Towards Seismic Imaging on HPC Architectures: Applications, Algorithms, and Software

Olaf Schenk

Institute of Computational Science

USI Lugano, Switzerland

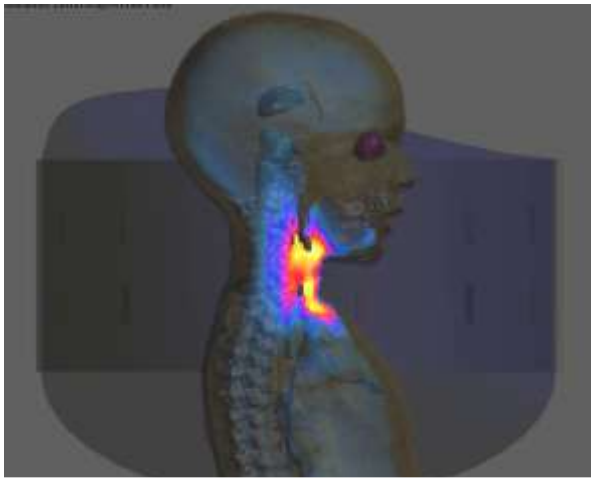
Joint work with:
Prof. D. Giardini, Dr. L. Boschi, Dr. Nissen-Meyer (ETH Zurich, Swiss Seismolgy Unit),
Dr. Christen (USI Lugano), Prof. Jeroen Tromp (Princeton),
Prof. M. Grote (University of Basel), Dr. A. Wächter (IBM T. Watson Research Center)

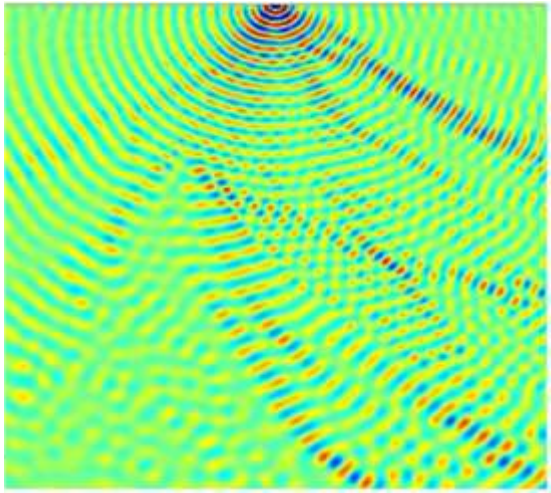
Supported by: Swiss HP2C, SNF, DOE, IBM Faculty Award, Marathon Oil

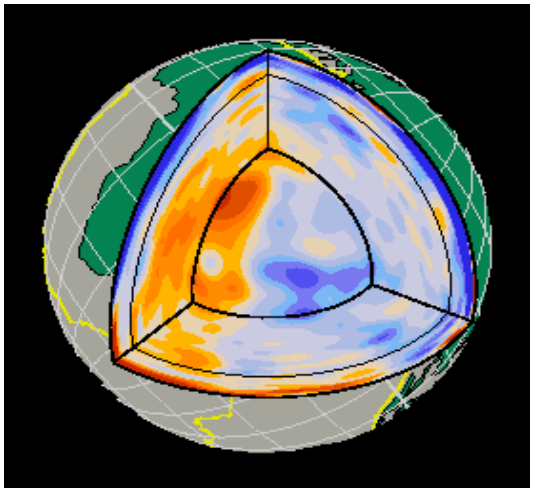
Outline

- Seismic Imaging on HPC Architectures
- Automatic Code Generation and Tuning for Stencil Kernels on Modern Microarchitecture
- High-Resolution Seismic Imaging Results
- Future Work/Conclusion

High-Resolution 3D Imaging







Biomedical Hyperthermia Cancer Therapy

IBM Faculty Award
SNF Project (ETH Zurich, U Basel)

Computational Wave Propagation

SNF Projects (U Basel, USI)

Seismic Inversion / Global Tomography

HP2C Petaquake (USI, ETH Zurich, CSCS)

HP2C High Performance and High Productivity Computing

$$\min_{\mathbf{x}} F(\mathbf{x})$$

$$\text{subject to } c_i(\mathbf{x}) = 0 \quad \text{for } i \in \mathcal{E}$$

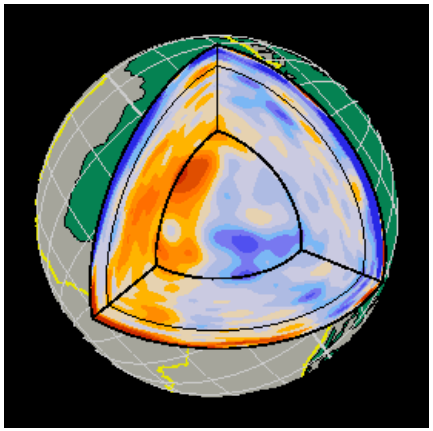
$$c_i(\mathbf{x}) \geq 0 \quad \text{for } i \in \mathcal{I}$$

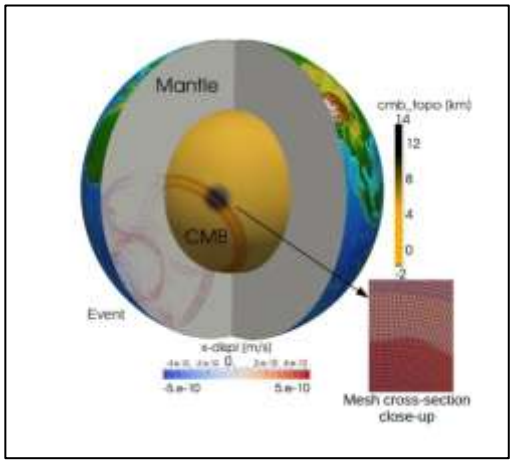
Application: High-Resolution 3D-Seismic Imaging

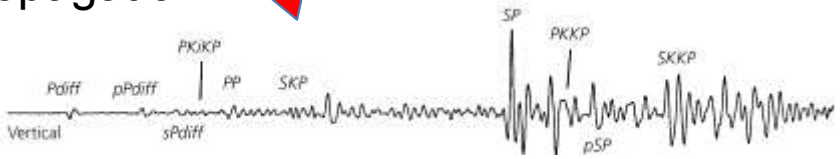
Forward modeling

Seismic wave propagation

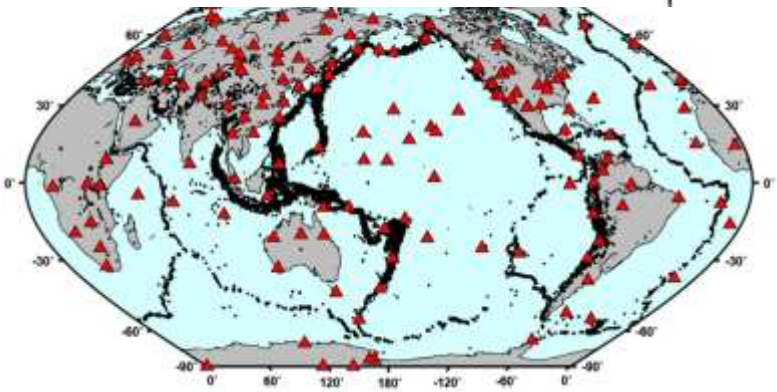
Model space: 3D wavespeeds





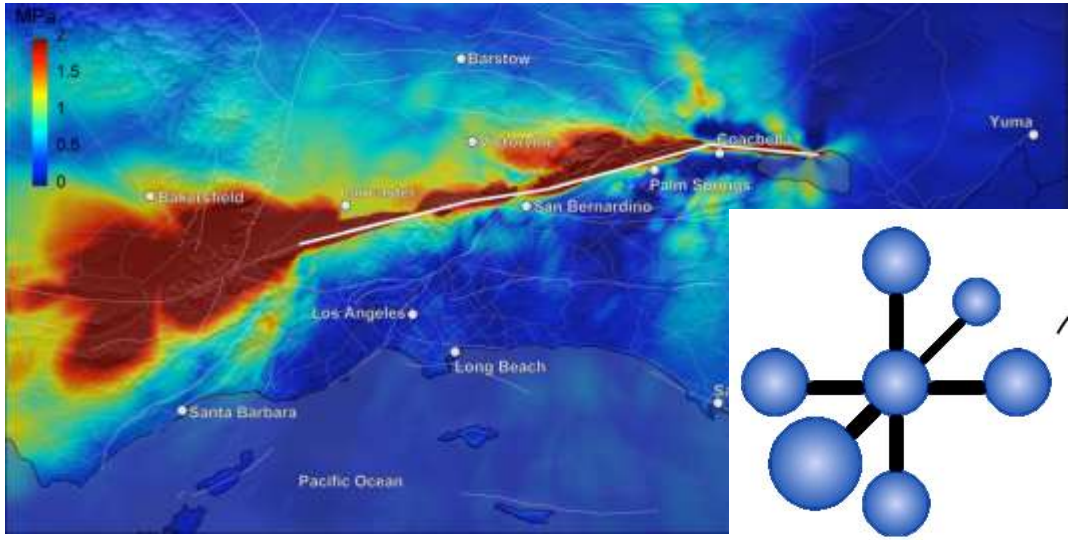


Data space: ground displacements



Inverse modeling/imaging

Application: Earthquakes&seismic hazard / AWP-ODC



$$\frac{\partial \dot{u}}{\partial t} = \rho^{-1}\nabla \cdot \sigma$$

$$\frac{\partial \sigma}{\partial t} = \lambda(\nabla \cdot \dot{u})I + \mu(\nabla \dot{u} + \nabla \dot{u}^T)$$

Coulomb failure stress changes in a simulation of an earthquake on the southern San Andreas Fault

Image courtesy: Southern California Earthquake Center

.

- AWP: Scientific modeling code for anelastic waves
- Capable of simulate accurate earthquake wave propagations
- Used to conduct multiple significant SCEC simulations
- 600 x 300 x 80 km domain, 100m resolution, 14.4 billion grids, 50k time steps.
- Gordon Bell finalist (SC 2010), 220 TFlop/s on 223K Jaguar cores
- Fortran + MPI

Simulation of an earthquake on the southern San Andreas Fault

Movie courtesy: Dalguer (ETH Zurich) et.al.

$$\frac{\partial \dot{u}}{\partial t} = \rho^{-1} \nabla \cdot \sigma$$

$$\frac{\partial \sigma}{\partial t} = \lambda(\nabla \cdot \dot{u})I + \mu(\nabla \dot{u} + \nabla \dot{u}^T)$$

Scalability of the AWP-ODC Stencil-Code on Jaguar

TABLE 2
EVOLUTION OF AWP-ODC

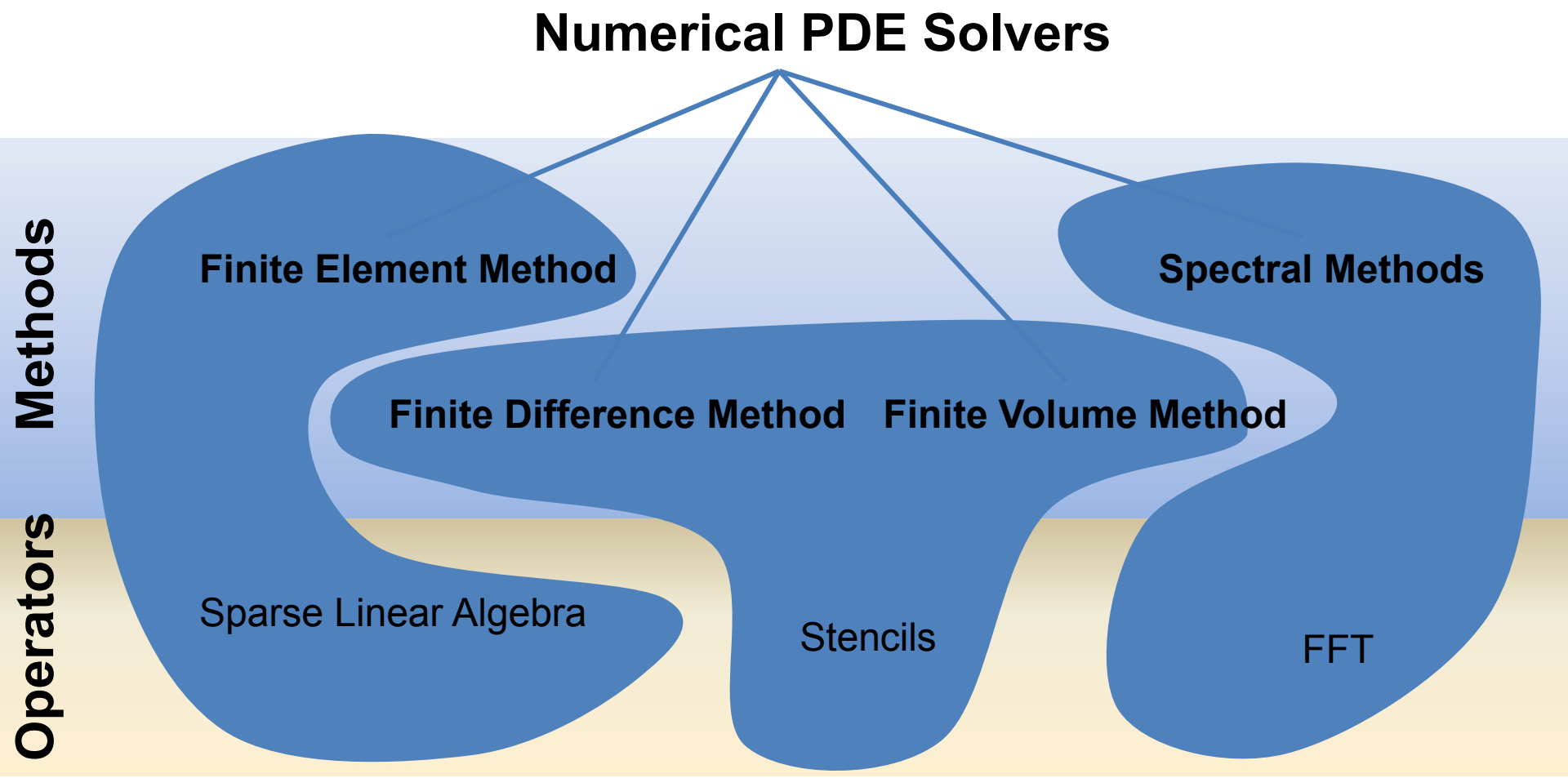
Year	Code version	Simulations	Optimization	SCEC alloc. SUs	Sustain. Tflop/s
2004	1.0	TeraShake-K	MPI tuning	0.5M	0.04
2005	2.0	TeraShake-D	I/O tuning	1.4M	0.68
2006	3.0	PN MQuake	partition. mesh	1.0M	1.44
2007	4.0	ShakeOut-K	incorp. SGSN	15M	7.29
2008	5.0	ShakeOut-D	asynchronous	27M	49.9
2009	6.0	W2W	single CPU opt	32M	86.7
2010	7.0 7.1 7.2	M8	overlap cache blocking reduced comm	61M	220

Highly scalable AWP-ODC code: **220 TFlop/s sustained** on **220k cores** (Jaguar)

Outline

- Seismic Imaging on HPC Architectures
- **Automatic Code Generation and Tuning for Stencil Kernels on Modern Microarchitecture**
- High-Resolution Seismic Imaging Results
- Future Work/Conclusion

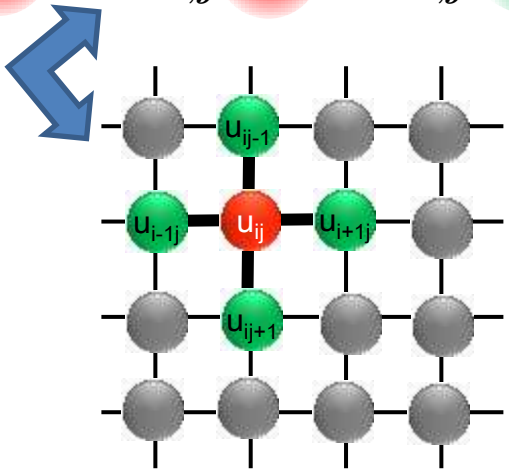
PDE Solution Techniques



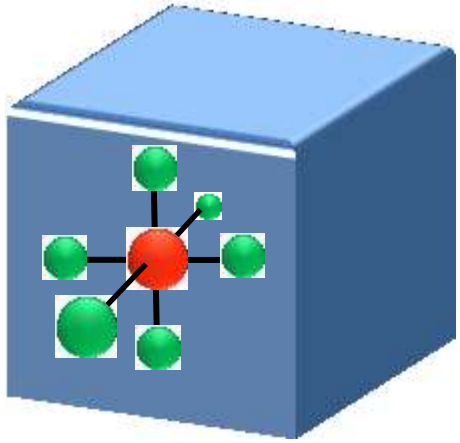
What Is a Stencil?

Weighted sum of subset of neighbors of a grid point

$$u_{i,j} \leftarrow c_{i,j}^{(0)} u_{i,j} + c_{i,j}^{(1)} u_{i-1,j} + c_{i,j}^{(2)} u_{i+1,j} + c_{i,j}^{(3)} u_{i,j-1} + c_{i,j}^{(4)} u_{i,j+1}$$



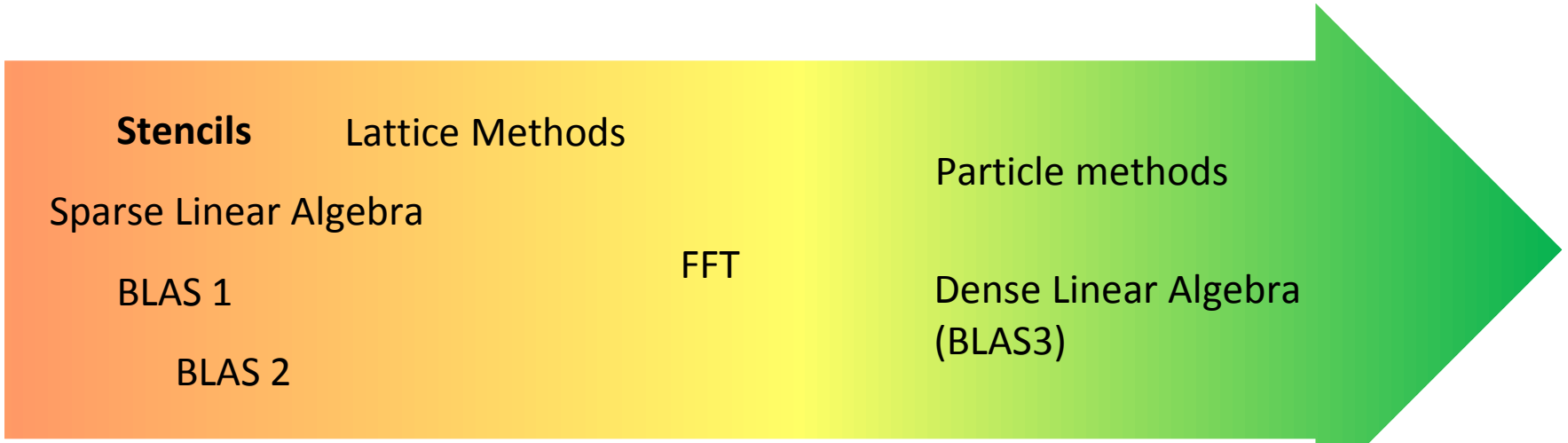
5-point stencil in a regular 2D grid



7-point stencil in 3D

Challenge: Arithmetic Intensity

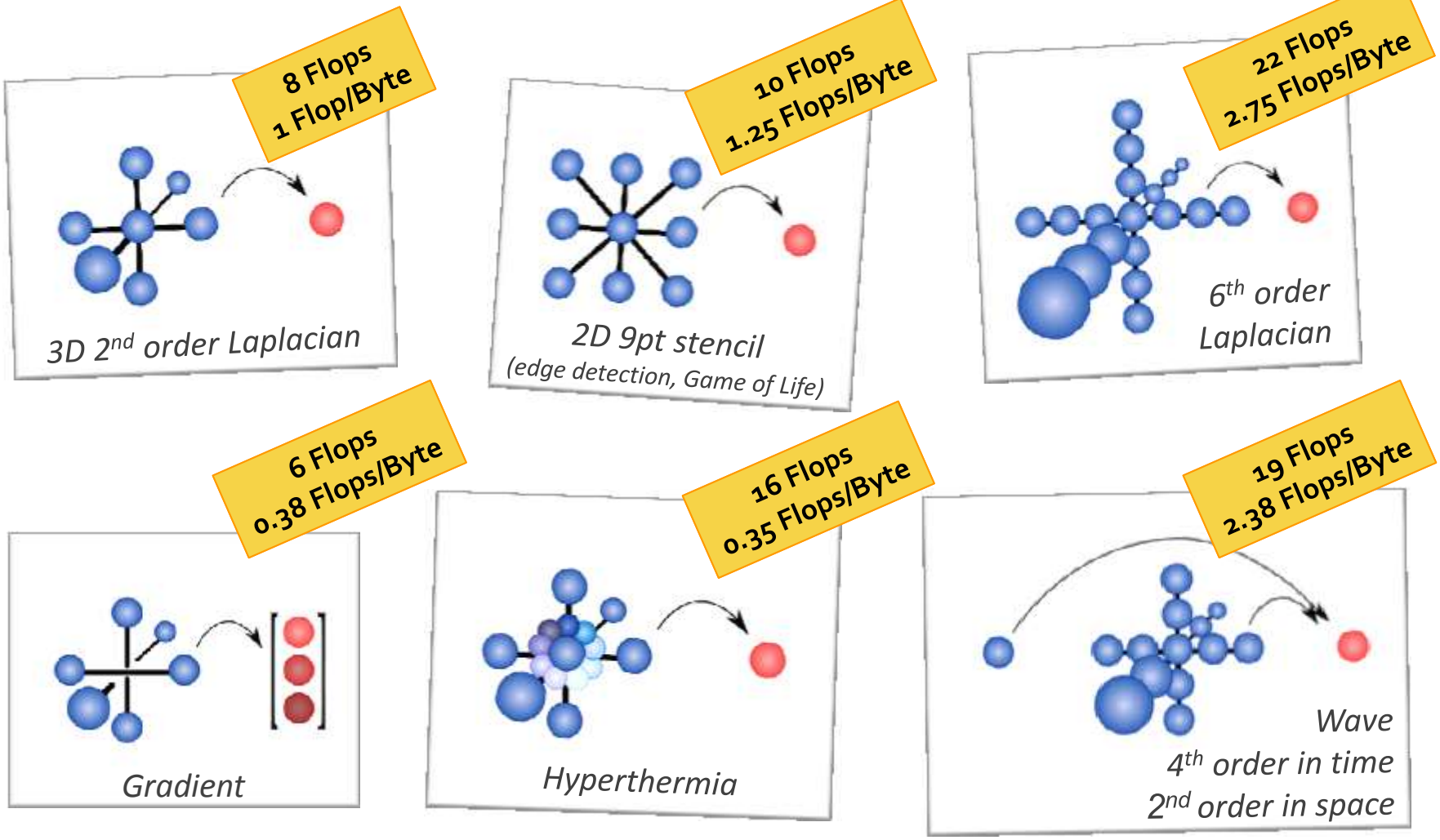
Arithmetic Intensity := Flops / Transferred Data



Low arithmetic intensity
$\Rightarrow$ memory bandwidth bound

High arithmetic intensity
$\Rightarrow$ processor bound

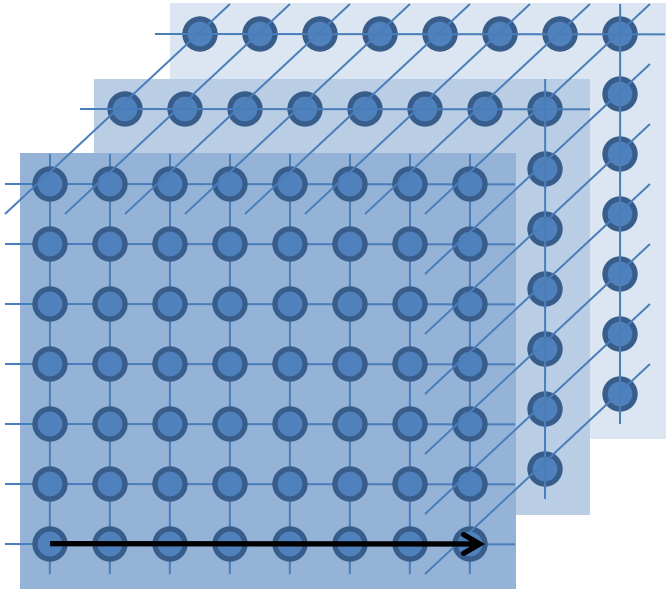
Stencil Variety



Reduce Memory Traffic on Multi-/Manycores

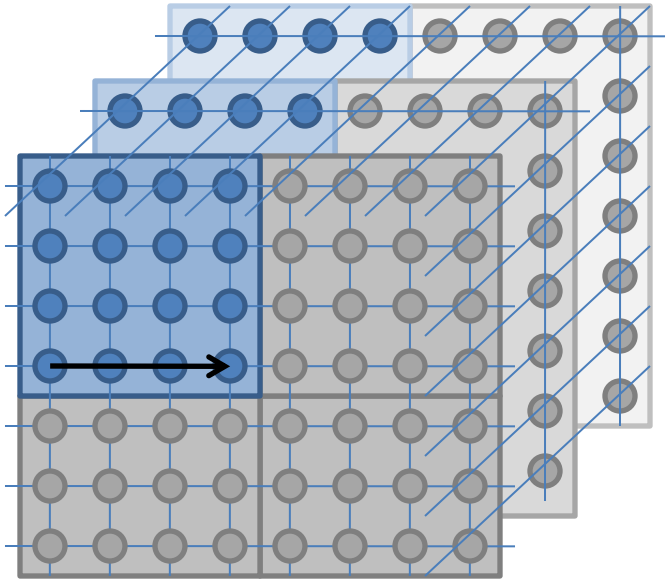
- Stencil performance usually limited by memory bandwidth
- **Goal**: Increase performance by minimizing memory traffic
  - Even more important for many/multicore!
- Concentrate on getting reuse both:
  - within an iteration (**spatial blocking**)
  - across time iterations (**temporal blocking**, Ax, A^2x, …, A^kx)
- Only interested in final result

Naïve Grid Traversal



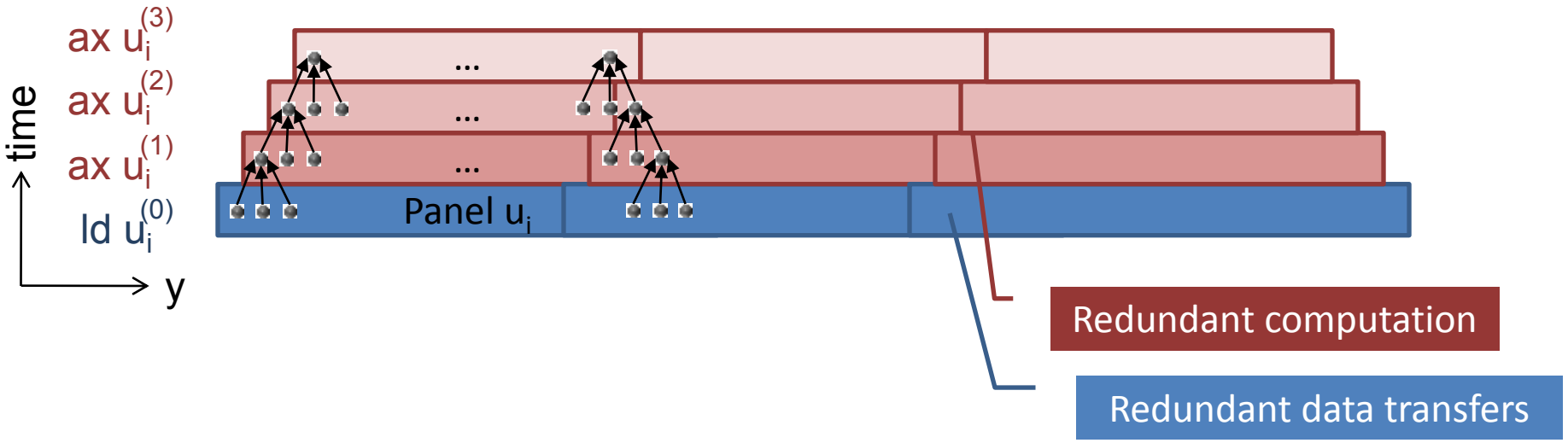
- Traverse the grid in the “usual” way
- Locality not exploited
- Performance will suffer if grid doesn’t fit into cache

Spatial Cache Blocking – Reuse data in space



- 3D block partitioning
- Reuse data within an iteration

Temporal Cache Blocking – Reuse data in time



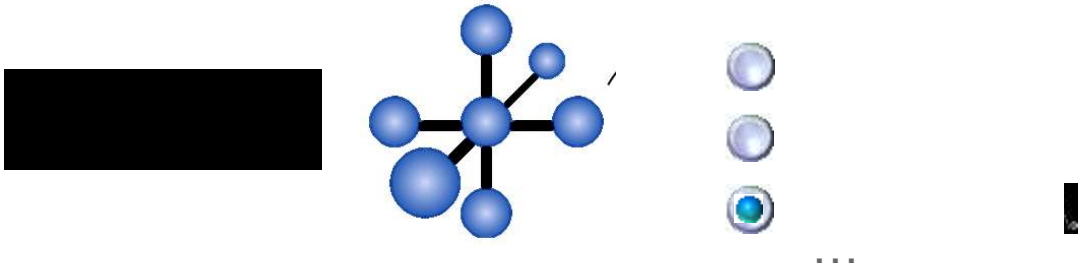
Sizes of the panels shrink with each stencil sweep

(-) Redundant computation and data transfers

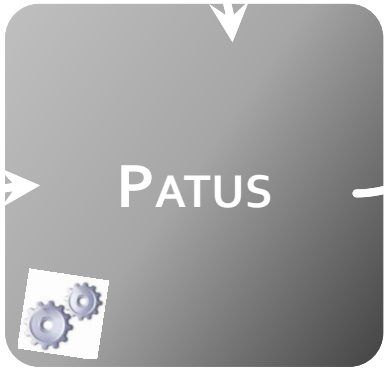
(+) Easily parallelizable along the horizontal (y) axis

PATUS Stencil Project in a Nutshell

- PATUS (Parallel Autotuning of Stencils) in a Nutshell

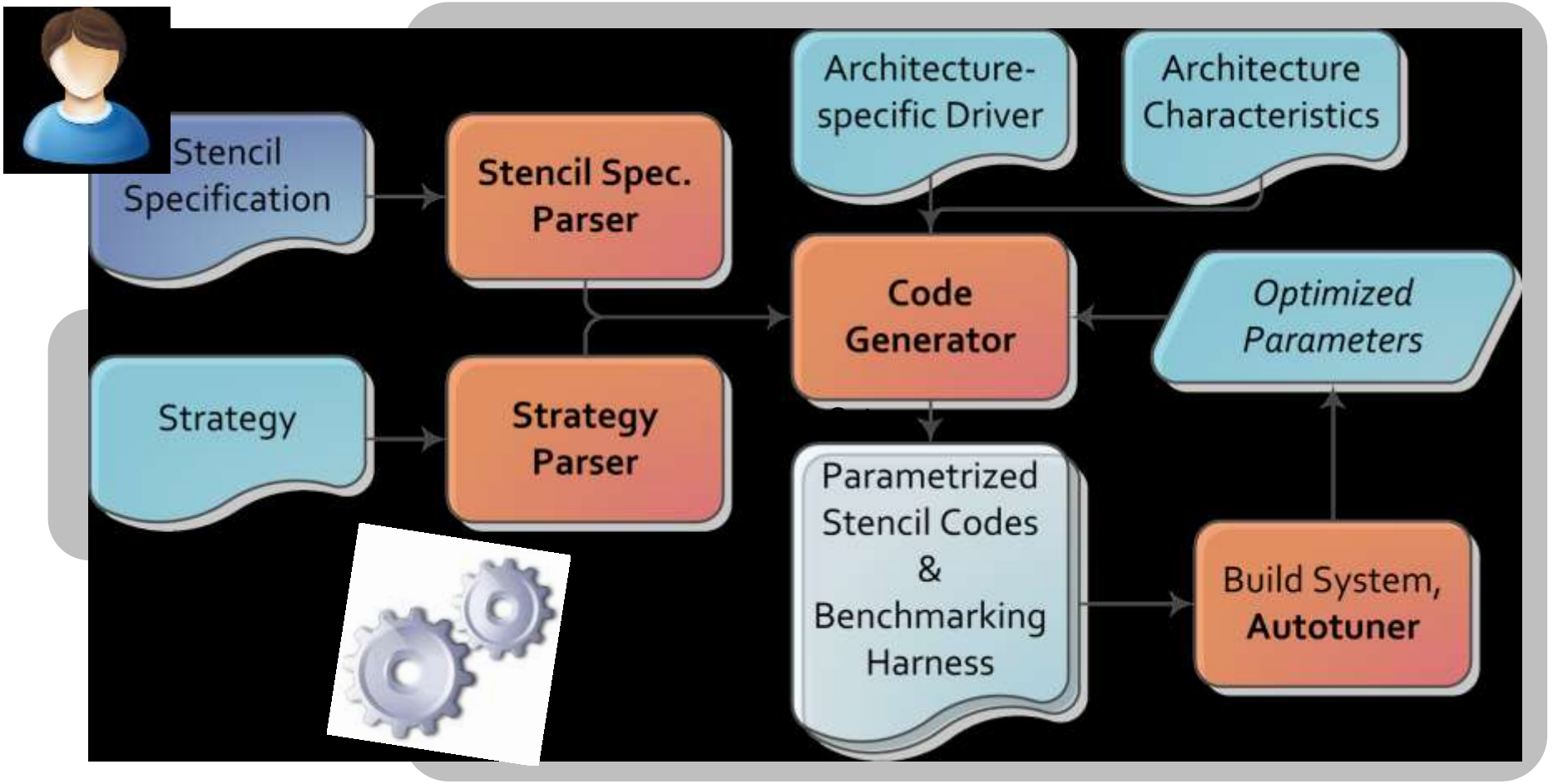


```
stencil laplacian
{
  operation(float grid u,
    float param alpha,
    float param beta)
  {
    u[x, y, z; t+1] =
      alpha * u[x, y, z; t] +
      beta  * (
        u[x+1,y,z;t]+u[x-1,y,z;t]+
        u[x,y+1,z;t]+u[x,y-1,z;t]+
        u[x,y,z+1;t]+u[x,y,z-1;t]);
  }
}
```




```
/*
(u[0][0][0][1][0]=((alpha*u[0][0][0
][0][0])+(beta*((u[1][0][0][0][0]+(
u[-
1][0][0][0][0]+u[0][1][0][0][0]))+(
u[0][-
1][0][0][0]+(u[0][0][1][0][0]+u[0][
0][-1][0][0]))))))
*/
__global__ void laplacian(float *
*  u_0_1_out, float *  u_0_0, float
*  u_0_1, float alpha, float beta,
int x_max, int y_max, int z_max,
int tbx, int tby, int tbz, int c)
{
  float *  const u__u_0[] =  {
u_0_0, u_0_1 } ;
  size_1_1=(y_max/blockDim.y);
  size_1_2=(z_max/blockDim.z);
  idx_1_2=(blockIdx.y/size_1_2);
```

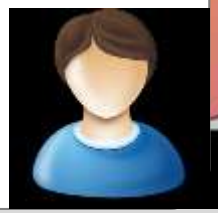
PATUS: Architectural Overview



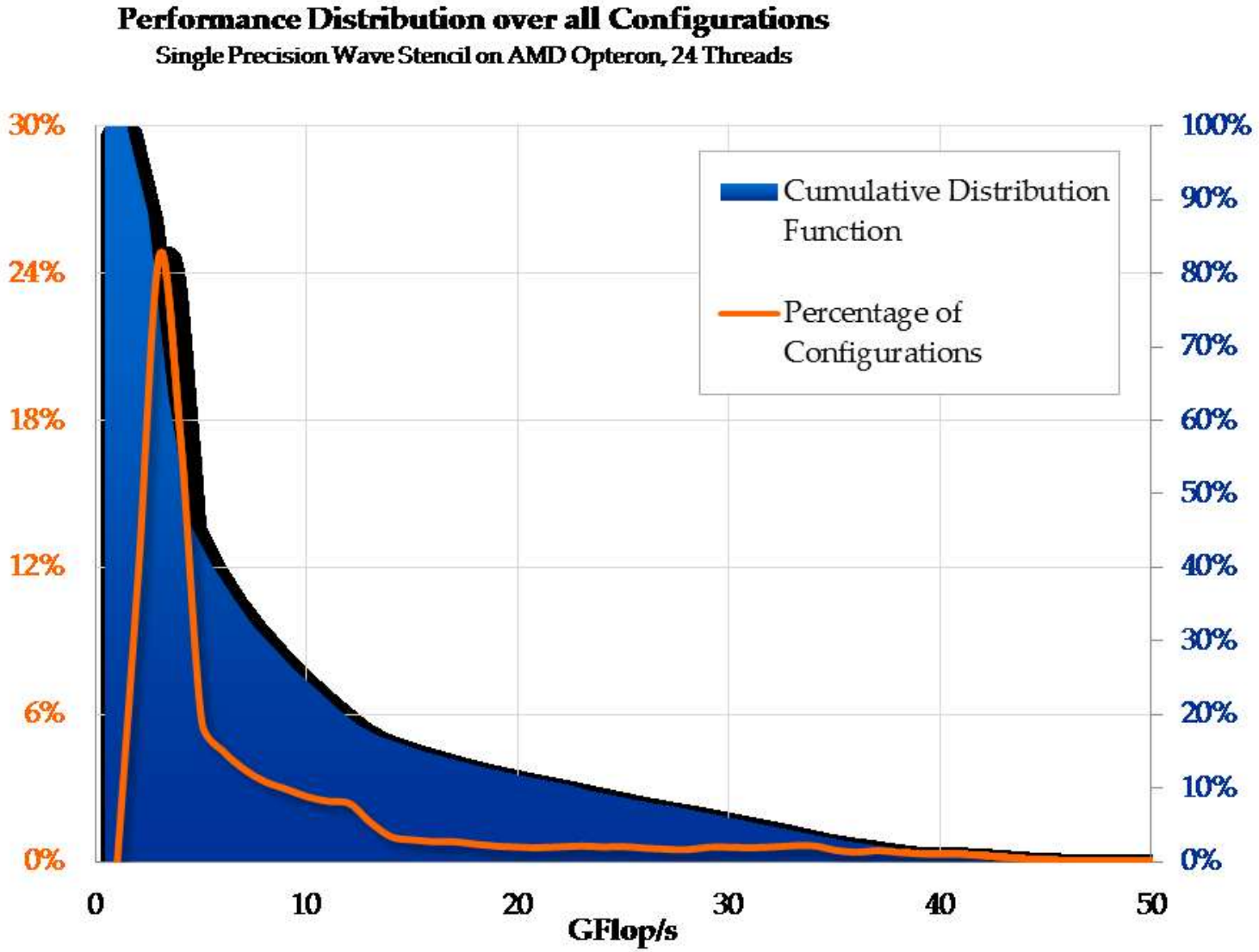
M. Christen, O. Schenk et al., *PATUS: A Code Generation and Autotuning Framework For Parallel Iterative Stencil Computations on Modern Microarchitectures*, IPDPS 2009, IPDPS 2010, IPDPS 2011

Specifying a Stencil

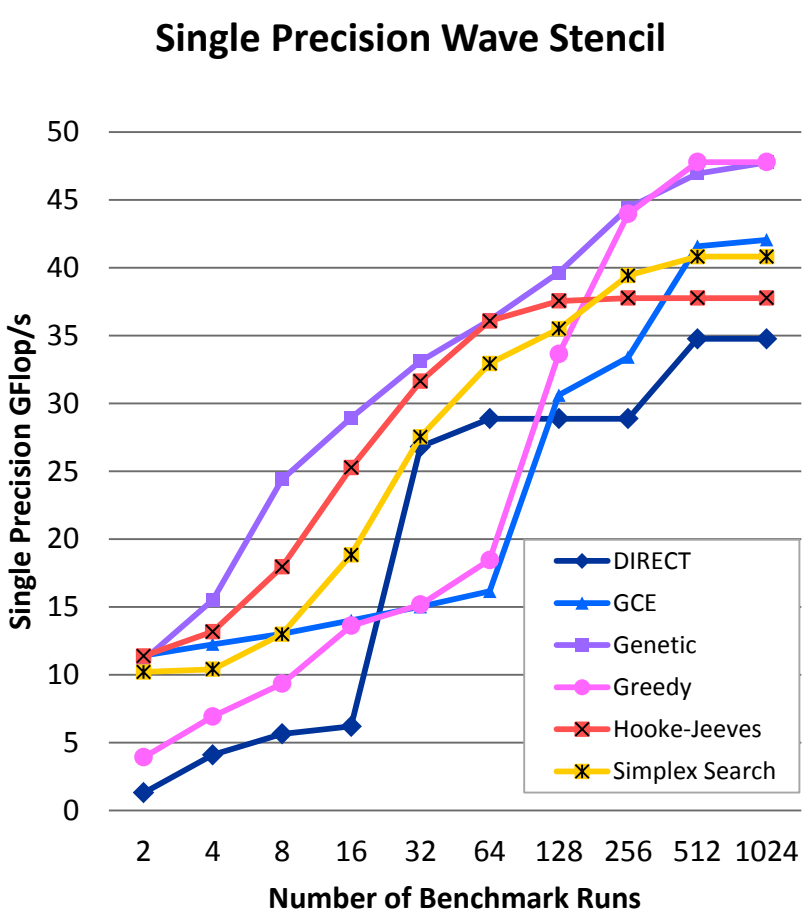
```
stencil pmcl3d_uxx1
{
  domainsize = (nxb .. nxe, nyb .. nye, nzb .. nze);
  t_max = 1;

  operation (
    const float grid d1(-1 .. nxt+2, -1 .. nyt+2, -1 .. nzt+2),
    float grid u1(-1 .. nxt+2, -1 .. nyt+2, -1 .. nzt+2), ...
    float param dth)
  {
    float d = 0.25*(d1[x,y,z]+d1[x,y-1,z]+d1[x,y,z-1]+d1[x,y-1,z-1]);

    u1[x,y,z; t+1] = u1[x,y,z; t] + (dth / d) * (
      c1 * (xx[x,y,z] - xx[x-1,y,z] + xy[x,y,z] - xy[x,y-1,z] +
        xz[x,y,z  ] - xz[x,y,z-1]) +
      c2 * (xx[x+1,y,z] - xx[x-2,y,z] + xy[x,y+1,z] - xy[x,y-2,z] +
        xz[x,y,z+1] - xz[x,y,z-2])
    );
  }
}
```

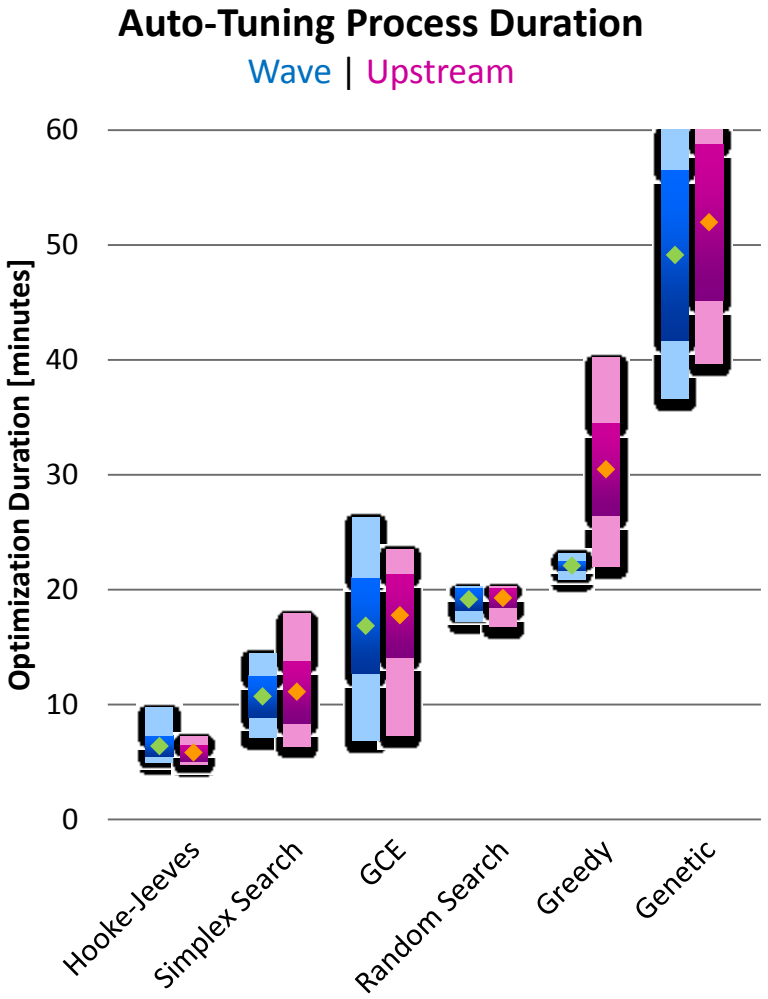


Auto-Tuning (Single-Precision Wave Stencil)



Search Methods (Single-Precision Wave Stencil)

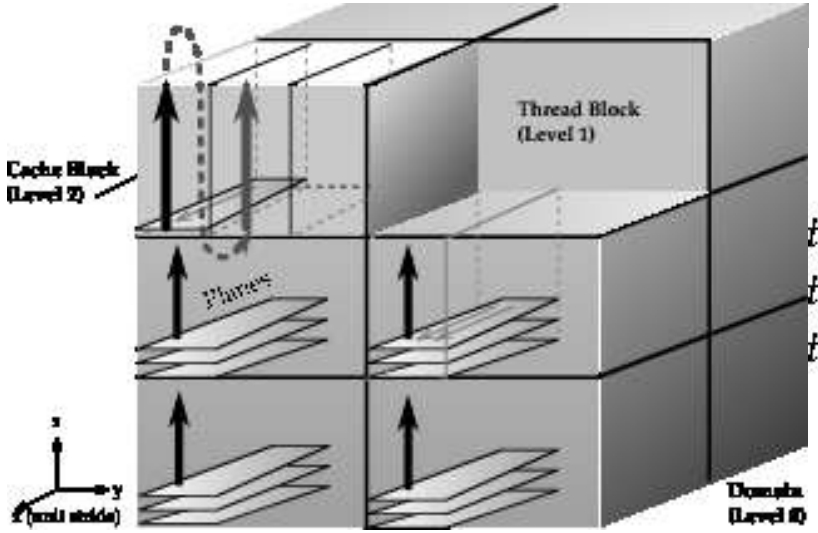




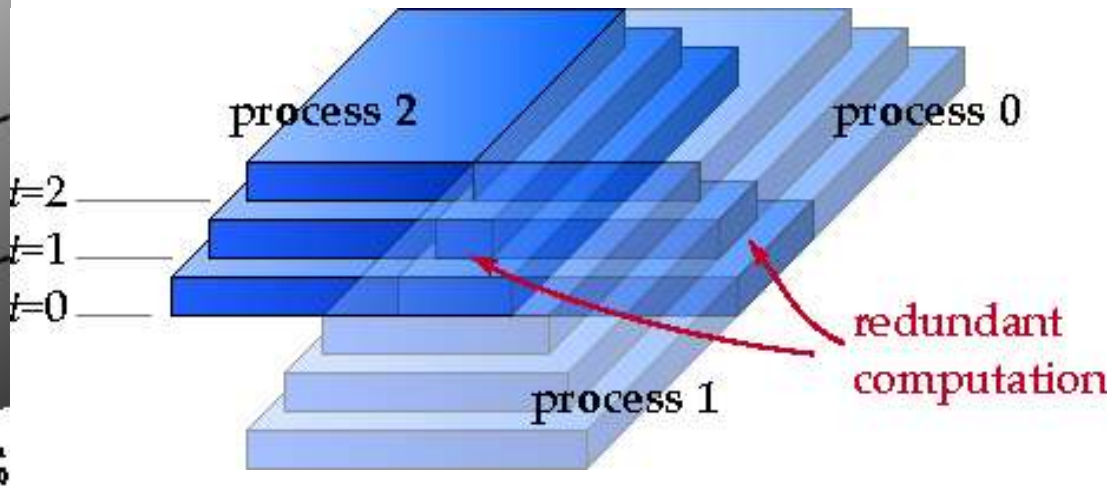
PERFORMANCE RESULTS

Earthquakes&seismic hazard / AWP-ODC Stencil Kernels

Kernel	Description	Discretization	Flops/Stencil	Arith. Intens.
uxx1	Velocity in one direction	4th order	20 Flops	0.83 Flop/Byte
xy1	Diagonal stress in one direction	4th order	16 Flops	0.80 Flop/Byte
xyz1	Stresses parallel to axes	4th order	90 Flops	2.04 Flop/Byte
xyzq	Stresses parallel to axes in viscous mode	4th order	129 Flops	1.61 Flop/Byte

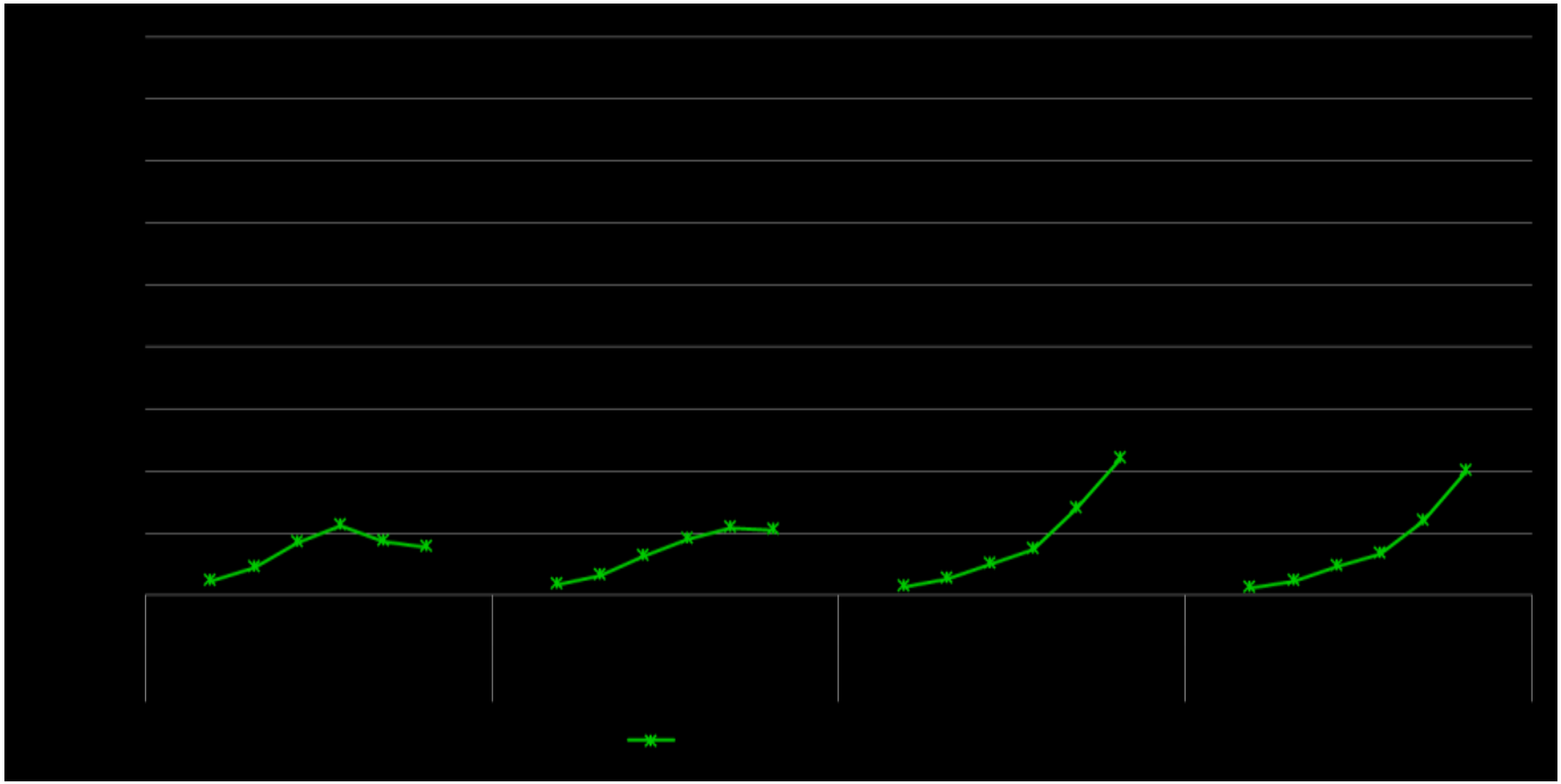


spatial blocking



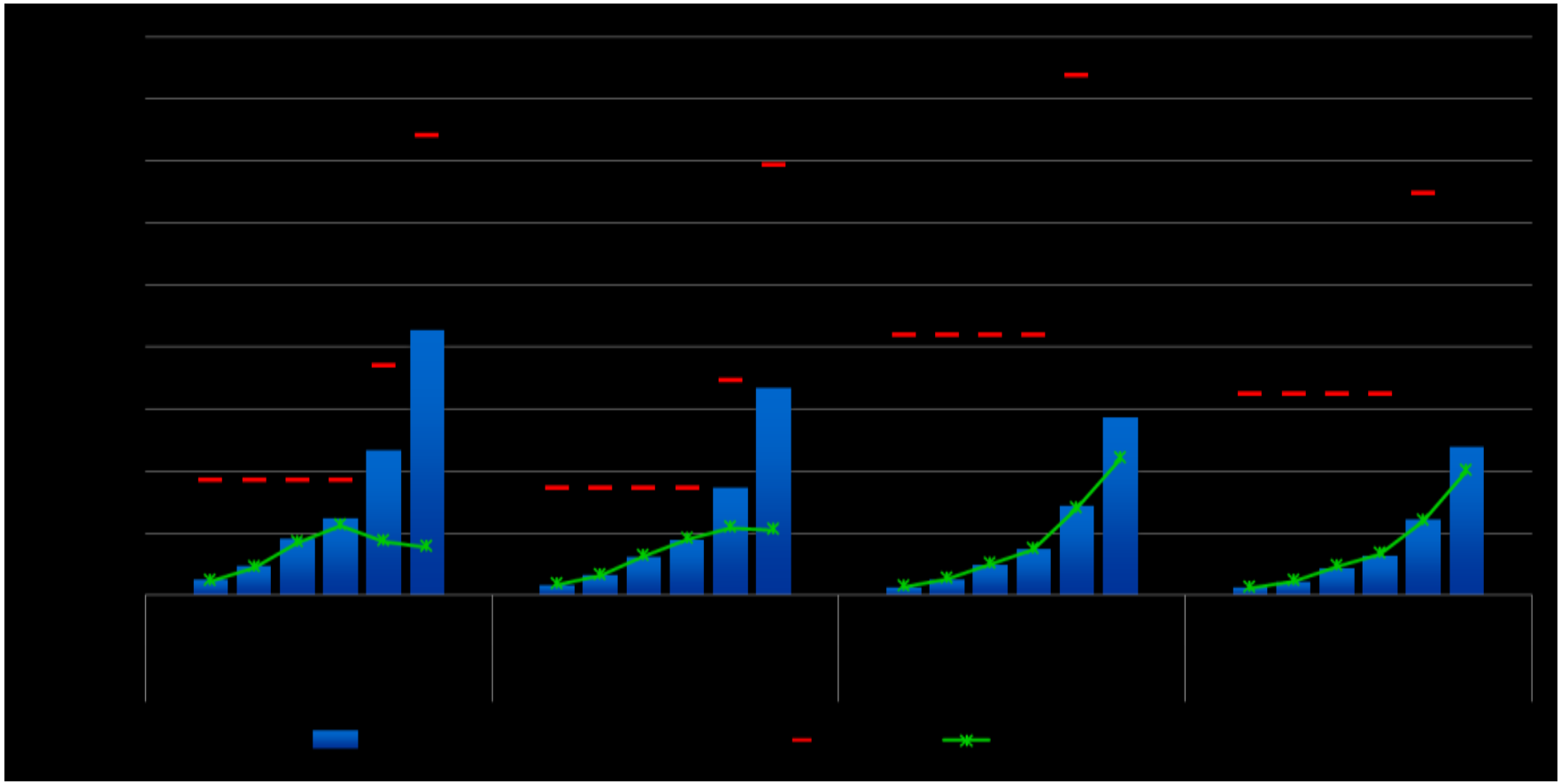
temporal blocking, Ax, A^2x, …, A^kx

Performance Benchmarks AWP-ODC Code on AMD Opteron "Magny Cours"



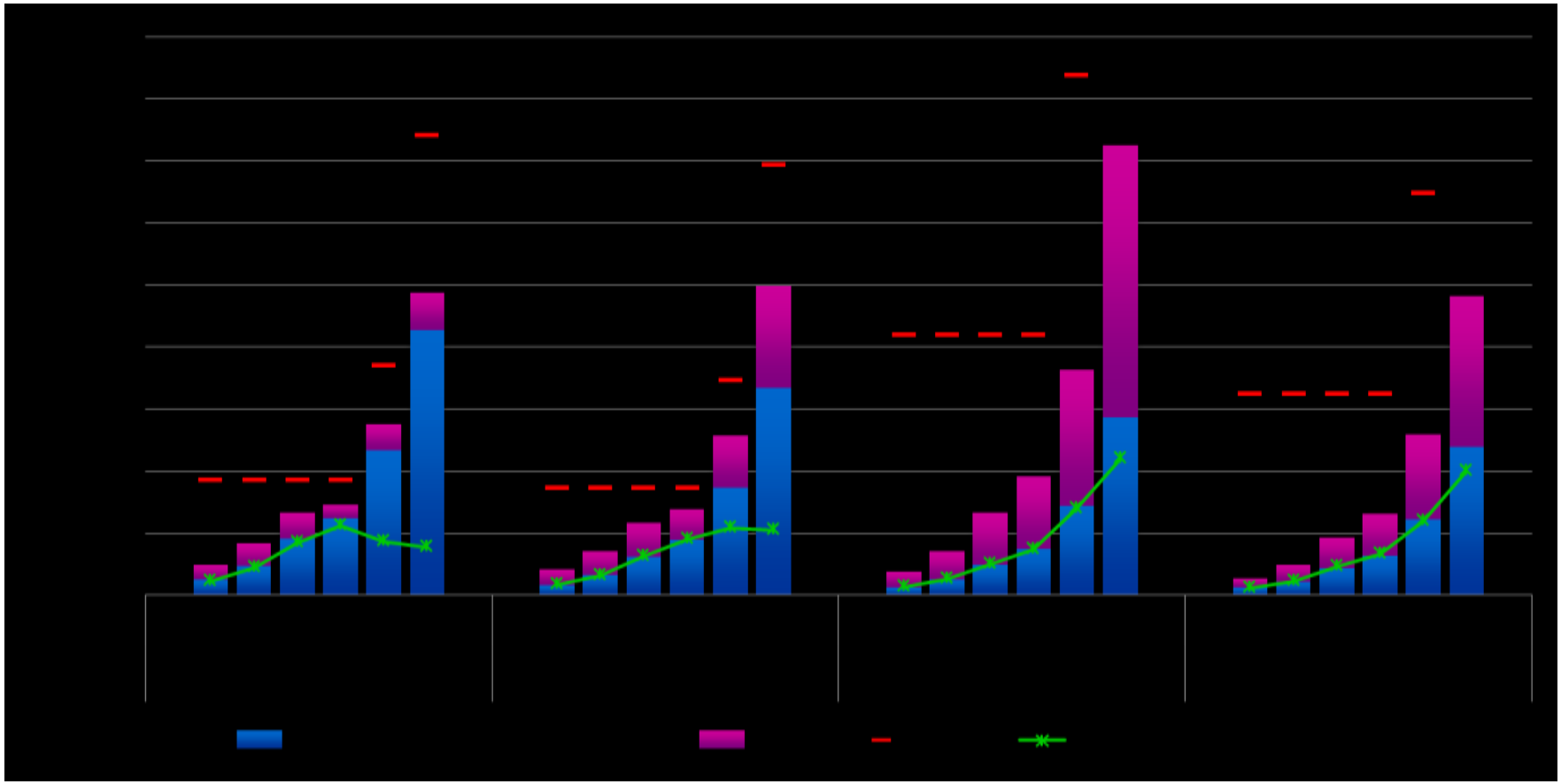
M. Christen, O. Schenk et al., *PATUS: A Code Generation and Autotuning Framework For Parallel Iterative Stencil Computations on Modern Microarchitectures*, IPDPS 2009, IPDPS 2010, IPDPS 2011

Performance Benchmarks AWP Code on AMD Opteron "Magny Cours"



M. Christen, O. Schenk et al., *PATUS: A Code Generation and Autotuning Framework For Parallel Iterative Stencil Computations on Modern Microarchitectures*, IPDPS 2009, IPDPS 2010, IPDPS 2011

Performance Benchmarks AWP Code on AMD Opteron "Magny Cours"



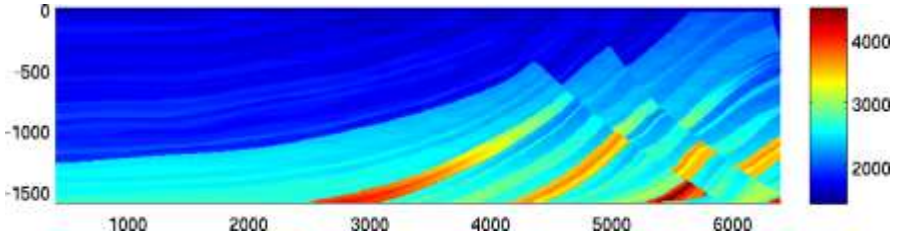
M. Christen, O. Schenk et al., *PATUS: A Code Generation and Autotuning Framework For Parallel Iterative Stencil Computations on Modern Microarchitectures*, IPDPS 2009, IPDPS 2010, IPDPS 2011

Stencil Code Library PATUS

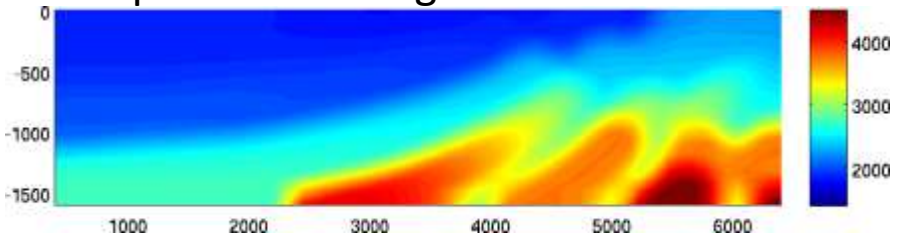
- Patus can be used to speed up stencil kernels, which can be re-integrated into the full application
- 2.4x between 4.7x speedup on the various kernel level on average (GCC 4.6.2 “-O3”)
- Threading will become more important in the future; hopefully Patus will help to generate good threaded stencil codes
- http://code.google.com/p/patus/

Realistic 3D Seismic Imaging (Marathon)

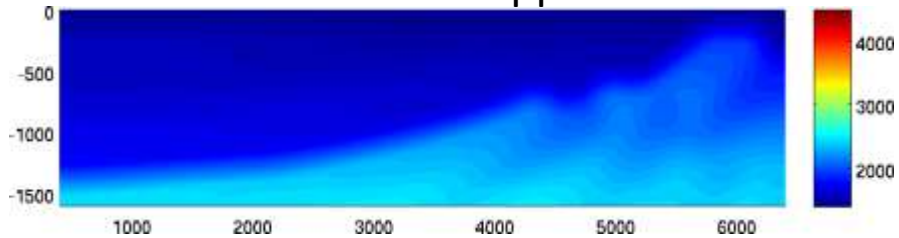
Marmousi «Truth»



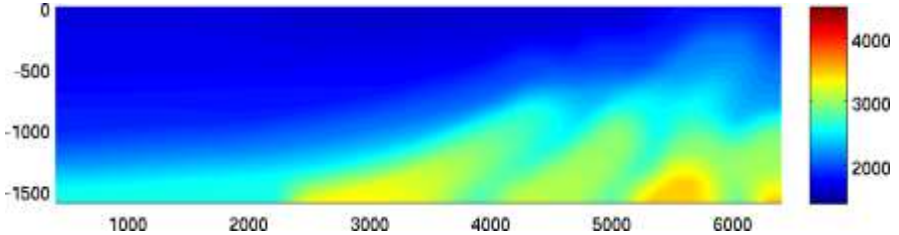
A priori knowledge: lower bound



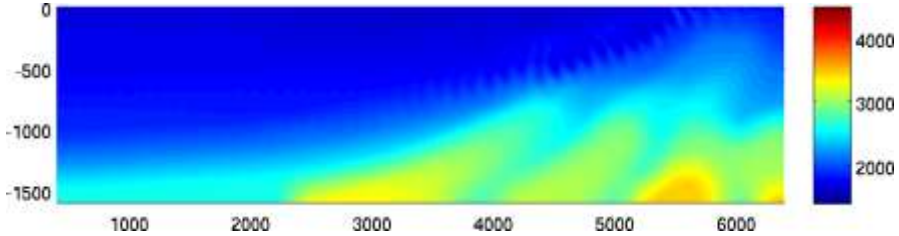
upper bound



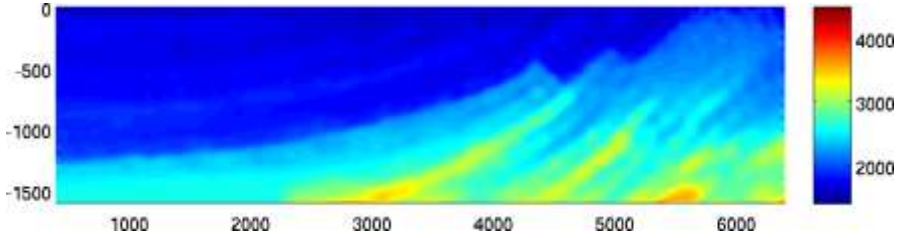
initial Model



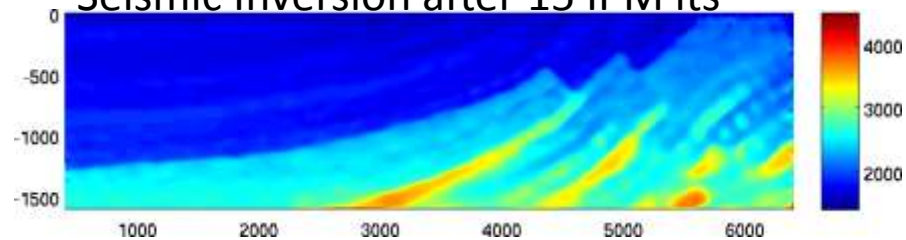
Seismic inversion after 5 IPM its



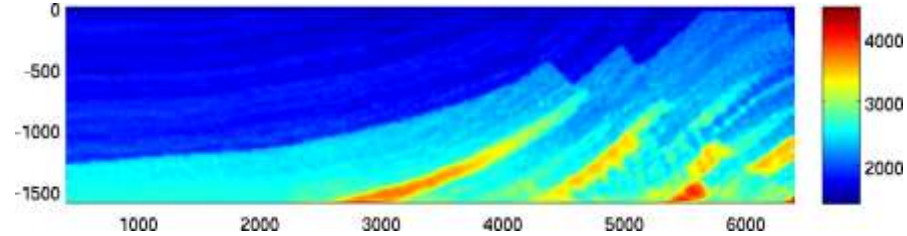
Seismic Inversion after 10 IPM its



Seismic Inversion after 15 IPM its



Seismic Inversion after 18 IPM its



Grote, Huber, S., *Towards Interior Point Methods for the Inverse Medium Problem on Massively Parallel Architectures*, ICCS 2011.

Summary and Outlook

Summary

- Code and **algorithmic optimization** necessary to tackle **global seismic tomography** (stencil codes, nonlinear optimization, etc.)

- **Stencil codes** occur in a wide range of HPC applications (seismic)
- **All** stencil kernels **are memory bound**, hence applying data locality techniques is crucial.
- **General code generation framework** for stencil codes of **arbitrary stencil** shapes for **different types of hardware**

Acknowledgments

- Swiss HP2C Petaquake project, SNF, DOE, IBM Faculty Award, Marathon Oil

Q & A

Thank you for your attention!

Questions?