Accelerating real-world stencil computations using temporal blocking: handling sparse sources and receivers

George Bisbas [1] Fabio Luporini [1](advisor) Mathias Louboutin [2](advisor) Gerard Gorman [1] (advisor) Paul H.J. Kelly [1](advisor)

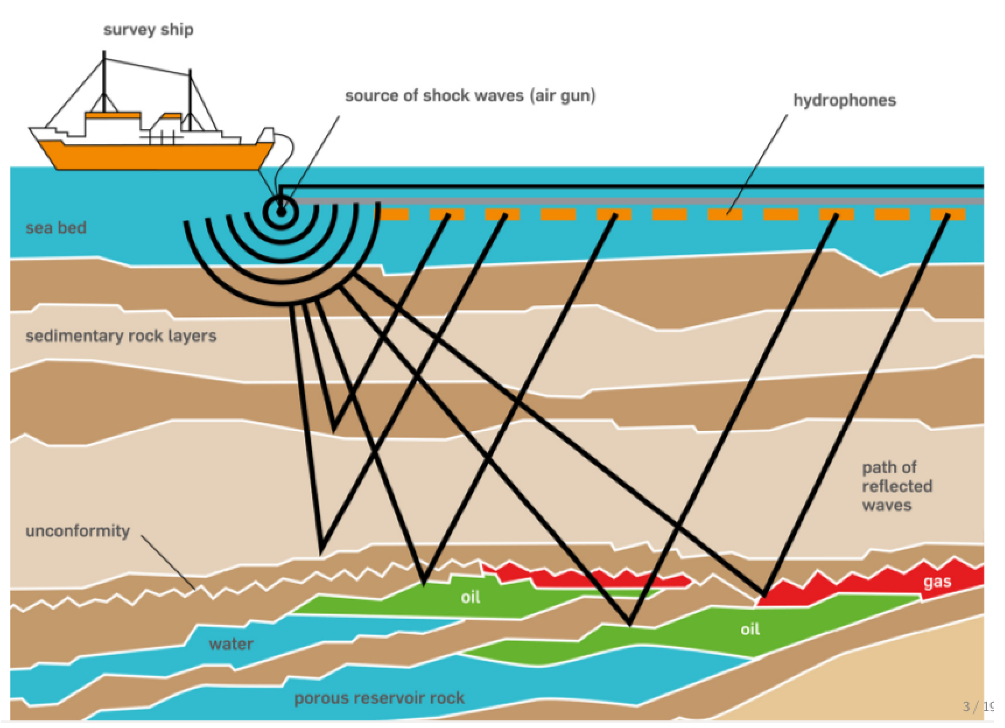
[1]Imperial College London, UK [2]Georgia Institute of Technology, Atlanta, USA

Background

- **Temporal blocking**, also known as time-tiling is a well-known technique for accelerating stencil based computations by enhancing data locality.
- Solving **partial differential equations** with the **finite difference method** yields stencil codes that can be very complex for real world problems.
- **Devito [1]** is a tool for generating **stencil codes** from a high-level symbolic abstraction aiming at real-world applications targetting mostly **seismic imaging**.
- Seismic inversion requires the **support for sparse operations** such as source injection and wavefield measurements at arbitrary grid locations. This is an obstacle to temporal blocking as non-affine accesses are involved, and interpolation spans tiles.

Contributions

- An algorithm consisting of an **inspector/executor** scheme for efficiently precomputing the effect of sparse operations such as source injection and receiver interpolation, with low overheads in time and space.
- A demostration that this enables temporal blocking, with a simple temporal blocking scheme (more sophisticated blocking schemes are also possible).
- Performance evaluation showed much better runtime improvement over (vectorized) space blocking alone for low order stencils and quite competitive for higher order.

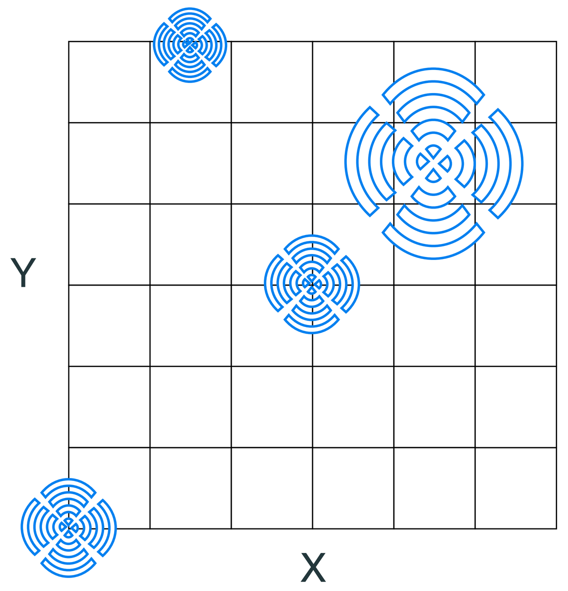


A seismic imaging survey. Acoustic sources inject a wavefield and receivers take measurements. **Source: Open Learn**

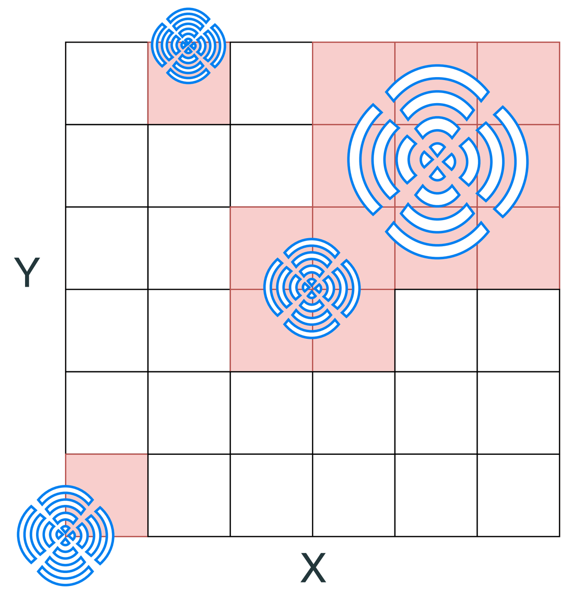
The algorithm

Our **inspector/executor** scheme for this loop nest transformation consists of the following 5 steps:

1. Inspection: Iterate over the sources (Fig:1a) for all timesteps to identify the unique points that are affected (Fig:1b) and then allocate space for a 2D array of size *unique_pts_affected* $\times$ *timesteps* .
2. Inspection: Two copies of our grid act as a binary (Fig:1c) and as an ID mask (Fig:1d) for the affected points.
3. Sparse array is saved in CSR-like format.
4. Execution: Iterate again over the sources, applying their effect (Fig:1f) to our allocated structure described in step 1
5. Execution: Transform the perfect loop nest for temporal blocking and adding the effect of the corresponding source using the binary mask created in steps 1 and 3



(a) Step 0: Sources are injected on a field.



(b) Step 1: Identify area affected from sources.

0	1	0	1	1	1
0	0	0	1	1	1
0	0	1	1	1	1
0	0	1	1	0	0
.	.	.	.	.	.
1	0	0	0	0	0

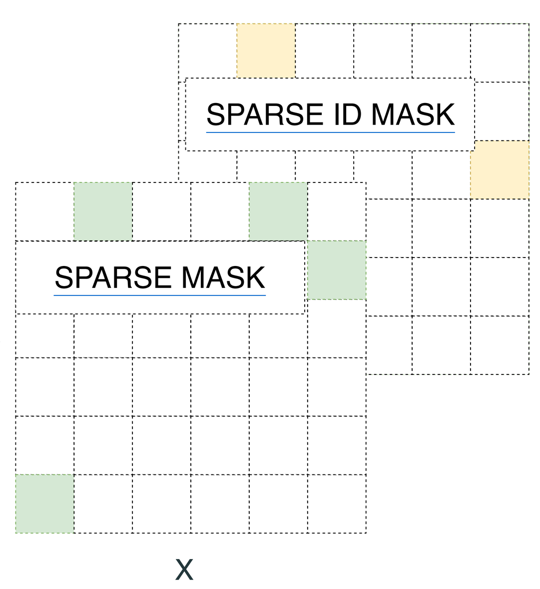
Y / X

(c) Step 2: Binary mask with 1s on affected points.

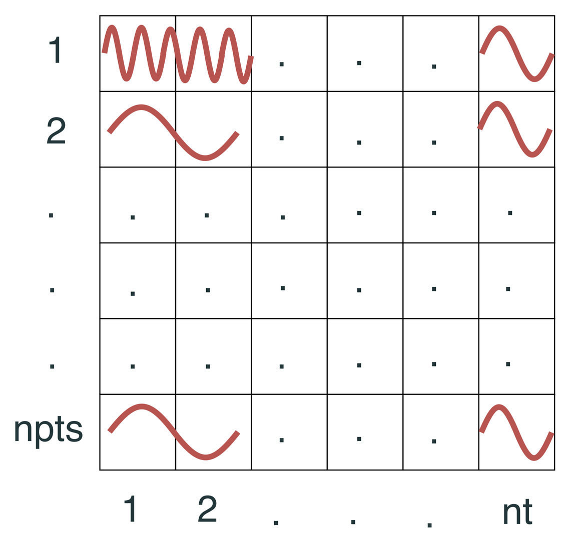
0	1	0	2	3	4
0	0	0	5	6	7
0	0	8	9	10	11
0	0	12	13	0	0
.	.	.	.	.	.
npts	0	0	0	0	0

Y / X

(d) Step 3: A unique ID for each affected unique point.

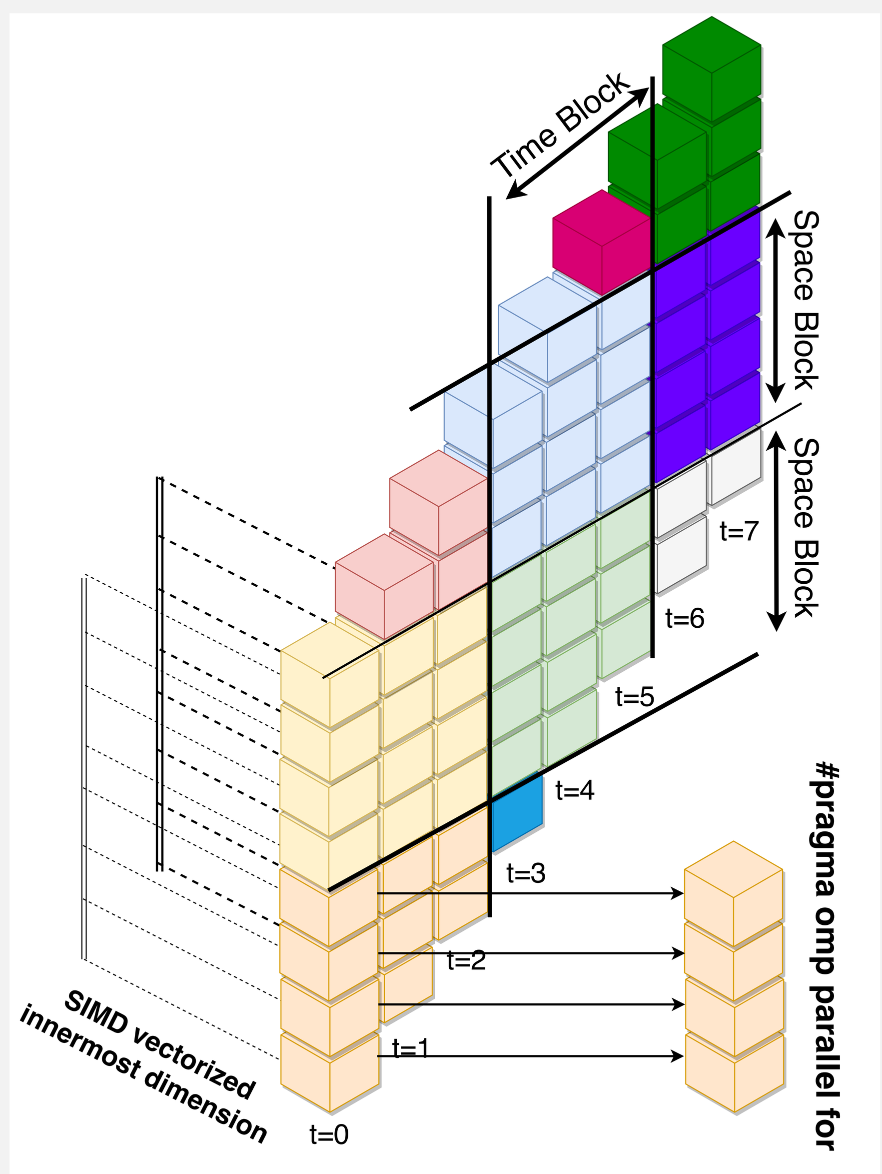


(e) Step 4: Save space by storing in a CRC format.



(f) Step 5: Store efficiently all the precomputed source injection.

Skewed temporal blocking



An example of a 8 x 8 x 8 grid computation **skewed by a factor of 1 in x and y dimensions**. An **OpenMP thread worker** is assgined to every **x,y** pair while the **innermost z dimension is SIMD vectorized** at its full extent. The space blocks have shape of 4 x 4 x 8.

Request code

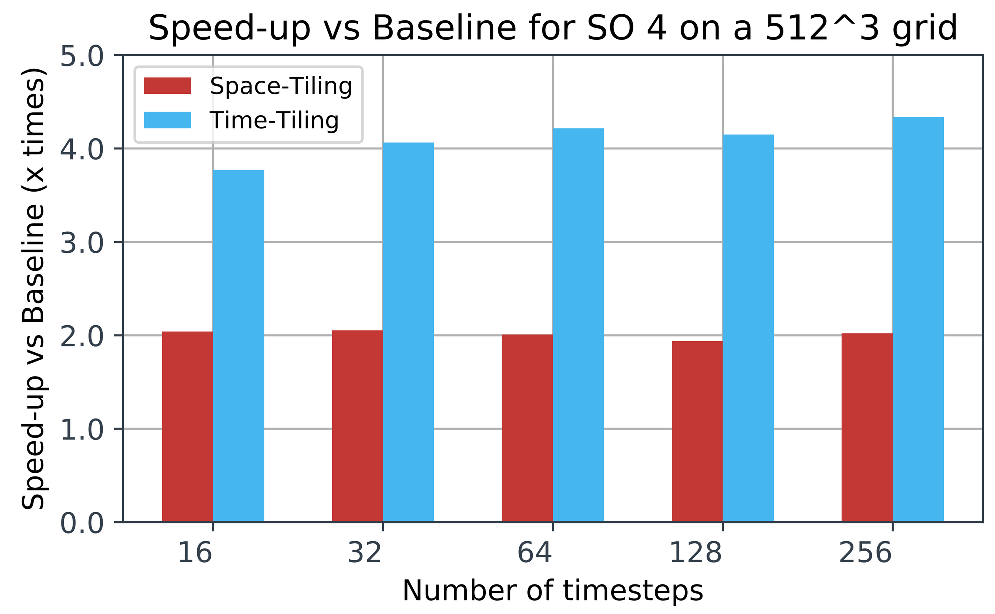
Future work: automate temporal blocking in Devito

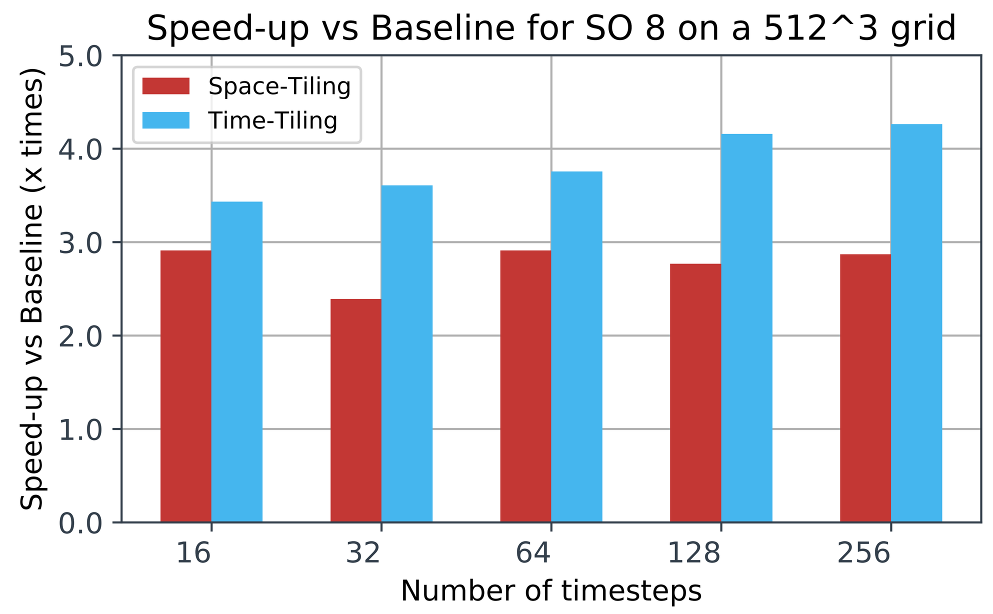
- Automated code generation from a high-level mathematical abstraction with various types of **performance optimizations** to be performed during code generation.
- **MPI/OpenMP** parallelism and **SIMD** vectorization.
- Sophisticated loop transformations (e.g., blocking and auto-tuning).
- Domain-specific symbolic optimizations.

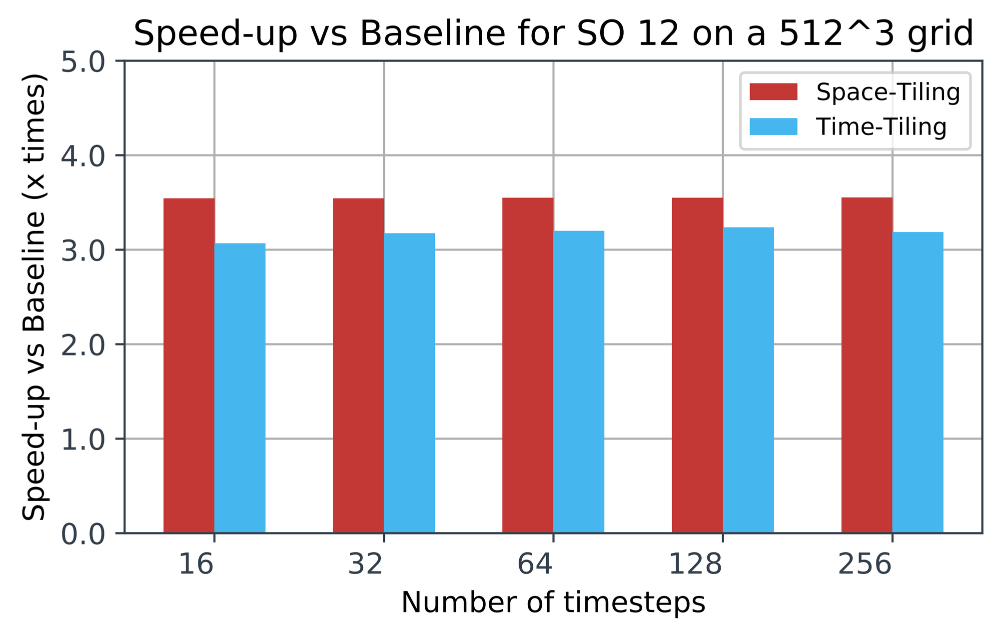
Goal: Enhance Devito with optimally tuned temporal blocking to increasing the performance of generated code, targetting real world applications.

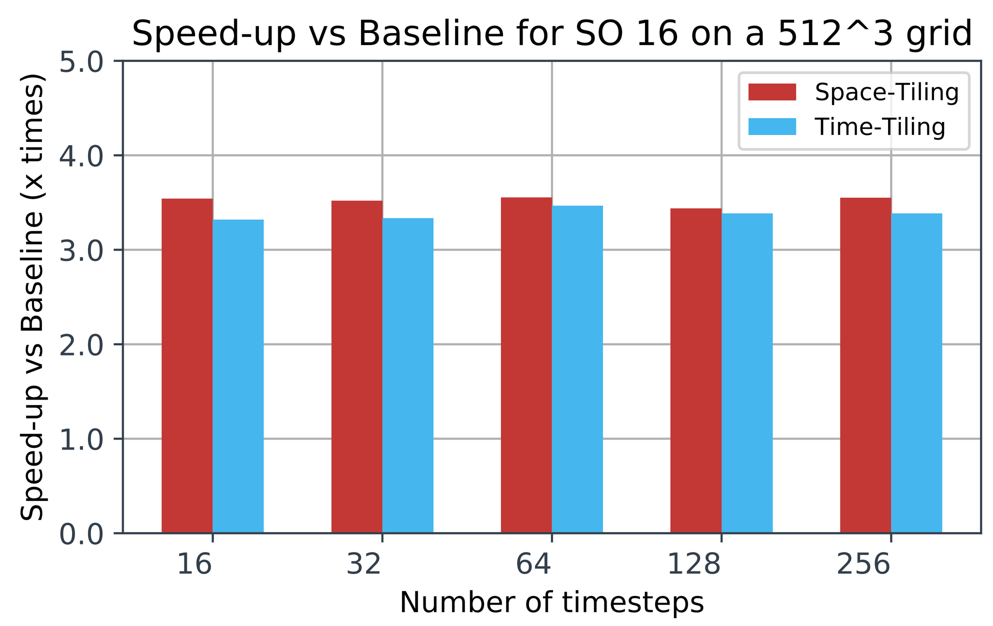
Evaluation results

Experiments were executed for 13p, 25p, 37p and 49p Jacobis (SO 4, 8, 12 and 16 respectively) on a 512^3 grid with 40 sources injecting for arbitrary number of timesteps and block sizes of 32x32x*MAX*. Temporal blocking speedups over space blocking tend to decrease with increasing space order. Peak-performance was improved by around 15-20% over space blocking for low space orders while remaining almost the same for higher ones. Improving that, is work in progress.









References

- Luporini, Fabio, et al. "Architecture and performance of Devito, a system for automated stencil computation." arXiv preprint arXiv:1807.03032 (2018).
- Sim, Nicholas. "Optimising finite-difference methods for PDEs through parameterised time-tiling in Devito." arXiv preprint arXiv:1806.08299 (2018).
- Louboutin, Mathias et al. "Devito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration." (2018).

Links:

- devitoproject.org
- github.com/opesci/devito
- twitter.com/opesciproject

Imperial College London

EPSRC
Engineering and Physical Sciences Research Council