INTRODUCTION TO PC-A

"Pencil Code with Astaroth inside"

Work by M. Väisilä, J. Pekkilä, O. Lappi, T. Puro, M. Rheinhardt, led by M. Korpi-Lagg @ CS, Aalto University, Espoo, Finland

MINDSET

- two different codes, although sharing some numerics,

  in two different repositories

- two different computers, although GPU dependent on CPU

  → everything needed for advancing the PDE variables

  - needs to be in GPU memory (parameters and initial conditions)
  - and in kernel code (recipes for right hand sides and boundary conditions)

- f-array is updated on CPU only when needed, df-array does not exist

PC-A tries to use all resources on a node with maximum concurrency!

HOW TO GET THE CODES

- both codes together with a fresh pull:

```
git clone -b gputestv6 --recurse-submodules https://<username>@github.com/pencil-code/pencil-code.git
```

or

```
git clone -b gputestv6 --recurse-submodules https://<username>@pencil-code.org/git/ pencil-code
source sourceme.sh
cd $PENCIL_HOME
cd src/astaroth/submodule
git checkout PCinterface_2019-8-12
```

- or add Astaroth to an existing PC installation:

```
cd $PENCIL_HOME
git checkout gputestv6
git submodule update --init -recursive
cd src/astaroth/submodule
git checkout PCinterface_2019-8-12
```

HOW TO BUILD WITH PC-A

- on Clusters, load appropriate CUDA (NVIDIA) or HIP (AMD) and MPI modules, perhaps `cmake`
- set in `Makefile.local`

```
GPU                             = gpu_astaroth
      GPU_VENDOR                = [nvidia|amd]                    # nvidia is default
      MULTITHREADING = openmp
```

- if you haven't used PC-A before in work directory, execute `pc_setupsrc`

  -> new symbolic links and directories in

  `src/astaroth`, esp. `src/astaroth/submodule` (consider black box)

- build Pencil Code as usual:
  - `make`: set `MODULE_[PRE|IN|SUF]FIX` environment variables as given in config file

    for chosen compiler + given machine
  - `pc_build`: use flag `-s|--serial` for safety
  - `gfortran`: set flag `-std95` to `f2003`,

    set `LDFLAGS_MAIN = -Bdynamic -Wl,--export-dynamic` in `Makefile.src`
  - CRAY: set `LDFLAGS_MAIN = -h dynamic -Wl,--export-dynamic` in `Makefile.src`

HOW TO BUILD PC-A

- build process creates
  - interface and DSL code according to the setup
  - several Astaroth libraries and the interface library

- for libraries: ∃ separate `Makefile` in `src/astaroth`

  -> libraries can be build separately (perhaps before: "`make clean`", answer "y")

- most importantly: kernels and their calling sequence in `src/astaroth/DSL/local`:

  `solve_two.ac` (`solve_single.ac`)

- which includes

  `steps_two.h` (`steps_single.h`): calling sequence of kernels

  `equations.h`: contains for each PDE a function named `d<variable>_dt`, e.g.,

  `duu_dt, dlnrho_dt, daa_dt`

  `boundconds.h`: boundary conditions, mandatory "step"

WHAT IS DSL CODE?

- DSL="Domain-specific language" for conveniently formulating rhss of PDEs for GPU kernels
- C-like, with some Python-like features
- to be understood pointwise on grid
- Example: continuity equation

```
  rhs=0.
          glnrho = gradient(LNRHO)                              // grad(rho) or
grad(lnrho)!

          if (ldensity_nolog){
               return  rhs - dot(UU, glnrho) - RHO*divergence(UU)
           }
           else {
              return  rhs - dot(UU, glnrho) - divergence(UU)
          }
```

WHAT THE BUILD YIELDS

- virgin build -> void rhs functions in `equations.h` -> user intervention needed:

inspect directories in `src/astaroth/DSL/`

```
density
        entropy
        forcing
        hydro
        magnetic
        shock
        supernova
```

for useful code snippets

- indicate, which physics modules are supported presently
- differential operators etc. are in `src/astaroth/DSL/stdlib`

CUSTOMIZE RHS

- to specify a rhs, modify e.g., function `dlnrho_dt` of `src/astaroth/DSL/local/equations.h`

```
dlnrho_dt(int step_num){ return 0. }
```

to

```
dlnrho_dt(int step_num){
    #include "../density/continuity.h"
}
```

or

```
duu_dt(int step_num){ return real3(0.,0.,0.) }
```

to

```
duu_dt(int step_num){
    #include "../hydro/momentum.h"
}
```

CUSTOMIZE RHS

- "physics branches" in DSL code can be selected by preprocessor statements like

```
#if LMAGNETIC
    ...
#endif
```

for each enabled physics module, a flag is predefined in `src/astaroth/PC_moduleflags.h`

- or conditionals in DSL syntax, like

```
if (ltemperature) {
...
}
```

- all switches from `src/cparam.inc` available

- changes to `equations.h` are permanent = not overwritten by future builds

(additional physics -> new empty rhs functions appear,
no longer needed functions do not disappear but are idle)

- `equations.h` is considered by `pc_newrun` and `cvsci_run`

CUSTOMIZE RHS

- Caveat: it is advisable to check predefined DSL code in the beginning,

  at least more complex functions like `denergy_dt`

- Note: to enable an additional physics module, the block in

  `src/gpu_astaroth.f90` has to be released;

  can require some non-standard code development

- Limitations:
  - particles/pointmasses/radiation/solid cells/self-gravity/testfields

  presently not supported
  - modifications of f-array in `*before/*after_boundary` routines not supported

  if needed every timestep
  - diagnostics, which are not only from `pencil_case/f-array` not calculated
  - not all boundary conditions "transpiled" yet (coming soon)

ADD MISSING PARAMETERS TO RHS

if additional parameters of a physics module needed,
they have to be pushed to the GPU, for that

- find `subroutine pushpars2c(p_par)` near the end of the physics module
- add lines

  `call copy_addr(<parameter>,p_par(<running index>))`

  for integer parameter: add `! int` at line end

  for real 1D array parameter: add `! (<dimension>)` at line end

- increase `n_pars` accordingly

- Note: manipulations of parameters, which can be performed in module initialization should not be coded in DSL
  -> push derived parameters!

- parameters from `src/cparam.h, src/cdata.h:` all available

RUNNING PC-A

Example SLURM batch script:

```
#SBATCH --nodes=2            # Total number of nodes
#SBATCH --ntasks-per-node=8
```

CRAY (LUMI, Dardel, Frontier):

```
#SBATCH --gpus-per-node=8      # Allocate one gpu per MPI rank
```

CSC machines:

```
#SBATCH --gres=gpu:v100:4

#SBATCH --cpus-per-task=7      # multithreading

source src/.moduleinfo
export LD_LIBRARY_PATH=${CRAY_LD_LIBRARY_PATH}:$LD_LIBRARY_PATH
export OMP_NUM_THREADS= =${SLURM_CPUS_PER_TASK}
export OMP_PROC_BIND=close,spread
export OMP_MAX_ACTIVE_LEVELS=2
export OMP_WAIT_POLICY=PASSIVE
./start.csh
export MPICH_GPU_SUPPORT_ENABLED=1   (MPICH on CRAY)
./run.csh
```

TROUBLESHOOTING

- enforce re-creation of interface code by

  `rm src/astaroth/PC_moduleflags.h` (tb improved)

  `pc_build …`

- obey DSL syntax meticulously - consult:

  https://bitbucket.org/jpekkila/astaroth/src/PCinterface_2019-8-12/acc-runtime/README.md

- consult `samples/gputest`

- consult touko.puro@aalto.fi or matthias.rheinhardt@aalto.fi

- DSL compiler can be A DIVA !

OUTLOOK

- in preparation:
  - run-time compilation -> all variables constant during time-loop,

    esp. logical variables, are replaced by their values from `start.in/run.in`

    performance!

full transpilation of the rhss to DSL -> manual DSL coding no longer needed!