# VECTORIZATION AND LOCALITY OPTIMIZATIONS FOR SEISMIC IMAGING METHODS THROUGH AUTOMATED CODE GENERATION

***F. Luporini[1], M. Lange[1], M. Louboutin[2], N. Kukreja[1], G.Gorman[1]***

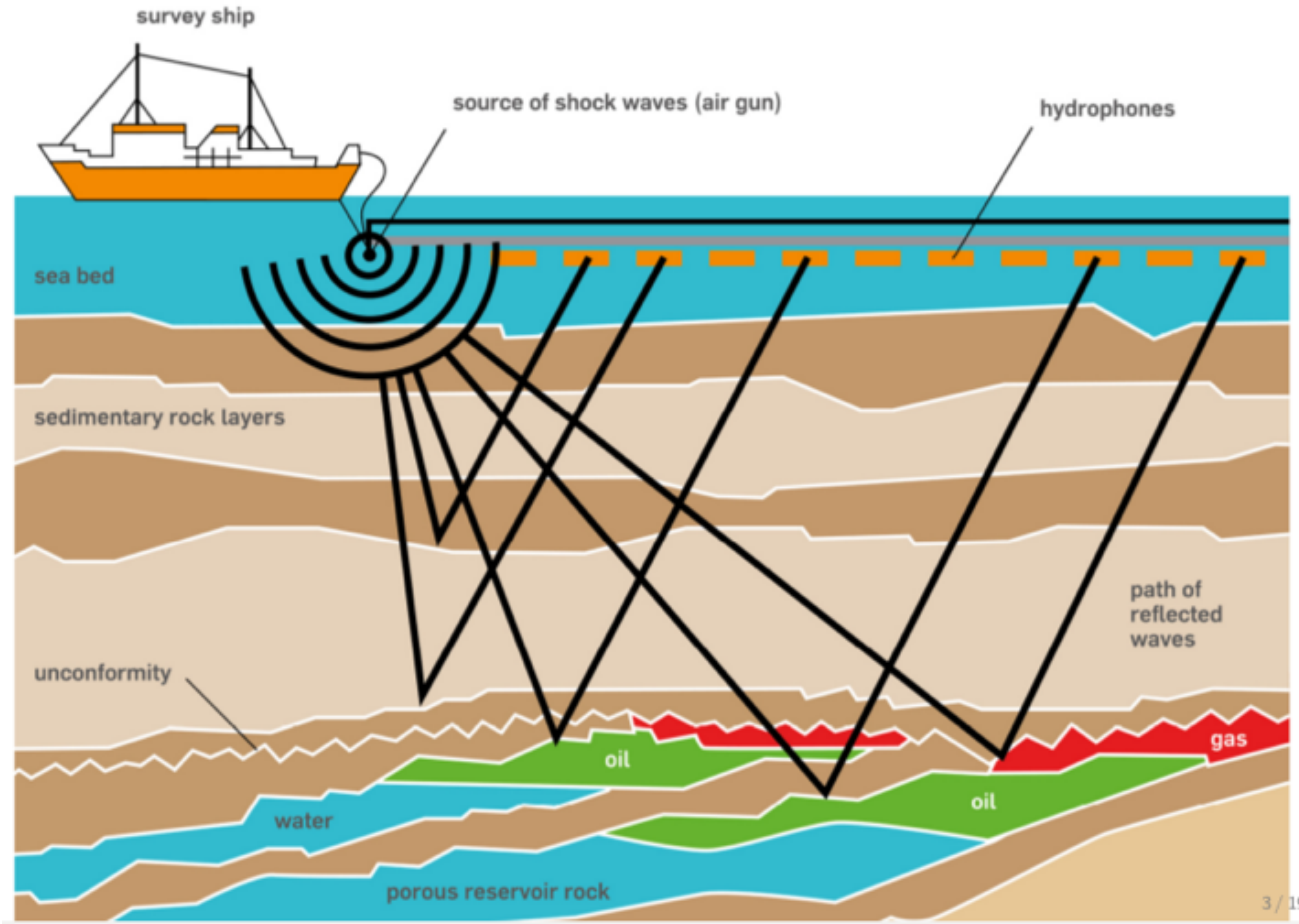
*[1]Imperial College London,*
*[2]The University of British Columbia (Seismic Lab for Imaging and Modeling)*

2017 SIAM Conference on Computational Science and Engineering

**MS: Efficiency of High-Order Methods on the 2nd Generation Intel Xeon Phi Processor**

27/02/2017

# Driving application: inversion problems for seismic imaging




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

- Challenging physics, many variants (e.g., wave equations),
- Big computational cost (wave simulation through subsurface)

# So, it is not the "usual" Poisson equation that we aim to solve…

$$\frac{m}{\rho}\frac{d^2p(x,t)}{dt^2} - (1+2\epsilon)(G_{\bar{x}\bar{x}} + G_{\bar{y}\bar{y}})p(x,t) - \sqrt{(1+2\delta)}G_{\bar{z}\bar{z}}r(x,t) = q,$$

$$\frac{m}{\rho}\frac{d^2r(x,t)}{dt^2} - \sqrt{(1+2\delta)}(G_{\bar{x}\bar{x}} + G_{\bar{y}\bar{y}})p(x,t) - G_{\bar{z}\bar{z}}r(x,t) = q,$$

$$p(.,0) = 0,$$

$$\frac{dp(x,t)}{dt}|_{t=0} = 0,$$

$$r(.,0) = 0,$$

$$\frac{dr(x,t)}{dt}|_{t=0} = 0,$$

(incomplete) specification of the TTI (Tilted Transverse Isotropy) forward operator

$$D_{x1} = cos(\theta)cos(\phi)\frac{d}{dx}\Big|_l + cos(\theta)sin(\phi)\frac{d}{dy} - sin(\theta)\frac{d}{dz}$$

$$D_{x2} = cos(\theta)cos(\phi)\frac{d}{dx} + cos(\theta)sin(\phi)\frac{d}{dy}\Big|_l - sin(\theta)\frac{d}{dz}|_l$$

$$G_{\bar{x}\bar{x}} = \frac{1}{2}\left(D_{x1}^T(\frac{1}{\rho})D_{x1} + D_{x2}^T(\frac{1}{\rho})D_{x2}\right)$$

rotated second order differential operators

# So, it is not the “usual” Poisson equation that we aim to solve…

```
u[t + 2][x][y][z] = 1.0F*((8.85879567828298e-1F*damp[x][y][z] - 2.0F*m[x][y][z])*u[t][x][y][z] + 1.5695652173913F*(1.25e-2F*((-2.5e-2F*v[t + 1][x][y][z] + 2.5e-2F*v[t + 1][x][y][z + 2])*cos(theta[x][y][z + 1]) + (-2.5e-2F*v[t +
1][x][y - 1][z + 1] + 2.5e-2F*v[t + 1][x][y + 1][z + 1])*sin(phi[x][y][z + 1])*sin(theta[x][y][z + 1]) + (-7.5e-2F*v[t + 1][x][y][z + 1] + 1.0e-1F*v[t + 1][x + 1][y][z + 1] - 2.5e-2F*v[t + 1][x + 2][y][z + 1])*sin(theta[x][y][z +
1])*cos(phi[x][y][z + 1]))*cos(theta[x][y][z + 1]) - 1.25e-2F*((2.5e-2F*v[t + 1][x][y][z] - 2.5e-2F*v[t + 1][x][y][z - 2])*cos(theta[x][y][z - 1]) + (-2.5e-2F*v[t + 1][x][y - 1][z - 1] + 2.5e-2F*v[t + 1][x][y + 1][z - 1])*sin(phi[x][y][z -
1])*sin(theta[x][y][z - 1]) + (-7.5e-2F*v[t + 1][x][y][z - 1] + 1.0e-1F*v[t + 1][x + 1][y][z - 1] - 2.5e-2F*v[t + 1][x + 2][y][z - 1])*sin(theta[x][y][z - 1])*cos(phi[x][y][z - 1]))*cos(theta[x][y][z - 1]) + 3.75e-2F*((-2.5e-2F*v[t + 1][x][y]
[z - 1] + 2.5e-2F*v[t + 1][x][y][z + 1])*cos(theta[x][y][z]) + (-2.5e-2F*v[t + 1][x][y - 1][z] + 2.5e-2F*v[t + 1][x][y + 1][z])*sin(phi[x][y][z])*sin(theta[x][y][z]) + (-7.5e-2F*v[t + 1][x][y][z] + 1.0e-1F*v[t + 1][x + 1][y][z] - 2.5e-2F*v[t
+ 1][x + 2][y][z])*sin(theta[x][y][z])*cos(phi[x][y][z]))*sin(theta[x][y][z])*cos(phi[x][y][z]) + 1.25e-2F*((-2.5e-2F*v[t + 1][x - 2][y][z - 1] + 2.5e-2F*v[t + 1][x - 2][y][z + 1])*cos(theta[x - 2][y][z]) + (-2.5e-2F*v[t + 1][x - 2][y - 1][z]
+ 2.5e-2F*v[t + 1][x - 2][y + 1][z])*sin(phi[x - 2][y][z])*sin(theta[x - 2][y][z]) + (-2.5e-2F*v[t + 1][x][y][z] - 7.5e-2F*v[t + 1][x - 2][y][z] + 1.0e-1F*v[t + 1][x - 1][y][z])*sin(theta[x - 2][y][z])*cos(phi[x - 2][y][z]))*sin(theta[x - 2][y]
[z])*cos(phi[x - 2][y][z]) - 5.0e-2F*((-2.5e-2F*v[t + 1][x - 1][y][z - 1] + 2.5e-2F*v[t + 1][x - 1][y][z + 1])*cos(theta[x - 1][y][z]) + (-2.5e-2F*v[t + 1][x - 1][y - 1][z] + 2.5e-2F*v[t + 1][x - 1][y + 1][z])*sin(phi[x - 1][y][z])*sin(theta[x -
1][y][z]) + (1.0e-1F*v[t + 1][x][y][z] - 7.5e-2F*v[t + 1][x - 1][y][z] - 2.5e-2F*v[t + 1][x + 1][y][z])*sin(theta[x - 1][y][z])*cos(phi[x - 1][y][z]))*sin(theta[x - 1][y][z])*cos(phi[x - 1][y][z]) + 1.25e-2F*((-2.5e-2F*v[t + 1][x][y][z] +
2.5e-2F*v[t + 1][x][y + 2][z])*sin(phi[x][y + 1][z])*sin(theta[x][y + 1][z]) + (-2.5e-2F*v[t + 1][x][y + 1][z - 1] + 2.5e-2F*v[t + 1][x][y + 1][z + 1])*cos(theta[x][y + 1][z]) + (-7.5e-2F*v[t + 1][x][y + 1][z] + 1.0e-1F*v[t + 1][x + 1][y
+ 1][z] - 2.5e-2F*v[t + 1][x + 2][y + 1][z])*sin(theta[x][y + 1][z])*cos(phi[x][y + 1][z]))*sin(phi[x][y + 1][z])*sin(theta[x][y + 1][z]) + 1.25e-2F*((-2.5e-2F*v[t + 1][x][y][z] + 2.5e-2F*v[t + 1][x + 2][y][z])*sin(theta[x + 1][y]
[z])*cos(phi[x + 1][y][z]) + (-7.5e-2F*v[t + 1][x + 1][y][z] + 1.0e-1F*v[t + 1][x + 1][y][z + 1] - 2.5e-2F*v[t + 1][x + 1][y][z + 2])*cos(theta[x + 1][y][z]) + (-7.5e-2F*v[t + 1][x + 1][y][z] + 1.0e-1F*v[t + 1][x + 1][y + 1][z] -
2.5e-2F*v[t + 1][x + 1][y + 2][z])*sin(phi[x + 1][y][z])*sin(theta[x + 1][y][z]))*sin(theta[x + 1][y][z])*cos(phi[x + 1][y][z]) - 1.25e-2F*((2.5e-2F*v[t + 1][x][y][z] - 2.5e-2F*v[t + 1][x][y - 2][z])*sin(phi[x][y - 1][z])*sin(theta[x][y - 1]
[z]) + (-2.5e-2F*v[t + 1][x][y - 1][z - 1] + 2.5e-2F*v[t + 1][x][y - 1][z + 1])*cos(theta[x][y - 1][z]) + (-7.5e-2F*v[t + 1][x][y - 1][z] + 1.0e-1F*v[t + 1][x + 1][y - 1][z] - 2.5e-2F*v[t + 1][x + 2][y - 1][z])*sin(theta[x][y - 1]
[z])*cos(phi[x][y - 1][z]))*sin(phi[x][y - 1][z])*sin(theta[x][y - 1][z]) - 1.25e-2F*((2.5e-2F*v[t + 1][x][y][z] - 2.5e-2F*v[t + 1][x - 2][y][z])*sin(theta[x - 1][y][z])*cos(phi[x - 1][y][z]) + (-7.5e-2F*v[t + 1][x - 1][y][z] + 1.0e-1F*v[t + 1][x
- 1][y][z + 1] - 2.5e-2F*v[t + 1][x - 1][y][z + 2])*cos(theta[x - 1][y][z]) + (-7.5e-2F*v[t + 1][x - 1][y][z] + 1.0e-1F*v[t + 1][x - 1][y + 1][z] - 2.5e-2F*v[t + 1][x - 1][y + 2][z])*sin(phi[x - 1][y][z])*sin(theta[x - 1][y][z]))*sin(theta[x - 1]
[y][z])*cos(phi[x - 1][y][z]) + 3.75e-2F*((-2.5e-2F*v[t + 1][x - 1][y][z] + 2.5e-2F*v[t + 1][x + 1][y][z])*sin(theta[x][y][z])*cos(phi[x][y][z]) + (-7.5e-2F*v[t + 1][x][y][z] + 1.0e-1F*v[t + 1][x][y][z + 1] - 2.5e-2F*v[t + 1][x][y][z +
2])*cos(theta[x][y][z]) + (-7.5e-2F*v[t + 1][x][y][z] + 1.0e-1F*v[t + 1][x][y + 1][z] - 2.5e-2F*v[t + 1][x][y + 2][z])*sin(phi[x][y][z])*sin(theta[x][y][z]))*sin(phi[
2.5e-2F*v[t + 1][x + 1][y][z])*sin(theta[x][y][z])*cos(phi[x][y][z]) + (-7.5e-2F*v[t + 1][x][y][z] + 1.0e-1F*v[t + 1][x][y][z + 1] - 2.5e-2F*v[t + 1][x][y][z + 2])*co
1][z] - 2.5e-2F*v[t + 1][x][y + 2][z])*sin(phi[x][y][z])*sin(theta[x][y][z]))*cos(theta[x][y][z]) + 1.25e-2F*((-2.5e-2F*v[t + 1][x - 1][y][z - 2] + 2.5e-2F*v[t + 1][x -
+ 1][x][y][z] - 7.5e-2F*v[t + 1][x][y][z - 2] + 1.0e-1F*v[t + 1][x][y][z - 1])*cos(theta[x][y][z - 2]) + (-7.5e-2F*v[t + 1][x][y][z - 2] + 1.0e-1F*v[t + 1][x][y + 1][z
[y][z - 2]))*cos(theta[x][y][z - 2]) - 5.0e-2F*((-2.5e-2F*v[t + 1][x - 1][y][z - 1] + 2.5e-2F*v[t + 1][x + 1][y][z - 1])*sin(theta[x][y][z - 1])*cos(phi[x][y][z - 1]) +
1][x][y][z + 1])*cos(theta[x][y][z - 1]) + (-7.5e-2F*v[t + 1][x][y][z - 1] + 1.0e-1F*v[t + 1][x][y + 1][z - 1] - 2.5e-2F*v[t + 1][x][y + 2][z - 1])*sin(phi
1.25e-2F*((-2.5e-2F*v[t + 1][x - 1][y - 2][z] + 2.5e-2F*v[t + 1][x + 1][y - 2][z])*sin(theta[x][y - 2][z])*cos(phi[x][y - 2][z]) + (-2.5e-2F*v[t + 1][x][y][z] - 7.5e-
[z])*sin(theta[x][y - 2][z]) + (-7.5e-2F*v[t + 1][x][y - 2][z] + 1.0e-1F*v[t + 1][x][y - 2][z + 1] - 2.5e-2F*v[t + 1][x][y - 2][z + 2])*cos(theta[x][y - 2][z]))*sin(phi[
- 1][z] + 2.5e-2F*v[t + 1][x + 1][y - 1][z])*sin(theta[x][y - 1][z])*cos(phi[x][y - 1][z]) + (1.0e-1F*v[t + 1][x][y][z] - 7.5e-2F*v[t + 1][x][y - 1][z] - 2.5e-2F*v[t + 1
+ 1][x][y - 1][z] + 1.0e-1F*v[t + 1][x][y - 1][z + 1] - 2.5e-2F*v[t + 1][x][y - 1][z + 2])*cos(theta[x][y - 1][z]))*sin(phi[x][y - 1][z])*sin(theta[x][y - 1][z]))*delta[x
2.5e-2F*u[t + 1][x][y + 2][z])*cos(phi[x][y + 1][z]) + (-7.5e-2F*u[t + 1][x][y + 1][z] + 1.0e-1F*u[t + 1][x + 1][y + 1][z] - 2.5e-2F*u[t + 1][x + 2][y + 1][z])*sin
[x][y][z] + 2.5e-2F*u[t + 1][x + 2][y][z])*sin(phi[x + 1][y][z]) - (-7.5e-2F*u[t + 1][x + 1][y][z] + 1.0e-1F*u[t + 1][x + 1][y + 1][z] - 2.5e-2F*u[t + 1][x + 1][y + 2
(2.5e-2F*u[t + 1][x][y][z] - 2.5e-2F*u[t + 1][x][y - 2][z])*cos(phi[x][y - 1][z]) + (-7.5e-2F*u[t + 1][x][y - 1][z] + 1.0e-1F*u[t + 1][x + 1][y - 1][z] - 2.5e-2F*
1.25e-2F*((2.5e-2F*u[t + 1][x][y][z] - 2.5e-2F*u[t + 1][x - 2][y][z])*sin(phi[x - 1][y][z]) - (-7.5e-2F*u[t + 1][x - 1][y][z] + 1.0e-1F*u[t + 1][x - 1][y + 1][z] - 2.5e
+ 3.75e-2F*(-(-2.5e-2F*u[t + 1][x][y - 1][z] + 2.5e-2F*u[t + 1][x][y + 1][z])*cos(phi[x][y][z]) + (-7.5e-2F*u[t + 1][x][y][z] + 1.0e-1F*u[t + 1][x + 1][y][z] -
1.25e-2F*(-(-2.5e-2F*u[t + 1][x - 2][y - 1][z] + 2.5e-2F*u[t + 1][x - 2][y + 1][z])*cos(phi[x - 2][y][z]) + (-2.5e-2F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x - 2][y][
[z]) - 3.75e-2F*((-2.5e-2F*u[t + 1][x - 1][y][z] + 2.5e-2F*u[t + 1][x + 1][y][z])*sin(phi[x][y][z]) - (-7.5e-2F*u[t + 1][x][y][z] + 1.0e-1F*u[t + 1][x][y + 1][z] -
1.25e-2F*((-2.5e-2F*u[t + 1][x - 1][y - 2][z] + 2.5e-2F*u[t + 1][x + 1][y - 2][z])*sin(phi[x][y - 2][z]) - (-2.5e-2F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x][y - 2][z] +
- 5.0e-2F*(-(-2.5e-2F*u[t + 1][x - 1][y - 1][z] + 2.5e-2F*u[t + 1][x - 1][y + 1][z])*cos(phi[x - 1][y][z]) + (1.0e-1F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x - 1][y][z]
+ 5.0e-2F*((-2.5e-2F*u[t + 1][x - 1][y - 1][z] + 2.5e-2F*u[t + 1][x + 1][y - 1][z])*sin(phi[x][y - 1][z]) - (1.0e-1F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x][y - 1][z] -
- 1.25e-2F*(-(-2.5e-2F*u[t + 1][x][y][z] + 2.5e-2F*u[t + 1][x][y][z + 2])*sin(theta[x][y][z + 1]) + (-2.5e-2F*u[t + 1][x][y - 1][z + 1] + 2.5e-2F*u[t + 1][x][y + 1
1][x][y][z + 1] + 1.0e-1F*u[t + 1][x + 1][y][z + 1] - 2.5e-2F*u[t + 1][x + 2][y][z + 1])*cos(phi[x][y][z + 1])*cos(theta[x][y][z + 1]))*sin(theta[x][y][z + 1])
2])*sin(theta[x][y][z - 1]) + (-2.5e-2F*u[t + 1][x][y - 1][z - 1] + 2.5e-2F*u[t + 1][x][y + 1][z - 1])*sin(phi[x][y][z - 1])*cos(theta[x][y][z - 1]) + (-7.5e-2F*u[t + 1
2][y][z - 1])*cos(phi[x][y][z - 1])*cos(theta[x][y][z - 1]))*sin(theta[x][y][z - 1]) + 3.75e-2F*(-(-2.5e-2F*u[t + 1][x][y][z - 1] + 2.5e-2F*u[t + 1][x][y][z + 1])*sin(th
1][z])*sin(phi[x][y][z])*cos(theta[x][y][z]) + (-7.5e-2F*u[t + 1][x][y][z] + 1.0e-1F*u[t + 1][x + 1][y][z] - 2.5e-2F*u[t + 1][x + 2][y][z])*cos(phi[x][y][z])*cos(theta[x][y][z]))*cos(phi[x][y][z])*cos(theta[x][y][z]) + 1.25e-2F*(-
(-2.5e-2F*u[t + 1][x - 2][y][z - 1] + 2.5e-2F*u[t + 1][x - 2][y][z + 1])*sin(theta[x - 2][y][z]) + (-2.5e-2F*u[t + 1][x - 2][y - 1][z] + 2.5e-2F*u[t + 1][x - 2][y + 1][z])*sin(phi[x - 2][y][z])*cos(theta[x - 2][y][z]) + (-2.5e-2F*u[t + 1][x][y]
[z] - 7.5e-2F*u[t + 1][x - 2][y][z] + 1.0e-1F*u[t + 1][x - 1][y][z])*cos(phi[x - 2][y][z])*cos(theta[x - 2][y][z]))*cos(phi[x - 2][y][z])*cos(theta[x - 2][y][z]) - 5.0e-2F*(-(-2.5e-2F*u[t + 1][x - 1][y][z - 1] + 2.5e-2F*u[t + 1][x - 1][y][z +
1])*sin(theta[x - 1][y][z]) + (-2.5e-2F*u[t + 1][x - 1][y - 1][z] + 2.5e-2F*u[t + 1][x - 1][y + 1][z])*sin(phi[x - 1][y][z])*cos(theta[x - 1][y][z]) + (1.0e-1F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x - 1][y][z] - 2.5e-2F*u[t + 1][x + 1][y]
[z])*cos(phi[x - 1][y][z])*cos(theta[x - 1][y][z]))*cos(phi[x - 1][y][z])*cos(theta[x - 1][y][z]) + 1.25e-2F*((-2.5e-2F*u[t + 1][x][y][z] + 2.5e-2F*u[t + 1][x][y + 2][z])*sin(phi[x][y + 1][z])*cos(theta[x][y + 1][z]) - (-2.5e-2F*u[t + 1][x]
[y + 1][z - 1] + 2.5e-2F*u[t + 1][x][y + 1][z + 1])*sin(theta[x][y + 1][z]) + (-7.5e-2F*u[t + 1][x][y + 1][z] + 1.0e-1F*u[t + 1][x + 1][y + 1][z] - 2.5e-2F*u[t + 1][x + 2][y + 1][z])*cos(phi[x][y + 1][z])*cos(theta[x][y + 1
[z]))*sin(phi[x][y + 1][z])*cos(theta[x][y + 1][z]) + 1.25e-2F*((-2.5e-2F*u[t + 1][x][y][z] + 2.5e-2F*u[t + 1][x + 2][y][z])*cos(phi[x + 1][y][z])*cos(theta[x + 1][y][z]) - (-7.5e-2F*u[t + 1][x + 1][y][z] + 1.0e-1F*u[t + 1][x + 1][y][z +
1] - 2.5e-2F*u[t + 1][x + 1][y][z + 2])*sin(theta[x + 1][y][z]) + (-7.5e-2F*u[t + 1][x + 1][y][z] + 1.0e-1F*u[t + 1][x + 1][y + 1][z] - 2.5e-2F*u[t + 1][x + 1][y + 2][z])*sin(phi[x + 1][y][z])*cos(theta[x + 1][y][z]))*cos(phi[x + 1][y]
[z])*cos(theta[x + 1][y][z]) - 1.25e-2F*((2.5e-2F*u[t + 1][x][y][z] - 2.5e-2F*u[t + 1][x][y - 2][z])*sin(phi[x][y - 1][z])*cos(theta[x][y - 1][z]) - (-2.5e-2F*u[t + 1][x][y - 1][z - 1] + 2.5e-2F*u[t + 1][x][y - 1][z + 1])*sin(theta[x][y - 1][z])
+ (-7.5e-2F*u[t + 1][x][y - 1][z] + 1.0e-1F*u[t + 1][x + 1][y - 1][z] - 2.5e-2F*u[t + 1][x + 2][y - 1][z])*cos(phi[x][y - 1][z])*cos(theta[x][y - 1][z]))*sin(phi[x][y - 1][z])*cos(theta[x][y - 1][z]) - 1.25e-2F*((2.5e-2F*u[t + 1][x][y][z] -
2.5e-2F*u[t + 1][x - 2][y][z])*cos(phi[x - 1][y][z])*cos(theta[x - 1][y][z]) - (-7.5e-2F*u[t + 1][x - 1][y][z] + 1.0e-1F*u[t + 1][x - 1][y][z + 1] - 2.5e-2F*u[t + 1][x - 1][y][z + 2])*sin(theta[x - 1][y][z]) + (-7.5e-2F*u[t + 1][x - 1][y][z] +
1.0e-1F*u[t + 1][x - 1][y + 1][z] - 2.5e-2F*u[t + 1][x - 1][y + 2][z])*sin(phi[x - 1][y][z])*cos(theta[x - 1][y][z]))*cos(phi[x - 1][y][z])*cos(theta[x - 1][y][z]) + 3.75e-2F*((-2.5e-2F*u[t + 1][x - 1][y][z] + 2.5e-2F*u[t + 1][x + 1][y]
[z])*cos(phi[x][y][z])*cos(theta[x][y][z]) - (-7.5e-2F*u[t + 1][x][y][z] + 1.0e-1F*u[t + 1][x][y][z + 1] - 2.5e-2F*u[t + 1][x][y][z + 2])*sin(theta[x][y][z]) + (-7.5e-2F*u[t + 1][x][y][z] + 1.0e-1F*u[t + 1][x][y + 1][z] - 2.5e-2F*u[t + 1][x]
[y + 2][z])*sin(phi[x][y][z])*cos(theta[x][y][z]))*sin(phi[x][y][z])*cos(theta[x][y][z]) - 3.75e-2F*((-2.5e-2F*u[t + 1][x - 1][y][z] + 2.5e-2F*u[t + 1][x + 1][y][z])*cos(phi[x][y][z])*cos(theta[x][y][z]) - (-7.5e-2F*u[t + 1][x][y][z] +
1.0e-1F*u[t + 1][x][y][z + 1] - 2.5e-2F*u[t + 1][x][y][z + 2])*sin(theta[x][y][z]) + (-7.5e-2F*u[t + 1][x][y][z] + 1.0e-1F*u[t + 1][x][y + 1][z] - 2.5e-2F*u[t + 1][x][y + 2][z])*sin(phi[x][y][z])*cos(theta[x][y][z]))*sin(theta[x][y][z]) -
1.25e-2F*((-2.5e-2F*u[t + 1][x - 1][y][z - 2] + 2.5e-2F*u[t + 1][x + 1][y][z - 2])*cos(phi[x][y][z - 2])*cos(theta[x][y][z - 2]) - (-2.5e-2F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x][y][z - 2] + 1.0e-1F*u[t + 1][x][y][z - 1])*sin(theta[x][y][z -
2]) + (-7.5e-2F*u[t + 1][x][y][z - 2] + 1.0e-1F*u[t + 1][x][y + 1][z - 2] - 2.5e-2F*u[t + 1][x][y + 2][z - 2])*sin(phi[x][y][z - 2])*cos(theta[x][y][z - 2]))*sin(theta[x][y][z - 2]) + 5.0e-2F*((-2.5e-2F*u[t + 1][x - 1][y][z - 1] + 2.5e-2F*u[t
+ 1][x + 1][y][z - 1])*cos(phi[x][y][z - 1])*cos(theta[x][y][z - 1]) - (1.0e-1F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x][y][z - 1] - 2.5e-2F*u[t + 1][x][y][z + 1])*sin(theta[x][y][z - 1]) + (-7.5e-2F*u[t + 1][x][y][z - 1] + 1.0e-1F*u[t + 1][x][y
+ 1][z - 1] - 2.5e-2F*u[t + 1][x][y + 2][z - 1])*sin(phi[x][y][z - 1])*cos(theta[x][y][z - 1]))*sin(theta[x][y][z - 1]) + 1.25e-2F*((-2.5e-2F*u[t + 1][x - 1][y - 2][z] + 2.5e-2F*u[t + 1][x + 1][y - 2][z])*cos(phi[x][y - 2][z])*cos(theta[x][y -
2][z]) + (-2.5e-2F*u[t + 1][x][y][z] - 7.5e-2F*u[t + 1][x][y - 2][z] + 1.0e-1F*u[t + 1][x][y - 1][z])*sin(phi[x][y - 2][z])*cos(theta[x][y - 2][z]) - (-7.5e-2F*u[t + 1][x][y - 2][z] + 1.0e-1F*u[t + 1][x][y - 2][z + 1] - 2.5e-2F*u[t + 1][x][y -
2][z + 2])*sin(theta[x][y - 2][z]))*sin(phi[x][y - 2][z])*cos(theta[x][y - 2][z]) - 5.0e-2F*((-2.5e-2F*u[t + 1][x - 1][y - 1][z] + 2.5e-2F*u[t + 1][x + 1][y - 1][z])*cos(phi[x][y - 1][z])*cos(theta[x][y - 1][z]) + (1.0e-1F*u[t + 1][x][y][z]
```

Snapshot of a C implementation of a finite difference scheme (2nd order in space and time) for TTI in a cubic grid

# Why we need HPC implementations

- Huge number of floating-point operations: more than 6000 per loop iteration for a 16th order TTI operator
- Realistic 3D grids may have more than $10^9$ grid points (e.g., 2.82 billions in SEAM benchmark)
- Often more than 3000 time steps
- Two operators: forward + adjoint
- Usually 30000 shots ("MPI level")
- Around 15 Full-Waveform Inversion (FWI) iterations
- $\approx$ 6000 x 2.82 x$10^9$ x 3000 x 2 x 30000 x 15 $\approx$ 46 billion TFLOPs
- $\approx$100 wall-clock days executing on the TACC Stampede (assuming Linpack-level performance)

# Devito: automated high performance finite difference

- **Real-world seismic imaging:**
  - Complex inversion methods (e.g., FWI)
  - Change of physics (e.g., acoustic, VTI, TTI — accuracy ⇒ complexity)
  - Change of discretization (FD schemes, up to very high order)
  - Boundary conditions, data acquisition, source/receivers modeling…
  - …
- **Devito (∈ OPESCI)**
  - Not "Yet another DSL for toy problems": language + escape hatches
  - Interdisciplinary research effort
  - Used by geophysicists to write inversion operators
  - Based on actual compiler technology (you can write your own passes!)
- **This talk: the Devito compiler, its performance optimizations, application to real-world Acoustic and TTI operators**

# The compilation flow: from symbolics to HPC code

Symbolic equations

Data objects

SymPy

NumPy

python™

---

Front-end

DSE - Devito Symbolic Engine

Loop scheduler

DLE - Devito Loop Engine

Declarations, headers, …

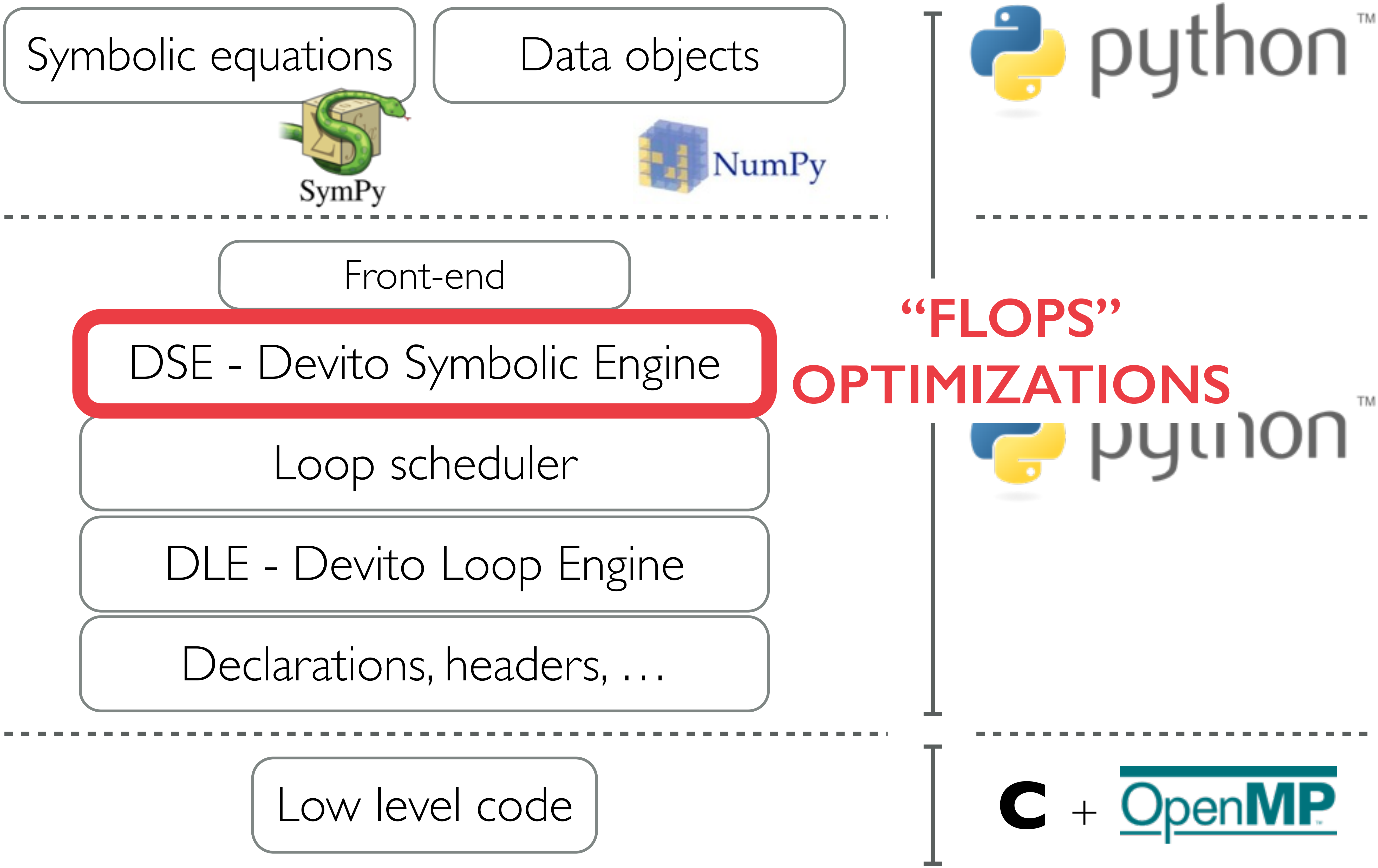
python™

---

Low level code

C + OpenMP

# The compilation flow: from symbolics to HPC code



# The compilation flow: from symbolics to HPC code

Symbolic equations

Data objects

Front-end

DSE - Devito Symbolic Engine

Loop scheduler

DLE - Devito Loop Engine

Declarations, headers, …

Low level code

“FLOPS” OPTIMIZATIONS

“MEMORY” OPTIMIZATIONS

C + OpenMP

# Devito Symbolic Engine

A sequence of compiler passes to reduce FLOPS (no loops at this stage!)

# Devito Symbolic Engine

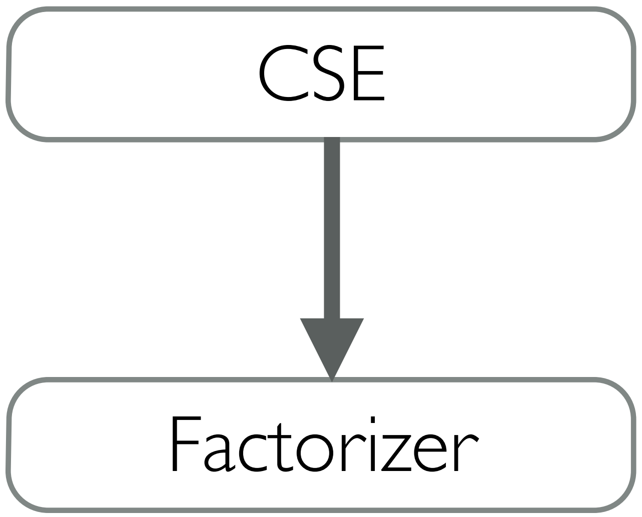
A sequence of compiler passes to reduce FLOPS (no loops at this stage!)

CSE

- Common sub-expressions elimination
  - C compilers do it already… but necessary for symbolic processing and compilation speed

# Devito Symbolic Engine

A sequence of compiler passes to reduce FLOPS (no loops at this stage!)

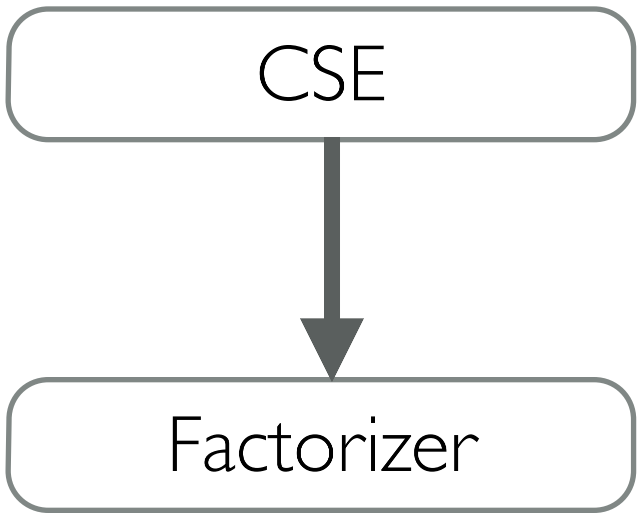



- Common sub-expressions elimination
  - C compilers do it already… but necessary for symbolic processing and compilation speed

- Heuristic re-factorization of recurrent terms
  - E.g., finite difference weights: 0.3*a + … + 0.3*b => 0.3*(a+b)
  - Many possibilities (doesn't leverage domain properties yet!)

# Devito Symbolic Engine

A sequence of compiler passes to reduce FLOPS (no loops at this stage!)




- Common sub-expressions elimination
  - C compilers do it already… but necessary for symbolic processing and compilation speed

- Heuristic re-factorization of recurrent terms
  - E.g., finite difference weights: 0.3*a + … + 0.3*b => 0.3*(a+b)
  - Many possibilities (doesn't leverage domain properties yet!)

**Factorizer impact:**

**TTI, space order 4: 1100 → 950**

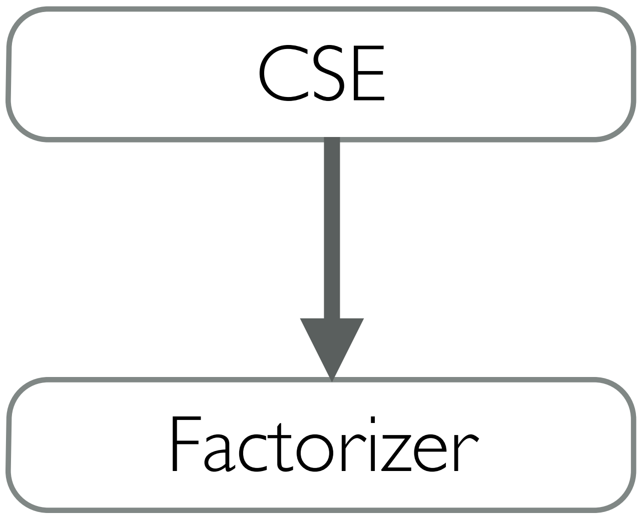
**TTI, space order 8: 2380 → 2120**

**TTI, space order 12: 4240 → 3760**

**TTI, space order 16: 6680 → 5760**

# Devito Symbolic Engine

A sequence of compiler passes to reduce FLOPS (no loops at this stage!)

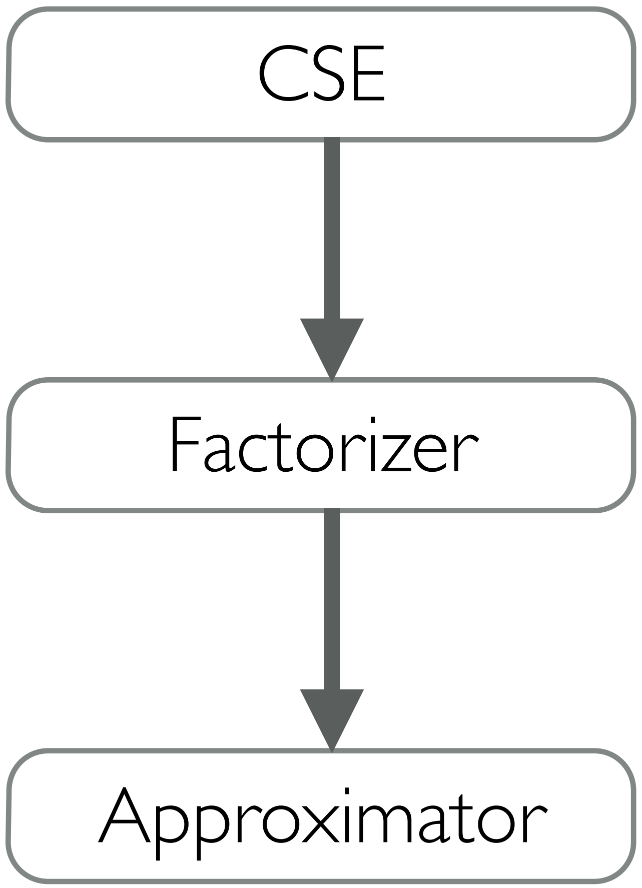



- Common sub-expressions elimination
  - C compilers do it already… but necessary for symbolic processing and compilation speed

- Heuristic re-factorization of recurrent terms
  - E.g., finite difference weights: 0.3*a + … + 0.3*b => 0.3*(a+b)
  - Many possibilities (doesn't leverage domain properties yet!)

# Devito Symbolic Engine

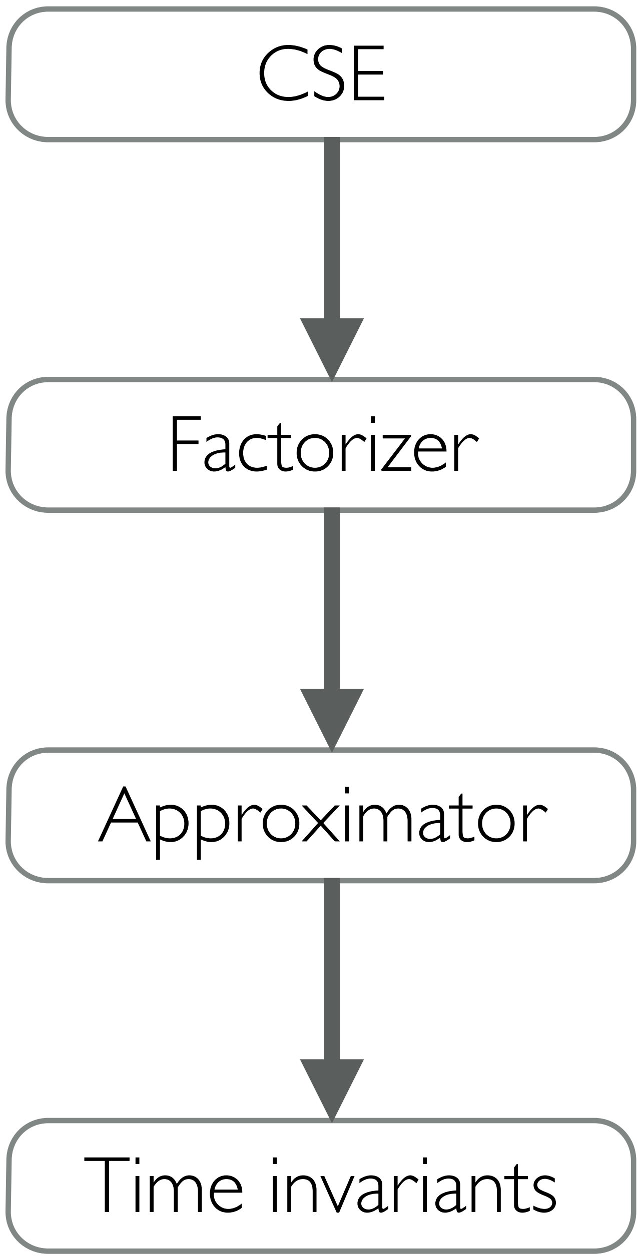
A sequence of compiler passes to reduce FLOPS (no loops at this stage!)




- Common sub-expressions elimination
  - C compilers do it already… but necessary for symbolic processing and compilation speed
- Heuristic re-factorization of recurrent terms
  - E.g., finite difference weights: 0.3*a + … + 0.3*b => 0.3*(a+b)
  - Many possibilities (doesn't leverage domain properties yet!)
- Trigonometric functions are evil
  - Extremely costly
  - Therefore, approximation with e.g. Taylor polynomials
  - Vectorizable, quicker to compile

# Devito Symbolic Engine

A sequence of compiler passes to reduce FLOPS (no loops at this stage!)




- Common sub-expressions elimination
  - C compilers do it already… but necessary for symbolic processing and compilation speed
- Heuristic re-factorization of recurrent terms
  - E.g., finite difference weights: 0.3*a + … + 0.3*b => 0.3*(a+b)
  - Many possibilities (doesn't leverage domain properties yet!)
- Trigonometric functions are evil
  - Extremely costly
  - Therefore, approximation with e.g. Taylor polynomials
  - Vectorizable, quicker to compile
- Heuristic hoisting of time-invariant quantities
  - Currently, only Approximator's output (but pass is general)…
  - … to minimize extra memory consumption
  - This is enhanced by the "aliases detection algorithm"

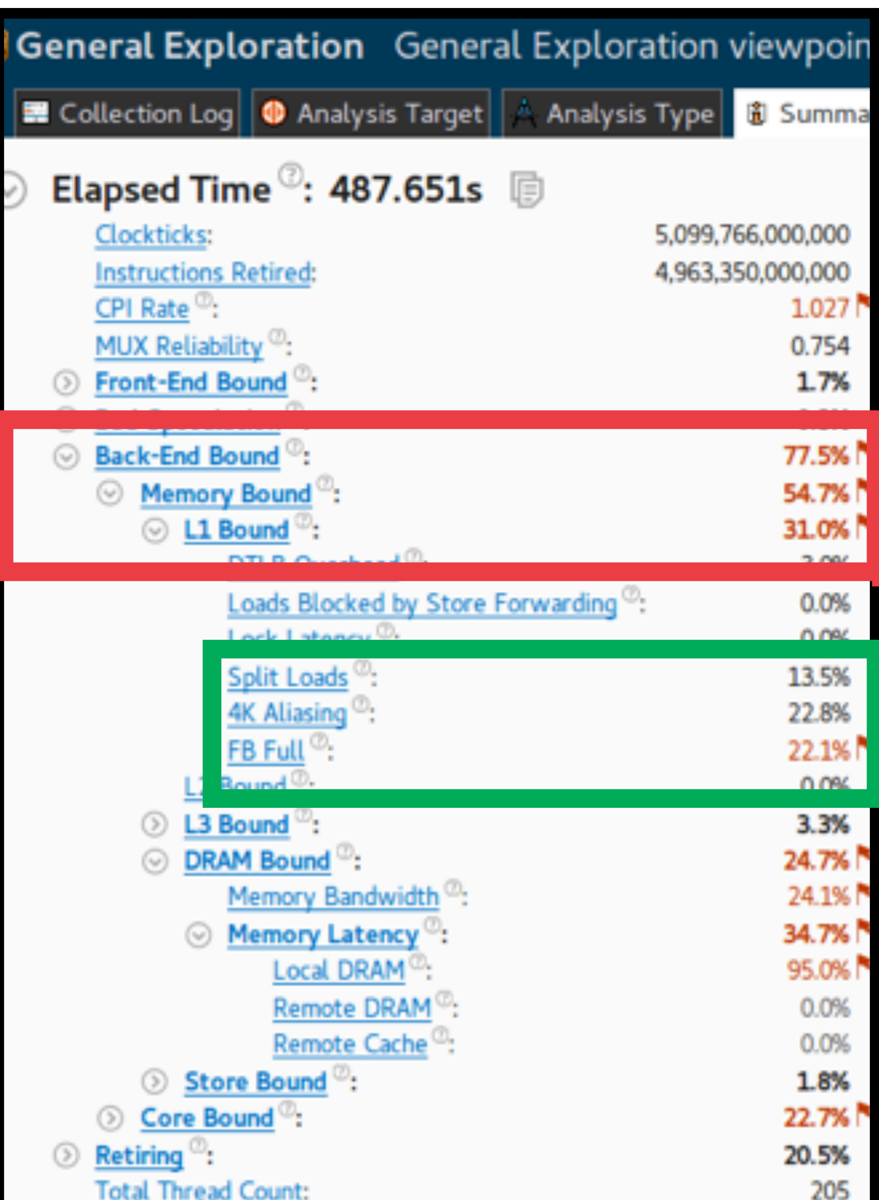
# Devito Loop Engine

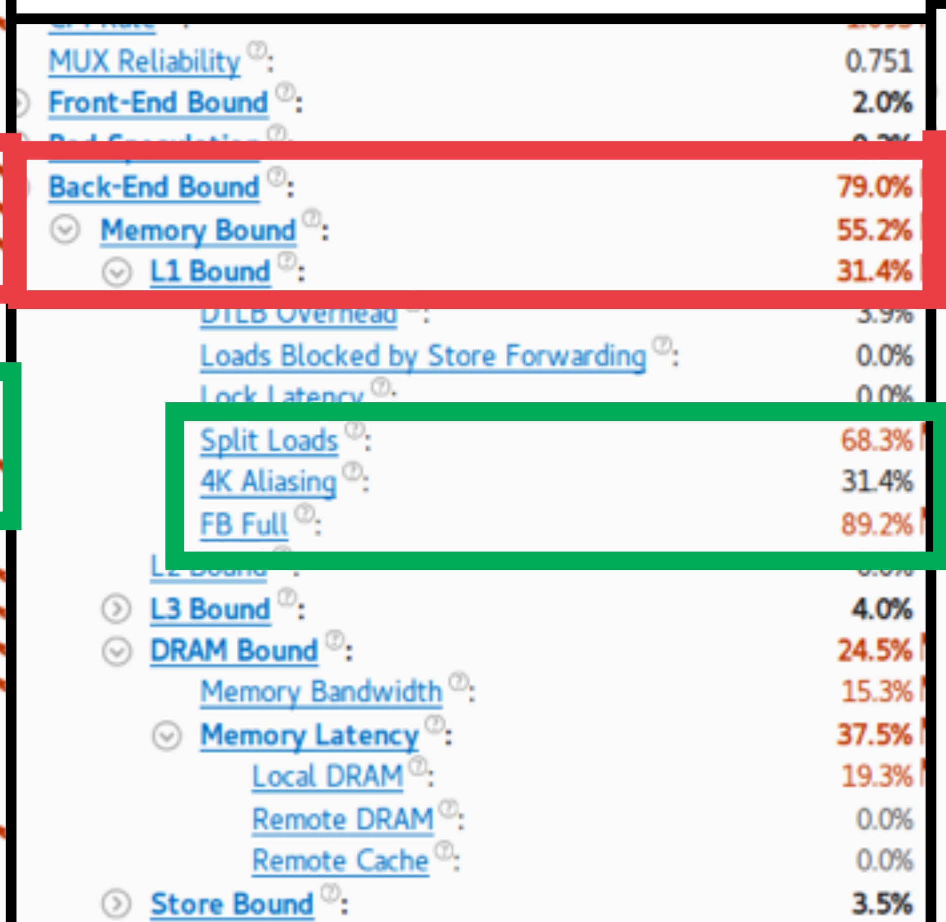
A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality

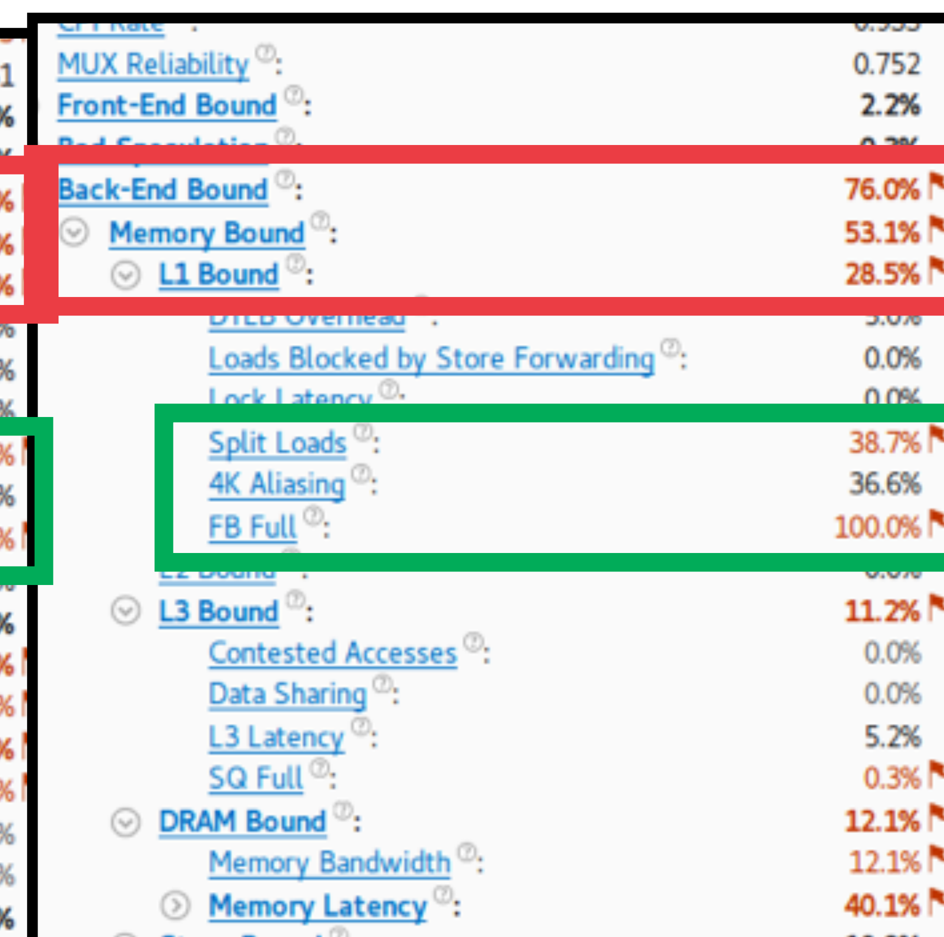
# Devito Loop Engine

A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality

*Intel VTune, Broadwell E5-2620 v4, TTI space orders 4-8-12*



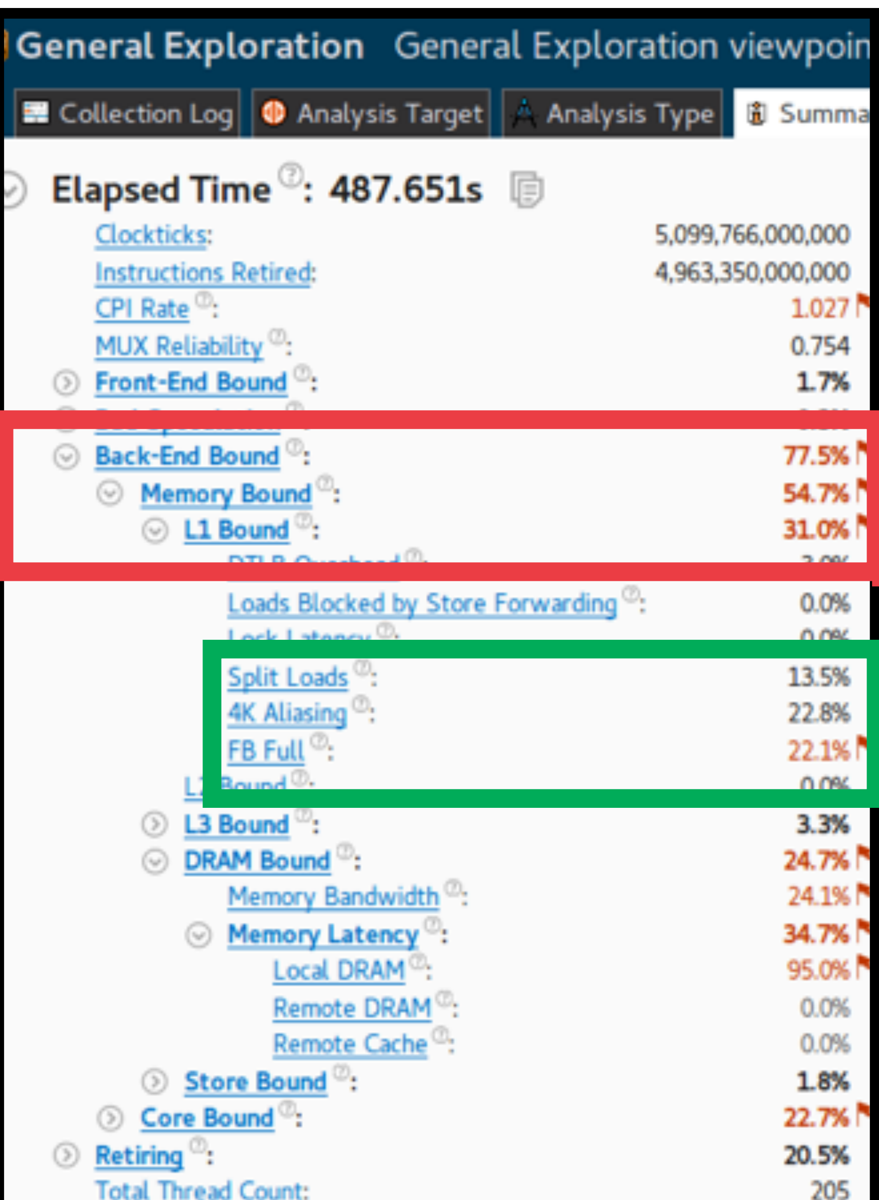
# Devito Loop Engine

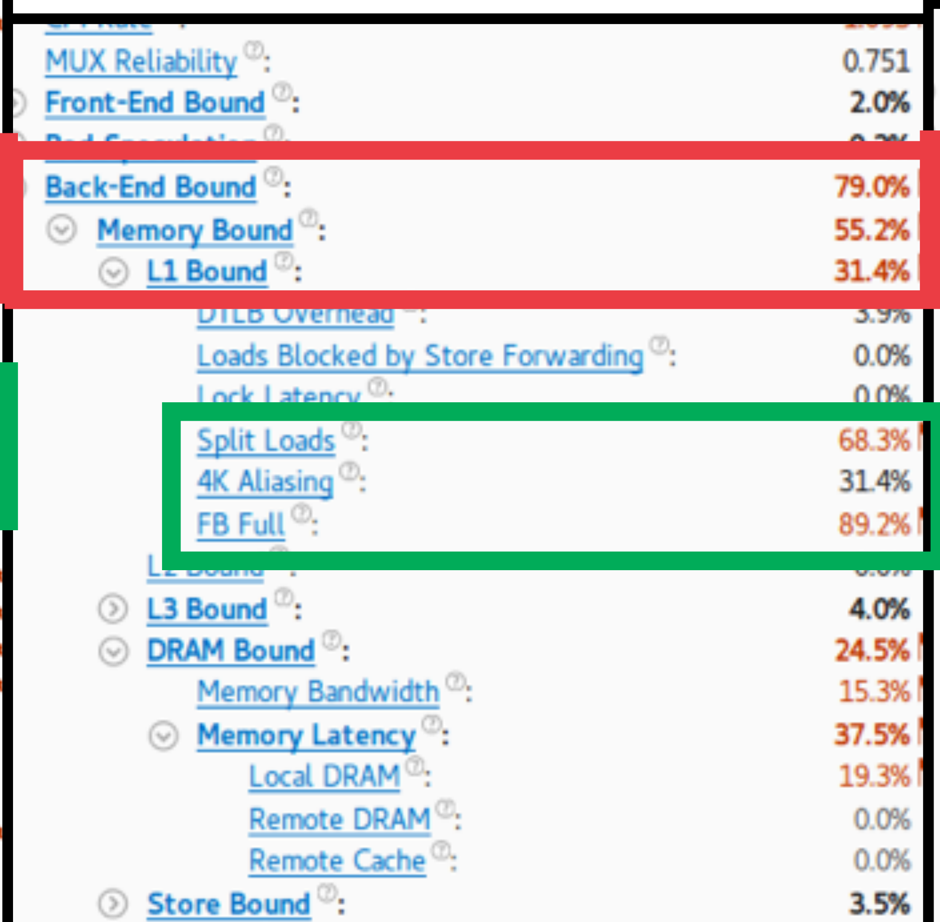
A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality

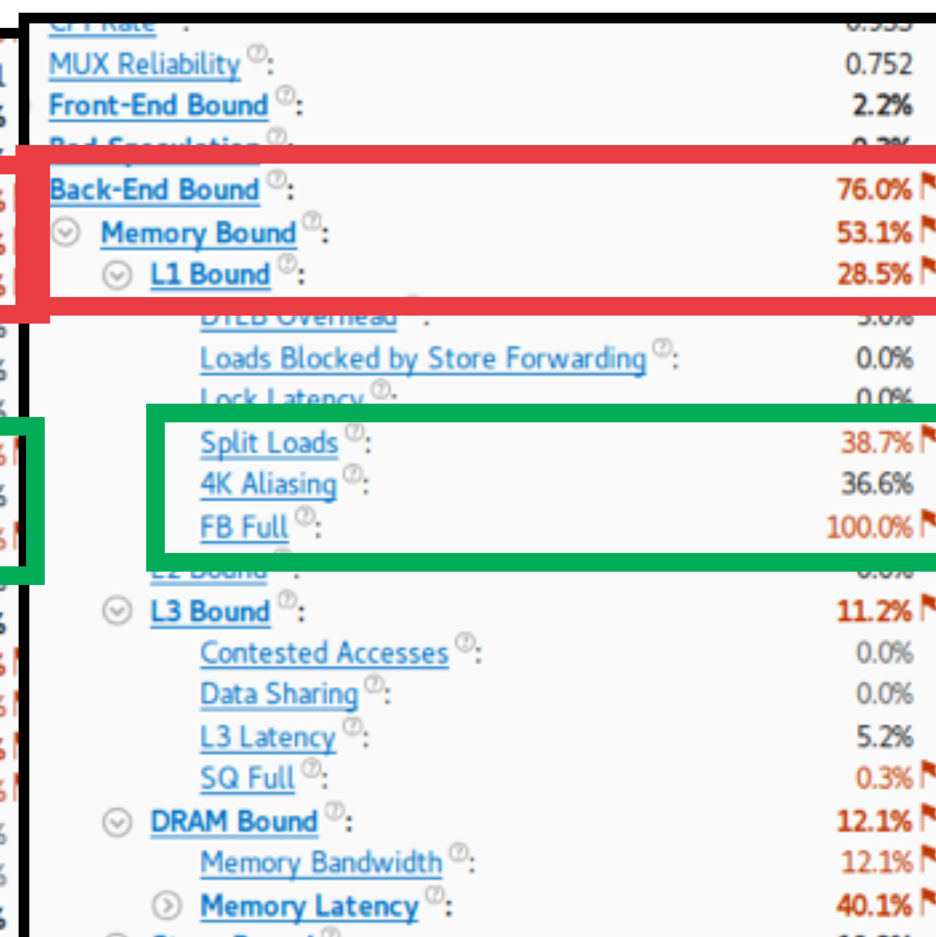
Cache opts

- Cache optimizations (mostly L1 cache)
  - Loop fission + elemental functions (register locality)
  - Padding + data alignment (split loads)
  - Work in progress: data layout transformations

*Intel VTune, Broadwell E5-2620 v4, TTI space orders 4-8-12*



# Devito Loop Engine

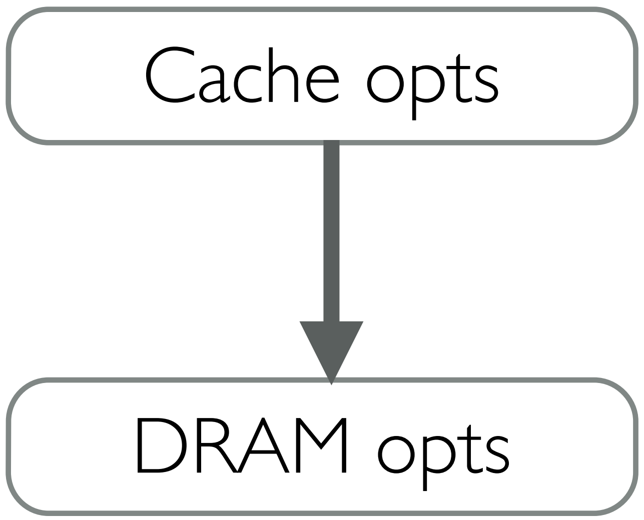
A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality

Cache opts

- Cache optimizations (mostly L1 cache)
  - Loop fission + elemental functions (register locality)
  - Padding + data alignment (split loads)
  - Work in progress: data layout transformations

# Devito Loop Engine

A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality

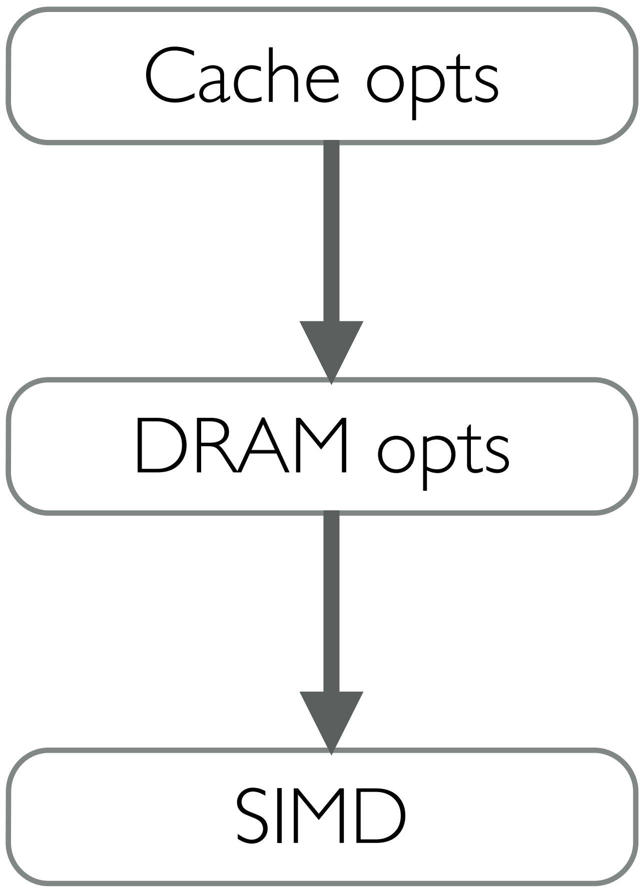



- Cache optimizations (mostly L1 cache)
    - Loop fission + elemental functions (register locality)
    - Padding + data alignment (split loads)
    - Work in progress: data layout transformations

- DRAM optimizations: loop blocking
    - 1D, 2D, 3D supported (but no time loop)
    - Auto-tuning supported

# Devito Loop Engine

A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality



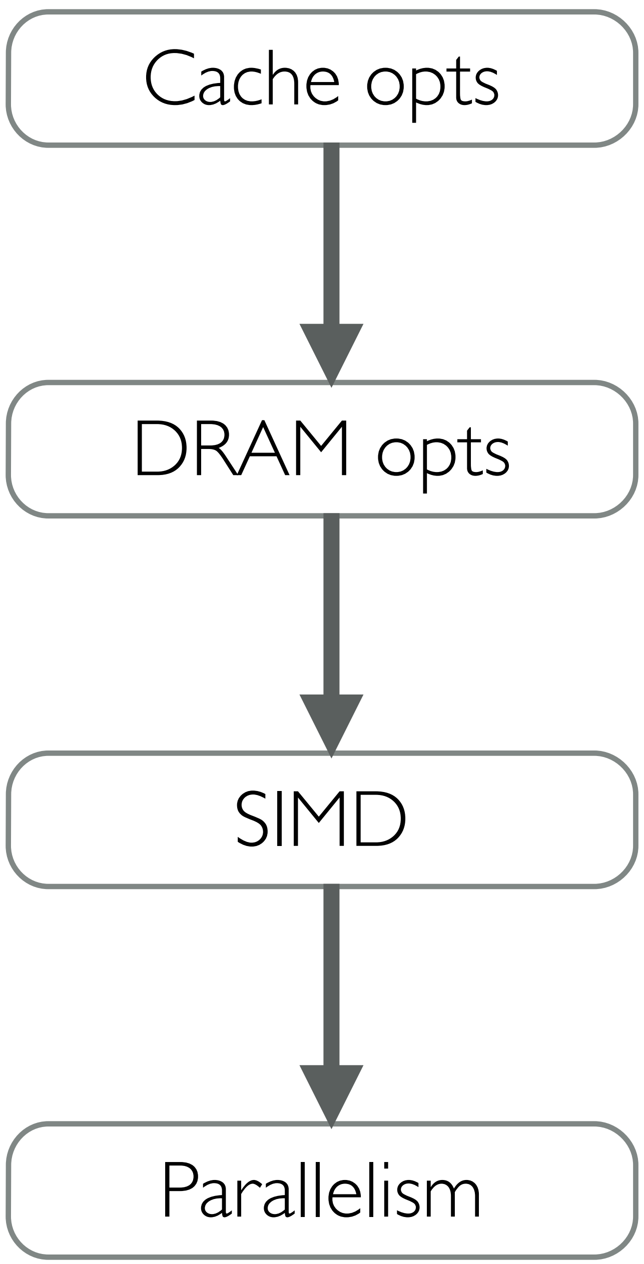

- Cache optimizations (mostly L1 cache)
  - Loop fission + elemental functions (register locality)
  - Padding + data alignment (split loads)
  - Work in progress: data layout transformations

- DRAM optimizations: loop blocking
  - 1D, 2D, 3D supported (but no time loop)
  - Auto-tuning supported

- SIMD vectorization
  - Through compiler auto-vectorization
  - Why should I bother using intrinsics?
  - Various #*pragma*s introduced (e.g., ivdep, alignment, …)

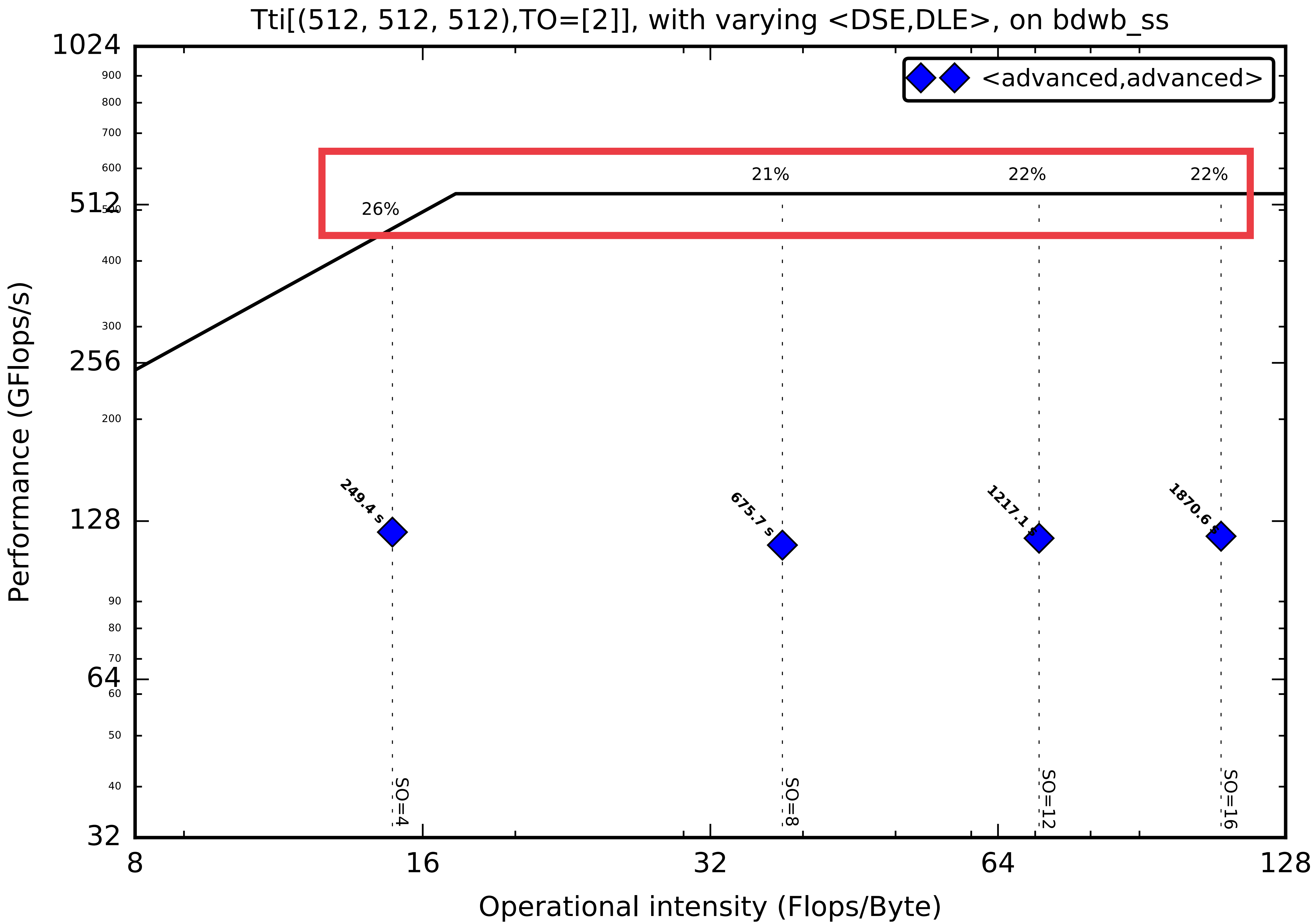
# Devito Loop Engine

A sequence of compiler passes to introduce parallelism, SIMD vectorization and to improve data locality

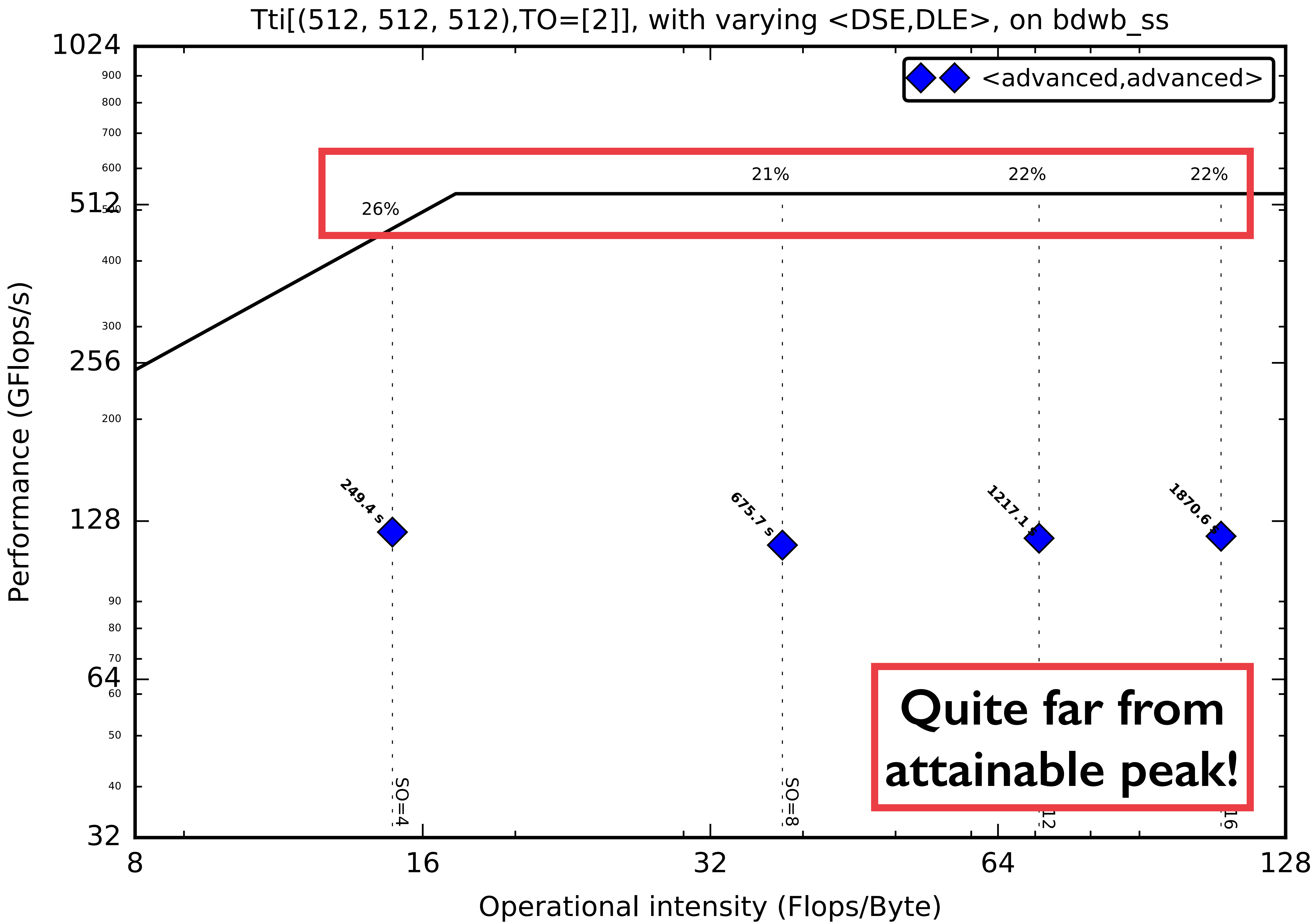



- Cache optimizations (mostly L1 cache)
  - Loop fission + elemental functions (register locality)
  - Padding + data alignment (split loads)
  - Work in progress: data layout transformations
- DRAM optimizations: loop blocking
  - 1D, 2D, 3D supported (but no time loop)
  - Auto-tuning supported
- SIMD vectorization
  - Through compiler auto-vectorization
  - Why should I bother using intrinsics?
  - Various #*pragma*s introduced (e.g., ivdep, alignment, …)
- OpenMP
  - #*pragma collapse* clause on the Xeon Phi

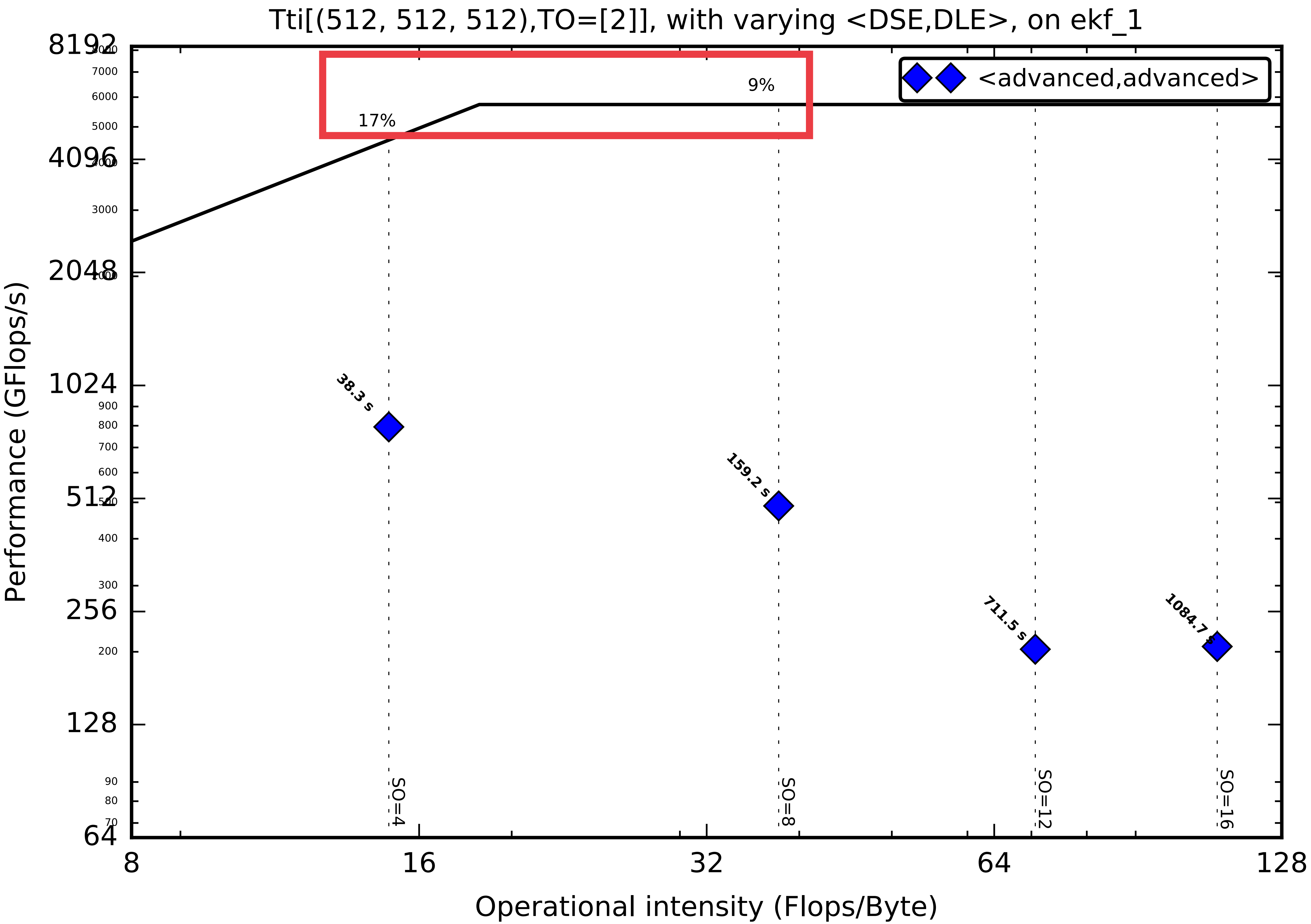
# TTI on Broadwell (8 threads, single socket)



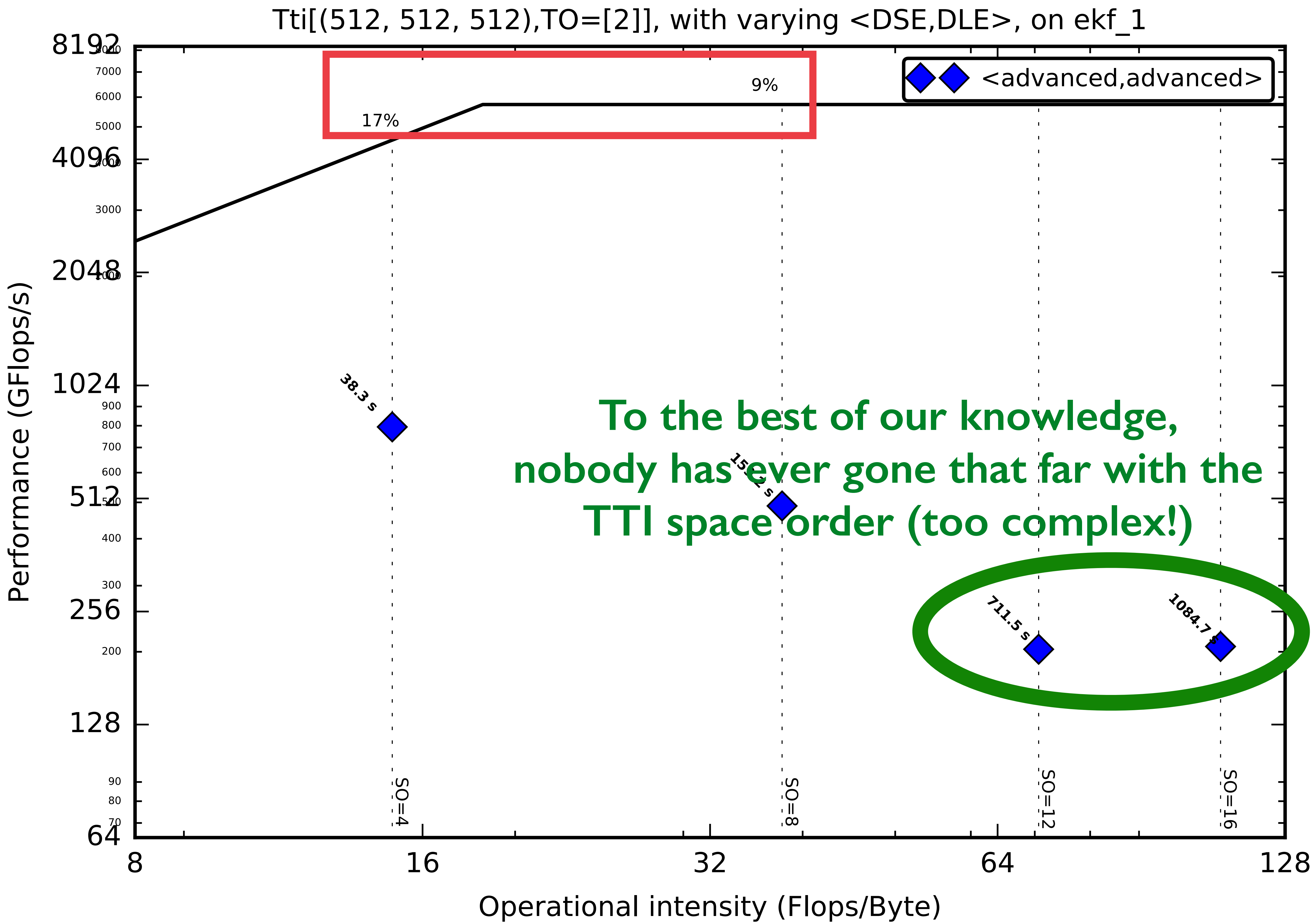
# TTI on Broadwell (8 threads, single socket)



# TTI on Xeon Phi (64 threads, cache mode, quadrant)

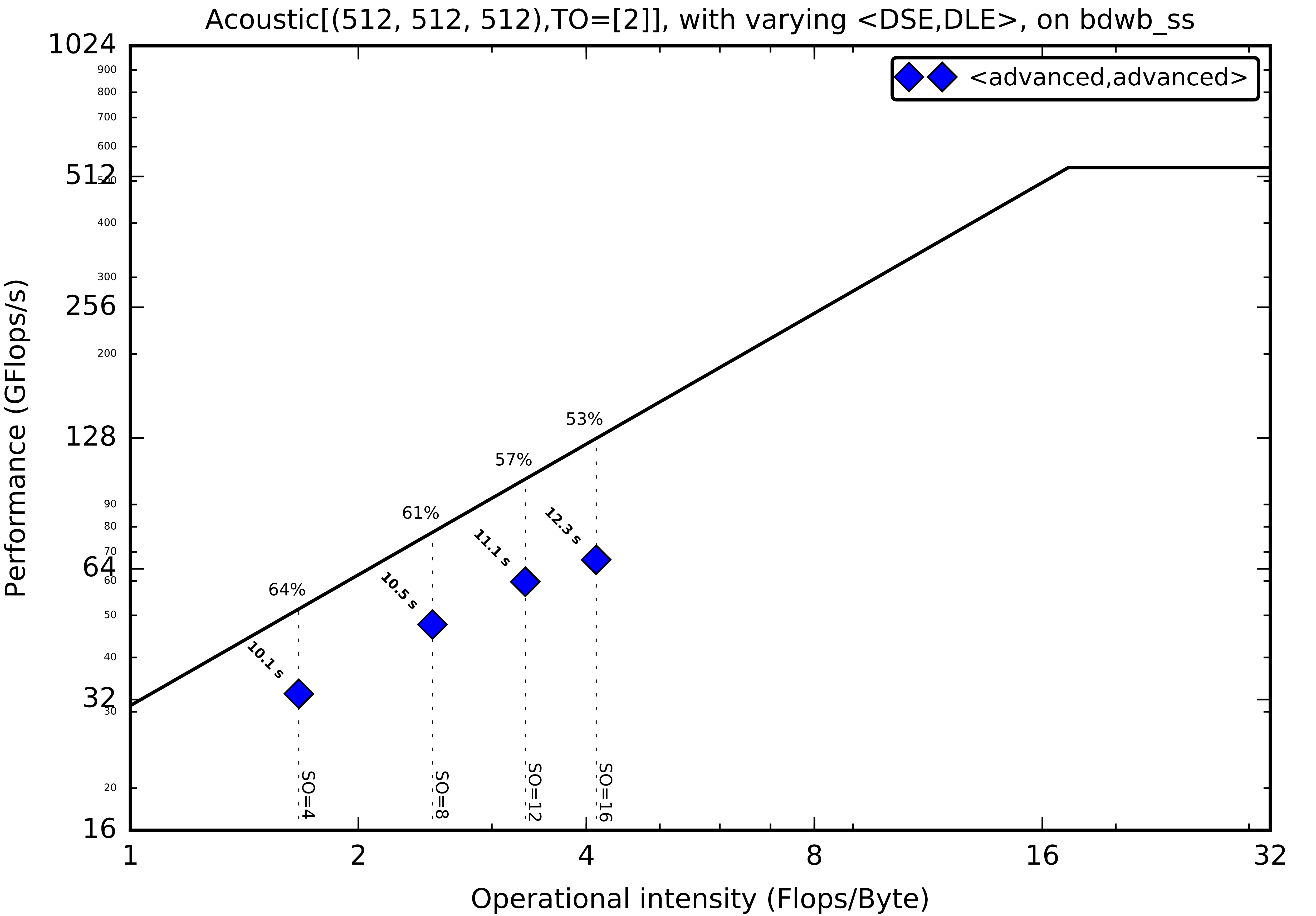


# TTI on Xeon Phi (64 threads, cache mode, quadrant)

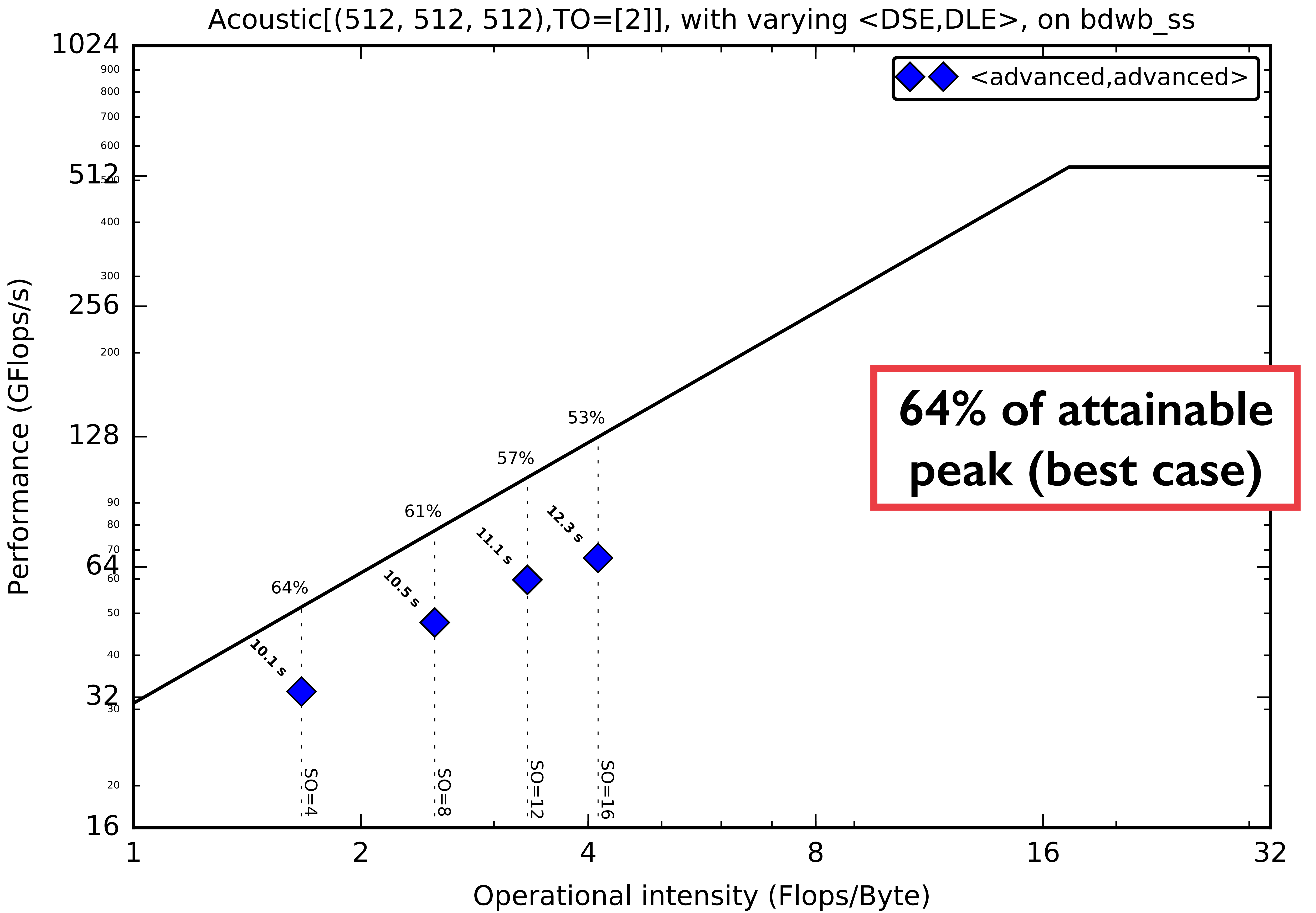



To the best of our knowledge, nobody has ever gone that far with the TTI space order (too complex!)

# Acoustic on Broadwell



# Acoustic on Broadwell



# Conclusions and resources

- Devito (part of OPESCI): towards an efficient and sustainable finite difference DSL
- Driven/inspired by real-world seismic imaging
- Based on actual compiler technology
- Performance: promising, but still quite a lot to do
- Future: plug in backends such as YASK

**Useful links**

- http://www.opesci.org
- https://github.com/opesci/devito

OPESCI

intel

BG GROUP

SINBAD

EPSRC

Engineering and Physical Sciences Research Council

# Appendix

# Experimentation details

- Compiler
  - ICC 17 -xHost -O3 (-O2 no difference)
  - -xMIC-AVX512 on Xeon Phi

- OpenMP
  - Single socket (still no support for NUMA issue through first touch)
  - Thread pinning
  - Numactl

- Intel(R) Xeon(R) E5-2620 v4 2.1Ghz "Broadwell" (8 cores per socket)

- Intel(R) XeonPhi(R) 7650
  - 68 cores (used only 64)
  - Quadrant mode (still no support for NUMA)
  - Tried 1, 2, 4 threads per core. Shown 1 thread (no critical differences)
  - Cache mode performs equivalently to Flat mode when datasets fit in MCDRAM

- Roofline calculations available at: https://gist.github.com/FabioLuporini/12485f08576674d8452fec8673d6f26e
  - Memory bandwidth: STREAM
  - CPU peak: pen & paper
  - Operational intensity: source-level analysis (automated through Devito)

# Summary of experimentation

| | Intel Broadwell | Intel Xeon Phi |
|---|---|---|
| Acoustic forward | | |
| TTI forward | | |

# Summary of experimentation

| | Intel Broadwell | Intel Xeon Phi |
|---|---|---|
| Acoustic forward | 🙂 | |
| TTI forward | | |

# Summary of experimentation

| | Intel Broadwell | Intel Xeon Phi |
|---|---|---|
| Acoustic forward | 🙂 | 😐 |
| TTI forward | | |

# Summary of experimentation

| | Intel Broadwell | Intel Xeon Phi |
|---|---|---|
| Acoustic forward | 🙂 | 😐 |
| TTI forward | 🤔 | |

# Summary of experimentation

| | Intel Broadwell | Intel Xeon Phi |
|---|---|---|
| Acoustic forward | 🙂 | 😐 |
| TTI forward | 🤔 | 🙈 |

# Summary of experimentation

| | Intel Broadwell | Intel Xeon Phi |
|---|---|---|
| Acoustic forward | 🙂 | 😐 |
| TTI forward | 🤔 | 🤔 |

# Devito Loop Engine (example output)

```
int Kernel(float *restrict damp_vec, float *restrict delta_vec, float *restrict epsilon_vec, float *restrict m_vec, float
*restrict phi_vec, …., const int x_size, const int y_size, const int z_size, const int x_block_size, const int y_block_size,
struct profiler *timings)
{
  …

  // PADDED BUFFERS
  float (*pu)[532][532][536];
  …
  posix_memalign((void**)&pti3, 64, sizeof(float[532][532][536]));

  // TIME INVARIANTS
  for (int x = 0; x < x_size; x += 1)
  {
    for (int y = 0; y < y_size; y += 1)
    {
      #pragma noinline
      f_2_0(phi_vec,x_size,x,y_size,y,z_size,(float*) pti0,(float*) pti1,(float*) pti2,(float*) pti3,theta_vec);
    }
  }

  for (int time = 0; time < time_size; time += 1)
  {
    // NEXT SLIDE
  }
  …
}
```

# Devito Loop Engine (example output)

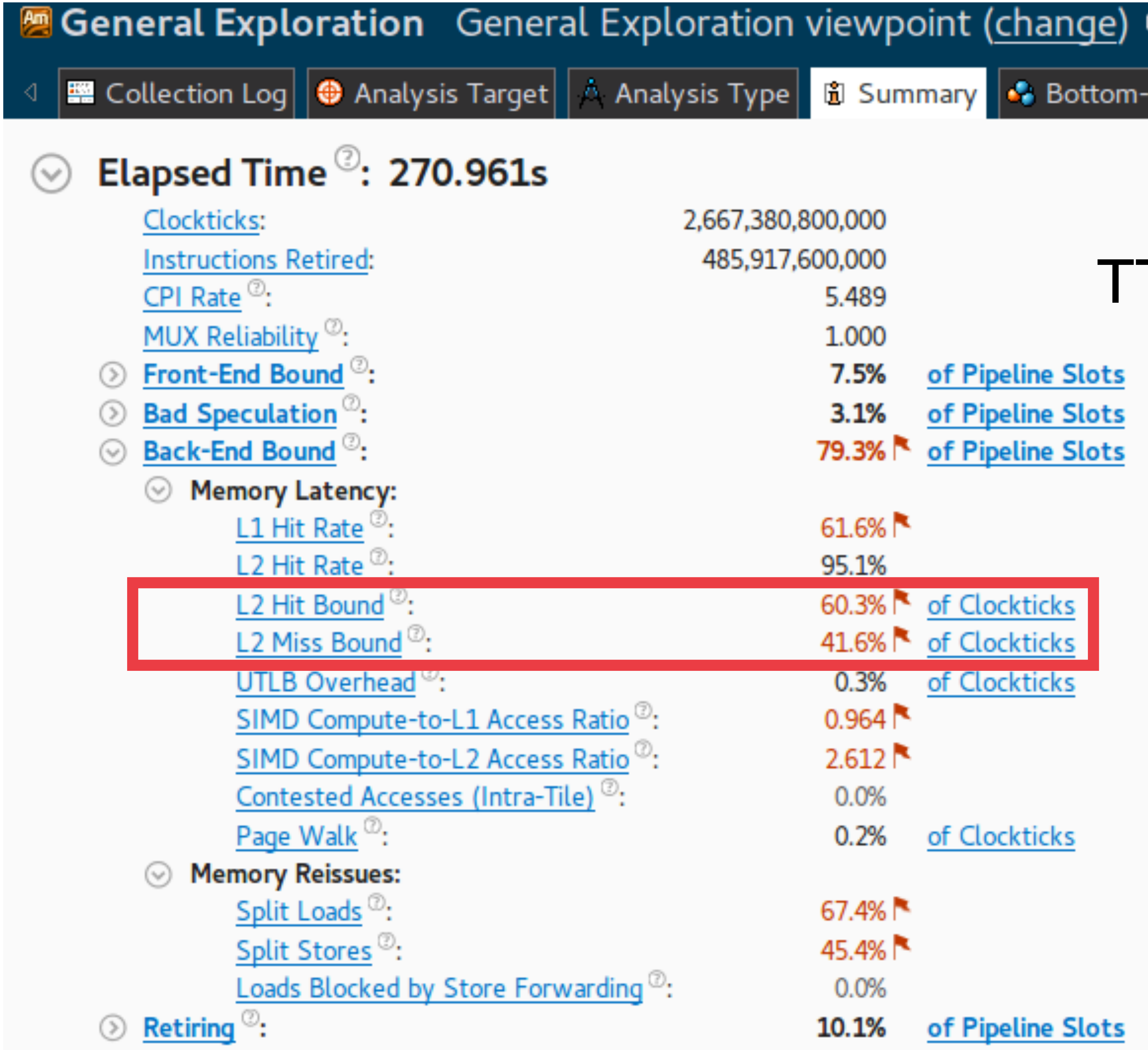
```
#pragma omp parallel
{
  /* Flush denormal numbers to zero in hardware */
  _MM_SET_DENORMALS_ZERO_MODE(_MM_DENORMALS_ZERO_ON);
  _MM_SET_FLUSH_ZERO_MODE(_MM_FLUSH_ZERO_ON);
  #pragma omp for schedule(static,1)
  for (int x_block = 4; x_block < x_size - (x_size - 8)%(x_block_size) - 4; x_block += x_block_size)
  {
    for (int y_block = 4; y_block < y_size - (y_size - 8)%(y_block_size) - 4; y_block += y_block_size)
    {
      double ptemp276[536] __attribute__((aligned(64)));
      double ptemp278[536] __attribute__((aligned(64)));
      // MORE PADDED TEMPORARIES

      for (int x = x_block; x < x_block + x_block_size; x += 1)
      {
        for (int y = y_block; y < y_block + y_block_size; y += 1)
        {
          #pragma noinline
          f_2_1((float*) ptemp276,z_size,(float*) pu,t_size,x_size,x,y_size,y,t1);
          …
          #pragma noinline
          f_2_119((float*) pts48,z_size,(float*) pts49,(float*) pts50,(float*) pv,t_size,x_size,x,y_size,y,t2);
        }
      }
    }
  }
}

// BLOCKING REMAINDER LOOPS

// MODEL SOURCES AND RECEIVERS
#pragma noinline
 f_2_477(m_vec,x_size,y_size,z_size,(float*) pu,t_size,(float*) pv,src_vec,time_size,src_coords_vec,d_size,time,t2);
#pragma noinline
 f_2_478((float*) pu,t_size,x_size,y_size,z_size,(float*) pv,rec_vec,time_size,rec_coords_vec,d_size,time,t2);
 …
```

Loop blocking loops

Intra-block loops

SIMD dimension within these functions

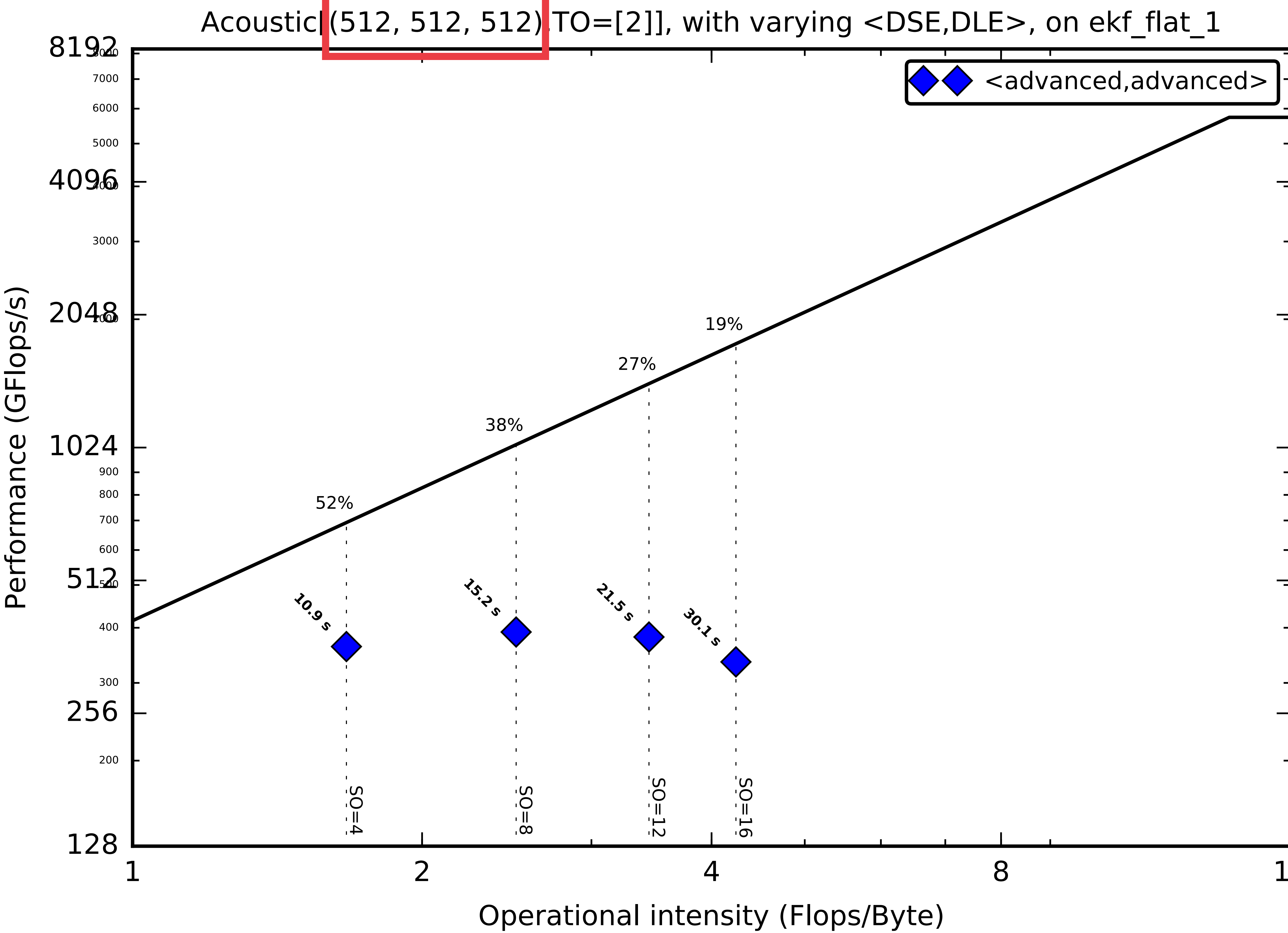
# DLE significantly benefited from Intel VTune



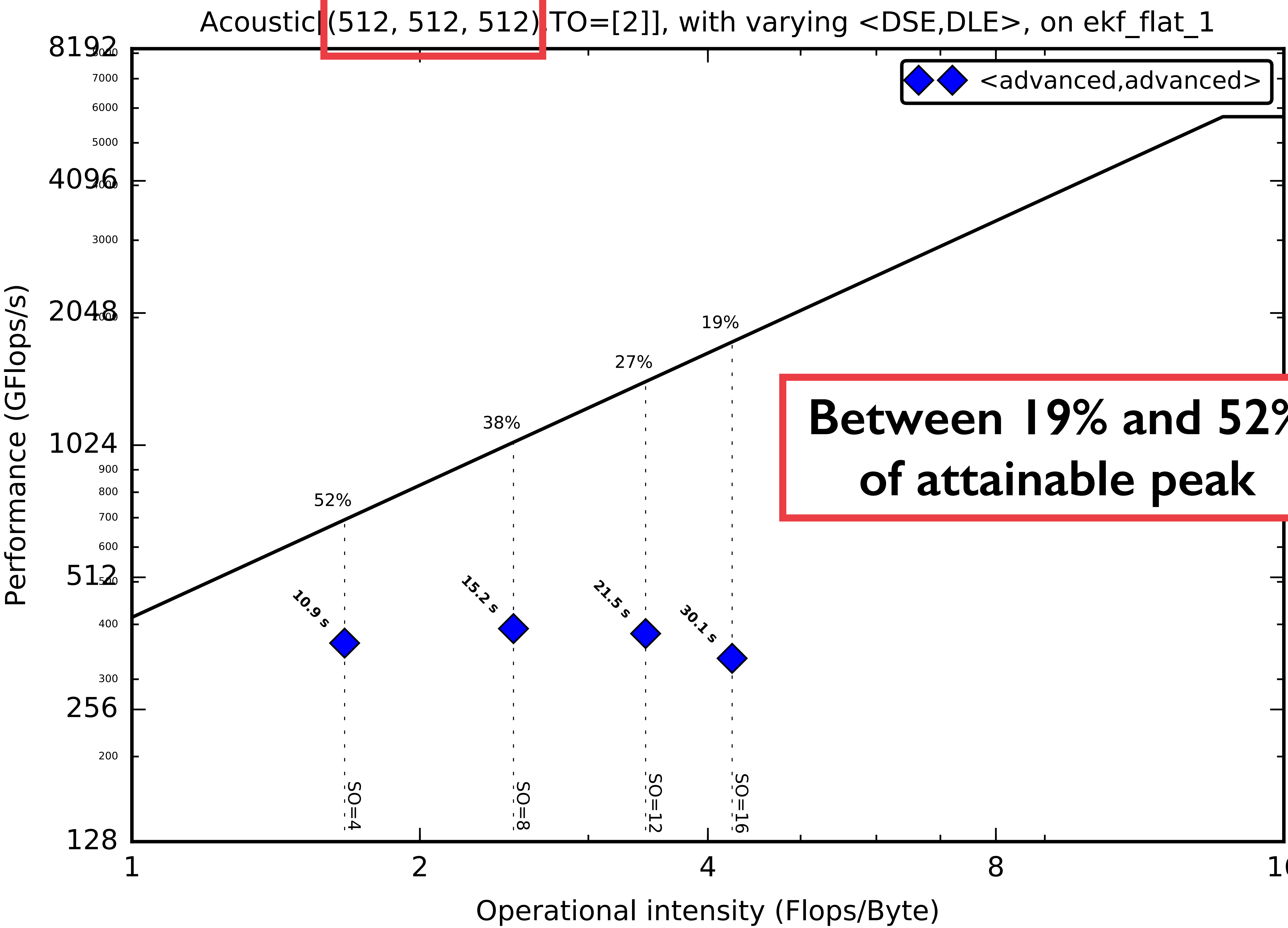

KNL 7650

TTI space order 4

# Acoustic on Xeon Phi (64 threads, in MCDRAM)

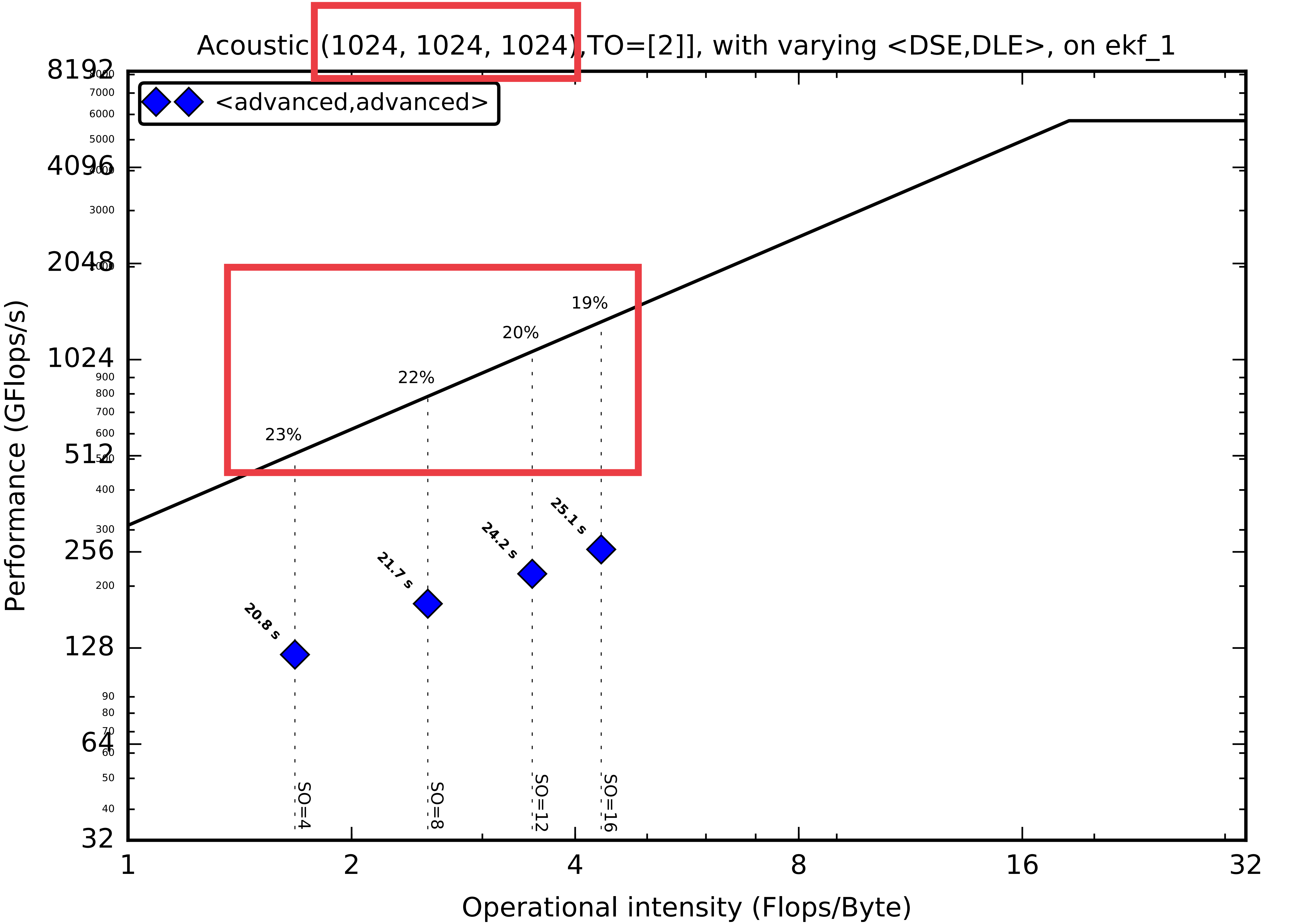


# Acoustic on Xeon Phi (64 threads, in MCDRAM)

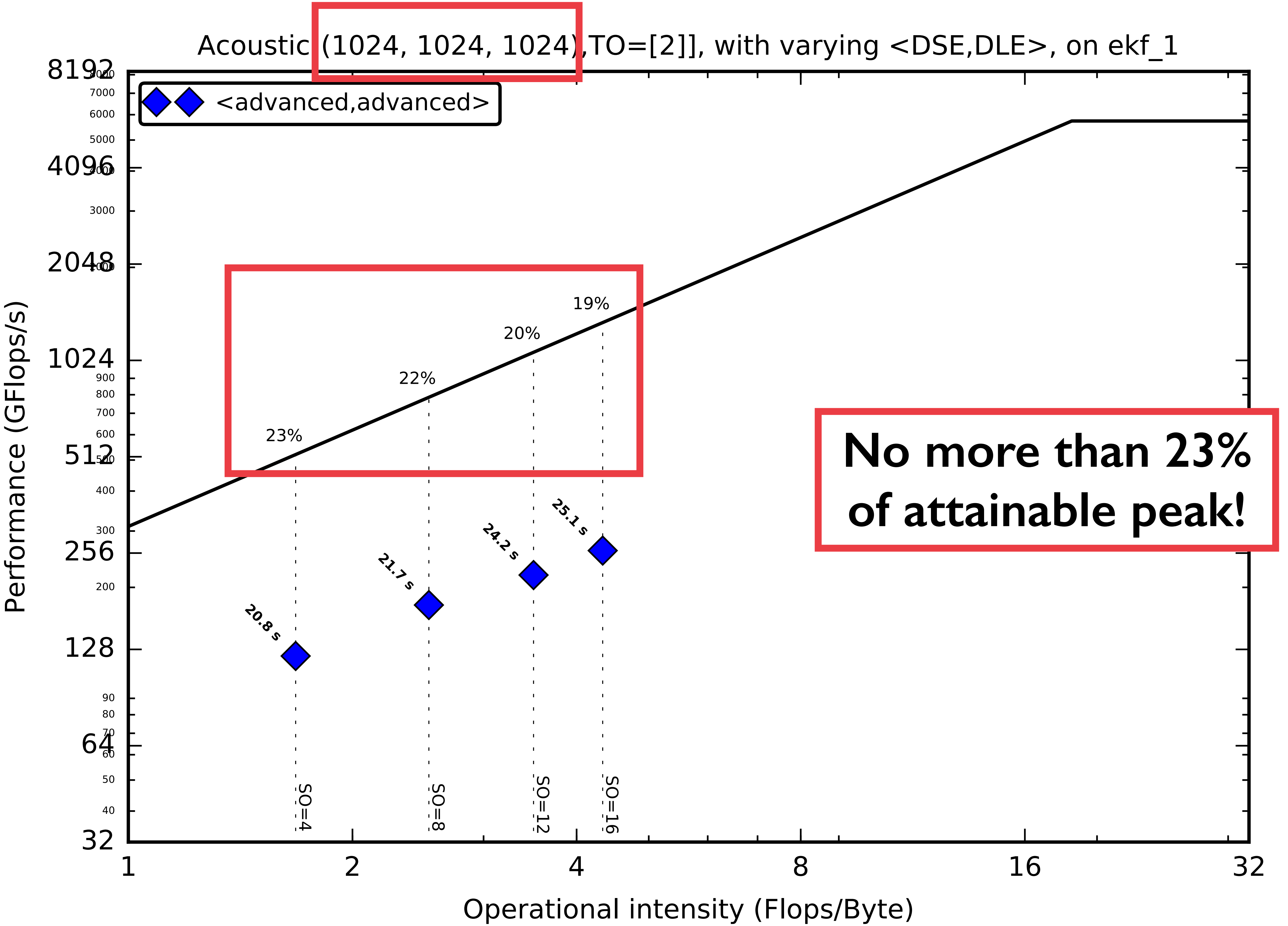



**Between 19% and 52% of attainable peak**

# Acoustic on Xeon Phi (64 threads, needs DRAM)



# Acoustic on Xeon Phi (64 threads, needs DRAM)



# Devito Symbolic Engine (example output)

[illegible]

**ti0[x][y][z] =** 1.6e+1F*(-fabs(theta[x][y][z]) + 3.1416F)*theta[x][y][z]/(-4.0F*(-fabs(theta[x][y][z]) + 3.1416F)*fabs(theta[x][y][z]) + 4.93483e+1F)

…

…

**temp33 =** 2.5e-2F*((-v[t + 1][x][y][z - 1] + v[t + 1][x][y][z + 1])*ti2[x][y][z] + (-v[t + 1][x][y - 1][z] + v[t + 1][x][y + 1][z])*ti0[x][y][z]*ti3[x][y][z]) + (-7.5e-2F*v[t + 1][x][y][z] + 1.0e-1F*v[t + 1][x + 1][y][z] - 2.5e-2F*v[t + 1][x + 2][y][z])*ti0[x][y][z]*ti1[x][y][z]

**temp34 =** …

**temp35 =** … temp33 …

…

# Devito Symbolic Engine (example output)

u[t + 2][x][y][z] = 1.0F*((8.85879567828298e-1F*damp[x][y][z] - 2.0F*m[x][y][z])*u[t][x][y][z] + 1.5695652173913F*(1.25e-2F*((-2.5e-2F*v[t + 1][x][y][z] + 2.5e-2F*v[t + 1][x][y][z + 2])*cos(theta[x][y][z + 1]) + [illegible] ))*sin(phi[x][y - 1][z])*cos(theta[x][y - 1][z]))*epsilon[x][y][z] + 4.0F*m[x][y][z]*u[t + 1][x][y][z])/(8.85879567828298e-1F*damp[x][y][z] + 2.0F*m[x][y][z])

**ti0[x][y][z] =** 1.6e+1F*(-fabs(theta[x][y][z]) + 3.1416F)*theta[x][y][z]/(-4.0F*(-fabs(theta[x][y][z]) + 3.1416F)*fabs(theta[x][y][z]) + 4.93483e+1F)

…

…

**temp33 =** 2.5e-2F*((-v[t + 1][x][y][z - 1] + v[t + 1][x][y][z + 1])*ti2[x][y][z] + (-v[t + 1][x][y - 1][z] + v[t + 1][x][y + 1][z])*ti0[x][y][z]*ti3[x][y][z]) + (-7.5e-2F*v[t + 1][x][y][z] + 1.0e-1F*v[t + 1][x + 1][y][z] - 2.5e-2F*v[t + 1][x + 2][y][z])*ti0[x][y][z]*ti1[x][y][z]

**temp34 =** …

**temp35 =** … temp33 …

…