CoMet Application Readiness on Crusher

Wayne Joubert, Oak Ridge Leadership Computing Facility

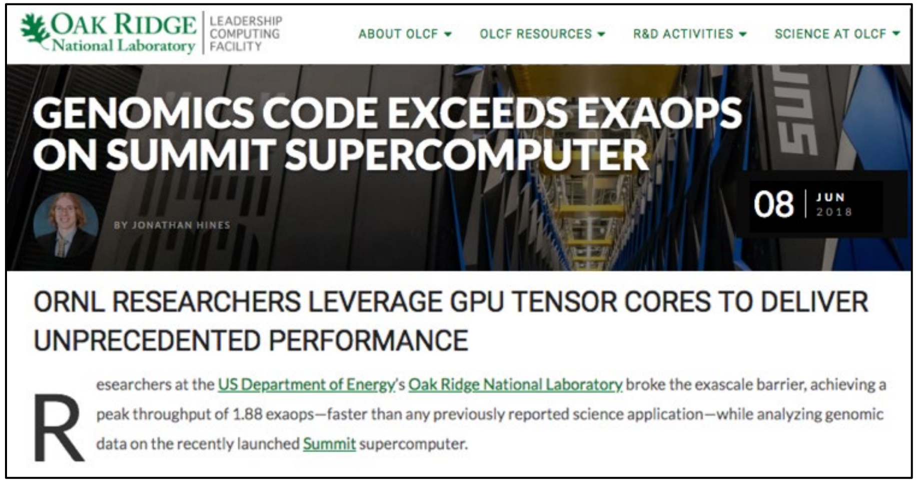
CAAR PI: Dan Jacobson

ORNL is managed by UT-Battelle LLC for the US Department of Energy

CoMet Application Overview

- CoMet is a data analytics application used to find relationships in large datasets
- Applications include Human Health (opioid addiction, Alzheimer’s disease, etc.), Bioenergy (poplar, switchgrass), climate (climatype analysis)
- Peak performance is over 2 ExaOps mixed precision on Summit
- Gordon Bell Prize Winner 2018

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



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.

CoMet readiness activities - overview

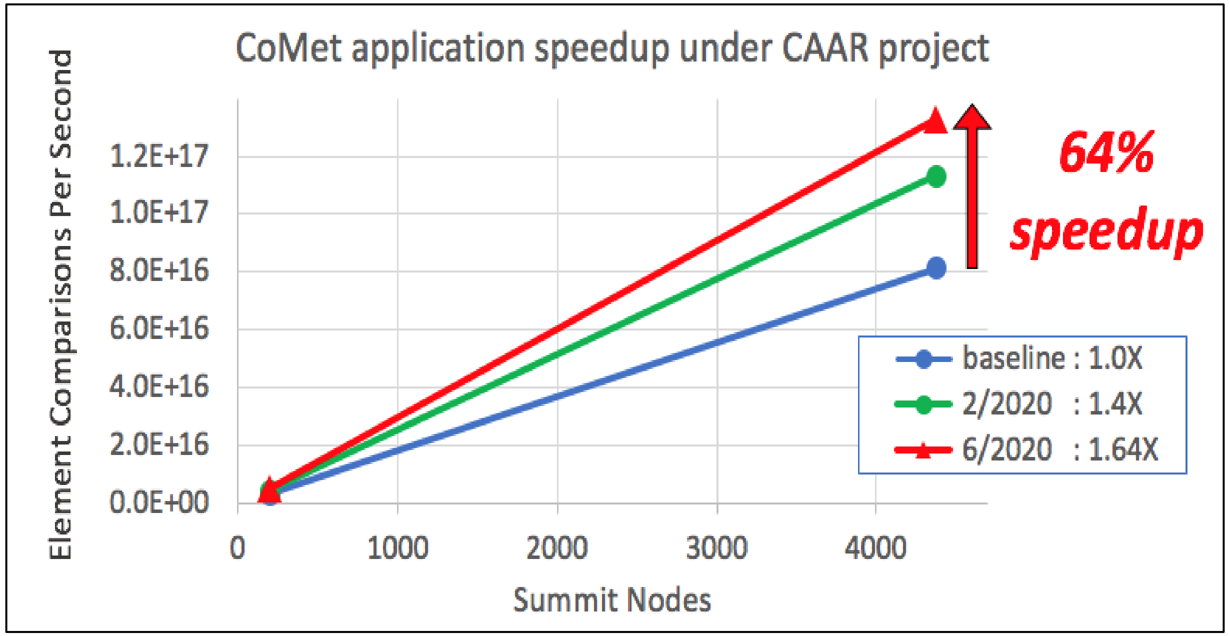
1. Modify CoMet to run on AMD GPUs, using HIP, ROCm, rocBLAS and other libraries
2. Implement several algorithmic changes to improve performance
3. Added improvements made in recent Crusher hackathon

Porting to HIP: CoMet programming model

- CoMet already has most CUDA dependencies in wrapper functions, thus easy to use #ifdefs in a few places to alternatively enable HIP or rocM calls
- Using hipMalloc etc. but not using HIP wrappers for CUDA calls
- cpp macro to abstract kernel launch syntax
- GEMM call:
  - #ifdefs for cuBlasGemmEx and rocblas_gemm_ex since arg lists are a little different
  - C++ traits classes are used to support differences
- Build system: use custom code to make CMAKE and HIP work together early on, now CMAKE+HIP is better supported

Algorithmic changes

- Moved calculation of "X" matrix to GPU – faster computation, less transfers
- Algorithmic change to compute 2 GEMMs instead of 3, gives identical result
- Lossless compression for sparsified matrix on GPU enables much larger problems, lower transfer costs
  - uses CUDA CUB and rocM rocPRIM libraries – identical calling sequences so seamless port

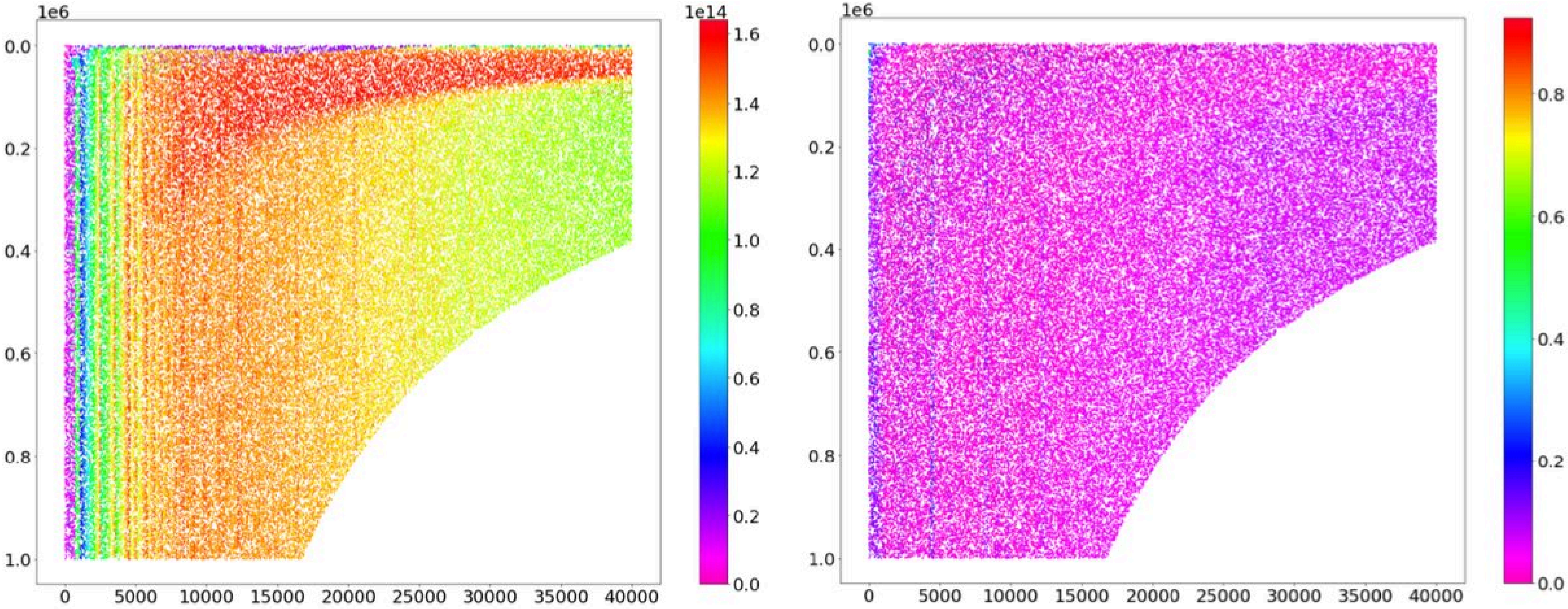


GEMM performance

- > 90% of runtime is spent in mixed FP16/FP32 GEMM
- the needed rocBLAS function originally did not use matrix-matrix instructions, thus ran ~ half speed; now fixed (still waiting on INT8/INT32 GEMM to use matrix-matrix instructions)
- FP16/FP32 GEMM running per GCD ~ 145 TOps (typical large sizes), sometimes ~ 160 TOps (a few large sizes)
- Variability in GEMM performance as a function of matrix size has been an issue – AMD is working on tuning GEMM operations to improve performance

GEMM performance results – Crusher, single GCD

- heat map for different choices of matrix dimensions m (horizontal), k (vertical)
- white spots are image background (no run data)
- curved envelope is memory limitation
- note k range is much wider, is visually compressed in the graph
- TOp rate (left) (160+ TF max), run-to-run variability range (right) (usu. < 10%)



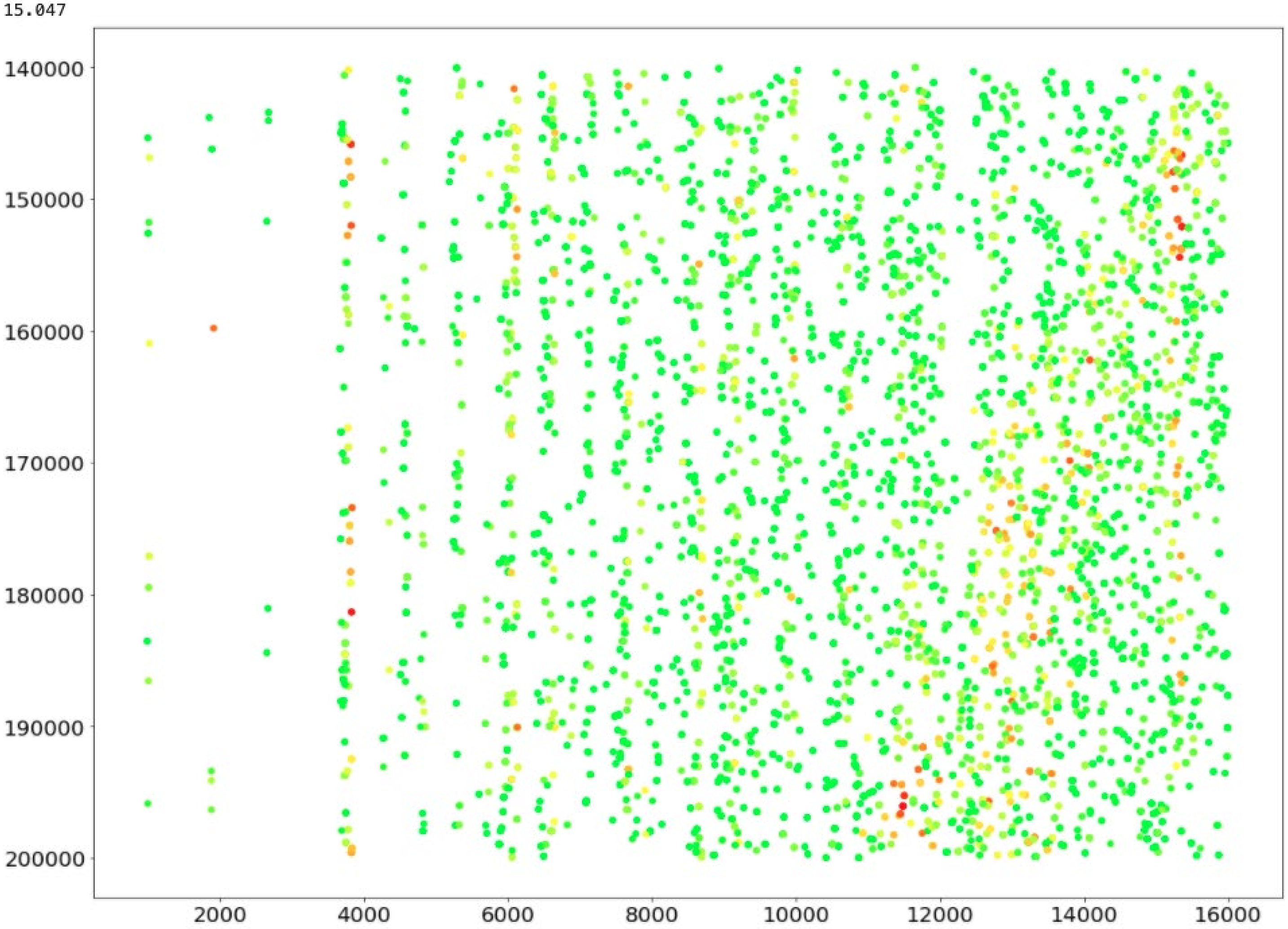
(a) (left) Crusher performance, best trial of 6; (right) Crusher, relative error across trials

rocBLAS 4.2 performance – detail - Spock

rocblas_gemm_ex, mixed FP16/FP32, (N,T) config

green = 145 TOps, red = 160 TOps

is higher performance possible across more m,k values



Other changes: OpenMP support

- Needed in order to speed up a few computations on the GPU
- Implemented CPU-side OpenMP threading at Crusher Hackathon
  - CPU-side OpenMP is supported under rocm/4.5.2
  - available under hipcc (no need for mixing compliers), needed to add this to link step:

```
-Wl,-rpath,$ROCM_PATH/llvm/lib $ROCM_PATH/llvm/lib/libomp.so
```

Other changes: SLURM adjustments

- Reduced unit test runtime from > 8 hours to 11 min.
  - now using proper `srun` bindings:

```
env OMP_NUM_THREADS=2 srun -N2 -n64 --cpus-per-task=2 \
        --ntasks-per-node=32 --gpu-bind=closest --gpus-per-node=8
```

Other changes: MAGMA library

- Modified pseudo-GEMM operation needed for some CoMet methods
- Was deployed under CUDA using modified MAGMA kernels
- (hip)MAGMA build was broken for some time, now building correctly
  - rocm/4.5.2 fixed the insufficient registers problem
- The MAGMA kernel being used is not well-optimized for MI250X, another 2X performance may be possible – working with AMD on this –

Other findings

- the `srun` flag “`-u`” causes screen output to be unbuffered, gives better sense of what parts of the run are taking more, time if your code writes output
- the gcc/clang optimization flag “`-freciprocal-math`” can cause small roundoff-level differences, this may matter in some contexts
- rocprof was useful (and easy) for getting profile data from HIP kernels
- HIP kernels with printf (e.g., for assertions) still can take a long time to compile, looking forward to improvements

Future work

- update CoMet build system to use up-to-date version of MAGMA
- experiment with `amdclang` and `CC` (Cray) compilers/wrappers instead of `hipcc`
- experiment with GPUDirect for faster communication
- scaling studies on full Frontier when available

Questions?

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.