News from SPPEXA Project ExaStencils

Christian Lengauer

University of Passau

IFIP WG 2.11 Meeting 2014, Pittsburgh

ExaStencils

DFG Priority Programme SPPEXA

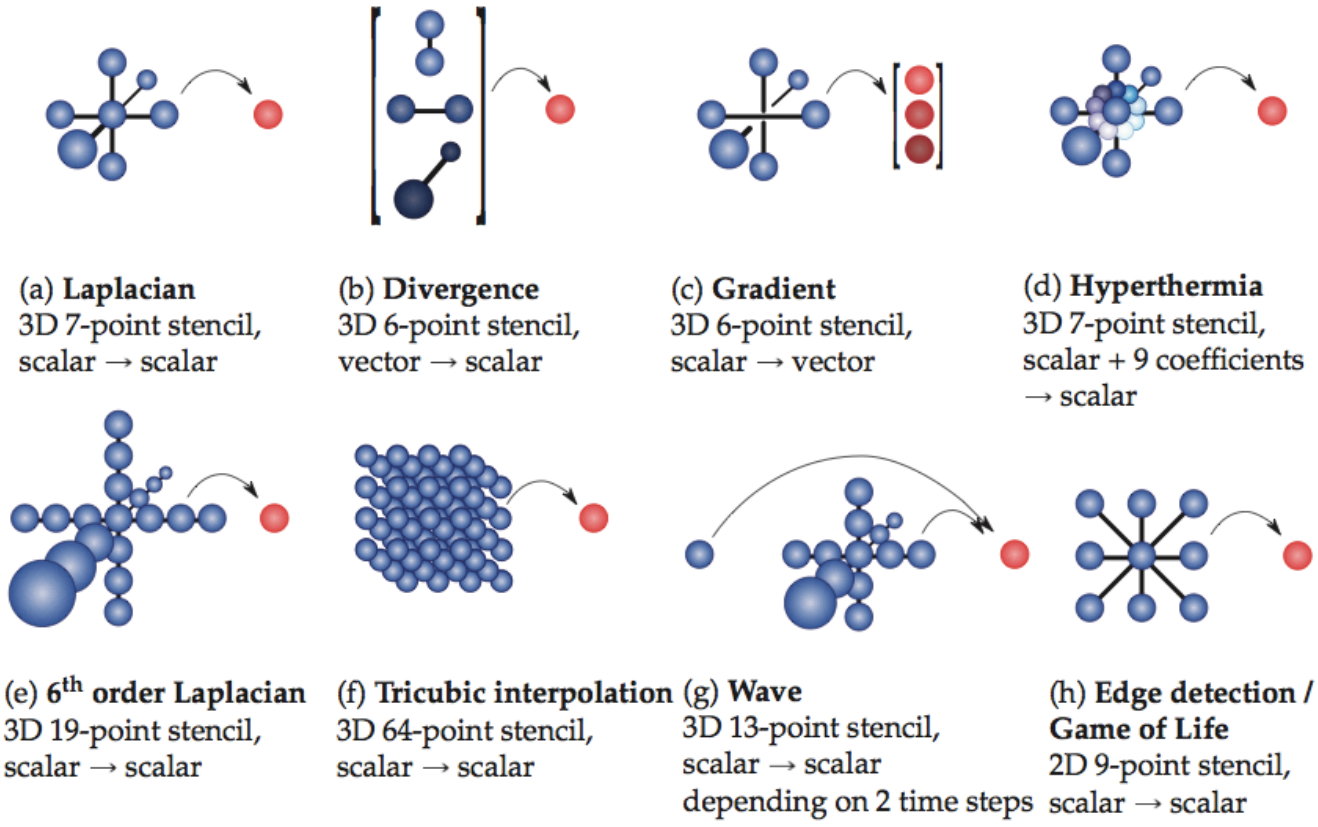
Goal: Reliable and Manageable Exascale Performance

- Exascale = 10^{18} FLOP/s (expected by 2020)
- Present highest performance = 10^{16} FLOP/s (TOP500 list)
- Distance yet to cover = Factor of 100

SPPEXA Approach

- Conservative Tier: extend existing software technology (APIs, middleware, applications)
- Radical Tier: develop new software technology
  $\Longrightarrow$ ExaStencils

Stencil Codes



(a) **Laplacian**
3D 7-point stencil,
scalar → scalar

(b) **Divergence**
3D 6-point stencil,
vector → scalar

(c) **Gradient**
3D 6-point stencil,
scalar → vector

(d) **Hyperthermia**
3D 7-point stencil,
scalar + 9 coefficients
→ scalar

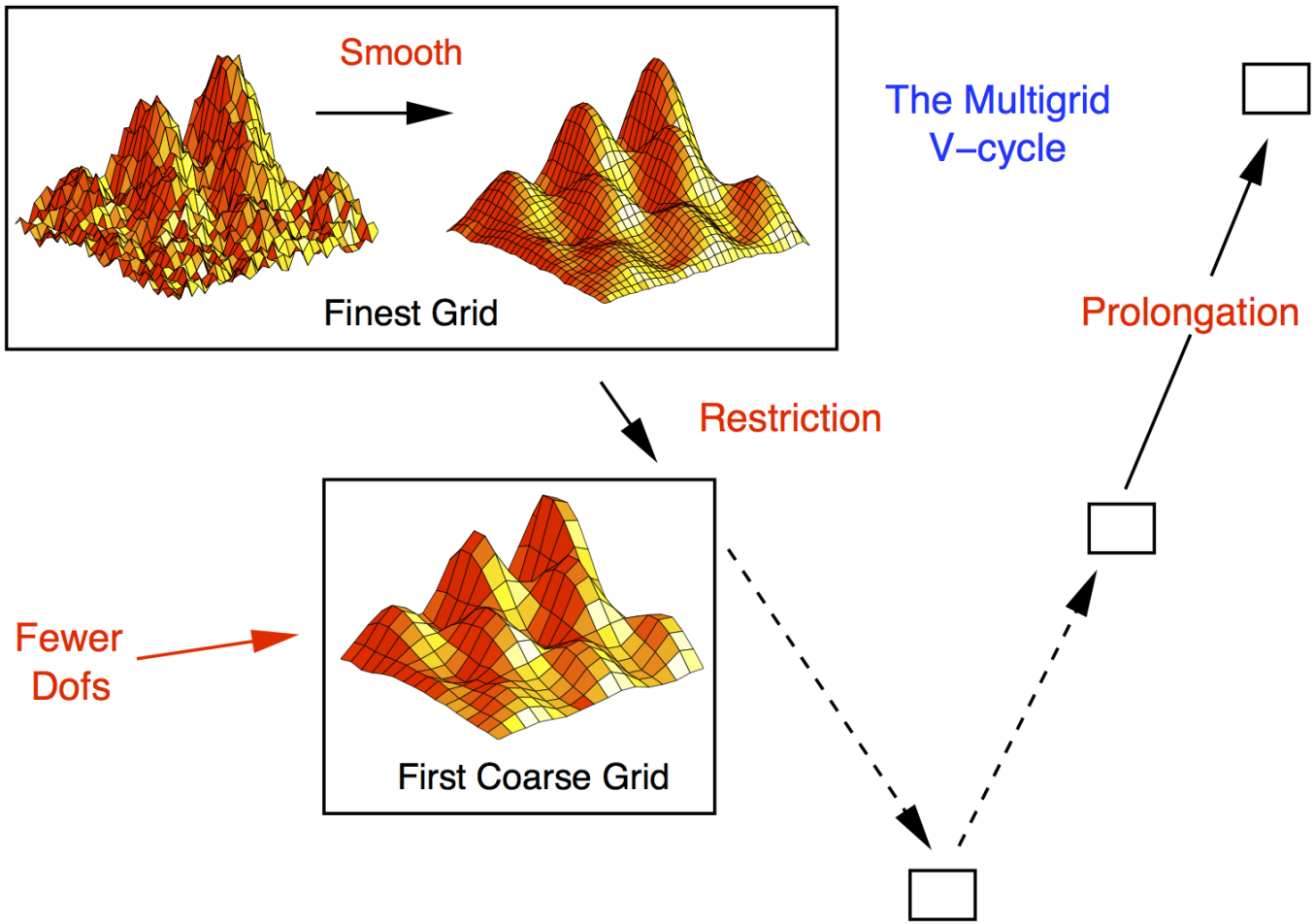
(e) **6th order Laplacian**
3D 19-point stencil,
scalar → scalar

(f) **Tricubic interpolation**
3D 64-point stencil,
scalar → scalar

(g) **Wave**
3D 13-point stencil,
scalar → scalar
depending on 2 time steps

(h) **Edge detection / Game of Life**
2D 9-point stencil,
scalar → scalar

Multigrid



©Robert Falgout, Lawrence Livermore National Laboratory, USA

ExaStencils

Goal: Optimization of Stencil Codes for Exascale Performance

- Covering a large domain of stencil codes
- Stencil- and platform-specific
- Applied repeatedly at successive levels of abstraction
- Exploiting similarities between codes
- Automatic code generation

Consortium

- Passau: software product lines, polyhedral optimization
- Erlangen: simulation science, hardware-software codesign
- Wuppertal: stencil mathematics

Code Generator

Preliminary Common Code Generator (completed)

- Covering a large variety of stencil codes (*that's new!*)
- Completely unoptimized
- Within a factor of 10 of hand-optimized codes

Cleaned-Up Version (in progress)

- In the spirit of Spiral
- Four increasingly concrete DSLs
- Hosted by Scala
- Feature orientation + polyhedral loop optimization

Feature Orientation

First Automatic Prediction of Efficient Configurations

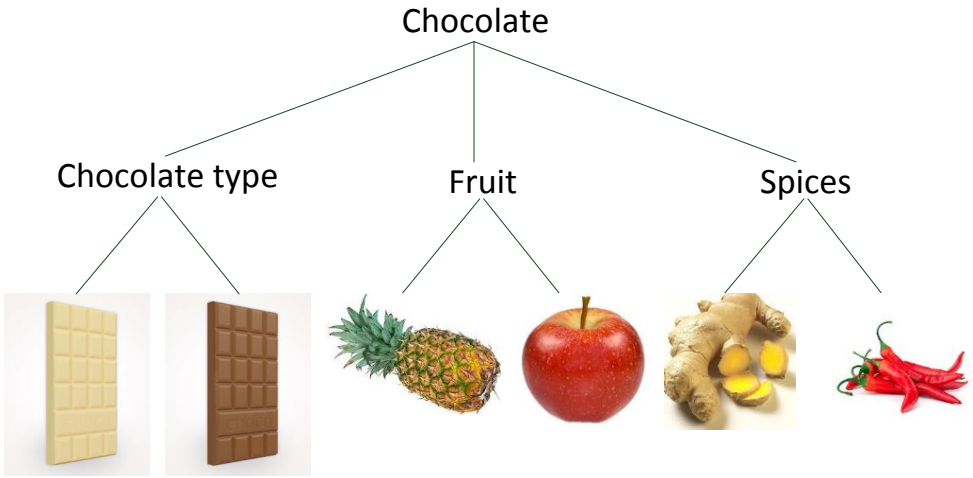
- Empirical, feature-oriented approach
- Identifies feature combinations with good performance
- Few experiments yield good results

Optimizing Performance of Stencil Code with SPL Conqueror

Alexander Grebhahn[1], Norbert Siegmund[1], Sven Apel[1],
Sebastian Kuckuk[2], Christian Schmitt[2], Harald Köstler[2]

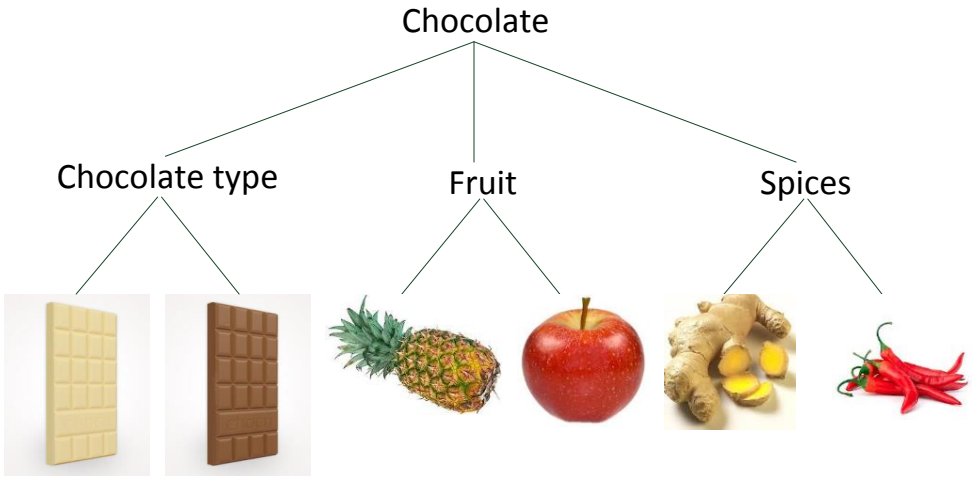
[1]University of Passau, [2]University of Erlangen-Nuremberg

Customizable Programs

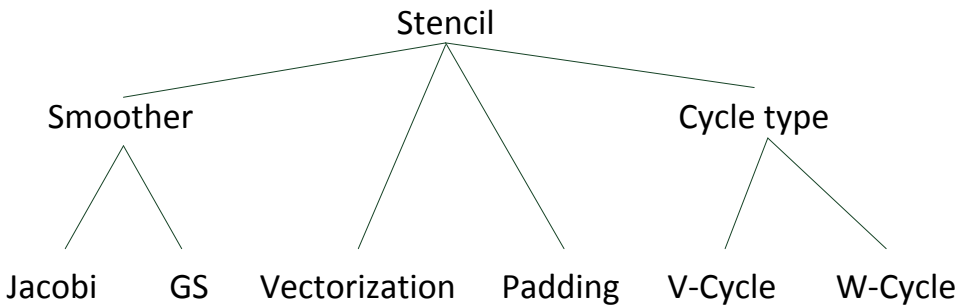


Which chocolate tastes best?

Customizable Programs



Which chocolate tastes best?

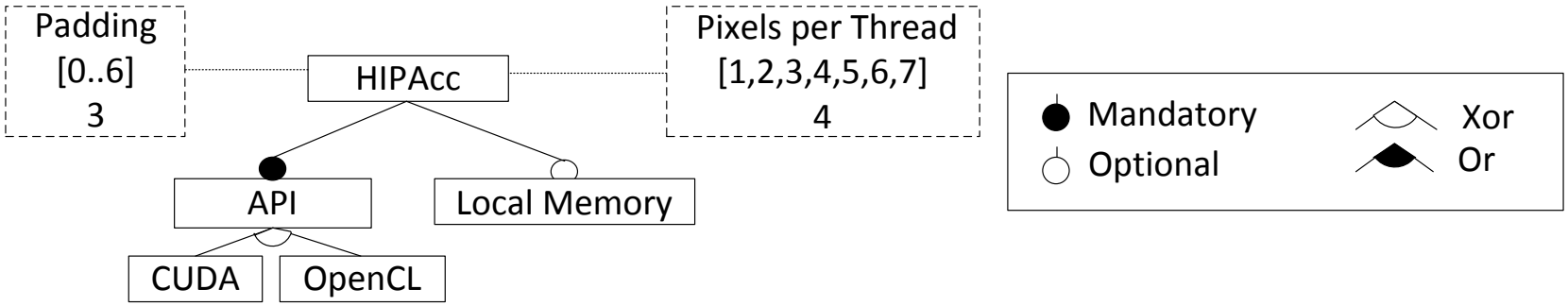


Which stencil configuration performs best?

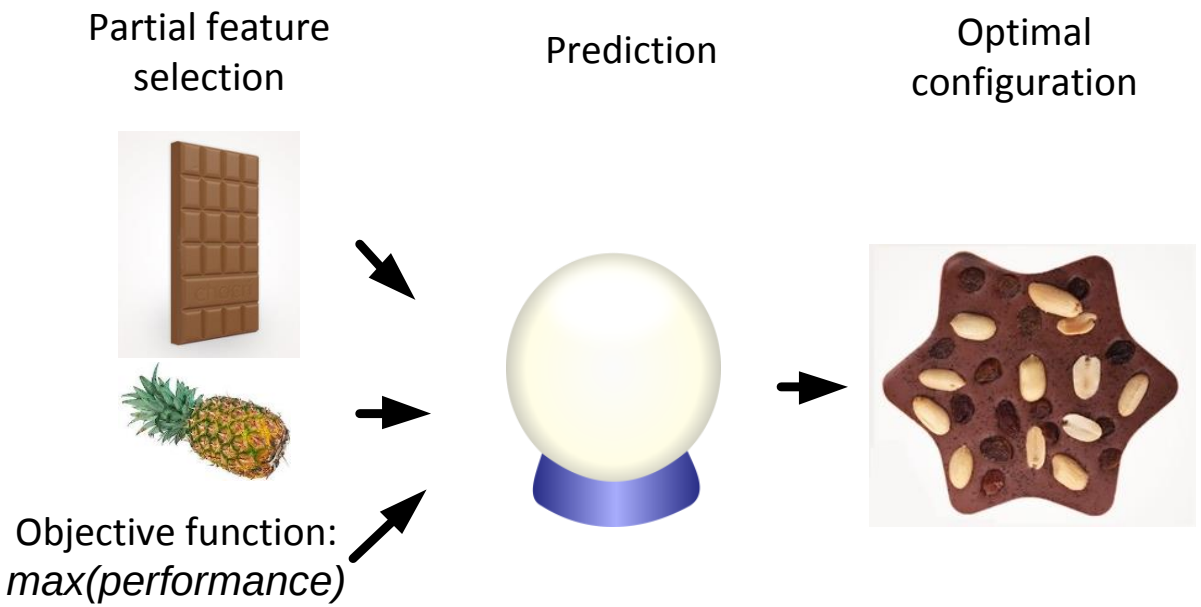
GS Padding W-Cycle	Jacobi Padding V-Cycle

Customizable Programs

Feature Models: [Kang et al., 1990]



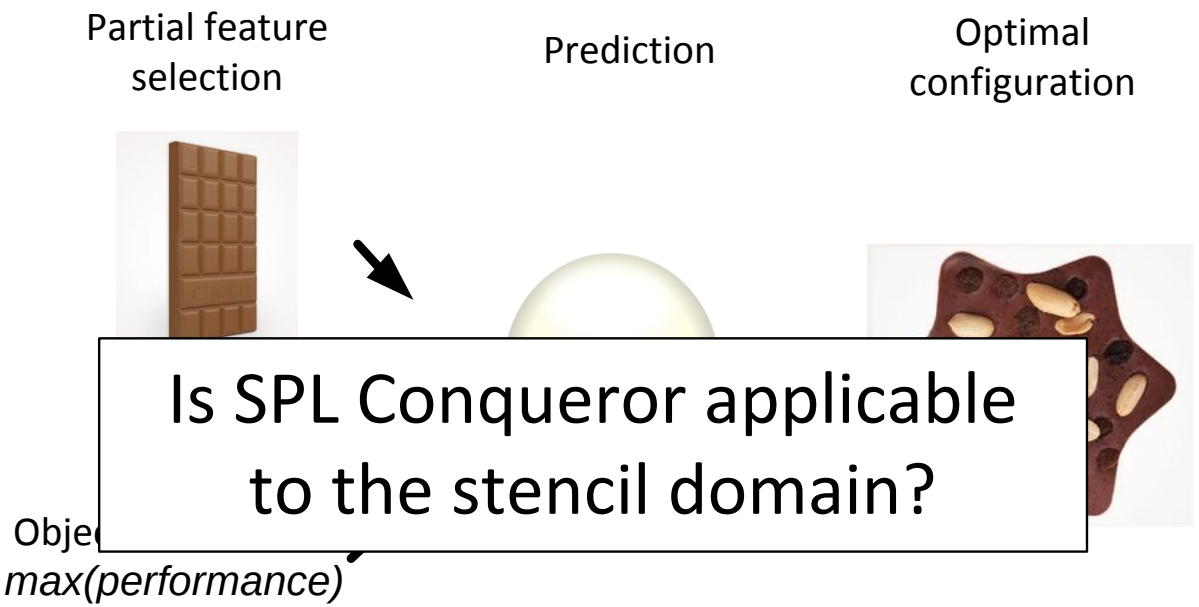
SPL Conqueror [Siegmund et al., 2012]



Advantages:

- Detection of feature interactions
- Transparent (i.e., influences of individual features and feature interactions explicitly modeled and quantified)

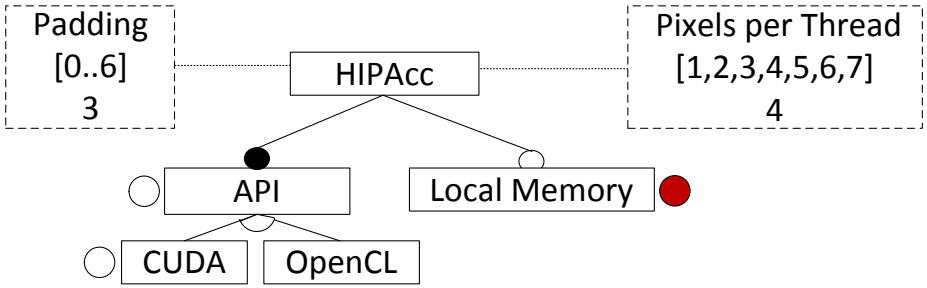
SPL Conqueror [Siegmund et al., 2012]



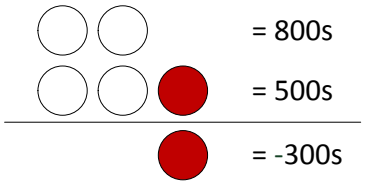
Advantages:

- Detection of feature interactions
- Transparent (i.e., influences of individual features and feature interactions explicitly modeled and quantified)

Influence of Individual Features

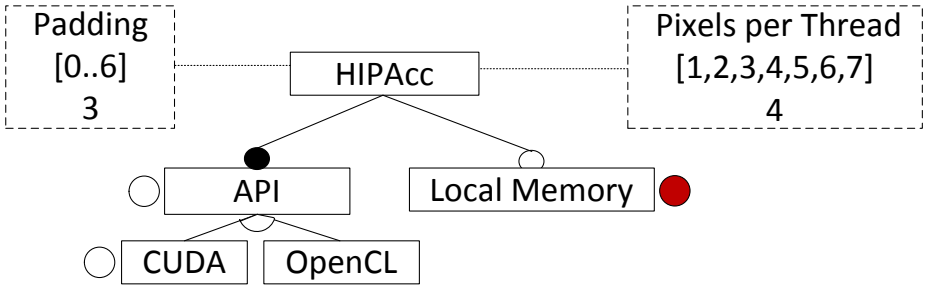


Identification:

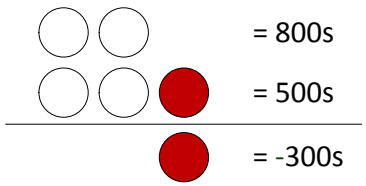


Performance difference is interpreted as contribution of the feature in question

Influence of Individual Features



Identification:

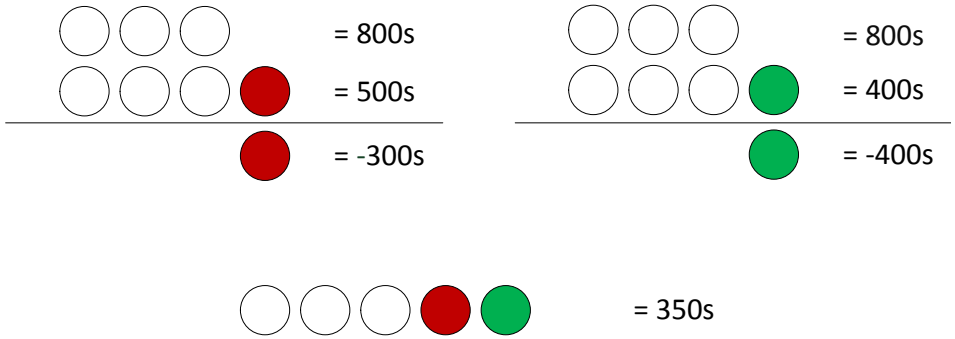


Performance difference is interpreted as contribution of the feature in question

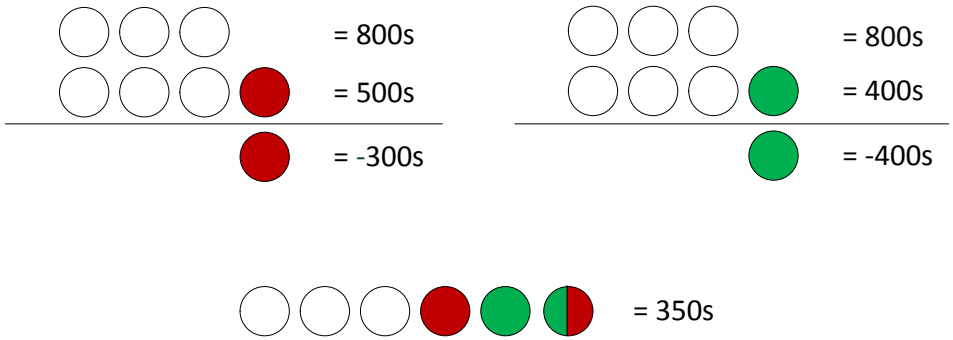
Heuristics:

Feature-wise (FW) heuristic: Quantifies the influence of individual features on performance

Interactions Between Features



Interactions Between Features

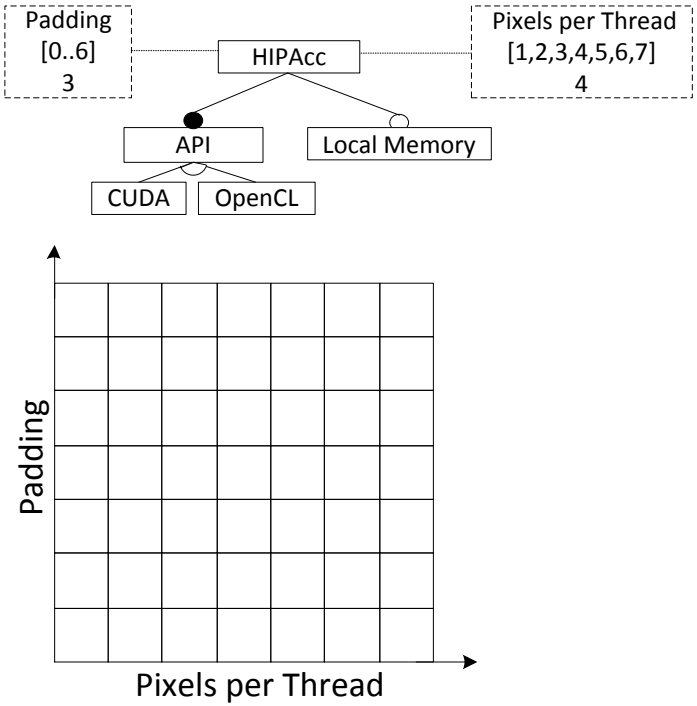


Heuristics:

- Pair-wise (PW) heuristic: interactions between two features
- Higher-order (HO) heuristic: interactions between three or more features
- Hot-spot (HS) heuristic: interactions of "hot-spot" features

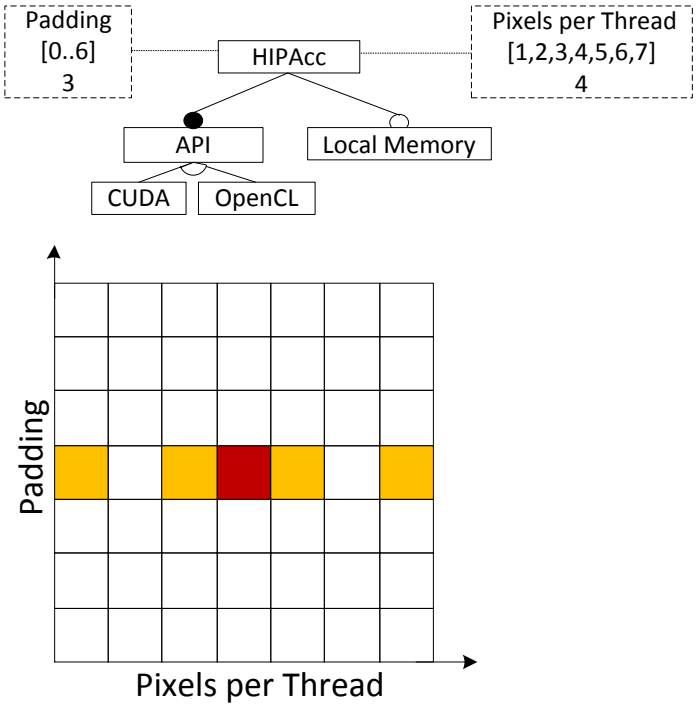
Influence of Parameters (Non-Boolean Features)

- Determine influence of parameter values on performance
- Learn performance function of pair (parameter, feature)
- Independent sampling of parameters



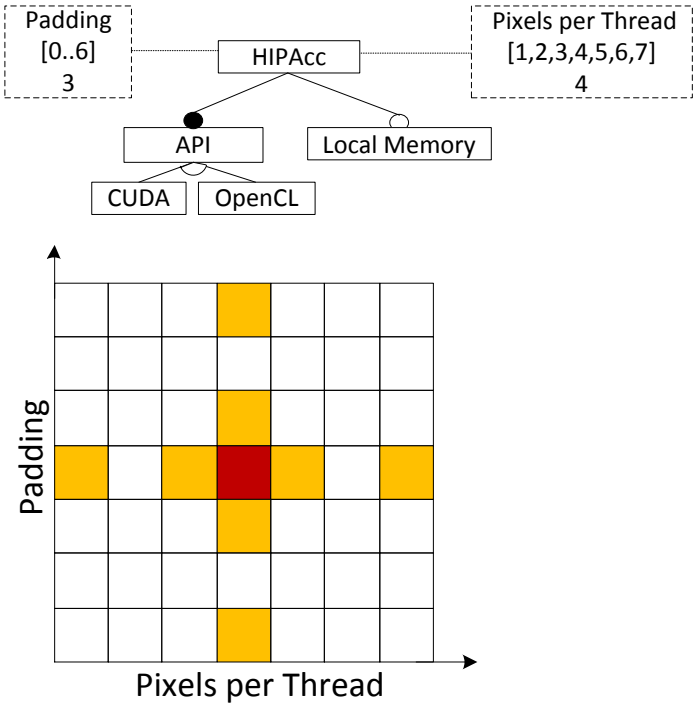
Influence of Parameters (Non-Boolean Features)

- Determine influence of parameter values on performance
- Learn function pair of parameter and feature
- Independent sampling of parameters



Influence of Parameters (Non-Boolean Features)

- Determine influence of parameter values on performance
- Learn function pair of parameter and feature
- Independent sampling of parameters



Heuristics:

- Function learning (FL) heuristic

Overview

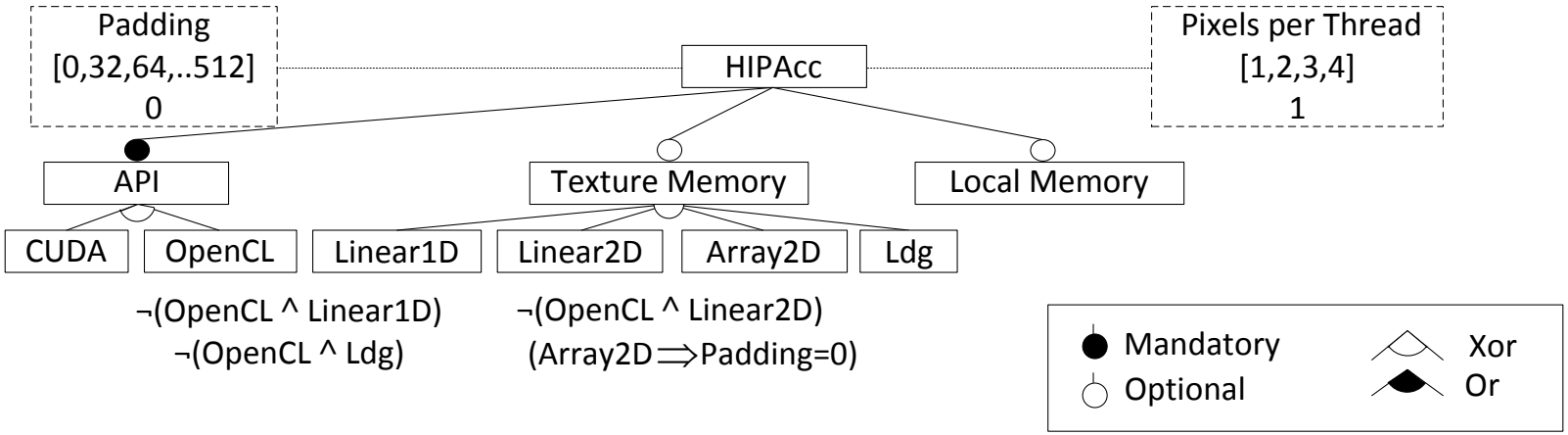
Research questions:

- What is the prediction accuracy of the different heuristics?
- Can we predict the performance-optimal configuration?

Customizable programs:

High Scalable Multi-Grid Solver

HIPA^{cc}


Padding
[0,32,64,..512]
0
HIPAcc
Pixels per Thread
[1,2,3,4]
1
API
Texture Memory
Local Memory
CUDA
OpenCL
Linear1D
Linear2D
Array2D
Ldg
¬(OpenCL ^ Linear1D)
¬(OpenCL ^ Ldg)
¬(OpenCL ^ Linear2D)
(Array2D ⟹ Padding=0)
Mandatory
Optional
Xor
Or

HIPA^{cc} – Results

Heu.
BF
FW
PW
HO
HS
FL

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

HIPA^{cc} – Results

Heu.	# M (in %)
BF	696 (100)
FW	26 (3.7)
PW	152 (21.8)
HO	277 (39.8)
HS	407 (58.5)
FL	52 (7.5)

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

HIPA^{cc} – Results

Heu.	# M (in %)	Time [ms]
BF	696 (100)	24 580.92
FW	26 (3.7)	698.30
PW	152 (21.8)	4 155.11
HO	277 (39.8)	9 384.90
HS	407 (58.5)	15 491.94
FL	52 (7.5)	1 513.92

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

HIPA^{cc} – Results

Heu.	# M (in %)	Time [ms]	$\mu \pm \sigma$
BF	696 (100)	24 580.92	0
FW	26 (3.7)	698.30	8.6 ± 10.2
PW	152 (21.8)	4 155.11	5.7 ± 9.9
HO	277 (39.8)	9 384.90	1.5 ± 3.7
HS	407 (58.5)	15 491.94	1.2 ± 3.1
FL	52 (7.5)	1 513.92	9.9 ± 15.7

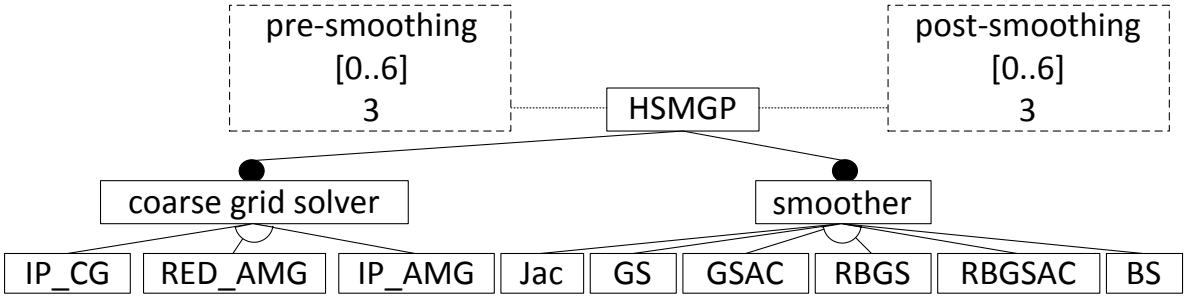
Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

HIPA^{cc} – Results

Heu.	# M (in %)	Time [ms]	$\mu \pm \sigma$	δ [%]
BF	696 (100)	24 580.92	0	0
FW	26 (3.7)	698.30	8.6 ± 10.2	2.1
PW	152 (21.8)	4 155.11	5.7 ± 9.9	0.1
HO	277 (39.8)	9 384.90	1.5 ± 3.7	0.1
HS	407 (58.5)	15 491.94	1.2 ± 3.1	0.1
FL	52 (7.5)	1 513.92	9.9 ± 15.7	<0.1

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

High Scalable Multi-Grid Solver



sum (pre-smoothing, post-smoothing) > 0

Legend:
IP_CG = In-Place Conjugate Gradient
IP_AMG = In-Place Algebraic Multi-Grid
RED_AMG = Algebraic Multi-Grid with data reduction
Jac = Jacobi
GS = Gauss-Seidel
GSAC = Gauss-Seidel with additional communication
RBGS = Red-Black Gauss-Seidel
RBGSAC = Red-Black Gauss-Seidel with additional communication
BS = Block-Smoother

High Scalable Multi-Grid Solver – Results

Heu.
BF
FW
PW
HO
HS
FL

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

High Scalable Multi-Grid Solver – Results

Heu.	# M (in %)
BF	864 (100)
FW	22 (2.5)
PW	192 (22.2)
HO	636 (73.6)
HS	864 (100)
FL	88 (10.2)

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

High Scalable Multi-Grid Solver – Results

Heu.	# M (in %)	Time [ms]
BF	864 (100)	2 230 326.94
FW	22 (2.5)	51 773.21
PW	192 (22.2)	518 721.48
HO	636 (73.6)	2 037 253.25
HS	864 (100)	2 230 326.94
FL	88 (10.2)	209 415.50

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

High Scalable Multi-Grid Solver – Results

Heu.	# M (in %)	Time [ms]	$\mu \pm \sigma$
BF	864 (100)	2 230 326.94	0
FW	22 (2.5)	51 773.21	51.9 ± 59.3
PW	192 (22.2)	518 721.48	12.2 ± 14.7
HO	636 (73.6)	2 037 253.25	8.5 ± 17
HS	864 (100)	2 230 326.94	0.2 ± 0.6
FL	88 (10.2)	209 415.50	11.2 ± 10.9

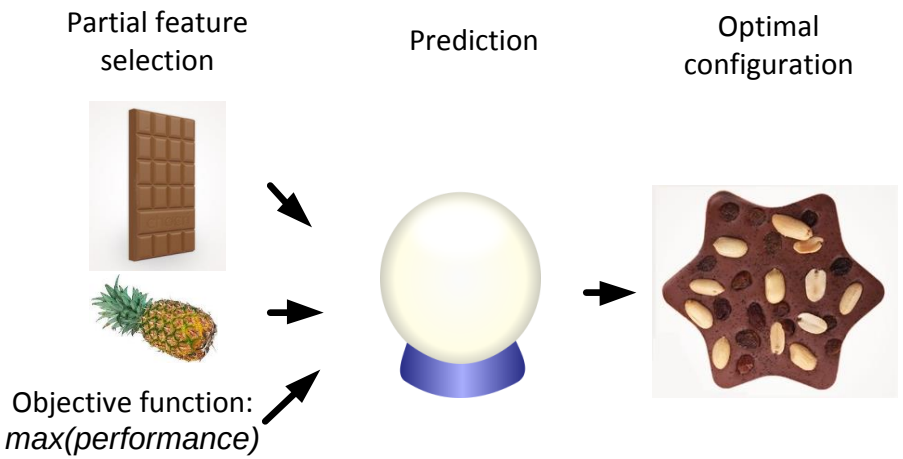
Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

High Scalable Multi-Grid Solver – Results

Heu.	# M (in %)	Time [ms]	$\mu \pm \sigma$	δ [%]
BF	864 (100)	2 230 326.94	0	0
FW	22 (2.5)	51 773.21	51.9 ± 59.3	8.3
PW	192 (22.2)	518 721.48	12.2 ± 14.7	16.3
HO	636 (73.6)	2 037 253.25	8.5 ± 17	217.5
HS	864 (100)	2 230 326.94	0.2 ± 0.6	0.4
FL	88 (10.2)	209 415.50	11.2 ± 10.9	61.1

Table: BF: brute force, FW: feature-wise, PW: pair-wise, HO: higher-order, HS: hot-spot, FL: function learning

Conclusion [Grebhahn et al., 2014]



Research questions:

- What is the prediction accuracy of the different heuristics?
- Can we predict the performance-optimal configuration?

HIPA^{cc}:

Performance-optimal configuration with error $<0.1\%$ when measuring 7% of all configurations

HSMGS:

Performance of all configurations with avg. error of 11% when measuring 10% of all configurations

References


Grebhahn, A., Siegmund, N., Kuckuk, S. A. S., Schmitt, C., and Köstler, H. (2014).
Optimizing performance of stencil code with SPL Conqueror.
In *Proc. HiStencils*, pages 7–14.


Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990).
Feature-oriented domain analysis (FODA) feasibility study.
Technical Report CMU/SEI-90-TR-2, CMU, SEI.


Siegmund, N., Kolesnikov, S. S., Kästner, C., Apel, S., Batory, D., Rosenmüller, M., and Saake, G. (2012).
Predicting performance via automated feature-interaction detection.
In *Proc. ICSE*, pages 167–177. IEEE.