ExaStencils

http://www.exastencils.org/

SPPEXA

Advanced Stencil-Code Generation

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

BERGISCHE UNIVERSITÄT WUPPERTAL

UNIVERSITÄT PASSAU
Fakultät für Informatik und Mathematik

Jürgen Teich, Frank Hannig

Ulrich Rüde, Harald Köstler

Matthias Bolten

Christian Lengauer, Armin Größlinger

Software Product-Line Group
Sven Apel

The Challenges of Exascale Computing

- *Exascale*
  - Goal for 2018–2020: 10^{18} FLOP/s
  - Present fastest supercomputers: 10^{16} FLOPS/s
- *Special Challenges*
  - massive parallelism on-chip: hundreds of cores
  - millions of cores in the cluster
  - heterogeneous architectures: different types of cores
  - power conservation
  - big data management
  - robust and performance-portable software

Two Alternative Approaches in SPPEXA

- *The Conservative Approach*
  - starting point: MPI, OpenMP and variants, OpenCL, Pthreads, ...
  - develop the above incrementally towards exascale by tuning their implementations adding some suitable features

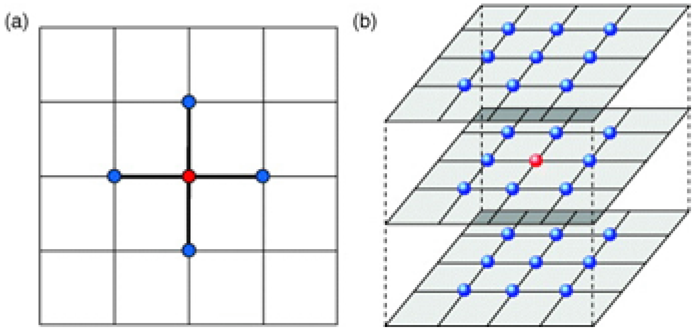
Two Alternative Approaches in SPPEXA

- *The Conservative Approach*
  - starting point: MPI, OpenMP and variants, OpenCL, Pthreads, ...
  - develop the above incrementally towards exascale by tuning their implementations adding some suitable features

- *The Radical Approach*
  - more various, abstract and domain-specific programming languages
  - powerful optimizing translation technology for small domains
  - more flexible and intelligent run-time systems
  - flexibly constructable and usable software development tools
  - more performance-portable implementations of application software

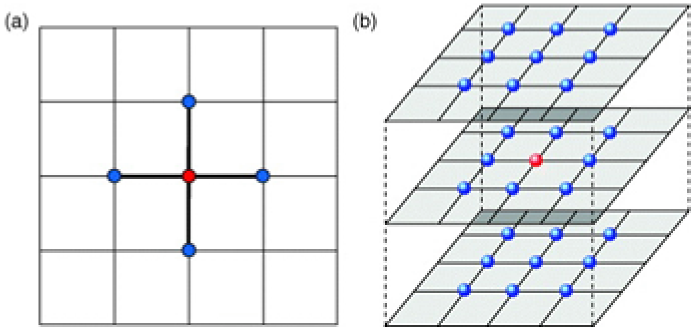
Project Charter

- *Domain: Stencil codes*
  - widely used
  - well delineated
  - massively parallel



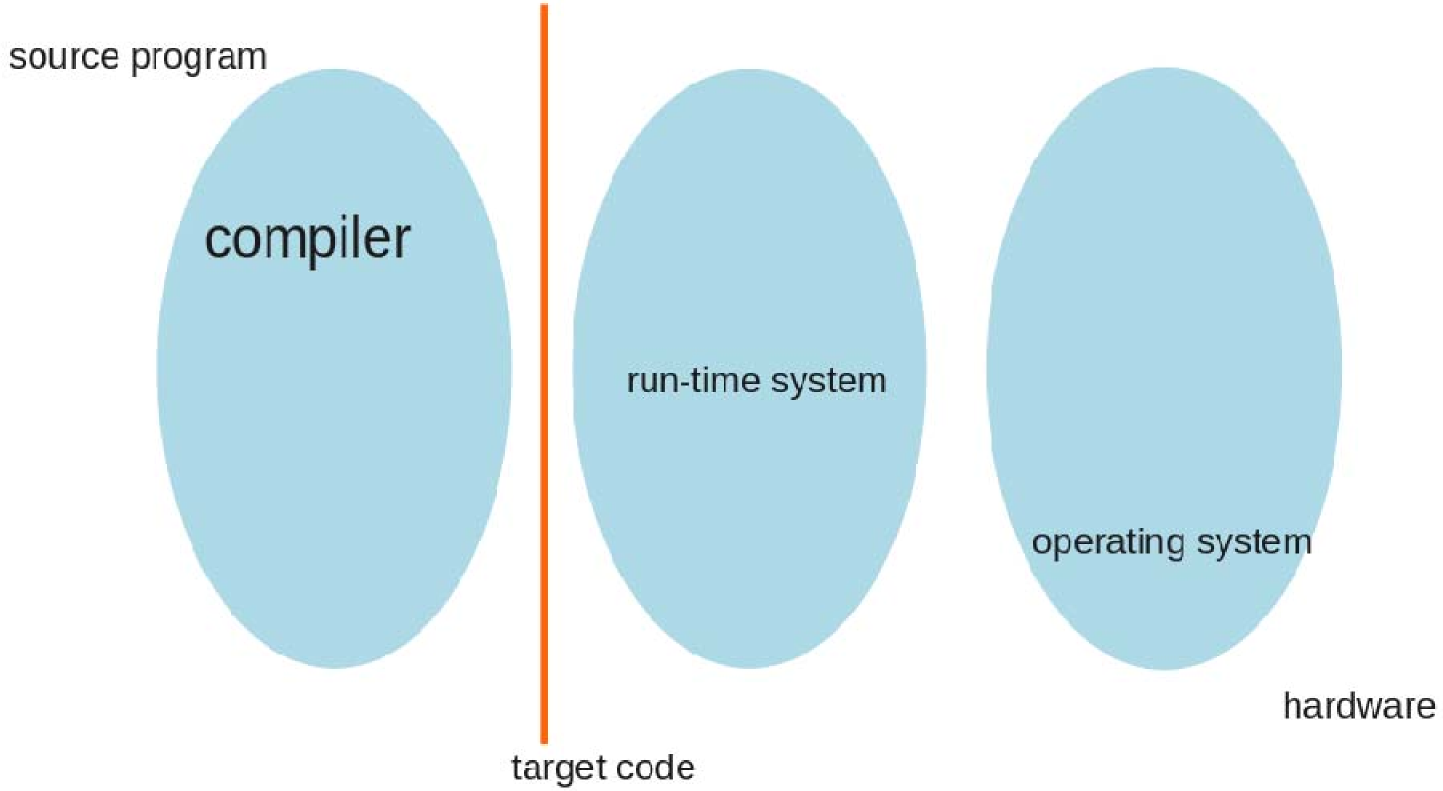
Project Charter

- *Domain: Stencil codes*
  - widely used
  - well delineated
  - massively parallel

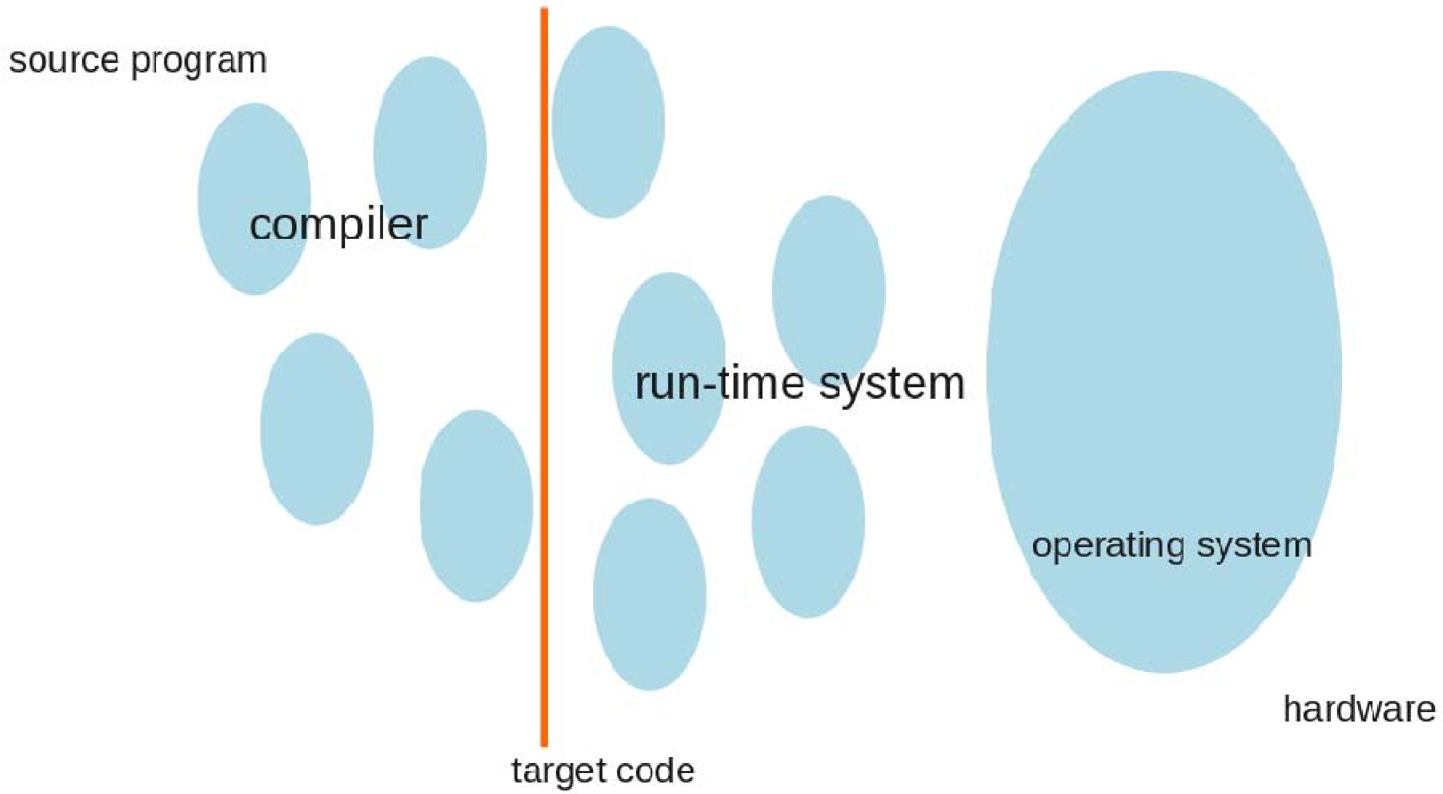


- *Approach: Co-design of application, algorithm and software*
  - domain-specific optimization *at every step of the design*
  - use of a variety of languages at various levels of abstraction
  - high automation via customized tool support
  - inspiration: project SPIRAL (optimizing compiler for linear transforms)

The Past: Power to the Compiler



The Future: More Power to the Run-time System



Work Areas

A: Algorithmic engineering

B: Domain-specific representation and modeling

C: Domain-specific optimization and generation

D: Polyhedral optimization and code generation

E: Platform-specific code optimization and generation

Work Areas

A: Algorithmic engineering

B: Domain-specific representation and modeling

C: Domain-specific optimization and generation

D: Polyhedral optimization and code generation

E: Platform-specific code optimization and generation

- Choice of PDE operators
- Choice and tuning of solvers
- Focus on multigrid
- Identification of variants (features)
  - algorithm
  - architecture

$$\phi_{\mathrm{MGM}}^{(k)}(\mathbf{x}_k, \mathbf{b}_k) = (\phi_{\mathrm{S}}^{(k)})^{\nu_2}((\phi_{\mathrm{S}}^{(k)})^{\nu_1}(\mathbf{x}_k, \mathbf{b}_k) + P_k((\phi_{\mathrm{MGM}}^{(k-1)})^{\gamma}(\mathbf{0}, R_k(\mathbf{b}_k - A_k\mathbf{x}_k))), \mathbf{b}_k)$$

Work Areas

A: Algorithmic engineering

B: Domain-specific representation and modeling

C: Domain-specific optimization and generation

D: Polyhedral optimization and code generation

E: Platform-specific code optimization and generation

- Most abstract executable representation
  - algorithmic features
  - architectural features
- Application view

```
Domain<2D, double> Dom = load_from_file(...);
Stencil<2D, double> Sten = {
    { 0, −1,  0},
    {−1,  4, −1},
    { 0, −1,  0}};
Restriction<2D, double> Rest = { ... };
Interpolation<2D, double> Inter = { ... };
MultiGridSolver Solver(Dom, Sten, Rest, Inter);
Solver.loadHardwareTopology("cluster.xml");
Solver.setSmoother(Jacobi);
```

Work Areas

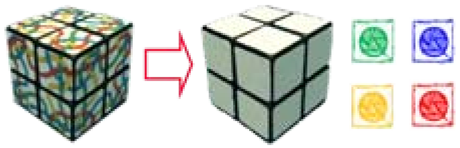
A: Algorithmic engineering

B: Domain-specific representation and modeling

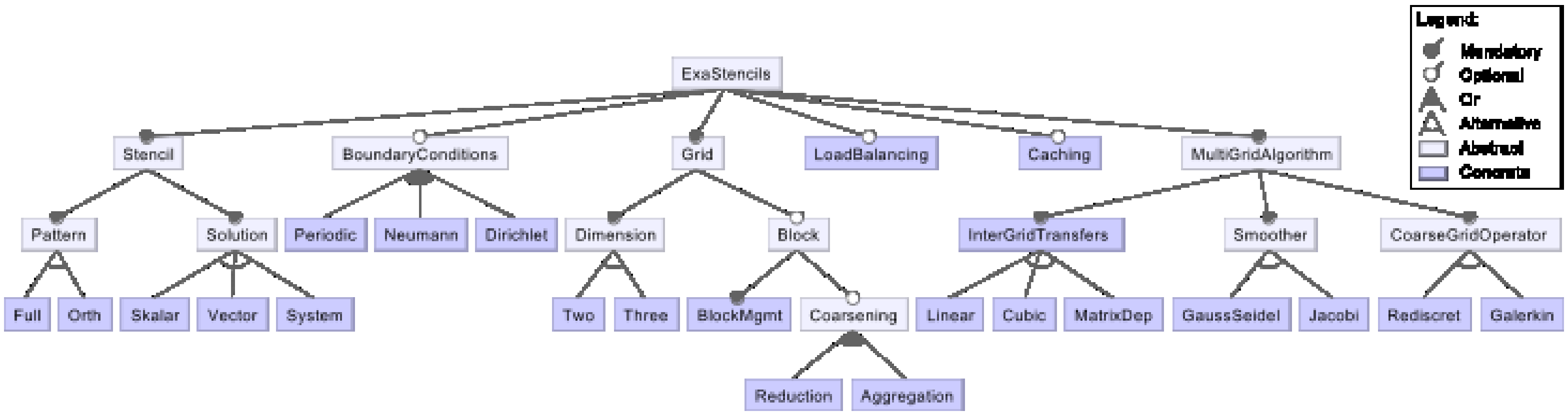
C: Domain-specific optimization and generation

D: Polyhedral optimization and code generation

E: Platform-specific code optimization and generation



- Feature orientation
- Exploitation of program similarity
- Product-specific optimization
- Automatic product weaving



Work Areas

A: Algorithmic engineering

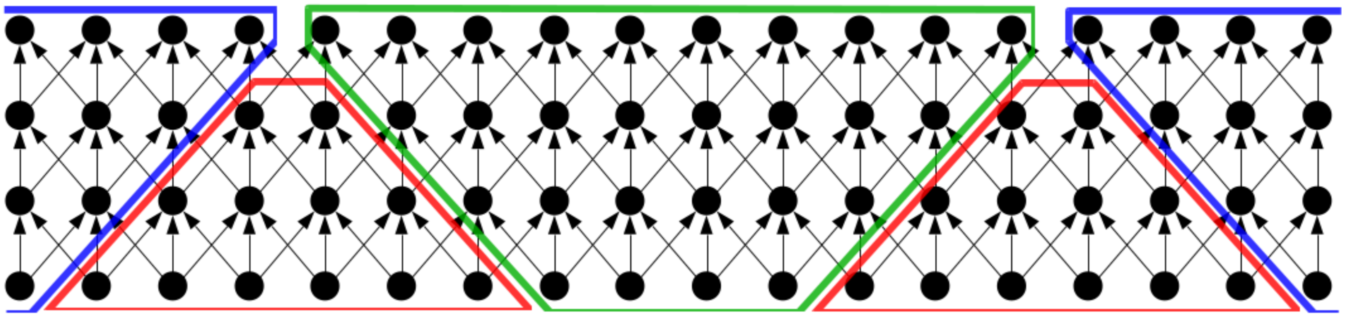
B: Domain-specific representation and modeling

C: Domain-specific optimization and generation

D: Polyhedral optimization and code generation

E: Platform-specific code optimization and generation

- Product-specific optimization
- Platform-specific optimization
  intra-node
  inter-node



Work Areas

A: Algorithmic engineering

B: Domain-specific representation and modeling

C: Domain-specific optimization and generation

D: Polyhedral optimization and code generation

E: Platform-specific code optimization and generation

- Addresses heterogeneity
- Methods considered:
  - Autotuning
  - Machine learning
  - Hardware/software codesign

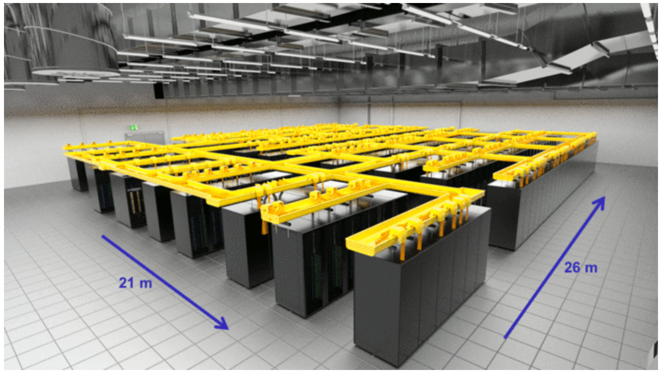
```
void GaussSeidel_rb(int lev, Array <double> *Sol, Array<double> *RHS) {
  int offset;

  #pragma omp parallel for private(offset)
  for (int i=1; i<Sol[lev].nrows()-1; i++) {
    offset = (i % 2 == 0 ? 2 : 1);
    for (int j=offset; j<Sol[lev].ncols()-1; j+=2) {
      Sol[lev] (i, j) = double(0.25)*(RHS[lev] (i, j) + Sol[lev] (i+1, j)
          + Sol[lev] (i-1, j) + Sol[lev] (i, j+1) + Sol[lev] (i, j-1));
    }
  }

  #pragma omp parallel for private(offset)
  for ( int i=1; i<Sol[lev].nrows()-1; i++ ) {
    offset = (i % 2 == 0 ? 1 : 2);
    for (int j=offset; j<Sol[lev].ncols()-1; j+=2) {
      Sol[lev] (i, j) = double(0.25)*(RHS[lev] (i, j) + Sol[lev] (i+1, j)
          + Sol[lev] (i-1, j) + Sol[lev] (i, j+1) + Sol[lev] (i, j-1));
    }
  }
}
```

Supercomputers

- *SuperMUC*
  - Leibniz Computation Centre
  - 6th on TOP500, 11/2012



- JuQUEEN
  - Jülich Research Centre
  - 5th on TOP500, 11/2012

- *TSUBAME 2.0*
  - Tokyo Institute of Technology
  - 17th on TOP500, 11/2012