Large-Scale Seismic Imaging on HPC Architectures: Applications, Algorithms and Software

Olaf Schenk

Institute of Computational Science

USI Lugano, Switzerland

March 29, 2012

Joint work with:
D. Giardini, T. Nissen-Meyer, L. Dalquer (ETH Zurich),
J. Tromp (Princeton), Y. Cui (San Diego Supercomputer Center),
A. Wächter (IBM Research), M. Christen, M.Rietmann (USI Lugano), P. Messmer (NVIDIA)

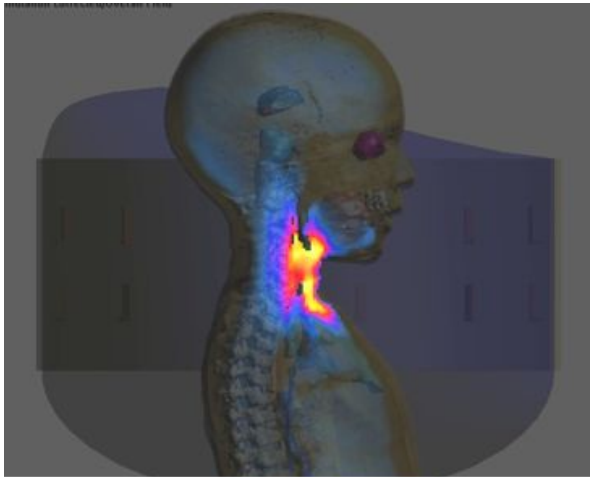
Outline

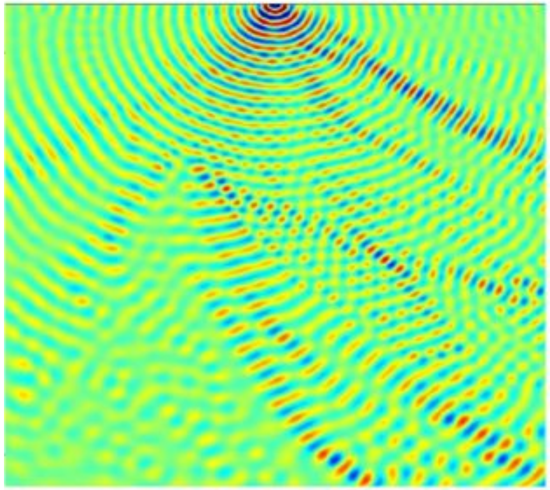
Seismic Imaging: Software and Inversion Algorithms on HPC Architectures

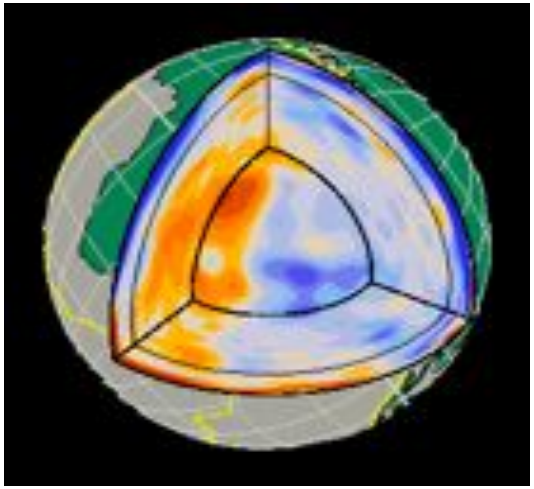
- Anelastic Wave Propagation Code AWP-ODC on XE6 («Rosa»)
- Spectral Element Wave Propagation SPECFEM on XK6 («TitanDev»)

- Full-Space Waveform Inversion Strategies for Seismic Imaging
- High-Resolution Seismic Imaging Results
- Conclusion / Future Work

HPC and PDE-Constrained Optimization (Inversion)







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

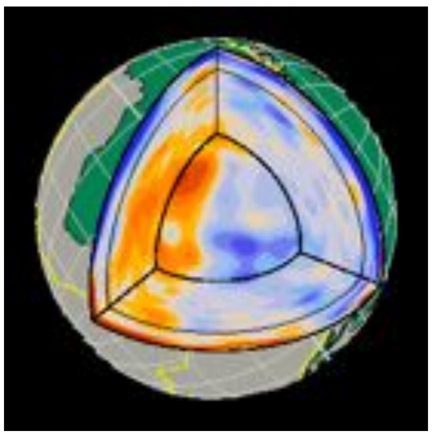
$$\min_{\mathbf{x}} F(\mathbf{x})$$
$$\text{subject to } c_i(\mathbf{x}) = 0 \quad \text{for } i \in \mathcal{E}$$
$$c_i(\mathbf{x}) \geq 0 \quad \text{for } i \in \mathcal{I}$$

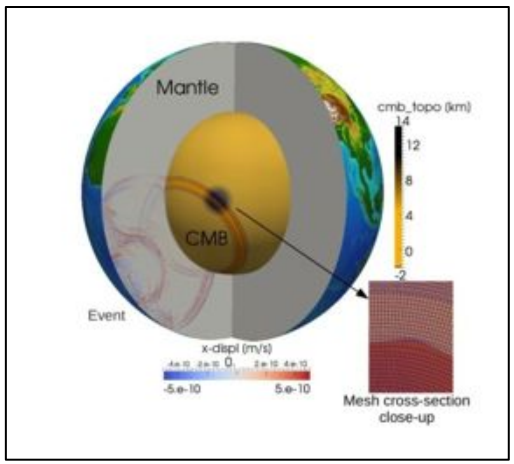
Application: High-Resolution 3D-Seismic Imaging

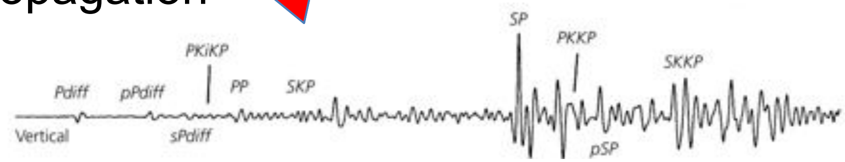
Forward modeling

Seismic wave propagation

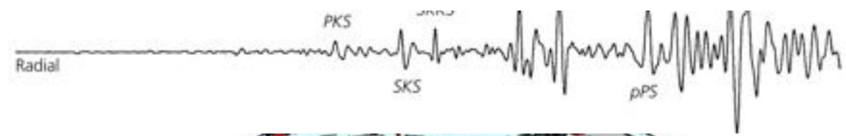
Model space: 3D wavespeeds

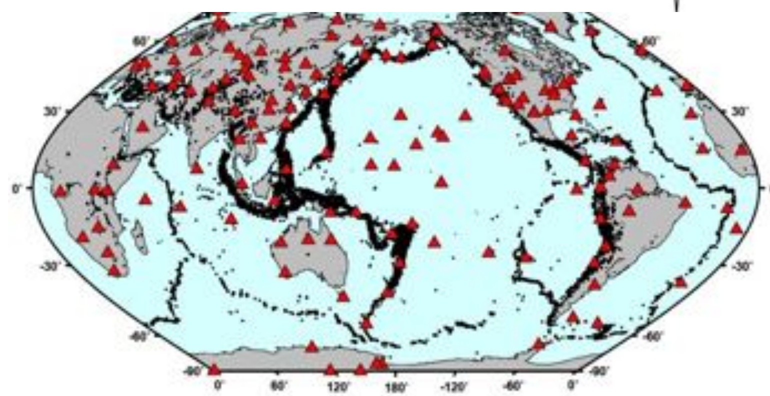






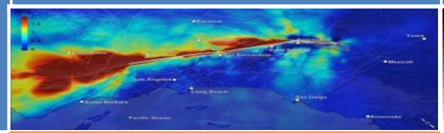
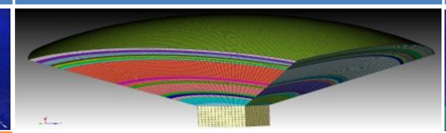
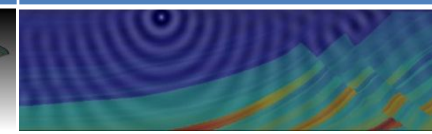
Data space: ground displacements



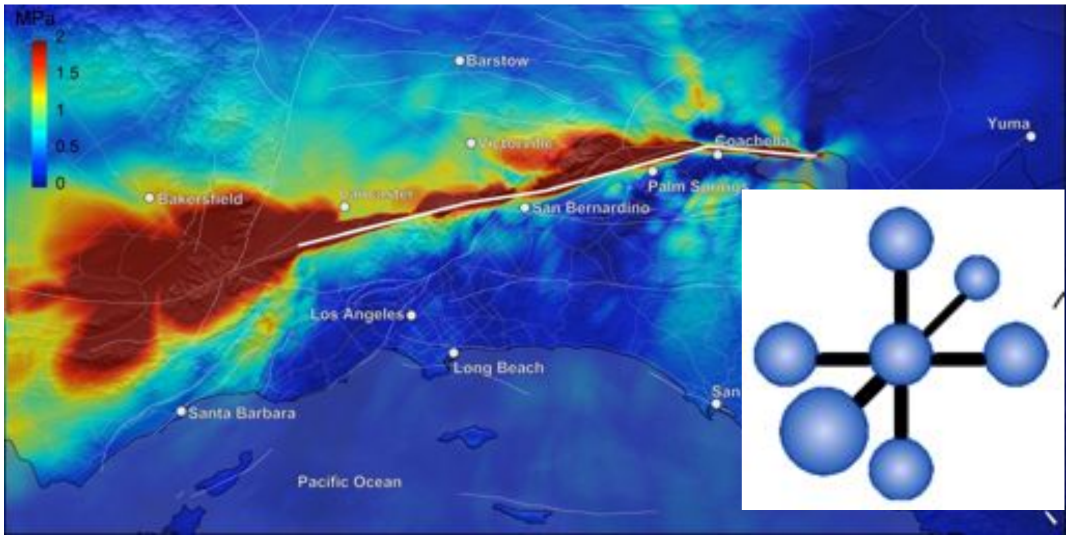


Inverse modeling/imaging

Overview of Codes in Computational Seismology and Optimization

	Nuclear Safety	Global Tomography	Seismic Inversion
Motifs	AWP-ODC Finalist Gordon-Bell SC'10	SPECFEM Gordon-Bell SC'03 Finalist Gordon-Bell SC'08	IPOPT Wilkinson-Price 2011 for Numerical Software
			
Structured Grids/ Stencil			
Unstructured Grids			
Spectral Methods			
Dense Linear Algebra			
Sparse Linear Algebra			
Particles			
Monte Carlo			

Application: Earthquakes&seismic hazard / AWP-ODC



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

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

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

Image courtesy:
Y. Cui, Southern California Earthquake Center

- 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.
- Authors: Y. Cui et.al. (Southern California Earthquake Center), L. Dalguer (ETHZ)

Scalability of the AWP-ODC Stencil-Code on Jagua

TABLE 2
EVOLUTION OF AWP-ODC

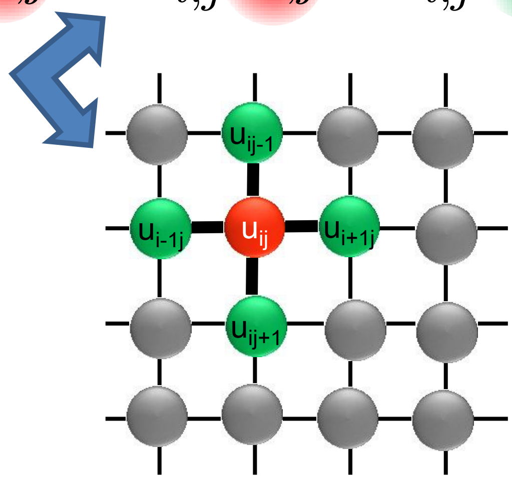
Year	Code version	Simulations	Optimization	SCEC alloc. SUs	Sustain. Tflop/s
2004	1.0	TeraShake-K	MPI tuning	0.5M	0.04
2005	2.0	TeraShake-D	I/O tuning	1.4M	0.68
2006	3.0	PN MQuake	partition. mesh	1.0M	1.44
2007	4.0	ShakeOut-K	incorp. SGSN	15M	7.29
2008	5.0	ShakeOut-D	asynchronous	27M	49.9
2009	6.0	W2W	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: **300 TFlop/s sustained** on **220k cores** (Jaguar)

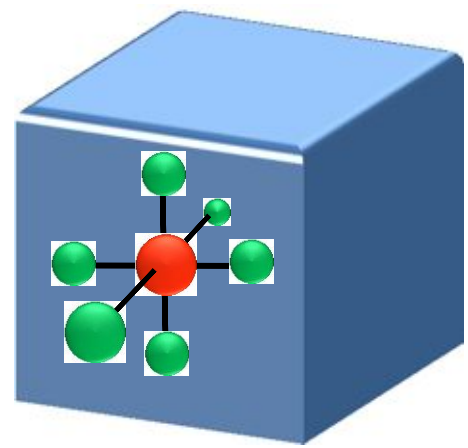
What is a Stencil?

Weighted sum of subset of neighbors of a grid point

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



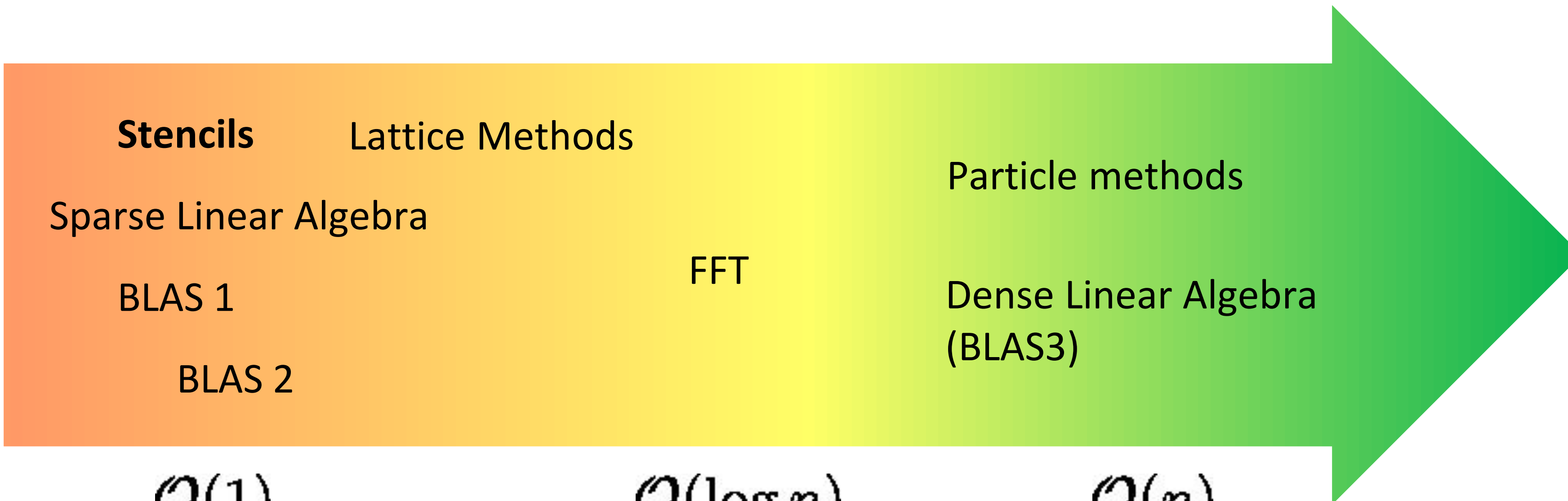
5-point stencil in a regular 2D grid



7-point stencil in 3D

Challenge: Arithmetic Intensity

Arithmetic Intensity := Flops / Transferred Data

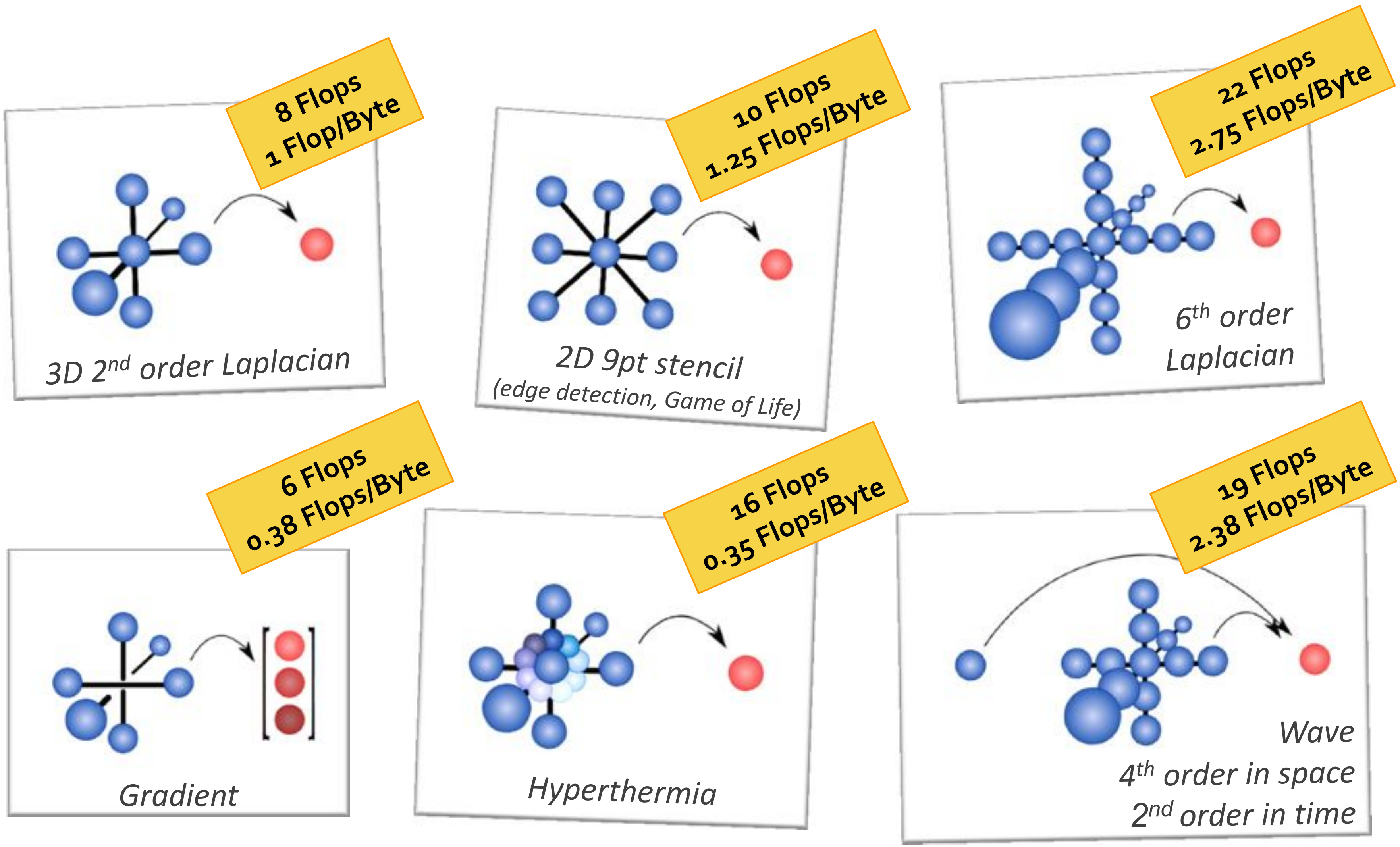


$\mathcal{O}(1)$ $\mathcal{O}(\log n)$ $\mathcal{O}(n)$

Low arithmetic intensity
$\Rightarrow$ memory bandwidth bound

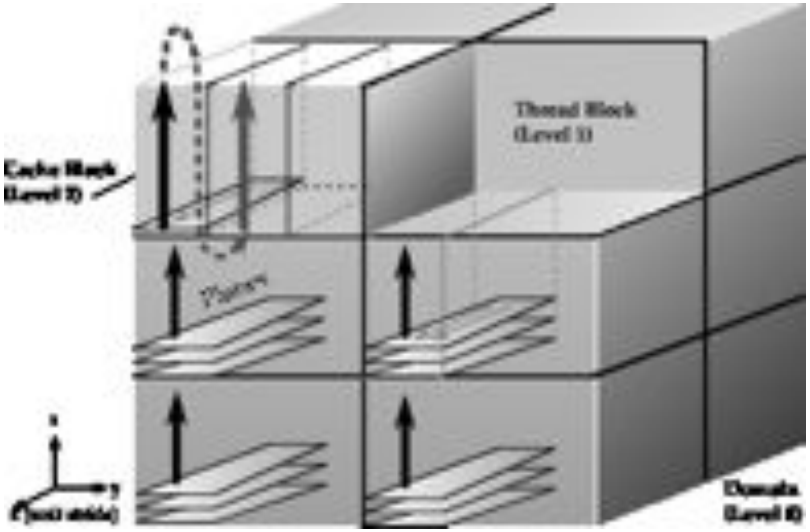
High arithmetic intensity
$\Rightarrow$ processor bound

Stencil Variety

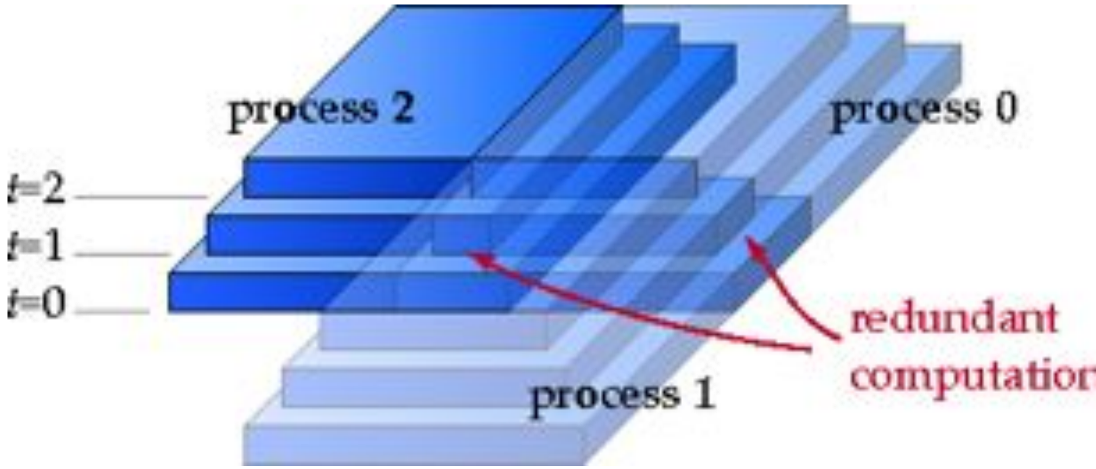


Earthquakes&seismic hazard / AWP-ODC Stencil Kernels

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

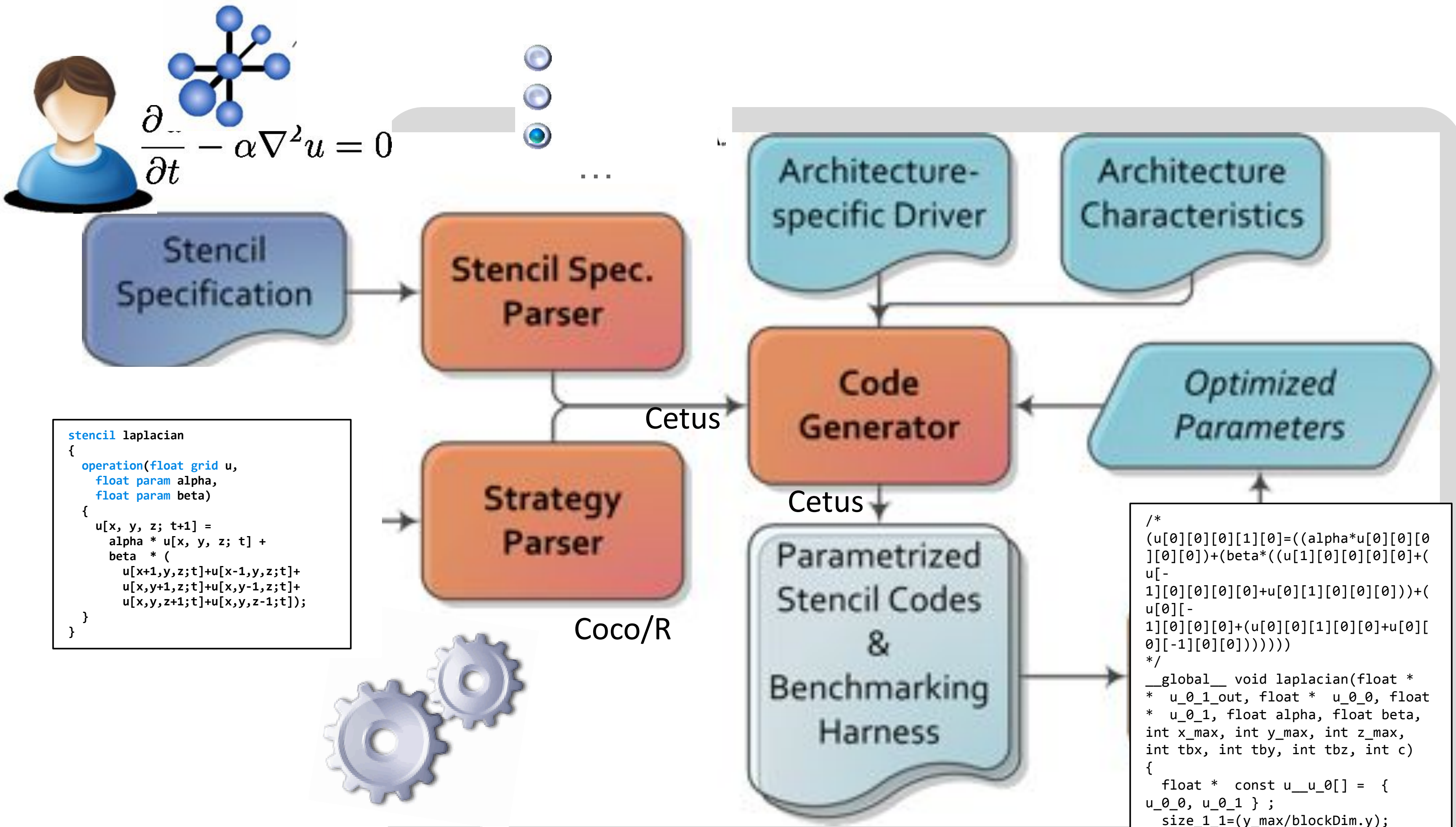


spatial blocking



temporal blocking, $Ax, A^2x, \ldots, A^kx$

Earthquakes&seismic hazard / AWP-ODC Code

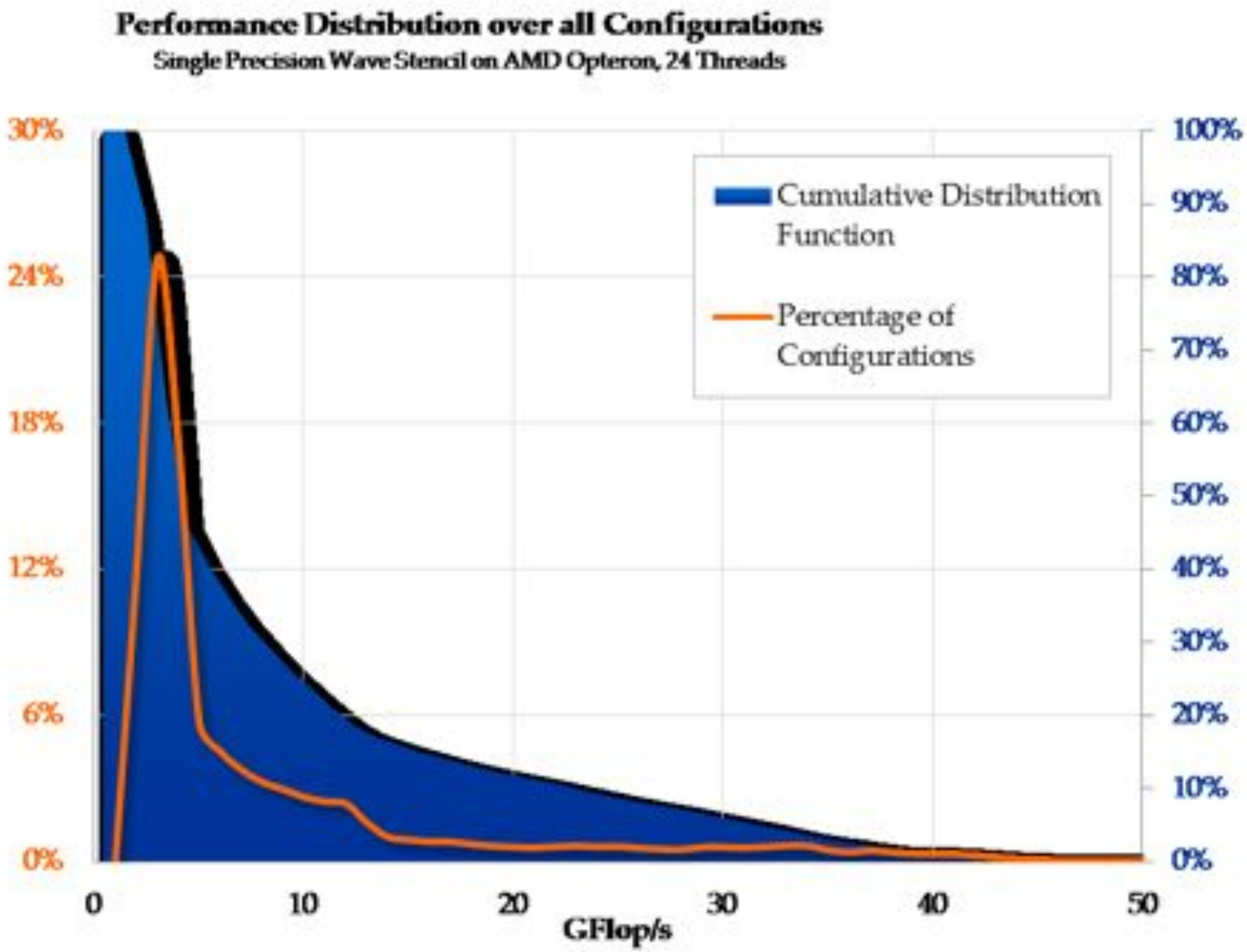



```
stencil laplacian
{
  operation(float grid u,
    float param alpha,
    float param beta)
  {
    u[x, y, z; t+1] =
      alpha * u[x, y, z; t] +
      beta  * (
        u[x+1,y,z;t]+u[x-1,y,z;t]+
        u[x,y+1,z;t]+u[x,y-1,z;t]+
        u[x,y,z+1;t]+u[x,y,z-1;t]);
  }
}
```

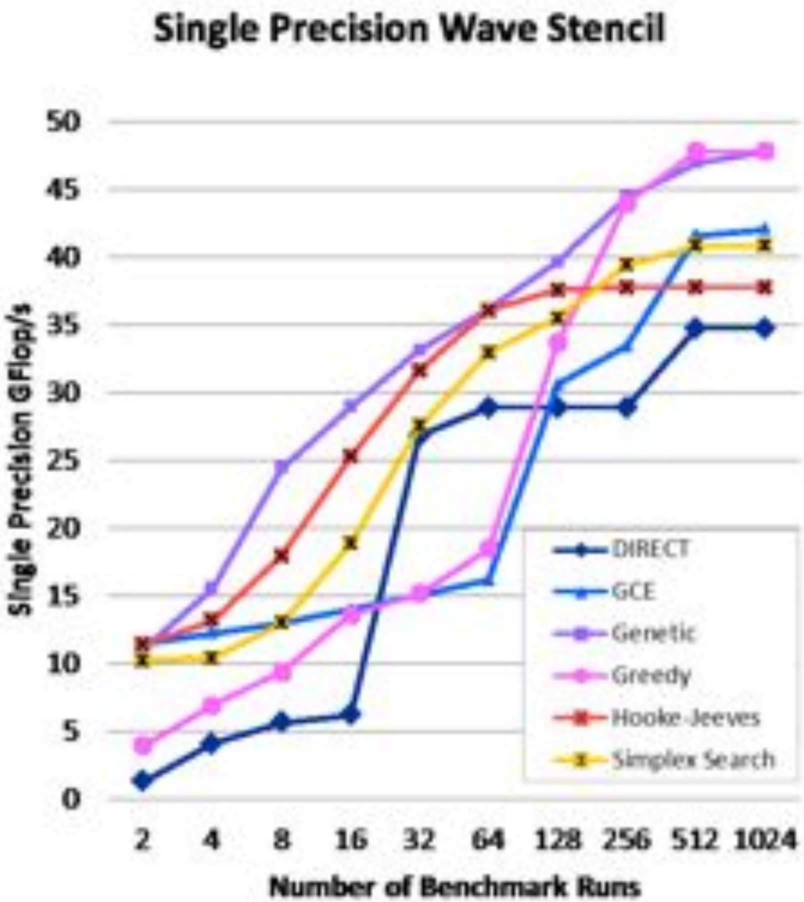
```
/*
(u[0][0][0][1][0]=((alpha*u[0][0][0
][0][0])+(beta*((u[1][0][0][0][0]+(
u[-
1][0][0][0][0]+u[0][1][0][0][0]))+(
u[0][-
1][0][0][0]+(u[0][0][1][0][0]+u[0][
0][-1][0][0]))))))
*/
__global__ void laplacian(float *
*  u_0_1_out, float *  u_0_0, float
*  u_0_1, float alpha, float beta,
int x_max, int 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);
```

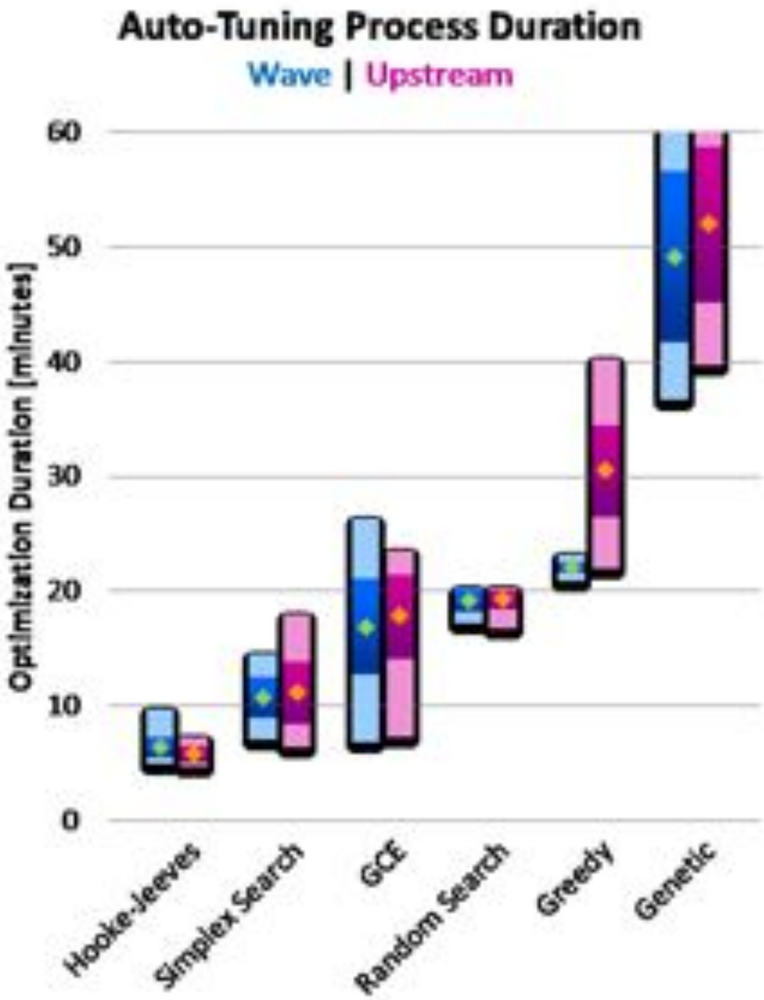
M. Christen, O. Schenk, *PATUS: A Code Generation and Autotuning Framework For Parallel Iterative Stencil Computations on Modern Microarchitectures*, IPDPS 2009, 2010, 2011

Auto-Tuning (Single-Precision Wave Stencil)

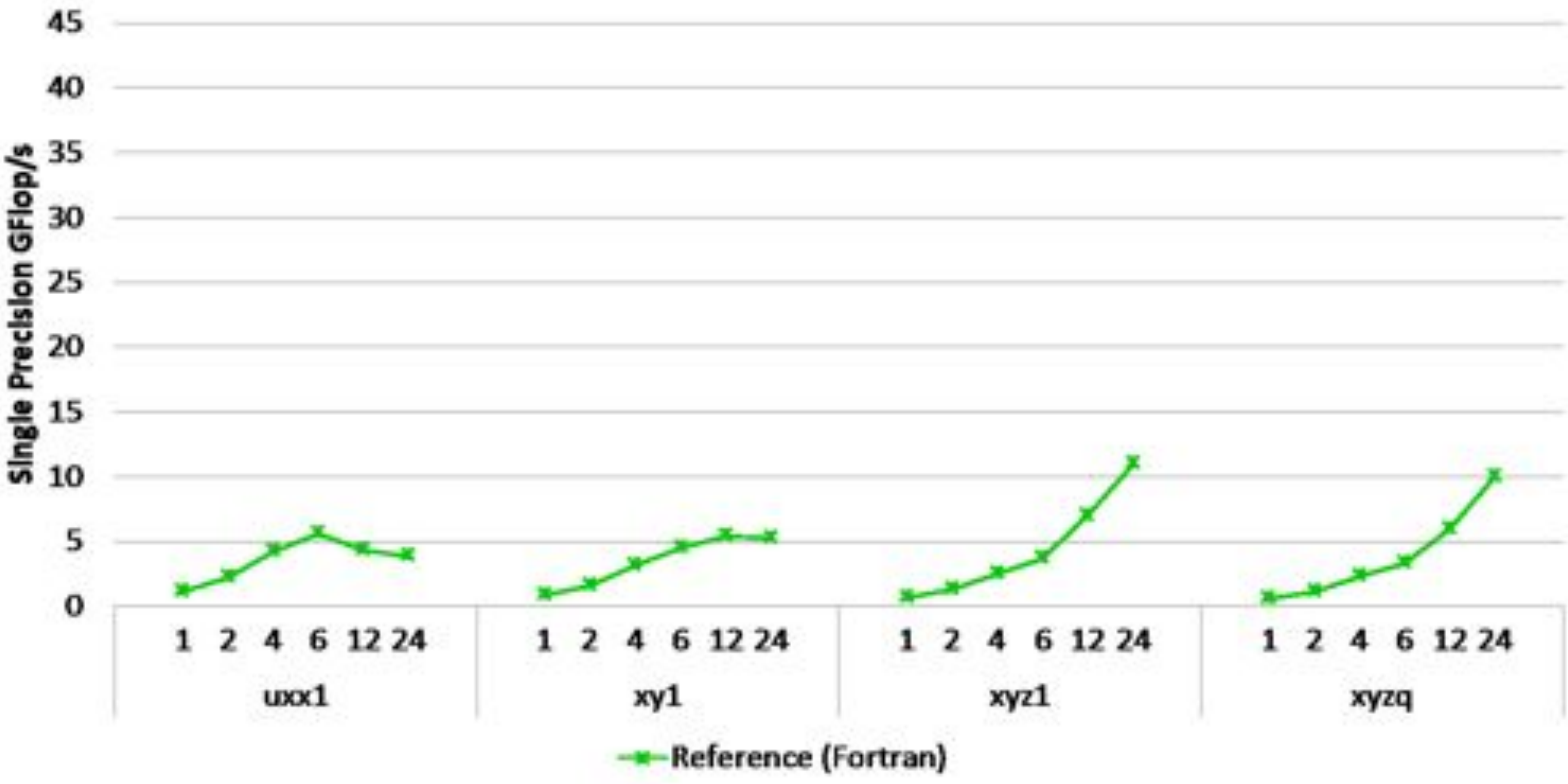


Search Methods (Single-Precision Wave Stencil)



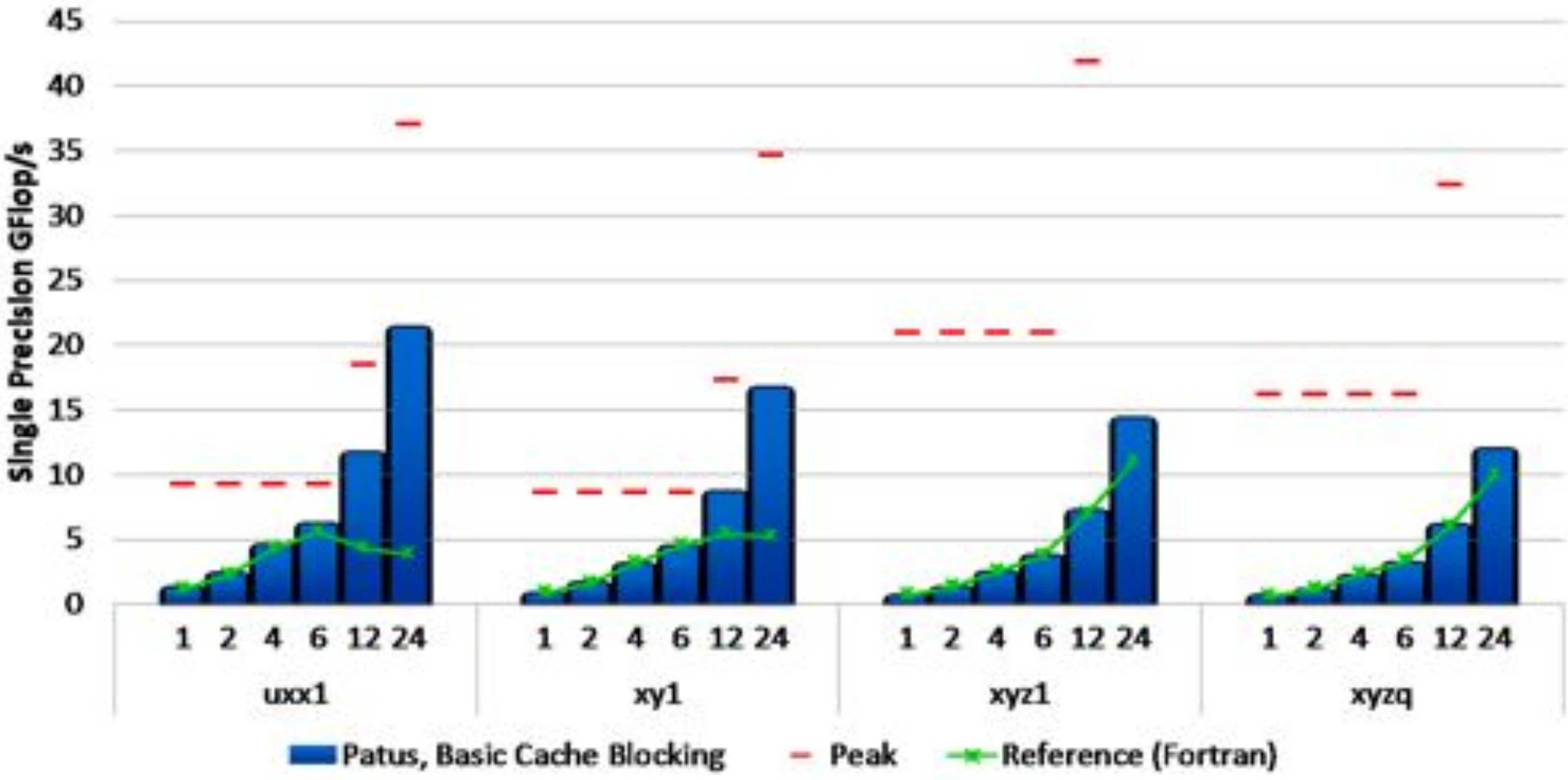


AWP-ODC Code on AMD Opteron "Magny Cours"



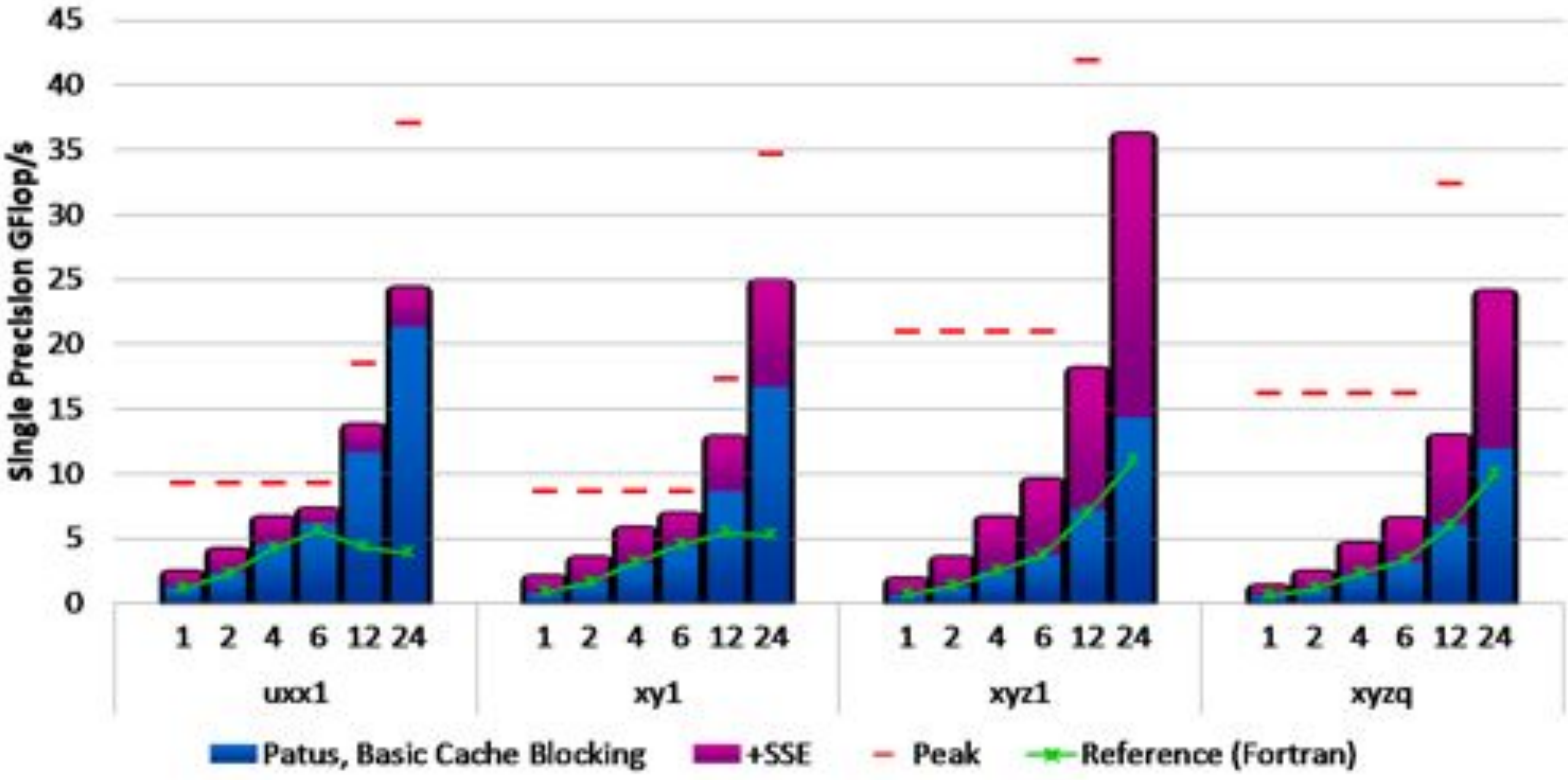
M. Christen, O. Schenk et al., *PATUS: A Code Generation and Autotuning Framework For Parallel Iterative Stencil Computations on Modern Microarchitectures*, IPDPS 2009, 2010, 2011

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, 2010, 2011

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, 2010, 2011

SPECFEM - Elastic Wave Propagation Code

Method: ***Spectral-elements***

- Highly efficient/scalable, explicit time marching
- Gordon Bell prize 2003
- solves all relevant physics at local to global scale
- open-source (geodynamics.org)
- used by several hundred groups
- Main Developers: D. Komatitsch (CNRS), D. Peter (Princeton), M. Rietmann (USI), J. Tromp (Princeton)

Computational cost:

Continental scale:

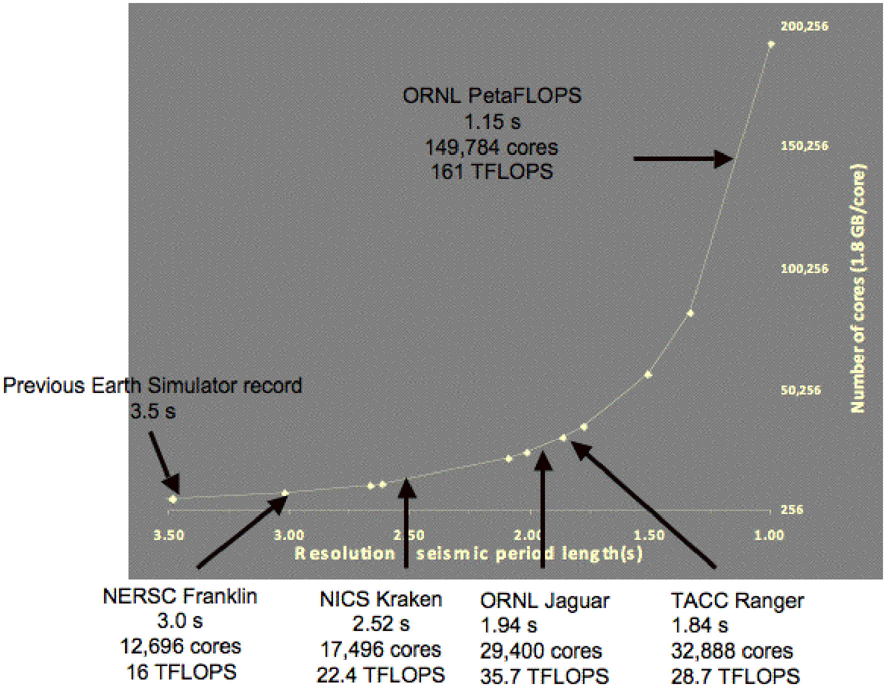
1 simulation: 3 hrs on 400 cores

Inverse problem: 12 x 10^6 CPU-hrs **(?!)**

Global scale:

1 simulation: 10 hrs on 1000 cores

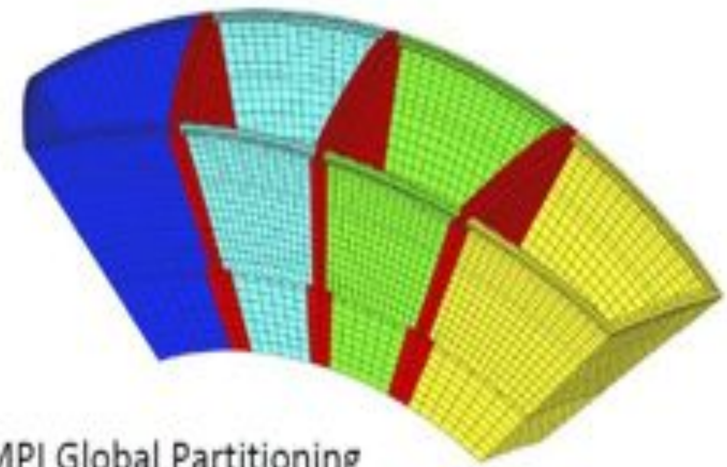
Inverse problem: 300 x 10^6 CPU-hrs



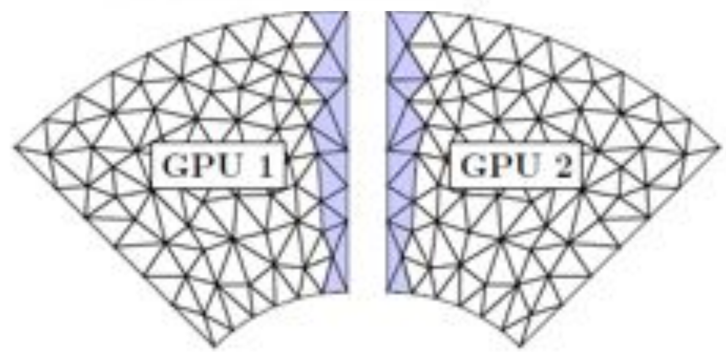
(Slide: T. Nissen-Meyer ETHZ)

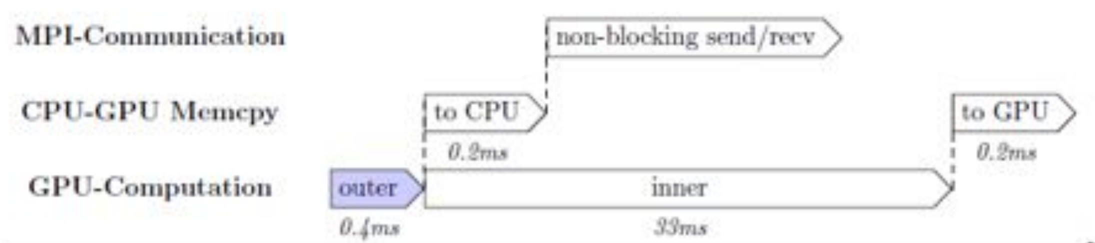
SPECFEM - Partitioning and Overlapped Halo Exchange

- Spectral Element Method
  - High order finite element method
  - Highly parallel, no global linear solve
- Arbitrary unstructured hexahdedral meshes
- Meshing using CUBIT
- MPI parallel
  - Mesh decomposition using SCOTCH
- CUDA accelerated
  - Heaviest kernel: Force comp.
- ~ 319*125 = 40kFLOPs per element



MPI Global Partitioning

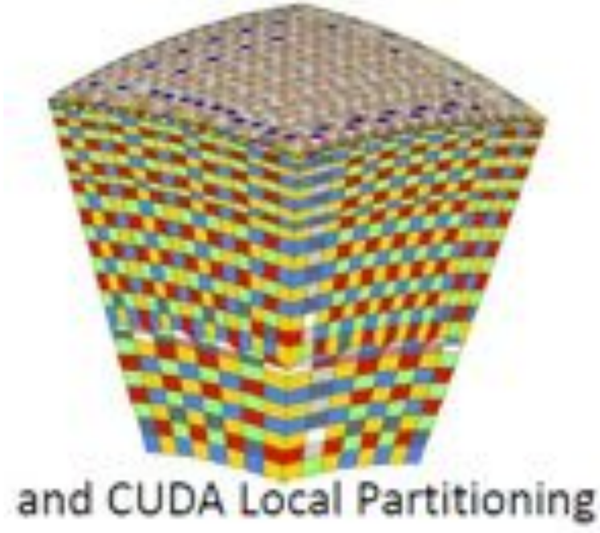


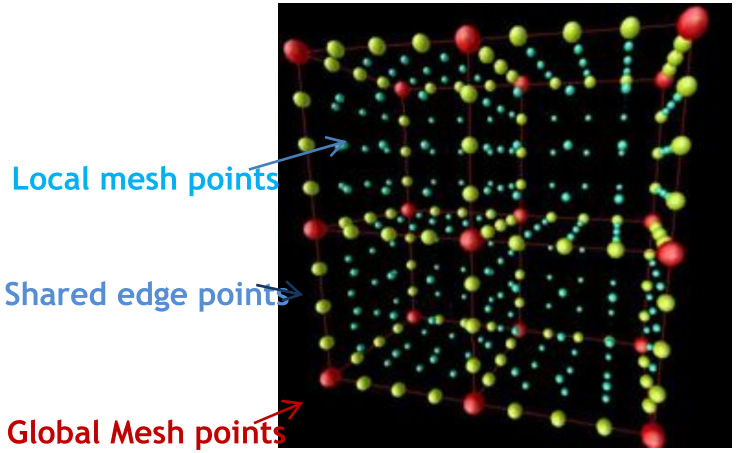


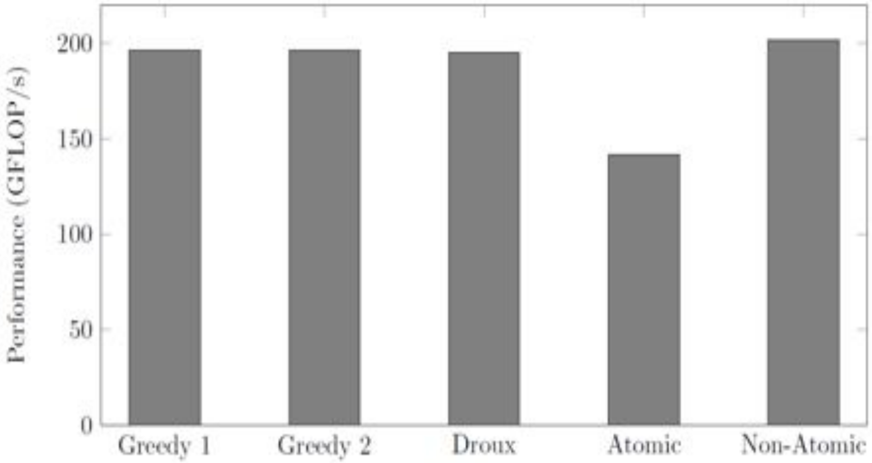
(Slide: T. Nissen-Meyer ETHZ)

SPECFEM – GPU Partitioning & Coloring

- One Block per Element
- One thread per interpolation point
- Interpolation weights in shared memory
  - 6k per element
- Race condition at element boundaries resolved by **mesh coloring**
  - Atomics could be used as well

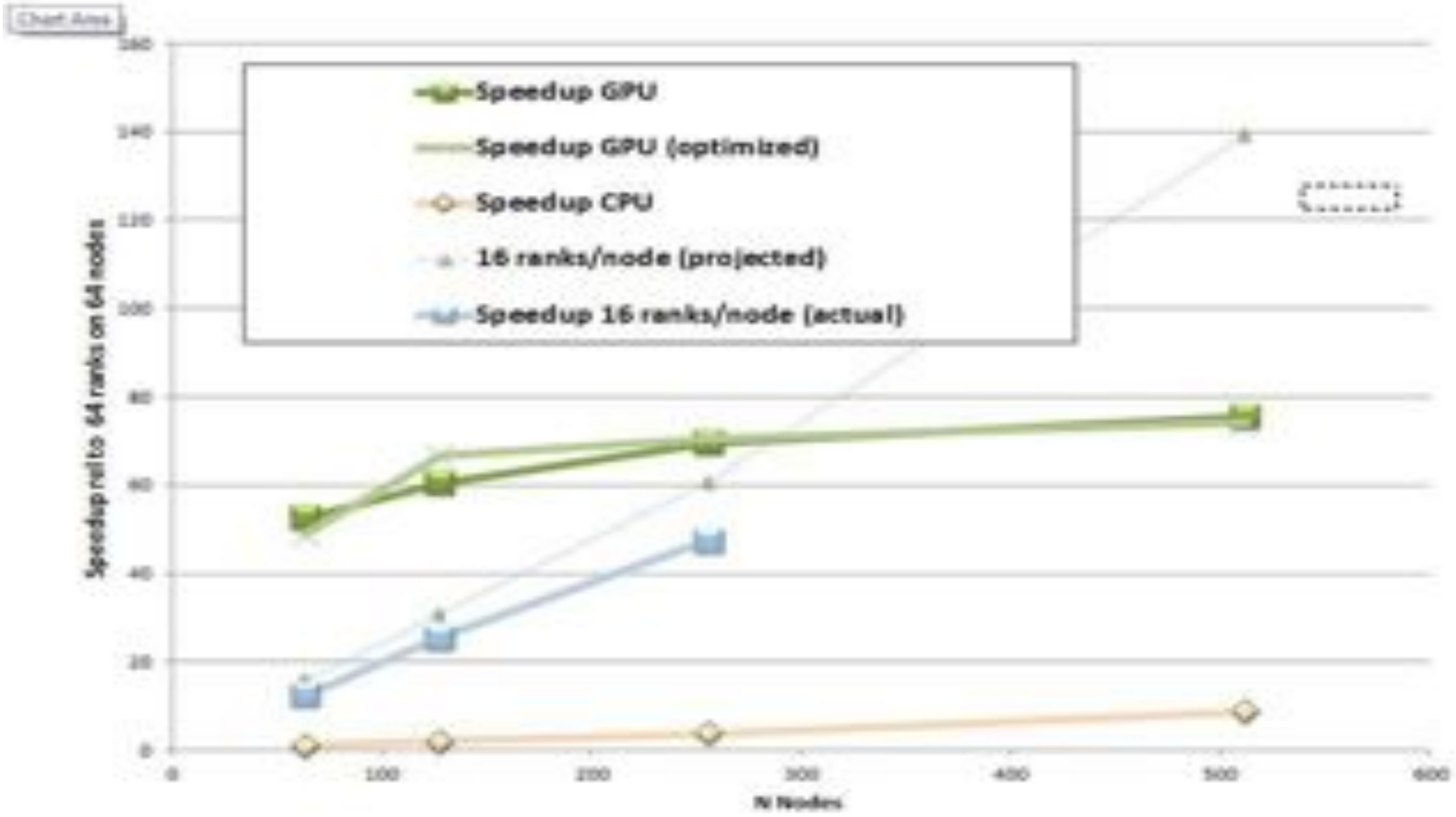






(Slide: T. Nissen-Meyer ETHZ)

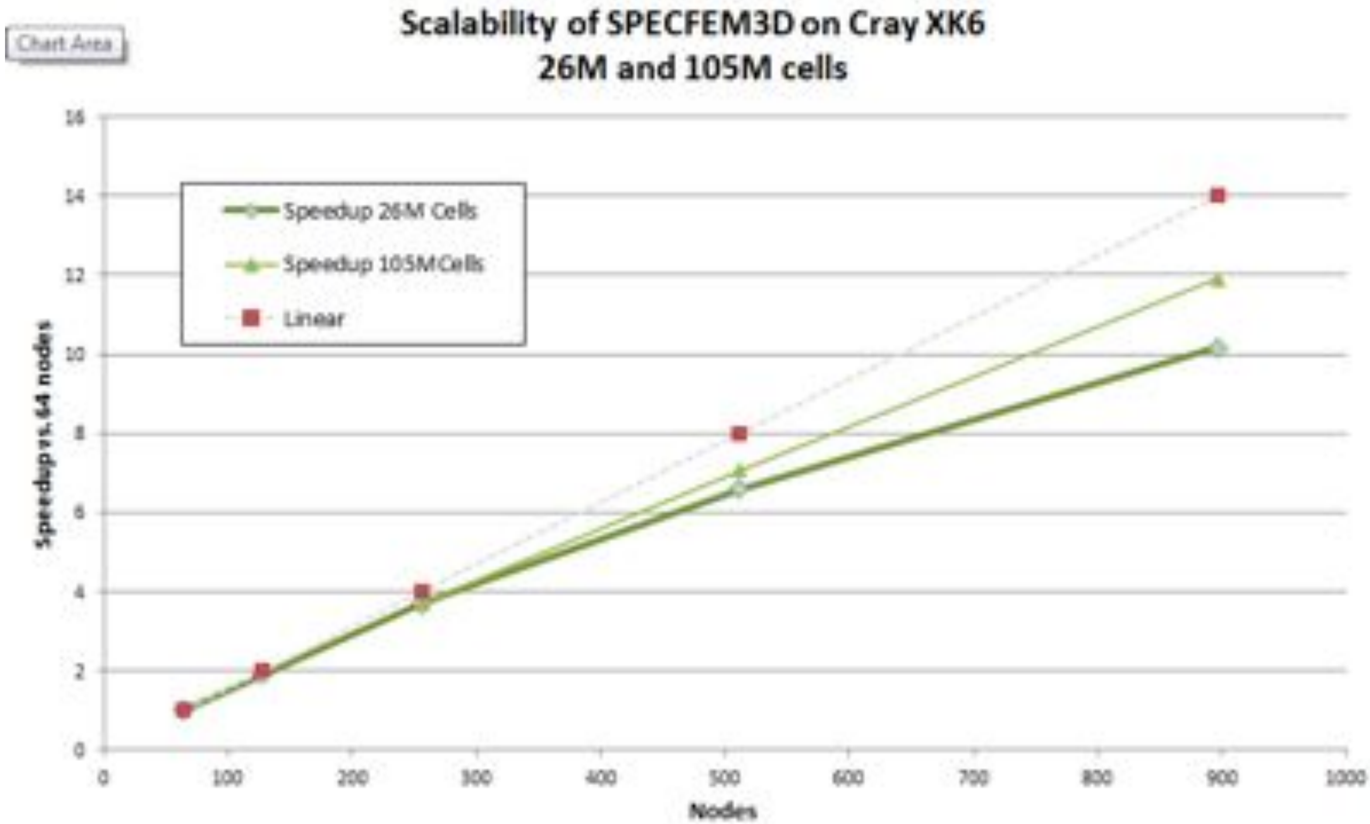
SPECFEM - Strong scaling on ORNL Titan, small problem



- Speedup of ~4.2 compared to 16 core CPU socket
- Small Mesh: 300 K elements

(Slide: P. Messmer NVIDIA)

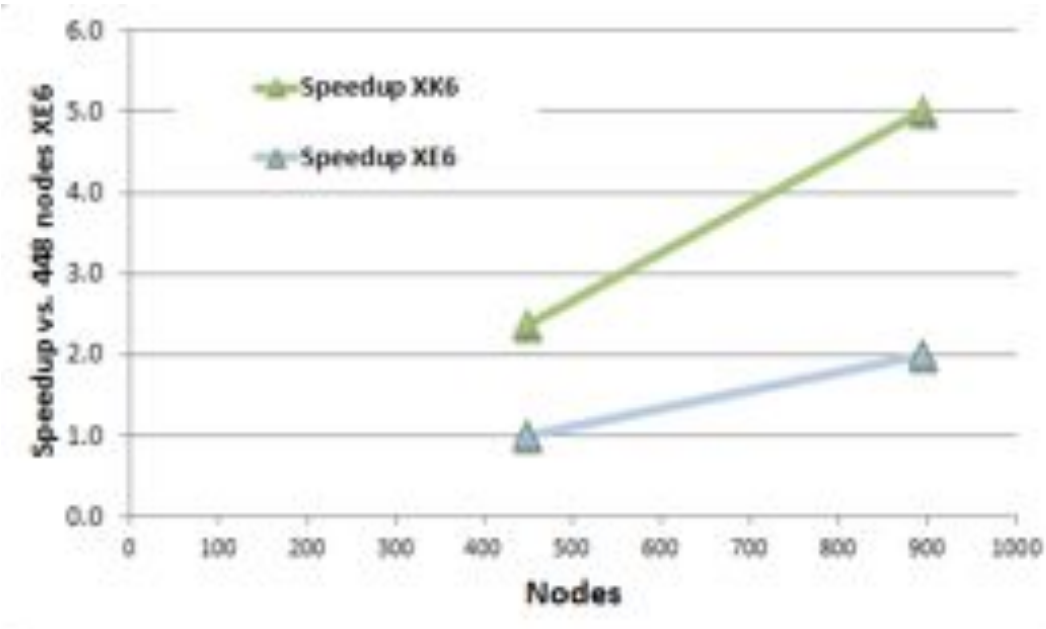
SPECFEM - Strong scaling on ORNL Titan, large problem



- Two problem sizes: 26 M cells and 105 Mcells
- Runs close to full system size (largest run on 896 GPU)

(Slide: P. Messmer NVIDIA)

SPECFEM - Comparison between XE6 and XK6

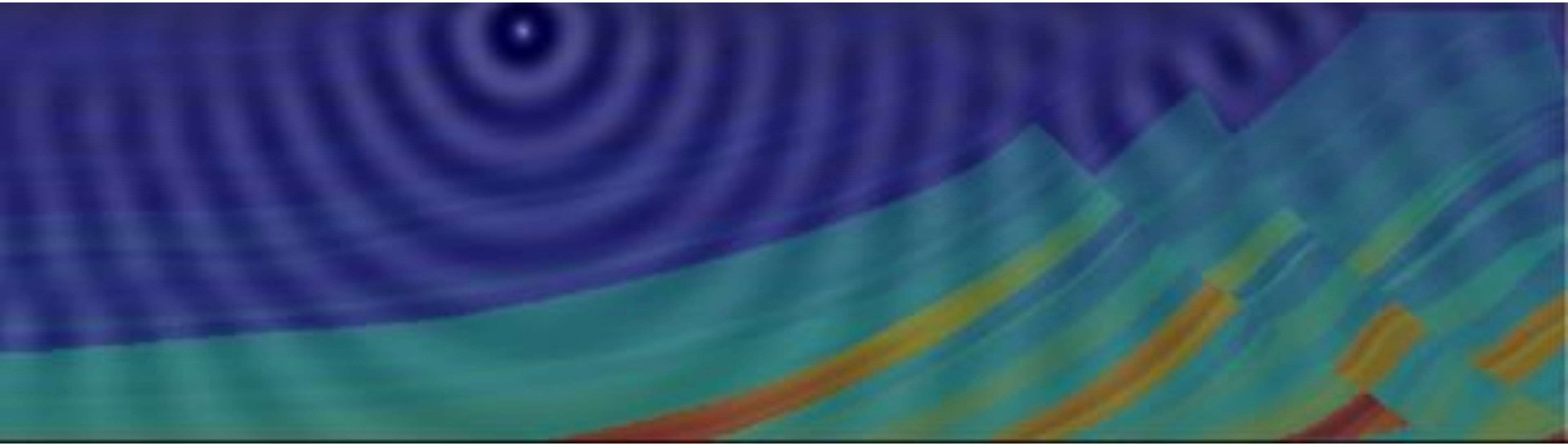


XK6 vs XK6-without-GPU speedup		XK6 vs XE6 speedup	
Single-node	Multi-node	Single-node	Multi-node
4.2 (64 nodes)	4.7 (896 nodes)	1.9 (64 nodes)	2.5 (896 nodes)

- 110M elements mesh
- Completely asynchronous halo exchange
- 144 GFLOPs/GPU -> 130 TFLOPs on 896 GPUs

(Slide: P. Messmer NVIDIA)

Seismic Inversion



- Problem to solve:

$$\min_{\mathbf{y}_i,\mathbf{k}} F(\mathbf{y}_i,\mathbf{k}) = \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{y}_{\mathbf{m},i} - \mathbf{y}_i(x_m)\|_2^2 + \frac{\tau}{2}R(k)$$

$$\text{subject to } \mathbf{A}(\mathbf{k})\mathbf{y}_i = \mathbf{f}_i(\mathbf{k}) \qquad \text{for } i = 1, \dots, N$$

$$\mathbf{b}_- \le \mathbf{k} \le \mathbf{b}_+$$

- Fast convergence: Hessian needed
- Reduced space: Hessian dense !!!
- Full-space optimization approach: Optimization variable x (y_1; ..., y_n, k)

Seismic Inversion

- Barrier subproblem:

$$\min_{\mathbf{x},\mathbf{s}} F(\mathbf{x}) - \mu \sum_{i \in \mathcal{I}} \log s_i$$

$$\text{subject to} \quad \mathbf{c}_{\mathcal{E}}(x) = 0$$

$$\mathbf{c}_{\mathcal{I}}(x) - \mathbf{s} = 0$$

- Lagrangian:

$$\mathcal{L}(\mathbf{x}, \mathbf{s}, \lambda) = F(\mathbf{x}) - \mu \sum_{i \in \mathcal{I}} \log s_i + \sum_{i \in \mathcal{E}} \lambda_i c_i(x) + \sum_{i \in \mathcal{I}} \lambda_i (c_i(x) - s_i)$$

- KKT Condition:

$$\nabla_x \mathcal{L} = \nabla_x F(\mathbf{x}) + \sum_{i \in \mathcal{E}} \lambda_i \nabla_x c_i(\mathbf{x}) + \sum_{i \in \mathcal{I}} \lambda_i \nabla_x c_i(\mathbf{x}) = \mathbf{0}$$

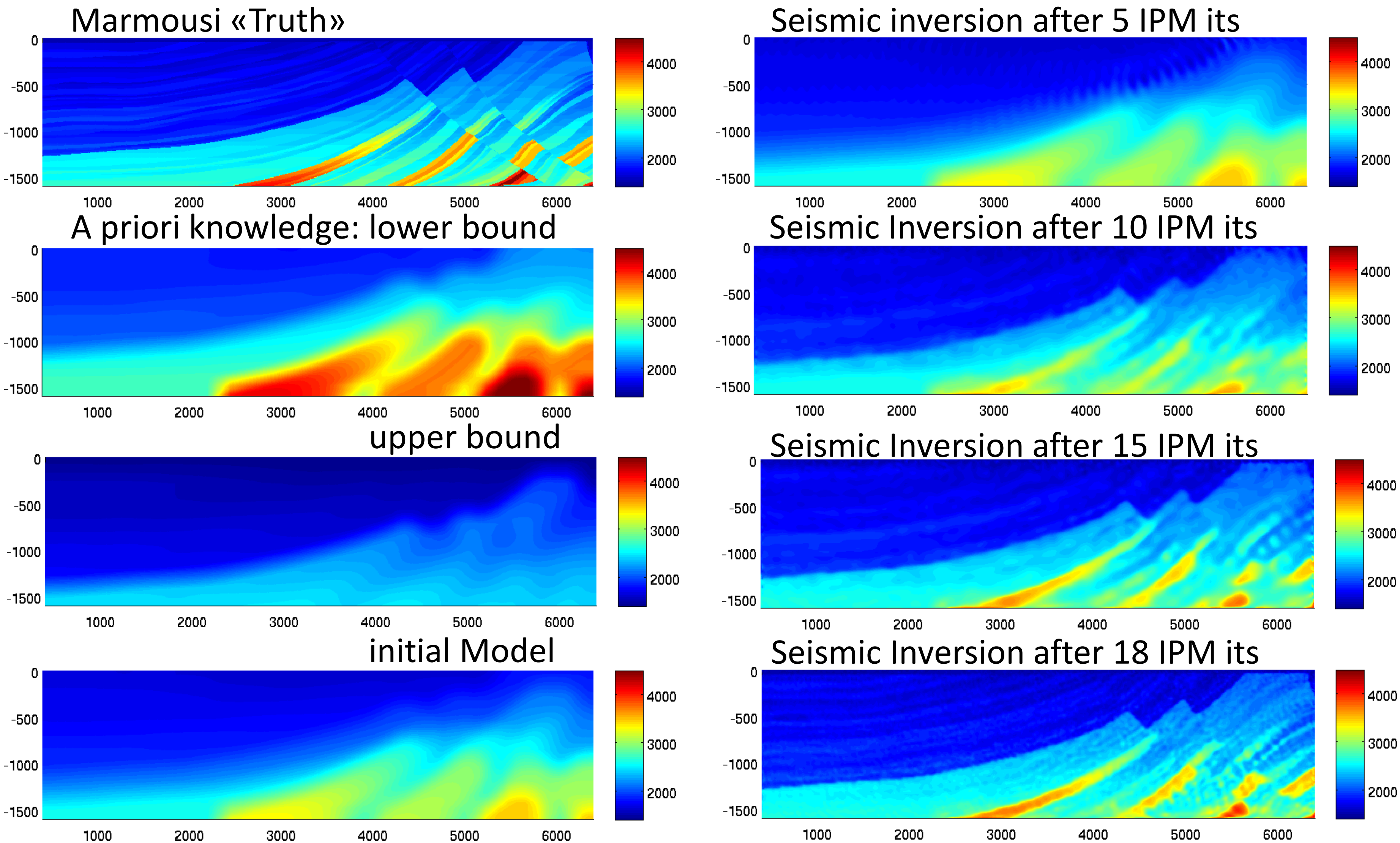
$$\nabla_s \mathcal{L} = -\mu \mathbf{S}^{-1}\mathbf{e} - \lambda_{\mathcal{I}} = \mathbf{0} \qquad \text{where } \mathbf{S} = diag(\mathbf{s}), \quad \mathbf{e} = (1, \ldots 1)^T$$

$$\nabla_{\lambda_{\mathcal{E}}} \mathcal{L} = \mathbf{c}_{\mathcal{E}}(x) = \mathbf{0}$$

$$\nabla_{\lambda_{\mathcal{I}}} \mathcal{L} = \mathbf{c}_{\mathcal{I}}(x) - \mathbf{s} = \mathbf{0}$$

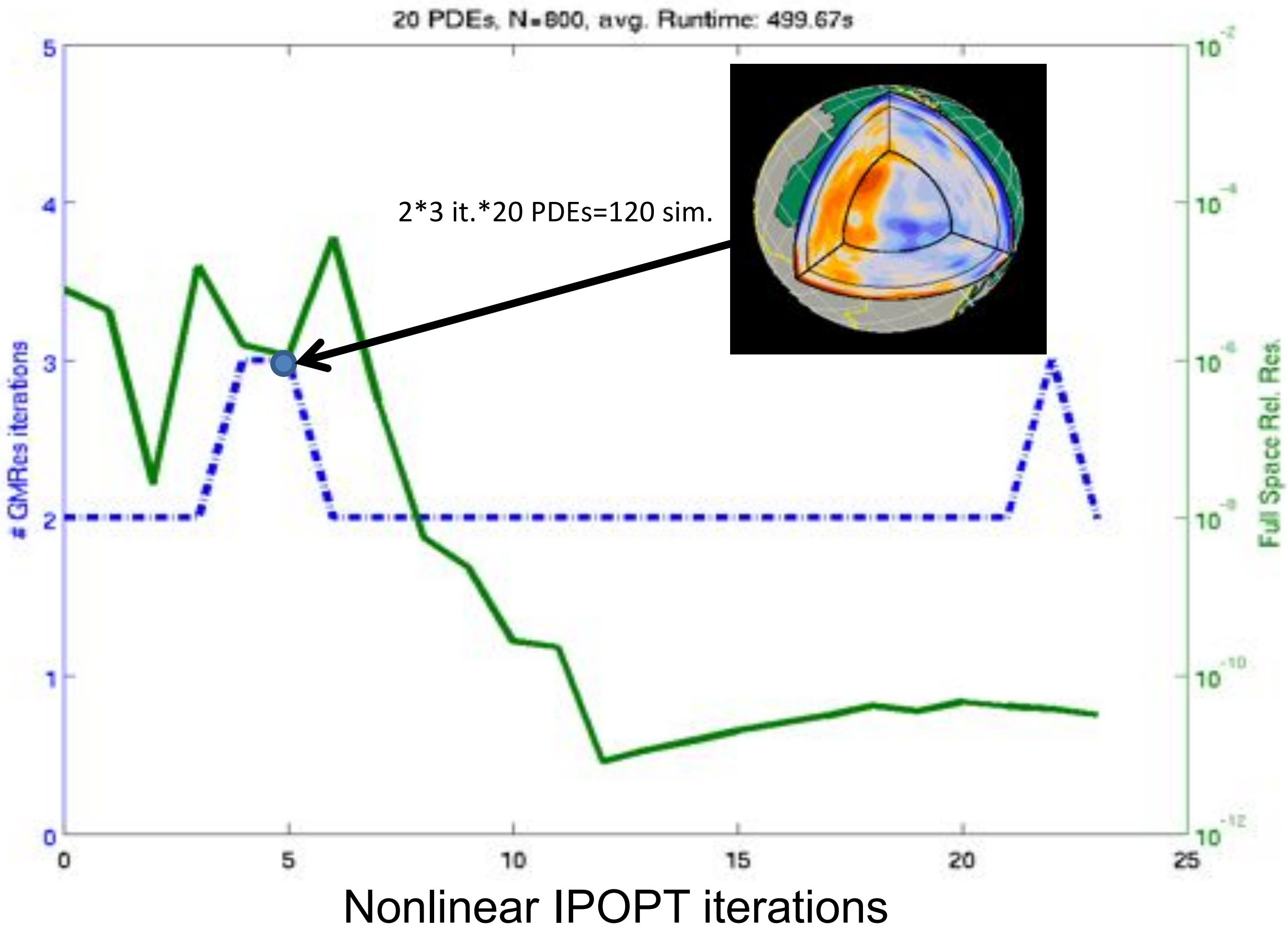
Curtis, S., Wächter, *An Interior-Point Algorithm for Large-Scale Nonlinear Optimization with Inexact Step Computations,* SIAM Sci.Comput,2010.
Curtis, Huber, S., Wächter*, On the Implementation of an IP Algorithm for Nonlinear Optimization with Inexactness*, Math Programming B, 2011

Seismic Inversion



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

Seismic Inversion – Convergence



Curtis, S., Wächter, *An Interior-Point Algorithm for Large-Scale Nonlinear Optimization with Inexact Step Computations,* SIAM Sci.Comput,2010.
Curtis, Huber, S., Wächter*, On the Implementation of an IP Algorithm for Nonlinear Optimization with Inexactness*, Math Programming B, 2012

Summary and Outlook

Summary

- **All stencil kernels are memory bound**, hence applying data locality techniques is crucial.
- **General code generation framework PATUS** for stencil codes of **arbitrary stencil** shapes for **different types of hardware.**

- Application within wave propagation code **AWP-ODC** from SDSC.
- **GPU** implementation of spectral element wave propagation code **SPECFEM** is almost finished and code can be released.

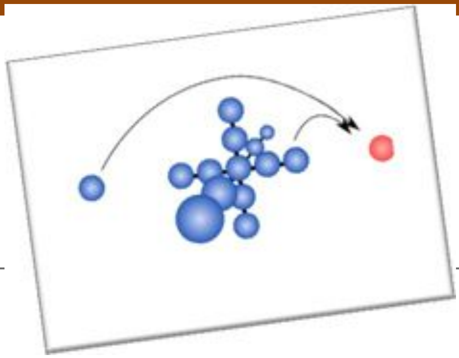
- Fully-parallel **wave propagation solvers** and **advanced inversion algorithms** for complex **3D seismic imaging.**

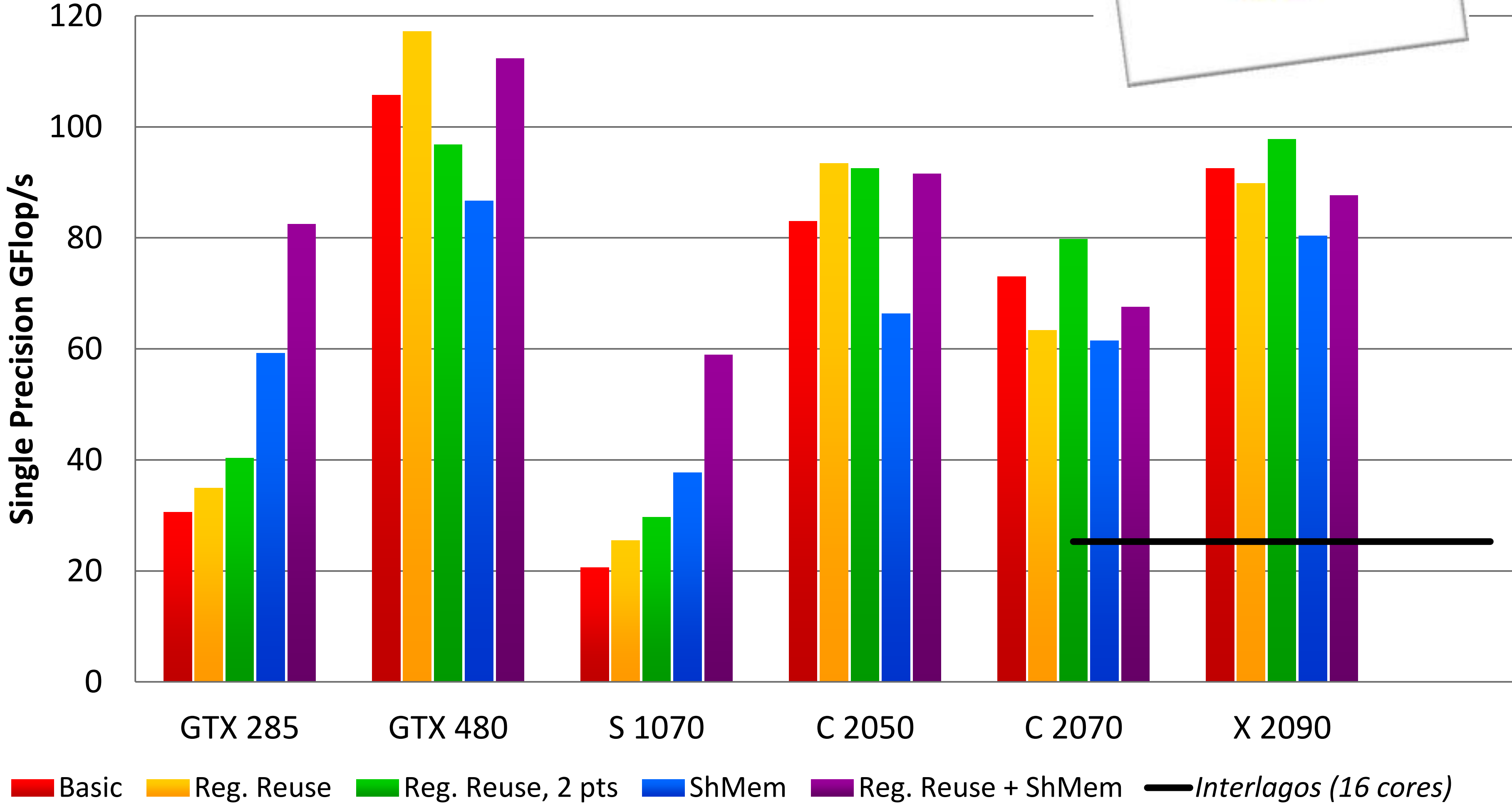
Q & A

Thank you for your attention!

Questions?

Preliminary Results:
Stencil AWP-ODC GPU Code Optimization

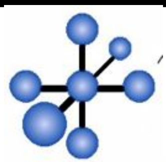




Simulation of an earthquake on the southern San Andreas Fault

Movie courtesy: Dalguer (ETH Zurich), Y. Cui (SDSC) et.al.

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



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