Allinea Forge DDT

Debugging parallel codes more efficiently

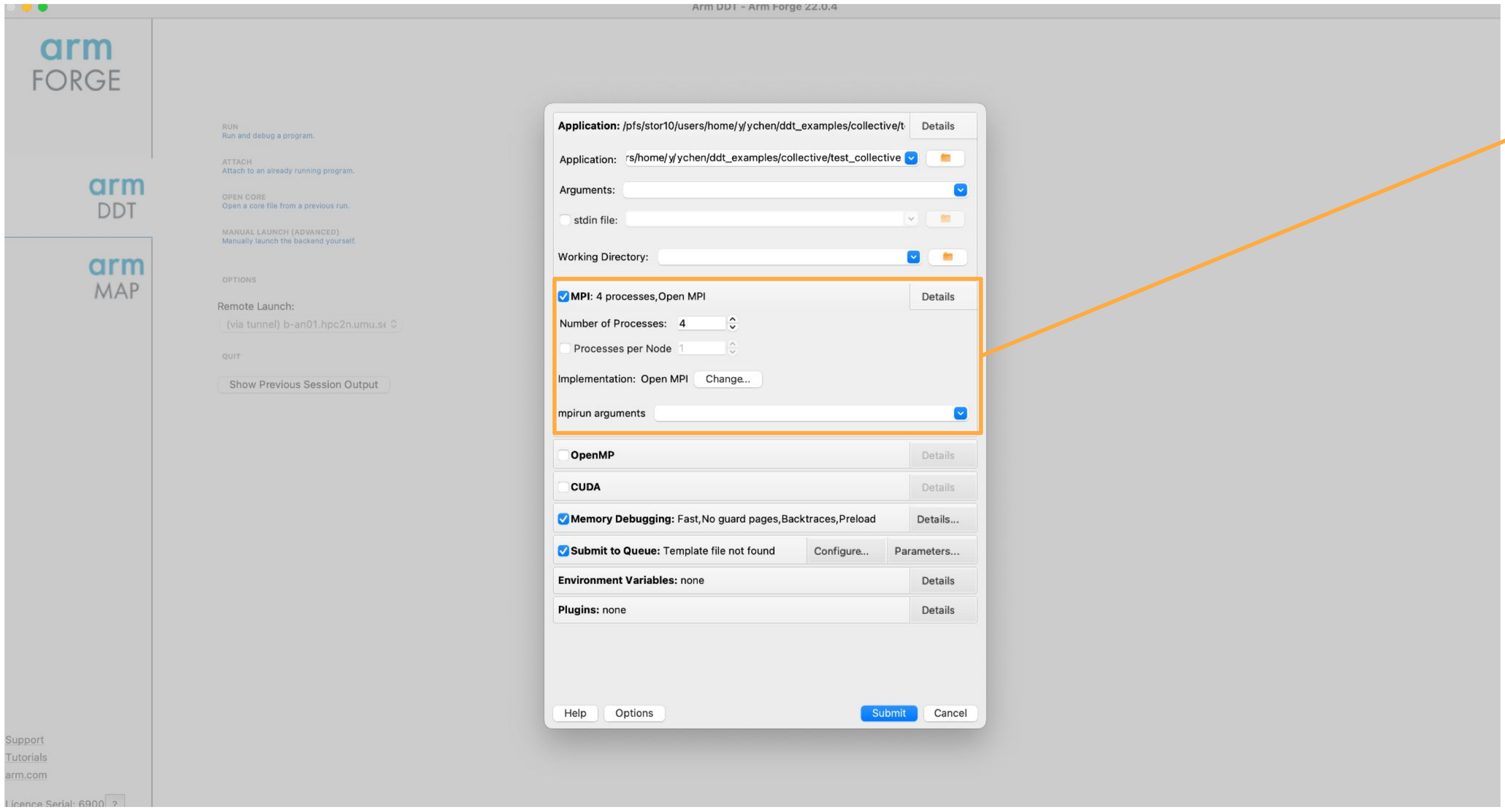
Mar 2024

Before usage

Checklist:

- Compiling with -g
- Check the version installed on the cluster:
    - Allinea: X11 forwarding, offline debugging
    - Arm/Linaro: remote client, X11 forwarding, offline debugging

Launch DDT



Arm DDT - Arm Forge 22.0.4

Remote Launch: (via tunnel) b-an01.hpc2n.umu.se

Licence Serial: 6900

Application: /pfs/stor10/users/home/y/ychen/ddt_examples/collective/t — Details

Application: rs/home/y/ychen/ddt_examples/collective/test_collective

Arguments:

stdin file:

Working Directory:

MPI: 4 processes,Open MPI — Details

Number of Processes: 4

Processes per Node 1

Implementation: Open MPI — Change...

mpirun arguments

OpenMP — Details

CUDA — Details

Memory Debugging: Fast,No guard pages,Backtraces,Preload — Details...

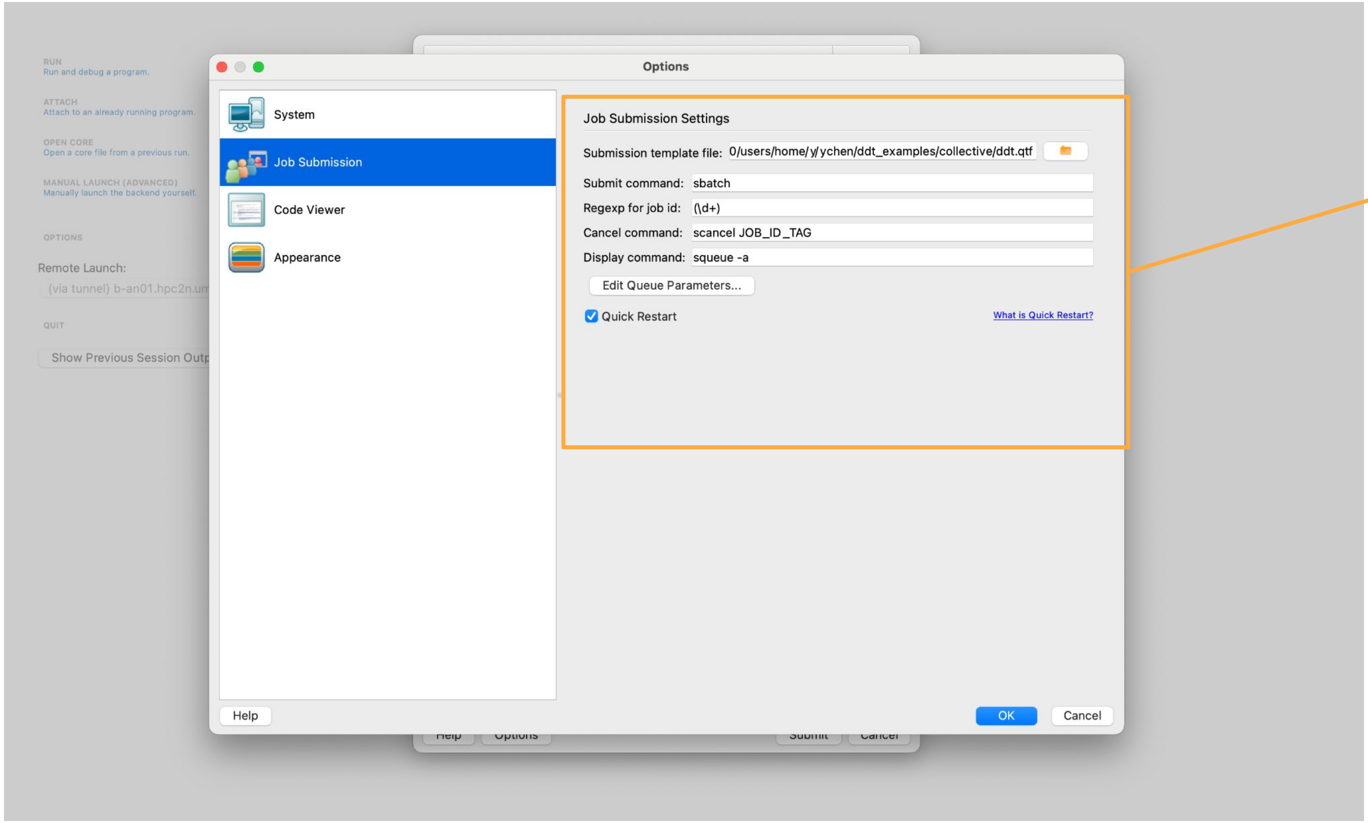
Submit to Queue: Template file not found — Configure... — Parameters...

Environment Variables: none — Details

Plugins: none — Details

set MPI

Launch DDT



Job Submission Settings

Submission template file: 0/users/home/y/ychen/ddt_examples/collective/ddt.qtf

Submit command: sbatch

Regexp for job id: (\d+)

Cancel command: scancel JOB_ID_TAG

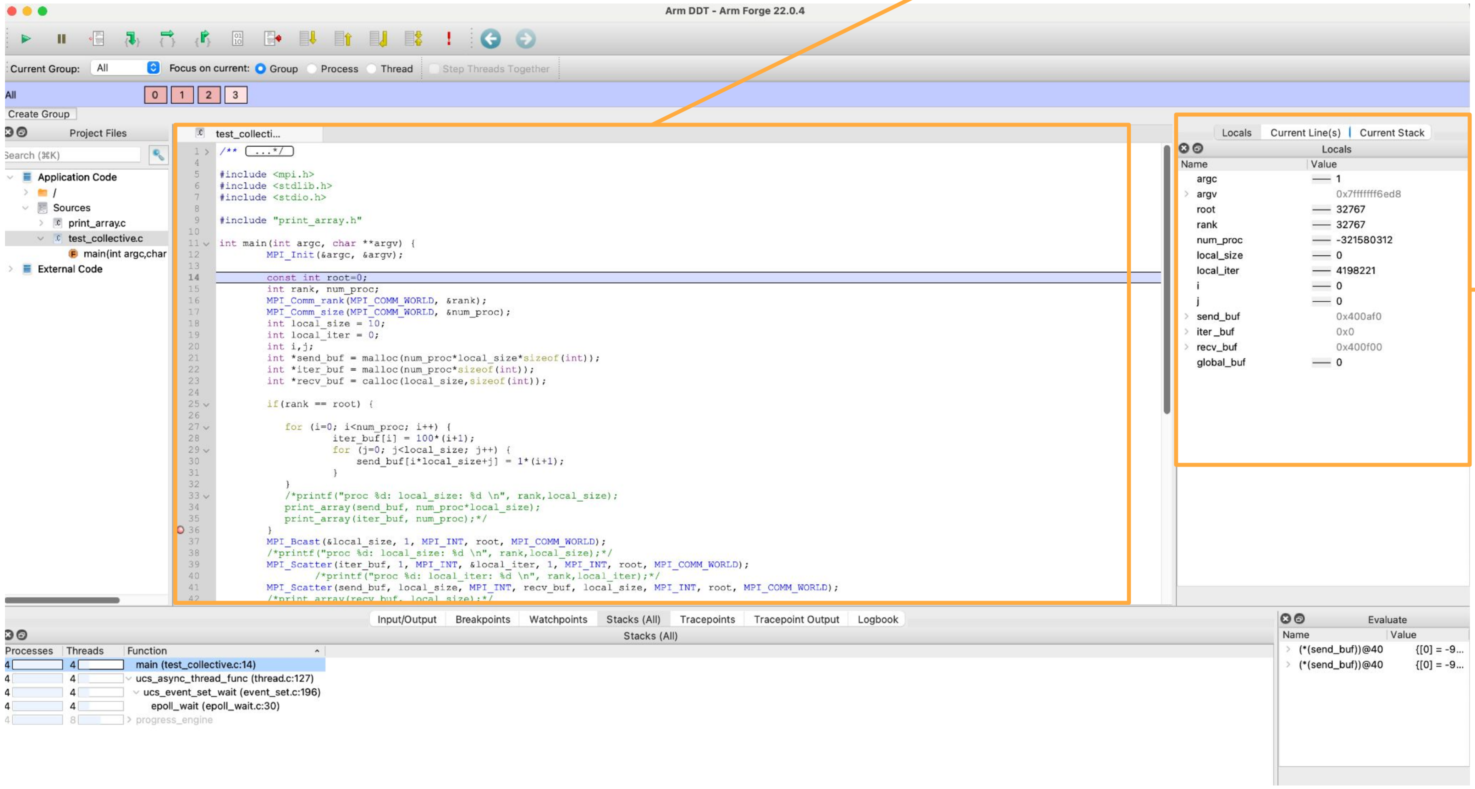
Display command: squeue -a

☑ Quick Restart

submit job via Slurm

load all required modules in .qtf

Debugging



```c
/**  ...*/

#include <mpi.h>
#include <stdlib.h>
#include <stdio.h>

#include "print_array.h"

int main(int argc, char **argv) {
        MPI_Init(&argc, &argv);

        const int root=0;
        int rank, num_proc;
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        MPI_Comm_size(MPI_COMM_WORLD, &num_proc);
        int local_size = 10;
        int local_iter = 0;
        int i,j;
        int *send_buf = malloc(num_proc*local_size*sizeof(int));
        int *iter_buf = malloc(num_proc*sizeof(int));
        int *recv_buf = calloc(local_size,sizeof(int));

        if(rank == root) {

           for (i=0; i<num_proc; i++) {
                 iter_buf[i] = 100*(i+1);
                 for (j=0; j<local_size; j++) {
                     send_buf[i*local_size+j] = 1*(i+1);
                 }
           }
           /*printf("proc %d: local_size: %d \n", rank,local_size);
           print_array(send_buf, num_proc*local_size);
           print_array(iter_buf, num_proc);*/
        }
        MPI_Bcast(&local_size, 1, MPI_INT, root, MPI_COMM_WORLD);
        /*printf("proc %d: local_size: %d \n", rank,local_size);*/
        MPI_Scatter(iter_buf, 1, MPI_INT, &local_iter, 1, MPI_INT, root, MPI_COMM_WORLD);
                /*printf("proc %d: local_iter: %d \n", rank,local_iter);*/
        MPI_Scatter(send_buf, local_size, MPI_INT, recv_buf, local_size, MPI_INT, root, MPI_COMM_WORLD);
        /*print_array(recv_buf, local_size);*/
```

Locals

Name	Value
argc	1
argv	0x7fffffff6ed8
root	32767
rank	32767
num_proc	-321580312
local_size	0
local_iter	4198221
i	0
j	0
send_buf	0x400af0
iter_buf	0x0
recv_buf	0x400f00
global_buf	0

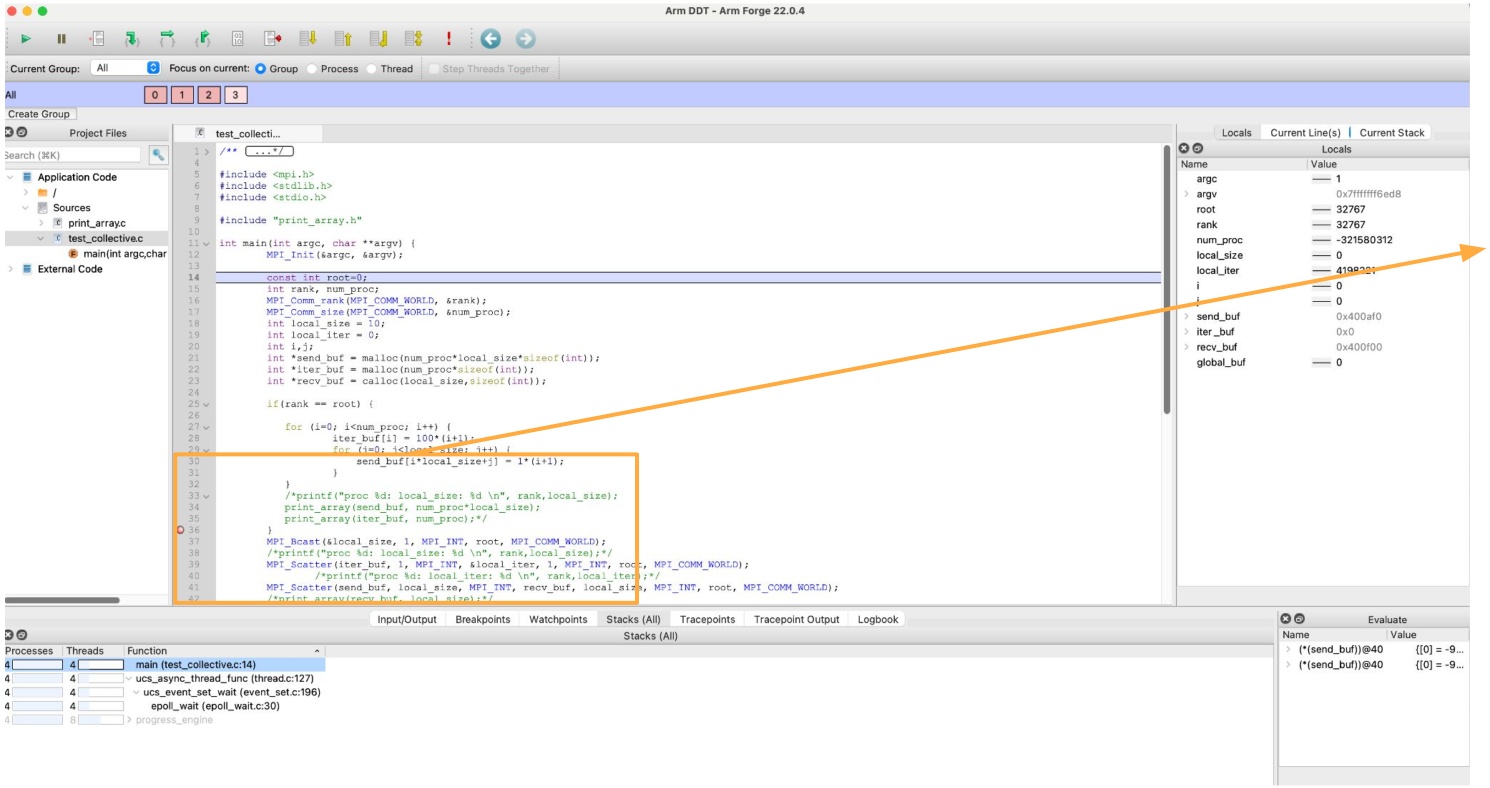
Stacks (All)

Processes	Threads	Function
4	4	main (test_collective.c:14)
4	4	ucs_async_thread_func (thread.c:127)
4	4	ucs_event_set_wait (event_set.c:196)
4	4	epoll_wait (epoll_wait.c:30)
4	8	progress_engine

Evaluate

Name	Value
(*(send_buf))@40	{[0] = -9...
(*(send_buf))@40	{[0] = -9...

Debugging



```
/**  ...*/

#include <mpi.h>
#include <stdlib.h>
#include <stdio.h>

#include "print_array.h"

int main(int argc, char **argv) {
        MPI_Init(&argc, &argv);

        const int root=0;
        int rank, num_proc;
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        MPI_Comm_size(MPI_COMM_WORLD, &num_proc);
        int local_size = 10;
        int local_iter = 0;
        int i,j;
        int *send_buf = malloc(num_proc*local_size*sizeof(int));
        int *iter_buf = malloc(num_proc*sizeof(int));
        int *recv_buf = calloc(local_size,sizeof(int));

        if(rank == root) {

           for (i=0; i<num_proc; i++) {
                   iter_buf[i] = 100*(i+1);
                   for (j=0; j<local_size; j++) {
                       send_buf[i*local_size+j] = 1*(i+1);
                   }
           }
           /*printf("proc %d: local_size: %d \n", rank,local_size);
           print_array(send_buf, num_proc*local_size);
           print_array(iter_buf, num_proc);*/
        }
        MPI_Bcast(&local_size, 1, MPI_INT, root, MPI_COMM_WORLD);
        /*printf("proc %d: local_size: %d \n", rank,local_size);*/
        MPI_Scatter(iter_buf, 1, MPI_INT, &local_iter, 1, MPI_INT, root, MPI_COMM_WORLD);
                /*printf("proc %d: local_iter: %d \n", rank,local_iter);*/
        MPI_Scatter(send_buf, local_size, MPI_INT, recv_buf, local_size, MPI_INT, root, MPI_COMM_WORLD);
        /*print_array(recv_buf, local_size);*/
```

Name	Value
argc	1
argv	0x7fffffff6ed8
root	32767
rank	32767
num_proc	-321580312
local_size	0
local_iter	4198221
i	0
j	0
send_buf	0x400af0
iter_buf	0x0
recv_buf	0x400f00
global_buf	0

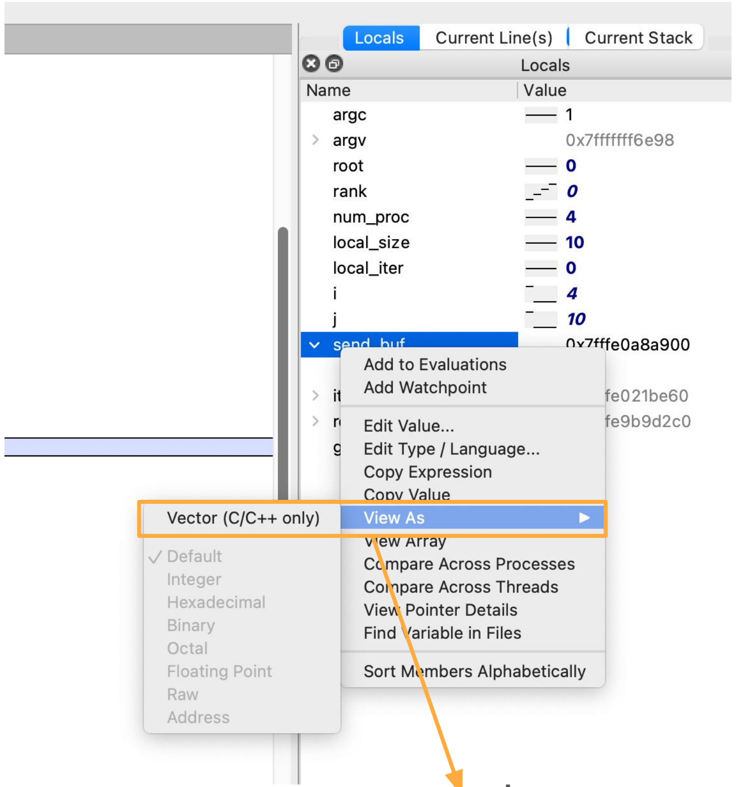
Stacks (All)

Processes	Threads	Function
4	4	main (test_collective.c:14)
4	4	ucs_async_thread_func (thread.c:127)
4	4	ucs_event_set_wait (event_set.c:196)
4	4	epoll_wait (epoll_wait.c:30)
4	8	progress_engine

Name	Value
(*(send_buf))@40	{[0] = -9...
(*(send_buf))@40	{[0] = -9...

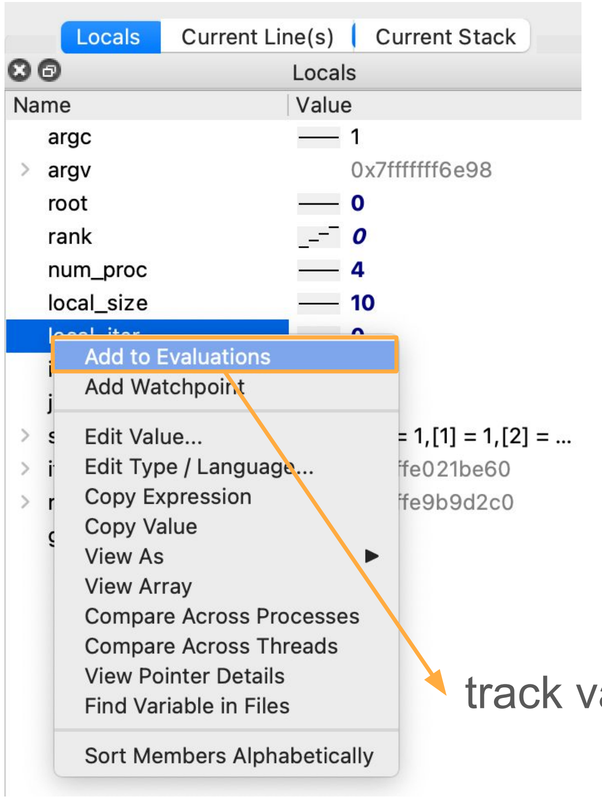
set breakpoints

Debugging



Name	Value
argc	1
argv	0x7fffffff6e98
root	0
rank	0
num_proc	4
local_size	10
local_iter	0
i	4
j	10
send_buf	0x7fffe0a8a900
it	fe021be60
r	fe9b9d2c0
g	

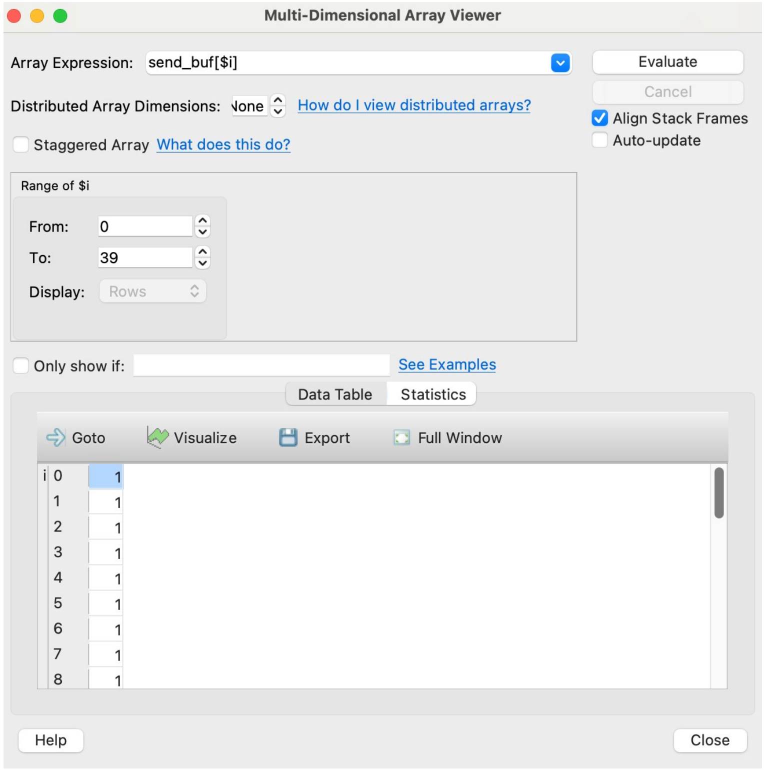
view array



Name	Value
argc	1
argv	0x7fffffff6e98
root	0
rank	0
num_proc	4
local_size	10
local_iter	0
i	
j	
s	= 1,[1] = 1,[2] = ...
i	ffe021be60
r	ffe9b9d2c0
g	

track variable

Debugging


Multi-Dimensional Array Viewer

Array Expression: send_buf[$i]

Distributed Array Dimensions: None How do I view distributed arrays?

Staggered Array What does this do?

Align Stack Frames [✓]
Auto-update

Range of $i
From: 0
To: 39
Display: Rows

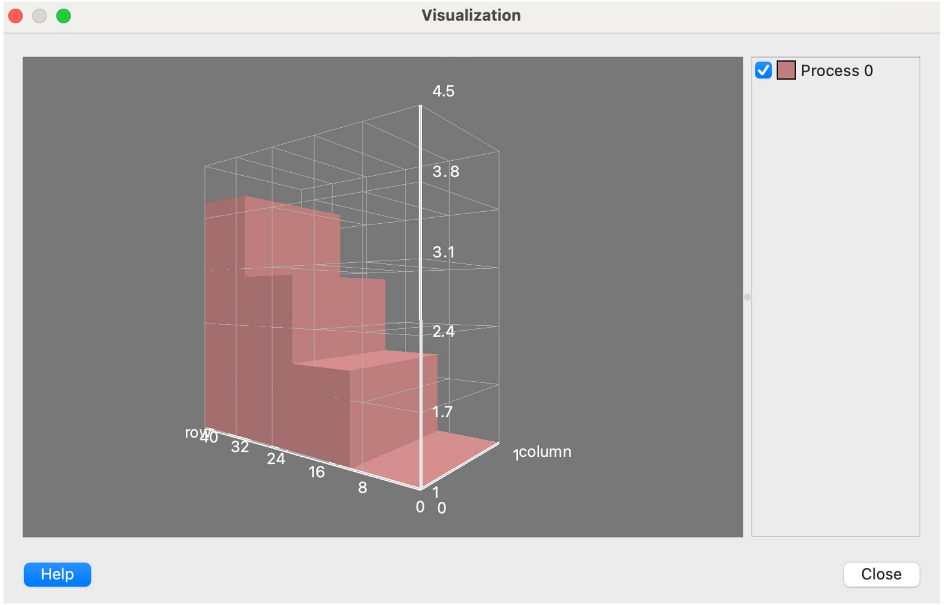
Only show if: See Examples

Data Table | Statistics

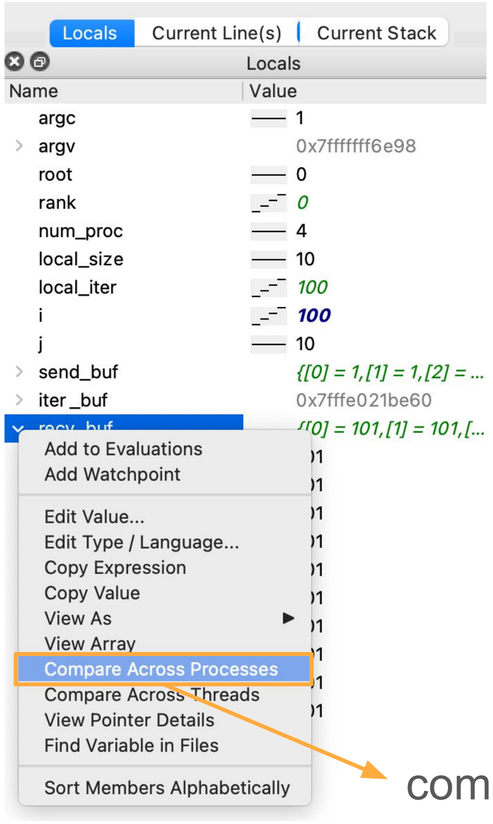
Goto | Visualize | Export | Full Window

i	
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1

Tools → Multi-Dimensional Array Viewer

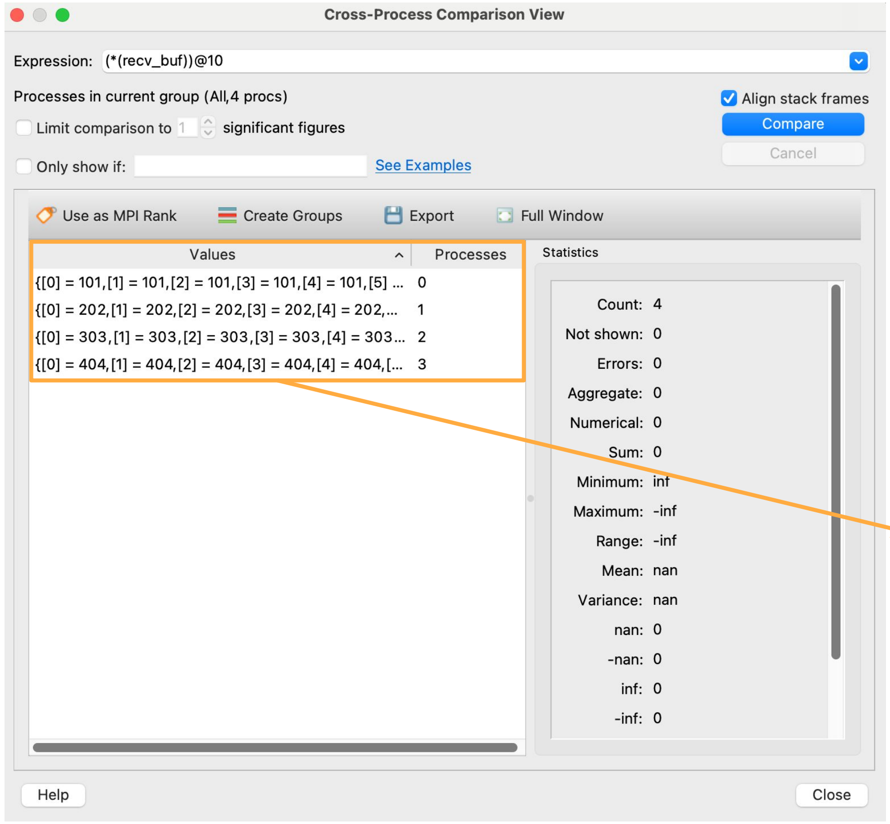


Debugging



Name	Value
argc	1
argv	0x7fffffff6e98
root	0
rank	*0*
num_proc	4
local_size	10
local_iter	*100*
i	***100***
j	10
send_buf	*{[0] = 1,[1] = 1,[2] = ...*
iter_buf	0x7fffe021be60
recv_buf	*{[0] = 101,[1] = 101,[...*

compare



Expression: (*(recv_buf))@10

Processes in current group (All,4 procs)

[✓] Align stack frames

[ ] Limit comparison to 1 significant figures

[ ] Only show if:

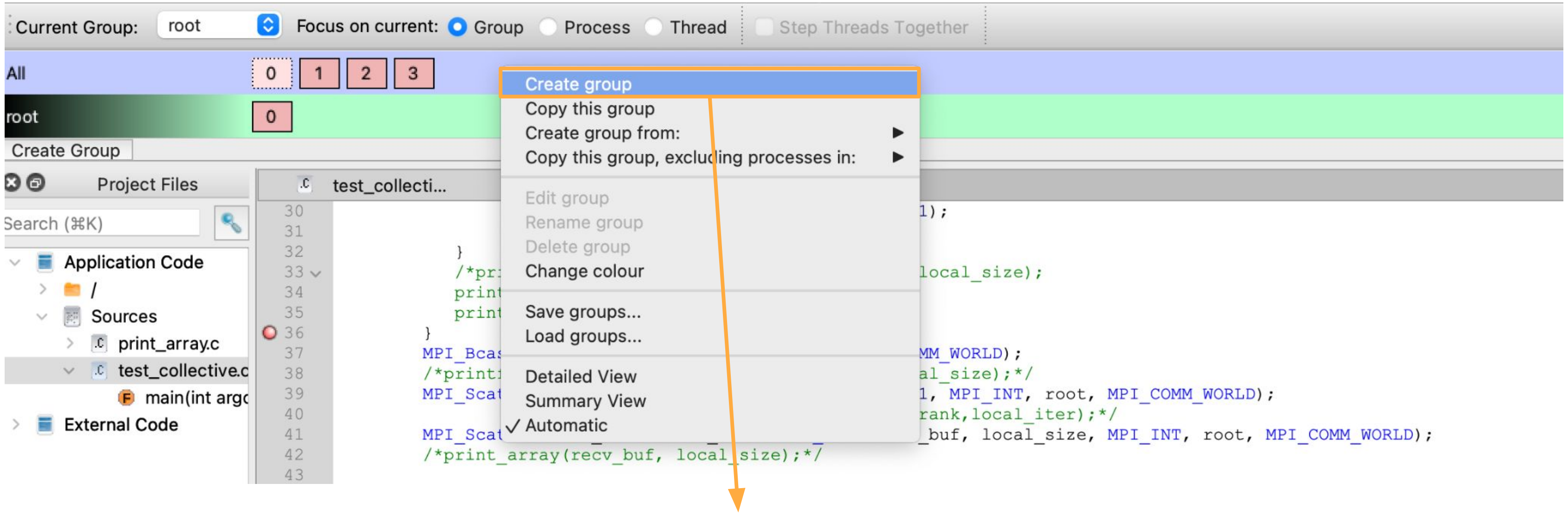
Values	Processes
{[0] = 101,[1] = 101,[2] = 101,[3] = 101,[4] = 101,[5] ...	0
{[0] = 202,[1] = 202,[2] = 202,[3] = 202,[4] = 202,...	1
{[0] = 303,[1] = 303,[2] = 303,[3] = 303,[4] = 303...	2
{[0] = 404,[1] = 404,[2] = 404,[3] = 404,[4] = 404,[...	3

Statistics

Count:	4
Not shown:	0
Errors:	0
Aggregate:	0
Numerical:	0
Sum:	0
Minimum:	inf
Maximum:	-inf
Range:	-inf
Mean:	nan
Variance:	nan
nan:	0
-nan:	0
inf:	0
-inf:	0

across processes

Debugging



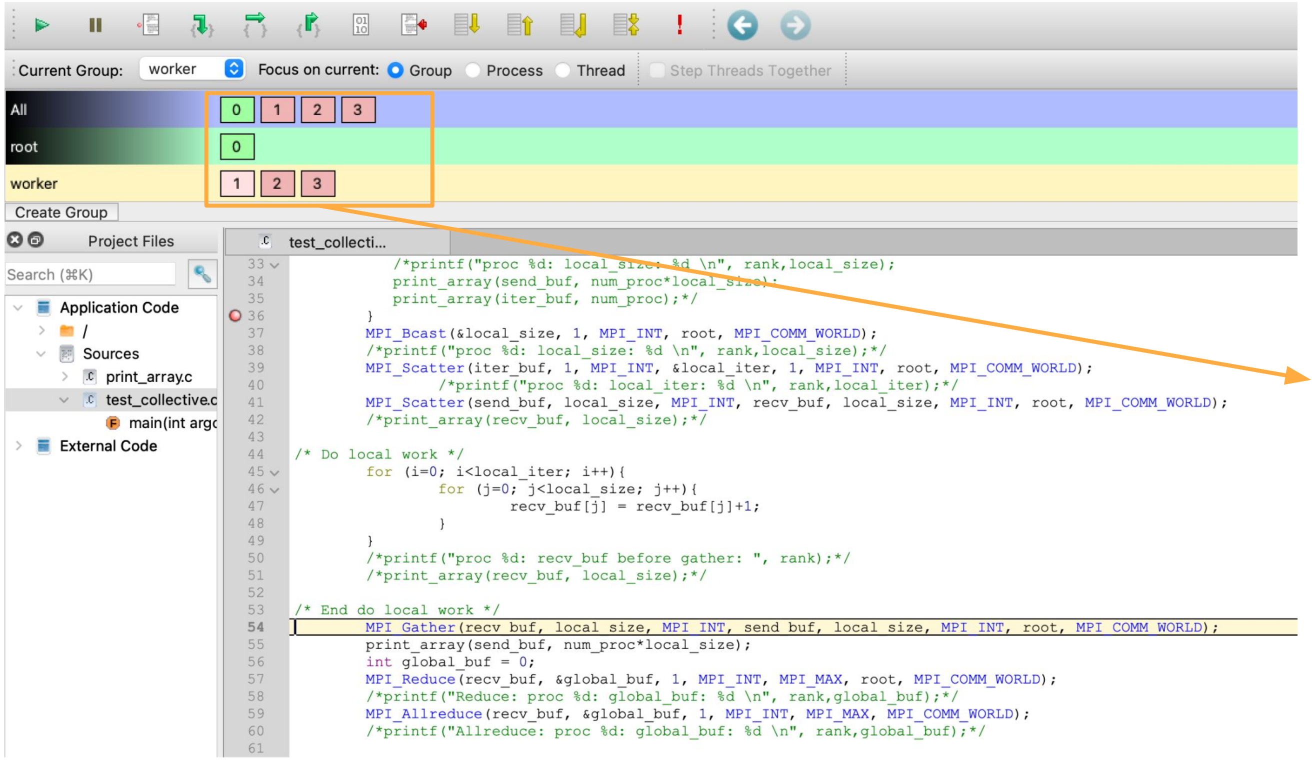
```
                                                    1);

            }
            /*pr                                    local_size);
            print
            print
        }
        MPI_Bcas                                    MM_WORLD);
        /*print                                     al_size);*/
        MPI_Scat                                    1, MPI_INT, root, MPI_COMM_WORLD);
                                                    rank,local_iter);*/
        MPI_Scat                                    _buf, local_size, MPI_INT, root, MPI_COMM_WORLD);
        /*print_array(recv_buf, local_size);*/
```

create group:

- root/worker rank
- communicator

Debugging



Current Group: worker

Focus on current: [✓ Group] Process Thread

```
           /*printf("proc %d: local_size: %d \n", rank,local_size);
           print_array(send_buf, num_proc*local_size);
           print_array(iter_buf, num_proc);*/
        }
        MPI_Bcast(&local_size, 1, MPI_INT, root, MPI_COMM_WORLD);
        /*printf("proc %d: local_size: %d \n", rank,local_size);*/
        MPI_Scatter(iter_buf, 1, MPI_INT, &local_iter, 1, MPI_INT, root, MPI_COMM_WORLD);
                /*printf("proc %d: local_iter: %d \n", rank,local_iter);*/
        MPI_Scatter(send_buf, local_size, MPI_INT, recv_buf, local_size, MPI_INT, root, MPI_COMM_WORLD);
        /*print_array(recv_buf, local_size);*/

/* Do local work */
        for (i=0; i<local_iter; i++){
                for (j=0; j<local_size; j++){
                        recv_buf[j] = recv_buf[j]+1;
                }
        }
        /*printf("proc %d: recv_buf before gather: ", rank);*/
        /*print_array(recv_buf, local_size);*/

/* End do local work */
        MPI_Gather(recv_buf, local_size, MPI_INT, send_buf, local_size, MPI_INT, root, MPI_COMM_WORLD);
        print_array(send_buf, num_proc*local_size);
        int global_buf = 0;
        MPI_Reduce(recv_buf, &global_buf, 1, MPI_INT, MPI_MAX, root, MPI_COMM_WORLD);
        /*printf("Reduce: proc %d: global_buf: %d \n", rank,global_buf);*/
        MPI_Allreduce(recv_buf, &global_buf, 1, MPI_INT, MPI_MAX, MPI_COMM_WORLD);
        /*printf("Allreduce: proc %d: global_buf: %d \n", rank,global_buf);*/
```

observe behaviors by group