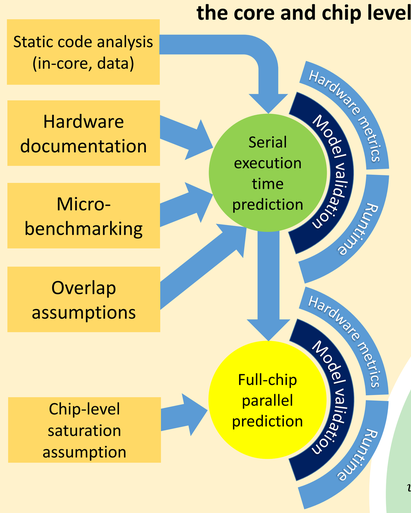
Applying the Execution-Cache-Memory Performance Model: Current State of Practice

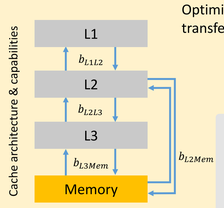
G. Hager[1], J. Eitzinger[1], J. Hornich[1], F. Cremonesi[2], C. A. Alappat[1], T. Röhl[1], G. Wellein[1]

[1] Erlangen Regional Computing Center (RRZE), Erlangen, Germany [2] École Polytechnique Fédérale de Lausanne (EPFL), Geneva, Switzerland

An analytic runtime model for loop code on the core and chip level



Cache architectures and transfer times



Optimistic transfer times: $T_i = \frac{V_i}{b_i}$

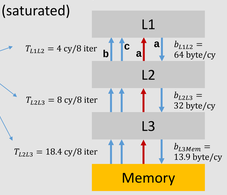
- from code analysis (interaction model)
- from machine parameters or microbenchmarks (machine model)

Calculate for some # of iterations, e.g., length of 1 cache line; consider cache reuse. $i \in \{ L1L2, L2L3, \dots \}$

Example: STREAM triad `a(:) = b(:) + s * c(:)` on Xeon Broadwell2.3 GHz in CoD mode: inclusive caches, $b_{L3Mem} = 32$ GB/s per NUMA domain (saturated)

$$V_{L1L2} = \frac{4 \times 64\ \text{byte}}{8\ \text{iter}} = 32\ \frac{\text{byte}}{\text{iter}}$$

$$V_{L2L3} = V_{L3Mem} = V_{L1L2}$$



Short notation: $\{T_{L1L2} | T_{L2L3} | T_{L3Mem}\} = \{ 4 \mid 8 \mid 18.4 \}$ cy/8 iter

Problem: Saturation

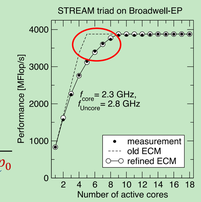
Plain ECM model too optimistic @ saturation for tight kernels with small T_{OL}

Refinement: **Adaptive latency penalty**, depends on bus utilization $u(n)$:

$$u(1) = \frac{T_{L3Mem}}{T_{Mem}^{ECM}} \leftarrow \text{single-core model}$$

$$u(n) = \frac{T_{L3Mem}}{T_{Mem}^{ECM} + (n-1)u(n-1)p_0}$$

Fit parameter, not code independent → future work



Hofmann et al., ISC 2018

Analysis tools:

- Pen & paper
- Layer conditions (for stencils)
- pycachesim
- Kerncraft

Traffic validation tools:

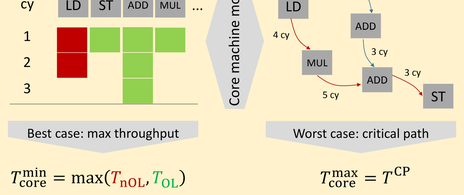
- Likwid-perfctr
- PAPI
- Linux perf

In-core execution model

```
do i=1,N
  a(i) = ...
  r = r + ...
enddo
```

```
.label:
vmovaps  ymm0,[rsi+8*rdx]
vaddpd ymm1,ymm1,ymm0
...
jb .label
```

Throughput / critical path analysis



$T_{\text{core}}^{\min} = \max(T_{\text{nOL}}, T_{\text{OL}})$ $\quad T_{\text{core}}^{\max} = T^{\text{CP}}$

T_{nOL} interacts with cache hierarchy, T_{OL} does not

Example: 2D DP 5-pt stencil on Xeon Skylake Gold 6148, AVX-512

```
do j=1,N
 do i=1,N
  y(i,j) = 0.25d0 *
  (x(i-1,j) + x(i+1,j)
 + x(i,j-1) + x(i,j+1))
 enddo
enddo
```

Throughput analysis

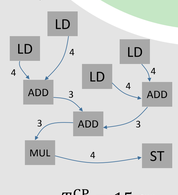
	#µops	#cy
LD	4	2
ST	1	1
ADD	3	1.5
MUL	1	0.5
Σ	9	2

Retirement limit!

$T_{\text{nOL}} = 2$ cy

$T_{\text{OL}} = 1.5 + 0.25$ cy

CP analysis



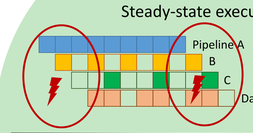
$T^{\text{CP}} = 15$ cy

Analysis tools:

- Pen & paper
- Intel Architecture Code Analyzer (IACA)
- Open-Source Architecture Code Analyzer (OSACA)
- Kerncraft

Problem: Non-steady-state, latencies

Main ECM model prerequisite: Steady-state execution



ECM too optimistic!

Irregular data accesses handled by best/worst case analysis only

`a[ind[i]]`

Best: `ind[i] = i+c` → streaming

Worst: `ind[i] = rnd` → latency penalty

From time to performance

f	CPU frequency
f_0	CPU base frequency
W	Amount of work

$$P = \frac{W}{T^{ECM}(f, f_0)}$$

Example: Non-overlapping CPU with on-chip clock domain down to L3, b_{Mem} independent of f

$$\{P_{L1}^{ECM} \rceil \dots \rceil P_{Mem}^{ECM}\} = \frac{W \cdot f}{\{ T_{L1}^{ECM} \rceil T_{L2}^{ECM} \rceil T_{L3}^{ECM} \rceil \max(T_{L3}^{ECM} + \underbrace{T_{L3Mem} \cdot f/f_0}_{f\text{-independent } P \text{ component}}, T_{OL}) \}}$$

Putting the model together: Overlap assumptions

Notation for model **contributions**:

$\{T_{\text{OL}} \parallel T_{\text{nOL}} | T_{L1L2} | T_{L2L3} | T_{L3Mem}\} = \{ 7 \parallel 2 \mid 4 \mid 8 \mid 18.4 \}$ cy/8 iter

How to put the contributions together? Overlap assumptions! (Part of the machine model)

Most pessimistic assumption: **no overlap** of data-related contributions

$$T_{ECM}^{Mem} = \max(T_{\text{OL}}, T_{\text{nOL}} + T_{L1L2} + T_{L2L3} + T_{L3Mem})$$

Any deviation from non-overlap behavior in the hardware makes model non-optimistic!

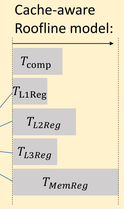
Most optimistic assumption: **full overlap** of data-related contributions

$$T_{ECM}^{Mem} = \max(T_{\text{OL}}, T_{\text{nOL}}, T_{L1L2}, T_{L2L3}, T_{L3Mem})$$

Fully optimistic (**light speed**) model, but not the same as Roofline:

Based on **measured** BW numbers: $T_i = \frac{V_i}{b_i^{meas}}$ $i \in \{ MemReg, \dots \}$

Cache-aware Roofline model:



Mixed model: **partial overlap** of data-related contributions

Example: no overlap at L1, full overlap of all other contributions

$$T_{ECM}^{Mem} = \max(T_{\text{OL}}, T_{\text{nOL}} + T_{L1L2}, T_{L2L3}, T_{L3Mem})$$

Notation for model predictions

$$\{T_{L1}^{ECM} \rceil T_{L2}^{ECM} \rceil T_{L3}^{ECM} \rceil T_{Mem}^{ECM}\}$$

Example: no-overlap model

$$\{ \max(T_{\text{OL}}, T_{\text{nOL}}) \rceil \quad \leftarrow L1$$
$$\max(T_{\text{OL}}, T_{\text{nOL}} + T_{L1L2}) \rceil \quad \leftarrow L2$$
$$\max(T_{\text{OL}}, T_{\text{nOL}} + T_{L1L2} + T_{L2L3}) \rceil \quad \leftarrow L3$$
$$\max(T_{\text{OL}}, T_{\text{nOL}} + T_{L1L2} + T_{L2L3} + T_{L3Mem}) \} \quad \leftarrow \text{Memory}$$

Outermost data in...

Example: AVX DP sum reduction w/ single accumulator on Broadwell EP CoD 2.3 GHz ($b_{L3Mem} = 32$ GB/s)

$\{ 6 \parallel 1 \mid 1 \mid 2 \mid 4.6 \}$ cy/8 iter

$\{ 6 \mid 6 \mid 6 \mid 7.6 \}$ cy/8 iter

→ Constant performance up to L3

```
do i=1,N
  s = s + a(i)
enddo
```

```
.label:
  vaddpd ymm1,[rdx+8*rsi]
  add rsi,4
  cmp rsi,rax
  jb .label
```

Multicore scaling and saturation

Optimistic assumption: Performance scaling is linear until a bandwidth bottleneck (e.g., b_{Mem}) is saturated → memory transfer time as lower limit

Runtime vs. cores (memory bottleneck):

$$T_{ECM}(n) = \max\left(\frac{T_{Mem}^{ECM}}{n}, T_{L3Mem}\right)$$

Roofline bandwidth ceiling

Number of cores at saturation:

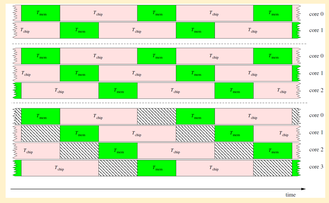
$$\Rightarrow \quad n_S = \left\lceil \frac{T_{ECM}^{Mem}}{T_{L3Mem}} \right\rceil$$

Example: AVX DP sum reduction from above

$\{ 6 \parallel 1 \mid 1 \mid 2 \mid 4.6 \}$ cy/8 iter

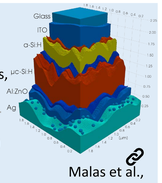
$\{ 6 \mid 6 \mid 6 \mid 7.6 \}$ cy/8 iter

$$\Rightarrow \quad n_S = \left\lceil \frac{7.6}{4.6} \right\rceil = 2$$



Application: Complex stencils

FDFD code solving Maxwell's Equations via THIIM for thin-film solar cells, **AVX2** vectorization via C intrinsics, **complex arithmetic**

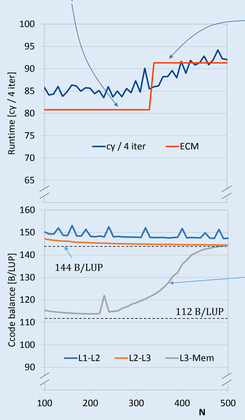


Malas et al., IPDPS 2016

```
// Hy_x = Chy_tx * Hy_x - Chy_x * (T(Ex_y + 
// Ex_z) - D(Ex_y + Ex_z) ) + Hy_old
// 1 of 12 kernels, 3D layer condition (LC)
for ( int k = 1; k < N - 1; ++k ) {
 for ( int j = 1; j < N - 1; ++j ) {
  for ( int i = 1; i < N - 1; ++i ) {
   diffRe = Exy[k][j][2*i] - Exy[k - 1][j][2*i] +
            Exz[k][j][2*i] - Exz[k - 1][j][2*i];
   diffIm = Exy[k][j][2*i + 1] - Exy[k - 1][j][2*i + 1] +
            Exz[k][j][2*i + 1] - Exz[k - 1][j][2*i + 1];
   asgn = Hyx[k][j][2*i] * tHyx[k][j][2*i] -
          Hyx[k][j][2*i + 1] * tHyx[k][j][2*i + 1] +
          HySrc[k][j][2*i] - cHyx[k][j][2*i] * diffRe +
          cHyx[k][j][2*i + 1] * diffIm;
   Hyx[k][j][2*i + 1] =
          Hyx[k][j][2*i] * tHyx[k][j][2*i + 1] +
          Hyx[k][j][2*i + 1] * tHyx[k][j][2*i] +
          HySrc[k][j][2*i + 1] - cHyx[k][j][2*i] * diffIm -
          Hyx[k][j][2*i + 1] * diffRe;
   Hyx[k][j][2*i] = asgn;
} } }
```

LC fulfilled ($B_C^{Mem} = 112$ B/LUP) $\{ 10 \parallel 8 \mid 18 \mid 18 \mid 36.8 \}$ cy/4 iter

LC broken ($B_C^{Mem} = 144$ B/LUP) $\{ 10 \parallel 8 \mid 18 \mid 18 \mid 47.3 \}$ cy/4 iter



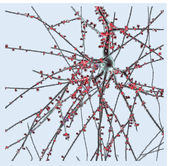
Intel "Haswell" Xeon E5-2695v3 @ 2.3 GHz Serial runtime predicted with ~5% accuracy

- **Limited** benefit from **spatial blocking**
- **Temporal blocking** is main optimization opportunity

Validation by data traffic measurements

- Predicted failure of 3D LC at N ≈330 observed
- Data volumes within 5% of prediction

Application: Blue Brain Project kernels



What is the computational cost of a synapse?
→ Study performance of brain simulation code via mini-apps
→ **SSE4.2** vectorization by compiler

Ewart et al., ISC 2017

Case 1: Synaptic current kernel

```
for(_iml = 0; _iml < _cntml; ++_iml) {
  _nd_idx = _ni[_iml]; _v = _vec_v[_nd_idx];
  mggate[_iml] = 1.0 / ( 1.0 + exp ( -0.062 *  _v )*(mg[_iml]/3.57) );
  g_AMPA[_iml] = gmax * ( B_AMPA[_iml] - A_AMPA[_iml] );
  g_NMDA[_iml] = gmax * ( B_NMDA[_iml] - A_NMDA[_iml] )*mggate[_iml];
  i_AMPA[_iml] = g_AMPA[_iml] * ( _v - e[_iml] );
  i_NMDA[_iml] = g_NMDA[_iml] * ( _v - e[_iml] );
  i[_iml] = i_AMPA[_iml] + i_NMDA[_iml];
  _g[_iml] = g_AMPA[_iml] + g_NMDA[_iml];
  _rhs[_iml] = i[_iml]; _mfact =  1.e2/(_nd_area[area_indices[_iml]]);
  _g[_iml] *=  _mfact; _rhs[_iml] *= _mfact;
  _vec_shadow_rhs[_iml] = _rhs[_iml]; _vec_shadow_d[_iml] = _g[_iml];
}
```

Challenges: `exp()` function call, some divides, some indirect accesses, integer register spill (many streams, call ABI restricts untouched registers)

- Streaming kernel, adjust data volume via knowledge about structure of index arrays (4% correction)
- Strong intra-iteration dependency chain
- `exp()` overhead (thruput/latency) measured by microbenchmark, added to ECM

	"Ivy Bridge" E5-2660v2	"Haswell" E5-2695v3
Throughput ass.	$\{ 32.5 \parallel 9.5 \mid 6.5 \mid 6.5 \mid 11.5 \}$ → $T_{\text{Mem}}^{\text{ECM}} = 34$ cy/iter	$T_{\text{Mem}}^{\text{ECM}} = 38.9$ cy/iter
CP ass.	$\{ 49 \parallel 9.5 \mid 6.5 \mid 6.5 \mid 11.5 \}$ → $T_{\text{Mem}}^{\text{ECM}} = 49$ cy/iter	$T_{\text{Mem}}^{\text{ECM}} = 50$ cy/iter
Measured	48.7 cy/iter	39.4 cy/iter

IVY close to CP prediction, HSW data bound!

Still saturating @ 3-5 cores on both CPUs!

Case 2: Sodium ion channel (NaTs2_t state)

```
for(_iml = 0; _iml < _cntml; ++_iml) {
  _nd_idx = _ni[_iml]; v = _vec_v[_nd_idx];
  mAlpha[_iml] = ( 0.182 * ( v + 32.0 ) ) /
                 ( 1.0 - ( exp ( - ( v + 32.0 ) / 6.0 ) ) ) ;
  mBeta[_iml] = ( 0.124 * ( - v - 32.0 ) ) /
                ( 1.0 - ( exp ( - ( - v - 32.0 ) / 6.0 ) ) ) ;
  mInf[_iml] = mAlpha[_iml] / ( mAlpha[_iml] + mBeta[_iml] ) ;
  mTau[_iml] = ( 1.0 / ( mAlpha[_iml] + mBeta[_iml] ) ) / 2.95 ;
    hAlpha[_iml] = ( - 0.015 * ( v + 60.0 ) ) /
                   ( 1.0 - ( exp ( ( v + 60.0 ) / 6.0 ) ) ) ;
  hBeta[_iml] = ( - 0.015 * ( - v - 60.0 ) ) /
                ( 1.0 - ( exp ( ( - v - 60.0 ) / 6.0 ) ) ) ;
  hInf[_iml] = hAlpha[_iml] / (hAlpha[_iml]+hBeta[_iml]) ;
  hTau[_iml] = (1.0/(hAlpha[_iml] + hBeta[_iml] ) ) / 2.95 ;
   m[_iml] = m[_iml] + (1. - exp(dt*( - 1.0 / mTau[_iml])))*
    (- ( mInf[_iml] / mTau[_iml] ) /
    ( - 1.0 / mTau[_iml] ) - m[_iml]) ;
   h[_iml] = h[_iml] +
     (1. - exp(dt*( - 1.0 / hTau[_iml])))*
     (- ( hInf[_iml] / hTau[_iml] ) /
     ( - 1.0 / hTau[_iml] ) - h[_iml]) ;
}
```

Challenges: `exp()` & divides dominate, no strong dependency chain, small data volume

"Ivy Bridge" E5-2660v2 : Full throughput T_{OL}

$\{ 140 \parallel 9.25 \mid 5.15 \mid 5.15 \mid 9 \}$ → $T_{\text{Mem}}^{\text{ECM}} \geq 140$ cy

28.6 cy

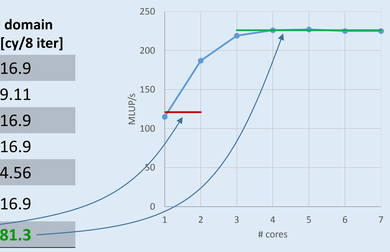
Saturation @ 16 cores!

Measurement: 191 cy

Application: Conjugate Gradient solver

2D 5-pt FD Poisson problem, Dirichlet BCs, matrix free, 40000 × 1000 grid
CPU: **Haswell E5-2695v3 CoD mode**

while(α_0 < tol):	ECM [cy/8 iter]	T_{Mem}^{ECM} [cy/8 iter]	n_s [cores]	Full domain limit [cy/8 iter]
$\vec{v} = A\vec{p}$	$\{ 8 \parallel 4 \mid 6.7 \mid 10 \mid 16.9 \}$	37.6	3	16.9
$\lambda = \alpha_0 / \langle \vec{v}, \vec{p} \rangle$	$\{ 2 \parallel 2 \mid 2.7 \mid 4 \mid 9.1 \}$	17.8	2	9.11
$\vec{x} = \vec{x} + \lambda\vec{p}$	$\{ 2 \parallel 4 \mid 6 \mid 16.9 \}$	29.0	2	16.9
$\vec{r} = \vec{r} - \lambda\vec{v}$	$\{ 2 \parallel 4 \mid 6 \mid 16.9 \}$	29.0	2	16.9
$\alpha_1 = \langle \vec{r}, \vec{r} \rangle$	$\{ 2 \parallel 2 \mid 1.3 \mid 2 \mid 4.6 \}$	9.90	3	4.56
$\vec{p} = \vec{r} + \frac{\alpha_1}{\alpha_0}\vec{p}$, $\alpha_0 = \alpha_1$	$\{ 2 \parallel 4 \mid 6 \mid 16.9 \}$	29.0	2	16.9
	Sum	152		81.3



- Multi-loop code well represented
- Single core performance predicted with 5% error
- Saturated performance predicted with < 0.5% error
- Saturation point predicted approximately
- **Future work**: Include GS preconditioner → HPCG modeling

Overlap assumptions for current architectures

Intel Skylake SP: no overlap in data transfers, victim L3

$$T_{ECM}^{Mem} = \max\left(T_{OL}, T_{nOL} + T_{L1L2} + T_{L2L3} + T_{L2Mem} + T_{L3Mem}\right)$$

Intel Xeon ≤ Broadwell EP: no overlap

$$T_{ECM}^{Mem} = \max(T_{OL}, T_{nOL} + T_{L1L2} + T_{L2L3} + T_{L3Mem})$$

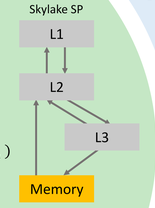
IBM Power8: overlap at L1

$$T_{ECM}^{Mem} = \max\left(T_{OL}, T_{nOL}, T_{L1L2}, T_{L2L3} + T_{L2Mem} + T_{L3Mem}\right)$$

AMD Zen (Epyc): full overlap

$$T_{ECM}^{Mem} = \max\left(T_{nOL}, T_{L1L2}, T_{L2L3}, T_{L3Mem}, T_{OL}\right)$$

preliminary



Poster artifacts

https://tiny.cc/ECM-SC18-AD

J. Hofmann et al: *On the accuracy and usefulness of analytic energy models for contemporary multicore processors*. DOI: 10.1007/978-3-319-92040-5_2

J. Hammer et al.: *Kerncraft: A Tool for Analytic Performance Modeling of Loop Kernels*. DOI: 10.1007/978-3-319-56702-0_1

G. Hager et al.: *Exploring performance and power properties of modern multicore chips via simple machine models*. DOI: 10.1002/cpe.3180

H. Stengel et al.: *Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model*. DOI: 10.1145/2751205.2751240

T. M. Malas et al.: *Optimization of an electromagnetics code with multicore wavefront diamond blocking and multi-dimensional intra-tile parallelization*. DOI: 10.1109/IPDPS.2016.87

T. Ewart et al.: *Neuromapp: A Mini-application Framework to Improve Neural Simulators*. DOI: 10.1007/978-3-319-58667-0_10

Layer Condition Calculator: tiny.cc/LayerConditions

Kerncraft: tiny.cc/kerncraft

Pycachesim: tiny.cc/pycachesim

IACA: tiny.cc/IACA

OSACA: tiny.cc/OSACA

LIKWID tools: tiny.cc/LIKWID

Supported by

Bundesministerium für Bildung und Forschung

Embedded MM content

https://tiny.cc/ECM-SC18