

DEVITO

AUTOMATED HIGH-PERFORMANCE FINITE DIFFERENCES FOR GEOPHYSICAL EXPLORATION

**F. Luporini¹, C. Yount⁴, M. Louboutin³, N. Kukreja¹, P. Witte²,
M. Lange⁵, P. Kelly¹, F. Herrmann³, G.Gorman¹**

¹Imperial College London

²The University of British Columbia

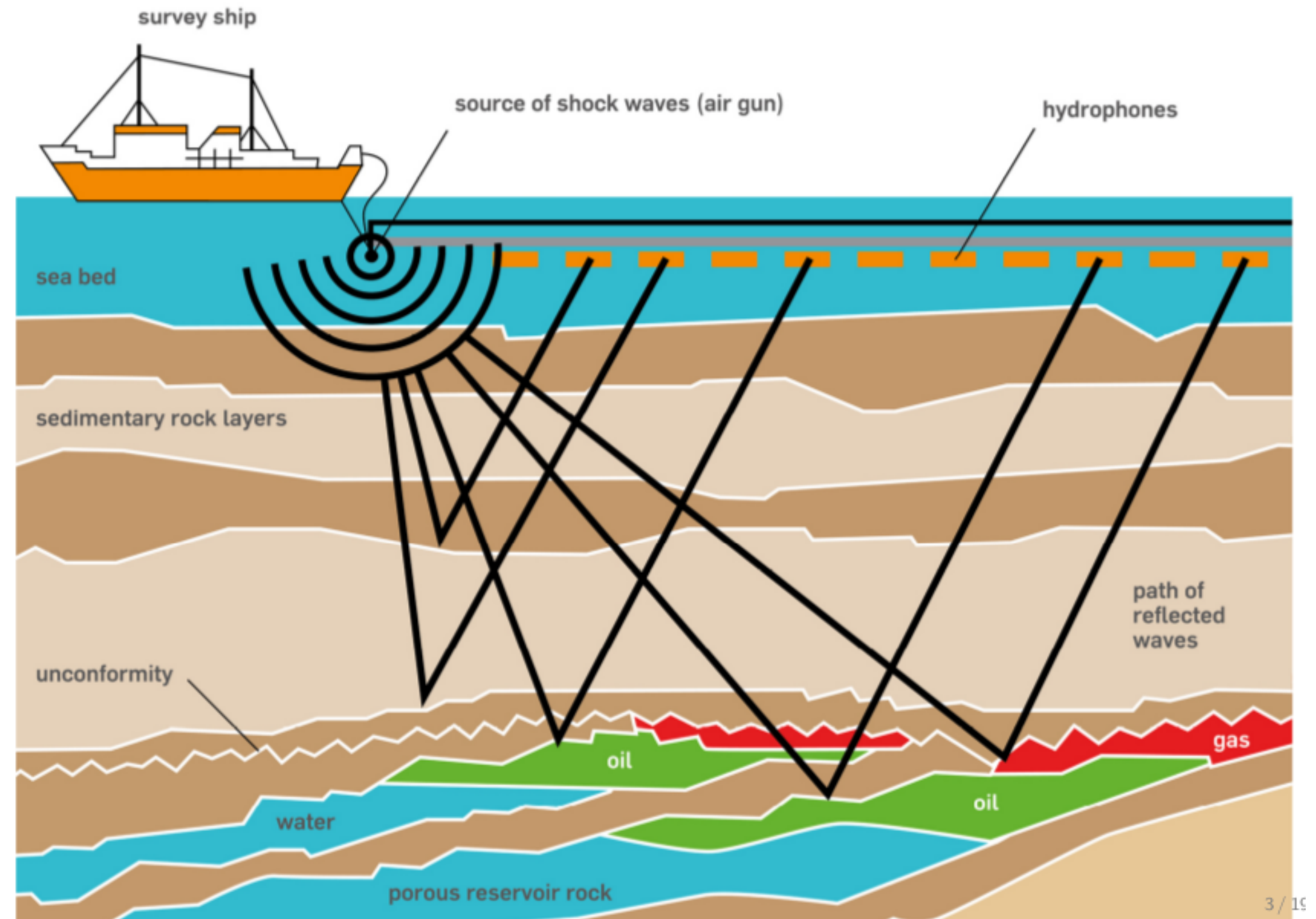
³Georgia Institute of Technology

⁴Intel Corporation

⁵European Centre for Medium-Range Weather Forecasts
(former Imperial College London)

IXPUG 2018

Driving application: inversion algorithms for seismic imaging



<http://www.open.edu/openlearn/science--maths--technology/science/environmental--science/earths--physical--resources--petroleum/content--section--3.2.1>

Issue I: Computational cost

Realistic full-waveform inversion (FWI) scenario:

- **$O(10^3)$ FLOPs per loop iteration** or **high memory pressure**
- Realistic 3D grids with **$>10^9$ grid points**
- Often more than **3000 time steps**
- **Two** operators: forward + adjoint, to be executed **~ 15 times**
- Usually **30000 shots**
- **$\approx O(\text{billions})$ TFLOPs**
- **$>>>$ Days, weeks, months on supercomputers**

Issue 2: Variations in physics and mathematics

- Overarching strategy for inversion
- Formulations of wave equations
- Space and time discretizations
- Boundary conditions, data acquisition, sources/receivers ...

Issue 3: Time flies...

- Proliferation of computer architectures
- Unmaintainable, impenetrable, non-portable legacy code
- Skepticism: C/C++/Fortran **IS** the way

Raising the level of abstraction

$$m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0$$



```
void finite_difference_solver(...)  
{  
    ...  
    <impenetrable code with crazy  
      performance optimizations>  
    ...  
}
```

Raising the level of abstraction

$$m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0$$



```
void finite_difference_solver(...)  
{  
    ...  
    <impenetrable code with crazy  
    performance optimizations>  
    ...  
}
```

Raising the level of abstraction

$$m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0$$

Raising the level of abstraction

$$m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0$$



```
u = TimeFunction(...)
m = Function(...)
eqn = Eq(m * u.dt2 + eta * u.dt - u.laplace, 0)
```


Raising the level of abstraction

$$m \frac{\partial^2 u}{\partial t^2} + \eta \frac{\partial u}{\partial t} - \Delta u = 0$$



```
u = TimeFunction(...)
m = Function(...)
eqn = Eq(m * u.dt2 + eta * u.dt - u.laplace, 0)
```

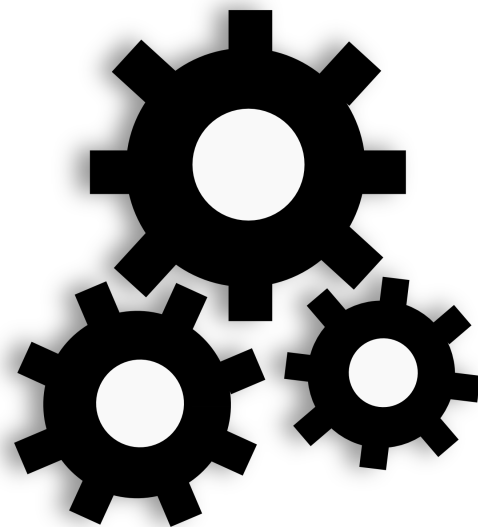


```
void finite_difference_solver(...) { ... }
```

Devito and its key ingredients



- SymPy to express stencil equations
- NumPy for data management
- >>> can exploit SciPy and Dask
- ... simplicity!



- “Real” compiler technology (not a template-based code generator)
- Symbolic processing
- AST processing



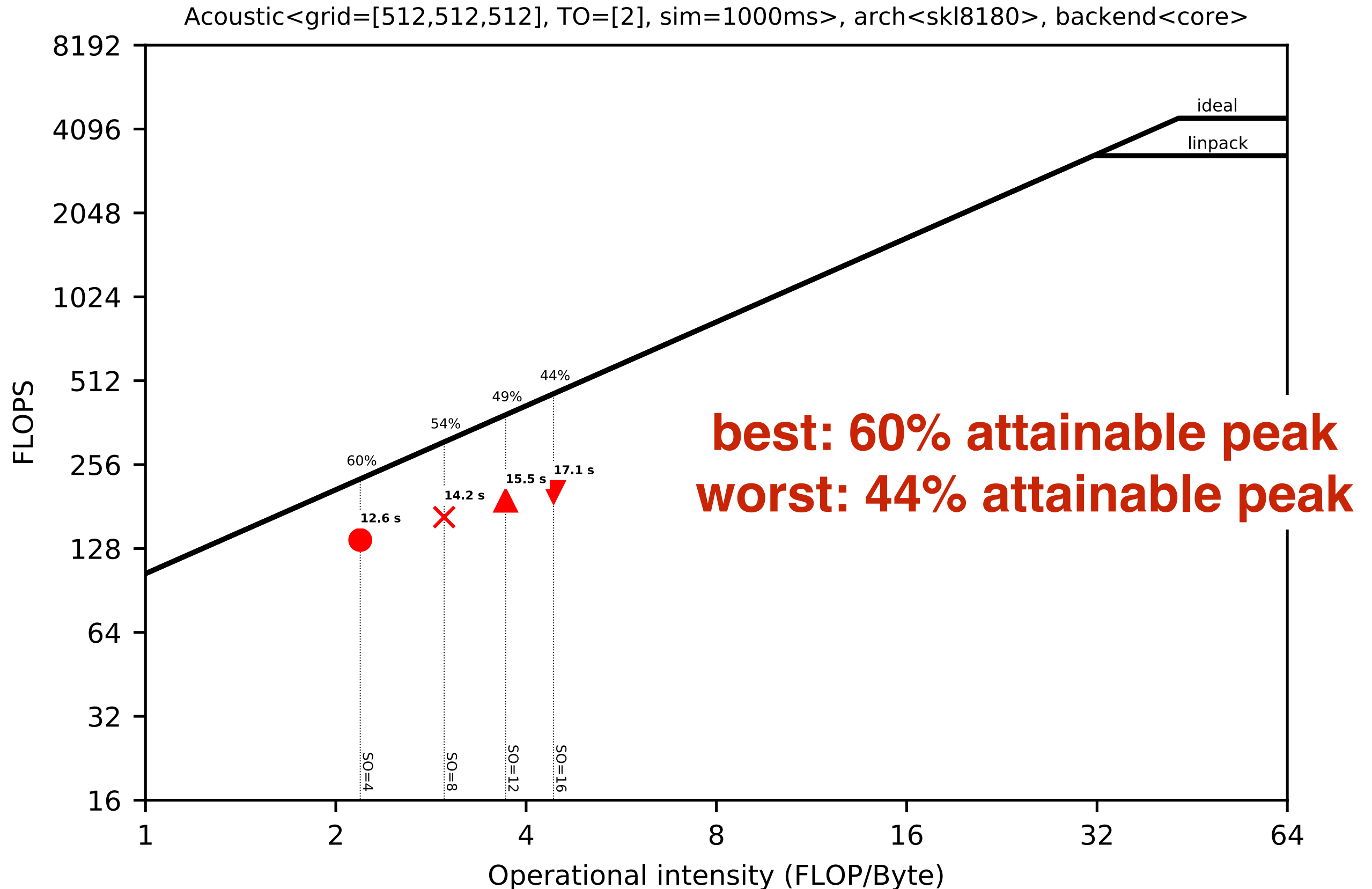
- Amongst the most advanced software to optimize the performance of stencil codes
- Data layout, multi-level blocking, prefetching, vector-folding, ...

Loop-level optimizations

- We started off with standard stuff:
 - Loop blocking (no time loop)
 - Empirical auto-tuning for loop blocking
 - OpenMP parallelism
 - SIMD vectorization via OpenMP 4.5 pragmas
- Our initial thought was:

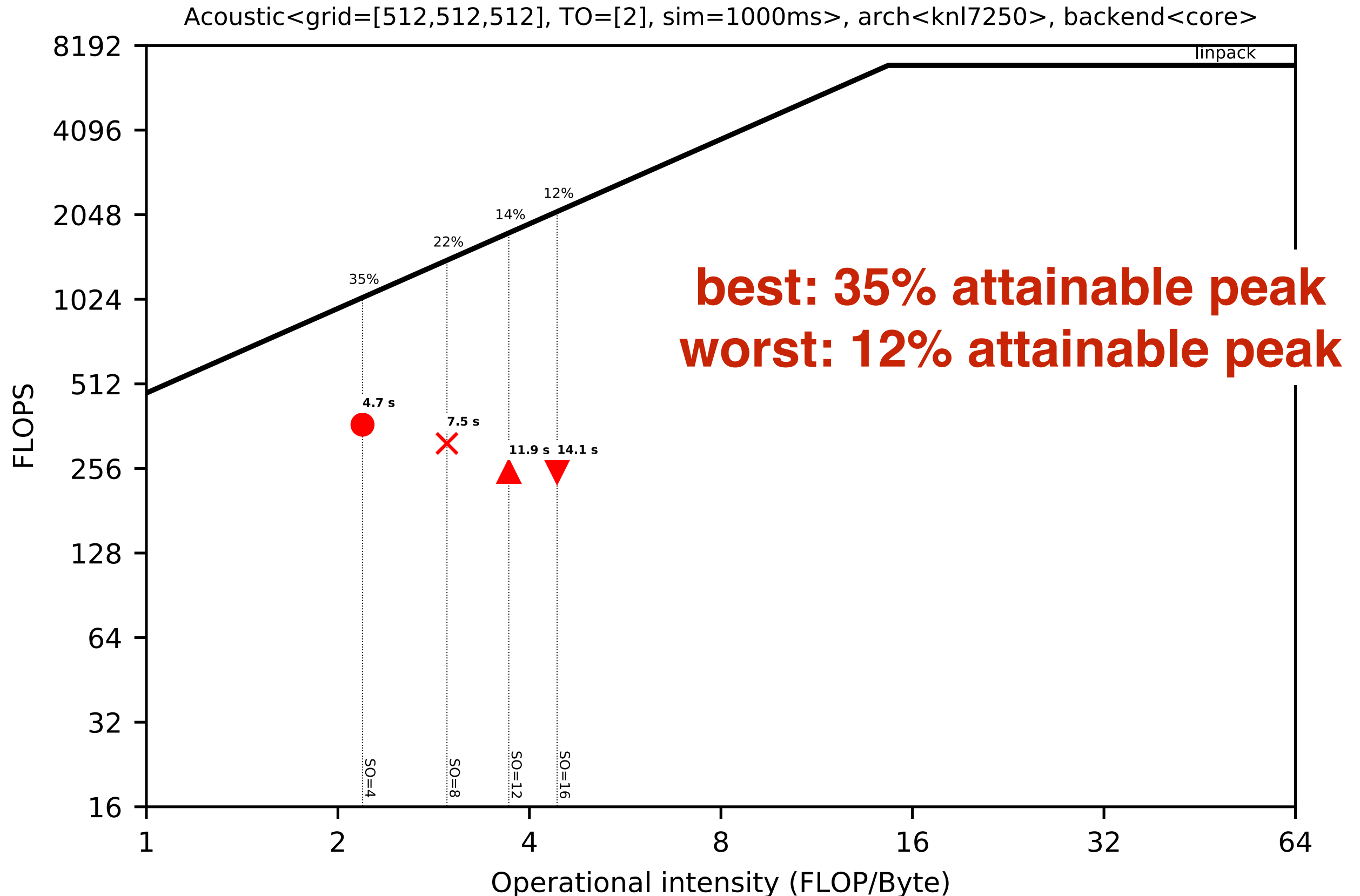
“well, OK, we don’t do anything fancy such as blocking over time, but this plus all the flop-level optimizations should be enough to get some decent performance”

Did it work? Acoustic on Skylake 8180



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Intel internal measurements as of Dec 2017 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4 and/or Intel® Xeon® processor 8108 with 128 GiB DDR. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Did it work? Acoustic on Xeon Phi 7250



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Intel internal measurements as of Dec 2017 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4 and/or Intel® Xeon® processor 8108 with 128 GiB DDR. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

A new backend for Devito: YASK

Now Devito dynamically generates and runs code that “offloads” parts of the computations (i.e., some stencil equations) onto dynamically generated YASK libraries

```
void f0(...) { ... }
void f1(...) { ... }

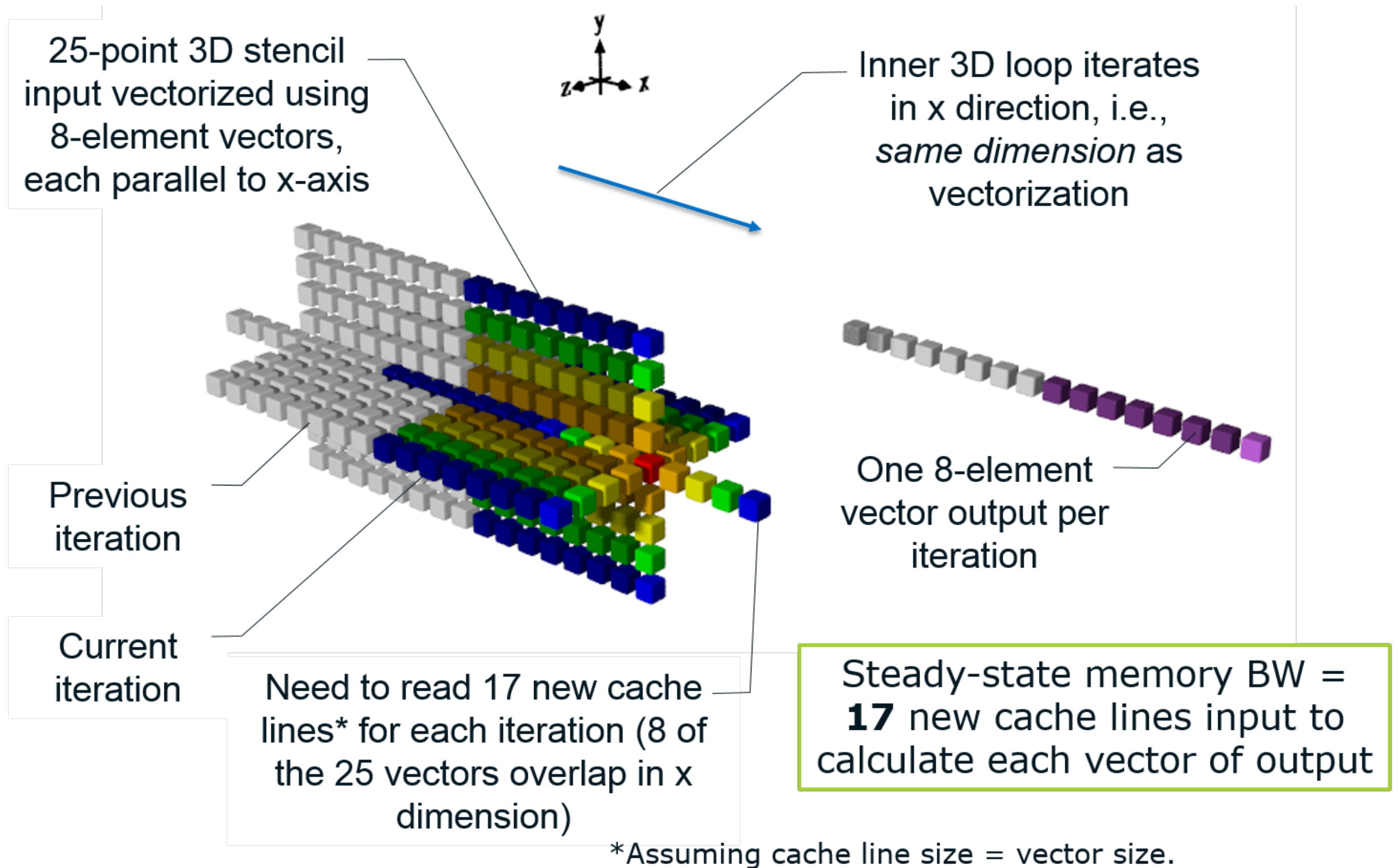
void entry_point_from_python ( ... ) {
    ...
    for t = 0 to num_of_timesteps:
        for i = 0 to Ni: // Devito-optimized loop
            f0( ... );

            yask_lib_0 ( ... ); // jump to YASK-land

        for j = 0 to Nj: // Some other Devito-optimized loop
            f1 ( ... );
    }
```

Towards “vector folding” in YASK

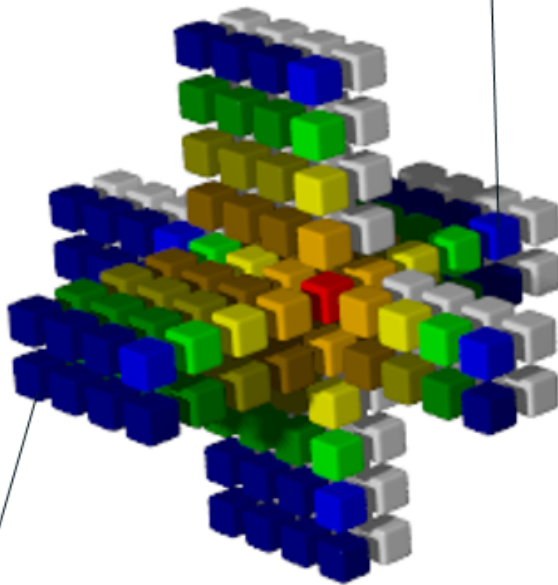
Traditional “ID” vectorization requires lots of bandwidth



Vector folding in YASK

Data layout transformation + cross-loop vectorization to optimize bandwidth usage

25-point 3D stencil
input vectorized using
8-element vectors,
*each containing a 4x2
grid in the x-y plane*



Need to read only 7 new
cache lines for each iteration
(vectors overlap in x-y
dimensions within an iteration
and in z dimension between
iterations)



Inner 3D loop iterates in z
direction, i.e., *perpendicular*
to 2D vector



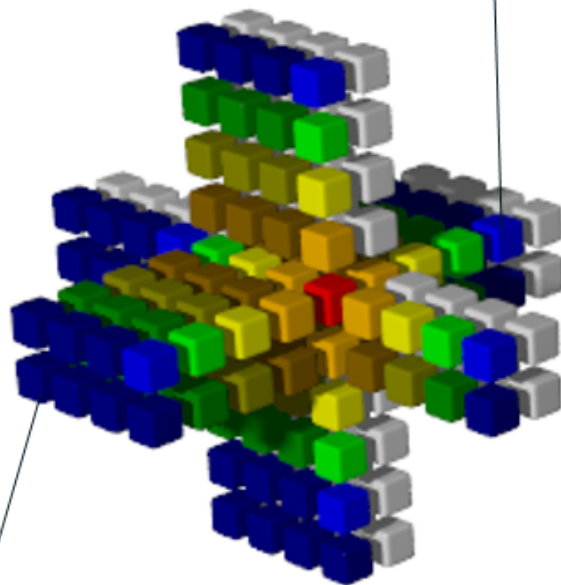
One 8-element (4x2)
vector output per
iteration

Steady-state memory BW = **7**
new cache lines input to
calculate each vector of output:
2.4x lower than in-line

Vector folding in YASK

Data layout transformation + cross-loop vectorization to optimize bandwidth usage

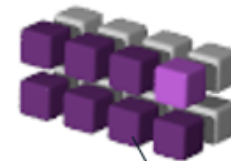
25-point 3D stencil
input vectorized using
8-element vectors,
*each containing a 4x2
grid in the x-y plane*



Need to read only 7 new
cache lines for each iteration
(vectors overlap in x-y
dimensions within an iteration
and in z dimension between
iterations)



Inner 3D loop iterates in z
direction, i.e., *perpendicular*
to 2D vector

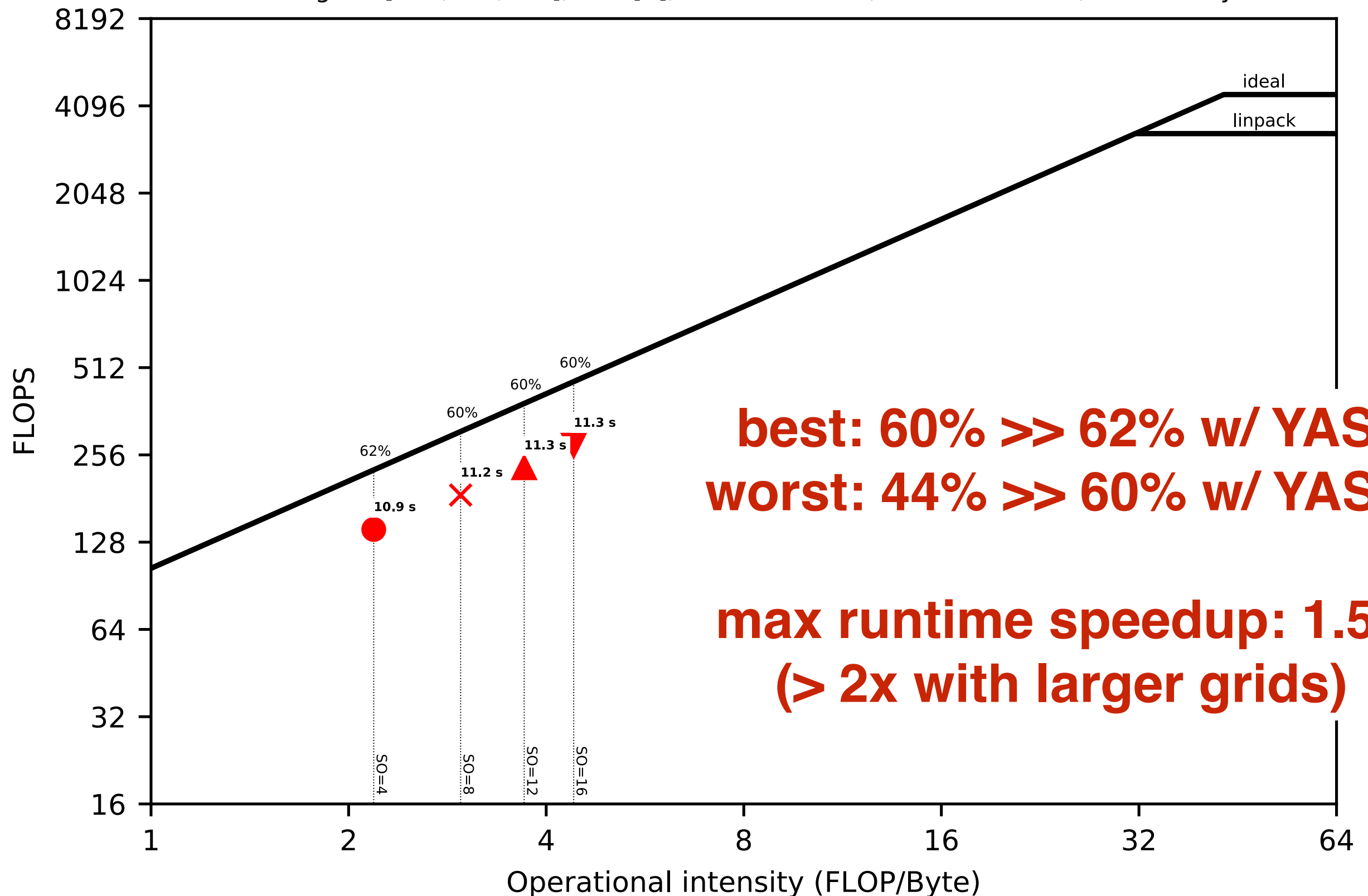


One 8-element (4x2)
vector output per
iteration

There's actually much more:
multi-level tiling
software prefetching
temporal wavefront blocking

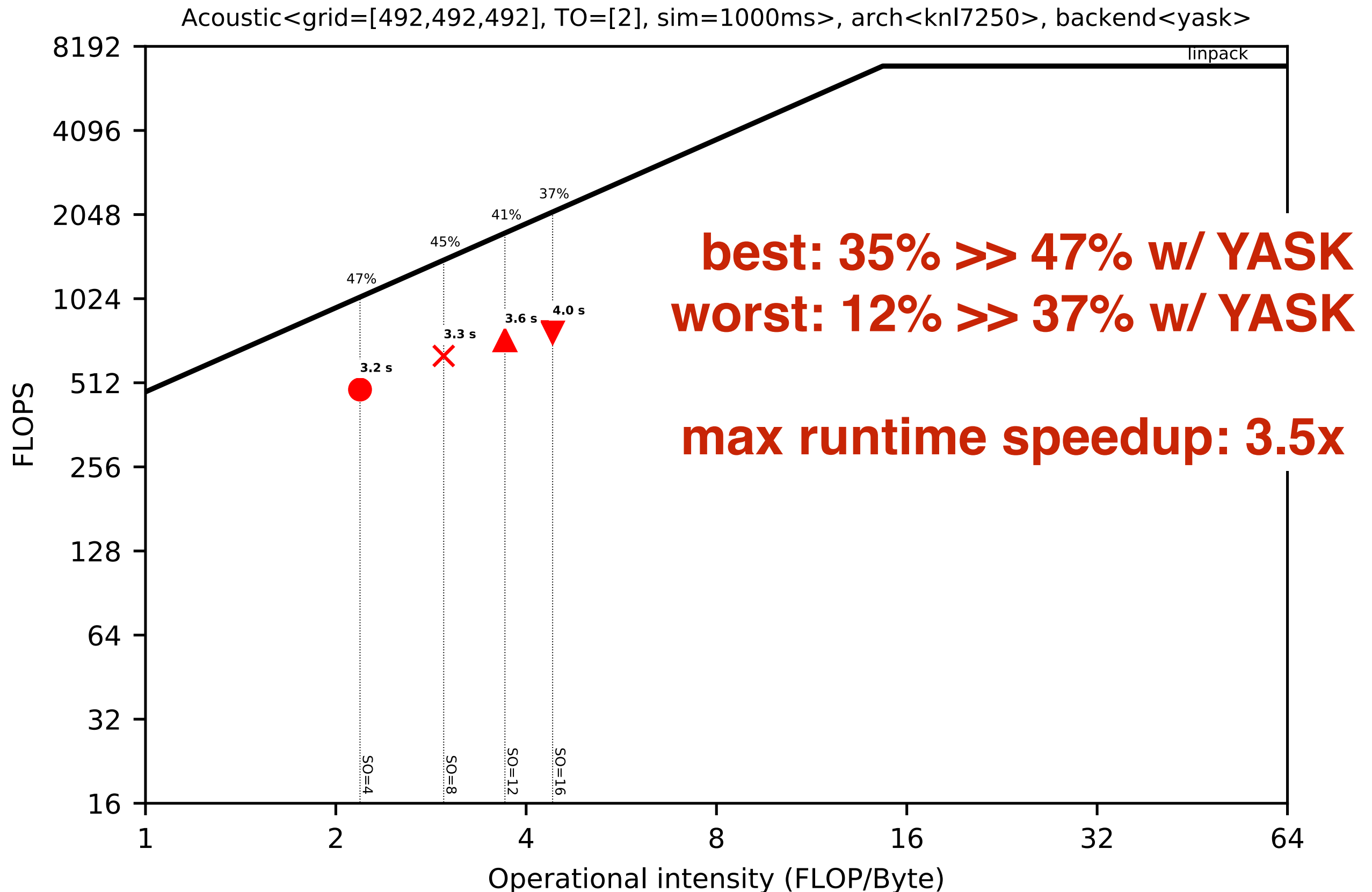
Acoustic on Skylake 8180 with YASK

Acoustic<grid=[492,492,492], TO=[2], sim=1000ms>, arch<skl8180>, backend<yask>



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Intel internal measurements as of Dec 2017 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4 and/or Intel® Xeon® processor 8108 with 128 GiB DDR. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Acoustic on Xeon Phi 7250 with YASK



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Intel internal measurements as of Dec 2017 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4 and/or Intel® Xeon® processor 8108 with 128 GiB DDR. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

More aggressive FLOP reduction strategies

- Common sub-expressions elimination, factorization, ...

More aggressive FLOP reduction strategies

- Common sub-expressions elimination, factorization, ...
- **Cross-iteration redundancies elimination**

```
for i, for j, ...  
    sin(phi[i,j]) + sin(phi[i-1,j-1]) + sin(phi[i+2,j+2])
```

Observations:

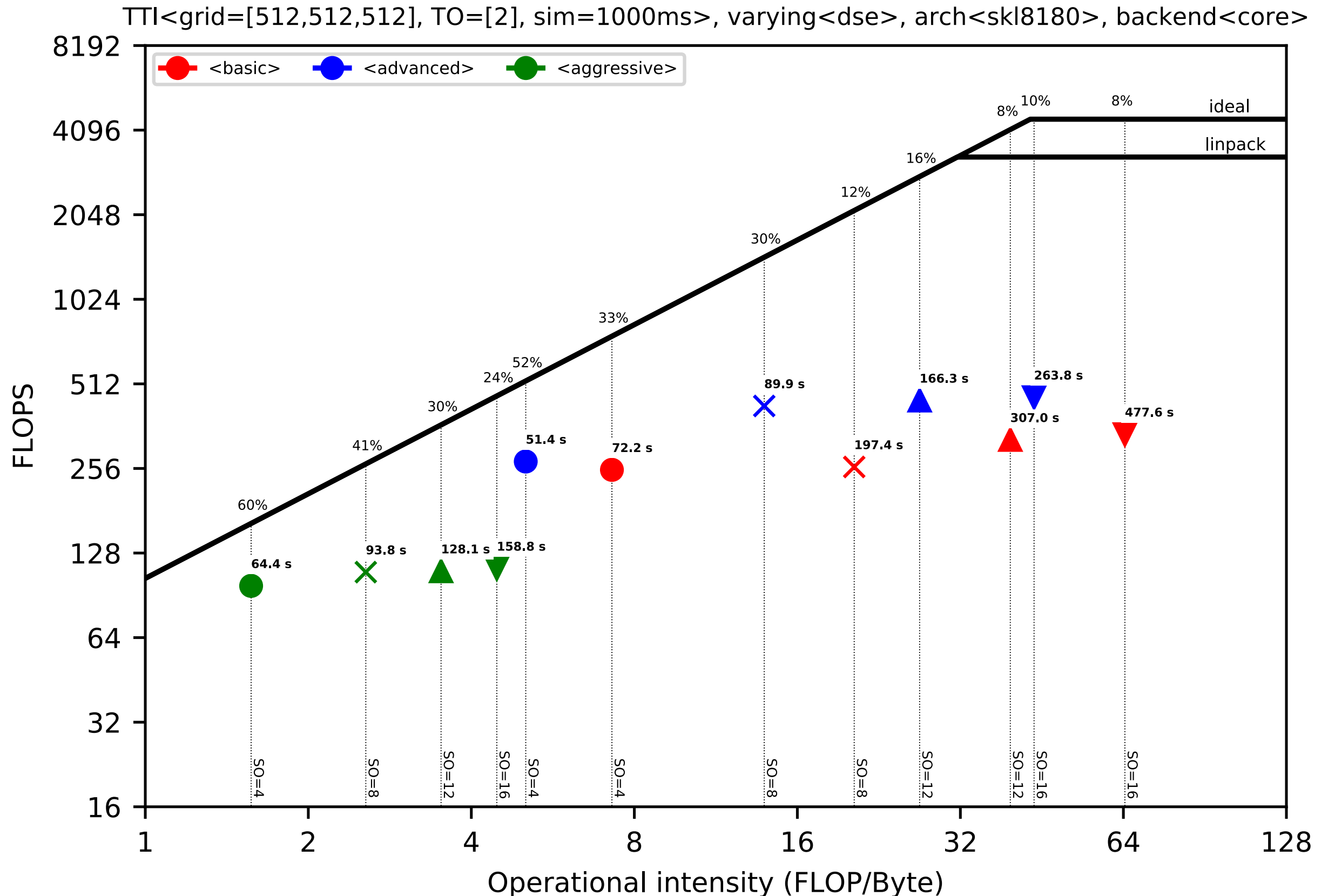
- Same operators (**sin**), same operands (**phi**), same indices (**i, j**)
- Linearly dependent index vectors (**[i, j]**, **[i-1, j-1]**, **[i+2, j+2]**)



```
for i, for j  
    B[i,j] = sin(phi[i,j])  
  
for i, for j, ...  
    B[i,j] + B[i-1,j-1] + B[i+2,j+2]
```

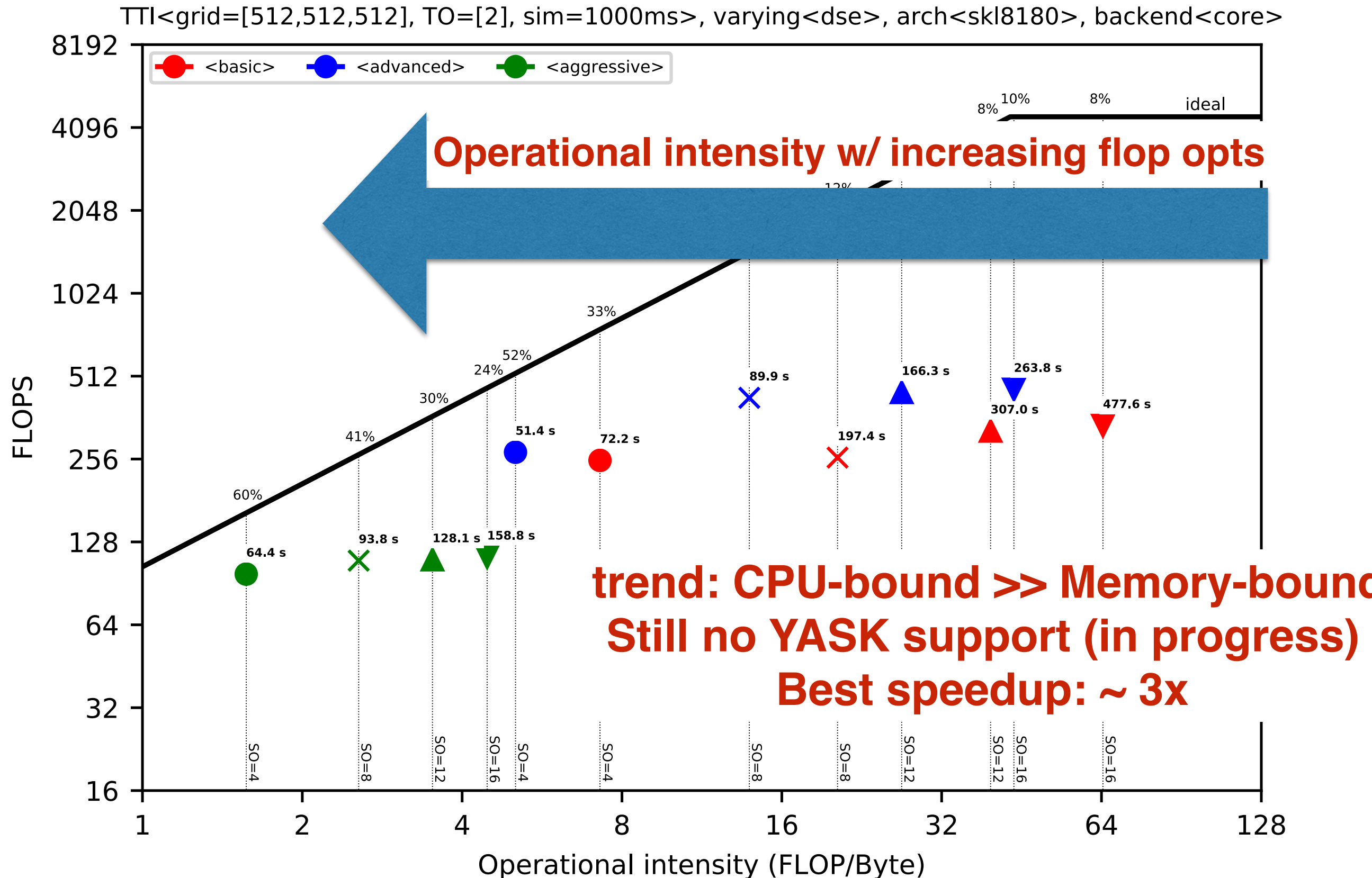
**Trading FLOPs for
storage?**

More aggressive FLOP reduction strategies



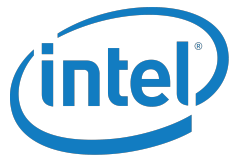
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Intel internal measurements as of Dec 2017 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4 and/or Intel® Xeon® processor 8108 with 128 GiB DDR. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

More aggressive FLOP reduction strategies



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Intel internal measurements as of Dec 2017 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4 and/or Intel® Xeon® processor 8108 with 128 GiB DDR. Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Legal Disclaimer & Optimization Notice



Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit <http://www.intel.com/performance>.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, Xeon, Xeon Phi, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

Optimization notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. (Notice revision #20110804)

Conclusions and resources

- Devito: an efficient and sustainable finite difference DSL
- Driven/inspired by real-world seismic imaging
- Based on actual compiler technology
- Interdisciplinary, interinstitutional research effort

Useful links

- Website: <http://www.devitoproject.org>
- GitHub: <https://github.com/opesci/devito>
- Slack: <https://opesci-slackin.now.sh>

Appendix

Experimentation details

- Architectures
 - Intel® Xeon® Platinum 8180 Processor (“Skylake”, 28 cores)
 - Intel(R) XeonPhi(R) 7250 (68 cores)
 - Quadrant mode (still no support for NUMA)
 - Tried 1, 2, 4 threads per core. Shown best.
- Compiler
 - ICC 18 -xHost -O3
 - -xMIC-AVX512 on Xeon Phi
 - -qopt-zmm-usage=high on Skylake
- OpenMP
 - Single socket
 - Thread pinning via Numactl
- Roofline calculations:
 - Memory bandwidth: STREAM
 - CPU peak: pen & paper
 - Operational intensity: source-level analysis (automated through Devito)

Philosophy: optimizations at the RIGHT level of abstraction

Example: optimizations for FLOPs reduction

$Operator([eqn1, eqn2, ..., eqn3])$

- Runtime constant propagation
- Equation clustering, **NOT** loop fusion
- Symbolic transformations to minimize the operation count of the equations

all based on Python and SymPy; no trace of loops yet!