The CoMet comparative genomics application

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

HPC User Forum
April 1-3, 2019
Mixed Precision for Accelerating Comparative Genomics

- **Mixed precision processors** are currently being developed to meet the explosive demands for deep learning / data analytics calculations.
- **Deep learning accelerators** is the #1 trend in the IEEE Computer Society 2019 list of Top Ten Technology Trends.
- Multiple vendors developing: NVIDIA T4 and Volta Tensor Core GPUs, AMD Radeon MI60 Accelerators, Google TPUs, ~50 startups developing custom DL processors, some with as low as 1-bit precision.
- Huge potential performance boost: NVIDIA Tensor Core half precision matrix multiplies are **16X faster** than double precision.
- This hardware is already being used for HPC: 4 of the 6 Gordon Bell Finalists at SC18 used the Tensor Cores on the ORNL Summit system for their calculations.
- In this talk we will describe how we used Summit to accelerate the CoMet comparative genomics code.
Summit – An Introduction

Officially launched June 8, 2018

Entered into general use Jan 1, 2019

Currently the world’s fastest supercomputer

Peak speed 200 Petaflops double precision, 3.2 ExaOps mixed precision

#1 on TOP500, @ 143.5 PF for HPL benchmark, November 2018

#1 Green500 level-3 measured system for power efficiency

#1 on HPCG benchmark

Additionally achieved world’s first ExaOp calculation by an application, @ 2.36 ExaOps

Already generating new science for 30+ projects from the DOE INCITE program and other programs
Summit Overview

Compute System
- 10.2 PB Total Memory
- 256 compute racks
- 4,608 compute nodes
- Mellanox EDR IB fabric
- 200 PFLOPS
- ~13 MW

Compute Rack
- 18 Compute Servers
- Warm water (70°F direct-cooled components)
- RDHX for air-cooled components

Components
- IBM POWER9
  - 22 Cores
  - 4 Threads/core
  - NVLink
- NVMe-compatible PCIe 1600 GB SSD
- 25 GB/s EDR IB- (2 ports)
- 512 GB DRAM- (DDR4)
- 96 GB HBM- (3D Stacked)
- Coherent Shared Memory

GPFS File System
- 250 PB storage
- 2.5 TB/s read, 2.5 TB/s write

NVIDIA GV100
- 7 TF
- 16 GB @ 0.9 TB/s
- NVLink

Slide courtesy Jack Wells
Summit Node Overview

Power Supplies (2x)
- 2200W
- 200V/AC, 277V/AC, 400V/DC input.

NVIDIA Volta GPU
- 3 per socket
- 800W form factor
- 300W
- NVLink 2.0
- Air/Water Cooled

Memory DIMM’s (16x)
- 6 DDR4 RDIMMs per socket
- 8, 16, 32, 64, 128GB DIMMs

TF 42 TF (6x7 TF)
HBM 96 GB (6x16 GB)
DRAM 512 GB (2x16x16 GB)
NET 25 GB/s (2x12.5 GB/s)
MM/s 83

HBM & DRAM speeds are aggregate (Read+Write).
All other speeds (X-Bus, NVLink, PCIe, IB) are bi-directional.
Using Summit to Solve Comparative Genomics Problems

• We seek the 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
• 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
How do we solve this 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 (2)

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

<table>
<thead>
<tr>
<th>method</th>
<th>Proportional Similarity (PS) method</th>
<th>Custom Correlation Coefficient (CCC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputs</td>
<td>real-valued inputs</td>
<td>2-bit allele values</td>
</tr>
<tr>
<td><strong>2-way</strong></td>
<td>$e_2(u, v) = 2 \left[ \frac{\sum_{q} \min(u_q, v_q)}{\sum_{q} (u_q + v_q)} \right]$</td>
<td>$e_2(u, v) = (3/2) \left[ \sum_{q} \min(u_q, v_q) + \min(u_q, v_q) \right]$</td>
</tr>
<tr>
<td><strong>3-way</strong></td>
<td>$e_3(u, v, w) = (3/2) \left[ \sum_{q} \min(u_q, v_q, w_q) + \min(u_q, v_q, w_q) \right]$</td>
<td>$e_3(u, v, w) = (3/2) \left[ \sum_{q} \min(u_q, v_q, w_q) + \min(u_q, v_q, w_q) \right]$</td>
</tr>
</tbody>
</table>
The Proportional Similarity Method on GPUs

- To implement on GPUs, the MAGMA linear algebra library developed at Jack Dongarra’s Innovative Computing Laboratory (ICL) is modified to support the needed operations

- The MAGMA GEMM kernels are modified to use CUDA minimum of scalars intrinsics $\texttt{fmin}$, $\texttt{fminf}$ for very high speed on the GPUs
CCC Method on GPUs: Method 1 (bitwise method)

- Also modify MAGMA library to take advantage of MAGMA’s highly optimized GEMM operations
- 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
CCC Method on GPUs: Method 2 (Tensor Core method)

• 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 in fact 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 $2 \times 2$ tables for all vector pairs
- FP16 is used to hold the 2-bit inputs; the result is accumulated as FP32
- Uses CUDA function cublasGemmEx
Mapping to Many-GPU Systems

- These methods need to run not just on single GPUs but also on multi-GPU systems with tens of thousands of GPUs.
- To parallelize the methods, we decompose the matrices by rows and columns.
- The resulting matrix blocks are then distributed to the many GPUs to compute the results.
- The enormous amount of data transfer must be done asynchronously to avoid interfering with the computations.

2-way method

3-way method
CoMet Results on Summit: Weak Scaling

- Scale-out performance test on full Summit system
- Wallclock time to solution is measured, with fixed problem size per node
- **All methods show near-perfect scale-out**
- Made possible by aggressive communication overlap and low-congestion Mellanox Infiniband fat tree network with adaptive routing
Performance on the Full Summit System

- CCC/sp/Tensor Cores: **2.36 ExaOps**
- This is 75% of Summit’s peak achievable performance of 3.2 ExaOps
- Equivalent to **86.4 TeraOps** per GPU
- 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
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.
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
- 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
Final Thoughts, Questions

• Reduced precision offers a great performance boost – are there many more applications can use them?

• Lower precisions are coming -- NVIDIA Turing architecture supports INT8, INT4 and INT1 in the Tensor Cores – can our apps use these?

• NVIDIA Tensor Cores only support matrix products – are there other operations that would be useful to run in special purpose compute units?

• “Compute jungle” of increasingly heterogeneous processors as CMOS scaling slows – this is challenging to our developers: code performance portability and maintainability

• Increasing gap between rapidly-growing flop rates between processor generations and slower-growing memory and interconnect bandwidths. This change in system balance is stressing our algorithms – can our codes continue to adapt?
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 additional performance 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 need to optimize our algorithms to current and future architectures to achieve high computational intensity.

- Also highlights the growing need to make use of the new kinds of hardware we’re increasingly seeing. “Whatever it takes”

- At the OLCF we are investigating further opportunities to exploit unconventional hardware features on current and future systems.
References / Acknowledgements


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

ORNL researchers leverage GPU tensor cores to deliver unprecedented performance.

Gordon Bell Award Winner, SC18, November 2018
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.
Data scheduling

We need to use nearly every hardware feature of the Summit node

1. GPFS parallel file system for input
2. Data exchange between nodes using Mellanox Infiniband fat tree network with adaptive routing
3. NVLINK-2 connection between POWER9 CPU and NVIDIA Volta GPU
4. Output written to high speed on-node NVMe SSD devices
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
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 (previously reported 25X)
- 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

<table>
<thead>
<tr>
<th>System</th>
<th># GPUs used</th>
<th>% of system used</th>
<th>comparisons /sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>17,955</td>
<td>96%</td>
<td>5.360e15</td>
</tr>
<tr>
<td>Summit</td>
<td>27,360</td>
<td>99%</td>
<td>295.633e15</td>
</tr>
</tbody>
</table>

![Summit performance graph]

**Summit performance**

- Floating point ops / sec x 10^18
- Summit compute nodes
- 1 EXAOP