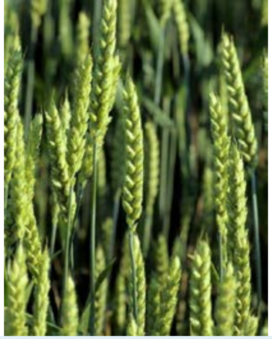
Australian Government

Bureau of Meteorology


Performance, accuracy and bit-reproducibility aspects in handling transcendental functions with Cray and Intel compilers for the Met Office Unified Model

Dr. Ilia Bermous, the Australian Bureau of Meteorology

Acknowledgements to

Dr. Martyn Corden (Intel), Dr. Zhang Zhang (Intel), Dr. Martin Dix (CSIRO)

Some requirements for meteorological applications

- **Performance**
  - **vectorisation**
  - **human factor: time spent to achieve a certain level in the performance gain**
- **Accuracy**
  - **single & double precision in weather forecasting models**
- **Results bit-reproducibility**

Note: all above mentioned aspects are applied for handling transcendental functions in an application and will be discussed in the presentation

Vector processing: Fortran compiler options related to the architecture

- **Intel compiler:** -x*code* for specific processor architecture
  - AVX – Intel Xeon processor E3/E5 and E3/E5/E7 v2 family
  - CORE-AVX2 - Intel Xeon processor E3/E5/E7 v3, v4 family
  - CORE-AVX512 – Intel Xeon Processor Scalable Family
  - Host – architecture on which the code is compiled

- **Cray compiler:** -h cpu=*target_system*

Compiler options set via modules on our XC-40 systems are

Module	Cray compiler	Intel compiler
craype-sandybridge	-hcpu=sandybridge	-mavx
craype-haswell	-hcpu=haswell	-xCORE-AVX2
craype-broadwell	-hcpu=broadwell	-xCORE-AVX2

Results bit-reproducibility

- Means getting identical results on a job rerun
- Benefits: useful in testing, debugging, tracking down code bugs or numerical instability
- Numerical implementation of UM includes a setting which provides results bit-reproducibility using different number of threads for the same horizontal decomposition

Rerun bit-reproducibility

- Intel Fortran compiler => need to specify

  `-fp-model precise`

  - may effect vectorisation of loops for some transcendental functions called within the loops

Note: below

- any reference to a compiler means Fortran compiler
- `-fp-model precise` is used in the Intel compiler case

- Cray compiler : results bit-reproducibility may depend on
  `-h flex_mp=`*`level`*

Australian Government
Bureau of Meteorology

Accuracy in calculation of transcendental functions

➢ **Accuracy is measured in terms of ULP**
(unit in the last place or unit of least precision)

- **accuracy for Cray CCE math libraries is controlled by `–hfpN` option**

Intel compiler		Cray compiler		
`default`	`-fp-model precise`	`-hfp0|-hfp1|-hfp2`	`-hfp3`	`-hfp4`
4 ULP	0.6 ULP	1.3 ULP	2.6 ULP	4 ULP

Note: `–hfp4` is used to compile the majority of UM files

Australian Government
Bureau of Meteorology

Vectorisation information for some transcendental functions in loops

With the Cray compiler and the Intel compiler with -fp-model precise, loops are

- **vectorised with SQRT, SIN, COS, EXP**
- **NOT vectorised with ASIN, ACOS, ATAN**

	Power (x^y)	ALOG	ALOG10	TANH	ATAN2
Cray compiler	Yes	Yes	No	No	Yes
Intel compiler	No	No	No	Yes	No

Australian Government
Bureau of Meteorology

Can loops for all transcendental functions be vectorised?

- Cray compiler: NO (as far as I know)
- Intel compiler: YES in two ways
  - `-fast-transcendentals -fimf-precision=<value>`

    `<value> = high | medium | low`

    high - 1 ULP, used in this presentation

    medium - 4 ULP, also tested with very similar performance

    low – not used

  - Intel Math Kernel Library (MKL) library using vector math library (the error usually is <0.75 ULP with high accuracy)

Australian Government
Bureau of Meteorology

Environment for implementation

- **Hardware**
  - Cray XC-40 (Terra), Intel Xeon E5-2690 v3, 2.6 GHz
    136 nodes with 24 cores/node (3264 cores)

- **Software**
  - Intel compiler v17.0.1.132
  - Cray cce 8.4.5
  - Cray MPICH 7.3.2
  - MPI_WTIME – for time measurements

Australian Government
Bureau of Meteorology

UM job description

- UM10.7 sources (Feb 2017)
- N768L70 (1532x1152x70 grid size) global model
- 24 hour (192 time steps) forecast with switched off I/O to tune computational capability of the model sources
- run configuration: 16x30 - horizontal decomposition with 2 threads on 960 cores
  - elapsed time for the job is 920-950 sec

Note: the job was used within the OpenMP coverage improvement collaboration project with the Met Office for UM10.4-10.8 releases

Australian Government
Bureau of Meteorology

Example: departure_point_eta_mod.F90

Subroutine includes 8 three-level nested loops:

```
DO k = 1, model_levels
  DO j = udims%j_start, udims%j_end
    DO i = udims%i_start, udims%i_end
. . .
      z_d = SQRT(1.0 - x_d**2 - y_d**2)

      depart_xi1(i,j,k) = xi1_u(i) +                               &
                          ATAN2(x_d,z_d*csxi2_p(j)-y_d*snxi2_p(j))
      depart_xi2(i,j,k) = ASIN(y_d*csxi2_p(j)+z_d*snxi2_p(j))
      depart_xi3(i,j,k) = eta_rho_levels(k) - timestep*w_a
    END DO
  END DO
END DO
```

Cray compiler loopmark output for the loop

Australian Government
Bureau of Meteorology

Loopmark output with `-rm` compiler option

```
251.  + M m----------<     DO k = 1, model_levels
252.  + M m 3--------<       DO j = udims%j_start, udims%j_end
253.    M m 3 Vp-----<         DO i = udims%i_start, udims%i_end
. . .
276.    M m 3 Vp                 z_d = SQRT(1.0 - x_d**2 - y_d**2)
277.    M m 3 Vp
278.    M m 3 Vp                 depart_xi1(i,j,k) = xi1_u(i) +                     &
279.    M m 3 Vp                                     ATAN2(x_d,z_d*csxi2_p(j)-y_d*snxi2_p(j))
280.    M m 3 Vp                 depart_xi2(i,j,k) = ASIN(y_d*csxi2_p(j)+z_d*snxi2_p(j))
281.    M m 3 Vp                 depart_xi3(i,j,k) = eta_rho_levels(k) - timestep*w_a
282.    M m 3 Vp----->         END DO
283.    M m 3-------->       END DO
284.    M m---------->     END DO

 A loop starting at line 253 was partially vectorized.
```

Comments:

- the inner loop is only partially vectorised
- the loopmark output does not provide details on what is not vectorised

Intel compiler diagnostic for the loop

```
-fp-model precise -qopt-report=5 -qopt-report-phase=vec
```

LOOP BEGIN at <path>/departure_point_eta_mod.F90(253,9)
remark #15382: vectorization support: call to function atan2 cannot be vectorized [<path>/departure_point_eta_mod.F90(279,31)]
remark #15382: vectorization support: call to function asin cannot be vectorized [<path>/departure_point_eta_mod.F90(280,31)]
remark #15344: loop was not vectorized: vector dependence prevents vectorization
LOOP END

```
-fp-model precise -fast-transcendentals -fimf-precision=high
   -qopt-report=5 -qopt-report-phase=vec
```

LOOP BEGIN at <path>/departure_point_eta_mod.F90(253,9)
. . .
remark #15305: vectorization support: vector length 4
remark #15309: vectorization support: normalized vectorization overhead 0.303
remark #15300: LOOP WAS VECTORIZED
remark #15476: scalar cost: 413
remark #15477: vector cost: 107.250
remark #15478: estimated potential speedup: 3.690
remark #15482: vectorized math library calls: 2
LOOP END

Australian Government
Bureau of Meteorology

Time measurements for departure_point_eta_mod loops

$$T = \left(\sum_{i=0}^{479} t_i\right) \Big/ 480$$ – average time taken by MPI process

t_i – total time for all 8 loops taken by MPI process with rank #*i* (the job is run on 480 MPI processes)

Compilation options for departure_point_eta_mod.F90

Cray compiler:

```
-hcpu=haswell -O3 -hvector3 -hscalar3
-hfp4 -hcache3 -haggress -hnocontiguous
–hconcurrent -hflex_mp=default
```

Australian Government
Bureau of Meteorology

Time measurements for departure_point_eta_mod loops (cont #2)

Intel compiler

STANDARD

`-align array64byte -qopenmp -O3 -fp-model precise -xavx` (1)

FAST_TRANS

`(1) "+" -fast-transcendentals -fimf-precision=high` (2)

FAST_TRANS+AVX2

`(2) with replacement –xavx => -xCORE-AVX2` (3)

Australian Government
Bureau of Meteorology

Time measurements for departure_point_eta_mod loops (cont #3)

Cray	Intel		
	STANDARD	FAST_TRANS	FAST_TRANS+AVX2
49.7 sec	**63.1 sec**	**29.8 sec**	**28.4 sec**

Conclusions:

- **usage of `-fast-transcendentals -fimf-precision=high` reduces the elapsed time by over 2 times**
- **the shortest elapsed time with the Intel compiler is 1.75 times better than with the Cray compiler**
- **minor speed up from AVX2**

Australian Government
Bureau of Meteorology

Usage of MKL in UM

Frequently called transcendental functions in the UM sources are managed by the following procedure:

```
SUBROUTINE exp_v(n,x,y)
. . .
#if defined(MKL)
CALL vdexp(n, x, y)
#else
DO i=1, n
  y(i) = EXP(x(i))
END DO
#endif
END SUBROUTINE exp_v
```

Similar routines are available in UM for

EXP, SQRT, LOG, SIN, COS, ASIN, ACOS, . . .

Performance results with Intel compiler

Build #1: STANDARD optimised build

Build #2: FAST_TRANS build for all model files

Build #3: FAST_TRANS build for all model files "+" pre-processor options are set to use the MKL library (-DMKL)

STANDARD	FAST_TRANS	FAST_TRANS + MKL
918 sec	874 sec	877 sec

Conclusions:

- **fast-transcendentals provide a reduction in the elapsed time for the job by ~5% with a relatively minor effort**
- **MKL has no additional benefit**

Performance results comparison for Intel and Cray compilers

Using the same run environment

Intel compiler => 874 sec

Cray compiler => 952 sec

with Cray compiler options set at the Met Office with enabled inter-procedural optimisation

Conclusion:

- **elapsed time with the Intel compiler is 9% better than with the Cray compiler**

Australian Government
Bureau of Meteorology

Conclusions

- Intel compiler
  - using fast-transcendentals and MKL library
    - vectorisation of some loops is improved
    - relatively good performance gains can be achieved
      - factor of 2 speed-up for SL departure point calculations loops
      - 5% improvement for full model
    - relatively minor effort required

- Cray compiler
  - at this stage it is not clear on whether a similar performance benefits are achievable with the compiler

Australian Government

Bureau of Meteorology

THANK YOU