OAK RIDGE National Laboratory

Debugging with GDB and Rocgdb

Elijah MacCarthy

Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory

February 02, 2023

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Intro to debuggers

- A Debugger helps in debugging and analyzing applications.
- It makes it possible to detect what is happening during program execution.
- Thus, helps in investigating improper program behavior, crash during execution and finding logical errors.
- These are referred to as bugs and can sometimes be difficult to diagnose
- It is argued that crashing bugs are sometimes the easiest kind to solve
- Whereas logical bugs can be the hardest to identify.

Debuggers

- Crashing bugs generate signals which can help in tracing the bug.
- Some of these crashing bug signals include
  - SIGSEGV: this is segmentation fault where you attempted to access memory outside the virtual address space
  - SIGABT: here the program or a library it uses realized something wrong and crashed intentionally

Debugging with gdb

- gdb is the gnu debugger and it supports debugging C, C++, Fortran, Java programs, among others.
- It gives the programmer control over the program execution as the programmer is able to:
  - pause or stop the program by specifying some conditions during the active debugging session
  - Modify values of variables and assess it's influence on the program behavior.

```
[maccarthy@login1.crusher ~]$ cat code.cpp

#include <iostream>
using namespace std;

int main()
{
        int balance=100;
        int target=1000;
        float rate = 0.1;
        int year = 0;
        do
        {
                float interest = balance * rate;
                balance = balance + interest;
                year++;
        } while ( balance <= target );

        printf("%d No. of years to achieve target balance.\n", year);

        return 0;
}
```

GDB Demo

Starting the gdb debugger

- -g flag adds debugging symbols to compiled applications to enhance debugging

```
[maccarthy@login1.crusher ~]$ g++ -g  code.cpp
[maccarthy@login1.crusher ~]$ gdb ./a.out
GNU gdb (GDB; SUSE Linux Enterprise 15) 11.1
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-suse-linux".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://bugs.opensuse.org/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb)
```

GDB Demo

Setting break points and stepping into, over and out

- Step (s), step into: step into any function calls on a line.
- next (n), step over: this continues to the next line in the current frame
- Finish (fin), Step out; complete the current function call and return

```
(gdb) b main
Breakpoint 1 at 0x4004ee: file code.cpp, line 5.
(gdb) run
Starting program: /autofs/nccs-svm1_home2/maccarthy/a.out

Breakpoint 1, main () at code.cpp:5
5                 int balance=100;
(gdb) p balance
$1 = 0
(gdb) s
6                 int target=1000;
(gdb) s
7                 float rate = 0.1;
(gdb) s
8                 int year = 0;
(gdb) s
11                        float interest = balance * rate;
(gdb) s
12                        balance = balance + interest;
(gdb) s
13                        year++;
(gdb) p balance
$2 = 110
(gdb)
```

GDB Demo

Setting break points and stepping into, over and out

- We can pause at specific lines (break) and when certain expressions are true
- Conditional breakpoints are useful for breaking in loops or frequently called functions.
- Watchpoints pauses when the value of an expression changes. Prints old and new values.

```
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) br main
Breakpoint 1 at 0x4004ee: file code.cpp, line 5.
(gdb) r
Starting program: /autofs/nccs-svm1_home2/maccarthy/a.out
Missing separate debuginfos, use: zypper install glibc-debuginfo-2.31-150300.37
1.x86_64

Breakpoint 1, main () at code.cpp:5
5               int balance=100;
(gdb) watch balance
Hardware watchpoint 2: balance
(gdb) s

Hardware watchpoint 2: balance

Old value = 0
New value = 100
main () at code.cpp:6
6               int target=1000;
(gdb)
```

GDB Demo

Setting break points and stepping into, over and out

- Setting conditional break points

```
[maccarthy@login1.crusher ~]$ gdb ./a.out
GNU gdb (GDB; SUSE Linux Enterprise 15) 11.1
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-suse-linux".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://bugs.opensuse.org/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) br 12 if balance==10
Breakpoint 1 at 0x400527: file code.cpp, line 12.
(gdb) r
Starting program: /autofs/nccs-svm1_home2/maccarthy/a.out
Missing separate debuginfos, use: zypper install glibc-debuginfo-2.31-150300.37.
1.x86_64
1 No. of years to achieve target balance.
[Inferior 1 (process 71479) exited normally]
```

GDB Demo

Setting break points and stepping into, over and out

- Analyzing and detecting bugs. While loop condition should be balance <= target

```
Starting program: /autofs/nccs-svm1_home2/maccarthy/a.out
Missing separate debuginfos, use: zypper install glibc-debuginfo-2.31-150300.37.
1.x86_64

Breakpoint 1, main () at code.cpp:5
5               int balance=100;
(gdb) c
Continuing.

Breakpoint 2, main () at code.cpp:11
11                      float interest = balance * rate;
(gdb) c
Continuing.

Breakpoint 3, main () at code.cpp:14
14              } while ( balance >= target );
(gdb) info local
balance = 110
target = 1000
rate = 0.100000001
year = 1
(gdb) c
Continuing.

Breakpoint 4, main () at code.cpp:15
15              printf("%d No. of years to achieve target balance.\n", year);
(gdb) c
Continuing.
1 No. of years to achieve target balance.
```

GDB Demo

Setting break points and stepping into, over and out

- Program behaves as expected once bug identified and fixed

```
[maccarthy@login1.crusher ~]$ gdb ./a.out
GNU gdb (GDB; SUSE Linux Enterprise 15) 11.1
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-suse-linux".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://bugs.opensuse.org/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) r
Starting program: /autofs/nccs-svm1_home2/maccarthy/a.out
Missing separate debuginfos, use: zypper install glibc-debuginfo-2.31-150300.37.1.x86_64
25 No. of years to achieve target balance.
[Inferior 1 (process 20021) exited normally]
(gdb)
```

Debugging with rocgdb

- Rocgdb is the rocm debugger used in debugging and analyzing applications.
- This is made available through rocm and it works similar to gdb.
- It however has significant enhancements compared to gdb for debugging on AMD GPUs.
- It has a non-stop mode which allows debugging across both CPU and GPU.
- It is however not multi-process, hence similar to gdb.

Rocgdb Demo

- Rocgdb is similar to gdb

```
maccarthy@crusher120:~/Test-hip/hello_jobstep> rocgdb ./hello_jobstep
GNU gdb (rocm-rel-5.1-36) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://github.com/ROCm-Developer-Tools/ROCgdb/issues>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./hello_jobstep...
(gdb)
```

Rocgdb Demo

- Just as gdb, we are able to set break points, watch variables, backtrace and print

```
Reading symbols from ./hello_jobstep...
(gdb) br main
Breakpoint 1 at 0x202d19: file hello_jobstep.cpp, line 30.
(gdb) run
Starting program: /autofs/nccs-svm1_home2/maccarthy/Test-hip/hello_jobstep/hello_jobstep
warning: File "/opt/cray/pe/gcc/11.2.0/snos/lib64/libstdc++.so.6.0.29-gdb.py" auto-loading has been declined by your `auto-load safe-path' set to "$debugdir:
$datadir/auto-load".
To enable execution of this file add
        add-auto-load-safe-path /opt/cray/pe/gcc/11.2.0/snos/lib64/libstdc++.so.6.0.29-gdb.py
line to your configuration file "/ccs/home/maccarthy/.config/gdb/gdbinit".
To completely disable this security protection add
        set auto-load safe-path /
line to your configuration file "/ccs/home/maccarthy/.config/gdb/gdbinit".
For more information about this security protection see the
"Auto-loading safe path" section in the GDB manual.  E.g., run from the shell:
        info "(gdb)Auto-loading safe path"
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

Breakpoint 1, main (argc=1, argv=0x7fffffff6dc8) at hello_jobstep.cpp:30
warning: Source file is more recent than executable.
30              MPI_Init(&argc, &argv);
(gdb) c
Continuing.
[New Thread 0x7fffd95df700 (LWP 42748)]
[New Thread 0x7ffeb7fff700 (LWP 42750)]
MPI 000 - OMP 000 - HWT 028 - Node crusher120 - RT_GPU_ID 0,1,2,3,4,5,6,7 - GPU_ID 0,1,2,3,4,5,6,7 - Bus_ID c1,c6,c9,ce,d1,d6,d9,de
[Thread 0x7ffeb7fff700 (LWP 42750) exited]
[Thread 0x7fffd95df700 (LWP 42748) exited]
```

Rocgdb Demo

- Since rocgdb debugs both CPU and GPU regions of code, we can step into GPU regions to debug.

```
(gdb) s
53              int num_devices = 0;
(gdb) s
54          hipErrorCheck( hipGetDeviceCount(&num_devices) );
(gdb) s
57              int thread_id = 0;
(gdb) s
59              if(num_devices == 0){
(gdb) s
74              std::string busid_list = "";
(gdb) info local
busid = "\000\000\000\000\000\000\000\000@M \000\000\000\000\000\fw\215\341hc\344n\fw?T\334L\344n", '\000' <repeats 20 times>, "\377\177\000\000\002\000\000\
000\000\000\000"
busid_list = {static npos = 18446744073709551615,
  _M_dataplus = {<std::allocator<char>> = {<__gnu_cxx::new_allocator<char>> = {<No data fields>}, <No data fields>},
    _M_p = 0xfffffffffffff730 <error: Cannot access memory at address 0xfffffffffffff730>}, _M_string_length = 10456123407733716748, {
    _M_local_buf = "\240\211 \000\000\000\000\000(n\377\377\377\177\000", _M_allocated_capacity = 2132384}}
rt_gpu_id_list = {static npos = 18446744073709551615,
  _M_dataplus = {<std::allocator<char>> = {<__gnu_cxx::new_allocator<char>> = {<No data fields>}, <No data fields>}, _M_p = 0x0},
  _M_string_length = 140737488317760, {_M_local_buf = "^\263q\354\000\000\000\000Pm\377\377\377\177\000", _M_allocated_capacity = 3966874462}}
size = 1
rank = 0
name = "crusher121\000\350\377\177\000\000\000\000\000\000\000\000\000\000\001\000\000\000\000\000\000\000\3201\377\377\377\177\000\000\001\000\000\000\000\0
00\000\000\3201\377\377\377\177\000\000 M \000\000\000\000\000\001\000\000\000\000\000\000\000\266\065 \000\000\000\000\000\3401\377\377\377\177\000\000\311\
065 \000\000\000\000\000\002\000\000\000\000\000\000\000=: ", '\000' <repeats 20 times>
resultlength = 10
gpu_id_list = 0x7fffffff9a61 "0,1,2,3,4,5,6,7"
rocr_visible_devices = 0x7fffffff9a61 "0,1,2,3,4,5,6,7"
num_devices = 8
hwthread = 0
thread_id = 0
```