UniSparse: An Intermediate Language for General Sparse Format Customization

Introduction

UniSparse is an intermediate language and compiler that provides a unified abstraction for representing and customizing sparse formats. Compared to prior sparse linear algebra compilers, UniSparse decouples the logical representation of the sparse tensor (i.e., the data structure) from its low-level memory layout, enabling the customization of both. UniSparse improves over current programming models that only provide limited support for customized sparse formats.

This repository implements UniSparse as an independent dialect on top of the MLIR infrastructure. The UniSparse dialect allows users to declaratively specify format conversion and compute kernels using UniSparse format encodings. The compiler tool automatically lowers the program and generates format conversion routines and sparse linear algebra kernels.

This artifact supports the evaluation claim in the paper #18 of OOPSLA'24:

- The format conversion and compute kernels generated by UniSparse achieve performance that matches prior MLIR SparseTensor and TACO compilers, while provides a broader coverage for handling a wider range of custom formats. (Section 7.3 and 7.4)

In the Step-by-Step Instructions section below, we provide instructions for 2 experiments. Experiment 1 and 2 reproduces the performance numbers in Section 7.3 and 7.4 respectively, which support the claim. We omit the experiments in Section 7.2 as they require multi-core CPUs and GPUs. We provide fewer datasets for each experiment than the paper presents, as more datasets make the docker image big and slow down the set up time.

Hardware Dependencies

To run experiments 1 and 2, an Intel CPU will work.

Getting Started

We first pull a docker image from dockerhub:

```
$docker pull sibylau/mlir-llvm:oopsla24-ae
```

Note that this docker image is 14.5GB and it may take time to download it.
Then we run a container from this docker image:

```
$docker run -it --entrypoint bash sibylau/mlir-llvm:oopsla24-ae
```

Inside this container, we clone the UniSparse repo:

```
$git clone https://github.com/cornell-zhang/UniSparse.git -b oopsla24-ae
```

Source the bash file under the UniSparse project directory path:

```
$cd UniSparse && source script/build.sh
```

Please also export environment variable

```
$export LD_LIBRARY_PATH=/install/taco/build/lib:$LD_LIBRARY_PATH
```

Please try to run `$cd evaluation/KernelGeneration && bash run.sh` and let us know if there are any issues.

Step-by-Step Instructions

Experiment 1

In the UniSparse project, go to
`$cd evaluation/FormatConversion`
where we run the first experiment.
Run `$bash run.sh` will compile and run format conversion programs generated by UniSparse and two baselines -- MLIR SparseTensor and TACO. The command line output shows execution time in seconds for each format conversion kernel and dataset.

The pre-built executables can be found in `evaluation/FormatConversion/executables`, and the expected output is shown in `evaluation/FormatConversion/output.log`. According to the results, we can found that the execution time of format conversion kernels generated by UniSparse is comparable or better than two baselines, while UniSparse supports automated conversion for more custom formats, such as DCSC to BCSR, CSB to DIA_Variant, COO to C2SR, and COO to CISR, as claimed in the paper.

Experiment 2

In the UniSparse project, go to
`$cd evaluation/KernelGeneration`
where we run the second experiment.
Run `$bash run.sh` will compile and run sparse linear algebra kernels generated by UniSparse and two baselines -- MLIR SparseTensor and TACO. The command line output shows execution time in seconds for each kernel and dataset.

The pre-built executables can be found in `evaluation/KernelGeneration/executables`, and the expected output is shown in `evaluation/KernelGeneration/output.log`. According to the results, we can found that the execution time of sparse linear algebra kernels generated by UniSparse is comparable or better than two baselines. Specifically, kernels generated by UniSparse achieve similar performance to those generated by MLIR SparseTensor, as UniSparse reuses kernel generation passes of MLIR SparseTensor. While TACO generated kernels are typically slower than UniSparse and MLIR SparseTensor. Note that all kernels are using double precision and execute in a single thread. As mentioned in the rebuttal, the CSC_CSC_CSC_SpGEMM kernel generated by TACO is not functionally correct, and therefore, we do not include the results for evaluation here.

Reusability Guide

In the UniSparse project, go to
`$cd evaluation/Reusability`
where we run the reusability experiment.
Run `$bash run.sh` will compile and run format conversion and compute kernels generated by UniSparse with other formats not presented in the paper. The command line output shows execution time in seconds for each kernel and dataset.

The pre-built executables can be found in `evaluation/Reusability/executables`, and the expected output is shown in `evaluation/Reusability/output.log`. We show 3 examples of kernels generated by UniSparse. One is automatically converting from CSR to DIA variant format, another is SpMM using CSC format, and the last is CSR_CSC_CSC_SpGEMM kernel. These three kernels demonstrate that UniSparse can be reused to generate format conversion kernels with more source and target formats, and compute kernels with a diverse combination of formats.