Towards a better E2E testing framework

Lixun Zhang

August 16, 2022

The Problem and History

MLIR#408: FP16 xdlops tolerance too high?

- The tolerance for fp16 tests is set to 15%
- For other datatypes, the tolerance is set to 0.0001%

PR#694: Changed verification threshold for fp16 to 0.0001%

- All fp16 tests passed in PR CI
- Failed in Nightly CI $\Rightarrow$ failures happened in the Random Tests stage

PR#696: Bump fp16 tolerance to 25%

- We could not even get back to 15%
- However, later it turned out all tests passed with 10%

Why do some fp16 E2E tests fail?

Experiment Setup

Sources of configs: tests from auto_e2e/, e2e_for_pr/, and misc_e2e/

- Tests in e2e_for_pr/ are enabled as fixed tests with GPU validation in PR CI
- Tests in auto_e2e/ are set as random tests with CPU validation in the Nightly CI
- Tests in auto_e2e/ and misc_e2e/ are also set as fixed tests with GPU validation in the Nightly CI

Settings of the test

- `-rand 1`: Generate random numbers as inputs
- `-rand_min` and `-rand_max`: choose the range of the random numbers
- `-pv` and `-pv_with_gpu`: choose the validation method
- `-x2`: enable xdlops

Hardware

- MI100 gfx908 ixt-rack-112
- MI200 gfx90a lockhart5

How to measure the difference between two outputs (val vs. kern)

Element wise difference

- Absolute difference $\text{maxAbsDiff} = \max_i(|\text{val}_i - \text{kern}_i|)$
- Relative difference (ignore zero denominators)

$$\text{maxRelDiff} = \max_i\left(\frac{|\text{val}_i - \text{kern}_i|}{|\text{val}_i|}\right), \ |\text{val}_i| > 0$$

- Relative difference (ignore small denominators)

$$\text{maxRelDiff}_{\text{old}} = \max_i\left(\frac{|\text{val}_i - \text{kern}_i|}{|\text{val}_i|}\right), \ |\text{val}_i| > 1 \times 10^{-3}$$

- Epsilon difference

$$\text{maxEpsilonDiff} = \max_i\left(\frac{|\text{val}_i - \text{kern}_i|}{\epsilon_{|\text{val}_i|}}\right)$$

where $\epsilon_{|\text{val}_i|}$ is the precision of fp16 numbers around val_i.

How to measure the difference between two outputs (val vs. kern)

RMS: the root mean square difference between two data set A and B

$$\frac{\sqrt{\sum_{i=1}^{N}(a_i - b_i)^2}}{\sqrt{N} \cdot \max(\max(|a_i|), \max(|b_i|))}$$

Experiment Results and Observations

- Four random number ranges are chosen:
  - r0: [-1, 1]
  - r1: [-10, 10]
  - r4: [1, 5]
  - r5: [5, 10]
- Numbers in the following tables are aggregated among all tests for all settings

Experiment Results and Observations — Input number magnitude

	r0	r1
maxEpsilonDiff ave	314.24	1,627.64
maxEpsilonDiff max	2,030.33	23,316.33
maxAbsDiff ave	0.15	3
maxAbsDiff max	5.67	26.67
maxRelDiff ave	4.42	2.79
maxRelDiff max	111.33	105.67
maxRelDiff old ave	0.02	0.1
maxRelDiff old max	0.1	2.2
RMS ave	2.12E-06	1.31E-06
RMS max	1.55E-05	6.57E-06

The left table lists aggregated results on MI200

- In general, errors with r1 ([-10, 10]) is larger than that of r0
- Exceptions are maxRelDiff and RMS because the denominators are smaller with r0

Experiment Results and Observations — Underflow and overflow

	r0	r4	r5
maxEpsilonDiff ave	314.24	0.73	0.72
maxEpsilonDiff max	2,030.33	1	1
pass rate	27.73%	100%	97.82%
maxAbsDiff ave	0.15	2.67	4.99
maxAbsDiff max	5.67	32	32
maxRelDiff ave	4.42	5.64E-04	4.69E-04
maxRelDiff max	111.33	9.80E-04	9.80E-04
maxRelDiff old ave	2.06E-02	5.64E-04	4.69E-04
maxRelDiff old max	9.50E-02	9.80E-04	9.80E-04
RMS ave	2.12E-06	5.34E-06	3.21E-06
RMS max	1.55E-05	2.47E-05	1.25E-05

The left table lists aggregated results on MI200

- In general, errors with r0 ([-1, 1]) is larger than that of r4 ([1, 5]) and r5 ([5, 10])
- Exceptions are maxAbsDiff and RMS because the magnitude of inputs with r4/r5 are lager
- A test passes if maxEpsilonDiff ≤ 1
- All tests with r4 pass since the maximum maxEpsilonDiff is 1
- However, some tests with r5 fail even the maximum maxEpsilonDiff is 1 $\Rightarrow$ overflow happens

Experiment Results and Observations — Validation methods

	xdlops+cpu	xdlops+gpu	nonxdlops+cpu
maxEpsilonDiff ave	352.98	818.53	285.98
maxEpsilonDiff max	3,959.25	9,306.00	5,746.25
maxAbsDiff ave	3.13	3.76	1.21
maxAbsDiff max	24.13	28.00	20.13
maxRelDiff ave	2.01	2.82	0.57
maxRelDiff max	68.5	81.50	12.75
maxRelDiff old ave	2.45E-02	4.96E-02	1.72E-02
maxRelDiff old max	3.20E-01	8.43E-01	5.60E-01
RMS ave	2.75E-06	4.57E-06	1.67E-06
RMS max	1.26E-05	2.03E-05	1.16E-05

The left table lists aggregated results on MI200

- cpu and non-xdlops gpu are closest to each other
- cpu and xdlops gpu have medium difference
- xdlops gpu and non-xdlops gpu have the largest difference

cpu: sequential implementation of the convolution operation
gpu non-xdlops: convert convolution to gemm and compute gemm without `mfma`
gpu: convert convolution to gemm and compute gemm with `mfma`

Experiment Results and Observations — MI200 vs. MI100

`mfma` instructions on MI200 flush denormalized numbers to zero

	MI200(r0)	MI100(r0)	MI200(r1)	MI100(r1)	MI200(r4)	MI100(r4)	MI200(r5)	MI100(r5)
maxEpsilonDiff ave	314.24	90.50	1,627.64	1,757.24	0.73	0.72	0.72	0.72
maxEpsilonDiff max	2,030.33	1,739.00	23,316.33	40,028.00	1.00	1.00	1.00	1.00
maxAbsDiff ave	0.15	0.15	3.00	2.95	2.67	2.67	4.99	4.99
maxAbsDiff max	5.67	5.67	26.67	26.67	32.00	32.00	32.00	32.00
maxRelDiff ave	4.42	1.70	2.79	2.68	5.64E-04	5.63E-04	4.69E-04	4.68E-04
maxRelDiff max	111.33	34.00	105.67	92.67	9.80E-04	9.80E-04	9.80E-04	9.80E-04
maxRelDiff old ave	0.02	0.01	0.10	0.10	5.64E-04	5.63E-04	4.69E-04	4.68E-04
maxRelDiff old max	0.10	0.09	2.20	2.22	9.80E-04	9.80E-04	9.80E-04	9.80E-04
RMS ave	2.12E-06	1.80E-06	1.31E-06	1.31E-06	5.34E-06	5.33E-06	3.21E-06	3.21E-06
RMS max	1.55E-05	1.55E-05	6.57E-06	6.77E-06	2.47E-05	2.47E-05	1.25E-05	1.25E-05

- With r4 and r5, MI200 and MI100 behave the same
- With r0, MI100 has smaller errors
- With r1, MI200 and MI100 behave similarly

Experiment Results and Observations — Exploration of a "better" validation function

relDiff old	original	accumulate 4	flush subnormals
0	801676(99.858000%)	802015(99.900226%)	802369(99.944321%)
(0,1e-6)	0(0.000000%)	0(0.000000%)	0(0.000000%)
[1e-6, 1e-5)	0(0.000000%)	0(0.000000%)	0(0.000000%)
[1e-5, 1e-4)	0(0.000000%)	0(0.000000%)	0(0.000000%)
[1e-4, 1e-3)	970(0.120825%)	704(0.087691%)	364(0.045340%)
[1e-3, 1e-2)	60(0.007474%)	60(0.007474%)	0(0.000000%)
[1e-2, 0.1)	3(0.000374%)	3(0.000374%)	0(0.000000%)
[0.1, 1)	0(0.000000%)	0(0.000000%)	0(0.000000%)
>= 1	0(0.000000%)	0(0.000000%)	0(0.000000%)
cpuVal == 0	0(0.000000%)	0(0.000000%)	0(0.000000%)
max	0.027	0.028	0.00097
cpuVal	-0.001073837	-0.001074791	0.125244141
gpuVal	-0.001045227	-0.001045227	0.125366211

The left table shows the histogram of maxRelDiff_old of the worst test on MI200

- `accumulate 4`: cpu truncates the sum of every 4 multiplications from fp64 to fp32 $\Rightarrow$ slightly improves the results
- `flush subnormals`: cpu flushes small numbers ($< 2^{-14}$) to zero $\Rightarrow$ greatly improves the results

Summary

Causes of numerical errors for fp16 tests:

1. Subnormal numbers flushed to zero
2. Computation order difference due to partition of gemm onto the grid
3. Truncation behavior difference during accumulation of intermediate results

Solutions or workarounds

- Avoid subnormal numbers by choosing a random range away from 0
- Use RMS as the metric

Proposals

1. MLIR#620 Apply both RMS and element wise metrics in the verification function
2. MLIR#621 Let each test decide how it is passed
   - choose which metric to use
   - choose the tolerance for each metric
3. MLIR#619 Auto generate E2E tests at configure or build time
   - Organize all configs in the same `.toml` file
   - For each config, generate tests for all data types, directions, and layouts.
   - The MLIR repository only maintains the `.toml` file and the generation script but does not maintain the generated tests.
   - Stop using fixed tests, all tests use `-rand 1`
   - Check all tests in the Nightly CI and select a subset for the PR CI. Adjust the metrics and their tolerance based on the selected tests.