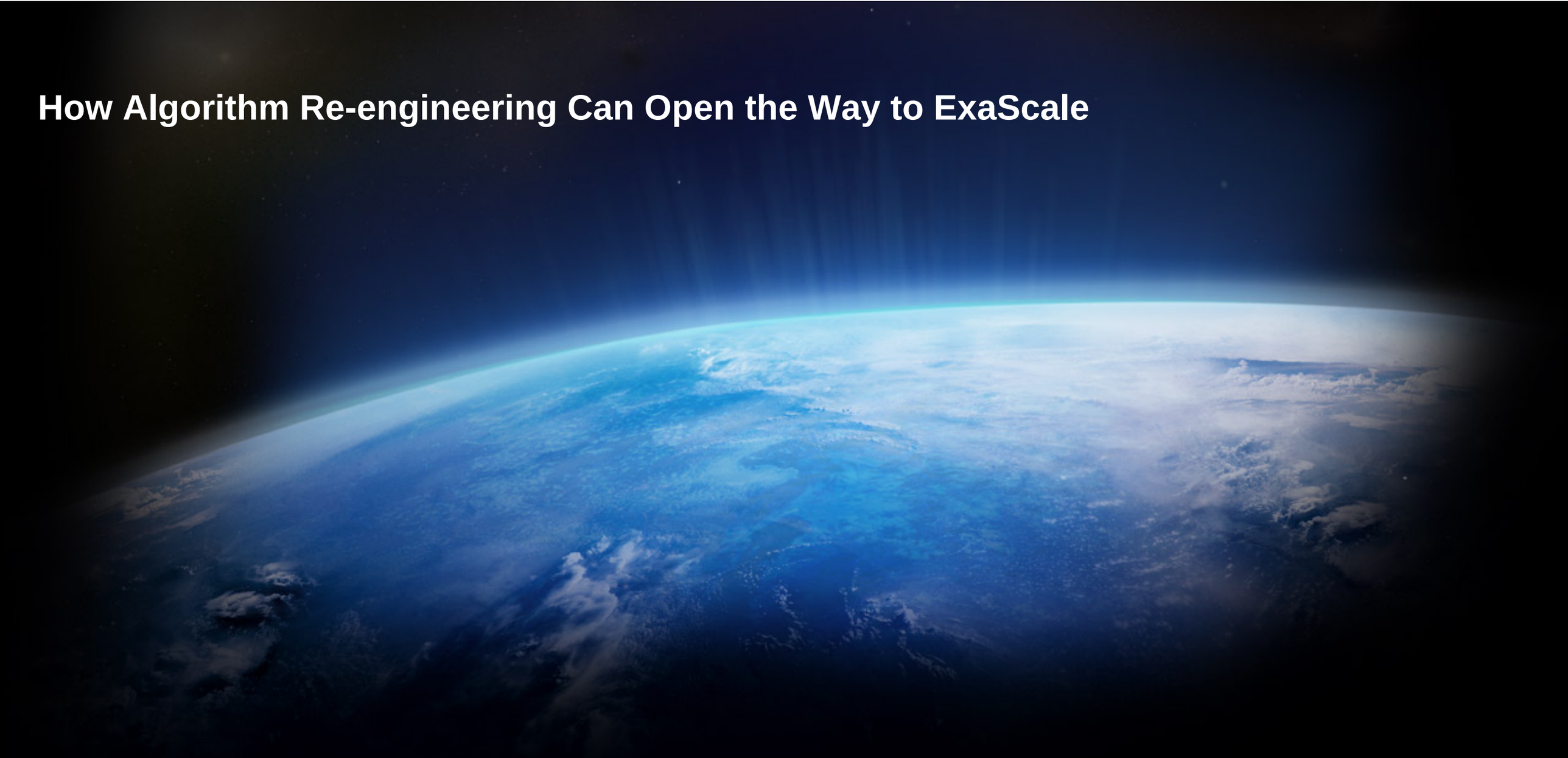
How Algorithm Re-engineering Can Open the Way to ExaScale



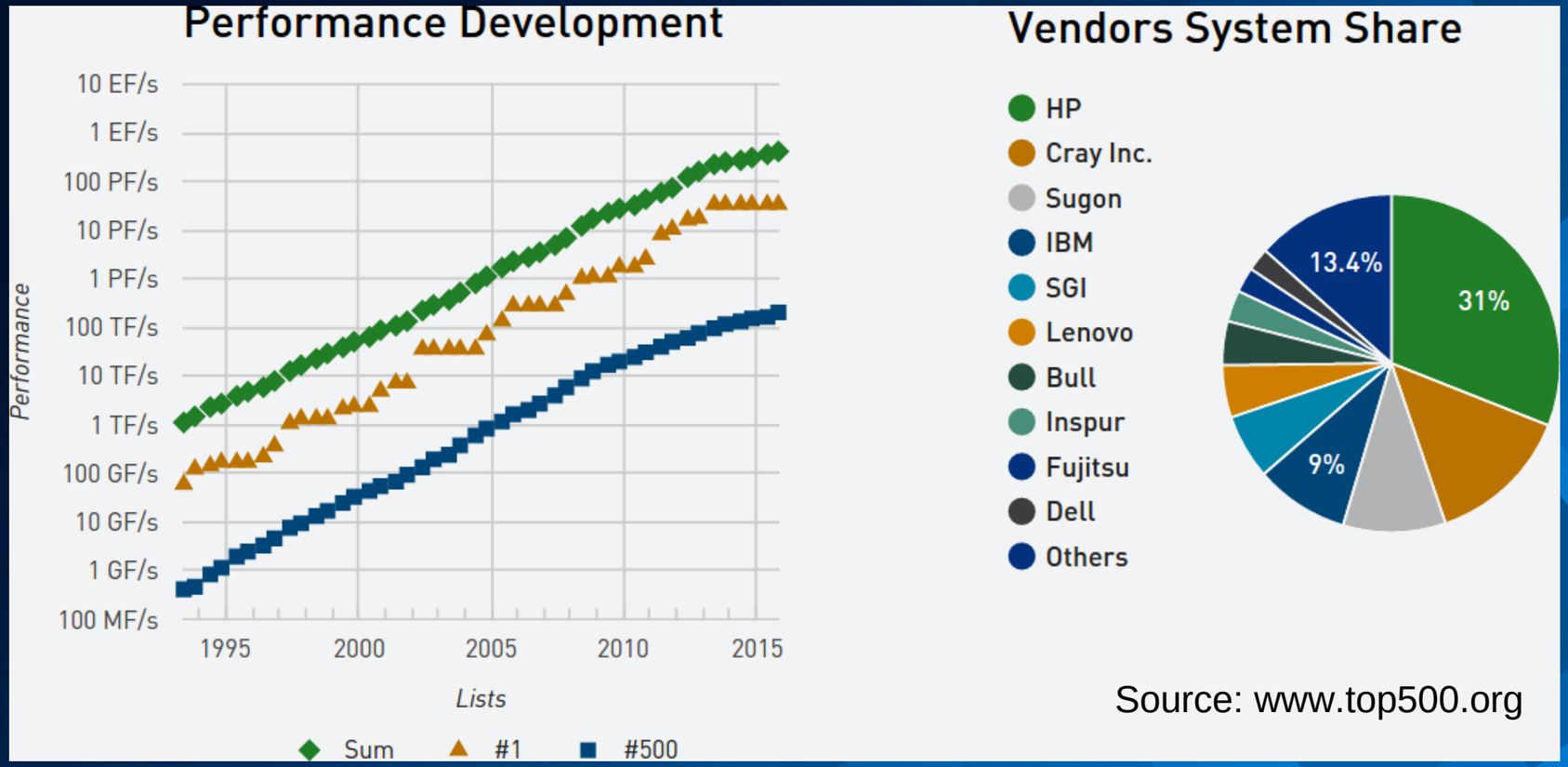
A. Cristiano I. Malossi, Yves Ineichen, Peter W. J. Staar, Costas Bekas
Foundations of Cognitive Solutions
IBM Research - Zurich

IBM

Measuring performance in HPC: TOP500

Initially designed in ~1979 to provide information on execution time to solve a dense linear system

From the '90 considered *de-facto* as the main metric to rank supercomputers



- ✔ Provide a reasonable indication of speed vs. problem size
- ✖ Emphasizes FLOPs and peak performance (do not account for network, bandwidth, energy, etc.)

Measuring performance in HPC: HPCG

Introduced in 2013 to better represent today's real applications (and maybe replace HPL in the future?)

HPCG Rank	Computer	HPCG [PFlops]	Rmax [PFlops]	HPCG/HPL [%]	HPL Rank
1	Tianhe-2	0.58	33.863	1.7	1 (=)
2	K computer	0.4608	10.51	4.4	4 (+2)
3	Titan	0.3223	17.59	1.8	2 (-1)
4	Trinity	0.1826	8.1009	2.3	6 (+2)
5	Mira	0.167	8.587	1.9	5 (=)
6	Hazel Hen	0.138	5.64	2.4	8 (+2)
7	Pleiades	0.1319	4.089	3.2	13 (+6)
8	Piz Daint	0.1246	6.271	2	7 (-1)
9	Shaheen II	0.1139	5.537	2.1	9 (=)
10	Stampede	0.0968	5.168	1.9	10 (=)
11	JUQUEEN	0.0955	5.0089	1.9	11 (=)

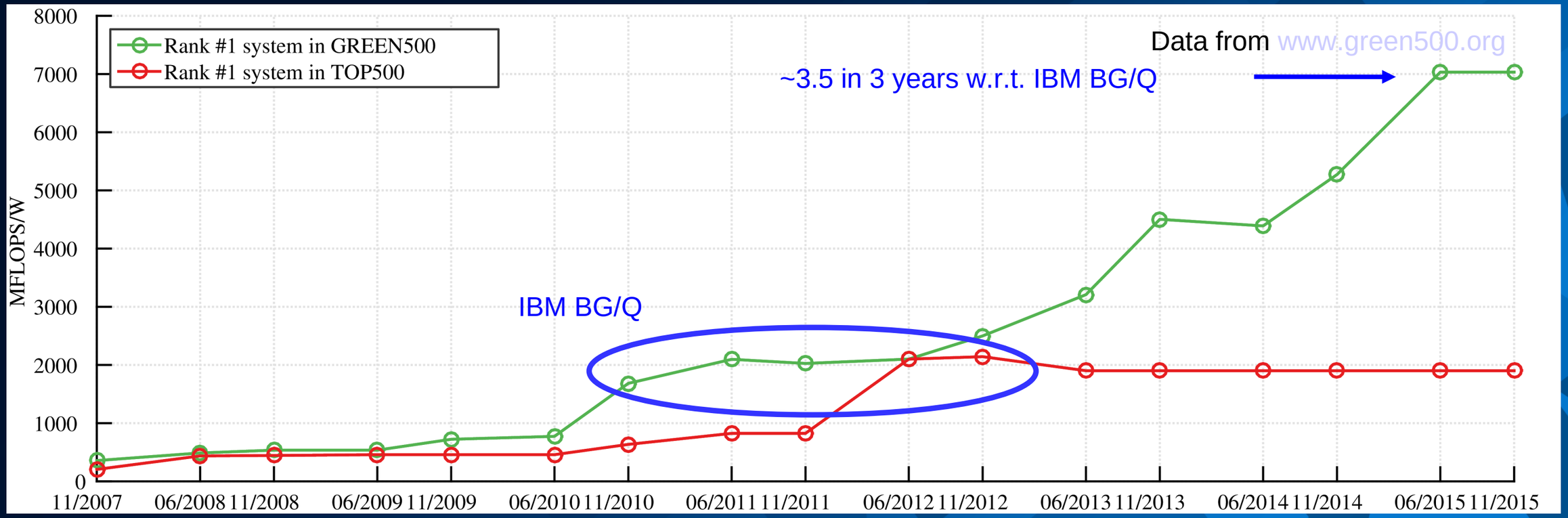
NOTE: Sequoia, 3rd in HPL, would score ~0.33 PFlops in HPCG (projection from Mira and JUQUEEN)

Source: http://www.hpcg-benchmark.org
November 2015

- ✔ Emphasis not only on Flops, but also on memory bandwidth and interconnects
- First positions ranking is similar to HPL, with one outlier
- Specific code optimization impact a lot performances, as well as the Nx-Ny-Nz shape of the domain
- ✖ Measure performances w.r.t. a regular stencil problem: what about more types of sparse matrices?

Measuring performance in HPC: GREEN500 (1)

Introduced in 2007 to complement TOP500 and rank top supercomputers by energy efficiency



- Increased energy awareness

Measuring performance in HPC: GREEN500 (2)

Introduced in 2007 to complement TOP500 and rank top supercomputers by energy efficiency

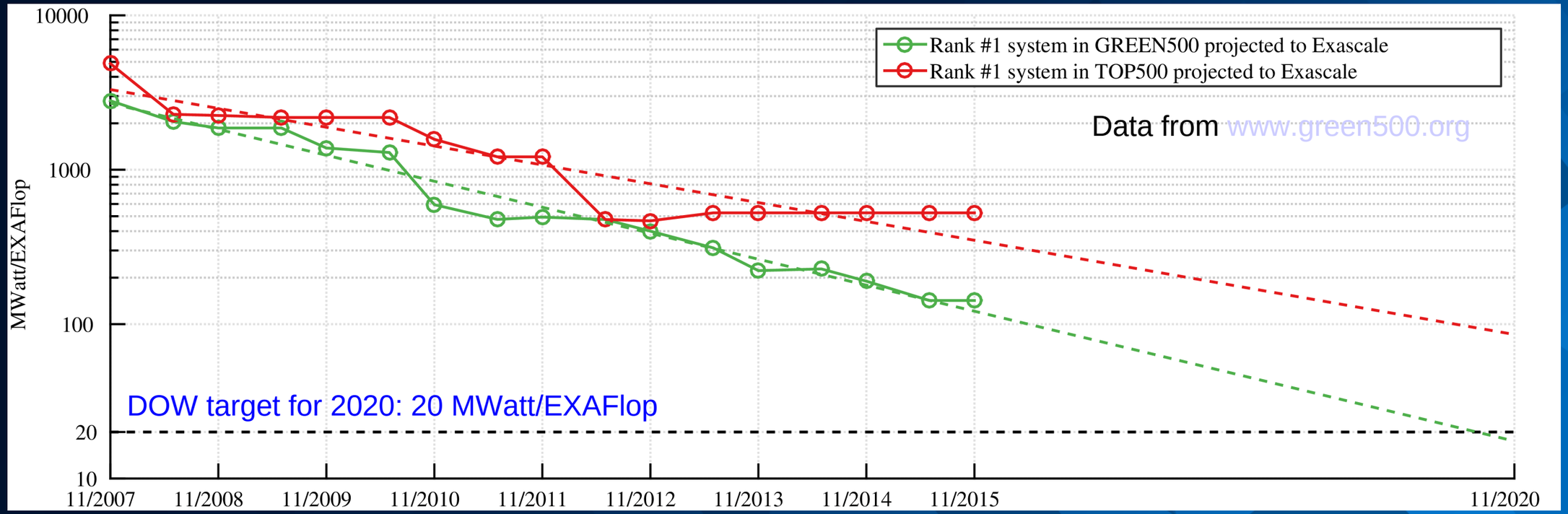
Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	7,031.58	Institute of Physical and Chemical Research (RIKEN)	Shoubu - ExaScaler-1.4 80Brick, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, PEZY-SC	50.32
2	5,331.79	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC/DL - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.1GHz, Infiniband FDR, NVIDIA Tesla K80	51.13
3	5,271.81	GSI Helmholtz Center	ASUS ESC4000 FDR/G2S, Intel Xeon E5-2690v2 10C 3GHz, Infiniband FDR, AMD FirePro S9150	57.15
4	4,778.46	Institute of Modern Physics (IMP), Chinese Academy of Sciences	Sugon Cluster W780I, Xeon E5-2640v3 8C 2.6GHz, Infiniband QDR, NVIDIA Tesla K80	65.00
5	4,112.11	Stanford Research Computing Center	XStream - Cray CS-Storm, Intel Xeon E5-2680v2 10C 2.8GHz, Infiniband FDR, Nvidia K80	190.00
6	3,856.90	IT Company	Inspur TS10000 HPC Server, Xeon E5-2620v3 6C 2.4GHz, 10G Ethernet, NVIDIA Tesla K40	58.00
7	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00
8	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00
9	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00
10	3,775.45	Internet Service	Inspur TS10000 HPC Server, Intel Xeon E5-2620v2 6C 2.1GHz, 10G Ethernet, NVIDIA Tesla K40	110.00

No large system in first 10 positions!

Source: www.green500.org (November 2015)

- ✔ Increased energy awareness
- ✖ Do not promote large supercomputers (do not account for problem size and scalability)
- ✖ Measure energy per Flop and not energy-to-solution

Race to Exascale: are we getting there for 2020?



… in theory yes, but in practice no!

How an ideal HPC metric should be designed to drive hardware development?

- Based on many different real applications (e.g., Colella's Dwarf approach)
- Time-to-solution vs. problem size
- Energy-to-solution vs. problem size

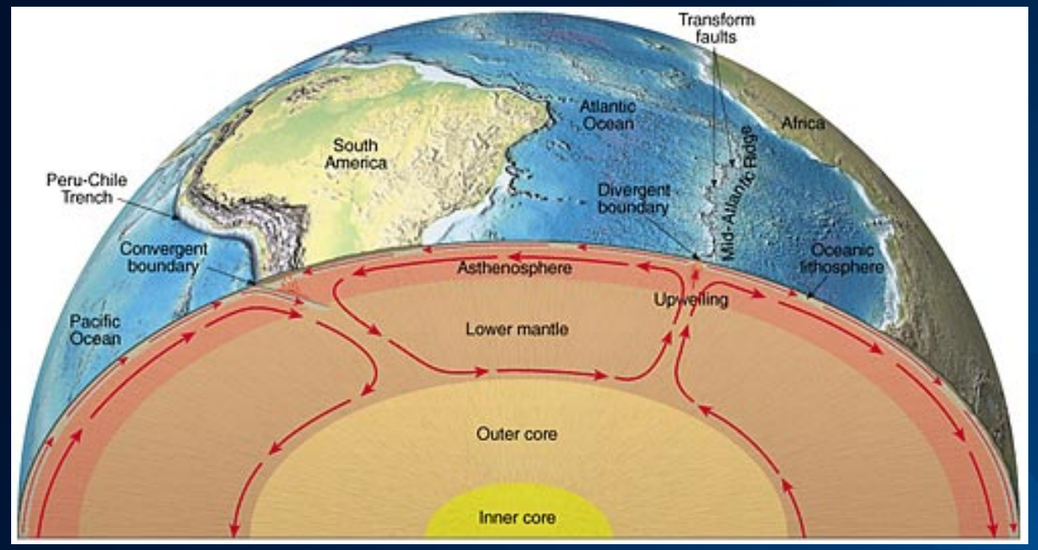
} Scalability (strong and weak)

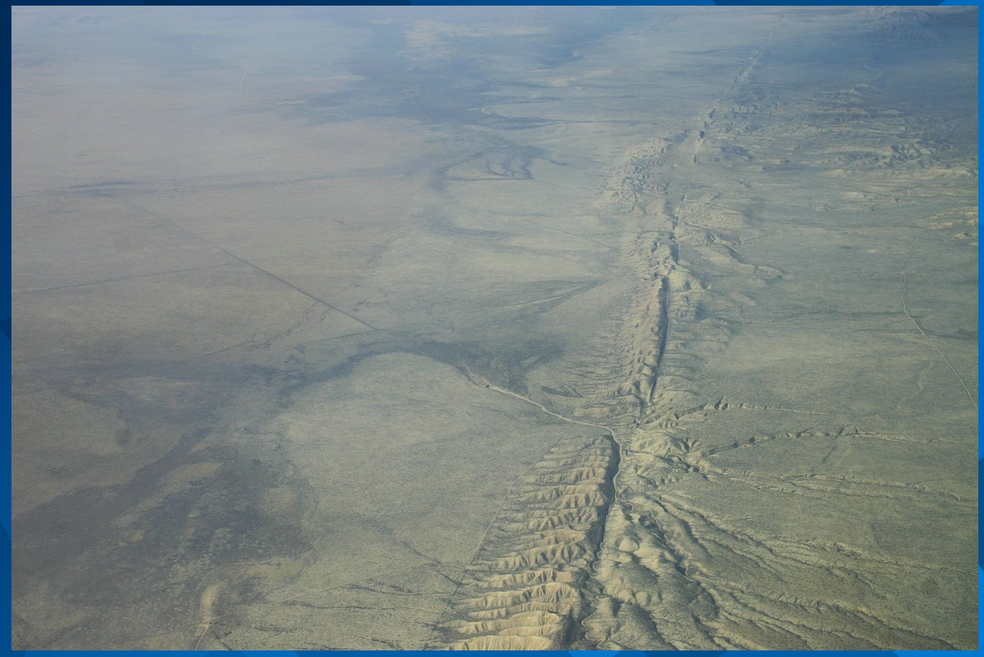
An Extreme-Scale Implicit Solver for Complex PDEs: Highly Heterogeneous Flow in Earth's Mantle (ACM Gordon Bell Winner 2015)

J. Rudi (UT Austin), A.C.I. Malossi (IBM), T. Isaac (UT Austin),
G. Stadler (NYU), M. Gurnis (Caltech), P.W.J. Staar (IBM), Y. Ineichen (IBM),
C. Bekas (IBM), A. Curioni (IBM), and O. Ghattas (UT Austin)

Mantle convection and plate tectonics

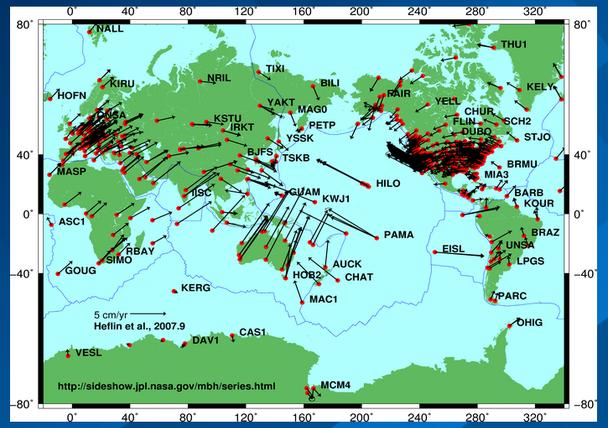
- Mantle convection is the thermal convection in the Earth's upper ~3000 km
- It controls the thermal and geological evolution of the Earth
- Solid rock in the mantle moves like viscous incompressible fluid on time scales of millions of years
- Driver for plate tectonics, earthquakes, volcanos, tsunamis





- Main drivers of plate motion: negative buoyancy forces or convective shear traction?
- Key process governing occurrence of great earthquakes: material properties between the plates or tectonic stress?

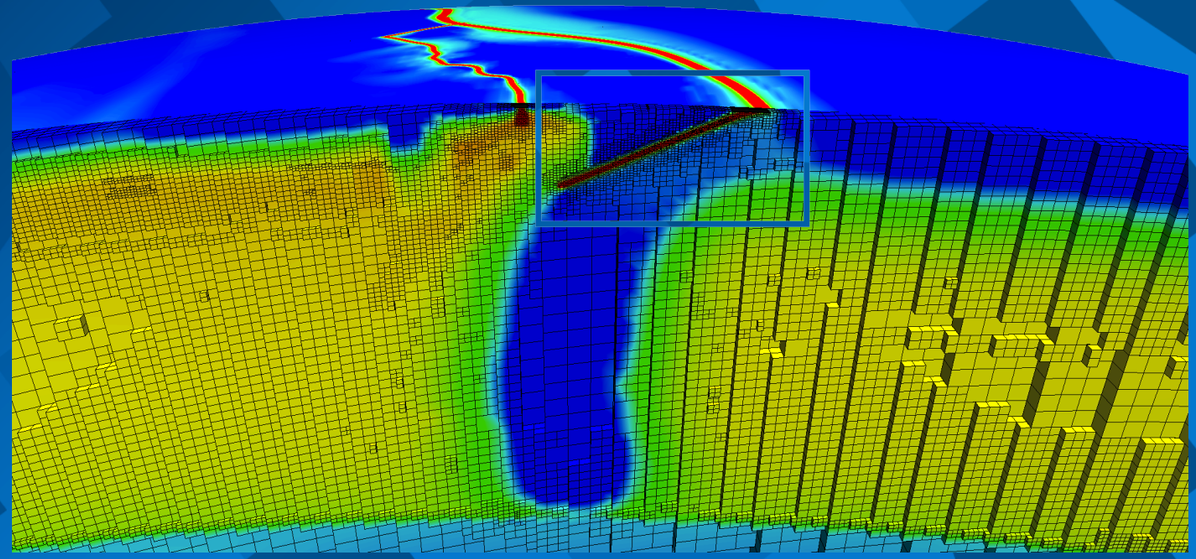
What we know (the data)



- Accurate present-day plate motion (from GPS)
- Topography (indicates traction normal to Earth's surface)
- State of stress between plates and for slabs/subducted plates (from earthquakes)
- Historic plate motion for last few 100M yrs (from magnetic orientation in rocks/plants/animals)
- Rock rheology extrapolated from laboratory experiments (very different temperature/pressure/time scales)
- Images of present-day Earth structure (by correlating seismic wave speed with temperature)

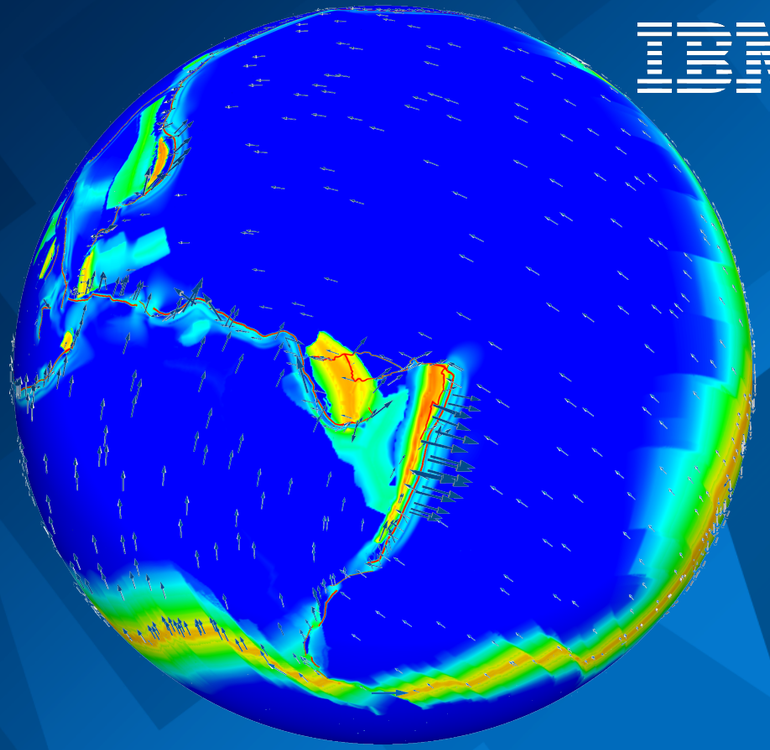
What we would like to learn/infer (from data+models)

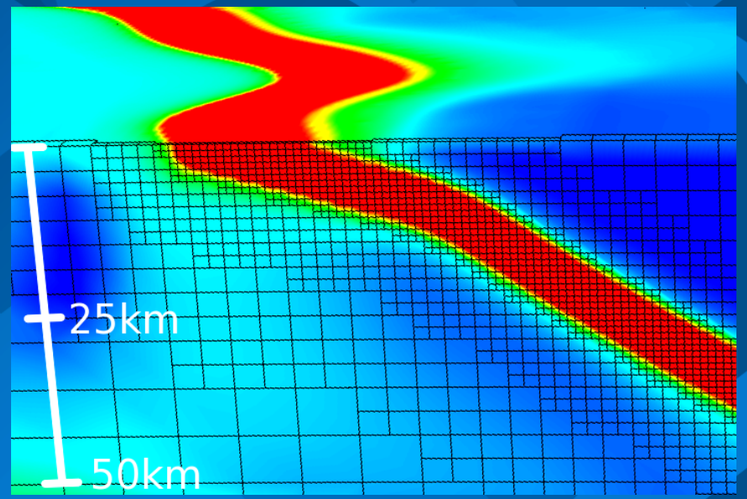
- Main drivers of plate motion: negative buoyancy forces or convective shear traction
- Energy dissipation in plate bending zones; strength of plate coupling
- Earth structure and history
- Role of slab (=subducted plate) geometries



Severe challenges for parallel scalable solvers

- Severe nonlinearity, heterogeneity & anisotropy of the Earth's rheology
- 6 orders of magnitude viscosity contrast; sharp viscosity gradients at plate boundaries
- Wide range of spatial scales and highly localized features w.r.t. Earth radius (~6371 km): plate thickness ~50 km & shearing zones at plate boundaries <5 km
- Desired resolution of ~1 km results in $O(10^{12})$ unknowns on a uniform mesh of Earth's mantle, so adaptive mesh refinement is essential
- High-order discretization essential for maximizing accuracy per memop
- Locally mass-conserving discretization essential for preserving physically meaningful flowfields; achieved via discontinuous pressure approximation
- State of art in extreme-scale implicit solvers: most or all of: linear, constant coefficient, scalar, low order, uniformly-refined meshes, <500K cores





Inexact Newton-Krylov nonlinear solver

$$-\nabla \cdot \left[\mu(T, \boldsymbol{u})\,(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)\right] + \nabla p = \boldsymbol{f}(T)$$

$$\nabla \cdot \boldsymbol{u} = 0$$

$$\dot{\varepsilon}_{\mathrm{II}}(\boldsymbol{u}) := \frac{1}{4}\sqrt{(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top) : (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)}$$

$\boldsymbol{u}$ . . . velocity

p . . . pressure

T . . .temperature

μ . . . viscosity

Rheology is shear-thinning with plastic yielding, and upper/lower viscosity bounds; exponential w.r.t. temperature:

$$\mu(T, \boldsymbol{u}) = \mu_{\min} + \min\left(\frac{\tau_{\mathrm{yield}}}{2\dot{\varepsilon}_{\mathrm{II}}(\boldsymbol{u})}, w \min\left(\mu_{\max}, a(T)\,\dot{\varepsilon}_{\mathrm{II}}(\boldsymbol{u})^{\frac{1-n}{n}}\right)\right)$$

Newton step is computed as (inexact) solution of:

$$-\nabla \cdot \left[\left(\mu\,\mathbf{I} + \dot{\varepsilon}_{\mathrm{II}}\frac{\partial \mu}{\partial \dot{\varepsilon}_{\mathrm{II}}}\frac{(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top) \otimes (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)}{\|(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top)\|_F^2}\right)(\nabla \tilde{\boldsymbol{u}} + \nabla \tilde{\boldsymbol{u}}^\top)\right] + \nabla \tilde{p} = -\boldsymbol{r}_1$$

$$\nabla \cdot \tilde{\boldsymbol{u}} = -r_2$$

- Full Newton Jacobian via regularized plastic yielding and viscosity bounds
- 4th order anisotropic tensor viscosity arises in differentiating strain-rate-dependent viscosity
- Inexact Newton–Krylov via Eisenstat–Walker termination
- Globalization via grid continuation and H^{-1} norm-based line search

Linear solver: preconditioned Krylov subspace method

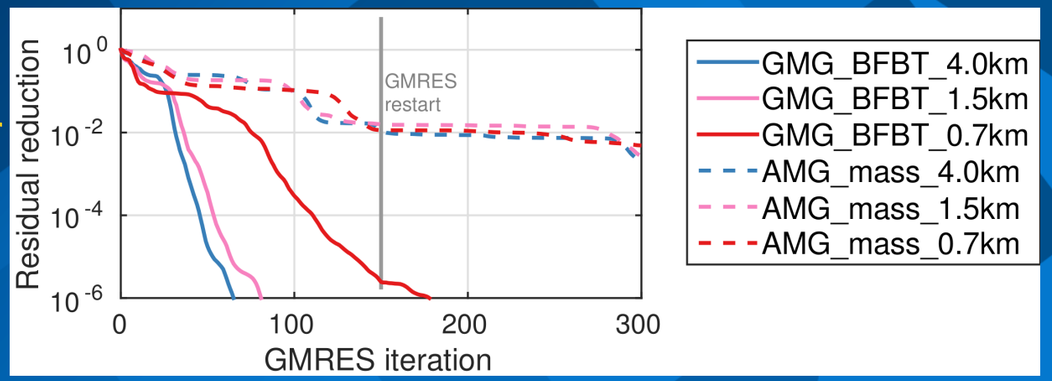
- GMRES with upper triangular block preconditioning:

$$\underbrace{\begin{bmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{0} \end{bmatrix}}_{\text{Stokes operator}} \underbrace{\begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{B}^\top \\ \mathbf{0} & -\tilde{\mathbf{S}} \end{bmatrix}^{-1}}_{\text{preconditioner}} \begin{bmatrix} \mathbf{u}' \\ \mathbf{p}' \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix}$$

- Approximate viscous block inverse $\tilde{\mathbf{A}}^{-1} \approx \mathbf{A}^{-1}$ via multigrid V-cycle
- Inverse Schur complement approximation, $\tilde{\mathbf{S}}^{-1} \approx \mathbf{S}^{-1} := (\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\top)^{-1}$
  - Inverse viscosity-weighted mass matrix typically used to approximate Schur complement, i.e., $\tilde{\mathbf{S}} := \frac{1}{\mu}\mathbf{M}$
  - Spectrally equivalent to Schur complement (Wathen/Silvester/Elman for constant viscosity, Olshanskii et al. for varying viscosity)
  - We use improved version of BFBT/Least Squares Commutator

$$\tilde{\mathbf{S}}^{-1} = (\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\top)^{-1}(\mathbf{B}\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\mathbf{B}^\top)(\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\top)^{-1}$$

with optimally-chosen diagonal scaling $\mathbf{D} := \mathrm{diag}(\mathbf{A})$

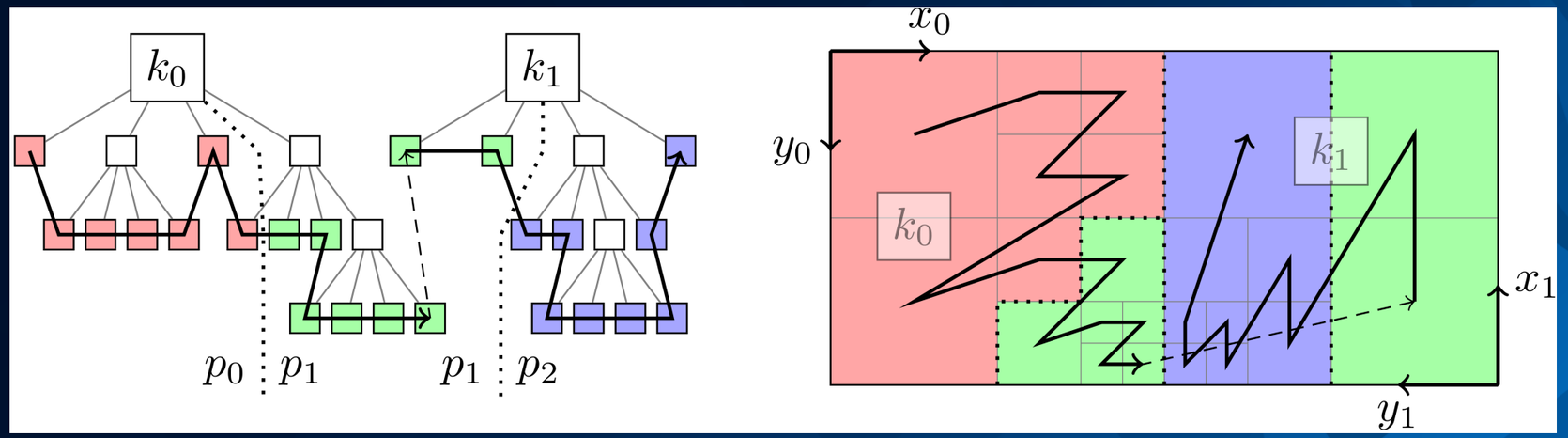


  - $(\mathbf{B}\mathbf{D}^{-1}\mathbf{B}^\top)$ too expensive and memory intensive to construct; approximate it by $-\nabla \cdot \boldsymbol{D}(\boldsymbol{x})\nabla$, i.e. continuous anisotropic Poisson operator with heterogeneous tensor coefficient $\boldsymbol{D}(\boldsymbol{x})$

Discretization and adaptive mesh refinement

$$\begin{bmatrix} \mathrm{A} & \mathrm{B}^\top \\ \mathrm{B} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathrm{u}} \\ \tilde{\mathrm{p}} \end{bmatrix} = \begin{bmatrix} \mathrm{f} \\ 0 \end{bmatrix}$$

- High-order finite element approximation
- Adaptively-refined h-nonconforming forest-of-octree-based meshes
- Scalable fast parallel mesh refinement/coarsening, 2:1 mesh balancing, and repartitioning via p4est library



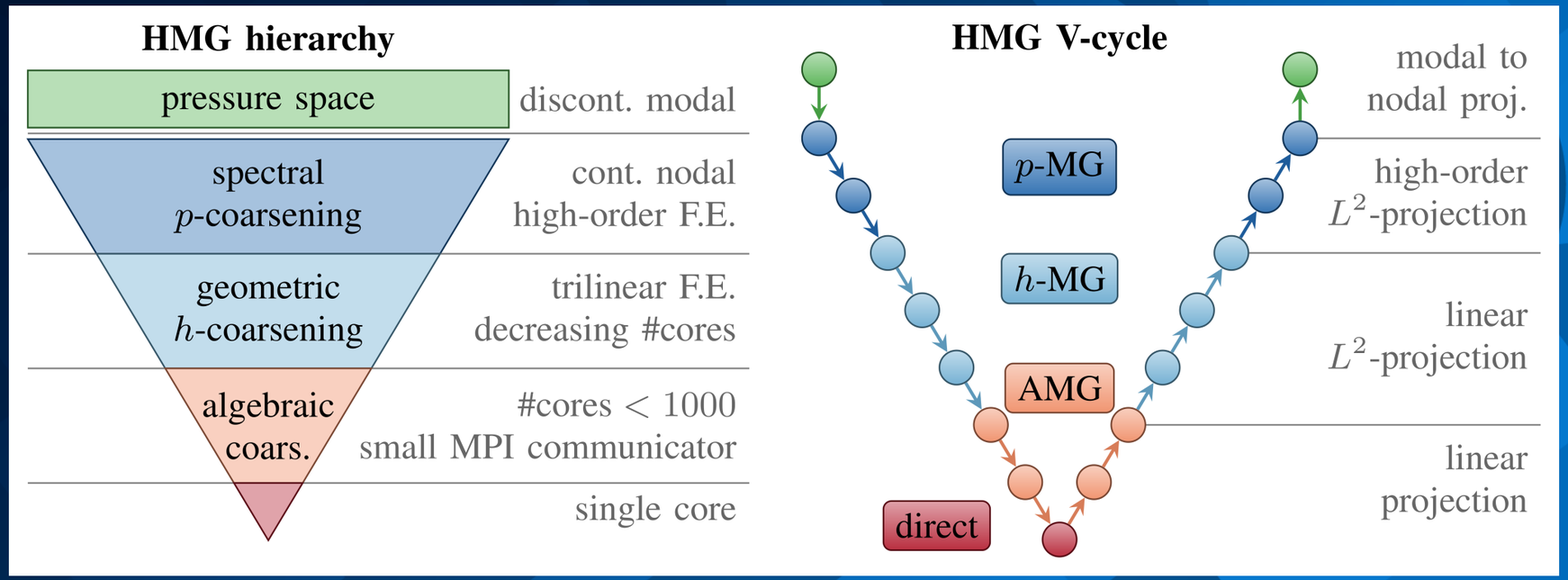
More details in:

C. Burstedde, L. C. Wilcox, O. Ghattas. p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees. SIAM J. Sci. Comp. 33(3), 1103-1133, 2011.

- Linear constraints on discretization at nonconforming interfaces to induce conforming FE approximation
- Inf-sup stable velocity-pressure pairings
- Locally mass conservative via discontinuous pressure space
- Fast, tensorized matrix-free application of finite element matrices

Parallel adaptive high-order spectral-geometric multigrid

- The multigrid hierarchy of nested meshes is generated from an adaptively refined octree-based mesh via spectral-geometric coarsening
- Parallel repartitioning of coarser meshes for load-balancing (sufficiently coarse meshes on subsets of cores)
- High-order L^2-projection of fields on coarser levels (restriction/interpolation are adjoints of each other in L^2-sense)
- Re-discretization of PDEs at coarser geometric multigrid levels
- Chebyshev accelerated Jacobi smoother (PETSc) with tensorized matrix-free high-order stiffness apply
- Coarse grid solver: AMG (PETSc's GAMG), invoked on small core counts



Implementation optimizations for BG/Q (200x speedup)

A) Before optimizations

B) Reduction of blocking MPI communication

C) Minimization of integer operations & cache misses

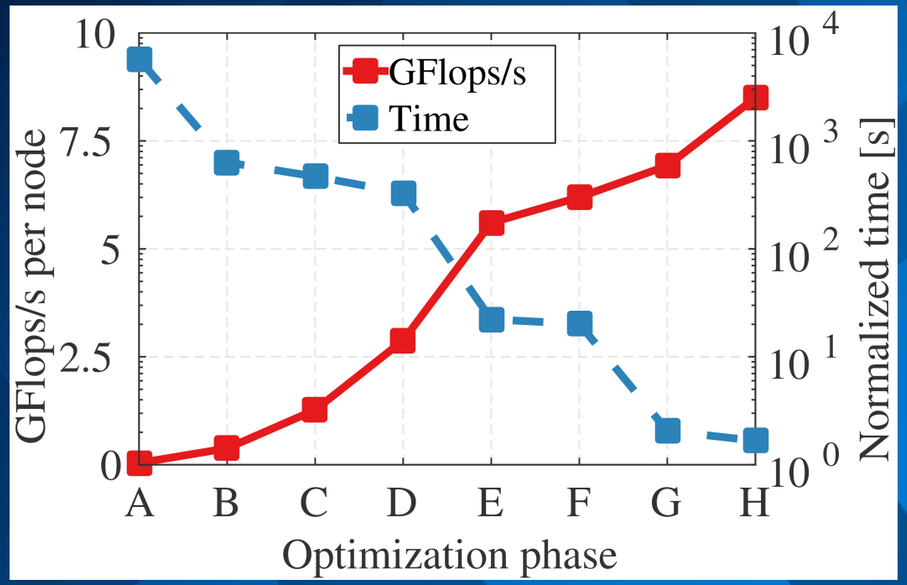
D) Optimization of element-local derivatives; SIMD vectorization

E) OpenMP threading of matrix-free apply loops (e.g. multigrid smoothing, intergrid projection)

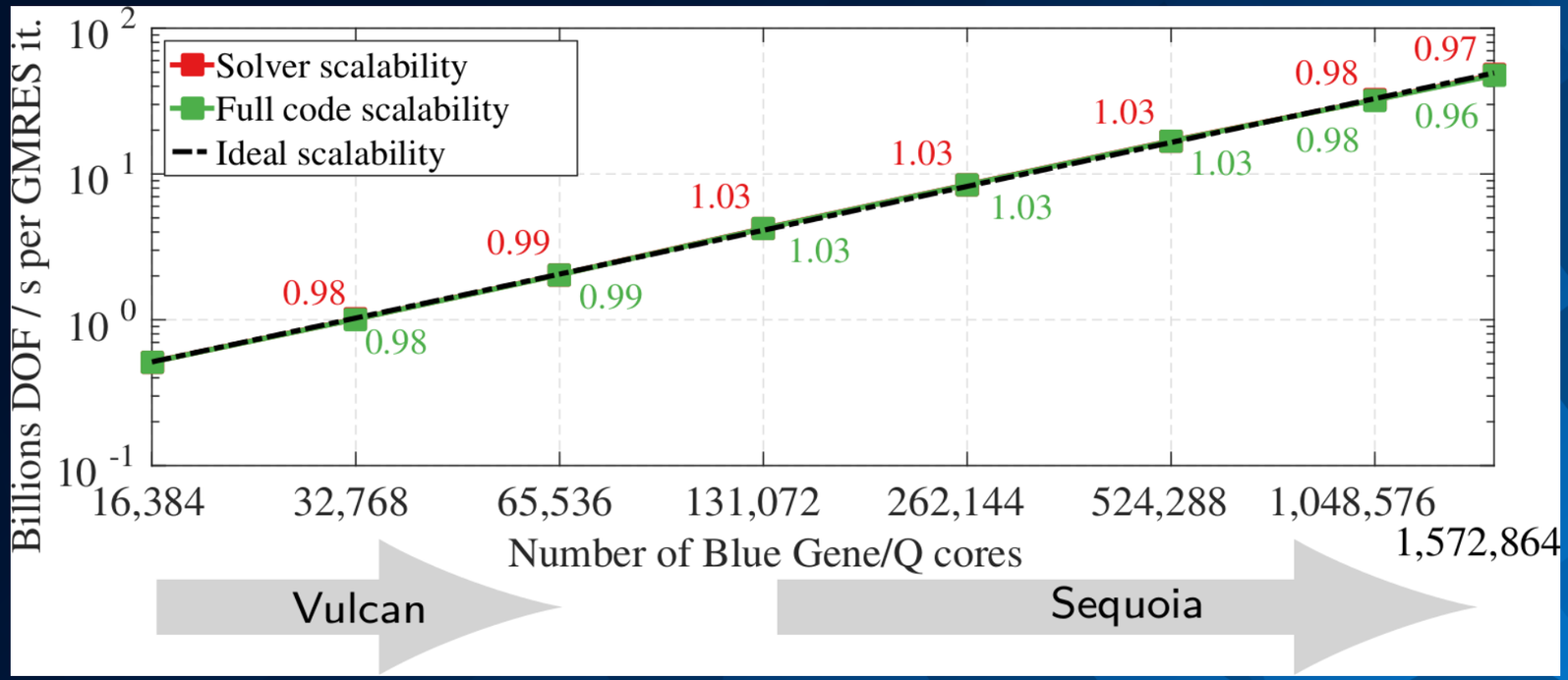
F) MPI communication reduction, overlapping with computations, OpenMP threading in intergrid operators

G) Finite element kernel optimizations (e.g. increase of flop-byte ratio, consecutive memory access, pipelining)

H) Low-level optimizations (e.g. boundary condition enforcement, interpolation of hanging finite element nodes)

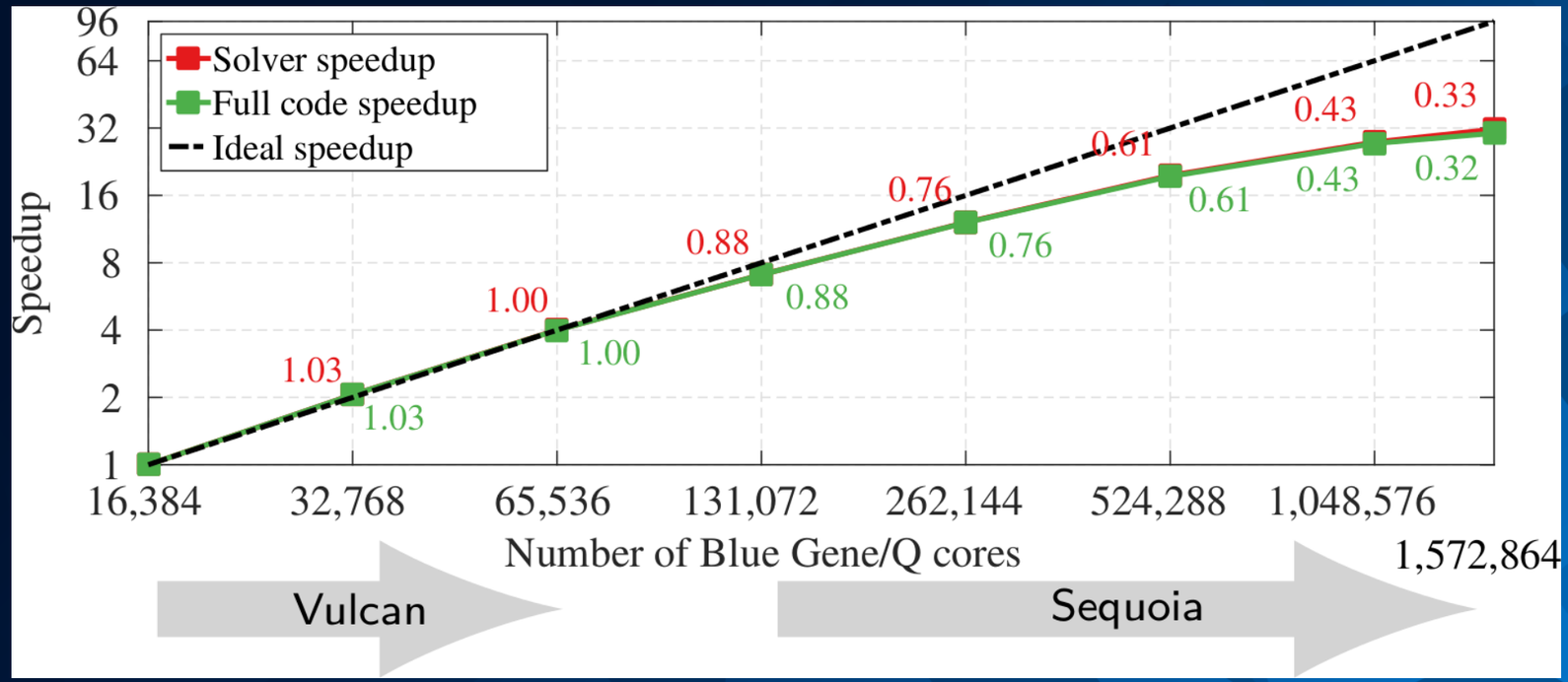


Weak scalability on Sequoia BG/Q: solver and full code



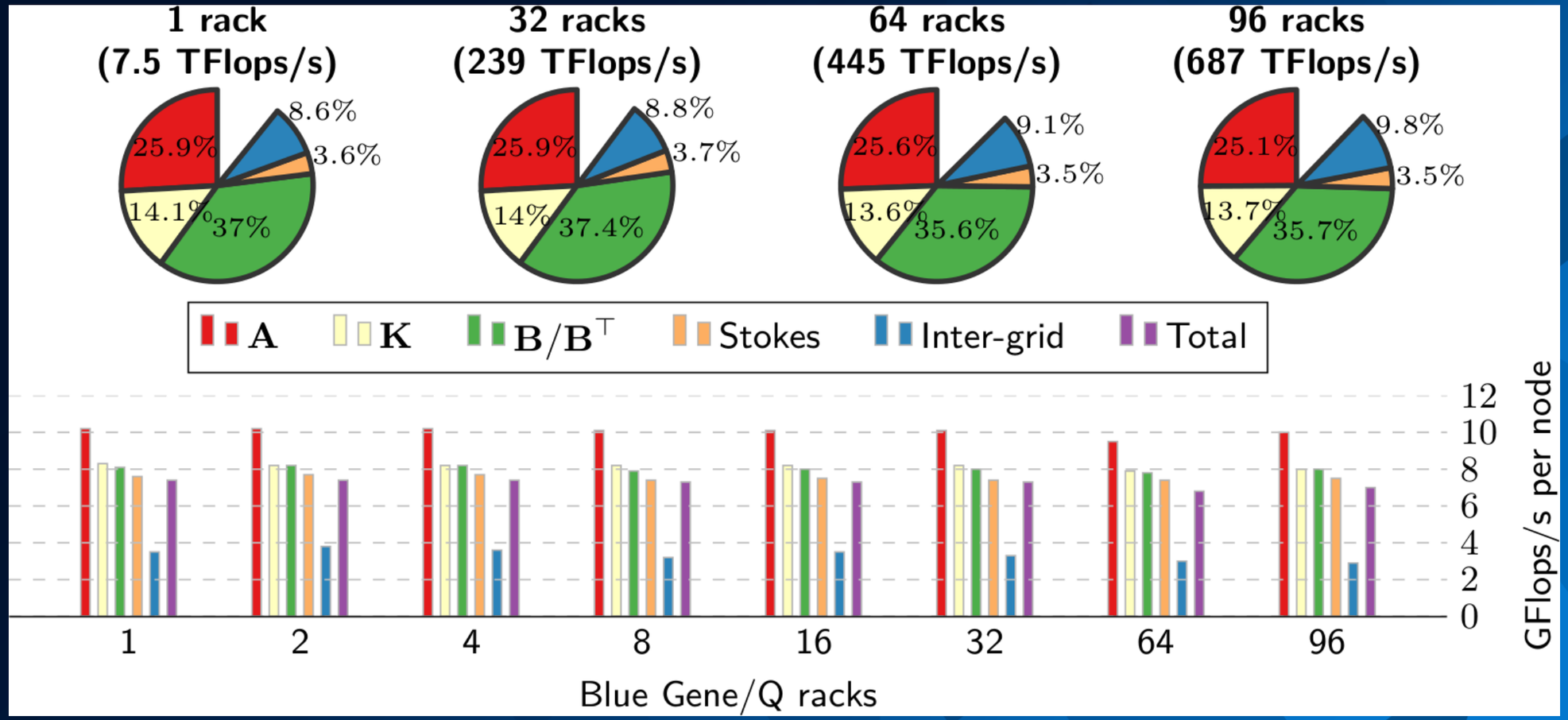
- Performance normalized by time and number of GMRES iterations
- Numbers indicate parallel efficiency w.r.t. ideal speedup (baseline = 1 rack)
  - Red indicates linear solver (10 iterations) only
  - Green indicates projected total runtime (includes measured I/O and setup time and estimate of total solver iterations to convergence)
- Largest problem size has 602 billion DOF on 96 racks

Strong scalability on Sequoia BG/Q: solver and full code



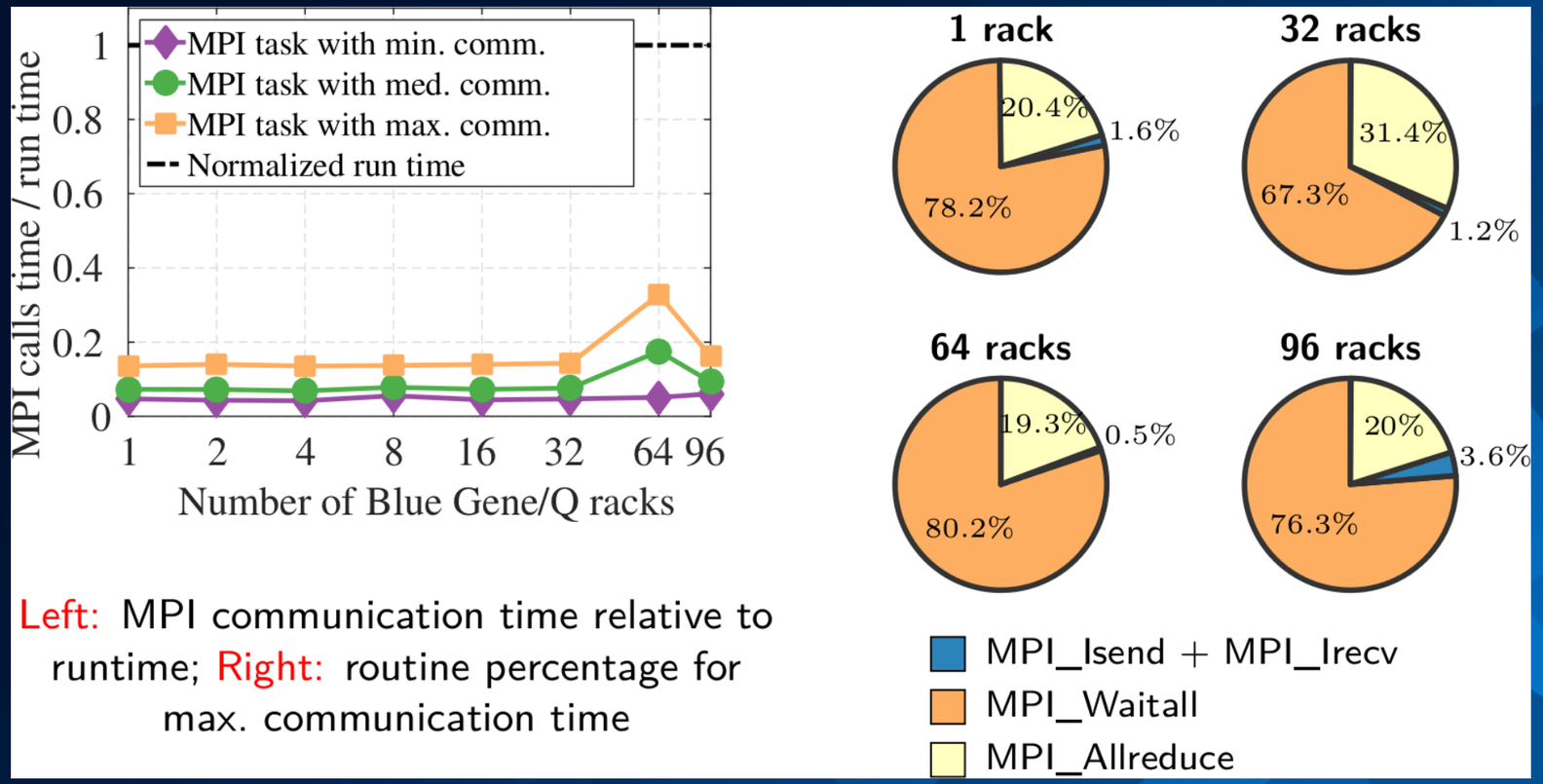
- Performance normalized by time and number of GMRES iterations
- Numbers indicate parallel efficiency w.r.t. ideal speedup (baseline = 1 rack)
  - Red indicates linear solver (10 iterations) only
  - Green indicates projected total runtime (includes measured I/O and setup time and estimate of total solver iterations to convergence)
- Problem size fixed at 8.3 billion DOF

Node performance on Sequoia BG/Q: weak scalability



- MatVecs and intergrid operators within Stokes solves
- Highly optimized matrix-free MatVecs dominate with ~80% of time
- MatVecs and intergrid times consistent across 1 to 96 racks

MPI communication on Sequoia BG/Q: weak scalability



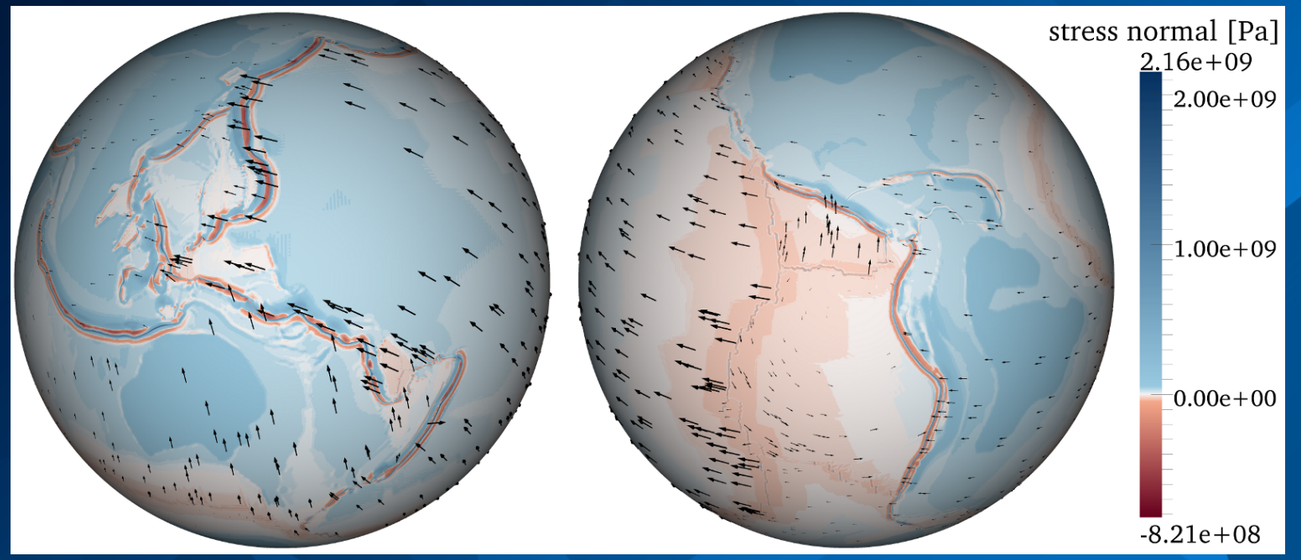
Left: MPI communication time relative to runtime; Right: routine percentage for max. communication time

- Percentage of time spent in MPI communication remains nearly constant
- 64 rack aberration due to lack of 5D torus connectivity in particular configuration
- Max. communication time occur in finer GMG levels (expected for multigrid)
- Min. communication time is more important and always below 3%
- Send/Receive well hidden behind computation

Implications for mantle flow modeling

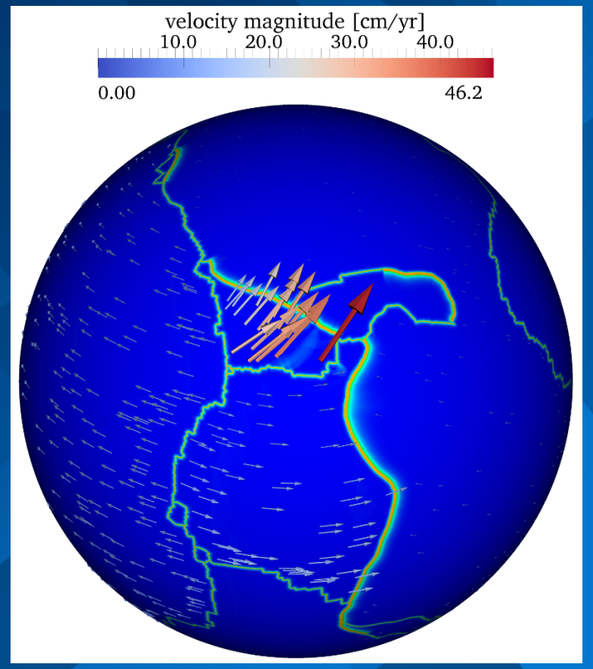
Total normal stress:

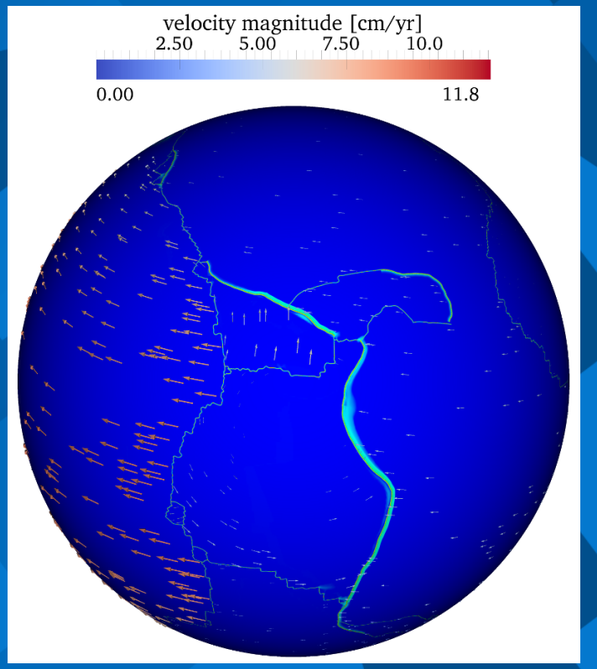
- North-westward motion of Pacific Plate (arrows) and total normal stress field at surface (color coded)
- Forward prediction of width (~50 km) and depth (~10 km) of oceanic trenches along plate boundaries on global scale while predicting plate motions
- First time in a global scale mantle model with plate boundaries



Sensitivity to plate boundary thickness & mesh:

- Comparison of plate velocities: low-fidelity model (left) and high-fidelity model (right) with thinner plate boundaries and finer mesh resolution
- Significant sensitivity of plate velocities of the Cocos Plate (in center) is observed





Solving Dense Symmetric Positive Definite (SPD) Linear Systems

Solving dense SPD linear systems

Consider a generic SPD linear system:

$$Ax = b$$

Typically this is a “no-brainer”: use Cholesky, BLAS3, thus optimal... but is it?

$$A = RR^T$$

R is upper triangular. Then solving Ax = b becomes

$$X = A^{-1}b = (R^TR)^{-1}b = R^{-T}R^{-1}b$$

- Inverting (solving: back substitution) triangular matrices is cheap: $O(n^2)$
- But the Cholesky decomposition costs $O(n^3)$
- With n = 1M, already requires ~Exaflop resources! Can we do better? Can we accelerate?

Dive in the past: Iterative Refinement (IR)

Consider the linear system: $Ax = b$ and assume we have an initial "guess" x_0

1) Compute the residual: $r = b - Ax_0$
2) Solve for the residual: $Ad = r$
3) Update the solution: $x_1 = x_0 + d$

Repeat steps 1-3 if remainder is not small enough: $||r||_2 < tol$

What if steps 1-3 could be done in infinite precision (*no rounding errors*):

- $d = A^{-1}r = A^{-1}(b - Ax_0)$
- $d = x - (A^{-1}A)x_0 = x - x_0$
- $x_1 = x_0 + x - x_0 = x$

Thus, we would have a completely accurate result in 1 step! But, round-off is inevitable. So, why does IR work?

Computing r and d "accurately enough" is adequate to bring improvement to x_1

Why low-precision iterative refinement works?

Theorem:

Low-Precision IR converges so long as the solver we use for a system Ay = c satisfies:

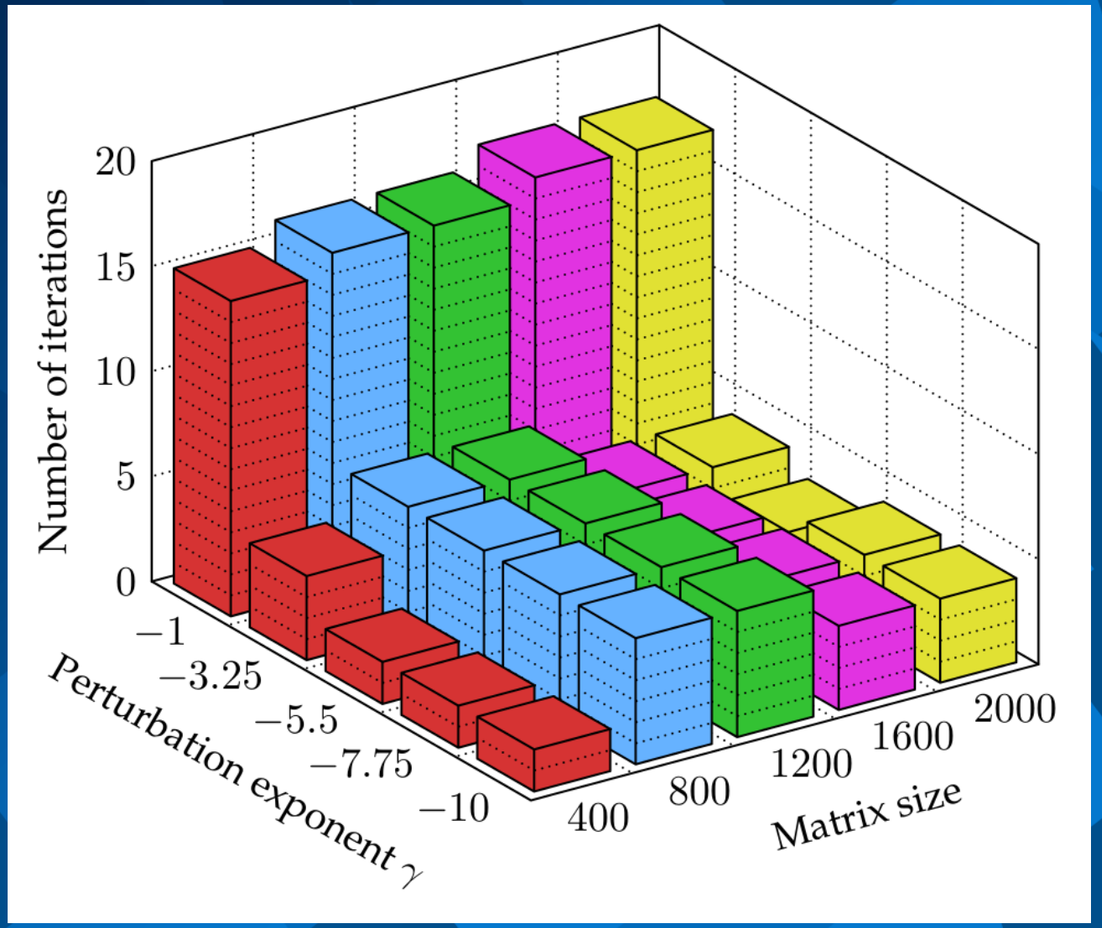
$$(A + E)\, y' = c, \qquad \|A^{-1} E\|_\infty < 1$$

where y' is the computed solution

Key observations:

Can we relax solver accuracy? Yes

Can we use "dirty/noisy" solvers? Yes



Mixed precision iterative refinement with Cholesky

Consider two modes of machine precision:

Low Precision: LP / High Precision: HP

1) Compute the Cholesky factorization:	$A=R^TR$.	Cost: $O(1/3n^3)$	ACCELERATION
2) Compute initial solution:	$R^T(R\ x_0) = b$.	Cost: $O(n^2)$	
3) Compute initial residual:	$r_0 = b - Ax_0$.	Cost: $O(n^2)$	
4) Initialize counter	$k = 0$		
5) Repeat			
(1) Solve for residual:	$R^T(R\ d_k) = r_k$	Cost: $O(n^2)$	
(2) Update solution:	$x_{k+1} = x_k + d_k$	Cost: $O(n)$	
(3) Compute residual:	$r_{k+1} = b - Ax_{k+1}$	Cost: $O(n^2)$	
(4) Check tolerance:	$\|\|r_{k+1}\|\| < tol$		
(5) Update counter	$k = k + 1$		

Key properties:

- Overall cost $O(1/3n^3)$ is performed in low precision. Cost in high precision is $O(n^2)$
- We can take great advantage of fast single precision hardware!
- We benefit from reduced memory traffic (compare 4 bytes of IEEE single precision to 8 bytes for IEEE double p.)

Mixed precision iterative refinement with Conjugate Gradient

The cubic complexity of standard iterative refinement stems from the Cholesky decomposition

We saw that we could utilize a significantly less accurate solver, therefore we can:

- Substitute the dense solver (Cholesky based) with an iterative one (CG for SPD linear systems)
- Perform only a small (constant) number of CG steps, p<<n

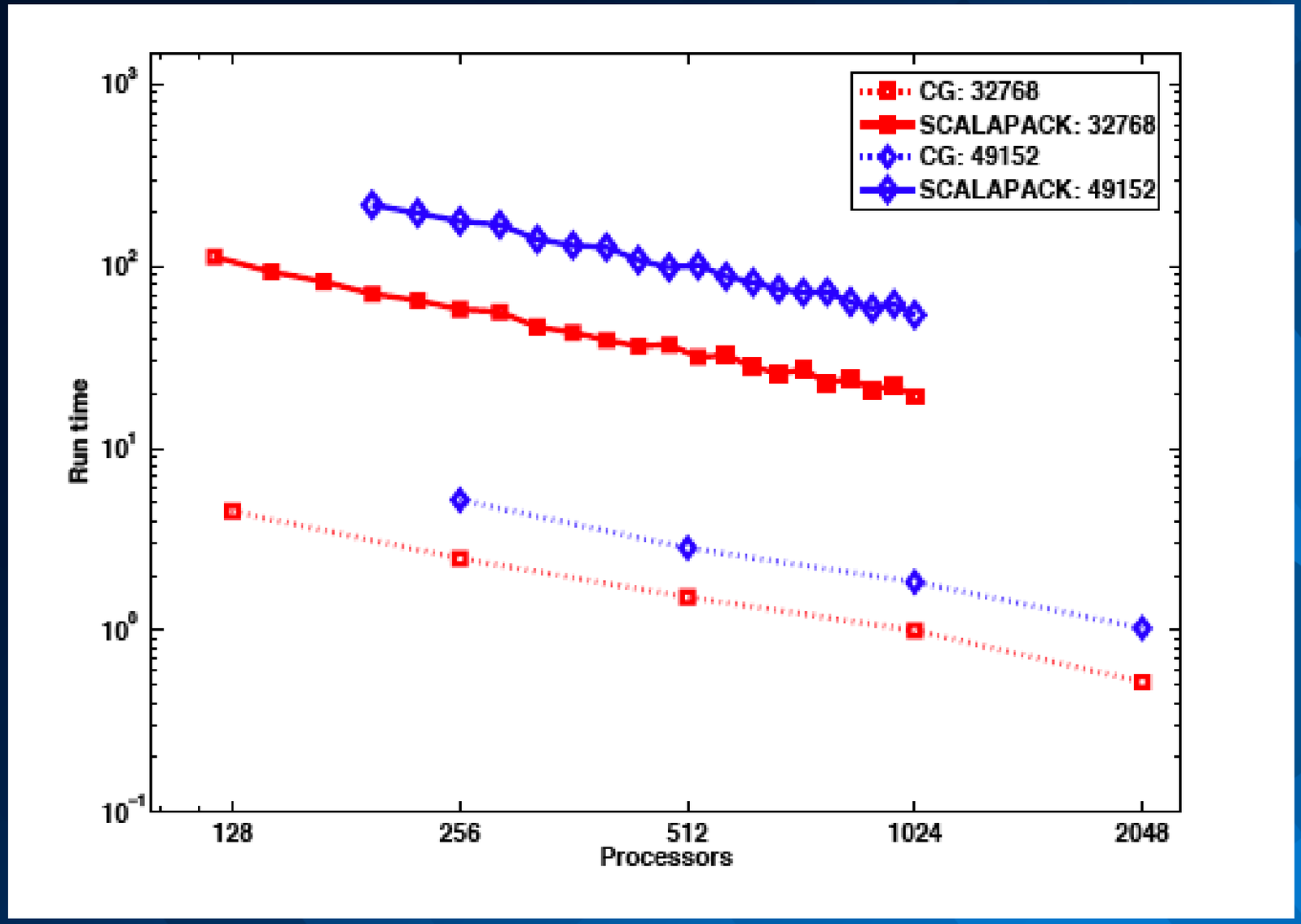
Low Precision: LP / High Precision: HP

1) Compute initial solution:	$x_0 = CG(A, b, p)$	Cost: $O(pn^2)$
2) Compute initial residual:	$r_0 = b - Ax_0$.	Cost: $O(n^2)$
3) Initialize counter	$k = 0$	
4) Repeat		
(1) Solve for residual:	$d_k = CG(A, r_k, p)$	Cost: $O(pn^2)$
(2) Update solution:	$x_{k+1} = x_k + d_k$	Cost: $O(n)$
(3) Compute residual:	$r_{k+1} = b - Ax_{k+1}$	Cost: $O(n^2)$
(4) Check tolerance:	$\|\|r_{k+1}\|\| < tol$	
(5) Update counter	$k = k + 1$	

Key property:

- Cost in low precision reduces from $O(n^3)$ to $O(pkn^2)$. Cost in high precision is $O(kn^2)$

Results scalability: CG IR vs Cholesky IR

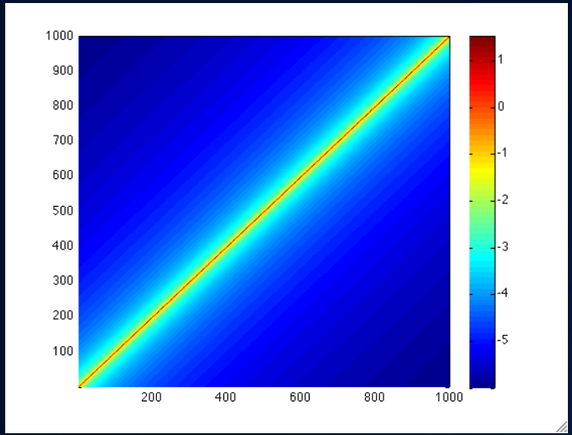


Next step (TODO): test on larger machines, e.g., 1-2 BG/Q racks

Can we push for more?

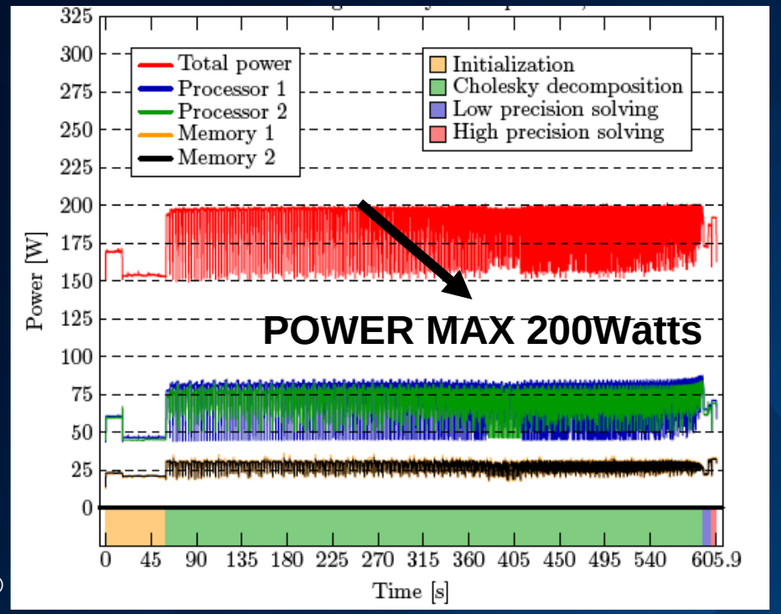
Data Analytics – working with covariance matrices:

Typically they exhibit a decaying behavior away from the main diagonal. What if we make it banded? Converges!

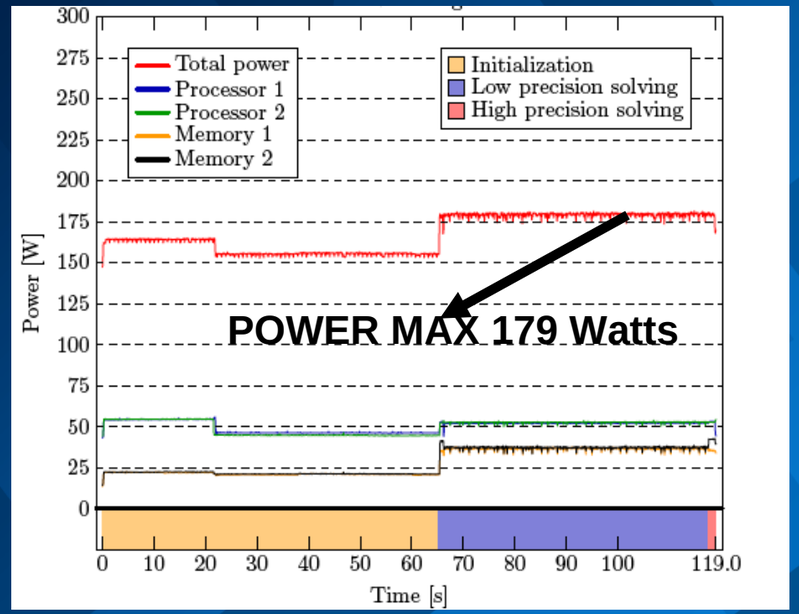


	RHS's	Iterations [low/high]	Time [s]	Average power [W]	Standard error [W]	Energy [kW·s]	TOP500 [GFlops]	Green500 [GFlops/W]
$d = 4$								
Cholesky	32	1 / 1	546.0	190.0	13.5	103.7	214.4	1.11
CG	1	85 / 1	53.8	179.0	1.8	9.6	15.7	0.09
CG	32	88 / 1	125.5	195.0	10.8	24.6	222.2	1.13
Banded CG	1	85 / 1	1.8	174.1	4.9	0.3	5.5	0.03
Banded CG	32	88 / 1	8.4	172.6	14.2	1.5	37.8	0.22

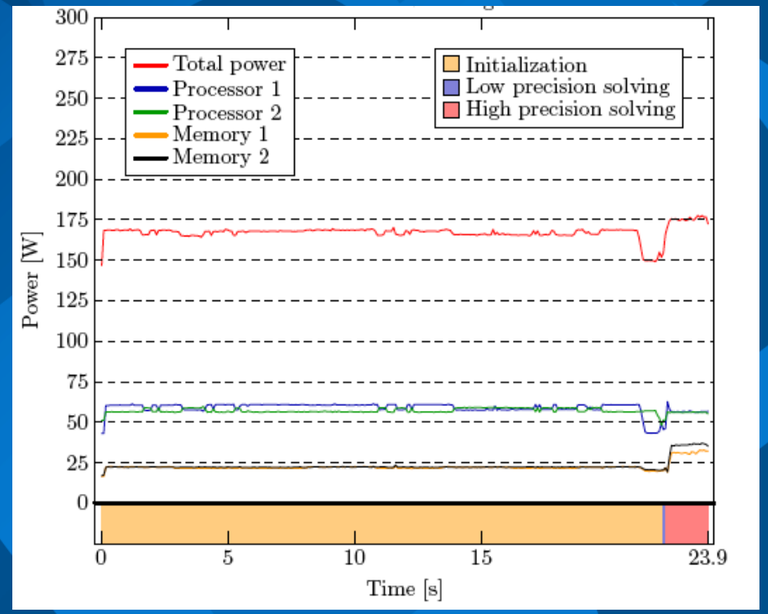
CHOLESKY



CG IT – 1 RHS



BANDED CG IT – 1 RHS



Fast Approximate Math Expressions for Big Data, Deep Learning, and more ...

Impact on Big Data

Main players are investing billions of Dollars in Data Analytics and Deep Learning:

- Knowledge extraction
- Image recognition (face, objects, captions generation, ...)
- Speech recognition (language identification, automatic translation, ...)
- Sentiment analysis (emotions, views, impact of thoughts, …)
- and many more (potentially unlimited) …

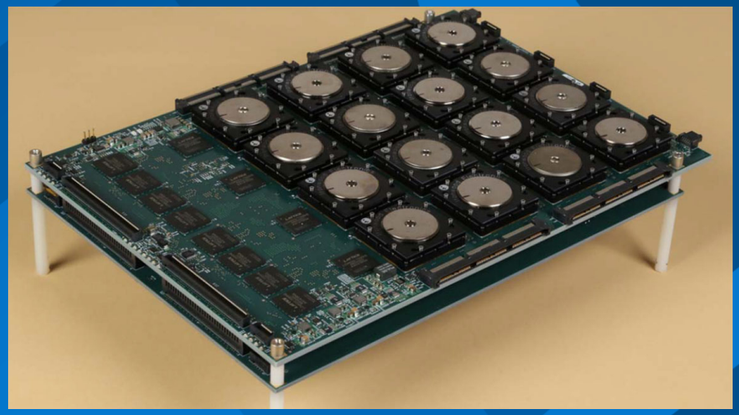
2014 benchmark records with LSTM RNNs, often at major IT companies:

1. Large vocabulary speech recognition (Sak et al., Google, Interspeech 2014)
2. English to French translation (Sutskever et al., Google, NIPS 2014)
3. Text-to-speech synthesis (Fan et al., Microsoft, Interspeech 2014)
4. Prosody contour prediction (Fernandez et al., IBM, Interspeech 2014)
5. Language identification (Gonzalez-Dominguez et al., Google, Intersp. 2014)
6. Medium vocabulary speech recognition (Geiger et al., Interspeech 2014)
7. Audio onset detection (Marchi et al., ICASSP 2014)
8. Social signal classification (Brueckner & Schulter, ICASSP 2014)
9. Arabic handwriting recognition (Bluche et al., DAS 2014)
10. Image caption generation (Vinyals et al., Google, 2014)
11. Video to textual description (Donahue et al., 2014)

A list of benchmark records summarized in a talk of J. Schmidhuber at ETHZ in 2014

Neural Networks

- IBM TrueNorth 2014 is the first neuromorphic chip, specifically made to simulate complex neural networks (NN), with ~268 millions programmable synapses
- Activation functions between synapse are triggered by functions as: exp(), log(), pow(), tanh()
- Accuracy control is a desirable feature while training NN

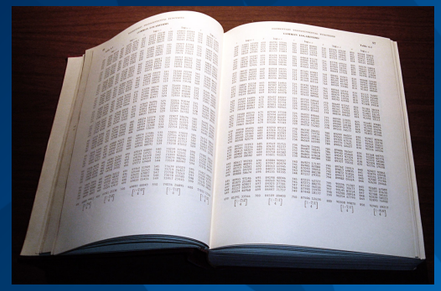


Drawbacks of state of the art practices

- Power series/Taylor expansions: $e^x = \sum_{k=1}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

  PROS: make use of arithmetics (can use SIMD unit)
  PROS: flexible accuracy
  CONS: convergence is very slow (unusable for high accuracy)
  CONS: even using Horner's rule it requires too many floating-point multiply-add

- Look-up tables: $e^x = 2^{x \log_2(e)} = 2^{x_i + x_f}$ →



  PROS: faster than Power series/Taylor expansions
  CONS: do not make use of arithmetics (only partial SIMD use)

- IEEE-745 manipulations, by N. N. Schraudolph in 1998: $(-1)^s(1+m)2^{x-x_0}$

sxxxxxxx	xxxxmmmm	mmmmmmmm	mmmmmmmm	mmmmmmmm	mmmmmmmm	mmmmmmmm	mmmmmmmm
1	2	3	4	5	6	7	8
iiiiiiii	iiiiiiii	iiiiiiii	iiiiiiii	jjjjjjjj	jjjjjjjj	jjjjjjjj	jjjjjjjj

$\texttt{int i} = A \cdot x + B - C$

with $A = S/\ln(2)$, $B = S \cdot 1023$, $C = 60801$, being $S = 2^{20}$

  PROS: extremely fast
  CONS: very inaccurate (max 1 or 2 digits accurate)

Idea: combine IEEE-745 manipulation with polynomial interpolation

Algorithm I **Input:** x and n; **Output:** $f(x) \approx e^x$

1: $x = x \cdot \log_2(e)$
2: $x_f = x - \texttt{floor}(x)$
3: $x = x - \mathcal{K}_n(x_f)$, with $\mathcal{K}_n(x_f) = a \cdot x_f^n + b \cdot x_f^{n-1} + c \cdot x_f^{n-2} + \cdots$
4: Compute the `long int i` as: $2^{52} \cdot x + \mathrm{B}$
5: Read the `long int i` as a `double` and return the value as the approximated exponential e^x

Main ingredients:

- IEEE-745 manipulation is **very fast**
- Polynomial interpolation is suitable for **100% SIMD unit use** (multiply-add instructions)

Main resulting advantages:

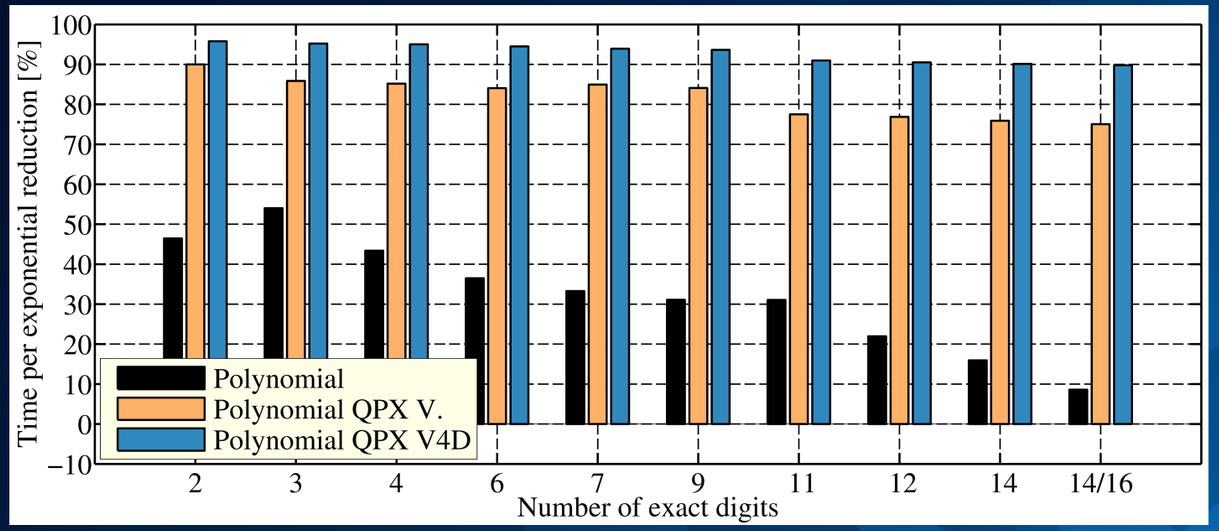
- Huge **reduction in the time-to-solution** (up to 96 % on IBM BG/Q, Power7 and Power8)
- Huge **reduction in the energy-to-solution** (up to 93 % on IBM BG/Q, Power7 and Power8)
- Low-to-High **accuracy flexibility** (the user can control the degree of the polynomial)
- **Architecture flexibility** (specific SIMD implementation possible on any modern architecture)

Further advantages:

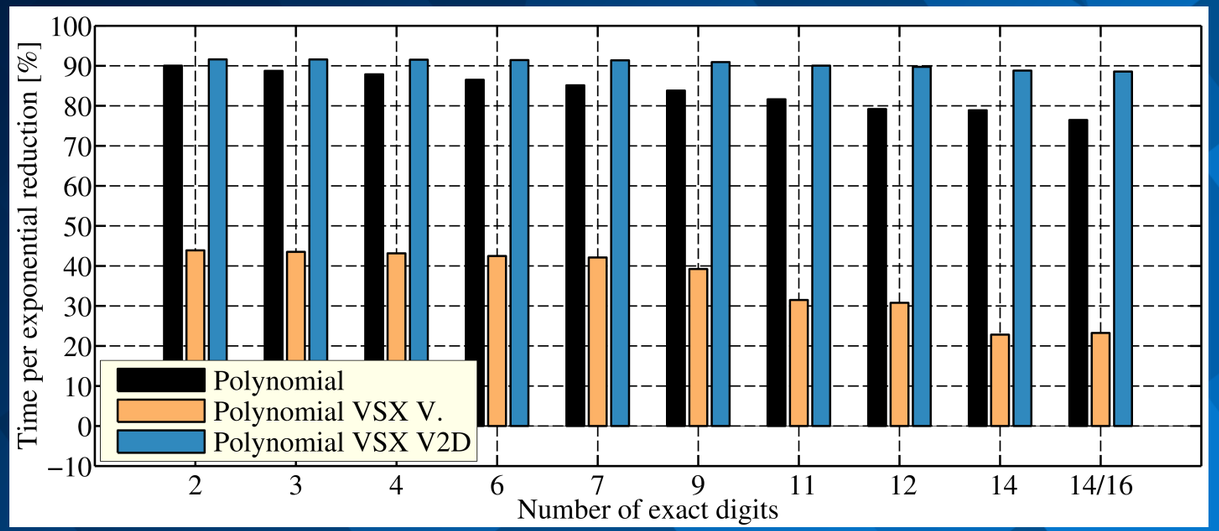
- Scalar versions (no SIMD) are still much faster and energy efficient than classical strategies
- OpenMP (multithread) implementations possible and efficient for big vectors

Performance analysis

exp(x) speedup on BG/Q



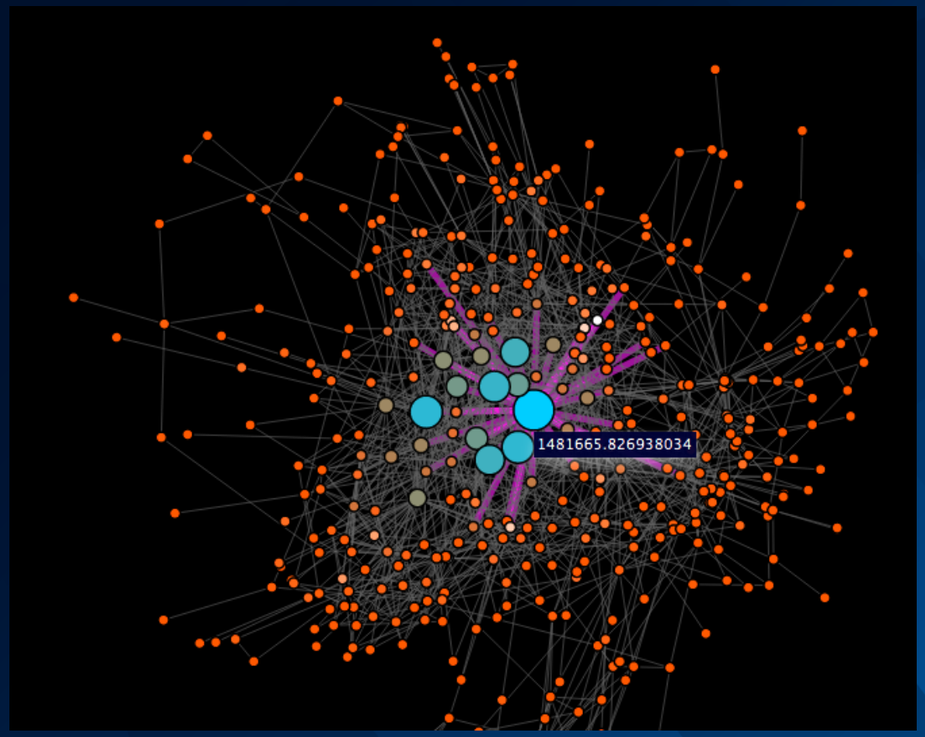
exp(x) speedup on POWER7



Stochastic Algorithms for Large Scale Graph Analytics

Graph analytics paradigm shift: trading accuracy for complexity

- Big data regime demands analytics for large graphs in tens of millions
- Graph with 1 Million nodes requires minutes on fastest machine on the planet
- We believe that accuracy has to be traded for algorithmic complexity and scalability, because hardware will not beat complexity
- O(cN) with c << N will be the only way to tackle big data problems

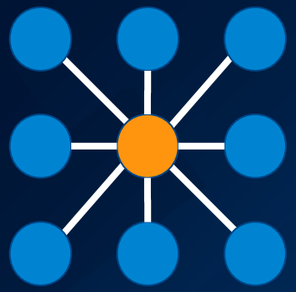


European Street Network:
51 Million nodes, 108 Million edges

Subgraph centrality: quantifying importance of nodes in a graph

Subgraph centrality measures the participation of each node in all subgraphs in a network [1]

[1] Estrada, Subgraph centrality in complex networks, Phys. Rev. E, 2005



Star graph

Subgraphs of size 2

Subgraphs of size 4

- In a star graph the center participates in all 8 subgraphs of size 2 (a line)
- Each other node participate in all 8 subgraphs of size 4, where the center participates in 64 subgraphs of size 4
- This can be computed by counting the weighted number of closed walks starting and ending at the same node
- Intuition: $A + A^2 + A^3 + A^4 + \ldots + A^n$ accumulates all walks with length 1, 2, 3, 4, …, n
- This can be expressed as the diagonal entires of the exponential of the (adjacency) matrix

$$C_e = \sum_{k=0}^{\infty} \frac{(A^k)_{ii}}{k!} \qquad C_e = diag\ e^A$$

where *k!* is a weighting term to prevent divergence, penalizing long walks

Computing the exponential of the adjacency matrix

There are several ways to compute e^A with high computational and storage costs

- Polynomial approximation, e.g., Padé, or Taylor series
- Eigenvalue decomposition

Let us consider the decomposition

$$A = V^T D V$$

where V is orthogonal and D diagonal matrices. With this the exponential can be computed as

$$e^A = V^T e^D V$$

But: Cubic cost of computing: $O(N^3)$

For moderate graph with 10^7 nodes → 10^{21} operations: takes days on state of the art clusters

We do not need all entries of e^A, a stochastic approach approximating individual entries might help..

Approximate exponential of the adjacency matrix

- Compute $A = V^T D V$ iteratively using Krylov subspace techniques
- After m << N iterations of Lanczos (for symmetric adjacency matrices) we can compute

$$A \approx V_m^T D_m V_m$$

where V_m (n x m) are orthonormal basis vectors, and D_m (m x m) is tridiagonal

- Every iteration dominated by one matrix vector product: O(mN) cost
- Lanczos might loose orthogonality. Including re-orthogonalization we have: $O(m^2N)$ cost
- The exponential of D_m can be computed, e.g., using the eigendecomposition

$$D_m = Y \Lambda Y^T; \quad e^{D_m} = Y e^{\Lambda} Y^T$$

where the diagonal matrix $(\exp(\Lambda))_{ii} = \exp(\lambda_i)$

- Since m << N the cost of computing this eigendecomposition $O(m^3)$ is acceptable

Missing ingredient: we need only the diagonal of the exponential, possibly in a matrix-free form fashion!

Matrix-free stochastic diagonal estimator

- In 1989 Hutchinson [1] described an unbiased stochastic trace estimator

$$tr(A) \approx \frac{1}{s}\sum_{k=1}^{s} v_k^T A v_k$$

[1] M.F. Hutchinson, A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines, Comm. in Stat.-Sim. and Comp., vol. 18, pp. 1059-1076, 1989.

where v_k are special "probe vectors" chosen to minimize variance: $v_{k,i}$ element of {-1,1} with equal probability ½

- The idea can be generalized for estimating the diagonal of a matrix: use of s (<< N) probe vectors v_k to estimate the diagonal D of the matrix function F(A)

$$D(\mathcal{F}(A)) = \left[\sum_{k=0}^{s} v_k \odot \mathcal{F}(A) v_k\right] \oslash \left[\sum_{k=0}^{s} v_k \odot v_k\right]$$

where we use element wise multiplication and division (Hadamard).

- F(A) v_k is computing (or approximating) a matrix free matrix vector multiplication
- The method requires only matrix vector products: total cost is O(sN) for a matrix with N elements, but s << N
- Why it works? On average the a_{ij} terms will converge to zero provided that the probe vectors v_k have uniform ± signs

$$D_i^s = a_{ii} + \sum_{j \neq i} a_{ij} \frac{\sum_{k=1}^{s} v_k^i v_k^j}{\sum_{k=1}^{s} \left(v_k^i\right)^2}$$

Approximate exponential of adjacency matrix: full picture

Perform m iterations of Lanczos (with re-orthogonalization) and compute: Cost $O(m^2\ nnz(A))$

$$A \approx V_m^T D_m V_m$$

Define the matrix function Cost $O(m^3)$

$$\mathcal{F}(A) = V_m^T e^{D_m} V_m$$

and plug into diagonal estimator: Cost s of the above Lanczos applications

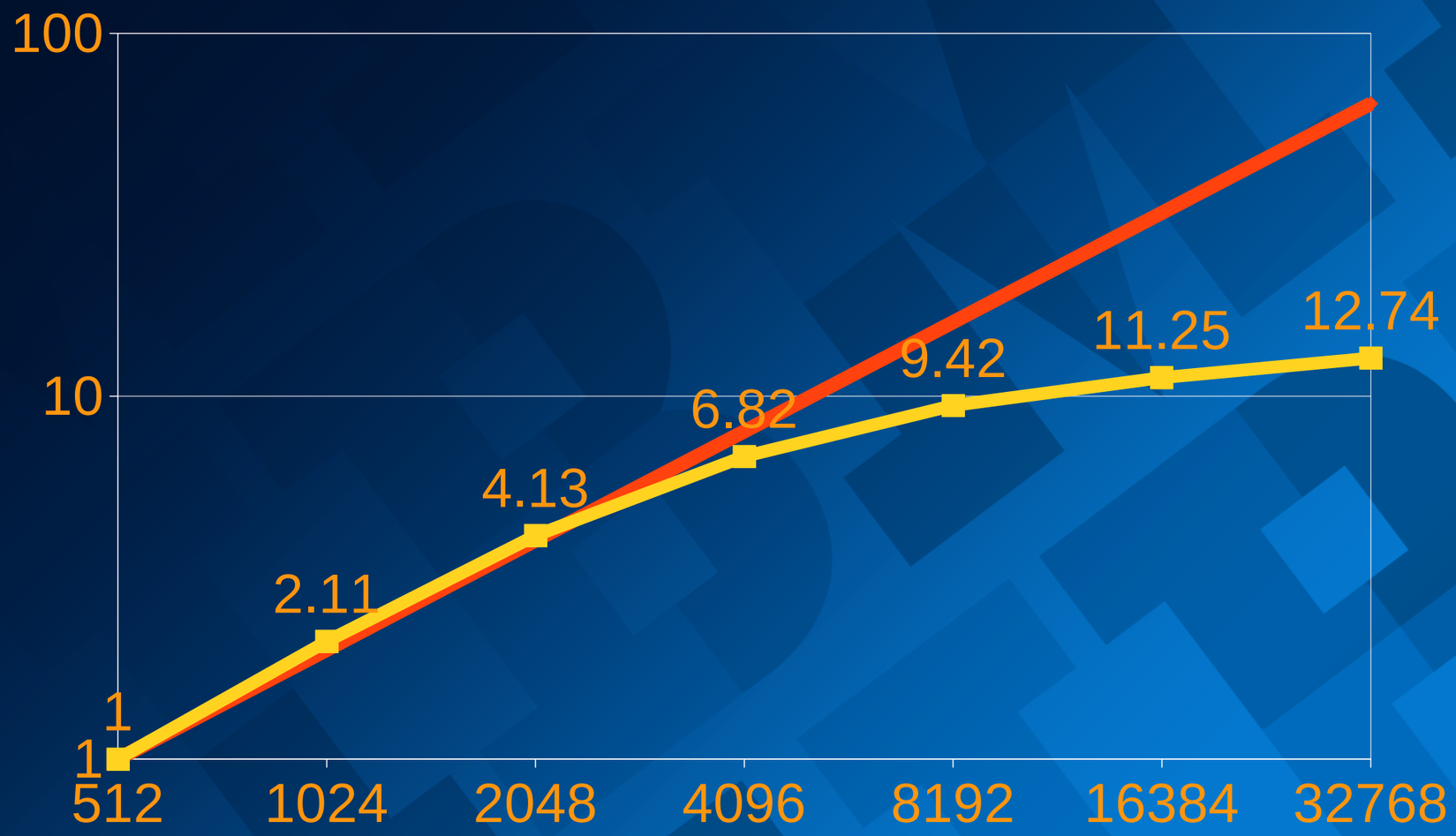
$$D^s = \left[\sum_{k=0}^{s} v_k \odot V_m^T e^{D_m} V_m v_k\right] \oslash \left[\sum_{k=0}^{s} v_k \cdot v_k\right]$$

Total cost: $O(s\ m^2\ nnz(A)+m^3) = O(s\ m^2\ d\ N)$ → Near linear cost

- d is the average degree
- s the number of probe vectors v_k

Strong scalability on BG/Q (up to 2 racks – early version of the code)

- We consider the European street network, ~51 million of nodes
- With conventional techniques, more than 1 ExaFLOP would be necessary to analyze this data
- On 2 BG/Q racks we compute the subgraph centrality in less than 30 seconds, with a stopping criteria of 1e−3



References

- J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. J. Staar, Y. Ineichen, C. Bekas, A. Curioni, O. Ghattas. An Extreme-scale Implicit Solver for Complex PDEs: Highly Heterogeneous Flow in Earth's Mantle. SC'15, ACM, 2015. – **ACM Gordon Bell Prize Winner 2015**
- A. C. I. Malossi, Y. Ineichen, C. Bekas, A. Curioni. Fast Exponential Computation on SIMD Architectures. HiPEAC 2015 - 1st Workshop On Approximate Computing (WAPCO), 2015
- P. Klavik, A. C. I. Malossi, C. Bekas, A. Curioni. Changing computing paradigms towards power efficiency. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372(2018), The Royal Society, 2014
- C. Bekas, E. Kokiopoulou, Y. Saad. An estimator for the diagonal of a matrix. Applied numerical mathematics, 2007
- G. Kollias, Y. Ineichen, H. Avron, V. Sindhwani, K. Clarkson, C. Bekas, A. Curioni. libSkylark: A Framework for High-Performance Matrix Sketching for Statistical Computing, SC'15, Poster, 2015
- F. E. Faisal, Y. Ineichen, A. C. I. Malossi, P. Staar, C. Bekas and A. Curioni. Massively Parallel and Near Linear Time Graph Analytics, SC'14, Poster, 2014

THANK YOU!