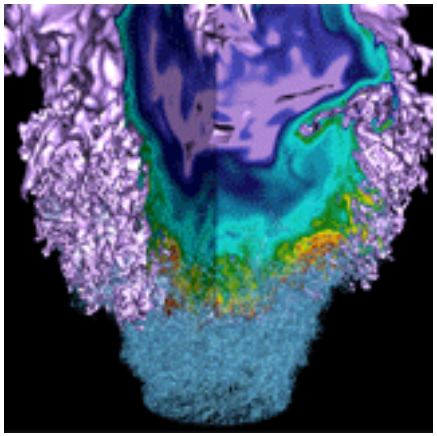
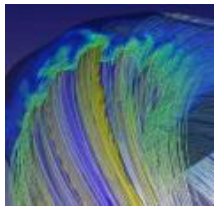
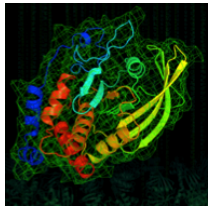
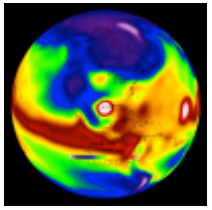
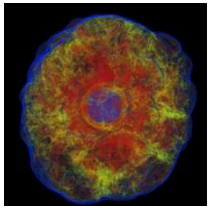
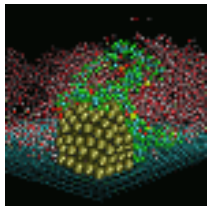
Introduction to Intel Advisor at NERSC



Mathieu Lobet, Tuomas Koskela

NERSC
November 3 2016

Purpose of Intel Advisor

Intel Advisor is composed of Vectorization advisor and Threading advisor that help to ensure that your code realizes full performance potential.

Vectorization advisor is a vectorization optimization tool:

- Help to identify time-consuming loops that can benefit from vectorization or already vectorized
- Help to identify vetorization and efficiency issues (dependencies, spilling, memory access...) and propose solutions
- Help to ensure that vectorization is safe and quantify effects of vectorization (vectorization efficiency, roofline performance model)

Factors that prevent vectorization

Loop-carried dependencies/data dependencies

```
For(i = 0 ; i < N ; i++ ) {
  A[i+M] = A[i] + B[i];
}
```

Call to sequential functions

```
For(i = 0 ; i < N ; i++ ) {
  sequential_function(i);
}
```

Pointer aliasing

```
Void function(float * A, float * B) {
  for(i = 0 ; i < N ; i++ ) {
    A[i] = B[i] * C;
  }
}
```

Inner/Outer loops

```
Do i=1,N
 A(i) = B(i) + C(i)
 S(1:M) = A(1:M)
Enddo
```

Factors that slow down vectorization

Indirect/strided memory access

```
For(i = 0 ; i < N ; i++ ) {
  A[C[i]] = B[D[i]];
}
```

Memory latency/throughput (low flop/ byte ratio)

```
For(i = 0 ; i < very big ; i++ ) {
  A[i] = c*B[i];
}
```

Small trip count not multiple of vector length

```
For(i = 0 ; i < small; i++ ) {
  A[i] = c*B[i];
}
```

Branching, divisions, exp...

```
For(i = 0 ; i < small; i++ ) {
  if (B[i] > 0) {
    A[i] = C[i]/B[i];
  }
}
```

Very Good references for Advisor

Intel Advisor Linux presentation (description of the different surveys, links toward specific command documentation):
https://software.intel.com/en-us/get-started-with-advisor-vectorization-linux

Intel Advisor Getting Started/FAQ:
https://software.intel.com/en-us/get-started-with-advisor
https://software.intel.com/en-us/articles/vectorization-advisor-faq

The NERSC website:
http://www.nersc.gov/users/software/performance-and-debugging-tools/advisor/

How to perform a collection

Compile for advisor:

```
Cray: CC -dynamic -g -openmp -O3 -xMIC-AVX512 mycode.c - o $APPNAME
Intel: CC -g -openmp -O3 -xMIC-AVX512 mycode.c - o $APPNAME
```

Perform analysis by command line :

```
salloc - N 1 - p debug - t 01:00:00
module load advisor

srun -n 1 -c 1 advixe-cl - collect $ANALYSIS
--project-dir $ANALYSIS_DIR
--search-dir src:r=$SRCDIR
--trace-mpi ./$APPNAME
```

Use the Gui interface for performing analysis on a login node:

```
salloc - N 1 - p debug - t 01:00:00
module load advisor

advixe-gui ./$ANALYSIS_DIR
```

Different kind of collections

```
srun -n 1 -c 1 advixe-cl -collect $ANALYSIS --project-dir  $ANALYSIS_DIR
--search-dir src:r=$SRCDIR --trace-mpi ./$APPNAME
```

$ANALYSIS=survey: general overview of the performances and the vectorization state of the code.

$ANALYSIS=tripcounts: improves the survey by dynamically exploring loop iteration execution and propose better decisions about your vectorization strategy. It measures #FLOP count and cumulative data traffic necessary for the Roofline performance model.

$ANALYSIS=dependencies: refine analysis by checking for real data dependencies in loops the compiler did not vectorize because of assumed dependencies.

$ANALYSIS=map: (Memory Access Pattern) refine analysis by checking for various memory issues, such as non-contiguous memory accesses and unit stride vs. non-unit stride accesses.

More information here: https://software.intel.com/en-us/get-started-with-advisor-cli-mpi

Other options

-no-auto-finalize: Results are not finalize after the run. This is useful for collections on KNL that takes longer time than on Haswell. Finalization can then be performed on Haswell or when opening the results via the GUI.

-flops-and-masks: take into account the masks in the vector operations on KNL.

Analyzing the results

Analyze the results via the GUI:

```
Module load advisor
advixe-gui ./$ANALYSIS_DIR
```

/global/project/projectdirs/mpccc/tkoskela/pmbs16/small_circ.unoptimized/advi.ixpug - Intel Advisor

File View Help

Welcome | cc01.000

Elapsed time: 13.40s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources | Loops And Functions | All Threads | Smart Mode

Summary | Survey Report | Roofline Chart | Refinement Reports

INTEL ADVISOR 2017

Function Call Sites and Loops	Vector Issues	Self Time	Ela. Ti.	Tot. Ti.	Ty.	FLOPS A... GF.	AI	Ma.	Why No Vectorization?	Vectorized Loops Ve.	Efficiency	Gain...	VL.	Trip Counts	Instruction Set Analysis Traits
derivs_elec_vec		4.336s 9.3%	0.3	13.	Fu.	2.2	0.0	10							2-Source Permutes; Appr. Reciprocals(AVX-512ER); Blends; ...
[loop in derivs_elec_vec at derivs_elec_vec.F90:124]	1 Ineffective ..	4.032s 8.7%	0.3	4.0	Ve.	3.3	0.0	12.		AV.	26%	2.08x	8	1	FMA; Gathers; Mask Manipulations; Scatters
[loop in derivs_elec_vec at derivs_elec_vec.F90:68]		3.582s 7.7%	0.3	3.5	Ve.	11.	0.4	46.		AV.	35%	2.81x	8	1; 1	Appr. Reciprocals(AVX-512ER); Divisions; FMA; Gathers; Ma...
[loop in search_tr2 at search.F90:742]	**1 Potentia...**	**3.370s 7.3%**	**0.**	**3.**	**Sc.**	**1.**	**0.**	**10.**	**loop with multiple exits...**					**1**	**FMA; Mask Manipulations**
efield_gk_elec2		3.214s 6.9%	0.2	3.2	Fu.	4.0	0.0	67.							FMA
eval_bicub_1		3.081s 6.6%	0.5	3.0	In...	4.6	0.1	10							Extracts; Unpacks
i_interpol_wo_pspline		2.728s 5.9%	0.2	2.7	Fu.	1.9	0.0								
search_tr2		1.949s 4.2%	0.2	5.3	Fu.	0.4	0.0								
bvec_interpol		1.588s	0.1	7.2	Fu.	4.9	0.1	10							
eval_bicub_2		1.438s	0.1	1.4	In...	12.	0.1	10							Extracts; Inserts; Unpacks
bicub_interpol1		1.398s	0.1	4.4	Fu.	0.8	0.0	10							Extracts; Unpacks
efield_gk_elec		1.380s	0.1	1.3	In...	1.6	0.0	67.							FMA
field		1.174s	0.1	5.3	Fu.	3.4	0.0	10							
[loop in derivs_elec_vec at derivs_elec_vec.F90:98]	1 Ineffective ..	1.094s	0.1	1.0	Ve.	1.9	0.1	19.		AV.	55%	4.38x	8	1; 1	Appr. Reciprocals(AVX-512ER); Divisions; FMA; Gathers; Ma...
derivs_single_with_e_elec_vec		0.958s	0.1	29.	Fu.										2-Source Permutes; Appr. Reciprocals(AVX-512ER); Divisio ...
field_following_pos2		0.756s	0.1	9.6	Fu.	1.7	0.0	10							Appr. Reciprocals(AVX-512ER); Divisions; FMA; Unpacks
[loop in pushe_single_vec at pushe_single_vec.F90:68]	1 Ineffective ..	0.736s	0.0	0.7	Ve.	11.	0.7	60.		AV.	43%	3.43x	8	1	Appr. Reciprocals(AVX-512ER); Divisions; FMA; Gathers; Ma...
derivs		0.681s	0.0	4.4	In...	2.4	0.0	25.							Appr. Reciprocals(AVX-512ER); Divisions; FMA
[loop in derivs_single_with_e_elec_vec at derivs_single_with_ ...	1 Data type ...	0.633s	0.0	14.	Sc.	2.4	0.0	10	inner loop was already vect ...					1	Appr. Reciprocals(AVX-512ER); Exponent extractions; FMA; ...
bicub_interpol2		0.610s	0.0	2.0	Fu.	0.6	0.0	10							Extracts; Inserts; Type Conversions; Unpacks
derivs		0.582s	0.0	4.0	In...	2.6	0.1	25.							Appr. Reciprocals(AVX-512ER); Divisions; FMA
i_interpol		0.552s	0.0	1.8	Fu.	0.5	0.0								
[loop in pushe_1step2_vec_ompparallel_for@39 at pushe_ ...		0.424s	0.0	37.	Sc.				inner loop was already vect ...						Extracts; Mask Manipulations; Permutes; Shifts
pushe_single_vec		0.400s	0.0	36.	Fu.	1.0	0.0	10							Appr. Reciprocals(AVX-512ER); Divisions; FMA; Gathers; Sc...
[loop in field_following_pos2 at charge.F90:581]		0.364s	0.0	8.8	Sc.	3.9	0.0	10	inner loop was already vect ...					1	Appr. Reciprocals(AVX-512ER); Divisions; FMA; Mask Manip...

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

All Compiler Diagnostics

Issue: Loop With Early Exits Cannot Be Vectorized Unless It Meets Search Loop Idiom Criteria

Cause: The compiler did not recognize a search idiom in a loop that may exit early. For example: The loop body contains a conditional exit or GOTO statement followed by calculations.

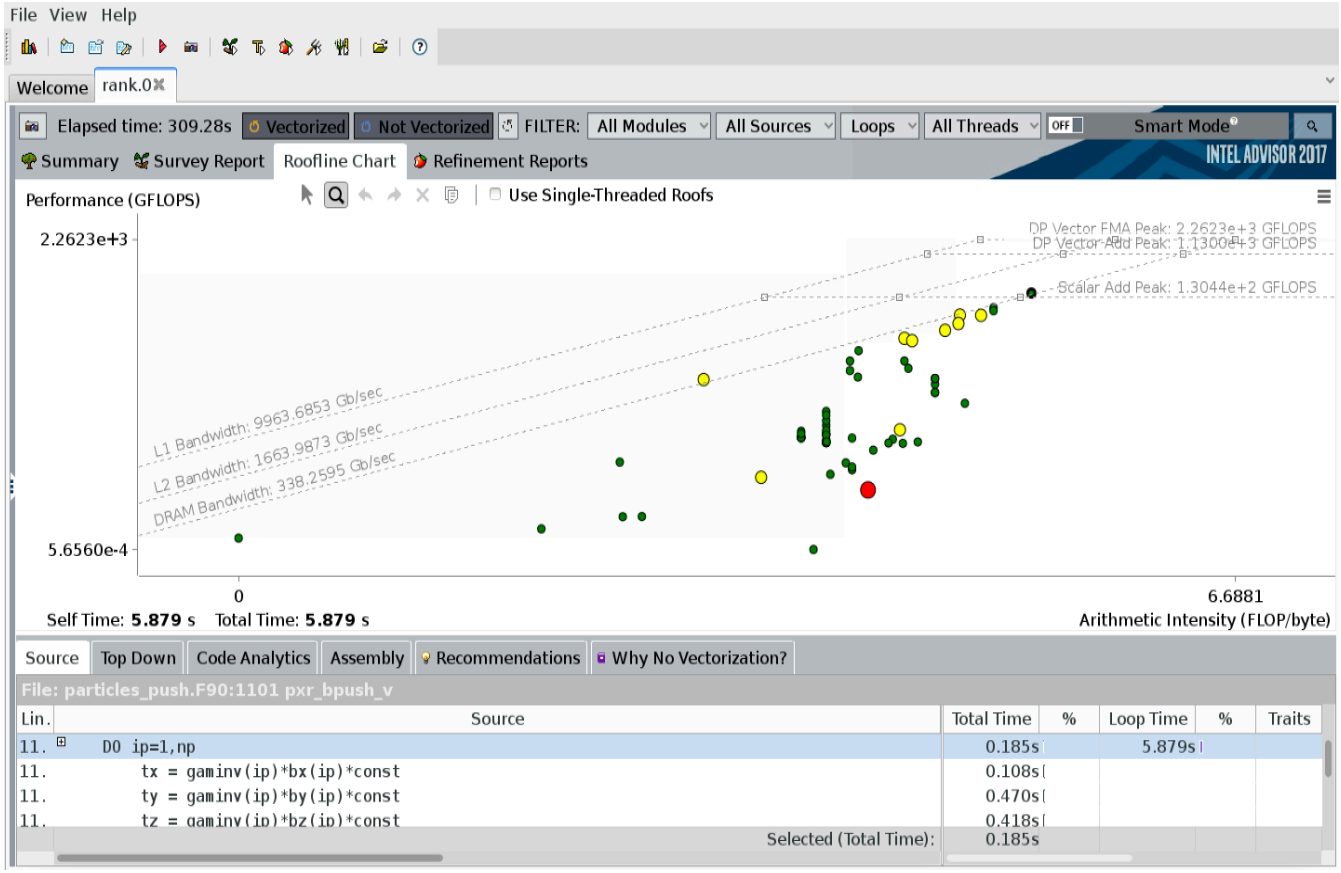
Fortran Example:

```
subroutine f15520(a,b,c,n)
  implicit none
  real, intent(in ), dimension(n) :: a, b
```

The cache-aware roofline model [2]: Roofline automation in Intel Advisor :

"Roofline is a visually intuitive performance model used to bound the performance of various numerical methods and operations running on multicore, manycore, or accelerator processor architectures." [1]

- Requires the survey and the tripcounts analysis
- Computation of the roofline: u-bench-based
- AVX-512 mask-aware
- Break-down by loops or functions
- Measure L1 <-> Register traffic: what CPU demands from memory sub-system to make a computation →Cumulative traffic through L1/L2/LLC/DRAM/MCDRAM
- GUI for quick viewing



[1] S. Williams et al. *CACM* (2009), crd.lbl.gov/departments/computer-science/PAR/research/roofline
[2] A. Ilic et al., *IEEE Computer Architecture Letters* (2014

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Very Good references for Advisor

Let’s have a short demonstration

Thank You