

WORKSHOP

---

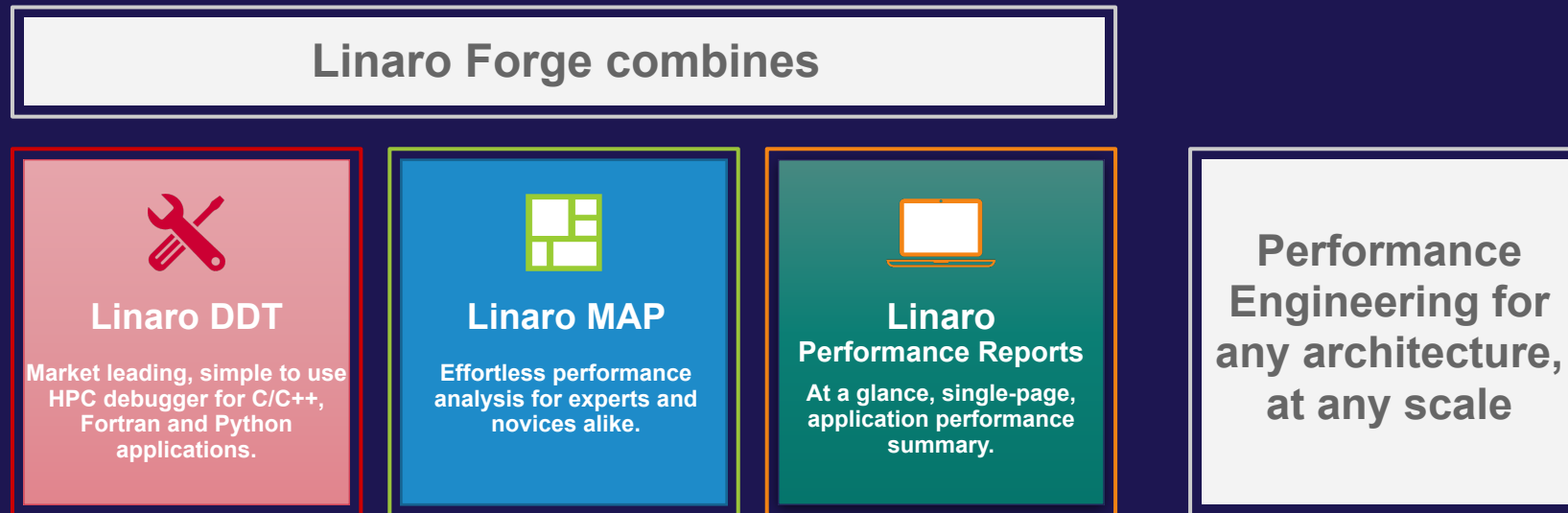
# ALCF Hands-on HPC Workshop

September 23-25 & October 7-9, 2025  
Argonne National Laboratory



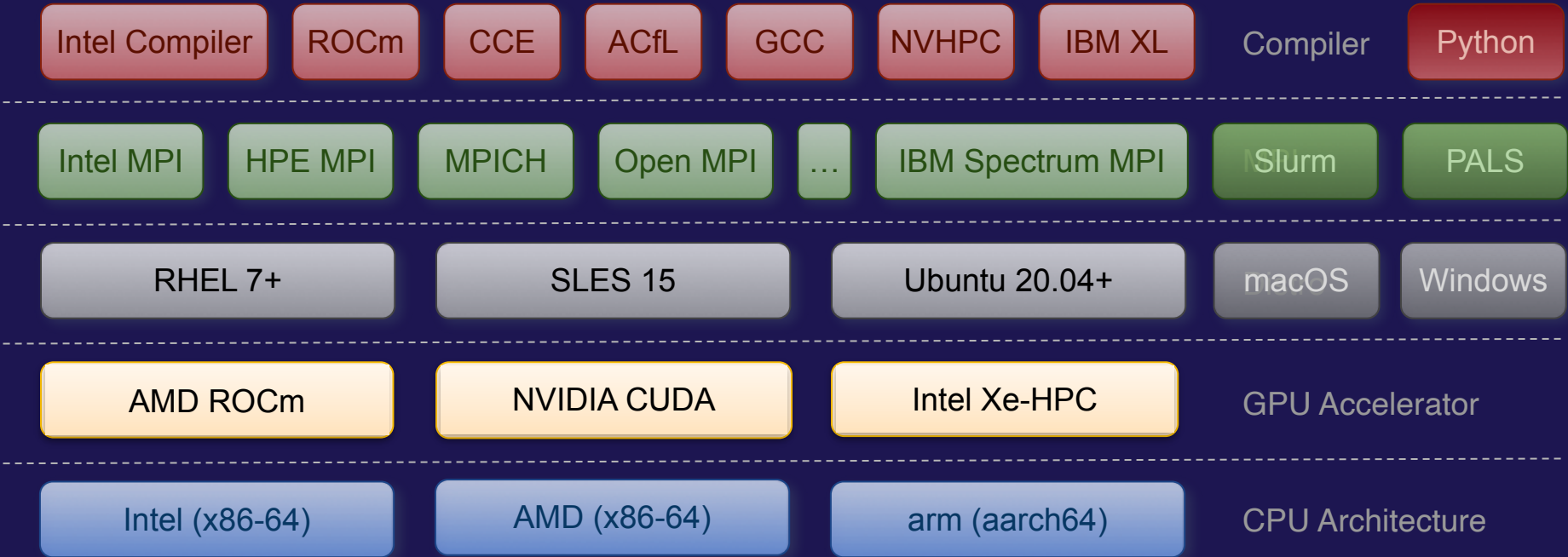
# HPC Development Solutions from Linaro

Best in class commercially supported tools for Linux and high-performance computing (HPC)



# DDT Supported Platforms

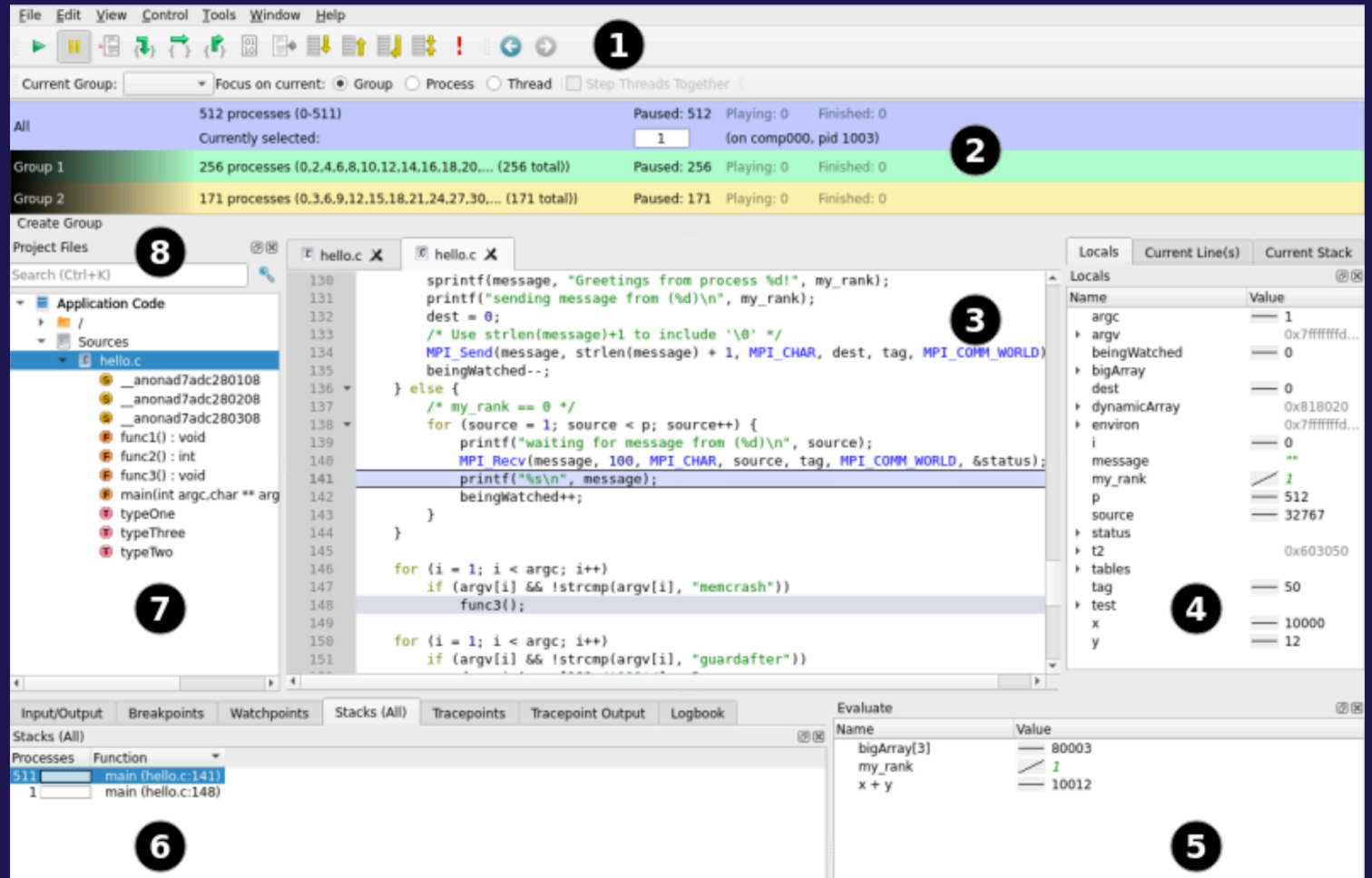
Works across hardware architectures and HPC technologies



# DDT UI

## Intuitive and scalable user interface

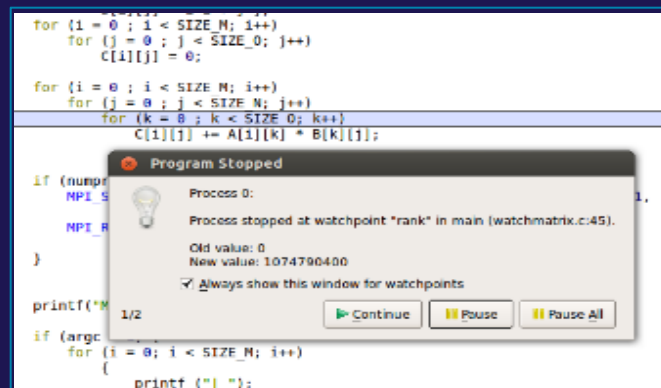
- 1 Process controls
- 2 Process groups
- 3 Source Code view
- 4 Variables
- 5 Evaluate window
- 6 Parallel Stack
- 7 Project files
- 8 Find a file or function



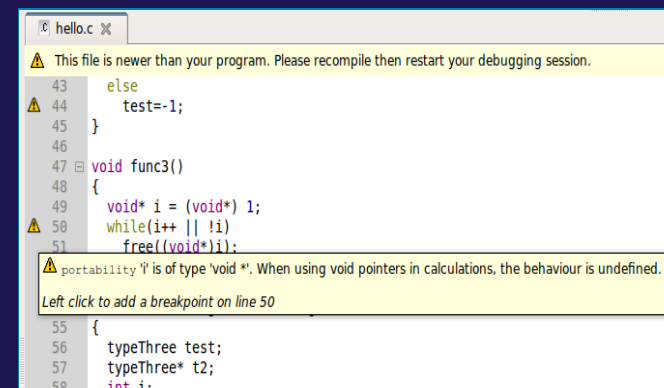
# Linaro DDT Debugger Highlights

Input/Output	Breakpoints	Watchpoints	Tracepoints	Tracepoint Output	Stacks (All)
Tracepoint Output					
Tracepoint	Processes	Values logged			
vhone f50 85	976, ranks 12,14-17,22-23,12...	mype	2172-3527	jcol	2-83 mod pay
vhone f50 81	960, ranks 12,14-17,22-23,12...	ls	1	kmax	pez
vhone f50 85	942, ranks 12,14-17,22-23,12...	mype	2172-3527	jcol	2-83 mod pay
vhone f50 81	929, ranks 12,14-17,22-23,12...	ls	1	kmax	pez
vhone f50 85	919, ranks 12,14-17,22-23,12...	mype	2172-3527	jcol	2-83 mod pay
vhone f50 81	898, ranks 12,14-17,22-23,12...	ls	1	kmax	pez

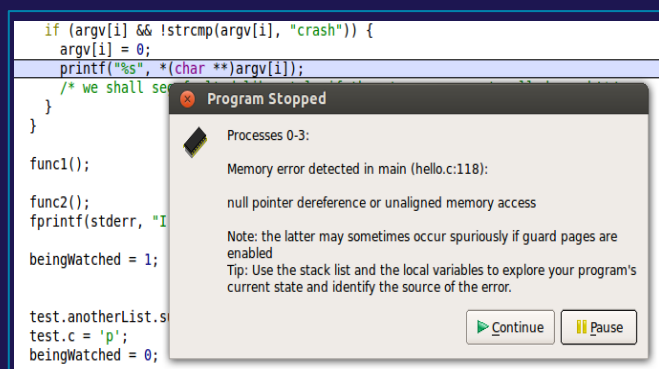
The scalable print alternative



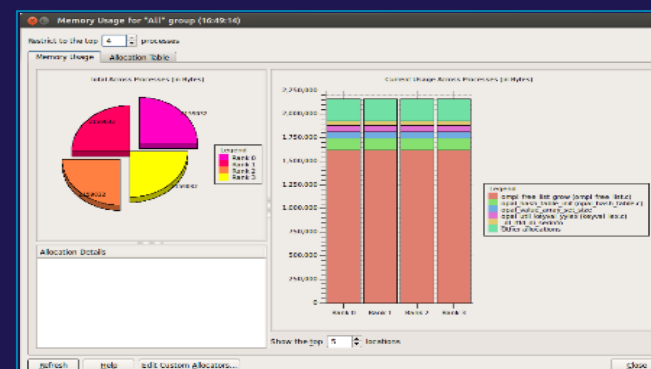
Stop on variable change



Static analysis warnings on code errors



Detect read/write beyond array bounds



Detect stale memory allocations

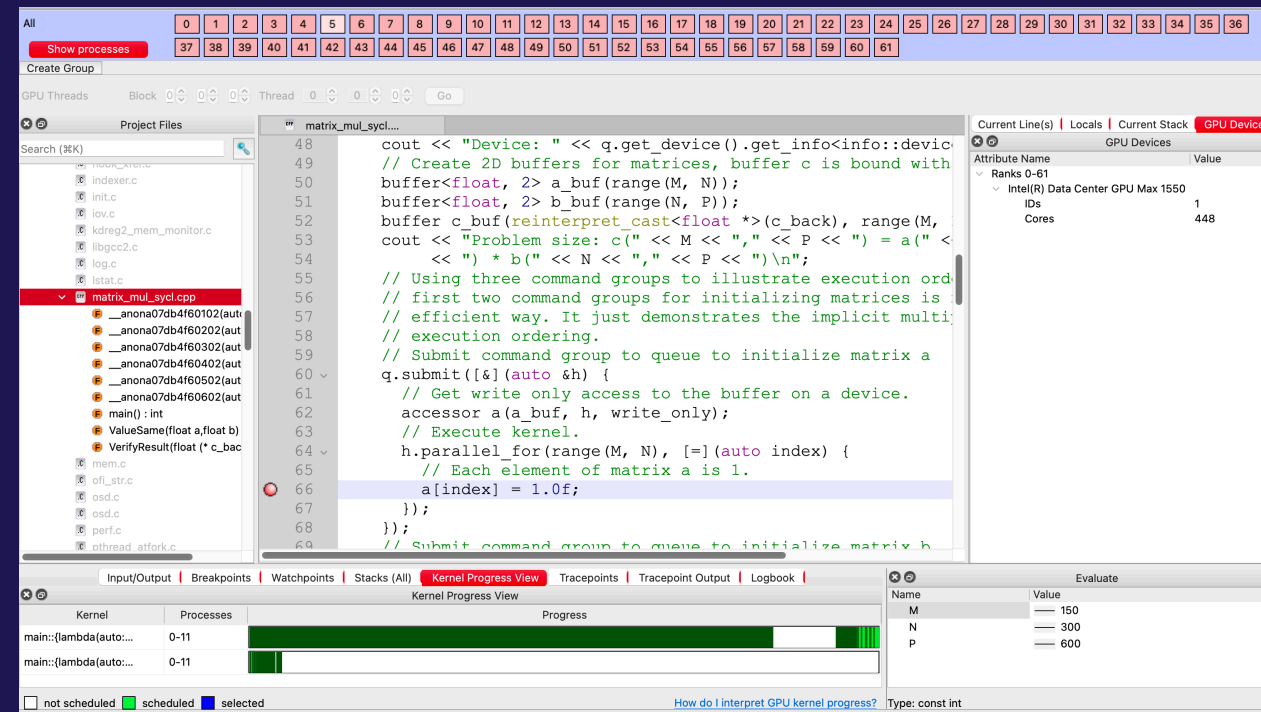
# Debugging Intel Xe GPUs

## Using Linaro DDT

Debug code simultaneously on the GPU and the CPU

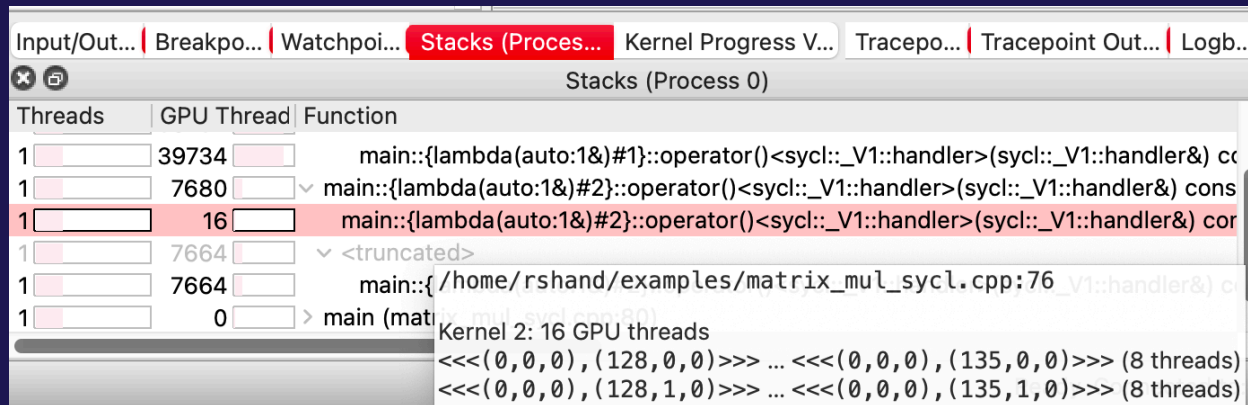
- Controlling the GPU execution:
  - All active threads in a Sub-group will execute in lockstep. Therefore, DDT will step 16 threads at a time.
  - Play/Continue runs all GPU threads
  - Pause will pause a running kernel
- Key (additional) GPU features:
  - Kernel Progress View
  - GPU thread in parallel stack view
  - GPU Thread Selector
  - GPU Device Pane

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





# Parallel Stack View



The screenshot shows a window titled "Stacks (Process 0)" with a tab labeled "Stacks (Proces...". The window contains a table with three columns: "Threads", "GPU Thread", and "Function". The table lists several threads, with the third thread (GPU Thread 16) highlighted in red. Below the table, there is a section titled "Kernel 2: 16 GPU threads" showing two ranges of threads: "<<<(0,0,0),(128,0,0)>>> ... <<<(0,0,0),(135,0,0)>>> (8 threads)" and "<<<(0,0,0),(128,1,0)>>> ... <<<(0,0,0),(135,1,0)>>> (8 threads)".

Threads	GPU Thread	Function
1	39734	main::{lambda(auto:1&)#1)::operator()<sycl::_V1::handler>(sycl::_V1::handler&) cc
1	7680	main::{lambda(auto:1&)#2)::operator()<sycl::_V1::handler>(sycl::_V1::handler&) cons
1	16	main::{lambda(auto:1&)#2)::operator()<sycl::_V1::handler>(sycl::_V1::handler&) cor
1	7664	<truncated>
1	7664	main::{ /home/rshand/examples/matrix_mul_sycl.cpp:76 _V1::handler&) c
1	0	> main (mat

Kernel 2: 16 GPU threads  
<<<(0,0,0),(128,0,0)>>> ... <<<(0,0,0),(135,0,0)>>> (8 threads)  
<<<(0,0,0),(128,1,0)>>> ... <<<(0,0,0),(135,1,0)>>> (8 threads)

Display location and number of threads

- 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

# Python Debugging

- Debug Features

- Sparklines for Python variables
- Tracepoints
- MDA viewer
- Mixed language support

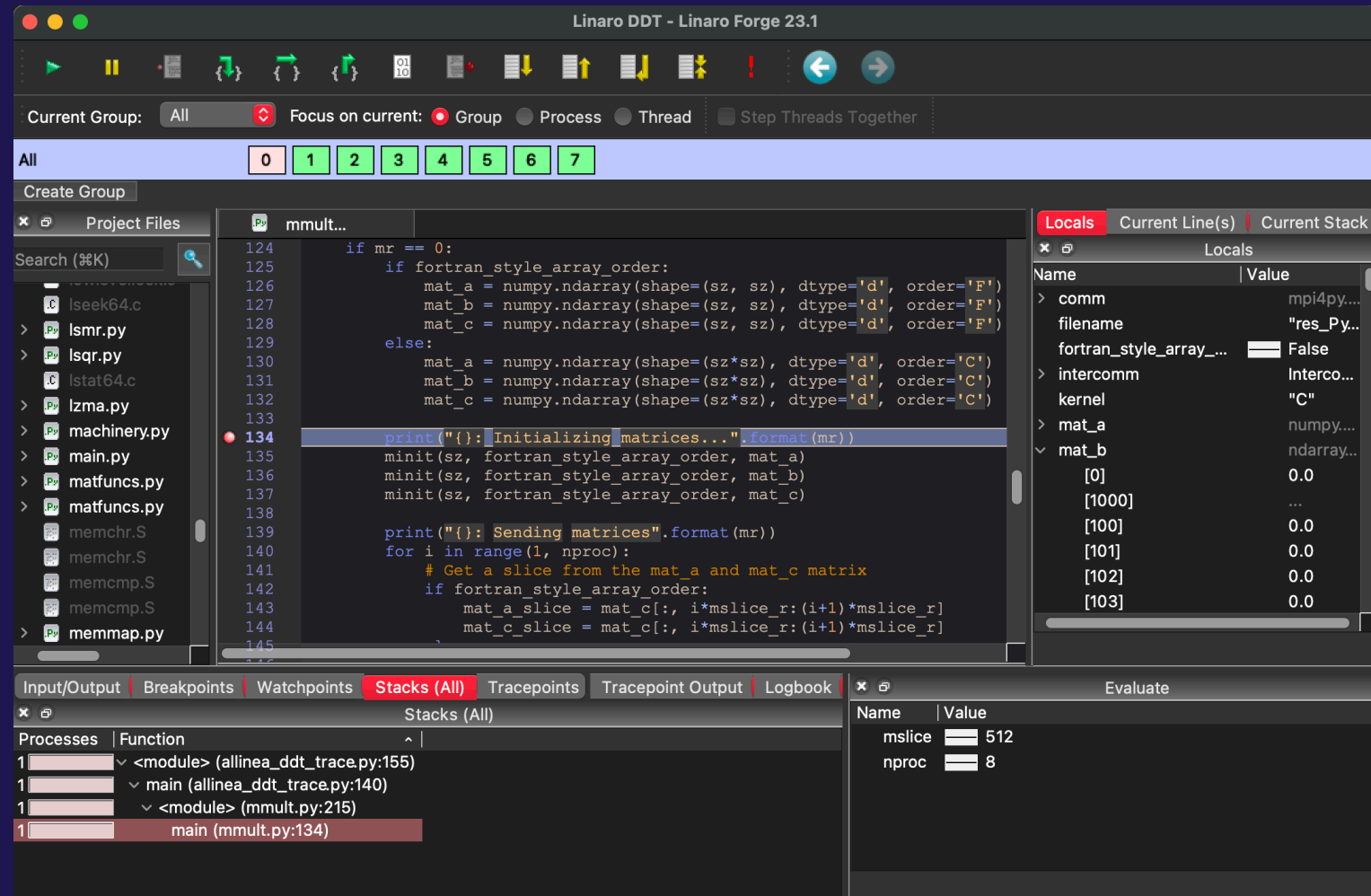
- Improved Evaluations:

- 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
```





# 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...

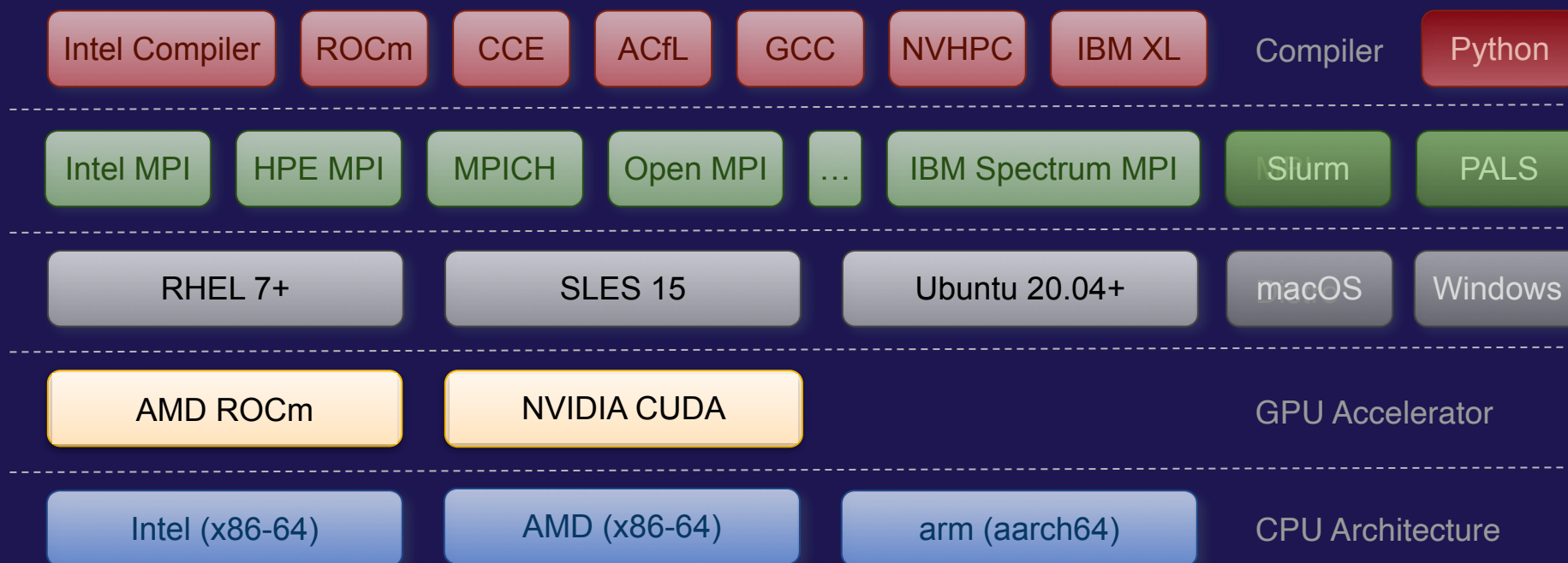
To do so, use following arguments:

- `$ 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 or 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
```

# MAP and Performance Reports Supported Platforms

Works across hardware architectures and HPC technologies



# Linaro Performance tools

Characterize and understand the performance of HPC application runs



Commercially  
supported by Linaro

Gather a rich set of data

- Analyses metric around CPU, memory, IO, hardware counters, etc.
- Possibility for users to add their own metrics



Accurate and  
Astute insight

Build a culture of application performance & efficiency awareness

- Analyses data and reports the information that matters to users
- Provides simple guidance to help improve workloads' efficiency

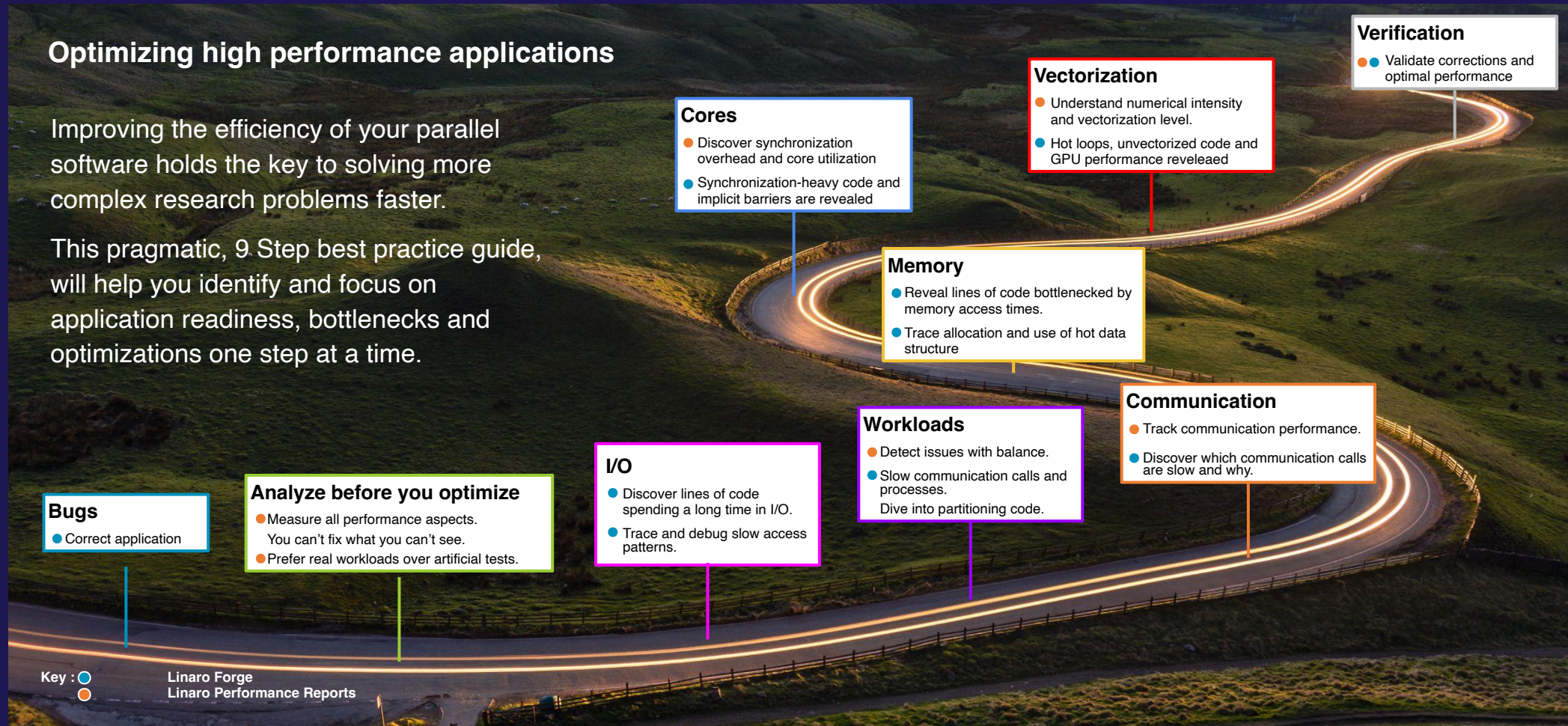


Relevant advice  
to avoid pitfalls

Adds value to typical users' workflows

- Define application behaviour and performance expectations
- Integrate outputs to various systems for validation (eg. continuous integration)
- Can be automated completely (no user intervention)

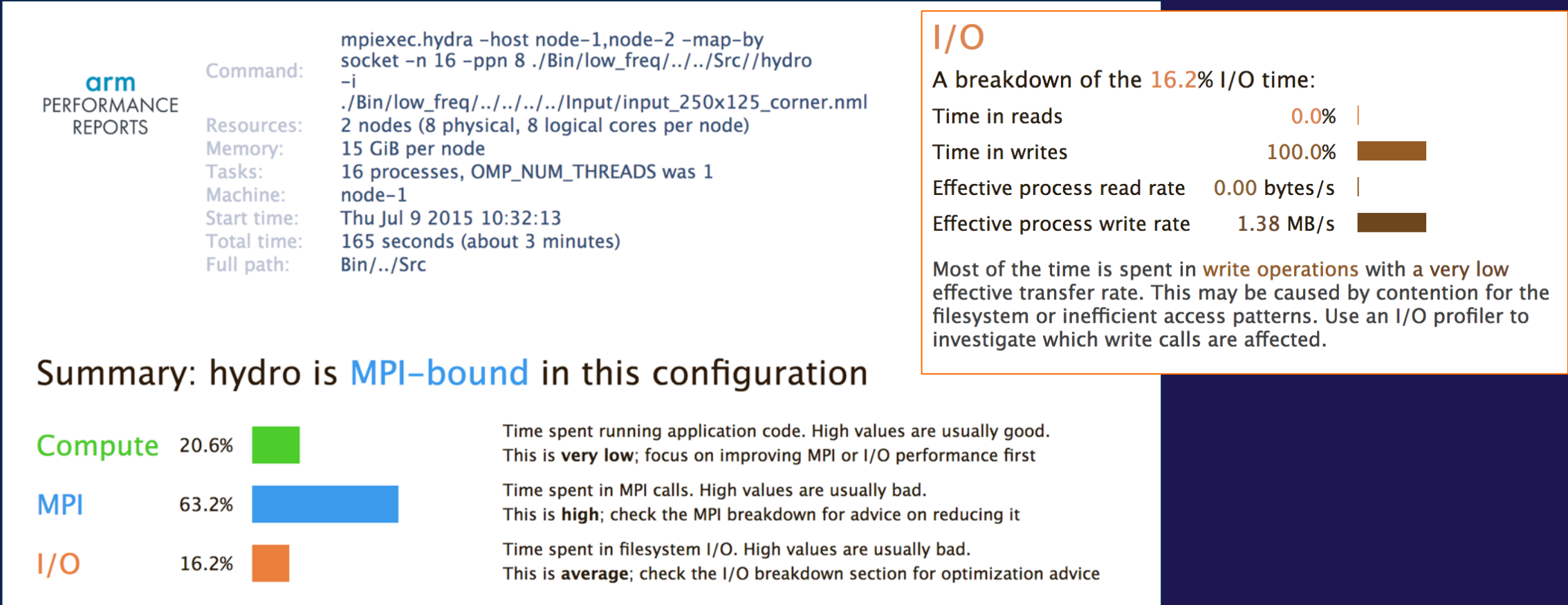
# 9 Step guide for optimising code





# Linaro Performance Reports

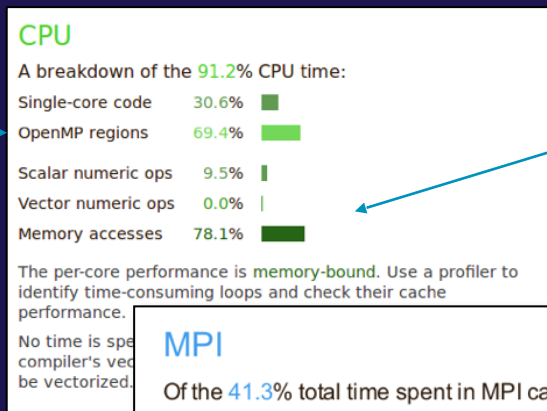
A high-level view of application performance with “plain English” insights



# Linaro Performance Reports Metrics

Lowers expertise requirements by explaining everything in detail right in the report

Multi-threaded  
parallelism



SIMD  
parallelism

## MPI

Of the 41.3% total time spent in MPI calls:

Time in collective calls	100.0%	■
Time in point-to-point calls	0.0%	■
Estimated collective rate	4.07 bytes/s	■
Estimated point-to-point rate	0 bytes/s	■

Load  
imbalance

## OpenMP

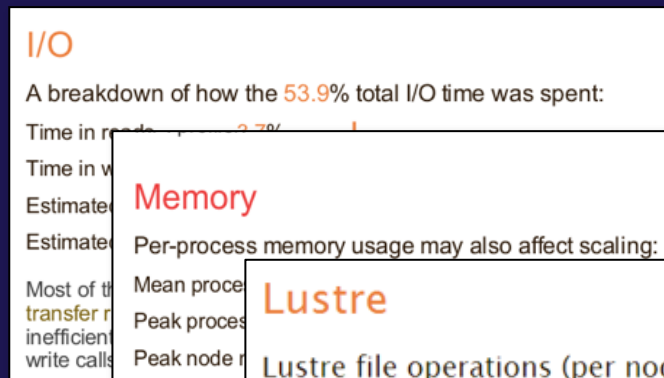
A breakdown of the 99.5% time in OpenMP regions:

Computation	58.9%	■
Synchronization	41.1%	■
Physical core utilization	100.0%	■
System load	99.7%	■

Significant time is spent **synchronizing** threads in parallel regions. Check the affected regions with a profiler.

This may be a sign of overly fine-grained parallelism (OpenMP regions in tight loops) or workload imbalance.

System  
usage



## Memory

Per-process memory usage may also affect scaling:

Mean process memory usage: 1.07 GB

Peak process memory usage: 1.07 GB

Peak node memory usage: 1.07 GB

## Lustre

Lustre file operations (per node)

Mean write rate: 1.07 MB/s

Peak write rate: 1.07 MB/s

Mean file operations: 1.07 MB/s

Mean metadata operations: 1.07 MB/s

## Energy

A breakdown of how the 32.3 Wh was used:

CPU	61.9%	■
System	38.1%	■
Mean node power	94.1 W	■
Peak node power	98.0 W	■

Significant time is spent waiting for memory accesses. Reducing the **CPU** clock frequency could reduce overall energy usage.



# MAP Capabilities

MAP is a sampling based scalable profiler

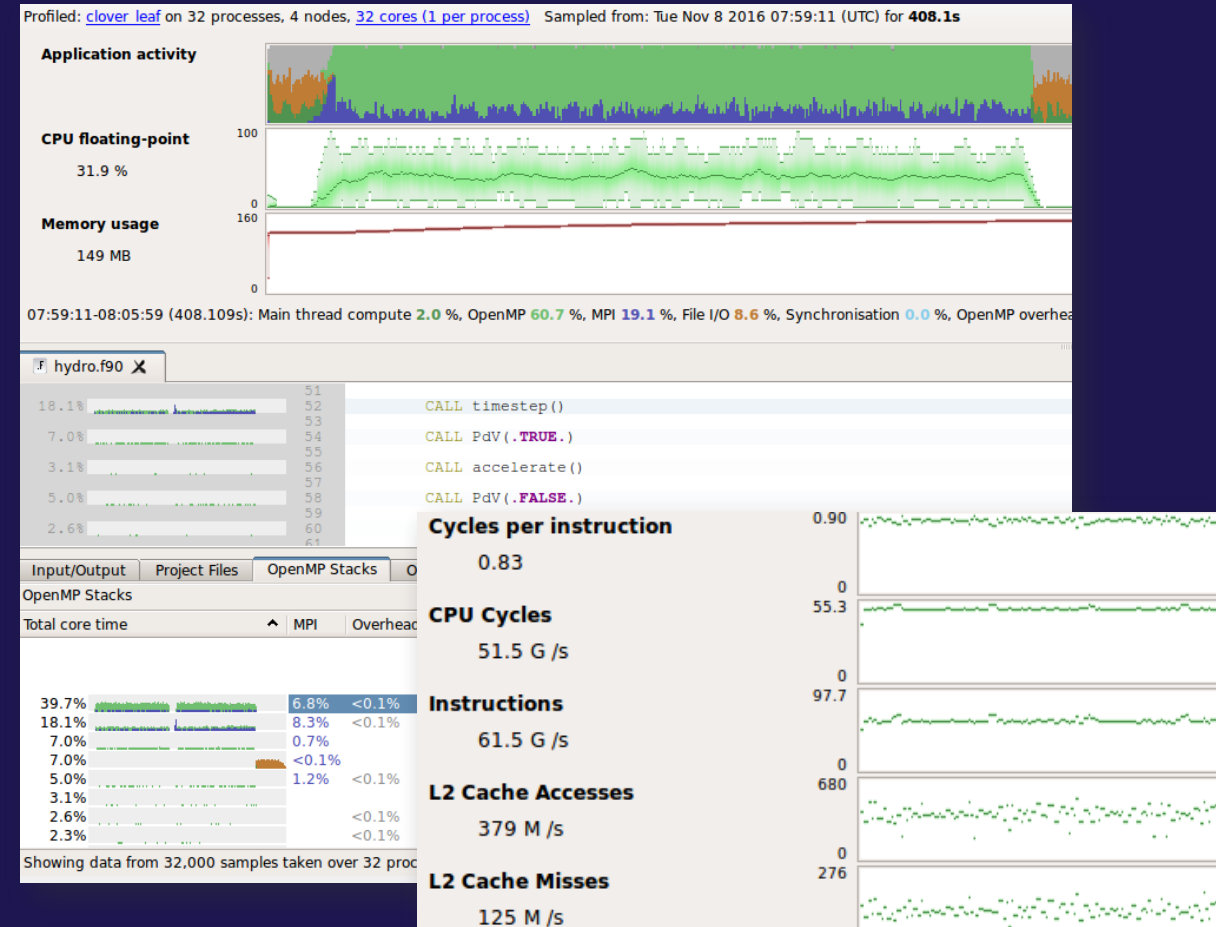
- Built on same framework as DDT
- Parallel support for MPI, OpenMP, CUDA
- Designed for C/C++/Fortran

Designed for 'hot-spot' analysis

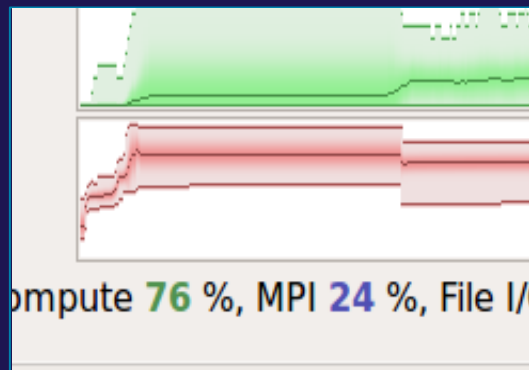
- Stack traces
- Augmented with performance metrics

Adaptive sampling rate

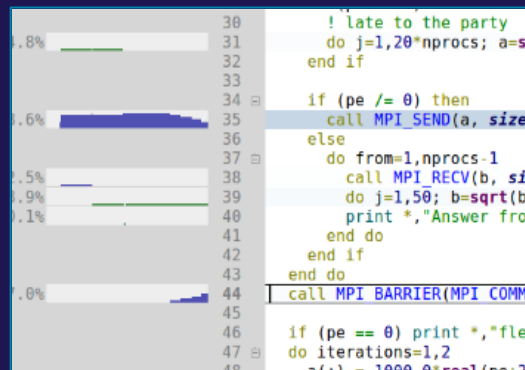
- Throws data away - 1,000 samples per process
- Low overhead, scalable and small file size



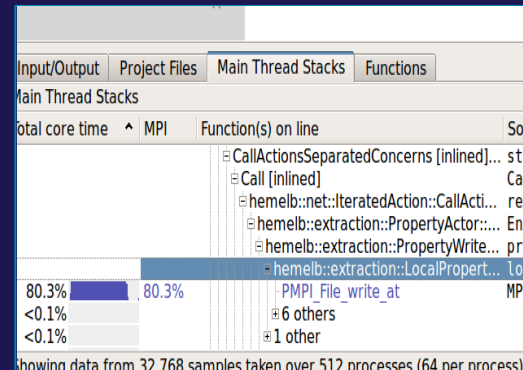
# MAP Highlights



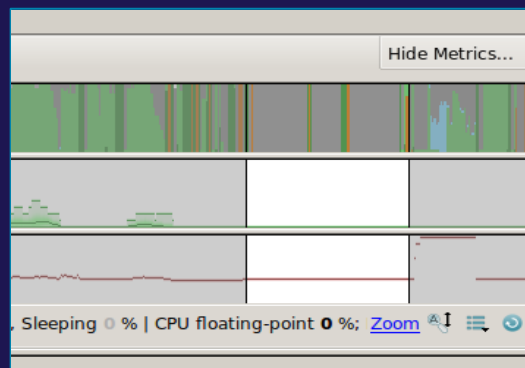
Find the peak memory use



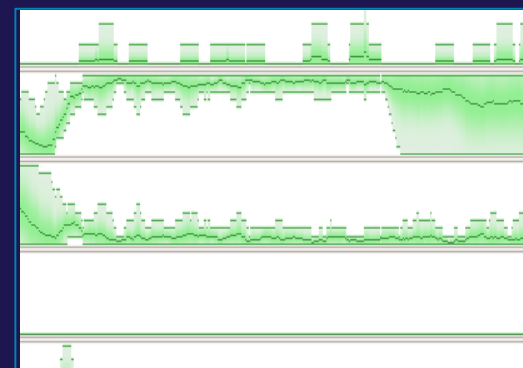
Fix an MPI imbalance



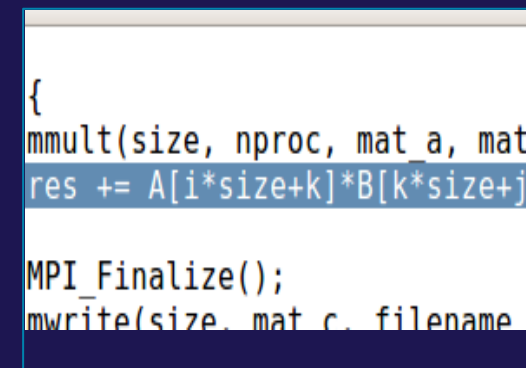
Remove I/O bottleneck



Make sure OpenMP regions make sense

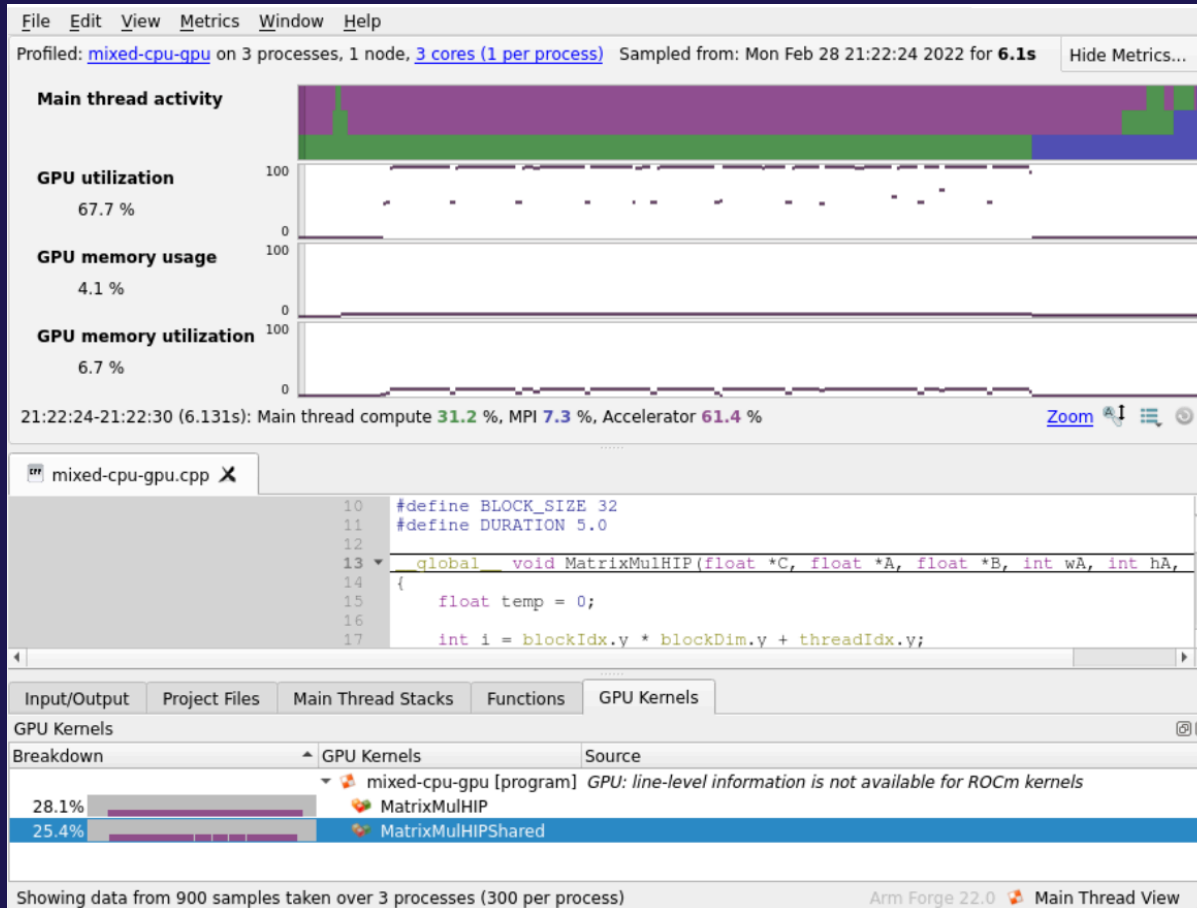


Improve memory access



Restructure for vectorization

# GPU profiling



## Profile

- Supports both AMD and Nvidia GPUs
- Able to bring up metadata of the profile
- Mixed CPU [green] / GPU [purple] application
- CPU time waiting for GPU Kernels [purple]
- GPU Kernels graph indicating Kernel activity

## GUI information

- GUI is consistent across platforms
- Zoom into main thread activity
- Ranked by highest contributors to app time

# Python Profiling

19.0 adds support for Python

- Call stacks
- Time in interpreter

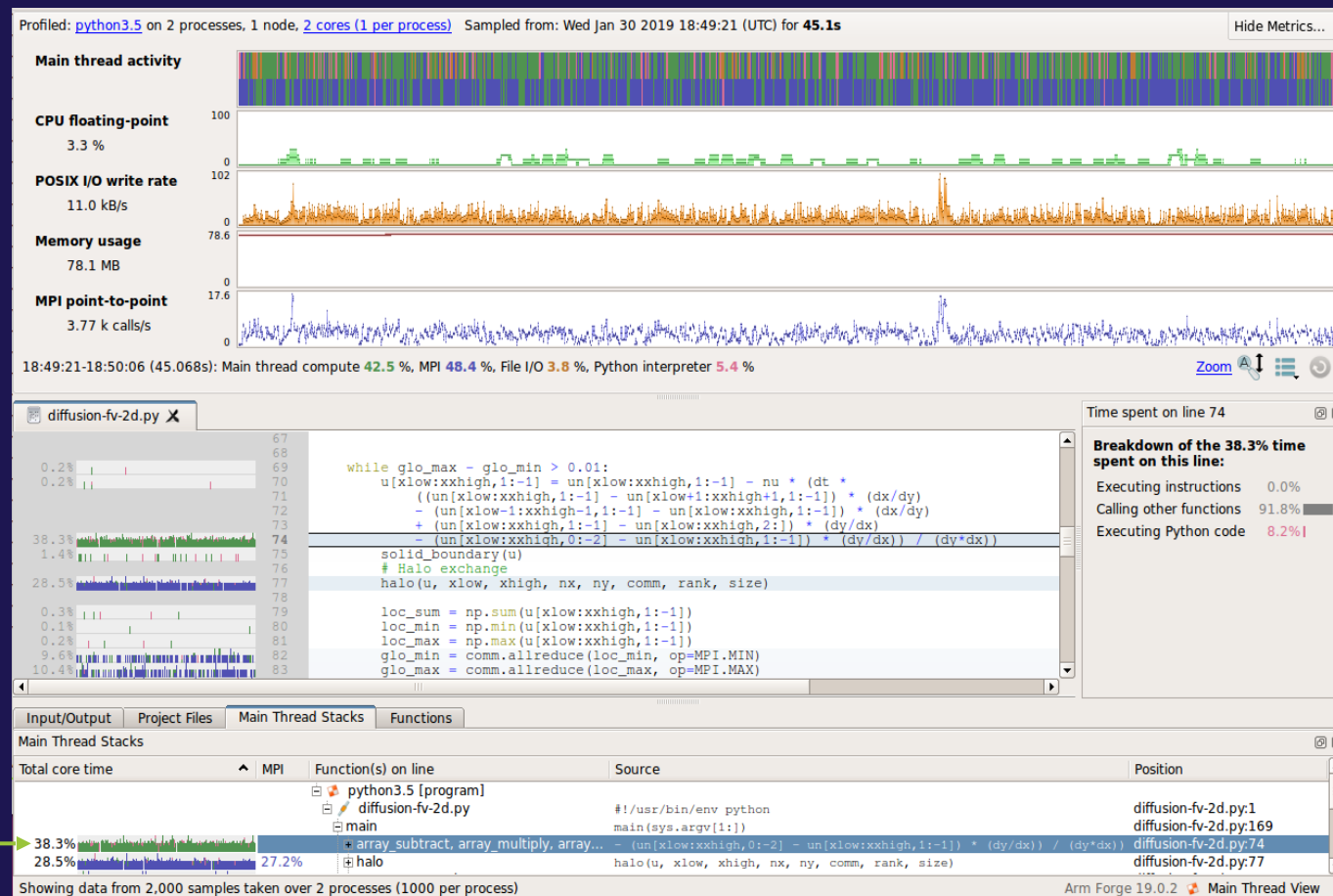
Works with MPI4PY

- Usual MAP metrics

Source code view

- Mixed language support

Note: Green as operation is on numpy array, so backed by C routine, not Python (which would be pink)



```
map --profile mpiexec -n 2 python ./diffusion-fv-2d.py
```

# Compiler Remarks

## Annotates source code with compiler remarks

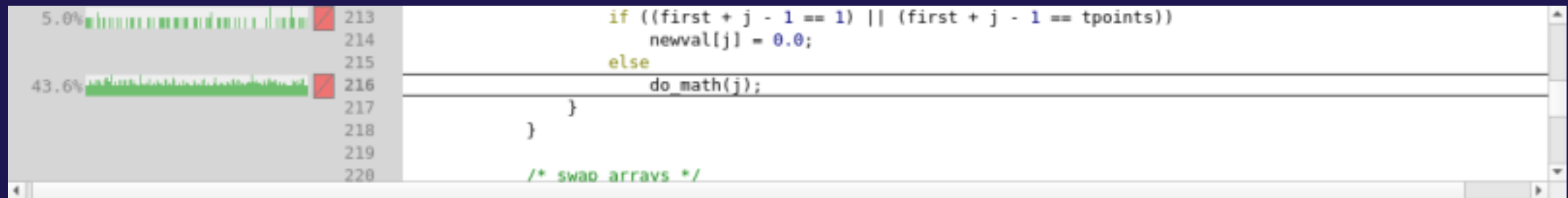
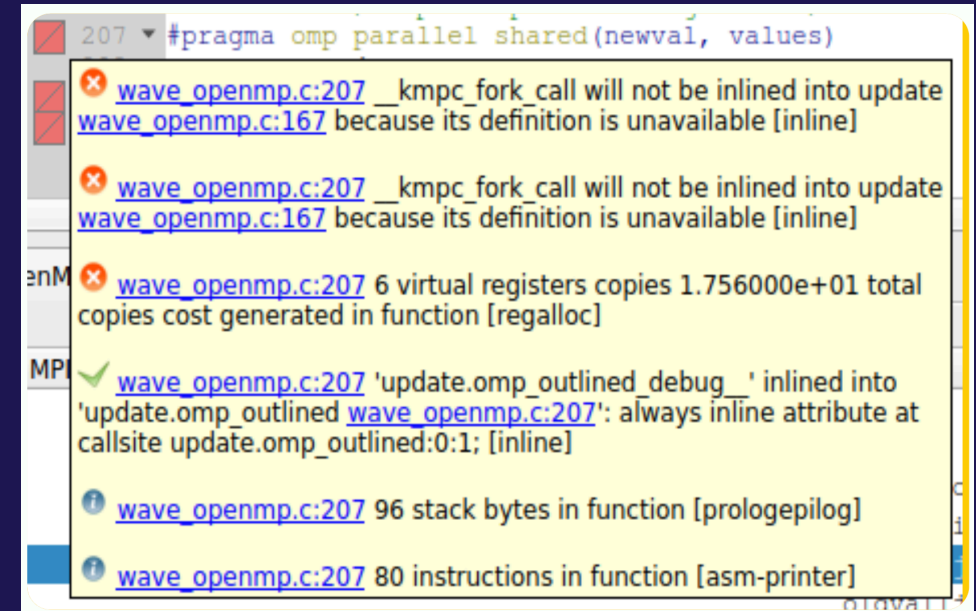
- Remarks are extracted from the compiler optimisation report
- Compiler remarks are displayed as annotations next to your source code

## Colour coded

- Their colour indicates the type of remark present in the following priority order:
- Red: failed or missed optimisations
- Green: successful or passed optimisations
- White: information or analysis notes

## Compiler Remarks menu.

- Specify build directories for non-trivial build systems
- Filter out remarks



# MAP Thread Affinity Advisor

List of Environment Variables which were set at launch which might be relevant to how threads are distributed.

List of Environment Variables which might affect the affinity of a given rank.

Launch Command:

```

run -n 16 python3 /global/homes/r/rshand/linaro-forge-training/performance/mmtul.py -s 3072

```

Process Command:

```

Select an individual process

```

Global (launcher) environment variables:

```


```

Exemplar node's topology (shading shows process affinity bindings):

OpenMP

Submission Script

Other

SLURM\_CPUS\_PER\_TASK

16

SLURM\_NPROCS

16

SLURM\_NTASKS

16

SLURM\_NTASKS\_PER\_NODE

16

SLURM\_TRES\_PER\_TASK

cpu:16

Machine

Package

NUMANode #0

L3Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

NUMANode #1

L3Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

NUMANode #2

L3Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L2Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

L1Cache

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Core

Process-specific env vars (ranks 0,4):

SLURMD\_DEBUG

2 (r

SLURM\_CPU\_BIND

quie

SLURM\_CPU\_BIND\_LIST

0x0

SLURM\_CPU\_BIND\_TYPE

mas

SLURM\_CPU\_BIND\_VERBOSE

quie

SLURM\_DISTRIBUTION

bloc

SLURM\_GTIDS

0,1;

SLURM\_LAUNCH\_NODE\_IPADDR

128

SLURM\_LOCALID

4 (r

SLURM\_MPL\_TYPE

cray

Commentary:

[ERROR]

nid004343, ranks 0-15 (processes 1166384-1166385,1166387,1166389,1166391,1166393-1166394,1166397,1166399,1166401,1166403,1166405,1166407,1166409,1166411,1166413)

contain at least one compute thread which has an overlapping thread affinity mask with another compute thread. e.g. threads 1166391 and 1166929.

[INFORMATION]

nid004343, number of threads allocated to node may be less than ideal. 48 are currently allocated, but consider using 128 (1 per core) for improved utilization.

Data taken at: Finalization

Available exemplar nodes:

nid004343 (0 similar nodes)

Processes on exemplar node:

Rank 0 (PID 1166384)

Rank 1 (PID 1166385)

Rank 2 (PID 1166387)

Rank 3 (PID 1166389)

Rank 4 (PID 1166391)

Rank 5 (PID 1166393)

Threads in selected processes:

✓ pthread (LWP 1167177) 000

✓ pthread (LWP 1166919) 000

✓ Main thread (LWP 1166384) 000

✓ pthread (LWP 1167181) 032

✓ pthread (LWP 1166929) 032

✓ Main thread (LWP 1166391) 032

Change at which point of a run the Affinity data is shown (*Library Load, Initialisation, Finalization*).

**Exemplar nodes**  
Selectable list of exemplars, allowing ability to switch data between nodes of a run. Nodes with similar affinity/structures are merged.

List of processes (by MPI rank) of the selected exemplar. Shows the key for the node topology diagram and selecting one shows all threads for the process.

List of all threads for the selected process. Selecting threads highlights which cores they are bound to in the topology view.

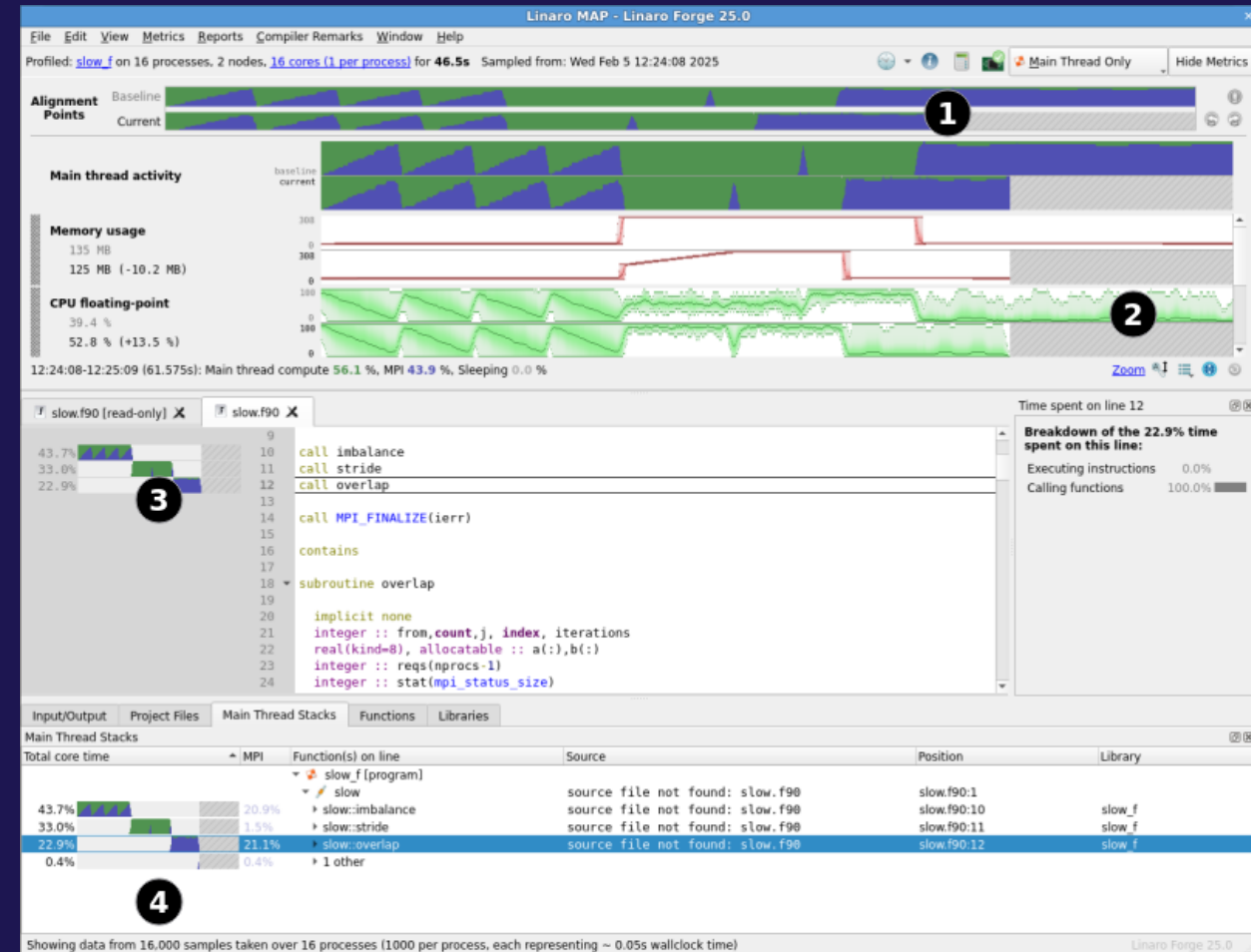
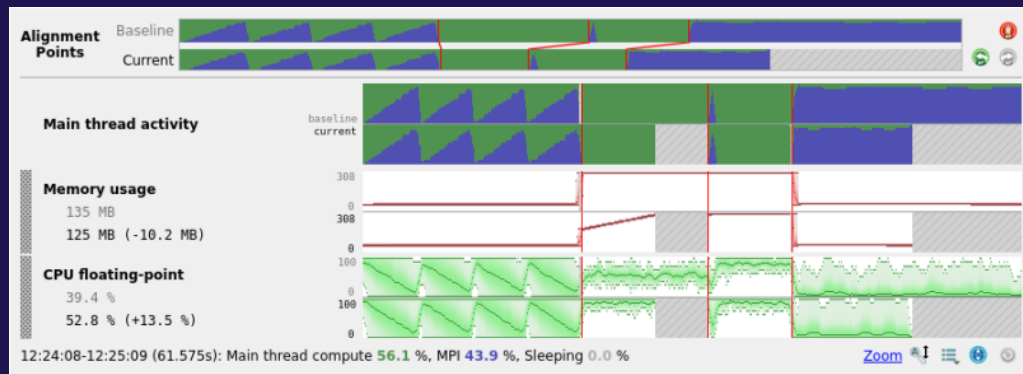
A list of commentary, providing information and advice on Memory Imbalance, Core Utilization etc.



# Differences between two profiles

## MAP Diff (--baseline support)

- MAP Diff allows comparisons of two MAP profiles, useful for identifying performance changes between different parameters, compilers, libraries and systems.
- Use the alignment points view (1) to line up phases of execution.
- Compare metric graphs of the two profiles (2), including metric summaries for each.
- See gaps in activity in the source code viewer (3) and stacks views (4), including OpenMP Regions View, Functions View, Library view and GPU Kernels/Memory Transfer View.



# Debug with DDT on Aurora

```
mpicxx -fsycl -g -O0 <application> -o <application-name>
```

```
qsub -l select=2 -l walltime=30:00 -l filesystems=flare -A <account> -q <queue> -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" ./<application>
```

# Cheat sheet (Polaris)

## ***Training material***

### *1. Getting the examples*

[google-drive](#)

```
tar -xf linaro-forge-training.tar.gz
```

### *2. Set the path to the forge training folder*

```
export FORGE_TRAINING=<path_to_training_folder>
```

## ***Forge Client (On local machine)***

Install Forge client <https://www.linaroforge.com/downloadForge>

## ***Running with a batch script***

```
qsub $FORGE_TRAINING/submit-polaris.sh
```

## ***Interactive Session***

```
qsub -l -l select=1 -l filesystems=home:eagle -l  
walltime=0:30:00 -q alcf_training -A alcf_training
```

```
module use /soft/modulefiles
```

```
module load forge cray-cti/2.19.0
```

## ***Forge commands***

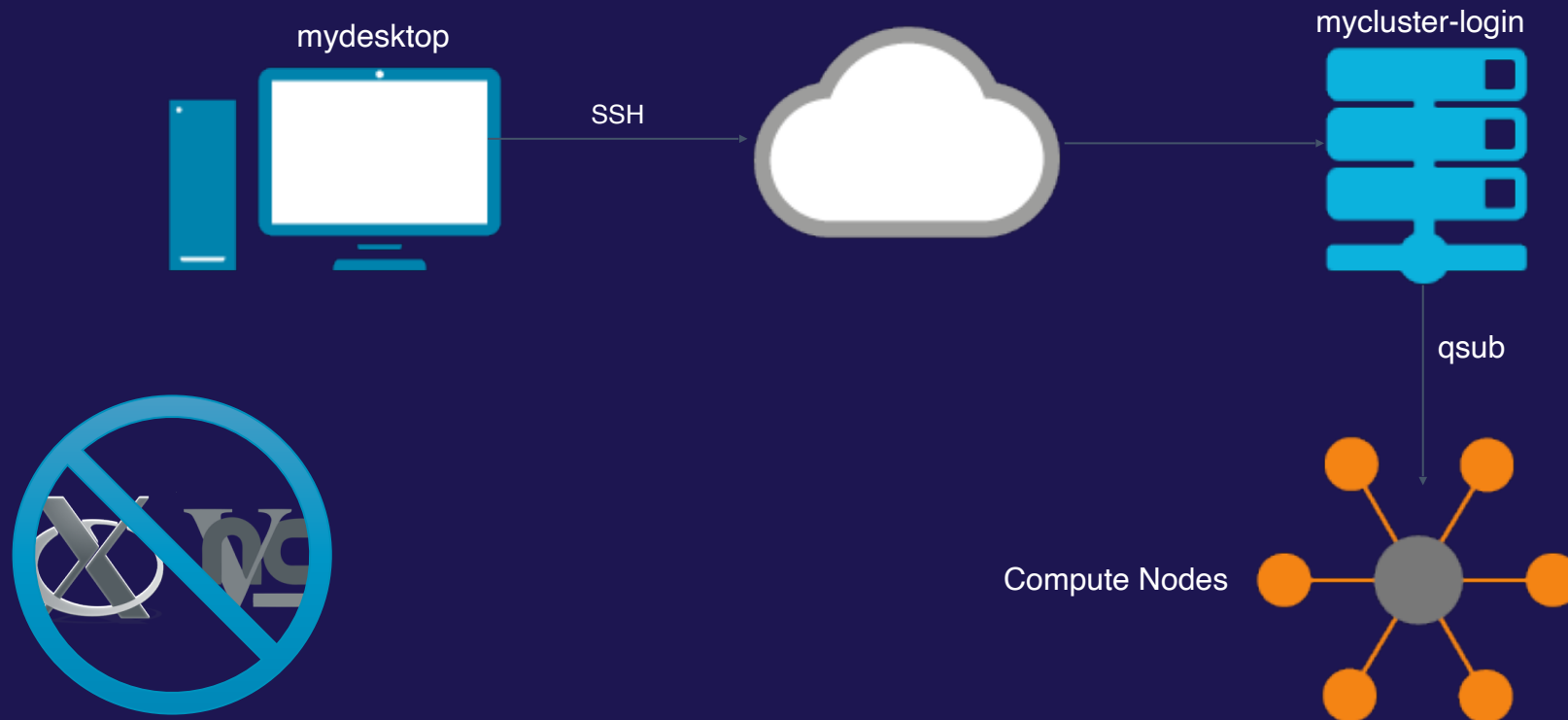
<code>ddt --connect</code>	<i># Reverse connect</i>
<code>ddt --offline</code>	<i># Run DDT without GUI</i>
<code>map --profile</code>	<i># Profile without GUI</i>
<code>perf-report</code>	<i># Generate Performance Report</i>

## ***Guides***

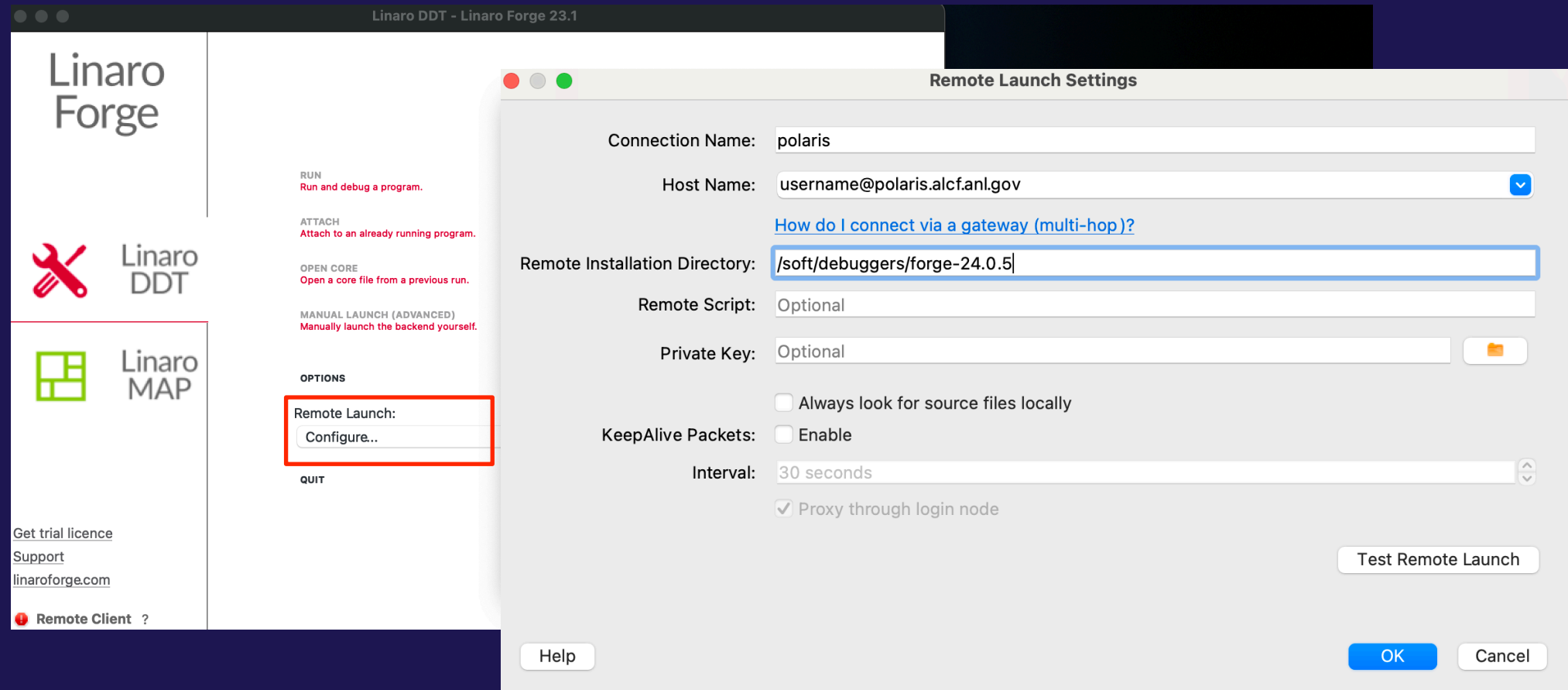
[Forge userguide](#)

# The Forge GUI and where to run it

Forge provides a powerful GUIs that can be run in a variety of configurations



# Remote connection to Polaris



# Debugging (Polaris)

## 1. *build deadlock, simple, memory\_debugging and split examples*

```
cd $FORGE_TRAINING/correctness/debug
make
```

## 2. *Get an interactive session*

```
qsub -l -l select=1 -l filesystems=home:eagle -l walltime=0:30:00 -q alcf_training -A alcf_training
module use /soft/modulefiles
module load forge cray-cti/2.19.0
```

## 3. *split*

```
ddt --connect mpiexec -n 16 ./split
```

## 4. *2\_layer\_net.py*

```
FORGE_MPIRUN=torchrun \
FORGE_STOP_AT_MAIN=1 FORGE_MPIRUN_OMIT_NUMBER_OF_PROCESSES=1 \
ddt --connect --np 2 --mpi="generic" \
--mpiargs="--no-python --standalone --nnodes=1 --nproc-per-node=2" $(which python) %allinea_python_debug% 2_layer_net.py
```



# Profiling (Polaris)

Worked Example: [https://docs.linaroforge.com/23.1.1/html/forge/worked\\_examples\\_appendix/mmult/analyze.html](https://docs.linaroforge.com/23.1.1/html/forge/worked_examples_appendix/mmult/analyze.html)

## 1. Setup the environment

```
qsub -l -l select=1 -l filesystems=home:eagle -l walltime=0:30:00 -q alcf_training -A alcf_training  
module use /soft/modulefiles  
module load forge cray-cti  
. .venv/bin/activate
```

## 2. Build the Python example

```
cd $FORGE_TRAINING/performance  
make -f mmult_py.makefile
```

## 3. Run the Python example

```
map --profile mpiexec -n 8 python ./mmult.py -s 3072
```

# Thank you

- [www.linaroforge.com](http://www.linaroforge.com)
- [support@forge.linaro.com](mailto:support@forge.linaro.com)
- <https://docs.linaroforge.com/24.0.5/html/forgel/index.html>