An Overview of High Performance Computing and Using Mixed Precision in Numerical Computations to Speedup Linear Algebra Solvers

Jack Dongarra
University of Tennessee
Oak Ridge National Laboratory
University of Manchester

Outline

- Overview of High Performance Computing
- Some ideas for using mixed precision in scientific computations

State of Supercomputing in 2020

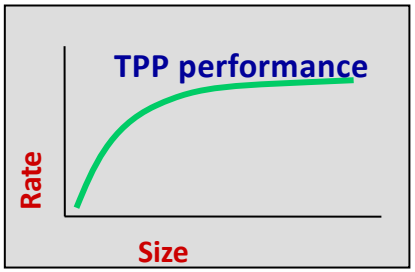
- Pflops (> 10^{15} Flop/s) computing fully established with all 500 systems.
- Three technology architecture possibilities or "swim lanes" are thriving.
  - Commodity (e.g. Intel)
  - Commodity + accelerator (e.g. GPUs) (144 systems; 134 NVIDIA, 6 Intel Phi + 4)
  - Lightweight cores (e.g. IBM BG, Xeon Phi, TaihuLight, ARM (3 systems))
- China: Top consumer and top producer overall.
- Interest in supercomputing is now worldwide, and growing in many new markets (~50% of Top500 computers are in industry).
- Intel processors largest share, 94%; followed by AMD, 2%.
- Exascale (10^{18} Flop/s) projects exist in many countries and regions.

TOP500 The List. Since 1993

H. Meuer, H. Simon, E. Strohmaier, & JD

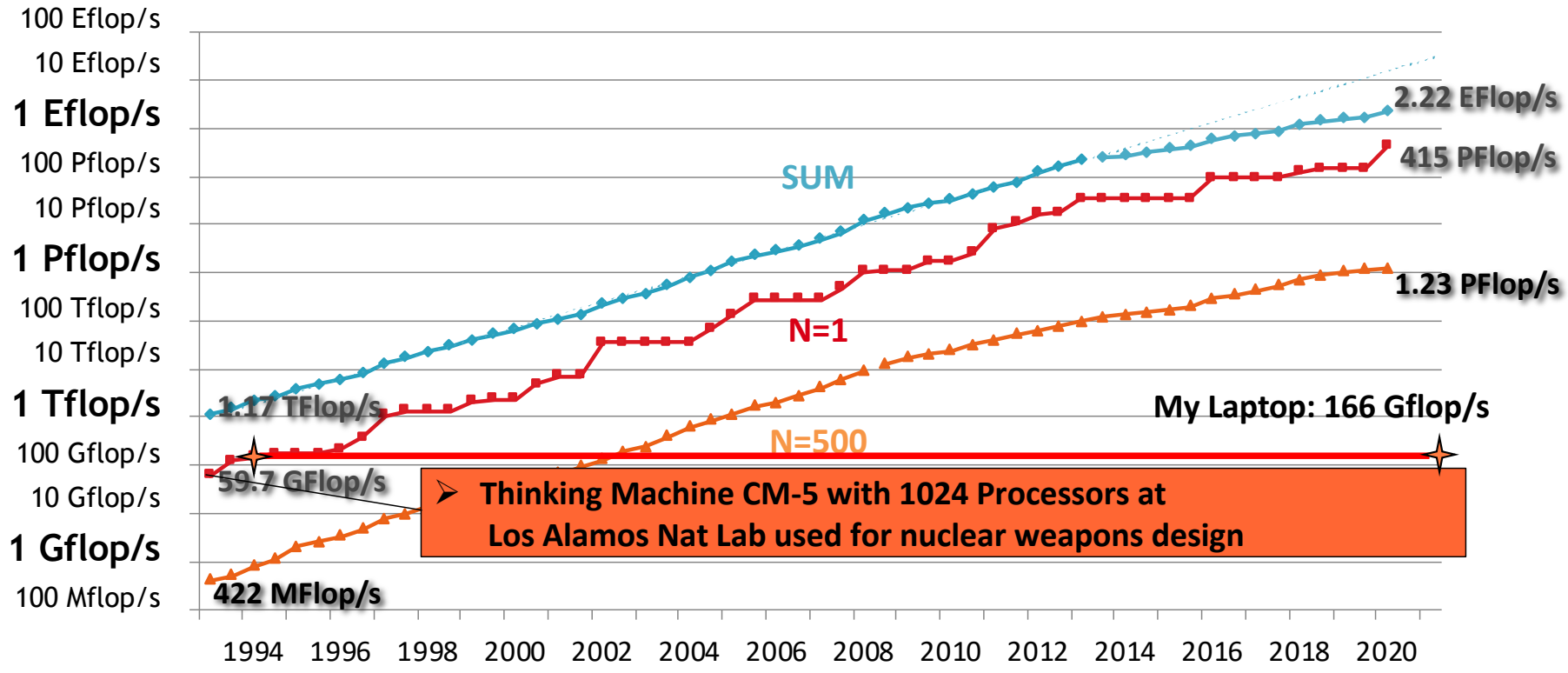
- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

  $Ax=b$, *dense problem*

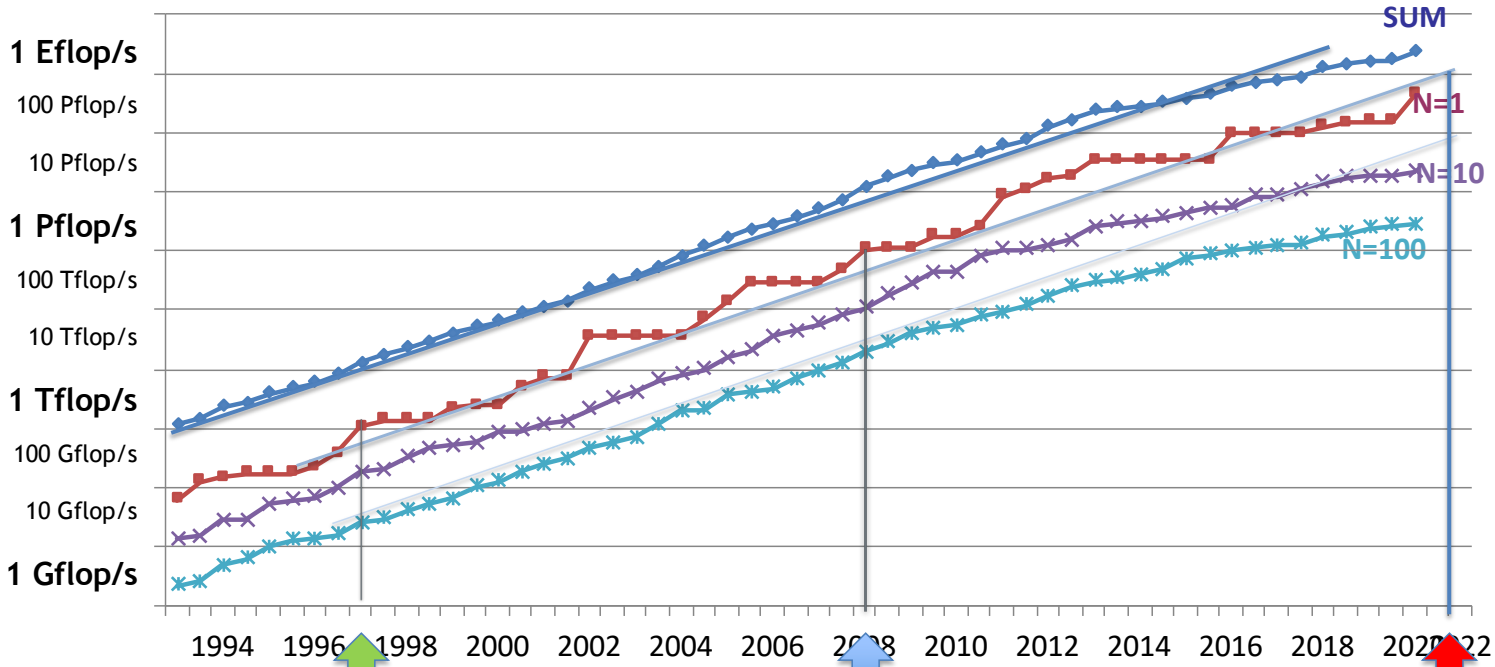


- Updated twice a year
  - SC‘xy in the States in November
  - Meeting in Germany in June
- All data available from **www.top500.org**

PERFORMANCE DEVELOPMENT



PERFORMANCE DEVELOPMENT



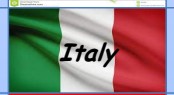
Tflops (10^{12}) Achieved ASCI Red Sandia NL

Pflops (10^{15}) Achieved RoadRunner Los Alamos NL

Eflops (10^{18}) Achieved?

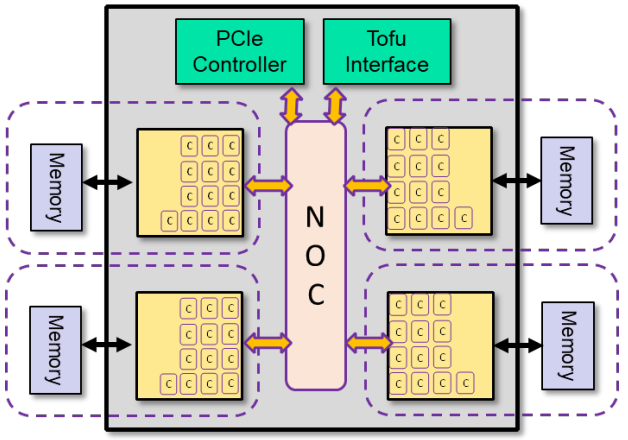
China says 2021-2022
U.S. says 2022-2023

June 2020: The TOP 10 Systems (1/3 of the Total Performance of Top500)

Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	GFlops/ Watt
1	RIKEN Center for Computational Science	Fugaku, ARM A64FX (48C, 2.2 GHz), Tofu D Interconnect		7,299,072	415.	81	28.3	14.7
2	DOE / OS Oak Ridge Nat Lab	Summit, IBM Power 9 (22C, 3.0 GHz), Nvidia GV100 (80C), Mellonox EDR		2,397,824	149.	74	10.1	14.7
3	DOE / NNSA L Livermore Nat Lab	Sierra, IBM Power 9 (22C, 3.1 GHz), Nvidia GV100 (80C), Mellonox EDR		1,572,480	94.6	75	7.44	12.7
4	National Super Computer Center in Wuxi	Sunway TaihuLight, SW26010 (260C) + Custom		10,649,000	93.0	74	15.4	6.05
5	National Super Computer Center in Guangzhou	Tianhe-2A NUDT, Xeon (12C) + MATRIX-2000 + Custom		4,981,760	61.4	61	18.5	3.32
6	Eni S.p.A	HPC5, Dell EMC PowerEdge C4140, Xeon (24C, 2.1 GHz) + Nvidia V100 (80C), Mellonax HDR		669,760	35.5	69	2.25	15.8
7	NVIDIA Corporation	Selene, Nvidia DGX AMD (64C, 2.25 GHz) + Nvidia A100 (108C), Mellanox HDR		277,760	27.6	80	1.34	20.6
8	Texas Advanced Computing Center / U of Texas	Frontera, Dell C6420, Xeon Platinum, 8280 28C 2.7 GHz, Mellanox HDR		448,448	23.5	61		
9	CINECA	Marconi-100, IBM Power System AC922, P9 (16C, 3 GHz) + Nvidia V100 (80C), Mellonox EDR		347,776	21.6	74	1.98	10.9
10	Swiss CSCS	Piz Daint, Cray XC50, Xeon (12C) + Nvidia P100 (56C) + Custom		387,872	21.2	78	2.38	8.90

Fugaku's Fujitsu A64fx Processor is…

- A Many-Core ARM CPU…
  - 48 compute cores + 2 or 4 assistant (OS) cores
  - New core design
  - Near Xeon-Class Integer performance core
  - ARM V8 --- 64bit ARM ecosystem
  - Interconnect Tofu-D

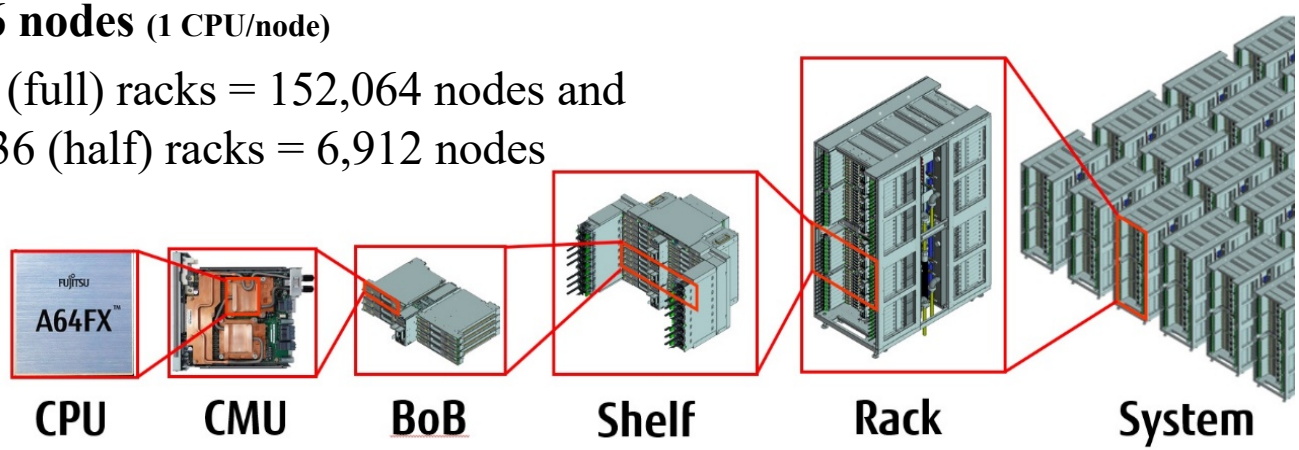


- …but also an accelerated GPU-like processor
  - SVE 512 bit x 2 vector extensions (ARM & Fujitsu)
    - Integer (1, 2, 4, 8 bytes) + Float (16, 32, 64 bytes)
  - Cache + memory localization (sector cache)
  - HBM2 on package memory – Massive Mem BW (Bytes/DPF ~0.4)
    - Streaming memory access, strided access, scatter/gather etc.
  - Intra-chip barrier synch. and other memory enhancing features

http://bit.ly/fugaku-report

Fugaku Total System Config & Performance

- **Total # Nodes: 158,976 nodes** (1 CPU/node)
  - 384 nodes/rack x 396 (full) racks = 152,064 nodes and
    192 nodes/rack x 36 (half) racks = 6,912 nodes



Footprint: 1,920 m^2

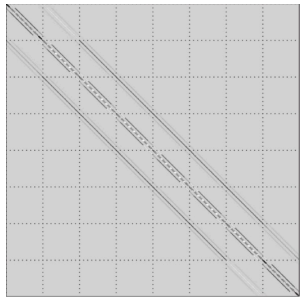
- **Theoretical Peak Compute Performances**
  - Normal Mode (CPU Frequency 2GHz)
    - **64 bit** Double Precision FP: **488 Petaflops**
    - **32 bit** Single Precision FP: **977 Petaflops**
    - **16 bit** Half Precision FP (AI training): **1.95 Exaflops**
    - **8 bit Integer** (AI Inference): **3.90 Exaops**
- **Theoretical Peak Memory BW: 163 Petabytes/s**

http://bit.ly/fugaku-report

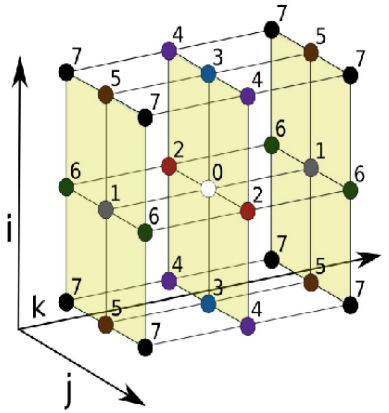
hpcg-benchmark.org

HPCG Results; The Other Benchmark

- High Performance Conjugate Gradients (HPCG).
- Solves *Ax=b, A* large, sparse, *b* known, *x* computed.
- An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs



- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collectives.
  - Multi-scale execution of kernels via MG (truncated) V cycle.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification (via spectral properties of PCG).

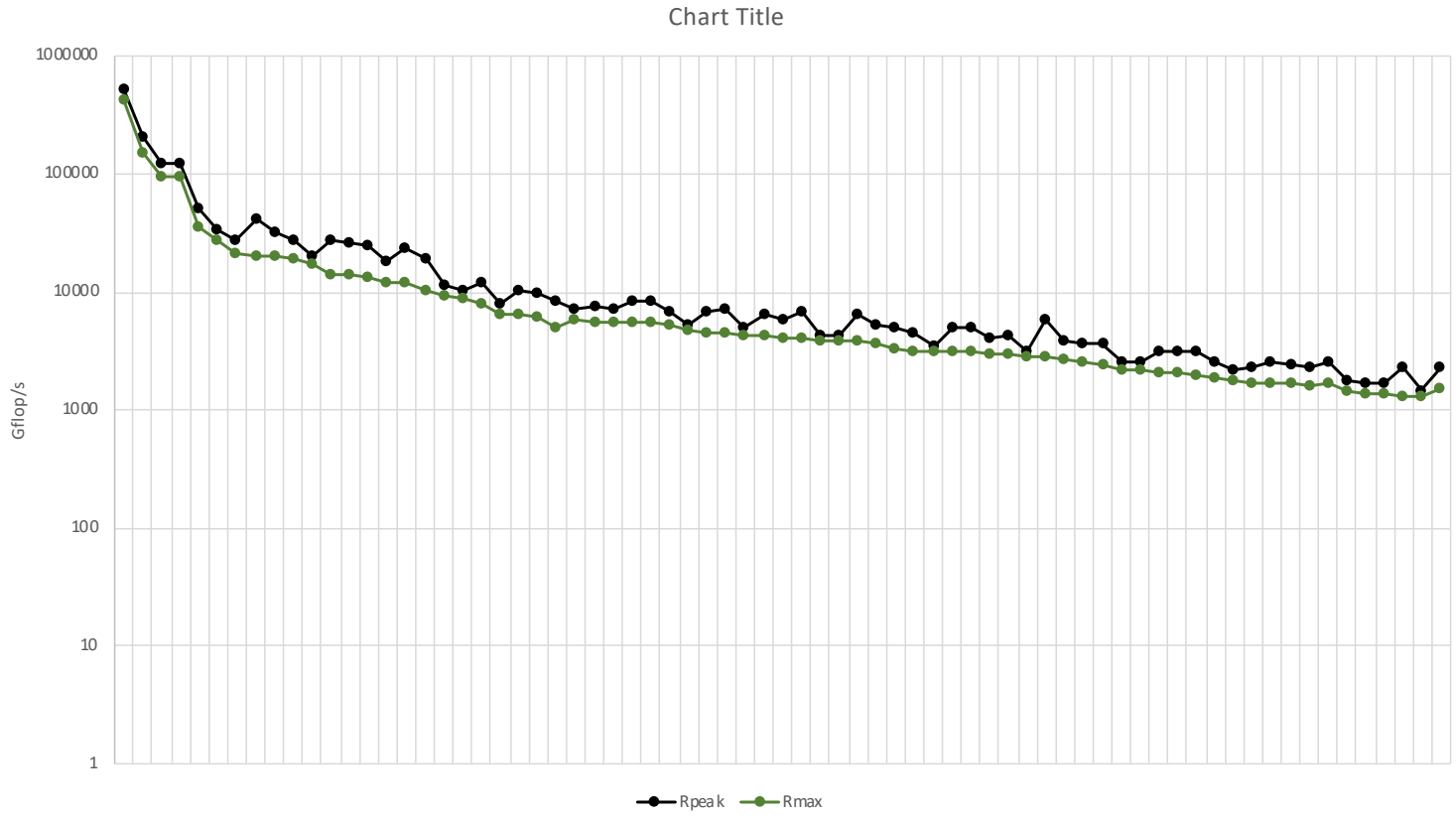


27-point stencil operator

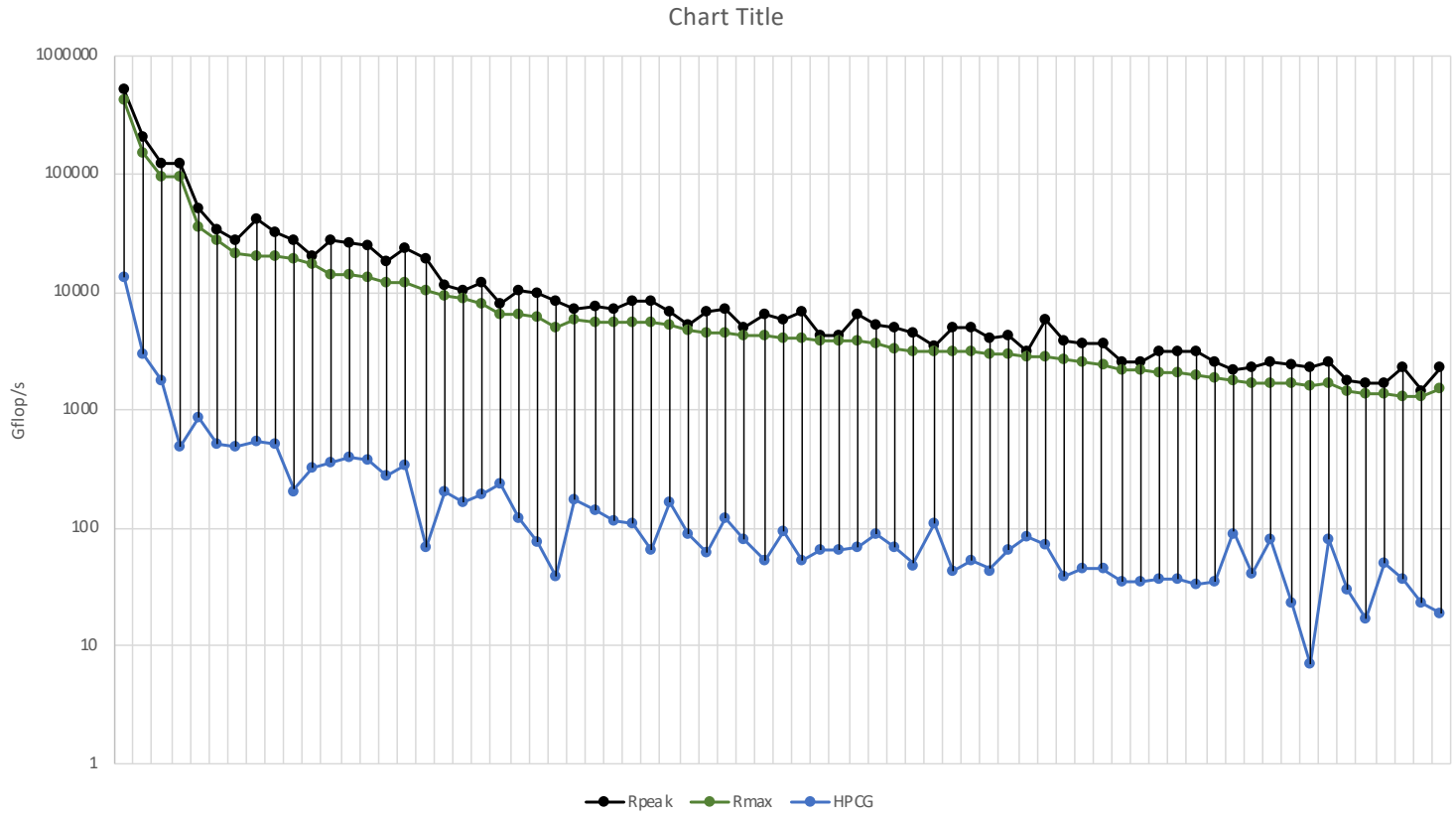
HPCG Benchmark June 2020

Rank	Site	Computer	Cores	HPL Rmax (Pflop/s)	TOP500 Rank	HPCG (Pflop/s)	Fraction of Peak
1	RIKEN Center for Computational Science Japan	**Fugaku**, Fujitsu A64FX, Tofu	7,299,072	415.53	1	13.4	2.5%
2	DOE/SC/ORNL USA	**Summit**, AC922, IBM POWER9 22C 3.7GHz, Dual-rail Mellanox FDR, NVIDIA Volta V100, IBM	2,414,592	143.50	2	2.92	1.5%
3	DOE/NNSA/LLNL USA	**Sierra**, S922LC, IBM POWER9 20C 3.1 GHz, Mellanox EDR, NVIDIA Volta V100, IBM	1,572,480	94.64	3	1.79	1.4%
4	Eni S.p.A. Italy	**HPC5**, PowerEdge, C4140, Xeon Gold 6252 24C 2.1 GHz, Mellanox HDR, NVIDIA Volta V100	669,760	35.45	6	0.860	2.4%
5	DOE/NNSA/LANL/SNL USA	**Trinity,** Cray XC40, Intel Xeon E5-2698 v3 16C 2.3GHz, Aries, Cray	979,072	20.16	11	0.546	1.3%
6	NVIDIA USA	**Selene**, DGX SuperPOD, AMD EPYC 7742 64C 2.25 GHz, Mellanox HDR, NVIDIA Ampere A100	277,760	27.58	7	0.5093	1.8%
7	Natl. Inst. Adv. Industrial Sci. and Tech. (AIST) Japan	**ABCI**, PRIMERGY CX2570M4, Intel Xeon Gold 6148 20C 2.4GHz, Infiniband EDR, NVIDIA Tesla V100, Fujitsu	391,680	16.86	12	0.5089	1.7%
8	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint**, Cray XC50, Intel Xeon E5-2690v3 12C 2.6GHz, Cray Aries, NVIDIA Tesla P100 16GB, Cray	387,872	19.88	10	0.497	1.8%
9	National Supercomputing Center in Wuxi China	**Sunway** TaihuLight, Sunway MPP, SW26010 260C 1.45GHz, Sunway, NRCPC	10,649,600	93.01	4	0.481	0.4%
10	Korea Institute of Science and Technology Information Republic of Korea	**Nurion**, CS500, Intel Xeon Phi 7250 68C 563584C 1.4GHz, Intel Omni-Path, Intel Xeon Phi 7250, Cray	570,020	13.93	18	0.391	1.5%

Comparison Peak, HPL, and HPCG: November 2020



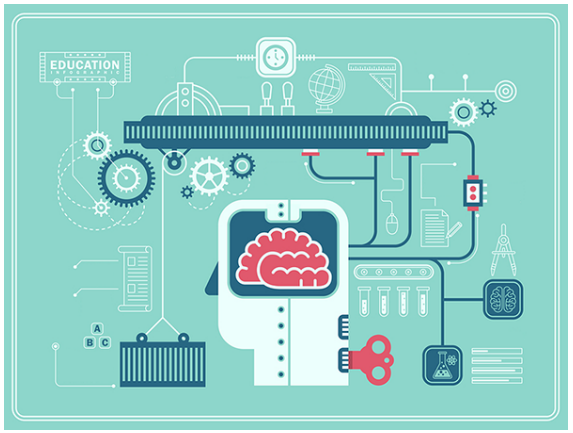
Comparison Peak, HPL, and HPCG: November 2020

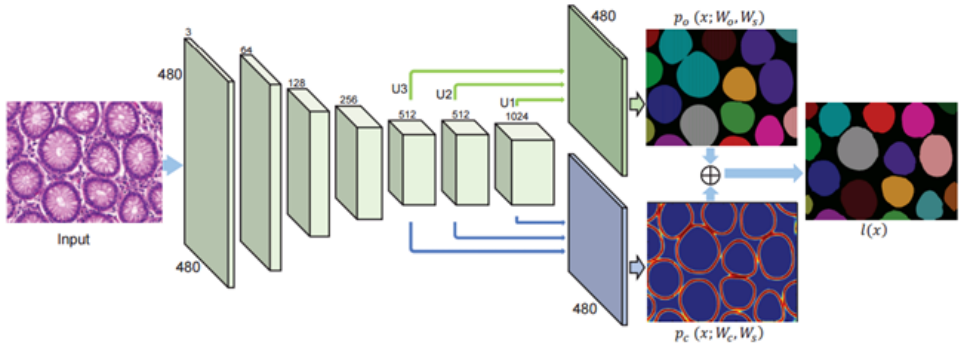


Machine Learning in Computational Science

Many fields are beginning to adopt machine learning to augment modeling and simulation methods

- Climate
- Biology
- Drug Design
- Epidemology
- Materials
- Cosmology
- High-Energy Physics



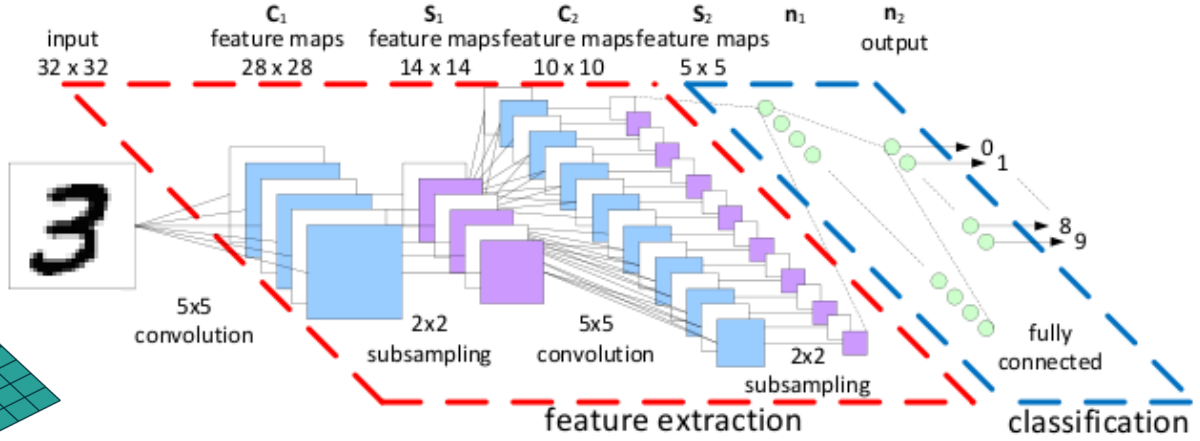


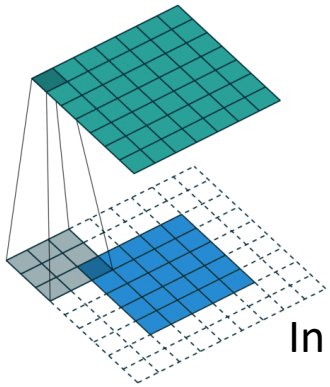
Deep Learning Needs Small Matrix Operations

Matrix Multiply is the time consuming part.

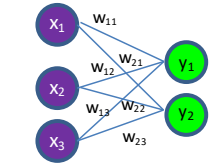
Convolution Layers and Fully Connected Layers require matrix multiply

There are many GEMM's of small matrices, perfectly parallel, can get by with 16-bit floating point





Convolution Step
In this case 3x3 GEMM



$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Fully Connected
Classification

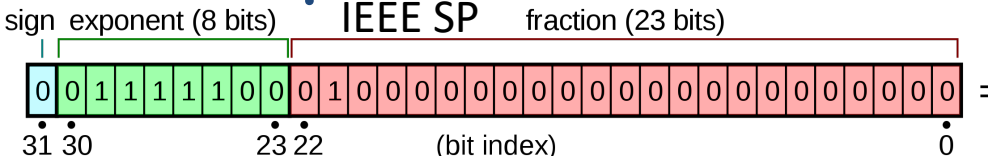
Mixed Precision

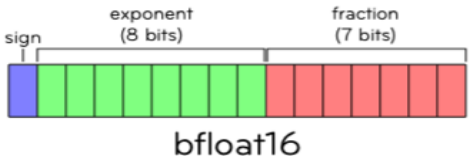
- Today many precisions to deal with (IEEE Standard)

Type	Size	Range	$u = 2^{-t}$
half	16 bits	$10^{\pm 5}$	$2^{-11} \approx 4.9 \times 10^{-4}$
single	32 bits	$10^{\pm 38}$	$2^{-24} \approx 6.0 \times 10^{-8}$
double	64 bits	$10^{\pm 308}$	$2^{-53} \approx 1.1 \times 10^{-16}$
quadruple	128 bits	$10^{\pm 4932}$	$2^{-113} \approx 9.6 \times 10^{-35}$

- Note the number range with half precision (16 bit fl.pt.)

largest fl pt number 65,504





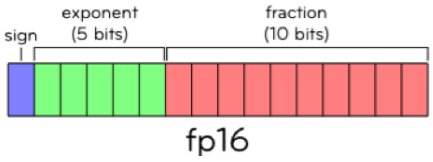
largest fl pt number $O(10^{38})$

Today's Floating-Point Precision Arithmetic

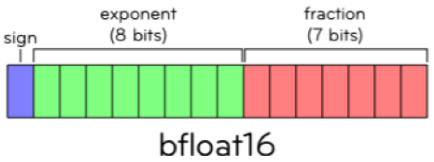
Type	Bits	Range	$u = 2^{-t}$
fp128 quad	128	$10^{\pm 4932}$	$2^{-113} \approx 1 \times 10^{-34}$
fp64 double	64	$10^{\pm 308}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32 single	32	$10^{\pm 38}$	$2^{-24} \approx 6 \times 10^{-8}$
fp16 half	16	$10^{\pm 5}$	$2^{-11} \approx 5 \times 10^{-4}$
bfloat16 half	16	$10^{\pm 38}$	$2^{-8} \approx 4 \times 10^{-3}$

Half precision increasingly supported by hardware

- Present: NVIDIA Pascal, Volta, and Ampere GPUs, AMD Radeon Instint M125 GPU, Google TPU, ARM NEON, and Fujitsu A64FX ARM
- Near furture: IBM AI chips, Intel Xeon Cooper Lake, and Intel Nervana Neural Network



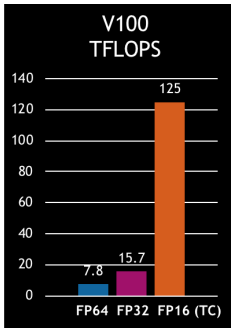
fp16



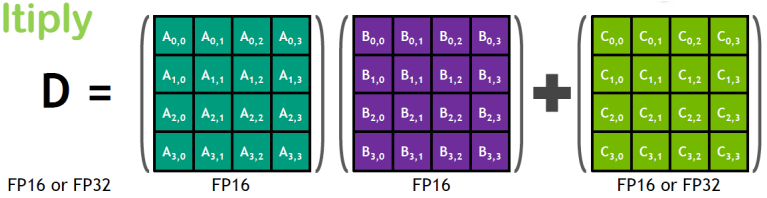
bfloat16

Nvidia Volta Peak Rates

- Four Performance levels for the different precision
  - 64 bit floating point (FMA): 7.5 Tflop/s peak
  - 32 bit floating point (FMA): 15 Tflop/s peak
  - 16 bit floating point (FMA): 30 Tflop/s peak
  - 16 bit floating point w/Tensor core: 120 Tflop/s peak

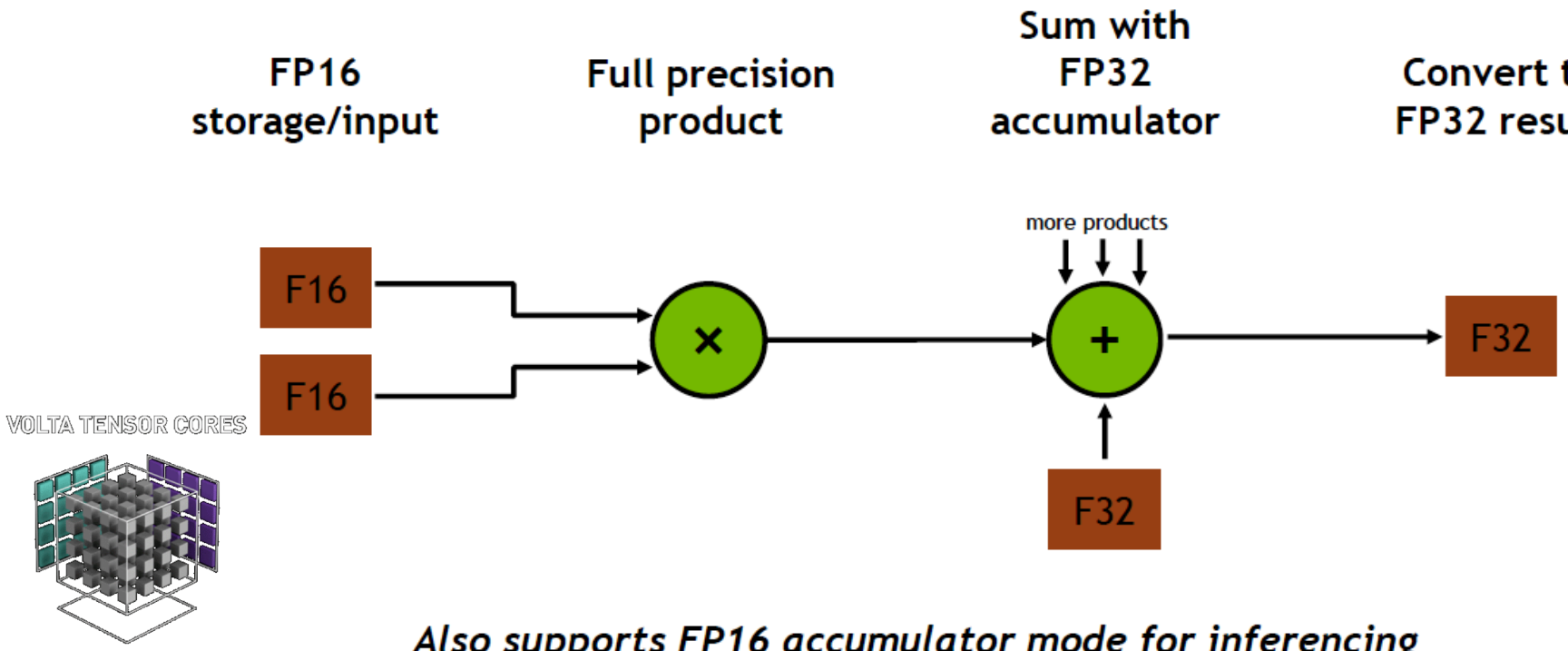


Tensor Core, special hardware for:
Mixed Precision Matrix Multiply
4x4 Matrices



D = AB + C

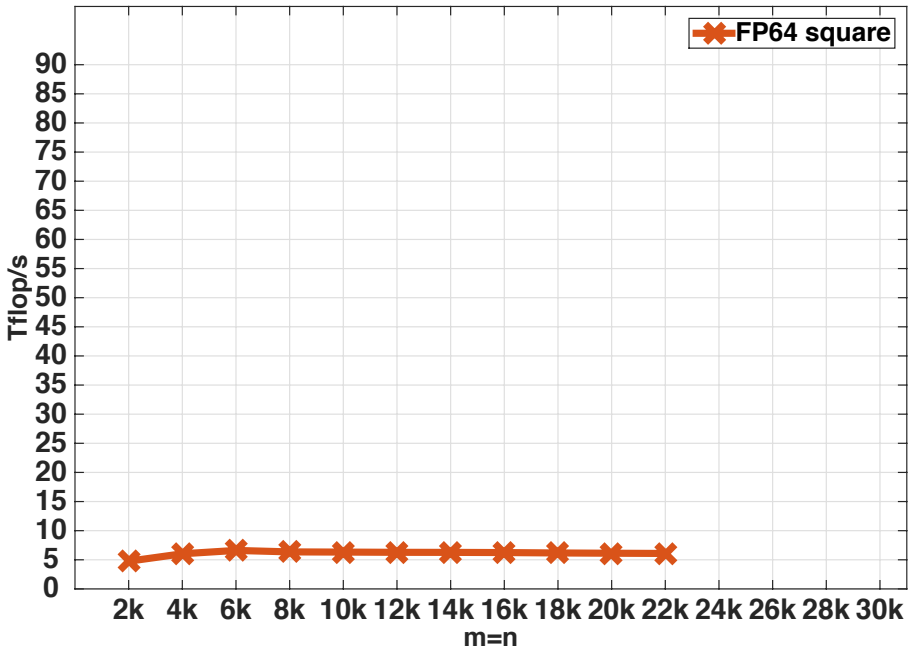
VOLTA TENSOR OPERATION



Also supports FP16 accumulator mode for inferencing

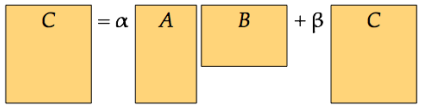
Leveraging Half Precision in HPC on V100

Study of the Matrix Matrix multiplication kernel on Nvidia V100



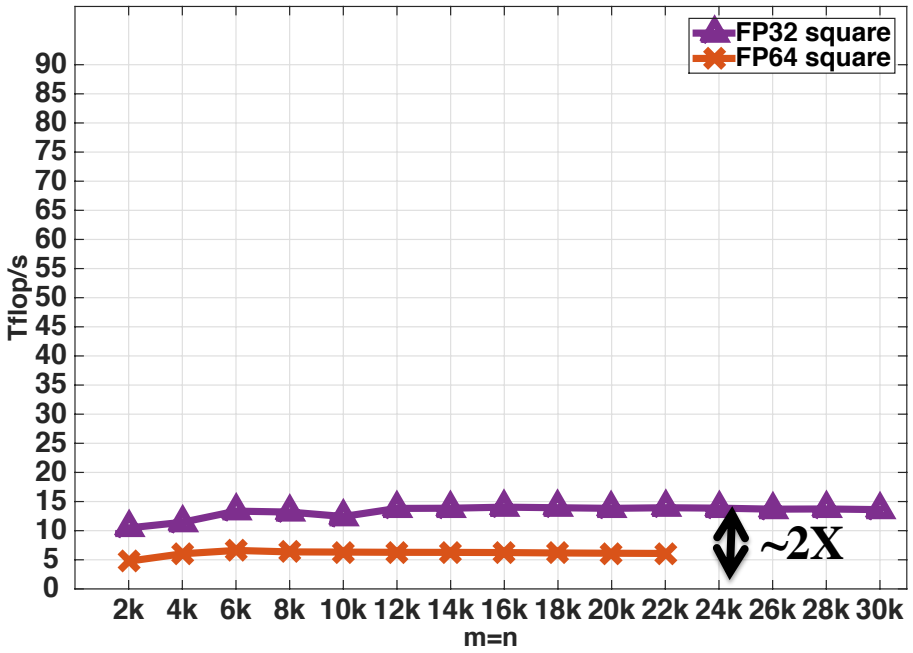
- dgemm achieve about 6.4 Tflop/s

Matrix matrix multiplication GEMM



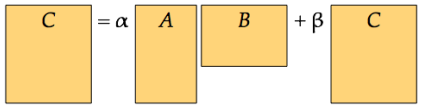
Leveraging Half Precision in HPC on V100

Study of the Matrix Matrix multiplication kernel on Nvidia V100



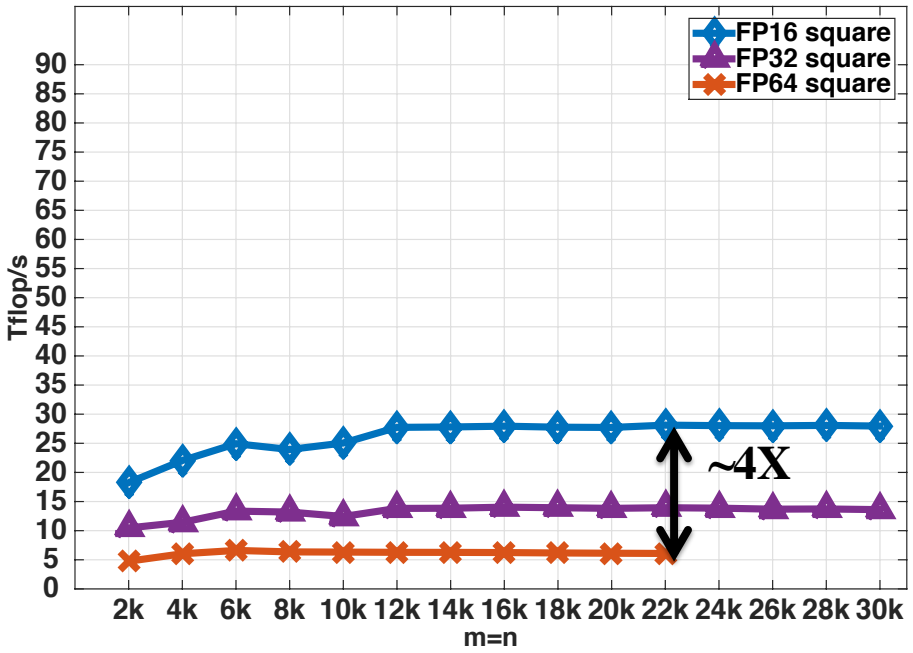
- dgemm achieve about 6.4 Tflop/s
- sgemm achieve about 14 Tflop/s

Matrix matrix multiplication GEMM



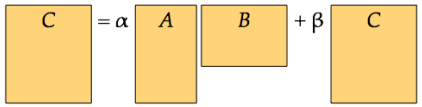
Leveraging Half Precision in HPC on V100

Study of the Matrix Matrix multiplication kernel on Nvidia V100



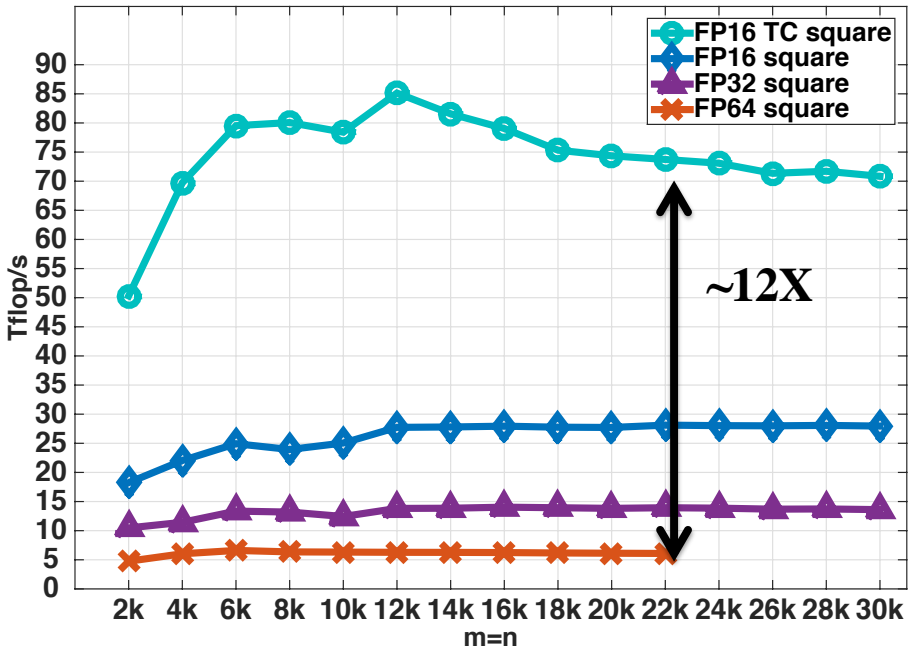
- dgemm achieve about 6.4 Tflop/s
- sgemm achieve about 14 Tflop/s
- hgemm achieve about 27 Tflop/s

Matrix matrix multiplication GEMM



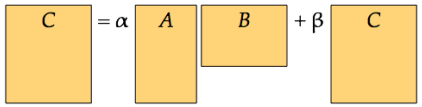
Leveraging Half Precision in HPC on V100

Study of the Matrix Matrix multiplication kernel on Nvidia V100



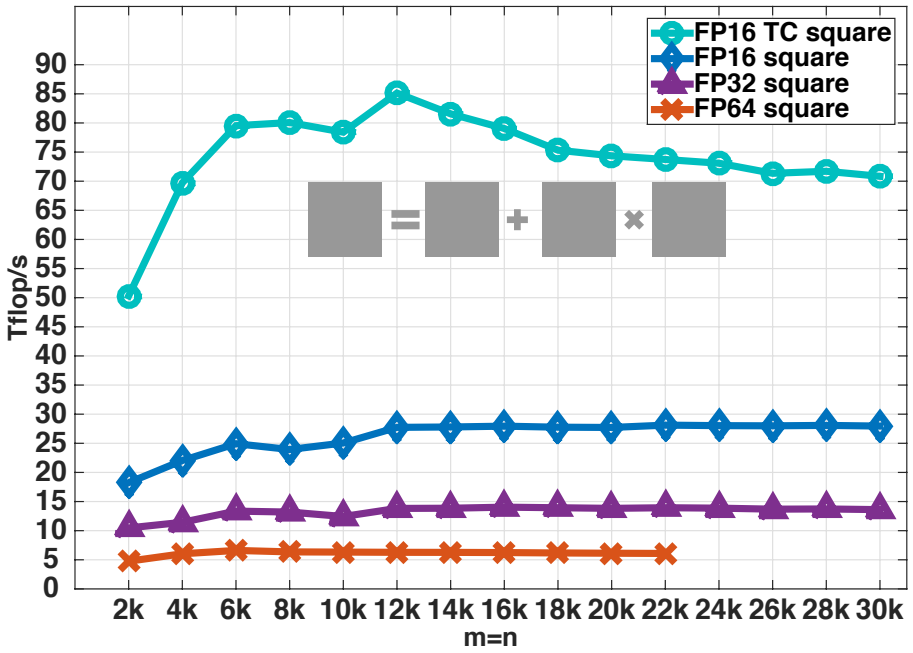
- dgemm achieve about 6.4 Tflop/s
- sgemm achieve about 14 Tflop/s
- hgemm achieve about 27 Tflop/s
- Tensor cores gemm reach about 85 Tflop/s

Matrix matrix multiplication GEMM



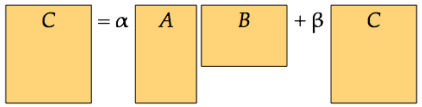
Leveraging Half Precision in HPC on V100

Study of the Matrix Matrix multiplication kernel on Nvidia V100



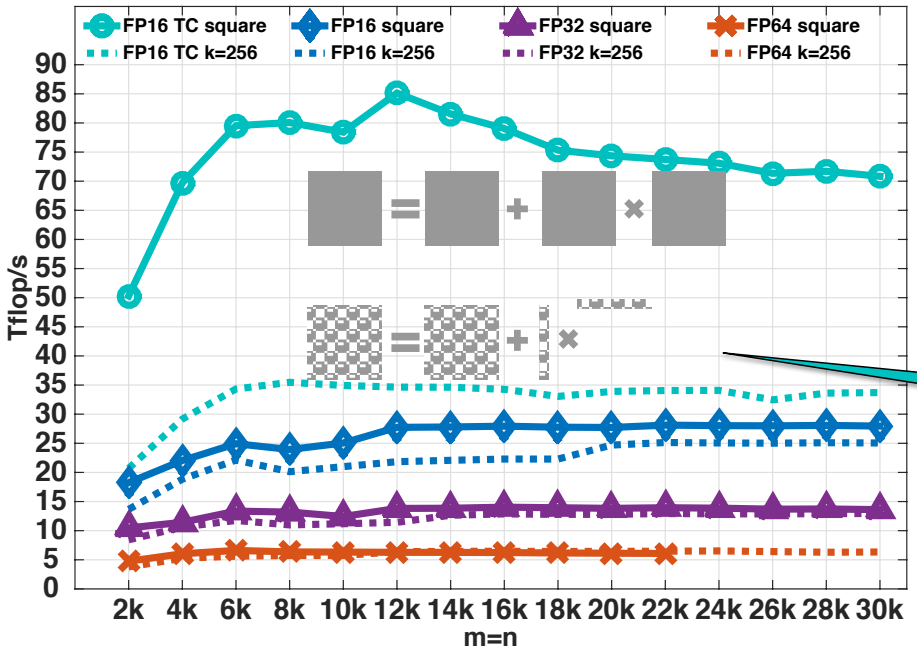
- dgemm achieve about 6.4 Tflop/s
- sgemm achieve about 14 Tflop/s
- hgemm achieve about 27 Tflop/s
- Tensor cores gemm reach about 85 Tflop/s

Matrix matrix multiplication GEMM



Leveraging Half Precision in HPC on V100

Study of the rank k update used by the LU factorization algorithm on Nvidia V100

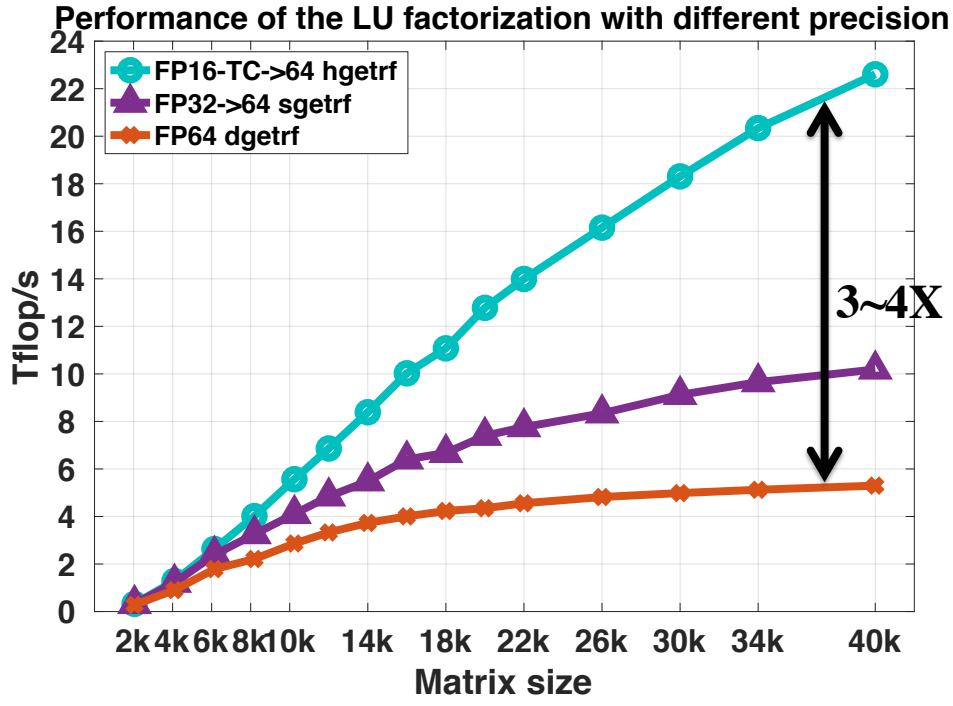


- In LU factorization need matrix multiple but operations is a rank-k update computing the Schur complement

Rank-k GEMM needed by LU does not perform as well as square but still OK

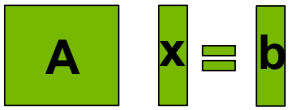
Leveraging Half Precision in HPC on V100

Study of the LU factorization algorithm on Nvidia V100



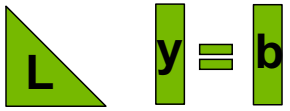
- LU factorization is used to solve a linear system Ax=b

A x = b



LUx = b

Ly = b



then
Ux = y

For the LU, half precision used only in GEMM, Panel and TRSM in SP.

Leveraging Half Precision in HPC on V100

Use Mixed Precision algorithms

- Achieve higher performance
  - → faster time to solution (benefit from operations and data movement)
- Reduce power consumption by decreasing the execution time
  - →Energy Savings !!!

– Reformulate to find correction to solution, rather than solution; Δx rather than x.

A. Haidar, P. Wu, S. Tomov, J. Dongarra,
Investigating Half Precision Arithmetic to Accelerate Dense Linear System Solvers,
SC-17, ScalA17: 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, ACM, Denver, Colorado, November 12-17, 2017.

A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham,
Harnessing GPU Tensor Cores for Fast FP16 Arithmetic to Speed up Mixed-Precision Iterative Refinement Solvers, SC-18, Dallas, IEEE.

Leveraging Half Precision in HPC on V100

Idea: use low precision to compute the expensive flops (LU $O(n^3)$) and then iteratively refine ($O(n^2)$) the solution in order to achieve the FP64 arithmetic

Iterative refinement for dense systems, *Ax = b*, can work this way.

L U = lu(A)		lower precision	$O(n^3)$
x = U\(L\b)		lower precision	$O(n^2)$
r = b – Ax (with original A)		FP64 precision	$O(n^2)$
WHILE || r || not small enough			
1. find a correction "z" to adjust x that satisfy Az=r solving Az=r could be done by either:			
➢ z = U\(L\r)	Classical Iterative Refinement	lower precision	$O(n^2)$
➢ GMRes preconditioned by the LU to solve Az=r	Iterative Refinement using GMRes	lower precision	$O(n^2)$
2. x = x + z		FP64 precision	$O(n^1)$
3. r = b – Ax (with original A)		FP64 precision	$O(n^2)$
END			

Higham and Carson showed can solve the inner problem with iterative method and not infect the solution.

- Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.
- It can be shown that using this approach we can compute the solution to 64-bit floating point precision.
- Need the original matrix to compute residual (r) and matrix cannot be too badly conditioned

E. Carson & N. Higham, "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions *SIAM J. Sci. Comput.*, 40(2), A817–A847.

Leveraging Half Precision in HPC on V100

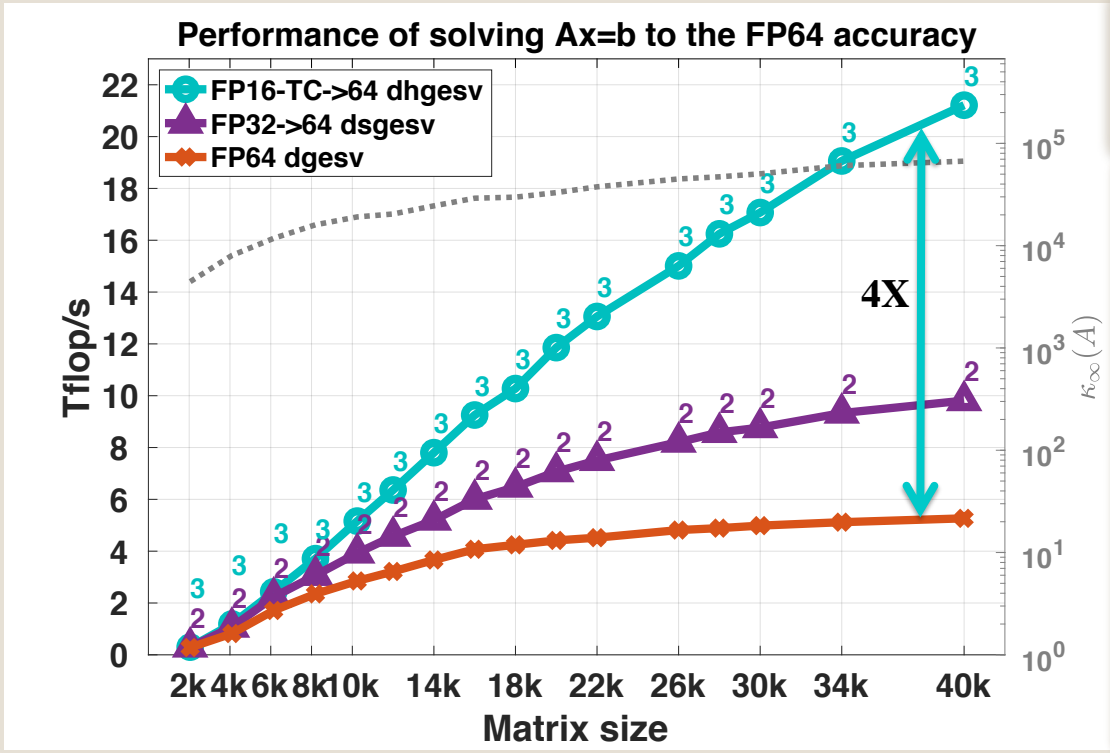
Idea: use low precision to compute the expensive flops (LU $O(n^3)$) and then iteratively refine ($O(n^2)$) the solution in order to achieve the FP64 arithmetic

Iterative refinement for dense systems, $Ax = b$, can work this way.

L U = lu(A)	lower precision	$O(n^3)$
x = U\(L\b)	lower precision	$O(n^2)$
GMRes preconditioned by the LU to solve Ax=b	FP64 precision	$O(n^2)$

- Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.
- It can be shown that using this approach we can compute the solution to 64-bit floating point precision.
- Need the original matrix to compute residual (r) and matrix cannot be too badly conditioned

Tensor Core Accelerated IRS solving linear system Ax = b Performance Behavior



Flops = $2n^3/(3$ time)
meaning twice higher is twice faster

- solving Ax = b using FP64 LU
- solving Ax = b using FP32 LU and iterative refinement to achieve FP64 accuracy
- solving Ax = b using FP16 Tensor Cores LU and iterative refinement to achieve FP64 accuracy

Results obtained using CUDA 10.2 and GV100 GPU.

Problem generated with an arithmetic distribution of the singular values $\sigma_i = 1 - (\frac{i-1}{n-1})(1 - \frac{1}{cond})$, and positive eigenvalues.

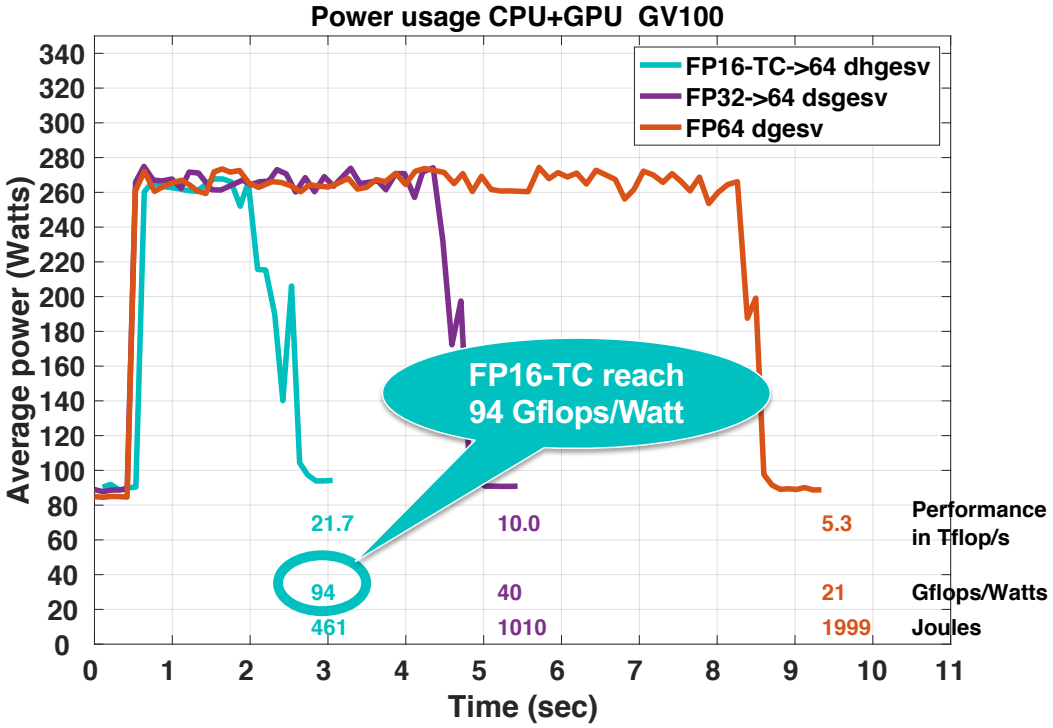
Leveraging Half Precision in HPC on V100

Use Mixed Precision algorithms

Idea: use lower precision to compute the expensive flops (LU $O(n^3)$) and then iteratively refine the solution in order to achieve the FP64 arithmetic

- Achieve higher performance → faster time to solution
- Reduce power consumption by decreasing the execution time → Energy Savings !!!

Tensor Core Accelerated IRS solving linear system Ax = b



Energy Efficiency

Mixed precision techniques can provide a large gain in energy efficiency

- Power consumption for a matrix of size 40K
- The **FP64 algorithm** achieve 5.3 Tflop/s providing about **22 Gflops/Watts**.
- The **FP32→64 algorithm** achieve 10 Tflop/s providing about **45 Gflops/Watts**.
- The **FP16→64 TC algorithm using Tensor Cores** achieve 22 Tflop/s providing about **94 Gflops/Watts**.

Results obtained using CUDA 10.2 and GV100 GPU.

Problem generated with an arithmetic distribution of the singular values $\sigma_i = 1 - (\frac{i-1}{n-1})(1 - \frac{1}{cond})$ and positive eigenvalues.

HPL-AI (A MIXED PRECISION BENCHMARK)

The HPL-AI benchmark seeks to highlight the emerging convergence of high-performance computing (HPC) and artificial intelligence (AI) workloads and highlight the advantages of mixed precision.

The benchmark is a combination of LU factorization (at lower precision) and iterative refinement method (like GMRes) to bring the solution back to 64-bit accuracy.

Iterative refinement for dense systems, *Ax = b*, can work this way.

L U = lu(*A*)	**lower precision**	$O(n^3)$
x = U\(L\b)	**lower precision**	$O(n^2)$
GMRes preconditioned by the LU to solve Ax=b	**FP64 precision**	$O(n^2)$

http://bit.ly/hpl-ai

Recent Results Run at Scale…

- Mixed precision iterative refinement approach solved

Rank	Site	Computer	Cores	HPL Rmax (Eflop/s)	TOP500 Rank	HPL-AI (Eflop/s)	Speedup
1	RIKEN Center for Computational Science Japan	**Fugaku**, Fujitsu A64FX, Tofu D	7,299,072	0.416	1	**1.42**	3.42x
2	DOE/SC/ORNL USA	**Summit**, AC922 IBM POWER9, IB Dual-rail FDR, NVIDIA Volta V100	2,414,592	0.144	2	**0.55**	3.83x

- Same accuracy compared to full 64 bit precision

Conclusion:

- We accelerated the solution of linear system **Ax = b** solver using hardware-accelerated FP16 arithmetic on GPUs;
- We introduced a framework for exploiting mixed-precision FP16-FP32/FP64 iterative refinement solvers and describe the path to draw high-performance and energy-aware GPU implementations;
  - Ideas can be applied to other 1 sided reductions (LU, LL^T, LDL^T, QR) and also for 2 sided in the case of eigen (singular) values/vectors,(where are few are required).
- Our technique shows that a number of problems can be **accelerated** up to **4X** by the usage of the **FP16-TC** or **2X** using the **FP32** arithmetic.
- We studied the energy-efficiency of our approach that showed significant energy savings, **5X energy savings** using the **FP16-TC** compared to the FP64 implementation.
- We illustrated a technique to use **V100 Tensor Cores FP16-TC** that achieves FP64 accuracy at a highly efficient/accelerated performance equating to **74 Gflops/Watt** and **24 Tflops/s**.
- There is a rigorous error analysis to support everything

Thanks to Collaborators

- Erich Strohmaier, LBNL
- Horst Simon, LBNL
- Azzam Haidar, Nvidia
- Nick Higham, U of Manchester
- Stan Tomov, UTK