



# ARM Forge DDT

# Quick intro to Debugging with Forge DDT

# Presented by Will Castillo

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



**U.S. DEPARTMENT OF  
ENERGY**

# Intro



- ARM Forge DDT (Distributed Debugging Tool)
- Commercial debugging tool originally developed by Allinea Software company, 2002
- Graphical interface to debug serial or highly parallelized codes within HPC
- As of 2016 DDT was used on 20 of the 25 fastest supercomputers in the world

# Paradigms



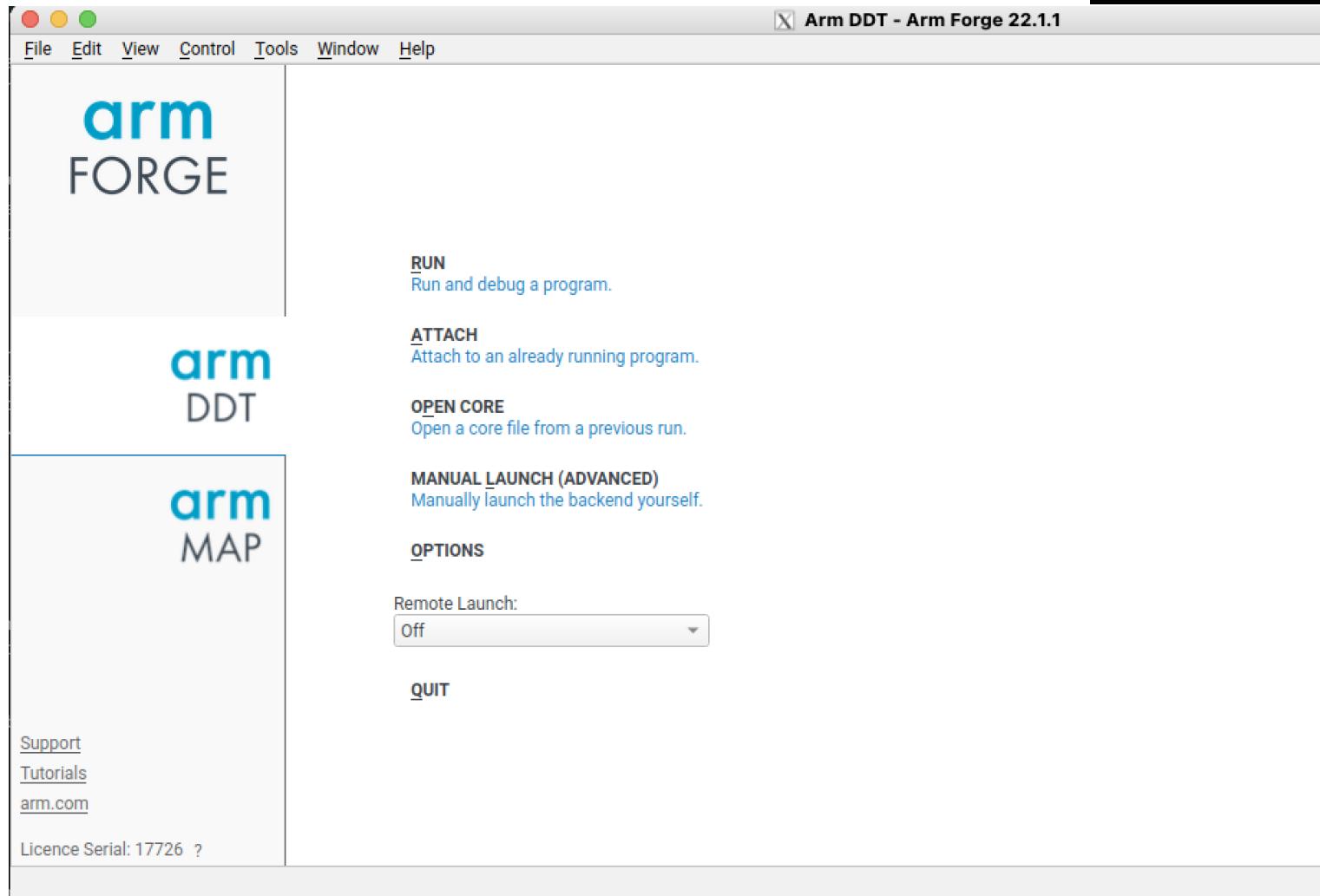
- Supports single and multithreaded processes
- OpenMP
- MPI
- Heterogenous software (GPU software)
- Hybrid codes e.g. MPI with OpenMP or MPI with CUDA

# Language Support



- C
- C++
- All flavors of Fortran, including f90
- Python (limited)
- GPU languages (CUDA, hipcc )

# A look into...



# Connecting



- Backend connects to all ranks

The screenshot shows the Arm DDT interface with the following details:

- Call Graph:** A tree diagram at the bottom left shows the execution flow. The root node is `slurmstepd`. It branches into two nodes:
  - The first node is `16*[vAdd_mpi—2*[{vAdd_mpi}]]`, which further branches into `4*[{slurmstepd}]`.
  - The second node is `slurmstepd—forge-backend—15*[forge-backend—gdb]`, which further branches into `forge-treeserve` and `gdb`, which then branches into `4*[{slurmstepd}]`.
- Code Editor:** The main window displays C code for `vAdd_mpi.cpp`. The code initializes MPI, gets processor information, and handles memory allocation for matrices A, B, and C.
- Locals View:** On the right, the "Locals" tab shows a variable `size` with a value of `0`.
- Processes View:** At the bottom left, the "Processes" tab lists several processes, including `main`, `ofi_uffd`, `poll`, and `???`, each with 16 threads.

# Functionality



- Control many processes of a program

The screenshot shows the Arm DDT debugger interface. At the top, there's a toolbar with various icons. Below it is a menu bar with 'File', 'Edit', 'Run', 'Breakpoints', 'Watchpoints', 'Stacks', 'Tracepoints', 'Input/Output', 'Help', and 'About'. A status bar at the bottom right says 'Ready Connected to: (via tunnel) login1:4201 -> login1'.

The main window has several panes:

- Current Group:** Shows 'All' with 16 ranks numbered 0 to 15. Rank 0 is highlighted with a red box.
- Create Group:** Buttons for 'Project Files' and 'Fortran Modules'.
- Search (⌘K):** A search bar.
- Application Code:** A tree view showing 'Sources' containing 'vAdd\_mpi.cpp'.
- Code Editor:** Displays the C++ code for 'vAdd\_mpi.cpp':#include <stdio.h>
#include <math.h>
#include <stdint.h>
#include <sched.h>
#include <mpi.h>

int main(int argc, char \*argv[]) {
 /\* MPI initialization \*/
 MPI\_Init(&argc, &argv);
 int size;
 MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

 int rank;
 MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

 char name[MPI\_MAX\_PROCESSOR\_NAME];
 int result\_length;
 MPI\_Get\_processor\_name(name, &result\_length);

 int hwthread = sched\_getcpu();

 long long int N = 32\*1024\*1024;
 size\_t buffer\_size = N \* sizeof(double);

 double \*A = NULL;
 double \*B = NULL;
 double \*C = NULL;

 if (rank == 0){
 A = (double\*)malloc(buffer\_size);
 }
}
- Locals:** Shows a table with 'size' set to 0.
- Stacks (All):** Shows a list of threads and their functions:

Processes	Threads	Function
16	16	main (vAdd_mpi.cpp:14)
16	16	ofi_uffd_handler (util_mem_monitor.c:488)
16	16	> poll (poll2.h:46)
16	16	> ??

These are all the MPI ranks that were requested for the job step. You can actually move through your program updating all of your ranks or select a single rank to move through

# Functionality



- Allows the user to step through a program

The screenshot shows the Arm DDT - Arm Forge 22.1.2 interface. The top bar includes standard OS controls, a menu, and a toolbar with several icons. Below the toolbar is a control panel with buttons for 'Current Group' (All), 'Focus on current:' (Group, Process, Thread), and 'Step Threads Together'. A status bar at the bottom shows 'Processes: 16' and 'Threads: 16'.

The main area consists of three panes:

- Source Code:** Displays the file `vAdd_mpi.cpp` with the following code:

```
#include <stdio.h>
#include <math.h>
#include <stdint.h>
#include <sched.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    /* MPI initialization ----- */
    MPI_Init(&argc, &argv);
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    char name[MPI_MAX_PROCESSOR_NAME];
    int result_length;
    MPI_Get_processor_name(name, &result_length);

    int hwthread = sched_getcpu();

    long long int N      = 32*1024*1024;
    size_t buffer_size = N * sizeof(double);

    double *A = NULL;
    double *B = NULL;
    double *C = NULL;

    if (rank == 0) {
        A = (double*)malloc(buffer_size);
    }
}
```
- Locals:** Shows a table with one entry: `size` with value `0`.
- Stacks:** Shows a list of threads and their stack frames. The first frame is expanded to show the call chain: `main (vAdd_mpi.cpp:14)`, `ofi_uffd_handler (util_mem_monitor.c:488)`, `poll (poll2.h:46)`, and `???`.

**Step Into** – Allows for stepping into the next line or first line of a function

**Step Over** – Moves to next line in source code, stepping over function calls

**Step Out** - step out of a function call

# Functionality



- Step into Functions

Code snippet from vAdd\_mpi...:

```
#include <stdio.h>
#include <math.h>
#include <stdint.h>
#include <sched.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    /* MPI initialization ----- */
    MPI_Init(&argc, &argv);

    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    char name[MPI_MAX_PROCESSOR_NAME];
    int result_length;
    MPI_Get_processor_name(name, &result_length);

    int hwthread = sched_getcpu();

    long long int N      = 32*1024*1024;
    size_t buffer_size = N * sizeof(double);

    double *A = NULL;
    double *B = NULL;
    double *C = NULL;

    if (rank == 0) {
        A = (double*)malloc(buffer_size);
    }
}
```

**Stepping into a Function** – allows you to see how the program is executing the function and if any errors are occurring within the function call

# Functionality



- Setting watchpoints

The screenshot shows the Arm DDT interface. In the center, a modal dialog titled "Add Watchpoint" is open. It has fields for "Process Group" (set to "All"), "Process" (set to "All"), "Expression" (empty), "Language" (set to "Auto"), and "Trigger On" (set to "Write only"). A warning message at the bottom says "Please enter an expression." Below the dialog, the code editor shows a C++ file named "vAdd\_mpi.cpp". The code defines a function main that initializes variables A, B, and C to NULL, then performs some operations. At the bottom of the interface, a toolbar has "Watchpoints" selected, and a status bar indicates "Watchpoints" is active.

**Watchpoints –** anytime a variable or expression you supply changes, DDT will stop for you to analyze your code and stack trace

# Functionality



- Breakpoints

The screenshot shows the Arm DDT - Arm Forge 22.1.2 interface. On the left, the Project Files panel shows a project structure with 'Application Code' and 'Sources' containing 'vAdd\_mpi.cpp'. The code editor displays C++ code for matrix multiplication, with a breakpoint set at line 52. The variable viewer on the right shows the variable 'A' with a value of 0x0. The bottom navigation bar highlights the 'Breakpoints' tab, which lists the current breakpoint at line 52 of 'vAdd\_mpi.cpp'.

```
double *A = NULL;
double *B = NULL;
double *C = NULL;

if (rank == 0){

    A = (double*)malloc(buffer_size);
    B = (double*)malloc(buffer_size);
    C = (double*)malloc(buffer_size);

    for(int i=0; i<N; i++){

        double random_value = (double)rand()/(double)RAND_MAX;
        A[i] = sin(random_value) * sin(random_value);
        B[i] = cos(random_value) * cos(random_value);
        C[i] = 0.0;

    }
}

double tolerance = 1.0e-14;

long long int chunk_size = 0;

if( (N % size) != 0){
    printf("N must be evenly divisible by size. Exiting...\n");
    MPI_Finalize();
    exit(-1);
}
else{
    chunk_size = N / size;
}

size_t sub_buffer_size = chunk_size * sizeof(double);
```

Processes	Threads	File	Line	Function	Condition	Start After	Trigger Every	Stop
All	all	vAdd_mpi.cpp	52	main(int,char**)		0	1	

**Setting a breakpoint** – allows you to run your code up until a specified point and then DDT will stop your program from executing so you can examine the stack trace and variables at that point

# Functionality



- Tracepoints

The screenshot shows the Arm DDT - Arm Forge 22.1.2 interface. On the left, the code editor displays a C++ file named vAdd\_mpi.cpp. The code initializes MPI, allocates memory for arrays A, B, and C, and performs matrix multiplication. A tracepoint is set at line 25. On the right, the debugger's locals window shows the variable N with a value of 140737173765180. At the bottom, the Tracepoint Output window shows the message "line 25 16,ranks 0-15 size: 16 MPI\_COMM\_WORLD: <No symbol \"MPI\_COMM\_WORLD\" in current context.>".

**Setting a Tracepoint** – allows you to run your code without stopping and record a variable, function or line within source code every time that point in execution is reached or specified condition is met

# Conclusion

- Powerful debugger with graphical interface
- Multiple ways of connecting your program to DDT
- Start/stop features are critical when debugging codes at scale
- A competing tool is called TotalView

# Demo

# Initial connect

The screenshot shows the Arm DDT debugger interface. The main window displays a C file named `mpi_bug1.c`. A warning message at the top of the code editor states: "mpi\_bug1.PLEASE recompile then restart your debugging session." The code itself contains MPI initialization and communication logic. The left sidebar shows the project structure with `Application Code` containing `mpi_bug1.c`. The bottom left shows a stack trace with three frames:

Processes	Threads	Function
2	2	main (mpi_bug1.c:20)
2	2	ofi_uffd_handler (util_mem_monitor.c:48)
2	2	> poll (poll2.h:46)

The right side of the interface shows the Locals and Current Line(s) panes, both currently displaying the variable `rank` with the value `2104240`.

```
1 > /*****************************************************************************  
2 #include "mpi.h"  
3 #include <stdio.h>  
4 #include <stdlib.h>  
5  
6 int main (int argc, char *argv[]){  
7 {  
8 int numtasks, rank, dest, tag, source, rc ,count;  
9 char inmsg, outmsg='x';  
10 MPI_Status Stat;  
11  
12 MPI_Init(&argc,&argv);  
13 MPI_Comm_size(MPI_COMM_WORLD, &numtasks);  
14 MPI_Comm_rank(MPI_COMM_WORLD, &rank);  
15 printf("Task %d starting...\n",rank);  
16  
17 if (rank == 0) {  
18 dest = rank + 1;  
19 source = dest;  
20 tag = rank;  
21 MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);  
22 printf("Sent to task %d...\n",dest);  
23 }  
24 else if (rank == 1) {  
25 dest = rank - 1;  
26 source = dest;  
27 tag = rank;  
28 MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);  
29 printf("Received from task %d...\n",source);  
30 }  
31 }
```

# Step into – local variables Updated

The screenshot shows the Arm DDT - Arm Forge 22.1.2 debugger interface. The main window displays the source code for `mpi_bug1.c`:

```
12 int main (int argc, char *argv[])
13 {
14     int numtasks, rank, dest, tag, source, rc ,count;
15     char inmsg, outmsg='x';
16     MPI_Status Stat;
17
18     MPI_Init(&argc,&argv);
19     MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
20     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
21     printf("Task %d starting...\n",rank);
22
23     if (rank == 0) {
24         dest = rank + 1;
25         source = dest;
26         tag = rank;
27         MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
28         printf("Sent to task %d...\n",dest);
29     }
30
31     else if (rank == 1) {
32         dest = rank - 1;
33         source = dest;
34         tag = rank;
35         MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
36         printf("Received from task %d...\n",source);
37     }
38
39
40
41     MPI_Finalize();
42 }
```

A warning message is displayed at the top: "mpi\_bug1. Please recompile then restart your debugging session." A "Dismiss" button is located next to it.

The right side of the interface shows the "Locals" pane, which lists the current values of local variables:

Name	Value
argc	1
> argv	0x7fffff6d58
numtasks	2
rank	0
dest	1
tag	0
source	1
rc	0
count	0
inmsg	0'\000'
outmsg	120 'x'
> Stat	

The bottom left shows the "Stacks (All)" tab, which displays the current stack frames:

Processes	Threads	Function
1	1	main (mpi_bug1.c:29)
1	1	main (mpi_bug1.c:39)
2	2	ofi_uffd_handler (util_mem_monitor.c:48)
2	2	> poll (poll2.h:46)

# Step into – Program is Hanging

The screenshot shows the Arm DDT - Arm Forge 22.1.2 debugger interface. The main window displays a Fortran source code file named `mpi_bug1.c`. The code implements a MPI application with two processes. Process 0 sends a message to Process 1, and Process 1 receives it. The code includes MPI initialization, communication setup, and finalization. A warning message at the top of the code editor states: "mpi\_bug1.PLEASE recompile then restart your debugging session." The code editor has syntax highlighting for MPI functions like `MPI_Init`, `MPI_Comm_size`, `MPI_Comm_rank`, `MPI_Send`, `MPI_Recv`, and `MPI_Finalize`.

```
int main (int argc, char *argv[])
{
    int numtasks, rank, dest, tag, source, rc ,count;
    char inmsg, outmsg='x';
    MPI_Status Stat;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    printf("Task %d starting...\n",rank);

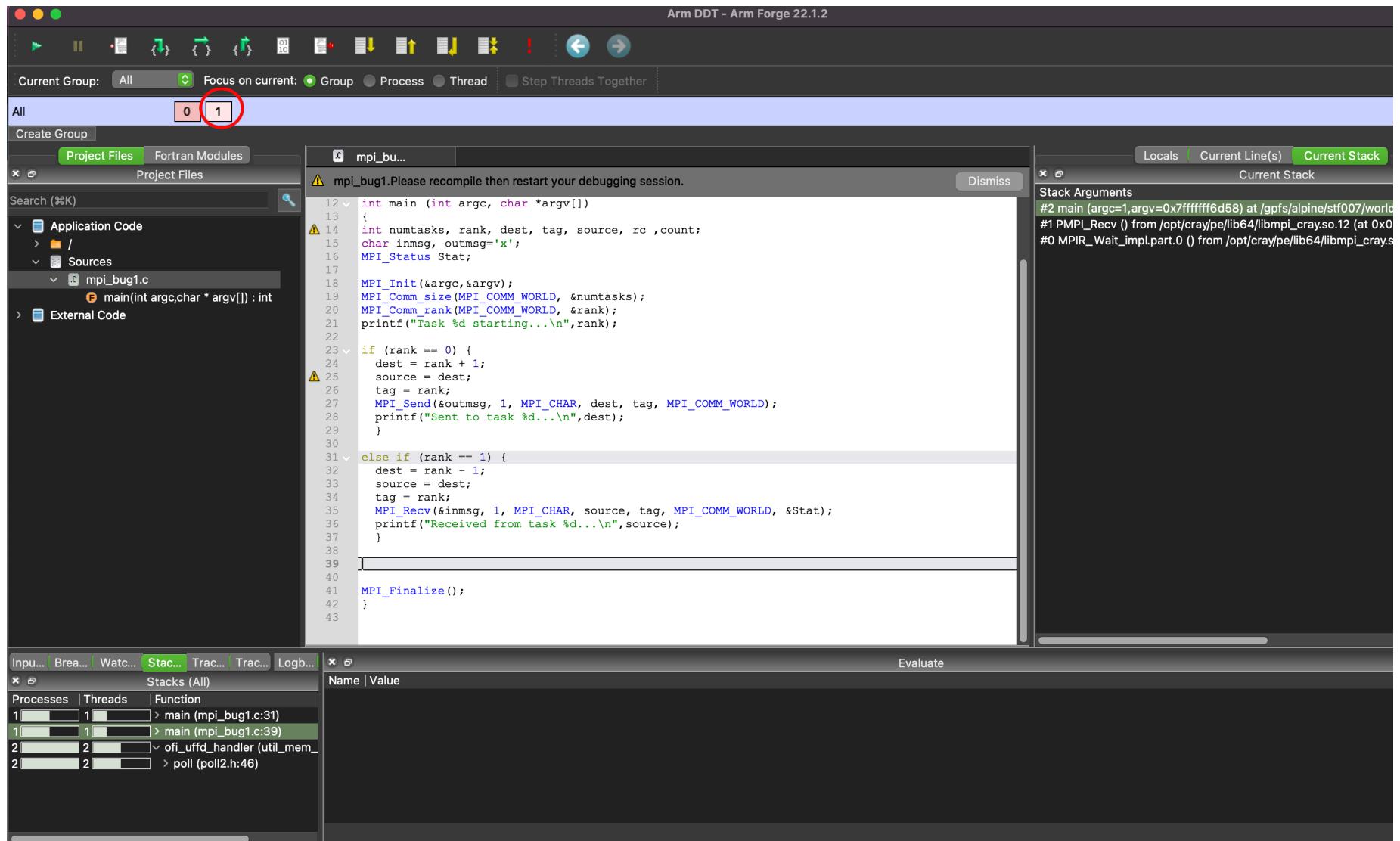
    if (rank == 0) {
        dest = rank + 1;
        source = dest;
        tag = rank;
        MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
        printf("Sent to task %d...\n",dest);
    }

    else if (rank == 1) {
        dest = rank - 1;
        source = dest;
        tag = rank;
        MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
        printf("Received from task %d...\n",source);
    }

    MPI_Finalize();
}
```

In the bottom right corner of the code editor, there is a message: "Process 0 is playing. In order to view local variables or the current stack you need to pause the process." The bottom navigation bar includes tabs for Input, Break, Watch, Stacks, Trace, Tracepoint, and Log.

# Step into – MPI\_WAIT on Rank 1



# Step into – ‘tag’ variable = 1

Current Group: All Focus on current: Group Process Thread Step Threads Together

All 0 1

mpi\_bug1.c

mpi\_bug1.c: Please recompile then restart your debugging session.

```
int main (int argc, char *argv[])
{
    int numtasks, rank, dest, tag, source, rc ,count;
    char inmsg, outmsg='x';
    MPI_Status Stat;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    printf("Task %d starting...\n",rank);

    if (rank == 0) {
        dest = rank + 1;
        source = dest;
        tag = rank;
        MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
        printf("Sent to task %d...\n",dest);
    }

    else if (rank == 1) {
        dest = rank - 1;
        source = dest;
        tag = rank;
        MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
        printf("Received from task %d...\n",source);
    }

    MPI_Finalize();
}
```

Locals

Name	Value
argc	1
> argv	0x7fffffff6d58
numtasks	2
rank	1
dest	2
tag	1
source	2
rc	0
count	0
inmsg	0 '\000'
outmsg	120 'x'
> Stat	

Input... Break... Watch... Stack... Trace... Log... Evaluate

Stacks (All)

Processes	Threads	Function
1	1	> main (mpi_bug1.c:31)
1	1	> main (mpi_bug1.c:39)
2	2	ofi_uffd_handler (util_mem_
2	2	> poll (poll2.h:46)

# Correcting the ‘tag’ = 0 for both ranks, program completes

```
[wcastil@crusher:MPI_bugs]$ srun -n2 ./mpi_bug1_fix
Task 1 starting...
Received from task 0...
Task 0 starting...
Sent to task 1...
```