Tensor Core Acceleration of the CoMet application

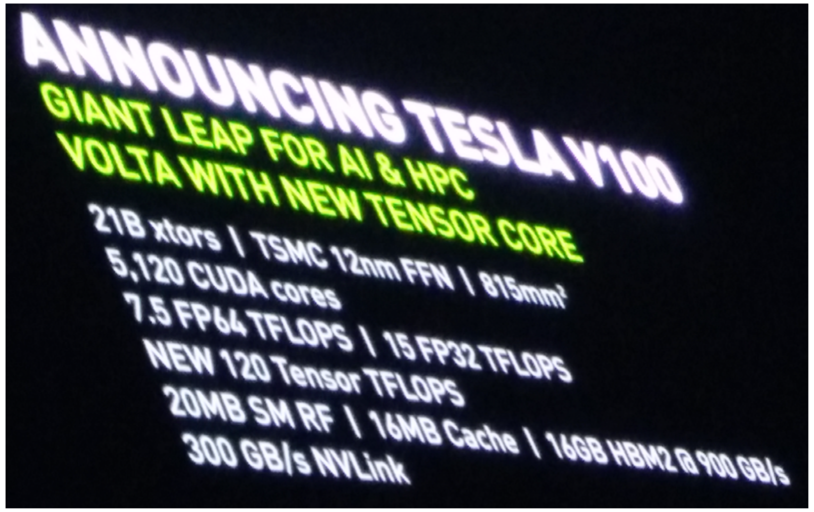
Wayne Joubert

Scientific Computing Group
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory

7th ADAC Workshop
March 25-26, 2019

Tensor Core Acceleration of Applications

- Tensor Cores as an “accelerator within an accelerator”
- On-package reduced precision hardware units
- Developed to address exploding computational needs in deep learning, data analytics
- Tensor Cores were announced NVIDIA GTC, May 11, 2017



ANNOUNCING TESLA V100
GIANT LEAP FOR AI & HPC
VOLTA WITH NEW TENSOR CORE
21B xtors | TSMC 12nm FFN | 815mm^2
5,120 CUDA cores
7.5 FP64 TFLOPS | 15 FP32 TFLOPS
NEW 120 Tensor TFLOPS
20MB SM RF | 16MB Cache | 16GB HBM2 @ 900 GB/s
300 GB/s NVLink

- Half precision matrix multiplies **16X faster** than double precision
- Other vendors also pursuing reduced precision – Google TPU; ~ 50 startups developing custom DL processors, reduced precision as low as 1-bit
- Represents a trend of growth in heterogeneity of processors and compute nodes
- On Summit the Tensor Cores already being used by applications – for example, at least 4 out of 6 Gordon Bell Finalist teams last November used the Tensor Cores in some fashion
- This talk will describe one of these codes, the CoMet computational genomics application

CoMet Solves a “Needle in a Haystack” Problem

https://commons.wikimedia.org/wiki/File:DNA_com_GGN.jpg

- Its purpose is to find the genetic causes of phenotypic traits such as the susceptibility to a given disease
- Traits like this can result from a complex interaction of elements in the genomic data
- It may be unknown a priori which of the millions of genomic features cause the traits and how these features interact
- The results is a huge combinatorial explosion of potential interactions to search through
- Mathematically, this problem can be formulated as an all-to-all comparison of vectors: we seek all pairs (or triples) of vectors that have a similarity property
- Formally, we have *n* vectors, of length *m,* interacting *k* vectors at a time (*k* = *2, 3, …*) representing pairwise (k=2), 3-way, or higher order interactions
- The relevant methods have computational complexity $O(n^k m)$
- Because this is an all-to-all computation, any solution method will necessarily require heavy communication and memory traffic

How to Solve?

- This is an $O(n^k m)$ computation on $O(nm)$ inputs (vectors) and $O(n^k)$ results (the result is a tensor)
- There is much more computation than data – suggesting that high computational intensity might be possible
- In fact, for k=2, vector similarity methods are *structurally identical* to DGEMM dense matrix-matrix products – in fact, cosine vector similarity (inner products) *is* a DGEMM
- Likewise, higher order methods (k > 2) are structurally identical to tensor contractions
- Because of this relationship, we can apply dense linear algebra methods and software to solve these problems

Specific Methods: PS and CCC

method	**Proportional Similarity (PS) method**	**Custom Correlation Coefficient (CCC)**
inputs	real-valued inputs	2-bit allele values
2-way	$c_2(u,v) = 2\left[\sum_q \boxed{min(u_q, v_q)}\right] / \left[\sum_q (u_q + v_q)\right]$	$\{v_i\}, \quad v_{i,q} \in \{0,1\} \times \{0,1\}, \quad a,b,c \in \{0,1\}$ $\rho_{i,q}(a) = \sum_r \chi_a((v_{i,q})_r), \quad f_i(a) = \frac{1}{2m} \sum_{q=1}^{m} \rho_{i,q}(a)$ $\rho_{i,j,q}(a,b) = \rho_{i,q}(a) \cdot \rho_{j,q}(b), \quad f_{i,j}(a,b) = \frac{1}{4m} \boxed{\sum_{q=1}^{m} \rho_{i,j,q}(a,b)}$ $CCC_{i,j}(a,b) = f_{i,j}(a,b)(1 - \gamma f_i(a))(1 - \gamma f_j(b))$
3-way	$c_3(u,v,w) = (3/2)\left[\sum_q min(u_q, v_q) + min(u_q, w_q) + min(v_q, w_q) - \boxed{min(u_q, v_q, w_q)}\right] / \sum_q (u_q + v_q + w_q).$	$\rho_{i,j,k,q}(a,b,c) = \rho_{i,q}(a) \cdot \rho_{j,q}(b) \cdot \rho_{k,q}(c), \quad f_{i,j,k}(a,b,c) = \frac{1}{8m} \boxed{\sum_{q=1}^{m} \rho_{i,j,k,q}(a,b,c)}$ $CCC_{i,j,k}(a,b,c) = f_{i,j,k}(a,b,c)(1 - \gamma f_i(a))(1 - \gamma f_j(b))(1 - \gamma f_k(c)).$

Scalar minimum of values

Counting number of occurrences of bit combinations

How to Map to Accelerated Processors (2-way case)

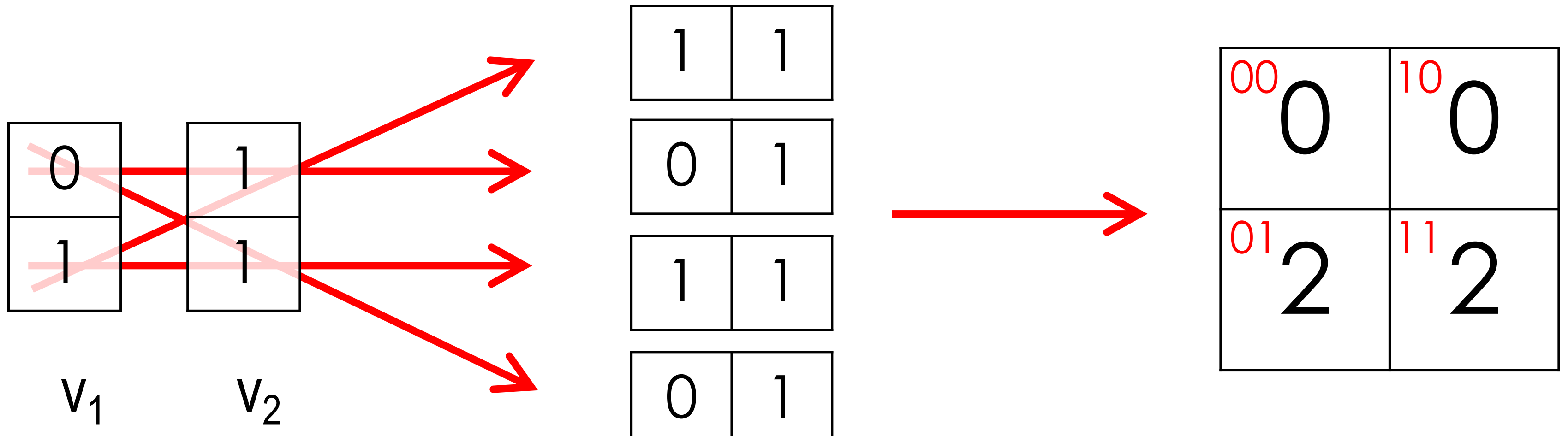
- <u>PS method</u>
  - MAGMA SGEMM, DGEMM kernels: replace <u>`c += a*b`</u> with <u>`c += min(a,b)`</u>
  - Use CUDA intrinsics **`fmin`**, **`fminf`** for speed

- <u>CCC, method 1 (bitwise calculation method)</u>
  - MAGMA ZGEMM: replace <u>`c += a*b`</u> with 64-bit AND, OR, NOT operations followed by binary popcount instructions
  - Use CUDA intrinsic **`__popcll`** for speed

- <u>CCC, method 2 (Tensor Core calculation method)</u>
  - Modify to use half precision GEMM (see following slides)

2-way CCC Computation Example

A counting problem: count number of occurrences of certain bit combinations:



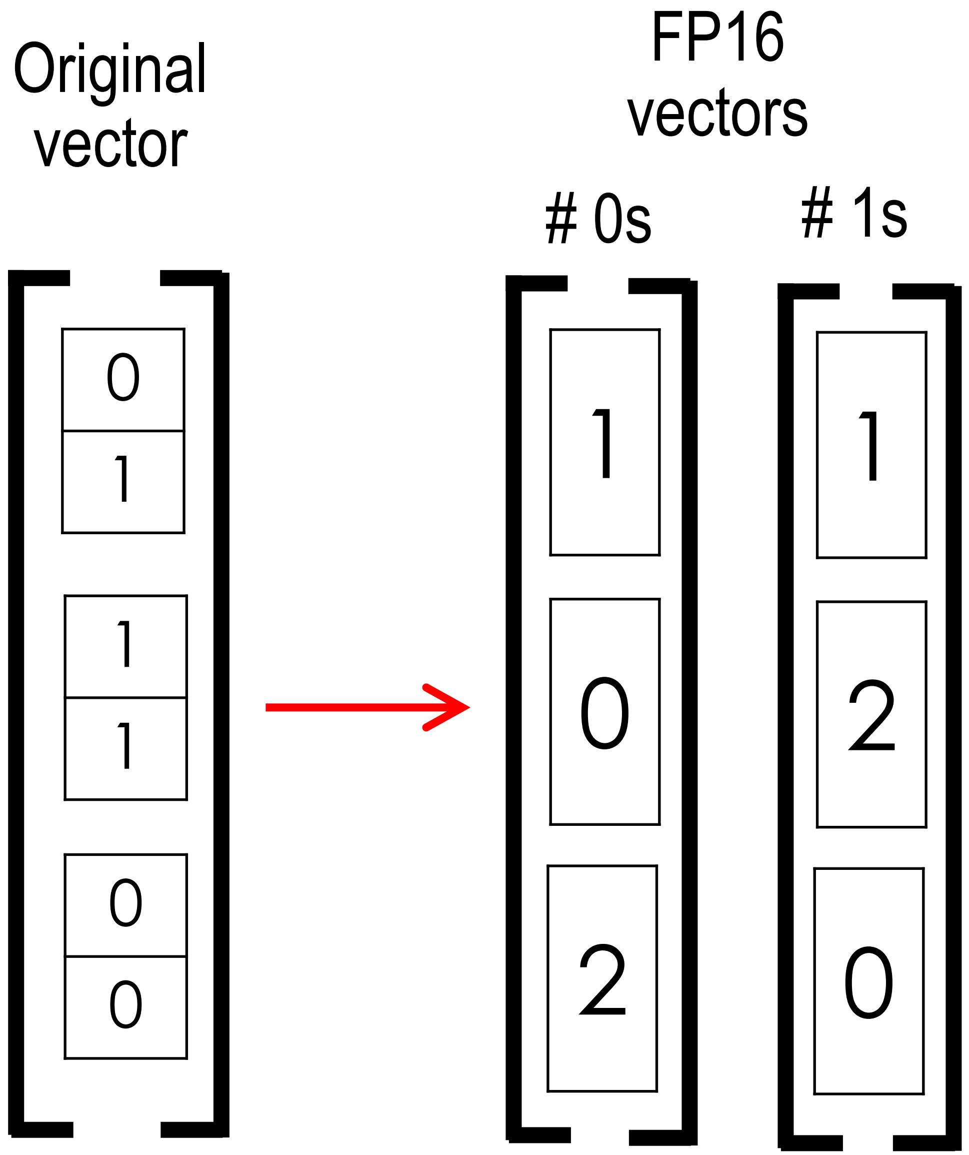
Two vectors of length 1, each entry has 2 bits

Enumeration of all pairings of bit values (“paths”)

Tally of counts of number of occurrences of each pairing type

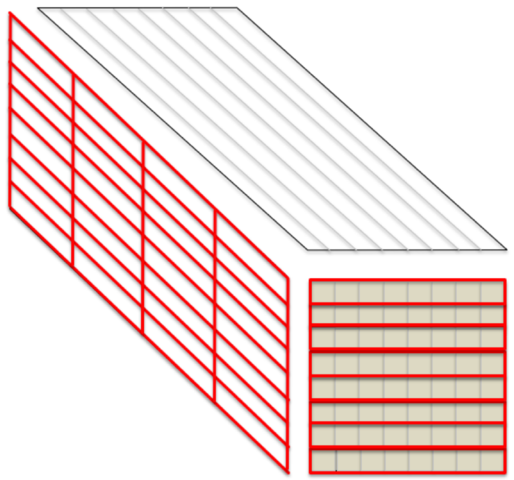
CCC Tensor Core Method on a GPU

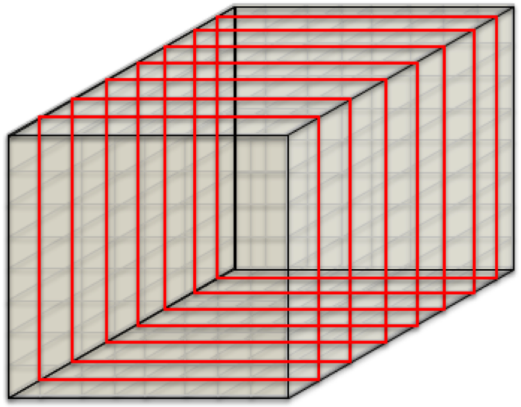
- Each vector is replaced by two vectors, each containing the number of 0s and 1s of each element of the original vector, forming a new matrix of vectors V
- Then taking the dense matrix-matrix product $V^T V$ generates all 2X2 tables for all vector pairs
- Bit-for-bit identical result to previous method
- FP16 is used to hold the 2-bit inputs; the result is accumulated as FP32
- Uses CUDA function **`cublasGemmEx`**
- ~4X faster than original bitwise method



Mapping to Many-GPU Systems

- This method must be mapped to many thousands of GPUs
- The algorithm is very similar to a distributed DGEMM or a tensor contraction
- Thus use a very similar decomposition to Parallel BLAS (PBLAS) or ScaLAPACK: subdivide the rows and columns of the matrices and tensors to achieve multidimensional parallelism by decomposition into blocks:


2-way inputs and results

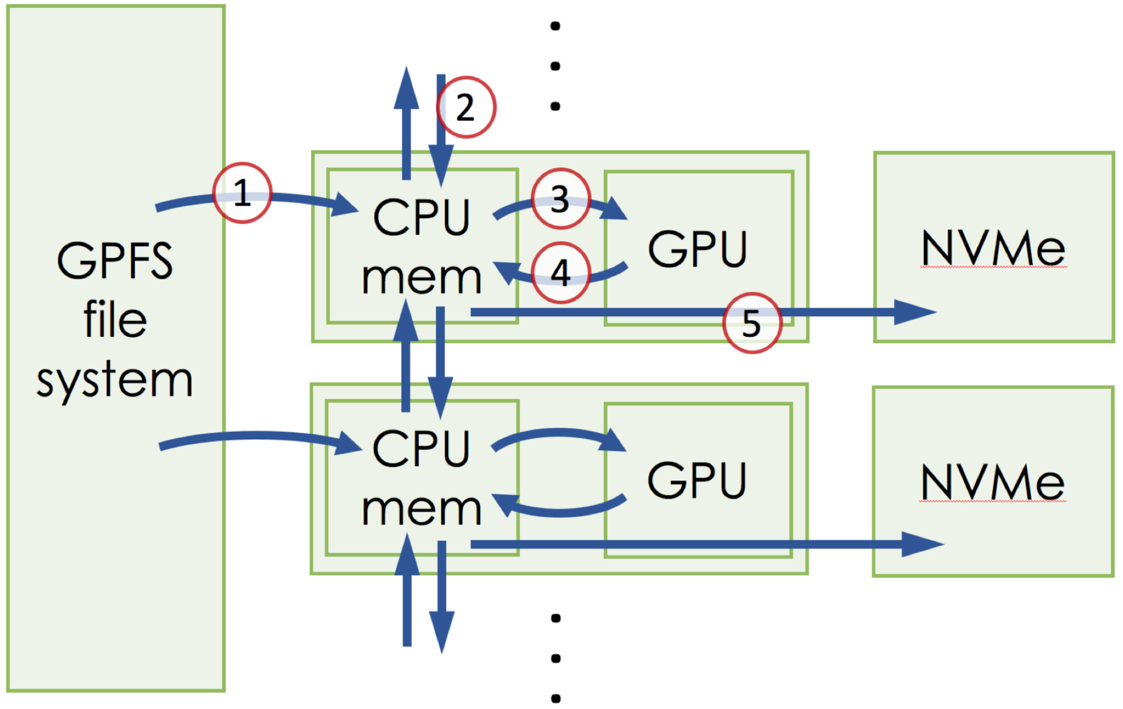

3-way results

Data motion pattern of algorithm

It is extremely important to overlap data motion with computation to get high efficiency

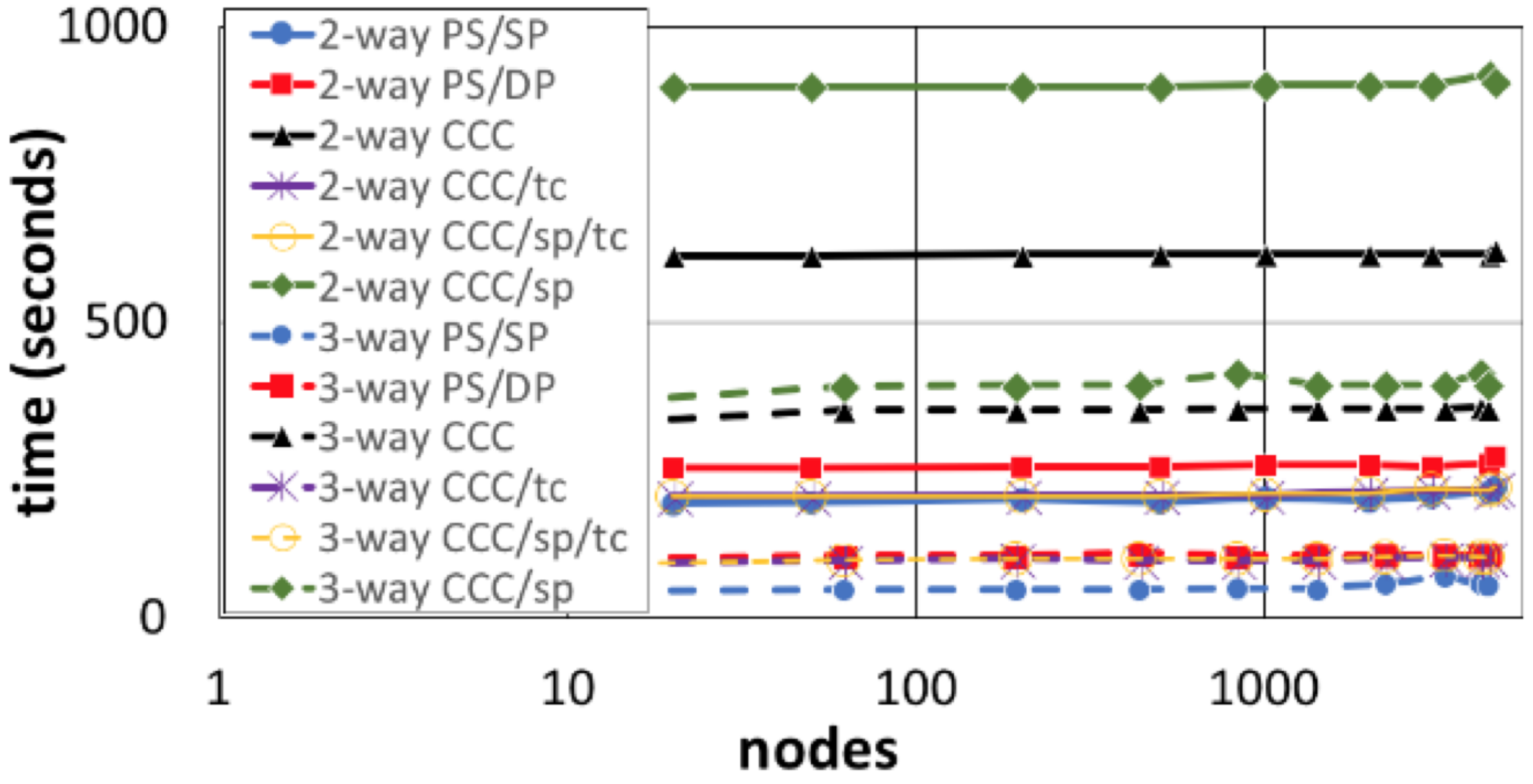
The image below shows the data motion pattern for the 2-way method:

1. A one-time data read from the GPFS file system
2. Stepwise all-to-all communication with other ranks, overlapped with computation
3. Send of data to GPU,
4. Send of result back to CPU, overlapped with computation
5. CPU filters results, writes significant values to node-local NVMe devices



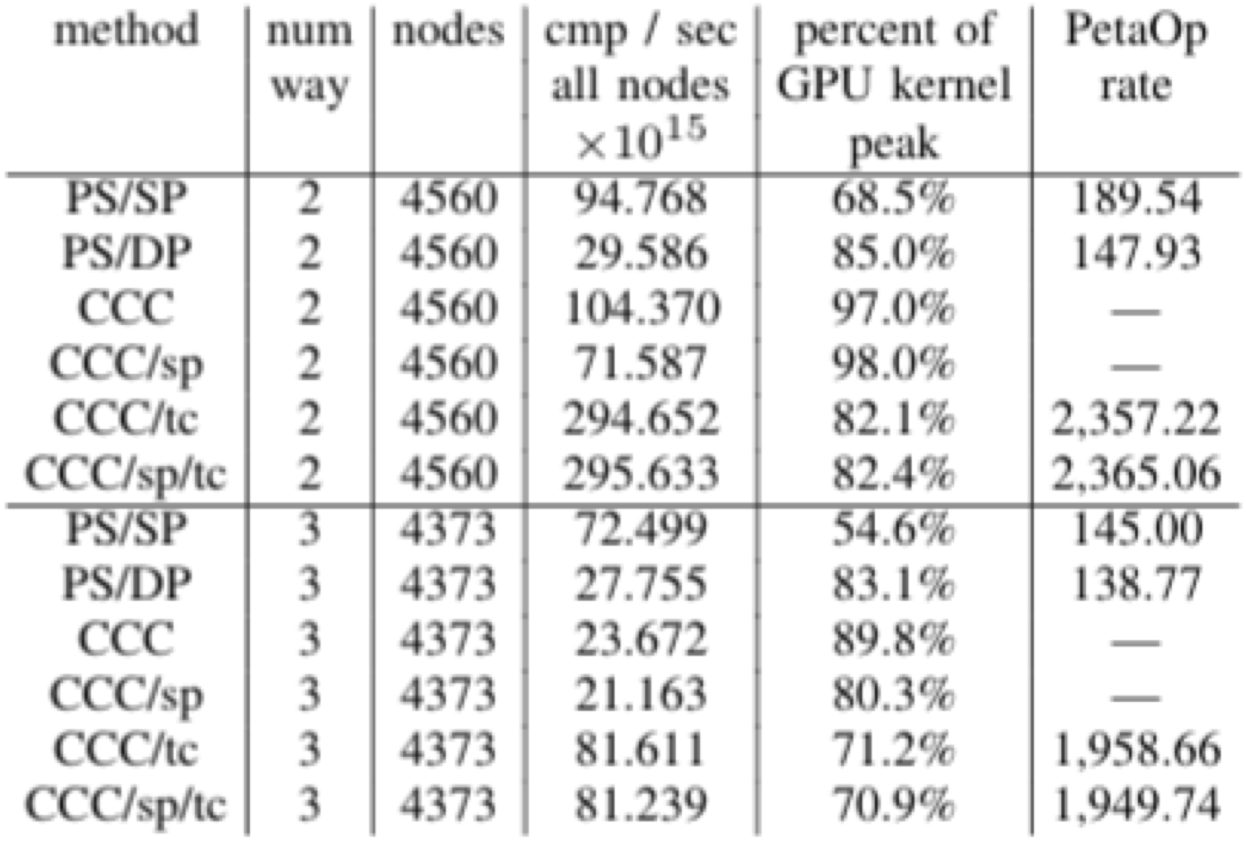
Results: CoMet Weak Scaling on Summit

- All methods are scaled to 99% (2-way methods) or 95% (3-way methods) of Summit
- Compute time is shown, without I/O (lower is better)
- **All methods show near-perfect weak scaling**
- Made possible by aggressive communication overlap and low-congestion Mellanox Infiniband fat tree network with adaptive routing



Summit Absolute Performance at Max Node Counts

- Performance for each method at scale is compared to the highest achievable performance for that method's GPU kernel
- CCC/bitwise method @ scale runs at **98%** of the peak performance that is achievable for its GPU kernel
- CCC Tensor Core method **82%**
- PS: **189.54 single precision PetaOps**
- CCC/sp/Tensor Cores: **2.36 ExaOps** – (equivalent to **86.4 TeraOps** per GPU)
- Tensor Core method comparison rate is **4.13X** higher than bitwise method

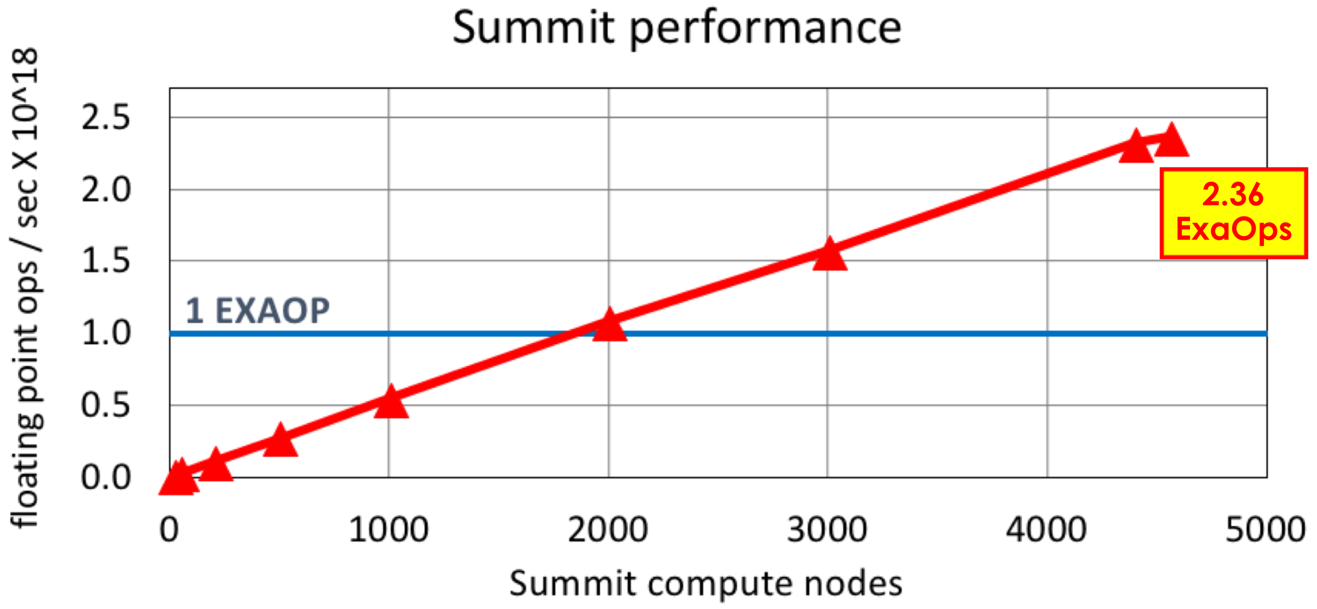


method	num way	nodes	cmp / sec all nodes $\times 10^{15}$	percent of GPU kernel peak	PetaOp rate
PS/SP	2	4560	94.768	68.5%	189.54
PS/DP	2	4560	29.586	85.0%	147.93
CCC	2	4560	104.370	97.0%	—
CCC/sp	2	4560	71.587	98.0%	—
CCC/tc	2	4560	294.652	82.1%	2,357.22
CCC/sp/tc	2	4560	295.633	82.4%	2,365.06
PS/SP	3	4373	72.499	54.6%	145.00
PS/DP	3	4373	27.755	83.1%	138.77
CCC	3	4373	23.672	89.8%	—
CCC/sp	3	4373	21.163	80.3%	—
CCC/tc	3	4373	81.611	71.2%	1,958.66
CCC/sp/tc	3	4373	81.239	70.9%	1,949.74

Summit Performance Compared to Titan

- The comparison rate (measure of science output) for Summit is **36.2X** *higher per GPU* than the Titan (bitwise) method (due to Summit's faster GPUs and Tensor Cores)
- This value normalized to full system is equivalent to Summit giving **53.6X** higher science output rate than Titan
- Note this exceeds the CORAL-2 Exascale performance target of 50X higher app performance than Titan
- We are achieving exascale-class science on a pre-exascale system, thanks to Tensor Cores

System	# GPUs used	% of system used	comparisons /sec
Titan	17,955	96%	5.360e15
Summit	27,360	99%	295.633e15



Issues encountered using Tensor Cores

- Matrices are tall and skinny – axis order had to be reversed to give shorter leading matrix dimension for better TC performance (about 2X faster) (thanks to Sean Treichler of NVIDIA for suggestion)
- HGEMM performance as a function of matrix size is irregular, hard to precisely predict – performed extensive timing tests with Baidu DeepBench benchmark to try to understand – advisable to pad up to a multiple of a small power of 2 (e.g., 8, 16, 32) – however too much padding will be wasteful
- There are many tuning options for HGEMM (~16 choices for the algorithm setting) – determined **`CUBLAS_GEMM_ALGO4_TENSOR_OP`** was the best – would prefer if default setting would give this performance (anticipate improvements with CUDA 10)
- TC/GEMM has surprising data-dependent performance: **~125 TF** theoretical peak, **113 TF** achievable on zero-filled matrices, **105 TF** peak on CCC matrices (random 2-bit entries), **~95 TF** peak on matrices with fully random FP16 entries

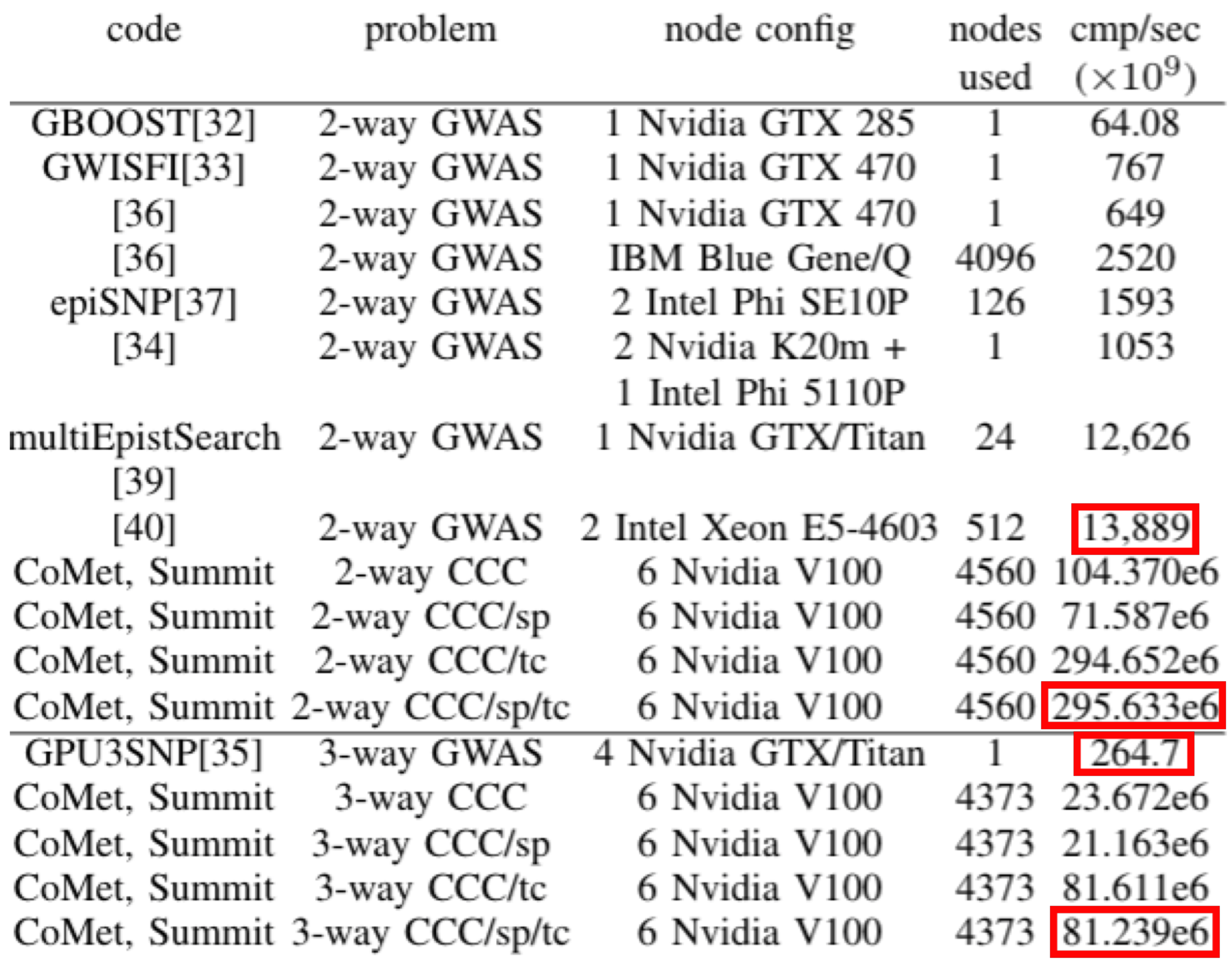
Comparison to Current State of the Art

Fastest known 2-way method was run on 512 nodes of Edison. CoMet exceeds this rate by **21,285X**

Fastest known 3-way method was run on 4 GTX/Titan GPUs. CoMet exceeds this rate by **306,910X**

CoMet runs **4 - 5 orders of magnitude** faster than best current state of the art

Made possible by first-time use of a many-GPU system to solve problems of this type



code	problem	node config	nodes used	cmp/sec ($\times 10^9$)
GBOOST[32]	2-way GWAS	1 Nvidia GTX 285	1	64.08
GWISFI[33]	2-way GWAS	1 Nvidia GTX 470	1	767
[36]	2-way GWAS	1 Nvidia GTX 470	1	649
[36]	2-way GWAS	IBM Blue Gene/Q	4096	2520
epiSNP[37]	2-way GWAS	2 Intel Phi SE10P	126	1593
[34]	2-way GWAS	2 Nvidia K20m + 1 Intel Phi 5110P	1	1053
multiEpistSearch [39]	2-way GWAS	1 Nvidia GTX/Titan	24	12,626
[40]	2-way GWAS	2 Intel Xeon E5-4603	512	$\boxed{13,889}$
CoMet, Summit	2-way CCC	6 Nvidia V100	4560	104.370e6
CoMet, Summit	2-way CCC/sp	6 Nvidia V100	4560	71.587e6
CoMet, Summit	2-way CCC/tc	6 Nvidia V100	4560	294.652e6
CoMet, Summit	2-way CCC/sp/tc	6 Nvidia V100	4560	$\boxed{295.633e6}$
GPU3SNP[35]	3-way GWAS	4 Nvidia GTX/Titan	1	$\boxed{264.7}$
CoMet, Summit	3-way CCC	6 Nvidia V100	4373	23.672e6
CoMet, Summit	3-way CCC/sp	6 Nvidia V100	4373	21.163e6
CoMet, Summit	3-way CCC/tc	6 Nvidia V100	4373	81.611e6
CoMet, Summit	3-way CCC/sp/tc	6 Nvidia V100	4373	$\boxed{81.239e6}$

Performance on a Real-World Problem

- Data from publicly available human genome dataset, 81M vectors of length 600K
- 2-way CCC/sp/tc method is run @ 2/3 of Summit (3,000 nodes)
- Inputs are read from preliminary AlpineTDS filesystem prior to Summit acceptance
- Output are written to on-node NVMe burst buffers

component	time (sec)	percent
core metrics computation	10,550.23	88.80
vectors initialization	0.24	0.00
metrics initialization	58.44	0.49
input	670.93	5.65
output	600.40	5.05
TOTAL	11,880.25	100.00

- The core computation consumes **89% of runtime**; I/O and other overheads only **11%**
- Output time is small because only writing 1 out of every 3 million results (“needle in haystack”)
- Highlights importance of optimizing entire workflow in order to take advantage of speedup from reduced precision
- Core computation runs at **1.50 ExaOps** on 2/3 of Summit, consistent with 2.36 ExaOps rate at 99% of Summit
- **Total job runtime is 3.3 hours on Summit** -- if run at the rate of best comparable state of the art, would require **15 years wallclock runtime** to complete

Opportunities / Considerations

- Generalizability of approach: can be used whenever it is needed to discover relationships between data elements for which connectivity is not known a priori but must be discovered (unlike e.g. PDEs with local, sparse connectivity)
- Other kinds of reduced precision: recent NVIDIA Turing architecture has Tensor Core support for INT8, INT4, INT1 – is this usable by applications?
- Porting effort: does the performance benefit of using mixed precision outweigh the extra development and maintenance effort? CoMet Tensor Core method required 2 weeks to prototype, 1 additional month code tuning, result was 4X speedup – in this case well worth it.
- It will be helpful if users can make use of libraries to take advantage of mixed precision, though this may not always be possible (cf. CUDA WMMA API)
- As developers we may need to repeatedly rethink and adapt our algorithms as we get new kinds of accelerators and heterogeneity in the future (cf. Extreme Heterogeneity workshop. PMES workshops)

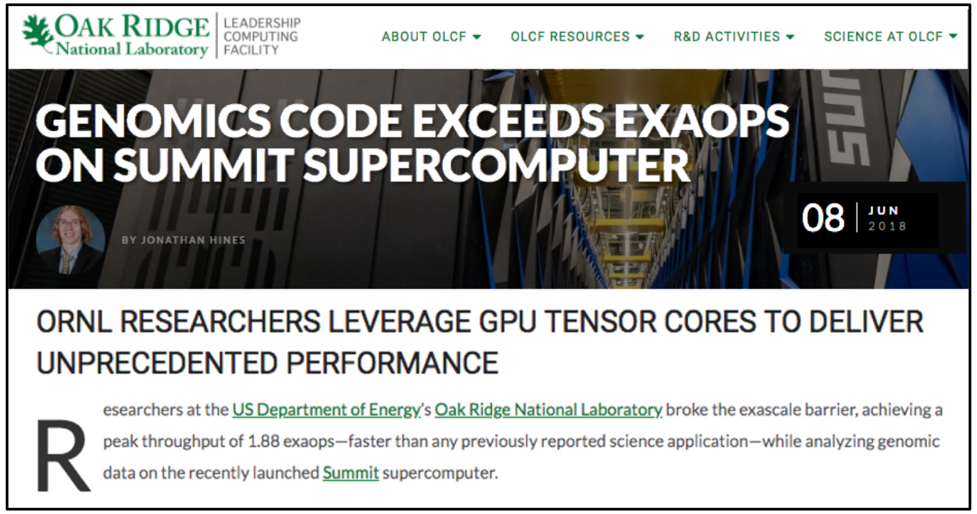
Conclusions

- We have found a way to map a data analytics application to GPUs and exploit fast low-precision hardware on Summit’s Volta GPUs
- Using the Tensor Cores for mixed precision gave us about 4X performance improvement over the previous implementation on Summit
- This enables a huge advance over state of the art and will allow us to solve previously unsolvable problems
- This work highlights the growing need to make use of the new kinds of hardware we’re increasingly seeing. “*Whatever it takes*”
- Also highlights the need to optimize data motion (and the entire workflow) to make best use of processor advances
- At the OLCF we are hoping to find more opportunities to exploit unconventional hardware features on current and future systems

References / Acknowledgements

- W. Joubert, J. Nance, D. Weighill, D. Jacobson, "Parallel Accelerated Vector Similarity Calculations for Genomics Applications," arxiv 1705.08210 [cs], *Parallel Computing,* 2018.
- W. Joubert, J. Nance, S. Climer, D. Weighill, D. Jacobson, "Parallel Accelerated Custom Correlation Coefficient Calculations for Genomics Applications," arxiv 1705.08213 [cs], *Parallel Computing,* 2019.
- Wayne Joubert, Deborah Weighill, David Kainer, Sharlee Climer, Amy Justice, Kjiersten Fagnan, Daniel Jacobson, "Attacking the Opioid Epidemic: Determining the Epistatic and Pleiotropic Genetic Architectures for Chronic Pain and Opioid Addiction," *SC18.*

CoMet: World's First Application to Achieve an ExaOp, Summit Launch, June 8, 2018



OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY
ABOUT OLCF
OLCF RESOURCES
R&D ACTIVITIES
SCIENCE AT OLCF

GENOMICS CODE EXCEEDS EXAOPS ON SUMMIT SUPERCOMPUTER

BY JONATHAN HINES

08 JUN 2018

ORNL RESEARCHERS LEVERAGE GPU TENSOR CORES TO DELIVER UNPRECEDENTED PERFORMANCE

Researchers at the US Department of Energy's Oak Ridge National Laboratory broke the exascale barrier, achieving a peak throughput of 1.88 exaops—faster than any previously reported science application—while analyzing genomic data on the recently launched Summit supercomputer.

Dan Jacobson and Wayne Joubert describe genomics ExaOp breakthrough to DOE Secretary Perry, Ginny Rometty, Jensun Huang and Thomas Zacharia – ORNL Summit Launch, June 8, 2018

Gordon Bell Award Winner, SC18, November 2018

ACM GORDON BELL PRIZE – WINNER
SUSTAINED PERFORMANCE
"Attacking the Opioid Epidemic: Determining the Epistatic and Pleiotropic Genetic Architectures for Chronic Pain and Opioid Addiction"
Research led by Wayne Joubert
Oak Ridge National Laboratory
SC18

Questions?

Wayne Joubert
joubert@ornl.gov

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.