Improving performance of numeric weather prediction codes

Roger Philp

Intel HPC Software Workshop Series 2016
HPC Code Modernization for Intel® Xeon and Xeon Phi™
February 18th 2016, Barcelona
Agenda

- Background
- Compilation
- Optimizations
- Comparisons
- Summary
Volume 2 includes the following chapters:

Foreword by Dan Stanzione, TACC
Chapter 1: Introduction
Chapter 2: Numerical Weather Prediction Optimization
Chapter 3: WRF Goddard Microphysics Scheme Optimization
Chapter 4: Pairwise DNA Sequence Alignment Optimization
Chapter 5: Accelerated Structural Bioinformatics for Drug Discovery
Chapter 6: Amber PME Molecular Dynamics Optimization
Chapter 7: Low Latency Solutions for Financial Services
Chapter 8: Parallel Numerical Methods in Finance
Chapter 9: Wilson Dslash Kernel From Lattice QCD Optimization
Chapter 10: Cosmic Microwave Background Analysis: Nested Parallelism In Practice
Chapter 11: Visual Search Optimization
Chapter 12: Radio Frequency Ray Tracing
Chapter 13: Exploring Use of the Reserved Core
Chapter 14: High Performance Python Offloading
Chapter 15: Fast Matrix Computations on Asynchronous Streams
Chapter 16: MPI-3 Shared Memory Programming Introduction
Chapter 17: Coarse-Grain OpenMP for Scalable Hybrid Parallelism
Chapter 18: Exploiting Multilevel Parallelism with OpenMP
Chapter 19: OpenCL: There and Back Again
Chapter 20: OpenMP vs. OpenCL: Difference in Performance?
Chapter 21: Prefetch Tuning Optimizations
Chapter 22: SIMD functions via OpenMP
Chapter 23: Vectorization Advice
Chapter 24: Portable Explicit Vectorization Intrinsics
Chapter 25: Power Analysis for Applications and Data Centers

We saw example after example get performance and performance portability with “just parallel programming”

Summarize as “inspired by 61 cores”

http://lotsofcores.com
Numerical Weather Prediction Optimization

Volume 2, Chapter 2

Tom Henderson, John Michalakes, Indraneil Gokhale, Ashish Jha

Increases in NWP forecast skill score over time, which are well correlated with increases in computing power.
The Nonhydrostatic Icosahedral Model (NIM) uses Icosahedral grid as shown. Except for 12 pentagons all cells are hexagons. Compared to other commonly used discretizations of the sphere, this grid minimizes variation in grid cell spacing across the globe allowing time step to be maximized.
Nomenclature

NWP – Numerical weather prediction

Split into:

• A dynamical core concerned with simulating large scale flow in the atmosphere

• Physical parametrizations approximate sub grid scale processes.

• Microphysics scheme is an important and costly physical parametrization that simulated process in the atmosphere that cause precipitation of rain, snow, graupel (also called soft hail, snow pellets)

WRF – Weather Research and Forecast

WSM6 – WRF Single Moment 6-class Microphysics scheme

Note the current model has high resolution of grid spacing sub 10km.
NIM dynamics

- Single source code
- Performs well on: AMD cpus, Nvidia GPUs, Intel mic
- Scaling from 10,000 cores to 240,000 core cores 96% efficiency
- Wsm6 most expensive physical parametrization in NIM consuming 18% of total run time
- WSM6 is logical first target
Note: Get out of jail free card!

Please note that times quoted in the chapter cannot be replicated using this simplified sample code. The chapter times measure execution of 72 time steps of a complete NWP model that includes a full dynamical core and suite of physics packages. This sample code contains only the WSM6 physics package and executes it only once.

However, it is possible to use this sample code to explore code modifications described in the chapter and to roughly reproduce relative performance differences, with caveats noted below.

Ref: ‘Readme’
Test hardware

Xeon head nodes

===== Processor composition =====
Processor name : Intel(R) Xeon(R) E5-2620 v2
Packages (sockets) : 2
Cores : 12
Processors (CPUs) : 24
Cores per package : 6
Threads per core : 2

XeonPhi mic cards

KNC 1.1GHz
Cores : 57
Logical processors : 228
Wsm6 compilation

/src

```
drwxr-xr-x  6 xx xx  47 Sep 26 23:04 gpt1
drwxr-xr-x  2 xx xx  2 Sep 26 23:04 gpt1_install
drwxr-xr-x  2 xx xx  7 Oct  5 17:53 kernel
drwxr-xr-x  2 xx xx  2 Sep 26 23:04 lib
-rw-r--r--  1 xx xx  328 Sep 26 23:04 macros.make.all
-rw-r--r--  1 xx xx  1883 Oct 12 14:29 macros.make.intelxeon
-rw-r--r--  1 xx xx  2035 Oct 12 15:28 macros.make.intelxeonphi
-rw-r--r--  1 xx xx  426 Sep 26 23:04 Makefile
-rwxr--r--  1 xx xx  17684 Oct 12 14:37 makewsm6
```

/src/kernel

```
-rw-r--r--  1 xx xx  573 Sep 26 23:04 Makefile
-rw-r--r--  1 xx xx 143630 Oct  5 17:53 wsm6_kernel.F90
```
PHYSFLAGS = -g -O3 -ftz -traceback -vec-report6 -I \ 
../include $(PRECISEFLAG) $(AGGRESSIVE)

# PHYSFLAGS = -g -O3 -ftz -traceback -vec-report6 \ 
# -align array64byte -I ../include $(PRECISEFLAG) \ 
# $(AGGRESSIVE)

# Flag to target native mode on Xeon-Phi
# empty for everyone except xeonphi
MICFLAGS = -mmic
Build process

- wsm6_kernel.F90 → cpp → wsm6_kernel.f90 → chunkfilter.pl
- wsm6_kernel_kcmd.f90 → literalkfilter.pl → wsm6_kernel_chunk.f90
- ifort → gptl library → wsm6_kernel.o
- wsm6kernel → runscript
- runintelxeonphi
  threads=228
Compilation:

${WSM6ROOT}/src/

${WSM6ROOT}/src/makewsm6 arch=intelxeonphi threading=yes translatei=no iinside=no fpmp=no chunk=8 nz=32 &> out

This creates:

${WSM6ROOT}/src_intelxeonphi_ompyes_CHUNK8_NZ32_XLATEIno_FPMPno
${WSM6ROOT}/run_intelxeonphi_ompyes_CHUNK8_NZ32_XLATEIno_FPMPno

cd ${WSM6ROOT}/run_intelxeonphi_ompyes_CHUNK8_NZ32_XLATEIno_FPMPno
chmod u+x runintelxeonphi

runintelxeonphi threads=228

cd RUN_real32level_228threads_193162
./runscript

Note for the Xeonphi need to ssh to appropriate directory
runscript

ulimit -s unlimited
....

# EDIT HERE: fix LD_LIBRARY_PATH if needed
export LD_LIBRARY_PATH=/opt/intel/composer_xe_2013.4.183/compiler/lib
/mic:$LD_LIBRARY_PATH

# Set OMP scheduling metric (default is static)
export OMP_SCHEDULE=guided
export OMP_NUM_THREADS=228
export KMP_AFFINITY="granularity=fine,scatter"

# magic stack size settings
export OMP_STACKSIZE=256M
export MIC_ULIMIT_STACKSIZE=365536
export MIC_TIMINGDEVICE=tsc
export ULIMIT_STACKSIZE=365536
export TIMINGDEVICE=tsc

./wsm6kernel > stdout
## Build options of interest

### `makewsm6` options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>arch=intelxeon</code></td>
<td>Build for Xeon: src/macros.make.intelxeon.</td>
</tr>
<tr>
<td><code>arch=intelxeonphi</code></td>
<td>Build for KNC: src/macros.make.intelxeonphi.</td>
</tr>
<tr>
<td>`threading=no</td>
<td>yes`</td>
</tr>
</tbody>
</table>

### Compile time constants:

- `chunk=4|8` "i" dimension compile-time constants
- `nz=32` "k" dimension compile-time constants

- `translatei=no` Do not translate "i" loop bounds and memory bounds to compile-time constants prior to compilation. By default this translation is done whenever threading=yes.

- `iinside=yes` Swap "k-inside" code to "i-inside" code
- `fpmp=no|yes` Turn on|off "-fp-model precise".
Optimizations

General build options:

```
PHYSFLAGS = -g -O3 -ftz -traceback -I ../include -opt-report-phase=hlo -vec-report6 -align array64byte -xHost $(PRECISEFLAG)
```

- Compile time constants
- Array alignment
- Loop restructuring
- openmp
- Vectorization
Alignment
Build options

Basic build options with –O3 without alignment:
PHYSFLAGS = -g -O3 -ftz -traceback -I ../include -opt-report-phase=hlo -vec-report6 -xHost $(PRECISEFLAG)

Basic build options with -O3 and alignment:
PHYSFLAGS = -g -O3 -ftz -traceback -I ../include -opt-report-phase=hlo -vec-report6 -align array64byte -xHost $(PRECISEFLAG)

<table>
<thead>
<tr>
<th></th>
<th>opts removed (s)</th>
<th>Opts (s) added</th>
</tr>
</thead>
<tbody>
<tr>
<td>xeon</td>
<td>2.498</td>
<td>1.81</td>
</tr>
<tr>
<td>xeonphi</td>
<td>13.895</td>
<td>12.459</td>
</tr>
</tbody>
</table>
Compiler alignment directives

```fortran
# ifdef ALIGN_OK
!DIR$ ASSUMEAligned qrs:64,den:64,denfac:64,rslope:64,rslopec:64,rslope2:64,rslope3:64,vt:64
!DIR$ VECTOR ALIGNED
#endif

  do k = kts, kte
     do i = its, ite
        if (lmask(i)) then
          if(qrs(i,k) .le. qcrmin) then
            rslope(i,k) = rslopermax
            rslopec(i,k) = rsloperbmax
            rslope2(i,k) = rsloper2max
            rslope3(i,k) = rsloper3max
          else
            rslope(i,k) = 1. / lamdar(qrs(i,k),den(i,k))
            rslopec(i,k) = rslope(i,k)**bvtr
            rslope2(i,k) = rslope(i,k)*rslope(i,k)
            rslope3(i,k) = rslope2(i,k)*rslope(i,k)
          endif
          vt(i,k) = pvtr*rslopec(i,k)*denfac(i,k)
        endif
     enddo
  enddo

END subroutine slope_rain_ii
```

19
Vectorization
Begin optimization report for: WSM62D

Report from: Vector optimizations [vec]
LOOP BEGIN at wsm6_kernel_kcmd.f90(291,7)
remark #15542: loop was not vectorized:
    inner loop was already vectorized

LOOP BEGIN at wsm6_kernel_kcmd.f90(292,9)
<Peeled>
remark #15389: vectorization support: ... qci has unaligned access   [wsm6_kernel_kcmd.f90(293,11) ]
           ...........
remark #15381: vectorization support: unaligned access used inside loop body
remark #15301: PEEL LOOP WAS VECTORIZED
LOOP END

.....
Code snippet for above vectorization report

```fortran
! padding 0 for negative values generated by dynamics
!
!
  do k = kts, kte
    do i = its, ite
      qci(i,k,1) = max(qci(i,k,1), 0.0)
      qrs(i,k,1) = max(qrs(i,k,1), 0.0)
      qci(i,k,2) = max(qci(i,k,2), 0.0)
      qrs(i,k,2) = max(qrs(i,k,2), 0.0)
      qrs(i,k,3) = max(qrs(i,k,3), 0.0)
    enddo
  enddo
```
Compile time constants
Build process

wsm6_kernel.F90 → cpp → wsm6_kernel.f90 → chunkfilter.pl

wsm6_kernel_kcmd.f90 → literalkfilter.pl → wsm6_kernel_chunk.f90

ifort → gptl library → wsm6_kernel.o

wsm6kernel

runscript

Runintelxeonphi
threads=224
#!/usr/bin/perl -p
# insert compile-time chunk sizes
s/\\b(do\\s+\\d*\\s*\\w+\\s*=\\s*)ii*ts\\s*,\\s*ii*te\\b/$11,8/gi; s/\\bii*[mt]s\\s*:\\s*ii*[mt]e\\b/1:8/gi;
Baseline: compile time constants

```plaintext
#ifdef __CHUNK__
  CHUNK = __CHUNK__
#else
  CHUNK = iite-iits+1
#endif

#ifdef __NZ__
  NZ_BUILD = __NZ__
  ! if specified, NZ_BUILD must match namelist setting for nvl
  if (NZ_BUILD/=kte-kts+1) then
    print *, 'ERROR: Build-time-specified NZ must equal namelist nz, values are: ',NZ_BUILD,kte-kts+1
    call flush(6)
    stop
  endif
#else
  NZ_BUILD = kte-kts+1
#endif
```
openmp
Adding openmp: firstTouch and wsm62D

!$OMP PARALLEL DO &
!$OMP PRIVATE ( j ) &
!$OMP SCHEDULE(runtime)
  do j = jjts,jjte
    CALL firstTouch(t(iits,kts,j), q(iims,kms,j)
  enddo ! j loop
!$OMP END PARALLEL DO

!$OMP PARALLEL DO &
!$OMP PRIVATE ( j,ret ) &
!$OMP SCHEDULE(runtime)
  do j = jjts,jjte
    CALL wsm62D(t(iits,kts,j), q(iims,kms,j)
  enddo ! j loop
!$OMP END PARALLEL DO
Xeon speedup chunk=4 nz=32

<table>
<thead>
<tr>
<th>#threads</th>
<th>Without alignment</th>
<th>With alignment</th>
<th>Speedup</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compact</td>
<td>scatter</td>
<td>compact</td>
<td>scatter</td>
</tr>
<tr>
<td>1</td>
<td>2.064</td>
<td>2.071</td>
<td>1.889</td>
<td>1.852</td>
</tr>
<tr>
<td>6</td>
<td>0.481</td>
<td>0.323</td>
<td>0.437</td>
<td>0.294</td>
</tr>
<tr>
<td>12</td>
<td>0.241</td>
<td>0.163</td>
<td>0.221</td>
<td>0.171</td>
</tr>
<tr>
<td>18</td>
<td>0.167</td>
<td>0.163</td>
<td>0.156</td>
<td>0.15</td>
</tr>
<tr>
<td>24</td>
<td>0.124</td>
<td>0.12</td>
<td>0.155</td>
<td>0.125</td>
</tr>
</tbody>
</table>

Xeon Speeedup vs number of threads

Without explicit alignment

With explicit alignment
XeonPhi speedup chunk=8 nz=32

<table>
<thead>
<tr>
<th>XeonPhi #threads</th>
<th>Without alignment</th>
<th>With alignment</th>
<th>Speedup W/O</th>
<th>Speedup W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compact scatter</td>
<td>compact scatter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16.855</td>
<td>13.066</td>
<td>12.016</td>
<td>12.928</td>
</tr>
<tr>
<td>57</td>
<td>0.223</td>
<td>0.222</td>
<td>0.21</td>
<td>0.213</td>
</tr>
<tr>
<td>114</td>
<td>0.195</td>
<td>0.195</td>
<td>0.185</td>
<td>0.186</td>
</tr>
<tr>
<td>171</td>
<td>0.18</td>
<td>0.178</td>
<td>0.174</td>
<td>0.172</td>
</tr>
<tr>
<td>228</td>
<td>0.169</td>
<td>0.168</td>
<td>0.159</td>
<td>0.16</td>
</tr>
</tbody>
</table>

XeonPhi speedup vs number of threads
Comparison between Xeon and XeonPhi

<table>
<thead>
<tr>
<th>#threads</th>
<th>Without alignment</th>
<th>With alignment</th>
<th>Speedup</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compact scatter</td>
<td>compact scatter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.064 2.071</td>
<td>1.889 1.852</td>
<td>1</td>
<td>1.092642</td>
</tr>
<tr>
<td>6</td>
<td>0.481 0.323</td>
<td>0.437 0.294</td>
<td>4.29106</td>
<td>4.723112</td>
</tr>
<tr>
<td>12</td>
<td>0.241 0.163</td>
<td>0.221 0.171</td>
<td>8.564315</td>
<td>9.339367</td>
</tr>
<tr>
<td>18</td>
<td>0.167 0.163</td>
<td>0.156 0.15</td>
<td>12.35928</td>
<td>13.23077</td>
</tr>
<tr>
<td>24</td>
<td>0.124 0.12</td>
<td>0.155 0.125</td>
<td>16.64516</td>
<td>13.31613</td>
</tr>
</tbody>
</table>

Xeon chunk=4 nz=32

<table>
<thead>
<tr>
<th>#threads</th>
<th>Without alignment</th>
<th>With alignment</th>
<th>Speedup</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>compact scatter</td>
<td>compact scatter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16.855 13.066</td>
<td>12.016 12.928</td>
<td>1</td>
<td>1.402713</td>
</tr>
<tr>
<td>57</td>
<td>0.223 0.222</td>
<td>0.21 0.213</td>
<td>75.58296</td>
<td>80.2619</td>
</tr>
<tr>
<td>114</td>
<td>0.195 0.195</td>
<td>0.185 0.186</td>
<td>86.4359</td>
<td>91.10811</td>
</tr>
<tr>
<td>171</td>
<td>0.18 0.178</td>
<td>0.174 0.172</td>
<td>93.63889</td>
<td>96.86782</td>
</tr>
<tr>
<td>228</td>
<td>0.169 0.168</td>
<td>0.159 0.16</td>
<td>99.73373</td>
<td>106.0063</td>
</tr>
</tbody>
</table>

XeonPhi chunk=8 nz=32
Loop reordering
Loop reordering: iinside=yes

```c
#define _NISLFV_RAIN_PLM_ nislfv_rain_plm
#define _NISLFV_RAIN_PLM6_ nislfv_rain_plm6

#endif

do k=kts,kte
  do i=its,ite
    ..........  
  enddo
enddo

call _NISLFV_RAIN_PLM_(...) 
call _NISLFV_RAIN_PLM6_(...) 
```
Xeon \textit{iinside}=yes

<table>
<thead>
<tr>
<th>Xeon</th>
<th>Compact (s)</th>
<th>Scatter (s)</th>
<th>c/speedup</th>
<th>s/speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.458</td>
<td>3.246</td>
<td>0.600925</td>
<td>0.640173</td>
</tr>
<tr>
<td>6</td>
<td>0.549</td>
<td>0.543</td>
<td>3.785064</td>
<td>3.826888</td>
</tr>
<tr>
<td>12</td>
<td>0.29</td>
<td>0.2283</td>
<td>7.165517</td>
<td>9.102059</td>
</tr>
<tr>
<td>18</td>
<td>0.277</td>
<td>0.272</td>
<td>7.501805</td>
<td>7.639706</td>
</tr>
<tr>
<td>24</td>
<td>0.203</td>
<td>0.205</td>
<td>10.23645</td>
<td>10.13659</td>
</tr>
</tbody>
</table>

**Graph:**
Xeon speedup vs number of threads for \textit{iinside}=yes

- \textit{compact} blue line
- \textit{scatter} orange line
### XeonPhi $iinside=\text{yes}$

<table>
<thead>
<tr>
<th>XeonPhi</th>
<th>Compact (s)</th>
<th>Scatter (s)</th>
<th>Compact speedup</th>
<th>Scatter Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23.105</td>
<td>23.812</td>
<td>0.755075</td>
<td>0.732656</td>
</tr>
<tr>
<td>57</td>
<td>0.377</td>
<td>0.378</td>
<td>46.27586</td>
<td>46.15344</td>
</tr>
<tr>
<td>114</td>
<td>0.296</td>
<td>0.298</td>
<td>58.93919</td>
<td>58.54362</td>
</tr>
<tr>
<td>171</td>
<td>0.259</td>
<td>0.257</td>
<td>67.35907</td>
<td>67.88327</td>
</tr>
<tr>
<td>228</td>
<td>0.243</td>
<td>0.24</td>
<td>71.79424</td>
<td>72.69167</td>
</tr>
</tbody>
</table>

#### XeonPhi speedup vs number of threads

$iinside=\text{yes}$

![Graph showing speedup vs number of threads](image-url)
## Comparison of Xeon and Xeon Phi

<table>
<thead>
<tr>
<th>Xeon</th>
<th>Compact (s)</th>
<th>Scatter (s)</th>
<th>Compact speedup</th>
<th>Scatter speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.458</td>
<td>3.246</td>
<td>0.600925</td>
<td>0.640173</td>
</tr>
<tr>
<td>6</td>
<td>0.549</td>
<td>0.543</td>
<td>3.785064</td>
<td>3.826888</td>
</tr>
<tr>
<td>12</td>
<td>0.29</td>
<td>0.2283</td>
<td>7.165517</td>
<td>9.102059</td>
</tr>
<tr>
<td>18</td>
<td>0.277</td>
<td>0.272</td>
<td>7.501805</td>
<td>7.639706</td>
</tr>
<tr>
<td>24</td>
<td>0.203</td>
<td>0.205</td>
<td>10.23645</td>
<td>10.13659</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XeonPhi</th>
<th>Compact (s)</th>
<th>Scatter (s)</th>
<th>Compact speedup</th>
<th>Scatter speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23.105</td>
<td>23.812</td>
<td>0.755075</td>
<td>0.732656</td>
</tr>
<tr>
<td>57</td>
<td>0.377</td>
<td>0.378</td>
<td>46.27586</td>
<td>46.15344</td>
</tr>
<tr>
<td>114</td>
<td>0.296</td>
<td>0.298</td>
<td>58.93919</td>
<td>58.54362</td>
</tr>
<tr>
<td>171</td>
<td>0.259</td>
<td>0.257</td>
<td>67.35907</td>
<td>67.88327</td>
</tr>
<tr>
<td>228</td>
<td>0.243</td>
<td>0.24</td>
<td>71.79424</td>
<td>72.69167</td>
</tr>
</tbody>
</table>

**Xeon chunk=4 nz=32 iinside=yes**

**XeonPhi chunk=8 nz=32 iinside=yes**
## Best performances

<table>
<thead>
<tr>
<th>#threads</th>
<th>Compact N/A (s)</th>
<th>Scatter N/A (s)</th>
<th>Compact A (s)</th>
<th>Scatter A (s)</th>
<th>Speedup Compact</th>
<th>Speedup Scatter</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>0.124</td>
<td>0.12</td>
<td>0.155</td>
<td>0.125</td>
<td>16.64516</td>
<td>13.31613</td>
</tr>
</tbody>
</table>

**Xeon chunk=4 nz=32**

<table>
<thead>
<tr>
<th>#threads</th>
<th>Compact N/A (s)</th>
<th>Scatter N/A (s)</th>
<th>Compact A (s)</th>
<th>Scatter A (s)</th>
<th>Speedup Compact</th>
<th>Speedup Scatter</th>
</tr>
</thead>
<tbody>
<tr>
<td>228</td>
<td>0.169</td>
<td>0.168</td>
<td>0.159</td>
<td>0.16</td>
<td>99.73373</td>
<td>106.0063</td>
</tr>
</tbody>
</table>

**XeonPhi chunk= 8 nz=32**

<table>
<thead>
<tr>
<th>#threads</th>
<th>Compact (s)</th>
<th>Scatter (s)</th>
<th>Compact Speedup</th>
<th>Scatter Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>0.203</td>
<td>0.205</td>
<td>10.23645</td>
<td>10.13659</td>
</tr>
</tbody>
</table>

**Xeon chunk=4 nz=32 iinside=yes**

<table>
<thead>
<tr>
<th>#threads</th>
<th>Compact (s)</th>
<th>Scatter (s)</th>
<th>Compact Speedup</th>
<th>Scatter Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>228</td>
<td>0.243</td>
<td>0.24</td>
<td>71.79424</td>
<td>72.69167</td>
</tr>
</tbody>
</table>

**XeonPhi  chunk=8 nz=32 iinside=yes**
Profiling: vtune
What does vtune say about the Xeon?

**OpenMP Region CPU Usage Histogram**
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously in an OpenMP region. Spin and Overhead time adds to the Idle CPU usage value. OpenMP regions in the drop-down list are sorted by Potential Gain (Elapsed Time) so it is recommended to start exploration from the top.

**OpenMP Region Duration Histogram**
This histogram shows the total number of region instances in your application executed with a specific duration. High number of slow instances may signal a performance bottleneck. Explore the data provided in the Bottom-up, Top-down Tree, and Timeline panes to identify code regions with the slow duration.
What does vtune say about the Xeon – cont:
Source profile

<table>
<thead>
<tr>
<th>Source</th>
<th>Effective Time by Utilization</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Idle</td>
<td>Poor</td>
</tr>
<tr>
<td>INTP : do k=KTS,kte</td>
<td>8.019ms</td>
<td>5.012ms</td>
</tr>
<tr>
<td>k=k=MAX(kb-1,kts)</td>
<td>8.019ms</td>
<td>5.012ms</td>
</tr>
<tr>
<td>kt=MAX(kt-1,kts)</td>
<td>8.019ms</td>
<td>5.012ms</td>
</tr>
<tr>
<td>! find kb and kt</td>
<td>8.019ms</td>
<td>5.012ms</td>
</tr>
<tr>
<td>intp_mask = ( zi(k, k) .LT. za(k, k+1) ) AND. !mask</td>
<td>4.010ms</td>
<td>4.010ms</td>
</tr>
<tr>
<td>tmask = intp_mask</td>
<td>9.022ms</td>
<td>4.010ms</td>
</tr>
<tr>
<td>minkb = 999</td>
<td>6.014ms</td>
<td>6.014ms</td>
</tr>
<tr>
<td>minkt = 999</td>
<td>1.002ms</td>
<td>1.002ms</td>
</tr>
<tr>
<td>DO i=its,itk</td>
<td>8.019ms</td>
<td>4.010ms</td>
</tr>
<tr>
<td>IF ( tmask(i) .AND. kb(i) .LT. minkb ) minkb = kb(i)</td>
<td>2.005ms</td>
<td>2.005ms</td>
</tr>
<tr>
<td>IF ( tmask(i) .AND. kt(i) .LT. minkt ) minkt = kt(i)</td>
<td>1.002ms</td>
<td>1.002ms</td>
</tr>
<tr>
<td>find_kb : do k=KTS,kte</td>
<td>31.075ms</td>
<td>31.075ms</td>
</tr>
<tr>
<td>DO it=its,itk</td>
<td>155.374ms</td>
<td>155.374ms</td>
</tr>
<tr>
<td>IF ( tmask(i) .AND. zi(i, k) .LE. za(i, k+1) ) THEN</td>
<td>123.296ms</td>
<td>123.296ms</td>
</tr>
<tr>
<td>kb(i) = kk</td>
<td>17.041ms</td>
<td>17.041ms</td>
</tr>
<tr>
<td>tmask(i) = .FALSE.</td>
<td>1.002ms</td>
<td>1.002ms</td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>enddo find_kb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>find_kt : do k=KTKT,kte</td>
<td>23.055ms</td>
<td>23.055ms</td>
</tr>
<tr>
<td>DO it=its,itk</td>
<td>128.308ms</td>
<td>128.308ms</td>
</tr>
<tr>
<td>IF ( tmask(i) .AND. zi(i, k+1) .LE. za(i, k) ) THEN</td>
<td>99.239ms</td>
<td>99.239ms</td>
</tr>
<tr>
<td>kt(i) = kk</td>
<td>33.080ms</td>
<td>33.080ms</td>
</tr>
<tr>
<td>tmask(i) = .FALSE.</td>
<td>4.010ms</td>
<td>4.010ms</td>
</tr>
<tr>
<td>ENDIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENDDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>155.374ms</td>
<td>0ms</td>
<td>0ms</td>
</tr>
</tbody>
</table>
Internal functions

!=================================================================
!   compute internal functions
!

cpmcal(x) = cpd*(1.-max(x,qmin))+max(x,qmin)*cpv
xlcal(x) = xlv0-xlv1*(x-t0c)

!----------------------------------------------------------------
!

diffus: diffusion coefficient of the water vapor
! viscos: kinematic viscosity(m2s-1)
! Optimizatin : A**B => exp(log(A)*(B))
!
diffus(x,y) = 8.794e-5 * exp(log(x)*(1.81)) / y  ! 8.794e-5*x**1.81/y
viscos(x,y) = 1.496e-6 * (x*sqrt(x)) /(x+120.)/y  ! 1.496e-6*x**1.5/(x+120.)/y
xka(x,y) = 1.414e3*viscos(x,y)*y

diffac(a,b,c,d,e) = d*a*a/(xka(c,d)*rv*c*c)+1.)/(e*diffus(c,b))
venfac(a,b,c) = exp(log((viscos(b,c)/diffus(b,a)))*(1.33333333)) &
                 /sqrt(viscos(b,c))*sqrt(sqrt(den0/c))
conden(a,b,c,d,e) = (max(b,qmin)-c)/(1.+d*d/(rv*e)*c/(a*a))
Summary

- Complex compile scenario
- Multiple compile paths
- Openmp scalability
- MPI scalability
- Further performance tweaks
Thankyou