Università della Svizzera italiana

Institute of Computational Science ICS

Debugging

valgrind and -fsanitize=address

Radim Janalík, Juraj Kardoš

Università della Svizzera italiana

October 17, 2018

Segmentation fault

- You can often see that your program crashes with following errors
  - Segmentation fault
  - SIGSEGV
  - signal 11
- This indicates problem related to memory
  - Often (but not always) you access not initialized memory

Segmentation fault

- In git repository we prepared a simple code with bug for you

```
$ ssh icsmaster
$ cd HPC_2018
$ git pull
$ cd Debugging
```

- Compile and run the program

```
$ module load gcc/6.1.0
$ make
$ salloc
$ ./dotProduct
```

- You can see error message like this

```
Segmentation fault (core dumped)
```

Valgrind

- Valgrind is very good tool for debugging problems related to memory
  - Accessing not allocated memory
  - Memory leaks
  - ...
- To use valgrind, compile with debugging information (flag -g)
- Then you can use valgrind

```
$ valgrind ./your_app
```

- You can also add more parameters to valgrind to see more info

```
$ valgrind --tool=memcheck -v --leak-check=yes --show-reachable=↩
    yes ./your_app
```

Valgrind

- Valgrind prints output like this

```
==28793== Invalid read of size 8
==28793==    at 0x400A06: main (dotProduct.cpp:31)
==28793==  Address 0x5aa1fa0 is 0 bytes after a block of size ↩
    800 alloc'd
==28793==    at 0x4C2A7AA: operator new[](unsigned long) (in /↩
    usr/lib64/valgrind/vgpreload_memcheck-amd64-linux.so)
==28793==    by 0x400969: main (dotProduct.cpp:17)
```

- You can see that the program reads not allocated memory in file `dotProduct.cpp` on line 31

```
29    for(int i=0; i< NMAX_ERR; i ++)
30    {
31       dotProduct += a[i] * b[i];
32    }
```

- As the problem is in loop body, it is likely that the condition in for loop is wrong

fsanitize

- The output of valgrind is not very nice and it requires some practice to find what you are looking for
- If your code is written in C++, you have another option
- Add flags `-g` and `-fsanitize=address` when you compile your app
- Then run your app

```
$ ./your_app
```

fsanitize

- When your app crashes, it will print info like this

```
==13942==ERROR: AddressSanitizer: heap-buffer-overflow on ↩
    address 0x61800000ffa0 at pc 0x000000400e61 bp 0↩
    x7fff3db4b2b0 sp 0x7fff3db4b2a8
READ of size 8 at 0x61800000ffa0 thread T0
      #0 0x400e60 in main /home/janalik/HPC_2018/Debugging/↩
          dotProduct.cpp:31
    #1 0x7fd4e0f68af4 in __libc_start_main (/usr/lib64/libc.so↩
        .6+0x21af4)
    #2 0x400c78  (/cluster_nfs/Data_Apps/home/janalik/HPC_2018/↩
        Debugging/dotProduct+0x400c78)

0x61800000ffa0 is located 0 bytes to the right of 800-byte ↩
    region [0x61800000fc80,0x61800000ffa0)
```

- It gives you the same info as valgrind and more, but it is easier to read