at: /home/vision/full_system_images/binaries/vmlinux.aarch32.ll_20131205.0-gem5
system.vncserver: Listening for connections on port 5900
system.terminal: Listening for connections on port 3456
0: system.remote_gdb: listening for remote gdb on port 7000
info: Using bootloader at address 0x10
info: Using kernel entry physical address at 0x80000000
info: Loading DTB file: /home/vision/full_system_images/binaries/vexpress.aarch32.ll_20131205.0-gem5.1cpu.dt
**** REAL SIMULATION ****
warn: Existing EnergyCtrl, but no enabled DVFSHandler found.
info: Entering event queue @ 0. Starting simulation...
warn: Replacement policy updates recently became the responsibility of SLICC state machines. Make sure to se
warn: Not doing anything for miscreg ACTLR
warn: Not doing anything for write of miscreg ACTLR
warn: The clidr register always reports 0 caches.
warn: clidr LoUIS field of 0b001 to match current ARM implementations.
warn: The cselr register isn't implemented.
warn: CP14 unimplemented crn[0], opc1[6], crm[0], opc2[0]
warn: CP14 unimplemented crn[0], opc1[6], crm[0], opc2[0]
warn: Cache maintenance operations are not supported in Ruby.
warn: instruction 'mcr icinvau' unimplemented
warn: instruction 'mcr bpiallis' unimplemented
warn: instruction 'mcr icialluis' unimplemented
4204647000: system.terminal: attach terminal 0
```

It took around 20,30 mins for the simulator to start. When the initialization process is complete, the client terminal will be able to execute commands.

➤ **root@genericarmv8:**

Here we need to enter the parsec-3.0 bin folder so that we can run the benchmark

- **cd parsec-3.0/bin**
- **./parsecmgmt -a build -p facesim**
- **./parsecmgmt -a run -p facesim**

On executing the above build and run commands, we observe this on the telnet terminal, which shows the simulation started as intended.

```
-sh: parsecmgmt: command not found
root@genericarmv8:~/parsec-3.0/bin#
root@genericarmv8:~/parsec-3.0/bin# ./parsecmgmt -a build -p facesim
./parsecmgmt -a build -p facesim
[PARSEC] Packages to build:  parsec.facesim

[PARSEC] [===== Building package parsec.facesim [1] =====]
[PARSEC] [----- Analyzing package parsec.facesim -----]
[PARSEC] Package parsec.facesim already exists, proceeding.
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Bienia. Benchmarking Modern Multiprocessors. Ph.D. Thesis, 2011.
[PARSEC]
[PARSEC] Done.
root@genericarmv8:~/parsec-3.0/bin#
root@genericarmv8:~/parsec-3.0/bin# ./parsecmgmt -a run -p facesim
./parsecmgmt -a run -p facesim
|
```

This screenshot shows ‘facesim’ benchmark running on gem5

```
[PARSEC] Done.
root@genericarmv8:~/parsec-3.0/bin#
root@genericarmv8:~/parsec-3.0/bin# ./parsecmgmt -a run -p facesim
./parsecmgmt -a run -p facesim
[PARSEC] Benchmarks to run:  parsec.facesim

[PARSEC] [===== Running benchmark parsec.facesim [1] =====]
[PARSEC] Deleting old run directory.
[PARSEC] Setting up run directory.
[PARSEC] No archive for input 'test' available, skipping input setup.
[PARSEC] Running 'time /home/root/parsec-3.0/bin/./pkgs/apps/facesim/inst/aarch64-linux.gcc/bin/facesim -h':
[PARSEC] [----- Beginning of output -----]
PARSEC Benchmark Suite Version 3.0-beta-20150206
Usage: /home/root/parsec-3.0/bin/./pkgs/apps/facesim/inst/aarch64-linux.gcc/bin/facesim [-restart <int>] [-lastframe <int>] [-threads <int>]
]
-restart    <no description available> (default 0)
-lastframe  <no description available> (default 300)
-threads    <no description available> (default 1)
-timing     <no description available>

real      0m0.008s
user      0m0.000s
sys       0m0.000s
[PARSEC] [----- End of output -----]
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Bienia. Benchmarking Modern Multiprocessors. Ph.D. Thesis, 2011.
[PARSEC]
[PARSEC] Done.
root@genericarmv8:~/parsec-3.0/bin#
root@genericarmv8:~/parsec-3.0/bin#
```


We tried running the benchmark using some input on gem5 also. We got the following expected results.

- **parsecmgmt -a run -p facesim -i simsmall**

```

Face_Data/Eftychis_840k/Front_370k/peak_isometric_stress.list.txt
Face_Data/Eftychis_840k/Front_370k/procerus_left.constitutive_data
Face_Data/Eftychis_840k/Front_370k/procerus_right.constitutive_data
Face_Data/Eftychis_840k/Front_370k/risorius_left.constitutive_data
Face_Data/Eftychis_840k/Front_370k/risorius_right.constitutive_data
Face_Data/Eftychis_840k/Front_370k/zygomatic_major_left.constitutive_data
Face_Data/Eftychis_840k/Front_370k/zygomatic_major_right.constitutive_data
Face_Data/Eftychis_840k/Front_370k/zygomatic_minor_left.constitutive_data
Face_Data/Eftychis_840k/Front_370k/zygomatic_minor_right.constitutive_data
Face_Data/Eftychis_840k/eftychis_cranium_collision_surface.phi
Face_Data/Eftychis_840k/eftychis_cranium_collision_surface.rgd
Face_Data/Eftychis_840k/eftychis_cranium_collision_surface.tri
Face_Data/Eftychis_840k/eftychis_cranium_collision_surface_smoothed.phi
Face_Data/Eftychis_840k/eftychis_cranium_collision_surface_smoothed.rgd
Face_Data/Eftychis_840k/eftychis_cranium_collision_surface_smoothed.tri
Face_Data/Eftychis_840k/eftychis_jaw_collision_surface.phi
Face_Data/Eftychis_840k/eftychis_jaw_collision_surface.rgd
Face_Data/Eftychis_840k/eftychis_jaw_collision_surface.tri
Face_Data/Eftychis_840k/eftychis_jaw_collision_surface_smoothed.phi
Face_Data/Eftychis_840k/eftychis_jaw_collision_surface_smoothed.rgd
Face_Data/Eftychis_840k/eftychis_jaw_collision_surface_smoothed.tri
Face_Data/Motion_Data/
Face_Data/Motion_Data/Storytelling_Controls/
Face_Data/Motion_Data/Storytelling_Controls/storytelling_controls.1
Face_Data/Motion_Data/Storytelling_Controls/storytelling_controls.2
[PARSEC] Running 'time /home/root/parsec-3.0/bin/./pkgs/apps/facesim/inst/aarch64-linux.gcc/bin/facesim -timing -threads 1':
[PARSEC] [----- Beginning of output -----]
PARSEC Benchmark Suite Version 3.0-beta-20150206
Creating directory using system("mkdir -p Storytelling/output")...Successful!
Simulation
Reading simulation model : ../Face_Data/Eftychis_840k/Front_370k/face_simulation_1.1
Total particles = 80598
Total tetrahedra = 372126
muscles = 32
attachments = 3

Frame 1

```

2.4. Challenges/Difficulties:

- 1) One challenge is cross compiling 'Facesim' benchmark with x86 or QEMU. Many people had trouble, I ensured that I was updating the config files correctly before compiling.
 - Config.sub and config.guess are the 2 files that need to be updated.
- 2) Another important issue occurred when we were trying to mount the expanded disk image so that Parsec benchmark could be added to it.

The issue occurred while executing the following command.

- `sudo mount -o loop,offset=$((($start_sector*$units)) expanded-linaro-minimal-aarch64.img disk_mnt`

Here the machine was not recognizing the command and was repeatedly throwing an error.

We resolved this issue by realizing the \$units command run before this command was not really returning any value and we had to give that value ourselves explicitly using the following command

- `sudo fdisk -l -u=sectors expanded-linaro-minimal-aarch64.img`

```
vision@kaliha-precision-workstation-t3500:~/full_system_images/disks$ sudo fdisk -l -u=sectors expanded-linaro-minimal-aarch64.img
Disk expanded-linaro-minimal-aarch64.img: 21 GiB, 22548316160 bytes, 44039680 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x20c7858d

Device                Boot Start      End  Sectors  Size Id Type
expanded-linaro-minimal-aarch64.img1  63 44039679 44039617  21G 83 Linux
vision@kaliha-precision-workstation-t3500:~/full_system_images/disks$
```

The command showed us the value of units which is 512.

We then ran the above command again, replacing the value

- `sudo mount -o loop,offset=$((($start_sector*512)) expanded-linaro-minimal-aarch64.img disk_mnt`

and this time it ran without error.

- 3) The last very challenging thing was getting the simulator working with 'Facesim' benchmark.

The usual commands as indicated on the github link and presentation on gem5 with parsec were not working.

```
./build/ARM/gem5.opt -d fs_results/<benchmark> configs/example/arm/starter_fs.py
--cpu="hpi" --num-cores=1 --disk-image=$M5_PATH/disks/expanded-linaro-minimal-aarch64.img
```


It said, when using `starter_fs.py`, we had to explicitly give more arguments like `-dtb` and `--kernel`.

We provided these 2 and saw the simulator initialized but it couldn't then run the parsec benchmark.

As a result, I tried a different approach and used `fs.py` config file to start gem5.

```
> ./build/ARM/gem5.opt -d fs_results/facesim configs/example/fs.py --disk-  
image=/home/vision/full_system_images/disks/expanded-linaro-minimal-aarch64.img --  
machine-type=VEpress_EMM64
```

The above command worked fine and enabled running the benchmark with and without inputs on gem5 simulator. This command initialized gem5 in 64 bit mode with the expanded disk image using `fs.py` file.