ARGONNE

ATPESC2024

EXTREME-SCALE COMPUTING

Linaro Forge

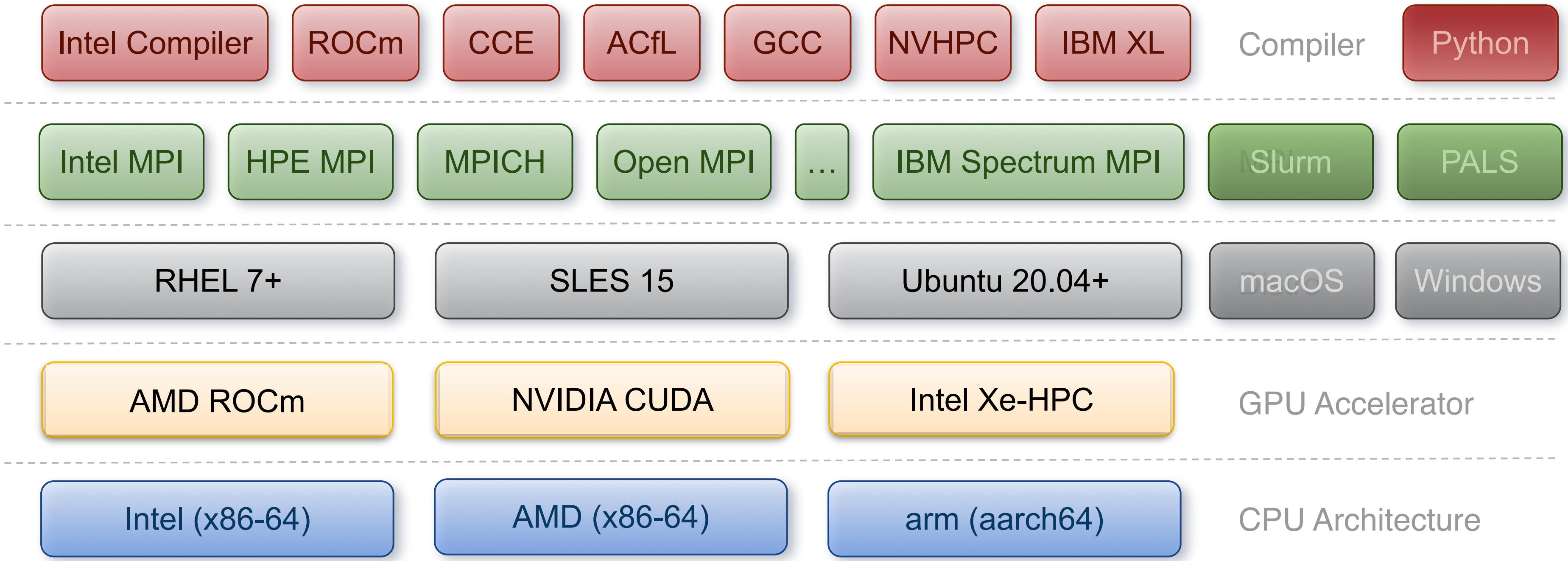
Rudy Shand
Field Application Engineer, Linaro Ltd

Agenda

- 15 minutes Overview of DDT
- 45 minutes DDT hands-on
- 15 minutes Overview of MAP and Performance Reports
- 30 minutes MAP and Performance Reports hands-on

DDT Supported Platforms

Works across hardware architectures and HPC technologies



Intel Compiler · ROCm · CCE · ACfL · GCC · NVHPC · IBM XL · Compiler · Python

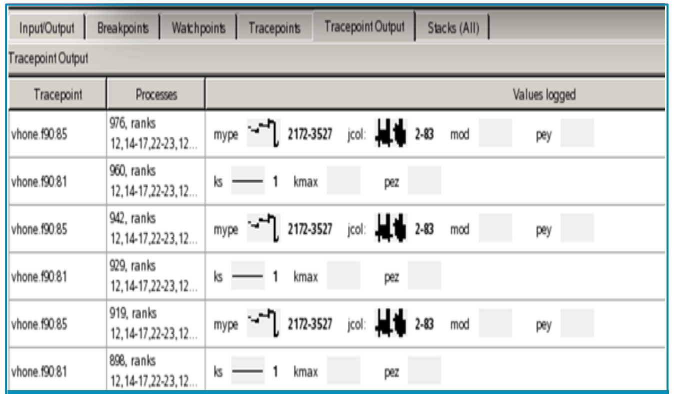
Intel MPI · HPE MPI · MPICH · Open MPI · … · IBM Spectrum MPI · Slurm · PALS

RHEL 7+ · SLES 15 · Ubuntu 20.04+ · macOS · Windows

AMD ROCm · NVIDIA CUDA · Intel Xe-HPC · GPU Accelerator

Intel (x86-64) · AMD (x86-64) · arm (aarch64) · CPU Architecture

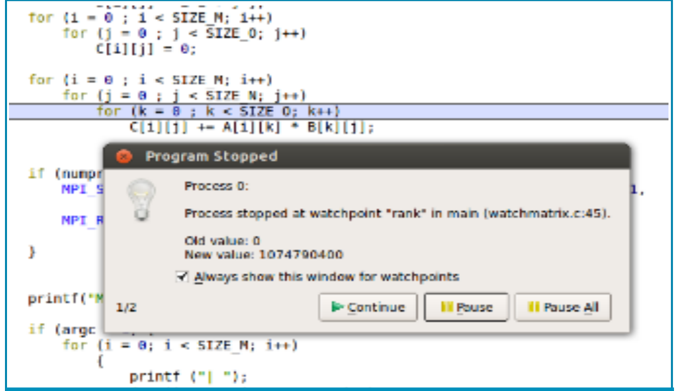
DDT Highlights



Tracepoint Output

Tracepoint	Processes	Values logged
vhone.f90:85	976, ranks 12,14-17,22-23,12...	mype 2172-3527 jcol: 2-83 mod pey
vhone.f90:81	960, ranks 12,14-17,22-23,12...	ks 1 kmax pez
vhone.f90:85	942, ranks 12,14-17,22-23,12...	mype 2172-3527 jcol: 2-83 mod pey
vhone.f90:81	929, ranks 12,14-17,22-23,12...	ks 1 kmax pez
vhone.f90:85	919, ranks 12,14-17,22-23,12...	mype 2172-3527 jcol: 2-83 mod pey
vhone.f90:81	898, ranks 12,14-17,22-23,12...	ks 1 kmax pez

The scalable print alternative



```
for (i = 0 ; i < SIZE_M; i++)
    for (j = 0 ; j < SIZE_O; j++)
        C[i][j] = 0;

for (i = 0 ; i < SIZE_M; i++)
    for (j = 0 ; j < SIZE_N; j++)
        for (k = 0 ; k < SIZE_O; k++)
            C[i][j] += A[i][k] * B[k][j];


if (numpr
    MPI_S                                                    1,

    MPI_R

}


printf("M

if (argc
    for (i = 0; i < SIZE_M; i++)
        {
            printf ("| ");
```

Program Stopped

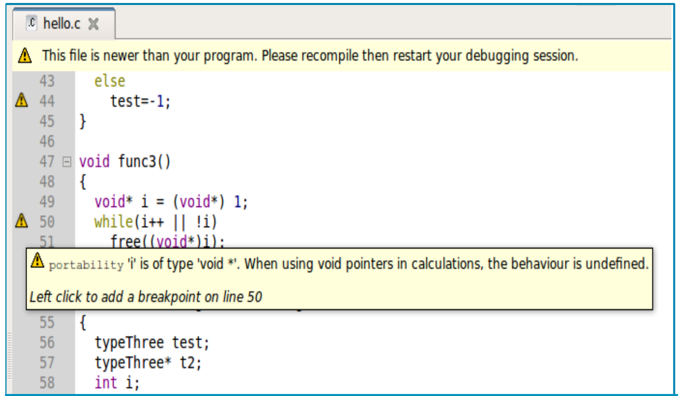
Process 0:

Process stopped at watchpoint "rank" in main (watchmatrix.c:45).

Old value: 0
New value: 1074790400

☑ Always show this window for watchpoints

Stop on variable change



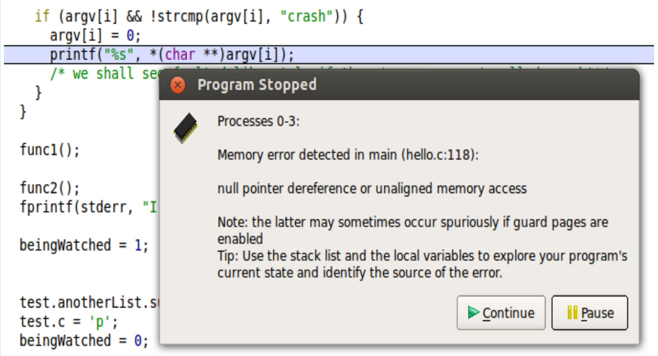
⚠ This file is newer than your program. Please recompile then restart your debugging session.

```
43      else
44 ⚠      test=-1;
45    }
46
47    void func3()
48    {
49      void* i = (void*) 1;
50 ⚠    while(i++ || !i)
51        free((void*)i);
55    {
56      typeThree test;
57      typeThree* t2;
58      int i;
```

⚠ portability 'i' is of type 'void *'. When using void pointers in calculations, the behaviour is undefined.

Left click to add a breakpoint on line 50

Static analysis warnings on code errors



```
  if (argv[i] && !strcmp(argv[i], "crash")) {
    argv[i] = 0;
    printf("%s", *(char **)argv[i]);
    /* we shall se
  }
}

func1();

func2();
fprintf(stderr, "I

beingWatched = 1;


test.anotherList.s
test.c = 'p';
beingWatched = 0;
```

Program Stopped

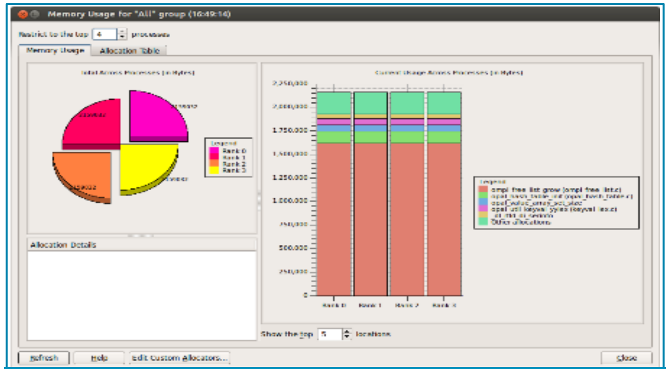
Processes 0-3:

Memory error detected in main (hello.c:118):

null pointer dereference or unaligned memory access

Note: the latter may sometimes occur spuriously if guard pages are enabled
Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the error.

Detect read/write beyond array bounds

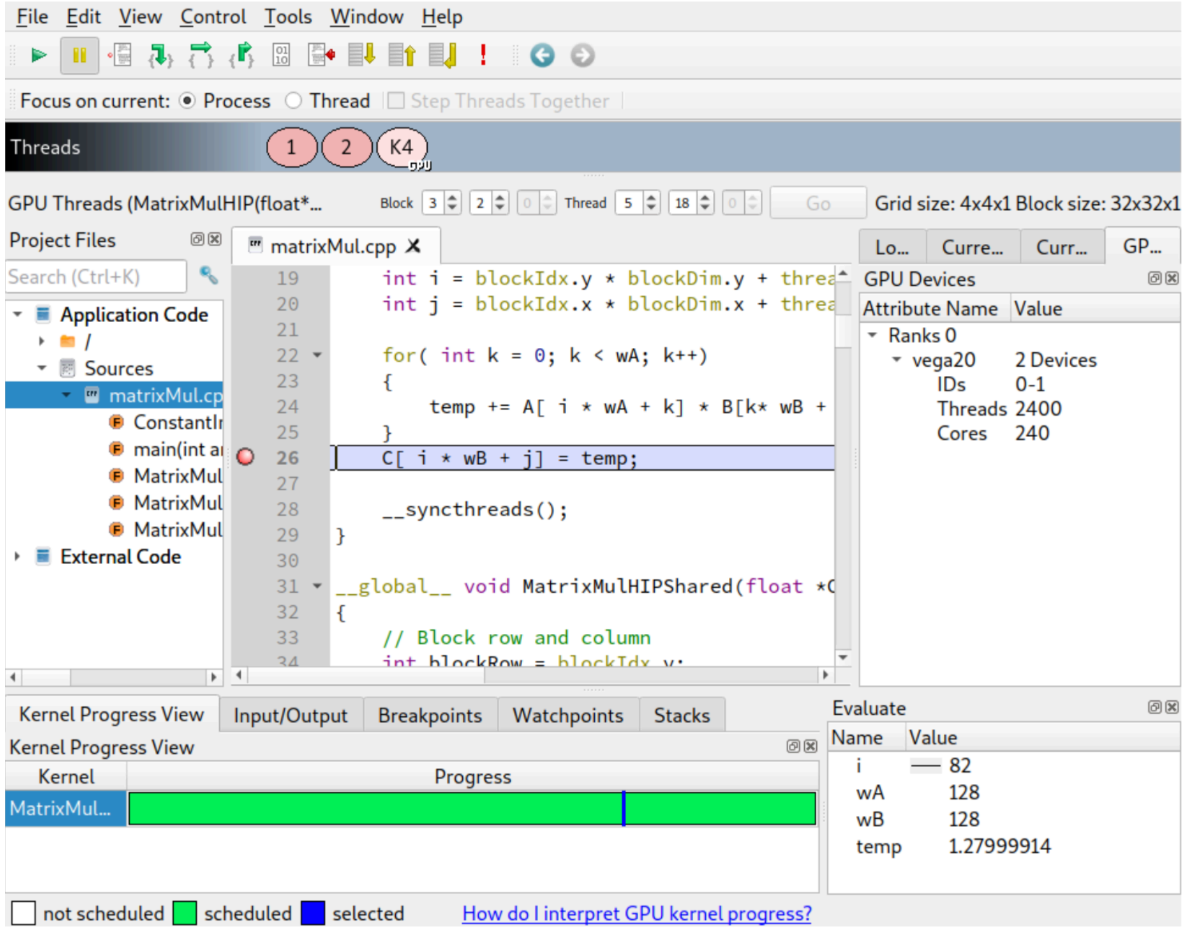


Memory Usage for "All" group (16:49:14)

Restrict to the top 4 processes

Detect stale memory allocations

GPU Debugging



GPU Threads (MatrixMulHIP(float*... Block 3 2 0 Thread 5 18 0 Go

Grid size: 4x4x1 Block size: 32x32x1

```
    int i = blockIdx.y * blockDim.y + threa
    int j = blockIdx.x * blockDim.x + threa

    for( int k = 0; k < wA; k++)
    {
        temp += A[ i * wA + k] * B[k* wB +
    }
    C[ i * wB + j] = temp;

    __syncthreads();
}

__global__ void MatrixMulHIPShared(float *C
{
    // Block row and column
    int blockRow = blockIdx.y;
```

GPU Devices

Attribute Name	Value
Ranks 0	
vega20	2 Devices
IDs	0-1
Threads	2400
Cores	240

Kernel Progress View

Kernel	Progress
MatrixMul...	🟩

□ not scheduled 🟩 scheduled 🟦 selected

Evaluate

Name	Value
i	82
wA	128
wB	128
temp	1.27999914

- Support both AMD and Nvidia GPUs
- Debug simultaneously on GPU and CPU

- Look and feel exactly the same
- Main Features work in GPU

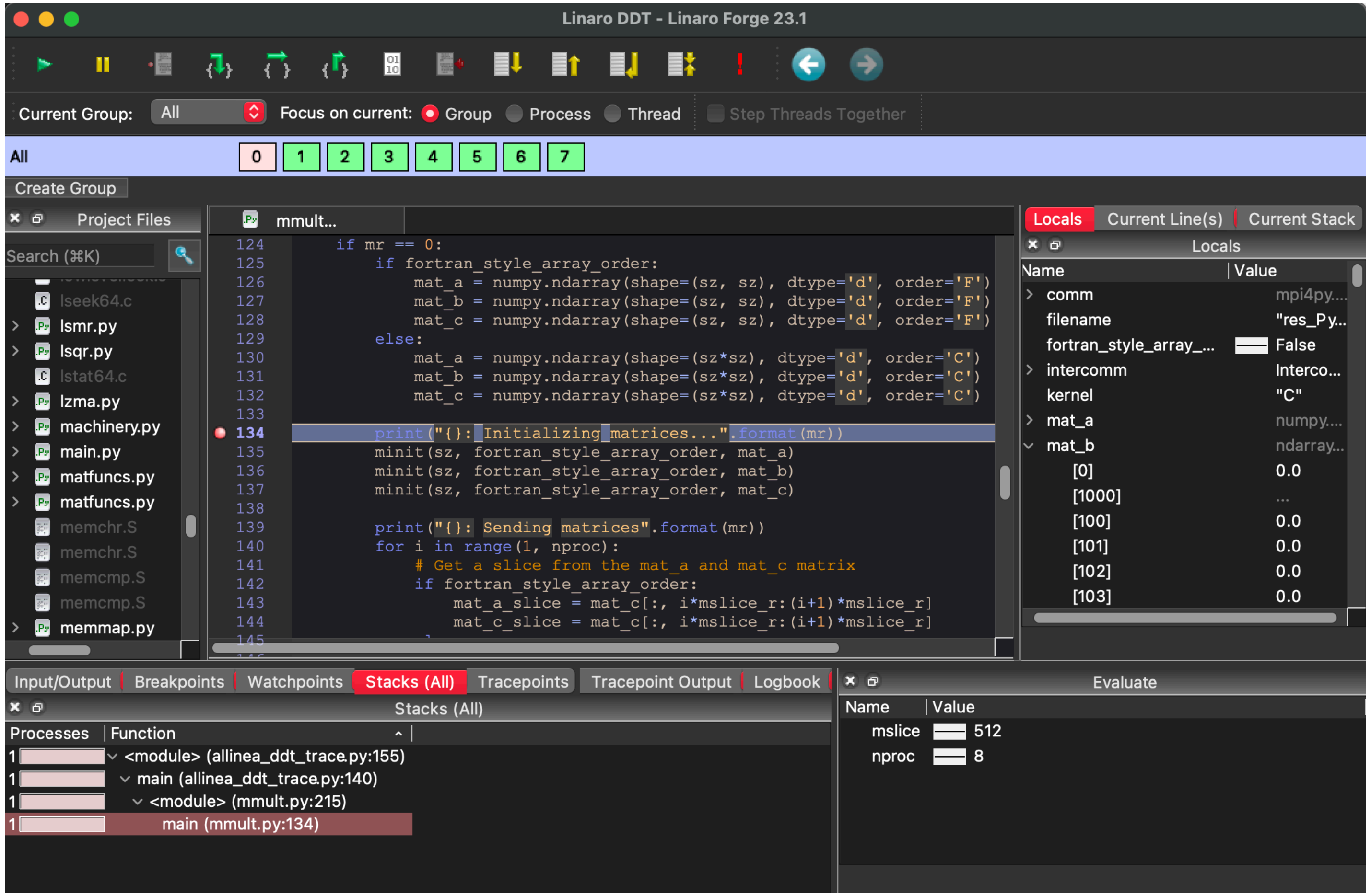
- Key (additional) GPU features:
  - Kernel Progress View
  - GPU thread in parallel stack view
  - GPU Thread Selector
  - GPU Device Pane

- For NVIDIA’s nvcc compiler, kernels must be compiled with the -g -G flags

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 in “Current Line” display
  - Mpi4py, NumPy, SciPy

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



```
    if mr == 0:
        if fortran_style_array_order:
            mat_a = numpy.ndarray(shape=(sz, sz), dtype='d', order='F')
            mat_b = numpy.ndarray(shape=(sz, sz), dtype='d', order='F')
            mat_c = numpy.ndarray(shape=(sz, sz), dtype='d', order='F')
        else:
            mat_a = numpy.ndarray(shape=(sz*sz), dtype='d', order='C')
            mat_b = numpy.ndarray(shape=(sz*sz), dtype='d', order='C')
            mat_c = numpy.ndarray(shape=(sz*sz), dtype='d', order='C')

        print("{}: Initializing matrices...".format(mr))
        minit(sz, fortran_style_array_order, mat_a)
        minit(sz, fortran_style_array_order, mat_b)
        minit(sz, fortran_style_array_order, mat_c)

        print("{}: Sending matrices".format(mr))
        for i in range(1, nproc):
            # Get a slice from the mat_a and mat_c matrix
            if fortran_style_array_order:
                mat_a_slice = mat_c[:, i*mslice_r:(i+1)*mslice_r]
                mat_c_slice = mat_c[:, i*mslice_r:(i+1)*mslice_r]
```

Locals

Name	Value
comm	mpi4py....
filename	"res_Py...
fortran_style_array_...	False
intercomm	Interco...
kernel	"C"
mat_a	numpy....
mat_b	ndarray...
[0]	0.0
[1000]	...
[100]	0.0
[101]	0.0
[102]	0.0
[103]	0.0

Stacks (All)

Processes	Function
1	<module> (allinea_ddt_trace.py:155)
1	main (allinea_ddt_trace.py:140)
1	<module> (mmult.py:215)
1	main (mmult.py:134)

Evaluate

Name	Value
mslice	512
nproc	8

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…

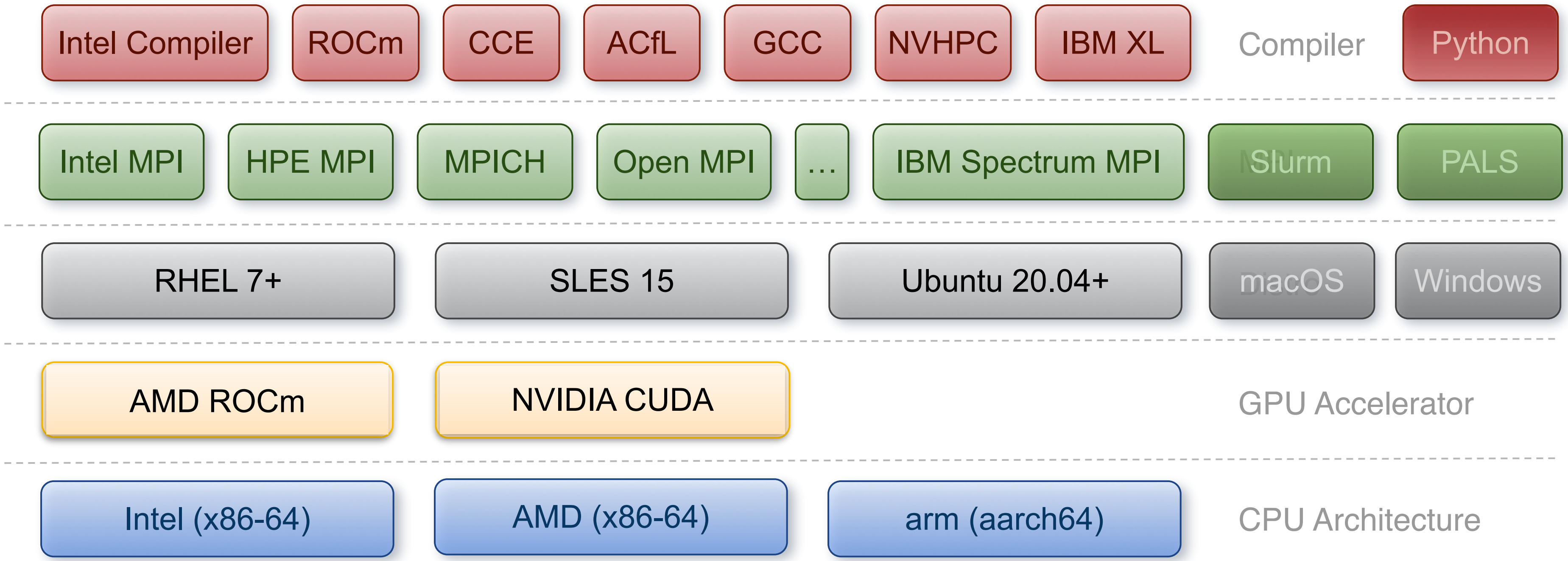
To do so, use following arguments:

- $ 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



Intel Compiler | ROCm | CCE | ACfL | GCC | NVHPC | IBM XL | Compiler | Python

Intel MPI | HPE MPI | MPICH | Open MPI | … | IBM Spectrum MPI | Slurm | PALS

RHEL 7+ | SLES 15 | Ubuntu 20.04+ | macOS | Windows

AMD ROCm | NVIDIA CUDA | GPU Accelerator

Intel (x86-64) | AMD (x86-64) | arm (aarch64) | CPU Architecture

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

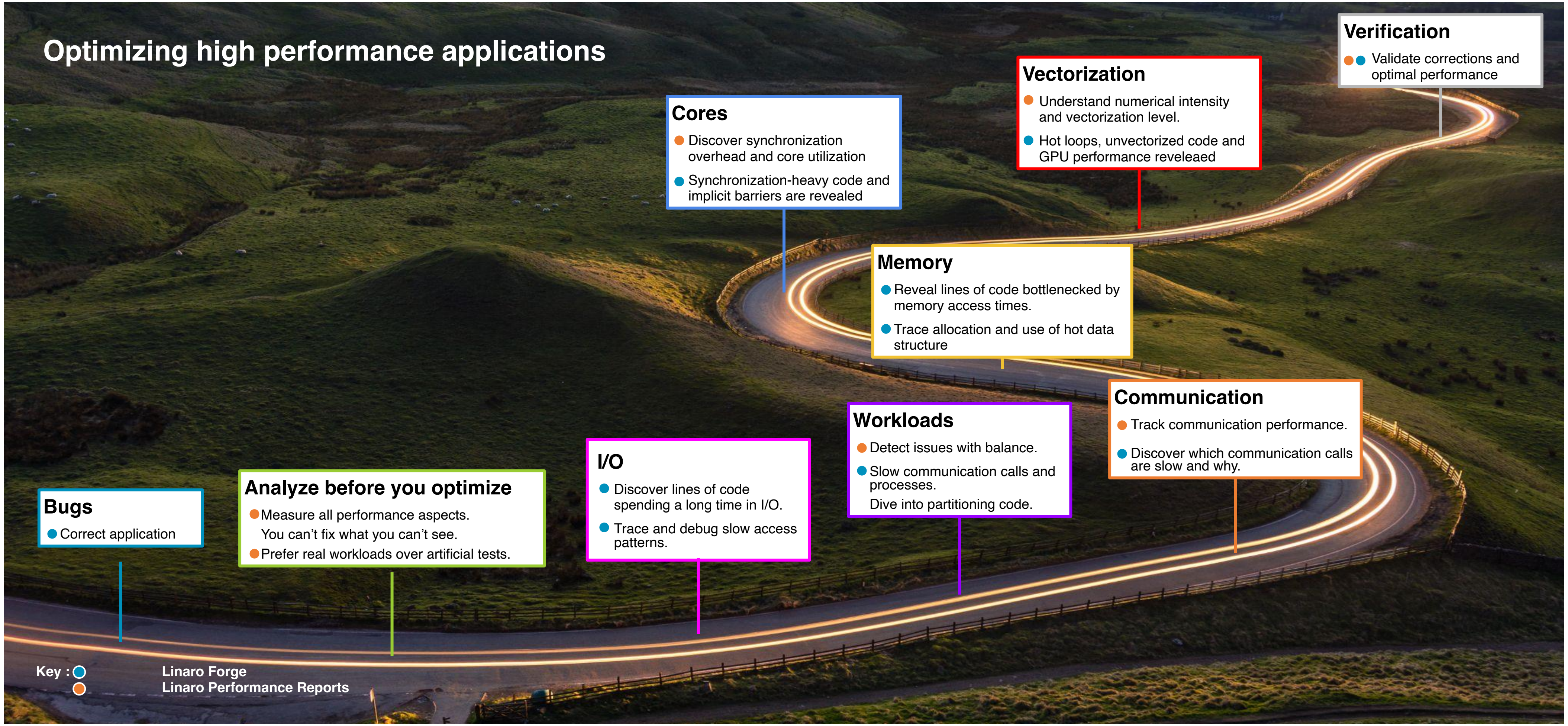
- Analyses data and reports the information that matters to users
- Provides simple guidance to help improve workloads' efficiency

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)

The Performance Roadmap



Optimizing high performance applications

Bugs
- 🔵 Correct application

Analyze before you optimize
- 🟠 Measure all performance aspects. You can't fix what you can't see.
- 🟠 Prefer real workloads over artificial tests.

I/O
- 🔵 Discover lines of code spending a long time in I/O.
- 🔵 Trace and debug slow access patterns.

Workloads
- 🟠 Detect issues with balance.
- 🔵 Slow communication calls and processes. Dive into partitioning code.

Communication
- 🟠 Track communication performance.
- 🔵 Discover which communication calls are slow and why.

Memory
- 🔵 Reveal lines of code bottlenecked by memory access times.
- 🔵 Trace allocation and use of hot data structure

Cores
- 🟠 Discover synchronization overhead and core utilization
- 🔵 Synchronization-heavy code and implicit barriers are revealed

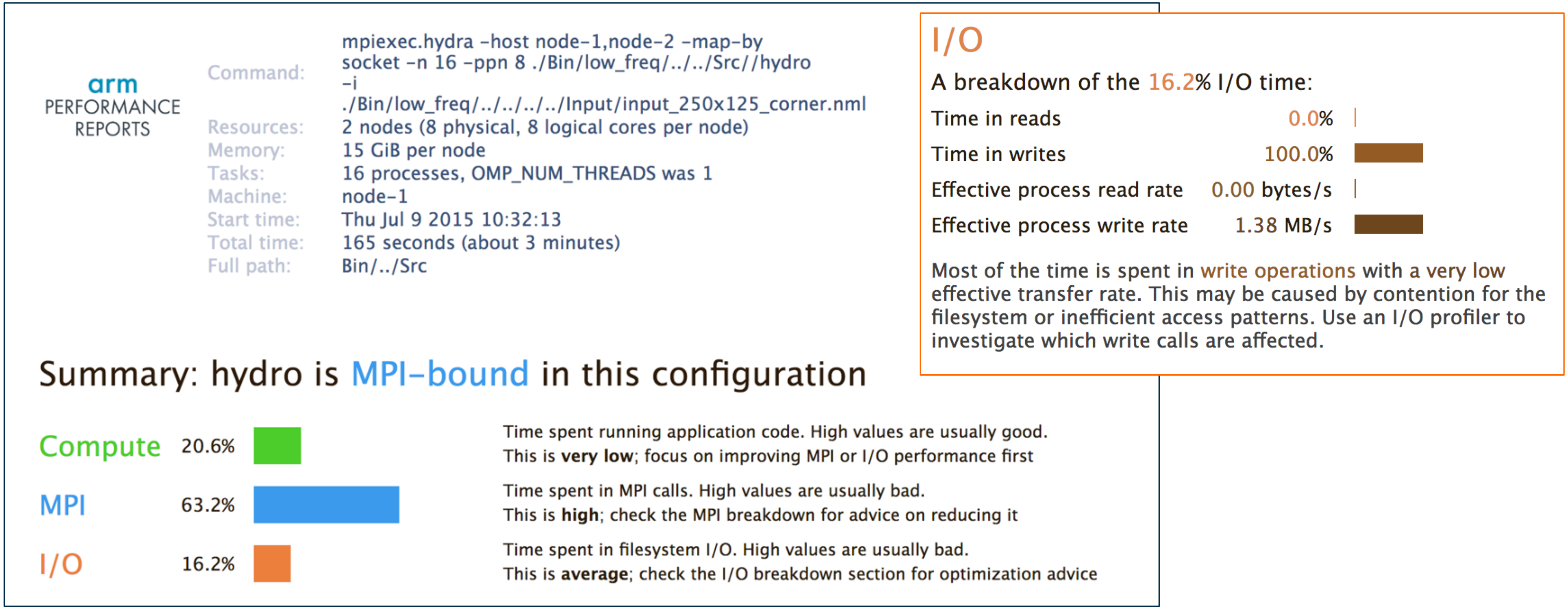
Vectorization
- 🟠 Understand numerical intensity and vectorization level.
- 🔵 Hot loops, unvectorized code and GPU performance reveleaed

Verification
- 🟠🔵 Validate corrections and optimal performance

Key : 🔵 Linaro Forge
🟠 Linaro Performance Reports

Linaro Performance Reports

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



arm PERFORMANCE REPORTS

Command:	mpiexec.hydra -host node-1,node-2 -map-by socket -n 16 -ppn 8 ./Bin/low_freq/../../Src//hydro -i ./Bin/low_freq/../../../../Input/input_250x125_corner.nml
Resources:	2 nodes (8 physical, 8 logical cores per node)
Memory:	15 GiB per node
Tasks:	16 processes, OMP_NUM_THREADS was 1
Machine:	node-1
Start time:	Thu Jul 9 2015 10:32:13
Total time:	165 seconds (about 3 minutes)
Full path:	Bin/../Src

Summary: hydro is MPI-bound in this configuration

Compute	20.6%	Time spent running application code. High values are usually good. This is **very low**; focus on improving MPI or I/O performance first
MPI	63.2%	Time spent in MPI calls. High values are usually bad. This is **high**; check the MPI breakdown for advice on reducing it
I/O	16.2%	Time spent in filesystem I/O. High values are usually bad. This is **average**; check the I/O breakdown section for optimization advice

I/O

A breakdown of the 16.2% I/O time:

Time in reads	0.0%
Time in writes	100.0%
Effective process read rate	0.00 bytes/s
Effective process write rate	1.38 MB/s

Most of the time is spent in write operations with a very low effective transfer rate. This may be caused by contention for the filesystem or inefficient access patterns. Use an I/O profiler to investigate which write calls are affected.

Linaro Performance Reports Metrics

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

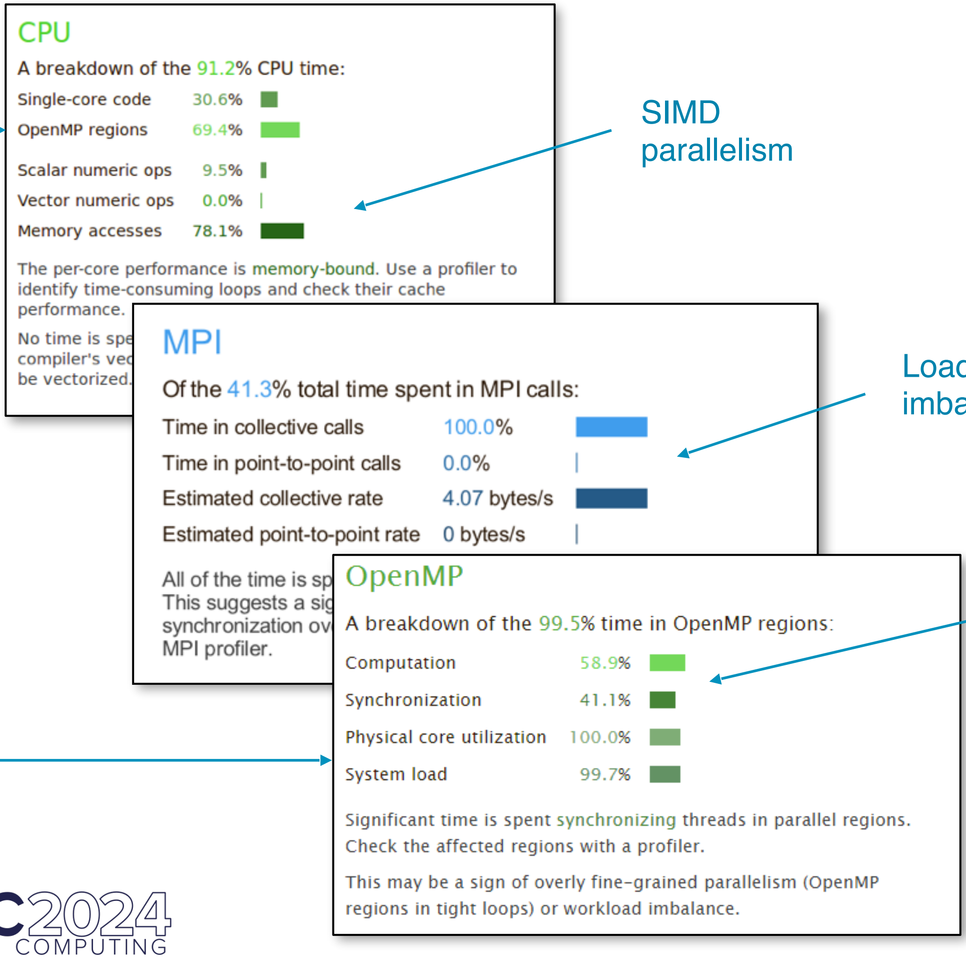
Multi-threaded parallelism

SIMD parallelism

Load imbalance

OMP efficiency

System usage



CPU

A breakdown of the 91.2% CPU time:

Single-core code	30.6%	🟩
OpenMP regions	69.4%	🟩
Scalar numeric ops	9.5%	🟩
Vector numeric ops	0.0%	|
Memory accesses	78.1%	🟩

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

No time is spe
compiler's vec
be vectorized.

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	|

All of the time is sp
This suggests a sig
synchronization ov
MPI profiler.

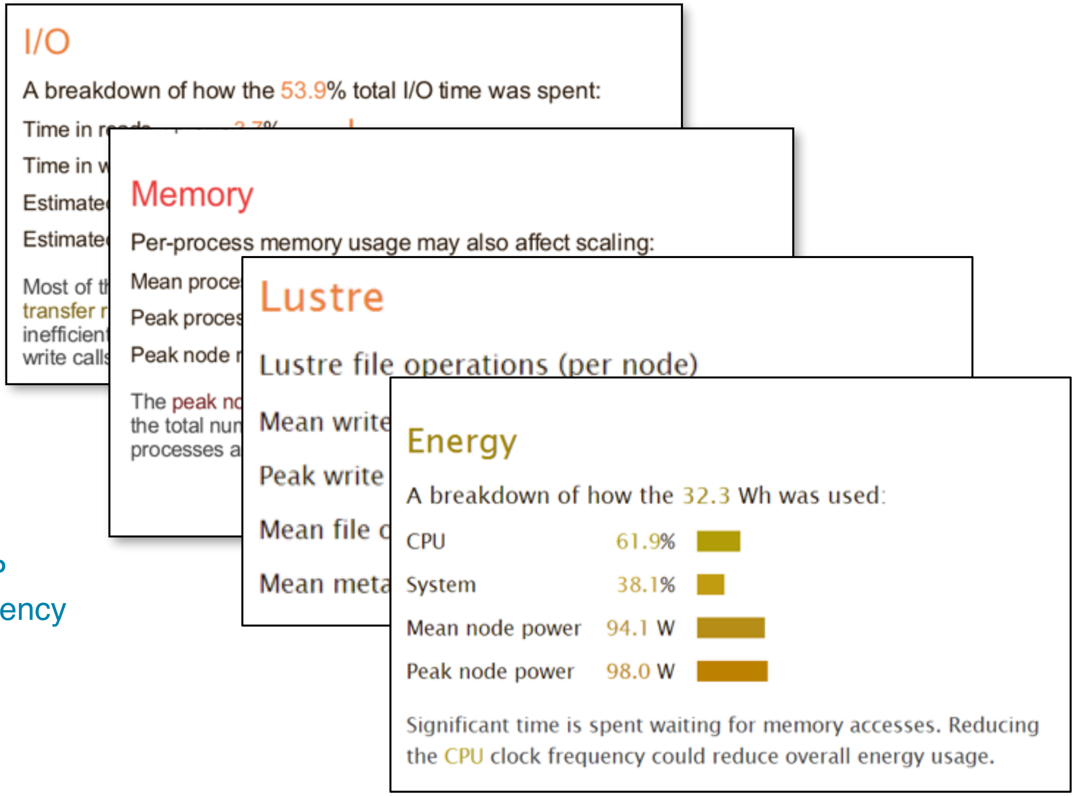
OpenMP

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.



I/O

A breakdown of how the 53.9% total I/O time was spent:

Time in r
Time in w
Estimated
Estimated

Most of th
transfer r
inefficient
write calls

Memory

Per-process memory usage may also affect scaling:

Mean proce
Peak proces
Peak node r

The peak no
the total nun
processes a

Lustre

Lustre file operations (per node)

Mean write
Peak write
Mean file c
Mean meta

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

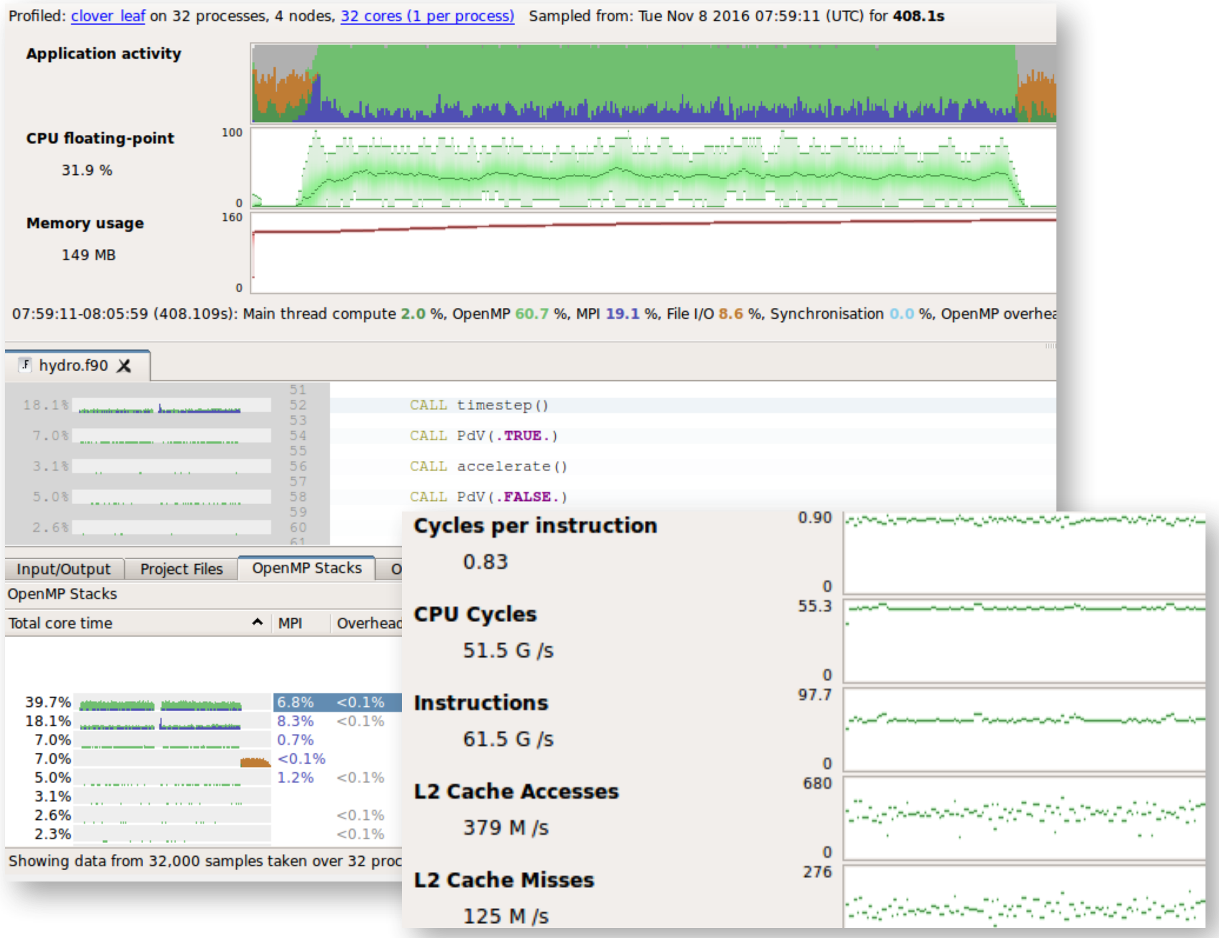
- Built on same framework as DDT
- Parallel support for MPI, OpenMP, CUDA
- Designed for C/C++/Fortran

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



Profiled: clover_leaf on 32 processes, 4 nodes, 32 cores (1 per process) Sampled from: Tue Nov 8 2016 07:59:11 (UTC) for **408.1s**

Application activity

CPU floating-point 31.9 %

Memory usage 149 MB

07:59:11-08:05:59 (408.109s): Main thread compute 2.0 %, OpenMP 60.7 %, MPI 19.1 %, File I/O 8.6 %, Synchronisation 0.0 %, OpenMP overhea

```
      CALL timestep()

      CALL PdV(.TRUE.)

      CALL accelerate()

      CALL PdV(.FALSE.)
```

OpenMP Stacks

Total core time	MPI	Overhead
39.7%	6.8%	<0.1%
18.1%	8.3%	<0.1%
7.0%	0.7%	
7.0%	<0.1%	
5.0%	1.2%	<0.1%
3.1%		
2.6%		<0.1%
2.3%		<0.1%

Showing data from 32,000 samples taken over 32 proc

Cycles per instruction 0.83

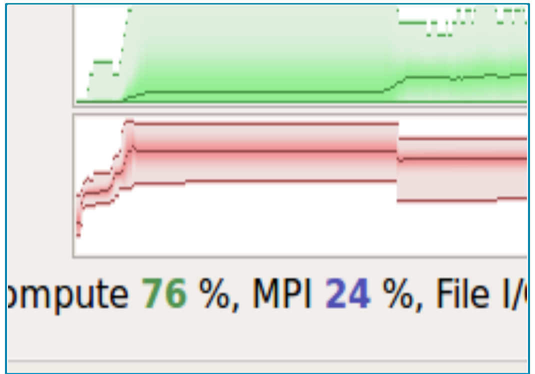
CPU Cycles 51.5 G /s

Instructions 61.5 G /s

L2 Cache Accesses 379 M /s

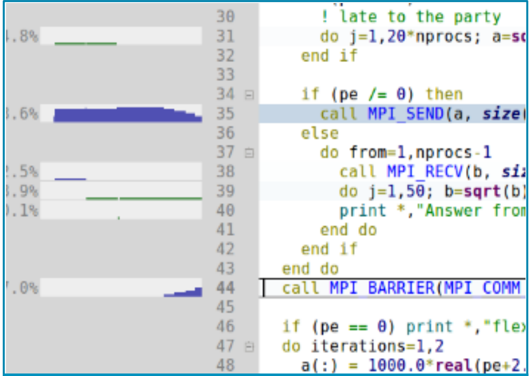
L2 Cache Misses 125 M /s

MAP Highlights



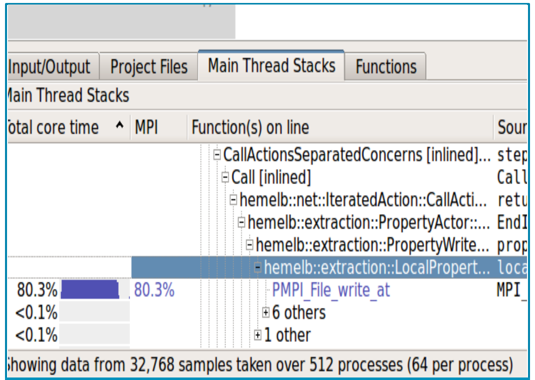
ompute 76 %, MPI 24 %, File I/(

Find the peak memory use



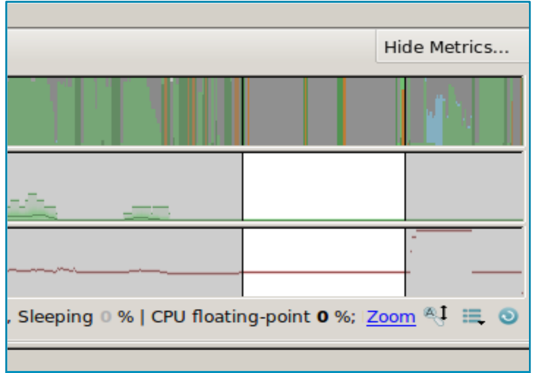
```
      ! late to the party
      do j=1,20*nprocs; a=sc
    end if

    if (pe /= 0) then
      call MPI_SEND(a, size(
    else
      do from=1,nprocs-1
        call MPI_RECV(b, siz
        do j=1,50; b=sqrt(b)
        print *,"Answer from
      end do
    end if
  end do
  call MPI_BARRIER(MPI_COMM

  if (pe == 0) print *,"flex
  do iterations=1,2
    a(:) = 1000.0*real(pe+2.
```

Fix an MPI imbalance



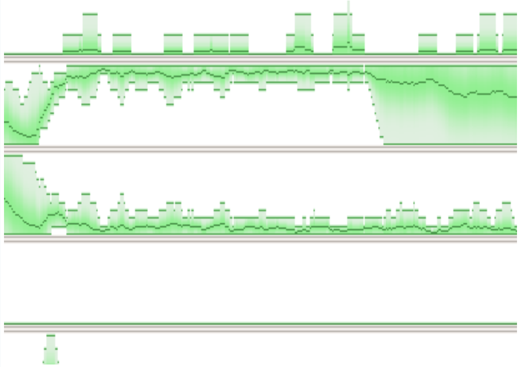
```
Main Thread Stacks
Total core time ^ MPI     Function(s) on line                        Sour
                          ⊟CallActionsSeparatedConcerns [inlined]... step
                           ⊟Call [inlined]                            Call
                            ⊟hemelb::net::IteratedAction::CallActi... retu
                             ⊟hemelb::extraction::PropertyActor::...  EndI
                              ⊟hemelb::extraction::PropertyWrite...   prop
                               ⊟hemelb::extraction::LocalPropert...   loca
80.3%             80.3%          PMPI_File_write_at                   MPI_
<0.1%                           ⊞6 others
<0.1%                          ⊞1 other
Showing data from 32,768 samples taken over 512 processes (64 per process)
```

Remove I?O bottleneck

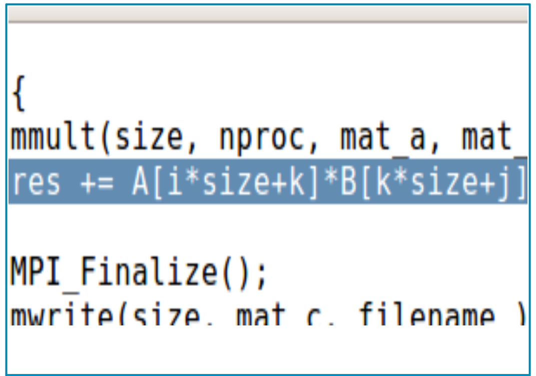


Sleeping 0 % | CPU floating-point 0 %; Zoom

Make sure OpenMP regions make sense



Improve memory access



```
{
mmult(size, nproc, mat_a, mat
res += A[i*size+k]*B[k*size+j]

MPI_Finalize();
mwrite(size, mat_c, filename )
```

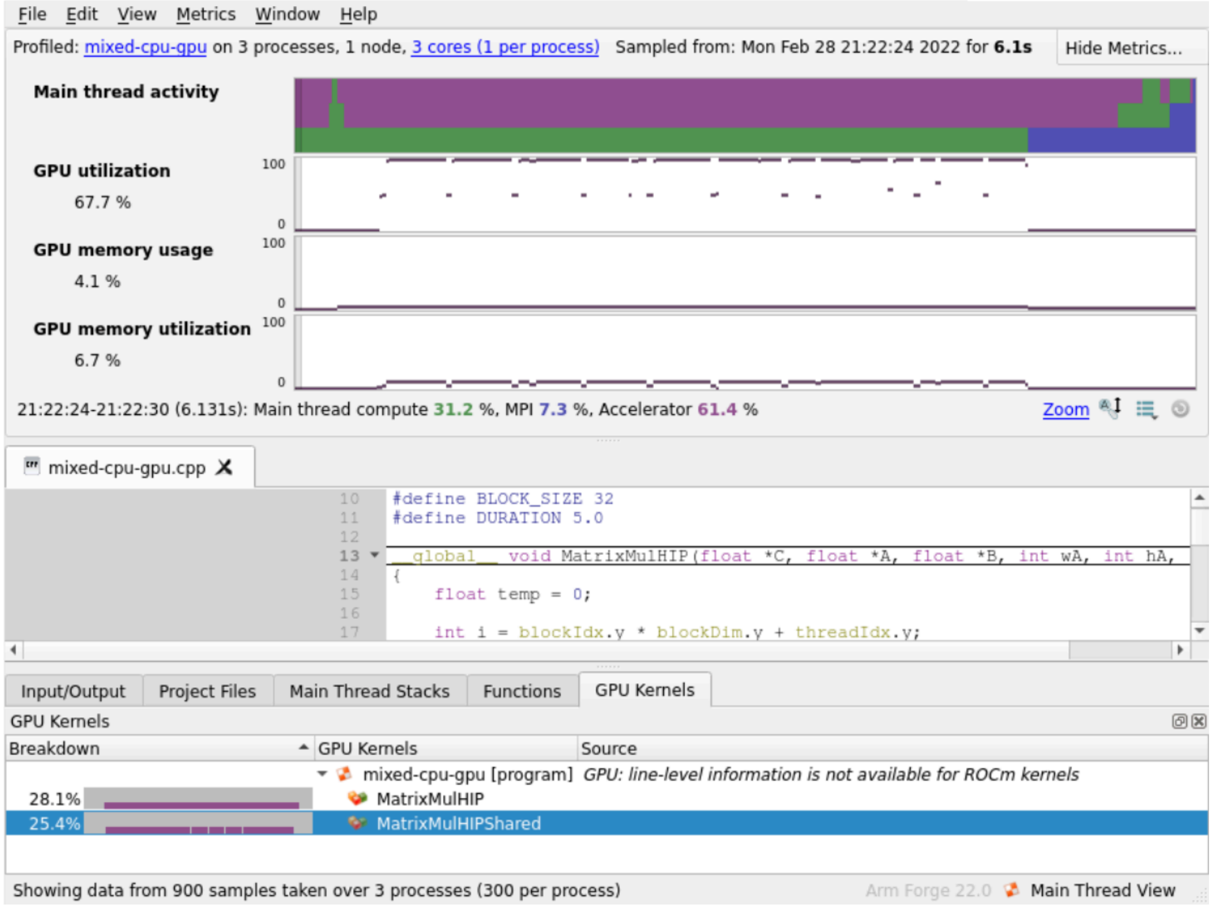
Restructure for vectorization

ARGONNE ATPESC2024 EXTREME-SCALE COMPUTING

extremecomputingtraining.anl.gov

Argonne NATIONAL LABORATORY

GPU profiling



Profiled: mixed-cpu-gpu on 3 processes, 1 node, 3 cores (1 per process) Sampled from: Mon Feb 28 21:22:24 2022 for **6.1s**

21:22:24-21:22:30 (6.131s): Main thread compute 31.2 %, MPI 7.3 %, Accelerator 61.4 %

```
#define BLOCK_SIZE 32
#define DURATION 5.0

__global__ void MatrixMulHIP(float *C, float *A, float *B, int wA, int hA,
{
    float temp = 0;

    int i = blockIdx.y * blockDim.y + threadIdx.y;
```

GPU Kernels

Breakdown	GPU Kernels	Source
	mixed-cpu-gpu [program]	*GPU: line-level information is not available for ROCm kernels*
28.1%	MatrixMulHIP	
25.4%	MatrixMulHIPShared	

Showing data from 900 samples taken over 3 processes (300 per process) Arm Forge 22.0 Main Thread View

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

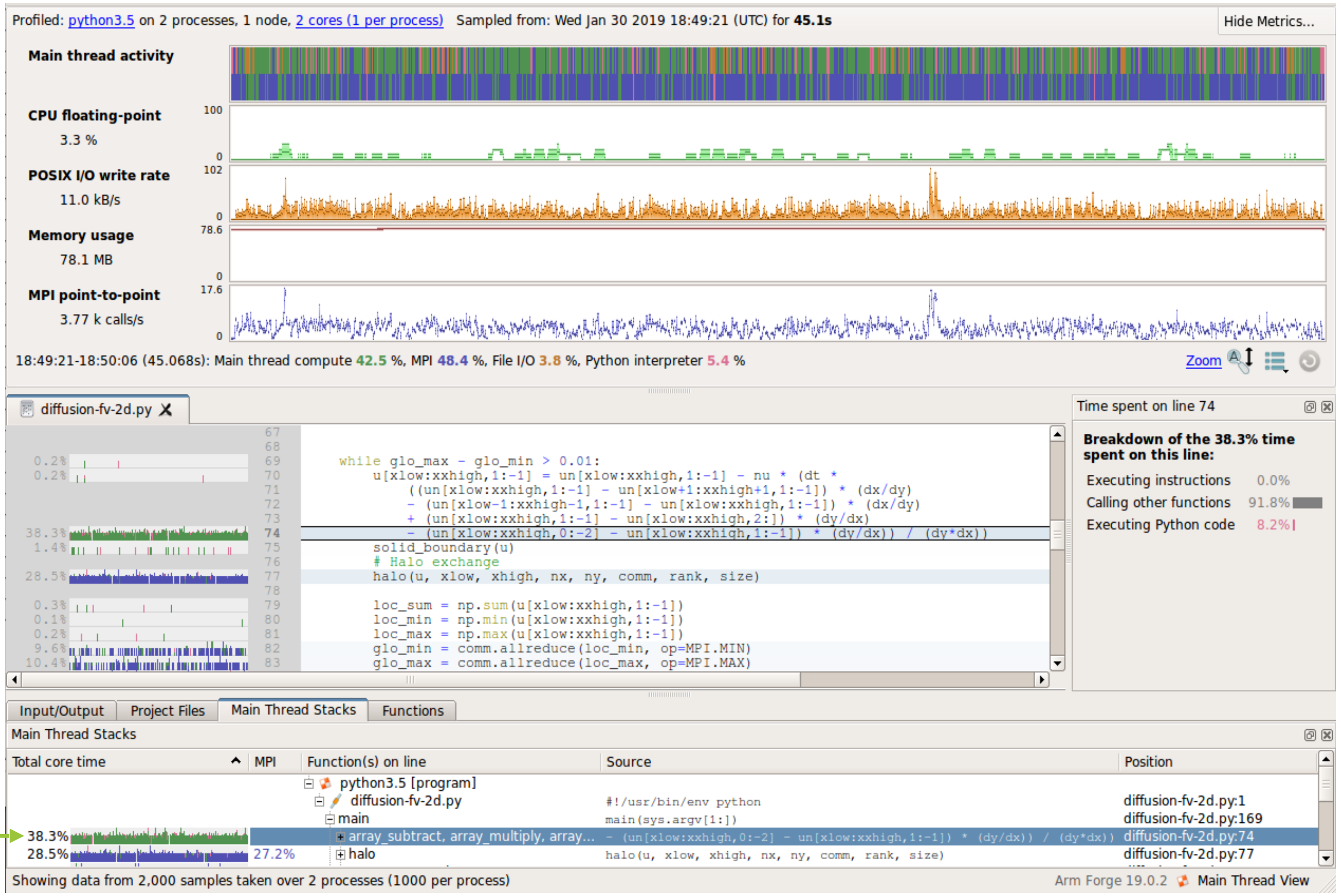
- Call stacks
- Time in interpreter

Works with MPI4PY

- Usual MAP metrics

Source code view

- Mixed language support



Profiled: python3.5 on 2 processes, 1 node, 2 cores (1 per process) Sampled from: Wed Jan 30 2019 18:49:21 (UTC) for 45.1s

Hide Metrics...

Main thread activity

CPU floating-point 3.3 %

POSIX I/O write rate 11.0 kB/s

Memory usage 78.1 MB

MPI point-to-point 3.77 k calls/s

18:49:21-18:50:06 (45.068s): Main thread compute 42.5 %, MPI 48.4 %, File I/O 3.8 %, Python interpreter 5.4 %

diffusion-fv-2d.py

```
67
68
69  0.2%     while glo_max - glo_min > 0.01:
70  0.2%         u[xlow:xxhigh,1:-1] = un[xlow:xxhigh,1:-1] - nu * (dt *
71                   ((un[xlow:xxhigh,1:-1] - un[xlow+1:xxhigh+1,1:-1]) * (dx/dy)
72                   - (un[xlow-1:xxhigh-1,1:-1] - un[xlow:xxhigh,1:-1]) * (dx/dy)
73                   + (un[xlow:xxhigh,1:-1] - un[xlow:xxhigh,2:]) * (dy/dx)
74  38.3%            - (un[xlow:xxhigh,0:-2] - un[xlow:xxhigh,1:-1]) * (dy/dx)) / (dy*dx))
75  1.4%         solid_boundary(u)
76               # Halo exchange
77  28.5%        halo(u, xlow, xhigh, nx, ny, comm, rank, size)
78
79  0.3%         loc_sum = np.sum(u[xlow:xxhigh,1:-1])
80  0.1%         loc_min = np.min(u[xlow:xxhigh,1:-1])
81  0.2%         loc_max = np.max(u[xlow:xxhigh,1:-1])
82  9.6%         glo_min = comm.allreduce(loc_min, op=MPI.MIN)
83  10.4%        glo_max = comm.allreduce(loc_max, op=MPI.MAX)
```

Time spent on line 74

Breakdown of the 38.3% time spent on this line:

Executing instructions	0.0%
Calling other functions	91.8%
Executing Python code	8.2%

Main Thread Stacks

Total core time	MPI	Function(s) on line	Source	Position
		python3.5 [program]		
		diffusion-fv-2d.py	#!/usr/bin/env python	diffusion-fv-2d.py:1
		main	main(sys.argv[1:])	diffusion-fv-2d.py:169
38.3%		array_subtract, array_multiply, array...	- (un[xlow:xxhigh,0:-2] - un[xlow:xxhigh,1:-1]) * (dy/dx)) / (dy*dx))	diffusion-fv-2d.py:74
28.5%	27.2%	halo	halo(u, xlow, xhigh, nx, ny, comm, rank, size)	diffusion-fv-2d.py:77

Showing data from 2,000 samples taken over 2 processes (1000 per process)

Arm Forge 19.0.2 Main Thread View

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

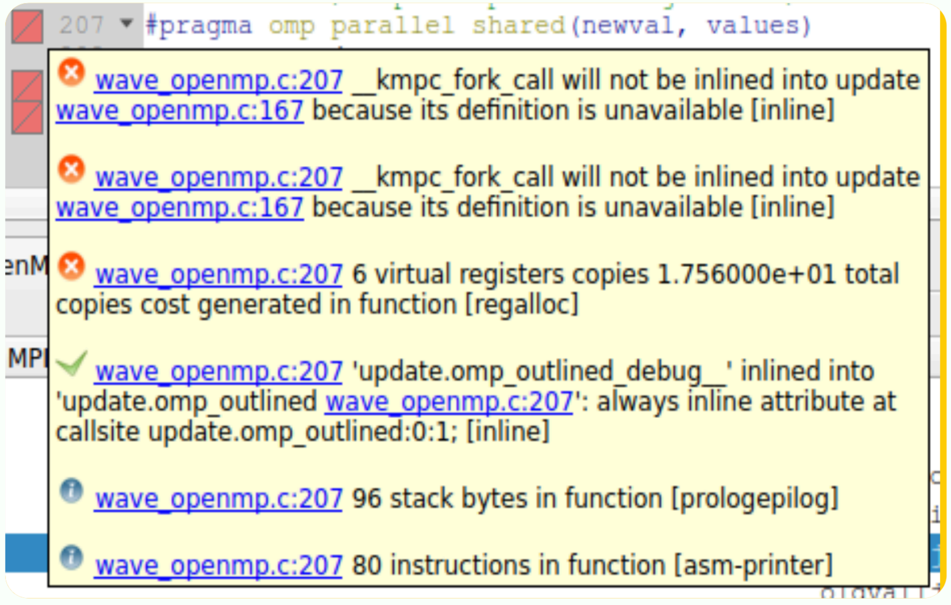
- Remarks are extracted from the compiler optimisation report
- Compiler remarks are displayed as annotations next to your source code

Colour coded

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

Compiler Remarks menu.

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



```
#pragma omp parallel shared(newval, values)
```

✗ wave_openmp.c:207 __kmpc_fork_call will not be inlined into update wave_openmp.c:167 because its definition is unavailable [inline]

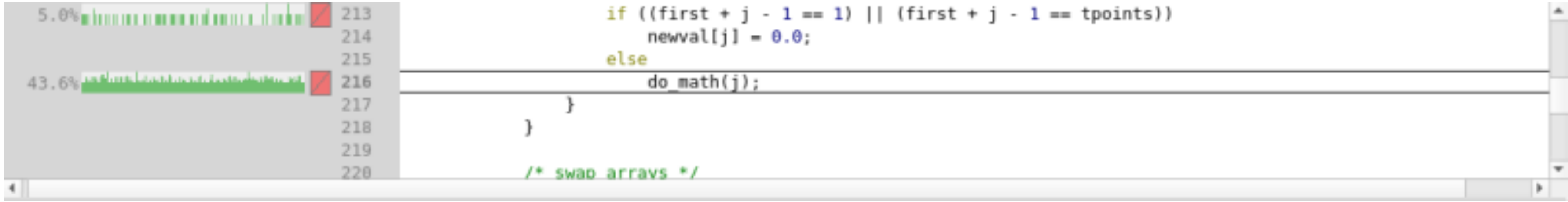
✗ wave_openmp.c:207 __kmpc_fork_call will not be inlined into update wave_openmp.c:167 because its definition is unavailable [inline]

✗ wave_openmp.c:207 6 virtual registers copies 1.756000e+01 total copies cost generated in function [regalloc]

✓ wave_openmp.c:207 'update.omp_outlined_debug__' inlined into 'update.omp_outlined wave_openmp.c:207': always inline attribute at callsite update.omp_outlined:0:1; [inline]

ⓘ wave_openmp.c:207 96 stack bytes in function [prologepilog]

ⓘ wave_openmp.c:207 80 instructions in function [asm-printer]



```
            if ((first + j - 1 == 1) || (first + j - 1 == tpoints))
               newval[j] = 0.0;
            else
               do_math(j);
         }
      }

      /* swap arrays */
```

Thank you

Linaro Website

www.linaro.org

Linaro Forge Website

www.linaroforge.com

Contacts

Sales: sales@forge.linaro.com

Support: support@forge.linaro.com