Priya Jeyaprakash

Professor. Mahmood

CSE 3100 - Section 001

13 December 2023

Extra Credit Final Report

When going through the instructions for the assignment and looking at the implementation that had already been completed, the Forward Passing functionality was much clearer to understand. Based on this fact and personally having a better understanding of how pipes worked, the submitted code/deliverable has the forward pass function implemented using pipes. Since the goal of parallelization was to compute half of the layers in the parent and the other half in the child, they're both running the same code, but in the child the first 50 layers are being processed and in the parent the second 50 layers are being processed. Once the code was compiled, it was running and had a core dump, to figure out where this valgrind was run and it seemed to be in the function call of TestSimpleNetwork() on line 234. To ensure the error wasn't somewhere else within the pipe/forking code, a copy of the code was made titled NeuralNetworkOriginal, any code not in the starter code file was commented out, and then it was run through valgrind. After this, there was still a memory leak present in the same place. Seeing as the code had a memory leak that I wasn't capable of fixing, I have attached screenshots of the valgrind tests and compilation in this summary. I am hoping that based off of this and the progress shown in my code I will be able to receive some of the extra credit.

```
[rbj21002@rbj21002-vm extra_credit]$ gcc -g -lm -o Original NeuralNetworkOriginal.c
[rbj21002@rbj21002-vm extra_credit]$ gcc -g -lm -o New NeuralNetwork.c
[rbj21002@rbj21002-vm extra_credit]$ ./Original
Segmentation fault (core dumped)
[rbj21002@rbj21002-vm extra_credit]$ ./New
Segmentation fault (core dumped)
```

```
[rbj21002@rbj21002-vm extra_credit]$ valgrind ./Original
```

Show Command Actions

Command executed 43 secs ago and failed (Exit Code 139)

```
by Julian Seward et al.
with -h for copyright info
==443806==
==443806== Invalid read of size 1
==443806==    at 0x4C3F117: strcmp (vg_replace_strmem.c:924)
==443806==    by 0x401257: main (NeuralNetworkOriginal.c:238)
==443806==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==443806==
==443806==
==443806== Process terminating with default action of signal 11 (SIGSEGV)
==443806==  Access not within mapped region at address 0x0
==443806==    at 0x4C3F117: strcmp (vg_replace_strmem.c:924)
==443806==    by 0x401257: main (NeuralNetworkOriginal.c:238)
==443806==  If you believe this happened as a result of a stack
==443806==  overflow in your program's main thread (unlikely but
==443806==  possible), you can try to increase the size of the
==443806==  main thread stack using the --main-stacksize= flag.
==443806==  The main thread stack size used in this run was 8388608.
==443806==
==443806== HEAP SUMMARY:
==443806==     in use at exit: 0 bytes in 0 blocks
==443806==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==443806==
==443806== All heap blocks were freed -- no leaks are possible
==443806==
==443806== For lists of detected and suppressed errors, rerun with: -s
==443806== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
```

```
[rbj21002@rbj21002-vm extra_credit]$ valgrind ./New
==443850== Memcheck, a memory error detector
==443850== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==443850== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==443850== Command: ./New
==443850==
==443850== Invalid read of size 1
==443850==    at 0x4C3F117: strcmp (vg_replace_strmem.c:924)
==443850==    by 0x401257: main (NeuralNetwork.c:238)
==443850==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==443850==
==443850==
==443850== Process terminating with default action of signal 11 (SIGSEGV)
==443850==  Access not within mapped region at address 0x0
==443850==    at 0x4C3F117: strcmp (vg_replace_strmem.c:924)
==443850==    by 0x401257: main (NeuralNetwork.c:238)
==443850==  If you believe this happened as a result of a stack
==443850==  overflow in your program's main thread (unlikely but
==443850==  possible), you can try to increase the size of the
==443850==  main thread stack using the --main-stacksize= flag.
==443850==  The main thread stack size used in this run was 8388608.
==443850==
==443850== HEAP SUMMARY:
==443850==     in use at exit: 0 bytes in 0 blocks
==443850==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==443850==
==443850== All heap blocks were freed -- no leaks are possible
==443850==
==443850== For lists of detected and suppressed errors, rerun with: -s
==443850== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
[rbj21002@rbj21002-vm extra_credit]$
```