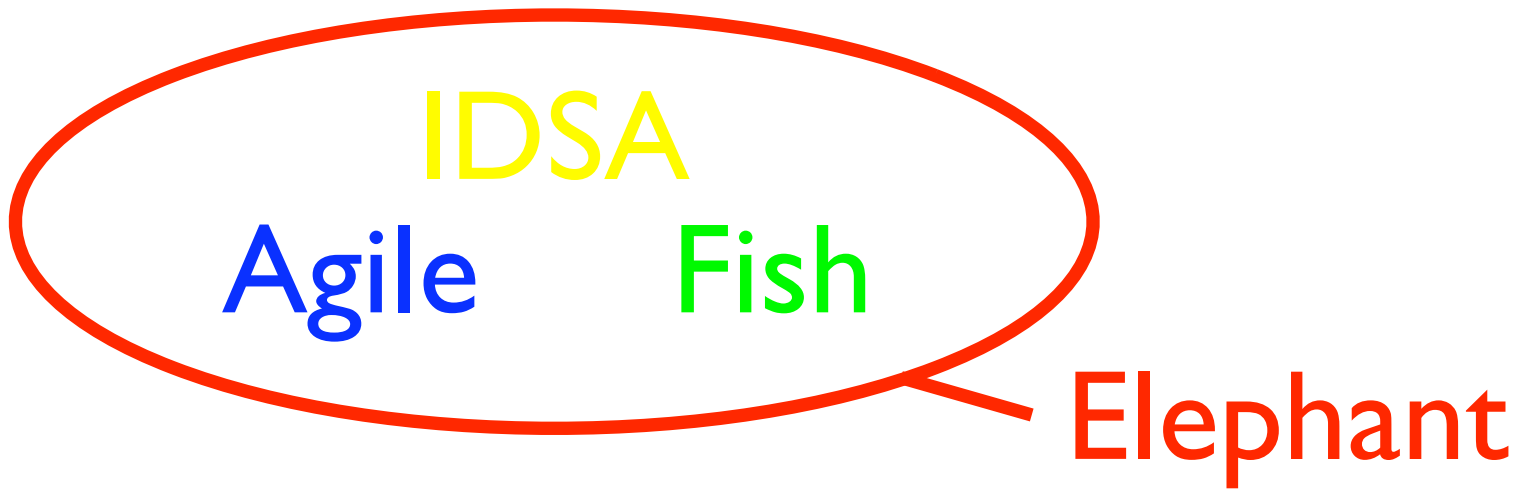
Kevin Ebinger, Claudio Gheller, Alistair Hart, Oliver Heinimann, Matthias Liebendörfer, Kuo-Chuan Pan

The Elephant Supernova Code

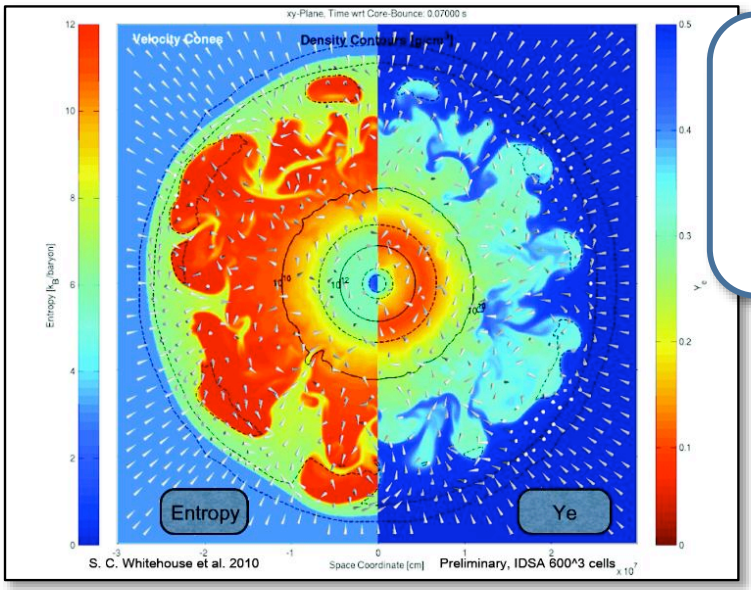
ELEPHANT = ELEgant Parallel Hydrodynamics with Approximate Neutrino Transport



University of Basel 2005-2015

EuroHack 2015

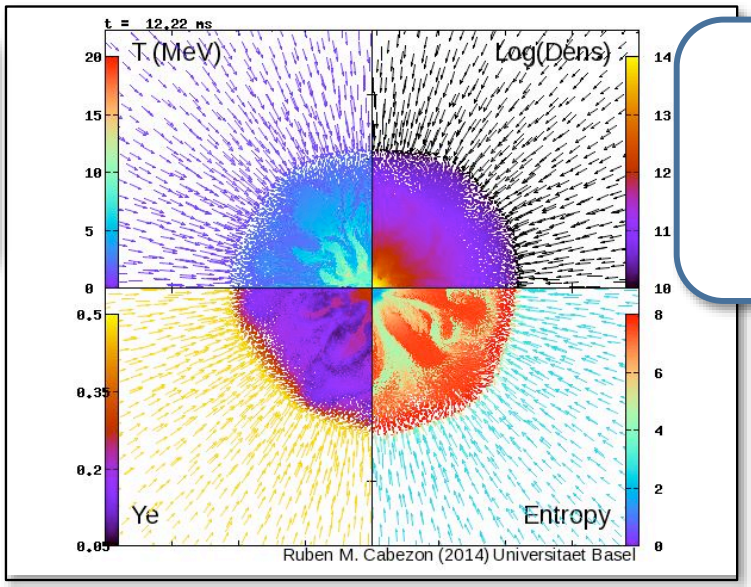
Four independent 3D supernova codes



Elephant

3D IDSA
Cartesian mesh
1D GR potential

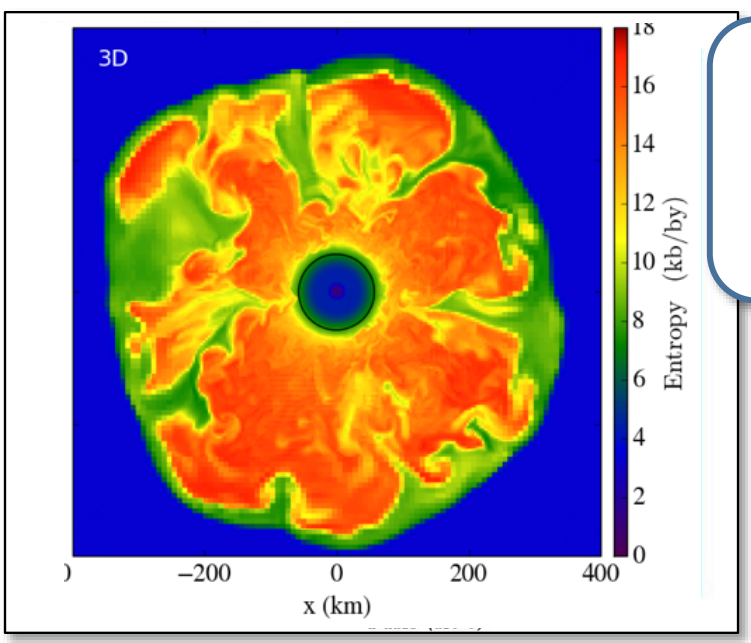
M. Liebendörfer
S. C. Whitehouse
R. Käppeli



SPHYNX

ASL
SPH
3D Newtonian

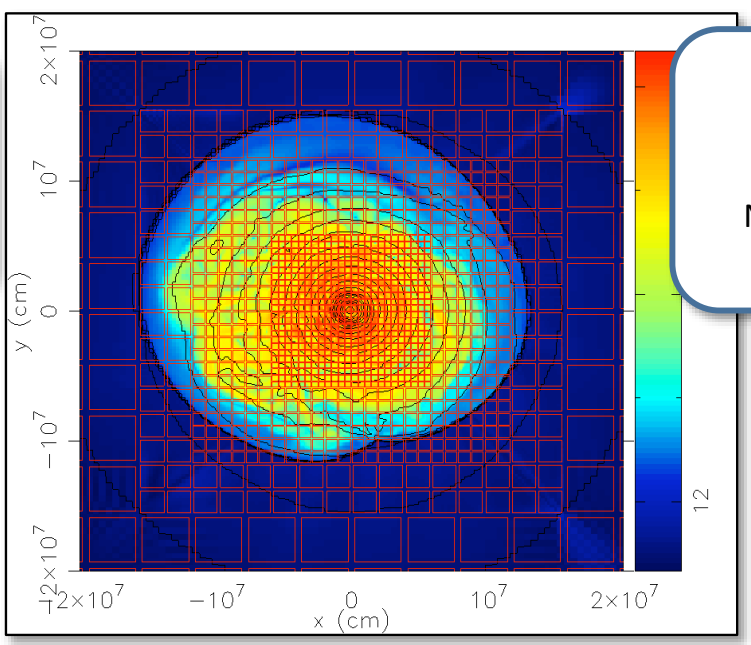
R. M. Cabezón



FLASH

3D IDSA
AMR
3D Newtonian

K.-C. Pan

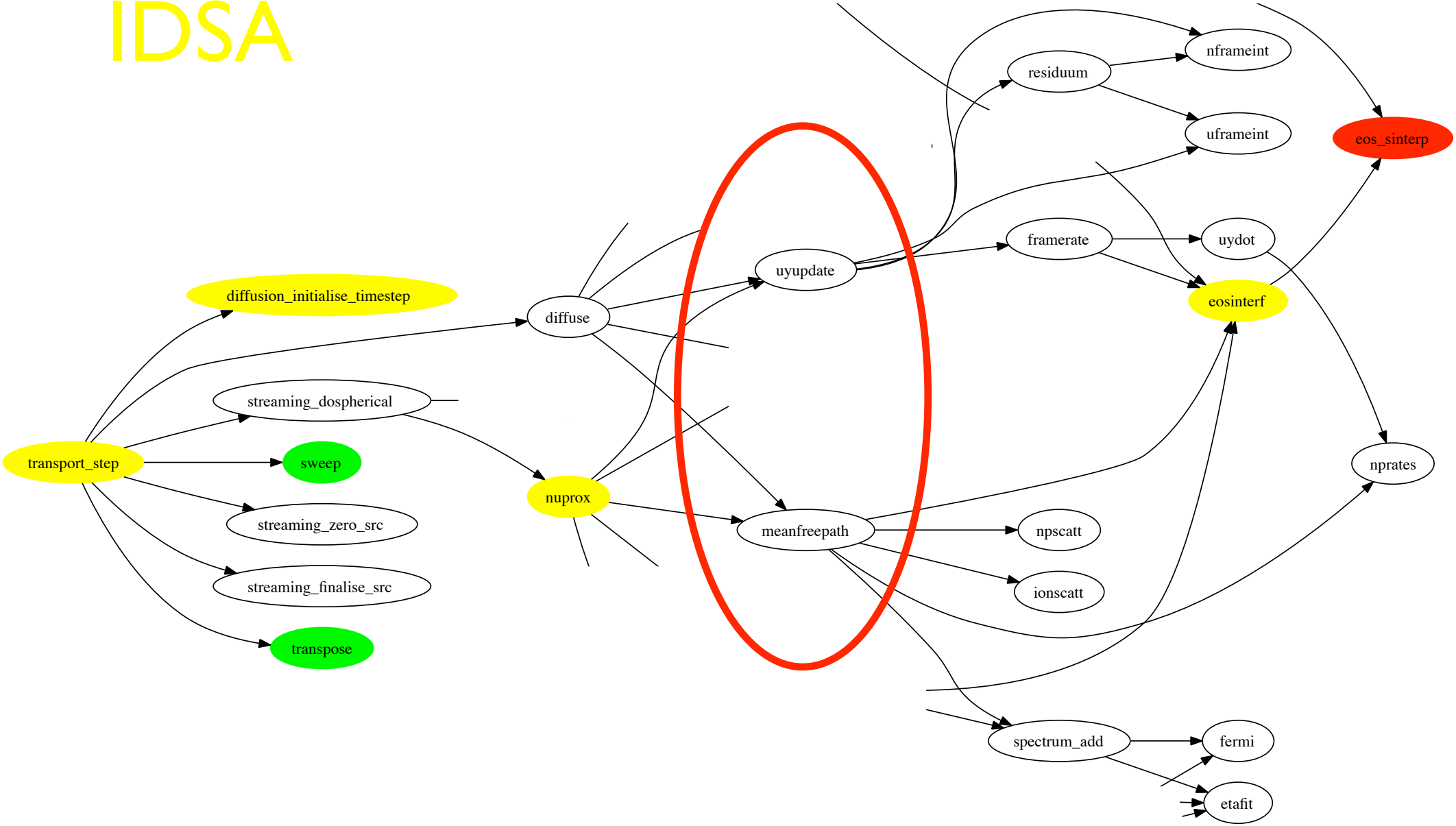


fGR_M1

M1
Nested meshes
3D GR

T. Kuroda

IDSA



(Liebendörfer, Whitehouse, Fischer, ApJ 2009)

Day 1

UNI BASEL

Observation

- ~90% of time spent in single-cell calls to meanfreepath() and uyupdate()
- Operations on cell i are logically complicated, but neither compute-intensive nor dependent on other cells.

Goal

- Accelerate meanfreepath() and uyupdate() using GPU.
- Call with an array of cells instead of single cells.

Problem

- Many failed attempts to inline routines with -ipafrom=...

Day 2

UNI BASEL

Solution

- Integrate independent idsa-subroutines into one idsa_module.f90.
- Create new possibility to call with array arguments.

Goal

- Compiler can inline all idsa-routines without special makefile.

Problem

- perftools do not work for our code.
- First run on GPU: Incorrect results! Race-condition?

Day 3

UNI BASEL

Solution

- Debugging with OpenMP instead of OpenACC. Race-condition found and fixed.
- perftools not compatible with -K trap=fp (thanks fgp!)

Problem

- Runs on GPU deliver correct results, but very slow. Huge data load time for equation of state table.
- We don't see what the GPU is doing.

Goal

- Load tables once at the beginning of the simulation.
- Investigate what the GPU is doing, speed up...

Day 4

UNI BASEL

Solution

- Learned about many environment variables and compile options to monitor compilation and code execution.
- nvvp viewer shows details about GPU usage.

Goal

- Optimise the GPU usage for the cell arrays.

Problem

- Copyin/out dominates computation.
- Hidden synchronisation with async(…) clause.

Day 5

UNI BASEL

Solution

- Manage data transfer manually and split into streams to hide memory copies during calculations.
- No solution for hidden async() synchronisation.

Goal

- Measure and compare performance.
- Include new idsa_module.f90 in FLASH and ELEPHANT.

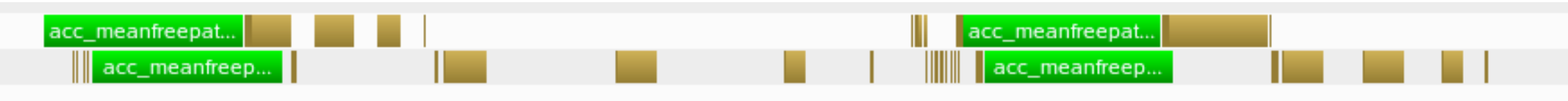
Problem

- Not many, just requires more work :-)
- Can we hide further mem-copies by reordering work?

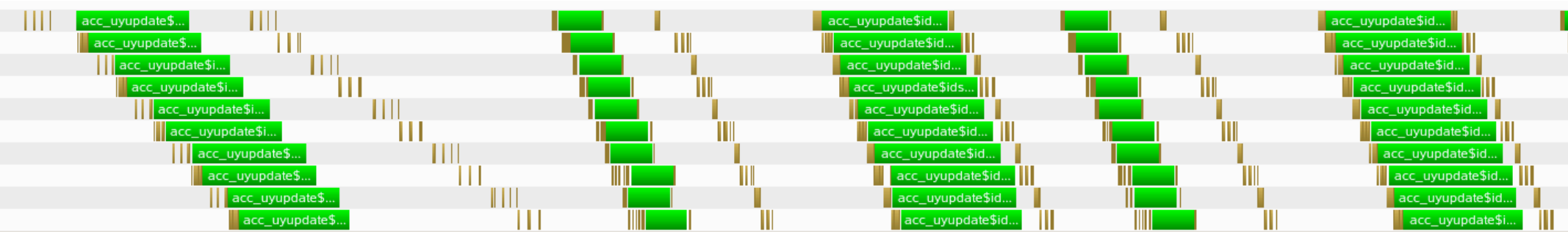
Results

UNI BASEL

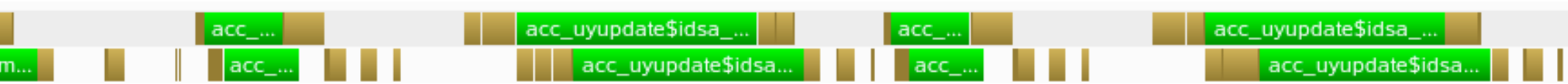
Starting with 2 streams...



Getting exited about 10 streams...



Leaving it at 2 streams.



Before (1 CPU):

Time%	Time	Imb. Time	Imb. Time%	Calls	Group Function
100.0%	0.676340	--	--	1628147.0	USER
17.1%	0.115953	--	--	252450.0	nprates$idsa_module_
16.9%	0.114248	--	--	201960.0	uydot$idsa_module_
11.5%	0.077870	--	--	141955.0	framerate$idsa_module_
10.7%	0.072137	--	--	50490.0	uyupdate$idsa_module_
8.6%	0.057929	--	--	50490.0	meanfreepath$idsa_module_
6.7%	0.045360	--	--	252450.0	idsa_eos$idsa_module_
5.9%	0.039914	--	--	50490.0	ionscatt$idsa_module_
5.6%	0.037816	--	--	50490.0	idsa_spectrum$idsa_module_
5.0%	0.033892	--	--	91465.0	residuum$idsa_module_
4.4%	0.029494	--	--	100980.0	etafit$idsa_module_
2.3%	0.015779	--	--	192445.0	nframeint$idsa_module_
1.6%	0.011071	--	--	50490.0	npscatt$idsa_module_

After (GPU):

Time%	Time	Imb. Time	Imb. Time%	Calls	Group Function
100.0%	0.055065	--	--	397.0	USER
46.7%	0.025741	--	--	18.0	acc_uyupdate$idsa_module_.ACC_ASYNC_COPY@li.238
27.9%	0.015390	--	--	18.0	acc_meanfreepath$idsa_module_.ACC_ASYNC_COPY@li.158
11.8%	0.006503	--	--	18.0	acc_uyupdate$idsa_module_.ACC_ASYNC_COPY@li.222
4.8%	0.002629	--	--	18.0	acc_meanfreepath$idsa_module_.ACC_ASYNC_KERNEL@li.147
2.5%	0.001359	--	--	18.0	acc_meanfreepath$idsa_module_.ACC_ASYNC_COPY@li.144
1.2%	0.000670	--	--	18.0	acc_uyupdate$idsa_module_.ACC_ASYNC_KERNEL@li.224

Conclusion

UNI BASEL

- We would never have been able to do this alone! Interdisciplinary work is crucial.
- Compared to other interdisciplinary efforts, this format works amazingly well!
- What we haven't achieved in 2 years has happened within one week. Speedup >= 104!

- Documentation of options to monitor and debug code compilation and performance.
- Simple walk-through example with explicit module list, environment setting, makefile, batch scripts, etc.
- Searchable collection of error messages and solut's.