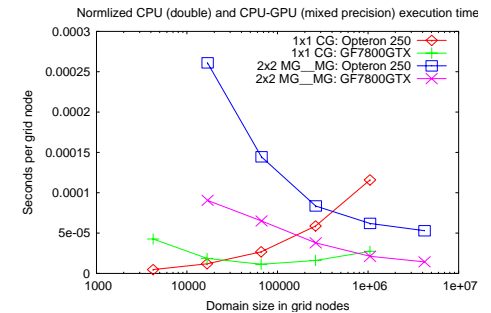
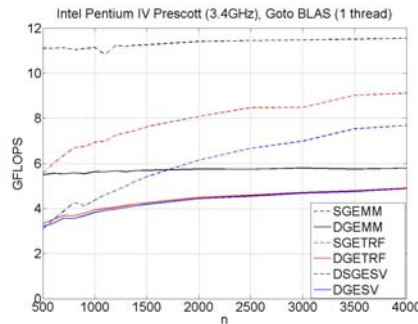
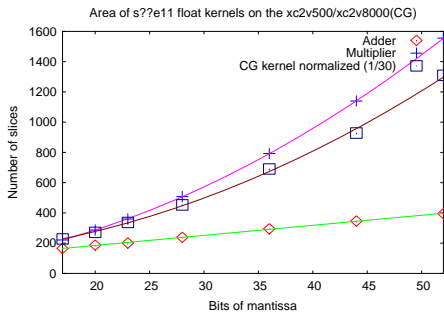


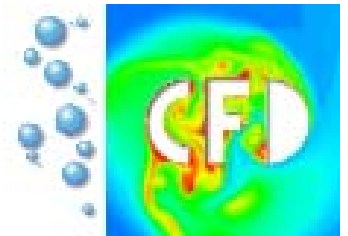
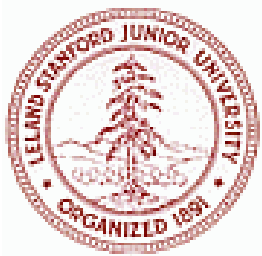
# Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations



# Collaboration

---

- Dominik Göddeke, Stefan Turek, FEAST Group (Dortmund)
- Patrick McCormick, Advanced Computing Lab (LANL)
- Robert Strzodka, Max Planck Center (Stanford)



# Hardware Oriented?

---

- **GPU:** 249 GFLOPS single precision  
160 GB/s internal bandwidth  
54 GB/s external bandwidth
- **FPGA:** 192 mad25x18 at 550MHz + logic  
almost unrestricted internal bandwidth  
120.0 GB/s external bandwidth (for all IO pins)
- **Clearspeed:** 50 GFLOPS double precision  
200.0 GB/s internal bandwidth  
6.4 GB/s external bandwidth
- **Cell BE:** 230 GFLOPS single, 21 GFLOPS double precision  
204.8 GB/s internal bandwidth  
25.6 GB/s external bandwidth
- **Buses:** PCI-X 1GB/s; PCIe 4GB/s

# Overview

---

- **Precision and Accuracy**
- **Hardware Resources**
- **Mixed Precision Iterative Refinement**
- **Large Range Scaling**

# Roundoff and Cancellation

---

Roundoff examples for the **float s23e8** format

additive roundoff	$a = 1 + 0.000000004$	$=_{fl} 1$
multiplicative roundoff	$b = 1.0002 * 0.9998$	$=_{fl} 1$
<b>cancellation</b> $c=a,b$	$(c-1) * 10^8$	$=_{fl} 0$

Cancellation promotes the small error **0.000000004** to the absolute error **4** and a relative error **1**.

**Order** of operations can be crucial:

$$1 + 0.000000004 - 1 =_{fl} 0$$

$$1 - 1 + 0.000000004 =_{fl} 0.000000004$$

# More Precision

Evaluating (with powers as multiplications)

[S.M. Rump, 1988]

$$f(x, y) = (333.75 - x^2) y^6 + x^2 (11x^2 y^2 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

for  $x_0 = 77617$ ,  $y_0 = 33096$  gives

float s23e8

1.1726

double s52e11

1.17260394005318

long double s63e15

1.172603940053178631

This is all wrong, even the sign is wrong!! The correct result is

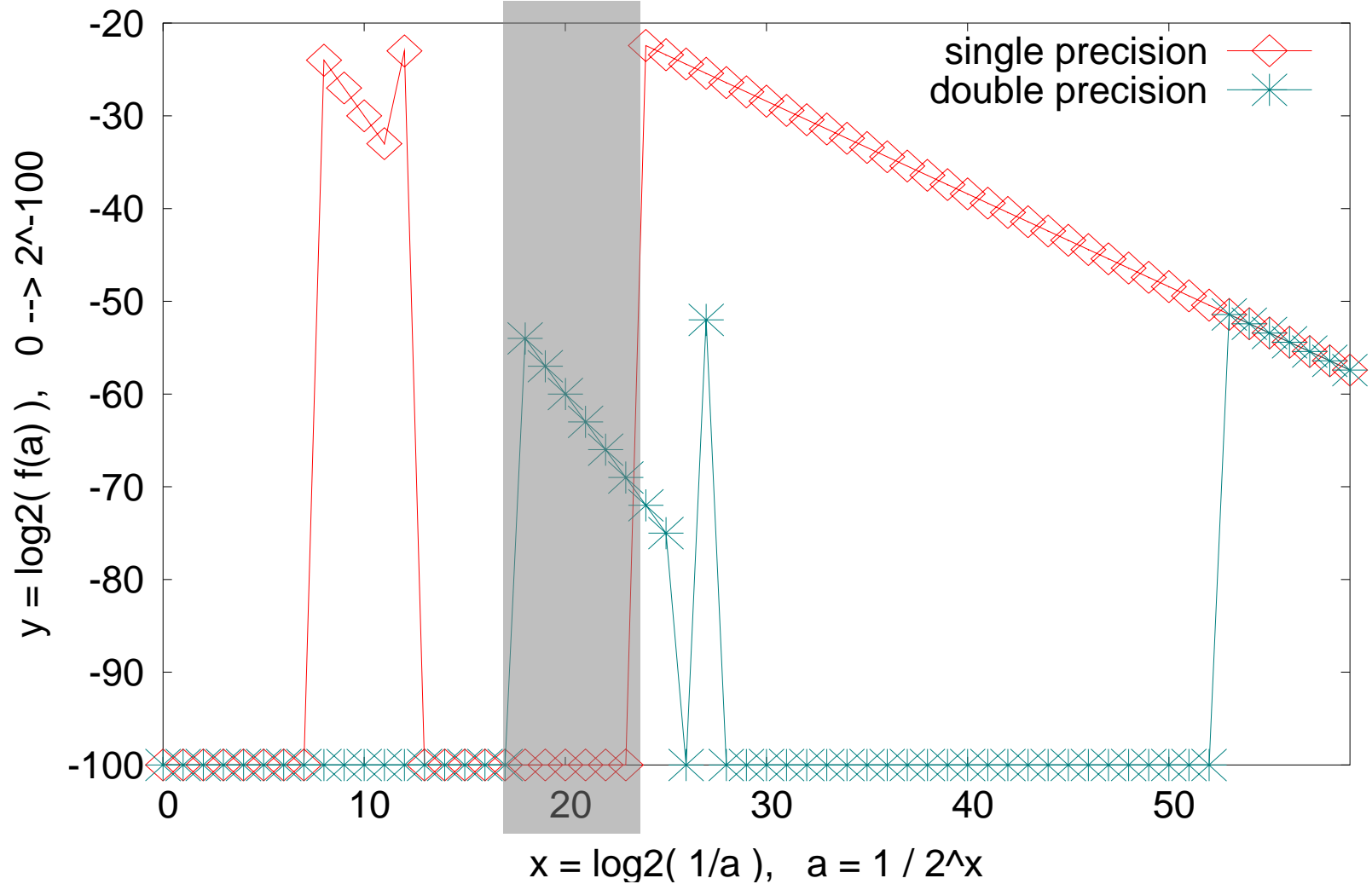
-0.82739605994682136814116509547981629...

Lesson learnt: **Computational Precision**  $\neq$  **Accuracy of Result**

# The Erratic Roundoff Error

Roundoff error for:  $0 = f(a) := |(1+a)^3 - (1+3a^2) - (3a+a^3)|$

← Smaller is better ←



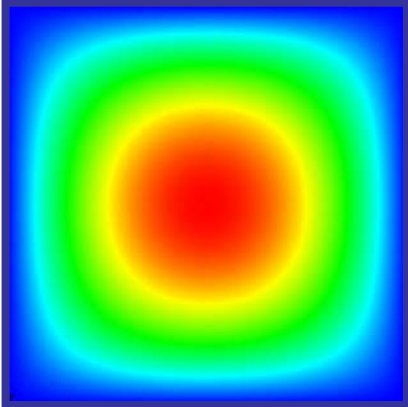
# Precision and Accuracy

---

- There is no **monotonic** relation between the **computational precision** and the **accuracy of the final result**.
- Increasing precision can **decrease** accuracy !
- The increase or decrease of precision in different parts of a computation can have **very different impact** on the accuracy.
- The above can be exploited to **significantly reduce** the precision in parts of a computation without a loss in accuracy.
- We obtain a **mixed precision method**.

# PDE Example: Poisson Problem

---

- Unit square
  - Bilinear conforming FEs (Q1)
  - Zero Dirichlet BCs
  - Regular refinement
- 
- FEM theory: pure discretization error
  - Expected error reduction of 4 (i.e.  $h^2$ ) in each level
  - Solved using MG until norms of residuals *indicate* three digit relative reduction
  - Comparison of integral L2 error against analytically known reference solution in double precision

# PDE Example: Poisson Problem

---

- Error reduction: single and double precision

NEQs	SINGLE	REDUCTION	DOUBLE	REDUCTION
$3^2$	5.208e-3		5.208e-3	
$5^2$	1.440e-3	3.62	1.440e-3	3.62
$9^2$	3.869e-4	3.72	3.869e-4	3.72
$17^2$	1.015e-4	3.81	1.015e-4	3.81
$33^2$	2.611e-5	3.89	2.607e-5	3.89
$65^2$	6.464e-6	4.04	6.612e-6	3.94
$129^2$	1.656e-6	3.90	1.666e-6	3.97
$257^2$	<b>5.927e-7</b>	<b>2.79</b>	4.181e-7	3.98
$513^2$	<b>2.803e-5</b>	<b>0.02</b>	1.047e-7	3.99
$1025^2$	<b>7.708e-5</b>	<b>0.36</b>	2.620e-8	4.00

# Overview

---

- Precision and Accuracy
- **Hardware Resources**
- Mixed Precision Iterative Refinement
- Large Range Scaling

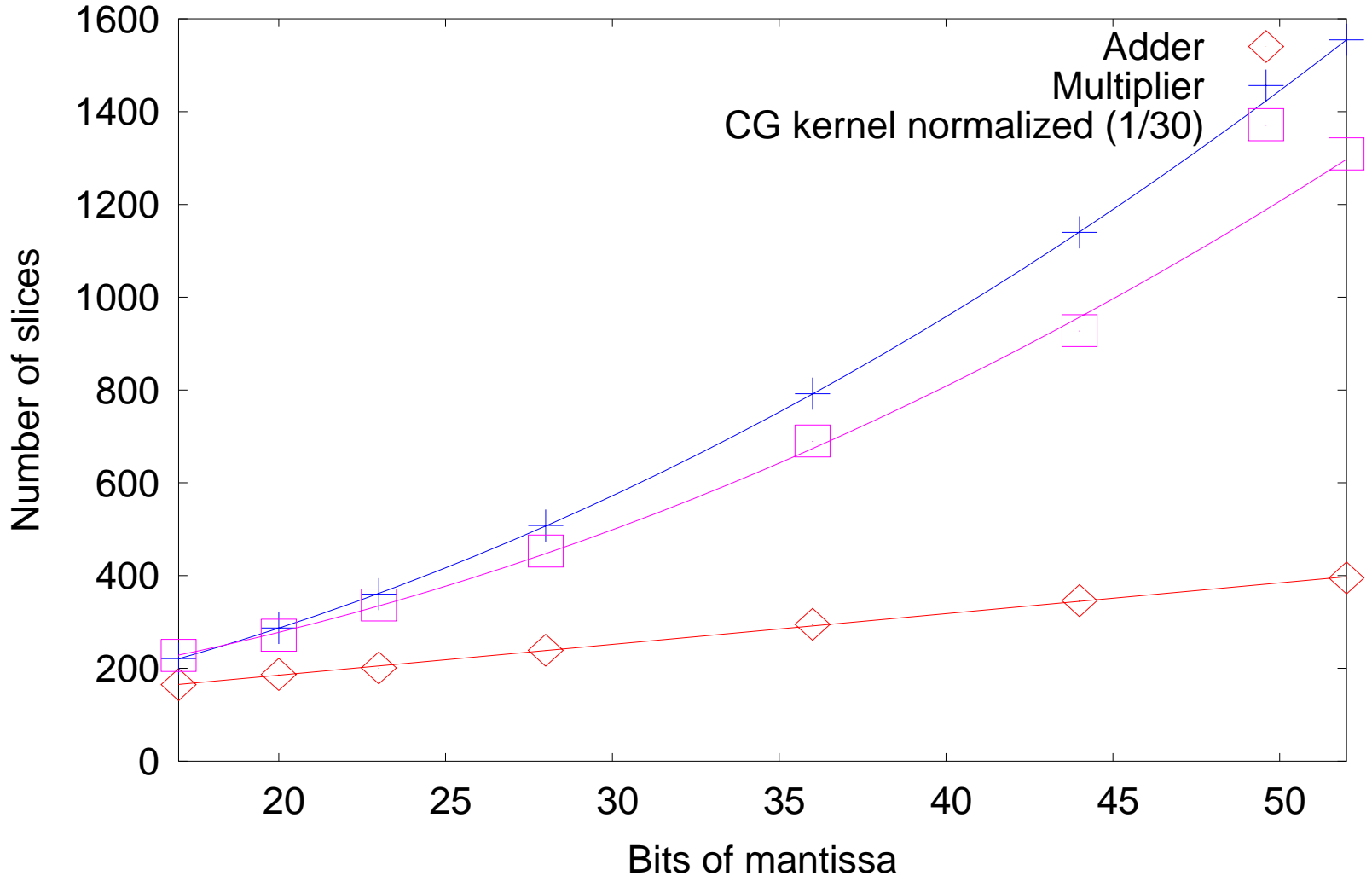
# Resource Consumption for Integer Operations

Operation	Area	Latency
$\min(r,0)$ $\max(r,0)$	<b><math>b+1</math></b>	<b>2</b>
$\text{add}(r_1,r_2)$ $\text{sub}(r_1,r_2)$	<b><math>2b</math></b>	<b><math>b</math></b>
$\text{add}(r_1,r_2,r_3) \rightarrow \text{add}(r_4,r_5)$	<b><math>2b</math></b>	<b>1</b>
$\text{mult}(r_1,r_2)$ $\text{sqr}(r)$	<b><math>b(b-2)</math></b>	<b><math>b \log(b)</math></b>
$\text{sqrt}(r)$	<b><math>2c(c-5)</math></b>	<b><math>c(c+3)</math></b>

**b: bitlength of argument, c: bitlength of result**

# Reconfigurable Logic Consumption on a FPGA

Area of s??e11 float kernels on the xc2v500/xc2v8000(CG)



# High Precision Emulation

---

- **Combine two native floating point values**
  - Effectively doubles mantissa (but < double float mantissa)
  - Common exponent used to align mantissas
  - Normalisation required after each operation
  - **Dynamic range** slightly smaller than native range
  - Increases op count by **11x (ADD)** and **18-32x (MUL)**
  - **Doubles bandwidth** requirements
- **Example**
  - Two **s23e8** yield a quasi **s46e8** float with same storage requirements as a full **s52e11** double
  - For large scale problems, we will need **quad** precision.

# Example: Addition $c=a+b$

---

- **Compute high-order sum and error**

$$t1 = a.hi + b.hi$$

$$e = t1 - a.hi$$

- **Compute low order term including error and overflows**

$$t2 = ((b.hi - e) + (a.hi - (t1 - e))) + a.lo + b.lo$$

- **Normalise to get final result**

$$c.hi = t1 + t2$$

$$c.lo = t2 - (c.hi - t1)$$

# Precision – Performance: Rule of Thumb

---

- **Reconfigurable device, e.g. FPGA**
  - $\frac{1}{2}$  precision (native format)  $\rightarrow$  4x performance
  - 2x precision (emulated format)  $\rightarrow$   $\frac{1}{4}$  performance
  - Emulated formats are (almost) **identical** to native formats
  
- **Hardwired device, e.g. CPU, GPU**
  - $\frac{1}{2}$  precision (native format)  $\rightarrow$  2x performance
  - 2x precision (emulated format)  $\rightarrow$   $< \frac{1}{10}$  performance
  - Emulated formats are **different** from native formats

# Overview

---

- Precision and Accuracy
- Hardware Resources
- **Mixed Precision Iterative Refinement**
- Large Range Scaling

# Mixed Precision Iterative Refinement

---

- Exploit the **speed** of low precision and obtain a result of high **accuracy**

$$\mathbf{d}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$$

$$\mathbf{A}\mathbf{c}_k = \mathbf{d}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{c}_k$$

$$k = k + 1$$

**Compute** in high precision (cheap)

**Solve** in low precision (fast)

**Correct** in high precision (cheap)

Iterate until convergence in high precision

- Low precision solution is used as a preconditioner in a high precision iterative method
  - A is small and dense: Solve  $\mathbf{A}\mathbf{c}_k = \mathbf{d}_k$  directly
  - A is large and sparse: Solve (approximately)  $\mathbf{A}\mathbf{c}_k = \mathbf{d}_k$  with an iterative method itself

# Direct Scheme for Small, Dense A

---

- **Algorithm**

- Compute  $PA=LU$  once in single precision
- Use LU decomposition to solve  $Ly=Pd_k, Uc_k=y$  in each step
- Included in the next release of LAPACK

- **Main reasons for speedup**

- Computation of LU decomposition is  $O(n^3)$
- Computation of LU is much faster in single than in double
- Solution using LU for several RHS is only  $O(n^2)$

- **Upper bound for iteration count**

- $\text{ceil}(t_d/(t_s-K))$ , where  $K, t_d, t_s$  are  $\log_{10}$  of matrix condition and double and single precision (e.g.  $t_d$  approx 16)

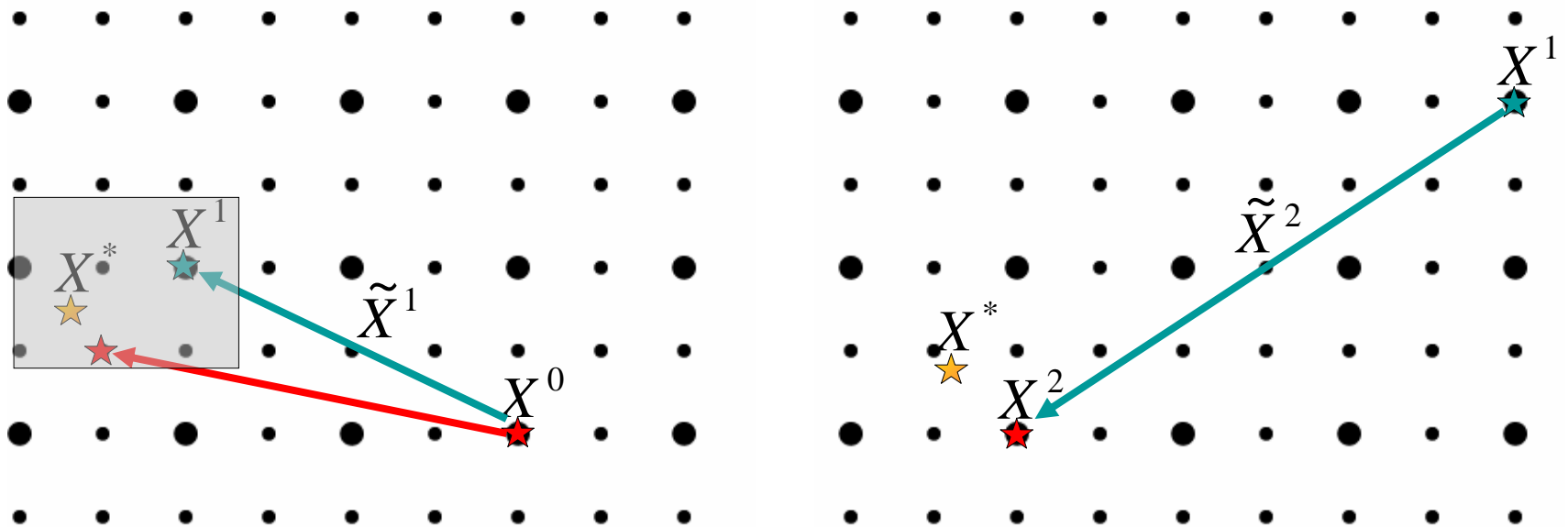
# Iterative Refinement: First and Second Step

We solve  $AX = B$  with iterative refinement

$$B^{k+1} := B - AX^k,$$

$$A\tilde{X}^{k+1} = B^{k+1},$$

$$X^{k+1} := X^k + \tilde{X}^{k+1}$$

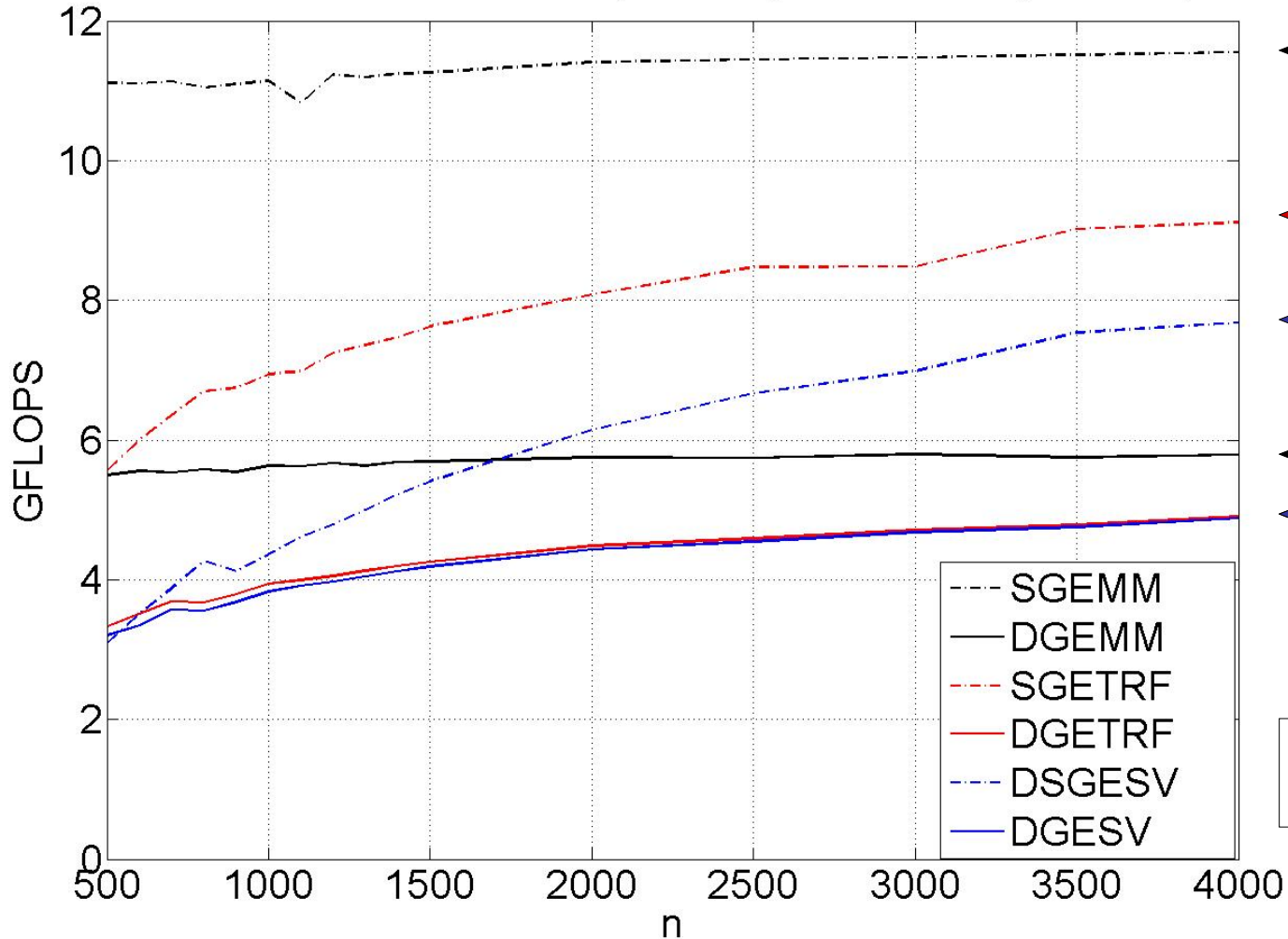


★ ← ★ High precision path  
through fine nodes

★ ← ★ Low precision path  
through coarse nodes

# CPU Results: LU Solver

Intel Pentium IV Prescott (3.4GHz), Goto BLAS (1 thread)



→ Larger is better →

chart courtesy  
of Jack Dongarra

[Langou et al. *Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)*, SC 2006]

# Cell Results: LU Solver

IBM Cell 3.2 GHz,  $Ax = b$  Performance

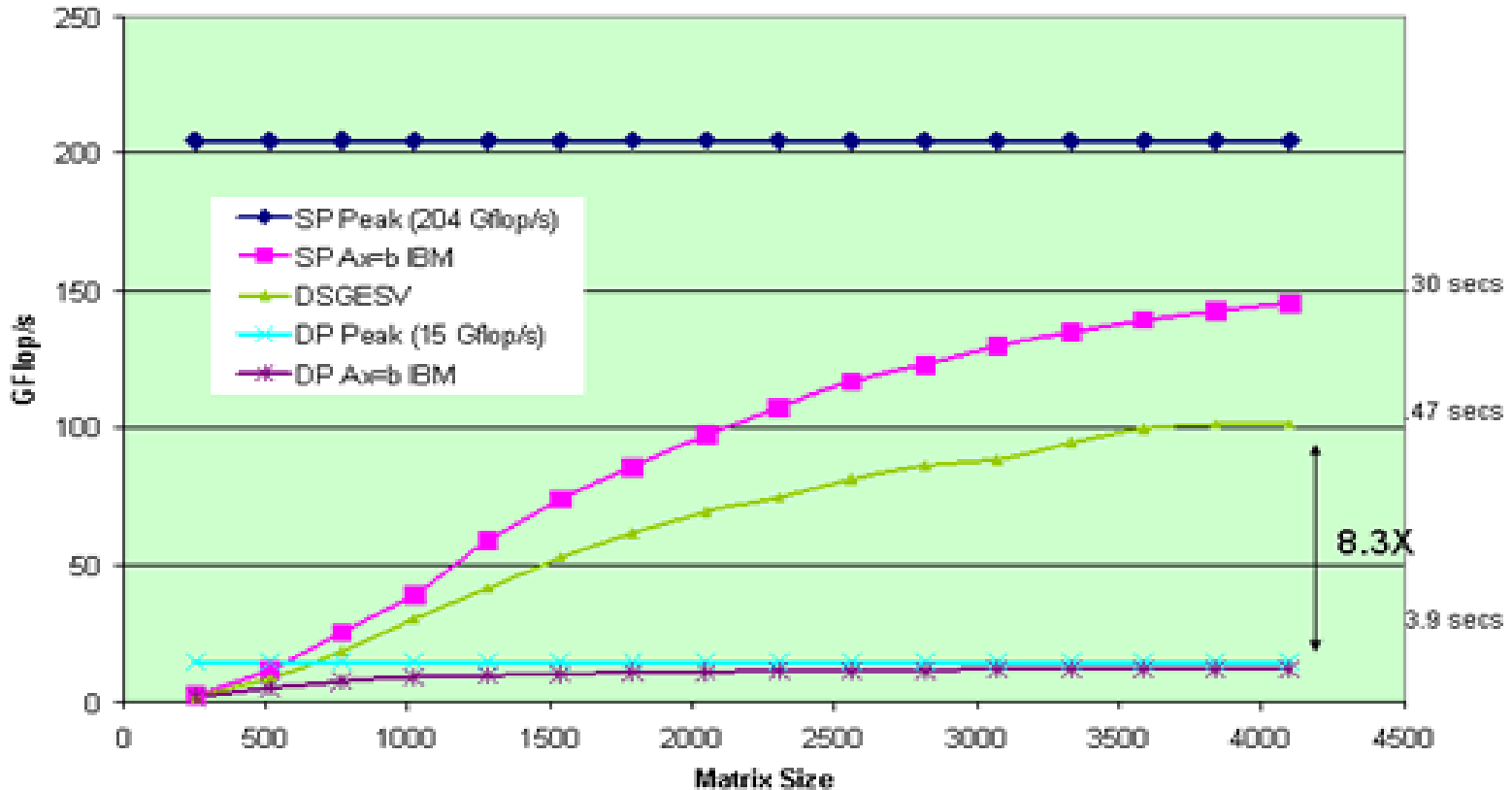


chart courtesy  
of Jack Dongarra

[Langou et al. *Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)*, SC 2006]

# Iterative Scheme for Large, Sparse A

---

- **Algorithm**

- Inner solver: CG, Multigrid
- Terminate inner solver after fixed number of iterations, fixed error reduction or convergence

- **Main reason for speedup**

- Inner solver can run on the GPU close to peak bandwidth
- To obtain speedups on CPUs one would need a SIMD optimized sparse matrix vector product

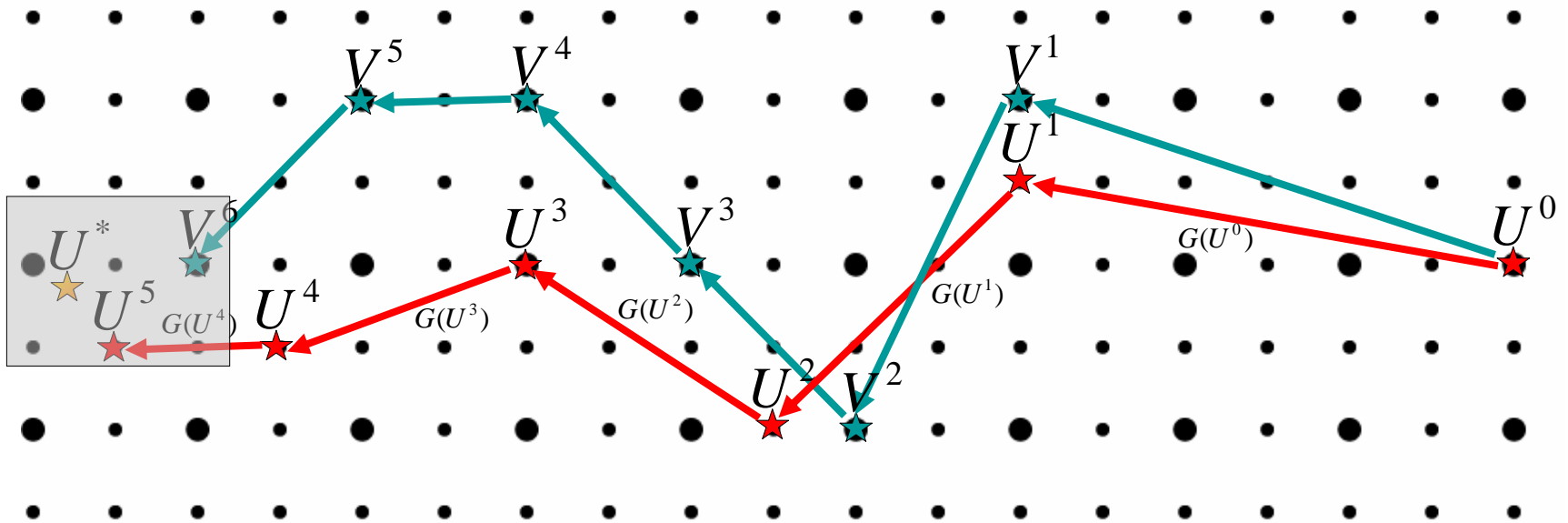
- **Result comparison**

- 4-5x against double, 2-3x against double emulation
- Memory requirements cut in half

# Iterative Convergence: First Correction

Convergent iterative scheme

$$U^{k+1} := U^k + G(U^k), \quad (U^k)_k \xrightarrow{k \rightarrow \infty} U^*, \quad (G(U^k))_k \xrightarrow{k \rightarrow \infty} 0$$



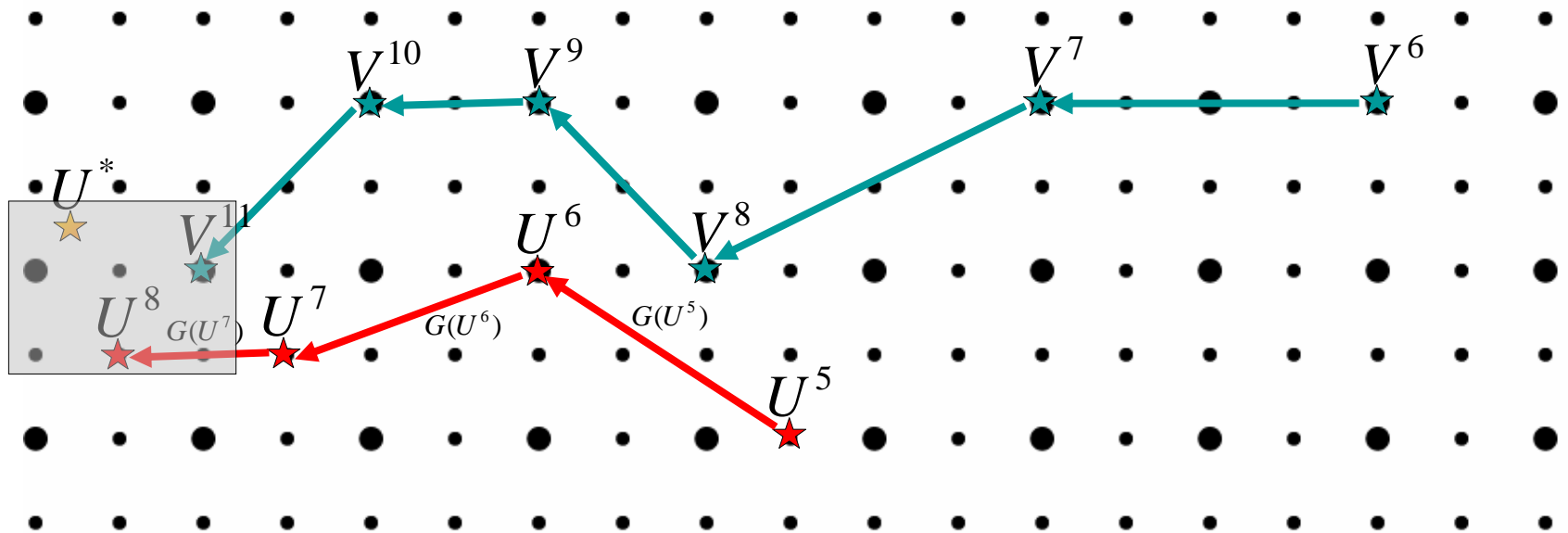
★ ← ★ High precision path  
through fine nodes

★ ← ★ Low precision path  
through coarse nodes

# Iterative Convergence: Second Correction

Convergent iterative scheme

$$U^{k+1} := U^k + G(U^k), \quad (U^k)_k \xrightarrow{k \rightarrow \infty} U^*, \quad (G(U^k))_k \xrightarrow{k \rightarrow \infty} 0$$

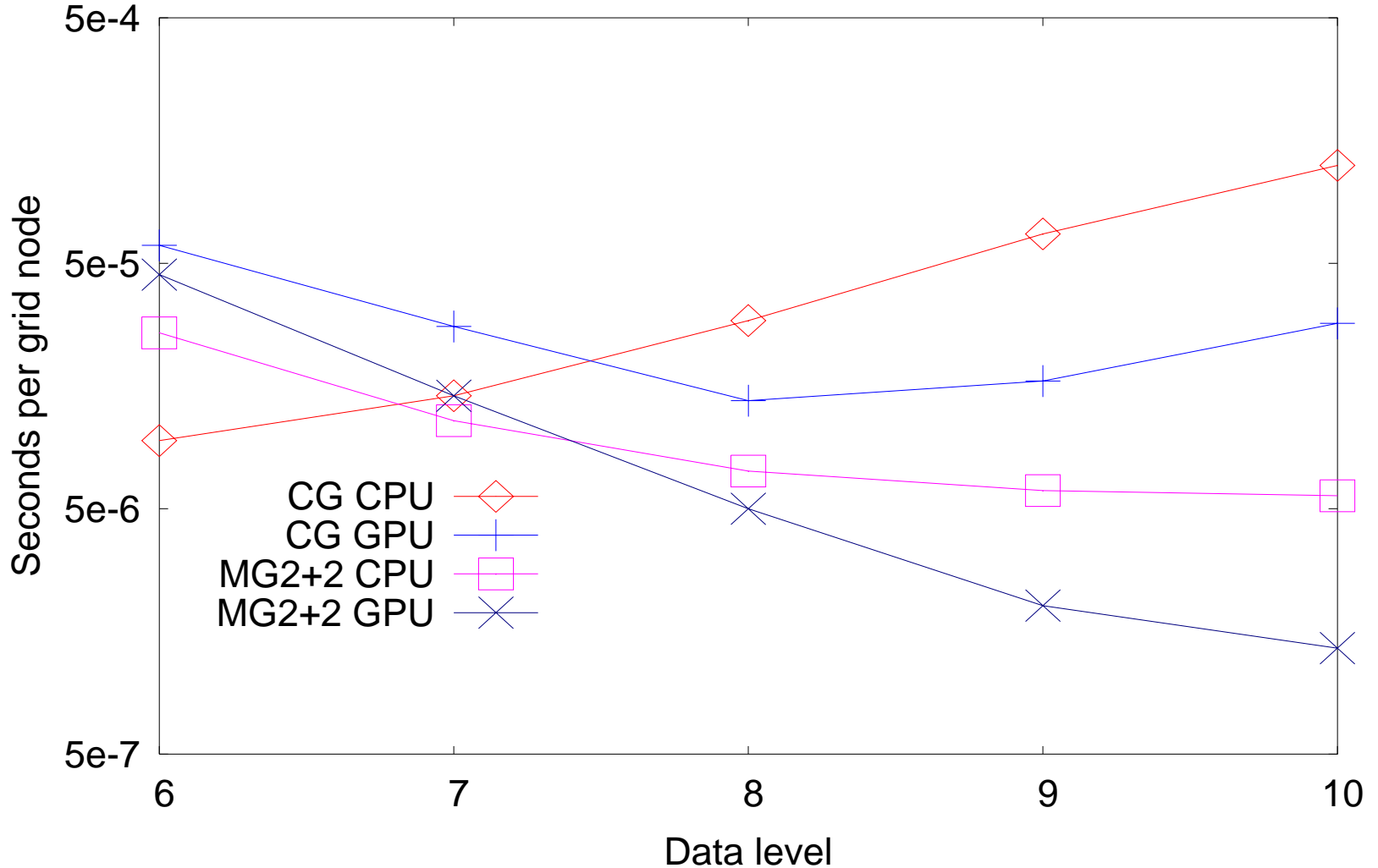


★ ← ★ High precision path  
through fine nodes

★ ← ★ Low precision path  
through coarse nodes

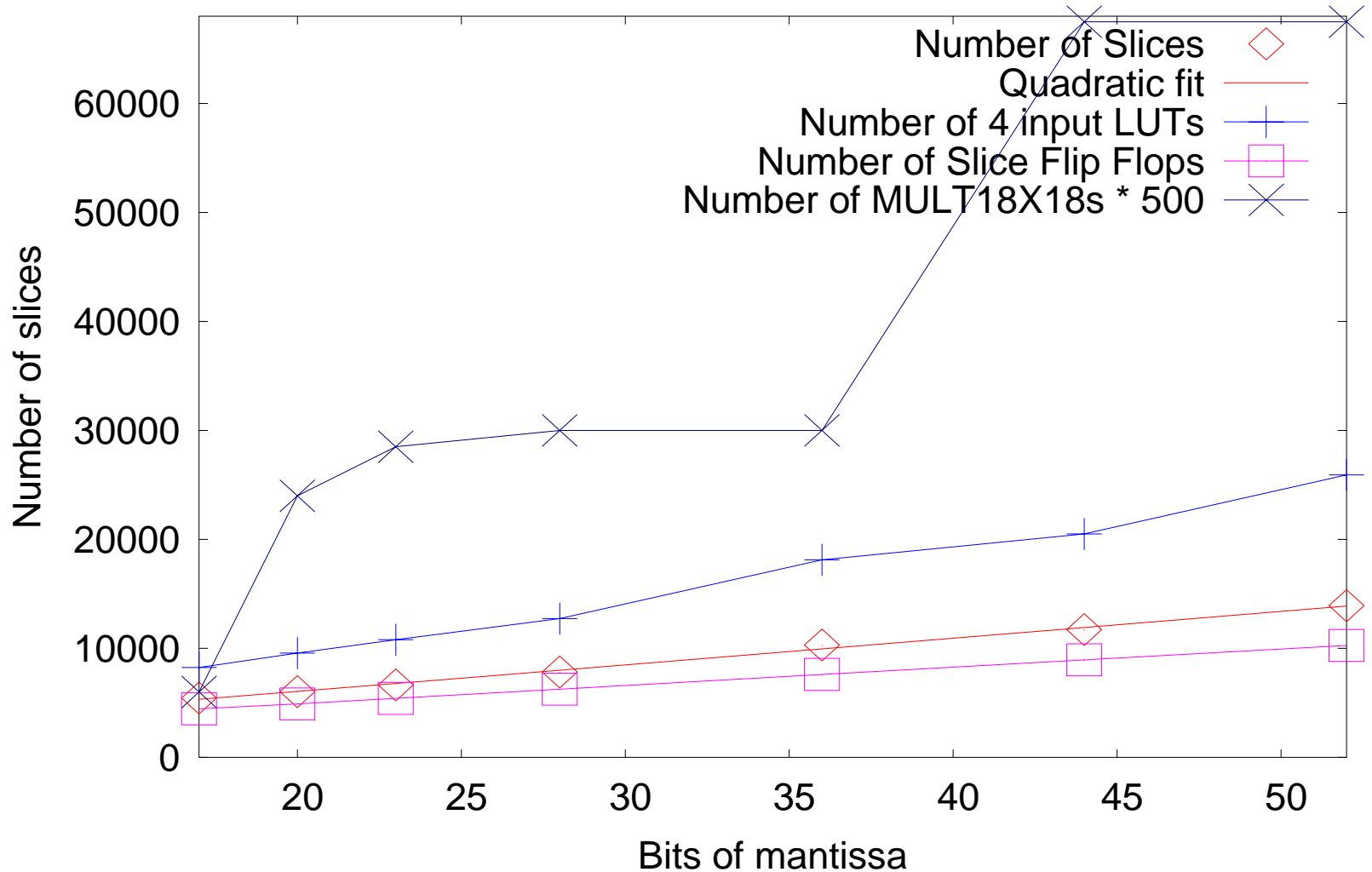
# GPU Results: Conjugate Gradient (CG) and Multigrid (MG)

Performance of double precision CPU and mixed precision CPU-GPU solvers



# FPGA Results: Conjugate Gradient with MUL18x18

Area of Conjugate Gradient s??e11 float kernels on the xc2v8000

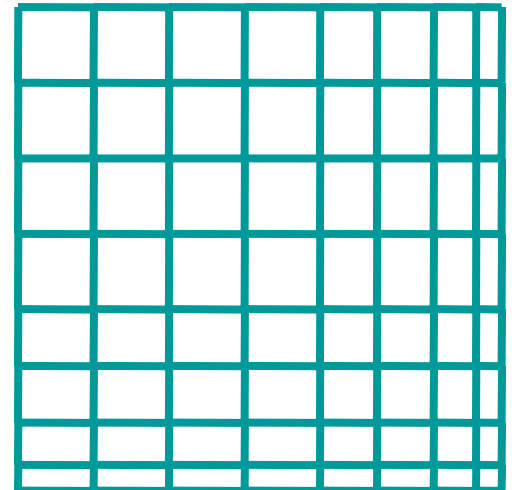
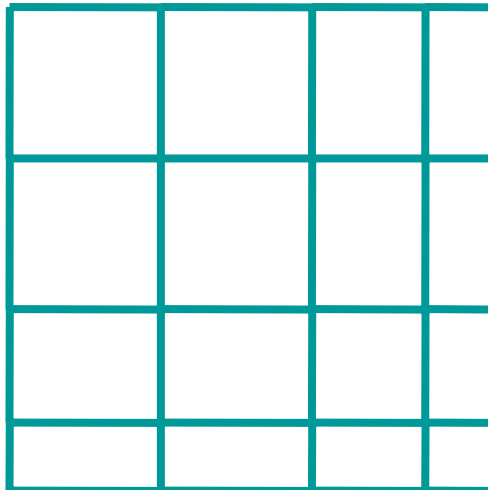
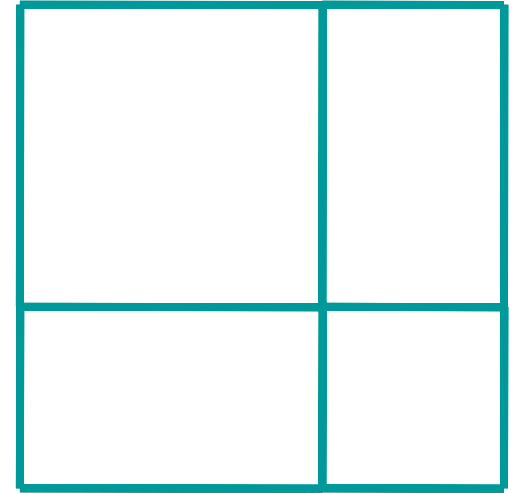
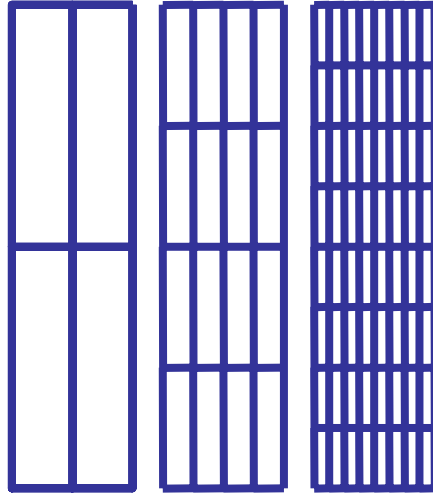


← Smaller is better ←

# Testing Grid Anisotropies

---

- Test iterative refinement with common FEM grids
- Uniform anisotropy, e.g. principal direction
- Anisotropic refinement, e.g. boundary layer



# Overview

---

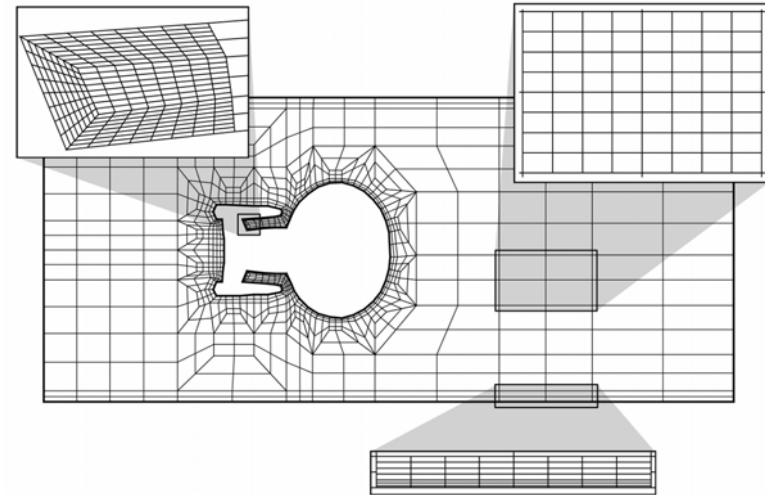
- Precision and Accuracy
- Hardware Resources
- Mixed Precision Iterative Refinement
- **Large Range Scaling**

# FEAST: Generalized Tensor-Product Grids

---

- **Sufficient flexibility in domain discretization**

- Global **unstructured** macro mesh, domain decomposition
- **(an)isotropic** refinement into local tensor-product grids



- **Efficient computation**

- High **data locality**, **large problems** map well to clusters
- **Problem specific solvers** depending on anisotropy level
- **Hardware accelerated solvers** on regular sub-problems

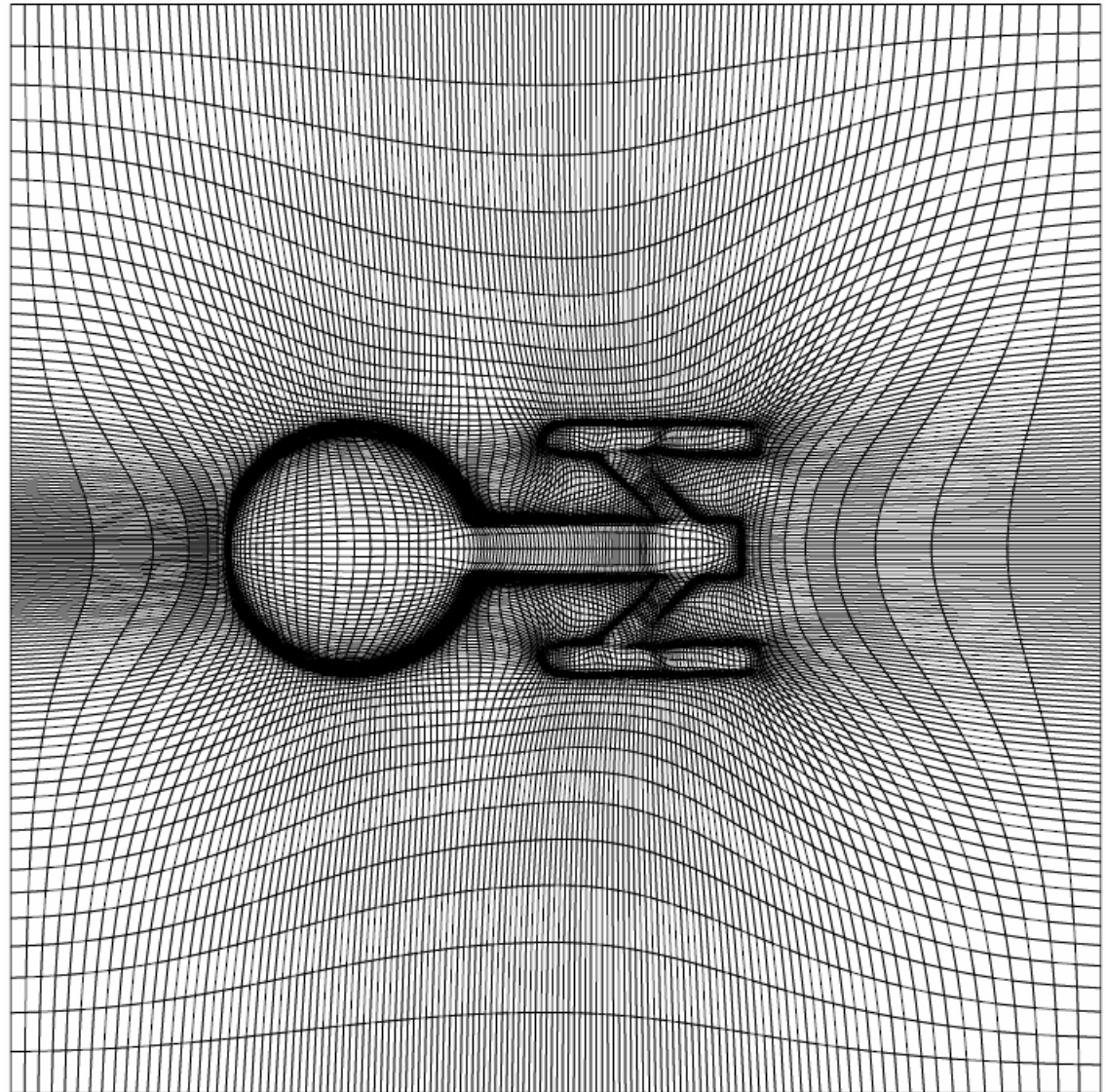
# FEAST and ScaRC

---

- **Find and exploit locally structured parts**
  - Regular data structures are key to good performance
  - High GFLOP/s rates
  - Offloading to hardware co-processors possible
- **Find and hide locally anisotropic parts**
  - Robust solver with good convergence rates
  - Local anisotropies do not impact global multigrid
- **Combine advantages of parallel MG and DD**
  - ScaRC = Scalable Recursive Clustering

# FEAST: Deformation Adaptivity

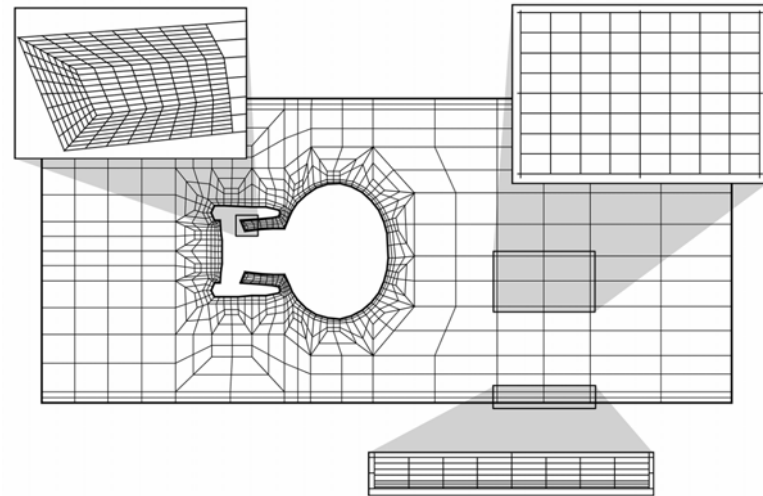
- This grid is a **tensor-product** !
- Easier to **accelerate in hardware** than resolution adaptive grids
- **Anisotropy level** determines optimal solver



# FEASTGPU

---

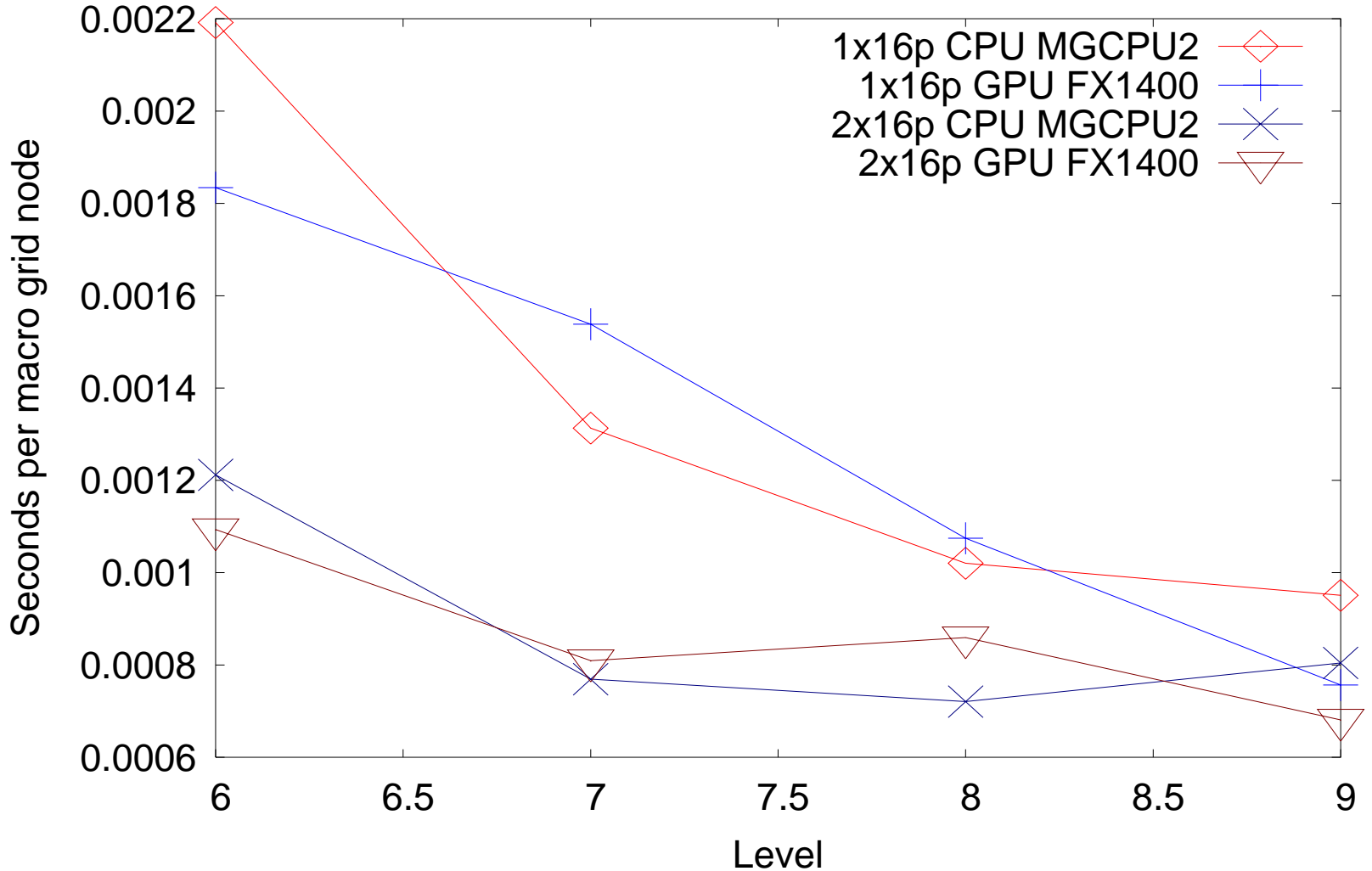
- **Minimally invasive integration**
  - Do not reinvent the wheel, do not rewrite FEAST fully
  - Offload time-critical parts to the GPU
- **CPU and GPU both execute what they are best at**
  - CPU: global high precision MG
  - CPU: communication
  - GPU: local smoothing
  - (CPU: complex local smoothers)
  - Mixed precision MG-MG



# FEAST: Ad-hoc GPU Cluster Performance

CPU, GPU Performance Study for 1x16p, 2x16p (Threshold=20K)

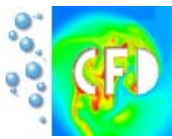
← Smaller is better ←



# Conclusions

---

- The relation between **computational precision** and **final accuracy** is not monotonic
- **Iterative refinement** allows to reduce the precision of many operations without a loss of final accuracy
- In **multiplier** dominated designs the resulting savings grow **quadratically** (area or time)
- Area or time improvements are particularly large for **parallel architectures**: FPGA, CPU, GPU, Cell, etc.
- **Minimally invasive** hardware integration enables portable code with hardware performance



**Robert Strzodka, Stanford University**  
**Dominik Goddeke, Universitt Dortmund**

---

# **Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations**

**[www.stanford.edu/~strzodka/](http://www.stanford.edu/~strzodka/)**

**[www.mathematik.uni-dortmund.de/~goeddeke/](http://www.mathematik.uni-dortmund.de/~goeddeke/)**