Towards Automatic Generation of Energy-aware Efficient Geometric Multigrid Solvers

Lisa Claus[1], Alexander Grebhahn[2], Stefan Kronawitter[2], Sebastian Kuckuk[3], Hannah Rittich[1], Christian Schmitt[3]
[1]University of Wuppertal; [2]University of Passau; [3]Friedrich-Alexander University Erlangen-Nürnberg (FAU)

ExaStencils

Programming
Prof. Christian Lengauer, Ph.D.
(Coordinator)

Software Engineering
Prof. Dr.-Ing. Sven Apel

High Performance Computing/SE
Prof. Dr. rer. nat. Matthias Bolten

Metaprogramming
Prof. Shigeru Chiba

Simulation Science
Prof. Dr. rer. nat. Ulrich Rüde
PD Dr.-Ing. Harald Köstler

Hardware/Software Co-Design
Prof. Dr.-Ing. Jürgen Teich
Dr.-Ing. Frank Hannig

Results and Goals

Overall charter

A tool-assisted fully automated domain-specific design approach for energy-aware stencil codes

Results Achieved:

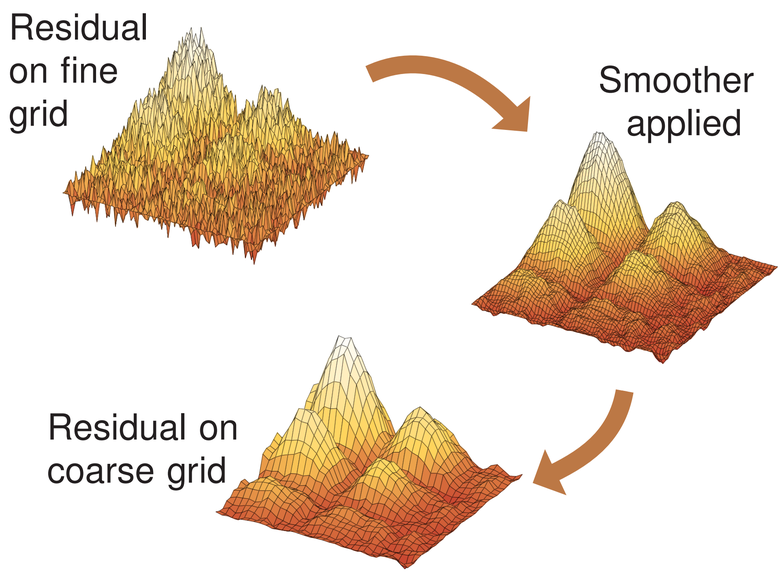
- Domain-specific language *ExaSlang* and transformation framework
- Scalability up to full breadth of the JUQUEEN supercomputer
- Low-level optimizations for BlueGene/Q and Intel CPUs
- Performance forecast via product-line sampling and machine learning

Further Goals:

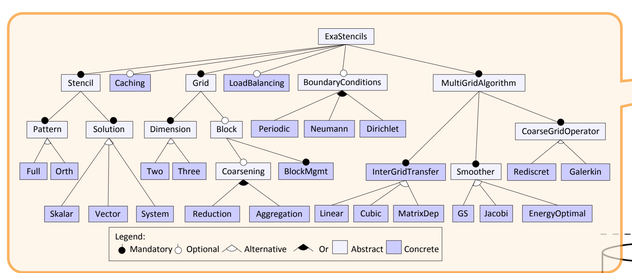
- Exascale technology for a broader range of PDEs
- Energy efficiency at all layers of abstraction
- Performance/power trade-offs on exascale machines
- Exploitation of heterogenous and hybrid architectures

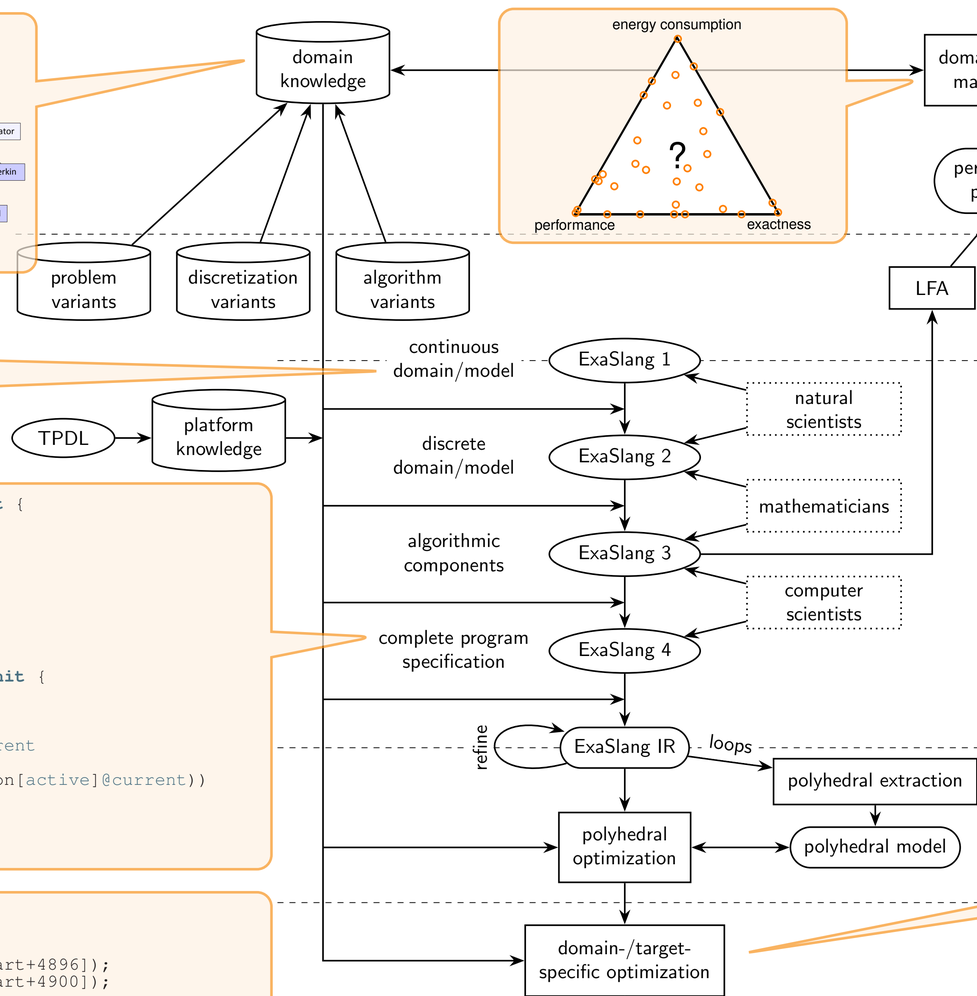
Geometric Multigrid

1. Pre-smoothing
2. Calculation of residual
3. Restriction
4. Recursive call(s) or solve (at coarsest level)
5. Prolongation
6. Correction
7. Post-smoothing



Design Flow

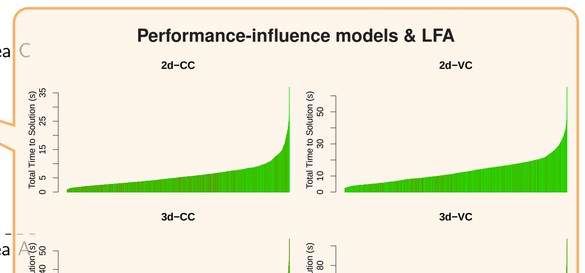




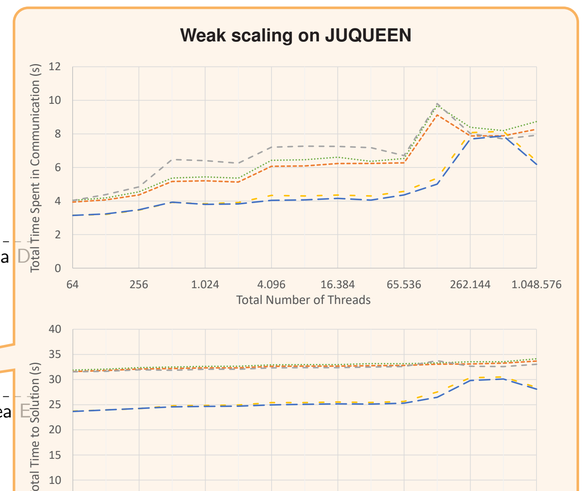
$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\frac{\nabla P}{\rho} + \nu \nabla^2 u$$

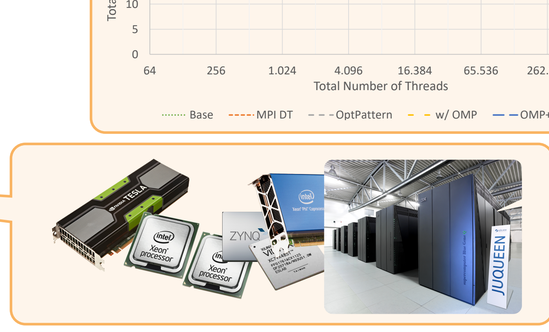
```
Function WCycle@((coarsest + 1) to finest) () : Unit {
  repeat 3 times { Smoother@current () }
  UpResidual@current ()
  Restriction@current ()
  SetSolution@coarser (0)
  repeat 2 times { WCycle@coarser () }
  Correction@current ()
  repeat 3 times { Smoother@current () }
}

Function Smoother@((coarsest + 1) to finest) () : Unit {
  communicate ghost of Solution[active]@current
  loop over fragments {
    loop over Solution@current {
      Solution[next]@current = Solution[active]@current
        + (omega * inverse(diag(Laplace@current))
          * (RHS@current - Laplace@current * Solution[active]@current))
    }
    advance Solution@current
  }
}
```

```
void Smoother_6() { ...
  vector4double vec00 = vec_splats(0.2);
  vector4double vec02 = vec_splats(0.1333);
  vector4double vec01 = vec_lda(0, &solution_old[start+4896]);
  vector4double vec08 = vec_lda(0, &solution_old[start+4900]);
  for (int i5 = start; i5 < end; i5 += 4) {
    vector4double vec03 = vec_lda(0, &rhs[i5]);
    vector4double vec10 = vec_lda(0, &solution_old[i5+76]);
    vector4double vec04 = vec_lda(0, &solution_old[i5+4828]);
    vector4double vec06 = vec01;
    vec01 = vec08;
    vec08 = vec_lda(0, &solution_old[i5+4904]);
    vector4double vec09 = vec_lda(0, &solution_old[i5+4972]);
    vector4double vec11 = vec_lda(0, &solution_old[i5+9724]);
    vector4double vec07 = vec_sldw(vec01, vec08, 1);
    vector4double vec05 = vec_sldw(vec06, vec01, 3);
    vector4double vec12 = vec_madd(vec02, vec_add(vec_add(vec11, vec_add(
   vec03, vec04)), vec_add(vec_add(vec05, vec07), vec_add(vec09, vec10))),
     vec_mul(vec00, vec01));
    vec_sta(vec12, 0, &solution_new[i5+4900]);
  } ...
}
```



case study	min	mean	optimum	speedup
2d-CC	0.854	5.832	0.599	1.42
2d-VC	2.070	12.817	1.488	1.39
3d-CC	1.460	9.431	1.341	1.08
3d-VC	3.673	17.440	2.966	1.24





Work Areas

A Algorithmic Engineering
- A.1 Extension to broader class of PDEs
- A.2 Convergence prediction
- A.3 Validation of the results computed using the generated code

B Domain-Specific Representation and Modeling
- B.1 Representation of power models
- B.2 Abstract language layers

C Domain-Specific Optimization and Generation
- C.1 Combining binary and numeric options
- C.2 Domain-knowledge integration and validation
- C.3 Integration of a feedback loop
- C.4 Energy efficiency as an objective

D Polyhedral Optimization and Code Generation
- D.1 Evaluating different objective functions
- D.2 Extension of color splitting

E Target-Specific Code Optimization and Generation
- E.1 Code generation for heterogeneous target platforms
- E.2 Generation of energy-aware code
- E.3 Performance prediction

F Evaluation and Demonstration
- F.1 Proof of concept / demonstrator
- F.2 Proof of productivity
- F.3 Proof of flexibility
- F.4 Proof of exascale performance and energy awareness

G Rapid Prototyping of Stencil DSLs
- G.1 An internal DSL for ExaStencils
- G.2 Case studies

References & Acknowledgements

- Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. “Performance-Influence Models for Highly Configurable Systems”. In: *Proc. European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM Press, 2015, pp. 284–294
- Stefan Kronawitter and Christian Lengauer. *Optimizations Applied by the ExaStencils Code Generator*. Tech. rep. MIP-1502. Faculty of Informatics and Mathematics, University of Passau, 2015
- Christian Schmitt, Sebastian Kuckuk, Frank Hannig, Harald Köstler, and Jürgen Teich. “ExaSlang: A Domain-Specific Language for Highly Scalable Multigrid Solvers”. In: *Proc. 4th Int. Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing (WOLFHPC)*. ACM, 2014, pp. 42–51
- Sebastian Kuckuk, Björn Gmeiner, Harald Köstler, and Ulrich Rüde. “A Generic Prototype to Benchmark Algorithms and Data Structures for Hierarchical Hybrid Grids”. In: *Proc. Int. Conf. on Parallel Computing (ParCo)*. IOS Press, 2013, pp. 813–822

SPPEXA

DFG Deutsche Forschungsgemeinschaft

ExaStencils is funded by the German Research Foundation (DFG) as part of the Priority Program 1648 (Software for Exascale Computing)