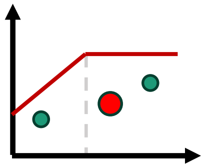
ARCHER2/Cirrus Lightning Talks

on the theme of code performance

SPINDLE

MLPerf



Roofline Models

OpenMP

Offload to GPU

Michael Bareford, EPCC, The University of Edinburgh

m.bareford@epcc.ed.ac.uk

Reusing this material

Spindle

Improving library load performance on ARCHER2 and Cirrus

https://computing.llnl.gov/projects/spindle

https://github.com/hpc/Spindle

Motivation

Spindle is a tool for improving the library-loading performance of dynamically linked HPC applications.

- Library loads scale as $N_p \times N_{lib} \times N_{path}$
- Provides a mechanism for **scalable loading of shared libraries, executables and python** files from a shared file system at scale without turning the file system into a bottleneck.
- Is a **pure user-space** approach. Users do not need to configure new file systems, load modules into their OS kernels or build special system components.
- Operates on stock binaries. **No application modification or special build flags** are required.

https://computing.llnl.gov/projects/spindle

Installation

```
./configure CC=gcc CXX=g++ \
    --enable-sec-none \
    --with-slurm-launch \
    --with-localstorage=/dev/shm \
    --prefix=${SPINDLE_ROOT}/${SPINDLE_VERSION}

make -j 8
make -j 8 install
```

Help on `configure` options

https://github.com/hpc/Spindle/blob/devel/INSTALL

Installation

```
./configure CC=gcc CXX=g++ \
     --enable-sec-none \
     --with-slurm-launch \
     --with-localstorage=/dev/shm \
     --prefix=${SPINDLE_ROOT}/${SPIND

make -j 8
make -j 8 install
```

Versions of Slurm later than 20.11 interfere/prevent Spindle from launching daemons during job startup.

ARCHER2 (Slurm 22.05.8), Cirrus (Slurm 21.08.8-2)

Use `--with-slurm-launch` (instead of `--with-rm=slurm`) and `srun` must be run with `-overlap` option.

Help on `configure` options

https://github.com/hpc/Spindle/blob/devel/INSTALL

Installation

```
./configure CC=gcc CXX=g++ \
    --enable-sec-none \
    --with-slurm-launch \
    --with-localstorage=/dev/shm \
    --prefix=${SPINDLE_ROOT}/${SPIND

make -j 8
make -j 8 install
```

Versions of Slurm later than 20.11 interfere/prevent Spindle from launching daemons during job startup.

ARCHER2 (Slurm 22.05.8), Cirrus (Slurm 21.08.8-2)

Use `--with-slurm-launch` (instead of `--with-rm=slurm`) and `srun` must be run with `-overlap` option.

On Cirrus must use `--with-localstorage=/tmp` because `/dev/shm` does not have execute privilege.

Help on `configure` options

https://github.com/hpc/Spindle/blob/devel/INSTALL

Installation

```
./configure CC=gcc CXX=g++ \
    --enable-sec-none \
    --with-slurm-launch \
    --with-localstorage=/dev/shm \
    --prefix=${SPINDLE_ROOT}/${SPIND

make -j 8
make -j 8 install
```

Versions of Slurm later than 20.11 interfere/prevent Spindle from launching daemons during job startup.

ARCHER2 (Slurm 22.05.8), Cirrus (Slurm 21.08.8-2)

Use `--with-slurm-launch` (instead of `--with-rm=slurm`) and `srun` must be run with `-overlap` option.

On Cirrus must use `--with-localstorage=/tmp` because `/dev/shm` does not have execute privilege.

Help on `configure` options

https://github.com/hpc/Spindle/blob/devel/INSTALL

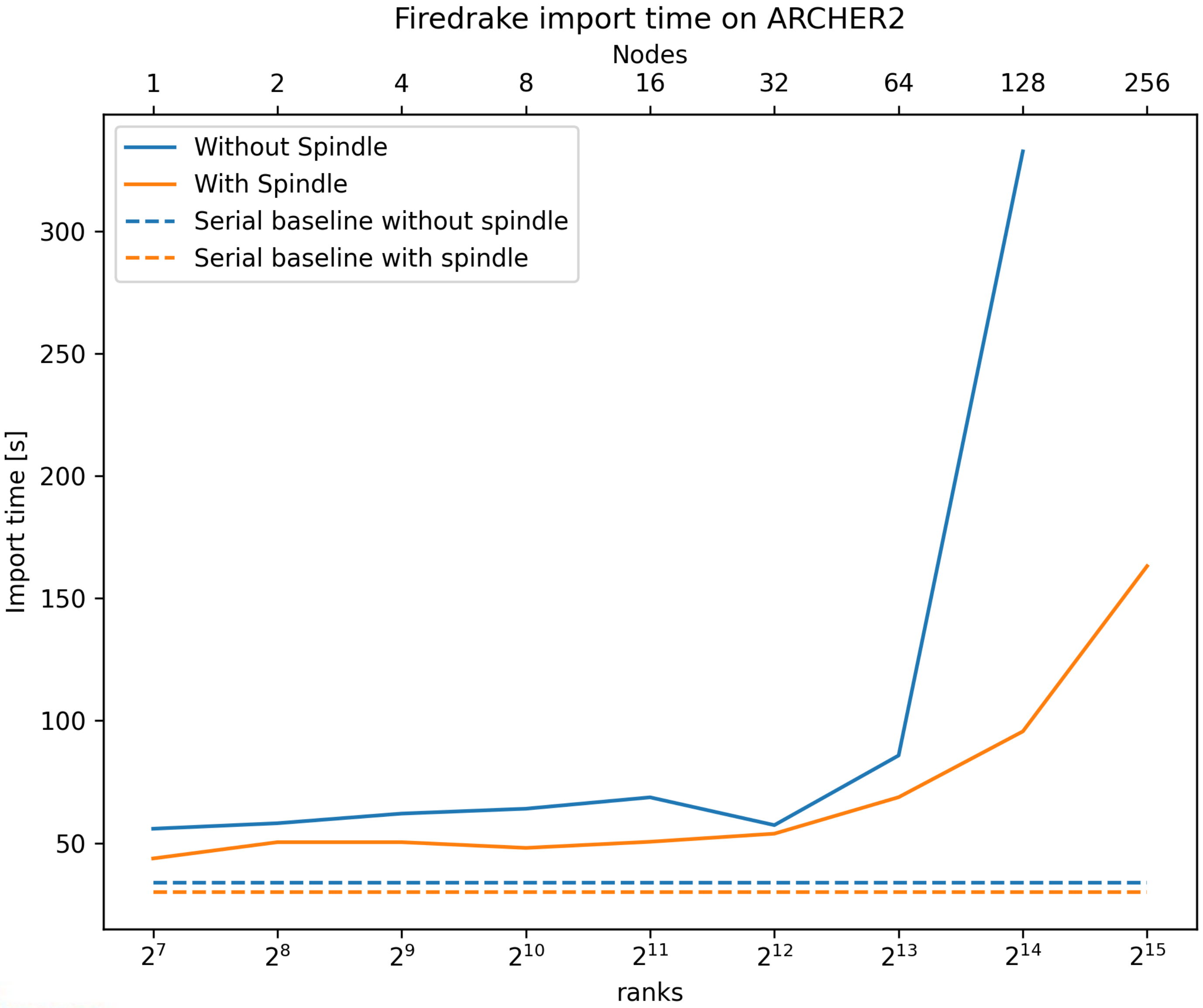
Instructions for ARCHER2 and Cirrus

https://github.com/hpc-uk/build-instructions/tree/main/utils/spindle

Running with Spindle

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --tasks-per-node=128
...

module -q load cray-python
module -q load spindle/0.13


export SRUN_CPUS_PER_TASK=${SLURM_CPUS_PER_TASK}


spindle --slurm --python-prefix=/opt/cray/pe/python/${CRAY_PYTHON_LEVEL} \
    srun --overlap --distribution=block:block --hint=nomultithread \
        python ${SLURM_SUBMIT_DIR}/test.py
```

Performance



Firedrake

- An automated system for the solution of partial differential equations using the finite element method (FEM)
- A Python code that generates C kernels for computationally demanding tasks.
- Code does many Python imports which impacts performance at scale.
- https://www.firedrakeproject.org/index.html

https://www.imperial.ac.uk/people/j.betteridge

Testing Spindle with Pynamic

- You can test Spindle yourself using the Pynamic tool.
  - Pynamic is a benchmark generator designed to test a system's ability to handle the dynamic linking and loading requirements of Python-based scientific applications.
  - https://github.com/LLNL/pynamic

Testing Spindle with Pynamic

- You can test Spindle yourself using the Pynamic tool.
  - Pynamic is a benchmark generator designed to test a system's ability to handle the dynamic linking and loading requirements of Python-based scientific applications.
  - https://github.com/LLNL/pynamic
- You first build Pynamic for a particular library and/or Python module load.
  - The required number of dummy libraries are built and installed locally. You then run the custom Pynamic benchmark with and without Spindle.
  - https://github.com/hpc-uk/build-instructions/tree/main/utils/pynamic

MLPerf HPC benchmarks

A suite of ML benchmarks suitable for HPC

- **CosmoFlow**: cosmological parameters from simulation output
- DeepCam: extreme weather pattern detection and classification
- OpenCatalyst: catalysis evaluation
- OpenFold: protein structure prediction

Demystifying the MLPerf Training Benchmark Suite
https://ieeexplore.ieee.org/document/9238612

Background

MLPerf is a product of the **MLCommons** consortium.

The mission of MLCommons™ is to make machine learning better for everyone.

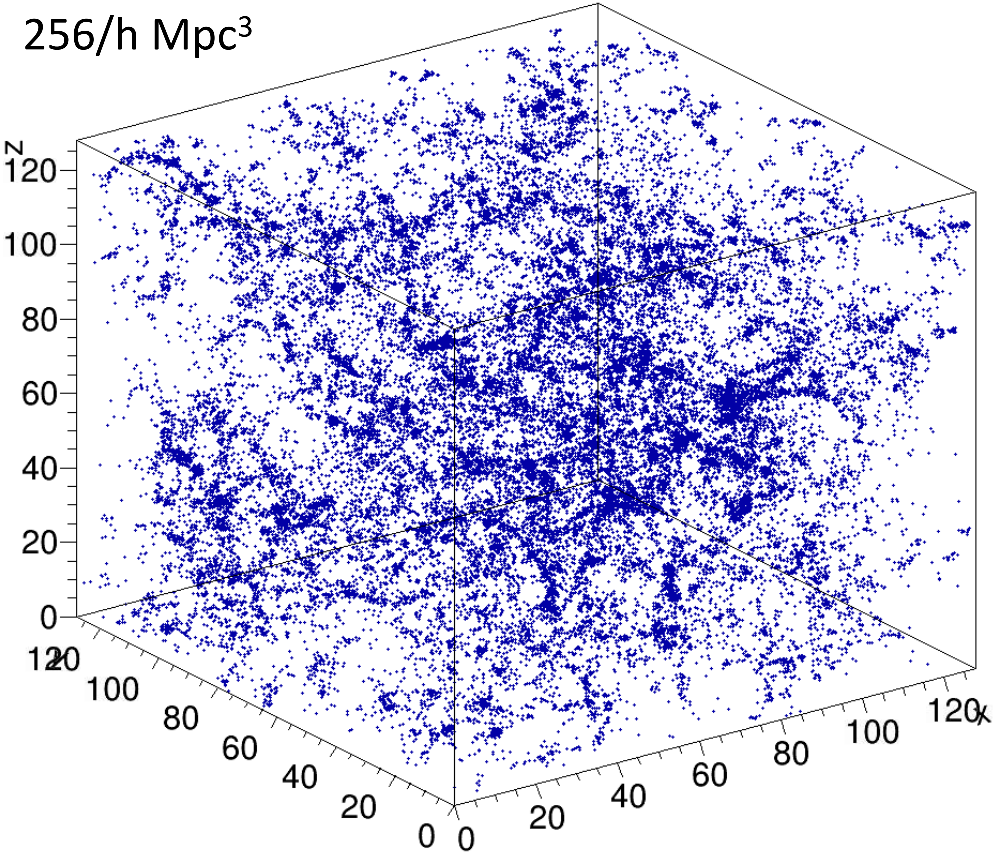
https://mlcommons.org/en/

https://github.com/mlcommons/hpc

CosmoFlow Benchmark

Dark matter simulation

256/h Mpc^3



Mathuriya et al. 2018

CosmoFlow - Using Deep Learning to Learn the Universe at Scale
Mathuriya et al. 2018

https://dl.acm.org/doi/10.1109/SC.2018.00068

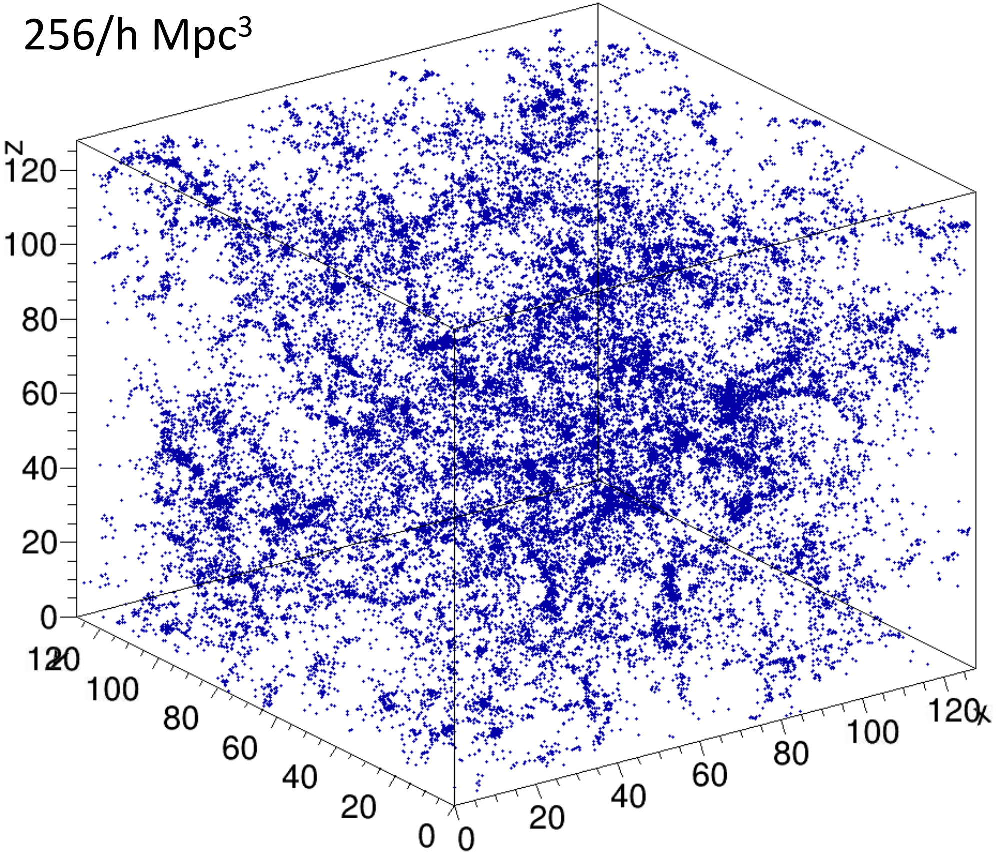
Train a DNN to predict three cosmological parameters.

- Ω_M , proportion of matter in the universe
- σ_s , amplitude of mass fluctuations
- n_s , scalar spectral index (spatial curvature)

CosmoFlow Benchmark

Dark matter simulation

256/h Mpc^3



Mathuriya et al. 2018

CosmoFlow - Using Deep Learning to Learn the Universe at Scale
Mathuriya et al. 2018

https://dl.acm.org/doi/10.1109/SC.2018.00068

Train a DNN to predict three cosmological parameters.

- Ω_M , proportion of matter in the universe
- σ_8 , amplitude of mass fluctuations
- n_s , scalar spectral index (spatial curvature)

First, run many cosmological simulations for a variety of parameter values constrained by observations of the CMB (Planck mission).

Slice simulation output into 128^3 voxel sub-volumes and use as input to DNN.

- MLPerf HPC v1.0 preliminary dataset, 1.7 TB
  - 524,288 training
  - 65,536 validation
  - 32,769 testing

Running CosmoFlow on ARCHER2

```
#!/bin/bash

...
#SBATCH --nodes=32
#SBATCH --tasks-per-node=8
#SBATCH --cpus-per-task=16

module -q load tensorflow/2.12.0


...
export OMP_NUM_THREADS=16
export TF_ENABLE_ONEDNN_OPTS=1

srun --distribution=block:block --hint=nomultithread --cpu-freq=2250000 \
    python ./train.py --distributed --omp-num-threads ${OMP_NUM_THREADS} \
                      --inter-threads 0 --intra-threads 0
```

Running CosmoFlow on ARCHER2

```
#!/bin/bash

...
#SBATCH --nodes=32
#SBATCH --tasks-per-node=8
#SBATCH --cpus-per-task=16

module -q load tensorflow/2.12.0


...
export OMP_NUM_THREADS=16
export TF_ENABLE_ONEDNN_OPTS=1


srun --distribution=block:block --hint=nomultithread --cpu-freq=2250000 \
    python ./train.py --distributed --omp-num-threads ${OMP_NUM_THREADS} \
                      --inter-threads 0 --intra-threads 0
```

`TF_ENABLE_ONEDNN_OPTS`

- oneDNN is Intel’s oneAPI Deep Neural Network library
  - Within TensorFlow there are `#ifdef` guards that are activated when oneDNN is enabled.
  - Turns out the “oneDNN” code also gives good performance on AMD processors.
  - 12x speedup on ARCHER2

Running CosmoFlow on ARCHER2

```
#!/bin/bash

...
#SBATCH --nodes=32
#SBATCH --tasks-per-node=8
#SBATCH --cpus-per-task=16

module -q load tensorflow/2.12.0


...
export OMP_NUM_THREADS=16

export TF_ENABLE_ONEDNN_OPTS=1


srun --distribution=block:block --hint=nomultithread --cpu-freq=2250000 \
    python ./train.py --distributed --omp-num-threads ${OMP_NUM_THREADS} \
                      --inter-threads 0 --intra-threads 0
```

Exploit parallelism implied by TensorFlow DNN graph

- if a node in the TF graph can be parallelised, the number of threads assigned will be the value of `--intra-threads`
- if there are separate nodes in the TF graph that can be run concurrently, the available thread count is the value of `--inter-threads`
- optimum intra/inter thread counts vary from case to case
- tell TensorFlow to choose by setting zero values

CosmoFlow Performance on ARCHER2

- Set `TF_ENABLE_ONEDNN_OPTS=1`

- Specify 8 MPI tasks per node and 16 OpenMP threads per task
  - seven times faster compared to running 128 tasks per node

CosmoFlow Performance on ARCHER2

- Set TF_ENABLE_ONEDNN_OPTS=1

- Specify 8 MPI tasks per node and 16 OpenMP threads per task
  - seven times faster compared to running 128 tasks per node

- Running TensorFlow 2.12.0 on 32 nodes, 10 epochs takes approx. 3 hrs 20 mins
  - Similar runtimes are achieved if Horovod is swapped out for the Cray PE DL Plugin (craype-dl-plugin-py3/22.12.1)
    - Cray PE DL Plugin requires PrgEnv-gnu and must be used with TensorFlow 2.9.3
    - Need to replace "import **horovod**.tensorflow.keras as hvd" with "import **dl_comm**.tensorflow.keras as hvd"

- Running with "--cpu-freq=2250000" reduces runtime by 8% but increases energy use by 8%.

CosmoFlow Performance on ARCHER2

- Set `TF_ENABLE_ONEDNN_OPTS=1`

- Specify 8 MPI tasks per node and 16 OpenMP threads per task
  - seven times faster compared to running 128 tasks per node

- Running TensorFlow 2.12.0 on 32 nodes, 10 epochs takes approx. 3 hrs 20 mins
  - Similar runtimes are achieved if Horovod is swapped out for the Cray PE DL Plugin (`craype-dl-plugin-py3/22.12.1`)
    - Cray PE DL Plugin requires `PrgEnv-gnu` and must be used with TensorFlow 2.9.3
    - Need to replace "`import horovod.tensorflow.keras as hvd`" with "`import dl_comm.tensorflow.keras as hvd`"

- Running with "`--cpu-freq=2250000`" reduces runtime by 8% but increases energy use by 8%.

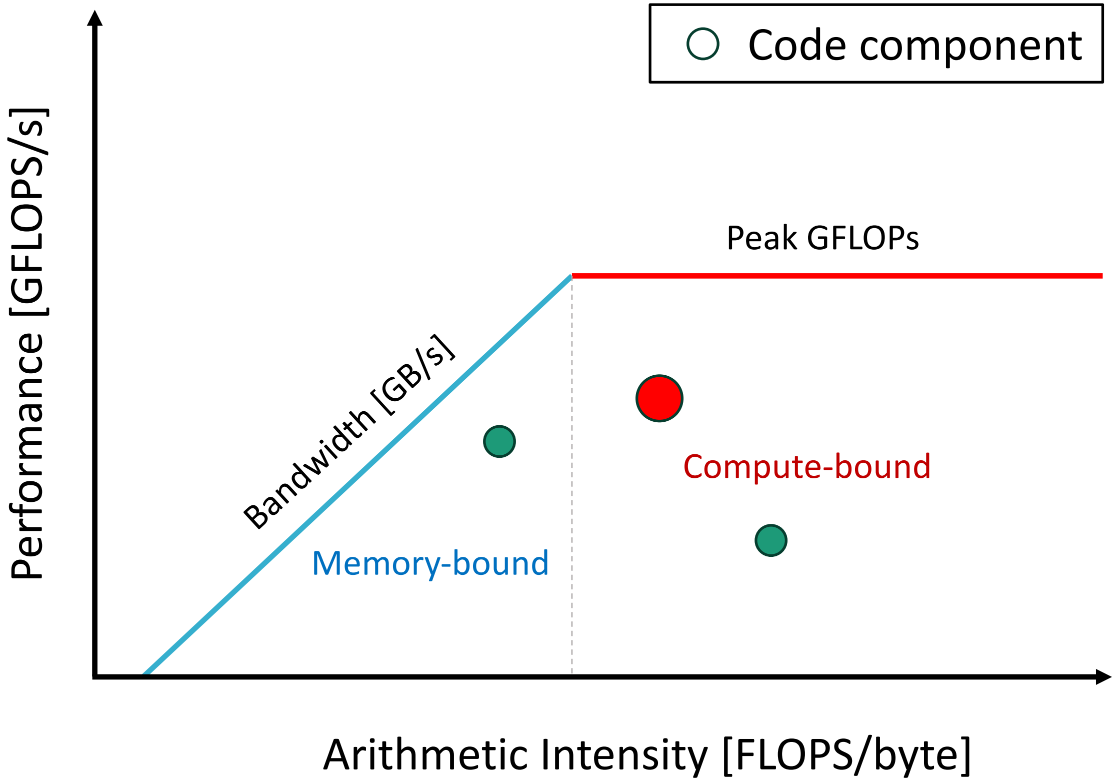
- Thanks to Eleanor Broadway for these findings.
  - Also benchmarked CosmoFlow on Cirrus (CPU/GPU) nodes.
  - Contact e.broadway@epcc.ed.ac.uk for details.

Roofline Models with Intel Advisor

Open-source post-processing visualization engine

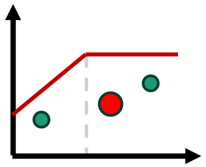
Roofline Plots

Axes are logarithmic

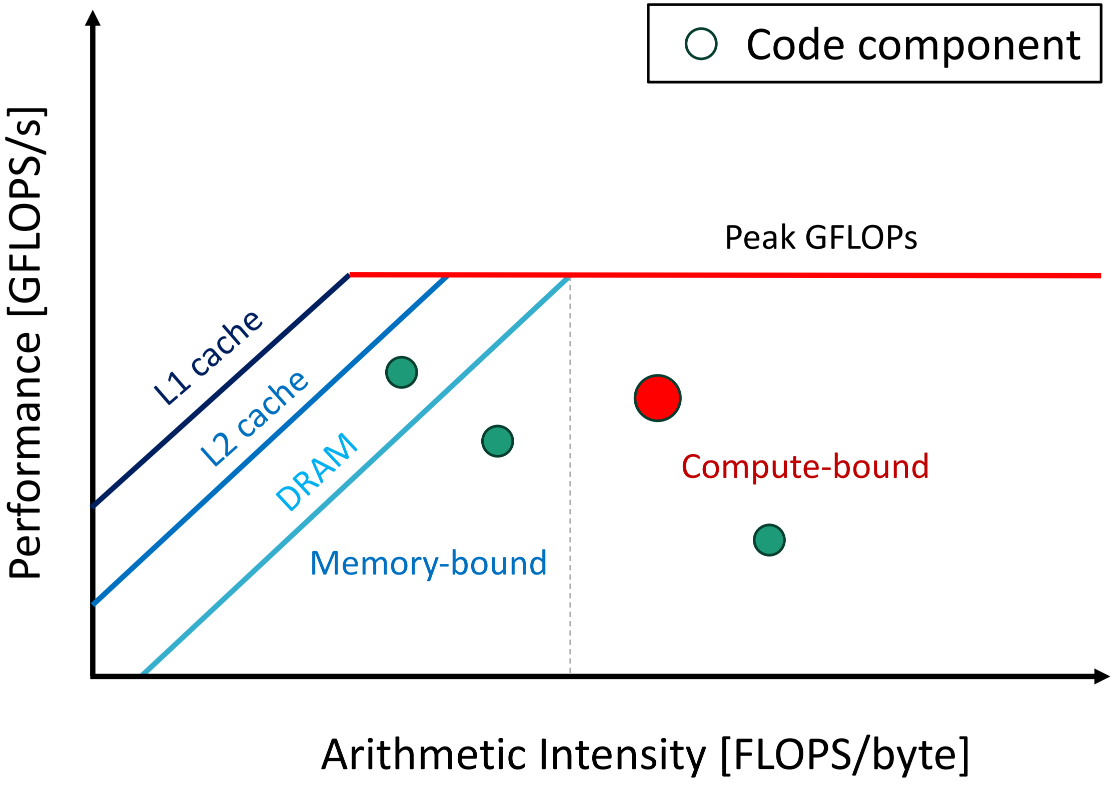


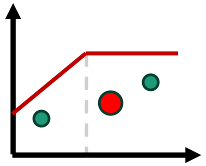
How is the performance of your code limited?

- Maximize compute performance
  - multithreading
  - vectorization
  - utilize FMA instructions
- Maximize memory bandwidth
  - use lower-level caches
  - NUMA-aware allocation
  - avoid non-contiguous memory access
- Maximize arithmetic intensity
  - minimize data movement
  - exploit cache reuse

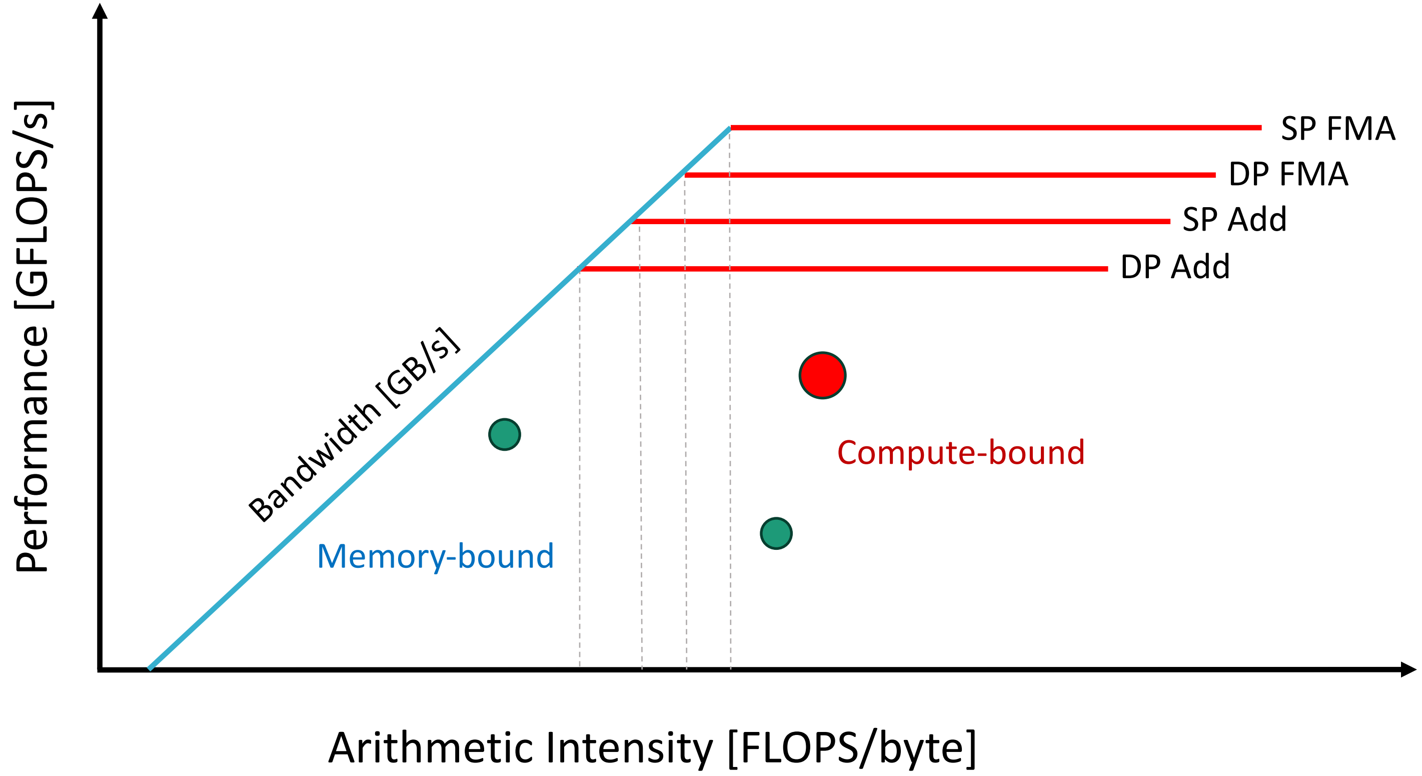


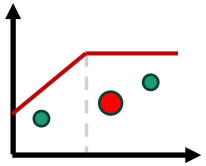
Cache-aware Roofline Plots





Multiple Compute Ceilings





Compiling with Intel on Cirrus

```
...

module -s load intel-20.4/compilers
module -s load intel-20.4/mpi

FC = mpiifort
FFLAGS = -O3 -g -fopenmp


...
```

Profiling with Intel Advisor on Cirrus

```
...
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=36
#SBATCH --cpus-per-task=1
...

module -s load intel-20.4/advisor

source ${ADVISOR_2020_DIR}/advixe-vars.sh

ADVISOR_DIR=${SLURM_SUBMIT_DIR}/advisor
mkdir -p ${ADVISOR_DIR}

...

srun ...
```

Profiling with Intel Advisor on Cirrus

```
...
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=36
```

```
...

srun --distribution=block:block --hint=nomultithread \
    advixe-cl --collect=survey --project-dir=${ADVISOR_DIR} -- \
        ${APP_EXE} ${APP_PARAMS}

srun --distribution=block:block --hint=nomultithread \
    advixe-cl --collect=tripcounts --flop --stacks --project-dir=${ADVISOR_DIR} -- \
        ${APP_EXE} ${APP_PARAMS}

...
```

```
srun ...
```

Profiling with Intel Advisor on Cirrus

Download Intel Advisor 2023.2.0 to local machine

- https://www.intel.com/content/www/us/en/developer/articles/tool/oneapi-standalone-components.html#advisor
- Windows, Linux, macOS

Intel Advisor User Guide

- https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2023-0/overview.html

Intel Advisor Cookbook

- https://www.intel.com/content/www/us/en/docs/advisor/cookbook/2023-0/overview.html
- Includes section on running Advisor on (HPE) Cray systems

Profiling with Intel Advisor on Cirrus

Download Intel Advisor 2023.2.0 to local machine

- https://www.intel.com/content/www/us/en/developer/articles/tool/oneapi-standalone-components.html#advisor
- Windows, Linux, macOS

Intel Advisor User Guide

- https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2023-0/overview.html

Intel Advisor Cookbook

- https://www.intel.com/content/www/us/en/docs/advisor/cookbook/2023-0/overview.html
- Includes section on running Advisor on (HPE) Cray systems

- Download Advisor results folder to local machine and launch the Intel Advisor UI.
- Select “`File | Open | Result...`”
- Expand the downloaded results folder and open the “`advisor.advixeproj`” file.

Profiling with Intel Advisor on Cirrus

Download Intel Advisor 2023.2.0 to local machine

- https://www.intel.com/content/www/us/en/developer/articles/tool/oneapi-standalone-components.html#advisor
- Windows, Linux, macOS

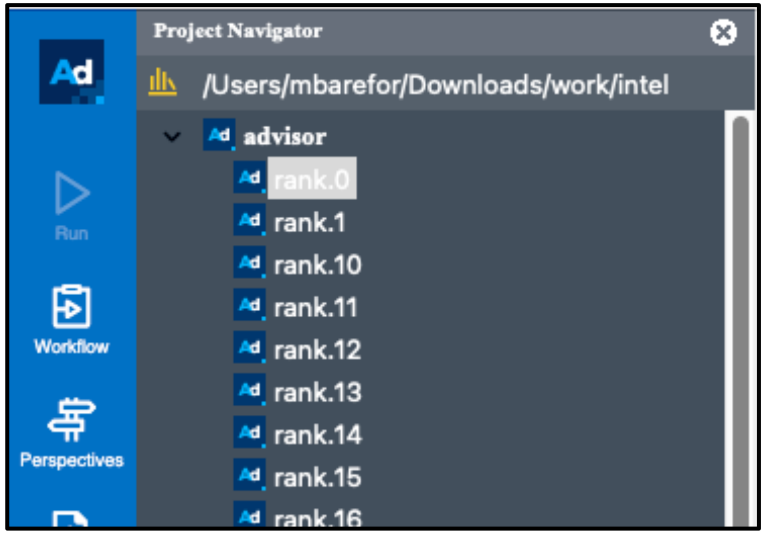
Intel Advisor User Guide

- https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2023-0/overview.html

Intel Advisor Cookbook

- https://www.intel.com/content/www/us/en/docs/advisor/cookbook/2023-0/overview.html
- Includes section on running Advisor on (HPE) Cray systems

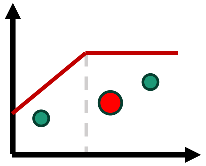
- Download Advisor results folder to local machine and launch the Intel Advisor UI.
- Select “`File | Open | Result...`”
- Expand the downloaded results folder and open the “`advisor.advixeproj`” file.



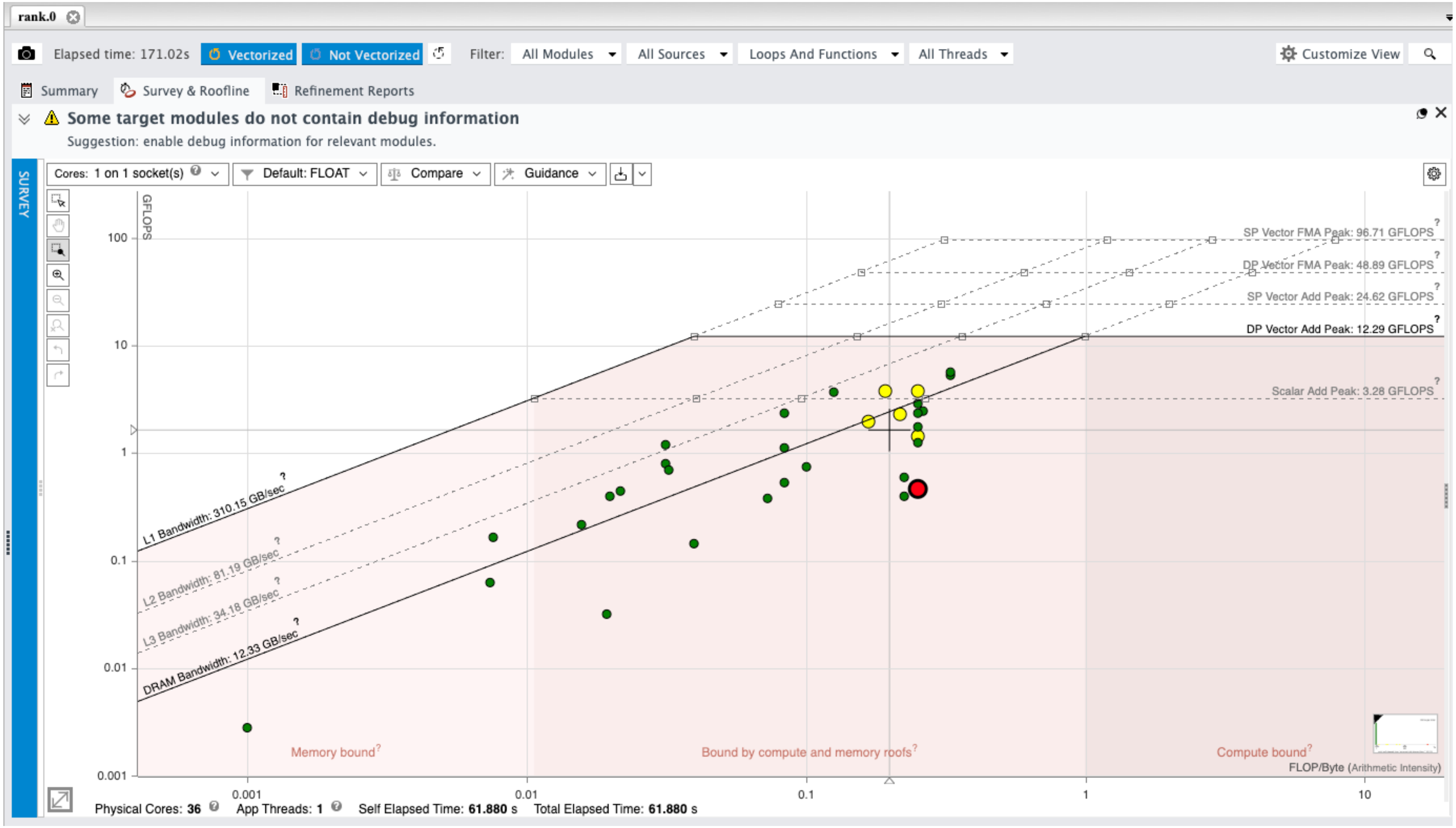
Project Navigator

/Users/mbarefor/Downloads/work/intel

- advisor
  - rank.0
  - rank.1
  - rank.10
  - rank.11
  - rank.12
  - rank.13
  - rank.14
  - rank.15
  - rank.16



Profiling with Intel Advisor on Cirrus



Elapsed time: 171.02s

Filter: All Modules · All Sources · Loops And Functions · All Threads

Some target modules do not contain debug information

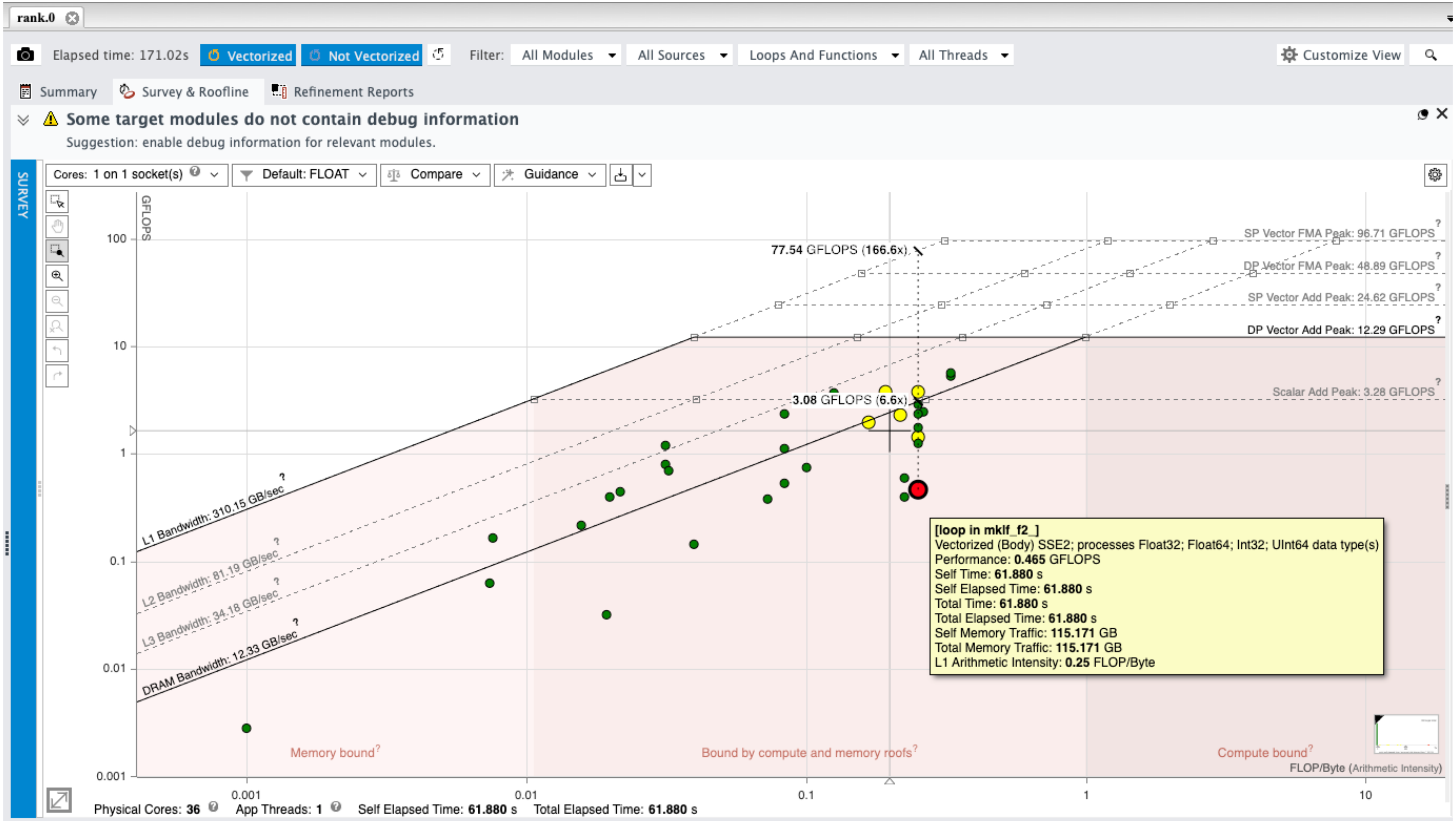
Suggestion: enable debug information for relevant modules.

Cores: 1 on 1 socket(s) · Default: FLOAT

Physical Cores: **36** App Threads: **1** Self Elapsed Time: **61.880** s Total Elapsed Time: **61.880** s

rank.0
Vectorized
Not Vectorized
Customize View
Summary
Survey & Roofline
Refinement Reports
Compare
Guidance
SURVEY
GFLOPS
100
10
1
0.1
0.01
0.001
0.001
0.01
0.1
1
10
FLOP/Byte (Arithmetic Intensity)
SP Vector FMA Peak: 96.71 GFLOPS
DP Vector FMA Peak: 48.89 GFLOPS
SP Vector Add Peak: 24.62 GFLOPS
DP Vector Add Peak: 12.29 GFLOPS
Scalar Add Peak: 3.28 GFLOPS
L1 Bandwidth: 310.15 GB/sec
L2 Bandwidth: 81.19 GB/sec
L3 Bandwidth: 34.18 GB/sec
DRAM Bandwidth: 12.33 GB/sec
Memory bound
Bound by compute and memory roofs
Compute bound

Profiling with Intel Advisor on Cirrus



rank.0
Vectorized
Not Vectorized
Summary
Survey & Roofline
Refinement Reports
Compare
Guidance
Customize View
SURVEY
GFLOPS
100
10
1
0.1
0.01
0.001
0.001
0.01
0.1
1
10
FLOP/Byte (Arithmetic Intensity)
SP Vector FMA Peak: 96.71 GFLOPS
DP Vector FMA Peak: 48.89 GFLOPS
SP Vector Add Peak: 24.62 GFLOPS
DP Vector Add Peak: 12.29 GFLOPS
Scalar Add Peak: 3.28 GFLOPS
L1 Bandwidth: 310.15 GB/sec
L2 Bandwidth: 81.19 GB/sec
L3 Bandwidth: 34.18 GB/sec
DRAM Bandwidth: 12.33 GB/sec
77.54 GFLOPS (166.6x)
3.08 GFLOPS (6.6x)
Memory bound
Bound by compute and memory roofs
Compute bound

Elapsed time: 171.02s

Filter: All Modules | All Sources | Loops And Functions | All Threads

Some target modules do not contain debug information

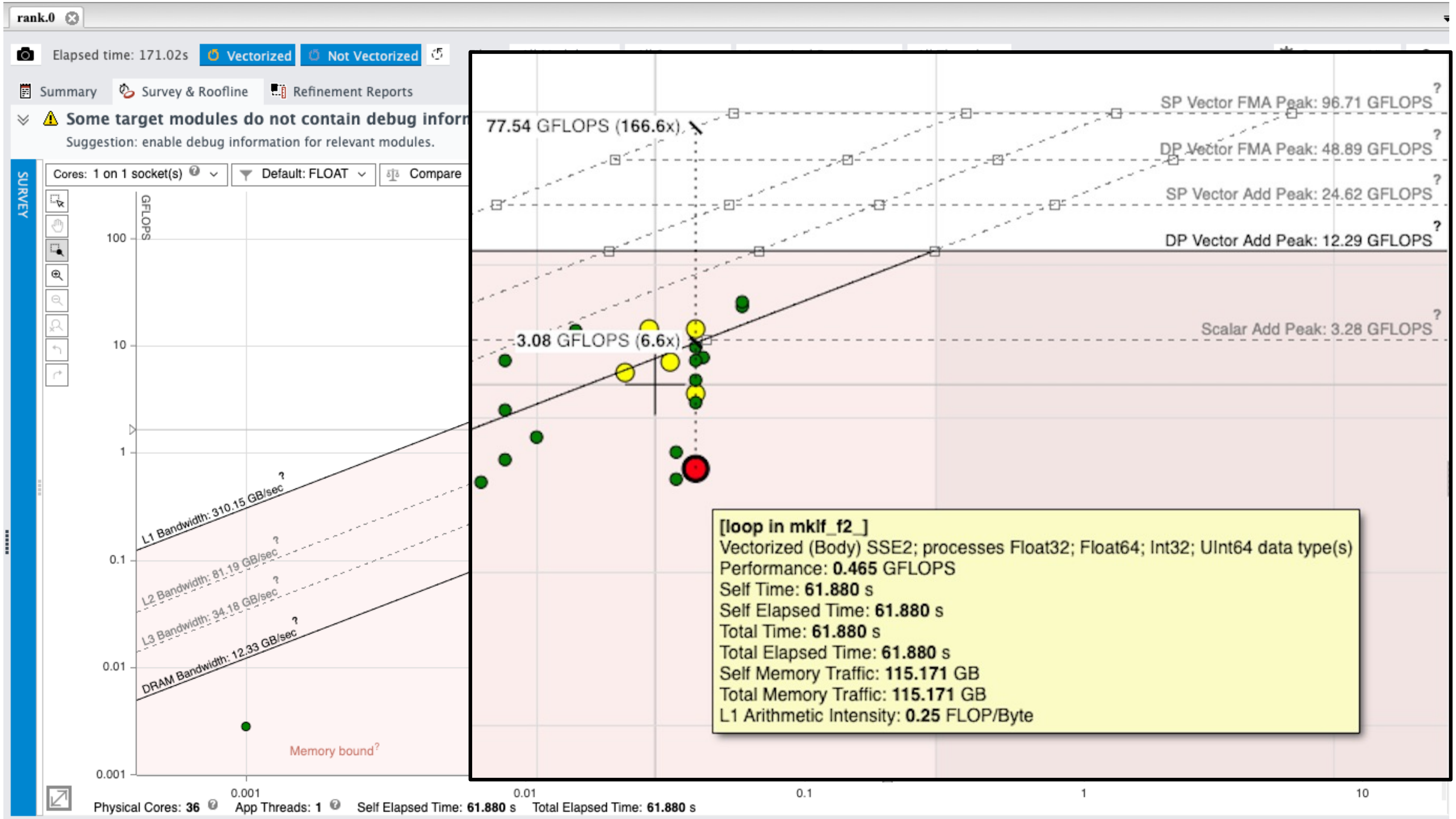
Suggestion: enable debug information for relevant modules.

Cores: 1 on 1 socket(s) | Default: FLOAT

[loop in mklf_f2_]
Vectorized (Body) SSE2; processes Float32; Float64; Int32; UInt64 data type(s)
Performance: **0.465** GFLOPS
Self Time: **61.880** s
Self Elapsed Time: **61.880** s
Total Time: **61.880** s
Total Elapsed Time: **61.880** s
Self Memory Traffic: **115.171** GB
Total Memory Traffic: **115.171** GB
L1 Arithmetic Intensity: **0.25** FLOP/Byte

Physical Cores: **36** App Threads: **1** Self Elapsed Time: **61.880** s Total Elapsed Time: **61.880** s

Profiling with Intel Advisor on Cirrus



rank.0

Elapsed time: 171.02s

⚠ Some target modules do not contain debug inforr

Suggestion: enable debug information for relevant modules.

Cores: 1 on 1 socket(s)

Default: FLOAT

[loop in mklf_f2_]
Vectorized (Body) SSE2; processes Float32; Float64; Int32; UInt64 data type(s)
Performance: **0.465** GFLOPS
Self Time: **61.880** s
Self Elapsed Time: **61.880** s
Total Time: **61.880** s
Total Elapsed Time: **61.880** s
Self Memory Traffic: **115.171** GB
Total Memory Traffic: **115.171** GB
L1 Arithmetic Intensity: **0.25** FLOP/Byte

Physical Cores: **36** App Threads: **1** Self Elapsed Time: **61.880 s** Total Elapsed Time: **61.880** s

Vectorized
Not Vectorized
Summary
Survey & Roofline
Refinement Reports
SURVEY
Compare
GFLOPS
100
10
1
0.1
0.01
0.001
0.001
0.01
0.1
1
10
L1 Bandwidth: 310.15 GB/sec
L2 Bandwidth: 81.19 GB/sec
L3 Bandwidth: 34.18 GB/sec
DRAM Bandwidth: 12.33 GB/sec
Memory bound
77.54 GFLOPS (166.6x)
3.08 GFLOPS (6.6x)
SP Vector FMA Peak: 96.71 GFLOPS
DP Vector FMA Peak: 48.89 GFLOPS
SP Vector Add Peak: 24.62 GFLOPS
DP Vector Add Peak: 12.29 GFLOPS
Scalar Add Peak: 3.28 GFLOPS

Compiling with Intel on ARCHER2

```
module use /work/z19/shared/adrianj/intel/oneapi/modulefiles
module -q load compiler/2023.0.0
module -q load mpi/2021.10.0


...


FC = mpiifort
FFLAGS = -O3 -fopenmp


...
```

Compiling with Intel on ARCHER2

```
module use /work/z19/shared/adrianj/intel/oneapi/modulefiles
module -q load compiler/2023.0.0
module -q load mpi/2021.10.0

...

FC = mpiifort
FFLAGS = -O3 -fopenmp

...
```

```
...
source ${CMPLR_ROOT}/env/vars.sh
source ${I_MPI_ROOT}/env/vars.sh -i_mpi_ofi_internal=0

LIBFABRIC_ROOT=/opt/cray/libfabric/1.12.1.2.2.0.0
export PATH=${LIBFABRIC_ROOT}/bin:${PATH}
export LD_LIBRARY_PATH=${LIBFABRIC_ROOT}/lib64:${LD_LIBRARY_PATH}
export FI_PROVIDER_PATH=${LIBFABRIC_ROOT}/lib64/libfabric.so

export I_MPI_OFI_LIBRARY_INTERNAL=0
export I_MPI_FABRICS=shm:ofi
```

Profiling with Intel Advisor on ARCHER2

```
...
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=1
...

module use /work/z19/shared/adrianj/intel/oneapi/modulefiles
module -q load advisor/2023.2.0

source ${ADVISOR_2023_DIR}/advisor-vars.sh

ADVISOR_DIR=${SLURM_SUBMIT_DIR}/advisor
mkdir -p ${ADVISOR_DIR}

...

srun ...
```

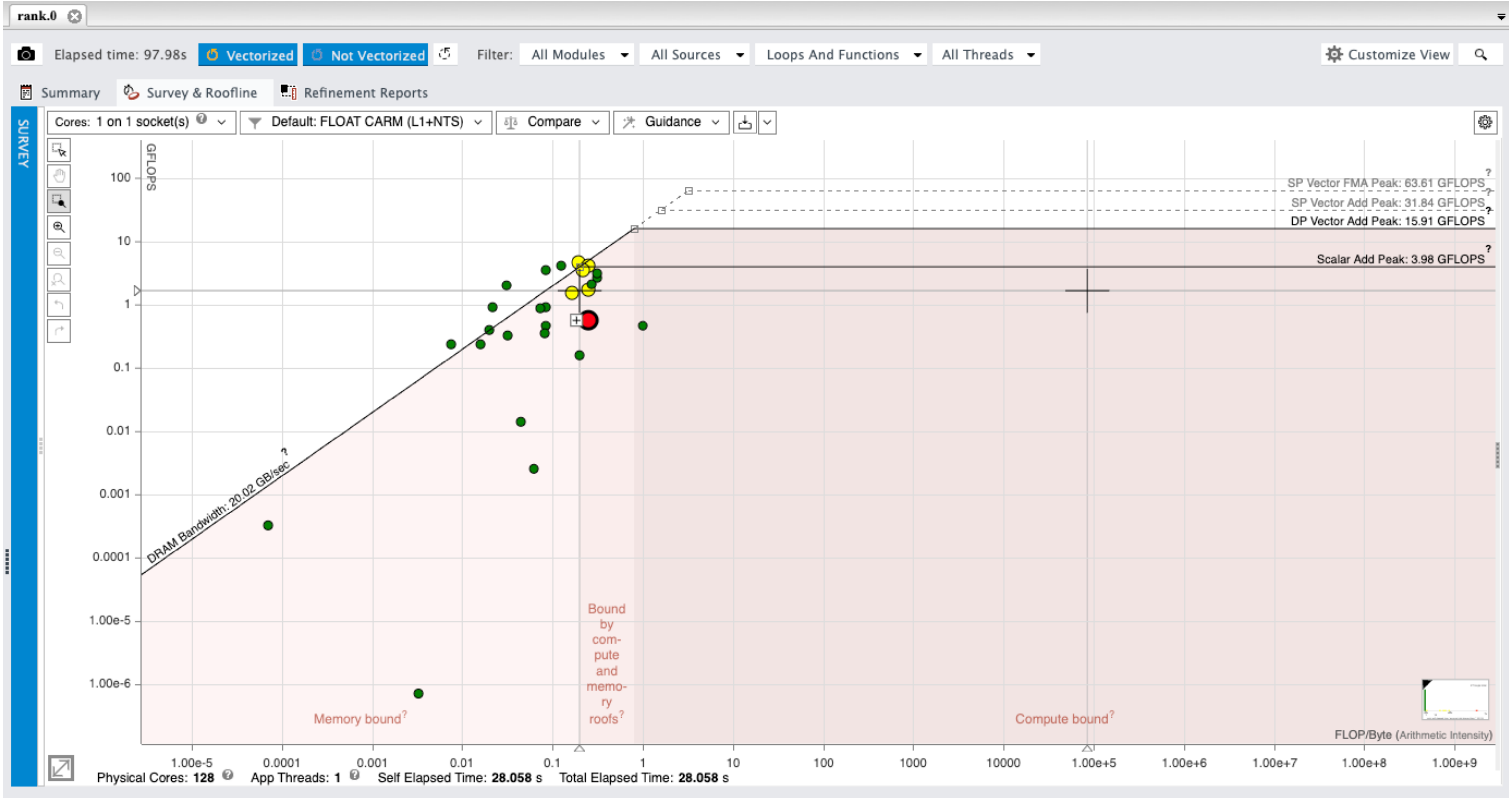
Profiling with Intel Advisor on ARCHER2

```
...
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=1
```

```
...

srun --mpi=pmi2 --distribution=block:block --hint=nomultithread \
    advixe-cl --collect=survey --project-dir=${ADVISOR_DIR} -- \
        ${APP_EXE} ${APP_PARAMS}

srun --mpi=pmi2 --distribution=block:block --hint=nomultithread \
    advixe-cl --collect=tripcounts --flop --stacks --project-dir=${ADVISOR_DIR} -- \
        ${APP_EXE} ${APP_PARAMS}

...
```

```
...

srun ...
```

Profiling with Intel Advisor on ARCHER2



Elapsed time: 97.98s

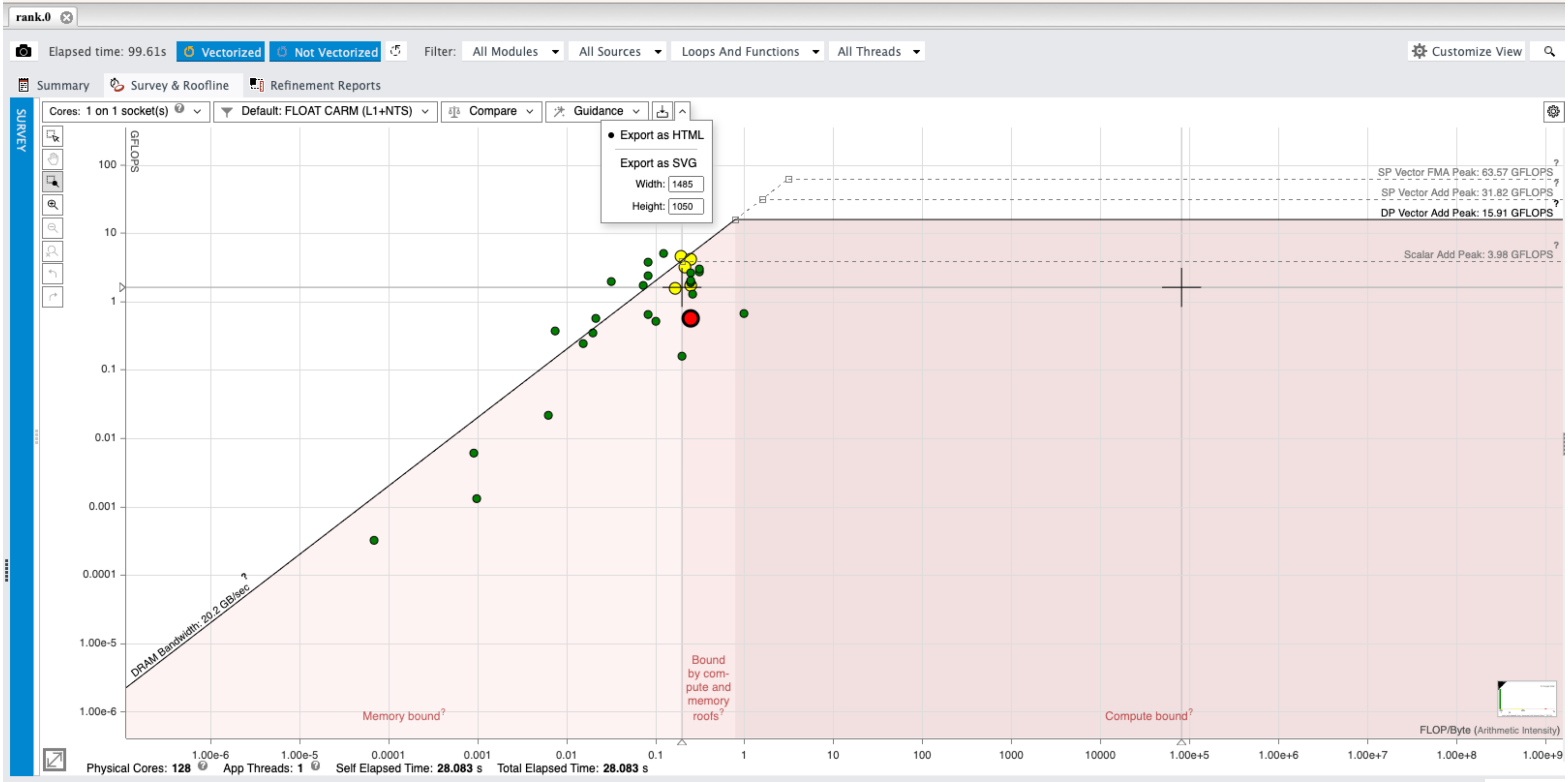
Filter: All Modules, All Sources, Loops And Functions, All Threads

Cores: 1 on 1 socket(s)

Default: FLOAT CARM (L1+NTS)

Physical Cores: 128 App Threads: 1 Self Elapsed Time: 28.058 s Total Elapsed Time: 28.058 s

Profiling with Intel Advisor on ARCHER2



Elapsed time: 99.61s

Filter: All Modules · All Sources · Loops And Functions · All Threads

Cores: 1 on 1 socket(s)

Default: FLOAT CARM (L1+NTS)

Width: 1485

Height: 1050

Physical Cores: 128 App Threads: 1 Self Elapsed Time: 28.083 s Total Elapsed Time: 28.083 s

OpenMP Offload to GPU

https://www.openmp.org/wp-content/uploads/openmp-examples-5.1.pdf
https://github.com/OpenMP/Examples/tree/v5.1

Cirrus GPU Hardware

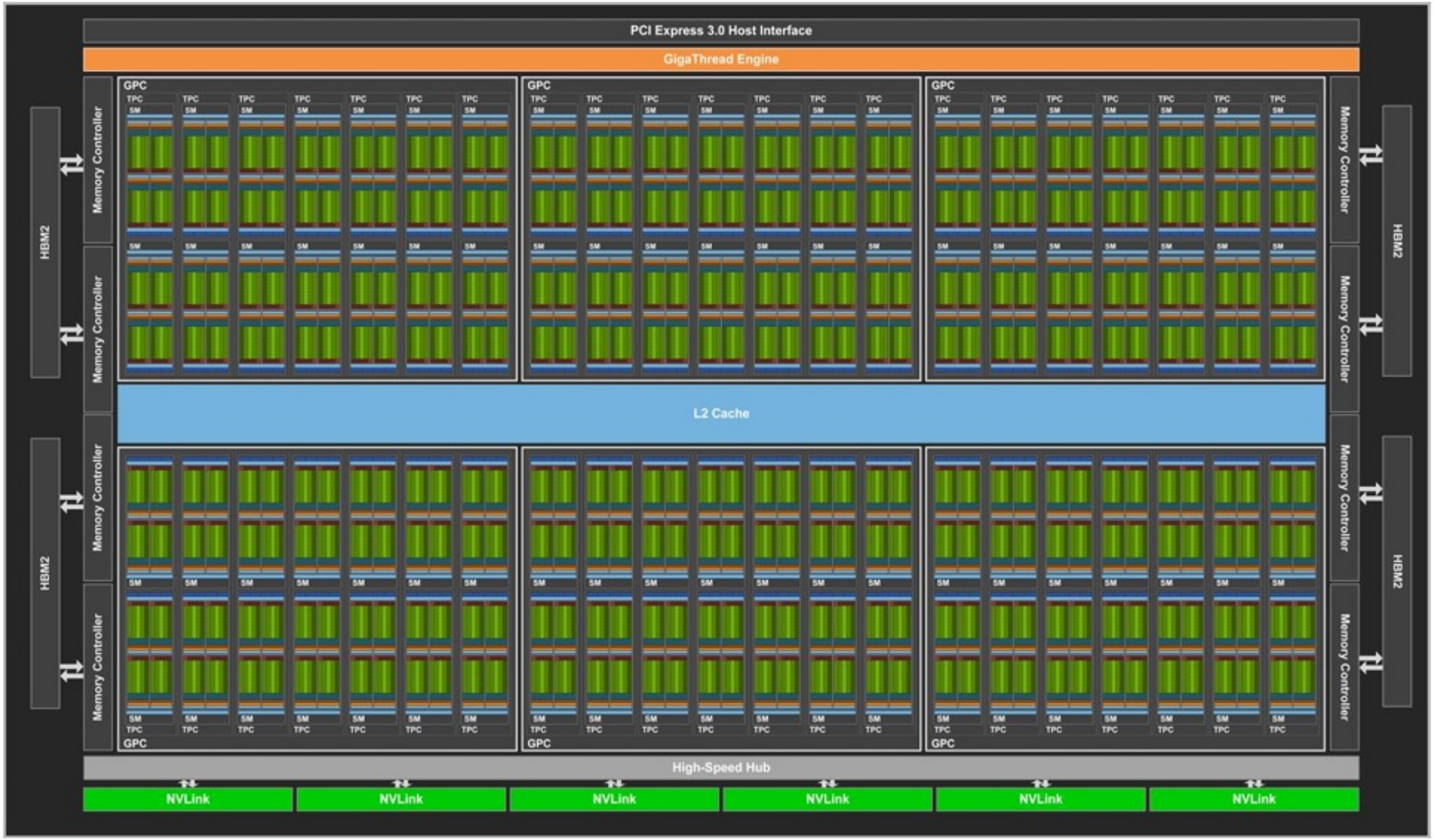
36 GPU nodes, each containing four **NVIDIA Tesla V100-SXM2-16GB (Volta)** cards

- 16 GB of high-bandwidth memory (bandwidth ≈ 900 GB/s)
- 5,120 CUDA cores
- 640 Tensor cores
- 80 Streaming multiprocessors (SM)

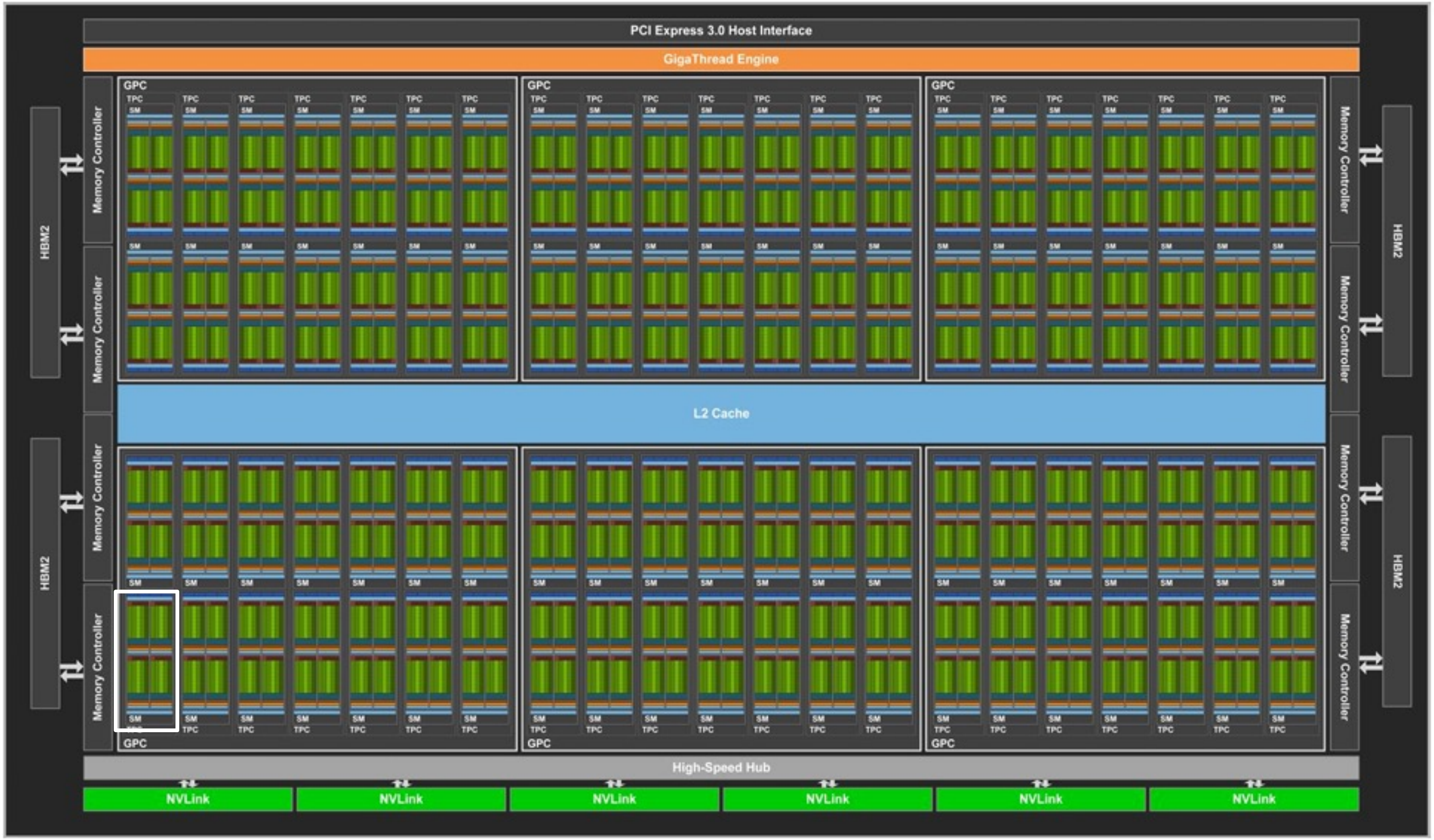
https://docs.cirrus.ac.uk/user-guide/gpu/

https://www.nvidia.com/en-gb/data-center/tesla-v100/

NVIDIA Tesla GV100 (Volta) Architecture



NVIDIA Tesla GV100 (Volta) Architecture



Cirrus programming environment

Centrally-installed TCL Modules

```
nvidia/nvhpc-nompi/22.2
openmpi/4.1.5-cuda-11.6
 gcc/12.2.0-gpu-offload
```

`mpifort` (`gfortran`) compile options

```
-O3
-fopenmp
-foffload=nvptx-none
-foffload="-lm -lgfortran -latomic"
-Wno-argument-mismatch
-std=legacy
-fdefault-real-8
-fdefault-double-8
```

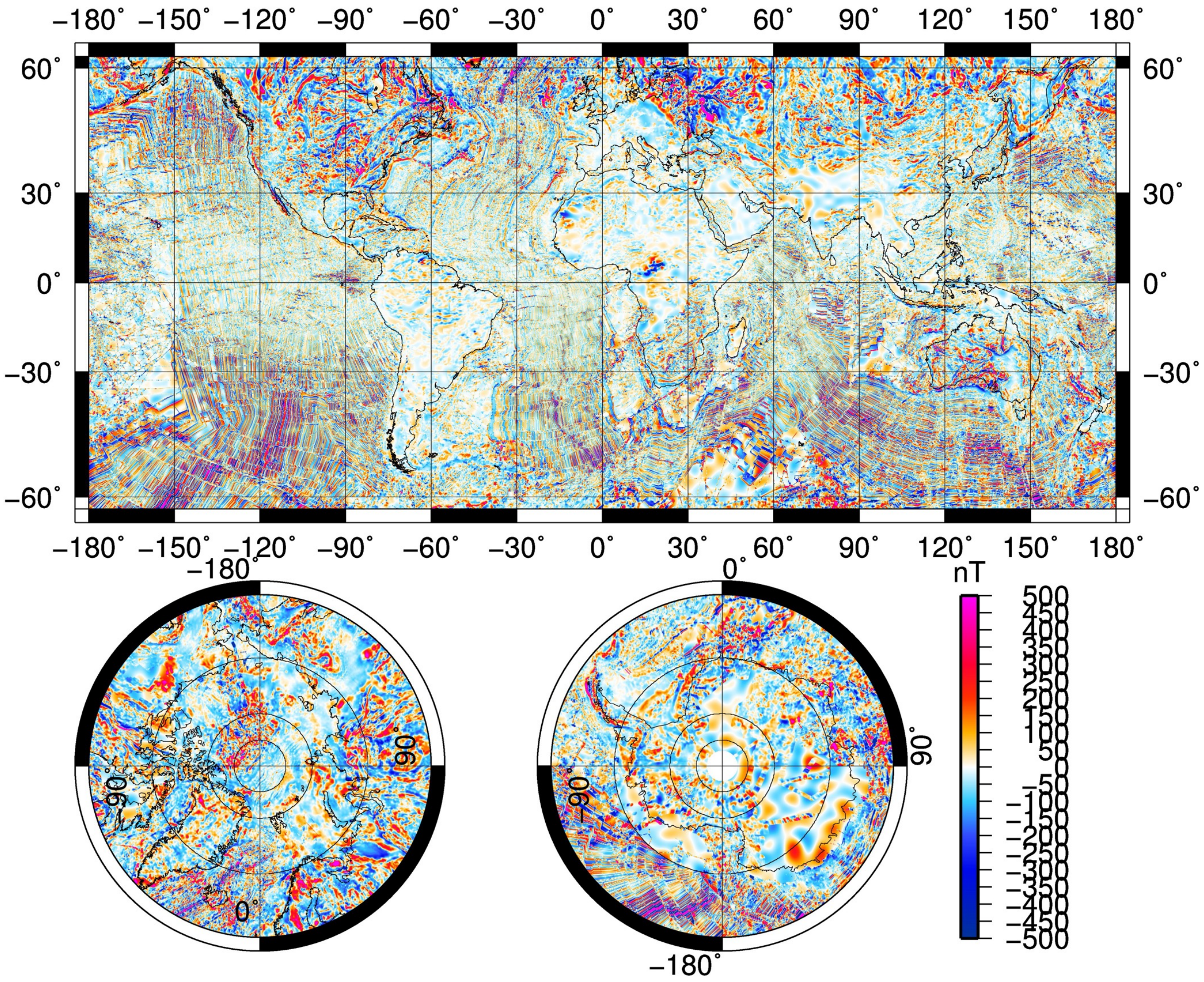
World Magnetic Anomaly Model (WMAM)

The WMAM code is used to produce spherical harmonic (SH) models of the natural magnetisation of the Earth’s lithosphere at high spatial resolution

The SH model is an interpolation of the scattered measurements of the scalar magnetic field.

These derived maps of the magnetic field can serve some important purposes.

- aid exploration for mineral deposits
- navigation
- research
  - crust thickness
  - plate tectonics



World Digital Magnetic Anomaly Map

http://wdmam.org/

WMAM hot loop

```
subroutine cpt_dat_vals_p2(...)

  integer :: nd, nb, nlocpts, nlocdatpts, shdeg
  real*8  :: d2a(0:shdeg), ppos(nd+1,nlocpts), bc(nb)
  real*8  :: wmam_fun, xyzf(nlocpts)

  ...

  do i=1,nlocpts
    xyzf(i) = wmam_fun((i > nlocdatpts),
                       shdeg, nb, nd,
                       d2a, bc, ppos(1,i))
  enddo

  ...

end subroutine
```

WMAM hot loop

```
subroutine cpt_dat_vals_p2(...)

  integer :: nd, nb, nlocpts, nlocdatpts, shdeg
  real*8  :: d2a(0:shdeg), ppos(nd+1,nlocpts), bc(nb)
  real*8  :: wmam_fun, xyzf(nlocpts)

  ...

  do i=1,nlocpts
    xyzf(i) = wmam_fun((i > nlocdatpts),
                       shdeg, nb, nd,
                       d2a, bc, ppos(1,i))
  enddo

  ...

end subroutine
```

The number of coefficients required by spherical harmonic model (SHM).

The maximum degree of SHM.

The current coefficient set for SHM.

The number of lithospheric positions assigned to MPI rank.

WMAM hot loop

```fortran
subroutine cpt_dat_vals_p2(...)

  integer :: nd, nb, nlocpts, nlocdatpts, shdeg
  real*8  :: d2a(0:shdeg), ppos(nd+1,nlocpts), bc(nb)
  real*8  :: wmam_fun, xyzf(nlocpts)

  ...

  do i=1,nlocpts
    xyzf(i) = wmam_fun((i > nlocdatpts),
                       shdeg, nb, nd,
                       d2a, bc, ppos(1,i))
  enddo

  ...

end subroutine
```

```
shdeg=200
resolution=1.0

nb=40400
nd=7

nlocpts=36261

d2a(201)
ppos(8,36261)
bc(40400)
xyzf(36261)
```

Rank 0 data sizes

WMAM hot loop

```
subroutine cpt_dat_vals_p2(...)

  integer :: nd, nb, nlocpts, nlocdatpts, shdeg
  real*8  :: d2a(0:shdeg), ppos(nd+1,nlocpts), bc(nb)
  real*8  :: wmam_fun, xyzf(nlocpts)

  ...

  do i=1,nlocpts
    xyzf(i) = wmam_fun((i > nlocdatpts),
                       shdeg, nb, nd,
                       d2a, bc, ppos(1,i))
  enddo

  ...

end subroutine
```

```
shdeg=200
resolution=1.0

nb=40400
nd=7

nlocpts=36261

d2a(201)          1.6 kB
ppos(8,36261)     2.3 MB
bc(40400)         323 kB
xyzf(36261)       290 kB
```

Rank 0 data sizes

OpenMP offload directives

```
subroutine cpt_dat_vals_p2(...)

!$OMP TARGET TEAMS DISTRIBUTE PARALLEL DO
!$omp& map(to: nb, nd, nlocpts, nlocdatpts, shdeg)
!$omp& map(to: d2a(0:shdeg))
!$omp& map(to: ppos(1:nd+1,1:nlocpts))
!$omp& map(to: bc(1:nb))
!$omp& map(from: xyzf(1:nlocpts))
!$omp& default(shared)
!$omp& schedule(static)
  do i=1,nlocpts
    xyzf(i) = wmam_fun(...)
  enddo
!$OMP END TARGET TEAMS DISTRIBUTE PARALLEL DO

end subroutine
```

```
shdeg=200

nb=40400
nd=7

nlocpts=36261
```

Rank 0 data sizes

OpenMP offload directives

```
subroutine cpt_dat_vals_p2(...)
...
logical, save :: firstcall = .TRUE.

!$OMP TARGET ENTER DATA if(firstcall)
!$omp& map(to: nb, nd, nlocpts, nlocdatpts, shdeg)
!$omp& map(to: d2a(0:shdeg))
!$omp& map(to: ppos(1:nd+1,1:nlocpts))
  if (firstcall) firstcall = .FALSE.

!$OMP TARGET TEAMS DISTRIBUTE PARALLEL DO
!$omp& map(to: bc(1:nb))
!$omp& map(from: xyzf(1:nlocpts))
...
  do i=1,nlocpts
    xyzf(i) = wmam_fun(...)
  enddo
!$OMP END TARGET TEAMS DISTRIBUTE PARALLEL DO

end subroutine
```

```
shdeg=200

nb=40400
nd=7

nlocpts=36261
```

Rank 0 data sizes

OpenMP offload – one rank per GPU

```
program mod_wmam

   ...

   call omp_set_default_device(MOD(rank,4))

   ...

end program
```

Essential, otherwise all ranks on the node use the same GPU.

OpenMP offload – one rank per GPU

```
program mod_wmam

   ...

   call omp_set_default_device(MOD(rank,4))

   ...

end program
```

Essential, otherwise all ranks on the node use the same GPU.

Code snippet above is suitable for single node job and would accommodate having multiple ranks per node, assuming that host-to-device memory bandwidth and #SMs are sufficient.

You would need to use node-local rank if running across multiple GPU nodes.

OpenMP offload teams

```
!$OMP TARGET TEAMS DISTRIBUTE PARALLEL DO
!$omp& num_teams(80) thread_limit(32)
!$omp& map(to: bc(1:nb))
!$omp& map(from: xyzf(1:nlocpts))
...
  do i=1,nlocpts
    xyzf(i) = wmam_fun(...)
  enddo
!$OMP END TARGET TEAMS DISTRIBUTE PARALLEL DO
```

It’s possible to specify the number of teams (GPU SMs) over which to distribute the loop iterations.
Can also specify the maximum number of threads per team (based on number of cores within SM).

In practice, these attributes do not need to be set as optimum values are chosen automatically.

Results and Conclusions

- Running 4 ranks on a single node, fastest (best of three) execution time is 1033 s
- If `cpt_dat_vals_p2()` loop offloaded to GPU (one per rank), fastest execution time is **452 s**.
- There is another hot loop that could be offloaded but this one requires an element-wise array reduction (of size 40400 doubles) and has a more complicated loop body.
  - Work in progress...
- For higher spherical harmonic degrees, e.g., 1440 and 2000, much more memory is required.
  - How easy is it to use pinned memory?
- Cirrus GPUs are rather old, a couple of generations behind NVIDIA H100.
  - NVIDIA A100 and AMD MI210/250 (ROCm 5) should improve performance further.