

APPENDIX A

ARTIFACT DESCRIPTION APPENDIX: IMPLEMENTING EFFICIENT DATA COMPRESSION AND ENCRYPTION IN A PERSISTENT KEY-VALUE STORE FOR HPC

A. Abstract

This artifact contains all components of our framework (we will call it the framework) presented in the SC 2018 poster titled Implementing Efficient Data Compression and Encryption in a Persistent Key-Value Store for HPC. It also includes scripts to run the framework on CSCS's Grand Tavé and NERSC's Cori.

B. Description

1) Check-list (artifact meta information):

- **Algorithm:** Key-Value Store.
- **Program:** C/C++ and MPI.
- **Compilation:** Intel Compiler 17.0.4 with Cray MPICH 7.6 in Grand Tavé, and Intel C Compiler 18.0.1 with Cray MPICH 7.6 in Cori.
- **Data set:** The Silesia compression corpus.
- **Run-time environment:** SUSE Linux Enterprise Server 12 in Grand Tavé and Cori.
- **Hardware:** CSCS's Grand Tavé and NERSC's Cori.
- **Output:** Average, minimum, and maximum of the total execution time. Database sizes.
- **Experiment workflow:** Download package, compile the source code and test applications, run the applications and check the outputs.
- **Publicly available?:** No.

2) *How software can be obtained (if available):* The framework can be cloned from our property Git server. The available URL will be provided if the poster is accepted.

3) *Hardware dependencies:* CSCS's Grand Tavé and NERSC's Cori. However, the framework will run on any general clusters equipped with Intel KNLs and/or NVM-based burst buffers.

4) *Software dependencies:* C++ compiler that supports C++11 standard. MPI library that supports a multi-threading level of MPI_THREAD_MULTIPLE. CMake. Make.

5) *Datasets:* Datasets are included with the package.

C. Installation

1) Clone the framework and set an environment variable.

```
$ cd $HOME
$ git clone https://$FRAMEWORK_GIT_SERVER/kvs.git
$ export KVS=$HOME/kvs
```

2) Build the framework with CMake and Make.

```
$ cd $KVS
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$KVS/install ..
$ make install
```

D. Experiment workflow

1) Build a microbenchmark application (kvs) in \$KVS/kv/apps/sc18 directory.

```
$ cd $KVS/kv/apps/sc18
$ make
```

2) kvs application is used in Fig. 3, Fig. 4, and Fig. 5. It takes five arguments as input dataset file path, length of keys, length of values, iteration count, and flag to force SSTables flushing. The scripts used in Grand Tavé and Cori are included in the package. For example, you can run the application to reproduce the results in Fig. 3 and Fig. 4 using Grand Tavé with the following script:

```
#!/bin/bash

#SBATCH --job-name=TAVE-KVS
#SBATCH --nodes=16
#SBATCH --ntasks-per-core=1
#SBATCH --cpus-per-task=1
#SBATCH --constraint=flat,quad
#SBATCH --partition=normal

NRANKS=(1 2 4 8 16 32 64 128 256 512 1024)
KEYLEN=32
VALLEN=(512 131072)
COUNT=(100000 1000)
FLUSH=1

run () {
for i in "${VALLEN[@]"; do
let c=COUNT[0]
if [ $i -eq 131072 ]; then
let c=COUNT[1]
fi
for j in "${NRANKS[@]"; do
echo "= VALLEN[$i] NRANKS[$j] COUNT[$c] ="
srun -n $j --ntasks-per-node=64 \
-N $((($j/64)<1?1:$((j/64)))) \
./kvs silesia.tar $KEYLEN $i $c $FLUSH
done
done
}

echo "***** TAVE *****"
export MPICH_MAX_THREAD_SAFETY=multiple
export PKV_REPO=$SCRATCH/pkv

echo "***** PKV *****"
export PAPYRUSKV_WRITEBUFFER=0
export PAPYRUSKV_MCDRAM=0
export PAPYRUSKV_COMPRESSION_B=0
run

echo "***** PKV+WB *****"
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_MCDRAM=0
export PAPYRUSKV_COMPRESSION_B=0
run

echo "***** PKV+WB+HBM *****"
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_MCDRAM=1
export PAPYRUSKV_COMPRESSION_B=0
run

echo "***** PKV+WB+HBM+LZ4 *****"
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_MCDRAM=1
export PAPYRUSKV_COMPRESSION_B=2
run

echo "***** PKV+WB+HBM+LZ4HC *****"
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_MCDRAM=1
export PAPYRUSKV_COMPRESSION_B=4
run

echo "***** PKV+WB+HBM+ZSTD *****"
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_MCDRAM=1
export PAPYRUSKV_COMPRESSION_B=7
run
```

You can run the application to reproduce the results in Fig 5. using Cori with the following script:

```
#!/bin/bash
```

```

#SBATCH -J CORI-KVS
#SBATCH -C haswell
#SBATCH -N 32
#DW jobdw capacity=2TB access_mode=striped \
    type=scratch pool=sm_pool

NRANKS=(1 2 4 8 16 32 64 128 256 512 1024)
KEYLEN=32
VALLEN=(512 131072)
COUNT=(100000 1000)
FLUSH=2

run () {
for i in "$VALLEN[@]"; do
    let c=COUNT[0]
    if [ $i -eq 131072 ]; then
        let c=COUNT[1]
    fi
    for j in "$NRANKS[@]"; do
        echo "= VALLEN[$i] NRANKS[$j] COUNT[$c] ="
        srun -n $j --ntasks-per-node=32 \
            -N $((($j/32)<1?1:$((j/32)))) \
            ./kvs silesia.tar $KEYLEN $i $c $FLUSH
    done
done
}

echo "***** CORI *****"
export MPICH_MAX_THREAD_SAFETY=multiple
export PAPYRUSKV_MCDRAM=0
export PKV_REPO=$DW_JOB_STRIPED/pkv

echo "***** PKV *****"
export PAPYRUSKV_ENCRYPTION_D=0
export PAPYRUSKV_WRITEBUFFER=0
export PAPYRUSKV_COMPRESSION_B=0
run

echo "***** PKV+WB *****"
export PAPYRUSKV_ENCRYPTION_D=0
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_COMPRESSION_B=0
run

echo "***** PKV+WB+LZ4 *****"
export PAPYRUSKV_ENCRYPTION_D=0
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_COMPRESSION_B=2
run

echo "***** PKV+WB+LZ4HC *****"
export PAPYRUSKV_ENCRYPTION_D=0
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_COMPRESSION_B=4
run

echo "***** PKV+WB+ZSTD *****"
export PAPYRUSKV_ENCRYPTION_D=0
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_COMPRESSION_B=7
run

echo "***** PKV+WB+CBC *****"
export PAPYRUSKV_ENCRYPTION_D=2
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_COMPRESSION_B=0
run

echo "***** PKV+WB+ZSTD+CBC *****"
export PAPYRUSKV_ENCRYPTION_D=2
export PAPYRUSKV_WRITEBUFFER=1
export PAPYRUSKV_COMPRESSION_B=7
run

```

E. Evaluation and expected result

The evaluation results can be found in the output log file. It contains the average, minimum, and maximum of total execution times for all MPI ranks in the application and total amount of MemTables and SSTables sizes.