These slides + links to papers: **hpcgarage.org/futuresparse23**

Didem, I have bad news for you …

THE FUTURE IS SPARSE

RICH VUDUC – NOVEMBER 17, 2023

Georgia Tech College of Computing
School of Computational Science and Engineering

Georgia Tech College of Computing
Center for Research into Novel Computing Hierarchies

https://tenor.com/view/whomp-whomp-whomp-whomp-so-sad-smallest-violin-violin-gif-22252865

These slides + links to papers: **hpcgarage.org/futuresparse23**

Didem, I have bad news for you …

THE FUTURE IS SPARSE

RICH VUDUC – NOVEMBER 17, 2023

Georgia Tech College of Computing
School of Computational Science and Engineering

Georgia Tech College of Computing
Center for Research into Novel Computing Hierarchies

https://tenor.com/view/whomp-whomp-whomp-whomp-so-sad-smallest-violin-violin-gif-22252865

The future is not sparse

The future is not sparse

Four “generations” of computing

Gregory Abowd (2016). “Beyond Weiser: From ubiquitous computing to collective computing.” DOI: 10.1109/MC.2016.22

OUTLOOK

TABLE 1. A framework for comparing computing generations, inspired by Mark Weiser.

Generation	Time frame	Human–computer ratio	Canonical device	Application	
				Initial	Follow-on
1	Mid-1930s	Many–1	Mainframe	Scientific calculation	Data processing
2	Late 1960s	1–1	PC	Spreadsheet	Database management, document processing
3	Late 1980s	1–many	Inch/foot/yard	Calendar and contact management, human–human communication	Location-based services, social media, app ecosystem, education
4	Mid-2000s	Many–many	Cloud/crowd/shroud	Personal navigation and entertainment	Health advisors, educational assistants, supply chain logistics

Four "generations" of computing

Gregory Abowd (2016). "Beyond Weiser: From ubiquitous computing to collective computing." DOI: 10.1109/MC.2016.22

OUTLOOK

TABLE 1. A framework for comparing computing generations, inspired by Mark Weiser.

Generation	Time frame	Human–computer ratio	Canonical device	Application	
				Initial	Follow-on
1	Mid-1930s	Many–1	Mainframe	Scientific calculation	Data processing
2	Late 1960s	1–1	PC	Spreadsheet	Database management, document processing
3	Late 1980s	1–many	Inch/foot/yard	Calendar and contact management, human–human communication	Location-based services, social media, app ecosystem, education
4	Mid-2000s	Many–many	Cloud/crowd/shroud	Personal navigation and entertainment	Health advisors, educational assistants, supply chain logistics

Four "generations" of computing

Gregory Abowd (2016). "Beyond Weiser: From ubiquitous computing to collective computing." DOI: 10.1109/MC.2016.22

OUTLOOK

TABLE 1. A framework for comparing computing generations, inspired by Mark Weiser.

Generation	Time frame	Human–computer ratio	Canonical device	Application	
				Initial	Follow-on
1	Mid-1930s	Many–1	Mainframe	Scientific calculation	Data processing
2	Late 1960s	1–1	PC	Spreadsheet	Database management, document processing
3	Late 1980s	1–many	Inch/foot/yard	Calendar and contact management, human–human communication	Location-based services, social media, app ecosystem, education
4	Mid-2000s	Many–many	Cloud/crowd/shroud	Personal navigation and entertainment	Health advisors, educational assistants, supply chain logistics

Top500
Nov. '23

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	4,742,808	585.34	1,059.33	24,687
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703

Top500
Nov. '23

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84

4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703

Top500
Nov. '23

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84

4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107

Myth 12: All HPC Will Be Subsumed by the Clouds!

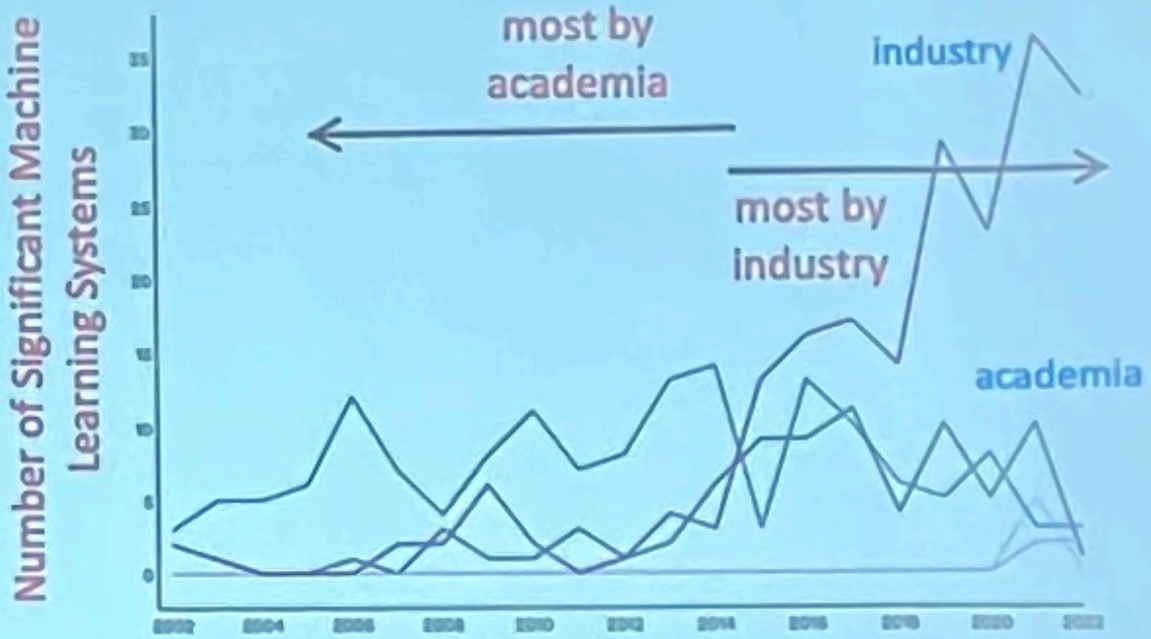
The rapidly advancing AI and new precision options has reignited the cloud discussion. The question whether clouds will subsume supercomputing has been ongoing for more than a decade, since the late 2000s Deelman et al. (2008), but remains inconclusive. Today's cloud offerings offer a wide spectrum for HPC customers, ranging from low-cost standard virtual machines to specialized top-gear HPC equipment in

S. Matsuoka (2023). *"Myths and legends in high-performance computing."* arXiv: 2301.02432

AI is emerging as "big science" in the tradition of nuclear and high energy physics

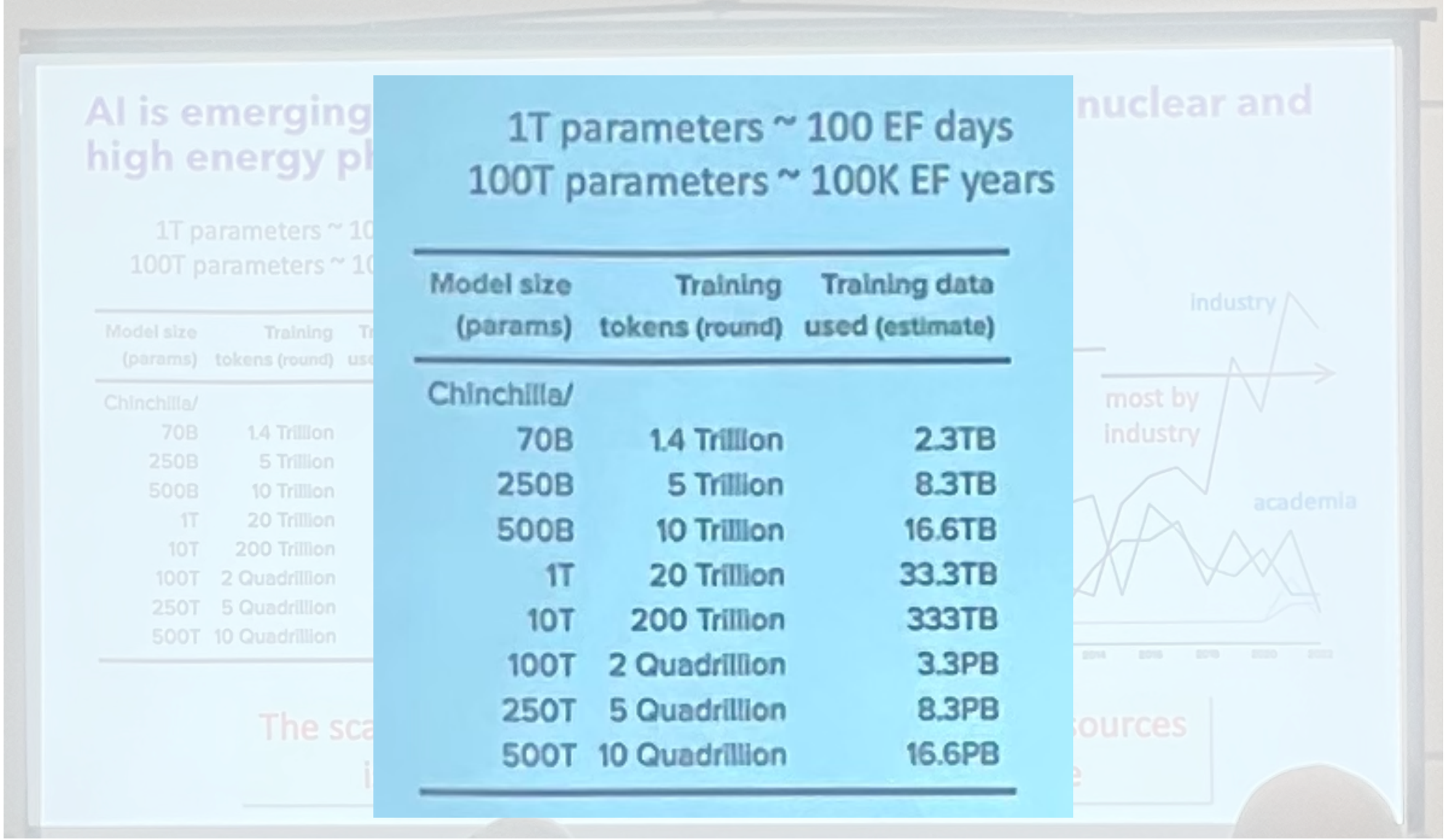
1T parameters ~ 100 EF days
100T parameters ~ 100K EF years

Model size (params)	Training tokens (round)	Training data used (estimate)
Chinchilla/		
70B	1.4 Trillion	2.3TB
250B	5 Trillion	8.3TB
500B	10 Trillion	16.6TB
1T	20 Trillion	33.3TB
10T	200 Trillion	333TB
100T	2 Quadrillion	3.3PB
250T	5 Quadrillion	8.3PB
500T	10 Quadrillion	16.6PB

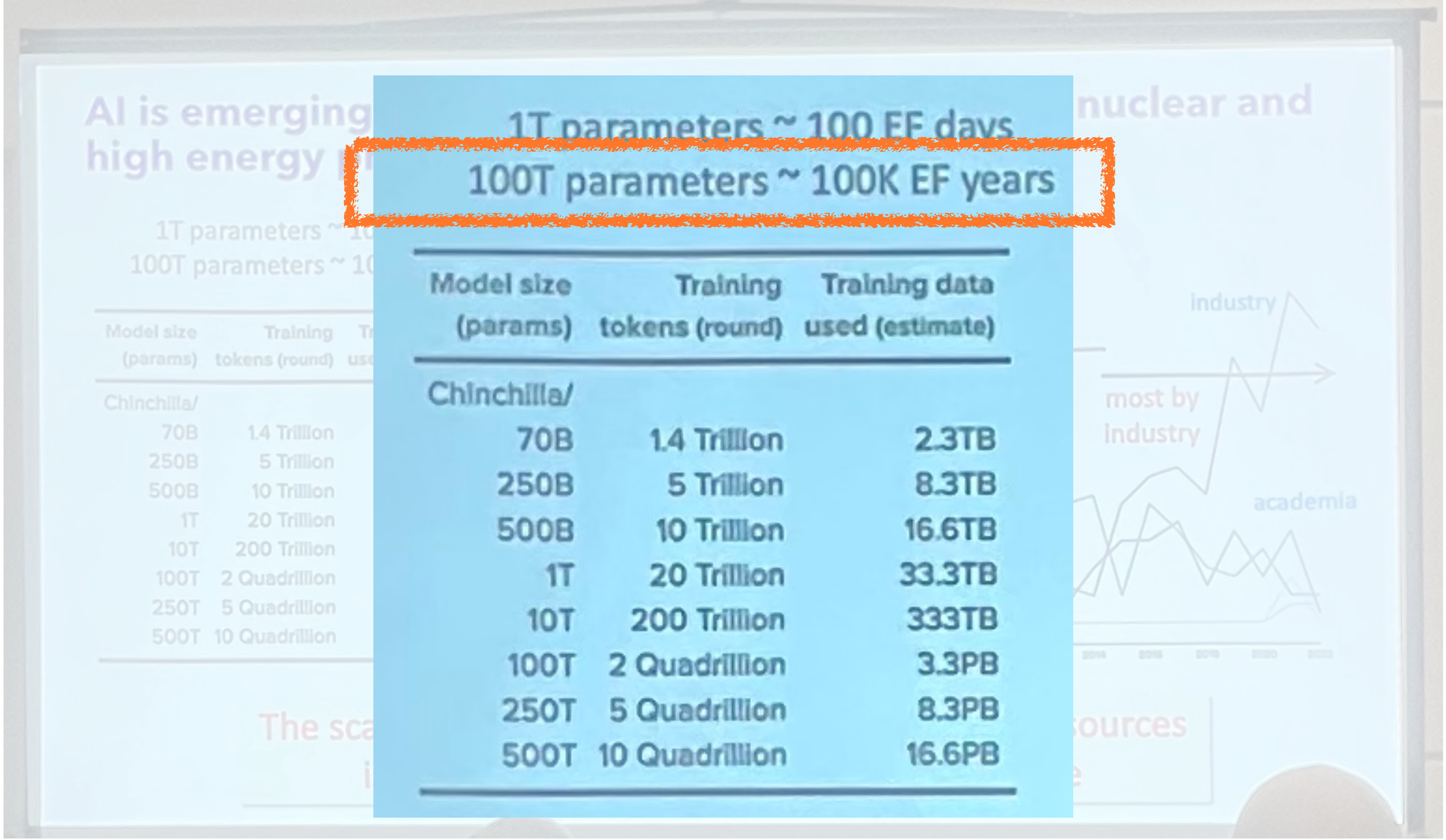


The scale of needed human and computational resources is beginning to reshape leadership in science

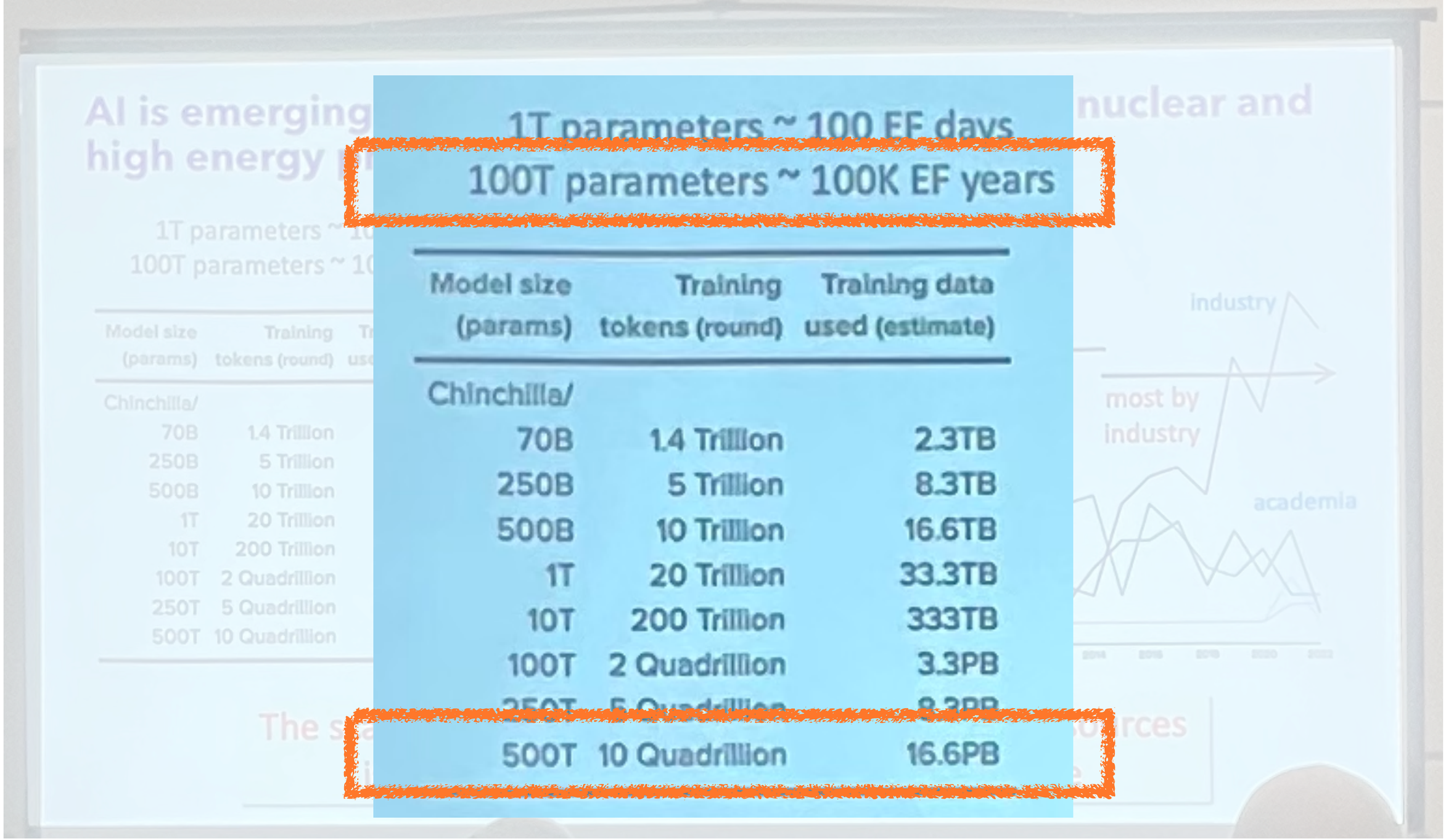
From: SC23 talk by Rick Stevens (Argonne National Lab)



From: SC23 talk by Rick Stevens (Argonne National Lab)



From: SC23 talk by Rick Stevens (Argonne National Lab)



From: SC23 talk by Rick Stevens (Argonne National Lab)

Q: Can these models be sparse?

A: Yes, but ML sparse is "fake" sparse.

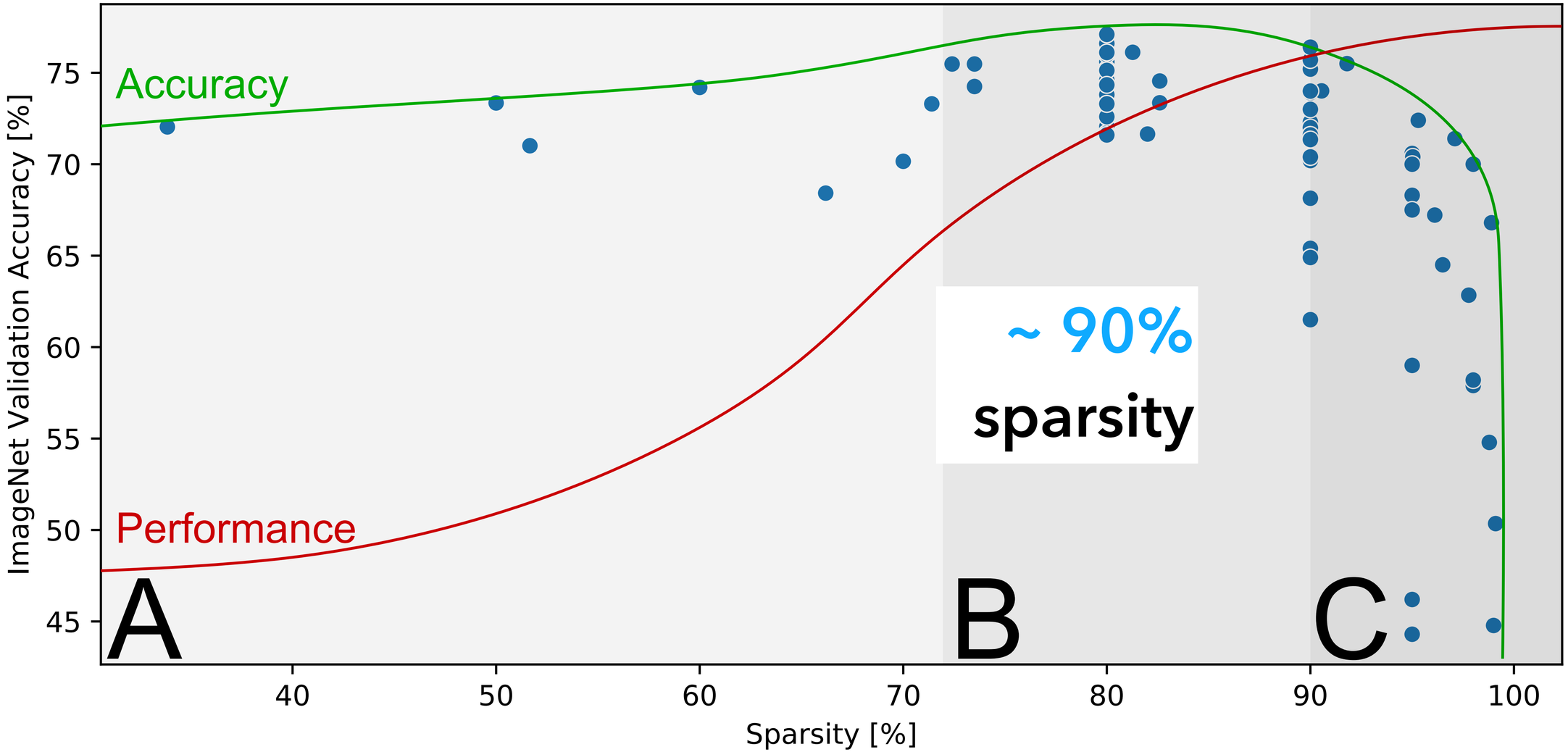


Fig. 4. Typical test error vs. sparsity showing Occam's hill (network: ResNet-50 on Top-1 ImageNet).

Hoefler et al. (2021). "Sparsity in Deep Learning: … arXiv:2102.00554

Frantar et al. (2023). "Scaling laws for sparsely-connected foundation models. arXiv:2309.08520v1

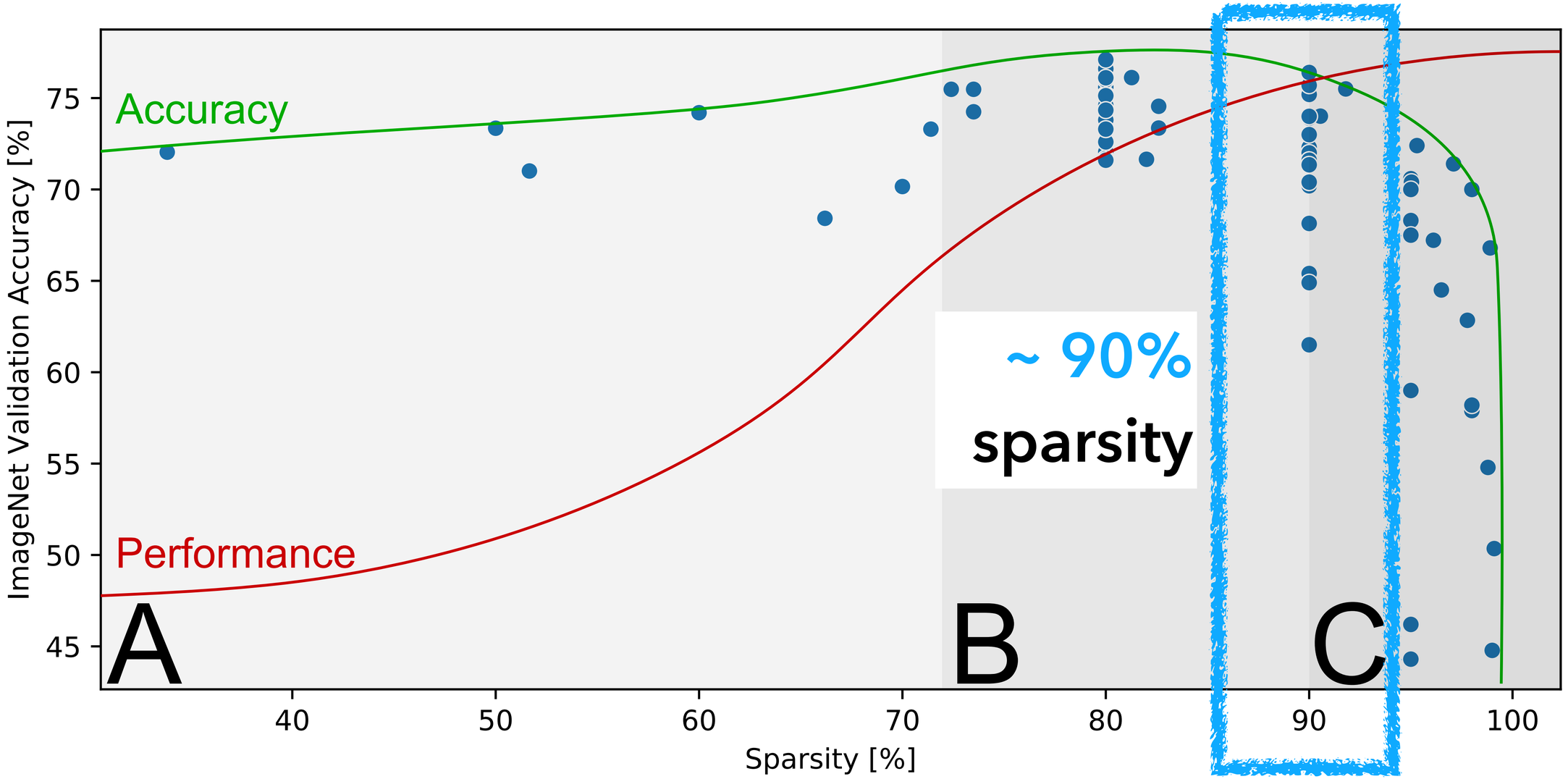
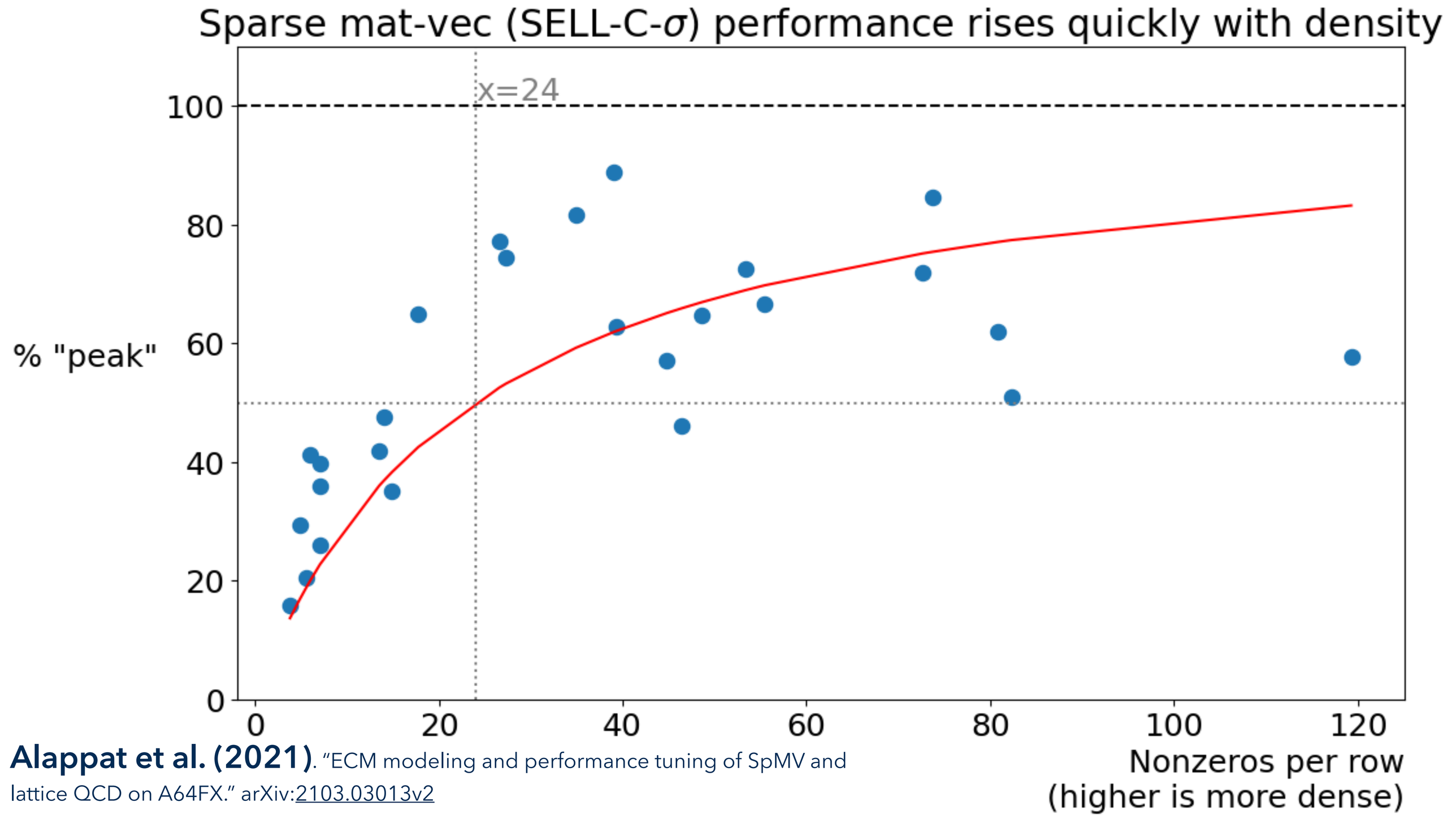


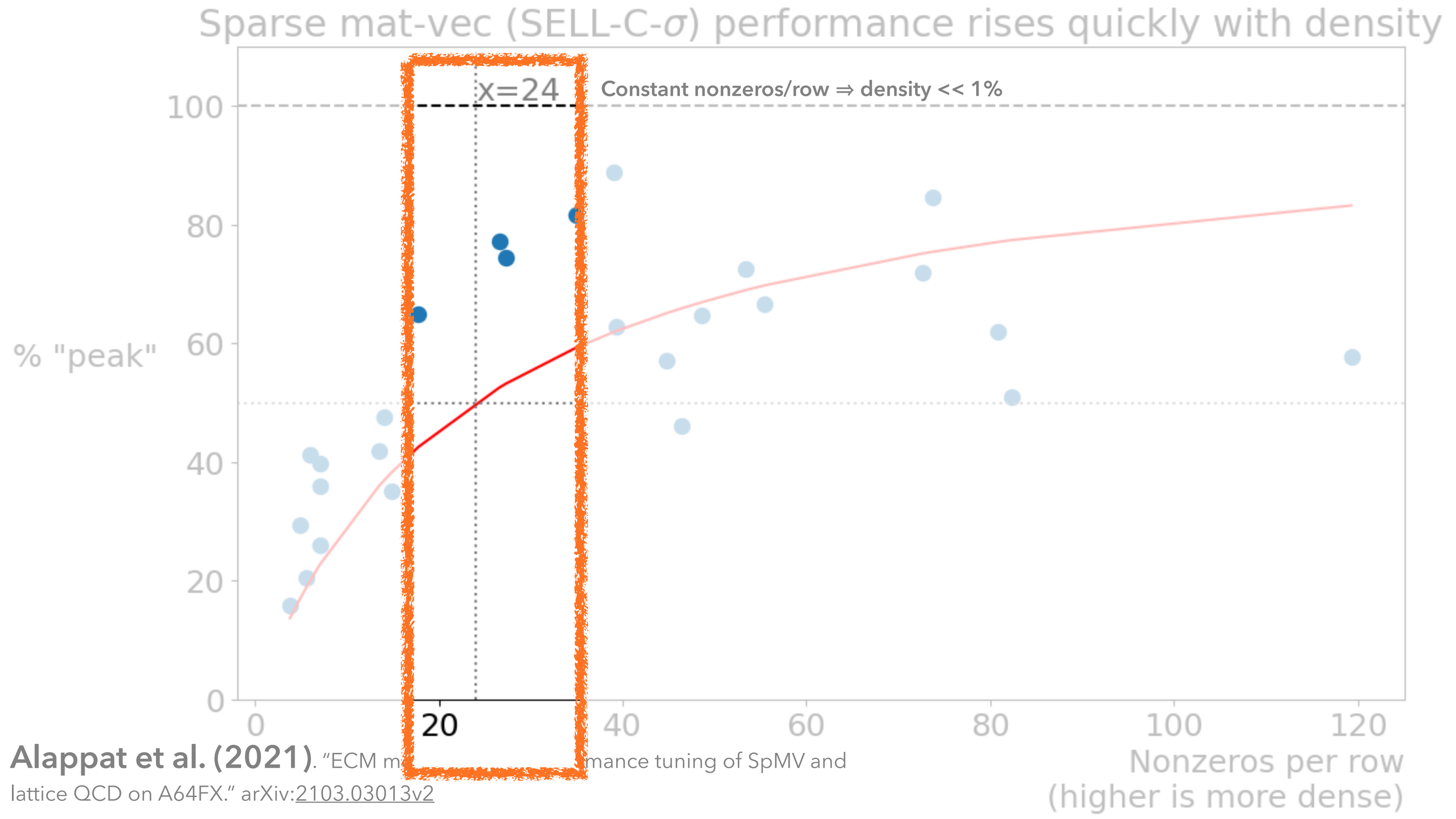
Fig. 4. Typical test error vs. sparsity showing Occam's hill (network: ResNet-50 on Top-1 ImageNet).

Hoefler et al. (2021). "Sparsity in Deep Learning: … arXiv:2102.00554

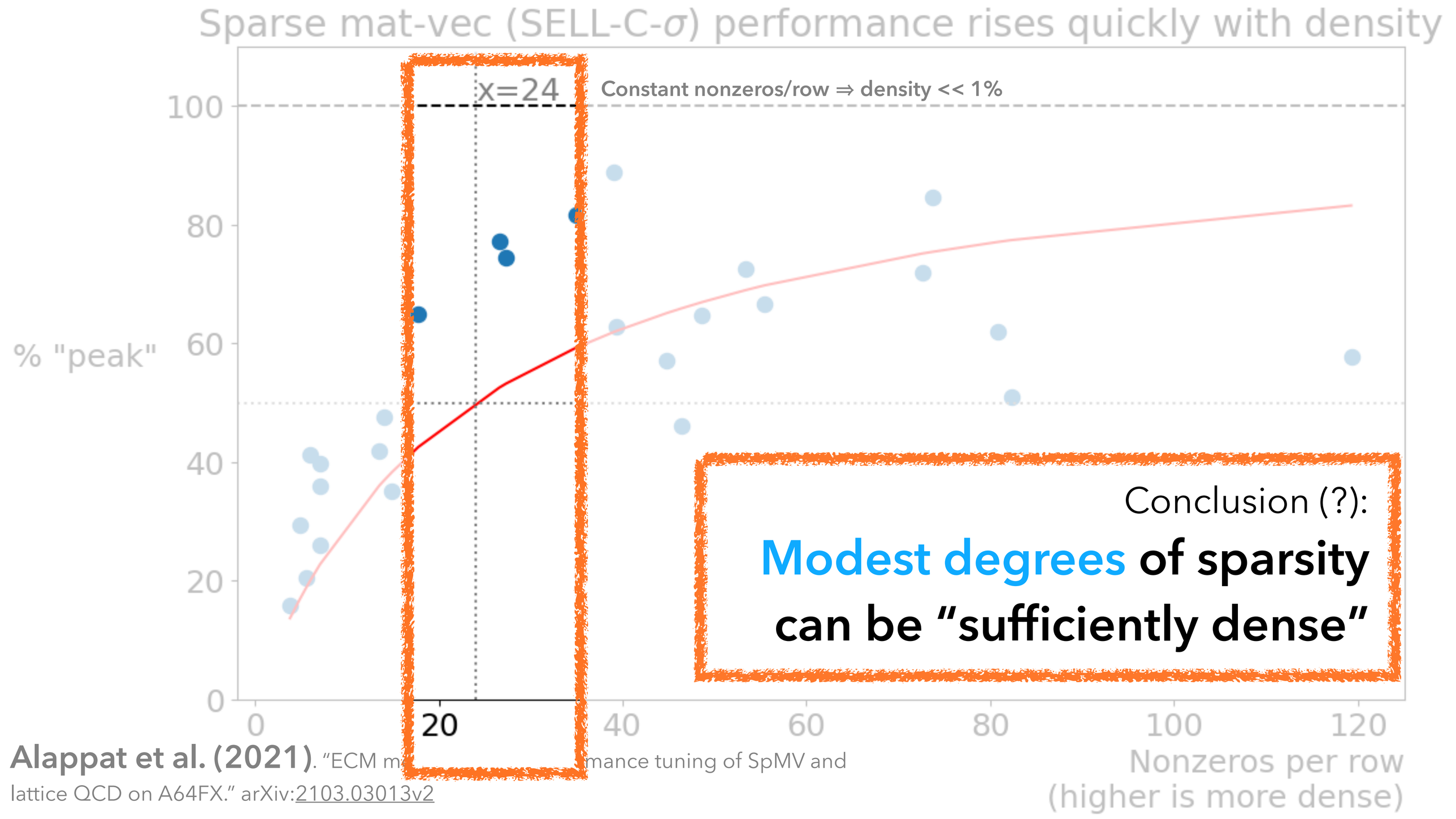
Frantar et al. (2023). "Scaling laws for sparsely-connected foundation models. arXiv:2309.08520v1



Alappat et al. (2021). "ECM modeling and performance tuning of SpMV and lattice QCD on A64FX." arXiv:2103.03013v2



Alappat et al. (2021). "ECM m mance tuning of SpMV and lattice QCD on A64FX." arXiv:2103.03013v2



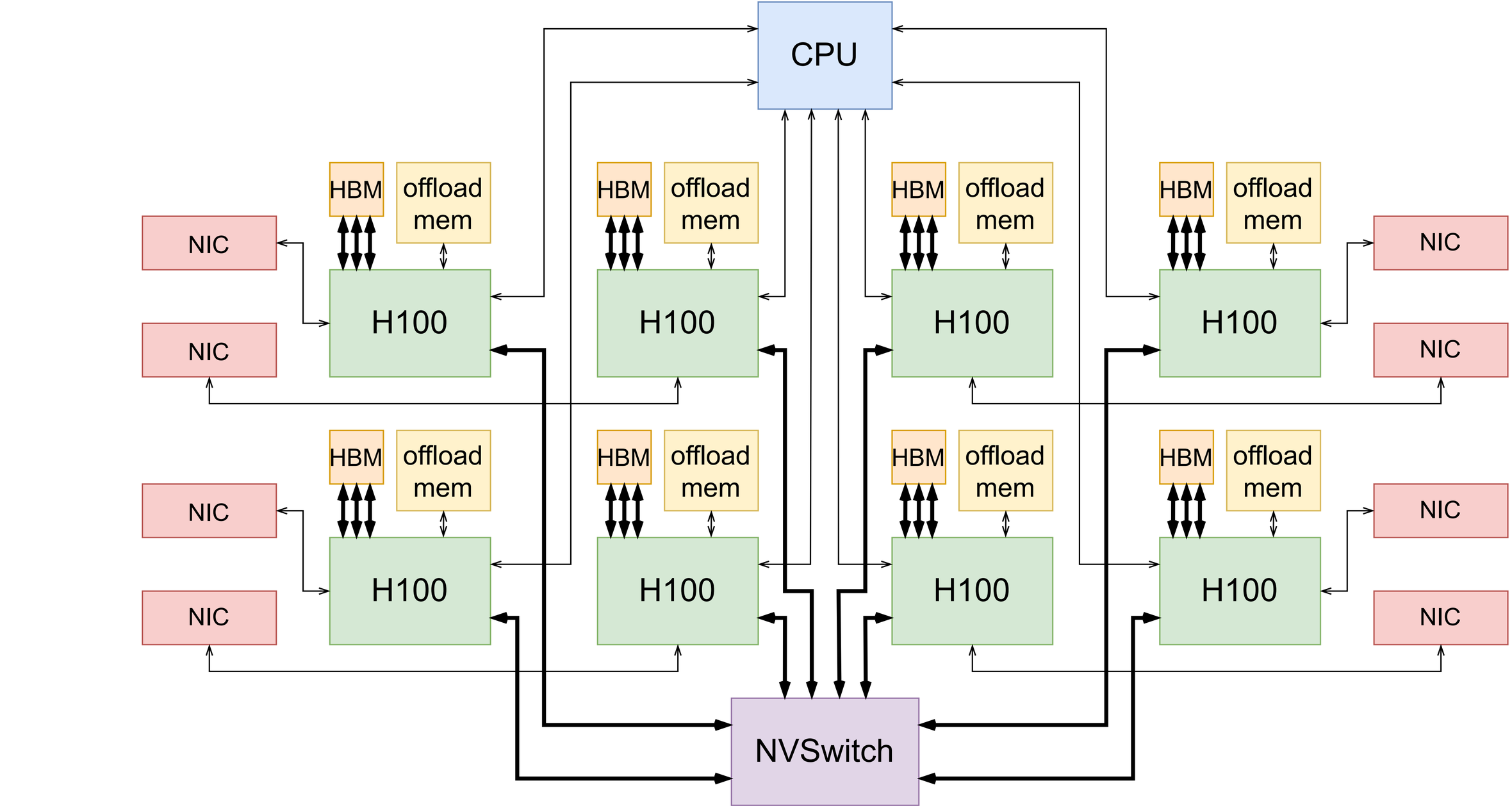
Alappat et al. (2021). "ECM m mance tuning of SpMV and lattice QCD on A64FX." arXiv:2103.03013v2

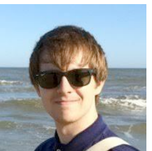
Conclusion so far:

Q: What system will be built for HPC?

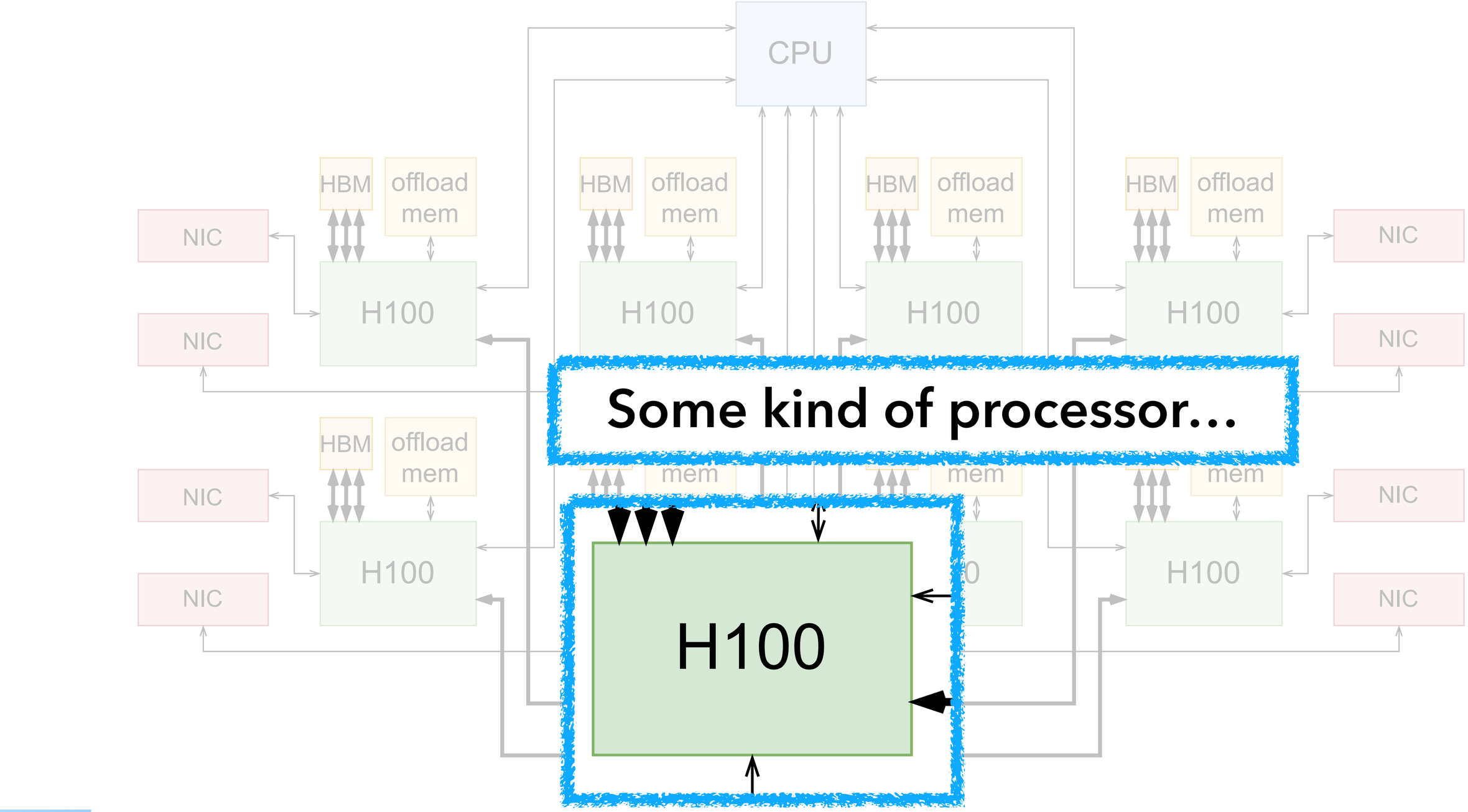
A: One for dense (and fake sparse) **foundation models.**

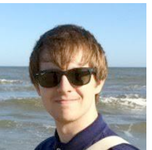
Q: What is an optimal machine for that case?



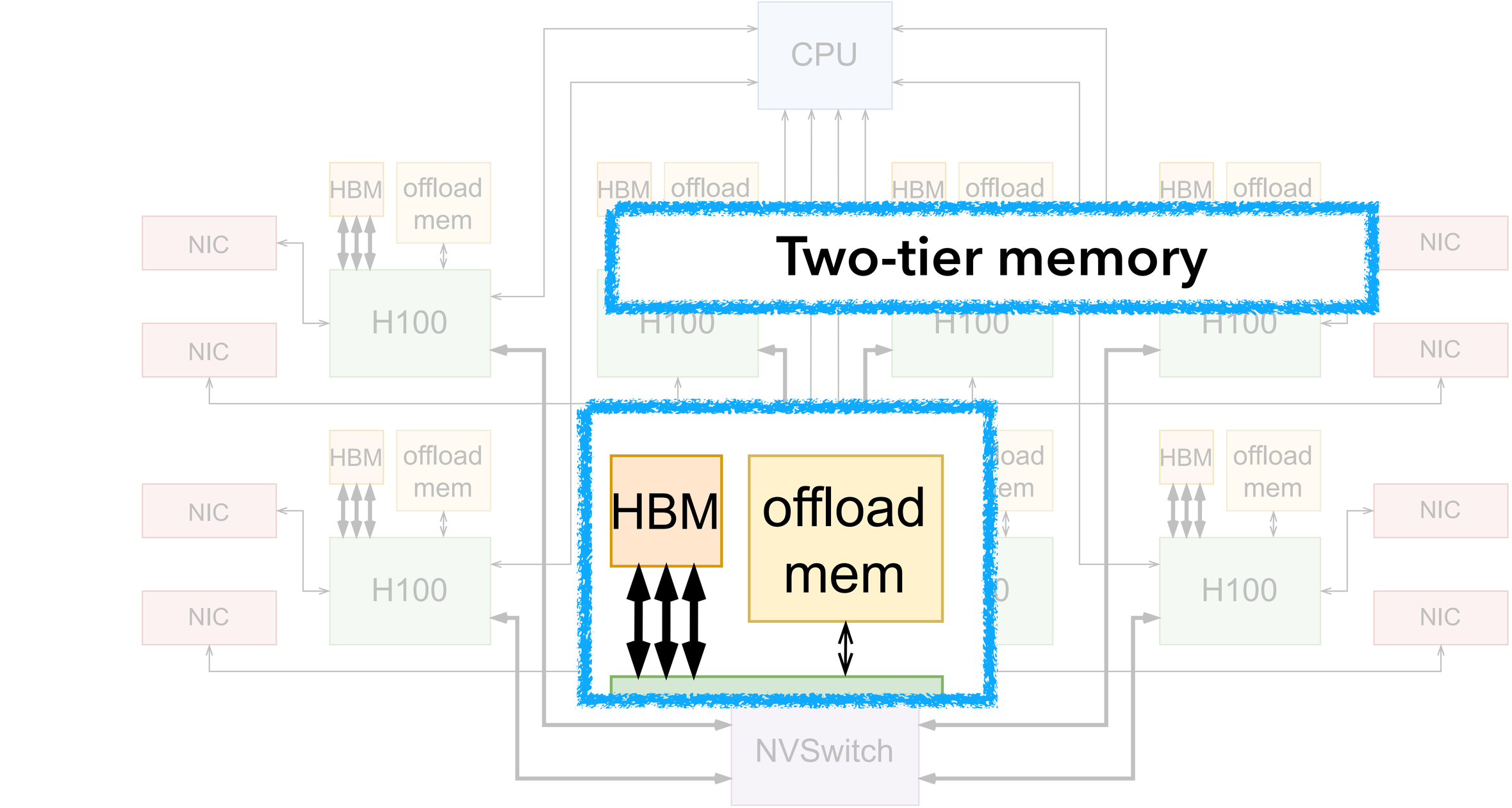


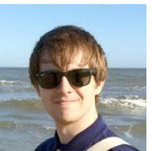
Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**, R. Vuduc** (SC23)



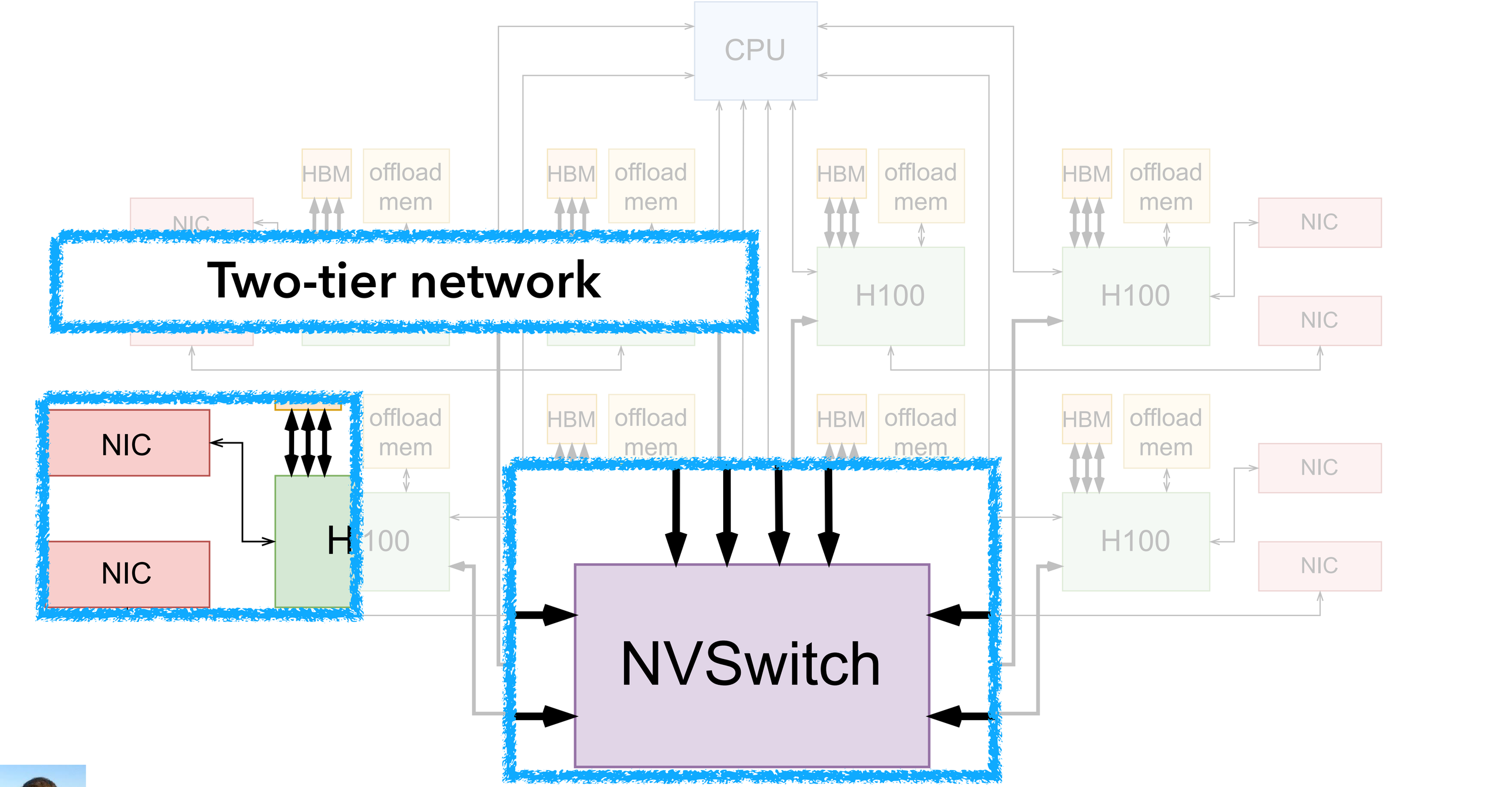


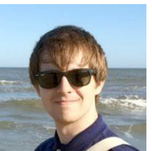
Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**,** **R. Vuduc** (SC23)





Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**, R. Vuduc** (SC23)





Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**,** **R. Vuduc** (SC23)

Canonical structure of a large language model

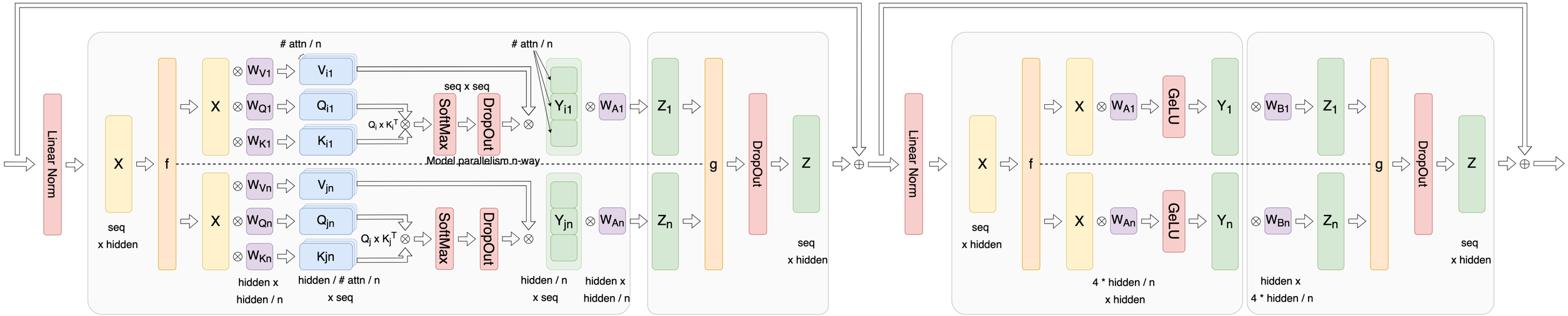
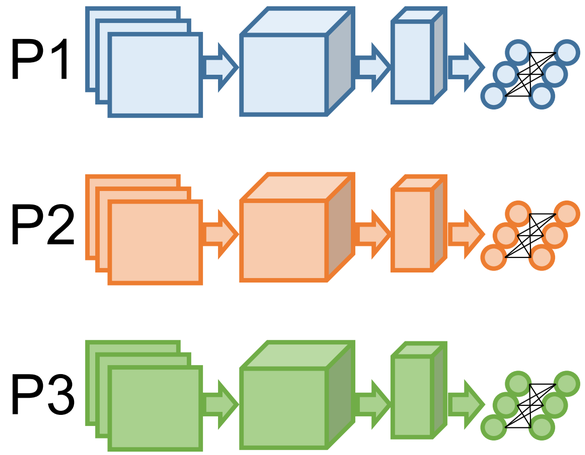


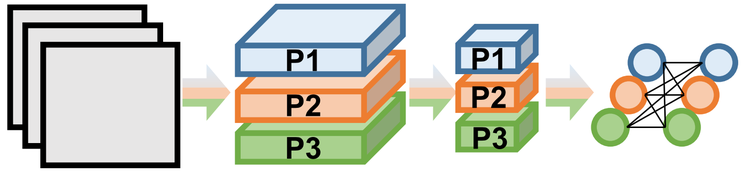
Figure 1: The transformer block structure of Megatron

Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**, R. Vuduc** (SC23)

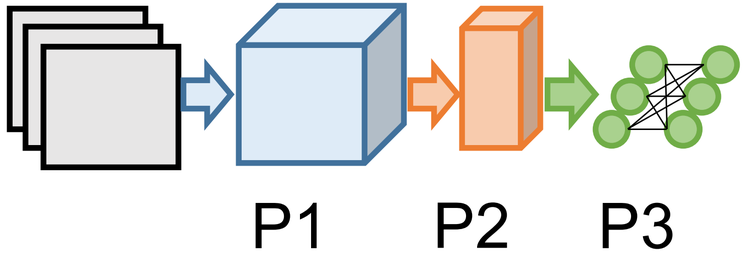
Myriad ways to map an LLM to a machine…



(a) Data Parallelism



(b) Model Parallelism

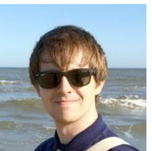


(c) Layer Pipelining

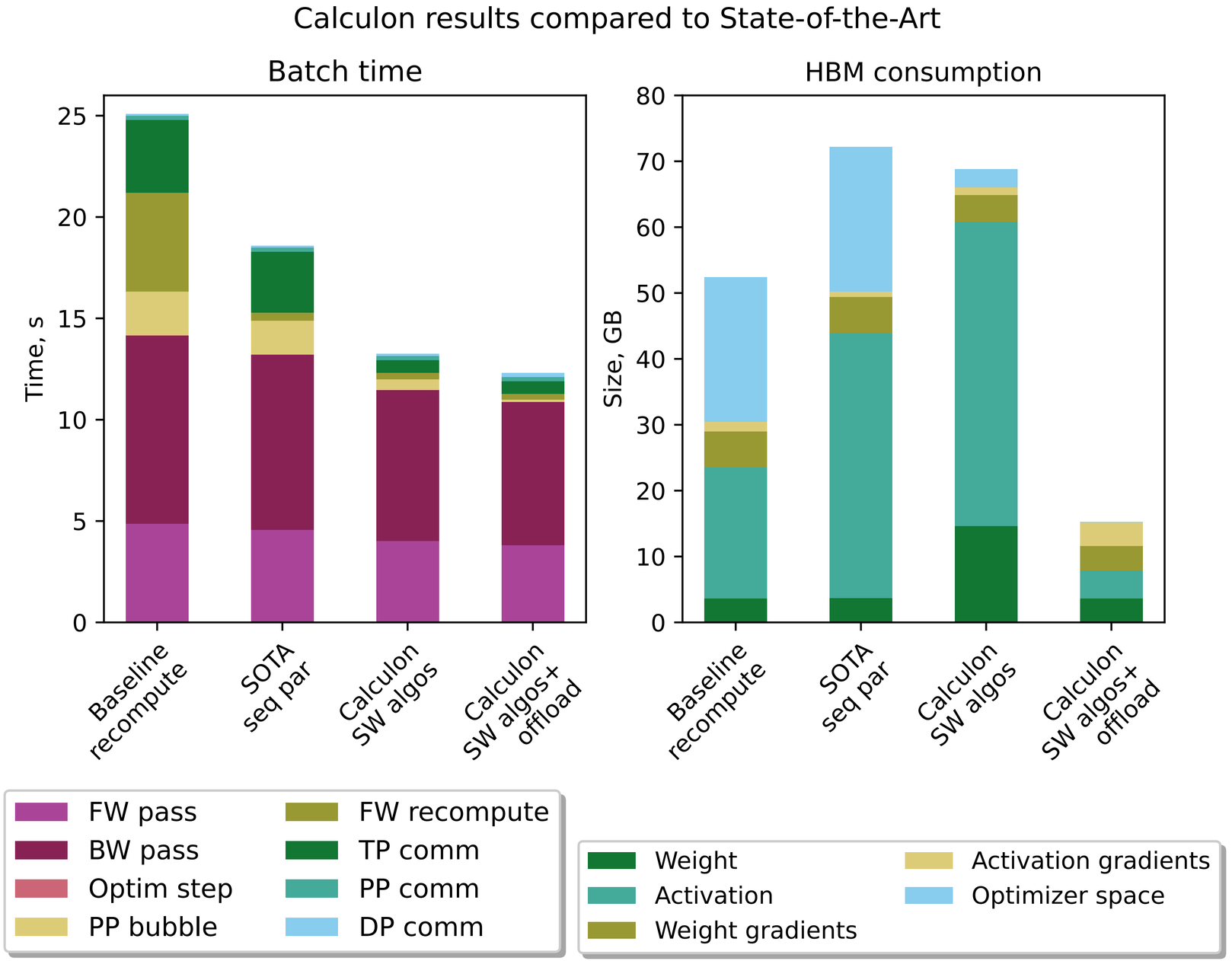
Fig. 14. Neural Network Parallelism Schemes

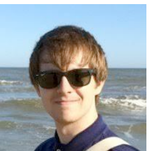
Ben-Nun & Hoefler (2019). "Demystifying parallel and distributed deep learning: an in-depth concurrency analysis." doi:10.1145/3320060

Optimization	Year	Related system	Comp time	Comp util	Mem time	Mem cap	Mem BW	Net time	Net BW	range
Data parallelism (DP) [61]	1989	network	–	↑	–	↑↑↑	–	↑	↑	1 .. batch
DP overlap [25]	2017	network	↑	↓	–	–	–	↓↓↓	–	true/false
Optimizer sharding [24]	2019	network	↓	–	–	↓↓	–	–	–	true/false
Recompute [5, 10]	2000	compute	↑↑	–	–	↓↓↓	–	–	–	full/attn/none
Fused layers [28]	2018	compute	–	↑↑	↓↓	↓↓	↓	–	–	true/false
Microbatch training [13]	2019	compute	–	↑↑	–	↑↑↑	–	–	–	1 .. batch/DP
Pipeline parallelism (PP) [7, 13]	2012	network	↑	↓↓	–	↓↓	–	↑	↑	1 .. blocks
PP 1F1B schedule [7, 32]	2012	network	–	–	–	↓↓	–	–	–	true/false
PP interleaving [33]	2021	network	↓	↑↑	–	↑	–	↑	↑↑	1 .. blocks/PP
PP RS + AG [21]	2022	network	–	–	–	–	–	↓	↓↓	true/false
Tensor parallelism (TP) [7, 22, 49]	2012	network	↓↓	↓	–	↓↓	↓↓	↑↑↑	↑↑↑	1 .. attn
TP RS + AG instead AR [33]	2021	network	–	–	↑	↑	–	↓	↓	true/false
Sequence parallelism (SP) [21]	2022	network	↓	–	↓	↓↓	↓	↑	↑	true/false
TP redo for SP [21]	2022	network	–	–	–	↓	–	↑	↑	true/false
TP overlap [58]	2022	network	↑	↓	–	–	–	↓↓	–	true/false
Weight offload [48]	2021	memory	–	–	↑	↓↓↓	↑	–	–	true/false
Activation offload [48]	2021	memory	–	–	↑	↓↓↓	↑	–	–	true/false
Optimizer offload [48]	2021	memory	–	–	↑	↓	↑	–	–	true/false

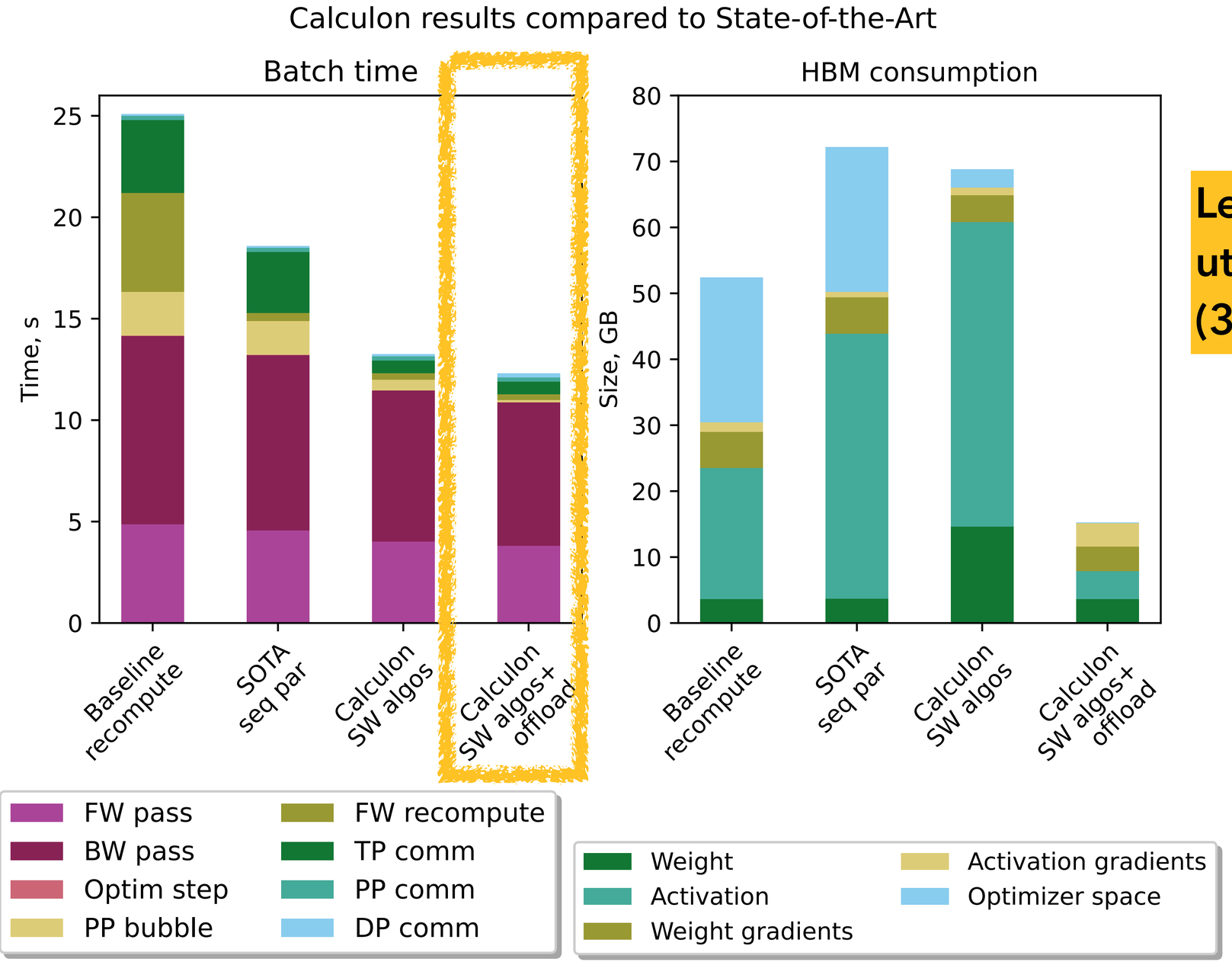


Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**,** **R. Vuduc** (SC23)

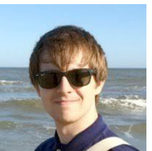




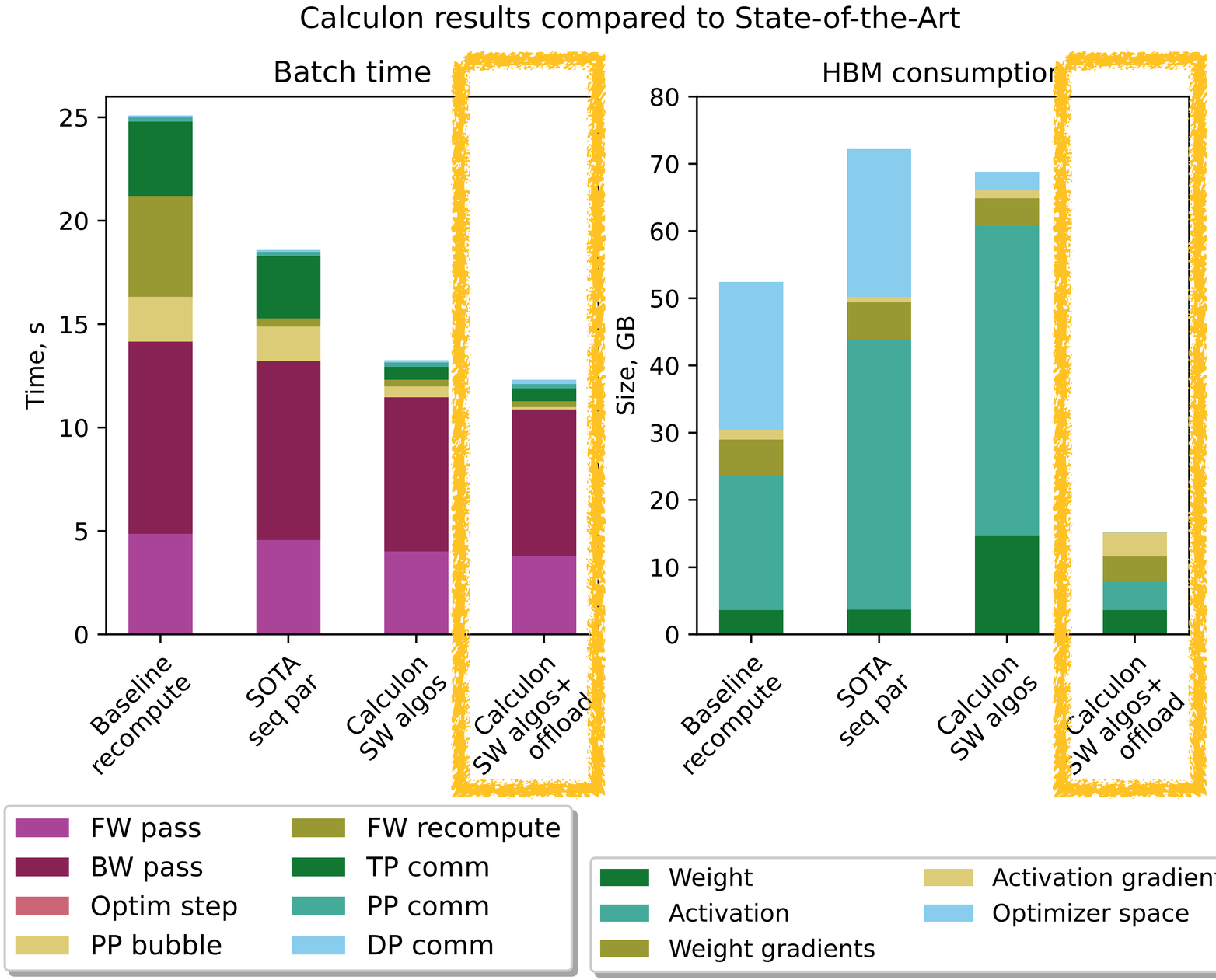
Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**, R. Vuduc** (SC23)



Less time, higher flop utilization
(38% → 75%)

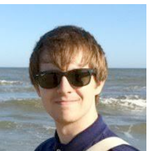


Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**, R. Vuduc** (SC23)



Less time, higher flop utilization (38% → 75%)

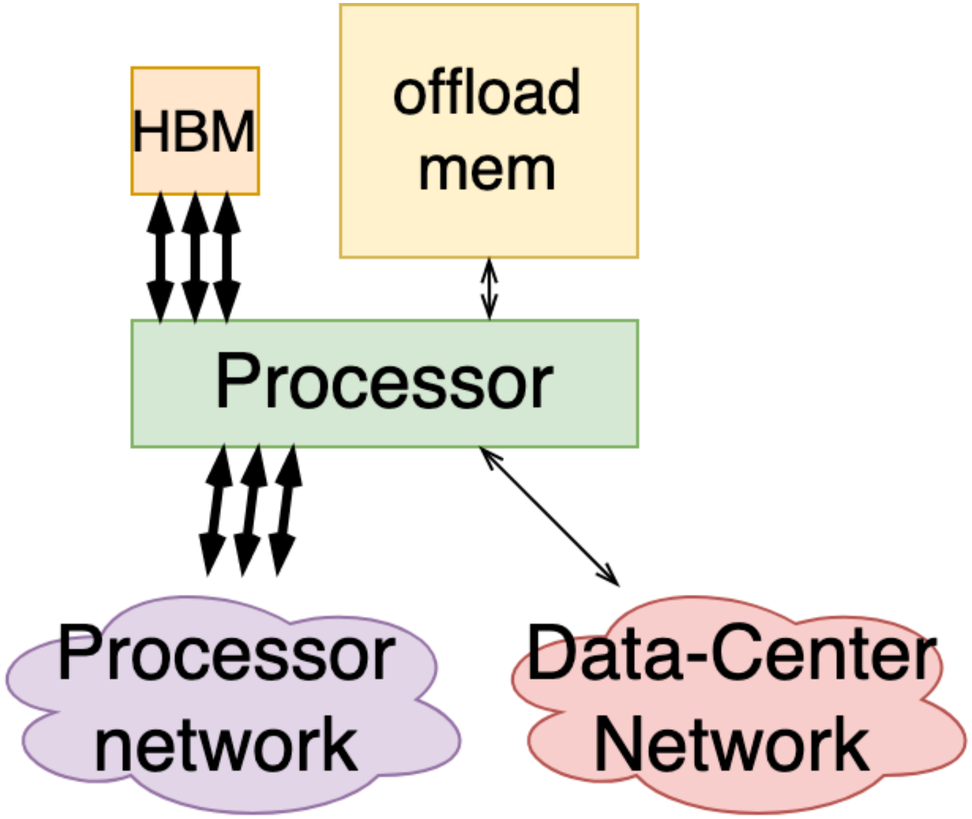
Less HBM – trade for more "slow" memory



Mike Isaev (GT Ph.D.)**, Nic McDonald** (NVIDIA)**, R. Vuduc** (SC23)

Q: What is an optimal machine for foundation models?

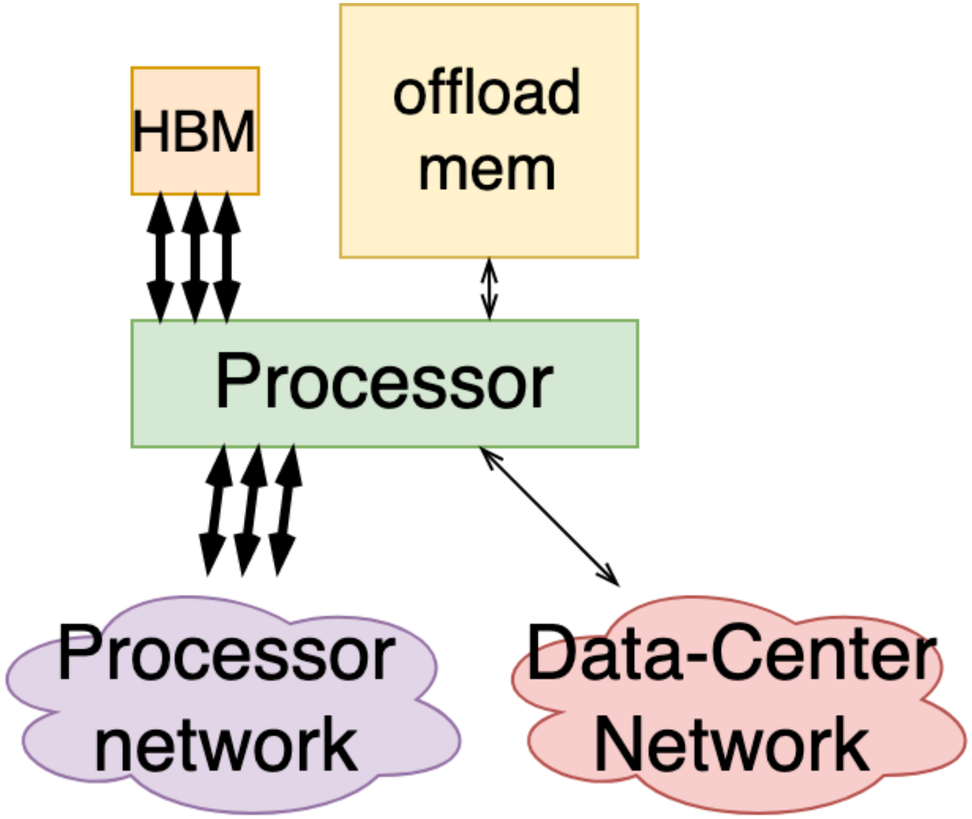
A: One tuned for dense compute (“big” procs) **and slow communication**, i.e., a little HBM, a lot of slow capacity mem, fast on-node network, slow internode network.



Q: What is an optimal machine for foundation models?

A: One tuned for dense compute (“big” procs) **and slow communication**, i.e., a little HBM, a lot of slow capacity mem, fast on-node network, slow internode network.

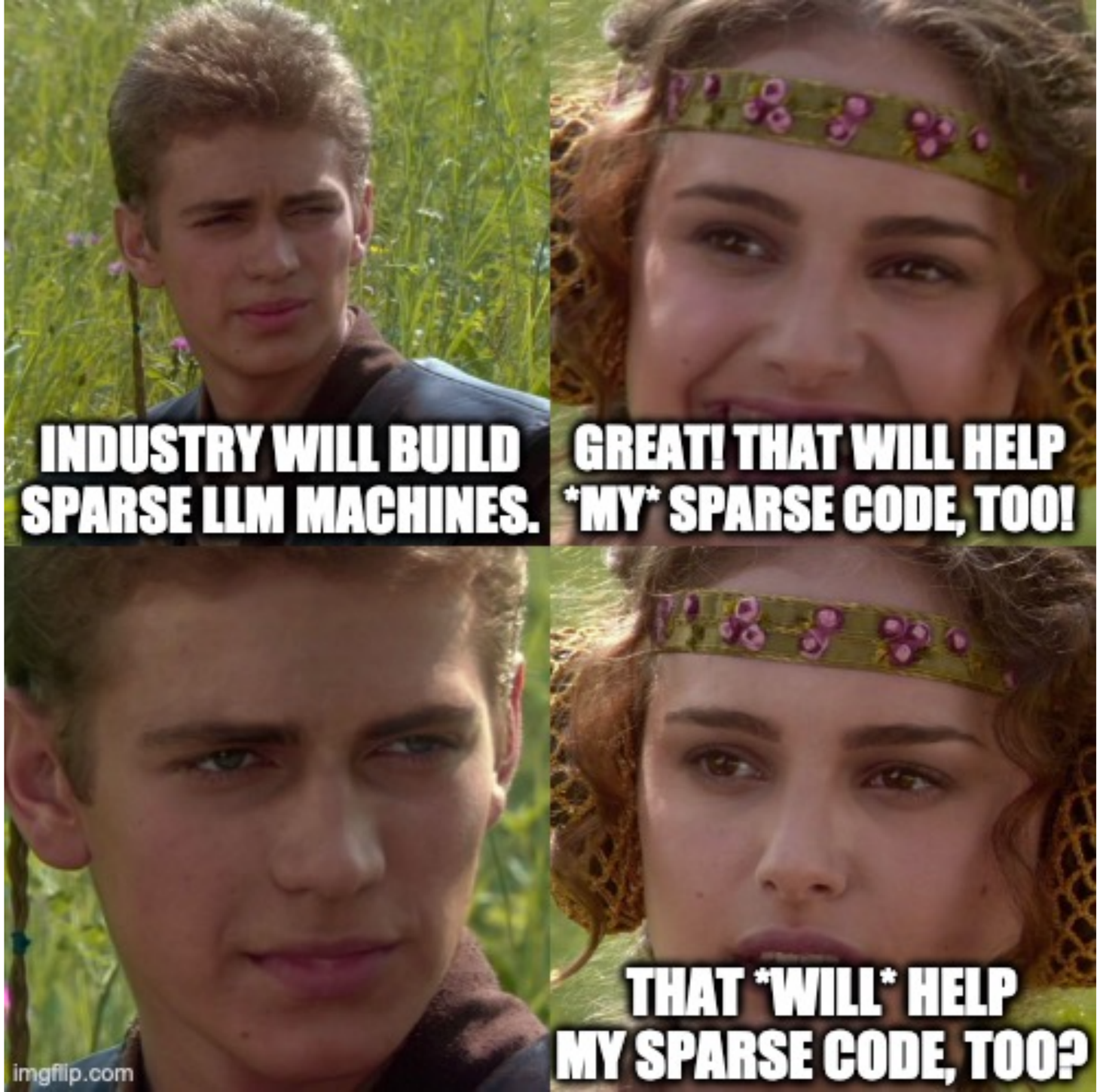
Will this design meet your needs?



So, now what?

DESPITE EVERYTHING I JUST SAID,
THE FUTURE *SHOULD* BE SPARSE!

P.S.: Weren't you going to talk about APSP?



Recall:

$$\mathcal{O}(N^2) \longrightarrow \mathcal{O}(N)$$

Reduces **energy**: fewer flops, less storage

Recall:

$$\mathcal{O}(N^2) \longrightarrow \mathcal{O}(N)$$

% time communicating increases

Algorithms for 2D Poisson Equation with N unknowns

Algorithm	Serial	PRAM	Memory	#Procs	(Keyes '04)
° Dense LU	N^3	N	N^2	N^2	
° Band LU	N^2	N	$N^{3/2}$	N	*1947*
° Jacobi	N^2	N	N	N	*1950*
° Explicit Inv.	N^2	log N	N^2	N^2	
° Conj.Grad.	$N^{3/2}$	$N^{1/2}$ *log N	N	N	*1971*
° RB SOR	$N^{3/2}$	$N^{1/2}$	N	N	
° Sparse LU	$N^{3/2}$	$N^{1/2}$	N*log N	N	*~ 1970s*
° FFT	N*log N	log N	N	N	
° Multigrid	N	$\log^2$ N	N	N	*1984*
° Lower bound	N	log N	N		

PRAM is an idealized parallel model with zero cost communication

https://sites.google.com/lbl.gov/cs267-spr2023

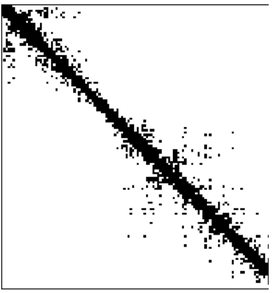
Algorithms for 2D Poisson Equation with N unknowns

Algorithm	Serial	PRAM	Memory	#Procs	(Keyes '04)
° Dense LU	N^3	N	N^2	N^2	
° Band LU	N^2	N	$N^{3/2}$	N	*1947*
° Jacobi	N^2	N	N	N	*1950*
° Explicit Inv.	N^2	log N	N^2	N^2	
° Conj.Grad.	$N^{3/2}$	$N^{1/2}$ *log N	N	N	*1971*
° RB SOR	$N^{3/2}$	$N^{1/2}$	N	N	
° Sparse LU	$N^{3/2}$	$N^{1/2}$	N*log N	N	*~ 1970s*
° FFT	N*log N	log N	N	N	
° Multigrid	N	$\log^2$ N	N	N	*1984*
° Lower bound	N	log N	N		

PRAM is an idealized parallel model with zero cost communication

https://sites.google.com/lbl.gov/cs267-spr2023

Suggestion: **Consider sparse LU (+ APSP) as a hardware design target** (dense-ish & sparse-ish)

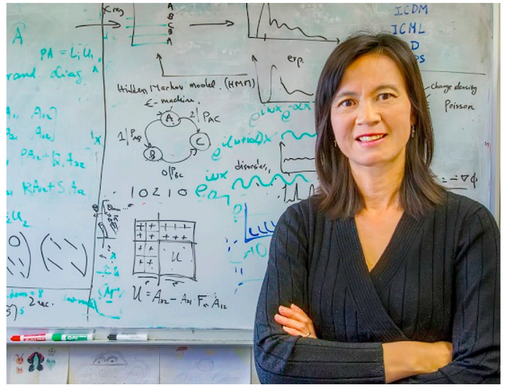


An algorithm

A **communication-avoiding** sparse direct solver

Thesis: Significant, and even asymptotic, improvements in the strong scaling of sparse direct solvers for linear systems and all-pairs shortest paths are possible by trading more storage for less communication.

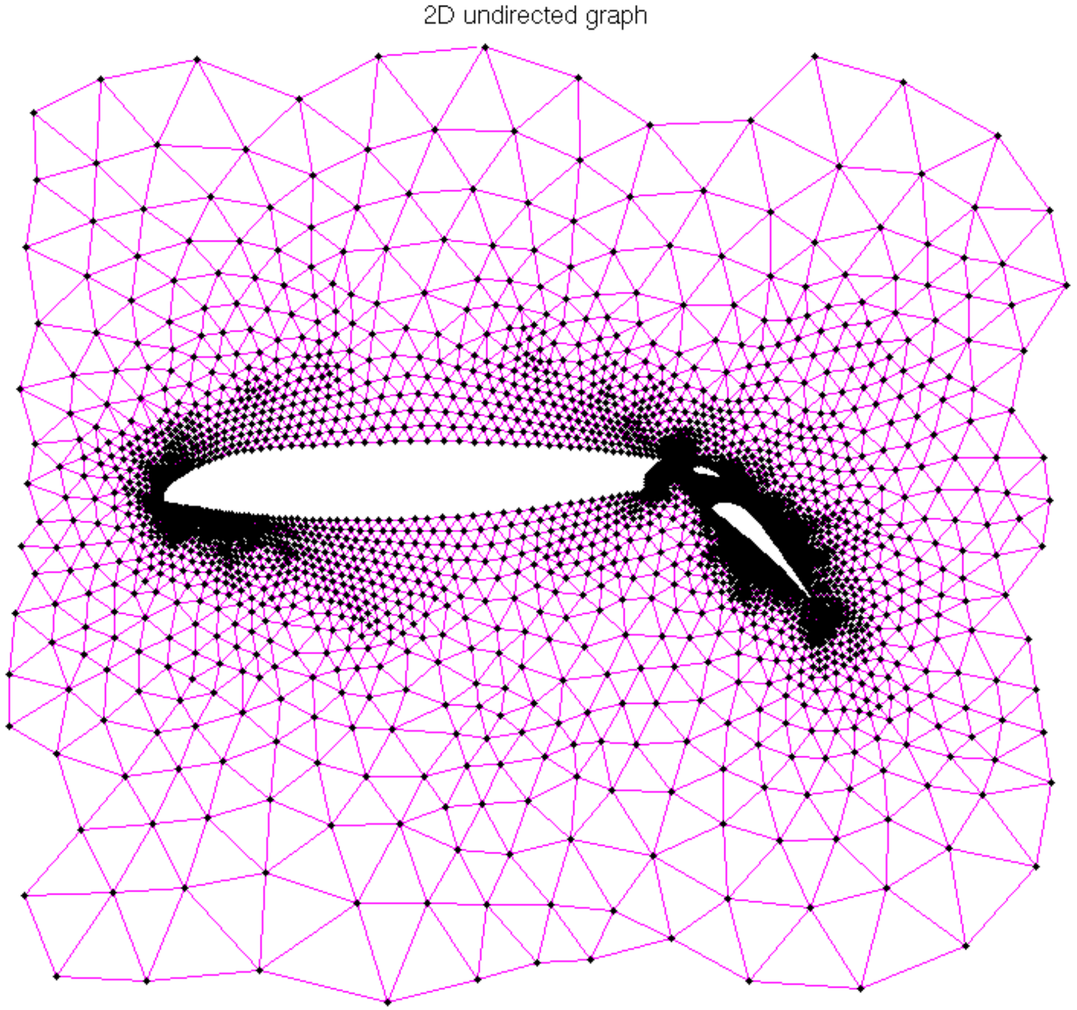
Piyush Sao
@piyush314 / ORNL

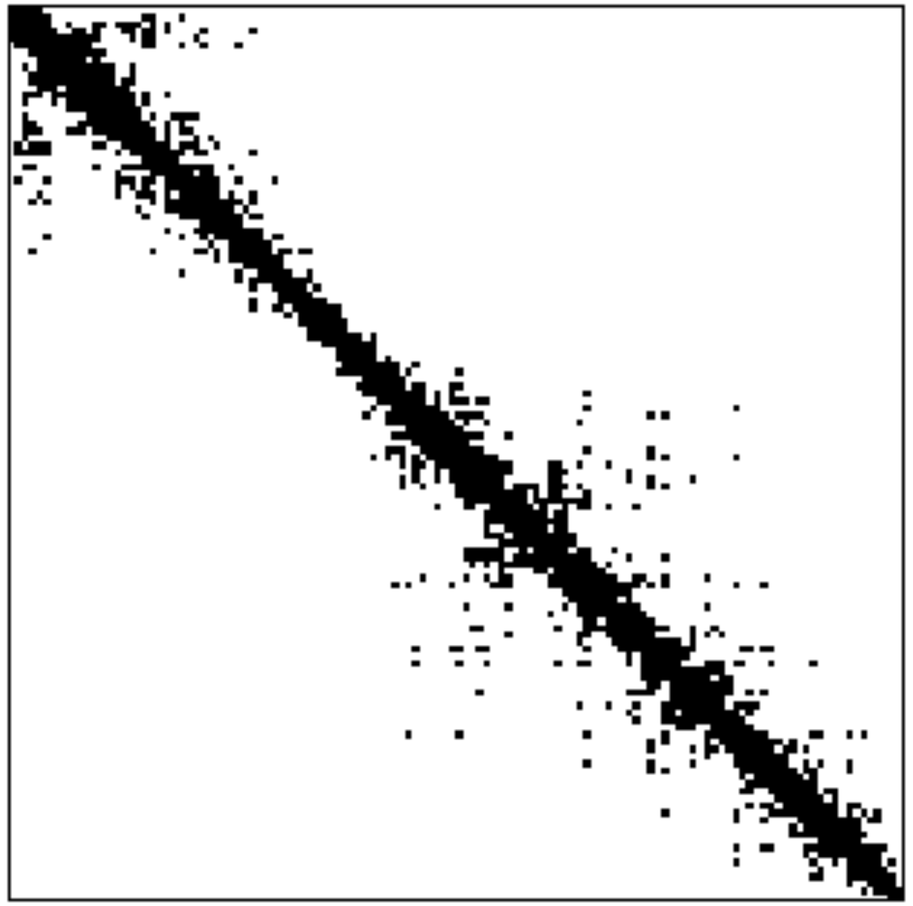


Xiaoye (Sherry) Li
LBNL

Ramki Kannan
ORNL

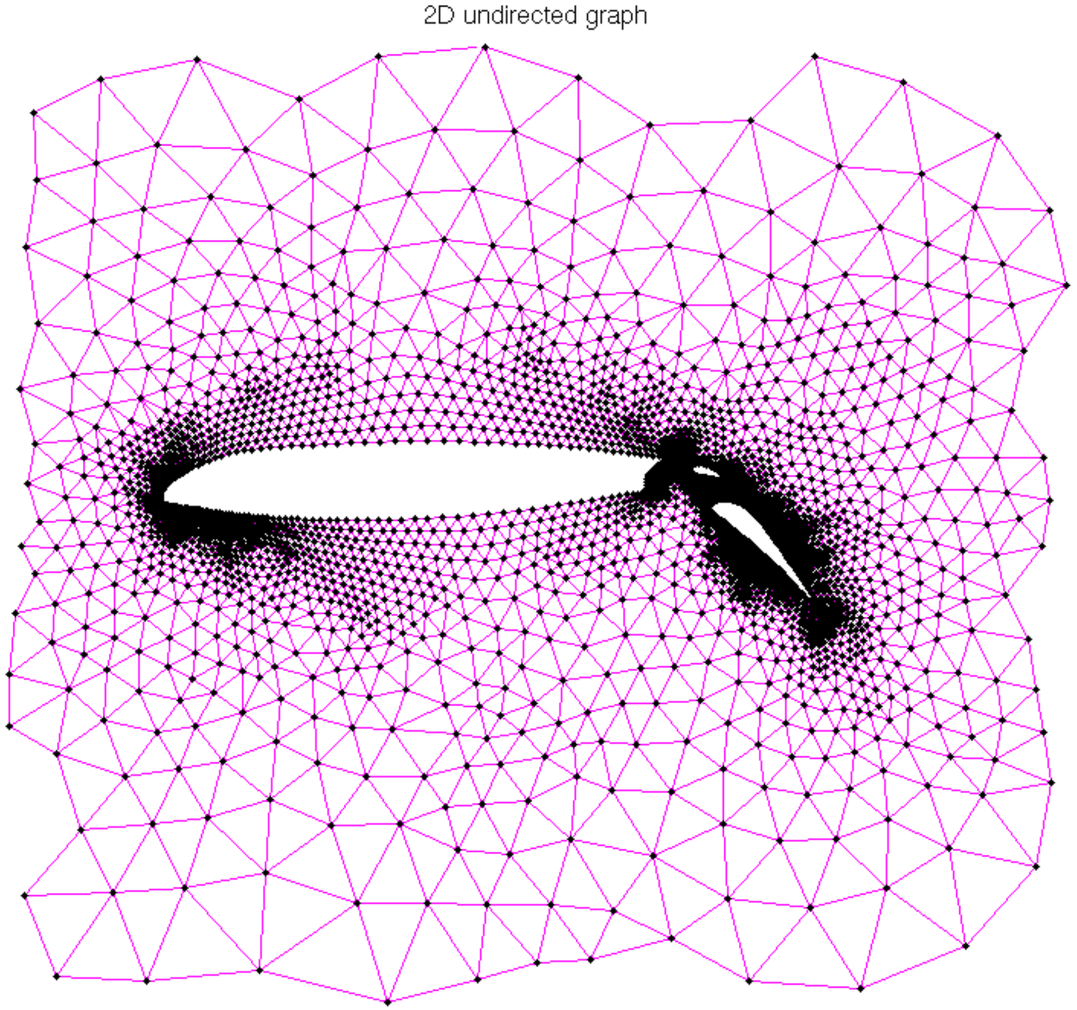
"A communication-avoiding 3D algorithm for sparse LU factorization on heterogeneous systems." JPDC'19
+ a bunch of follow-on papers

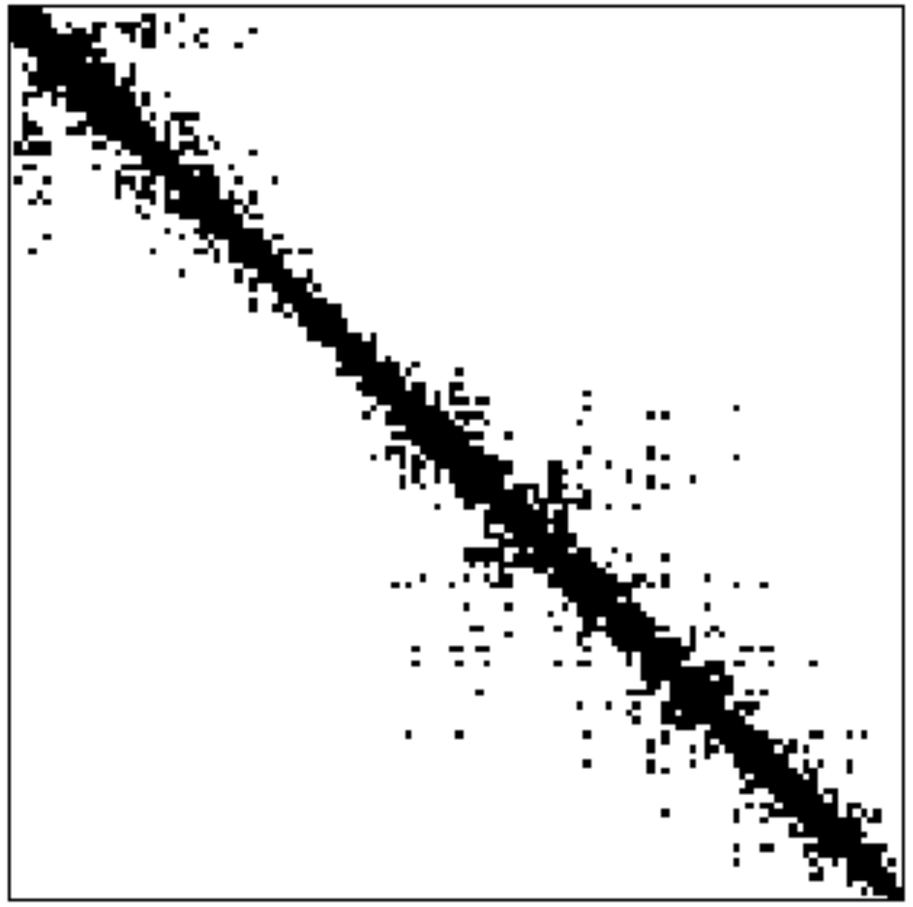




$$Ax = b$$

$$\text{nnz}(A) = \mathcal{O}(N)$$



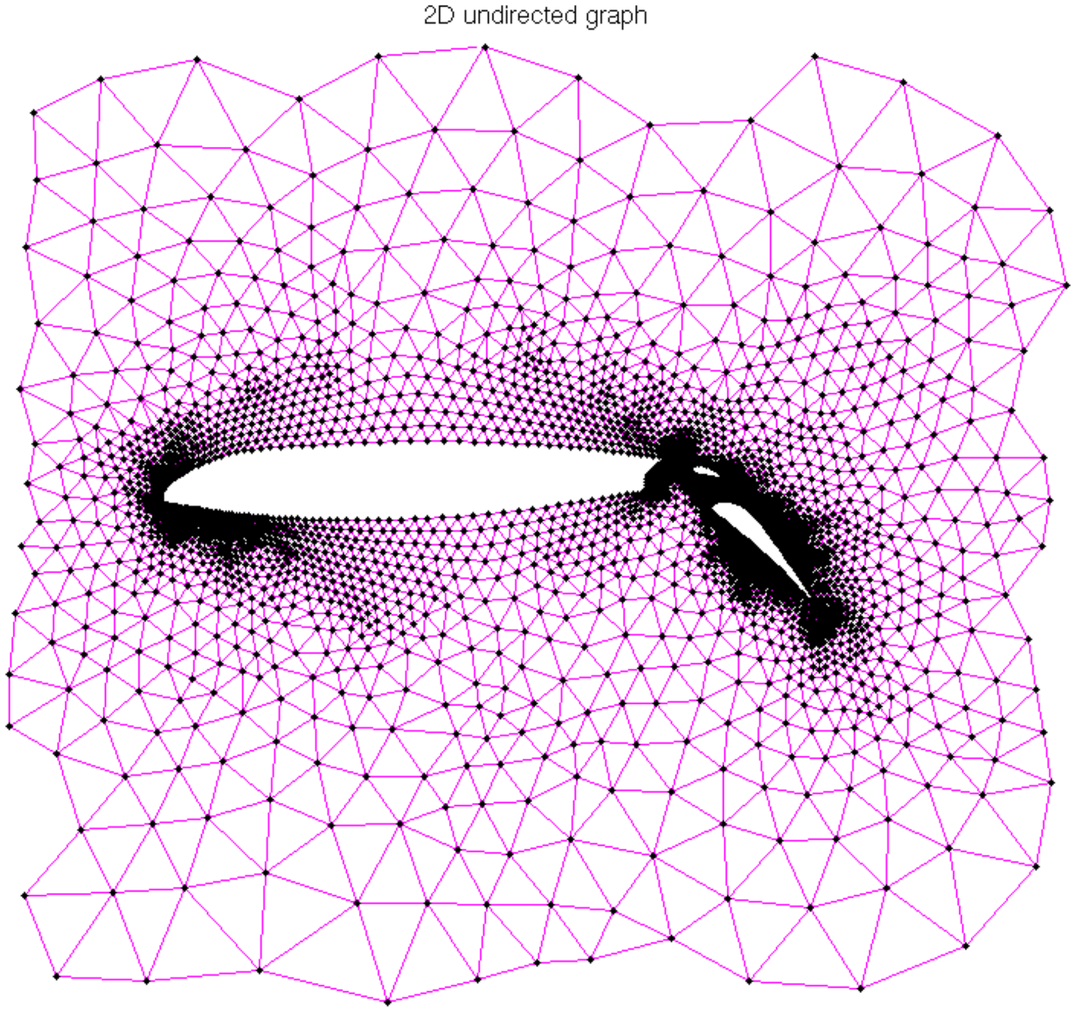


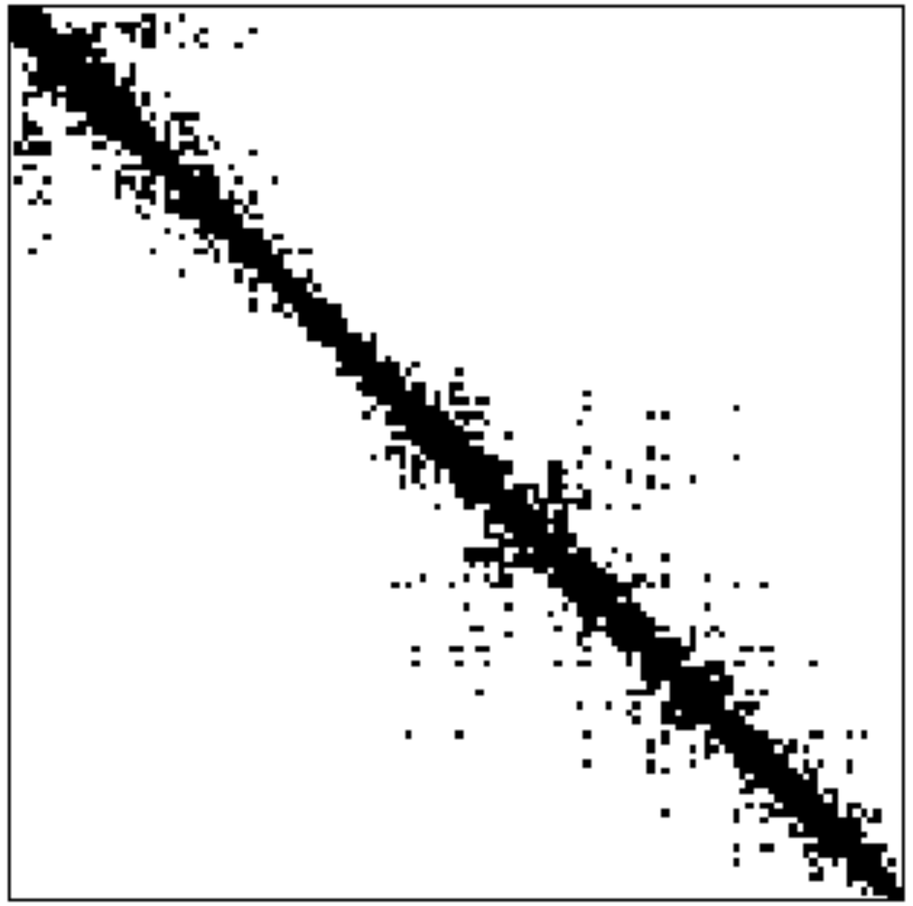
$$PA = LU$$

L = unit lower triangular

U = upper triangular

P = permutation (pivoting)





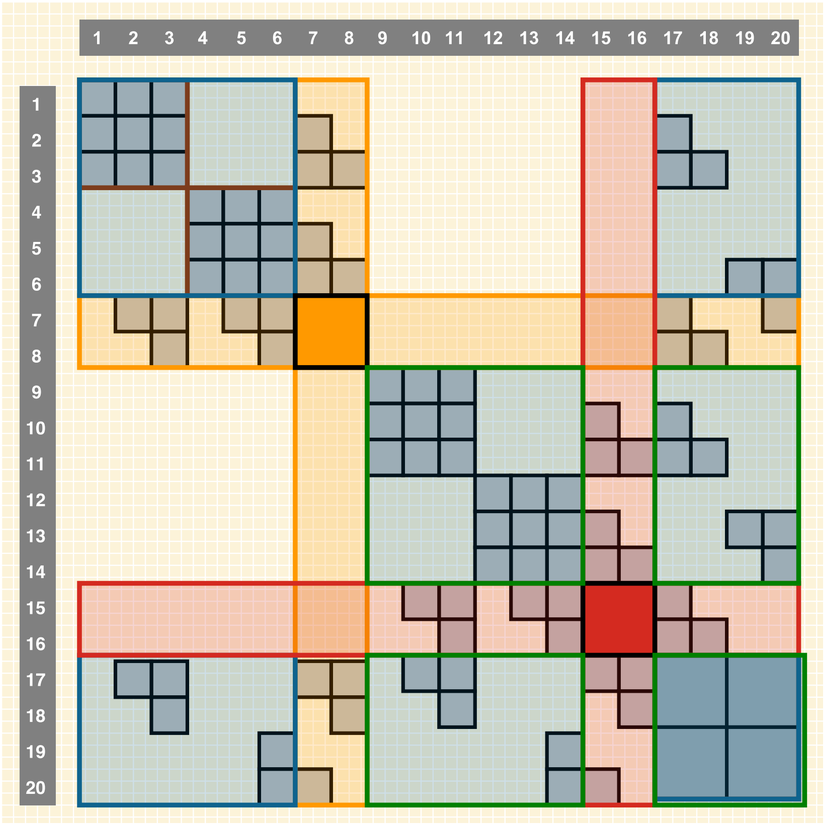
$$Ly = Pb \quad \text{(forward)}$$

$$Ux = y \quad \text{(backward)}$$

Aside: All-pairs shortest paths
~ LU but in the tropical semiring

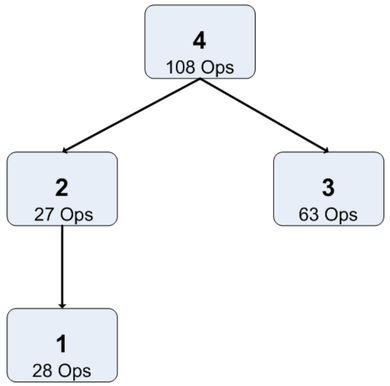
FLOYDWARSHALL(W)

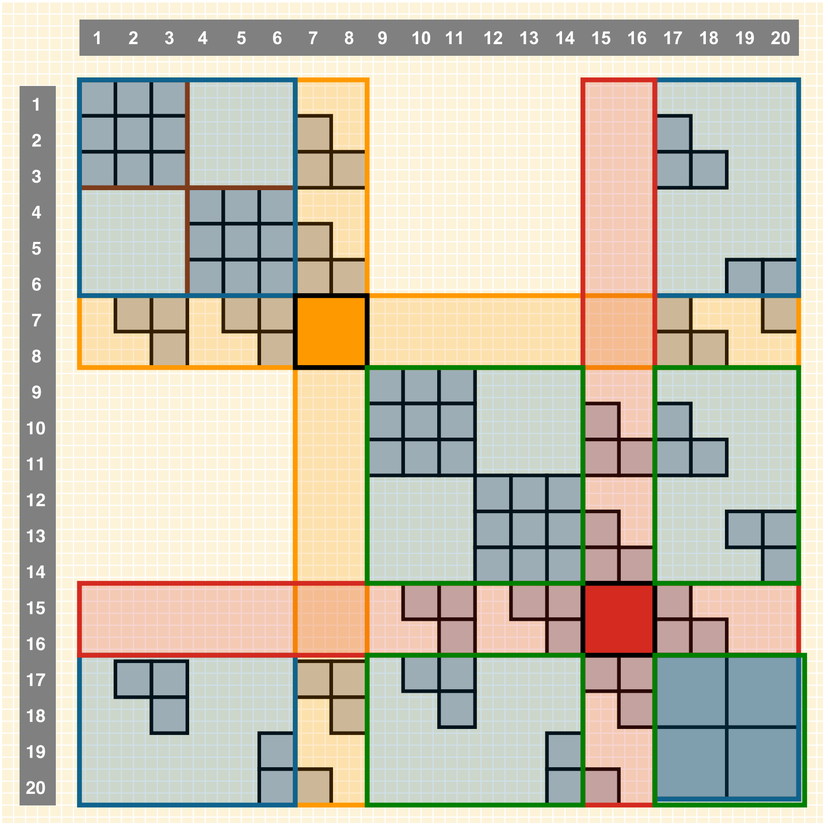
1 **for** $1 \leq i, j \leq n$
2 $\quad c_{i,j} \leftarrow w_{i,j}$
3 **for** $1 \leq r \leq n$
4 $\quad$ **for** $1 \leq i, j \leq n$
5 $\quad\quad c_{i,j} \leftarrow c_{i,j} \oplus [c_{i,r} \odot c_{r,j}]$
6 **return** $C \equiv [c_{i,j}]$



***Same machinery applies**! Reorderings, supernodes, elimination trees, data structures, distribution, GPUs, …*
Gordon Bell Finalist (SC20 & SC22)

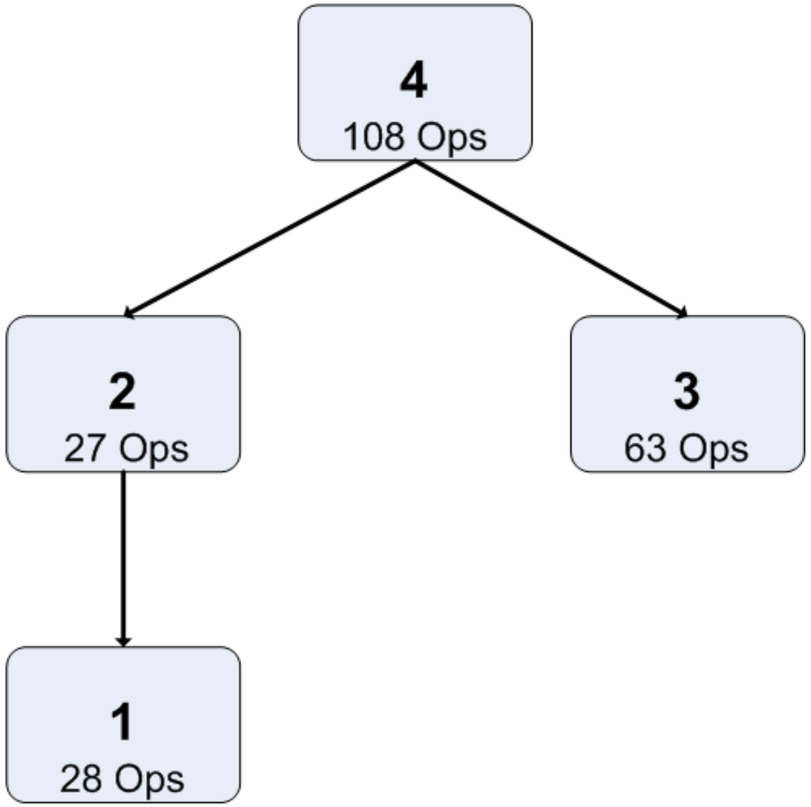
Sparse LU has rich computational structure

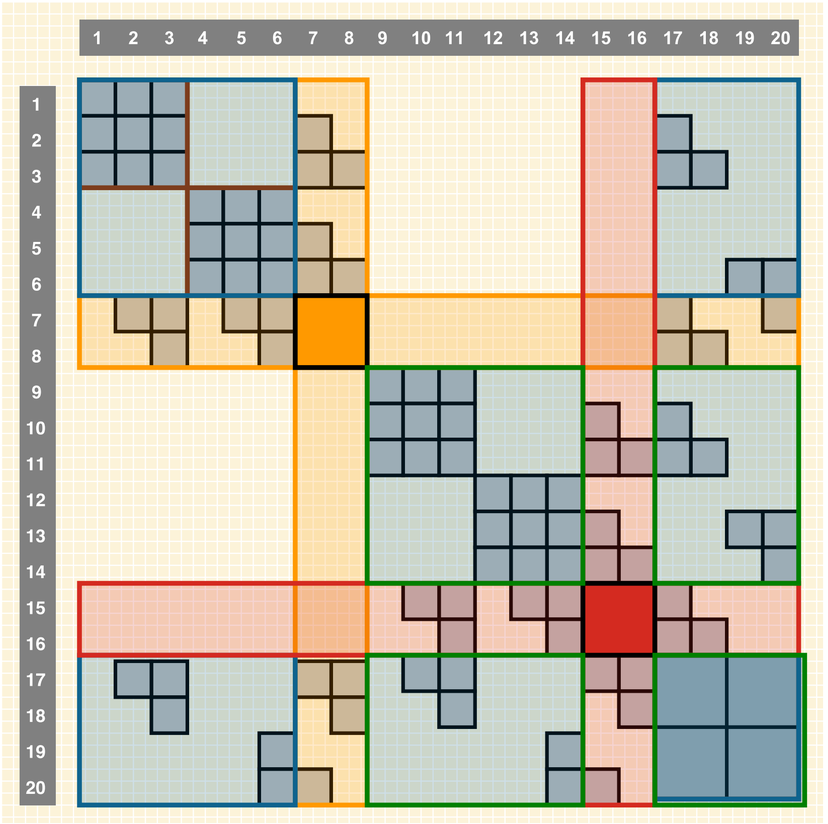




***Same machinery applies**! Reorderings, supernodes, elimination trees, data structures, distribution, GPUs, …*
Gordon Bell Finalist (SC20 & SC22)

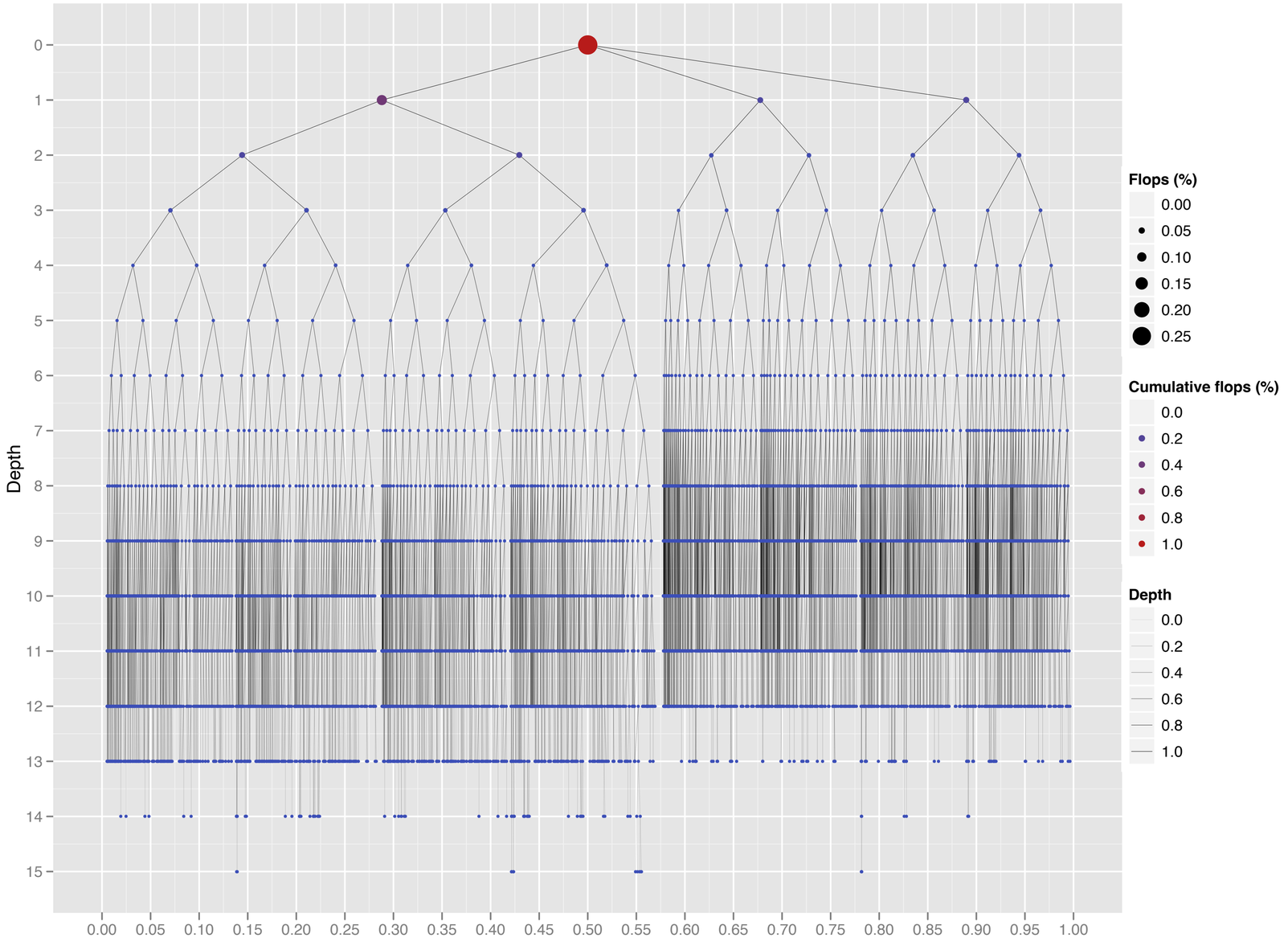
Parallel dependencies = “elimination tree”



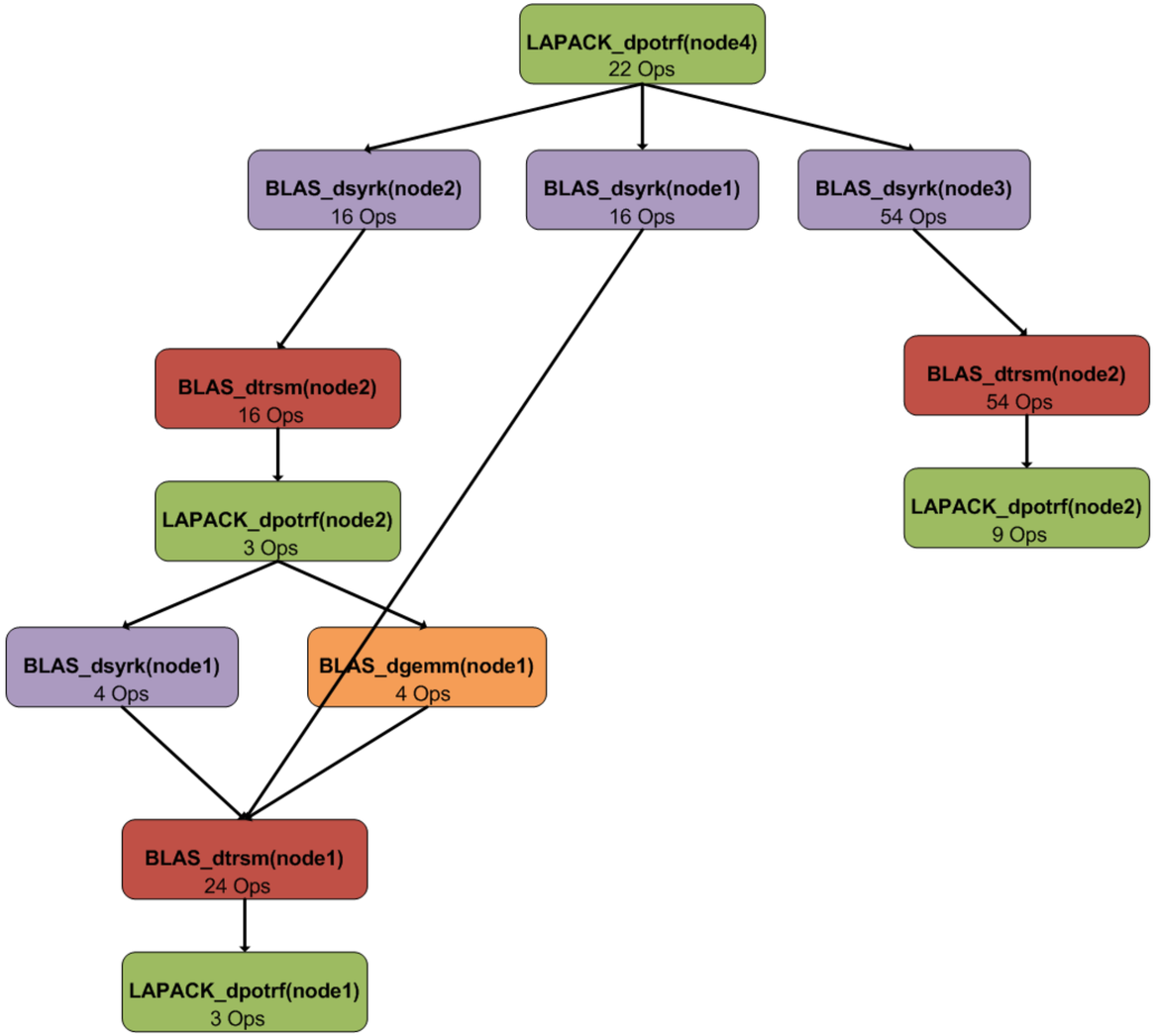


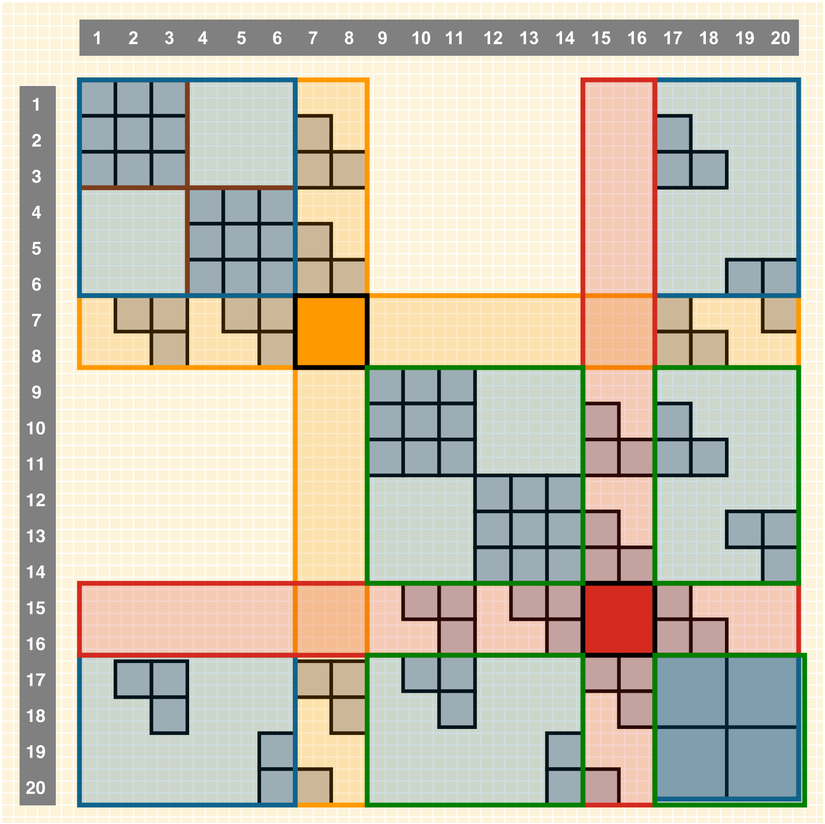
***Same machinery applies**! Reorderings, supernodes, elimination trees, data structures, distribution, GPUs, …*
Gordon Bell Finalist (SC20 & SC22)

E-tree is really a (fine-grained) task DAG



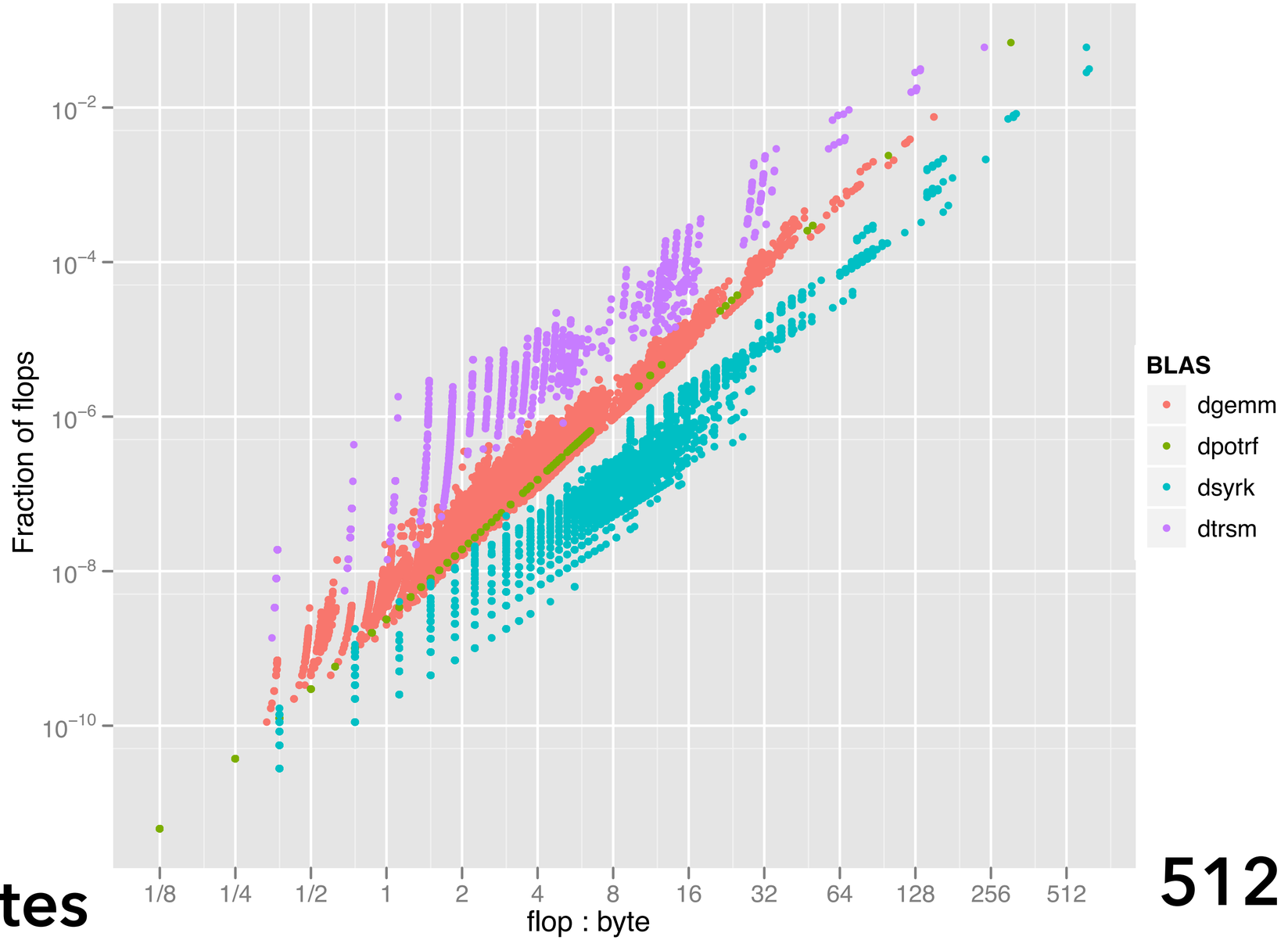
E-tree is really a (fine-grained) task DAG





Colored boxes = different BLAS-like operations

Tasks have a complex mix of intensities (flop:byte)

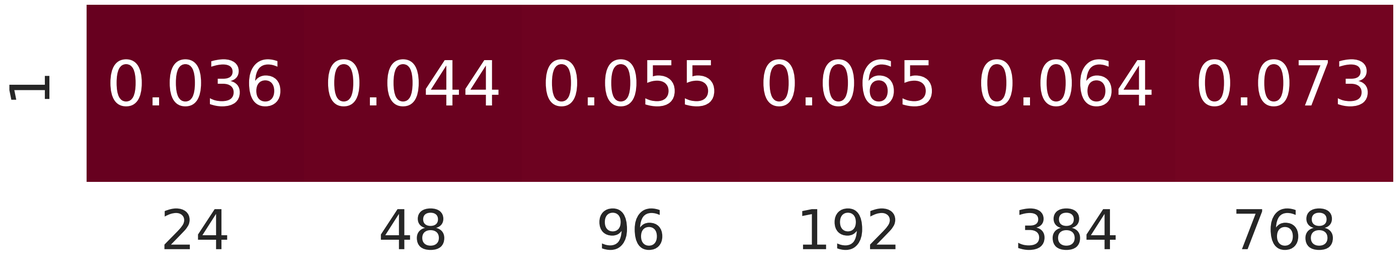


1 flop:8 bytes

512 f:b

Baseline: SuperLU_DIST

“2D” algorithm (strong scaling)



Teraflop/s

(32x procs → 2x speedup)

Baseline: SuperLU_DIST

 # MPI procs

Arranged in a 2-D logical grid

Teraflop/s

(32x procs → 2x speedup)

24 48 96 192 384 768

MPI processes (2D process grid; 4 cores / process)

Baseline: SuperLU_DIST

Example:

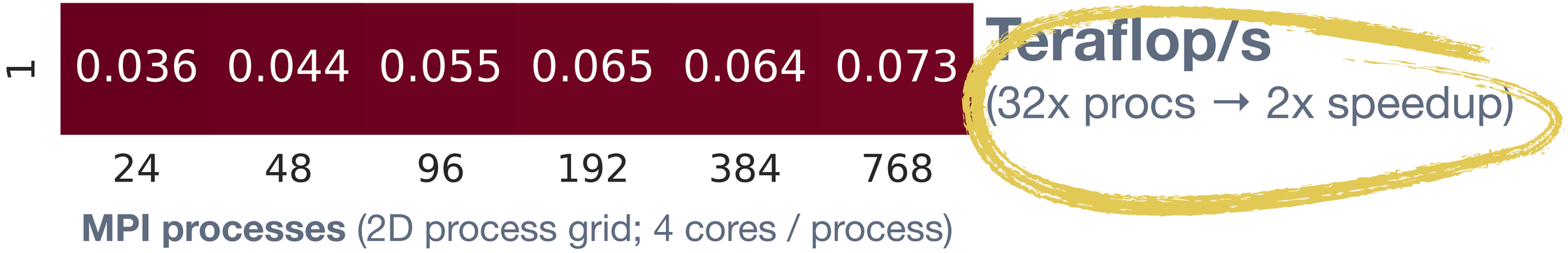
$$P_x \times P_y = 96$$

(Best configuration shown)

Teraflop/s
(32x procs → 2x speedup)

MPI processes (2D process grid; 4 cores / process)

Baseline: SuperLU_DIST



Sao: CA for sparse LU (2019–2022)

(communication avoidance)

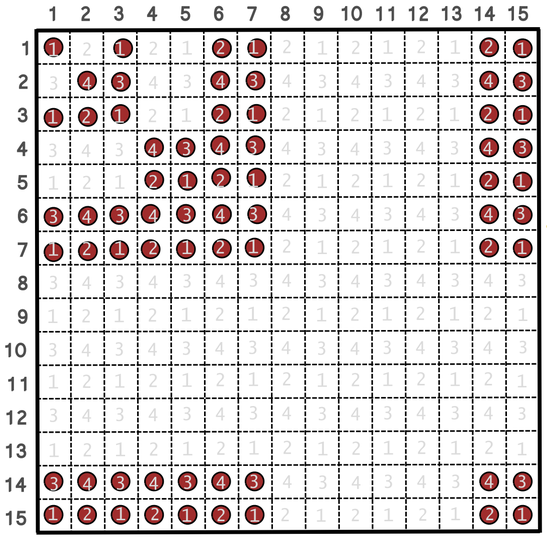
All known “3D-LU” algorithms† are for **dense LU**. They reduce communication volume but increase latency.

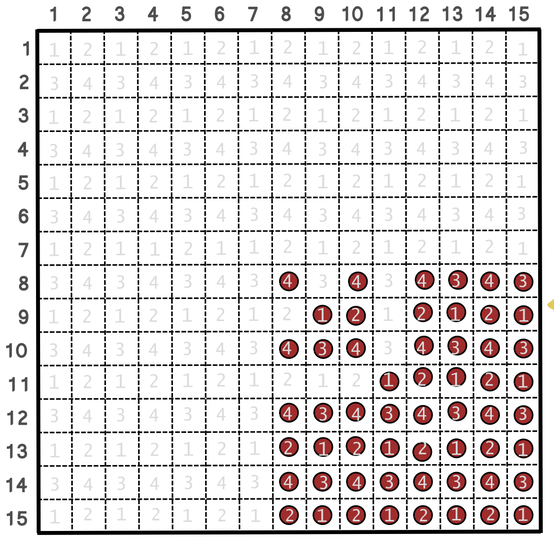
For sparse LU, we can **reduce** both the latency and bandwidth for “planar” problems **asymptotically**, and achieve constant-factor reductions for “non-planar” ones.

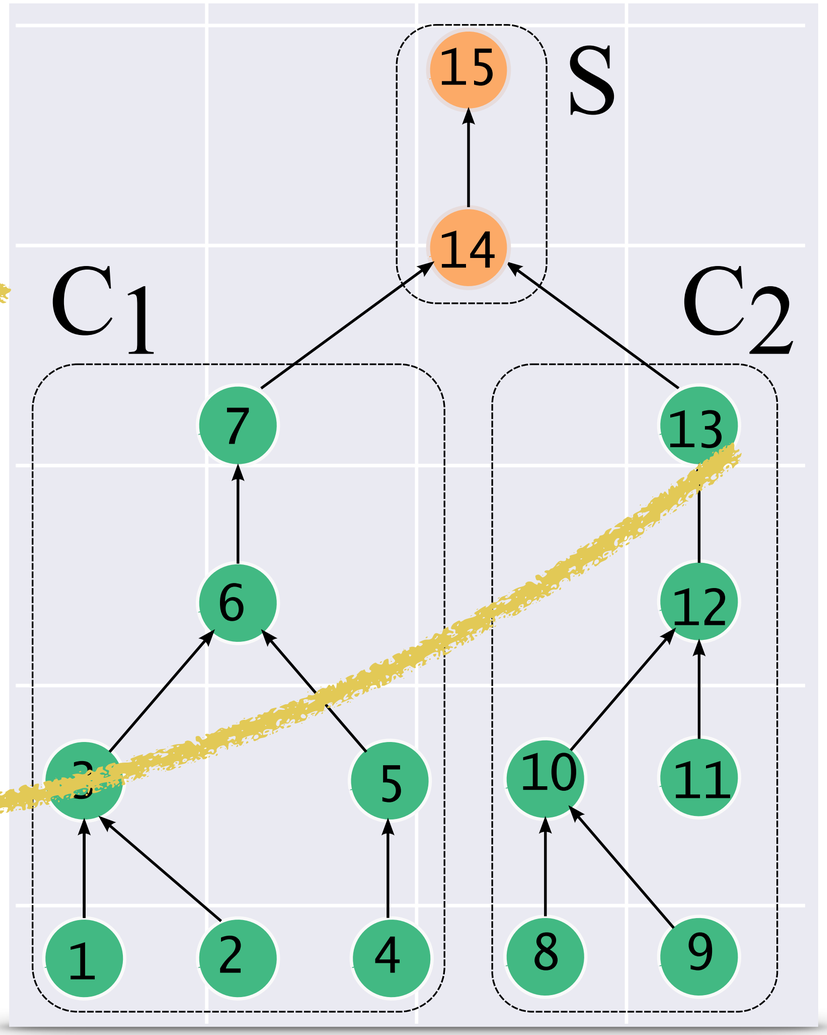
There are other memory-for-communication techniques,‡ including **multifrontal methods**. We claim better memory and process scalability. See our papers!

† Ashcraft (1991); Irony & Toledo (2002); Solomonik & Demmel (2011)

‡ Hulbert & Zmijewski (1991); Gupta et al. (1997)







(For experts) How? **Partition elimination tree** among 2-D slides of a 3-D process grid

Piyush: Extend to sparse LU

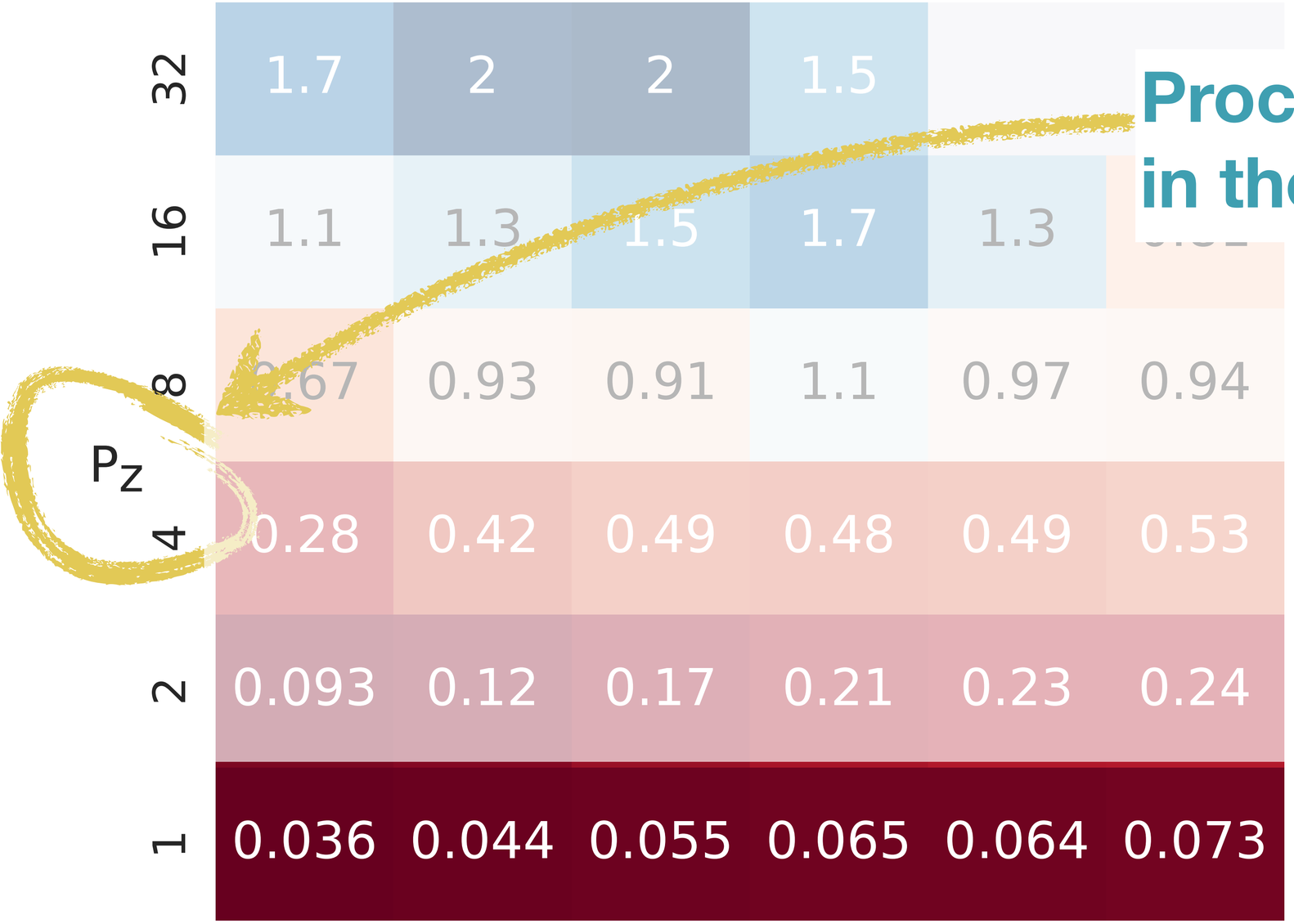
Example:

$$P_x \times P_y = 96$$

(Best configuration shown)

Teraflop/s

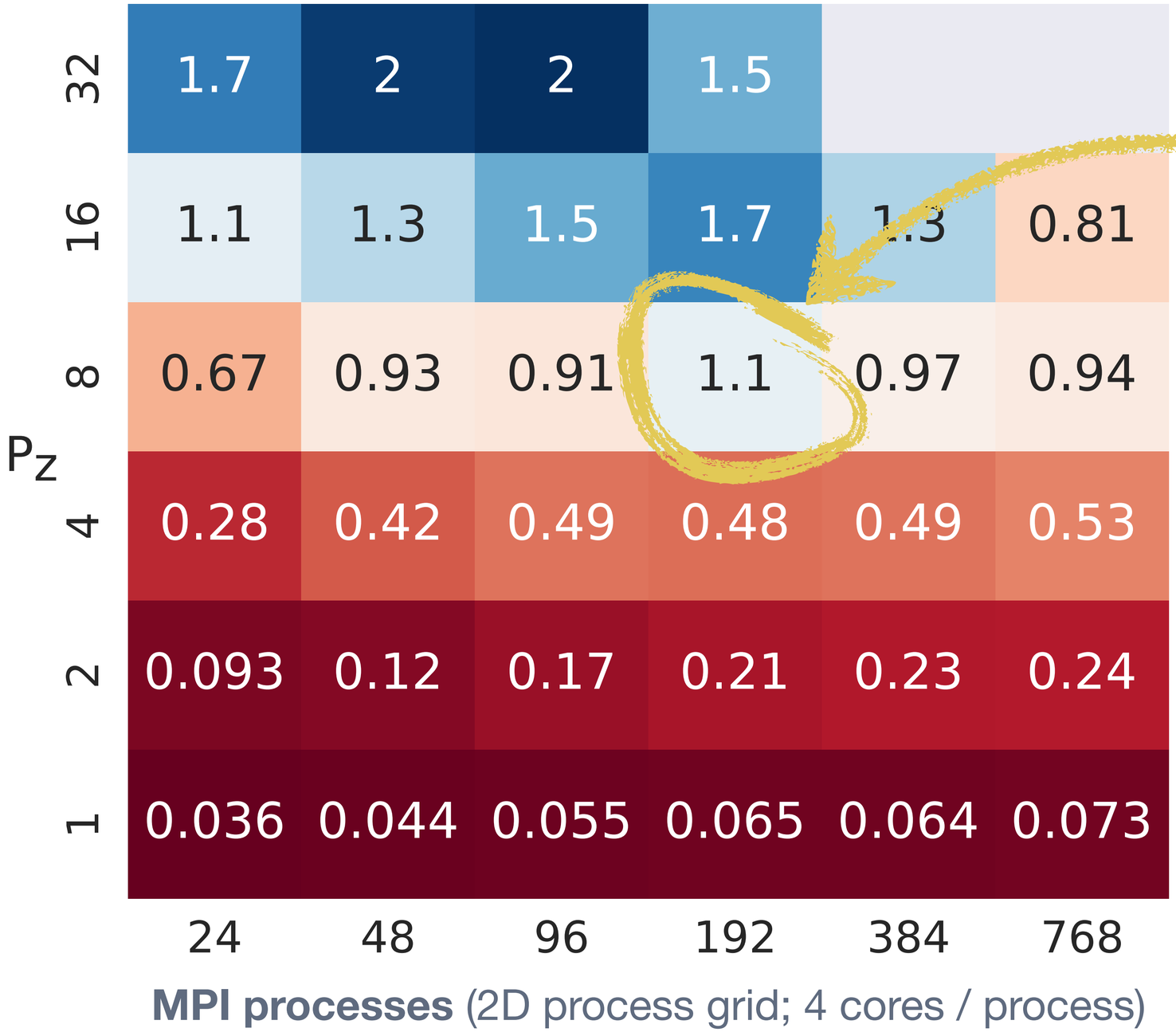
(32x procs → 2x speedup)



Process grid length in the 3rd (“z”) axis

Teraflop/s

(32x procs → 2x speedup)



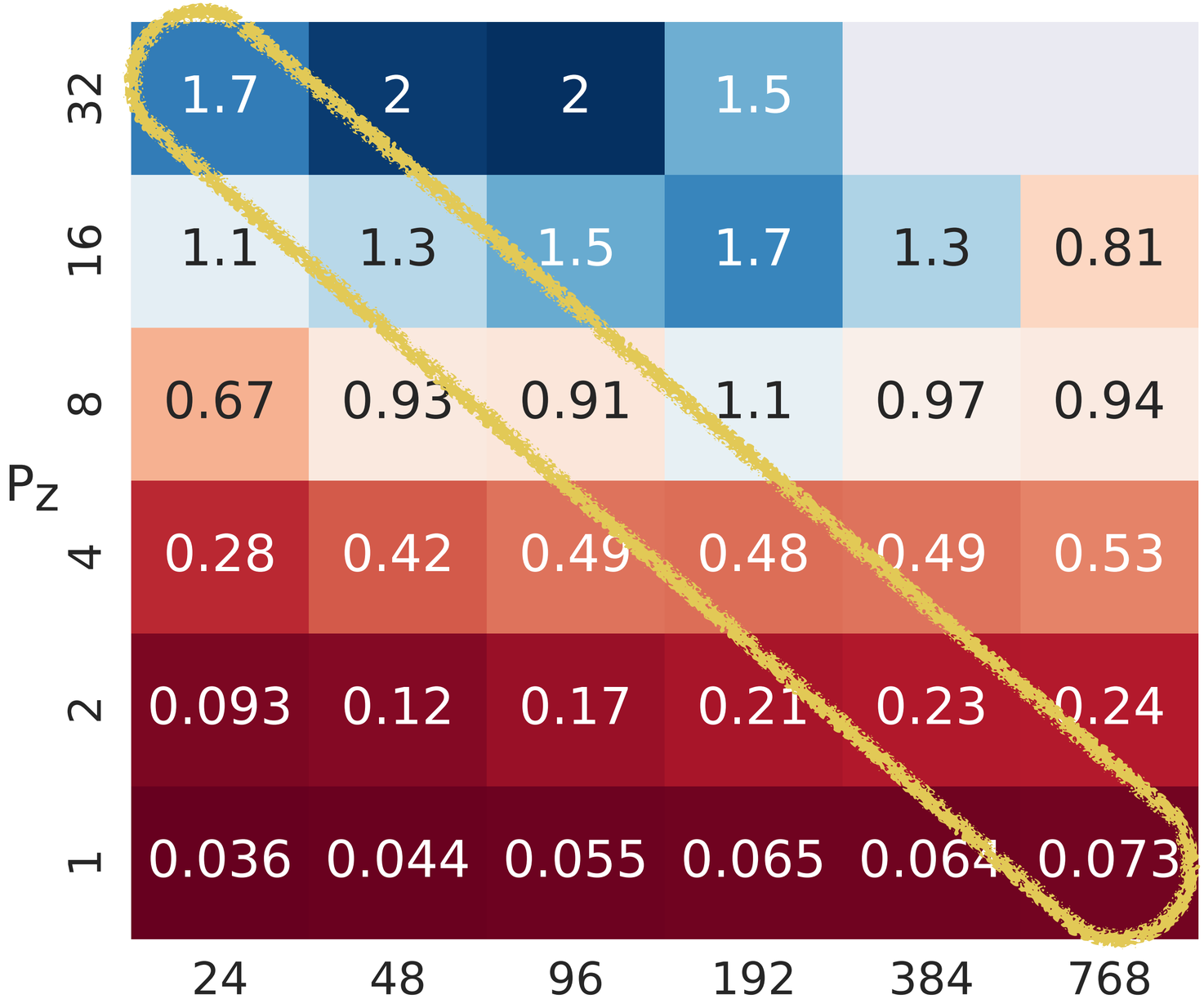
Example:

$$\underbrace{P_x \times P_y}_{=192} \times \underbrace{P_z}_{=8} = 1{,}536$$

(Best configuration shown for the 2D part)

Teraflop/s

(32x procs → 2x speedup)



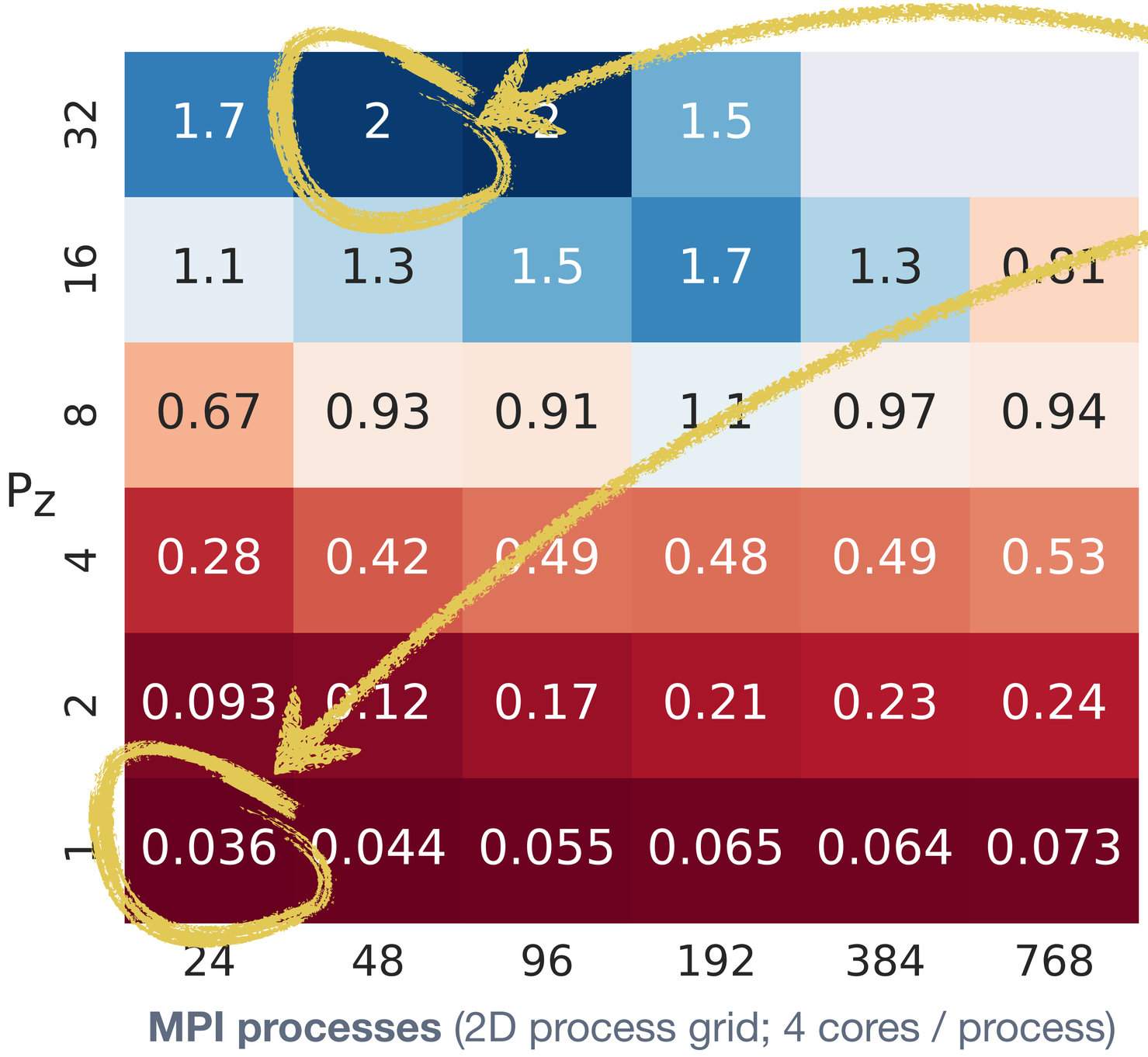
2D to 3D:

→ **23x speedup**

(24 x 32)

Teraflop/s

(32x procs → 2x speedup)



64x procs

→ 55x speedup

Teraflop/s

(32x procs → 2x speedup)

Summary

hpcgarage.org/futuresparse23

Assume that industry will not build you an efficient machine for your ***truly*** sparse computations.

Maybe we should band together around a common set of such computations that can drive hardware design projects. I have suggested sparse LU (sparse APSP) as one whose characteristics—semi-irregular parallelism, dynamic structure, variable intensity—makes it one “model problem” for co-design, but there can, and should, be many others, including yours!

FORZA

AGILE

ADVANCED GRAPHIC INTELLIGENCE LOGICAL COMPUTING ENVIRONMENT

†

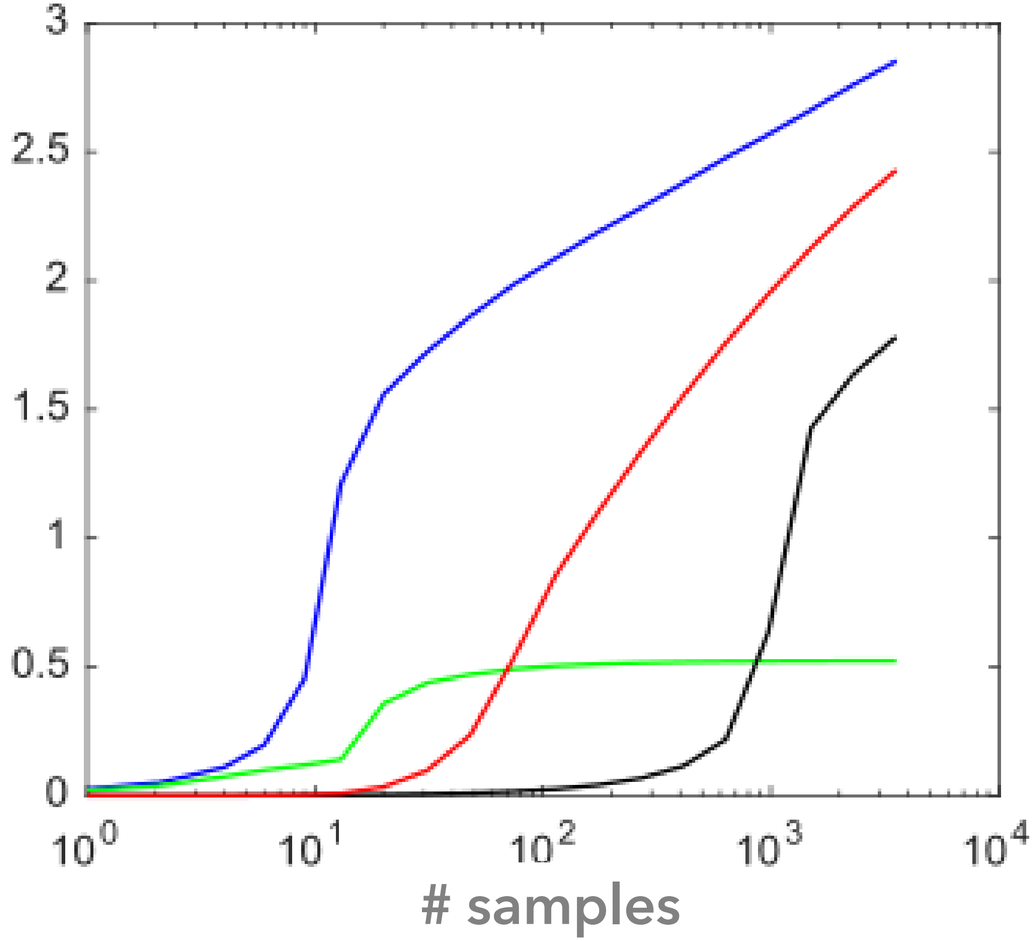
Bonus / Outtakes / BTS

Why deep learning may be intrinsically “dense”

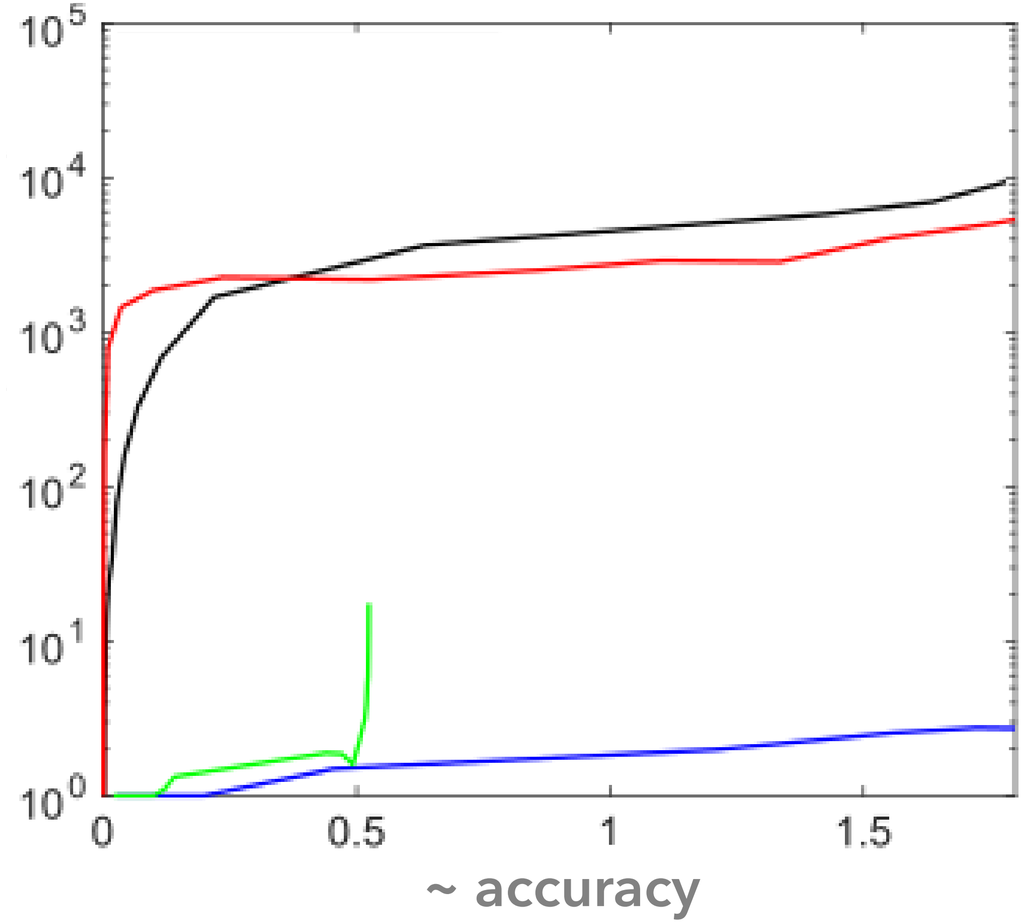
Multiple regression

$$X \equiv \begin{bmatrix} x_{i,0} & x_{i,1} & x_{i,2} & \cdots & x_{i,n-1} \end{bmatrix}$$

~ Accuracy



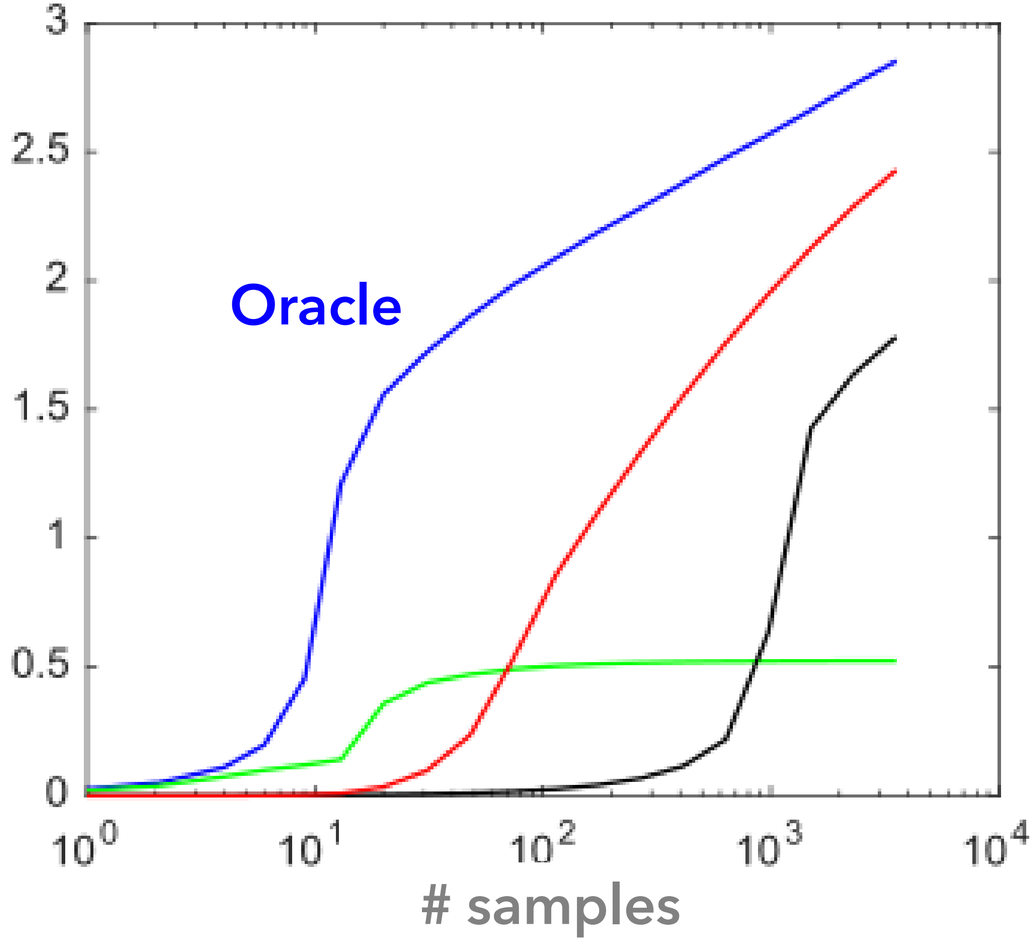
~ Ops (~ time)



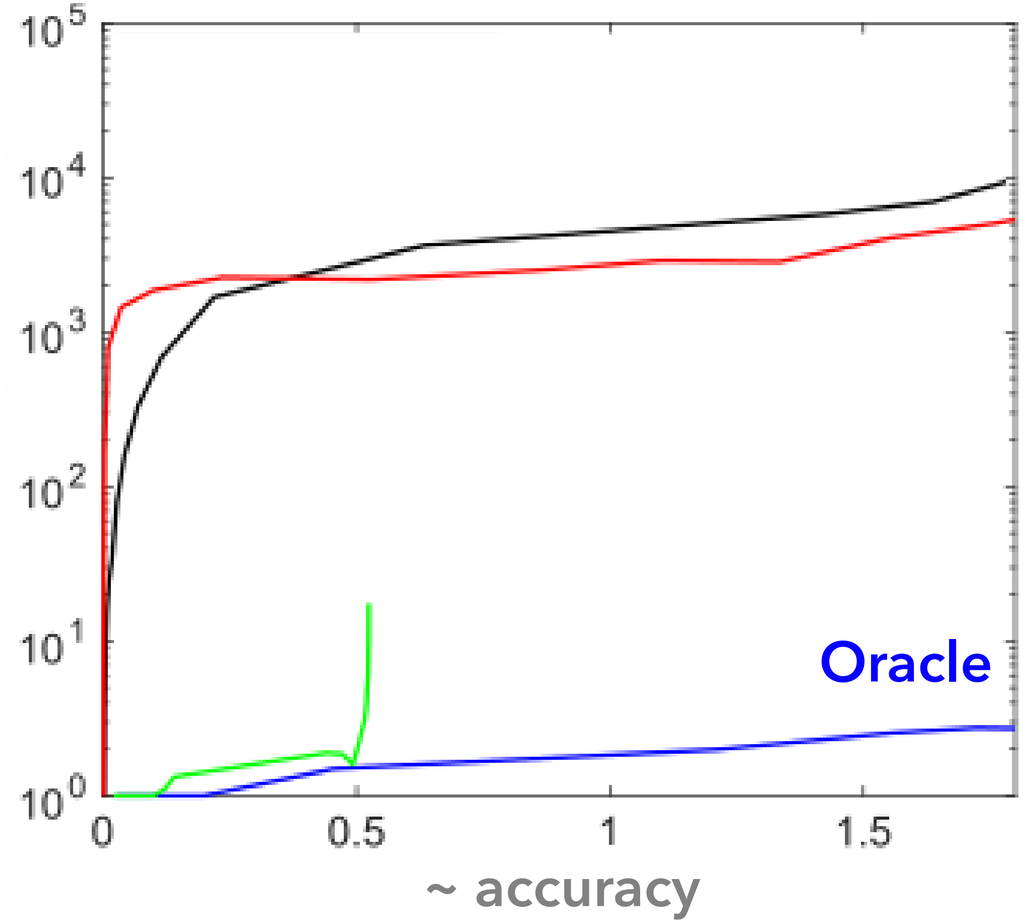
Linear regression example: Thompson et al., "The computational limits of deep learning" (July 2020). arXiv:2007.05558v1

More samples → More accuracy, reasonable time

~ Accuracy

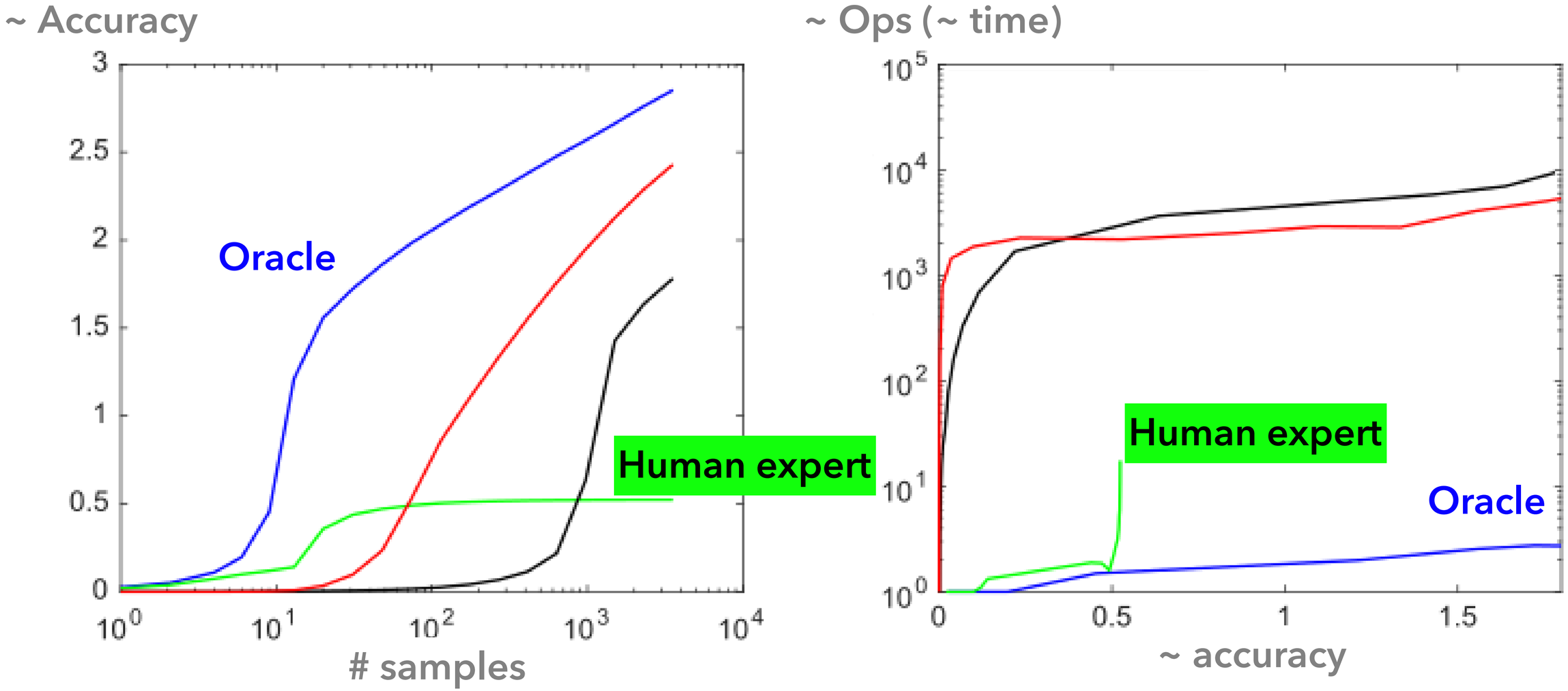


~ Ops (~ time)



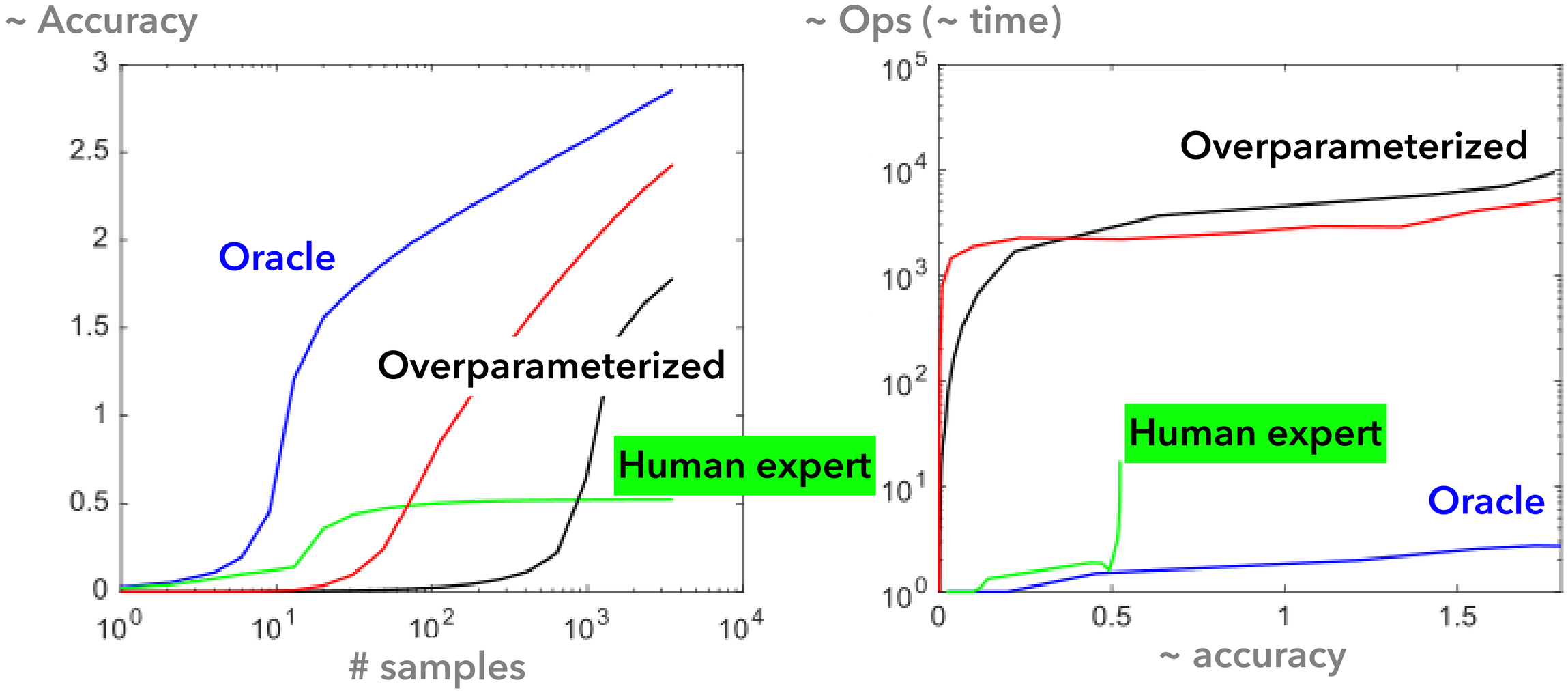
Linear regression example: Thompson et al., "The computational limits of deep learning" (July 2020). arXiv:2007.05558v1

Accuracy plateaus and costs rise



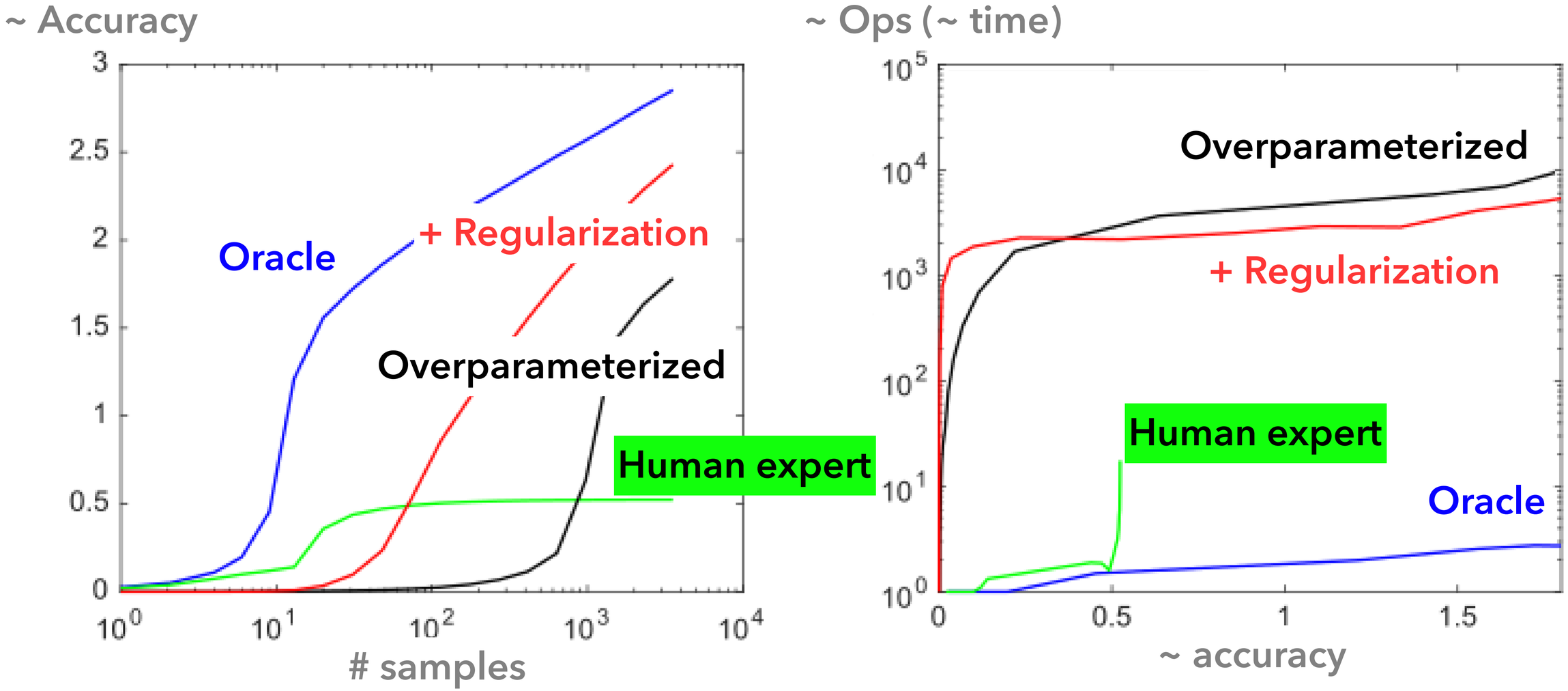
Linear regression example: Thompson et al., "The computational limits of deep learning" (July 2020). arXiv:2007.05558v1

With enough data, more accuracy but a high cost

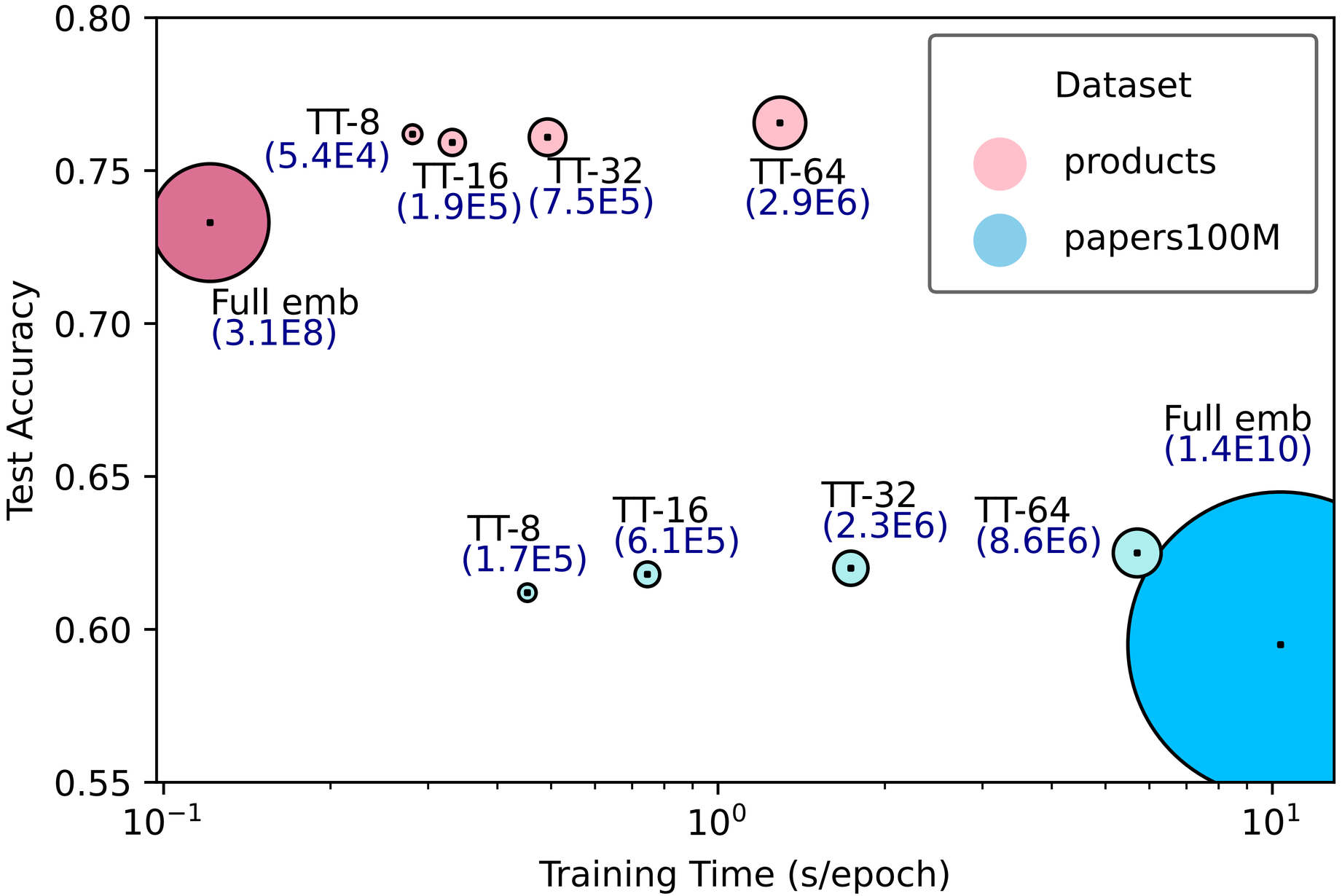


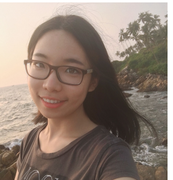
Linear regression example: Thompson et al., "The computational limits of deep learning" (July 2020). arXiv:2007.05558v1

Better accuracy with fewer samples, but still expensive



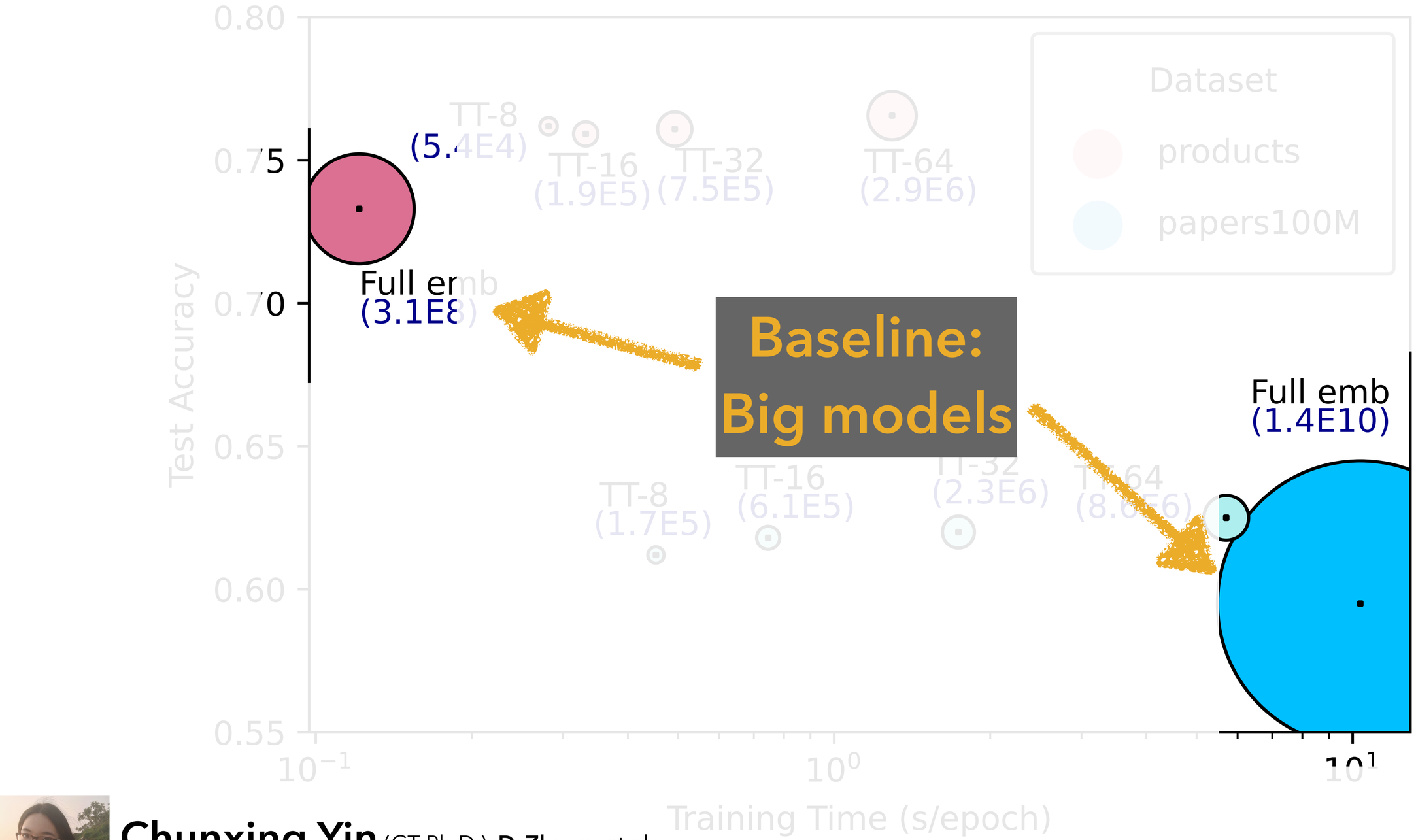
Linear regression example: Thompson et al., "The computational limits of deep learning" (July 2020). arXiv:2007.05558v1

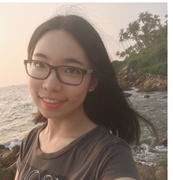




Chunxing Yin (GT Ph.D.), **D. Zheng** (Amazon), I. Nisrat, C. Faloutsos, G. Karypis, R. Vuduc.

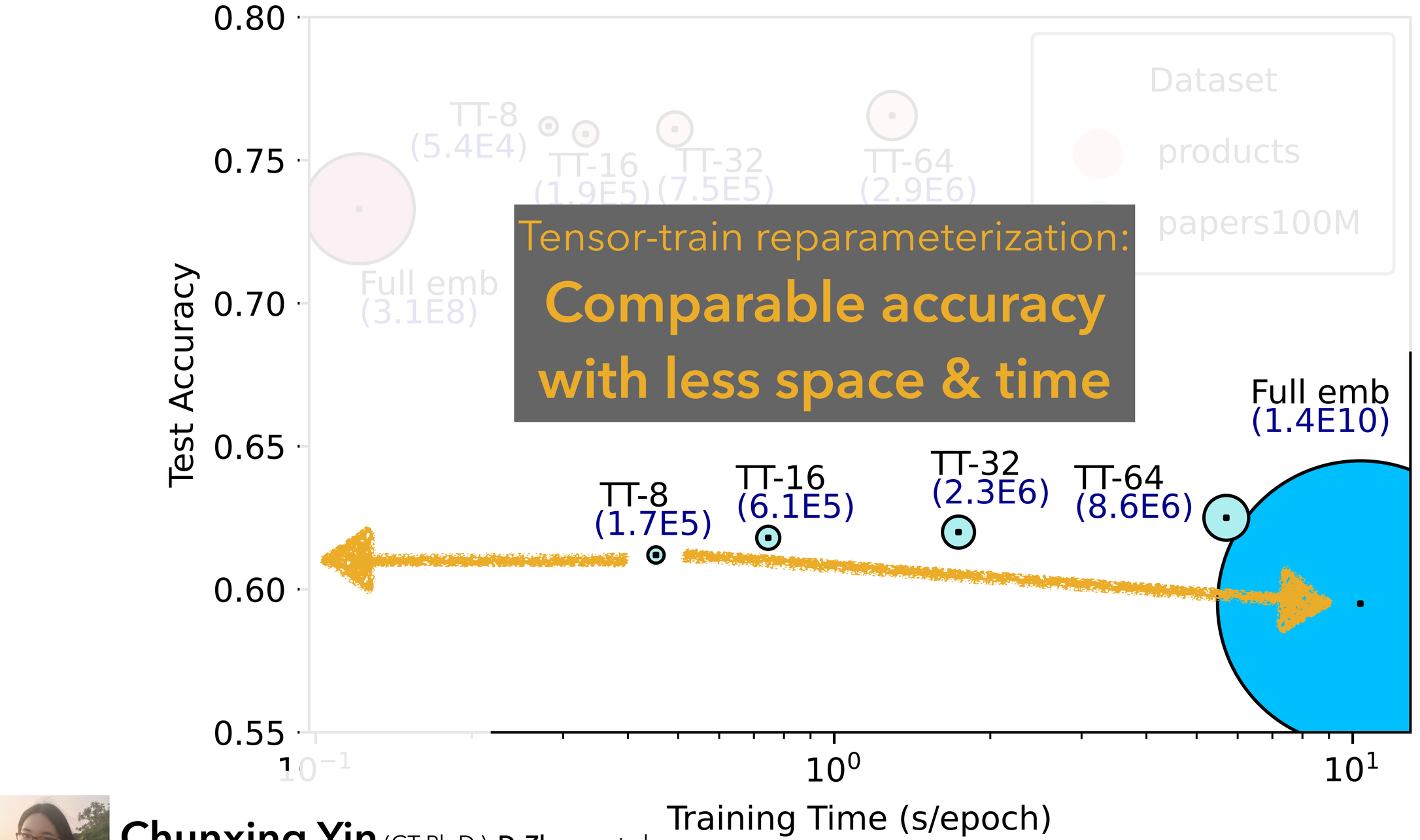
"Nimble GNN embedding with **tensor-train decomposition**." In *KDD'22*. doi:10.1145/3534678.3539423





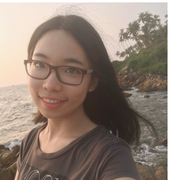
Chunxing Yin (GT Ph.D.), **D. Zheng,** et al.

"Nimble GNN embedding with **tensor-train decomposition**." In *KDD'22*. doi:10.1145/3534678.3539423



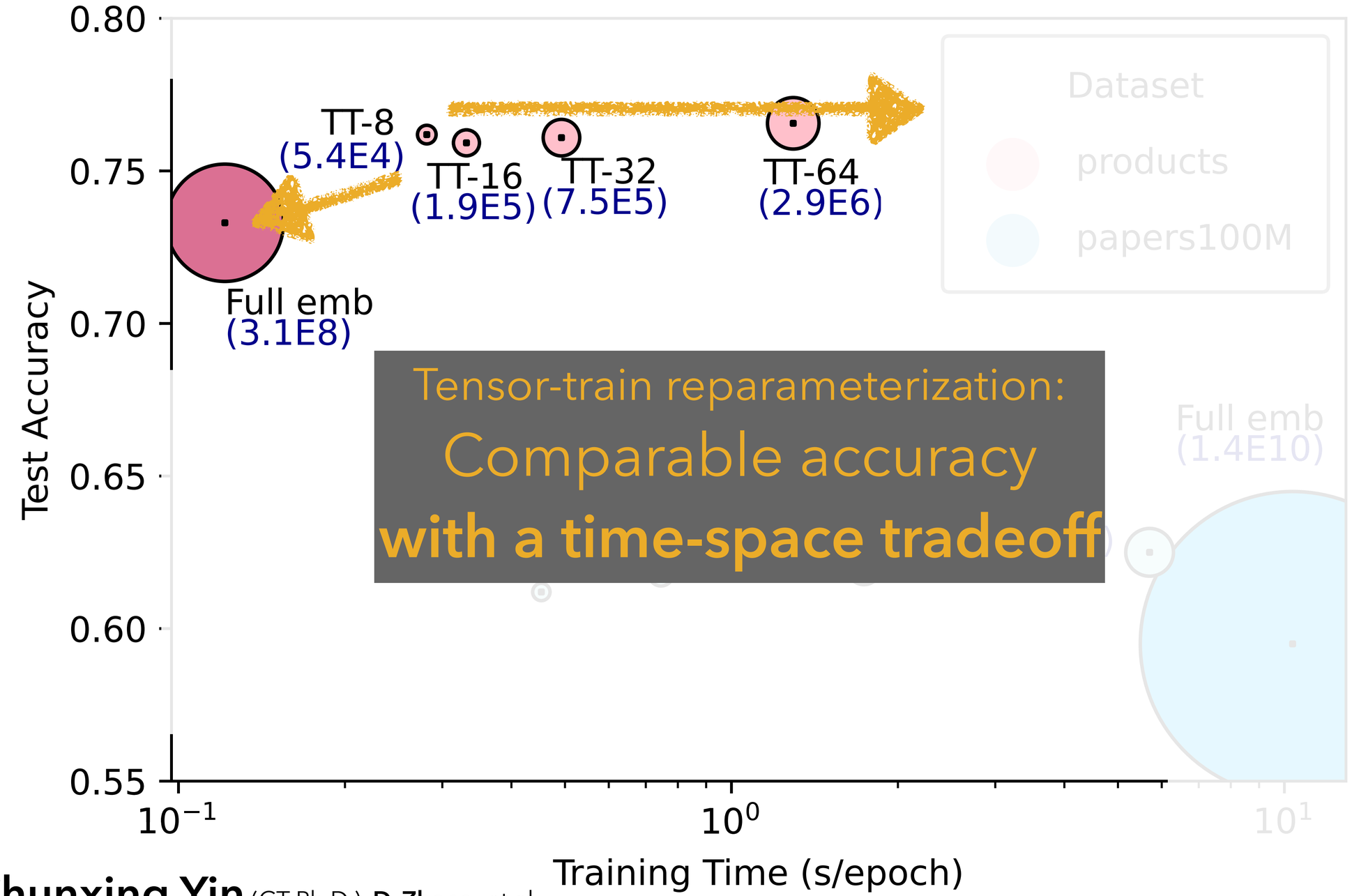
Chunxing Yin (GT Ph.D.), **D. Zheng,** et al.

"Nimble GNN embedding with **tensor-train decomposition**." In *KDD'22*. doi:10.1145/3534678.3539423



Chunxing Yin (GT Ph.D.), **D. Zheng,** et al.

"Nimble GNN embedding with **tensor-train decomposition**." In *KDD'22*. doi:10.1145/3534678.3539423



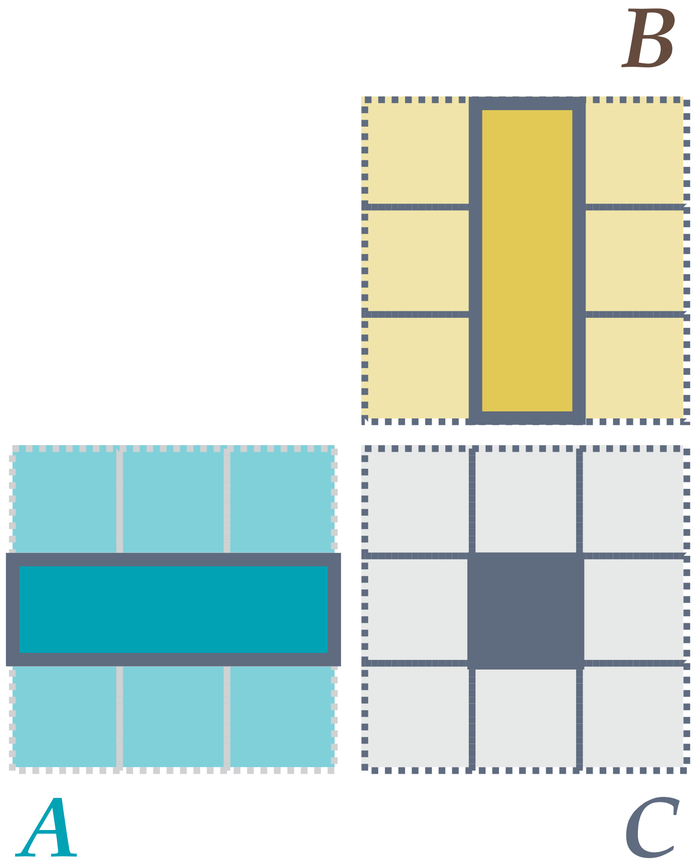
Communication avoidance 101

Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors

Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors



Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors

$\Longrightarrow$

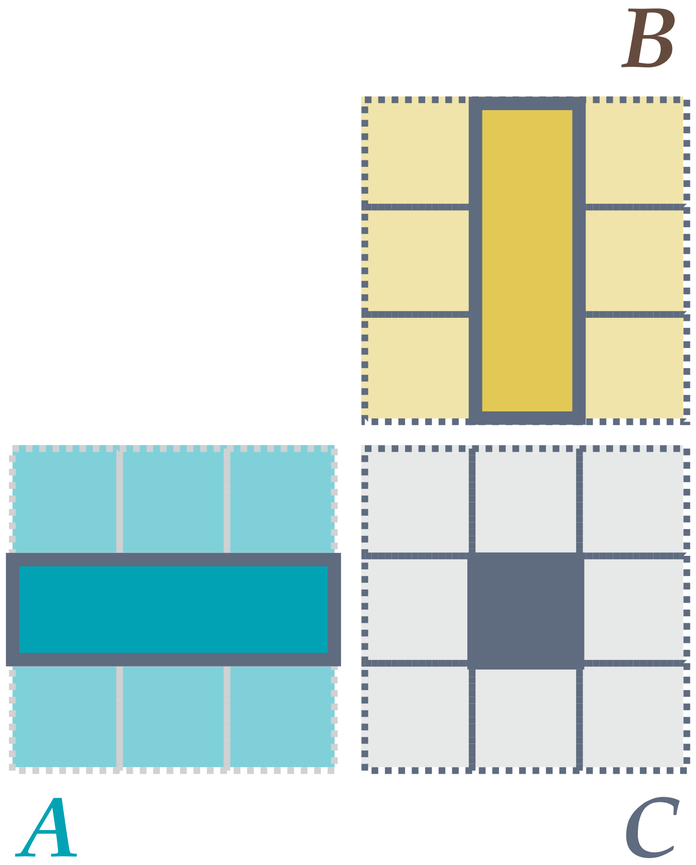
$N \times N$ matrices

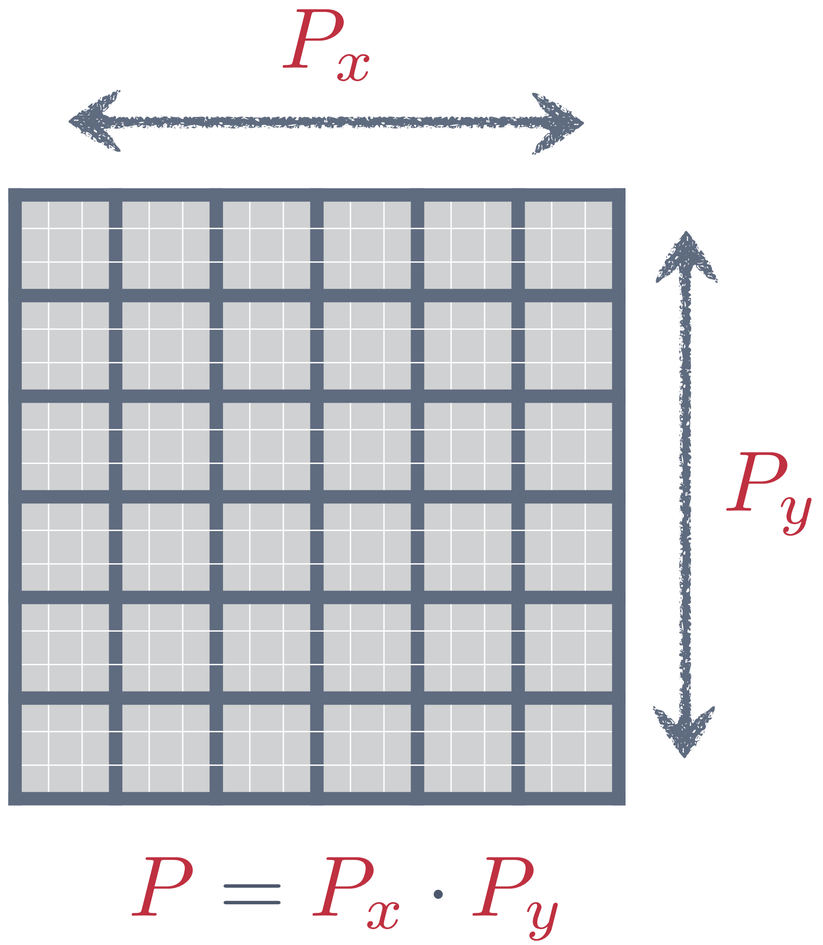
P processes

Time for flops $\propto \frac{N^3}{P}$

Communication-avoiding idea

For matrix multiplication, $C \mathrel{+}= A \cdot B$, on P processors

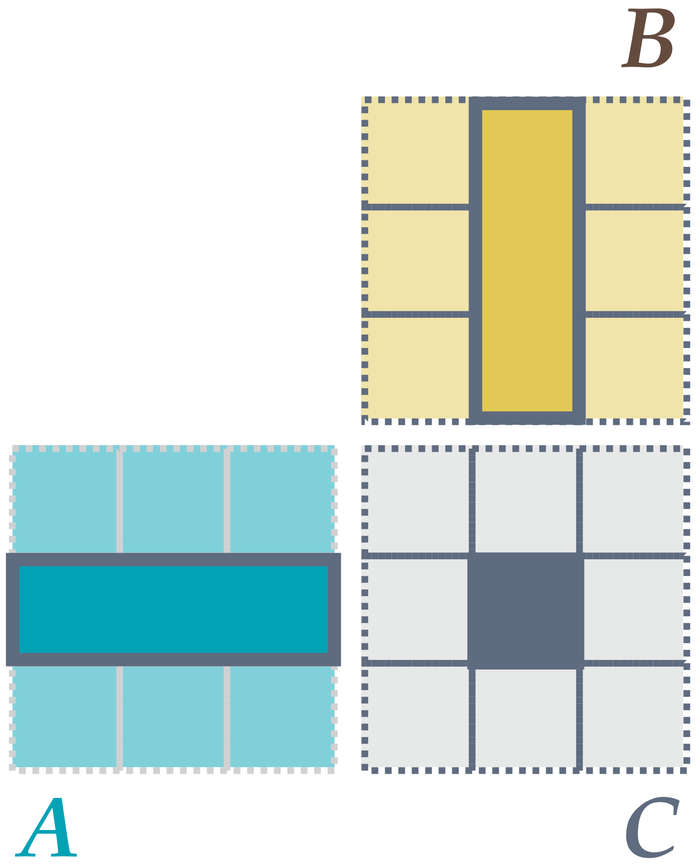




2D process grid

Communication-avoiding idea

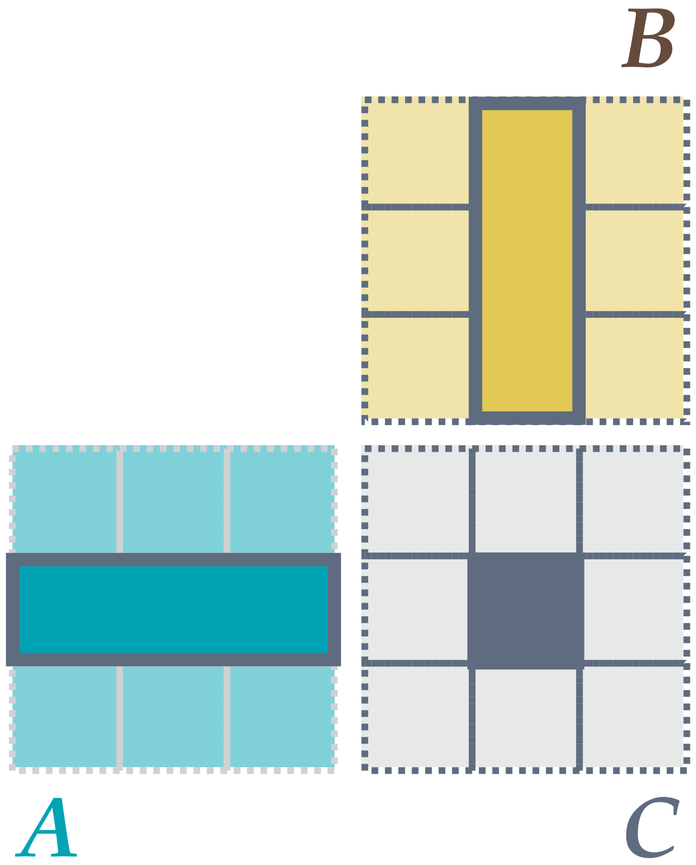
For matrix multiplication, C += A·B, on P processors



$\sqrt{P} \times \sqrt{P}$ process grid

Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors



$\sqrt{P} \times \sqrt{P}$ process grid

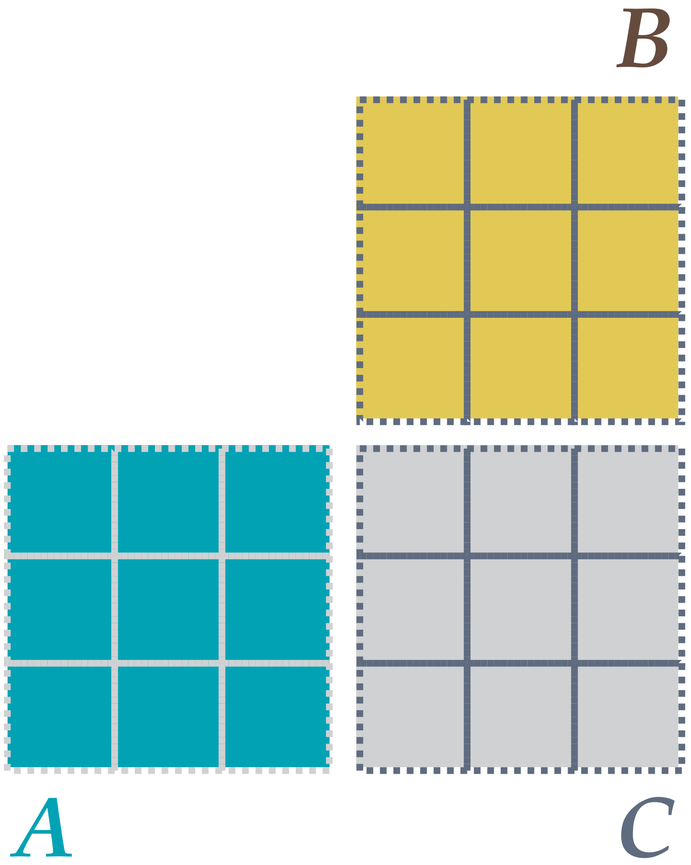
$\Theta\left(\frac{N^2}{P}\right)$ elems per proc

$\Longrightarrow \quad \Omega\left(\sqrt{P} + \frac{N^2}{\sqrt{P}}\right)$ comm time

Attained by **Cannon's algorithm** (1969), for instance

Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors



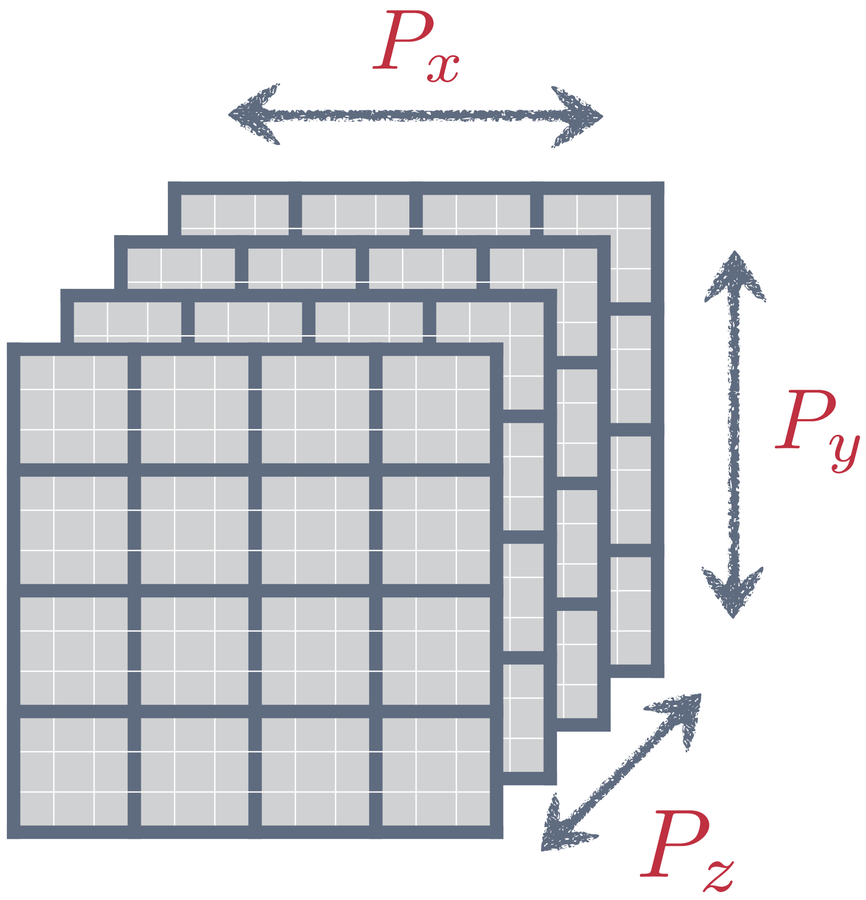
$\sqrt{P} \times \sqrt{P}$ process grid

$\Theta\left(\frac{N^2}{P}\right)$ elems per proc

$\Longrightarrow \quad \Omega\left(\sqrt{P} + \frac{N^2}{\sqrt{P}}\right)$ comm time

Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors

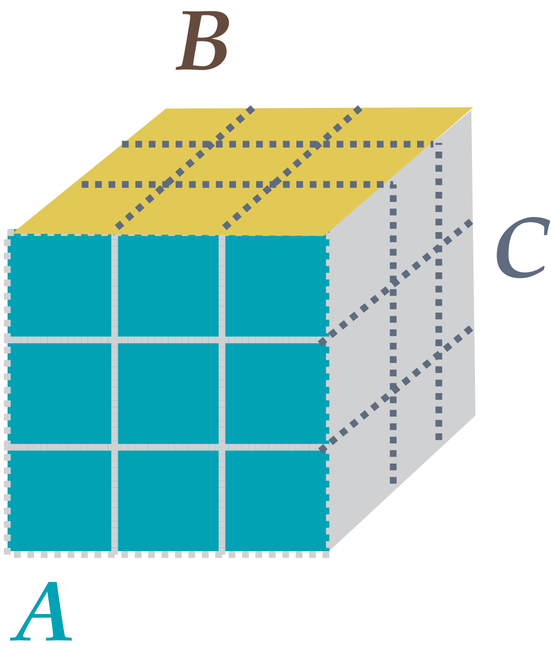


3D process grid

$$P = P_x \cdot P_y \cdot P_z$$

Communication-avoiding idea

For matrix multiplication, C += $A \cdot B$, on P processors



$P^{\frac{1}{3}} \times P^{\frac{1}{3}} \times P^{\frac{1}{3}}$ process grid

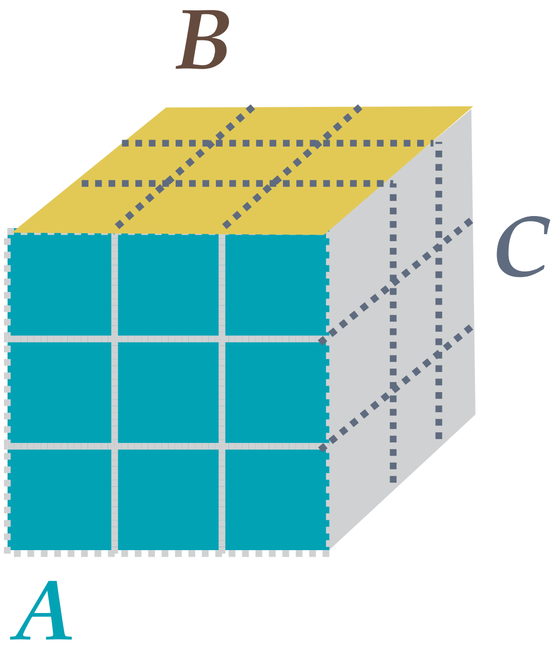
$\Theta\left(\frac{N^2}{P^{\frac{2}{3}}}\right)$ elems per proc (replicate)

Idea: Use a 3-D process grid and replicate

Dekel et al. (1981); Agarwal et al. (1995); + more

Communication-avoiding idea

For matrix multiplication, C += A·B, on P processors



$$\Longrightarrow$$

$P^{\frac{1}{3}} \times P^{\frac{1}{3}} \times P^{\frac{1}{3}}$ process grid

$\Theta\left(\frac{N^2}{P^{\frac{2}{3}}}\right)$ elems per proc (replicate)

$$\text{Memory}^{(3D)} = \text{Memory}^{(2D)} \times P^{\frac{1}{3}}$$

$$\text{CommTime}^{(3D)} \approx \frac{\text{CommTime}^{(2D)}}{P^{\frac{1}{3}}}$$

Trades more memory for less communication

Piyush: CA for sparse LU

All known “3D-LU” algorithms† are for **dense LU**. They reduce communication volume but increase latency.

For sparse LU, we can **reduce** both the latency and bandwidth for “planar” problems **asymptotically**, and achieve constant-factor reductions for “non-planar” ones.

There are other memory-for-communication techniques,‡ including **multifrontal methods**. We claim better memory and process scalability. See our papers!

† Ashcraft (1991); Irony & Toledo (2002); Solomonik & Demmel (2011)
‡ Hulbert & Zmijewski (1991); Gupta et al. (1997)

An “iron law”

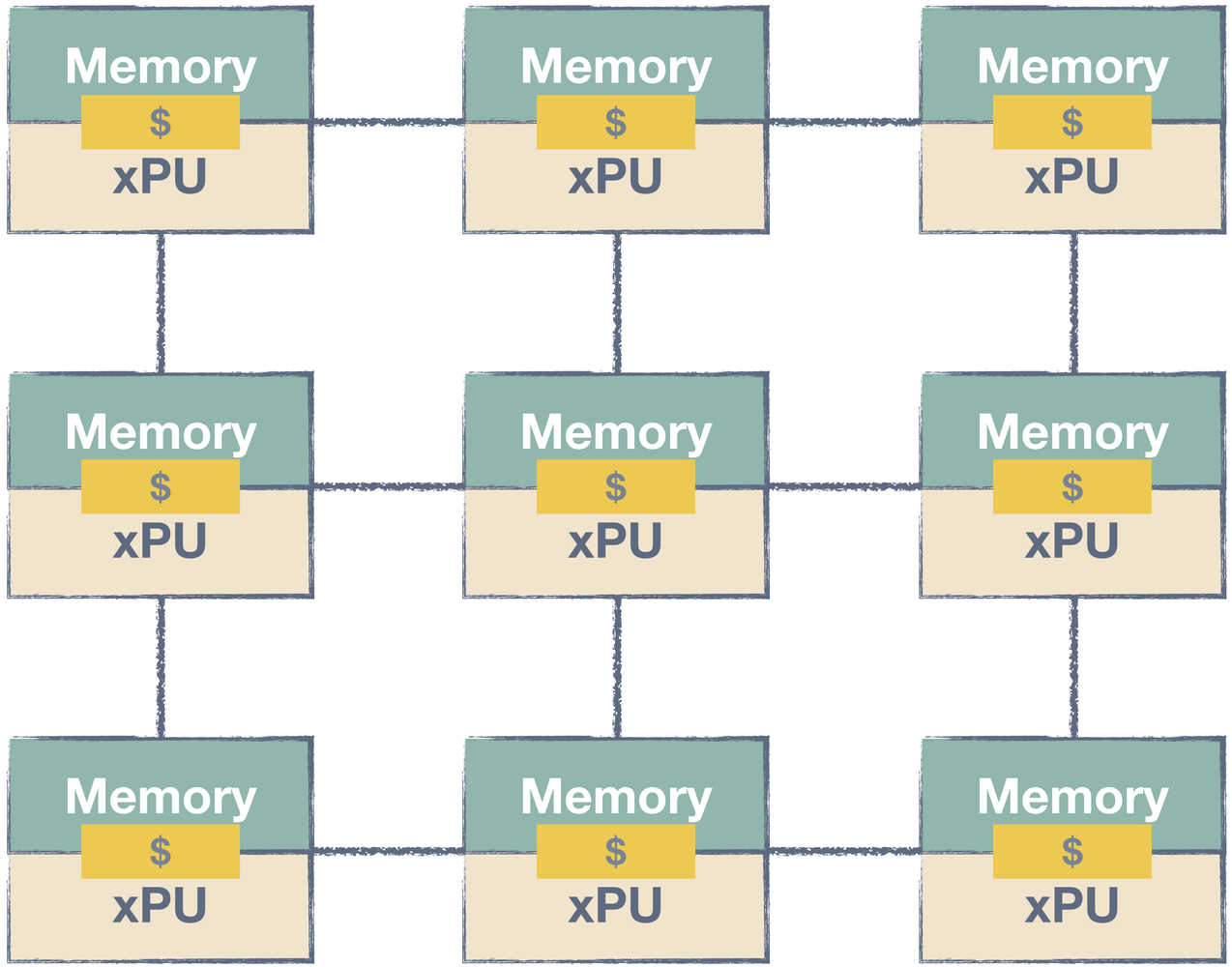
An Iron Law of Parallel and Distributed Computation

A modern cluster or supercomputer is, to first order, a collection of processing nodes. Each node has a processor (“xPU”) and a two-level memory hierarchy. Nodes are connected by a network.

As a program executes on this system, it incurs two types of communication cost.

“Vertical” communication occurs in the memory system between, say, RAM and cache.

“Horizontal” communication occurs between nodes across the network.



Two costs: $T_{\text{network}} + T_{\text{memory}}$

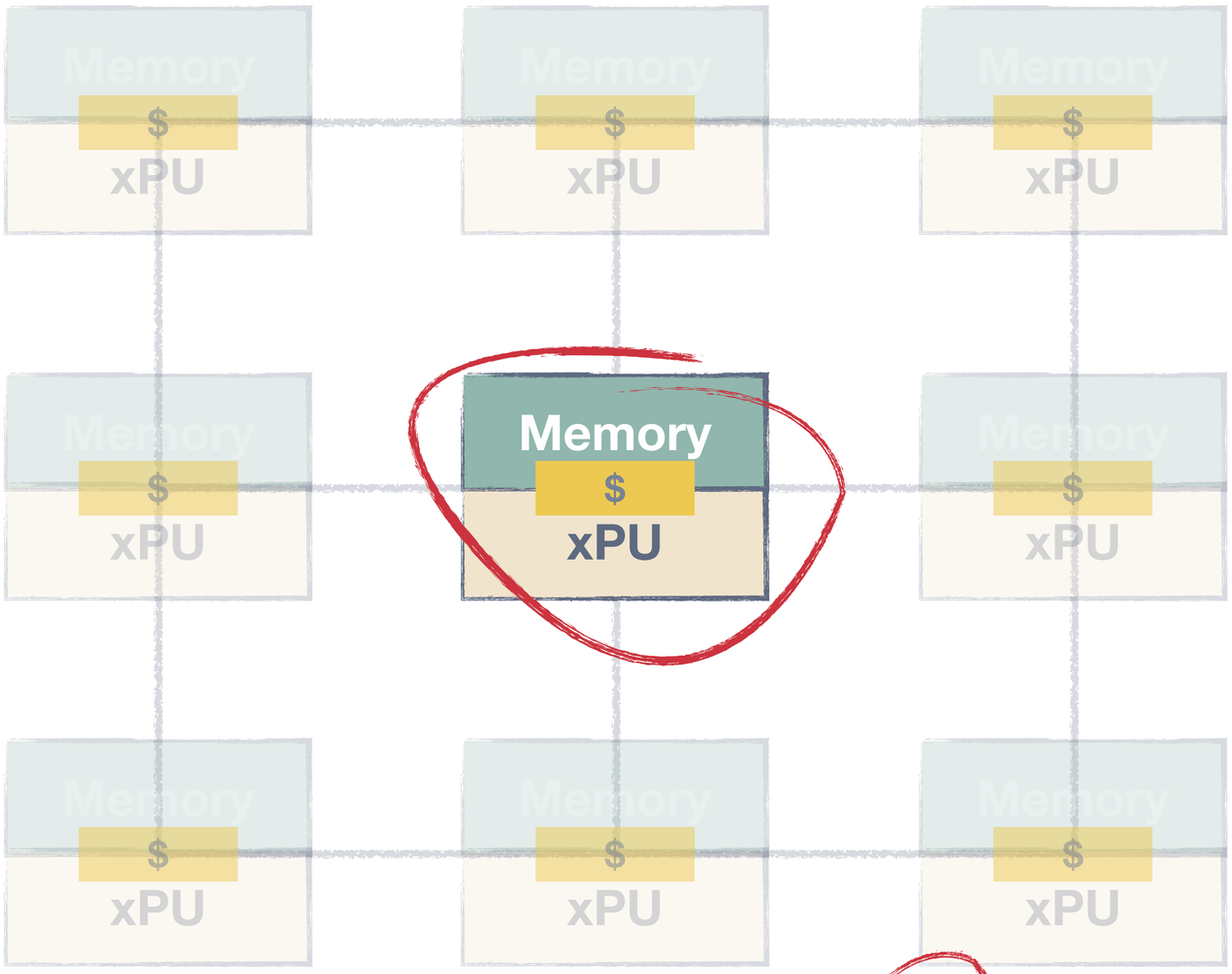
An Iron Law of Parallel and Distributed Computation

A modern cluster or supercomputer is, to first order, a collection of processing nodes. Each node has a processor (“xPU”) and a two-level memory hierarchy. Nodes are connected by a network.

As a program executes on this system, it incurs two types of communication cost.

“Vertical” communication occurs in the memory system between, say, RAM and cache.

“Horizontal” communication occurs between nodes across the network.



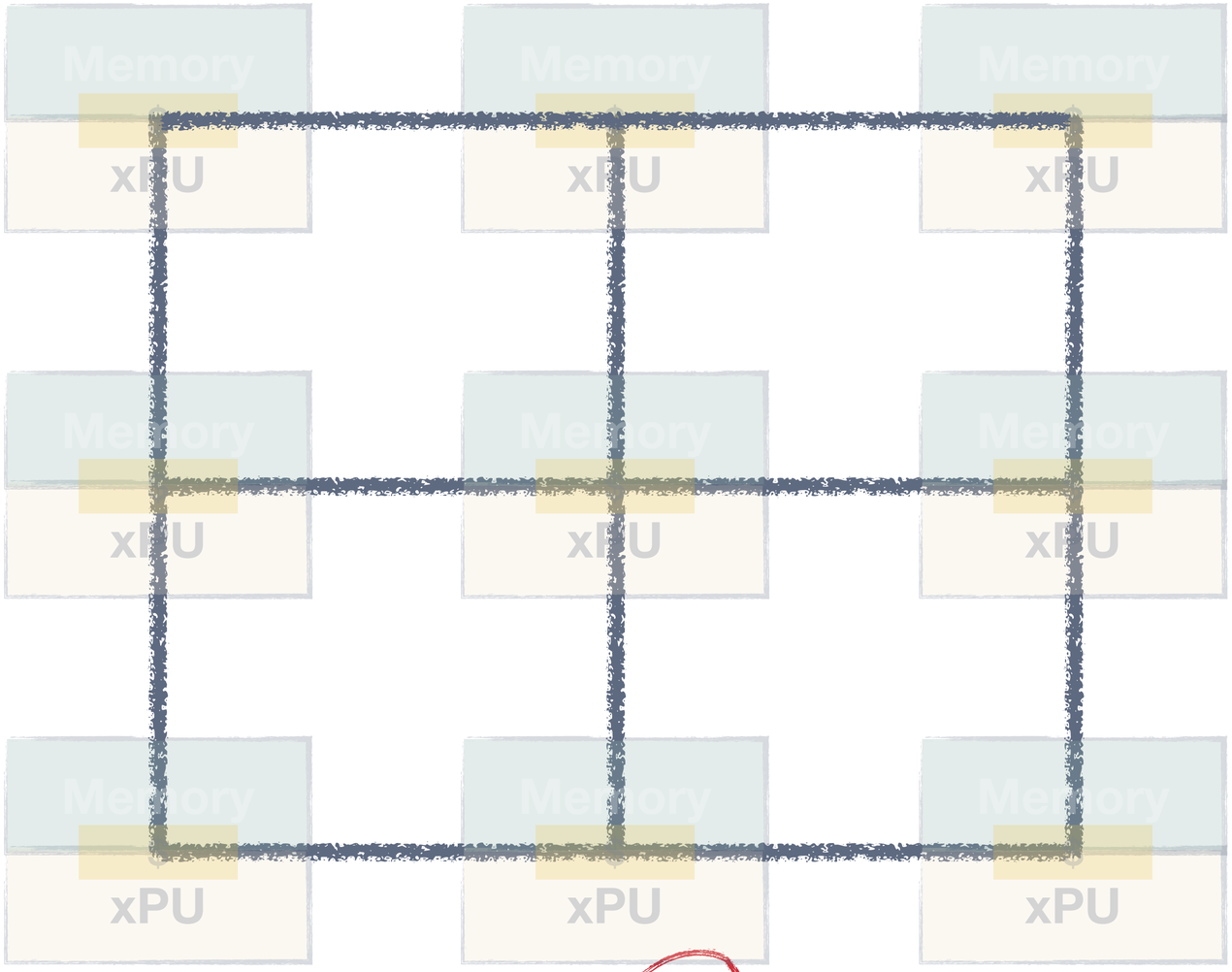
An Iron Law of Parallel and Distributed Computation

A modern cluster or supercomputer is, to first order, a collection of processing nodes. Each node has a processor (“xPU”) and a two-level memory hierarchy. Nodes are connected by a network.

As a program executes on this system, it incurs two types of communication cost.

“Vertical” communication occurs in the memory system between, say, RAM and cache.

“Horizontal” communication occurs between nodes across the network.



(Asymptotic running time – rules-of-thumb)

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

P-fold speedup, ideally

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

P-fold speedup, ideally

Memory time

$$\frac{W(n)}{P \cdot f(Z)}$$

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

P-fold speedup, ideally

Memory time

$$\frac{W(n)}{P \cdot f(Z)}$$

e.g., $\sqrt{Z}$, $\log Z$

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

P-fold speedup, ideally

Memory time

$$\frac{W(n)}{P \cdot f(Z)}$$

e.g., $\sqrt{Z}$, $\log Z$

Network time

$$\frac{W(n)}{h(n)} \cdot \frac{g(P)}{P}$$

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

P-fold speedup, ideally

Memory time

$$\frac{W(n)}{P \cdot f(Z)}$$

e.g., $\sqrt{Z}$, $\log Z$

Network time

$$\frac{W(n)}{h(n)} \cdot \frac{g(P)}{P}$$

Asymptotic reduction

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

(Asymptotic running time – rules-of-thumb)

Compute time

$$\frac{W(n)}{P}$$

P-fold speedup, ideally

Memory time

$$\frac{W(n)}{P \cdot f(Z)}$$

e.g., $\sqrt{Z}$, $\log Z$

Network time

$$\frac{W(n)}{h(n)} \cdot \frac{g(P)}{P}$$

Asymptotic reduction

Tradeoff

Jeff Young (GT), R.V. (2016) – "Finding balance in post-Moore's Law era." [link]

Sara Karamati

DPUs in modern clusters

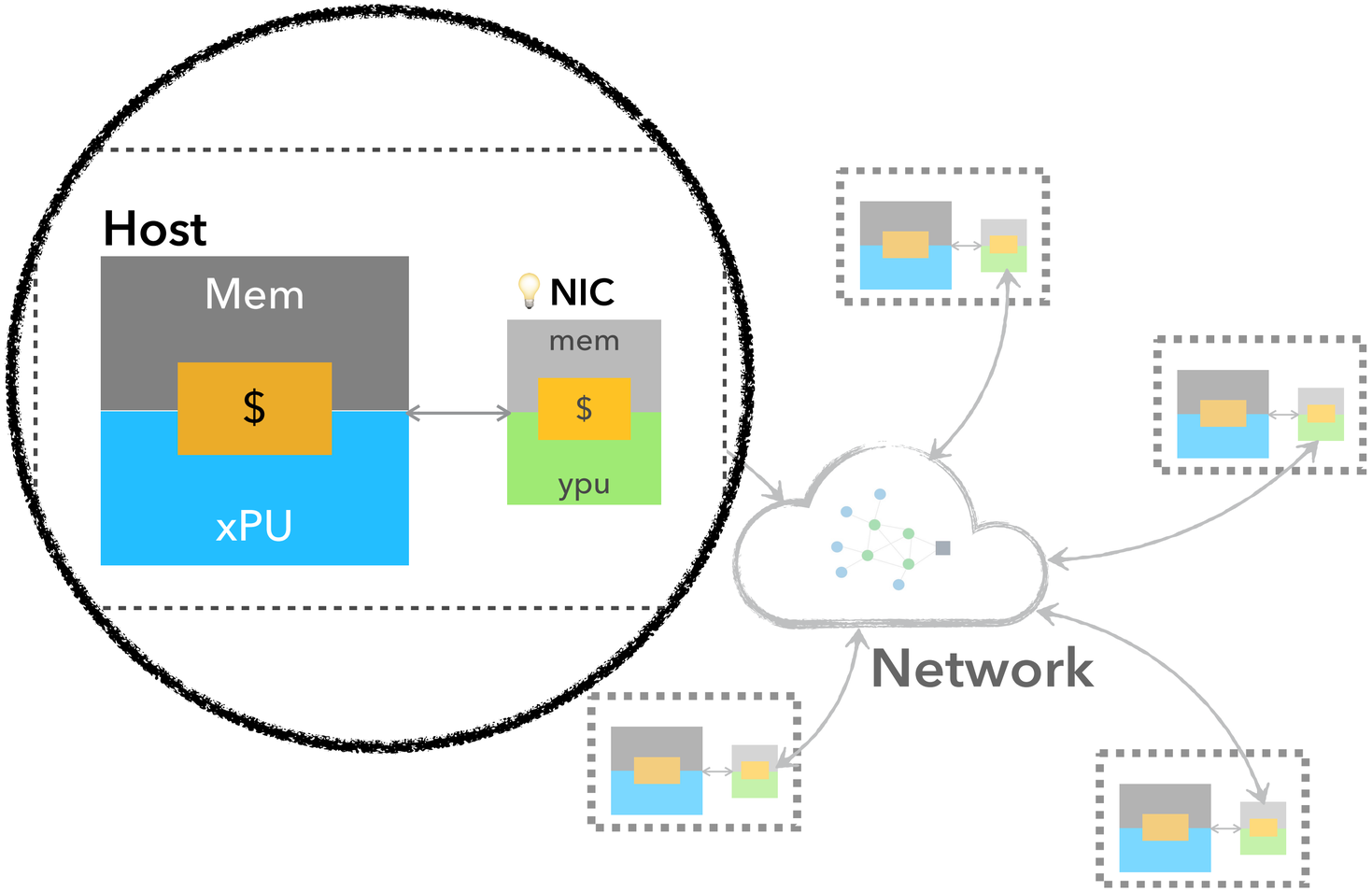
The basic building block of a distributed-memory cluster or supercomputer is a node.

Each node includes a host, which is a processor (xPU) + memory hierarchy.

The host can communicate with other hosts via its NIC (network interface controller).

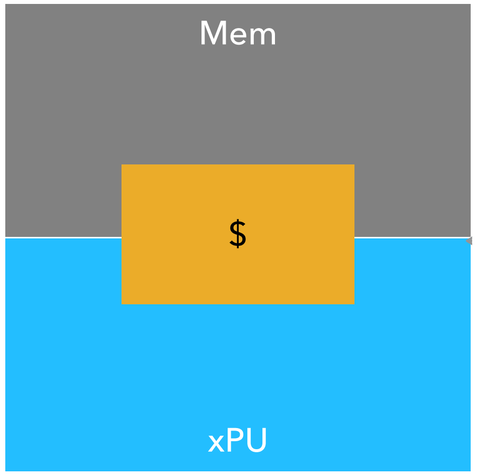
A network connects the nodes. The nodes may be arranged in some topology, which determines the network's carrying capacity and cost.

In a **smartNIC**, the NIC becomes "**host-like**" via the addition of processing (ypu) and memory.



Hypothetical: Multi-SmartNIC

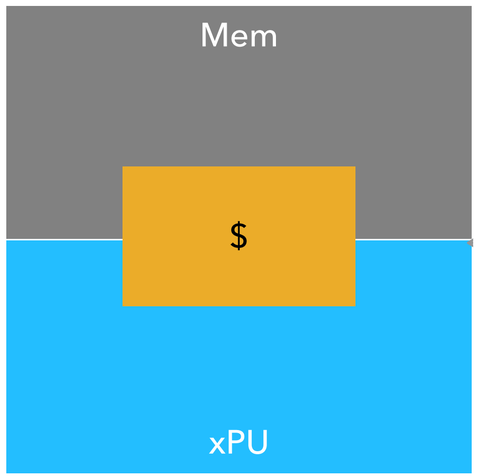
One host xPU (16 cores)



S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

Hypothetical: Multi-SmartNIC

One host xPU (16 cores)

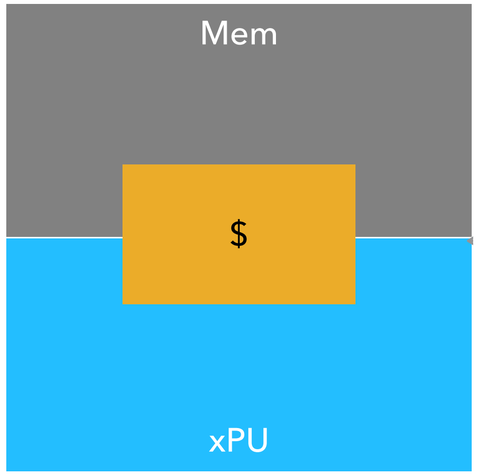


657 GF/s

S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

Hypothetical: Multi-SmartNIC

One host xPU (16 cores)



657 GF/s

76.8 GB/s

S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

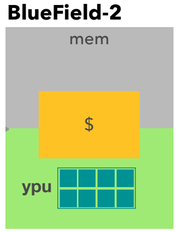
Hypothetical: Multi-SmartNIC

One host xPU (16 cores)

657 GF/s

76.8 GB/s

BF-2 yPUs (no host)



S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

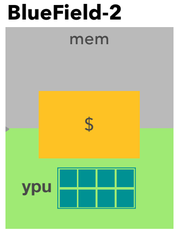
Hypothetical: Multi-SmartNIC

One host xPU (16 cores)

657 GF/s

76.8 GB/s

BF-2 yPUs (no host)



80 GF/s

25.6 GB/s

S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

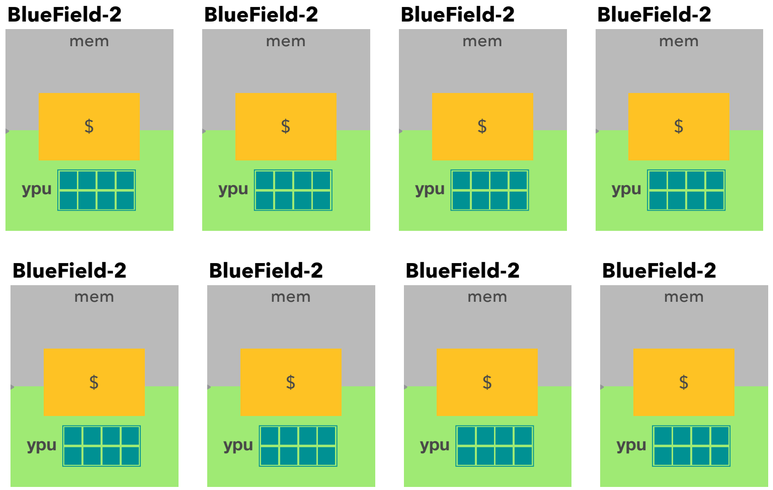
Hypothetical: Multi-SmartNIC

One host xPU (16 cores)

657 GF/s

76.8 GB/s

8 x BF-2 yPUs (no host)



640 GF/s

204 GB/s

(aggregate)

S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

Hypothetical: Multi-SmartNIC

One host xPU (16 cores)

~ 8.5 F:B

8 x BF-2 yPUs (no host)

~ 3.1 F:B

S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

Hypothetical: Multi-SmartNIC

One host xPU (16 cores)

Time = "1"

using all cores

8 x BF-2 yPUs (no host)

Speedup ~ 1.7x

Real measurement on MiniMD!

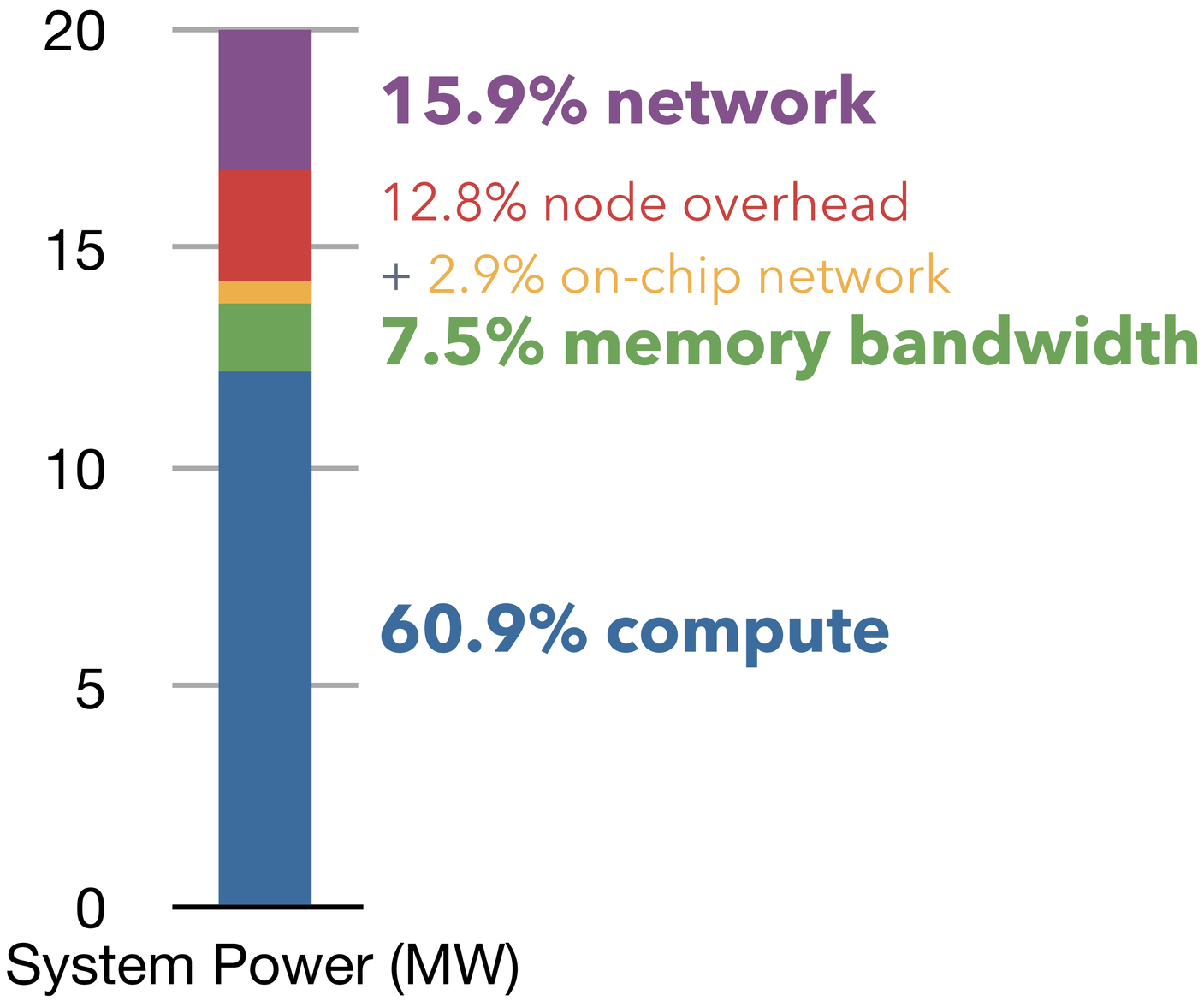
S. Karamati (GT), J. Young, R.V., et al. (2016) – "Smarter NICs for faster molecular dynamics: a case study." doi:10.1109/IPDPS53621.2022.00063

What else could one build?

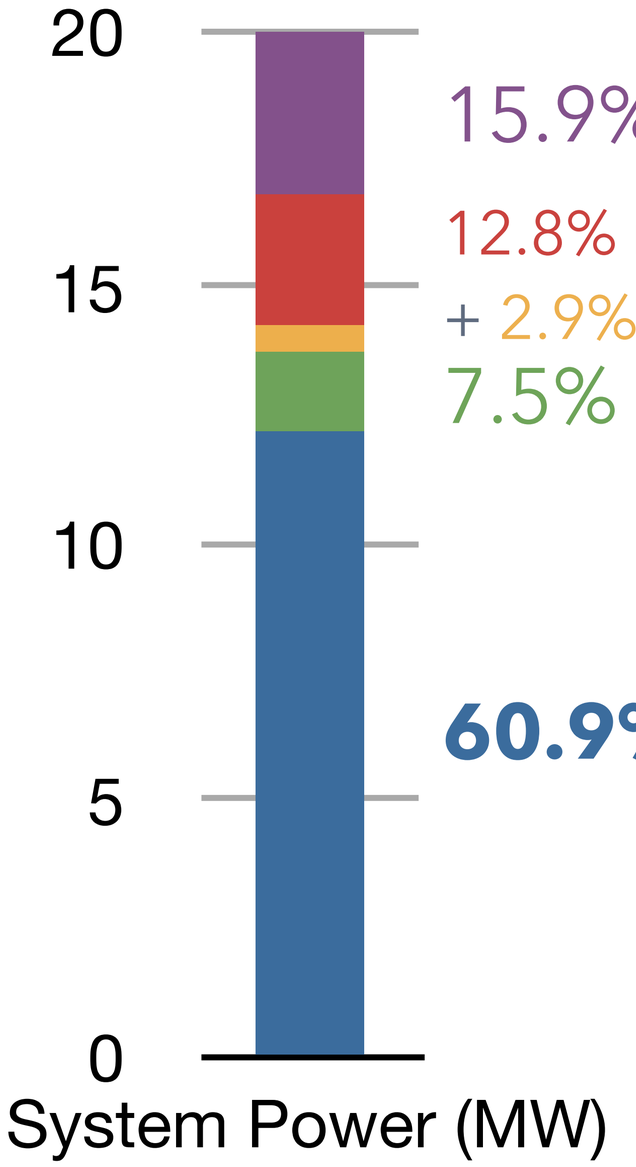
Power allocation for an “optimal” matrix multiply machine?

K. Czechowski, R. Vuduc. A theoretical framework for algorithm-architecture co-design. In *IPDPS'13*. doi:10.1109/IPDPS.2013.99

Power allocation for an “optimal” matrix multiply machine



Power allocation for an “optimal” matrix multiply machine



15.9% network

12.8% node overhead

+ 2.9% on-chip network

7.5% memory bandwidth

60.9% compute

ORNL Summit (13-14 MW):

67.0% GPU compute

14.9% CPU compute

4.8% memory

5.3% network + disk

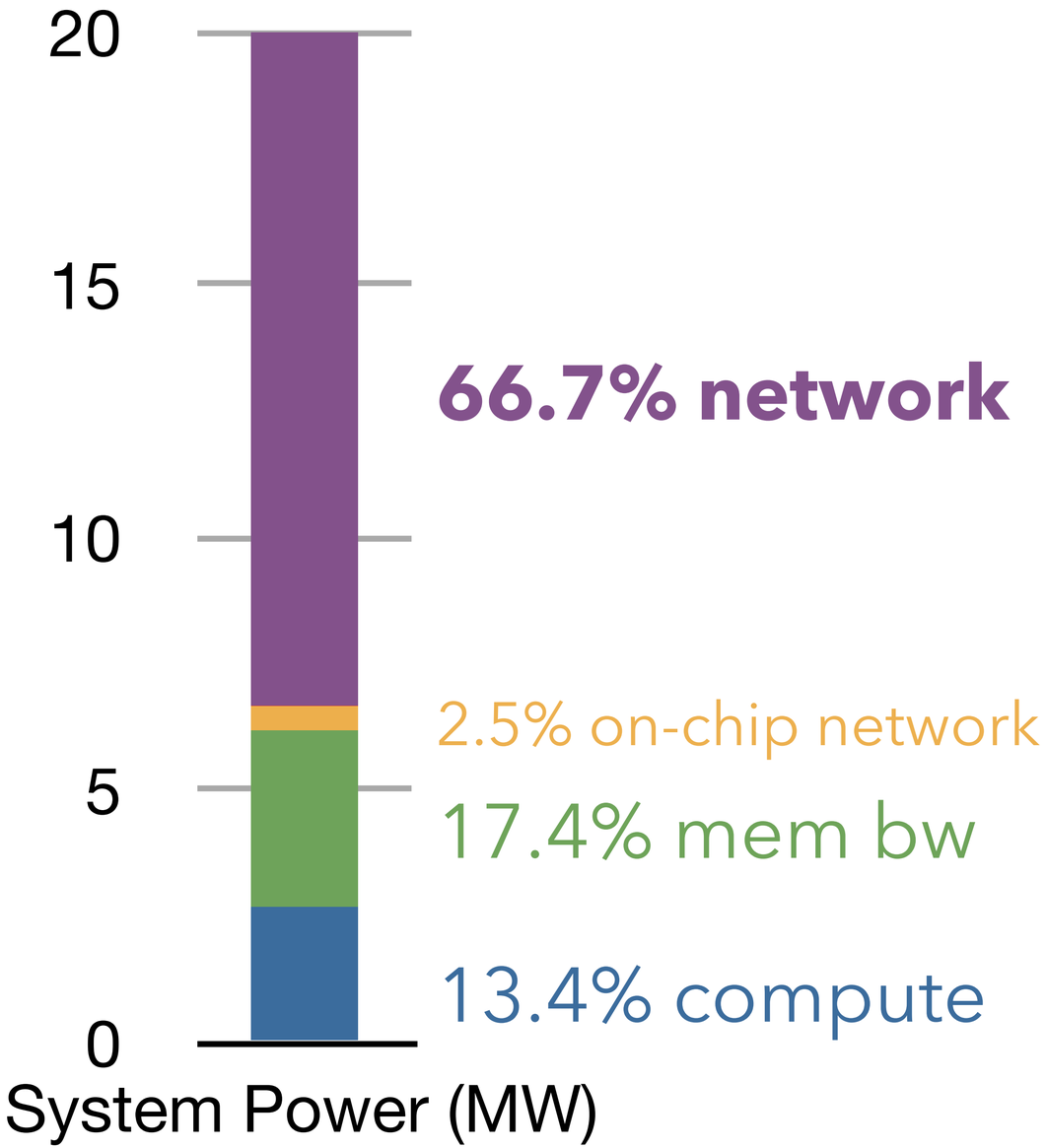
8% node overhead

P.S.: R_{max} / R_{peak} ~ 75%

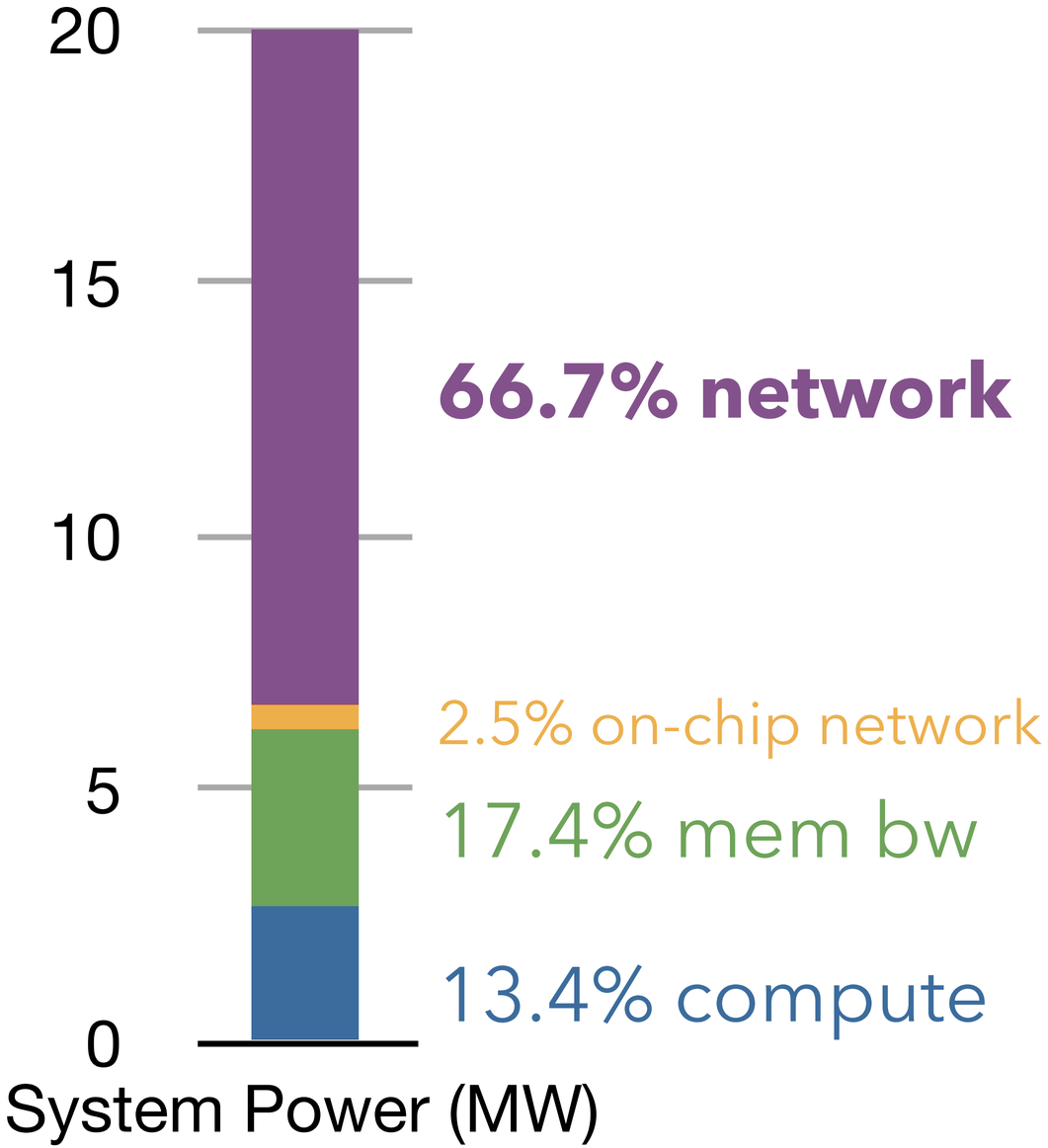
Power allocation for an “optimal” 3D FFT machine?

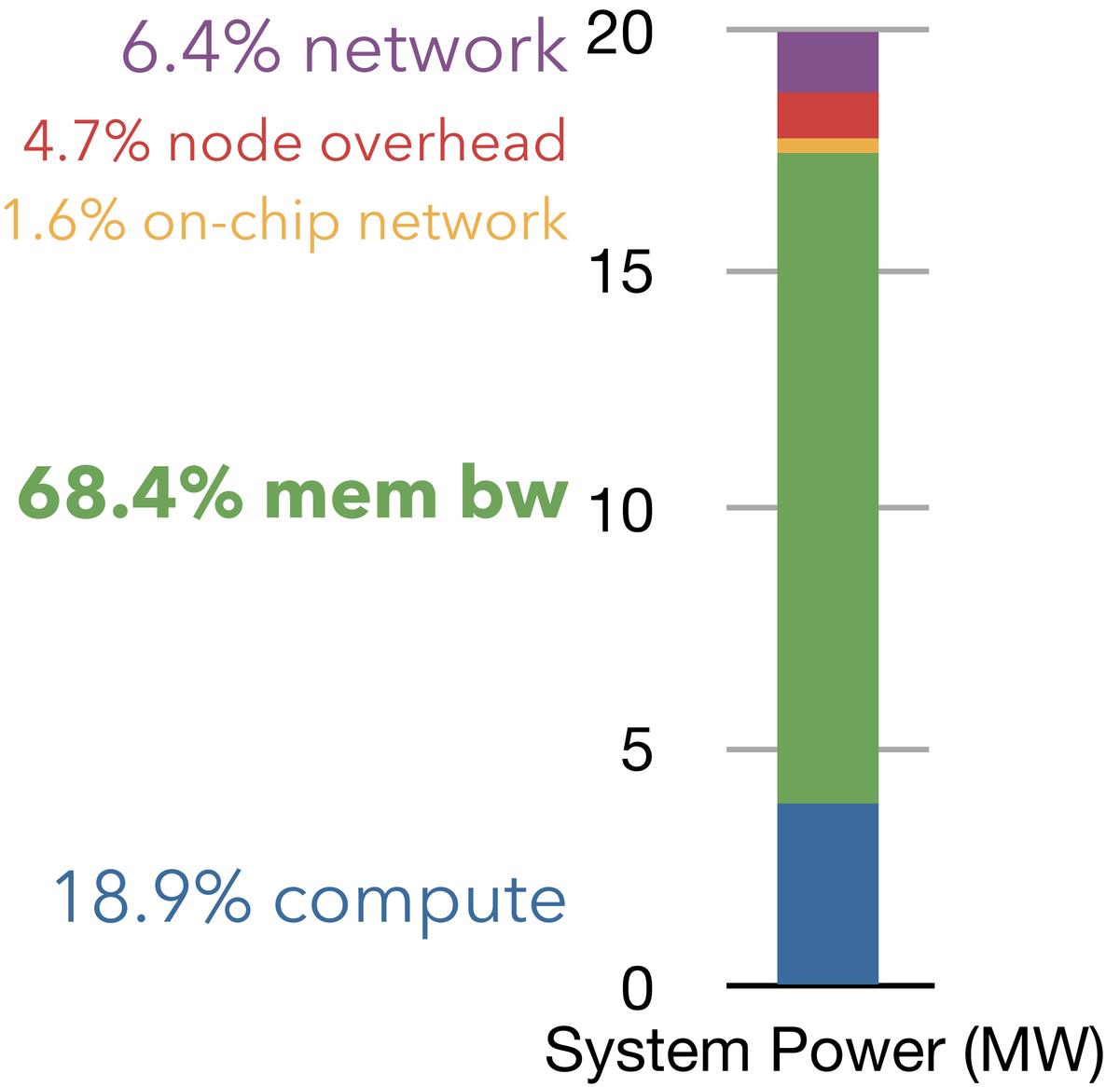
K. Czechowski, R. Vuduc. A theoretical framework for algorithm-architecture co-design. In *IPDPS'13*. doi:10.1109/IPDPS.2013.99

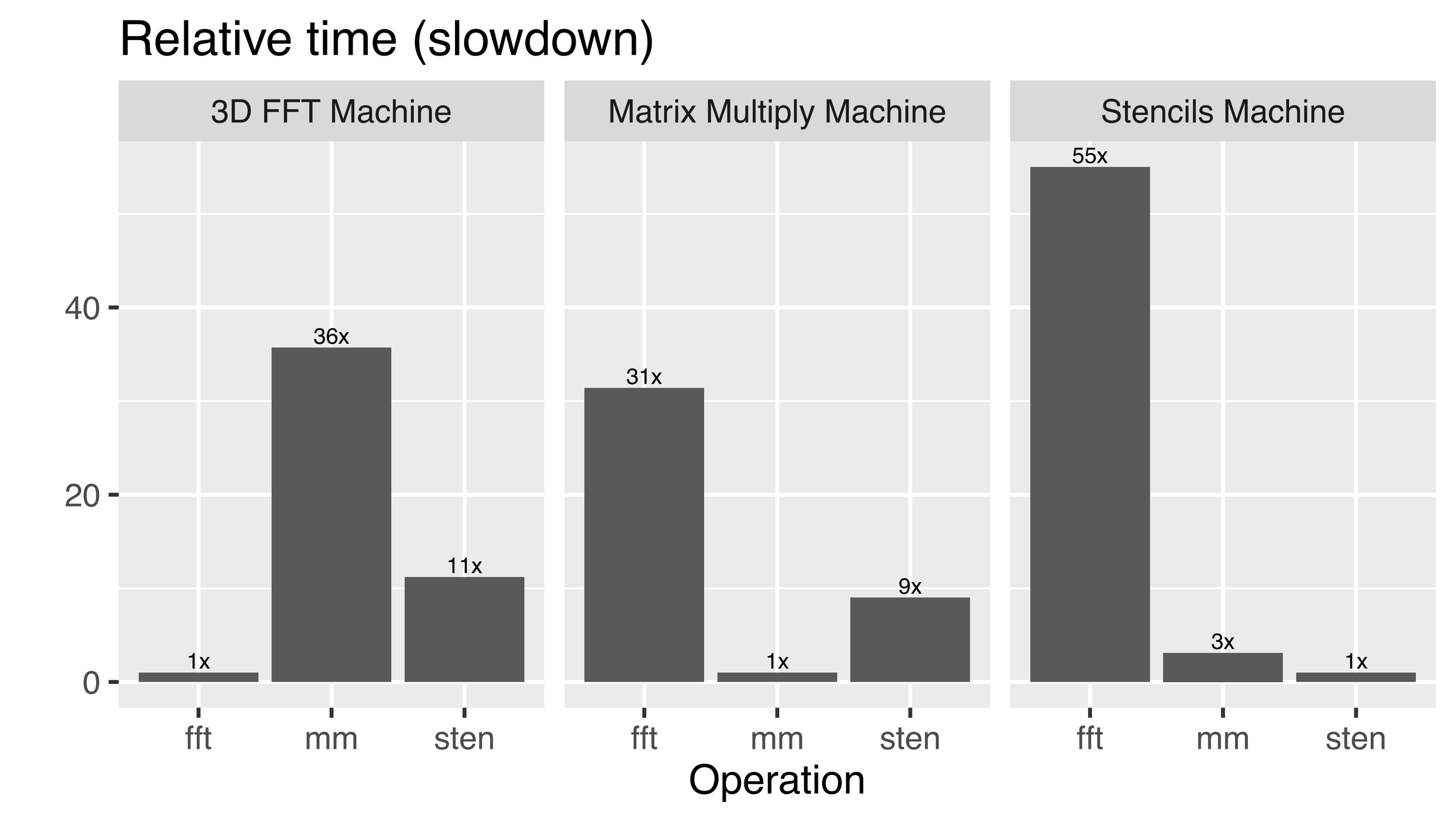
Power allocation for an “optimal” 3D FFT machine

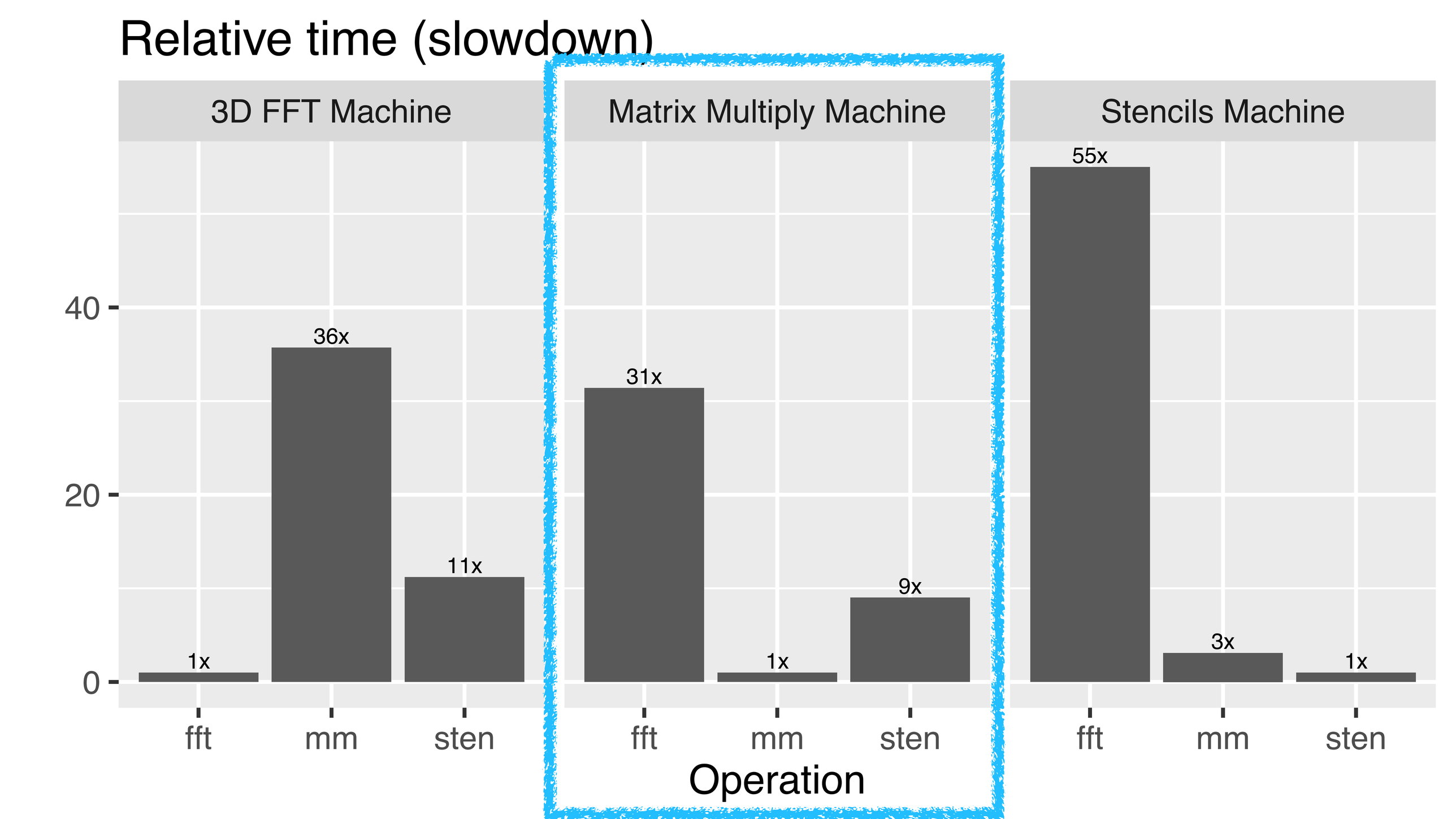


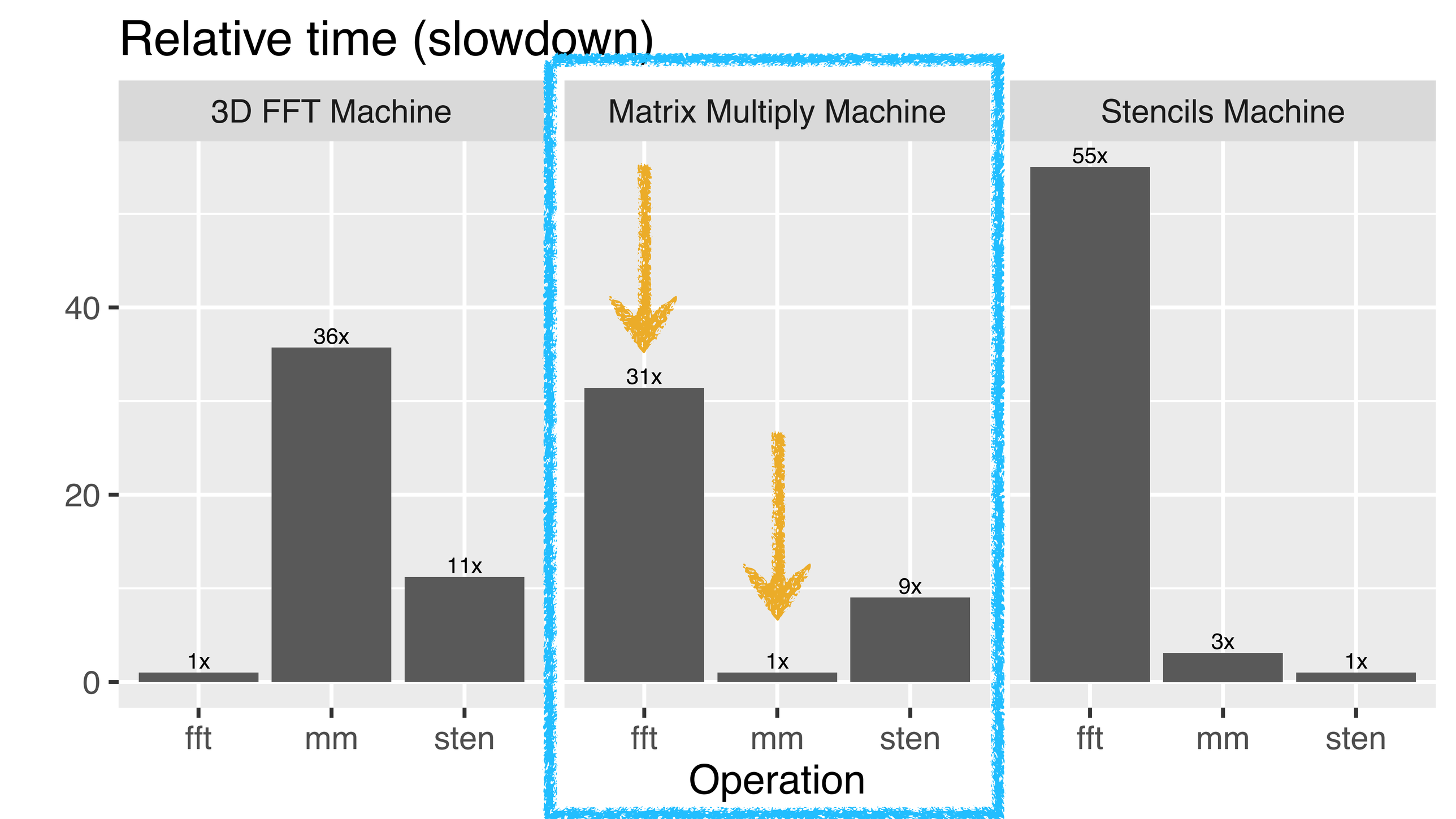
3D FFT vs. “Stencil” machines






Relative time (slowdown)
3D FFT Machine
Matrix Multiply Machine
Stencils Machine
40
20
0
36x
31x
55x
11x
9x
3x
1x
1x
1x
fft
mm
sten
fft
mm
sten
fft
mm
sten
Operation


Relative time (slowdown)
3D FFT Machine
Matrix Multiply Machine
Stencils Machine
40
20
0
36x
11x
1x
31x
1x
9x
55x
3x
1x
fft
mm
sten
fft
mm
sten
fft
mm
sten
Operation


Relative time (slowdown)
3D FFT Machine
Matrix Multiply Machine
Stencils Machine
40
20
0
36x
11x
1x
31x
1x
9x
55x
3x
1x
fft
mm
sten
fft
mm
sten
fft
mm
sten
Operation

Intelligence Advanced Research Projects Activity

IARPA

Creating Advantage through Research and Technology

AGILE

ADVANCED GRAPHIC INTELLIGENCE LOGICAL COMPUTING ENVIRONMENT

Schematic of the AGILE Co-Design Process

of the applications. AGILE system designs must emphasize optimization of the fully integrated system rather than independent optimization of individual functionalities (e.g., memory, computation, or communication), and must not be constrained by existing component interfaces and protocols, legacy architectures, or current practices.

A fundamental rethinking of computer architectures that can revitalize performance growth trends in computing capabilities is long overdue. Currently, there is a renewed interest in developing specialized hardware components. However, this approach will not resolve the fundamental data movement challenges that restrict the historical performance growth trends. The AGILE program will seed a new generation

The AGILE BAA was released in November 2021 and the program is slated to run for three years.

TESTING AND EVALUATION PARTNERS

- Lawrence Berkeley National Laboratory
- Sandia National Laboratory
- Pacific Northwest National Laboratory

KEYWORDS

- Computer Architecture
- Data analytics
- Co-Design
- Data movement
- Modeling and simulation