OAK RIDGE National Laboratory

Using Innovative Compute Hardware for Breakthrough Science: Computational Genomics at Exascale With the CoMet Application

Wayne Joubert

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

2019 DOE CSGF Annual Program Review
July 14-18, 2019

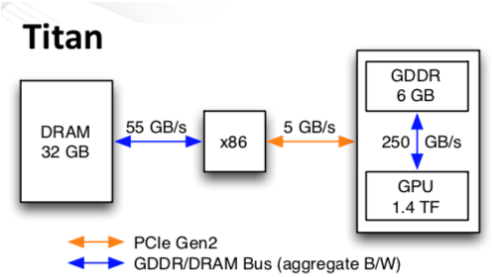
U.S. DEPARTMENT OF ENERGY

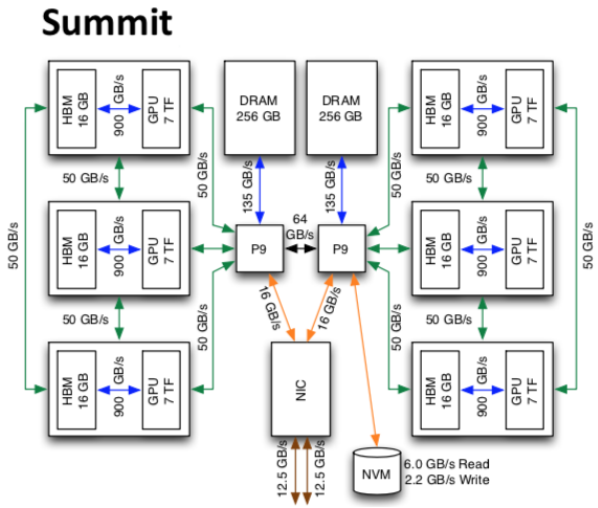
The Rise of Heterogeneous Computing

- Computing devices are becoming increasingly *heterogeneous*
  - totally different kinds of processors being developed for different end uses
  - individual processors and compute nodes that have a combination of very different hardware features and capabilities
- In the broad computing market …
  - Servers: in data centers are using unconventional hardware in production: FPGAs (Microsoft), TPUs (Google), GPUs (multiple cloud providers)
  - Mobile devices: Apple iPhone has 30 different special purpose compute accelerators on the device
  - this trend observed by Herb Sutter, 2012: “Welcome to the [Compute] Jungle” – trend of increasing processor variety – and resulting challenge for software developers

A World of Heterogeneous Computing (2)

- HPC market
  - discrete GPU accelerators coupled with conventional CPUs are becoming common on large HPC systems
  - June 2018: over half of the floating point computing capability on the TOP500 list is now supplied by GPUs
  - computing system nodes are becoming more complex
  - more features being added to processors, for example reduced precision hardware to support deep learning
- Why this is happening
  - fundamental reason: year-over-year exponential growth in conventional CPU speeds is slowing
  - processor performance is limited by power constraints
  - scaling down semiconductor feature sizes is becoming more challenging
  - in response, vendors are designing more complex, heterogeneous processors to continue to improve performance by other means





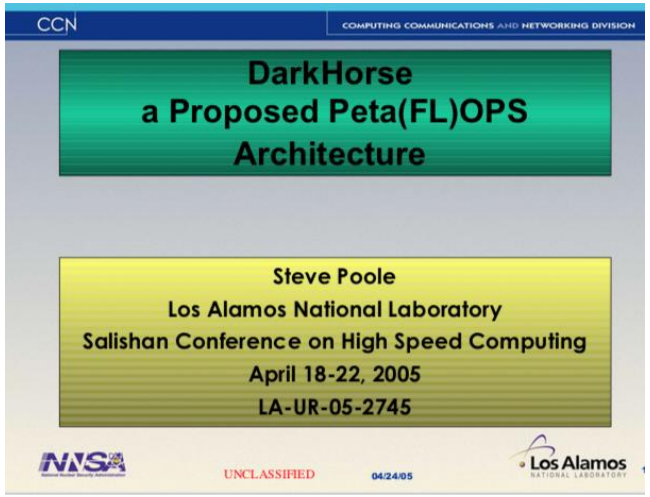
Heterogeneous Computing: Implications

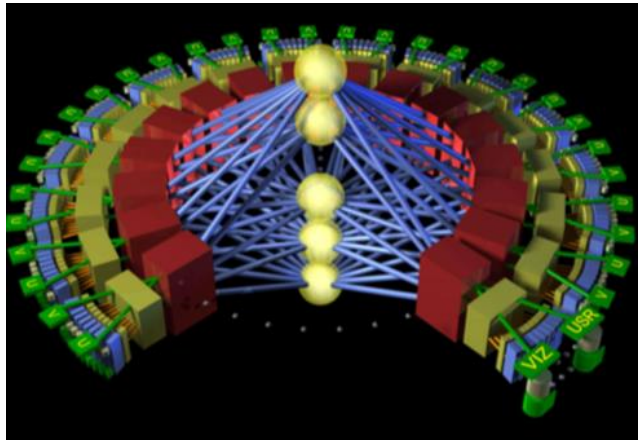
- Bad News: ☹
  - different kinds of compute hardware – difficult to program, challenging to maintain portable code to different hardware that performs well on all platforms
  - longer term: our ability to continually increase the speed of scientific simulations faces an uncertain future

- Good News: 😀
  - “A New Golden Age for Computer Architecture” – an exciting time for innovation
  - Exciting opportunities for developing new algorithms that map well to new hardware, offering tremendous potential gains in performance for science applications

The Origins of Heterogeneous Computing

- The “supercomputer market crash” of early 1990s gave rise to status quo of distributed MPI computing with commodity processors
- Early 2000s, LANL DarkHorse Project
  - recognized a radical new approach was needed to get to exascale
  - original ideas 1997-1998
  - prescient: GPUs, FPGAs, 3-D stacked memory (HBM)
- LANL Advanced Architectures LDRD project, 2003-2005
  - ported DOE science applications to GPUs and FPGAs
- LANL Roadrunner
  - 2006 contract signed, 2008 broke Petascale barrier
  - Challenging, ambitious effort
  - was difficult to program (3 kinds of processing units)
  - successful tenure through 2013 solving science problems of importance to DOE



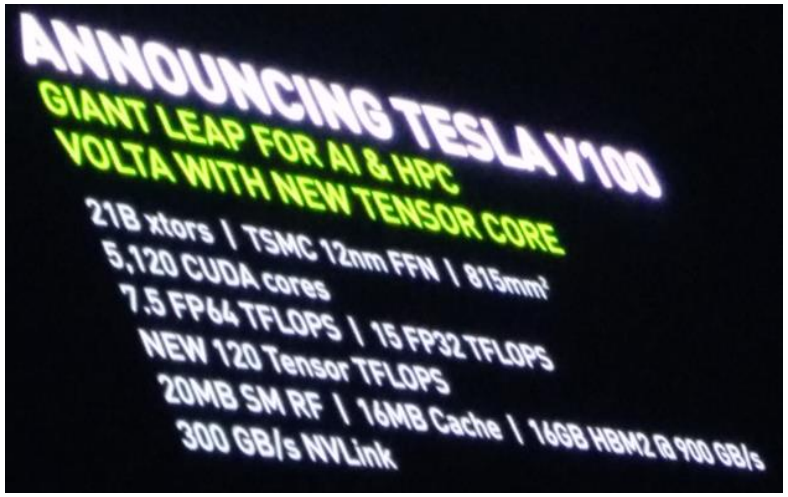


The rise of heterogeneous computing

- 2008 Exascale Computing Study (Kogge et al.) (DARPA, IPTO, AFRL)
  - predicted that a 1 ExaFlops system would require 290-470 MW power (cost up to $1/2 B / year for a single system!!) (the actual number will be ~ 10X less, thanks in part to GPUs)
- April 2009 - ORNL and Cray started planning the Titan GPU-based system
  - many early intense discussions of how to port applications to this system
  - deployed for general use in 2013, will be decommissioned Aug. 1, 2019
  - easier to program - CUDA, OpenACC (later OpenMP 4/5, Kokkos, RAJA, ...)
  - has had a successful run delivering billions of core hours to its users
- 2019: now ALL next-generation leadership class systems announced in the DOE complex will be accelerated …
- … yet systems are still becoming even *more* heterogeneous (cf. Summit Volta Tensor Cores)

Tensor Core Acceleration of Applications

- On-package reduced precision hardware units
- Developed to address exploding computational needs in deep learning, data analytics
- NVIDIA first announced @ NVIDIA GTC, May 11, 2017
- Half precision matrix multiplies **16X faster** than double precision



- Other vendors also providing reduced precision processors – Google TPU; AMD Radeon GPUs; ~ 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 of the 2018 Gordon Bell Finalist teams used the Tensor Cores in some fashion
- This talk will describe one of these codes, the CoMet computational genomics application

The CoMet comparative genomics application

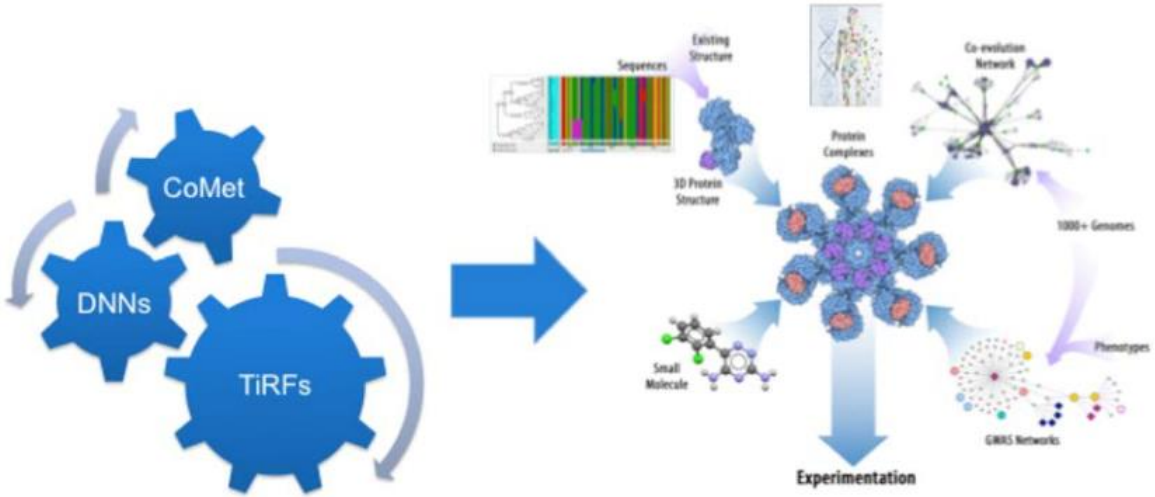
A new biosciences application, CoMet = Combinatorial Metrics code

Used to identify genomic features within a population with applications to finding genetic causes of diseases

Not a “traditional” modeling and simulation code (e.g., continuum PDE solver, PIC, Monte Carlo, etc.)

Also is not a deep learning app per se, though is part of an AI workflow

Best described as a *data analytics application* used in comparative genomics studies



CoMet: Solving a "Needle in a Haystack" Problem

- Is used to find genetic causes of individual traits such as susceptibility to a disease
- These traits can be caused by a complex interaction of genetic features
- However it is unknown beforehand which of the millions of genetic features are interacting to cause these traits
- It is a huge combinatorial problem to search across all these combinations of features to find the important ones

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

- To solve this we use vector similarity search, to find all clusters of vectors (representing the genetic features) that are similar to each other
- The computational complexity of these methods is exponential in the cluster size – $O(n^k m)$ complexity for m vectors of length n clustered into groups of k vectors
- An extremely expensive computation and is also highly network intensive because of the all-to-all comparison, all compute nodes must exchange data with all other compute nodes

Solving the Vector Similarity Problem

- Vector similarity search is very similar to another known problem
- Has an identical computational pattern to the dense matrix multiply operation, “GEMM” general dense matrix-matrix product
- GEMMs can already be computed efficiently by existing high performance software libraries (e.g., NVIDIA CUBLAS, Intel MKL)
- These libraries schedule the GEMM computations to make best use of the memory hierarchy (registers, caches) on the processor
- These libraries can thus be adapted to perform the required vector similarity calculations

Vector Similarity Methods in CoMet

1. **Proportional Similarity (PS) Metric**:
   - very much like a GEMM but replaces the floating point multiply with a “minimum of scalars” operation
   - this “minimum of scalars” is implemented in hardware on many modern processors
2. **Custom Correlation Coefficient (CCC)**:
   - this method operates on binary allele data – it counts the occurrences of joint relationships between genetic features
   - its computation can exploit the “population count” hardware instruction present on many modern processors

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

The CCC Method: Implementing on GPUs

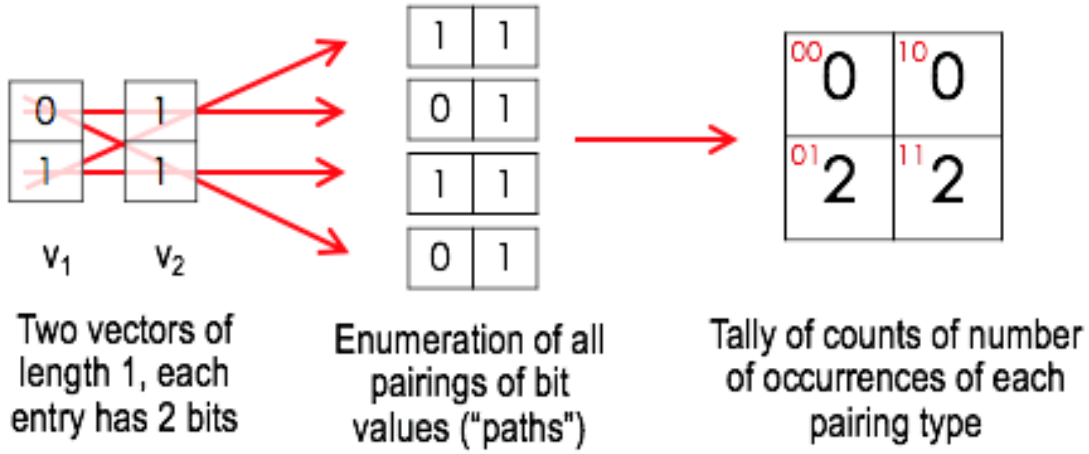
Performs calculations on allele data stored as 2-bit entries stored into vectors

The original method on GPUs uses bitwise operations on 64-bit ints

The 2-bit input values are packed into 64-bit words and operated on with binary AND, OR, NOT operations

CUDA intrinsic **`__popcll`** hardware population count is used for high speed

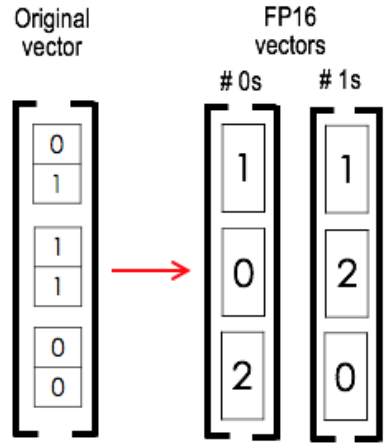
Implemented with modified MAGMA library to take advantage of MAGMA's highly optimized GEMM operations



CCC Method on GPUs Using Tensor Cores

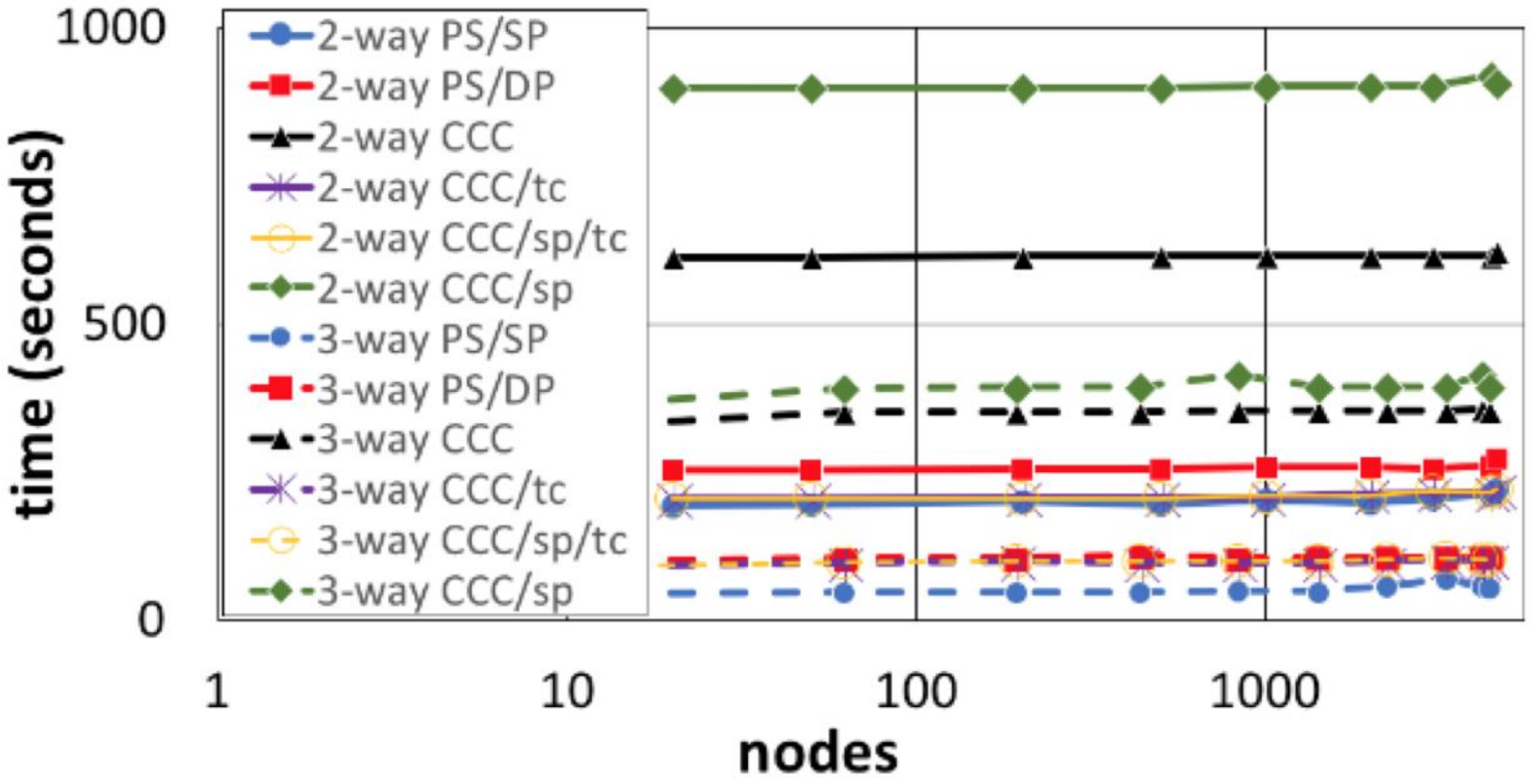
- New Tensor Core method uses a mathematical “trick” to convert CCC calculation into a standard GEMM matrix multiply to count the values
- This can be done in half precision with no loss of accuracy
- We use the Tensor Cores – originally designed for deep learning applications but we have adapted to this use

- 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
- FP16 is used to hold the 2-bit inputs; the result is accumulated as FP32
- Uses CUDA function **`cublasGemmEx`**



CoMet Results on Summit: Weak Scaling

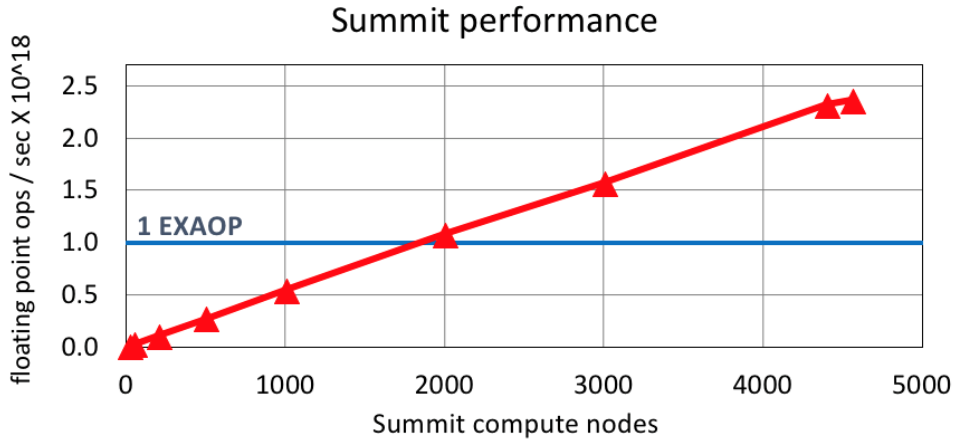
- Scale-out performance test on up to 99% of Summit nodes
- Fixed work per node, measuring wallclock time to solution
- **All methods show near-perfect scale-out**
- Very communication-intensive algorithm exploits Summit's low-congestion Mellanox Infiniband fat tree network with adaptive routing



Performance on the Full Summit System

- CCC/sp/Tensor Cores: **2.36 ExaOps**
- First reported ExaOp calculation by an application, June 8, 2018
- This is 75% of Summit's peak achievable performance of ~ 3.2 ExaOps
- Use of Tensor Cores improves performance by **4.13X** compared to original bitwise method
- We are already achieving Exascale-class performance on the pre-exascale Summit system
- True double precision Exascale performance will come with the Frontier system and other exascale systems in the 2021-22 timeframe

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



Comparison to State of the Art Implementations

Comparison with other efforts reported in the literature to adapt these methods to GPUs and to parallel systems

Fastest known 2-way method (cluster size k = 2) 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	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	295.633e6
GPU3SNP[35]	3-way GWAS	4 Nvidia GTX/Titan	1	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	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 AlpineTDS GPFS parallel filesystem
- 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%**
- 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

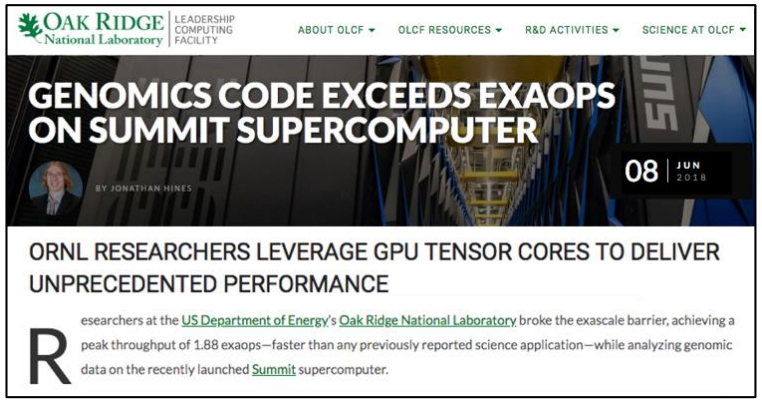
Team Acknowledgements

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,* accepted.

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

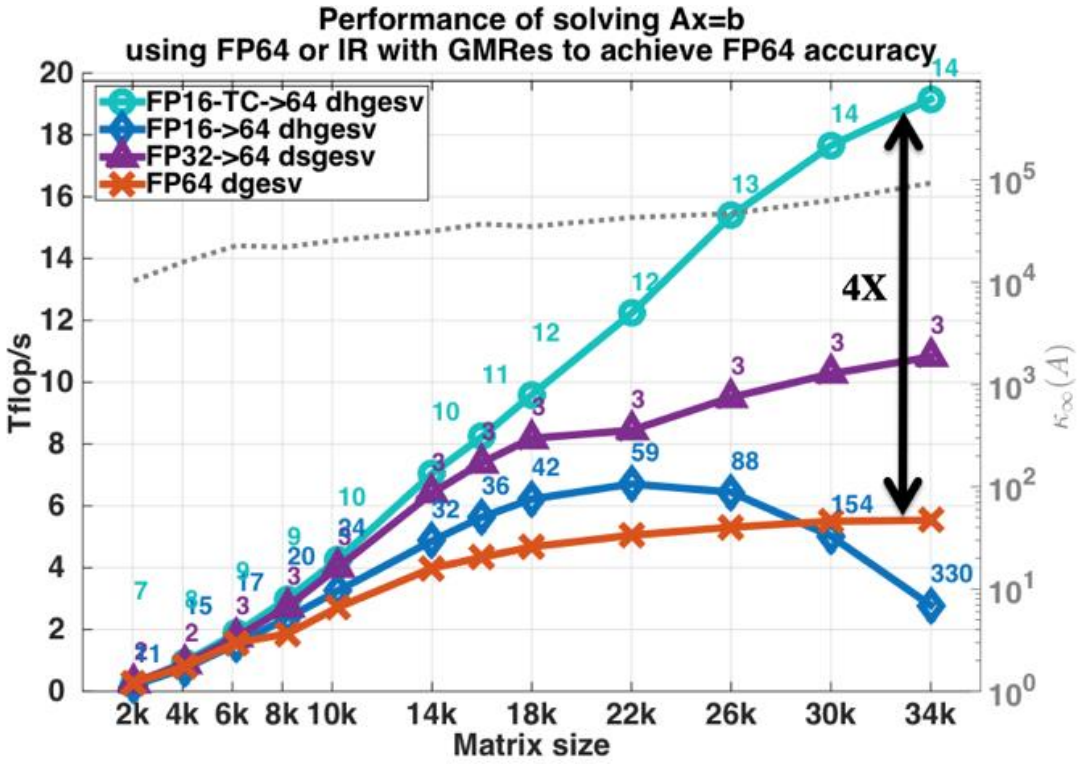


Dan Jacobson and Wayne Joubert describe genomics ExaOp breakthrough to DOE Secretary Perry, Ginny Rometty, Jensun Huang and Thomas Zacharia – ORNL Summit Launch, June 8, 2018

Gordon Bell Award Winner SC18, November 2018

Growing Interest in Mixed Precision

- Multiple Gordon Bell entries 2018, 2019 used mixed precision
- SC18 ~ 15 presentations, posters, sessions on mixed precision methods: dense linear algebra iterative refinement, FFTs, multigrid solvers, finite element simulations, Newton methods, tools
- UTK Innovative Computing Laboratory using mixed precision to accelerate dense linear algebra
- ORNL mixed precision working group, exploring other opportunities, discussions with NVIDIA



Graphic courtesy Stan Tomov, ICL

The Future of Heterogeneous Computing

- We are not at the end, but at the beginning of a period of increasingly heterogeneous compute nodes – GPUs, reduced precision units are just first steps
- Predictions:
  - David Patterson @SCALE 2018 keynote: the future is Domain Specific Architectures, customized to specific problem domains
  - Jack Dongarra, ISC19: ”By 2030, there will be no more general purpose computing”
  - ASCR Workshop on Extreme Heterogeneity, 2018 – a future of increasingly diverse accelerators, eventuating possibly in neuromorphic or quantum accelerators
  - IEEE International Roadmap: Beyond CMOS: analog computing, probabilistic circuits, reversible computing
  - Bill Dally: NVIDIA experimenting with analog computing

Sidebar: The Need to Accelerate Scientific Codes

- Scientist's dream: "I just want to write my code to express my science, I want it to run fast, I don't want to worry about hardware."
- Many attempts and claims have been made for programming languages and tools to solve this problem. No silver bullet has emerged, though some things help (directives, abstraction layers, domain-specific languages).
- Understandable to wish we would never need to port codes to GPUs (20 years ago similar concerns about MPI)
- Not all codes need to be GPU-accelerated (or even parallel)
- On the other hand, at the high end, some science applications that have not or cannot use GPUs have been abandoned for use in leadership computing – e.g., computational chemistry code previously used at OLCF, doesn't use GPUs, no longer used, replaced by QMCPACK

Critical Role of New (Parallel) Algorithms

- Real revolutions in computing have come from breakthroughs in algorithm complexity, e.g. FFT $O(n^2) \rightarrow O(n \log(n))$, Multigrid $O(n^{3/2})$, $O(n^{4/3}) \rightarrow O(n)$
- May be even more important than faster hardware (e.g., MIP solvers over 25 years: 500,000X speedup from hardware, 1,500,000X speedup from software/heuristics/algorithms)
- Finding new ways to map difficult algorithms to parallel or accelerated hardware has led to breakthrough results, sometimes changing the effective computational complexity achievable on parallel systems
  - Ray tracing algorithms mapped to GPUs
  - Event-based methods to map Monte Carlo radiation transport to GPUs
  - Aggressive coarsening methods to map 3-D algebraic multigrid solvers to parallel systems
  - High order tensor elements to improve computational intensity (ECP CEED codesign center)
  - Communication-avoiding Krylov solver methods for bandwidth, latency sensitive problems
  - Reduced precision dense and sparse linear solvers
  - KBA methods to map wavefront algorithms to parallel and accelerated hardware

Example Challenge Problems:

1. How can you accelerate your science algorithm on a processor that doesn't have or a floating point unit (or an instruction set) (neuromorphic processor) (has been done for simple diffusion solvers, others – with huge decrease in power consumption)
2. What if a specific linear algebra operation could be done in O(n) time instead of $O(n^2)$ time on an accelerator, could your science application exploit it (quantum processor)
3. What if you could multiply pairs of dense matrices 100X faster by representing the inputs as 1-bit values – could you use this feature (available now on NVIDIA Turing architecture)
4. If you had a very fast analog computing circuit that could give you an initial approximate guess at a solution, could this help you solve your science problem faster
5. (Reverse question: can we abstract computational kernels useful to multiple applications, present this as a "wish list" for custom hardware to vendors?)

Conclusions

- In HPC we are entering a (difficult, challenging / exciting, fun) time for developing new computational methods and software
- New kinds of compute hardware will continue to emerge in the coming years – some totally unexpected (Tensor Cores, TPUs were totally unforeseen 10 years ago)
- Some applications will not survive these changes, some will adapt, some apps will be rewritten, some brand new applications will be developed and thrive
- CoMet is an example of an app making a huge speedup over state of the art by exploiting new hardware capabilities
- Code performance portability, maintenance will be a real challenge in this environment
- As computational scientists and algorithm developers, we need to understand current approaches and prior art and also be ready to take a blank-sheet-of-paper fresh look at how to solve the science problems that we care about
- Diverse skills and approaches to problem solving will continue to be important for our science teams in this environment

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.