The Linear Algebra of Native Hardware Event Identification

Daniel Barry

ICL Lunch Talk

Feb. 5, 2021

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Problem to Solve!

- How do we define a PAPI preset event/metric using native events?

- Ex.) PAPI_SP_OPS (on Intel Skylake) =

FP_ARITH:SCALAR_SINGLE

*+ 2 * FP_ARITH:128B_PACKED_SINGLE*

*+ 4 * FP_ARITH:256B_PACKED_SINGLE*

*+ 8 * FP_ARITH:512B_PACKED_SINGLE*

FLOPs Presets

Architecture	PAPI_DP_OPS	PAPI_SP_OPS
Skylake	FP_ARITH:SCALAR_DOUBLE + 2*FP_ARITH:128B_PACKED_DOUBLE + 4*256B_PACKED_DOUBLE + 8*512B_PACKED_DOUBLE	FP_ARITH:SCALAR_SINGLE + 4*FP_ARITH:128B_PACKED_SINGLE + 8*256B_PACKED_SINGLE + 16*512B_PACKED_SINGLE
Broadwell	FP_ARITH:SCALAR_DOUBLE + 2*FP_ARITH:128B_PACKED_DOUBLE + 4*256B_PACKED_DOUBLE	FP_ARITH:SCALAR_SINGLE + 4*FP_ARITH:128B_PACKED_SINGLE + 8*256B_PACKED_SINGLE
Power9	PM_DP_QP_FLOP_CMPL	PM_SP_FLOP_CMPL

Credit: Barry *et al.*

Bandwidth Presets

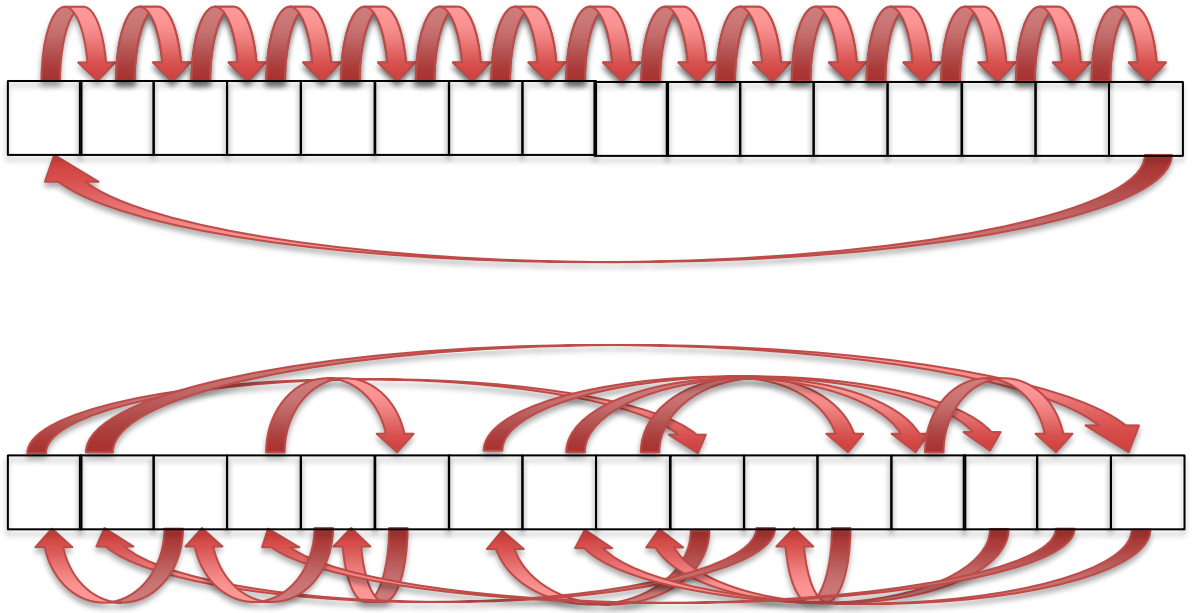
- Bandwidth (on Intel Skylake) =
  64*skx_unc_imc0::UNC_M_CAS_COUNT:WR:cpu=0
  + 64*skx_unc_imc1::UNC_M_CAS_COUNT:WR:cpu=0
  + 64*skx_unc_imc2::UNC_M_CAS_COUNT:WR:cpu=0
  + 64*skx_unc_imc3::UNC_M_CAS_COUNT:WR:cpu=0
  + 64*skx_unc_imc4::UNC_M_CAS_COUNT:WR:cpu=0
  + 64*skx_unc_imc5::UNC_M_CAS_COUNT:WR:cpu=0
  + 64*skx_unc_imc0::UNC_M_CAS_COUNT:RD:cpu=0
  + 64*skx_unc_imc1::UNC_M_CAS_COUNT:RD:cpu=0
  + 64*skx_unc_imc2::UNC_M_CAS_COUNT:RD:cpu=0
  + 64*skx_unc_imc3::UNC_M_CAS_COUNT:RD:cpu=0
  + 64*skx_unc_imc4::UNC_M_CAS_COUNT:RD:cpu=0
  + 64*skx_unc_imc5::UNC_M_CAS_COUNT:RD:cpu=0
  + 64*skx_unc_imc0::UNC_M_CAS_COUNT:WR:cpu=18
  + 64*skx_unc_imc1::UNC_M_CAS_COUNT:WR:cpu=18
  + 64*skx_unc_imc2::UNC_M_CAS_COUNT:WR:cpu=18
  + 64*skx_unc_imc3::UNC_M_CAS_COUNT:WR:cpu=18
  + 64*skx_unc_imc4::UNC_M_CAS_COUNT:WR:cpu=18
  + 64*skx_unc_imc5::UNC_M_CAS_COUNT:WR:cpu=18
  + 64*skx_unc_imc0::UNC_M_CAS_COUNT:RD:cpu=18
  + 64*skx_unc_imc1::UNC_M_CAS_COUNT:RD:cpu=18
  + 64*skx_unc_imc2::UNC_M_CAS_COUNT:RD:cpu=18
  + 64*skx_unc_imc3::UNC_M_CAS_COUNT:RD:cpu=18
  + 64*skx_unc_imc4::UNC_M_CAS_COUNT:RD:cpu=18
  + 64*skx_unc_imc5::UNC_M_CAS_COUNT:RD:cpu=18

Bandwidth Presets

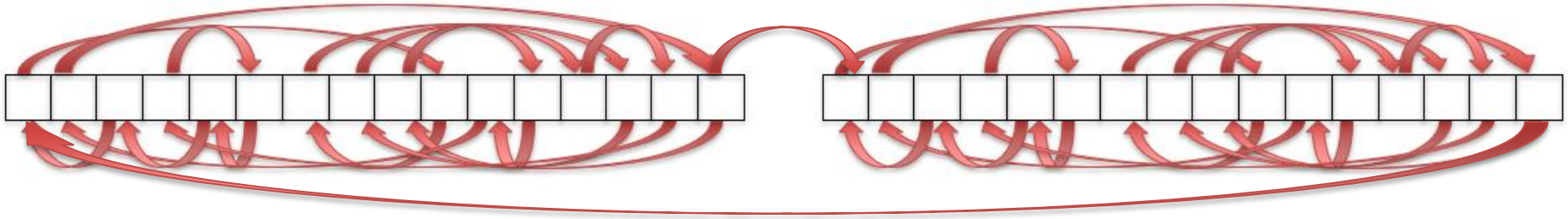
- Broadwell
  - *bdx_unc_imc[0|1|4|5]::UNC_M_CAS_COUNT:[RD|WR]:cpu=0*
- Skylake
  - *skx_unc_imc[0-5]::UNC_M_CAS_COUNT:[RD|WR]:cpu=[0|18]*
- POWER9
  - *pcp:::perfevent.hwcounters.nest_mba[0-7]_imc. PM_MBA[0-7]_[READ|WRITE]_BYTES.value: cpu[84|172]*

Benchmark Structure

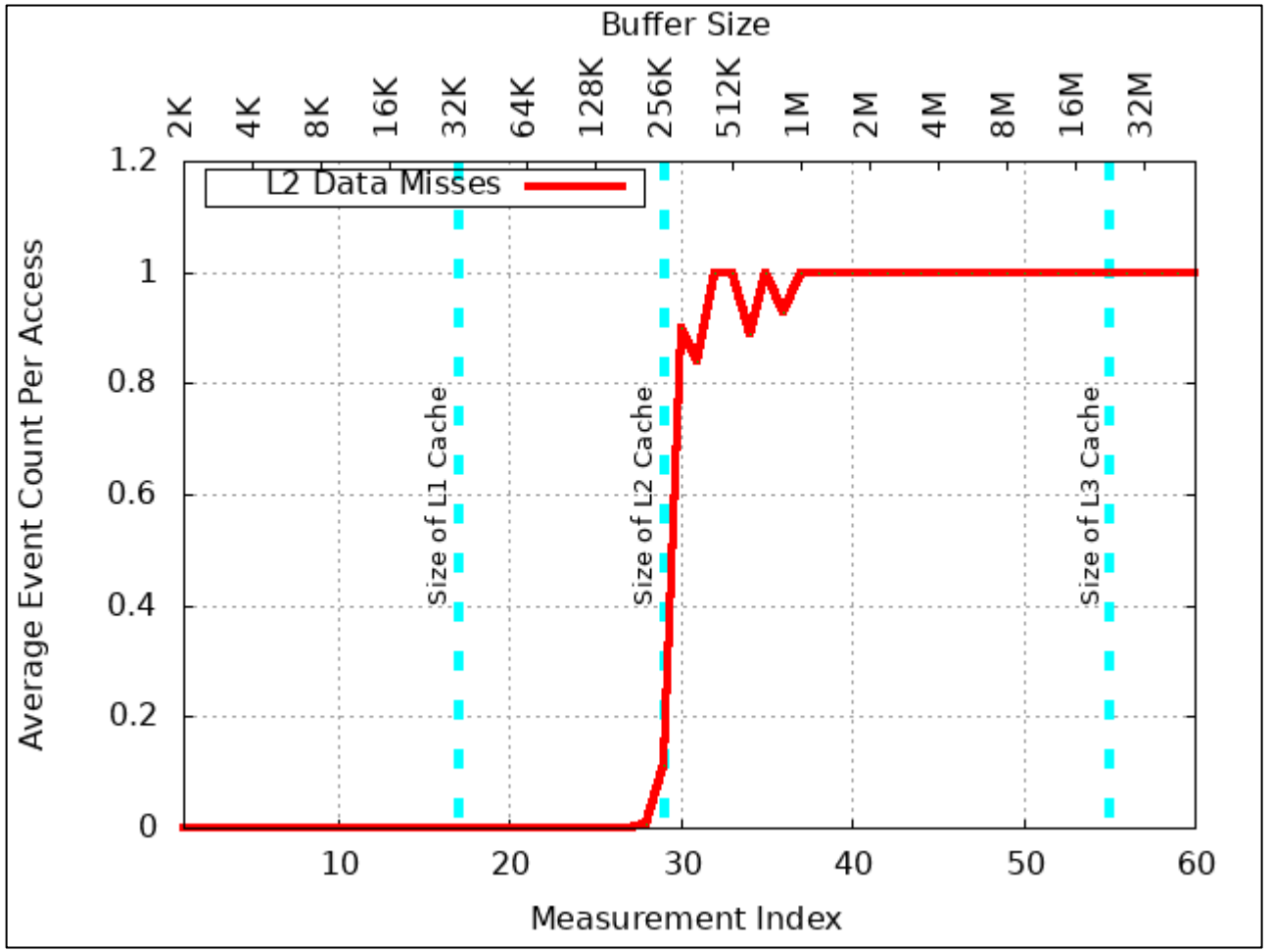
- Ex.) Pointer-chaining algorithm for Data Cache benchmarks.

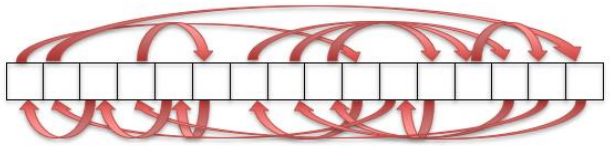


Benchmark Structure

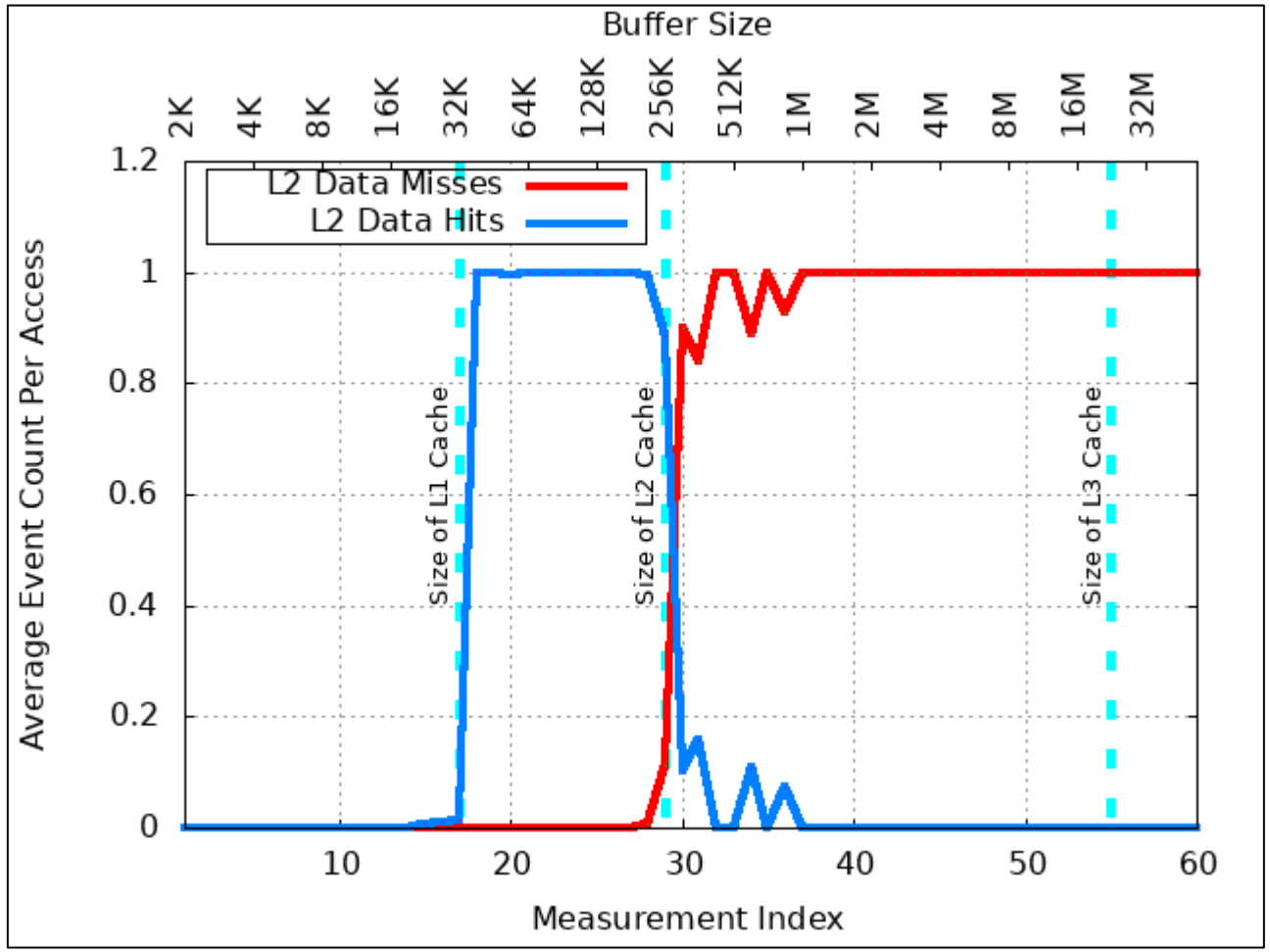


L2 Data Cache Misses – Intel Haswell

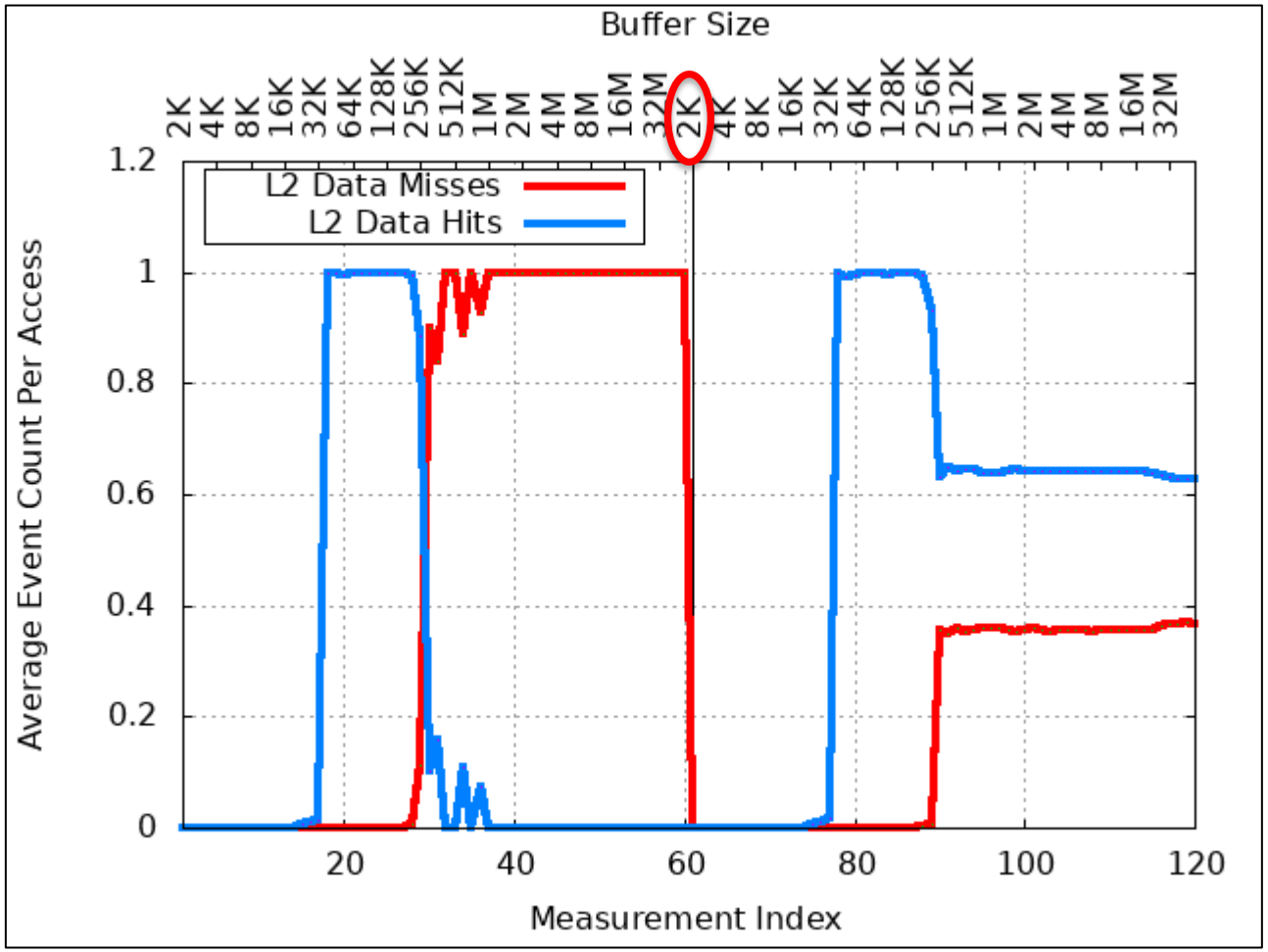




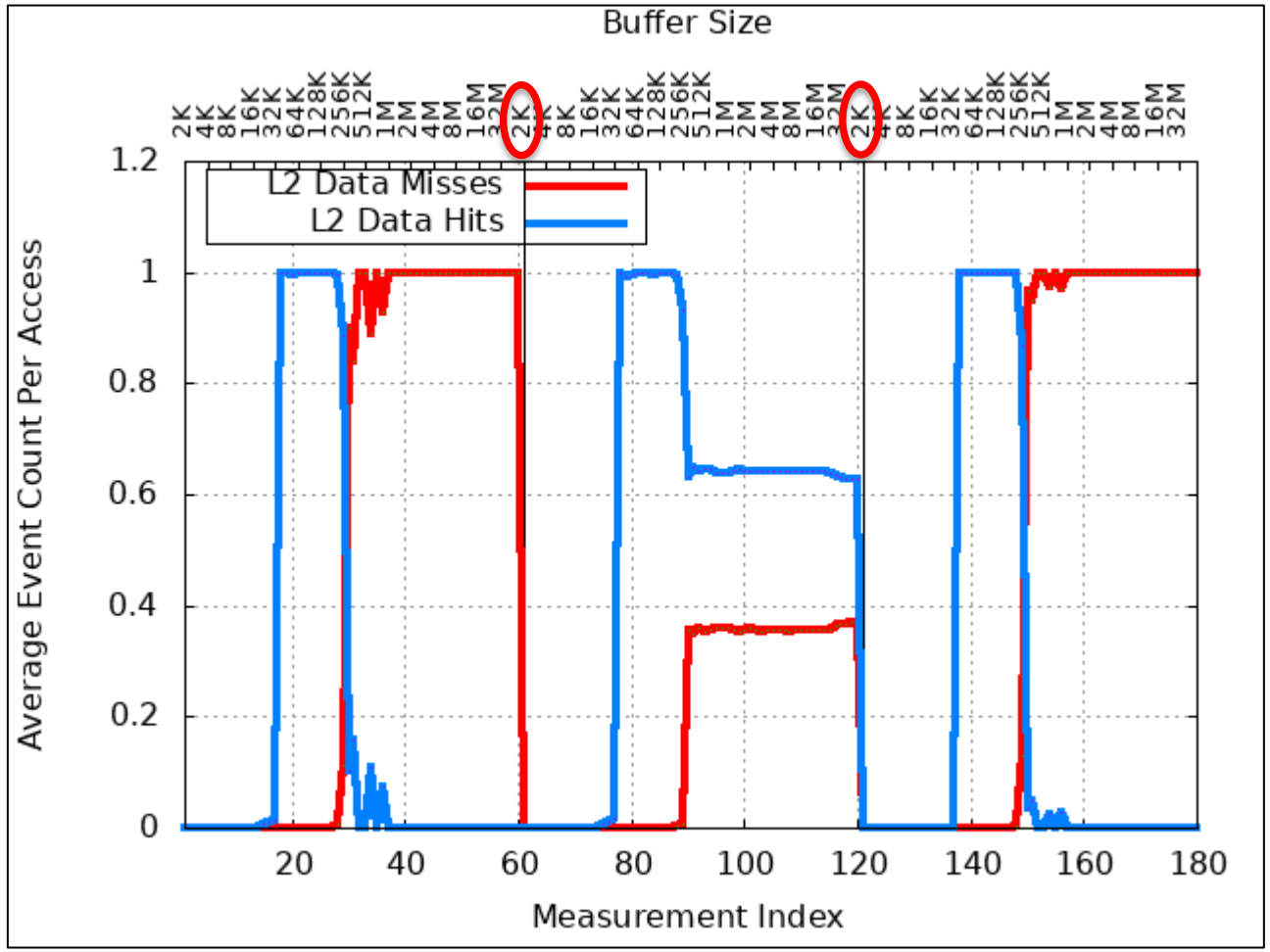
L2 Data Cache Misses – Intel Haswell



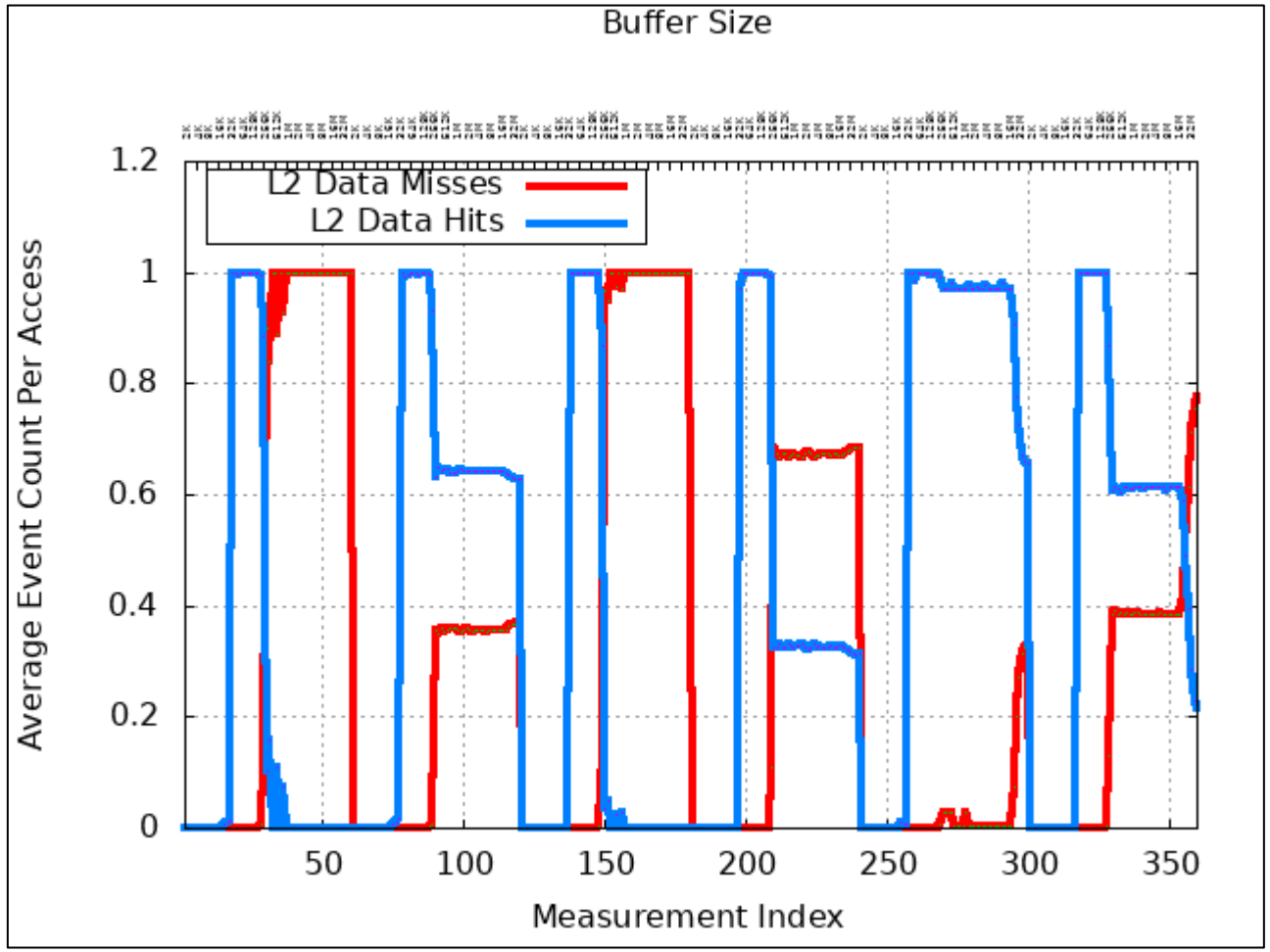
L2 Data Cache Misses – Intel Haswell



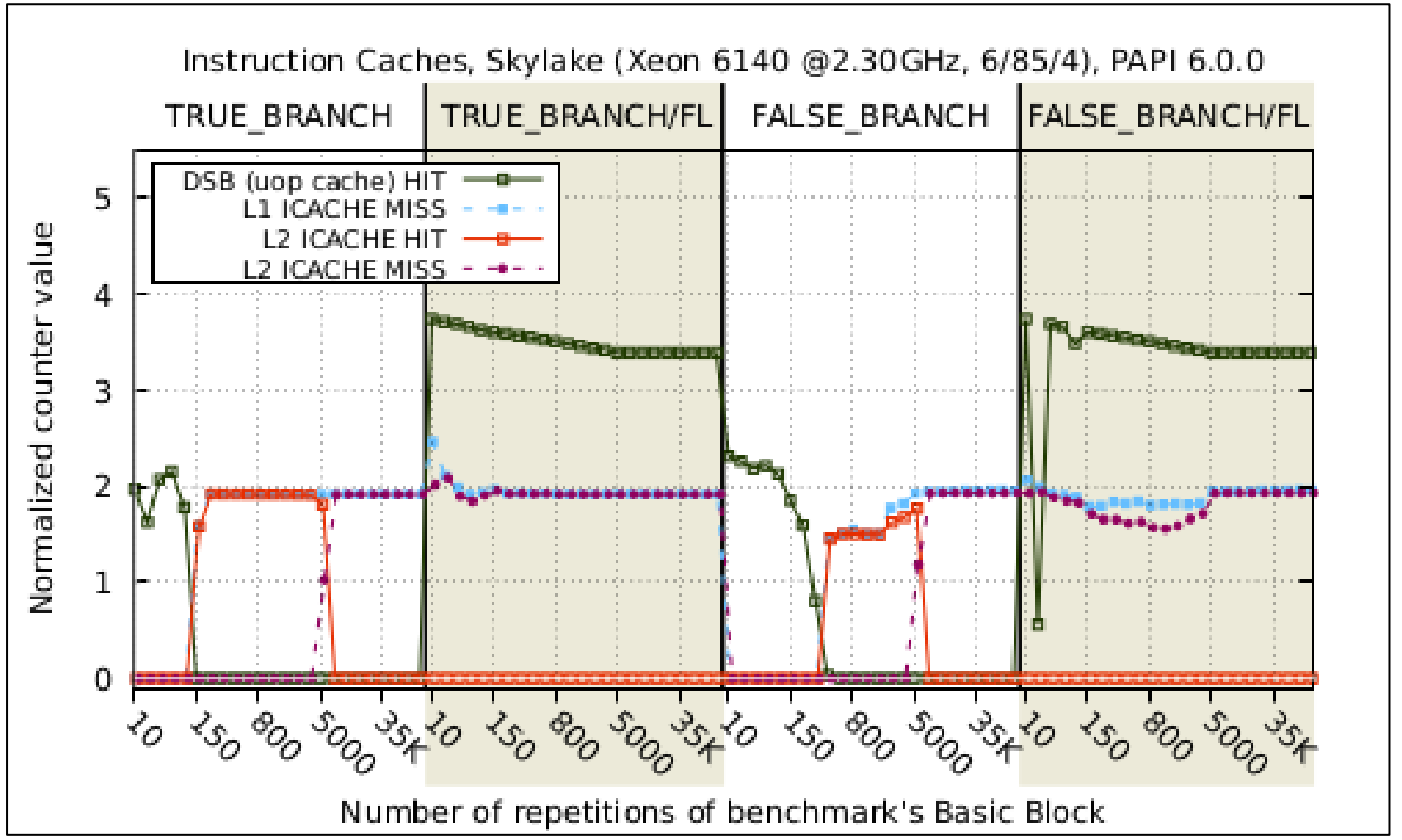
L2 Data Cache Misses – Intel Haswell



L2 Data Cache Misses – Intel Haswell

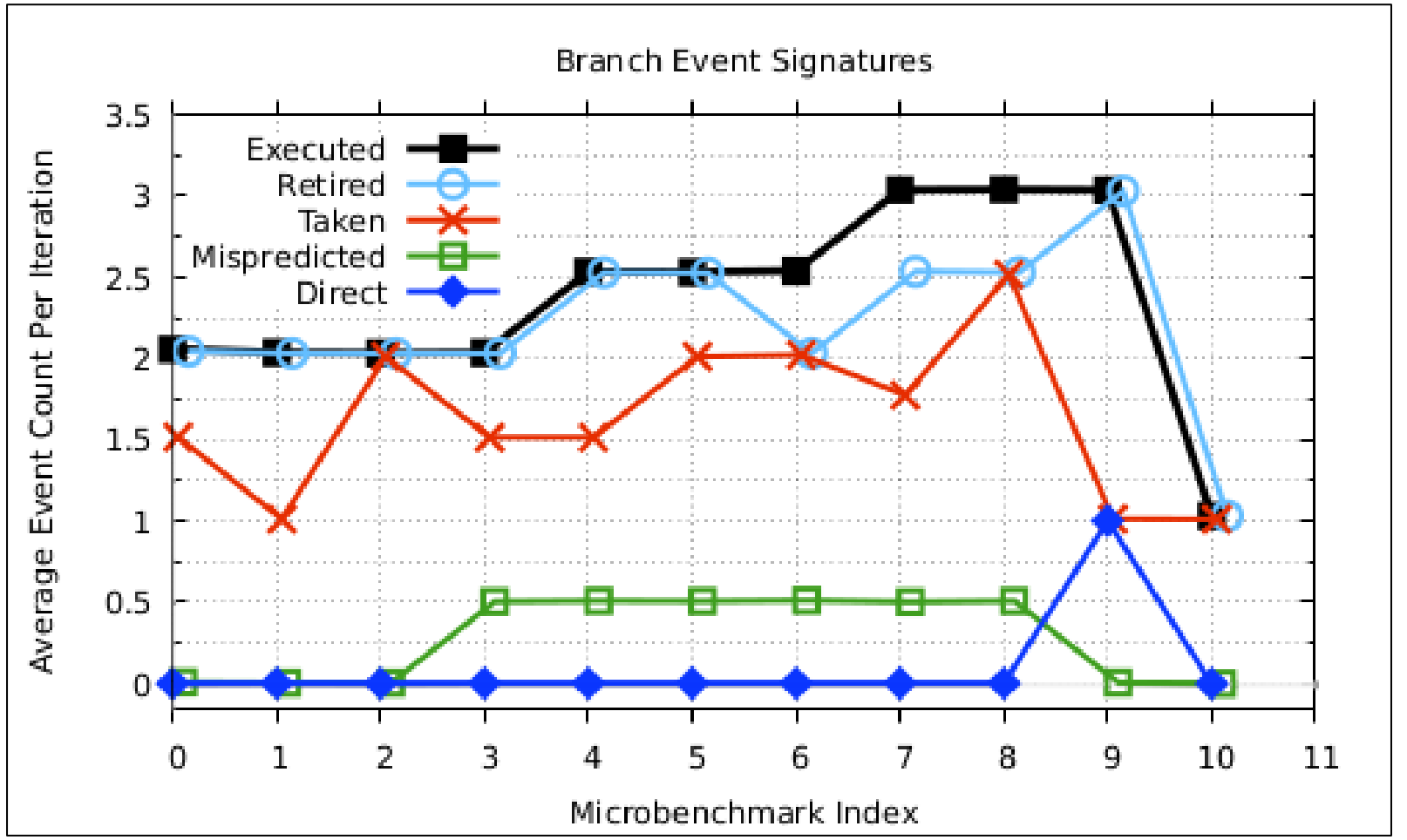


Instruction Cache



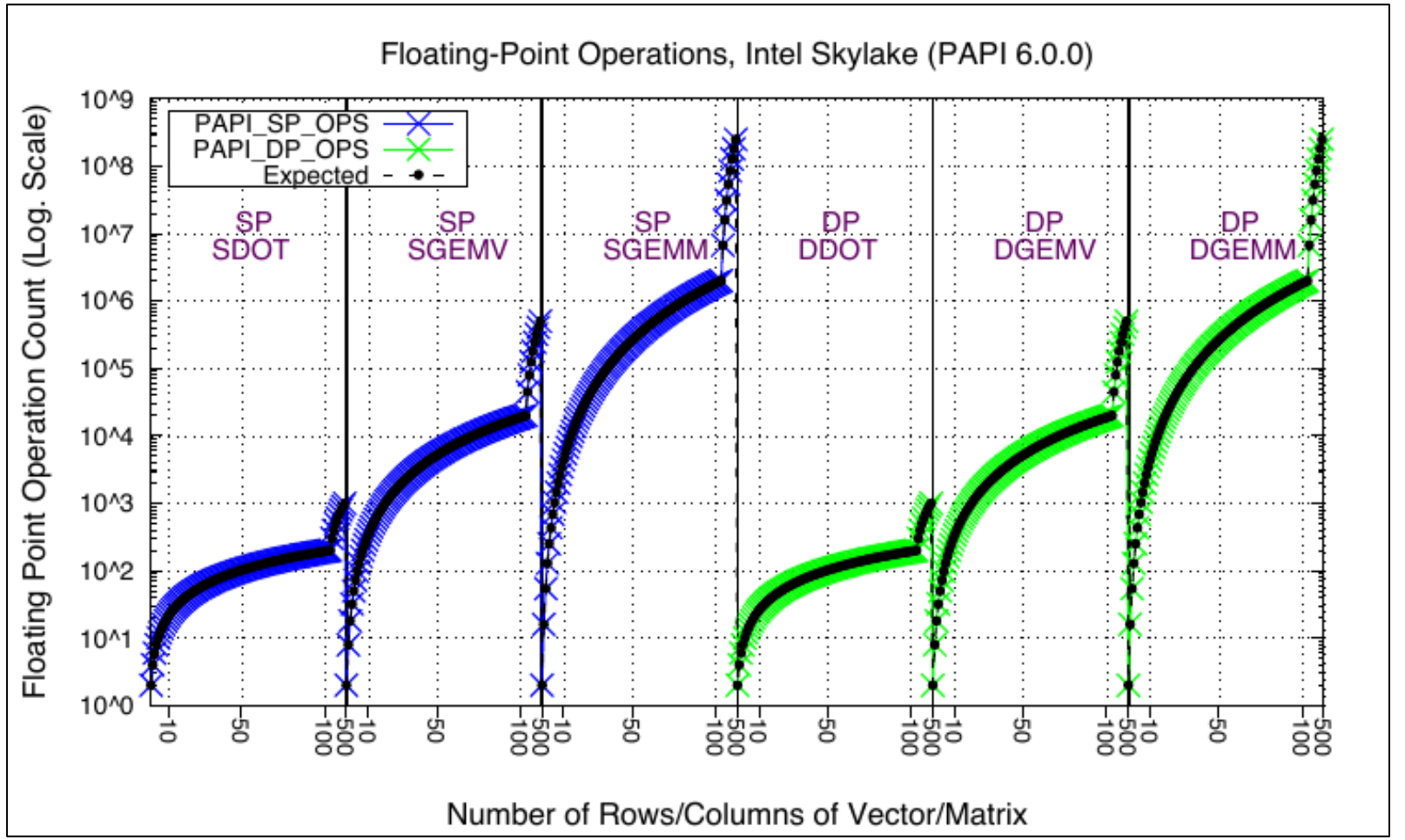
Credit: Barry *et al.*

Branching



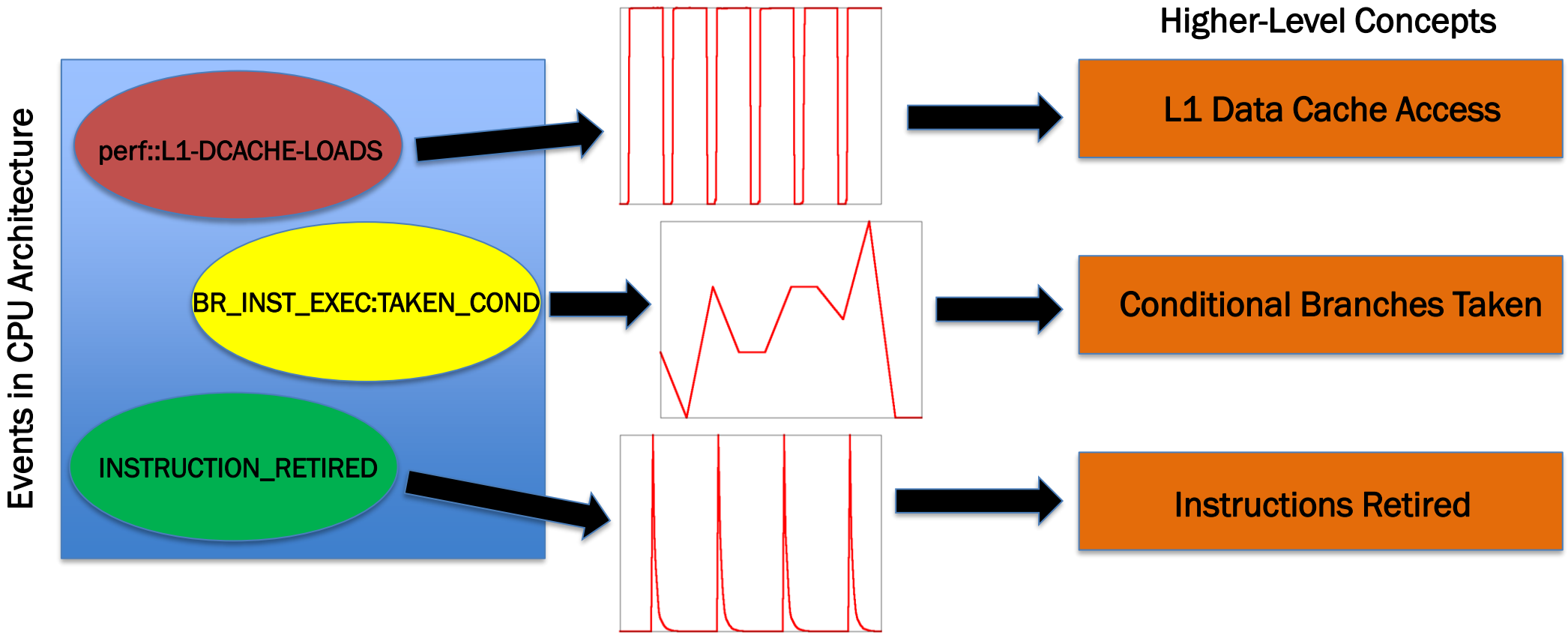
Credit: Barry *et al.*

Floating-Point Operations



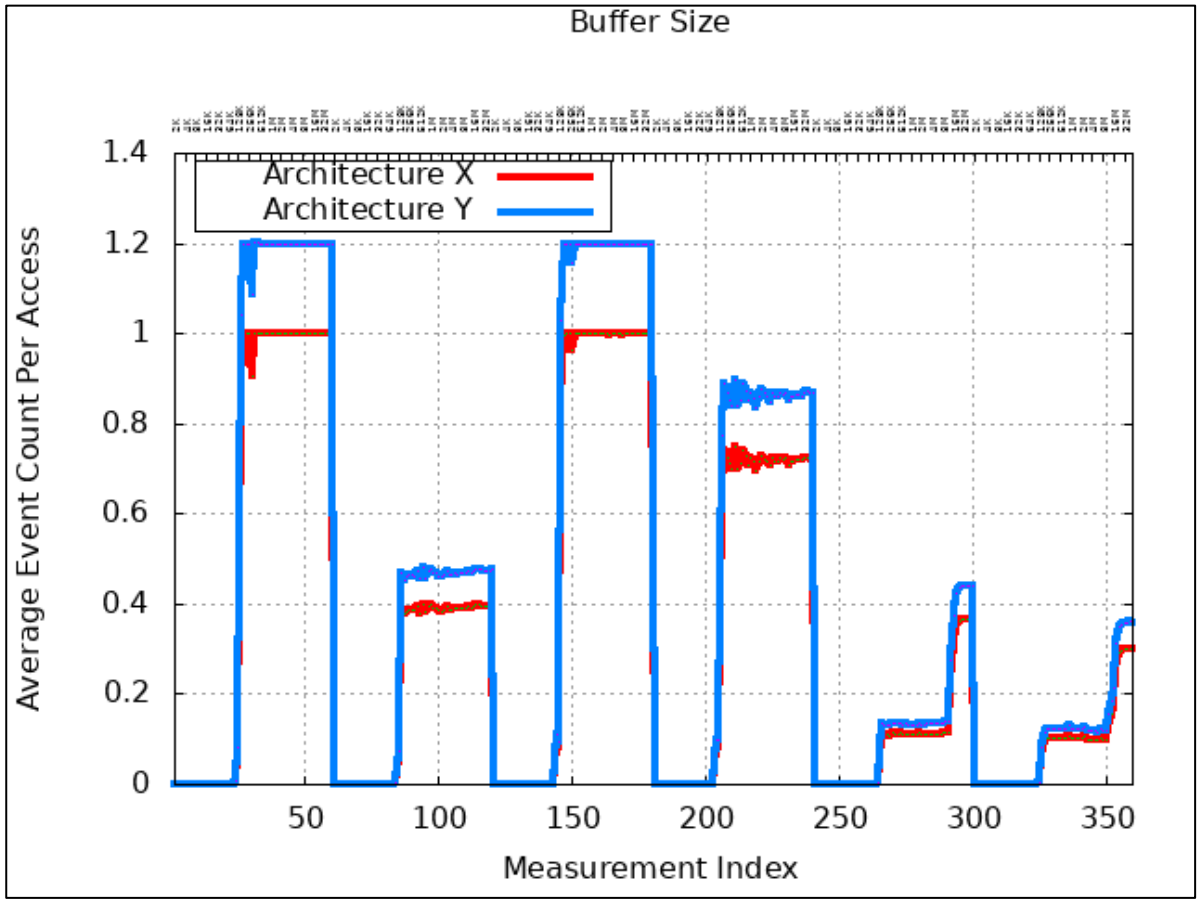
Credit: Barry *et al.*

The CAT Mapping

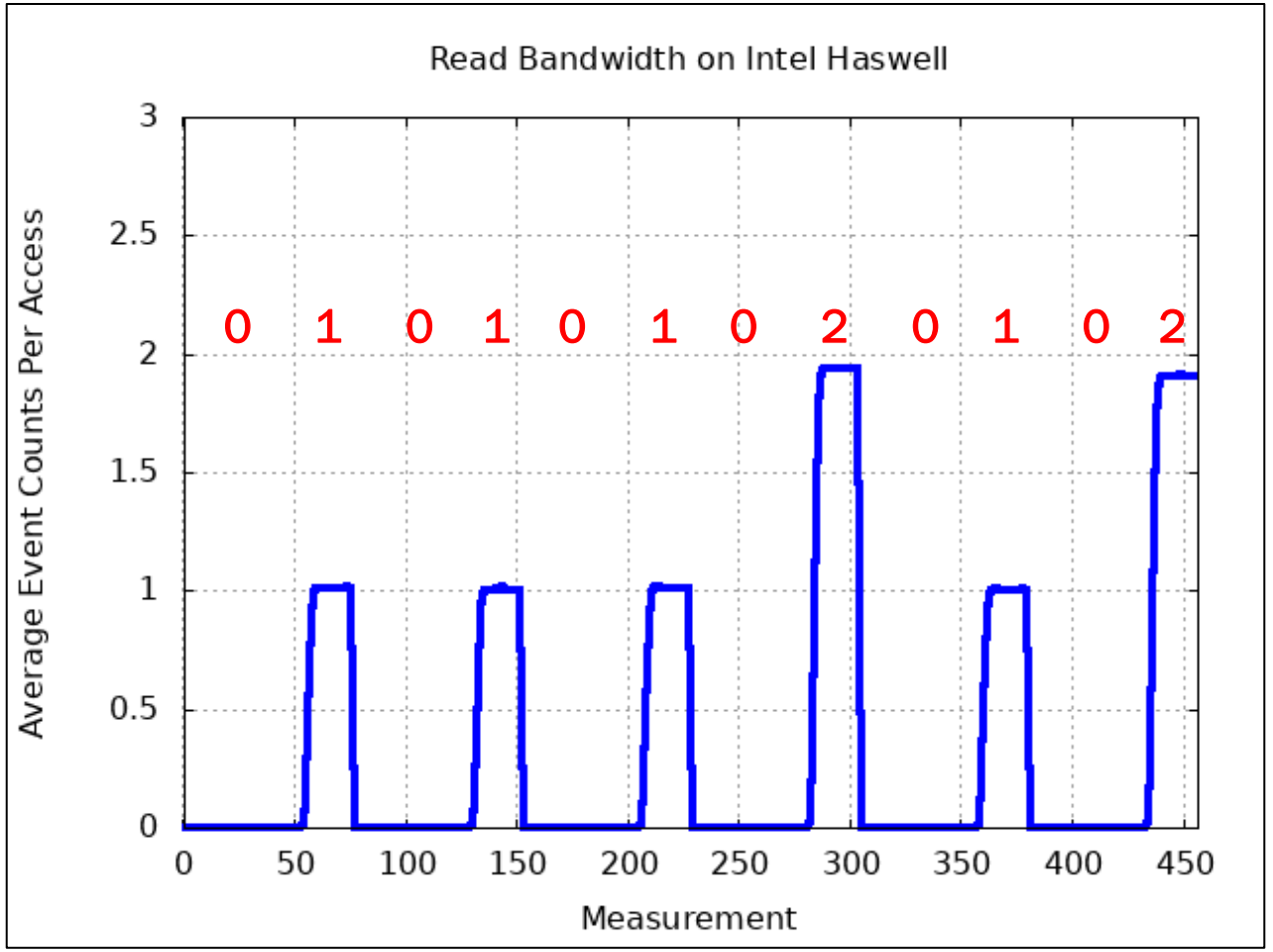


Data Analysis Challenges

- Two signatures can be identical (to a scale).



Data Representation



Linear Algebra

- Example signature: $\boldsymbol{a}_k = (0,1,0,1,0,1,0,2,0,1,0,2)$

- Set of signatures $\{\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n-1}\}$

$$\mathrm{A} = \begin{bmatrix} | & & | \\ \boldsymbol{a}_0 & \cdots & \boldsymbol{a}_{n-1} \\ | & & | \end{bmatrix} \in R^{mxn}$$

Method of Least Squares

- Preset vector (or *expected* measurement), **b**
- Gives linear regression coefficients $\{\beta_0, \beta_1, \ldots, \beta_{n-1}\}$

$$\boldsymbol{b} \cong \beta_0 \boldsymbol{a}_0 + \beta_1 \boldsymbol{a}_1 + \cdots + \beta_{n-1} \boldsymbol{a}_{n-1}$$

- Minimizes the L2-Norm:

$$\min_{\boldsymbol{\beta}} \|\mathrm{A}\boldsymbol{\beta} - \boldsymbol{b}\|_2$$

Dot Products

$$A^T A = \begin{bmatrix} | & & | \\ \boldsymbol{a}_0 & \dots & \boldsymbol{a}_{n-1} \\ | & & | \end{bmatrix}^T \begin{bmatrix} | & & | \\ \boldsymbol{a}_0 & \dots & \boldsymbol{a}_{n-1} \\ | & & | \end{bmatrix}$$

$$\Rightarrow (A^T A)_{i,j} = \langle \boldsymbol{a}_i, \boldsymbol{a}_j \rangle$$

Method of Least Squares

- Does not fare well if A contains linearly dependent columns
- An intuitive example:

$$L1\ Data\ Cache\ Accesses = L1\ Hits + L1\ Misses$$

- What happens if A contains signatures for each of the above?

Re-formulation?

- What if make the presets the columns of A?
- Set of p presets $\{\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1}\}$

$$\mathrm{B} = \begin{bmatrix} | & & | \\ \boldsymbol{b}_0 & \ldots & \boldsymbol{b}_{n-1} \\ | & & | \end{bmatrix} \in R^{mxp}$$

- Native event vector, **a**

Re-formulated Least Squares

- Gives linear regression coefficients $\{\beta_0, \beta_1, \ldots, \beta_{n-1}\}$

$$\boldsymbol{a} \cong \beta_0 \boldsymbol{b}_0 + \beta_1 \boldsymbol{b}_1 + \cdots + \beta_{n-1} \boldsymbol{b}_{n-1}$$

- Minimizes the L2-Norm:

$$\min_{\boldsymbol{\beta}} \|\mathrm{B}\boldsymbol{\beta} - \boldsymbol{a}\|_2$$

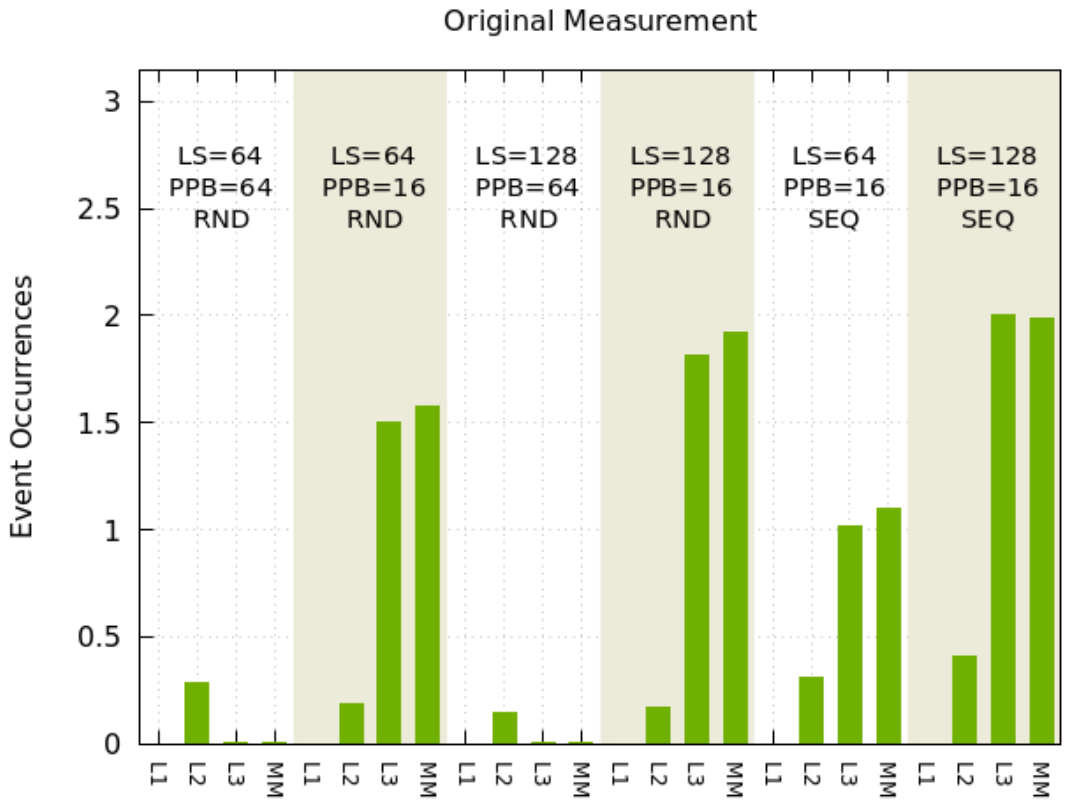
Re-formulated Least Squares

- Why is this useful?

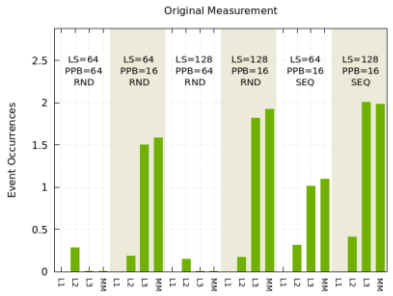
- $\boldsymbol{\beta}$ now tells us where in the meaningful event-space (as defined by presets) a given native event exists

Example: Least Squares

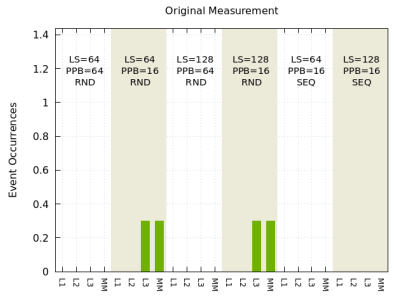
- **Benchmark Category:** Data Cache (Reading)
- **Event:** *L2_RQSTS:ALL_PF*



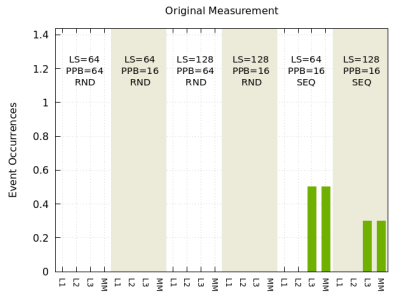
Least Squares Visualized



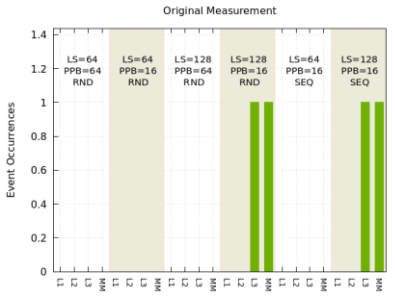
$\cong \boldsymbol{\beta_0}$

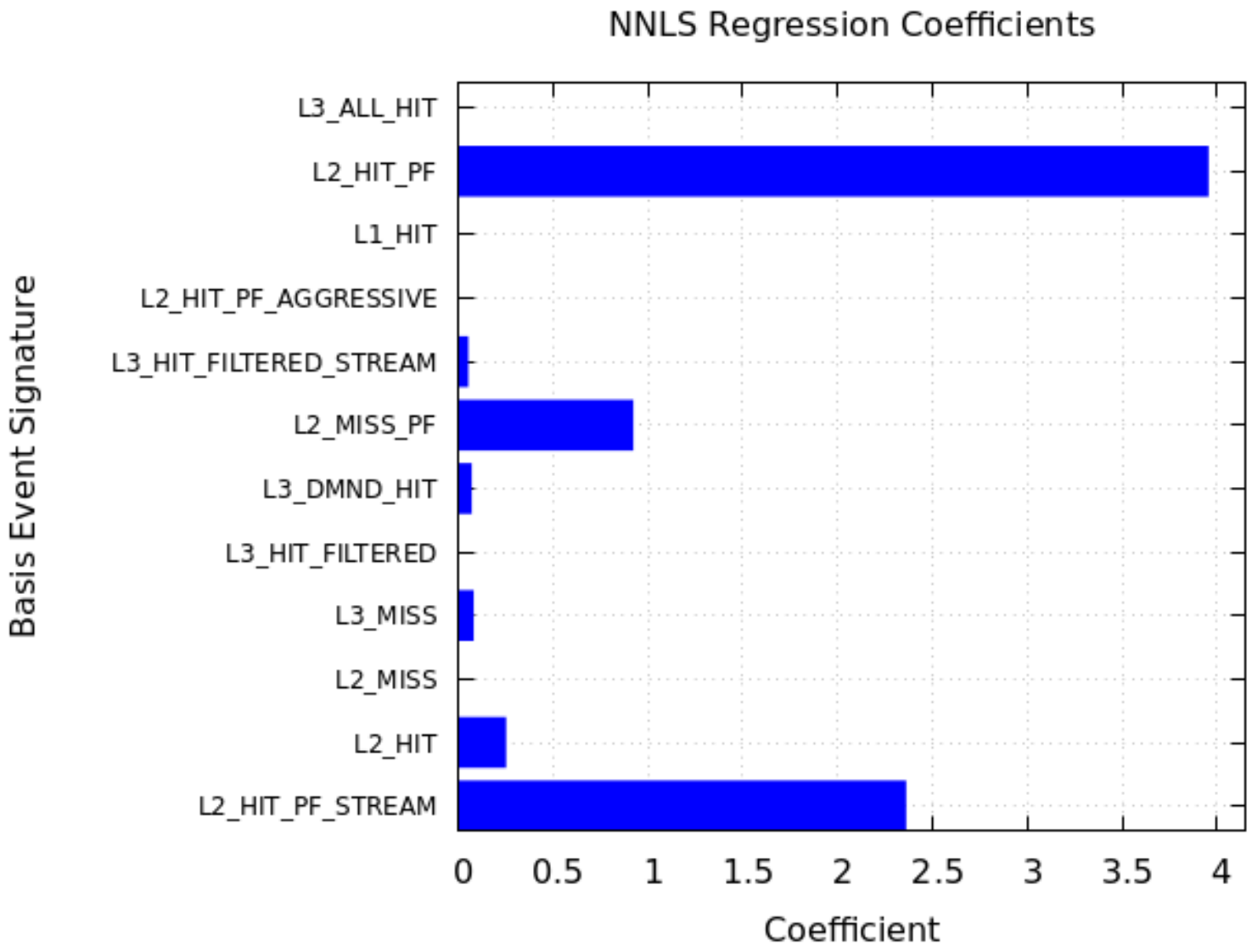


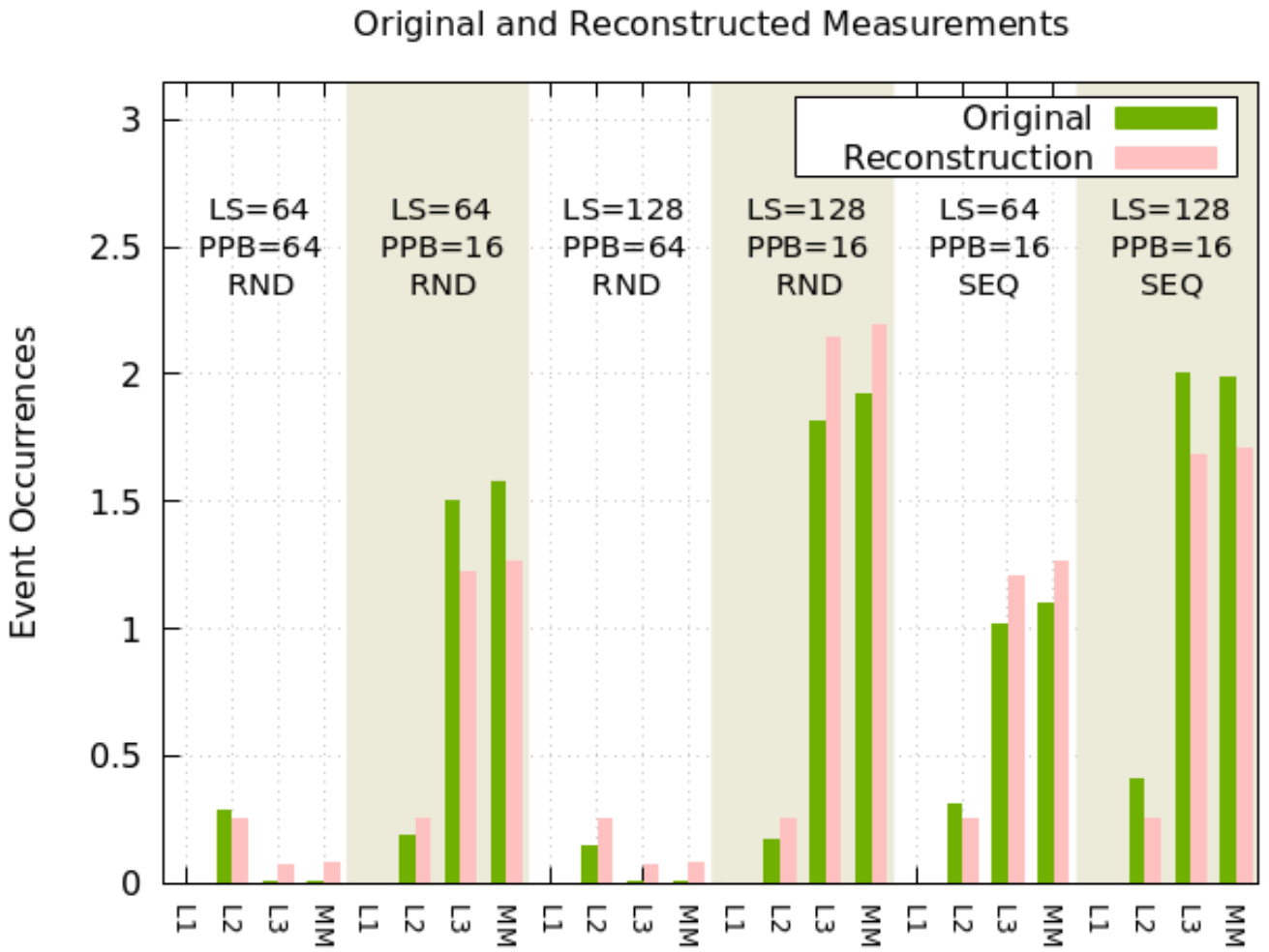
$+ \boldsymbol{\beta_1}$



$+ \cdots + \boldsymbol{\beta_{n-1}}$




NNLS Regression Coefficients
Basis Event Signature
L3_ALL_HIT
L2_HIT_PF
L1_HIT
L2_HIT_PF_AGGRESSIVE
L3_HIT_FILTERED_STREAM
L2_MISS_PF
L3_DMND_HIT
L3_HIT_FILTERED
L3_MISS
L2_MISS
L2_HIT
L2_HIT_PF_STREAM
0
0.5
1
1.5
2
2.5
3
3.5
4
Coefficient

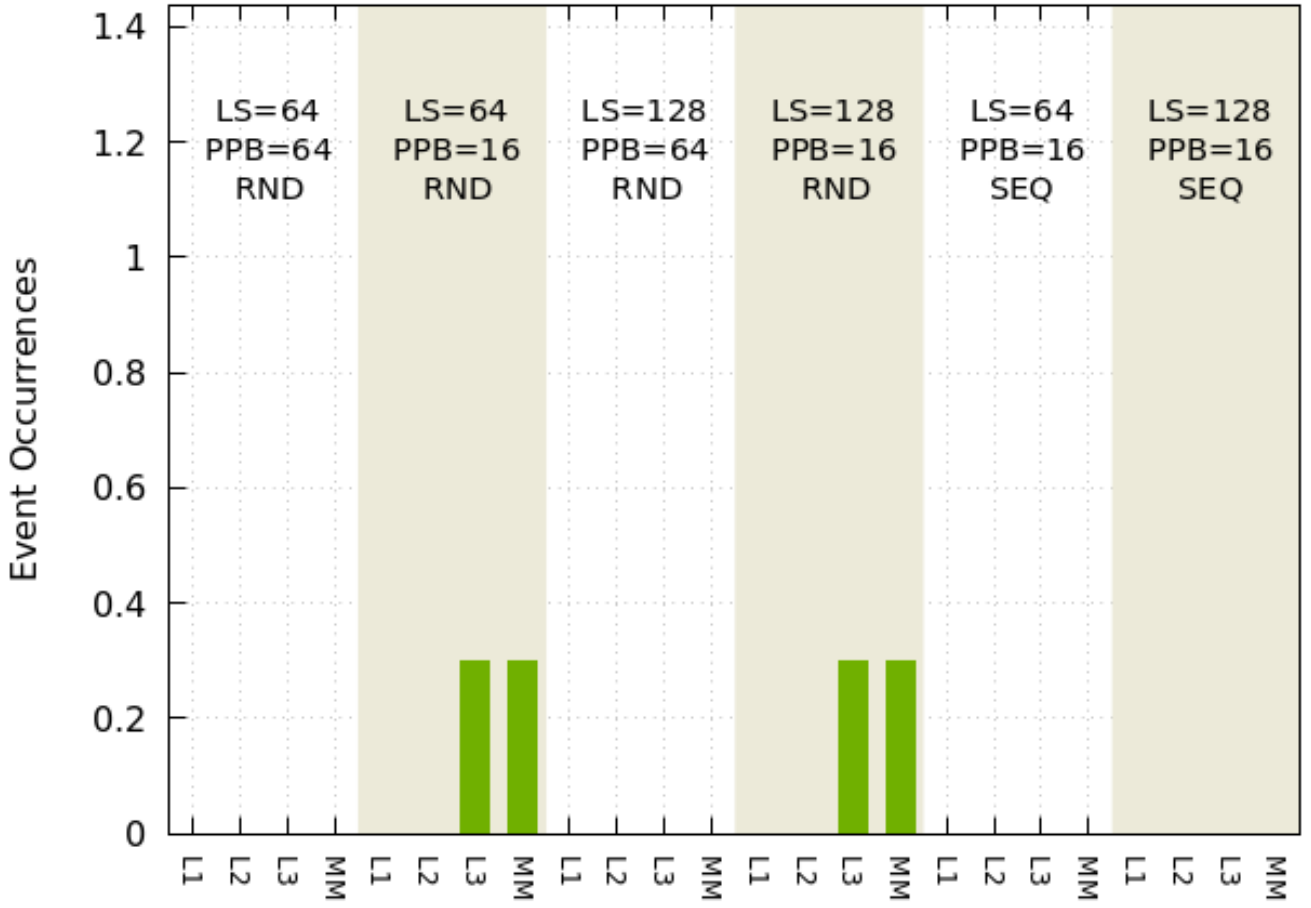

Original and Reconstructed Measurements
Event Occurrences
Original
Reconstruction
LS=64 PPB=64 RND
LS=64 PPB=16 RND
LS=128 PPB=64 RND
LS=128 PPB=16 RND
LS=64 PPB=16 SEQ
LS=128 PPB=16 SEQ
0
0.5
1
1.5
2
2.5
3
L1
L2
L3
MM

What could go wrong?

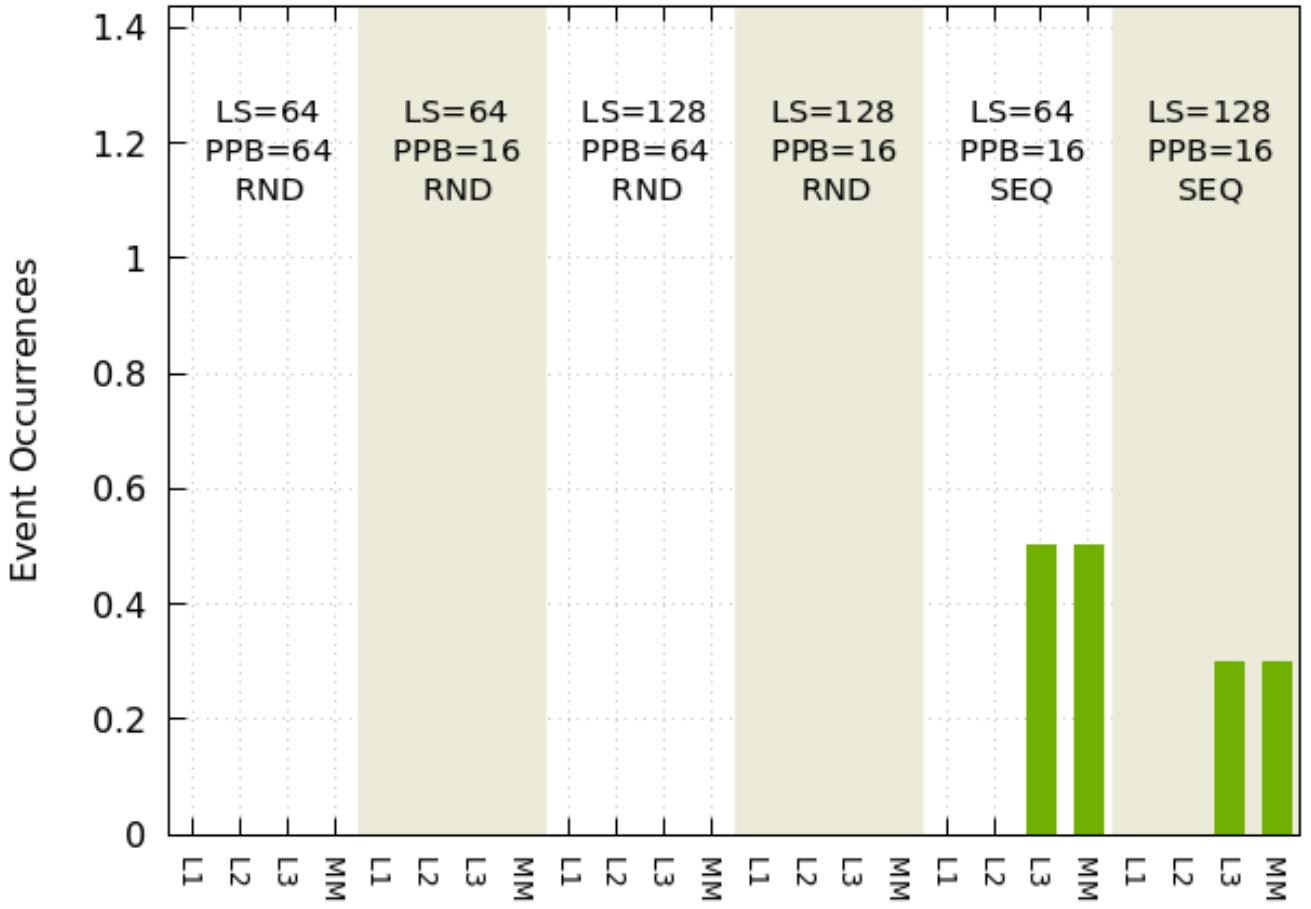
- Too much nuance (*e.g.*, prefetching and prefetching-when-it's-difficult-to-prefetch)

- Each basis depends on micro-architecture

L2 Prefetch Hits



L2 Prefetch Hits – Stream



Back to Original Formulation

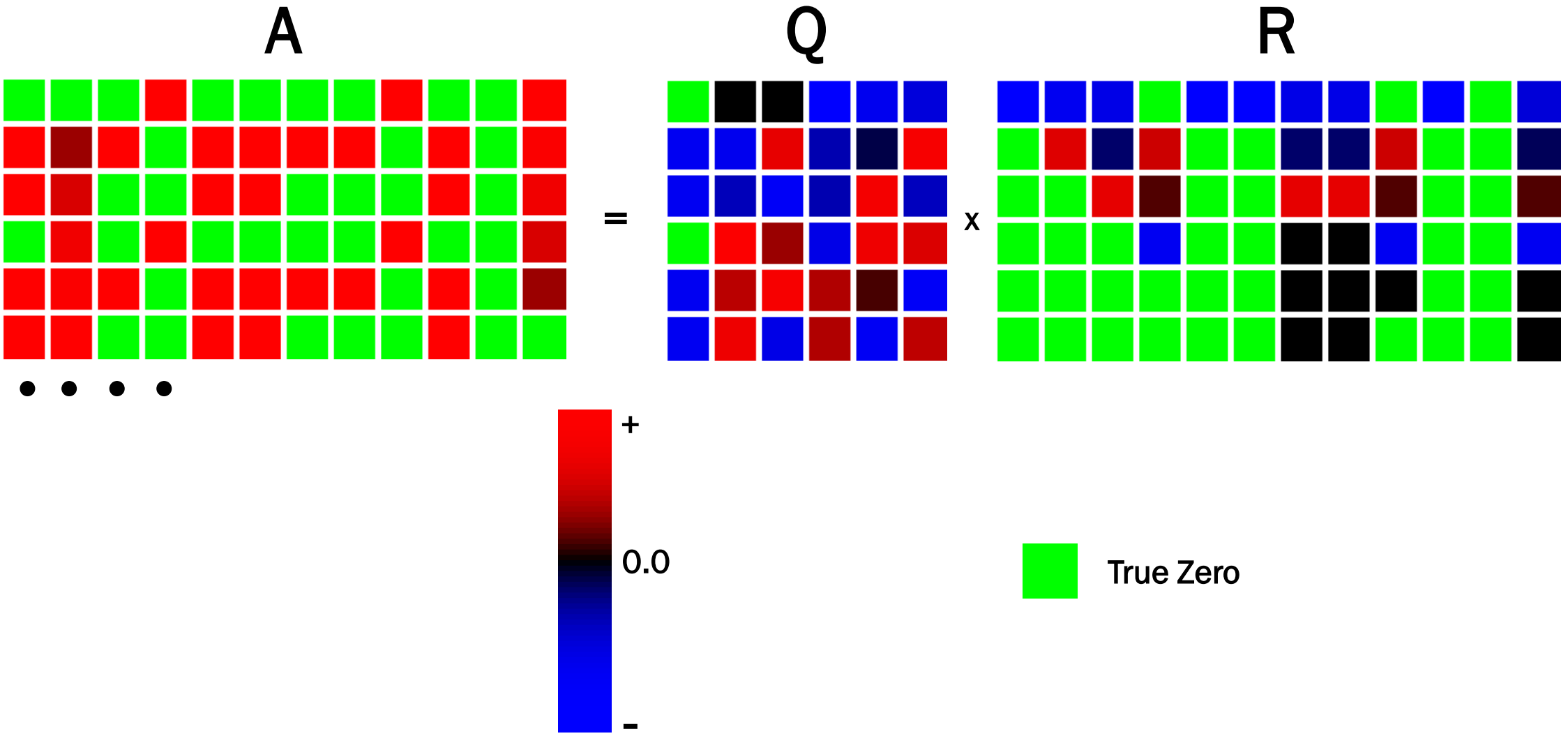
- Preset vector, **b**
- **A** contains measurements native events

$$\mathrm{A} = \begin{bmatrix} | & & | \\ \boldsymbol{a}_0 & \cdots & \boldsymbol{a}_{n-1} \\ | & & | \end{bmatrix}$$

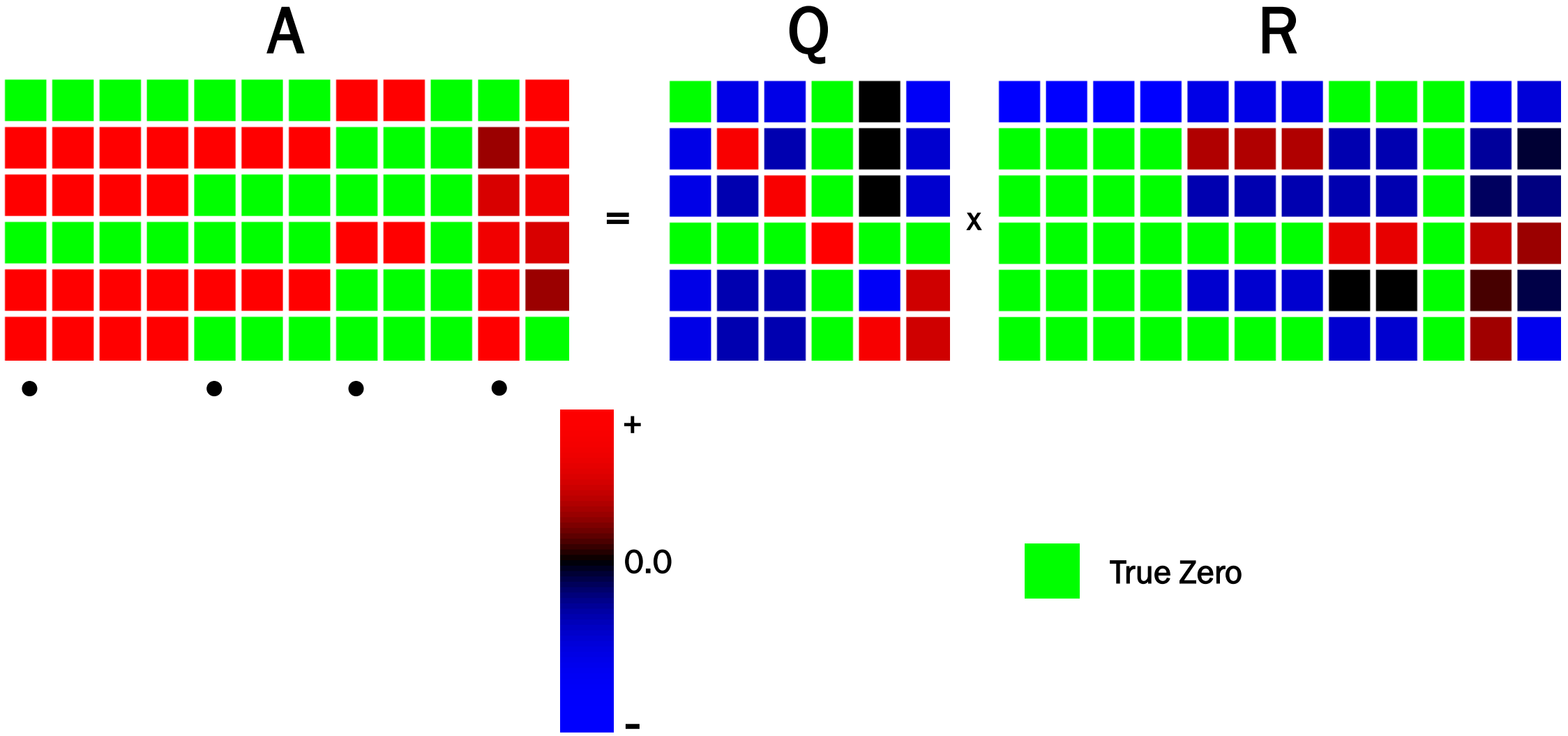
Outstanding Problems

- How do we remove linear dependence in the columns of A?
- How do we do that **without** losing the structure of A?
- How do we squelch the random noise keeping the columns numerically linearly independent?

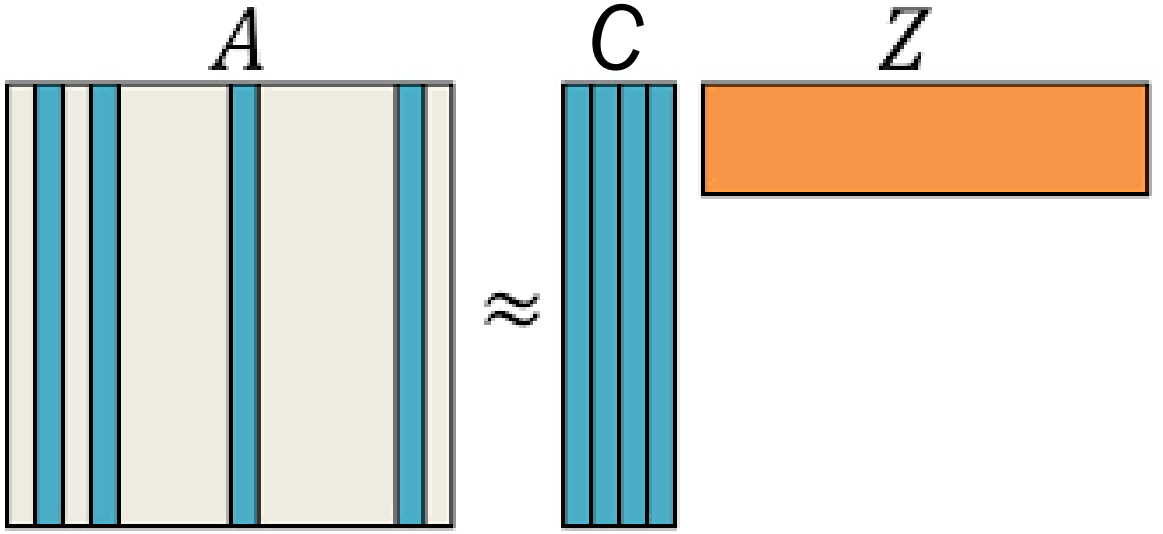
QR Factorization



QR Factorization – Counterexample

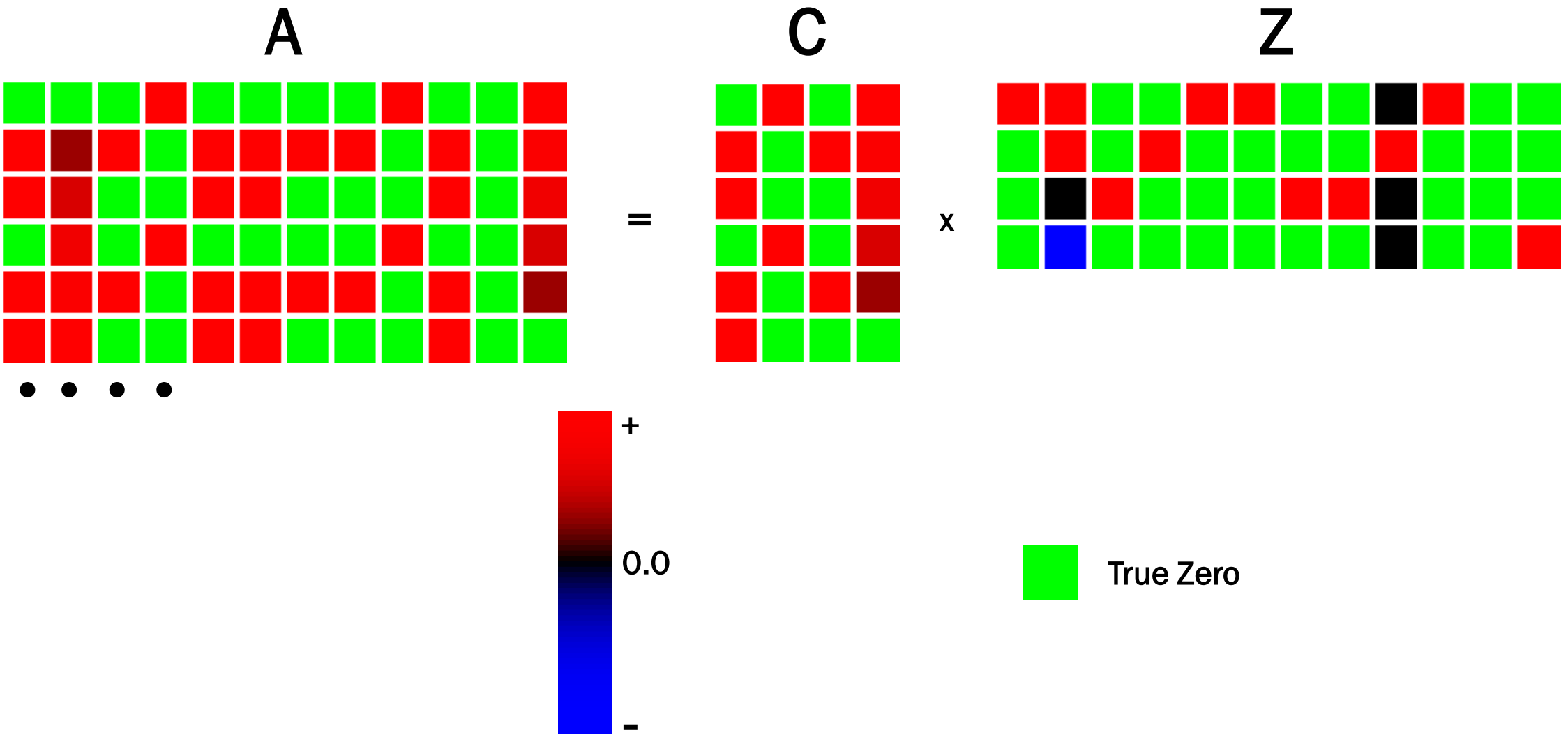


Interpolative Decomposition (ID)

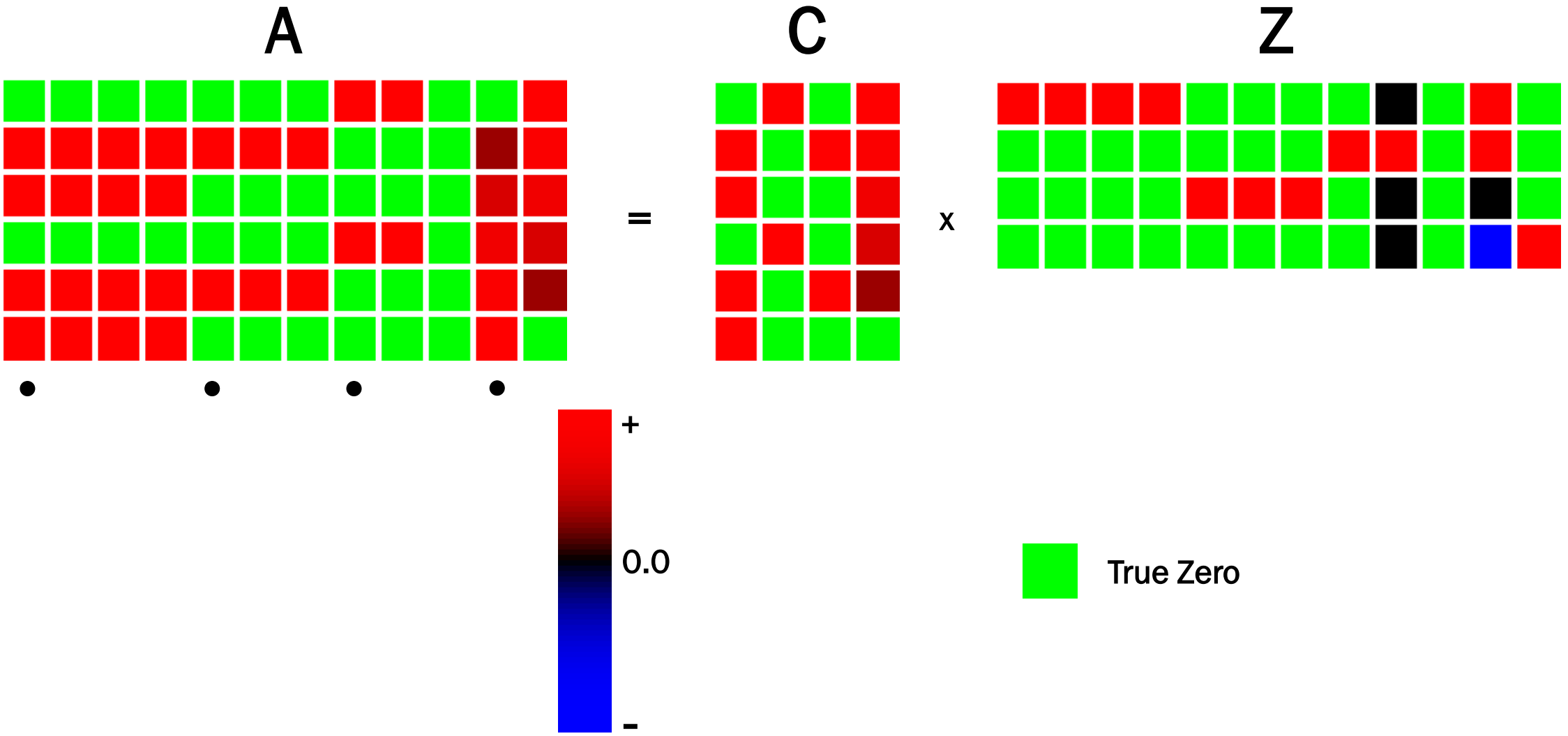


Credit: Ying *et al.*

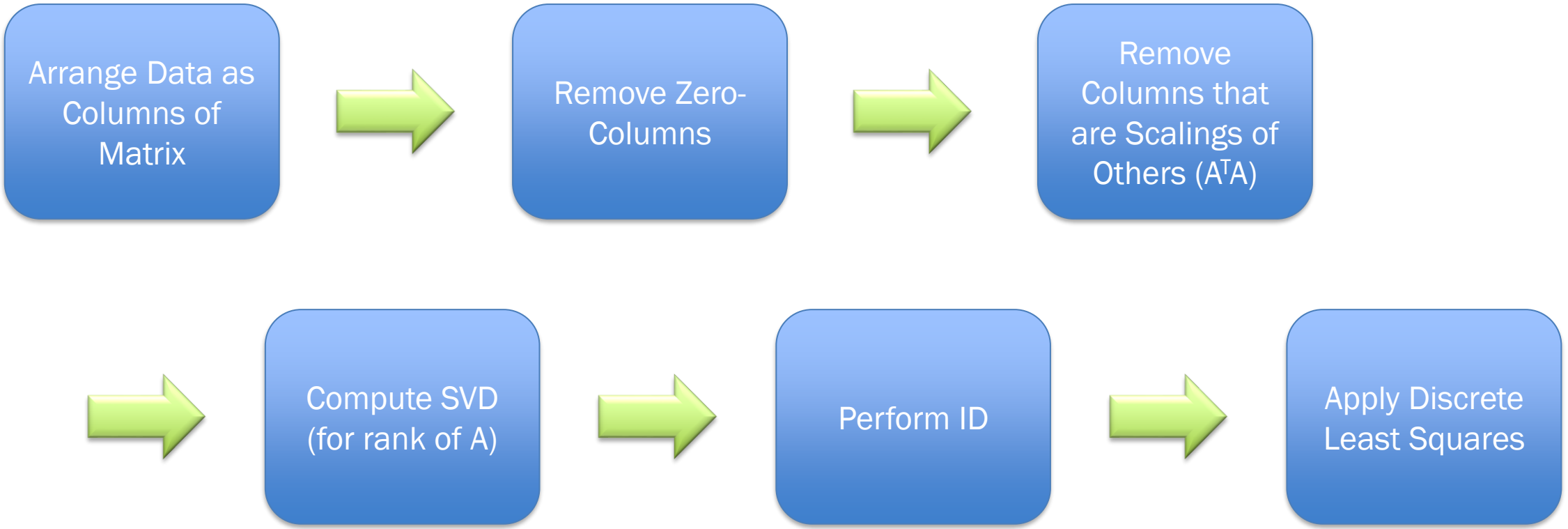
Interpolative Decomposition



Interpolative Decomposition – “Unlucky” A



Status of Data Analysis – Workflow



Conclusions

- CAT and ongoing data analyses allow us to identify structure of performance hardware events.

- Multi-stage analysis has proven useful.

Acknowledgement

Thank you to Natalie Beams and Hartwig Anzt for their linear algebra consulation!

References

- P.G. Martinsson, V. Rokhlin, Y. Shkolnisky, M. Tygert. “ID: a software package for low-rank approximation of matrices via interpolative decompositions, version 0.2.” 2014. (http://tygert.com/id_doc.4.pdf)
- L. Ying, A. Damle, L. Lin, J. Lu. “Interpolative Decomposition and its Applications in Quantum Chemistry.” 2018. (https://www.ki-net.umd.edu/activities/presentations/9_871_cscamm.pdf)
- D. Barry, H. Jagode, A. Danalis. “Effortless Monitoring of Arithmetic Intensity with PAPI’s Counter Analysis Toolkit.” 2021.