Manual: Automated Stencil Compilers Comparison

By Gsponer S., Guerrera D. and Maffia A.

In this document is given a overview over the structure of this little framework and how to use it.

I. General Structure

The root directory ($BENCHROOT) contains three folders:

```
$BENCHROOT
├── StencilCompilers
│   ├── installAll.sh
│   └── <CompilerName>
│       └── install<CompilerName>.sh
├── wave
│   ├── cleanAll.sh
│   ├── runAll.sh
│   ├── gatheringOutputs.sh
│   └── <CompilterName>
│       ├── run
│       ├── output
│       └── src
├── workflow_outputs
├── manual
├── workflow.sh
└── SetupEnv.sh
```

A. StencilCompilers

This folder contains all the compilers. The shell script *installAll.sh* call the corresponding *installCompiler.sh* in the sub folders. This scripts downloads the source, extract it and compile the files.

B. wave

Every stencil compiler needs its own prepared code for the comparison. The wave directory contains prepared code for Patus, Pochoir, Pluto, Halide and a naive approach. For each compiler the structure in the sub folder is the same and should be self explaining. Additionally to the compiler specific files some scripts

are also provided.

- runAll.sh: Compiles the stencil with each compiler and run it for fixed number of threads.
- cleanAll.sh: Cleans up the sub folders from the last compilation.
- geatheringOutputs.sh: Collects the output of the different compilers and arrange them in the result.dat file.

C. *workflow_outputs*

In this folder will be stored the graphs generated by the execution of the workflow, showing comparison among compilers.

D. *manual*

In the manual folder all files for this document are stored.

II. Usage

There are two ways to use the workflow: a fully automated and a step-by-step one.

A. *Fully automated usage*

```
$ ./workflow.sh run|install
```

You can use this command to install the compilers or automatically run the tests. In both cases you can specify one or more compilers to be installed or executed. At the end of its execution a graph will be produced and placed into the *workflow_outputs* folder.

B. *Step-by-step usage*

First thing you need to do is source the SetupEnv.sh file. This sets some environment variables for all the shell scripts like the root directory.

```
$ source SetupEnv
```

If the compilers are not installed yet switch to the *StencilCompilers* folder and run the *installAll.sh* script.

```
$ cd StencilCompilers
$ ./installAll.sh
```

It is also possible to run the script only for a specific compiler. To achieve this, simply pass the name of the compiler as argument.
Each compiler comes with its own dependencies. At the end of this document will be provided a list for each compiler.
After installing all of the compilers you can begin to run the comparison. For doing this, change to the *wave* folder and execute the *runAll.sh*.

```
$ cd ../wave
$ ./runAll.sh
```

This step will take some time since it actually runs the wave stencil with all of the compiler for different number of threads. By default this runs the stencil for 1, 2, 4, 8, 16, 32 threads but this can be changed in the *SetupEnv.sh* file.
When the computation is finished the *gatheringOutputs.sh* script can be executed: it creates the result.dat file containing all of the outputs. The data stored into it can afterwards be plotted exploiting the *gnuplot.gp* file.

```
$ ./gatheringOutputs.sh
$ gnuplot gnuplot.gp --persist
```

After this step a plot sorted by number of threads shows up.

III. Dependencies

A. General

- wget
- Make

B. Patus

- Java 7 or newer
- Maxima
- gcc

C. Pochoir

- Intel C++ Compiler version 12.0.0 or later (up to v 13.1.2) with Cilk Plus extension
- Glasgow Haskell Compilation System version 6.12.1 or later
- Parsec-2.1.0.1 or later

D. Pluto

- gmp (GNU multiple precision arithmetic library)

E. Halide

- llvm 3.2 or higher
- clang version matching to the llvm version