Peter Willendrup DTU Physics & ESS DMSC

Porting McStas to GPU via code-generation and OpenACC pragmas

Main events on timeline of current developments

2017: E. Farhi
initial cogen
modernisation

Fall 2018 onwards: J. Garde
further cogen modernisation and
restructuring

October 2019 onwards:
J. Garde & P. Willendrup:
New RNG, test system, multiple
functional instruments.

March 2018:
Participation at
Dresden Hackathon.
1st “null” instrument
prototype runs.

mentor: Vishal Metha

October 2019:
Participation at Espoo
Hackathon. First meaningful
data extracted. Work on
cogen and realising we need
another RNG.

mentor: Christian Hundt

November-
December 2019:
First good look at
benchmarks and
overview of what
needs doing for first
release with limited
GPU support.

McStas heading for the GPU… March 2018

McCode on GPU?

1st prototype, “null”-instrument with only one component.

Based on NVIDIA compiler technology,

PGCC and OpenACC

pragmas

More to come in 2019!

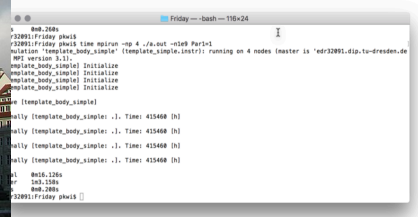
McStas / McXtrace instrument

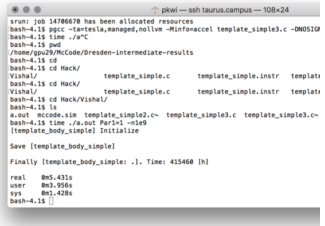
Port heavy part of component to GPU (Shows some potential)

Port neutron loop to GPU
- Lots of code
- Code generation
- Compiles! Speed up today

CPU MPI 4 cores (95 % usage)

GPU 5% usage





16.12 s (Single core 56.0 s)

5.43 s

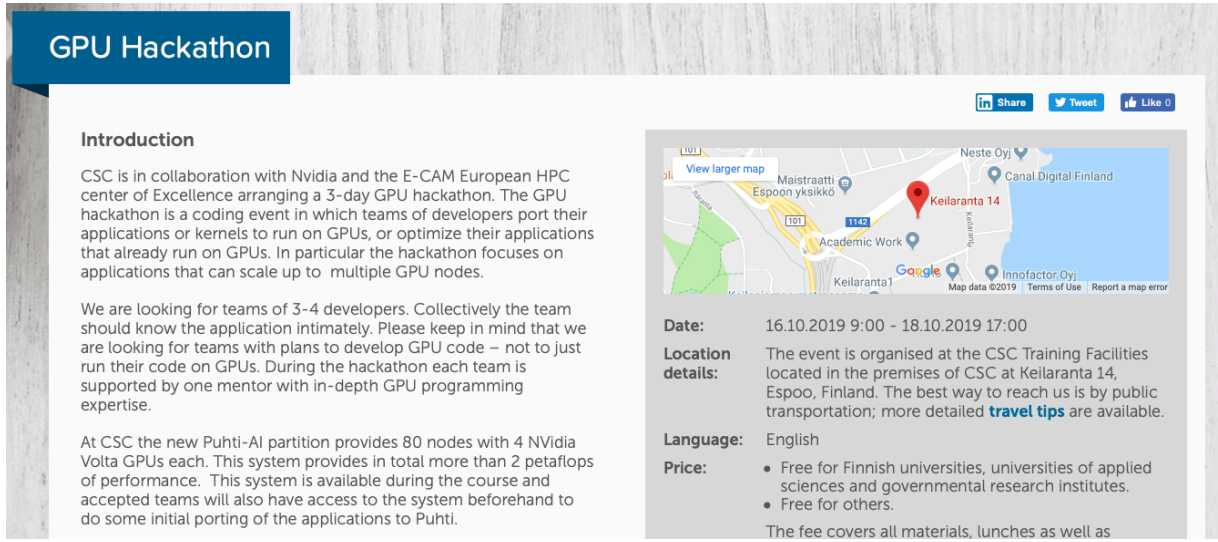
McStas heading for the GPU… October 2019

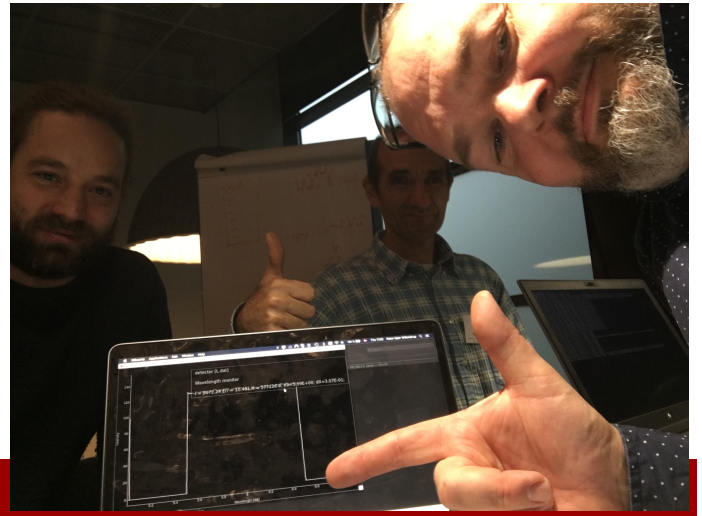
Rewritten code-generation with automated additions of OpenACC pragmas.

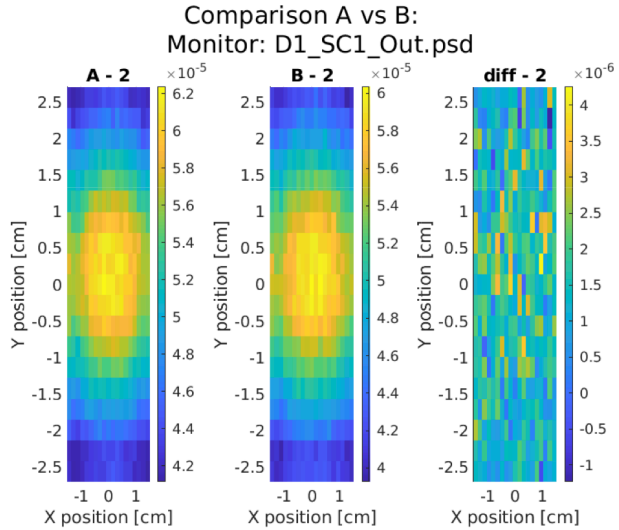
Quite transparent wrt. CPU vs. GPU

First simulations with meaningful output

Speed on DELL with Quadro-card ~ on par with running on CPU with MPI



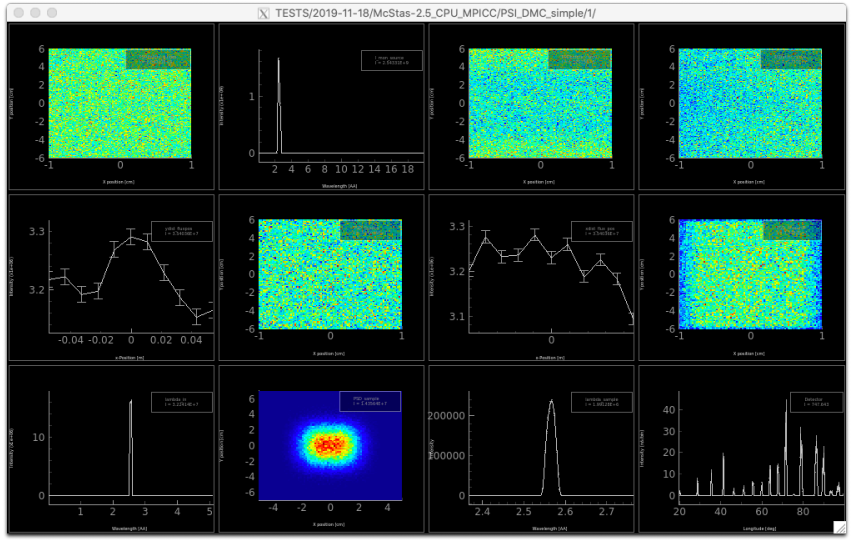




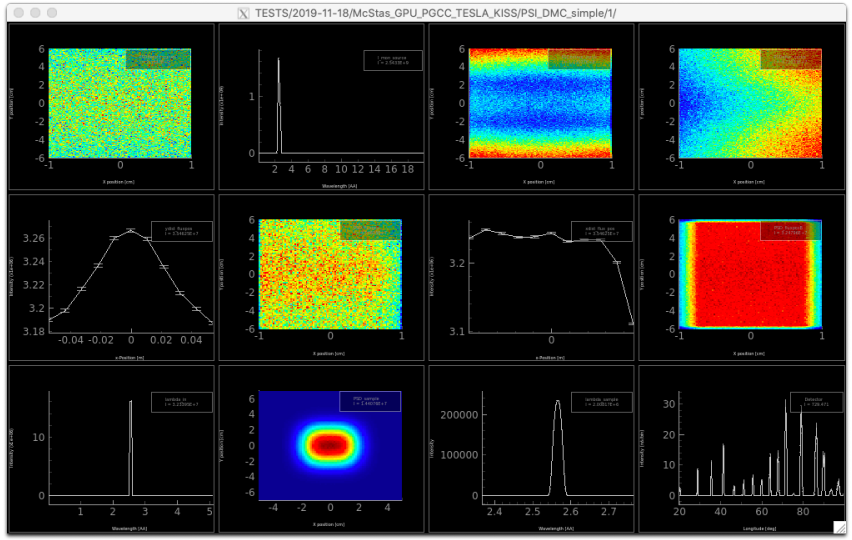
McStas heading for the GPU… November 2019

9 instruments fully ported, also realistic ones like PSI_DMC*

10-core MPI run, **1e7** in 2 secs



Tesla V100 run, **1e9** in 22 secs



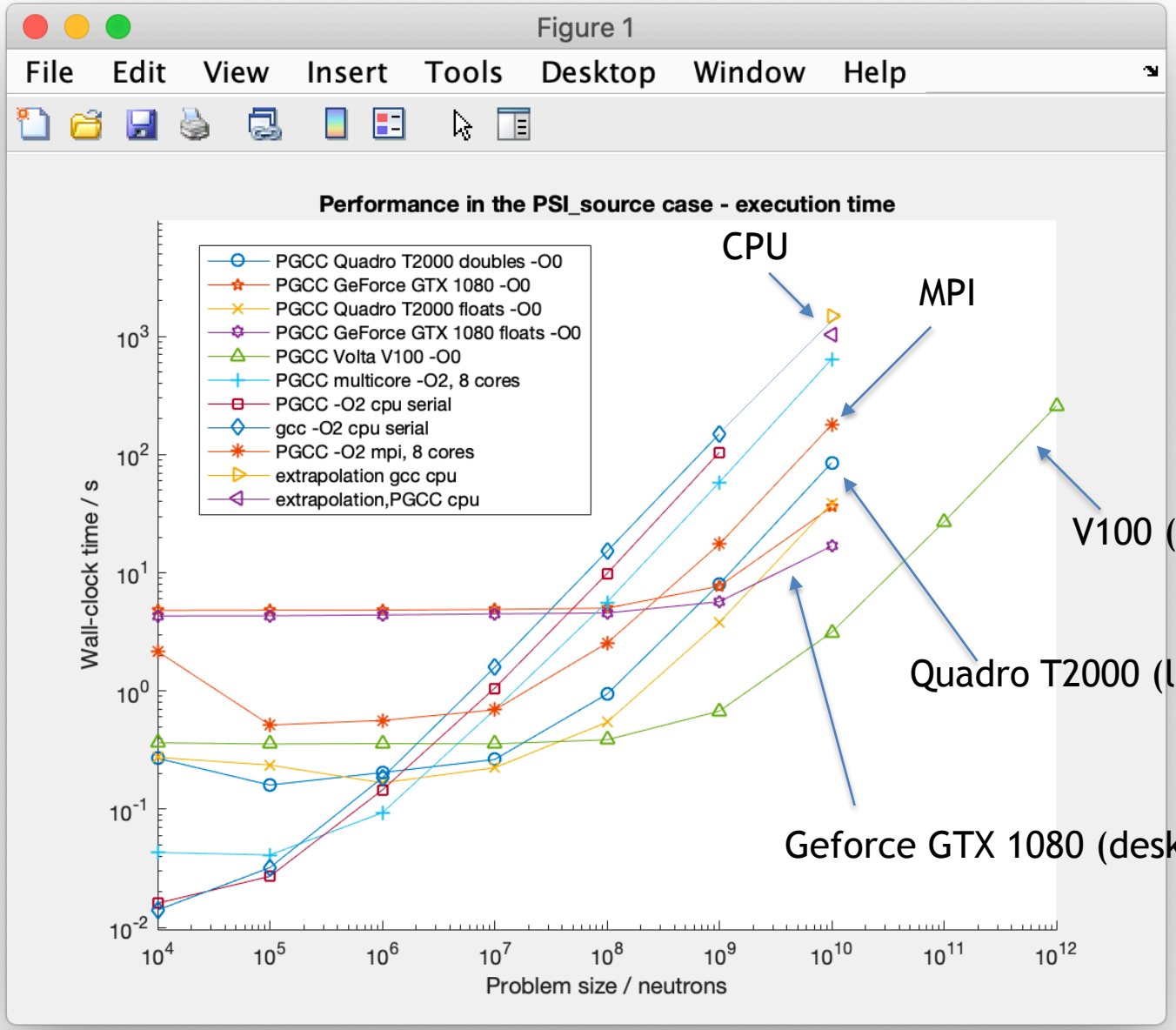
~ i.e. 2 orders of magnitude wrt. a single, modern CPU core

*Guide component without reflection-file support, SPLIT disabled, OFF geometry disabled

McStas heading for the GPU… November 2019 - first good look at performance. Wallclocks:

Idealised instrument with source and monitor only - i.e. without any use of the ABSORB macro.

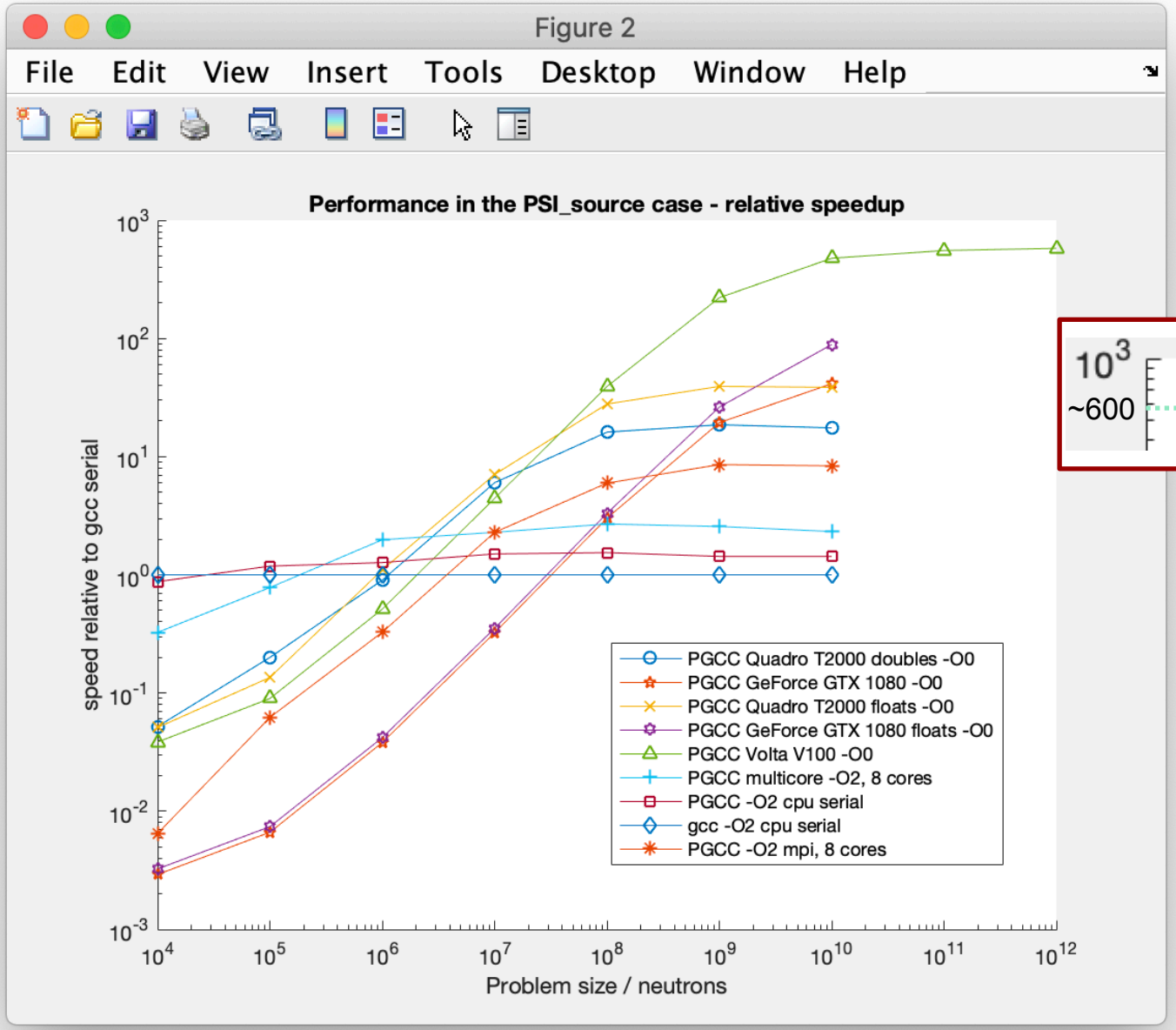
(Likely a good indication of maximal speedup achievable.)



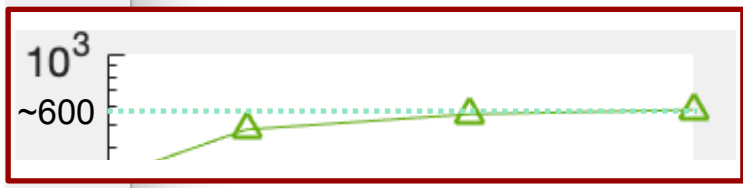
McStas heading for the GPU… November 2019 - first good look at performance. Speedup:

Idealised instrument with source and monitor only - i.e. without any use of the ABSORB macro.

(Likely a good indication of maximal speedup achievable.)



Looks like a factor of ~600



Renormalised to wall-clock of single-core gcc standard simulation

McStas heading for the GPU… December 2019 - today's compilation status:

Numerical output with graphics:
http://new-nightly.mccode.org/2019-12-06/2019-12-06_output.html

Statistics:

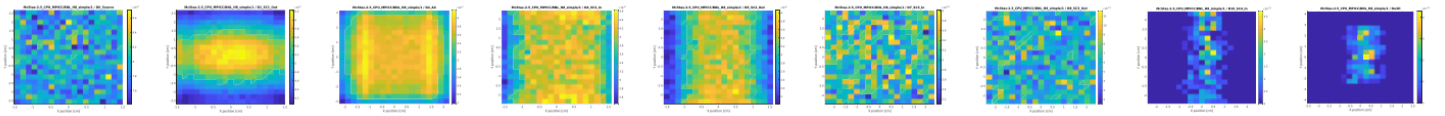
http://new-nightly.mccode.org/2019-12-06/stats.txt

(38 of 142 instruments, 62 of 207 components

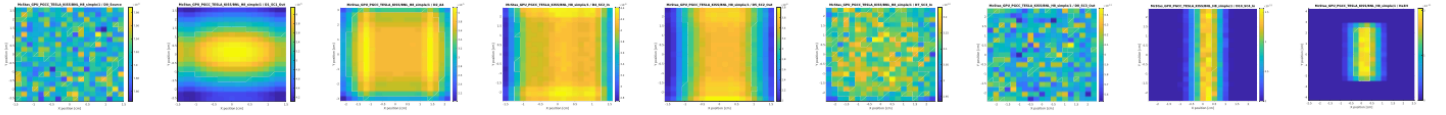
BNL_H8_simple/1 - comparison McStas-2.5_CPU_MPICC vs McStas_GPU_PGCC_TESLA_KISS

(Click to access files)

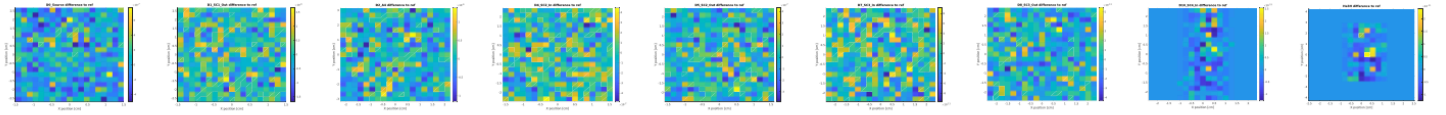
McStas-2.5_CPU_MPICC (reference)



McStas_GPU_PGCC_TESLA_KISS



Difference



test results

ref user: pkwi

	McStas-2.5_CPU_MPICC (ref) - 1e7 n-62-23-6 Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz 20191120_0127_38	McStas_CPU_GCC_KISS - 1e6 n-62-21-99 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz 20191205_0058_36	McStas_CPU_GCC_MT - 1e6 n-62-21-99 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz 20191205_0122_14	McStas_CPU_MPICC_KISS - 1e7 n-62-23-9 Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz 20191205_0034_53	McStas_GPU_PGCC_TESLA_KISS - 1e9 n-62-20-6 Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz Tesla V100-PCIE-16GB 20191205_0012_04
BNL_H8_simple	5.80 s | 1.99 s | 9.7e-10 | 99%	4.32 s | 1.03 s | 1.3e-09 | 136%	4.35 s | 0.99 s | 1e-09 | 106%	4.11 s | 1.50 s | 1.1e-09 | 110%	17.59 s | 8.16 s | 9.7e-10 | 98%

McStas 3.0 - next generation code generator - release plans

- Limited-functionality “beta” release to be made public soon (jan-mar) after **2.6 (january)**
  - Expect bugs!
  - Only a subset of components / instruments
  - Event interchange with 2.6 possible via MCPL

- Main purpose: **get this working in ‘the wild’**
  - Your instruments will likely **require (limited) rewriting**
    - E.g. the declare section **cannot include assignments**
  - Your own components will **likely require rewriting**
    - E.g. the declare section **cannot include assignments**
    - Arrays must be declared/initialized using a new set of functions (i.e. not double PSD_I[nx][ny] with definition parms)

- Hence some backward compatibility is lost and we need **to increment major release #**