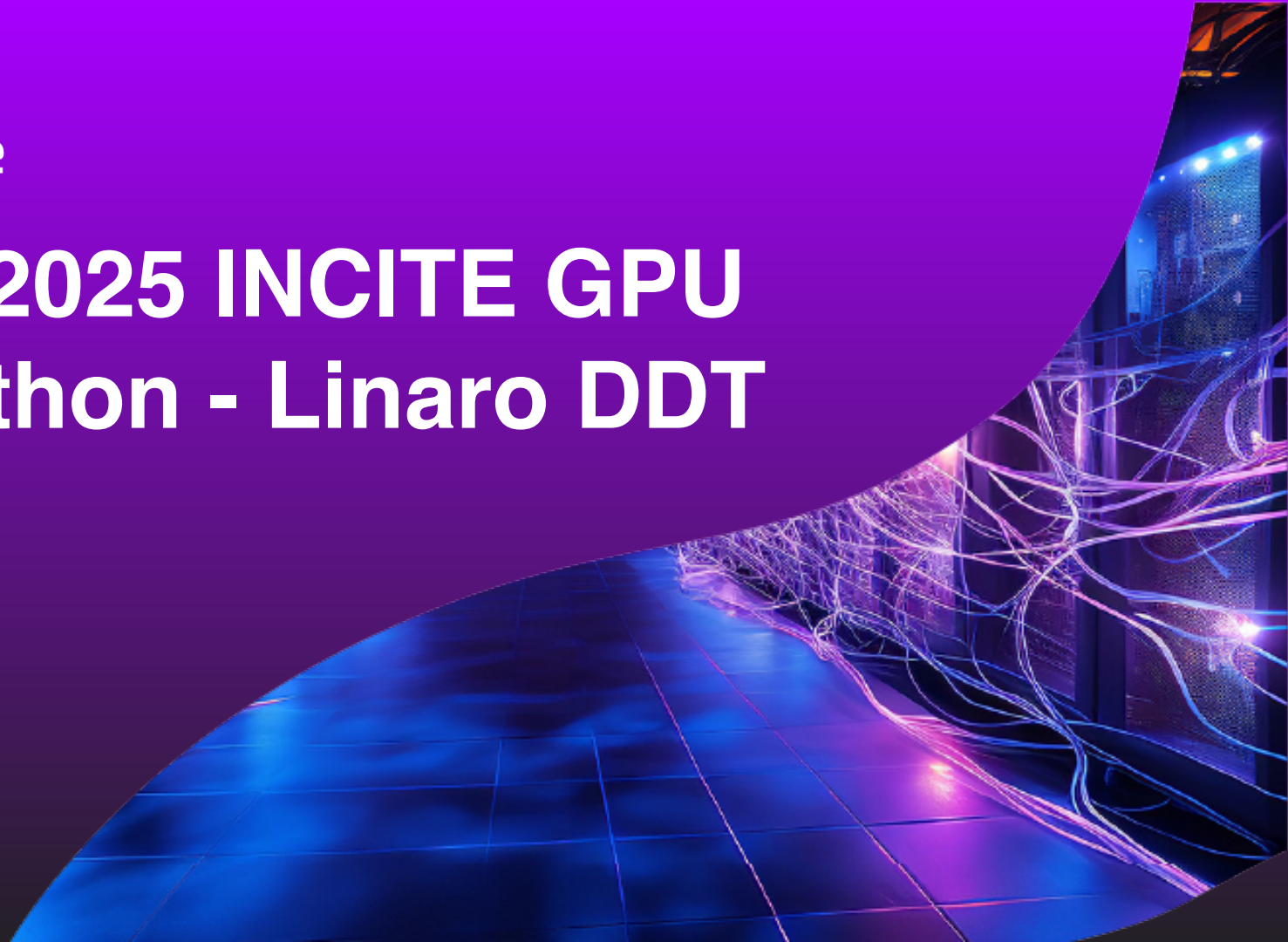
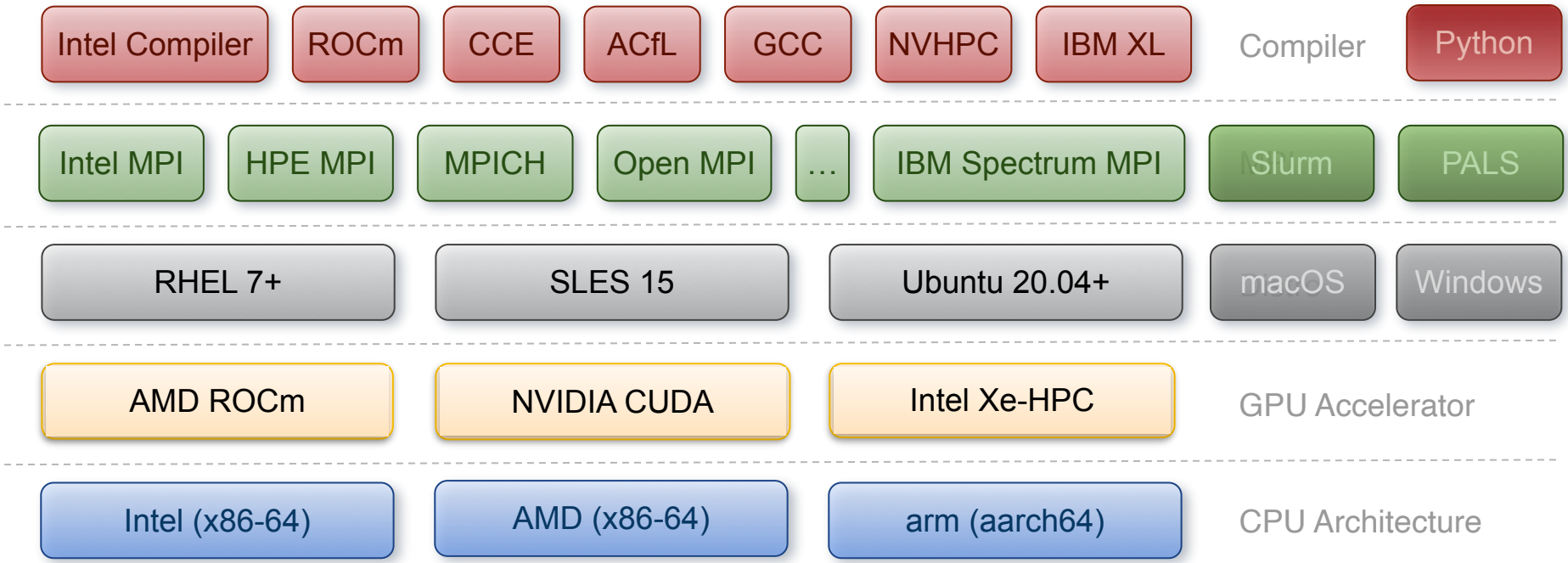
linaroforge

ALCF 2025 INCITE GPU Hackathon - Linaro DDT

Rudy Shand
Principal FAE

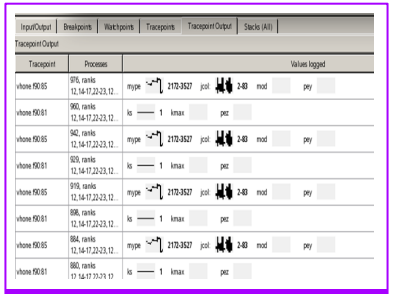


Supported Platforms

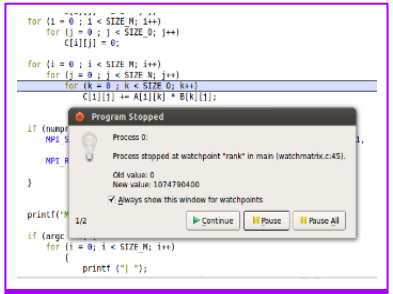


linaroforge

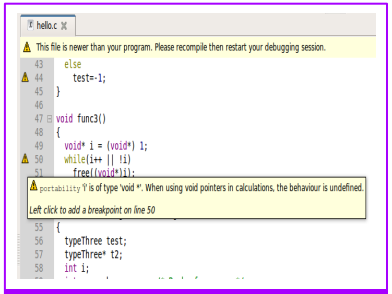
Linaro DDT Debugger Highlights



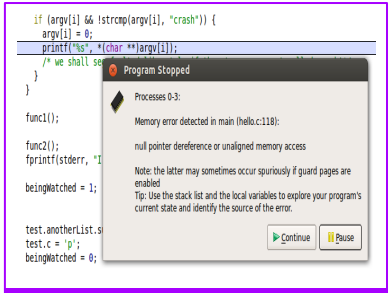
The scalable print alternative



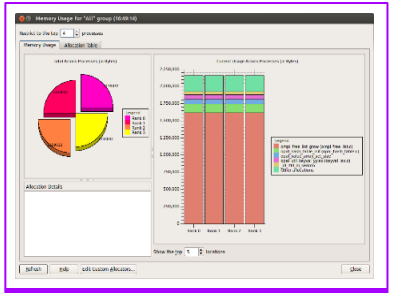
Stop on variable change



Static analysis warnings on code errors



Detect read/write beyond array bounds



Detect stale memory allocations

Memory Debugging

```
# Enable reading of debug environment variables:
host:~/demo> export NEOReadDebugKeys=1

# Disable RTLD_DEEPBIND, so that malloc/free calls bind to
# our memory debug library, instead of glibc:
host:~/demo> export NEODisableDeepBind=1
```

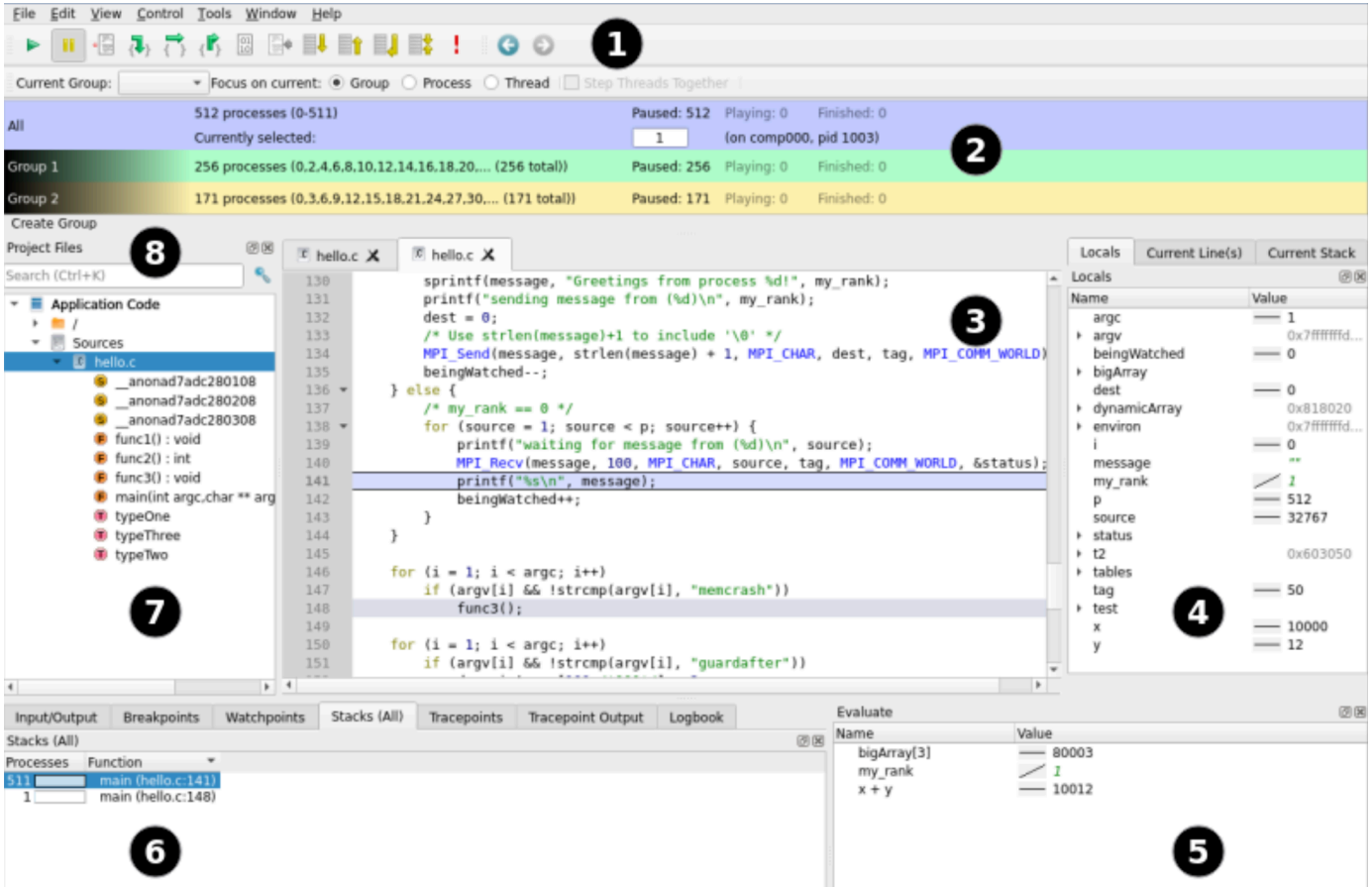
Terminology

	NVIDIA GPU	AMD GPU	Intel GPU
Name	CUDA	ROCm	Xe
Language	CUDA C/C++ and Fortran	HIP	SYCL
Execution Unit	Warp	Wavefront	Sub-group
EU Size	32 Threads	64 Threads	8, 16 or 32 Threads
EU GDB	…	GDB Thread	GDB Thread
EU Thread	…	Lane	Lane (work-item)
Forge GPU Thread	Lane	Lane	Lane

linaroforge

DDT UI

- 1 Process controls
- 2 Process groups
- 3 Source Code view
- 4 Variables
- 5 Evaluate window
- 6 Parallel Stack
- 7 Project files
- 8 Find a file or function



linaroforge

Debugging Intel Xe GPUs

Using Linaro DDT

Debug code simultaneously on the GPU and the CPU

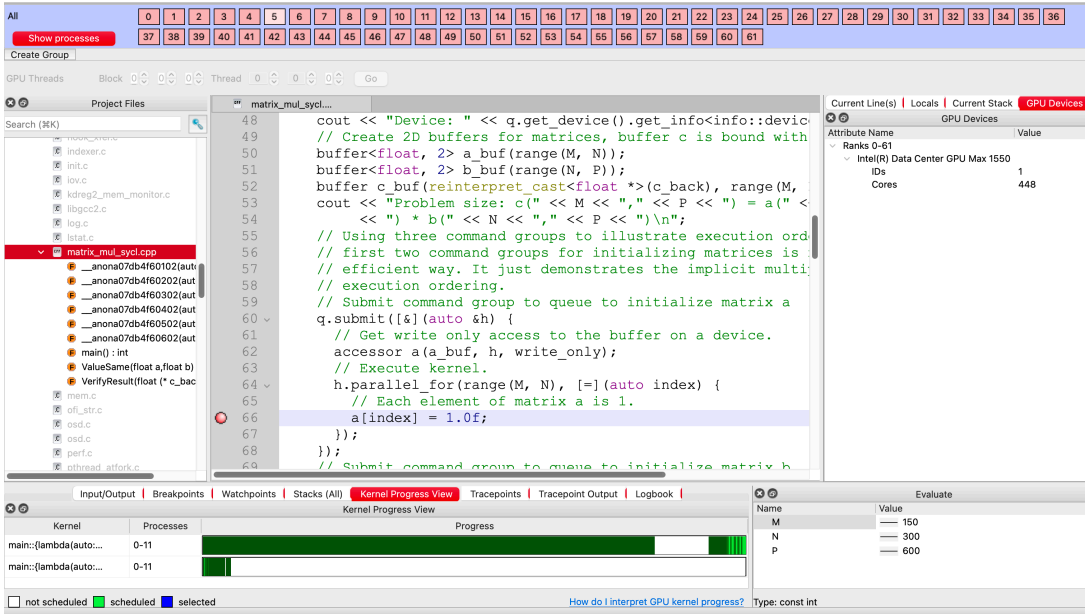
Controlling the GPU execution:

- All active threads in a Sub-group will execute in lockstep. Therefore, DDT will step16 threads at a time.
- Play/Continue runs all GPU threads
- Pause will pause a running kernel

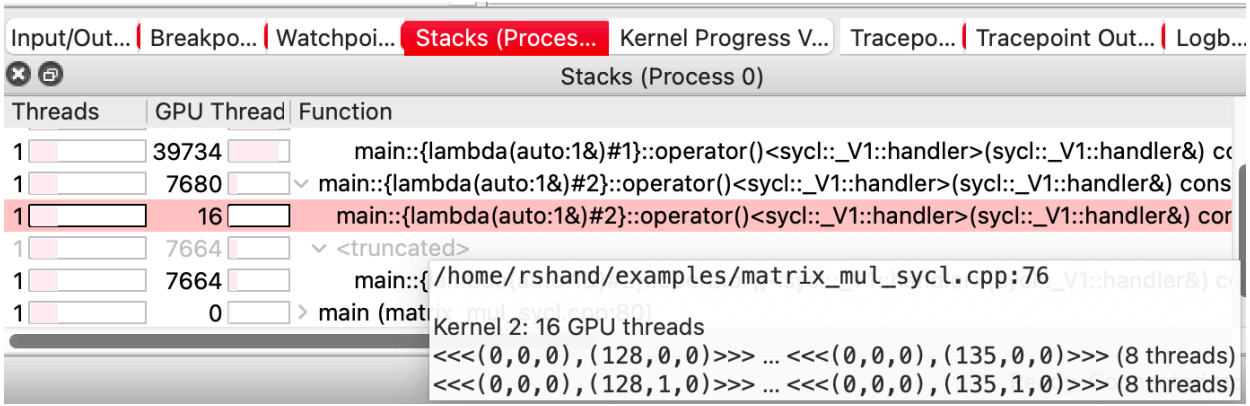
Key (additional) GPU features:

- Kernel Progress View
- GPU thread in parallel stack view
- GPU Thread Selector
- GPU Device Pane

Kernels must be compiled with the -g and -O0 flags



Parallel Stack View

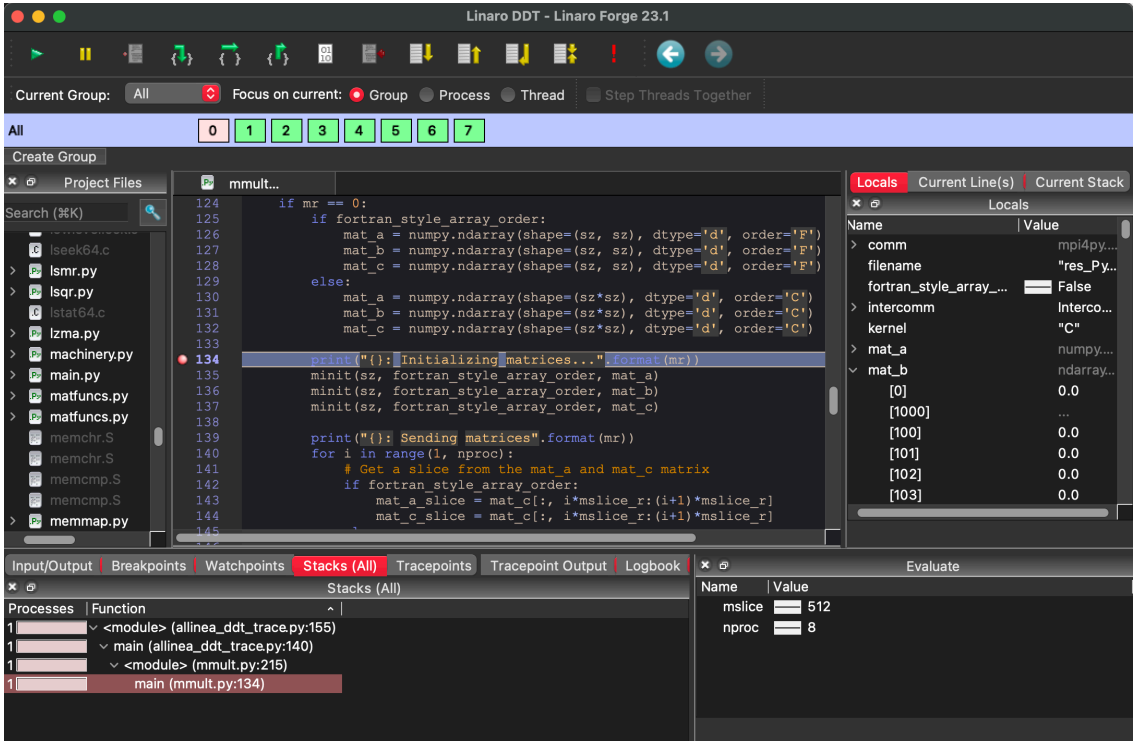


- Display location and number of threads
- Click item:
  - Select GPU Thread.
  - Update variable display.
  - Move Source Code Viewer.
- Tooltip displays:
  - GPU Thread Ranges.
  - Size of each range.

linaroforge

Python Debugging

- Debug Features
  - Sparklines for Python variables
  - Tracepoints
  - MDA viewer
  - Mixed language support
- Improved Evaluations:
  - Matrix objects
  - Array objects
  - Pandas DataFrame
  - Series objects
- Python Specific:
  - Stop on uncaught Python exception
  - Show F-string variables
  - Mpi4py, NumPy, SciPy



ddt --connect mpiexec -n 8 python3
%allinea_python_debug% ./mmult.py

Run DDT in offline mode

Run the application under DDT and halt or report when a failure occurs

You can run the debugger in non-interactive mode

- For long-running jobs / debugging at very high scale
- For automated testing, continuous integration…

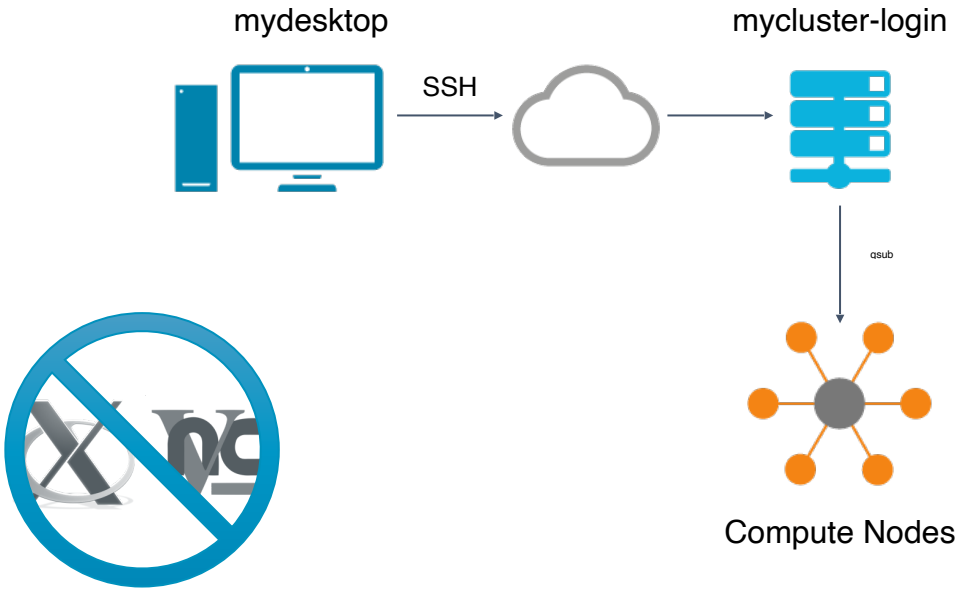
To do so, use following arguments:

- $ ddt **--offline --output=report.html** mpirun ./jacobi_omp_mpi_gnu.exe
  - **--offline** enable non-interactive debugging
  - **--output** specifies the name and output of the non-interactive debugging session
    - Html
    - Txt
  - Add **--mem-debug** to enable memory debugging **and memory leak detection**

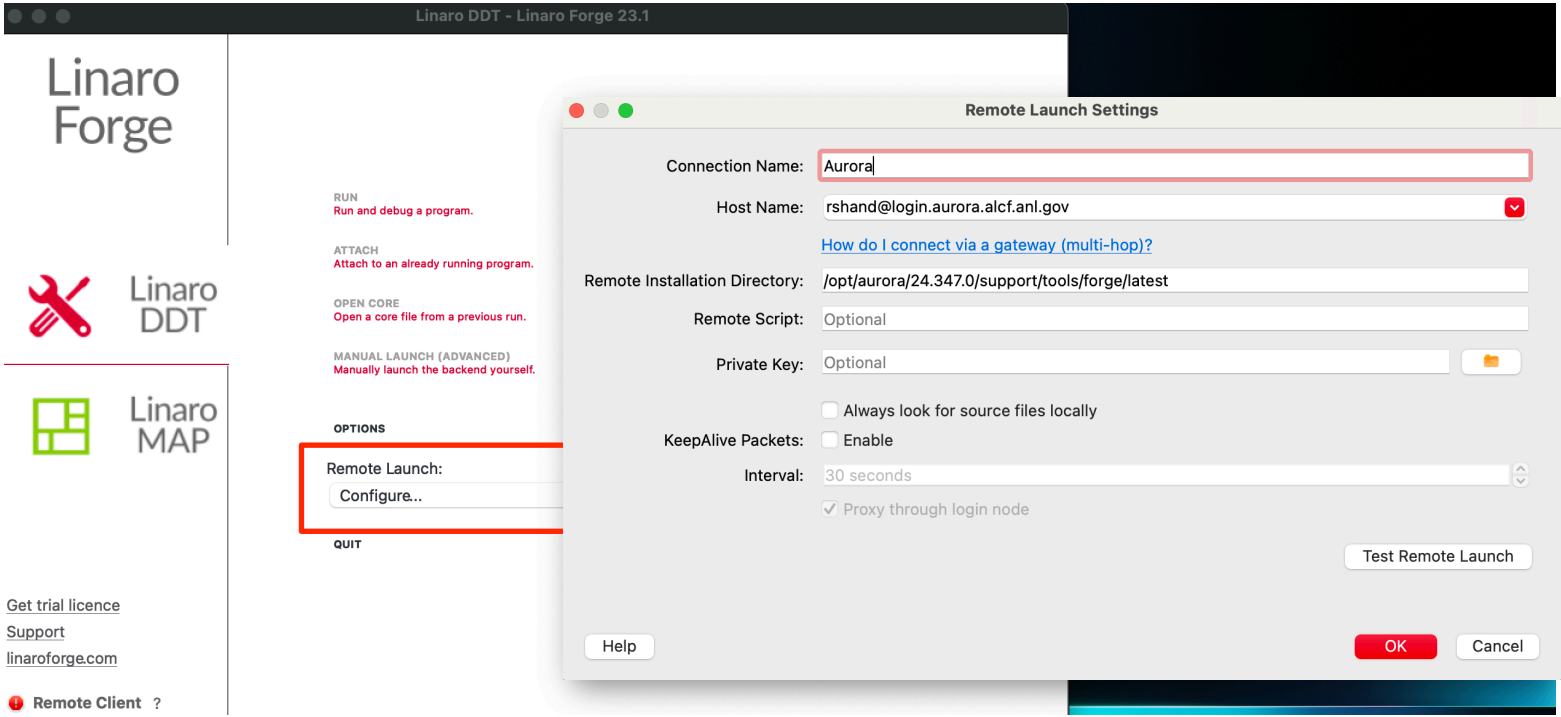
```
ddt --offline -o jacobi_omp_mpi_gnu_debug.txt \
                                 --trace-at _jacobi.F90:83,residual \
                                 mpiexec ./jacobi_omp_mpi_gnu.exe
```

The Forge GUI and where to run it

DDT provides a powerful GUI that can be run in a variety of configurations.



Remote connection to AWS



linaroforge

Debug with DDT on Aurora

```
mpicxx -fsycl -g -O0 matrix_mul_sycl.cpp -o matrixmul

qsub -l select=2 -l walltime=30:00 -l filesystems=flare -A gpu_hack -q gpu_hack_prio -I

. /soft/compilers/oneapi/2025.1.0/debugger/2025.1/env/vars.sh

https://docs.alcf.anl.gov/aurora/debugging/ddt-aurora/#invoking-the-ddt-server-from-aurora

ddt --np=24 --connect --mpi=generic --mpiargs="--ppn 12 --envall" ./matrixmul
```

linaroforge

Thank you

rudy.shand@linaro.org
support@forge.linaro.com

Go to www.linaroforge.com