Roofline Model Analysis at LANL

Ronald W. Green
CCS-7

Future Architectures and Applications

UNCLASSIFIED

Outline

- Performance Analysis at LANL
  - Who, What, When
- Feedback on Intel Advisor XE Roofline Analysis tool
- Next Steps

Perf. Analysis at LANL - WHO

- Several Org Groups involved in performance
  - HPC: HPC Applications Readiness Team
    - *ATS clusters*: Port and Tune ASC and related production codes to new HPC clusters
  - CCS: IC Advanced Arch. & Apps team
    - *CTS clusters*, some ATS use (Open Sci period)
    - CTS Users, Open Science Apps
  - Other: misc codes with other funding
    - Exascale, DoD, NSF, etc.

Perf. Analysis at LANL - WHO

- Common threads:
  - Often physicists but with strong CS backgrounds, HPC experience
  - Multi-architecture and multi-vendor charters
  - Have a wealth of tools to choose from

Perf. Analysis at LANL - WHAT

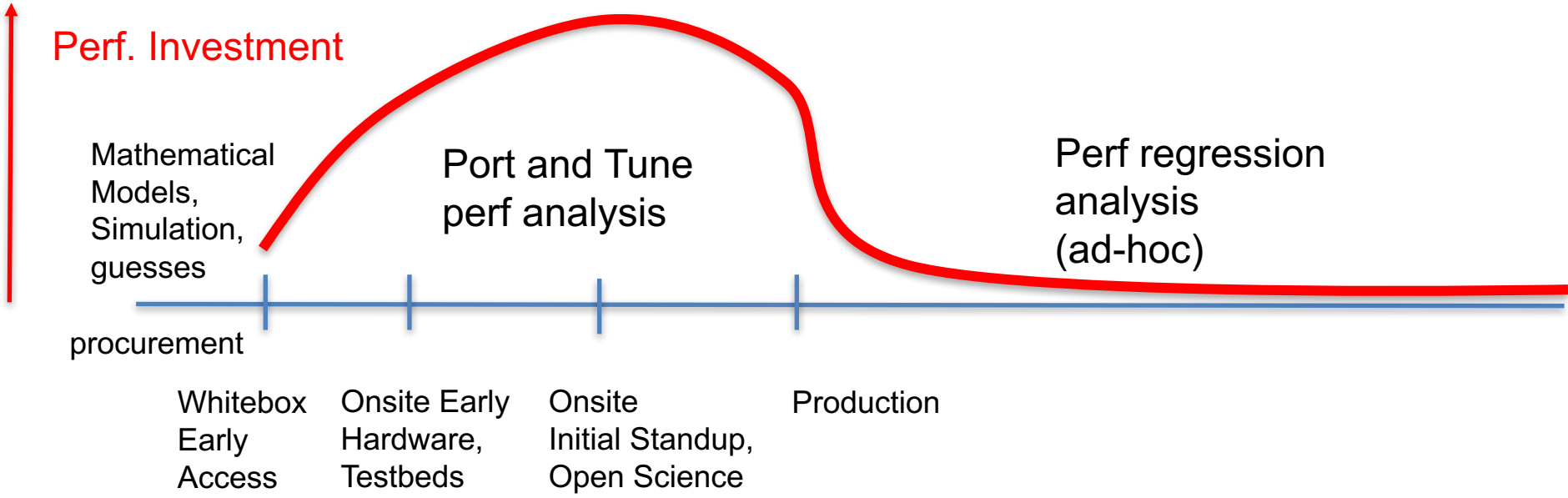
- Codes Profiled with Advixe Roofline:
  - Open Science: VPIC, Genesis, Petavision
  - Others: MPAS-Ocean, CICE, SNAP, key ASC codes
- Toolbox includes: CrayPat, Intel PSXE, Allinea MAP, NVPROF, PGPROF, Tau, Vampir, gprof, etc.

Perf. Analysis at LANL - WHAT

- Mostly static set of Codes
  - Some new code development but mostly within same codes (new features)
  - Code profiles well-known
  - Often with Internal timers around key kernels
- On Intel Architecture
  - Vtune & Intel compiler opt-reports well known and used,

Perf. Analysis at LANL - WHEN

- Investments and interest in Perf: Timeline



Cluster Life Timeline

Feedback on
Intel Advisor XE Roofline Analysis Tool
All Groups and Teams

Feedback on Advixe Roofline Tool
What We Like

- **Advisor suggests code changes, pragmas, compiler options to improve performance in language a programmer can (almost) understand**
- **Loop focus and code focus (not cryptic micro-architecture)**
- Convenience: Opt-report data is integrated in report
  - no hunting around for opt-report files
  - No separate windows needed for opt-reports
- Tripcounts for loops helpful; are not in Vtune
- Mixed data type expresssions called out
- Roofline gives quick visual to indicate memory bound, compute-bound
- Roofline shows performance to roofline peaks on specific architecture

Feedback – Enhancements

- History/Journaling – would be good to compare code changes for particular loops
  - How did my changes affect AI – roofline display for version A, B, (C, D, etc) to show progress
    - Currently need to save objects, opt-reports, sources for a collection (too much to save/copy etc)
    - Mark a loop: save data on version, AI, opt-report for that loop (source?) for subsequent comparison?
- Reliable Start/Stop API for complex code
  - Once per run
  - Start/Stop many (millions) of times per run

Feedback on Advixe Roofline Tool What We Did Not Like

- Collection is cumbersome and problematic
  - New users try to collect ALL ranks for entire run
    - Is it every really necessary to gather more than 1 rank?
    - Need to educate users before first use
      - Target 1 rank or …
      - Get better testcases with less ranks, less runtime
  - Areas of interest may only get invoked after hours/days of a long run
    - Is API for start/stop working in 2018??

Feedback on Advixe Roofline Tool Comments from Users

- I'd like to use/like it, but so far I've not seen too much evidence that it tells me anything [more actionable] than a vec-report
- I already knew everything it told me. I cannot make improvements without wholescale re-write or blank-slate refactoring (which isn't going to happen)

Feedback on Advixe Roofline Tool Comments from Users

- It seems useful IF you are unfamiliar with your code. But our codes are well known.
- I'm trying to find a niche or value beyond what I'm getting from Vtune and opt-report

Feedback is mixed. Users familiar with Vtune and opt-report are not as favorable. Others like the code-focus in Advisor, and the suggestions for improving code (something Vtune does not do).

Overall Consensus

- ***Conceptually***, a Roofline Analysis Model seems useful as ***part-of*** performance analysis activities
- ***Practically***,
  - Would like to see Intel Advisor XE Roofline tool mature in ease-of-use particularly in collection
  - Interest to see how this adds value above and beyond Vtune and opt-reports

Trinity Methodology

- In-house launch script shared by IC APT
  - Simplifies srun, reduces user errors on advixe-cl command line
  - We use a similar script for Vtune collection
  - Collects on Rank 0 (or any 1 rank) ONLY
- CICE on Trinity example:

```
srun -n 32 -N 4 --cpus-per-task=32 \
--hint=multithread --vm-overcommit=enable \
bash roofline_cice.sh
```

Roofline collection with Intel MPI

- CTS clusters (no KNL)
- LANL Darwin (KNL nodes)
- mpirun <mpi args> \
  –gtool “advixe-cl <args>:0” bash ./script.sh

Runs collection on a specific rank OR
any sets of ranks (lists, ranges, etc)

Script

```
DOSURVEY=1
DOTRIPCOUNTS=0

# Knights Landing supports additional data.  set to 1 for KNL, else 0.
DOKNL=1
# DOKNL=0

# Choose platform MPI - comment/uncomment as needed
# export RUN_CRAY=1
export RUN_CRAY=0

# set to null or comment out if not using numactl on KNL
# NUMACTL="numactl -p 1 "
NUMACTL=""

# what is the name of executable and args to the executable
MYAPP="./ocean_model"
ARGGS=" -n namelist.ocean -s streams.ocean"

# set full path to application
# example PATHAPP="/lustre/ttscratch1/green/collectorbug"
PATHAPP="/global/cscratch1/sd/rwgreen/mpas/MPAS/testing_and_setup/compass/wc

# set full path to working directory
# example WORKDIR=$PATHAPP
WORKDIR=$PATHAPP

# Advisor XE 2017 Project Directory
# set where to store results, project directory
#   nodename is saved with data - don't switch nodes between steps and don't
# on Cray this MUST BE A LUSTRE FILESYSTEM
PROJDIR="/global/cscratch1/sd/rwgreen/roofline_mpas2"

# ONLY NEEDED IF YOU FINALIZE AFTER COLLECTION
# set root path to source files.  Recursive search.  Multiple paths OK
#   make sure sources are not duplicated in 2 locations under the search pat
#  -search-dir src:r=$SRCSRCH2 \    etc etc
# example SRCSRCH="/lustre/ttscratch1/green/collectorbug"
SRCSRCH="/global/cscratch1/sd/rwgreen/mpas/MPAS/src"
# SRCSRCH2=""
```

```
# SURVEY collection
if [ $DOSURVEY = 1 ]; then
  # set the collection type and args
  COLL="survey"
  COLLARGS="-data-limit=0 "

  if [ $DOKNL = 1 ]; then
    COLLARGS="$COLLARGS -no-auto-finalize"
  fi

  if [ $SLURM_PROCID = 0 ]; then
    echo "Starting collection $COLL rank $SLURM_PROCID"
    advixe-cl -collect ${COLL} ${COLLARGS} \
      -project-dir ${PROJDIR} \
      -app-working-dir=$WORKDIR \
      -executable-of-interest=$MYAPP \
      -search-dir bin:r=$BINSRCH \
      -search-dir src:r=$SRCSRCH \
      -search-dir sym:r=$SYMSRCH \
       -- $NUMACTL $PATHAPP/$MYAPP $ARGGS
  else
    echo "Starting rank $SLURM_PROCID"
    $NUMACTL $PATHAPP/$MYAPP $ARGGS
  fi
fi
```

roofline_<app>.sh

- Edit script to set:
  - Collection = survey or tripcounts
  - KNL or non-KNL, numactl
    - KNL triggers option –no-auto-finalize
    - numactl is optional NUMACTL="numactl -p 1 "
  - Set app, args, app path and workdir,
  - proj-dir, binary/symbol/source search dirs
    - Search paths may not be needed with KNL –no-auto-finalize BUT helpful when/if we get a -finalize option for advixe-cl (feature requested)

Workflow Trinity roofline_xxx.sh

1. Copy script to run dir
2. Edit to add paths, etc.
3. Edit: ‘SURVEY=1 ; DOTRIPCOUNTS=0’
4. srun <args> bash roofline_xxx.sh
5. Edit: ‘SURVEY=0 ; DOTRIPCOUNTS=1’
6. srun <args> bash roofline_xxx.sh
7. Haswell terminal: open results in GUI (finalize KNL results)

OpenMPI

```
# Collections - uncomment a COLL varialbe in blocks below comment out others
# choose your KNOBS carefully, you may or may not need them

# ... General Exploration ...
# COLL="-collect general-exploration -trace-mpi"
# KNOBS="${KNOBS} -knob analyze-openmp=true"
# KNOBS="${KNOBS} -knob enable-user-tasks=false"
# KNOBS="${KNOBS} -knob dram-bandwidth-limits=true"
# KNOBS="${KNOBS} -knob collect-memory-bandwidth=true"
# KNOBS="${KNOBS} -data-limit=10000"

# ... HPC Performance ...
# COLL="-collect hpc-performance -trace-mpi"
# KNOBS="${KNOBS} -knob sampling-interval=10"
# KNOBS="${KNOBS} -knob analyze-openmp=true"
# KNOBS="${KNOBS} -knob enable-stack-collection=true"
# KNOBS="${KNOBS} -knob collect-memory-bandwidth="true"
# KNOBS="${KNOBS} -knob dram-bandwidth-limits=true"
# KNOBS="${KNOBS} -data-limit=0"

# ... Basic Hotspots ...
COLL="-collect hotspots -trace-mpi"
# KNOBS="${KNOBS} -knob sampling-interval=10"
KNOBS="${KNOBS} -knob analyze-openmp=true"
# KNOBS="${KNOBS} -knob enable-user-tasks=false"
KNOBS="${KNOBS} -data-limit=0"
```

```
export PMI_RANK=${OMPI_COMM_WORLD_NODE_RANK}
export PMI_PE=${OMPI_COMM_WORLD_NODE_RANK}

if [ $ONERANK = 1 ] ; then
  if [ $PMI_RANK = $THERANK ]; then
      amplxe-cl $COLL \
        $KNOBS \
        -r $PROJDIR \
        -search-dir=$SYMSRCH \
        -search-dir=$BINSRCH \
        -source-search-dir=$SRCSRCH \
       -- $PATHAPP/$MYAPP $ARGGS
  else
      $PATHAPP/$MYAPP $ARGGS
  fi
else
        amplxe-cl $COLL \
        $KNOBS \
        -r $PROJDIR \
        -search-dir=$SYMSRCH \
        -search-dir=$BINSRCH \
        -source-search-dir=$SRCSRCH \
       -- $PATHAPP/$MYAPP $ARGGS
fi
```

Next Steps

Next Steps

- Work with LANL Ptools team to get PSXE 2018 Update 1 installed everywhere
- Integrate Advixe Roofline tool in upcoming (2018) Vectorization for AVX-512 class to support upcoming Skylake-based systems

Questions / Discussion

Backup

Trinity Methodology

- In-house launch script shared by IC APT
  - Simplifies srun, reduces user errors on advixe-cl command line
  - We use a similar script for Vtune collection
  - Collects on Rank 0 (or any 1 rank) ONLY
- CICE on Trinity example:

```
srun -n 32 -N 4 --cpus-per-task=32 \
--hint=multithread --vm-overcommit=enable \
bash roofline_cice.sh
```

roofline_<app>.sh

- Edit script to set:
  - Collection = survey or tripcounts
  - KNL or non-KNL, numactl
    - KNL triggers option –no-auto-finalize
    - numactl is optional NUMACTL="numactl -p 1 "
  - Set app, args, app path and workdir,
  - proj-dir, binary/symbol/source search dirs
    - Search paths may not be needed with KNL –no-auto-finalize BUT helpful when/if we get a -finalize option for advixe-cl (feature requested)

Workflow Trinity roofline_xxx.sh

1. Copy script to run dir
2. Edit to add paths, etc.
3. Edit: ‘SURVEY=1 ; DOTRIPCOUNTS=0’
4. srun <args> bash roofline_xxx.sh
5. Edit: ‘SURVEY=0 ; DOTRIPCOUNTS=1’
6. srun <args> bash roofline_xxx.sh
7. Haswell terminal: open results in GUI (finalize KNL results)

Workflow - Futures

- Possible: New outer-wrapper script that generates the launch script, sbatches both the survey and tripcount collection
- Possible: A GUI or Webpage to browse paths, set up a batch script(s), sbatch the collection

Roofline collection with Intel MPI

- CTS clusters (no KNL)
- LANL Darwin (KNL nodes)
- mpirun <mpi args> \
  –gtool “advixe-cl <args>:0” ./myapp.exe

Runs collection on a specific rank OR
any sets of ranks (lists, ranges, etc)

2016/2017 LANL Experiences with Roofline Model Analysis

Experience – Early bugs

- Intel had not tested on Cray ALPs Moab/Torque stack used at LANL
- Had to use argument –trace-mpi
- Also had to manually set PMI_ vars manually (not set by ALPS)

```
# Choose platform MPI - comment uncomment as needed
export RUN_CRAY=1
# The following env vars are needed for Cray clusters
if [ $RUN_CRAY = 1 ]; then
  export PMI_RANK=${ALPS_APP_PE}
  export PMI_PE=${ALPS_APP_PE}
  export PMI_NO_FORK=1
fi
```

No longer an issue, LANL now using SLURM

Experiences - Limitations

- On a Cray, must use LUSTRE for project directory
- Limitations on collection:
  - KNL too many ranks*threads for complete node collection
  - In practice ~ (#ranks * #threads) <= 16 or 32
  - Intel recommends N-1 total threads where N=#cores
    - Have not come close to that with VPIC (4 ranks @ 4 threads)
  - This is why our run script only collects Rank 0
- Experience: Users always try to collect the entire application and fail (bad out of box)
- Question/Education: why would you want or need more than 1 rank's data??

Experiences - Overhead

- Survey is fast, tripcounts is not
  - Survey: ~1.1x slowdown
  - Tripcounts ~3x-5x slowdown, 8x possible
  - -flops-and-masks is heavy overhead but critical to flop counting
- Finalization – depends on application
  - MPAS-Ocean Broadwell ~1hour finalization

Help with Finalization Slowdown

- Options from NERSC Roofline Workshop (Intel's Zakhar Matveev)
- Change default call stacks processing mode, especially Fortran

```
-collect survey -stackwalk-mode=online \
                -no-stack-stitching …
```

- Focus processing to only modules of interest:

```
-collect survey \
  -module-filter-mode=include \
  -module-filter=myHotspot.so …
```

Help with Slow Tripcounts

- Default: -collect tripcounts –flops-and-masks
- Flops only, don't collect tripcounts:

```
-collect tripcounts –flops-and-masks –no-trip-counts
```

- Tripcounts only, no FLOPS

```
-collect tripcounts
```

Note: this disables Roofline feature, gives traditional Vector Advisor info

Case Study: MPAS-Ocean

Roofline Summary, Rank 0

Summary | Survey & Roofline | Refinement Reports

Vectorization Advisor

Vectorization Advisor is a vectorization analysis toolset that lets you identify loops that will benefit most from vector parallelism, discover performance issues preventing from effective vectorization and characterize your memory vs. vectorization bottlenecks with Advisor Roofline model automation.

Program metrics

Elapsed Time	21.98s		
Vector Instruction Set	AVX, AVX2, SSE2	Number of CPU Threads	4
Total GFLOP Count	64.52	Total GFLOPS	2.94
Total Arithmetic Intensity	0.09		

Loop metrics

Total CPU time	69.89s	100.0%
Time in **83** vectorized loops	7.79s	11.1%
Time in scalar code	62.10s	88.9%

Vectorization Gain/Efficiency

Vectorized Loops Gain/Efficiency	2.40x	63%
Program Approximate Gain	1.16x	

Top time-consuming loops

Loop	Self Time	Total Time	Trip Counts
[loop in ocn_tracer_advection_mono_tend at mpas_ocn_tracer_advection_mono.f90:301]	4.170s	4.170s	81
[loop in ocn_tracer_advection_mono_tend at mpas_ocn_tracer_advection_mono.f90:343]	1.300s	1.300s	64
[loop in ocn_time_integrator_split at mpas_ocn_time_integration_split.f90:1001]	1.190s	1.869s	397
[loop in ocn_diagnostic_solve at mpas_ocn_diagnostics.f90:357]	1.020s	1.020s	61
[loop in ocn_fuperp at mpas_ocn_diagnostics.f90:1131]	0.922s	0.922s	5

Recommendations

Loop	Self Time	Recommendations
[loop in ocn_vel_coriolis_tend at mpas_ocn_vel_coriolis.f90:175]	0.740s	Add data padding
[loop in ocn_time_integrator_split at mpas_ocn_time_integration_split.f90:1007]	0.680s	Add data padding
[loop in pgrad_jacobian_common_level at mpas_ocn_vel_pressure_grad.f90:399]	0.650s	Add data padding
[loop in ocn_equation_of_state_jm_density at mpas_ocn_equation_of_state_jm.f90:342]	0.640s	Add data padding
[loop in ocn_equation_of_state_jm_density at mpas_ocn_equation_of_state_jm.f90:386]	0.410s	Add data padding

Collection details

Platform information

MPI rank	0
CPU Name	Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz
Frequency	2.10 GHz
Logical CPU Count	72

Just 2GF per rank @ 4 threads

< 0.1 Flops per byte!

Low overall vectorization

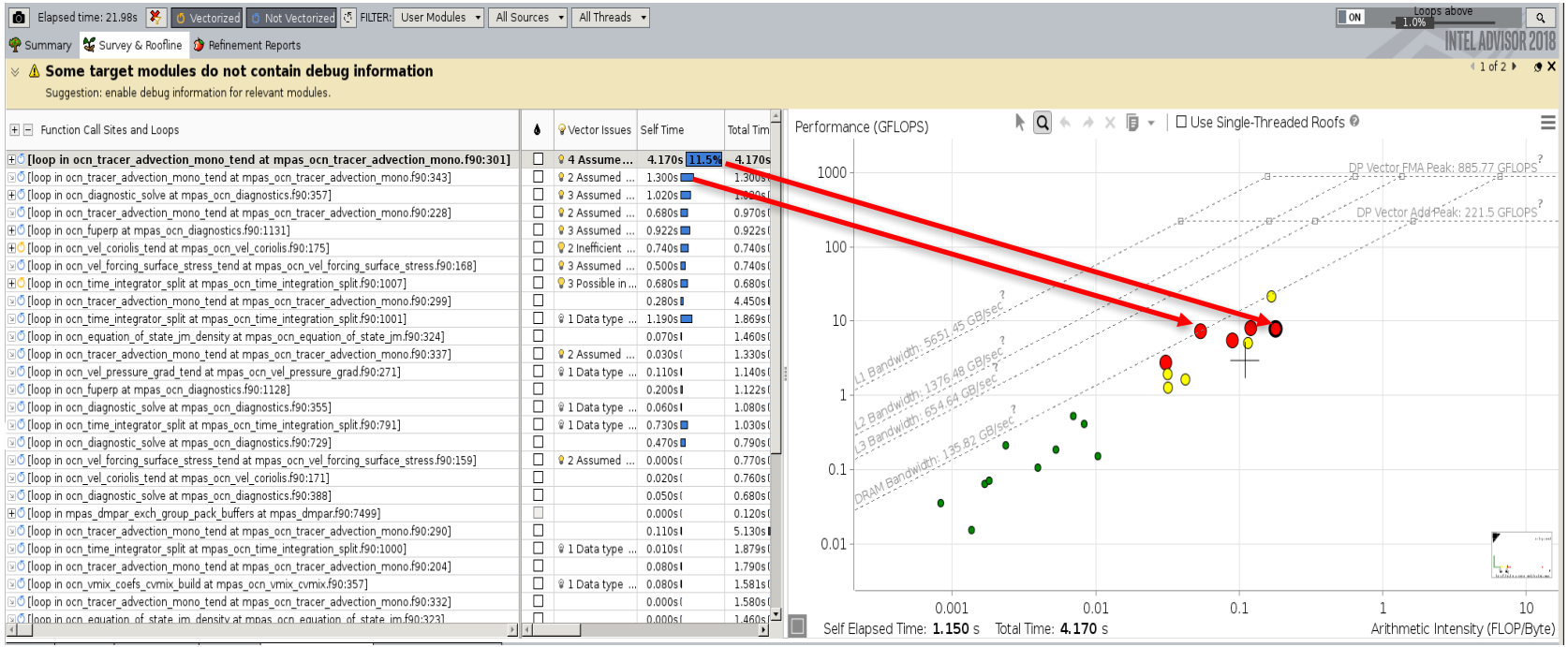
mpas_ocn_tracer_advection_mono

Low trip counts, non-vector length loops/arrays

Here's an advantage of Advisor over Vtune – there's obviously something systemically wrong with the data structures

Roofline Graph, > 1%+ of Runtime

Notes: Low Arithmetic Intensity. < 10GFs inside key loops (peak 1TF).
Under DRAM bandwidth curve, well under L1 bandwidth



Top Loops: mpas_ocn_tracer_advection_mono_tend:301

- mpas_ocn_tracer_advection_mono_tend:301
  - Scalar, non-vectorized
    - Vector dependency
  - Multi-versioned due to assumed-shaped arrays (striding possible)
    - 1 vector candidate (non-strided) BUT not vectorized due to dependence, 1 scalar loop strided (not cost effective to vectorize due to strided loads/stores)
  - FMA used (scalar)
  - 1 type convert (integer(8) to real(8)) – Integer mask array (WHY??)
    - highOrderAdvectionMask(k, iEdge) * (adv_coefs(i,iEdge)
    - Mask variables throughout code seem to be Integer(8) also!
  - Avg trip count = 90 iterations, call count 18,557,280
  - 8.582 Gflops,
  - Application Intensity (AI) flops/byte = 0.179

Looking at Recommendations

Source | Top Down | Code Analytics | Assembly | Recommendations | Why No Vectorization?

All known issues with all possible recommendations: C++ / Fortran

Issue: Assumed dependency present

The compiler assumed there is an anti-dependency (Write after read - WAR) or a true dependency (Read after write - RAW) in the loop. Improve performance by investigating the assumption and handling accordingly.

Recommendation: Confirm dependency is real — Confidence: Need More Data

There is no confirmation that a real (proven) dependency is present in the loop. To confirm: Run a Dependencies analysis.

Issue: Potential underutilization of FMA instructions

Your current hardware supports the AVX2 instruction set architecture (ISA), which enables the use of fused multiply-add (FMA) instructions. Improve performance by utilizing FMA instructions.

Recommendation: Force vectorization if possible — Confidence: Low

The loop contains FMA instructions (so vectorization could be beneficial) but is not vectorized. To fix: Review corresponding compiler diagnostics to check if vectorization enforcement is possible and profitable.

Read More:

- Vectorization Resources for Intel® Advisor Users

Issue: Inefficient memory access patterns present

There is a high of percentage memory instructions with irregular (variable or random) stride accesses. Improve performance by investigating and handling accordingly.

Recommendation: Use the Fortran 2008 CONTIGUOUS attribute — Confidence: Low

The loop is multi-versioned for unit and non-unit strides in assumed-shape arrays or pointers, but marked versions of the loop have unit stride access only. The CONTIGUOUS attribute specifies the target of a pointer or an assumed-shape array is contiguous. It can make it easier to enable optimizations that rely on the memory layout of an object occupying a contiguous block of memory. Note: The results are indeterminate and could result in wrong answers and segmentation faults if the user assertion is wrong and the data is not contiguous at runtime.

Example:

```
real, pointer, contiguous :: ptr(:)
  real, contiguous :: arrayarg(:, :)
```

Read More:

- CONTIGUOUS
- Fortran Array Data and Arguments and Vectorization and Vectorization Resources for Intel® Advisor Users

Issue: Data type conversions present

There are multiple data types within loops. Utilize hardware vectorization support more effectively by avoiding data type conversion.

Recommendation: Use the smallest data type — Confidence: Low

The source loop contains data types of different widths. To fix: Use the smallest data type that gives the needed precision to use the entire vector register width.

Example: If only 16-bits are needed, using a short rather than an int can make the difference between eight-way or four-way SIMD parallelism, respectively.

Pointers!

OMP SIMD

CONTIGUOUS!

Mixing data types

Looking at Opt Report for Top Loop

CONTIGOUS would remove striding assumption, multiversioning and efficiency

```
LOOP BEGIN at mpas_ocn_tracer_advection_mono.f90(300,13)
 <Multiversioned v1>
   remark #25233: Loop multiversioned for stride tests on Assumed shape arrays
   remark #15344: loop was not vectorized: vector dependence prevents vectorization
   remark #15346: vector dependence: assumed FLOW dependence
        between HIGH_ORDER_HORIZ_FLUX(k,iedge) (305:15) and
                TRACER_CUR(k,icell) (305:15)
   remark #15346: vector dependence: assumed ANTI dependence
        between TRACER_CUR(k,icell) (305:15) and
                HIGH_ORDER_HORIZ_FLUX(k,iedge) (305:15)
 LOOP END
```

HIGH_ORDER_HORIZ_FLUX and TRACER_CUR are pointers to real 2d arrays, compiler assumes could point to same data (obviously they never will)

The Loop in Question

Integer mask array causes costly type conversion:
could this be RKIND (0.0 or 1.0)?

```
! Compute 3rd or 4th fluxes where requested.
do i = 1, nAdvCellsForEdge(iEdge)
  iCell = advCellsForEdge(i,iEdge)
  do k = 1, maxLevelCell(iCell)
    tracer_weight = highOrderAdvectionMask(k, iEdge) * (adv_coefs(i,iEdge) &
                  + coef_3rd_order*sign(1.0_RKIND,normalThicknessFlux(k,iEdge))*adv_coefs_3rd(i,iEdge))
    tracer_weight = normalThicknessFlux(k,iEdge)*tracer_weight
    high_order_horiz_flux(k,iEdge) = high_order_horiz_flux(k,iEdge) + tracer_weight * tracer_cur(k,iCell)
  end do ! k loop
end do ! i loop over nAdvCellsForEdge
```

Pointers to 2D R_KIND arrays, could be aliased
can't vectorize w/o OMP SIMD directive

Vectorization Example and Suggestions

Study: Take 2 Top hotspots and Vectorize

```
integer, dimension(:,:),   contiguous, pointer :: highOrderAdvectionMask
real (kind=RKIND), dimension(:,:), contiguous, intent(in) :: normalThicknessFlux
real (kind=RKIND), dimension(:,:), contiguous, intent(in) :: adv_coefs
real (kind=RKIND), dimension(:,:), contiguous, intent(in) :: adv_coefs_3rd

real (kind=RKIND), dimension(:,:), contiguous, pointer :: tracer_cur, upwind_tendency, flux_outgoing, flux_incoming
real (kind=RKIND), dimension(:,:), contiguous, pointer :: tracer_new, inv_h_new, tracer_max, tracer_min
real (kind=RKIND), dimension(:,:), contiguous, pointer :: high_order_horiz_flux, high_order_vert_flux
```

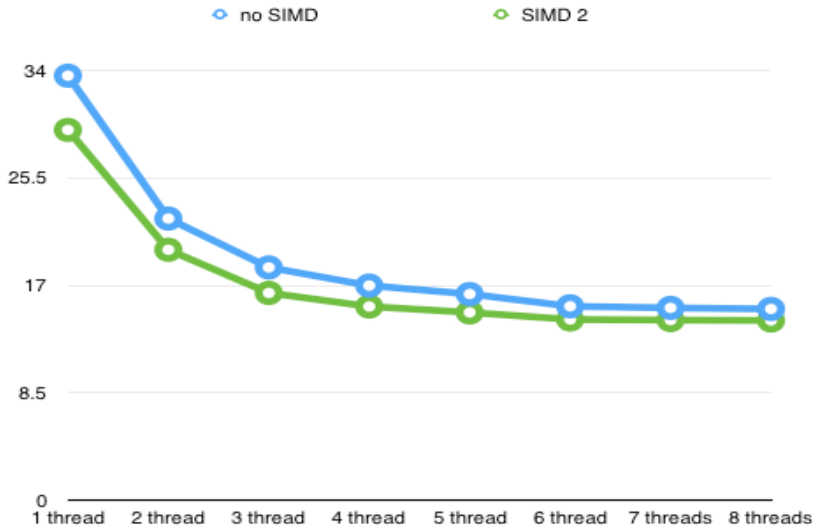
```
! Compute 3rd or 4th fluxes where requested.
!$omp simd private(tracer_weight)
do i = 1, nAdvCellsForEdge(iEdge)
  iCell = advCellsForEdge(i,iEdge)
  do k = 1, maxLevelCell(iCell)
    tracer_weight = highOrderAdvectionMask(k, iEdge) * (adv_coefs(i,iEdge) &
                  + coef_3rd_order*sign(1.0_RKIND,normalThicknessFlux(k,iEdge))*adv_coefs_3rd(i,iEdge))
    tracer_weight = normalThicknessFlux(k,iEdge)*tracer_weight
    high_order_horiz_flux(k,iEdge) = high_order_horiz_flux(k,iEdge) + tracer_weight * tracer_cur(k,iCell)
  end do ! k loop
end do ! i loop over nAdvCellsForEdge
```

Study: Take 2 Top hotspots and Vectorize

```
LOOP BEGIN at mpas_ocn_tracer_advection_mono.f90(304,13)
   remark #15389: vectorization support: reference HIGH_ORDER_HORIZ_FLUX(k,iedge) has unaligned access
      [ mpas_ocn_tracer_advection_mono.f90(309,15) ]
   remark #15389: vectorization support: reference HIGH_ORDER_HORIZ_FLUX(k,iedge) has unaligned access
      [ mpas_ocn_tracer_advection_mono.f90(309,48) ]
   remark #15389: vectorization support: reference normalthicknessflux(k,iedge) has unaligned access
      [ mpas_ocn_tracer_advection_mono.f90(308,31) ]
   remark #15389: vectorization support: reference TRACER_CUR(k,icell) has unaligned access
      [ mpas_ocn_tracer_advection_mono.f90(309,97) ]
   remark #15389: vectorization support: reference highorderadvectionmask(k,iedge) has unaligned access
      [ mpas_ocn_tracer_advection_mono.f90(305,31) ]
   remark #15381: vectorization support: unaligned access used inside loop body
   remark #15305: vectorization support: vector length 8
   remark #15309: vectorization support: normalized vectorization overhead 0.282
   remark #15301: OpenMP SIMD LOOP WAS VECTORIZED
   remark #15450: unmasked unaligned unit stride loads: 4
   remark #15451: unmasked unaligned unit stride stores: 1
   remark #15475: --- begin vector cost summary ---
   remark #15476: scalar cost: 36
   remark #15477: vector cost: 8.870
   remark #15478: estimated potential speedup: 3.940
   remark #15487: type converts: 1
   remark #15488: --- end vector cost summary ---
LOOP END
```

Runs with 2 top loops vectorized

	1 thread	2 thread	3 thread	4 thread	5 thread	6 thread	7 threads	8 threads
no SIMD	33.569	22.274	18.41	16.9767	16.31	15.34	15.215	15.13
SIMD 2	29.281	19.806	16.395	15.324	14.855	14.298	14.251	14.22
speedup	15%	12%	12%	11%	10%	7%	7%	6%



Findings from Roofline/Vect Advisor

Summary

- 4 threads/rank is thread scaling elbow
- Vectorization inhibitors
  - Prolific use of pointers to 2D arrays of real kind=8 (assumed aliasing)
  - Pointers means cannot assume OMP SIMD ALIGNED, code gen unaligned accesses
  - Assumed shape array arguments need CONTIGUOUS clause (avoid indexed loads/stores)
  - # Levels (innermost loops) not even multiples of vector size (remainder loops), and small trip counts
  - OpenMP SIMD clauses can help, but needed everywhere.
- Lots of work to do

References

- **NEW!!** How to use Intel® Advisor on the Intel® Xeon Phi™ processor (codename: Knights Landing) https://software.intel.com/en-us/articles/how-to-use-intel-advisor-on-the-intel-xeon-phi-processor-codename-knights-landing
- Roofline model proposed by Williams, Waterman, Patterson: http://www.eecs.berkeley.edu/~waterman/papers/roofline.pdf
- "Cache-aware Roofline model: Upgrading the loft" (Ilic, Pratas, Sousa, INESC-ID/IST, Thec Uni of Lisbon) http://www.inesc-id.pt/ficheiros/publicacoes/9068.pdf