



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



# Summer School 2015

MPI Debugging

<http://github.com/eth-cscs/SummerSchool2015/wiki> - DAY3

July 2015

# Summary

- Debugging 02.MPI\_pt2pt/C/6.bugs\_buffer.c
  - with gdb
  - with ddt



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich

## 02.MPI\_pt2pt/C/6.bugs\_buffer.c

---

## 02.MPI\_pt2pt/C/6.bugs\_buffer.c

### The problem

- When ran with  $\geq 11$  MPI tasks, the program crashes.

### The crash

```
course51@daint103:~ aprun -n16 6.exe
Hello! This rank 1 has the data computed with root 1.0000.
Hello! This rank 2 has the data computed with root 1.4142.
...
Hello! This rank 15 has the data computed with root 3.8730

_pmiu_daemon(SIGCHLD): [NID 00013] [c0-0c0s3n1]
[Tue Jul 21 11:14:52 2015] PE RANK 10 exit signal Aborted
[NID 00013] 2015-07-21 11:14:52
Apid 166999: initiated application termination
Application 166999 exit codes: 134
Application 166999 exit signals: Killed
Command exited with non-zero status 137
```

### The error message

```
Rank 13 [Tue Jul 21 11:14:52 2015] [c0-0c0s3n1]
Fatal error in MPI_Recv: Message truncated, error stack:
MPI_Recv(199): MPI_Recv(buf=0xab0700, count=58, MPI_CHAR,
                  src=0, tag=1, MPI_COMM_WORLD, status=0x1) failed
MPIDI_CH3U_Receive_data_found(144):
Message from rank 0 and tag 1 truncated; 59 bytes received
but buffer size is 58
```

# Debugging with gdb

## Compiling with debug flags

```
GNU:    CC -O0 -g                               6.bugs\_buffer.c -o 6.exe
INTEL:  CC -O0 -debug all -g                     6.bugs\_buffer.c -o 6.exe
PGI:    CC -O0 -dynamic -traceback -g            6.bugs\_buffer.c -o 6.exe
CCE:    CC -G0                                   6.bugs\_buffer.c -o 6.exe
```

## Crashing will dump the corefile

```
course51@daint103:~ aprun -n16 6.exe
Command exited with non-zero status 137
course51@daint103:~ file core
core: ELF 64-bit LSB core file x86-64 from '6.exe'
```

## Reading the corefile with gdb

```
course51@daint103:~ gdb -c core -e 6.exe
% GNU gdb (GDB) 7.6.2-5.0.1-42607
% Program terminated with signal 6, Aborted.
% (gdb) bt
#1 0x00002aaaaacefee61 in abort from /lib64/libc.so.6
#2 0x00002aaaaab095c36 in MPID_Abort from libmpich_pgi.so.3
#3 0x00002aaaaab0461a2 in handleFatalError from libmpich_pgi.so.3
#4 0x00002aaaaab04635f in MPIR_Err_return_comm from libmpich_pgi.so.3
#5 0x00002aaaaafe34a1 in PMPI_Recv from libmpich_pgi.so.3
#6 0x00000000000402b60 in ?? ***
#7 0x00000000000000001 in ?? ***
% (gdb) quit
```

# Debugging with ddt (demo)

The screenshot displays the Alinea DDT - Alinea Forge 5.0.1-42607 interface. The main window shows a C code file named `6 bugs_buffer_solution.c`. The code includes a `main` function that initializes a buffer and a `recv_size` variable. It then enters a loop where it receives data from a remote process (rank 10) and prints the received size. The code is as follows:

```
size = (unsigned long*)malloc((np-1)*sizeof(unsigned long));  
  
/* Non-blocking send buffer */  
for( k = 1; k < np; k++ ) {  
    root[k-1] = sqrt(k);  
    sprintf(buffer, "%d\\! This rank %d has the data computed with root %.4f ", k, root[k-1];  
    MPI_Isend(buffer, strlen(buffer), MPI_CHAR, k, 1, MPI_COMM_WORLD, &req[k-1]);  
    size[k-1] = (unsigned long)(MAX_SIZE_COMPUTE*root[k-1]);  
    MPI_Isend(&size[k-1], 1, MPI_UNSIGNED_LONG, k, 2, MPI_COMM_WORLD, &req[k-1+np-1]);  
}  
  
for( k = 1; k < np; k++ ) {  
    compute_and_send_size(k-1, root[k-1], k, &req[k-1+2*(np-1)]);  
}  
  
MPI_Waitall((3*(np-1), req, MPI_STATUS_IGNORE);  
free(req);  
free(buffer);  
}  
  
/* EASY BUG:  
 * With more than 9 ranks the string buffer as a size of MAX_SIZE_STRING+1,  
 * because the number of ranks require 2 characters.  
 */  
  
// receive string  
MPI_Recv(buffer, MAX_SIZE_STRING, MPI_CHAR, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);  
printf("%s\\n", buffer);  
  
// receive size  
MPI_Recv(&recv_size, 1, MPI_UNSIGNED_LONG, 0, 2, MPI_COMM_WORLD, MPI_STATUS_IGNORE);  
printf("rank %d received size of %d\\n", my_rank, recv_size);  
return 0; }  

```

The interface also shows a `Stacks (All)` window with the following data:

Processes	Threads	Function
1	1	main 6 bugs_buffer_solution.c:54
2	1	main 6 bugs_buffer_solution.c:54
3	1	main 6 bugs_buffer_solution.c:54
4	1	main 6 bugs_buffer_solution.c:54
5	1	main 6 bugs_buffer_solution.c:54
6	1	main 6 bugs_buffer_solution.c:54
7	1	main 6 bugs_buffer_solution.c:54
8	1	main 6 bugs_buffer_solution.c:54
9	1	main 6 bugs_buffer_solution.c:54
10	1	main 6 bugs_buffer_solution.c:54
11	1	main 6 bugs_buffer_solution.c:54

The `Locals` window shows the following data:

Variable Name	Value
argc	1
argv	0x7fffffffbad
buffer	0x7fffffffbad "0\\! This rank 10 has the data computed with
data	0x7fffffffbad
my_rank	10
np	12
recv_size	14073748037368
req	0x0
root	0x0
size	0x7fffffffbad

The `Input/Output` window shows the following data:

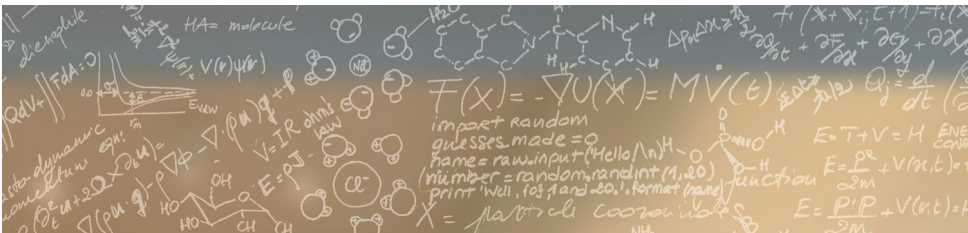
Variable Name	Value
buffer	0x7fffffffbad "0\\! This rank 10 has the data computed with root 3.1622"



**CSCS**

Centro Svizzero di Calcolo Scientifico  
Swiss National Supercomputing Centre

**ETH** zürich



Thank you for your attention.