Application Knowledge Required: Analytical Performance Modeling and its application to SpMV

Georg Hager and Gerhard Wellein

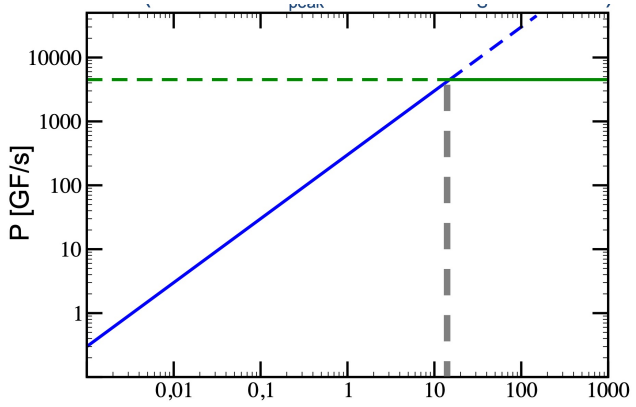
Erlangen National High Performance Computing Center (NHR@FAU)

Department of Computer Science

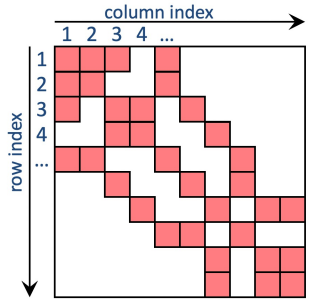
Friedrich-Alexander-Universität Erlangen-Nürnberg

Agenda

1. Analytical, resource-based, first-principles performance models



2. Performance Modelling of Sparse Matrix Vector Multiplication (SpMV)



Analytical Performance Model?

A “simple” mathematical formula predicting the performance or runtime of a program using various input data

Analytical, Resource-Based Performance Model?

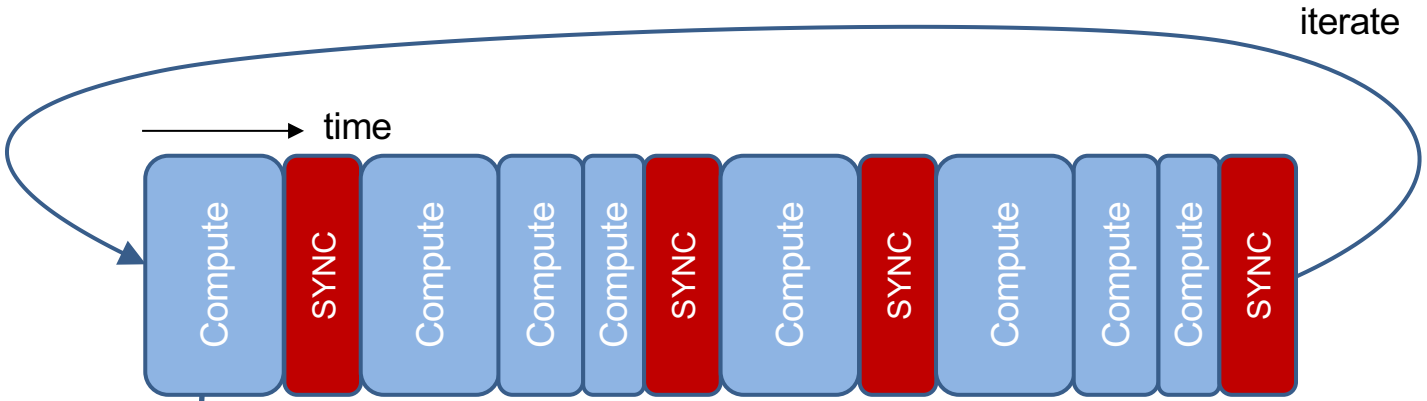
A “simple” mathematical formula predicting the performance or runtime of a program using hardware resource limits and code requirements

Analytical, Resource-Based, First-Principles Performance Model?

a.k.a. white-box models

A mathematical representation of hardware-software interaction based on simplified machine and application models, which predicts the performance or runtime of a program using hardware resource limits and code requirements

Typical Solver Structure – General runtime model




```
Parallel_for(i=0..N) { // N>>1
  update(data);
}
```

"Steady state"
- Repetitive
- Negligible startup/wind-down overhead

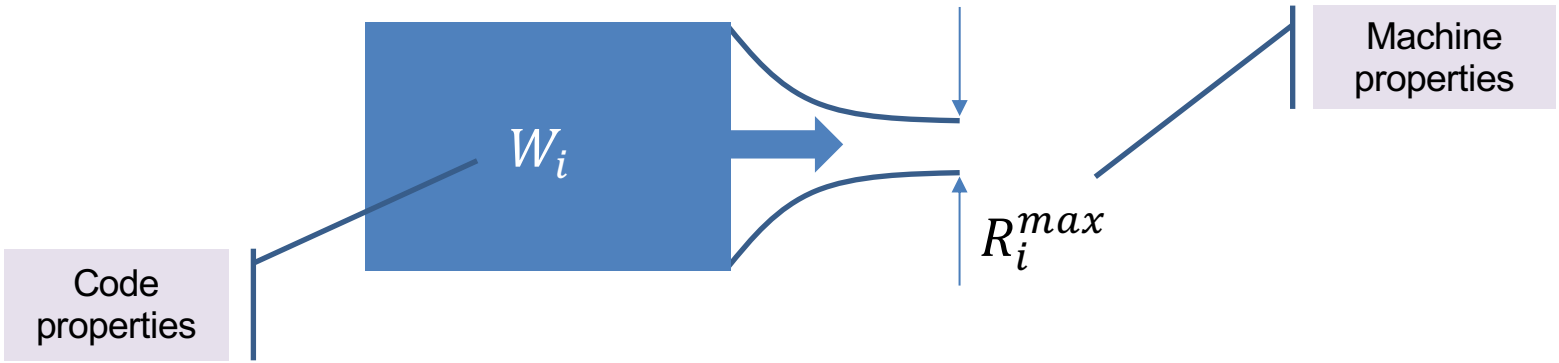
Runtime model: $T = f(\$\mathbf{STUFF}, \$\mathbf{HARDWARE})$

Performance model: $\mathrm{P} = \frac{\$\boldsymbol{WORK}}{\boldsymbol{T}}$

Resource bottlenecks – simplified model

What is the maximum performance when limited by a bottleneck?

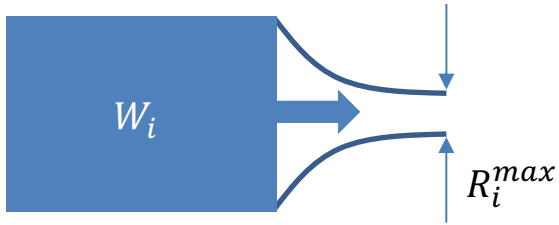
- Resource bottleneck i delivers resources at maximum rate R_i^{max}
- W_i = needed amount of resources



Resource bottlenecks

Minimum runtime due to bottleneck i:

$$T_i = \frac{W_i}{R_i^{max}} + \lambda_i$$



- Multiple bottlenecks?
  → multiple minimum runtimes: $T_{\min} = f(T_1, \ldots T_n)$

- Overall performance:

$$P_{\max} = \frac{W}{T_{\min}}$$

Simple two-bottleneck models for single loops

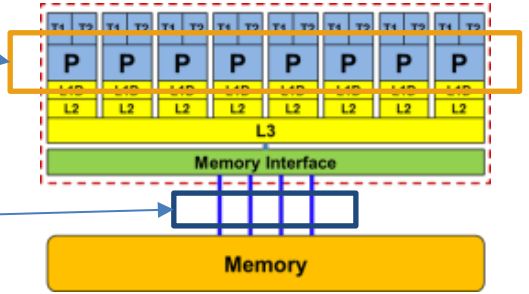
```
#pragma omp parallel for
for(i=0; i<10^7; ++i)
  a[i] = a[i] + s * c[i];
```

$W_{flops} = 2{\times}10^7$ flops

$W_{BW} = 3{\times}8{\times}10^7$ bytes

$R^{max}_{flops} = 192 \frac{\text{Gflops}}{\text{s}}$

$R^{max}_{BW} = 40 \frac{\text{Gbyte}}{\text{s}}$



8-core CPU
(3 GHz Intel Sandy Bridge)

$$T_{flops} = \frac{2{\times}10^7 \text{ flops}}{192 \frac{\text{Gflops}}{\text{s}}} = 104\ \mu s$$

$$T_{BW} = \frac{2.4{\times}10^8 \text{ bytes}}{40 \frac{\text{Gbyte}}{\text{s}}} = 6.0 \text{ ms}$$

!!! Note: $\lambda_1 = \lambda_2 = 0$!!!

Bottleneck models for single loops

How do we reconcile the multiple bottlenecks?
I.e., what is the functional form of $f(T_1, \ldots T_n)$?

→ pessimistic (no overlap): $f(T_1, \ldots T_n) = \sum_i T_i$
→ optimistic (full overlap): $f(T_1, \ldots T_n) = \max(T_1, \ldots T_n)$

Our example (two bottlenecks: flops + data transfer; $\lambda_i = 0$):

$$\boldsymbol{T_{\min}} = \boldsymbol{max}\left(\boldsymbol{T_{flops}}, \boldsymbol{T_{BW}}\right) = 6 \text{ ms}$$

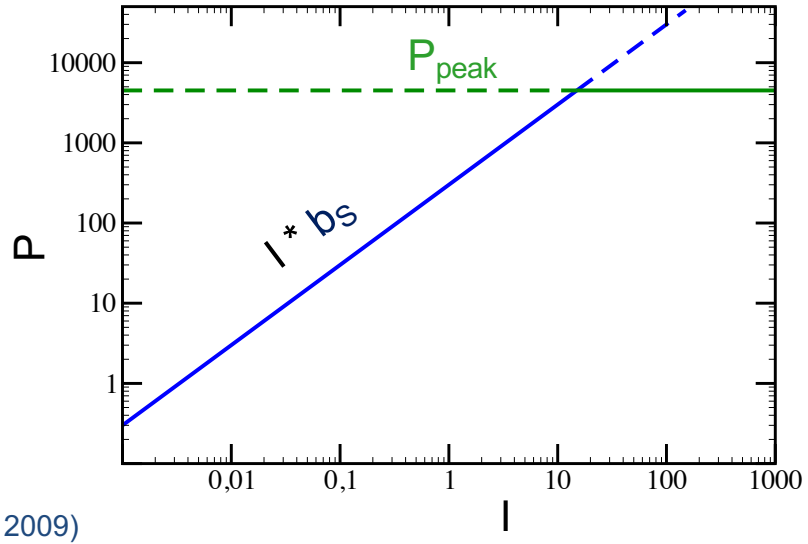
Maximum performance ("light speed"): $P_{upper} = \frac{2 \times 10^7}{6.0 \times 10^{-3}} \frac{\text{flops}}{\text{s}} = 3.3 \text{ Gflop/s}$

Roofline Performance Model (RLM)

- $T_{\min} = max(T_{flops}, T_{BW})$ ($\leftrightarrow P_{\max} = \frac{\#\text{Flops}}{T_{min}}$)

- Roofline Model: $P_{\max} = \min(P_{peak}, I * b_S)$
  - Peak Performance: P_{peak}
  - Memory bandwidth: b_S [Byte/s]
  - Computational Intensity: I [Flop/Byte]

- Single chip or compute node!!!

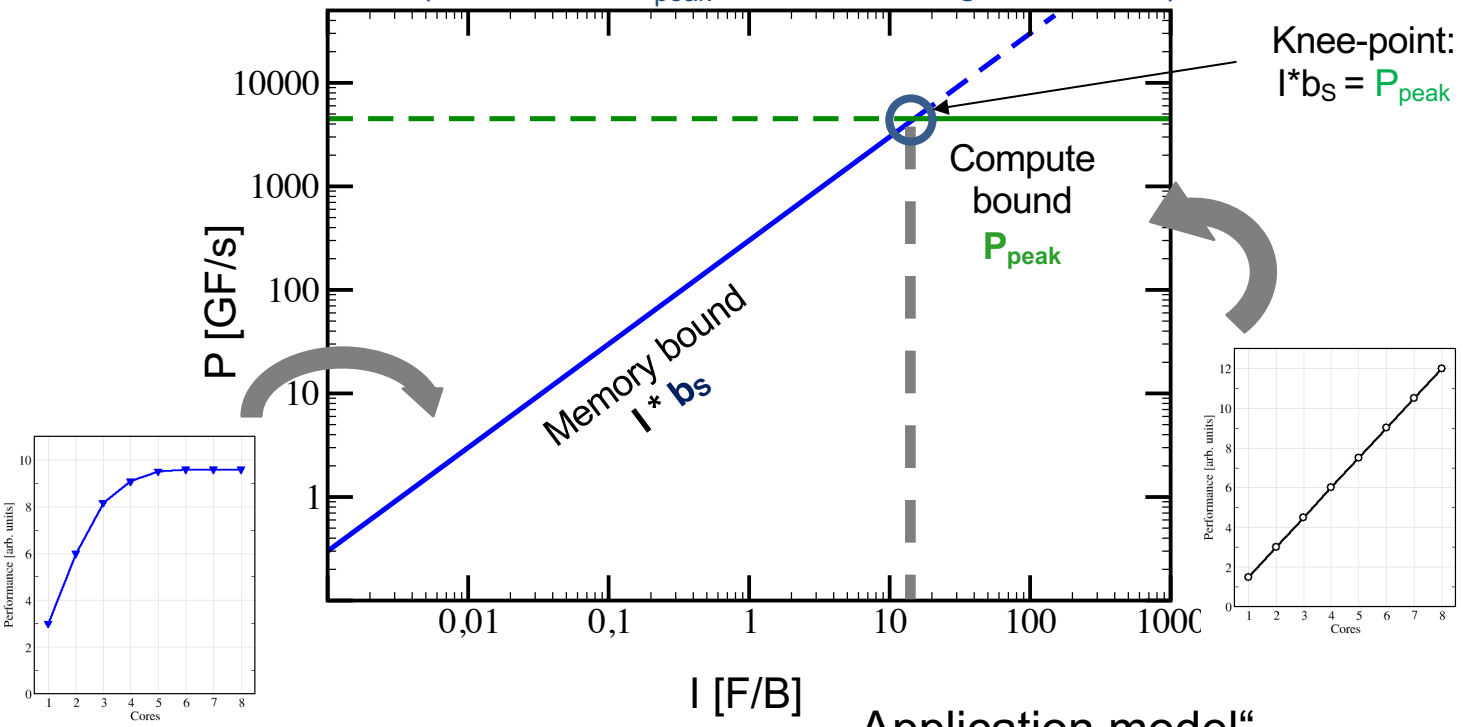


R.W. Hockney and I.J. Curington. Parallel Computing 10, 277-286 (1989).
S. Williams. UCB Tech. Report No. UCB/EECS-2008-164. PhD thesis (2008)
S. Williams, A. Waterman, and D. Patterson. 2009. Commun. ACM 52, 4 (April 2009)

Roofline Model – Characteristic behaviour

Machine model with P_{peak}=4,5 TF/s and b_S=300 GB/s

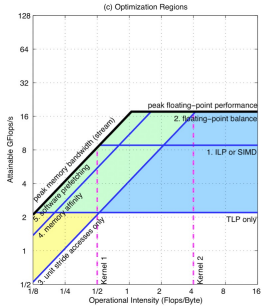
(Deflt Blue: P_{peak}=3 TF/s and b_S=256 GB/s)



„Application model“

Analytic modelling – where we are: Examples

Roofline Model

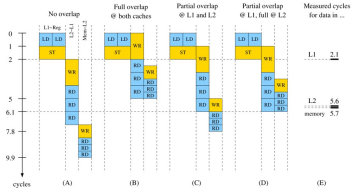


S. Williams, A. Waterman, D. Patterson (2009)
DOI:10.1145/1498765.1498785

Energy:
J. W. Choi, D. Bedard, R. Fowler, R. Vuduc (2013) DOI: 10.1109/IPDPS.2013.77.

Cache-Aware:
A. Ilic, F. Pratas, L. Sousa (2014)
DOI: 10.1109/L-CA.2013.6.

Execution Cache Memory Model



Hager, Treibig, Habich, Wellein (2016)
DOI: 10.1002/cpe.3180.

Power/Energy:
Hofmann, Hager, Fey (2018).
https://doi.org/10.1007/978-3-319-92040-5_2

$$\frac{W}{f(T_1, \ldots, T_n)}$$

Communication models

LogP and variants

Proven/useful for

- CPU-type
- GPU-type
- Vector-type

Data + Flops/Instructions – Throughputs / Latencies

RLM / Analytical modelling – Use Cases

- Typical code / application structures
  - “Streaming kernels” – consecutive data access
  - Dense Matrix Kernels (incl. Tensor Operations & Tall&Skinny)
  - Stencil Kernels
  - …

- Insights:
  - Hardware bottleneck
  - Code quality - is there room for performance improvement? How much?
  - Estimating code optimization parameters, e.g. layer conditions, tiling sizes
  - …

NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

Performance Modelling of Sparse Matrix Vector Multiplication (SpMV)

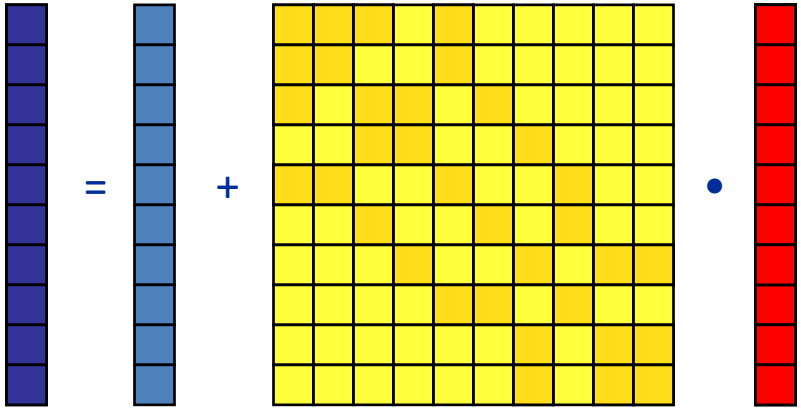
RLM Performance Modelling?

Optimal Performance of SpMV?

Motivation

Algorithm 2. HPCG

1: **while** $k \leq iter$ & $r_{norm}/r_0 > tol$ **do**
2: $z = MG(A,r)$
3: $oldrtz = rtz$
4: $rtz = \langle r,z \rangle$
5: $\beta = rtz/oldrtz$
6: $p = \beta * p + z$
7: $Ap = A * p$
8: $pAp = \langle p,Ap \rangle$
9: $\alpha = rtz/pAp$
10: $x = x + \alpha * p$
11: $r = r - \alpha * Ap$
12: $r_{norm} = \langle r,r \rangle$
13: $r_{norm} = sqrt(r_{norm})$
14: $k++$

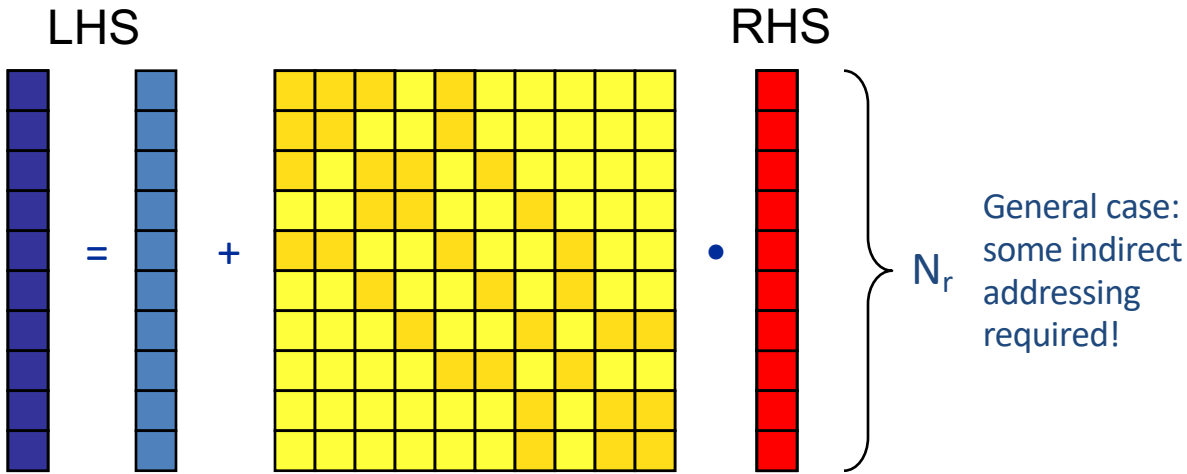


Assume memory bound: $P_{\max} = I * b_S$

How much data is (at least) loaded from main memory? → Intensity

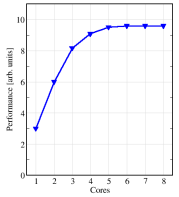
Sparse Matrix Vector Multiplication (SpMV)

- Key ingredient in numerous sparse linear algebra solvers
- Store only N_{nz} nonzero elements of matrix and RHS, LHS vectors with N_r (number of matrix rows) entries
- "Sparse": $N_{nz} \sim N_r$
- Average number of nonzeros per row: $N_{nzr} = N_{nz}/N_r$



SpMV characteristics

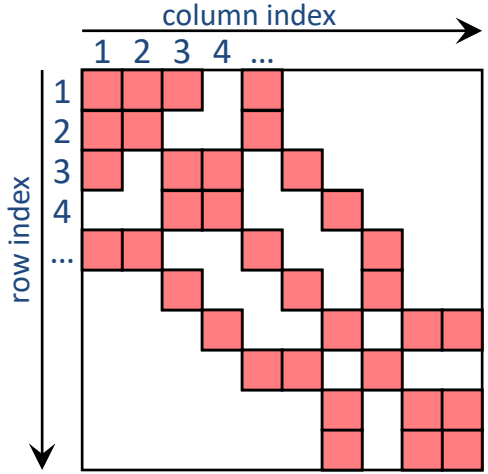
- For large problems, SpMV is inevitably memory-bound
  - Intra-socket saturation effect on modern multicores



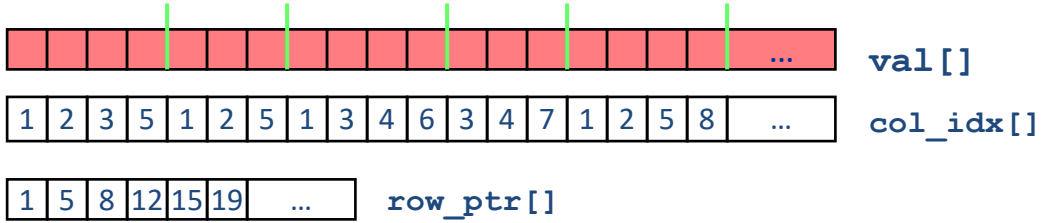
- SpMV is easily parallelizable in shared and distributed memory
  - Load balancing
  - Communication overhead

- Data storage format is crucial for performance properties
  - Most useful general format on CPUs: Compressed Row Storage (CRS)
  - Depending on compute architecture

CRS matrix storage scheme



- **`val[]`** stores all the nonzeros (length N_{nz})
- **`col_idx[]`** stores the column index of each nonzero (length N_{nz})
- **`row_ptr[]`** stores the starting index of each new row in **`val[]`** (length: N_r)



Case study: Sparse matrix-vector multiply

- Strongly memory-bound for large data sets
  - Mainly streaming data access mixed with partially indirect access:

```
!$OMP parallel do
do i = 1,Nr
 do j = row_ptr(i), row_ptr(i+1) - 1
  C(i) = C(i) + val(j) * B(col_idx(j))
 enddo
enddo
!$OMP end parallel do
```

  - Usually many spMVs required to solve a problem
- Irregular data access to `B(col_idx(j))`
- What is the computation intensity I (or comp. balance $B_C = I^{-1}$)??

SpMV node performance model – CRS (1)

```
do i = 1, Nr                                    real*8    val(Nnz)
 do j = row_ptr(i), row_ptr(i+1) - 1            integer*4 col_idx(Nnz)
  C(i) = C(i) + val(j) * B(col_idx(j))          integer*4 row_ptr(Nr)
 enddo                                          real*8    C(Nr)
enddo                                           real*8    B(Nc)
```

Min. load traffic [B]: $(8 + 4)\,N_{nz} + (4 + 8)N_r + 8\,N_c$

Min. store traffic [B]: $8\,N_r$

Total FLOP count [F]: $2\,N_{nz}$

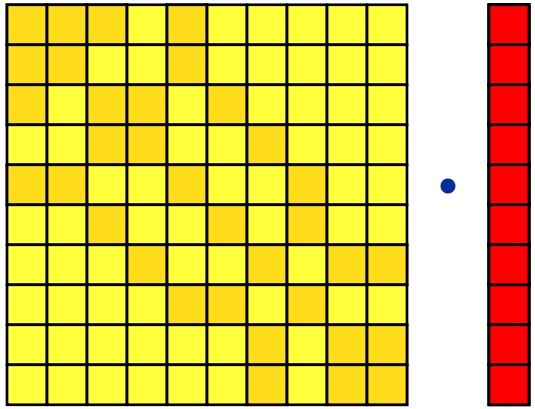
$$B_{C,min} = \frac{12\,N_{nz} + 20\,N_r + 8\,N_c}{2\,N_{nz}}\frac{B}{F} = \frac{12 + 20/N_{nzr} + 8/N_{nzc}}{2}\frac{B}{F}$$

Nonzeros per row ($N_{nzr} = {}^{N_{nz}}/_{N_r}$) or column ($N_{nzc} = {}^{N_{nz}}/_{N_c}$)

Lower bound for code balance: $B_{C,min} \geq 6\,\frac{\mathrm{B}}{\mathrm{F}}$ → $I_{\max} \leq \frac{1}{6}\frac{\mathrm{F}}{\mathrm{B}}$

SpMV node performance model – CRS (2)

```
do i = 1, Nr
 do j = row_ptr(i), row_ptr(i+1) - 1
  C(i) = C(i) + val(j) * B(col_idx(j))
 enddo
enddo
```



$$B_{C,min} = \frac{12 + 20/N_{nzr} + 8/N_{nzc}}{2} \frac{B}{F}$$

$$B_C(\alpha) = \frac{12 + 20/N_{nzr} + 8\,\alpha}{2} \frac{B}{F}$$

Parameter (α) quantifies additional traffic for `B(:)` (irregular access):

$$\alpha \geq 1/N_{nzc}$$

$$\alpha N_{nzc} \geq 1$$

Consider square matrices: $N_{nzc} = N_{nzr}$ and $N_c = N_r$

Note: $B_C\left(1/N_{nzr}\right) = B_{C,min}$

The "α effect"

CRS code balance

- α quantifies the traffic for loading the Right Hand Side (RHS) vector
  - $\alpha = 0$ → RHS is in cache (RHS << cache size)
  - $\alpha = 1/N_{nzr}$ → RHS loaded once
  - $\alpha = 1$ → no cache
  - $\alpha > 1$ → Houston, we have a problem!

$$B_C(\alpha) = \frac{12 + 20/N_{nzr} + 8\,\alpha}{2}\,\frac{B}{F}$$

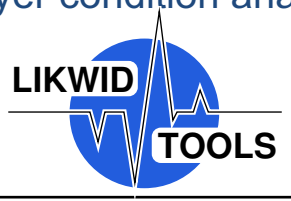
$$= \left(6 + 4\,\alpha + \frac{10}{N_{nzr}}\right)\frac{B}{F}$$

→ α * N_{nzr} ←→ #times RHS vector is loaded from main memory

Can we predict α?

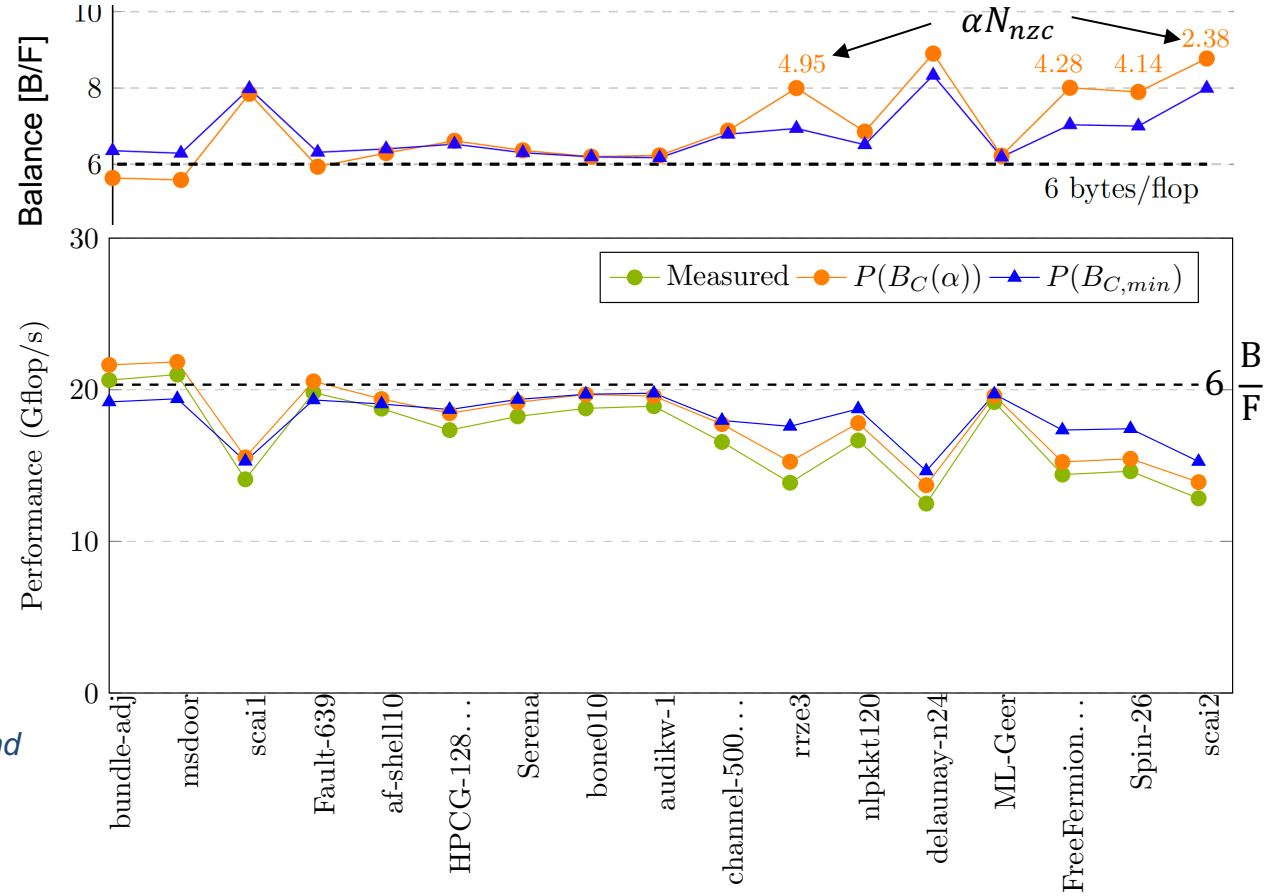
- Not in general
- Simple cases (banded, block-structured): Similar to layer condition analysis

→ Determine α by measuring the actual memory traffic
(→ measured code balance B_C^{meas})



SpMV node performance model – CLX-AP

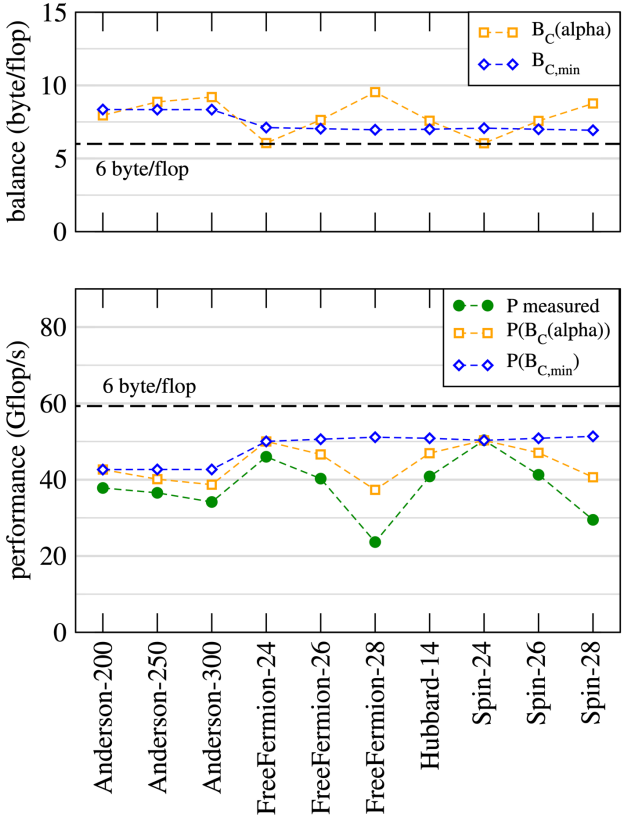
Intel Xeon Platinum 9242
24c@2.8GHz (turbo)
$b_S = 122\ GB/s$



C. L. Alappat et al.: *ECM modeling and performance tuning of SpMV and Lattice QCD on A64FX.*

SpMV node performance model – more data

CRS



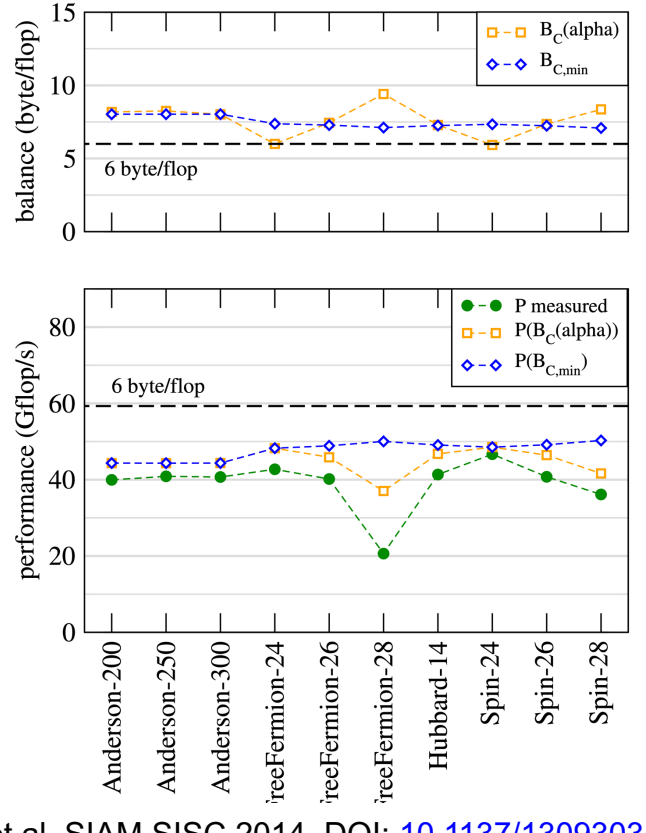
Different processor!!!
Different matrices!!!

Add. data layout

Intel Ice Lake 8360 Y

$b_S = 356\ \mathrm{GB/s}$

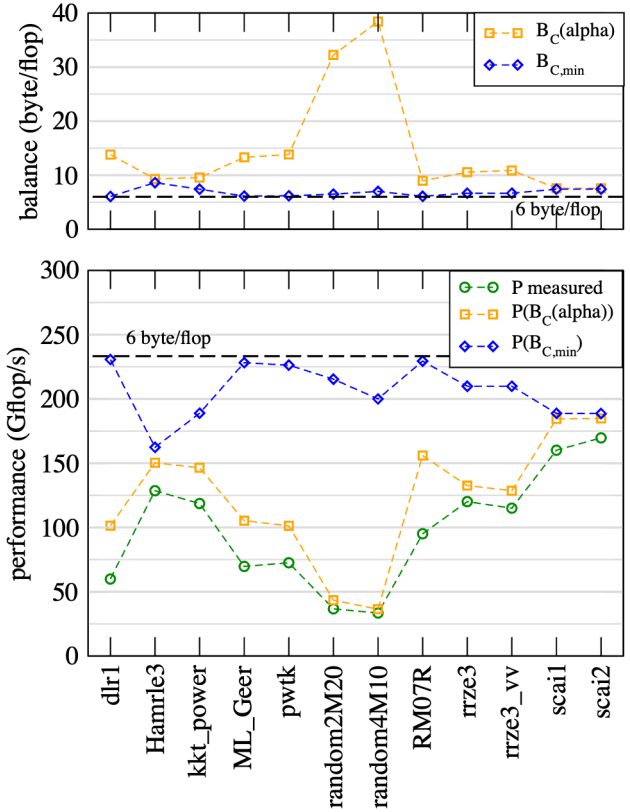
SELL-32-128 [1]



[1]M. Kreutzer et al, SIAM SISC 2014, DOI: 10.1137/130930352

SpMV node performance model – GPU

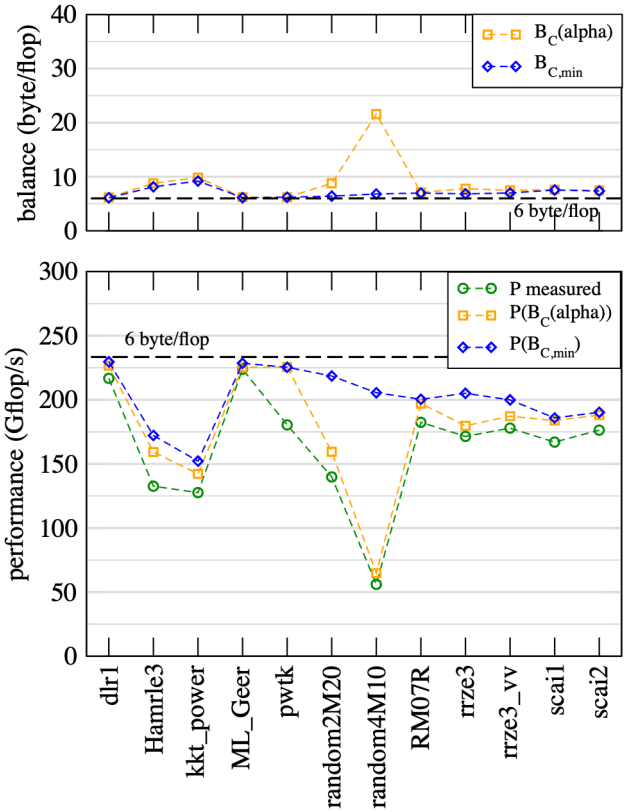
CRS (1 thread per row)



NVIDIA Ampere A100

$b_S = 1400\ \mathrm{GB/s}$

SELL-32-128[1]



[1]M. Kreutzer et al, SIAM SISC 2014, DOI: 10.1137/130930352

Summary & Outlook

- Qualitative modelling of SpMV kernels on node level
- CPU and GPU
- Impact of irregular access can be quanitfied
- Consequences:
  - Large α → try bandwidth reduction for matrix (e.g. RCM)
  - Core bottlenecks can be identfied, e.g. CRS on GPU or A64FX
  - Quality of black box libraries can be tested
  - Whenever you do SpMV: check $B_{C,min} \geq 6 \frac{\mathrm{B}}{\mathrm{F}}$

- Tomorrow: Improving code balance of SpMV based algorithms some old and a new idea