GenTen Performance Portable Dense TTM Kernels

TDAlab

Benjamin Cobb and Ümit V. Çatalyürek

Collaborators: Eric Phipps and Hemanth Kolla (Sandia National Laboratories)

Georgia Tech College of Computing
Computational Science and Engineering

Introduction

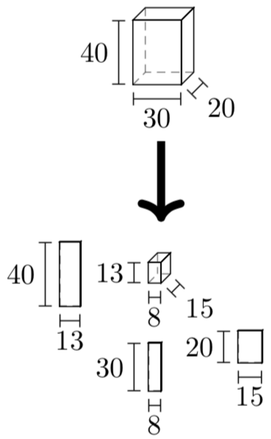
The tensor times matrix (TTM) kernel is one of the computational bottlenecks of the Sequentially Truncated-Higher Order Singular Value Decomposition (ST-HOSVD)[7].

Primary contributions:

- We present several TTM implementations that utilize the Kokkos [2] programming model to enable performance portability.
- We compare our implementations to state of the art TTM implementations.
- We show that our TTM implementations attain DGEMM like performance.

Motivation: ST-HOSVD

Sequentially Truncated Higher Order Singular Value Decomposition (ST-HOSVD) compresses a tensor into a smaller core tensor and series of factor matrices. The core tensor can be multipled by the factor matrices to recover an approximation of the original tensor.



ST-HOSVD of 3'rd order $40 \times 30 \times 20$ tensor

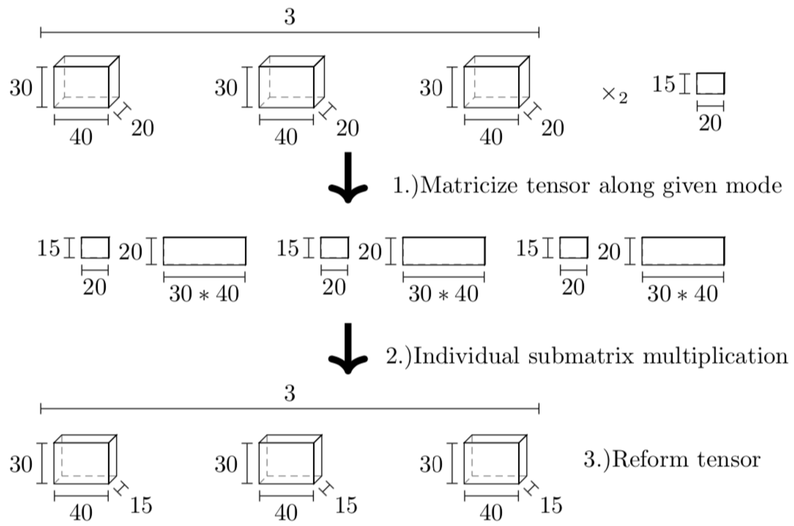
Overview

Given that $\mathcal{X}$ is a tensor of order N with dimension sizes: $I_0 \times \cdots \times I_{N-1}$ and $\mathcal{U}$ is a matrix of size $J \times I_n$, then $\times_n$ denotes a TTM along the $n'th$ dimension (mode) of $\mathcal{X}$:

$$\mathcal{Y} = \mathcal{X} \times_n \mathcal{U} \tag{1}$$

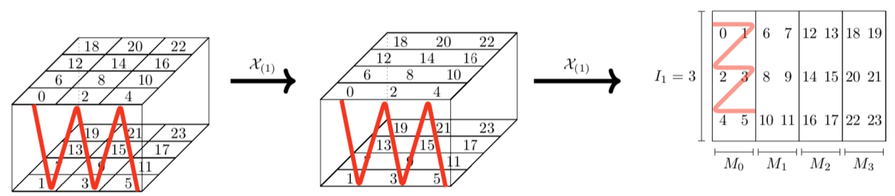
Where $\mathcal{Y}$ is the resulting tensor with dimensions: $I_0 \dots I_{n-1} \times J \times I_{n+1} \dots I_{N-1}$

Tensor Times Matrix



$30 \times 40 \times 20 \times 3$ tensor times 15×20 matrix along mode 2

Matricization

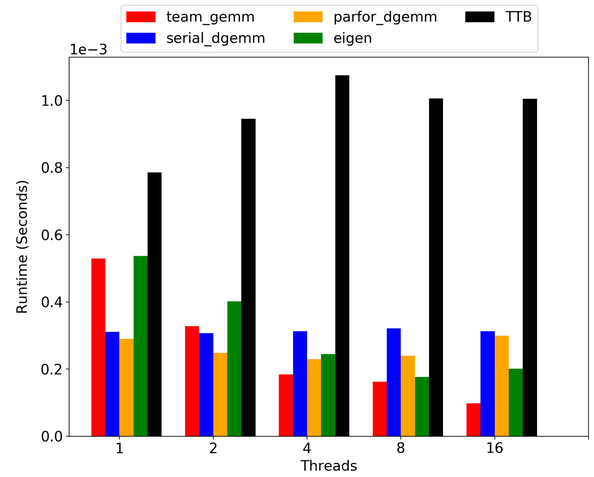


Matricization of $2 \times 3 \times 4$ tensor along mode-1

Implementations

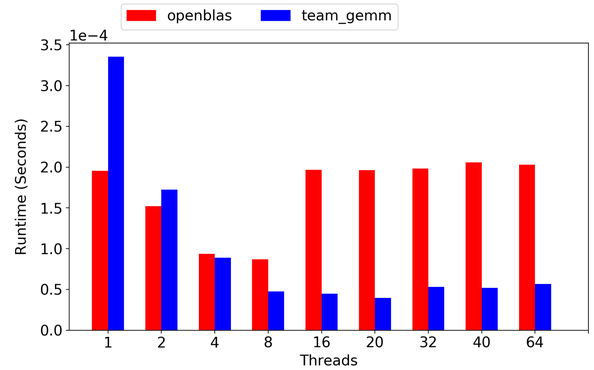
- **team_gemm**: proposed Kokkos based portable implementation. Utilizes KokkosKernels' TeamGemm function and Kokkos hierarchical parallelism. Is highly optimized for small problem sizes. Runs on both CPU and GPU.
- **parfor/serial dgemm**: proposed implementations that uses linked BLAS installation such as MKL [1] or OpenBLAS[8]. Uses either KokkosKernels, TeamGEMM or cuBLAS to port to GPU.
- **cuBLAS**[5]: strided batched cuBLAS function. Only runs on CUDA enabled GPUs.
- **TTB**[4]: Matlab Tensor Toolbox.
- **Eigen**[3]: C++ template library utilized by Google's TensorFlow.

Results



$(16 \times 16 \times 16 \times 16) \times_2 (16 \times 16)$ on Intel Xeon-2683

- Proposed implementations consistently outperform and outscale Eigen and TTB implementations for targeted problem size.



$(16 \times 16) \times (16 \times 16^3)$ vs $(16 \times 16 \times 16 \times 16) \times_1 (16 \times 16)$ on IBM Power9

- team_gemm outperforms OpenBLAS DGEMM of comparable size.

Mode	0	1	2	3
TeamGemm	0.00020	0.00011	0.00018	0.00011
cuBlas	0.00015	0.00018	0.00004	0.00003

Table: $(16 \times 16 \times 16 \times 16)$ TTM Kepler GPU times in Seconds

- TeamGEMM implementation ports to GPU.

References

[1] Intel math kernel library users manual, 2020.

[2] H. C. Edwards, C. R. Trott, and D. Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns.

[3] G. Guennebaud, B. Jacob, et al. Eigen v3.

[4] T. G. Kolda and B. W. Bader. Tensor decompositions and applications.

[5] NVIDIA Corporation. *cuBLAS documentation*.

[6] E. T. Phipps and T. G. Kolda. Software for sparse tensor decomposition on emerging computing architectures.

[7] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen. A new truncation strategy for the higher-order singular value decomposition.

[8] Z. Xianyi, M. Kroeker, W. Saar, W. Qian, Z. Chothia, C. Shaohu, and L. Wen. Openblas, 2020.