ExaStencils

SPPEXA http://www.exastencils.org/

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

UNIKASSEL VERSITÄT

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Matthias Bolten
Lisa Claus
Hannah Rittich

design
Jürgen Teich
Frank Hannig
Christian Schmitt

HLSS
Ulrich Rüde
Harald Köstler
Sebastian Kuckuk
Georg Altmann

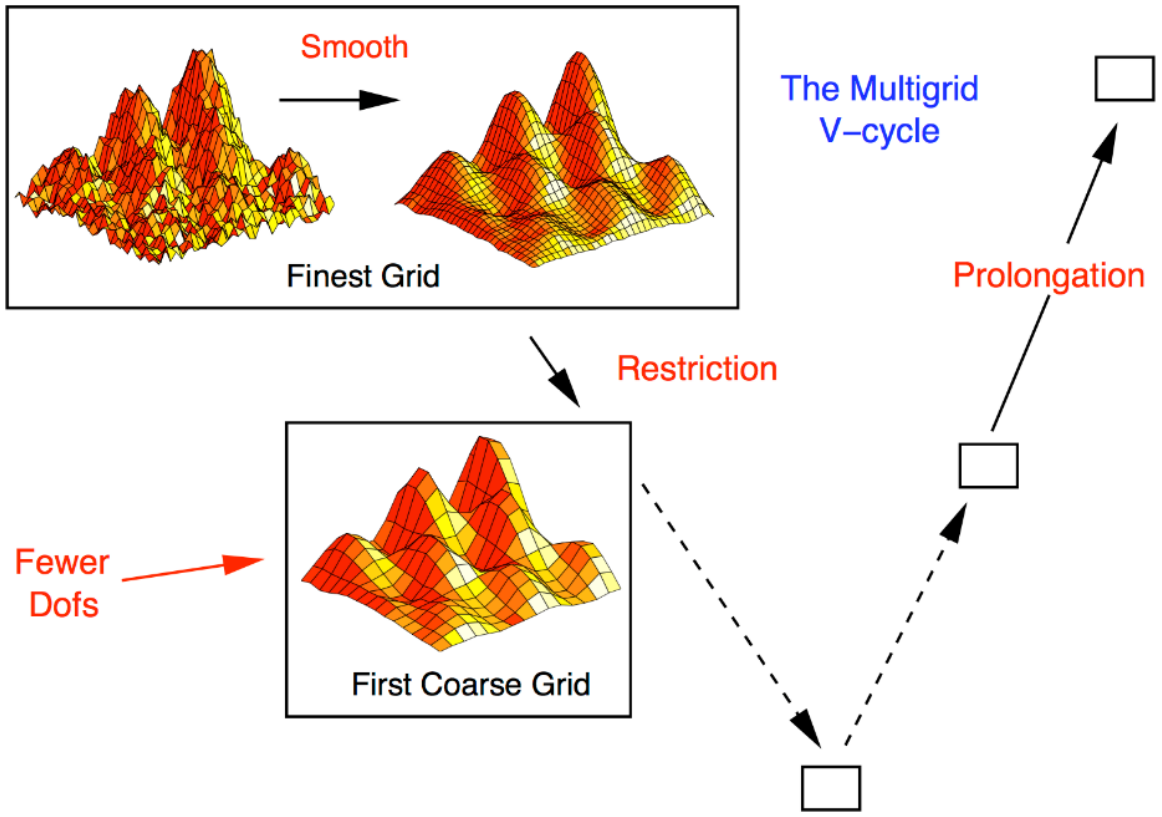
Christian Lengauer
Armin Größlinger
Stefan Kronawitter

Chair of Software Engineering
Sven Apel
Alexander Grebhahn

東京大学 THE UNIVERSITY OF TOKYO

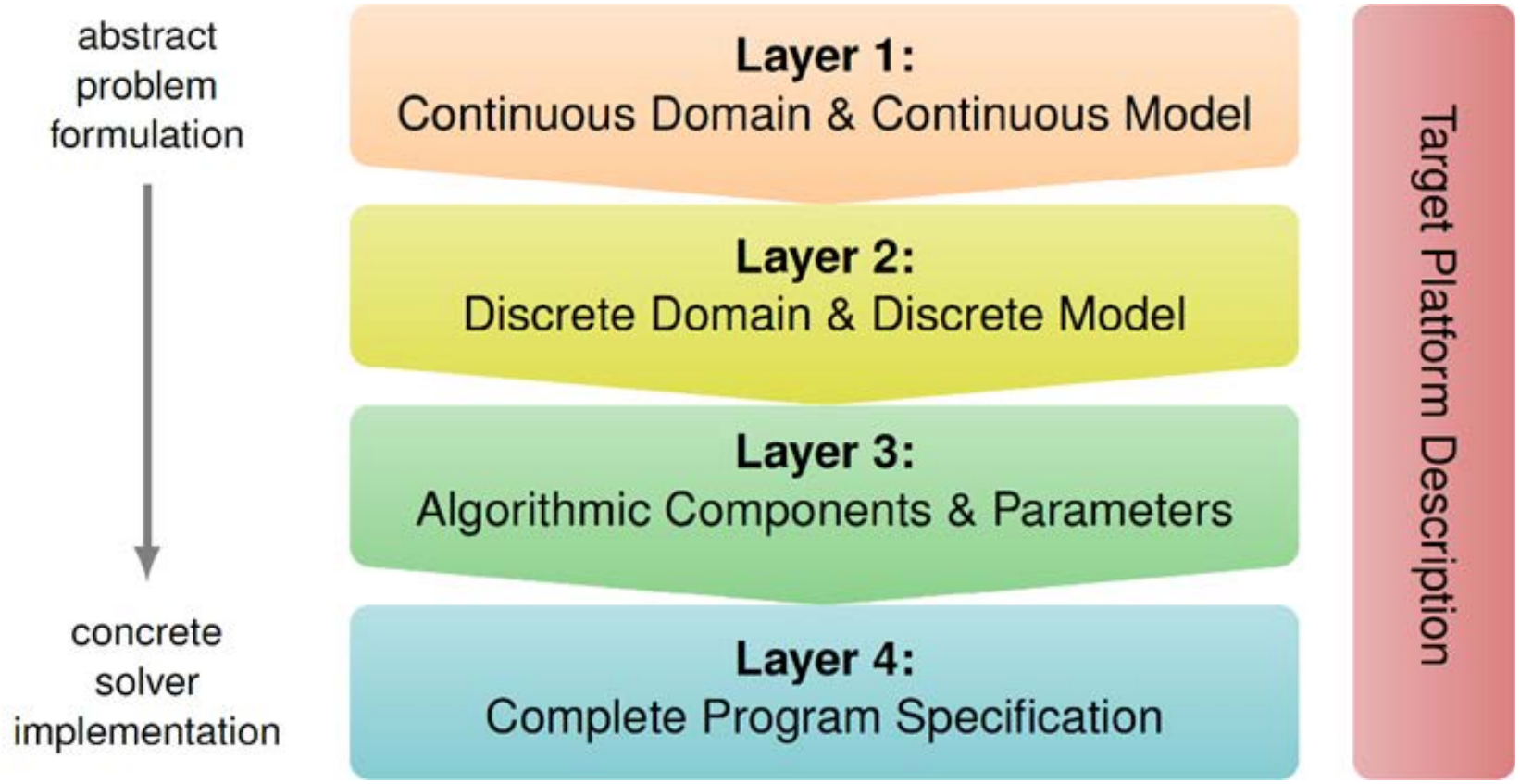
千葉 滋 研究室
Core Software Group
Shigeru Chiba

Stencil Domain: Multigrid



- Elliptic PDEs and systems thereof
- Discretization using finite differences or volumes
- Patch-based domains

Domain-Specific Stencil Language ExaSlang



Example: 2D Poisson with Damped Jacobi

- Layer 1 (continuous): Ops in Unicode or LaTeX symbols, partial derivative and Laplace predefined. Domains: unions of rectangles. Stencil weights calculated automatically.
- Layer 2 (discrete): Discretized functions are fields (data type, grid location), tied to a domain. Geometric information as ”virtual fields”, resolved to constants or field accesses.
- Layer 3 (Multigrid): Cycle and navigation through grid, access to grid levels with relative referencing. Grid transfer ops composed from default ops. Also more advanced types than Jacobi possible.
- Layer 4 (implementation): I/O and communication composition of solver parts, e.g., length of cycle.

```
/** Layer 1 **/

// domain – information is carried over from L1
Domain:  Ω = (0,1) × (0,1)      // alt.: \Omega = (0,1) \times (0,1)
        ∂Ω = (x^2 – 0.5 * y^2)  // alt.: \partial\Omega = 0
Equation: -Δ(u) = 0             // alt.: -\Laplace(u) = 0

// optional: specify mappings for ‘mnemonic’ names in subsequent layers to
// override defaults
Mapping: Ω -> global
         u -> Solution
```

ExaSlang 2

```
/** Layer 2 **/

// domain – information is carried over from L1
Domain global

// fields from L1
Field Solution@all with Real on Node of global = 0.0
Field RHS@all with Real on Node of global = 0.0

// boundary conditions from L1
Field Solution@finest on boundary = vf_boundaryCoord_x ** 2 - 0.5 *
vf_boundaryCoord_y ** 2
Field Solution@(all but finest) on boundary = 0.0

// (discretized) operators from L1
Operator Laplace from Stencil {
  [ 0,  0] =>  2.0 / ( vf_gridWidth_x ** 2 ) + 2.0 / ( vf_gridWidth_y ** 2 )
  [-1,  0] => -1.0 / ( vf_gridWidth_x ** 2 )
  [ 1,  0] => -1.0 / ( vf_gridWidth_x ** 2 )
  [ 0, -1] => -1.0 / ( vf_gridWidth_y ** 2 )
  [ 0,  1] => -1.0 / ( vf_gridWidth_y ** 2 )
}
```

ExaSlang 3

```
/** Layer 3 **/

/* fields and operators from L2 are carried over automatically */

// new fields
Field Residual      from Solution

// override inherited boundary conditions where applicable
override bc for Residual@finest with 0.0

// generate default inter-grid operators for node-based discretizations
Operator RestrictionStencil from default restriction on Node with 'linear'
Operator CorrectionStencil from default prolongation on Node with 'linear'

// create smoother function
Function Smoother@all {
  Val omega : Real = 0.8
  repeat 3 times {
    Solution =
      Solution + omega * diag_inv ( Laplace ) * ( RHS - Laplace * Solution )
  }
}
```

ExaSlang 3 (continued)

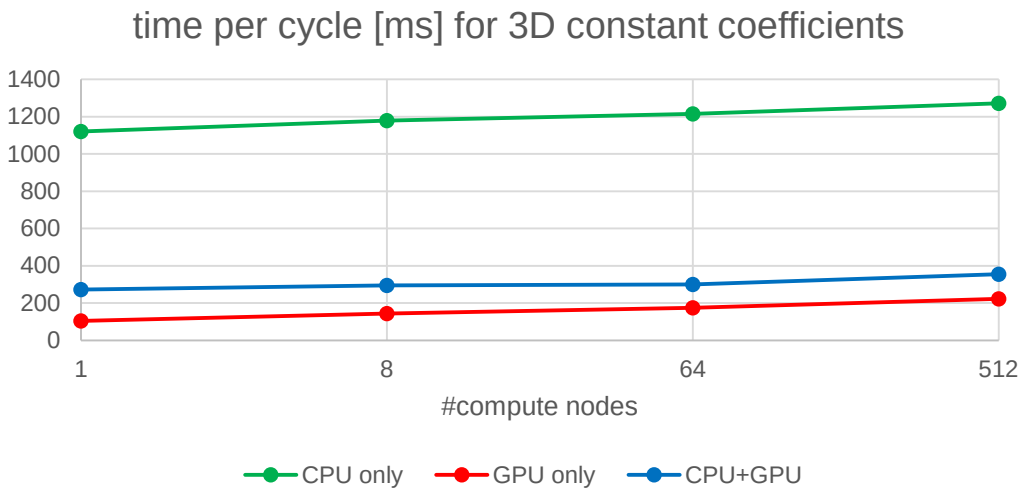
```
/** Layer 3 **/

// create v-cycle function
Function VCycle@(all but coarsest) {
  Smoother ( )

  Residual = RHS - ( Laplace * Solution )
  RHS@coarser = RestrictionStencil * Residual

  Solution@coarser = 0.0
  VCycle@coarser ( )

  Solution += CorrectionStencil * Solution@coarser

  Smoother ( )
}

Function VCycle@coarsest {
  /* CGS */
}
```

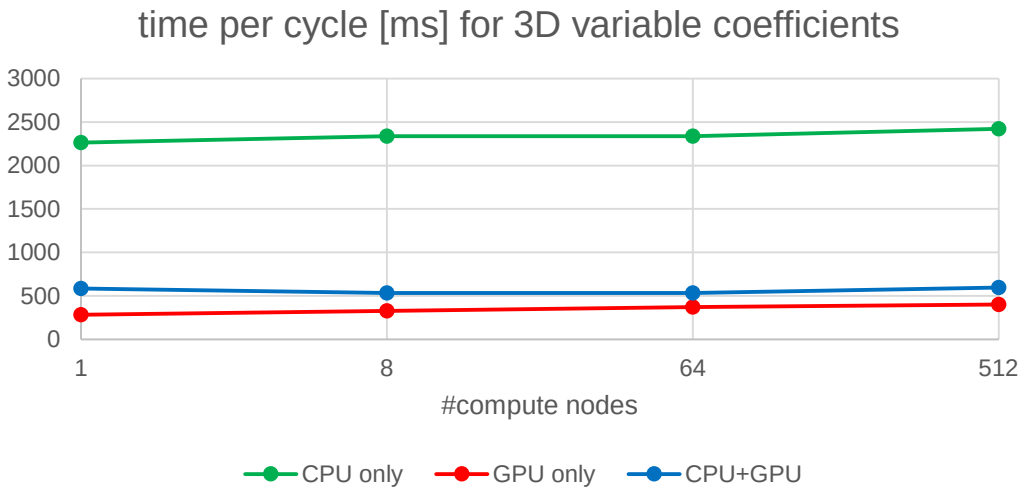
ExaSlang 4

```
/** Layer 4 **/

// create main function
Function Application ( ) : Unit {
  /* init code */

  Var resStart : Real = NormResidual@finest ( )
  Var curRes   : Real = resStart

  repeat until curRes < 1.0E-5 * resStart {
    VCycle@finest ( )
    curRes = NormResidual@finest ( )
  }

  /* de-init code */
}
```

Platform Variability

- (The same) ExaSlang input can be mapped to different hardware platforms
  - ”classical” CPUs
  - GPUs
  - FPGAs
  - ARM
  - and blends of these
- (Automatic) Parallelization using
  - MPI
  - OpenMP (on CPUs)
  - CUDA (on GPUs)
  - ...and combinations
- First results from PizDaint@CSCS
  - Weak scaling
  - 4 MPI threads per node
  - 512^3 unknowns per node





Problem Variability

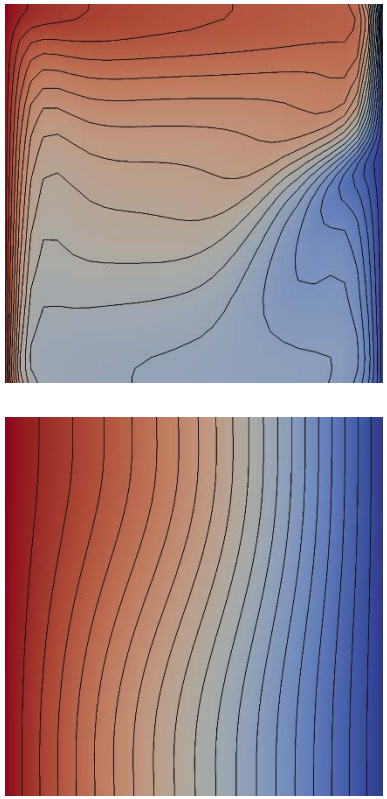
- Beyond Poisson's equation
  - Stokes
  - Navier-Stokes
  - Image processing
  - ...
- Each with specialized
  - Grids (may be non-uniform, non-axisparallel, staggered)
  - Discretizations and boundary treatment
  - Solvers (e.g. block smoothers, non-linear multigrid, etc.)

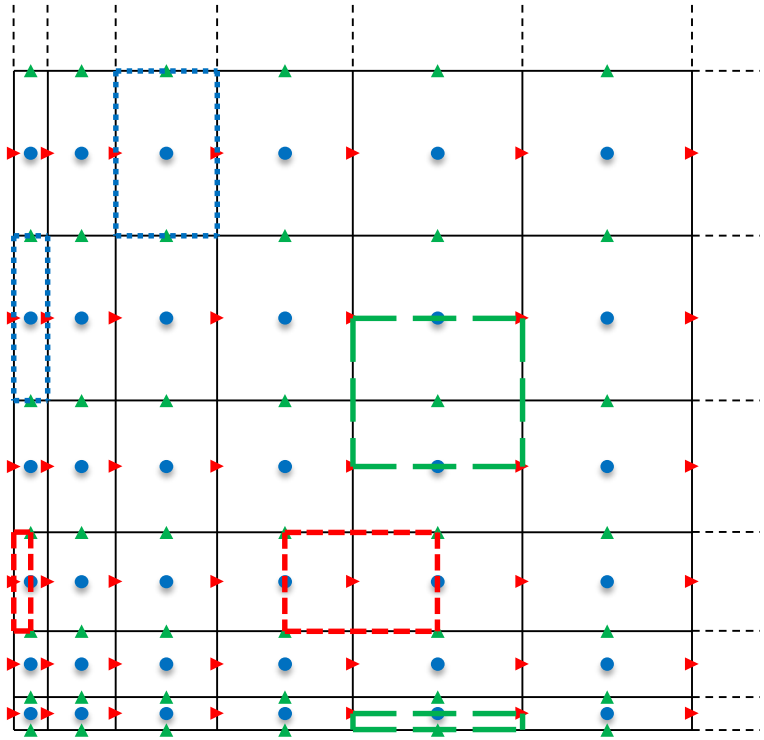
A Block Smoother for Stokes

```
loop over p {
 solve locally {
  u@[0, 0] => rhs_u@[0, 0] ==
    Laplace * u@[0, 0] + dxLeft  * p@[0, 0]
  u@[1, 0] => rhs_u@[1, 0] ==
    Laplace * u@[1, 0] + dxLeft  * p@[1, 0]
  …
  p@[0, 0] => rhs_p@[0, 0] ==
    dxRight * u@[0, 0] + dyRight * v@[0, 0]
 }
}
```

ExaSlang 4

- Fully Generated Real-World Application
  - Collaboration with Austria (U. Graz, G. Haase) and Chile (U. Santiago de Chile, D. Vasco)
  - 3D non-Newtonian and non-isothermal fluid flows
  - Finite volume discretization on non-uniform staggered grids



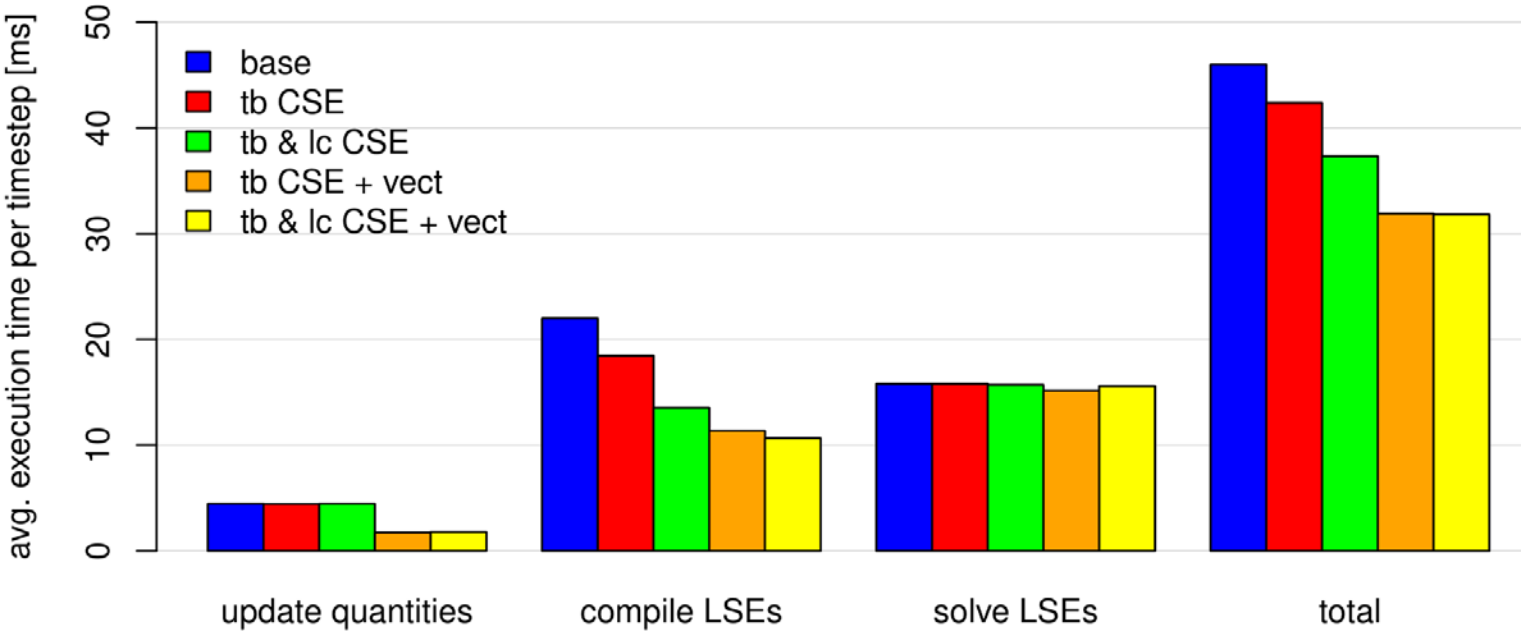


- values associated with the cell centers, e.g. p and θ
- values associated with the x-staggered grid, e.g. U
- values associated with the y-staggered grid, e.g. V
- control volumes associated with cell-centered values
- control volumes associated with x-staggered values
- control volumes associated with y-staggered values

Node-Level Optimizations: Compute-Bound

- Compute Optimizations
  - Arithmetic optimization (to maintain the numerical stability of stencil codes)
  - Common subexpression elimination (CSE)
    - text-based (tb) version: in a loop iteration
    - loop-carried (lc) version: between loop iterations
  - Vectorization for Intel x86 (incl. Xeon Phi), IBM BlueGene/Q and ARM NEON

Example: non-Newtonian flow simulation

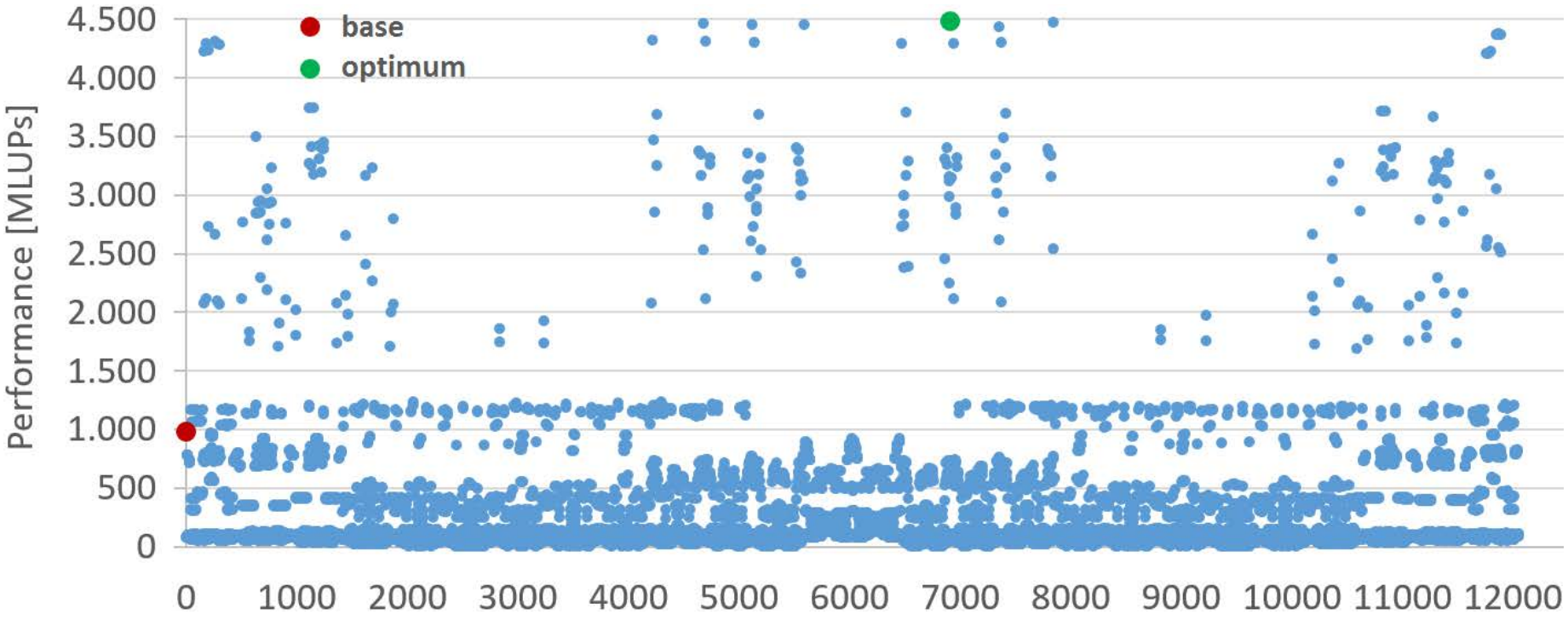


ExaStencils

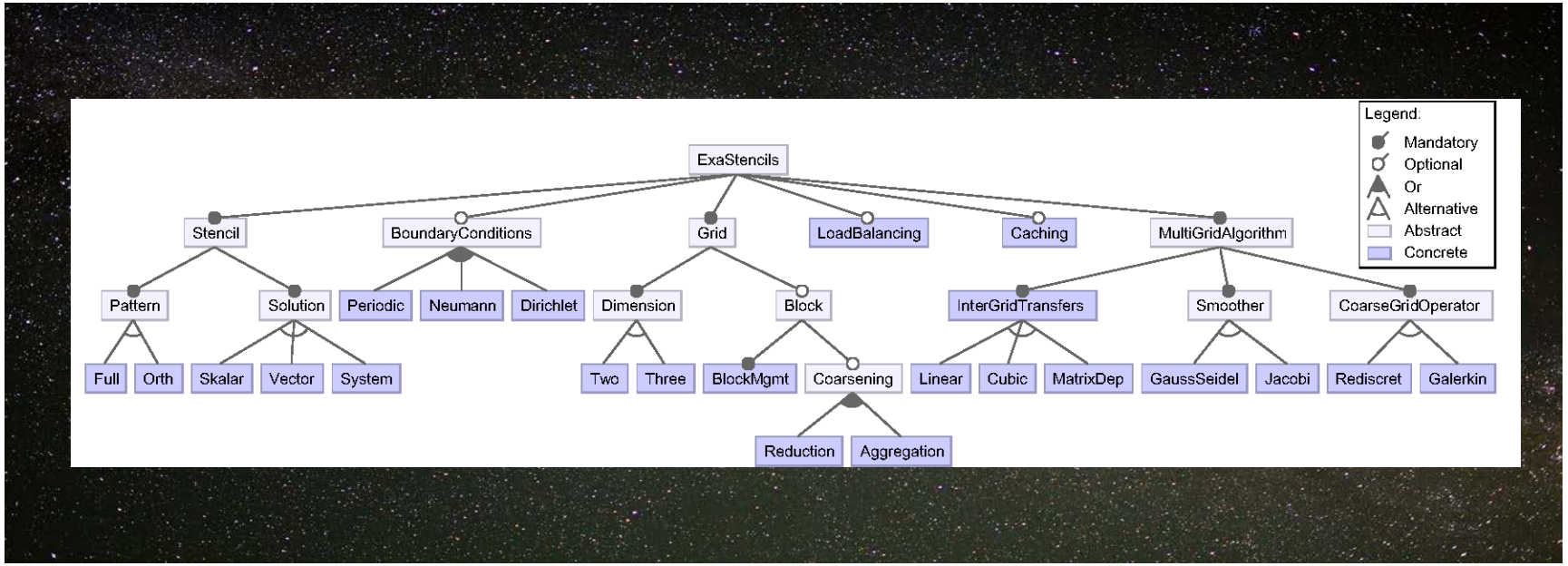
Node-Level Optimizations Memory-Bound

- Memory-Bandwidth Optimizations
  - Polyhedral transformations (temporal blocking)
    - perform search-space exploration
    - choose schedule heuristically

Example: exploration for 3D 7-point Jacobi



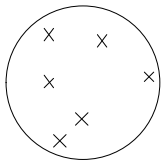
Taming the Variability of the Code-Generator



Sampling

Learning

Performance-Influence Model



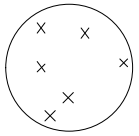
SPL Conqueror

189392.754871324 * root + -5341.1497460973 * cells + -
1097.39668559875 * post + 6852.58846993721 * GradientSolver +
752.542387648181 * BiCGSTABSolver + 1.88645994097499 * cells * post *
post + 547.236997276589 * GradientSolver * pre * pre + -
113.547148735501 * cells * GradientSolver + 0.706383967601375 * cells *
cells + 385.840181761216 * GradientSolver * post * post +
2.72707363219899 * cells * pre + -39.9822314861407 * BiCGSTABSolver *
pre * pre + -11.9701281275527 * cells * GradientSolver * pre * pre + -
8.2288112361045 * cells * GradientSolver * post * post +
112.152829956782 * post * SeqSOR

Identifying Performance-Optimal Configurations

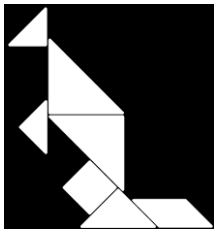
Can we identify performance-optimal parameter settings for different mathematical problems?

Sampling → Learning → Performance-Influence Model → Mixed-Integer Nonlinear Optimizer



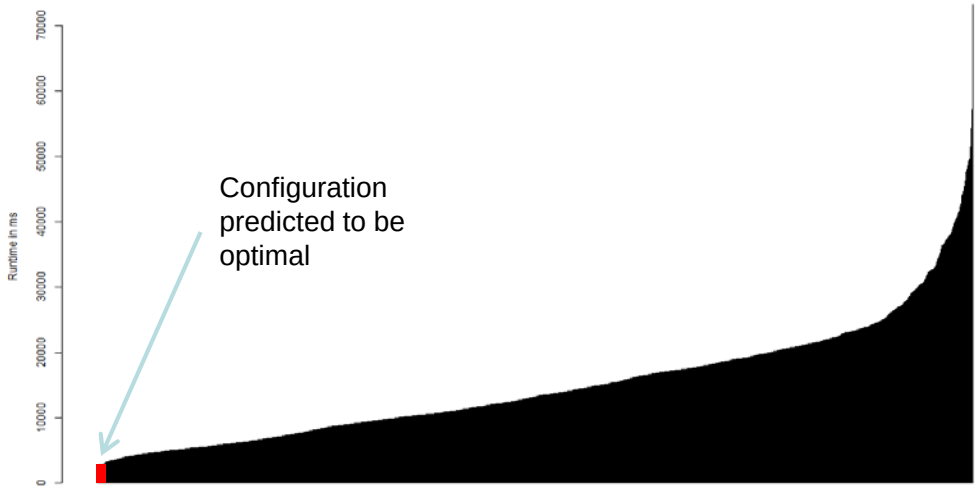
SPL Conqueror

189392.754871324 * root + -5341.1497460973 * cells + -
1097.39668559875 * post + 6852.58846993721 * GradientSolver +
752.542387648181 * BiCGSTABSolver + 1.88645994097499 * cells * post *
post + 547.236997276589 * GradientSolver * pre * pre + -
113.547148735501 * cells * GradientSolver + 0.706383967601375 * cells *
cells * cells + 385.840181761216 * GradientSolver * post * post +
2.72707363219899 * cells * pre + -39.9822314861407 * BiCGSTABSolver *
pre * pre + -11.9701281275527 * cells * GradientSolver * pre * pre + -
8.2288112361045 * cells * GradientSolver * post * post +
112.152829956782 * post * SeqSOR



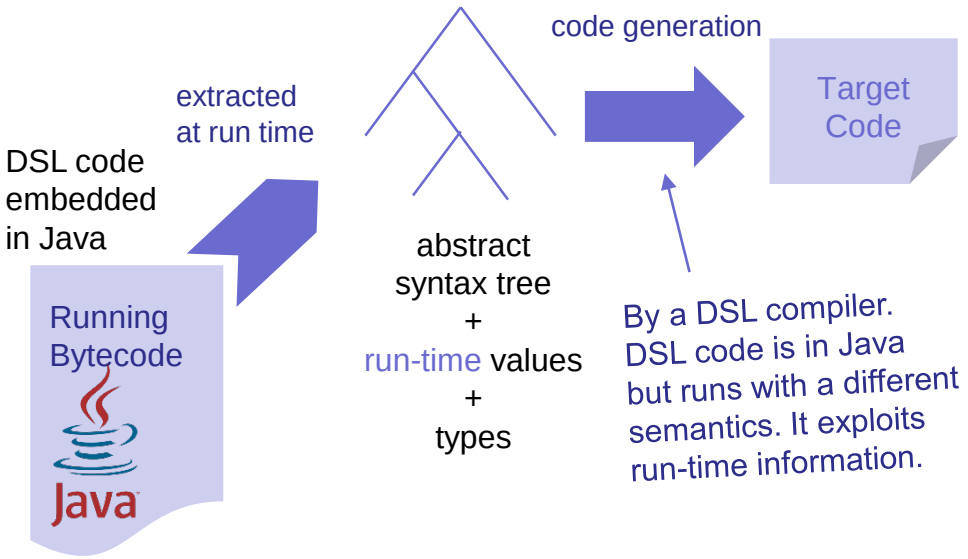
Performance-influence model for 2D Poisson with const. coeff.

2667.4 - 343.8 * log2(numNodes) * numPre - 926.5 * log2(numNodes) * numPost - 33.3 * log2(numNodes) - 317.5 * log2(numNodes) - 76.9 * mpi_CustomDatatypes + 1545.2 * numPre - 6.4E7 * numPost - 0.1 * log2(numNodes) * (64 - numNodes) - 0.3 * log2(numNodes) * (64 - numNodes) * log2(ranksPerNode) + 2.3 * log2(numNodes) * numPre * (64.0 - numNodes) - 0.4 * log2(numNodes) * numPre * (64.0 - numNodes) * log2(tileSize_x) - 0.5 * log2(numNodes) * numPre * (64.0 - numNodes) * log2(ranksPerNode) - 0.1 * log2(numNodes) * numPre * (64.0 - numNodes) * log2(ranksPerNode) * vectorize + 0.06 * numPost * (1.0E9 - tileSize_x) + 101.0 * log2(numNodes) * numPost * log2(numNodes) - 0.4 * log2(ranksPerNode) * (64.0 - ranksPerNode) + 1.0 * log2(ranksPerNode) * numNodes - 111.1 * log2(ranksPerNode) - 174.3 * log2(numNodes) + 0.006 * log2(ranksPerNode) * numNodes * (64 - numNodes) - 0.7 * log2(numNodes) * log2(ranksPerNode) - 0.03 * log2(numNodes) * (64.0 - ranksPerNode) - 0.03 * log2(numNodes) * (64 - numNodes) - 4.5 * log2(numNodes) - 2.7 * mpi_CustomDatatypes * ranksPerNode - 16.3 * mpi_CustomDatatypes * (4.0 - minLevel) - 86.7 * omp_parallelizeLoopOverDimensions + 2.2 * mpi_CustomDatatypes * ranksPerNode * log2(numNodes)



Japan: An Embedded DSL for Stencil Computing

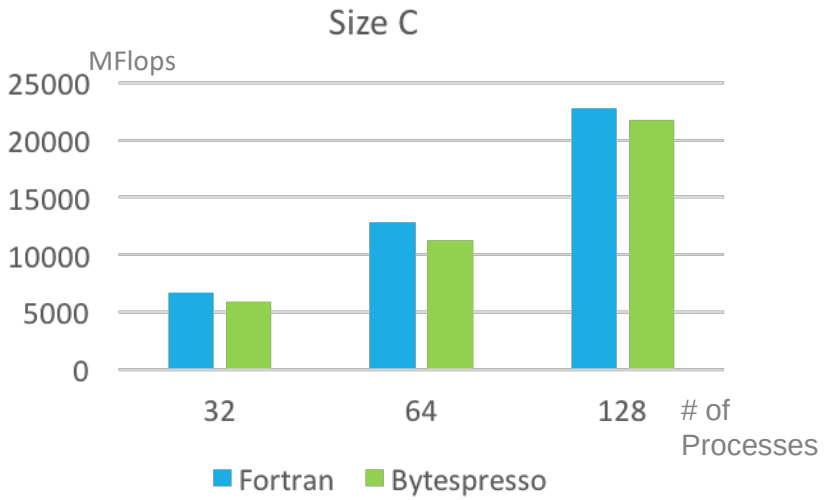
Bytespresso:
A new approach of *deep reification* for language embedding of domain-specific code.
DSL code embedded in Java is dynamically extracted and compiled into target (native) code.
In ExaStencils: a platform for an ExaSlang-like embedded DSL.




```
q.set(0.0); z.set(0.0); r.set(x); p.set(r);
double rho = r.norm();
for (int cgit = 1; cgit <= cgitmax; cgit++) {
  q.setToMult(a, p);
  double d = inner(p, q);
  double alpha = rho / d;
  z.setToAdd(z, alpha, p);
  r.setToAdd(r, -alpha, q);
  double rho0 = rho;
  rho = r.norm();
  double beta = rho / rho0;
  p.setToAdd(r, beta, p);
}
        :
```

Preliminary experiment:
NAS Parallel Benchmark 3.3 CG
with MPI Vector-Matrix DSL on Bytespresso

on the Fujitsu FX-10 supercomputer
$ mpifcc –Kfast



Case Study: Generation of Multigrid Solvers in SPIRAL

- SPIRAL: rule-driven optimizing code generator for digital signal-processing transforms, linear algebra
- Simple multigrid solver (cooperation with Franz Franchetti, CMU):
  square 2D Poisson, Richardson smoother, Dirichlet boundary
- Compared to ExaStencils, SPIRAL's domain is much narrower
- Special single-core optimization, but missing Kronecker optimization (fusion of sum of products)
- Algebraic view, comparable to ExaSlang 3 without knowledge of PDEs

$$\text{MGSolvePDE}_{n,\omega,r,m} \to [\text{I}_{n^2} \mid 0_{n^2}] \cdot \left(\prod_{i=0}^{m-1} \text{MGCycle}_{n,\omega,r} \right) \cdot \begin{bmatrix} 0_{n^2} \\ \text{I}_{n^2} \end{bmatrix}$$

$$\text{MGCycle}_{n,\omega,r} \to \text{CGC}_{n,\omega,r} \cdot \text{Richardson}_{n,\omega,r}$$

$$\text{CGC}_{n,\omega,r} \to \begin{bmatrix} \text{CoarseError}_{n,\omega,r} \\ 0_{n^2} \mid \text{I}_{n^2} \end{bmatrix}$$

$$\text{CoarseError}_{n,\omega,r} \to \text{Interpolate}_n \cdot \text{Scatter}_n \cdot \text{Solve}_{n,\omega,r} \cdot \text{Gather}_n \cdot \text{Residual}_n$$

$$\text{Interpolate}_n \to \text{Tridiag}_n \left(\sqrt{2}/2, \sqrt{2}, \sqrt{2}/2\right) \otimes \text{Tridiag}_n \left(\sqrt{2}/2, \sqrt{2}, \sqrt{2}/2\right)$$

$$\text{Scatter}_n \to \text{S}_{1,2}^{n \times (n-1)/2} \otimes \text{S}_{1,2}^{n \times (n-1)/2}$$

$$\text{Solve}_{n,\omega,r} \to \begin{cases} \frac{1}{4}\,\text{I}_1, & n = 1 \\ \left[\text{I}_{((n-1)/2)^2} \mid 0_{((n-1)/2)^2}\right] \cdot \text{MGCycle}_{(n-1)/2,\omega,r} \cdot \begin{bmatrix} 0_{((n-1)/2)^2} \\ \text{I}_{((n-1)/2)^2} \end{bmatrix}, & n > 1 \end{cases}$$

$$\text{Gather}_n \to \text{G}_{1,2}^{(n-1)/2 \times n} \otimes \text{G}_{1,2}^{(n-1)/2 \times n}$$

$$\text{Residual}_n \to [\text{Tridiag}_n(1, -2, 1) \otimes \text{I}_n + \text{I}_n \otimes \text{Tridiag}_n(1, -2, 1) \mid \text{I}_{n^2}]$$

$$\text{Richardson}_{n,\omega,r} \to \prod_{i=0}^{r-1} \begin{bmatrix} \text{ResidueLaplace}_{n,\omega} & \omega\,\text{I}_{n^2} \\ 0_{n^2} & \text{I}_{n^2} \end{bmatrix}$$

$$\text{ResidueLaplace}_{n,\omega} \to \text{Tridiag}_n(\omega, 0.5 - 2\omega, \omega) \otimes \text{I}_n + \text{I}_n \otimes \text{Tridiag}_n(\omega, 0.5 - 2\omega, \omega)$$

Networking

- Cooperation
  - SPIRAL (CMU / Imperial College / Terra-Neo)
  - RSDFT (Computational Materials Chemistry, Ruhr-Universität Bochum)
  - Non-Newtonian flows (University of Graz / Mechanical Engineering, University of Santiago, Chile)
  - Performance-influence models for Multigrid (Applied Mathematics, University of Zaragoza)
  - Stencil variablity for porous media (EXA-DUNE)

- Special Section in *Concurrency and Computation: Practice and Experience*
  - Six papers from Dagstuhl Seminar 15161 on Advanced Stencil-Code Engineering (April 2015)
  - Of these, five papers are new collaborations, four of them with participation of ExaStencils

Thanks for your attention!