A ARTIFACT DESCRIPTION: [STRASSEN'S ALGORITHM FOR TENSOR CONTRACTION]

A.1 Abstract

The artifact contains all components of the implementations and benchmarks of Strassen's algorithm for tensor contraction [1].

A.2 Description

A.2.1 Check-list (artifact meta information).

- **Algorithm:** **ABC Strassen**, **AB Strassen**, **Naive Strassen**, and TBLIS tensor contraction [2].
- **Program:** `C99`, `C++11`, `MPI`, `AVX inline assembly`
- **Compilation:** MPI C/C++ compiler (C99/C++11 capable); we use icc/icpc version 15.0.3 with the gcc 4.9.3 Standard Template Library for single node experiment; mpicc/mpicxx in mvapich2/2.1 for distributed memory experiment.
- **Data set:** Sythetic dataset and Tensor Contraction Benchmark.
- **Hardware:** Maverick supercomputer at TACC.
- **Output:** *Effective* GFLOPS performance, as shown ① in Fig. 4.
- **Experiment workflow:** Decompress the source code, configure and compile the code, check the dataset and environment, and execute the code.
- **Publicly available?:** Yes.

A.2.2 How software can be obtained (if available). The code can be downloaded from Github: https://github.com/flame/tblis-strassen

A.2.3 Hardware dependencies. x86-64 processor that supports `AVX` instructions.

A.2.4 Software dependencies. Linux or macOS/OS X, `autotools`, Intel icc/icpc compiler 15 or later and compatible GCC 4.9 or later, mvapich2/2.1 for `MPI`, `OpenMP` environment, SLURM queuing system (for distributed experiments—the details of the TACC Maverick job queues are assumed in the provided scripts).

A.2.5 Datasets. Synthetic data and some representative use cases from Tensor Contraction Benchmark [3].

A.3 Installation

Follow the build instructions in `README.md`, found in the source code top folder after extracting the `tblis-strassen.tar.gz` from the URL above.

A.4 Experiment workflow

- Download `tblis-strassen.tar.gz` from the URL above.
- Compile the source code: Replace `$INSTALL_PATH` with the installation path the user wants to specify.

```
$ tar -zxvf tblis-strassen.tar.gz
$ cd tblis-strassen
$ autoreconf -i
$ ./configure --prefix=$INSTALL_PATH \
    --disable-shared --enable-config=sandybridge
$ make && make install
```

- Set up environment variables: Replace `$core_num` with the number of cores the user wants to run.

```
$ export OMP_NUM_THREADS=$core_num
$ export KMP_AFFINITY=compact
```

Note: if hyper-threading is enabled, the following alternative must be used:

```
$ export KMP_AFFINITY=compact,1
```

- Single node experiment (synthetic tensor contractions): Replace `$NIm` with N_{I_m}, `$NJn` with N_{J_n}, `$NPk` with N_{P_k}, `$niter` with the iterations the user wants to run, `$L` with number of levels of STRASSEN to employ ($L = 0$ will invoke regular TBLIS TC routine), and `$IMPL` with the implementations the user wants to test. `$IMPL` has the following options, [1: **ABC Strassen**, 2: **AB Strassen**, 3: **Naive Strassen**].

```
$ cd bin
$ ./test_strassen -m $NIm -n $NJn -k $NPk \
    -niter $niter -level $L -impl $IMPL
```

We have prepared the testing scripts in `bin` folder so the user can run it directly to reproduce the experiments in this paper (e.g. `square_run_1core.sh`).

- Single node experiment (real-world benchmark):

```
$ cd bin
$ ./test_strassen -file $benchmark_filename
```

We have prepared the benchmark from [3] as `tcb.txt` and the test scripts (e.g. `tcb_run_1core.sh`) to make it easier for the user to reproduce the result presented in this paper.

- Single node experiment (shape-dependence experiment):

```
$ cd bin
$ ./test_strassen -file $shape_filename
```

We have prepared the test cases we use as `shape.txt` and the test scripts (e.g. `shape_run_1core.sh`) to make it easier for the user to reproduce the result presented in this paper.

- Distributed memory experiment: Replace `$P` with the edge length of the $P \times P$ mesh the user wants to test on.

```
$ cd dist
$ make
$ sbatch ./submit_${P}x${P}.sh
```

A.5 Evaluation and expected result

The output will include the following components:

- Input problem size.
- Running time (in seconds).
- Total GFLOPS (floating point operation number).
- *Effective* GFLOPS (① in Fig. 4).
- Accuracy ($\frac{\|\mathcal{T}-\mathcal{T}_{ref}\|_2}{\|\mathcal{T}_{ref}\|_2}$).

The user can compare the relative *Effective* GFLOPS for different implementations. The trend should match the performance curves shown in this paper. Since the machines may be different from ours, the absolute GFLOPS could be different.

A.6 Experiment customization

The format information for the user specified benchmark can be found in `README.md`. For instance, to evaluate the example $\mathcal{C}_{a,b,c} \mathrel{+}= \mathcal{A}_{d,c,a} \cdot \mathcal{B}_{d,b}$ in Figure **??**, the user can specify the following input:

```
abc dca db & a:4;b:8;c:2;d:8;
```

REFERENCES

[1] Jianyu Huang, Devin A. Matthews, and Robert A. van de Geijn. 2017. *Strassen's Algorithm for Tensor Contraction.* FLAME Working Note #84, TR-17-02. The University of Texas at Austin, Department of Computer Science. https://apps.cs.utexas.edu/apps/sites/default/files/tech_reports/TensorStrassen.pdf

[2] Devin A Matthews. 2016. High-performance tensor contraction without Transposition. *CoRR* abs/1607.00291 (2016).

[3] Paul Springer and Paolo Bientinesi. 2016. Tensor Contraction Benchmark v0.1. Available online. (December 2016). https://github.com/hpac/tccg/tree/master/benchmark