Performance Modeling and Engineering Using Kerncraft

Julian Hammer <julian.hammer@fau.de>
Friedrich-Alexander-University of Erlangen-Nuremberg
Regional Computing Center Erlangen
Georg Hager (advisor), Gerhard Wellein (advisor)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

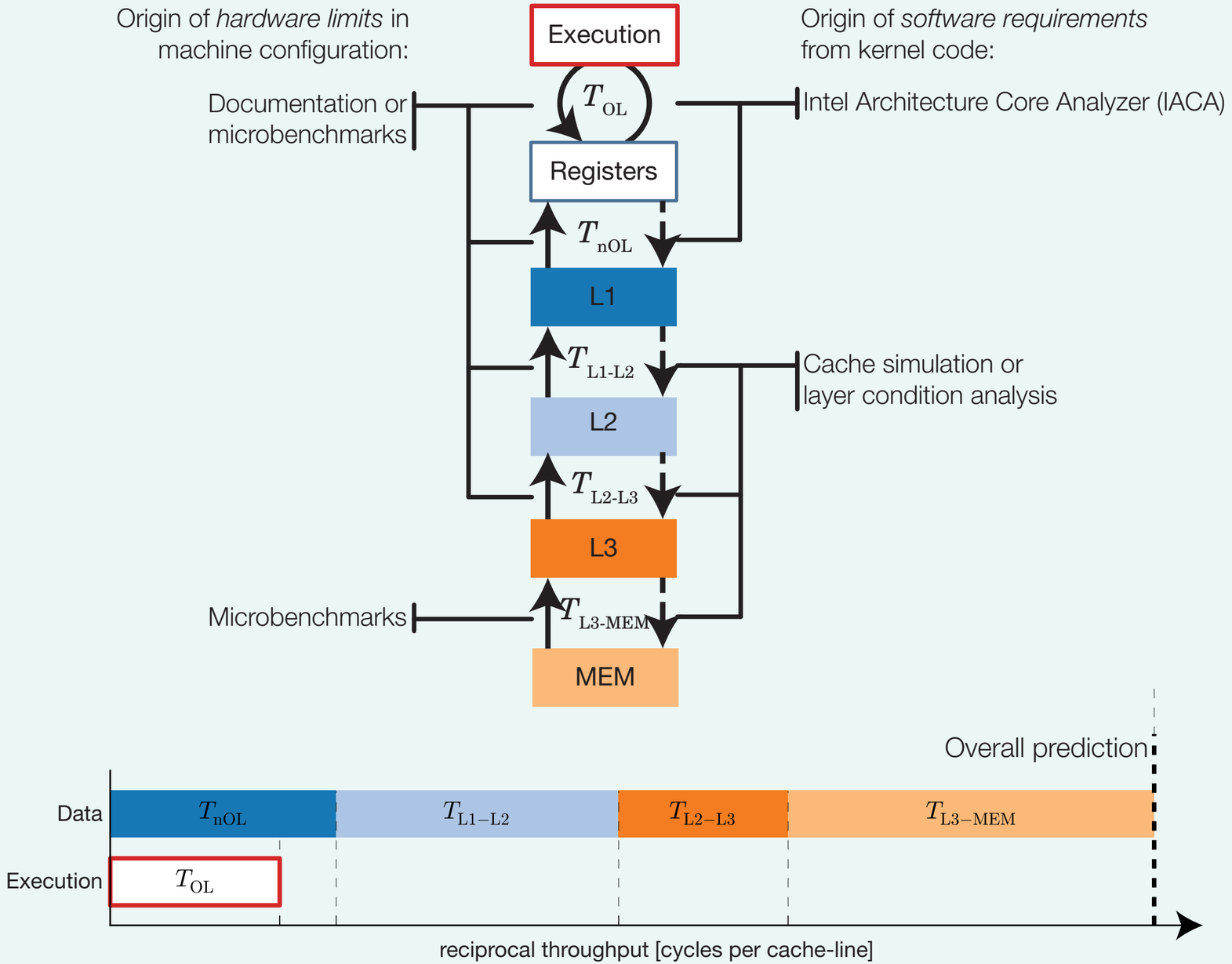
Goal

Automatically predict single-socket performace of regular algorithms
Select the most efficient core configuration
Select good tiling sizes to increase cache utilization

Approach

1. Build Execution-Cache-Memory model
2. Predict serial performance and optimal scaling
3. Analytically select optimal spatial tiling

Execution-Cache-Memory (ECM) Model[0] for x86



Kerncraft[1]

Released under AGPLv3
github.com/RRZE-HPC/kerncraft

User Input

Kernel Code

Machine Configuration

Automatic Modeling

ECM[0] & Roofline[3] Model

Layer Conditions[5] Model

Results and Validation

ECM Prediction

Parameter Study

Optimal Scaling Point

Blocking Suggestion

Kernel Code

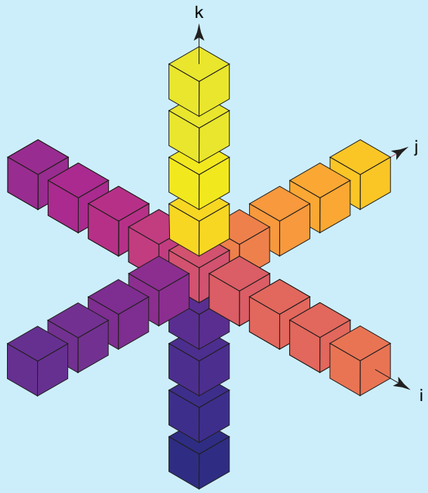
User Input

Machine Configuration

3D-long-range kernel code:

```
double U[M][N][N], V[M][N][N], ROC[M][N][N];
double c0, c1, c2, c3, c4, lap;

for(int k=4; k < M-4; k++) {
for(int j=4; j < N-4; j++) {
for(int i=4; i < N-4; i++) {
    lap = c0*V[k][j][i]
        + c1*(V[ k ][ j ][i+1] + V[ k ][ j ][i-1] + V[ k ][j+1][ i ])
        + c1*(V[ k ][j-1][ i ] + V[k+1][ j ][ i ] + V[k-1][ j ][ i ])
        + c2*(V[ k ][ j ][i+2] + V[ k ][ j ][i-2] + V[ k ][j+2][ i ])
        + c2*(V[ k ][j-2][ i ] + V[k+2][ j ][ i ] + V[k-2][ j ][ i ])
        + c3*(V[ k ][ j ][i+3] + V[ k ][ j ][i-3] + V[ k ][j+3][ i ])
        + c3*(V[ k ][j-3][ i ] + V[k+3][ j ][ i ] + V[k-3][ j ][ i ])
        + c4*(V[ k ][ j ][i+4] + V[ k ][ j ][i-4] + V[ k ][j+4][ i ])
        + c4*(V[ k ][j-4][ i ] + V[k+4][ j ][ i ] + V[k-4][ j ][ i ]);
    U[k][j][i] = 2.f * V[k][j][i] - U[k][j][i] + ROC[k][j][i] * lap;
}}}
```



Accesses to array **V** (stencil)

Sandy-Bridge Machine configuration file:

CPU & compiler information

```
model name: Intel(R) Xeon(R) CPU
            E5-2680 @ 2.70GHz
sockets: 2
cores per socket: 2
```

Memory subsystem

```
memory hierarchy:
    - level: L1
      cache per group: {
          'sets': 64, 'ways': 8,
          'cl_size': 64,
          'replacement_policy': 'LRU',
          'write_allocate': True,
          'write_back': True,
          'load_from': 'L2',
          'store_to': 'L2'}
      cores per group: 1
      cycles per cacheline transfer: 2
```

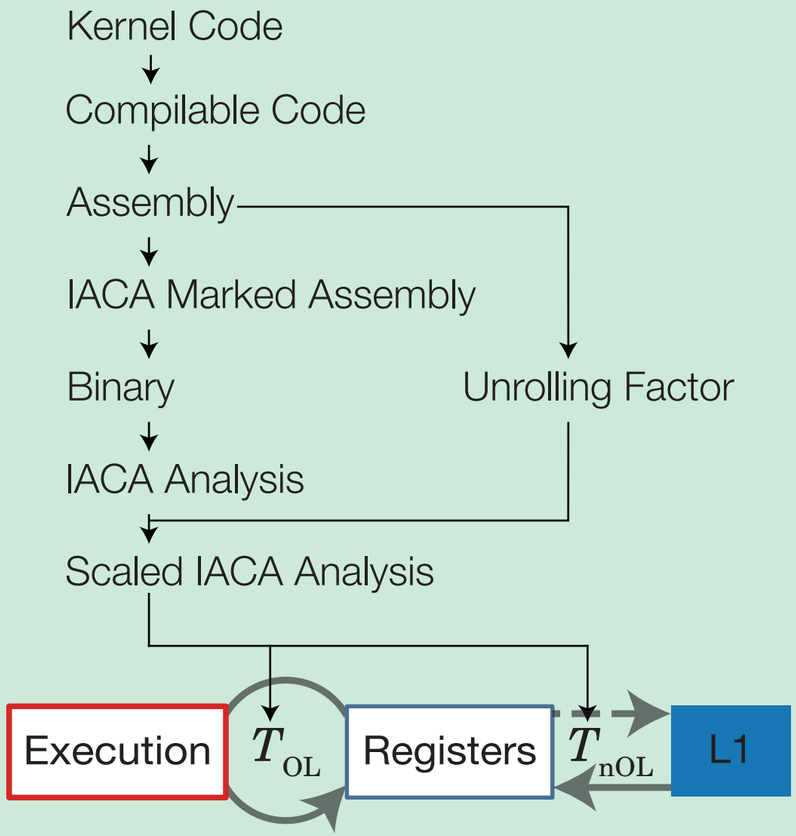
Benchmark results

```
benchmarks: {measurements: {MEM: {results:
    update: [18.91 GB/s, 32.43 GB/s,
             37.28 GB/s, 39.98 GB/s,
             40.99 GB/s, 40.92 GB/s,
             40.61 GB/s, 40.34 GB/s]}}
```

ECM[0] & Roofline[3] Model

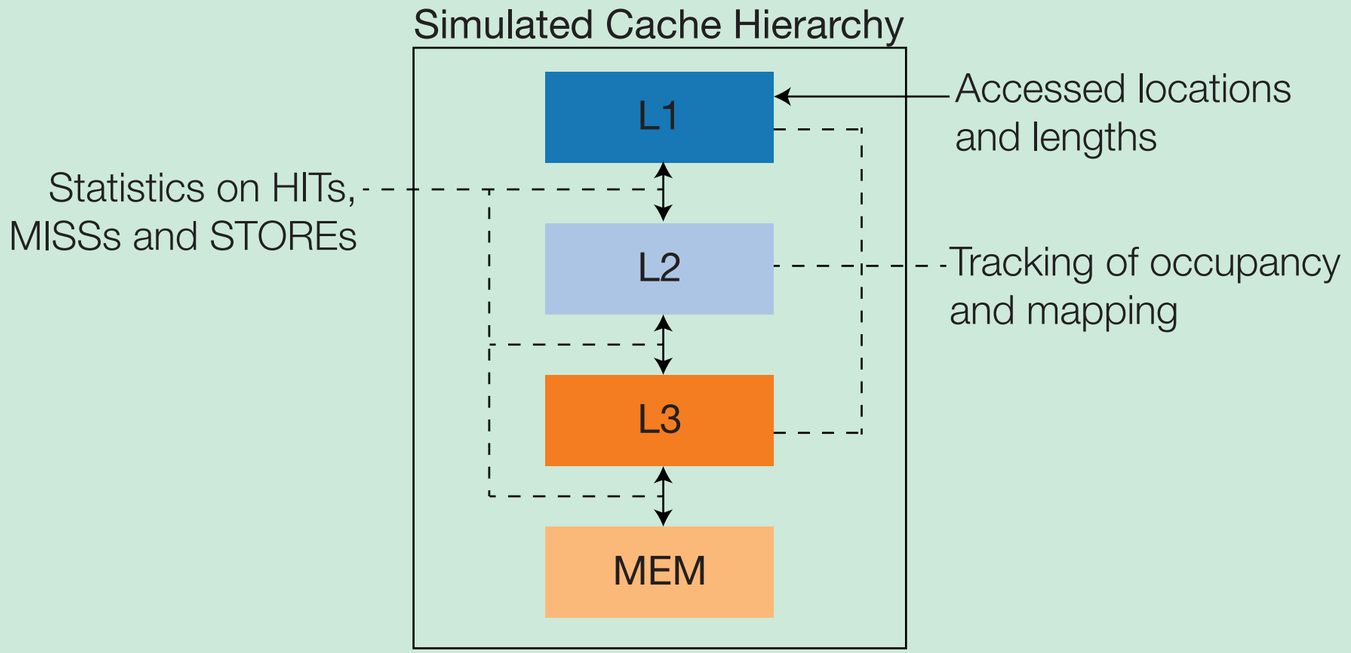
Automatic Modeling

In-Core with IACA



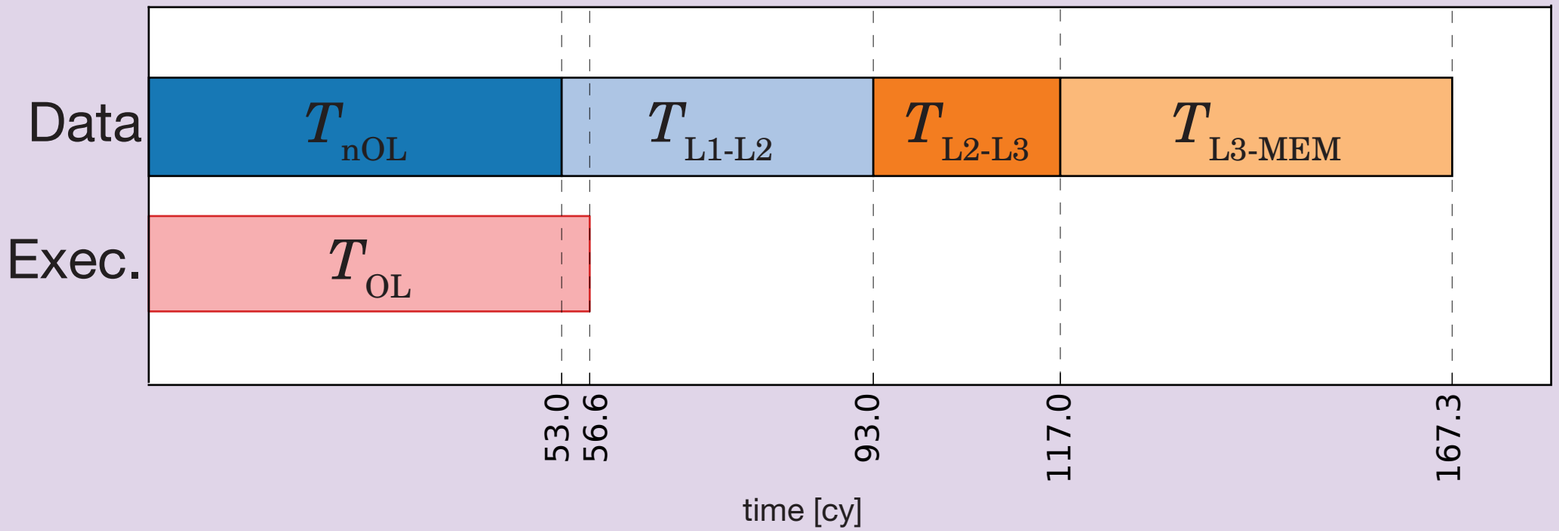
Cache with pycachesim

Released under AGPLv3
github.com/RRZE-HPC/pycachesim



ECM Prediction

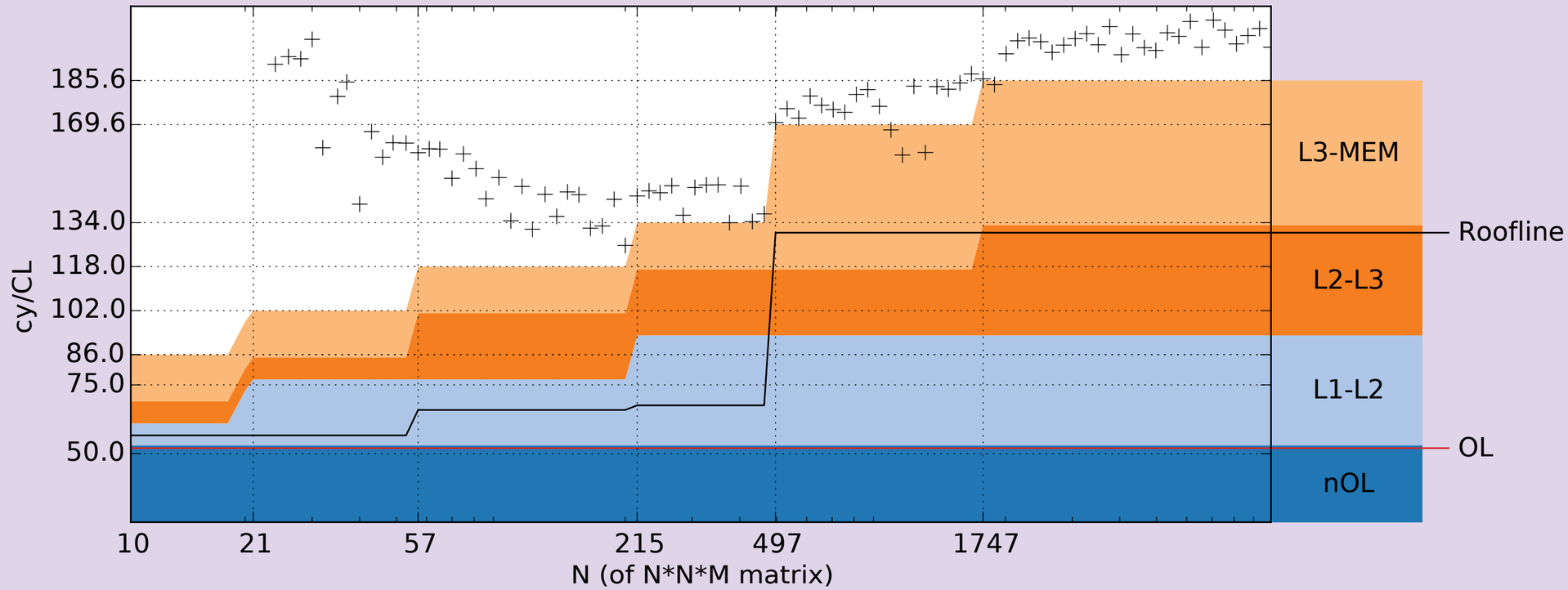

```
$ kerncraft -p ECM -m E5-2680_2.7.yaml 3d-long-range.c
            -D N 1024 -D M 1024
========================= kerncraft ============================
3d-long-range-stencil.c                          -m E5-2680_2.7.yaml
-D M 1000 -D N 1000
---------------------------- ECM -------------------------------
{ 56.6 || 53.0 | 40.0 | 24.0 | 50.3 } cy/CL
{ 56.65 \ 93.0 \ 117.0 \ 167.3 } cy/CL
[...]
$
```



Parameter Study

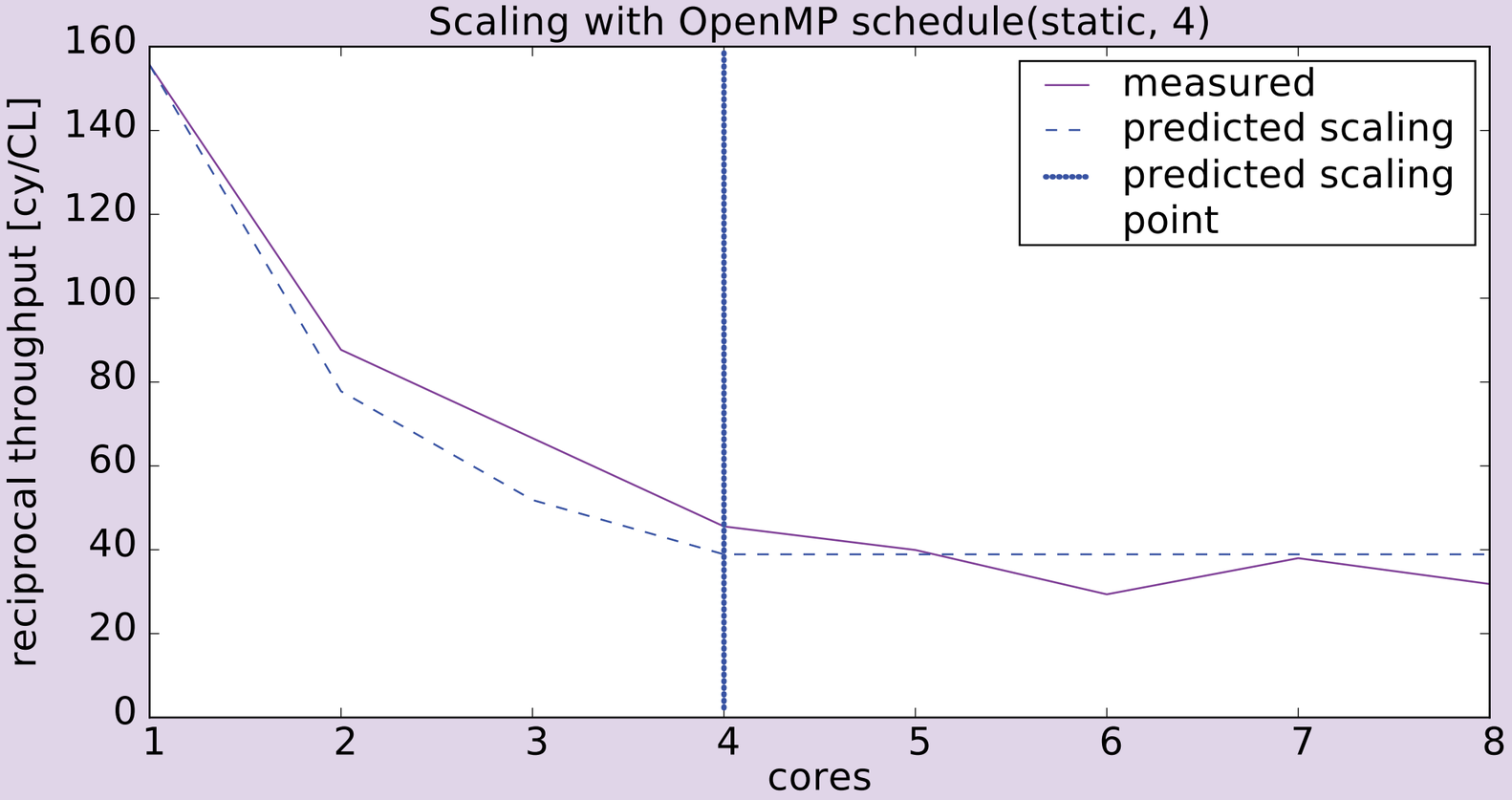
Results and Validation

```
$ kerncraft -p ECM -p LC -p Benchmark -m E5-2680_2.7.yaml
            3d-long-range.c -D N 10-5000:100:log10 -D M [...]
[... analysis results for 100 data points ...]
$
```



Optimal Scaling Point

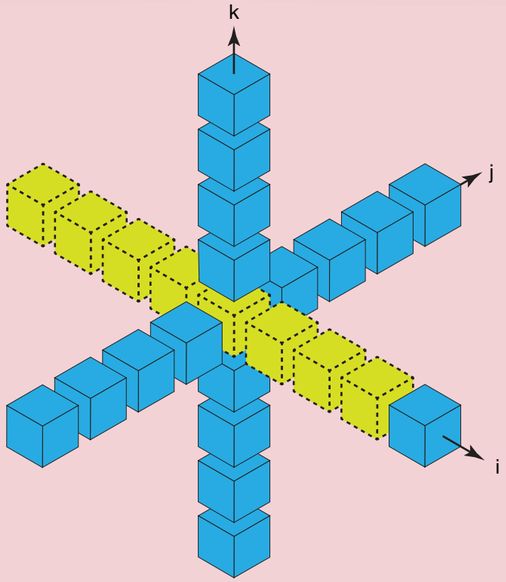

```
$ kerncraft -p ECM -m E5-2680_2.7.yaml 3d-long-range.c \
            -D M 1000 -D N 1000
[...]
saturating at 4 cores
$
```



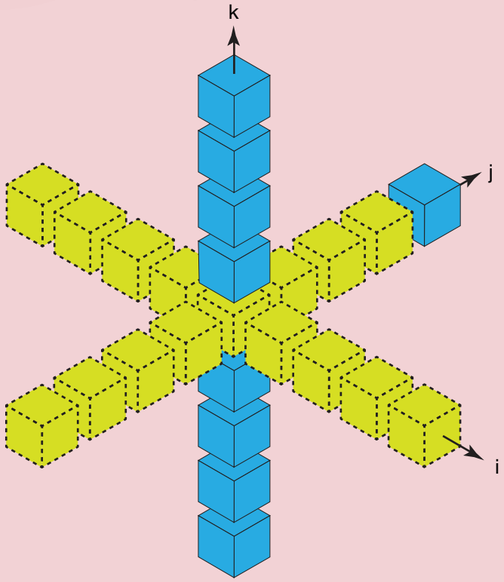
Automatic Modeling

Layer Conditions[5] Model

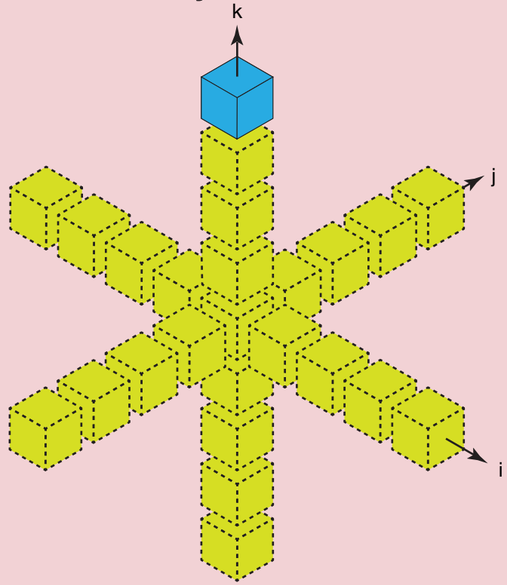
1D layer condition

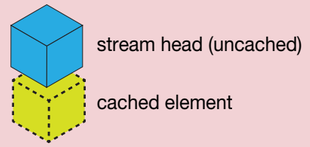


2D layer condition



3D layer condition





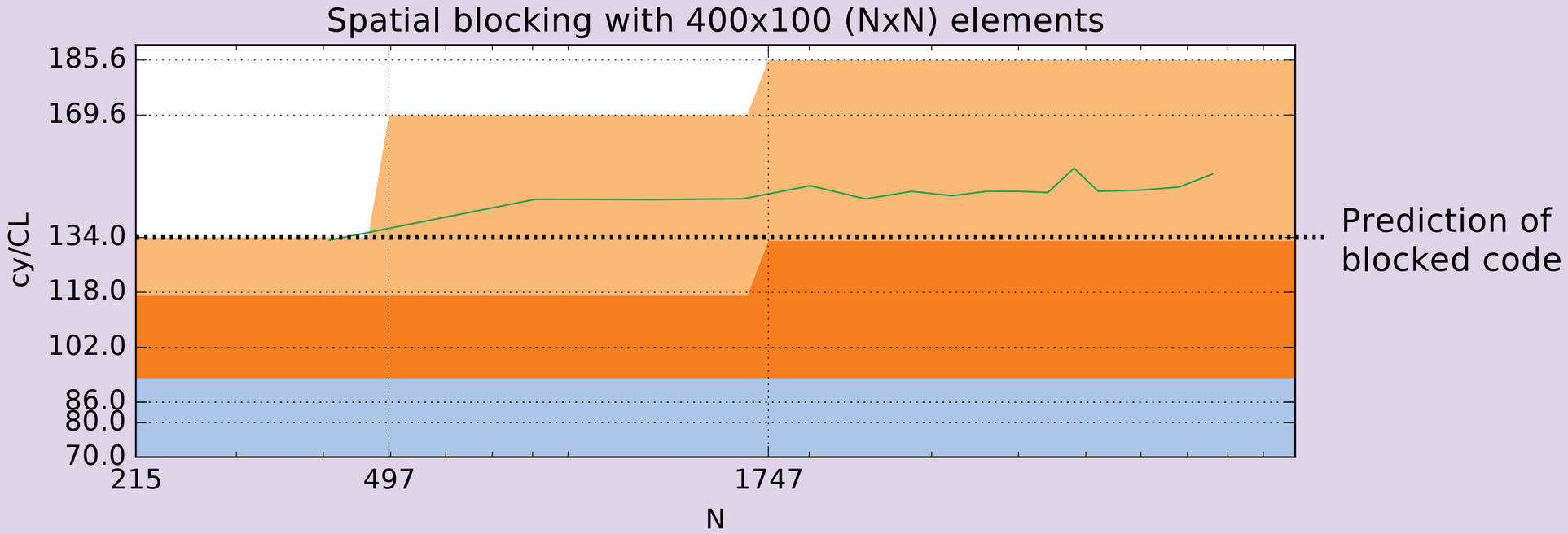
$$C_{\text{req.}} = \left(\sum L_{\text{rel.offsets}} + \max(L_{\text{rel.offsets}}) * n_{\text{slices}} \right) * s$$

- $C_{\text{req.}}$: Required size
- $\sum L_{\text{rel.offsets}}$: Sum over all relative offsets between accesses in slices
- $\max(L_{\text{rel.offsets}})$: Longest relative offset over all slices
- n_{slices}: Number of slices in dimension
- s: Bytes per element

Blocking Suggestion

Results and Validation

```
$ kerncraft -p LC -m E5-2680_2.7.yaml 3d-long-range.c
            -D N 1024 -D M 1024
[...]
2D Layer-Condition:              3D Layer-Condition:
L1: N <= 216                     L1: N <= 19
L2: N <= 1725                    L2: N <= 55
L3: N <= 137971                  L3: N <= 488
$
```



Julian Hammer <julian.hammer@fau.de>

Interactive LC Calculator:

https://rrze-hpc.github.io/layer-condition/#calculator

Open Source and freely available at:

github.com/RRZE-HPC/kerncraft

github.com/RRZE-HPC/pycachesim

Future Work

In-core simulation

LLVM-Polly integration

Graph applications

Simpler Layer Conditions

Supported by

Federal Ministry of Education and Research

SKAMPY

DAAD

Deutscher Akademischer Austauschdienst
German Academic Exchange Service

FITweltweit

We would also like to thank

Prof. Pingali at ICES, UT Austin
Prof. Hack at CDL, Saarland University