GlobalISel CI/CD

Authors:
Dhruv Chawla
Madhur Amilkanthwar

Presented by:
Neil Hickey

Agenda

- Introduction and Features
- Internals
- Future Directions

Introduction

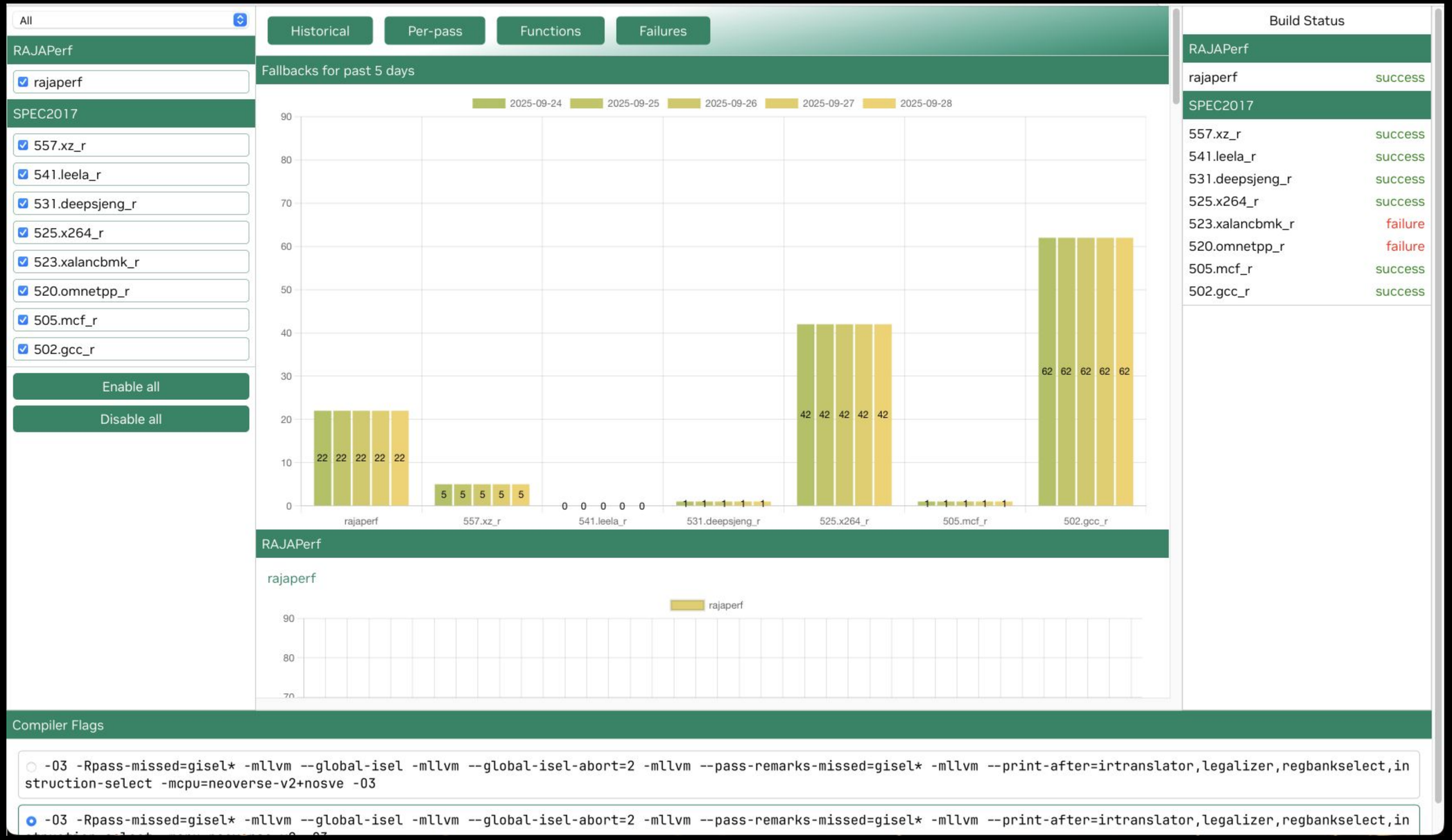
What this is

- The GlobalISel CI/CD is an automation that runs a set of benchmarks and try to extract data about the fallbacks to SelectionDAG that occurred
- It consists of a backend organized as a set of shell scripts and a frontend written using the React.JS framework
- The frontend and the backend communicate using the JSON format

Features

- Compiler is updated every day to ToT
- Fallbacks for all benchmarks are collected daily
  - Fallbacks are collected per-phase for the 4 major passes (IRTranslator, Legalizer, RegBankSelect, InstructionSelect)
- Compiler logs and detailed information about fallbacks are maintained for runs done in the previous 5 days
  - Statistics (number of fallbacks) are maintained for all runs
- Ability to filter results shown based on the flagset used to compile the benchmark

Dashboard View

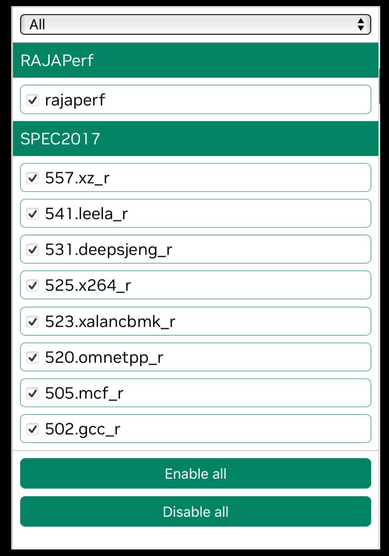


Internals

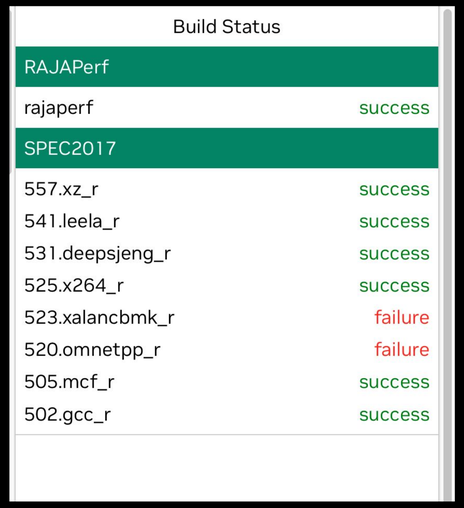
How it works - frontend

- The frontend is a React.JS application that asynchronously loads multiple JSON data sources together
- It does a primary grouping of runs by the flags that they were compiled with
  - In our case, it allows comparing runs done with SVE vs. runs done without SVE
  - The UI filters the results based on the flags that are selected

- A secondary grouping is done based on the benchmarks of each suite
  - This allows choosing which benchmarks are to be shown in the output



- The interface also shows the build status of each benchmark matching the current set of flags chosen



Internals

Frontend - views

- There are four main views in the UI:
  - Historical – an aggregated view of total fallbacks for all benchmarks over the past 5 days, along with per-benchmark views for the past 20 days
  - Per-pass – an aggregated view of fallbacks per-benchmark per-pass (`IRTranslator`, `Legalizer`, `RegBankSelect`, `InstructionSelect`) for the last run along with per-benchmark fallbacks for each pass
  - Functions – the list of functions that fell back to SelectionDAG for each benchmark
  - Failures - the specific failure that caused a fallback

- The functions and failures views should be combined, however it is not always possible to correlate the instruction that caused a failure back to the function it came from

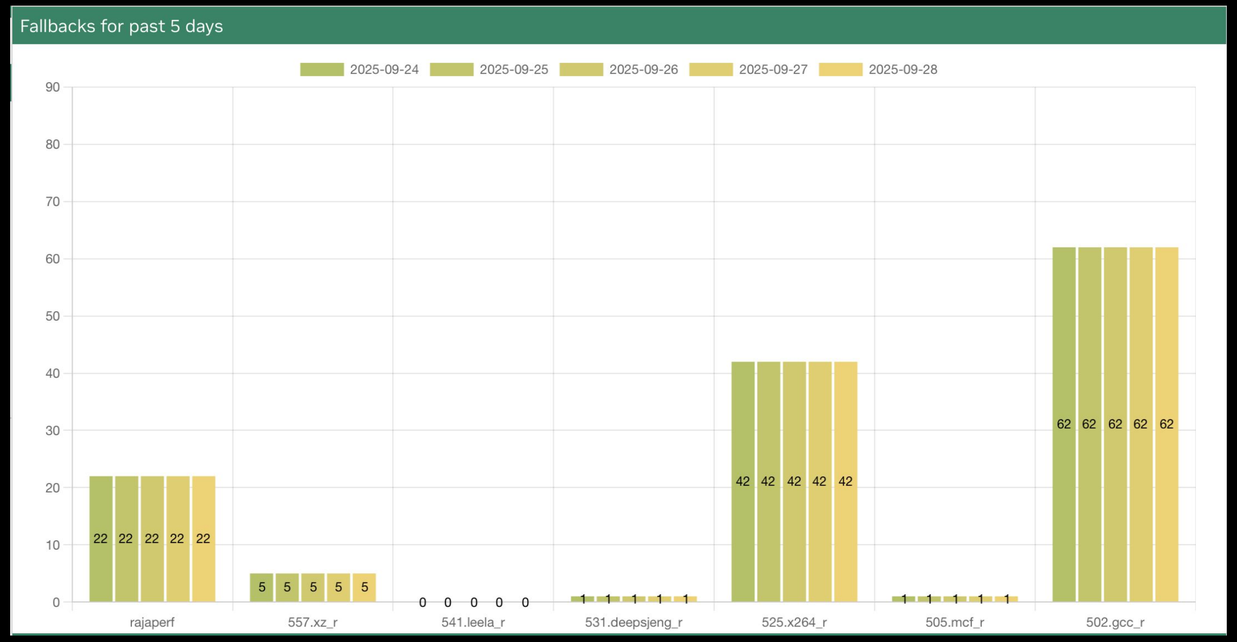
Internals

How it works - backend

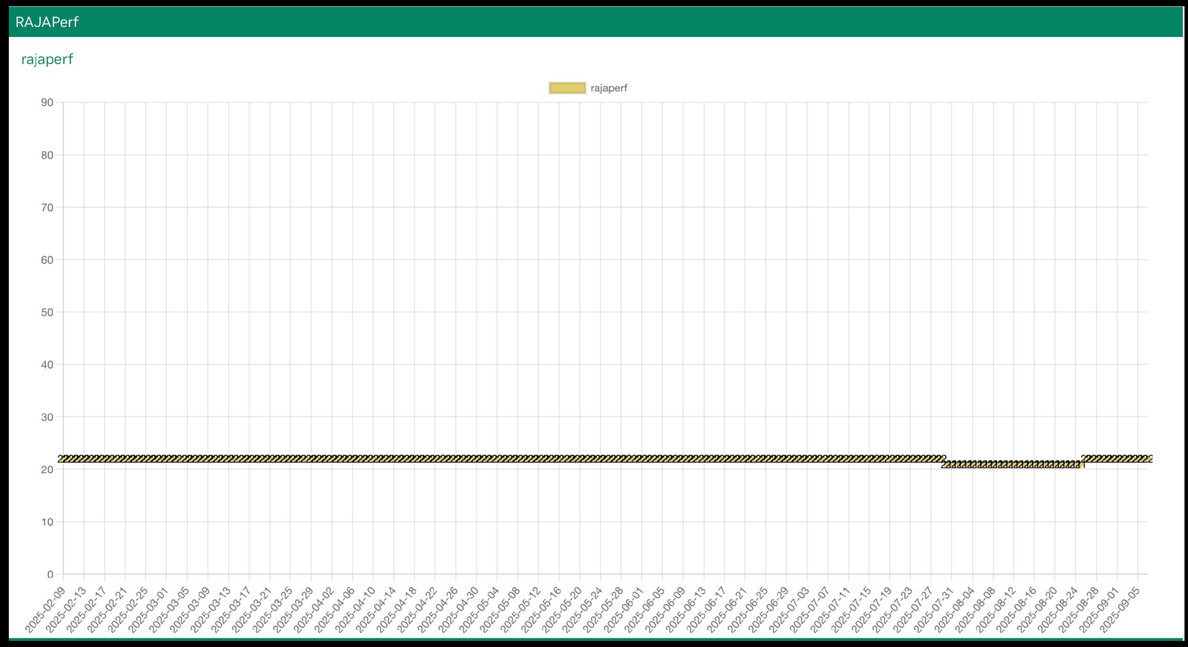
- The backend scripts are organized into:
- A top-level `run.sh` script that invokes scripts for each benchmark of each suite
  - A suite is a set of benchmarks that share the same set of scripts
  - Example: The SPEC2017 suite has various benchmarks like gcc/x264/…
    - Suites can also have one benchmark, like RAJAPerf - rajaperf
- For each benchmark:
  - `verify-env.sh`: To ensure that the environment for the benchmark is setup correctly
  - `run.sh`: To compile the benchmark
  - `get-buildfiles.sh`: To get the list of files containing the compiler output – these files are automatically pattern matched to find fallbacks
  - `get-otherfiles.sh`: To get a list of temporary files that should be saved before the build directory is deleted
- The collected data is then organized into a JSON file that contains per-benchmark information about fallbacks

Screenshots

Historical view



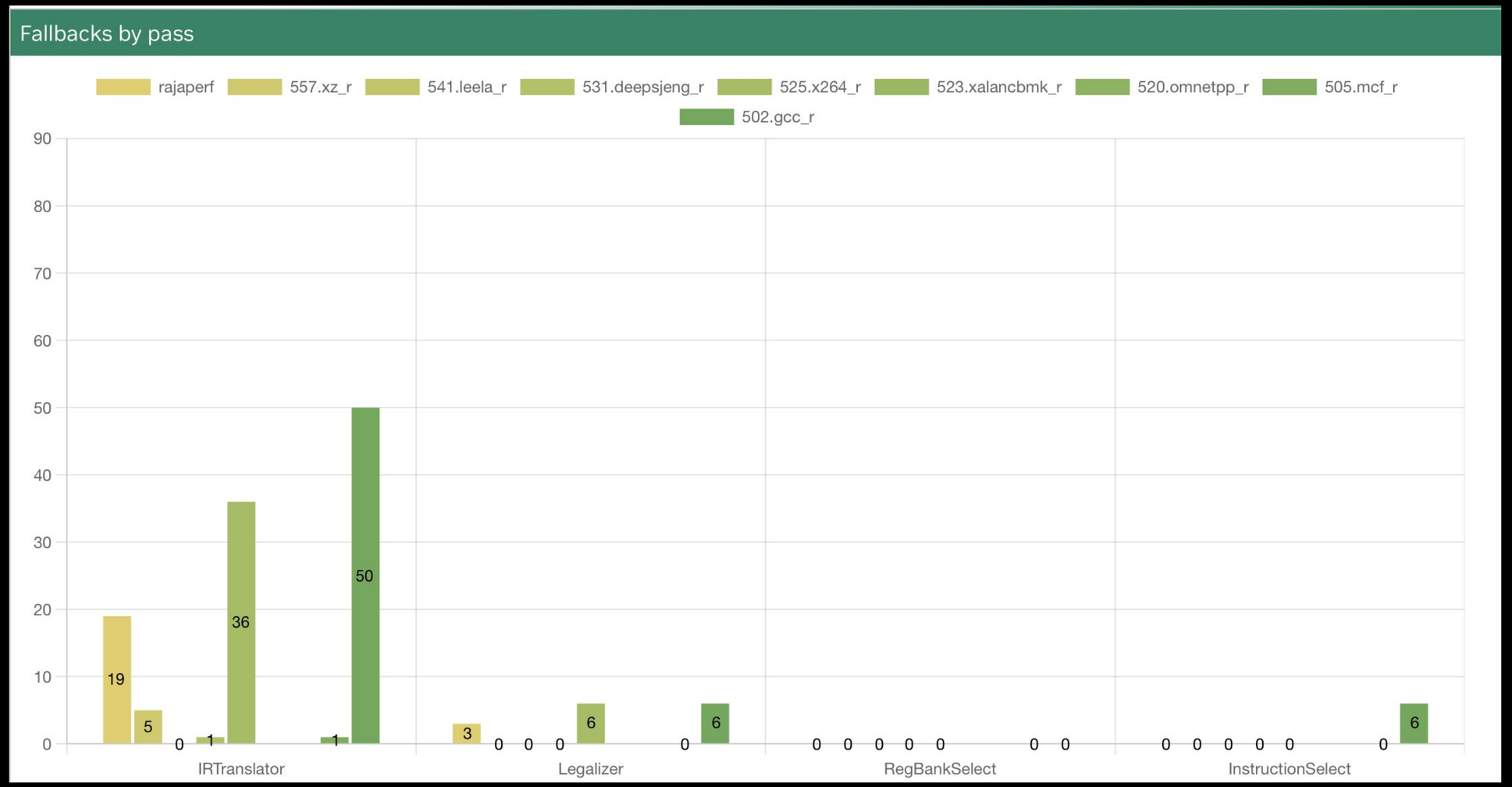
Aggregated fallbacks



Per-benchmark fallbacks

Screenshots

Historical view



Per-pass fallbacks in previous run

Screenshots

Functions view

IRTranslator

RAJAPerf

rajaperf

- _ZN8rajaperf5basic9PI_ATOMIC13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf5basic13REDUCE_STRUCT5setUpENS_9VariantIDEm
- _ZN8rajaperf5basic9PI_REDUCE13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf5basic8TRAP_INT13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf4apps3FIR13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf4apps13LTIMES_NOVIEW13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf4apps6LTIMES13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf4apps8PRESSURE13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench13POLYBENCH_2MM13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench13POLYBENCH_3MM13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench14POLYBENCH_ATAX13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench14POLYBENCH_GEMM13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench17POLYBENCH_GESUMMV13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench16POLYBENCH_GEMVER13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9polybench13POLYBENCH_MVT13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf6stream3DOT13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf9algorithm10REDUCE_SUM13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf8Executor19writeChecksumReportERSo
- _ZN8rajaperf8Executor14writeFOMReportERSoRSt6vectorINS0_8FOMGroupESaIS3_EE

Legalizer

RAJAPerf

rajaperf

- _ZN8rajaperf5basic5COPY813runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf5lcals12DIFF_PREDICT13runSeqVariantENS_9VariantIDEm
- _ZN8rajaperf8Executor10setupSuiteEv

Per-pass view (passes/benchmarks with no fallbacks are not shown)

Screenshots

Failures view

RAJAPerf

rajaperf

File	Line	Column	Function	
COPY8-Seq.cpp	21	13	_ZN8rajaperf5basic5COPY813runSeqVariantENS_9VariantIDEm	unable to legalize instruction: G_STORE %113
PI_ATOMIC-Seq.cpp	21	17	_ZN8rajaperf5basic9PI_ATOMIC13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
REDUCE_STRUCT.cpp	69	21	_ZN8rajaperf5basic13REDUCE_STRUCT5setUpENS_9VariantIDEm	unable to translate instruction: inserteleme
PI_REDUCE-Seq.cpp	21	17	_ZN8rajaperf5basic9PI_REDUCE13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
TRAP_INT-Seq.cpp	23	16	_ZN8rajaperf5basic8TRAP_INT13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
FIR-Seq.cpp	37	20	<unknown>	unable to translate instruction: load: ' %40
LTIMES_NOVIEW-Seq.cpp	30	33	<unknown>	unable to translate instruction: load: ' %67
LTIMES-Seq.cpp	84	27	<unknown>	unable to translate instruction: load: ' %69
PRESSURE-Seq.cpp	21	16	_ZN8rajaperf4apps8PRESSURE13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
DIFF_PREDICT-Seq.cpp	21	20	_ZN8rajaperf5lcals12DIFF_PREDICT13runSeqVariantENS_9VariantIDEm	unable to legalize instruction: G_STORE %142
POLYBENCH_2MM-Seq.cpp	21	21	_ZN8rajaperf9polybench13POLYBENCH_2MM13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
POLYBENCH_3MM-Seq.cpp	150	35	<unknown>	unable to translate instruction: load: ' %12
POLYBENCH_ATAX-Seq.cpp	112	31	<unknown>	unable to translate instruction: load: ' %53
POLYBENCH_GEMM-Seq.cpp	22	22	_ZN8rajaperf9polybench14POLYBENCH_GEMM13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
POLYBENCH_GESUMMV-Seq.cpp	89	36	<unknown>	unable to translate instruction: load: ' %45
POLYBENCH_GEMVER-Seq.cpp	23	24	_ZN8rajaperf9polybench16POLYBENCH_GEMVER13runSeqVariantENS_9VariantIDEm	unable to translate instruction: inserteleme
POLYBENCH_MVT-Seq.cpp	113	33	<unknown>	unable to translate instruction: load: ' %48
DOT-Seq.cpp	110	15	<unknown>	unable to translate instruction: load: ' %53
params_base.hpp	57	83	<unknown>	unable to translate instruction: load: ' %44
Executor.cpp	246	16	_ZN8rajaperf8Executor10setupSuiteEv	unable to legalize instruction: %761:_(<16 x
Executor.cpp	1281	16	_ZN8rajaperf8Executor19writeChecksumReportERSo	unable to translate instruction: inserteleme
Executor.cpp	1000	16	_ZN8rajaperf8Executor14writeFOMReportERSoRSt6vectorINS0_8FOMGroupESaIS3_EE	unable to translate instruction: inserteleme

Future Directions

- If we have enough interest from the community, we are interested in open-sourcing this so that people can add more benchmarks, configurations and views of the database


NVIDIA
Thank You!