Technische Universität Berlin

Cost Modelling for Vectorization on ARM

Angela Pohl, Biagio Cosenza and Ben Juurlink

ARM Research Summit 2018

Technische Universität Berlin

Challenges of Auto-Vectorization in Compilers

1. Is it **possible** to vectorize the code?
   - Passes: Loop Level Vectorization (LLV), Superword-Level Parallelism (SLP) …
   - Algorithms: loop transformations, partial vectorization, SLP padding …
   - Challenges: dependences, missing instructions …

2. Is it **beneficial** to vectorize the code?
   - Transformations might add overhead
   - Code can run slower afterwards

AES Embedded Systems Architecture

Technische Universität Berlin

Short Introduction to Cost Modelling

```
int s121() {
   for (int i = 0; i < LEN; i++){
      j = i + 1;
      a[i] = a[j] + b[i];
   }
   return 0;
}
```

C Source

```
%indvars.iv.next = add nuw nsw i64 %indvars.iv, 1, !dbg !55
%1 = load float, float* %arrayidx, align 4, !dbg !56, !tbaa !22
%2 = load float, float* %arrayidx7, align 4, !dbg !57, !tbaa !22
%add8 = fadd fast float %2, %1, !dbg !58
store float %add8, float* %arrayidx10, align 4, !dbg !60, !tbaa !22
%exitcond = icmp eq i64 %indvars.iv.next, 31999, !dbg !61
```

LLVM's Intermediate Representation

$$C_{add,VF=1} = 1$$
$$C_{load,VF=1} = 1$$
$$C_{load,VF=1} = 1$$
$$C_{fadd,VF=1} = 2$$
$$C_{store,VF=1} = 1$$
$$C_{icmp,VF=1} = 1$$

Block Cost Calculation

$$C_{loop} = \frac{1}{VF} \sum C_{instr,VF}$$

$$S_{est} = \frac{C_{scalar}}{C_{vec}}$$

Estimate Benefit

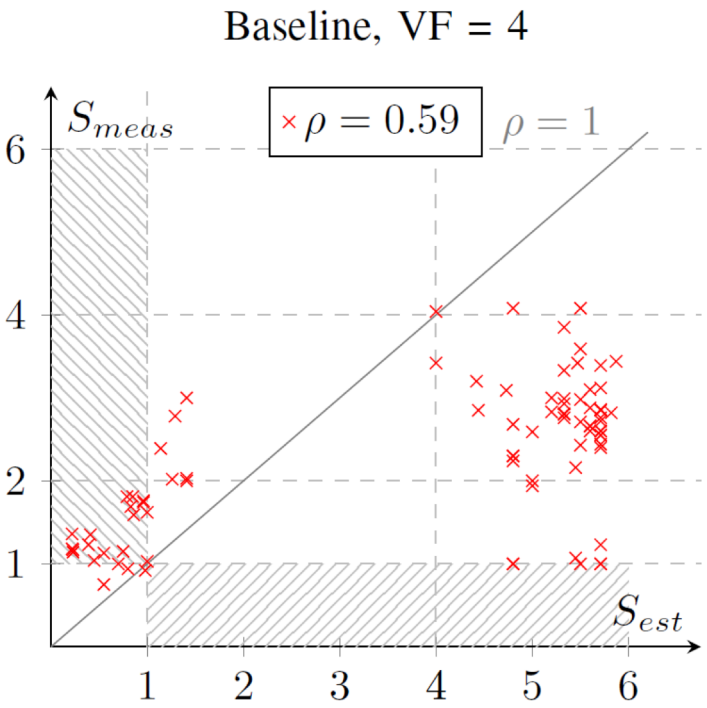
AES Embedded Systems Architecture

Technische Universität Berlin

State of the Art Analysis

- LLV pass of LLVM 6.0 on ARMv8
- TSVC Benchmark: 151 basic loop patterns
- Vectorization with overwritten cost model
- No unrolling, no interleaving

Metric	Baseline
Vectorized loops	61
$f\ominus$	14
$f\oplus$	0
t_{norm}	117.2



AES Embedded Systems Architecture

Technische Universität Berlin

Linear Modelling Approach

- Formulate basic block as linear equation for each test pattern

$$c_{vec} = \sum n_i c_i$$

- Determine basic block target cost based on measurement

$$c_{target} = \frac{c_{scalar}}{S_{meas}} = \sum n_i c_{i'}$$

- Apply mathematical fitting to set of linear equations

AES Embedded Systems Architecture

Technische Universität Berlin

Linear Modelling: Example

```
for (int i = 0; i < LEN; i++){
      a[i] = b[k] + 1.0;
}
```

$c_{scalar} = 8 \quad S_{meas} = 2.76 \Rightarrow \quad c_{target} = 2.89$

```
for (int i = 0; i < LEN; i++){
      prod *= a[i];
}
```

$c_{scalar} = 6 \quad S_{meas} = 2.30 \quad \Rightarrow \quad c_{target} = 2.61$

$$\mathbf{2.89} = \mathbf{1} \times c_{load} + \mathbf{1} \times c_{fadd} + \mathbf{0} \times c_{fmult} + \cdots$$

$$\mathbf{2.61} = \mathbf{1} \times c_{load} + \mathbf{0} \times c_{fadd} + \mathbf{1} \times c_{fmult} + \cdots$$

AES Embedded Systems Architecture

Technische Universität Berlin

Modelling Speedup Instead of Cost

Problem:

- Target costs vary significantly, i.e. data has to be fitted across large interval
- But: fitting benefits from smaller target intervals

Idea:

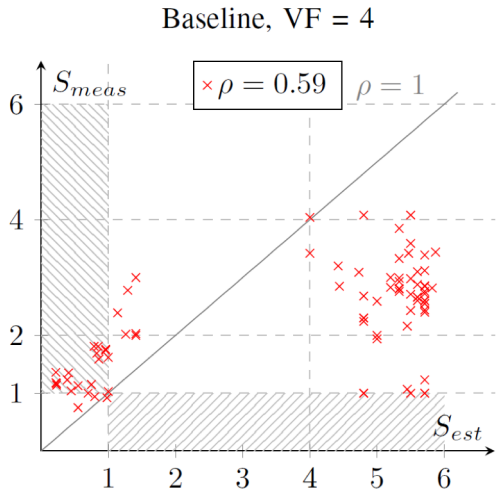
- Model speedup instead of cost
- Limits interval to (0, vectorization factor)
- Linear equations become $S_{meas} = \sum n_i w_i$

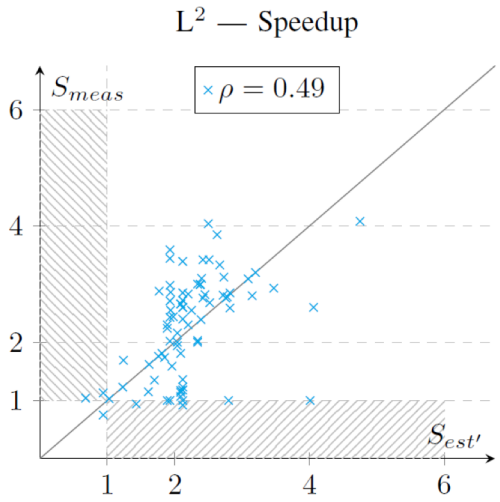
  with n_i = numer of instructions of same type, w_i = corresponding weight

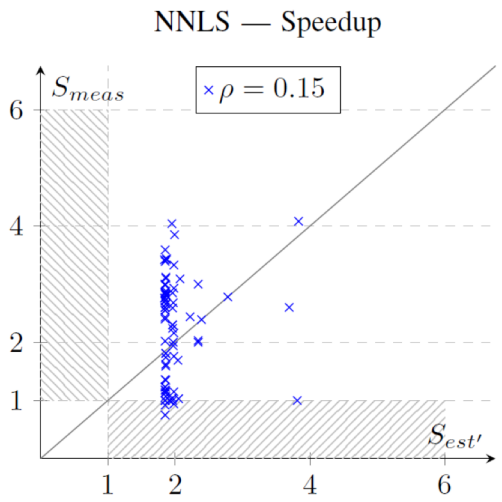
AES Embedded Systems Architecture

Technische Universität Berlin

Results: Fitted for Speedup







- L2: Least Squares (minimizes Euclidian L2 Norm)
- NNLS: Non-Negative Least Squares (all coefficients > 0)

AES Embedded Systems Architecture

Technische Universität Berlin

Adding Block Composition as Feature

Problem:

- Current cost model only cares about individual instructions
- Arithmetic intensity can have major impact on speedup, e.g. if code is memory bound

Idea:

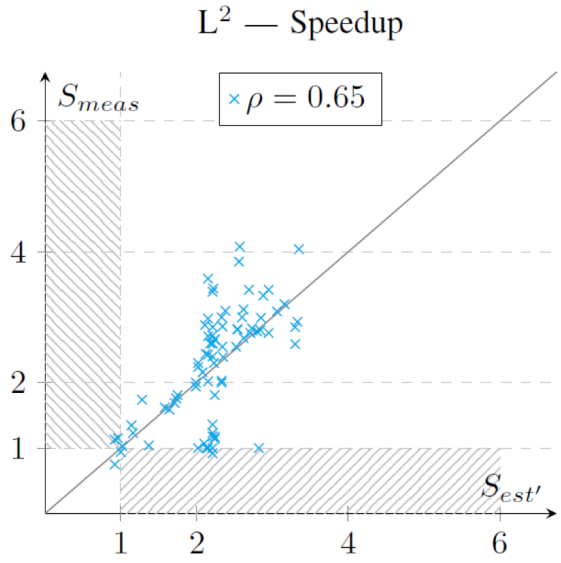
- Replace count of instruction type with overall percentage, e.g. 20% load, 10% cmp, …

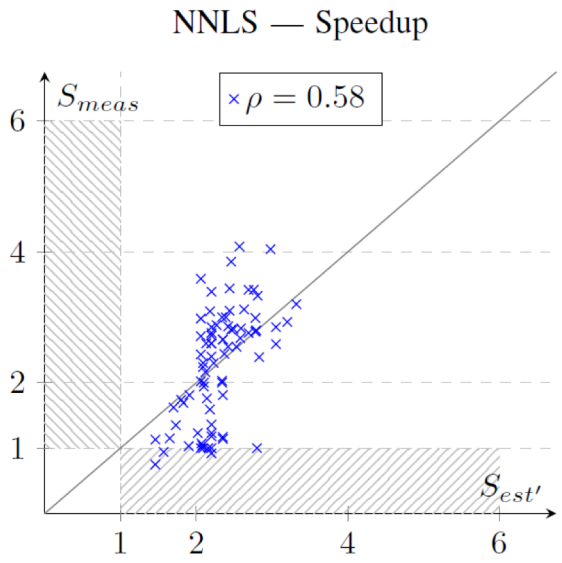
$$S_{meas} = \sum \frac{n_i}{n_{total}} w_i$$

AES Embedded Systems Architecture

Technische Universität Berlin

Results: Fitted with Rated Instruction Count





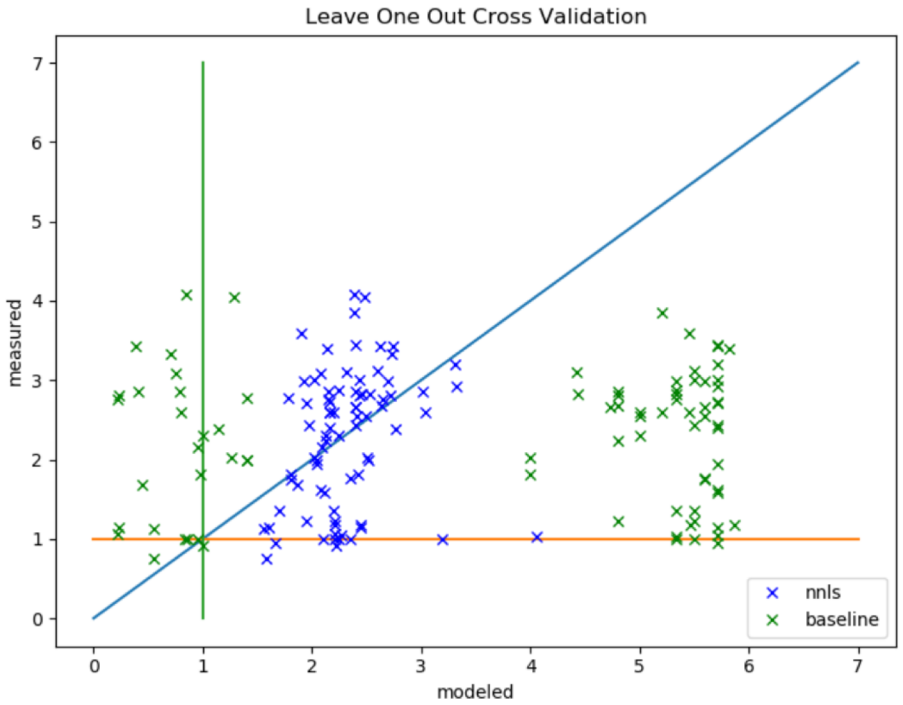
Metric	Baseline	L^2	$NNLS$
Vectorized loops	61	76	80
$f\ominus$	14	2	0
$f\oplus$	0	1	3
t_{norm}	117.2	113.2	113.4

AES Embedded Systems Architecture

Technische Universität Berlin

Leave One Out Cross Validation: NNLS

- Fitting is done with whole training data set except for one kernel
- Kernel is then predicted with that model



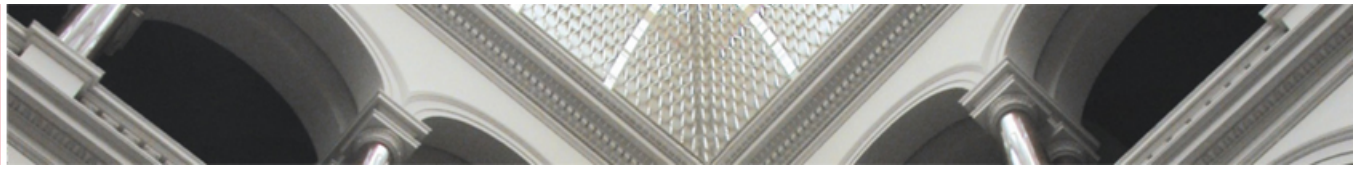
AES Embedded Systems Architecture

Technische Universität Berlin

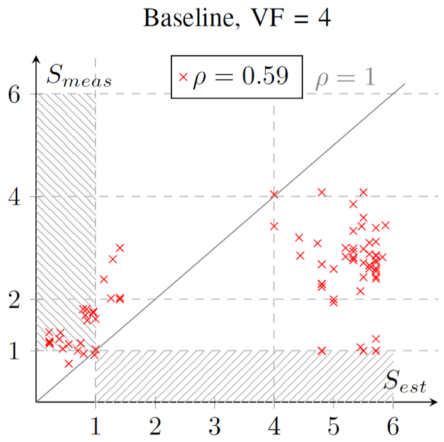
Conclusion

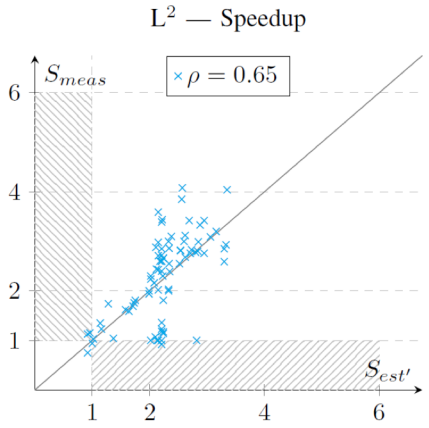
- Compilers need more accurate cost models to avoid mispredictions
- Aligned cost models enable comparison of different transformation options
- With our refined cost model we
  - Increase the correlation between estimated and measured speedup
  - Decrease the number of false predictions
  - Lower execution times
- Next steps: add more code features and tests to cover all instruction types

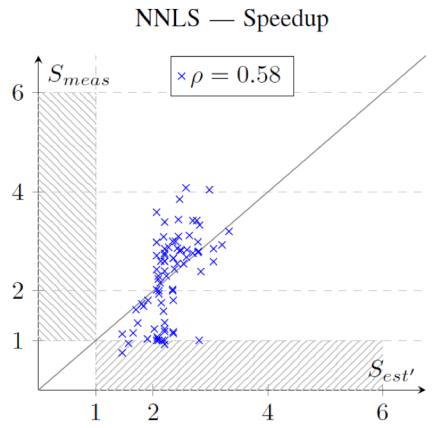
AES Embedded Systems Architecture



Technische Universität Berlin







THANK YOU

AES Embedded Systems Architecture

Technische Universität Berlin

BACKUP

AES Embedded Systems Architecture

Why a More Accurate Cost Model?

1. Improve classification results to decide whether code should be vectorized or not.

2. Enable comparison of different transformations with each other, not just to baseline.

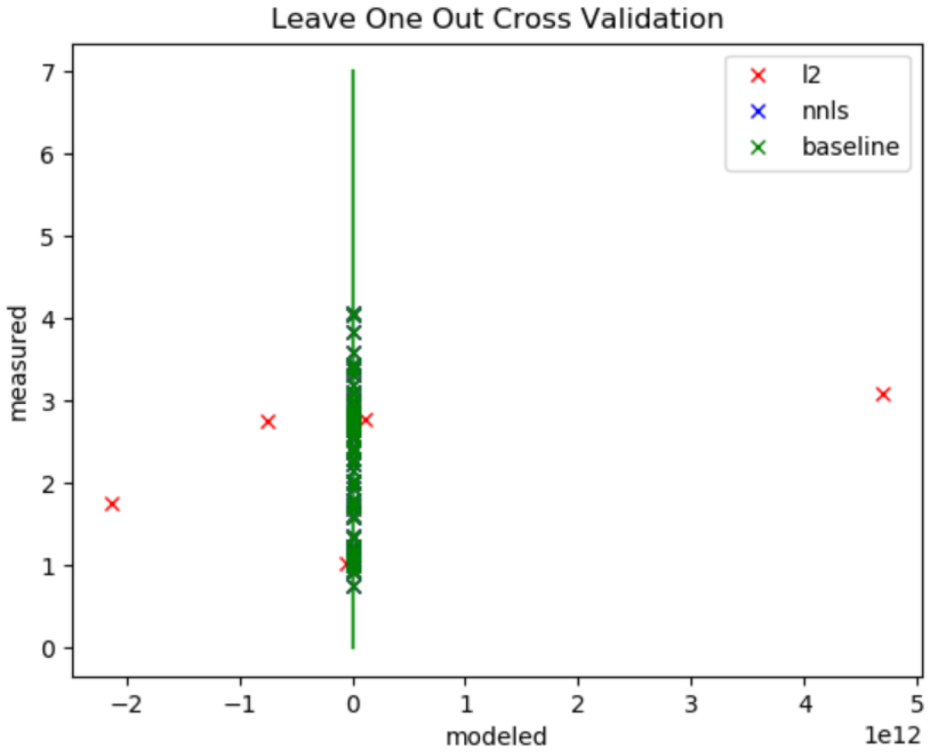
```
for (int i = 0; i < LEN/2; i++){
      k = j + 1;
      a[i] = b[k] – d[i];
      j = k + 1
      b[k] = a[i] + c[k];
}
```

	LLV	SLP
Predicted Speedup	0.96	1.00
Measured Speedup	1.13	1.20

Example code run on Intel i5

Technische Universität Berlin

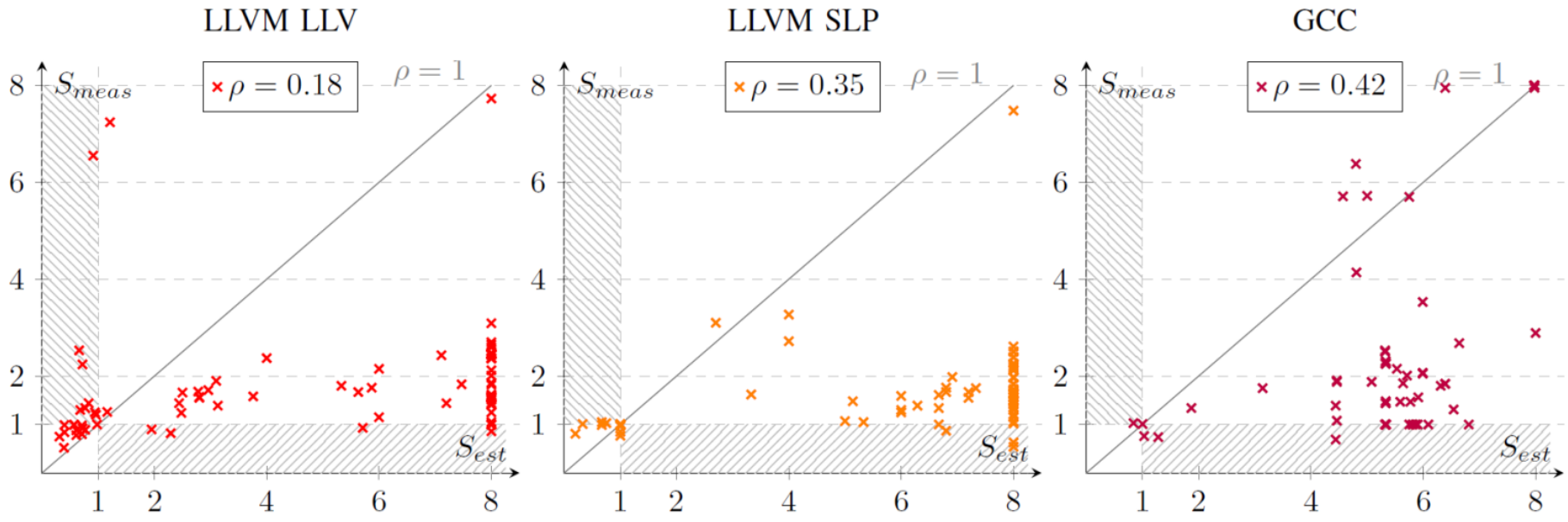
L2- LOOCV Validation Results



AES Embedded Systems Architecture

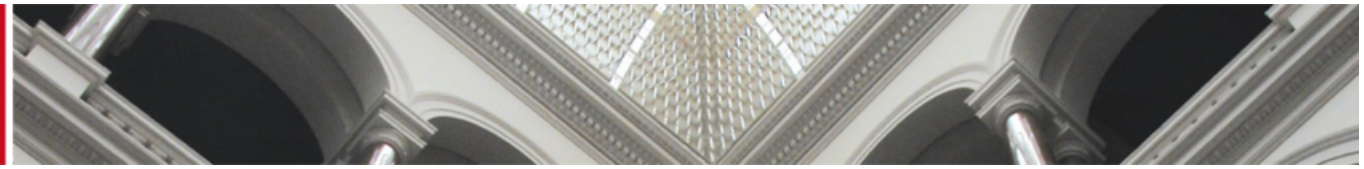
Technische Universität Berlin

State of the Art Analysis – x86



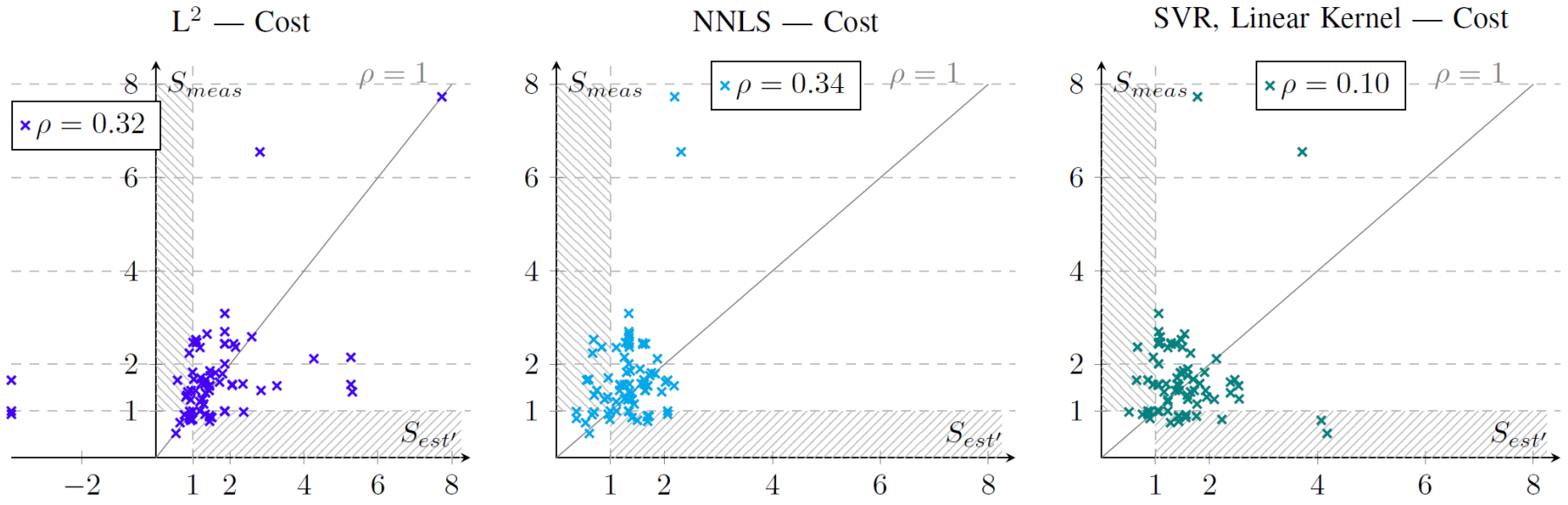
- TSVC benchmark: 151 basic loop patterns
- SLP vectorization applied after loop unrolling
- Results shown for Intel Xeon E5 with AVX2

AES Embedded Systems Architecture



Technische Universität Berlin

Results: Fitted for Cost – x86

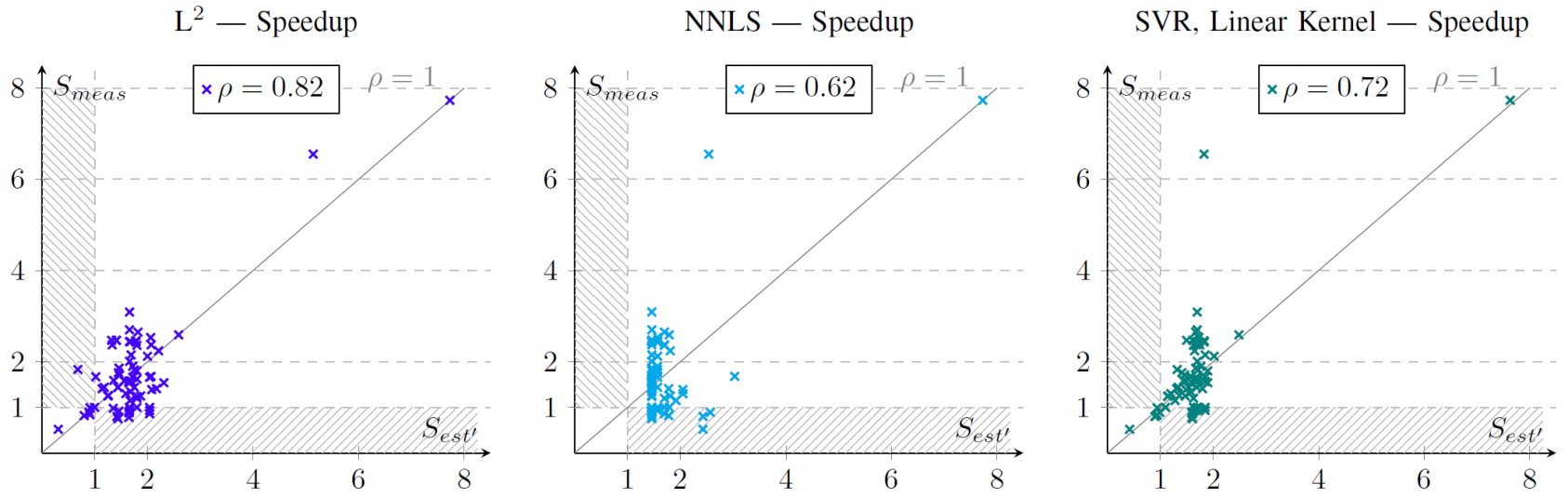


- Correlation between estimated and measured speedup in LLVM‘s LLV pass after fitting
- L2: Least Squares (minimizes Euclidian L2 Norm)
- NNLS: Non-Negative Least Squares (all coefficients > 0)
- SVR: Support Vector Regression

AES Embedded Systems Architecture

Technische Universität Berlin

Results: Fitted for Speedup – x86



- All three approaches improve correlation further
- False negatives reduced (L2) or eliminated (NNLS, SVR)
- But: small increase in false positives

AES Embedded Systems Architecture