A study on computational complexity of optimized finite-difference weights for reverse time migration kernels

DANILO L. COSTA [1]

LUCIANO LEITE [1] / JOSIAS SILVA [1] / ALVARO COUTINHO [1]

LEONARDO BORGES [2] / ALAOR NETO [2]

[1] NACAD - HIGH PERFORMANCE COMPUTING CENTER

COPPE/FEDERAL UNIVERSITY OF RIO DE JANEIRO

WWW.NACAD.UFRJ.BR

[2] INTEL SOFTWARE & SERVICES GROUP

INTEL CORPORATION - SOFTWARE.INTEL.COM

This Study Examines

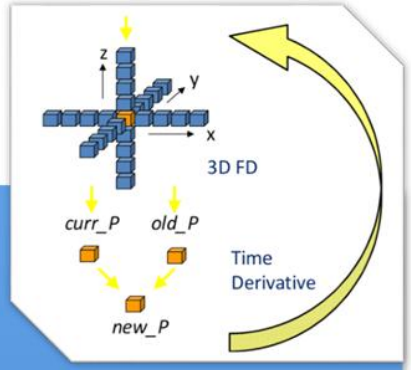
Seismic Reverse Time Migration Kernel

- 3D
- Acoustic
- Isotropic
- Full Wave Equation
- Finite-Difference Stencil
- Explicit 2th Order Time
- Implemented in Fortran

Computational/Memory Demands

- Stencil Optimizations
- Changes in Derivative Weights
- Changes in Finite-Difference Order
- Intel Xeon E5 Processor Architecture
- Intel Xeon Phi Coprocessor Architecture

RTM Stencil




```
do k , Nz
   do j , Ny
      do i , Nx
         new_P(i,j,k)  =   2 *   curr_P(i  ,j  ,k ) -  old_P(i  ,j  ,k ) +      C(I  ,j  ,k ) * ( a0  *  curr_P(i  ,j  ,k ) +
                           a1 * (  curr_P(i-1,j  ,k ) + curr_P(i+1,j  ,k ) + curr_P(i  ,j-1,k ) + curr_P(i  ,j+1,k ) + curr_P(i  ,j  ,k-1) + curr_P(i  ,j  ,k+1)  )  +
                           a2 * (  curr_P(i-2,j  ,k ) + curr_P(i+2,j  ,k ) + curr_P(i  ,j-2,k ) + curr_P(i  ,j+2,k ) + curr_P(i  ,j  ,k-2) + curr_P(i  ,j  ,k+2)  )  +
                           a3 * (  curr_P(i-3,j  ,k ) + curr_P(i+3,j  ,k ) + curr_P(i  ,j-3,k ) + curr_P(i  ,j+3,k ) + curr_P(i  ,j  ,k-3) + curr_P(i  ,j  ,k+3)  )  +
                           a4 * (  curr_P(i-4,j  ,k ) + curr_P(i+4,j  ,k ) + curr_P(i  ,j-4,k ) + curr_P(i  ,j+4,k ) + curr_P(i  ,j  ,k-4) + curr_P(i  ,j  ,k+4)  )  +
                           a5 * (  curr_P(i-5,j  ,k ) + curr_P(i+5,j  ,k ) + curr_P(i  ,j-5,k ) + curr_P(i  ,j+5,k ) + curr_P(i  ,j  ,k-5) + curr_P(i  ,j  ,k+5)  )  +
                           a6 * (  curr_P(i-6,j  ,k ) + curr_P(i+6,j  ,k ) + curr_P(i  ,j-6,k ) + curr_P(i  ,j+6,k ) + curr_P(i  ,j  ,k-6) + curr_P(i  ,j  ,k+6)  )  +
                           a7 * (  curr_P(i-7,j  ,k ) + curr_P(i+7,j  ,k ) + curr_P(i  ,j-7,k ) + curr_P(i  ,j+7,k ) + curr_P(i  ,j  ,k-7) + curr_P(i  ,j  ,k+7)  )  +
                           a8 * (  curr_P(i-8,j  ,k ) + curr_P(i+8,j  ,k ) + curr_P(i  ,j-8,k ) + curr_P(i  ,j+8,k ) + curr_P(i  ,j  ,k-8) + curr_P(i  ,j  ,k+8)  )
      enddo
   enddo
enddo
```

16th order | 3D | Acoustic | Isotropic | 61 Flops

RTM - Computational Characteristics

- RTM algorithm is classified as Memory-Bound
  - Low arithmetic intensity per data transfer
  - Limited not only by the processor's Flops
  - Memory Bandwidth is also a **bottleneck**

- Large geological areas demand large amount of computational resources
  - Disk storage (PetaBytes)
  - Physical Memory (TeraBytes)
  - Massive processing resources (hundreds of TeraFlops)
  - Large amount of processing time (thousands of hours)

3D Streamer - Typical Domain

Whole Domain

Streamer Acquisition

- 10000 Km^2 x 8 Km
- ~50 000 shots

Isotropic Equation

- ~1 TeraBytes

Shot Domain

Streamer Acquisition

- 70 Km^2 x 8 Km
- 1 shot

Isotropic Equation

- ~7 GigaBytes

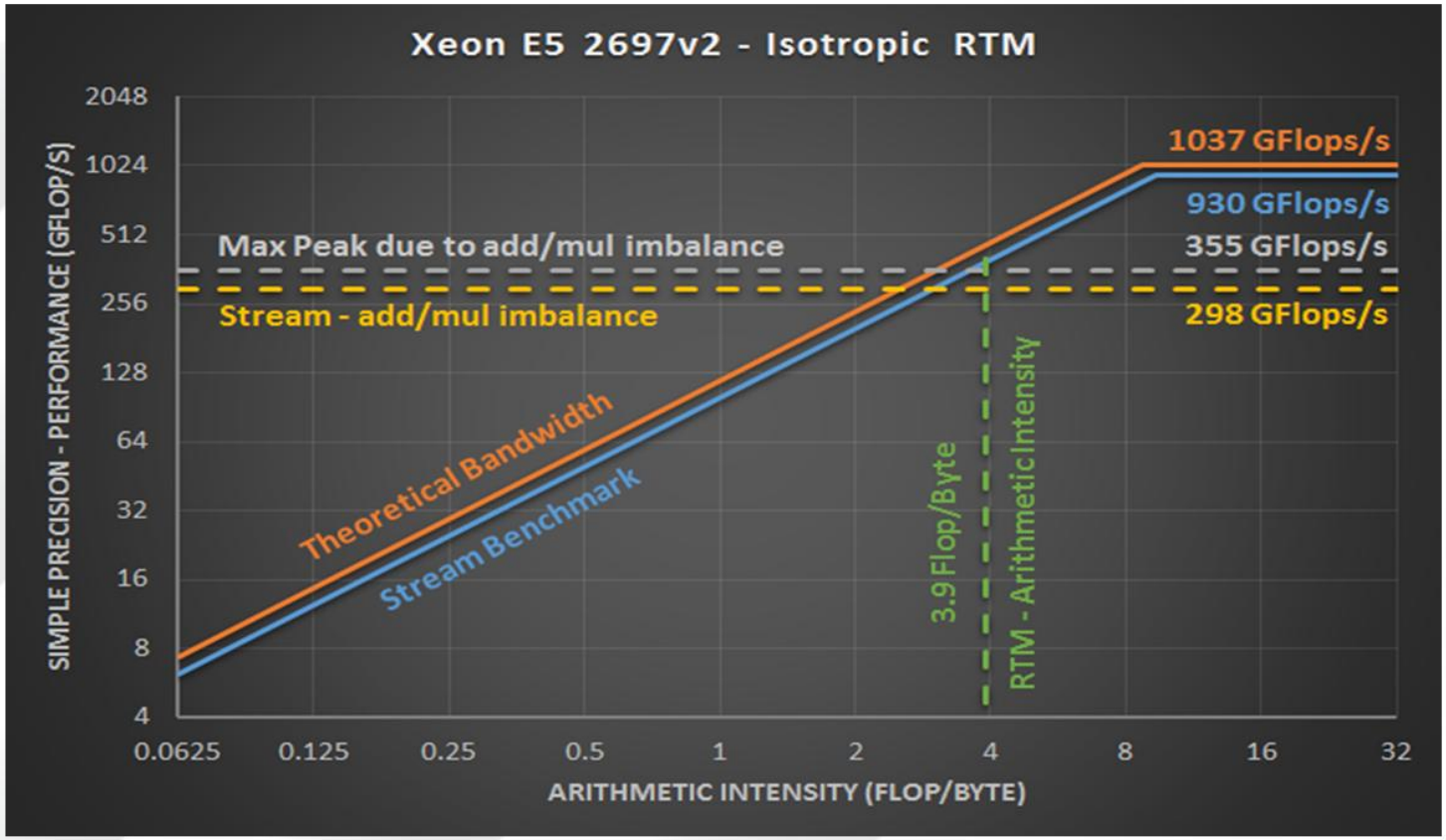
Testbed Platforms

Xeon E5 2697v2

30 MB Cache
1866 MHz DDR3
2.7 GHz Clock Speed
2 Sockets x 12 Cores
256 bits Vector
1.04 Tflop/s SP
119 GB/s Bandwidth

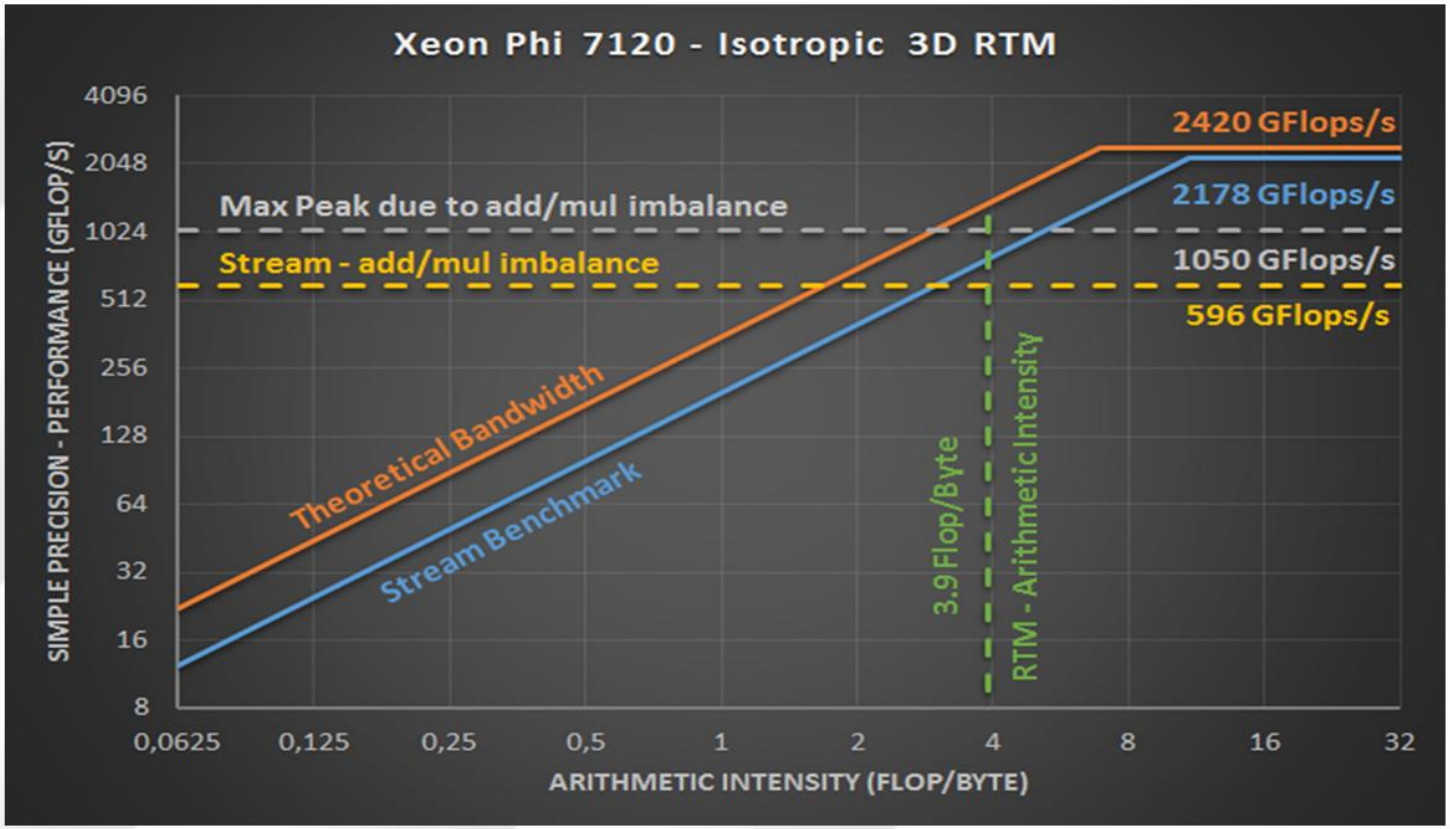
30 MB Cache
16 GB DDR5 Mem
1.2 GHz Clock Speed
61 Physical Cores
512 bits Vector
2.4 Tflop/s SP
352 GB/s Bandwidth

Xeon Phi 7120



Roofline Model – Single Precision

ANDREOLLI, C., THIERRY, P., BORGES, L., YOUNT, C., SKINNER, G., 2014, "Genetic Algorithm Auto-Tuning of Seismic Applications on Multi and Manycore Computers", EAGE Workshop on High Performance Computing for Upstream.



Roofline Model – Single Precision

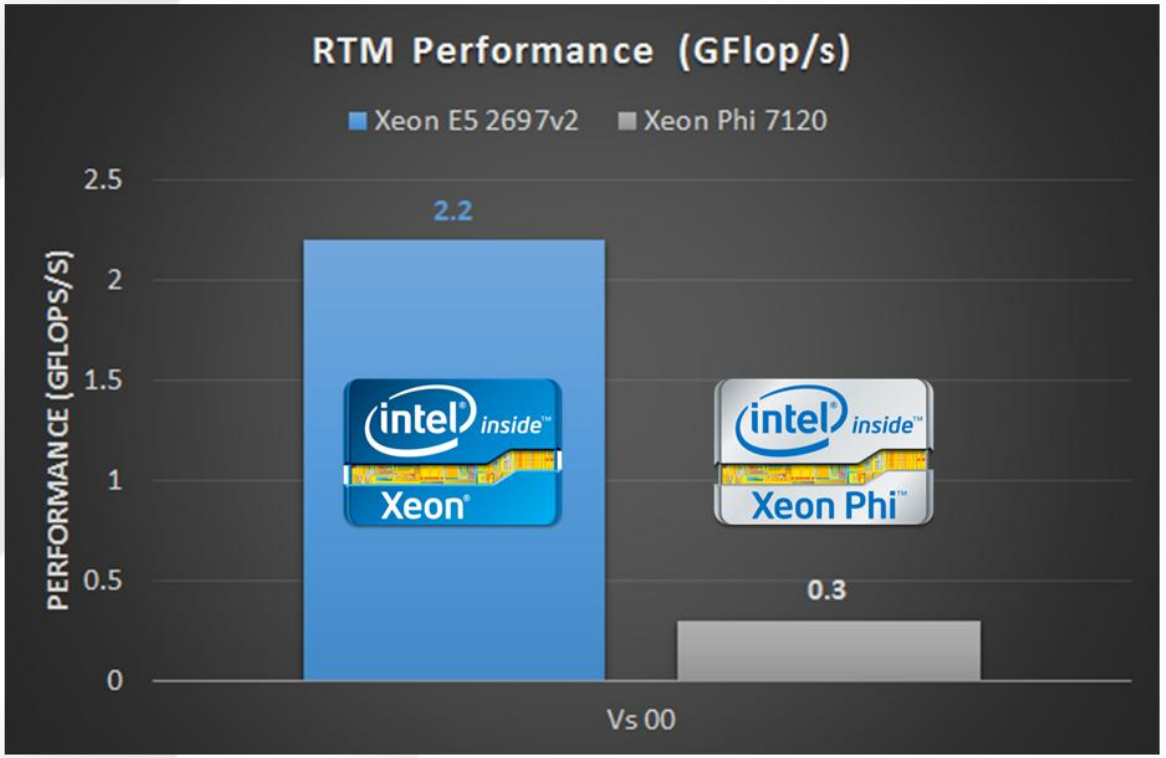
ANDREOLLI, C., THIERRY, P., BORGES, L., YOUNT, C., SKINNER, G., 2014, "Genetic Algorithm Auto-Tuning of Seismic Applications on Multi and Manycore Computers", EAGE Workshop on High Performance Computing for Upstream.

Base Code
Vs 00

Automatic compiler optimization
- -O3

No vectorization
- -no-vec

No parallelization
- -openmp-stubs



Parallelization
Vs 01

Speedup = 21x

Speedup = 43x

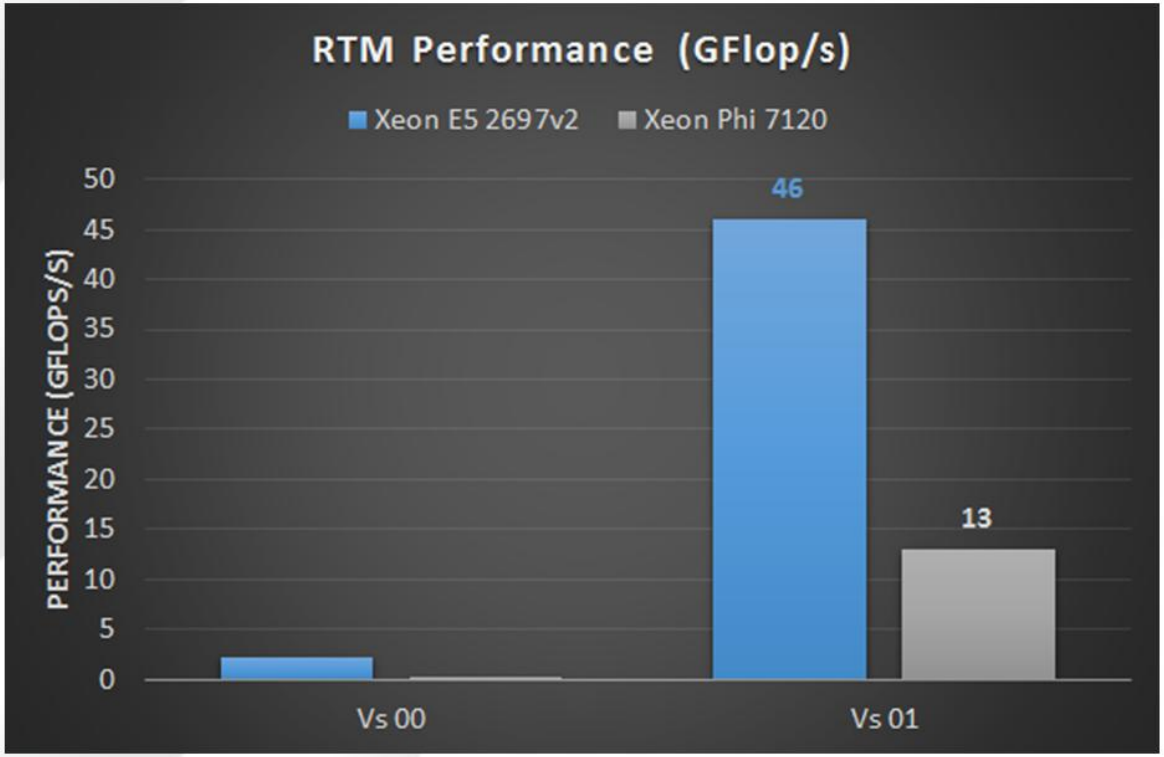
Automatic compiler optimization
- -O3

No vectorization
- -no-vec

Parallelization
- !$omp parallel do
- -openmp

omp_num_threads = 24

omp_num_threads = 244



Vectorization
Vs 02

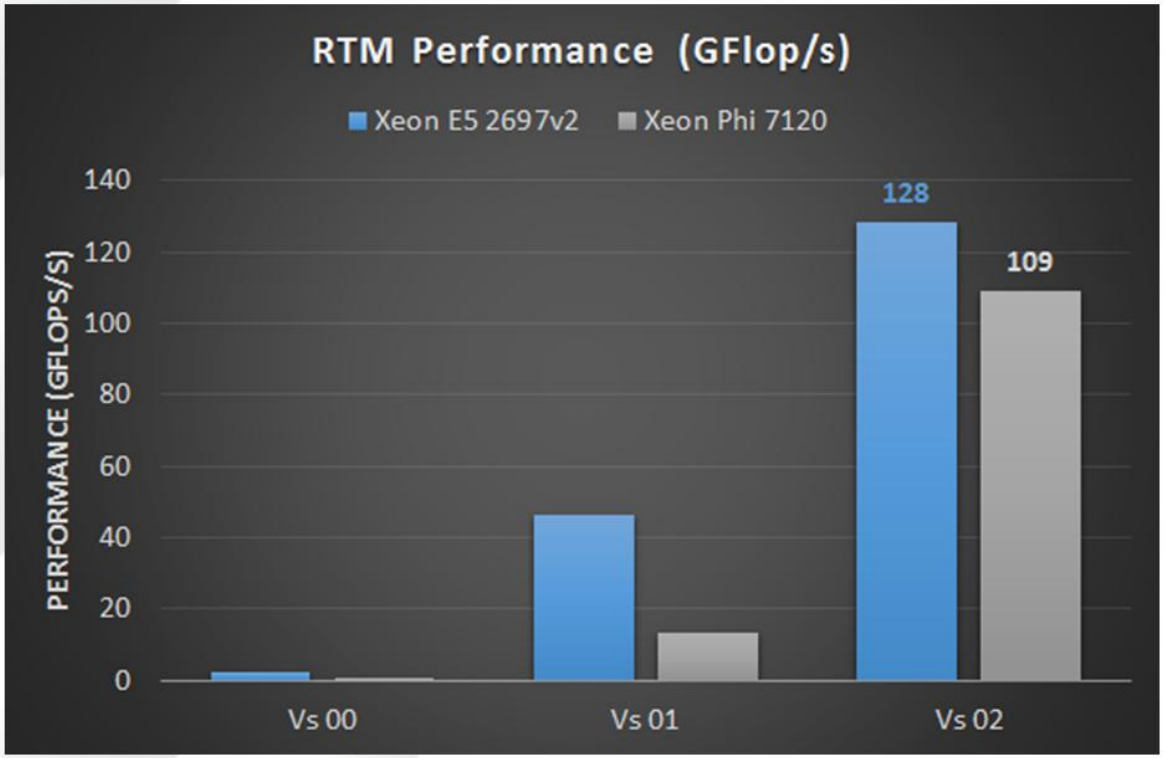
Speedup = 2.8x

Speedup = 8.4x

Automatic compiler optimization
- -O3

Vectorization
- -vec
- -xAVX

Parallelization
- !$omp parallel do
- -openmp



Memory Improvements Vs 03

Speedup = 1.07x

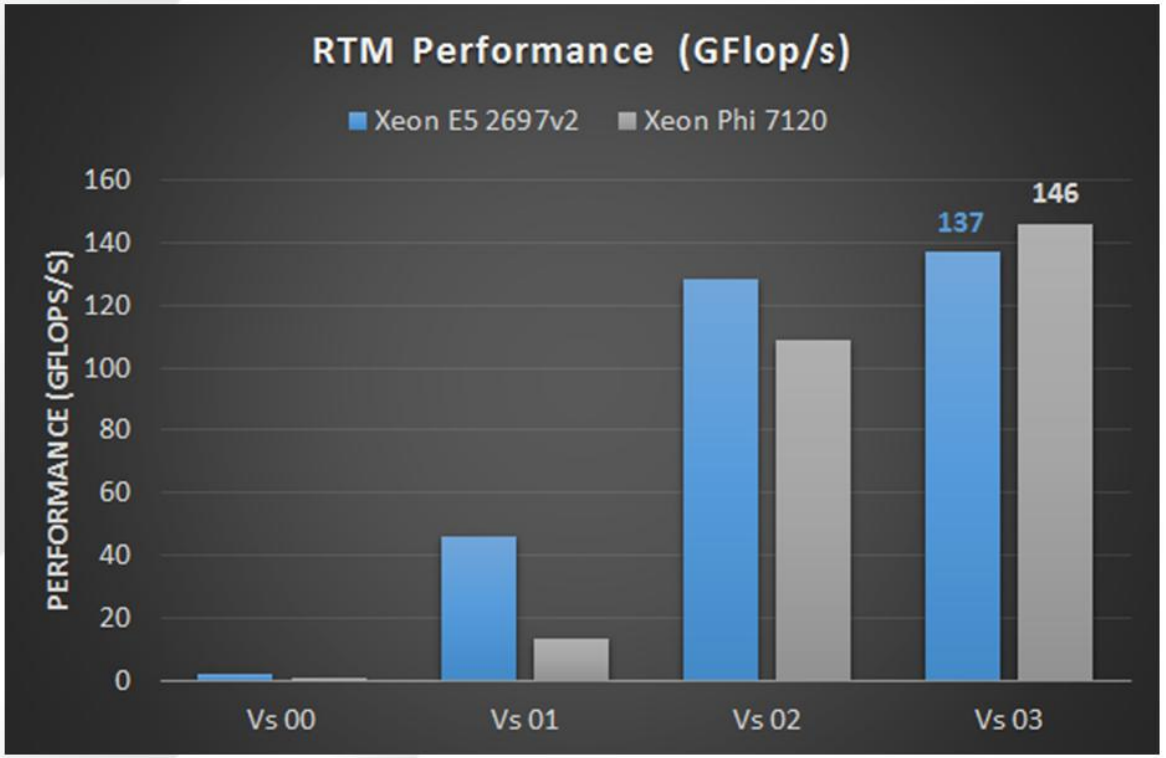
Speedup = 1.34x

Loop unrolling

- Register reuse
- Reduces memory traffic
- Cut down TLB misses

Cache blocking

- Improves data locality
- Optimal cache line use
- Reduces cache misses



Thread Affinity Vs 04

Speedup = 1.36x

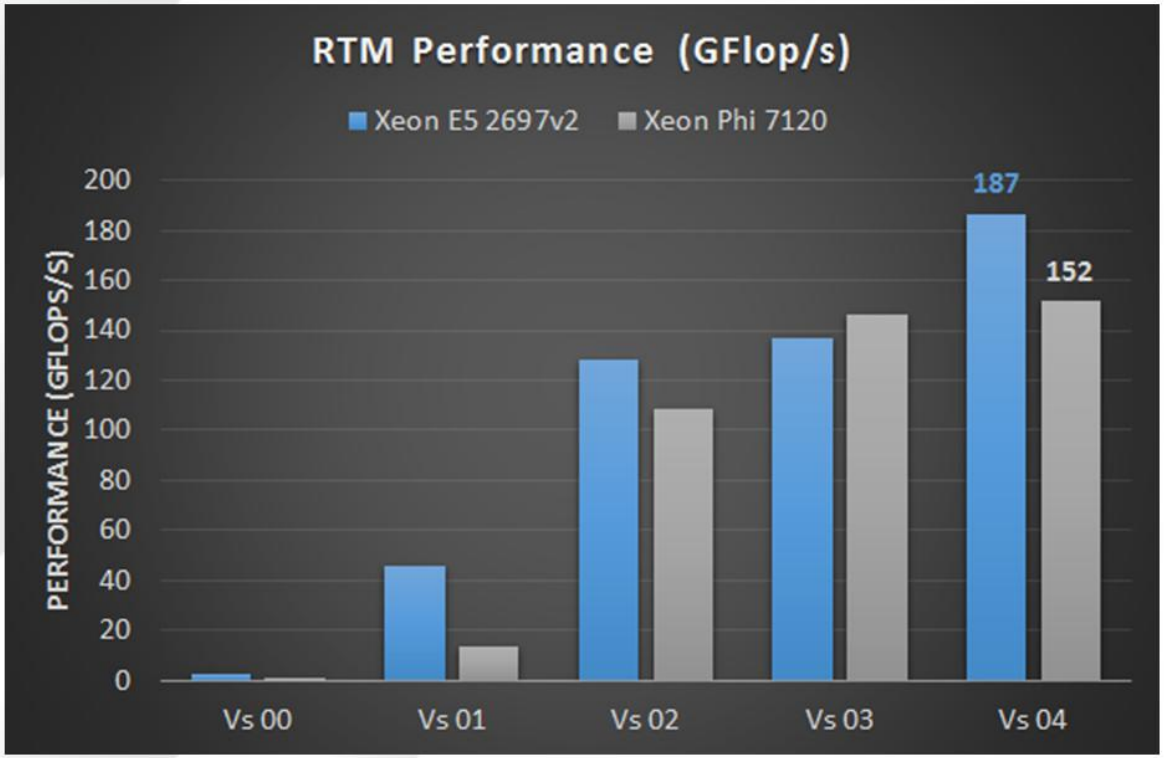
Speedup = 1.04

NUMA arch

- First Touch
- numactl --interleave
- kmp_affinity
- !$omp schedule (static)

DDR5 arch

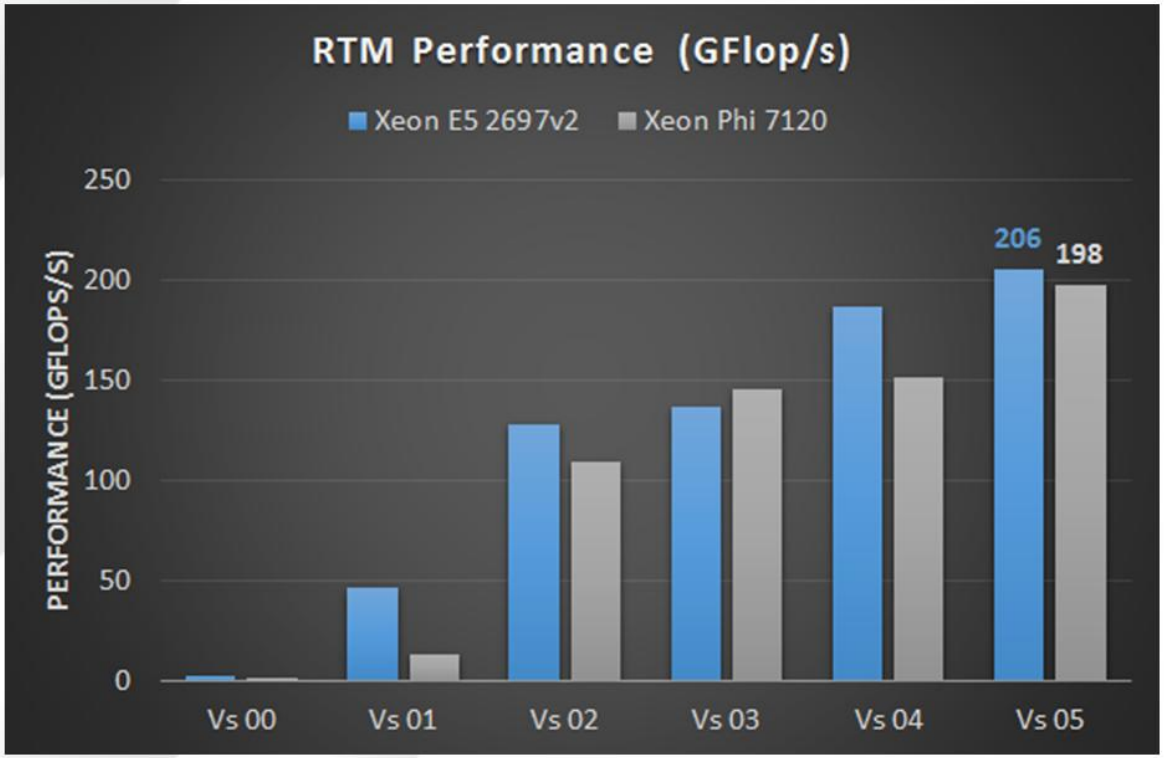
- kmp_affinity
- !$omp schedule (dynamic)



Return to Basic Vs 05

Speedup = 1.10x

Speedup = 1.30x



- Fast data access
- Avoid memory fragmentation
- Static Allocation
- Managed efficiently by CPU
- Improves previus and next optimizations

Linux: ulimit -s unlimited

Alignment and Padding Vs 06

Speedup = 1.17x

Speedup = 1.02x

64 Byte

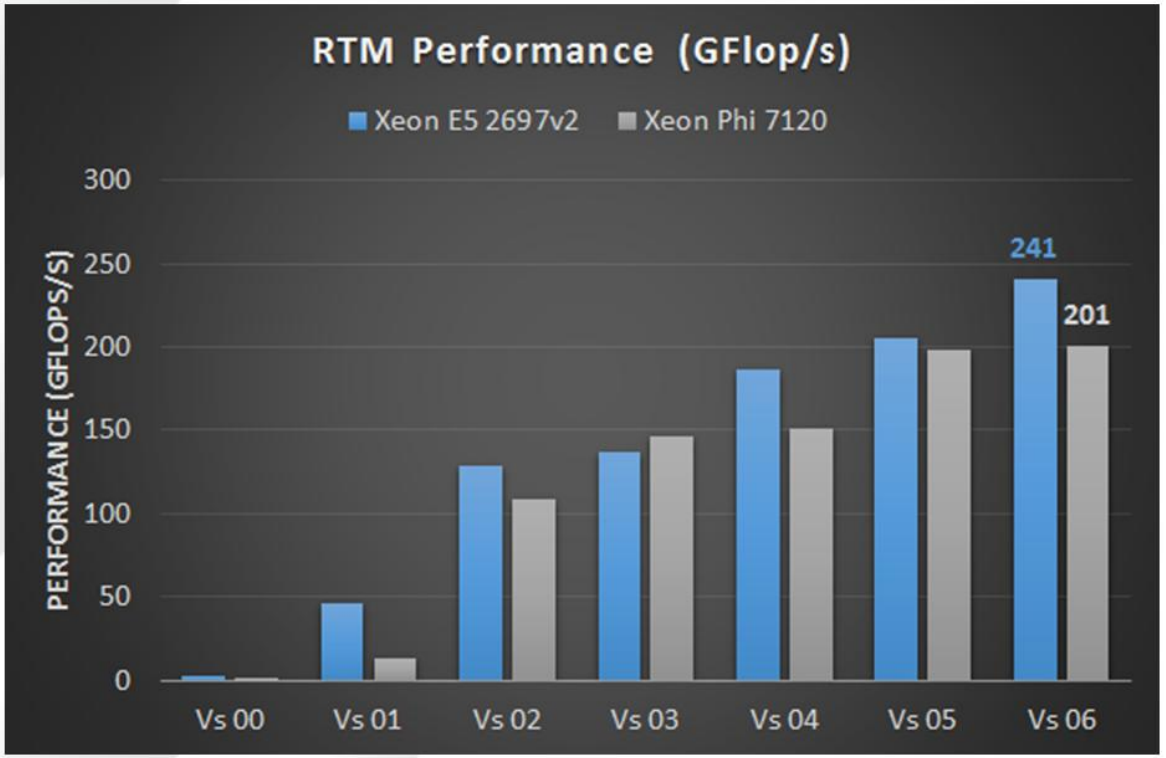
Alignment

- -align array64byte
- !dir$ assume_aligned
- !$omp simd aligned

64 Byte

Padding

- -opt-assume-safe-padding
- Add memory positions (Nx+pad , Ny+pad , Nz+pad)



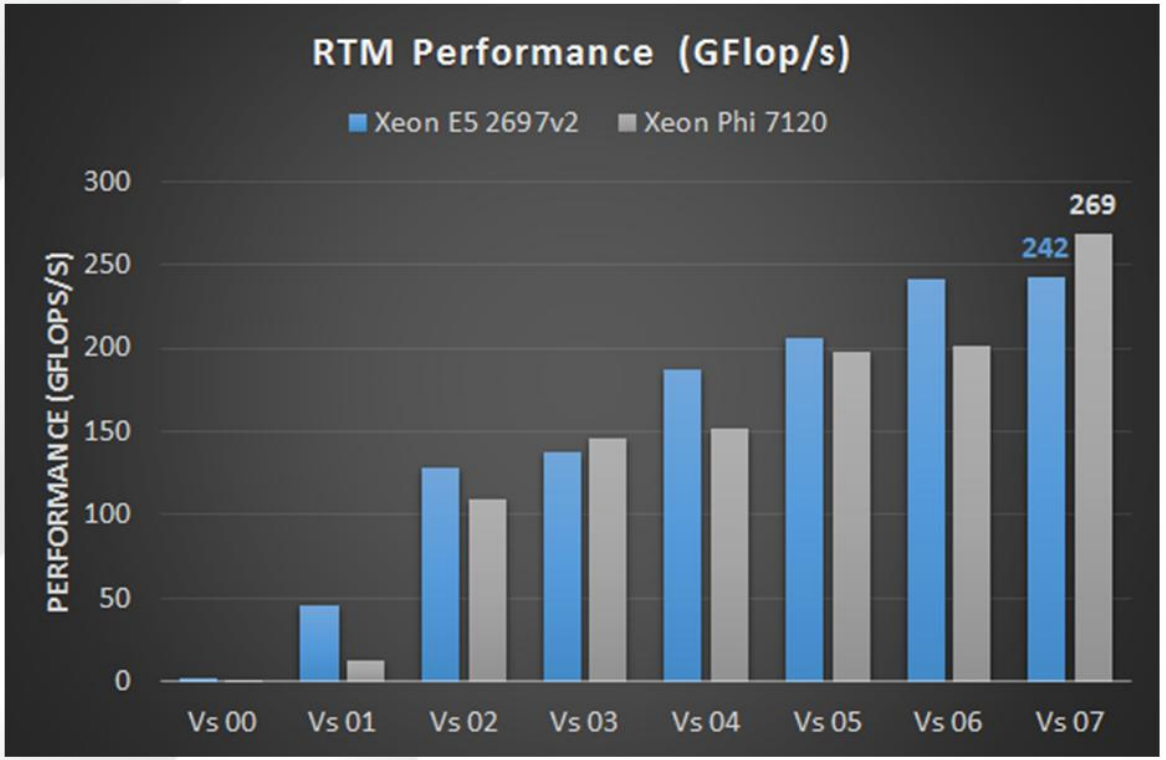
Memory Prefetch Vs 07

Speedup = 1.00x

Speedup = 1.34x

Prefetching

-opt-prefetch-distance=d1,d2	!$omp simd safelen(d)

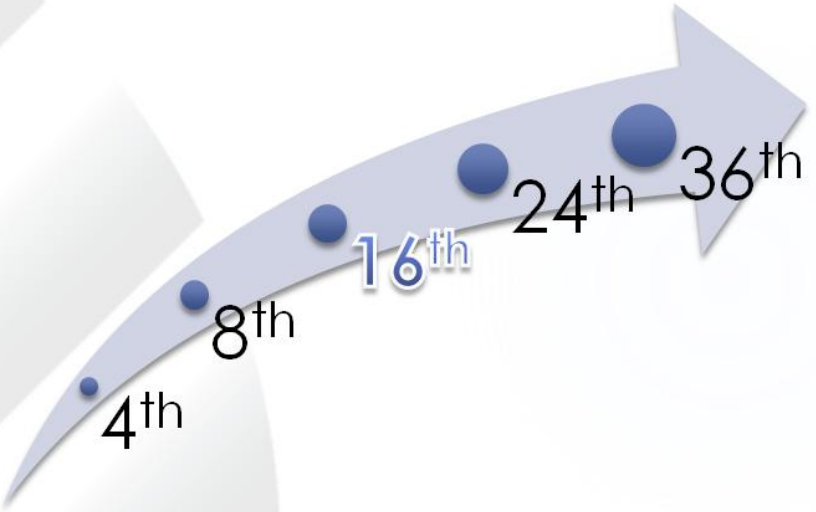


81% of the roofline limit

45% of the roofline limit

What about other discretization orders?

- Vs 07 optimizations is STD to all orders
- Some fine adjusts to each order



Mesh Spacing (Δx)

Array Cells

Arithmetic Intensity

Memory Amount

Time Iterations

Time Step (Δt)

? Performance

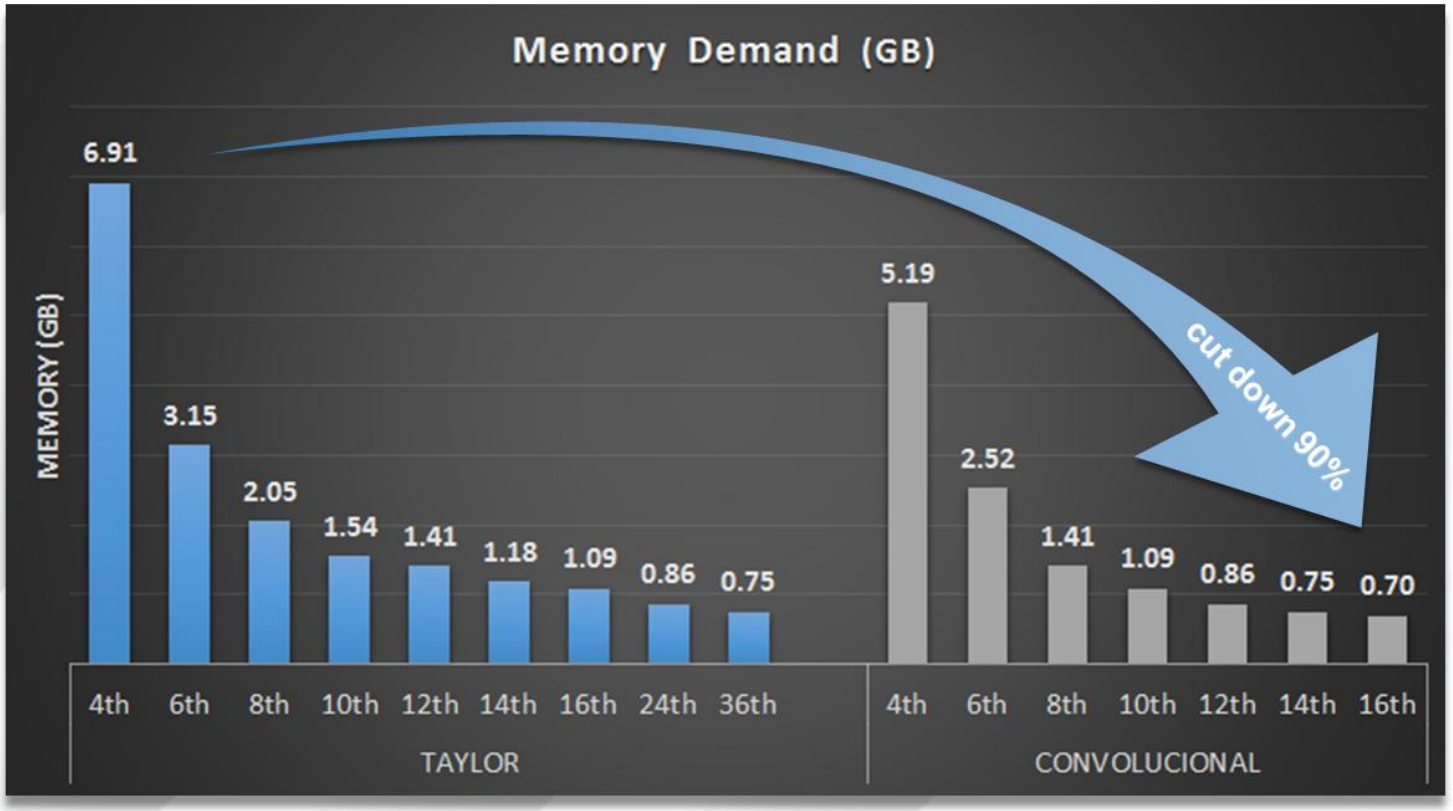
? Overall Time

Derivative Weights

* ZHOU, B., GREENHALGH, S. A., 1992, "Seismic scalar wave equation modeling by a convolutional differentiator", Bulletin of the Seismological Society of America, v. 82, pp. 289–303.

** CHU, C., STOFFA, P. L., 2012, "Determination of finite-difference weights using scaled binomial windows", Geophysics, v. 77, pp. 57–67.

*** Silva, B. S., 2014 "Avaliação de Operadores Convolucionais na Solução da Equação Acústica da Onda", Dissertation, Federal University of Rio de Janeiro, Civil Engineering Program. <www.coc.ufrj.br/index.php/dissertacoes-de-mestrado/380-2014/4429-bruno-de-souza-silva>



Convol. 16th order

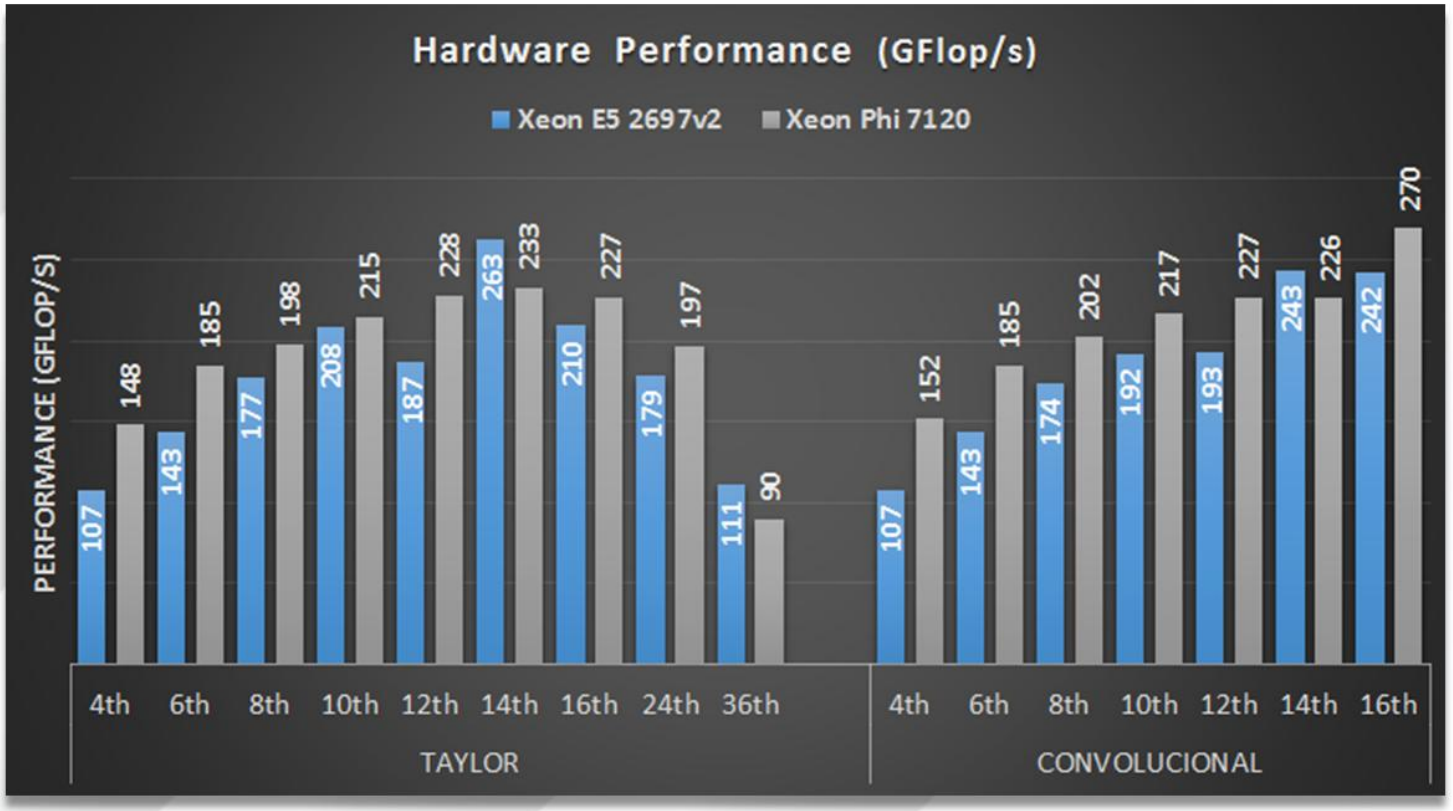
As effecient as Taylor 36th	Saves 90% in memory

Mesh Spacing (Δx)

Array Cells

Memory Traffic

Memory Amount

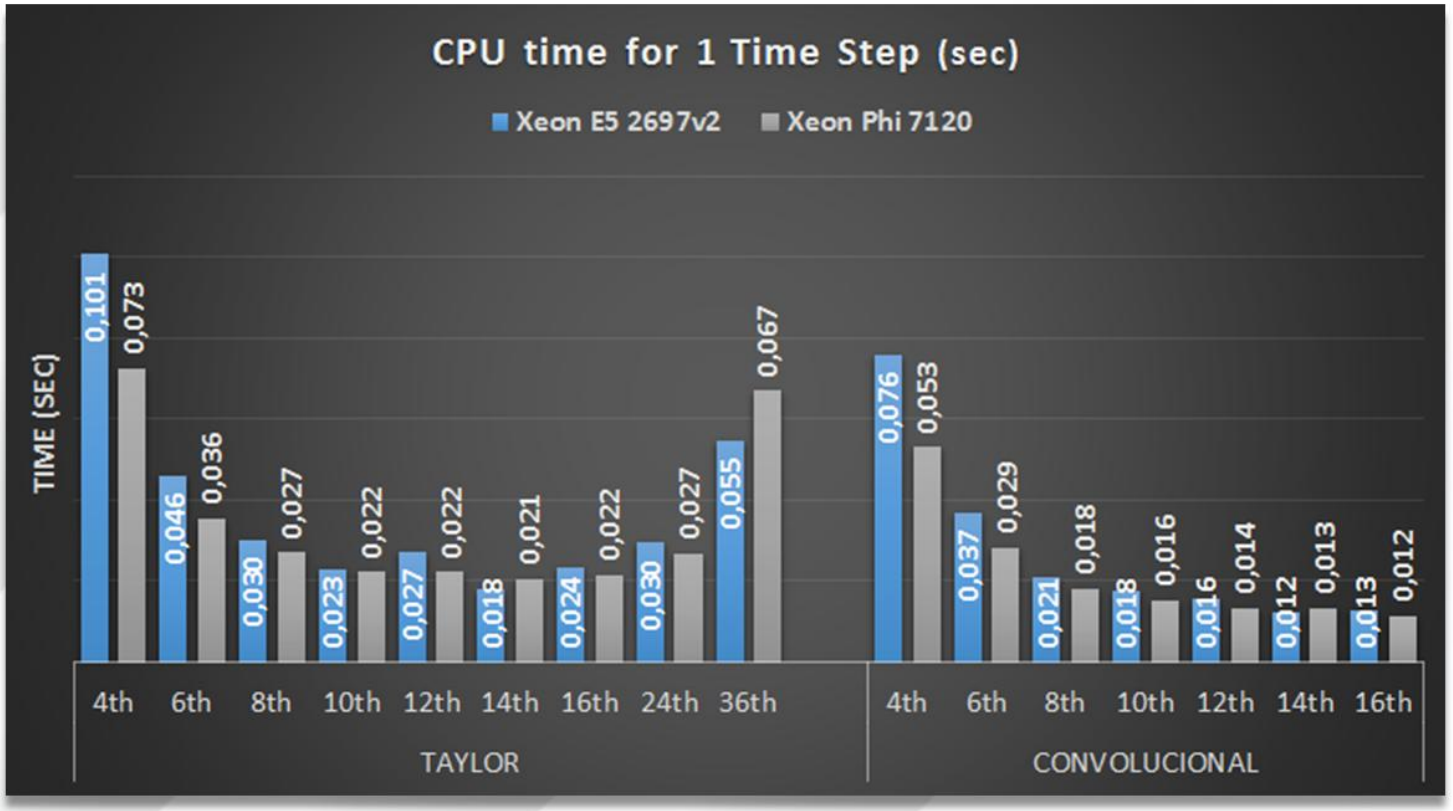


Arithmetic Intensity

Array Cells

Performance

Overall Time



14th Convol.

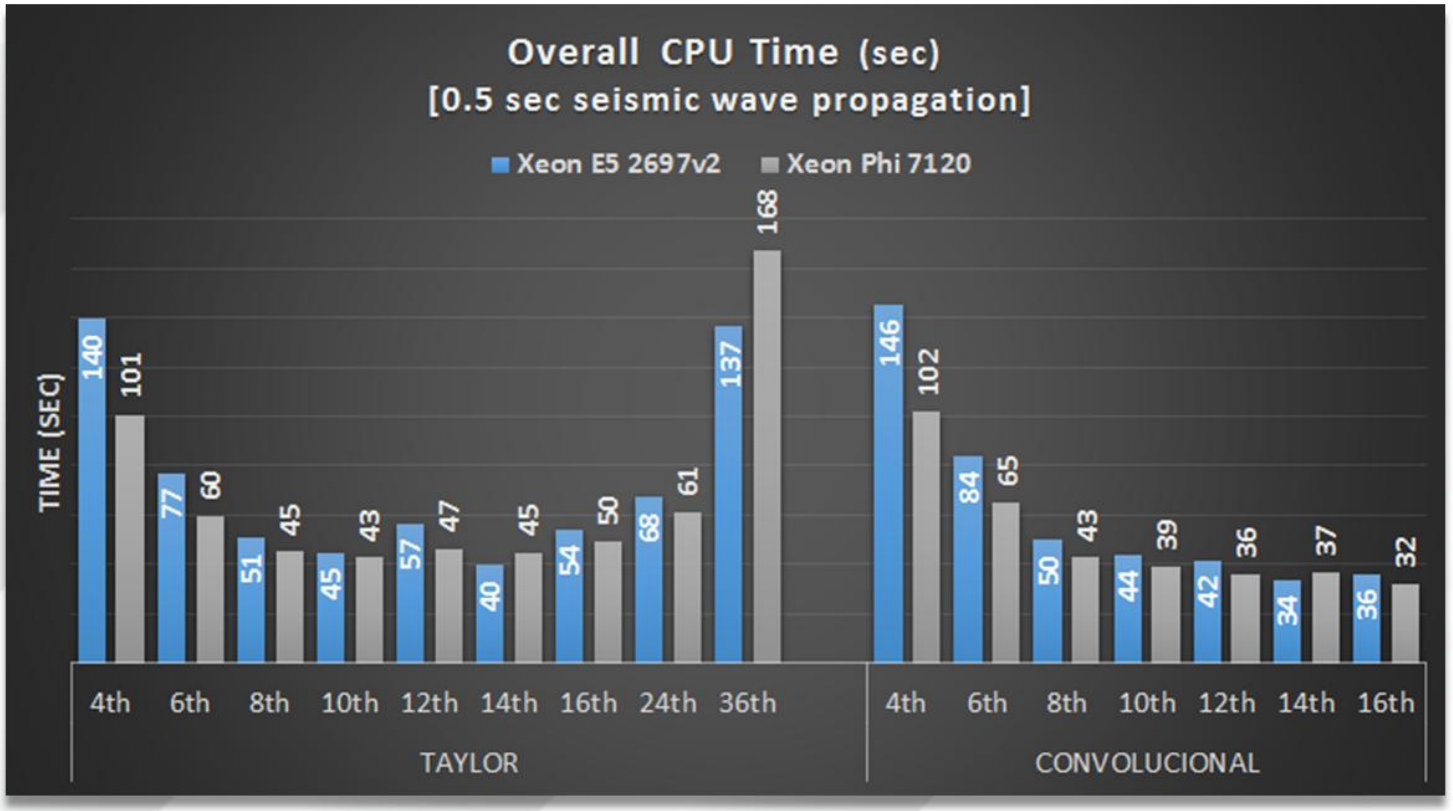
16th Convol.

Arithmetic Intensity

Array Cells

Time Iterations

Time Step (Δt)

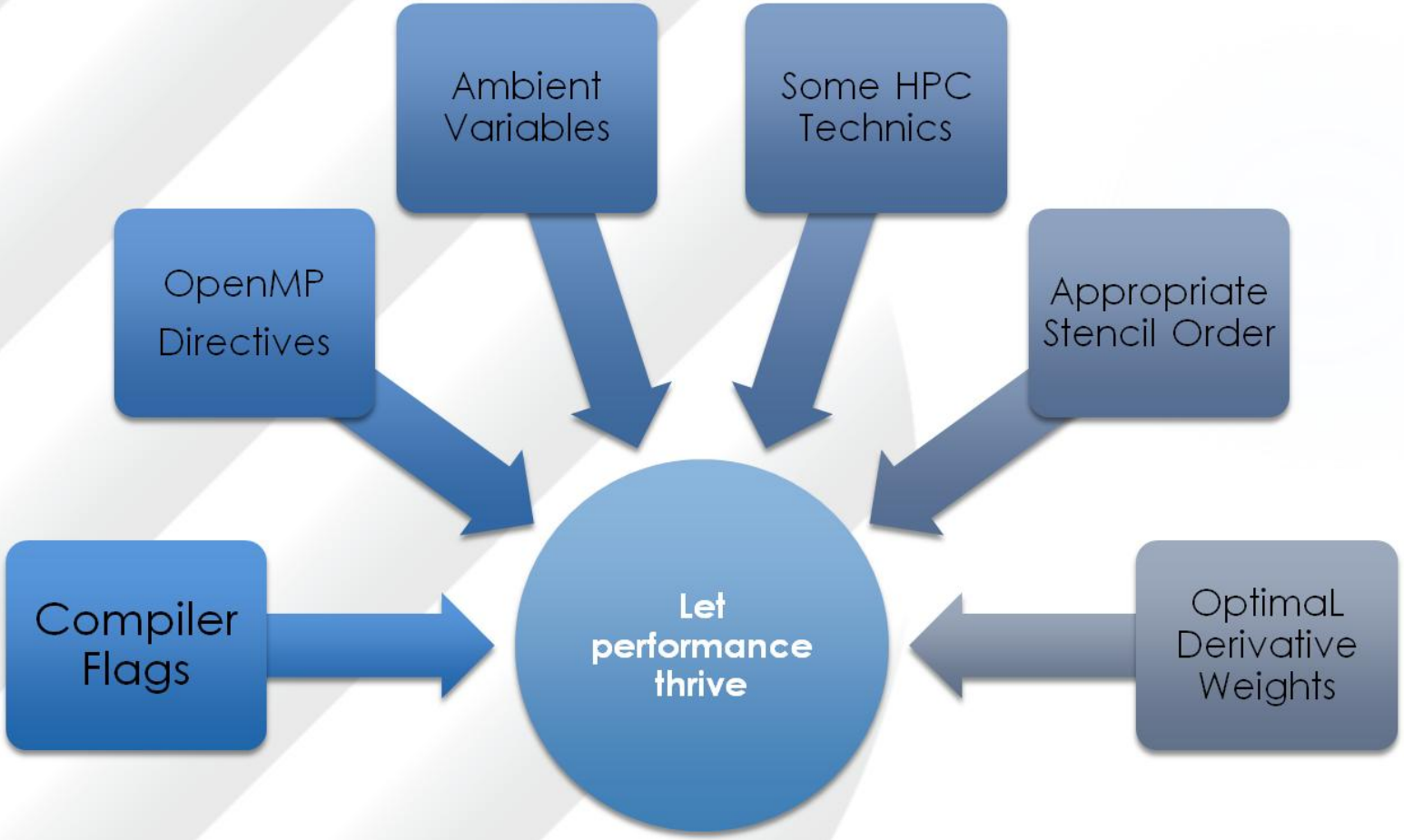

Overall CPU Time (sec)
[0.5 sec seismic wave propagation]
Xeon E5 2697v2
Xeon Phi 7120
TIME (SEC)
140
101
77
60
51
45
45
43
57
47
40
45
54
50
68
61
137
168
146
102
84
65
50
43
44
39
42
36
34
37
36
32
4th
6th
8th
10th
12th
14th
16th
24th
36th
TAYLOR
CONVOLUCIONAL


intel inside
Xeon
14th Convol.
intel inside
Xeon Phi
16th Convol.


Time
Iterations
Time Step
(Δt)


Performance
Overall
Time

AT LONG LAST

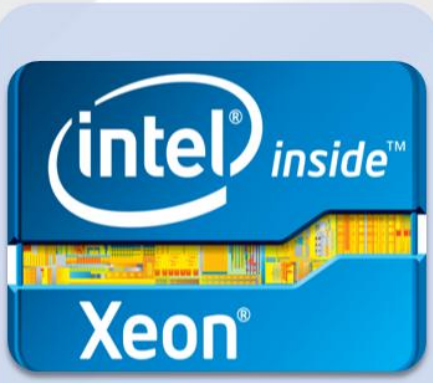

Ambient Variables
Some HPC Technics
OpenMP Directives
Appropriate Stencil Order
Compiler Flags
Let performance thrive
OptimaL Derivative Weights

Future Works

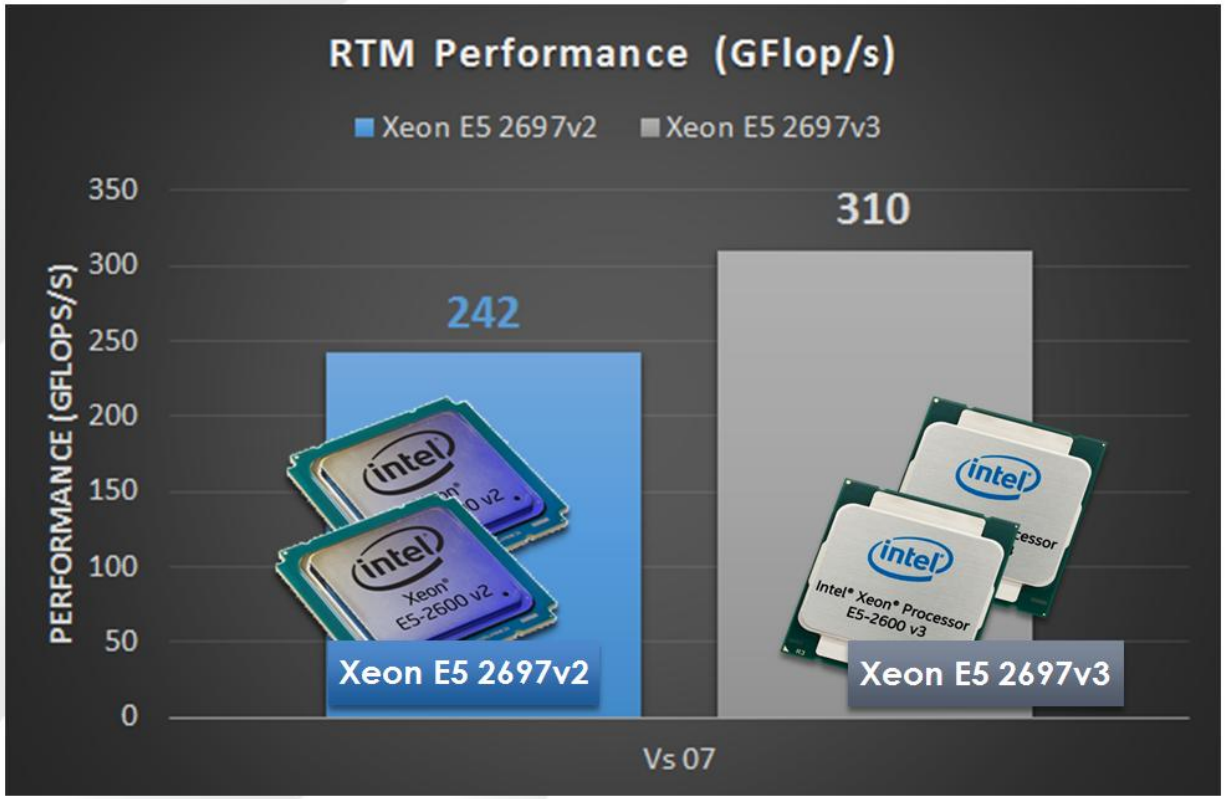
- Low level optimizations for Xeon Phi
- Apply optimizations to the anisotropic 3D TTI RTM kernel
- Two level parallelism
- Towards U.Q.

A little bit more

New Xeon E5 v3 Family



35 MB Cache
2133 MHz DDR4
2.6 GHz Clock Speed
2 Sockets x 14 Cores
256 bits Vector
1.16 Tflop/s SP
136 GB/s Bandwidth
Fused MUL/ADD



Danilo L. Costa
danilolc26@gmail.com