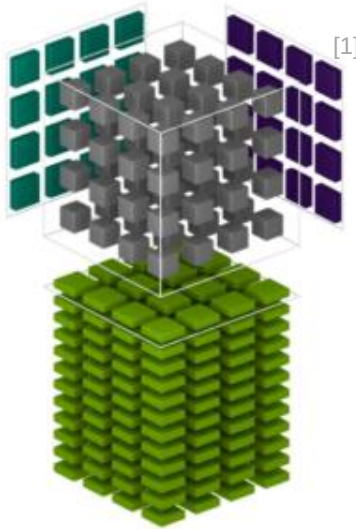


[1]

Matrix Engines for HPC: A Paragon of Performance or Grasping at Straws?

Jens Domke, Emil Vatai, Aleksandr Drozd, Peng Chen, Yosuke Oyama, Lingqi Zhang, Shweta Salaria, Daichi Mukunoki, Artur Podobas, Mohamed Wahib, Satoshi Matsuoka

(Collab. betw. researcher of RIKEN R-CCS, AIST, Tokyo Tech, and KTH)

Jens Domke, Dr. rer. nat. < jens.domke@riken.jp >

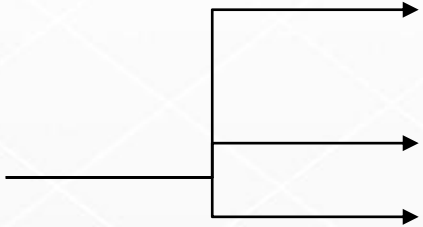
High Performance Big Data Research Team, RIKEN R-CCS, Kobe, Japan

Outline

- Motivation for Matrix Engines
- GEMM utilization in…
  - Historical Data of the K Computer
  - Software Libraries and Dependencies
  - Traditional and emerging HPC Workloads
- Discussion
  - Extrapolating a few (realistic?) Scenarios
  - Pros & Cons of adapting MEs
- Conclusion

Motivation – Tensor Cores for Deep Learning

- Deep Learning is driving commercial availability of matrix-multiplication units, as well as custom silicon (e.g., NVIDIA Tesla, IBM Power10, Intel Sapphire Rapids, Google TPUs, etc.)



- **Motivation:**
  - **More ME for DL**
    - **Saves power, consumes dark silicon**
    - **Faster DL training / inference**
    - **Can be used to accelerate other linear algebra (e.g., BLAS3)**
- Resulting Research Questions for HPC field:
  - **Q1**: How much do HPC workloads actually depend on BLAS / GEMM operations?
  - **Q2**: How well do our DL workloads utilize existing matrix engines?
  - **Q3**: Based on Q1/Q2, how much practical benefit can we expect for HPC centers?
  - **Q4**: Should the HPC community actively invest in ME units?

Motivation – TCs are escaping GPUs

- Compute in DL, for now, is formulated as dense matrix operations (i.e., convolution as im2col+gemm)
- Vendors reaction to DL workloads ➔ **Matrix Engines (MEs)**
- Matrix engines are **dedicated matrix-matrix multiply units** (e.g., implemented via systolic arrays)
- TCs and MEs (various sizes) yield perf. improvements for low-precision ops

Type	System	Tech.	Die size	ME size	Tflop/s (f16)	Tflop/s (f32)	Tflop/s (f64)	Support
General	Intel Sapphire Rapids[1]	10 nm	—	16x32	—	—	—	f16
	IBM Power10 [2]	7 nm	602 mm^2	4x4	16.4 (27.2 GF/mm^2)	8.2 (13.6 GF/mm^2)	4.1 (6.8 GF/mm^2)	f16, f32, f64
	NVIDIA Tesla V100	12 nm	815 mm^2	4x4x4	125.0 (153.4 GF/mm^2)	15.7 (19.3 GF/mm^2)	7.8 (9.6 GF/mm^2)	f16
	NVIDIA Tesla A100[3]	7 nm	826 mm^2	4x4x4	312.0 (377.7 GF/mm^2)	19.5 (23.6 GF/mm^2)	19.5 (23.6 GF/mm^2)	f16, f32, f64
AI	Google TPUv2	20 nm	—	128x128	45.0 (—)	—	—	f16
	Google TPUv3	16 nm	—	128x128	90.0 (—)	—	—	f16
	Habana Labs Gaudi	16 nm	500 mm^2	Shared	—	—	—	f16, f32
	Huawei Ascend 910[4]	7 nm	1228 mm^2	16x16x16	256.0 (208.5 GF/mm^2)	—	—	f16

Overview of existing and emerging general-purpose and AI architectures that leverage matrix engines to accelerate computations

Motivation – Scalar ➔ Vector ➔ Matrix?

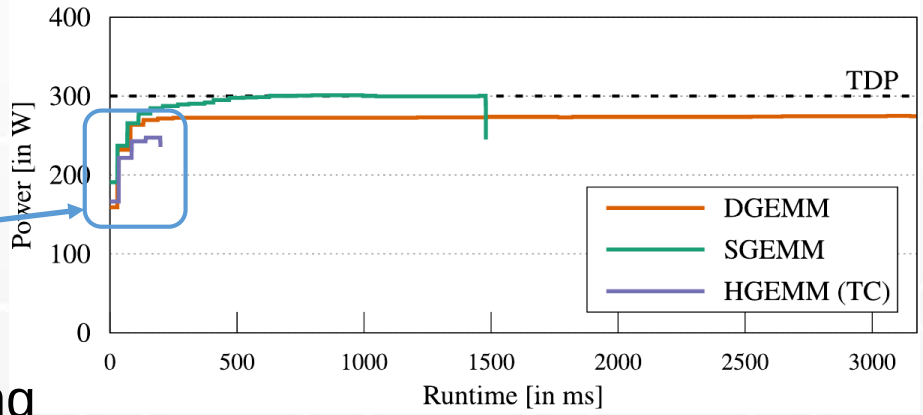
- **Vectors** units can efficiently **increase performance** and energy efficiency
  (2.3x higher energy eff. w/ AVX2)

- Going from vector to **matrix units** seems like a **natural extension** (➔ clear runtime and power benefit for FP16 GEMM)

- *Counter-question:* Are MEs really what we should be spending our silicon on, given that Moore's law is dying out?

Precision	Vector extension	Walltime	Energy-efficiency
DGEMM	—	34.22 s	1.23 Gflop/J
	AVX2	12.49 s	2.92 Gflop/J
SGEMM	—	16.79 s	2.65 Gflop/J
	AVX2	6.36 s	5.92 Gflop/J

Energy-eff. of Vector Extensions on a Intel Xeon CPU (measured with PCM)



Power consumption evaluation of GPU cores and TCs on a single Tesla V100 GPU

GEMM in… Historical Data of the K Computer

- RIKEN's operations team collected light job statistics
  - Runtime, binary names, symbol table of main binary (via `nm`), etc.
- Recording period of **April '18 to March '19**
  - **487,563 scientific applications consuming 543 million node-hours**
  - For 96% of these we have `nm` data*

➔ Analysis

- **Binary/jobs with GEMM** calls in them: **277,258,182 node-hours (53.4%)**
- Amount of GEMM within each job unknown ☹
- **Best case** (all GEMM; infinitely fast MEs): $\approx 1/2$ **node-hours** (Amdahl's law)

(*remark: nm from external libs missing, but Fujitsu's mathlib SSL2 usually linked statically; and functions are included on as-needed basis)

GEMM in… Software Lib and Dependencies

- Spack: popular package manager for **4371 science/HPC codes**
- Provides **numerous math libraries** (Atlas, BLIS, Eigen, MKL, etc.) and **tracks package dependencies**
  - We identified all libraries which provide dense linear algebra functions
  - We create dependency trees to also catch indirect dependency on BLAS

➔ Analysis

- **14 math libs**
- **226 (or 9%) directly** and 1311 indirectly dependent on “BLAS”
- **Best case: 51%** of packages can be fully/partially **enhanced by MEs**

Dependency Distance	# and % of Packages	excluding py-* & R-*
0	14 (0.32)	14 (0.55)
1	239 (5.47)	226 (8.87)
2	762 (17.43)	541 (21.23)
3	968 (22.15)	714 (28.02)
1–∞	3061 (70.03)	1311 (51.45)

Dependency Analysis of Dense Linear Algebra Libs for Spack (w/ & w/o Python and R sub-packages)

GEMM in… Deep Learning Workloads

- Utilize benchmarker: https://github.com/undertherain/benchmarker/
- Examining different, **common AI models and kernels**

Set	Name	Sci. / Eng. / AI Domain	Name	Sci. / Eng. / AI Domain	Name	Sci. / Eng. / AI Domain
Deep Learning	BERT	Natural Language Processing	DeepLabV3	Image Segmentation	GRU	Single Layer
	Cosmoflow	Computational Cosmology	SSD300	Object Detection	LSTM	Single Layer
	VGG16	Image Recognition	NCF	Recommender Systems	Conv2D	Single Layer
	Resnet50	Image Recognition	GEMM	Single Layer	Attention	Single Layer

- Execute **proxy workloads (similar** characteristic **to MLperf**)**
- Collect metrics via (py)NVML and NVIDIA's nvprof on single server
- Speedup: FP32 (non-TC) vs. mixed-precision training (w/ TCs)*

(*reason: limited control given to user by pytorch & cuDNN;
**1. MLPerf is the most widely used benchmark suite for AI/ML [www.mlperf.org])

GEMM in… Deep Learning Workloads

- How much does DL generally benefits from MEs?
- Identifying TC kernels from profiling on Tesla V100
  - **Avg. 2x** (eg. ConvNets) and up to 4x (for Transformers)
  - Not as high as GEMM (**7.6x**)
  - Yet substantial speedup (➔ justifies TCs)
- Recall: speedup partially result of lower precision and MEs

Benchmark	Speedup	% TC	% TC comp	% Mem
BERT	3.39x	50.86	55.26	7.97
Cosmoflow	1.16x	0.04	0.05	22.90
VGG16	1.71x	12.30	12.74	3.45
Resnet50	1.97x	16.32	16.78	2.76
DeepLabV3	1.75x	16.33	16.44	0.69
SSD300	1.78x	8.55	8.66	1.32
NCF	0.97x	22.37	26.79	16.50
GEMM	7.59x	20.08	99.90	79.90
GRU	3.67x	6.59	7.48	11.94
LSTM	5.69x	11.63	13.85	16.03
Conv2D	1.12x	0.27	0.32	16.78
Attention	3.49x	44.49	58.19	23.55

Throughput Improvement from FP32 to Mixed Prec. + TCs

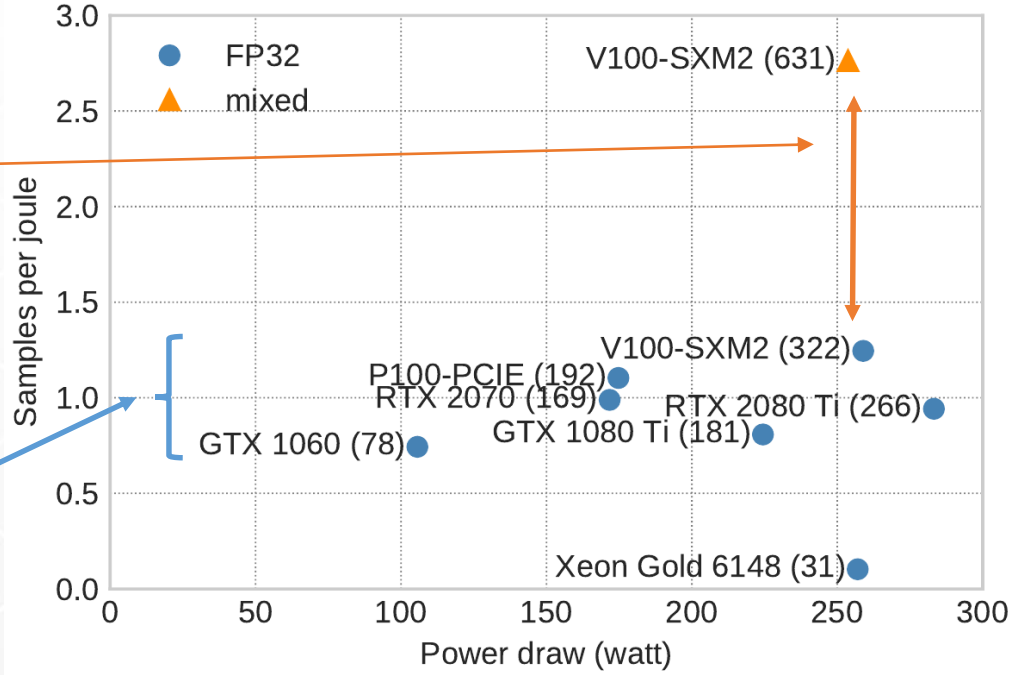
%TC: percentage of time on TCs (relative to total time); %TC comp: comp. time s pent on TCs excl. data move.; and %Mem: time for data mov. between host->dev

- **Note: assuming AI/ML loads cont. to be formulated as dense matrix ops**

GEMM in… Deep Learning Workloads

…not only speedup, but:

- Mixed precision (TCs) yields > 2x improvement in energy efficiency (over FP32)
- Before, architectural improvement for FP32-based Resnet50 were marginal (only 0.8 ➔ 1.3 images/Joule from GTX1060 to V100)



Energy-efficiency of ResNet50 training

GEMM in… 77 traditional HPC Workloads

- TOP500 benchmarks: **HPL and HPCG** (Intel's version)
  - Used by community for world-wide supercomputer ranking
- Exascale Computing Project **(ECP) Proxy Applications**
  - Used for procurement of exascale systems by HPC centers in USA
  - Version 1.0 contains 12 workloads (we excluded CANDLE)
- RIKEN CCS' **Fiber Miniapp Suite**
  - 8 proxy apps used in procurement of Supercomputer Fugaku
  - Represent the priority areas of the Japanese government
- **SPEC Benchmarks** (CPU 2017 V1.1 & OMP 2012 V1.1 & MPI 2007 V2.0.1)
  - Widely accepted benchmark set for industry and HPC vendors

GEMM in… 77 traditional HPC Workloads

Computer simulations create the future

Set	Name	Sci. / Eng. / AI Domain	Name	Sci. / Eng. / AI Domain	Name	Sci. / Eng. / AI Domain
TOP500	HPL	Math/Computer Science	HPCG	Math/Computer Science		
ECP	AMG	Physics and Bioscience	miniAMR	Geoscience/Earthscience	SW4lite	Geoscience/Earthscience
	CoMD	Material Science/Engineering	miniFE	Physics	SWFFT	Physics
	Laghos	Physics	miniTRI	Math/Computer Science	XSBench	Physics
	MACSio	Math/Computer Science	Nekbone	Engineering (Mechanics, CFD)		
RIKEN	FFB	Engineering (Mechanics, CFD)	mVMC	Physics	NTChem	Chemistry
	FFVC	Engineering (Mechanics, CFD)	NGSA	Bioscience	QCD	Lattice QCD
	MODYLAS	Physics and Chemistry	NICAM	Geoscience/Earthscience		
SPEC CPU	blender(R)	Math/Computer Science	exchange2	Artificial Intelligence	omnetpp	Math/Computer Science
	cam4(R)	Geoscience/Earthscience	fotonik3d	Physics	perlbench	Math/Computer Science
	namd(R)	Material Science/Engineering	gcc	Math/Computer Science	pop2	Geoscience/Earthscience
	parest(R)	Bioscience	imagick	Math/Computer Science	wrf	Geoscience/Earthscience
	povray(R)	Math/Computer Science	lbm	Engineering (Mechanics, CFD)	roms	Geoscience/Earthscience
	bwaves	Physics	leela	Artificial Intelligence	x264	Math/Computer Science
	cactuBSSN	Physics	mcf	Math/Computer Science	xalancbmk	Math/Computer Science
	deepsjeng	Artificial Intelligence	nab	Material Science/Engineering	xz	Math/Computer Science
SPEC OMP	applu331	Engineering (Mechanics, CFD)	fma3d	Physics	mgrid331	Engineering (Mechanics, CFD)
	botsalgn	Bioscience	ilbdc	Engineering (Mechanics, CFD)	nab	Chemistry
	botsspar	Math/Computer Science	imagick	Math/Computer Science	smithwa	Bioscience
	bt331	Engineering (Mechanics, CFD)	kdtree	Math/Computer Science	swim	Geoscience/Earthscience
	bwaves	Engineering (Mechanics, CFD)	md	Material Science/Engineering		
SPEC MPI	[d]leslie3d	Engineering (Mechanics, CFD)	[l]GemsFDTD	Physics	socorro	Material Science/Engineering
	[d]milc	Lattice QCD	lu	Engineering (Mechanics, CFD)	tachyon	Math/Computer Science
	fds4	Engineering (Mechanics, CFD)	[l]wrf2	Geoscience/Earthscience	tera_tf	Geoscience/Earthscience
	GAPgeofem	Physics	pop2	Geoscience/Earthscience	zeusmp2	Engineering (Mechanics, CFD)
	lammps	Material Science/Engineering	RAxML	Bioscience		

GEMM in… 77 traditional HPC Workloads

Measurement methodology to identify GEMM kernels

- Created a **Score-P library wrapper** for all dense compute functions **of MKL** ((C)BLAS, PBLAS, ScaLAPACK, etc.)
- TOP500, ECP, and Fiber proxies:
  - Kernel isolation; compiler settings; input selection acc. to previous work [1]
  - **Link against Score-P wrapper** and manual instrument all source-code location referring to GEMM or Fortran's `matmul` intrinsic
- SPEC benchmarks (unfortunately all external libraries striped out):
  - Prioritize GNU compilers (with `–O3 –march=native`) and `mtrain` input set
  - Find **compute-intensive kernels with Intel Advisor** ➔ Manually inspect 598 source code locations ➔ **instrument all GEMM** ops (via Score-P)

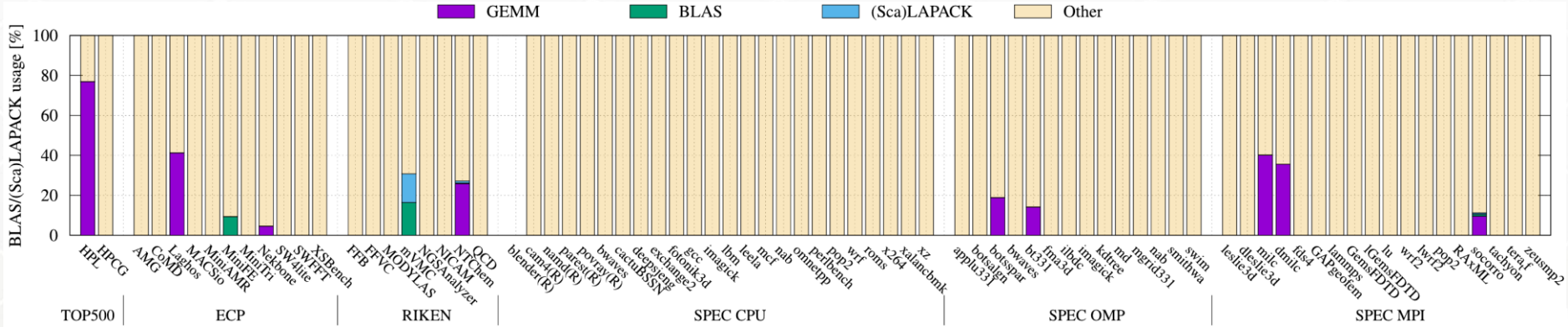
[1] J. Domke et al., "Double-precision FPUs in High-Performance Computing: an Embarrassment of Riches?" in IPDPS19, 2019.

GEMM in… 77 traditional HPC Workloads

Computer simulations create the future

- **HPL: highest GEMM** util. with 76.81% and 0.14% in other BLAS calls
- Other BMs performing GEMM: **Laghos (41.24%), NTChem (25.78%),** Nekbone (4.58%), and SPEC OMP's botsspar (18.9%) and bt331 (14.16%), and SPEC MPI's **milc & dmilc (40.16% & 35.57%**) and socorro (9.52%)
- **Non-GEMM BLAS & LAPACK even less utilized**: highest in mVMC (31%)

➔ Assume **equal distribution: only 3.5%** spent in GEMM (cf. ≈12.5d per year)



GEMM, BLAS (non-GEMM functions), and (Sca)LAPACK utilization across 77 HPC benchmarks

Remark: non-GEMM & (Sca)LAPACK kernels not profiled for SPEC BMs

What if… we have/had MEs?

- Time for **thought experiment**!, we have/know:
  - Breakdown of GEMM vs. non-GEMM cycles (last slide)
  - Node-hours per science domain for RIKEN [1] and ANL [2] and hypothetical future HPC system with 20% AI cycles
- Methodology:
  - Select **one application (of our 77) with highest GEMM** percentage for each science domain (material science, chemistry, biology, etc.)
  - Assume **app rep. all cycles spent per domain** (assume 10% for "other")
  - No MEs≅100% node-hours; assume **various ME speedup** conf. (2, … , ∞)
  - Ignore other inefficiencies: downtime, I/O, init/post-proc., under-utilization, etc.

➔ **Estimate reduction in node-hours** per system / HPC site

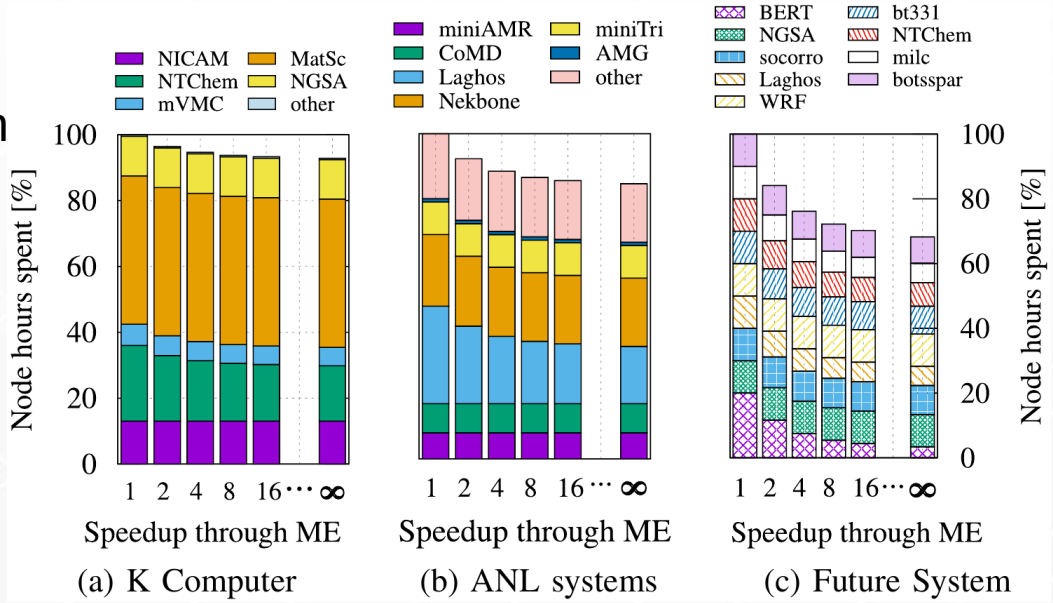
[1] AICS, "K computer Annual Report 2016-17," RIKEN Advanced Institute for Computational Science, Tech Report.
[2] J. Collins et al., "2016 Annual Report - Argonne Leadership Computing Facility," Argonne National Laboratory, TR: ANL/ALCF-17/1.

What if… we have/had MEs?

Computer simulations create the future

Node-hour reduction extrapolation

- Future system w/ 10% equal distribution per science domain and **20% AI/DL/ML**
  (➔ realistic for convergence of HPC and AI?)
- Under **ideal conditions w/ 4x ME speedup***:
  - **5.3%** on K & **11.5%** @ANL
  - 23.8% future system



Node hour reduction by utilizing hypothetical ME

➔ Please, do your own extrapolation based on <u>your</u> workloads

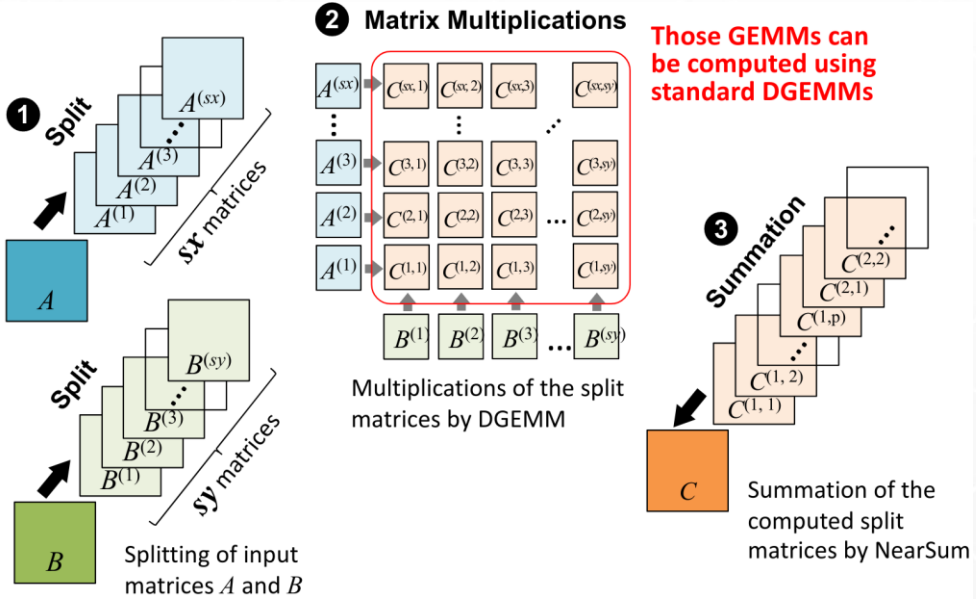
(*remark: 2-4x realistic w/o precision reduction as shown earlier)

What if… we only have low-precision MEs?

- Assuming: avail. **MEs without support FP64 ops**

  ➔ Are those low-precision MEs still valuable?

- Using **Osaki scheme** [1] to "***emulate"* higher precision numerics** using low-precision arithmetics by separately calc. of mantissa and exponent
- Number of splits depends on required accuracy and value range of inputs [2]



GEMM with Ozaki scheme (image from [2])

[1] K. Ozaki et al., "Error-free transformations of matrix multiplication by using fast routines of matrix multiplication & its applications," in Numer. Algor., 2012.

[2] D. Mukunoki et al., "Accurate BLAS Implementations:OzBLASand BLAS-DOT2," in LSPANC 2020 January, 2020

What if… we only have low-precision MEs?

- Experiments on a NVIDIA Tesla V100
- Comparing cuBLAS routines and emulated GEMM using TCs
- NVML used for W measurement
- [S|D]GEMM-TC can't outperform cuBLAS on V100
- …but can mitigate the lack of FP32 FP64 MEs with reasonable perf. loss

Implementation	Condition	Tflop/s	Watt	Gflop/J
cublasGemmEx	FP16/FP32-mixed	92.28	270.9	340.70
cublasSgemm	—	14.54	276.1	52.66
cublasDgemm	—	7.20	286.5	25.14
SGEMM-TC	input range: 1e+8	4.721	284.1	16.62
	input range: 1e+16	2.140	278.9	7.67
	input range: 1e+32	1.758	264.5	6.65
DGEMM-TC	input range: 1e+8	1.097	273.7	4.01
	input range: 1e+16	0.716	271.4	2.64
	input range: 1e+32	0.618	270.4	2.29

Performance of cuBLAS routines vs. GEMM-TC (software emulation using TCs; m = n = k = 8192)

(remark: DGEMM-TC outperforms cublasDgemm on a NVIDIA Titan RTX due to limited FP64 units)

What if…? ➔ Opportunities of MEs in HPC

- Effective use of "**Dark Silicon**"
  - NVIDIA's FPUs and TCs cannot be used simultaneously
  - What other resources would we put on GPUs instead of TCs?
- **Other compute patterns** benefiting from matrix engines
  - Accelerating sparse matrix multiplication [1] ➔ Blocked sparse formats
  - Automatically transform compute-intense nest loops to TCs [2]
- **Lower/mixed precision** and AI in scientific computing
  - Convergence of DL/ML and traditional HPC?
  - Making mixed precision common in used numerical methods [3]

[1] O. Zachariadis et al., "Accelerating sparse matrix–matrix multiplication with GPU Tensor Cores," Computers & Electrical Engineering, 2020.
[2] S. G. Bhaskaracharya et al., "Automatic Kernel Generation for Volta Tensor Cores," 2020.
[3] A. Abdelfattah et al., "A Survey of Numerical Methods Utilizing Mixed Precision Arithmetic," 2020.

What if…? ➔ Challenges of MEs in HPC

- **Inefficiency for Level-1 and Level-2 BLAS**
  - MEs mostly build from Systolic Arrays which are usable for L1/L2 BLAS but inefficient ➔ SIMD/vector units suitable for all [1]
- **Programmability** burden (mostly hidden behind APIs and LAG libraries)
  - For some compute patterns hand-tuned or new libraries need to be written
  - Auto-generation of code for MEs still in very early stage [2]
- Reduced **portability**
  - SIMD & GPU-comp. already caused #ifdef nightmare ➔ won't get better?
- Is the Dark Silicon effect generalizable for other CPUs and GPUs?

[1] M. I. Soliman, "Performance Evaluation of Multi-Core Intel Xeon Processors on Basic Linear Algebra Subprograms," in ICCES 2008, 2008.
[2] S. G. Bhaskaracharya et al., "Automatic Kernel Generation for Volta Tensor Cores," 2020.

What if…? ➔ Challenges of MEs in HPC

- Overhead of **data staging** to MEs ➔ only current issue?
- **Vectors in A64FX already highly efficient** [1] ➔ make them longer?

	Main Processor	HPL-AI Measured Performance	FP16 Peak Performance (full machine)	Efficiency	HPL-AI Performance /Chip	Top500 /Linpack FP64 Measured Performance	FP64 Peak Performance	Efficiency
1. Fugaku	Fujitsu A64FX	**2.00 EF**	**2.14 EF**	**93.2%**	**12.6TF**	442.01 PF	537.21 PF	82.3%
2. Summit	NVIDIA V100	**0.55 EF**	**3.46 EF**	**15.9%**	**19.9TF**	148.60PF	200.79 PF	74.0%
3. Selene	NVIDIA A100	**0.25 EF**	**2.55 EF**	**9.8%**	**30.6TF**	63.46 PF	79.22 PF	80.1%

- Most **HPC** problems (currently? or inherently?) **memory-bound** [2]
- **DL** might be moving towards **sparse models / algorithms / data** [3]

[1] S. Matsuoka, “How we might achieve another 100x for Fugaku-Next,” in 3rd R-CCS International Symposium, 2021.
[2] J. Domke et al., “Double-precision FPUs in High-Performance Computing: an Embarrassment of Riches?” in IPDPS19, 2019.
[3] T. Hoefler et al., “Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks,” Tech Report, Feb. 2021.

Summary & Conclusion

We have shown that:

- Less than 54% of CPU cycles was consumed by workloads which could have called GEMM in 1 year of RIKEN's K computer operation
- Less than 9% of Spack's (scientific) software directly links to BLAS
- Occurrence/**usage of matrix operations** in our experiments is **underwhelming** (excluding DL and HPL)
- In 77 benchmarks: **only 3.4% of time** in aggregate spend in GEMM
- **Typical speedup is ≈2x in DL** workloads (up to 4x possible) despite the 8x theoretical advantage of tensor cores for GEMM
- Lower precision TCs can be used to emulate high-prec. GEMM ops
- There are a **few opportunities; but no clear evidence** signaling that the future of **HPC would be radically transformed** by MEs

Summary & Conclusion

Utility of Matrix Engines: let's recapitulate our initial questions

Q1: How much do **HPC workloads** actually depend on BLAS / GEMM operations?

➔ **Very little**; on avg. ≤ 4% across 77 BMs

Q2: How well do our **DL workloads** utilize existing matrix engines?

➔ **Less than expected;** avg. of 2x speedup

Q3: Based on Q1/Q2, how much **practical benefit** can we expect for HPC centers?

➔ Below 25% in ideal case; **realistic <<10%**

Q4: Should the HPC community **actively invest** in ME units & **demand FP64** MEs?

"free as in beer"

Open-Source & Acknowledgements

- Reproducing our data? Doing your own analysis?
  ➔ Use our framework:

https://gitlab.com/domke/MEstudy or

This work was supported by

- Japan Society for the Promotion of Science **KAKENHI Grant Number 19K20286**;
- JST, **PRESTO Grant Number JPMJPR20MA**, and **CREST JPMJCR1687**, Japan;
- New Energy and Industrial Technology Development Organization (**NEDO**);
- **AIST**/TokyoTech Real-world Big-Data Computation Open Innovation Laboratory (RWBC- OIL);
- **Cygnus HPC system** installed at the Multi-disciplinary Cooperative Research Program of the Center for Computational Sciences, University of Tsukuba

Figure sources

- [1] https://developer.nvidia.com/blog/programming-tensor-cores-cuda-9/