New communication-avoiding algorithms, and fixing old “bugs” in the BLAS and LAPACK

Jim Demmel, EECS & Math Depts., UC Berkeley

And many, many others …

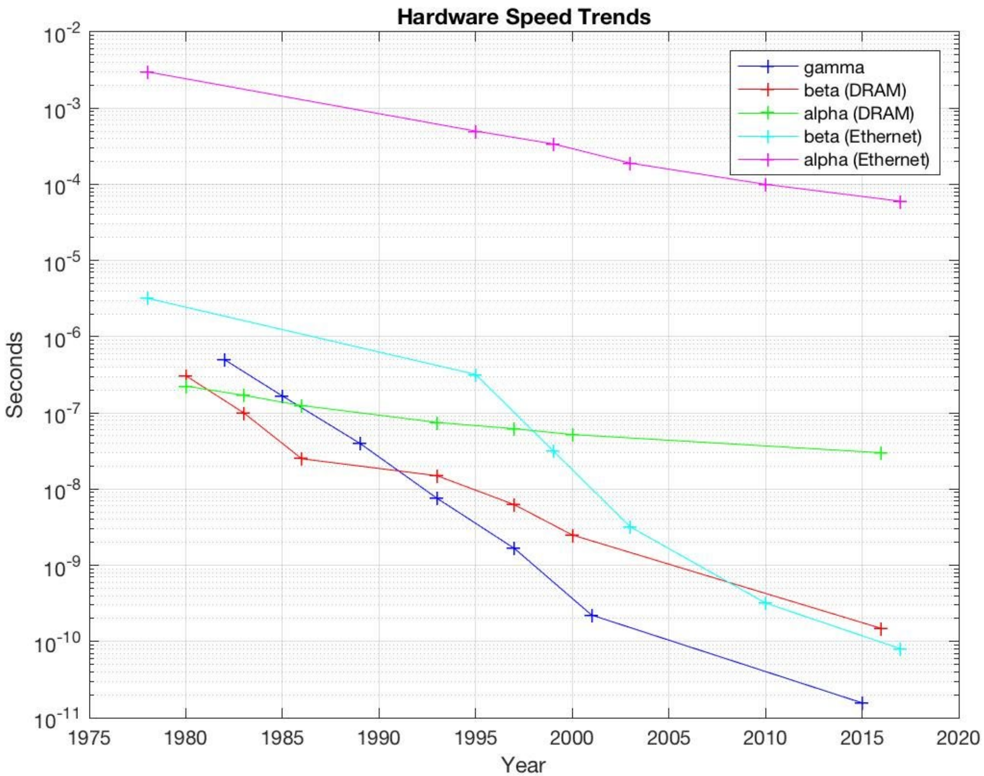
Outline

- Communication-Avoiding Algorithms
  - What is communication, and why we want to avoid it
  - Examples of past algorithms (linear algebra, ML, …)
  - Optimal tiling for mixed precision matmul
  - Optimal tiling for Dense * random (eg Gaussian, ± 1, …)
- Fixing old ”bugs” in the BLAS and LAPACK, i.e. making them resilient to exceptions
  - Why better exception handling is increasingly important
  - Examples of problems: inconsistent answers, car crashes,…
  - Tentative plan to fix these problems (comments welcome!)

Why avoid communication?

- Running time of an algorithm is sum of 3 terms:
  - # flops * time_per_flop
  - # words moved / bandwidth
  - # messages * latency

  (the last two terms: communication)
- Time_per_flop (γ) << 1/ bandwidth (β) << latency (α)



Same story for saving energy

Patterson & Hennessey, 2019

Sample Speedups

- Doing same operations, just in a different order
  - Up to **12x** faster for 2.5D dense matmul on 64K core IBM BG/P
  - Up to **100x** faster for 1.5D sparse-dense matmul on 1536 core Cray XC30
  - Up to **6.2x** faster for 2.5D All-Pairs-Shortest-Path on 24K core Cray XE6
  - Up to **11.8x** faster for direct N-body on 32K core IBM BG/P
- Mathematically identical answer, but different algorithm
  - Up to **13x** faster for Tall Skinny QR on Tesla C2050 Fermi NVIDIA GPU
  - Up to **6.7x** faster for symeig(band A) on 10 core Intel Westmere
  - Up to **4.2x** faster for BiCGStab (MiniGMG bottom solver) on 24K core Cray XE6
  - Up to **5.1x** faster for coordinate descent LASSO on 3K core Cray XC30
- Different algorithm, different approximate answer
  - Up to **16x** faster for SVM on a 1536 core Cray XC30
  - Up to **135x** faster for ImageNet training on 2K Intel KNL nodes

Sample Speedups

- Doing same operations, just in a different order

Ideas adopted by Nervana, "deep learning" startup, acquired by Intel in August 2016

Kwasniewski, Hoefler, et al (Best Student Paper, SC'19)

- Mathematically identical answer, but different algorithm

SIAG on Supercomputing Best Paper Prize, 2016
(D., Grigori, Hoemmen, Langou)

Released in LAPACK 3.7, Dec 2016

Latest Release: Householder Reconstruction (Kozachenko, D.)

- Different algorithm, different approximate answer

IPDPS 2015 Best Paper Prize (You, D. Czechowski, Song, Vuduc)

ICPP 2018 Best Paper Prize (You, Zhang, Hsieh, D., Keutzer)

2019: Idea (LARS) adopted by industry standard benchmark MLPerf

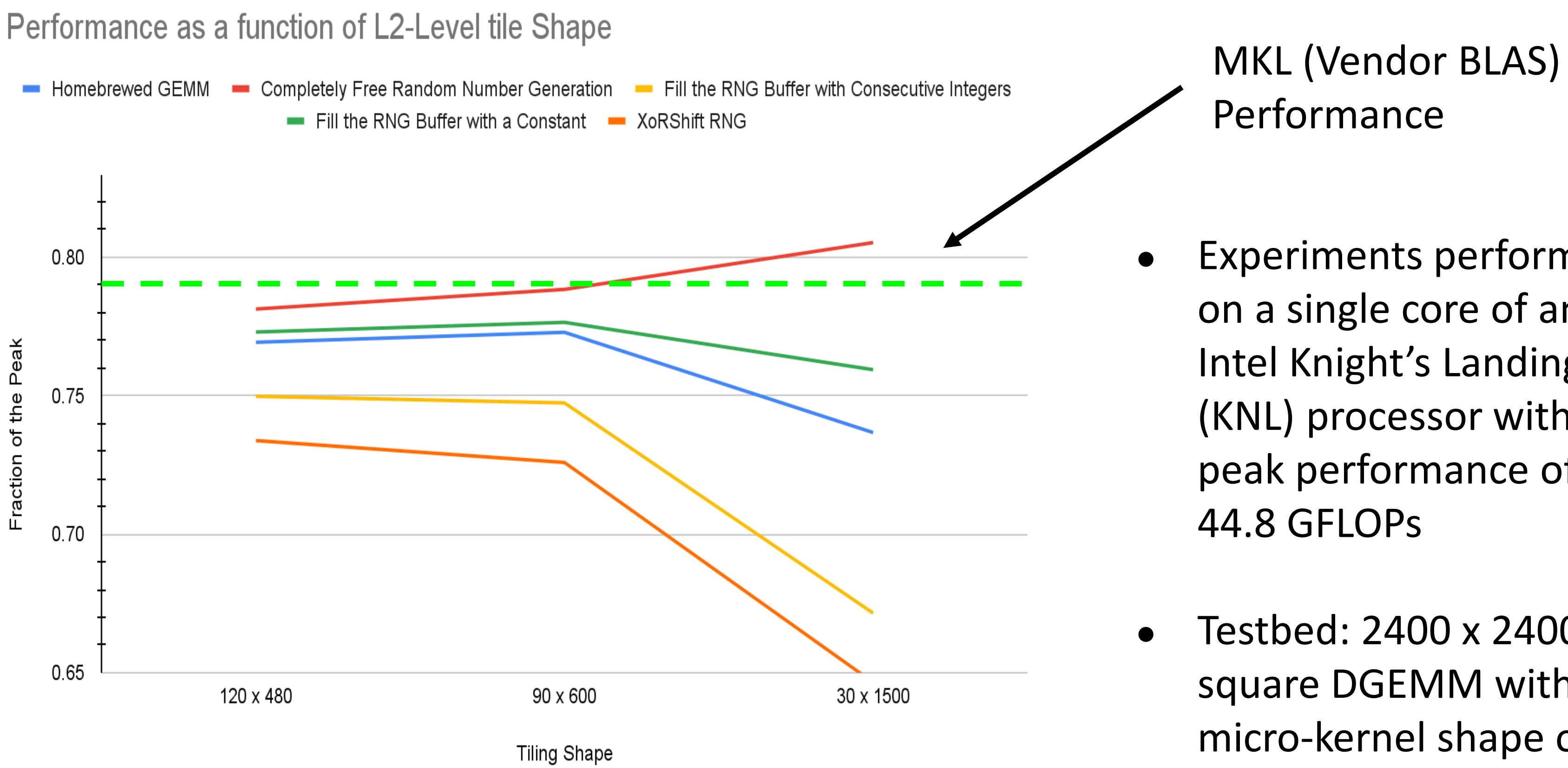
Optimal mixed precision matmul

- Suppose in C=A*B, each entry of B, C occupies 1 "word", A occupies $\rho \leq 1$, what is optimal tiling?
- Still use Loomis-Whitney (M = cache size)
  - Break execution into "segments" of x loads/stores
  - #iterations/segment $\leq$ (#A*#B*#C)$^{1/2}$
  - ρ#A+#B+#C $\leq$ x+M bounds data available in segment
  - #iterations/segment $\leq \ \rho^{-1/2}(\frac{x+M}{3})^{3/2}$
  - #words_moved $\geq \ \rho^{\frac{1}{2}} * 2 * \#iterations/M^{1/2}$
- Optimal tiling:
  - A: $(\frac{M}{\rho})^{1/2}$ x $(\frac{M}{\rho})^{1/2}$, B and C: $(\frac{M}{\rho})^{1/2}$ x $(M * \rho)^{1/2}$

Optimal random*dense matmul

- Suppose each entry of A is a random number that costs $\rho \leq 1$ to recompute, what is optimal tiling?
  - ρ = cost to recompute A(i,j) / cost to load 1 word from memory
- Similar analysis, tiling for A being lower precision
- Tiling depends on cost of random number
  - Eg Rademacher < Gaussian

Performance Impact of Varying Tile Shape



- Experiments performed on a single core of an Intel Knight’s Landing (KNL) processor with a peak performance of 44.8 GFLOPs
- Testbed: 2400 x 2400 square DGEMM with a micro-kernel shape of 30 x 8, varying tiling to minimize DRAM -> L2 memory movement

*Micro-kernel shape depends on the number of SIMD registers on KNL (32 of them; we use 30 to accumulate matrix C, 1 for matrix B, and 1 scratch register).
KNL DGEMM algorithm implemented based on https://doi.org/10.1007/s10586-018-2810-y.

Making BLAS, LAPACK more resilient to numerical exceptions

- 1/0, 0/0, sqrt(-1), …can cause problems:
  - Crash of Ariane 5 rocket
  - Naval propulsion failure
  - Crash in a robotic car race:

Reddit post by engineer in charge of control system:

“During this initialization lap something happened which apparently cause the steering control signal to go to NaN”

“Bug” 1/3 in BLAS: IxAMAX

- IxAMAX returns index of first entry of largest “absolute value”
- ISAMAX:
  - ISAMAX([0,NaN,2]) = 3 and ISAMAX([NaN,0,2]) = 1
  - NaNs do not propagate consistently
- ICAMAX
  - OV = overflow threshold
  - ICAMAX([OV + i*OV, Inf + i*0]) = 1
  - ICAMAX points to finite entry instead of Inf

"Bug" 2/3 in BLAS: GER and SYR

- GER computes $A = A + \alpha x y^T$
- GER checks if $y(i) = 0$, does not multiply by it
  - Inf/NaN in x does not propagate to column i of A
  - If all $y(i) = 0$, no Infs/NaNs in x propagate
  - No checking for zeros in x
- SYR computes $A = A + \alpha x x^T$ when $A = A^T$
  - Can update upper or lower triangle of A
  - Code only checks for 0 in x^T, so can get different answer for upper and lower triangle

“Bug” 3/3 in BLAS: TRSV

- TRSV solves $T * x = b$ or $T^T * x = b$
- TRSV checks for zeros in x like GER and SYR
- Ex: $T = \begin{vmatrix} 1 & NaN & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}, b = \begin{vmatrix} 2 \\ 1 \\ 1 \end{vmatrix}$ yields $x = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix}$
- NaN does not propagate
- Solving $(T^T)^T * x = b$ does not check for zeros, so NaN does propagate
- BLAS Bugs 1,2 and 3 combine so that SGESV does not propagate NaNs

Future Work

- Detailed plan under construction to identify, fix these “bugs”
- Will automatically check for Inf and NaN inputs on most drivers (that already compute norms), as in LAPACKE
- Possibility: Provide “wrappers” to allow more extensive checks for Infs and NaNs if requested

A few of the many collaborators

- Vivek Bharadwaj
- Jack Dongarra (happy birthday!), Mark Gates, Greg Henry, Igor Kozachenko, Julie Langou, Julien Langou, Xiaoye Li, Piotr Luszczek, Michael Mahoney, Riley Murray, Jason Riedy, Weslley Pereira, Peter Tang, …

- Twitter post, including video of robo-car crash:

https://twitter.com/dogryan100/status/1321800383505657856?s=21

Extra slides

”Bug” in SGESV

- Assume version that calls GER to update Schur complement, not newer recursive version that uses GEMM
- Solve $\begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$ * x = $\begin{vmatrix} 0 \\ 1 \end{vmatrix}$
- ISAMAX chooses 1 as pivot, not NaN
- GER updates 2 – NaN*0 = 2, NaN does not propagate
- TRSV does not multiply by 0 in x, NaN does not propagate, get x = [0; .5]