Uncertainty in Seismic Imaging: Exploring a New Frontier on HPC

Alvaro Coutinho

Professor at the Federal University of Rio de Janeiro

COPPE iPCC (Intel Parallel Computing Center)

SUMMARY

- **Uncertainty Quantification & Seismic Imaging**

    1 - Seismic **imaging** in Exploration & Production Process (E&P).

    2 - **Uncertainties** in seismic imaging.

    3 - Example**.**

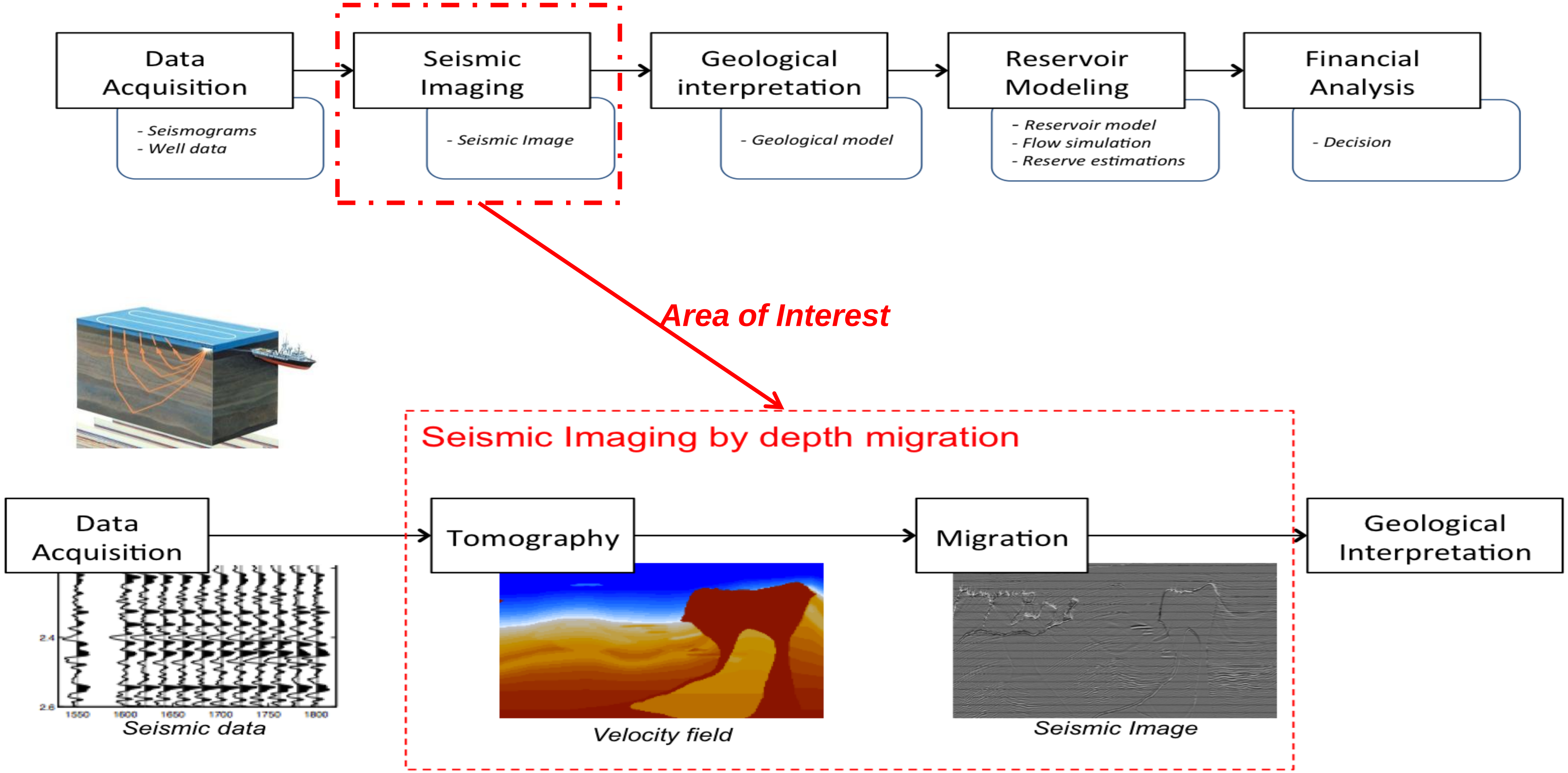
- **Scientific Workflow**

    - Debugging, Profiling & Tuning Scientific Workflows using **Chiron**

    - Performance analysis

- **Kernel Computational optimizations applied on RTM (Reverse Time Migration) algorithm.**

Exploration & Production (E&P) decision process



Computationally intensive. Requires thousands of shots and a post-processing step to obtain an image

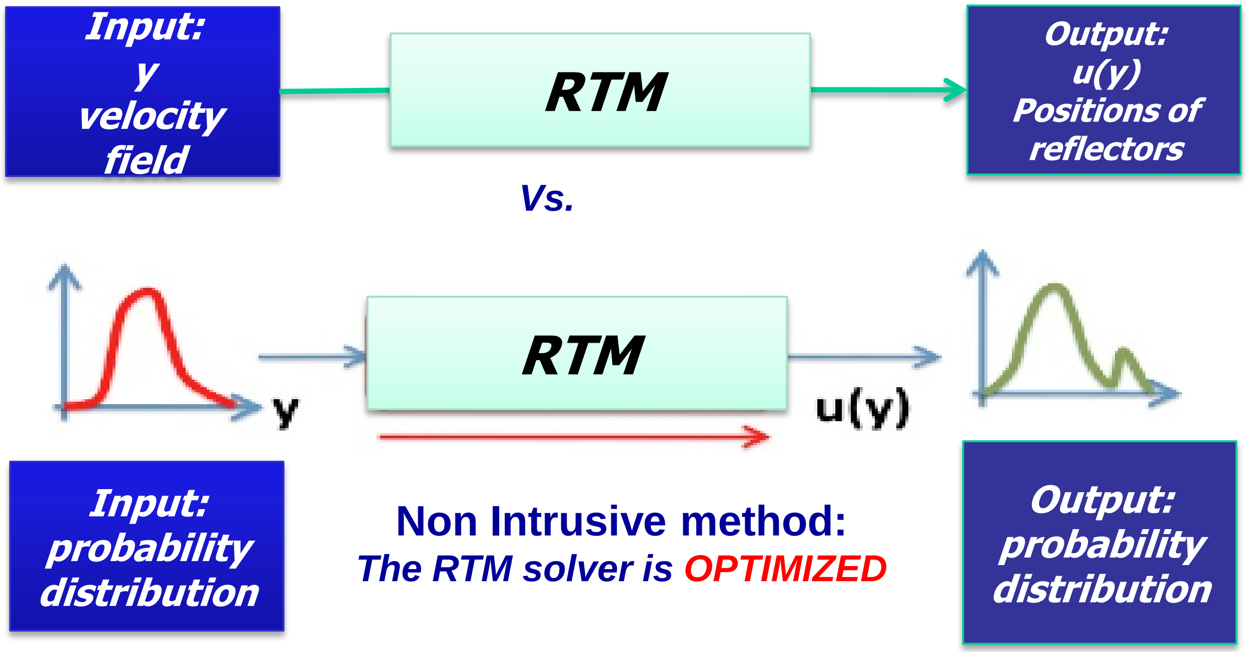
The migration uses the output of *ill-posed problem*: *the tomography, it is an admissible velocity field of the subsurface*

There exists **uncertainties** :
- Low fidelity physical models for the wave propagation
- Noisy measured data
- etc

Can lead to drastically wrong results / decisions

Uncertainty Quantification technique

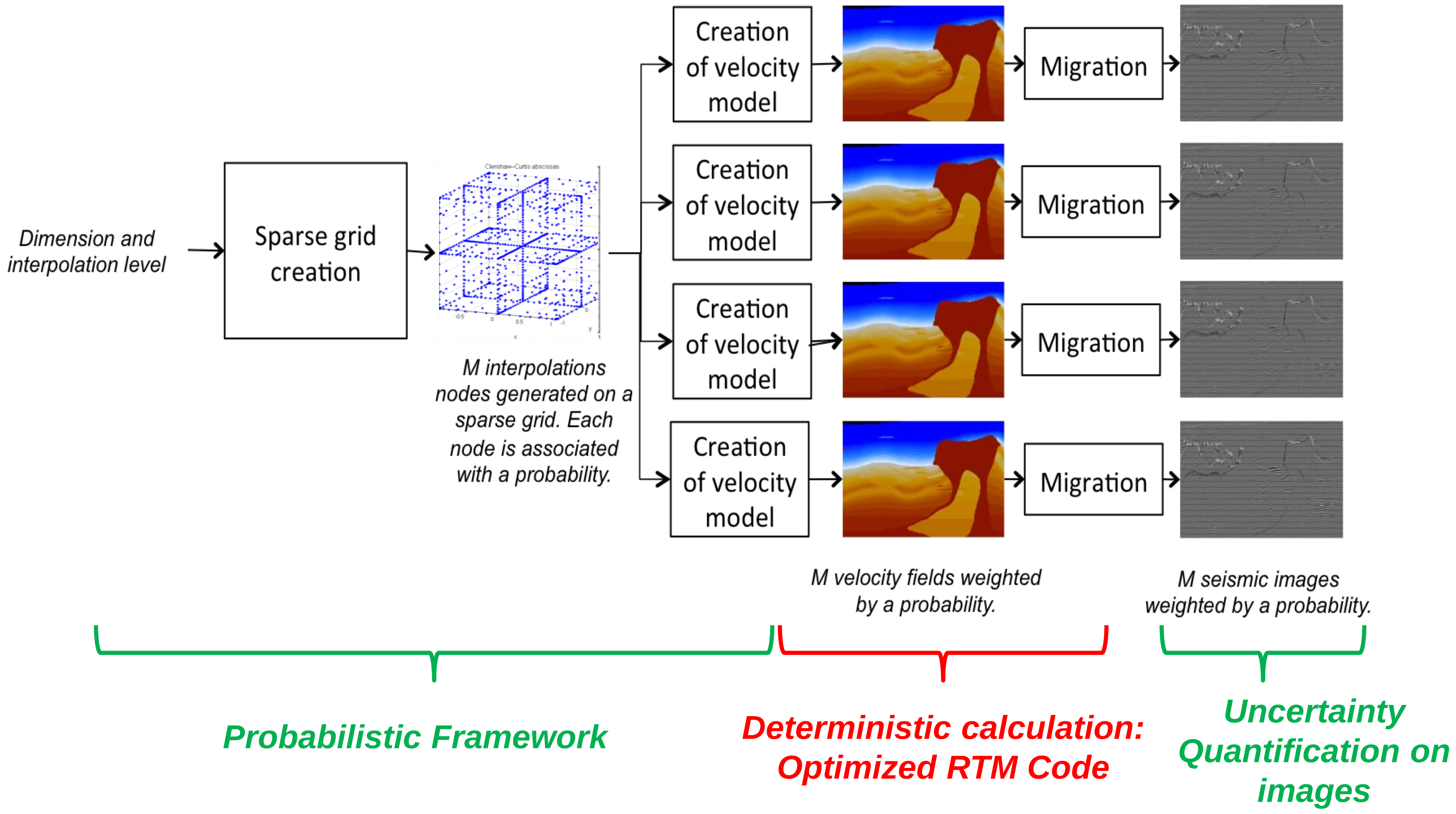
- **We want to define a framework for the propagation of velocity field uncertainties in RTM. The framework needs to be:**
  - **Probabilistic**: the output of tomography is stochastic, the formulation needs to be probabilistic.
  - **Non-intrusive**: we have a well **optimized RTM code**.



  - **Effective**: we want to run the less possible number of RTM.

- **Choice of <u>stochastic collocation</u> (interpolation approach).**

Uncertainty Quantification in seismic imaging



Uncertainty Quantification in seismic imaging needs Scientific Workflow

- **Requires M RTMs.**
  - If we consider the number of points needed around **1,000**

    the computational cost will be 1,000 * 3 min[1] ≈ **50h for one shot** RTM but **5000h for multiple shots** RTM (100 shots).
  - Requires a lot of **computational power**.
- **Scientific Workflow Management System needed to support the intensive computations:**

CHIRON: : a parallel workflow execution engine

Dataflow oriented by a workflow relational algebra

Applications
- Computational Fluid Dynamics
- Risers,
- UQ
- RTM

Strong Provenance Support
- Debugging scientific workflows
at runtime

User steering at runtime

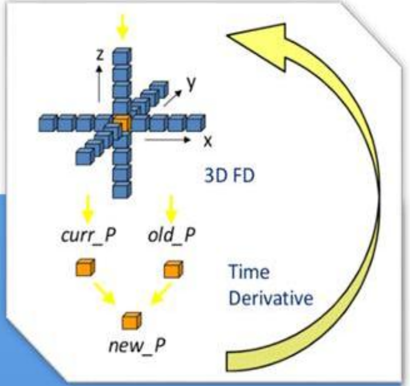
Data provenance can be defined by "the description of the origins of a piece of data and the process by which it arrived in a database"(Buneman et al.)

Dias, Jonas, et al. "Data-centric iteration in dynamic workflows." Future Generation Computer Systems (2014)

OGASAWARA, E., DIAS, J., SILVA, V., et al.,2013,"Chiron: A Parallel Engine for Algebraic Scientific Workflows",Concurrency and Computation, v.25, n.16, pp.2327–2341.

Computational Optimizations for RTM kernels

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
```

16^{th} order → 3D → Acoustic → Isotropic → 61 Flops

Computational optimizations for RTM kernels

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 also is a **bottleneck**
- **Large geological areas demands 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)

RTM – Model used

- **Volume size: 12km x 6km x 8 km**
- **Mesh size 21.5 meters (structured grid)**
- **Points in direction X: 558**
- **Points in direction Y: 279**
- **Points in direction Z: 372**

Testbed Platforms

intel inside Xeon	intel inside Xeon Phi
30 MB Cache	30 MB Cache
1866 MHz DDR3	16 GB DDR3 Mem
2.7 GHz Clock Speed	1.2 GHz Clock Speed
2 Sockets x 12 Cores	61 Physical Cores
256 bits Vector	512 bits Vector
1.04 Tflop/s SP	2.4 Tflop/s SP
119 GB/s Bandwidth	352 GB/s Bandwidth

Computational optimizations for RTM kernels - Methods

BASE CODE
Vs 00

Automatic compiler optimization
- -O3

No vectorization
- -no-vec

No parallelization
- -openmp-stubs

PARALLELIZATION
Vs 01

Automatic compiler optimization
- -O3

No vectorization
- -no-vec

Parallelization
- !$omp parallel do
- -openmp

VECTORIZATION
Vs 02

Automatic compiler optimization
- -O3

Vectorization
- -vec
- -xAVX

Parallelization
- !$omp parallel do
- -openmp

MEMORY IMPROVEMENTS
Vs 03

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

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

DDR5 arch
- kmp_affinity
- !$omp schedule (dynamic)

RETURN TO BASIS
Vs 05

Fast data access

Avoid memory fragmentation

Static Allocation

Managed efficiently by CPU

Improves previus and next optimizations

ALLIGN & PADDING
Vs 06

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

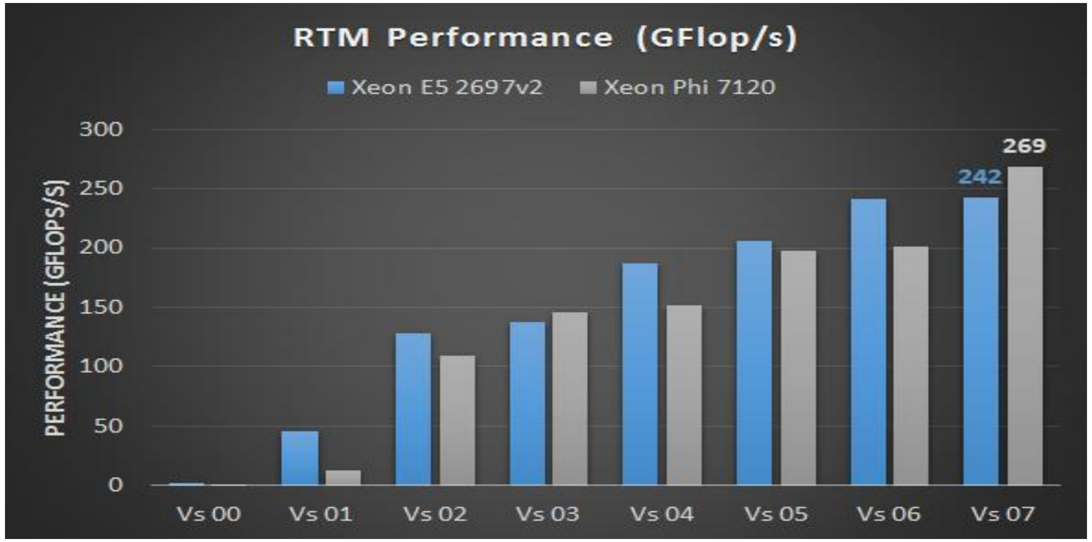
Padding
- -opt-assume-safe-padding
- Add memory positions (Nx+pad , Ny+pad , Nz+pad)

MEMORY PREFETCH
Vs 07

Prefetching

-opt-prefetch-distance=d1,d2

!$omp simd safelen(d)

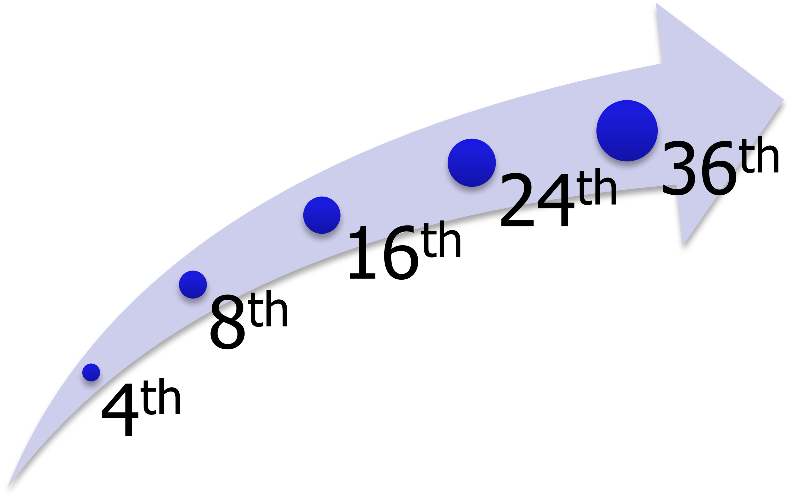


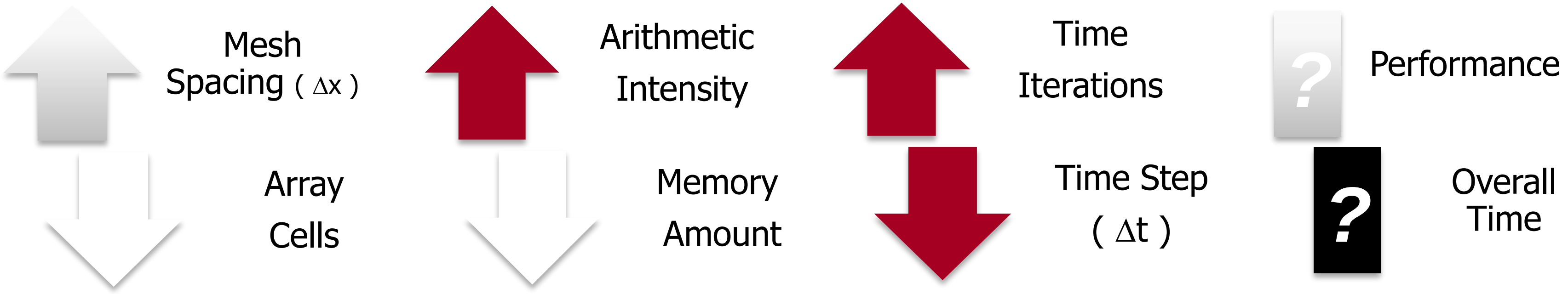
Xeon E5 2697v2: *from 2.2 up to 242 Gflops/s* **Xeon Phi 7120 *: from 0.3 up to 269 Gflops/s***

Computational optimizations for RTM kernels

Spatial discretization order

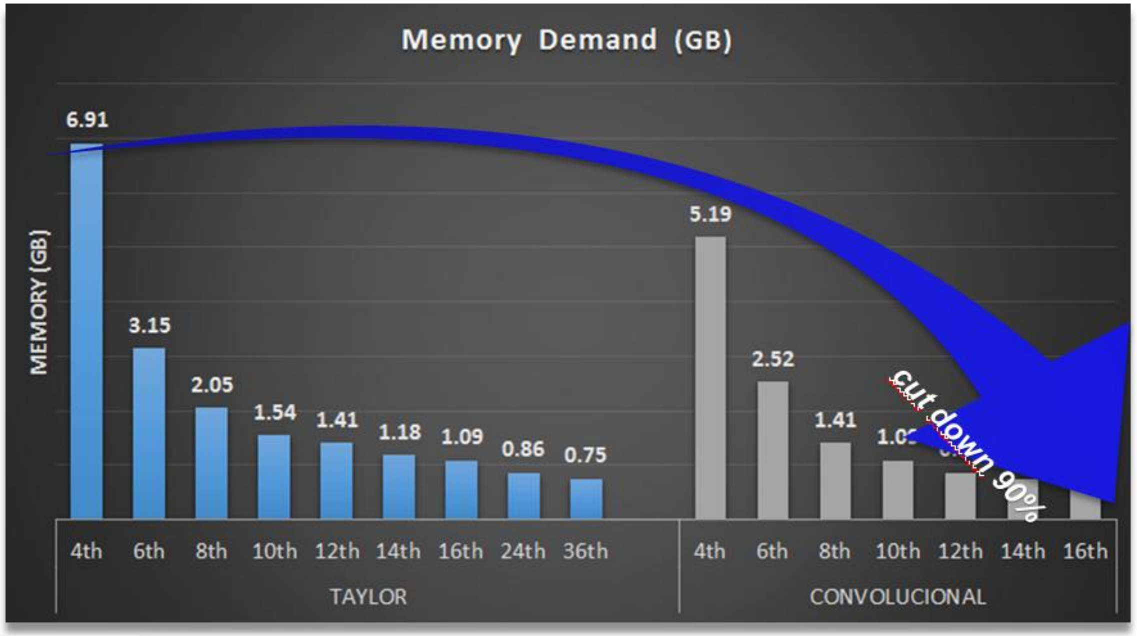
Goal: find higher order stencil formulation that leads to less grid cells and less memory required, without loss of accuracy.





Computational optimizations for RTM kernels

Lower memory requirement allow larger surveys to fit on coprocessor memory

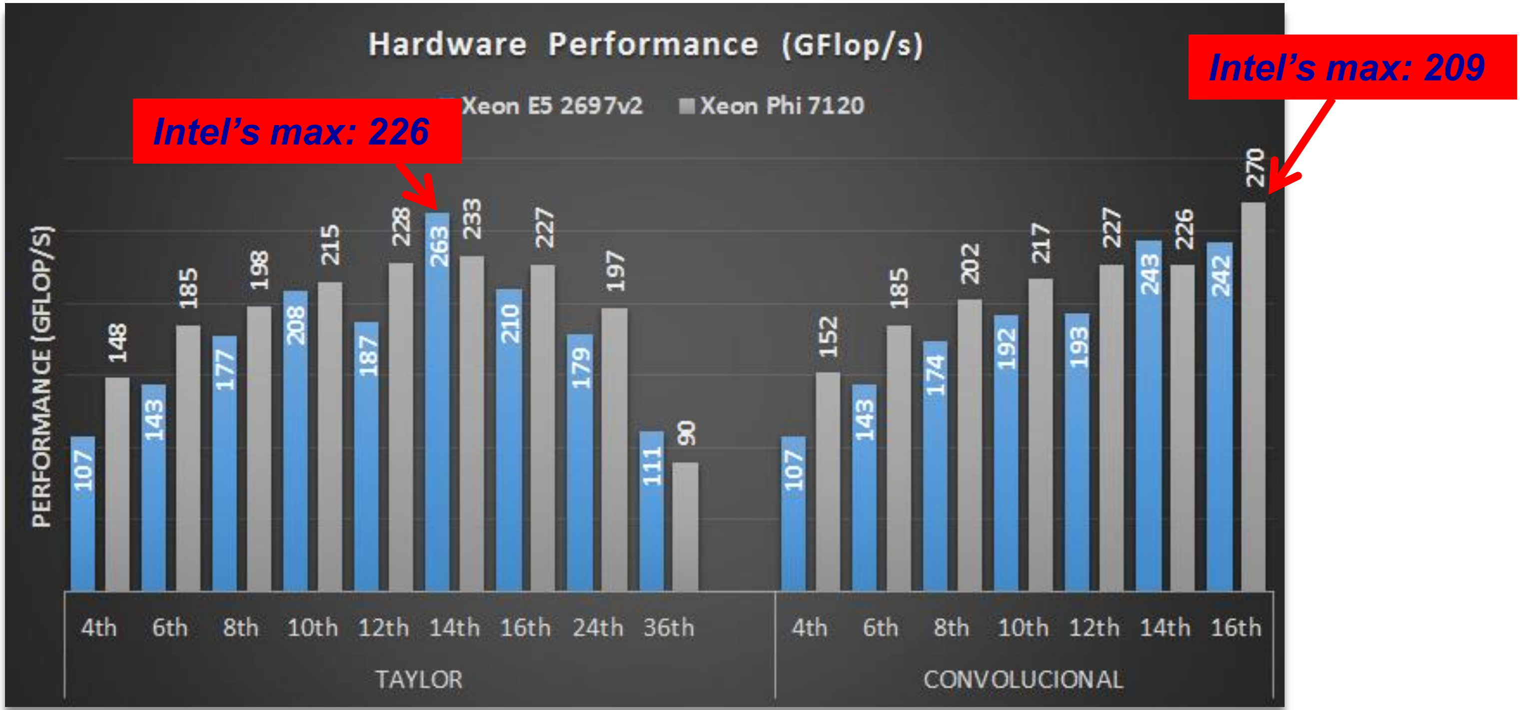


Convol. 16th order

As efficient as Taylor 36th	Saves 90% in memory

Computational optimizations for RTM kernels

New convolutional formulation maintains scalability to 16th order stencils



14th Taylor

16th Convol.

Arithmetic Intensity

Array Cells

? Performance

? Overall Time

CONCLUSION / Remarks / Future Works

- **Optimization** of the RTM algorithm on the **Intel *Xeon Phi***

- Development of a **non-intrusive UQ technique** to the optimized algorithm using **Stochastic Colocation**
  - Test and implementation of different dimension reduction and Interpolation levels in the Sparse grid collocation scheme
  - Compare the error with the Monte Carlo Method
  - **Visualization** and getting insights from UQ

- **Use of a Scientific Workflow Management System (CHIRON)**
  - Development of a strategy to gather and query performance data while scientific workflows are executing
  - Improvement in this architecture to obtain a better performance using 1,000 cores
  - RTM workflow execution using **large core counts**
  - **Problem of storage capacity: limited** for a large-scale data app like RTM

  (In our Cluster Uranus, we produced ~40 GB of files using a small dataset)