Evaluation of new Languages

Iris Christadler, Leibniz Supercomputing Centre, Germany
Carlo Cavazzoni, Giovanni Erbacci, Filippo Spiga, CINECA, Italy

CAPACITIES

e-infrastructure

Evaluation of new languages - Outline

1. Evaluation Setup
   - Languages
   - Benchmarks
   - Assessment
   - Hardware
   - Mapping of languages & hardware
   - Porting table
2. Productivity Results
   - Developer diary example
   - Source lines
   - Development time
   - Dev. time vs. source lines
3. Performance Results
   - Overview of results
   - Dense Matrix-matrix multiplication
   - Sparse matrix-vector multiplication
   - 1D-FFT
   - PGAS languages
   - GPGPU languages
   - Accelerator languages

How-to assess the productivity of new programming languages?

EVALUATION SETUP

Languages

- Current standards:
  - MPI
  - OpenMP,
  - mixed MPI+OpenMP
- PGAS languages:
  - UPC
  - CAF
  - Titanium
- DARPA HPCS:
  - Chapel
  - X10
  - Fortress
- Hardware accelerators:
  - CellSs
  - CUDA
  - OpenCL
  - HMPP
  - RapidMind
- FPGA programming:
  - VHDL
  - Harwest-C

Benchmarks

Taken from the EuroBen benchmark suite.

- *mod2am*:
  Dense matrix-matrix multiplication.
- *mod2as*:
  Sparse (CSR-format) matrix-vector multiplication.
- *mod2f*:
  One dimensional fast fourier transform.
- *mod2h*:
  Random Number Generator.

Assessment (1/2)

1. C and Fortran versions (serial+parallel) for all kernels
2. Reference Input Data Sets (RIDS)
   - Ensure that problem sizes are relevant for science
   - Chosen also to show advantages of certain architectures
3. Porting of kernels to different languages
   - Combined "Future Technologies" & "Software"-WP effort
   - Usually one person responsible for one language
   - Started with mod2am and mod2as
   - Ported mod2f and mod2h only if time allowed

Assessment (2/2)

4. Productivity measured
   - Developer diaries kept during porting:
     Keep track on programming time vs. performance gains
   - Questionnaire completed afterwards:
     Problems encountered, overall success, number of source lines
5. Source files committed to subversion repository
6. Performance measurements

Hardware

- **“clearspeed”** (CATS units)
  96 Gflops per CSX700 accelerator
- **„itanium“** (SGI Altix)
  1.6 GHz > 6.4 Gflops per Montecito core
- **„huygens“** (IBM Power6 cluster)
  4.7 GHz > 18.8 Gflops per Power6 core
- **„louhi“** (Cray XT5)
  2.3 GHz > 9.2 Gflops per AMD Barcelona core
- **“cell-cluster”** (QS22-blade cluster)
  102.4 Gflops per PowerXCell8i accelerator
- **”nehalem”** (various systems)
  2.53 GHz > 10.12 Gflops per Intel Nehalem EP core
- **“uchu”** (NVIDIA Tesla S1070)
  78 Gflops (dp) per C1060 GPU

Mapping of languages and hardware

systems	mod2am	mod2as	mod2f
CAF	"louhi"	n.a.	"louhi"
CAPS	"uchu"	"uchu"	n.a.
CellSs	"cell-cluster"	"cell-cluster"	"cell-cluster"
Chapel	"louhi"	"louhi"	n.a.
Cn	"clearspeed"	"clearspeed"	"clearspeed"
CUDA	"uchu"	"uchu"	"uchu"
CUDA+MPI	"uchu"	"uchu"	n.a.
FPGA-VHDL	"maxwell"	n.a.	n.a.
FPGA-HCE	"maxwell"	n.a.	n.a.
MKL	"nehalem"	"nehalem"	"nehalem"
MPI+OpenMP	"nehalem"	"nehalem"	n.a.
OpenCL	"uchu"	n.a.	n.a.
RapidMind	"uchu"	"uchu"	n.a.
UPC	"itanium"	"itanium"	"itanium"
X10	"huygens"	"huygens"	n.a.

Porting table

porting status	mod2am	mod2as	mod2f
CAF	successful	not started (compiler bugs)	successful
CAPS	successful	successful	running out of time
CellSs	successful	partly	partly
Chapel	partly	partly	sdk example used
Cn	successful	successful	successful
CUDA	successful	successful	successful
CUDA+MPI	partly	partly	running out of time
FPGA-VHDL	partly	running out of time	running out of time
FPGA-HCE	partly	running out of time	running out of time
MPI+OpenMP	successful	successful	partly
OpenCL	successful	buggy	running out of time
RapidMind	successful	successful	running out of time
UPC	successful	successful	successful
X10	successful	successful	running out of time

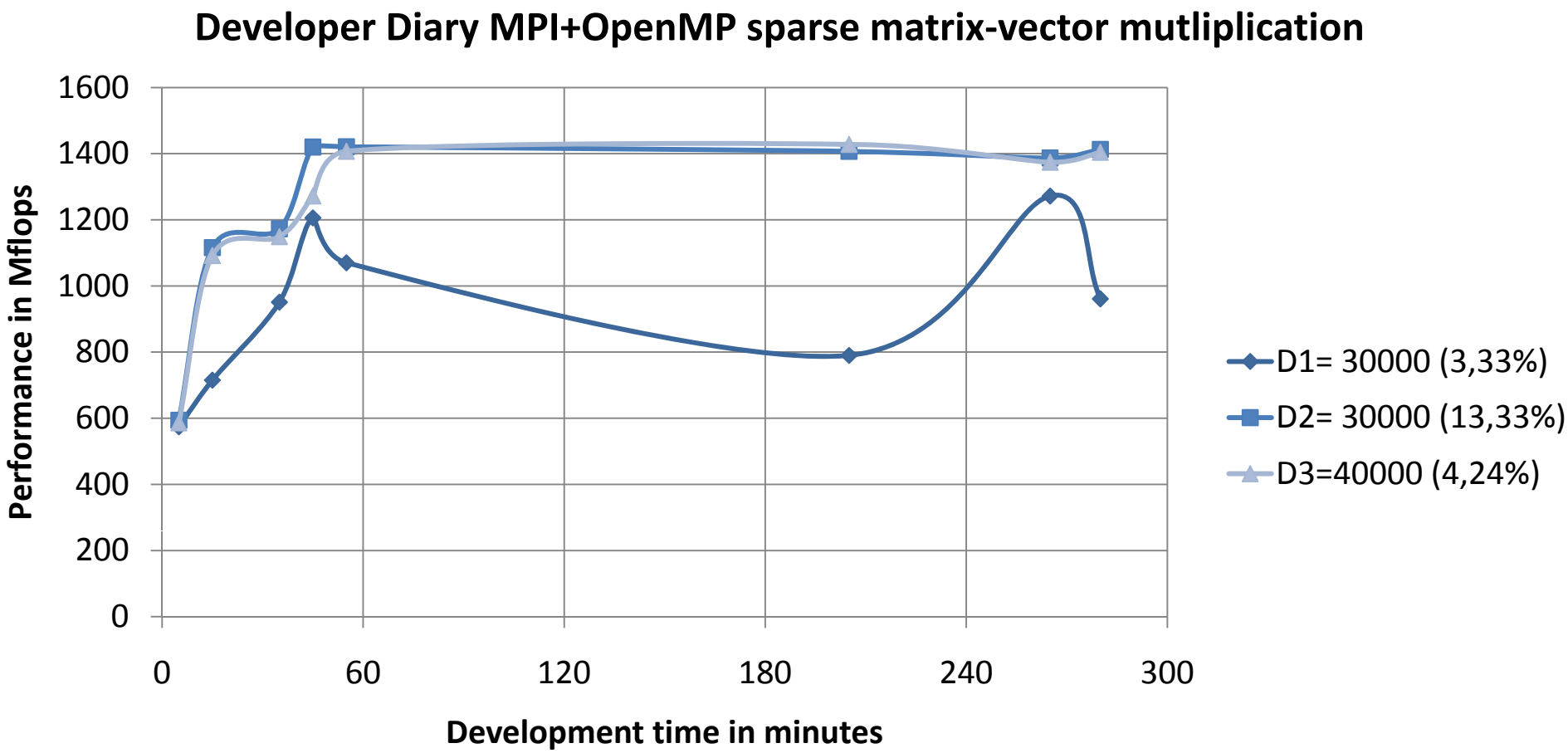
A snapshot of the current HPC languages

PRODUCTIVITY RESULTS

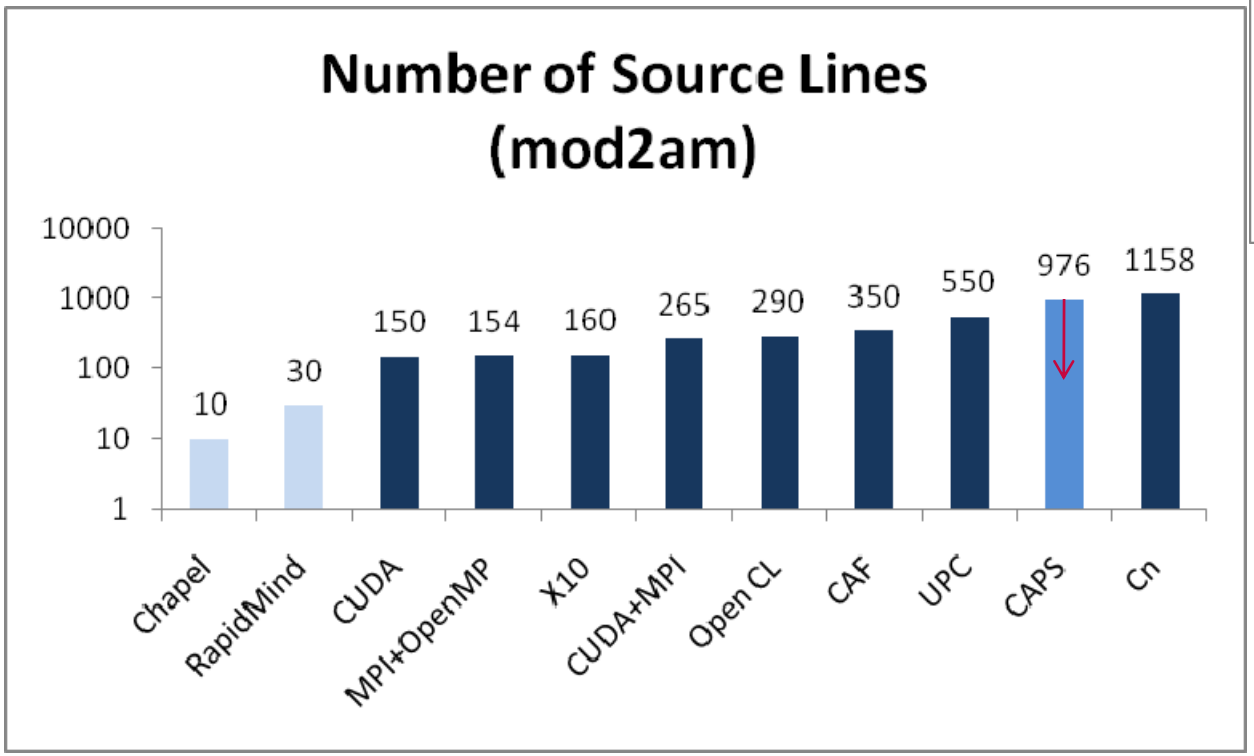
Developer diary example (1/2)

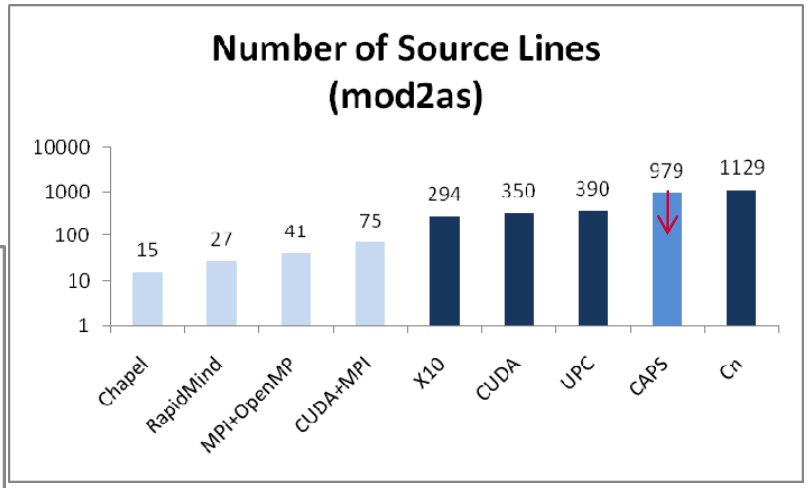
Dev. Time	# Cores	Comments
5m	1	Original C kernel
10m	4	Basic parallelization using OpenMP.
20m	4	Rewriting of the main loop. This optimizations works only with PGCC compiler ICC is not able to perform vectorization on the internal loop.
20m	4	Full 0-index CSR, only OpenMP parallelization. Refer to MKL library user guide to understand the differences between CSR mode and the 1-index.
30m	1	Matrix-vector multiplication with Sparse BLAS library, not multi-threaded.
2h	4	OpenMP parallelization with Sparse BLAS from INTEL MKL 0.2 using 1-index CSR format. MKL supports multi-threading.
1h	4 MPI + 4 OMP (16)	Trivial block-striped partitioning among all processors: sparse matrix is distributed but the dense vector is replicated on all processors.
15m	4 MPI + 4 OMP (16)	Same MPI strategy of v1.1 but this version uses INTEL MKL library 10.2 for local computation.

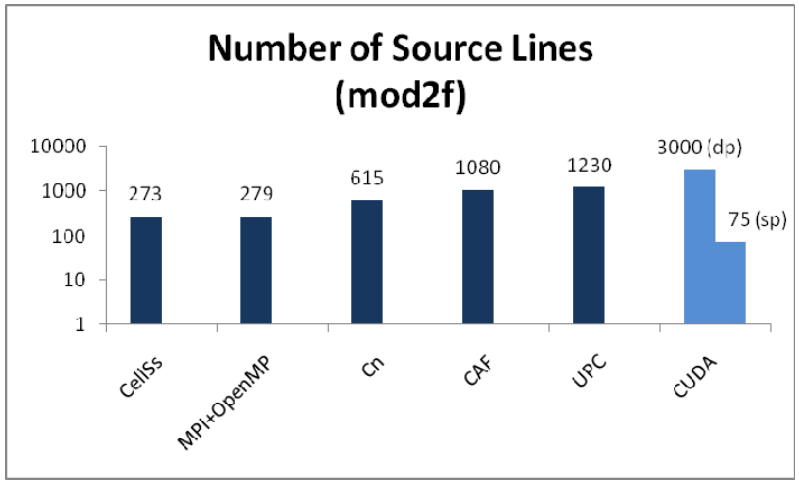
Developer diary example (2/2)



Source lines (as reported)

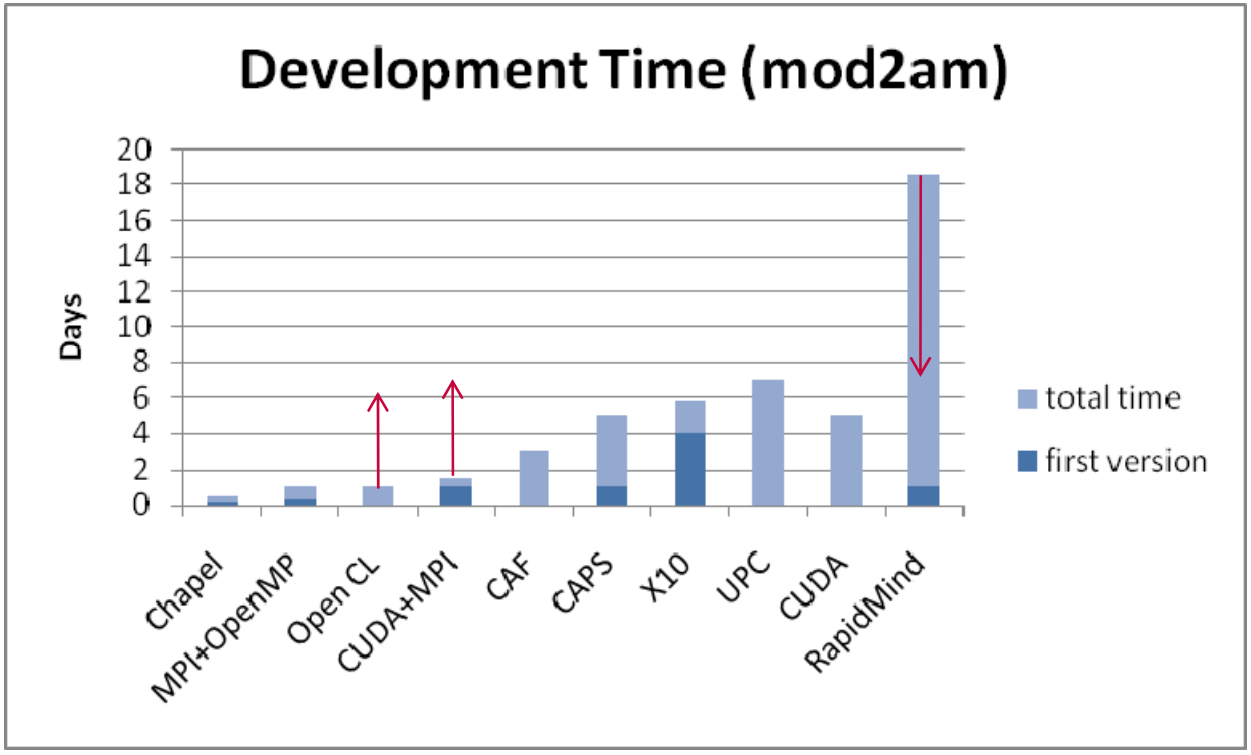


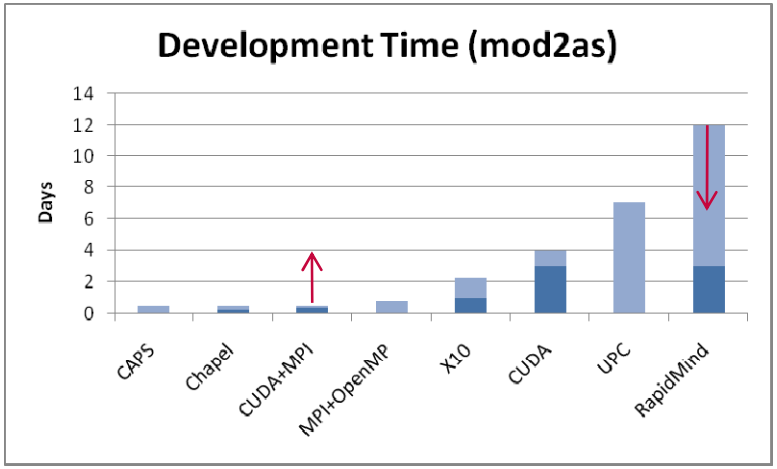


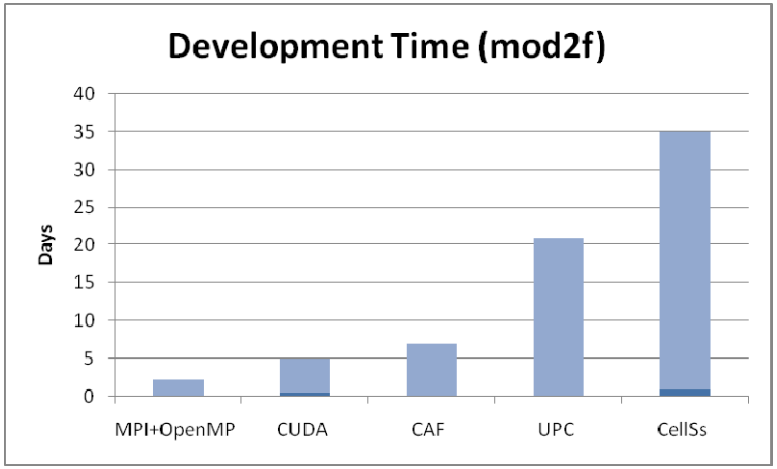


* The CAPS developer reported accumulated source lines for different versions.

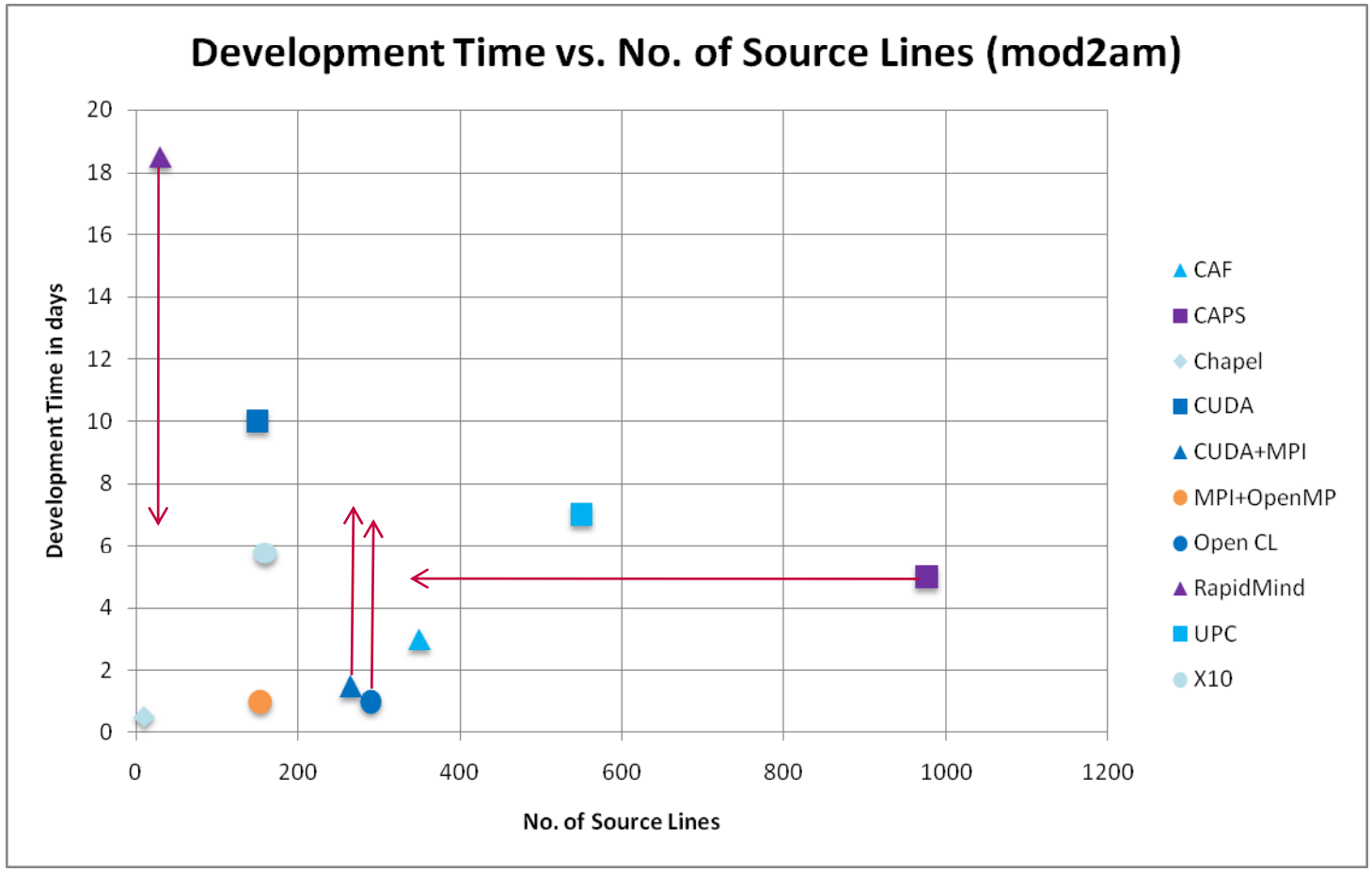
Development time (as reported)

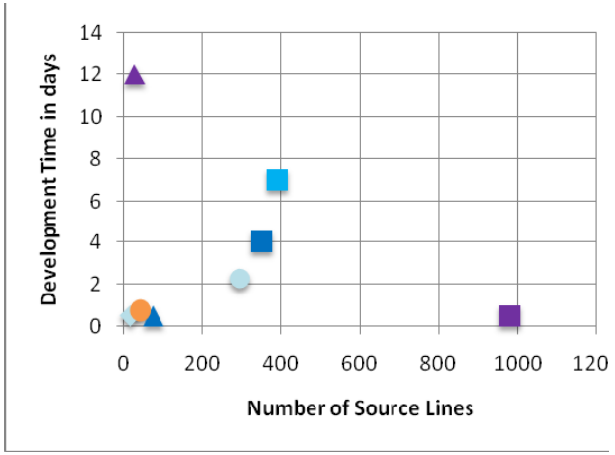


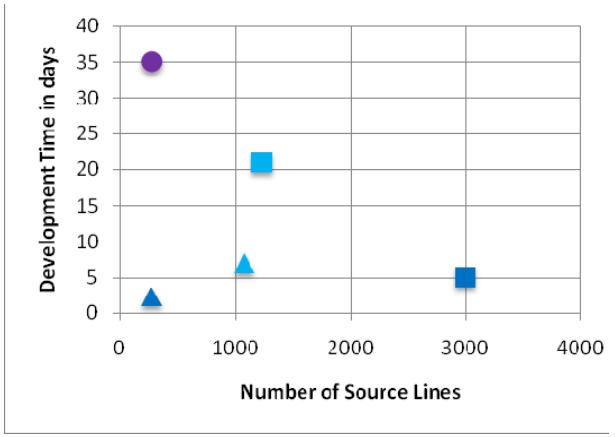




*Both OpenCL and CUDA+MPI have been based on the CUDA ports.
The RapidMind developer reported additional benchmarking time.





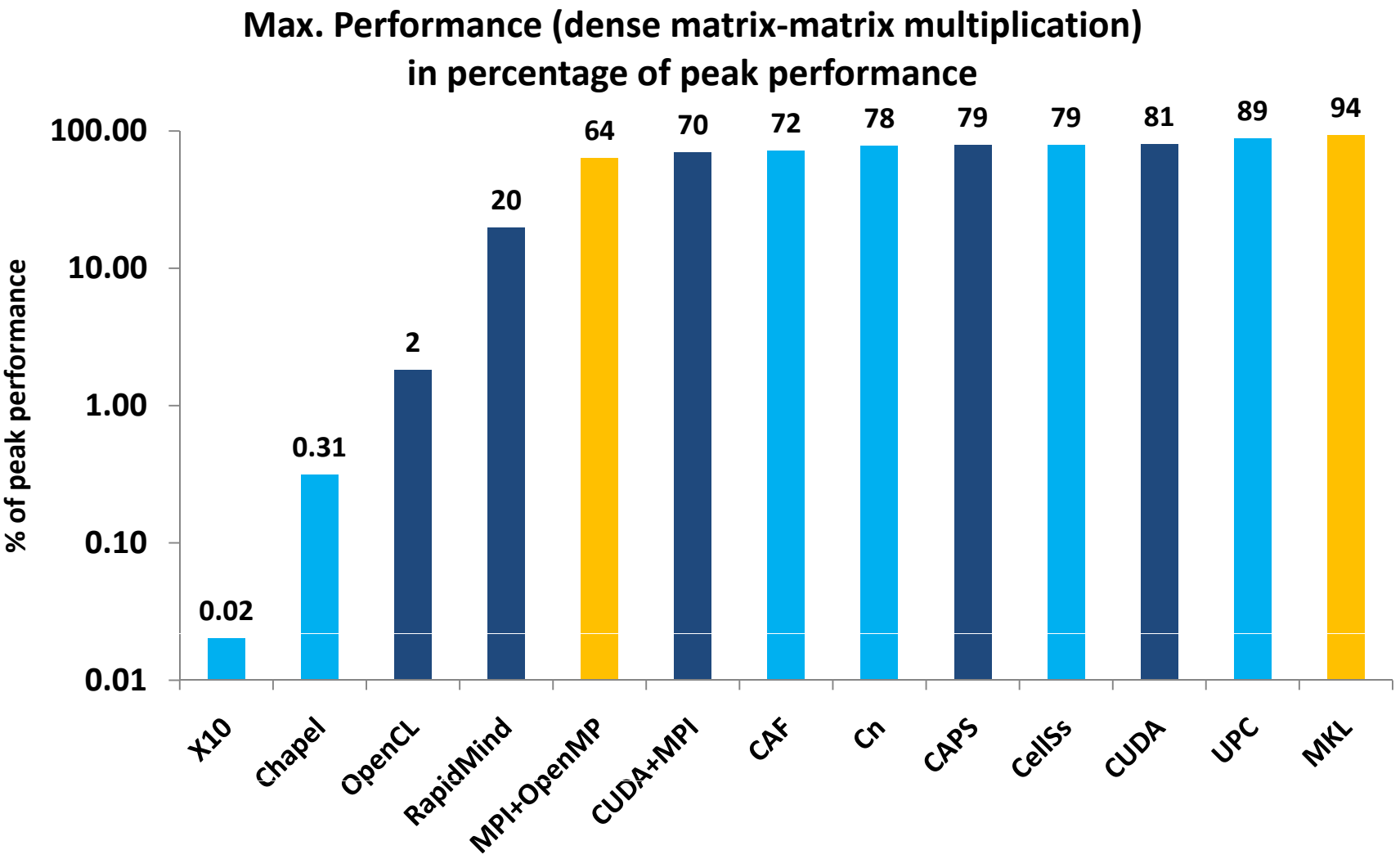


*Both OpenCL and CUDA+MPI have been based on the CUDA ports.
The RapidMind developer reported additional benchmarking time.
The CAPS developer reported accumulated source lines for different versions.

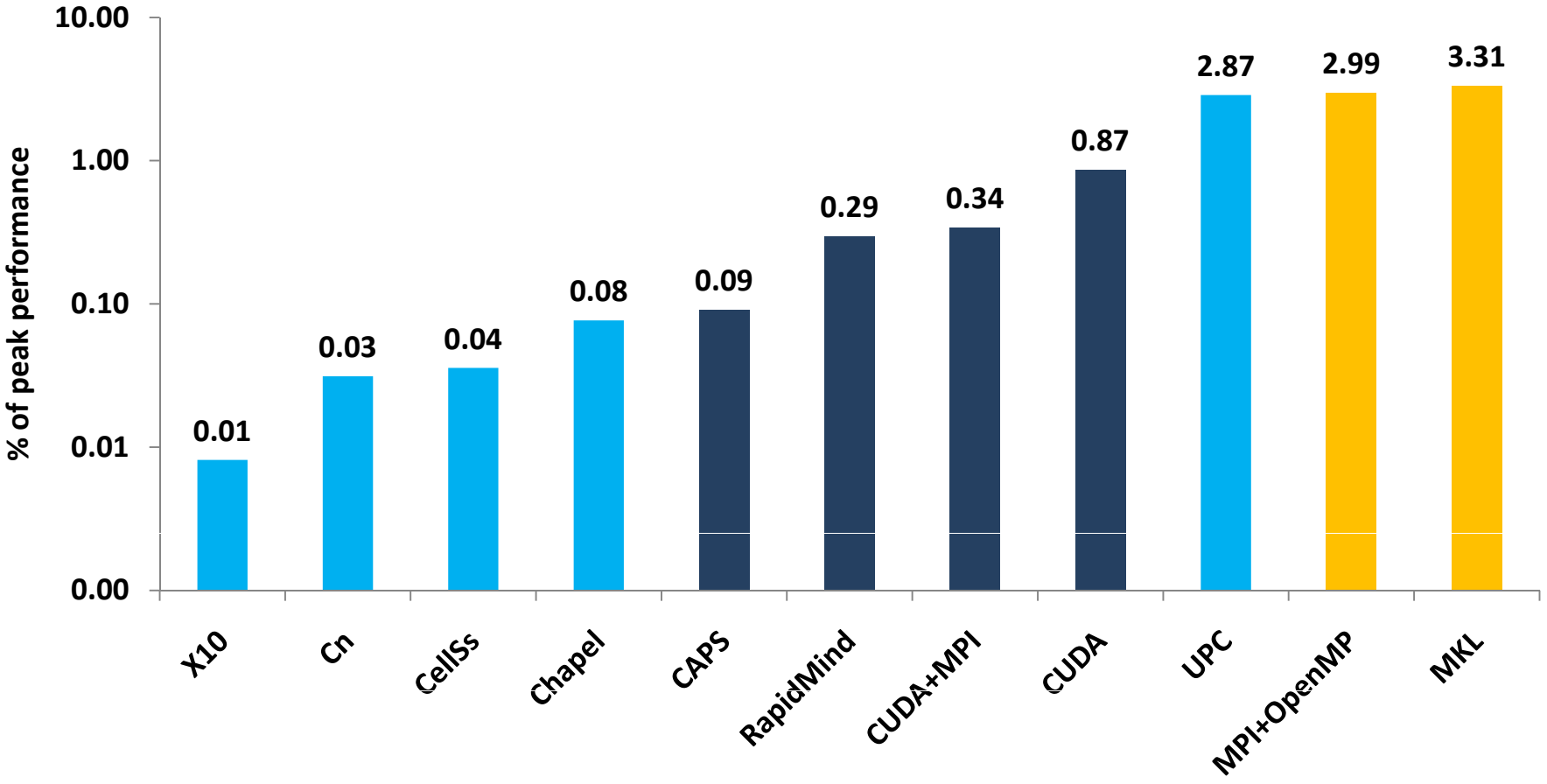
Comparing apples with oranges.

PERFORMANCE RESULTS

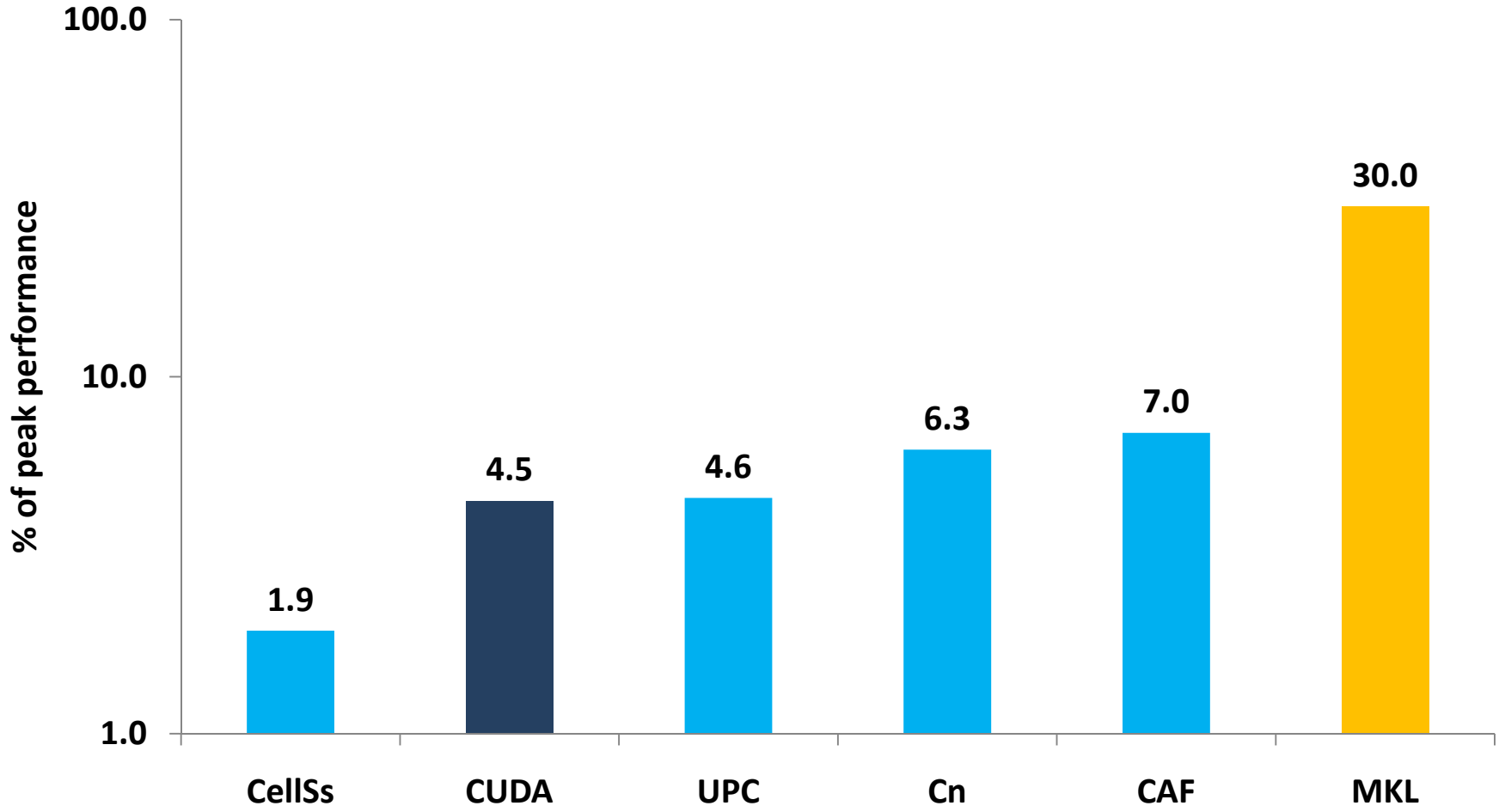
max perf in % of peak perf	**System**	**measured on**	**mod2am**	**mod2as**	**mod2f**
CAF	"louhi"	4 cores	72	n.a.	7
CAPS	"uchu"	1 C1060	79	0,09	n.a.
CellSs	"maricell"	1 PowerXCell8i	79	0,04	2
Chapel	"louhi"	1 core	0,31	0,08	n.a.
Cn	"clearspeed"	1 CSX700	78	0,03	6
CUDA	"uchu"	1 C1060	81	0,87	4
CUDA+MPI	"uchu"	1 C1060	70	0,34	n.a.
FPGA-VHDL (raw)	"maxwell"	1 FPGA	1625 Mflops	n.a.	n.a.
FPGA-HCE (raw)	"maxwell"	1 FPGA	14 Mflops	n.a.	n.a.
MKL	"nehalem"	8 cores	94	3,31	30
MPI+OpenMP	"nehalem"	4x4 cores	64	2,99	n.a.
OpenCL	"uchu"	1 C1060	2	n.a.	n.a.
RapidMind	"uchu"	1 C1060	20	0,29	n.a.
UPC	"itanium"	4 cores	89	2,87	5
X10	"huygens"	1 core	0,02	0,01	n.a.

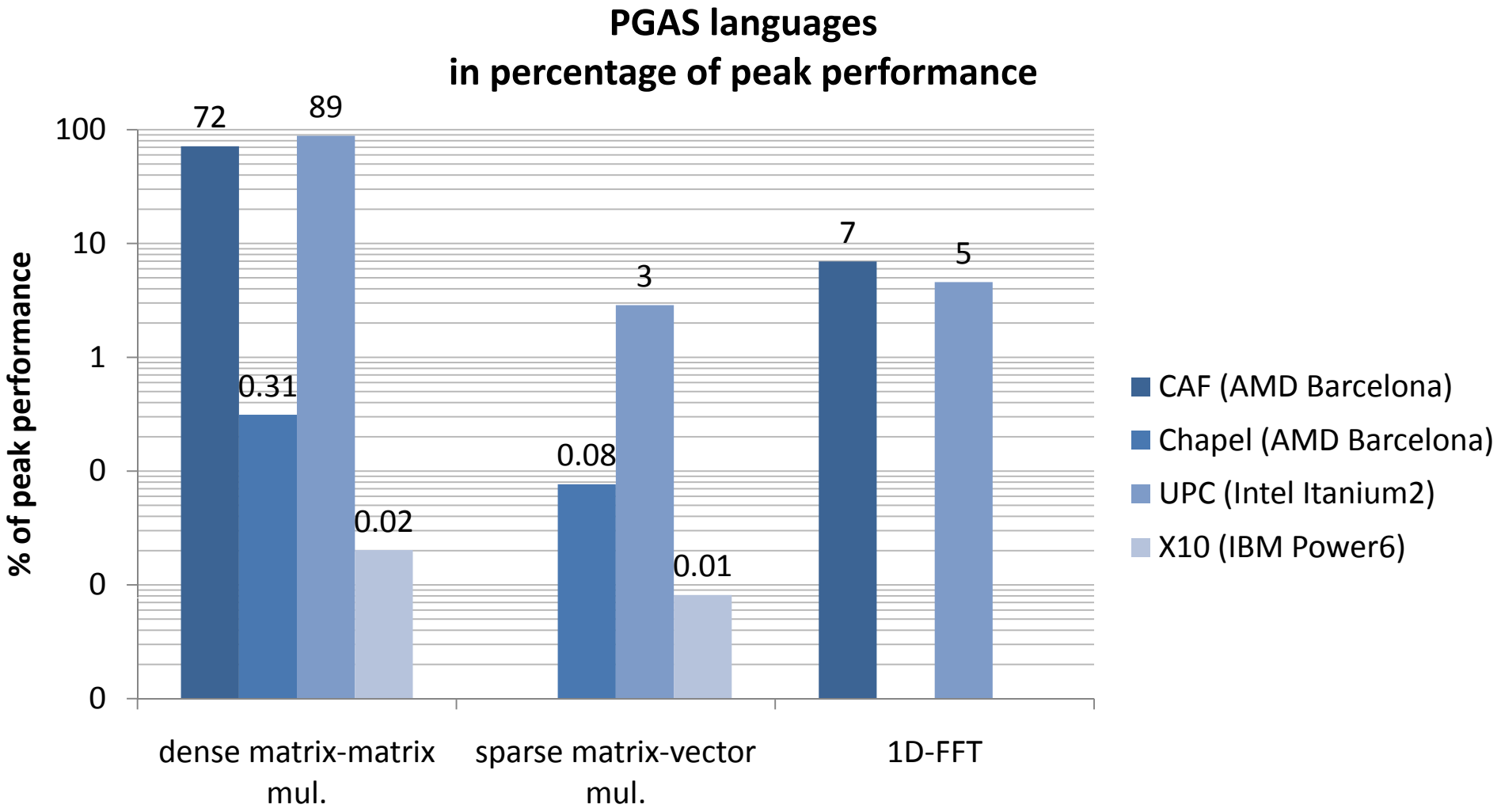

Max. Performance (dense matrix-matrix multiplication)
in percentage of peak performance
% of peak performance
100.00
10.00
1.00
0.10
0.01
0.02
0.31
2
20
64
70
72
78
79
79
81
89
94
X10
Chapel
OpenCL
RapidMind
MPI+OpenMP
CUDA+MPI
CAF
Cn
CAPS
CellSs
CUDA
UPC
MKL

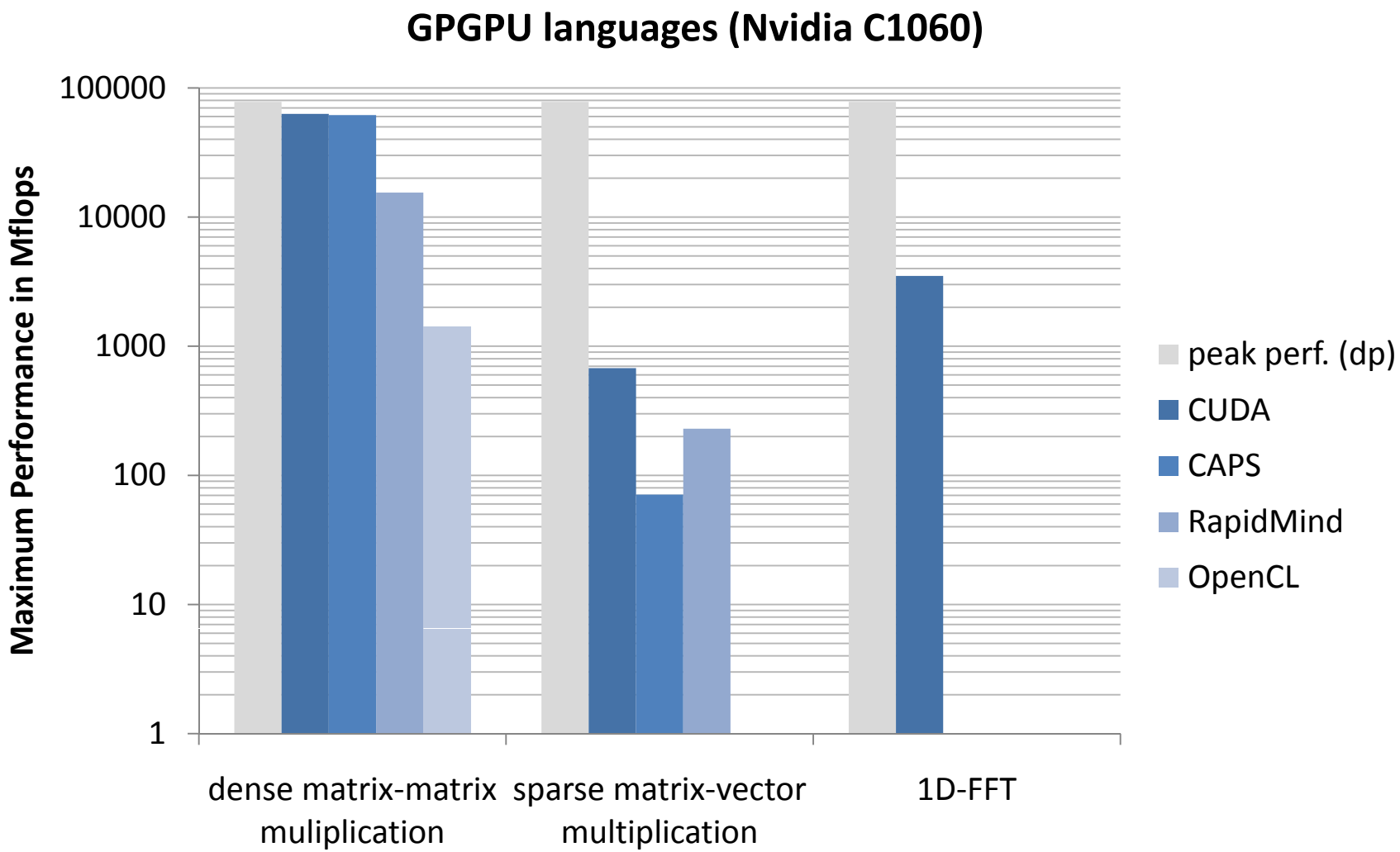
Max. Performance (sparse matrix-vector multiplication) in percentage of peak performance



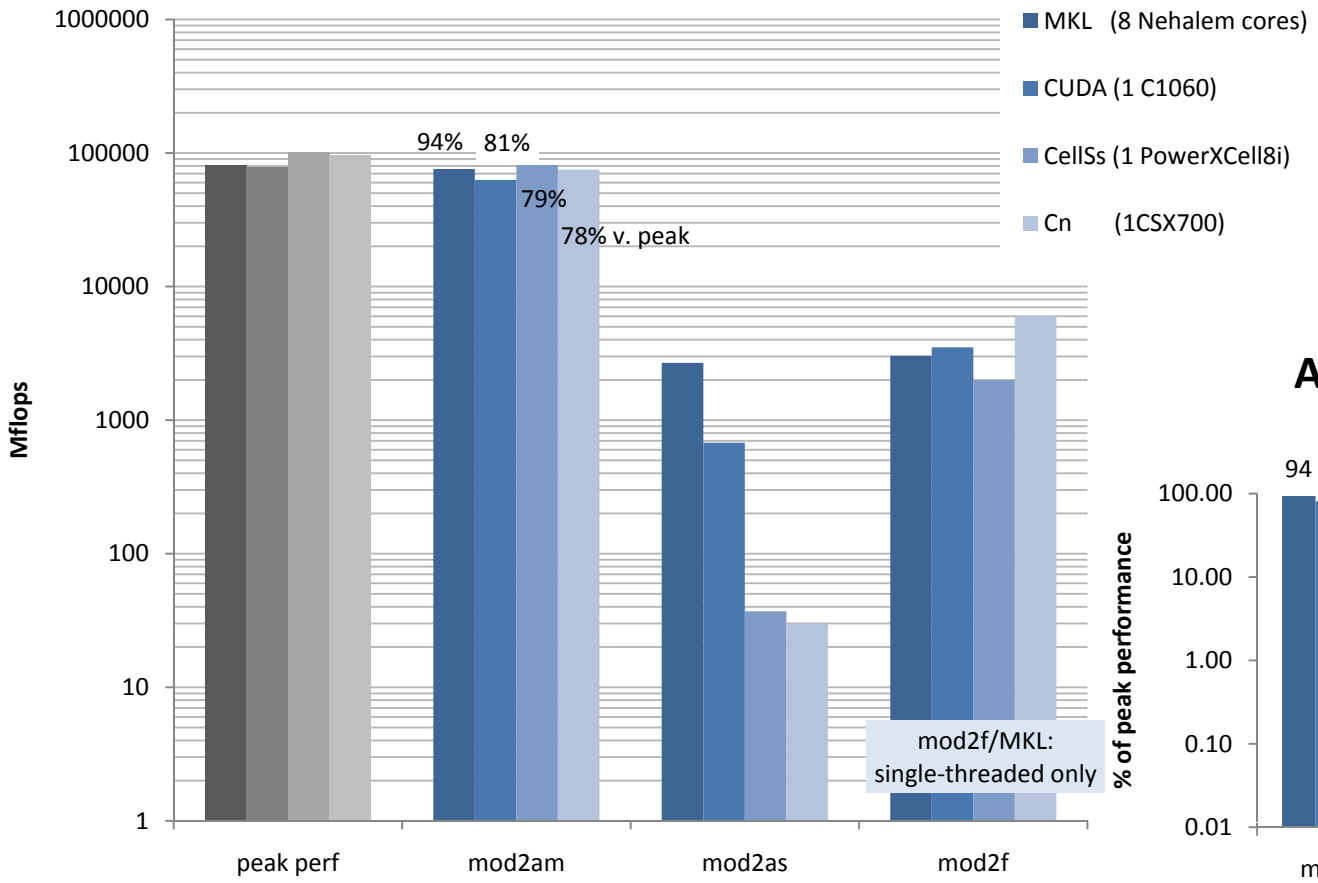
Max. Performance (1D-FFT) in percentage of peak performance



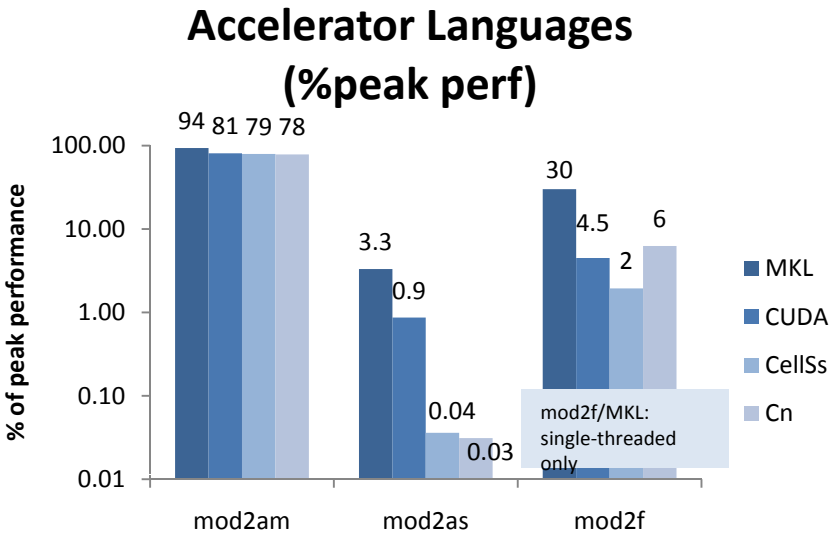

PGAS languages
in percentage of peak performance
% of peak performance
100
10
1
0
0
0
72
0.31
89
0.02
0.08
3
0.01
7
5
dense matrix-matrix mul.
sparse matrix-vector mul.
1D-FFT
CAF (AMD Barcelona)
Chapel (AMD Barcelona)
UPC (Intel Itanium2)
X10 (IBM Power6)


GPGPU languages (Nvidia C1060)
Maximum Performance in Mflops
100000
10000
1000
100
10
1
dense matrix-matrix muliplication
sparse matrix-vector multiplication
1D-FFT
peak perf. (dp)
CUDA
CAPS
RapidMind
OpenCL

Accelerator Languages (absolute performance)



Accelerator Languages (%peak perf)



For further information

See **PRACE Deliverable D6.6**

(to be published at http://www.prace-project.eu/documents/public-deliverables-1/)

Contact:

Iris Christadler (christadler@lrz.de), LRZ, Germany

Carlo Cavazzoni, CINECA, Italy

Giovanni Erbacci, CINECA, Italy

Filippo Spiga, CINECA, Italy

THANK YOU FOR YOUR ATTENTION!

Acknowledgements:
Guillaume Colin de Verdière, CEA (CAPS hmpp)
Maciej Cytowski , PSNC (CellSs)
Jose Gracia, HLRS (Chapel)
Hans Hacker, LRZ (CUDA, MKL)
Olli-Pekka Letho, CSC (OpenCL, CUDA+MPI)
Walter Lioen, SARA (X10)
James Perry, EPCC (VHDL, HCE)
Sami Saarinen, CSC (UPC, CAF)
Ramnath Sai Sagar, BSC (CellSs)
Filippo Spiga, CINECA (OpenMP+MPI)
Aad van der Steen, NCF (C, Fortran, MPI)
Volker Weinberg, LRZ (Rapidmind)